Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Μελέτη της Επίδρασης της Επικοινωνίας με το Δίσκο στην Δρομολόγηση Εικονικών Μηχανών σε Περιβάλλον Υπολογιστικού Νέφους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΙΑΝΝΗΣ ΣΠΗΛΙΟΠΟΥΛΟΣ

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2014

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Μελέτη της Επίδρασης της Επικοινωνίας με το Δίσκο στην Δρομολόγηση Εικονικών Μηχανών σε Περιβάλλον Υπολογιστικού Νέφους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΓΙΑΝΝΗΣ ΣΠΗΛΙΟΠΟΥΛΟΣ

Επιβλέπων : Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Ιουλίου 2014.

Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Νικόλαος Παπασπύρου Αναπ. Καθηγητής Ε.Μ.Π.	Γεώργιος Γκούμας Λέκτορας Ε.Μ.Π.

Αθήνα, Ιούλιος 2014

Γιάννης Σπηλιόπουλος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Τα τελευταία χρόνια παρατηρείται μία ιδιαίτερα σημαντική αύξηση του πλήθους και του μεγέθους των νέων κέντρων δεδομένων. Πέρα όμως από τα αναμφισβήτητα πολλαπλά οφέλη που συνεπάγεται, η αύξηση των υπολογιστικών υποδομών έχει γίνει το επίκεντρο ανησυχιών λόγω της ολοένα αυξανόμενη συνεισφοράς της στην κατανάλωση ηλεκτρικής ενέργειας και στην παραγωγή ρύπων παγκοσμίως. Ως εκ τούτου, όλο και περισσότεροι ερευνητές αναζητούν τρόπους να αυξήσουν την αποδοτικότητα των υπολογιστικών υποδομών.

Μία από τις πιο δημοφιλής τεχνικές είναι η δυναμική ανακατανομή του διαθέσιμου φόρτου εργασίας με σκοπό την συγκέντρωση του σε όσο το δυνατόν λιγότερους διακομιστές και την απενεργοποίηση του υπόλοιπου εξοπλισμού του κέντρου δεδομένων. Σε αυτή την διπλωματική εργασία, παρουσιάζεται μια πολιτική δρομολόγησης εικονικών μηχανών που λαμβάνει υπόψη της τον φόρτο τόσο της κεντρικής μονάδας επεξεργασίας όσο και του συστήματος αποθήκευσης δεδομένων.

Λέξεις κλειδιά

Υπολογιστικό νέφος, εικονική μηχανή, σταθερά αποθηκευτικά μέσα, επικοινωνία, επίδραση, πολιτική δυναμικής τοποθέτησης, μετεγκατάσταση, προσομοίωση, CloudSim

Abstract

During the pas few years we have become witnesses of a significant increase in the number and size of data centers. Undoubtedly, this increase had multiple benefits for our society. Nevertheless, it has become the center of attention due to several side effects. The most important of these side effects have been the ever increasing contribution of data centers in global energy consumption and the concomitant contribution in pollutant production. As a result, many researchers are seeking ways to increase the efficiency of computing infrastructure.

One of the most popular techniques is the dynamic redistribution of work in order to concentrate the available load in the least amount of servers possible and deactivating the rest of the data center. This diploma thesis presents a policy for energy aware consolidation of Virtual Machines that takes into account the utilization of both CPU and Storage.

Key words

cloud, virtual machine, storage, I.O., effect, simulation, CloudSim, dynamic placement, migration

Ευχαριστίες

Η διπλωματική αυτή σηματοδοτεί το τέλος της φοίτησης μου στο τμήμα Ηλεκτρολόγων και Μηχανικών Ηλεκτρονικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου. Πιστεύω λοιπόν πως οι τελευταίες παράγραφοι της διπλωματικής θα πρέπει να αφιερωθούν στους ανθρώπους που με βοήθησαν να επιτύχω σε αυτή την προσπάθεια.

Αρχικά, θα ήθελα να ευχαριστήσω την οικογένεια μου που μου πρόσφερε απλόχερα την δύναμη της όποτε την είχα ανάγκη και τους φίλους μου που ήταν εκεί όποτε ένιωθα την ανάγκη να ξεφύγω από τις υποχρεώσεις της καθημερινότητας.

Ακόμα, θα ήθελα να ευχαριστήσω όσα μέλη του διδακτικού προσωπικού της σχολής απολαμβάνουν ακόμα να διδάσκουν και ειδικότερα τους καθηγητές Νεκτάριο Κοζύρη και Παναγιώτη Τσανάκα για την καθοδήγηση και βοήθεια τους.

Τέλος, θα ήθελα να ευχαριστήσω τον υποψήφιο διδάκτορα Ευάγγελο Αγγέλου για την πολύτιμη βοήθεια του σε κάθε πρόβλημα που αντιμετώπισα κατά την διάρκεια της διπλωματικής.

Γιάννης Σπηλιόπουλος,

Αθήνα, 29η Ιουλίου 2014

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

When W. Shockley, J. Bardeen and Walter Brattain developed the first transistor, in 1947, they inadvertently initiated the transformation of our society from an analog to a digital one. The effects of this ongoing shift can be seen almost in every human activity, from using a mobile device to communicate with friends through a social network to utilizing entire data centers to shift through petabytes of data to spot new subatomic particles.

It is true that humans had been building machines to automate computation for more than two millennial before the development of the transistor, even very complex machines like the *Antikythera mechanism*[13] (early 1st century BC). Furthermore, these devices were not just academic eccentricities, they also had business applications (LEO I[1]). Most importantly, they played a critical role in ending World War II[35].

Despite their many achievements, the different technologies behind these early machines lacked the feature that gave transistor its' transformative power, *low cost*. As every new technology, transistors at first had several significant issues and their commercial status at that point was best described by Donald G. Fink's analogy:

> "Is it a pimpled adolescent, now awkward, but promising future vigor? Or has it arrived at maturity, full of languor, surrounded by disappointments?" [10]

Fortunately, this stage was short-lived and soon technological breakthroughs like Shockley's junction transistor and Integrated Circuits revealed the true potential of transistors. The first transistorized computers for the commercial market appeared in 1957 and 1958 and compared to their vacuum-tubed predecessors, where found superior in every way - smaller, faster, more reliable and economical, and much more powerful.[4]

Obviously, computing would have never become such an essential part of our lives if, as Gordon Moore observed[23] in 1965, the density of transistors at minimum component cost had not doubled approximately every two years. This exponential rate in technological advancement resulted both in halving the prices and doubling the performance per watt every two years. This trend has held for more than 50 years and it is expected to continue for the near future.

As prices plummeted and performance and efficiency climbed, both the public and private sectors started acquiring mainframes, big devices that were used, mainly, in a time-shared fashion. Soon after that, fast and inexpensive microprocessors made their appearance and

[1] http://www .leo-computers.org.uk/

IT infrastructure shifted to collections of commodity servers that had to stay close to where it would be used due to the unavailability of efficient computer networks. These early data center where configured to handle the theoretical load peaks of the organization they belonged to. As a result they were often underutilized. In addition to that they had a high upfront cost and a high maintenance cost.

Increasingly fast communication networks provided people a way to alleviate these problems. In a way similar to how efficient transmission of electricity converted power generation from a distributed model to a centralized one, data centers could now be consolidated and computing could become a utility.[6]

1.1 Cloud Computing

In recent years a new paradigm started forming around the concept of utility Computing; *Cloud computing*. Though Cloud computing is still an evolving model, it is generally accepted that it possesses five essential characteristics[22]:

- *Resource pooling*: All computing resources (e.g. storage, network bandwidth, memory and processing) are pooled and are used to serve multiple users in a multi-tenant model. The resources are dynamically allocated depending on customer demand. Commonly the only control that clients may have over the physical resources they are assigned is the Service Level Agreement[2] they have established with the provider.
- *Measured service*: To optimize their resource usage and provide accurate accounting, Cloud systems incorporate a metering capability at some level of abstraction appropriate to the type of service (e.g. storage, processing, bandwidth, and active user accounts). This is a prerequisite for reliable and fair SLAs.
- *Broad network access*: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).
- *On-demand self-service*: Clients are able to change the amount and/or type of provisioned resources *without any human* interaction with the service provider.
- *Rapid elasticity*: Resources are be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. Essentially, the client has an abstract view of a data center with unlimited capabilities that can be used at any time.

Rapid elasticity is maybe the defining characteristic of cloud computing and, along with the elimination of upfront costs, one of its' strongest selling points. It provides perfect scalability of performance versus cost. Using 1000 computers for 1 hour should cost the same as using 1 computer for a 1000 hours.

Additionally to these five essential characteristics, Cloud computing has three service models[22]:

[2] SLAs are, usually formal, agreements between a service provider and a customer. They are used to define (among other things) the service provided, performance metrics, Quality Of Service guarantees, rewards and penalties.

- *Software as a Service (SaaS)*: Clients are provided with access to a set of applications, specified by the cloud provider, running on the cloud infrastructure. The client has control only over application specific configuration settings. Underlying layers including, among others, hardware and operating systems are completely abstracted out.
- *Platform as a Service (PaaS)*: Clients are given the capability to deploy on the cloud infrastructure applications of their own choosing. Clients are limited by the platform (e.g. the programming languages, libraries, services and tools) supported by the provider. As before the client has no control over underlying layers but only over the deployed applications and perhaps over some configuration settings for the application-hosting environment.
- *Infrastructure as a Service (IaaS)*: Clients are given access to a certain amount of computing recourses like storage, processing power, memory and networking. The client has unlimited control over the software he deploys including operating systems and applications. Beyond that, the user has no control over the underlying cloud infrastructure. This is perhaps the model that closer resembles utility computing.

At this point, an observant reader may wonder how is it possible for a client of an IaaS provider to be provisioned with a certain amount of computing resources but have no control over the underlying infrastructure. In addition to that, how can the provider redistribute his resources without disrupting the operation of his clients while reallocating them. To make all this possible Cloud computing utilizes *hardware virtualization*.

1.2 Virtualization

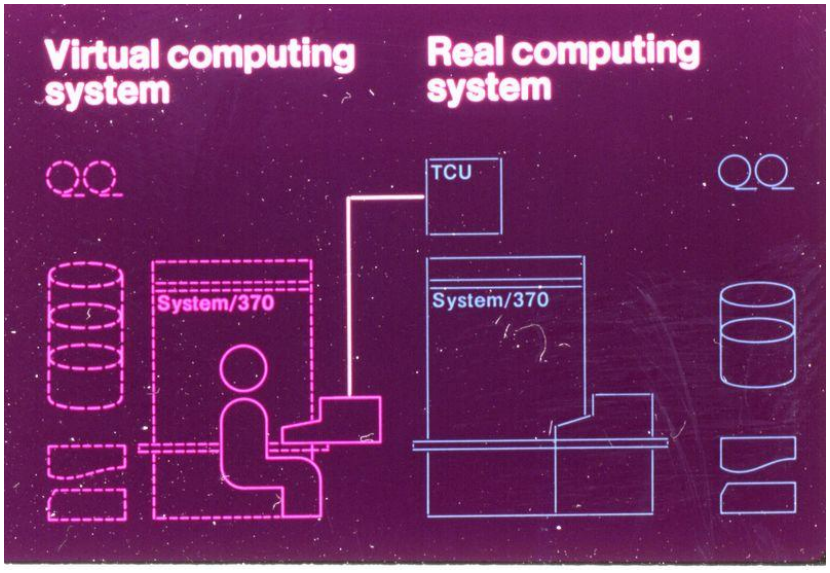

Figure 1.1: Diagram used in the IBM VM/360 product announcement

Virtualization is the concept of creating and interacting with a virtual representation of something. This concept has been used extensively in computing to create abstractions of underlying infrastructure including memory, storage and networks. It can trace its' routes to the

age of the mainframes and IBM's efforts to create an operating system for S/360 that supported time-sharing, after S/360 was rejected from Bell Labs and they lost the Project MAC competition to General Electric[33].

Formally defined, virtualization is the construction of an isomorphism that maps a virtual device (or guest) to a physical device (or host)[30]. As illustrated in Figure 1.2 the isomorphism should map the state of the guest to the state of the host and any sequence of actions on the virtual device to a sequence of actions on the physical device.

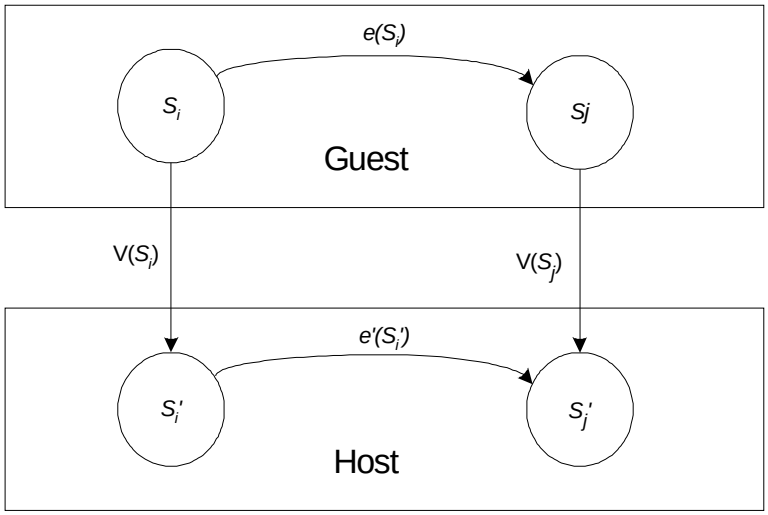

Figure 1.2: Virtualization Isomorphism

Obviously, we could modify the underlying isomorphism without modifying the virtual device. This allows to move a virtual device between hosts that provide this translation layer.Although this provides us with the option to abstract away details of physical devices, that is not strictly necessary. A virtual device could be a lot more complex than the host it is run on.

Hardware virtualization refers to the concept of creating a *virtual machine*(VM) that acts like a real computer. That provides us with a way to create any number of VMs in a physical device with limited resources, while providing the users the illusion of direct access to their own hardware.

Modern hardware virtualization systems have the following attributes [15]:

- *Isolation*: A fault in one virtual machine should not affect another virtual machine that runs on the same host. Additionally resource allocation is managed so that one virtual machine's performance is isolated from another's
- *Encapsulation*: Everything that is considered as a virtual machine's state can be captured into one simple file representing that virtual machine.
- *Portability*: A virtual machine can easily be transfered between physical devices that provide the same virtualization system.
- *Interposition*: All actions originating from a virtual machine pass through a monitoring layer. That layer can inspect, modify and even deny operations.

Clearly utilizing hardware virtualization in a cloud computing system provides an easy way to:

- Maximize uptime of services. A server can easily be migrated from a host that has to be shutdown (i.e. for a scheduled or unscheduled maintenance)
- Efficiently use available resources. A host can accommodate several different services, each in a separate virtual machine, instead of using one host for every service.

1.3 Energy Consumption Considerations

The incredible growth of the data center sector has attracted the attention of many who wish to investigate the unintended impacts of their use. One of the side effects gathering a lot of attention[19, 5, 18] is the amount of electricity that data centers use.

At first glance that may not seem like a great problem since according to *Koomey's Law[17]* the amount of energy needed at a fixed computing load decreases by a factor of two every year and a half. This however is not the case.

According to J. Koomey and data provided by IDC (see Figure 1.3) from 2000 to 2005 the number of installed servers almost doubled. From 2005 to 2010 there was a significant slow-down of installed server growth (only 128%) that was attributed to the financial crisis of 2008. This, however, was just a temporary respite; IDC predicted on 2012 that rapid growth of data centers should be expected for the coming years.

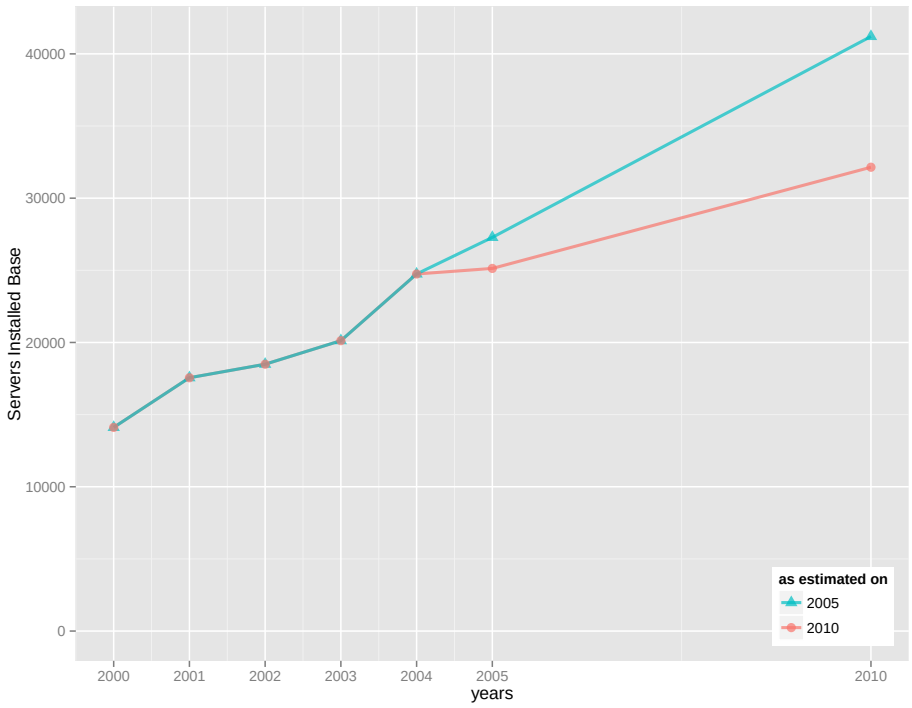

Figure 1.3: Servers installed base as estimated on 2005 and 2010

Additionally, the growth of performance of a typical server outpaced by far the growth of performance per watt (Figure 1.4).

As a result worldwide electricity use for data centers rose from 35.4 in 2000 to 76.2 in 2005 to an estimated value between 130.2 and 92.0 Billions kWh in 2010 (Figure 1.5). To put that in perspective 10 million typical US residential consumers would use approximately 108 Billion kWh[3].

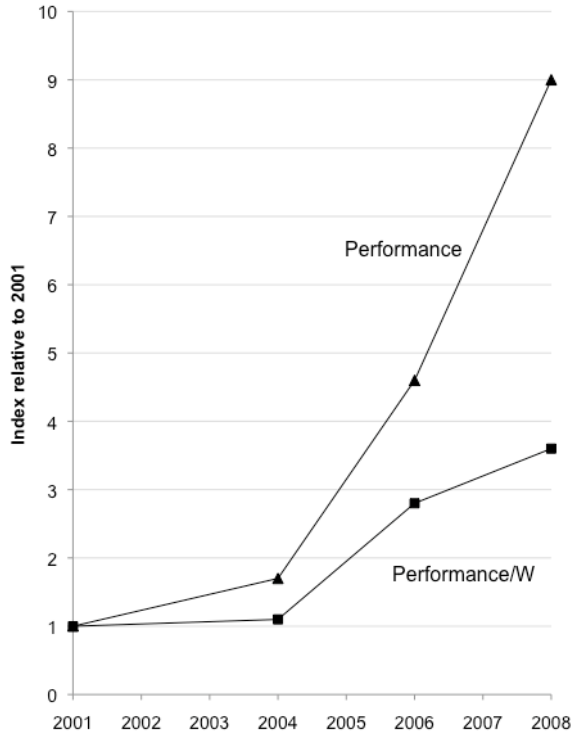

Figure 1.4: Performance and Performance per Watt for a typical server

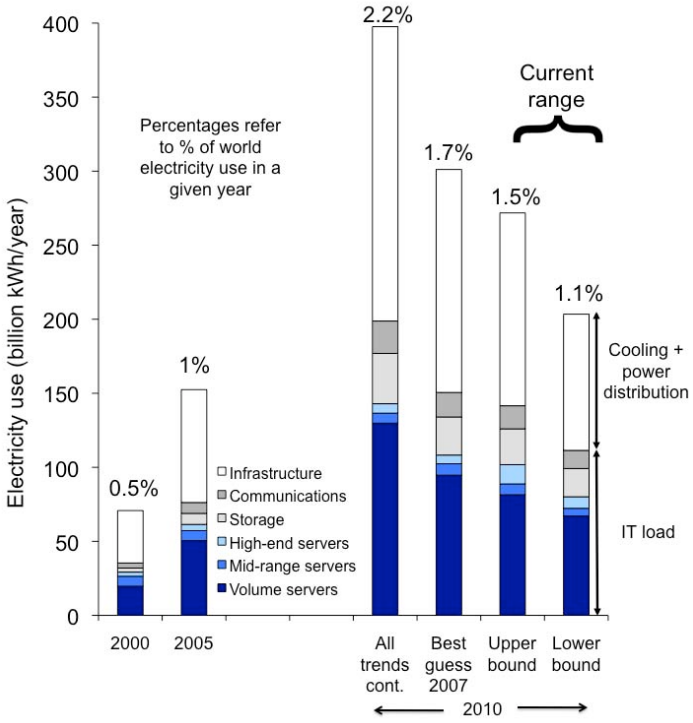

Figure 1.5: Worldwide electricity use for data centers

To make matters even worse, due to Laundauer's Principle [4], Koomey's Law has an expiration date that if the current trend holds will be reached by 2048.

In addition to the strain data centers put to electrical systems worldwide, they are among the biggest producers of greenhouse gases. It was estimated that in 2007 the data center sector produced 116.2 Megatons of CO_2 equiv. y^{-1} or about 0.3% of the global production of greenhouse gases [34]. In an article published in 2013, Masanet et al. estimated that a typical US data center with 20,000 volume servers and a US average PUE produced 59 kilotons CO_2 equiv.y^{-1} [21].

Beyond the obvious ecological considerations, there also serious financial issues with the sustained growth of data centers. By some estimates the cost of cooling and power for a data center has exceeded the cost of the it equipment they support [27].

It stands to reason both ecologically and financially to find a way to decrease the electricity consumption of our data centers.

[3] Average annual electricity consumption for a U.S. residential consumer: http://www.eia.gov/tools/faqs/faq.cfm?id=97&t=3

[4] http://en.wikipedia.org/wiki/Landauer%27s_principle

1.3.1 Fighting back

Techniques[5] for decreasing energy consumption of a data center can be classified to 2 broad categories:

- Decreasing the consumption of a single server
- Decreasing the consumption of a cluster of servers

Techniques to decrease the energy consumption of a single server include *Dynamic Component Deactivation* (DCD) and *Dynamic Performance Scaling*, both of which operate at the hardware level. DCD as its' name suggests deactivates components of a server when the implemented policy decides that there is a significant chance they will be idle. For components that support a range of power and performance modes we generally use DPS. For example typical current generation CPUs support *Dynamic Voltage and Frequency Scaling* (DVFS). DVFS is able to reduce the power consumption of a CPU by reducing the frequency at which it operates as shown by

$$P = CfV^2 + P_{static} \tag{1.1}$$

where P is the power consumption of the CPU, C is the capacitance of the transistor gates (which depends on feature size), f is the operating frequency, and V is the supply voltage. V is determined by f and if f is reduced then V can also be reduced. Due to the V^2 relationship between P and V, reducing the voltage can lead to significant reductions in Power[29].

Nevertheless, a typical energy-efficient server has a small dynamic range as can be seen in Figure 1.6. This power penalty that we pay for just powering up a server, even if we leave him idle, is comprised of two components:

1. The static power consumption P_{static} of a CMOS circuit. Typically the static power consumption of CMOS had been praised as low. However, static power dissipation increases considerably with shrinking dimensions and lower operational voltages. As a result reducing static power has become a major concern in recent generations of CMOS circuits.
2. The power consumption of server components that do not provide either DPS or DCD techniques for managing their power consumption and components with a narrow power range. Processors that typically were the largest power consumer are being displaced by memory (see Figure 1.7) a component with a dynamic power range of less than 50%. Other components with narrow range are hard disk drives (2.5%) and networking switches (15%).[1]

[5] We will consider only techniques that control power dynamically and not techniques like the optimization of component circuitry.

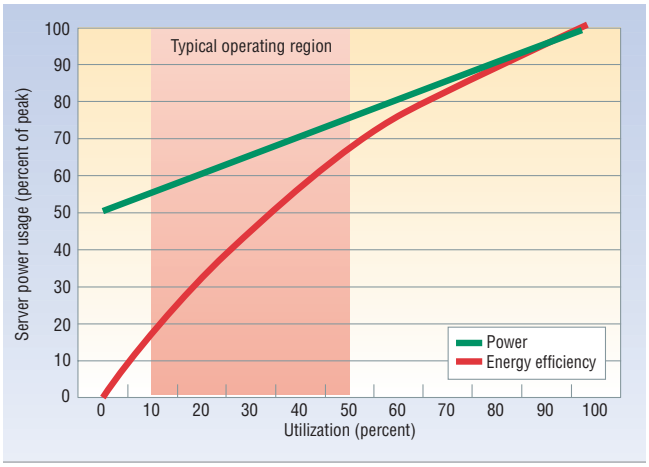

Figure 1.6: Dynamic range and energy efficiency of a typical server[1]

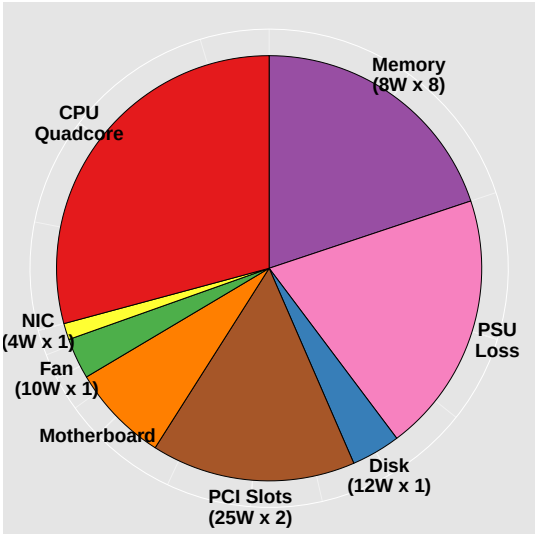

Figure 1.7: Power consumption of each server component[26]

As can be seen in Figure 1.6 a server's energy efficiency increases as its' utilization increases and is most efficient at 100% utilization. That leads to one of the most important ideas in decreasing energy consumption in a cluster of servers, consolidating applications from different underutilized servers in a single one and switching off the rest. This process is known as *workload consolidation*. To illustrate the power savings we can achieve by applying workload consolidation we could look at Figure 1.8.

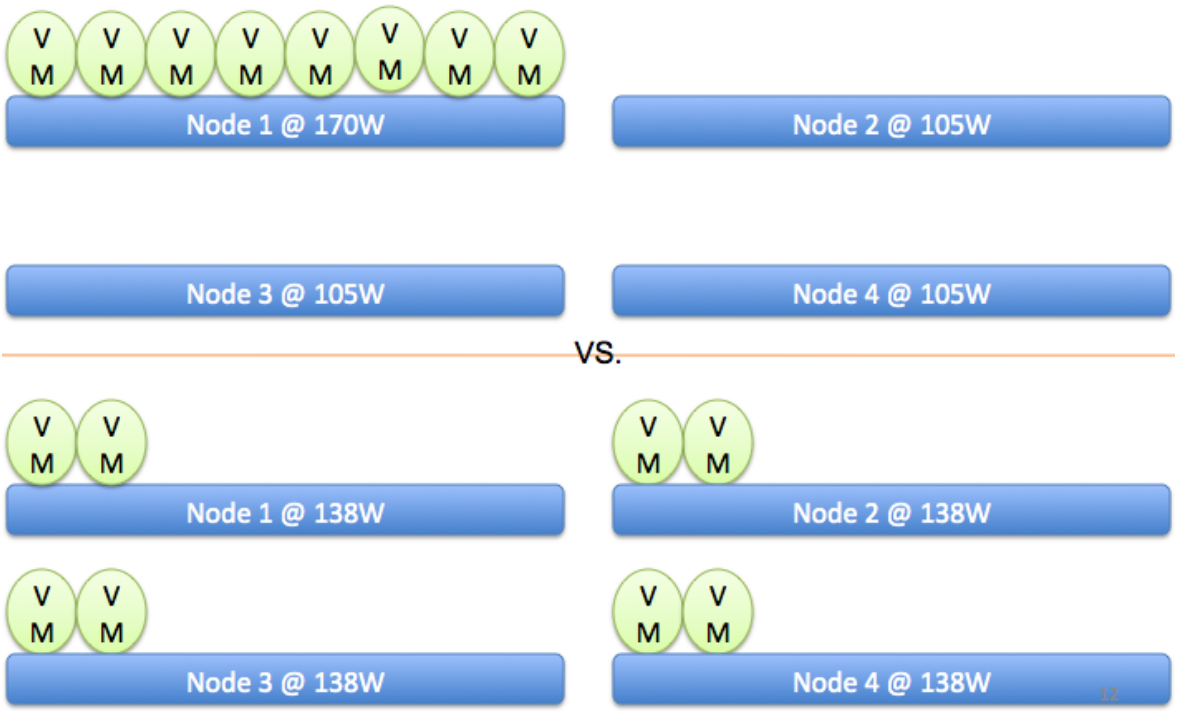

Figure 1.8: Balanced workload versus Consolidated workload

By balancing the workload through the 4 servers we get a total power consumption of 552W. If the servers support DVFS then by consolidating the workloads we get a power consumption

of 485W, almost 70W less. While DVFS gave us some good results, by switching off the now idle hosts we achieve a power consumption of just 170W (or 69% savings).Dynamic workload consolidation, however, has one significant drawback; you risk degrading your quality of service by over-utilizing your servers.

In 2008 a survey by consulting firm McKinsey found that server utilization in data centers rarely exceeds 6%[12]. This is caused because data centers are mostly designed to handle peak loads and therefore are aggressively over provisioned. Consequently, there is significant room for workload consolidation without degrading the quality of service

Figure 1.9 shows beautifully that the efficiency of IT equipments is the most significant factor in reducing both the energy consumption and green house gases production. In the same article from which this figure was retrieved, Masanet et. al conclude that PUE, industry's de facto data center energy-efficiency metric, is a suboptimal metric and should be replaced by a metric of equipment efficiency.

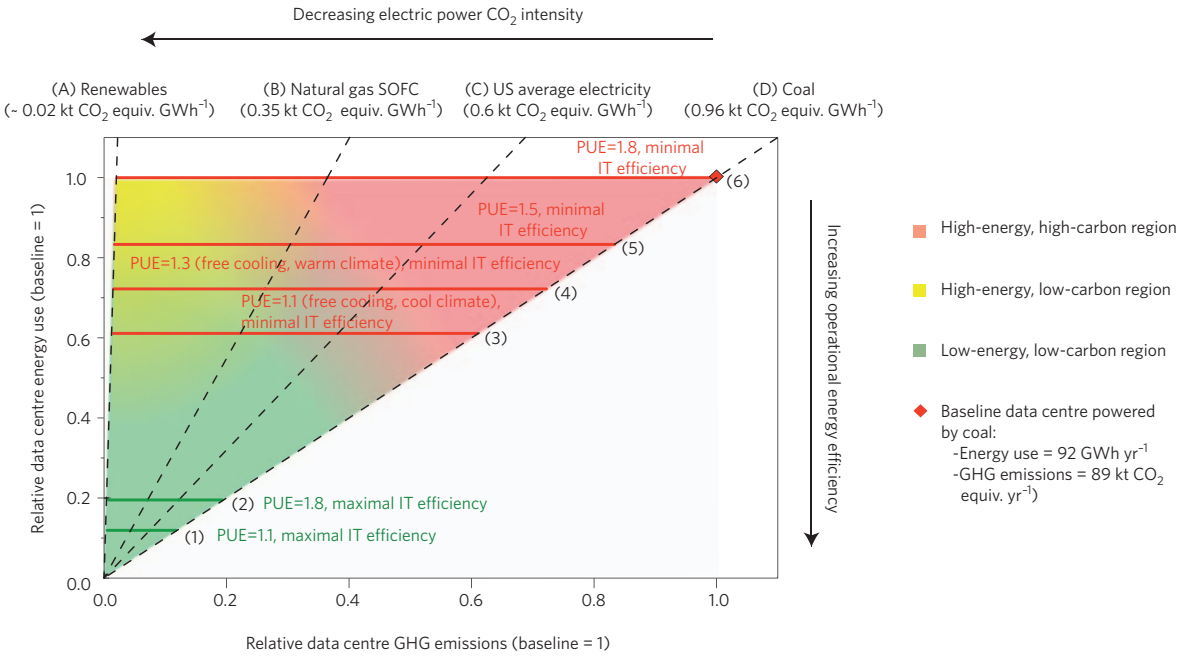

Figure 1.9: Energy-carbon performance map[21]

Another way to decrease the power consumption of a cluster of servers is by looking at the power consumed by supporting machinery specifically the cooling infrastructure. Modern blade and 1U rack servers due to their high power density and complicated heat dissipation require almost as much energy to cool them as running them.

One way to tackle this problem is by continuously monitoring the temperature of the servers and changing the placement of the workloads when servers become overheated. Unfortunately this conflicts with keeping the servers at high utilization in order to achieve high efficiency. Additionally, cooling problems can be easily mitigated by choosing a location that provides free cooling for a data center or recycling the heat produced by the servers. An example of this approach is the Calcul Quebec data center that uses the heat from the servers to produce hot water for the Laval university campus and cold outside air to cool the data center.

1.4 Thesis motivation and scope

As we have demonstrated in the previous sections it is imperative to decrease the amount of energy that our data centers consume. Additionally, we showed (1.9) that increasing the efficiency of our servers is the most important factor of energy savings and (1.6,1.8) that switching off most servers and maximizing the utilization of the rest leads to increased efficiency and energy savings.

Current approaches in workload consolidation use almost exclusively cpu and ram utilization as a metric for server utilization. However, it is our belief that since many workloads are at some degree bound by the performance of storage devices, storage utilization should be included as another metric of server utilization.

As a result, this thesis presents a study of the effect that storage utilization has on dynamic workload placement that minimizes the costs of energy consumption and SLA violations in an IaaS environment (1.1).

Chapter 2

Virtual Machine Placement

2.1 Related Work[1]

There has been extensive research on the subject of efficient workload allocation (or VM placement when virtualization is used) in data centers. In the following section we will briefly present important work that we believe relates with ours.

2.1.1 Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems

In 2001, Pinheiro et al.[25] proposed a strategy for distributing workload in a non-virtualized cluster of servers. It was based on the concept of what they called *Load Concentration*; distribute the incoming load to the smallest possible subset of servers while honouring SLAs. This come in contrast with previous research in workload placement that focused mainly on balancing load between all available servers, and with previous research in power conservation which was focused on battery powered devices.

The authors included in their model 3 resources cpu, network and disk. Additionally, they used throughput and execution time as proxies for QoS. To predict performance degradation they used a very simple model based on the demand for resources of each load. Adding or removing a node is done in a per node basis and depends on whether the current performance degradation is acceptable. In their implementation, the number of active nodes can only change by one at a time since as they claimed reconfiguration operations are time-consuming. As a result, it is unsuitable for fast changes in load. Finally they chose not to take into account that servers have different power demands at different utilization levels or consider heterogeneous clusters.

2.1.2 Managing energy and server resources in hosting centers

Almost concurrently with Pinheiro et al., Chase et al.[7] proposed a technique based on the same idea of dynamically scaling back computing resources, by putting idle servers on low power modes (e.g sleep, shut down), when a certain QoS can be provided. Their proposed

[1] This section is based on the following two studies of the state of the art in energy-efficient computing systems: [16], [3]

architecture was designed for Internet hosting centers, therefore they assumed that servers are shared among multiple service applications. Furthermore, they defined SLAs in terms of throughput and latency.

Their allocation scheme is based on a resource economy where each customer bids for resources, a function of volume and quality. After that the system tries to maximize the utility by balancing the inherent cost (consumed energy) of used resources with the profits and penalties from the defined by the bids.

In contrast to the approach of Pinheiro et al. they chose to model only processing time as a resource and they did not take into account the cost of cluster reconfiguration (i.e. activating/deactivating servers). Similarly to Pinheiro et al. their proposed implementation handles only homogeneous pools of servers.

2.1.3 Energy Conservation in Heterogeneous Server Clusters

The first investigation of workload distribution in heterogeneous clusters was made by Heath et al.[25]. They model both resources (e.g. cpu, disk) and requests in terms of throughput. Then they used analytical models of the expected load to predict overall throughput and power consumption as a function of the request distribution. To find the request distribution that minimizes the ratio $\frac{power}{throughput}$ for each workload intensity level they use simulated annealing. Because this optimization step is time-consuming, it is done offline. A weakness of this approach is that it is not workload agnostic, since you need prior knowledge to compute the throughput demand of each request type.

2.1.4 VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems

Nathuji and Schwan[24] were the first to consider ways to integrate power management mechanisms and policies in virtualization technologies used in data centers. They wanted to take advantage of the power policies already included on guest VMs. As a result, they extended the Xen hypervisor to export to guest VMs *soft* version of the hardware power states for which their policies are designed.

After intercepting the guest VMs' ACPI[2] calls, the systems uses them as hints for actual changes in the hosts' hardware power states and in the allocation of resources to VMs. They also provided an alternative to hardware resource scaling, *soft* scaling that utilizes the hypervisor's scheduling attributes for a VM to emulate more power states than provided by the host.

Finally the authors proposed a two level approach in resource management, splitting it into local and global policies. Local policies, running on each physical machine, driven by VMs's desired power states use state-based guidance to determine the appropriate *shadow* state for each VM. Global policies use information about the shadow states assigned to VMs by local policies and the platforms on which they run to consolidate them.

[2] Advanced Configuration and Power Interface (ACPI) is an open industry specification for power management. http://www.acpi.info/

The authors considered only CPU as a resource.

2.1.5 Power and Performance Management of Virtualized Computing Environments Via Lookahead Control

Kusic et al.[20] researched the problem of dynamic provisioning VMs for multi-tiered web applications. They model SLAs as stepwise pricing function of the response time. The service provider for $r_t > r_{thr}$ receives revenue, while for $r_t < r_{thr}$ pays penalties. The objective is to maximize the provider's profit by minimizing costs caused by power consumption and SLA violations.

They tackle the problem of maximizing profit as one of sequential optimization and address it by Limited Lookahead Control (LLC). LLC provides at each step a decision for the number of active hosts in the data center, the number of VMs to allocate for each service, the CPU share allocated to each VM and the fraction of the workload to allocate to each VM. DVFS is not performed since it is deemed by the authors that it provides only a small power reduction compared to migrating all VMs from a lightly loaded host and switching it off. Additionally their model includes the costs incurred by the time lost while switching hosts on and off.

The authors, while implementing their proposed approach, came against the exponential increase in worst-case complexity with longer prediction horizons and increasing number of control options. They tried several optimizations, including decomposing the problem int smaller sub-problems and local-search techniques. Nevertheless, complexity still remained high and the average execution time for the LLC just for 15 hosts is about 30 minutes. In addition, the controller requires simulation-based learning to make application-specific adjustments. Finally the authors considered only CPU as a resource.

2.1.6 Energy Aware Consolidation for Cloud Computing

Srikantaiah et al.[31] experimentally studied the relationship between performance, energy consumption and resource (e.g. CPU and disk) utilization for applications serving stateless requests in a data center. The experimental results (see Figures 2.1, 2.2) allowed them to reach several interesting conclusions. Following that, they used they insights gained from the experimental study to propose an energy aware consolidation algorithm.

As can be seen from Figure 2.1 the energy per transaction has a surface similar to an elliptic paraboloid. The power consumption is influenced both by CPU and disk utilizations and the minimum power consumption is achieved at approximately 70% CPU utilization and 50% disk utilization. At first glance, this seems to conflict previous data that showed that a server get more efficient with higher utilization. That is not the case; high resource utilization results in performance degradation and consequently longer execution time. Additionally, as the authors note, this energy optimal point of operation is dependent on the performance degradation levels that we can tolerate. Another interesting conclusion is that energy consumption is more sensitive to variations in CPU utilization as can be seen in the steeper gradient of the curve along the CPU utilization axis.

Furthermore, from Figure 2.2 they reasoned that disk utilization is more significant than CPU utilization for performance degradation and that disk utilization is a limiting factor for consolidated performance.

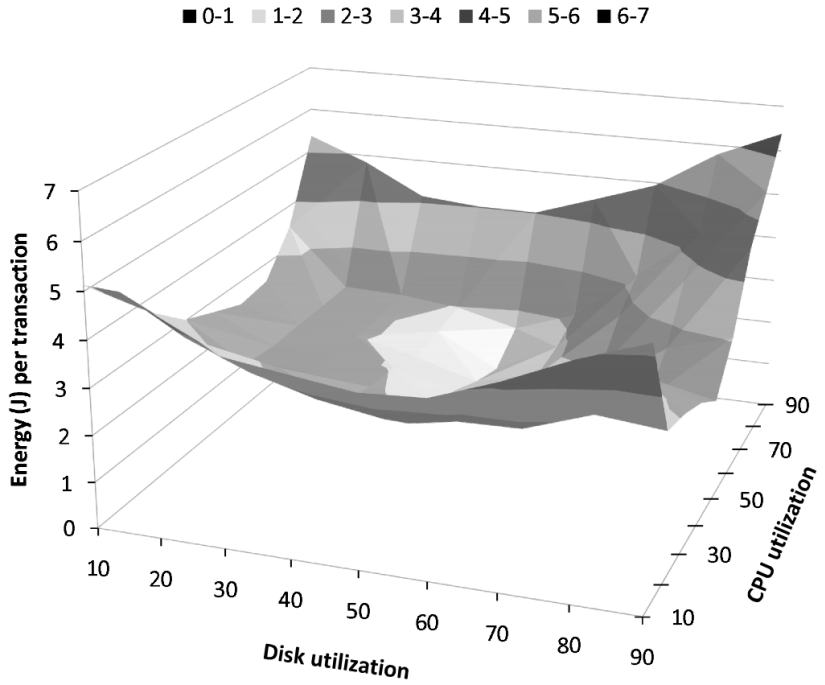

Figure 2.1: Energy consumption per transaction

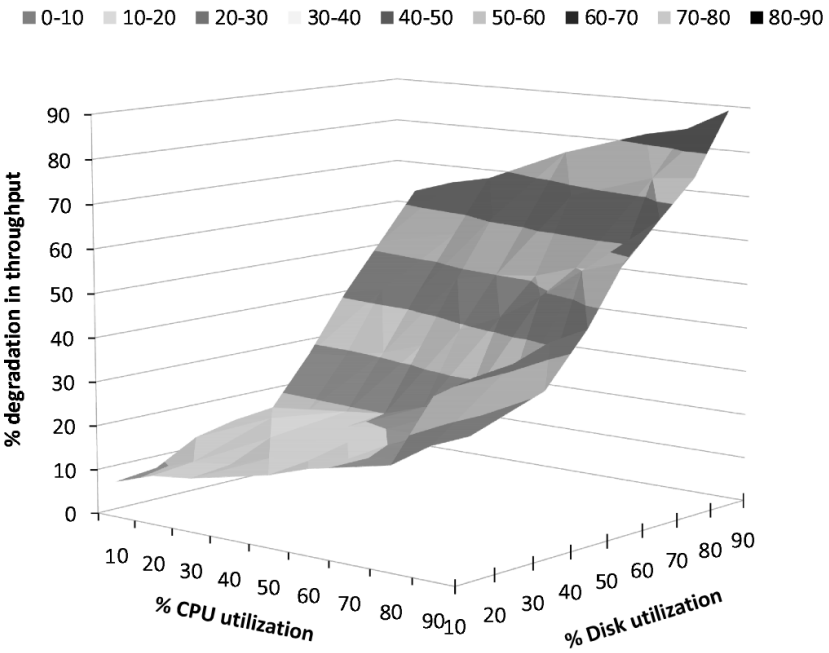

Figure 2.2: Performance degradation

The goal of their proposed consolidation algorithm is to keep servers at utilization levels that efficiently amortize the idle power costs and in the same time evade energy penalties due to internal contentions. They modeled the problem as a 2-dimensional bin packing problem. Each active server represents a bin and each dimension a resource of that server. The bin size along each dimension is the optimal utilization level as determined by the experimental

study. Based on their intuition that after an allocation we can use a bin to each fullest potential if there is maximum space in each dimension, the authors used as a placement heuristic the minimization of the 2-dimensional Euclidean distance.

Experimental evaluations showed that their proposed algorithm achieved energy consumption 5.4% greater than the optimal at 20% performance degradation tolerance. A drawback of the proposed method is not including in their model the costs of migrations and switching host on/off. Another one is the necessity of experimental study to determine the resource requirements of applications and the optimal utilization levels of the hosts.

2.1.7 Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers

Beloglazov et al.[2] studied the energy-performance trade-off in a Cloud computing environment. They argued that due to the inherent variability of workloads experienced in such an environment, VM placement should optimized continuously in an online manner. To understand the implications of the online aspect of the problem, they used competitive analysis to study the *single virtual machine migration problem* and the *dynamic virtual machine consolidation problem*.

They modeled the single VM migration problem as a single Host with several VMs allocated to it. The host is oversubscribed; that is if all VMs request their maximum CPU performance, specified in the SLA, the total CPU demand exceeds the host's capacity. The goal is to select the time at which to migrate a single VM from the host in order to minimize costs from energy and SLA violations. They proved the cost of the optimal offline algorithm and the ratio of the cost of an optimal online deterministic algorithm to the cost of the optimal offline algorithm.

Consequently, they modeled the dynamic virtual machine consolidation problem as multiple hosts and multiple VMs. In this setting VMs have variable workloads and can be migrated between hosts, using live migration, with a VM-specific migration time. Additionally, hosts without VMs are switched off. They proved the competitive ratio for the optimal online deterministic algorithm.

Following that, they propose three metrics for the Quality of Service that they believe are application-agnostic and thus suitable for an IaaS environment. They split the problem of VM consolidation in four distinct parts:

- Determining when a host is overutilized and one or more VMs should be migrated from it.
- Determining when a host is underutilized and all VMs should be migrated from it in order to switch off the host
- Selecting the VMs that should be migrated from an overutilized host
- Finding a new placement for the VMs marked for migration.

They propose several Heuristics for the host overloadind detection and the vm selection parts, a modification of the BFD algorithm for finding a new VM placement, and a simple approach for determining underutilized hosts.

Finally, they used CloudSim[3] and workloads from PlanetLab[4] to evaluate their approach. Using a metric that combines both Energy and Sla violations they statistically significant difference between their approach and both simple DVFS and not using any power management policy at all.

The approach proposed by Beloglazov et al. satisfies all the necessary conditions of an energy-aware consolidation technique for IaaS infrastructure. It is online and consequently it can handle varying workloads. Additionally, its' heuristic nature makes this approach scalable. As a result, this approach has become the basis for our study of the influence of disk utilization on energy-aware dynamic VM placement.

In the following section we will describe the heuristics, proposed by Beloglazov et al., that we have evaluated.

2.2 Evaluated Heuristics

2.2.1 Host Overloading Detection

Static Threshold (*THR*)

The simplest of the proposed heuristics is a static utilization threshold (*THR*). If the utilization of a host exceeds this threshold then one or more VMs should be migrated from the host to prevent potential SLAV. Since fixed values of utilization thresholds are unsuitable for environments with dynamic and unpredictable workloads like Cloud computing data centers, this approach will be used as benchmark for the other heuristics.

Adaptive utilization threshold

The idea behind adaptive utilization threshold is to adjust the utilization threshold based on the variability of the utilization. Higher variability means there is a higher chance that there will be a big outlier that will cause an SLA violation. We use robust statistics to produce estimators of the dispersion of the utilization that are not affected from small departures from model assumptions. We evaluate the following two heuristics based on this idea:

Median Absolute Deviation (*MAD*)

MAD is a robust statistic of dispersion, more resilient to outliers than the standard deviation. This is easily visible when we see the two measures of statistical dispersion in the form of equations.

$$\sigma = \sqrt{E[(X - \mu)^2]} \tag{2.1}$$

$$MAD = median_i(|X_i - median_j(X_j)|) \tag{2.2}$$

[3] CloudSim is a framework for modeling and simulating Cloud computing infrastructure and services: http://www.cloudbus.org/cloudsim/

[4] PlanetLab is an open platform for developing, deploying, and accessing planetary-scale services. It consists of 1211 nodes at 593 sites all over the world: https://www.planet-lab.org/

In standard deviation (Equation 2.1) the distances from mean are squared. As a result outliers weigh more heavily in standard deviation than in *MAD*.

The adaptive threshold is given by the following equation:

$$T_u = 1 - s * MAD, s \in \mathbb{R}^+ \tag{2.3}$$

where s is a parameter that allows the adjustment of the aggressiveness of consolidation.

Interquartile Range (*IQR*)

IQR is the difference between the first and the third quartile. IQR is more robust than total range since it has a breakdown point of 25% and thus more than 25% of the observations have to be outliers for it to be affected.

$$IQR = Q_3 - Q_1 \tag{2.4}$$

The adaptive threshold is given by the following equation:

$$T_u = 1 - s * IQR, s \in \mathbb{R}^+ \tag{2.5}$$

where s is again a parameter that allows the adjustment of the aggressiveness of consolidation.

Model fitting - Piecewise Regression

The idea behind the following two heuristics is to partition the historical data of utilization and fit a simple model to each segment. We then use the fitted model to approximate the next utilization value.

Local Regression (*LR*)

In *LR* we use a variation of the LOESS[8] method. LOESS is a non-parametric regression method and as such it does not make any assumption about the shape of the curve of the samples.

In LOESS for each of the segments of data we perform a weighted least squares regression to fit a low-order (typically the order is 1 or 2) polynomial. The weights are given by the following function:

$$w_i(x) = T\left(\frac{\Delta_i(x)}{\Delta_{max}(x)}\right) \tag{2.6}$$

where $\Delta_i(x)$ is the distance between observation (x_i, y_i) and (x, y), $|x_i - x|$ and T is the tricube weight function:

$$T(u) = \begin{cases} (1 - |u|^3)^3 & \text{if } |u| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

In this variation we fit a polynomial to the last k observations of utilization for a single point; the last observation. Since we want to use this fitted polynomial to approximate the next value we choose a polynomial of degree 1 to reduce bias at the boundary[9].

This algorithm finds a host overloaded when:

$$s * \hat{g}(x_{k+1}) \geq 1, \quad x_{k+1} - x_k \leq t_m \tag{2.8}$$

where t_m is the maximum time required for a migration of any of the VMs currently residing in this host and $s \in \mathbb{R}_+$ is once again a parameter that allows the tuning of the method's consolidation aggressiveness.

Local Regression Robust (*LRR*)

LRR is based on an iterative variation of LOESS, proposed by Cleveland[9], that is more robust to outliers.

Initially, we perform LOESS, as described in *LR* and then calculate the residuals $r_i = y_i - \hat{y}_i$ for each observation (x_i, y_i)

In the second step we compute the robust weights for each observation as shown below:

$$rw_i = w_i(x) * B\left(\frac{\hat{r_i}}{6 * median|r|}\right) \tag{2.9}$$

where B is the bisquare function:

$$B(u) = \begin{cases} (1-u^2)^2 & \text{if } |u| < 1 \\ 0 & otherwise \end{cases} \tag{2.10}$$

And as in *LR* we use weighted least squares to fit a polynomial of degree 1.

We perform these two steps for a number of iterations and then we use the estimated trend line to approximate the next value as in *LR*.

2.2.2 Virtual Machine Selection

When a host is found overutilized, we apply the following policies multiple times, selecting each time one VM to migrate, until the host detection policy decides that the host is not overloaded any more.

Minimum Migration Time (*MMT*)

In the *MMT* policy we choose for migration the VM v that requires the minimum migration time from all the VMs residing in the host. The idea behind this policy is to minimize the costs associated with a VM migration.Migration time is estimated using the following simple model:

$$\frac{Mem_h(v)}{Net_h} \tag{2.11}$$

where $Mem_h(v)$ is the share of host's h RAM allocated to VM v and Net_h is host's h network bandwidth. The condition is formalized below:

$$v \in V_h \,|\, \forall\, i \in V_h,\ \frac{Mem_h(v)}{Net_h} \leq \frac{Mem_h(a)}{Net_h} \tag{2.12}$$

Maximum Correlation (*MC*)

In the *MC* policy we select for migration the VM v, which utilization has the maximum correlation with the utilization of all the other VMs residing in host h. This is based on the idea by Verma et al.[32] that when there is a high correlation between workloads in a server then there is a higher chance that workloads will request their peak utilization at the same time and overload the server.

To evaluate the correlation between the utilization of the VMs we use the *multiple correlation coefficient* [28].

Minimum Utilization (*MU*)

In the *MU* we select for migration the VM v that has the lowest utilization from all VMs residing in host h. Since this policy removes the least amount of requested utilization from the host, it should leave the host in a more efficient state.

$$v \in V_h \,|\, \forall\, i \in V_h,\, U_v(t) \leq U_a(t) \tag{2.13}$$

where $U_i(t)$ is the utilization of VM i at time t.

Random Selection (*RS*)

In the *RS* we select a random VM for migration each time.

2.2.3 Virtual Machine Placement

Beloglazov et al. modeled the VM placement problem as a bin packing problem. They proposed a variation of the *Best Fit Decreasing* (BFD) algorithm that has been shown to use no more than $11/9 * OPT + 1 bins$[36], where OPT is the number of bins used by the optimal solution. They sort all the VMs, that have been marked for migration, in decreasing order of their utilization and allocate every VM to the host that can accommodate it without becoming overloaded and produces the least increase in consumed power after the migration. They call this algorithm *Power Aware Best Fit Decreasing* or *PABFD* (see Algorithm 2.1 for pseudocode).

```
1  Input: hostList, vmList
2  Output: allocationMap
3
4  sort vmList by decreasing utilization
5
6  for vm in vmList do
7      minPower ← MAX
8      allocatedHost ← NULL
9
10     for host in hostList do
11         if host has enough resources for vm ) then
12             power ← estimatePower(host,vm)
13             if power < minPower then
14                 allocatedHost ← host
15                 minPower ← power
16             end
17         end
18     end
19
20     if allocatedHost ≠ NULL then
21         allocationMap.add(vm, allocatedHost)
22     end
23 end
```

Algorithm 2.1: Power Aware Best Fit Decreasing

2.2.4 Underloaded Hosts

To detect underutilized hosts, Beloglazov et al., used a very simple approach. Every host that is not considered overutilized, is considered potentially underutilized.

We pick the host with the least utilization and we try to find a new placement for all the VMs that reside in it, without forcing any host to become overutilized. If such a placement is possible, the VMs are scheduled for migration and once all migrations have been completed, the host is put in a low power state. This step is repeated for every potentially underutilized host. A detailed description in pseudocode can be found in Algorithm 2.2.

```
1  Input: hostList, overutilizedHostList, switchedOffHostList
2  Output: allocationMap
3
4  for host in overutilizedHostList do
5  |  remove host from hostList
6  end
7
8  for host in switchedOffHostList do
9  |  remove host from hostList
10 end
11
12 underutilizedHostList ← hostList
13 newVmPlacementHostList ← hostList
14 host ← findHostWithMinimumUtilization(hostList)
15
16 while host ≠ NULL do
17 |  remove host from underutilizedHostList
18 |  remove host from newVmPlacementHostList
19 |
20 |  vmList ← getListOfVmNotAlreadyInMigration(host)
21 |  hostAllocationMap ← pabfd(newVmPlacementHostList,vmList)
22 |
23 |  for allocation in hostAllocationMap do
24 |  |  allocationMap.add(allocation)
25 |  |  allocatedHost ← getAllocatedHost(allocation)
26 |  |  remove allocatedHost from underutilizedHostList
27 |  end
28 |
29 |  host ← findHostWithMinimumUtilization(hostList)
30 end
```

Algorithm 2.2: Detect underutilized hosts and find a new placement for their VMs

Chapter 3

CloudSim

Our target environment is that of an Infrastructure as a Service data center. Nevertheless, performing large-scale experiments on real infrastructure is extremely difficult. Even if experimenting on a real data center was feasible, we would have no way to produce easily repeatable and verifiable results. As a result we relied on extensive simulations to study the effect of storage utilization on the effectiveness of dynamic power-aware resource allocation.

Our simulator of choice was the CloudSim toolkit. This gave us the additional benefit of being able to reproduce the results of Beloglazov et al. [2] and using them as a benchmark for our approach.

CloudSim is a discrete-event simulator for Cloud environments. In Figure 3.1 we can see the environment modeled by CloudSim.

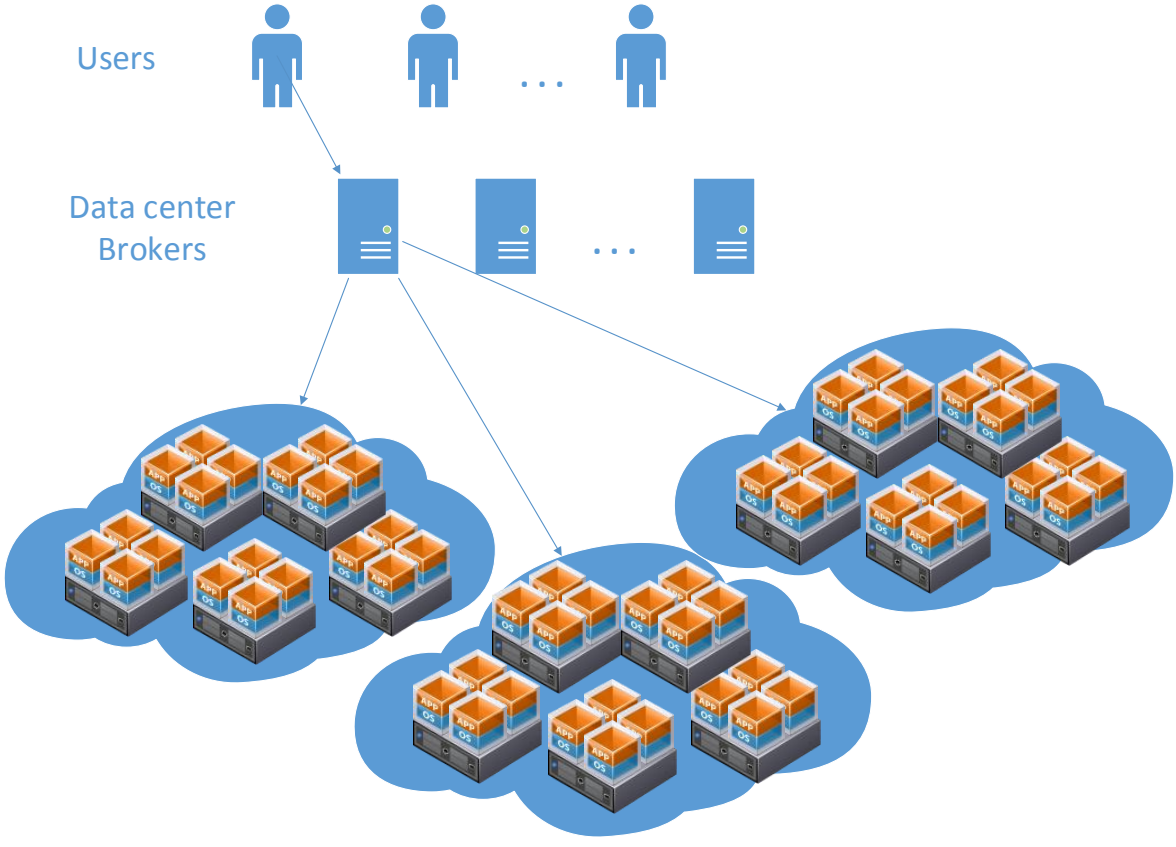

Figure 3.1: Cloud computing environment modeled by CloudSim

There are 6 different entities:

User:
A *User* is a service consumer; someone who pays to receive computing resources from the IaaS provider. That includes individuals, companies, and SaaS and PaaS providers.

Broker:
Brokers are entities that act on behalf of Users. Their role is to search for Data centers IaaS providers suitable for the needs of the User and optionally negotiate a price for the requested resources.

Data center:
Data centers are collections of computing resources belonging to different IaaS providers.

Host:
A *Host* represents a physical server with specific computing resources available to it. A physical server may host 0 or more VMs.

VM:
A *VM* is a representation of the contract between a service consumer and a provider. It specifies the amount of one or more computing resources that should be available for the applications that the consumer want to deploy on this machine.

Cloudlet:
Cloudlets are representations of the applications the consumer deploys.

Since we are not immediately interested in multiple users or multiple providers, we will simplify our model and use a single user and a single provider connected by a broker that simply forwards all requests for computing resources from the User to the provider. Additionally we will simulate only one cloudlet per VM.

In the next section we will briefly present the architecture of the CloudSim toolkit.

3.1 Architecture

User code
At the top-most layer we can specify essential characteristics of the simulation than we want to run, including among others the behavior of the deployed applications, the SLA for the VMs, the resources of the hosts and the behavior of the broker.

CloudSim
The CloudSim simulation layer provides the functionality needed to simulate virtualized data centers. Resource provisioning, application execution and network simulation are all handled at this layer.

User Interface Structures
This layer defines the core entities VM and Cloudlet.

VM Services

This layer provides functionality for the execution of cloudlets and VMs. At this layer we define the logic for dividing the host's resources to the VMs and for deviding the resources available to a VM to its' cloudlets. You could for example simulate a virtual machine with a cpu that is space-shared but not time-shared by extending the Cloudlet Execution functionality.

Cloud Services

The Cloud Services layer is responsible for allocating VMs to hosts and for providing accounting for a host's resources. Alternative migration policies are implemented at this level.

Cloud Resources

At this layer all events related with a data center are handled, including VM migrations and submitting new requests for resources.

Network

As the name suggests this layer is responsible for simulating an underlying network and its' effects on communication between entities.

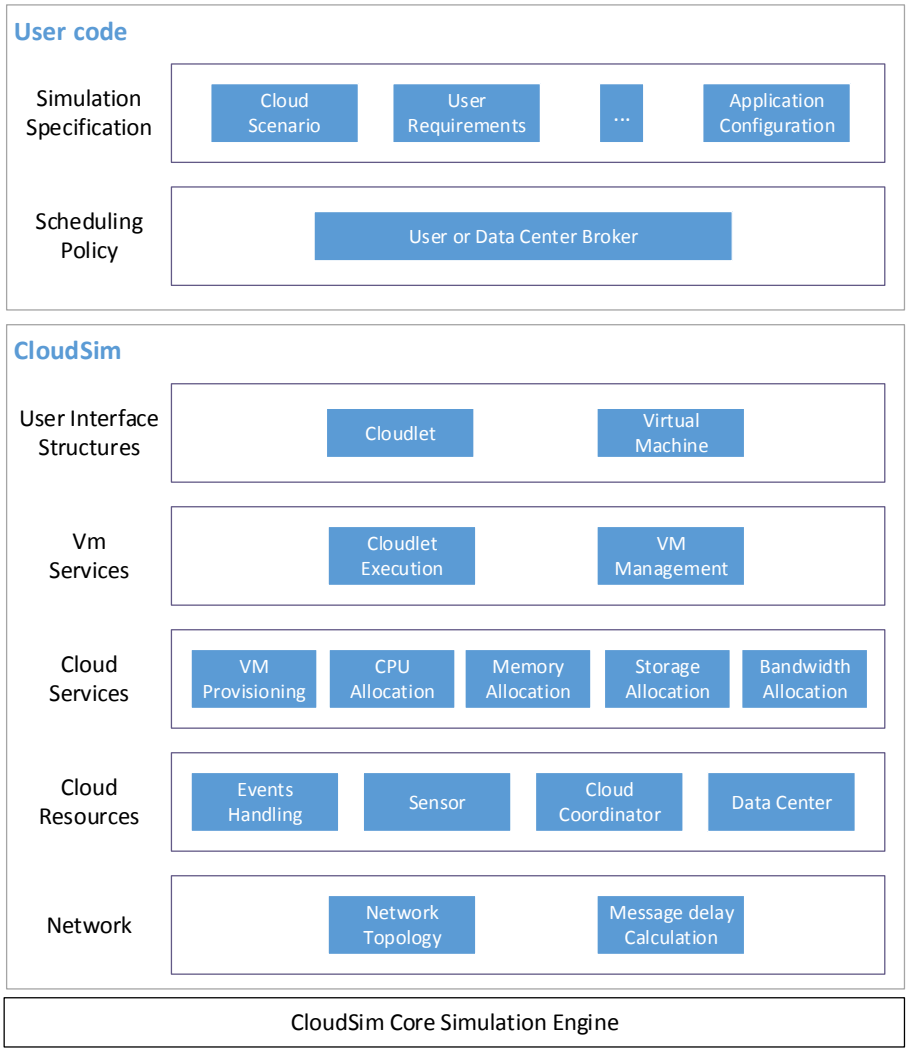

Figure 3.2: CloudSim layered architecture

3.2 Important Entities

CloudSim is a fairly large project with more than 60.000 lines of code. Consequently, it is not feasible to describe it in its entirety. In the following section we will present the most important classes of CloudSim and the extensions we implemented to support storage utilization.

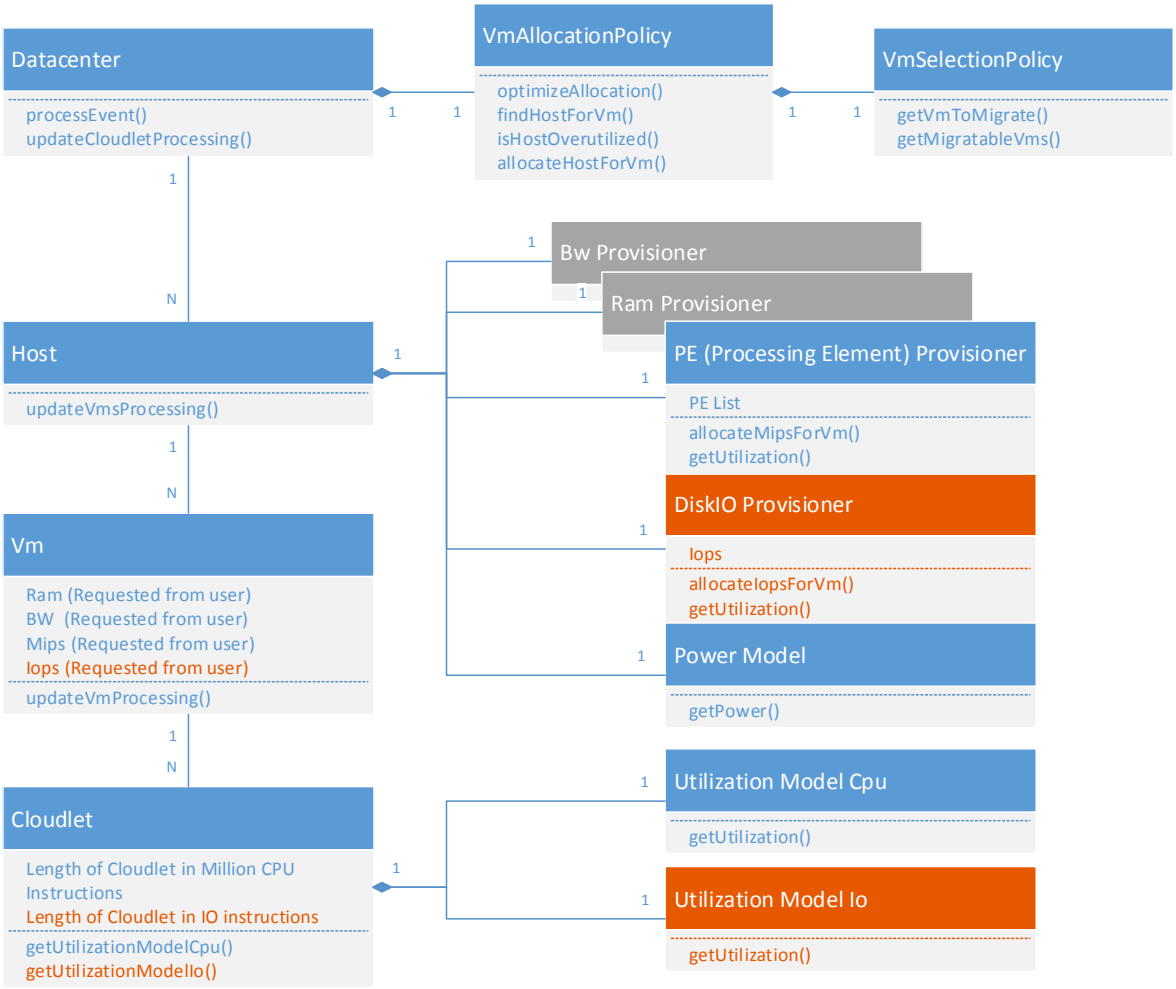

Figure 3.3: CloudSim class diagram

3.2.1 Cloudlet

As we have already said a cloudlet represents an application that can be deployed to a VM. Each cloudlet has a CPU execution length and storage execution length specified in MIPS and IOPS respectively. When a cloudlet has executed at least as many MIPS and IOPS as specified in these two lengths then it is considered completed and the resources that were allocated to that cloudlet are released. Additionally each cloudlet has a utilization model for each of the resources. Since we are mainly interested in studying the effects of storage utilization, we statically allocate network bandwidth and ram (without any over-subscription) and use a zero utilization model for these two resources. This is consistent with the methodology followed by Beloglazov et al. in their simulations.

3.2.2 Storage and CPU utilization models

The utilization models we implemented where tailored to the two sets of workloads available to us:

- The first set was included with CloudSim and consisted of CPU utilizations for approximately 1000 VMs, belonging to the PlanetLab platform, for 10 random days, in March and April of 2011. This was the workload used by Beloglazov et al. We were unable to retrieve the relevant storage utilization information since, as we informed, they were lost after a hardware failure[1]. This workload was used as sanity check for the extensions we had to implement to support storage utilization.
- The second set was provided to us by professor Vivek Pai and were produced by the CoMon Infrastructure[2]. It consisted of CPU and storage utilizations for approximately 500 VMs, belonging to the PlanetLab platform, for 10 days, in July of 2009.

Since the first set of workloads did not supply any information about storage utilization we needed a way to extrapolate reasonable storage utilization from cpu utilization. To do so, we used Amdhal's balanced system law for IO as was revised by J. Gray and P. Shenoy.

> "Random IO's happen about once each 50,000 instructions. Based on rule 10, sequential IOs are much larger and so the instructions per IO are much higher for sequential workloads."[14]

Consequently, for an application that issues mostly random IOs, we issue 20 IOPS for every MIPS.

Furthermore we had to model the effect that waiting for IOs to complete would have on a CPU. We chose the following simple approach:

1. Compute the MIPS and IOPS requested by the Cloudlet based on the utilization values in the workload and the maximum resources specified by the VM (see Section 3.2.3).
2. Store the IOPS requested and the ratio $\frac{MIPS}{IOPS}$.
3. When we learn the resources allocated from the host to this VM (and since we have one cloudlet per VM, to this cloudlet as well), we check if the allocated IOPS are equal to or more than the requested IOPS. If not then we add to the array of outstanding IOPS the number of IOPS that where not completed ($(requestedIOPS - allocatedIOPS)$) together with the ratio $\frac{MIPS}{IOPS}$ we stored in the previous step.
4. While computing the requested MIPS for the next time span we iterate through the outstanding IOPS array and add $outstandingIOPS * \frac{MIPS}{IOPS}$

[1] http://lists.planet-lab.org/pipermail/users/2012-November/004159.html
[2] CoMon provided monitoring statistics for PlanetLab: http://comon.cs.princeton.edu/

3.2.3 VM

A VM has several properties that represent the SLA of the user and the service provider. These properties include:

1. The maximum MIPS the VM can request from the host that if not supplied an SLA violation has occurred.
2. The maximum IOPS the VM can request from the host that if not supplied an SLA violation has occurred.
3. The amount of RAM that must be statically allocated to the VM.
4. The amount of Storage that must be statically allocated to the VM.
5. The amount of network bandwidth that must be statically allocated to the VM.

In addition, a VM has properties for the currently allocated, from the host, MIPS and IOPS. This properties get updated every time an updateVmProcessing event occurs.

Finally a VM has a cloudlet scheduler that provides all the functionality needed for simulating the execution of multiple cloudlets and allocating resources (e.g. simulating a time-shared virtual CPU). However, in our simulations each cloudlet is allocated to a separate VM. As a result, the cloudlet scheduler that we implemented just updates the cloudlet with the currently allocated resources and checks whether the cloudlet has finished its' execution.

3.2.4 Host

A host represents a physical machine. It has provisioners that each represents a resource of the host and a VM scheduler that is responsible for the allocation of resources to the VMs residing in the host. Additionally a host has a power model that estimates the power consumption of the physical machine. Finally each host is responsible for updating the allocated resources and the processing of each of its' VMs.

3.2.5 Provisioners

Provisioners are responsible for the provisioning of a resource. They export functionality to the host for allocating and deallocating portions of the resources to VMs. In addition to that, they provide information about the currently available resources, the allocated resources for a VM and whether a host is suitable for a VM.

There are four provisioners:

1. Network bandwidth provisioner
2. RAM provisioner
3. Processing Elements (PE) Provisioner
4. Storeage IO provisioner

3.2.6 VM scheduler

The VM scheduler uses the functionality exported by the provisioners to implement policies for sharing the resources between multiple VMs running in a host.

The policy we implemented for sharing both MIPS and IOPS is time-sharing with over-subscription. More specifically if the amount of resource requested from all VMs is less than the total amount of resource available to the host then every VM receives the full amount of resource it requested. However, when the total amount of resource requested is greater than the total amount available every VM gets the amount it requested scaled by $\frac{totalAvailable}{totalRequested}$ ((see Algorithm 3.1 for pseudocode).

```
1  Input: vmList, totalAvailable
2  Output: resourceAllocationMap
3
4  totalRequested ← 0
5  for vm in vmList do
6  |  vmRequested ← vm.getRequested()
7  |  totalRequested ← totalRequested + vmRequested
8  end
9
10 scaleFactor ← 1
11 if totalRequested > totalAvailable then
12 |  scaleFactor ← totalAvailable / totalRequested
13 end
14
15 for vm in vmList do
16 |  resourceAllocationMap.add(vm,vm.getRequested() * scaleFactor)
17 end
```

Algorithm 3.1: Time-sharing with over-subscription

3.2.7 Power Model

The power models we implemented extrapolate the power consumption of a server from its CPU utilization based on measurements of total system power against CPU utilization of real systems.This kind of models has been shown to be fairly accurate for modeling a single machine. More importantly, it has been shown that when they are used to model the behavior of a large group of servers, they can achieve errors less than 1%[11].

We use two power models based on the following real servers:

1. Hewlett-Packard ProLiant ML110 G4[3]
2. Hewlett-Packard ProLiant ML110 G5[4]

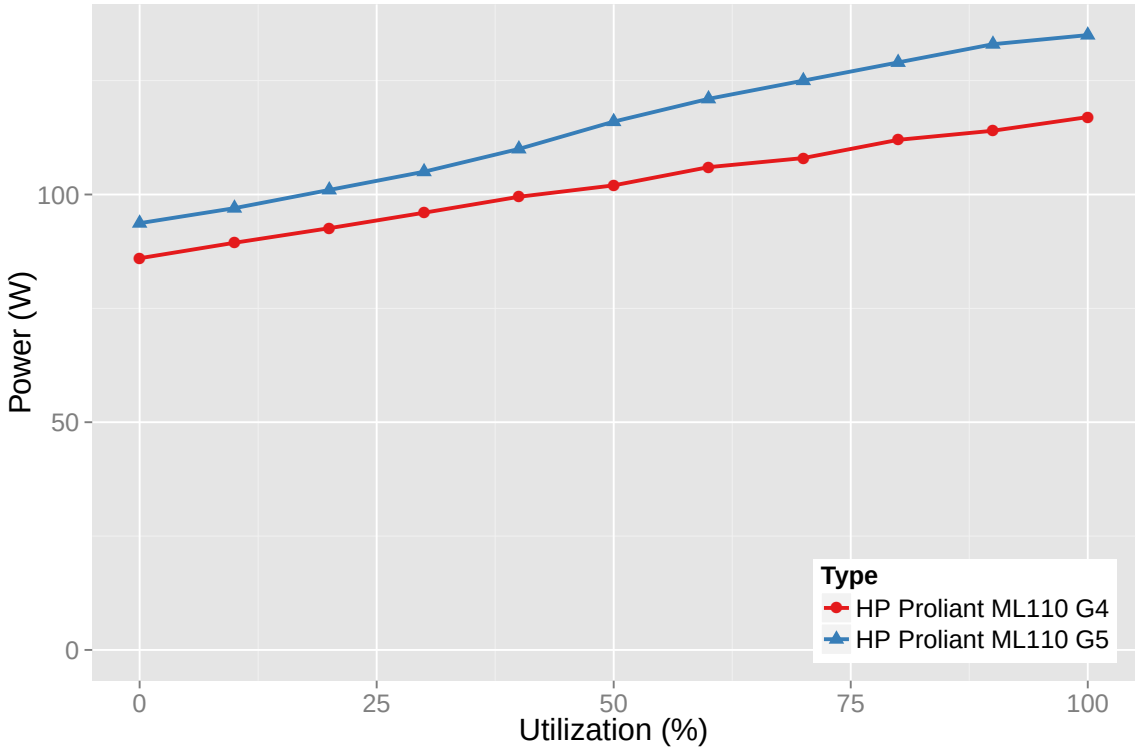

Figure 3.4: Power Models for HP Proliant ML110 G4 and HP Proliant ML110 G5

3.2.8 Datacenter

The datacenter class implements functionality that deals with User/Broker actions like creating VMs or submitting cloudlets for execution and with events that should be handled at the data center level like migrating a VM between two hosts. In addition, the datacenter class has a vm allocation policy that is responsible for allocating a VM to the most suitable host available and even optimizing the allocation of all VMs in the data center. Finally, this class

[3] http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110127-00342.html
[4] http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00339.html

is responsible for updating the processing for every host, VM and cloudlet in the data center (See a sequence diagram of the process in Figure 3.5).

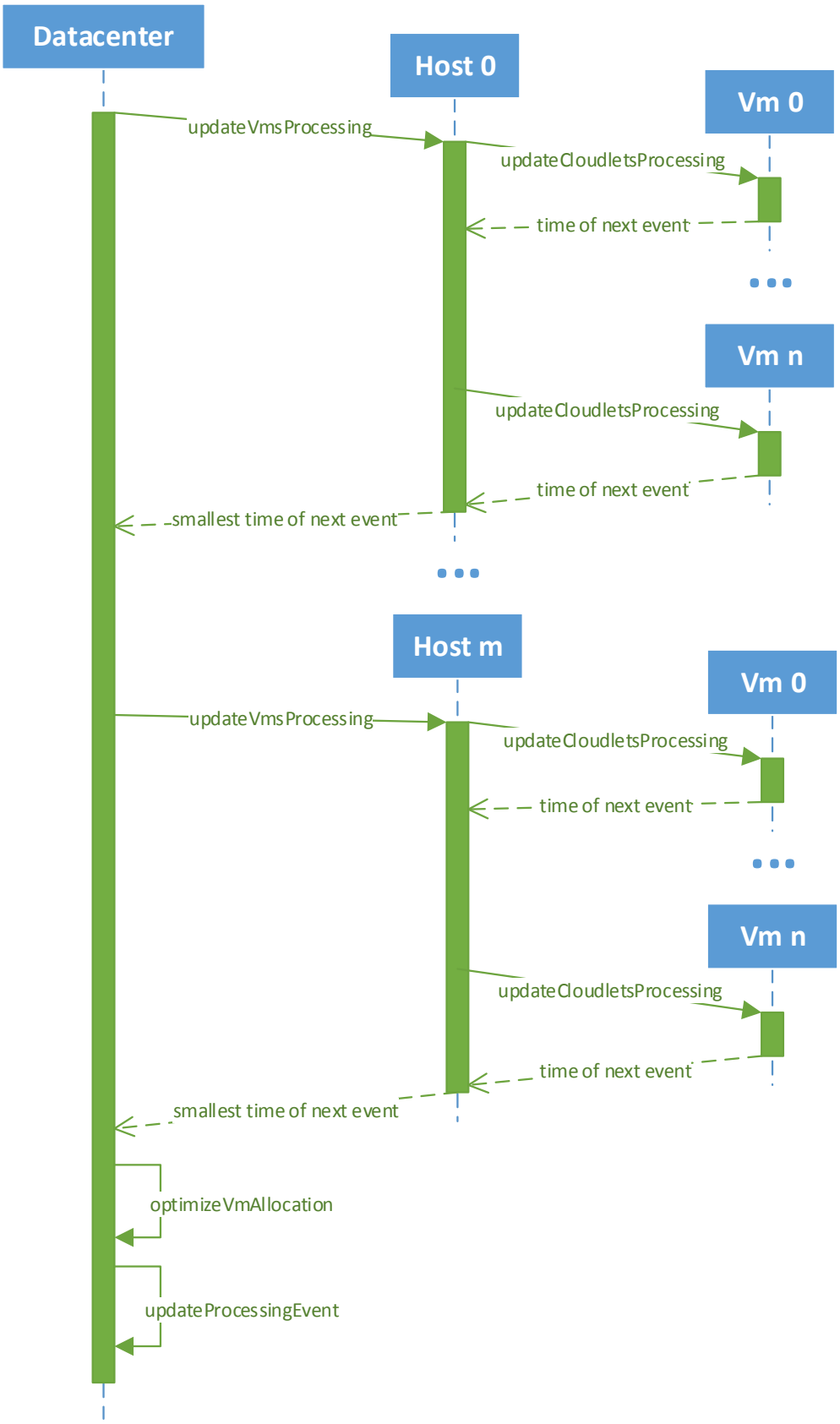

Figure 3.5: Updating processing in the data center

3.2.9 Vm Allocation Policy

As we briefly discussed in the previous section, the two main functions of a Vm Allocation Policy are i) to allocate a VM to a suitable host and ii) to optimize the allocation of VMs to hosts over the entire data center.

Finding a Host for a VM

Similarly to the Power Aware Best Fit Decreasing algorithm (see Algorithm 2.1) we choose the host that can satisfy the VM's current resource demands (e.g. the host has enough free MIPS to satisfy the VMs current utilization level), can potentially satisfy the VM's maximum resource demands (e.g. the maximum IOPS of the host are greater than the maximum IOPS of the VM) and produces the least increase in power consumption after the migration (Flowchart in Figure 3.6).

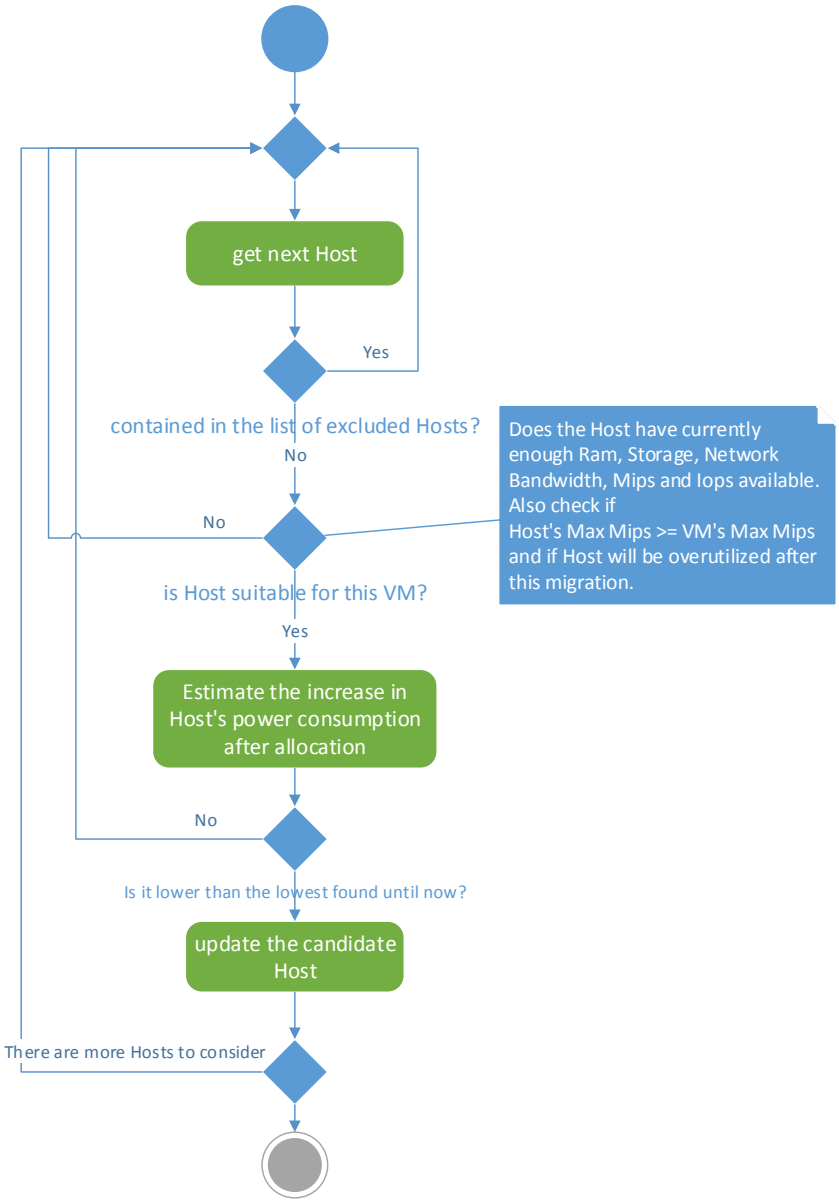

Figure 3.6: Flowchart for finding the most suitable host for a VM

Optimizing Allocation

As we described in Section 2.1.7 optimizing the allocation of VMs to hosts can be split to the following 4 steps:

Detect overutilized hosts:

We detect overutilization of a single resource using the heuristics described in Section 2.2.1. After detecting which hosts overutilize MIPS and which hosts overutilize IOPS we split all the hosts in the following four categories:

1. Hosts that overutilize only MIPS
2. Hosts that overutilize only IOPS
3. Hosts that overutilize both MIPS and IOPS
4. Hosts that are not overutilized.

Select VMs to migrate from overutilized hosts

Selecting VMs to migrate from overutilized hosts has three steps that correspond to the three different categories of overutilized hosts (see Flowchart in Figure 3.7).

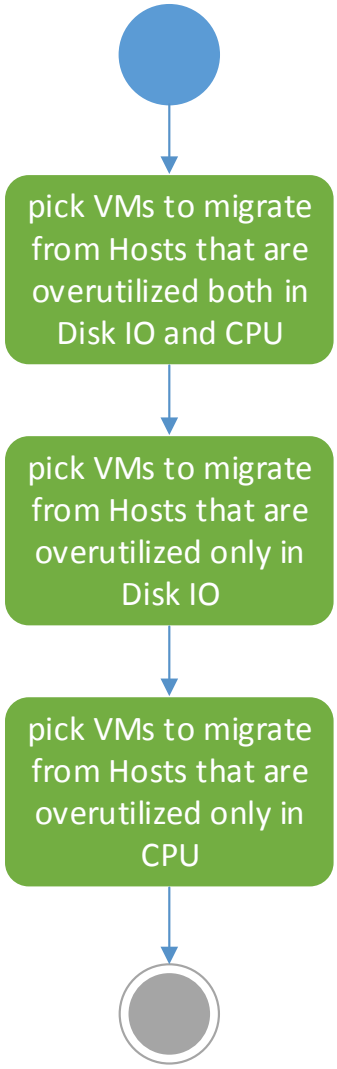

Figure 3.7: Flowchart for picking VMs to migrate from overutilized hosts

1. For each host that overutilizes only MIPS, we repeatedly use the specified VM selection Policy (Section 3.2.10) for MIPS utilization, to pick a single VM to migrate until the host is no longer overutilized (see Flowchart in Figure 3.8).
2. For hosts that overutilize only IOPS the process is similar with hosts that overutilized only MIPS. The only difference is that we use the VM selection policy specified for IOPS utilization (see Flowchart in Figure 3.8).

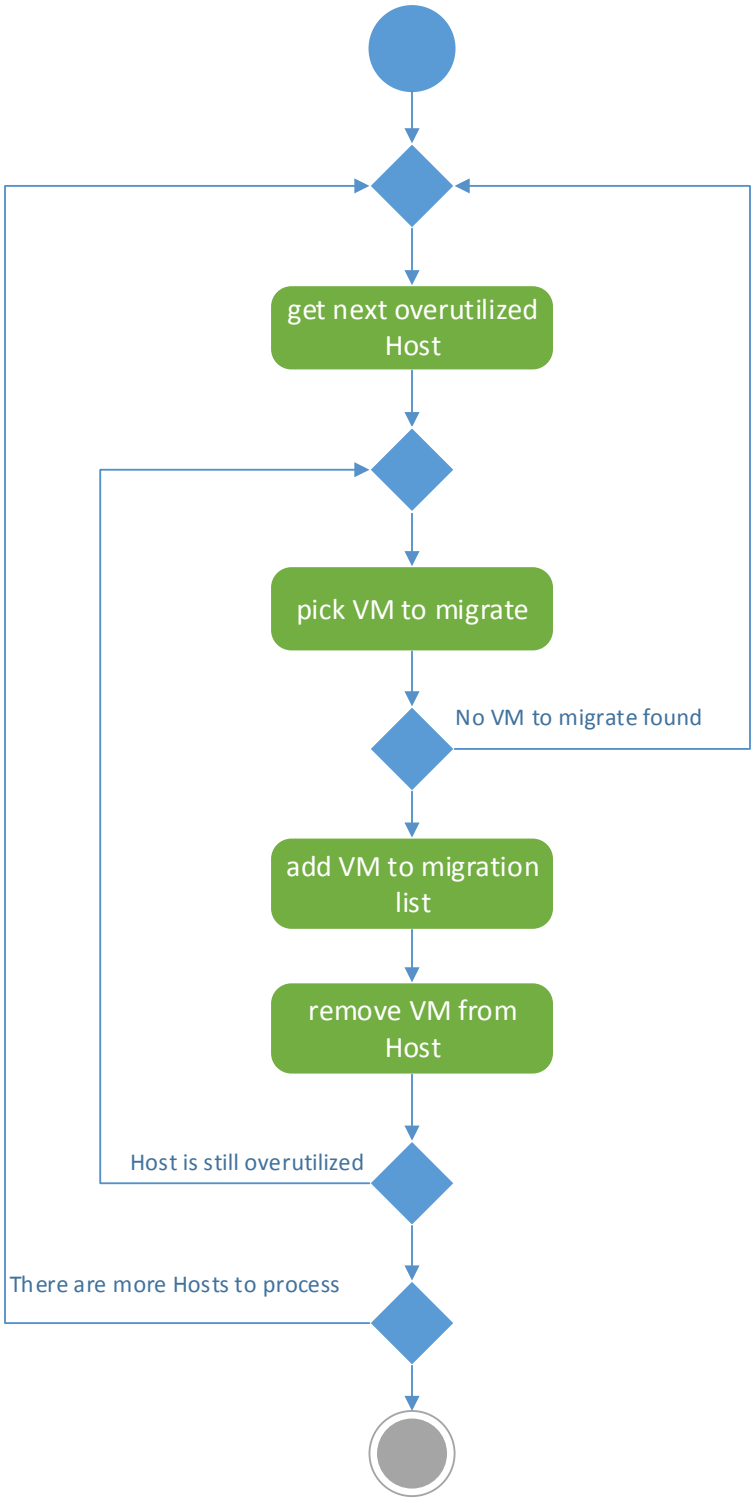

Figure 3.8: Flowchart for picking VMs to migrate from hosts that overutilize either only IOPS or only MIPS

3. For hosts that overutilize both MIPS and IOPS depending on which resource we have specified as more important we follow one of the following procedures (see Flowchart in Figure 3.9):

   MIPS utilization is more important:
   We pick VMs to migrate from hosts based only on MIPS utilization. After that, we iterate through the list of overutilized hosts and remove those that due to the removed VMs are not overutilized anymore. Finally, we pick VMs to migrate from the remaining hosts based only on IOPS utilization.

   IOPS utilization is more important:
   We pick VMs to migrate from hosts based only on IOPS utilization. After that, we iterate through the list of overutilized hosts and remove those that due to the removed VMs are not overutilized anymore. Finally, we pick VMs to migrate from the remaining hosts based only on MIPS utilization.

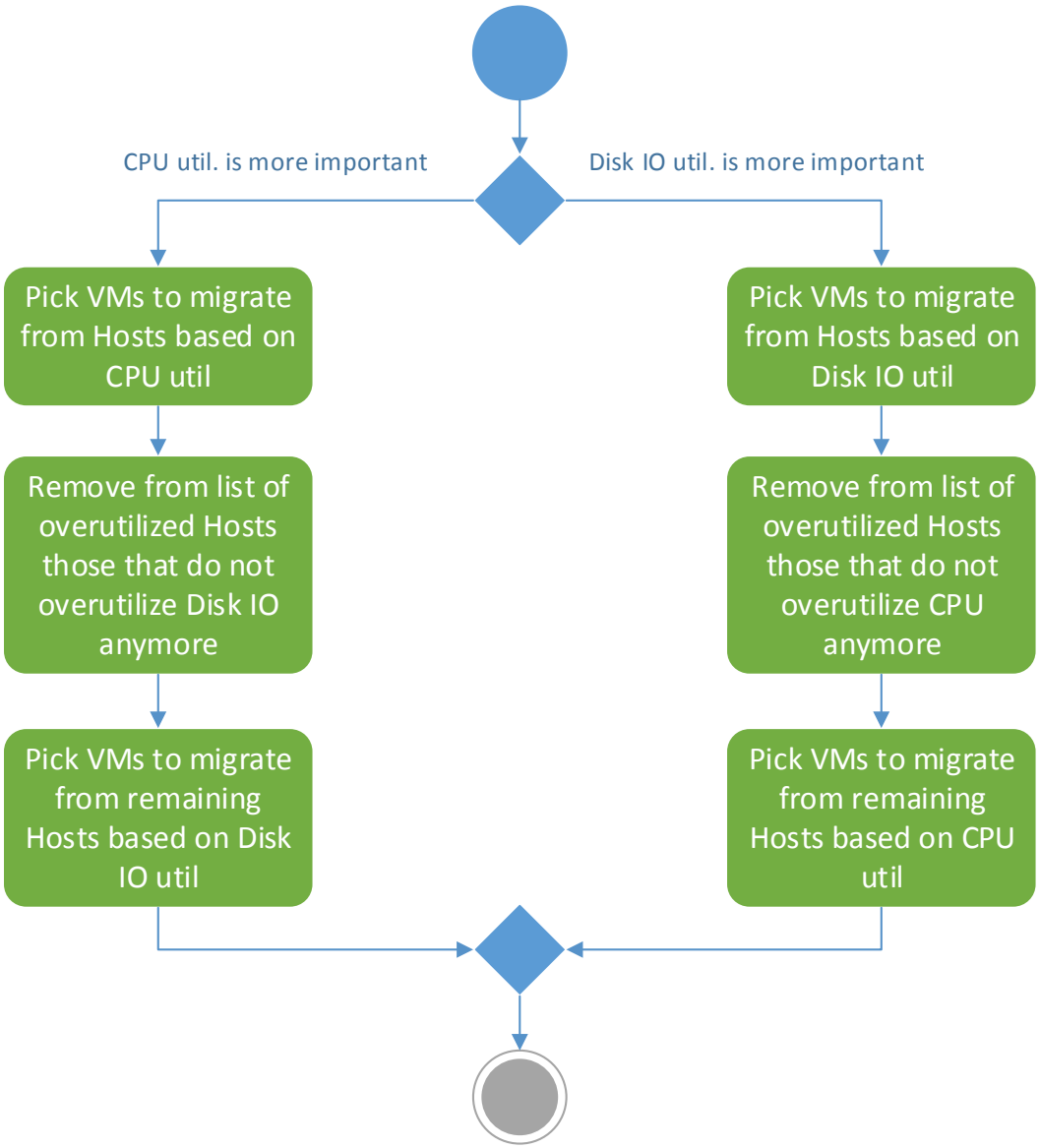

Figure 3.9: Flowchart for picking VMs to migrate from hosts that overutilized both MIPS and IOPS

Find a new placement for the VMs coming from overutilized hosts

From the previous step we receive i) a list containing VMs marked for migration because their host overutilized MIPS and ii) a list containing VMs marked for migration because their host overutilized IOPS. Depending on which resource we have specified as more important we follow one of the following procedures (see Flowchart in Figure 3.10):

MIPS utilization is more important:

We use the Power Aware Best Fit Decreasing algorithm (see Algorithm 2.1) to find new hosts for the VMs in the first list and then we use the same algorithm to find new hosts for the VMs in the second list.

IOPS utilization is more important:

We use the Power Aware Best Fit Decreasing algorithm (see Algorithm 2.1) to find new hosts for the VMs in the second list and then we use the same algorithm to find new hosts for the VMs in the first list.

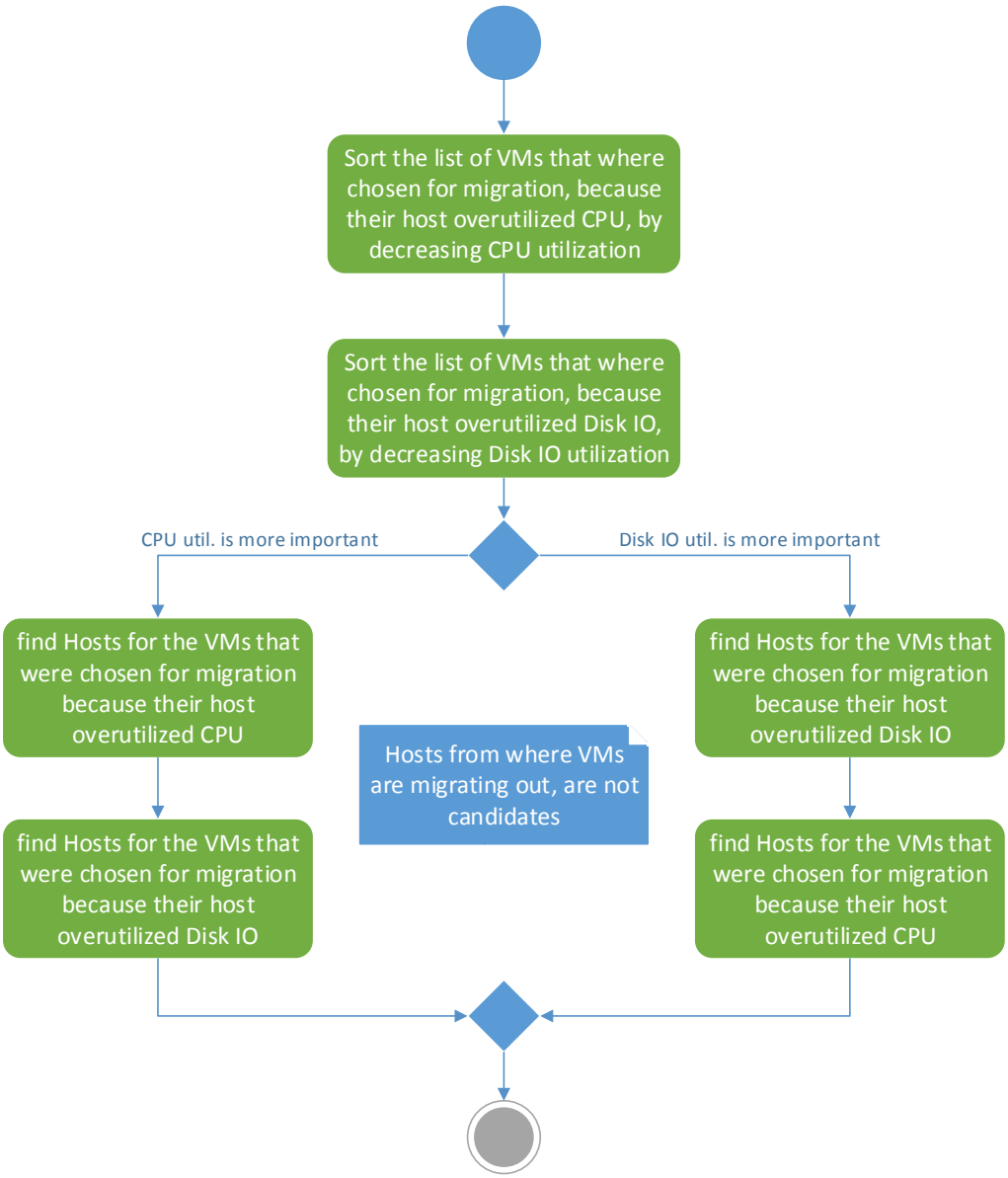

Figure 3.10: Flowchart for finding new placement for VMs marked for migration

Detect underutilized hosts and find a new placement for all their VMs

To handle underutilized hosts we use the simple approach described in Section 2.2.4 with one modification. Depending on which resource we have specified as more important we find each time the host with the least utilization in MIPS or IOPS respectively (see Flowchart in Figure 3.11).

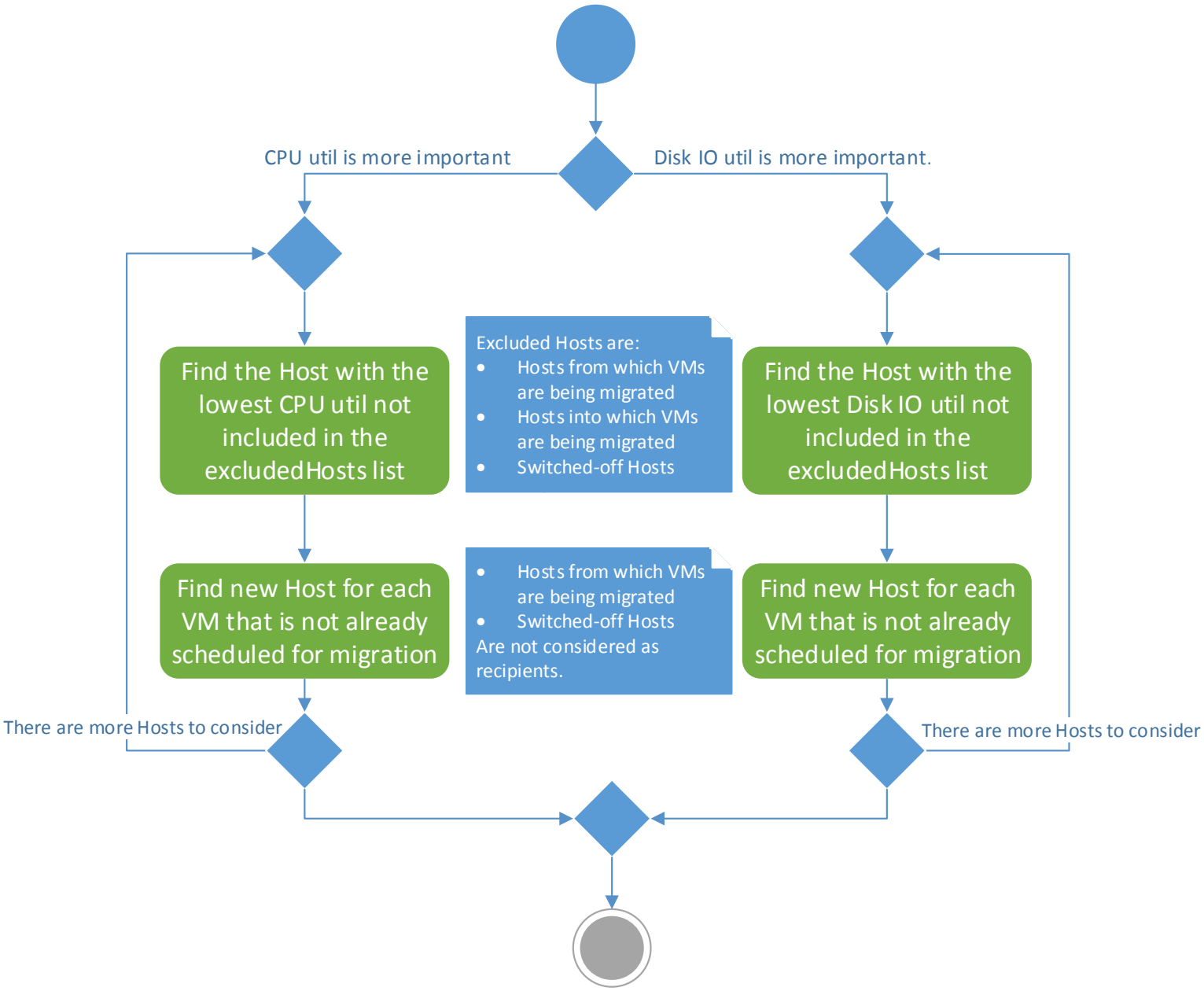

Figure 3.11: Flowchart for detecting underutilized hosts and finding a new placement for their VMs

3.2.10 VM Selection Policy

The subclasses of vmSelectionPolicy implement the various heuristics that we described in Section 2.2.2. Their only function is to select which VM should be migrated next.

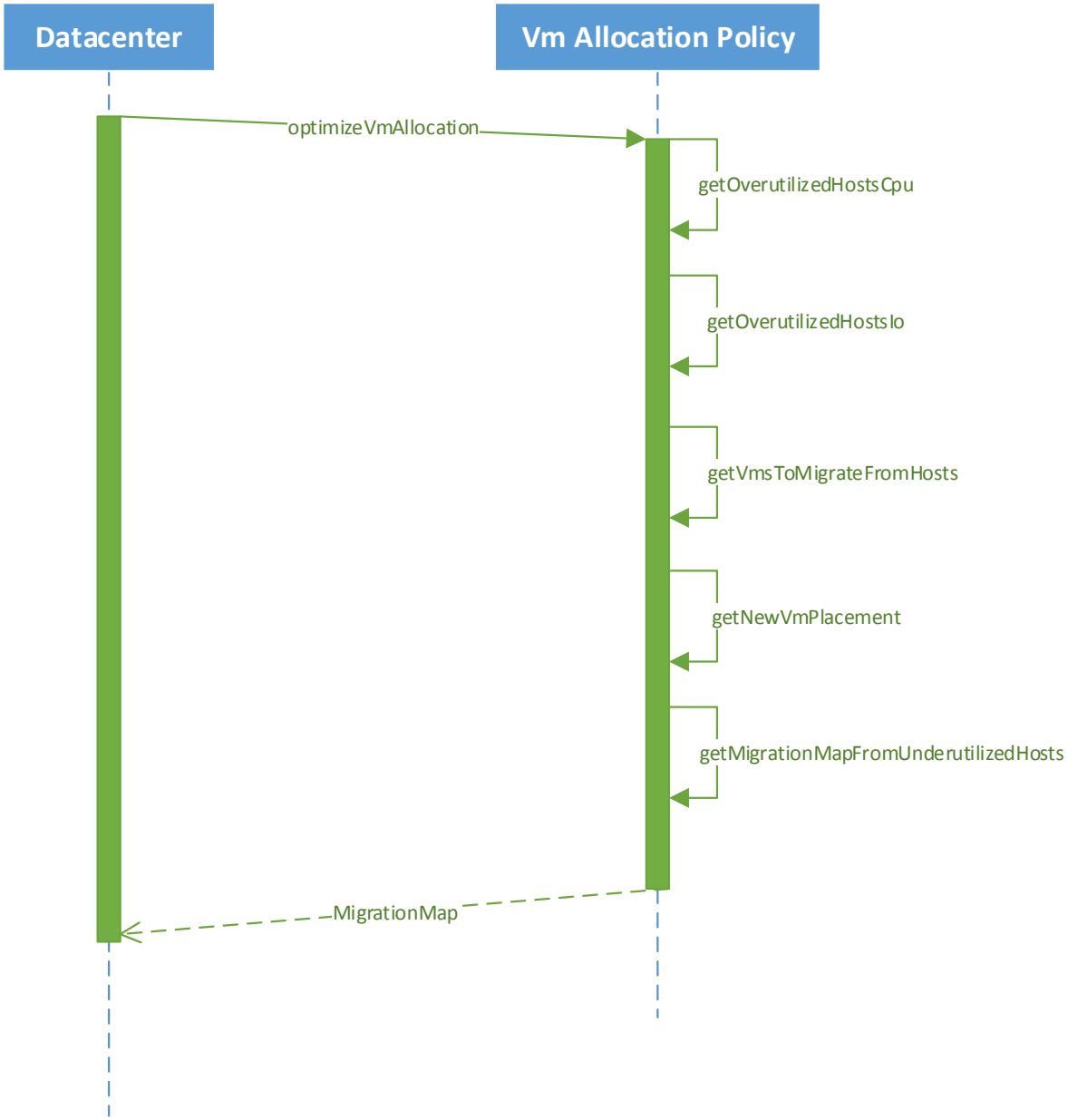

Figure 3.12: Flowchart for optimizing the allocation of VMs

Chapter 4

Simulation

In the following sections we will present the attributes of the simulated environment, the characteristics of the workloads, the metrics used and discuss the results and our conclusions.

4.1 Metrics

4.1.1 SLA Violation Metrics

In cloud computing as in every other business, customer satisfaction is crucial. Consequently, meeting the QoS requirements, as formalized in SLAs between customers and providers, is of utmost importance. To understand this importance, one has only to take a look at one such SLA. For instance Amazon's SLA for the EC2 service provides the following service commitments for monthly uptime percentage and penalties:

- If the monthly uptime percentage is less than 99.95% but greater than 99.9%, Amazon will credit your account with 10% of the total charges paid.
- If the monthly uptime percentage is less than 99.9%, Amazon will credit your account with 30% of the total charges paid.

Since service commitments vary greatly for different providers and applications, we will be using generic application independent metrics proposed by A. Beloglazov et al.[2]. For our simulations, we will consider that the QoS requirements are fulfilled when the performance requested by the application at any time is delivered. As we described in the previous chapter (3.2.3), the application performance requirements are bounded by the characteristics of the VM on which it is deployed.

Since application residing in a fully utilized host are almost certainly experiencing SLA violations[1], one of the metrics we will be using will be the percentage of time, during which active hosts have experienced utilization of 100%. We call this metric SLA violation Time per Active Host (SLATAH).

$$SLATAH = \frac{1}{N}\sum_{i=1}^{N}\frac{T_{s_i}}{T_{a_i}} \tag{4.1}$$

[1] An exception would be if the performance requested by all applications matches exactly the resources of the host.

where N is the number of hosts, T_{s_i} is the total time host i experienced 100% utilization and T_{a_i} is the total time host i was in an active state.

Another source of SLA violation is the performance degradation experienced by applications in VMs that are being migrated[2]. As a result our second metric of SLA violation will be the percentage of requested performance that was not provided to VMs while being migrated. We call this metric Performance Degradation due to Migration (PDM).

$$PDM = \frac{1}{M} \sum_{j=1}^{M} \frac{C_{d_j}}{C_{r_j}} \tag{4.2}$$

where M is the number of VMs, C_{d_j} is the performance requested by VM j but not delivered while being migrated and C_{r_j} is the total performance requested by VM j.

Since these two metrics characterize two different sources of SLA violation we will combine them in a single metric called SLA Violation (SLAV).

$$SLAV = PDM * SLATAH \tag{4.3}$$

4.1.2 Performance Metrics

The goal of every power aware resource management system is to minimize the combined costs of energy consumption and SLA violations. However, these two parameters, energy consumption and SLA violations, are inversely correlated and as such we can not use only one of those parameters to evaluate our approach. Consequently we will be using the combined metric, proposed by A. Beloglazov et al., Energy and SLA Violations (ESV).

$$ESV = Energy * SLAV \tag{4.4}$$

4.2 Workloads

As we have already mentioned in previous chapters, we used two separate workload sets to evaluate our approach. Both sets contain data from PlanetLab machines that were provided as part of the CoMon project[3].

The first set of workloads (Table 4.1) is the same set used by A. Beloglazov et al. and besides allowing us to draw some interesting results, it is also used as a sanity check for the extensions we implemented. Unfortunately, it contains no data about storage utilization and we were unable to retrieve them due to a hardware failure experienced by the CoMon project that led to its' shutdown[4].

[2] For our simulations we model the performance degradation that a VM experience during a migration as 10% of the total performance requested

[3] CoMon provided monitoring statistics for PlanetLab: http://comon.cs.princeton.edu/

[4] http://lists.planet-lab.org/pipermail/users/2012-November/004159.html

This set includes workloads for 10 random days between March and April of 2011. The number of VMs range approximately from 900 to 1500. As we can see the mean of CPU utilization is about 12% and the median is less than 10% leaving a lot of room for consolidating CPU usage.

Date	Number of VMs	Mean	St. dev.	Quartile 1	Median	Quartile 3
03/03/2011	1052	12.31%	17.09%	2%	6%	15%
06/03/2011	898	11.44%	16.83%	2%	5%	13%
09/03/2011	1061	10.70%	15.57%	2%	4%	13%
22/03/2011	1516	9.26%	12.78%	2%	5%	12%
25/03/2011	1078	10.56%	14.14%	2%	6%	14%
03/04/2011	1463	12.39%	16.55%	2%	6%	17%
09/04/2011	1358	11.12%	15.09%	2%	6%	15%
11/04/2011	1233	11.56%	15.07%	2%	6%	16%
12/04/2011	1054	11.54%	15.15%	2%	6%	16%
20/04/2011	1033	10.43%	15.21%	2%	4%	12%

Table 4.1: 1st Workload - CPU utilization

The second set of workloads (Table 4.2 and Table 4.3) was provided to us by Professor Vivek Pai, after we contacted him to inquire about the storage utilization workloads of the first set.

Compared to the first set we can see that the number of VMs is significantly lower (approximately 500). Additionally, while CPU utilization still remains at very low levels, means and medians have increased. Finally we can see that Storage utilization is also low leaving room for consolidating storage usage.

Date	Number of Vms	Mean	St. Dev.	Quartile 1	Median	Quartile 3
06/07/2009	579	15.51%	15.97%	3%	9%	25%
07/07/2009	571	14.33%	14.82%	2%	8%	22%
08/07/2009	568	16.11%	15.35%	3%	11%	26%
09/07/2009	562	17.29%	15.00%	4%	13%	27%
10/07/2009	575	22.56%	16.70%	7%	21%	33%
11/07/2009	572	21.48%	15.75%	7%	21%	31%
12/07/2009	580	18.17%	15.79%	4%	15%	29%
13/07/2009	567	16.30%	15.26%	3%	11%	26%
14/07/2009	577	19.07%	15.15%	6%	17%	29%
15/07/2009	385	12.36%	15.31%	0%	4%	22%

Table 4.2: 2nd Workload - CPU utilization

Date	Number of Vms	Mean	St. Dev.	Quartile 1	Median	Quartile 3
06/07/2009	579	15.39%	24.32%	1.54%	5.34%	16.67%
07/07/2009	571	16.65%	26.43%	1.47%	5.12%	17.53%
08/07/2009	568	17.70%	27.23%	1.52%	5.52%	19.24%
09/07/2009	562	17.11%	26.16%	1.55%	5.65%	18.91%
10/07/2009	575	17.32%	25.93%	1.73%	6.02%	19.32%
11/07/2009	572	14.51%	23.82%	1.38%	4.56%	15.32%
12/07/2009	580	15.04%	24.05%	1.47%	5.12%	16.03%
13/07/2009	567	14.98%	24.26%	1.33%	4.79%	16.01%
14/07/2009	577	15.81%	24.61%	1.43%	5.51%	17.19%
15/07/2009	385	11.25%	22.89%	0.00%	1.49%	9.52%

Table 4.3: 2nd Workload - Storage utilization

4.3 Simulated Environment

For the 1st workload we simulated a data center with 800 Hosts. Half of them are HP ProLiant ML110 G4 servers and the other half are HP ProLiant ML110 G5. Both G4 and G5 servers have 2 cores and each core has 1860 (G4) or 2660 (G5) MIPS. Their power consumption is simulated using the models described in Section 3.2.7.

To study the effect that under-provisioning and over-provisioning of the storage system can have on a data center we perform 4 simulations with different type of storage systems:

- Slow storage each providing 30000 IOPS
- Fast storage each providing 40000 IOPS
- Very fast storage each providing 100000 IOPS
- Half the hosts have slow storage (30000 IOPS) and the other half have fast (45000 IOPS) storage

The IOPS ratings mentioned above may initially seem excessive but they are a side-effect of the utilization model we used to extrapolate storage utilization from CPU utilization. Since for each MIPS we need 20 IOPS, a G4 server that experiences 100% utilization of its' CPU would need more than 70000 IOPS.

For the 2nd workload we simulated a data center with 400 Hosts. Once again, half of them are HP ProLiant ML110 G4 servers with a storage system rated at 3000 IOPS and the other half are HP ProLiant ML110 G5 with a storage system rated at 4500 IOPS.

4.4 Results

For each simulation we used 3 different dynamic VM placement approaches. Two of them used both Storage and CPU utilization to decide about the placement, but one gives a slight priority to Storage utilization (Disk util. First) and the other to CPU utilization (CPU util. First) (see Chapter 3.2.9). The last approach uses only CPU utilization (CPU util. only) and is the approach proposed by A. Beloglazov et al.

4.4.1 1st Workload

For the 1st workload we used the safety parameter values suggested by A. Beloglazov et al.: THR-0.8, IQR-1.5, MAD-2.5, LRR-1.2, LR-1.2.

ESV

It is immediately obvious that for all 4 simulations, selecting to migrate the VMs that will spend the least time in migration is the best VM selection policy. This is due to the fact that a shorter migration duration means less performance degradation (less PDM) due to migration and if the host happens to be overloaded, a faster recovery (less SLATAH). Additionally, we see that there is almost no difference between the two approaches that use both CPU and Storage utilization to decide for the placement of VMs.

When the storage system for the data center is under-provisioned (see Figure 4.2) then CPU util. only achieves less ESV for almost every combination of policies and the ESV for the best combination (lr-mmt) for Disk util. first and CPU util. first is approximately 50% greater than the ESV for the best combination (thr-mmt) for CPU util. only.

On the other hand, when the storage system for the data center is over-provisioned (see Figure 4.1) then CPU util. first and Disk util. first give better results for almost every combination of policies. However, the difference between the best combination for CPU util. only (thr-mmt) and the best combination for CPU util. first and Disk util. first is insignificant.

From the previous two paragraphs it becomes apparent that when we can not change the effect that the storage system has on the execution of applications by changing the allocation of resources, it is better to use an approach that uses only CPU utilization to decide about the placement. For example, if regardless of the allocation of VMs to hosts the storage system is overutilized then an allocation system that migrates VMs in response to storage overutilization will perform more migrations than a system that migrates VMs only in responce to CPU utilization (see Section 4.4.1), increasing PDM and consequently ESV.

When the speed of the storage system is more suitable for the workload, the results are very different. In the case where each host (see Figure 4.3) has fast storage then for almost every combination CPU util. first and Disk util. first give better results than CPU util. only. Additionally the ESV of the best combination fo CPU util. only is 50% greater. In the case where half the hosts have slow storage and half fast storage (see Figure 4.4), Disk util. first and CPU util. first outperform CPU util. only for every combination and the best case for CPU util. only is approximately 100% greater.

Another important observation is that the heterogeneity in host resources allows us to respond better to the heterogeneity of the application workloads. That becomes evident when we consider that despite providing less total IOPS ($800 * 40000$ versus $400 * 30000 + 400 * 45000$) we achieve 28% better results when half the hosts have fast storage (45000 IOPS) and the other half slow (30000 IOPS) compared to when all hosts have fast storage (40000 IOPS).

Finally, we notice that, with the exception of very fast storage, generally the Local Regression method for Host Overload Detection provides better results than the other policies.

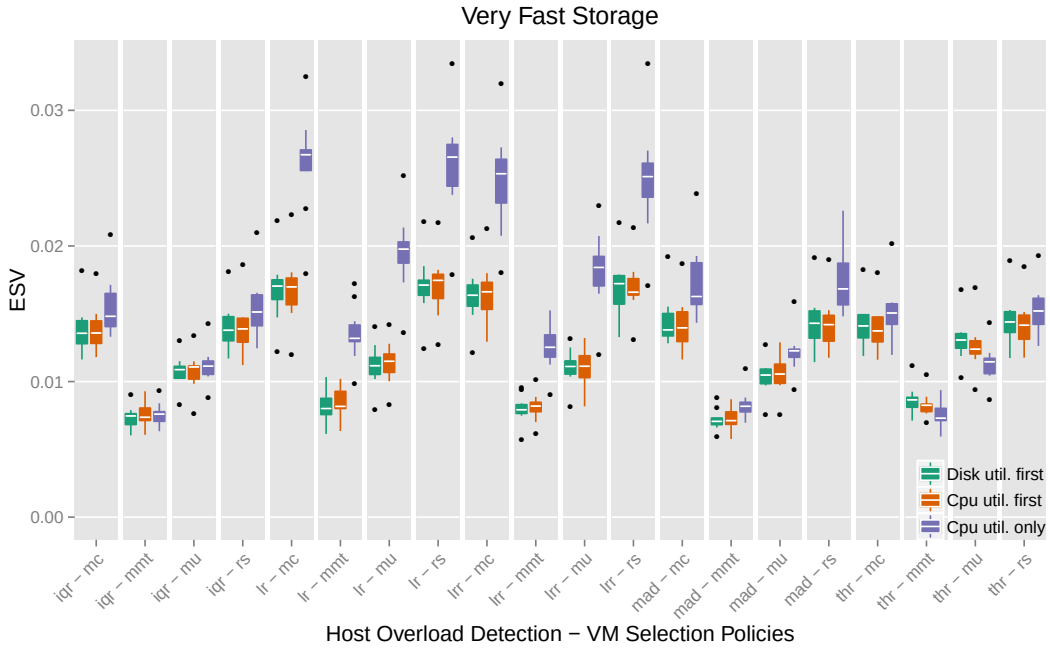

Figure 4.1: ESV - Very fast storage

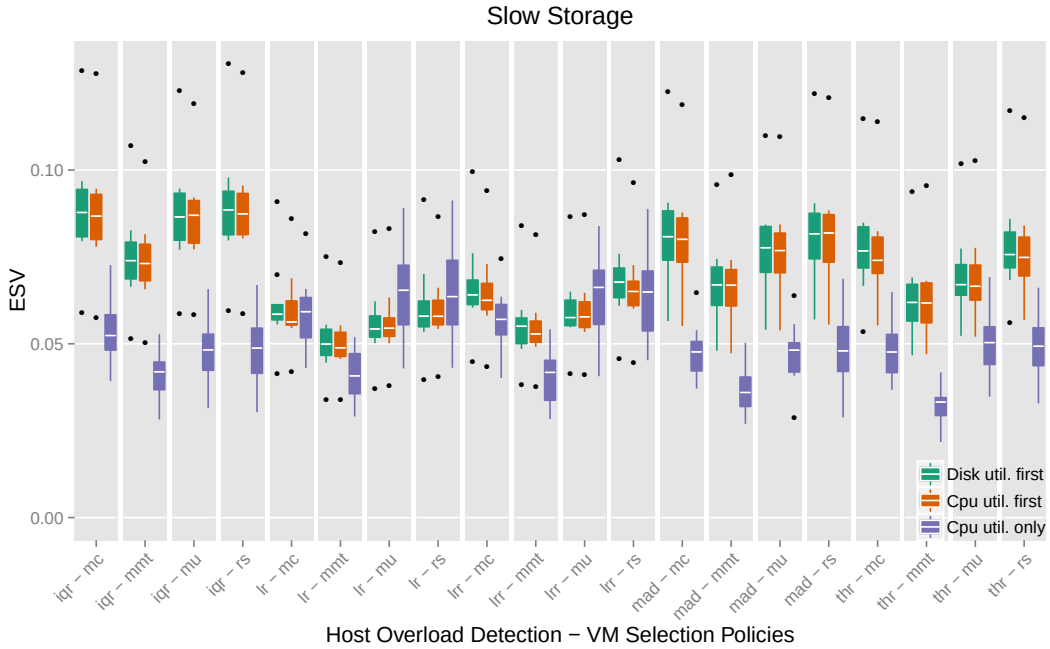

Figure 4.2: ESV - Slow storage

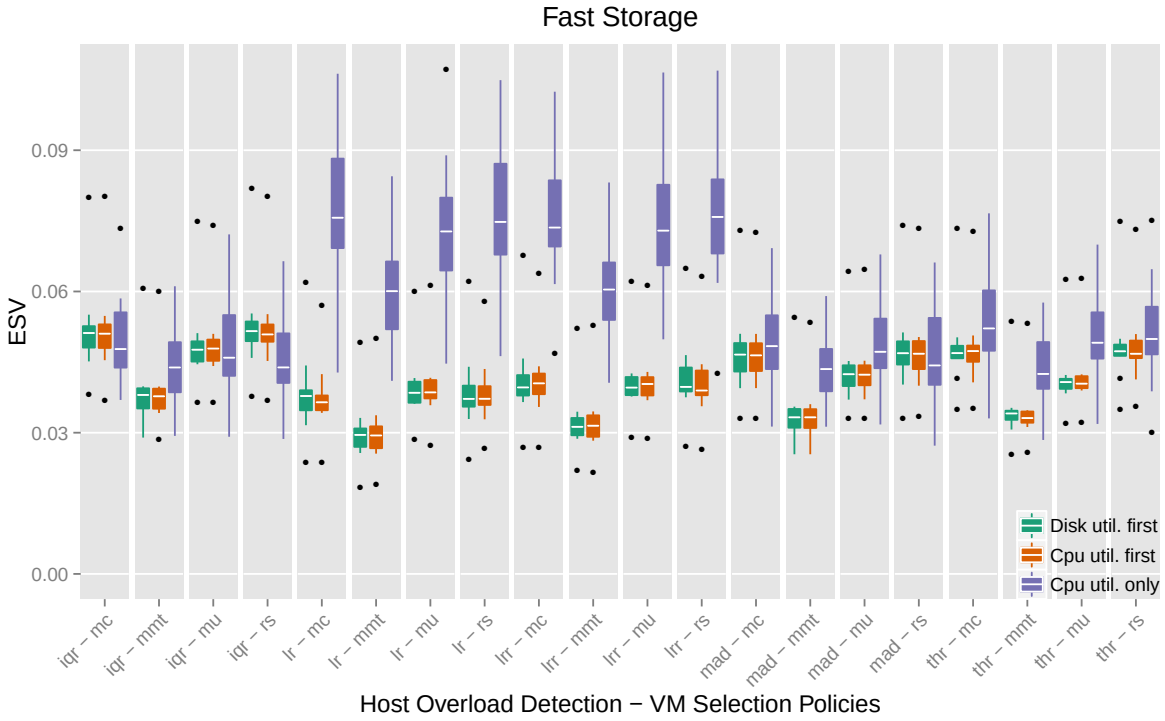

Figure 4.3: ESV - Fast storage

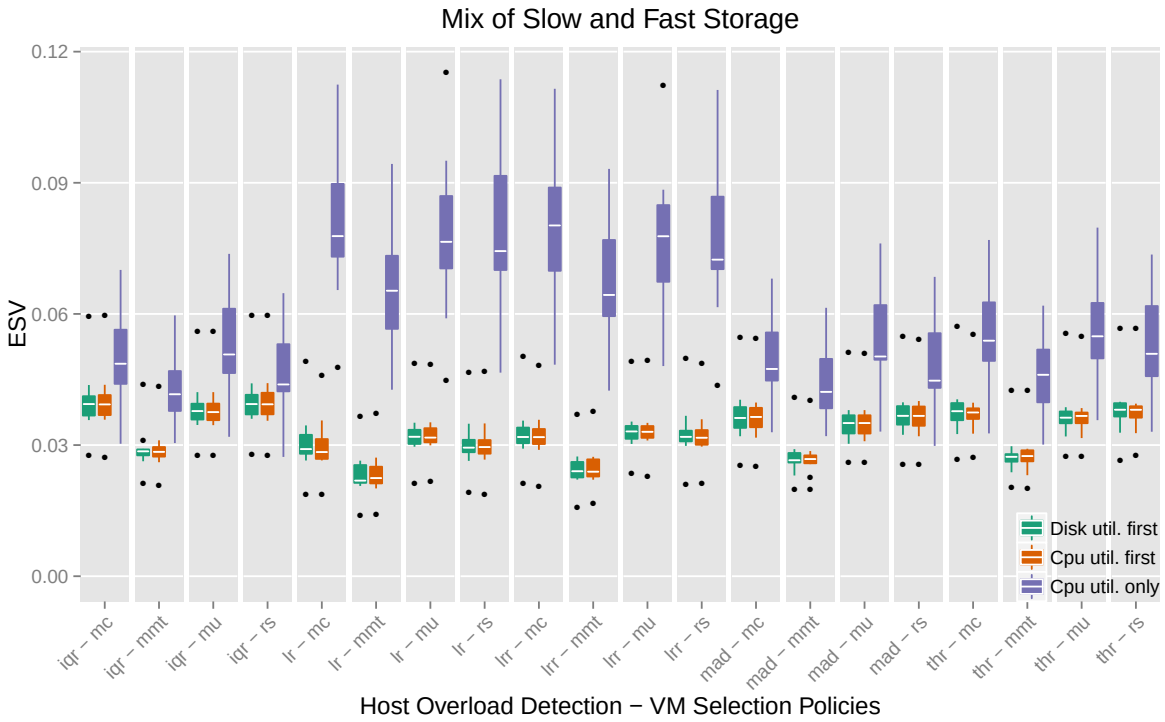

Figure 4.4: ESV - Mix of slow and fast storage

Energy

It is apparent that generally Disk util. first and CPU util. first are trading a bigger energy consumption for less SLA violations. Despite the increase in power consumption for our proposed approaches, the consumption is still far less than that of a Non-Power aware system (2419.2 kWh). In addition to that, we can see that by providing more and more performance to the storage system we achieve significantly lower power consumption, more than 100% better in some cases. This can be explained by the fact that when applications are not bound by the performance of the storage system they can finish their execution much faster and possibly by the fact that we have not included in our model the power consumption of the storage system. Finally, although we do not observe large differences in power consumption between different combination of policies, we can safely say that Local Regression and Local Regression Robust provide better power consumption.

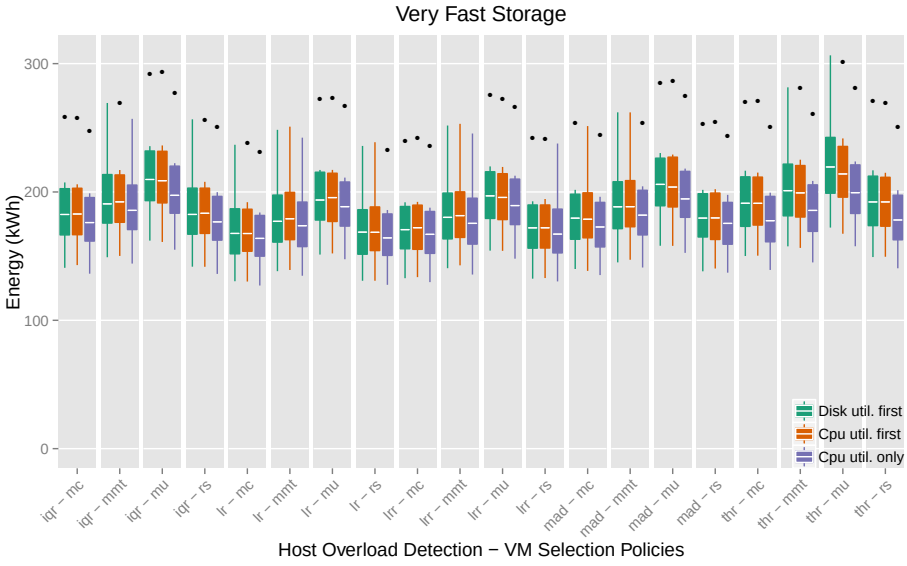

Figure 4.5: Energy - Very fast storage

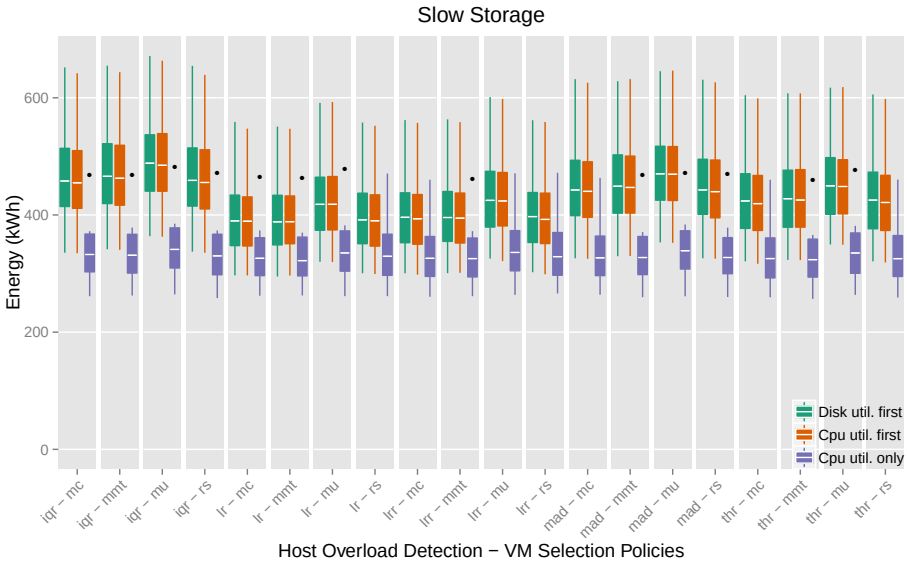

Figure 4.6: Energy - Slow storage

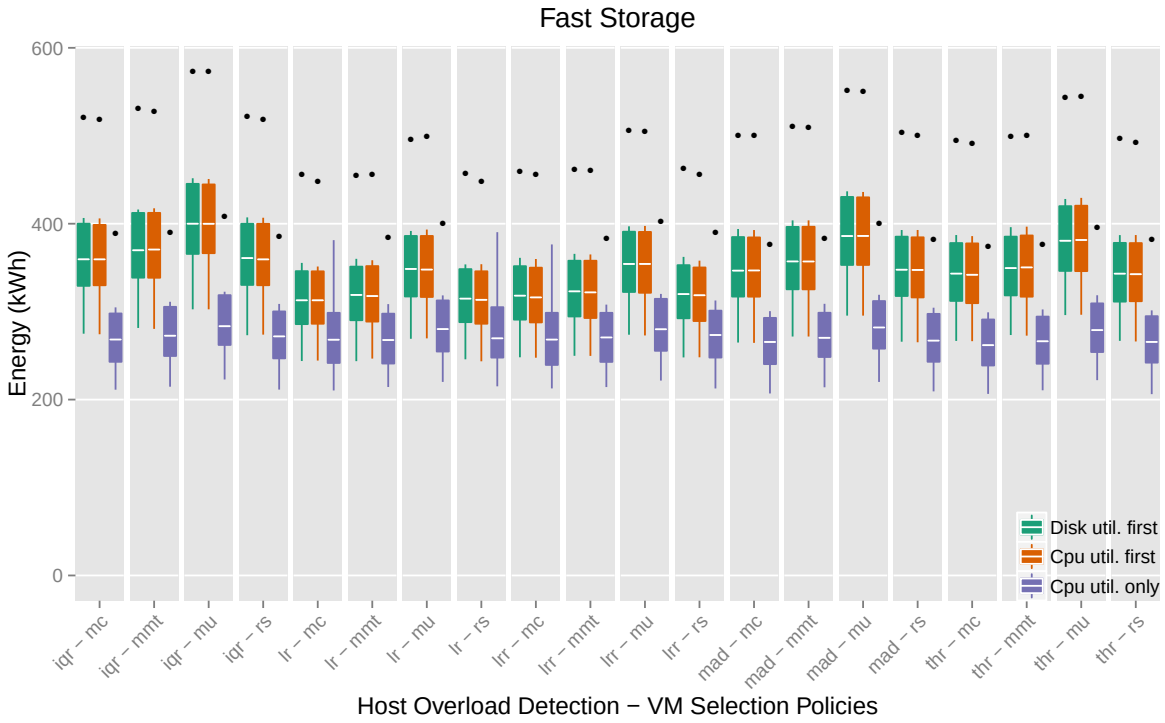

Figure 4.7: Energy - Fast storage

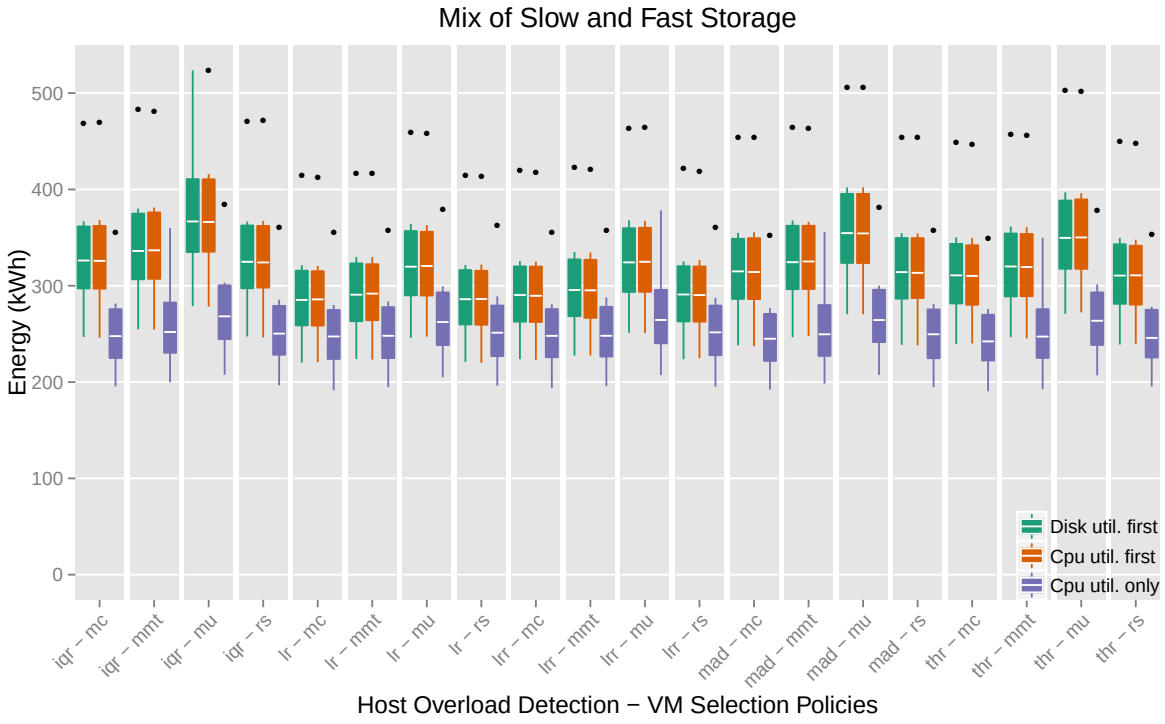

Figure 4.8: Energy - Mix of slow and fast storage

Migrations

With the exception of Very Fast Storage, Disk util. first and CPU util. first generally more migrations than CPU util. only. This reaches extreme levels when we use Slow Storage when our approaches perform approximately twice as much migrations and it becomes perhaps the greatest factor in increasing the achieved ESV values. Once again providing a better storage system results in less migrations and providing heterogeneity in the storage system allows the allocation system to respond better to heterogeneity in application workloads (see Figure 4.12 in comparison to Figure 4.11).

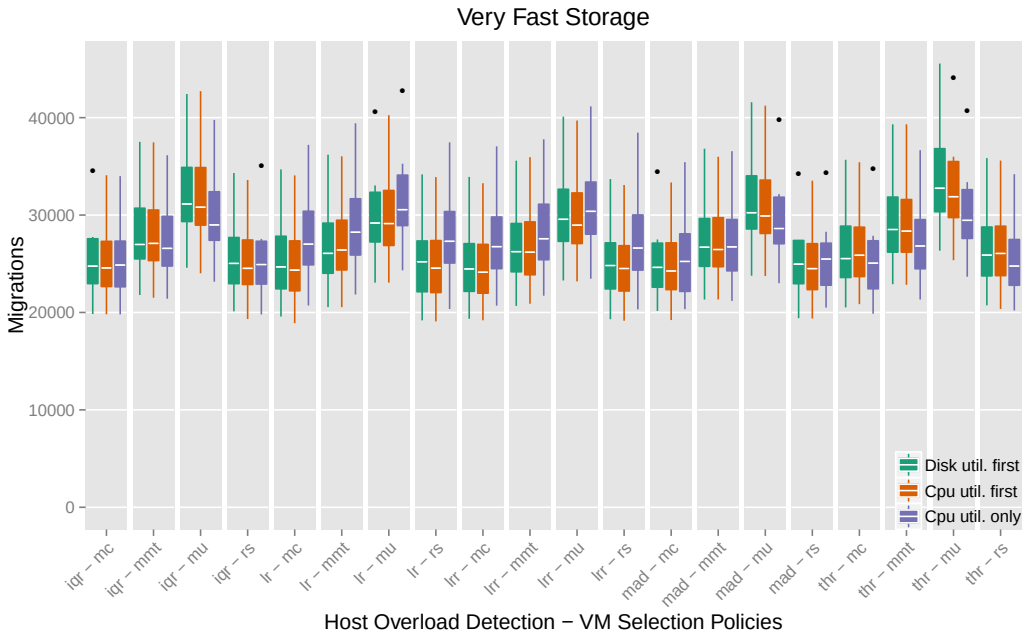

Figure 4.9: Migrations - Very fast storage

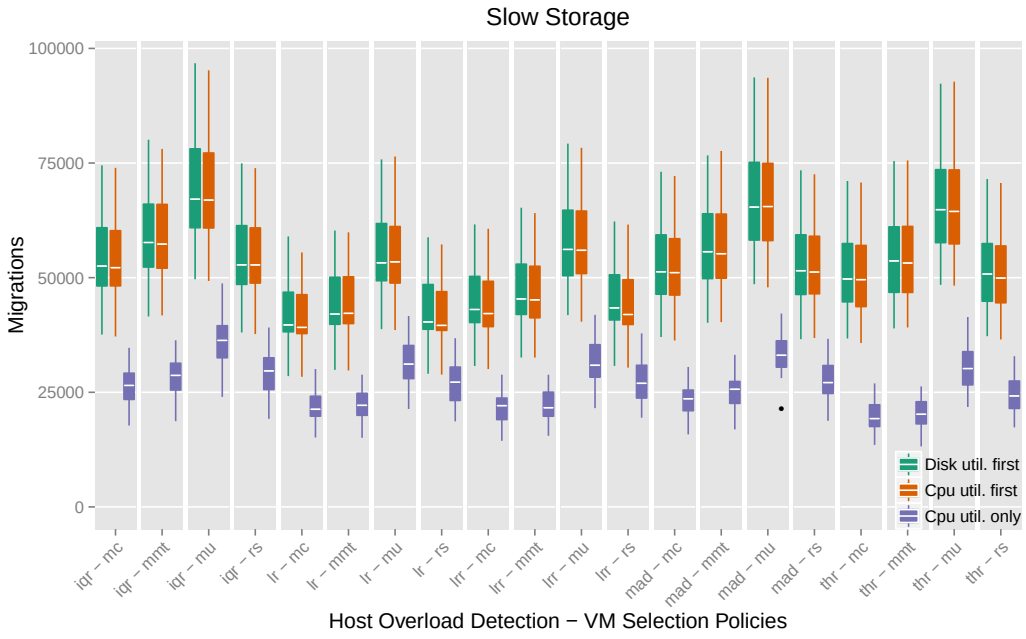

Figure 4.10: Migrations - Slow storage

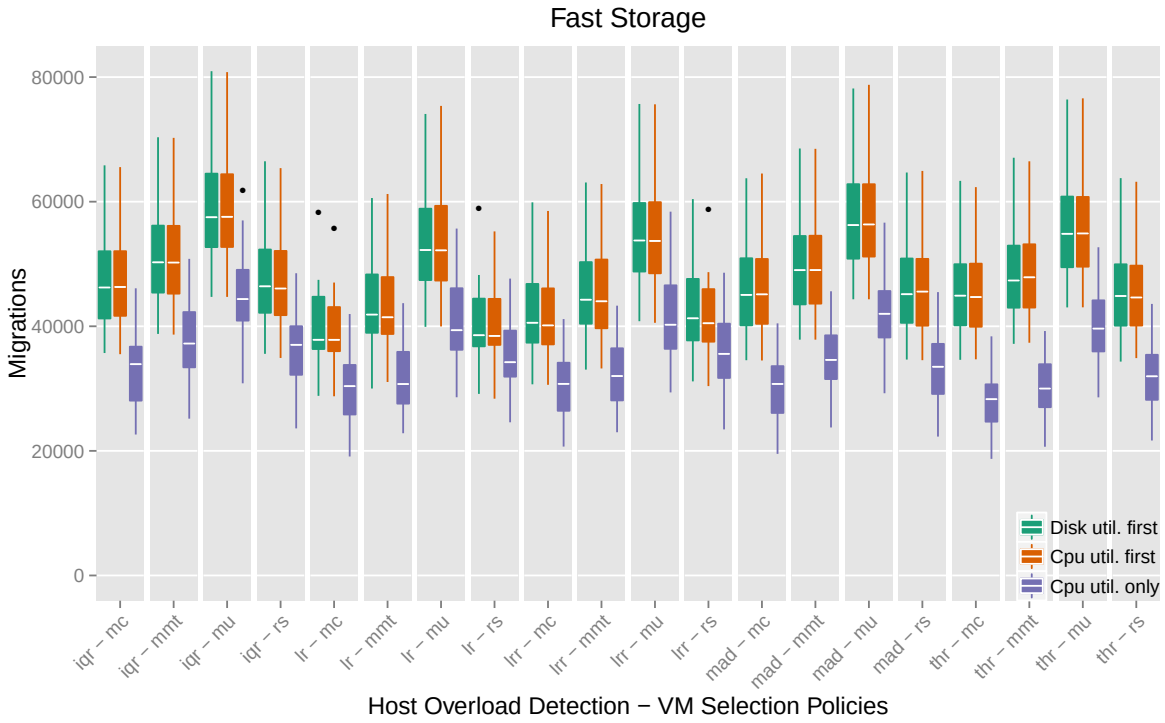

Figure 4.11: Migrations - Fast storage

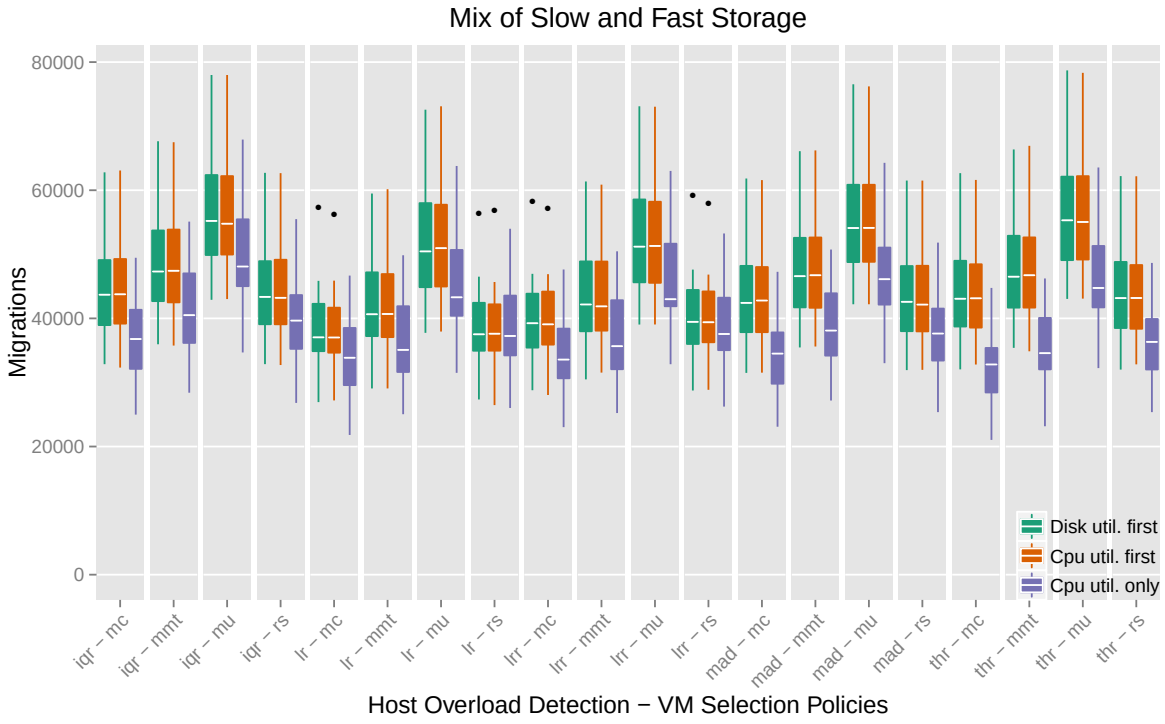

Figure 4.12: Migrations - Mix of slow and fast storage

SLATAH

SLATAH is where Disk util. first and CPU util. first far outperform CPU util.only, with the exception of an overprovisioned storage system (see Figure 4.13). Once again heterogeneity in the storage system provides better results and the Minimum Migration Time is the best policy for selecting Vms for migration. It is obvious that using both Storage and CPU utilization to determine whether a host is overloaded provides far better results.

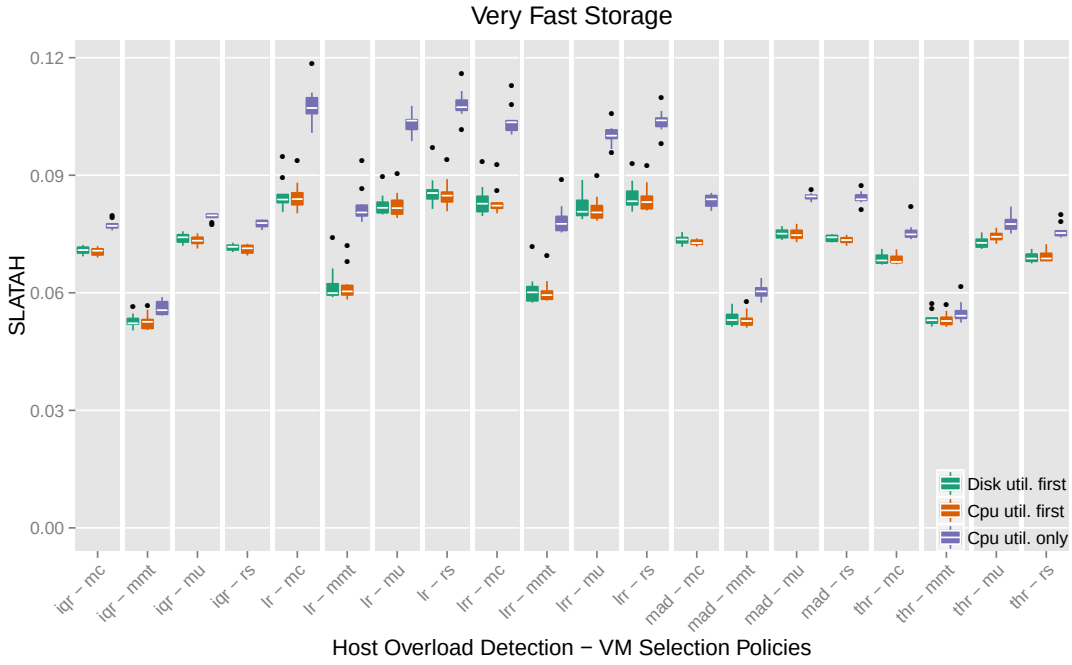

Figure 4.13: SLATAH - Very fast storage

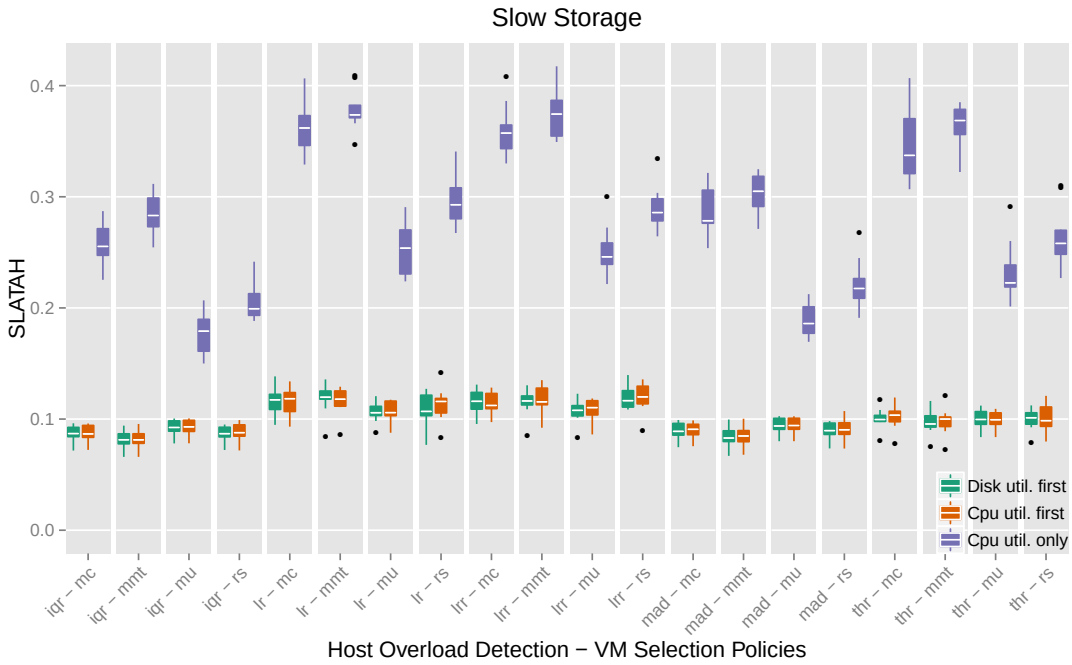

Figure 4.14: SLATAH - Slow storage

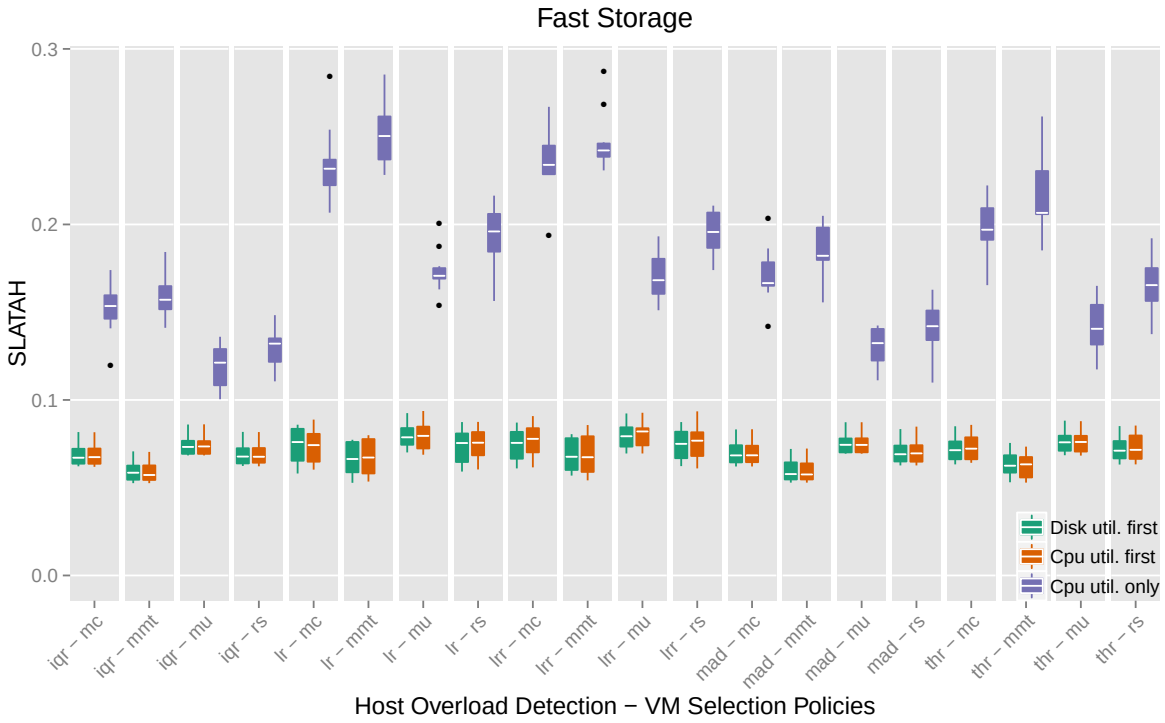

Figure 4.15: SLATAH - Fast storage

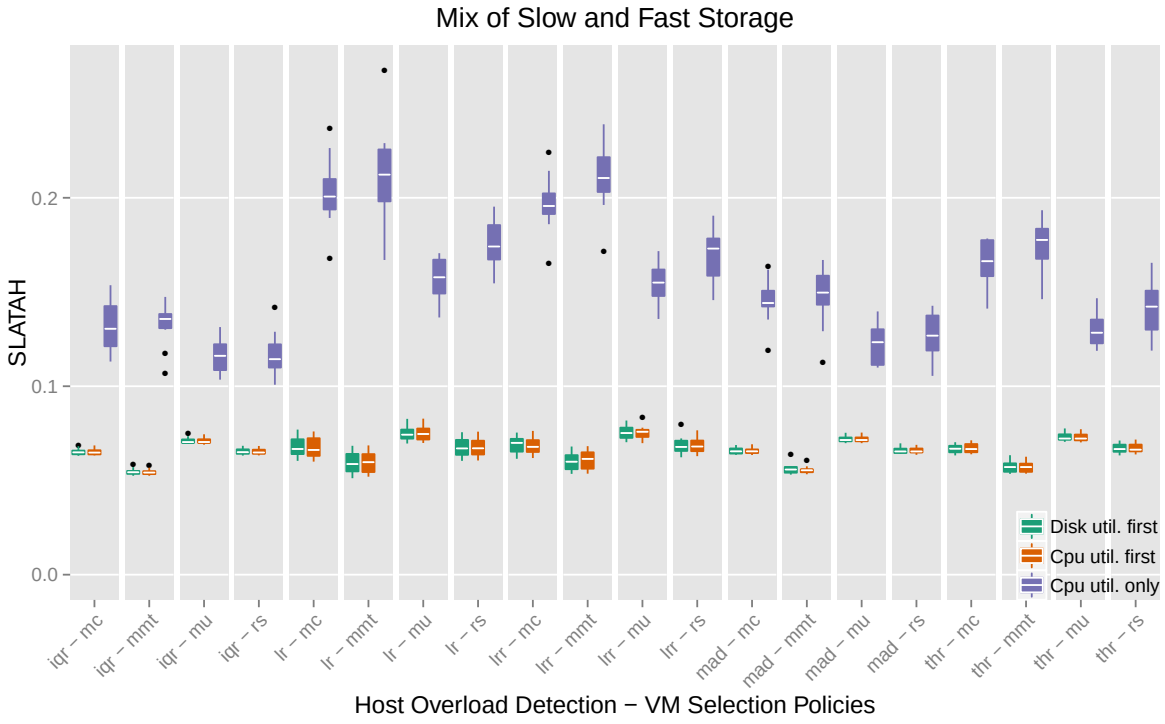

Figure 4.16: SLATAH - Mix of slow and fast storage

PDM

Since the greatest factor for PDM is the number of migrations, one would expect that the results for PDM should reflect those for the number of migrations. Although there small variations, that is the case.

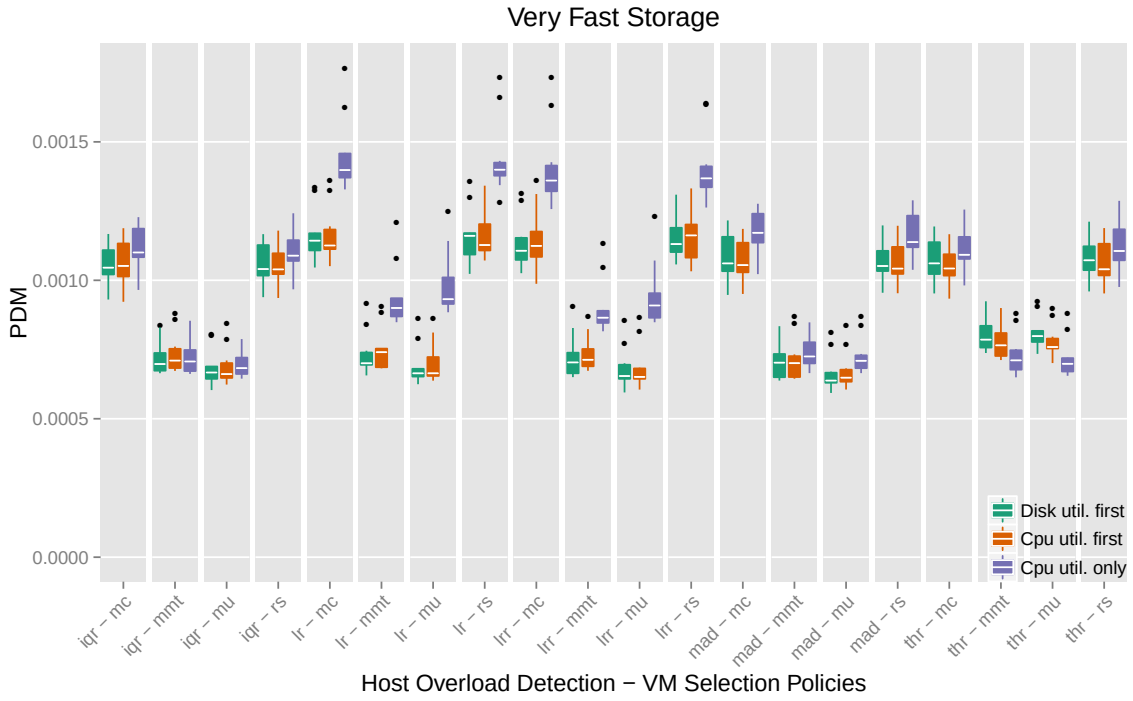

Figure 4.17: PDM - Very fast storage

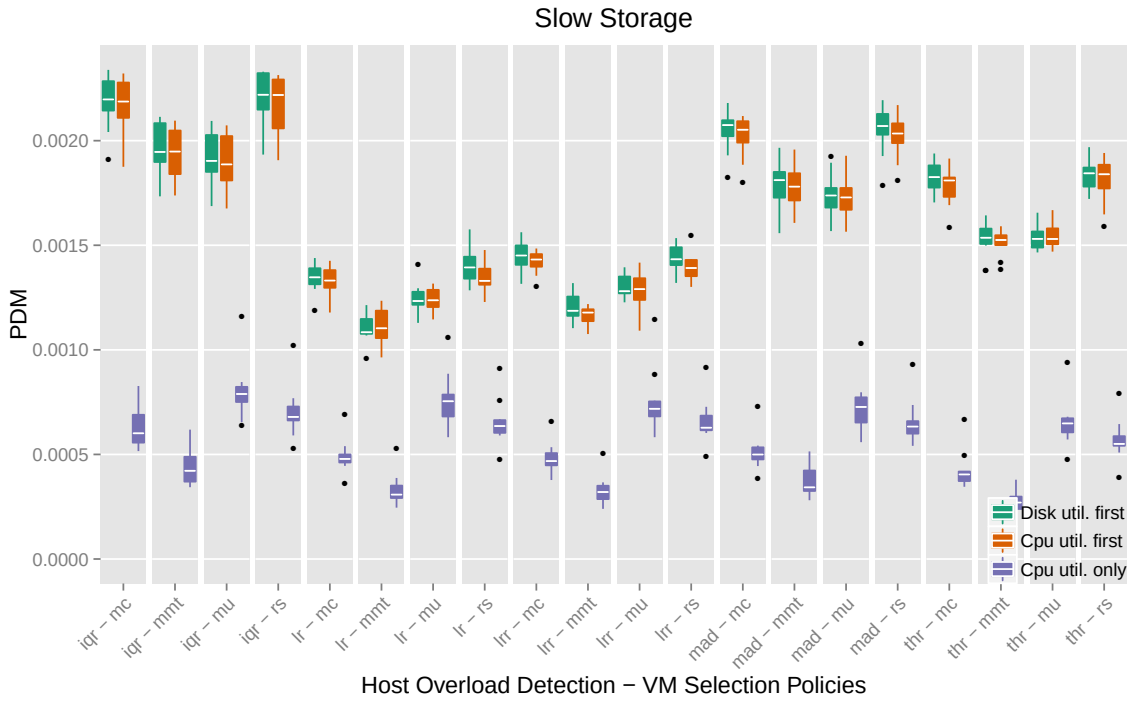

Figure 4.18: PDM - Slow storage

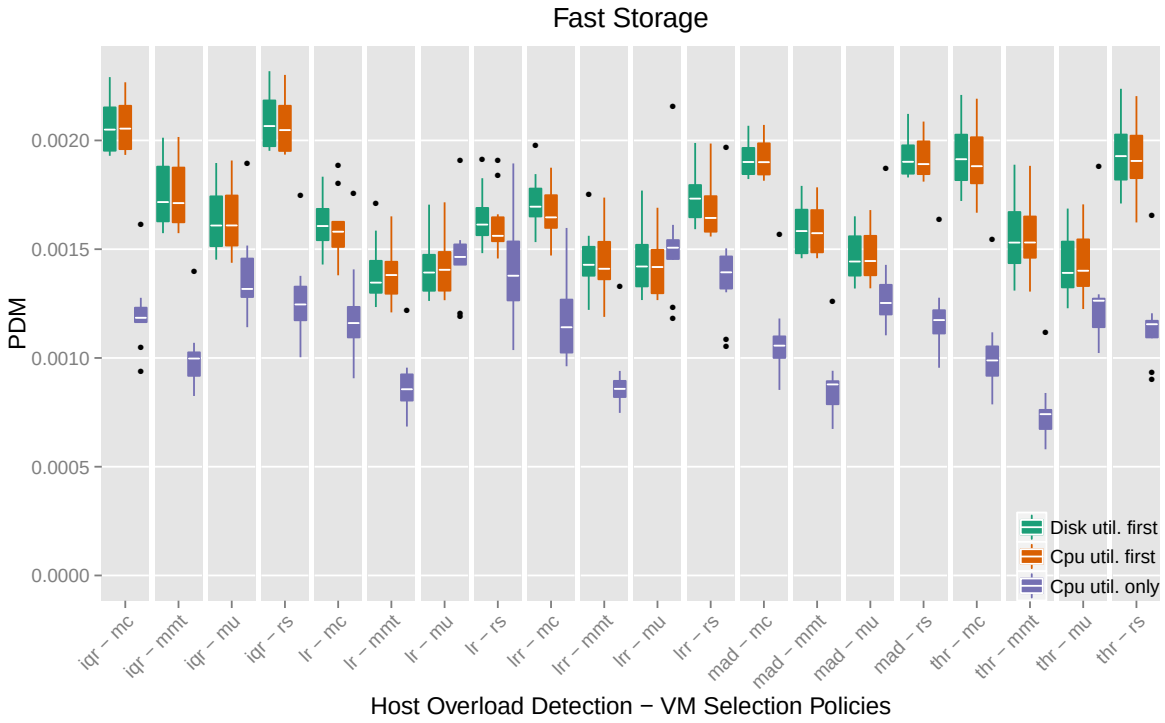

Figure 4.19: PDM - Fast storage

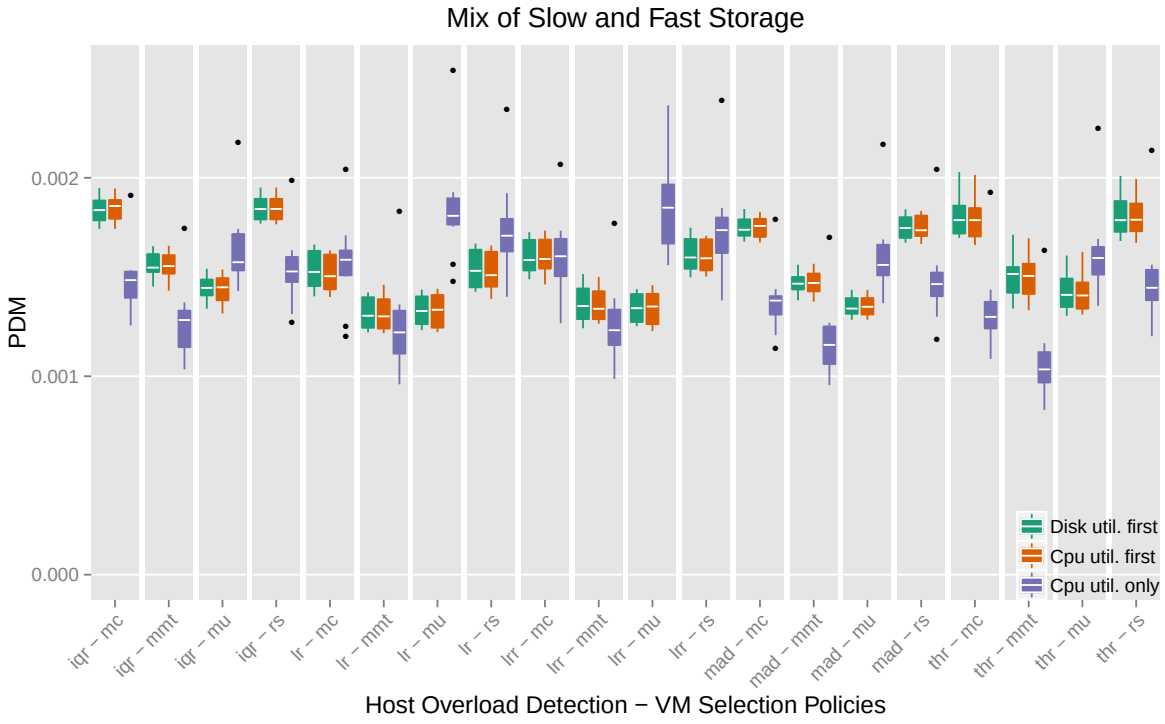

Figure 4.20: PDM - Mix of slow and fast storage

4.4.2 2nd Workload

For the 2nd workload we run simulations for a range of safety parameter values and we used the median values to chose the best values (see Figure 4.21): 1. THR-0.8 2. IQR-0.5 3. MAD-1.5 4. LRR-1.2 5. LR-1.2

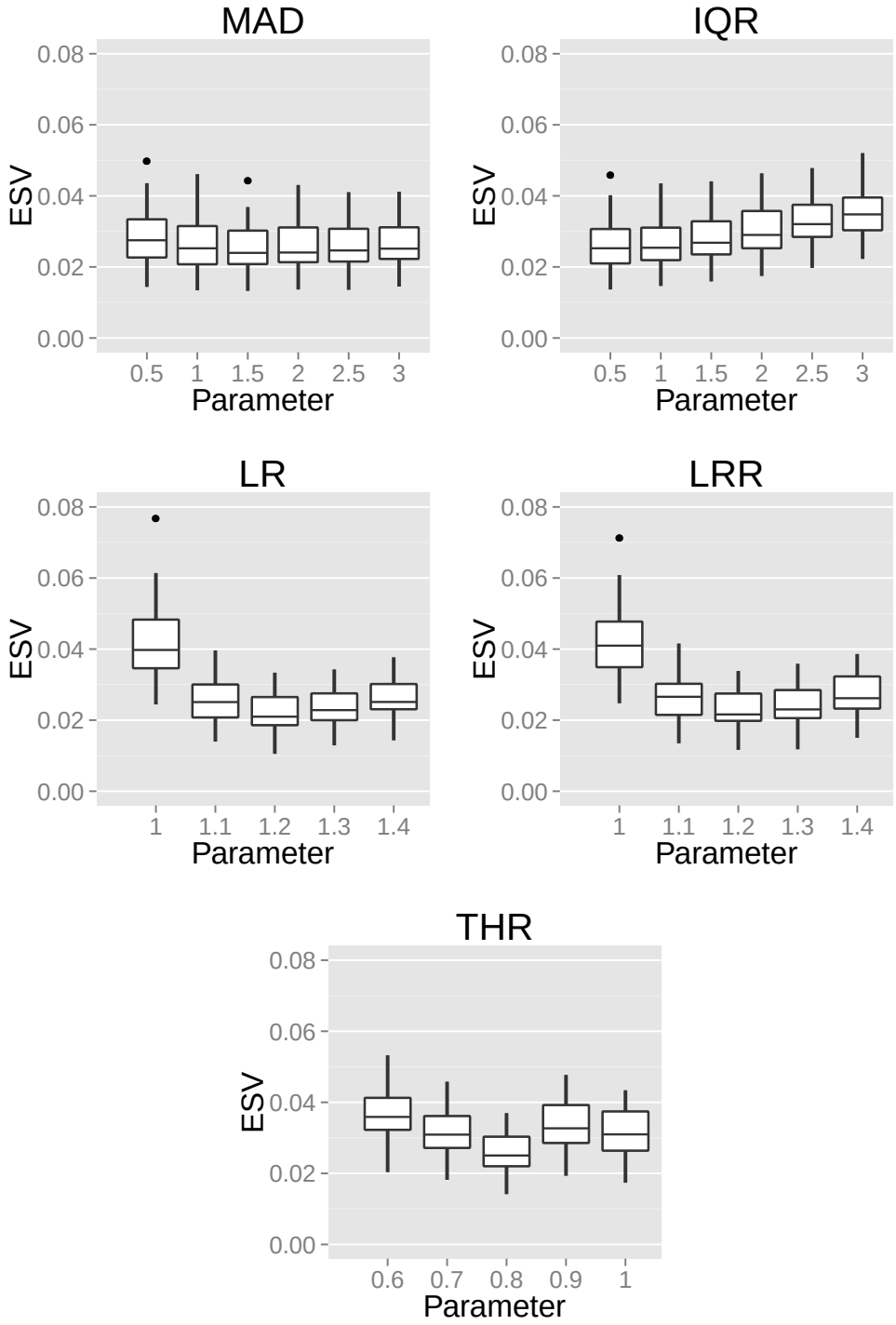

Figure 4.21: ESV for a range of safety parameter values

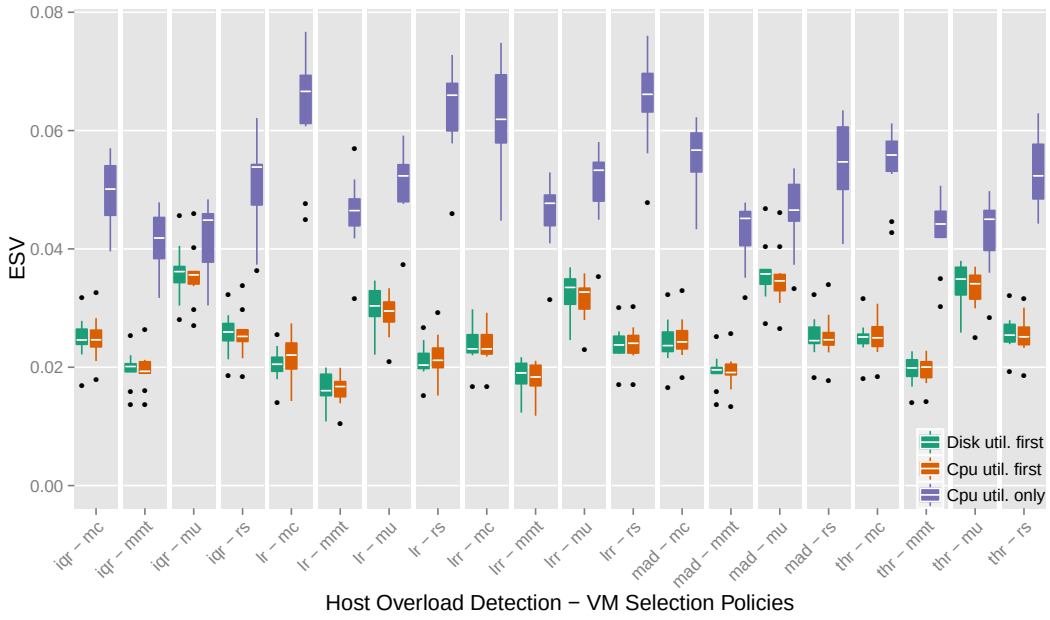

Figure 4.22: ESV

We can see that for the 2nd workload, Disk util. first and CPU util. first produce better results compare to CPU util. only. To confirm that there is a significant difference between the three approaches we performed statistical analysis. At first we tried to perform the ANOVA test, however the data failed to pass the Shapiro-Wilk test for normality and the Fligner-Killeen test of homogeneity of variances, two of the prerequisites of ANOVA. To make matters worse, failing the Fligner-Killeen means that we are unable to perform even non-parametric difference tests like Kruskal-Wallis.

We transform our data using the square root function. After the transformation the shapes of the distribution are similar enough (see Figure 4.23) to allow us to perform the Kruskal-Wallis test.

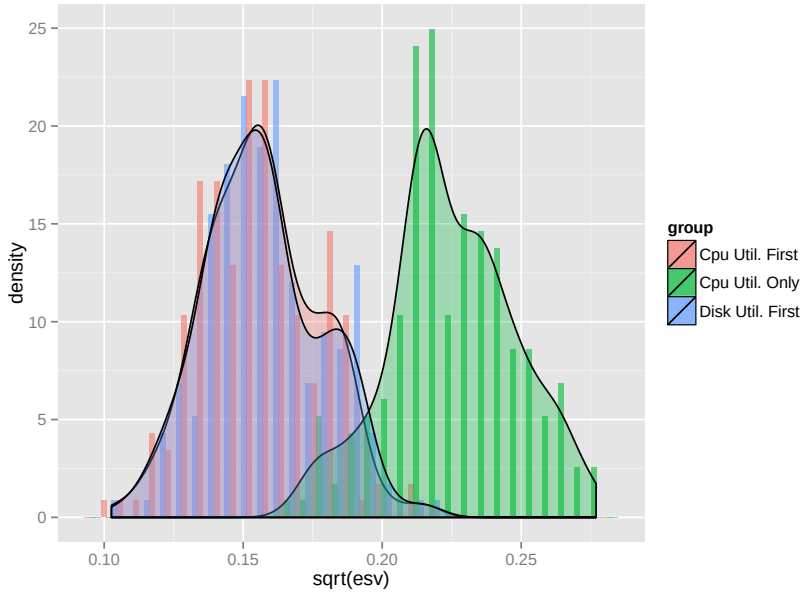

Figure 4.23: Transformed distributions

The Kruskal-Wallis test returns $p < 2.2 * 10^{-16}$. Consequently, the null hypothesis that the distributions are the same, is rejected. Following the Kruskal-Wallis test we performed multiple Wilcoxon signed-rank tests to determine the difference between each pair of distributions. The results are presented in Table 4.4. As we can see and as we intuitively concluded for workload 1, there is a statistical difference between CPU util. only and both Disk util. first and CPU util. first, but not between CPU util.first and Disk util. first.

Policy 1	**Policy 2**	**Difference** ($*10^{-2}$)	**95% C.I.** ($*10^{-2}$)	**P-value**
CPU util. only	CPU util. first	0.472	(0.413, 0.536)	$< 2.2 * 10^{-16}$
CPU util. only	Disk util. first	0.472	(0.410, 0.531)	$< 2.2 * 10^{-16}$
CPU util. first	Disk util. first	No statistical difference	-	0.905

Table 4.4: Wilcoxon signed-rank tests

Following the exact same process we see that Local Regression has a significant difference from all other host overload detection policies, and Minimum Migration Time has a significant difference from all other VM selection policies.

Chapter 5

Conclusions

In this study, we investigated the effect that storage utilization may have on VM consolidation mechanisms. We evaluated our approach using extensive simulations using a very large amount of workloads from real machines. The experiment results showed that in almost every case, taking into account both storage and CPU utilization leads to better consolidation decisions. Additionally, it became apparent that using the MMT policy for selecting VMs for migration and the LR policy for host overloading detection leads to statistical significant gains. Moreover, we showed that an underprovisioned system can have many adverse effects in the execution of a workload, including bigger energy consumption and more SLA violations.

Obviously, in order to confirm our conclusions real-world experiments are imperative. As a result, an implementation of our proposed system in a real Cloud computing platform like Openstack or Eucalyptus would be necessary. Another interesting idea for future work would be to use separate heuristics for detecting overutilization of different resources. Finally, further investigation for more robust and efficient heuristics might also bring greater gains.

Bibliography

[1] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. IEEE computer, 40(12):33–37, 2007.

[2] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. Concurrency and Computation: Practice and Experience, 24(13):1397–1420, 2012.

[3] Anton Beloglazov, Rajkumar Buyya, Young Choon Lee, and Albert Zomaya. A taxonomy and survey of energy-efficient data centers and cloud computing systems. arXiv preprint arXiv:1007.0066, 2010.

[4] The point contact transistor. http://ds.haverford.edu/bitbybit/bit-by-bit-contents/chapter-eight/8-2-the-point-contact-transistor/.

[5] Richard Brown et al. Report to congress on server and data center energy efficiency: Public law 109-431. Lawrence Berkeley National Laboratory, 2008.

[6] R. Buyya, J. Broberg, and A.M. Goscinski. Cloud Computing: Principles and Paradigms. Wiley Series on Parallel and Distributed Computing. Wiley, 2010.

[7] Jeffrey S Chase, Darrell C Anderson, Prachi N Thakar, Amin M Vahdat, and Ronald P Doyle. Managing energy and server resources in hosting centers. In ACM SIGOPS Operating Systems Review, volume 35, pages 103–116. ACM, 2001.

[8] William S Cleveland. Robust locally weighted regression and smoothing scatterplots. Journal of the American statistical association, 74(368):829–836, 1979.

[9] William S Cleveland and Clive Loader. Smoothing by local regression: Principles and methods. In Statistical theory and computational aspects of smoothing, pages 10–49. Springer, 1996.

[10] Problem child. http://content.time.com/time/magazine/article/0,9171,818829,00.html.

[11] Xiaobo Fan, Wolf-Dietrich Weber, and Luiz Andre Barroso. Power provisioning for a warehouse-sized computer. In ACM SIGARCH Computer Architecture News, volume 35, pages 13–23. ACM, 2007.

[12] W Forrest and K Brill. Revolutionizing data center efficiency. McKinsey & Company, 2008.

[13] Tony Freeth, Y Bitsakis, X Moussas, JH Seiradakis, A Tselikas, H Mangou, M Zafeiropoulou, R Hadland, D Bate, A Ramsey, et al. Decoding the ancient greek astronomical calculator known as the antikythera mechanism. Nature, 444(7119):587–591, 2006.

[14] Jim Gray and Prashant Shenoy. Rules of thumb in data engineering. In Data Engineering, 2000. Proceedings. 16th International Conference on, pages 3–10. IEEE, 2000.

[15] Introduction to virtual machines. https://labs.vmware.com/download/52/.

[16] Si-Yuan Jing, Shahzad Ali, Kun She, and Yi Zhong. State-of-the-art research study for green cloud computing. The Journal of Supercomputing, 65(1):445–468, 2013.

[17] J.G. Koomey, S. Berard, M. Sanchez, and H. Wong. Implications of historical trends in the electrical efficiency of computing. Annals of the History of Computing, IEEE, 33(3):46–54, March 2011.

[18] Jonathan Koomey. Growth in data center electricity use 2005 to 2010. A report by Analytical Press, completed at the request of The New York Times, 2011.

[19] Jonathan G Koomey. Estimating total power consumption by servers in the us and the world, 2007.

[20] Dara Kusic, Jeffrey O Kephart, James E Hanson, Nagarajan Kandasamy, and Guofei Jiang. Power and performance management of virtualized computing environments via lookahead control. Cluster computing, 12(1):1–15, 2009.

[21] Eric Masanet, Arman Shehabi, and Jonathan Koomey. Characteristics of low-carbon data centres. Nature Climate Change, 3(7):627–630, 2013.

[22] Peter Mell and Tim Grance. The nist definition of cloud computing. 2011.

[23] Gordon E. Moore. Cramming more components onto integrated circuits. In Mark D. Hill, Norman P. Jouppi, and Gurindar S. Sohi, editors, Readings in computer architecture, pages 56–59. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[24] Ripal Nathuji and Karsten Schwan. Virtualpower: coordinated power management in virtualized enterprise systems. In ACM SIGOPS Operating Systems Review, volume 41, pages 265–278. ACM, 2007.

[25] Eduardo Pinheiro, Ricardo Bianchini, Enrique V Carrera, and Taliver Heath. Load balancing and unbalancing for power and performance in cluster-based systems. In Workshop on compilers and operating systems for low power, volume 180, pages 182–195. Barcelona, Spain, 2001.

[26] The problem of power consumption in servers. https://software.intel.com/en-us/articles/the-problem-of-power-consumption-in-servers.

[27] In the data center power and cooling costs more than the it equipment it supports. http://www.electronics-cooling.com/2007/02/in-the-data-center-power-and-cooling-costs-more-than-the-it-equipment-it-supports/.

[28] Neil JJ Salkind. Encyclopedia of measurement and statistics. Sage Publications, 2006.

[29] Amit Sinha and Anantha Chandrakasan. Dynamic power management in wireless sensor networks. Design & Test of Computers, IEEE, 18(2):62–74, 2001.

[30] J Edward Smith and Ravi Nair. Introduction to virtual machines. Virtual machines: versatile platforms for systems and processes, pages 9–10, 2005.

[31] Shekhar Srikantaiah, Aman Kansal, and Feng Zhao. Energy aware consolidation for cloud computing. In Proceedings of the 2008 conference on Power aware computing and systems, volume 10. San Diego, California, 2008.

[32] Akshat Verma, Gargi Dasgupta, Tapan Kumar Nayak, Pradipta De, and Ravi Kothari. Server workload analysis for power minimization using consolidation. In Proceedings of the 2009 conference on USENIX Annual technical conference, pages 28–28. USENIX Association, 2009.

[33] History of virtualization. http://www.everythingvm.com/content/history-virtualization.

[34] Molly Webb et al. Smart 2020: Enabling the low carbon economy in the information age. The Climate Group. London, 1(1):1–1, 2008.

[35] Code-breaking at bletchley park during world war ii, 1939-1945. http://www.ieeeghn.org/wiki/index.php/Milestones:Code-breaking_at_Bletchley_Park_during_World_War_II,_1939-1945.

[36] Minyi Yue. A simple proof of the inequality ffd (l)≤ 11/9 opt (l)+ 1,□ l for the ffd bin-packing algorithm. Acta mathematicae applicatae sinica, 7(4):321–331, 1991.