National Technical University of Athens
School of Electrical and Computer Engineering
Division of Computer Science

Communication performance prediction on large-scale systems

Ph.D. Thesis

Nikela Papadopoulou

Electrical and Computer Engineer, Dipl.-Ing.

Athens, Greece
July, 2017

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Computer Science

Communication performance prediction on large-scale systems

Ph.D. Thesis

Nikela Papadopoulou

Electrical and Computer Engineer, Dipl.-Ing.

Advisory Committee: Georgios Goumas
Nectarios Koziris
Panayiotis Tsanakas

Approved by the examining committee on July 20, 2017.

Georgios Goumas
Assistant Professor, NTUA

Nectarios Koziris
Professor, NTUA

Panayiotis Tsanakas
Professor, NTUA

Andreas-Georgios Stafylopatis
Professor, NTUA

Dimitrios Tsoumakos
Associate Professor,
Ionian University

Nikolaos Pleros
Assistant Professor, AUTH

Holger Fröning
Juniorprofessor,
RKU Heidelberg

Athens, Greece
July, 2017

Nikela Papadopoulou
Ph.D., National Technical University of Athens, Greece

This work has received funding from IKY fellowships of excellence for postgraduate studies in Greece - SIEMENS program.

To my parents

Abstract

On the path to exascale, supercomputers will grow to host hundreds of million of cores and various complex heterogeneous processing elements, yet even today, users fail to leverage the existing compute power of large-scale systems, as large classes of typical HPC applications are bound by non-scalable communication phases. The ability to predict the communication time of parallel applications can assist users, com- pilers, runtime systems and schedulers with decision-making for optimal resource utilization, performance optimizations, power saving and resilience.

This thesis presents a methodology for predictive communication modeling of HPC applications. Communication time depends on a complex set of parameters, relevant to the application, the system architecture, the runtime configuration and runtime conditions. To handle this complexity, we follow an empirical modeling approach. We define features that can be extracted from the application, the process mapping and the allocation shape ahead of execution, deploy a single benchmark to sweep over the parameter space and develop predictive models for communication time on three large-scale computing systems, Vilje, Piz Daint and ARIS, using different subsets of our features, statistical and machine-learning methods and training sets. We compare the predictive performance of our models on various communication patterns and applications, for multiple problem sizes, executions and runtime configurations, ranging from a few dozen to a few thousand cores. Our methodology is successful across all tested communication patterns on all systems and exhibits high prediction accuracy and goodness-of-fit. Our models are applicable just-in-time ahead of the execution of an HPC application, and, as we demonstrate in this thesis, their high accuracy make them suitable for communication-aware decision making, towards the optimization of resource utilization on large-scale systems.

Keywords: Performance Modeling, Predictive Modeling, Communication Time, HPC Applications, MPI, Supercomputers, Clusters, Statistical Learning, Machine Learning

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgement

The work presented in this thesis was conducted at the Computing Systems Laboratory of the School of Electrical and Computer Engineering at the National Technical University of Athens. It compiles the research and results of my five-year post-graduate studies in the School of Electrical and Computer Engineering at NTUA. With the completion of my studies, I would like to reflect on the people who have supported and helped me throughout this period.

First, I would like to thank my advisor, Assistant Professor Georgios Goumas, for his contribution in multiple levels. His confidence in me allowed for the beginning and completion of this course of studies. His first contribution was transferring his enthusiasm about his research interests. As my advisor, he has generously shared his knowledge, his research methodology and his unique approach to problem solving. He has guided, helped and supported my work and has always found a way to motivate me and encourage me towards overcoming any obstacle throughout my studies. The completion of this thesis would not have been possible without the effort and time he put into its supervision. I warmly thank him for everything.

I would like to express my sincere thanks to Professor Nectarios Koziris, member of my advising committee, for his confidence and support, his scientific guidance and his willingness to assist in any way towards the completion of this thesis. I would also like to thank Professor Panayiotis Tsanakas, member of my advising committee, for his help throughout my studies. I thank Professor Andreas-Georgios Stafylopatis, Associate Professor Dimitrios Tsoumakos, Assistant Professor Nikolaos Pleros and Associate Professor Holger Fröning, for participating in my thesis examination committee.

Throughout my post-graduate studies, I spent most of my time at the Computing Systems Laboratory. My daily co-existence with all the members of the lab has contributed decisively to my shaping as a researcher. I first thank Dr.

Kostis Nikas, for his willingness to discuss any matter and assist with any problem, research-related or other, even when it was trivial. Therefore, I also thank him for his patience. I thank Dr. Anastassios Nanos, for the enthusiasm with which he shares his knowledge and interests and his motivational and collaborative spirit. I also need to thank him for his extreme support. I want to equally thank current and former members of our research group, Dimitris Siakavaras, Athena Elafrou, Stefanos Gerangelos, Chloe Alverti, Kostis Papazafeiropoulos, Christina Giannoula, Dr. Vasilis Karakostas, Stratos Psomadakis, Alexandros Haritatos, Tasos Katsigiannis and Dr. Nikos Anastopoulos, as well as all other fellow PhD students, post-doc researchers and administrative staff of the laboratory, for their feedback, their assistance with little or big technical issues, our stimulating weekly discussions and, of course, for being such a pleasant company. Extra thanks to Vasilis and Chloe for finding the time to read and debug this thesis.

As a PhD student, I assisted with the supervision of diploma thesis in the Computing Systems Laboratory, which gave me the opportunity to study research topics related with my thesis topic and explore alternative research directions, as well as the opportunity to collaborate with excellent students and interesting people. I would like to especially thank Sotiris Apostolakis, for his excellent work and his contribution to the early steps of this work. I also want to thank Panagiotis Moullotou, Foivos Kotomatas, Georgios Christodoulis and Georgios Zachariadis.

From April to August 2016, I worked as a research aide at the Argonne National Laboratory in Chicago, U.S.A., with the Programming Models and Runtime Systems group. I would like to express my gratitude to the group leader, Dr. Pavan Balaji, for giving me the opportunity to join his group and work in a highly stimulating research environment. During these few months, I gained significant experience from his immense knowledge and his way of tackling research problems. I would like to equally thank Dr. Lena Oden, who mentored my research at Argonne National Laboratory and offered her guidance and time without any hesitation. My collaboration with her has been an unexpectedly enjoyable and memorable experience.

The research presented in this thesis focuses on large-scale computing systems and demanded access to such systems. I especially thank NTNU, CSCS, and GRNET for granting us access to their supercomputers and I also thank all the people who assisted with reaching out to these organizations.

Finally, I would like to thank my closest friends, for being around to offer their moral and emotional support. Above all, I thank my parents, Aris and Anthoula, for their unconditional love, support, and encouragement, not only during the last five years but during the last twenty-eight years. I dedicate this thesis to them.

Concluding, I would like to mention that the research presented in this thesis was conducted in an open, public university, which, despite the stranglehold of research funding in these past years of the crisis, continues its output of high-quality research results and does not interfere with the choice of research directions. In the context of recent years, I find it important to thank all those who have struggled to keep universities public and open to everyone. I hope and wish that the public character of education and research will strengthen in the years to come and that the effort and contributions of all people who work in these institutions, even when conditions are adverse, will be acknowledged.

1

Introduction

1.1 Motivation

Supercomputing first emerged in the 1960s, as a response to the need of the scientific community to solve problems which by far exceeded the computational capability of a single processor. In the early 1990s, large-scale systems of thousands of processors appeared. Since then, the computing power of supercomputers doubles every 1.5 years, hand-to-hand with an increase in the number of processors [Council et al., 2005]. In mid-2016, the Chinese Sunway TaihuLight supercomputer was built, comprising more than 10 million cores and achieving more than 90 PetaFlops of performance (90×10^{15} floating point operations per second) [top]. The next stop in the rapid evolution of supercomputers is the exascale era; supercomputers are projected to hit ExaFlop performance (10^{18} floating point operations per second) by 2022.

The move to exascale comes with many diverse challenges. The need to constrain power consumption at 20MW per ExaFlop and capital costs at 200 million dollars [Shalf et al., 2010; McMorrow, 2013] is driving a dramatic change in system architecture and software. Next generation compute nodes will comprise hundreds of processors and many more compute threads, as well as general purpose and specialized accelerators, in an effort to draw performance out of parallelism and heterogeneity, as core frequency can no longer increase and the per node power consumption needs to be constrained [Kogge and Shalf, 2013]. The memory per node will increase, yet only by a small fraction of the increase in Flops, becuase of budget constraints and the slow increase of memory density [McMorrow, 2013]. Node bandwidth will only moderately increase as well, because of energy consumption constraints. Eventually, next generation compute nodes will attach to deep memory hierarchies, i.e., caches, 3D-stacked memory and non-volatile RAM, to moderate power consumption

[Shalf et al., 2010]. Still, despite the changes in node architecture, node performance is expected to increase only by a factor of 10 to 100.

To reach exascale performance, supercomputers will need to host thousands of nodes, interconnected with high-end off-chip interconnection networks, of high performance, low latency and high bandwidth. The interconnection networks need to meet the communication demands of node (hundreds of threads) and system (thousand of nodes) concurrency. The power/performance constraints for interconnection networks are encouraging a shift in technology, towards silicon photonics. Additionaly, network components are being redesigned, so that the hardware can meet the communication demands of applications. However, the cost of communication for millions of threads can be overwhelming, and the interconnection network topologies are also critical to achieve high performance. As such, low-diameter topologies that can provide high bisection bandwidth and are partitionable will dominate future systems [Shalf et al., 2010; Jain et al., 2017]. System software is also rapidly evolving to adapt to the new hardware. Operating systems, network interfaces and communication software, compilers, programming models, resource managers, application libraries and tools, that constitute the rich software ecosystem of supercomputers, are being re-designed to meet exascale demands. One of these demands is resilience to failures, where the software will be assisting the hardware. Overall, the re-design of hardware for exascale focuses on tackling power consumption, resilience and data movement. The latter is particularly important for performance. Eventually, ExaFlop performance will emanate from the ability of parallel applications to scale up to the hundreds of threads and scale out to the thousands of nodes of the new supercomputing systems.

The parallel applications which execute on supercomputers are typically simulations of the real world and consist of multiple computational kernels, which exhibit various phases of computation and communication, as well as file I/O for the instantiation of the simulation or for checkpointing/restarting the application. Their parallel performance strongly depends on the ability of all the phases to scale efficiently. Computation phases are expected to scale efficiently enough on new large-scale systems, exploiting the computational capabilities of simple cores, vector units, heterogeneous accelerators and specialized hardware. However, to achieve ExaFlop performance, parallel applications will execute over millions of computational threads, which will require communication both within the node as well as with remote nodes, over the interconnection networks. Subsequently, communication phases will be burdened with more message exchange and/or greater data volumes. Despite the technology evolution regarding interconnection networks, communication cost can rise enough to outweigh computation scalability, thus rendering communication a

serious obstacle towards exascale performance. This reflects one of the three fundamental concerns for exascale performance, which is data movement, i.e. communication of data both within a node and between nodes.

Predictions of communication time of parallel applications can support and enhance various decision-making scenarios. Supercomputer users would be able to predict the scalability of their applications and allocate the optimal number and combination of resources, avoiding the dissipation of valuable resources and long waiting times. In addition, the ability to select between appropriate resource allocations, i.e., number of nodes, cores, even networking components, is critical to the optimization of performance [Goumas et al., 2009b], power consumption [Brooks et al., 2000; Bekas and Curioni, 2010], and fault-resilience [Li et al., 2014; Yu et al., 2014]. For example, a prediction for communication time can contribute in selecting optimal combinations of computational resources, with respect to energy and/or fault-resilience and enhance resource allocation policies and scheduling policies on the nodes of a supercomputer. Moreover, a communication prediction model can be used to support code optimization and tuning for a wide range of communication optimization techniques, like computation/communication overlapping [Goumas et al., 2009a], message compression [Filgueira et al., 2011], communication avoiding [Solomonik and Demmel, 2011] and hybrid MPI/OpenMP implementations [Drosinos and Koziris, 2004]. In fact, an effective communication model can be used at runtime, for the auto-tuning of applications, enabling and disabling various optimizations. Finally, communication performance models constitute a core component in trace-driven simulators for parallel applications, such as SimGrid [Casanova et al., 2015] and SST/Macro [sst].

Communication performance is dependent on a multitude of factors, and as such, its accurate modeling poses significant challenges. The communication time for a specific communication pattern on a specific system heavily depends on the traffic pattern, produced by the mapping of the application communication graph on the system allocation, dedicated to the execution of the application, the system architecture and its network topology properties. The traffic created by a single application can be enough to induce contention on network components and congestion on network links, with equivalent delays. However, on large-scale systems, where multiple applications co-execute, they compete for shared resources, interfere and are able to cause unexpected delays to the communication time of each other. The setup of each system, its communication software, operating system and resource management components influence communication time of applications in a complex manner. Also, computation phases of applications affect communication time, as they can overlap with communication or skew communication time due to load imbalance. Therefore, communication time prediction is intricate, as it requires

the encapsulation of as many as possible of the various and complex factors that affect communication time, in a single model, in a meaningful way, that will allow for predictions ahead of execution.

In this thesis, we present a methodology for predictive modeling of communication on HPC applications. The motivation for this work is the behavior of large classes of applications, which exploit the increase in the performance of large-scale systems for strong scaling. Such classes include partial differential equations (PDEs) on structured and unstructured grids, unstructured graph processing algorithms, multi-scale or multi-model codes and more. Applications of these classes are expected to retain their communication costs on next generation supercomputers, as data movement costs will be increased on the enormous numbers of threads, and as long as the message passing model remains the prevalent programming model [Kogge et al., 2008]. In addition, applications of these classes that do not scale linearly will retain their non-linear, suboptimal scaling behavior on systems of higher performance, due to communication costs. Consequently, the ability to predict communication time of parallel applications before their execution is imperative for the prediction of their expected performance and scalability and their optimization, as well as for the efficient utilization and resource management on next-generation systems.

Our approach aims at predicting the time for an entire communication phase of an application and offers predictions before the execution of the application. To our knowledge, this is the first approach to provide communication modeling that predicts the communication of a wide set of point-to-point communication patterns without any customization, ahead of execution, delivering significant prediction accuracy, as indicated by our experimental results on three large-scale systems. To accomplish this, we consider features that affect communication performance taken from the application characteristics, its mapping on the target system and the configuration of the underlying system architecture. We construct a simple benchmark to create a generic training set for use across all tested point-to-point communication patterns and deploy statistical-learning and machine-learning methods to construct a number of predictive models for communication time, that expose different levels of accuracy and applicability. We apply our methodology on two supercomputers and one large-scale cluster, with state-of-art interconnection networks: i) Vilje, an InfiniBand SGI Altix system, ii) Piz Daint, a Cray XC30 system and iii)ARIS, an IBM NeXtScale cluster with InfiniBand. Throughout our evaluation, we validate two significant assumptions that have driven our work. First, generic, application-agnostic benchmarking, which makes our methodology applicable to multiple applications, is adequate for training an accurate model on all platforms. Second, a predictive communication model needs to consider

multiple features that well describe the traffic pattern, in order to incorporate the complex effects of communication over a large-scale system. We compare the predictive ability of our models against a baseline analytical model and semi-empirical models, following the methodology proposed by Ghavari et al. [Gahvari et al., 2011, 2014]. Our empirical approach outperforms any other approach in terms of prediction accuracy and provides accurate and meaningful predictions in scenarios of communication-aware decision-making.

1.2 Thesis contributions

The contributions of this thesis can be summarized as follows:

- We propose a generic methodology for the prediction of communication time of parallel applications on any large-scale computing system, using statistical [Papadopoulou et al., 2015] and machine-learning methods [Papadopoulou et al., 2017b].
- We apply the methodology for communication time prediction on three large-scale systems with diverse interconnection network architectures and topologies.
- Through the modeling process, we study and characterize communication and extract conclusions on the characteristics of the interconnection networks that affect communication performance.
- Through extensive evaluations, we demonstrate the accuracy and goodness-of-fit of our models on real-world parallel applications. Our models significantly improve the prediction accuracy, reducing relative prediction errors by more than 50%, in comparison with analytical and semi-empirical approaches proposed in literature.
- We demonstrate the utility and viability of predictions for communication time, through their utilization in decision-making scenarios regarding the optimization of the quality-of-service and overall throughput of large-scale systems [Papadopoulou et al., 2017a].

1.3 Thesis outline

The remainder of this thesis is organized as follows: Chapter 2 defines the problem and provides background details on its various aspects, i.e., large-scale systems, HPC applications, communication time and its modeling. The chapter also presents the problem statement and discusses related work. Chapter 3 presents our empirical methodology for constructing predictive communication models. In this chapter, we define the features we use for modeling,

our empirical benchmarking to collect a training set and the statistical and machine-learning methods used in this thesis for the construction of predictive models for communication time.

Chapter 4 presents the application of our statistical-learning and machine-learning methodology on Vilje, an enhanced hypercube InfiniBand supercomputer, along with the extensive evaluation of our predictions on this system and the application of our models in communication-aware decision making. Chapter 5 presents the application of our statistical-learning and machine-learning methodology on Piz Daint, a Cray XC30 dragonfly supercomputer, the evaluation of all our constructed models on the prediction of communication time, as well as the application of our predictive models in decision-making scenarios for the optimization of the system utilization. Chapter 6 presents the application of our machine-learning methodology on ARIS, an InfiniBand fat-tree cluster, along with the evaluation of our predictive models on real-world communication patterns and the application of our models in decision-making tasks for the optimization of the system utilization. We omit the application of our statistical learning methodology on ARIS, as this system entered its production phase in June 2015, almost three years after the inception of this thesis. Chapter 7 concludes this thesis and discusses future directions.

2

Background and problem definition

2.1 Introduction

In this thesis, we present a methodology for constructing predictive models for the communication time of HPC applications running on large-scale clusters and supercomputers. Communication performance of parallel applications on large-scale systems depends on the mapping of the application communication graph on the underlying system. Large-scale clusters and supercomputers are built with various node and interconnection network architectures, network topologies, operating systems, resource management software, middleware, constituting a complex entity. Moreover, parallel applications enclose multiple computation and communication phases with particular characteristics. Their mapping on a large-scale system creates a specific traffic pattern, which may be unique to every execution. Therefore, the prediction of communication time is an intricate and challenging task, as it requires the apprehension of the application, the system and their interaction and the formulation of this interaction in a quantifiable way. In the following sections, we provide definitions and background details for large-scale systems, HPC applications, communication time and performance models for communication, as they are discussed in the scope of this thesis, as well as any assumptions we make to solve the problem of predicting communication performance. In addition, we present related work on models for communication performance and outline their properties, virtues and pitfalls.

2.2 Large-scale HPC systems

This work focuses on large-scale high-performance computing systems, i.e., large-scale clusters and supercomputers. Such systems come as installations of

hundreds or thousands of compute nodes, interconnected with high-performance interconnection network. They also include a deep software stack, from the operating system to application libraries. In the following paragraphs, we outline the hardware and software architecture, as well as details on the programming models supported by large-scale systems.

2.2.1 Node architecture

Compute nodes of supercomputers consist of one or more chip multiprocessors (also known as symmetric multi-processors or SMPs), with shared address space, where each processor has its own cache memory hierarchy. If more than one processors co-exist on the same node, they are interconnected with point-to-point processor interconnects. If a node hosts more than one memory nodes, the result is the NUMA (non-uniform memory access) architecture, where the access time to a memory address depends on the position of the core on the node. The NUMA architecture introduces asymmetries in the latency of data movement within the node. Additionally, compute nodes often host coprocessors, such as Intel Xeon Phi, or accelerators, as GPUs, connected over the PCI/PCIe. However on current systems, these components are used as offload engines and do not directly communicate with each other. The nodes are connected over the PCI/PCIe with the interconnection network adapters. We should note that, on NUMA architectures, all devices are connected on one of the processors over the PCI/PCIe, bringing on an additional element of asymmetry on the access times from cores to devices within the same node. Therefore, besides intra-node communication latencies, inter-node communication latencies can also vary with the distance of the core from the communication device [Hager et al., 2009].

2.2.2 Interconnection network architecture

Large-scale systems use high-end interconnection networks to connect the thousands of compute nodes. According to the latest Top500 list [top], the most popular interconnection network technologies are Ethernet[Metcalfe and Boggs, 1976], InfiniBand [Association et al., 2001], Cray Aries [Alverson et al., 2012], and the newer Intel OmniPath [Birrittella et al., 2015]. Each network's technology defines its latency and bandwidth, as well as other critical parameters, such as the size and buffering capacity of switches and routing mechanisms. The technologies vary in the architecture of networking adapters and switches, as well as the network interface. Current interconnection networks deploy interfaces that can offload communication operations from the CPU to the network, towards decoupling communication from the CPU and allowing for computa-

tion/communication overlapping [Di Girolamo et al., 2015]. Also, most vendors work towards full support of one-sided and collective (many-to-many) communication operations on hardware, so as to minimize the costs of these operations in time and power and allow for overlapping [Schneider et al., 2013].

Each network technology is usually compatible with specific network topologies. The network topology determines the number of network interface cards, switches, routers and links that build the system, depending on the system scale, as well as the diameter, path diversity and number of nodes per switch or router. Alongside with the technology and its routing protocol, the topology also determines the network's bisection and global bandwidth and the hop count for different paths. Topologies can be direct or indirect. On direct topologies, as are k-ary n-cubes and the dragonfly topology, each switch is terminal and connects to at least one compute node. On indirect topologies, as are tree-like topologies, non-terminal switches also exist [Dally and Towles, 2004].

The share of topologies per technology among the world's fastest supercomputers is as following: Ethernet comes in hierarchical tree-like topologies, InfiniBand comes mostly in fat-tree topologies, as well as in the hypercube topology, Cray Aries comes in the Dragonfly topology, Cray Gemini comes in a 3D-torus topology, IBM BlueGene/Q comes in a 5D-torus topology, Intel OmniPath comes in fat-tree topologies (but will also enable the Dragonfly topology) and Fujitsu Tofu comes in a 6D-torus topology. Among those topologies, the fat-tree and dragonfly topologies offer low diameter, therefore low latency, due to their high radix routers, and high bisection bandwidth. However, fat trees go with high wiring costs and power consumption at large scale. Besides, tori, which are also widely used, go with low-radix routers and result in higher diameters, however they offer high bisection bandwidth in high dimensions. Exascale systems will embrace fat-tree, Dragonfly and high-dimension tori topologies, depending on their power and budget constraints and the applications demands for low latency via low diameter, high throughput via high bisection bandwidth and avoidance of interference with other applications via the partitionability of the topology [Jain et al., 2016, 2017].

2.2.3 Software stack

Efficient supercomputer operation relies on a deep software stack that ranges from the operating system to the application. We present a conceptual version of such a software stack in Figure 2.1. The operating system is common for compute nodes and typically uses an open-source kernel version. Moreover, supercomputers often deploy a parallel file system for all nodes or the nodes of a dedicated I/O partition. A network management software stack is used for the operation of the interconnection network. At a higher level, the software stack

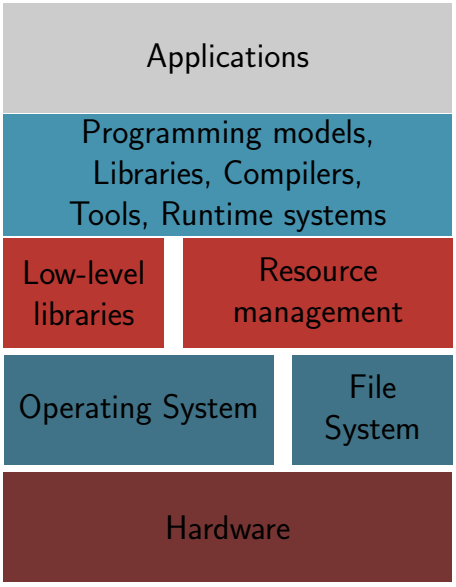


Figure 2.1: An example of the software stack of a supercomputer.

includes application development tools, compilers, communication and other parallel programming libraries, math and application-specific code libraries, utilized by the applications. All libraries may implement their own runtime systems.

Dedicated software layers manage the system resources. At a higher level, a resource management software (e.g., SLURM [Yoo et al., 2003], Torque[Staples, 2006], etc.) undertakes resource distribution and scheduling. At a lower level, software layers such as the operating system are responsible for feeding higher levels with information about the system status and operation. On supercomputers, users submit a job to the system, requesting a set of resources for the execution of their application, e.g., a number of nodes with certain characteristics, a specific number of cores, some amount of memory, nodes with accelerators and more. The resource management software allocates a set of resources that satisfies the user request, isolates the resources, schedules the jobs and monitors the system, the job and resource status. While the resource management software mainly serves the users requests, it also enforces scheduling policies to optimize the system utilization, targeting either the user (e.g., reducing waiting times) or the application (e.g., maximizing performance) or the system (e.g., maximize utilization). In any case, for a single job submission, the resource management software selects a particular resource partition (node allocation). Consequently, for a single execution of an application, the selected partition, the system topology and the application produce a unique resource

allocation and process mapping.

Scheduling, placement and mapping decisions, taken by the resource manager, are critical to communication performance. As we discuss in detail in following paragraphs, the mapping of the application communication graph on the given node allocation and resulting network graph produces a specific traffic pattern that defines communication performance. Being able to influence the decisions of the resource manager is important and will become critical for large-scale systems, to minimize data movement and optimize communication performance [Dongarra et al., 2011].

2.3 HPC applications

The parallel applications that execute on large-scale systems are mainly simulations of real-world phenomena from multiple scientific domains. In their majority, they are designed for execution on large-scale systems, following specific programming models that allow them to scale efficiently on HPC systems. HPC applications include multiple kernels and go through several phases of computation, communication and I/O. We describe the dominant programming models for HPC applications and discuss their communication patterns in the following paragraphs.

2.3.1 Programming models

The de facto programming model for distributed memory large-scale systems is the message passing model, best expressed through the Message Passing Interface [Snir, 1998], widely known as MPI. MPI is a rich programming model with various implementations, open-source of industrial, and is being actively supported and progressed by the MPI forum since 1992, in an effort to provide a portable software that bridges the gaps between application demands in performance and programmability and the system architecture. MPI implements SPMD (single program, multiple data) parallelism, and the parallelism entity is the process. Each process has a private virtual memory, where it stores its private data, and communicates with other processes through message passing. The MPI standard offers multiple functions for communication, which may be point-to-point (between at most two processes), collective (between multiple processes) or one-sided, where a process can read or write data from/to the remote memory of another process. The various MPI implementations deploy the network device drivers and their libraries and/or communication middleware to implement efficient and scalable communication functions for any type of node and network architecture. MPI is not a thin communication layer; apart from supporting multiple communication devices for maximum

portability, all MPI implementations make decisions, depending on the process mapping on the system, the type of communication, the message length and more, to provide optimal application performance. Such decisions involve selection of the optimal communication interface, e.g., the network interface or the memory system interface, the optimal transport layer and the optimal data transfer protocol, e.g., short (immediate transfer), rendezvous (transfer after ”handshake” between processes), buffered copy (transfer after copying data to internal buffers). In addition, they implement collective communication with low-complexity algorithms and may select the optimal algorithm at runtime, to optimize performance. Thus, the MPI implementation and its design affect communication performance in multiple ways.

The message passing model offers high performance, as its communication semantics are close to the native communication functions of the architecture, and at the same time offers high-level abstractions for the programmer to easily express the parallelism of their applications. For high-level abstractions of communication and/or parallelism, new programming languages have emerged in recent years, such as the PGAS (Partitioned Global Address Space) languages or task-centric languages. PGAS languages, e.g., UPC [Consortium et al., 2005] and OpenSHMEM [Chapman et al., 2010], offer a shared address space view of the distributed memory and the user can implement communication with simple read/write semantics, while knowing the location of data. Task-based languages are best represented by Charm++ [Kale and Krishnan, 1993] , an object-oriented language where the core parallelism entity is the task. Although these languages abstract the message passing for the user, their implementations utilize the message passing model (either MPI or communication middleware). MPI and other languages or libraries for programming over distributed memory address space are often combined with shared address space programming models, as is OpenMP [Dagum and Menon, 1998], to exploit on-node parallelism, or programming models and libraries for heterogeneous hardware and accelerators.

2.3.2 Communication patterns in HPC applications

Parallel applications running on HPC systems have certain characteristics that profile their communication. An application may include multiple communication phases, which can be point-to-point, one-sided or collective. A communication phase is characterized by a specific *communication pattern*, which determines the processes that will communicate, and by the number and lengths of messages exchanged by each process. The aforementioned characteristics constitute the *communication graph* of an application communication phase. As most large-scale applications iteratively execute some computational ker-

nels, the communication pattern of a phase can be static, i.e., be invariable during all iterations of the application, or dynamic. In addition, a communication phase can be discrete and synchronized, where all processes begin message exchanges at about the same moment, following a computation phase, or it can be non-discrete, where message exchanges are interleaved with computation, or non-synchronized, where processes begin message exchanges with skewed times due to computation load imbalance.

The communication pattern of an application phase depends on the problem that it solves, the solution method and the problem decomposition. The resulting message sizes and communication volume depend on the problem size. Collective communication predefines the communication pattern, since each function defines which processes exchange messages, in contrast to point-to-point or one-sided communication. However, specific problems come with specific point-to-point communication patterns. We follow the problem classification for high-performance systems given by Berkeley scientists in 2006 [Asanovic et al., 2006] to comment on the communication patterns that appear in each class of problems:

- *Dense Linear Algebra*: Dense linear algebra includes operations on dense matrices and vectors. In these problems, point-to-point communication regards matrix row/column exchanges or vector element exchanges and is limited. These problems usually use symmetric collective functions, rather than point-to-point communication, to distribute vectors and matrices, exchange vectors and collect results. Moreover, communication patterns in these problems are static.

- *Sparse Linear Algebra*: Sparse linear algebra problems include operations on sparse matrices and vectors. Communication in these problems also regards rows, vectors and elements of the matrices and vectors, as in dense linear algebra. However, due to sparsity, point-to-point communication is used more often. Sparse linear algebra problems exhibit static point-to-point communication patterns which follow the matrix shape and density. Each process may exchange different numbers of messages with other processes, while the message lengths depend on the non-zero elements of the matrix. Note that, while the communication pattern is static, the communication graph can be dynamic, i.e. the message lengths may change per iteration, depending on the problem. For example, sparse matrix-vector multiplication results in a static communication pattern and graph, but sparse matrix-matrix multiplication can result in a static communication pattern and dynamic communication graph. The problems in this class exhibit the most irregular, but static, point-to-point communication patterns.

- *Structured Grids*: Problems of this class regard solving partial differential equations on structured grids. The structured grid is expressed as an n-dimensional Cartesian grid and the problem is spatially decomposed to the grid. Communication in these problems is mostly point-to-point and static and matches with problem geometry. Processes communicate with a specific set of neighboring processes and exchange phases, edges, corners in the grid, depending on the numerical method and the problem dimension. Partial differential equations (PDEs) that appear in this class exhibit stencil-like computations (e.g., 6-point stencil in a 3D-grid) that exactly define the communication pattern.
- *Unstructured Grids*: Problems of this class, similarly to problems of the previous class, solve partial differential equations on unstructured grids, on a surface or volume with irregular geometry. In this class, the grid points are represented as a graph and subsequently, the grid is distributed to processes, either statically or with adaptive mesh refinement techniques. Communication is static and point-to-point. Each process exchanges messages of preset length with neighboring processes (nearest-neighbor pattern) which hold grid points in a common geometric neighbor.
- *N-Body*: N-Body problems involve dynamic systems of multiple particles/bodies, interacting with each other due to gravitation. Depending on the numerical method, problems of this class exhibit different communication patterns. Some methods statically decompose the space, using Cartesian geometry, and each process undertakes a Cartesian space with a number of particles. In this case, processes exchange particles with their neighboring processes in a point-to-point manner, however the communication graph is dynamic: as particles move in space under gravitation and/or interactions, processes exchange different numbers of particles, hence different message lengths. Other numerical methods use octrees to decompose the 3D-space. In these methods, communication is point-to-point, following the pattern of a hypercube, and the communication graph is static.
- *Spectral Methods*: Spectral methods, such as Fourier transforms, transform data from a temporal or special field to a spectral field and vice versa. Communication in these methods is collective, of all-to-all type, where all processes or subsets of processes exchange data with all other processes (or all other processes in the same subset).

Other broad classes of applications running on distributed high-performance systems are MapReduce applications and graph algorithms. In MapReduce applications, communication is negligible. On graph algorithms, communi-

cation is point-to-point or one-sided, heavily depends on the graph in use and both the communication pattern and the communication graph are irregular.

2.3.3 Communication performance

The communication performance of an application depends on the synergy of the application, the interconnection network, the system software and the mapping of the application on the system during execution. The application comes with multiple *communication patterns*, while the problem size and problem decomposition determine the sizes of the messages and designate the communication graphs and total communication volume for each pattern. Every time a user submits a job to the system for the execution of an application, the scheduler of the system undertakes the *allocation* of the resources requested by the user and the *placement* of the processes. This mapping of the application for the specific execution interacts with the system to produce a unique *traffic pattern* for each communication pattern in the application.

The *traffic pattern* determines the amount of data that flows through different points of the system and stresses them accordingly. First, a portion of data is exchanged between processes that reside on the same node. Intra-node communication is performed over the shared memory, stressing the memory system. If the node architecture is NUMA, some data exchange takes place between the memory chips of the node, over the on-chip interconnect, creating some latency. However, intra-node communication is in general faster than inter-node communication. The remaining portion of data is injected in the form of packets from the node to the backbone of the interconnection network. Contention effects may appear at this point due to limited injection bandwidth or due to the limited capacity of queues or buffers on the network interface card [Bhatele et al., 2015]. Once the packets are injected in the network, they travel through a number of intermediate switches and links to reach their destination. The path of each packet is determined by the topology and the routing protocol and it can be deterministic or dependent on current link congestion [Hoefler and Snir, 2011]. In the case of deterministic, min-hop routing, packets take the shortest path available, minimizing their latency by minimizing hops on the network, but are prone to congest the intermediate switches, even if links are not congested [Bhatelé and Kalé, 2009]. In the case of adaptive routing, some packets take longer paths and suffer greater latencies.

It is clear that the traffic pattern is a solid determinant of communication performance. The traffic pattern is unique to a specific execution of an application with a specific mapping of processes on the system. Alternate mappings can produce very diverse traffic patterns, with a varying outcome of communication performance, as the distribution of traffic changes and stresses dif-

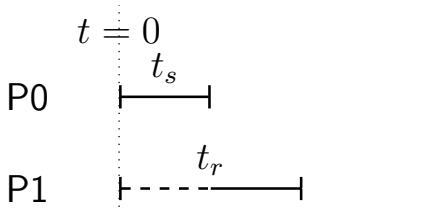
ferent points of the network[Jain et al., 2013]. However, the traffic pattern is not the sole determinant of communication performance. Time variability often occurs on real-world systems, due to non-deterministic sources at runtime. For example, allocations that populate more networking components are more prone to interference with co-running applications, also known as inter-application contention [Bhatele et al., 2013; Jokanovic et al., 2015] and may suffer from performance degradation, due to contention on these components. Depending on the system, performance degradation may occur due to switch sharing or non-deterministic routing. In addition to inter-application contention, system noise may originate from the nodes due to OS services or due to a low byte-to-flop ratio [Ferreira et al., 2013]. System noise causes a delay that may propagate throughout the communication phase or be overlapped by regular communication synchronization delays [Hoefler et al., 2010b]. Therefore, communication performance can vary arbitrarily.

2.4 Communication time

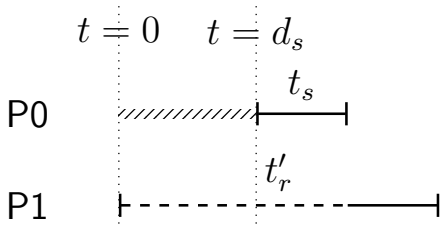
Besides the multiple factors that affect communication performance, the definition of communication time, in the context of an application, is not straightforward. There are significant differences between communication time per operation, communication time per process ("local" communication time) and communication time for all processes ("global" communication time). Also, each of the three can vary under different scenarios. In the following paragraphs, we discuss what we can define as communication time of an application.

2.4.1 Communication events and communication time

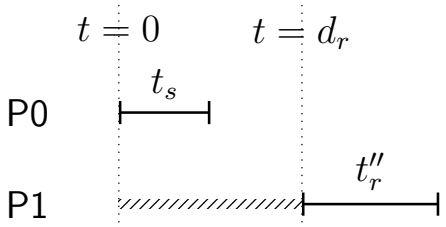
Communication within an application is a series of message transmissions between two or more processes, including the communication events that trigger or complete the transmissions. For point-to-point communication, communication events are explicitly defined by the programmer of the application, while for collective communication, the message passing library (e.g., MPI), transparently translates the operation to the necessary communication events. In theory, one can measure i) communication time for a single communication event, i.e., the communication time per operation, or ii) the time for a series of communication events happening on a single process, i.e., the communication time per process, or iii) the times for a set of communication events for all processes involved, i.e. the communication time for a phase. In practice, the definition of communication time, for all three cases, depends on a multitude of factors, which we demonstrate with an example in Figure 2.2.



(a) Synchronized send/receive



(b) Late sender



(c) Late receiver

Figure 2.2: Communication time for a send/receive operation between two processes

In this simple example, two processes communicate: Process 0 (P0) sends a message to Process 1 (P1). For simplicity, we assume non-blocking sends and blocking receives. In the first case, shown in Figure 2.2a, the two processes issue the communication events at the same time, $t = 0$. The time for the send operation by P0 is t_s, while the time for the receive operation by P1 is t_r. However, t_r does not necessarily correspond to the time needed for the receive operation: as the receive is blocking, t_r includes some idle time for process 1, waiting for the message transmission to begin. The communication time for process 0 in this case is t_{P0}=t_s and for process 1 is t_{P1}=t_r, including idle time for the latter. The global communication time is $t = max\{t_{P0}, t_{P1}\} = t_r$.

The second case, shown in Figure 2.2b, demonstrates the effect of the *late sender*. Process 1 issues its blocking receive operation at $t = 0$, however Process 0 does not issue its send operation until $t = d_s$. In this case, the duration of the send operation is still t_s, as it is non-blocking, but the duration of the receive operation is $t'_r > t_r$, including more idle time on the side of Process 1. The commu-

nication time for Process 0 is $t_{P0} = t_s$ and for Process 1 is t_{P1}=t'_r, including the additional idle time, here equal to d_s. The global communication time can now be defined in two ways. As in the case of synchronized communication events, we can define the global communication time as $t = max\{t_{P0}, t_{P1}\} = t'_r$, or we can assume that Process 0 has used the time between 0 and d_s to do some computation, thus we can assume that this period of time corresponds to computation/communication overlapping and we can thus define the global communication time as $t = max\{t_{P0}, t_{P1}\} - t_{overlap} = t'_r - d_s$.

The third case (see Figure 2.2c) is the case of the late receiver, where Process 1 does not begin reception of the message until $t = d_r$. In this scenario, t''_r may not include any idle time, if the transmission of the message over the network is complete. Local times can be defined as before. However, global communication time cannot be defined: the two communication events are disjoint: they do not overlap in time. t''_r depends on the period $d_r - t_s$ and whether transmission is complete within this period or not. Wolf et al. [Wolf et al., 2005] argue that communication time cannot be measured in the third case, when a send communication event has finished before its corresponding receive communication event has started.

Overall, the definition of communication time is not straightforward, even in the simple example of a single, point-to-point message transmission. This also affects the issue of measuring communication time. Micro-benchmarking of MPI communication requires fine-grained process synchronization and high-precision timers, to exclude skewing effects, as is the late sender/late receiver effect described in our example, so as to provide meaningful and reproducible measurements [Hoefler et al., 2010a; Hunold and Carpen-Amarie, 2016].

2.4.2 Communication time in HPC applications

Defining communication time in the context of an HPC application is even more intricate than the case of a simple message transmission, due to multiple reasons. First, processes communicate multiple times with various communication patterns. Second, communication patterns are more complex than a single send-receive operation. A collective operation alone, such as a broadcast, involves multiple message exchanges. Third, communication phases interleave with computation phases, and the latter can influence the duration of the former. Fourth, the number of processes used in HPC applications is in the order of thousands or millions. It is thus difficult to track communication events and their interactions, even post-mortem. Fifth, HPC applications avoid the use of explicit synchronization, like barriers.

A simple model for the total time of an application is the following:

$$T_{total} = T_{comp} + T_{comm} \quad (1)$$

This model assumes a global measurement for computation time and a global measurement for communication time. A more elaborate model for the total time of communication is given in [Barker et al., 2009]:

$$T_{total} = T_{comp} + T_{comm} - T_{overlap} \quad (2)$$

to account for overlapping between computation and communication. This model assumes an application with a single computation and a single communication phase, where computation is perfectly balanced, thus T_{comp} is equal for all processes and T_{comm} corresponds to the maximum communication time among all processes. For an application that involves M phases of computation and N phases of communication, we can extend this model as following:

$$T_{total} = \sum_{i=1}^{M} t_{comp,i} + \sum_{j=1}^{N} t_{comm,j} - T_{overlap} \quad (3)$$

We refer to a communication phase as a set of spatially and temporally related communication events and to a computation phase as a quantifiable set of computations. For this model to hold, several assumptions must also hold. First, it is necessary to be able to measure a global time $t_{comp,i}$ and $t_{comm,j}$ for each computation phase i and communication phase j respectively. Thus, one needs to be able to assume that, for any phase, they can either observe equal time across all processes or they can safely assume the maximum time across all processes as the duration of the phase. However, if computation is imbalanced for a single process, it can cause a direct delay in the subsequent communication event for one process, which can propagate throughout the communication phase, skewing spatially related communication events in a way that they are no longer temporally related. Thus, for the model to hold, one needs to assume the load of all computation phases is balanced across all processes. Second, it is necessary to be able to measure the time interval of overlapping of computation and communication $T_{overlap}$. This time interval depends on the available time for hiding communication, on the amount of computation time that is overlapped and on data dependencies between successive computation and communication phases [Sancho et al., 2006]. The ability to measure it requires knowledge of the exact times of request arrivals for send/receive operations [Chen et al., 2009]. Note, however, that when computation and communication phases overlap, the notion of communication as a phase collapses and direct or indirect delays may appear. Some recent studies perform post-mortem trace analysis to identify application phases, including communication phases, either by identifying logical phases which do not appear as temporal phases due to some source of delay [Isaacs et al., 2014], or by segmenting traces into meaningful clusters of events [Alawneh et al., 2016].

To summarize, a definition of the communication time of an application can only be given when i) there is no imbalance between processes, ii) there is no overlapping between computation and communication. In the case of applications that computation and communication overlap due to the nature of the algorithm, e.g., applications with wavefront patterns, only the per process communication time of the application can be defined. Thus, in the context of this thesis, we define the time of an application as following:

$$T_{total} = \sum_{i=1}^{M} t_{comp,i} + \sum_{j=1}^{N} t_{comm,j} \quad (4)$$

This model assumes no overlapping and discrete, balanced computation and communication phases.

2.5 Modeling communication time

Modeling communication time is as complex as the notion of communication time and communication performance. The task presents multiple challenges, and one can take different approaches to model communication time, depending also on the objective of the modeling process, namely the purpose for which the model is built. We discuss the various properties of a communication model and the different approaches, as well as their relation to the various factors that affect communication performance.

2.5.1 Properties of models for communication time

The two fundamental properties of a model for communication time are *accuracy* and *time of prediction*. Accuracy refers to the model's ability to provide a prediction for communication time with the smallest possible deviation from the actual (measured) communication time. Time of prediction refers to the time before, during or after the lifetime of the application when the model is able to provide a prediction for its communication time. Both properties define the possible uses of the model. An absolutely accurate model can support any scenario where predictions for communication time are required, however, in practice, accurate predictions are not always required for decision-making. For example, if the target of the prediction model is to decide whether a certain optimization for communication should be enabled for an application running on a specific system, then the accuracy of the model need only be good-enough to distinguish between cases when the optimization will reduce communication time and cases where the optimization will have no impact. Therefore, lesser accuracy can be tolerated if the model serves the purpose for which it is built. Time of prediction is equally important for the usage of the model. Ideally,

a model should be able to provide predictions for communication time way ahead of the execution of the application. In this case, the model can support decisions at compilation time, e.g., enable and disable optimizations, before runtime, such as scheduling decisions, or at runtime, such as auto-tuning. If the prediction is available after the execution of the application, the model is no longer predictive but explanatory and is useful only for analyzing an application's communication performance and behavior or for model-based simulation of an application. If the time of prediction is just-in-time before the execution of the application, the model can support runtime decisions. Similarly to accuracy, time of prediction needs to be good-enough for the purpose of prediction. Both accuracy and time of prediction heavily depend on the factors of communication performance that a model incorporates, as we discuss in the following subsection.

Another important property of a model for communication time is the prediction granularity, namely whether the model targets the modeling of communication time for a communication operation, a process or a communication phase. Modeling a communication operation can be particularly useful for hardware/software co-design and for communication software design. Modeling communication time for a process can support decisions regarding the redesign of communication patterns and other application optimizations. Models for a communication phase are useful for decisions on the placement and mapping of an application on a system or the scalability of a communication phase.

An additional property of models for communication time is their dependence on the target platform, which classifies models into analytical, semi-empirical or empirical. Analytical models provide an asymptotic form for communication time, based on nominal values of the system, such as the system's latency and bandwidth. An analytical model can be built for any platform without any dependence on the existence of the platform itself. As such, analytical models can be valuable to the design of a system or hardware/software co-design. Semi-empirical models extend the assumptions of analytical models with measurements and observations from the target platform. In practice, using information from the target platform improves the accuracy of analytical models and makes semi-empirical models suitable for more elaborate decisions on the target platform. Empirical models employ measurements and observations of the target platform to produce the model form, rather than extending the theoretical assumptions of analytical models. Empirical models can be far more accurate than analytical and semi-empirical models, albeit their dependence on the target platform renders them unsuited for system design. Apart from being a model property, dependence on the target platform defines the approach taken for communication performance modeling.

2.5.2 Tradeoffs in communication modeling

A minimal model for communication time would require information from the application and the architecture of the system, e.g. the application communication graph and network technology attributes. Such a model could be analytical and would provide an excellent time of prediction, at static time (or early at runtime, once the input parameters of the application are set). However, such a model would sacrifice accuracy for simplicity. To achieve the highest possible accuracy, a model should incorporate additional critical information, related to the traffic pattern. However, the shaping of the traffic pattern, which more accurately captures the distribution of the data flow within the network, demands knowledge of the process mapping and the allocation shape. This information only becomes available just-in-time before the execution of the application. Thus, a model incorporating details of the traffic pattern would sacrifice its time of prediction for more accuracy. Additionally, it would be more dependent on the platform, thus being semi-empirical or empirical.

Even with the knowledge of the traffic pattern at hand, non-deterministic sources of variability, such as OS noise and inter-application contention, can only be monitored at runtime. A model that would incorporate such information, if possible at all, would lose its predictive property and would be an explanatory model. The accuracy of any predictive model will always be subject to the limitation of runtime effects that can arbitrarily affect communication time. Similarly, the presence of computation load imbalance and computation/communication overlapping all affect communication time. To capture their effect, a model would require knowledge of exact times of request arrivals for communication events and/or delays and propagation of delays throughout a communication phase. Even in that case, communication time modeling and prediction under the presence of imbalance have little or no value: load balancing is the primary concern in this case. To the best of our knowledge, there is no prior work in communication time prediction that considers applications with load imbalance or computation/communication overlapping. The effect of computation on communication time can be taken into account only in holistic approaches for scalability modeling of parallel applications, as in [Shudler et al., 2015].

2.6 Problem statement

The aim of this work is to predict communication time $\hat{t}_i$ for an entire communication phase i of an application and provide a prediction for the total communication time of the application, $\hat{T} = \sum_i \hat{t}_i$. Our modeling approach is empirical, as we employ benchmarking on the target system to build predictive

Table 2.1: Target systems: a summary

	Vilje	Piz Daint	ARIS
Vendor	SGI Altix ICE X	Cray XC30	IBM NeXtScale nx360M5
Nodes	1404	5272	426
Node architecture	2 x 8-core Intel SandyBridge	8-core Intel SandyBridge + NVIDIA Tesla	2 x 10-core Intel Xeon E5-2680v2
Interconnection network	InfiniBand FDR	Cray Aries	InfiniBand FDR
Topology	Enhanced hypercube	Dragonfly	Fat tree
MPI Implementation	SGI MPI	Cray MPICH	Intel MPI

models for communication time. We need to note that, contrary to analytical models [Hockney, 1994; Culler et al., 1993; Alexandrov et al., 1995], which predict communication time in the scope of a single process, i.e., the time to complete a number of message transmissions, as measured by the sender process, we focus on prediction of the time of an entire communication phase, as measured by all participating processes. As to the model properties, our approach provides predictive models, namely with time of prediction ahead of the execution of the application. With respect to accuracy, we aim to provide models that are able to predict communication time of a balanced communication phase of a parallel application, with enough accuracy to allow for meaningful comparisons between communication time and computation time, so that they can enable application optimizations and resource management optimizations. Additionally, we aim to provide models that exhibit high goodness-of-fit, i.e., that they can distinguish between different communication configurations, problem sizes, numbers of nodes and cores and order communication time accordingly, to support relevant communication optimizations.

2.6.1 Target systems

The proposed methodology for communication time prediction is a generic methodology that targets large-scale systems and supercomputers that deploy high-performance interconnection networks. In this thesis, we apply the proposed methodology on three large-scale systems, for which we develop predictive communication models and evaluate prediction accuracy. The first system is the *Piz Daint* supercomputer, now a Cray XC50/XC40 system (Cray XC30 system until 2016), which uses the custom Cray Aries interconnect in a dragonfly topology. The second system is the *Vilje* supercomputer, an SGI system that uses InfiniBand FDR as an interconnect in an enhanced hypercube topology. The third system is *ARIS*, an IBM NeXtScale nx360M5 system, also using InfiniBand FDR to interconnect its nodes in a two-level fat tree topology. The three systems differ in their network technology and/or topology and represent 36.2% of the Top500 systems [top] (28.2% share for InfiniBand FDR and

8% for Cray Aries). InfiniBand technology is the most popular interconnect technology in the Top500 list, representing 37.4% of the systems. Cray Aries share is smaller, yet the dragonfly topology is expected to be utilized more in future HPC systems, due to its high bisection bandwidth. We summarize the properties of the three systems in Table 2.1.

Vilje[1] is an SGI Altix ICE X system at the Norwegian University of Science and Technology (NTNU), consisting of 1404 nodes of two 8-core Intel Sandy Bridge CPUs and 32GB of memory. The basic building block of the system is the Individual Rack Unit (IRU), which hosts 18 nodes and two 36-port FDR InfiniBand *switches*. Each IRU is a node of the hypercube topology. IRUs are stacked in groups of eight and interconnected in a 3D-hypercube topology to form a *rack*. The 19.5 racks of the system form a dual-rail, 8D enhanced hypercube, where redundant links are used to connect switches at the lower dimensions of the hypercube, providing higher overall bandwidth. Congestion control is achieved through a signaling mechanism: the source node is notified to throttle its injection of packets whenever congestion is detected on a switch on the path to the destination. The default MPI setup for Vilje is SGI MPI implementation.

The *Piz Daint* supercomputer [2], now a Cray XC50/XC40 installation at the Swiss National Supercomputing Center (CSCS), was until late 2016 a Cray XC30 supercomputer of 5272 compute nodes, comprising an 8-core Intel Sandy Bridge CPU, an NVIDIA Tesla K20X GPU and 32GB of RAM. On the upgrade from XC30 to XC50/XC40, the compute nodes have been upgraded, but the interconnection network and node topology remain identical. The work presented in this thesis was conducted on the Cray XC30 version of the system. The core component of the fabric is the *Aries SoC*, with four NICs connecting four nodes, forming a blade. Sixteen blades are accommodated on a *chassis*. A pair of cabinets, each hosting three chassis, forms a *group* of the network. Aries chips within a group are interconnected in an all-to-all topology via the chassis backplane and copper cables, while the 14 groups of Piz Daint are interconnected in an all-to-all topology through optical cables of slightly lower bandwidth. Congestion control is achieved through adaptive packet-level routing: a non-minimal path is selected if the estimated link load is lower than that of the minimal path. More details on Cray Aries can be found in [Alverson et al., 2012]. The default MPI setup for Piz Daint is Cray MPICH.

ARIS[3] is an IBM NeXtScale nx360M5 system at the Greek Research and Technology Network SA (GRNET). Its main partition includes 426 nodes of

[1] https://www.hpc.ntnu.no/display/hpc/Vilje
[2] http://www.cscs.ch/computers/piz_daint/index.html
[3] http://www.hpc.grnet.gr

two 10-core Intel Xeon E5-2680v2 CPUs and 64GB of memory. All nodes are connected on a Mellanox SX6536 648-port switch, which internally implements a two-level fat tree topology. In practice, the fat tree topology unfolds as following: the 648-port switch is implemented with 36 36-port switches. At the lower level of the fat tree, groups of 18 nodes are connected on one of 18 36-port switch. At the higher level of the fat tree, 18 36-port switches connect to all the switches of the lower level. As such, the topology is 1:1 and non-blocking, providing full bisection bandwidth. The system offers multiple MPI implementations to the users. We use Intel MPI for all our experiments.

2.6.2 Target applications

This work focuses on a typical class of large-scale applications that exhibit point-to-point communication phases that are symmetric as to the number of messages processes exchange. According to the classification of applications presented in the previous chapter, we target communication patterns that appear in problems on structured and unstructured grids, n-body problems solved on Cartesian grids and linear algebra applications that exhibit some point-to-point communication phases. Our target applications include, but are not limited to, quantum chromodynamics (QCD), hydrodynamics, weather, molecular dynamics and other simulations. In this way, we aim at developing a methodology that will deal with the core challenges in communication time prediction and that will provide accurate prediction models for a large number of large-scale applications. As our target is prediction of communication as a phase, we focus on applications where computation phases do not significantly affect communication time, i.e., on applications with balanced computation and no computation/communication overlap.

Regarding the programming model, we assume that communication is non-blocking, as well as that exchanged messages are explicitly packed in buffers, if necessary, thus excluding MPI derived datatypes, as their modeling falls into the category of computation time prediction and not communication time prediction. Algorithm 2.1 presents the kernel of a simplified application model with a single computation phase and a single communication phase. To ensure that no imbalance exists and communication time is not skewed, we utilize an *MPI_Barrier* function to synchronize all processes before measuring communication time, for comparison with our predictions. We note that, although this thesis is motivated by large-scale applications running on supercomputers, the presented approach is applicable to any application with point-to-point communication, as is, for example, large-scale graph processing under the BSP

model [Valiant, 1990] on Hama [4] or Pregel [Malewicz et al., 2010].

```
1: compute()
2: for ms in MessagesToSend do
3:     pack(ms)
4: end for
5: for mr in MessagesToReceive do
6:     MPI_Irecv(mr)
7: end for
8: for ms in MessagesToSend do
9:     MPI_Isend(ms)
10: end for
11: MPI_Waitall(MessagesToSend, MessagesToRecv)
12: for mr in MessagesToReceive do
13:     unpack(mr)
14: end for
```

Algorithm 2.1: Pseudocode for an MPI process in the application model

2.7 Related work

Analytical performance modeling of the communication time of parallel applications has been a hot topic in the past two decades. Hockney's model [Hockney, 1994] for point-to-point communication was a meaningful approach to express communication time for a pair of processors as a function of network-related parameters, namely startup time and maximal bandwidth. Culler et al. [Culler et al., 1993] attempted an elaborate specification of end-to-end communication time based upon network properties and the message or packet length, giving birth to the Log(G)P model family. LogGP [Alexandrov et al., 1995] and LogGPS [Ino et al., 2001] extend LogP with parameters for different message sizes and have been state-of-art in communication performance modeling, however, they come with several weaknesses and restrictions. Their major weakness is their focus on a local instance of the network, disregarding global network effects, such as overlapping, multiple hops, contention or congestion, which are addressed separately in LogGP extensions (LoGPC [Moritz and Frank, 1998], LoPC [Frank et al., 1997], LoGPG [Moritz and Frank, 2001]). As novel network architectures with offload-enabled network interfaces have

[4] http://hama.apache.org

emerged, models of the LogGP family are no longer applicable [Di Girolamo et al., 2015]; new parameters need to be introduced to capture operations on the network interfaces. In addition, building a model for a real-life application under Log(G)P parameters is not straightforward and several works, [Mudalige et al., 2008; Bauer et al., 2012] deploy the LogGP model to build communication performance models for large-scale applications on specific machines. More recent works [Bédaride et al., 2013; Zhu et al., 2015] attempt to encapsulate network contention in novel network models, limiting to specific interconnection network architectures, still restraining, though, to the modeling of network characteristics and communication primitives, as do the models of the Log(G)P family. In general, analytical models are architecture-agnostic, as they provide a generic form for communication time, based on design principles that apply to all HPC systems and interconnects. They are also cross-application, as they focus on a per-operation or local view of communication time that applies to any application under the message passing model. Finally, they are predictive, as their time of prediction is ahead of the execution of the application. However, the simple expressions of communication time given by analytical models cannot encapsulate all parameters that affect communication. As such, the accuracy of analytical models is limited and bound by the strength and number of parameters they use. Overall, analytical models trade accuracy for low awareness to runtime parameters and early time of prediction through simplicity [Hoefler et al., 2011]. Considering their purpose, analytical models are meaningful for hardware/software co-design and communication software design.

The alternative to analytical modeling is empirical modeling, i.e. the utilization of measurements on the target system for the modeling and prediction of application performance, through benchmarking or information collection at runtime. Empirical models, contrary to analytical ones, can achieve high accuracy by incorporating high levels of the system's characteristics. Related work on empirical modeling includes the work of Jain et al. [Jain et al., 2013] and Bhatele et al. [Bhatele et al., 2015], where performance models for communication are built for Blue Gene/Q deploying the network's performance counters. These models are highly accurate, as performance counter measurements allow for full awareness of the traffic pattern and runtime conditions on the given allocation. However, these models are explanatory: the values of the features can only be observed after the execution of an application, not allowing for communication time prediction ahead of execution. In addition, their methodology is strictly limited to systems as the BlueGene/Q, where network components in use are dedicated to the allocation for the application execution and performance counter measurements correspond only to traffic generated from the application in study. A semi-empirical approach is taken by Gah-

vari et al. [Gahvari et al., 2011, 2014], where the latency-bandwidth model is extended with empirical parameters for the modeling of the Algebraic Multigrid. This approach enhances the accuracy of the latency-bandwidth model by adding awareness of the process mapping, network architecture, and topology, and its time of prediction is at runtime prior to the execution, for many applications that expose point-to-point communication patterns, as the Fast Multipole Method [Ibeid et al., 2016]. These semi-empirical models are cross-platform, i.e., they rely on a methodology that can be applied to any target system with high-performance interconnection networks, although they require the definition of empirical parameters for the target system. In addition, similar to analytical models, they are cross-application and predictive. Both semi-empirical and empirical models increase prediction accuracy and are meaningful for enhancing decisions taken at runtime, e.g. scheduling decisions and application tuning at runtime. However, the drawback of empirical models is that they require an extensive set of measurements on the target system.

In this work, we provide a methodology for empirical predictive modeling of communication time of large-scale applications on HPC systems. Our approach, in contrast to prior art on empirical modeling, relies on features that can be evaluated ahead of the execution of the application, thus, our models are predictive and not explanatory. Additionally, in contrast to prior art in empirical modeling, our methodology is cross-platform, without any limitations. Moreover, our methodology is cross-application.

3

Building predictive communication models for HPC applications

3.1 Introduction

As discussed in the previous chapter, the parameter space for communication performance is large and complex. The first step towards communication time modeling is to enumerate and quantify communication-descriptive features. We investigate features that can lead to predictive models, applicable just-in-time before the application execution, thus we focus on features which can be extracted up to just after the allocation of processes (and before the actual execution of the application). We define multiple features and classify them according to their level of awareness to application, system and mapping characteristics, so as to explore the tradeoffs between accuracy and time of prediction. Then, we employ a benchmarking step to collect a set of measurements which will serve as a training set for the model construction. In the third step, we decide upon effective machine-learning methods, perform feature selection, if necessary, and build models with different numbers of features, methods and training sets. We present two different modeling approaches, one based on statistical learning methods and one based on machine-learning methods, all falling under the category of supervised learning.

In the following sections, we provide some preliminaries on supervised learning, we present the two core components of our methodology, namely features for communication time prediction and specifics for the benchmarking process. We then describe the steps we take to build various models for communication time using statistical learning and machine-learning methods. Finally, we present alternative models for communication time, drawn from recent related work, which we utilize later for purposes of comparison.

3.2 Model building with supervised learning

We assume there exists a vector of features $X = (x_1, x_2, ..., x_k)$ and a function f which determine the communication time t of an application, namely $t = f(X) + \varepsilon$. The goal of our work is to synthesize an approximation $\hat{f}$ of the function f to predict communication time $\hat{t} = \hat{f}(X)$ for any known or unknown input X, with an error $\varepsilon = t - \hat{t}$ that we can tolerate. The task falls under the category of *supervised learning*, where a learning algorithm utilizes a training set $Tr = \{(X_i, t_i), i = 1, ..., n\}$ to learn *by example* the approximation $\hat{f}$ [Friedman et al., 2009]. Since the output t is continuous, the task falls under the category of *regression*.

Feature selection

For each system we examine, we consider a plethora of features taken from the application and its mapping on the system. However, all these features are not necessarily significant for the prediction of communication time. When single-variable modeling is pursued, one needs to identify only one important feature, highly-correlated with the output. In the case of multiple variable modeling, systematic feature selection can assist in understanding the data and improving prediction accuracy [Chandrashekar and Sahin, 2014]. Techniques for feature selection include association filtering and recursive feature elimination and opt to eliminate redundant or irrelevant features which would add redundant information or noise to the model. The result of feature selection is a vector of features $X' = (x_1, x_2, ..., x_{k'}), k' < k$.

Methods for regression

There exist multiple methods for regression, with the simplest being *linear models* of the form $\hat{t}(X'') = b_0 + \sum_{i=1}^{k''} b_i x_i''$, where the elements of X'' include features from the original vector X, interactions between them, polynomial or non-linear transformations and possibly dummy variables. This method requires the specification of the function form by the user, while the coefficients are then computed through least squares. A linear model can be very effective for prediction, however, it may be arbitrarily complex in terms and branches, especially in the case of multiple variables. Moreover, the user is responsible for identifying all underlying relationships between the features at hand and the output (i.e. in our case, communication time). On the other hand, linear models provide a closed form function, where the relationship between a feature and communication time is clear, while other automated methods, as ensemble methods, only provide ranking scores for the selected features. We

refer to regression with the linear model as statistical learning, since the model form is "manually" specified, in contrast to machine-learning methods, where the model form results from the method itself. However, statistical learning is, in effect, a subset of machine learning.

Ensemble methods based on decision trees are more recent methods which have been proven to work well for regression problems. Decision trees are utilized by ensemble methods [Zhang and Ma, 2012] as weak learners. Bagging methods, such as random forests and extremely randomized trees, build multiple decision trees simultaneously which are then averaged to reduce variance. Boosting methods, such as AdaBoost and Gradient Boosting Machines subsequently build weak estimators, with each new one focusing on points neglected by the previous one, in order to reduce bias. The main advantage of ensemble methods is that they provide a black-box approach to regression. However, all ensemble methods come with a significant drawback. Unlike other regression methods, they are not able to extrapolate beyond the range of the training set. For example, assume building a model for a target variable y from a single variable x and assume that variable y is described by an identity function with x, that is $\hat{y} = f(x) = x$. Additionally, assume that we train two models for y, one using linear regression and one using an ensemble method, using values of x and y ranging from 0 to 1. If we try to predict y for $x = 2$, the linear regression model will predict $\hat{t} = 2$, while the ensemble model will predict $\hat{t} = 1$, which is the highest value in the training set. The inability of ensemble methods to perform extrapolations can be particularly restrictive, thus the training set needs to be selected carefully.

Evaluation metrics

To evaluate a model for communication time, we focus on its ability to predict communication time for any given communication pattern and execution configuration with a minimum error, as well as on its goodness-of-fit, i.e., how well the predicted values fit the actual values. For this purpose, we consider and examine several evaluation metrics. We note that no evaluation metric can definitively describe the accuracy and goodness-of-fit of a model and one should evaluate each predicted data point separately; however, different metrics can expose different properties of a model's accuracy and/or goodness-of-fit and all metrics combined are particularly useful in model comparison or comparison on different sets of points. In our workflow, a model predicts the communication times $\hat{t}_i, i = 1, ..., m$ for a set of points $T = \{(X_i, t_i), i = 1, ..., m\}$, collected from benchmark or application runs on the target system. The relative prediction error is then defined as:

$$e_i = (\hat{t}_i - t_i)/t_i$$

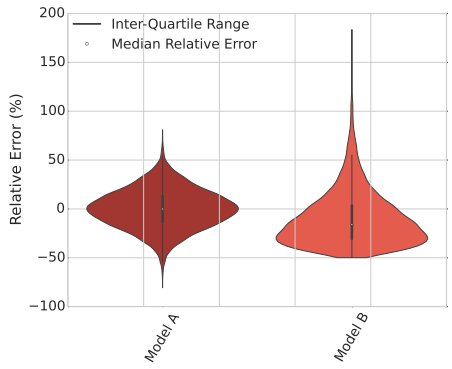


Figure 3.1: Sample violinplots for the relative errors of prediction with two fictional models

We examine the distribution of the relative prediction errors, along with the minimum, median and maximum relative error values. We use violinplots [Hintze and Nelson, 1998] as the means to visualize the relative error distribution and compare the relative error distribution between different models. Violinplots are a qualitative alternative to histograms. An example of violinplots for prediction errors with two fictional models is given in Figure 3.1. The relative errors of Model A follow a normal distribution and the relative errors of Model B follow a Gamma distribution. The ends of each violinplot correspond to the minimum and maximum relative error. The violinplots in our example show that Model A has a smaller range of errors than Model B. The width (or amplitude) of the violinplot for a specific level of the relative error indicates how many points in the examined set exhibit the specific error. For example, for Model A, the violinplot is wider for values of relative error close to 0% than 50%, thus more points exhibit an error of approximately 0%. For Model B, the violinplot is wider for values of relative error close to -40%, thus more points exhibit relative errors approximate to -40%. Also, for Model B, the violinplot is thin for relative errors higher than 100%, thus only a few points exhibit such high errors. In addition, the violinplot indicates the median relative error and the inter-quartile range, i.e., the points with relative errors between the 1st quartile and the 3rd quartile. Violinplots help us compare different models. In our example, Model A has a better distribution of relative errors than Model B: most of its errors concentrate around 0% and they are normally distributed around 0%, and its range of errors is smaller than that of Model B.

We also take into account the Mean Magnitude of Relative Errors *MMRE* [Conte et al., 1986], as a scalar alternative to the distribution of relative errors:

$$MMRE = \frac{\sum_{i=1}^{m} |e_i|}{m}$$

To assess the accuracy and goodness-of-fit of a predictive model, we examine the percentage of predictions at level 0.25, $Pred_{0.25}$:

$$Pred_{0.25}(\%) = \frac{\#predictions\ with\ |e_i| \leq 0.25}{\#predictions} * 100\%$$

$Pred_{0.25}$ takes values between 0 and 100 %, with higher values denoting higher accuracy. $Pred_{0.25}$ measures prediction accuracy at a fixed threshold for the relative error (in our case, 0.25 or 25%). A high value of $Pred_{0.25}$ indicates that the accuracy of the model is high for the particular set of points: most of the relative errors lie at most within a range of relative errors of $\pm$25% and thus concentrate around 0%.

We also examine the coefficient of determination R^2:

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(t_i - \hat{t}_i)^2}{\sum_{i=1}^{m}(t_i - \bar{t})^2}$$

and the rank correlation coefficient *RCC*:

$$RCC = \sum_{1 \leq i \leq m} \sum_{1 \leq j \leq m} concordant_{ij} / \frac{m(m-1)}{2}$$

where

$$concordant_{ij} = \begin{cases} 1, & if\ t_i < t_j\ and\ \hat{t}_i < \hat{t}_j \\ 1, & if\ t_i > t_j\ and\ \hat{t}_i > \hat{t}_j \\ 0, & otherwise \end{cases}$$

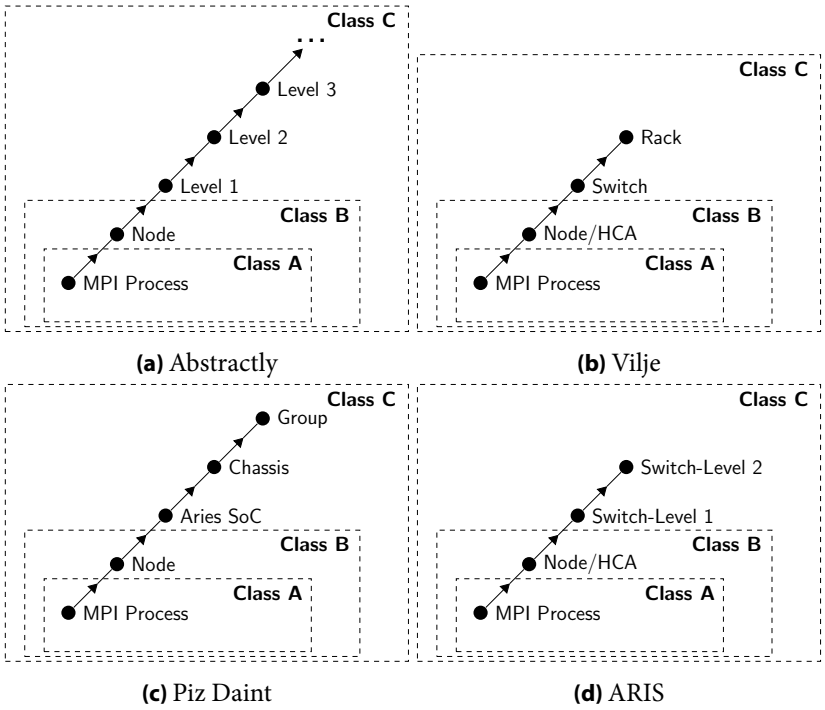
The highest value of R^2, denoting optimal fit, is 1, but the metric can also take negative values when the predicted data have not been used in the model-fitting process. *RCC* ranges from 0 to 1, with higher values denoting higher accuracy and goodness-of-fit. R^2 measures how well the model approximates the actual values, while *RCC* measures how well the ordering of the data is predicted. A high value of R^2 implies that the predicted values fit well the actual data and, as such, is a good metric for both accuracy and goodness-of-fit. A high value of *RCC* implies that the model is able to distinguish between different communication configurations; if one communication configuration leads to higher communication time than another, a model with a high *RCC* score also predicts higher communication time for the first communication configuration. We should clarify here the difference between R^2 and *RCC*: the former evaluates both a model's accuracy and goodness-of-fit, while the latter only evaluates goodness-of-fit. For example, if a model predicts the ordering of data accurately, with a constant error of 20%, the value for *RCC* would be 1, yet the value for R^2 would be lower, to reflect the 20% error.

3.3 Features for predictive communication modeling

3.3.1 Features for point-to-point communication performance

As a first step towards modeling point-to-point communication time, we define quantifiable features for the application communication profile, traffic pattern and allocation shape for a given execution setup on the underlying system. We divide our features into three classes by their level of awareness to the application communication profile, its mapping and the system architecture. This division allows us to construct predictors with different degrees of applicability and potentially prediction accuracy. *Class A* features are collected from the application communication graph and its execution configuration. Before the execution of the application, the user decides the number of nodes and processes per node they require. For a given problem size, the message lengths and numbers and per process traffic become known. Values for these features can be extracted at static time with minimal effort and thus any predictor built upon class A features has high applicability, e.g. it can support decision making regarding application design and algorithmic optimizations. *Class B* augments class A with features related to the mapping of the processes of the application on the given node allocation for the specific execution configuration. The mapping is either set by the user at the time of job submission or by the resource manager. Class B features measure the "local" traffic pattern, at the node level, but are unaware of the node allocation and the system architecture. Finally, *Class C* features sketch the traffic pattern from the application side, as well as the allocation shape. These features can be extracted only at runtime after the system scheduler provides the node allocation, just-in-time before the execution of the application. Thus, any model built with Class C features is still predictive, incorporates nonetheless high awareness to various parameters that affect communication performance.

The majority of the defined features capture the traffic (in bytes) and message rate (number of messages) injected from processes to nodes and to upper levels of the interconnection network. Fig. 3.2 demonstrates an abstraction of how the classes are specified for the three machines, exposing different levels of awareness of the system's topology and organization. All features related with data and messages sketch the outgoing traffic in bytes and the outgoing number of messages, from the processes of the application to those system components (nodes, switches, Aries SoCs etc.) utilized by the allocation for the specific execution of the application. Class A features are presented in Table 3.1. These features restrict themselves to knowledge extracted from the application, namely the application communication profile (message length, process data, and messages) and the size of the allocation, the number of nodes and pro-



(a) Abstractly **(b)** Vilje

(c) Piz Daint **(d)** ARIS

Figure 3.2: Classes of features a) on any system, b) on Vilje, c) on Piz Daint and d) on Aris. Class A features capture traffic from the processes, class B features capture traffic from the nodes, while class C features differentiate for the three systems and capture traffic on the upper levels of the topology. Notice there are more levels on Piz Daint than on Vilje and ARIS.

Table 3.1: Class A Features: Cross-Platform

Feature	Name	Description
n	Nodes	Number of nodes of the allocation
ppn	Processes Per Node	MPI processes per node of the allocation
l	Message Length	The length (in bytes) of messages sent by each process [max]
PD	Process Data	Data (bytes) sent by each process [max]
PM	Process Messages	Number of messages sent by each process [max]

Table 3.2: Class B Features: Cross-Platform

Feature	Name	Description
ND	Node Data	Data (in bytes) injected from a node to the network [min, avg, max]
NM	Node Messages	Messages injected from a node to the network [min, avg, max]
iND	Intranode Data	Data (in bytes) sent intranode [min, avg, max]
iNM	Intranode Messages	Messages sent intranode [min, avg, max]
TD	Total Data	Data (in bytes) injected by all nodes of the allocation
TM	Total Messages	Messages injected by all nodes of the allocation

cesses per node, which are provided by the user. Class B features, presented in Table 3.2 are aware of the mapping of the application on the allocated cores and nodes, thus include the intranode data and messages that may stress the memory system, internode data and messages that may induce contention due to limited injection bandwidth or limited capacity of network interface cards [Bhatele et al., 2015], as well as the total data and messages, denoting the total internode communication volume of the application. Features of classes A and B are cross-platform.

Class C features for Vilje are presented in Table 3.3. Apart from data and messages for switches and racks of the allocation, which can reveal congestion on intermediate switches, we also consider the number of switches, racks, the average number of nodes per switch and the average number of switches per rack, as descriptive features of the allocation shape, which is usually irregular on Vilje. Similarly, we define Class C features for Piz Daint in Table 3.4. The switches and racks of Vilje are substituted by the Aries SoCs, the chassis, and groups of Piz Daint. As in Piz Daint routing is adaptive and packets select a minimal or non-minimal path, depending on current link congestion, we additionally define the group-to-group data and messages, as indicative of excessive inter-group traffic that could stress the optical link between two groups and induce non-minimal routing. In Table 3.5, we present Class C features on ARIS. Note that ARIS has a two-level fat tree topology, where all switches at the higher level (Level 2) connect to all switches at the lower level (Level 1). Thus, we only consider traffic and messages at Level 1-switches, as there is no

Table 3.3: Class C Features: Vilje

Feature	Name	Description
sw	Switches	Number of switches where the nodes of the allocation reside
r	Racks	Number of racks where the nodes of the allocation reside
n/sw	Nodes per Switch	Nodes per switch in the allocation [avg, max]
sw/r	Switches per Rack	Switches per rack in the allocation [avg,max]
SD	Switch Data	Data (in bytes) injected from a switch to the network [min, avg, max]
SM	Switch Messages	Messages injected from a switch to the network [min, avg, max]
iSD	Intra-Switch Data	Data (in bytes) sent between nodes attached on the same switch [min, avg, max]
iSM	Intra-Switch Messages	Messages sent between nodes attached on the same switch [min, avg, max]
RD	Rack Data	Data (in bytes) sent from a rack [min, avg, max]
RM	Rack Messages	Messages sent from rack [min, avg, max]
iRD	Intra-Rack Data	Data (in bytes) sent between switches residing on the same rack [min, avg, max]
iRM	Intra-Rack Messages	Messages sent between switches residing on the same rack [min, avg, max]

way of knowing which or how many of the Level 2-switches are in use by any allocation. Additionally, to sketch the shape of the allocation, we consider the number of Level 1 switches, the number of nodes per Level-1 switches and the number of Level 1-switches per Level 2-switch. We should note that values for features like *ppn*, *n*, *sw* or *g*, that refer to the allocation are constant for an application execution, while the values for all other features are constant for a communication phase.

Table 3.4: Class C Features: Piz Daint

Feature	Name	Description
a	Aries SoCs	Number of Aries SoCs where the nodes of the allocation reside
c	Chassis	Number of chassis where the nodes of the allocation reside
g	Groups	Number of groups where the nodes of the allocation reside
a/c	Aries SoCs per Chassis	Aries SoCs per chassis in the allocation [avg,max]
c/g	Chassis per Group	Chassis per group in the allocation [avg,max]
AD	Aries Data	Data (in bytes) injected from an Aries SoC to the network [min, avg, max]
AM	Aries Messages	Messages injected from an Aries SoC to the network [min, avg, max]
iAD	Intra-Aries Data	Data (in bytes) sent between nodes attached on the same Aries SoC [min, avg, max]
iAM	Intra-Aries Messages	Messages sent between nodes attached on the same Aries SoC [min, avg, max]
CD	Chassis Data	Data (in bytes) sent from a chassis [min, avg, max]
CM	Chassis Messages	Messages sent from a chassis [min, avg, max]
iCD	Intra-Chassis Data	Data (in bytes) sent between Aries SoCs residing on the same chassis [min, avg, max]
iCM	Intra-Chassis Messages	Messages sent between Aries SoCs residing on the same chassis [min, avg, max]
GD	Group Data	Data (in bytes) sent from a group [min, avg, max]
GM	Group Messages	Messages sent from a group [min, avg, max]
iGD	Intra-Group Data	Data (in bytes) sent between chassis of the same group [min, avg, max]
iGM	Intra-Group Messages	Messages sent between chassis of the same group [min, avg, max]
GGD	Group to Group Data	Data (in bytes) sent between two groups [max]
GGM	Group to Group Messages	Messages sent from a group [max]

3.3.2 Extracting features from HPC applications

The features we define reflect the flows of data that result from the mapping of the communication graph to the network graph of a specific allocation. Features of different classes require different pieces of information, which can be extracted at different times in the execution lifetime of an application. Class A features only require information from the communication pattern, in the form of a time-independent trace, i.e. tuples of the form (*sender*, *receiver*, *message size*). This information can be automatically extracted with tracing tools, such as mpiP [Vetter and Chambreau, 2005] or TAU [Shende and Malony, 2006] and *tau2simgrid* [Desprez et al., 2011]. If the source code is available, this information can be extracted by simple code inspection. As we only need time-independent traces, i.e. we do not require time stamps for communication events, the collection of Class A features can be performed by tracing the application on fewer cores than the target scale, e.g. on a single or a few nodes. Class B features require information from the process mapping, i.e. the placement of

Table 3.5: Class C Features: ARIS

Feature	Name	Description
sw1	Level 1 Switches	Number of switches at the lower level of the fat tree, where the nodes of the allocation reside
n/sw1	Nodes per Level 1-Switch	Nodes per switch at the lower level of the fat tree in the allocation [min, avg, max]
sw1/sw2	Level 1-Switches per Level 2-Switch	Switches at the lower level of the fat tree per switch at the higher level of the fat tree in the allocation [min, avg, max]
S1D	Level 1-Switch Data	Data (in bytes) injected from a switch at the lower level of the fat tree to the network [min, avg, max]
S1M	Level 1-Switch Messages	Messages injected from a switch at the lower level of the fat tree to the network [min, avg, max]
iS1D	Intra-Level 1-Switch Data	Data (in bytes) sent between nodes attached on the same switch of the lower level of the fat tree [min, avg, max]
iS1M	Intra-Level 1-Switch Messages	Messages sent between nodes attached on the same switch of the lower level of the fat tree [min, avg, max]

processes on nodes, in the form of tuples (*rank*, *node*). The mapping is usually known to the user for an execution of an application, as the common practice is that the user defines either a default mapping for processes, such as block or round-robin placement of processes on nodes, or defines their own mapping, as a parameter to the *mpirun* wrapper script, or as an environment variable to Torque or SLURM scripts for job submission on HPC systems. Class C features require the list of nodes of the given allocation, which is given by the resource manager, Torque or SLURM, just before the execution of the application. They also require system-specific information from the underlying topology, which, in the case of InfiniBand networks, can be extracted by analyzing the output of

ibstat[1] and in the case of Cray systems, it can be extracted by analyzing the output of *xtnodestat*[2]. In both cases, this output provides the position of any node on the underlying topology and allows the extraction of Class C features. It should be noted that the Select Plugin of SLURM takes the decision of node allocation after submission, thus Class C information can become available long before the execution of the application and not "just-in-time".

3.4 Benchmarking

Benchmarking is a core process in our methodology as it allows us to explore the potential relationship between the defined features and communication time, observe ubiquitous performance effects, and finally build communication models for the target system. Benchmarking supplies an incisive dataset to train predictive models. We devise our benchmark to resemble the communication phase of an application and to be parametric to various features, allowing us to sweep an ample space of communication configurations and traffic patterns. Note that the benchmarking step needs to be applied only once (e.g. after the initial deployment of the execution platform) and thus its complexity and execution overhead is not on the critical path. In addition, our benchmarking is generic and serves for the communication time prediction of any application phase with point-to-point communication.

Our benchmark for point-to-point communication builds upon WICON [Bhatelé and Kalé, 2009], where random node pairs, with a single process hosted on each node, communicate in a ping-pong fashion simultaneously. WICON's goal is to quantify message latencies in the presence of contention. To capture contention and congestion effects due to multiple processes per node, we augmented the benchmark so that multiple processes hosted on each node communicate simultaneously with processes hosted on the paired node. To break the symmetry of this scheme, we switched from random node pairs to random MPI rank pairs, creating a fully randomized communication pattern. Moreover, to assess the impact of the number of messages sent by each process, we paired each process with M other randomly chosen processes, resulting in a scheme where each process sends M messages of the same length to M random processes and receives a reply. We also switched from blocking to non-blocking communication, as the latter allows overlapping at the system level between consecutive message transmissions and is a common practice for most MPI applications. The pseudocode for the core benchmark operations is listed in Listing 3.1.

[1] https://linux.die.net/man/8/ibstat

[2] http://pubs.cray.com/#/Collaborate/00256453-FA

```
1: for l = 1 to max_length do
2:     for iter = 1 to Iterations do
3:         MPI_Barrier(MPI_COMM_WORLD)
4:         gettimeofday(start_time)
5:         for m = 1 to ProcessMessages do
6:             MPI_Irecv(pong[m], l, MPI_UNSIGNED_CHAR, pair[m],...)
7:             MPI_Isend(ping[m], l, MPI_UNSIGNED_CHAR, pair[m],...)
8:         end for
9:         MPI_Waitall(2*m, ...)
10:        gettimeofday(stop_time)
11:        time[iter]=stop_time - start_time
12:    end for
13:    sort(time)
14:    local_reported_time=time[3*Iterations/4]
15:    MPI_Reduce(local_reported_time,global_reported_time,MPI_MAX...)
16: end for
```

Algorithm 3.1: Pseudocode for the core benchmark operations and benchmark timing

Our benchmark allows us to sweep the parameter space explicitly for four class A features: the number of nodes (n), the number of processes per node (ppn), the message length (l) and the messages per process (PM). Implicitly, by sweeping these four features, a large parameter space for all class A and B features is swept as well. Values of class C features are related to the allocation given to its job by the system schedulers, so there are two options for sweeping the parameter space for class C features: either to explicitly request specific node allocations with some specific configuration, or to execute the benchmark multiple times on different allocations, randomly picked by the scheduler. We followed the latter approach for all three systems, as on Vilje, the former approach is not applicable. We perform multiple executions of the benchmark on each system, as well as additional executions for short message lengths, in cases where we observed high variability for different allocations. Moreover, between different executions, we change internal benchmark parameters that affect the selection of random pairs of processes, so as to collect data for different ratios of inter-node and intra-node communication. The dataset collected through benchmarking associates all the features with communication time and serves as the training set for the construction of the communication model on each system. The wall-clock time to collect the training set for each system is less than four hours.

To collect communication time measurements, the benchmark calls a bar-

rier function, *MPI_Barrier*, before starting the timer (see Alg. 3.1), to avoid measuring process skew, (i.e. idle time), as communication time. We should note that although the *MPI_Barrier* is commonly utilized for process synchronization [Hunold and Carpen-Amarie, 2015], its synchronization quality depends on the system and alternative synchronization methods can provide better measurement quality. Each process measures its local communication time for each iteration and reports the 3rd quantile of the times measured across all iterations. Through reduction, the root process reports the maximum of the locally reported times that designates the completion of the communication face. We chose the 3rd quantile of the local communication time as a good trade-off between the average and the maximum time among iterations, to avoid extreme outliers that could occur for a certain iteration and at the same time to take into account some discrepancies and unexpected increases in communication time due to noise, congestion or interference from nearby jobs.

3.5 Predictive communication modeling with statistical learning

As a first approach to communication time modeling, we utilize statistical regression techniques that allow us to understand the exact relationship between the features we define and communication time. The aim of modeling is to associate communication time with the set of features described in Section 3.3, serving as explanatory variables. Assuming that communication is affected by multiple factors, reflected in the multiple features we have defined, we use multiple variable linear regression modeling, combined with robust regression to estimate the model coefficients, to deal with heteroskedastic errors and outliers that can affect the coefficients' values. In this modeling approach, we use all Class A and Class B features, while we only use Class C features that refer to the allocation shape. This early feature selection is driven by the inherent difficulty in the definition of the model form. The number of features under consideration needs to be constrained, in order to enable the examination of the relation of each feature with communication time and any interactions. Moreover, as most of our features describe traffic through different points of the network, many of the features are interrelated. Using interrelated features in the multiple variable linear model may prevent the regression method from converging in the calculation of the model's coefficients.

After selecting the method and the initial set of features, we select those features that will serve as explanatory variables of the model. First, we apply Pearson's correlation coefficients to decide which features have a high impact on communication time. Pearson's correlation coefficients or Pearson's r, for a

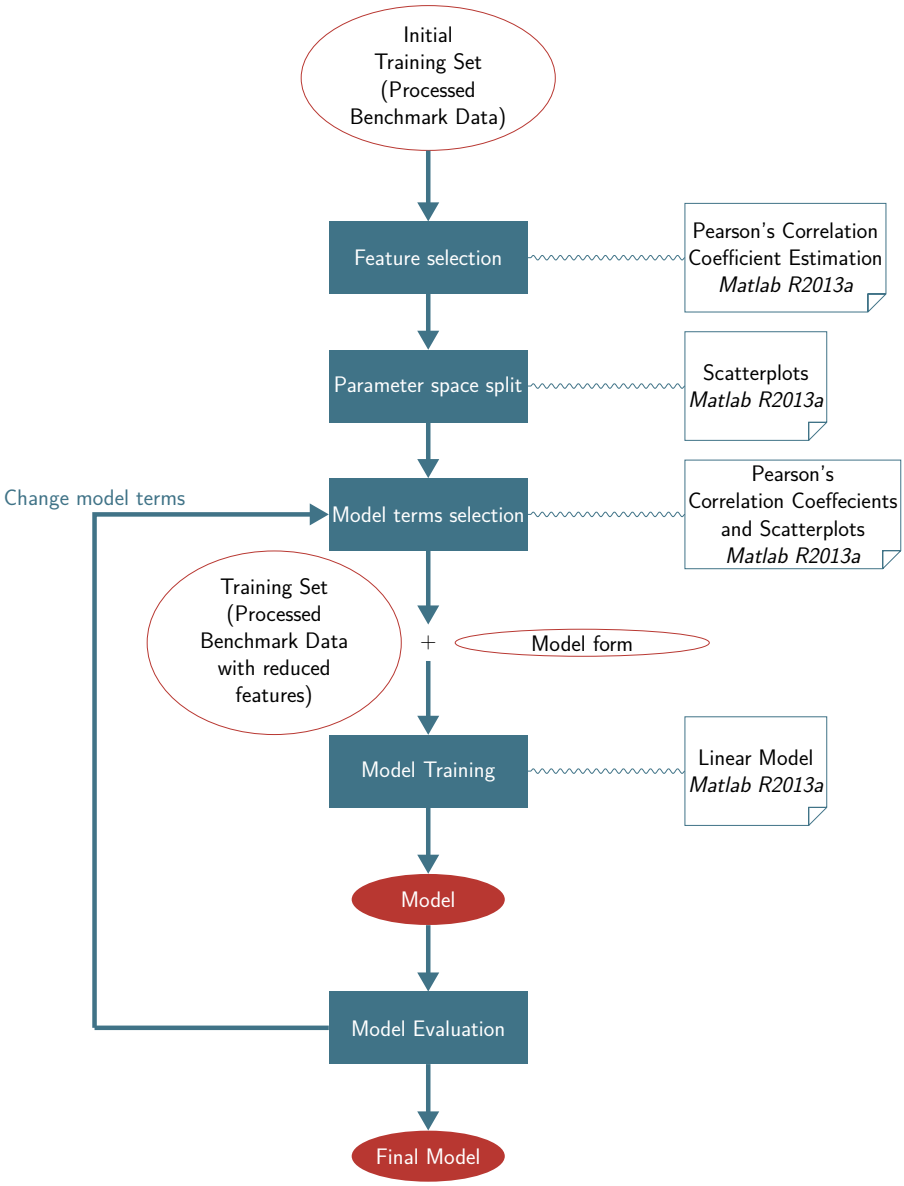


Figure 3.3: Building a model with multiple variable linear regression

feature x and communication time t is defined as follows:

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(t_i-\bar{t})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(t_i-\bar{t})^2}}$$

where n is the number of points in the training set. A value of (-)1 denotes a perfectly (positive) linear correlation between x and t, while a value of 0 denotes

no correlation. We also utilize scatterplots to inspect whether the correlation is linear and whether there is a need to build more than one models for different ranges of values of the various features. The scatterplots are also able to reveal the existence of interactions between features, and if this is the case, we perform clustering against other features, to extract interaction terms of the model(s), namely products of two or more variables. Second, we construct the model form by incorporating the selected variables and interaction terms. Finally, we apply robust multiple variable regression on the benchmark data to calculate the model coefficients, using the *LinearModel* class available on Matlab R2013a. The process of building a multiple variable linear regression model is visualized in Figure 3.3. We denote the models created with multiple variable regression as *LinReg - MV*.

Although the multitude of factors affecting communication time advocate for multiple variable models for communication time, in the process of modeling, we also detected features which were very highly correlated with communication time. This observation urges us to construct additional single-variable linear models for communication time. We work towards this direction to test the importance of utilizing multiple features for communication prediction. As single variable models are very easy to build, we select the most highly correlated feature for each system, from any class of features, to build single variable regression models. To identify the relationship between the selected feature and the target, and decide upon the model form, we construct a vector of polynomial and non-linear transformations of the target feature x, $X = \{1, x, x^2, x^3, log_2 x, log_2^2 x, log_2^3 x, \sqrt{x}\}$ and generate 7806 linear models of the form $\hat{t} = b_0 + \sum_{i=1}^{3} b_i x_j x_k$, $x_j, x_k \in X, j \neq k$, trained with the benchmark data. We then select the best-fitting model, denoted as *LinReg - SV*, standing for single-variable linear regression. We visualize the process of building single variable regression models in Figure 3.4.

3.6 Predictive communication modeling with machine learning

A significant drawback in using statistical learning methods to build predictive models is that they require significant effort from the user. They require manual feature selection, identification of linear or non-linear relationships between features and communication time, identification of interaction terms, high-order terms and parameter spaces that can accurately describe the relationship between a feature and the target. Contrarily, machine-learning methods can significantly automate model building, as they can perform feature selection and build models that incorporate the complex relationship of the features and

the target while hiding this complexity from the user.

We prioritize ensemble methods such as Random Forests, Extra Randomized Trees, Gradient Boosting Regression Trees and AdaBoost Regression, all available in Python's *scikit-16.1* [Pedregosa et al., 2011]. Automation of the model building process is the clear advantage of these methods which, in addition, eliminates the need to identify linear and non-linear relationships between the features and the target, interactions between features, high-order terms and parameter spaces that may more precisely describe the relationship between a feature and the target. Also, as ensemble methods combine multiple models, they improve in accuracy and robustness, compared to a single model[Mendes-Moreira et al., 2012]. These properties make ensemble methods an attractive solution for cross-system modeling, requiring no special effort to fit a model for each particular system. Finally, we note that the training overhead is negligible as the training time itself was low (i.e. less than two minutes for the most time-consuming methods) and it needs to be applied only once.

In this machine-learning approach, we build three different models for each system. The first model uses Class A features only, the second model uses Class A and B features and the third model also uses Class C features for each system. We refer to the three models as *Class A*, *Class B* and *Class C* model respectively. In this way, we can compare and assess the value of adding additional knowledge of the mapping and the system architecture and topology to a model, with respect to accuracy and time of prediction. Due to the design of our benchmark, which we use to collect measurements for training, we only use the maximum value for Class A features l, PD and PM, as in the benchmark, for a single execution on any configuration, the length and number of messages and process data are constant.

We start the model construction by feature selection on Class C features. Feature selection for Class A and Class B predictors is not necessary, as the default feature selection of the methods we apply is effective for the limited number of features of these classes. We rank Class C features using recursive feature elimination based on support vector regression, a method first proposed in [Guyon et al., 2002] for genetic diagnostics. Support vector regression provides high generalization ability, while recursive feature elimination is important to avoid high ranking of features that overfit the training set. We experiment with feeding 15 to 40 of the highest-ranked features as inputs to the aforementioned methods.

After feature selection, we use Gradient Boosting Regression Trees (GBRT) to build a model for communication time. GBRT is a boosting method, where a new weak learner, i.e. a regression decision tree, is iteratively fit on the negative gradient of a given loss function. GBRT offers high accuracy, alongside with automation and special tuning parameters that assist us in overcoming

the perks of communication time prediction. We utilize the least absolute deviation as the loss function, which allows for robustness to outliers. This property of the loss function is critical for our task, as noise is present in measurements of communication time and can result in several outliers. To prevent our models from overfitting to the training set, we perform regularization by employing shrinkage, namely defining a learning rate by which the impact of each consecutive boosting iteration is shrunk, and by tuning the shape of the decision trees. To find the optimal configuration for the GBRT method and the optimal predictor, according to our training set, we test the method with different values for the learning rate (*learning_rate*) and the minimum samples per leaf (*min_samples_leaf*) of the decision trees and minimum samples at which a split is performed (*min_samples_split*), as well as different numbers of decision trees (*n_estimators*) and select the values that give the model with the best fit, according to the evaluation metrics defined in Section 3.2. We denote the models constructed with GBRT as *GBRT-Class A*, *GBRT-Class B* and *GBRT-Class C*, according to the class of features and the utilized training set. We summarize the modeling process in Figure 3.5.

3.7 Analytical and semi-empirical models

For comparison purposes, we also implement analytical and semi-empirical models for communication time, as proposed by Gahvari et al. [Gahvari et al., 2011, 2014]. The baseline analytical approach is the $\alpha - \beta$ model, which is the equivalent of Hockney's latency-bandwidth model, where α is the latency and β is the inverse bandwidth. A third parameter, γ, denotes the per-hop delay and introduces a distance penalty, resulting in the α-β-γ model. We measure the three parameters with the HPCC benchmark[3]. The models can be enhanced with three penalties. A penalty can be added to the β parameter, to reflect the limits of achievable bandwidth and network contention due to link sharing. A second penalty can be added to the α parameter, to capture the effect of multiple processes accessing the network from the same node (multicore penalty). The same penalty can be added to the γ parameter, as multiple processes can create contention on every hop of a message on the network. The addition of these penalties results in 5 possible models. We evaluate all five models and present prediction results for the analytical α-β model and the best of the five semi-empirical, penalized models on each system. Note that, while we do not expect the α-β model to capture the complexities of communication time and give accurate prediction results, it constitutes the simplest expression for com-

[3] http://icl.cs.utk.edu/hpcc/

munication time, captures the effect of the problem size and decomposition and thus is a lower bound for prediction accuracy.

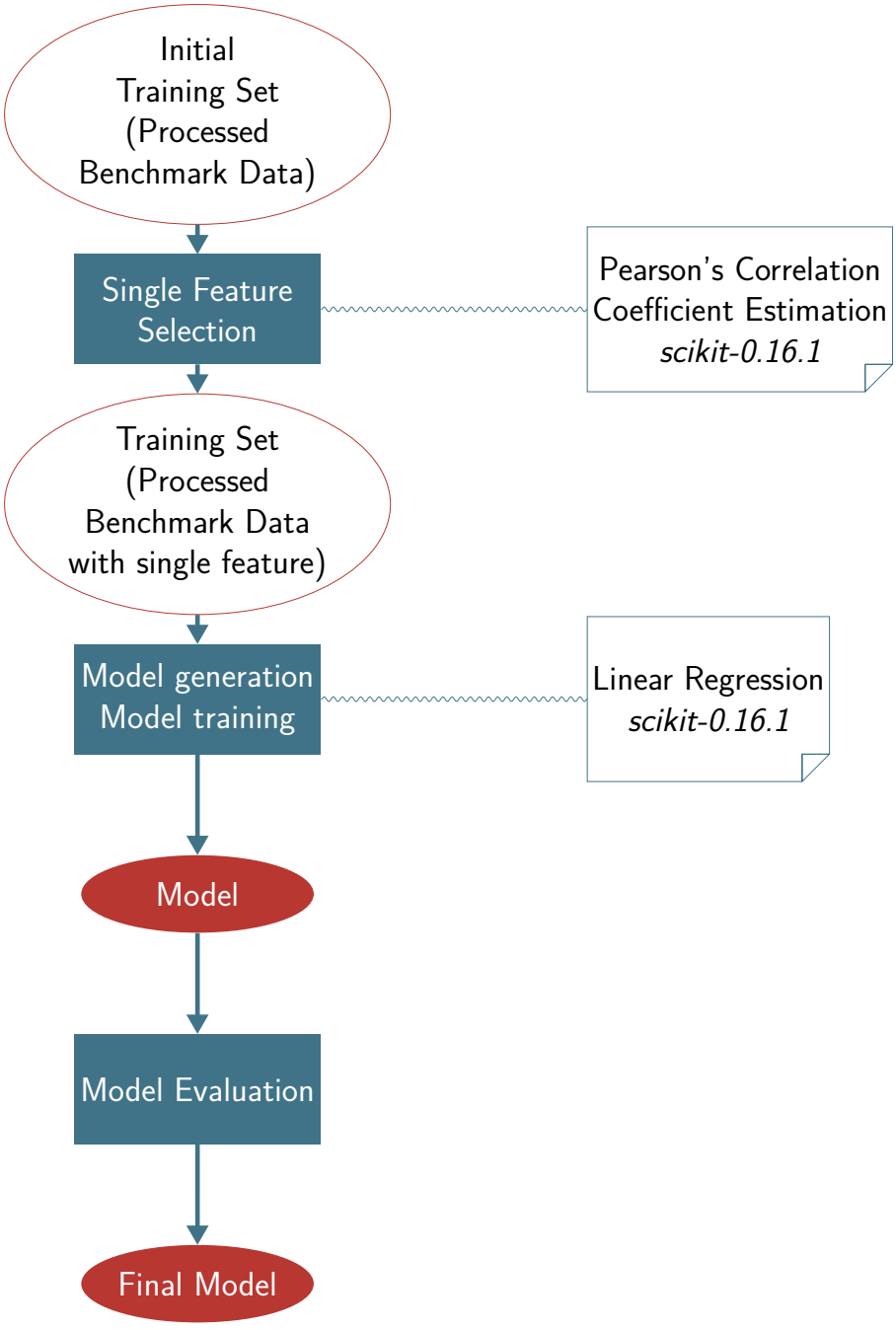


Figure 3.4: Building a model with single variable linear regression on non-linear terms

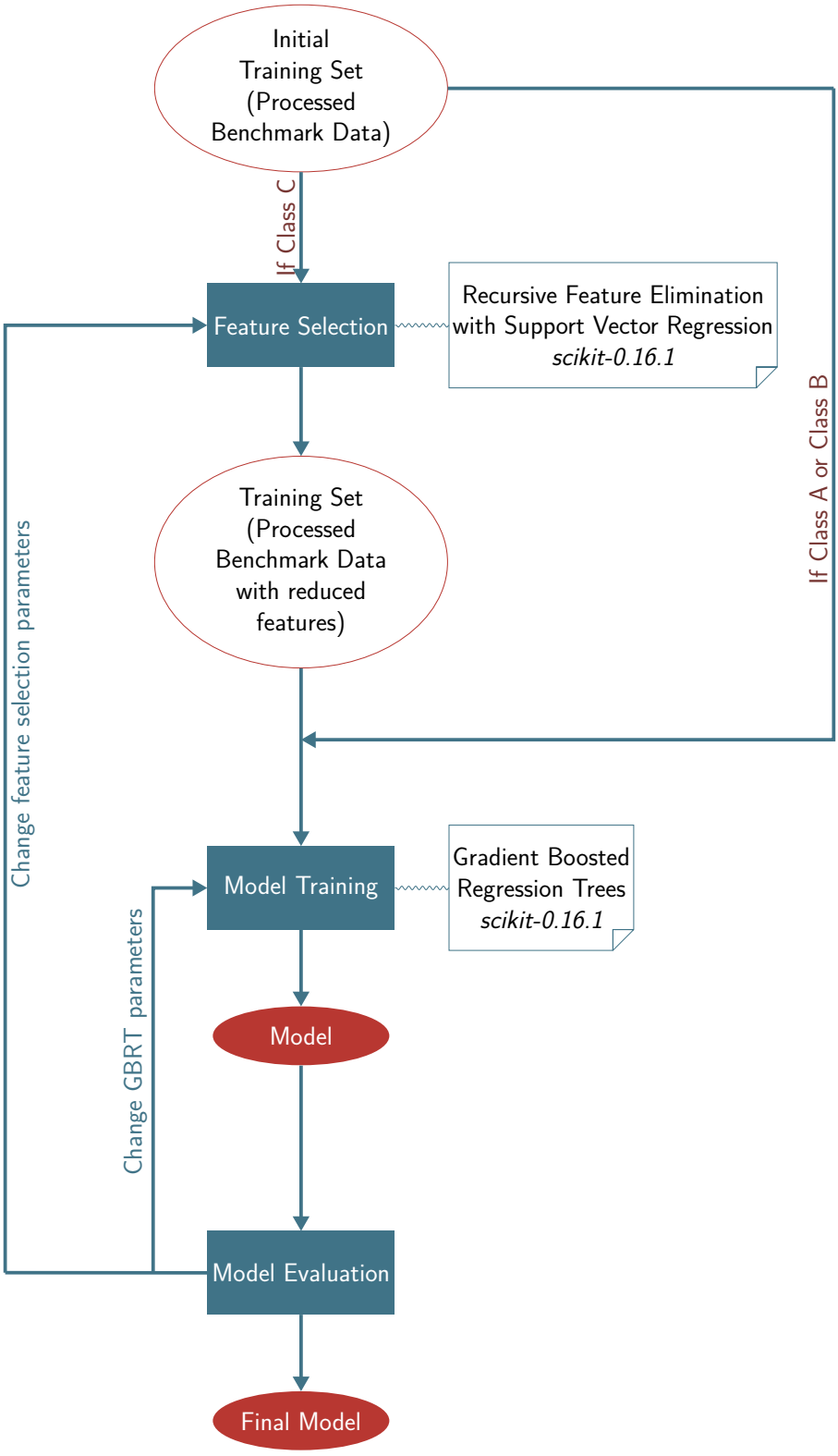


Figure 3.5: Building a model with Gradient Boosting Regression Trees

4

Predictive communication modeling on *Vilje*

4.1 Introduction

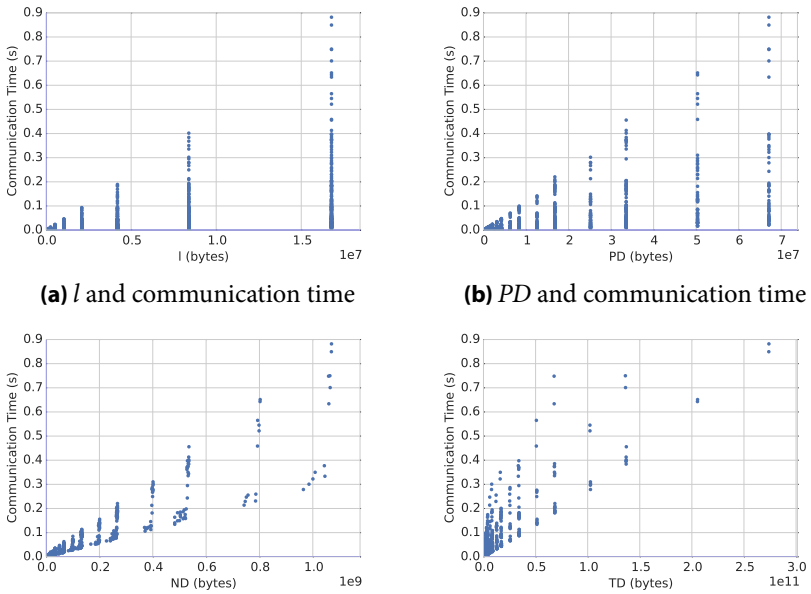
In this chapter, we present the application of our methodology (see Sections 3.5 and 3.6) for predictive communication modeling on the supercomputer *Vilje*. We detail the model building process both with statistical and machine-learning methods. We then evaluate the predictive ability of our models on several applications; we compare the various models and provide a detailed evaluation of the best performing model. Finally, we use our models for communication time in decision-making scenarios targeting the optimization of resource utilization on Vilje.

4.2 Modeling with statistical learning

To model communication on Vilje using the multiple variable linear regression model, we only utilized features of Class A and Class B, while we only selected three features from Class C, which are representative of the allocation shape: the number of switches *sw*, the number of racks *r* and the average number of switches per rack *sw*/*r*. For Class A and B characteristics referring to data or number of messages, we only used the average value. In addition, we omit the two Class B features related with intra-node traffic (data, *iND*, and messages, *iNM*). Instead, we use a feature that counts the ratio of inter-node to total communication volume, denoted as *I*. In this way, we reduce the number of features to 13, to facilitate the detection of relationships between all features and communication time and the construction of the model form. For the training set collection, we executed our benchmark on Vilje for various configurations,

Table 4.1: Correlation coefficients between features and communication time on Vilje - based on benchmark data

Feature	Correlation	Feature	Correlation
l	0.62	*TD*	0.87
PM	0.10	*NM*	0.24
ppn	0.20	*TM*	0.22
n	0.07	*sw*	0.07
I	0	*r*	0.05
PD	0.68	*sw/r*	0.06
ND	0.93		



(a) *l* and communication time

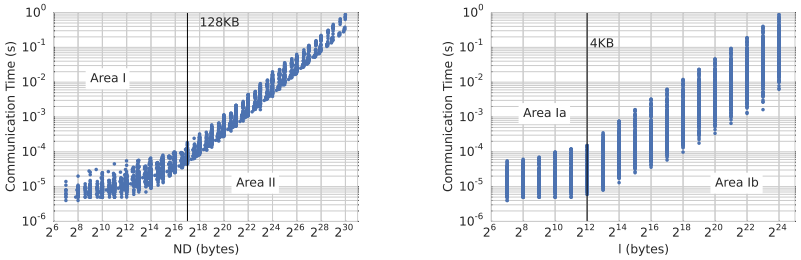
(b) *PD* and communication time

(c) *ND* and communication time

(d) *TD* and communication time

Figure 4.1: Scatterplots for features that exhibit high correlation with communication time on Vilje

from 8 to 128 nodes, 1 to 16 processes per node, 1 to 4 messages per process and various message sizes, ranging from 128 bytes to 16 megabytes. We repeated the benchmarking for all configurations, so as to collect different values for those features that depend on the allocation shape, that is, for features *sw*, *r* and *sw/r*, resulting in a training set of 4320 measurements of communication time. The two full executions of the benchmark also vary slightly in the communication pattern, as we changed the benchmark variables that create the



(a) Partitioning to Areas I and II

(b) Partitioning to Areas Ia and Ib

Figure 4.2: Partitioning the parameter space into sub-areas on Vilje

random pairs of communicating processes. Table 4.1 shows Pearson correlation coefficients between all features and communication time, where we note a significantly high correlation for Node Data *ND* and Total Data *TD* and a high correlation for Process Data *PD* and the message length *l*. To verify that this high correlation implies a linear relationship between these features and communication time, we inspected their scatterplots, shown in Figure 4.1, where a linear relationship becomes evident. However, a scatterplot for *ND* in logarithmic scale, as depicted in Figure 4.2a, revealed that the relationship of this particular feature with communication time has a different slope for lower and higher values of the features. Therefore, we partitioned the training set into two subsets/areas, Area I ($ND \leq 128 kilobytes$) and Area II ($ND > 128 kilobytes$). For Area I, the inspection of a scatterplot of the message length l with communication time, in logarithmic scale, depicted in Figure 4.2b also revealed different slopes for different message lengths, driving us to further partition Area I into two sub-areas, Area Ia ($l \leq 4 kilobytes$) and Area Ib ($l > 4 kilobytes$). We examined the three areas separately and constructed a model for each.

For each of the three areas, we re-computed the correlation coefficients, to construct simple linear models, by selecting the highest-correlated features for each area:

- Model Ia: $t_{comm} = b_0 + b_1 \times ND + b_2 \times NM$
- Model Ib: $t_{comm} = b_0 + b_1 \times ND + b_2 \times TD$
- Model II: $t_{comm} = b_0 + b_1 \times ND$

During model construction, extending the simple models given above with other highly correlated features did not improve the model accuracy. However, the scatterplots in Figure 4.1 showed some heteroskedasticity, namely a variance in communication time for a specific value of a feature. This observation led us to explore interaction terms, which are products of the selected features with other, not necessarily highly correlated features. To determine interac-

Table 4.2: Predictive model for Area Ia on Vilje: Terms and coefficients

Terms	Coefficients
Intercept	6.6111×10^{-6}
$ppn \times ND$	5.8226×10^{-12}
$l \times ND$	-1.42×10^{-13}
$r \times ND$	-5.967×10^{-11}
$ppn \times NM$	2.1315×10^{-08}
$l \times NM$	1.0384×10^{-09}
$r \times NM$	8.1344×10^{-08}
$ppn \times l \times ND$	-5.9903×10^{-16}
$ppn \times r \times ND$	1.2994×10^{-12}
$l \times r \times ND$	1.2228×10^{-14}
$ppn \times r \times NM$	-4.7418×10^{-09}
$ppn \times l \times r \times ND$	-3.7129×10^{-16}

Table 4.3: Predictive model for Area Ib on Vilje: Terms and coefficients

Terms	Coefficients
Intercept	0
l	7.7787×10^{-10}
sw/r	1.2193×10^{-06}
$l \times sw/r$	-6.1059×10^{-11}
ND	2.8278×10^{-10}
TD	1.2398×10^{-12}
$l \times ND$	-5.7408×10^{-15}
$sw/r \times ND$	2.6589×10^{-11}
$l \times TD$	3.6211×10^{-18}
$sw/r \times TD$	-1.8116×10^{-13}
$l \times sw/r \times ND$	6.9513×10^{-16}
$l \times sw/r \times TD$	2.3784×10^{-18}

tions, we took the following steps: initially, we excluded features which are products of other features and show inter-correlations, since they would lead to high order polynomial terms, instead of interaction terms. Subsequently, we used correlation coefficients to perform an initial selection of other features. For example, message length l show high correlation with communication time in all areas. Third, we chose to include at least one feature referring to the allocation shape. We experimented with various model forms, resulting in the model forms presented in Tables 4.2, 4.3 and 4.4, along with their coefficients,

Table 4.4: Predictive model for Area II on Vilje: Terms and coefficients

Terms	Coefficients
Intercept	0
l	-3.5128×10^{-11}
$ppn \times l$	-1.3962×10^{-11}
$sw \times l$	1.8077×10^{-11}
$ppn \times l \times sw$	-1.0744×10^{-12}
ND	2.5808×10^{-10}
$ppn \times ND$	-3.8401×10^{-12}
$l \times ND$	4.5466×10^{-18}
$sw \times ND$	1.3029×10^{-11}
$ppn \times l \times ND$	1.5495×10^{-19}
$ppn \times sw \times ND$	3.5872×10^{-13}
$l \times sw \times ND$	-1.6864×10^{-19}
$ppn \times l \times sw \times ND$	2.1983×10^{-20}

computed with robust regression using Matlab R2013a. We use the models for the three areas as a single model, denoted as *LinReg-MV*.

The final models allow us to make certain assumptions on the architecture and the factors that affect communication performance in each area. The length of the message l is definitive for the prediction ability of all models, as a baseline metric that resolves the meaning of other metrics that rely upon it. The number of processes per node ppn is included in models Ia and II, implying the existence of interference between processes residing on the same node. Model Ia, which refers to an area of small messages and low node traffic, incorporates the node messages metric NM, implying that the number of messages in flight has some effect in communication time, possibly due to an "eager" protocol for small messages. Model Ia also embraces the number of racks r, suggesting that dilation is important when the messages are small and more hops over the network add up to latency. Model Ib refers to an area of large messages but low node traffic, hence node data ND is not the only critical parameter: communication time also depends on the total communication data TD, as well as the shape of the allocation and the process mapping, as suggested by the existence of the estimated switches per rack sw/r in the model. Finally, Model II encloses an area of large messages and high node traffic, thus an area where more contention and congestion effects may arise at all levels, as implied by the encapsulation of the number of switches sw in the model.

To build a single variable model with statistical learning on Vilje, we followed the process described in Section 3.5. We first computed Pearson's cor-

related coefficients for all Class A, B and C features on the system and selected the feature with the highest correlation. In the case of Vilje, this feature was *SD (max)*, referring to the maximum outgoing data per switch. This finding is in accordance with our expectations about what affects communication performance: obviously, the amount of data flowing through switches, the core networking component on Vilje, significantly affects communication performance. However, we should note that this feature only refers to traffic produced by the application we examine (in this case, the benchmark), while on Vilje, switches may be shared between various allocations, on which different applications execute. In addition, switch data is inter-related with node data *ND*, which is the Class B feature with the highest correlation with communication time. After selecting the feature, we built and trained more than 7000 models based on *SD (max)*, using non-linear transformations of the feature, as explained in Section 3.5. Finally, we selected the model that best fitted our benchmark data. The resulting model for Vilje is of the following form:

$$t = 2.390 \times 10^{-5} + 4.335 \times 10^{-16} \times log_2^5 x + 1.915 \times 10^{-13} \times x log_2^2 x + 1.565 \times 10^{-14} \times x\sqrt{x} \text{ (seconds), where } x = SD(max) \text{ in bytes}$$

We denote this model as *LinReg-MV*.

4.3 Modeling with machine-learning

Following the process described in Section 3.6, we built three predictive models for communication time on Vilje, using Gradient Boosting Regression Trees, where each of the three models utilizes different classes of features. The first model, *GBRT-Class A* only uses Class A features, the second model, *GBRT-Class B*, uses Class A and Class B features, and the third model, *GBRT-Class C* uses features from all three classes. Unlike modeling with statistical learning, we did not perform any manual feature selection, thus we did not manually reduce the set of features for any of the three models. We should note, however, that we only use the maximum value for features *l*, *PD* and *PM*, since, due to our benchmark design, the values of these features are equal among all processes for a single data point collected with benchmarking. We applied systematic feature selection for the Class C model, using recursive feature elimination based on support vector regression. For Class A and B models, where the number of features is relatively small, we relied on the Gradient Boosting Regression Trees method to select the meaningful features for regression.

To train our GBRT models, we significantly augmented the original training set used in our statistical learning methodology. In particular, we added

Table 4.5: Parameter space for benchmark executions on Vilje for modeling with machine learning

Parameter	Range	
n	8-256	8-256
ppn	1-16	1-16
l	16B-16MB	16B-16KB
PM	1-4	1-4
#executions	3	2
#points	9210	

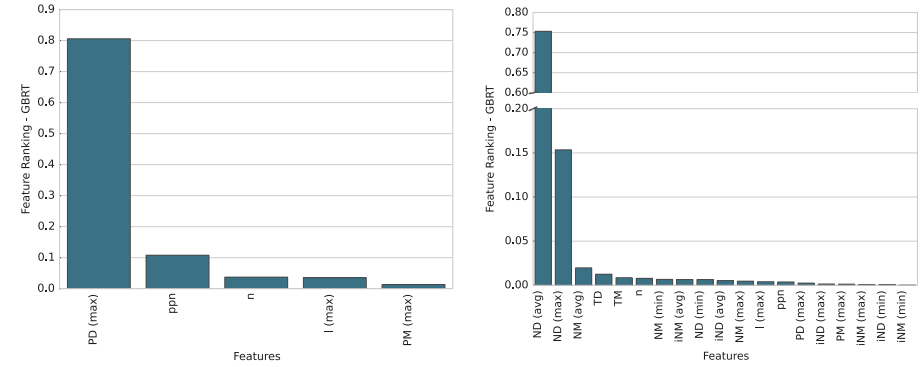
Table 4.6: Tested range of values and selected values for the parameters of the GBRT method and number of features on Vilje

Parameter	Range	Selected Value
n_estimators	100 - 2000	500
learning_rate	0.001 - 1	0.003
min_samples_leaf	1 - 5	2
min_samples_split	2 - 8	3
max_depth	3 - 8	7
features_classA	-	5
features_classB	-	19
features_classC	15 - 40	24

a full collection of benchmark data for all message sizes, numbers of messages and configurations, where we again slightly altered the variables that determine the random communication pattern. We also collected additional data points for short messages, as we observed high variance in communication time between different allocations and executions. The specifics of this enhanced training set, comprising of 9210 data points, are presented in Table 4.5. The enhancement of the training set was critical for the training of the Class C model, where different values of the features can only be produced through multiple executions on different allocations.

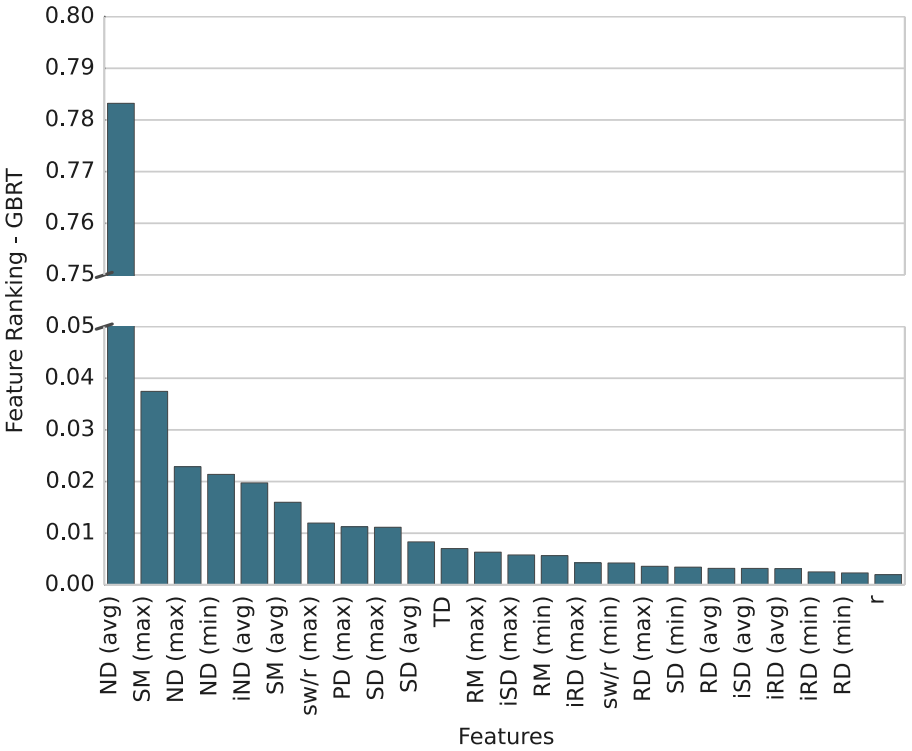
To find the optimal number of features and configuration for the GBRT method and to result in an optimal predictor, we tested the method with different values for its various parameters and selected the model with the best fit. The range of values tested and the selected values for the parameters of the method are presented in Table 4.6, along with the number of features (*features-classX*) utilized by the model for each class. The resulting predictive models occur from the GBRT method, tuned with these selected values and number

of features, trained with the augmented training set.



(a) GBRT-Class A

(b) GBRT-Class B



(c) GBRT-Class C

Figure 4.3: Feature ranking for the GBRT models on Vilje.

GBRT do not supply a closed analytical form for communication time, however, *scikit-learn* provides a method to rank the features based on their impact to the output, on a scale from 0 to 1, allowing us to draw useful obser-

Table 4.7: Details of the testing set on Vilje

	Vilje		
Pattern	Halo-3D	Halo-4D	LULESH
Domain Size	128^3, 256^3, 512^3, 1024^3, 2048^3	128^4, 256^4	240^3, 480^3
Iterations	256	256	100
n	16-512	16-512	8-225
ppn	1-16	1-16	1-16
#executions	3	2	1
#points	648		

vations about factors that affect communication time. It is important to note, though, that this ranking is relative to the specific method and the training set and is not an accurate measure of the relevance of the various features to communication time: important features may have been ranked low or omitted, while highly-ranked features are usually utilized for splitting the dataset in higher levels of the decision trees but would not provide accurate predictions if not combined with the remaining features. Nevertheless, an overlook of the selected features (see Figure 4.3) can later explain the predictive ability of each model. For model *GBRT-Class A*, Process Data (*PD*) is the highest ranked feature, followed by the allocation size, as defined by nodes *n* and processes per node *ppn*. The method does not ignore the new features belonging to class B and C when given as inputs, as indicated by the ranking of features for *GBRT-Class B* and *GBRT-Class C*, showing that the GBRT method is able to identify their correlation to communication time and validating our assumption that more information, extracted from the process mapping and the system architecture, can enhance the prediction ability of the method, as we will also demonstrate later. The predominant feature for both *GBRT-Class B* and *GBRT-Class C* is Node Data (*ND*), scoring higher than 0.7. Verifying our observations from model building with statistical learning, features that describe data flows at the node and the switch level show high rankings in the *GBRT-Class C* model, affecting primarily the modeling process. Interestingly, the distribution of data to messages is also critical for communication time prediction, as indicated by the importance of metrics for messages.

4.4 Evaluation

4.4.1 Communication patterns

We experimented with two communication patterns that are commonly encountered in real-world parallel applications, i.e. a *Halo-3D* exchange pattern, drawn from the 7-point-Jacobi relaxation, and a *Halo-4D* exchange pattern, drawn from Lattice QCD simulations. The *Halo-3D* (*Halo-4D*) exchange pattern is implemented with MPI as follows: processes are arranged in a virtual 3D (4D) cartesian grid and the original 3D (4D) domain is decomposed into smaller 3D subdomains (4D subdomains). Each process exchanges a 2D (3D)-face with each of the six (eight) neighboring processes.

We also applied our methodology to the point-to-point communication phases of LULESH [Karlin et al., 2012], the Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics proxy application. LULESH is a stencil-based code in three dimensions and exposes three point-to-point communication patterns in each simulation time step. The first pattern, *LULESH-1*, is a 27-point 3D-halo exchange for the communication of force vectors, the second pattern, *LULESH-2*, is a 7-point 3D-halo exchange for the communication of artificial viscosity and the third pattern, *LULESH-3*, is a 27-point 3D-wavefront for the communication of positions and velocities.

The selected communication phases expose four diverse, but very common nearest-neighbor patterns with different communication characteristics. In *Halo-3D* and *LULESH-2*, communication consists of 6 messages of equal size, i.e. the six 2D-faces of the 3D-subdomain. This simple pattern, which frequently appears in applications, is also found in the LLNL AMG2013[1] and Kripke[2] proxy applications, the ExMatEx CoMD[3] proxy application, CloverLeaf3D, miniAMR and miniGhost of the Mantevo[4] MiniApp suite, MG and SP of the NAS[5] Parallel Benchmarks and the LBL ExaCT miniGMG[6] proxy application. In *Halo-4D*, communication is denser, consisting of 8 messages of equal size, i.e. the eight 3D-faces of the 4D-subdomain. This pattern is present in all quantum chromodynamics codes, as is tmLQCD[7], MILC[8] and PRACE QCD benchmarks[9]. *LULESH-1* processes exchange 26 messages of three dif-

[1] https://codesign.llnl.gov/amg2013.php
[2] https://codesign.llnl.gov/kripke.php
[3] http://www.exmatex.org/comd.html
[4] https://mantevo.org/
[5] https://www.nas.nasa.gov/publications/npb.html
[6] https://ccse.lbl.gov/ExaCT/index.html
[7] https://github.com/etmc/tmLQCD
[8] http://physics.indiana.edu/~sg/milc.html
[9] http://www.prace-ri.eu/ueabs/#QCD

Table 4.8: Measured parameters for the analytical and semi-empirical models on Vilje

Parameter	Value
α	0.305 us
β	0.215 ns
γ	0.257 us

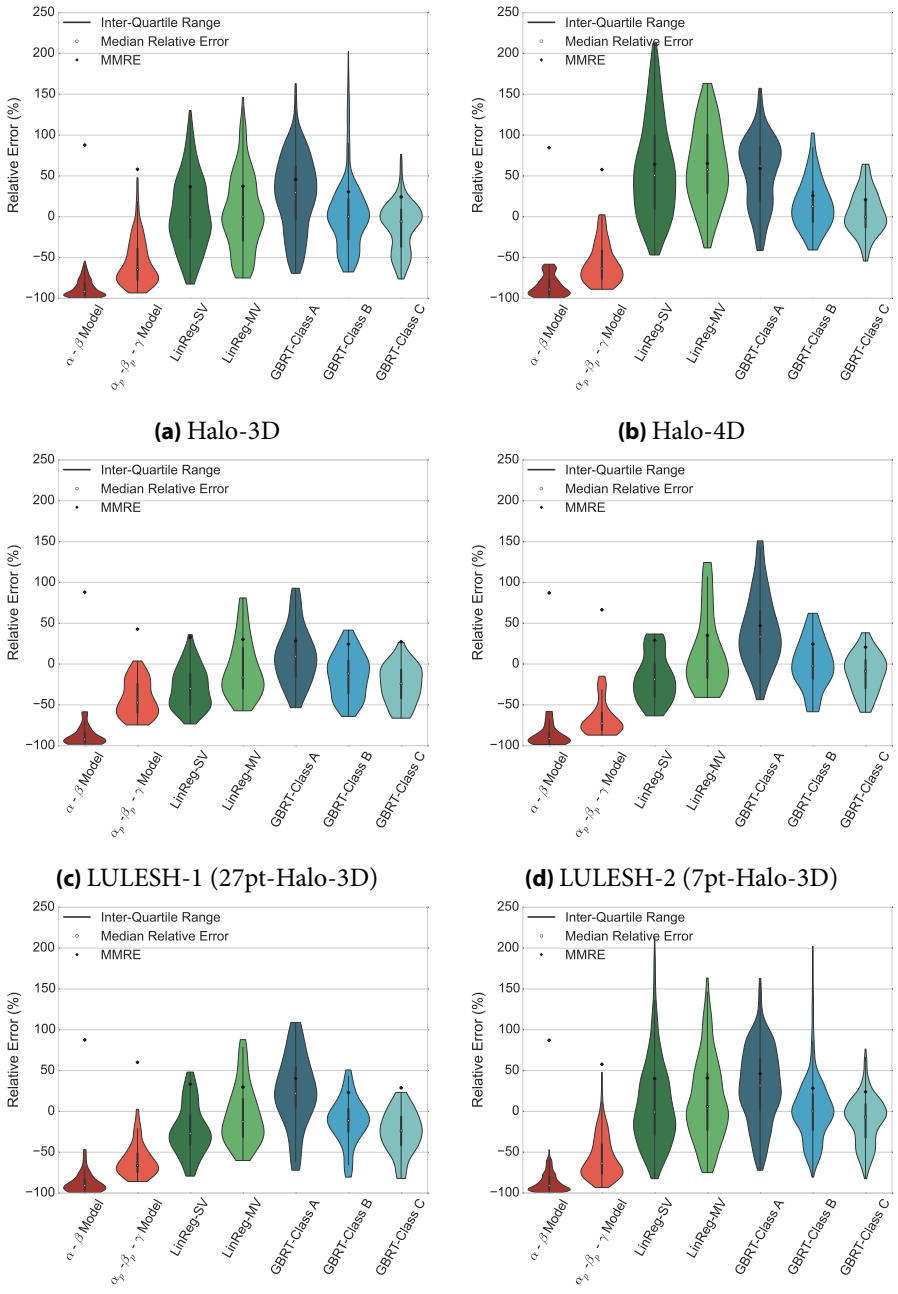
ferent sizes, arising from the geometry of the 3D-subdomain: six 2D-faces, twelve 1D-edges and eight corners of one element. This pattern also appears in numerous HPC applications and can be found in the LLNL Lassen[10] and AMG2013 proxy applications, the ANL CESAR NekBone[11] proxy application, as well as in HPCCG, miniFE and miniGhost of the Mantevo suite. *LULESH-3* exposes a wavefront pattern, in which processes exchange messages of three different sizes, namely faces, edges and corners, but communication takes place diagonally, i.e. each process sends only 13 messages to thirteen neighbors and receives 13 messages from the remaining thirteen neighbors.

We predict communication time for *Halo-3D*, *Halo-4D* and LULESH for various problem sizes and execution configurations, as well as for multiple executions, in order to test the ability of class C features to describe the effects of distinct allocations. The specifics of the testing set for Vilje is given in Table 4.7. LULESH by design expects the number of processes to be a power of 3, so all execution configurations for LULESH are bound by this constraint.

4.4.2 Model comparison

To evaluate the predictive ability of our models, we utilize the metrics defined in Section 3.2. We compare the predictive accuracy and goodness-of-fit of the two models built with statistical learning, i.e., *LinReg-SV* and *LinReg-MV* and the three models built with machine-learning, i.e., *GBRT-Class A*, *GBRT-Class B* and *GBRT-Class C*. We also compare the predictive accuracy for the analytical $\alpha - \beta$ model, denoted as $\alpha - \beta$ *Model*, and the semi-empirical $\alpha - \beta - \gamma$ model with penalties on the α and β parameters, denoted as $\alpha_p - \beta_p - \gamma$ *Model*, which we promoted as the best-fitting semi-empirical model on Vilje. The measured values for the α, β and γ parameters are given in Table 4.8. Note that the penalty on the α parameter is a multi-core penalty, while the penalty on the β parameter is a penalty for limited bandwidth and network contention.

[10] https://codesign.llnl.gov/lassen.php
[11] https://cesar.mcs.anl.gov/content/software/thermal_hydraulics



(a) Halo-3D

(b) Halo-4D

(c) LULESH-1 (27pt-Halo-3D)

(d) LULESH-2 (7pt-Halo-3D)

(e) LULESH-3 (27pt-Wave-3D)

(f) Aggregate

Figure 4.4: Model comparison on Vilje.

Table 4.9: Model comparison through accuracy and goodness-of-fit metrics on Vilje

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.00	-0.094	0.800
$\alpha_p - \beta_p - \gamma$ *Model*	12.667	0.130	0.898
LinReg-SV	40.88	0.801	0.915
LinReg-MV	41.33	0.667	0.922
GBRT-Class A	29.11	0.730	0.926
GBRT-Class B	50.89	0.774	0.936
GBRT-Class C	62.44	0.785	0.939

(a) Halo-3D

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.00	-0.450	0.742
$\alpha_p - \beta_p - \gamma$ *Model*	12.50	0.004	0.794
LinReg-SV	28.33	0.363	0.893
LinReg-MV	20.00	0.590	0.931
GBRT-Class A	24.16	0.458	0.912
GBRT-Class B	59.17	0.949	0.937
GBRT-Class C	67.50	0.938	0.935

(b) Halo-4D

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.0	-2.042	0.585
$\alpha_p - \beta_p - \gamma$ *Model*	26.92	-0.238	0.790
LinReg-SV	42.31	0.568	0.873
LinReg-MV	48.08	0.579	0.849
GBRT-Class A	53.85	0.384	0.882
GBRT-Class B	57.69	0.829	0.893
GBRT-Class C	50.0	0.738	0.875

(c) LULESH-1

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.0	-1.445	0.386
$\alpha_p - \beta_p - \gamma$ *Model*	9.62	-0.699	0.636
LinReg-SV	40.38	0.669	0.765
LinReg-MV	48.08	0.498	0.860
GBRT-Class A	36.54	-0.219	0.878
GBRT-Class B	61.54	0.863	0.892
GBRT-Class C	65.38	0.891	0.900

(d) LULESH-2

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.0	-2.041	0.535
$\alpha_p - \beta_p - \gamma$ *Model*	7.69	-0.735	0.799
LinReg-SV	40.38	0.506	0.811
LinReg-MV	42.31	0.605	0.839
GBRT-Class A	42.31	-0.072	0.835
GBRT-Class B	65.38	0.708	0.842
GBRT-Class C	51.92	0.469	0.817

(e) LULESH-3

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.0	0.030	0.825
$\alpha_p - \beta_p - \gamma$ *Model*	13.09	0.309	0.896
LinReg-SV	38.84	0.635	0.927
LinReg-MV	38.84	0.721	0.928
GBRT-Class A	31.54	0.667	0.937
GBRT-Class B	54.55	0.926	0.942
GBRT-Class C	61.85	0.922	0.942

(f) Aggregate

Figure 4.4 shows the distribution of the relative errors of predictions, for all models on each communication pattern, as well as on aggregate (see Figure 4.4f), in the form of violinplots, while Table 4.9 summarizes the scores of the $Pred_{0.25}$, R^2 and *RCC* metrics. We can, therefore, make the following observations: first, the baseline analytical $\alpha - \beta$ model severely underpredicts communication time in all cases, because it only relies on the latency and bandwidth for prediction, and shows poor goodness-of-fit. The semi-empirical $\alpha_p - \beta_p - \gamma$ model improves upon the analytical model, yet it still underpredicts communication time in all cases. The inadequacy of this model owes partly to the perks of the enhanced hypercube topology and resource allocation of Vilje;

the penalty on β attempts to model contention in relation to the maximum attainable per-node bandwidth and link sharing. However, on Vilje, contention effects arise mainly at the switch level, which may be shared by multiple applications, while the links of the hypercube topology may be under-utilized, due to the dimension-order routing.

Focusing on our models, the two models built with statistical learning, *LinReg-SV* and *LinReg-MV*, exhibit almost equal accuracy and goodness-of-fit on all patterns and on aggregate, performing moderately on overall accuracy, as indicated by the values for $Pred_{0.25}$ and R^2 and by the high relative errors for patterns *Halo-3D*, *Halo-4D* and *LULESH-2*. However, their *RCC* scores are high, and they are thus able to distinguish between different communication configurations. Clearly, the best performing models are *GBRT-Class B* and *GBRT-Class C*. The two models score almost equally in R^2 and *RCC* scores, however, the *GBRT-Class B* model often overpredicts communication time, leading to a higher range of relative errors, while the *GBRT-Class C* model achieves better error ranges and a better $Pred_{0.25}$ score on aggregate. Finally, *GBRT-Class A* model overpredicts communication time for all patterns. This behavior proves that Class A features, in combination with our generic benchmarking, are insufficient for sketching the traffic pattern and achieving accurate predictions on Vilje, where communication performance is highly influenced by effects occurring at the node and switch level, and the distribution of data to different messages is important, as indicated by the feature ranking for Class B and Class C models. The accuracy of *GBRT-Class A* can be boosted by augmenting the training set with data from irregular patterns, in order to include minimum and average values for the features *l*, *PD* and *PM*.

4.4.3 Detailed evaluation

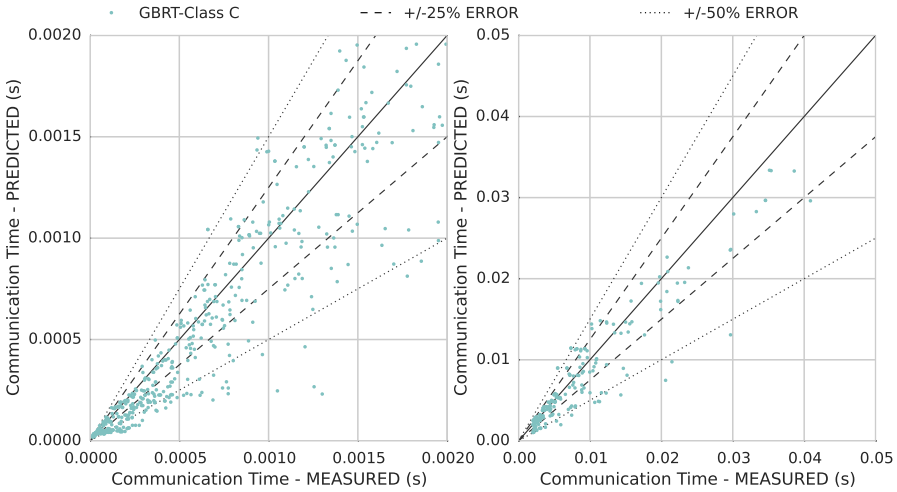


Figure 4.5: Scatterplots for predictions with *GBRT-Class C* on Vilje. Communication time is normalized to a single iteration.

Based on the model comparison in the previous paragraphs, we promote *GBRT-Class C* to be the best model for Vilje, as it achieves better error ranges than *GBRT-Class B*, which otherwise scores highly on all metrics and, in some cases, outperforms *GBRT-Class C*. We consider this model to be the most useful, as its time of prediction is just-in-time before the execution of the application, its accuracy is very high, scoring 61.43% in $Pred_{0.25}$, and its goodness-of-fit is excellent, scoring 0.926 in R^2 and 0.942 in *RCC*. In addition, the *GBRT-Class C* model achieves to reduce the *MMRE* score on aggregate to 23.98%, in comparison to the *MMRE* of 44.64% of the penalized $\alpha_p - \beta_p - \gamma$ model. For this reason, we further analyze the predictive performance of this model. Figure 4.5 presents a comparison of the measured communication time and predicted communication time with *GBRT-Class C*, for all points in the testing set, normalized to one iteration, in the form of scatterplots, broken down into two sets by communication time for the sake of visibility. The majority of predictions lie within the $\pm 25\%$ error range, while 87.6% of predictions lie within the $\pm 50\%$ error range. A set of 60 points with communication time lower than 1.5 milliseconds lie below the -50% error line. These points correspond to configurations with short message lengths and high core counts, where communication time measurement is noisy, mainly due to on-chip effects, and communication exhibits high time variability, hence our model tends to under-predict these configurations.

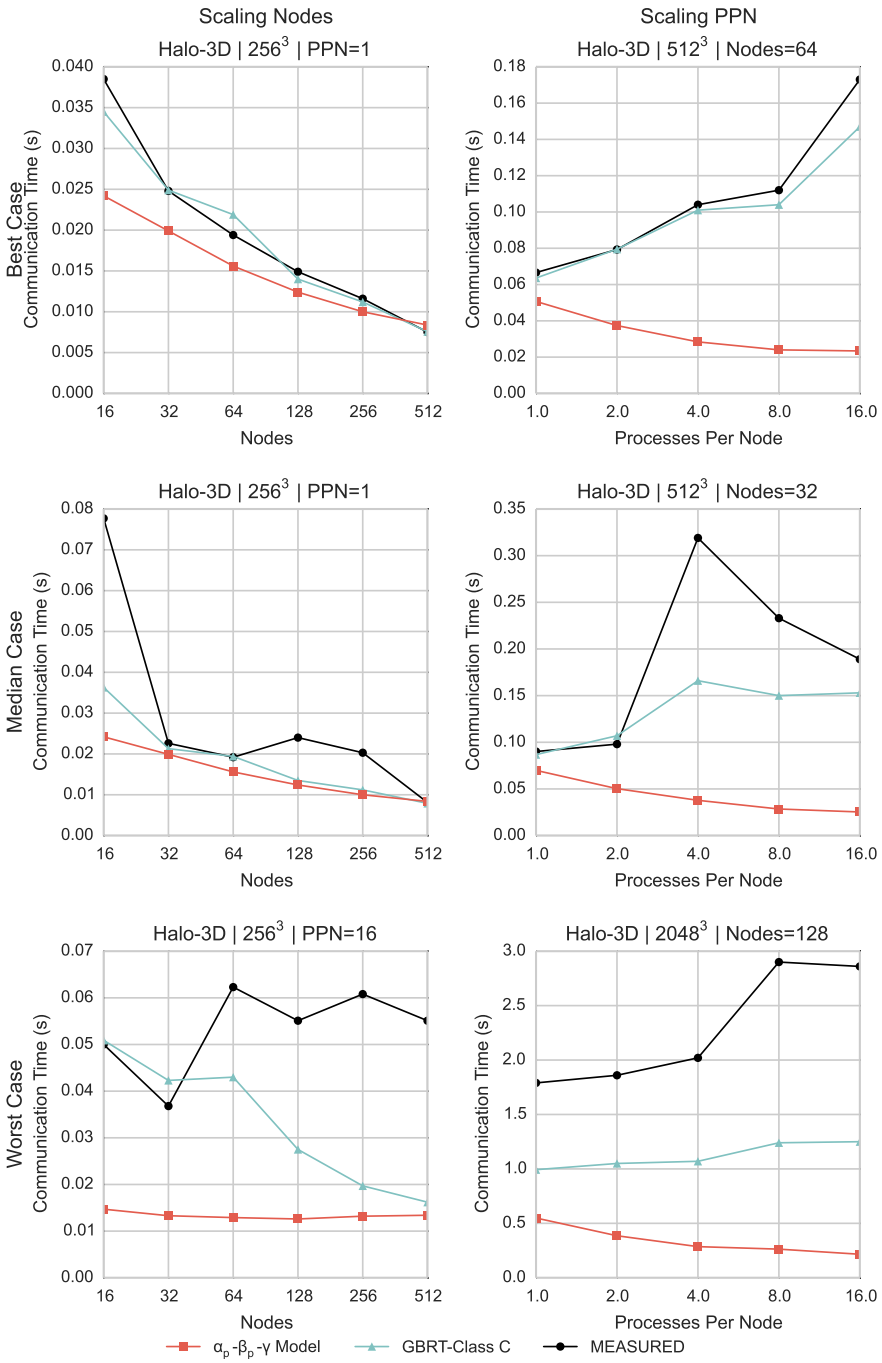


Figure 4.6: Predictions for Halo-3D with *GBRT-Class C* on Vilje.

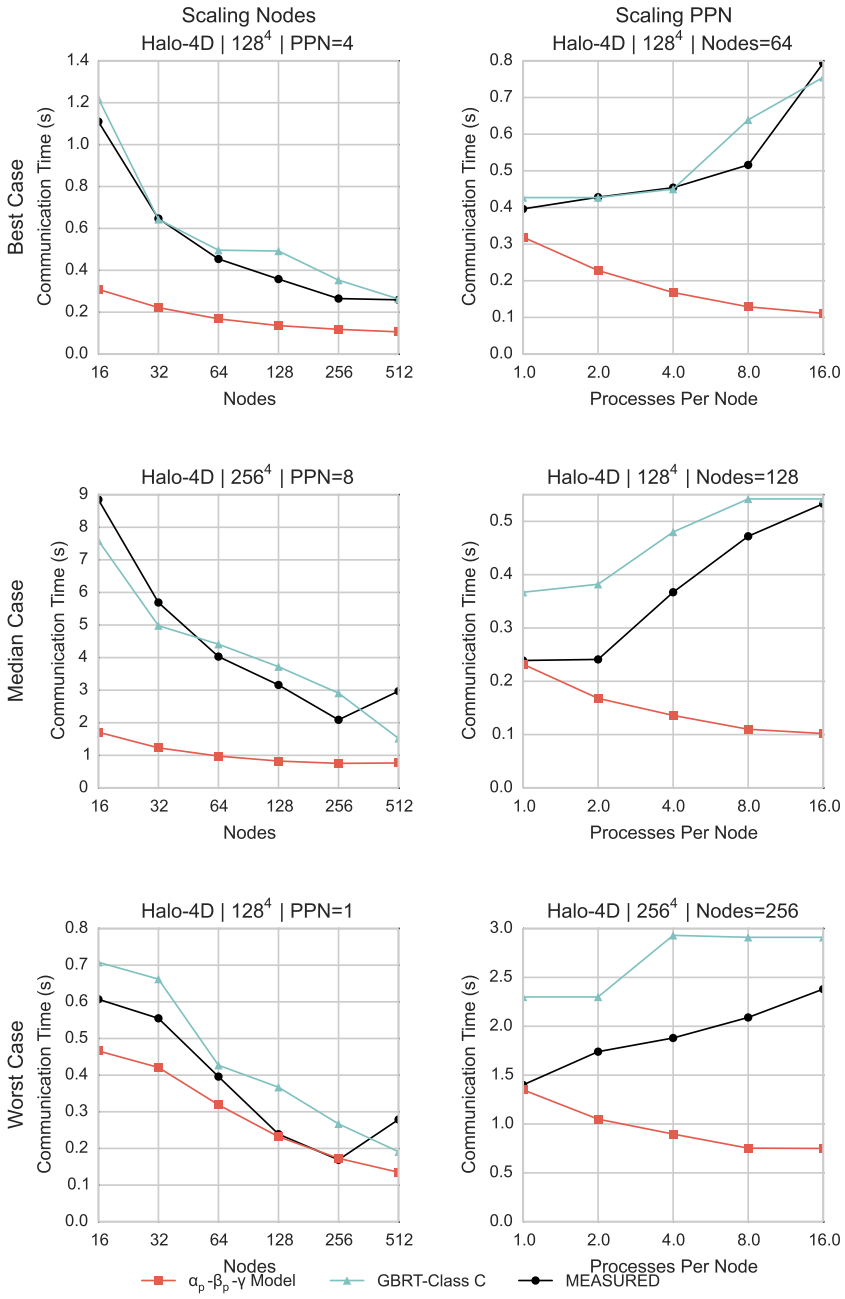


Figure 4.7: Predictions for Halo-4D with *GBRT-Class C* on Vilje.

We then evaluate the *GBRT-Class C* model on the different communica-

tion patterns. For the *Halo-3D* and *Halo-4D* patterns, our testing set includes a multitude of configurations, problem sizes and distinct executions on different allocations. To test the ability of our model to predict communication time scalability when the number of nodes or processes per node is scaled, we utilized the product of the R^2 and *RCC* metrics, i.e., $R^2 \times RCC$, to rank the various subsets of measurements on a scale of 0 to 1, with 1 being the best and 0 being the worst predicted subset. We present the best, median and worst predicted subsets for *Halo-3D* in Figure 4.6 and for *Halo-4D* in Figure 4.7. We refer the reader to Appendix A for the complete set of predictions. We also compare our predictions against predictions with the semi-empirical $\alpha_p - \beta_p - \gamma$ model. For *Halo-3D*, predictions are excellent in the best and median case, both when scaling nodes and when scaling processes per node. We note that the best and median case when scaling nodes refers to the same problem size (256^3) and the same number of processes per node (PPN=1), but to different executions on different node allocations. The worst case presented, when nodes are scaled, corresponds to a small problem size with short messages on a full node (PPN=16), where we have observed extreme increases in communication time, which are not consistent with our benchmark results. We believe this is the outcome of on-node effects (cache sharing and memory effects), which appear due to increased intra-node communication. The $\alpha_p - \beta_p - \gamma$ model fails to predict the scaling behavior of communication when processes per node are scaled, despite the multicore penalty on α. Similarly, when nodes are scaled, the semi-empirical model performs well in our best and median case, where the number of processes per node is 1, but severely under-predicts communication time in our worst case, where the node is full. Regarding Halo-4D in Figure 4.7, our *GBRT-Class C* model predicts communication time with excellent accuracy, even in the worst case, missing, however, a scalability break in the median and worst case, when scaling nodes. The predictive performance of the semi-empirical model is similar to the case of *Halo-3D*: the model fails to capture the way communication time scales with the number of processes per node.

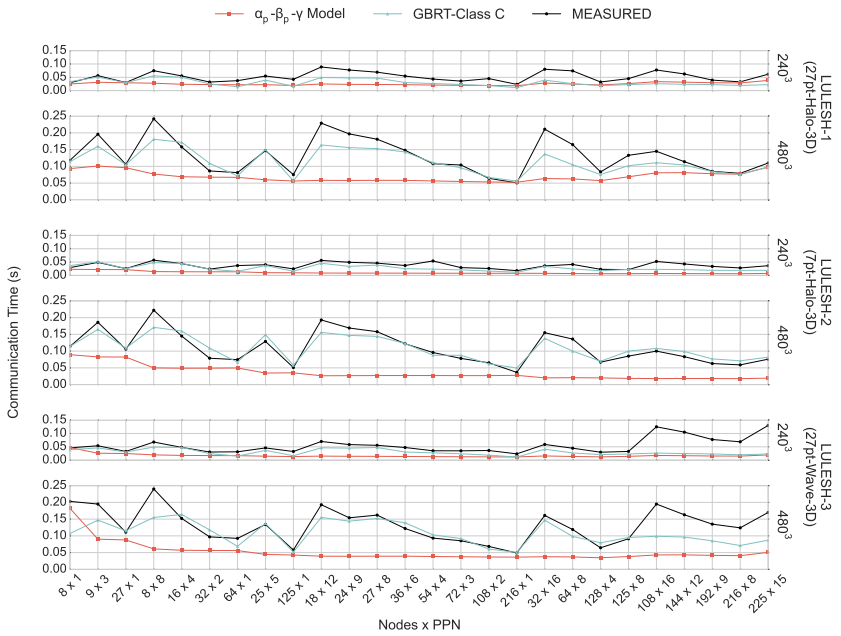


Figure 4.8: Predictions for LULESH with *GBRT-Class C* on Vilje.

In Figure 4.8, we present the predicted communication time for all executed configurations of LULESH, sorted by the number of cores. For all three patterns (*LULESH-1*, *LULESH-2* and *LULESH-3*) and two problem sizes (240^3 and 480^3), our *GBRT-Class C* model predicts communication time with high accuracy up to 216 cores (216 x 1). Configurations up to 512 cores (128 x 4) are also well distinguished by our model. In the case of *LULESH-3*, we observe under-predicted communication times for more than 512 cores. We should note that *LULESH-3* has similar communication volume with *LULESH-1*, sourcing from a different communication graph, processes in *LULESH-3* exchange roughly half the number of messages compared to *LULESH-1*, with double length; yet, communication time rises in *LULESH-3* for more than 512 cores. This behavior is inconsistent with observations in our training set, which explains the shortcoming of our model. We also evaluate the predictions acquired with $\alpha_p - \beta_p - \gamma$ model, which consistently underpredicts the communication time for all configurations with more than one processes per node.

4.4.4 Communication-aware decision-making

The accuracy of a predictive model for communication time is further evaluated by its ability to provide accurate and meaningful predictions in a spe-

Table 4.10: Pareto Fronts for cores and communication time minimization on Vilje

Pattern	Problem size	Execution	Pareto front configurations	Predicted configurations	Matches	Distance (maximum)
Halo-3D	128^3	#1	4	4	3	1
Halo-3D	128^3	#2	5	5	**5**	-
Halo-3D	128^3	#3	5	5	4	1
Halo-3D	256^3	#1	4	6	4	1
Halo-3D	256^3	#2	6	6	**6**	-
Halo-3D	256^3	#3	6	6	**6**	-
Halo-3D	512^3	#1	4	6	4	1
Halo-3D	512^3	#2	6	6	**6**	-
Halo-3D	512^3	#3	6	6	**6**	-
Halo-3D	1024^3	#1	4	6	4	2
Halo-3D	1024^3	#2	6	6	**6**	-
Halo-3D	1024^3	#3	6	6	**6**	-
Halo-3D	2048^3	#1	4	6	4	1
Halo-3D	2048^3	#2	6	6	**6**	-
Halo-3D	2048^3	#3	6	6	**6**	-
Halo-4D	128^4	#1	5	6	5	2
Halo-4D	128^4	#2	5	6	5	1
Halo-4D	256^4	#1	5	8	5	5
Halo-4D	256^4	#2	5	6	5	1
LULESH-1	240^3	#1	2	4	2	3
LULESH-1	480^3	#1	5	5	**5**	-
LULESH-2	240^3	#1	4	4	3	1
LULESH-2	480^3	#1	5	5	**5**	-
LULESH-3	240^3	#1	4	4	3	1
LULESH-3	480^3	#1	5	4	4	1

cific context, in order to enable decision-making. We select our most accurate model, *GBRT-Class C* and use it to predict critical points in decision-making scenarios regarding optimal resource utilization from the side of the user. Supercomputer users utilize large-scale systems with three objectives: a) to maximize the performance of their applications (i.e. to minimize execution time), b) to minimize the consumption of the budget and c) to minimize their waiting times in the job queue. We can argue that these three demands correspond to the quality-of-service demands of the user. The three objectives are contradicting: users tend to request a larger number of cores to minimize execution time, expecting that their application will scale up. At the same time, requesting more cores usually results in longer waiting times, and, if the application does not scale as expected, the result is a waste of cores-hours or node-hours. In addition, supercomputer users request a number of nodes and processes per node, and possibly OpenMP threads, if their application follows the hybrid MPI/OpenMP model, with barely any means to predict which configuration results in the lower execution time. In fact, a common practice among users is to run the application on all possible nodes/processes per node/threads configurations, in order to select the best for performance.

We address these problems from the side of communication by utilizing our *GBRT-Class C* models to predict the optimal nodes/processes per node configurations, namely the configuration that minimizes communication time for a specific number of cores. To achieve this, we formulate the problem as a problem of simultaneously minimizing the number of cores and the communication time and use the notion of Pareto optimality to select the set of Pareto-optimal configurations. This set is known as the Pareto front and marks the optimal set of choices, in the sense that the user is neither better nor worse off by making either of these choices. For a single communication pattern on a specific problem size, assume two different communication configurations, one corresponding to c cores and t communication time, namely (c, t), and the other corresponding to c' cores and t' communication time, namely (c', t'), if $c < c'$ and $t \leq t'$, then the first point is definitely a better configuration than the second, or in other terms, strictly dominates the second point. If, however, $c < c'$ and $t > t'$, we cannot decide whether the first is a better configuration than the second. This set of configurations that do not dominate others constitute the non-dominated Pareto front [Luke, 2009]. The non-dominated Pareto front in our case also signifies the configurations that minimize core-hour consumption due to communication on Vilje. We note that Vilje charges the user on a core-hours policy.

We construct the Pareto fronts based on the predictions acquired with *GBRT-Class C* model for each communication pattern, problem size, and execution and compare the predicted configurations with the equivalent Pareto fronts based on the measured communication time on Vilje. To assess how far away a predicted configuration lies from the actual Pareto front, we use the concept of the Pareto front rank. The first Pareto front we compute has rank equal to 0. If we remove the Pareto front points from the dataset and recompute the Pareto front, we acquire a new, suboptimal Pareto front with rank equal to 1, or else distance $d = 1$ from the original Pareto front. We can iteratively remove points to acquire the i-th Pareto front with distance $d = i$. A point on the i-th Pareto front corresponds to the i-th optimal configuration. We use the distance to evaluate predicted Pareto configurations.

Figure 4.9 shows two examples of the Pareto fronts on Vilje. Each graph shows the measured communication time for all configurations for a specific communication pattern, problem size, and execution on Vilje. The Pareto front and the second-best Pareto front (with distance equal to 1) are denoted on the graph. The first graph in Figure 4.9a demonstrates the case where our predictions are 100% successful: all predicted configurations match the Pareto-optimal configurations, as computed with the measured values for communication time. The second graph in Figure 4.9b demonstrates a case where our model does not accurately predict the Pareto front: 5 out of 6 points predicted

configurations match the Pareto front configurations, however the predicted configurations include an additional point for 512 cores, the configuration of 512 $\times$ 1 (nodes $\times$ processes per node), while in our measurements, the communication time for this particular configuration is higher than the communication time of the previous Pareto point (256 nodes $\times$ 1 process per node) and is thus not included in the Pareto front. This unmatched configuration actually belongs to the Pareto front with distance equal to 2 ($d = 2$). Table 4.10 summarizes the results of the Pareto front points for measured and predicted communication time and cores. We refer the reader to Appendix A for the detailed evaluation. We denote as "Matches" the number of predicted configurations that match with the configurations of the Pareto front. In cases where there exist mismatched predicted configurations, we assign each mismatched configuration a rank d that corresponds to the distance of the suboptimal Pareto front to which it belongs. We report the maximum distance among mismatched points. For the 25 constructed Pareto fronts, measured and predicted Pareto fronts match in 11 cases, while in the remaining cases, the fronts differ by at most 3 points (see *Halo-4D* on a 256^4 problem size - execution #1). For 10 of the remaining cases, the maximum distance is 1, thus the mispredicted configurations correspond to the second best optimal configurations. For reasons of comparison, we constructed the predicted Pareto fronts using predictions acquired with the $\alpha_p - \beta_p - \gamma$ model, and observed zero matches between the 25 measured and predicted Pareto fronts.

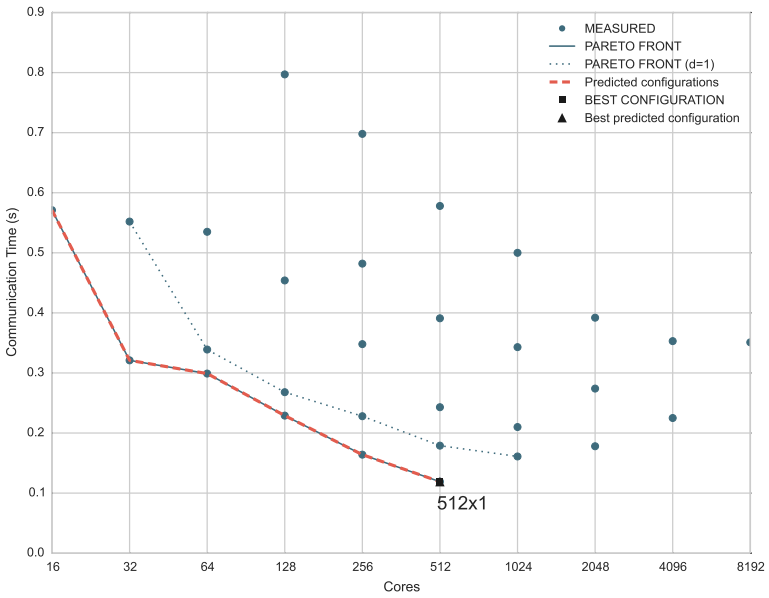
An interesting observation from the Pareto front construction is that, on Vilje, for any number of cores and for all communication patterns, the optimal configuration never utilizes more than 1 process per node. Although the configurations we examine are only optimal for communication time, we can extract two useful conclusions for resource allocation on Vilje. First, since the system bills the user on the basis of core-hours, if the computation of the application is not affected by co-running applications, users should consider selecting configurations with a single process per node, to minimize their core-hour consumption. The system could then co-locate other applications on the remaining cores of the node. Second, if the users need to select a configuration involving OpenMP threads, they should select 1 process per node for MPI and fill the remaining cores of the node with OpenMP processes, or otherwise, select the lowest possible number of processes per node, depending on the scalability of the computation of the application.

The Pareto fronts for cores and execution time offer additional interesting information. The point with the highest number of cores included in the Pareto front corresponds to the configuration that leads to the minimum communication time. If the communication pattern and/or problem size under consideration is not scalable, then the point with the highest core count in the Pareto

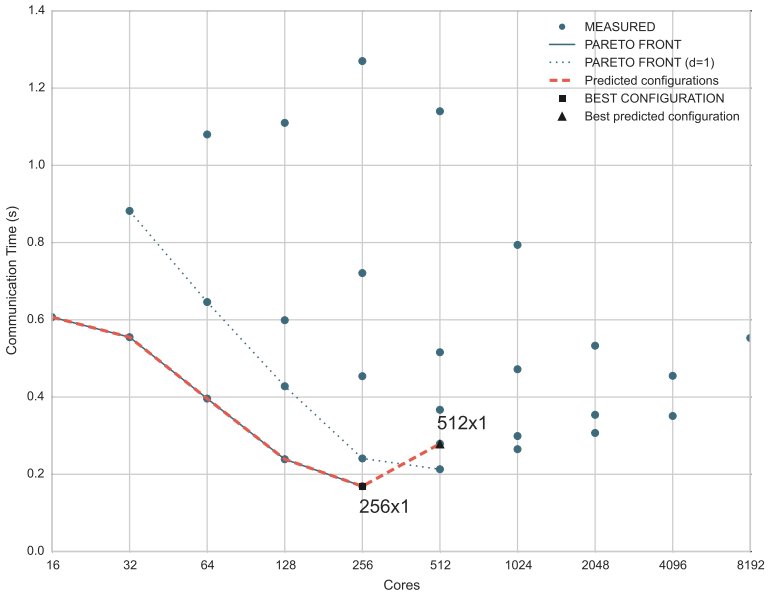
Table 4.11: Minimum communication time configurations on Vilje

Pattern	Problem size	Execution	Measured ($n \times ppn$)	Predicted ($n \times ppn$)	Result	Distance
Halo-3D	128^3	#1	512×1	128×1	False	1
Halo-3D	128^3	#2	256×1	256×1	**True**	-
Halo-3D	128^3	#3	512×1	256×1	False	1
Halo-3D	256^3	#1	512×1	512×1	**True**	-
Halo-3D	256^3	#2	512×1	512×1	**True**	-
Halo-3D	256^3	#3	512×1	512×1	**True**	-
Halo-3D	512^3	#1	512×1	512×1	**True**	-
Halo-3D	512^3	#2	512×1	512×1	**True**	-
Halo-3D	512^3	#3	512×1	512×1	**True**	-
Halo-3D	1024^3	#1	512×1	512×1	**True**	-
Halo-3D	1024^3	#2	512×1	512×1	**True**	-
Halo-3D	1024^3	#3	512×1	512×1	**True**	-
Halo-3D	2048^3	#1	512×1	512×1	**True**	-
Halo-3D	2048^3	#2	512×1	512×1	**True**	-
Halo-3D	2048^3	#3	512×1	512×1	**True**	-
Halo-4D	128^4	#1	256×1	512×1	False	2
Halo-4D	128^4	#2	512×1	512×1	**True**	-
Halo-4D	256^4	#1	256×1	512×8	False	5
Halo-4D	256^4	#2	256×1	512×1	False	1
LULESH-1	240^3	#1	216×1	216×1	**True**	-
LULESH-1	480^3	#1	216×1	216×1	**True**	-
LULESH-2	240^3	#1	216×1	216×1	**True**	-
LULESH-2	480^3	#1	216×1	216×1	**True**	-
LULESH-3	240^3	#1	216×1	216×1	**True**	-
LULESH-3	480^3	#1	216×1	216×1	**True**	-

front corresponds to the maximum number of cores that can be deployed before communication scalability breaks. The identification of communication scalability breaks is particularly useful for the user, who can leverage this information and combine it with any information on the scalability of computation, so as to select the maximum number of cores that they should use for the execution of their application. We annotate the measured and predicted optimal (best) configurations in Figure 4.9. Table 4.11 shows the measured and predicted (with *GBRT-Class C*) minimum time configurations for our communication patterns on Vilje. We use the notion of Pareto front distance to assess the mispredicted configurations: we compute the distance of the Pareto front on which the mispredicted configuration belongs. Our model correctly predicts the configuration that results in the minimum communication time for 20 out of the 25 cases. Wrong predictions mostly occur for the *Halo-4D* pattern, where, although our model shows small relative errors, it tends to underpredict communication time and therefore predicts the minimum communication time configuration at a higher number of cores than it actually occurs. This pattern is the only one where the predicted best configuration lies on the 6th Pareto front (distance equals 5). In the same scenario, the $\alpha_p - \beta_p - \gamma$ model predicts correctly only 5 out 25 configurations for the minimum communication time.



(a) All points match: *Halo-3D* on a 1024^3 problem size (execution #2).



(b) 5 out of 6 points match: *Halo-4D* on a 128^4 problem size (execution #1).

Figure 4.9: Examples of predicting Pareto-optimal configurations for communication time on Vilje.

5

Predictive communication modeling on *Piz Daint*

5.1 Introduction

In this chapter, we present the application of our predictive communication modeling methodology (see Sections 3.5 and 3.6) on the *Piz Daint* supercomputer. We describe the model building steps with statistical and machine learning methods and evaluate the predictive ability of our models on various communication patterns. We compare the predictive performance of our models and evaluate the best performing model in detail. We additionally use our models in communication-aware decision-making scenarios targeting the optimization of resource utilization on Piz Daint.

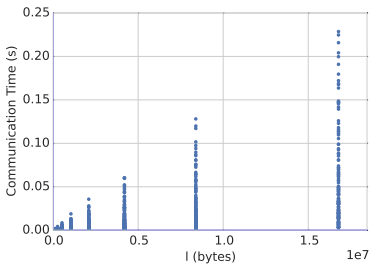
5.2 Modeling with statistical learning

Following the statistical modeling methodology presented in Chapter 3, we first constructed multiple variable linear regression models on Piz Daint. For these models, we utilized features from Classes A and B, as well as the six features from Class C that sketch the allocation shape on Piz Daint: the maximum number of nodes per Aries SoC (n/a), the maximum number of Aries SoCs on chassis (a/c), the maximum number of chassis per dragonfly group (c/g), and the numbers of Aries SoCs, chassis and groups (a, c and g). Similarly to Vilje, for all features that refer to data and messages, we only used their average value. We also substituted the features for intra-node traffic with the percentage of internode to total traffic. This preliminary feature selection reduced the number of our features to 16.

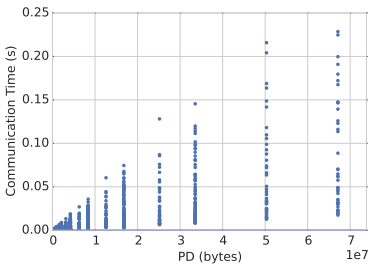
For the training set collection, we executed our benchmark on Piz Daint

Table 5.1: Correlation coefficients between features and communication time on Piz Daint - based on benchmark data

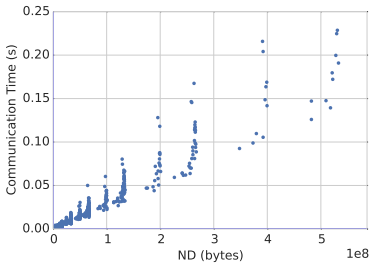
Feature	Correlation	Feature	Correlation
l	0.64	NM	0.04
PM	−0.059	TM	−0.04
ppn	0.1434	n/a	0.06
n	−0.08	a/c	−0.06
I	−0.02	c/g	−0.05
PD	0.7741	a	−0.08
ND	0.9703	c	−0.05
TD	0.80	g	−0.04



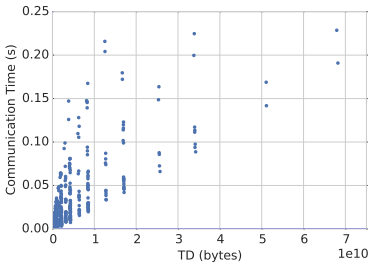
(a) l and communication time



(b) PD and communication time



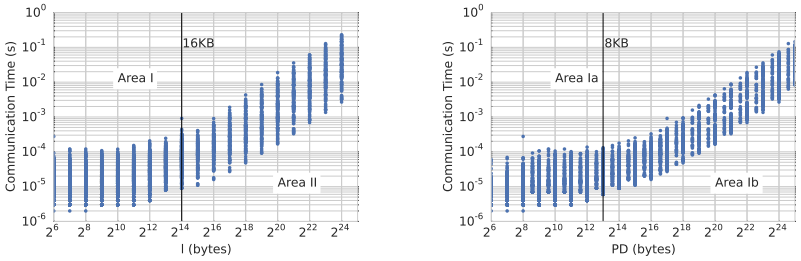
(c) ND and communication time



(d) TD and communication time

Figure 5.1: Scatterplots for features that exhibit high correlation with communication time on Piz Daint

for various configurations, from 8 to 128 nodes, 1 to 8 processes per node, 1 to 4 messages per process and various message sizes, from 64 bytes to 16 megabytes. As on Vilje, we repeated the benchmarking for all configurations, in order to collect diverse values for Class C features that refer to the allocation shape, acquiring a training set of 3040 points. We then computed the Pearson's correlation coefficients between each feature and communication time, shown



(a) Partitioning to Areas I and II **(b)** Partitioning to Areas Ia and Ib

Figure 5.2: Partitioning the parameter space into sub-areas on Piz Daint

Table 5.2: Predictive model for Area Ia on Piz Daint: Terms and coefficients

Terms	Coefficients
Intercept	1.816×10^{-5}
NM	5.0144×10^{-8}
n/a	-2.927×10^{-6}
$NM \times n/a$	3.2788×10^{-8}
c	-6.4923×10^{-7}
$NM \times c$	1.3872×10^{-8}
$n/a \times c$	2.9738×10^{-7}
$NM \times c \times n/a$	-1.8538×10^{-9}

in Table 5.1, using our benchmark data. We noted a very high correlation for Node Data *ND* and Total Data *TD*, as well as high correlations for Process Data *PD* and the message length *l*, similarly to Vilje. To identify linear relationships or the absence thereof between the four features and communication time, we plotted the scatterplots, shown in Figure 5.1, which clearly show linear relationships. We then inspected scatterplots in logarithmic scale, where we identified changes in the slope of the linear curves for all features. We thus partitioned our space of parameters, first based on the message length *l* and the change in slope indicated in Figure 5.2a, into Areas I ($l \leq 8kilobytes$) and II ($l > 8kilobytes$). We further partitioned Area I in two sub-areas, based on the change in the slope of the *PD*, as shown in Figure 5.2b: Area Ia ($PD \leq 8kilobytes$) and Area Ib ($PD > 8kilobytes$).

For each of the three areas, we re-examined the correlation between features and communication time and began model construction using the highest-correlated features in each area, resulting in the following linear models with additive terms:

- Model Ia: $t_{comm} = b_0 + b_1 \times NM$

Table 5.3: Predictive model for Area Ib on Piz Daint: Terms and coefficients

Terms	Coefficients
Intercept	2.0604×10^{-5}
ppn	-1.7625×10^{-5}
ND	1.9318×10^{-10}
NM	2.7981×10^{-7}
a/c	7.5479×10^{-7}
$ppn \times ND$	5.3778×10^{-13}
$ppn \times NM$	-1.9800×10^{-8}
$ND \times a/c$	2.7822×10^{-12}

Table 5.4: Predictive model for Area II on Piz Daint: Terms and coefficients

Terms	Coefficients
Intercept	7.0526×10^{-5}
ppn	-2.7899×10^{-5}
ND	2.2196×10^{-10}
$ND \times ppn$	1.4061×10^{-11}

- Model Ib: $t_{comm} = b_0 + b_1 \times ND + b_2 \times ND$
- Model II: $t_{comm} = b_0 + b_1 \times ND$

As on Vilje, extending these models with additional terms did not improve the model accuracy and turned us into searching for interaction terms. Figure 5.1 indicated high heteroskedasticity for all terms and communication time. To determine interactions, we took the same steps as on Vilje: we excluded features which are products of other features and show inter-correlations, we computed correlation coefficients to perform an initial selection of additional features and we chose to include at least one feature referring to the allocation shape. We experimented with various model forms and finally trained our models, which are presented, along with their coefficients in Tables 5.2, 5.3 and 5.4. The models were trained using robust regression of the *LinearModel* class in Matlab R2013a. Note that the predictive model for Area II only includes three terms, *ND* and *ppn* and their product, and no feature from Class C. For this particular area, the two features and their product were sufficient for model accuracy and adding features from the allocation shape did not improve the model's accuracy or goodness-of-fit.

The final models help us to draw conclusions on the impact of the architecture of Piz Daint on communication performance. Node injection is critical to performance: features *ND* and *NM* that sketch outgoing node traffic are in-

cluded in the models for all three areas. For Area Ia, where both the message length and the number of messages per process are small (due to the parameter space partitioning using *PD*), the number of messages injected by each node is more important than the message length. In addition, the density or sparsity of the allocation of nodes on Aries SoCs is important, as indicated by the presence of feature *n / a*, and, similarly to Vilje, for this area, the allocation shape and hence the distance (number of hops) are critical to performance. For Area Ib, where the number of messages is small but processes produce more traffic, communication performance is also affected by the number of processes per node *ppn*, as a contention factor, as well as by the number of network components, i.e., Arias SoCs *a*, used by the allocation. Finally, in Area II, performance is primarily influenced by the data injected into the network by each node and the number of processes per node. Due to the larger message sizes in this area, any congestion effects occur on the interconnection network.

Besides the multiple variable linear regression model, we followed the process described in Section 3.5 to construct a single variable model with statistical learning. We first computed Pearson's correlation coefficients for all features in all three classes (A, B and C) and selected the feature with the highest correlation. On Piz Daint, the highest-correlated feature is *AD (max)*. It is noteworthy that this feature is equivalent to the highest-correlated feature on Vilje, *SD (max)*. Both features belong to Class C and highlight traffic through the core networking component of each interconnection network, which, in the case of Piz Daint, is the Aries SoC. Using the selected feature, we built and trained 7806 models, using non-linear transformations of the feature, as we described in Section 3.5. We promoted the best fitting model on Piz Daint to be the following, hereafter denoted as *LinReg-MV*:

$$t = 7.939\times10^{-6}+1.183\times10^{-7}\times\sqrt{x}-1.278\times10^{-15}\times log_2^3 x+5.151\times10^{-15}\times x log_2^3 x$$
(seconds), where $x = AD(max)$ in bytes

5.3 Modeling with machine-learning

To construct machine-learning predictive communication models on Piz Daint, we applied the machine-learning methodology described in Section 3.6 and built three predictive models for communication time on Piz Daint with Gradient Boosting Regression Trees. Each model uses different classes of features: model *GBRT-Class A* uses only Class A features, model *GBRT-Class B* uses Class A and Class B features and model *GBRT-Class C* uses features from all classes. As on Vilje, we only use the maximum value for features *l*, *PD*, and *PM*,

Table 5.5: Parameter space for benchmark executions on Piz Daint for modeling with machine learning

Parameter	**Range**	
n	8-128	8-256
ppn	1-8	1-8
l	16B-16MB	16B-16KB
PM	1-4	1-4
#executions	3	4
#points	6912	

since for all points in our training set, the minimum, average and maximum values for these features are equal, due to the design of our benchmark. We did not perform any manual feature selection but applied systematic feature selection for the *GBRT-Class C* with support vector regression and recursive feature elimination.

To train the machine-learning models on Piz Daint, we augmented the training set used in our statistical learning methodology with a full collection of benchmark data for all message sizes, number of messages and configurations. Additionally, we collected extra data points for short messages, for which we observed variance in communication time, in order to boost the accuracy of the GBRT method in this area. The observed variance on Piz Daint stems from the adaptive routing on the Cray Aries interconnect, which can route packets through non-minimal paths. The effect of the adaptive routing is stronger on short messages, as it can alter their latency. The resulting training set included 6912 data points, derived from benchmarking on various numbers of nodes, cores, and configurations. We present the specifics of the training set in Table 5.5.

As the GBRT method needs to be tuned for a number of parameters, we test the method with multiple different values for these parameters, as well as different numbers of features for the Class C model and selected the model with the best fit on the training set. We present the range of values for which we tested the method, along with the selected values, in Table 5.6. The final predictive models result from the training of GBRT model with our training set, after setting the number of features and the method parameters with the selected values.

The GBRT method ranks the features of each model on a scale from 0 to 1, based on their impact on the output. This ranking only reflects the usage of the features by the model: a feature with a high score is used more often or at a higher level of the decision trees, to split the training set. However, we

Table 5.6: Tested range of values and selected values for the parameters of the GBRT method and number of features on Piz Daint

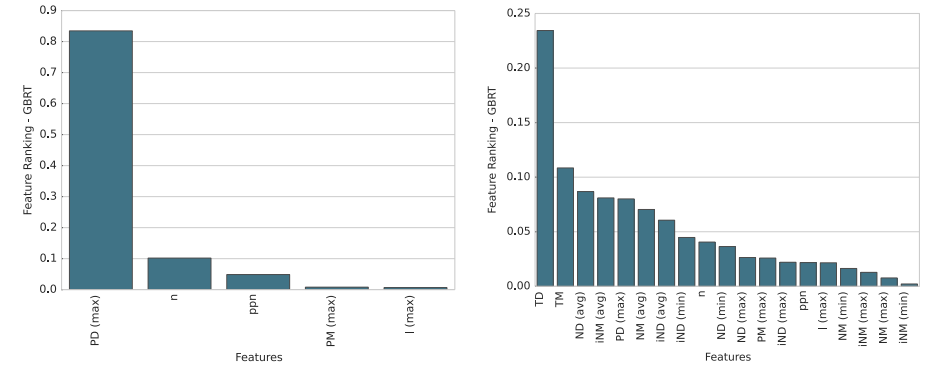
Parameter	Range	Selected Value
n_estimators	100 - 2000	1000
learning_rate	0.001 - 1	0.009
min_samples_leaf	1 - 5	2
min_samples_split	2 - 8	3
max_depth	3 - 8	7
features_classA	-	5
features_classB	-	19
features_classC	15 - 40	23

examine the feature ranking for each of our three models to gain an understanding of the predictive ability and limitations of our models. We present the feature ranking for GBRT models on Piz Daint in Figure 5.3. For model *GBRT-Class A*, the highest ranked feature is Process Data (*PD*), followed by *n* and *ppn* that determine the allocation size. Moving from Class A to Class B to Class C, the method does not ignore the additional features. For*GBRT-Class B*, the most important feature is Total Data (*TD*), however, it does not dominate the model building process, as it scores less than 0.25. For Class C, the highest ranked feature is again *PD*, although scoring less than 0.15. Overall, the feature ranking for *GBRT-Class C* on Piz Daint demonstrates that decision trees take into account the traffic through various points of the network at more levels in all depths of the trees. However, only the amount of data matters for the model: features that refer to numbers of messages are pruned by feature selection. Overall, on Piz Daint, GBRT feature ranking does not match ranking with Pearson's correlation coefficients; multiple Class B and C features are taken into account by the corresponding models with almost equal importance.

5.4 Evaluation

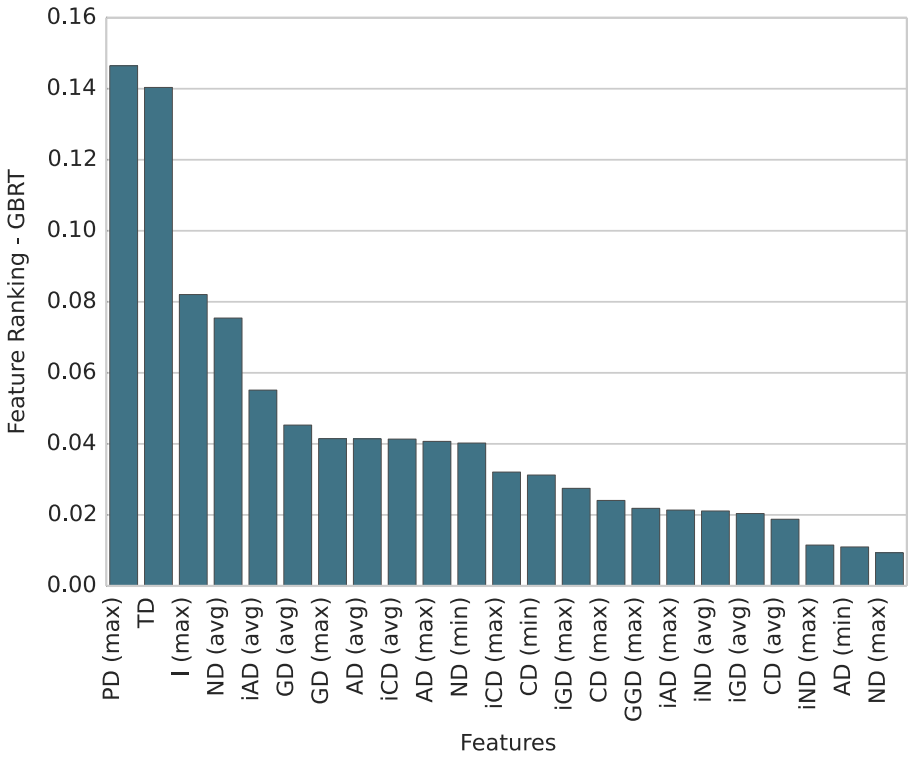
5.4.1 Communication patterns

To evaluate our predictive models, we used a testing set similar to that of Vilje. We experimented with the *Halo-3D* exchange pattern (6 2D-face exchanges with neighbors on a 3D cartesian grid) and the *Halo-4D* exchange pattern (8 3D-face exchanges with neighbors on a 4D cartesian grid). We also experimented with the point-to-point communication phases of the LULESH proxy application: *LULESH-1*, a 27-point 3D-halo exchange, *LULESH-2*, a 6-point



(a) GBRT-Class A

(b) GBRT-Class B



(c) GBRT-Class C

Figure 5.3: Feature ranking for the GBRT models on Piz Daint.

3D-halo exchange and *LULESH-3*, a 27-point wavefront. We refer the reader to Section 4.4.1 for more information on the patterns and applications.

We predict communication time for *Halo-3D*, *Halo-4D* and LULESH for various problem sizes and execution configurations, as well as for multiple executions, in order to test the ability of class C features to describe the effects

Table 5.7: Details of the testing set on Piz Daint

	Piz Daint		
Pattern	Halo-3D	Halo-4D	LULESH
Domain Size	128^3, 256^3, 512^3, 1024^3, 2048^3	128^4, 256^4	240^3, 480^3
Iterations	256	256	100
n	16-1024	16-1024	8-1000
ppn	1-8	1-8	1-8
#executions	3	2	1
#points	613		

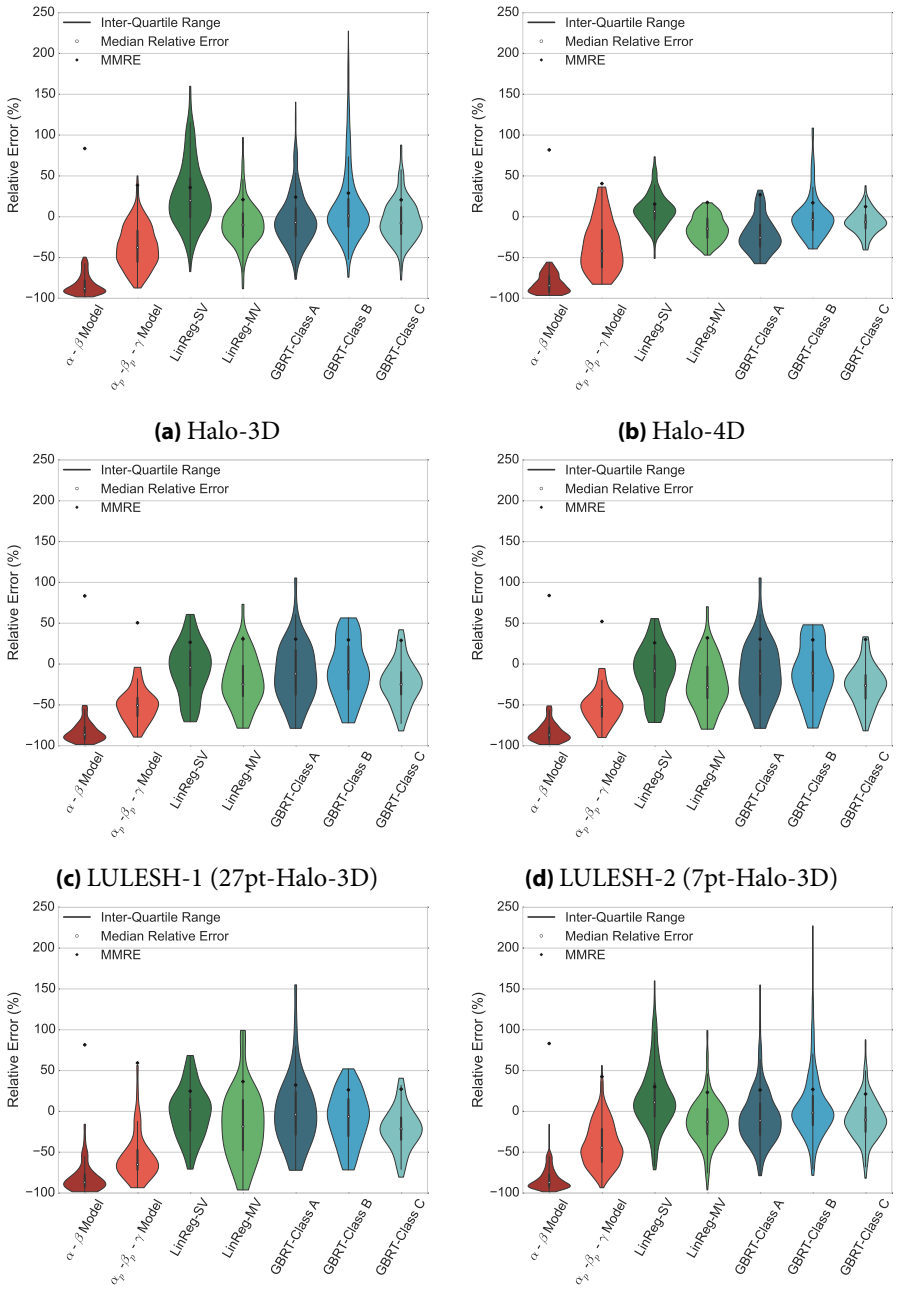
Table 5.8: Measured parameters for the analytical and semi-empirical models on Piz Daint

Parameter	**Value**
α	0.238 us
β	0.114 ns
γ	0.453 us

of distinct allocations. The specifics of the testing set for Piz Daint is given in Table 5.7. LULESH by design expects the number of processes to be a power of 3, so all execution configurations for LULESH are bound by this constraint.

5.4.2 Model comparison

We begin our evaluation by comparing the predictive performance of our models, using the metrics defined in Section 3.2. We evaluate the two models built with statistical learning, i.e., *LinReg-SV* and *LinReg-MV* and the three models built with machine-learning, i.e., *GBRT-Class A*, *GBRT-Class B* and *GBRT-Class C*. We also compare the predictive performance for the analytical $\alpha - \beta$ model, denoted as $\alpha - \beta$ *Model*, and the semi-empirical $\alpha - \beta - \gamma$ model with penalties on the α and β parameters, denoted as $\alpha_p - \beta_p - \gamma$ *Model*, which we promoted as the best-fitting semi-empirical model on Piz Daint. The measured values for the α, β and γ parameters are given in Table 5.9. The penalty on the α parameter corresponds to a multi-core penalty, while the penalty on the β parameter is a penalty for limited bandwidth and network contention.



(a) Halo-3D

(b) Halo-4D

(c) LULESH-1 (27pt-Halo-3D)

(d) LULESH-2 (7pt-Halo-3D)

(e) LULESH-3 (27pt-Wave-3D)

(f) Aggregate

Figure 5.4: Model comparison on Piz Daint.

Table 5.9: Model comparison through accuracy and goodness-of-fit metrics on Piz Daint

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.00	0.092	0.856
$\alpha_p - \beta_p - \gamma$ *Model*	30.47	0.466	0.916
LinReg-SV	48.33	0.945	0.934
LinReg-MV	68.81	0.976	0.933
GBRT-Class A	61.90	0.942	0.927
GBRT-Class B	64.76	0.979	0.920
GBRT-Class C	68.33	0.771	0.940

(a) Halo-3D

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.00	-0.187	0.828
$\alpha_p - \beta_p - \gamma$ *Model*	31.25	0.297	0.884
LinReg-SV	82.14	0.864	0.948
LinReg-MV	71.42	0.962	0.953
GBRT-Class A	46.42	0.657	0.937
GBRT-Class B	78.57	0.982	0.949
GBRT-Class C	89.28	0.983	0.967

(b) Halo-4D

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.00	-0.770	0.733
$\alpha_p - \beta_p - \gamma$ *Model*	12.50	0.083	0.825
LinReg-SV	58.92	0.643	0.810
LinReg-MV	42.86	0.605	0.862
GBRT-Class A	48.21	0.281	0.841
GBRT-Class B	44.64	0.692	0.832
GBRT-Class C	44.64	0.796	0.850

(c) LULESH-1

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.00	-0.784	0.733
$\alpha_p - \beta_p - \gamma$ *Model*	8.93	0.060	0.825
LinReg-SV	60.71	0.637	0.806
LinReg-MV	44.64	0.619	0.862
GBRT-Class A	48.21	0.281	0.841
GBRT-Class B	44.64	0.701	0.827
GBRT-Class C	42.85	0.792	0.852

(d) LULESH-2

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	1.786	-0.623	0.729
$\alpha_p - \beta_p - \gamma$ *Model*	10.714	-0.053	0.813
LinReg-SV	58.93	0.824	0.834
LinReg-MV	41.07	0.097	0.845
GBRT-Class A	48.21	-0.420	0.853
GBRT-Class B	51.79	0.753	0.833
GBRT-Class C	53.57	0.800	0.858

(e) LULESH-3

	$Pred_{0.25}$ (%)	R^2	*RCC*
$\alpha - \beta$ *Model*	0.14	0.182	0.874
$\alpha_p - \beta_p - \gamma$ *Model*	25.86	0.515	0.905
LinReg-SV	56.43	0.912	0.927
LinReg-MV	63.00	0.974	0.930
GBRT-Class A	56.14	0.788	0.929
GBRT-Class B	62.71	0.987	0.924
GBRT-Class C	66.57	0.986	0.940

(f) Aggregate

Figure 5.4 shows the distribution of the relative errors of predictions, for all models on each communication pattern, as well as on aggregate (in Figure 5.4f), in the form of violinplots, while Table 5.9 summarizes the scores of the $Pred_{0.25}$, R^2 and *RCC* metrics. First, both the analytical and the semi-empirical model underpredict communication, with the latter improving predictions upon the first. We should note that the semi-empirical $\alpha_p - \beta_p - \gamma$ model has better accuracy and goodness-of-fit on Piz Daint, in comparison with Vilje, as indicated by its $Pred_{0.25}$ and R^2 scores for all patterns. Thus, the semi-empirical model manages to capture some of the topology properties of Piz Daint.

Regarding our statistical learning models, model *LinReg-SV* achieves remarkable accuracy, especially for the *Halo-4D* pattern, as the selected feature, *AD (max)*, that corresponds to traffic through an Aries SoC, highly characterizes the traffic pattern and communication performance on Piz Daint. Unlike Vilje, on Piz Daint, node allocations are more dense and usually an Aries SoC is used by a single job (i.e. an allocation for a single application). Thus, the *AD (max)* feature captures all, or at least most of, the traffic through the SoC and is particularly meaningful for the performance of patterns that can cause congestion on the Aries SoC, as is the *Halo-4D* pattern. The multiple variable linear regression model, *LinReg-MV*, achieves high accuracy on patterns that are more regular in the lengths of messages, as are *Halo-3D* and *Halo-4D*, but shows high goodness-of-fit across all patterns, as indicated by its high *RCC* scores. Overall, its accuracy is comparable to that of *GBRT-Class B*, which relies on the same features for traffic. *GBRT-Class A* scores decently in all patterns and on aggregate. Nevertheless, its accuracy could be improved by by augmenting the training set with data from irregular patterns, since its set of features and the generic benchmarking prevent the model from capturing more complex communication graphs, as are those of *LULESH-1* and *LULESH-3*. The best fitting model on Piz Daint is *GBRT-Class C*, which outperforms all other models on aggregate and exhibits excellent goodness-of-fit, while constraining the range of relative errors, contrary to *GBRT-Class B*, which also performs well, but results in a few extreme over-predictions for *Halo-3D* and *Halo-4D*.

5.4.3 Detailed evaluation

Following the comparison of the predictive models, we select *GBRT-Class C* as the best model for Piz Daint, achieving very high scores on all metrics and the smallest error ranges. Its time of prediction is just-in-time before the execution of the application, its accuracy and goodness-of-fit is very high, scoring 66.57% in $Pred_{0.25}$, 0.986 in R^2 and 0.940 in *RCC*. Moreover, *GBRT-Class C* reduces the *MMRE* score on aggregate to 21.32%, from 42.60% of the semi-empirical $\alpha_p - \beta_p - \gamma$ model. We thus further analyze the predictive performance of this model. First, we examine all points in the testing set, in the form of a scatterplot, in Figure 5.5, normalized to a single iteration, broken in two sets by communication time, for the sake of visibility. The majority of predictions lie within a $\pm 25\%$ error range, while 92.14% of predictions have relative errors within the $\pm 50\%$ range and an insignificant number of outliers occur only when communication time is less than 100us. The scatterplots show that no systematic predictive errors appear on Piz Daint: the model has good predictive ability for all communication patterns.

Subsequently, we evaluate the *GBRT-Class C* model on each of the com-

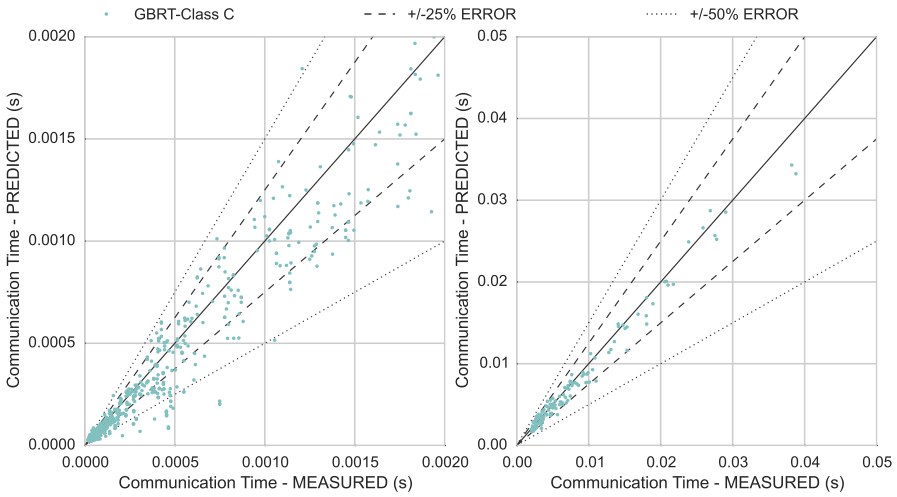


Figure 5.5: Scatterplots for predictions with *GBRT-Class C* on Piz Daint. Communication time is normalized to a single iteration.

munication patterns. As out testing set for *Halo-3D* and *Halo-4D* contains multiple configurations, problem sizes and distinct executions, we utilize the product of R^2 and *RCC*, i.e., $R^2 \times RCC$, to rank the various subsets of measurements in a scale of 0 to 1, with 1 being the best predicted subset and 0 being the worst predicted subset. We show the scalability predictions for the two patterns, when nodes are scaled and when processes per node are scaled, for the best, worst and median case, in Figure 5.6 for *Halo-3D* and in Figure 5.7 for *Halo-4D*. We refer the reader to Appendix B for the full evaluation. We also compare our predictions with the *GBRT-Class C* model against the predictions with the semi-empirical $\alpha_p - \beta_p - \gamma$ model. Focusing on *Halo-3D*, our model provides excellent predictions when scaling nodes. Even in the worst case, it is able to capture the scalability trend of the communication pattern. Similarly, when scaling the number of processes per node, our model results in small errors in all three cases. Notably, the $\alpha_p - \beta_p - \gamma$ model manages to capture the scalability behavior when the number of processes per node is scaled, in contrast to Vilje, where the effect of multiple processes per node was not reflected by this model. However, in most cases, it results to higher errors in comparison to our model. For *Halo-4D* in Figure 5.7, our *GBRT-Class C* model accurately captures the scalability of the pattern even in the worst case, although with some errors, while the $\alpha_p - \beta_p - \gamma$ model fails to capture communication performance when scaling the number of processes per node for this pattern.

Finally, in Figure 5.8, we present the predicted communication time for the three point-to-point patterns of LULESH, for all tested configurations, sorted

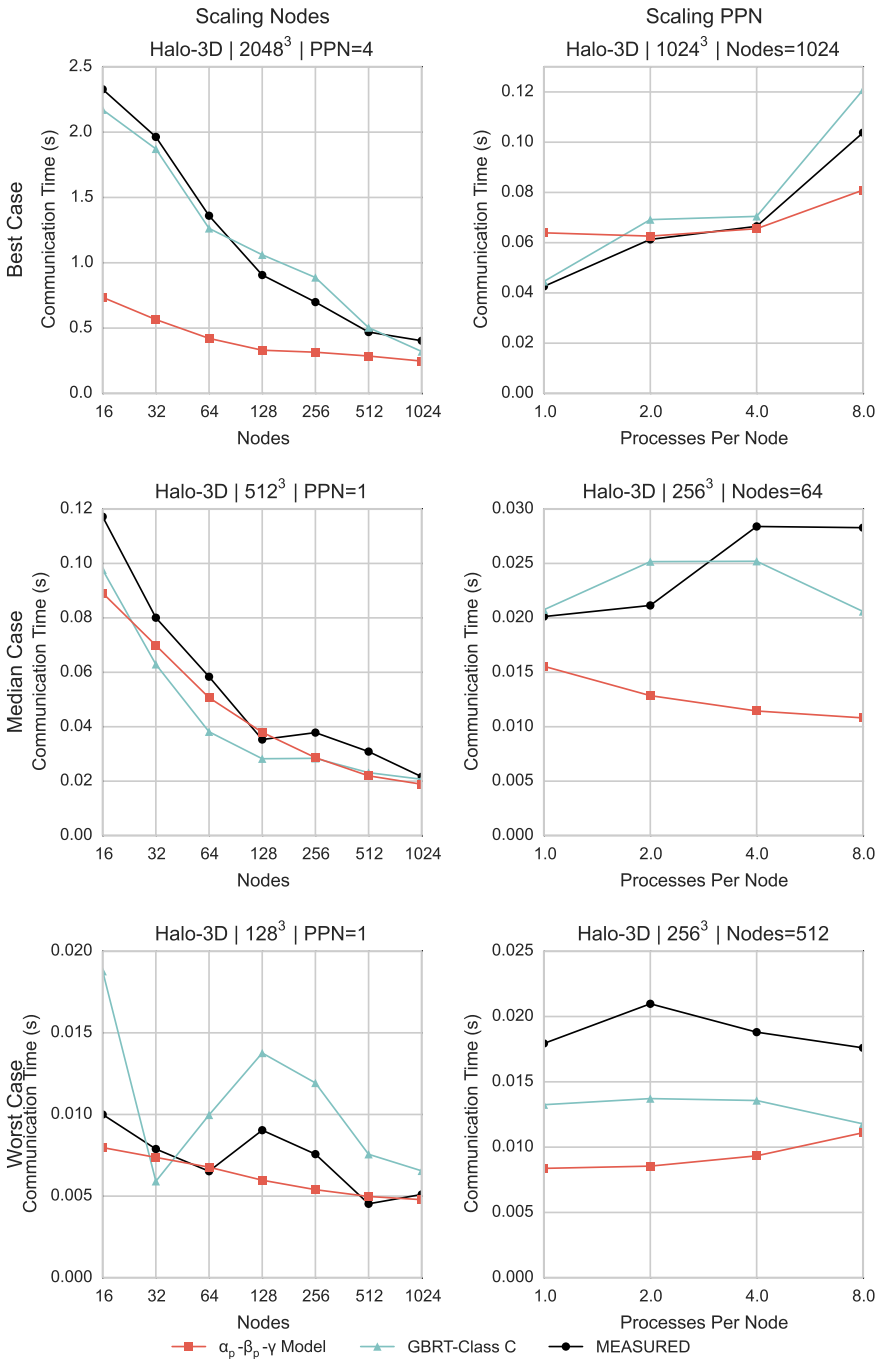


Figure 5.6: Predictions for Halo-3D with *GBRT-Class C* on Piz Daint.

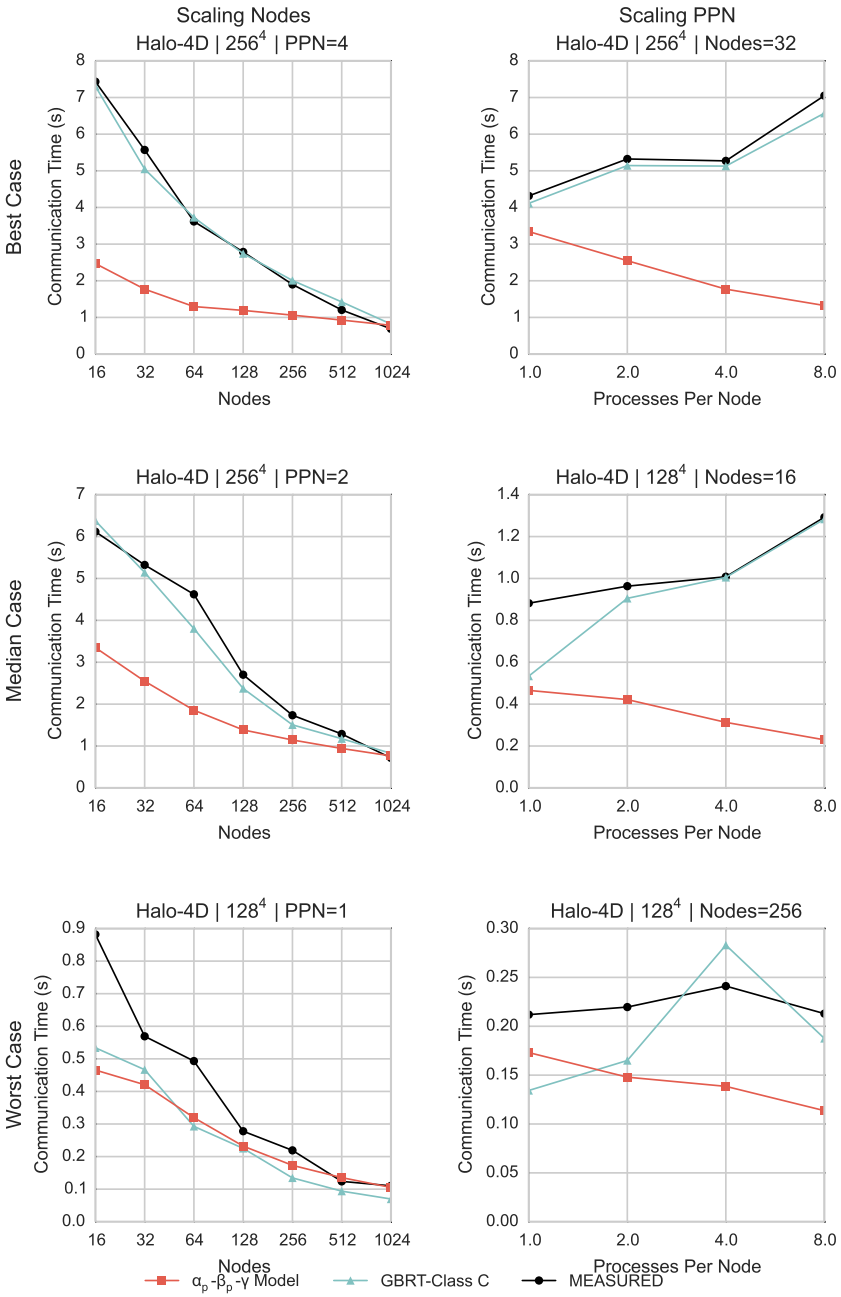


Figure 5.7: Predictions for Halo-4D with *GBRT-Class C* on Piz Daint.

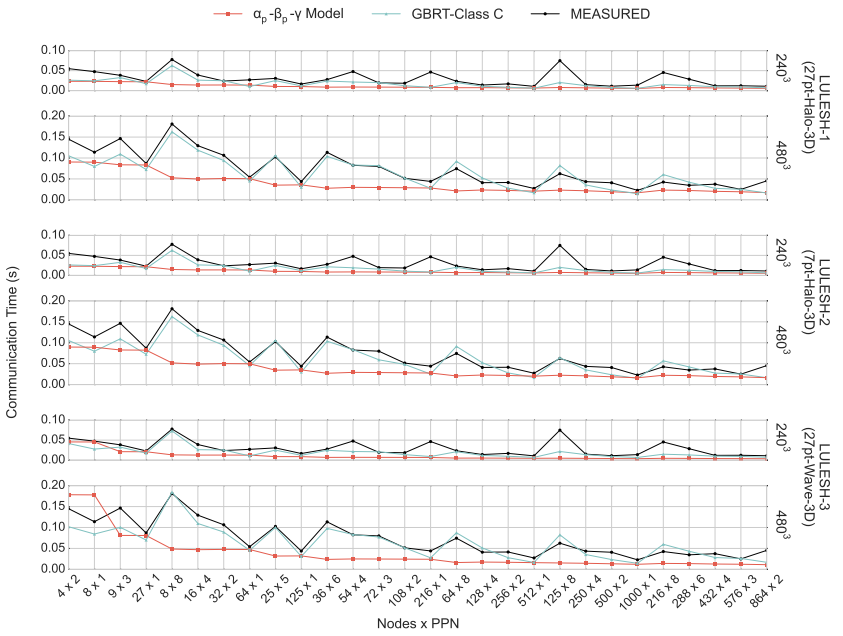


Figure 5.8: Predictions for LULESH with *GBRT-Class C* on Piz Daint.

by the number of cores. For the majority of configurations, *GBRT-Class C* accomplishes accurate predictions, with the exception of four configurations (54×4, 216×1, 125×8, 216×8) where communication time is underpredicted in all three patterns for the small problem size (240^3) and two configurations (8×1, 864×2) where communication time is underpredicted for the large problem size (480^3), though these configurations are dissimilar and mispredictions may be due to noise or imbalance occurring from the presence of computation in the application or interference with other applications at the time of execution. The $\alpha_p - \beta_p - \gamma$ model underpredicts most of the configurations in all three patterns of LULESH, and fails to capture the scalability behavior of all three patterns.

5.4.4 Communication-aware decision-making

We further evaluate the accuracy of our predictive models for communication time in the context of providing accurate and meaningful predictions in the context of communication-aware decision making. We select our most accurate model on Piz Daint, *GBRT-Class C* and use it to predict critical points in decision-making scenarios regarding optimal resource utilization from the side of the user. As explained in Section 4.4.4, supercomputer users attempt to

Table 5.10: Pareto Fronts for nodes and communication time minimization on Piz Daint

Pattern	Problem size	Execution	Pareto front configurations	Predicted configurations	Matches	Distance (maximum)
Halo-3D	128^3	#1	2	2	1	2
Halo-3D	128^3	#2	4	2	1	1
Halo-3D	128^3	#3	5	2	2	0
Halo-3D	256^3	#1	4	4	3	2
Halo-3D	256^3	#2	7	5	2	2
Halo-3D	256^3	#3	6	6	3	1
Halo-3D	512^3	#1	6	6	5	1
Halo-3D	512^3	#2	7	6	6	0
Halo-3D	512^3	#3	6	6	5	0
Halo-3D	1024^3	#1	7	7	**7**	-
Halo-3D	1024^3	#2	7	7	**7**	-
Halo-3D	1024^3	#3	6	7	4	3
Halo-3D	2048^3	#1	7	7	**7**	-
Halo-3D	2048^3	#2	7	7	**7**	-
Halo-3D	2048^3	#3	7	7	**7**	-
Halo-4D	128^4	#1	7	7	6	0
Halo-4D	128^4	#2	7	7	2	2
Halo-4D	256^4	#1	7	7	5	1
Halo-4D	256^4	#2	7	7	6	1
LULESH-1	240^3	#1	12	8	5	4
LULESH-1	480^3	#1	13	11	7	4
LULESH-2	240^3	#1	12	11	6	5
LULESH-2	480^3	#1	13	10	7	4
LULESH-3	240^3	#1	12	9	5	5
LULESH-3	480^3	#1	13	9	7	2

optimize their utilization of the system by maximizing performance of their applications, minimizing the consumption of their budget and minimizing their waiting times in the job queue. To achieve their goals, they need to be able to select optimal configurations of nodes, processes per node and/or OpenMP threads. We address this problem from the side of communication by utilizing our *GBRT-Class C* model to predict the optimal nodes/processes per node configurations, i.e., the configurations that minimize communication time for a specific number of nodes. The budget policy on the Piz Daint system is to charge the user for system utilization by node-hours. We thus formulate the problem as a problem of simultaneously minimizing the number of nodes and communication time and employ the notion of non-dominated Pareto fronts to select the set of Pareto-optimal configurations of nodes and communication time (n, t). Moreover, we use suboptimal Pareto fronts and their distance from the optimal Pareto front to assess our predicted configurations. We refer the reader to Section 4.4.4 for a more detailed specification of the Pareto front.

Figure 5.9 shows two examples of the Pareto fronts on Piz Daint. Each graph shows the measured communication time for all configurations for a

specific communication pattern, problem size and execution on Piz Daint. The Pareto front and the second-best Pareto front (with distance equal to 1) are denoted on the graph. The first graph in Figure 5.9a demonstrates a case where our predictions are successful and all predicted configurations match the Pareto-optimal configurations, as computed with the measured values for communication time. The second graph in Figure 5.9b demonstrates a case where our model does not accurately predict the Pareto front: the Pareto front contains 13 points, while our model predicts 9 Pareto-optimal configurations, 7 of which match the configurations on the Pareto front. However, the distance of the mismatched configurations is at most 2. Table 5.10 summarizes the results of the Pareto front points for measured and predicted communication time and nodes. We refer the reader to Appendix B for the detailed evaluation. We denote as "Matches" the number of predicted configurations that match with the configurations of the Pareto front. In cases where there exist mismatched predicted configurations, we assign each mismatched configuration a rank d that corresponds to the distance of the suboptimal Pareto front on which it belongs. We report the maximum distance among mismatched points. For the 25 constructed Pareto fronts, measured and predicted Pareto fronts match in 5 cases. In the remaining cases, the fronts differ by at most 7 points(see *LULESH-1* on the 240^3 problem size). For 4 of the remaining cases, the maximum is distance is 0: the predicted configurations belong on the Pareto front, but not all configurations of the Pareto front are predicted. In 9 cases, the maximum distance is at most 2. The majority of unmatched configurations corresponds to LULESH, for which our testing set includes multiple different numbers of nodes and the Pareto fronts include 12 to 13 points. In these cases, our model achieves to predict around 50% of the optimal configurations. For reasons of comparison, we constructed the predicted Pareto fronts using predictions acquired with the $\alpha_p - \beta_p - \gamma$ model, and observed zero matches between the 25 measured and predicted Pareto fronts.

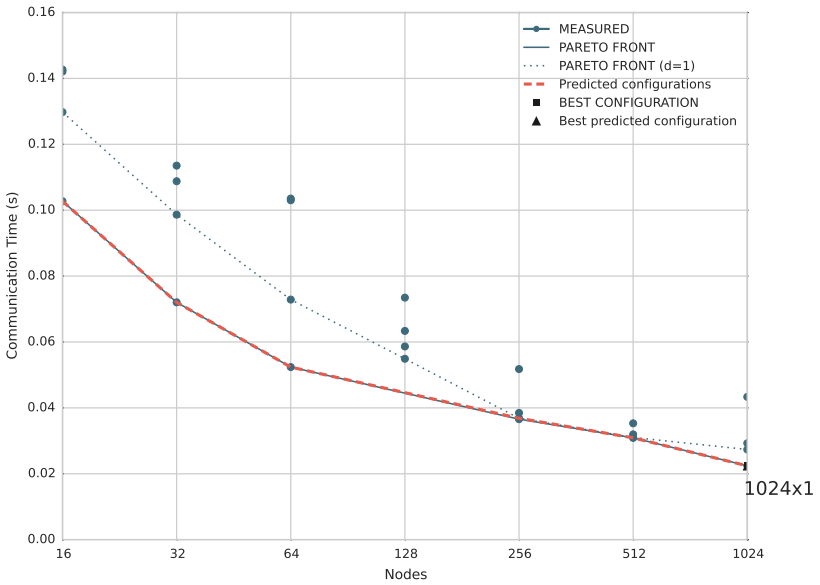
The characteristics of the Pareto-optimal configurations on Piz Daint are not as uniform as those of the Pareto front on Vilje: the optimal configuration does not always utilize as few processes per node as possible. In fact, the results vary both with the communication pattern, the problem size and the execution. Therefore, guiding the user towards a specific configuration (e.g. utilize as few processes per node as possible) does not suffice for the optimization of node-hours on Piz Daint. This observation validates the necessity for accurate prediction models for communication time, that can distinguish between various configurations and assist the user in selecting the best configuration for their application.

The Pareto fronts for nodes and execution time offer additional information as to the configuration that leads to the minimum communication time.

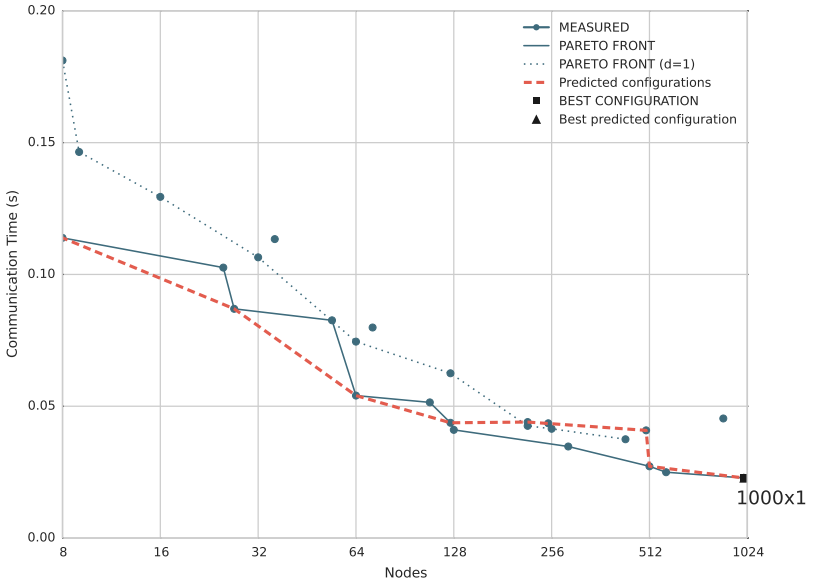
Table 5.11: Minimum communication time configurations on Piz Daint

Pattern	Problem size	Execution	Measured ($n \times ppn$)	Predicted ($n \times ppn$)	Result	Distance
Halo-3D	128^3	#1	128×4	128×8	False	2
Halo-3D	128^3	#2	512×1	32×1	False	0
Halo-3D	128^3	#3	256×2	32×1	False	0
Halo-3D	256^3	#1	128×1	128×1	**True**	-
Halo-3D	256^3	#2	1024×1	1024×4	False	2
Halo-3D	256^3	#3	1024×2	1024×4	False	1
Halo-3D	512^3	#1	1024×4	1024×1	False	1
Halo-3D	512^3	#2	1024×1	1024×1	**True**	-
Halo-3D	512^3	#3	1024×1	1024×1	**True**	-
Halo-3D	1024^3	#1	1024×1	1024×1	**True**	-
Halo-3D	1024^3	#2	1024×1	1024×1	**True**	-
Halo-3D	1024^3	#3	512×2	1024×1	False	3
Halo-3D	2048^3	#1	1024×1	1024×1	**True**	-
Halo-3D	2048^3	#2	1024×1	1024×1	**True**	-
Halo-3D	2048^3	#3	1024×1	1024×1	**True**	-
Halo-4D	128^4	#1	1024×1	1024×1	**True**	-
Halo-4D	128^4	#2	1024×2	1024×1	False	2
Halo-4D	256^4	#1	1024×1	1024×1	**True**	-
Halo-4D	256^4	#2	1024×1	1024×1	**True**	-
LULESH-1	240^3	#1	512×1	1000×1	False	3
LULESH-1	480^3	#1	1000×1	1000×1	**True**	-
LULESH-2	240^3	#1	512×1	1000×1	False	3
LULESH-2	480^3	#1	1000×1	1000×1	**True**	-
LULESH-3	240^3	#1	512×1	1000×1	False	3
LULESH-3	480^3	#1	1000×1	1000×1	**True**	-

This configuration corresponds to the point with the highest number of nodes included in the Pareto front. If the communication pattern and/or problem size under consideration is not scalable, then the point with the highest node count in the Pareto front corresponds to the maximum number of nodes (and cores) that can be deployed before communication scalability breaks. The identification of communication scalability breaks is particularly useful for the user, who can combine it with any information on the scalability of computation and select the maximum number of nodes (and cores) that they should use for the execution of their application. We annotate the measured and predicted optimal (best) configurations in Figure 5.9. Table 5.11 shows the measured and predicted (with *GBRT-Class C*) minimum time configurations for our communication patterns on Piz Daint. We use the notion of Pareto front distance to assess the mispredicted configurations: we compute the distance of the Pareto front on which the mispredicted configuration belongs. Our model correctly predicts the configuration that results in the minimum communication time for 14 out of the 25 cases. For the remaining cases, the predicted best configurations lie at worst on the 4th Pareto front (distance equals 3). For 2 cases, the best configuration is mispredicted by a configuration that lies on the Pareto front but does not result in the minimum time. In the same scenario, the $\alpha_p - \beta_p - \gamma$ model predicts correctly only 4 out 25 configurations for the minimum communication time.



(a) All points match: *Halo-3D* on a 512^3 problem size (execution #3).



(b) 7 points match: *LULESH-3* on a 480^3 problem size.

Figure 5.9: Examples of predicting Pareto-optimal configurations for communication time on Piz Daint.

6

Predictive communication modeling on *ARIS*

6.1 Introduction

In this chapter, we present the application of our predictive communication modeling methodology (see Section 3.6) on the *ARIS* cluster. We describe the steps to build machine-learning models [1] for communication time and evaluate the predictive ability of our models on various communication patterns. We compare the predictive performance of our models and evaluate the best performing model in detail. We finally utilize this best-fitting model in communication-aware decision-making scenarios that target the optimization of resource utilization on *ARIS*.

6.2 Modeling with machine-learning

To construct machine-learning predictive communication models on ARIS, we applied the machine-learning methodology described in Section 3.6 and built three predictive models for communication time, using Gradient Boosting Regression Trees. Each of the three models uses features from different classes: model *GBRT-Class A* uses Class A features only, *GBRT-Class B* uses Class A and Class B features and model *GBRT-Class C* uses features from all three classes, include *GBRT-Class C*. As on Vilje and Piz Daint, we use the maximum value for features *l*, *PD*, and *PM*, as the minimum, average and maximum values for these features are equal for all points of our training set. We applied systematic feature selection for the *GBRT-Class C* model, using recursive feature elimina-

[1] We did not build statistical learning models on ARIS, as we acquired access to this system after already evaluating our methodologies on *Vilje* and *Piz Daint*.

Table 6.1: Parameter space for benchmark executions on ARIS for modeling with machine learning

Parameter		**Range**	
n	8-256	8-128	8 - 128
ppn	1-20	1-20	4 - 20
l	16B-16MB	16B-16MB	16B - 16MB
PM	1-8	1-8	
#executions	1	1	1
#points		7040	

Table 6.2: Tested range of values and selected values for the parameters of the GBRT method and number of features on ARIS

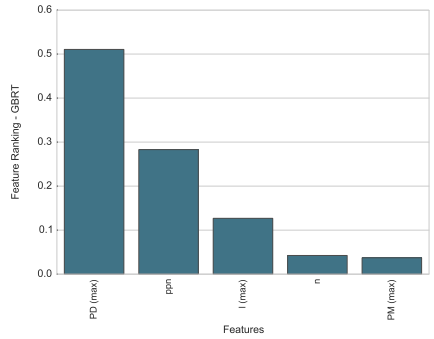
Parameter	**Range**	**Selected Value**
n_estimators	100 - 2000	500
learning_rate	0.001 - 1	0.01
min_samples_leaf	1 - 5	1
min_samples_split	2 - 8	2
max_depth	3 - 8	6
features_classA	-	5
features_classB	-	19
features_classC	15 - 37	37

tion in combination with support vector regression. However, for the case of ARIS, eventually, no features were eliminated.

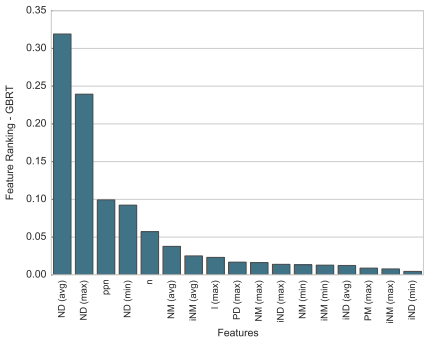
To train our machine-learning models for ARIS, we collected a training set by executing our benchmark on ARIS, for various configurations, message sizes and numbers of messages. We repeated the benchmarking process twice, for two reasons. The first reason is to collect varying values for Class C features. The second reason is to collect varying values for Class B features, by slightly changing the randomness of our pattern, to the direction of increasing intra-node communication, as the nodes of ARIS have a NUMA architecture with 20 cores per node, thus it is important for our models to capture the inter-node to intra-node ratio of communication, through sufficient values in the training set. The resulting training set includes 7040 points and its specifics are presented in Table 6.1.

As the GBRT method needs to be tuned for a number of parameters, we test the method with multiple different values for these parameters, as well as different numbers of features for the Class C model and selected the model

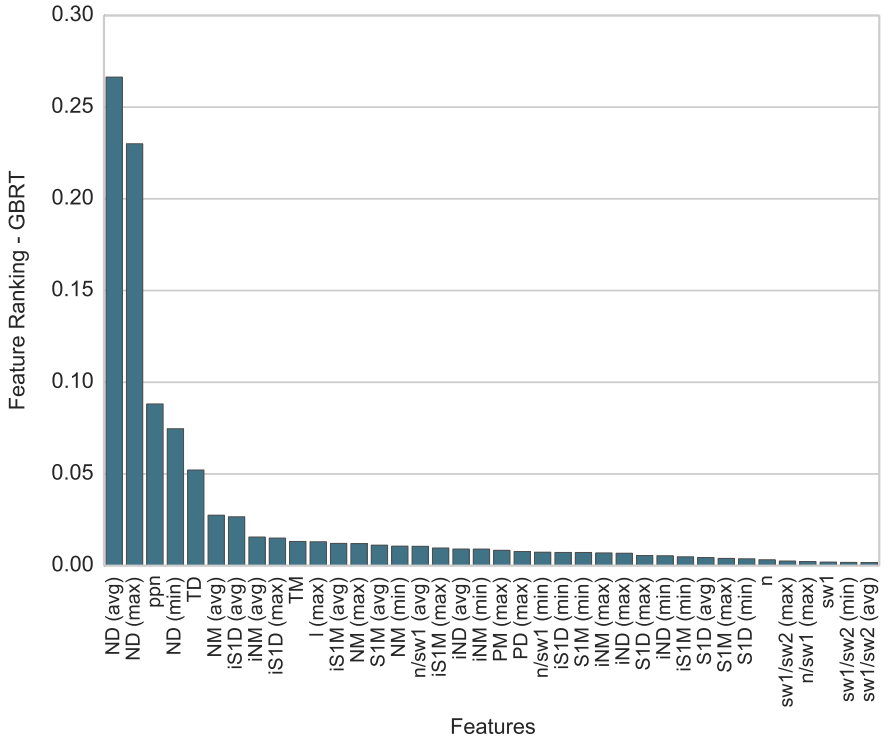
with the best fit on the training set. We present the range of values for which we tested the method, along with the selected values, in Table 6.2. The final predictive models result from the training of GBRT model with our training set, after setting the number of features and the method parameters with the selected values.



(a) GBRT-Class A



(b) GBRT-Class B



(c) GBRT-Class C

Figure 6.1: Feature ranking for the GBRT models on ARIS.

We use the ranking of features provided by the GBRT method (on a scale from 0 to 1), to shed some light on the factors that impact communication performance on ARIS. However, as we have explained before, this ranking only reflects the utilization of the features by the specific method and a low score does not necessarily correspond to a lower impact of the feature to communication performance or a lack of causality. We present the feature ranking for the three models in Figure 6.1. For *GBRT-Class A*, the most important feature is *PD*, followed by the number of processes per node, *ppn*, which we expected to have a high impact on communication time, due to the high number of cores per node on ARIS. The number of nodes is not equally influential on ARIS, unlike Vilje and Piz Daint. This is owed to the non-blocking topology of ARIS which allows multiple nodes to communicate simultaneously, maintaining high throughput and bandwidth. For both *GBRT-Class B* and *GBRT-Class C*, data flowing through the node primarily affects communication, as indicated by the high importance of the features *ND (avg)* and *ND (max)*. The features are followed by the number of processes per node, which maintain their importance across all models. In *GBRT-Class C*, all Class C features are utilized by the model with almost equal, but not high importance, which implies that they are used at the lower levels of the decision trees.

6.3 Evaluation

6.3.1 Communication patterns

To evaluate our predictive models, we used three applications that encapsulate various communication patterns. We experimented with the point-to-point communication phases of the LULESH proxy application: *LULESH-1*, a 27-point 3D-halo exchange, *LULESH-2*, a 6-point 3D-halo exchange and *LULESH-3*, a 27-point wavefront. We refer the reader to Section 4.4.1 for more information on LULESH and its patterns. We also experimented with the HPCG benchmark (v.2.0) [Dongarra et al., 2015]. In its un-optimized version, the HPCG benchmark solves the Preconditioned Conjugate Gradient algorithm on a discretized 3D rectangular grid of dimensions $nx \times ny \times nz$ on a 3D-cartesian grid of processes $px \times py \times pz$. The HPCG benchmark exposes two point-to-point communication patterns. The first, *HPCG-SpMV*, is a 27-point 3D-halo exchange, similar to *LULESH-1*, which is the communication pattern of the sparse matrix-vector multiplication (SpMV) operation on the sparse matrix of HPCG. Each process, depending on its place on the grid, may exchange up to 6 2D-faces, 12 1D-edges and 8 corners with neighboring processes. The second, *HPCG-MG*, comprises of multiple 27-point 3D-halo exchanges on different sizes of the grid. It is the communication pattern of the precondition-

Table 6.3: Details of the testing set on ARIS

	ARIS		
Pattern	LULESH	HPCG	QCD-Kernel D
Domain Size	120^3, 240^3, 480^3	480^3, 960^3	$32 \times 32 \times 32 \times 40$, $32 \times 32 \times 64 \times 40$
Iterations	100	100	100
n	9-216	27-256	16-256
ppn	1-20	1-20	1-20
#executions	1	1	1
#points		429	

ing method of HPCG, the three-level hierarchical multigrid method, which coarsens the grid, i.e. decreases the problem size at each depth by a factor of $2^{3\times depth}$, where $depth = 0, 1, 2, 3$. The multigrid method calls the SpMV and SYMGS methods and performs three 27-point 3D-halo exchanges at depths 0,1 and 2, and one 27-point 3D-halo exchange at depth 3. Thus, the communication of multigrid involves 10 distinct communication phases of 27-point 3D-halo exchanges for four different problem sizes, resulting from the coarsening of the grid. The third application we experiment with is Kernel D of the PRACE QCD benchmark suite, based on tmLQCD [Jansen and Urbach, 2009]. The kernel solves the core matrix-vector multiplication for standard Wilson fermions. The operation refers to the application of the Dirac operator (the Wilson hopping matrix) on a lattice vector and requires the exchange of the boundary cites of the lattice vector (the Fermion field). The vector is 4D and the resulting communication pattern, denoted as *QCD-KernelD* is a halo-exchange of 3D-field slices with all 8 neighbours in a 4D cartesian grid.

We predict communication time for *LULESH*, *HPCG* and *QCD-KernelD*, for various problem sizes and execution configurations. The specifics of the testing set for ARIS are given in Table 6.3. LULESH by design expects the number of processes to be a power of 3, so all execution configurations for LULESH are bound by this constraint. The same constraint holds for HPCG, as we strong-scale the application. Kernel D of QCD is bound by two constraints on the dimension of the local 4D grid: the lattice size on the Z dimension and the product of the lattice sizes on the T, X and Y dimensions both require to be even. Thus, the execution configurations for our selected problem sizes for QCD are bound by these constraints.

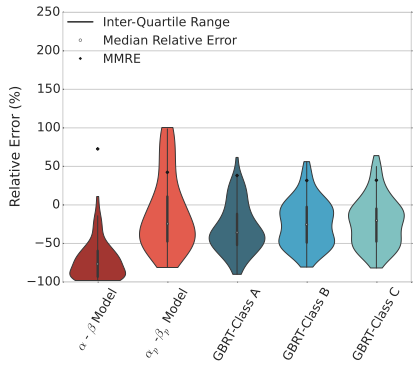
Table 6.4: Measured parameters for the analytical and semi-empirical models on ARIS

Parameter	Value
α	1.431 us
β	0.195 ns
γ	0.596 us

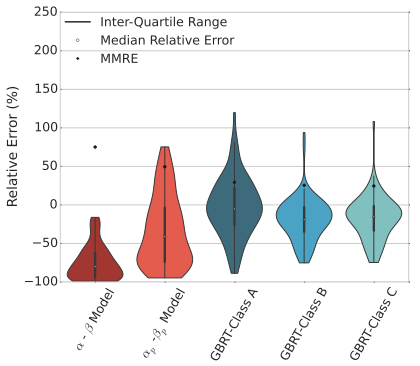
6.3.2 Model comparison

We start our evaluation by comparing the predictive performance of our three models built with the Gradient Boosting Regression Trees method. For comparison purposes, we also constructed the analytical $\alpha - \beta$ model, denoted as $\alpha - \beta Model$, and the semi-empirical $\alpha - \beta$ model with penalties on the α and β parameters, which we denote as the $\alpha_p - \beta_p Model$. Note that semi-empirical models can take five different forms, depending on the penalties. We promoted the $\alpha_p - \beta_p$ model as the best-fitting on ARIS. This model omits the γ parameter, which captures distance as a penalty. We give the measured values for the α, β and γ parameters in Table 6.4. Although the value of γ is high, compared to the value of α, the inclusion of this parameter in the model resulted in poor accuracy and goodness-of-fit.

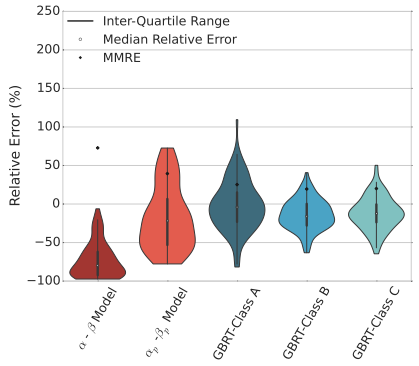
Figure 6.2 shows the distribution of the relative errors of predictions, for all five models, on each communication pattern and on aggregate. Respectively, Table 6.5 presents the scores of the $Pred_{0.25}$, R^2 and *RCC* metrics. A first observation is that the performance of the analytical model, $\alpha - \beta Model$, is poor. The model underpredicts all patterns and demonstrates extremely low R^2 scores, thus its parameters do not suffice to explain the variations in communication time. The semi-empirical $\alpha_p - \beta_p$ model performs better than the analytical model and exhibits very high accuracy for the *HPCG-MG* pattern, which is the only pattern in our experiments that includes many short messages (< 1 kilobyte), coming from the coarsening of the initial grid. It also exhibits its lowest accuracy in the case of *QCD-Kernel D*, that includes many long messages (in the orders of hundreds of kilobytes). We can thus conclude that the semi-empirical model on ARIS is effective for predictions involving short messages, but fails to capture communication performance in the case of heavy traffic.



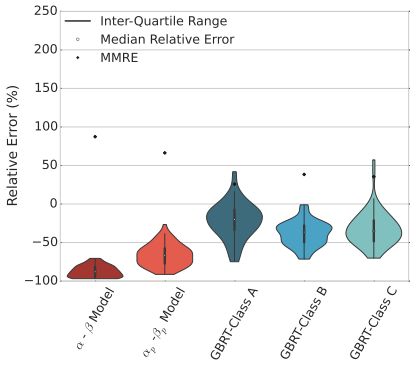
(a) LULESH-1 (27pt-Halo-3D)



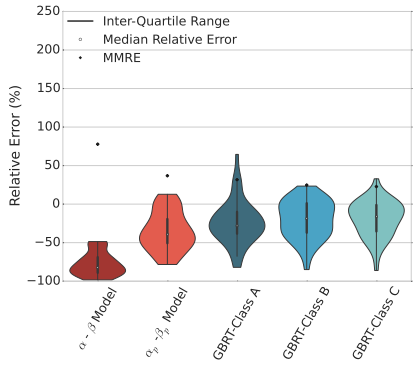
(b) LULESH-2 (7pt-Halo-3D)



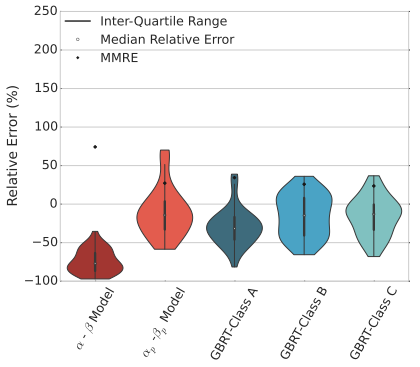
(c) LULESH-3 (27pt-Wave-3D)



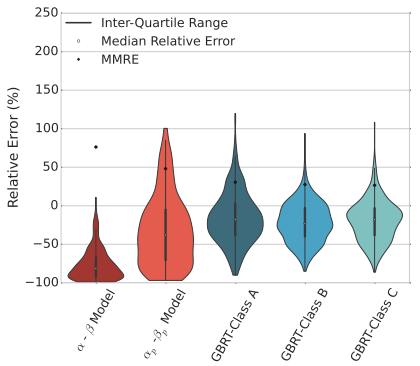
(d) QCD-Kernel D



(e) HPCG - SpMV (27pt-Halo-3D)



(f) HPCG - MG (multiple 27pt-Halo-3D)



(g) Aggregate

Figure 6.2: Model comparison on ARIS.

Table 6.5: Model comparison through accuracy and goodness-of-fit metrics on ARIS

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β *Model*	5.56	-0.664	0.593
α_p -β_p *Model*	27.78	0.191	0.878
GBRT-Class A	30.00	0.603	0.859
GBRT-Class B	42.22	0.574	0.879
GBRT-Class C	43.33	0.549	0.879

(a) LULESH-1

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β *Model*	5.56	-0.419	0.617
α_p -β_p *Model*	21.11	-0.110	0.830
GBRT-Class A	50.00	0.700	0.871
GBRT-Class B	64.44	0.583	0.902
GBRT-Class C	63.33	0.552	0.901

(b) LULESH-2

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β *Model*	5.56	-0.494	0.640
α_p -β_p *Model*	30.00	0.187	0.875
GBRT-Class A	55.56	0.794	0.892
GBRT-Class B	66.67	0.907	0.929
GBRT-Class C	72.22	0.912	0.923

(c) LULESH-3

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β *Model*	0.00	-1.026	0.705
α_p -β_p *Model*	0.00	-0.550	0.836
GBRT-Class A	54.41	0.689	0.856
GBRT-Class B	17.65	0.472	0.896
GBRT-Class C	27.94	0.498	0.890

(d) QCD-Kernel D

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β *Model*	0.00	-1.002	0.575
α_p -β_p *Model*	34.78	-0.042	0.880
GBRT-Class A	36.96	0.585	0.857
GBRT-Class B	58.70	0.598	0.890
GBRT-Class C	63.04	0.559	0.890

(e) HPCG - SpMV

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β *Model*	0.00	-1.117	0.552
α_p -β_p *Model*	52.17	0.462	0.871
GBRT-Class A	28.26	0.566	0.869
GBRT-Class B	52.17	0.667	0.871
GBRT-Class C	58.70	0.632	0.875

(f) HPCG - MG

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β *Model*	3.49	-0.111	0.736
α_p -β_p *Model*	25.81	0.562	0.766
GBRT-Class A	43.95	0.779	0.884
GBRT-Class B	50.93	0.801	0.908
GBRT-Class C	54.88	0.786	0.909

(g) Aggregate

Concerning our *GBRT* models, *GBRT-Class A* performs remarkably well on most patterns and on aggregate on ARIS. In particular, it outperforms the *GBRT-Class B* and *GBRT-Class C* models in the case of *QCD-Kernel D*. As explained in the previous paragraph, this pattern creates heavy traffic on the network on a regular communication pattern, with multiple long messages. The high values of the dominant feature in *GBRT-Class A*, Process Data *PD* suffice to capture the communication times of this pattern on ARIS. *GBRT-Class B* and *GBRT-Class C* models show very high accuracy across all patterns. The addi-

tion of Class C features to the latter does improve its $Pred_{0.25}$ and *RCC* scores, however, as the dominant feature in both models is Node Data, the differences in accuracy and goodness-of-fit between the two models are not very high. Overall, we observe a bias towards underprediction by all our GBRT models on all patterns. This owes to the properties of our training set and the way the GBRT method produces the final model: our training set only includes configurations with numbers of nodes that are a power of two and five different values for processes per node. Our testing set includes more"irregular" configurations, due to the constraints posed by the applications. The GBRT method interpolates between values in the training set, resulting to underpredictions for the communication time on certain configurations. Augmenting the training set with additional configurations would help reduce this bias.

6.3.3 Detailed evaluation

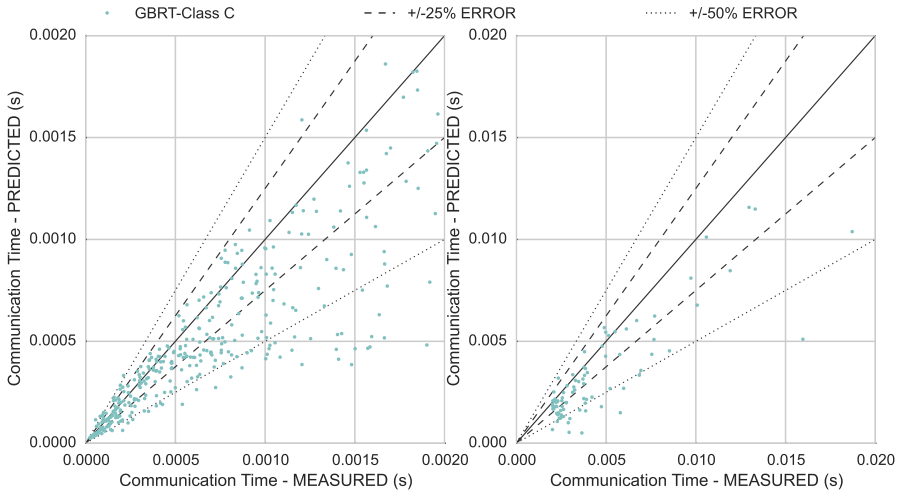


Figure 6.3: Scatterplots for predictions with *GBRT-Class C* on ARIS. Communication time is normalized to a single iteration.

From our model comparison, we promote *GBRT-Class C* as the model with the best predictive performance on ARIS. The model offers predictions just-in-time before the execution of the application and has very high accuracy and goodness-of-fit, as it scores 54.88% for $Pred_{0.25}$, 0.786 for R^2 and 0.909 for *RCC* on aggregate. Additionally, it reduces the mean magnitude of relative errors *MMRE* to 26.61%, from 44.84% of the semi-empirical $\alpha_p - \beta_p$ model. We further analyze the predictive performance of this model, on aggregate and on each communication pattern. First, we examine all points in the testing set,

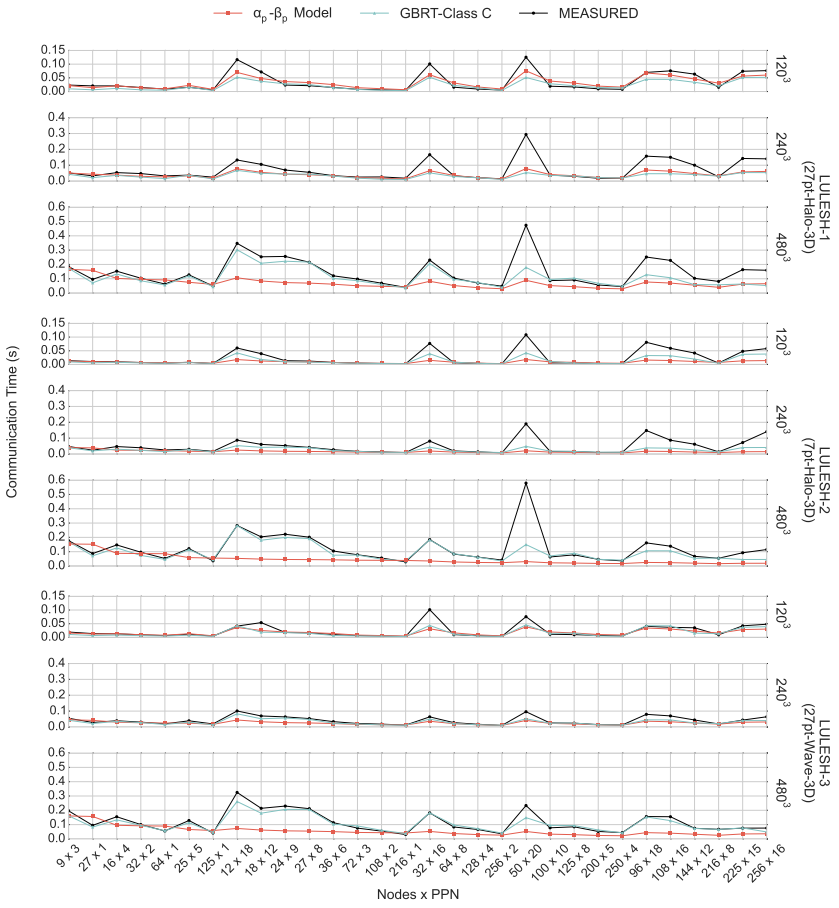


Figure 6.4: Predictions for LULESH with *GBRT-Class C* on ARIS.

in the form of a scatterplot, shown in Figure 6.3, normalized to a single iteration, broken in two sets by communication time, for the sake of visibility. The majority of predictions (54.88%) lie within the $\pm 25\%$ error range, while 85.82% of predictions lie within the $\pm 50\%$ error range. 57 predictions are underpredicted, with a relative error lower than -50%, while 75% of them have a communication time that is lower than 2 milliseconds. We thus conclude that, although our model shows some bias towards underpredictions, these mostly occur when the communication time of the tested patterns is insignificantly small.

We then examine the *GBRT-Class C* predictions on all communication pat-

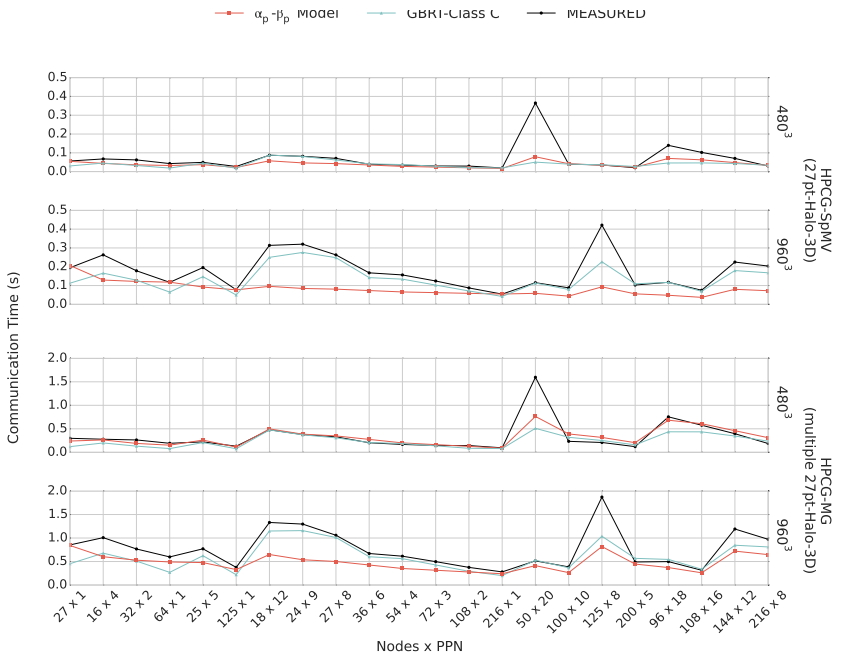


Figure 6.5: Predictions for HPCG with *GBRT-Class C* on ARIS.

terns. Figure 6.4 presents the predictions for the three communication patterns and three problem sizes of LULESH, for all tested configurations, sorted by the number of cores. Our model achieves very good predictions and captures the scalability of all three communication patterns of LULESH. Configurations with more than a 1000 cores are underpredicted for *LULESH-1* and *LULESH-2*, especially for the two smaller problem sizes (120^3 and 240^3), where message sizes are small. Two configurations with lower number of cores (32×16 and 50×20) are systematically underpredicted for all patterns and problem sizes. Both correspond to high numbers of processes per node and the high communication time indicates that our model does not sufficiently capture contention effects of multiple processes per node, although similar configurations exist in our training set. We believe that contention effects arise from the co-existence of computation phases with communication phases in the context of the application and are primarily memory effects and not network effects. We compare our predictions with those acquired with the $\alpha_p - \beta_p$ model, which fails to capture the scalability behavior of the patterns and predicts minimal variations among the different configurations.

Figure 6.5 shows the predictions for the two point-to-point communication patterns in HPCG, *HPCG-SpMV* and *HPCG-MG*. Our model accurately

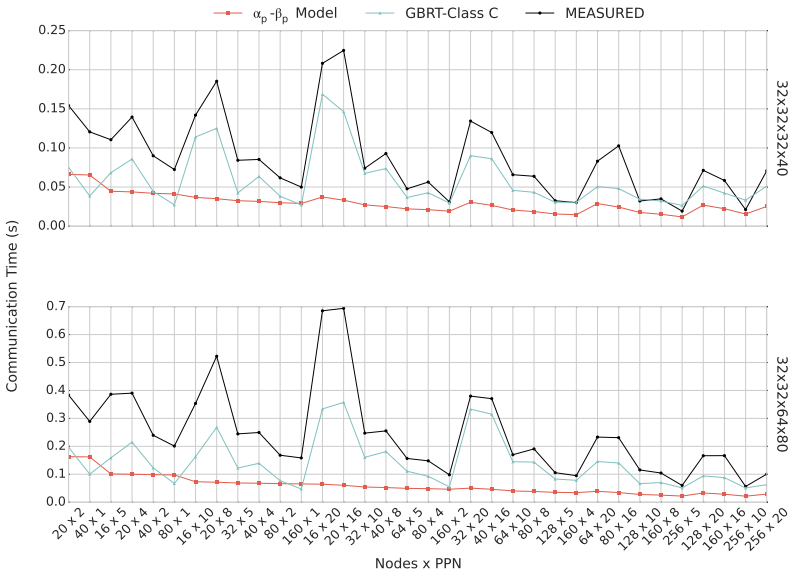


Figure 6.6: Predictions for QCD - Kernel D with *GBRT-Class C* on ARIS.

predicts communication scalability for both patterns and both problem sizes, 480^3 and 960^3. For *HPCG-SpMV*, which has the same communication pattern as *LULESH-1*, we observe a notable underprediction for the same configuration, 50×20 nodes (nodes $\times$ processes per node). *HPCG-MG* also comes with one underprediction for one configuration, 125×8 (nodes $\times$ processes per node), which does not have any similarities with configurations that are systematically underpredicted, thus we consider it an outlier. The $\alpha_p - \beta_p$ model shows good prediction results for the smaller problem size of HPCG, 480^3, the messages of which are significantly shorter than the equivalent of *LULESH*, however it fails to capture the scaling behavior of communication for the larger problem size.

Finally, Figure 6.6 demonstrates the predictions for *QCD-Kernel D*, for the two problem sizes in our testing set. For this pattern, our model exhibits remarkable goodness-of-fit, as it predicts accurately all increases and decreases in communication time for any configuration, and also shows very good accuracy for large numbers of cores for both problem sizes. Contrarily, the $\alpha_p - \beta_p$ model falsely predicts very low values for the communication time of *QCD-Kernel D* and does not capture the variations in communication time between different configurations.

Table 6.6: Pareto Fronts for cores and communication time minimization on ARIS

Pattern	Problem size	Pareto front configurations	Predicted configurations	Matches	Distance (maximum)
LULESH-1	120^3	4	4	4	-
LULESH-1	240^3	5	4	4	0
LULESH-1	480^3	4	4	4	-
LULESH-2	120^3	4	5	4	1
LULESH-2	240^3	4	5	4	1
LULESH-2	480^3	4	5	4	1
LULESH-3	120^3	4	4	4	-
LULESH-3	240^3	5	5	5	-
LULESH-3	480^3	4	4	4	-
HPCG-SpMV	480^3	4	3	3	0
HPCG-SpMV	960^3	4	4	4	-
HPCG-MG	480^3	4	3	3	0
HPCG-MG	960^3	4	4	4	-
QCD-Kernel D	$32 \times 32 \times 32 \times 40$	6	4	4	0
QCD-Kernel D	$32 \times 32 \times 64 \times 80$	7	3	3	0

6.3.4 Communication-aware decision-making

We further evaluate the accuracy of our predictive models for communication time in the context of providing accurate and meaningful predictions in the context of communication-aware decision making. We use *GBRT-Class C* to predict critical points in decision-making scenarios regarding optimal resource utilization from the side of the user. As explained in Section 4.4.4, supercomputer users attempt to optimize their utilization of the system by maximizing the performance of their applications, minimizing the consumption of their budget and minimizing their waiting times in the job queue. To achieve their goals, they need to be able to select optimal configurations of nodes, processes per node and/or OpenMP threads. We address this decision-making scenario from the side of communication of parallel applications, by utilizing our *GBRT-Class C* model to predict the optimal nodes/processes per node configurations, i.e., the configurations that minimize communication time for a specific number of cores. The budget policy on ARIS is to charge the user for system utilization by core-hours. Therefore, we formulate the problem as a problem of simultaneously minimizing the number of cores and communication time and employ the notion of non-dominated Pareto fronts to select the set of Pareto-optimal configurations of cores and communication time (c, t). Moreover, we use suboptimal Pareto fronts and their distance from the optimal Pareto front to assess our predicted configurations. We refer the reader to Section 4.4.4 for a more detailed specification of the Pareto front.

Figure 6.7 shows two examples of the Pareto fronts on ARIS. Each graph shows the measured communication time for all configurations for a specific communication pattern and problem on ARIS. The Pareto front and the second-

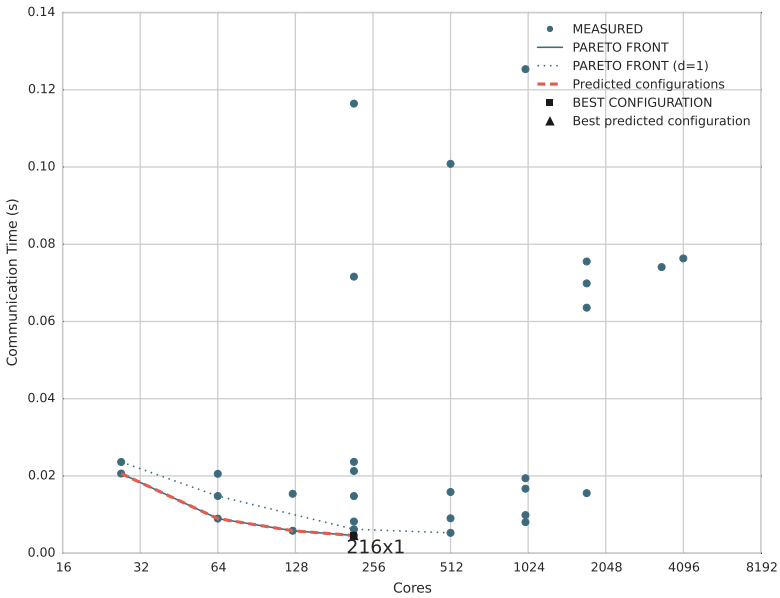
best Pareto front (with distance equal to 1) are denoted on the graph. The first graph in Figure 6.7a demonstrates the case of *LULESH-1* on the smallest problem size, where our predictions are successful and all predicted configurations match the Pareto-optimal configurations, as computed with the measured values for communication time. Figure 6.7b demonstrates a case where our model does not accurately predict the Pareto front: the Pareto front contains 4 points, while our model predicts 5 Pareto-optimal configurations. The 4 points in the predicted Pareto-optimal configurations match the Pareto-optimal configurations. The fifth predicted point, in fact, belongs to the second best Pareto front (distance equals 1). Table 6.6 summarizes the results of the Pareto front points for measured and predicted communication time and cores. We refer the reader to Appendix C for the corresponding graphs. We denote as "Matches" the number of predicted configurations that match with the configurations of the Pareto front. In cases where there exist mismatched predicted configurations, we assign each mismatched configuration a rank d that corresponds to the distance of the suboptimal Pareto front to which it belongs. We report the maximum distance among mismatched points. For the 15 constructed Pareto fronts, measured and predicted Pareto fronts match in 7 cases. In the remaining cases, the fronts differ by at most 4 points(see *QCD-Kernel D* on the $32 \times 32 \times 64 \times 80$ problem size). For 5 of the mismatched Pareto fronts, the maximum is distance is 0: the predicted configurations belong on the Pareto front, but not all configurations of the Pareto front are predicted. For the remaining 3 cases, the maximum distance is 1. For reasons of comparison, we constructed the predicted Pareto fronts using predictions acquired with the $\alpha_p - \beta_p$ model, and observed 9 matches between the 15 measured and predicted Pareto fronts. Thus, despite its mispredictions, the $\alpha_p - \beta_p$ model is also successful on predicting critical configurations on ARIS.

The Pareto fronts for cores and execution time offer additional interesting information. The point with the highest number of cores included in the Pareto front corresponds to the configuration that leads to the minimum communication time. If the communication pattern and/or problem size under consideration is not scalable, then the point with the highest core count in the Pareto front corresponds to the maximum number of cores that can be deployed before communication scalability breaks. We annotate the measured and predicted optimal (best) configurations in Figure 6.7. Table 6.7 shows the measured and predicted (with *GBRT-Class C*) minimum time configurations for our communication patterns on ARIS. We use the notion of Pareto front distance to assess the mispredicted configurations. Our model correctly predicts the configuration that results in the minimum communication time for 9 out of the 15 cases. The wrong predictions occurring for LULESH and HPCG correspond to neighboring configurations, for which communication time dif-

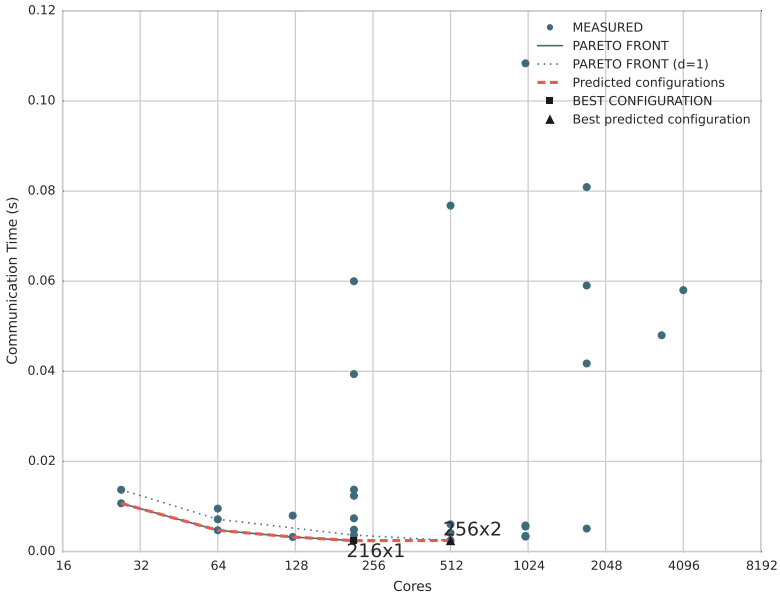
Table 6.7: Minimum communication time configurations on ARIS

Pattern	Problem size	Measured ($n \times ppn$)	Predicted ($n \times ppn$)	Result	Distance
LULESH-1	120^3	216×1	216×1	**True**	-
LULESH-1	240^3	256×2	216×1	False	0
LULESH-1	480^3	216×1	216×1	**True**	-
LULESH-2	120^3	216×1	256×2	False	1
LULESH-2	240^3	256×2	256×2	**True**	-
LULESH-2	480^3	216×1	256×2	False	1
LULESH-3	120^3	216×1	216×1	**True**	-
LULESH-3	240^3	216×1	216×1	**True**	-
LULESH-3	480^3	216×1	216×1	**True**	-
HPCG-SpMV	480^3	216×1	125×1	False	0
HPCG-SpMV	960^3	216×1	216×1	**True**	-
HPCG-MG	480^3	216×1	125×1	False	0
HPCG-MG	960^3	216×1	216×1	**True**	-
QCD-Kernel D	$32 \times 32 \times 32 \times 40$	256×5	256×5	**True**	-
QCD-Kernel D	$32 \times 32 \times 64 \times 80$	256×10	160×1	False	0

fers by 1 millisecond. In the same scenario, the $\alpha_p - \beta_p$ model predicts correctly an equal number of configurations, that is 9 out of 15 configurations for the minimum communication time.



(a) All points match: *LULESH-1* on a 120^3 problem size.



(b) 4 points match: *LULESH-2* on a 120^3 problem size.

Figure 6.7: Examples of predicting Pareto-optimal configurations for communication time on ARIS.

7

Conclusions

In this thesis, we proposed and presented a methodology for the construction of predictive models for the communication of HPC applications. Following an empirical approach to predictive modeling, we defined features for communication performance on three large-scale systems: Vilje, an SGI Altix supercomputer using InfiniBand in an enhanced hypercube topology, Piz Daint, a Cray XC30 (now Cray XC50) supercomputer, using Cray Aries in the Dragonfly topology, and ARIS, an IBM NextScale cluster using InfiniBand in a fat-tree topology. We collected a set of measurements on each system, with a generic benchmark for point-to-point communication and constructed various predictive models for communication time, employing statistical and machine learning methods.

During the modeling process, we were able to identify features that affect communication performance on each system. On Vilje, data flows through the compute nodes and the InfiniBand switches primarily affect communication time. On Piz Daint, the amount of data that flow through the various points of the system, i.e., cores (processes), nodes, Aries SoCs, chassis and groups of the Dragonfly topology, all affect communication time. On ARIS, traffic generated from each node dominates communication time prediction.

We evaluated the predictive ability of our models on different point-to-point communication patterns, drawn from real-world HPC applications, and conclude that multiple well-defined features, generic benchmarking and automated machine-learning methods, as gradient-boosted regression trees, offer a predictive model for point-to-point communication patterns that exhibits high accuracy and goodness-of-fit, across all tested communication patterns and execution configurations, on all three systems. Our methodology provides predictions just-in-time before the execution of an application and is suitable for decision-making regarding the optimization of resource utilization on super-

computers and clusters.

To the best of our knowledge, this is the first work that presents a cross-application, cross-platform methodology for communication time prediction of HPC applications ahead of execution, able to deliver high accuracy and goodness-of-fit, as indicated by our *MMRE*, R^2, *RCC* and $Pred_{0.25}$ scores, which reach 23.98%, 0.922, 0.942 and 61.43% on Vilje, 21.32%, 0.986, 0.940 and 66.57% on Piz Daint and 26.61%, 0.786, 0.909 and 54.88% on ARIS. Our models significantly improve upon the predictive accuracy of other, semi-empirical approaches for communication time prediction, reducing the *MMRE* by 50% on aggregate. Additionally, we evaluated the accuracy of our models on communication-aware decision-making scenarios, targeting the selection of communication-optimal execution configurations. We observed significant accuracy, especially for Vilje and Piz Daint, where alternative semi-empirical models have little or no success in predicting optimal configurations.

Future directions of this thesis include the extension of our methodology to target collective communication. Collective communication modeling shares many of the challenges with point-to-point communication modeling, however, the communication pattern is predetermined. As such, we anticipate that some regular collective patterns can be well-modeled with a single or a few features, e.g. the number of processes, as indicated by the work of Shudler et al. [Shudler et al., 2015], while other collective patterns can be modeled as point-to-point communication since MPI implementations also internally use point-to-point communication for several irregular and many-to-many collectives. Another important future direction is the extension of this methodology to the direction of performance projections to significantly larger scales of cores and threads than those available in the context of this thesis. Finally, we aim to provide an automated toolchain for the prediction of communication time on large-scale systems and work towards the integration of a prediction framework into resource management software.

List of publications

The work in this thesis produced a number of publications in a series of well-known, high-quality journals and conferences of the HPC community.

Journal publications

- N. Papadopoulou, G. Goumas, N. Koziris. "Predictive communication modeling for HPC applications", Cluster Computing, 20(3): 2725–2747, Sep 2017b.
- A. Haritatos, N. Papadopoulou, K. Nikas, G. Goumas, N. Koziris: "Contention-Aware Scheduling Policies for Fairness and Throughput", Co-Scheduling of HPC Applications 28 (2017): 22

Conference publications

- N. Papadopoulou, L. Oden, P. Balaji, "A performance study of UCX over InfiniBand", Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 345-354. IEEE Press, 2017
- N. Papadopoulou, G. Goumas, N. Koziris, "Optimizing resource utilization on large-scale systems through predictive communication modelling", NovExS-RAMbO Workshop on Novel Exascale Systems / Rack-Scale Memory Systems and Models, HiPEAC 2017, 2017
- N. Papadopoulou, G. Goumas, N. Koziris: "A machine-learning approach for communication prediction on large-scale applications", Cluster Computing (CLUSTER), 2015 IEEE International Conference on. IEEE, 2015
- A. Abdel-Rehim, C. Alexandrou, N. Anastopoulos, G. Koutsou, I. Liabotis, N. Papadopoulou, "PLQCD library for Lattice QCD on multi-core machines", Proceedings of the 31st International Symposium on Lattice

Field Theory (LATTICE 2013). 29 July-3 August, 2013. Mainz, Germany. Published online at http://pos. sissa. it/cgi-bin/reader/conf. cgi? confid= 187, id. 419. 2013

Formal Acknowledgements

This research was supported with computational resources at the Norwegian Institute of Science and Technology (NTNU) provided by NOTUR, with a grant of computational resources from the Swiss National Supercomputing Center (CSCS) under project ID g83 and with computational time granted from the Greek Research & Technology Network (GRNET) in the national HPC facility (ARIS) under project IDs pa001006, pa006010 and pa170302. The author has received funding from IKY fellowships of excellence for postgraduate studies in Greece - SIEMENS program.

Bibliography

The structural simulation toolkit. http://sst-simulator.org.

Top500 supercomputer sites. URL http://www.top500.org.

Luay Alawneh, Abdelwahab Hamou-Lhadj, and Jameleddine Hassine. Segmenting large traces of inter-process communication with a focus on high performance computing systems. *Journal of Systems and Software*, 120:1–16, 2016.

Albert Alexandrov, Mihai F Ionescu, Klaus E Schauser, and Chris Scheiman. LogGP: incorporating long messages into the LogP model—one step closer towards a realistic model for parallel computation. In *Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, pages 95–105. ACM, 1995.

Bob Alverson, Edwin Froese, Larry Kaplan, and Duncan Roweth. Cray xc series network. *Cray Inc., White Paper WP-Aries01-1112*, 2012.

Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, et al. The landscape of parallel computing research: A view from berkeley. Technical report, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

InfiniBand Trade Association et al. Infiniband specification. *InfiniBand™ Architecutre Release*, 1, 2001.

Kevin J Barker, Kei Davis, Adolfy Hoisie, Darren J Kerbyson, Michael Lang, Scott Pakin, and JoséCarlos Sancho. Using performance modeling to design large-scale systems. *Computer*, 42(10):0042–49, 2009.

Greg Bauer, Steven Gottlieb, and Torsten Hoefler. Performance modeling and comparative analysis of the MILC lattice QCD application su3_rmd. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 652–659. IEEE, 2012.

Paul Bédaride, Augustin Degomme, Stéphane Genaud, Arnaud Legrand, George Markomanolis, Martin Quinson, Mark Lee Stillwell, Frédéric Suter, Brice Videau, et al. Toward better simulation of MPI applications on Ethernet/TCP networks. In *PMBS13-4th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, 2013.

Costas Bekas and Alessandro Curioni. A new energy aware performance metric. *Computer Science-Research and Development*, 25(3-4):187–195, 2010.

Abhinav Bhatelé and Laxmikant V Kalé. Quantifying network contention on large parallel machines. *Parallel Processing Letters*, 19(04):553–572, 2009.

Abhinav Bhatele, Kathryn Mohror, Steven H Langer, and Katherine E Isaacs. There goes the neighborhood: performance degradation due to nearby jobs. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, page 41. ACM, 2013.

Abhinav Bhatele, Andrew R Titus, Jayaraman J Thiagarajan, Nikhil Jain, Todd Gamblin, Peer-Timo Bremer, Martin Schulz, and Laxmikant V Kale. Identifying the culprits behind network congestion. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 113–122. IEEE, 2015.

Mark S Birrittella, Mark Debbage, Ram Huggahalli, James Kunz, Tom Lovett, Todd Rimmer, Keith D Underwood, and Robert C Zak. Intel® omni-path architecture: Enabling scalable, high performance fabrics. In *High-Performance Interconnects (HOTI), 2015 IEEE 23rd Annual Symposium on*, pages 1–9. IEEE, 2015.

David M Brooks, Pradip Bose, Stanley E Schuster, Hans Jacobson, Prabhakar N Kudva, Alper Buyuktosunoglu, J-D Wellman, Victor Zyuban, Manish Gupta, and Peter W Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *Micro, IEEE*, 20(6):26–44, 2000.

Henri Casanova, Frédéric Desprez, George S Markomanolis, and Frédéric Suter. Simulation of mpi applications with time-independent traces. *Concurrency and Computation: Practice and Experience*, 27(5):1145–1168, 2015.

Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

Barbara Chapman, Tony Curtis, Swaroop Pophale, Stephen Poole, Jeff Kuehn, Chuck Koelbel, and Lauren Smith. Introducing openshmem: Shmem for the pgas community. In *Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model*, page 2. ACM, 2010.

WenGuang Chen, JiDong Zhai, Jin Zhang, and WeiMin Zheng. LogGPO: An accurate communication model for performance prediction of MPI programs. *Science in China Series F: Information Sciences*, 52(10):1785–1791, 2009.

UPC Consortium et al. Upc language specifications v1. 2. *Lawrence Berkeley National Laboratory*, 2005.

Samuel Daniel Conte, Hubert E Dunsmore, and Vincent Y Shen. *Software engineering metrics and models*. Benjamin-Cummings Publishing Co., Inc., 1986.

National Research Council et al. *Getting up to speed: The future of supercomputing*. National Academies Press, 2005.

David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten Von Eicken. *LogP: Towards a realistic model of parallel computation*, volume 28. ACM, 1993.

Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

William James Dally and Brian Patrick Towles. *Principles and practices of interconnection networks*. Elsevier, 2004.

Frédéric Desprez, George S Markomanolis, Martin Quinson, and Frédéric Suter. Assessing the performance of mpi applications through time-independent trace replay. In *2011 40th International Conference on Parallel Processing Workshops*, pages 467–476. IEEE, 2011.

Salvatore Di Girolamo, Pierre Jolivet, Keith D Underwood, and Torsten Hoefler. Exploiting offload enabled network interfaces. In *High-Performance Interconnects (HOTI), 2015 IEEE 23rd Annual Symposium on*, pages 26–33. IEEE, 2015.

Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, et al. The international exascale software project roadmap. *The international journal of high performance computing applications*, 25(1):3–60, 2011.

Jack Dongarra, Michael A Heroux, and Piotr Luszczek. Hpcg benchmark: a new metric for ranking high performance computing systems. *Knoxville, Tennessee*, 2015.

Nikolaos Drosinos and Nectarios Koziris. Performance comparison of pure mpi vs hybrid mpi-openmp parallelization models on smp clusters. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 15. IEEE, 2004.

Kurt B Ferreira, Patrick G Bridges, Ron Brightwell, and Kevin T Pedretti. The impact of system design parameters on application noise sensitivity. *Cluster computing*, 16(1):117–129, 2013.

Rosa Filgueira, David E Singh, Jesús Carretero, Alejandro Calderón, and Félix García. Adaptive-CoMPI: Enhancing MPI-based applications' performance and scalability by using adaptive compression. *International Journal of High Performance Computing Applications*, 25(1):93–114, 2011.

Matthew I Frank, Anant Agarwal, and Mary K Vernon. *LoPC: modeling contention in parallel algorithms*, volume 32. ACM, 1997.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning: Data mining, inference, and prediction. *Springer Series in Statistics*, 2009.

Hormozd Gahvari, Allison H Baker, Martin Schulz, Ulrike Meier Yang, Kirk E Jordan, and William Gropp. Modeling the performance of an algebraic multigrid cycle on hpc platforms. In *Proceedings of the international conference on Supercomputing*, pages 172–181. ACM, 2011.

Hormozd Gahvari, William Gropp, Kirk E Jordan, Martin Schulz, and Ulrike Meier Yang. Algebraic multigrid on a dragonfly network: First experiences on a cray xc30. In *International Workshop on Performance Modeling,*

Benchmarking and Simulation of High Performance Computer Systems, pages 3–23. Springer, 2014.

G Goumas, Nikos Anastopoulos, Nectarios Koziris, and Nikolas Ioannou. Overlapping computation and communication in SMT clusters with commodity interconnects. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–10. IEEE, 2009a.

Georgios Goumas, Nikolaos Drosinos, and Nectarios Koziris. Communication-aware supernode shape. *Parallel and Distributed Systems, IEEE Transactions on*, 20(4):498–511, 2009b.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

Georg Hager, Gabriele Jost, and Rolf Rabenseifner. Communication characteristics and hybrid mpi/openmp parallel programming on clusters of multicore smp nodes. In *Proceedings of Cray User Group Conference*, volume 4, page 5455, 2009.

Jerry L Hintze and Ray D Nelson. Violin plots: a box plot-density trace synergism. *The American Statistician*, 52(2):181–184, 1998.

Roger W Hockney. The communication challenge for MPP: Intel Paragon and Meiko CS-2. *Parallel computing*, 20(3):389–398, 1994.

Torsten Hoefler and Marc Snir. Generic topology mapping strategies for large-scale parallel architectures. In *Proceedings of the international conference on Supercomputing*, pages 75–84. ACM, 2011.

Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. Accurately measuring overhead, communication time and progression of blocking and nonblocking collective operations at massive scale. *International Journal of Parallel, Emergent and Distributed Systems*, 25(4):241–258, 2010a.

Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. Characterizing the influence of system noise on large-scale applications by simulation. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE Computer Society, 2010b.

Torsten Hoefler, William Gropp, William Kramer, and Marc Snir. Performance modeling for systematic performance tuning. In *State of the Practice Reports*, page 6. ACM, 2011.

Sascha Hunold and Alexandra Carpen-Amarie. On the impact of synchronizing clocks and processes on benchmarking MPI collectives. In *EuroMPI*, pages 8:1–8:10. ACM, 2015.

Sascha Hunold and Alexandra Carpen-Amarie. Reproducible mpi benchmarking is still not as easy as you think. *IEEE Transactions on Parallel and Distributed Systems*, 27(12):3617–3630, 2016.

Huda Ibeid, Rio Yokota, and David Keyes. A performance model for the communication in fast multipole methods on high-performance computing platforms. *International Journal of High Performance Computing Applications*, page 1094342016634819, 2016.

Fumihiko Ino, Noriyuki Fujimoto, and Kenichi Hagihara. LogGPS: a parallel computational model for synchronization analysis. In *ACM SIGPLAN Notices*, volume 36, pages 133–142. ACM, 2001.

KE Isaacs, T Gamblin, A Bhatele, M Schulz, B Hamann, and PT Bremer. Ordering traces logically to identify lateness in parallel programs. Technical report, Technical Report LLNL-TR-656141, Lawrence Livermore National Laboratory, 2014.

Nikhil Jain, Abhinav Bhatele, Michael P Robson, Todd Gamblin, and Laxmikant V Kale. Predicting application performance using supervised learning on communication features. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, page 95. ACM, 2013.

Nikhil Jain, Abhinav Bhatele, Sam White, Todd Gamblin, and Laxmikant V Kale. Evaluating hpc networks via simulation of parallel workloads. In *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for*, pages 154–165. IEEE, 2016.

Nikhil Jain, Abhinav Bhatele, Xiang Ni, Todd Gamblin, and Laxmikant V Kale. Partitioning low-diameter networks to eliminate inter-job interference. 2017.

Karl Jansen and Carsten Urbach. tmlqcd: a program suite to simulate wilson twisted mass lattice qcd. *Computer Physics Communications*, 180(12):2717–2738, 2009.

Ana Jokanovic, Jose Carlos Sancho, Germán Rodríguez, Alejandro Lucero, Cyriel Minkenberg, and Jesús Labarta. Quiet neighborhoods: Key to protect job performance predictability. *29th IEEE International Parallel & Distributed Processing Symposium (IPDPS-2015)*, 2015.

Laxmikant V Kale and Sanjeev Krishnan. Charm++: a portable concurrent object oriented system based on c++. In *ACM Sigplan Notices*, volume 28, pages 91–108. ACM, 1993.

Ian Karlin, Abhinav Bhatele, Bradford L.. Chamberlain, Jonathan. Cohen, Zachary Devito, Maya Gokhale, Riyaz Haque, Rich Hornung, Jeff Keasler, Dan Laney, Edward Luke, Scott Lloyd, Jim McGraw, Rob Neely, David Richards, Martin Schulz, Charle H. Still, Felix Wang, and Daniel Wong. Lulesh programming model and performance ports overview. Technical Report LLNL-TR-608824, December 2012.

Peter Kogge and John Shalf. Exascale computing trends: Adjusting to the" new normal" for computer architecture. *Computing in Science & Engineering*, 15 (6):16–26, 2013.

Peter Kogge, Keren Bergman, Shekhar Borkar, Dan Campbell, W Carson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Kerry Hill, et al. Exascale computing study: Technology challenges in achieving exascale systems. 2008.

Dong Li, Edgar A Leon, and Bronis R de Supinski. Adaptive parallelism: Integrated performance, power, and resilience modeling. In *DOE Workshop on Modeling & Simulation of Systems and Applications (MODSIM'14)*, 2014.

Sean Luke. *Essentials of metaheuristics*, volume 113. Lulu Raleigh, 2009.

Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

Dan McMorrow. Technical challenges of exascale computing. *MITRE Corporation, McLean, VI, USA*, 2013.

João Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *ACM Computing Surveys (CSUR)*, 45(1):10, 2012.

Robert M Metcalfe and David R Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, 1976.

Csaba Andras Moritz and Matthew I Frank. LoGPC: Modeling network contention in message-passing programs. *ACM SIGMETRICS Performance Evaluation Review*, 26(1):254–263, 1998.

Csaba Andras Moritz and Matthew I Frank. LoGPG: Modeling network contention in message-passing programs. *Parallel and Distributed Systems, IEEE Transactions on*, 12(4):404–415, 2001.

Gihan R Mudalige, Mary K Vernon, and Stephen A Jarvis. A plug-and-play model for evaluating wavefront computations on parallel architectures. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–14. IEEE, 2008.

Nikela Papadopoulou, Georgios Goumas, and Nectarios Koziris. A machine-learning approach for communication prediction of large-scale applications. In *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*, pages 120–123. IEEE, 2015.

Nikela Papadopoulou, Georgios Goumas, and Nectarios Koziris. Optimizing resource utilization on large-scale systems through predictive communication modeling. NovExS-RAMbO: Workshop on Novel Exascale Systems / Rack-Scale Memory Systems and Models, HiPEAC'17, 2017a.

Nikela Papadopoulou, Georgios Goumas, and Nectarios Koziris. Predictive communication modeling for hpc applications. *Cluster Computing*, 20(3): 2725–2747, Sep 2017b. ISSN 1573-7543. doi: 10.1007/s10586-017-0821-8. URL https://doi.org/10.1007/s10586-017-0821-8.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

José Carlos Sancho, Kevin J Barker, Darren J Kerbyson, and Kei Davis. Quantifying the potential benefit of overlapping communication and computation in large-scale scientific applications. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, pages 17–17. IEEE, 2006.

Timo Schneider, Torsten Hoefler, Ryan E Grant, Brian W Barrett, and Ron Brightwell. Protocols for fully offloaded collective operations on accelerated network adapters. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 593–602. IEEE, 2013.

John Shalf, Sudip Dosanjh, and John Morrison. Exascale computing technology challenges. In *International Conference on High Performance Computing for Computational Science*, pages 1–25. Springer, 2010.

Sameer S Shende and Allen D Malony. The tau parallel performance system. *International Journal of High Performance Computing Applications*, 20(2):287–311, 2006.

Sergei Shudler, Alexandru Calotoiu, Torsten Hoefler, Alexandre Strube, and Felix Wolf. Exascaling your library: Will your implementation meet your expectations? In *Proceedings of the 29th ACM on International Conference on Supercomputing*, pages 165–175. ACM, 2015.

Marc Snir. *MPI–the Complete Reference: the MPI core*, volume 1. MIT press, 1998.

Edgar Solomonik and James Demmel. Communication-optimal parallel 2.5 D matrix multiplication and LU factorization algorithms. In *Euro-Par 2011 Parallel Processing*, pages 90–109. Springer, 2011.

Garrick Staples. Torque resource manager. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 8. ACM, 2006.

Leslie G Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

Jeffrey Vetter and Chris Chambreau. mpip: Lightweight, scalable mpi profiling. 2005.

Felix Wolf, Allen Malony, Sameer Shende, and Alan Morris. Trace-based parallel performance overhead compensation. *High Performance Computing and Communications*, pages 617–628, 2005.

Andy Yoo, Morris Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.

Li Yu, Dong Li, Sparsh Mittal, and Jeffrey S Vetter. Quantitatively modeling application resilience with the data vulnerability factor. In *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*, pages 695–706. IEEE, 2014.

Cha Zhang and Yunqian Ma. *Ensemble machine learning*. Springer, 2012.

Jun Zhu, Alexey Lastovetsky, Shoukat Ali, Rolf Riesen, and Khalid Hasanov. Asymmetric communication models for resource-constrained hierarchical ethernet networks. *Concurrency and Computation: Practice and Experience*, 27(6):1575–1590, 2015.

Appendix A

In this appendix, we present the plots for scalability predictions with *GBRT-Class C* on the *Vilje* supercomputer, as a supplement to Chapter 4, where we only present the best, median and worst subsets of predictions. We also present the plots for Pareto-optimal configuration predictions on *Vilje*, as a supplement to Chapter 4, where we indicatively present two cases.

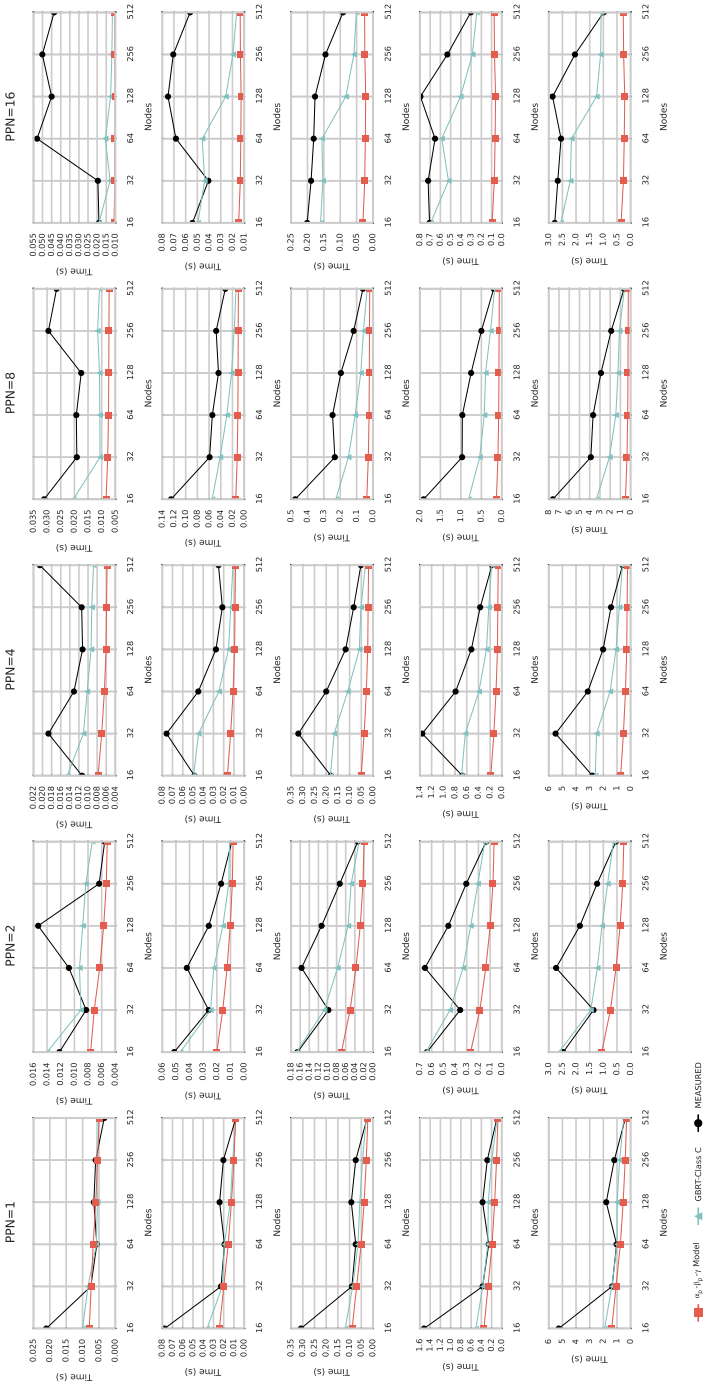


Figure 1: Predictions for *Halo-3D* (execution #1) for all problem sizes on Vilje with *GBRT-Class C* and the $\alpha_p - \beta_p - \gamma$ Model

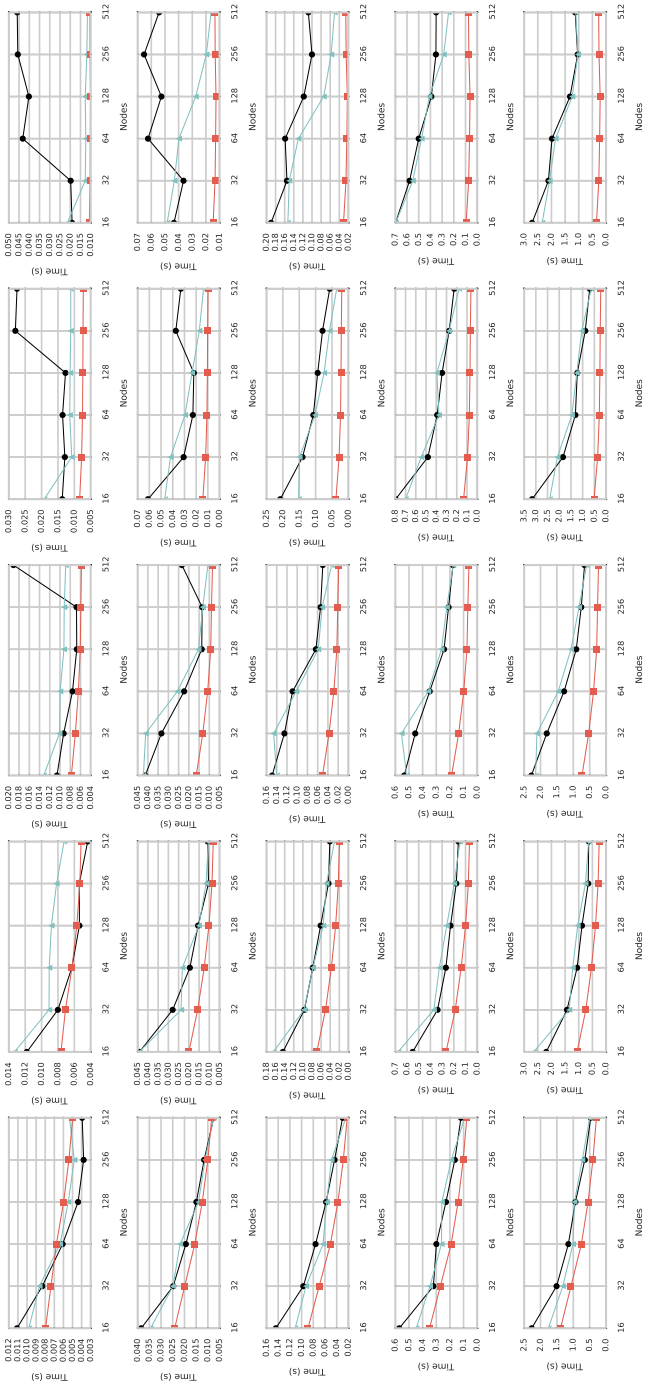


Figure 2: Predictions for *Halo-3D* (execution #2) for all problem sizes on Vilje with *GBRT-Class C* and the $\alpha_p - \beta_p - \gamma$ Model

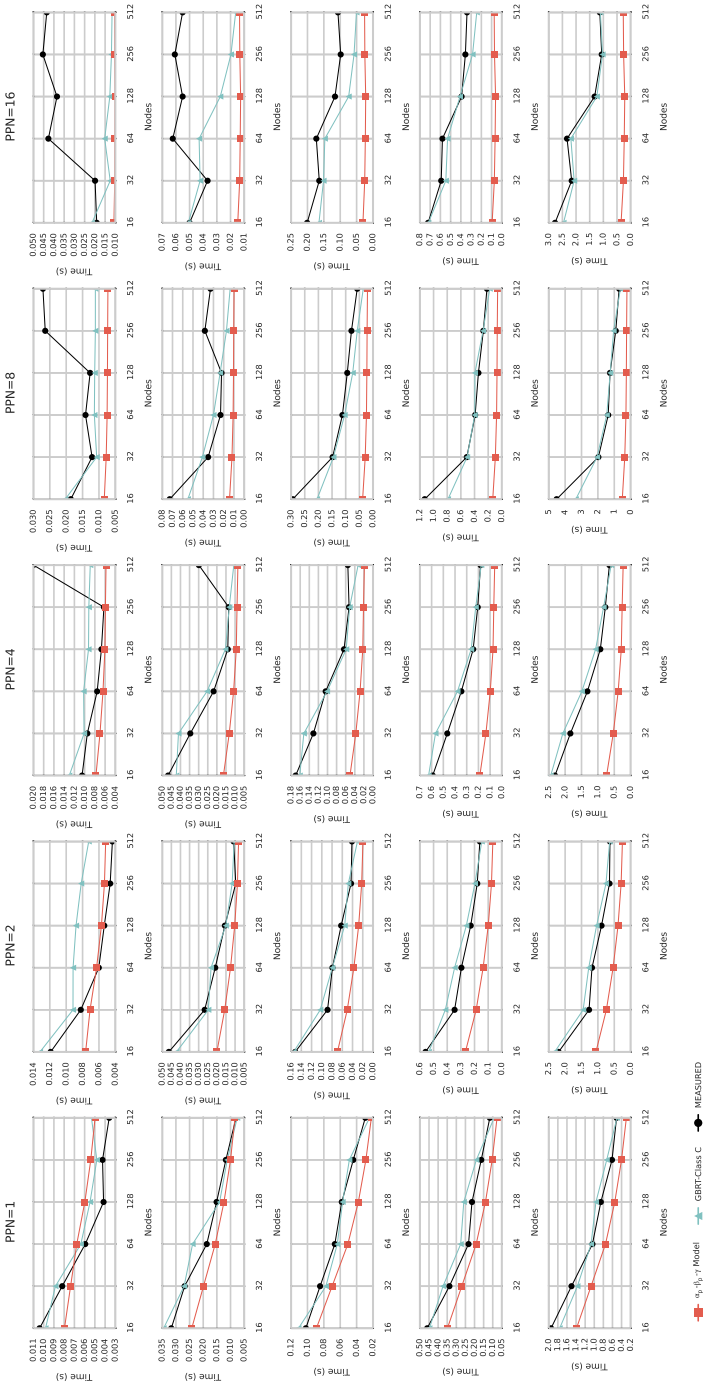


Figure 3: Predictions for *Halo-3D* (execution #3) for all problem sizes on Vilje with *GBRT-Class C* and the $\alpha_p - \beta_p - \gamma$ Model

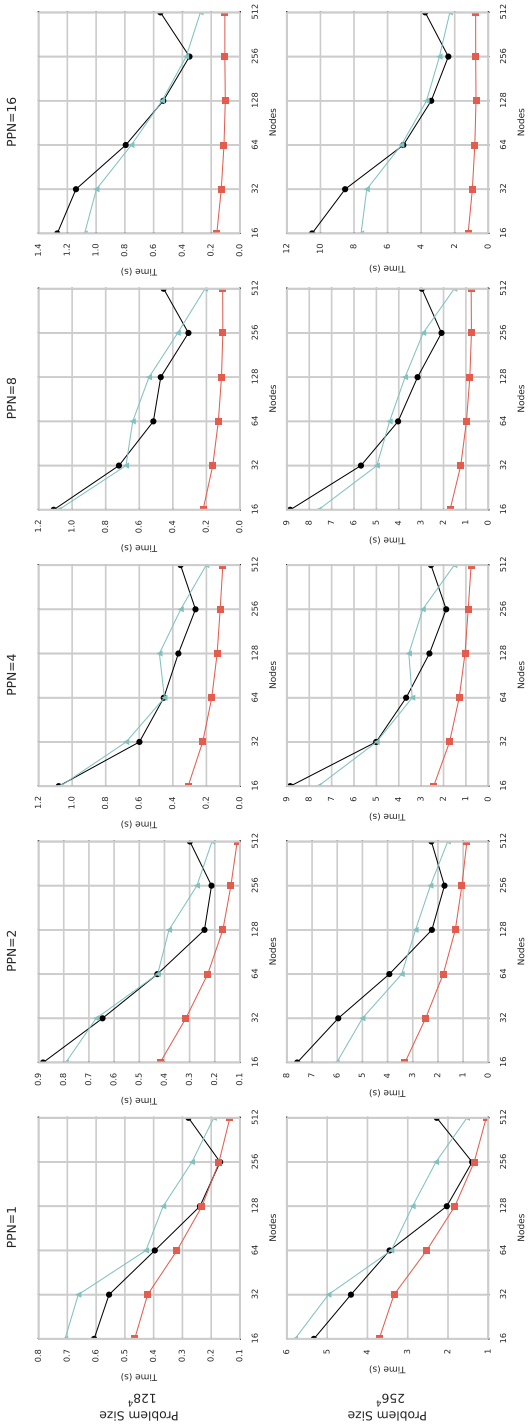


Figure 4: Predictions for *Halo-4D* (execution #1) for all problem sizes on Vilje with *GBRT-Class C* and the $\alpha_p - \beta_p - \gamma$ Model

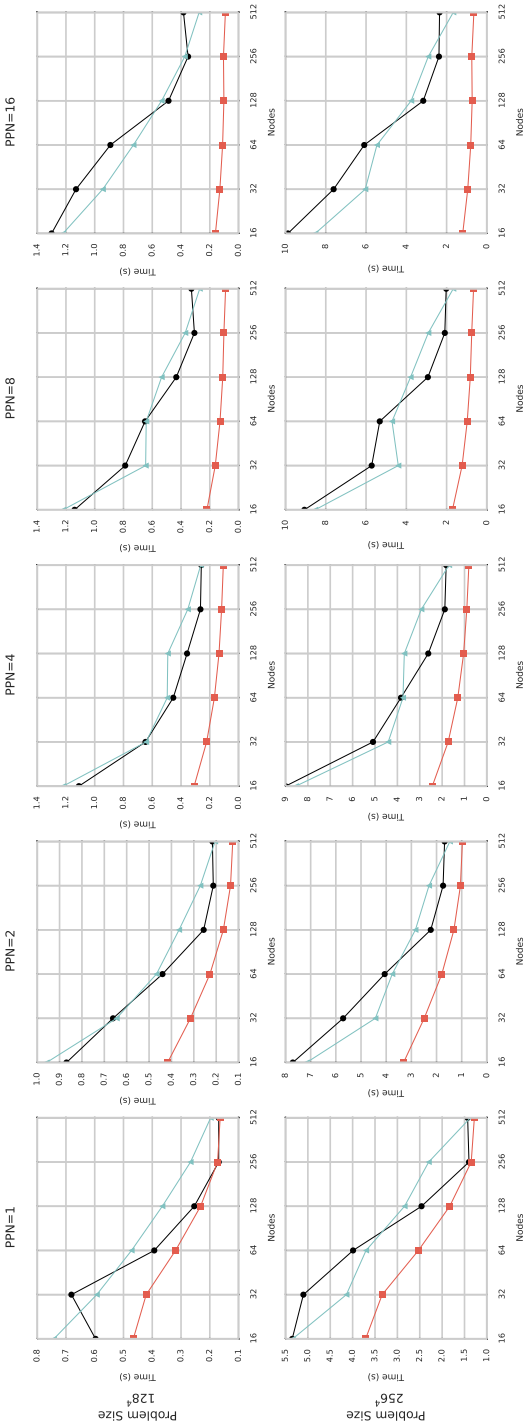
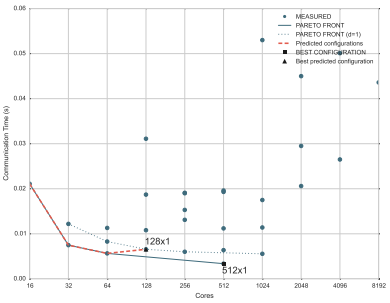
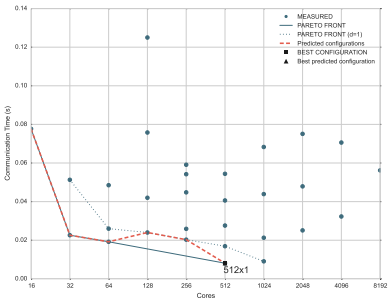


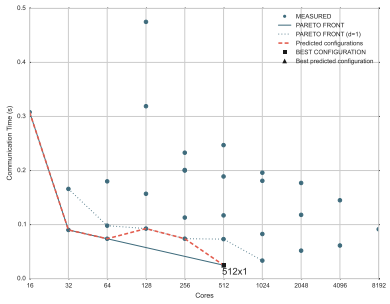
Figure 5: Predictions for *Halo-4D* (execution #2) for all problem sizes on Vilje with *GBRT-Class C* and the $\alpha_p - \beta_p - \gamma$ Model



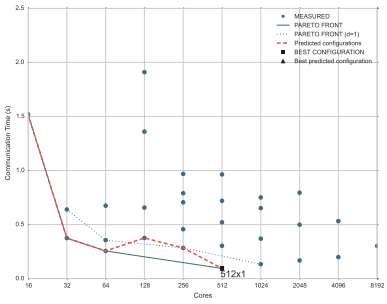
(a) Problem size: 128^3



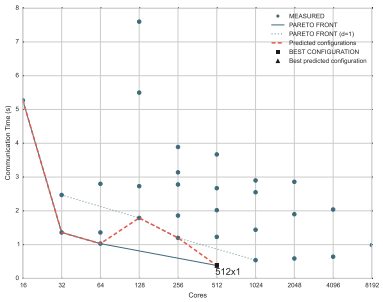
(b) Problem size: 256^3



(c) Problem size: 512^3

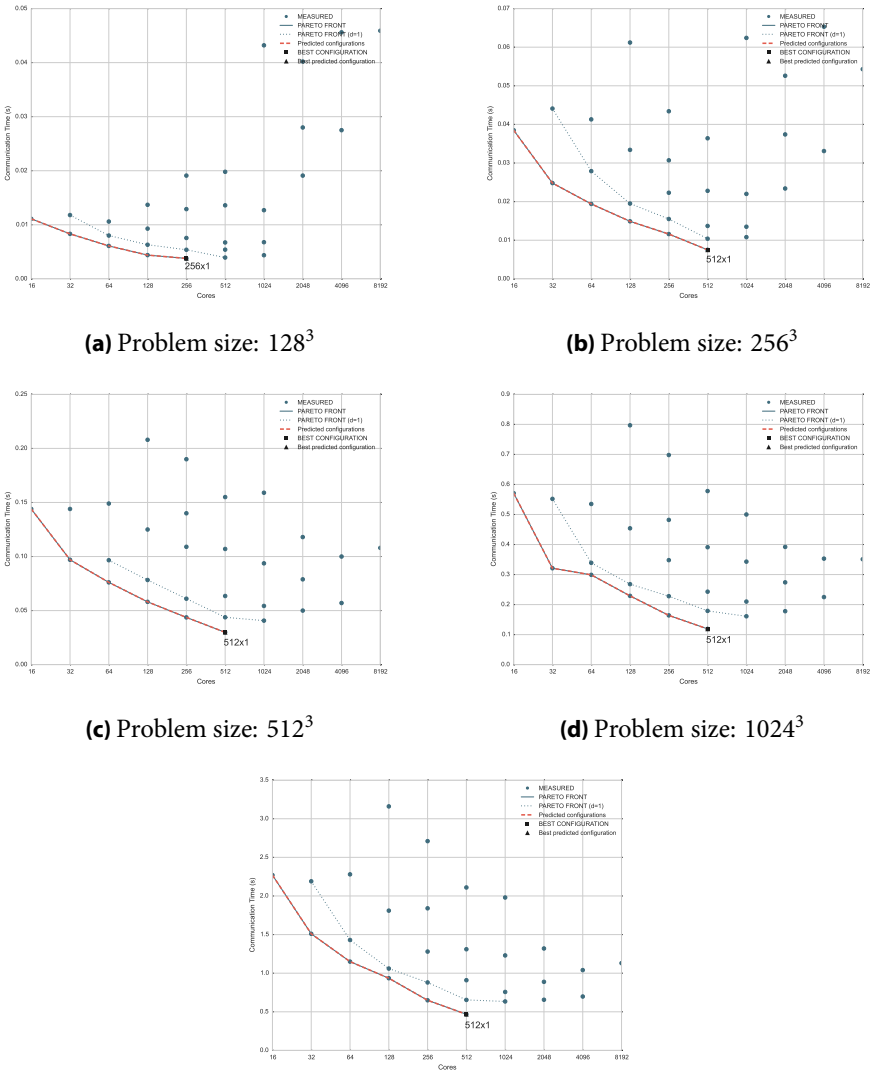


(d) Problem size: 1024^3



(e) Problem size: 2048^3

Figure 6: Pareto-optimal configuration predictions for *Halo-3D* (execution #1) on Vilje



(a) Problem size: 128^3

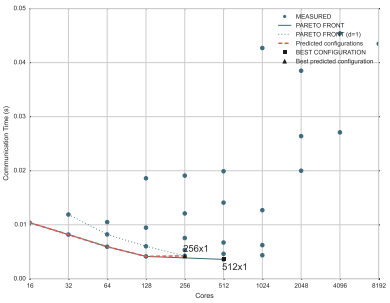
(b) Problem size: 256^3

(c) Problem size: 512^3

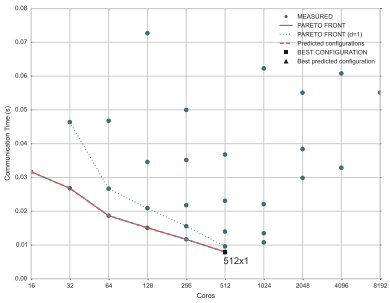
(d) Problem size: 1024^3

(e) Problem size: 2048^3

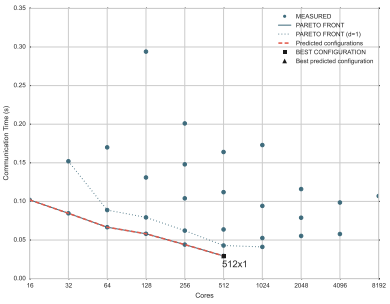
Figure 7: Pareto-optimal configuration predictions for *Halo-3D* (execution #2) on Vilje



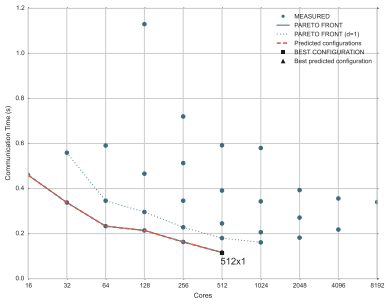
(a) Problem size: 128^3



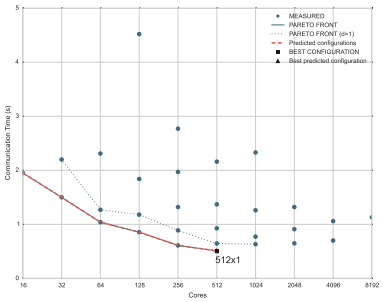
(b) Problem size: 256^3



(c) Problem size: 512^3

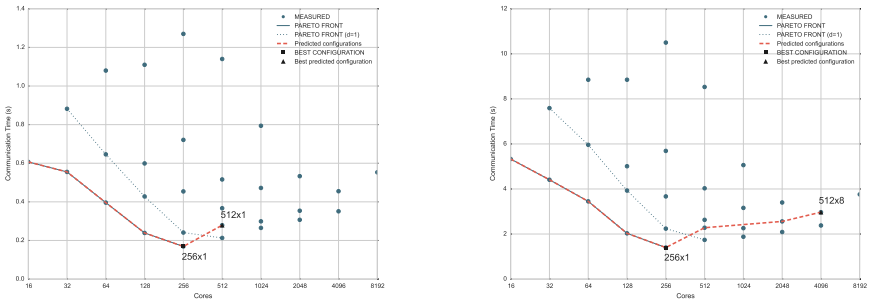


(d) Problem size: 1024^3



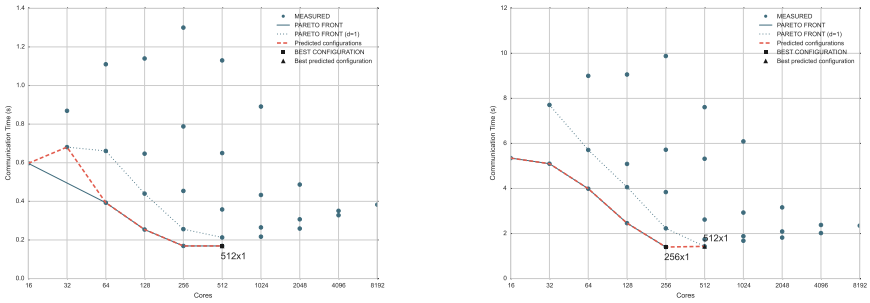
(e) Problem size: 2048^3

Figure 8: Pareto-optimal configuration predictions for *Halo-3D* (execution #3) on Vilje



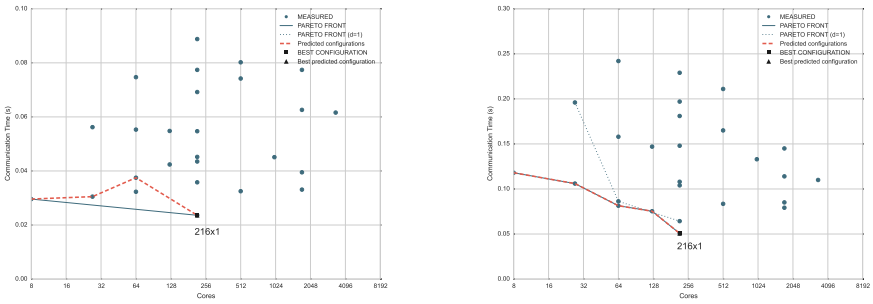
(a) Problem size: 128^4 **(b)** Problem size: 256^4

Figure 9: Pareto-optimal configuration predictions for *Halo-4D* (execution #1) on Vilje



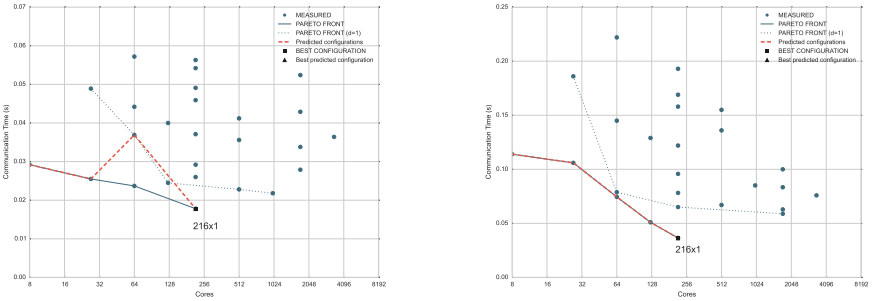
(a) Problem size: 128^4 **(b)** Problem size: 256^4

Figure 10: Pareto-optimal configuration predictions for *Halo-4D* (execution #2) on Vilje



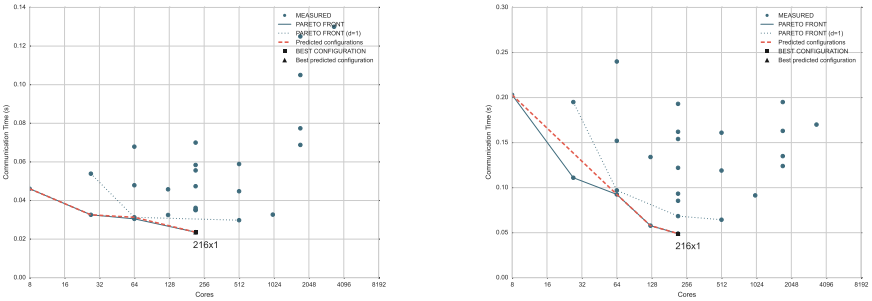
(a) Problem size: 240^3 **(b)** Problem size: 480^3

Figure 11: Pareto-optimal configuration predictions for *LULESH-1* on Vilje



(a) Problem size: 240^3 **(b)** Problem size: 480^3

Figure 12: Pareto-optimal configuration predictions for *LULESH-2* on Vilje



(a) Problem size: 240^3 **(b)** Problem size: 480^3

Figure 13: Pareto-optimal configuration predictions for *LULESH-3* on Vilje

Appendix B

In this appendix, we present the plots for scalability predictions with *GBRT-Class C* on the *Piz Daint* supercomputer, as supplemental material to Chapter 5, where we only present the best, median and worst subsets of predictions. We also present the plots for Pareto-optimal configuration predictions on *Piz Daint*, as a supplement to Chapter 5, where we indicatively present two cases.

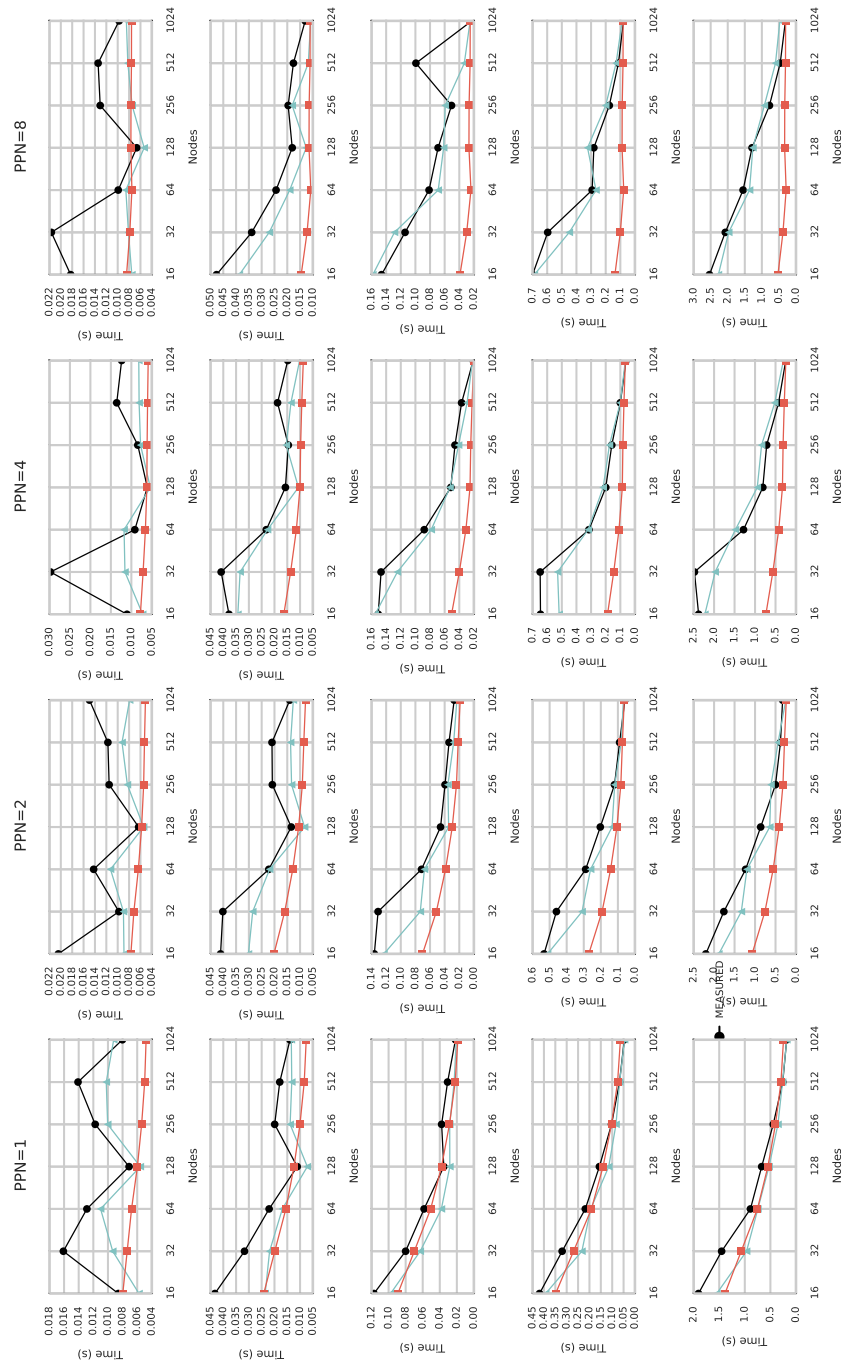


Figure 14: Predictions for *Halo-3D* (execution #1) for all problem sizes on Piz Daint with *GBRT-Class C* and the $\alpha_p - \beta_p - \gamma$ Model

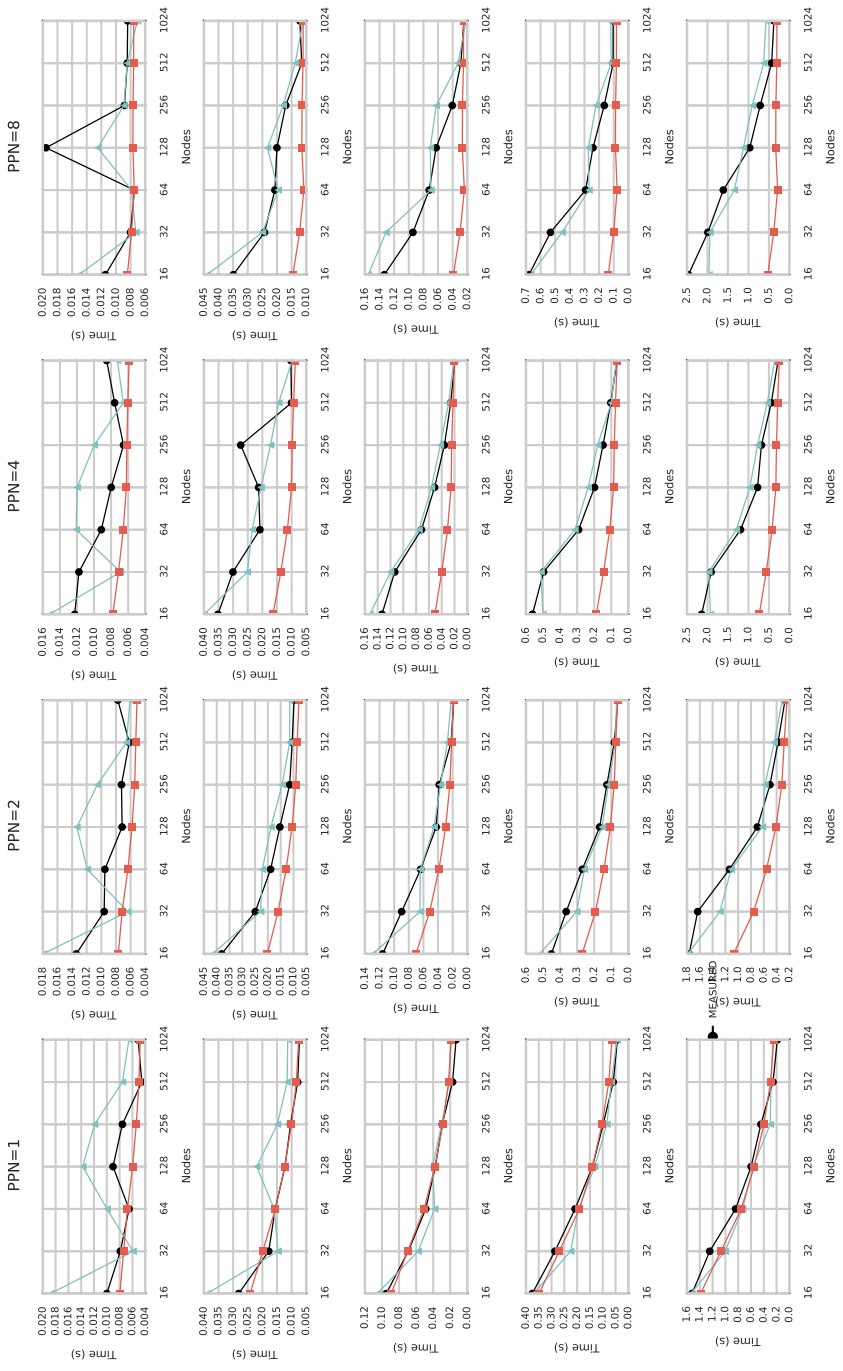


Figure 15: Predictions for *Halo-3D* (execution #2) for all problem sizes on Piz Daint with *GBRT-Class C* and the $\alpha_p - \beta_p - \gamma$ Model

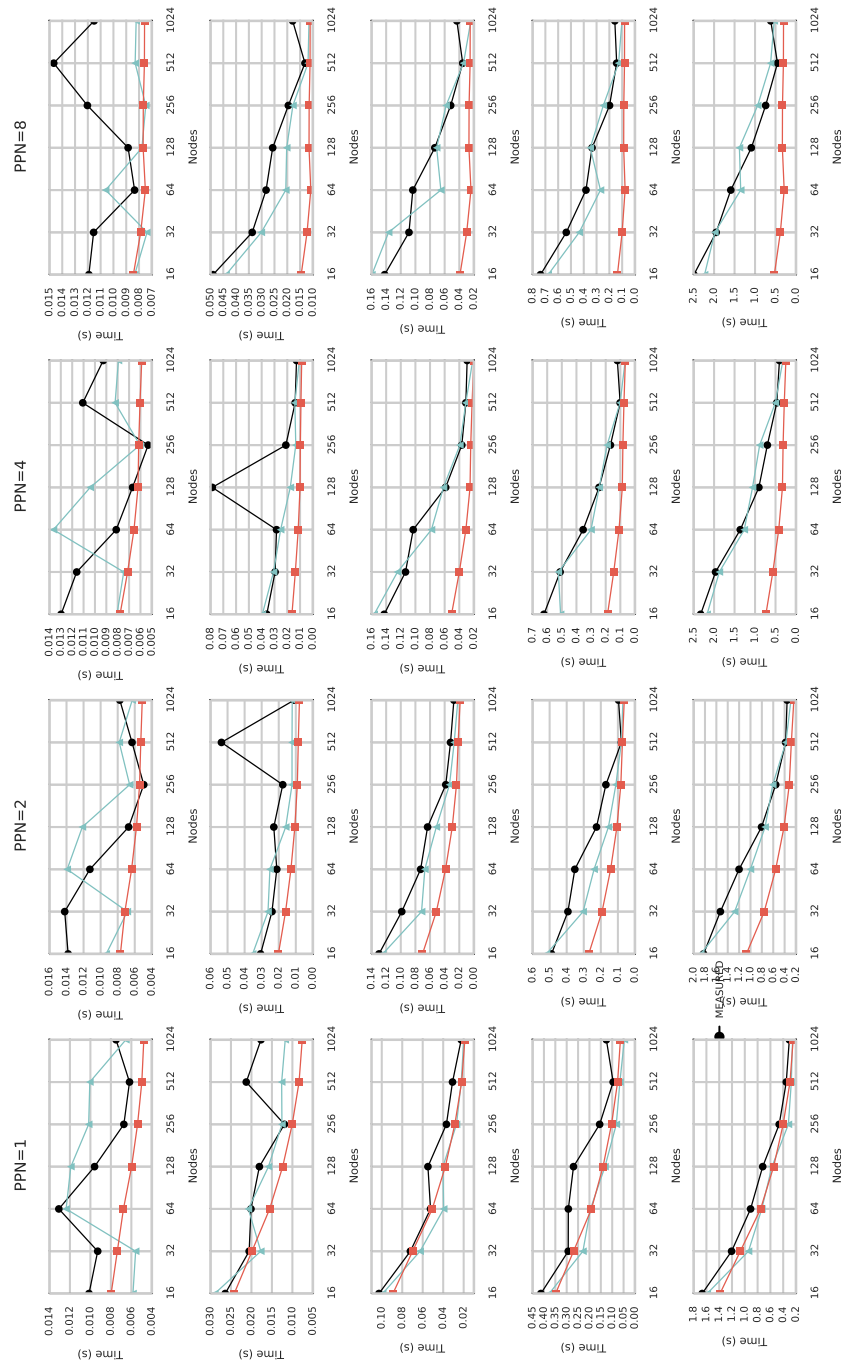


Figure 16: Predictions for *Halo-3D* (execution #3) for all problem sizes on Piz Daint with *GBRT-Class C* and the $\alpha_p - \beta_p - \gamma$ Model

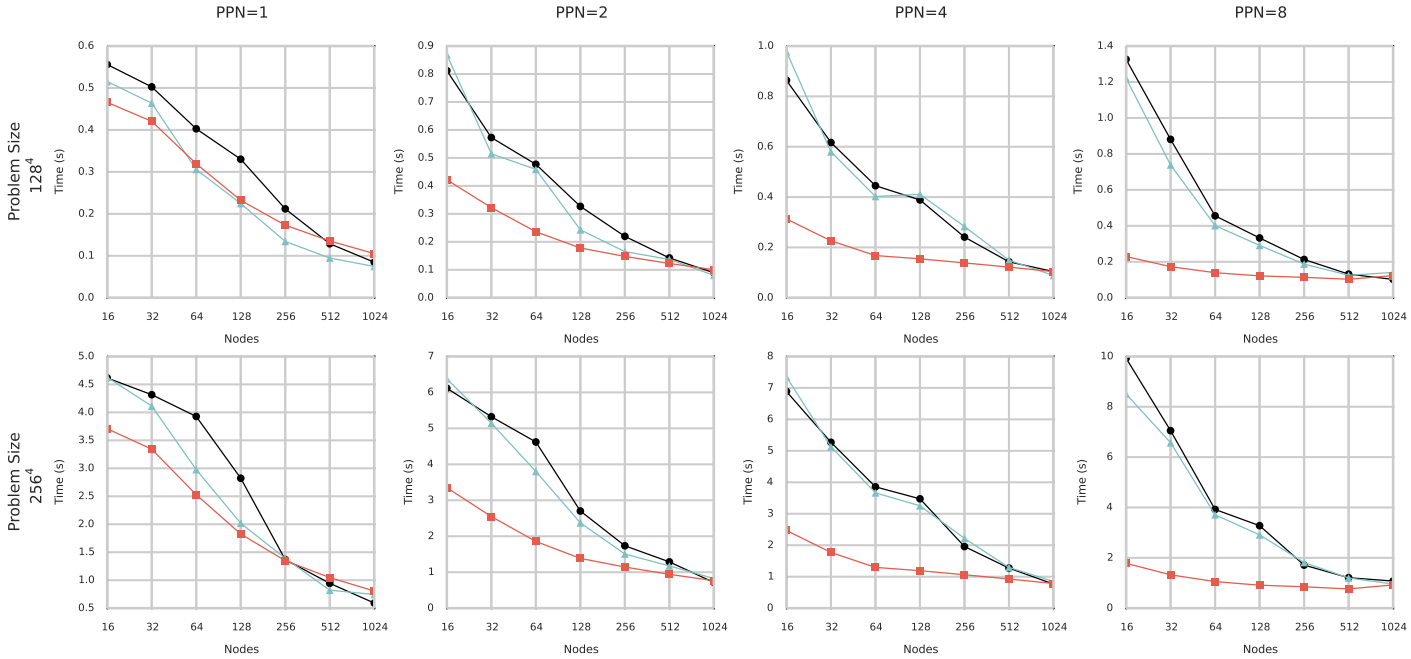


Figure 17: Predictions for *Halo-4D* (execution #1) for all problem sizes on Piz Daint with *GBRT-Class C* and the $\alpha_p - \beta_p - \gamma$ Model

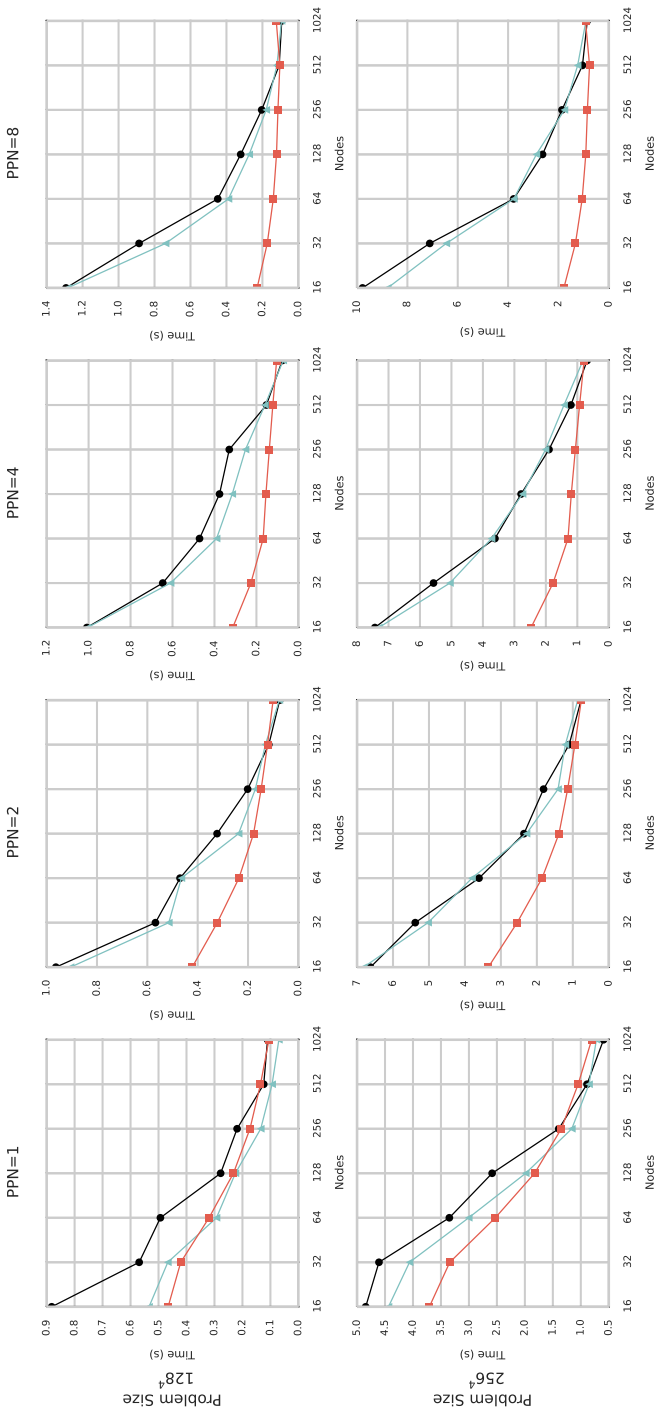
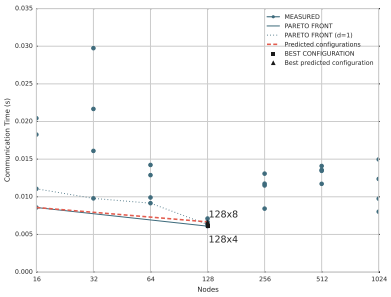
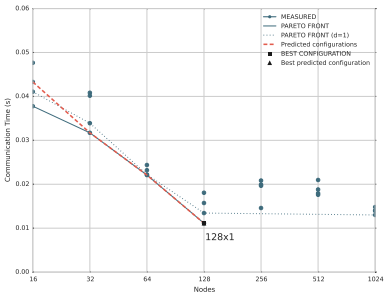


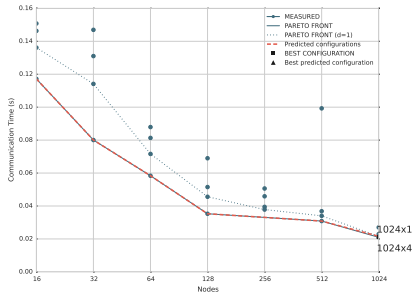
Figure 18: Predictions for *Halo-4D* (execution #2) for all problem sizes on Piz Daint with *GBRT-Class C* and the $\alpha_p - \beta_p - \gamma$ Model



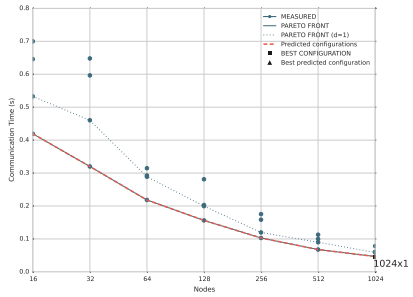
(a) Problem size: 128^3



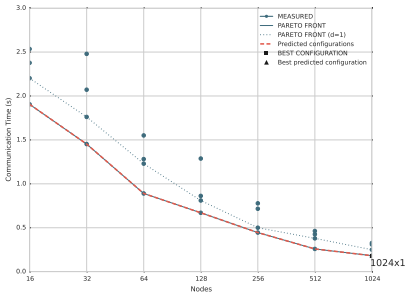
(b) Problem size: 256^3



(c) Problem size: 512^3

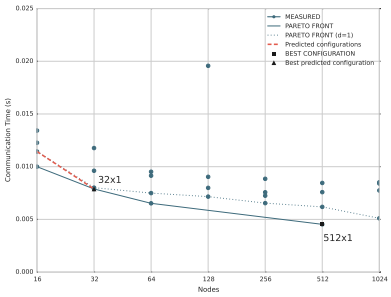


(d) Problem size: 1024^3

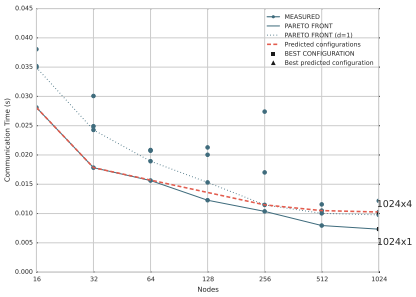


(e) Problem size: 2048^3

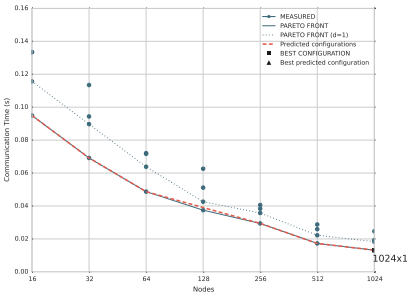
Figure 19: Pareto-optimal configuration predictions for *Halo-3D* (execution #1) on Piz Daint



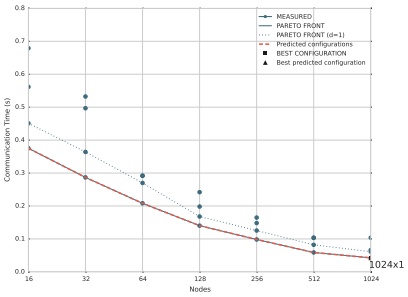
(a) Problem size: 128^3



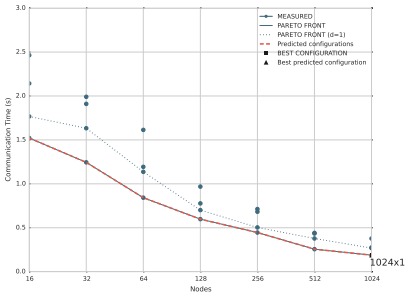
(b) Problem size: 256^3



(c) Problem size: 512^3

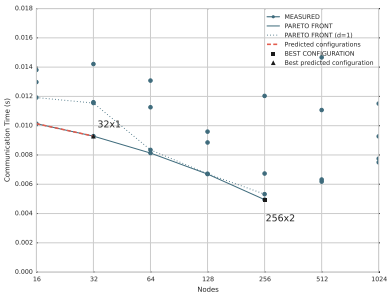


(d) Problem size: 1024^3

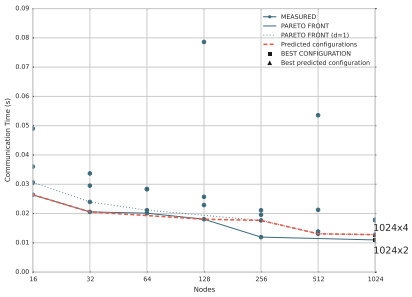


(e) Problem size: 2048^3

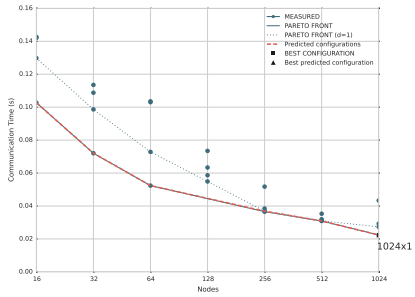
Figure 20: Pareto-optimal configuration predictions for *Halo-3D* (execution #2) on Piz Daint



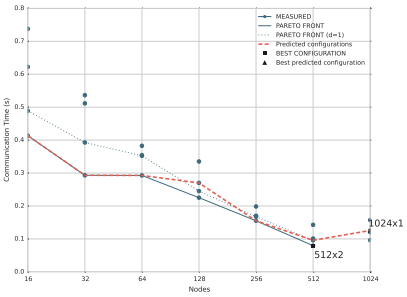
(a) Problem size: 128^3



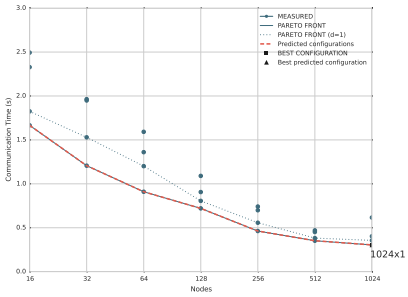
(b) Problem size: 256^3



(c) Problem size: 512^3

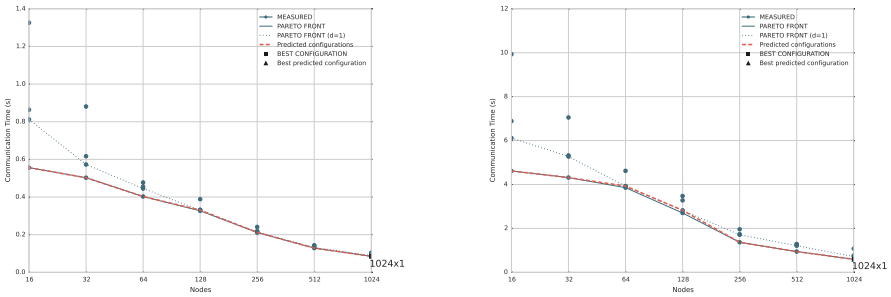


(d) Problem size: 1024^3



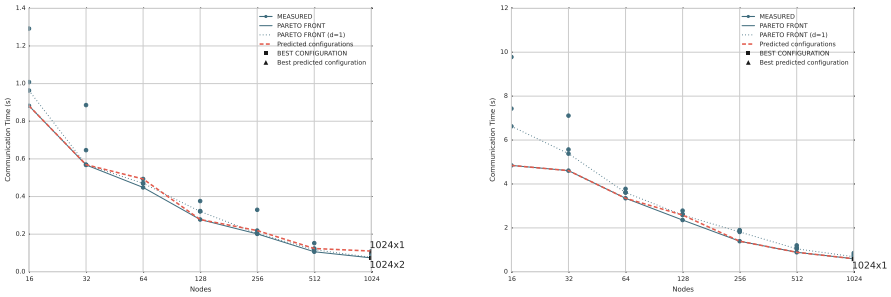
(e) Problem size: 2048^3

Figure 21: Pareto-optimal configuration predictions for *Halo-3D* (execution #3) on Piz Daint



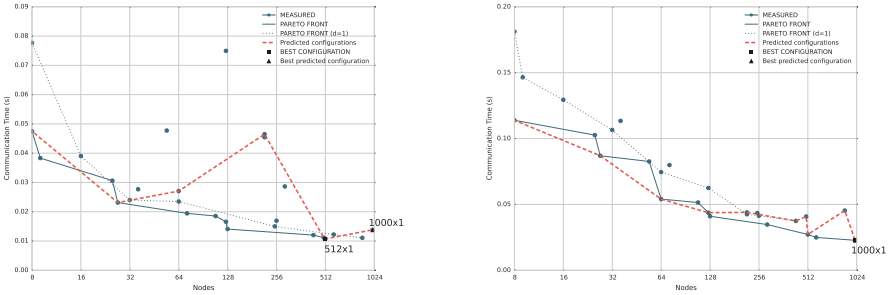
(a) Problem size: 128^4 **(b)** Problem size: 256^4

Figure 22: Pareto-optimal configuration predictions for *Halo-4D* (execution #1) on Piz Daint



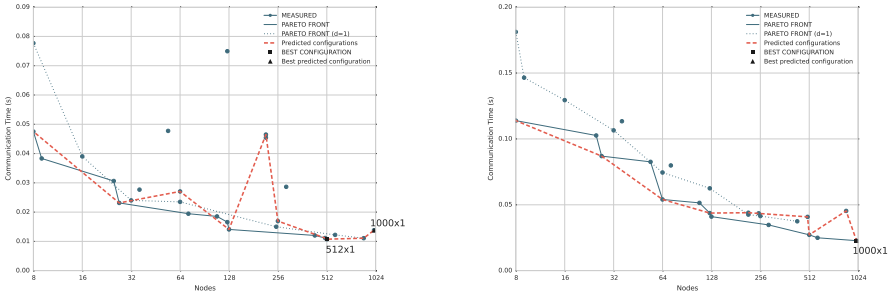
(a) Problem size: 128^4 **(b)** Problem size: 256^4

Figure 23: Pareto-optimal configuration predictions for *Halo-4D* (execution #2) on Piz Daint



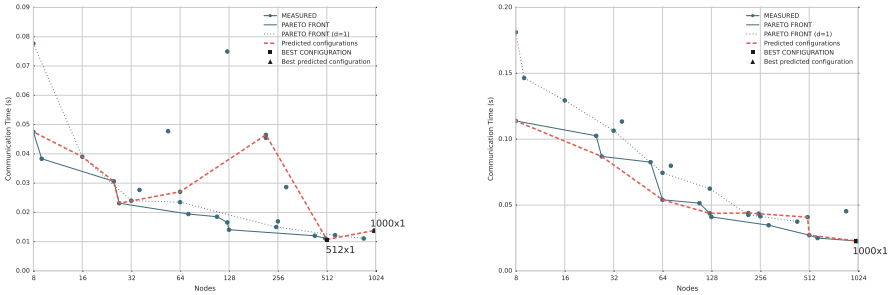
(a) Problem size: 240^3 **(b)** Problem size: 480^3

Figure 24: Pareto-optimal configuration predictions for *LULESH-1* on Piz Daint



(a) Problem size: 240^3 **(b)** Problem size: 480^3

Figure 25: Pareto-optimal configuration predictions for *LULESH-2* on Piz Daint

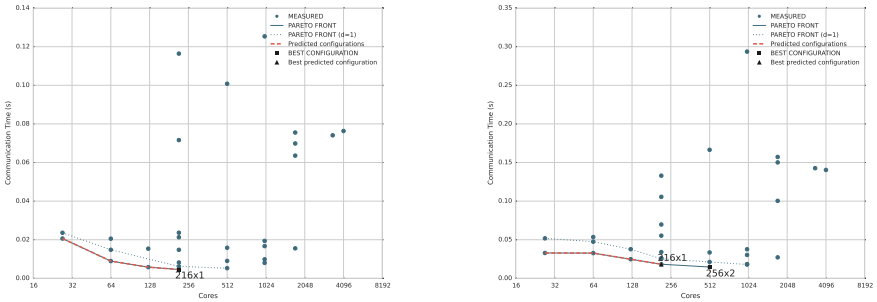


(a) Problem size: 240^3 **(b)** Problem size: 480^3

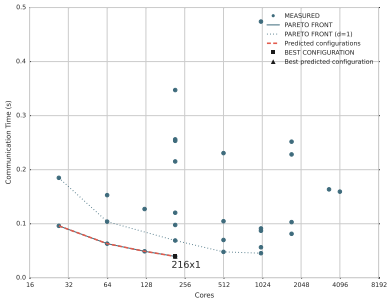
Figure 26: Pareto-optimal configuration predictions for *LULESH-3* on Piz Daint

Appendix C

In this appendix, we present the plots for Pareto-optimal configurations predictions with *GBRT-Class C* on the *ARIS* cluster, as supplemental material to Chapter 6, where we indicatively present two cases.

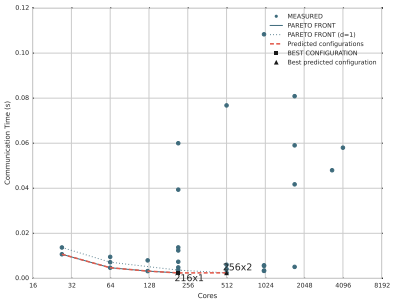


(a) Problem size: 120^3 **(b)** Problem size: 240^3

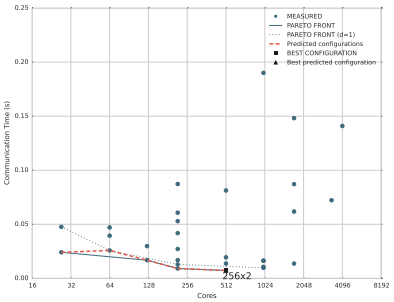


(c) Problem size: 480^3

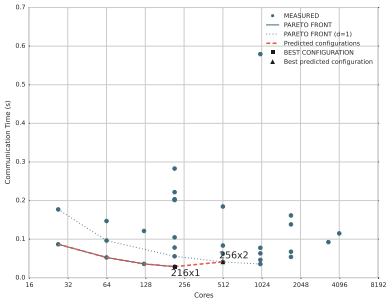
Figure 27: Pareto-optimal configuration predictions for *LULESH-1* on ARIS.



(a) Problem size: 120^3

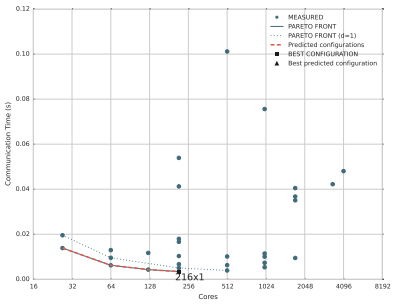


(b) Problem size: 240^3

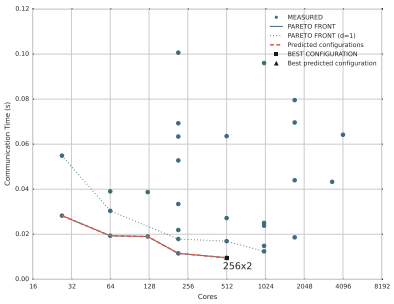


(c) Problem size: 480^3

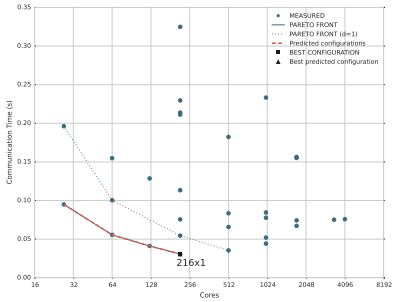
Figure 28: Pareto-optimal configuration predictions for *LULESH-2* on ARIS.



(a) Problem size: 120^3

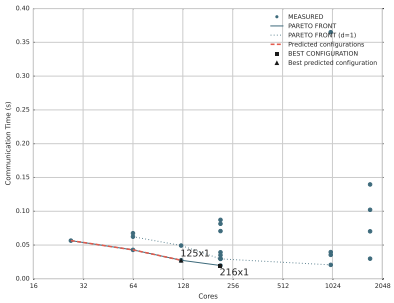


(b) Problem size: 240^3

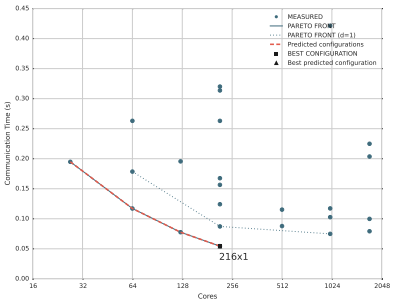


(c) Problem size: 480^3

Figure 29: Pareto-optimal configuration predictions for *LULESH-3* on ARIS.

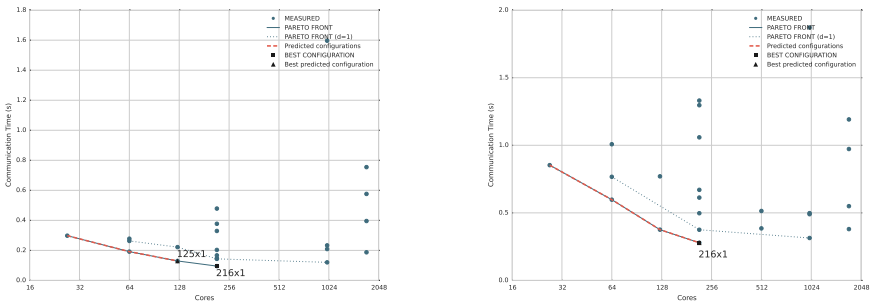


(a) Problem size: 480^3



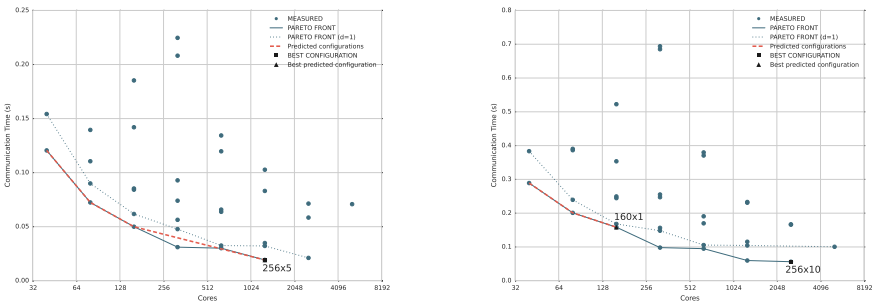
(b) Problem size: 960^3

Figure 30: Pareto-optimal configuration predictions for *HPCG-SpMV* on ARIS.



(a) Problem size: 480^3 **(b)** Problem size: 960^3

Figure 31: Pareto-optimal configuration predictions for *HPCG-MG* on ARIS.



(a) Problem size: $32 \times 32 \times 32 \times 40$ **(b)** Problem size: $32 \times 32 \times 64 \times 80$

Figure 32: Pareto-optimal configuration predictions for *QCD-Kernel D* on ARIS.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Πρόβλεψη επίδοσης επικοινωνίας σε συστήματα μεγάλης κλίμακας

Διδακτορική Διατριβή

Νικέλα Παπαδοπούλου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Η/Υ

Αθήνα
Ιούλιος, 2017

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Πρόβλεψη επίδοσης επικοινωνίας σε συστήματα μεγάλης κλίμακας

Διδακτορική Διατριβή

Νικέλα Παπαδοπούλου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Η/Υ

Συμβουλευτική Επιτροπή: Γεώργιος Γκούμας
Νεκτάριος Κοζύρης
Παναγιώτης Τσανάκας

Εγκρίθηκε από την επταμελή επιτροπή την 20ή Ιουλίου 2017.

Γεώργιος Γκούμας Επ. Καθηγητής Ε.Μ.Π.	Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Παναγιώτης Τσανάκας Καθηγητής Ε.Μ.Π.
Ανδρέας-Γεώργιος Σταφυλοπάτης Καθηγητής Ε.Μ.Π.	Δημήτριος Τσουμάκος Αν. Καθηγητής Ιονίου Πανεπιστημίου	Νικόλαος Πλέρος Επ. Καθηγητής Α.Π.Θ.
	Holger Fröning Juniorprofessor RKU Heidelberg	

Αθήνα
Ιούλιος, 2017

Νικέλα Παπαδοπούλου
Διδάκτωρ Εθνικού Μετσοβίου Πολυτεχνείου

Η εκπόνηση της διατριβής χρηματοδοτήθηκε από το Ίδρυμα Κρατικών Υποτροφιών στο πλαίσιο των υποτροφιών αριστείας ΙΚΥ Β' Κύκλου Σπουδών (Διδακτορικό) στην Ελλάδα - Πρόγραμμα Siemens.

Στους γονείς μου

Περίληψη

Οδεύοντας προς την εποχή των υπερυπολογιστικών συστημάτων με επιδόσεις της τάξης των ExaFlops, οι υπερυπολογιστές θα αποτελούνται από εκατοντάδες εκατομμύρια πυρήνες και διάφορα σύνθετα ετερογενή επεξεργαστικά στοιχεία. Ωστόσο, ήδη σήμερα, οι χρήστες αποτυγχάνουν να αξιοποιήσουν την υπάρχουσα υπολογιστική ισχύ των συστημάτων μεγάλης κλίμακας, όπως συμβαίνει με μεγάλες κατηγορίες παράλληλων εφαρμογών μεγάλης κλίμακας, η επίδοση των οποίων περιορίζεται από φάσεις επικοινωνίας που δεν κλιμακώνουν. Η δυνατότητα πρόβλεψης του χρόνου επικοινωνίας των παράλληλων εφαρμογών μπορεί να βοηθήσει τους χρήστες, τους μεταγλωττιστές, τα συστήματα χρόνου εκτέλεσης και τους χρονοδρομολογητές στη λήψη αποφάσεων για βέλτιστη χρήση πόρων, βελτιστοποιήσεις επιδόσεων, εξοικονόμηση ενέργειας και ελαστικότητα σε σφάλματα.

Η παρούσα διατριβή παρουσιάζει μια μεθοδολογία για την μοντελοποίηση της επικοινωνίας των παράλληλων εφαρμογών μεγάλης κλίμακας με στόχο την πρόβλεψη. Ο χρόνος επικοινωνίας εξαρτάται από ένα πολύπλοκο σύνολο παραμέτρων, σχετικών με την εφαρμογή, την αρχιτεκτονική του συστήματος, τις ρυθμίσεις χρόνου εκτέλεσης και τις συνθήκες εκτέλεσης. Για την ενσωμάτωση αυτής της πολυπλοκότητας σε ένα μοντέλο πρόβλεψης, ακολουθούμε μια προσέγγιση εμπειρικής μοντελοποίησης. Ορίζουμε χαρακτηριστικά που μπορούν να εξαχθούν από την εφαρμογή, την απεικόνιση των διεργασιών στο σύστημα και το σχήμα κατανομής των υπολογιστικών πόρων, πριν από την εκτέλεση, αναπτύσσουμε ένα πρόγραμμα μετρήσεων αναφοράς για τη σάρωση του χώρου των παραμέτρων, και αναπτύσσουμε μοντέλα πρόβλεψης για τον χρόνο επικοινωνίας σε τρία υπολογιστικά συστήματα μεγάλης κλίμακας, τα συστήματα Vilje, Piz Daint και ARIS, χρησιμοποιώντας διαφορετικά υποσύνολα των χαρακτηριστικών μας, μεθόδους στατιστικής και μηχανικής μάθησης και διάφορα σύνολα εκπαίδευσης. Συγκρίνουμε την πρόβλεψη των μοντέλων μας σε διάφορα σχήματα επικοινωνίας και εφαρμογές, για πολλαπλά μεγέθη προβλημάτων, πολλαπλές εκτελέσεις και διαφορετικές ρυθμίσεις του χρόνου εκτέλεσης, που κυμαίνονται από μερικές δεκάδες έως μερικές χιλιάδες πυρήνες. Η μεθοδολογία μας είναι επιτυχής στην πρόβλεψη του χρόνου επικοινωνίας σε όλα τα σχήματα επικοινωνίας που εξετάζουμε, σε όλα τα συστή-

ματα, και παρουσιάζει υψηλή ακρίβεια πρόβλεψης και καλή προσαρμογή. Τα μοντέλα που προτείνονται αποδίδουν προβλέψεις ακριβώς πριν από την εκτέλεση μίας παράλληλης εφαρμογής και, όπως καταδεικνύουμε σε αυτή τη διατριβή, η υψηλή ακρίβεια τους τα καθιστά κατάλληλα για λήψη αποφάσεων με επίγνωση της επικοινωνίας, προς την κατεύθυνση της βελτιστοποίησης της χρήσης των υπολογιστικών πόρων σε συστήματα μεγάλης κλίμακας.

Λέξεις Κλειδιά: Μοντελοποίηση επίδοσης, Πρόβλεψη επίδοσης, Χρόνος επικοινωνίας, Παράλληλες εφαρμογές, MPI, Υπερυπολογιστές, Συστοιχίες υπολογιστών, Στατιστική μάθηση, Μηχανική μάθηση

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κατάλογος Αλγορίθμων

Αντί προλόγου

Η παρούσα διατριβή εκπονήθηκε στο Εργαστήριο Υπολογιστικών Συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Συγκεντρώνει την έρευνα και τα αποτελέσματα των πενταετών μεταπτυχιακών μου σπουδών στη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του ΕΜΠ. Καθώς αυτός ο κύκλος κλείνει, θα ήθελα να ευχαριστήσω τους ανθρώπους που συνέβαλαν στην ολοκλήρωση αυτής της διατριβής.

Πρώτον, θα ήθελα να ευχαριστήσω τον επιβλέποντά μου, Επίκουρο Καθηγητή Γεώργιο Γκούμα, για τη συμβολή του σε πολλαπλά επίπεδα. Η έναρξη και η ολοκλήρωση αυτού του κύκλου σπουδών οφείλεται πρώτα στην εμπιστοσύνη που μου έδειξε. Ως καθηγητής στις προπτυχιακές σπουδές μου, μου μετέδωσε τον ενθουσιασμό του για τα ερευνητικά του ενδιαφέροντα. Ως επιβλέπων καθηγητής, μοιράστηκε με γενναιοδωρία τις γνώσεις του, την ερευνητική του μεθοδολογία και την ιδιαίτερη προσέγγισή του στην επίλυση προβλημάτων. Καθοδήγησε, βοήθησε και υποστήριξε τη δουλειά μου και πάντα έβρισκε τον τρόπο να με παρακινεί και να με ενθαρρύνει προς την υπέρβαση των όποιων εμποδίων. Η ολοκλήρωση αυτής της διατριβής δε θα ήταν εφικτή χωρίς την προσπάθεια και το χρόνο που κατέβαλε στην επίβλεψή της. Τον ευχαριστώ θερμά για όλα.

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον Καθηγητή Νεκτάριο Κοζύρη, μέλος της τριμελούς συμβουλευτικής μου επιτροπής, για την εμπιστοσύνη του και την υποστήριξή του, την επιστημονική καθοδήγηση και την προθυμία του να συμβάλει με οποιοδήποτε τρόπο στην ολοκλήρωση αυτής της διατριβής. Θα ήθελα επίσης να ευχαριστήσω τον Καθηγητή Παναγιώτη Τσανάκα, μέλος της τριμελούς συμβουλευτικής μου επιτροπής, για τη βοήθειά του σε όλη τη διάρκεια των σπουδών μου. Ευχαριστώ τους καθηγητές Ανδρέα-Γεώργιο Σταφυλοπάτη, Δημήτριο Τσουμάκο, Νικόλαο Πλέρο και Holger Frön-

ing για τη συμμετοχή τους στην επιτροπή εξέτασης της διδακτορικής μου διατριβής.

Κατά τη διάρκεια των μεταπτυχιακών σπουδών μου, σημείο αναφοράς υπήρξε το Εργαστήριο Υπολογιστικών Συστημάτων. Η καθημερινή μου συνύπαρξη με όλα τα μέλη του εργαστηρίου συνέβαλε καθοριστικά στη διαμόρφωσή μου ως ερευνήτριας. Ευχαριστώ, αρχικά, το Δρ. Κωστή Νίκα, για την προθυμία του να συζητήσει οποιοδήποτε θέμα και να βοηθήσει σε οποιοδήποτε πρόβλημα, ερευνητικό ή άλλο, ακόμα και ασήμαντο. Επομένως, τον ευχαριστώ και για την υπομονή του. Ευχαριστώ το Δρ. Αναστάσιο Νάνο, για τον ενθουσιασμό με τον οποίο μοιράζεται τις γνώσεις και τα ενδιαφέροντά του, καθώς και για το πνεύμα συνεργασίας και παρακίνησης που τον διακατέχει. Τον ευχαριστώ επίσης για την τρομερή υποστήριξη. Θέλω εξίσου να ευχαριστήσω τα μέλη της ερευνητικής μας ομάδας, Δημήτρη Σιακαβάρα, Αθηνά Ελαφρού, Στέφανο Γεράγγελο, Χλόη Αλβέρτη, Κωστή Παπαζαφειρόπουλο, Χριστίνα Γιαννούλα, Δρ. Βασίλη Καρακώστα, Στράτο Ψωμαδάκη, Αλέξανδρο Χαριτάτο, Τάσο Κατσιγιάννη και Δρ. Νίκο Αναστόπουλο, καθώς και όλους τους συναδέλφους του εργαστηρίου, υποψήφιους διδάκτορες, μεταδιδακτορικούς ερευνητές και διοικητικούς υπαλλήλους του εργαστηρίου, για την ανταλλαγή απόψεων, τη συνδρομή τους σε μικρά ή μεγάλα τεχνικά ζητήματα, τις ενδιαφέρουσες εβδομαδιαίες συζητήσεις και, φυσικά, για την παρέα. Επιπλέον, ευχαριστώ το Βασίλη και τη Χλόη που αφιέρωσαν χρόνο στην ανάγνωση και αποσφαλμάτωση αυτής της διατριβής.

Ως υποψήφια διδάκτορας, βοήθησα στην επίβλεψη διπλωματικών εργασιών στο Εργαστήριο Υπολογιστικών Συστημάτων, γεγονός που μου έδωσε την ευκαιρία να μελετήσω ζητήματα που σχετίζονται με το θέμα της διατριβής μου και να εξερευνήσω εναλλακτικές ερευνητικές κατευθύνσεις, καθώς και την ευκαιρία να συνεργαστώ με εξαιρετικούς φοιτητές και ενδιαφέροντες ανθρώπους. Ευχαριστώ ιδιαίτερα το Σωτήρη Αποστολάκη, για την εξαιρετική του δουλειά και τη συμβολή του στα πρώιμα στάδια αυτής της διατριβής. Ευχαριστώ επίσης τους Παναγιώτη Μουλλωτού, Φοίβο Κοτοματά, Γιώργο Χριστοδουλή και Γιώργο Ζαχαριάδη.

Από τον Απρίλιο έως και τον Αύγουστο του 2016, εργάστηκα ως βοηθός έρευνας στο Argonne National Laboratory στο Σικάγο των Η.Π.Α, ως μέλος της ομάδας Parallel Models and Runtime Systems. Θέλω να εκφράσω τις ευχαριστίες μου στον επικεφαλής της ομάδας, Δρ. Pavan Balaji,που μου έδωσε την ευκαιρία να ενταχθώ στην ομάδα του και να εργαστώ σε ένα εξαιρετικό περιβάλλον για την έρευνα. Στη διάρκεια αυτών των λίγων μηνών, αποκόμισα σημαντικές εμπειρίες από τις βαθιές του γνώσεις και τον τρόπο του να αντιμετωπίζει ερευνητικά προβλήματα. Εξίσου θέλω να ευχαριστήσω τη Δρ. Lena Oden, που καθοδήγησε την έρευνά μου στο Argonne National Laboratory και μου πρόσφερε απλόχερα τη βοήθειά της και το χρόνο της. Η συνεργασία μου

μαζί της ήταν μία απροσδόκητα ευχάριστη και αξιομνημόνευτη εμπειρία.

Η έρευνα που παρουσιάζεται σε αυτή τη διατριβή αφορά υπολογιστικά συστήματα μεγάλης κλίμακας και απαίτησε πρόσβαση σε αυτά. Ευχαριστώ τους φορείς NTNU, CSCS και ΕΔΕΤ, οι οποίοι μάς παρείχαν πρόσβαση στους υπερυπολογιστές τους και ευχαριστώ επίσης τους ανθρώπους που βοήθησαν ώστε να με φέρουν σε επαφή με τους παραπάνω οργανισμούς.

Τέλος, θέλω να ευχαριστήσω τους κοντινούς μου ανθρώπους για την ηθική και συναισθηματική υποστήριξη. Περισσότερο απ' όλους, ευχαριστώ τους γονείς μου, Άρη και Ανθούλα, για την άνευ όρων αγάπη, υποστήριξη και ενθάρρυνση, όχι μόνο τα τελευταία πέντε, αλλά τα τελευταία εικοσιοκτώ χρόνια. Σ'αυτούς αφιερώνω αυτή τη διατριβή.

Κλείνοντας, θέλω να αναφέρω ότι η έρευνα που παρουσιάζεται σε αυτή τη διατριβή διεξήχθη σε ένα ανοιχτό, δημόσιο πανεπιστήμιο, το οποίο, παρά το στραγγαλισμό της χρηματοδότησης για την έρευνα κατά τα πρόσφατα χρόνια της κρίσης, συνεχίζει να παράγει υψηλής ποιότητας ερευνητικά αποτελέσματα και να μην παρεμβαίνει στον προσανατολισμό της έρευνας. Στο πλαίσιο αυτών των πρόσφατων χρόνων, θεωρώ σημαντικό να ευχαριστήσω όσους αγωνίζονται για να κρατήσουν τα πανεπιστήμια δημόσια και ανοιχτά για όλους. Ελπίζω και εύχομαι ότι ο δημόσιος χαρακτήρας της έρευνας και της παιδείας θα ενισχυθεί στα επόμενα χρόνια, και ότι η προσπάθεια και η συνεισφορά των ανθρώπων που εργάζονται σε αυτές, ακόμα και σε αντίξοες συνθήκες, θα αναγνωριστεί.

1

Εισαγωγή

1.1 Κίνητρο της διατριβής

Οι υπερυπολογιστές πρωτοεμφανίστηκαν τη δεκαετία του 1960, ως απάντηση στην ανάγκη της επιστημονικής κοινότητας να επιλύσει επιστημονικά προβλήματα που υπερβαίνουν κατά πολύ την υπολογιστική ισχύ ενός μόνο επεξεργαστή. Στις αρχές της δεκαετίας του 1990, αναπτύχθηκαν συστήματα μεγάλης κλίμακας χιλιάδων επεξεργαστών. Από τότε, κάθε 1,5 χρόνια, η υπολογιστική ισχύς των υπερυπολογιστών διπλασιάζεται, ενώ αντίστοιχα αυξάνεται το πλήθος των επεξεργαστών. Στα μέσα του 2016, κατασκευάστηκε ο κινεζικός υπερυπολογιστής Sunway TaihuLight, ο οποίος περιλαμβάνει περισσότερα από 10 εκατομμύρια πυρήνες και επιτυγχάνει επίδοση που ξεπερνά τα 90 PetaFlops (90×10^{15} πράξεις κινητής υποδιαστολής ανά δευτερόλεπτο) [top]. Ο επόμενος σταθμός στην ταχύτατη εξελικτική πορεία των υπερυπολογιστών είναι η επίτευξη επίδοσης που θα ξεπερνά το 1 ExaFlop (10^{18} πράξεις κινητής υποδιαστολής ανά δευτερόλεπτο) και προβάλλεται για το 2022.

Οι προκλήσεις στην επίτευξη επίδοσης της τάξης των ExaFlops είναι μεγάλες. Η ανάγκη περιορισμού της κατανάλωσης ισχύος στα 20MW και του κόστους κεφαλαίου στα 200 εκατομμύρια δολάρια οδηγεί σημαντικές αλλαγές στην αρχιτεκτονική και στο λογισμικό των συστημάτων. Οι υπολογιστικοί κόμβοι της επόμενης γενιάς θα περιλαμβάνουν εκατοντάδες επεξεργαστές και πολλά περισσότερα υπολογιστικά νήματα, καθώς και επιταχυντές γενικού και ειδικού σκοπού, σε μια προσπάθεια να αντλήσουν επίδοση από τον παραλληλισμό και την ετερογένεια, καθώς η συχνότητα του επεξεργαστή δεν μπορεί πλέον να αυξηθεί και η κατανάλωση ενέργειας ανά κόμβο πρέπει να περιοριστεί [Kogge and Shalf, 2013]. Η μνήμη ανά κόμβο θα αυξηθεί, αλλά μόνο κατά ένα μικρό κλάσμα της αύξησης των Flops, λόγω των περιορισμών στο κόστος κεφαλαίου και της αργής αύξησης της πυκνότητας μνήμης [McMor-

row, 2013]. Το εύρος ζώνης του κόμβου θα αυξηθεί επίσης μόνο μετρίως, λόγω των περιορισμών στην κατανάλωση ενέργειας. Τελικά, οι υπολογιστικοί κόμβοι επόμενης γενιάς θα περιλαμβάνουν βαθιές ιεραρχίες μνήμης, όπως κρυφές μνήμες, 3Δ-συσσωρευμένες μνήμες και μη διαγραφόμενες μνήμες (NVRAM), για να μετριάσουν την κατανάλωση ισχύος [Shalf et al., 2010]. Παρόλα αυτά, παρά τις αλλαγές στην αρχιτεκτονική των κόμβων, η επίδοση των κόμβων αναμένεται να αυξηθεί κατά 10 φορές. Για να επιτευχθεί επίδοση της τάξης των ExaFlops, οι υπερυπολογιστές θα φιλοξενούν χιλιάδες κόμβους, διασυνδεδεμένους με δίκτυα υψηλής επίδοσης, με χαμηλό χρόνο απόκρισης και υψηλό εύρος ζώνης. Τα δίκτυα διασύνδεσης πρέπει να πληρούν τις απαιτήσεις επικοινωνίας του κόμβου (των εκατοντάδων νημάτων) και του συστήματος (των χιλιάδων κόμβων) ταυτόχρονα. Οι περιορισμοί ισχύος και επίδοσης για τα δίκτυα διασύνδεσης ενθαρρύνουν τη μετατόπιση της τεχνολογίας προς τη φωτονική πυριτίου. Επιπλέον, τα στοιχεία του δικτύου επανασχεδιάζονται, έτσι ώστε το υλικό να μπορεί να ανταποκριθεί στις απαιτήσεις επικοινωνίας των εφαρμογών. Παρόλα αυτά, το κόστος της επικοινωνίας για εκατομμύρια νήματα μπορεί να είναι συντριπτικό και οι τοπολογίες του δικτύου διασύνδεσης είναι επίσης κρίσιμες για την επίτευξη υψηλών επιδόσεων. Έτσι, οι τοπολογίες χαμηλής διαμέτρου, που μπορούν να παρέχουν υψηλό εύρος τομής και είναι κατανεμητέες, θα κυριαρχούν στα μελλοντικά συστήματα [Shalf et al., 2010; Jain et al., 2017]. Το λογισμικό συστήματος εξελίσσεται επίσης ταχύτατα για να προσαρμοστεί στο νέο υλικό. Τα λειτουργικά συστήματα, οι διεπαφές δικτύου και το λογισμικό επικοινωνίας, οι μεταγλωττιστές, τα προγραμματιστικά μοντέλα, τα λογισμικά διαχείρισης πόρων, οι βιβλιοθήκες των εφαρμογών και τα εργαλεία των εφαρμογών, που αποτελούν το πλούσιο οικοσύστημα του λογισμικού των υπερυπολογιστών, επαναπροσδιορίζονται για να ανταποκρίνονται στις απαιτήσεις της επίδοσης των ExaFlops. Μία από αυτές τις απαιτήσεις είναι η ανθεκτικότητα σε σφάλματα, όπου το λογισμικό αναμένεται να βοηθά το υλικό. Συνολικά, τα υπολογιστικά συστήματα της επόμενης γενιάς, της γενιάς exascale, θα χρειαστεί να αντιμετωπίσουν προβλήματα που σχετίζονται με την κατανάλωση ισχύος, που οδηγεί τις αλλαγές στο υλικό, την ανθεκτικότητα σε σφάλματα και την κίνηση δεδομένων. Το τελευταίο είναι ιδιαίτερα σημαντικό για την επίδοση: τελικά, η επίδοση της τάξης των ExaFlops θα προέλθει από την ικανότητα των παράλληλων εφαρμογών να κλιμακώνουν εντός του κόμβου, στις εκατοντάδες των νημάτων, και εκτός του κόμβου, στις χιλιάδες των κόμβων των νέων υπερυπολογιστικών συστημάτων.

Οι παράλληλες εφαρμογές που εκτελούνται σε υπερυπολογιστές είναι προσομοιώσεις του πραγματικού κόσμου και αποτελούνται από πολλαπλούς υπολογιστικούς πυρήνες, οι οποίοι παρουσιάζουν διάφορες φάσεις υπολογισμών και επικοινωνίας, καθώς και είσοδο/έξοδο αρχείων, για την αρχικοποίηση της προσομοίωσης ή για τον έλεγχο και την επανεκκίνηση της εφαρμογής. Η πα-

ράλληλη επίδοση εξαρτάται σε μεγάλο βαθμό από την ικανότητα όλων των φάσεων να κλιμακώνουν αποτελεσματικά. Οι φάσεις των υπολογισμών αναμένεται να κλιμακώνουν επαρκώς στα συστήματα μεγάλης κλίμακας, αξιοποιώντας τις υπολογιστικές δυνατότητες των απλών πυρήνων, των διανυσματικών επεξεργαστών, των ετερογενών επιταχυντών και του εξειδικευμένου υλικού. Ωστόσο, για την επίτευξη της επίδοσης του ExaFlop, οι παράλληλες εφαρμογές θα εκτελούνται σε εκατομμύρια υπολογιστικών νημάτων, τα οποία θα απαιτούν επικοινωνία τόσο εντός του κόμβου όσο και με απομακρυσμένους κόμβους, μέσω των δικτύων διασύνδεσης. Συνεπώς, οι φάσεις επικοινωνίας θα επιβαρυνθούν με περισσότερη ανταλλαγή μηνυμάτων ή/και μεγαλύτερο όγκο δεδομένων. Παρά την εξέλιξη της τεχνολογίας όσον αφορά τα δίκτυα διασύνδεσης, το κόστος επικοινωνίας μπορεί να αυξηθεί τόσο ώστε να αντισταθμιστεί η δυνατότητα κλιμάκωσης των υπολογισμών, καθιστώντας έτσι την επικοινωνία σοβαρό εμπόδιο για την επίτευξη επίδοσης της τάξης των ExaFlops. Αυτό αντικατοπτρίζει ένα από τις τρία θεμελιώδη εμπόδια για την επίδοση ExaFlop, που είναι η κίνηση δεδομένων.

Οι προβλέψεις για το χρόνο επικοινωνίας των παράλληλων εφαρμογών μπορούν να υποστηρίξουν και να ενισχύσουν διάφορα σενάρια λήψης αποφάσεων. Οι χρήστες των υπερυπολογιστών θα είναι σε θέση να προβλέψουν την κλιμακωσιμότητα των εφαρμογών τους και να δεσμεύσουν το βέλτιστο αριθμό και συνδυασμό πόρων, αποφεύγοντας τη σπατάλη πολύτιμων πόρων και τους μεγάλους χρόνους αναμονής. Επιπλέον, η δυνατότητα επιλογής μεταξύ των κατάλληλων κατανομών πόρων, δηλαδή του αριθμού των κόμβων, των πυρήνων, ακόμη και των δικτυακών στοιχείων, είναι κρίσιμη για τη βελτιστοποίηση της επίδοσης [Goumas et al., 2009], της κατανάλωσης ενέργειας [Brooks et al., 2000; Bekas and Curioni, 2010] και της ανοχής σε σφάλματα [Li et al., 2014; Yu et al., 2014]. Για παράδειγμα, μια πρόβλεψη για το χρόνο επικοινωνίας μπορεί να συμβάλει στην επιλογή βέλτιστων συνδυασμών υπολογιστικών πόρων, σε σχέση με την ενέργεια ή/και την ανθεκτικότητα σε σφάλματα, και να ενισχύσει τις πολιτικές δέσμευσης πόρων και τις πολιτικές χρονοδρομολόγησης στους κόμβους ενός υπερυπολογιστή. Επιπλέον, ένα μοντέλο για την πρόβλεψη της επικοινωνίας μπορεί να υποστηρίξει τη βελτιστοποίηση και τη ρύθμιση κώδικα, για ένα ευρύ φάσμα τεχνικών βελτιστοποίησης της επικοινωνίας, όπως η επικάλυψη υπολογισμών/επικοινωνίας, η συμπίεση μηνυμάτων [Filgueira et al., 2011], η αποφυγή επικοινωνίας [Solomonik and Demmel, 2011] και η χρήση υβριδικών προγραμματιστικών μοντέλων, όπως τα MPI/ OpenMP [Drosinos and Koziris, 2004]. Στην πραγματικότητα, ένα αποτελεσματικό μοντέλο επικοινωνίας μπορεί να χρησιμοποιηθεί κατά το χρόνο εκτέλεσης, για την αυτόματη ρύθμιση εφαρμογών, την ενεργοποίηση και την απενεργοποίηση διαφόρων βελτιστοποιήσεων. Τέλος, τα μοντέλα επίδοσης επικοινωνίας αποτελούν βασική συνιστώσα σε προσομοιωτές που βασίζονται

σε ίχνη (traces) παράλληλων εφαρμογών, όπως το SimGrid [Casanova et al., 2015] και το SST/Macro [sst].

Η επίδοση της επικοινωνίας εξαρτάται από πολλούς παράγοντες και, ως εκ τούτου, η ακριβής μοντελοποίησή της θέτει σημαντικές προκλήσεις. Ο χρόνος επικοινωνίας για ένα συγκεκριμένο σχήμα επικοινωνίας σε ένα συγκεκριμένο σύστημα εξαρτάται σε μεγάλο βαθμό από το σχήμα κίνησης δεδομένων (traffic pattern), που παράγεται από την απεικόνιση του γράφου επικοινωνίας της εφαρμογής στην κατανομή εκείνων των πόρων του συστήματος που έχουν δεσμευθεί για την εκτέλεση της εφαρμογής, καθώς και από την αρχιτεκτονική του συστήματος και τις ιδιότητες της τοπολογίας του δικτύου. Η κίνηση δεδομένων που προκύπτει από μία μόνο εφαρμογή μπορεί να είναι αρκετή ώστε να προκαλέσει φαινόμενα ανταγωνισμού στα δικτυακά στοιχεία και συμφόρηση στις συνδέσεις του δικτύου, με αντίστοιχες καθυστερήσεις. Ωστόσο, σε συστήματα μεγάλης κλίμακας, όπου πολλαπλές εφαρμογές εκτελούνται από κοινού, αυτές ανταγωνίζονται για κοινόχρηστους πόρους και είναι σε θέση να προκαλέσουν απρόσμενες καθυστερήσεις στον χρόνο επικοινωνίας των συνεκτελούμενων εφαρμογών. Η αρχιτεκτονική κάθε συστήματος, το λογισμικό επικοινωνίας, το λειτουργικό σύστημα και το λογισμικό διαχείρισης πόρων επηρεάζουν τον χρόνο επικοινωνίας των εφαρμογών με πολύπλοκο τρόπο. Επίσης, οι φάσεις υπολογισμού των εφαρμογών επηρεάζουν τον χρόνο επικοινωνίας, καθώς μπορούν να επικαλύπτονται με την επικοινωνία ή να λοξεύουν το χρόνο επικοινωνίας εξαιτίας της ανισορροπίας φορτίου.

Στην παρούσα διατριβή παρουσιάζουμε μια μεθοδολογία πρόβλεψης της επικοινωνίας των εφαρμογών μεγάλης κλίμακας. Το κύριο κίνητρο για αυτή την εργασία είναι η συμπεριφορά πολλών κατηγοριών εφαρμογών, οι οποίες εκμεταλλεύονται την αύξηση της επίδοσης των συστημάτων μεγάλης κλίμακας για ισχυρή κλιμάκωση (strong scaling). Τέτοιες κατηγορίες περιλαμβάνουν μερικές διαφορικές εξισώσεις (PDEs) σε δομημένα και αδόμητα πλέγματα, αλγόριθμους επεξεργασίας αδόμητων γράφων, κώδικες πολλαπλών κλιμάκων ή πολλαπλών μοντέλων και άλλες εφαρμογές. Οι εφαρμογές αυτών των κατηγοριών αναμένεται να διατηρήσουν το κόστος επικοινωνίας τους στους υπερυπολογιστές της επόμενης γενιάς, καθώς το κόστος κίνησης των δεδομένων θα αυξηθεί στον τεράστιο αριθμό των νημάτων, εφόσον το μοντέλο περάσματος μηνυμάτων παραμένει το επικρατέστερο προγραμματιστικό μοντέλο [Kogge et al., 2008]. Επιπλέον, οι εφαρμογές αυτών των κατηγοριών που δεν κλιμακώνουν γραμμικά θα διατηρήσουν τη μη γραμμική, μη βέλτιστη, συμπεριφορά κλιμάκωσης σε συστήματα υψηλότερων επιδόσεων, λόγω του κόστους επικοινωνίας. Συνεπώς, η δυνατότητα πρόβλεψης του χρόνου επικοινωνίας των παράλληλων εφαρμογών πριν από την εκτέλεσή τους είναι επιτακτική για την πρόβλεψη της αναμενόμενης επίδοσης και κλιμακωσιμότητάς τους, για τη βελτιστοποίησή τους, καθώς και για την αποδοτική χρήση και διαχείριση των

πόρων των συστημάτων της επόμενης γενιάς.

Η προσέγγισή μας στοχεύει στην πρόβλεψη του χρόνου για μια ολόκληρη φάση επικοινωνίας μιας εφαρμογής και προσφέρει προβλέψεις πριν από την εκτέλεση της εφαρμογής. Εξ όσων γνωρίζουμε, αυτή είναι η πρώτη προσέγγιση που παρέχει μοντέλα που προβλέπουν την επικοινωνία ενός ευρέος συνόλου σχημάτων επικοινωνίας από σημείο σε σημείο (point-to-point) χωρίς οποιαδήποτε προσαρμογή, πριν από την εκτέλεση, παρέχοντας σημαντική ακρίβεια πρόβλεψης, όπως καταδεικνύουν τα πειραματικά μας αποτελέσματα σε τρία συστήματα μεγάλης κλίμακας. Για να επιτύχουμε τα παραπάνω, εξετάζουμε χαρακτηριστικά που επηρεάζουν την επίδοση της επικοινωνίας, που εξάγονται από τα χαρακτηριστικά της εφαρμογής, την απεικόνισή της στο σύστημα και τη σύνθεση της αρχιτεκτονικής του υποκείμενου συστήματος. Κατασκευάζουμε ένα απλό πρόγραμμα συλλογής μετρήσεων αναφοράς (benchmark) για τη συλλογή ενός γενικού συνόλου εκπαίδευσης, προς χρήση με όλα τα σχήματα point-to-point επικοινωνίας στο σύνολο δοκιμής και αξιοποιούμε μεθόδους στατιστικής και μηχανικής μάθησης για την κατασκευή ενός πλήθους μοντέλων πρόβλεψης του χρόνου επικοινωνίας. Κάθε μοντέλο παρουσιάζει διαφορετικά επίπεδα ακρίβειας και διαφορετικό χρόνο πρόβλεψης (time of prediction). Εφαρμόζουμε τη μεθοδολογία μας σε δύο υπερυπολογιστές και ένα σύστημα μεγάλης κλίμακας (cluster) με υπερσύγχρονα δίκτυα διασύνδεσης: i) στο Vilje, ένα σύστημα SGI Altix με InfiniBand, ii) στο Piz Daint, ένα σύστημα Cray XC30 και iii) στο ARIS, ένα σύστημα IBM NeXtScale με InfiniBand. Στην πειραματική αποτίμηση των αποτελεσμάτων μας, επιβεβαιώνουμε δύο σημαντικές υποθέσεις που καθοδήγησαν την παρούσα εργασία: α) η συλλογή μετρήσεων αναφοράς με γενικό τρόπο, που καθιστά τη μεθοδολογία μας εφαρμόφισμη σε πολλαπλές εφαρμογές, είναι επαρκής για την κατασκευή ενός μοντέλου με υψηλή ακρίβεια πρόβλεψης σε οποιοδήποτε σύστημα και β) ένα μοντέλο πρόβλεψης της επικοινωνίας πρέπει να εξετάζει πολλαπλά χαρακτηριστικά που περιγράφουν καλά το σχήμα κίνησης δεδομένων, προκειμένου να ενσωματώνει τα πολύπλοκα φαινόμενα της επικοινωνίας σε ένα σύστημα μεγάλης κλίμακας. Συγκρίνουμε την προβλεπτική ικανότητα των μοντέλων μας με ένα αναλυτικό μοντέλο βάσης και ημι-εμπειρικά μοντέλα, ακολουθώντας τη μεθοδολογία που προτείνεται από τον Ghavari και τους συνεργάτες του [Gahvari et al., 2011, 2014]. Η εμπειρική μας προσέγγιση ξεπερνά κάθε άλλη προσέγγιση στην ακρίβεια της πρόβλεψης και παρέχει ακριβείς και ουσιαστικές προβλέψεις σε σενάρια λήψης αποφάσεων με επίγνωση της επικοινωνίας.

1.2 Συμβολή της διατριβής

Η συνεισφορά της παρούσας διατριβής συνοψίζεται στα ακόλουθα σημεία:

- Προτείνουμε μια γενική μεθοδολογία για την πρόβλεψη του χρόνου επικοινωνίας των παράλληλων εφαρμογών σε οποιοδήποτε υπολογιστικό σύστημα υπολογιστών, χρησιμοποιώντας μεθόδους στατιστικής και μηχανικής μάθησης.
- Εφαρμόζουμε τη μεθοδολογία για την πρόβλεψη του χρόνου επικοινωνίας σε τρία συστήματα μεγάλης κλίμακας με διαφορετικές αρχιτεκτονικές και τοπολογίες δικτύων διασύνδεσης.
- Μέσω της διαδικασίας μοντελοποίησης, μελετάμε και χαρακτηρίζουμε την επικοινωνία και εξάγουμε συμπεράσματα σχετικά με τα χαρακτηριστικά των δικτύων διασύνδεσης που επηρεάζουν τις επιδόσεις της επικοινωνίας.
- Μέσω εκτεταμένης πειραματικής αποτίμησης, καταδεικνύουμε την ακρίβεια πρόβλεψης και την καλή προσαρμογή των μοντέλων μας σε παράλληλες εφαρμογές του πραγματικού κόσμου. Τα μοντέλα μας βελτιώνουν σημαντικά την ακρίβεια της πρόβλεψης, μειώνοντας τα σχετικά σφάλματα πρόβλεψης κατά περισσότερο από 50%, σε σύγκριση με τις αναλυτικές και ημι-εμπειρικές προσεγγίσεις που προτείνονται στη βιβλιογραφία.
- Καταδεικνύουμε τη χρησιμότητα και τη βιωσιμότητα των προβλέψεων για το χρόνο επικοινωνίας, μέσω της αξιοποίησής τους σε σενάρια λήψης αποφάσεων σχετικά με τη βελτιστοποίηση της ποιότητας των υπηρεσιών (Quality-of-Service) και της συνολικής απόδοσης των συστημάτων μεγάλης κλίμακας.

1.3 Δομή της διατριβής

Το κείμενο της παρούσας διατριβής οργανώνεται ως εξής: Το Κεφάλαιο 2 ορίζει το πρόβλημα και παρέχει λεπτομερείς πληροφορίες σχετικά με τις διάφορες πτυχές του, δηλαδή τα συστήματα μεγάλης κλίμακας, τις παράλληλες εφαρμογές, το χρόνο επικοινωνίας και τη μοντελοποίηση αυτού. Στο ίδιο κεφάλαιο παρουσιάζεται το πρόβλημα και συζητείται η σχετική βιβλιογραφία. Το Κεφάλαιο 3 παρουσιάζει την εμπειρική μας μεθοδολογία για την κατασκευή μοντέλων πρόβλεψης της επικοινωνίας. Σε αυτό το κεφάλαιο, ορίζουμε τα χαρακτηριστικά που χρησιμοποιούμε για τη μοντελοποίηση, το πρόγραμμα συλλογής μετρήσεων αναφοράς που κατασκευάζουμε για τη συλλογή ενός συνόλου εκπαίδευσης και τις μεθόδους στατιστικής και μηχανικής μάθησης που χρησιμοποιούνται σε αυτή τη διατριβή για την κατασκευή μοντέλων πρόβλεψης του χρόνου επικοινωνίας.

Το Κεφάλαιο 4 παρουσιάζει την εφαρμογή της μεθοδολογίας στατιστικής μάθησης και μηχανικής μάθησης στο Vilje, έναν υπερυπολογιστή που χρησιμοποιεί το InfiniBand ως δίκτυο διασύνδεσης σε τοπολογία ενισχυμένου υπερκύβου, την εκτεταμένη αξιολόγηση των προβλέψεών μας σε αυτό το σύστημα και την εφαρμογή των μοντέλων μας στη λήψη αποφάσεων σχετικά με την επικοινωνία. Το Κεφάλαιο 5 παρουσιάζει την εφαρμογή της μεθοδολογίας στατιστικής μάθησης και μηχανικής μάθησης στο Piz Daint, έναν υπερυπολογιστή Cray XC30 με τοπολογία dragonfly, την αξιολόγηση όλων των μοντέλων μας για την πρόβλεψη του χρόνου επικοινωνίας, καθώς και την εφαρμογή των μοντέλων πρόβλεψης σε σενάρια λήψης αποφάσεων για τη βελτιστοποίηση της χρήσης του συστήματος. Το Κεφάλαιο 6 παρουσιάζει την εφαρμογή της μεθοδολογίας μηχανικής μάθησης στο ARIS, μία συστοιχία υπολογιστών που χρησιμοποιεί το InfiniBand σε τοπολογία fat tree, μαζί με την αξιολόγηση των μοντέλων πρόβλεψης επικοινωνίας σε εφαρμογές του πραγματικού κόσμου, και την εφαρμογή των μοντέλων μας σε εργασίες λήψης αποφάσεων για τη βελτιστοποίηση της χρήσης του συστήματος. Δεν εφαρμόζουμε τη μεθοδολογία κατασκευής μοντέλων με τεχνικές στατιστικής μάθησης στο ARIS, καθώς το σύστημα αυτό εισήλθε στη φάση παραγωγής τον Ιούνιο του 2015, σχεδόν τρία χρόνια μετά την έναρξη αυτής της διατριβής. Το Κεφάλαιο 7 ολοκληρώνει αυτή τη διατριβή και συζητά τις δυνατές μελλοντικές κατευθύνσεις.

2

Θεωρητικό υπόβαθρο και ορισμός προβλήματος

2.1 Εισαγωγή

Στην παρούσα διατριβή παρουσιάζουμε μια μεθοδολογία για την κατασκευή μοντέλων πρόβλεψης για το χρόνο επικοινωνίας των εφαρμογών HPC που εκτελούνται σε συστοιχίες μεγάλης κλίμακας και υπερυπολογιστές. Η επίδοση της επικοινωνίας των παράλληλων εφαρμογών σε συστήματα μεγάλης κλίμακας εξαρτάται από την απεικόνιση του γράφου επικοινωνίας της εφαρμογής στο υποκείμενο σύστημα. Οι υπερυπολογιστές και οι συστοιχίες μεγάλης κλίμακας κατασκευάζονται με διάφορες αρχιτεκτονικές κόμβων και δικτύων διασύνδεσης, τοπολογίες δικτύων, λειτουργικά συστήματα, λογισμικό διαχείρισης πόρων, ενδιάμεσο λογισμικό, συνιστώντας μια πολύπλοκη οντότητα. Επιπλέον, οι παράλληλες εφαρμογές περικλείουν πολλαπλές φάσεις υπολογισμών και επικοινωνίας με συγκεκριμένα χαρακτηριστικά. Η απεικόνισή τους σε ένα σύστημα μεγάλης κλίμακας δημιουργεί ένα συγκεκριμένο σχέδιο κυκλοφορίας, το οποίο μπορεί να είναι μοναδικό για κάθε εκτέλεση. Επομένως, η πρόβλεψη του χρόνου επικοινωνίας είναι ένα περίπλοκο και δύσκολο έργο, καθώς απαιτεί την κατανόηση της εφαρμογής, του συστήματος και της αλληλεπίδρασής τους και της διατύπωσης αυτής της αλληλεπίδρασης με ποσοτικοποιήσιμο τρόπο. Στις επόμενες ενότητες, παρέχουμε ορισμούς και το θεωρητικό υπόβαθρο για τα συστήματα μεγάλης κλίμακας, τις παράλληλες εφαρμογές, το χρόνο επικοινωνίας και τα μοντέλα επίδοσης της επικοινωνίας, όπως εμφανίζονται στο πλαίσιο της παρούσας εργασίας, καθώς και τις υποθέσεις που κάνουμε για την επίλυση του προβλήματος της πρόβλεψης επίδοσης της επικοινωνίας. Επιπλέον, παρουσιάζουμε τις προσεγγίσεις για τη μοντελοποίηση της επίδοσης της επικοινωνίας, όπως εμφανίζονται στη σχετική βιβλιογραφία, και περιγράφουμε τις ιδιότητες κάθε προσέγγισης, τα πλεονεκτήματα

και τα μειονεκτήματά της.

2.2 Υπολογιστικά συστήματα μεγάλης κλίμακας

Η παρούσα εργασία επικεντρώνεται σε συστήματα μεγάλης κλίμακας, δηλαδή συστοιχίες μεγάλης κλίμακας και υπερυπολογιστές. Τέτοια συστήματα προκύπτουν από εγκαταστάσεις εκατοντάδων ή χιλιάδων υπολογιστικών κόμβων, διασυνδεδεμένων με κάποιο δίκτυο διασύνδεσης υψηλής επίδοσης. Περιλαμβάνουν επίσης μια βαθιά στοίβα λογισμικού, από το λειτουργικό σύστημα έως τις βιβλιοθήκες εφαρμογών. Στις επόμενες παραγράφους, περιγράφουμε την αρχιτεκτονική υλικού και λογισμικού, καθώς και λεπτομέρειες για τα προγραμματιστικά μοντέλα που υποστηρίζονται από συστήματα μεγάλης κλίμακας.

2.2.1 Αρχιτεκτονική κόμβου

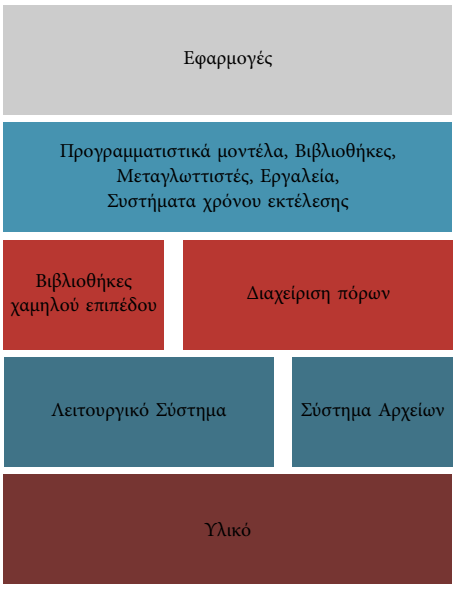
Οι υπολογιστικοί κόμβοι των υπερυπολογιστών αποτελούνται από έναν ή περισσότερους επεξεργαστές (γνωστούς ως συμμετρικούς πολυεπεξεργαστές ή SMPs), με κοινό χώρο διευθύνσεων, όπου κάθε επεξεργαστής έχει τη δική του ιεραρχία κρυφών μνημών. Εάν περισσότεροι από ένας επεξεργαστές συνυπάρχουν στον ίδιο κόμβο, διασυνδέονται με point-to-point δίκτυα επεξεργαστών. Εάν ένας κόμβος φιλοξενεί περισσότερους από έναν κόμβους μνήμης, το αποτέλεσμα είναι η αρχιτεκτονική NUMA (Non-Uniform Memory Access - μη ομοιόμορφη πρόσβαση στην μνήμη), στην οποία ο χρόνος πρόσβασης σε μια διεύθυνση μνήμης εξαρτάται από τη θέση του πυρήνα στον κόμβο. Έτσι, η αρχιτεκτονική NUMA φέρνει ασυμμετρίες στο χρόνο απόκρισης της κίνησης των δεδομένων εντός του κόμβου. Επιπρόσθετα, οι υπολογιστικοί κόμβοι συχνά φιλοξενούν συνεπεξεργαστές, όπως το Intel Xeon Phi, ή επιταχυντές, όπως κάρτες γραφικών, που συνδέονται μέσω PCI / PCIe. Ωστόσο, σε συστήματα της τρέχουσας γενιάς, αυτά τα επεξεργαστικά στοιχεία χρησιμοποιούνται ως μηχανές εκφόρτωσης (offload engines) και δεν επικοινωνούν απευθείας μεταξύ τους. Οι κόμβοι συνδέονται μέσω του PCI/PCIe με τους προσαρμογείς του δικτύου διασύνδεσης. Αξίζει να σημειωθεί ότι σε αρχιτεκτονικές NUMA, όλες οι εξωτερικές συσκευές συνδέονται σε έναν από τους επεξεργαστές μέσω του διαύλου PCI/PCIe, προκαλώντας μια πρόσθετη ασυμμετρία στους χρόνους πρόσβασης από τους πυρήνες στις συσκευές, εντός του ίδιου κόμβου. Επομένως, εκτός από τους χρόνους απόκρισης της επικοινωνίας εντός του κόμβου, π χρόνος απόκρισης της επικοινωνίας μεταξύ κόμβων μπορεί επίσης να διαφέρει ανάλογα με την απόσταση του πυρήνα από τη συσκευή επικοινωνίας [Hager et al., 2009].

2.2.2 Αρχιτεκτονική δικτύου διασύνδεσης

Τα συστήματα μεγάλης κλίμακας χρησιμοποιούν δίκτυα διασύνδεσης υψηλών επιδόσεων για τη σύνδεση των χιλιάδων υπολογιστικών κόμβων. Σύμφωνα με την πιο πρόσφατη λίστα Top500 [top], οι πιο δημοφιλείς τεχνολογίες δικτύου διασύνδεσης είναι αυτές του Ethernet [Metcalfe and Boggs, 1976], του InfiniBand [Association et al., 2001], του Cray Aries [Alverson et al., 2012] και του νεότερου Intel OmniPath [Birrittella et al., 2015]. Η τεχνολογία κάθε δικτύου καθορίζει το χρόνο απόκρισης και το εύρος ζώνης, καθώς και άλλες κρίσιμες παραμέτρους, όπως το μέγεθος και τη δυνατότητα αποθήκευσης των διακοπτών (switches) και τους μηχανισμούς δρομολόγησης. Οι τεχνολογίες διαφέρουν στην αρχιτεκτονική των προσαρμογέων δικτύου και των διακοπτών, και στη διεπαφή του δικτύου. Τα υπάρχοντα δίκτυα διασύνδεσης αξιοποιούν διεπαφές που μπορούν να εκφορτώσουν τις λειτουργίες επικοινωνίας από τον επεξεργαστή στο δίκτυο, με στόχο την αποσύνδεση της επικοινωνίας από τον επεξεργαστή, επιτρέποντας την επικάλυψη υπολογισμών/επικοινωνίας [Di Girolamo et al., 2015]. Επιπρόσθετα, οι περισσότεροι κατασκευαστές εργάζονται για την πλήρη υποστήριξη μονόπλευρων (one-sided) και συλλογικών (collective) λειτουργιών επικοινωνίας στο υλικό, ώστε να ελαχιστοποιούν το κόστος αυτών των λειτουργιών, σε χρόνο και ισχύ, και να επιτρέπουν την επικάλυψη [Schneider et al., 2013].

Κάθε τεχνολογία δικτύου είναι συνήθως συμβατή με συγκεκριμένες τοπολογίες δικτύου. Η τοπολογία του δικτύου καθορίζει τον αριθμό των καρτών διεπαφής δικτύου, των διακοπτών, των δρομολογητών και των συνδέσεων που απαρτίζουν το σύστημα, ανάλογα με την κλίμακα του συστήματος, καθώς και τη διάμετρο, το πλήθος των δυνατών μονοπατιών και τον αριθμό των κόμβων ανά διακόπτη / δρομολογητή. Παράλληλα με την τεχνολογία και το πρωτόκολλο δρομολόγησης, η τοπολογία καθορίζει επίσης το εύρος τομής και το συνολικό εύρος ζώνης του δικτύου και τον αριθμό των βημάτων (hops) για διαφορετικές διαδρομές. Μια τοπολογία μπορεί να είναι άμεση ή έμμεση. Στις άμεσες τοπολογίες, όπως είναι οι k-διάστατοι n-κύβοι και η τοπολογία dragonfly, κάθε διακόπτης είναι τερματικός και συνδέεται με τουλάχιστον έναν υπολογιστικό κόμβο, ενώ σε έμμεσες τοπολογίες, όπως είναι οι δενδρικές τοπολογίες, υπάρχουν και μη τερματικοί διακόπτες.

Οι τοπολογίες ανά τεχνολογία στους ταχύτερους υπερυπολογιστές στον κόσμο μοιράζονται ως εξής: το Ethernet χρησιμοποιεί ιεραρχικές δενδρικές τοπολογίες, το InfiniBand χρησιμοποιεί κυρίως τοπολογίες fat trees, καθώς και την τοπολογία υπερκύβου, το Cray Aries χρησιμοποιεί την τοπολογία Dragonfly, το Cray Gemini χρησιμοποιεί την τοπολογία του 3Δ-τόρου, το IBM BlueGene/Q χρησιμοποιεί τοπολογία 5Δ-τόρου, το Intel OmniPath χρησιμοποιεί τοπολογίες fat trees (αλλά αναμένεται να χρησιμοποιήσει την τοπολογία



Σχήμα 2.1: Ένα παράδειγμα της στοίβας λογισμικού ενός υπερυπολογιστή.

dragonfly στο μέλλον), και το Fujitsu Tofu χρησιμοποιεί τοπολογία 6Δ-τόρου. Μεταξύ των τοπολογιών αυτών, οι τοπολογίες fat trees και dragonfly προσφέρουν χαμηλή διάμετρο και επομένως χαμηλό χρόνο απόκρισης, εξαιτίας της αρχιτεκτονικής των δρομολογητών τους, που αποτελούνται από πολλές θύρες, καθώς και υψηλό εύρος τομής. Ωστόσο, τα fat trees εμφανίζουν μεγάλο κόστος συνδέσεων και υψηλή κατανάλωση ενέργειας σε μεγάλη κλίμακα. Από την άλλη πλευρά, οι τοπολογίες τόρων, οι οποίες επίσης χρησιμοποιούνται ευρέως, αξιοποιούν δρομολογητές με μικρό αριθμό θυρών και οδηγούν σε υψηλότερες διαμέτρους, ωστόσο προσφέρουν υψηλό εύρος τομής σε μεγάλες διαστάσεις. Τα συστήματα της επόμενης γενιάς θα περιλαμβάνουν τοπολογίες fat trees, Dragonfly και τόρων υψηλής διάστασης, ανάλογα με τους περιορισμούς ισχύος και προϋπολογισμού τους και τις απαιτήσεις εφαρμογών για χαμηλό χρόνο απόκρισης μέσω μικρής διαμέτρου, για υψηλή απόδοση μέσω υψηλού εύρους τομής και για αποφυγή παρεμβολών με άλλες εφαρμογές μέσω της διαμερισιμότητας (partitionability) της τοπολογίας [Jain et al., 2016, 2017].

2.2.3 Στοίβα λογισμικού

Η αποδοτική λειτουργία των υπερυπολογιστών βασίζεται σε μια πλούσια στοίβα λογισμικού, από το λειτουργικό σύστημα μέχρι την εφαρμογή. Παρουσιάζουμε μια εννοιολογική έκδοση μιας τέτοιας στοίβας λογισμικού στο Σχήμα 2.1. Το λειτουργικό σύστημα είναι κοινό για τους υπολογιστικούς κόμ-

βους και συνήθως χρησιμοποιεί κάποια έκδοση πυρήνα ανοιχτού κώδικα. Επιπλέον, οι υπερυπολογιστές αξιοποιούν ένα παράλληλο σύστημα αρχείων για όλους τους κόμβους ή τους κόμβους μίας διαμέρισης του συστήματος, που χρησιμοποιείται αποκλειστικά για είσοδο/έξοδο. Για τη λειτουργία του δικτύου διασύνδεσης χρησιμοποιείται μια στοίβα λογισμικού διαχείρισης δικτύου. Σε υψηλότερο επίπεδο, η στοίβα λογισμικού περιλαμβάνει εργαλεία ανάπτυξης εφαρμογών, μεταγλωττιστές, βιβλιοθήκες επικοινωνίας και άλλες βιβλιοθήκες παράλληλου προγραμματισμού, μαθηματικές βιβλιοθήκες κώδικα και βιβλιοθήκες εφαρμογών. Όλες οι βιβλιοθήκες μπορούν να αναπτύσσουν τα δικά τους συστήματα χρόνου εκτέλεσης.

Συγκεκριμένα στρώματα λογισμικού διαχειρίζονται τους πόρους του συστήματος. Σε υψηλότερο επίπεδο, κάποιο λογισμικό διαχείρισης πόρων, π.χ. το SLURM [Yoo et al., 2003], το Torque [Staples, 2006] και άλλα, αναλαμβάνει την κατανομή πόρων και τη δρομολόγηση των εργασιών. Σε χαμηλότερο επίπεδο, άλλα στρώματα λογισμικού, όπως το λειτουργικό σύστημα, είναι υπεύθυνα για την τροφοδοσία των υψηλότερων στρωμάτων με πληροφορίες σχετικά με την κατάσταση και τη λειτουργία του συστήματος. Στους υπερυπολογιστές, οι χρήστες υποβάλλουν μια εργασία στο σύστημα, ζητώντας ένα σύνολο πόρων για την εκτέλεση της εφαρμογής τους, π.χ. έναν αριθμό κόμβων με ορισμένα χαρακτηριστικά, έναν συγκεκριμένο αριθμό πυρήνων, κάποια ποσότητα μνήμης, κόμβους με επιταχυντές και άλλα. Το λογισμικό διαχείρισης πόρων δεσμεύει ένα σύνολο πόρων που ικανοποιεί το αίτημα του χρήστη, απομονώνει τους πόρους, δρομολογεί τις εργασίες και παρακολουθεί το σύστημα και την κατάσταση των εργασιών και των πόρων. Αν και το λογισμικό διαχείρισης πόρων εξυπηρετεί κυρίως τα αιτήματα των χρηστών, επιβάλλει επίσης πολιτικές χρονοδρομολόγησης για τη βελτιστοποίηση της χρήσης του συστήματος, στοχεύοντας στην εξυπηρέτηση είτε του χρήστη (π.χ. μείωση χρόνου αναμονής), είτε της εφαρμογής (π.χ. μεγιστοποίηση της απόδοσης), είτε του συστήματος (π.χ. μεγιστοποίηση της χρήσης). Σε κάθε περίπτωση, για κάθε υποβολή εργασίας, το λογισμικό διαχείρισης πόρων επιλέγει ένα συγκεκριμένο σύνολο πόρων (κατανομή κόμβων). Συνεπώς, για μία μοναδική εκτέλεση μιας εφαρμογής, το επιλεγμένο σύνολο πόρων, η τοπολογία του συστήματος και η εφαρμογή παράγουν μια μοναδική κατανομή κόμβων και μία συγκεκριμένη απεικόνιση των διεργασιών στο σύστημα.

Οι αποφάσεις δρομολόγησης, τοποθέτησης και απεικόνισης, που λαμβάνονται από το λογισμικό διαχείρισης πόρων, είναι κρίσιμες για την απόδοση της επικοινωνίας. Όπως αναλύουμε λεπτομερώς στις επόμενες παραγράφους, η απεικόνιση του γράφου επικοινωνίας της εφαρμογής στη δεδομένη κατανομή κόμβων και ο προκύπτων γράφος του δικτύου παράγουν ένα συγκεκριμένο σχήμα κίνησης δεδομένων, που καθορίζει την επίδοση της επικοινωνίας. Η ικανότητα επιρροής των αποφάσεων του λογισμικού διαχειρισης πόρων εί-

ναι σημαντική και θα είναι κρίσιμη για τα συστήματα μεγάλης κλίμακας, για την ελαχιστοποίηση της κίνησης των δεδομένων και τη βελτιστοποίηση της επίδοσης της επικοινωνίας [Dongarra et al., 2011].

2.3 Παράλληλες εφαρμογές μεγάλης κλίμακας

Οι παράλληλες εφαρμογές που εκτελούνται σε συστήματα μεγάλης κλίμακας είναι κυρίως προσομοιώσεις πραγματικών φαινομένων από πολλούς επιστημονικούς τομείς. Στην πλειονότητά τους, έχουν κατασκευαστεί για να εκτελούνται σε συστήματα μεγάλης κλίμακας, ακολουθώντας συγκεκριμένα προγραμματιστικά μοντέλα, που τους επιτρέπουν να κλιμακώνουν αποτελεσματικά στα συστήματα υψηλής επίδοσης. Οι παράλληλες εφαρμογές μεγάλης κλίμακας περιλαμβάνουν πολλαπλούς υπολογιστικούς πυρήνες και περνούν από διάφορες φάσεις υπολογισμών, επικοινωνίας και εισόδου/εξόδου. Περιγράφουμε τα κυρίαρχα προγραμματιστικά μοντέλα για παράλληλες εφαρμογές μεγάλης κλίμακας και συζητούμε τα σχήματα επικοινωνίας τους στις ακόλουθες παραγράφους.

2.3.1 Προγραμματιστικά μοντέλα

Το de facto μοντέλο προγραμματισμού για συστήματα μεγάλης κλίμακας κατανεμημένης μνήμης είναι το μοντέλο περάσματος μηνυμάτων, που εκφράζεται καλύτερα μέσω του Interface Passing Interface [Snir, 1998], ευρέως γνωστού ως MPI. Το MPI είναι ένα πλούσιο προγραμματιστικό μοντέλο με ποικίλες υλοποιήσεις, ανοιχτού κώδικα ή εταιρικές, και υποστηρίζεται ενεργά και προωθείται από το φόρουμ του MPI από το 1992, σε μια προσπάθεια να παραχθεί ένα φορητό λογισμικό που γεφυρώνει τα κενά μεταξύ των απαιτήσεων των εφαρμογών για υψηλή επίδοση και προγραμματισιμότητα και της αρχιτεκτονικής των συστημάτων. Το MPI υλοποιεί τον παραλληλισμό SPMD (single program, multiple data - ένα πρόγραμμα, πολλαπλά δεδομένα) και η στοιχειώδης οντότητα παραλληλισμού είναι η διεργασία. Κάθε διεργασία έχει ιδιωτική εικονική μνήμη, όπου αποθηκεύει τα ιδιωτικά δεδομένα της και επικοινωνεί με άλλες διεργασίες με ανταλλαγή μηνυμάτων. Το πρότυπο του MPI προσφέρει πολλαπλές λειτουργίες επικοινωνίας, οι οποίες μπορεί να είναι point-to-point (μεταξύ δύο διεργασιών το ανώτερο), συλλογικές (μεταξύ πολλαπλών διεργασιών) ή μονόπλευρες, όπου μια διεργασία μπορεί να διαβάζει ή να γράφει δεδομένα από/προς την απομακρυσμένη μνήμη μιας άλλης διεργασίας. Οι διάφορες υλοποιήσεις του MPI χρησιμοποιούν τα προγράμματα οδήγησης συσκευών δικτύου και τις βιβλιοθήκες τους ή/και ενδιάμεσο λογισμικό επικοινωνίας, για την υλοποίηση αποτελεσματικών και κλιμακώσιμων λειτουργιών επικοινωνίας για κάθε τύπο κόμβου και αρχιτεκτονική δικτύου. Το MPI δεν

είναι ένα λεπτό στρώμα επικοινωνίας: εκτός από την υποστήριξη πολλών συσκευών επικοινωνίας για μέγιστη φορητότητα, όλες οι υλοποιήσεις του MPI λαμβάνουν αποφάσεις, ανάλογα με την απεικόνιση των διεργασιών στο σύστημα, τον τύπο επικοινωνίας, το μέγεθος του μηνύματος και άλλα, για να παρέχουν βέλτιστη επίδοση στις εφαρμογές. Τέτοιες αποφάσεις περιλαμβάνουν την επιλογή της βέλτιστης διεπαφής επικοινωνίας, π.χ. τη διεπαφή δικτύου ή τη διεπαφή του συστήματος μνήμης, το βέλτιστο στρώμα μεταφοράς και το βέλτιστο πρωτόκολλο μεταφοράς δεδομένων, π.χ. short (άμεση μεταφορά), ραντεβού (μεταφορά μετά από "χειραψία"), buffered (μεταφορά μετά την αντιγραφή δεδομένων σε εσωτερικούς αποθηκευτικούς χώρους). Επιπλέον, υλοποιούν τη συλλογική επικοινωνία με αλγόριθμους χαμηλής πολυπλοκότητας και μπορούν να επιλέγουν τον βέλτιστο αλγόριθμο κατά την εκτέλεση, για να βελτιστοποιήσουν την επίδοση. Έτσι, η υλοποίηση του MPI και ο σχεδιασμός της επηρεάζουν ποικιλοτρόπως την επίδοση της επικοινωνίας.

Το μοντέλο περάσματος μηνυμάτων προσφέρει υψηλή επίδοση, καθώς η σημασιολογία της επικοινωνίας είναι κοντά στις εγγενείς λειτουργίες επικοινωνίας της αρχιτεκτονικής, ενώ ταυτόχρονα προσφέρει αφαιρετικές λειτουργίες σε υψηλότερο επίπεδο, για να διευκολύνει τον προγραμματιστή στην έκφραση του παραλληλισμού των εφαρμογών του. Τα τελευταία χρόνια έχουν εμφανιστεί νέες γλώσσες προγραμματισμού, που προσφέρουν αφαιρετικές λειτουργίες επικοινωνίας ή παραλληλισμού, όπως οι γλώσσες PGAS (Partitioned Global Address Space - διαμερισμένου παγκόσμιου χώρου διευθύνσεων) ή οι εργασιοκεντρικές (task-centric) γλώσσες προγραμματισμού. Οι γλώσσες PGAS, όπως η UPC [Consortium et al., 2005] και η OpenSHMEM [Chapman et al., 2010], προσφέρουν μια κοινή προβολή του χώρου διευθύνσεων της κατανεμημένης μνήμης και ο χρήστης μπορεί να υλοποιήσει την επικοινωνία με απλή σημασιολογία αναγνώσεων/εγγραφών, γνωρίζοντας την ακριβή θέση των δεδομένων. Οι εργασιοκεντρικές γλώσσες αντιπροσωπεύονται καλύτερα από τη γλώσσα Charm++ [Kale and Krishnan, 1993], μια αντικειμενοστραφή γλώσσα, όπου η στοιχειώδης οντότητα του παραλληλισμού είναι η εργασία (task). Αν και αυτές οι γλώσσες κρύβουν το πέρασμα μηνυμάτων από το χρήστη, οι υλοποιήσεις τους χρησιμοποιούν το μοντέλο περάσματος μηνυμάτων (είτε το MPI είτε ενδιάμεσο λογισμικό επικοινωνίας). Το MPI και οι άλλες γλώσσες ή βιβλιοθήκες για τον προγραμματισμό σε κατανεμημένο χώρο διευθύνσεων συχνά συνδυάζονται με μοντέλα προγραμματισμού κοινού χώρου διευθύνσεων, όπως το OpenMP [Dagum and Menon, 1998], για την εκμετάλλευση του παραλληλισμού εντός του κόμβου, ή με μοντέλα προγραμματισμού και βιβλιοθήκες για ετερογενές υλικό και επιταχυντές.

2.3.2 Σχήματα επικοινωνίας των παράλληλων εφαρμογών

Οι παράλληλες εφαρμογές έχουν κάποια χαρακτηριστικά επικοινωνίας που συνιστούν το προφίλ επικοινωνίας τους. Μια εφαρμογή μπορεί να εμφανίζει πολλαπλές φάσεις επικοινωνίας, οι οποίες μπορεί να είναι point-to-point, μονόπλευρη ή συλλογική. Μια φάση επικοινωνίας χαρακτηρίζεται από ένα συγκεκριμένο σχήμα επικοινωνίας, που καθορίζει ποιες διεργασίες θα επικοινωνήσουν, και από το πλήθος και τα μεγέθη των μηνυμάτων που ανταλλάσσει κάθε διεργασία. Όλα τα παραπάνω συνιστούν το *γράφο επικοινωνίας* μιας φάσης της εφαρμογής. Καθώς οι περισσότερες εφαρμογές μεγάλης κλίμακας εκτελούν επαναληπτικά κάποιους υπολογιστικούς πυρήνες, το σχήμα επικοινωνίας για μια φάση μπορεί να είναι στατικό, δηλαδή να παραμένει ίδιο σε κάθε επανάληψη ως προς το πλήθος των μηνυμάτων, τα μεγέθη των μηνυμάτων και τις διεργασίες-προορισμούς, ή δυναμικό. Επιπλέον, μια φάση επικοινωνίας μπορεί να είναι διακριτή και συγχρονισμένη, όπου όλες οι διεργασίες, μετά από μια φάση υπολογισμών, ξεκινούν περίπου την ίδια χρονική στιγμή ανταλλαγές μηνυμάτων, ή μη διακριτή, όπου ανταλλαγές μηνυμάτων και υπολογισμοί εναλλάσσονται, ή μη συγχρονισμένη, όπου, λόγω ανισοκατανομής του φόρτου υπολογισμών, οι διεργασίες ξεκινούν ανταλλαγές μηνυμάτων για μια φάση επικοινωνίας έχοντας μεγάλες χρονικές αποκλίσεις.

Το σχήμα επικοινωνίας μιας φάσης της εφαρμογής εξαρτάται από το είδος του προβλήματος που επιλύει, τη μέθοδο επίλυσης και το διαχωρισμό του προβλήματος στους επεξεργαστές (problem decomposition). Τα μεγέθη των μηνυμάτων που προκύπτουν και ο συνολικός όγκος δεδομένων που ανταλλάσσεται εξαρτάται από το μέγεθος του προβλήματος που επιλύεται. Η συλλογική επικοινωνία προδιαγράφει το σχήμα επικοινωνίας, αφού κάθε συνάρτηση καθορίζει ποιες διεργασίες ανταλλάσσουν δεδομένα, σε αντίθεση με την point-to-point ή μονόπλευρη επικοινωνία. Ωστόσο, συγκεκριμένα προβλήματα εμφανίζουν συγκεκριμένα σχήματα point-to-point επικοινωνίας. Ακολουθώντας την κατηγοριοποίηση προβλημάτων για συστήματα υψηλής επίδοσης σε 7 κατηγορίες, όπως δόθηκε από επιστήμονες του Berkeley το 2006 [Asanovic et al., 2006], περιγράφουμε τα σχήματα επικοινωνίας που εμφανίζονται στα προβλήματα αυτά:

- *Γραμμική Άλγεβρα*: Η αριθμητική γραμμική άλγεβρα περιλαμβάνει πράξεις σε πυκνούς πίνακες και διανύσματα. Σε αυτά τα προβλήματα, η point-to-point επικοινωνία που εμφανίζεται αφορά ανταλλαγές γραμμών ή στηλών των πινάκων και είναι περιορισμένη. Συχνότερα, χρησιμοποιούνται συμμετρικές συναρτήσεις collective επικοινωνίας για το διαμοιρασμό των πινάκων/διανυσμάτων, την ανταλλαγή διανυσμάτων και τη συλλογή αποτελεσμάτων. Επιπλέον, τα σχήματα επικοινωνίας σε αυτά τα προβλήματα είναι στατικά.

- *Γραμμική άλγεβρα σε αραιούς πίνακες*: Τα προβλήματα σε αυτή την κατηγορία περιλαμβάνουν πράξεις σε αραιούς πίνακες και διανύσματα. Η επικοινωνία αφορά επίσης ανταλλαγές γραμμών/στηλών των πινάκων και στοιχείων διανυσμάτων, ωστόσο εξαιτίας των αραιών πινάκων, η point-to-point επικοινωνία είναι πιο διαδεδομένη. Στα προβλήματα αραιών πινάκων, εμφανίζονται στατικά σχήματα point-to-point επικοινωνίας, που εξαρτώνται από τη μορφή του αραιού πίνακα. Κάθε διεργασία ανταλλάσσει διαφορετικό πλήθος μηνυμάτων με διαφορετικές διεργασίες, με μεγέθη μηνυμάτων που εξαρτώνται από τα μη μηδενικά στοιχεία των αραιών πινάκων. Αξίζει να σημειωθεί πως, ενώ το σχήμα επικοινωνίας είναι στατικό, ο γράφος της επικοινωνίας μπορεί να είναι δυναμικός, δηλαδή τα μεγέθη των μηνυμάτων μπορούν να αλλάζουν ανά επανάληψη, ανάλογα με το πρόβλημα. Για παράδειγμα, ο πολλαπλασιασμός αραιού πίνακα-διανύσματος εμφανίζει στατικό σχήμα και γράφο επικοινωνίας , αλλά ο πολλαπλασιασμός αραιών πινάκων εμφανίζει στατικό σχήμα επικοινωνίας και δυναμικό γράφο επικοινωνίας. Τα προβλήματα αυτής της κατηγορίας εμφανίζουν τα πιο ακανόνιστα (irregular), αλλά στατικά, σχήματα point-to-point επικοινωνίας.

- *Δομημένα πλέγματα*: Τα προβλήματα σε αυτή την κατηγορία αφορούν επίλυση μερικών διαφορικών εξισώσεων σε δομημένα πλέγματα. Το δομημένο πλέγμα εκφράζεται ως Καρτεσιανό πλέγμα n-διαστάσεων και το πρόβλημα διαμοιράζεται αντίστοιχα στο πλέγμα. Η επικοινωνία σε αυτά τα προβλήματα είναι point-to-point και στατική και ταυτίζεται με τη γεωμετρία του προβλήματος. Οι διεργασίες επικοινωνούν με συγκεκριμένες από τις γειτονικές τους διεργασίες και ανταλλάσσουν όψεις, ακμές ή/και κορυφές του πλέγματος, ανάλογα με τη μέθοδο επίλυσης και τη διάσταση του προβλήματος. Οι επιλυτές ΜΔΕ σε αυτή την κατηγορία εμφανίζουν σχήματα υπολογισμών τύπου stencil (π.χ. stencil 6 σημείων σε τρισδιάστατο πλέγμα), που καθορίζουν ακριβώς το σχήμα επικοινωνίας.

- *Αδόμητα πλέγματα*: Σε αυτή την κατηγορία ανήκουν προβλήματα, που αντίστοιχα με αυτά της προηγούμενης κατηγορίας, επιλύουν μερικές διαφορικές εξισώσεις σε αδόμητα πλέγματα, που αφορούν μια επιφάνεια ή έναν όγκο με ακανόνιστη γεωμετρία. Ο διαχωρισμός τέτοιων προβλημάτων σε διεργασίες γίνεται με την αναπαράσταση των σημείων του πλέγματος ως γράφου και τον μετέπειτα διαχωρισμό του πλέγματος, είτε στατικά είτε με την τεχνική AMR (adaptive mesh refinement, προσαρμοστική εκλέπτυνση πλέγματος). Η επικοινωνία είναι point-to-point και στατική. Κάθε διεργασία ανταλλάσσει μηνύματα συγκεκριμένου μεγέθους με γειτονικές διεργασίες (σχήμα nearest-neighbor) που περιλαμ-

βάνουν σημεία του πλέγματος με κοινή γεωμετρική γειτονιά.

- *Ν-Σωματίδια*: Τα προβλήματα Ν-σωματιδίων (N-body) αφορούν δυναμικά συστήματα πολλών σωματιδίων υπό την επίδραση φυσικών δυνάμεων. Ανάλογα με τη μέθοδο επίλυσης, τα προβλήματα αυτού του τύπου εμφανίζουν διαφορετικό σχήμα επικοινωνίας. Ορισμένες μέθοδοι διαχωρίζουν στατικά το χώρο με καρτεσιανή γεωμετρία και κάθε διεργασία αναλαμβάνει ένα καρτεσιανό χώρο με κάποιο πλήθος σωματιδίων. Σε αυτή την περίπτωση, οι διεργασίες ανταλλάσσουν σωματίδια με τις γειτονικές τους διεργασίες, με point-to-point τρόπο. Ωστόσο, ο γράφος της επικοινωνίας είναι δυναμικός: καθώς τα σωματίδια κινούνται στο χώρο, υπό την επίδραση βαρυτικών δυνάμεων ή/και αλληλεπιδράσεων, οι διεργασίες ανταλλάσσουν διαφορετικά πλήθη σωματιδίων, επομένως και διαφορετικά μεγέθη μηνυμάτων. Άλλες μέθοδοι επίλυσης χρησιμοποιούν τη δενδρική μέθοδο αναπαράστασης octree, για το διαχωρισμό του τρισδιάστατου χώρου. Σε αυτές τις μεθόδους, η επικοινωνία είναι point-to-point σε σχήμα υπερκύβου και είναι στατική.
- *Φασματικές Μέθοδοι*: Οι φασματικές μέθοδοι, όπως ο μετασχηματισμός Fourier, μετασχηματίζουν δεδομένα από χρονικό ή χωρικό πεδίο σε φασματικό και αντίστροφα. Η επικοινωνία στις μεθόδους αυτές είναι συλλογική και τύπου all-to-all, όπου όλες οι διεργασίες ή υποσύνολα διεργασιών ανταλλάσσουν δεδομένα με όλες τις άλλες διεργασίες (ή στο ίδιο υποσύνολο).

Άλλες ευρείες κατηγορίες εφαρμογών που εκτελούνται σε κατανεμημένα συστήματα υψηλής επίδοσης είναι οι εφαρμογές MapReduce και οι αλγόριθμοι γράφων. Στις εφαρμογές MapReduce, η επικοινωνία είναι αμελητέα. Σε αλγόριθμους γραφημάτων, η επικοινωνία είναι point-to-point ή μονόπλευρη, εξαρτάται σε μεγάλο βαθμό από το γράφο και τον αλγόριθμο και, τόσο το σχήμα επικοινωνίας όσο και ο γράφος επικοινωνίας είναι ακανόνιστα.

2.3.3 Επίδοση επικοινωνίας

Η επίδοση της επικοινωνίας προκύπτει ως το αποτέλεσμα της αλληλεπίδρασης μεταξύ της εφαρμογής, του δικτύου διασύνδεσης, του λογισμικού συστήματος και τελικά της απεικόνισης της εφαρμογής στο σύστημα κατά την εκτέλεσή της. Μια εφαρμογή εμφανίζει πολλαπλά *σχήματα επικοινωνίας*, ενώ το μέγεθος του προβλήματος και ο διαχωρισμός του προβλήματος (problem decomposition) καθορίζουν τα μεγέθη των μηνυμάτων και ορίζουν τους γράφους επικοινωνίας και τον συνολικό όγκο επικοινωνίας για κάθε σχήμα. Κάθε φορά που ένας χρήστης υποβάλλει μια εργασία στο σύστημα, για την εκτέλεση μιας εφαρμογής, ο χρονοδρομολογητής του συστήματος αναλαμβάνει

τη *δέσμευση* (allocation) των πόρων που ζητά ο χρήστης και την *τοποθέτηση* (placement) των διεργασιών. Αυτή η απεικόνιση της εφαρμογής για τη συγκεκριμένη εκτέλεση αλληλεπιδρά με το σύστημα και παράγει ένα μοναδικό *σχήμα κίνησης δεδομένων* (traffic pattern) για κάθε σχήμα επικοινωνίας στην εφαρμογή.

Το *σχήμα κίνησης δεδομένων* καθορίζει την ποσότητα των δεδομένων που ρέουν μέσω των διαφορετικών σημείων του συστήματος και ασκεί πίεση σε αυτά. Αρχικά, μέρος των δεδομένων ανταλλάσσεται μεταξύ διεργασιών που απεικονίζονται στον ίδιο κόμβο. Η επικοινωνία εντός του κόμβου (intranode communication) γίνεται μέσω της κοινής μνήμης και ασκεί πίεση στο σύστημα της μνήμης. Αν ο κόμβος είναι αρχιτεκτονικής NUMA, τότε κάποια δεδομένα ανταλλάσσονται μεταξύ των μνημών των επεξεργαστών του κόμβου μέσω των διαύλων μνήμης και εμφανίζουν καθυστερήσεις. Ωστόσο, η επικοινωνία εντός του κόμβου είναι γενικά ταχύτερη από την επικοινωνία μεταξύ κόμβων (internode communication). Το άλλο μέρος των δεδομένων εγχέεται με τη μορφή πακέτων από τους κόμβους στο δίκτυο διασύνδεσης. Σε αυτό το σημείο, ενδέχεται να εμφανιστούν φαινόμενα ανταγωνισμού, λόγω περιορισμένου εύρους ζώνης έγχυσης (injection bandwidth) ή λόγω της περιορισμένης χωρητικότητας των ουρών ή άλλων αποθηκευτικών χώρων στις κάρτες δικτύου [Bhatele et al., 2015]. Ακολούθως, τα πακέτα εγχέονται στο δίκτυο και ταξιδεύουν μέσω ενδιάμεσων διακοπτών και συνδέσεων προς τον κόμβο-προορισμό. Το μονοπάτι που ακολουθεί κάθε πακέτο εξαρτάται από την τοπολογία και το πρωτόκολλο δρομολόγησης και μπορεί να είναι ντετερμινιστικό ή προσαρμοστικό, όπου εξαρτάται από την τρέχουσα συμφόρηση στις συνδέσεις του δικτύου [Hoefler and Snir, 2011]. Στην περίπτωση της ντετερμινιστικής δρομολόγησης, τα πακέτα συνήθως ακολουθούν το συντομότερο μονοπάτι, ελαχιστοποιώντας έτσι τις καθυστερήσεις λόγω πολλαπλών βημάτων (hops) στο δίκτυο, ωστόσο είναι πιθανό να προκαλέσουν συμφόρηση στους ενδιάμεσους διακόπτες, ακόμα και αν δεν υπάρχει συμφόρηση στις συνδέσεις [Bhatelé and Kalé, 2009]. Στην περίπτωση της προσαρμοστικής δρομολόγησης, κάποια πακέτα ακολουθούν μεγαλύτερα μονοπάτια για να αποφύγουν τη συμφόρηση, εμφανίζοντας ωστόσο κάποιες καθυστερήσεις.

Είναι σαφές ότι το σχήμα κίνησης δεδομένων είναι καθοριστικός παράγοντας της επίδοσης της επικοινωνίας. Το σχήμα κίνησης δεδομένων είναι μοναδικό για μία συγκεκριμένη εκτέλεση μιας εφαρμογής με συγκεκριμένη απεικόνιση των διεργασιών στο σύστημα. Εναλλακτικές απεικονίσεις μπορούν να παράγουν πολύ διαφορετικά σχήματα κίνησης δεδομένων, με διαφορετική επίδοση επικοινωνίας, καθώς η κατανομή της κίνησης δεδομένων μεταβάλλεται και ασκεί πίεση σε διαφορετικά σημεία του δικτύου [Jain et al., 2013]. Ωστόσο, το σχήμα κίνησης δεδομένων δεν καθορίζει αποκλειστικά την επίδοσης της επικοινωνίας. Συχνά, στα συστήματα του πραγματικού κόσμου, εμφανίζονται

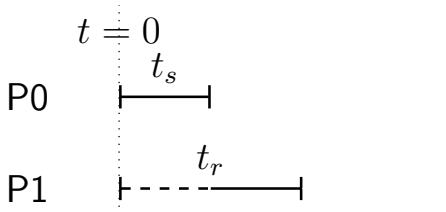
διακυμάνσεις στο χρόνο επικοινωνίας, εξαιτίας μη ντετερμινιστικών παραγόντων κατά το χρόνο εκτέλεσης. Για παράδειγμα, κατανομές κόμβων που χρησιμοποιούν περισσότερα δικτυακά στοιχεία είναι πιο επιρρεπείς σε παρεμβολές από συνεκτελούμενες εφαρμογές, γνωστές και ως ανταγωνισμός μεταξύ των εφαρμογών, και είναι πιθανό να αντιμετωπίσουν μείωση της επίδοσής τους, εξαιτίας του ανταγωνισμού σε αυτά τα στοιχεία. Ανάλογα με το σύστημα, η υποβάθμιση της επίδοσης ενδέχεται να παρουσιαστεί λόγω του διαμοιρασμού των διακοπτών ή της μη ντετερμινιστικής δρομολόγησης. Εκτός από τον ανταγωνισμό μεταξύ των εφαρμογών, το σύστημα είναι πιθανό να εμφανίσει θόρυβο, που προέρχεται από τους κόμβους λόγω υπηρεσιών του λειτουργικού συστήματος ή λόγω χαμηλού λόγου byte-to-flop [Ferreira et al., 2013]. Ο θόρυβος του συστήματος προκαλεί καθυστέρηση, η οποία μπορεί να διαδοθεί σε όλη τη φάση επικοινωνίας ή να επικαλυφθεί με τις κανονικές καθυστερήσεις που προκύπτουν από το συγχρονισμό της επικοινωνίας [Hoefler et al., 2010b]. Επομένως, η επίδοση της επικοινωνίας μπορεί να μεταβληθεί οσοδήποτε, εξαιτίας του θορύβου.

2.4 Χρόνος επικοινωνίας

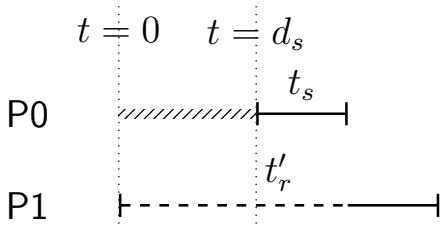
Εκτός από τους πολλαπλούς παράγοντες που επηρεάζουν την επίδοση της επικοινωνίας, ο ορισμός του χρόνου επικοινωνίας, στο πλαίσιο μιας εφαρμογής, δεν είναι εύκολος. Υπάρχουν σημαντικές διαφορές μεταξύ του χρόνου επικοινωνίας ανά λειτουργία, του χρόνου επικοινωνίας ανά διεργασία ("τοπικός" χρόνος επικοινωνίας) και του χρόνου επικοινωνίας για όλες τις διεργασίες ("συνολικός" χρόνος επικοινωνίας). Επίσης, ο καθένας από τους τρεις μπορεί να διαφέρει σε διαφορετικά σενάρια. Στις επόμενες παραγράφους, συζητάμε τι είναι δυνατό να οριστεί ως χρόνος επικοινωνίας μιας εφαρμογής.

2.4.1 Συμβάντα επικοινωνίας και χρόνος επικοινωνίας

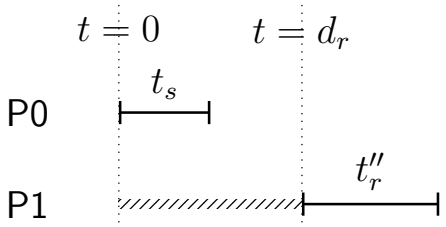
Η επικοινωνία εντός μιας εφαρμογής είναι μια σειρά μεταδόσεων μηνυμάτων μεταξύ δύο ή περισσοτέρων διεργασιών, συμπεριλαμβανομένων των συμβάντων (events) επικοινωνίας που ενεργοποιούν ή ολοκληρώνουν τις μεταδόσεις. Για την point-to-point επικοινωνία, τα συμβάντα επικοινωνίας ορίζονται σαφώς από τον προγραμματιστή της εφαρμογής, ενώ για τη συλλογική επικοινωνία, η βιβλιοθήκη που διαβιβάζει το μήνυμα, π.χ. το MPI, μεταφράζει με διαφάνεια τη λειτουργία στα απαιτούμενα συμβάντα επικοινωνίας. Θεωρητικά, μπορεί κανείς να μετρήσει i) το χρόνο επικοινωνίας για ένα μόνο συμβάν επικοινωνίας, δηλαδή τον χρόνο επικοινωνίας ανά λειτουργία, ή ii) το χρόνο για μια σειρά συμβάντων επικοινωνίας που συμβαίνουν σε μία μόνο διεργασία, δηλαδή το χρόνο επικοινωνίας ανά διεργασία, ή iii) το χρόνο για ένα σύνολο



(i) Συγχρονισμένη αποστολή/λήψη



(ii) Καθυστέρηση του αποστολέα



(iii) Καθυστέρηση του παραλήπτη

Σχήμα 2.2: Χρόνος επικοινωνίας για μια λειτουργία αποστολής/λήψης μεταξύ δύο διεργασιών

συμβάντων επικοινωνίας για όλες τις σχετικές διεργασίες, δηλαδή το χρόνο επικοινωνίας για μια φάση. Στην πράξη, ο ορισμός του χρόνου επικοινωνίας και για τις τρεις περιπτώσεις εξαρτάται από ένα πλήθος παραγόντων, τους οποίους αναδεικνύουμε με το παράδειγμα στο Σχήμα 2.2.

Σε αυτό το απλό παράδειγμα, δύο διεργασίες επικοινωνούν: η διαδικασία 0 (P0) στέλνει ένα μήνυμα στη διεργασία 1 (P1). Για λόγους ευκρίνειας του παραδείγματος, υποθέτουμε ότι η αποστολή μηνυμάτων είναι ασύγχρονη (non-blocking) και η λήψη μηνυμάτων είναι σύγχρονη (blocking). Στην πρώτη περίπτωση, που φαίνεται στο Σχήμα 2.2i, οι δύο διεργασίες εκκινούν ταυτόχρονα τα συμβάντα επικοινωνίας, τη στιγμή $t = 0$. Ο χρόνος για τη λειτουργία αποστολής από τη διεργασία P0 είναι t_s, ενώ ο χρόνος για τη λειτουργία λήψης από τη διεργασία P1 είναι t_r. Ωστόσο, ο χρόνος t_r δεν αντιστοιχεί απαραίτητα στον χρόνο που απαιτείται για τη λειτουργία λήψης: καθώς η λήψη είναι σύγχρονη, ο χρόνος t_r περιλαμβάνει κάποιο χρόνο αναμονής για τη διεργασία 1, η οποία

περιμένει την έναρξη της μετάδοσης του μηνύματος. Ο χρόνος επικοινωνίας για τη διεργασία 0 στην περίπτωση αυτή είναι $t_{P0} = t_s$ και για τη διεργασία 1 είναι $t_{P1} = t_r$, συμπεριλαμβανομένου του άεργου χρόνου (idle time) για τη δεύτερη. Ο συνολικός χρόνος επικοινωνίας είναι $t = max\{t_{P0}, t_{P1}\} = t_r$.

Η δεύτερη περίπτωση, που φαίνεται στο Σχήμα 2.2ii, καταδεικνύει την επίδραση του *καθυστερημένου αποστολέα*. Η διαδικασία 1 εκκινεί τη σύγχρονη λειτουργία λήψης τη χρονική στιγμή $t = 0$, ωστόσο η διεργασία 0 δεν εκκινεί τη λειτουργία αποστολής μέχρι τη στιγμή $t = d_s$. Σε αυτήν την περίπτωση, ο χρόνος για τη λειτουργία αποστολής εξακολουθεί να είναι t_s, καθώς η αποστολή είναι ασύγχρονη, αλλά ο χρόνος για τη λειτουργία λήψης είναι $t'_r > t_r$ και περιλαμβάνει περισσότερο άεργο χρόνο στην πλευρά της διεργασίας 1. Ο χρόνος επικοινωνίας για τη διαδικασία 0 είναι $t_{P0} = t_s$ και για τη διαδικασία 1 είναι $t_{P1} = t'_r$, συμπεριλαμβανομένου του πρόσθετου άεργου χρόνου, εδώ ίσου με d_s. Ο συνολικός χρόνος επικοινωνίας μπορεί τώρα να οριστεί με δύο τρόπους. Όπως και στην περίπτωση των συγχρονισμένων συμβάντων επικοινωνίας, μπορούμε να ορίσουμε το συνολικό χρόνο επικοινωνίας ως $t = max\{t_{P0}, t_{P1}\} = t'_r$ ή μπορούμε να υποθέσουμε ότι η διαδικασία 0 χρησιμοποίησε το χρόνο μεταξύ 0 και d_s για να πραγματοποιήσει κάποιο υπολογισμό. Έτσι, μπορούμε να υποθέσουμε ότι αυτή η χρονική περίοδος αντιστοιχεί στην επικάλυψη υπολογισμών/επικοινωνίας και μπορούμε να ορίσουμε το συνολικό χρόνο επικοινωνίας ως $t = max\{t_{P0}, t_{P1}\} - t_{overlap} = t'_r - d_s$.

Η τρίτη περίπτωση (βλ. Σχήμα 2.2iii) είναι η περίπτωση του καθυστερημένου παραλήπτη, όπου η διεργασία 1 δεν ξεκινά τη λήψη του μηνύματος μέχρι τη στιγμή $t = d_r$. Σε αυτό το σενάριο, ο χρόνος t''_r μπορεί να μην περιλαμβάνει καθόλου άεργο χρόνο, αν η μετάδοση του μηνύματος έχει ολοκληρωθεί. Ο τοπικός χρόνος επικοινωνίας για κάθε διεργασία μπορεί να οριστεί όπως πριν. Ωστόσο, ο συνολικός χρόνος επικοινωνίας δεν είναι δυνατόν να οριστεί: τα δύο συμβάντα επικοινωνίας δεν επικαλύπτονται. Ο χρόνος t''_r εξαρτάται από την περίοδο $d_r - t_s$ και από το αν η μετάδοση ολοκληρώνεται εντός αυτής της περιόδου ή όχι. Ο Wolf και οι συνεργάτες του [Wolf et al., 2005] υποστηρίζουν ότι ο χρόνος επικοινωνίας δεν μπορεί να μετρηθεί στην τρίτη περίπτωση, όταν ένα συμβάν αποστολής έχει ολοκληρωθεί πριν ξεκινήσει το αντίστοιχο συμβάν λήψης.

Συνολικά, ο ορισμός του χρόνου επικοινωνίας δεν είναι εύκολος, ακόμη και στο απλό παράδειγμα της μετάδοσης ενός μοναδικού μηνύματος από μία διεργασία σε μία άλλη. Το γεγονός αυτό επηρεάζει ανάλογα το ζήτημα της μέτρησης του χρόνου επικοινωνίας. Η συλλογή μετρήσεων αναφοράς της επικοινωνίας στο MPI απαιτεί λεπτό συγχρονισμό των διεργασιών και χρονισμό υψηλής ακρίβειας, ώστε να εξαιρούνται φαινόμενα στρέβλωσης του χρόνου επικοινωνίας (time skewing), όπως και τα φαινόμενα του καθυστερημένου αποστολέα/καθυστερημένου παραλήπτη, που περιγράφονται στο παράδειγμά

μας, έτσι ώστε να λαμβάνονται έγκυρες και αναπαράξιμες μετρήσεις [Hoefler et al., 2010a; Hunold and Carpen-Amarie, 2016].

2.4.2 Χρόνος επικοινωνίας σε παράλληλες εφαρμογές

Ο ορισμός του χρόνου επικοινωνίας στο πλαίσιο μιας εφαρμογής είναι ακόμα πιο περίπλοκος από ό,τι στην περίπτωση απλής μετάδοσης μηνύματος, για πολλούς λόγους. Πρώτον, στις παράλληλες εφαρμογές μεγάλης κλίμακας, οι διεργασίες επικοινωνούν πολλές φορές με διάφορα σχήματα επικοινωνίας. Δεύτερον, τα σχήματα επικοινωνίας είναι πιο πολύπλοκα από μια απλή λειτουργία αποστολής-λήψης. Μία μοναδική κλήση σε κάποια συνάρτηση συλλογικής επικοινωνίας, όπως, για παράδειγμα, μια εκπομπή (broadcast), περιλαμβάνει πολλαπλές ανταλλαγές μηνυμάτων. Τρίτον, οι φάσεις επικοινωνίας αλληλεπιδρούν με τις φάσεις υπολογισμού, και οι τελευταίες μπορούν να επηρεάσουν τη διάρκεια των πρώτων. Τέταρτον, ο αριθμός των διεργασιών που χρησιμοποιούνται από τις παράλληλες εφαρμογές μεγάλης κλίμακας είναι της τάξης των χιλιάδων ή εκατομμυρίων. Επομένως, η παρακολούθηση των συμβάντων επικοινωνίας και των μεταξύ τους επιδράσεων είναι δύσκολη, ακόμα και μετά την εκτέλεση της εφαρμογής. Πέμπτον, οι παράλληλες εφαρμογές αποφεύγουν τη χρήση συναρτήσεων ρητού συγχρονισμού, όπως τα barriers.

Ένα απλό μοντέλο για το συνολικό χρόνο μιας εφαρμογής είναι το εξής:

$$T_{total} = T_{comp} + T_{comm} \quad (1)$$

Αυτό το μοντέλο υποθέτει μια κοινή μέτρηση για τον χρόνο υπολογισμού και μια κοινή μέτρηση για τον χρόνο επικοινωνίας, για όλες τις διεργασίες. Ένα πιο περίπλοκο μοντέλο για το συνολικό χρόνο επικοινωνίας δίνεται στο [Barker et al., 2009] ως εξής:

$$T_{total} = T_{comp} + T_{comm} - T_{overlap} \quad (2)$$

λαμβάνοντας υπόψη τις επικαλύψεις μεταξύ επικοινωνίας και υπολογισμών (overlap). Αυτό το μοντέλο υποθέτει μια εφαρμογή με μία φάση υπολογισμών και μία φάση επικοινωνίας, όπου οι υπολογισμοί ισοκατανέμονται τέλεια μεταξύ των διεργασιών, οπότε ο χρόνος T_{comp} είναι ίσος για όλες τις διεργασίες και ο χρόνος T_{comm} αντιστοιχεί στον μέγιστο χρόνο επικοινωνίας μεταξύ όλων των διεργασιών. Για μια εφαρμογή που περιλαμβάνει M φάσεις υπολογισμού και N φάσεις επικοινωνίας, μπορούμε να επεκτείνουμε αυτό το μοντέλο ως εξής:

$$T_{total} = \sum_{i=1}^{M} t_{comp,i} + \sum_{j=1}^{N} t_{comm,j} - T_{overlap} \quad (3)$$

Αναφερόμαστε σε μια φάση επικοινωνίας ως ένα σύνολο συμβάντων επικοινωνίας που σχετίζονται στο χώρο και στο χρόνο, και σε μια φάση υπολογισμών ως ένα μετρήσιμο σύνολο υπολογισμών. Για να αληθεύει το παραπάνω μοντέλο, πρέπει να να ισχύουν ορισμένες υποθέσεις. Πρώτον, είναι αναγκαίο να είναι δυνατή η μέτρηση ενός συνολικού χρόνου $t_{comp,i}$ και $t_{comm,j}$ για κάθε φάση υπολογισμού i και φάση επικοινωνίας j αντίστοιχα. Έτσι, πρέπει κανείς να είναι σε θέση να υποθέσει ότι, για μια φάση, είτε είναι δυνατή η παρατήρηση ίσου χρόνου για όλες τις διεργασίες είτε είναι δυνατή η υπόθεση ότι ο συνολικός χρόνος της φάσης ισούται με το μέγιστο χρόνο της φάσης μεταξύ των διεργασιών. Ωστόσο, αν ο φόρτος των υπολογισμών δεν ισοκατανέμεται μεταξύ των διεργασιών, μία φάση υπολογισμών μπορεί να προκαλέσει άμεση καθυστέρηση στο επόμενο συμβάν επικοινωνίας μίας διεργασίας, η οποία μπορεί να διαδοθεί σε όλη τη φάση επικοινωνίας, στρεβλώνοντας χωρικά σχετιζόμενα γεγονότα επικοινωνίας με τέτοιο τρόπο που να αναιρεθεί η χρονική τους συσχέτιση. Έτσι, για να ισχύει το παραπάνω μοντέλο, πρέπει να υποθέσουμε ότι το φορτίο όλων των φάσεων υπολογισμού είναι ισορροπημένο μεταξύ των διεργασιών. Δεύτερον, είναι απαραίτητο να είναι κανείς σε θέση να μετρήσει το χρόνο αλληλεπικάλυψης των υπολογισμών και της επικοινωνίας $T_{overlap}$. Αυτός ο χρόνος εξαρτάται από το διαθέσιμο χρόνο για την απόκρυψη της επικοινωνίας, το χρόνο των υπολογισμών που επικαλύπτεται και τις εξαρτήσεις δεδομένων μεταξύ διαδοχικών φάσεων υπολογισμού και επικοινωνίας [Sancho et al., 2006]. Η ικανότητα μέτρησης του χρόνου επικάλυψης απαιτεί γνώση των ακριβών χρόνων των αφίξεων αιτημάτων για αποστολές/λήψεις μηνυμάτων [Chen et al., 2009]. Ωστόσο, πρέπει να σημειώσουμε ότι, όταν οι φάσεις υπολογισμού και επικοινωνίας αλληλεπικαλύπτονται, η έννοια της επικοινωνίας ως φάσης καταρρέει και είναι πιθανό να εμφανιστούν άμεσες ή έμμεσες καθυστερήσεις. Ορισμένες πρόσφατες μελέτες πραγματοποιούν ανάλυση μετά την εκτέλεση της εφαρμογής (post-mortem), για τον εντοπισμό των φάσεων της εφαρμογής, συμπεριλαμβανομένων των φάσεων επικοινωνίας, είτε προσδιορίζοντας λογικές φάσεις οι οποίες δεν εμφανίζονται ως χρονικές φάσεις, λόγω κάποιας πηγής καθυστερήσεων [Isaacs et al., 2014], είτε με κατάτμηση των ιχνών της εφαρμογής σε σύνολα σχετιζόμενων συμβάντων [Alawneh et al., 2016].

Συνοψίζοντας, ο ορισμός του χρόνου επικοινωνίας μιας εφαρμογής μπορεί να αποδοθεί μόνο όταν i) δεν υπάρχει ανισορροπία φορτίου μεταξύ των διεργασιών και ii) δεν υπάρχει αλληλεπικάλυψη μεταξύ υπολογισμών και επικοινωνίας. Στην περίπτωση εφαρμογών όπου ο υπολογισμός και η επικοινωνία αλληλεπικαλύπτονται λόγω της φύσης του αλγορίθμου, π.χ. εφαρμογές με σχήματα κυματομορφών, είναι δυνατός ο ορισμός μόνο του χρόνου επικοινωνίας ανά διεργασία της εφαρμογής. Έτσι, στο πλαίσιο αυτής της εργασίας, ορίζουμε το συνολικό χρόνο μιας εφαρμογής ως εξής:

$$T_{total} = \sum_{i=1}^{M} t_{comp,i} + \sum_{j=1}^{N} t_{comm,j} \quad (4)$$

Το συγκεκριμένο μοντέλο υποθέτει ότι δεν εμφανίζονται αλληλεπικαλυπτόμενες φάσεις και ότι οι όλες οι φάσεις επικοινωνίας είναι διακριτές και ισορροπημένες.

2.5 Μοντελοποίηση του χρόνου επικοινωνίας

Η μοντελοποίηση του χρόνου επικοινωνίας είναι τόσο περίπλοκη όσο και η έννοια του χρόνου επικοινωνίας και της επίδοσης της επικοινωνίας. Η διαδικασία της μοντελοποίησης παρουσιάζει πολλαπλές προκλήσεις και μπορεί κανείς να υιοθετήσει διαφορετικές προσεγγίσεις για τη μοντελοποίηση του χρόνου επικοινωνίας, ανάλογα με τον στόχο της διαδικασίας μοντελοποίησης, δηλαδή τον σκοπό για τον οποίο κατασκευάζεται το μοντέλο. Σε αυτή την ενότητα, συζητάμε τις διάφορες ιδιότητες ενός μοντέλου για την επικοινωνία και τις διαφορετικές προσεγγίσεις, καθώς και τη σχέση τους με τους διάφορους παράγοντες που επηρεάζουν την επίδοση της επικοινωνίας.

2.5.1 Ιδιότητες των μοντέλων για το χρόνο επικοινωνίας

Οι δύο θεμελιώδεις ιδιότητες ενός μοντέλου για το χρόνο επικοινωνίας είναι η *ακρίβεια* και ο *χρόνος πρόβλεψης*. Η ακρίβεια αναφέρεται στην ικανότητα του μοντέλου να παρέχει μια πρόβλεψη για το χρόνο επικοινωνίας με τη μικρότερη δυνατή απόκλιση από τον πραγματικό (μετρημένο) χρόνο επικοινωνίας. Ο χρόνος της πρόβλεψης αναφέρεται στη χρονική στιγμή πριν, κατά τη διάρκεια ή μετά τη διάρκεια της εκτέλεσης μίας εφαρμογής, όταν το μοντέλο είναι σε θέση να αποδώσει μια πρόβλεψη για το χρόνο επικοινωνίας του. Και οι δύο ιδιότητες καθορίζουν τις πιθανές χρήσεις ενός μοντέλου. Ένα απόλυτα ακριβές μοντέλο μπορεί να υποστηρίξει οποιοδήποτε σενάριο που απαιτεί προβλέψεις για το χρόνο επικοινωνίας, ωστόσο, στην πράξη, δεν απαιτούνται πάντα ακριβείς προβλέψεις για τη λήψη αποφάσεων. Για παράδειγμα, εάν ο στόχος του μοντέλου πρόβλεψης είναι να αποφασίσει εάν θα πρέπει να ενεργοποιηθεί μια συγκεκριμένη βελτιστοποίηση για την επικοινωνία, για μια εφαρμογή που εκτελείται σε ένα συγκεκριμένο σύστημα, τότε η ακρίβεια του μοντέλου πρέπει να είναι αρκετά καλή ώστε να μπορεί να διακρίνει μεταξύ περιπτώσεων όπου η βελτιστοποίηση θα μειώσει το χρόνο επικοινωνίας και περιπτώσεων όπου η βελτιστοποίηση δεν θα έχει αντίκτυπο στον χρόνο επικοινωνίας. Επομένως, μικρότερη ακρίβεια μπορεί να γίνει ανεκτή εάν το μοντέλο εξυπηρετεί το σκοπό για τον οποίο έχει κατασκευαστεί. Ο χρόνος πρόβλεψης είναι εξίσου σημαντικός για τη χρήση του μοντέλου. Στην ιδανική περίπτωση, ένα μοντέλο θα πρέπει να είναι σε θέση να παρέχει προβλέψεις για τον χρόνο επικοινωνίας

πολύ νωρίτερα από την εκτέλεση της εφαρμογής, στατικά. Σε αυτή την περίπτωση, το μοντέλο μπορεί να υποστηρίξει αποφάσεις κατά τον χρόνο μεταγλώττισης, π.χ. ενεργοποίηση και απενεργοποίηση βελτιστοποιήσεων, ή πριν από την εκτέλεση, όπως αποφάσεις χρονοδρομολόγησης, ή κατά τη διάρκεια εκτέλεσης, όπως αυτόματη ρύθμιση της εφαρμογής. Εάν ο χρόνος πρόβλεψης είναι μετά την εκτέλεση της εφαρμογής, το μοντέλο δεν είναι πλέον προγνωστικό αλλά επεξηγηματικό και είναι συνεπώς χρήσιμο μόνο για την ανάλυση της επίδοσης και της συμπεριφοράς της εφαρμογής ή για προσομοιώσεις που στηρίζονται σε μοντέλα για μία εφαρμογή. Εάν ο χρόνος πρόβλεψης είναι ακριβώς πριν (just-in-time) από την εκτέλεση της εφαρμογής, το μοντέλο μπορεί να υποστηρίξει αποφάσεις κατά τη διάρκεια εκτέλεσης. Ομοίως με την ακρίβεια, ο χρόνος πρόβλεψης χρειάζεται να είναι αρκετά καλός για τον σκοπό της πρόβλεψης. Τόσο η ακρίβεια όσο και ο χρόνος πρόβλεψης εξαρτώνται σε μεγάλο βαθμό από τους παράγοντες της επίδοσης της επικοινωνίας που ενσωματώνει ένα μοντέλο, όπως συζητάμε στην επόμενη υποενότητα.

Μια άλλη σημαντική ιδιότητα ενός μοντέλου για το χρόνο επικοινωνίας είναι το αντικείμενο της πρόβλεψης, δηλαδή αν το μοντέλο στοχεύει στη μοντελοποίηση του χρόνου επικοινωνίας για μια λειτουργία επικοινωνίας, για μια διεργασία ή για μια φάση επικοινωνίας. Η μοντελοποίηση μιας λειτουργίας επικοινωνίας μπορεί να είναι ιδιαίτερα χρήσιμη για τον από κοινού σχεδιασμό υλικού/λογισμικού και για το σχεδιασμό του λογισμικού επικοινωνίας. Η μοντελοποίηση του χρόνου επικοινωνίας για μια διεργασία μπορεί να υποστηρίξει αποφάσεις σχετικά με τον επανασχεδιασμό των σχημάτων επικοινωνίας και άλλων βελτιστοποιήσεων για παράλληλες εφαρμογές. Τα μοντέλα για μια φάση επικοινωνίας είναι χρήσιμα για αποφάσεις σχετικά με την τοποθέτηση και την απεικόνιση μιας εφαρμογής σε ένα σύστημα ή την κλιμακωσιμότητα μιας φάσης επικοινωνίας.

Μια πρόσθετη ιδιότητα των μοντέλων για το χρόνο επικοινωνίας είναι η εξάρτησή τους από το υποκείμενο σύστημα, η οποία ταξινομεί τα μοντέλα σε αναλυτικά, ημι-εμπειρικά ή εμπειρικά. Τα αναλυτικά μοντέλα παρέχουν μία ασυμπτωτική μορφή για το χρόνο επικοινωνίας, με βάση τις ονομαστικές τιμές του συστήματος, όπως ο χρόνος απόκρισης και το εύρος ζώνης. Ένα αναλυτικό μοντέλο μπορεί να κατασκευαστεί για οποιοδήποτε σύστημα, χωρίς καμία εξάρτηση από την ύπαρξη του ίδιου του συστήματος. Ως εκ τούτου, τα αναλυτικά μοντέλα αποκτούν εξαιρετική χρησιμότητα για το σχεδιασμό ενός συστήματος ή τον από κοινού σχεδιασμό υλικού / λογισμικού. Τα ημι-εμπειρικά μοντέλα επεκτείνουν τις παραδοχές των αναλυτικών μοντέλων με μετρήσεις και παρατηρήσεις από το υποκείμενο σύστημα. Στην πράξη, η χρήση πληροφορίας από το υποκείμενο σύστημα βελτιώνει την ακρίβεια των αναλυτικών μοντέλων και καθιστά τα ημι-εμπειρικά μοντέλα κατάλληλα για πιο περίπλοκες αποφάσεις σχετικά με το συγκεκριμένο σύστημα. Τα εμπειρικά μοντέλα

βασίζονται στο υποκείμενο σύστημα, χρησιμοποιώντας μετρήσεις και παρατηρήσεις από το σύστημα για την παραγωγή της μορφής του μοντέλου, αντί να επεκτείνουν τις θεωρητικές υποθέσεις των αναλυτικών μοντέλων. Τα εμπειρικά μοντέλα οδηγούν σε σημαντικά μεγαλύτερη ακρίβεια από τα αναλυτικά και ημι-εμπειρικά μοντέλα, αν και η εξάρτησή τους από το υποκείμενο σύστημα τα καθιστά ακατάλληλα για το σχεδιασμό του συστήματος. Η εξάρτηση από το υποκείμενο σύστημα, εκτός από το γεγονός ότι αποτελεί ιδιότητα του μοντέλου, καθορίζει επιπλέον την προσέγγιση που ακολουθείται για τη μοντελοποίηση της επίδοσης της επικοινωνίας.

2.5.2 Συμβιβασμοί στη μοντελοποίηση της επικοινωνίας

Ένα ελάχιστο μοντέλο για το χρόνο επικοινωνίας απαιτεί πληροφορίες από την εφαρμογή και την αρχιτεκτονική του συστήματος, π.χ. το γράφο επικοινωνίας της εφαρμογής και τα χαρακτηριστικά της τεχνολογίας δικτύου. Ένα τέτοιο μοντέλο μπορεί να είναι αναλυτικό και παρέχει προβλέψεις σε στατικό χρόνο (ή νωρίς κατά το χρόνο εκτέλεσης, μετά τον καθορισμό των παραμέτρων εισόδου της εφαρμογής). Ωστόσο, τέτοια μοντέλα θυσιάζουν την ακρίβεια για την απλότητα: για να επιτευχθεί η υψηλότερη δυνατή ακρίβεια, ένα μοντέλο πρέπει να περιλαμβάνει πρόσθετες κρίσιμες πληροφορίες, που σχετίζονται με το σχήμα κίνησης δεδομένων. Όμως, η διαμόρφωση του σχήματος κίνησης δεδομένων, που καταγράφει με μεγαλύτερη ακρίβεια την κατανομή της ροής δεδομένων στο δίκτυο, απαιτεί γνώση της απεικόνισης των διεργασιών και του σχήματος της κατανομής των κόμβων. Αυτή η πληροφορία γίνεται διαθέσιμη μόλις πριν από την εκτέλεση της εφαρμογής. Έτσι, ένα μοντέλο που ενσωματώνει τις λεπτομέρειες του σχήματος κίνησης δεδομένων θυσιάζει το χρόνο πρόβλεψης για μεγαλύτερη ακρίβεια. Επιπλέον, παρουσιάζει εξάρτηση από το υποκείμενο σύστημα, με αποτέλεσμα να είναι ημι-εμπειρικό ή εμπειρικό.

Ακόμη και με επίγνωση του σχήματος κίνησης δεδομένων, οι μη ντετερμινιστικές πηγές διακύμανσης του χρόνου, όπως ο θόρυβος του λειτουργικού συστήματος και ο ανταγωνισμός μεταξύ των εφαρμογών, μπορούν να παρακολουθούνται μόνο κατά το χρόνο εκτέλεσης. Ένα μοντέλο που ενσωματώνει τέτοιες πληροφορίες, αν είναι δυνατή η ενσωμάτωσή τους, χάνει την προγνωστική ιδιότητά του και αποτελεί ένα επεξηγηματικό μοντέλο. Η ακρίβεια κάθε μοντέλου πρόβλεψης υπόκειται πάντοτε στον περιορισμό των φαινομένων του χρόνου εκτέλεσης που μπορούν να επηρεάσουν αυθαίρετα τον χρόνο επικοινωνίας. Παρομοίως, η παρουσία της ανισοκατανομής φορτίου υπολογισμών και η επικάλυψη υπολογισμών και επικοινωνίας επηρεάζουν το χρόνο επικοινωνίας. Για την ενσωμάτωση αυτών των φαινομένων σε ένα μοντέλο, απαιτείται γνώση των ακριβών χρόνων άφιξης αιτήσεων για συμβάντα επικοινωνίας

ή/και καθυστερήσεων και διάδοσης καθυστερήσεων σε όλη τη φάση επικοινωνίας. Ακόμη και στην περίπτωση αυτή, η μοντελοποίηση του χρόνου επικοινωνίας και η πρόβλεψη υπό την παρουσία ανισοκατονομής φορτίου έχουν μικρή ή καθόλου αξία: η εξισορρόπηση του φορτίου είναι το κύριο μέλημα στην περίπτωση αυτή. Εξ όσων γνωρίζουμε, δεν υπάρχει προηγούμενη εργασία στην πρόβλεψη του χρόνου επικοινωνίας που να στοχεύει σε εφαρμογές με ανισοκατανομή φορτίου ή αλληλεπικάλυψη υπολογισμών/επικοινωνίας. Η επίδραση των υπολογισμών στον χρόνο επικοινωνίας μπορεί να ληφθεί υπόψη μόνο σε ολιστικές προσεγγίσεις για τη μοντελοποίηση της κλιμάκωσης παράλληλων εφαρμογών, όπως στην εργασία των Shudler κ.ά. [Shudler et al., 2015].

2.6 Ορισμός προβλήματος

Ο σκοπός αυτής της εργασίας είναι η πρόβλεψη του χρόνου επικοινωνίας $\hat{t}_i$ για μια ολόκληρη φάση επικοινωνίας i μιας εφαρμογής και να αποδώσει μια πρόβλεψη για το συνολικό χρόνο επικοινωνίας της εφαρμογής, $\hat{T} = \sum_i \hat{t}_i$. Η προσέγγισή μας για τη μοντελοποίηση είναι εμπειρική, καθώς χρησιμοποιούμε μετρήσεις αναφοράς από το υποκείμενο σύστημα για την κατασκευή μοντέλων πρόβλεψης του χρόνου επικοινωνίας. Πρέπει να σημειώσουμε ότι, αντίθετα με τα αναλυτικά μοντέλα [Hockney, 1994; Culler et al., 1993; Alexandrov et al., 1995], τα οποία προβλέπουν τον χρόνο επικοινωνίας από την οπτική μίας μόνο διεργασίας, δηλαδή του χρόνου ολοκλήρωσης ενός αριθμού μεταδόσεων μηνυμάτων, όπως μετράται από τη διαδικασία αποστολέα, εστιάζουμε στην πρόβλεψη του χρόνου μιας ολόκληρης φάσης επικοινωνίας, όπως μετράται από όλες τις συμμετέχουσες διεργασίες. Όσον αφορά τις ιδιότητες του μοντέλου, η προσέγγισή μας παρέχει προγνωστικά μοντέλα, με χρόνο πρόβλεψης πριν από την εκτέλεση της εφαρμογής. Όσον αφορά την ακρίβεια, στοχεύουμε στην κατασκευή μοντέλων που είναι σε θέση να προβλέψουν τον χρόνο επικοινωνίας μιας ισορροπημένης φάσης επικοινωνίας μιας παράλληλης εφαρμογής, με ακρίβεια ικανή να επιτρέπει ουσιαστικές συγκρίσεις μεταξύ του χρόνου επικοινωνίας και του χρόνου υπολογισμού, ώστε να είναι δυνατή η χρήση των μοντέλων για την επιλογή/ενεργοποίηση τεχνικών βελτιστοποίησης των παράλληλων εφαρμογών και για τη βελτιστοποίηση της διαχείρισης πόρων σε συστήματα μεγάλης κλίμακας. Επιπρόσθετα, στοχεύουμε στην κατασκευή μοντέλων που παρουσιάζουν καλή προσαρμογή, ώστε να είναι δυνατή η διάκριση μεταξύ ρυθμίσεων εκτέλεσης μίας εφαρμογής για διαφορετικά μεγέθη προβλημάτων, αριθμούς κόμβων και πυρήνων και η διάταξη αυτών, ώστε να είναι δυνατή η χρήση των μοντέλων για σχετικές βελτιστοποιήσεις επικοινωνίας.

Πίνακας 2.1: Συστήματα στην παρούσα διατριβή: μία σύνοψη

	Vilje	**Piz Daint**	**ARIS**
Κατασκευαστής	SGI Altix ICE X	Cray XC30	IBM NeXtScale nx360M5
Κόμβοι	1404	5272	426
Αρχιτεκτονική κόμβου	2 x 8-core Intel SandyBridge	8-core Intel SandyBridge + NVIDIA Tesla	2 x 10-core Intel Xeon E5-2680v2
Δίκτυο Διασύνδεσης	InfiniBand FDR	Cray Aries	InfiniBand FDR
Τοπολογία	Enhanced hypercube	Dragonfly	Fat tree
Υλοποίηση MPI	SGI MPI	Cray MPICH	Intel MPI

2.6.1 Συστήματα

Η προτεινόμενη μέθοδος για την πρόβλεψη του χρόνου επικοινωνίας είναι μια γενική μεθοδολογία που στοχεύει συστήματα μεγάλης κλίμακας και υπερυπολογιστές που χρησιμοποιούν δίκτυα διασύνδεσης υψηλής επίδοσης. Στην παρούσα εργασία εφαρμόζουμε την προτεινόμενη μεθοδολογία σε τρία συστήματα μεγάλης κλίμακας, για τα οποία αναπτύσσουμε μοντέλα πρόβλεψης της επικοινωνίας και αξιολογούμε την ακρίβεια των προβλέψεων. Το πρώτο σύστημα είναι ο υπερυπολογιστής *Piz Daint*, που πλέον είναι ένα σύστημα Cray XC50 / XC40 (σύστημα Cray XC30 μέχρι το 2016), το οποίο χρησιμοποιεί το δίκτυο Cray Aries σε τοπολογία Dragonfly. Το δεύτερο σύστημα είναι ο υπερυπολογιστής *Vilje*, ένα σύστημα SGI που χρησιμοποιεί το InfiniBand FDR ως δίκτυο διασύνδεσης, με τοπολογία ενισχυμένου υπερκύβου. Το τρίτο σύστημα είναι το σύστημα *ARIS*, ένα σύστημα IBM NeXtScale nx360M5, το οποίο χρησιμοποιεί επίσης το InfiniBand FDR για τη διασύνδεση των κόμβων του σε τοπολογία fat tree δύο επιπέδων. Τα τρία συστήματα διαφέρουν ως προς την τεχνολογία δικτύου ή/και την τοπολογία και αντιπροσωπεύουν το 36,2 % των συστημάτων Top500 [top] (μερίδιο 28.2% για το InfiniBand FDR και 8 % για το Cray Aries). Η τεχνολογία InfiniBand είναι η πιο δημοφιλής τεχνολογία διασύνδεσης στη λίστα Top500, όπου χρησιμοποιείται από το 37.4% των συστημάτων. Το δίκτυο Cray Aries χρησιμοποιείται λιγότερο, ωστόσο η τοπολογία Dragonfly αναμένεται να χρησιμοποιηθεί περισσότερο στα μελλοντικά συστήματα μεγάλης κλίμακας, λόγω του υψηλού της εύρους τομής. Συνοψίζουμε τις ιδιότητες των τριών συστημάτων στον Πίνακα 2.1.

Το *Vilje* [1] είναι ένα σύστημα SGI Altix ICE X στο Νορβηγικό Πανεπιστήμιο Επιστήμης και Τεχνολογίας (NTNU), το οποίο αποτελείται από 1404 κόμβους, που απαρτίζονται από δύο επεξεργαστές Intel Sandy Bridge 8 πυρήνων και 32GB μνήμης. Το βασικό δομικό στοιχείο του συστήματος είναι το Individual Rack Unit, που φιλοξενεί 18 κόμβους και δύο *διακόπτες* FDR InfiniBand 36 θυρών. Κάθε IRU είναι ένας κόμβος της τοπολογίας του υπερκύβου. Τα IRUs στοιβάζονται σε ομάδες των οκτώ και αλληλοσυνδέονται σε μια τοπο-

[1] https://www.hpc.ntnu.no/display/hpc/Vilje

λογία 3Δ-υπερκύβων, για να σχηματίσουν ένα *rack*. Τα 19,5 racks του συστήματος σχηματίζουν έναν ενισχυμένο 8Δ-υπερκύβο διπλού δρόμου (dual rail), όπου χρησιμοποιούνται πλεονασματικές συνδέσεις για τη σύνδεση διακοπτών στις κατώτερες διαστάσεις του υπερκύβου, παρέχοντας μεγαλύτερο συνολικό εύρος ζώνης. Ο έλεγχος συμφόρησης επιτυγχάνεται μέσω ενός μηχανισμού σηματοδότησης: ο κόμβος προέλευσης ειδοποιείται και στραγγαλίζει την έγχυση πακέτων του κάθε φορά που ανιχνεύεται συμφόρηση σε ένα διακόπτη στη διαδρομή προς τον προορισμό. Η προεπιλεγμένη υλοποίηση MPI για το Vilje είναι η υλοποίηση SGI MPI.

Ο υπερυπολογιστής *Piz Daint*[2], που τώρα αποτελεί μια εγκατάσταση Cray XC50 / XC40 στο Ελβετικό Εθνικό Υπερυπολογιστικό Κέντρο (CSCS), ήταν μέχρι τα τέλη του 2016 ένας υπερυπολογιστής Cray XC30 με 5272 υπολογιστικούς κόμβους. Κάθε κόμβος περιλαμβάνει έναν επεξεργαστή Intel Sandy Bridge 8 πυρήνων, μία NVIDIA Tesla K20X GPU και 32 GB μνήμης RAM. Στην αναβάθμιση από XC30 σε XC50 / XC40, οι υπολογιστικοί κόμβοι έχουν αναβαθμιστεί, αλλά το δίκτυο διασύνδεσης και η τοπολογία κόμβων παραμένουν ίδια. Η εργασία που παρουσιάζεται σε αυτή τη διατριβή πραγματοποιήθηκε στην έκδοση Cray XC30 του συστήματος. Η βασική δομική μονάδα του δικτύου είναι το *Aries SoC*, με τέσσερις κάρτες δικτύου (NIC) που συνδέουν τέσσερις κόμβους, σχηματίζοντας ένα blade. Δεκαέξι blades είναι τοποθετημένα σε ένα *chassis*. Ένα ζεύγος των τριών chassis, σχηματίζει μια *ομάδα* (group) του δικτύου. Τα Cray Aries chips μέσα σε μια ομάδα αλληλοσυνδέονται σε πλήρη γράφο μέσω του δικτύου του chassis, με συνδέσεις χαλκού, ενώ οι 14 ομάδες του Piz Daint αλληλοσυνδέονται σε πλήρη γράφο μέσω οπτικών καλωδίων ελαφρώς χαμηλότερου εύρους ζώνης. Ο έλεγχος συμφόρησης επιτυγχάνεται μέσω προσαρμοστικής δρομολόγησης σε επίπεδο πακέτων: επιλέγεται μια μη ελάχιστη διαδρομή εάν το εκτιμώμενο φορτίο συνδέσμου είναι χαμηλότερο από αυτό της ελάχιστης διαδρομής. Περισσότερες λεπτομέρειες σχετικά με το Cray Aries βρίσκονται στο [Alverson et al., 2012]. Η προεπιλεγμένη υλοποίηση του MPI για το Piz Daint είναι το Cray MPICH.

Το *ARIS* [3] είναι ένα σύστημα IBM NeXtScale nx360M5 στο Εθνικό Δίκτυο Έρευνας και Τεχνολογίας (ΕΔΕΤ Α.Ε.). Το κύριο μέρος του περιλαμβάνει 426 κόμβους δύο επεξεργαστών Intel Xeon Ivy Bridge των 10 πυρήνων και 64GB μνήμης. Όλοι οι κόμβοι συνδέονται σε έναν διακόπτη Mellanox SX6536, 648 θυρών, ο οποίος υλοποιεί εσωτερικά μια τοπολογία fat tree δύο επιπέδων. Στην πράξη, η τοπολογία των fat trees αναπτύσσεται ως εξής: ο διακόπτης των 648 θυρών υλοποιείται με 36 διακόπτες των 36 θυρών. Στο χαμηλότερο επίπεδο του fat tree, ομάδες των 18 κόμβων συνδέονται σε έναν από τους 18 διακό-

[2] http://www.cscs.ch/computers/piz_daint/index.html
[3] http://www.hpc.grnet.gr

πτες 36 θυρών. Στο υψηλότερο επίπεδο του fat tree, 18 διακόπτες των 36 θυρών συνδέονται με όλους τους διακόπτες του κατώτερου επιπέδου. Ως εκ τούτου, η τοπολογία είναι 1:1 και ασύγχρονη, παρέχοντας πλήρες εύρος τομής. Το σύστημα προσφέρει πολλαπλές υλοποιήσεις MPI στους χρήστες. Χρησιμοποιούμε το Intel MPI για όλα τα πειράματά μας.

2.6.2 Παράλληλες εφαρμογές

Η παρούσα εργασία επικεντρώνεται σε μια τυπική κατηγορία εφαρμογών μεγάλης κλίμακας που παρουσιάζουν φάσεις point-to-point επικοινωνίας, που είναι συμμετρικές ως προς τον αριθμό των μηνυμάτων που ανταλλάσσουν οι διεργασίες. Σύμφωνα με την ταξινόμηση των εφαρμογών που παρουσιάσαμε στο προηγούμενο κεφάλαιο, στοχεύουμε σε σχήματα επικοινωνίας που εμφανίζονται σε προβλήματα σε δομημένα και αδόμητα πλέγματα, προβλήματα N-σωματιδίων που επιλύονται σε καρτεσιανά πλέγματα και εφαρμογές γραμμικής άλγεβρας που παρουσιάζουν κάποιες φάσεις point-to-point επικοινωνίας. Οι εφαρμογές-στόχοι μας περιλαμβάνουν, μεταξύ άλλων, προσομοιώσεις κβαντικής χρωμοδυναμικής (QCD), υδροδυναμικής, καιρού, μοριακής δυναμικής και άλλες. Με αυτόν τον τρόπο, στοχεύουμε στην ανάπτυξη μιας μεθοδολογίας που θα αντιμετωπίζει τις βασικές προκλήσεις στην πρόβλεψη του χρόνου επικοινωνίας και θα παρέχει ακριβή μοντέλα πρόβλεψης για μεγάλο αριθμό εφαρμογών μεγάλης κλίμακας. Δεδομένου ότι ο στόχος μας είναι η πρόβλεψη της επικοινωνίας ως φάσης, εστιάζουμε σε εφαρμογές όπου οι φάσεις υπολογισμού δεν επηρεάζουν σημαντικά τον χρόνο επικοινωνίας, δηλαδή σε εφαρμογές με ισορροπημένες φάσεις υπολογισμού και εφαρμογές που οι υπολογισμοί δεν επικαλύπτονται με την επικοινωνία.

Σε σχέση με το προγραμματιστικό μοντέλο, υποθέτουμε ότι η επικοινωνία είναι ασύγχρονη, καθώς και ότι τα ανταλλασσόμενα μηνύματα αποθηκεύονται σε βοηθητικούς προσωρινούς αποθηκευτικούς χώρους (buffers), εάν είναι απαραίτητο, αποκλείοντας έτσι τους τύπους δεδομένων του MPI, καθώς η μοντελοποίησή τους εμπίπτει στην κατηγορία της πρόβλεψης του χρόνου υπολογισμών και όχι του χρόνου επικοινωνίας. Ο Αλγόριθμος 2.1 παρουσιάζει τον πυρήνα ενός απλουστευμένου μοντέλου της εφαρμογής με μία μόνο φάση υπολογισμού και μία μόνο φάση επικοινωνίας. Για να διασφαλιστεί ότι δεν υπάρχει ανισοκατανομή φορτίου και ότι ο χρόνος επικοινωνίας δεν είναι στρεβλός, χρησιμοποιούμε τη συνάρτηση *MPI_Barrier* για τον συγχρονισμό όλων των διεργασιών πριν μετρήσουμε το χρόνο επικοινωνίας, για σύγκριση με τις προβλέψεις μας. Σημειώνουμε ότι, παρόλο που αυτή η διατριβή έχει ως κίνητρο τις εφαρμογές μεγάλης κλίμακας που εκτελούνται σε υπερυπολογιστές, η προτεινόμενη προσέγγιση είναι εφαρμόσιμη σε κάθε εφαρμογή με point-to-point επικοινωνία, όπως για παράδειγμα η επεξεργασία γράφων μεγάλης

κλίμακας στο μοντέλο BSP [Valiant, 1990] στο Hama[4] ή στο Pregel[Malewicz et al., 2010].

```
1: compute()
2: for ms in MessagesToSend do
3:     pack(ms)
4: end for
5: for mr in MessagesToReceive do
6:     MPI_Irecv(mr)
7: end for
8: for ms in MessagesToSend do
9:     MPI_Isend(ms)
10: end for
11: MPI_Waitall(MessagesToSend, MessagesToRecv)
12: for mr in MessagesToReceive do
13:     unpack(mr)
14: end for
```

Αλγόριθμος 2.1: Ψευδοκώδικας για μια διεργασία MPI στο μοντέλο της εφαρμογής.

2.7 Σχετική βιβλιογραφία

Κατά την προηγούμενη εικοσαετία, η επιστημονική κοινότητα ασχολήθηκε με την κατασκευή αναλυτικών μοντέλων για το χρόνο επικοινωνίας. Το μοντέλο του Hockney [Hockney, 1994] για την point-to-point επικοινωνία, που χρησιμοποιείται έως και σήμερα, ήταν μια σημαντική προσπάθεια για την έκφραση του χρόνου επικοινωνίας για ένα ζεύγος επεξεργαστών ως συνάρτηση δύο παραμέτρων του δικτύου, συγκεκριμένα του χρόνου απόκρισης και του μέγιστου εύρους ζώνης. Παράλληλα, το μοντέλο LogP [Culler et al., 1993] έδωσε μια πιο περιγραφική έκφραση του χρόνου επικοινωνίας, βασισμένη σε χαρακτηριστικά του δικτύου και στο μέγεθος των μηνυμάτων ή των πακέτων, αποτελώντας και το πρώτο από μια σειρά αντίστοιχων μοντέλων. Τα μοντέλα LogGP [Alexandrov et al., 1995] και LogGPS [Ino et al., 2001] επέκτειναν το μοντέλο LogP με παραμέτρους για διαφορετικά μεγέθη μηνυμάτων και πρωτόκολλα μεταφοράς και χρησιμοποιούνται κατά κόρον για τη μοντελοποίηση της επίδοσης της επικοινωνίας, έχοντας ωστόσο πολλές αδυναμίες και περιορισμούς. Βασική τους αδυναμία είναι ότι επικεντρώνονται σε ένα τοπικό πα-

[4] http://hama.apache.org

ράδειγμα του δικτύου, αγνοώντας σφαιρικά δικτυακά φαινόμενα, όπως επικαλύψεις, πολλαπλά βήματα (hops), φαινόμενα ανταγωνισμού και συμφόρησης. Επεκτάσεις του LogGP προσπαθούν να αποδώσουν αυτά τα φαινόμενα, ωστόσο ξεχωριστά και όχι αθροιστικά. Για παράδειγμα, το μοντέλο LoGPC [Moritz and Frank, 1998] ενσωματώνει φαινόμενα ανταγωνισμού στις κάρτες δικτύου, το μοντέλο LoPC [Frank et al., 1997] ενσωματώνει φαινόμενα ανταγωνισμού από πολλαπλά αιτήματα επικοινωνίας από την εφαρμογή και το μοντέλο LoGPG [Moritz and Frank, 2001] ενσωματώνει φαινόμενα συμφόρησης. Καθώς αναδύονται νέες αρχιτεκτονικές δικτύου, με διεπαφές δικτύου που επιτρέπουν στις κάρτες δικτύου να εκτελούν λειτουργίες που μέχρι πρότινος εκτελούσε ο επεξεργαστής, τα μοντέλα της οικογένειας Log(G)P δεν είναι εφαρμόσιμα [Di Girolamo et al., 2015] και χρειάζεται να επεκταθούν με νέες παραμέτρους για τη μοντελοποίηση των λειτουργιών των διεπαφών δικτύου. Επιπλέον, η κατασκευή μοντέλων για την επικοινωνία μιας εφαρμογής μεγάλης κλίμακας με τη χρήση των παραμέτρων των μοντέλων της οικογένειας Log(G)P δεν είναι εύκολη. Εργασίες στη σχετική βιβλιογραφία [Mudalige et al., 2008; Bauer et al., 2012] αξιοποιούν αυτές τις παραμέτρους για την κατασκευή μοντέλων επίδοσης συγκεκριμένων εφαρμογών μεγάλης κλίμακας σε συγκεκριμένα συστήματα. Κάποιες πιο πρόσφατες εργασίες [Bédaride et al., 2013; Zhu et al., 2015] προτείνουν νέα μοντέλα δικτύων, προσπαθώντας να ενσωματώσουν φαινόμενα ανταγωνισμού, σε συγκεκριμένες, ωστόσο, αρχιτεκτονικές δικτύων και περιορίζονται, όπως και τα προηγούμενα μοντέλα, στη μοντελοποίηση χαρακτηριστικών του δικτύου και στοιχειωδών λειτουργιών επικοινωνίας. Συνολικά, τα αναλυτικά μοντέλα θυσιάζουν ακρίβεια στη μοντελοποίηση, αφού δεν ενσωματώνουν περίπλοκα χαρακτηριστικά και φαινόμενα της επικοινωνίας, για να διατηρήσουν απλές τις εκφράσεις της επίδοσης της επικοινωνίας και να είναι εφαρμόσιμα σε πολλές αρχιτεκτονικές [Hoefler et al., 2011]. Σε σχέση με τη χρήση τους, τα αναλυτικά μοντέλα είναι αποτελεσματικά για το σχεδιασμό λογισμικού επικοινωνίας και για την ταυτόχρονη σχεδίαση υλικού και λογισμικού.

Η εναλλακτική προσέγγιση στην αναλυτική μοντελοποίηση είναι η εμπειρική μοντελοποίηση, δηλαδή η αξιοποίηση μετρήσεων από το σύστημα για τη μοντελοποίηση και την πρόβλεψη της επίδοσης της επικοινωνίας μιας εφαρμογής, πραγματοποιώντας μετρήσεις αναφοράς (benchmarking) ή/και συλλέγοντας πληροφορίες κατά το χρόνο εκτέλεσης. Τα εμπειρικά μοντέλα, αντίθετα από τα αναλυτικά, μπορούν να επιτύχουν υψηλή ακρίβεια, ενσωματώνοντας αρκετή γνώση από το σύστημα, θυσιάζοντας, ωστόσο, κάποιες από τις δυνατότητες εφαρμογής τους. Η σχετική βιβλιογραφία στην εμπειρική πρόβλεψη της επίδοσης της επικοινωνίας περιλαμβάνει τις εργασίες των Jain, Bhatele κ.ά. [Jain et al., 2013; Bhatele et al., 2015], που ανέπτυξαν μοντέλα επίδοσης της επικοινωνίας στο σύστημα IBM BlueGene/Q, αξιοποιώντας τους μετρητές επί-

δοσης του δικτύου. Τα συγκεκριμένα μοντέλα παρουσιάζουν υψηλή ακρίβεια, αφού η χρήση των μετρητών επίδοσης παρέχει πλήρη γνώση του σχήματος κίνησης δεδομένων και των συνθηκών εκτέλεσης για μια δοσμένη κατανομή κόμβων, αλλά δεν είναι κατάλληλα για πρόβλεψη, αφού οι τιμές των χαρακτηριστικών που χρησιμοποιούν μπορούν να μετρηθούν μόνο μετά την εκτέλεση μιας εφαρμογής. Επιπλέον, η μεθοδολογία τους περιορίζεται σε συστήματα όπως το BlueGene/Q, όπου τα δικτυακά στοιχεία που χρησιμοποιεί μια κατανομή κόμβων για την εκτέλεση μιας εφαρμογής αποτελούν μέρος της κατανομής και δε χρησιμοποιούνται από άλλες εφαρμογές, επομένως και οι τιμές που λαμβάνονται από τους μετρητές επίδοσης αντιστοιχούν στην κίνηση δεδομένων της συγκεκριμένης εφαρμογής. Μια ημι-εμπειρική προσέγγιση ακολουθείται από τον Gahvari και τους συνεργάτες του [Gahvari et al., 2011, 2014], οι οποίοι επεκτείνουν το μοντέλο του Hockney προσθέτοντας εμπειρικές παραμέτρους, για την πρόβλεψη της επικοινωνίας της εφαρμογής Algebraic Multigrid. Η συγκεκριμένη προσέγγιση βελτιώνει την ακρίβεια του απλού αναλυτικού μοντέλου του Hockney, ενσωματώνοντας γνώση από την απεικόνιση των διεργασιών, την αρχιτεκτονική του δικτύου και την τοπολογία, ενώ είναι επίσης εφαρμόσιμη πριν την εκτέλεση της εφαρμογής, επομένως και κατάλληλη για πρόβλεψη. Επίσης, είναι εφαρμόσιμη σε πολλές εφαρμογές που εμφανίζουν point-to-point φάσεις επικοινωνίας και έχει χρησιμοποιηθεί για την πρόβλεψη της εφαρμογής Fast Multipole Method [Ibeid et al., 2016]. Συνοπτικά, τα εμπειρικά μοντέλα βελτιώνουν την ακρίβεια μοντελοποίησης ή/και πρόβλεψης και είναι αποτελεσματικά για τη βελτίωση της λήψης αποφάσεων στο χρόνο εκτέλεσης, όπως αποφάσεις χρονοδρομολόγησης και ρύθμισης των παραμέτρων των εφαρμογών. Το μειονέκτημά τους είναι ότι απαιτούν τη λήψη μετρήσεων στο σύστημα για το οποίο προορίζονται, επομένως προϋποθέτουν τη διαθεσιμότητα του συστήματος για την κατασκευή τους. Σε αντίθεση με τα προαναφερθέντα εμπειρικά μοντέλα, η μεθοδολογία μας στηρίζεται σε χαρακτηριστικά που μπορούν να αποτιμηθούν πριν την εκτέλεση της εφαρμογής. Επομένως, τα μοντέλα μας είναι προγνωστικά και όχι επεξηγηματικά. Επιπρόσθετα, η μεθοδολογία που προτείνουμε είναι εφαρμόσιμη σε οποιοδήποτε σύστημα, χωρίς κανέναν περιορισμό.

3

Κατασκευή μοντέλων πρόβλεψης του χρόνου επικοινωνίας

3.1 Εισαγωγή

Όπως συζητήθηκε στο προηγούμενο κεφάλαιο, ο χώρος των παραμέτρων που επηρεάζουν την επίδοση επικοινωνίας είναι μεγάλος και περίπλοκος. Το πρώτο βήμα προς τη μοντελοποίηση του χρόνου επικοινωνίας είναι η απαρίθμηση και η ποσοτικοποίηση χαρακτηριστικών που σκιαγραφούν την επίδοση της επικοινωνίας. Εξετάζουμε τα χαρακτηριστικά που μπορούν να οδηγήσουν σε μοντέλα πρόβλεψης, με χρόνο πρόβλεψης ακριβώς πριν από την εκτέλεση της εφαρμογής, επομένως εξετάζουμε χαρακτηριστικά που μπορούν να αποτιμώνται αμέσως μετά τη δέσμευση της κατανομής των κόμβων (και πριν από την έναρξη της εκτέλεσης της εφαρμογής).

Ορίζουμε πολλαπλά χαρακτηριστικά και τα ταξινομούμε σύμφωνα με την επίγνωσή τους σε χαρακτηριστικά της εφαρμογής, του συστήματος και της απεικόνισης των διεργασιών, ώστε να διερευνήσουμε τους συμβιβασμούς μεταξύ ακρίβειας και χρόνου πρόβλεψης. Στη συνέχεια, πραγματοποιούμε συλλογή μετρήσεων αναφοράς, για τη συλλογή ενός συνόλου μετρήσεων που θα χρησιμεύσουν ως σύνολο εκπαίδευσης για την κατασκευή του μοντέλου. Στο τρίτο βήμα, αποφασίζουμε επί των αποτελεσματικότερων μεθόδων μηχανικής μάθησης, επιλέγουμε χαρακτηριστικά, αν απαιτείται, και κατασκευάζουμε μοντέλα με διαφορετικό αριθμό χαρακτηριστικών, διαφορετικές μεθόδους και σύνολα εκπαίδευσης. Παρουσιάζουμε δύο διαφορετικές προσεγγίσεις μοντελοποίησης: η μία βασίζεται σε μεθόδους στατιστικής μάθησης και η δεύτερη βασίζεται σε μεθόδους μηχανικής μάθησης. Και οι δύο εμπίπτουν στην κατηγορία της εποπτευόμενης μάθησης.

Στις επόμενες ενότητες, παρέχουμε ορισμένες εισαγωγικές πληροφορίες σχετικά με την εποπτευόμενη μάθηση και παρουσιάζουμε τα δύο βασικά στοι-

χεία της μεθοδολογίας μας, δηλαδή τα χαρακτηριστικά για την πρόβλεψη του χρόνου επικοινωνίας και τις λεπτομέρειες για τη διαδικασία συλλογής του συνόλου αναφοράς. Στη συνέχεια, περιγράφουμε τα βήματα που ακολουθούμε για να κατασκευάσουμε διάφορα μοντέλα για το χρόνο επικοινωνίας χρησιμοποιώντας μεθόδους στατιστικής μάθησης και μηχανικής μάθησης. Τέλος, παρουσιάζουμε εναλλακτικά μοντέλα για το χρόνο επικοινωνίας, τα οποία αντλούνται από την πρόσφατη σχετική βιβλιογραφία, τα οποία αργότερα χρησιμοποιούμε για σκοπούς σύγκρισης.

3.2 Κατασκευή μοντέλων με εποπτευόμενη μάθηση

Υποθέτουμε ότι υπάρχει ένα διάνυσμα των χαρακτηριστικών $X = (x_1, x_2, ..., x_k)$ και μία συνάρτηση f που καθορίζουν το χρόνο επικοινωνίας t μιας εφαρμογής, δηλαδή $t = f(X) + \varepsilon$. Ο τελικός στόχος της εργασίας μας είναι να συνθέσουμε μία προσέγγιση $\hat{f}$ της συνάρτησης f για την πρόβλεψη του χρόνου επικοινωνίας $\hat{t} = \hat{f}(X)$ για οποιαδήποτε γνωστή ή άγνωστη είσοδο X, με ένα σφάλμα $\varepsilon = t - \hat{t}$ που μπορούμε να ανεχθούμε. Η εργασία εμπίπτει στην κατηγορία της *εποπτευόμενης μάθησης* (supervised learning), όπου ένας αλγόριθμος μάθησης χρησιμοποιεί ένα σύνολο εκπαίδευσης $Tr = \{(X_i, t_i), i = 1, ..., n\}$ για να μάθει *μέσω παραδειγμάτων* την προσέγγιση $\hat{f}$ [Friedman et al., 2009]. Δεδομένου ότι η έξοδος t είναι συνεχής, η εργασία εμπίπτει επίσης στην κατηγορία της *παλινδρόμησης*.

Επιλογή χαρακτηριστικών

Για κάθε σύστημα που εξετάζουμε, εξετάζουμε επίσης μια πληθώρα χαρακτηριστικών που σχετίζονται με την εφαρμογή και την απεικόνισή της στο σύστημα. Ωστόσο, όλα αυτά τα χαρακτηριστικά δεν είναι απαραίτητα σημαντικά για την πρόβλεψη του χρόνου επικοινωνίας. Όταν επιδιώκεται μοντελοποίηση με μία μεταβλητή, πρέπει να προσδιοριστεί μόνο ένα σημαντικό χαρακτηριστικό, με μεγάλο βαθμό συσχέτισης (correlation) με την έξοδο. Στην περίπτωση της μοντελοποίησης με πολλαπλές μεταβλητές, η συστηματική επιλογή χαρακτηριστικών μπορεί να βοηθήσει στην κατανόηση των δεδομένων και να βελτιώσει την ακρίβεια πρόβλεψης [Chandrashekar and Sahin, 2014]. Στις τεχνικές για την επιλογή χαρακτηριστικών συμπεριλαμβάνεται το φιλτράρισμα με βάση τη σχέση (association filtering) και η αναδρομική απαλοιφή χαρακτηριστικών, με στόχο να εξαλείψουν πλεονάζοντα ή άσχετα χαρακτηριστικά, που θα προσέθεταν περιττές πληροφορίες ή θόρυβο στο μοντέλο. Το αποτέλεσμα της επιλογής χαρακτηριστικών είναι ένα διάνυσμα χαρακτηριστικών $X' = (x_1, x_2, ..., x_{k'}), k' < k$.

Μέθοδοι παλινδρόμησης

Για την ανάπτυξη μοντέλων παλινδρόμησης, υπάρχουν διάφορες μέθοδοι. Η απλούστερη μέθοδος είναι τα *γραμμικά μοντέλα* της μορφής $\hat{t}(X'') = b_0 + \sum_{i=1}^{k''} b_i x_i''$, όπου τα στοιχεία του διανύσματος X'' περιλαμβάνουν χαρακτηριστικά του αρχικού διανύσματος χαρακτηριστικών X, αλληλεπιδράσεις μεταξύ αυτών με τη μορφή γινομένων, πολυωνυμικούς ή μη γραμμικούς μετασχηματισμούς ή και ψευδομεταβλητές. Αυτή η μέθοδος απαιτεί τον καθορισμό της μορφής της προσεγγιστικής συνάρτησης από το χρήστη, ενώ οι συντελεστές του μοντέλου υπολογίζονται με βάση τη μέθοδο των ελαχίστων τετραγώνων. Ένα γραμμικό μοντέλο μπορεί όμως να είναι πολύ αποτελεσματικό στην πρόβλεψη, όμως μπορεί να είναι οσοδήποτε πολύπλοκο σε όρους και κλάδους, ειδικά στην περίπτωση των πολλών μεταβλητών. Επιπλέον, είναι ευθύνη του κατασκευαστή του μοντέλου να ταυτοποιήσει όλες τις υπάρχουσες σχέσεις μεταξύ των χαρακτηριστικών και της εξόδου (στην περίπτωσή μας, του χρόνου επικοινωνίας). Παρόλα αυτά, τα γραμμικά μοντέλα παρέχουν μία συνάρτηση κλειστού τύπου, όπου η σχέση μεταξύ ενός χαρακτηριστικού και του χρόνου επικοινωνίας είναι σαφής, ενώ άλλες, αυτοματοποιημένες μέθοδοι, όπως οι μέθοδοι συλλογικής μάθησης, παρέχουν μόνο μία κατάταξη των επιλεγμένων χαρακτηριστικών. Αναφερόμαστε στην παλινδρόμηση με το γραμμικό μοντέλο ως στατιστική μάθηση, δεδομένου ότι η μορφή του μοντέλου καθορίζεται από τον κατασκευαστή του μοντέλου, σε αντίθεση με τις μεθόδους μηχανικής μάθησης, όπου η μορφή μοντέλου προκύπτει από την ίδια τη μέθοδο. Ωστόσο, η στατιστική μάθηση είναι, στην πραγματικότητα, ένα υποσύνολο μηχανικής μάθησης.

Οι μέθοδοι *συλλογικής μάθησης* (ensemble methods) που βασίζονται σε δέντρα απόφασης είναι πιο πρόσφατες μέθοδοι που έχει αποδειχθεί πως φέρουν ικανοποιητικά αποτελέσματα σε προβλήματα παλινδρόμησης. Τα δέντρα αποφάσεων χρησιμοποιούνται από τις μεθόδους συλλογικής μάθηδης [Zhang and Ma, 2012] ως αδύναμοι εκπαιδευόμενοι (weak learners). Οι μέθοδοι εμφωλίασης (bagging methods), στις οποίες ανήκουν οι αλγόριθμοι Random Forests και Extremely Randomized Trees, χτίζουν ταυτόχρονα πολλά δέντρα αποφάσεων και στη συνέχεια χρησιμοποιούν το μέσο όρο τους, ώστε να μειώσουν τη διακύμανση (variance), δηλαδή το αναμενόμενο σφάλμα λόγω του συνόλου εκπαίδευσης που χρησιμοποιήθηκε. Οι μέθοδοι ενδυνάμωσης (boosting methods), στις οποίες ανήκουν αλγόριθμοι όπως ο AdaBoost και τα Gradient Boosting Machines, χτίζουν διαδοχικά αδύναμους εκτιμητές, ο καθένας εκ των οποίων επικεντρώνεται σε σημεία που ο προηγούμενος παρέλειψε, προκειμένου να μειωθεί η προκατάληψη (bias), δηλαδή το αναμενόμενο σφάλμα του εκτιμητή σε νέα δεδομένα. Το βασικό πλεονέκτημα των μεθόδων συλλογικής μάθησης είναι ότι παρέχουν εκτιμητές χωρίς να απαιτούν γνώση της ακριβούς

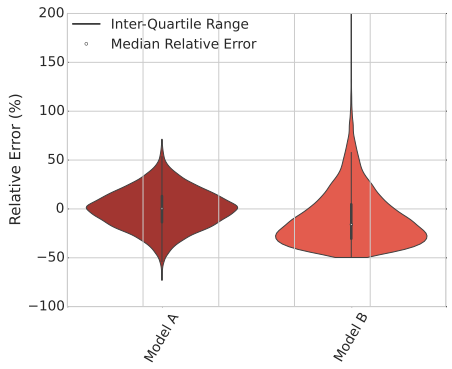
σχέσης μεταξύ των χαρακτηριστικών και του στόχου της πρόβλεψης. Ωστόσο, όλες οι μέθοδοι συλλογικής μάθησης έχουν ένα σημαντικό μειονέκτημα: αντίθετα με άλλες μεθόδους παλινδρόμησης, δεν είναι σε θέση να παρεκκλίνουν πέρα από το εύρος του συνόλου εκπαίδευσης. Για παράδειγμα, υποθέτουμε την κατασκευή ενός μοντέλου για μια μεταβλητή στόχου y από μια ενιαία μεταβλητή x και υποθέτουμε ότι η μεταβλητή y περιγράφεται από μία συνάρτηση ταυτότητας με τη μεταβλητή x, δηλαδή $\hat{y} = f(x) = x$. Επιπλέον, υποθέτουμε ότι εκπαιδεύουμε δύο μοντέλα για τη μεταβλητή y, ένα χρησιμοποιώντας γραμμική παλινδρόμηση και ένα χρησιμοποιώντας μια μέθοδο συλλογικής μάθησης, χρησιμοποιώντας τιμές x και y που κυμαίνονται από 0 έως 1. Αν προσπαθήσουμε να προβλέψουμε την τιμή της y για $x = 2$, το μοντέλο γραμμικής παλινδρόμησης θα προβλέψει $\hat{t} = 2$, ενώ το μοντέλο της συλλογικής μάθησης θα προβλέψει $\hat{t} = 1$, που είναι η υψηλότερη τιμή στο σύνολο εκπαίδευσης. Η αδυναμία των μεθόδων συλλογικής μάθησης να προεκτείνουν τις προβλέψεις τους έξω από το σύνολο εκπαίδευσης μπορεί να αποβεί ιδιαίτερα περιοριστική, οπότε το σύνολο εκπαίδευσης πρέπει να επιλεγεί προσεκτικά.

Μετρικές αποτίμησης

Για να αξιολογήσουμε ένα μοντέλο για το χρόνο επικοινωνίας, εστιάζουμε στη δυνατότητά του να προβλέπει τον χρόνο επικοινωνίας για οποιοδήποτε σχήμα επικοινωνίας και οποιεσδήποτε ρυθμίσεις εκτέλεσης με το ελάχιστο δυνατό σφάλμα, καθώς και την καλή προσαρμογή του, δηλαδή πόσο καλά οι προβλεπόμενες τιμές προσαρμόζονται στις πραγματικές τιμές. Για το σκοπό αυτό, εξετάζουμε διάφορες μετρικές αξιολόγησης. Αξίζει να σημειωθεί ότι καμία μετρική αξιολόγησης δεν μπορεί οριστικά να αποφανθεί για την ακρίβεια και την καλή προσαρμογή ενός μοντέλου και κάθε σημείο στο σύνολο αξιολόγησης θα πρέπει να αξιολογηθεί ξεχωριστά. Ωστόσο, διαφορετικές μετρικές μπορούν να καταδείξουν διαφορετικές ιδιότητες της ακρίβειας ενός μοντέλου ή / και της προσαρμογής του και όλες οι μετρικές σε συνδυασμό είναι ιδιαίτερα χρήσιμες για τη σύγκριση μοντέλων ή τη σύγκριση σε διαφορετικά σύνολα αξιολόγησης. Στη ροή εργασίας μας, ένα μοντέλο προβλέπει το χρόνο επικοινωνίας $\hat{t}_i, i = 1, ..., m$ για ένα σύνολο σημείων $T = \{(X_i, t_i), i = 1, ..., m\}$, που συλλέγονται με το πρόγραμμα συλλογής μετρήσεων αναφοράς ή από την εκτέλεση της εφαρμογής στο υποκείμενο σύστημα. Το σχετικό σφάλμα πρόβλεψης ορίζεται ως εξής:

$$e_i = (\hat{t}_i - t_i)/t_i$$

Εξετάζουμε την κατανομή των σχετικών σφαλμάτων πρόβλεψης, μαζί με την ελάχιστη, διάμεση και μέγιστη τιμή των σχετικών σφαλμάτων. Χρησιμοποιούμε τα διαγράμματα “βιολιου” (violinplots) [Hintze and Nelson, 1998] ως



Σχήμα 3.1: Υποδείγματα διαγραμμάτων "βιολιού" για τα σχετικά σφάλματα πρόβλεψης δύο υποθετικών μοντέλων

μέσο για την απεικόνιση της κατανομής των σχετικών σφαλμάτων και συγκρίνουμε την κατανομή των σχετικών σφαλμάτων μεταξύ διαφορετικών μοντέλων. Τα διαγράμματα βιολιού είναι μια ποιοτική εναλλακτική λύση στα ιστογράμματα. Ένα παράδειγμα διαγραμμάτων βιολιού για σφάλματα πρόβλεψης με δύο υποθετικά μοντέλα δίνεται στο Σχήμα 3.1. Τα σχετικά σφάλματα του μοντέλου Α ακολουθούν μια κανονική κατανομή και τα σχετικά σφάλματα του μοντέλου Β ακολουθούν μια κατανομή Gamma. Τα άκρα κάθε βιολιού αντιστοιχούν στο ελάχιστο και στο μέγιστο σχετικό σφάλμα. Τα διαγράμματα βιολιού στο παράδειγμά μας δείχνουν ότι το μοντέλο Α έχει μικρότερο εύρος σφαλμάτων από το μοντέλο Β. Το πλάτος (ή εύρος) του διαγράμματος βιολιού για ένα συγκεκριμένο επίπεδο του σχετικού σφάλματος υποδεικνύει πόσα σημεία στο εξεταζόμενο σύνολο παρουσιάζουν το συγκεκριμένο σφάλμα. Για παράδειγμα, για το μοντέλο Α, το διάγραμμα βιολιού είναι ευρύτερο για τιμές σχετικού σφάλματος κοντά στο 0% από ό,τι στο 50%, συνεπώς περισσότερα σημεία εμφανίζουν σφάλμα περίπου 0%. Για το μοντέλο Β, το εύρος του διαγράμματος βιολιού είναι ευρύτερο για τιμές σχετικού σφάλματος κοντά στο -40%. Έτσι περισσότερα σημεία παρουσιάζουν σχετικά σφάλματα περίπου ίσα με -40%. Επίσης, για το μοντέλο Β, το διάγραμμα βιολιού είναι λεπτό για σχετικά σφάλματα υψηλότερα από 100%, επομένως μόνο λίγα σημεία παρουσιάζουν τέτοια υψηλά σφάλματα. Επιπλέον, το διάγραμμα βιολιού υποδεικνύει τη διάμεσο των σχετικών σφαλμάτων και το εύρος μεταξύ τεταρτημορίων, δηλαδή τα σημεία με σχετικά σφάλματα μεταξύ του 1ου τεταρτημορίου και του 3ου τεταρτημορίου. Τα διαγράμματα βιολιού μας επιτρέπουν να συγκρίνουμε διαφορετικά μοντέλα. Στο παράδειγμά μας, το μοντέλο Α παρουσιάζει καλύτερη κατανομή των σχετικών σφαλμάτων από το μοντέλο Β: τα περισσότερα σφάλματα συγκεντρώνονται γύρω από το 0% και κατανέμονται κανονικά γύρω από το 0%, ενώ και το εύρος των σφαλμάτων είναι μικρότερο από αυτό

του μοντέλου B.

Λαμβάνουμε επίσης υπόψη τη μέση τιμή των σχετικών σφαλμάτων *MMRE* (Mean Magnitude of Relative Errors) [Conte et al., 1986], ως εναλλακτική της κατανομής των σχετικών σφαλμάτων:

$$MMRE = \frac{\sum_{i=1}^{m} |e_i|}{m}$$

Για να εκτιμήσουμε την ακρίβεια και την καλή προσαρμογή ενός μοντέλου πρόβλεψης, εξετάζουμε το ποσοστό των προβλέψεων στο επίπεδο 0,25, $Pred_{0,25}$:

$$Pred_{0.25}(\%) = \frac{\#predictions\ with\ |e_i| \leq 0.25}{\#predictions} * 100\%$$

Η μετρική $Pred_{0.25}$ παίρνει τιμές μεταξύ 0 και 100%, με τις υψηλότερες τιμές να υποδηλώνουν υψηλότερη ακρίβεια. Η μετρικής $Pred_{0.25}$ μετρά την ακρίβεια της πρόβλεψης σε ένα σταθερό κατώφλι για το σχετικό σφάλμα (στη δική μας περίπτωση, 0.25 ή 25%). Μια υψηλή τιμή της μετρικής $Pred_{0.25}$ υποδεικνύει ότι η ακρίβεια του μοντέλου είναι υψηλή για το συγκεκριμένο σύνολο σημείων: τα περισσότερα από τα σχετικά σφάλματα βρίσκονται το πολύ σε ένα εύρος σχετικών σφαλμάτων $\pm$ 25% και έτσι συγκεντρώνονται γύρω από το 0%.

Εξετάζουμε επίσης τον συντελεστή προσδιορισμού (coefficient of determination) R^2:

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(t_i - \hat{t}_i)^2}{\sum_{i=1}^{m}(t_i - \bar{t})^2}$$

και ο συντελεστής συσχέτισης βαθμών (rank correlation coefficient) *RCC*:

$$RCC = \sum_{1 \leq i \leq m} \sum_{1 \leq j \leq m} concordant_{ij} / \frac{m(m-1)}{2}$$

όπου:

$$concordant_{ij} = \begin{cases} 1, & if\ t_i < t_j\ and\ \hat{t}_i < \hat{t}_j \\ 1, & if\ t_i > t_j\ and\ \hat{t}_i > \hat{t}_j \\ 0, & otherwise \end{cases}$$

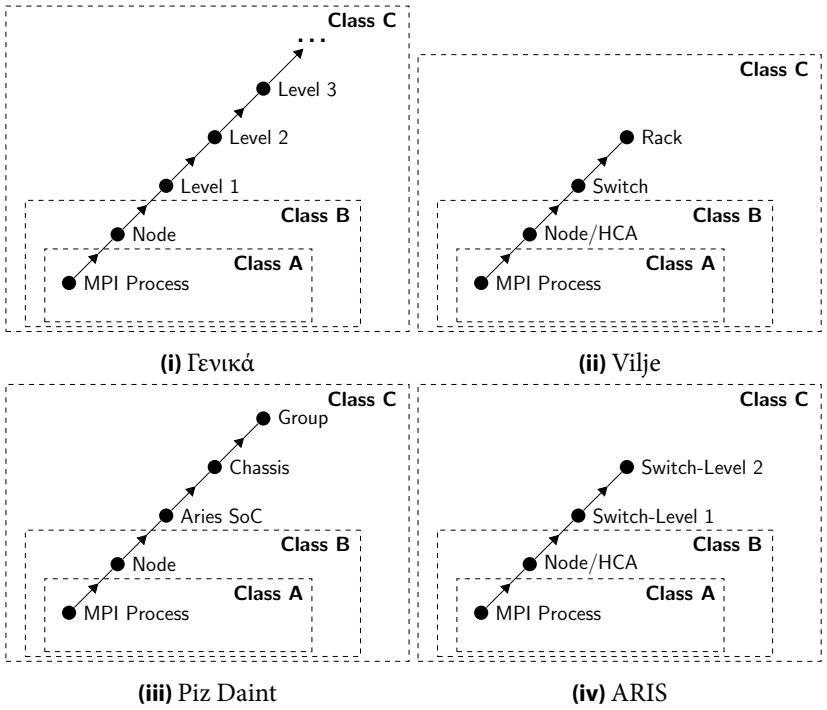
Η υψηλότερη τιμή του R^2, που υποδηλώνει βέλτιστη προσαρμογή, είναι 1, αλλά η μετρική μπορεί επίσης να λάβει αρνητικές τιμές, όταν τα δεδομένα στο σύνολο ελέγχου δεν έχουν χρησιμοποιηθεί στη διαδικασία εκπαίδευσης του μοντέλου. Η μετρική *RCC* κυμαίνεται από 0 έως 1, με τις υψηλότερες τιμές να δηλώνουν καλύτερη προσαρμογή. Η μετρική R^2 μετρά πόσο καλά το

μοντέλο προσεγγίζει τις πραγματικές τιμές, ενώ η μετρική *RCC* μετρά πόσο καλά προβλέπεται η διάταξη των δεδομένων. Μια υψηλή τιμή της R^2 υποδηλώνει ότι οι προβλεπόμενες τιμές ταιριάζουν καλά με τα πραγματικά δεδομένα και, ως εκ τούτου, είναι μια καλή μετρική τόσο για την ακρίβεια όσο και για την καλή προσαρμογή. Μια υψηλή τιμή της *RCC* υποδηλώνει ότι το μοντέλο είναι σε θέση να διακρίνει μεταξύ διαφορετικών ρυθμίσεων του χρόνου εκτέλεσης. Εάν μία ρύθμιση χρόνου εκτέλεσης οδηγεί σε υψηλότερο χρόνο επικοινωνίας από μία άλλη, ένα μοντέλο με υψηλή τιμή *RCC* προβλέπει επίσης υψηλότερο χρόνο επικοινωνίας για την πρώτη ρύθμιση χρόνου εκτέλεσης. Θα πρέπει να διευκρινίσουμε τη διαφορά μεταξύ R^2 και *RCC*: η πρώτη αξιολογεί τόσο την ακρίβεια του μοντέλου όσο και την προσαρμογή του, ενώ η τελευταία αξιολογεί μόνο την προσαρμογή. Για παράδειγμα, αν ένα μοντέλο προβλέπει την ακριβή διάταξη των δεδομένων, με ένα σταθερό σφάλμα 20%, η τιμή για τη μετρική *RCC* θα είναι 1, αλλά η τιμή για τη μετρική R^2 θα είναι χαμηλότερη, ώστε να αντικατοπτρίζει την απόκλιση κατά 20%.

3.3 Χαρακτηριστικά για την πρόβλεψη της επικοινωνίας

3.3.1 Χαρακτηριστικά για την επίδοση της point-to-point επικοινωνίας

Ως πρώτο βήμα στην κατεύθυνση της μοντελοποίησης του χρόνου επικοινωνίας, ορίζουμε μετρήσιμα χαρακτηριστικά για το προφίλ επικοινωνίας της εφαρμογής, το σχήμα κίνησης δεδομένων (traffic pattern) και το σχήμα της κατανομής των πόρων (allocation shape), για μία δεδομένη εκτέλεση της εφαρμογής με συγκεκριμένες παραμέτρους στο σύστημα υπό εξέταση. Διαχωρίζουμε τα χαρακτηριστικά σε τρεις κατηγορίες, ανάλογα με το αν παρουσιάζουν ή όχι επίγνωση του προφίλ επικοινωνίας της εφαρμογής, της απεικόνισής της στο σύστημα και της αρχιτεκτονικής του συστήματος. Ο διαχωρισμός αυτός επιτρέπει την κατασκευή μοντέλων με τις διαφορετικές κατηγορίες χαρακτηριστικών, που παρουσιάζουν και αντίστοιχη εφαρμοσιμότητα και ακρίβεια πρόβλεψης. Τα χαρακτηριστικά της κατηγορίας Α συγκεντρώνονται από το προφίλ επικοινωνίας της εφαρμογής και τις παραμέτρους της εκτέλεσης (μέγεθος προβλήματος, αριθμός κόμβων και διεργασιών) πριν από την εκτέλεση της εφαρμογής, ο χρήστης αποφασίζει τον αριθμό κόμβων και διεργασιών ανά κόμβο που απαιτεί η εκτέλεση της εφαρμογής του. Για ένα δεδομένο μέγεθος προβλημάτων, είναι γνωστά τα μεγέθη και τα πλήθη των μηνυμάτων και η κίνηση δεδομένων ανά διεργασία. Οι τιμές αυτών των χαρακτηριστικών μπορούν να εξαχθούν σε στατικό χρόνο με ελάχιστη προσπάθεια. Έτσι, οποιοδήποτε μοντέλο βασίζεται στα χαρακτηριστικά κατηγορίας Α έχει πολλές δυνατότητες εφαρμογής, π.χ. μπορεί να υποστηρίξει τη λήψη αποφάσεων σχετικά με το σχεδιασμό εφαρμογών και αλγοριθμικές βελτιστοποιήσεις. Η κατηγο-



Σχήμα 3.2: Κατηγορίες χαρακτηριστικών i) σε οποιοδήποτε σύστημα, ii) στο Vilje, iii) στο Piz Daint και iv) στο Aris.

ρία Β ενισχύει την κατηγορία Α με χαρακτηριστικά που σχετίζονται με την απεικόνιση των διεργασιών σε μια δοσμένη κατανομή κόμβων με συγκεκριμένες παραμέτρους εκτέλεσης. Η απεικόνιση ορίζεται είτε από τον χρήστη κατά την υποβολή της εφαρμογής στο σύστημα, είτε από το λογισμικό διαχείρισης πόρων. Τα χαρακτηριστικά της κατηγορίας Β μετρούν το σχήμα κίνησης δεδομένων "τοπικά", στο επίπεδο του κόμβου, αλλά χωρίς επίγνωση της θέσης των κόμβων στο σύστημα και της αρχιτεκτονικής του δικτύου. Τέλος, τα χαρακτηριστικά της κατηγορίας Γ σκιαγραφούν το σχήμα κίνησης δεδομένων, όπως προκύπτει από την εφαρμογή, καθώς και το σχήμα της κατανομής των πόρων. Αυτά τα χαρακτηριστικά μπορούν να εξαχθούν μόνο κατά το χρόνο εκτέλεσης, αφού ο χρονοδρομολογητής του συστήματος επιλέξει την κατανομή των κόμβων, το αργότερο πριν την εκκίνηση της εκτέλεσης της εφαρμογής (just-in-time). Επομένως, οποιοδήποτε μοντέλο κατασκευάζεται με αυτά τα χαρακτηριστικά εξακολουθεί να έχει τη δυνατότητα πρόβλεψης, δηλαδή τη δυνατότητα να παράσχει μια εκτίμηση του χρόνου επικοινωνίας πριν την εκτέλεση της εφαρμογής, ενώ ταυτόχρονα έχει σε μεγάλο βαθμό επίγνωση των περισσότερων παραμέτρων που επηρεάζουν την επίδοση της επικοινωνίας.

Πίνακας 3.1: Χαρακτηριστικά της Κατηγορίας Α για οποιοδήποτε σύστημα

Χαρακτηριστικό	**Όνομα**	**Περιγραφή**
n	Nodes	Το πλήθος των κόμβων στην κατανομή πόρων
ppn	Processes Per Node	Ο αριθμός διεργασιών MPI ανά κόμβο στην κατανομή των πόρων
l	Message Length	Το μήκος (σε bytes) των μηνυμάτων που στέλνει κάθε διεργασία [min,avg,max]
PD	Process Data	Τα δεδομένα (σε bytes) που στέλνει κάθε διεργασία [min,avg,max]
PM	Process Messages	Το πλήθος των μηνυμάτων που στέλνει κάθε διεργασία [min,avg,max]

Η πλειονότητα των χαρακτηριστικών που ορίζουμε αφορούν την κίνηση (σε bytes) και το πλήθος των μηνυμάτων που εγχέονται από τις διεργασίες στους κόμβους και στα ανώτερα επίπεδα του δικτύου διασύνδεσης. Το Σχήμα 3.2 εικονίζει με αφαιρετικό τρόπο τις κατηγορίες των χαρακτηριστικών και πώς προσδιορίζονται στα συστήματα που εξετάζουμε, αναδεικνύοντας διαφορετικά επίπεδα επίγνωσης της τοπολογίας και της οργάνωσης του δικτύου. Όλα τα χαρακτηριστικά που σχετίζονται με δεδομένα και μηνύματα σκιαγραφούν την εξερχόμενη κίνηση σε bytes και πλήθος μηνυμάτων από τις διεργασίες της εφαρμογής προς τα δομικά στοιχεία του συστήματος (κόμβους, διακόπτες, Aries SoCs κλπ) όπως αυτά χρησιμοποιούνται στην κατανομή των πόρων για μια συγκεκριμένη εκτέλεση. Τα χαρακτηριστικά της κατηγορίας Α παρουσιάζονται στον Πίνακα 3.1. Περιορίζονται στην περιγραφή γνώσης που αφορά την εφαρμογή, δηλαδή το προφίλ επικοινωνίας της εφαρμογής (μέγεθος μηνύματος, δεδομένα και μηνύματα διεργασίας) και το μέγεθος της κατανομής, τον αριθμό κόμβων και διεργασιών ανά κόμβο, που δίνονται από το χρήστη. Τα χαρακτηριστικά της κατηγορίας Β, που περιγράφονται στον Πίνακα 3.2, έχουν επίγνωση της απεικόνισης των διεργασιών της εφαρμογής στους πυρήνες και κόμβους που δεσμεύονται από το σύστημα για μία εκτέλεση, επομένως συμπεριλαμβάνουν και δεδομένα και μηνύματα εντός του κόμβου, που μπορεί να ασκούν πίεση στο σύστημα μνήμης, δεδομένα και μηνύματα που ανταλλάσσονται μεταξύ κόμβων, που μπορούν να προκαλέσουν ανταγωνισμό λόγω περιορισμένου εύρους έγχυσης ή περιορισμένης χωρητικότητας των καρτών

Πίνακας 3.2: Χαρακτηριστικά της Κατηγορίας Β για οποιοδήποτε σύστημα

Χαρακτηριστικό	**Όνομα**	**Περιγραφή**
ND	Node Data	Τα δεδομένα (σε bytes) που εγχύονται από κάθε κόμβο στο δίκτυο διασύνδεσης [min, avg, max]
NM	Node Messages	Τα μηνύματα που εγχύονται από κάθε κόμβο στο δίκτυο διασύνδεσης [min, avg, max]
iND	Intranode Data	Τα δεδομένα (σε bytes) που ανταλλάσσονται εντός κάθε κόμβου [min, avg, max]
iNM	Intranode Messages	Τα μηνύματα που ανταλλάσσονται εντός κάθε κόμβου [min, avg, max]
TD	Total Data	Τα δεδομένα (σε bytes) που εγχύονται από όλους τους κόμβους της κατανομής
TM	Total Messages	Τα μηνύματα που εγχύονται από όλους του κόμβους της κατανομής

του δικτύου [Bhatele et al., 2015], καθώς και τα συνολικά δεδομένα και μηνύματα, που αφορούν στο συνολικό όγκο επικοινωνίας της εφαρμογής μεταξύ κόμβων. Τα χαρακτηριστικά των κατηγοριών Α και Β είναι ανεξάρτητα του συστήματος (cross-platform).

Τα χαρακτηριστικά της κατηγορίας Γ για το Vilje παρουσιάζονται στον Πίνακα 3.3. Εκτός από τα δεδομένα και τα μηνύματα στους διακόπτες (switches) και στα racks, που μπορούν να αποκαλύψουν φαινόμενα συμφόρησης σε ενδιάμεσους διακόπτες, εξετάζουμε επίσης τον αριθμό των διακοπτών, των racks, το μέσο αριθμό κόμβων ανά διακόπτη και το μέσο αριθμό διακοπτών ανά rack, ως περιγραφικά χαρακτηριστικά του σχήματος της κατανομής των κόμβων, το οποίο είναι συνήθως ακανόνιστο στο Vilje. Ομοίως, ορίζουμε τα χαρακτηριστικά κατηγορίας Γ για το Piz Daint στον Πίνακα 3.4. Οι διακόπτες και τα racks του Vilje αντικαθίστανται από τα Aries SoCs, τα chassis και τις ομάδες (groups) του Piz Daint. Καθώς στο Piz Daint η δρομολόγηση είναι προσαρμοστική και τα πακέτα επιλέγουν μια ελάχιστη ή μη ελάχιστη διαδρομή, ανάλογα με την τρέχουσα συμφόρηση του συνδέσμου, ορίζουμε ως επιπλέον χαρακτηριστικά τα δεδομένα και τα μηνύματα από ομάδα σε ομάδα, ως ενδεικτικά υπερβολικής

Πίνακας 3.3: Χαρακτηριστικά της Κατηγορίας Γ για το σύστημα Vilje

Χαρακτηριστικό	Όνομα	Περιγραφή
sw	Switches	Το πλήθος των διακοπτών (switches) στους οποίους συνδέονται οι κόμβοι της κατανομής
r	Racks	Το πλήθος των racks στα οποία βρίσκονται οι κόμβοι της κατανομής
n/sw	Nodes per Switch	Κόμβοι ανά διακόπτη στην κατανομή [avg]
sw/r	Switches per Rack	Διακόπτες ανά rack στην κατανομή [avg]
SD	Switch Data	Τα δεδομένα (σε bytes) που εγχύονται από έναν διακόπτη στο δίκτυο διασύνδεσης [min, avg, max]
SM	Switch Messages	Τα μηνύματα που εγχύονται από έναν διακόπτη στο δίκτυο διασύνδεσης [min, avg, max]
iSD	Intra-Switch Data	Τα δεδομένα (σε bytes) που στέλνονται μεταξύ κόμβων συνδεδεμένων στον ίδιο διακόπτη [min, avg, max]
iSM	Intra-Switch Messages	Τα μηνύματα που στέλνονται μεταξύ κόμβων συνδεδεμένων στον ίδιο διακόπτη [min, avg, max]
RD	Rack Data	Τα δεδομένα (σε bytes) που στέλνονται από ένα rack [min, avg, max]
RM	Rack Messages	Τα μηνύματα που στέλνονται από ένα rack [min, avg, max]
iRD	Intra-Rack Data	Τα δεδομένα (σε bytes) που στέλνονται μεταξύ διακοπτών που βρίσκονται στο ίδιο rack [min, avg, max]
iRM	Intra-Rack Messages	Τα μηνύματα που στέλνονται μεταξύ διακοπτών που βρίσκονται στο ίδιο rack [min, avg, max]

κίνησης μεταξύ ομάδων που θα μπορούσε να ασκήσει πίεση στην οπτική σύνδεση μεταξύ δύο ομάδων και να προκαλέσει μη-ελάχιστη δρομολόγηση. Στον Πίνακα 3.5, παρουσιάζουμε τα χαρακτηριστικά της κατηγορίας Γ στο ARIS. Ας σημειωθεί ότι το ARIS έχει τοπολογία fat tree δύο επιπέδων, όπου όλοι οι διακόπτες στο υψηλότερο επίπεδο (Level 2) συνδέονται με όλους τους διακόπτες στο χαμηλότερο επίπεδο (Level 1). Επομένως, ορίζουμε την κίνηση μόνο στους διακόπτες του επιπέδου 1 (Level 1-switches), καθώς δεν υπάρχει τρόπος να γνωρίζουμε ποιοι ή πόσοι από τους διακόπτες του επιπέδου 2 χρησιμοποιούνται από οποιαδήποτε κατανομή κόμβων. Επιπλέον, για να σκιαγραφήσουμε το σχήμα της κατανομής των κόμβων, χρησιμοποιούμε τον αριθμό των διακοπτών επιπέδου 1, τον αριθμό των κόμβων ανά διακόπτες επιπέδου 1 και τον αριθμό των διακοπτών επιπέδου 1 ανά διακόπτη επιπέδου 2. Πρέπει να σημειώσουμε ότι οι τιμές για χαρακτηριστικά όπως *ppn*, *n*, *sw* ή *g*, που αναφέρονται στην κατανομή των κόμβων, είναι σταθερές για την εκτέλεση μιας εφαρμογής, ενώ οι τιμές για όλα τα άλλα χαρακτηριστικά είναι σταθερές για μια φάση επικοινωνίας.

3.3.2 Εξαγωγή τιμών χαρακτηριστικών

Τα χαρακτηριστικά που ορίζουμε αντικατοπτρίζουν τις ροές δεδομένων που προκύπτουν από την απεικόνιση του γράφου επικοινωνίας της εφαρμογής

Πίνακας 3.4: Χαρακτηριστικά της Κατηγορίας Γ για το σύστημα Piz Daint

Χαρακτηριστικό	Όνομα	Περιγραφή
a	Aries SoCs	Το πλήθος των Aries SoCs που καταλαμβάνουν οι κόμβοι της κατανομής
c	Chassis	Το πλήθος των chassis που καταλαμβάνουν οι κόμβοι της κατανομής
g	Groups	Το πλήθος των ομάδων που καταλαμβάνουν οι κόμβοι της κατανομής
a/c	Aries SoCs per Chassis	Το πλήθος των Aries SoCs ανά chassis [avg,max]
c/g	Chassis per Group	Το πλήθος των chassis ανά ομάδα [avg,max]
AD	Aries Data	Τα δεδομένα (σε bytes) που εγχύονται από ένα Aries SoC στο δίκτυο διασύνδεσης [min, avg, max]
AM	Aries Messages	Τα μηνύματα που εγχύονται από ένα Aries SoC στο δίκτυο διασύνδεσης [min, avg, max]
iAD	Intra-Aries Data	Τα δεδομένα (σε bytes) που στέλνονται μεταξύ κόμβων συνδεδεμένων στο ίδιο Aries SoC [min, avg, max]
iAM	Intra-Aries Messages	Τα μηνύματα που στέλνονται μεταξύ κόμβων συνδεδεμένων στο ίδιο Aries SoC [min, avg, max]
CD	Chassis Data	Τα δεδομένα (σε bytes) που στέλνονται από ένα πλαίσιο [min, avg, max]
CM	Chassis Messages	Τα μηνύματα που στέλνονται από ένα πλαίσιο [min, avg, max]
iCD	Intra-Chassis Data	Τα δεδομένα (σε bytes) που στέλνονται μεταξύ Aries SoCs που βρίσκονται στο ίδιο πλαίσιο [min, avg, max]
iCM	Intra-Chassis Messages	Τα μηνύματα που στέλνονται μεταξύ Aries SoCs που βρίσκονται στο ίδιο πλαίσιο [min, avg, max]
GD	Group Data	Τα δεδομένα (σε bytes) που στέλνονται από μία ομάδα [min, avg, max]
GM	Group Messages	Τα μηνύματα που στέλνονται από μία ομάδα [min, avg, max]
iGD	Intra-Group Data	Τα δεδομένα (σε bytes) που στέλνονται μεταξύ πλαισίων της ίδιας ομάδας [min, avg, max]
iGM	Intra-Group Messages	Τα μηνύματα που στέλνονται μεταξύ πλαισίων της ίδιας ομάδας [min, avg, max]
GGD	Group to Group Data	Τα δεδομένα (σε bytes) που στέλνονται μεταξύ δύο ομάδων [max]
GGM	Group to Group Messages	Τα μηνύματα που στέλνονται μεταξύ δύο ομάδων [max]

στο γράφο του δικτύου, για μια συγκεκριμένη κατανομή πόρων. Τα χαρακτηριστικά των διαφορετικών κατηγοριών απαιτούν διαφορετικές πληροφορίες, που μπορούν να εξαχθούν σε διαφορετικές χρονικές στιγμές στον κύκλο ζωής μιας εφαρμογής. Τα χαρακτηριστικά της κατηγορίας Α απαιτούν γνώση μόνο του σχήματος επικοινωνίας της εφαρμογής, στη μορφή ίχνους (trace) χωρίς χρονικές εξαρτήσεις (time-independent trace), δηλαδή πλειάδες της μορφής *{Αποστολέας, Παραλήπτης, Μέγεθος Μηνύματος}*. Η εξαγωγή αυτής της πληροφορίας μπορεί να γίνει αυτόματα με εργαλεία ιχνογράφησης (tracing tools), όπως είναι το mpiP [Vetter and Chambreau, 2005] ή το TAU [Shende and Malony, 2006] και το *tau2simgrid* [Desprez et al., 2011]. Αν ο κώδικας της εφαρμογής είναι διαθέσιμος, η εξαγωγή αυτής της πληροφορίας μπορεί να γίνει με

Πίνακας 3.5: Χαρακτηριστικά της Κατηγορίας Γ για το σύστημα ARIS

Χαρακτηριστικό	Όνομα	Περιγραφή
sw1	Level 1 Switches	Το πλήθος των διακοπτών στο χαμηλότερο επίπεδο του fat tree που καταλαμβάνουν οι κόμβοι της κατανομής
n/sw1	Nodes per Level 1-Switch	Το πλήθος των κόμβων ανά διακόπτη στο χαμηλότερο επίπεδο του fat tree [min, avg, max]
sw1/sw2	Level 1-Switches per Level 2-Switch	Το πλήθος των διακοπτών χαμηλότερου επιπέδου ανά διακόπτη υψηλότερου επιπέδου του fat tree [min, avg, max]
S1D	Level 1-Switch Data	Τα δεδομένα (σε bytes) που εγχύονται από έναν διακόπτη χαμηλότερου επιπέδου του fat tree στο δίκτυο [min, avg, max]
S1M	Level 1-Switch Messages	Τα μηνύματα που εγχύονται από έναν διακόπτη χαμηλότερου επιπέδου του fat tree στο δίκτυο [min, avg, max]
iS1D	Intra-Level 1-Switch Data	Τα δεδομένα (σε bytes) που στέλνονται μεταξύ κόμβων που βρίσκονται στον ίδιο διακόπτη χαμηλότερου επιπέδου του fat tree [min, avg, max]
iS1M	Intra-Level 1-Switch Messages	Τα μηνύματα (σε bytes) που στέλνονται μεταξύ κόμβων που βρίσκονται στον ίδιο διακόπτη χαμηλότερου επιπέδου του fat tree [min, avg, max]

απλή επισκόπηση του κώδικα. Καθώς χρειαζόμαστε ίχνη χωρίς χρονικές εξαρτήσεις, δηλαδή δεν απαιτούμε χρονικές σημάνσεις των γεγονότων της επικοινωνίας, η συλλογή των τιμών των χαρακτηριστικών της κατηγορίας Α μπορεί να πραγματοποιηθεί με εκτέλεση και ιχνογράφηση της εφαρμογής σε πολύ μικρότερο αριθμό πυρήνων από αυτόν στον οποίο τελικά επιθυμούμε να προβλέψουμε το χρόνο επικοινωνίας, για παράδειγμα σε έναν μόνο ή λιγοστούς κόμβους, αντί χιλιάδων. Τα χαρακτηριστικά της κατηγορίας Β απαιτούν γνώση της απεικόνισης των διεργασιών στους κόμβους, δηλαδή την αντιστοιχία διεργασίας και κόμβου, σε πλειάδες της μορφής *{rank, node}*. Η απεικόνιση είναι συνήθως γνωστή στο χρήστη για μία εκτέλεση μιας εφαρμογής, αφού η συνήθης πρακτική που ακολουθούν οι χρήστες υπερυπολογιστικών συστημάτων είναι να επιλέγουν ένα από τα προεπιλεγμένα σχήματα τοποθέτησης των διεργασιών στους κόμβους, όπως συνεχόμενη ή κυκλική ανάθεση διεργασιών, ή να ορίζουν το σχήμα τοποθέτησης των διεργασιών, ως παράμετρο στο script που εκκινεί την εκτέλεση μιας εφαρμογής MPI (π.χ. mpirun) ή ως μεταβλητή περιβάλλοντος στα scripts για την υποβολή εργασιών στο υπερυπολογιστικό σύστημα, στον Torque ή στο SLURM. Τα χαρακτηριστικά της κατηγορίας Γ απαιτούν τη λίστα των κόμβων για μια συγκεκριμένη κατανομή όπου θα εκτελεστεί η εφαρμογή, η οποία παρέχεται από το χρονοδρομολογητή/λογισμικό διαχείρισης πόρων, όπως ο Torque ή ο SLURM, αμέσως πριν την εκτέλεση της εφαρ-

μογής. Απαιτούν, επιπρόσθετα, πληροφορία για την τοπολογία του εκάστοτε συστήματος, που, στην περίπτωση των δικτύων InfiniBand, είναι δυνατό να εξαχθεί με την ανάλυση της εξόδου του εργαλείου ibstat[1] και στην περίπτωση των συστημάτων Cray, με ανάλυση της εξόδου του εργαλείου *xtnodestat* [2]. Και στις δύο περιπτώσεις, αυτά τα εργαλεία αποδίδουν τη θέση οποιουδήποτε κόμβου στην τοπολογία του συστήματος και επιτρέπουν την εξαγωγή των χαρακτηριστικών της κατηγορίας Γ. Αξίζει να σημειωθεί ότι το Select Plugin του λογισμικού SLURM λαμβάνει αποφάσεις για τη δέσμευση κόμβων αμέσως με την υποβολή μιας εφαρμογής στο χρονοδρομολογητή, επομένως η πληροφορία που απαιτούν τα χαρακτηριστικά της κατηγορίας Γ γίνεται διαθέσιμη πολύ νωρίτερα από την εκτέλεση της εφαρμογής και όχι αμέσως πριν από αυτή.

3.4 Συλλογή μετρήσεων αναφοράς

Η συλλογή μετρήσεων αναφοράς βρίσκεται στον πυρήνα της μεθοδολογίας μας, καθώς μάς επιτρέπει να διερευνήσουμε τυχόν σχέσεις μεταξύ των χαρακτηριστικών που έχουμε ορίσει και του χρόνου επικοινωνίας, να παρατηρήσουμε αξιόλογα φαινόμενα επίδοσης και τελικά, να κατασκευάσουμε μοντέλα πρόβλεψης της επικοινωνίας για το σύστημα υπό εξέταση. Η συλλογή μετρήσεων αναφοράς μάς εφοδιάζει με ένα περιεκτικό σύνολο δεδομένων, ικανό να εκπαιδεύσει μοντέλα πρόβλεψης. Για το σκοπό αυτό, κατασκευάσαμε ένα πρόγραμμα μετρήσεων αναφοράς (benchmark) που προσομοιάζει μια φάση επικοινωνίας μιας εφαρμογής μεγάλης κλίμακας και είναι παραμετρικό σε πολλά χαρακτηριστικά, επιτρέποντάς μας να σαρώνουμε έναν ευρύ χώρο των παραμέτρων επικοινωνίας και σχημάτων κίνησης δεδομένων. Αξίζει να σημειώσουμε ότι η διαδικασία συλλογής μετρήσεων αναφοράς στη μεθοδολογία μας χρειάζεται να γίνει μόνο μία φορά (για παράδειγμα, αμέσως μετά την εγκατάσταση ενός συστήματος μεγάλης κλίμακας) και επομένως, η πολυπλοκότητα της συλλογής μετρήσεων αναφοράς και ο χρόνος που απαιτείται για τη συλλογή τους δεν επηρεάζουν το χρόνο πρόβλεψης. Επιπλέον, η διαδικασία συλλογής μετρήσεων αναφοράς είναι γενική και χρησιμεύει για την πρόβλεψη χρόνου επικοινωνίας οποιασδήποτε φάσης point-to-point επικοινωνίας οποιασδήποτε εφαρμογής.

Το benchmark μας έχει βασιστεί στο WICON [Bhatelé and Kalé, 2009], όπου τυχαία ζεύγη διεργασιών, με μία μόνο διεργασία ανά κόμβο, επικοινωνούν ταυτόχρονα με point-to-point τρόπο. Ο στόχος του WICON είναι να ποσοτικοποιήσει τις καθυστερήσεις των μηνυμάτων υπό την παρουσία φαινομένων ανταγωνισμού. Για να ενσωματώσουμε στις μετρήσεις μας φαινόμενα

[1] https://linux.die.net/man/8/ibstat

[2] http://pubs.cray.com/#/Collaborate/00256453-FA

ανταγωνισμού και συμφόρησης που εμφανίζονται εξαιτίας της τοποθέτησης πολλών διεργασιών ανά κόμβο, ως πρώτο βήμα, τροποποιήσαμε το WICON ώστε πολλές διεργασίες που απεικονίζονται στον ίδιο κόμβο να επικοινωνούν ταυτόχρονα με διεργασίες σε έναν άλλο κόμβο, με τυχαία ζεύγη κόμβων. Για να σπάσουμε τη συμμετρία αυτού του σχήματος, μεταβάλλαμε τα τυχαία ζεύγη κόμβων σε τυχαία ζεύγη διεργασιών MPI, συνθέτοντας ένα τελείως τυχαίο σχήμα επικοινωνίας. Επιπλέον, για να αξιολογήσουμε την επίδραση του πλήθους των μηνυμάτων που αποστέλλει κάθε διεργασία, δημιουργήσαμε M τυχαία ζεύγη για κάθε διεργασία, καταλήγοντας σε ένα σχήμα επικοινωνίας όπου κάθε διεργασία στέλνει M μηνύματα ίδιου μεγέθους σε M τυχαίες διεργασίες και λαμβάνει απάντηση. Αλλάξαμε, επίσης, τις συναρτήσεις σύγχρονης επικοινωνίας σε συναρτήσεις ασύγχρονης επικοινωνίας, καθώς οι τελευταίες επιτρέπουν κάποια επικάλυψη, σε επίπεδο συστήματος, μεταξύ διαδοχικών αποστολών μηνυμάτων και αποτελεί τη συνήθη πρακτική για τις περισσότερες εφαρμογές MPI. Ο ψευδοκώδικας για τον πυρήνα των λειτουργιών του benchmark παρατίθεται στον Αλγόριθμο 3.1.

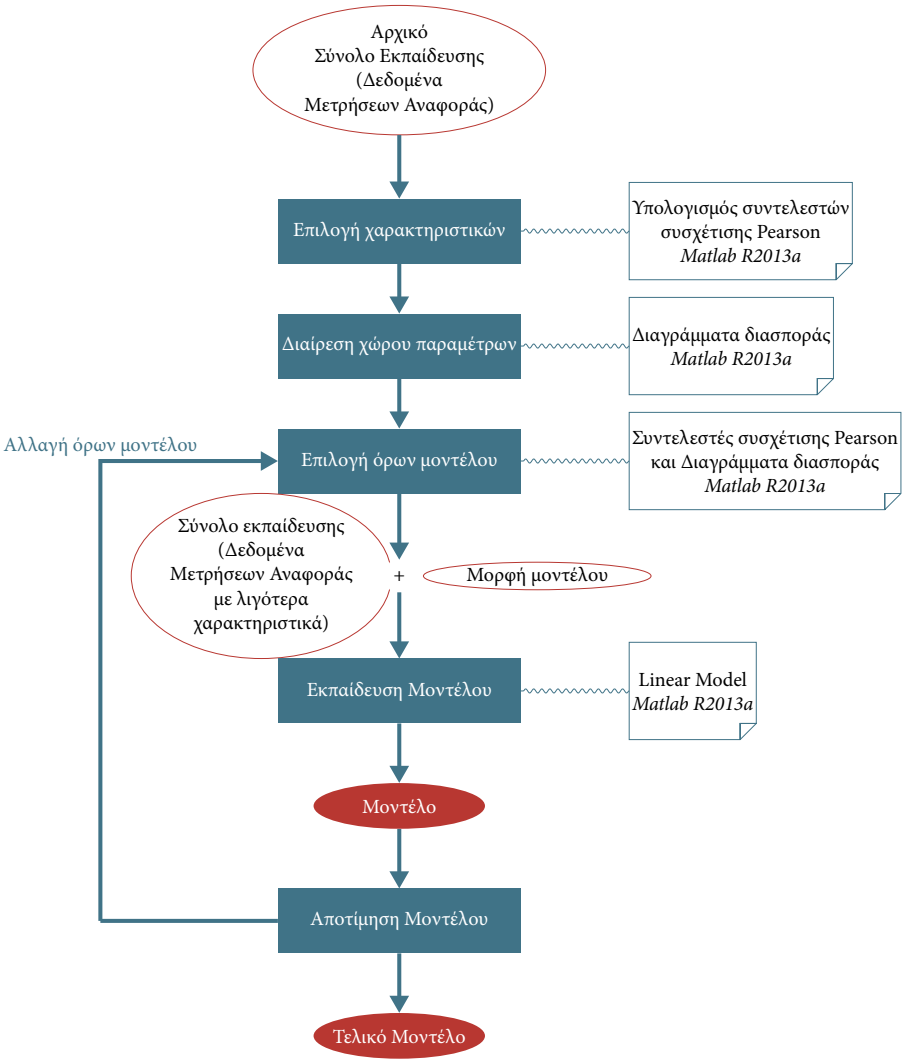
```
1:  for l = 1 to max_length do
2:      for iter = 1 to Iterations do
3:          MPI_Barrier(MPI_COMM_WORLD)
4:          gettimeofday(start_time)
5:          for m = 1 to ProcessMessages do
6:              MPI_Irecv(pong[m], l, MPI_UNSIGNED_CHAR, pair[m],...)
7:              MPI_Isend(ping[m], l, MPI_UNSIGNED_CHAR, pair[m],...)
8:          end for
9:          MPI_Waitall(2*m, ...)
10:         gettimeofday(stop_time)
11:         time[iter]=stop_time - start_time
12:     end for
13:     sort(time)
14:     local_reported_time=time[3*Iterations/4]
15:     MPI_Reduce(local_reported_time,global_reported_time,MPI_MAX...)
16: end for
```

Αλγόριθμος 3.1: Ψευδοκώδικας για τις βασικές λειτουργίες του benchmark και το χρονισμό

Το benchmark μας μάς επιτρέπει να σαρώσουμε το χώρο των παραμέτρων ρητά για τέσσερα από τα χαρακτηριστικά της κατηγορίας Α: το πλήθος των κόμβων (n), το πλήθος των διεργασιών ανά κόμβο (ppn), το μήκος του μηνύματος (l) και τα μηνύματα ανά διεργασία (PM). Έμμεσα, σαρώνοντας αυτά τα

τέσσερα χαρακτηριστικά, σαρώνουμε επιπλέον ένα μεγάλο χώρο τιμών για τα χαρακτηριστικά της κατηγορίας Α και Β. Οι τιμές για τα χαρακτηριστικά της κατηγορίας Γ σχετίζονται με την κατανομή των κόμβων που δίνεται σε μια εκτέλεση της εφαρμογής από το χρονοδρομολογητή του συστήματος. Έτσι, έχουμε δύο επιλογές για να σαρώσουμε διαφορετικές τιμές για τα χαρακτηριστικά της κατηγορίας Γ: είτε να ζητήσουμε ρητά από το χρονοδρομολογητή κατανομές κόμβων με συγκεκριμένα χαρακτηριστικά, είτε να εκτελέσουμε το benchmark πολλές φορές σε διαφορετικές κατανομές κόμβων, που επιλέγονται τυχαία από το χρονοδρομολογητή. Ακολουθούμε τη δεύτερη προσέγγιση και στα δύο συστήματα, καθώς στο σύστημα Vilje η πρώτη προσέγγιση δεν είναι δυνατή: η παρούσα ρύθμιση του λογισμικού συστήματος δεν επιτρέπει στο χρήστη την επιλογή συγκεκριμένων κόμβων για την εκτέλεση της εφαρμογής του. Επιτελούμε πολλαπλές εκτελέσεις του benchmark σε κάθε σύστημα, καθώς και πρόσθετες εκτελέσεις για μικρά μεγέθη μηνυμάτων, σε περιπτώσεις όπου παρατηρείται μεγάλη μεταβλητότητα για διαφορετικές κατανομές. Επιπλέον, μεταξύ διαφορετικών εκτελέσεων, αλλάζουμε εσωτερικές παραμέτρους του benchmark που επηρεάζουν την επιλογή των τυχαίων ζευγών διεργασιών, έτσι ώστε να συλλέγονται δεδομένα για διαφορετικές αναλογίες επικοινωνίας εντός και εκτός του κόμβου. Τα δεδομένα που συλλέγουμε από τη διαδικασία του benchmarking συσχετίζουν όλα τα χαρακτηριστικά με το χρόνο επικοινωνίας και εξυπηρετούν ως σύνολο εκπαίδευσης για την κατασκευή μοντέλων επικοινωνίας σε κάθε σύστημα. Ο χρόνος που απαιτείται για τη συλλογή του συνόλου εκπαίδευσης σε κάθε σύστημα δεν ξεπερνά τις τέσσερις ώρες.

Για να συλλέξουμε μετρήσεις για το χρόνο επικοινωνίας, το benchmark μας πραγματοποιεί μια κλήση συγχρονισμού των διεργασιών στο MPI, με τη συνάρτηση *MPI_Barrier*, πριν την εκκίνηση ενός μετρητή του χρόνου, ώστε να αποφύγουμε τη μέτρηση ασυμμετριών στο χρόνο μεταξύ των διεργασιών (άεργο χρόνο) ως χρόνο επικοινωνίας. Ωστόσο, πρέπει να σημειώσουμε ότι παρόλο που η συνάρτηση *MPI_Barrier* χρησιμοποιείται κατά κόρον για το συγχρονισμό των διεργασιών [Hunold and Carpen-Amarie, 2015], η ποιότητα του συγχρονισμού που προσφέρει εξαρτάται από το σύστημα. Εναλλακτικές προσεγγίσεις για το συγχρονισμό μπορούν να αποδώσουν καλύτερη ποιότητα μετρήσεων. Κάθε διεργασία μετράει το δικό της, τοπικό, χρόνο επικοινωνίας για κάθε επανάληψη και αναφέρει το 3ο ποσοστημόριο των χρόνων που μετρήθηκαν για όλες τις επαναλήψεις. Με αναγωγή (reduction), μία διεργασία-ρίζα αναφέρει το μέγιστο από τους τοπικούς χρόνους που αναφέρθηκαν από όλες τις διεργασίες, ως το χρόνο ολοκλήρωσης της φάσης επικοινωνίας. Επιλέξαμε τη χρήση του 3ου ποσοστημορίου του τοπικού χρόνου επικοινωνίας, αντί της διαμέσου ή του μεγίστου, για να αποφύγουμε τυχόν ακραίες τιμές που θα μάς έδινε ο μέγιστος τοπικός χρόνος επικοινωνίας, χωρίς ωστόσο να αποκλείουμε εντελώς διαφορές και απρόβλεπτες αυξήσεις στο χρόνο επικοι-



Σχήμα 3.3: Κατασκευή μοντέλου με γραμμική παλινδρόμηση πολλών μεταβλητών.

νωνίας, λόγω θορύβου, συμφόρησης ή παρεμβολής από γειτονικές εφαρμογές που χρησιμοποιούν το δίκτυο.

3.5 Πρόβλεψη χρόνου επικοινωνίας με στατιστική μάθηση

Ως πρώτη προσέγγιση στη μοντελοποίηση του χρόνου επικοινωνίας, χρησιμοποιούμε τεχνικές στατιστικής παλινδρόμησης, που μας επιτρέπουν να κατανοήσουμε την ακριβή σχέση μεταξύ των χαρακτηριστικών που ορίζουμε και

του χρόνου επικοινωνίας. Ο στόχος της μοντελοποίησης είναι να συνδέσουμε το χρόνο επικοινωνίας με το σύνολο των χαρακτηριστικών που περιγράφονται στην Ενότητα 3.3, που λειτουργούν ως επεξηγηματικές μεταβλητές (explanatory variables). Υποθέτοντας ότι η επικοινωνία επηρεάζεται από πολλούς παράγοντες, οι οποίοι αντικατοπτρίζονται στα πολλαπλά χαρακτηριστικά που έχουμε ορίσει, χρησιμοποιούμε ως καταλληλότερο υπόδειγμα το πολυμεταβλητό υπόδειγμα γραμμικής παλινδρόμησης (multiple variable linear regression model), καθώς και σθεναρή παλινδρόμηση (robust regression) για την εκτίμηση των συντελεστών του υποδείγματος υπό την υπόθεση ότι τα σφάλματα είναι ετεροσκεδαστικά και ότι υπάρχουν ακραίες τιμές που μπορούν να επηρεάσουν τις τιμές των συντελεστών. Σε αυτήν την προσέγγιση μοντελοποίησης, χρησιμοποιούμε μεταβλητές από τα χαρακτηριστικά των κατηγοριών Α και Β, ενώ από τα χαρακτηριστικά της κατηγορίας Γ χρησιμοποιούμε μόνο εκείνα που αφορούν το σχήμα της κατανομής των πόρων. Σε αυτή την πρώιμη επιλογή χαρακτηριστικών μάς οδήγησε η εγγενής δυσκολία να οριστεί η μορφή του μοντέλου. Ο αριθμός των χαρακτηριστικών υπό εξέταση πρέπει να είναι περιορισμένος, προκειμένου να καταστεί δυνατή η εξέταση της σχέσης κάθε χαρακτηριστικού με το χρόνο επικοινωνίας και των αλληλεπιδράσεων (interactions). Άλλωστε, καθώς τα περισσότερα από τα χαρακτηριστικά μας περιγράφουν την κίνηση μέσω διαφορετικών σημείων του δικτύου, πολλά από τα χαρακτηριστικά παρουσιάζουν υψηλή συσχέτιση μεταξύ τους, δηλαδή είναι αλληλένδετα (interrelated). Η χρήση αλληλένδετων χαρακτηριστικών στο πολυμεταβλητό γραμμικό μοντέλο μπορεί να εμποδίσει τη σύγκλιση της μεθόδου παλινδρόμησης κατά τον υπολογισμό των συντελεστών του μοντέλου.

Μετά την επιλογή της μεθόδου και του αρχικού συνόλου χαρακτηριστικών, επιλέγουμε εκείνα τα χαρακτηριστικά που θα χρησιμεύσουν ως επεξηγηματικές μεταβλητές του μοντέλου. Αρχικά, εφαρμόζουμε τους συντελεστές συσχέτισης του Pearson για να αποφασίσουμε ποια χαρακτηριστικά έχουν μεγάλη επίδραση στον χρόνο επικοινωνίας. Οι συντελεστές συσχέτισης του Pearson r, για ένα χαρακτηριστικό x και το χρόνο επικοινωνίας t ορίζονται ως εξής:

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(t_i-\bar{t})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(t_i-\bar{t})^2}}$$

όπου n είναι ο αριθμός των σημείων στο σύνολο εκπαίδευσης. Μια τιμή του συντελεστή ίση με (-) 1 δηλώνει μια τέλεια (θετική) γραμμική συσχέτιση μεταξύ x και t, ενώ μια τιμή 0 δηλώνει την απουσία οποιασδήποτε συσχέτισης. Χρησιμοποιούμε επίσης διαγράμματα διασποράς (scatterplots) για να ελέγξουμε εάν η συσχέτιση είναι γραμμική και αν υπάρχει ανάγκη να δημιουργηθούν περισσότερα από ένα μοντέλα για διαφορετικές κλίμακες τιμών των διαφόρων χαρακτηριστικών. Τα διαγράμματα διασποράς είναι επίσης σε θέση να αποκαλύψουν την ύπαρξη αλληλεπιδράσεων μεταξύ χαρακτηριστικών, και

αν υπάρχουν τέτοιες αλληλεπιδράσεις, εκτελούμε ομαδοποίηση σε σχέση με άλλα χαρακτηριστικά, για να εξάγουμε τους όρους αλληλεπίδρασης του μοντέλου (-ων), δηλαδή προϊόντα δύο ή περισσοτέρων μεταβλητών. Δεύτερον, κατασκευάζουμε τη μορφή του μοντέλου, ενσωματώνοντας τις επιλεγμένες μεταβλητές και τους όρους αλληλεπίδρασης. Τέλος, εφαρμόζουμε σθεναρή πολυμεταβλητή παλινδρόμηση στα δεδομένα του συνόλου εκπαίδευσης για να υπολογίσουμε τους συντελεστές μοντέλου, χρησιμοποιώντας την κλάση *LinearModel* που είναι διαθέσιμη στο λογισμικό Matlab R2013a. Η διαδικασία κατασκευής ενός πολυμεταβλητού μοντέλου γραμμικής παλινδρόμησης απεικονίζεται στο Σχήμα 3.3. Αναφερόμαστε στα μοντέλα που έχουν δημιουργηθεί με πολυμεταβλητή γραμμική παλινδρόμηση ως *LinReg - MV*.

Αν και το πλήθος των παραγόντων που επηρεάζουν το χρόνο επικοινωνίας διαισθητικά μάς οδηγεί προς μοντέλα πολλών μεταβλητών για το χρόνο επικοινωνίας, κατά τη διαδικασία μοντελοποίησης, εντοπίσαμε χαρακτηριστικά τα οποία εμφάνιζαν υψηλό βαθμό συσχέτισης με το χρόνο επικοινωνίας. Αυτή η παρατήρηση μάς προέτρεψε να κατασκευάσουμε επιπλέον μοντέλα για το χρόνο επικοινωνίας, τα οποία είναι γραμμικά μοντέλα μίας μεταβλητής. Εργαζόμαστε προς την κατεύθυνση αυτή ώστε να ελέγξουμε τη σημασία της χρήσης πολλαπλών χαρακτηριστικών για την πρόβλεψη της επικοινωνίας. Καθώς τα μοντέλα απλής μεταβλητής είναι πολύ εύκολο να κατασκευαστούν, επιλέγουμε το ένα χαρακτηριστικό με την υψηλότερη τιμή του συντελεστή συσχέτισης για κάθε σύστημα, από οποιαδήποτε κατηγορία χαρακτηριστικών, για την κατασκευή μοντέλων παλινδρόμησης μίας μεταβλητής. Για να προσδιορίσουμε τη σχέση μεταξύ του επιλεγμένου χαρακτηριστικού και του στόχου, και να αποφασίσουμε σχετικά με τη μορφή του μοντέλου, κατασκευάζουμε ένα διάνυσμα πολυωνυμικών και μη γραμμικών μετασχηματισμών του επιλεγμένου χαρακτηριστικού x, $X = \{1, x, x^2, x^3, log_2 x, log_2^2 x, log_2^3 x, \sqrt{x}\}$ και δημιουργούμε 7806 γραμμικά μοντέλα της μορφής $\hat{t} = b_0 + \sum_{i=1}^{3} b_i x_j x_k, x_j, x_k \in X, j \neq k$, τα οποία εκπαιδεύουμε με το σύνολο εκπαίδευσης που συλλέγουμε με τη διαδικασία συλλογής μετρήσεων αναφοράς. Στη συνέχεια, επιλέγουμε εκείνο το μοντέλο που παρουσιάζει βέλτιστη προσαρμογή στο σύνολο εκπαίδευσης, στο οποίο αναφερόμαστε ως *LinReg - SV*, δηλαδή γραμμική παλινδρόμηση απλής μεταβλητής. Απεικονίζουμε τη διαδικασία κατασκευής μοντέλων παλινδρόμησης μίας μεταβλητής στο Σχήμα 3.4.

3.6 Πρόβλεψη χρόνου επικοινωνίας με μηχανική μάθηση

Ένα σημαντικό πρόβλημα στη χρήση μεθόδων στατιστικής μάθησης για την κατασκευή μοντέλων πρόβλεψης είναι ότι απαιτούν σημαντική προσπάθεια από τον κατασκευαστή - χρήστη. Απαιτούν την επιλογή χαρακτηριστι-

κών, την ταυτοποίηση σχέσεων, γραμμικών ή μη γραμμικών, μεταξύ των χαρακτηριστικών και του χρόνου επικοινωνίας, την εύρεση όρων αλληλεπίδρασης, πολυβάθμιων όρων, καθώς και χώρους παραμέτρων που μπορούν να περιγράψουν με ακρίβεια τη σχέση μεταξύ ενός χαρακτηριστικού και του στόχου. Σε αντίθεση με τα παραπάνω, οι τεχνικές μηχανικής μάθησης μπορούν να αυτοματοποιήσουν τη διαδικασία κατασκευής σε πολύ μεγάλο βαθμό, αφού δύνανται να επιλέγουν τα κατάλληλα χαρακτηριστικά και να κατασκευάζουν μοντέλα που εμπεριέχουν τις περίπλοκες σχέσεις μεταξύ των χαρακτηριστικών και του στόχου, αποκρύπτοντας ωστόσο αυτή την πολυπλοκότητα από τον κατασκευαστή-χρήστη του μοντέλου.

Δίνουμε προτεραιότητα σε μεθόδους συλλογικής μάθησης, όπως τα Random Forests, τα Extra Randomized Trees, τα Gradient Boosted Regression Trees και η AdaBoost Regression, που όλες είναι διαθέσιμες στη βιβλιοθήκη λογισμικού *scikit-16.1* της Python. Η αυτοματοποίηση της διαδικασίας κατασκευής του μοντέλου είναι το βασικό πλεονέκτημα των μεθόδων αυτών, ενώ επιπλέον οι μέθοδοι συλλογικής μάθησης δεν απαιτούν την ταυτοποίηση των όποιων γραμμικών ή μη γραμμικών σχέσεων μεταξύ των χαρακτηριστικών και του στόχου, των αλληλεπιδράσεων μεταξύ χαρακτηριστικών, των πολυβάθμιων όρων και των διαστημάτων τιμών των χαρακτηριστικών που περιγράφουν με μεγαλύτερη ακρίβεια τη σχέση μεταξύ ενός χαρακτηριστικού και του στόχου. Επίσης, καθώς οι συλλογικές μέθοδοι συνδυάζουν πολλαπλά μοντέλα, βελτιώνουν την ακρίβεια και τη σθεναρότητα, σε σύγκριση με ένα μόνο μοντέλο [Mendes-Moreira et al., 2012]. Όλες οι παραπάνω ιδιότητες καθιστούν τις μεθόδους συλλογικής μάθησης ελκυστική λύση για την κατασκευή μοντέλων σε οποιοδήποτε σύστημα, αφού δεν απαιτούν ιδιαίτερη προσπάθεια για την κατασκευή του μοντέλου σε κάθε σύστημα. Τέλος, σημειώνουμε ότι η χρονική επιβάρυνση για την εκπαίδευση του μοντέλου είναι αμελητέα, αφού ο χρόνος εκπαίδευσης είναι μικρός (χρειάζονται λιγότερο από δύο λεπτά για τις πιο χρονοβόρες μεθόδους) και η εκπαίδευση γίνεται μόνο μία φορά για κάθε μοντέλο.

Σε αυτή την προσέγγιση, με τη χρήση μηχανικής μάθησης, κατασκευάζουμε τρία διαφορετικά μοντέλα για κάθε σύστημα. Το πρώτο μοντέλο χρησιμοποιεί χαρακτηριστικά μόνο της κατηγορίας Α, το δεύτερο μοντέλο χρησιμοποιεί χαρακτηριστικά των κατηγοριών Α και Β και το τρίτο μοντέλο χρησιμοποιεί επίσης χαρακτηριστικά της κατηγορίας Γ για κάθε σύστημα. Αναφερόμαστε στα τρία μοντέλα ως *Class A*, *Class B* και *Class C* αντίστοιχα. Με αυτόν τον τρόπο, μπορούμε να συγκρίνουμε και να αξιολογήσουμε την αξία της προσθήκης επιπλέον γνώσης από την απεικόνιση των διεργασιών και την αρχιτεκτονική και τοπολογία του συστήματος σε ένα μοντέλο, σε σχέση με την ακρίβεια και τον χρόνο πρόβλεψης. Λόγω του σχεδιασμού του προγράμματος για τη συλλογή μετρήσεων αναφοράς, το οποίο χρησιμοποιούμε για τη συλλογή μετρήσεων για εκπαίδευση, χρησιμοποιούμε μόνο τη μέγιστη τιμή για τα

ακόλουθα χαρακτηριστικά της κατηγορίας Α: *l*, *PD* και *PM*, αφού στο benchmark μας, για μία συγκεκριμένη εκτέλεση σε κάποιες ρυθμίσεις εκτέλεσης, το μέγεθος και ο αριθμός των μηνυμάτων και τα δεδομένα ανά διεργασία είναι σταθερά.

Ξεκινάμε την κατασκευή του μοντέλου με επιλογή χαρακτηριστικών για τα χαρακτηριστικά της κατηγορίας Γ. Η επιλογή χαρακτηριστικών για τα μοντέλα των κατηγοριών Α και Β δεν είναι απαραίτητη, καθώς οι μέθοδοι συλλογικής μάθησης που εφαρμόζουμε για την κατασκευή του μοντέλου είναι αποδοτικές με τον περιορισμένο αριθμό των χαρακτηριστικών των κατηγοριών αυτών. Βαθμονομούμε τα χαρακτηριστικά του μοντέλου της κατηγορίας Γ χρησιμοποιώντας τη μέθοδο αναδρομικής απαλοιφής χαρακτηριστικών (recursive feature elimination) βασισμένης σε παλινδρόμηση με διανύσματα υποστήριξης (support vector regression). Η μέθοδος αυτή προτάθηκε αρχικά στην εργασία [Guyon et al., 2002] που αναφέρεται σε μοντέλα για γενετική διάγνωση. Η παλινδρόμηση με διανύσματα υποστήριξης παρέχει υψηλή ικανότητα γενίκευσης, ενώ η αναδρομική απαλοιφή χαρακτηριστικών συμβάλλει στη μείωση του βαθμού των χαρακτηριστικών εκείνων που προσαρμόζονται υπερβολικά καλά (overfit) στο σύνολο εκπαίδευσης. Πειραματιζόμαστε με την τροφοδότηση 15 έως 40 από τα χαρακτηριστικά με τους περισσότερο υψηλούς βαθμούς ως εισόδους στις μεθόδους συλλογικής μάθησης.

Μετά την επιλογή χαρακτηριστικών, χρησιμοποιούμε Gradient Boosting Regression Trees (GBRT) για την κατασκευή ενός μοντέλου για το χρόνο επικοινωνίας. Η μέθοδος GBRT είναι μια μέθοδος ενισχυτικής μάθησης, όπου ένας νέος, αδύναμος εκπαιδευόμενος (weak learner), και συγκεκριμένα ένα δέντρο απόφασης παλινδρόμησης, προσαρμόζεται επαναληπτικά στην αρνητική κλίση μίας δοσμένης συνάρτησης απωλειών (loss function). Η μέθοδος αυτή προσφέρει υψηλή ακρίβεια, αυτοματοποίηση της διαδικασίας κατασκευής του μοντέλου, καθώς και ένα πλήθος ειδικών παραμέτρων προς ρύθμιση, που διευκολύνουν την κατασκευή μοντέλων για την πρόβλεψη του χρόνου επικοινωνίας. Χρησιμοποιούμε ως συνάρτηση απωλειών την ελάχιστη απόλυτη απόκλιση, που συμβάλλει στη σθεναρότητα του μοντέλου σε ακραίες τιμές (outliers). Αυτή η ιδιότητα της συνάρτησης απωλειών είναι κρίσιμη για την εργασία μας, καθώς υπάρχει θόρυβος στις μετρήσεις του χρόνου επικοινωνίας και συχνά εμφανίζονται ακραίες τιμές στο σύνολο εκπαίδευσης. Για να αποτρέψουμε την υπερπροσαρμογή (overfitting) των μοντέλων μας στο σύνολο εκπαίδευσης, εφαρμόσαμε κανονικοποίηση (regularization) με τη μέθοδο της συρρίκνωσης (shrinkage), ορίζοντας το ρυθμό εκμάθησης (learning rate) και ρυθμίζοντας το σχήμα των δέντρων απόφασης.

Για να βρούμε τις βέλτιστες ρυθμίσεις για τη μέθοδο GBRT και να κατασκευάσουμε το βέλτιστο μοντέλο, για το διαθέσιμο σύνολο εκπαίδευσης, δοκιμάζουμε τη μέθοδο με διαφορετικές τιμές για το ρυθμό εκμάθησης (*learn-*

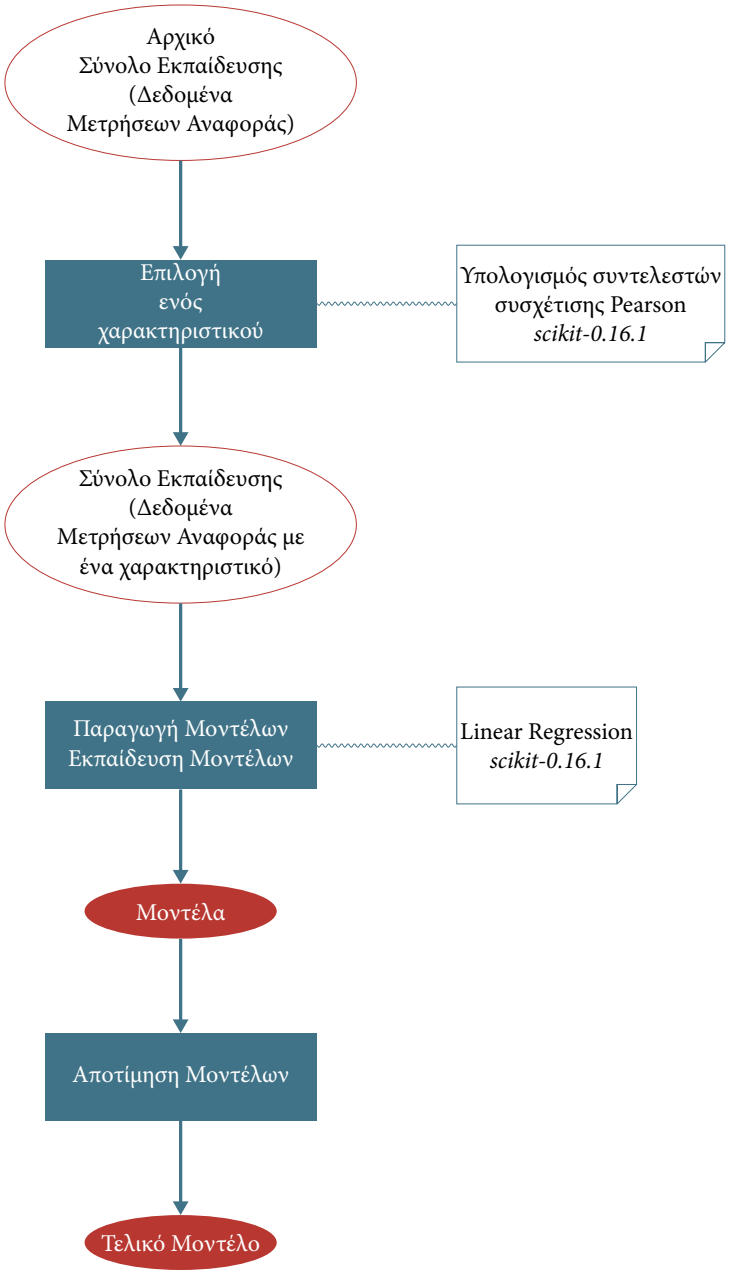
ing_rate) και τα ελάχιστα δείγματα ανά φύλλο (*min_samples_leaf*) των δέντρων αποφάσεων και τα ελάχιστα δείγματα για τη διαίρεση ενός κόμβου στο δέντρο απόφασης (*min_samples_split*), καθώς και διαφορετικά πλήθη δέντρων αποφάσεων (*n_estimators*) και επιλέξαμε τις τιμές που απέδιδαν το μοντέλο με την καλύτερη προσαρμογή, εξετάζοντας τις μετρικές αποτίμησης που ορίσαμε στο Κεφάλαιο 3. Αναφερόμαστε, στο εξής, στα μοντέλα που κατασκευάζουμε με τη μέθοδο GBRT για κάθε κατηγορία ως *GBRT-Class A*, *GBRT-Class B* και *GBRT-Class C* για τις κατηγορίες Α, Β και Γ αντίστοιχα. Συνοψίζουμε αυτή τη διαδικασία μοντελοποίησης στο Σχήμα 3.5.

3.7 Σύγκριση με αναλυτικά και ημι-εμπειρικά μοντέλα

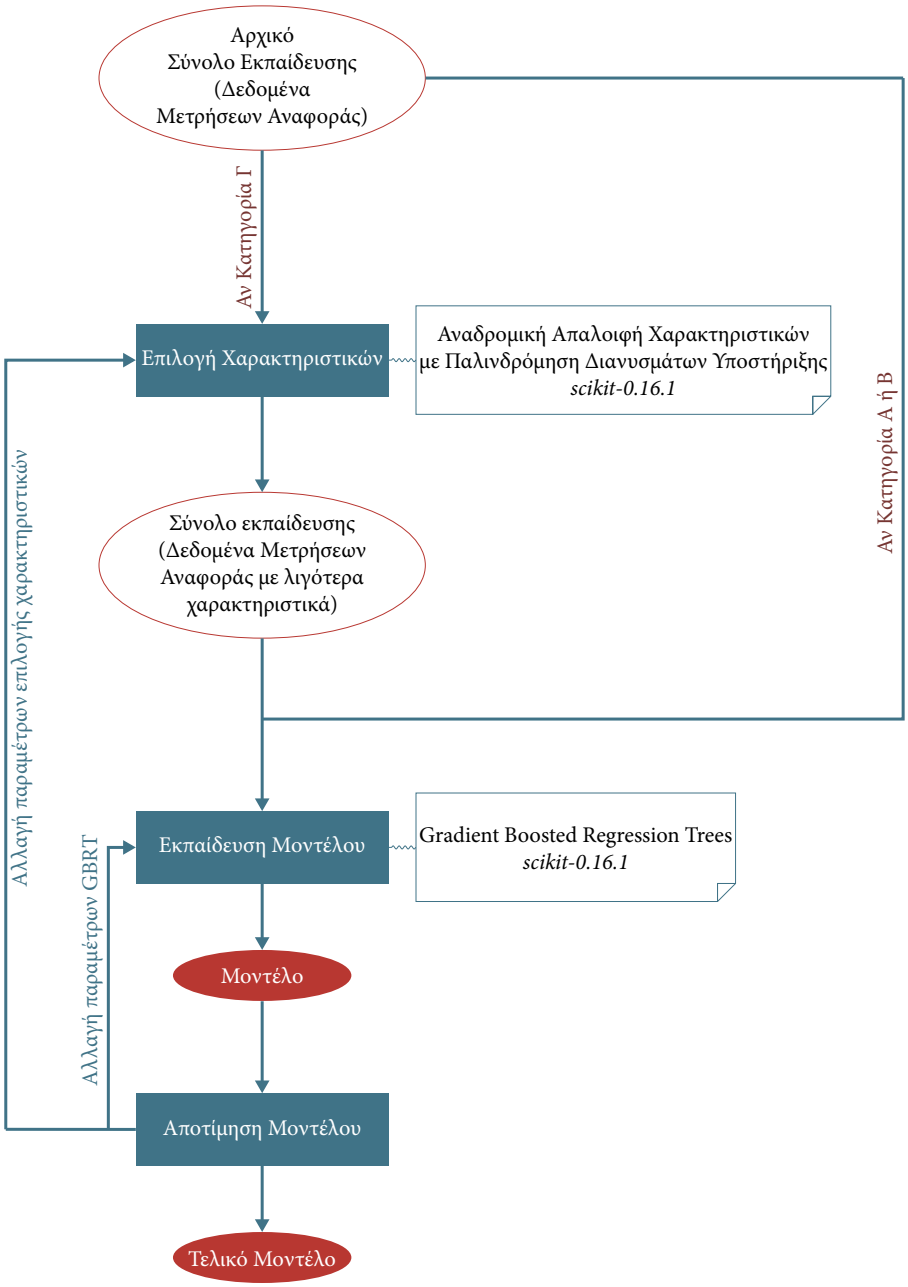
Για λόγους σύγκρισης, υλοποιήσαμε αναλυτικά και ημι-εμπειρικά μοντέλα όπως προτείνονται από τον Gahvari και τους συνεργάτες του [Gahvari et al., 2011, 2014]. Η βασική αναλυτική προσέγγιση είναι το μοντέλο $\alpha - \beta$, που είναι αντίστοιχο του μοντέλου χρόνου απόκρισης - εύρους ζώνης του Hockney, όπου α ο χρόνος απόκρισης και β το ανάστροφο του εύρους ζώνης. Μια τρίτη παράμετρος γ αναφέρεται στην καθυστέρηση ανά βήμα (per-hop delay) και εισάγει στο μοντέλο μια ποινή λόγω απόστασης, καταλήγοντας στο μοντέλο α-β-γ. Μετρούμε τις τρεις παραμέτρους με τη χρήση του HPCC benchmark[3] Τα δύο μοντέλα μπορούν να ενισχυθούν με τρεις επιπλέον "ποινές". Η πρώτη προστίθεται προαιρετικά στην παράμετρο β, για να ενσωματώσει τα όρια του πραγματικού εύρους ζώνης που μπορεί να επιτευχθεί και τη συμφόρηση του δικτύου λόγω διαμοιρασμού των συνδέσεων. Η δεύτερη προστίθεται προαιρετικά στην παράμετρο α, για να ενσωματώσει το φαινόμενο των πολλαπλών διεργασιών που προσπαθούν να αποκτήσουν πρόσβαση στο δίκτυο από τον ίδιο κόμβο (multicore penalty). Η ίδια ποινή μπορεί να προστεθεί στην παράμετρο γ, καθώς πολλαπλές διεργασίες μπορούν να προκαλέσουν φαινόμενα ανταγωνισμού σε κάθε βήμα ενός μηνύματος στο δίκτυο.

Η προσθήκη αυτών των τριών ποινών καταλήγει σε πέντε δυνατά μοντέλα. Αξιολογούμε και τα πέντε μοντέλα και παρουσιάζουμε τα αποτελέσματα της πρόβλεψης για το αναλυτικό μοντέλο α - β και για το καλύτερο από τα πέντε ημι-εμπειρικά μοντέλα με ποινές για κάθε σύστημα. Αξίζει να σημειωθεί πως, ενώ δεν προσδοκούμε από το μοντέλο α-β να αποδώσει όλη την πολυπλοκότητα της επίδοσης της επικοινωνίας και ακριβή αποτελέσματα προβλέψεων, αποτελεί την απλούστερη έκφραση του χρόνου επικοινωνίας, ενσωματώνει την επίδραση του μεγέθους του προβλήματος και του διαμοιρασμού του προβλήματος στις διεργασίες και επομένως αποτελεί ένα κατώτατο όριο για την ακρίβεια της πρόβλεψης.

[3] http://icl.cs.utk.edu/hpcc/



Σχήμα 3.4: Κατασκευή μοντέλου με γραμμική παλινδρόμηση μίας μεταβλητής με μη γραμμικούς όρους.



Σχήμα 3.5: Κατασκευή μοντέλου με τη μέθοδο Gradient Boosting Regression Trees.

4

Πρόβλεψη επίδοσης επικοινωνίας στο σύστημα *Vilje*

4.1 Εισαγωγή

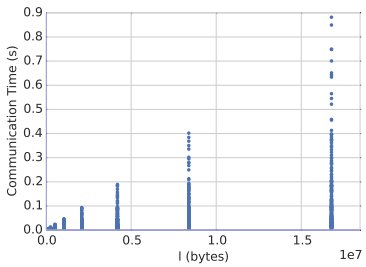
Σε αυτό το κεφάλαιο, παρουσιάζουμε την εφαρμογή της μεθοδολογίας μας (βλ. Ενότητες 3.5 και 3.6) για τη μοντελοποίηση της επικοινωνίας με στόχο την πρόβλεψη, στο σύστημα *Vilje*. Αναλύουμε τη διαδικασία κατασκευής μοντέλων με τεχνικές στατιστικής μάθησης και τεχνικές μηχανικής μάθησης. Ακολούθως, αποτιμούμε την προβλεπτική ικανότητα των μοντέλων μας σε αρκετές εφαρμογές, συγκρίνουμε τα διάφορα μοντέλα και παρέχουμε λεπτομερή αποτίμηση για το μοντέλο με την καλύτερη ικανότητα πρόβλεψης. Τέλος, χρησιμοποιούμε τα μοντέλα μας για το χρόνο επικοινωνίας σε σενάρια λήψης αποφάσεων που αφορούν τη βελτιστοποίηση της χρήσης των πόρων του συστήματος Vilje.

4.2 Μοντελοποίηση με στατιστική μάθηση

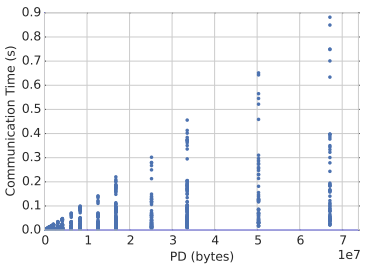
Για να μοντελοποιήσουμε το χρόνο επικοινωνίας στο Vilje με τη χρήση του πολυμεταβλητού γραμμικού μοντέλου παλινδρόμησης, χρησιμοποιήσαμε χαρακτηριστικά των κατηγοριών Α και Β, ενώ από τα χαρακτηριστικά της κατηγορίας Γ, επιλέξαμε μόνο τα χαρακτηριστικά που αντικατοπτρίζουν το σχήμα της κατανομής των κόμβων: το πλήθος των διακοπτών *sw*, το πλήθος των racks *r* και το μέσο αριθμό διακοπτών ανά rack *sw/r*. Για τα χαρακτηριστικά των κατηγοριών Α και Β που αναφέρονται σε δεδομένα ή πλήθη μηνυμάτων, χρησιμοποιήσαμε μόνο τη μέση τιμή. Επιπρόσθετα, παραλείψαμε τα δύο χαρακτηριστικά της κατηγορίας Β που αναφέρονται σε κίνηση εντός του κόμβου (δεδομένα, *iND*, και μηνύματα, *iNM*). Αντί αυτών, χρησιμοποιούμε ένα

Πίνακας 4.1: Συσχέτιση των χαρακτηριστικών και του χρόνου επικοινωνίας στο σύστημα Vilje

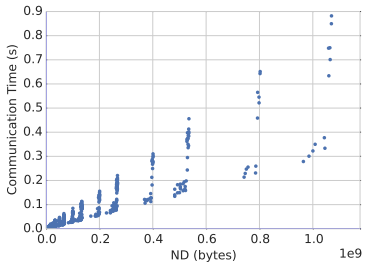
Χαρακτηριστικό	Συσχέτιση	Χαρακτηριστικό	Συσχέτιση
l	0.62	*TD*	0.87
PM	0.10	*NM*	0.24
ppn	0.20	*TM*	0.22
n	0.07	*sw*	0.07
I	0	*r*	0.05
PD	0.68	*sw/r*	0.06
ND	0.93		



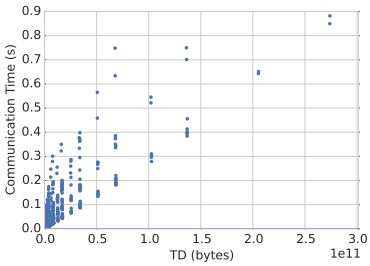
(i) *l* και χρόνος επικοινωνίας



(ii) *PD* και χρόνος επικοινωνίας



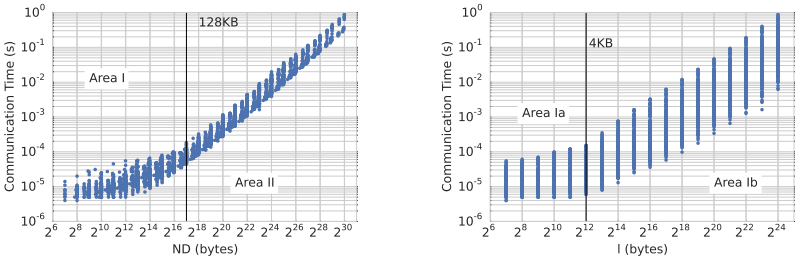
(iii) *ND* και χρόνος επικοινωνίας



(iv) *TD* και χρόνος επικοινωνίας

Σχήμα 4.1: Διαγράμματα διασποράς για τα χαρακτηριστικά με υψηλή συσχέτιση με το χρόνο επικοινωνίας στο Vilje.

χαρακτηριστικό που μετρά το λόγο της κίνησης μεταξύ κόμβων προς το συνολικό όγκο της επικοινωνίας, που συμβολίζουμε ως *I*. Με αυτό τον τρόπο, ελαττώνουμε τον αριθμό των χαρακτηριστικών σε 13, για να διευκολύνουμε την ανίχνευση σχέσεων μεταξύ όλων των χαρακτηριστικών και του χρόνου επικοινωνίας και την κατασκευή της μορφής του μοντέλου. Για τη συλλογή του συνόλου εκπαίδευσης, εκτελέσαμε το benchmark μας στο Vilje για διαφορετικές παραμέτρους εκτέλεσης, από 8 έως 128 κόμβους, 1 έως 16 διεργα-



(i) Διαίρεση στις περιοχές I και II **(ii)** Διαίρεση στις περιοχές Ia και Ib

Σχήμα 4.2: Διαίρεση του χώρου των παραμέτρων σε υπο-περιοχές στο Vilje.

σίες ανά κόμβο, 1 έως 4 μηνύματα ανά διεργασία και διάφορα μεγέθη μηνυμάτων, από 128 bytes έως 16 megabytes. Επαναλάβαμε τη διαδικασία συλλογής μετρήσεων αναφοράς για όλες τις ρυθμίσεις εκτέλεσης, ώστε να συλλέξουμε διαφορετικές τιμές για τα χαρακτηριστικά εκείνα που εξαρτώνται από το σχήμα της κατανομής των κόμβων, δηλαδή τα χαρακτηριστικά *sw*, *r* και *sw/r*, καταλήγοντας σε ένα σύνολο εκπαίδευσης 4320 μετρήσεων για το χρόνο επικοινωνίας. Οι δύο πλήρεις εκτελέσεις του benchmark διαφέρουν επίσης στο σχήμα επικοινωνίας, καθώς μεταβάλλαμε εκείνες τις παραμέτρους του benchmark που καθορίζουν τη δημιουργία των τυχαίων ζευγών των διεργασιών που επικοινωνούν. Στον Πίνακα 4.1 δείχνουμε τους συντελεστές συσχέτισης του Pearson μεταξύ όλων των χαρακτηριστικών και του χρόνου επικοινωνίας, όπου παρατηρούμε πολύ υψηλή συσχέτιση για τα χαρακτηριστικά *ND* και *TD* και υψηλή συσχέτιση για τα χαρακτηριστικά *PD* και *l*. Για να επιβεβαιώσουμε ότι αυτή η υψηλή συσχέτιση υπονοεί γραμμική σχέση μεταξύ των χαρακτηριστικών αυτών και του χρόνου επικοινωνίας, επιθεωρήσαμε τα διαγράμματα διασποράς τους, όπως εμφανίζονται στο Σχήμα 4.1, όπου εντοπίσαμε όντως γραμμική σχέση. Ωστόσο, το διάγραμμα διασποράς για το χαρακτηριστικό *ND* σε λογαριθμική κλίμακα, όπως φαίνεται στο Σχήμα 4.2i, αποκάλυψε ότι η σχέση μεταξύ του συγκεκριμένου χαρακτηριστικού με το χρόνο επικοινωνίας εμφανίζει διαφορετική κλίση για μικρότερες και μεγαλύτερες τιμές του χαρακτηριστικού. Επομένως, διαιρέσαμε το σύνολο εκπαίδευσης σε δύο υποσύνολα/υποπεριοχές, τις περιοχές I ($ND \leq 128 kilobytes$) και II ($ND > 128 kilobytes$). Για την περιοχή I, η επιθεώρηση του διαγράμματος διασποράς για το μέγεθος του μηνύματος l και το χρόνο επικοινωνίας, σε λογαριθμική κλίμακα, (βλ. Σχήμα 4.2ii) επίσης αποκάλυψε διαφορετικές κλίσεις για διαφορετικά μεγέθη μηνυμάτων, οδηγώντας μας να διαιρέσουμε περαιτέρω την περιοχή I σε δύο υποπεριοχές, τις Ia ($l \leq 4 kilobytes$) και Ib ($l > 4 kilobytes$). Εξετάσαμε τις τρεις περιοχές ξεχωριστά και κατασκευάσαμε ένα μοντέλο για την καθεμιά.

Πίνακας 4.2: Μοντέλο πρόβλεψης επικοινωνίας για την Περιοχή Ia στο σύστημα Vilje: Όροι και συντελεστές

Όροι	**Συντελεστές**
Σταθερός Όρος	6.6111×10^{-6}
$ppn \times ND$	5.8226×10^{-12}
$l \times ND$	-1.42×10^{-13}
$r \times ND$	-5.967×10^{-11}
$ppn \times NM$	2.1315×10^{-08}
$l \times NM$	1.0384×10^{-09}
$r \times NM$	8.1344×10^{-08}
$ppn \times l \times ND$	-5.9903×10^{-16}
$ppn \times r \times ND$	1.2994×10^{-12}
$l \times r \times ND$	1.2228×10^{-14}
$ppn \times r \times NM$	-4.7418×10^{-09}
$ppn \times l \times r \times ND$	-3.7129×10^{-16}

Πίνακας 4.3: Μοντέλο πρόβλεψης επικοινωνίας για την Περιοχή Ib στο σύστημα Vilje: Όροι και συντελεστές

Όροι	**Συντελεστές**
Σταθερός Όρος	0
l	7.7787×10^{-10}
sw/r	1.2193×10^{-06}
$l \times sw/r$	-6.1059×10^{-11}
ND	2.8278×10^{-10}
TD	1.2398×10^{-12}
$l \times ND$	-5.7408×10^{-15}
$sw/r \times ND$	2.6589×10^{-11}
$l \times TD$	3.6211×10^{-18}
$sw/r \times TD$	-1.8116×10^{-13}
$l \times sw/r \times ND$	6.9513×10^{-16}
$l \times sw/r \times TD$	2.3784×10^{-18}

Για κάθε μία από τις τρεις περιοχές, υπολογίσαμε εκ νέου τους συντελεστές συσχέτισης, για να κατασκευάσουμε απλά γραμμικά μοντέλα, επιλέγοντας τα χαρακτηριστικά με τους μεγαλύτερους βαθμούς συσχέτισης σε κάθε περιοχή:

- Μοντέλο Ia: $t_{comm} = b_0 + b_1 \times ND + b_2 \times NM$
- Μοντέλο Ib: $t_{comm} = b_0 + b_1 \times ND + b_2 \times TD$
- Μοντέλο II: $t_{comm} = b_0 + b_1 \times ND$

Πίνακας 4.4: Μοντέλο πρόβλεψης επικοινωνίας για την Περιοχή ΙΙ στο σύστημα Vilje: Όροι και συντελεστές

Όροι	**Συντελεστές**
Σταθερός Όρος	0
l	-3.5128×10^{-11}
$ppn \times l$	-1.3962×10^{-11}
$sw \times l$	1.8077×10^{-11}
$ppn \times l \times sw$	-1.0744×10^{-12}
ND	2.5808×10^{-10}
$ppn \times ND$	-3.8401×10^{-12}
$l \times ND$	4.5466×10^{-18}
$sw \times ND$	1.3029×10^{-11}
$ppn \times l \times ND$	1.5495×10^{-19}
$ppn \times sw \times ND$	3.5872×10^{-13}
$l \times sw \times ND$	-1.6864×10^{-19}
$ppn \times l \times sw \times ND$	2.1983×10^{-20}

Κατά την κατασκευή των μοντέλων, η επέκταση των μοντέλων που δίνονται παραπάνω με άλλα χαρακτηριστικά με υψηλή συσχέτιση δε βελτίωσε την ακρίβειά τους. Ωστόσο, τα διαγράμματα διασποράς στο Σχήμα 4.1 κατέδειξαν την ύπαρξη κάποιας ετεροσκεδαστικότητας, δηλαδή διακύμανσης του χρόνου επικοινωνίας για μία συγκεκριμένη τιμή ενός χαρακτηριστικού. Αυτή η παρατήρηση μάς οδήγησε να διερευνήσουμε την ύπαρξη όρων αλληλεπίδρασης, που αποτελούν γινόμενα των επιλεγμένων χαρακτηριστικών με άλλα χαρακτηριστικά, που δεν εμφανίζουν απαραίτητα υψηλή συσχέτιση με το χρόνο επικοινωνίας. Για να προσδιορίσουμε τους όρους αλληλεπίδρασης, ακολουθήσαμε τα παρακάτω βήματα: αρχικά, αποκλείσαμε χαρακτηριστικά που αποτελούν γινόμενα άλλων χαρακτηριστικών και είναι αλληλένδετα, δηλαδή εμφανίζουν υψηλή συσχέτιση μεταξύ τους, καθώς θα οδηγούσαν σε πολυβάθμιους όρους, αντί για όρους αλληλεπίδρασης. Ακολούθως, χρησιμοποιήσαμε τους συντελεστές συσχέτισης για να πραγματοποιήσουμε μία αρχική επιλογή άλλων χαρακτηριστικών. Για παράδειγμα, το μέγεθος του μηνύματος εμφανίζει υψηλή συσχέτιση με το χρόνο επικοινωνίας σε όλες τις υποπεριοχές. Τρίτον, επιλέξαμε να συμπεριλάβουμε τουλάχιστον ένα χαρακτηριστικό που αναφέρεται στο σχήμα της κατανομής των κόμβων. Πειραματιστήκαμε με πολλαπλές μορφές του μοντέλου, καταλήγοντας στις μορφές που απεικονίζονται στους πίνακες 4.2, 4.3 και 4.4, μαζί με τους συντελεστές τους, που υπολογίσαμε με σθεναρή παλινδρόμηση χρησιμοποιώντας το λογισμικό Matlab R2013a. Χρησιμοποιούμε τα μοντέλα για τις τρεις περιοχές ως ένα μοντέλο, στο οποίο ανα-

φερόμαστε ως *LinReg-MV*.

Τα τελικά μοντέλα μάς επιτρέπουν να κάνουμε ορισμένες υποθέσεις για την αρχιτεκτονική του συστήματος και τους παράγοντες που επηρεάζουν το χρόνο επικοινωνίας σε κάθε περιοχή. Το μέγεθος του μηνύματος *l* είναι καθοριστικό για την προβλεπτική ικανότητα όλων των μοντέλων, λειτουργώντας σαν μία μετρική βάσης που επιλύει τη σημασία των άλλων μετρικών που στηρίζονται σε αυτή. Ο αριθμός των διεργασιών ανά κόμβο *ppn* συμπεριλαμβάνεται στα μοντέλα των περιοχών Ia και II, υπονοώντας την ύπαρξη παρεμβολών μεταξύ των διεργασιών που απεικονίζονται στον ίδιο κόμβο. Το μοντέλο της περιοχής Ia, που αφορά σε μικρά μηνύματα και χαμηλή κίνηση ανά κόμβο, ενσωματώνει το χαρακτηριστικό που αναφέρεται σε πλήθος μηνυμάτων ανά κόμβο *NM*, υπονοώντας ότι το πλήθος των μηνυμάτων που μεταδίδονται έχει κάποια επίδραση στο χρόνο επικοινωνίας, πιθανότατα λόγω της χρήσης "eager" πρωτοκόλλου για τα μικρά μηνύματα. Το ίδιο μοντέλο επίσης χρησιμοποιεί των αριθμό των racks *r*, που υποδηλώνει ότι το εύρος της κατανομής των κόμβων (dilation) είναι σημαντικό όταν τα μηνύματα είναι μικρά, όπου περισσότερα βήματα (hops) στο δίκτυο επηρεάζουν σημαντικά το χρόνο απόκρισης. Το μοντέλο της περιοχής Ib αφορά μεγάλα μηνύματα αλλά χαμηλή κίνηση ανά κόμβο, επομένως τα δεδομένα ανά κόμβο *ND* δεν είναι η μόνη κρίσιμη παράμετρος: ο χρόνος επικοινωνίας εξαρτάται επίσης από το συνολικό όγκο των δεδομένων *TD*, καθώς και από το σχήμα της κατανομής των κόμβων και από την απεικόνιση των διεργασιών, όπως υπονοεί η ύπαρξη του χαρακτηριστικού *sw/r* στο μοντέλο. Τέλος, το μοντέλο της περιοχής II αφορά μεγάλα μηνύματα και υψηλή κίνηση ανά κόμβο, επομένως αφορά σε μία περιοχή όπου είναι δυνατό να εμφανιστούν περισσότερα φαινόμενα ανταγωνισμού και συμφόρησης σε διάφορα σημεία του δικτύου, όπως φαίνεται και από την ενσωμάτωση του αριθμού των διακοπτών *sw* στο μοντέλο.

Για την κατασκευή μοντέλου μίας μεταβλητής με στατιστική μάθηση στο Vilje, ακολουθήσαμε τη διαδικασία που περιγράφουμε στην Ενότητα 3.5. Αρχικά, υπολογίσαμε τους συντελεστές συσχέτισης του Pearson για τα χαρακτηριστικά όλων των κατηγοριών, Α, Β και Γ, στο σύστημα και επιλέξαμε το χαρακτηριστικό με τη μεγαλύτερη συσχέτιση. Στην περίπτωση του Vilje, αυτό το χαρακτηριστικό ήταν το *SD (max)*, που αναφέρεται στη μέγιστη εκροή δεδομένων ανά διακόπτη. Αυτό το εύρημα συμφωνεί με τις προσδοκίες μας για τους λόγους που επηρεάζουν την επίδοση της επικοινωνίας. Ωστόσο, πρέπει να σημειώσουμε ότι το χαρακτηριστικό αυτό αναφέρεται μόνο στην κίνηση που παράγεται από την εφαρμογή που εξετάζουμε (στην προκείμενη περίπτωση, το benchmark), ενώ στο Vilje, οι διακόπτες συχνά είναι μοιραζόμενοι μεταξύ πολλών κατανομών κόμβων, στις οποίες εκτελούνται διαφορετικές εφαρμογές. Μετά την επιλογή του χαρακτηριστικού, κατασκευάσαμε και εκτελέσαμε περισσότερα από 7000 μοντέλα που χρησιμοποιούν το χαρακτηριστικό

SD (max), χρησιμοποιώντας μη γραμμικούς μετασχηματισμούς του χαρακτηριστικού, όπως αναλύεται στην Ενότητα 3.5. Τελικά, επιλέξαμε το μοντέλο με την καλύτερη προσαρμογή στα δεδομένα του συνόλου εκπαίδευσης. Το τελικό μοντέλο για το σύστημα Vilje είναι το ακόλουθο:

$$t = 2.390 \times 10^{-5} + 4.335 \times 10^{-16} \times log_2^5 x + 1.915 \times 10^{-13} \times x log_2^2 x + 1.565 \times 10^{-14} \times x\sqrt{x}$$ (δευτερόλεπτα), όπου $x = SD(max)$ σε bytes

Αναφερόμαστε σε αυτό το μοντέλο ως *LinReg-MV*.

4.3 Μοντελοποίηση με μηχανική μάθηση

Ακολουθώντας τη διαδικασία που περιγράφουμε στην Ενότητα 3.6, κατασκευάσαμε τρία μοντέλα πρόβλεψης για το χρόνο επικοινωνίας στο Vilje, με τη χρήση της μεθόδου Gradient Boosting Regression Trees, όπου κάθε μοντέλο χρησιμοποιεί διαφορετικές κατηγορίες χαρακτηριστικών. Το πρώτο μοντέλο, που συμβολίζουμε ως *GBRT-Class A* χρησιμοποιεί χαρακτηριστικά μόνο της κατηγορίας Α, το δεύτερο μοντέλο, που συμβολίζουμε ως *GBRT-Class B*, χρησιμοποιεί χαρακτηριστικά των κατηγοριών Α και Β και το τρίτο μοντέλο, στο οποίο αναφερόμαστε ως *GBRT-Class C*, χρησιμοποιεί χαρακτηριστικά όλων των κατηγοριών. Αντίθετα με τη μοντελοποίηση με στατιστική μάθηση, δεν πραγματοποιήσαμε επιλογή χαρακτηριστικών χειροκίνητα, επομένως δε μειώσαμε το σύνολο των χαρακτηριστικών για κανένα από τα τρία μοντέλα. Σημειώνουμε, ωστόσο, ότι για τα χαρακτηριστικά *l*, *PD* and *PM* χρησιμοποιούμε μόνο τη μέγιστη τιμή, καθώς, εξαιτίας του σχεδιασμού του benchmark μας, οι τιμές τους είναι ίδιες για όλες τις διεργασίες για κάθε σημείο δεδομένων στο σύνολο εκπαίδευσης. Εφαρμόσαμε συστηματική επιλογή χαρακτηριστικών για το μοντέλο της κατηγορίας Γ, με την τεχνική της αναδρομικής απαλοιφής χαρακτηριστικών, βασισμένης σε παλινδρόμηση με διανύσματα υποστήριξης. Για τα μοντέλα των κατηγοριών Α και Β, όπου το πλήθος των χαρακτηριστικών είναι σχετικά μικρό, βασιστήκαμε στην επιλογή σημαντικών χαρακτηριστικών για την παλινδρόμηση, όπως αυτή πραγματοποιείται αυτόματα από τη μέθοδο Gradient Boosting Regression Trees.

Για να εκπαιδεύσουμε τα μοντέλα μας, ενισχύσαμε το αρχικό σύνολο εκπαίδευσης που χρησιμοποιήσαμε στη μεθοδολογία στατιστικής μάθησης. Συγκεκριμένα, προσθέσαμε μία πλήρη συλλογή μετρήσεων με το benchmark για όλα τα μεγέθη μηνυμάτων, όλους τους αριθμούς μηνυμάτων και όλες τις δυνατές παραμέτρους εκτέλεσης, όπου αλλάξαμε ξανά τις παραμέτρους που καθορίζουν το τυχαίο σχήμα επικοινωνίας. Επιπλέον, συλλέξαμε επιπρόσθετες

Πίνακας 4.5: Χώρος παραμέτρων για τις εκτελέσεις του benchmark και τη συλλογή του συνόλου εκπαίδευσης στο σύστημα Vilje

Παράμετρος	**Εύρος**	
n	8-256	8-256
ppn	1-16	1-16
l	16B-16MB	16B-16KB
PM	1-4	1-4
#εκτελέσεις	3	2
#σημεία	9210	

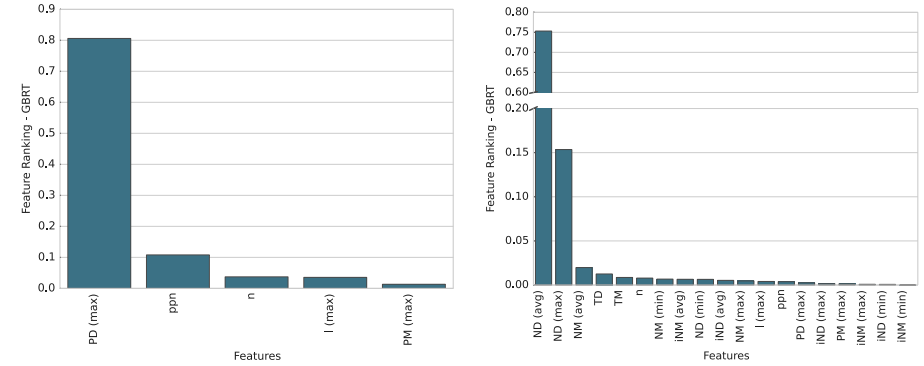
Πίνακας 4.6: Εύρος τιμών των παραμέτρων και επιλεγείσες τιμές για τις παραμέτρους της μεθόδου GBRT και το πλήθος των χαρακτηριστικών στο σύστημα Vilje

Παράμετρος	**Εύρος**	**Επιλεγείσα Τιμή**
n_estimators	100 - 2000	500
learning_rate	0.001 - 1	0.003
min_samples_leaf	1 - 5	2
min_samples_split	2 - 8	3
max_depth	3 - 8	7
features_classA	-	5
features_classB	-	19
features_classC	15 - 40	24

μετρήσεις για μικρά μεγέθη μηνυμάτων, όπου παρατηρήσαμε μεγάλες διακυμάνσεις του χρόνου επικοινωνίας μεταξύ διαφορετικών εκτελέσεων και διαφορετικών κατανομών κόμβων. Οι λεπτομέρειες του συνόλου εκπαίδευσης, που αποτελείται από 9210 σημεία, δίνονται στον Πίνακα 4.5. Η ενίσχυση του συνόλου εκπαίδευσης ήταν κρίσιμη για την εκπαίδευση του μοντέλου της κατηγορίας Γ, αφού μπορούμε να συλλέξουμε διαφορετικές τιμές των χαρακτηριστικών της κατηγορίας Γ μόνο με πολλαπλές εκτελέσεις σε διαφορετικές κατανομές κόμβων.

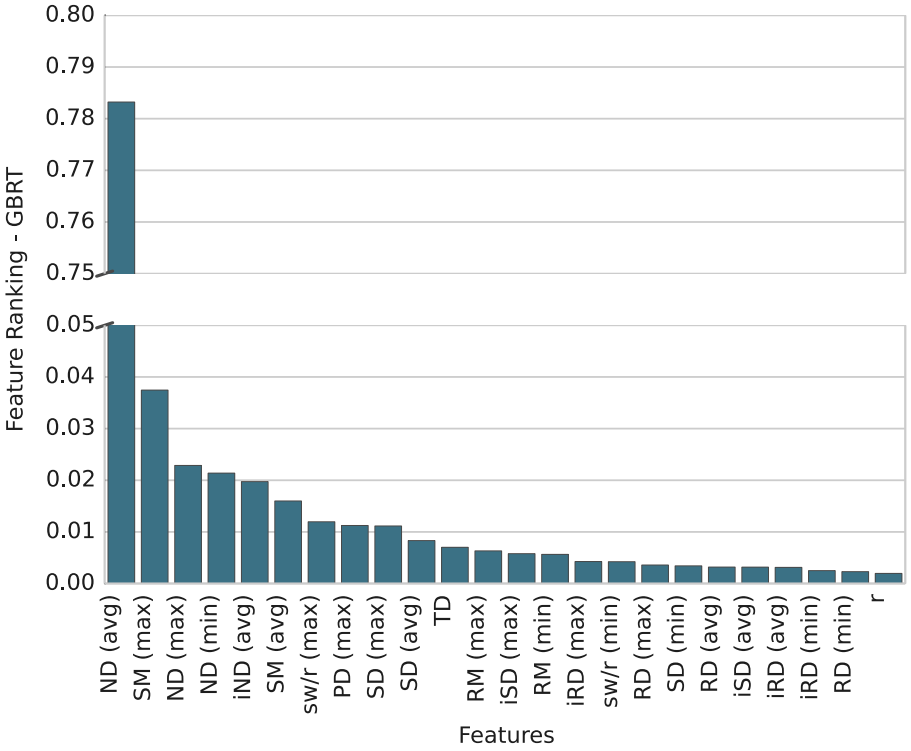
Για να βρούμε το βέλτιστο αριθμό χαρακτηριστικών και τις βέλτιστες ρυθμίσεις για τη μέθοδο GBRT, ώστε να καταλήξουμε στο βέλτιστο μοντέλο, δοκιμάσαμε τη μέθοδο με διαφορετικές τιμές για τις διάφορες παραμέτρους της και επιλέξαμε το μοντέλο που παρουσιάζει την καλύτερη προσαρμογή. Το εύρος των τιμών των παραμέτρων που δοκιμάσαμε, καθώς και οι τιμές που επιλέξαμε, παρουσιάζονται στον Πίνακα 4.6, μαζί με τον αριθμό των χαρακτηριστικών (*features-classX*) που χρησιμοποιεί το μοντέλο κάθε κατηγορίας. Τα

τελικά μοντέλα προκύπτουν από την εκπαίδευση της μεθόδου GBRT, με τις συγκεκριμένες τιμές των παραμέτρων και τα συγκεκριμένα χαρακτηριστικά, με το ενισχυμένο σύνολο εκπαίδευσης.



(i) GBRT-Class A

(ii) GBRT-Class B



(iii) GBRT-Class C

Σχήμα 4.3: Κατάταξη χαρακτηριστικών για τα μοντέλα GBRT στο Vilje.

Η μέθοδος GBRT δεν παρέχει κλειστή αναλυτική μορφή για το χρόνο επι-

κοινωνίας, ωστόσο η βιβλιοθήκη *scikit-learn* παρέχει μία μέθοδο για την κατάταξη των χαρακτηριστικών, που βασίζεται στην επιρροή τους στην έξοδο, σε κλίμακα από 0 έως 1, που μας επιτρέπει να εξάγουμε χρήσιμα συμπεράσματα για τα φαινόμενα που επηρεάζουν το χρόνο επικοινωνίας. Είναι σημαντικό να αποσαφηνίσουμε ότι αυτή η κατάταξη σχετίζεται με τη συγκεκριμένη μέθοδο και το συγκεκριμένο σύνολο εκπαίδευσης και δεν αποτελεί ακριβή μέτρηση της συγγένειας των διάφορων χαρακτηριστικών με το χρόνο επικοινωνίας: σημαντικά χαρακτηριστικά μπορούν να εμφανιστούν χαμηλά στην κατάταξη, ή καθόλου, ενώ χαρακτηριστικά που βρίσκονται υψηλά στην κατάταξη συνήθως χρησιμοποιούνται για τη διαίρεση του συνόλου εκπαίδευσης σε υψηλά επίπεδα του δέντρου απόφασης, αλλά δε θα απέδιδαν ακριβείς προβλέψεις αν δε συνδυάζονταν με τα υπόλοιπα χαρακτηριστικά. Παρόλα αυτά, η επισκόπηση των επιλεγμένων χαρακτηριστικών (βλ. Σχήμα 4.3) μπορεί αργότερα να εξηγήσει την προβλεπτική ικανότητα κάθε μοντέλου. Για το μοντέλο *GBRT-Class A*, το χαρακτηριστικό με τον υψηλότερο βαθμό είναι τα δεδομένα ανά διεργασία (*PD*), που ακολουθούνται από το μέγεθος της κατανομής, όπως το χαρακτηρίζουν οι κόμβοι *n* και οι διεργασίες ανά κόμβο *ppn*. Η μέθοδος δεν αγνοεί τα επιπλέον χαρακτηριστικά που ανήκουν στις κατηγορίες Β και Γ, όταν δίνονται στην είσοδο, όπως καταδεικνύει η βαθμονόμηση των χαρακτηριστικών των μοντέλων *GBRT-Class B* και *GBRT-Class C*, όπου φαίνεται πως η μέθοδος GBRT είναι σε θέση να αναγνωρίσει τη συσχέτισή τους με το χρόνο επικοινωνίας. Αυτή η παρατήρηση επιβεβαιώνει την υπόθεσή μας ότι περισσότερη πληροφορία, σχετική με την απεικόνιση των διεργασιών και την αρχιτεκτονική του συστήματος, μπορεί να ενισχύσει την προβλεπτική ικανότητα της μεθόδου, όπως επίσης θα δείξουμε αργότερα. Το επικρατέστερο χαρακτηριστικό τόσο για το μοντέλο *GBRT-Class B*, όσο και για το μοντέλο *GBRT-Class C* είναι τα εξερχόμενα δεδομένα ανά κόμβο *ND*, με βαθμό μεγαλύτερο από 0.7. Επιβεβαιώνοντας τις παρατηρήσεις μας από την κατασκευή μοντέλων με τεχνικές στατιστικής μάθησης, τα χαρακτηριστικά που περιγράφουν ροές δεδομένων στο επίπεδο του κόμβου και του διακόπτη εμφανίζονται υψηλά στην κατάταξη των χαρακτηριστικών από το μοντέλο *GBRT-Class C*, επηρεάζοντας πρωταρχικά τη διαδικασία μοντελοποίησης. Με ενδιαφέρον παρατηρούμε ότι η κατανομή των δεδομένων σε μηνύματα είναι επίσης σημαντική για την πρόβλεψη του χρόνου επικοινωνίας, όπως καταδεικνύεται από την κατάταξη χαρακτηριστικών που αφορούν σε πλήθος μηνυμάτων.

Πίνακας 4.7: Λεπτομέρειες του συνόλου ελέγχου στο σύστημα Vilje

	Vilje		
Σχήμα επικοινωνίας	Halo-3D	Halo-4D	LULESH
Μέγεθος προβλήματος	128^3, 256^3, 512^3, 1024^3, 2048^3	128^4, 256^4	240^3, 480^3
Επαναλήψεις	256	256	100
n	16-512	16-512	8-225
ppn	1-16	1-16	1-16
#εκτελέσεις	3	2	1
#σημεία	648		

4.4 Πειραματική αποτίμηση

4.4.1 Σχήματα επικοινωνίας

Πειραματιστήκαμε με δύο σχήματα επικοινωνίας που συναντώνται συχνά σε εφαρμογές μεγάλης κλίμακας, και συγκεκριμένα το σχήμα επικοινωνίας *Halo-3D*, που εμφανίζεται στον επιλυτή Jacobi-3Δ 7 σημείων, και το σχήμα επικοινωνίας *Halo-4D*, που αποτελεί το κύριο σχήμα επικοινωνίας σε προσομοιώσεις κβαντομηχανικής (Lattice Quantum Chormodynamics). Το σχήμα επικοινωνίας *Halo-3D* (*Halo-4D*) υλοποιείται με MPI ως εξής: οι διεργασίες οργανώνονται σε ένα εικονικό καρτεσιανό πλέγμα 3 διαστάσεων (4 διαστάσεων) και το αρχικό τρισδιάστατο (τετραδιάστατο) χωρίο αποσυντίθεται σε μικρότερα 3Δ (4Δ) υποχωρία. Κάθε διεργασία ανταλλάσσει μία όψη δύο (τριών) διαστάσεων με καθεμία από τις έξι (οκτώ) γειτονικές της διεργασίες.

Εφαρμόσαμε επίσης τη μεθοδολογία μας στις point-to-point φάσεις επικοινωνίας της εφαρμογής LULESH [Karlin et al., 2012], που αναπτύχθηκε από το εργαστήριο LLNL ως αντιπροσωπευτική εφαρμογή προσομοιώσεων υδροδυναμικής σε αδόμητα πλέγματα. Η εφαρμογή LULESH στηρίζεται σε υπολογισμούς τύπου stencil στις τρεις διαστάσεις και εμφανίζει τρία σχήματα point-to-point επικοινωνίας σε κάθε βήμα της προσομοίωσης. Το πρώτα σχήμα, που ονομάζουμε *LULESH-1*, εμφανίζει επικοινωνία τύπου halo (ανταλλαγών) στις τρεις διαστάσεις με 26 γειτονικές διεργασίες (26-point), όπου οι διεργασίες ανταλλάσσουν διανύσματα δυνάμεων. Στο δεύτερο σχήμα, που ονομάζουμε *LULESH-2*, εμφανίζεται επικοινωνία όμοια με αυτή του *Halo-3D*, όπου οι διεργασίες ανταλλάσσουν τιμές του ιξώδους. Στο τρίτο σχήμα, που ονομάζουμε *LULESH-3*, εμφανίζεται επικοινωνία τύπου wavefront (οι διεργασίες λαμβάνουν μηνύματα από κάποιες γειτονικές τους και στέλνουν μηνύματα σε κάποιες άλλες γειτονικές τους) με 26 γειτονικές διεργασίες, για την επικοινωνία των θέσεων και ταχυτήτων.

Οι φάσεις επικοινωνίας που επιλέξαμε εκθέτουν τέσσερα διαφορετικά, αλλά

πολύ συχνά εμφανιζόμενα, σχήματα επικοινωνίας γειτόνων, με διαφορετικά χαρακτηριστικά επικοινωνίας. Στην περίπτωση των *Halo-3D* και *LULESH-2*, η επικοινωνία αποτελείται από έξι μηνύματα ίδιας τάξης μεγέθους, τις έξι δισδιάστατες όψεις του τρισδιάστατου υποχωρίου. Αυτό το απλό σχήμα επικοινωνίας εμφανίζεται συχνά σε εφαρμογές. Το συναντάμε επίσης στις αντιπροσωπευτικές εφαρμογές του LLNL AMG2013[1] και Kripke[2], στην εφαρμογή CoMD[3] της σουίτας ExMatEx, στις μικρο-εφαρμογές CloverLeaf3D, miniAMR και miniGhost της σουίτας Mantevo[4], στις εφαρμογές MG και SP των NAS[5] Parallel Benchmarks και στην αντιπροσωπευτική εφαρμογή miniGMG[6] της σουίτας ExaCT από το εργαστήριο LBL. Στην περίπτωση του *Halo-4D*, η επικοινωνία είναι πιο πυκνή και αποτελείται από 8 μηνύματα ίδιας τάξης μεγέθους, τις 8 τρισδιάστατες όψεις του τετραδιάστατου υποχωρίου. Αυτό το σχήμα επικοινωνίας εμφανίζεται σε όλους τους κώδικες που αφορούν κβαντική χρωμοδυναμική, όπως οι tmLQCD[7], MILC[8] και τα QCD benchmarks του PRACE[9]. Οι διεργασίες στο σχήμα *LULESH-1* ανταλλάσσουν 26 μηνύματα τριών διαφορετικών τάξεων μεγέθους, που προκύπτουν από τη γεωμετρία του τρισδιάστατου υποχωρίου: 6 δισδιάστατες όψεις, 12 μονοδιάστατες ακμές και 8 γωνίες του ενός στοιχείου. Το ίδιο σχήμα εμφανίζεται σε πολυάριθμες παράλληλες εφαρμογές μεγάλης κλίμακας, όπως οι αντιπροσωπευτικές εφαρμογές Lassen[10] και AMG2013 από το εργαστήριο LLNL, η εφαρμογή NekBone[11] της σουίτας CESAR του εργαστηρίου ANL, καθώς και οι εφαρμογές HPCG, miniFE και miniGhost της σουίτας Mantevo. Στο σχήμα *LULESH-3* εμφανίζεται επικοινωνία τύπου wavefront, όπου οι διεργασίες ανταλλάσσουν μηνύματα τριών διαφορετικών τάξεων μεγέθους (όψεις, ακμές, γωνίες), όμως η επικοινωνία συντελείται διαγώνια: κάθε διεργασία στέλνει μόνο 13 μηνύματα στις 13 γειτονικές διεργασίες και λαμβάνει 13 μηνύματα από τις εναπομείνασες 13 γειτονικές διεργασίες.

Προβλέπουμε το χρόνο επικοινωνίας για τα σχήματα *Halo-3D* και *Halo-4D* και την εφαρμογή LULESH, για πολλά μεγέθη προβλημάτων και ποικίλες κατανομές πόρων, καθώς και πολλαπλές εκτελέσεις, ώστε να ελέγξουμε την ικανότητα των χαρακτηριστικών της κατηγορίας Γ να περιγράψουν τα φαινόμενα των διαφορετικών σχημάτων των κατανομών κόμβων. Οι λεπτομέρειες

[1] https://codesign.llnl.gov/amg2013.php
[2] https://codesign.llnl.gov/kripke.php
[3] http://www.exmatex.org/comd.html
[4] https://mantevo.org/
[5] https://www.nas.nasa.gov/publications/npb.html
[6] https://ccse.lbl.gov/ExaCT/index.html
[7] https://github.com/etmc/tmLQCD
[8] http://physics.indiana.edu/~sg/milc.html
[9] http://www.prace-ri.eu/ueabs/#QCD
[10] https://codesign.llnl.gov/lassen.php
[11] https://cesar.mcs.anl.gov/content/software/thermal_hydraulics

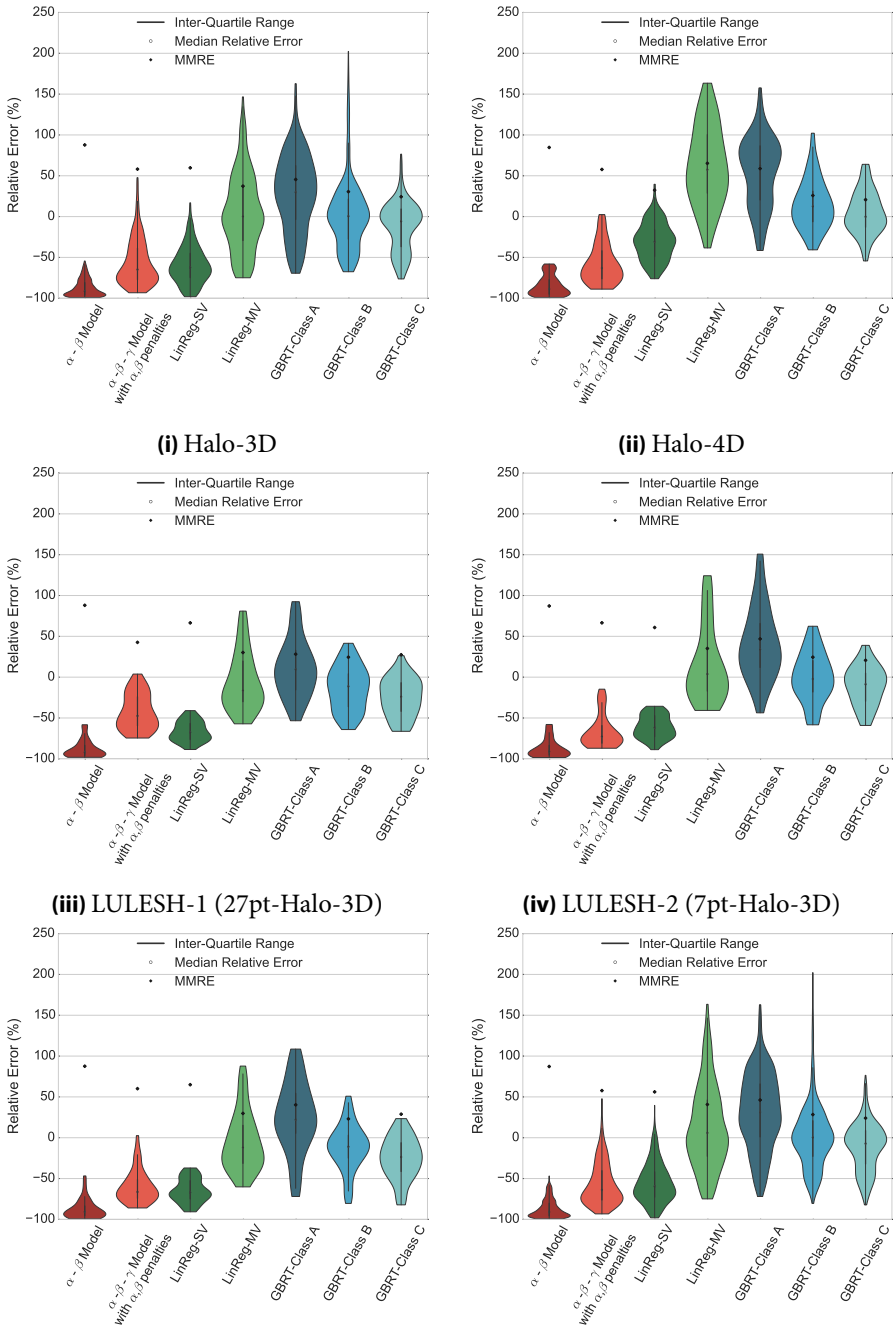
Πίνακας 4.8: Μετρηθείσες παράμετροι για τα αναλυτικά και ημι-εμπειρικά μοντέλα στο σύστημα Vilje

Παράμετρος	Τιμή
α	0.305 us
β	0.215 ns
γ	0.257 us

του συνόλου ελέγχου για τα δύο συστήματα εμφανίζονται στον Πίνακα 4.7. Ο σχεδιασμός της εφαρμογής LULESH απαιτεί το πλήθος των διεργασιών να είναι δύναμη του 3, συνεπώς όλες οι κατανομές των πόρων για τις εκτελέσεις της εφαρμογής LULESH δεσμεύονται από αυτή την απαίτηση.

4.4.2 Σύγκριση των μοντέλων

Για να αποτιμήσουμε την προβλεπτική ικανότητα των μοντέλων μας, χρησιμοποιούμε τις μετρικές που ορίσαμε στην Ενότητα 3.2. Συγκρίνουμε την ακρίβεια πρόβλεψης και την καλή προσαρμογή των δύο μοντέλων που κατασκευάσαμε με στατιστική μάθηση, δηλαδή των *LinReg-SV* και *LinReg-MV*, και των τριών μοντέλων που κατασκευάσαμε με μηχανική μάθηση, δηλαδή των *GBRT-Class A*, *GBRT-Class B* και *GBRT-Class C*. Συγκρίνουμε επίσης την ακρίβεια πρόβλεψης και την καλή προσαρμογή του αναλυτικού μοντέλου $\alpha - \beta$, καθώς και του ημι-εμπειρικού μοντέλου $\alpha - \beta - \gamma$ με ποινές στις παραμέτρους α και β, που συμβολίζουμε ως μοντέλο $\alpha_p - \beta_p - \gamma$. Οι τιμές που μετρήσαμε για τις παραμέτρους α, β και γ δίνονται στον Πίνακα 4.8. Αξίζει να σημειώσουμε πως η ποινή στην παράμετρο α αναφέρεται στην επίδραση των πολλών διεργασιών ανά κόμβο, ενώ η ποινή στην παράμετρο β αφορά το περιορισμένο εύρος ζώνης και τον ανταγωνισμό στο δίκτυο.



(i) Halo-3D **(ii)** Halo-4D

(iii) LULESH-1 (27pt-Halo-3D) **(iv)** LULESH-2 (7pt-Halo-3D)

(v) LULESH-3 (27pt-Wave-3D) **(vi)** Συνολικά

Σχήμα 4.4: Σύγκριση των μοντέλων στο σύστημα Vilje.

Πίνακας 4.9: Σύγκριση των μοντέλων με μετρικές ακρίβειας και καλής προσαρμογής στο σύστημα Vilje

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.00	-0.094	0.800
$\alpha_p - \beta_p - \gamma$ *Model*	12.667	0.130	0.898
LinReg-SV	40.88	0.801	0.915
LinReg-MV	41.33	0.667	0.922
GBRT-Class A	29.11	0.730	0.926
GBRT-Class B	50.89	0.774	0.936
GBRT-Class C	62.44	0.785	0.939

(i) Halo-3D

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.00	-0.450	0.742
$\alpha_p - \beta_p - \gamma$ *Model*	12.50	0.004	0.794
LinReg-SV	28.33	0.363	0.893
LinReg-MV	20.00	0.590	0.931
GBRT-Class A	24.16	0.458	0.912
GBRT-Class B	59.17	0.949	0.937
GBRT-Class C	67.50	0.938	0.935

(ii) Halo-4D

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.0	-2.042	0.585
$\alpha_p - \beta_p - \gamma$ *Model*	26.92	-0.238	0.790
LinReg-SV	42.31	0.568	0.873
LinReg-MV	48.08	0.579	0.849
GBRT-Class A	53.85	0.384	0.882
GBRT-Class B	57.69	0.829	0.893
GBRT-Class C	50.0	0.738	0.875

(iii) LULESH-1

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.0	-1.445	0.386
$\alpha_p - \beta_p - \gamma$ *Model*	9.62	-0.699	0.636
LinReg-SV	40.38	0.669	0.765
LinReg-MV	48.08	0.498	0.860
GBRT-Class A	36.54	-0.219	0.878
GBRT-Class B	61.54	0.863	0.892
GBRT-Class C	65.38	0.891	0.900

(iv) LULESH-2

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.0	-2.041	0.535
$\alpha_p - \beta_p - \gamma$ *Model*	7.69	-0.735	0.799
LinReg-SV	40.38	0.506	0.811
LinReg-MV	42.31	0.605	0.839
GBRT-Class A	42.31	-0.072	0.835
GBRT-Class B	65.38	0.708	0.842
GBRT-Class C	51.92	0.469	0.817

(v) LULESH-3

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.0	0.030	0.825
$\alpha_p - \beta_p - \gamma$ *Model*	13.09	0.309	0.896
LinReg-SV	38.84	0.635	0.927
LinReg-MV	38.84	0.721	0.928
GBRT-Class A	31.54	0.667	0.937
GBRT-Class B	54.55	0.926	0.942
GBRT-Class C	61.85	0.922	0.942

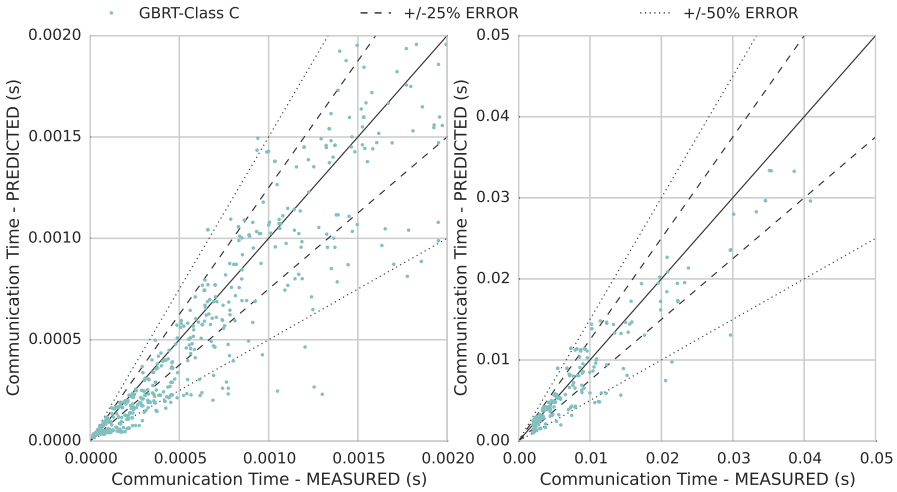
(vi) Συνολικά

Το Σχήμα 4.4 δείχνει την κατανομή των σχετικών σφαλμάτων των προβλέψεων, για κάθε μοντέλο σε κάθε σχήμα επικοινωνίας, καθώς και συνολικά (βλ. Σχήμα 4.4vi), στη μορφή διαγραμμάτων βιολιού, ενώ ο Πίνακας 4.9 συνοψίζει τις τιμές των μετρικών $Pred_{0.25}$, R^2 και RCC. Μπορούμε, επομένως, να κάνουμε τις ακόλουθες παρατηρήσεις: πρώτον, το στοιχειώδες αναλυτικό μοντέλο $\alpha-\beta$ προβλέπει σημαντικά μικρότερο χρόνο επικοινωνίας από τον πραγματικό σε όλες τις περιπτώσεις, αφού στηρίζεται μόνο στο χρόνο απόκρισης και το εύρος ζώνης του δικτύου για τις προβλέψεις, και παρουσιάζει κακή προσαρμογή. Το ημι-εμπειρικό μοντέλο $\alpha_p - \beta_p - \gamma$ βελτιώνει τις προβλέψεις σε σχέση με

το αναλυτικό, ωστόσο και αυτό προβλέπει μικρότερο χρόνο επικοινωνίας από τον πραγματικό σε όλες τις περιπτώσεις. Η ανεπάρκεια αυτού του μοντέλου οφείλεται στις ιδιαιτερότητες της τοπολογίας ενισχυμένου υπερκύβου, καθώς και στην πολιτική δέσμευσης πόρων στο Vilje. Η ποινή στην παράμετρο β στοχεύει να ενσωματώσει τον ανταγωνισμό που οφείλεται στο διαμοιρασμό των συνδέσεων και στο μέγιστο δυνατό εύρος ζώνης ανά κόμβο. Ωστόσο, στο Vilje, τα φαινόμενα ανταγωνισμού αναπτύσσονται κυρίως στο επίπεδο του διακόπτη, αφού οι διακόπτες συχνά είναι μοιραζόμενοι μεταξύ εφαρμογών, ενώ οι συνδέσεις της τοπολογίας του υπερκύβου δε χρησιμοποιούνται πλήρως, εξαιτίας της πολιτικής δρομολόγησης με βάση τη διάσταση (dimension-order routing).

Αναφορικά με τα δικά μας μοντέλα, τα δύο μοντέλα που κατασκευάστηκαν με στατιστική μάθηση, δηλαδή τα *LinReg-SV* και *LinReg-MV*, εμφανίζουν αντίστοιχη ακρίβεια και καλή προσαρμογή σε όλα τα σχήματα και συνολικά, επιτυγχάνοντας μέτρια ακρίβεια, όπως καταδεικνύουν οι τιμές των μετρικών $Pred_{0.25}$ και R^2, αλλά και τα μεγάλα σχετικά σφάλματα για τα σχήματα επικοινωνίας *Halo-3D*, *Halo-4D* και *LULESH-2*. Ωστόσο, επιτυγχάνουν υψηλό βαθμό στη μετρική *RCC*, επομένως έχουν τη δυνατότητα να διακρίνουν μεταξύ διαφορετικών κατανομών και ρυθμίσεων της επικοινωνίας. Εμφανώς, τα δύο μοντέλα που επιτυγχάνουν τα καλύτερα αποτελέσματα είναι τα μοντέλα *GBRT-Class B* και *GBRT-Class C*. Τα δύο μοντέλα επιτυγχάνουν περίπου ίσο βαθμό στις μετρικές R^2 και *RCC*, ωστόσο το μοντέλο *GBRT-Class B* συχνά προβλέπει μεγαλύτερο χρόνο επικοινωνίας από τον πραγματικό, καταλήγοντας σε μεγαλύτερο εύρος σχετικών σφαλμάτων, ενώ το μοντέλο *GBRT-Class C* επιτυγχάνει καλύτερο εύρος για τα σχετικά σφάλματα και καλύτερο βαθμό στη μετρική $Pred_{0.25}$ συνολικά. Τέλος, το μοντέλο *GBRT-Class A* προβλέπει μεγαλύτερους χρόνους επικοινωνίας από τους πραγματικούς, για όλα τα σχήματα επικοινωνίας. Αυτή η συμπεριφορά αποδεικνύει ότι τα χαρακτηριστικά της κατηγορίας Α, σε συνδυασμό με το γενικό μας τρόπο για τη συλλογή του συνόλου εκπαίδευσης, δεν επαρκούν για τη σκιαγράφηση του σχήματος κίνησης δεδομένων και για την παραγωγή ακριβών προβλέψεων στο Vilje, όπου η επίδοση της επικοινωνίας επηρεάζεται σημαντικά από φαινόμενα στο επίπεδο των κόμβων και των διακοπτών, και όπου η κατανομή των δεδομένων σε διαφορετικά μηνύματα είναι σημαντική, όπως κατέδειξε και η κατάταξη των χαρακτηριστικών για τα μοντέλα των κατηγοριών Β και Γ. Η ακρίβεια του μοντέλου της κατηγορίας Α μπορεί να ενισχυθεί με την αύξηση του συνόλου εκπαίδευσης με δεδομένα από ακανόνιστα σχήματα επικοινωνίας, ώστε το μοντέλο να ενσωματώσει ελάχιστες και μέσες τιμές για τα χαρακτηριστικά *l*, *PD* και *PM*.

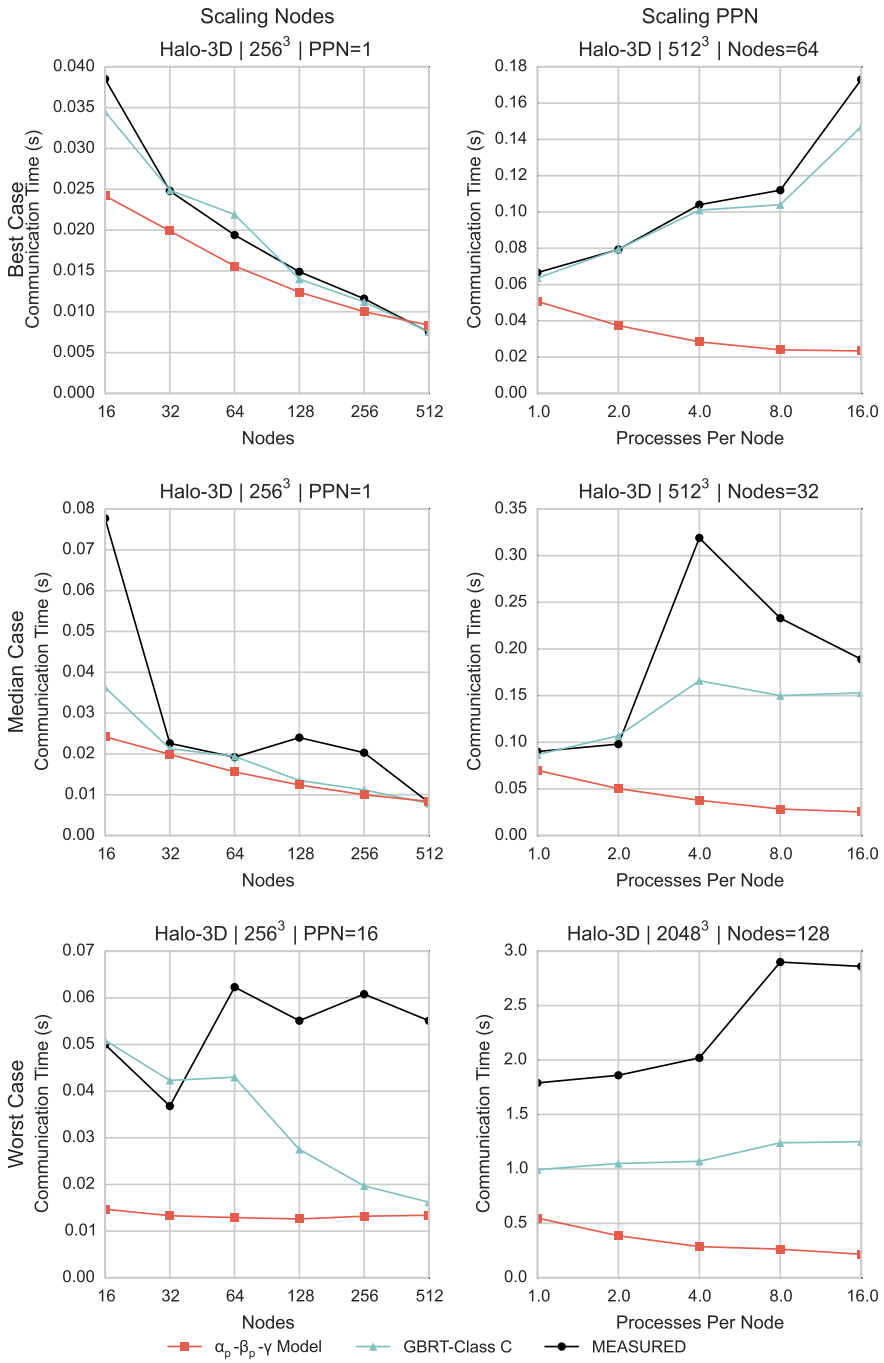
4.4.3 Λεπτομερής αποτίμηση



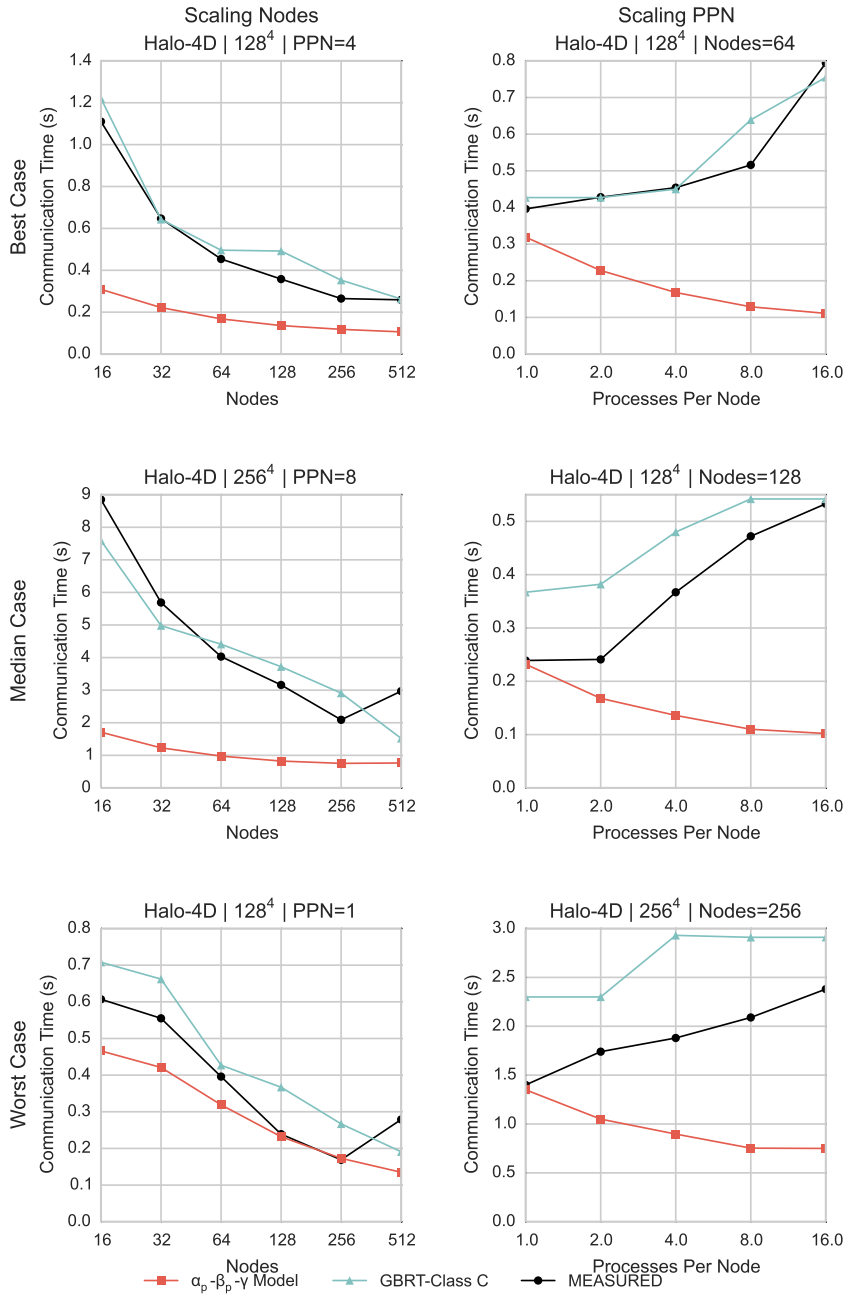
Σχήμα 4.5: Διαγράμματα διασποράς για τις προβλέψεις του μοντέλου *GBRT-Class C* στο Vilje. Ο χρόνος επικοινωνίας έχει κανονικοποιηθεί σε μία επανάληψη.

Με βάση τη σύγκριση των μοντέλων στις προηγούμενες παραγράφους, προάγουμε το μοντέλο *GBRT-Class C* ως το βέλτιστο μοντέλο για το σύστημα Vilje, αφού επιτυγχάνει καλύτερο εύρος σχετικών σφαλμάτων από το μοντέλο *GBRT-Class B*, μολονότι το δεύτερο επιτυγχάνει αντίστοιχα υψηλούς βαθμούς σε όλες τις μετρικές και σε ορισμένες περιπτώσεις ξεπερνά σε ακρίβεια το πρώτο. Θεωρούμε το μοντέλο *GBRT-Class C* ως το πιο χρήσιμο, αφού ο χρόνος πρόβλεψής του είναι ακριβώς πριν την εκτέλεση της εφαρμογής (just-in-time), η ακρίβειά του είναι υψηλή, με βαθμό 61.43% στη μετρική $Pred_{0.25}$ και η προσαρμογή του είναι άριστη, με βαθμούς 0.926 στη μετρική R^2 και 0.942 στη μετρική *RCC*. Επιπλέον, το μοντέλο *GBRT-Class C* επιτυγχάνει τη μείωση της μετρικής *MMRE* συνολικά σε 23.98%, συγκριτικά με το 44.64% του ημι-εμπειρικού μοντέλου $\alpha_p - \beta_p - \gamma$. Για τους παραπάνω λόγους, αναλύουμε περαιτέρω την επίδοση των προβλέψεων του συγκεκριμένου μοντέλου. Το Σχήμα 4.5 παρουσιάζει τη σύγκριση μεταξύ του μετρημένου χρόνου επικοινωνίας και των αντίστοιχων προβλέψεων με το μοντέλο *GBRT-Class C*, για όλα τα σημεία στο σύνολο ελέγχου, κανονικοποιημένων σε μία επανάληψη, σε μορφή διαγραμμάτων διασποράς. Οι μετρήσεις και προβλέψεις παρουσιάζονται σε δύο υποσύνολα, με βάση το χρόνο επικοινωνίας, για λόγους ευκρίνειας. Η πλειοψηφία των προβλέψεων εμφανίζει σφάλματα στο εύρος $\pm 25\%$, ενώ το 87.6% των προβλέψεων εμφανίζουν σφάλματα στο εύρος $\pm 50\%$. Ένα σύνολο 60 σημείων, με χρόνο επικοινωνίας μικρότερο από 1.5 ms, βρίσκεται κάτω από τη γραμμή

σφαλμάτων του -50%. Αυτά τα σημεία αντιστοιχούν σε κατανομές με μικρά μεγέθη μηνυμάτων και υψηλό αριθμό πυρήνων, όπου οι μετρήσεις του χρόνου επικοινωνίας εμφανίζουν θόρυβο, κυρίως εξαιτίας φαινομένων εντός του κόμβου, και η επικοινωνία εμφανίζει υψηλές διακυμάνσεις στο χρόνο, εξ ου και το μοντέλο μας τείνει να προβλέπει μικρότερες τιμές για τις συγκεκριμένες κατανομές.



Σχήμα 4.6: Προβλέψεις για το σχήμα επικοινωνίας Halo-3D με το μοντέλο *GBRT-Class C* στο Vilje.

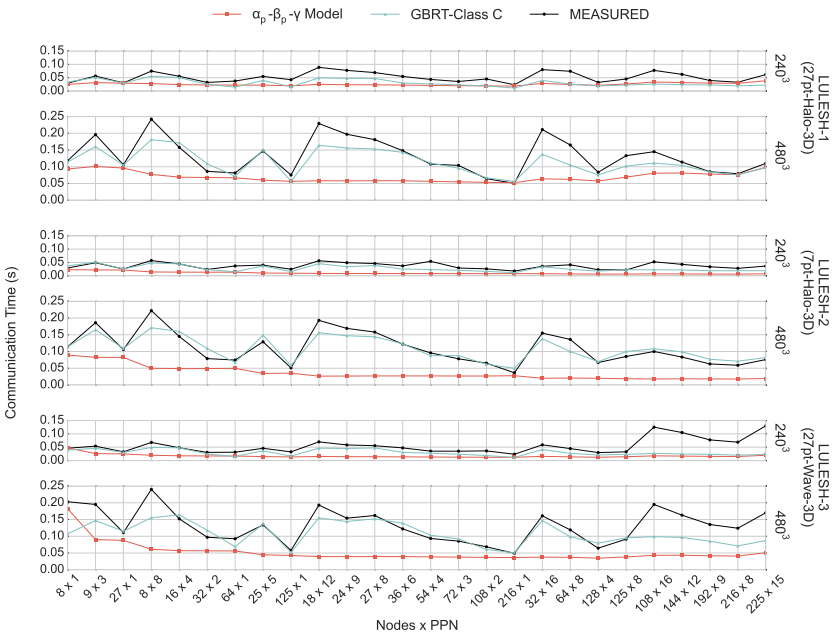


Σχήμα 4.7: Προβλέψεις για το σχήμα επικοινωνίας Halo-4D με το μοντέλο *GBRT-Class C* στο Vilje.

Ακολούθως, αποτιμούμε το μοντέλο *GBRT-Class C* στα διάφορα σχήματα επικοινωνίας. Για τα σχήματα *Halo-3D* και *Halo-4D*, το σύνολο ελέγχου μας περιλαμβάνει μία πληθώρα κατανομών, μεγεθών προβλημάτων και διακριτών εκτελέσεων σε διαφορετικές κατανομές κόμβων. Για να ελέγξουμε την ικανότητα του μοντέλου μας να προβλέπει την κλιμάκωση του χρόνου επικοινωνίας όταν αυξάνει ο αριθμός των κόμβων ή των διεργασιών ανά κόμβο, χρησιμοποιούμε το γινόμενο των μετρικών R^2 και *RCC*, $R^2 \times RCC$, για να βαθμονομήσουμε τα διάφορα υποσύνολα των μετρήσεων σε μία κλίμακα από 0 έως 1, όπου το 1 αντιστοιχεί στο υποσύνολο με τις καλύτερες προβλέψεις και το 0 στο υποσύνολο με τις χειρότερες προβλέψεις. Παρουσιάζουμε την καλύτερη, τη διάμεση και τη χειρότερη περίπτωση προβλέψεων στα διάφορα υποσύνολα για το σχήμα επικοινωνίας *Halo-3D* στο Σχήμα 4.6 και για το σχήμα επικοινωνίας *Halo-4D* στο Σχήμα 4.7. Παραπέμπουμε τον αναγνώστη στο Παράρτημα Α για το πλήρες σύνολο των προβλέψεων. Επιπλέον, συγκρίνουμε τις προβλέψεις μας με τις προβλέψεις του ημι-εμπειρικού μοντέλου $\alpha_p - \beta_p - \gamma$.

Για το σχήμα επικοινωνίας *Halo-3D*, οι προβλέψεις μας είναι άριστες στη βέλτιστη και στη διάμεση περίπτωση, τόσο όταν κλιμακώνουμε τον αριθμό των κόμβων, όσο και όταν κλιμακώνουμε τον αριθμό των διεργασιών ανά κόμβο. Ας σημειωθεί πως η βέλτιστη και διάμεση περίπτωση, όταν κλιμακώνει ο αριθμός των κόμβων, αφορά το ίδιο μέγεθος προβλήματος (256^3) και τον ίδιο αριθμό διεργασιών ανά κόμβο (PPN=1), αλλά το κάθε σύνολο σημείων προκύπτει από διαφορετικές εκτελέσεις σε διαφορετικές κατανομές κόμβων. Η περίπτωση των χειρότερων προβλέψεων, όταν κλιμακώνει ο αριθμός των κόμβων, αφορά σε ένα μικρό μέγεθος προβλήματος με μικρά μεγέθη μηνυμάτων σε πλήρη κόμβο (PPN=16), όπου έχουμε παρατηρήσει υψηλές αυξήσεις του χρόνου επικοινωνίας, που δε συνάδουν με τα αποτελέσματά μας από τη συλλογή μετρήσεων αναφοράς. Αποδίδουμε το αποτέλεσμα αυτό σε φαινόμενα εντός του κόμβου (εξαιτίας της μοιραζόμενης κρυφής μνήμης και φαινομένων στη μνήμη), που εμφανίζονται λόγω της αυξημένης επικοινωνίας εντός του κόμβου. Το μοντέλο $\alpha_p - \beta_p - \gamma$ αποτυγχάνει να προβλέψει τη συμπεριφορά κλιμάκωσης της επικοινωνίας όταν αυξάνει ο αριθμός των διεργασιών ανά κόμβο, παρά την ποινή για τις πολλαπλές διεργασίες ανά κόμβο στην παράμετρο α. Παρομοίως, όταν κλιμακώνει το πλήθος των κόμβων, το ημι-εμπειρικό μοντέλο αποδίδει καλές προβλέψεις στην καλύτερη και διάμεση περίπτωση, όπου χρησιμοποιείται μόνο μία διεργασία ανά κόμβο, αλλά προβλέπει σημαντικά χαμηλότερους χρόνους επικοινωνίας στη χειρότερη περίπτωσή μας, όπου ο κόμβος είναι γεμάτος. Αναφορικά με το σχήμα επικοινωνίας *Halo-4D* στο Σχήμα 4.7, το μοντέλο μας *GBRT-Class C* προβλέπει το χρόνο επικοινωνίας με εξαιρετική ακρίβεια, ακόμα και στη χειρότερη περίπτωση, ωστόσο αποτυγχάνει να προβλέψει το σημείο όπου το σχήμα επικοινωνίας σταματά να κλιμακώνει (scalability break), στη διάμεση και χειρότερη περίπτωση, όταν κλιμακώνουμε τον

αριθμό των κόμβων. Η προβλεπτική ικανότητα του ημι-εμπειρικού μοντέλου είναι αντίστοιχη της περίπτωσης του σχήματος *Halo-3D*: το μοντέλο αποτυγχάνει να συλλάβει τον τρόπο κλιμάκωσης της επικοινωνίας με τον αριθμό των διεργασιών ανά κόμβο.



Σχήμα 4.8: Προβλέψεις για την εφαρμογή LULESH με το μοντέλο *GBRT-Class C* στο Vilje.

Στο Σχήμα 4.8, παρουσιάζουμε τις προβλέψεις του χρόνου επικοινωνίας για όλες τις ρυθμίσεις εκτέλεσης της εφαρμογής LULESH, ταξινομημένες με βάση τον αριθμό των πυρήνων. Για τα τρία σχήματα επικοινωνίας (*LULESH-1*, *LULESH-2* και *LULESH-3*) και τα δύο μεγέθη προβλημάτων (240^3 και 480^3), το μοντέλο μας *GBRT-Class C* προβλέπει το χρόνο επικοινωνίας με υψηλή ακρίβεια έως και τους 216 πυρήνες (216 x 1). Επιπλέον, το μοντέλο μας διακρίνει ικανοποιητικά διαφορετικές ρυθμίσεις εκτέλεσης έως και 512 πυρήνων (128 x 4). Στην περίπτωση του *LULESH-3*, παρατηρούμε υπο-εκτιμήσεις του χρόνου επικοινωνίας για περισσότερους από 512 πυρήνες. Πρέπει να σημειώσουμε ότι το σχήμα επικοινωνίας *LULESH-3* έχει παρόμοιο όγκο επικοινωνίας με το το σχήμα επικοινωνίας *LULESH-1*, που προκύπτει από διαφορετικό γράφο επικοινωνίας: οι διεργασίες στο σχήμα *LULESH-3* ανταλλάσσουν περίπου τα μισά μηνύματα σε σχέση με το σχήμα *LULESH-1*, με το διπλάσιο μέγεθος μηνυμάτων. Ωστόσο, ο χρόνος επικοινωνίας αυξάνει για το σχήμα

Πίνακας 4.10: Μέτωπα Παρέτο για ελαχιστοποίηση πυρήνων και χρόνου επικοινωνίας στο σύστημα Vilje

Σχήμα επικοινωνίας	Μέγεθος προβλήματος	Εκτέλεση	Ρυθμίσεις εκτέλεσης μετώπου Παρέτο	Προβλεφθείσες ρυθμίσεις εκτέλεσης	Matches (Ταιριάσματα)	Απόσταση (μέγιστη)
Halo-3D	128^3	#1	4	4	3	1
Halo-3D	128^3	#2	5	5	**5**	-
Halo-3D	128^3	#3	5	5	4	1
Halo-3D	256^3	#1	4	6	4	1
Halo-3D	256^3	#2	6	6	**6**	-
Halo-3D	256^3	#3	6	6	**6**	-
Halo-3D	512^3	#1	4	6	4	1
Halo-3D	512^3	#2	6	6	**6**	-
Halo-3D	512^3	#3	6	6	**6**	-
Halo-3D	1024^3	#1	4	6	4	2
Halo-3D	1024^3	#2	6	6	**6**	-
Halo-3D	1024^3	#3	6	6	**6**	-
Halo-3D	2048^3	#1	4	6	4	1
Halo-3D	2048^3	#2	6	6	**6**	-
Halo-3D	2048^3	#3	6	6	**6**	-
Halo-4D	128^4	#1	5	6	5	2
Halo-4D	128^4	#2	5	6	5	1
Halo-4D	256^4	#1	5	8	5	5
Halo-4D	256^4	#2	5	6	5	1
LULESH-1	240^3	#1	2	4	2	3
LULESH-1	480^3	#1	5	5	**5**	-
LULESH-2	240^3	#1	4	4	3	1
LULESH-2	480^3	#1	5	5	**5**	-
LULESH-3	240^3	#1	4	4	3	1
LULESH-3	480^3	#1	5	4	4	1

LULESH-3 για περισσότερους από 512 πυρήνες. Αυτή η συμπεριφορά είναι αναντίστοιχη των παρατηρήσεων στο σύνολο εκπαίδευσής μας, γεγονός που εξηγεί την ανεπάρκεια του μοντέλου μας στη συγκεκριμένη περίπτωση. Επιπλέον, αποτιμούμε τις προβλέψεις του μοντέλου $\alpha_p - \beta_p - \gamma$, το οποίο συστηματικά προβλέπει χαμηλότερους χρόνους επικοινωνίας από τους πραγματικούς, για όλες τις ρυθμίσεις εκτέλεσης με περισσότερες από μία διεργασίες ανά κόμβο.

4.4.4 Λήψη αποφάσεων με επίγνωση της επικοινωνίας

Η ακρίβεια ενός μοντέλου πρόβλεψης για το χρόνο επικοινωνίας αποτιμάται περαιτέρω με βάση την ικανότητά του να αποδίδει ακριβείς προβλέψεις σε ένα συγκεκριμένο πλαίσιο, ώστε να διευκολύνει διαδικασίες λήψης αποφάσεων. Επιλέγουμε το πιο ακριβές από τα μοντέλα μας, το μοντέλο *GBRT-Class C* και το χρησιμοποιούμε για την πρόβλεψης σημείων που είναι κρίσιμα σε σενάρια λήψης αποφάσεων που αφορούν τη βέλτιστη χρήση πόρων από την πλευρά του χρήστη. Οι χρήστες των υπερυπολογιστικών συστημάτων χρησιμοποιούν τα συστήματα μεγάλης κλίμακας με τρεις στόχους: α) να μεγιστοποιήσουν την επίδοση των εφαρμογών τους (δηλαδή να ελαχιστοποιήσουν το χρόνο εκτέλεσης), β) να ελαχιστοποιήσουν την κατανάλωση του προϋπολογισμού τους σε ώρες πυρήνων (core-hours) ή ώρες κόμβων (node-hours) και γ) να ελαχιστοποιήσουν το χρόνο αναμονής τους στις ουρές εργασιών

του συστήματος. Υποστηρίζουμε πως αυτοί οι τρεις στόχοι ανταποκρίνονται στις απαιτήσεις του χρήστη για ποιότητα υπηρεσιών (quality-of-service). Οι τρεις στόχοι είναι αντικρουόμενοι: οι χρήστες τείνουν να αιτούνται μεγαλύτερο αριθμό πυρήνων για την εκτέλεση των εφαρμογών τους, ώστε να ελαχιστοποιήσουν το χρόνο εκτέλεσης, προσδοκώντας πως η εφαρμογή τους θα κλιμακώσει. Ταυτόχρονα, τα αιτήματα για μεγαλύτερους αριθμούς πυρήνων έχουν ως αποτέλεσμα μεγαλύτερους χρόνους αναμονής, ενώ, αν η εφαρμογή δεν εμφανίζει την προσδοκώμενη κλιμακωσιμότητα, το αποτέλεσμα είναι η σπατάλη ωρών πυρήνων ή ωρών κόμβων. Επιπρόσθετα, οι χρήστες των υπερυπολογιστών αιτούνται ένα πλήθος κόμβων και διεργασιών ανά κόμβο, ή και νημάτων OpenMP, αν η εφαρμογή τους ακολουθεί το υβριδικό προγραμματιστικό μοντέλο MPI/OpenMP, ενώ δε έχουν στη διάθεσή τους κάποιον τρόπο να προβλέψουν ποια ρύθμιση εκτέλεσης θα αποδώσει το μικρότερο χρόνο εκτέλεσης. Στην πραγματικότητα, μία κοινή πρακτική των χρηστών είναι η εκτέλεση της εφαρμογής τους σε όλους τους δυνατούς συνδυασμούς κόμβων / διεργασιών ανά κόμβο / νημάτων, ώστε να επιλέξουν εκείνη τη ρύθμιση εκτέλεσης που αποδίδει καλύτερη επίδοση.

Αντιμετωπίζουμε αυτά τα προβλήματα από την πλευρά της επικοινωνίας χρησιμοποιώντας το μοντέλο μας *GBRT-Class C* για να προβλέψουμε τις βέλτιστες ρυθμίσεις κόμβων/διεργασιών ανά κόμβο, δηλαδή εκείνες τις ρυθμίσεις που ελαχιστοποιούν το χρόνο επικοινωνίας για ένα συγκεκριμένο πλήθος πυρήνων. Για να το επιτύχουμε, διατυπώνουμε το πρόβλημα ως πρόβλημα ταυτόχρονης ελαχιστοποίησης του αριθμού των πυρήνων και του χρόνου επικοινωνίας και χρησιμοποιούμε την έννοια της κατά Παρέτο βελτιστότητας για την επιλογή του κατά Παρέτο βέλτιστου συνόλου ρυθμίσεων. Αυτό το σύνολο είναι γνωστό ως μέτωπο Παρέτο και επισημαίνει το βέλτιστο σύνολο επιλογών, με την έννοια ότι ο χρήστης δε βρίσκεται ούτε σε καλύτερη ούτε σε χειρότερη θέση επιλέγοντας οποιοδήποτε από αυτά τα σημεία. Για ένα συγκεκριμένο σχήμα επικοινωνίας και ένα συγκεκριμένο μέγεθος προβλήματος, έστω δύο διαφορετικές ρυθμίσεις εκτέλεσης της επικοινωνίας, μία σε c πυρήνες με χρόνο επικοινωνίας t, δηλαδή (c, t), και μία σε c' πυρήνες με χρόνο επικοινωνίας t', δηλαδή (c', t'). Αν $c < c'$ και $t \leq t'$, τότε η πρώτη ρύθμιση εκτέλεσης είναι σίγουρα καλύτερη από τη δεύτερη, ή αλλιώς το πρώτο σημείο κυριαρχεί αυστηρά στο δεύτερο. Αν, όμως, $c < c'$ και $t > t'$, δεν μπορούμε να αποφασίσουμε αν η πρώτη ρύθμιση είναι καλύτερη από τη δεύτερη. Αυτό το σύνολο ρυθμίσεων, όπου καμία δεν κυριαρχεί επί της άλλης, αποτελεί το μέτωπο Παρέτο [Luke, 2009]. Το μη-κυριαρχούν (non-dominated) μέτωπο Παρέτο στην περίπτωσή μας σημειώνει επίσης εκείνες τις ρυθμίσεις εκτέλεσης που ελαχιστοποιούν την κατανάλωση ωρών πυρήνων (core-hours) λόγω της επικοινωνίας στο Vilje. Σημειώνουμε πως το σύστημα Vilje χρεώνει τους χρήστες με πολιτική core-hours.

Κατασκευάζουμε τα μέτωπα Παρέτο, βασιζόμενοι στις προβλέψεις που λαμβάνουμε με το μοντέλο *GBRT-Class C* για κάθε σχήμα επικοινωνίας, μέγεθος προβλήματος και εκτέλεση, και συγκρίνουμε τις προβλεφθείσες ρυθμίσεις εκτέλεσης με τα αντίστοιχα μέτωπα Παρέτο του πραγματικού χρόνου επικοινωνίας στο Vilje. Για να αξιολογήσουμε πόσο μακριά βρίσκεται μία προβλεφθείσα ρύθμιση εκτέλεσης από το πραγματικό μέτωπο Παρέτο, χρησιμοποιούμε την έννοια του βαθμού του μετώπου Παρέτο. Το πρώτο μέτωπο Παρέτο που υπολογίζουμε έχει βαθμό ίσο με το 0. Αν αφαιρέσουμε τα σημεία του μετώπου Παρέτο από το σύνολο των σημείων και υπολογίσουμε εκ νέου το μέτωπο Παρέτο, λαμβάνουμε ένα νέο, υπο-βέλτιστο μέτωπο Παρέτο με βαθμό ίσο με 1, ή αλλιώς απόσταση $d = 1$ από το αρχικό μέτωπο Παρέτο. Μπορούμε επαναληπτικά να αφαιρούμε σημεία για να λάβουμε το i-οστό μέτωπο Παρέτο με απόσταση $d = i$.Ένα σημείο στο i-οστό μέτωπο Παρέτο αντιστοιχεί στην i-οστή βέλτιστη ρύθμιση εκτέλεσης. Χρησιμοποιούμε την έννοια της απόστασης για να αποτιμήσουμε τις προβλεφθείσες ρυθμίσεις Παρέτο.

Το Σχήμα 4.9 δείχνει δύο παραδείγματα μετώπων Παρέτο στο Vilje. Κάθε γράφημα δείχνει τον μετρημένο χρόνο επικοινωνίας για όλες τις ρυθμίσεις εκτέλεσης για ένα συγκεκριμένο σχήμα επικοινωνίας και μέγεθος προβλήματος και μία συγκεκριμένη εκτέλεση. Τα γραφήματα απεικονίζουν το μέτωπο Παρέτο και το δεύτερο καλύτερο μέτωπο Παρέτο (με απόσταση ίση με 1). Το πρώτο γράφημα, στο Σχήμα 4.9i, καταδεικνύει την περίπτωση όπου οι προβλέψεις μας είναι 100% επιτυχείς: όλες οι προβλεφθείσες ρυθμίσεις εκτέλεσης συμπίπτουν με τις κατά Παρέτο βέλτιστες ρυθμίσεις εκτέλεσης, όπως έχουν υπολογιστεί με βάση τον πραγματικό χρόνο επικοινωνίας. Το δεύτερο γράφημα, στο Σχήμα 4.9ii, καταδεικνύει μία περίπτωση όπου το μοντέλο μας δεν προβλέπει με ακρίβεια το μέτωπο Παρέτο: 5 από τις 6 προβλεφθείσες ρυθμίσεις εκτέλεσης συμπίπτουν με τις ρυθμίσεις του μετώπου Παρέτο, ωστόσο, οι προβλεφθείσες ρυθμίσεις περιλαμβάνουν ένα επιπλέον σημείο για 512 πυρήνες, που αντιστοιχεί στη ρύθμιση εκτέλεσης 512×1 (κόμβοι $\times$ διεργασίες ανά κόμβο), ενώ στις μετρήσεις μας, ο χρόνος επικοινωνίας για τη συγκεκριμένη ρύθμιση εκτέλεσης είναι υψηλότερος από το χρόνο επικοινωνίας του προηγούμενου σημείου Παρέτο (256 κόμβοι $\times$ 1 διεργασία ανά κόμβο) και επομένως δεν περιλαμβάνεται στο μέτωπο Παρέτο. Η συγκεκριμένη ρύθμιση εκτέλεσης στην πραγματικότητα ανήκει στο μέτωπο Παρέτο με απόσταση ίση με 2 ($d = 2$). Ο Πίνακας 4.10 συνοψίζει τα αποτελέσματα των σημείων των μετώπων Παρέτο για τον πραγματικό χρόνο επικοινωνίας και τις προβλέψεις του χρόνου επικοινωνίας και το πλήθος των πυρήνων. Παραπέμπουμε τον αναγνώστη στο Παράρτημα Α για την αναλυτική αποτίμηση. Σημειώνουμε ως "Matches" (ταιριάσματα) το πλήθος των προβλεφθεισών ρυθμίσεων εκτέλεσης που συμπίπτουν με τις ρυθμίσεις του μετώπου Παρέτο. Στις περιπτώσεις που υπάρχουν προβλεφθείσες ρυθμίσεις εκτέλεσης που δε συμπίπτουν, αποδί-

δουμε σε κάθε τέτοια ρύθμιση έναν βαθμό d που αντιστοιχεί στην απόσταση του υπο-βέλτιστου μετώπου Παρέτο στο οποίο ανήκει. Αναφέρουμε τη μέγιστη απόσταση που καταγράφεται για αυτές τις ρυθμίσεις εκτέλεσης. Για τα 25 μέτωπα Παρέτο που κατασκευάσαμε, σε 11 περιπτώσεις, το προβλεφθέν και το πραγματικό μέτωπο Παρέτο συμπίπτουν. Για τις εναπομείνασες περιπτώσεις, τα μέτωπα Παρέτο διαφέρουν το πολύ κατά 3 σημεία (βλ. *Halo-4D* για μέγεθος προβλήματος 256^4 - εκτέλεση #1). Για 10 από τις υπόλοιπες περιπτώσεις, η μέγιστη απόσταση είναι 1. Επομένως, οι ρυθμίσεις εκτέλεσης που έχουν προβλεφθεί αλλά δεν ανήκουν στο πραγματικό μέτωπο Παρέτο, ανήκουν στο δεύτερο βέλτιστο μέτωπο Παρέτο. Για λόγους σύγκρισης, κατασκευάσαμε τα μέτωπα Παρέτο χρησιμοποιώντας προβλέψεις του μοντέλου $\alpha_p - \beta_p - \gamma$ και παρατηρήσαμε ότι για τα 25 μέτωπα Παρέτο, κανένα προβλεφθέν μέτωπο Παρέτο δε συμπίπτει με το πραγματικό.

Μία ενδιαφέρουσα παρατήρηση που προέκυψε από την κατασκευή των μετώπων Παρέτο είναι πως, στο Vilje, για οποιοδήποτε αριθμό πυρήνων και για όλα τα σχήματα επικοινωνίας, η βέλτιστη ρύθμιση εκτέλεσης δε χρησιμοποιεί ποτέ περισσότερες από 1 διεργασίες ανά κόμβο. Αν και οι ρυθμίσεις εκτέλεσης που εξεταζουμε είναι βέλτιστες μόνο για το χρόνο επικοινωνίας, μπορούμε να εξάγουμε δύο χρήσιμα συμπεράσματα για τη δέσμευση και χρήση πόρων στο Vilje. Πρώτον, εφόσον το σύστημα χρεώνει το χρήστη σε core-hours, αν οι υπολογισμοί της εφαρμογής δεν επηρεάζονται από άλλες εφαρμογές που εκτελούνται ταυτόχρονα, οι χρήστες θα πρέπει να εξετάζουν το ενδεχόμενο να επιλέγουν μία διεργασία ανά κόμβο, για να ελαχιστοποιούν την κατανάλωσή τους σε core-hours. Το σύστημα σε αυτή την περίπτωση μπορεί να τοποθετεί άλλες εφαρμογές στους εναπομείναντες επεξεργαστές του κόμβου. Δεύτερον, αν οι χρήστες επιθυμούν να επιλέξουν μία ρύθμιση εκτέλεσης που περιλαμβάνει νήματα OpenMP, καλό είναι να επιλέγουν 1 διεργασία ανά κόμβο για το MPI και να χρησιμοποιούν τους υπόλοιπους πυρήνες του κόμβου για νήματα του OpenMP, ή διαφορετικά, να επιλέγουν το μικρότερο δυνατό αριθμό διεργασιών ανά κόμβο, ανάλογα με την κλιμακωσιμότητα των υπολογισμών της εφαρμογής τους.

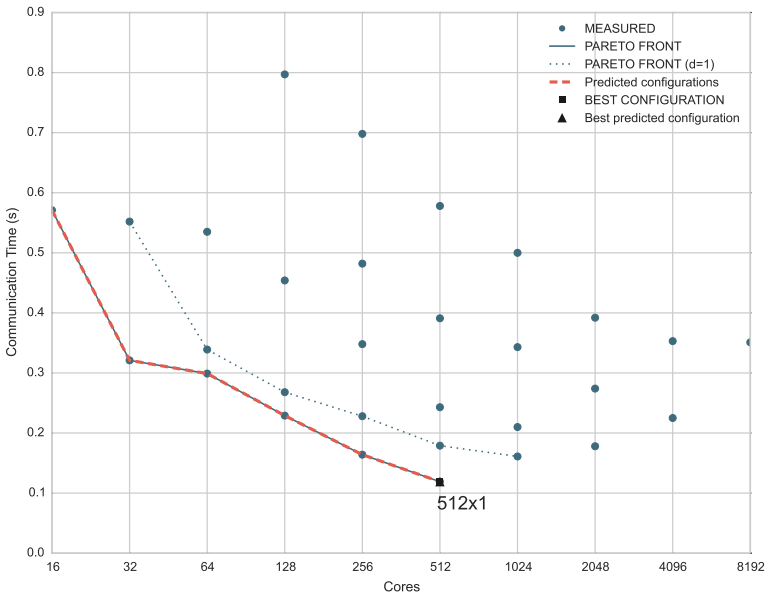
Τα μέτωπα Παρέτο για το πλήθος των πυρήνων και το χρόνο επικοινωνίας προσφέρουν επιπλέον ενδιαφέρουσα πληροφορία. Το σημείο με τον υψηλότερο αριθμό πυρήνων που περιλαμβάνεται στο μέτωπο παρέτο αντιστοιχεί σε εκείνη τη ρύθμιση εκτέλεσης που αποδίδει το μικρότερο χρόνο επικοινωνίας. Αν το σχήμα επικοινωνίας ή/και το μέγεθος του προβλήματος υπό εξέταση δεν κλιμακώνει, τότε το ίδιο σημείο του μετώπου Παρέτο αντιστοιχεί στο μέγιστο αριθμό πυρήνων που μπορεί να αξιοποιηθεί πριν η επικοινωνία σταματήσει να κλιμακώνει. Η ταυτοποίηση τέτοιων σημείων είναι ιδιαίτερα χρήσιμη για το χρήστη, που μπορεί να αξιοποιήσει αυτή την πληροφορία και να τη συνδυάσει με όποια πληροφορία διαθέτει για την κλιμακωσιμότητα των υπολογισμών,

Πίνακας 4.11: Ρυθμίσεις εκτέλεσης ελάχιστου χρόνου επικοινωνίας στο σύστημα Vilje

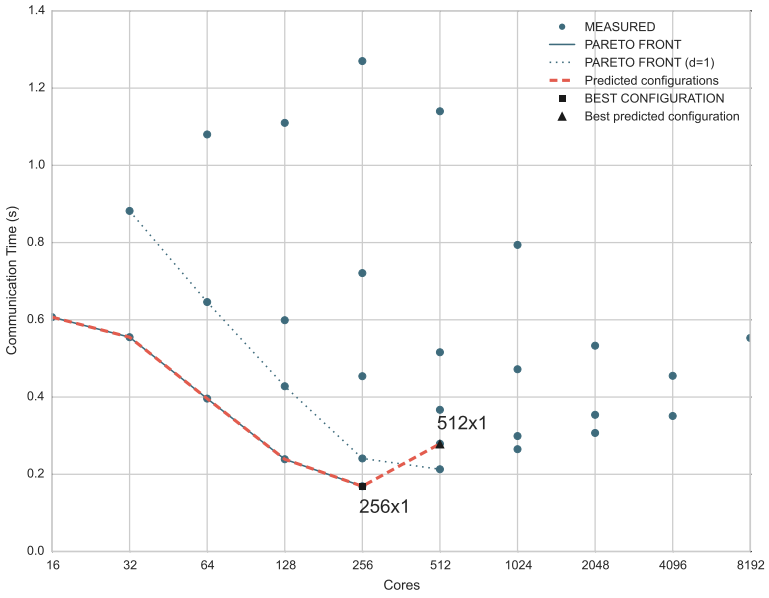
Σχήμα επικοινωνίας	Μέγεθος προβλήματος	Εκτέλεση	Μέτρηση ($n \times ppn$)	Πρόβλεψη ($n \times ppn$)	Αποτέλεσμα	Απόσταση
Halo-3D	128^3	#1	512×1	128×1	False	1
Halo-3D	128^3	#2	256×1	256×1	**True**	-
Halo-3D	128^3	#3	512×1	256×1	False	1
Halo-3D	256^3	#1	512×1	512×1	**True**	-
Halo-3D	256^3	#2	512×1	512×1	**True**	-
Halo-3D	256^3	#3	512×1	512×1	**True**	-
Halo-3D	512^3	#1	512×1	512×1	**True**	-
Halo-3D	512^3	#2	512×1	512×1	**True**	-
Halo-3D	512^3	#3	512×1	512×1	**True**	-
Halo-3D	1024^3	#1	512×1	512×1	**True**	-
Halo-3D	1024^3	#2	512×1	512×1	**True**	-
Halo-3D	1024^3	#3	512×1	512×1	**True**	-
Halo-3D	2048^3	#1	512×1	512×1	**True**	-
Halo-3D	2048^3	#2	512×1	512×1	**True**	-
Halo-3D	2048^3	#3	512×1	512×1	**True**	-
Halo-4D	128^4	#1	256×1	512×1	False	2
Halo-4D	128^4	#2	512×1	512×1	**True**	-
Halo-4D	256^4	#1	256×1	512×8	False	5
Halo-4D	256^4	#2	256×1	512×1	False	1
LULESH-1	240^3	#1	216×1	216×1	**True**	-
LULESH-1	480^3	#1	216×1	216×1	**True**	-
LULESH-2	240^3	#1	216×1	216×1	**True**	-
LULESH-2	480^3	#1	216×1	216×1	**True**	-
LULESH-3	240^3	#1	216×1	216×1	**True**	-
LULESH-3	480^3	#1	216×1	216×1	**True**	-

ώστε να επιλέξει το μέγιστο αριθμό πυρήνων που πρέπει να χρησιμοποιήσει για την εκτέλεση της εφαρμογής του. Επισημειώνουμε τις βέλτιστες ρυθμίσεις εκτέλεσης με βάση τους πραγματικούς χρόνους επικοινωνίας και τις προβλέψεις στο Σχήμα 4.9. Ο Πίνακας 4.11 δείχνει τις πραγματικές και προβλεφθείσες ρυθμίσεις εκτέλεσης με τον ελάχιστο χρόνο επικοινωνίας για τα διάφορα σχήματα επικοινωνίας στο Vilje. Χρησιμοποιούμε την έννοια της απόστασης του μετώπου Παρέτο για να αξιολογήσουμε τις λάθος προβλέψεις: υπολογίζουμε την απόσταση από το μέτωπο Παρέτο που ανήκουν οι προβλεφθείσες ρυθμίσεις εκτέλεσης που δε συμπίπτουν με τις πραγματικές. Το μοντέλο μας προβλέπει σωστά τις ρυθμίσεις εκτέλεσης που αποδίδουν τον ελάχιστο χρόνο επικοινωνίας για 20 από τις 25 περιπτώσεις. Λάθος προβλέψεις προκύπτουν κυρίως για το σχήμα επικοινωνίας *Halo-4D*, όπου, παρότι το μοντέλο μας εμφανίζει μικρά σχετικά σφάλματα πρόβλεψης, τείνει να προβλέπει χρόνο επικοινωνίας χαμηλότερο από τον πραγματικό και επομένως προβλέπει ότι η βέλτιστη ρύθμιση εκτέλεσης αντιστοιχεί σε υψηλότερο αριθμό πυρήνων από ό,τι στην πραγματικότητα. Το συγκεκριμένο σχήμα είναι το μοναδικό για το οποίο

η προβλεφθείσα ρύθμιση εκτέλεσης ανήκει στο 6ο μέτωπο Παρέτο (με απόσταση 5). Στο ίδιο σενάριο, το μοντέλο $\alpha_p - \beta_p - \gamma$ προβλέπει σωστά μόνο 5 από τις 25 ρυθμίσεις εκτέλεσης για τον ελάχιστο χρόνο επικοινωνίας.



(i) Όλα τα σημεία συμπίπτουν: *Halo-3D* με μέγεθος προβλήματος 1024^3 (εκτέλεση #2).



(ii) 5 από τα 6 σημεία συμπίπτουν: *Halo-4D* με μέγεθος προβλήματος 128^4 (εκτέλεση #1).

Σχήμα 4.9: Παραδείγματα πρόβλεψης των κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το χρόνο επικοινωνίας στο Vilje.

5

Πρόβλεψη επίδοσης επικοινωνίας στο σύστημα *Piz Daint*

5.1 Εισαγωγή

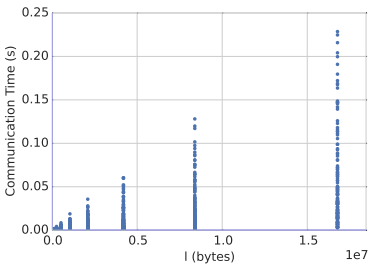
Στο παρόν κεφάλαιο, παρουσιάζουμε την εφαρμογή της μεθοδολογίας μας για την κατασκευή μοντέλων πρόβλεψης της επικοινωνίας (βλ. Ενότητες 3.5 και 3.6) στον υπερυπολογιστή *Piz Daint*. Περιγράφουμε τα βήματα που ακολουθούμε για την κατασκευή μοντέλων με τις μεθοδολογίες στατιστικής και μηχανικής μάθησης και αποτιμούμε την ικανότητα πρόβλεψης των μοντέλων μας σε διάφορα σχήματα επικοινωνίας. Συγκρίνουμε την προβλεπτική ικανότητα των μοντέλων μας και αποτιμούμε το βέλτιστο μοντέλο αναλυτικά. Επιπλέον, χρησιμοποιούμε τα μοντέλα μας σε σενάρια λήψης αποφάσεων με επίγνωση της επικοινωνίας, στοχεύοντας στη βελτιστοποίηση της χρήσης πόρων στο σύστημα Piz Daint.

5.2 Μοντελοποίηση με στατιστική μάθηση

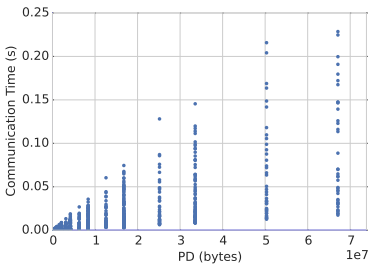
Ακολουθώντας τη μεθοδολογία κατασκευής μοντέλων με στατιστική μάθηση, που παρουσιάσαμε στο Κεφάλαιο 3, αρχικά κατασκευάσαμε πολυμεταβλητά γραμμικά μοντέλα παλινδρόμησης στο Piz Daint. Για τα μοντέλα αυτά, χρησιμοποιήσαμε χαρακτηριστικά από τις κατηγορίες Α και Β, καθώς και 6 χαρακτηριστικά της κατηγορίας Γ που αποτυπώνουν το σχήμα της κατανομής των κόμβων στο σύστημα Piz Daint: το μέγιστο αριθμό κόμβων ανά Aries SoC (*n/a*), το μέγιστο αριθμό Aries SoCs ανά chassis (*a/c*), το μέγιστο αριθμό chassis ανά ομάδα της τοπολογίας Dragonfly (*c/g*), και τα πλήθη των Aries SoCs, chassis και ομάδων (*a*, *c* και *g*). Αντίστοιχα με το σύστημα Vilje, χρησιμοποιούμε μόνο τη μέση τιμή των χαρακτηριστικών που αναφέρονται σε δεδο-

Πίνακας 5.1: Συσχέτιση των χαρακτηριστικών και του χρόνου επικοινωνίας στο σύστημα Piz Daint

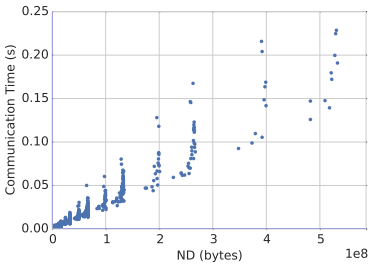
Χαρακτηριστικό	Συσχέτιση	Χαρακτηριστικό	Συσχέτιση
l	0.64	NM	0.04
PM	−0.059	TM	−0.04
ppn	0.1434	n/a	0.06
n	−0.08	a/c	−0.06
I	−0.02	c/g	−0.05
PD	0.7741	a	−0.08
ND	0.9703	c	−0.05
TD	0.80	g	−0.04



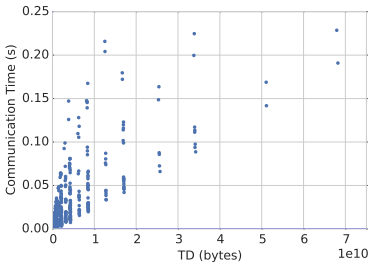
(i) l και χρόνος επικοινωνίας



(ii) PD και χρόνος επικοινωνίας



(iii) ND και χρόνος επικοινωνίας

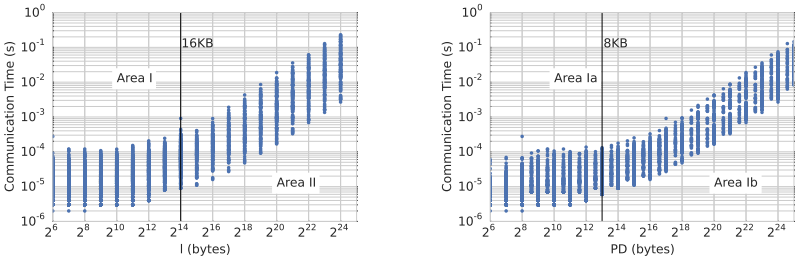


(iv) TD και χρόνος επικοινωνίας

Σχήμα 5.1: Διαγράμματα διασποράς για τα χαρακτηριστικά που εμφανίζουν υψηλή συσχέτιση με το χρόνο επικοινωνίας στο Piz Daint

μένα και πλήθη μηνυμάτων. Ακόμα, αντικαταστήσαμε τα χαρακτηριστικά που περιγράφουν την κίνηση δεδομένων εντός του κόμβου με το λόγο της επικοινωνίας μεταξύ κόμβων προς τη συνολική επικοινωνία. Αυτή η πρώιμη επιλογή των χαρακτηριστικών μείωσε τον αριθμό των χαρακτηριστικών μας σε 16.

Για τη συλλογή του συνόλου εκπαίδευσης, εκτελέσαμε το benchmark μας στο σύστημα Piz Daint, για διάφορες ρυθμίσεις εκτέλεσης, από 8 έως και 128



(i) Διαίρεση στις περιοχές I και II **(ii)** Διαίρεση στις περιοχές Ia και Ib

Σχήμα 5.2: Διαίρεση του χώρου των παραμέτρων σε υποπεριοχές στο Piz Daint

κόμβους, 1 έως 8 διεργασίες ανά κόμβο, 1 έως 4 μηνύματα ανά διεργασία και πολλαπλά μεγέθη μηνυμάτων, από 64 bytes έως 16 megabytes. Όπως και στο σύστημα Vilje, επαναλάβαμε τη συλλογή μετρήσεων αναφοράς για όλες τις διαφορετικές ρυθμίσεις εκτέλεσης, ώστε να συλλέξουμε διακριτές τιμές για τα χαρακτηριστικά της κατηγορίας Γ, που αφορούν το σχήμα της κατανομής των κόμβων, καταλήγοντας σε ένα σύνολο εκπαίδευσης που αποτελείται από 3040 σημεία. Έπειτα, υπολογίσαμε τους συντελεστές συσχέτισης του Pearson μεταξύ κάθε χαρακτηριστικού και του χρόνου επικοινωνίας. Οι τιμές των συντελεστών εμφανίζονται στον Πίνακα 5.1 και αφορούν στο σύνολο εκπαίδευσης. Παρατηρήσαμε πολύ υψηλή συσχέτιση για τα χαρακτηριστικά που αφορούν σε δεδομένα του κόμβου (*ND*) και στα συνολικά δεδομένα (*TD*), καθώς και υψηλή συσχέτιση για το χαρακτηριστικό που αφορά σε δεδομένα ανά διεργασία (*PD*) και στο μέγεθος των μηνυμάτων (*l*), όμοια με το σύστημα Vilje. Για να εντοπίσουμε την ύπαρξη ή μη γραμμικών σχέσεων μεταξύ των τεσσάρων χαρακτηριστικών και του χρόνου επικοινωνίας, χρησιμοποιήσαμε διαγράμματα διασποράς, όπως αυτά εμφανίζονται στο Σχήμα 5.1, τα οποία καταδεικνύουν γραμμικές σχέσεις. Στη συνέχεια, επιθεωρήσαμε τα ίδια διαγράμματα διασποράς σε λογαριθμική κλίμακα, όπου εντοπίσαμε αλλαγές στην κλίση των γραμμικών καμπυλών για όλα τα χαρακτηριστικά. Έτσι, διαιρέσαμε το χώρο των παραμέτρων, αρχικά με βάση το μήκος του μηνύματος l και την αλλαγή στην κλίση που εμφανίζεται στο Σχήμα 5.2i, στις υποπεριοχές I ($l \leq 8kilobytes$) και II ($l > 8kilobytes$). Διαιρέσαμε περαιτέρω την Περιοχή I σε δύο υποπεριοχές, με βάση την αλλαγή στην κλίση για το χαρακτηριστικό *PD*, όπως φαίνεται στο Σχήμα 5.2ii: την Περιοχή Ia ($PD \leq 8kilobytes$) και την Περιοχή Ib ($PD > 8kilobytes$).

Για κάθε μία από τις τρεις περιοχές, επανεξετάσαμε τη συσχέτιση μεταξύ των χαρακτηριστικών και του χρόνου επικοινωνίας και ξεκινήσαμε την κατασκευή γραμμικών πολυμεταβλητών μοντέλων με τη χρήση των χαρακτηριστικών με την υψηλότερη συσχέτιση, καταλήγοντας στα τρία ακόλουθα μοντέλα:

Πίνακας 5.2: Μοντέλο πρόβλεψης επικοινωνίας για την Περιοχή Ia στο σύστημα Piz Daint: Όροι και συντελεστές

Όροι	Συντελεστές
Σταθερός Όρος	1.816×10^{-5}
NM	5.0144×10^{-8}
n/a	-2.927×10^{-6}
$NM \times n/a$	3.2788×10^{-8}
c	-6.4923×10^{-7}
$NM \times c$	1.3872×10^{-8}
$n/a \times c$	2.9738×10^{-7}
$NM \times c \times n/a$	-1.8538×10^{-9}

Πίνακας 5.3: Μοντέλο πρόβλεψης επικοινωνίας για την Περιοχή Ib στο σύστημα Piz Daint: Όροι και συντελεστές

Όροι	Συντελεστές
Σταθερός Όρος	2.0604×10^{-5}
ppn	-1.7625×10^{-5}
ND	1.9318×10^{-10}
NM	2.7981×10^{-7}
a/c	7.5479×10^{-7}
$ppn \times ND$	5.3778×10^{-13}
$ppn \times NM$	-1.9800×10^{-8}
$ND \times a/c$	2.7822×10^{-12}

Πίνακας 5.4: Μοντέλο πρόβλεψης επικοινωνίας για την Περιοχή II στο σύστημα Piz Daint: Όροι και συντελεστές

Όροι	Συντελεστές
Σταθερός Όρος	7.0526×10^{-5}
ppn	-2.7899×10^{-5}
ND	2.2196×10^{-10}
$ND \times ppn$	1.4061×10^{-11}

- Μοντέλο Ia: $t_{comm} = b_0 + b_1 \times NM$
- Μοντέλο Ib: $t_{comm} = b_0 + b_1 \times ND + b_2 \times ND$
- Μοντέλο II: $t_{comm} = b_0 + b_1 \times ND$

Όπως και στο σύστημα Vilje, η επέκταση των παραπάνω μοντέλων με επιπλέον όρους δε βελτίωσε την ακρίβεια του μοντέλου και μας έτρεψε στην ανα-

ζήτηση όρων αλληλεπίδρασης. Τα διαγράμματα διασποράς του Σχήματος 5.1 κατέδειξαν υψηλή ετεροσκεδαστικότητα για όλα τα χαρακτηριστικά σε σχέση με το χρόνο επικοινωνίας. Για να προσδιορίσουμε τους όρους αλληλεπίδρασης, ακολουθήσαμε τρία βήματα, όπως και για το σύστημα Vilje. Εξαιρέσαμε χαρακτηριστικά που αποτελούν γινόμενα άλλων χαρακτηριστικών και εμφανίζουν αλληλοσυσχέτιση, υπολογίσαμε τους συντελεστές συσχέτισης για να πραγματοποιήσουμε μία πρώτη επιλογή επιπλέον χαρακτηριστικών και επιλέξαμε να συμπεριλάβουμε τουλάχιστον ένα χαρακτηριστικό που σχετίζεται με το σχήμα της κατανομής των κόμβων. Πειραματιστήκαμε με διάφορες μορφές μοντέλων και τελικά εκπαιδεύσαμε τα μοντέλα μας, τα οποία παρουσιάζουμε στους Πίνακες 5.2, 5.3 και 5.4, μαζί με τους συντελεστές τους. Εκπαιδεύσαμε τα μοντέλα μας με σθεναρή παλινδρόμηση που παρέχεται στην Κλάση *LinearModel* του λογισμικού Matlab R2013a. Αξίζει να σημειώσουμε πως το μοντέλο πρόβλεψης για την περιοχή II περιλαμβάνει μόνο τρεις όρους, το χαρακτηριστικό*ND*, το χαρακτηριστικό *ppn* και το γινόμενό τους, και κανένα χαρακτηριστικό της κατηγορίας Γ. Για τη συγκεκριμένη περιοχή, τα δύο χαρακτηριστικά και το γινόμενό τους αποδείχθηκαν επαρκή για την ακρίβεια του μοντέλου, ενώ η προσθήκη χαρακτηριστικών της κατηγορίας Γ δε βελτίωσε την ακρίβεια και την καλή προσαρμογή του μοντέλου.

Τα τελικά μοντέλα μάς επιτρέπουν να εξάγουμε συμπεράσματα για την επίδραση της αρχιτεκτονικής του συστήματος Piz Daint στην επίδοση της επικοινωνίας. Η εκροή δεδομένων από τον κόμβο είναι κρίσιμη για την επίδοση της επικοινωνίας: τα χαρακτηριστικά *ND* και *NM* που σκιαγραφούν την εξερχόμενη κίνηση από τον κόμβο συμπεριλαμβάνονται στα μοντέλα και των τριών περιοχών. Για την περιοχή Ia, όπου τόσο το μήκος όσο και το πλήθος των μηνυμάτων ανά διεργασία είναι μικρά (εξαιτίας της διαίρεσης του χώρου των παραμέτρων με βάση το χαρακτηριστικό *PD*), ο αριθμός των μηνυμάτων που εγχέονται από κάθε κόμβο στο δίκτυο είναι πιο σημαντικός από το μήκος του μηνύματος. Επιπρόσθετα, η πυκνότητα ή η αραιότητα της κατανομής των κόμβων στα Aries SoCs είναι σημαντική, όπως καταδεικνύει η παρουσία του χαρακτηριστικού *n / a*, και, όμοια με το σύστημα Vilje, γι'αυτή την περιοχή, το σχήμα της κατανομής των κόμβων και άρα η απόσταση (αριθμός βημάτων) στο δίκτυο είναι σημαντική για την επίδοση. Για την περιοχή Ib, όπου ο αριθμός των μηνυμάτων ανά διεργασία είναι μικρός αλλά οι διεργασίες παράγουν περισσότερη κίνηση, η επίδοση της επικοινωνίας επηρεάζεται επιπλέον από τον αριθμό των διεργασιών ανά κόμβο *ppn*, ως παράγοντα ανταγωνισμού, καθώς και από τον αριθμό των δικτυακών στοιχείων, δηλαδή των Aries SoCs *a*, που χρησιμοποιούνται από την κατανομή. Τέλος, στην περιοχή II, η επίδοση επηρεάζεται πρωταρχικά από τα δεδομένα που εγχέονται στο δίκτυο από κάθε κόμβο και από το πλήθος των διεργασιών ανά κόμβο. Εξαιτίας των μεγαλύτερων μηνυμάτων σε αυτή την περιοχή, τα όποια φαινόμενα συμφόρησης προκύπτουν στο

δίκτυο διασύνδεσης.

Εκτός από το πολυμεταβλητό γραμμικό μοντέλο παλινδρόμησης, εφαρμόσαμε τη μεθοδολογία που περιγράφεται στην Ενότητα 3.5 για την κατασκευή ενός μοντέλου μίας μεταβλητής με στατιστική μάθηση. Αρχικά, υπολογίσαμε τους συντελεστές συσχέτισης του Pearson για όλα τα χαρακτηριστικά και των τριών κατηγοριών (Α, Β και Γ) και επιλέξαμε το χαρακτηριστικό με την υψηλότερη συσχέτιση. Στο Piz Daint, το χαρακτηριστικό με την υψηλότερη συσχέτιση είναι το *AD (max)*. Αξίζει να σημειώσουμε ότι αυτό το χαρακτηριστικό είναι ανάλογο του χαρακτηριστικού με την υψηλότερη συσχέτιση στο σύστημα Vilje, δηλαδή το *SD (max)*. Και τα δύο χαρακτηριστικά ανήκουν στην κατηγορία Γ και σκιαγραφούν την κίνηση στο κύριο δικτυακό στοιχείο κάθε δικτύου διασύνδεσης, που στην περίπτωση του Piz Daint, είναι το Aries SoC. Χρησιμοποιώντας το επιλεγμένο χαρακτηριστικό, κατασκευάσαμε και εκπαιδεύσαμε 7806 μοντέλα, χρησιμοποιώντας μη γραμμικούς μετασχηματισμούς του χαρακτηριστικού, όπως περιγράφουμε στην Ενότητα 3.5. Επιλέξαμε, κατόπιν, το μοντέλο με τη βέλτιστη προσαρμογή στο σύστημα Piz Daint, στο οποίο αναφερόμαστε ως *LinReg-MV*, και είναι το ακόλουθο:

$$t = 7.939\times10^{-6}+1.183\times10^{-7}\times\sqrt{x}-1.278\times10^{-15}\times log_2^3 x+5.151\times10^{-15}\times x log_2^3 x$$

(δευτερόλεπτα), όπου $x = AD(max)$ σε bytes

5.3 Μοντελοποίηση με μηχανική μάθηση

Για την κατασκευή μοντέλων πρόβλεψης της επικοινωνίας με μηχανική μάθηση στο σύστημα Piz Daint, εφαρμόσαμε τη μεθοδολογία που περιγράφουμε στην Ενότητα 3.6 και κατασκευάσαμε τρία μοντέλα πρόβλεψης του χρόνου επικοινωνίας για το Piz Daint με τη χρήση της μεθόδου Gradient Boosting Regression Trees. Κάθε μοντέλο αξιοποιεί διαφορετικές κατηγορίες χαρακτηριστικών: το μοντέλο *GBRT-Class A* χρησιμοποιεί μόνο χαρακτηριστικά της κατηγορίας Α, το μοντέλο *GBRT-Class B* χρησιμοποιεί χαρακτηριστικά των κατηγοριών Α και Β και το μοντέλο *GBRT-Class C* χρησιμοποιεί χαρακτηριστικά όλων των κατηγοριών. Αντίστοιχα με το σύστημα Vilje, χρησιμοποιούμε μόνο τη μέγιστη τιμή των χαρακτηριστικών *l*, *PD* και *PM* της κατηγορίας Α, αφού για όλα τα σημεία στο σύνολο εκπαίδευσης, η ελάχιστη, η μέση και η μέγιστη τιμή αυτών των χαρακτηριστικών συμπίπτουν, εξαιτίας του σχεδιασμού του benchmark μας. Δεν πραγματοποιήσαμε επιλογή χαρακτηριστικών χειροκίνητα, αλλά εφαρμόσαμε συστηματική επιλογή χαρακτηριστικών για το μοντέλο *GBRT-Class C* με τη χρήση παλινδρόμησης με διανύσματα υποστήριξης και αναδρομική απαλοιφή χαρακτηριστικών.

Πίνακας 5.5: Χώρος παραμέτρων για τις εκτελέσεις του benchmark και τη συλλογή του συνόλου εκπαίδευσης στο σύστημα Piz Daint

Παράμετρος	**Εύρος**	
n	8-128	8-256
ppn	1-8	1-8
l	16B-16MB	16B-16KB
PM	1-4	1-4
#εκτελέσεις	3	4
#σημεία	6912	

Για την εκπαίδευση των μοντέλων μηχανικής μάθησης στο σύστημα Piz Daint, ενισχύσαμε το σύνολο εκπαίδευσης που χρησιμοποιήσαμε στη μεθοδολογία στατιστικής μάθησης με μία επιπλέον πλήρη συλλογή μετρήσεων αναφοράς για όλα τα μεγέθη μηνυμάτων και πλήθη μηνυμάτων και όλες τις ρυθμίσεις εκτέλεσης. Επιπρόσθετα, συλλέξαμε επιπλέον δεδομένα για μηνύματα μικρού μήκους, για τα οποία παρατηρήσαμε μεγάλες διακυμάνσεις στο χρόνο επικοινωνίας, ώστε να βελτιώσουμε την ακρίβεια της μεθόδου GBRT σε αυτή την περιοχή. Η παρατηρούμενη διακύμανση στο Piz Daint οφείλεται στην προσαρμοστική δρομολόγηση στο δίκτυο διασύνδεσης Cray Aries, που επιτρέπει τη δρομολόγηση πακέτων διαμέσου μη ελάχιστων μονοπατιών. Η επίδραση της προσαρμοστικής δρομολόγησης είναι ισχυρότερη σε μηνύματα μικρού μήκους, καθώς μπορεί να αλλάξει δραστικά το χρόνο απόκρισης αυτών. Το τελικό σύνολο εκπαίδευσης περιλαμβάνει 6192 σημεία, από τη συλλογή μετρήσεων αναφοράς σε διάφορα πλήθη κόμβων και πυρήνων και διαφορετικές ρυθμίσεις εκτέλεσης. Παρουσιάζουμε τις λεπτομέρειες του συνόλου εκπαίδευσης στον Πίνακα 5.5.

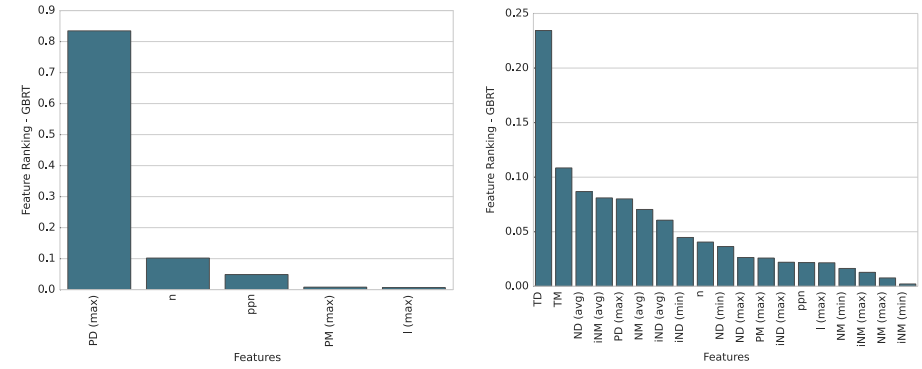
Η μέθοδος GBRT απαιτεί ρύθμιση για μία πληθώρα παραμέτρων. Έτσι, ελέγξαμε τη μέθοδο με πολλές διαφορετικές τιμές των παραμέτρων αυτών, καθώς και διαφορετικά πλήθη χαρακτηριστικών της κατηγορίας Γ και επιλέξαμε το μοντέλο με την καλύτερη προσαρμογή στο σύνολο εκπαίδευσης. Παρουσιάζουμε το εύρος των τιμών των παραμέτρων, για το οποίο ελέγξαμε τη μέθοδο, μαζί με τις επιλεγμένες τιμές, στον Πίνακα 5.6. Τα τελικά μοντέλα πρόβλεψης προκύπτουν από την εκπαίδευση της μεθόδου GBRT στο σύνολο εκπαίδευσής μας, μετά την επιλογή του πλήθους των χαρακτηριστικών και τη ρύθμιση των παραμέτρων της μεθόδου στις επιλεγμένες τιμές.

Η μέθοδος GBRT κατατάσσει τα χαρακτηριστικά κάθε μοντέλου σε μία κλίμακα από το 0 έως το 1, με βάση την επίδρασή τους στην έξοδο. Αυτή η κατάταξη αντικατοπτρίζει μόνο τη χρήση των χαρακτηριστικών από το μοντέλο: ένα χαρακτηριστικό με υψηλό βαθμό χρησιμοποιείται πιο συχνά ή στα

Πίνακας 5.6: Εύρος τιμών των παραμέτρων και επιλεγείσες τιμές για τις παραμέτρους της μεθόδου GBRT και το πλήθος των χαρακτηριστικών στο σύστημα Piz Daint

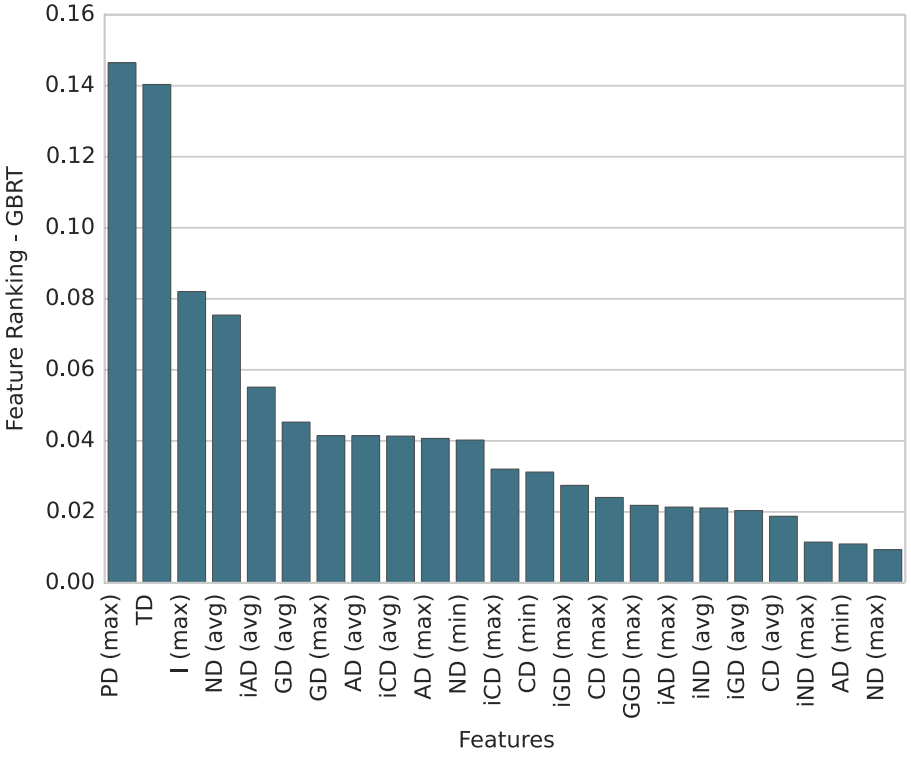
Παράμετρος	**Εύρος**	**Επιλεγείσα Τιμή**
n_estimators	100 - 2000	1000
learning_rate	0.001 - 1	0.009
min_samples_leaf	1 - 5	2
min_samples_split	2 - 8	3
max_depth	3 - 8	7
features_classA	-	5
features_classB	-	19
features_classC	15 - 40	23

υψηλότερα επίπεδα των δέντρων αποφάσεων, για τη διαίρεση του συνόλου εκπαίδευσης. Ωστόσο, μελετάμε την κατάταξη των χαρακτηριστικών για καθένα από τα τρία μοντέλα μας, ώστε να κατανοήσουμε την προβλεπτική ικανότητα και τους περιορισμούς των μοντέλων μας. Παρουσιάζουμε την κατάταξη των χαρακτηριστικών για τα μοντέλα της μεθόδου GBRT στο Piz Daint στο Σχήμα 5.3. Για το μοντέλο της κατηγορίας Α, *GBRT-Class A*, το χαρακτηριστκό που κατατάσσεται υψηλότερα είναι το*PD*, που αφορά σε δεδομένα ανά διεργασία, ενώ ακολουθείται από τα χαρακτηριστικά *n* και *ppn* που καθορίζουν το μέγεθος της κατανομής των πόρων. Μεταβαίνοντας από την κατηγορία Α στην κατηγορία Β, η μέθοδος δεν αγνοεί τα επιπλέον χαρακτηριστικά της κατηγορίας Β. Για το μοντέλο*GBRT-Class B*, το πιο σημαντικό χαρακτηριστικό είναι το*TD* που αφορά στα συνολικά δεδομένα του δικτύου διασύνδεσης, ωστόσο το συγκεκριμένο χαρακτηριστικό δεν κυριαρχεί στη διαδικασία κατασκευής του μοντέλου, αφού ο βαθμός του είναι χαμηλότερος από 0.25. Για την κατηγορία Γ, το χαρακτηριστικό που κατατάσσεται υψηλότερα είναι ξανά το *PD*, αν και βαθμονομείται με λιγότερο από 0.15. Συνολικά, η κατάταξη των χαρακτηριστικών για το μοντέλο *GBRT-Class C* στο Piz Daint δείχνει ότι τα δέντρα απόφασης λαμβάνουν υπόψη την κίνηση στα διάφορα σημεία του δικτύου, σε περισσότερα επίπεδα σε όλα τα βάθη των δέντρων απόφασης. Ωστόσο, το μοντέλο λαμβάνει υπόψη μόνο το πλήθος των δεδομένων: τα χαρακτηριστικά που αφορούν πλήθη μηνυμάτων κλαδεύονται στην επιλογή χαρακτηριστικών. Συνολικά, στο σύστημα Piz Daint, η κατάταξη των χαρακτηριστικών από τη μέθοδο GBRT δεν συμπίπτει με τους συντελεστές συσχέτισης του Pearson. Πολλά χαρακτηριστικά των κατηγοριών Β και Γ λαμβάνονται υπόψη από τα σχετικά μοντέλα με σχεδόν ίση σημασία.



(i) GBRT-Class A

(ii) GBRT-Class B



(iii) GBRT-Class C

Σχήμα 5.3: Κατάταξη χαρακτηριστικών για τα μοντέλα GBRT στο Piz Daint.

Πίνακας 5.7: Λεπτομέρειες του συνόλου ελέγχου στο σύστημα Piz Daint

	Piz Daint		
Σχήμα επικοινωνίας	Halo-3D	Halo-4D	LULESH
Μέγεθος προβλήματος	128^3, 256^3, 512^3, 1024^3, 2048^3	128^4, 256^4	240^3, 480^3
Επαναλήψεις	256	256	100
n	16-1024	16-1024	8-1000
ppn	1-8	1-8	1-8
#εκτελέσεις	3	2	1
#σημεία	613		

5.4 Πειραματική αποτίμηση

5.4.1 Σχήματα επικοινωνίας

Για να αποτιμήσουμε τα μοντέλα πρόβλεψης, χρησιμοποιήσαμε ένα σύνολο ελέγχου αντίστοιχο με αυτό του συστήματος Vilje. Πειραματιστήκαμε με το σχήμα επικοινωνίας *Halo-3D* (6 ανταλλαγές 2Δ-όψεων με τους γείτονες στο 3Δ-πλέγμα) και με το σχήμα επικοινωνίας *Halo-4D* (8 ανταλλαγές 3Δ-όψεων με τους γείτονες στο 4Δ-πλέγμα). Επίσης πειραματιστήκαμε με τα point-to-point σχήματα επικοινωνίας της αντιπροσωπευτικής εφαρμογής LULESH: το σχήμα *LULESH-1*, με 26 ανταλλαγές στο 3Δ-πλέγμα, το *LULESH-2*, με 6 ανταλλαγές στο 3Δ-πλέγμα και το *LULESH-3*, με επικοινωνία με τους 13 από τους 26 γείτονες στο 3Δ-πλέγμα (σχήμα wavefront). Παραπέμπουμε τον αναγνώστη στην Ενότητα 4.4.1 για περισσότερες πληροφορίες σχετικά με τα σχήματα επικοινωνίας και τις εφαρμογές.

Προβλέπουμε το χρόνο επικοινωνίας για τα σχήματα *Halo-3D* και *Halo-4D* και την εφαρμογή LULESH, για διάφορα μεγέθη προβλημάτων και διαφορετικές ρυθμίσεις εκτέλεσης, καθώς και για πολλαπλές εκτελέσεις, ώστε να ελέγξουμε την ικανότητα των χαρακτηριστικών της κατηγορίας Γ να περιγράφουν τις επιδράσεις των διαφορετικών κατανομών κόμβων. Οι λεπτομέρειες του συνόλου ελέγχου για το σύστημα Piz Daint δίνονται στον Πίνακα 5.7. Η εφαρμογή LULESH απαιτεί, λόγω του σχεδιασμού της, πλήθη διεργασιών που αποτελούν δυνάμεις του 3, άρα όλες οι ρυθμίσεις εκτέλεσης για την εφαρμογή LULESH υπόκεινται σε αυτόν τον περιορισμό.

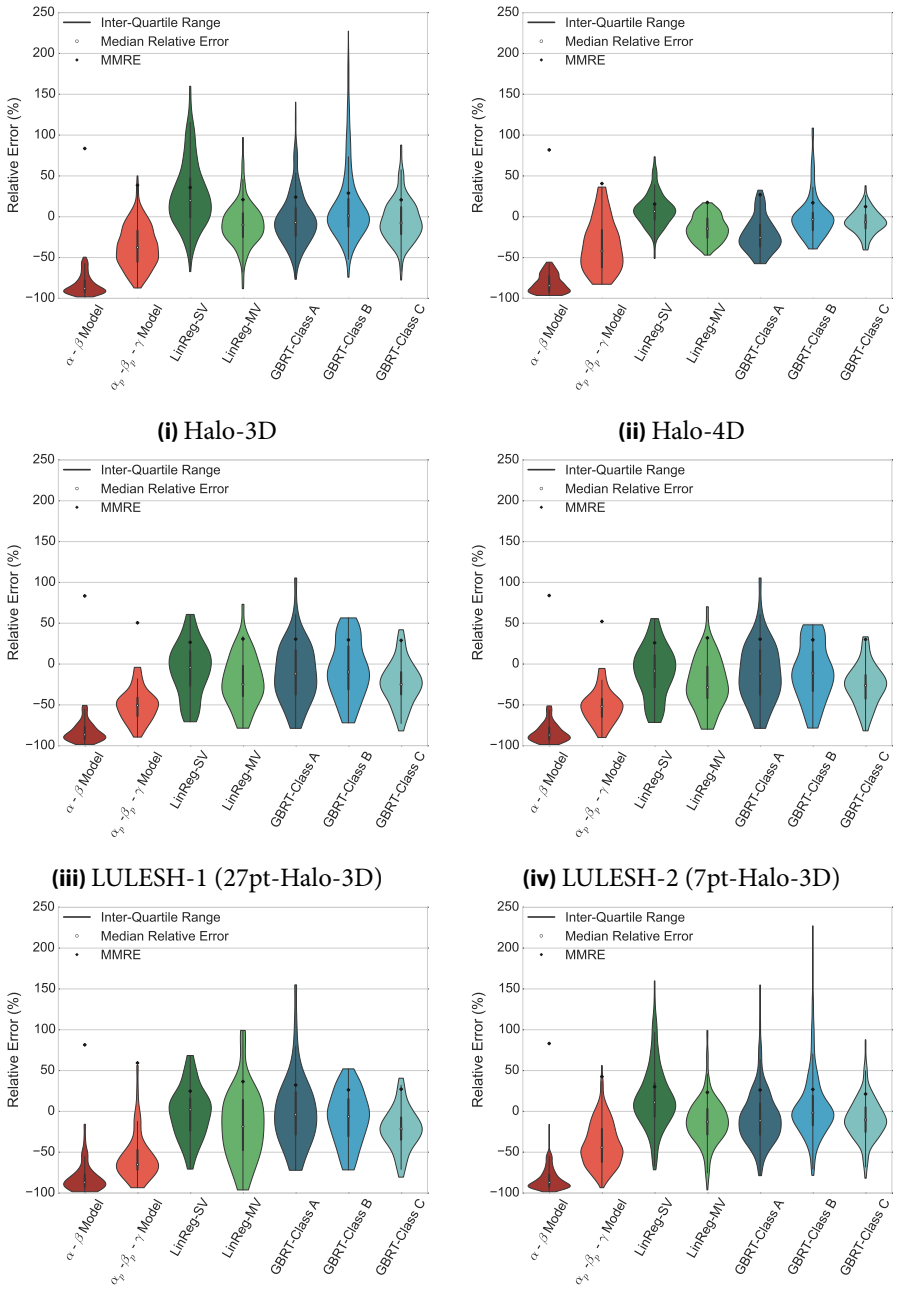
5.4.2 Σύγκριση των μοντέλων

Για την πειραματική αποτίμηση, αρχικά συγκρίνουμε την προβλεπτική ικανότητα των μοντέλων μας, χρησιμοποιώντας τις μετρικές που ορίσαμε στην Ενότητα 3.2. Αποτιμούμε τα δύο μοντέλα που κατασκευάσαμε με τεχνικές

Πίνακας 5.8: Μετρηθείσες παράμετροι για τα αναλυτικά και ημι-εμπειρικά μοντέλα στο σύστημα Piz Daint

Παράμετρος	Τιμή
α	0.238 us
β	0.114 ns
γ	0.453 us

στατιστικής μάθησης, δηλαδή τα μοντέλα *LinReg-SV* και *LinReg-MV*, καθώς και τα τρία μοντέλα που κατασκευάσαμε με τεχνικές μηχανικής μάθησης, δηλαδή τα μοντέλα *GBRT-Class A*, *GBRT-Class B* και *GBRT-Class C*. Επιπλέον, συγκρίνουμε την προβλεπτική ικανότητα του αναλυτικού μοντέλου $\alpha - \beta$ και του ημι-εμπειρικού μοντέλου $\alpha - \beta - \gamma$ με ποινές στις παραμέτρους α και β, που συμβολίζουμε ως μοντέλο $\alpha_p - \beta_p - \gamma$. Το συγκεκριμένο ημι-εμπειρικό μοντέλο αποδείχθηκε το ημι-εμπειρικό μοντέλο με τη βέλτιστη προσαρμογή στο σύστημα Piz Daint. Οι τιμές των παραμέτρων α, β και γ δίνονται στον Πίνακα 5.9. Η ποινή στην παράμετρο α αφορά τις πολλαπλές διεργασίες ανά κόμβο, ενώ η ποινή στην παράμετρο β αφορά το περιορισμένο εύρος ζώνης και τον ανταγωνισμό στο δίκτυο.



(i) Halo-3D

(ii) Halo-4D

(iii) LULESH-1 (27pt-Halo-3D)

(iv) LULESH-2 (7pt-Halo-3D)

(v) LULESH-3 (27pt-Wave-3D)

(vi) Συνολικά

Σχήμα 5.4: Σύγκριση μοντέλων στο σύστημα Piz Daint.

Πίνακας 5.9: Σύγκριση των μοντέλων με μετρικές ακρίβειας και καλής προσαρμογής στο σύστημα Piz Daint

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.00	0.092	0.856
$\alpha_p - \beta_p - \gamma$ *Model*	30.47	0.466	0.916
LinReg-SV	48.33	0.945	0.934
LinReg-MV	68.81	0.976	0.933
GBRT-Class A	61.90	0.942	0.927
GBRT-Class B	64.76	0.979	0.920
GBRT-Class C	68.33	0.771	0.940

(i) Halo-3D

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.00	-0.187	0.828
$\alpha_p - \beta_p - \gamma$ *Model*	31.25	0.297	0.884
LinReg-SV	82.14	0.864	0.948
LinReg-MV	71.42	0.962	0.953
GBRT-Class A	46.42	0.657	0.937
GBRT-Class B	78.57	0.982	0.949
GBRT-Class C	89.28	0.983	0.967

(ii) Halo-4D

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.00	-0.770	0.733
$\alpha_p - \beta_p - \gamma$ *Model*	12.50	0.083	0.825
LinReg-SV	58.92	0.643	0.810
LinReg-MV	42.86	0.605	0.862
GBRT-Class A	48.21	0.281	0.841
GBRT-Class B	44.64	0.692	0.832
GBRT-Class C	44.64	0.796	0.850

(iii) LULESH-1

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.00	-0.784	0.733
$\alpha_p - \beta_p - \gamma$ *Model*	8.93	0.060	0.825
LinReg-SV	60.71	0.637	0.806
LinReg-MV	44.64	0.619	0.862
GBRT-Class A	48.21	0.281	0.841
GBRT-Class B	44.64	0.701	0.827
GBRT-Class C	42.85	0.792	0.852

(iv) LULESH-2

	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	1.786	-0.623	0.729
$\alpha_p - \beta_p - \gamma$ *Model*	10.714	-0.053	0.813
LinReg-SV	58.93	0.824	0.834
LinReg-MV	41.07	0.097	0.845
GBRT-Class A	48.21	-0.420	0.853
GBRT-Class B	51.79	0.753	0.833
GBRT-Class C	53.57	0.800	0.858

(v) LULESH-3

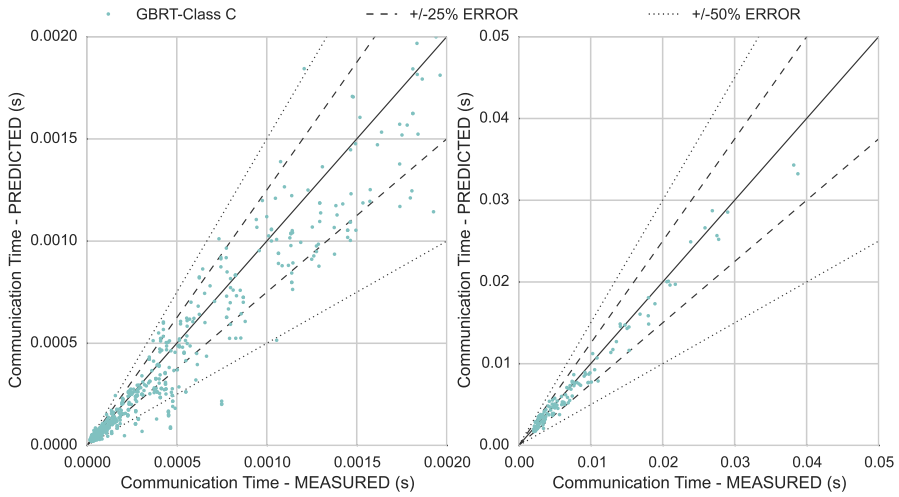
	$Pred_{0.25}$ (%)	R^2	RCC
$\alpha - \beta$ *Model*	0.14	0.182	0.874
$\alpha_p - \beta_p - \gamma$ *Model*	25.86	0.515	0.905
LinReg-SV	56.43	0.912	0.927
LinReg-MV	63.00	0.974	0.930
GBRT-Class A	56.14	0.788	0.929
GBRT-Class B	62.71	0.987	0.924
GBRT-Class C	66.57	0.986	0.940

(vi) Συνολικά

Το Σχήμα 5.4 απεικονίζει την κατανομή των σχετικών σφαλμάτων των προβλέψεων, για όλα τα μοντέλα, σε κάθε σχήμα επικοινωνίας, καθώς και συνολικά (βλ. Σχήμα 5.4vi), με τη μορφή των διαγραμμάτων βιολιού, ενώ στον Πίνακα 5.9 συνοψίζονται οι βαθμοί των μετρικών $Pred_{0.25}$, R^2 και RCC. Αρχικά, τόσο το αναλυτικό όσο και το ημι-εμπειρικό μοντέλο προβλέπουν μικρότερο χρόνο επικοινωνίας από τον πραγματικό, αν και το ημι-εμπειρικό μοντέλο παρουσιάζει καλύτερη ακρίβεια από το αναλυτικό. Αξίζει να σημειώσουμε πως το ημι-εμπειρικό μοντέλο $\alpha_p - \beta_p - \gamma$ παρουσιάζει μεγαλύτερη ακρίβεια και καλύτερη προσαρμογή στο σύστημα Piz Daint, σε σχέση με το σύστημα Vilje,

όπως φαίνεται και από τους βαθμούς των μετρικών $Pred_{0.25}$ και R^2 για όλα τα σχήματα επικοινωνίας. Επομένως, το ημι-εμπειρικό μοντέλο καταφέρνει να ενσωματώσει ορισμένες από τις ιδιότητες της τοπολογίας του συστήματος Piz Daint.

Αναφορικά με τα μοντέλα στατιστικής μάθησης, το μοντέλο *LinReg-SV* επιτυγχάνει εξαιρετική ακρίβεια, ειδικά για το σχήμα επικοινωνίας *Halo-4D*, αφού το επιλεγμένο χαρακτηριστικό, *AD (max)*, που αφορά στην κίνηση μέσω ενός Aries SoC, χαρακτηρίζει σε μεγάλο βαθμό το σχήμα κίνησης δεδομένων και την επίδοση της επικοινωνίας στο σύστημα Piz Daint. Αντίθετα με το σύστημα Vilje, στο Piz Daint, οι κατανομές των κόμβων είναι πυκνές και συνήθως ένα Aries SoC χρησιμοποιείται μόνο από μία εργασία (δηλαδή μία κατανομή για μία συγκεκριμένη εφαρμογή). Έτσι, το χαρακτηριστικό *AD (max)* ενσωματώνει όλη, ή το μεγαλύτερο μέρος της κίνησης διαμέσου του Aries SoC και αποκτά ιδιαίτερη σημασία για την επίδοση εκείνων των σχημάτων επικοινωνίας που μπορούν να προκαλέσουν συμφόρηση στο Aries SoC, όπως το σχήμα επικοινωνίας *Halo-4D* pattern. Το πολυμεταβλητό γραμμικό μοντέλο παλινδρόμησης *LinReg-MV* επιτυγχάνει υψηλή ακρίβεια σε σχήματα επικοινωνίας που είναι πιο συμμετρικά στα μεγέθη των μηνυμάτων, όπως τα σχήματα επικοινωνίας *Halo-3D* και *Halo-4D*, αλλά επιδεικνύει καλή προσαρμογή σε όλα τα σχήματα επικοινωνίας, όπως καταδεικνύουν οι υψηλοί βαθμοί της μετρικής *RCC*. Συνολικά, η ακρίβειά του είναι συγκρίσιμη με αυτή του μοντέλου *GBRT-Class B*, που στηρίζεται στα ίδια χαρακτηριστικά για την κίνηση των δεδομένων. Το μοντέλο *GBRT-Class A* αποδίδει σχετικά καλά σε όλα τα σχήματα επικοινωνίας. Παρόλα αυτά, η ακρίβειά του θα μπορούσε να βελτιωθεί με την ενίσχυση του συνόλου εκπαίδευσης με δεδομένα από ασύμμετρα σχήματα επικοινωνίας, καθώς το σύνολο των χαρακτηριστικών που χρησιμοποιεί και ο γενικός τρόπος συλλογής μετρήσεων αναφοράς λειτουργούν αποτρεπτικά για την ικανότητα του μοντέλου να αποδώσει την επίδοση περίπλοκων γράφων επικοινωνίας, όπως είναι αυτοί των σχημάτων *LULESH-1* και *LULESH-3*. Το μοντέλο με τη βέλτιστη προσαρμογή στο σύστημα Piz Daint είναι το μοντέλο της κατηγορίας Γ, *GBRT-Class C*, που επιτυγχάνει καλύτερη επίδοση πρόβλεψης από κάθε άλλο μοντέλο συνολικά και επιδεικνύει άριστη προσαρμογή, ενώ ταυτόχρονα συγκρατεί το εύρος των σχετικών σφαλμάτων, σε αντίθεση με το μοντέλο *GBRT-Class B*, το οποίο επίσης παρουσιάζει καλή προβλεπτική ικανότητα, αλλά προβλέπει σημαντικά μεγαλύτερους χρόνους επικοινωνίας για ορισμένα σημεία του συνόλου ελέγχου, για τα σχήματα επικοινωνίας *Halo-3D* και *Halo-4D*.



Σχήμα 5.5: Διαγράμματα διασποράς για τις προβλέψεις του μοντέλου *GBRT-Class C* στο Piz Daint. Ο χρόνος επικοινωνίας έχει κανονικοποιηθεί σε μία επανάληψη.

5.4.3 Λεπτομερής αποτίμηση

Σε συνέχεια της σύγκρισης των μοντέλων πρόβλεψης, επιλέγουμε το μοντέλο *GBRT-Class C* ως το βέλτιστο μοντέλο για το σύστημα Piz Daint, αφού επιτυγχάνει υψηλούς βαθμούς σε όλες τις μετρικές και μικρό εύρος σχετικών σφαλμάτων. Το μοντέλο αποδίδει προβλέψεις σε χρόνο αμέσως πριν (just-in-time) την εκτέλεση της εφαρμογής, ενώ η ακρίβειά του και η προσαρμογή του είναι πολύ υψηλές, με βαθμούς 66.57% στη μετρική $Pred_{0.25}$, 0.986 στη μετρική R^2 και 0.940 στη μετρική *RCC*. Επιπλέον, το μοντέλο *GBRT-Class C* μειώνει το βαθμό της μετρικής *MMRE* συνολικά σε 21.32%, από 42.60% του ημι-εμπειρικού μοντέλου $\alpha_p - \beta_p - \gamma$. Επομένως, αναλύουμε περαιτέρω την προβλεπτική επίδοση του μοντέλου αυτού. Αρχικά, μελετάμε όλα τα σημεία στο σύνολο ελέγχου, στη μορφή διαγραμμάτων διασποράς, όπως εμφανίζονται στο Σχήμα 5.5, με χρόνους κανονικοποιημένους σε μία επανάληψη, έχοντας διαιρέσει το σύνολο των σημείων σε δύο υποσύνολα με βάση το χρόνο επικοινωνίας, για λόγους ευκρίνειας. Η πλειονότητα των προβλέψεων κείται στο εύρος σφαλμάτων $\pm 25\%$, ενώ το 92.14% των προβλέψεων παρουσιάζουν σφάλματα εντός του εύρους $\pm 50\%$ και μόνο ένας ασήμαντος αριθμός ακραίων προβλέψεων εμφανίζεται, όταν ο χρόνος επικοινωνίας είναι μικρότερος των 100 μs. Τα διαγράμματα διασποράς δείχνουν ότι δεν υπάρχουν συστηματικά ακραία σφάλματα πρόβλεψης στο σύστημα Piz Daint: το μοντέλο παρουσιάζει καλή ικανότητα πρόβλεψης οποιουδήποτε σχήματος επικοινωνίας.

Κατόπιν, αποτιμούμε το μοντέλο *GBRT-Class C* σε κάθε ένα σχήμα επι-

κοινωνίας. Το σύνολο ελέγχου μας, για τα σχήματα επικοινωνίας *Halo-3D* και *Halo-4D* περιλαμβάνει πολλαπλές ρυθμίσεις εκτέλεσης, μεγέθη προβλημάτων και διακριτές εκτελέσεις. Για το λόγο αυτό, χρησιμοποιούμε το γινόμενο των μετρικών R^2 και RCC, $R^2 \times RCC$, για να βαθμονομήσουμε τα διάφορα υποσύνολα των μετρήσεων σε μια κλίμακα από 0 έως 1, όπου το 1 αντιστοιχεί στο σύνολο με τις βέλτιστες προβλέψεις και το 0 αντιστοιχεί στο σύνολο με τις χειρότερες προβλέψεις. Παρουσιάζουμε τις προβλέψεις της κλιμάκωσης των δύο σχημάτων επικοινωνίας, όταν κλιμακώνει ο αριθμός των κόμβων και όταν κλιμακώνει ο αριθμός των διεργασιών ανά κόμβο, για το βέλτιστο, το χειρότερο και το διάμεσο υποσύνολο, στο Σχήμα 5.6 για το σχήμα επικοινωνίας *Halo-3D* και στο Σχήμα 5.7 για το σχήμα επικοινωνίας *Halo-4D*. Παραπέμπουμε τον αναγνώστη στο Παράρτημα Β για την πλήρη αποτίμηση. Επιπλέον, συγκρίνουμε τις προβλέψεις του μοντέλου μας *GBRT-Class C* με τις προβλέψεις του ημι-εμπειρικού μοντέλου $\alpha_p - \beta_p - \gamma$. Για το σχήμα επικοινωνίας *Halo-3D*, το μοντέλο μας παρέχει άριστες προβλέψεις όταν κλιμακώνουμε το πλήθος των κόμβων. Ακόμα και στη χειρότερη περίπτωση, έχει τη δυνατότητα να προβλέπει την τάση κλιμάκωσης του σχήματος επικοινωνίας. Αντίστοιχα, όταν κλιμακώνουμε το πλήθος των διεργασιών ανά κόμβο, το μοντέλο μας παρουσιάζει μικρά σφάλματα πρόβλεψης και στις τρεις περιπτώσεις. Αξίζει να σημειώσουμε πως το μοντέλο $\alpha_p - \beta_p - \gamma$ κατορθώνει να προβλέψει τη συμπεριφορά κλιμάκωσης όταν κλιμακώνουμε το πλήθος των διεργασιών ανά κόμβο, σε αντίθεση με το σύστημα Vilje, όπου το ίδιο μοντέλο αδυνατεί να ενσωματώσει την επίδραση των πολλαπλών διεργασιών ανά κόμβο. Ωστόσο, στις περισσότερες περιπτώσεις, εμφανίζει υψηλότερα σφάλματα σε σύγκριση με το δικό μας μοντέλο. Για το σχήμα επικοινωνίας *Halo-4D* στο Σχήμα 5.7, το μοντέλο μας *GBRT-Class C* αντικατοπτρίζει άριστα την κλιμάκωση του σχήματος επικοινωνίας ακόμα και στη χειρότερη περίπτωση, παρόλο που εμφανίζει κάποια μεγαλύτερα σφάλματα, ενώ το μοντέλο $\alpha_p - \beta_p - \gamma$ αποτυγχάνει να προβλέψει την επίδοση της επικοινωνίας όταν κλιμακώνει ο αριθμός των διεργασιών ανά κόμβο, για αυτό το σχήμα επικοινωνίας.

Τέλος, στο Σχήμα 5.8, παρουσιάζουμε τις προβλέψεις του χρόνου επικοινωνίας για τρία point-to-point σχήματα επικοινωνίας της εφαρμογής LULESH, για όλες τις ρυθμίσεις εκτέλεσης στο σύνολο ελέγχου, ταξινομημένες με βάση το πλήθος των πυρήνων. Για την πλειοψηφία των ρυθμίσεων εκτέλεσης, το μοντέλο *GBRT-Class C* επιτυγχάνει άριστες προβλέψεις, με την εξαίρεση τεσσάρων σημείων(54×4, 216×1, 125×8, 216×8 κόμβων $\times$ διεργασιών ανά κόμβο) όπου το μοντέλο προβλέπει χαμηλότερο χρόνο επικοινωνίας και για τα τρία σχήματα για το μικρό μέγεθος προβλήματος(240^3), καθώς και δύο σημείων(8×1, 864×2) όπου το μοντέλο προβλέπει χαμηλότερο χρόνο επικοινωνίας από τον πραγματικό για το μεγαλύτερο μέγεθος προβλήματος(480^3). Ωστόσο, οι συγκεκριμένες ρυθμίσεις εκτέλεσης δεν παρουσιάζουν ομοιότητες

Πίνακας 5.10: Μέτωπα Παρέτο για ελαχιστοποίηση κόμβων και χρόνου επικοινωνίας στο σύστημα Piz Daint

Σχήμα επικοινωνίας	Μέγεθος προβλήματος	Εκτέλεση	Ρυθμίσεις εκτέλεσης μετώπου Παρέτο	Προβλεφθείσες ρυθμίσεις εκτέλεσης	Matches (Ταιριάσματα)	Απόσταση (μέγιστη)
Halo-3D	128^3	#1	2	2	1	2
Halo-3D	128^3	#2	4	2	1	1
Halo-3D	128^3	#3	5	2	2	0
Halo-3D	256^3	#1	4	4	3	2
Halo-3D	256^3	#2	7	5	2	2
Halo-3D	256^3	#3	6	6	3	1
Halo-3D	512^3	#1	6	6	5	1
Halo-3D	512^3	#2	7	6	6	0
Halo-3D	512^3	#3	6	6	5	0
Halo-3D	1024^3	#1	7	7	**7**	-
Halo-3D	1024^3	#2	7	7	**7**	-
Halo-3D	1024^3	#3	6	7	4	3
Halo-3D	2048^3	#1	7	7	**7**	-
Halo-3D	2048^3	#2	7	7	**7**	-
Halo-3D	2048^3	#3	7	7	**7**	-
Halo-4D	128^4	#1	7	7	6	0
Halo-4D	128^4	#2	7	7	2	2
Halo-4D	256^4	#1	7	7	5	1
Halo-4D	256^4	#2	7	7	6	1
LULESH-1	240^3	#1	12	8	5	4
LULESH-1	480^3	#1	13	11	7	4
LULESH-2	240^3	#1	12	11	6	5
LULESH-2	480^3	#1	13	10	7	4
LULESH-3	240^3	#1	12	9	5	5
LULESH-3	480^3	#1	13	9	7	2

και οι λάθος προβλέψεις μπορεί να οφείλονται σε θόρυβο ή ανισορροπία του χρόνου επικοινωνίας μεταξύ των διεργασιών λόγω της παρουσίας υπολογισμών στην εφαρμογή, ή και παρεμβολών από άλλες εφαρμογές κατά το χρόνο εκτέλεσης. Το μοντέλο $\alpha_p - \beta_p - \gamma$ αποδίδει προβλέψεις του χρόνου επικοινωνίας μικρότερες του πραγματικού για τις περισσότερες ρυθμίσεις εκτέλεσης, και για τα τρία σχήματα επικοινωνίας της εφαρμογής LULESH και αποτυγχάνει να συλλάβει τη συμπεριφορά κλιμάκωσης της επικοινωνίας και για τα τρία σχήματα επικοινωνίας.

5.4.4 Λήψη αποφάσεων με επίγνωση της επικοινωνίας

Αποτιμούμε περαιτέρω την ακρίβεια των μοντέλων πρόβλεψης για το χρόνο επικοινωνίας, στο πλαίσιο της απόδοσης ακριβών προβλέψεων για τη λήψη αποφάσεων με επίγνωση της επικοινωνίας. Επιλέγουμε το βέλτιστο μοντέλο στο σύστημα Piz Daint, το μοντέλο *GBRT-Class C* και το χρησιμοποιούμε για να προβλέψουμε σημεία που είναι κρίσιμα σε σενάρια λήψης αποφάσεων για τη βελτιστοποίηση της χρήσης των πόρων του συστήματος από την πλευρά του χρήστη. Όπως εξηγήσαμε στην Ενότητα 4.4.4, οι χρήστες των υπερυπολογιστών επιδιώκουν να βελτιστοποιήσουν τη χρήση του συστήματος μεγιστοποιώντας την επίδοση των εφαρμογών τους, ελαχιστοποιώντας την κατανάλωση του προϋπολογισμού τους και ελαχιστοποιώντας τους χρόνους αναμονής τους στις ουρές του συστήματος. Για να επιτύχουν τους στόχους τους,

πρέπει να είναι σε θέση να επιλέξουν βέλτιστες ρυθμίσεις για τους κόμβους, τις διεργασίες ανά κόμβο ή/και τα νήματα του OpenMP. Αντιμετωπίζουμε αυτό το πρόβλημα από την πλευρά της επικοινωνίας με τη χρήση του μοντέλου μας*GBRT-Class C* για να προβλέψουμε τις βέλτιστες ρυθμίσεις κόμβων και διεργασιών ανά κόμβο, δηλαδή τις ρυθμίσεις εκτέλεσης που ελαχιστοποιούν το χρόνο επικοινωνίας για έναν συγκεκριμένο αριθμό πυρήνων. Η πολιτική του συστήματος Piz Daint είναι η χρέωση των χρηστών του με βάση τις ώρες κόμβων (node hours). Επομένως, διατυπώνουμε το πρόβλημα σαν πρόβλημα ταυτόχρονης ελαχιστοποίησης του πλήθους των κόμβων και του χρόνου επικοινωνίας και εκμεταλλευόμαστε την έννοια του μη-κυριαρχούντος (non-dominated) μετώπου Παρέτο για την επιλογή των κατά Παρέτο βέλτιστων ρυθμίσεων κόμβων και χρόνου επικοινωνίας (n, t). Επιπλέον, χρησιμοποιούμε υποβέλτιστα μέτωπα Παρέτο και την απόστασή τους από το βέλτιστο μέτωπο Παρέτο ώστε να αποτιμήσουμε τις προβλέψεις μας για τις ρυθμίσεις εκτέλεσης. Παραπέμπουμε τον αναγνώστη στην Ενότητα 4.4.4 για τη λεπτομερή περιγραφή του μετώπου Παρέτο.

Το Σχήμα 5.9 δείχνει δύο παραδείγματα μετώπων Παρέτο στο σύστημα Piz Daint. Κάθε γράφημα δείχνει τις μετρήσεις του χρόνου επικοινωνίας για όλες τις ρυθμίσεις εκτέλεσης ενός συγκεκριμένου σχήματος επικοινωνίας, για ένα μέγεθος προβλήματος και μία εκτέλεση στο σύστημα Piz Daint. Το μέτωπο Παρέτο και το δεύτερο βέλτιστο μέτωπο Παρέτο (με απόσταση ίση με 1) σημειώνονται στο γράφημα. Το πρώτο γράφημα του Σχήματος 5.9i παρουσιάζει μία περίπτωση όπου οι προβλέψεις μας είναι επιτυχείς και όλες οι προβλεφθείσες ρυθμίσεις εκτέλεσης συμπίπτουν με τις κατά Παρέτο βέλτιστες ρυθμίσεις εκτέλεσης, που έχουν υπολογιστεί με βάση τις πραγματικές τιμές του χρόνου επικοινωνίας. Το δεύτερο γράφημα του Σχήματος 5.9ii παρουσιάζει μία περίπτωση όπου το μοντέλο μας δεν προβλέπει με ακρίβεια το μέτωπο Παρέτο: το μέτωπο Παρέτο περιλαμβάνει 13 σημεία, ενώ το μοντέλο μας προβλέψει 9 κατά Παρέτο βέλτιστες ρυθμίσεις εκτέλεσης, από τις οποίες οι 7 συμπίπτουν με τις ρυθμίσεις του μετώπου Παρέτο. Ωστόσο, η απόσταση των ρυθμίσεων εκτέλεσης που δε συμπίπτουν είναι το πολύ 2, δηλαδή ανήκουν στο δεύτερο ή τρίτο βέλτιστο μέτωπο Παρέτο. Ο Πίνακας 5.10 συνοψίζει τα αποτελέσματα για τα σημεία του μετώπου Παρέτο για τον πραγματικό χρόνο επικοινωνίας και τις προβλέψεις του χρόνου επικοινωνίας και το πλήθος των κόμβων. Παραπέμπουμε τον αναγνώστη στο Παράρτημα Β για τη λεπτομερή αποτίμηση. Σημειώνουμε ως "Matches" (ταιριάσματα) το πλήθος των προβλεφθεισών ρυθμίσεων εκτέλεσης που συμπίπτουν με τις ρυθμίσεις εκτέλεσης στο μέτωπο Παρέτο. Στις περιπτώσεις όπου οι προβλεφθείσες ρυθμίσεις εκτέλεσης δε συμπίπτουν με αυτές του μετώπου Παρέτο, αναθέτουμε σε κάθε ρύθμιση εκτέλεσης που δε συμπίπτει έναν βαθμό d που ανταποκρίνεται στην απόσταση του υπο-βέλτιστου μετώπου Παρέτο όπου ανήκει. Στον πίνακα αναφέρουμε

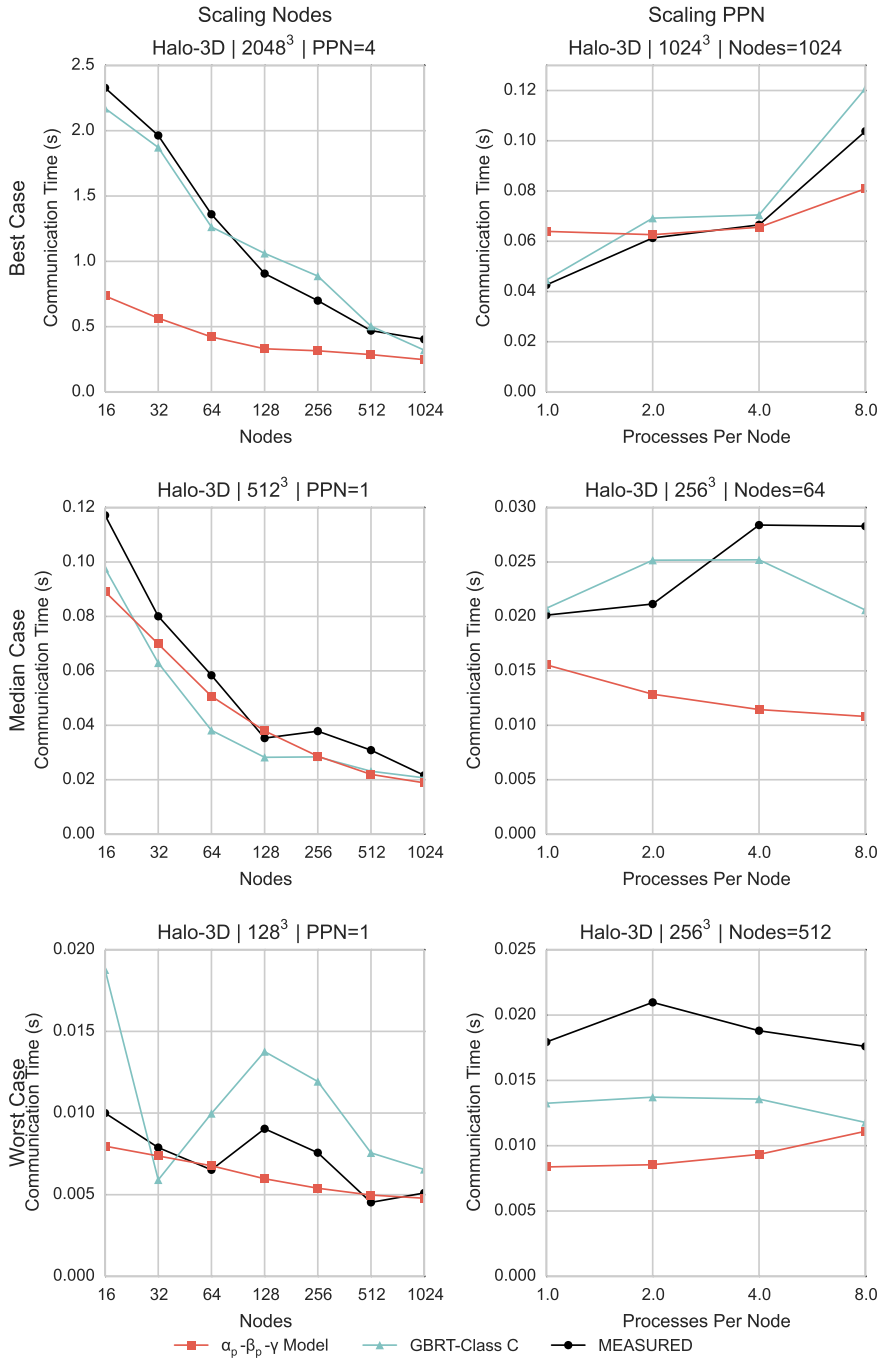
Πίνακας 5.11: Ρυθμίσεις εκτέλεσης ελάχιστου χρόνου επικοινωνίας στο σύστημα Piz Daint

Σχήμα επικοινωνίας	Μέγεθος προβλήματος	Εκτέλεση	Μέτρηση ($n \times ppn$)	Πρόβλεψη ($n \times ppn$)	Αποτέλεσμα	Απόσταση
Halo-3D	128^3	#1	128×4	128×8	False	2
Halo-3D	128^3	#2	512×1	32×1	False	0
Halo-3D	128^3	#3	256×2	32×1	False	0
Halo-3D	256^3	#1	128×1	128×1	**True**	-
Halo-3D	256^3	#2	1024×1	1024×4	False	2
Halo-3D	256^3	#3	1024×2	1024×4	False	1
Halo-3D	512^3	#1	1024×4	1024×1	False	1
Halo-3D	512^3	#2	1024×1	1024×1	**True**	-
Halo-3D	512^3	#3	1024×1	1024×1	**True**	-
Halo-3D	1024^3	#1	1024×1	1024×1	**True**	-
Halo-3D	1024^3	#2	1024×1	1024×1	**True**	-
Halo-3D	1024^3	#3	512×2	1024×1	False	3
Halo-3D	2048^3	#1	1024×1	1024×1	**True**	-
Halo-3D	2048^3	#2	1024×1	1024×1	**True**	-
Halo-3D	2048^3	#3	1024×1	1024×1	**True**	-
Halo-4D	128^4	#1	1024×1	1024×1	**True**	-
Halo-4D	128^4	#2	1024×2	1024×1	False	2
Halo-4D	256^4	#1	1024×1	1024×1	**True**	-
Halo-4D	256^4	#2	1024×1	1024×1	**True**	-
LULESH-1	240^3	#1	512×1	1000×1	False	3
LULESH-1	480^3	#1	1000×1	1000×1	**True**	-
LULESH-2	240^3	#1	512×1	1000×1	False	3
LULESH-2	480^3	#1	1000×1	1000×1	**True**	-
LULESH-3	240^3	#1	512×1	1000×1	False	3
LULESH-3	480^3	#1	1000×1	1000×1	**True**	-

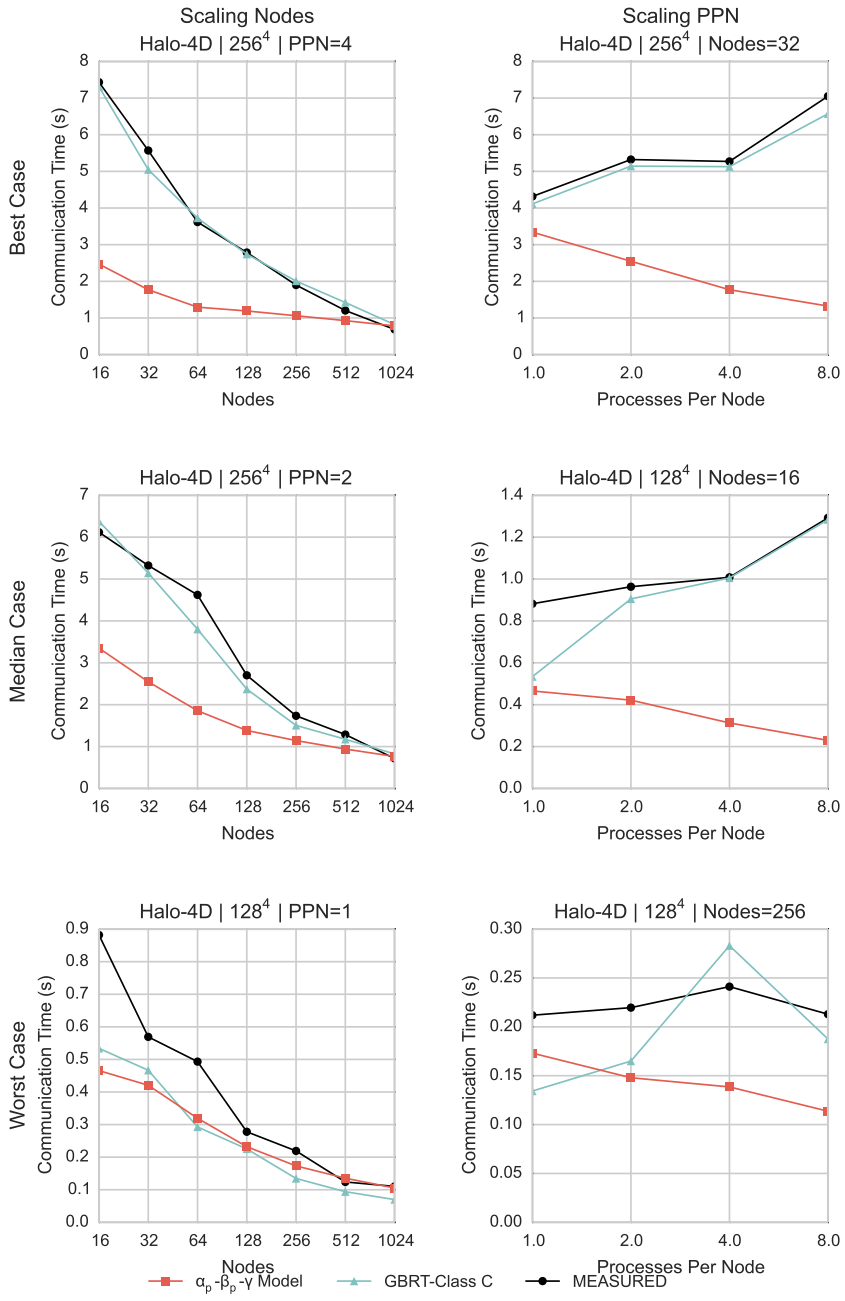
τη μέγιστη απόσταση των προβλεφθέντων σημείων που δε συμπίπτουν. Για τα 25 μέτωπα Παρέτο που κατασκευάσαμε, σε 5 περιπτώσεις, οι προβλέψεις συμπίπτουν ακριβώς με τα σημεία του μετώπου Παρέτο. Στις υπόλοιπες περιπτώσεις, τα μέτωπα διαφέρουν το πολύ κατά 7 σημεία (βλ. *LULESH-1* για το μέγεθος προβλήματος 240^3). Για 4 από τις εναπομείνασες περιπτώσεις, η μέγιστη απόσταση είναι 0: οι προβλεφθείσες ρυθμίσεις εκτέλεσης ανήκουν στο μέτωπο Παρέτο, αλλά το μοντέλο μας δεν κατορθώνει να προβλέψει όλα τα σημεία του μετώπου Παρέτο. Σε 9 περιπτώσεις, η μέγιστη απόσταση είναι το πολύ 2. Η πλειοψηφία των ρυθμίσεων εκτέλεσης που δε συμπίπτουν αφορούν την εφαρμογή LULESH, για την οποία το σύνολο ελέγχου μας περιλαμβάνει πολλά διαφορετικά πλήθη κόμβων και τα μέτωπα Παρέτο περιλαμβάνουν 12 με 13 σημεία. Σε αυτές τις περιπτώσεις, το μοντέλο μας κατορθώνει να προβλέψει περίπου το 50% των βέλτιστων ρυθμίσεων εκτέλεσης. Για λόγους σύγκρισης, κατασκευάσαμε τα μέτωπα Παρέτο στηριζόμενοι στις προβλέψεις του μοντέλου $\alpha_p - \beta_p - \gamma$ και παρατηρήσαμε ότι το πραγματικό και το προβλεφθέν μέτωπο Παρέτο δε συμπίπτουν σε καμία από τις 25 περιπτώσεις.

Τα χαρακτηριστικά των κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης στο σύστημα Piz Daint δεν είναι το ίδιο ομοιόμορφα όπως αυτά στο σύστημα Vilje: η βέλτιστη ρύθμιση δε χρησιμοποιεί πάντα τις λιγότερες δυνατές διεργασίες ανά κόμβο. Στην πραγματικότητα, τα αποτελέσματα διαφέρουν, ανάλογα με το σχήμα επικοινωνίας, το μέγεθος προβλήματος και την εκτέλεση. Συνεπώς, η καθοδήγηση του χρήστη προς συγκεκριμένες ρυθμίσεις εκτέλεσης (π.χ. τη χρήση των λιγότερων δυνατών διεργασιών ανά κόμβο) δεν επαρκεί για τη βελτιστοποίηση της κατανάλωσης ωρών κόμβων (node hours) στο σύστημα Piz Daint. Αυτή η παρατήρηση επιβεβαιώνει την ανάγκη για ακριβή μοντέλα πρόβλεψης του χρόνου επικοινωνίας, που μπορούν να διακρίνουν μεταξύ διαφορετικών ρυθμίσεων εκτέλεσης και να βοηθήσουν το χρήστη να επιλέξει τη βέλτιστη ρύθμιση εκτέλεσης για την εφαρμογή του.

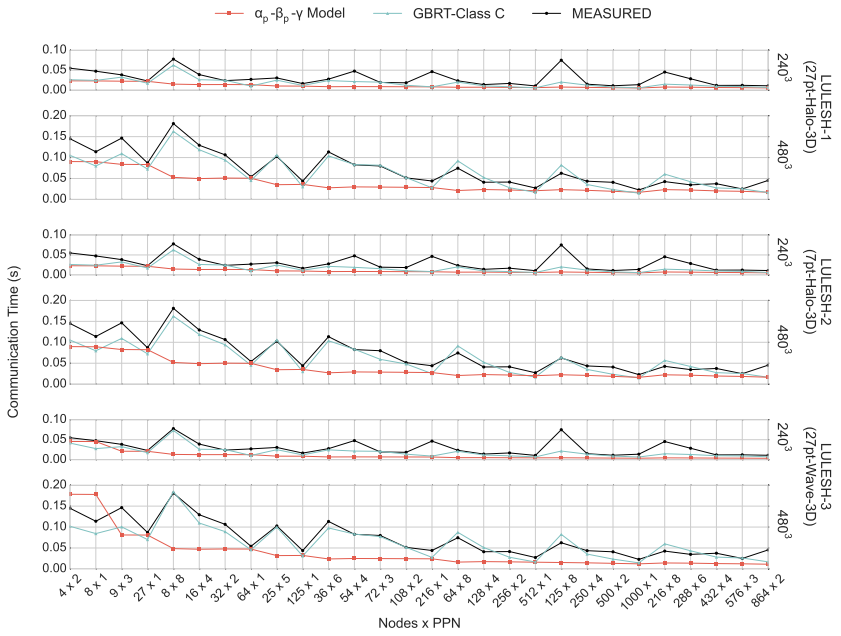
Τα μέτωπα Παρέτο για τους κόμβους και το χρόνο επικοινωνίας προσφέρουν επιπλέον πληροφορία για τη ρύθμιση εκτέλεσης που οδηγεί στον ελάχιστο χρόνο επικοινωνίας. Αυτή η ρύθμιση εκτέλεσης αφορά στο σημείο του μετώπου Παρέτο με το μεγαλύτερο πλήθος κόμβων. Αν το σχήμα επικοινωνίας ή/και το μέγεθος του προβλήματος υπό εξέταση δεν κλιμακώνει, τότε το σημείο με τον υψηλότερο αριθμό κόμβων στο μέτωπο Παρέτο αποτελεί τη ρύθμιση εκτέλεσης με το μεγαλύτερο αριθμό κόμβων που ο χρήστης πρέπει να χρησιμοποιήσει, πριν το σχήμα επικοινωνίας της εφαρμογής του σταματήσει να κλιμακώνει. Η αναγνώριση τέτοιων σημείων, όπου η επικοινωνία σταματά να κλιμακώνει, είναι ιδιαίτερα χρήσιμη για το χρήστη, που μπορεί να αξιοποιήσει αυτή την πληροφορία μαζί με άλλη πληροφορία για την κλιμάκωση των υπολογισμών της εφαρμογής του, ώστε να επιλέξει το μέγιστο αριθμό των πυρήνων που πρέπει να χρησιμοποιήσει για την εκτέλεση της εφαρμογής του. Επισημειώνουμε τη βέλτιστη ρύθμιση εκτέλεσης με βάση τη μέτρηση και την πρόβλεψη του χρόνου επικοινωνίας στο Σχήμα 5.9. Ο Πίνακας 5.11 δείχνει τις βέλτιστες ρυθμίσεις εκτέλεσης με βάση τη μέτρηση του χρόνου επικοινωνίας και την πρόβλεψη με το μοντέλο *GBRT-Class C* για τα σχήματα επικοινωνίας μας στο σύστημα Piz Daint. Χρησιμοποιούμε την έννοια της απόστασης των μετώπων Παρέτο για να αξιολογήσουμε τις λανθασμένες προβλέψεις ρυθμίσεων εκτέλεσης: υπολογίζουμε την απόσταση του μετώπου Παρέτο στο οποίο ανήκει η ρύθμιση εκτέλεσης που έχουμε προβλέψει. Το μοντέλο μας προβλέπει σωστά τη ρύθμιση εκτέλεσης που αποδίδει τον ελάχιστο χρόνο επικοινωνίας για 14 από τις 25 περιπτώσεις. Για τις εναπομείνασες περιπτώσεις, οι προβλέψεις για τη βέλτιστη ρύθμιση εκτέλεσης ανήκουν στη χειρότερη περίπτωση στο 4ο βέλτιστο μέτωπο Παρέτο (απόσταση ίση με 3). Για 2 περιπτώσεις, η βέλτιστη ρύθμιση εκτέλεσης που προβλέπεται ανήκει στο μέτωπο Παρέτο, αλλά δεν οδηγεί στον ελάχιστο χρόνο επικοινωνίας. Στο ίδιο σενάριο, το μοντέλο $\alpha_p - \beta_p - \gamma$ προβλέπει σωστά μόνο 4 από τις 25 ρυθμίσεις εκτέλεσης με τον ελάχιστο χρόνο επικοινωνίας.



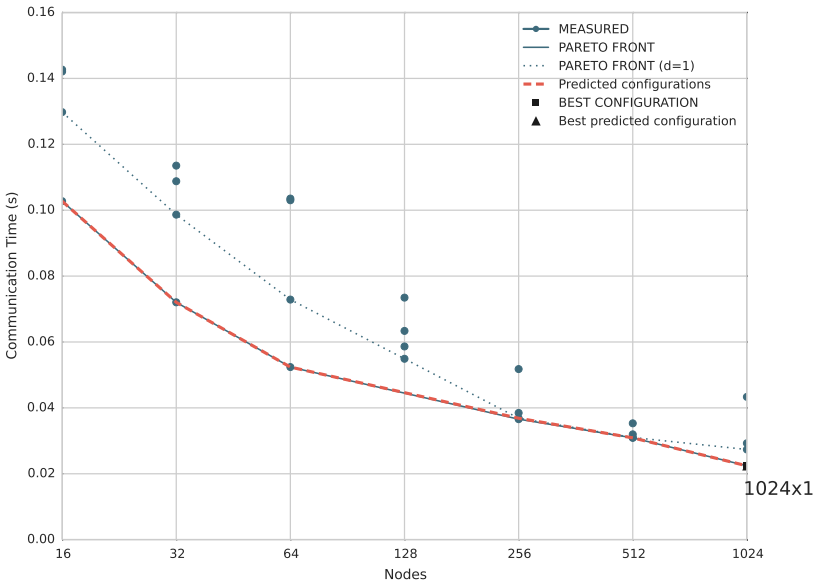
Σχήμα 5.6: Προβλέψεις για το σχήμα επικοινωνίας Halo-3D με το μοντέλο *GBRT-Class C* στο Piz Daint.



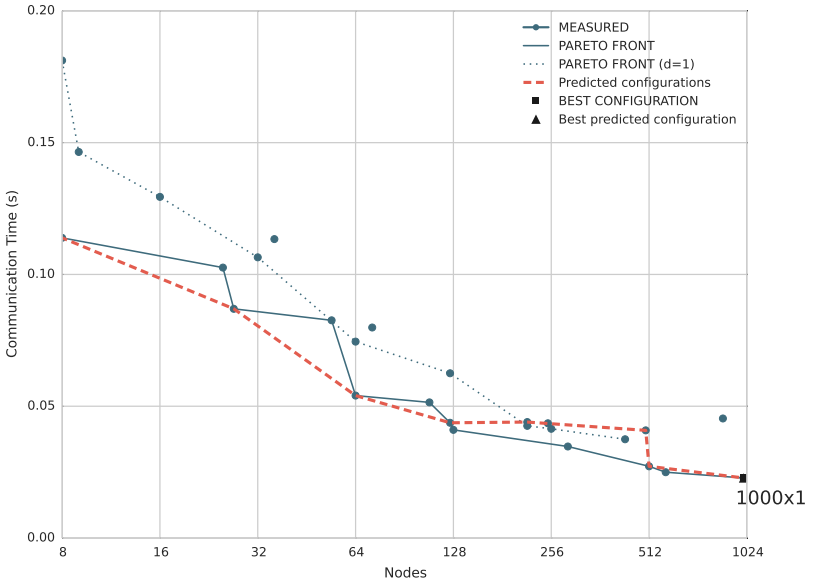
Σχήμα 5.7: Προβλέψεις για το σχήμα επικοινωνίας Halo-4D με το μοντέλο *GBRT-Class C* στο Piz Daint.



Σχήμα 5.8: Προβλέψεις για την εφαρμογή LULESH με το μοντέλο *GBRT-Class C* στο Piz Daint.



(i) Όλα τα σημεία συμπίπτουν: *Halo-3D* με μέγεθος προβλήματος 512^3 (εκτέλεση #3).



(ii) 7 σημεία συμπίπτουν: *LULESH-3* με μέγεθος προβλήματος 480^3.

Σχήμα 5.9: Παραδείγματα πρόβλεψης των κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το χρόνο επικοινωνίας στο Piz Daint.

6

Πρόβλεψη επίδοσης επικοινωνίας στο σύστημα *ARIS*

6.1 Εισαγωγή

Σε αυτό το κεφάλαιο, παρουσιάζουμε την εφαρμογή της μεθοδολογίας μας για την κατασκευή μοντέλων πρόβλεψης επίδοσης στη συστοιχία *ARIS*. Περιγράφουμε τα βήματα για την κατασκευή μοντέλων πρόβλεψης του χρόνου επικοινωνίας με τεχνικές μηχανικής μάθησης[1] και αποτιμούμε την προβλεπτική ικανότητα των μοντέλων μας σε διάφορα σχήματα επικοινωνίας. Συγκρίνουμε την επίδοση των προβλέψεων των μοντέλων μας και αποτιμούμε λεπτομερώς το μοντέλο με τη βέλτιστη επίδοση. Τέλος, χρησιμοποιούμε το μοντέλο με τη βέλτιστη επίδοση σε σενάρια λήψης αποφάσεων με επίγνωση της επικοινωνίας, που στοχεύουν στη βελτιστοποίηση της χρήσης των πόρων του συστήματος *ARIS*.

6.2 Μοντελοποίηση με τεχνικές μηχανικής μάθησης

Για την κατασκευή μοντέλων πρόβλεψης της επικοινωνίας με τεχνικές μηχανικής μάθησης στο σύστημα ARIS, εφαρμόσαμε τη μεθοδολογία που περιγράφουμε στην Ενότητα 3.6 και κατασκευάσαμε τρία μοντέλα πρόβλεψης για το χρόνο επικοινωνίας, με τη χρήση της μεθόδου Gradient Boosting Regression Trees. Καθένα από τα τρία μοντέλα αξιοποιεί χαρακτηριστικά από τις τρεις διαφορετικές κατηγορίες: το μοντέλο *GBRT-Class A* χρησιμοποιεί μόνο χαρακτηριστικά της κατηγορίας A, το μοντέλο *GBRT-Class B* χρησιμοποιεί χα-

[1] Δεν κατασκευάζουμε μοντέλα πρόβλεψης της επικοινωνίας με τεχνικές στατιστικής μάθησης στο σύστημα ARIS, καθώς αποκτήσαμε πρόσβαση στο συγκεκριμένο σύστημα μετά την αποτίμηση των μεθοδολογιών μας στα συστήματα Vilje και Piz Daint.

Πίνακας 6.1: Χώρος παραμέτρων για τις εκτελέσεις του benchmark και τη συλλογή του συνόλου εκπαίδευσης στο σύστημα ARIS

Παράμετρος		**Εύρος**	
n	8-256	8-128	8 - 128
ppn	1-20	1-20	4 - 20
l	16B-16MB	16B-16MB	16B - 16MB
PM	1-8	1-8	
#εκτελέσεις	1	1	1
#σημεία		7040	

ρακτηριστικά των κατηγοριών Α και Β και το μοντέλο *GBRT-Class C* χρησιμοποιεί χαρακτηριστικά και των τριών κλάσεων. Όπως και στα συστήματα Vilje και Piz Daint, χρησιμοποιούμε τη μέγιστη τιμή για τα χαρακτηριστικά *l*, *PD* και *PM*, αφού οι ελάχιστες, μέσες και μέγιστες τιμές αυτών των χαρακτηριστικών είναι ίσες για όλα τα σημεία στο σύνολο εκπαίδευσής μας. Εφαρμόσαμε συστηματική επιλογή χαρακτηριστικών για το μοντέλο της κατηγορίας Γ *GBRT-Class C*, με τη χρήση αναδρομικής απαλοιφής χαρακτηριστικών, μετά τη βαθμονόμηση των χαρακτηριστικών με τη μέθοδο παλινδρόμησης με διανύσματα υποστήριξης. Ωστόσο, τελικά, στην περίπτωση του ARIS, κανένα χαρακτηριστικό δεν απαλείφθηκε.

Για την εκπαίδευση των μοντέλων μας με τεχνικές μηχανικής μάθησης στο σύστημα ARIS, συλλέξαμε ένα σύνολο εκπαίδευσης με τη συλλογή μετρήσεων αναφοράς στο σύστημα ARIS, για πολλαπλές ρυθμίσεις εκτέλεσης, πολλαπλά μεγέθη μηνυμάτων και πολλαπλά πλήθη μηνυμάτων. Επαναλάβαμε τη διαδικασία συλλογής μετρήσεων αναφοράς με το benchmark μας δύο φορές, για δύο λόγους. Ο πρώτος λόγος αφορά στη συλλογή διακριτών τιμών για τα χαρακτηριστικά της κατηγορίας Γ. Ο δεύτερος λόγος αφορά στη συλλογή διακριτών τιμών για τα χαρακτηριστικά της κατηγορίας Β, εφαρμόζοντας ελάχιστες αλλαγές στον τρόπο παραγωγής των τυχαίων ζευγών των διεργασιών, ώστε να προκαλέσουμε αύξηση της επικοινωνίας εντός του κόμβου. Οι κόμβοι του συστήματος ARIS είναι αρχιτεκτονικής NUMA με 20 πυρήνες ανά κόμβο, επομένως είναι σημαντικό τα μοντέλα μας να είναι σε θέση να ενσωματώσουν την επίδραση της επικοινωνίας εντός του κόμβου, μέσω επαρκών σημείων στο σύνολο εκπαίδευσης. Το τελικό σύνολο εκπαίδευσης περιλαμβάνει 7040 σημεία και τα χαρακτηριστικά του εμφανίζονται στον Πίνακα 6.1.

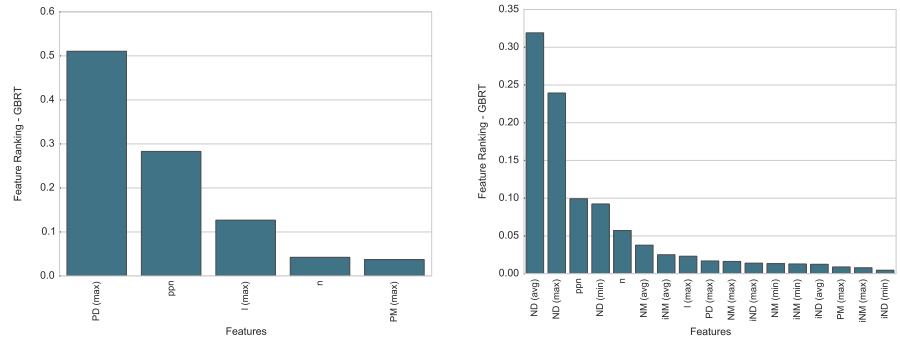
Καθώς η μέθοδος GBRT απαιτεί τη ρύθμιση των πολλών παραμέτρων της, ελέγξαμε τη μέθοδο με ένα εύρος τιμών για αυτές τις παραμέτρους, καθώς και για διάφορα πλήθη χαρακτηριστικών για το μοντέλο της κατηγορίας Γ και επιλέξαμε το μοντέλο με την καλύτερη προσαρμογή στο σύνολο εκπαίδευσης.

Πίνακας 6.2: Εύρος τιμών των παραμέτρων και επιλεγείσες τιμές για τις παραμέτρους της μεθόδου GBRT και το πλήθος των χαρακτηριστικών στο σύστημα ARIS

Παράμετρος	**Εύρος**	**Επιλεγείσα Τιμή**
n_estimators	100 - 2000	500
learning_rate	0.001 - 1	0.01
min_samples_leaf	1 - 5	1
min_samples_split	2 - 8	2
max_depth	3 - 8	6
features_classA	-	5
features_classB	-	19
features_classC	15 - 37	37

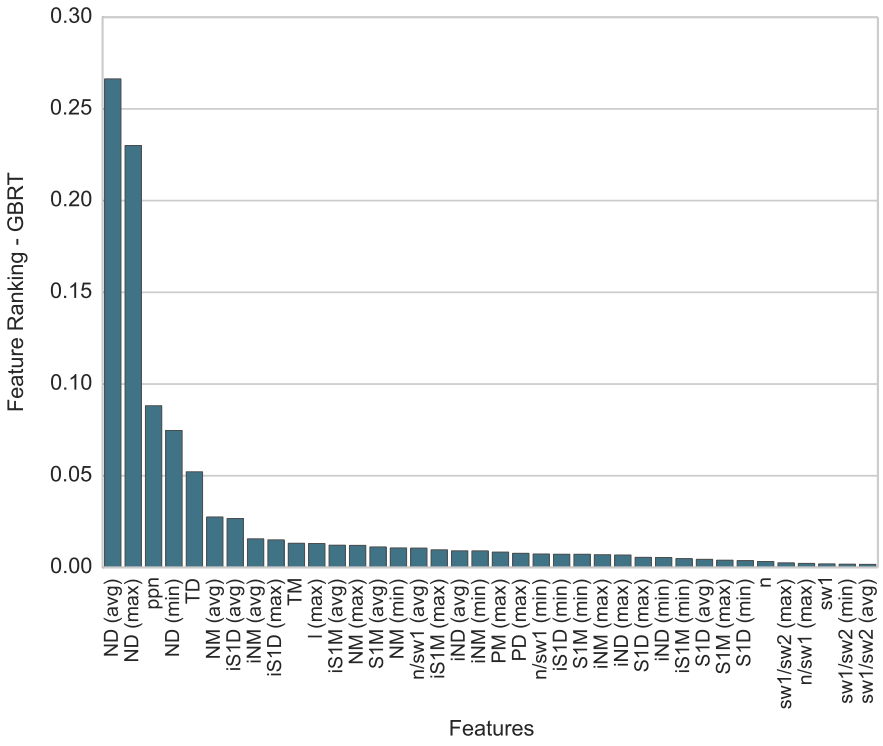
Παρουσιάζουμε το εύρος τιμών των παραμέτρων που ελέγξαμε, μαζί με τις τιμές που τελικά επελέγησαν, στον Πίνακα 6.2. Τα τελικά μοντέλα πρόβλεψης προκύπτουν από την εκπαίδευση της μεθόδου GBRT με το σύνολο εκπαίδευσής μας, έχοντας θέσει το πλήθος των χαρακτηριστικών και τις παραμέτρους της μεθόδου στις επιλεγμένες τιμές.

Χρησιμοποιούμε την κατάταξη των χαρακτηριστικών που μας παρέχει η μέθοδος GBRT (σε κλίμακα από 0 έως 1), ώστε να εντοπίσουμε παράγοντες που επιδρούν στην επίδοση της επικοινωνίας στο σύστημα ARIS. Ωστόσο, όπως έχουμε εξηγήσει στα προηγούμενα κεφάλαια, αυτή η κατάταξη αντικατοπτρίζει μόνο τη χρήση των χαρακτηριστικών από τη συγκεκριμένη μέθοδο και ένας χαμηλός βαθμός σε κάποιο χαρακτηριστικό δεν αντιστοιχεί απαραίτητα σε μικρότερη επίδραση του χαρακτηριστικού στην επίδοση της επικοινωνίας ή στην απουσία αιτιότητας. Παρουσιάζουμε την κατάταξη των χαρακτηριστικών για τα τρία μοντέλα της μεθόδου GBRT στο Σχήμα 6.1. Για το μοντέλο *GBRT-Class A*, το χαρακτηριστικό με τον υψηλότερο βαθμό είναι το χαρακτηριστικό *PD*, που αφορά σε δεδομένα ανά διεργασία, ακολουθούμενο από το πλήθος των διεργασιών ανά κόμβο, *ppn*, που αναμέναμε να έχει υψηλή επίδραση στο χρόνο επικοινωνίας, εξαιτίας του υψηλού αριθμού πυρήνων ανά κόμβο στο σύστημα ARIS. Το πλήθος των κόμβων δεν επιδρά εξίσου στο ARIS, αντίθετα με τα συστήματα Vilje και Piz Daint. Αυτό οφείλεται στην ασύγχρονη τοπολογία του συστήματος, που επιτρέπει σε πολλούς κόμβους να επικοινωνούν ταυτόχρονα, διατηρώντας υψηλό εύρος ζώνης. Τόσο για το μοντέλο *GBRT-Class B*, όσο και για το μοντέλο *GBRT-Class C*, τα δεδομένα που εγχέονται από κόμβους επηρεάζουν σημαντικά την επικοινωνία, όπως δείχνει ο υψηλός βαθμός των χαρακτηριστικών *ND (avg)* και *ND (max)*. Τα χαρακτηριστικά αυτα ακολουθούνται στην κατάταξη από το πλήθος των διεργασιών



(i) GBRT-Class A

(ii) GBRT-Class B



(iii) GBRT-Class C

Σχήμα 6.1: Κατάταξη χαρακτηριστικών για τα μοντέλα GBRT στο σύστημα ARIS.

ανά κόμβο, χαρακτηριστικό που διατηρεί την υψηλή του σημασία σε όλα τα μοντέλα. Το μοντέλο *GBRT-Class C* χρησιμοποιεί όλα τα χαρακτηριστικά της κατηγορίας Γ με την ίδια, αλλά όχι υψηλή συχνότητα, που υποδεικνύει πως χρησιμοποιούνται στα κατώτερα επίπεδα των δέντρων απόφασης.

6.3 Πειραματική αποτίμηση

6.3.1 Σχήματα επικοινωνίας

Για την αποτίμηση των μοντέλων πρόβλεψης επικοινωνίας που κατασκευάσαμε, χρησιμοποιούμε τρεις εφαρμογές που ενθυλακώνουν διάφορα σχήματα επικοινωνίας. Πειραματιστήκαμε με τις point-to-point φάσεις επικοινωνίας της αντιπροσωπευτικής εφαρμογής LULESH: το σχήμα *LULESH-1*, που εμφανίζει ανταλλαγή μηνυμάτων 27 σημείων σε 3 διαστάσεις, το σχήμα *LULESH-2*, που εμφανίζει ανταλλαγή μηνυμάτων 6 σημείων σε 3 διαστάσεις και το σχήμα *LULESH-3*, που εμφανίζει σχήμα wavefront (αποστολές μηνυμάτων προς μία κατεύθυνση) 27-σημείων. Παραπέμπουμε τον αναγνώστη στην Ενότητα 4.4.1 για περισσότερες πληροφορίες σχετικά με την εφαρμογή και τα σχήματα επικοινωνίας της. Επίσης, πειραματιστήκαμε με το HPCG benchmark (v.2.0) [Dongarra et al., 2015]. Στη μη βελτιστοποιημένη έκδοσή του, το HPCG benchmark επιλύει τον αλγόριθμο συζυγών κλίσεων με προετοιμασία (Preconditioned Conjugate Gradient), σε ένα διακριτό ορθογώνιο πλέγμα τριών διαστάσεων, μεγέθους $nx \times ny \times nz$, από ένα 3Δ-καρτεσιανό πλέγμα διεργασιών μεγέθους $px \times py \times pz$. Το HPCG benchmark εμφανίζει δύο σχήματα point-to-point επικοινωνίας. Το πρώτο, που συμβολίζουμε ως *HPCG-SpMV*, είναι το σχήμα επικοινωνίας του πολλαπλασιασμού αραιού πίνακα με διάνυσμα (SpMV) για τον αραιό πίνακα της HPCG και αποτελεί ανταλλαγή 27 σημείων σε 3 διαστάσεις, παρόμοια με το σχήμα επικοινωνίας *LULESH-1*. Κάθε διεργασία, ανάλογα με τη θέση της στο πλέγμα, μπορεί να ανταλλάσσει έως και 6 όψεις δύο διαστάσεων, έως και 12 ακμές μίας διάστασης και έως 8 κορυφές, με τις γειτονικές της διεργασίες. Το δεύτερο σχήμα επικοινωνίας, που συμβολίζουμε ως *HPCG-MG*, περιλαμβάνει πολλαπλές ανταλλαγές τύπου halo 27 σημείων σε 3Δ-πλέγμα, για διαφορετικά μεγέθη πλέγματος. Πρόκειται για το σχήμα επικοινωνίας της μεθόδου προετοιμασίας (preconditioning) της HPCG, της ιεραρχικής πολυπλεγματικής (multigrid) μεθόδου τριών επιπέδων, που εκτραχύνει το πλέγμα, δηλαδή μειώνει το μέγεθος του προβλήματος σε κάθε επίπεδο κατά έναν συντελεστή $2^{3 \times depth}$, όπου $depth = 0, 1, 2, 3$. Η πολυπλεγματική μέθοδος καλεί τις μεθόδους SpMV και SYMGS και πραγματοποιεί τρεις ανταλλαγές 27 σημείων στο 3Δ-πλέγμα στα επίπεδα 0,1 και 2 και μία ανταλλαγή 27 σημείων στο 3Δ-πλέγμα στο επίπεδο 3. Έτσι, η επικοινωνία της πολυπλεγματικής μεθόδου περιλαμβάνει 10 διακριτές φάσεις επικοινωνίας με ανταλλαγές 27 σημείων στο 3Δ-πλέγμα, για τέσσερα διαφορετικά μεγέθη προβλημάτων, που προκύπτουν από την εκτράχυνση του πλέγματος. Η τρίτη εφαρμογή με την οποία πειραματιστήκαμε είναι ο Πυρήνας D (Kernel D) από τη σουίτα προγραμμάτων PRACE QCD, που στηρίζεται στον κώδικα tmLQCD [Jansen and Urbach, 2009]. Ο πυρήνας πραγματοποιεί πολλαπλασιασμό πίνακα με διάνυ-

Πίνακας 6.3: Λεπτομέρειες του συνόλου ελέγχου στο σύστημα ARIS

	ARIS		
Σχήμα επικοινωνίας	LULESH	HPCG	QCD-Kernel D
Μέγεθος προβλήματος	120^3, 240^3, 480^3	480^3, 960^3	$32 \times 32 \times 32 \times 40$, $32 \times 32 \times 64 \times 40$
Επαναλήψεις	100	100	100
n	9-216	27-256	16-256
ppn	1-20	1-20	1-20
#εκτελέσεις	1	1	1
#σημεία		429	

σμα για τα φερμιόνια Wilson. Η λειτουργία αφορά την εφαρμογή ενός τελεστή Dirac (τον hopping matrix του Wilson) σε ένα διάνυσμα πλέγματος (lattice vector) και απαιτεί την ανταλλαγή των συνοριακών σημείων του διανύσματος πλέγματος (των πεδίων Fermion). Το διάνυσμα είναι τεσσάρων διαστάσεων και το σχήμα επικοινωνίας που προκύπτει, που συμβολίζουμε ως *QCD-KernelD*, αποτελεί ανταλλαγή 3Δ-τεμαχισμών του πεδίου με τους 8 γείτονες σε ένα τετραδιάστατο καρτεσιανό πλέγμα.

Προβλέπουμε το χρόνο επικοινωνίας για τις εφαρμογές *LULESH*, *HPCG* και *QCD-KernelD*, για διάφορα μεγέθη προβλημάτων και πολλαπλές ρυθμίσεις εκτέλεσης. Παρουσιάζουμε τα στοιχεία του συνόλου ελέγχου για το σύστημα ARIS στον Πίνακα 6.3. Η εφαρμογή LULESH από το σχεδιασμό της απαιτεί πλήθος διεργασιών που είναι δύναμη του 3, επομένως όλες οι ρυθμίσεις εκτέλεσης της εφαρμογής LULESH υπόκεινται σε αυτόν τον περιορισμό. Ο ίδιος περιορισμός ισχύει και για την εφαρμογή HPCG, αφού εφαρμόζουμε strong scaling. Ο πυρήνας D της εφαρμογής QCD περιλαμβάνει δύο περιορισμούς για τις διαστάσεις του τοπικού 4Δ-πλέγματος: απαιτεί το μέγεθος του πλέγματος στη διάσταση Z και το γινόμενο των μεγεθών του πλέγματος στις διαστάσεις T, X, και Y να είναι άρτιοι αριθμοί. Επομένως, οι ρυθμίσεις εκτέλεσης για τη συγκεκριμένη εφαρμογή υπόκεινται σε αυτούς τους δύο περιορισμούς.

6.3.2 Σύγκριση των μοντέλων

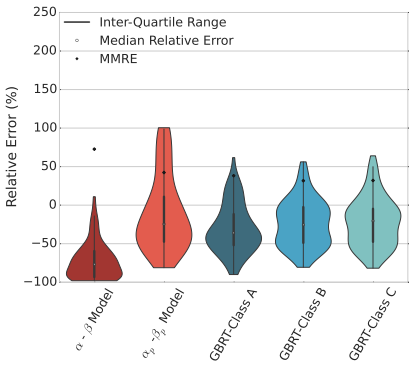
Για την πειραματική αποτίμηση, αρχικά συγκρίνουμε την προβλεπτική ικανότητα των τριών μοντέλων που κατασκευάσαμε με τη χρήση της μεθόδου Gradient Boosting Regression Trees. Για λόγους σύγκρισης, κατασκευάσαμε επιπρόσθετα το αναλυτικό μοντέλο $\alpha - \beta$, που συμβολίζουμε ως $\alpha - \beta$, και το ημι-εμπειρικό μοντέλο $\alpha - \beta$ με ποινές στις παραμέτρους α και β, που συμβολί-

Πίνακας 6.4: Μετρηθείσες παράμετροι για τα αναλυτικά και ημι-εμπειρικά μοντέλα στο σύστημα ARIS

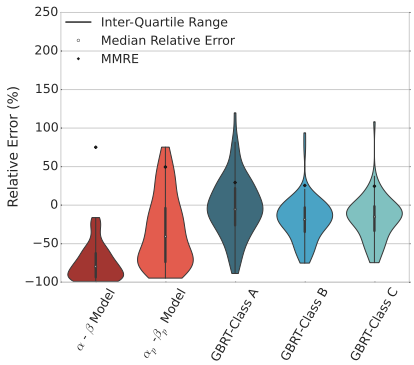
Παράμετρος	Τιμή
α	1.431 us
β	0.195 ns
γ	0.596 us

ζουμε ως $\alpha_p - \beta_p Model$. Αξίζει να σημειώσουμε πως τα ημι-εμπειρικά μοντέλα μπορούν να έχουν πέντε διαφορετικές μορφές, ανάλογα με τις ποινές. Χρησιμοποιήσαμε το μοντέλο $\alpha_p - \beta_p$, αφού εμφάνισε την καλύτερη προσαρμογή στο σύστημα ARIS. Το συγκεκριμένο μοντέλο παραβλέπει την παράμετρο γ, που λειτουργεί ως ποινή στην απόσταση. Δίνουμε τις τιμές των παραμέτρων α, β και γ, όπως τις μετρήσαμε στο σύστημα ARIS, στον Πίνακα 6.4. Παρόλο που η τιμή της παραμέτρου γ είναι υψηλή, σε σύγκριση με την τιμή της παραμέτρου α, η χρήση της στο ημι-εμπειρικό μοντέλο οδηγεί σε κακή ακρίβεια και προσαρμογή.

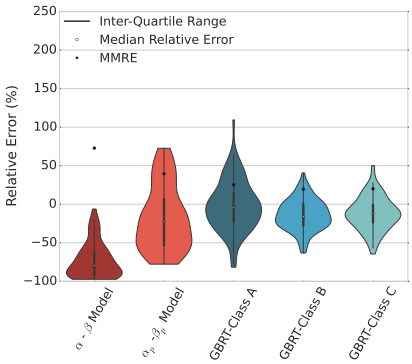
Το Σχήμα 6.2 απεικονίζει την κατανομή των σχετικών σφαλμάτων των προβλέψεων, για τα πέντε μοντέλα, σε κάθε σχήμα επικοινωνίας και συνολικά. Αντίστοιχα, ο Πίνακας 6.5 παρουσιάζει τους βαθμούς των μετρικών $Pred_{0.25}$, R^2 και *RCC*. Μία πρώτη παρατήρηση είναι πως η επίδοση του αναλυτικού μοντέλου, $\alpha - \beta Model$, είναι κακή. Το μοντέλο προβλέπει χαμηλότερο χρόνο επικοινωνίας από τον πραγματικό για όλα τα σχήματα επικοινωνίας και παρουσιάζει ιδιαίτερα χαμηλούς βαθμούς στη μετρική R^2, επομένως οι παράμετροί του δεν επαρκούν για να εξηγήσουν τις διακυμάνσεις του χρόνου επικοινωνίας. Το ημι-εμπειρικό μοντέλο $\alpha_p - \beta_p$ εμφανίζει καλύτερη προβλεπτική ικανότητα από το αναλυτικό μοντέλο και παρουσιάζει πολύ υψηλή ακρίβεια για το σχήμα επικοινωνίας *HPCG-MG*, που είναι το μόνο σχήμα επικοινωνίας στο σύνολο ελέγχου μας που περικλείει πολύ μικρά μηνύματα (< 1 kilobyte), εξαιτίας της εκτράχυνσης του αρχικού πλέγματος. Ταυτόχρονα, εμφανίζει πολύ χαμηλή ακρίβεια στην περίπτωση του σχήματος *QCD-Kernel D*, το οποίο περιλαμβάνει πολλαπλά μεγάλα μηνύματα (της τάξης των εκατοντάδων kilobytes). Μπορούμε, επομένως, να συμπεράνουμε πως το ημι-εμπειρικό μοντέλο στο σύστημα ARIS είναι αποτελεσματικό για προβλέψεις που αφορούν σε μικρά μηνύματα, αλλά αδυνατεί να συλλάβει τη συμπεριφορά και επίδοση της επικοινωνίας για την περίπτωση που η κίνηση δεδομένων στο δίκτυο είναι έντονη.



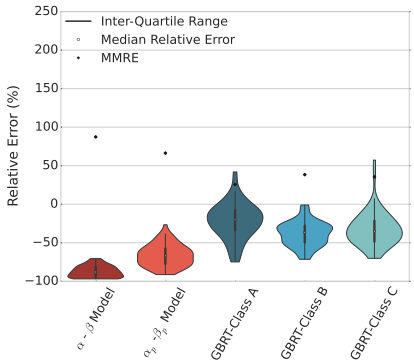
(i) LULESH-1 (27pt-Halo-3D)



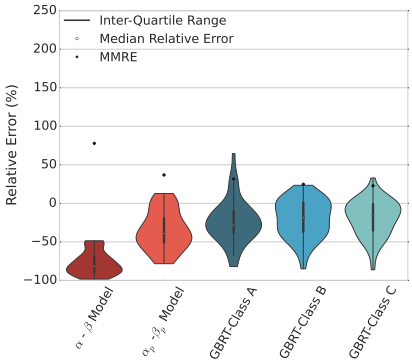
(ii) LULESH-2 (7pt-Halo-3D)



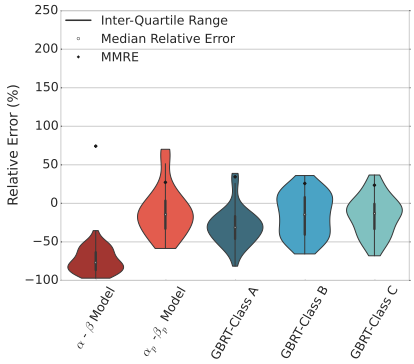
(iii) LULESH-3 (27pt-Wave-3D)



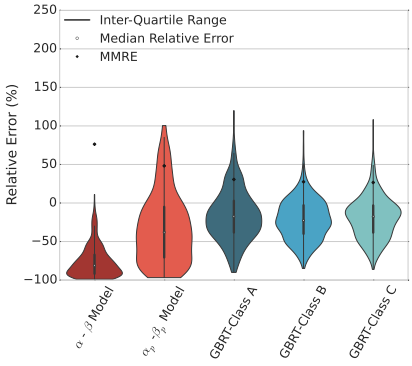
(iv) QCD-Kernel D



(v) HPCG - SpMV (27pt-Halo-3D)



(vi) HPCG - MG (multiple 27pt-Halo-3D)



(vii) Συνολικά

Σχήμα 6.2: Σύγκριση μοντέλων στο σύστημα ARIS.

Πίνακας 6.5: Σύγκριση των μοντέλων με μετρικές ακρίβειας και καλής προσαρμογής στο σύστημα ARIS

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β Model	5.56	-0.664	0.593
α_p -β_p *Model*	27.78	0.191	0.878
GBRT-Class A	30.00	0.603	0.859
GBRT-Class B	42.22	0.574	0.879
GBRT-Class C	43.33	0.549	0.879

(i) LULESH-1

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β Model	5.56	-0.419	0.617
α_p -β_p *Model*	21.11	-0.110	0.830
GBRT-Class A	50.00	0.700	0.871
GBRT-Class B	64.44	0.583	0.902
GBRT-Class C	63.33	0.552	0.901

(ii) LULESH-2

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β Model	5.56	-0.494	0.640
α_p -β_p *Model*	30.00	0.187	0.875
GBRT-Class A	55.56	0.794	0.892
GBRT-Class B	66.67	0.907	0.929
GBRT-Class C	72.22	0.912	0.923

(iii) LULESH-3

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β Model	0.00	-1.026	0.705
α_p -β_p *Model*	0.00	-0.550	0.836
GBRT-Class A	54.41	0.689	0.856
GBRT-Class B	17.65	0.472	0.896
GBRT-Class C	27.94	0.498	0.890

(iv) QCD-Kernel D

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β Model	0.00	-1.002	0.575
α_p -β_p *Model*	34.78	-0.042	0.880
GBRT-Class A	36.96	0.585	0.857
GBRT-Class B	58.70	0.598	0.890
GBRT-Class C	63.04	0.559	0.890

(v) HPCG - SpMV

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β Model	0.00	-1.117	0.552
α_p -β_p *Model*	52.17	0.462	0.871
GBRT-Class A	28.26	0.566	0.869
GBRT-Class B	52.17	0.667	0.871
GBRT-Class C	58.70	0.632	0.875

(vi) HPCG - MG

	$Pred_{0.25}$ (%)	R^2	*RCC*
α - β Model	3.49	-0.111	0.736
α_p -β_p *Model*	25.81	0.562	0.766
GBRT-Class A	43.95	0.779	0.884
GBRT-Class B	50.93	0.801	0.908
GBRT-Class C	54.88	0.786	0.909

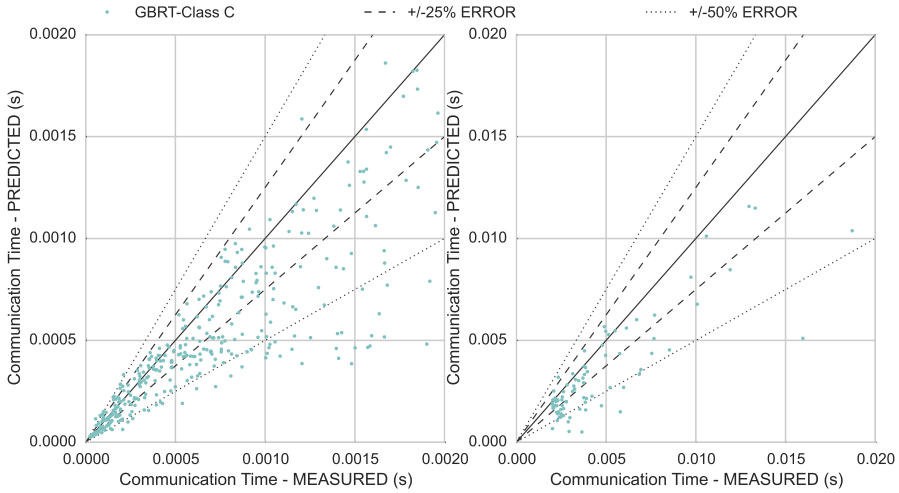
(vii) Συνολικά

Σε ό,τι αφορά τα μοντέλα μας με τη μέθοδο *GBRT*, το μοντέλο της κατηγορίας Α, *GBRT-Class A* αποδίδει καλές προβλέψεις στα περισσότερα σχήματα επικοινωνίας και συνολικά στο σύστημα ARIS. Συγκεκριμένα, αποδίδει καλύτερες προβλέψεις από ό,τι τα μοντέλα *GBRT-Class B* και *GBRT-Class C* στην περίπτωση του σχήματος επικοινωνίας *QCD-Kernel D*. Όπως εξηγήσαμε στην προηγούμενη παράγραφο, το συγκεκριμένο σχήμα επικοινωνίας δημιουργεί έντονη κίνηση δεδομένων στο δίκτυο, ως ένα συμμετρικό σχήμα επικοινω-

νίας με μεγάλα μηνύματα. Οι υψηλές τιμές του επικρατέστερου χαρακτηριστικού, των δεδομένων ανά διεργασία *PD*, στο μοντέλο *GBRT-Class A*, επαρκούν για τη σκιαγράφηση των χρόνων επικοινωνίας στο σύστημα ARIS. Τα μοντέλα *GBRT-Class B* και *GBRT-Class C* εμφανίζουν πολύ υψηλή ακρίβεια σε όλα τα σχήματα επικοινωνίας. Η προσθήκη των χαρακτηριστικών της κατηγορίας Γ στο δεύτερο βελτιώνει τους βαθμούς του στις μετρικές $Pred_{0.25}$ και *RCC*. Ωστόσο, καθώς το επικρατέστερο χαρακτηριστικό και στα δύο μοντέλα είναι αυτό που μετρά έγχυση δεδομένων από τον κόμβο στο δίκτυο, *ND*, οι διαφορές στην ακρίβεια και στην καλή προσαρμογή μεταξύ των δύο μοντέλων δεν είναι σημαντικές. Συνολικά, παρατηρούμε μία προκατάληψη (bias) σε όλα τα μοντέλα μας με τη μέθοδο GBRT, προς την πρόβλεψη χαμηλότερων χρόνων επικοινωνίας από τους πραγματικούς. Αυτό οφείλεται στις ιδιότητες του συνόλου εκπαίδευσής μας και στον τρόπο με τον οποίο η μέθοδος GBRT παράγει το τελικό μοντέλο: το σύνολο εκπαίδευσής μας περιλαμβάνει μόνο ρυθμίσεις εκτέλεσης με πλήθη κόμβων που αποτελούν δυνάμεις του δύο και πέντε διαφορετικές τιμές για το πλήθος των διεργασιών ανά κόμβο. Το σύνολο ελέγχου μας περιλαμβάνει περισσότερο "ασύμμετρες" ρυθμίσεις εκτέλεσης, εξαιτίας των περιορισμών που θέτουν οι εφαρμογές. Η μέθοδος GBRT πραγματοποιεί παρεμβολή (interpolation) μεταξύ των τιμών του συνόλου εκπαίδευσης, καταλήγοντας σε αυτό το bias προς χαμηλότερες τιμές του χρόνου επικοινωνίας για ορισμένες ρυθμίσεις εκτέλεσης. Η επέκταση του συνόλου εκπαίδευσης με τις επιπλέον ρυθμίσεις εκτέλεσης θα μπορούσε να βοηθήσει στη μείωση αυτής της προκατάληψης.

6.3.3 Λεπτομερής αποτίμηση

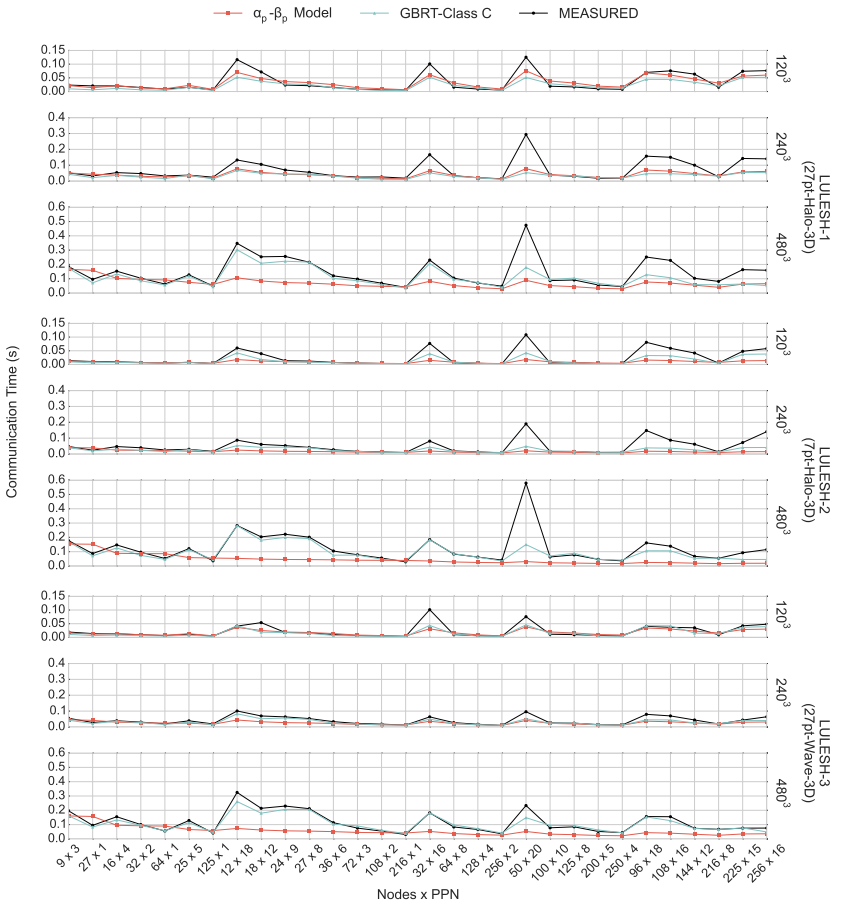
Από τη σύγκριση των μοντέλων μας, επιλέγουμε το μοντέλο *GBRT-Class C* ως το μοντέλο με τη βέλτιστη προβλεπτική ικανότητα στο σύστημα ARIS. Το μοντέλο αυτό προσφέρει προβλέψεις ακριβώς πριν την εκτέλεση της εφαρμογής και παρουσιάζει υψηλή ακρίβεια και βέλτιστη προσαρμογή, με βαθμούς 54.88% για τη μετρική $Pred_{0.25}$, 0.786 για τη μετρική R^2 και 0.909 για τη μετρική *RCC*, συνολικά. Επιπλέον, ελαττώνει τη μέση τιμή των απολύτων τιμών των σχετικών σφαλμάτων *MMRE* σε 26.61%, από 44.84% του ημι-εμπειρικού μοντέλου $\alpha_p - \beta_p$. Αναλύουμε περαιτέρω την επίδοση των προβλέψεων του συγκεκριμένου μοντέλου, τόσο συνολικά, όσο και διακριτά σε κάθε σχήμα επικοινωνίας. Αρχικά, εξετάζουμε όλα τα σημεία του συνόλου ελέγχου στη μορφή διαγραμμάτων διασποράς, που απεικονίζονται στο Σχήμα 6.3. Στο Σχήμα, οι χρόνοι επικοινωνίας έχουν κανονικοποιηθεί σε μία επανάληψη και το σύνολο των σημείων έχει διαιρεθεί σε δύο υποσύνολα, με βάση το χρόνο επικοινωνίας, για λόγους ευκρίνειας. Η πλειονότητα των προβλέψεων (54.88%) κείται εντός του εύρους σφαλμάτων $\pm 25\%$, ενώ το 85.82% των προβλέψεων εμφα-



Σχήμα 6.3: Διαγράμματα διασποράς για τις προβλέψεις του μοντέλου *GBRT-Class C* στο ARIS. Ο χρόνος επικοινωνίας έχει κανονικοποιηθεί σε μία επανάληψη.

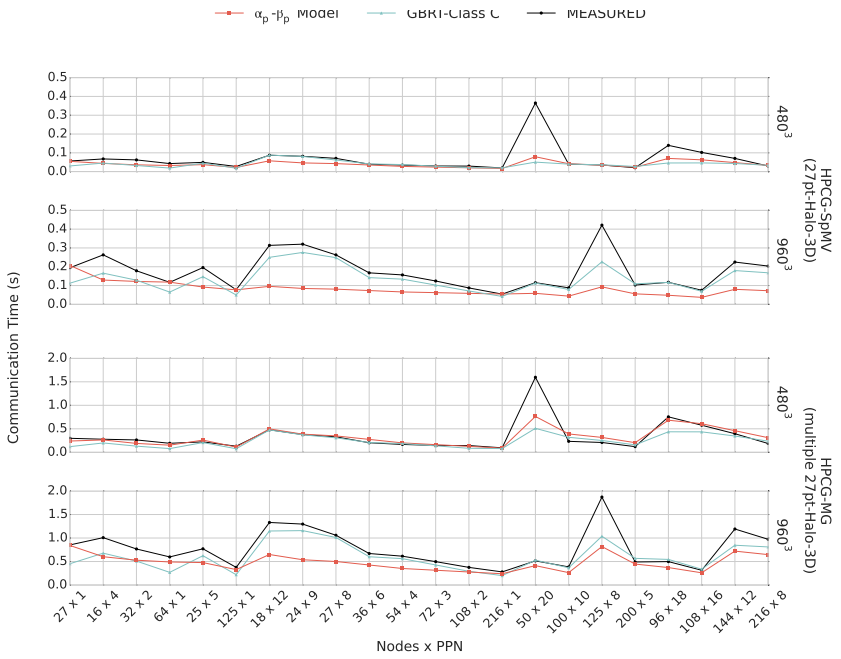
νίζουν σχετικά σφάλματα στο εύρος$\pm 50\%$. 57 σημεία εμφανίζουν προβλέψεις του χρόνου επικοινωνίας με μικρότερο χρόνο από τον πραγματικό, με σχετικά σφάλματα μικρότερα του -50%, ενώ το 75% αυτών έχει χρόνο επικοινωνίας μικρότερο από 2 ms. Επομένως συμπεραίνουμε πως, παρόλο που το μοντέλο μας δείχνει κάποια προκατάληψη προς υποεκτιμήσεις του πραγματικού χρόνου επικοινωνίας, αυτές προκύπτουν κυρίως όταν ο χρόνος επικοινωνίας των σχημάτων στο σύνολο ελέγχου μας είναι ασήμαντα μικρός.

Ακολούθως, εξετάζουμε τις προβλέψεις του μοντέλου *GBRT-Class C* σε όλα τα σχήματα επικοινωνίας. Στο Σχήμα 6.4 παρουσιάζουμε τις προβλέψεις για τρία σχήματα επικοινωνίας και τα τρία μεγέθη προβλημάτων της εφαρμογής LULESH, για όλες τις ρυθμίσεις εκτέλεσης στο σύνολο ελέγχου, ταξινομημένες με βάση τον αριθμό των πυρήνων. Το μοντέλο μας επιτυγχάνει πολύ καλές προβλέψεις και συλλαμβάνει τη συμπεριφορά της κλιμάκωσης της επικοινωνίας και για τα τρία σχήματα επικοινωνίας της εφαρμογής LULESH. Ρυθμίσεις εκτέλεσης με περισσότερους από 1000 πυρήνες εμφανίζουν υποεκτιμήσεις του χρόνου επικοινωνίας για τα σχήματα *LULESH-1* και *LULESH-2* και ειδικότερα για τα δύο μικρότερα μεγέθη προβλημάτων (120^3 και 240^3), όπου τα μεγέθη των μηνυμάτων είναι μικρά. Δύο ρυθμίσεις εκτέλεσης με μικρότερο αριθμό πυρήνων (32×16 και 50×20) εμφανίζουν συστηματικά αρνητικά σχετικά σφάλματα για όλα τα σχήματα επικοινωνίας και όλα τα μεγέθη προβλημάτων. Και οι δύο αφορούν σε μεγάλο πλήθος διεργασιών ανά κόμβο και ο υψηλός χρόνος επικοινωνίας υποδηλώνει πως το μοντέλο μας δεν κα-



Σχήμα 6.4: Προβλέψεις για την εφαρμογή LULESH με το μοντέλο *GBRT-Class C* στο ARIS.

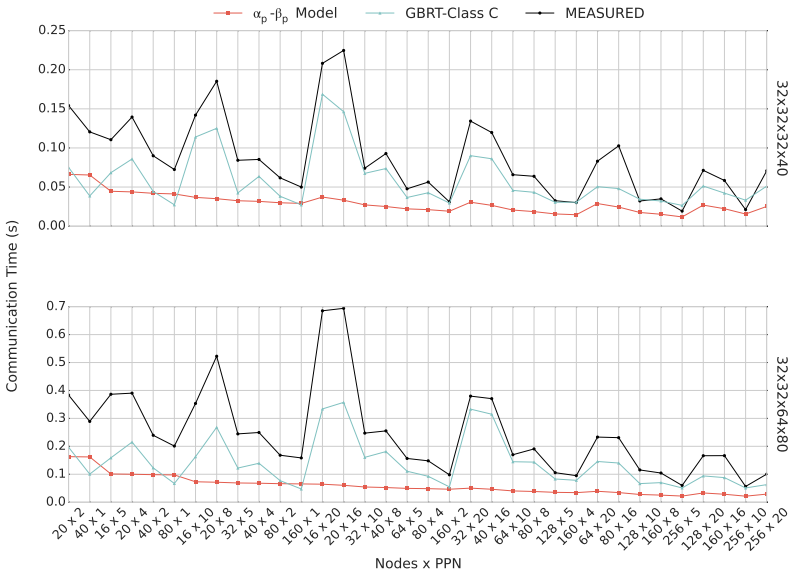
τορθώνει να συλλάβει επαρκώς φαινόμενα ανταγωνισμού που οφείλονται στη συνύπαρξη πολλαπλών διεργασιών ανά κόμβο, παρόλο που το σύνολο εκπαίδευσής μας εμπεριέχει αντίστοιχες ρυθμίσεις εκτέλεσης. Πιστεύουμε πως ορισμένα φαινόμενα ανταγωνισμού προκύπτουν από τη συνύπαρξη των φάσεων επικοινωνίας με τις φάσεις υπολογισμών της εφαρμογής και αποτελούν κατά κύριο λόγο φαινόμενα στη μνήμη και όχι στο δίκτυο. Συγκρίνουμε τις προβλέψεις μας με τις προβλέψεις του μοντέλου $\alpha_p - \beta_p$, που αποτυγχάνει να σκιαγραφήσει την κλιμάκωση των σχημάτων επικοινωνίας και προβλέπει ελάχιστες διακυμάνσεις του χρόνου επικοινωνίας για τις διαφορετικές ρυθμίσεις



Σχήμα 6.5: Προβλέψεις για την εφαρμογή HPCG με το μοντέλο *GBRT-Class C* στο ARIS.

εκτέλεσης.

Το Σχήμα 6.5 εικονίζει τις προβλέψεις για τα δύο σχήματα point-to-point επικοινωνίας της εφαρμογής HPCG, το σχήμα *HPCG-SpMV* και το σχήμα *HPCG-MG*. Το μοντέλο μας προβλέπει με ακρίβεια την κλιμάκωση της επικοινωνίας για τα δύο σχήματα και για τα δύο μεγέθη προβλημάτων, 480^3 και 960^3. Για το σχήμα επικοινωνίας *HPCG-SpMV*, που είναι παρόμοιο με το σχήμα επικοινωνίας *LULESH-1*, παρατηρούμε σημαντικά χαμηλότερες προβλέψεις του χρόνου επικοινωνίας από τους πραγματικούς για την ίδια ρύθμιση εκτέλεσης, 50×20 (κόμβοι $\times$ διεργασίες ανά κόμβο). Το σχήμα επικοινωνίας *HPCG-MG* επίσης εμφανίζει μία αντίστοιχη πρόβλεψη χρόνου επικοινωνίας, με τιμή σημαντικά χαμηλότερη από τον πραγματικό χρόνο επικοινωνίας, για μία ρύθμιση εκτέλεσης, σε 125×8 κόμβους $\times$ διεργασίες ανά κόμβο, που δεν παρουσιάζει κάποια ομοιότητα με τις άλλες ρυθμίσεις εκτέλεσης με συστηματικά μεγάλα αρνητικά σφάλματα πρόβλεψης, επομένως θεωρούμε τη συγκεκριμένη τιμή ακραία. Το μοντέλο $\alpha_p - \beta_p$ αποδίδει καλές προβλέψεις για το μικρότερο μέγεθος προβλήματος της HPCG, 480^3, όπου τα μεγέθη των μηνυμάτων είναι σημαντικά μικρότερα από αυτά της εφαρμογής LULESH, αποτυγχάνει ωστόσο να αποδώσει τη συμπεριφορά της κλιμάκωσης της επικοινωνίας για



Σχήμα 6.6: Προβλέψεις για την εφαρμογή QCD-Kernel D με το μοντέλο *GBRT-Class C* στο ARIS.

το μεγαλύτερο μέγεθος προβλήματος.

Τέλος, το Σχήμα 6.6 απεικονίζει τις προβλέψεις για την εφαρμογή *QCD-Kernel D*, για τα δύο μεγέθη προβλημάτων στο σύνολο ελέγχου. Στο συγκεκριμένο σχήμα επικοινωνίας, το μοντέλο μας παρουσιάζει εξαιρετικά καλή προσαρμογή, αφού προβλέπει άριστα όλες τις αυξομοιώσεις του χρόνου επικοινωνίας για οποιαδήποτε ρύθμιση εκτέλεσης. Επιπλέον, εμφανίζει πολύ καλή ακρίβεια για μεγάλα πλήθη πυρήνων και στα δύο μεγέθη προβλημάτων. Αντιθέτως, το μοντέλο $\alpha_p - \beta_p$ προβλέπει λανθασμένα πολύ χαμηλές τιμές του χρόνου επικοινωνίας της εφαρμογής *QCD-Kernel D* και δεν αποδίδει τις μεταβολές του χρόνου επικοινωνίας μεταξύ των διάφορων ρυθμίσεων εκτέλεσης.

6.3.4 Λήψη αποφάσεων με επίγνωση της επικοινωνίας

Αποτιμούμε επιπλέον την ακρίβεια των μοντέλων πρόβλεψης για το χρόνο επικοινωνίας, στο πλαίσιο της απόδοσης ακριβών προβλέψεων για τη λήψη αποφάσεων με επίγνωση της επικοινωνίας. Χρησιμοποιούμε το μοντέλο *GBRT-Class C* για την πρόβλεψη σημείων που είναι κρίσιμα σε σενάρια λήψης αποφάσεων για τη βέλτιστη χρήση των πόρων από την πλευρά του χρήστη. Όπως εξηγούμε στην Ενότητα 4.4.4, οι χρήστες των υπερυπολογιστικών συστημά-

Πίνακας 6.6: Μέτωπα Παρέτο για ελαχιστοποίηση πυρήνων και χρόνου επικοινωνίας στο σύστημα ARIS

Σχήμα επικοινωνίας	Μέγεθος προβλήματος	Ρυθμίσεις εκτέλεσης μετώπου Παρέτο	Προβλεφθείσες ρυθμίσεις εκτέλεσης	Matches (Ταιριάσματα)	Απόσταση (μέγιστη)
LULESH-1	120^3	4	4	4	-
LULESH-1	240^3	5	4	4	0
LULESH-1	480^3	4	4	4	-
LULESH-2	120^3	4	5	4	1
LULESH-2	240^3	4	5	4	1
LULESH-2	480^3	4	5	4	1
LULESH-3	120^3	4	4	4	-
LULESH-3	240^3	5	5	5	-
LULESH-3	480^3	4	4	4	-
HPCG-SpMV	480^3	4	3	3	0
HPCG-SpMV	960^3	4	4	4	-
HPCG-MG	480^3	4	3	3	0
HPCG-MG	960^3	4	4	4	-
QCD-Kernel D	$32 \times 32 \times 32 \times 40$	6	4	4	0
QCD-Kernel D	$32 \times 32 \times 64 \times 80$	7	3	3	0

των επιδιώκουν να βελτιστοποιήσουν τη χρήση του συστήματος, μεγιστοποιώντας την επίδοση των εφαρμογών τους, ελαχιστοποιώντας την κατανάλωση του προϋπολογισμού τους και ελαχιστοποιώντας το χρόνο αναμονής τους στις ουρές του συστήματος. Για να επιτύχουν αυτούς τους στόχους, πρέπει να είναι σε θέση να επιλέγουν βέλτιστες κατανομές (ρυθμίσεις εκτέλεσης) κόμβων, διεργασιών ανά κόμβο ή και νημάτων OpenMP. Αντιμετωπίζουμε το συγκεκριμένο σενάριο λήψης αποφάσεων από την πλευρά της επικοινωνίας των παράλληλων εφαρμογών, χρησιμοποιώντας το μοντέλο μας *GBRT-Class C* για την πρόβλεψη των βέλτιστων κατανομών κόμβων/διεργασιών ανά κόμβο, δηλαδή για την επιλογή εκείνων των ρυθμίσεων εκτέλεσης που ελαχιστοποιούν το χρόνο επικοινωνίας για έναν συγκεκριμένο αριθμό πυρήνων. Η πολιτική χρέωσης των χρηστών στο σύστημα ARIS γίνεται στη βάση των ωρών πυρήνων (core-hours). Επομένως, διατυπώνουμε το πρόβλημα ως πρόβλημα ταυτόχρονης ελαχιστοποίησης του αριθμού των πυρήνων και του χρόνου επικοινωνίας και χρησιμοποιούμε την έννοια των μη-κυριαρχούντων (non-dominated) μετώπων Παρέτο για την επιλογή του συνόλου των κατά Παρέτο βέλτιστων ρυθμίσεων κόμβων και χρόνου επικοινωνίας (c, t). Επιπλέον, χρησιμοποιούμε υποβέλτιστα μέτωπα Παρέτο και την απόστασή τους από το βέλτιστο μέτωπο Παρέτο για να αποτιμήσουμε τις προβλέψεις μας. Παραπέμπουμε τον αναγνώστη στην Ενότητα 4.4.4 για τη λεπτομερή περιγραφή του μετώπου Παρέτο και των ιδιοτήτων του.

Το Σχήμα 6.7 δείχνει δύο παραδείγματα μετώπων Παρέτο στο σύστημα ARIS. Κάθε γράφημα δείχνει τον πραγματικό χρόνο επικοινωνίας για όλες τις ρυθμίσεις εκτέλεσης, σε ένα συγκεκριμένο σχήμα επικοινωνίας και μέγεθος προβλήματος στο ARIS. Το μέτωπο Παρέτο και το δεύτερο βέλτιστο μέτωπο Παρέτο (με απόσταση ίση με 1) επισημαίνονται στο γράφημα. Το πρώτο

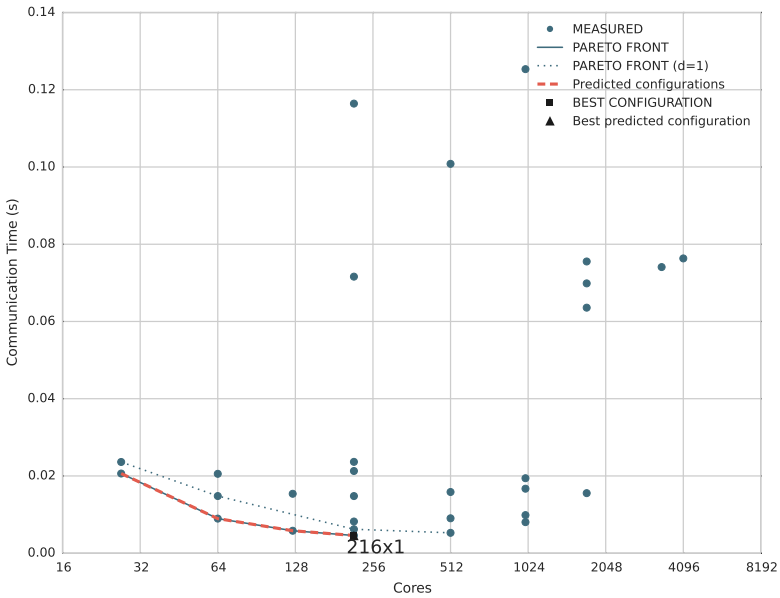
γράφημα, στο Σχήμα 6.7i, δείχνει την περίπτωση του σχήματος επικοινωνίας *LULESH-1* στο μικρότερο μέγεθος προβλήματος, όπου οι προβλέψεις μας είναι επιτυχείς και όλες οι προβλεφθείσες ρυθμίσεις εκτέλεσης συμπίπτουν με τις κατά Παρέτο βέλτιστες ρυθμίσεις εκτέλεσης, όπως τις υπολογίσαμε με βάση τους πραγματικούς χρόνους επικοινωνίας. Το Σχήμα 6.7ii δείχνει μία περίπτωση όπου το μοντέλο μας δεν προβλέπει με ακρίβεια το μέτωπο Παρέτο: το μέτωπο Παρέτο περιλαμβάνει 4 σημεία, ενώ το μοντέλο μας προβλέπει 5 κατά Παρέτο βέλτιστες ρυθμίσεις εκτέλεσης. Τα 4 προβλεφθέντα σημεία αντιστοιχούν σε σημεία του μετώπου Παρέτο. Το πέμπτο σημείο στην πραγματικότητα ανήκει στο δεύτερο βέλτιστο μέτωπο Παρέτο (με απόσταση ίση με 1). Ο Πίνακας 6.6 συνοψίζει τα αποτελέσματα των σημείων των μετώπων Παρέτο με βάση τον πραγματικό και προβλεφθέντα χρόνο επικοινωνίας, σε σχέση με το πλήθος των πυρήνων. Παραπέμπουμε τον αναγνώστη στο Παράρτημα Γ για τα σχετικά διαγράμματα. Σημειώνουμε ως "Matches" (ταιριάσματα) το πλήθος των προβλεφθεισών ρυθμίσεων εκτέλεσης που συμπίπτουν με τις ρυθμίσεις εκτέλεσης στο μέτωπο Παρέτο. Στις περιπτώσεις όπου οι προβλεφθείσες ρυθμίσεις εκτέλεσης δε συμπίπτουν με αυτές του μετώπου Παρέτο, αναθέτουμε σε κάθε ρύθμιση εκτέλεσης που δε συμπίπτει ένα βαθμό d που αντιστοιχεί στην απόσταση του υπο-βέλτιστου μετώπου Παρέτο όπου ανήκει. Στον πίνακα αναφέρουμε τη μέγιστη απόσταση των προβλεφθέντων σημείων που δε συμπίπτουν. Για τα 15 μέτωπα Παρέτο που κατασκευάσαμε, το πραγματικό και το προβλεφθέν μέτωπο Παρέτο ταιριάζουν απόλυτα σε 7 περιπτώσεις. Για τις υπόλοιπες περιπτώσεις, τα δύο μέτωπα διαφέρουν το πολύ κατά 4 σημεία (βλ. *QCD-Kernel D* στο μέγεθος προβλήματος $32 \times 32 \times 64 \times 80$). Για 5 από τα μέτωπα Παρέτο που δε συμπίπτουν, η μέγιστη απόσταση είναι 0: οι προβλεφθείσες ρυθμίσεις εκτέλεσης ανήκουν στο μέτωπο Παρέτο, αλλά το μοντέλο μας δεν κατορθώνει να εντοπίσει όλα τα σημεία του μετώπου Παρέτο. Για τις υπόλοιπες 3 περιπτώσεις, η μέγιστη απόσταση είναι 1. Για λόγους σύγκρισης, προβλέψαμε τα σημεία των μετώπων Παρέτο με τη χρήστη του ημι-εμπειρικού μοντέλου $\alpha_p - \beta_p$ και παρατηρήσαμε πως σε 9 από τις 15 περιπτώσεις, τα προβλεφθέντα σημεία ταιριάζουν ακριβώς στα αντίστοιχα σημεία των μετώπων Παρέτο. Επομένως, παρά τα σφάλματα πρόβλεψης, το μοντέλο $\alpha_p - \beta_p$ επιτυγχάνει να προβλέψει κρίσιμες ρυθμίσεις εκτέλεσης στο σύστημα ARIS.

Τα μέτωπα Παρέτο για το πλήθος των πυρήνων και το χρόνο επικοινωνίας περικλείουν επιπλέον ενδιαφέρουσα πληροφορία. Το σημείο με τον υψηλότερο αριθμό πυρήνων που περιλαμβάνεται στο μέτωπο Παρέτο αντιστοιχεί στη ρύθμιση εκτέλεσης που καταλήγει στον ελάχιστο χρόνο επικοινωνίας. Αν το σχήμα επικοινωνίας ή/και το μέγεθος προβλήματος υπό εξέταση δεν κλιμακώνει, τότε το σημείο με το υψηλότερο πλήθος πυρήνων στο μέτωπο Παρέτο αντιστοιχεί στο μέγιστο πλήθος των πυρήνων που μπορεί να αξιοποιήσει η εφαρμογή πριν σταματήσει η κλιμάκωση του συγκεκριμένου σχήματος επι-

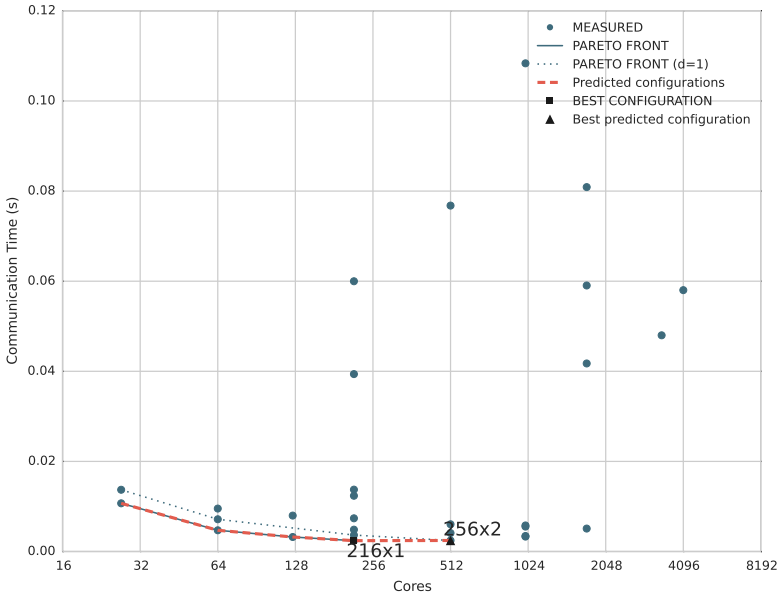
Πίνακας 6.7: Ρυθμίσεις εκτέλεσης ελάχιστου χρόνου επικοινωνίας στο σύστημα ARIS

Σχήμα επικοινωνίας	**Μέγεθος προβλήματος**	**Μέτρηση** ($n \times ppn$)	**Πρόβλεψη** ($n \times ppn$)	**Αποτέλεσμα**	**Απόσταση**
LULESH-1	120^3	216×1	216×1	**True**	-
LULESH-1	240^3	256×2	216×1	False	0
LULESH-1	480^3	216×1	216×1	**True**	-
LULESH-2	120^3	216×1	256×2	False	1
LULESH-2	240^3	256×2	256×2	**True**	-
LULESH-2	480^3	216×1	256×2	False	1
LULESH-3	120^3	216×1	216×1	**True**	-
LULESH-3	240^3	216×1	216×1	**True**	-
LULESH-3	480^3	216×1	216×1	**True**	-
HPCG-SpMV	480^3	216×1	125×1	False	0
HPCG-SpMV	960^3	216×1	216×1	**True**	-
HPCG-MG	480^3	216×1	125×1	False	0
HPCG-MG	960^3	216×1	216×1	**True**	-
QCD-Kernel D	$32 \times 32 \times 32 \times 40$	256×5	256×5	**True**	-
QCD-Kernel D	$32 \times 32 \times 64 \times 80$	256×10	160×1	False	0

κοινωνίας. Επισημαίνουμε τις βέλτιστες ρυθμίσεις εκτέλεσης στο Σχήμα 6.7, όπως αυτές προκύπτουν από τις μετρήσεις και από τις προβλέψεις του χρόνου επικοινωνίας. Ο Πίνακας 6.7 δείχνει τις πραγματικές και τις προβλεφθείσες ρυθμίσεις εκτέλεσης με τον ελάχιστο χρόνο επικοινωνίας για όλα τα σχήματα επικοινωνίας στο ARIS. Χρησιμοποιούμε ξανά την έννοια της απόστασης του μετώπου Παρέτο για να αξιολογήσουμε τις εσφαλμένα προβλεφθείσες ρυθμίσεις εκτέλεσης. Το μοντέλο μας προβλέπει σωστά τη ρύθμιση εκτέλεσης με το μικρότερο χρόνο επικοινωνίας για 9 από τις 15 περιπτώσεις. Οι εσφαλμένες προβλέψεις που προκύπτουν στις εφαρμογές LULESH και HPCG αφορούν σε γειτονικές ρυθμίσεις εκτέλεσης, όπου ο χρόνος επικοινωνίας διαφέρει κατά 1 ms. Στο ίδιο σενάριο, το μοντέλο $\alpha_p - \beta_p$ προβλέπει σωστά ίδιο πλήθος ρυθμίσεων εκτέλεσης, δηλαδή 9 από τις 15 ρυθμίσεις εκτέλεσης με τον ελάχιστο χρόνο επικοινωνίας.



(i) Όλα τα σημεία συμπίπτουν: *LULESH-1* με μέγεθος προβλήματος 120^3.



(ii) 4 σημεία συμπίπτουν: *LULESH-2* με μέγεθος προβλήματος 120^3.

Σχήμα 6.7: Παραδείγματα πρόβλεψης των κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το χρόνο επικοινωνίας στο ARIS.

7

Συμπεράσματα

Στην παρούσα διατριβή, προτείναμε και παρουσιάσαμε μία μεθοδολογία για την κατασκευή μοντέλων πρόβλεψης της επίδοσης της επικοινωνίας παράλληλων εφαρμογών μεγάλης κλίμακας. Ακολουθώντας μία εμπειρική προσέγγιση για τη μοντελοποίηση με στόχο την πρόβλεψη, ορίσαμε χαρακτηριστικά για την επίδοση της επικοινωνίας σε τρία υπολογιστικά συστήματα μεγάλης κλίμακας: το Vilje, έναν υπερυπολογιστή SGI Altix που χρησιμοποιεί το δίκτυο InfiniBand σε τοπολογία ενισχυμένου υπερκύβου, το Piz Daint, έναν υπερυπολογιστή Cray XC30 (πλέον Cray XC50), που χρησιμοποιεί το δίκτυο Cray Aries σε τοπολογία Dragonfly, και το ARIS, μία συστοιχία υπολογιστών μεγάλης κλίμακας IBM NextScale, που χρησιμοποιεί το δίκτυο InfiniBand σε τοπολογία fat-tree. Συλλέξαμε ένα σύνολο μετρήσεων σε κάθε σύστημα, με τη χρήση ενός γενικού προγράμματος συλλογής μετρήσεων αναφοράς για point-to-point επικοινωνία (από σημείο σε σημείο), και κατασκευάσαμε διάφορα μοντέλα πρόβλεψης για το χρόνο επικοινωνίας, αξιοποιώντας μεθόδους στατιστικής και μηχανικής μάθησης.

Κατά τη διαδικασία της μοντελοποίησης, εντοπίσαμε χαρακτηριστικά που επηρεάζουν την επίδοση της επικοινωνίας σε κάθε σύστημα. Στο Vilje, οι ροές δεδομένων μέσω των υπολογιστικών κόμβων και των διακοπτών του δικτύου InfiniBand επηρεάζουν πρωταρχικά το χρόνο επικοινωνίας. Στο Piz Daint, ο όγκος των δεδομένων που ρέει σε οποιοδήποτε σημείο του συστήματος, δηλαδή πυρήνες, κόμβους, Aries SoCs, chassis και ομάδες της τοπολογίας Dragonfly, επιδρά στο χρόνο επικοινωνίας. Στο σύστημα ARIS, η κίνηση που παράγεται από κάθε κόμβο κυριαρχεί στην πρόβλεψη του χρόνου επικοινωνίας.

Αποτιμήσαμε την προβλεπτική ικανότητα των μοντέλων μας σε διάφορα σχήματα point-to-point επικοινωνίας που εμφανίζονται σε παράλληλες εφαρμογές μεγάλης κλίμακας και συμπεραίνουμε πως τα πολλαπλά, καλώς ορισμένα χαρακτηριστικά, ο γενικός τρόπος συλλογής μετρήσεων αναφοράς και

οι αυτοματοποιημένες μέθοδοι μηχανικής μάθησης, όπως η μέθοδος Gradient Boosting Regression Trees, αποδίδουν ένα μοντέλο πρόβλεψης για το χρόνο επικοινωνίας που εμφανίζει υψηλή ακρίβεια και καλή προσαρμογή, για όλα τα σχήματα επικοινωνίας και όλες τις ρυθμίσεις εκτέλεσης στην πειραματική μας αποτίμηση, και για τα τρία συστήματα. Η μεθοδολογία μας αποδίδει προβλέψεις ακριβώς πριν την εκτέλεση μιας εφαρμογής (just-in-time) και είναι κατάλληλη για τη λήψη αποφάσεων που αφορά στη βελτιστοποίηση της χρήσης των πόρων των υπερυπολογιστικών συστημάτων.

Εξ όσων γνωρίζουμε, αυτή είναι η πρώτη εργασία που παρουσιάζει μία μεθοδολογία για την πρόβλεψη του χρόνου επικοινωνίας παράλληλων εφαρμογών μεγάλης κλίμακας, πριν την εκτέλεσή τους, για οποιαδήποτε εφαρμογή σε οποιοδήποτε σύστημα, με τη δυνατότητα να αποδίδει ακριβείς προβλέψεις, όπως καταδεικνύεται από τις μετρικές *MMRE*, *RCC* και $Pred_{0.25}$, που λαμβάνουν τιμές 23.98%, 0.922, 0.942 και 61.43% στο Vilje, 21.32%, 0.986, 0.940 και 66.57% στο Piz Daint και 26.61%, 0.786, 0.909 και 54.88% στο ARIS. Τα μοντέλα μας βελτιώνουν σημαντικά την ακρίβεια πρόβλεψης, σε σχέση με άλλες ημι-εμπειρικές προσεγγίσεις για την πρόβλεψη του χρόνου επικοινωνίας, μειώνοντας τη μέση τιμή των απολύτων τιμών των σχετικών σφαλμάτων *MMRE* κατά 50% συνολικά. Επιπλέον, αποτιμήσαμε την ακρίβεια των μοντέλων μας σε σενάρια λήψης αποφάσεων με επίγνωση της επικοινωνίας, με στόχο την επιλογή ρυθμίσεων εκτέλεσης που είναι βέλτιστες για την επικοινωνία. Παρατηρήσαμε σημαντική ακρίβεια, ιδιαίτερα για τα συστήματα Vilje και Piz Daint, όπου εναλλακτικά ημι-εμπειρικά μοντέλα αποτυγχάνουν να προβλέψουν τις βέλτιστες ρυθμίσεις εκτέλεσης.

Στις μελλοντικές κατευθύνσεις της παρούσας διατριβής περιλαμβάνεται η επέκταση της μεθοδολογίας που προτείναμε για την πρόβλεψη της επίδοσης της συλλογικής επικοινωνίας. Η μοντελοποίηση της συλλογικής επικοινωνίας μοιράζεται πολλές από τις προκλήσεις της μοντελοποίησης της point-to-point επικοινωνίας, ωστόσο το σχήμα επικοινωνίας είναι προκαθορισμένο. Επομένως, αναμένουμε ότι κάποια συμμετρικά σχήματα συλλογικής επικοινωνίας είναι δυνατό να μοντελοποιηθούν εύκολα με ένα ή λίγα χαρακτηριστικά, όπως, π.χ., το πλήθος των διεργασιών, όπως καταδεικνύει και η εργασία των Shudler και συνεργατών [Shudler et al., 2015], ενώ άλλα σχήματα συλλογικής επικοινωνίας μπορούν να μοντελοποιηθούν ως point-to-point επικοινωνία, αφού οι υλοποιήσεις του MPI εσωτερικά χρησιμοποιούν επίσης point-to-point επικοινωνία για πολλές ρουτίνες ασύμμετρης συλλογικής επικοινωνίας. Μία επιπλέον σημαντική μελλοντική κατεύθυνση είναι η επέκταση της παρούσας μεθοδολογίας για την πρόβλεψη της επίδοσης σε τάξεις μεγεθών πυρήνων και νημάτων κατά πολύ μεγαλύτερων από αυτών που υπήρξαν διαθέσιμες στο πλαίσιο της παρούσας εργασίας. Τέλος, στοχεύουμε να αποδώσουμε μια εργαλειοθήκη για την αυτόματη πρόβλεψη του χρόνου επικοινωνίας σε υπολογι-

στικά συστήματα μεγάλης κλίμακας και να εργαστούμε στην κατεύθυνση της ενσωμάτωσης της πρόβλεψης της επικοινωνίας σε λογισμικό διαχείρισης πόρων.

Κατάλογος δημοσιεύσεων

Κατά τη διάρκεια της διατριβής αυτής παρήχθη ένα σύνολο δημοσιεύσεων σε έγκριτα διεθνή περιοδικά και συνέδρια υψηλής ποιότητας της κοινότητας HPC.

Διεθνή περιοδικά

- N. Papadopoulou, G. Goumas, N. Koziris: "Predictive communication modeling for HPC applications", Cluster Computing, Volume 20, Issue 3, pp. 2725-2747, 2017.
- A. Haritatos, N. Papadopoulou, K. Nikas, G. Goumas, N. Koziris: "Contention-Aware Scheduling Policies for Fairness and Throughput", Co-Scheduling of HPC Applications 28 (2017): 22

Διεθνή συνέδρια

- N. Papadopoulou, L. Oden, P. Balaji, "A performance study of UCX over InfiniBand", Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 345-354. IEEE Press, 2017
- N. Papadopoulou, G. Goumas, N. Koziris, "Optimizing resource utilization on large-scale systems through predictive communication modelling", NovExS-RAMbO Workshop on Novel Exascale Systems / Rack-Scale Memory Systems and Models, HiPEAC 2017, 2017
- N. Papadopoulou, G. Goumas, N. Koziris: "A machine-learning approach for communication prediction on large-scale applications", Cluster Computing (CLUSTER), 2015 IEEE International Conference on. IEEE, 2015
- A. Abdel-Rehim, C. Alexandrou, N. Anastopoulos, G. Koutsou, I. Liabotis, N. Papadopoulou, "PLQCD library for Lattice QCD on multi-core machines", Proceedings of the 31st International Symposium on Lattice Field Theory (LATTICE 2013). 29 July-3 August, 2013. Mainz, Germany.

Published online at http://pos. sissa. it/cgi-bin/reader/conf. cgi? confid= 187, id. 419. 2013

Ευχαριστίες

Η παρούσα έρευνα υποστηρίχθηκε με υπολογιστικούς πόρους στο Norwegian Institute of Science and Technology (NTNU) που παρέχονται από το NOTUR, με τη χορηγεία υπολογιστικών πόρων από το Swiss National Supercomputing Center (CSCS) στο πλαίσιο του έργου g83 και με υπολογιστικό χρόνο που παρασχέθηκε από το Εθνικό Δίκτυο Έρευνας και Τεχνολογίας (ΕΔΕΤ) στην εθνική υπερυπολογιστική υποδομή (ARIS) στο πλαίσιο των έργων pa001006, pa006010 και pa170302. Η συγγραφέας έλαβε χρηματοδότηση από το Ίδρυμα Κρατικών Υποτροφιών στο πλαίσιο των υποτροφιών αριστείας ΙΚΥ Β' Κύκλου Σπουδών (Διδακτορικό) στην Ελλάδα - Πρόγραμμα Siemens.

Βιβλιογραφία

The structural simulation toolkit. http://sst-simulator.org.

Top500 supercomputer sites. URL http://www.top500.org.

Luay Alawneh, Abdelwahab Hamou-Lhadj, and Jameleddine Hassine. Segmenting large traces of inter-process communication with a focus on high performance computing systems. *Journal of Systems and Software*, 120:1–16, 2016.

Albert Alexandrov, Mihai F Ionescu, Klaus E Schauser, and Chris Scheiman. LogGP: incorporating long messages into the LogP model—one step closer towards a realistic model for parallel computation. In *Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, pages 95–105. ACM, 1995.

Bob Alverson, Edwin Froese, Larry Kaplan, and Duncan Roweth. Cray xc series network. *Cray Inc., White Paper WP-Aries01-1112*, 2012.

Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, et al. The landscape of parallel computing research: A view from berkeley. Technical report, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

InfiniBand Trade Association et al. Infiniband specification. *InfiniBand™ Architecutre Release*, 1, 2001.

Kevin J Barker, Kei Davis, Adolfy Hoisie, Darren J Kerbyson, Michael Lang, Scott Pakin, and JoséCarlos Sancho. Using performance modeling to design large-scale systems. *Computer*, 42(10):0042–49, 2009.

Greg Bauer, Steven Gottlieb, and Torsten Hoefler. Performance modeling and comparative analysis of the MILC lattice QCD application su3_rmd. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 652–659. IEEE, 2012.

Paul Bédaride, Augustin Degomme, Stéphane Genaud, Arnaud Legrand, George Markomanolis, Martin Quinson, Mark Lee Stillwell, Frédéric Suter, Brice Videau, et al. Toward better simulation of MPI applications on Ethernet/TCP networks. In *PMBS13-4th International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, 2013.

Costas Bekas and Alessandro Curioni. A new energy aware performance metric. *Computer Science-Research and Development*, 25(3-4):187–195, 2010.

Abhinav Bhatelé and Laxmikant V Kalé. Quantifying network contention on large parallel machines. *Parallel Processing Letters*, 19(04):553–572, 2009.

Abhinav Bhatele, Andrew R Titus, Jayaraman J Thiagarajan, Nikhil Jain, Todd Gamblin, Peer-Timo Bremer, Martin Schulz, and Laxmikant V Kale. Identifying the culprits behind network congestion. In *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, pages 113–122. IEEE, 2015.

Mark S Birrittella, Mark Debbage, Ram Huggahalli, James Kunz, Tom Lovett, Todd Rimmer, Keith D Underwood, and Robert C Zak. Intel® omnipath architecture: Enabling scalable, high performance fabrics. In *High-Performance Interconnects (HOTI), 2015 IEEE 23rd Annual Symposium on*, pages 1–9. IEEE, 2015.

David M Brooks, Pradip Bose, Stanley E Schuster, Hans Jacobson, Prabhakar N Kudva, Alper Buyuktosunoglu, J-D Wellman, Victor Zyuban, Manish Gupta, and Peter W Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *Micro, IEEE*, 20(6):26–44, 2000.

Henri Casanova, Frédéric Desprez, George S Markomanolis, and Frédéric Suter. Simulation of mpi applications with time-independent traces. *Concurrency and Computation: Practice and Experience*, 27(5):1145–1168, 2015.

Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

Barbara Chapman, Tony Curtis, Swaroop Pophale, Stephen Poole, Jeff Kuehn, Chuck Koelbel, and Lauren Smith. Introducing openshmem: Shmem for the pgas community. In *Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model*, page 2. ACM, 2010.

WenGuang Chen, JiDong Zhai, Jin Zhang, and WeiMin Zheng. LogGPO: An accurate communication model for performance prediction of MPI programs. *Science in China Series F: Information Sciences*, 52(10):1785–1791, 2009.

UPC Consortium et al. Upc language specifications v1. 2. *Lawrence Berkeley National Laboratory*, 2005.

Samuel Daniel Conte, Hubert E Dunsmore, and Vincent Y Shen. *Software engineering metrics and models*. Benjamin-Cummings Publishing Co., Inc., 1986.

David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten Von Eicken. *LogP: Towards a realistic model of parallel computation*, volume 28. ACM, 1993.

Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

Frédéric Desprez, George S Markomanolis, Martin Quinson, and Frédéric Suter. Assessing the performance of mpi applications through time-independent trace replay. In *2011 40th International Conference on Parallel Processing Workshops*, pages 467–476. IEEE, 2011.

Salvatore Di Girolamo, Pierre Jolivet, Keith D Underwood, and Torsten Hoefler. Exploiting offload enabled network interfaces. In *High-Performance Interconnects (HOTI), 2015 IEEE 23rd Annual Symposium on*, pages 26–33. IEEE, 2015.

Jack Dongarra, Pete Beckman, Terry Moore, Patrick Aerts, Giovanni Aloisio, Jean-Claude Andre, David Barkai, Jean-Yves Berthou, Taisuke Boku, Bertrand Braunschweig, et al. The international exascale software project roadmap. *The international journal of high performance computing applications*, 25(1):3–60, 2011.

Jack Dongarra, Michael A Heroux, and Piotr Luszczek. Hpcg benchmark: a new metric for ranking high performance computing systems. *Knoxville, Tennessee*, 2015.

Nikolaos Drosinos and Nectarios Koziris. Performance comparison of pure mpi vs hybrid mpi-openmp parallelization models on smp clusters. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 15. IEEE, 2004.

Kurt B Ferreira, Patrick G Bridges, Ron Brightwell, and Kevin T Pedretti. The impact of system design parameters on application noise sensitivity. *Cluster computing*, 16(1):117–129, 2013.

Rosa Filgueira, David E Singh, Jesús Carretero, Alejandro Calderón, and Félix García. Adaptive-CoMPI: Enhancing MPI-based applications' performance and scalability by using adaptive compression. *International Journal of High Performance Computing Applications*, 25(1):93–114, 2011.

Matthew I Frank, Anant Agarwal, and Mary K Vernon. *LoPC: modeling contention in parallel algorithms*, volume 32. ACM, 1997.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. The elements of statistical learning: Data mining, inference, and prediction. *Springer Series in Statistics*, 2009.

Hormozd Gahvari, Allison H Baker, Martin Schulz, Ulrike Meier Yang, Kirk E Jordan, and William Gropp. Modeling the performance of an algebraic multigrid cycle on hpc platforms. In *Proceedings of the international conference on Supercomputing*, pages 172–181. ACM, 2011.

Hormozd Gahvari, William Gropp, Kirk E Jordan, Martin Schulz, and Ulrike Meier Yang. Algebraic multigrid on a dragonfly network: First experiences on a cray xc30. In *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*, pages 3–23. Springer, 2014.

Georgios Goumas, Nikolaos Drosinos, and Nectarios Koziris. Communication-aware supernode shape. *Parallel and Distributed Systems, IEEE Transactions on*, 20(4):498–511, 2009.

Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.

Georg Hager, Gabriele Jost, and Rolf Rabenseifner. Communication characteristics and hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes. In *Proceedings of Cray User Group Conference*, volume 4, page 5455, 2009.

Jerry L Hintze and Ray D Nelson. Violin plots: a box plot-density trace synergism. *The American Statistician*, 52(2):181–184, 1998.

Roger W Hockney. The communication challenge for MPP: Intel Paragon and Meiko CS-2. *Parallel computing*, 20(3):389–398, 1994.

Torsten Hoefler and Marc Snir. Generic topology mapping strategies for large-scale parallel architectures. In *Proceedings of the international conference on Supercomputing*, pages 75–84. ACM, 2011.

Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. Accurately measuring overhead, communication time and progression of blocking and non-blocking collective operations at massive scale. *International Journal of Parallel, Emergent and Distributed Systems*, 25(4):241–258, 2010a.

Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine. Characterizing the influence of system noise on large-scale applications by simulation. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE Computer Society, 2010b.

Torsten Hoefler, William Gropp, William Kramer, and Marc Snir. Performance modeling for systematic performance tuning. In *State of the Practice Reports*, page 6. ACM, 2011.

Sascha Hunold and Alexandra Carpen-Amarie. On the impact of synchronizing clocks and processes on benchmarking MPI collectives. In *EuroMPI*, pages 8:1–8:10. ACM, 2015.

Sascha Hunold and Alexandra Carpen-Amarie. Reproducible mpi benchmarking is still not as easy as you think. *IEEE Transactions on Parallel and Distributed Systems*, 27(12):3617–3630, 2016.

Huda Ibeid, Rio Yokota, and David Keyes. A performance model for the communication in fast multipole methods on high-performance computing platforms. *International Journal of High Performance Computing Applications*, page 1094342016634819, 2016.

Fumihiko Ino, Noriyuki Fujimoto, and Kenichi Hagihara. LogGPS: a parallel computational model for synchronization analysis. In *ACM SIGPLAN Notices*, volume 36, pages 133–142. ACM, 2001.

KE Isaacs, T Gamblin, A Bhatele, M Schulz, B Hamann, and PT Bremer. Ordering traces logically to identify lateness in parallel programs. Technical report, Technical Report LLNL-TR-656141, Lawrence Livermore National Laboratory, 2014.

Nikhil Jain, Abhinav Bhatele, Michael P Robson, Todd Gamblin, and Laxmikant V Kale. Predicting application performance using supervised learning on communication features. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, page 95. ACM, 2013.

Nikhil Jain, Abhinav Bhatele, Sam White, Todd Gamblin, and Laxmikant V Kale. Evaluating hpc networks via simulation of parallel workloads. In *High Performance Computing, Networking, Storage and Analysis, SC16: International Conference for*, pages 154–165. IEEE, 2016.

Nikhil Jain, Abhinav Bhatele, Xiang Ni, Todd Gamblin, and Laxmikant V Kale. Partitioning low-diameter networks to eliminate inter-job interference. 2017.

Karl Jansen and Carsten Urbach. tmlqcd: a program suite to simulate wilson twisted mass lattice qcd. *Computer Physics Communications*, 180(12):2717–2738, 2009.

Laxmikant V Kale and Sanjeev Krishnan. Charm++: a portable concurrent object oriented system based on c++. In *ACM Sigplan Notices*, volume 28, pages 91–108. ACM, 1993.

Ian Karlin, Abhinav Bhatele, Bradford L.. Chamberlain, Jonathan. Cohen, Zachary Devito, Maya Gokhale, Riyaz Haque, Rich Hornung, Jeff Keasler, Dan Laney, Edward Luke, Scott Lloyd, Jim McGraw, Rob Neely, David Richards, Martin Schulz, Charle H. Still, Felix Wang, and Daniel Wong. Lulesh programming model and performance ports overview. Technical Report LLNL-TR-608824, December 2012.

Peter Kogge and John Shalf. Exascale computing trends: Adjusting to the" new normal" for computer architecture. *Computing in Science & Engineering*, 15 (6):16–26, 2013.

Peter Kogge, Keren Bergman, Shekhar Borkar, Dan Campbell, W Carson, William Dally, Monty Denneau, Paul Franzon, William Harrod, Kerry Hill, et al. Exascale computing study: Technology challenges in achieving exascale systems. 2008.

Dong Li, Edgar A Leon, and Bronis R de Supinski. Adaptive parallelism: Integrated performance, power, and resilience modeling. In *DOE Workshop on Modeling & Simulation of Systems and Applications (MODSIM'14)*, 2014.

Sean Luke. *Essentials of metaheuristics*, volume 113. Lulu Raleigh, 2009.

Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 135–146. ACM, 2010.

Dan McMorrow. Technical challenges of exascale computing. *MITRE Corporation, McLean, VI, USA*, 2013.

João Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *ACM Computing Surveys (CSUR)*, 45(1):10, 2012.

Robert M Metcalfe and David R Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, 1976.

Csaba Andras Moritz and Matthew I Frank. LoGPC: Modeling network contention in message-passing programs. *ACM SIGMETRICS Performance Evaluation Review*, 26(1):254–263, 1998.

Csaba Andras Moritz and Matthew I Frank. LoGPG: Modeling network contention in message-passing programs. *Parallel and Distributed Systems, IEEE Transactions on*, 12(4):404–415, 2001.

Gihan R Mudalige, Mary K Vernon, and Stephen A Jarvis. A plug-and-play model for evaluating wavefront computations on parallel architectures. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–14. IEEE, 2008.

José Carlos Sancho, Kevin J Barker, Darren J Kerbyson, and Kei Davis. Quantifying the potential benefit of overlapping communication and computation in large-scale scientific applications. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, pages 17–17. IEEE, 2006.

Timo Schneider, Torsten Hoefler, Ryan E Grant, Brian W Barrett, and Ron Brightwell. Protocols for fully offloaded collective operations on accelerated network adapters. In *Parallel Processing (ICPP), 2013 42nd International Conference on*, pages 593–602. IEEE, 2013.

John Shalf, Sudip Dosanjh, and John Morrison. Exascale computing technology challenges. In *International Conference on High Performance Computing for Computational Science*, pages 1–25. Springer, 2010.

Sameer S Shende and Allen D Malony. The tau parallel performance system. *International Journal of High Performance Computing Applications*, 20(2):287–311, 2006.

Sergei Shudler, Alexandru Calotoiu, Torsten Hoefler, Alexandre Strube, and Felix Wolf. Exascaling your library: Will your implementation meet your expectations? In *Proceedings of the 29th ACM on International Conference on Supercomputing*, pages 165–175. ACM, 2015.

Marc Snir. *MPI–the Complete Reference: the MPI core*, volume 1. MIT press, 1998.

Edgar Solomonik and James Demmel. Communication-optimal parallel 2.5 D matrix multiplication and LU factorization algorithms. In *Euro-Par 2011 Parallel Processing*, pages 90–109. Springer, 2011.

Garrick Staples. Torque resource manager. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, page 8. ACM, 2006.

Leslie G Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

Jeffrey Vetter and Chris Chambreau. mpip: Lightweight, scalable mpi profiling. 2005.

Felix Wolf, Allen Malony, Sameer Shende, and Alan Morris. Trace-based parallel performance overhead compensation. *High Performance Computing and Communications*, pages 617–628, 2005.

Andy Yoo, Morris Jette, and Mark Grondona. Slurm: Simple linux utility for resource management. In *Job scheduling strategies for parallel processing*, pages 44–60. Springer, 2003.

Li Yu, Dong Li, Sparsh Mittal, and Jeffrey S Vetter. Quantitatively modeling application resilience with the data vulnerability factor. In *High Performance*

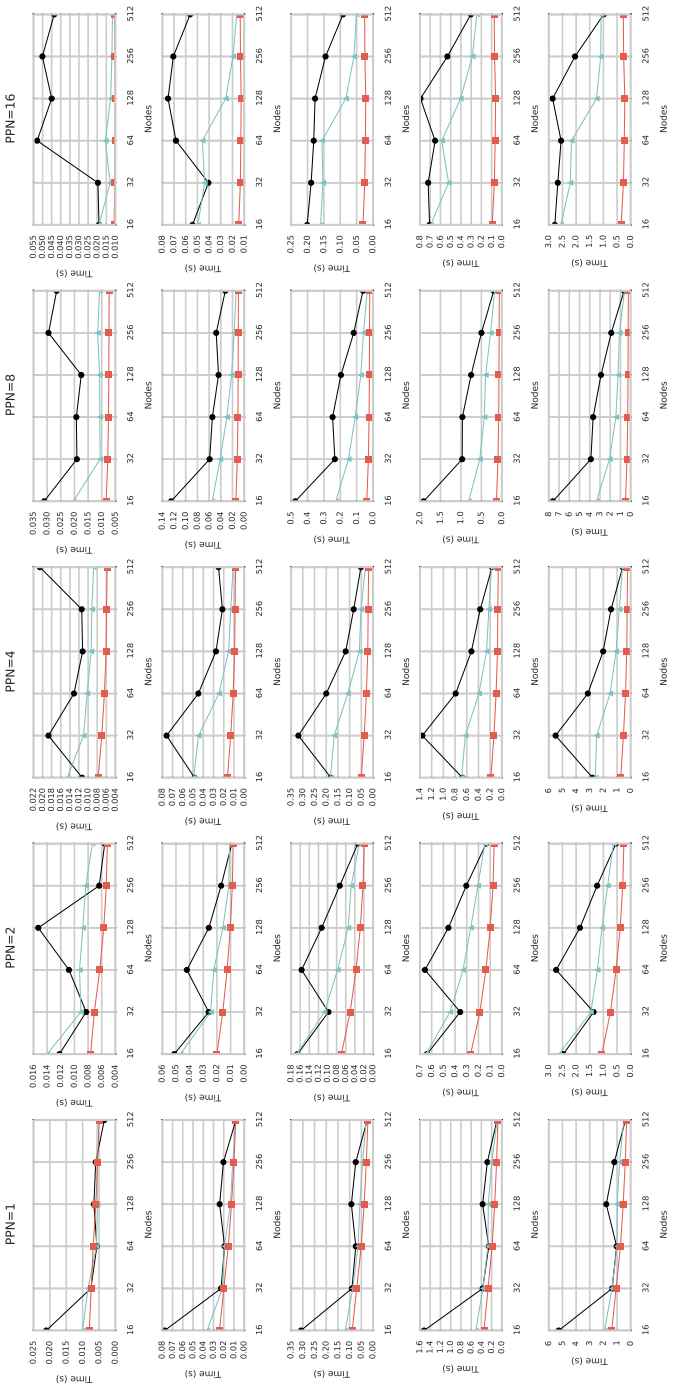
Computing, Networking, Storage and Analysis, SC14: International Conference for, pages 695–706. IEEE, 2014.

Cha Zhang and Yunqian Ma. *Ensemble machine learning*. Springer, 2012.

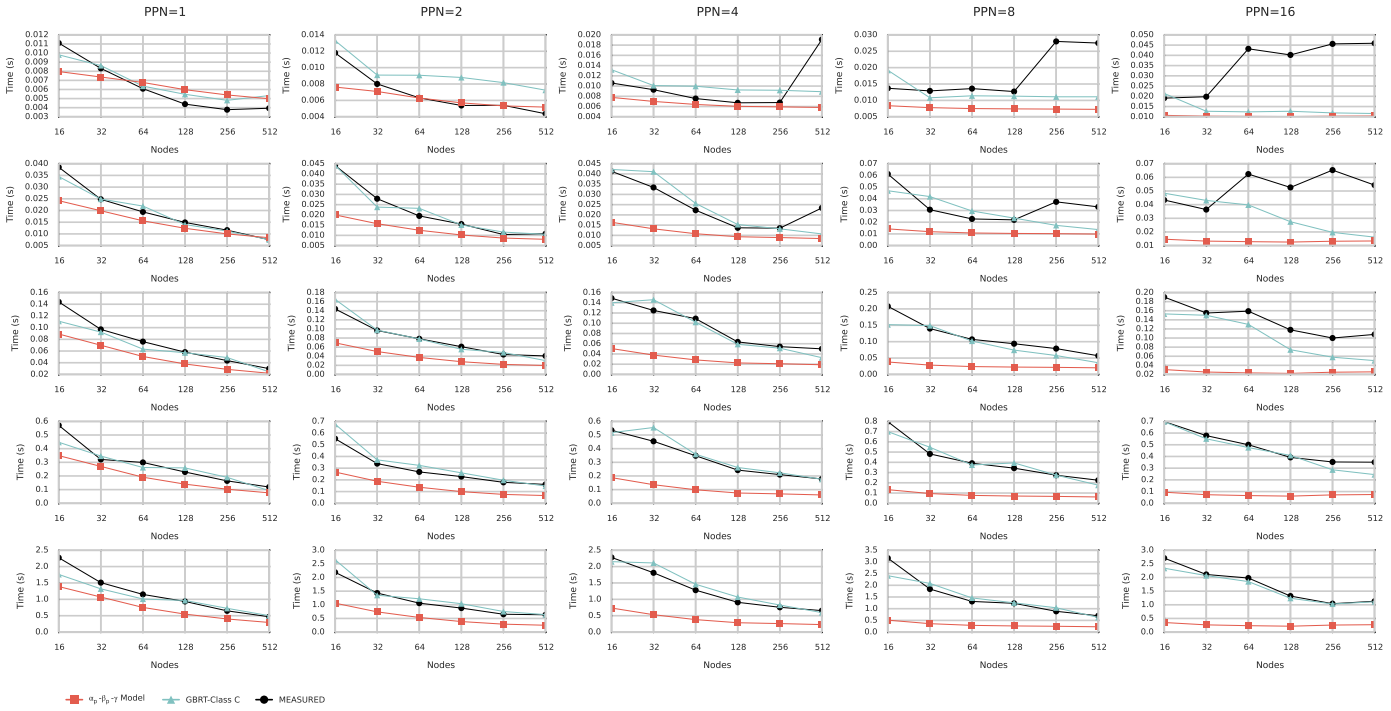
Jun Zhu, Alexey Lastovetsky, Shoukat Ali, Rolf Riesen, and Khalid Hasanov. Asymmetric communication models for resource-constrained hierarchical ethernet networks. *Concurrency and Computation: Practice and Experience*, 27(6):1575–1590, 2015.

Παράρτημα Α

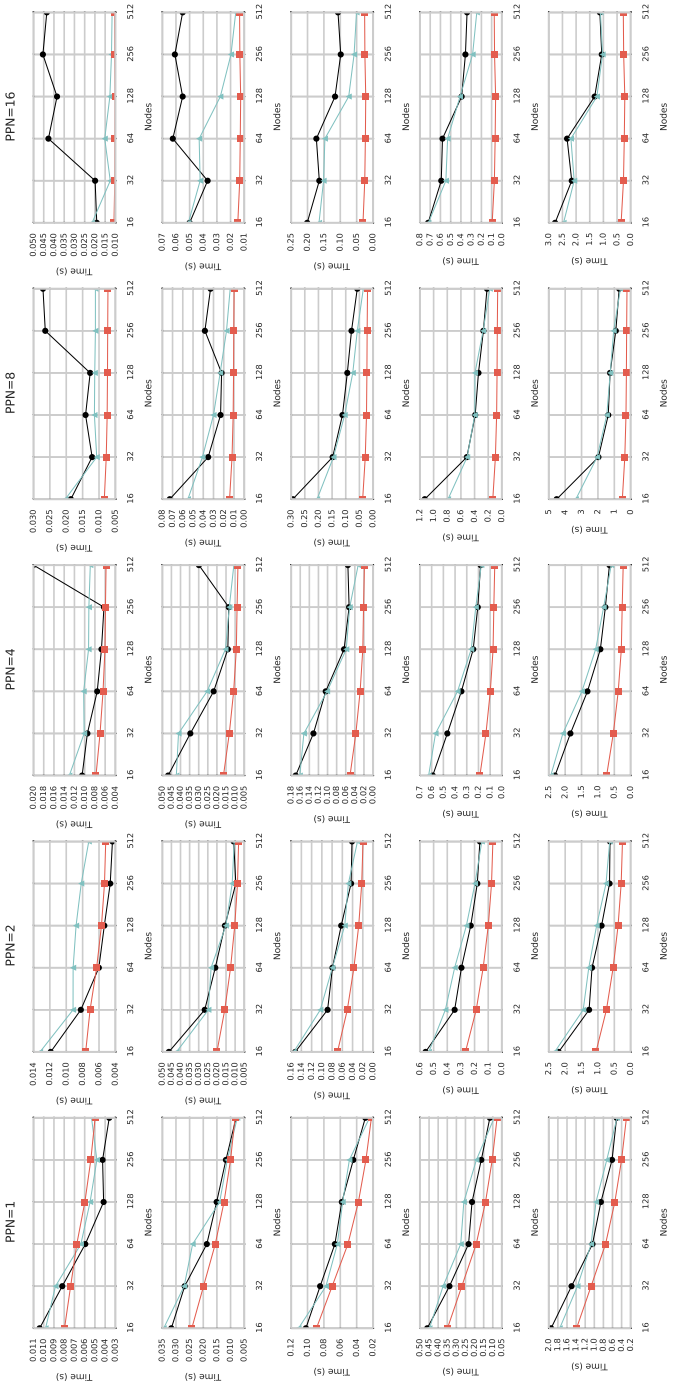
Στο παρόν παράρτημα, παρουσιάζουμε τα διαγράμματα για τις προβλέψεις κλιμάκωσης με το μοντέλο *GBRT-Class C* στον υπερυπολογιστή Vilje, ως συμπλήρωμα του Κεφαλαίου 4, όπου παρουσιάζουμε μόνο το βέλτιστο, διάμεσο και χειρότερο υποσύνολο προβλέψεων. Επιπλέον παρουσιάζουμε τα διαγράμματα για τις κατά Παρέτο βέλτιστες ρυθμίσεις εκτέλεσης στο σύστημα Vilje, ως συμπλήρωμα του Κεφαλαίου 4, όπου παρουσιάζουμε ενδεικτικά δύο περιπτώσεις.



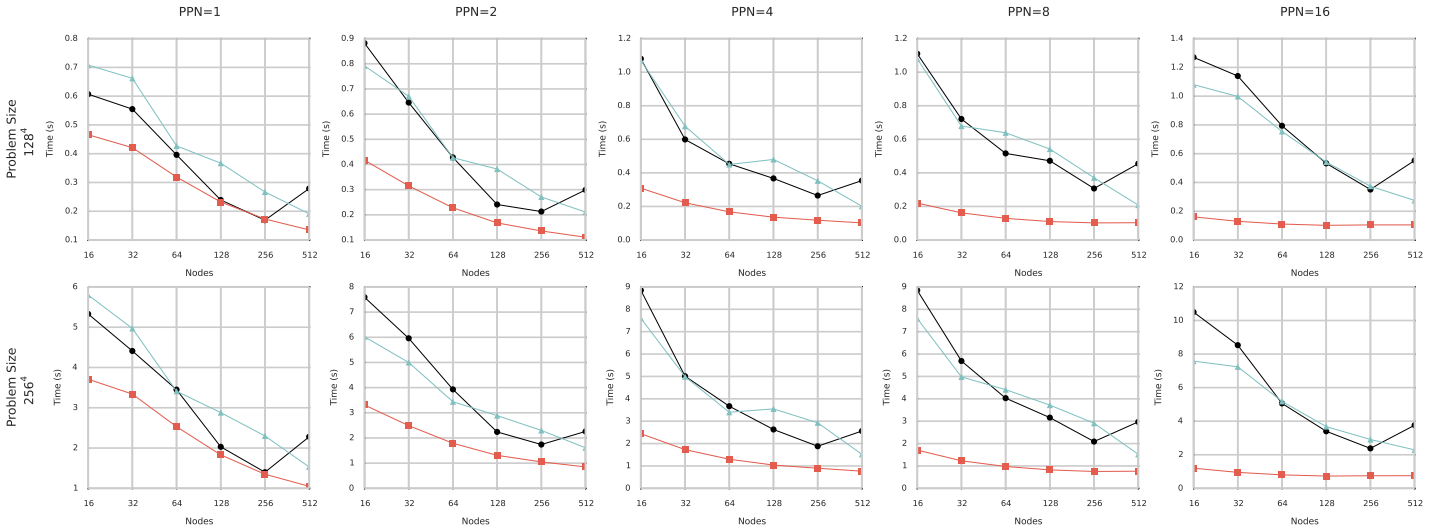
Σχήμα 1: Προβλέψεις για το σχήμα επικοινωνίας *Halo-3D* (εκτέλεση #1) για όλα τα μεγέθη προβλημάτων στο Vilje με το μοντέλο *GBRT-Class C* και το μοντέλο $\alpha_p - \beta_p - \gamma$



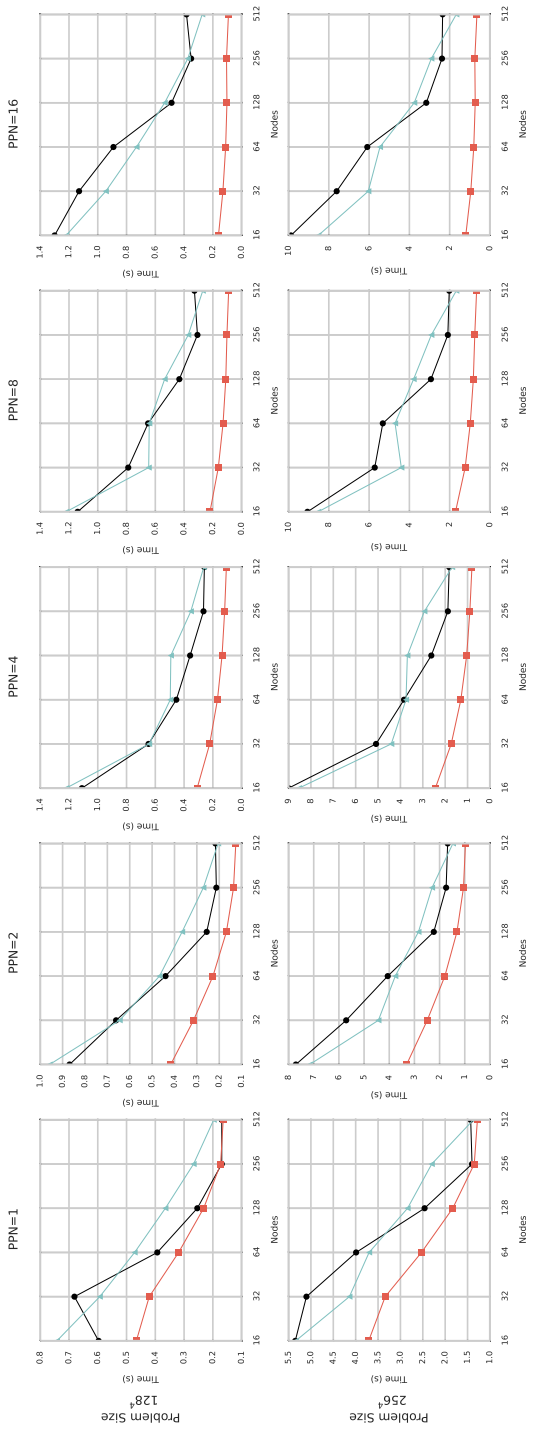
Σχήμα 2: Προβλέψεις για το σχήμα επικοινωνίας *Halo-3D* (εκτέλεση #2) για όλα τα μεγέθη προβλημάτων στο Vilje με το μοντέλο *GBRT-Class C* και το μοντέλο $\alpha_p - \beta_p - \gamma$



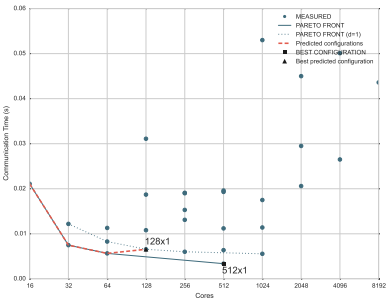
Σχήμα 3: Προβλέψεις για το σχήμα επικοινωνίας *Halo-3D* (εκτέλεση #3) για όλα τα μεγέθη προβλημάτων στο Vilje με το μοντέλο *GBRT-Class C* και το μοντέλο $\alpha_p - \beta_p - \gamma$



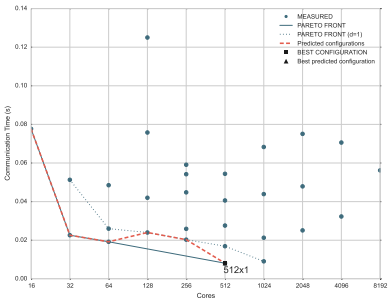
Σχήμα 4: Προβλέψεις για το σχήμα επικοινωνίας *Halo-4D* (εκτέλεση #1) για όλα τα μεγέθη προβλημάτων στο Vilje με το μοντέλο *GBRT-Class C* και το μοντέλο $\alpha_p - \beta_p - \gamma$



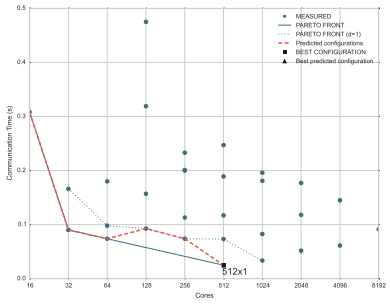
Σχήμα 5: Προβλέψεις για το σχήμα επικοινωνίας *Halo-4D* (εκτέλεση #2) για όλα τα μεγέθη προβλημάτων στο Vilje με το μοντέλο *GBRT-Class C* και το μοντέλο $\alpha_p - \beta_p - \gamma$



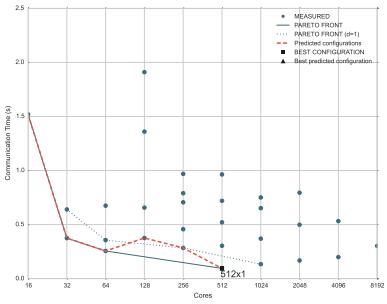
(i) Μέγεθος προβλήματος: 128^3



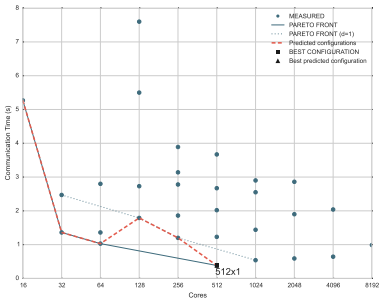
(ii) Μέγεθος προβλήματος: 256^3



(iii) Μέγεθος προβλήματος: 512^3

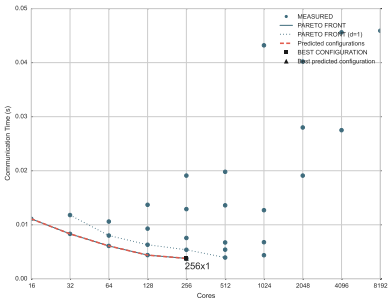


(iv) Μέγεθος προβλήματος: 1024^3

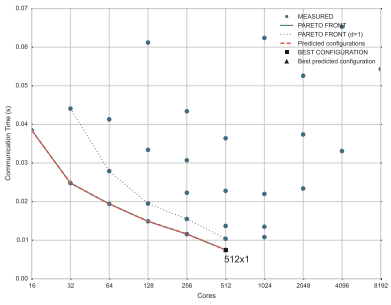


(v) Μέγεθος προβλήματος: 2048^3

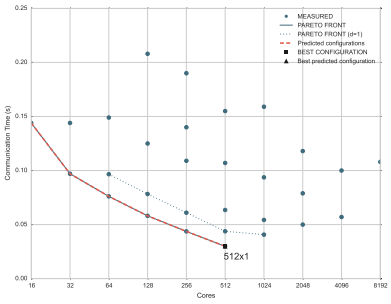
Σχήμα 6: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *Halo-3D* (εκτέλεση #1) στο Vilje



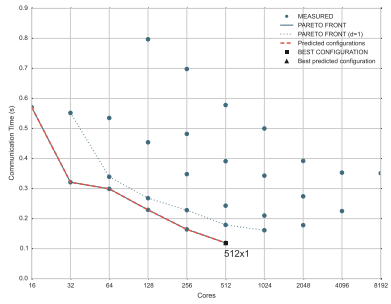
(i) Μέγεθος προβλήματος: 128^3



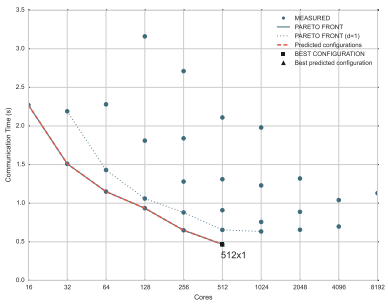
(ii) Μέγεθος προβλήματος: 256^3



(iii) Μέγεθος προβλήματος: 512^3

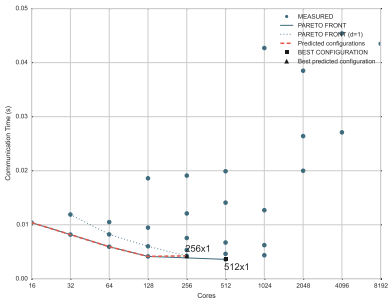


(iv) Μέγεθος προβλήματος: 1024^3

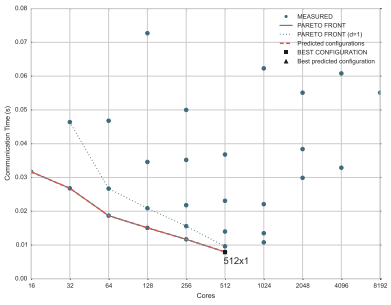


(v) Μέγεθος προβλήματος: 2048^3

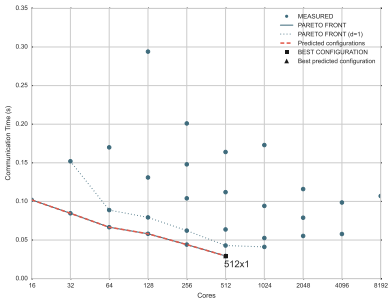
Σχήμα 7: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *Halo-3D* (εκτέλεση #2) στο Vilje



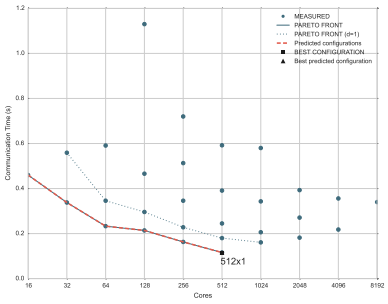
(i) Μέγεθος προβλήματος: 128^3



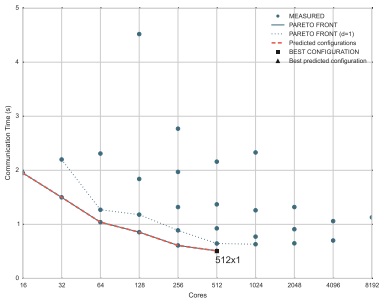
(ii) Μέγεθος προβλήματος: 256^3



(iii) Μέγεθος προβλήματος: 512^3

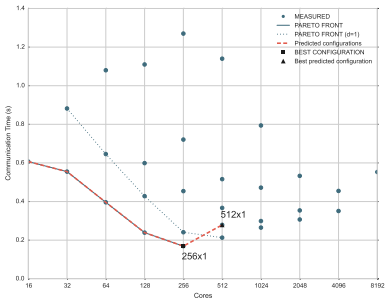


(iv) Μέγεθος προβλήματος: 1024^3

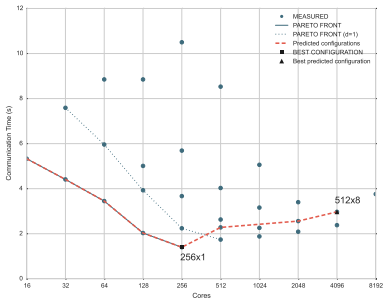


(v) Μέγεθος προβλήματος: 2048^3

Σχήμα 8: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *Halo-3D* (εκτέλεση #3) στο Vilje

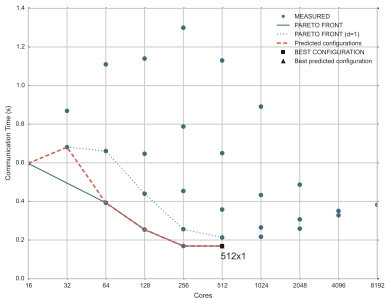


(i) Μέγεθος προβλήματος: 128^4

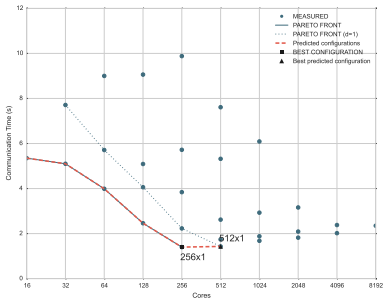


(ii) Μέγεθος προβλήματος: 256^4

Σχήμα 9: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *Halo-4D* (εκτέλεση #1) στο Vilje

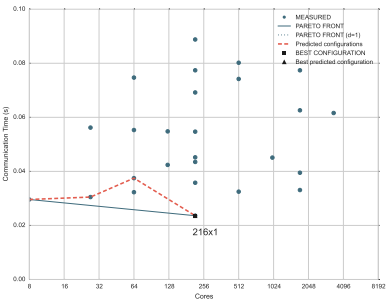


(i) Μέγεθος προβλήματος: 128^4

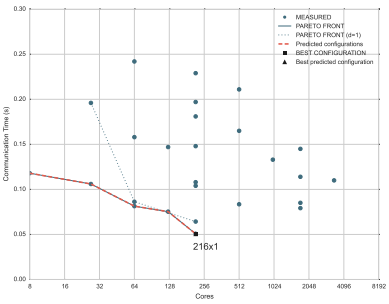


(ii) Μέγεθος προβλήματος: 256^4

Σχήμα 10: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *Halo-4D* (εκτέλεση #2) στο Vilje

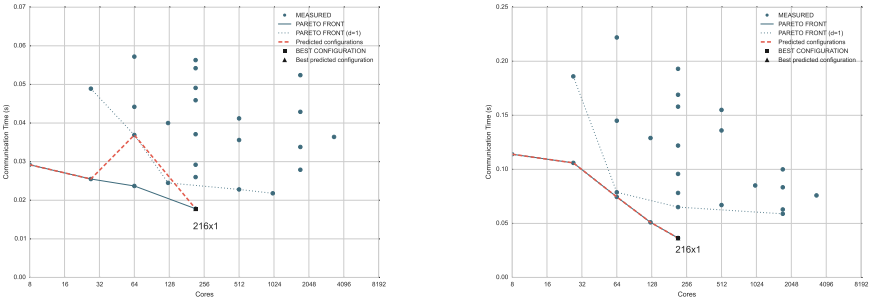


(i) Μέγεθος προβλήματος: 240^3



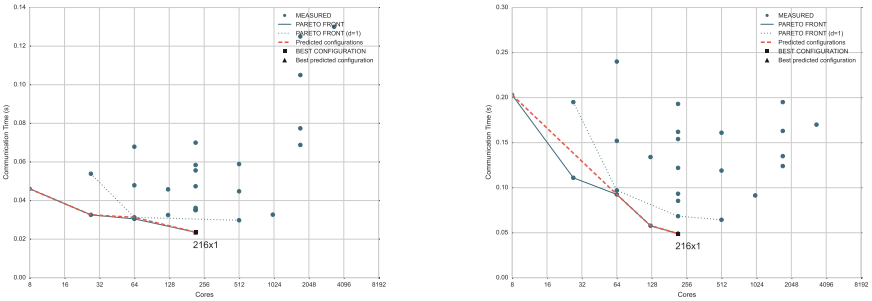
(ii) Μέγεθος προβλήματος: 480^3

Σχήμα 11: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *LULESH-1* στο Vilje



(i) Μέγεθος προβλήματος: 240^3 **(ii)** Μέγεθος προβλήματος: 480^3

Σχήμα 12: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *LULESH-2* στο Vilje

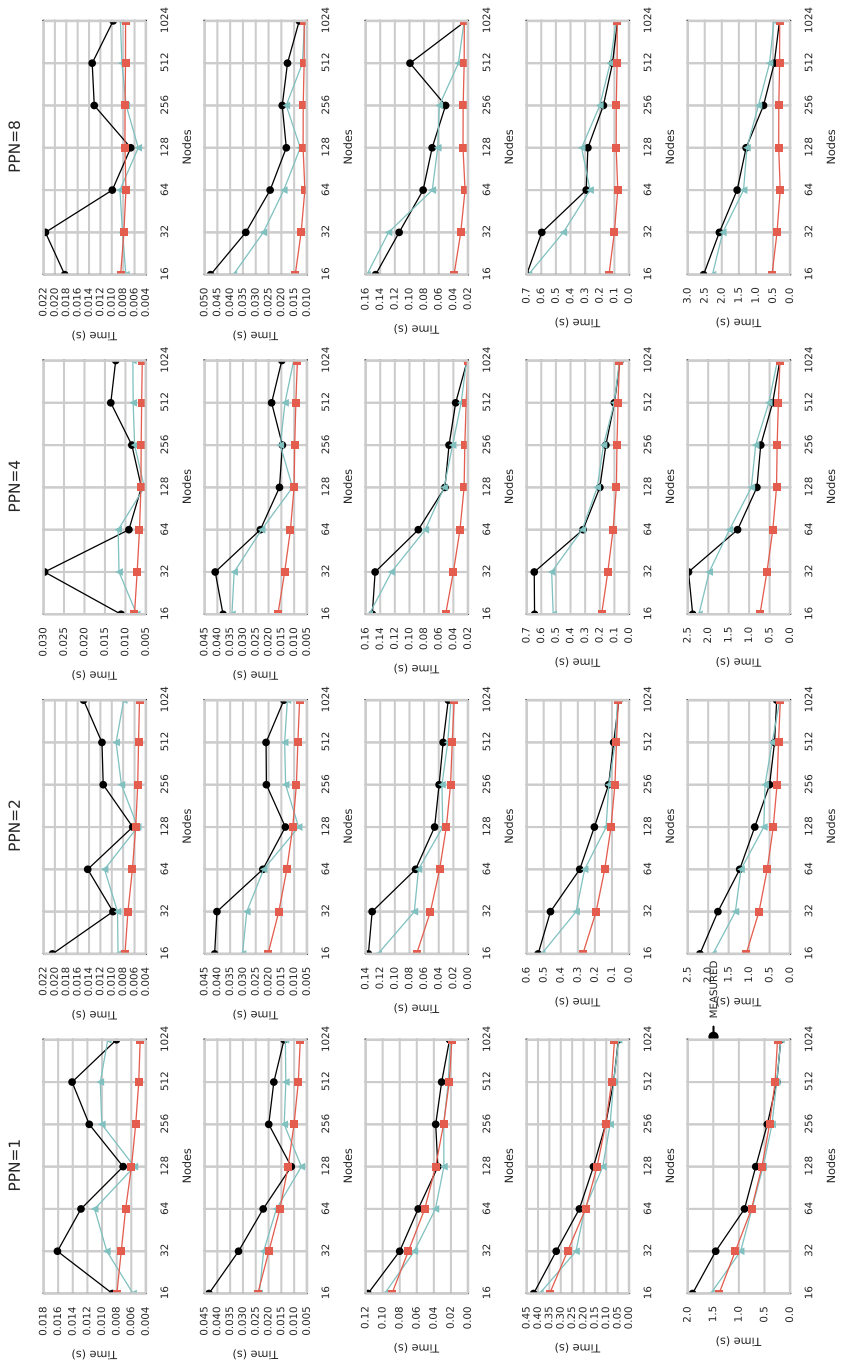


(i) Μέγεθος προβλήματος: 240^3 **(ii)** Μέγεθος προβλήματος: 480^3

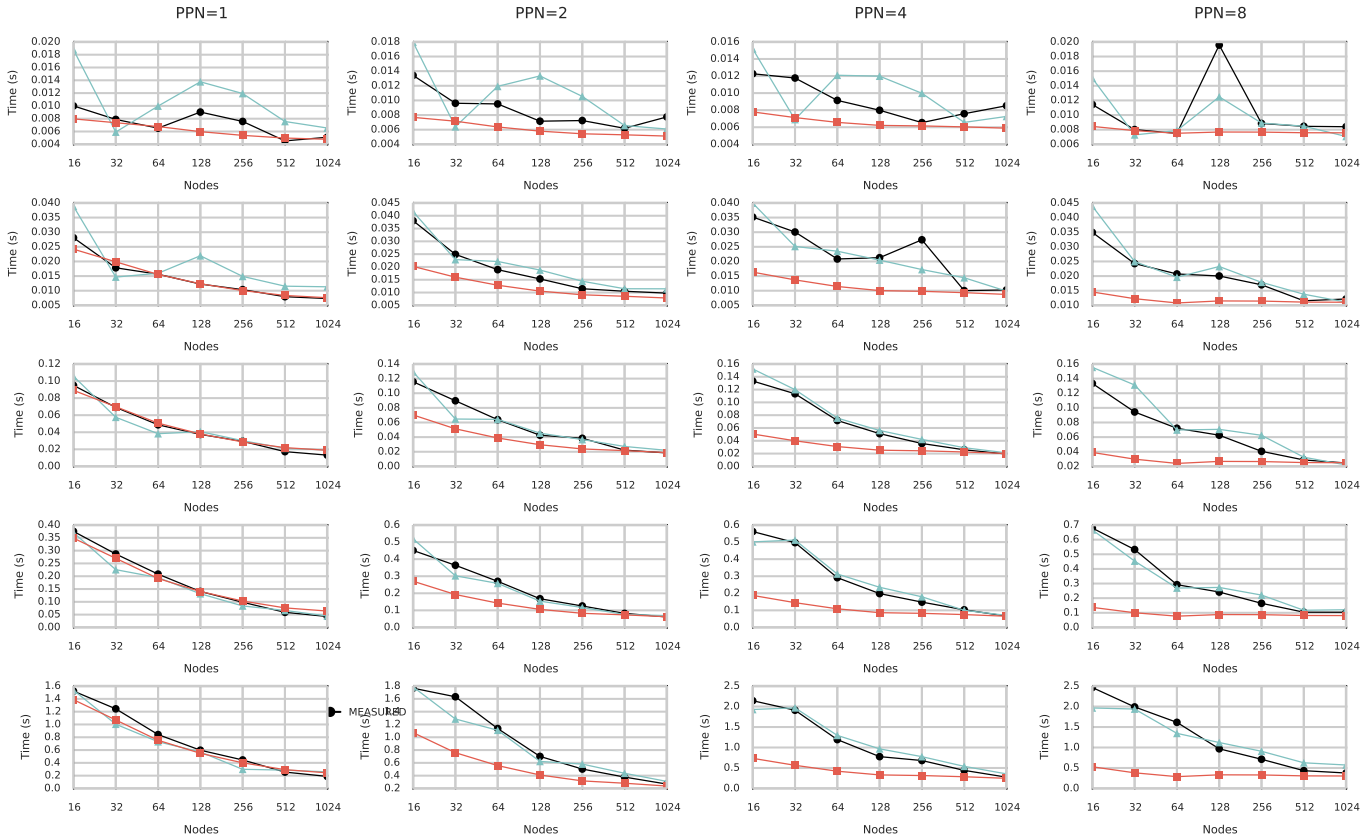
Σχήμα 13: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *LULESH-3* στο Vilje

Παράρτημα Β

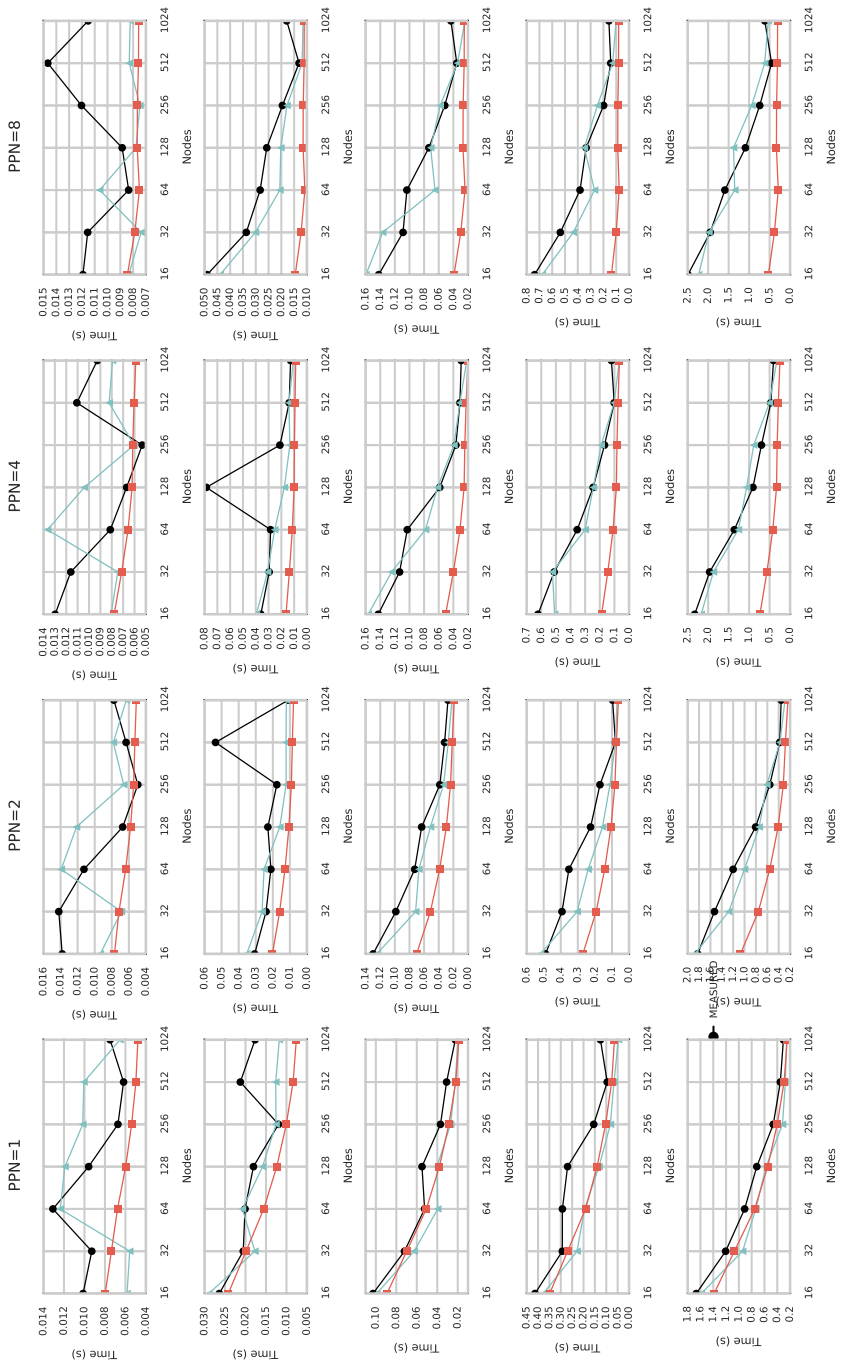
Στο παρόν παράρτημα, παρουσιάζουμε τα διαγράμματα για τις προβλέψεις κλιμάκωσης με το μοντέλο *GBRT-Class C* στον υπερυπολογιστή Piz Daint, ως συμπλήρωμα του Κεφαλαίου 5, όπου παρουσιάζουμε μόνο το βέλτιστο, διάμεσο και χειρότερο υποσύνολο προβλέψεων. Επιπλέον παρουσιάζουμε τα διαγράμματα για τις κατά Παρέτο βέλτιστες ρυθμίσεις εκτέλεσης στο σύστημα Piz Daint, ως συμπλήρωμα του Κεφαλαίου 5, όπου παρουσιάζουμε ενδεικτικά δύο περιπτώσεις.



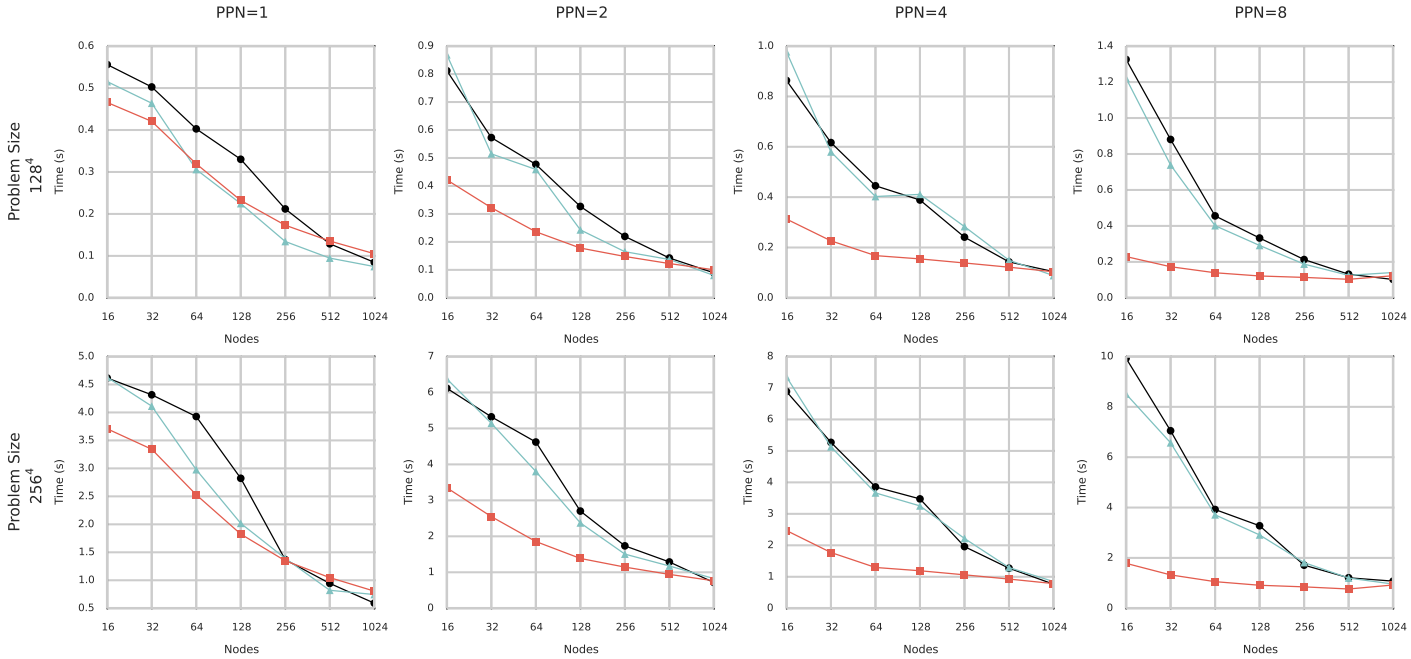
Σχήμα 14: Προβλέψεις για το σχήμα επικοινωνίας *Halo-3D* (execution #1) για όλα τα μεγέθη προβλημάτων στο Piz Daint με το μοντέλο *GBRT-Class C* και το μοντέλο $\alpha_p - \beta_p - \gamma$



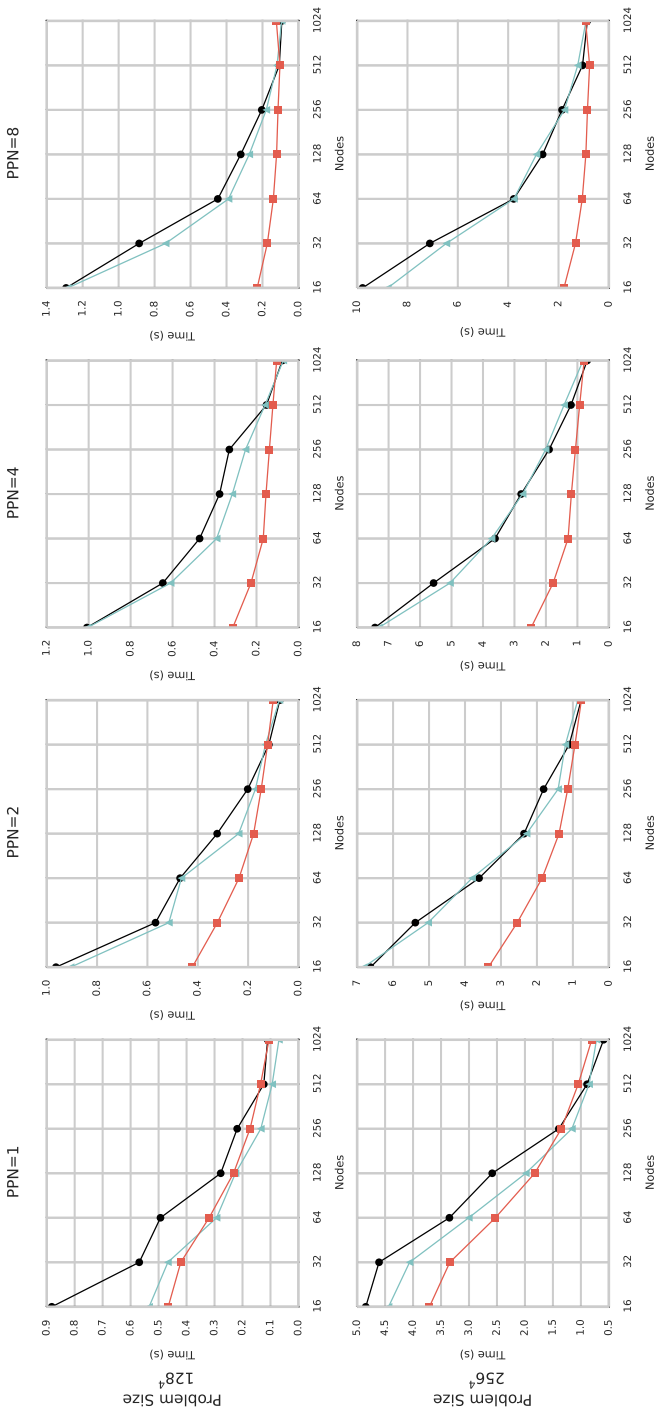
Σχήμα 15: Προβλέψεις για το σχήμα επικοινωνίας *Halo-3D* (execution #2) για όλα τα μεγέθη προβλημάτων στο Piz Daint με το μοντέλο *GBRT-Class C* και το μοντέλο $\alpha_p - \beta_p - \gamma$



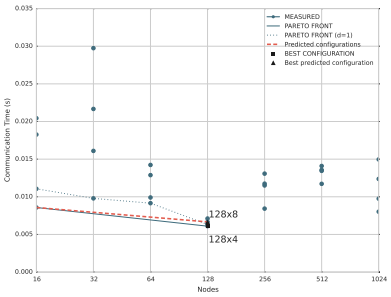
Σχήμα 16: Προβλέψεις για το σχήμα επικοινωνίας *Halo-3D* (execution #3) για όλα τα μεγέθη προβλημάτων στο Piz Daint με το μοντέλο *GBRT-Class C* και το μοντέλο $\alpha_p - \beta_p - \gamma$



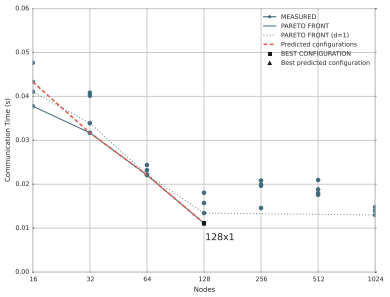
Σχήμα 17: Προβλέψεις για το σχήμα επικοινωνίας *Halo-4D* (execution #1) για όλα τα μεγέθη προβλημάτων στο Piz Daint με το μοντέλο *GBRT-Class C* και το μοντέλο $\alpha_p - \beta_p - \gamma$



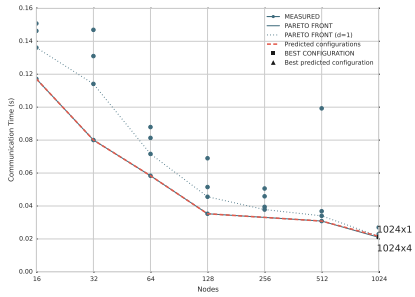
Σχήμα 18: Προβλέψεις για το σχήμα επικοινωνίας *Halo-4D* (execution #2) για όλα τα μεγέθη προβλημάτων στο Piz Daint με το μοντέλο *GBRT-Class C* και το μοντέλο $\alpha_p - \beta_p - \gamma$



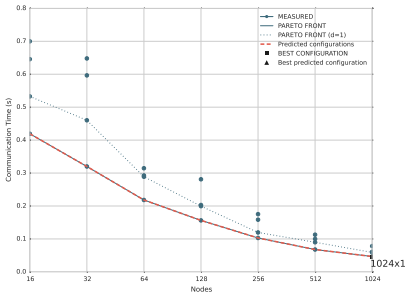
(i) Μέγεθος προβλήματος: 128^3



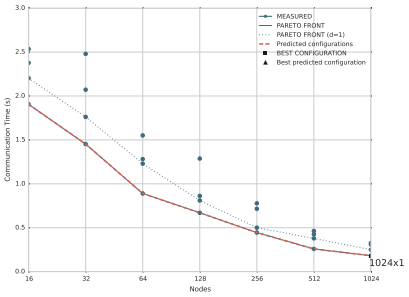
(ii) Μέγεθος προβλήματος: 256^3



(iii) Μέγεθος προβλήματος: 512^3

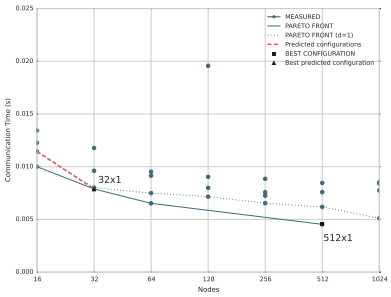


(iv) Μέγεθος προβλήματος: 1024^3

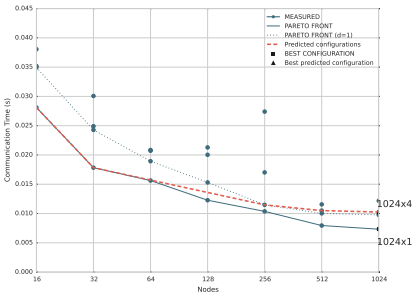


(v) Μέγεθος προβλήματος: 2048^3

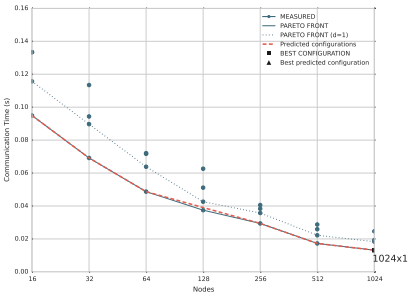
Σχήμα 19: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *Halo-3D* (execution #1) στο Piz Daint



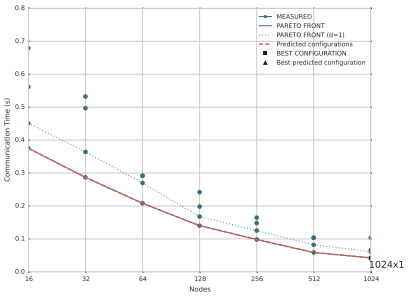
(i) Μέγεθος προβλήματος: 128^3



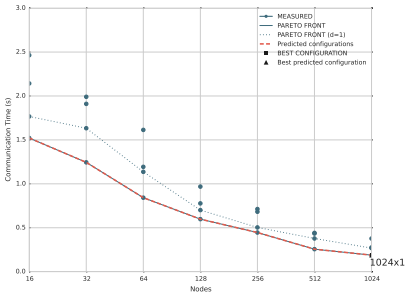
(ii) Μέγεθος προβλήματος: 256^3



(iii) Μέγεθος προβλήματος: 512^3

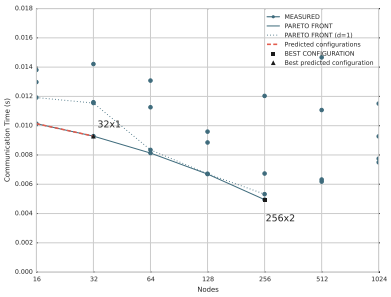


(iv) Μέγεθος προβλήματος: 1024^3

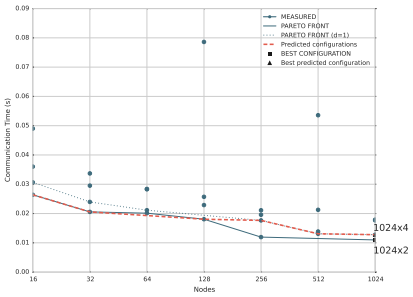


(v) Μέγεθος προβλήματος: 2048^3

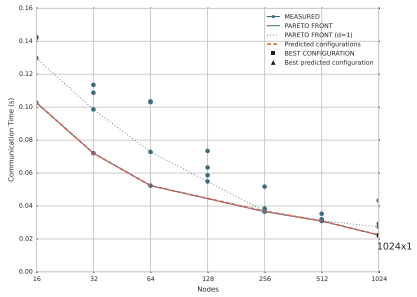
Σχήμα 20: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *Halo-3D* (execution #2) στο Piz Daint



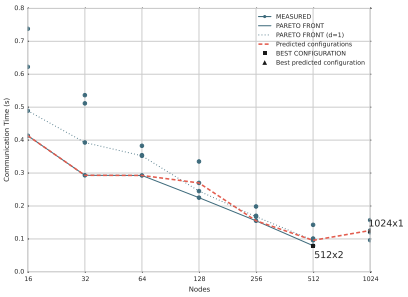
(i) Μέγεθος προβλήματος: 128^3



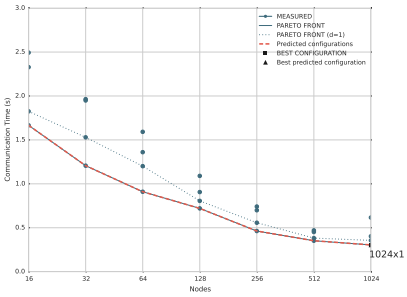
(ii) Μέγεθος προβλήματος: 256^3



(iii) Μέγεθος προβλήματος: 512^3

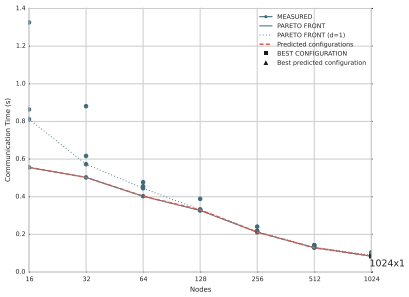


(iv) Μέγεθος προβλήματος: 1024^3

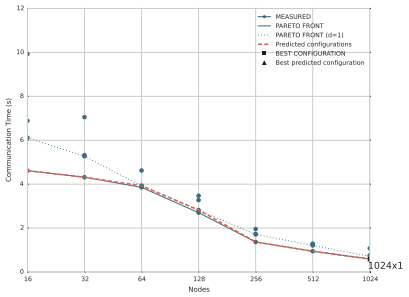


(v) Μέγεθος προβλήματος: 2048^3

Σχήμα 21: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *Halo-3D* (execution #3) στο Piz Daint

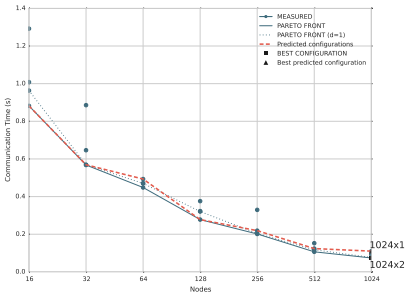


(i) Μέγεθος προβλήματος: 128^4

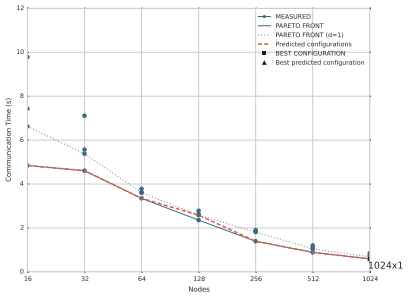


(ii) Μέγεθος προβλήματος: 256^4

Σχήμα 22: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *Halo-4D* (execution #1) στο Piz Daint

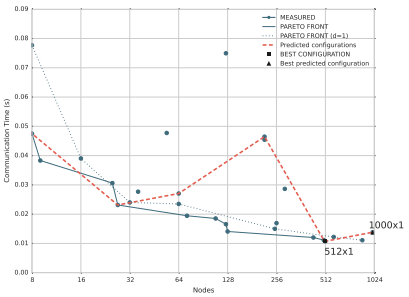


(i) Μέγεθος προβλήματος: 128^4

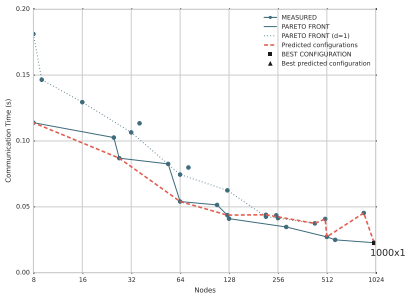


(ii) Μέγεθος προβλήματος: 256^4

Σχήμα 23: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *Halo-4D* (execution #2) στο Piz Daint

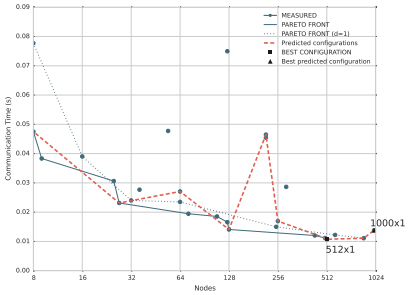


(i) Μέγεθος προβλήματος: 240^3

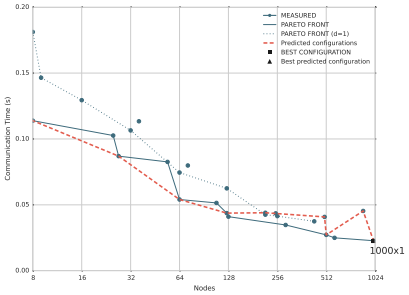


(ii) Μέγεθος προβλήματος: 480^3

Σχήμα 24: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *LULESH-1* στο Piz Daint

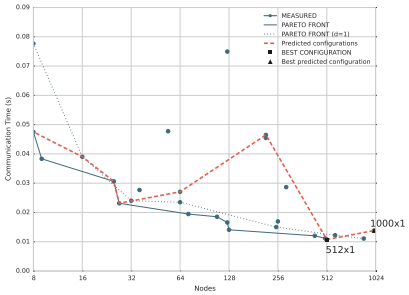


(i) Μέγεθος προβλήματος: 240^3

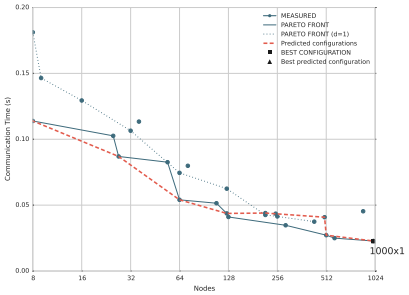


(ii) Μέγεθος προβλήματος: 480^3

Σχήμα 25: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *LULESH-2* στο Piz Daint



(i) Μέγεθος προβλήματος: 240^3

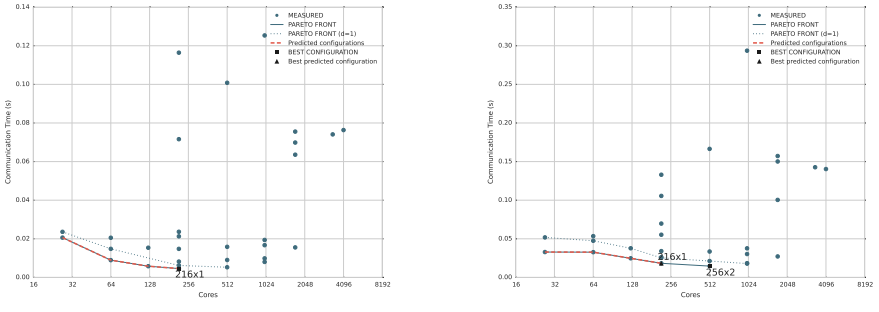


(ii) Μέγεθος προβλήματος: 480^3

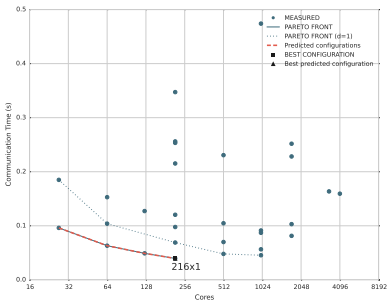
Σχήμα 26: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *LULESH-3* στο Piz Daint

Παράρτημα Γ

Στο παρόν παράρτημα, παρουσιάζουμε τα διαγράμματα για τις προβλέψεις των κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης με το μοντέλο *GBRT-Class C* στο σύστημα *ARIS*, ως συμπληρωματικό υλικό στο Κεφάλαιο 6, όπου ενδεικτικά παρουσιάζουμε δύο περιπτώσεις.

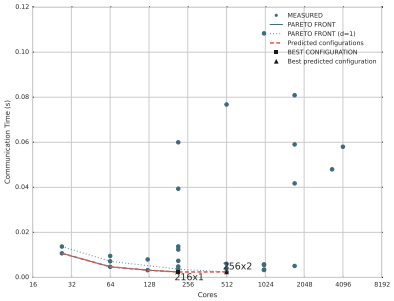


(i) Μέγεθος προβλήματος: 120^3 **(ii)** Μέγεθος προβλήματος: 240^3

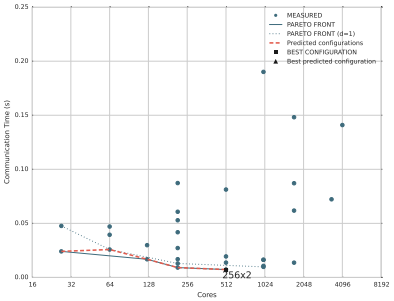


(iii) Μέγεθος προβλήματος: 480^3

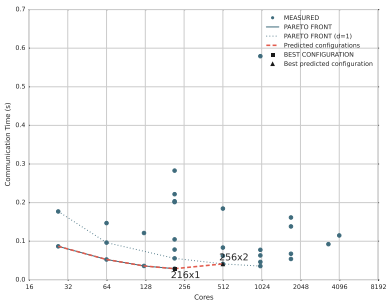
Σχήμα 27: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *LULESH-1* στο ARIS



(i) Μέγεθος προβλήματος: 120^3

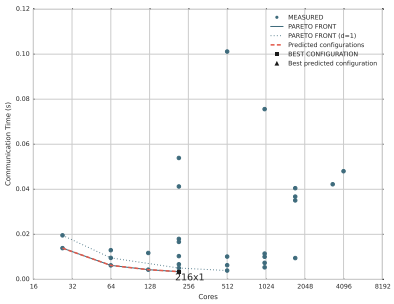


(ii) Μέγεθος προβλήματος: 240^3

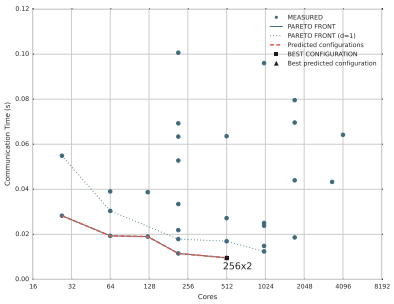


(iii) Μέγεθος προβλήματος: 480^3

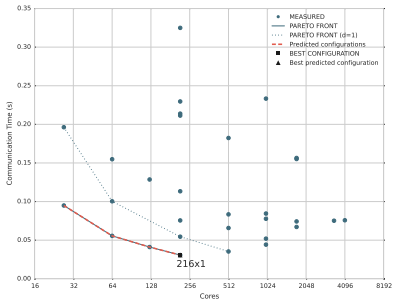
Σχήμα 28: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *LULESH-2* στο ARIS



(i) Μέγεθος προβλήματος: 120^3

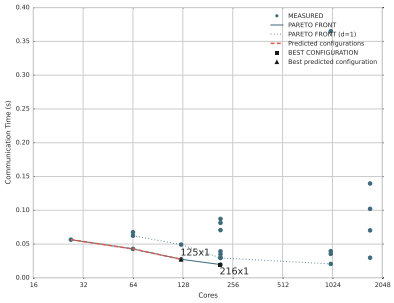


(ii) Μέγεθος προβλήματος: 240^3

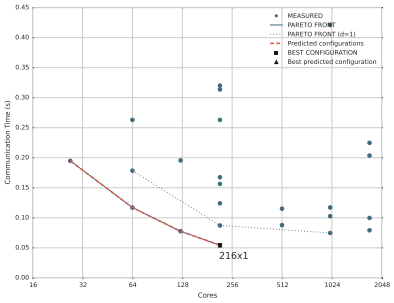


(iii) Μέγεθος προβλήματος: 480^3

Σχήμα 29: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *LULESH-3* στο ARIS

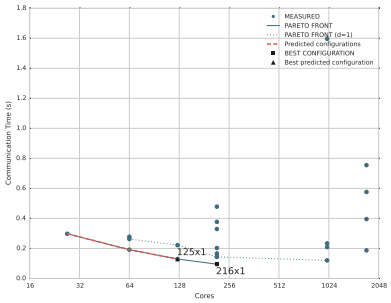


(i) Μέγεθος προβλήματος: 480^3

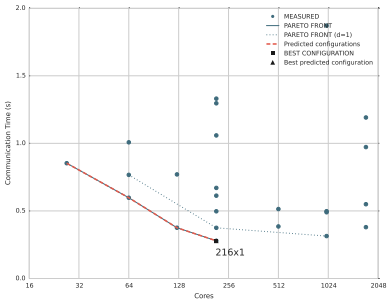


(ii) Μέγεθος προβλήματος: 960^3

Σχήμα 30: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *HPCG-SpMV* στο ARIS

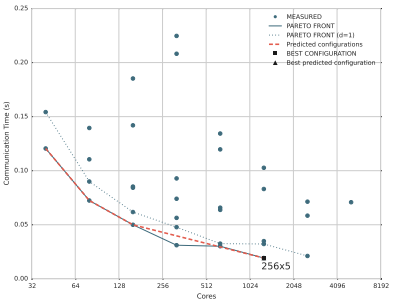


(i) Μέγεθος προβλήματος: 480^3

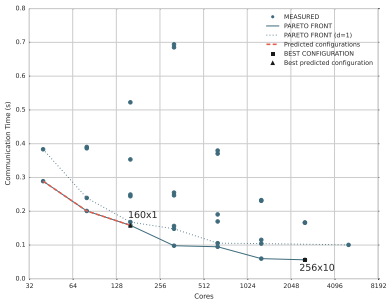


(ii) Μέγεθος προβλήματος: 960^3

Σχήμα 31: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *HPCG-MG* στο ARIS



(i) Μέγεθος προβλήματος: $32 \times 32 \times 32 \times 40$



(ii) Μέγεθος προβλήματος: $32 \times 32 \times 64 \times 80$

Σχήμα 32: Προβλέψεις κατά Παρέτο βέλτιστων ρυθμίσεων εκτέλεσης για το σχήμα επικοινωνίας *QCD-Kernel D* στο ARIS