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

Μοντελοποίηση Εφαρμογών σε Αρχιτεκτονικές NUMA με Τεχνικές Μηχανικής Μάθησης

Διπλωματική Εργασία

Συγγραφέας:
Φανούριος Αραπίδης

Επιβλέπων:
Γεώργιος Γκούμας

Αθήνα, Νοέμβριος, 2018

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Μοντελοποίηση Εφαρμογών σε Αρχιτεκτονικές NUMA με Τεχνικές Μηχανικής Μάθησης

Διπλωματική Εργασία

Συγγραφέας:
Φανούριος Αραπίδης

Επιβλέπων:
Γεώργιος Γκούμας

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις
19 Νοεμβρίου 2018.

Νεκτάριος Κοζύρης	Γεώργιος Γκούμας	Νικόλαος Παπασπύρου
Καθηγητής, Ε.Μ.Π	Καθηγητής, Ε.Μ.Π.	Καθηγητής, Ε.Μ.Π.

Αθήνα, Νοέμβριος 2018

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

Περίληψη

Σε αυτή τη διπλωματική εργασία παρουσιάζουμε μια προσέγγιση με τεχνικές μηχανικής μάθησης για να προβλέψουμε τον αντίκτυπο που έχει στην απόδοση εφαρμογών η τοποθέτηση των πυρήνων και της μνήμης σε συστήματα μη κοινής πρόσβασης μνήμης. Ο αντίκτυπος στην απόδοση εξαρτάται από την αρχιτεκτονική του συστήματος και τα χαρακτηριστικά των εφαρμογών. Επικεντρώσαμε την έρευνα μας σε χαρακτηριστικά που μπορούμε εύκολα να τα μετρήσουμε με τη χρήση των hardware performance counters που βρίσκονται ενσωματωμένοι στο σύστημα. Εκτελέσαμε μια μεγάλη ποικιλία από μονονηματικές εφαρμογές από τις σουίτες Spec2006 και Parsec κάτω από διαφορετικά σενάρια τοποθέτησης, και χρησιμοποιήσαμε τα δεδομένα που πήραμε από τις μετρήσεις για να εκπαιδεύσουμε μοντέλα πρόβλεψης τα οποία μπορούν να χρησιμοποιηθούν για να προβλέψουμε την απόδοση. Τα αποτελέσματα μας δείχνουν αξιοσημείωτη ακρίβεια στην πρόβλεψη της απόδοσης με σχετικά απλά μοντέλα. Τέλος, μελετήσαμε το τρόπο που μπορούμε να μεταφέρουμε τη μνήμη μιας εφαρμογής κατά την διάρκεια της εκτέλεσης.

Λέξεις κλειδιά

απόδοση, μοντελοποίηση, NUMA, τοποθέτηση

Abstract

In this thesis we present a machine-learning approach to predict the impact on performance of core and memory placement in non-uniform memory access (NUMA) systems. The impact on performance depends on the architecture and the application's characteristics. We focus our study on features that can be easily extracted with hardware performance counters that are found in commodity off-the-self systems. We run various single-threaded benchmarks from Spec2006 and Parsec under different placement scenarios, and we use this benchmarking data to train multiple regression models that could serve as performance predictors. Our experimental results show notable accuracy in predicting the impact on performance with relatively simple prediction models. Finally, we studied how we can transfer the memory of an application during the execution time.

Key words

performance, modeling, NUMA, placement

Ευχαριστίες

Τα τελευταία έξι χρόνια είχα ένα αρκετά μεγάλο ταξίδι για την ακαδημαϊκή μου κατάρτιση. Σε αυτό το ταξίδι χρειάστηκε να αντιμετωπίσω αρκετές δυσκολίες, όμως για καλή μου τύχη δεν ήμουν μόνος μου. Θα ήθελα να ευχαριστήσω πολύ θερμά την μητέρα μου και τα δύο μικρότερα αδέλφια μου που με στήριξαν σε όλα τα χρόνια της ζωής και συνεχίζουν να με στηρίζουν ακόμα και σήμερα.

Θα ήθελα να ευχαριστήσω τους παιδικούς μου φίλους Άρη και Σπύρο που μαζί τους έχω ζήσει αρκετές δυνατές στιγμές και έχουμε μια πολύ δυνατή φιλία. Ακόμα θα ήθελα να ευχαριστήσω τους συμφοιτητές και φίλους που έκανα χάρη της σχολής τον Πέτρο, τον Γιώργο, την Έλλη και το Κοπέλι. Επίσης θέλω να ευχαριστήσω τον κ. Σαμιοτάκη που με βοήθησε αρκετά.

Θα ήθελα να ευχαριστήσω του καθηγητές μου κ. Παπασπύρου, κ. Γκούμα και κ. Κοζύρη, που ο καθένας με γαλούχησε με τον δικό του μοναδικό τρόπο και με μύησαν στην τέχνη των υπολογιστών. Τέλος, θα ήθελα να ευχαριστήσω τον Βασίλη, την Νικέλα και τον Στέφανο που ήταν μέντορες μου και με καθοδήγησαν σε αυτήν την διπλωματική εργασία.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Η εξέλιξη της τεχνολογίας έχει οδηγήσει στην καθιέρωση πολυεπεξεργαστικών υπολογιστικών συστημάτων με τον αριθμό των διαθέσιμων πυρήνων να αυξάνεται συνεχώς. Τα προηγούμενα χρόνια τέτοια συστήματα χρησιμοποιούνταν κυρίως για ερευνητικούς σκοπούς, όμως πλέον τα χρησιμοποιούμε αρκετά έντονα στην καθημερινή μας ζωή. Αυτό το φαινόμενο το παρατηρούμε συχνά, από τα κινητά τηλέφωνα μας μέχρι και τους προσωπικούς υπολογιστές που έχουμε στο σπίτι που διαθέτουν αρκετά μεγάλο αριθμό από πυρήνες σε σχέση με τα προηγούμενα χρόνια.

Την τελευταία δεκαετία έχουμε καταφέρει να φτιάξουμε συστήματα τα οποία αποτελούνται από εκατοντάδες πυρήνες. Χαρακτηριστικό παράδειγμα σε αυτό είναι ο υπερυπολογιστής (supercomputer) “Sunway TaihuLight” που κέρδισε την πρώτη θέση στον κατάλογο Top500 το 2016 με 10.649.600 πυρήνες. Όπως γνωρίζουμε από τον “Νόμο του Μουρ” (Moore’s law) ’ο αριθμός των τρανζίστορ ενός πυκνού ολοκληρωμένου κυκλώματος διπλασιάζεται κάθε δύο χρόνια’, αυτό πλέον συμβαίνει κάθε 18 μήνες. Έτσι ο αριθμός των τρανζίστορ που αποτελούνται οι πυρήνες αυξάνει εκθετικά σε σχέση με τον χρόνο. Όμως δεν συμβαίνει το ίδιο και με την μνήμη.

Η ταχύτητα πρόσβασης της μνήμης αυξάνει γραμμικά με τον χρόνο. Αυτό έχει σαν αποτέλεσμα το γνωστό πρόβλημα “CPU and Memory Gap”. Οι επεξεργαστές γίνονται πολύ πιο γρήγοροι σε σχέση με την μνήμη και οι εφαρμογές “λιμοκτονούν” περιμένοντας δεδομένα από την κύρια μνήμη. Αυτό το πρόβλημα διογκώνεται συνεχώς καθώς το κενό επεξεργαστών και μνήμης αυξάνει.

Μια ειδική κατηγορία πολυεπεξεργαστικών συστημάτων είναι τα συστήματα μη κοινής πρόσβασης μνήμης ή αλλιώς NUMA (Non-Uniform Memory Access) συστήματα. Στα NUMA συστήματα ο χρόνος πρόσβασης στην μνήμη εξαρτάται από την σχετική θέση της μνήμης ως προς τους επεξεργαστές. Ένας επεξεργαστής του NUMA συστήματος, έχει μικρότερο χρόνο πρόσβασης στην τοπική του μνήμη σε σχέση με μνήμη που δεν είναι δεσμευμένη τοπικά (μνήμη τοπική σε άλλον επεξεργαστή ή μνήμη που μοιράζεται μεταξύ επεξεργαστών). Εκμεταλλευόμαστε καλά τις δυνατότητες των NUMA συστημάτων σε εφαρμογές που έχουν συγκεκριμένο φόρτο εργασίας (job centric), κυρίως σε διακομιστές (servers) όπου τα δεδομένα συσχετίζονται συχνά με συγκεκριμένες εργασίες ή χρήστες.

Τα NUMA συστήματα προσπαθούν να αντιμετωπίσουν το πρόβλημα του “CPU and Memory Gap” παρέχοντας ξεχωριστή μνήμη για κάθε επεξεργαστή. Έτσι αποφεύγουν τα προβλήματα που προκύπτουν όταν πολλοί επεξεργαστές θέλουν να κάνουν ταυτόχρονη πρόσβαση σε κοινή μνήμη. Για τα προβλήματα που αφορούν τη διάδοση δεδομένων (spread data), που είναι συχνό φαινόμενο σε διακομιστές και παρόμοιες εφαρμογές, τα NUMA συστήματα μπορούν να βελτιώσουν σημαντικά την απόδοση σε μια εφαρμογή

με κοινή μνήμη κατά περίπου τον αριθμό των επεξεργαστών.

Όπως βλέπουμε τα NUMA συστήματα είναι ένας αρκετά ενδιαφέρον ερευνητικός τομέας, καθώς λύνουν αρκετά σημαντικά προβλήματα και είναι πολύ διαδεδομένη η χρήση τους σε διακομιστές. Άρα είναι πολύ σημαντικό να εκμεταλλευόμαστε τους πόρους που μας παρέχει το σύστημα στο έπακρον. Όμως πολλές φορές αυτό δεν είναι δυνατό καθώς υπάρχει ανισοκατανομή εργασιών μέσα στο σύστημα, που περιορίζει τις δυνατότητες του. Αυτό είναι συχνό φαινόμενο στους cloud computing servers.

Οι cloud computing servers παρέχουν στους χρήστες τους εικονικές μηχανές ή αλλιώς VMs (Virtual Machine). Οι VMs είναι τοποθετημένοι πάνω σε ένα NUMA σύστημα. Καθώς κατασκευάζουμε και καταστρέφουμε VMs η αρχική εικόνα του συστήματος αλλάζει. Μετά από κάποιο χρονικό διάστημα παρατηρείται ανισοκατανομή εργασίας στο σύστημα. Δηλαδή υπάρχουν κομμάτια του NUMA συστήματος που δουλεύουν πιο έντονα και ανταγωνίζονται για τους πόρους του συστήματος και άλλα που μπορεί να είναι σχεδόν ανενεργά. Ακόμα υπάρχουν και περιπτώσεις που οι πόροι που διαθέτει ένα σύστημα δεν είναι αρκετοί και χρειάζεται κάποια VMs να πάνε σε ένα άλλο σύστημα.

Τέτοια και άλλα παρόμοια προβλήματα χρειάζεται να λυθούν για να μπορέσουμε να εκμεταλλευτούμε πλήρως ένα NUMA σύστημα. Στις περισσότερες περιπτώσεις μπορούμε να διορθώσουμε αυτό το πρόβλημα μεταφέροντας τις εφαρμογές μας μέσα στο σύστημα. Όμως δεν επηρεάζονται όλες οι εφαρμογές μας με τον ίδιο τρόπο. Οπότε χρειαζόμαστε ένα τρόπο να μπορέσουμε να αξιολογήσουμε ποιες εφαρμογές είναι καλύτερες για την μεταφορά τους μέσα στο σύστημα.

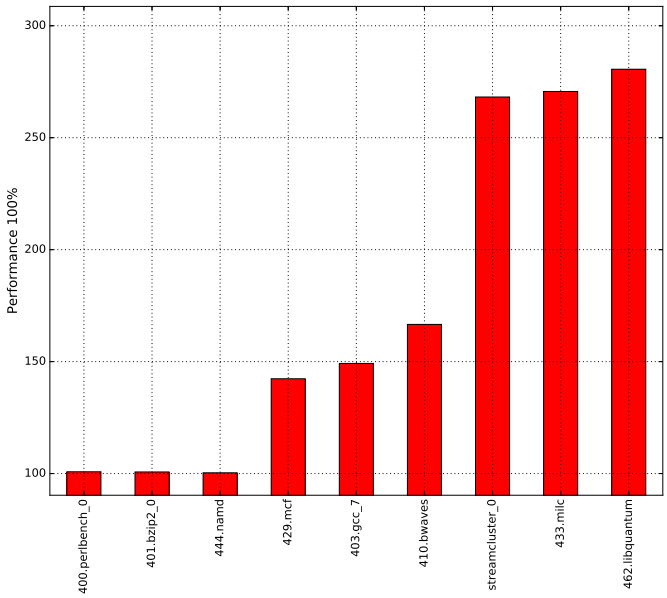
Συνεπώς, στην παρούσα διπλωματική εργασία αποφασίσαμε να μελετήσουμε τη συμπεριφορά που έχουν διάφορες εφαρμογές μέσα σε ένα NUMA σύστημα. Πιο συγκεκριμένα, μελετάμε πως επηρεάζονται διάφορες εφαρμογές από την αρχιτεκτονική τέτοιων συστημάτων με σκοπό να μπορέσουμε να κατασκευάσουμε ένα μοντέλο που θα μας λέει ποιες εφαρμογές είναι καταλληλότερες για μεταφορά μέσα στο σύστημα. Αυτό το μοντέλο θα μπορεί να χρησιμοποιηθεί σαν ένα αξιόπιστο κριτήριο από τα NUMA συστήματα για να λύνουμε προβλήματα σαν αυτό που είδαμε.

1.1 Κίνητρο και έρευνα

Το κίνητρο μας για την διπλωματική εργασία είναι ότι δεν επηρεάζονται όλες οι εφαρμογές που τρέχουν σε ένα NUMA σύστημα με τον ίδιο τρόπο από την θέση της μνήμης τους. Όπως βλέπουμε στο σχήμα 1.1 η απόδοση τον εφαρμογών ποικίλει τόσο από μηχάνημα σε μηχάνημα όσο και μέσα στο ίδιο το μηχάνημα. Υπάρχουν εφαρμογές που φαίνεται ότι επηρεάζονται αρκετά από την θέση που βρίσκεται η μνήμη και άλλες που τους είναι αδιάφορο.

Θέλουμε να κατασκευάσουμε ένα μοντέλο που θα αξιολογεί ποιες εφαρμογές είναι πιο κατάλληλες για να τις μεταφέρουμε στο σύστημα μας. Εφαρμογές που δεν επηρεάζονται από την μνήμη τους είναι ιδανικές για αυτήν την διαδικασία. Όμως δεν γίνεται να γνωρίζουμε εκ των προτέρων ποιες είναι αυτές οι εφαρμογές. Οπότε για να κάνουμε μια καλή αξιολόγηση χρειάζεται να μπορούμε να προβλέψουμε ποιες είναι αυτές οι εφαρμογές που δεν επηρεάζονται.

Στην έρευνα μας μελετήσαμε την συμπεριφορά που έχουν οι εφαρμογές μας σε ένα NUMA σύστημα. Για να μπορέσουμε να ερμηνεύσουμε αυτήν τους την συμπεριφορά χρησιμοποιήσαμε διάφορα εργαλεία για να μετρήσουμε μερικές παραμέτρους τους. Οι μέθοδοι που χρησιμοποιούμε καθώς και τα εργαλεία μας πρέπει να είναι μετακινήσιμα



Σχήμα 1.1: Παράδειγμα εφαρμογών που επηρεάζονται από την μνήμη.

από μηχάνημα σε μηχάνημα, έτσι ώστε η έρευνα μας να μην περιορίζεται σε ένα εξειδικευμένο μόνο μηχάνημα.

Στην συνέχεια χρησιμοποιούμε της παραμέτρους που μετρήσαμε σε κάθε μηχάνημα για να κατασκευάσουμε το μοντέλο που θα μπορεί να κάνει την πρόβλεψη. Για να το κάνουμε αυτό κατασκευάσαμε κάποιους παραγώγους συναντήσεων (function generators) που μας παρείχαν όλες τις μη-γραμμικές συναρτήσεις που θέλαμε. Αυτές τις συναρτήσεις στην συνέχεια τις εκπαιδεύσαμε μέσω τεχνικών μηχανικής μάθησης που πέφτουν κάτω από την ομπρέλα της επιβλεπόμενης μάθησης (supervised learning). Έπειτα αξιολογήσαμε πόσο καλά μπορούν αυτές οι συναρτήσεις να προβλέψουν την μεταβολή της απόδοσης των εφαρμογών μας.

Τέλος, για να έχουμε μια πλήρη εικόνα για την συμπεριφορά των εφαρμογών μας, αποφασίσαμε να δούμε πως μπορούμε να μεταφέρουμε την μνήμη τους κατά την διάρκεια της εκτέλεσης τους. Έτσι θα μπορέσουμε να χρησιμοποιήσουμε το μοντέλο πρόβλεψης όχι μόνο στην αρχή μιας εφαρμογής αλλά και κατά τη διάρκεια λειτουργίας τους.

1.2 Στόχοι διπλωματικής

Οι στόχοι της διπλωματικής εργασίας είναι οι εξής:

- Να βρούμε και να μετρήσουμε κατάλληλες παραμέτρους από τις εφαρμογές μας, που μπορούν να ερμηνεύσουν τη συμπεριφορά τους, ανεξάρτητα από το σύστημα που εξετάζουμε στην εκάστοτε περίπτωση.
- Να μοντελοποιήσουμε αυτές τις εφαρμογές με σκοπό να κατασκευάσουμε μια μέθοδο που θα μπορούμε να προβλέψουμε την μεταβολή της απόδοσης τους σε σχέση με την θέση της μνήμης τους σε ένα NUMA σύστημα.
- Να μελετήσουμε τους τρόπους που μπορούμε να μεταφέρουμε την μνήμη μιας εφαρμογής μέσα σε ένα NUMA σύστημα.

1.3 Διάρθρωση κειμένου

Η δομή των κεφαλαίων της διπλωματικής εργασίας είναι η ακόλουθη:

Κεφάλαιο 2: Θεωρητικό Υπόβαθρο
Στο κεφάλαιο αυτό αρχικά αναπτύσσουμε ορισμένες έννοιες γύρω από τα NUMA συστήματα. Θα δούμε κάποιους βασικούς ορισμούς που θα χρησιμοποιήσουμε στην συνέχεια της έρευνας μας καθώς και τα στοιχεία από τα οποία αποτελούνται αυτά τα συστήματα.

Κεφάλαιο 3: Ανάλυση Μηχανημάτων
Σε αυτό το κεφάλαιο θα δούμε τα αρχιτεκτονικά χαρακτηριστικά των μηχανημάτων που χρησιμοποιήσαμε για την έρευνα μας. Όλα τα πειράματα τα εκτελέσαμε εξίσου και στα δύο μηχανήματα.

Κεφάλαιο 4: Βασικές Μεθοδολογίες
Σε αυτό το κεφάλαιο θα ασχοληθούμε με τα αρχικά ζητήματα που μας προβληματίσαν στην έρευνα μας. Θα δούμε κάποια βασικά προβλήματα που είχαμε καθώς και τον τρόπο που τα αντιμετωπίσαμε για να θέσουμε τα θεμέλια για την έρευνα μας. Επίσης, θα αναλύσουμε τα εργαλεία που χρησιμοποιήσαμε για να πραγματοποιήσουμε την έρευνα μας. Οι μέθοδοι σε αυτό το κεφάλαιο θεωρούνται δομικός λίθος για όλη την έρευνα που πραγματοποιήθηκε.

Κεφάλαιο 5: Επίδοση Εφαρμογών και Μνήμης
Σε αυτό το κεφάλαιο θα ασχοληθούμε με το σημαντικότερο κομμάτι της έρευνας που κάναμε. Εδώ θα μετρήσουμε συγκεκριμένα χαρακτηριστικά εφαρμογών και θα προσπαθήσουμε να βρούμε μια σχέση που θα περιγράφει το πώς επηρεάζεται η απόδοση τον εφαρμογών σε σχέση με την μνήμη τους. Στην συνέχεια θα γενικεύσουμε αυτήν την μέθοδο έτσι ώστε να μπορούμε να προβλέψουμε την συμπεριφορά της εκάστοτε εφαρμογής όταν θέλουμε να κάνουμε μια λειτουργία στην μνήμη της.

Κεφάλαιο 6: Μεταφορά Μνήμης
Σε αυτό το κεφάλαιο θα δούμε κάποιες μεθόδους που έχουμε στην διάθεση μας για να μεταφέρουμε την μνήμη μιας εφαρμογής κατά την διάρκεια εκτέλεσης της. Θα δούμε τα βασικά χαρακτηριστικά τους καθώς και πως μπορούμε να εκμεταλλευτούμε τις λειτουργίες τους. Τέλος θα δούμε κάποια από τα αποτελέσματα που πήραμε με την χρήση αυτών των μεθόδων.

Κεφάλαιο 7: Επίλογος
Με το κεφάλαιο αυτό ολοκληρώνεται η παρούσα διπλωματική εργασία. Στο κεφάλαιο αυτό παρουσιάζουμε ορισμένα γενικότερα συμπεράσματα που βγάλαμε από τη διπλωματική και αναφέρουμε ορισμένες μελλοντικές επεκτάσεις που έχουν ενδιαφέρον.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό αρχικά αναπτύσσουμε ορισμένες έννοιες γύρω από τα NUMA συστήματα. Θα δούμε κάποιους βασικούς ορισμούς που θα χρησιμοποιήσουμε στην συνέχεια της έρευνας μας καθώς και τα στοιχεία από τα οποία αποτελούνται αυτά τα συστήματα.

2.1 Βασικές ορολογίες

Σε αυτό το υποκεφάλαιο παρέχουμε θεωρητικό υπόβαθρο το οποίο βοηθάει στην κατανόηση της έρευνας που κάναμε. Θα δούμε τα βασικά στοιχεία από τα οποία αποτελείται ένα NUMA σύστημα και θα δώσουμε κάποιους ορισμούς για αυτά τα συστήματα. Ακόμα θα δούμε κάποιες διαφορές που έχουν αυτά τα συστήματα σε σχέση με τους κλασικά υπολογιστικά συστήματα.

Τα NUMA συστήματα είναι μια ειδική κατηγορία πολυεπεξεργαστικών υπολογιστικών συστημάτων. Συνήθως αποτελούνται από πολλούς πολυνηματικούς πυρήνες και διαθέτουν μεγάλες ποσότητες μνήμης RAM. Τέτοια συστήματα δεν βρίσκουν έντονη χρήση στην καθημερινότητα μας, όμως χρησιμοποιούνται πολύ συχνά σε ερευνητικά κέντρα που διαθέτουν υπερυπολογιστές. Η πιο διαδεδομένη χρήση στην αγορά είναι σε διακομιστές (servers) από cloud computing υπηρεσίες που παρέχουν VMs στους χρήστες τους.

Ας δούμε τα κύρια στοιχεία από τα οποία αποτελείται ένα NUMA σύστημα:

Κόμβοι

Οι κόμβοι ενός συστήματος NUMA ίσως είναι το πιο χαρακτηριστικό κομμάτι ενός τέτοιου συστήματος. Ως κόμβο ενός συστήματος NUMA ορίζουμε μια διακριτή ομάδα από επεξεργαστές οι οποίοι είναι συνδεδεμένοι πάνω στο ίδιο socket. Ένα NUMA σύστημα αποτελείται από τουλάχιστον 2 κόμβους και πάνω.

Δίκτυο διασύνδεσης

Ένα εξίσου σημαντικό χαρακτηριστικό των NUMA συστημάτων είναι το δίκτυο διασύνδεσης τους. Ως δίκτυο διασύνδεσης ορίζουμε τον τρόπο με τον οποίο είναι συνδεδεμένοι οι κόμβοι του μηχανήματος μεταξύ τους. Ο κάθε κόμβος του συστήματος πρέπει να είναι συνδεδεμένος τουλάχιστον με έναν άλλο κόμβο του συστήματος με τέτοιο τρόπο ώστε όλοι οι κόμβοι του συστήματος να μπορούν να επικοινωνήσουν μεταξύ τους. Ωστόσο, δεν είναι αναγκαίο ότι κάθε κόμβος χρειάζεται να έχει απευθείας επικοινωνία με όλους τους άλλους κόμβους του συστήματος. Τέλος, τα δίκτυα διασύνδεσης μπορεί να διαφέρουν πολύ από σύστημα σε σύστημα, καθώς εξαρτώνται άμεσα από το πλήθος

των κόμβων που διαθέτει το σύστημα και από τον τρόπο με τον οποίο είναι συνδεδεμένοι αυτοί οι κόμβοι.

Κύρια μνήμη (RAM)

Ένα άλλο σημαντικό χαρακτηριστικό στοιχείο των NUMA συστημάτων είναι η κύρια μνήμη τους. Αυτό είναι το κύριο χαρακτηριστικό που ξεχωρίζει τα NUMA συστήματα από τα άλλα πολυεπεξεργαστικά συστήματα. Η κύρια μνήμη ενός NUMA συστήματος δεν είναι συνεχόμενη όπως συμβαίνει στα κλασικά υπολογιστικά συστήματα αλλά είναι χωρισμένη, συνήθως σε ισάριθμα κομμάτια, στους κόμβους του συστήματος. Έτσι ο κάθε κόμβος έχει ένα κομμάτι μνήμης που είναι κοντά του και την ονομάζουμε τοπική μνήμη (local memory) και όλο την υπόλοιπη μνήμη που είναι μακριά του και την ονομάζουμε απομακρυσμένη μνήμη (remote memory).

Η τοπική και απομακρυσμένη μνήμη εξαρτώνται από την σχετική θέση αυτής και του αντίστοιχου κόμβου. Για παράδειγμα ένα κομμάτι μνήμης είναι τοπικό για ένα κόμβο και για έναν άλλο κόμβο το ίδιο κομμάτι μνήμης είναι απομακρυσμένο. Ο χρόνος που χρειάζεται ένας επεξεργαστής για να κάνει πρόσβαση στην τοπική του μνήμη είναι πάντα μικρότερος από τον χρόνο που χρειάζεται για μια απομακρυσμένη. Ακόμα ο χρόνος που χρειάζεται για να κάνει πρόσβαση σε μια απομακρυσμένη μνήμη δεν είναι ίδιος για όλα τα κομμάτια της. Δηλαδή μπορεί κάποια κομμάτια απομακρυσμένης μνήμης να είναι πιο μακριά από κάποια άλλα.

Επειδή όλοι οι κόμβοι του συστήματος μπορούν να επικοινωνήσουν μεταξύ τους, αυτό σημαίνει ότι όλες οι διεργασίες που τρέχουν πάνω σε ένα NUMA σύστημα έχουν πρόσβαση σε όλη την κύρια μνήμη. Κάθε κόμβος του συστήματος έχει και ένα ελεγκτή για την κύρια μνήμη. Οι ελεγκτές της κύριας μνήμης είναι υπεύθυνοι για την σωστή οργάνωση και λειτουργία του συστήματος όσον αφορά τις προσβάσεις των εφαρμογών στην κύρια μνήμη.

Συμμετρικά και Ασύμμετρα Μηχανήματα

Όπως είδαμε όλοι οι κόμβοι έχουν την δυνατότητα να μπορούν να επικοινωνήσουν μεταξύ τους. Το πόσος χρόνος χρειάζεται για να επικοινωνήσουν δύο κόμβοι ονομάζεται χρόνος επικοινωνίας και ορίζει την μεταξύ τους απόσταση. Όσο μεγαλύτερη είναι η απόσταση δύο κόμβων, τόσο πιο αργή είναι και η επικοινωνία τους. Ακόμα, η απόσταση δύο κόμβων δεν είναι μονοσήμαντη αλλά μπορεί να έχει κατεύθυνση. Για παράδειγμα, έστω ότι έχουμε τους κόμβους A και B. Η απόσταση που βλέπει ένας επεξεργαστής που βρίσκεται στον κόμβο A για την μνήμη του κόμβου B δεν είναι απαραίτητα η ίδια με την απόσταση που βλέπει ένας επεξεργαστής που βρίσκεται στον κόμβο B για την μνήμη του κόμβου A. Όταν και οι δύο κόμβοι βλέπουν την ιδία απόσταση για την μνήμη που βρίσκεται στον άλλο κόμβο τότε αυτή η απόσταση ονομάζεται συμμετρική. Δηλαδή αν ισχύει:

$$\text{Distance}(\text{processor}_A, \text{memory}_B) = \text{Distance}(\text{processor}_B, \text{memory}_A)$$

Αν αυτό δεν ισχύει, τότε η απόσταση ονομάζεται ασύμμετρη.

Τα NUMA συστήματα μπορούμε να τα χωρίσουμε σε δύο βασικές κατηγορίες ανάλογα με είδος των αποστάσεων που έχουν. Αυτές οι δύο κατηγορίες είναι τα συμμετρικά μηχανήματα και τα ασύμμετρα. Συμμετρικό λέμε ένα μηχάνημα που όλες οι αποστάσεις μεταξύ των κόμβων είναι συμμετρικές. Αν έστω και μια απόσταση είναι μη συμμετρική, τότε το σύστημα λέγετε ασύμμετρο. Αυτός ο διαχωρισμός είναι αρκετά σημαντικός καθώς τα ασύμμετρα μηχανήματα είναι πιο πολύπλοκα σε σχέση με τα συμμετρικά. Όμως, και τα

δύο είδη μπορούμε να τα προσεγγίσουμε το ίδιο καλά με τις τεχνικές που θα αναπτύξουμε στη συνέχεια.

Μια γρήγορη τεχνική για να καταλάβουμε αν ένα σύστημα είναι συμμετρικό ή όχι, είναι να ελέγξουμε αν ο πίνακας αποστάσεων των κόμβων είναι συμμετρικός ως προς την πάνω αριστερά διαγώνιο. Αν αυτό ισχύει, τότε οι απόσταση που έχουμε στον πάνω τριγωνικό είναι ίδιες με τις αποστάσεις που έχουμε στον κάτω τριγωνικό πίνακα. Άρα ισχύει ότι:

$$\text{Distance}(i, j) = \text{Distance}(j, i), \forall (i, j)$$

Όπου i και j είναι όλοι οι κόμβοι που διαθέτει το σύστημα μας.

Στην συνέχεια θα δούμε κάποιους όρους που χρησιμοποιήσαμε ευρέως στην διπλωματική εργασία.

Επεκτεταμένη μορφή

Στη έρευνα μας χρησιμοποιήσαμε αρκετά benchmark για να πάρουμε μετρήσεις. Σε κάποια από τα benchmark που χρησιμοποιήσαμε είχαμε περισσότερα από ένα αρχεία εισόδου. Για να μπορούμε να ξεχωρίζουμε τις διαφορετικές εκτελέσεις του ίδιου benchmark με άλλο αρχείο εισόδου κατασκευάσαμε την επεκτεταμένη μορφή. Η επεκτεταμένη μορφή περιέχει πληροφορία για το όνομα του benchmark και το αρχείο εισόδου του. Κατασκευάζουμε την επεκτεταμένη μορφή ως εξής:

"benchmark name"_"number of input file"

Έκδοση εκτέλεσης

Ένας άλλος όρος που χρησιμοποιούμε πολύ συχνά στη διπλωματική είναι η έκδοση. Η έκδοση περιγράφει το τρόπο που έχουμε επιλέξει κάθε φορά να τρέξουμε μια εφαρμογή. Για παράδειγμα έχουμε την περίπτωση που θέλουμε να έχουμε όλη την μνήμη μιας εφαρμογής σε έναν απομακρυσμένο κόμβο και την περίπτωση που θέλουμε να μεταφέρουμε την μνήμη της διεργασίας κατά την ώρα λειτουργίας της. Αυτές οι δύο περιπτώσεις ορίζουν και δύο διαφορετικές εκδόσεις. Για κάθε διαφορετική παράμετρο που θέλουμε να αλλάξουμε στον τρόπο που θα εκτελέσουμε ένα benchmark, αυτό ορίζει και μια νέα έκδοση.

2.2 Σχετική έρευνα

Τα NUMA συστήματα υπάρχουν στα σημερινά κέντρα δεδομένων, καθώς παρέχουν την απαραίτητη υποδομή για scale-up εργασίες. Η παρουσία των NUMA συστημάτων οφείλεται στην ικανότητά τους να παρέχουν μεγάλες ποσότητες μνήμης σε εφαρμογές, πάνω από φυσικά κατανεμημένους κόμβους μνήμης. Ωστόσο, οι πολλαπλές μονάδες μνήμης εισάγουν μεγάλη ποικιλία στις τιμές latency και bandwidth, γεγονός που καθιστά την τοποθέτηση των νημάτων εφαρμογής και μνήμης κρίσιμη για την απόδοση της εφαρμογής.

Η κοινή λογική απαιτεί κυρίως η μνήμη να δεσμεύεται τοπικά, δηλαδή ένα νήμα εφαρμογής και η μνήμη του θα πρέπει να βρίσκεται στον ίδιο κόμβο, λαμβάνοντας επίσης υπόψη το bandwidth και το contention της μνήμης. Ως εκ τούτου, η μέχρι τώρα έρευνα εστιάζονταν στο χαρακτηρισμό και/ή στη μοντελοποίηση της επίδρασης του NUMA συστήματος σε εφαρμογές πολλαπλών νημάτων [22], [23], [24], [27], [28], εστιάζοντας κυρίως στην επιλογή της βέλτιστης τοποθέτησης των πολλαπλών νημάτων της εφαρμογής στους κόμβους NUMA.

Η δουλειά μας σε αυτή την εργασία έχει σαν κίνητρο ένα διαφορετικό προβληματικό σενάριο που μπορεί να προκύψει σε συστήματα NUMA το οποίο επιδιώκει να βελτιώσει την αξιοποίηση και την αποτελεσματικότητα των πόρων του συστήματος. Αντί για την βέλτιστη τοποθέτηση μιας μόνο εφαρμογής πολλαπλών νημάτων που μπορεί να χρησιμοποιήσει όσο το δυνατόν περισσότερους πόρους ή την τοποθέτηση πολλών εφαρμογών που εκτελούνται ταυτόχρονα και συνεπώς χρειάζεται να έχουν τους πυρήνες και τη μνήμη κοντά ο ένας στον άλλο, εμείς στοχεύουμε στην περίπτωση κατά την οποία είναι αδύνατο να παρέχονται όλες οι εφαρμογές με (κυρίως) τοπική μνήμη και κάποιες αναγκαστικά θα εκτελεστούν με απομακρυσμένη μνήμη. Έτσι απαιτείται αποτελεσματική μοντελοποίηση για να επιλεχθούν ποιες εφαρμογές θα πρέπει να εκτελούνται με απομακρυσμένη μνήμη.

Η πολυπλοκότητα ενός NUMA συστήματος εξαρτάται από την αποδοτική τοποθέτηση των νημάτων και της μνήμης από τις εφαρμογές πάνω στο σύστημα για την καλύτερη αξιοποίηση του. Σαν αποτέλεσμα αυτού, έχει γίνει σημαντική έρευνα στην μοντελοποίηση της απόδοσης των NUMA συστημάτων ή εφαρμογών που τρέχουν πάνω σε NUMA συστήματα. Ο Majo και οι συνεργάτες του [23] έχουν αναπτύξει ένα μοντέλο για να χαρακτηρίσουν τον χωρισμό της τοπικής και μακρινής ροής μνήμης σε ένα NUMA σύστημα.

Ο McCormick και οι συνεργάτες του [24] έχουν προτείνει ένα memory-access μοντέλο για να βελτιστοποιήσουν τις αποφάσεις ενός task-scheduling χρονοδρομολογητή ο οποίος χρονοδρομολογεί τις δουλειές κοντά στα δεδομένα τους. Όμως, το μοντέλο εξαρτάται μόνο από χαρακτηριστικά της αρχιτεκτονικής και δεν λαμβάνει υπόψιν τα χαρακτηριστικά των εφαρμογών. Ο Wang και οι συνεργάτες του [28] έχουν προτείνει ένα μοντέλο που μπορεί να προβλέπει την ροή της μνήμης και την βέλτιστη τοποθέτηση της μνήμης σε πολυνηματικές εφαρμογές πάνω σε ένα NUMA σύστημα. Τέλος ο Luo και οι συνεργάτες του [22] έχουν προτείνει ένα συνθετικό μοντέλο για την επιλογή του καλύτερου πυρήνα και μνήμης για την τοποθέτηση των εφαρμογών σε ένα NUMA σύστημα. Όμως, το μοντέλο τους επικεντρώνεται κυρίως σε πολυνηματικές εφαρμογές και χρειάζεται να χρησιμοποιήσουν τεχνικές profiling οι οποίες μπορεί να επηρεάσουν την απόδοση των εφαρμογών.

Οι τεχνικές μηχανικής μάθησης είχαν πολλές εφαρμογές ως εμπειρικές μέθοδοι για την μοντελοποίηση συστημάτων και εφαρμογών σε σχέση με την απόδοση την προηγούμενη δεκαετία. Ενδεικτικά, ο Barnes και οι συνεργάτες του [16] έχουν χρησιμοποιήσει regression με πολλές μεταβλητές για να εξάγει την απόδοση των παράλληλων εφαρμογών σε υψηλότερες μετρήσεις πυρήνα και να προβλέψει την επεκτασιμότητα τους. Ο Shudler και οι συνεργάτες του [26] έχουν χρησιμοποιήσει regression με μια μεταβλητή για την πρόβλεψη της επεκτασιμότητας της απόδοσης με αναπαραγόμενα μη-γραμμικά μοντέλα. Ο Calotoiu και οι συνεργάτες του [16] χρησιμοποίησαν τις ίδιες τεχνικές για αυτοματοποιημένη παράγωγή μοντέλων για να μοντελοποιήσουν την απόδοση από πολύπλοκες παράλληλες εφαρμογές. Η Νικέλα Παπαδοπούλου και οι συνεργάτες της [25] έχουν χρησιμοποιήσει regression με πολλές μεταβλητές για να υπολογίσουν το χρόνο υπολογισμού μια MPI εφαρμογής. Σχετικά με τα NUMA συστήματα, ο Su και οι συνεργάτες του [27] έχουν χρησιμοποιήσει τεχνικές μηχανικής μάθησης για να προβλέψουν το walltime από πολυνηματικές εφαρμογές σε NUMA συστήματα, για διαφορετικά σενάρια τοποθέτησης. Η προσέγγισή τους είναι κάθετη σε σχέση με την δική μας, καθώς η πρόβλεψη για την απόδοση σχετίζεται με την τοποθέτηση της μνήμης αντί των νημάτων.

Κεφάλαιο 3

Ανάλυση Μηχανημάτων

Σε αυτό το κεφάλαιο θα δούμε τα αρχιτεκτονικά χαρακτηριστικά των μηχανημάτων που χρησιμοποιήσαμε για την έρευνα μας. Όλα τα πειράματα τα εκτελέσαμε εξίσου και στα δύο μηχανήματα.

3.1 Sandman

Το πρώτο μηχάνημα που θα αναλύσουμε είναι το μηχάνημα “Sandman”. Το μηχάνημα Sandman αποτελείται από τέσσερις NUMA κόμβους συνολικά. Το μοντέλο επεξεργαστή του κάθε κόμβου είναι το Intel Xeon E5-4620, τέταρτης γενιάς, χρονισμένο στα 2.2GHz. Ο κάθε κόμβος του συστήματος αποτελείται από 8 πυρήνες, που ο καθένας έχει 2 νήματα. Στον πίνακα 3.1 βλέπουμε την αντιστοίχηση των επεξεργαστών και των κόμβων του συστήματος χρησιμοποιώντας το εργαλείο numactl.

Node	**CPUs**															
0	0	1	2	3	4	5	6	7	32	33	34	35	36	37	38	39
1	8	9	10	11	12	13	14	15	40	41	42	43	44	45	46	47
2	16	17	18	19	20	21	22	23	48	49	50	51	52	53	54	55
3	24	25	26	27	28	29	30	31	56	57	58	59	60	61	62	63

Πίνακας 3.1: Επεξεργαστές και κόμβοι του μηχανήματος Sandman.

Ο κάθε πυρήνας του συστήματός μας έχει δύο επίπεδα κρυφής μνήμης, την L1-cache και την L2-cache. Οι δύο κρυφές μνήμες είναι μοιραζόμενες και στα δύο νήματα του πυρήνα. Ακόμα σε κάθε κόμβο έχουμε και ένα τρίτο επίπεδο κρυφής μνήμης την L3-cache ή αλλιώς όπως αναφερόμαστε σε αυτήν στην έρευνα μας LLC (Last Level Cache). Η LLC είναι μοιραζόμενη σε όλους τους πυρήνες του κάθε κόμβου. Στο πίνακα 3.2 βλέπουμε το μέγεθος της κάθε κρυφής μνήμης.

Cache	**Size**	**Shared**
L1	32 KB	Per core
L2	256 KB	Per core
L3 (LLC)	16 MB	Per numa node

Πίνακας 3.2: Μεγέθη των κρυφών μνημών του μηχανήματος Sandman.

Ο κάθε κόμβος του συστήματος είναι συνδεδεμένος με την δίκια του μνήμη. Στον πίνακα 3.3 βλέπουμε το μέγεθος της μνήμης που διαθέτει ο κάθε κόμβος. Ο κάθε κόμβος είναι διασυνδεδεμένος με κάθε άλλο κόμβο του συστήματος μέσω του Intel QuickPath Interconnect. Οι αποστάσεις που απέχει ο κάθε κόμβος από κάθε άλλο κόμβο φαίνονται στον πίνακα 3.4. Για να κατασκευάσουμε αυτούς τους δύο πίνακες χρησιμοποιήσαμε το εργαλείο numactl.

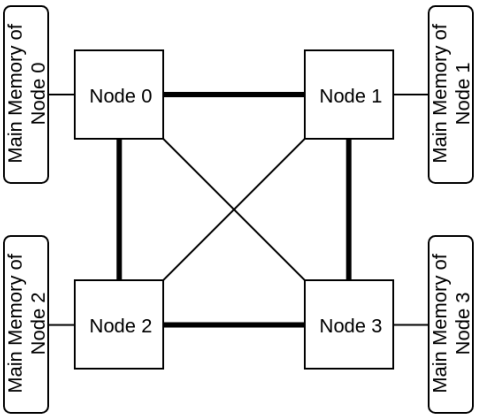
Node	Memory (MB)
0	64404
1	64509
2	64509
3	64508

Πίνακας 3.3: Μνήμη που είναι συνδεδεμένη στους κόμβους του Sandman.

	DST 0	DST 1	DST 2	DST 3
SRC 0	10	21	21	30
SRC 1	21	10	30	21
SRC 2	21	30	10	21
SRC 3	30	21	21	10

Πίνακας 3.4: Αποστάσεις κόμβων του μηχανήματος Sandman.

Παρατηρούμε ότι ο πίνακας αποστάσεων είναι συμμετρικός ως προς την πάνω αριστερά διαγώνιο. Άρα συμπεραίνουμε ότι το μηχάνημα Sandman είναι συμμετρικό. Στο σχήμα 3.1 βλέπουμε την γραφική αναπαράσταση του μηχανήματος Sandman. Οι πιο παχιές γραμμές του σχήματος δηλώνουν την ταχύτερη επικοινωνία μεταξύ των κόμβων του συστήματος, όπως προκύπτει από τον πίνακα 3.4.



Σχήμα 3.1: Γραφική αναπαράσταση του μηχανήματος Sandman.

3.2 Broady

Το άλλο μηχάνημα που χρησιμοποιήσαμε στην έρευνα μας είναι το μηχάνημα “Broady”. Το μηχάνημα Broady αποτελείται από δύο κόμβους συνολικά. Το μοντέλο του κάθε κόμβου είναι το Intel Xeon E5-2699 v4, έκτης γενιάς (δύο γενιές νεότερο σε σχέση με το άλλο μηχάνημα), χρονισμένο στα 2.2GHz. Ο κάθε κόμβος του συστήματος αποτελείται από 22 πυρήνες, όπου ο καθένας έχει 2 νήματα. Στον πίνακα 3.5 βλέπουμε την αντιστοίχηση των επεξεργαστών και των κόμβων του συστήματος όπως τις πήραμε από το εργαλείο numactl.

Node	**CPUs**										
0	0	1	2	3	4	5	6	7	8	9	10
	11	12	13	14	15	16	17	18	19	20	21
	44	45	46	47	48	49	50	51	52	53	54
	55	56	57	58	59	60	61	62	63	64	65
1	22	23	24	25	26	27	28	29	30	31	32
	33	34	35	36	37	38	39	40	41	42	43
	66	67	68	69	70	71	72	73	74	75	76
	77	78	79	80	81	82	83	84	85	86	87

Πίνακας 3.5: Επεξεργαστές και κόμβοι του μηχανήματος Broady.

Τα δύο μηχανήματα έχουν παρόμοια ιεραρχία στην κρυφή μνήμη. Ο κάθε πυρήνας του συστήματός μας έχει δύο επίπεδα κρυφής μνήμης, την L1-cache και την L2-cache. Οι δύο κρυφές μνήμες είναι μοιραζόμενες και στα δύο νήματα του πυρήνα. Ακόμα σε κάθε κόμβο έχουμε και ένα τρίτο επίπεδο κρυφής μνήμης την L3-cache ή αλλιώς LLC . Η LLC είναι μοιραζόμενη σε όλους τους πυρήνες του κάθε κόμβου. Στο πίνακα 3.6 βλέπουμε το μέγεθος της κάθε κρυφής μνήμης.

Cache	**Size**	**Shared**
L1	32 KB	Per core
L2	256 KB	Per core
L3 (LLC)	56 MB	Per numa node

Πίνακας 3.6: Μεγέθη των κρυφών μνημών του μηχανήματος Broady.

Ο κάθε κόμβος του συστήματος είναι συνδεδεμένος με την δίκια του μνήμη. Στον πίνακα 3.7 βλέπουμε το μέγεθος της μνήμης που διαθέτει ο κάθε κόμβος. Οι δύο κόμβοι του συστήματος είναι συνδεμένοι μεταξύ τους. Οι αποστάσεις που απέχουν οι δύο κόμβοι είναι στο πίνακα 3.8. Για να κατασκευάσουμε αυτούς τους δύο πίνακες χρησιμοποιήσαμε το εργαλείο numactl.

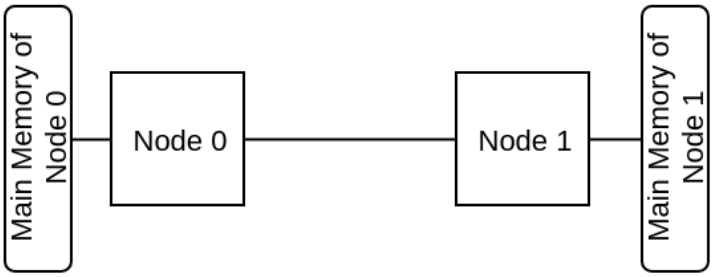
Node	**Memory (MB)**
0	257843
1	258033

Πίνακας 3.7: Μνήμη που είναι συνδεδεμένη στους κόμβους του Broady.

Καθώς παρατηρούμε τον πίνακα αποστάσεων βλέπουμε ότι ο πίνακας είναι συμμετρικός ως προς την πάνω αριστερά διαγώνιο. Άρα συμπεραίνουμε ότι το μηχάνημα Broady

	DST 0	**DST 1**
SRC 0	10	21
SRC 1	21	10

Πίνακας 3.8: Αποστάσεις κόμβων του μηχανήματος Broady.



Σχήμα 3.2: Γραφική αναπαράσταση του μηχανήματος Broady.

είναι συμμετρικό. Στο σχήμα 3.2 βλέπουμε την γραφική αναπαράσταση του μηχανήματος Broady.

Κεφάλαιο 4

Βασικές Μεθοδολογίες

Σε αυτό το κεφάλαιο θα ασχοληθούμε με τα αρχικά ζητήματα που μας προβλημάτισαν στην έρευνα μας. Θα δούμε κάποια βασικά προβλήματα που είχαμε καθώς και τον τρόπο που τα αντιμετωπίσαμε για να θέσουμε τα θεμέλια για την έρευνα μας. Επίσης θα αναλύσουμε τα εργαλεία που χρησιμοποιήσαμε για να πραγματοποιήσουμε την έρευνα μας. Οι μέθοδοι σε αυτό το κεφάλαιο θεωρούνται δομικός λίθος για όλη την έρευνα που πραγματοποιήθηκε.

4.1 Επόπτης

Για να μπορέσουμε να εκτελέσουμε όλα τα πειράματα μας με σαφή και κομψό τρόπο κατασκευάσαμε τον επόπτη. Ο επόπτης είναι ένα εργαλείο που μπορούμε να το χρησιμοποιήσουμε και στα δύο μηχανήματα μας για να εκτελέσουμε οποιοδήποτε πείραμα θέλαμε. Μας δίνει την δυνατότητα να μην χρειάζεται να γράφουμε πολύπλοκες εντολές για την κάθε εκτέλεση ενός πειράματος και διαδικασίες οι οποίες είναι ίδιες και για τα δύο μηχανήματα, τις κάνει αυτόματα. Για να πετύχει όλες τις λειτουργίες που θέλουμε, κατασκευάζει κάποιες νέες διεργασίες για τις οποίες είναι υπεύθυνος ώστε να ελέγχει ότι λειτουργούν σωστά.

Το πρώτο πρόβλημα που χρειάστηκε να αντιστοιχίσουμε ήταν ο συντονισμός όλων των διεργασιών, που θα έπαιρναν μετρήσεις πάνω στην εκάστοτε εφαρμογή που θέλαμε. Για να πετύχουμε τον συντονισμό των διεργασιών, κατασκευάσαμε τον επόπτη. Ο επόπτης βρίσκεται στην καρδιά όλων των πειραμάτων μας, καθώς είναι υπεύθυνος για την σωστή λειτουργιά της κάθε εκτέλεσης.

4.1.1 Οι βασικές λειτουργίες του επόπτη

Οι βασικές λειτουργίες που χρειάστηκε να επιτελέσει ο επόπτης είναι:

- Διάβασμα παραμέτρων από τα αρχεία εισόδου
- Επεξεργασία των παραμέτρων
- Ο συντονισμός των διεργασιών με την εφαρμογή

Διάβασμα παραμέτρων από τα αρχεία εισόδου: Για το κάθε πείραμα χρειαζόμαστε πληθώρα από αρχεία εισόδου καθώς και ορίσματα. Αρχίζοντας με ορίσματα που δέχονταν ο επόπτης:

- *Όρισμα 1:* Ένας θετικός ακέραιος, ο οποίος αντιστοιχεί με μια λίστα. Η λίστα αυτή είναι γραμμένη σε φυσική γλώσσα και παρέχει στο χρήστη συνοπτικές πληροφορείς για την εκάστοτε έκδοση που τρέχουμε.
- *Όρισμα 2:* Ένας ακέραιος, ο οποίος χρησιμοποιούμε σαν αναγνωριστικό από τον επόπτη. Αυτό το όρισμα έχει νόημα όταν θέλουμε να τρέξουμε πάνω από μια εφαρμογές στο σύστημα μας για να δούμε πως συμπεριφέρονται μεταξύ τούς. Το εύρος των τιμών που παίρνει είναι $[0, n)$, όπου n είναι το μέγιστο πλήθος εφαρμογών που θέλουμε να τρέξουμε ταυτόχρονα. Κάθε εφαρμογή που τρέχει έχει δικό της μοναδικό αναγνωριστικό μέσα στο παραπάνω εύρος. Η εφαρμογή με το αναγνωριστικό 0 είναι εκείνη που μετράμε.
- *Όρισμα 3:* Το όνομα του benchmark που θέλουμε να μετρήσουμε σε επεκτεταμένη μορφή.
- *Όρισμα 4:* Το όνομα της σουίτας από την οποία πήραμε το benchmark (Parsec, Spec).

Τα αρχεία εισόδου επειδή ήταν πολλά τα ενσωματώσαμε όλα σε ένα. Το αρχείο αυτό παρέχει τις εξής πληροφορίες:

- Τον αριθμό των νημάτων που θα χρησιμοποιήσει το κάθε benchmark.
- Την μέθοδο με την οποία θα τοποθετήσουμε τα benchmark πάνω στα cpu.
- Το ποσοστό του χρόνου που θα κάνουμε την μεταφορά μνήμης από έναν κόμβο σε έναν άλλο.
- Τον αριθμό των κόμβων που διαθέτει το σύστημα.
- Μια προ-επεξεργασμένη μορφή του numactl –hardware, έτσι ώστε να είναι πιο εύκολα διαχωρίσιμη από τον επόπτη. Η τελική μορφή είναι ένα πίνακας. Ο πίνακας περιέχει πληροφορία για την τοποθεσία των cpu σε σχέση με τους κόμβους του συστήματος.
- Μια λίστα με τους επιθυμητούς κόμβους που θέλουμε να μελετήσουμε. Ο κόμβος 0 είναι πάντα ο πρώτος κόμβους μελέτης (δεν περιλαμβάνετε σε αυτή την λίστα).
- Μια λίστα με ποσοστά που χρησιμοποιείται μόνο από την movepages, όποτε τις περισσότερες φορές είναι κενή.
- Μια συμβολική λίστα με τους performance counters που θέλουμε να μετρήσουμε. Η συμβολική λίστα βρίσκετε στον πίνακα 4.3

Επεξεργασία των παραμέτρων: Το επόμενο βήμα που χρειάζεται να κάνει ο επόπτης είναι να επεξεργαστεί όλες τις παραμέτρους που έχει διαβάσει. Αυτό το κάνει σε 2 στάδια, το πρώτο είναι κατά την εκκίνηση και το δεύτερο είναι στο τέλος κάθε κύκλου επανάληψης του.

Κατά την εκκίνηση, η πρώτη λειτουργιά που έχουμε να κάνουμε σε αυτό το στάδιο, είναι να βρούμε που πρέπει να τοποθετήσουμε την κάθε εφαρμογή που θέλουμε να χρησιμοποιήσουμε. Αυτό το κάνουμε χρησιμοποιώντας το όρισμα 2, τον αριθμό των νημάτων, την μέθοδο τοποθέτησης και την προ-επεξεργασμένη μορφή του numactl –hardware. Στα πειράματα που πραγματοποιήσαμε, χρησιμοποιήσαμε 2 μεθόδους τοποθέτησης:

1. Αρχίζουμε και τοποθετούμε τα benchmark το ένα μετά την άλλο στα cpu, με την σειρά που μας υποδηλώνουν τα αναγνωριστικά. Αφήνουμε τόσα cpu όσα και ο αριθμός των νημάτων που θα χρησιμοποιήσει η κάθε εφαρμογή. Αυτό το κάνουμε μέχρι να γεμίσει ο κάθε αρχικός μας κόμβος (κόμβος 0). Στην συνέχεια πάμε στον επόμενο κόμβο (κόμβος 1) όπου επαναλαμβάνουμε την ίδια διαδικασία.
2. Τοποθετούμε τα benchmark εναλλάξ μεταξύ των κόμβων 0 και 1, πάλι όπως μας υποδεικνύουν τα αναγνωριστικά. Έτσι καταλήγουμε να έχουμε στον κόμβο 0 τα

benchmark που έχουν ζυγό αναγνωριστικό και στον κόμβο 1 αυτά που έχουν μονό. Πάλι αφήνουμε τόσα cpu ελεύθερα όσα και ο αριθμός των νημάτων και αυτό το επαναλαμβάνουμε μέχρι να γεμίσουν οι κόμβοι 0 και 1.

Και στις 2 παραπάνω περιπτώσεις που μελετήσαμε λάβαμε υπόψιν μας το hyper threading. Ακόμα πρέπει να τονίσουμε ότι έχουμε περιορισμό για το πόσα benchmark μπορούμε να τρέξουμε μαζί, με αυτές τις μεθόδους χωρίς να έχουμε επικαλύψεις στα νήματα. Οι παρακάτω τύποι μας λένε τον μέγιστο αριθμό από benchmark που μπορούμε να τρέξουμε ταυτόχρονα:

$$n_1 = 2\frac{\text{number of cpus per node}}{\text{hyper threading}}$$

$$n_2 = 2\frac{\text{number of cpus per node}}{\text{number of threads}}$$

Το n_1 αφορά μονονηματικά benchmark και το n_2 αφορά πολυνηματικά benchmark όπου ο αριθμός των νημάτων είναι ακέραιο πολλαπλάσιο του hyper threading. Αφού βρούμε σε ποια cpu θα τοποθετήσουμε τα benchmark μας ενδιαφέρει να βρούμε που θα τοποθετήσουμε τις επιπλέον διεργασίες που θα δημιουργήσει ο επόπτης για την κάθε μέτρηση. Αυτό το βήμα είναι αρκετά σημαντικό, γιατί πρέπει να τοποθετήσουμε τις επιπλέον διεργασίες όσο πιο μακριά γίνεται από τα benchmark έτσι ώστε να έχουμε όσο το δυνατόν πιο καθαρές μετρήσεις (απαλλαγμένες από το θόρυβο και τους επιπλέον πόρους που θα ζητάνε οι επιπλέον διεργασίες). Για να το πετύχουμε αυτό τοποθετούμε στον πιο απομακρυσμένο κόμβο που μπορούμε (για το Sandman είναι ο κόμβος 3 και για το Broady είναι ο κόμβος 1) αρχίζοντας από τα τα τελευταία cpu και πηγαίνοντας προς τα πρώτα. Εδώ θέλουμε να έχουμε επικαλύψεις στα νήματα για να εξοικονομήσουμε όσο το δυνατόν περισσότερους πόρους. Οπότε για όλες τις επιπλέον διεργασίες που θα δημιουργήσει ο επόπτης του δίνουμε 1 cpu. Όσον αφορά τον επόπτη δεν μπορούμε να τον δέσουμε σε ένα συγκεκριμένο cpu, γιατί μετά δεν μας επιτρέπεται να δέσουμε κανένα από τα παιδιά που δημιουργεί σε κάποιο άλλο cpu.

Έπειτα από την επεκτεταμένη μορφή του ονόματος παίρνουμε την μη επεκτεταμένη. Αυτό το κάνουμε ψάχνοντας για το διαχωριστικό "_". Έτσι χωρίζουμε την επεκτεταμένη μορφή στο πραγματικό όνομα του benchmark και στο input που πάρει το benchmark.

Στην συνέχεια υπολογίζουμε τον αριθμό κύκλων επανάληψης που κάνει ο επόπτης για να πάρει όλες τις μετρήσεις που του έχουμε ζητήσει. Ο τύπος που μας δίνει τον αριθμό των επαναλήψεων είναι ο παρακάτω:

$$\text{loop} = (\text{perf\_cnt})[1 + (\text{dest\_nodes})(\text{percent})]$$

Όπου το perf_cnt είναι ο αριθμός των performance counters, το dest_nodes είναι ο αριθμός των επιθυμητών κόμβων που θέλουμε να μετρήσουμε και το percent είναι το μέγεθος της λίστας με τα ποσοστά για την movepages. Αν δεν θέλουμε να μελετήσουμε την συμπεριφορά της movepages τότε η τιμή του percent τίθεται στο 1.

Στο τέλος του πρώτου κύκλου επανάληψης ελέγχουμε τον χρόνο που χρειάστηκε το benchmark για να ολοκληρώσει την εκτέλεση του. Ανάλογα με το ποια εκδοσή τρέχουμε, χρησιμοποιούμε αυτό το χρόνο για να πούμε στις διεργασίες πότε χρειάζεται να κάνουν διάφορες λειτουργίες πάνω στην μνήμη του benchmark.

Ο συντονισμός των διεργασιών με την εφαρμογή: Εδώ ο επόπτης χρειάζεται να συντονίσει σωστά όλες τις διεργασίες που μετράνε στοιχειά του benchmark και κάνουν λειτουργίες πάνω στην μνήμη του με το ίδιο το benchmark. Μπορούμε να χωρίσουμε τις διεργασίες σε 3 δομικές μονάδες:

- Ο εκκινητής του benchmark και του perf stat.
- Ο παρακολουθητής μνήμης.
- Ο μεταφορέας μνήμης.

Η ακριβής λειτουργία της κάθε δομικής μονάδας αναλυθεί περαιτέρω στο επόμενο υποκεφάλαιο, εδώ θα δούμε πως επιτυγχάνουμε τον συντονισμό μεταξύ τους.

Για κάθε δομική μονάδα ο επόπτης δημιουργεί μια νέα διεργασία. Οι διεργασίες δημιουργούνται με την σειρά που περιγράφουμε τις δομικές μονάδες. Αφού ο εκκινητής βάλει το benchmark να τρέχει, βρίσκουμε το pid του. Στην συνέχεια δημιουργούμε τον παρακολουθητή, στον οποίο δίνουμε το pid το benchmark που θέλουμε να κοιτάει. Εδώ έχουμε περίπου 2 με 3 δευτερόλεπτα καθυστέρηση μεταξύ του παρακολουθητή και του εκκινητή. Αυτή η καθυστέρηση οφείλετε στην δημιουργία του παρακολουθητή καθώς και στον συντονισμό που χρειάζονται οι 2 διεργασίες. Έπειτα ανάλογα με την έκδοση που τρέχουμε εκκινούμε τον μεταφορέα.

Αφού έχουμε δημιουργήσει όλες τις διεργασίες ο επόπτης πέφτει για ύπνο. Ξυπνάει όταν το benchmark έχει τελειώσει με την εκτέλεση του, όπου αρχίζει και μαζεύει όλες τις διεργασίες που είχε δημιουργήσει. Τέλος αυξάνει τους κατάλληλους μετρητές για να πάει στο επόμενο κύκλο επανάληψης, όπου επαναλαμβάνει την ίδια διαδικασία.

4.1.2 Οι δομικές μονάδες του επόπτη

Εδώ θα περιγράψουμε τις λειτουργίες που κάνει κάθε δομική μονάδα. Υπενθυμίζουμε σε αυτό το κομμάτι ότι η διεργασία που θα μετρήσουμε είναι εκείνη που έχει αναγνωριστικό (όρισμα 2) 0.

Ο εκκινητής: Η βασικότερη δουλειά που έχει να κάνει ο εκκινητής είναι να ξεκινήσει το benchmark που του έχουμε ζητήσει, όμως δεν είναι η μόνη δουλειά που κάνει.

Κάνοντας χρήση του numactl θέτει τα cpu που θα χρησιμοποιήσει το κάθε benchmark. Ακόμα σε περίπτωση που του έχουμε ζητήσει να αρχίζει το benchmark με απομακρυσμένη μνήμη το κάνει πάλι μέσω του numactl.

Στην συνέχεια κάνοντας χρήση του perf stat βρίσκει τους performance counters που του έχουμε ζητήσει να μετρήσει. Τα αποτελέσματα για τους performance counters μας τα δίνει σε ένα αρχείο εξόδου που του έχουμε ορίσει, κάθε 1 δευτερόλεπτο. Εδώ δεν έχουμε το πρόβλημα του παραθύρου καθώς η μέτρηση των performance counters είναι συνεχόμενη. Σε επόμενο κεφάλαιο θα μελετήσουμε περισσότερο τούς performance counters και τον τρόπο με τον οποίο παίρνουμε μετρήσεις.

Τέλος βρίσκουμε την τοποθεσία του benchmark καθώς και του αρχείο εισόδου του και αρχίζουμε την εκτέλεση του. Από την στιγμή που ο εκκινητής ξεκινήσει το benchmark ολοκληρώνει η εκτέλεση του. Αυτό το κάνουμε για να επηρεαστούν οι μετρήσεις μας από τον εκκινητή.

Ο παρακολουθητής: Είναι υπεύθυνος για την καταγραφή της μνήμης που χρησιμοποιεί το benchmark. Για να το κάνει αυτό η μόνο πληροφορία που χρειάζεται είναι το pid του benchmark. Την καταγραφή της μνήμης την κάνουμε μέσω του numa_maps.

Κάθε 1 δευτερόλεπτο αντιγράφουμε την παρούσα κατάσταση από το numa_maps σε ένα αρχείο εξόδου, μέχρι να ολοκληρωθεί το benchmark. Μετά το πέρας της εκτέλεσης των πειραμάτων θα επεξεργαστούμε κατάλληλα αυτά τα αρχεία για να δημιουργήσουμε μια εικόνα για την συμπεριφορά της μνήμης του benchmark. Αυτή η μέθοδος έχει το πρόβλημα ότι δεν μπορεί να εντοπίσουμε αλλαγές στην μνήμη που συμβαίνουν εντός του παραθύρου του ενός δευτερολέπτου.

Ο μεταφορέας: Η λειτουργία που έχει να κάνει είναι να μεταφέρει την μνήμη από έναν κόμβο σε έναν άλλον. Ο κόμβος αναχώρησης είναι πάντα ο 0, ο κόμβος άφιξης, η μέθοδος μεταφοράς που θα χρησιμοποιήσουμε καθώς και η χρονική στιγμή που θα κάνουμε την μεταφορά εξαρτώνται από την έκδοση που τρέχουμε. Ακόμα ανάλογα με ποια έκδοση θέλουμε να τρέξουμε ο μεταφορέας μπορεί να μην χρειάζεται, όπως όταν θέλουμε να δούμε την συμπεριφορά του benchmark με εξαρχής όλη την μνήμη απομακρυσμένη.

Ο μεταφορέας καλείται μια φορά να μεταφέρει την μνήμη σε κάποιο απομακρυσμένο κόμβο κατά την διάρκεια της εκτέλεσης. Αυτό έχει σαν αποτέλεσμα ότι, αν το benchmark δεσμεύσει νέα μνήμη αυτή δεν θα πάει στο απομακρυσμένο κόμβο άλλα στον τοπικό. Η μεταφορά μνήμης γίνεται είτε μέσω της migratepages είτε μέσω της movepages, τις οποίες τις αναλύουμε σε επόμενο κεφάλαιο.

4.1.3 Επηρεασμός των μετρήσεων λόγω του επόπτη

Ένας προβληματισμός που είχαμε είναι αν ο επόπτης και οι διεργασίες που δημιουργεί επηρεάζουν άμεσα τις μετρήσεις μας. Για να ελαχιστοποιήσουμε αυτόν τον παράγοντα κάναμε όλες τις διεργασίες καθώς και τον επόπτη να κοιμούνται όσον χρόνο δεν κάνουν κάποια λειτουργία. Ακόμα οι ίδιες οι λειτουργίες που χρειάζεται να κάνουν οι διεργασίες πρέπει να είναι αρκετά απλές, έτσι ώστε να μην έχουν μεγάλα κομμάτια υπολογισμού και καταναλώνουν πόρους από το σύστημα.

Όλες οι διεργασίες κοιμούνται και ξυπνάνε μέσω τις κλήσης waitpid[15]. Η κλήση waitpid δέχεται σαν όρισμα ένα pid. Όταν μια διεργασία καλέσει την συνάρτηση με ένα συγκεκριμένο pid έστω P τότε, πέφτει για ύπνο. Διατηρείται σε αυτήν την κατάσταση μέχρι να τερματίσει η διεργασία που είχε το pid P. Έτσι ο επόπτης καλεί τόσες φορές αυτήν την συνάρτηση με τα pids των παιδιών του για να βεβαιωθεί ότι έχουν τελειώσει την λειτουργία τους.

4.2 Πρόβλημα της τοπολογίας

Ένα άλλο ζήτημα που μας απασχόλησε είναι ο τρόπος με τον οποίο θα προσεγγίσουμε ένα τόσο πολύπλοκο σύστημα, για να το μελετήσουμε. Ένα NUMA σύστημα αποτελείται από πολλά στοιχεία που είναι συνδεδεμένα μεταξύ τους όπως επεξεργαστές, μνήμες, ελεγκτές και άλλα. Όπως παρατηρούμε ένα τέτοιο σύστημα κρύβει μεγάλη πολυπλοκότητα μέσα του, όποτε ο τρόπος με τον οποίο θα το προσεγγίσουμε είναι καθοριστικός.

4.2.1 Ανάλυση της πολυπλοκότητας ενός συστήματος NUMA

Ένα από τα κύρια χαρακτηριστικά είναι ο τρόπος που είναι οργανωμένοι οι επεξεργαστές και η σχέση αυτών με την μνήμη. Οι επεξεργαστές είναι οργανωμένοι σε ομάδες, τους κόμβους. Κάθε επεξεργαστής ανήκει μόνο σε ένα κόμβο. Ο κάθε κόμβος είναι συνδεδεμένος με την δικιά του μνήμη, καθώς και με άλλους κόμβους.

Για να μελετήσουμε την συμπεριφορά των εφαρμογών που θέλουμε πάνω σε ένα τέτοιο σύστημα χρειάζεται να πάρουμε μετρήσεις πάνω σε αυτό. Όμως ο τρόπος με τον οποίο θα πάρουμε αυτές τις μετρήσεις δεν είναι πολύ προφανής λόγω της πολυπλοκότητας του συστήματος. Ας χρησιμοποιήσουμε σαν παράδειγμα το Sandman, για να γίνει πιο κατανοητό το πρόβλημα. Έστω ότι θέλουμε να πάρουμε μετρήσεις για το benchmark A. Για χάρη απλότητας ας θεωρήσουμε ότι το A είναι μονονηματικό και ότι θα του δώσουμε

μνήμη μόνο από έναν κόμβο. Έχουμε 64 πιθανές τοποθετήσεις του Α πάνω στο σύστημα, μια για κάθε επεξεργαστή. Ακόμα το Α μπορεί να δεσμεύσει μνήμη μόνο από έναν από τους τέσσερις κόμβους που έχει το σύστημα. Άρα συνολικά έχουμε 256 δυνατούς τρόπους για να τρέξουμε το Α.

Λόγω περιορισμών που έχουν τα εργαλεία που χρησιμοποιούμε, σε κάθε εκτέλεση μπορούμε να μετράμε μια με δυο παραμέτρους του κάθε benchmark, το πολύ. Θα αναλύσουμε αυτό το ζήτημα σε επόμενο κεφάλαιο. Οπότε στο προηγούμενο παράδειγμα μας αν θέλαμε να μετρήσουμε 5 παραμέτρους σε όλες τις δυνατές εκτελέσεις, Θα έπρεπε να τρέχαμε 768 φορές το Α. Προφανώς αυτό ο αριθμός εκτελέσεων είναι υπερβολικά μεγάλος για να πάρουμε μετρήσεις μόνο από ένα benchmark.

Ο αριθμός των δυνατών εκτελέσεων αυξάνει καθώς αρχίζουν να μας απασχολούν και άλλα ζητήματα. Όπως είναι οι πολυνηματικές εφαρμογές. Συγκεκριμένα για μια εφαρμογή με μόνο 2 νήματα έχουμε στο Sandman 2016 δυνατές τοποθετήσεις. Ακόμα ένας παράγοντας που διογκώνει ακόμα το πρόβλημα είναι οι εφαρμογές που παίρνουν μνήμη από δύο κόμβος και πάνω. Επίσης ο αριθμός των διαφορετικών διεργασιών παίζει σημαντικό ρόλο καθώς αυτές επηρεάζουν με διαφορικό τρόπο την εφαρμογή που μετράμε ανάλογα με την τοποθέτηση τους και την μνήμη τούς.

Όπως φαίνεται το παράδειγμα που μελετήσαμε είναι μια πολύ απλουστευμένη μορφή του γενικού προβλήματος. Ο αριθμός των δυνατών εκτελέσεων μας ενδιαφέρει για να ξέρουμε πόσες μετρήσεις χρειάζεται να πάρουμε σε ένα σύστημα για να έχουμε εικόνα για αυτό. Όμως όλες αυτές οι δυνατές εκτελέσεις δεν μας δίνουν όλες χρήσιμη πληροφορία που μπορούμε να επεξεργαστούμε.

4.2.2 Αξιοποίηση πληροφορία

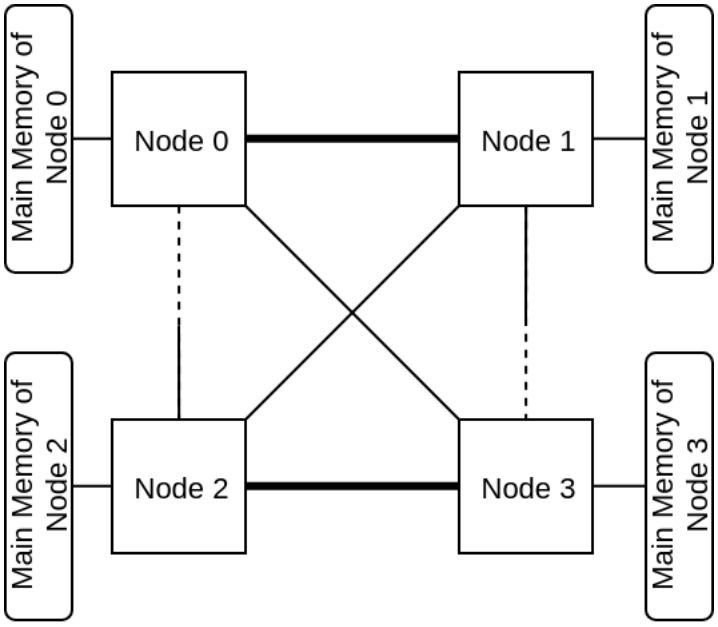
Όπως είδαμε στο προηγούμενο υποκεφάλαιο οι δυνατές εκτελέσεις για ένα benchmark είναι αρκετά σημαντικές για να κατανοήσουμε την συμπεριφορά του συστήματος που μελετάμε. Όμως πολλές από τις δυνατές εκτελέσεις δεν μας δίνουν νέες πληροφορίες για την συμπεριφορά του συστήματος. Εδώ θα μελετήσουμε ποιες από τις δυνατές εκτελέσεις μας δίνουν χρήσιμη νέα πληροφορία. Ως χρήσιμη νέα πληροφορία θεωρούμε, εκείνη την πληροφορία όπου μας αποκαλύπτει χαρακτηριστικά του συστήματος που δεν έχουμε δει μέχρι στιγμής ή φανερώνουν μια διαφορετική συμπεριφορά από την ήδη υπάρχουσα.

Για την καλύτερη κατανόηση μας θα χρησιμοποιήσουμε σαν παράδειγμα το σύστημα του σχήματος 4.1. Αποτελείται από τέσσερις κόμβος όπου ο κάθε κόμβος έχει την δικιά του μνήμη. Ο κάθε κόμβος αποτελείται από 4 πυρήνες. Ο κάθε πυρήνας έχεις 2 επεξεργαστές όπως φαίνεται στο σχήμα 4.2. Όπως παρατηρούμε από το σχήμα το σύστημα είναι ασύμμετρο. Όλοι οι κόμβοι καθώς και όλες οι μνήμες του συστήματος είναι ίδιες. Οι αποστάσεις των κόμβων του συστήματος φαίνονται στον πίνακα 4.1. Η πρώτη στήλη του πίνακα είναι οι κόμβοι αναχώρησης και η πρώτη γραμμή είναι οι κόμβοι προορισμού.

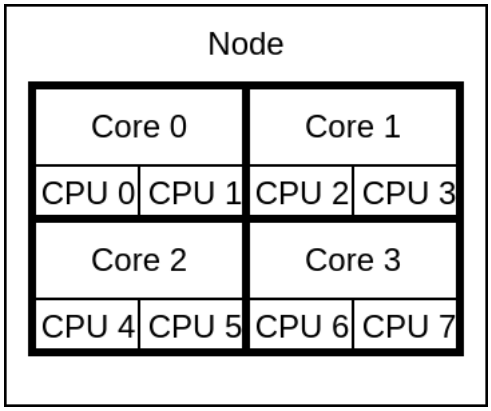
Το σύστημα του σχήματος 4.1 παρόλο που είναι μια ειδική μορφή συστήματος μπορεί πολύ εύκολα να περιγράψει μια γενική μορφή ενός NUMA συστήματος. Οπότε για να πάρουμε όσο το δυνατόν περισσότερη αξιοποιήσιμη πληροφορία κάναμε τις παρακάτω παραδοχές. Οι παραδοχές που κάνουμε εδώ ισχύουν τόσο για το παράδειγμα που μελετάμε όσο και για τα μηχανήματα που ελέγξαμε (Sandman, Broady).

1. Το σύστημα αποτελείται από παρόμοια βασικά στοιχεία.
2. Η εκτέλεση ενός benchmark είναι ανεξάρτητη του κόμβου.
3. Τα benchmark δεν επηρεάζονται από την τοποθέτηση των νημάτων στον κόμβο.
4. Οι ίδιες αποστάσεις επηρεάζουν με τον ίδιο τρόπο το benchmark.

5. Τα νήματα και η μνήμη είναι ανταλλάξιμα.



Σχήμα 4.1: Ασύμμετρο NUMA σύστημα με τέσσερις κόμβους.



Σχήμα 4.2: Κόμβος του συστήματος 4.1.

	DST 0	DST 1	DST 2	DST 3
SRC 0	-	10	40	20
SRC 1	10	-	20	30
SRC 2	30	20	-	10
SRC 3	20	40	10	-

Πίνακας 4.1: Αποστάσεις κόμβων σχήματος 4.1.

Το σύστημα αποτελείται από παρόμοια βασικά στοιχεία: Ως βασικά στοιχεία θεωρούμε τους επεξεργαστές, τις μνήμες, τους δίαυλου επικοινωνίας και τα λοιπά. Αυτή η παραδοχή ισχύει στην πλειονότητα των NUMA συστημάτων. Οπότε μπορούμε να δούμε μακροσκοπικά το σύστημα μας ότι αποτελείται από 4 ίδιους κόμβους οι οποίοι είναι συνδεδεμένοι μετάξι τους.

Η εκτέλεση ενός benchmark είναι ανεξάρτητη του κόμβου: Εδώ αναφερόμαστε σε εκτελέσεις όπου τα νήματα και η μνήμη βρίσκονται στον ίδιο κόμβο. Αυτή η παραδοχή είναι λογική συνέπεια τις πρώτης. Οπότε αν εκτελέσουμε το benchmark A στον κόμβο 0 και στον κόμβο 1 θα πάρουμε τις ίδιες μετρήσεις. Προφανώς το μηχάνημα μας πρέπει να βρίσκετε στην ίδια κατάσταση και στις δύο εκτελέσεις.

Τα benchmark δεν επηρεάζονται από την τοποθέτηση των νημάτων στον κόμβο: Ας θεωρήσουμε αρχικά ότι έχουμε μόνο ένα μονονηματικό benchmark που θέλουμε να τρέξουμε και ότι δεν έχουμε άλλα προγράμματα που μας επηρεάζουν. Μπορούμε να το τοποθετήσουμε οπουδήποτε αυτό το benchmark σε κάποιον κόμβο του συστήματος και θα πάρουμε τις ίδιες μετρήσεις. Αυτό συμβαίνει γιατί το benchmark δεν μοιράζεται πόρους με κάποια άλλη διεργασία.

Ας επεκτείνουμε τώρα το πρόβλημα σε 2 μονονηματικές διεργασίες. Αν αυτές οι διεργασίες δεν μοιράζονται L1-cache, L2-cache, νήματα, δηλαδή αν μπορούμε να θεωρήσουμε ότι τρέχουν σε διαφορετικούς πυρήνες, τότε πάλι θα πάρουμε τις ίδιες μετρήσεις για τα benchmark ανεξάρτητα από το σε ποιους πυρήνες τρέχουν. Δεν εννοούμε ότι τα δύο benchmark δεν θα επηρεάσει το ένα το άλλο, αλλά ότι ο τρόπος με τον οποίο θα επηρεαστούν θα είναι ο ίδιος. Δηλαδή αν τρέξουμε το benchmark A στο core 0 και το benchmark B στο core 1 θα πάρουμε τις ίδιες μετρήσεις με οποιωνδήποτε άλλο συνδυασμό $\{A, B\} \times \{0, 1, 2, 3\}$ όσο τα δύο benchmark δεν βρίσκονται στο ίδιο core.

Η παραπάνω παραδοχή επεκτείνεται και για n benchmarks σε n cores όταν τρέχουν στον ίδιο κόμβο. Αντίστοιχα πράγματα ισχύουν και για τα πολυνηματικά benchmark όταν τα νήματα τοποθετούνται το ένα δίπλα στο άλλο και τα benchmark μεταξύ τους δεν μοιράζονται πυρήνες.

Οι ίδιες αποστάσεις επηρεάζουν με τον ίδιο τρόπο το benchmark: Έστω ότι τρέχουμε το benchmark A στον κόμβο 0 και την χρονική στιγμή t μεταφέρουμε την μνήμη του στον κόμβο 1. Μετράμε την συμπεριφορά του A σε όλη την διάρκεια της εκτέλεσης του. Επαναλαμβάνουνε την ίδια διαδικασία για το A με τους κόμβους 2 και 3 αντίστοιχα. Η συμπεριφορά του A θα είναι η ίδια και στις δύο περιπτώσεις. Αυτό συμβαίνει γιατί, ο κόμβος 2 είναι ίδιος με τον κόμβο 0 καθώς και ο κόμβος 3 είναι ίδιος με τον κόμβο 1 και η σχέση των 0-1 είναι ίδια με την σχέση των 2-3. Ουσιαστικά τα δύο παραπάνω σενάρια δεν έχουν καμία σημαντική διαφορά πέρα από τα ονόματα που δίνουμε στους κόμβους. Το ίδιο συμβαίνει τις σχέσεις:

- $0 \rightarrow 1$ και $2 \rightarrow 3$
- $0 \rightarrow 2$ και $3 \rightarrow 1$
- $2 \rightarrow 0$ και $1 \rightarrow 3$

Εδώ πρέπει να παρατηρήσουμε ότι οι σχέσεις $(0 \rightarrow 2\ ,\ 3 \rightarrow 1)$ και $(2 \rightarrow 0\ ,\ 1 \rightarrow 3)$ είναι διαφορετικές καθώς η απόσταση του κόμβου 2 από τον κόμβο 0 δεν είναι ίδια από την απόσταση του κόμβου 0 από τον κόμβο 2, όπως φαίνεται από τον πίνακα 4.1. Αυτό συμβαίνει διότι το σύστημα μας είναι ασύμμετρο.

Τα νήματα και η μνήμη είναι ανταλλάξιμα: Όταν έχουμε πάρει μια μέτρηση για μια σχέση του συστήματος στην οποία ισχύει ότι: $distance_(i \rightarrow j) = distance_(j \rightarrow i)$, τότε δεν χρειάζεται να πάρουμε μέτρηση για την αντίστροφη σχέση. Αυτό είναι συνέπεια της παραπάνω παραδοχής όταν έχουμε συμμετρία στις αποστάσεις. Εδώ χρειάζεται να δώσουμε έμφαση στο γεγονός ότι οι διαδικασίες της μεταφοράς της μνήμης σε έναν άλλο κόμβο και το να μεταφέρουμε τα νήματα σε έναν άλλο κόμβο, δεν είναι απαραίτητο να πάρουν τον ίδιο χρόνο για να ολοκληρωθούν ούτε να καταναλώσουν τους ίδιους πόρους.

Αυτό που μας ενδιαφέρει είναι ο τρόπος με τον οποίο θα επηρεαστούν οι διεργασίες μετά την ολοκλήρωση της διαδικασίας, όπου και στις δύο περιπτώσεις θα είναι ο ίδιος.

Συνοψίζοντας, αρχικά παρατηρούμε ότι δυνατές εκτελέσεις για ένα benchmark δεν είναι καθοριστικός παράγοντας για να πάρουμε αξιοποιήσιμη πληροφορία για το σύστημα μας, καθώς πολλές από τις δυνατές εκτελέσεις θα μας δώσουν πληροφορίες για το σύστημα που ήδη έχουμε. Στην συνέχεια κάνοντας χρήση των παραδοχών που έχουμε ορίσει μπορούμε να ορίσουμε την τοπολογία. Η τοπολογία περιγράφει τον τρόπο με τον οποίο επηρεάζονται οι διεργασίες όταν μεταφέρουμε την μνήμη ή τα νήματα της διεργασίας από ένα κόμβο σε έναν άλλο μέσο μια απόστασης d. Ακόμα η τοπολογία άφορα και την τοπική εκτέλεση μιας διεργασίας, δηλαδή εκτελέσεις όπου τα νήματα και η μνήμη βρίσκονται στον ίδιο κόμβο, για κάθε διαφορετικό είδος κόμβου που έχουμε στο σύστημα μας.

Οπότε μπορούμε να πάρουμε αξιοποιήσιμη πληροφορία για το σύστημα από κάθε διαφορετική τοπολογία. Άρα αυτό που χρειάζεται να κάνουμε είναι να βρούμε όλες τις διαφορετικές τοπολογίες ενός συστήματος και να πάρουμε μετρήσεις πάνω σε αυτές αντί να μετρήσουμε όλες τις δυνατές εκτελέσεις. Στο παράδειγμα που μελετήσαμε έχουμε τις εξής τοπολογίες:

- *Τοπολογία 0:* Τοπική εκτέλεση του benchmark σε έναν κόμβο του συστήματος. Για παράδειγμα στον κόμβο 0.
- *Τοπολογία 1:* Εκτέλεση του benchmark μεταξύ δυο κόμβων που απέχουν απόσταση 10. Για παράδειγμα τα νήματα του benchmark μπορεί να βρίσκονται στον κόμβο 0 και η μνήμη στον κόμβο 1.
- *Τοπολογία 2:* Εκτέλεση του benchmark μεταξύ δυο κόμβων που απέχουν απόσταση 20. Για παράδειγμα τα νήματα του benchmark μπορεί να βρίσκονται στον κόμβο 0 και η μνήμη στον κόμβο 3.
- *Τοπολογία 3:* Εκτέλεση του benchmark μεταξύ δυο κόμβων που απέχουν απόσταση 30. Για παράδειγμα τα νήματα του benchmark μπορεί να βρίσκονται στον κόμβο 1 και η μνήμη στον κόμβο 3 και να έχουμε κατεύθυνση από τον κόμβο 1 στον κόμβο 3.
- *Τοπολογία 4:* Εκτέλεση του benchmark μεταξύ δυο κόμβων που απέχουν απόσταση 40. Για παράδειγμα τα νήματα του benchmark μπορεί να βρίσκονται στον κόμβο 0 και η μνήμη στον κόμβο 2 και να έχουμε κατεύθυνση από τον κόμβο 0 στον κόμβο 2.

Η παραπάνω μεθοδολογία μπορεί να επεκταθεί και σε σενάρια στα οποία η μνήμη βρίσκεται σε δύο ή και παραπάνω κόμβους του συστήματος και τα νήματα σε κάποιον άλλον κόμβο. Τέτοιου είδους σενάρια δεν μας απασχόλησαν στην παρούσα εργασία.

4.2.3 Τοπολογίες μηχανημάτων Sandman και Broady

Σε αυτό το υποκεφάλαιο θα δούμε τις τοπολογίες που έχουν τα μηχανήματα που χρησιμοποιήσαμε στην ερευνά μας.

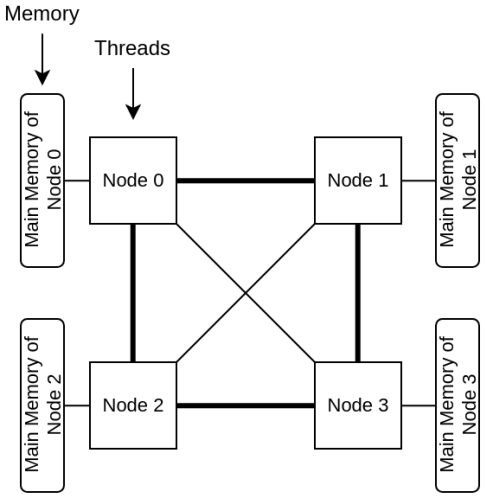
Sandman

Ας αρχίσουμε με το μηχάνημα Sandman. Εδώ έχουμε τις εξής τοπολογίες:

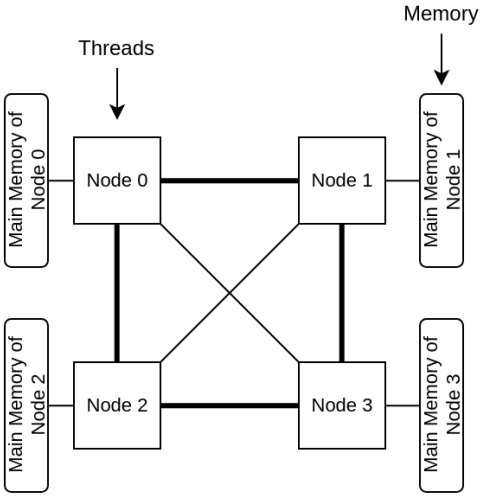
- *Τοπολογία 0:* Τοπική εκτέλεση του benchmark σε έναν κόμβο του συστήματος. Για παράδειγμα στον κόμβο 0, σχήμα 4.3 (a).

- *Τοπολογία 1:* Εκτέλεση του benchmark μεταξύ δυο κόμβους που απέχουν απόσταση 21. Για παράδειγμα τα νήματα του benchmark μπορεί να βρίσκονται στον κόμβο 0 και η μνήμη στον κόμβο 1, σχήμα 4.3 (b).
- *Τοπολογία 2:* Εκτέλεση του benchmark μεταξύ δυο κόμβους που απέχουν απόσταση 30. Για παράδειγμα τα νήματα του benchmark μπορεί να βρίσκονται στον κόμβο 0 και η μνήμη στον κόμβο 3, σχήμα 4.3 (c).

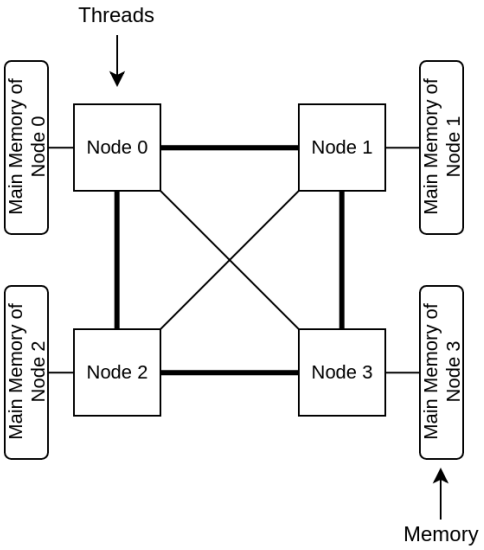
Οι αποστάσεις των κόμβων βρίσκονται στον πίνακα 3.4.



(a) Τοπολογία 0 (local).



(b) Τοπολογία 1 (remote near).



(c) Τοπολογία 2 (remote far).

Σχήμα 4.3: Τοπολογίες του μηχανήματος Sandman.

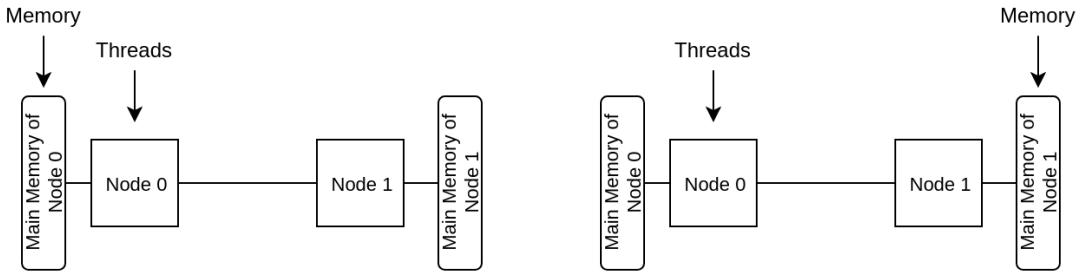
Broady

Στην συνέχεια έχουμε το μηχάνημα Broady. Εδώ έχουμε τις εξής τοπολογίες:

- *Τοπολογία 0:* Τοπική εκτέλεση του benchmark σε έναν κόμβο του συστήματος. Για παράδειγμα στον κόμβο 0, σχήμα 4.4 (a).

- *Τοπολογία 1:* Εκτέλεση του benchmark μεταξύ δυο κόμβους που απέχουν απόσταση 21. Για παράδειγμα τα νήματα του benchmark μπορεί να βρίσκονται στον κόμβο 0 και η μνήμη στον κόμβο 1, σχήμα 4.4 (b).

Οι αποστάσεις των κόμβων βρίσκονται στον πίνακα 3.8.



(a) Τοπολογία 0 (local). (b) Τοπολογία 1 (remote).

Σχήμα 4.4: Τοπολογίες του μηχανήματος Broady.

Παρατηρώντας τις τοπολογίες των συστημάτων μας, βλέπουμε ότι είναι συγκριτικά λίγες σε σχέση με το παράδειγμα του προηγούμενου υποκεφάλαιο. Αυτό οφείλεται στο γεγονός ότι και τα δύο μηχανήματα που μελετήσαμε είναι συμμετρικά και ότι αποτελούνται από μικρό σχετικά αριθμό κόμβων.

4.3 Ανάλυση εργαλείων

Εδώ θα αναλύσουμε τα εργαλεία που χρησιμοποιήσαμε για να πάρουμε μετρήσεις πάνω στα benchmark. Θα αναφέρουμε τις βασικές λειτουργίες που επιτελούν και το λόγο για τα οποία τα χρησιμοποιήσαμε. Τέλος θα πούμε κάποια προβλήματα που αντιμετωπίσαμε με κάποια εργαλεία.

4.3.1 Το εργαλείο numactl

Το πρώτο εργαλείο που θα αναλύσουμε είναι το numactl[1]. Το numactl το χρησιμοποιήσαμε σε πολλά σημεία της έρευνας για διάφορους σκοπούς. Οι βασικότερες λειτουργίες που επιτέλεσε είναι οι εξής:

- Παροχή πληροφοριών για το σύστημα
- Τοποθέτηση νημάτων στους επεξεργαστές
- Δέσμευση μνήμης στους επιθυμητούς κόμβους

Παροχή πληροφοριών για το σύστημα: Η πρώτη λειτουργία που καλούμε το εργαλείο numactl να κάνει είναι να μας δώσει κάποιες βασικές πληροφορίες για το σύστημα το οποίο μελετάμε. Αυτό το κάνουμε μέσο της κλήσης numactl –hardware. Με αυτήν την μέθοδο παίρνουμε γρήγορα και με σαφή τρόπο τα χαρακτηριστικά τα οποία θα χρησιμοποιήσουμε σε όλη την διάρκεια της μελέτης μας, από την κατασκευή των τοπολογιών του κάθε μηχανήματος μέχρι την εκτέλεση των benchmark.

Στα πρώτα στάδια της μελέτης μας όπως έχουμε αναφέρει, είναι κύριας σημασία να κάνουμε ανάλυση του μηχανήματος μας με σκοπό να βρούμε της τοπολογίες που θα μετρήσουμε. Το numactl μας παρέχει τις απαραίτητες πληροφορίες που χρειαζόμαστε για

αυτόν τον σκοπό, βάση των παραδοχών που έχουμε κάνει. Το πρώτο κομμάτι πληροφοριών που μας δίνει είναι οι πινάκες των αποστάσεων των κόμβων. Οι πινάκες για τα μηχανήματα Sandman και Broady είναι οι πίνακες 3.4 και 3.8 αντίστοιχα. Χρησιμοποιώντας αυτούς τους πίνακες και την μεθοδολογία που αναπτύξαμε στο κεφάλαιο 4.2.2 καταλήξαμε στις τοπολογίες που θα μελετήσουμε.

Το επόμενο κομμάτι δεδομένων που μας παρέχει άφορα την τοποθεσία των επεξεργαστών σε σχέση με τους κόμβους του συστήματος. Στους πίνακες 3.1 και 3.5 βλέπουμε την οργάνωση των επεξεργαστών σε σχέση με τους κόμβους για τα μηχάνημα Sandman και Broady αντίστοιχα. Η μορφή με την οποία μας παρέχεται αυτή η πληροφορία είναι πολύ εύκολα κατανοητή από τον άνθρωπο, αφού το αρχείο είναι γραμμένο σχεδόν σε φυσική γλώσσα, όμως δεν είναι εύκολα διαχειρίσιμο από υπολογιστή.

Για αυτό το λόγο κάνουμε μια προ-επεξεργασία σε αυτό το αρχείο για να το φέρουμε σε μια μορφή πίνακα έτσι ώστε να είναι ευκολότερη η διαχείριση του. Αυτό το αρχείο είναι ένα από τα κύρια αρχεία που χρησιμοποιεί ο επόπτης στο κομμάτι της επεξεργασίας των δεδομένων για να αποφασίσει την θέση στην οποία θα τοποθετήσει τα νήματα των benchmark.

Τέλος μας παρέχει πληροφορίες για το μέγεθος της μνήμης που είναι συνδεδεμένη με κάθε κόμβο καθώς και για το πόση είναι ελεύθερη την παρούσα στιγμή. Στους πίνακες 3.3 και 3.7 βλέπουμε την μνήμη που διαθέτει ο κάθε κόμβος για τα μηχανήματα Sandman και Broady αντίστοιχα.

Οι επόμενες δύο λειτουργίες του numactl χρησιμοποιούνται εξολοκλήρου από την δομική μονάδα του εκκινητή στον επόπτη.

Τοποθέτηση νημάτων στους επεξεργαστές: Το numactl μας παρέχει την δυνατότητα να τοποθετήσουμε τα νήματα ενός benchmark, σε όλη την διάρκεια της εκτέλεσης του, πάνω στους επιθυμητούς επεξεργαστές. Αυτό το κάνει μέσω της κλήσης numactl -C cpus, όπου το cpus είναι μια λίστα με τους επεξεργαστές που θέλουμε να πάρει το benchmark. Την λίστα με τους επεξεργαστές την κατασκευάζει ο επόπτης στο κομμάτι της επεξεργασίας τον δεδομένων και την δίνει στην δομική μονάδα του εκκινητή για να χρησιμοποιηθεί από το numactl. Έτσι τοποθετούμε σωστά όλα τα benchmark στους επεξεργαστές που θέλουμε.

Δέσμευση μνήμης στους επιθυμητούς κόμβους: Εδώ θα δούμε μια από τις σημαντικότερες λειτουργίες του numactl. Το numactl μας επιτρέπει να εκκινήσουμε ένα benchmark και να του ορίσουμε από ποιους κόμβους του συστήματος μπορεί να πάρει μνήμη, για όλη την διάρκεια της εκτέλεσης του. Για να το κάνει αυτό έχει δύο βασικούς τρόπους, όταν θέλουμε να πάρουμε μνήμη πέρα από του τοπικού.

1. Με χρήση του numactl -m nodes.
2. Με χρήση του numactl -i nodes.

Και οι δύο κλήσεις δέχονται σαν όρισμα μια λίστα με τους κόμβους του συστήματος που θέλουμε να μελετήσουμε.

Ας δούμε μια μια τις παραπάνω, καθώς είναι πολύ σημαντικές και οι δυο για την έρευνα που κάναμε. Στο πίνακα 4.2 βλέπουμε κάποια παραδείγματα χρήσης του numactl από τον επόπτη για της τοπολογιών τον μηχανημάτων Sandman και Broady.

numactl -m nodes

Αρχίζοντας με την κλήση numactl -m nodes ή αλλιώς όπως την ονομάζουμε στην εργασία membind ή remote. Μέσω αυτής της κλήσης μας δίνετε η δυνατότητα να τρέξουμε

	Topology	Example
Sandman	0	numactl -C 0
	1	numactl -C 0 -m 1
		numactl -C 0 - i 0, 1
	2	numactl -C 0 - m 2
		numactl -C 0 - i 0, 2
Broady	0	numactl -C 0
	1	numactl -C 0 -m 1
		numactl -C 0 - i 0, 1

Πίνακας 4.2: Λίστα με τους performance counters που μετρήσαμε για κάθε επιθυμητή μέτρηση.

τα benchmark με όλη τους την μνήμη σε κάποιον απομακρυσμένο κόμβο του συστήματος. Αυτό είναι πολύ χρήσιμο, γιατί μπορούμε να παρακολουθήσουμε τον τρόπο με τον οποίο επηρεάζεται η συμπεριφορά των benchmark, όταν η μνήμη βρίσκεται μακριά από την εφαρμογή.

Για να πάρουμε πλήρως όλες τις μετρήσεις που χρειαζόμαστε για να έχουμε μια καλή εικόνα για το σύστημα που μελετάμε, χρειάζεται να πάρουμε μια μέτρηση για κάθε benchmark και κάθε τοπολογία πέρα από την τοπική με χρήση αυτού του εργαλείου.

Ακόμα το εργαλείο μας επιτρέπει να του δώσουμε πάνω από έναν επιθυμητούς κόμβους. Αυτή η λειτουργιά είναι χρήσιμη όταν γεμίσει ο πρώτος απομακρυσμένος κόμβος που του έχουμε δώσει οπότε πάει στον επόμενο να δεσμεύσει μνήμη. Σε εμάς δεν μας φάνηκε χρήσιμη αυτή η λειτουργία γιατί όλες οι εφαρμογές μας ζητούσαν χώρο που ήταν μικρότερος από την μνήμη που είχε ο κάθε κόμβος. Όποτε στην έρευνα μας του δίναμε σαν όρισμα μόνο τον απομακρυσμένο κόμβο που θέλαμε να μελετήσουμε σε κάθε τοπολογία.

numactl -i nodes

Στην συνέχεια ας δούμε την κλήση numactl -i nodes ή αλλιώς όπως την ονομάζουμε στην εργασία interleave. Αυτή η κλήση μοιάζει πολύ με την παραπάνω, καθώς και αυτή μας επιτρέπει να δεσμεύουμε μνήμη σε απομακρυσμένος κόμβους του συστήματος όμως διαφέρει πολύ ο τρόπος με τον οποίο γίνεται η δέσμευση. Όταν η λίστα κόμβων που δέχεται σαν παράμετρο η κλίση περιέχει μόνο ένα απομακρυσμένο κόμβο και κανένα άλλο τότε οι δύο κλήσεις δεν έχουν κάποια ουσιαστική διάφορα.

Η διαφοροποίηση των δύο κλήσεων αρχίζει να συμβαίνει όταν έχουμε δύο κόμβους και πάνω. Έστω λιπών ότι έχουμε δύο κόμβους στην λίστα, μέσα σε αυτούς τους κόμβους μπορεί να περιέχεται και ο τοπικός. Τότε κάθε φορά που το benchmark ζητάει να δεσμεύσει νέα μνήμη και αυτό έχει σαν αποτέλεσμα να χρειαστεί να δεσμευτεί μια νέα εικονική σελίδα στο σύστημα μας τότε η νέα σελίδα δεν θα δεσμευτεί κατά ανάγκη στον πρώτο κόμβο της λίστας όπως συνέβαινε πριν. Ο τρόπος που δεσμεύονται οι νέες σελίδες στο σύστημα είναι κυκλικός (round robin) μεταξύ των κόμβων της λίστας έτσι ώστε να υπάρχει ίσος αριθμός από σελίδες σε όλους τους κόμβους που του έχουμε δώσει.

Με αυτόν τον τρόπο μπορούμε να μοιράζουμε τις σελίδες και αντίστοιχα την μνήμη των benchmark σε δύο κόμβους και πάνω, κατά την διάρκεια της εκτέλεσης τους χωρίς να χρειάζεται να επεμβαίνουμε εξωτερικά. Χρησιμοποιήσαμε αυτήν την κλήση στην έρευνα μας για να μελετήσουμε τον τρόπο με τον οποίο επηρεάζονται τα benchmark όταν έχουνε ένα πόστο της μνήμης τους σε κάποιον απομακρυσμένο κόμβο. Συγκεκριμένα χρησιμοποιήσαμε αυτήν την κλήση για όλες τις μη τοπικές τοπολογίες δίνοντας της σαν όρισμα

τον απομακρυσμένο κόμβο και τον τοπικό. Έτσι καταφέραμε να έχουμε το 50% την μνήμης του benchmark στον απομακρυσμένο κόμβο και το άλλο 50% στον τοπικό κόμβο για όλη την διάρκεια της εκτέλεσης του.

4.3.2 Το εργαλείο perf

Το σημαντικότερο εργαλείο που χρησιμοποιήσαμε στην ερευνά μας ήταν το perf[2]. Στα πειράματα μας η έκδοση του perf που χρησιμοποιήσαμε ήτα το perf_3.16. Το perf είναι ένα εργαλείο που μετράει διάφορες παραμέτρους των benchmark κάνοντας χρήση performance counters. Οι μετρήσεις των επιθυμητών παραμέτρων γίνονται κατά την εκτέλεση των benchmark.

Ας δούμε πρώτα τι είναι οι performance counters και πως χρησιμοποιούνται, για να έχουμε μια καλή ιδέα για τη λειτουργία τους. Οι performance counters είναι μετρητές που μετράνε το πλήθος από συγκεκριμένα γεγονότα που συμβαίνουν στο σύστημα μας. Μπορούν να μετράνε γεγονότα τα οποία συμβαίνουν καθολικά σε όλο το σύστημα ή μεμονωμένα για μια συγκεκριμένη εφαρμογή. Οι performance counters είναι απευθείας συνδεδεμένοι με το υλικό του συστήματος μας.

Οι performance counters χωρίζονται σε δύο βασικές κατηγορίες:

1. Hardware performance counters
2. Software performance counters

Οι hardware performance counters είναι μετρητές που μετράνε γεγονότα τα οποία συμβαίνουν στο υλικό του συστήματος μας, όπως γεγονότα τα οποία συμβαίνουν στην CPU, PMU (Performance Monitoring Unit) καθώς και σε άλλα κομμάτια του υλικού. Οι software performance counters μετράνε γεγονότα τα οποία συμβαίνουν στο επίπεδο του λογισμικού. Ένα τυπικό παράδειγμα τέτοιων μετρητών είναι οι performance counters που μετράνε τα page faults που συμβαίνουν στο σύστημα. Στην έρευνα που κάναμε χρησιμοποιήσαμε hardware performance counters.

Οι performance counters δεν είναι απαραίτητα κοινοί σε όλα τα μηχανήματα. Δηλαδή κάποια μηχανήματα μπορεί να έχουν κάποιους εξειδικευμένους performance counters που μετράνε κάποια γεγονότα και να μην υπάρχουν σε κάποια άλλο μηχανήματα. Αυτό ήταν από τους σημαντικότερους προβληματισμούς μας όσον αφορά τους performance counters. Ας δώσουμε ένα παράδειγμα για να γίνει πιο κατανοητός ο προβληματισμός μας. Έστω ότι αποφασίζουμε να χρησιμοποιήσουμε στην έρευνα μας τον performance counter P ο οποίος υπάρχει στο μηχάνημα Sandman και δεν υπάρχει για το μηχάνημα Broady. Εδώ δεν έχει καμία απολύτως σημασία τι μετράει ο P, αλλά έστω ότι αυτό που μετράει είναι σημαντικό για την έρευνα μας. Τότε προκύπτουν τα εξής προβλήματα:

- Δεν έχουμε την δυνατότητα να πάρουμε αντίστοιχες μετρήσεις για το μηχάνημα Broady.
- Δεν μπορούμε να επαληθεύσουμε αν οι συσχετίσεις που κάνουμε με τους άλλους μετρητές είναι έγκυρες.
- Δεν μπορούμε να πιστοποιήσουμε εύκολα την σωστή λειτουργία του P.
- Το μοντέλο πρόβλεψης που θα κατασκευάσουμε με την χρήση του P έχει περιορισμένο εύρος εφαρμογών. Καθώς αν δεν υπάρχει ο P για να μας δώσει την πληροφορία που χρειαζόμαστε για το μοντέλο, δεν θα μπορούμε καν να δημιουργήσουμε το μοντέλο για το συγκεκριμένο μηχάνημα.

Τέτοια και άλλα παρόμοια προβλήματα προκύπτουν αν χρησιμοποιήσουμε performance counters που είναι πολύ εξειδικευμένη η λειτουργία που κάνουν ή δεν υπάρχουν ευρέως στα NUMA συστήματα.

Για να αντιμετωπίσουμε το παραπάνω πρόβλημα αποφασίσαμε να χρησιμοποιήσουμε performance counters που είναι πολύ διαδεδομένοι στα NUMA συστήματα. Στον πίνακα 4.3 βλέπουμε όλους του performance counters που χρησιμοποιήσαμε στην έρευνα μας. Οι δύο πρώτες γραμμές του πίνακα που είναι για το MPKI, IPC και TLB είναι κοινοί και για τα δυο μηχανήματα. Οι τελευταίες τρεις γραμμές είναι λίγο πιο πολύπλοκες καθώς δεν είναι απόλυτα κοινές μεταξύ των δύο μηχανημάτων. Σε αυτόν τον πίνακα θα επανέλθουμε στην συνέχεια, όταν γίνουν διακριτά κάποια προβλήματα που αντιμετωπίσαμε με τους performance counters.

ID	**Result**	**Performance Counters**			
		1	2	3	4
1	MPKI, IPC	instructions	cpu-cycles	LLC-load-misses	LLC-store-misses
2	TLB	instructions	dTLB-load-misses	iTLB-load-misses	dTLB-store-misses
3	IMC1	uncore_imc_0/cas_count_read/	uncore_imc_0/cas_count_write/	uncore_imc_1/cas_count_read/	uncore_imc_1/cas_count_write/
4	IMC2 Sandman	uncore_imc_2/cas_count_read/	uncore_imc_2/cas_count_write/	uncore_imc_3/cas_count_read/	uncore_imc_3/cas_count_write/
	IMC2 Broady	uncore_imc_4/cas_count_read/	uncore_imc_4/cas_count_write/	uncore_imc_5/cas_count_read/	uncore_imc_5/cas_count_write/

Πίνακας 4.3: Λίστα με τους performance counters που μετρήσαμε για κάθε επιθυμητή μέτρηση.

Ένας από τους κύριους περιορισμούς που έχουν οι performance counters και κατά συνέπεια και το εργαλείο perf είναι ότι δεν μπορούμε να μετρήσουμε πολλές παραμέτρους ταυτόχρονα. Αυτό προκύπτει από το γεγονός ότι αν ζητήσουμε από το perf να μετρήσει πολλούς performance counters, τότε κάνει δειγματοληψία (sampling) μεταξύ των performance counters. Δηλαδή αντί να μας δώσει την πραγματική τιμή από παράμετρο που του έχουμε πει να μας μετρήσει, μας δίνει ένα ποσοστό και την μέτρηση που πήρε μέσα σε αυτό το ποσοστό.

Όπως καταλαβαίνουμε η δειγματοληψία προκαλεί πρόβλημα στις μετρήσεις μας. Αυτό είναι αρκετά σημαντικό ζήτημα, γιατί χρειαζόμαστε στο στάδιο συλλογής των δεδομένων να έχουμε πάρει όσο πιο ακριβείς μετρήσεις είναι δυνατόν. Προφανώς η ανάγκη μας για ρεαλιστικές μετρήσεις είναι επιτακτική, έτσι ώστε να μπορέσουμε να βασίσουμε την έρευνα μας σε παραδειγματικά δεδομένα.

Μετά από πειράματα που πραγματοποιήσαμε και στα δύο μηχανήματα μας βρήκαμε ότι ο μέγιστος αριθμός από performance counters που μπορούμε να μετρήσουμε ταυτόχρονα είναι τέσσερα. Με δεδομένο αυτόν τον περιορισμό, αναγκαστήκαμε να μετρήσουμε πολλές φορές την ίδια εκτέλεση ενός benchmark για να πάρουμε όλες τις επιθυμητές μετρήσεις των παραμέτρων του που θέλαμε. Για να ελαχιστοποιήσουμε τον αριθμό από μετρήσεις που έπρεπε να κάνουμε σε μια εκτέλεση, σεβόμενοι πάντα τον περιορισμό, προσπαθήσαμε να συγχωνεύσουμε όσο το δυνατόν περισσότερη πληροφορία σε μια μέτρηση. Στον πίνακα 4.3 βλέπουμε τον τρόπο που συγχωνεύσαμε αυτές τις μετρήσεις, για παράδειγμα μπορούμε σε μια μέτρηση να πάρουμε πληροφορία για τα MPKI και IPC μιας εφαρμογής. Όμως για να πάρουμε πλήρως πληροφορία για το IMC χρειάζεται να πάρουμε δύο διαφορετικές μετρήσεις (IMC1 και IMC2).

Το perf μας δίνει την δυνατότητα να του ορίσουμε κάθε πότε θέλουμε να μας δίνει της μετρήσεις που του έχουμε ζητήσει να πρέπει. Μπορούμε να του ορίσουμε να μας δίνει τις μετρήσεις κάθε ένα συγκεκριμένο χρονικό διάστημα ή να μας δώσει όλες τις μετρήσεις μαζί στο τέλος. Κάνοντας χρήση του χρονικού διαστήματος μπορούμε να βλέπουμε πως εξελίσσονται οι παράμετροι στον χρόνο. Στην έρευνα μας χρησιμοποιήσαμε το χρονικό διάστημα του ενός δευτερολέπτου, έτσι ώστε να έχουμε καλύτερη εικόνα για το πως συμπεριφέρονται οι εφαρμογές μας. Εδώ δεν έχουμε πρόβλημα με τις αλλαγές που συμβαίνουν μέσα σε ένα δευτερόλεπτο, γιατί οι μετρήσεις των performance counters είναι συνεχόμενες, οπότε δεν χάνουμε κάποια πληροφορία. Προφανώς οι μετρητές μηδενίζο-

νται στην αρχή του κάθε χρονικού διαστήματος και αρχίζουν να μετράνε από την αρχή.

Τέλος θα ασχοληθούμε με ένα πρόβλημα που είχαμε με το εργαλείο perf και τους performance counters που μετράνε το IMC. Αρχικά το πρώτο πράγμα που παρατηρούμε για το IMC είναι ότι, παρόλο που και στα δύο μηχανήματα θέλουμε να μετρήσουμε την ίδια παράμετρο βλέπουμε ότι δεν υπάρχουν ακριβώς οι ίδιοι performance counters. Όπως βλέπουμε στον πίνακα 4.3, για το IMC1 και στα δύο μηχανήματα υπάρχουν ακριβώς οι ίδιοι performance counters, ενώ για το IMC2 δεν συμβαίνει αυτό. Για το IMC2 στο έχουμε, για το μηχάνημα Sandman έχουμε τους:

- uncore_imc_2/cas_count_read/
- uncore_imc_2/cas_count_write/
- uncore_imc_3/cas_count_read/
- uncore_imc_3/cas_count_write/

ενώ για το μηχάνημα Broady έχουμε τους:

- uncore_imc_4/cas_count_read/
- uncore_imc_4/cas_count_write/
- uncore_imc_5/cas_count_read/
- uncore_imc_5/cas_count_write/

Όπως αναφέραμε και πριν, προκύπτουν πολλά προβλήματα όταν έχουμε διαφορετικούς performance counters μεταξύ των μετρήσεων μας, ακόμα και όταν μετράνε θεωρητικά την ίδια ποσότητα.

Συνεχίζοντας, οι performance counters για το IMC δεν συμβαδίσουν με το εγχειρίδιο του perf. Αυτό προκύπτει από το γεγονός ότι, το perf_3.16 list μας λέει ότι υπάρχουν τέσσερις performance counters και στα δύο μας συστήματα για να μετράνε τέσσερις ελεγκτές. Δεδομένο το οποίο δεν ισχύει για το μηχάνημα Broady, καθώς αποτελείται από δύο μόνο κόμβους και ο κάθε κόμβος έχει μόνο έναν ελεγκτή για την μνήμη του. Αυτά τα δυο γεγονότα μας έχουν βάλει σε σκέψεις για την αξιοπιστία αυτών των performance counters.

Ακόμα, τα αποτελέσματα που μας έδιναν αυτοί οι performance counters δεν επιβεβαιωνόντουσαν από την μέχρι τώρα θεωρία που είχαμε. Ας αρχίσουμε με το μηχάνημα Sandman. Εδώ οι performance counters που μετράνε το IMC1 δίνουν συνέχεια αποτέλεσμα μηδέν, ανεξάρτητα από την τοπολογία, του benchmark ή οποιοδήποτε άλλο πείραμα προσπαθήσαμε να εκτελέσουμε πάνω τους. Παρόλο που φαίνεται να λειτουργούν κανονικά δεν μπορούμε να εξηγήσουμε την συμπεριφορά τους.

Τώρα για το IMC2 στο μηχάνημα Sandman οι performance counters $j = \{2, 3\}$, θεωρητικά αντιστοιχούν στους κόμβους 2 και 3 του συστήματος. Όμως αυτό δεν επαληθεύεται από τις μετρήσεις που πήραμε, γιατί όταν για παράδειγμα μετράμε την τοπολογία 0 θα έπρεπε να βλέπαμε ότι οι κόμβοι 2 και 3 έχουν IMC2 ≈ 0, καθώς σε εκείνους τους κόμβους δεν υπάρχει κάποια δεσμευμένοι μνήμη για να έχουμε ροή προς κάποιον άλλο κόμβο. Γεγονός το οποίο δεν συμβαίνει σε καμία περίπτωση. Αυτό φαίνεται ξεκάθαρα στους πίνακες των μετρήσεων μας, πίνακες 8.1-8.5 και 8.9-8.13. Για το μηχάνημα Broady, για το IMC1 και το IMC2 παρατηρούμε παρόμοια συμπεριφορά με αυτήν του IMC2 που είχαμε στο μηχάνημα Sandman. Δηλαδή στις περιπτώσεις που θα έπρεπε να φαίνεται ότι δεν υπάρχει ροή μεταξύ των κόμβων, φαίνεται να υπάρχει. Αυτό μπορούμε να το δούμε στους πίνακες 8.6-8.8 και 8.14-8.16.

Επίσης υπήρχε μια ακόμα δυσμορφία σε αυτούς τους μετρητές. Συνέχιζαν να κάνουν καταγραφή των γεγονότων ακόμα και μετά την ολοκλήρωση του benchmark. Η μέτρηση συνέχιζε επ' άπειρων χωρίς να φαίνεται ότι θα ολοκληρωνόταν κάποια στιγμή. Για να ολοκληρωθεί η διαδικασία χρειάστηκε να δώσουμε σαν όρισμα στο perf το -a. Ο όρος -a

κάνει το perf να παίρνει μετρήσεις από όλο το σύστημα, γεγονός το οποίο αλλοιώνει τις μετρήσεις μας.

Όπως έγινε προφανές το perf δεν είναι το πιο κατάλληλο εργαλείο για να μετρήσουμε όλες τις παραμέτρους που θέλουμε σε μια εφαρμογή. Αυτό ισχύει ιδιαίτερα όταν θέλουμε να μετρήσουμε την ροή της μνήμης μεταξύ των κόμβων του συστήματος. Για να έχουμε λιπών πιο αξιόπιστη πληροφορία όσον αφορά την ροή της μνήμης, αποφασίσαμε να χρησιμοποιήσουμε κάποιο άλλο εργαλείο για να κάνουμε αυτήν την μέτρηση.

4.3.3 Το εργαλείο pcm-memory.x

Όπως είδαμε στο προηγούμενο υποκεφάλαιο το perf είναι ένα πολύ χρήσιμο εργαλείο για την ερευνά μας, όμως έχει τους περιορισμούς του. Λόγο του προβλήματος που αντιμετωπίσαμε με τους performance counters που μετρούσαν το IMC, αποφασίσαμε να χρησιμοποιήσουμε ένα νέο εργαλείο για να μετρήσουμε την ροή της μνήμης.

Το νέο εργαλείο που αποφασίσαμε να χρησιμοποιήσουμε είναι το pcm-memory.x. Το pcm-memory.x προέρχεται από την σουίτα προγραμμάτων Intel Performance Counter Monitor[5]. Εμείς χρησιμοποιήσαμε την έκδοση V2.11 της σουίτας.

Η σουίτα Intel Performance Counter Monitor μας παρέχει πολλά εργαλεία για την μελέτη των benchmark. Επειδή είχαμε ήδη πάρει όλες της απαραίτητες μετρήσεις που θέλαμε με το εργαλείο perf, αποφασίσαμε να μην χρησιμοποιήσουμε εξολοκλήρου την σουίτα, παρά το γεγονός ότι είχαν αντίστοιχες δυνατότητες. Ουσιαστικά αυτό που θέλαμε να κάνουμε είναι μια επαλήθευση των αποτελεσμάτων που πήραμε με το εργαλείο perf για το IMC. Σε περίπτωση που τα αποτελέσματα που είχαμε πάρει με το εργαλείο perf δεν επαληθευόντουσαν θα χρησιμοποιούσαμε τις νέες μετρήσεις στη θέση των παλιών.

Το εργαλείο pcm-memory.x μας δίνει την δυνατότητα να πάρουμε μετρήσεις για την ροή της μνήμης μεταξύ των κόμβων του συστήματος μας. Τα αποτελέσματα που πήραμε κάνοντας χρήση του εργαλείου pcm-memory.x συμβάδιζαν με την θεωρία μας. Αρχικά το pcm-memory.x αναγνώριζε σωστά τους ελεγκτές της μνήμης και στα δύο μας συστήματα, δηλαδή στο μηχάνημα Sandman έβρισκε οι έχουμε τέσσερις ελεγκτές και στο μηχάνημα Broady έβρισκε οι έχουμε δύο ελεγκτές. Στην συνέχεια τα αποτελέσματα που παίρνουμε όσον αφορά την ροή είναι τα αναμενόμενα. Δηλαδή στην τοπικές τοπολογίες βλέπουμε, ότι έχουμε ροή μόνο στον τοπικό κόμβο και δεν πηγαίνει προς κάποιον άλλο κόμβο η ροή. Για της μη τοπικές τοπολογίες βλέπουμε πάλι σωστή συμπεριφορά. Για παράδειγμα στην τοπολογία 1 του μηχανήματος Sandman, σχήμα 4.3 (b), βλέπουμε ότι η ροή είναι μεταξύ των κόμβων 0 και 1.

Όπως έχει αρχίσει να φαίνεται το εργαλείο pcm-memory.x είναι καλύτερο στο να μετράει την ροή της μνήμης από το εργαλείο perf. Όμως όπως θα δούμε στο κεφάλαιο 5.1.3 τα αποτελέσματα που μας δίνουν και τα δύο εργαλεία είναι ανάλογα στο ευρύτερο πλαίσιο. Στο κεφάλαιο 5.1.3 αναλύουμε περαιτέρω αυτό το ζήτημα.

Το εργαλείο pcm-memory.x, όπως και το εργαλείο perf, έχει την δυνατότητα να του ορίσουμε ένα χρονικό διάστημα στο οποίο θέλουμε να μας δίνει της μετρήσεις που του έχουμε ζητήσει. Για να έχουμε συνάφεια μεταξύ των μεθόδων μας αποφασίσαμε να παίρνουμε μετρήσεις κάθε ένα δευτερόλεπτο. Οι μετρήσεις που του έχουμε ζητήσει να μας κάνει, μας της δίνει σε ένα αρχείο που του έχουμε ορίσει σε μορφή csv. Προφανώς αυτή η μορφή μας είναι πολύ εύκολα διαχωρίσιμη.

Μια αξιοσημείωτη διαφορά μεταξύ των εργαλείων pcm-memory.x και perf είναι ότι το εργαλείο pcm-memory.x δεν παρατηρήσαμε να κάνει δειγματοληψία. Αυτό το γεγονός διευκόλυνε πολύ την έρευνα μας, γιατί δεν χρειάζεται να πάρουμε πολλαπλές μετρήσεις

για την ροή όπως συνέβαινε με το εργαλείο perf. Το εργαλείο pcm-memory.x, σε μια μέτρηση, μας παρέχει πληροφορία για το τι ροή μνήμης περνάει από όλους τους ελεγκτές του συστήματος. Ακόμα διαχωρίζει αυτήν ροή σε τρις κατηγορίες.

1. Mem Read (MB/s)
2. Mem Write (MB/s)
3. Memory (MB/s)

Όπως βλέπουμε και οι τρεις κατηγορίες μετράνε την ροή σε MB/s. Για τον κάθε ελεγκτή έχουμε διαφορετική μέτρηση και στις τρεις κατηγορίες.

Η πρώτη κατηγορία αφορά την ροή που προέρχεται από διάβασμα που συμβαίνει στην μνήμη, ενώ η δεύτερη κατηγορία αφορά την ροή που προέρχεται από εγγραφές που συμβαίνει στην μνήμη. Η τρίτη κατηγορία είναι το άθροισμα της ροής των δύο προηγούμενων κατηγοριών. Εμείς χρησιμοποιήσαμε την τρίτη κατηγορία στην ερευνά μας, γιατί δεν θέλαμε να διαχωρίσουμε την ροή σε δυο κατηγορίες, ανάγνωσης και εγγραφής. Τέλος οι δύο πρώτες κατηγορίες χωρίζονται σε δύο υποκατηγορίες, όμως δεν μας απασχόλησαν στην έρευνά μας.

Όπως είδαμε το εργαλείο pcm-memory.x μας φάνηκε αρκετά χρήσιμο στην έρευνα μας. Καταφέραμε να πάρουμε πιο ξεκάθαρες και ρεαλιστικές μετρήσεις με την χρήση του. Μας επαλήθευσε την θεωρία που είχαμε για τα συστήματα μας και μας έλυσε το πρόβλημα που είχαμε με το perf στο κομμάτι του IMC. Παρόλο που στην συνέχεια θα δούμε ότι οι μετρήσεις που μας έδωσαν και τα δύο εργαλεία είναι ανάλογες, το pcm-memory.x το θεωρούμε πιο αξιόπιστο από το perf για να μετρήσουμε την ροή της μνήμης στα συστήματα μας. Γιατί επαληθεύονταν από την θεωρία και επίσης η χρήση του ήταν πιο εύκολη σε σχέση με το perf όσον αφορά το κομμάτι των μετρήσεων.

4.3.4 Τα αρχεία numa_maps

Ένα άλλο αρκετά σημαντικό εργαλείο που χρησιμοποιήσαμε είναι το numa_maps[6]. Το numa_maps είναι ένα εργαλείο το οποίο μας παρέχει πληροφορείς για την κατάσταση της μνήμης των εφαρμογών μας. Κατά την διάρκεια εκτέλεσης μια εφαρμογής αλλά και μετά το πέρας της θέλουμε να ξέρουμε πόση μνήμη δεσμεύτηκε και σε ποίους κόμβους, αυτήν την πληροφορία μας την δίνει το numa_maps.

Έστω ότι θέλουμε να δούμε την κατάσταση της μνήμης του benchmark A, που έχει pid x. Για να το κάνουμε αυτό μπορούμε να κοιτάξουμε το αρχείο /proc/x/numa_maps, που περιέχει την στιγμιαία κατάσταση στην οποία βρίσκεται η μνήμη του A την χρονική στιγμή που κοιτάξαμε το αρχείο. Η δομή αυτού του αρχείου είναι πολύ απλή και εύκολα χρησιμοποιείται για να πληροφορείς τόσο κατά την εκτέλεση του benchmark όσο και μετά.

Το αρχείο /proc/<pid>/numa_maps μας παρέχει πληροφορίες για:

- Την θέση την μνήμης σε σχέση με το heap και το stack.
- Την εικονική διεύθυνση των σελίδων.
- Την τοποθεσία τους σε σχέση με τους κόμβους του συστήματος.
- Το ποιες σελίδες είναι καθαρές και ποιες όχι.
- Το μέγεθος της κάθε σελίδας.
- Τις ανώνυμες σελίδες που έχει η εφαρμογή.
- Τα ανοιχτά αρχεία που έχει η εφαρμογή και για το μέγεθος τους.

Πολλές από τις παραπάνω πληροφορίες τις συνδυάζουμε για να βρούμε χαρακτηριστικά των εφαρμογών μας σε αρκετά κομμάτια της έρευνας μας. Τα δύο βασικότερα από αυτά είναι, στο κομμάτι της μεταφοράς της μνήμης μεταξύ των κόμβων του συστήματος

μας και στην κατανόηση και επαλήθευση των benchmark.

Πριν συνεχίσουμε με τους τρόπους που χρησιμοποιούμε το numa_map, χρειάζεται να τονίσουμε μια ιδιομορφία που έχει το numa_map. Τα αρχεία numa_map υπάρχουν μόνο κατά την διάρκεια εκτέλεσης του benchmark και παρέχουν πληροφορίες για την στιγμιαία κατάσταση στην οποία βρίσκεται η μνήμη του. Αυτό σημαίνει ότι τα αρχεία διαγράφονται μετά το πέρας της εκτέλεση του benchmark και ότι δεν παρέχουν μια συνέχh εικόνα για την κατάσταση της μνήμης του. Για να αντιμετωπίσουμε αυτό το πρόβλημα αποθηκεύουμε την προσωρινή κατάσταση κάθε ένα δευτερόλεπτο σε ένα αρχείο. Έτσι μετά το πέρας της εκτέλεσης του μπορούμε να δημιουργήσουμε μια εικόνα για την μνήμη του. Προφανώς με αυτόν τον τρόπο δεν μπορούμε να εντοπίσουμε αλλαγές στην μνήμη του benchmark που συμβαίνουν μέσα σε ένα δευτερόλεπτο.

Ας μελετήσουμε πρώτα την εφαρμογή που βρίσκει το numa_maps στην μεταφορά μνήμης και συγκεκριμένα στην χρήση του movepages. Η κλήση movepages όπως θα δούμε στο κεφάλαιο 6.3, δεν είναι τόσο απλή όσο η migratepages. Για να λειτουργήσει σωστά, χρειάζεται κατά την διάρκεια της εκτέλεσης της εφαρμογής μας να γνωρίσουμε ακριβώς την διεύθυνση των εικονικών σελίδων που θέλουμε να μεταφέρουμε καθώς και τον κόμβο στον οποίο βρίσκονται πριν την μεταφορά.

Το numa_maps είναι το κατάλληλο εργαλείο για να μας δώσει αυτήν την πληροφορία. Αυτό συμβαίνει γιατί μας παρέχει την πληροφορία σε μια πολύ απλή δομή οπότε δεν χρειάζεται να χάσουμε πολύ χρόνο στην επεξεργασία του αρχείου. Ακόμα μας δίνει όλες τις παραμέτρους που χρειάζεται να ξέρουμε για να χρησιμοποιήσουμε το movepages. Αυτό είναι που σημαντικό καθώς δεν χρειάζεται να χρησιμοποιήσουμε και κάποιο άλλο εργαλείο για να πάρουμε κάποια παράμετρο που θα έλειπε. Έτσι έχουμε ελάχιστη επίδραση στην απόδοση των εφαρμογών κατά την διάρκεια της εκτέλεσης τους. Αυτό το κομμάτι βρίσκει εφαρμογή στην δομική μονάδα μεταφορέα στον επόπτη.

Ένα άλλο σημαντικό κομμάτι που βρίσκει εφαρμογή το numa_map είναι στην κατανόηση και επαλήθευση της συμπεριφοράς των benchmark. Η δομική μονάδα του παρακολουθητή στον επόπτη είναι υπεύθυνη να μαζέψει αυτήν την πληροφορία από το numa_map. Αρχικά μέσω αυτού του τρόπου μπορούμε πολύ εύκολα να επαληθεύσουμε ότι το εκάστοτε πείραμα με την μνήμη που κάναμε ολοκληρώθηκε με επιτυχία. Δηλαδή ότι η μνήμη της εφαρμογής μας βρισκόταν στην σωστή θέση την σωστή χρονική στιγμή. Ακόμα μπορούμε να καταλάβουμε, με γραφική αναπαράσταση της χρονικής εξέλιξης της μνήμης, αν οι εφαρμογές μας δεσμεύουν και αποδεσμεύουν μνήμη συχνά, αν δεσμεύουν μνήμη που δεν γράφουν καθώς και άλλα.

Όπως φάνηκε τα αρχεία numa_map είναι ένα αρκετά χρήσιμα, τόσο για την εκτέλεση των πειραμάτων μας όσο και για την επαλήθευση τους.

Κεφάλαιο 5

Επίδοση Εφαρμογών και Μνήμη

Σε αυτό το κεφάλαιο θα ασχοληθούμε με το σημαντικότερο κομμάτι της έρευνας που κάναμε. Εδώ θα μετρήσουμε συγκεκριμένα χαρακτηριστικά εφαρμογών και θα προσπαθήσουμε να βρούμε μια σχέση, για το πώς επηρεάζεται η απόδοση τον εφαρμογών σε σχέση με την μνήμη τους. Στην συνέχεια θα γενικεύσουμε αυτήν την μέθοδο έτσι ώστε να μπορούμε να προβλέψουμε την συμπεριφορά της εκάστοτε εφαρμογής όταν θέλουμε να κάνουμε μια λειτουργία στην μνήμη της.

5.1 Ανάλυση εφαρμογών

Εδώ θα αναλύσουμε τις εφαρμογές που χρησιμοποιήσαμε στην έρευνα μας. Αυτό το κομμάτι είναι αρκετά σημαντικό για να αποκτήσουμε καλή εικόνα για τον τρόπο που επηρεάζονται τα προγράμματα από την μνήμη. Αρχικά θα δούμε ποιες εφαρμογές χρησιμοποιήσαμε στην έρευνα μας. Στην συνέχεια θα δούμε τις παραμέτρους που μετρήσαμε πάνω σε αυτές τις εφαρμογές. Τέλος θα δούμε πως επηρεάστηκαν αυτές οι εφαρμογές από την μνήμη και τις συσχετίσεις των μετρήσεων μεταξύ τους.

5.1.1 Οι εφαρμογές που χρησιμοποιήσαμε

Ας αρχίσουμε την ανάλυση μας παρουσιάζοντας τα benchmarks που χρησιμοποιήσαμε για να πάρουμε μετρήσεις. Για να έχουμε μια ποικιλία από benchmarks αποφασίσαμε να χρησιμοποιήσουμε δύο σουίτες από benchmarks. Η πρώτη σουίτα από benchmark είναι η Spec2006[20] και δεύτερη σουίτα είναι η Parsec[17].

Σε αυτό το κομμάτι της έρευνας ασχοληθήκαμε μόνο με μονονηματικές εφαρμογές. Αυτό το κάναμε για να περιορίσουμε το χώρο αναζήτησης του προβλήματος μας. Όμως όλες οι μεθοδολογίες που θα αναπτύξουμε σε αυτό το κεφάλαιο καθώς και στα επόμενα μπορούν πολύ εύκολα να γενικευτούν και για πολυνηματικές εφαρμογές.

Η ποικιλία από πολλά διαφορετικά benchmark είναι αρκετά σημαντική. Για να μπορέσουμε να γενικεύσουμε την έρευνα μας χρειαζόμαστε να παρατηρήσουμε πολλές και διαφορετικές συμπεριφορές από τα benchmark. Αυτό δεν αφορά μόνο τον τρόπο με τον οποίο επηρεάζονται οι εφαρμογές από την μνήμη αλλά και τις ίδιες τις παραμέτρους των εφαρμογών μας. Δηλαδή για να αποκτήσουμε εποπτική εικόνα για το πρόβλημα που μελετάμε χρειαζόμαστε:

1. Πολλές μετρήσεις από benchmark.
2. Πολλές διαφορετικές συμπεριφορές ανάμεσα στα benchmark σε σχέση με το πώς επηρεάζονται από την μνήμη τους.

3. Benchmark που όταν μετρήσουμε τις παραμέτρους τους, θα μας δώσουν μεγάλη ποικιλία από μετρήσεις.
4. Benchmark που έχουν συνοχή μεταξύ των εκτελέσεων. Δηλαδή αν τρέξουμε το ίδιο benchmark κάτω από τις ίδιες συνθήκες θα μας δώσουν τα ίδια αποτελέσματα όσον αφορά της μετρήσεις και την απόδοση.

Για να μπορέσουμε να μελετήσουμε ρεαλιστικά το πρόβλημα μας πρέπει να ισχύουν και τα τέσσερα χαρακτηριστικά ταυτόχρονα. Αν έστω και ένα από αυτά τα χαρακτηριστικά λείπει στην μελέτη μας έχουμε αρκετά μεγάλο πρόβλημα. Ας δούμε γιατί, μελετώντας μια μια τις περιπτώσεις να λείπει ένα από τα παραπάνω χαρακτηριστικά ενώ ισχύουν τα άλλα τρία. Προφανώς άμα λείπουν δύο και πάνω χαρακτηριστικά πάλι έχουμε πρόβλημα.

Έστω ότι λείπει το χαρακτηριστικό 1: Τότε δεν μπορούμε να σχηματίσουμε εικόνα για το ποιες περιπτώσεις είναι πολύ ειδικές και για το πια είναι η γενική συμπεριφορά.

Έστω ότι λείπει το χαρακτηριστικό 2: Αν έχουμε εικόνα μόνο από μια συγκεκριμένη συμπεριφορά εφαρμογών, τότε δεν θα μπορούμε να χειριστούμε καλά συμπεριφορές από εφαρμογές που δεν ανήκουν σε αυτήν την γκάμα.

Έστω ότι λείπει το χαρακτηριστικό 3: Έστω ότι έχουμε πάρει μετρήσεις από εφαρμογές που έχουν όλες σχεδόν τις ίδιες παραμέτρους αλλά έχουν διαφορικές συμπεριφορές για την μνήμη. Το συγκεκριμένο σενάριο δεν είναι πολύ ρεαλιστικό, γιατί κάτι τέτοιο δεν συμβαίνει συχνά στην πραγματικότητα. Όμως και πάλι έχουμε πρόβλημα, γιατί τότε δεν θα μπορούσαμε να ξεχωρίσουμε ποιοι παράμετροί είναι χρήσιμοι και ποιοι όχι. Οπότε πολύ πιθανόν να χρησιμοποιούσαμε άχρηστες πληροφορείς.

Έστω ότι λείπει το χαρακτηριστικό 4: Αν το benchmark δεν έχει συναφή συμπεριφορά τότε δεν μπορούμε να το μελετήσουμε καθόλου. Αυτό συμβαίνει γιατί δεν θα μπορούσαμε ούτε να ερμηνεύσουμε ούτε να διαχωρίσουμε τις διαφορετικές συμπεριφορές του. Ακόμα δεν θα μπορούσαμε να παίρναμε τις επαναλαμβανόμενες μετρήσεις που χρειαζόμασταν.

Όπως είδαμε το μεγάλο εύρος από μετρήσεις παίζει αρκετά σημαντικό ρόλο στην έρευνα μας. Για να αποκτήσουμε αυτό το μεγάλο εύρος χρησιμοποιήσαμε δυο διαφορετικές σουίτες από εφαρμογές. Η πρώτη σουίτα από benchmark που χρησιμοποιήσαμε είναι η Spec2006 ή αλλιώς όπως την αναφέρουμε στην έρευνα Spec. Η σουίτα Spec είναι ευρέως διαδεδομένη και έχει χρησιμοποιηθεί σε πολλές έρευνες και μελέτες. Στην έρευνα μας δεν χρησιμοποιήσαμε όλα τα benchmark από τα Spec, γιατί είτε κάποια δεν ήταν υλοποιημένα για μετρήσεις στα συστήματα μας είτε γιατί δεν μας δίναν αξιοποιήσιμη πληροφορία.

Ένα κάλο παράδειγμα από benchmark που δεν μας δίναν αξιοποιήσιμη πληροφορία είναι το 998.Specrand, γιατί η ολοκλήρωση του ήταν πάρα πολύ γρήγορη, κάτω από ένα δευτερόλεπτο, οπότε δεν μπορούσαμε να πάρουμε έγκυρες μετρήσεις πάνω του. Θυμίζουμε ότι αν ένα benchmark είναι πάρα πολύ γρήγορο τότε δεν μπορούμε να το μετρήσουμε σωστά λόγο του χρόνου που χάνουμε για τον συγχρονισμό στον επόπτη. Το κάτω όριο είναι τρία δευτερόλεπτα.

Η άλλη σουίτα από benchmark που χρησιμοποιήσαμε είναι η Parsec. Όπως η σουίτα Spec είναι πολύ διαδεδομένη στην έρευνα έτσι και η σουίτα Parsec είναι αρκετά διαδεδομένη. Η σουίτα Parsec μας παρέχει και αυτή αρκετά benchmark. Όμως ούτε εδώ χρησιμοποιήσαμε ολόκληρη την σουίτα. Αυτό συνέβη για τους ίδιους λόγους με την σουίτα Spec.

Σε αυτήν την σουίτα εντοπίσαμε benchmark που του έλειπε το χαρακτηριστικό 4. Το benchmark αυτό ήταν το x264. Κάθε φορά που προσπαθούσαμε να το παρατηρήσουμε άλλαζε η συμπροφορά του και οι μετρήσεις που παίρναμε, οπότε δεν το συμπεριλάβαμε στην λίστα με τα benchmark που μελετήσαμε.

Ακόμα η σουίτα Parsec μας έδινε την δυνατότητα να τρέξουμε τα benchmark που μελετούσαμε με πολλά νήματα. Μια δυνατότητα που δεν είχαμε με την σουίτα Spec, καθώς η σουίτα Spec αφορά μόνο μονονηματικές εφαρμογές. Στην σουίτα Parsec μπορούσαμε τις ίδιες εφαρμογές μετά ίδια αρχεία να τις τρέξουμε με όσα νήματα επιθυμούσαμε.

Suite	Benchmark	Inputs
Spec2006	400.perlbench	3
	401.bzip	6
	403.gcc	9
	410.bwaves	1
	416.gamess	3
	429.mcf	1
	433.milc	1
	434.zeusmp	1
	435.gromacs	1
	436.cactusADM	1
	437.leslie3d	1
	444.namd	1
	445.gobmk	5
	447.dealII	1
	450.soplex	2
	453.povray	1
	456.hmmer	2
	458.sjeng	1
	459.GemsFDTD	1
	462.libquantum	1
	464.h264ref	3
	465.tonto	1
	470.lbm	1
	471.omnetpp	1
	473.astar	2
	482.sphinx3	1
Parsec	blackscholes	1
	bodytrack	1
	canneal	1
	dedup	1
	facesim	1
	ferret	1
	fluidanimate	1
	freqmine	1
	rtview	1
	swaptions	1
	streamcluster	1
	vips	1

Πίνακας 5.1: Τα benchmark που χρησιμοποιήσαμε στην έρευνα.

Τέλος κάποιες από τις εφαρμογές από την σουίτα Spec είχαμε πολλαπλά αρχεία εισόδου. Αποφασίσαμε να συμπεριλάβουμε όλα τα benchmark που είχαν πολλαπλά αρχεία

εισόδου στην έρευνα μας. Στον πίνακα 5.1 βλέπουμε όλα τα benchmark που χρησιμοποιήσαμε στην έρευνα μας, από ποια σουίτα προέρχονται και πόσα αρχεία εισόδου είχαν. Στην συνέχεια της έρευνα μας θεωρήσαμε ότι κάθε διαφορετική εκτέλεση από κάποιο ίδιο benchmark με διαφορετικό αρχείο εισόδου είναι και ένα διαφορετικό benchmark.

5.1.2 Χαρακτηριστικά των εφαρμογών

Σε αυτό το υποκεφάλαιο θα δούμε τα αποτελέσματά που πήραμε για τις παραμέτρους των benchmark. Αρχικά χρειάζεται να ορίσουμε πλήρως της παραμέτρους των εφαρμογών μας βάση των εργαλείων που χρησιμοποιήσαμε. Πρώτα θα ορίσουμε τις παραμέτρους του εργαλείου perf και στην συνέχεια την παράμετρο από το εργαλείο pcm-memory.x.

Ας δούμε τι είναι οι παράμετροι που μετράμε και πως τις υπολογίζουμε από τους performance counters που έχουμε μετρήσει. Σε όλη την παρακάτω ανάλυση έχουμε θεωρήσει ότι, το benchmark A χρειάζεται n δευτερόλεπτα για να ολοκληρώσει την εκτέλεση του, οπότε για τον performance counters P έχουμε πάρει n τιμές για αυτόν.

MPKI

Το MPKI ή αλλιώς LLC-MPKI (Last level Cache Misses per Kilo Instructions), μας δείχνει τον μέσο όρο των αριθμών των σφαλμάτων που συμβαίνει στην τελευταία cache κάθε χίλιες εντολές του προγράμματος μας. Το MPKI είναι μια πολύ διαδεδομένη μετρική προγραμμάτων, που σχετίζεται άμεσα με την εφαρμογή που μετράμε καθώς και με την αρχιτεκτονική του μηχανήματος μας. Όσο μικρότερο MPKI έχουμε τόσο λιγότερο επηρεάζεται το πρόγραμμα μας από την LLC. Ο τύπος για τον υπολογισμό του από τους performance counters είναι ο παρακάτω:

$$\text{MPKI} = \frac{\sum_{i=1}^{n} \text{LLC-load-misses[i]} + \sum_{i=1}^{n} \text{LLC-store-misses[i]}}{\sum_{i=1}^{n} \text{instructions[i]}} \cdot 1000$$

IPC

Το IPC (Instructions per Cycle), μας δείχνει τον μέσο όρο των αριθμών των εντολών που εκτελούνται (γίνονται committed) σε κάθε κύκλο λειτουργίας. Το IPC είναι πάρα πολύ διαδεδομένη μετρική, καθώς χρησιμοποιείται σε πολλά κομμάτια της έρευνας για να μετράμε την απόδοση ενός προγράμματος. Όσο μεγαλύτερο IPC έχουμε τόσο γρηγορότερη είναι η εκτέλεση του προγράμματος μας. Ο τύπος για τον υπολογισμό του από τους performance counters είναι ο παρακάτω:

$$\text{IPC} = \frac{\sum_{i=1}^{n} \text{instructions[i]}}{\sum_{i=1}^{n} \text{cpu-cycles[i]}}$$

TLB

Το TLB ή αλλιώς TLB-MPKI (Translation Lookaside Buffer Misses per Kilo Instructions), μας δείχνει τον μέσο όρο των αριθμών των αποτυχιών που συμβαίνει στον TLB κάθε χίλιες εντολές του προγράμματος μας. Αυτή η μετρική ας λέει αν μία εφαρμογή είναι πολύ εξαρτημένη από την μνήμη (memory intensive), δηλαδή όσο χαμηλότερη είναι η εξάρτηση

τόσο χαμηλότερο είναι και το TLB. Ο τύπος για τον υπολογισμό του από τους performance counters είναι ο παρακάτω:

$$\text{TLB} = \frac{\sum_{i=1}^{n} \text{dTLB-load-misses[i]} + \sum_{i=1}^{n} \text{dTLB-store-misses[i]}}{\sum_{i=1}^{n} \text{instructions[i]}} \cdot 1000$$

Εδώ αποφασίσαμε να μην συμπεριλάβουμε στα αποτελέσματα μας τον μετρητή iTLB-load-misses, γιατί παρατηρήσαμε ότι δεν μας έδινε κάποια επιπλέον πληροφορία για της παραμέτρους και το μόνο που έκανε είναι ότι δημιουργούσε θόρυβο.

IMC

Το IMC (Intel Memory Controller), μας δείχνει τον μέσο όρο μνήμης που διέρχεται μέσω ενός ελεγκτή μνήμης. Το IMC το υπολογίζουμε για κάθε έναν ελεγκτή που μας υποδηλώνει το perf_3.16 list σε κάθε ένα από τα μηχανήματα μας. Ο τύπος για τον υπολογισμό του από τους performance counters είναι ο παρακάτω:

$$\text{IMC (}\mathbf{j}\text{)} = \frac{\sum_{i=1}^{n} \text{uncore\_imc\_}\mathbf{j}\text{/cas\_count\_read/[i]} + \sum_{i=1}^{n} \text{uncore\_imc\_}\mathbf{j}\text{/cas\_count\_write/[i]}}{n}$$

Όπου το **j** είναι ο εκάστοτε ελεγκτής. Για το μηχάνημα Sandman το perf_3.16 list μας λέει ότι έχουμε τους ελεγκτής $\{0, 1, 2, 3\}$, ενώ για το μηχάνημα Broady οι ελεγκτής που έχουμε είναι οι $\{0, 1, 4, 5\}$. Το αποτέλεσμα αυτών των performance counters είναι σε MiB/s.

Στην συνέχεια ας δούμε ποια είναι η άλλη μας παράμετρο από το εργαλείο pcm-memory.x και πως την ορίσαμε. Έστω ότι τρέχουμε το benchmark A και χρειάζεται t δευτερόλεπτα για να ολοκληρωθεί. Ακόμα έστω ότι έχουν n ελεγκτές στο σύστημα μας έναν για κάθε κόμβο. Τότε για τον κάθε ελεγκτή έχουμε πάρει t μετρήσεις μια ανά ένα δευτερόλεπτο.

BW

Το BW (Bandwidth), μας δείχνει τον μέσο όρο της ροής της μνήμης που διέρχεται στο σύστημα μας. Το BW δεν αφορά μόνο έναν κόμβο ή μια τοπολογία αλλά αφορά όλο το σύστημα σαν μια ενιαία οντότητα. Όσο χαμηλότερο είναι το BW τόσο λιγότερη μνήμη ρέει μέσα στο σύστημα μας. Ο τύπος για τον υπολογισμό της συνολικής ροής του συστήματος μας είναι ο παρακάτω:

$$\text{BW} = \frac{\sum_{j=0}^{n} (\sum_{i=1}^{t} \text{Memory[j][i]})}{t}$$

Στους πίνακες 8.1-8.16 του παρατήματος Α, βλέπουμε τα αποτελέσματα που πήραμε για τις παραμέτρους των benchmark. Οι πίνακες είναι χωρισμένοι ανά σουίτα, μηχάνημα, τοπολογία και εκτέλεση όσον αφορά το numactl, με αυτήν τη σειρά.

- Η πρώτη στήλη των πινάκων μας λέει το όνομα του benchmark σε επεκτεταμένη μορφή.
- Η δεύτερη στήλη των πινάκων, το TT (Total Time) μας λέει το συνολικό χρόνο που χρειάστηκε το benchmark για να ολοκληρώσει την εκτέλεση του μετρημένο σε δευτερόλεπτα.

- Η τρίτη στήλη των πινάκων μας λέει το MPKI του benchmark.
- Η τέταρτη στήλη των πινάκων μας λέει το IPC του benchmark.
- Η πέμπτη στήλη των πινάκων μας λέει το TLB του benchmark.
- Η τέσσερις τελευταίες στήλες των πινάκων μας λένε το IMC του benchmark. Στην κορυφή κάθε στήλης λέμε τον ελεγκτή που μετράει το εργαλείο perf

Για να πάρουμε πλήρως όλες τις μετρήσεις χρειαστήκαμε περίπου ~ 275 ώρες. Αυτός ο αριθμός ωρών είναι ο ελάχιστος που μπορούμε να έχουμε βάση των περιορισμών που έχουμε. Προφανώς χρειαστήκαμε πολλές περισσότερες ώρες για τις μετρήσεις μας για να βεβαιωθούμε ότι τα αποτελέσματα μας αντικατοπτρίζουν την πραγματικότητα.

5.1.3 Απόδοση εφαρμογών και συσχετίσεις

Σε αυτό το υποκεφάλαιο θα μελετήσουμε τον τρόπο που επηρεάστηκαν οι εφαρμογές από τον τοποθεσία που ήταν η μνήμη τους. Στην συνέχεια θα δούμε πως σχετίζονται οι παράμετροι αυτών των εφαρμογών με την απόδοση αλλά και πως συσχετίζονται οι παράμετροι μεταξύ τους. Τέλος θα βάλουμε τα απαραίτητα θεμέλια που χρειαζόμαστε για την μοντελοποίηση του προβλήματος μας.

Αρχικά θα κάνουμε μερικές διευκρινίσεις για να έχουμε καλύτερη κατανόηση του τι συμβαίνει.

1. Ως επίπεδο αναφοράς (baseline) έχουμε θεωρήσει τις τοπικές τοπολογίες (τοπολογίες 0). Αυτό ισχύει και για τα δύο μηχανήματα και για τους δύο τρόπους εκτέλεσης (interleave, remote).
2. Κάθε benchmark σε κάθε τοπολογία και με κάθε τρόπο εκτέλεσης αντιπροσωπεύει ένα μοναδικό σημείο.
3. Το κάθε σημείο χαρακτηρίζεται από τις παραμέτρους του εκάστοτε benchmark και σχετίζεται απευθείας μόνο, με τα άλλα σημεία του ίδιου benchmark.
4. Θεωρούμε ότι έχουμε πλήρη εικόνα για όλες τις τοπικές τοπολογίες. Δηλαδή γνωρίζουμε όλες τις παραμέτρους που έχουμε μετρήσει, για όλα τα benchmark.
5. Όσον αφορά τις άλλες τοπολογίες και εκτελέσεις το μόνο που γνωρίζουμε είναι ο χρόνος που χρειάστηκαν για να ολοκληρώσουν την εκτέλεση τους.
6. Θεωρούμε ότι έχουμε σχετικό σφάλμα στις μετρήσεις μας κάτω από 2%.

Το πρώτο κομμάτι που θα αναλύσουμε εδώ, είναι τα αποτελέσματα που είχε η θέση της μνήμης στις εφαρμογές μας. Για να το κάνουμε αυτό χρειαζόμαστε να ορίσουμε την απόδοση μια εφαρμογής. Την απόδοση την ορίσαμε ως εξής:

$$\text{Performance}(i,j) = \frac{\text{Total Time}(i,j)}{\text{Total Time}_{local}} \cdot 100\% \equiv \frac{\text{IPC}_{local}}{\text{IPC}(i,j)} \cdot 100\%$$

Οπού το i μας δείχνει την τοπολογία και το j μας δείχνει τον τρόπο μεταφοράς. Με αυτόν τον ορισμό βρίσκουμε την απόδοση και κάθε τοπολογία και εκτέλεση. Η επιλογή αυτό του ορισμού έγινε καθαρά για χρηστικούς λόγους. Η baseline απόδοση όλων τον εφαρμογών είναι το 100%.

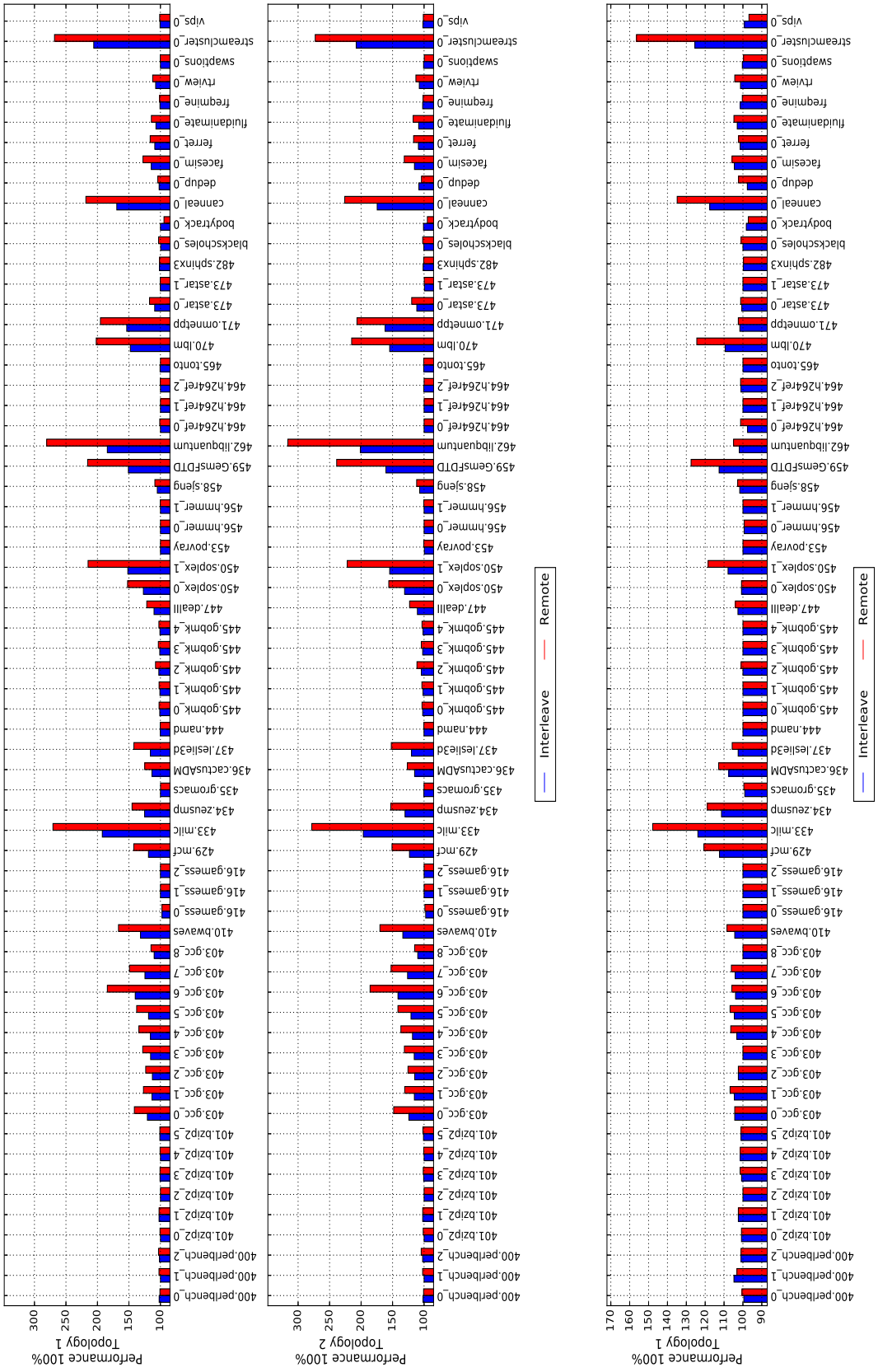
Αν θέλουμε να δούμε την μεταβολή της απόδοσης, τότε μπορούμε να την υπολογίσουμε πολύ εύκολα από τον τύπο:

$$\triangle P_{(i,j)} = \frac{\text{Total Time}(i,j)}{\text{Total Time}_{local}} \cdot 100\% - 100\% = (\frac{\text{Total Time}(i,j)}{\text{Total Time}_{local}} - 1) \cdot 100\%$$

Το πρόσημο εδώ έχει φυσική σημασία. Αν είναι θετικό τότε το πρόγραμμα μας επιβραδύνθηκε ενώ αν είναι αρνητικό τότε το πρόγραμμα μας επιταχύνθηκε.

		Performance %					
		Sandman				**Broady**	
		interleave		remote		interleave	remote
Suite	**Benchmark**	Topology 1	Topology 2	Topology 1	Topology 2	Topology 1	Topology 1
Spec	400.perlbench_0	102.0161	101.6129	100.8065	100.8065	99.5050	100.4950
	400.perlbench_1	100.0000	100.0000	102.2989	102.2989	104.7619	103.1746
	400.perlbench_2	101.5267	102.2901	103.0534	104.5802	100.9901	100.9901
	401.bzip2_0	100.0000	100.0000	100.7194	101.4388	100.7812	100.7812
	401.bzip2_1	102.1739	102.1739	102.1739	102.1739	102.4390	102.4390
	401.bzip2_2	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000
	401.bzip2_3	100.6757	100.6757	100.6757	101.3514	100.7407	101.4815
	401.bzip2_4	100.0000	100.0000	100.6173	100.6173	101.3986	101.3986
	401.bzip2_5	100.9346	100.9346	100.9346	101.8692	101.0101	101.0101
	403.gcc_0	120.6897	124.1379	141.3793	148.2759	104.3478	104.3478
	403.gcc_1	113.4615	115.3846	126.9231	130.7692	104.5455	106.8182
	403.gcc_2	112.7660	114.8936	123.4043	125.5319	102.4390	102.4390
	403.gcc_3	115.6250	115.6250	128.1250	131.2500	100.0000	100.0000
	403.gcc_4	115.7895	118.4211	134.2105	136.8421	103.2258	106.4516
	403.gcc_5	118.8679	120.7547	137.7358	141.5094	104.5455	106.8182
	403.gcc_6	140.0000	141.4286	184.2857	185.7143	103.9216	105.8824
	403.gcc_7	124.5902	126.2295	149.1803	152.4590	104.0816	106.1224
	403.gcc_8	110.0000	110.0000	115.0000	115.0000	100.0000	100.0000
	410.bwaves	131.5702	133.7190	166.6116	170.0826	104.3373	108.4337
	416.gamess_0	97.4684	97.4684	97.4684	98.7342	100.0000	100.0000
	416.gamess_1	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000
	416.gamess_2	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000
	429.mcf	118.9979	123.1733	142.3480	150.9434	112.4138	120.6897
	433.milc	192.2942	196.4974	270.6468	278.2753	123.8659	147.8346
	434.zeusmp	125.0725	130.2899	145.0725	152.8986	111.3636	118.8636
	435.gromacs	100.0000	100.2920	99.8540	100.1460	99.0148	99.5074
	436.cactusADM	113.4900	115.0566	124.8908	126.5502	107.4605	112.8571
	437.leslie3d	115.8730	119.9295	142.1517	152.0282	102.5316	105.6962
	444.namd	100.1757	100.1757	100.3521	100.3521	100.0000	100.0000
	445.gobmk_0	101.0989	102.1978	102.1978	103.2967	100.0000	100.0000
	445.gobmk_1	100.8658	101.7316	102.1645	103.4632	100.0000	100.0000
	445.gobmk_2	102.6087	104.3478	107.8261	111.3043	100.0000	100.9524
	445.gobmk_3	101.1111	102.2222	103.3333	104.4444	100.0000	100.0000
	445.gobmk_4	100.8264	101.6529	102.4793	103.3058	100.0000	100.0000
	447.dealII	110.1770	110.6195	121.6814	123.0088	102.6525	103.9788
	450.soplex_0	127.0718	130.9392	152.1978	156.0440	100.7634	100.7634
	450.soplex_1	151.7730	154.6099	214.8936	221.9858	107.8947	118.4211
	453.povray	100.0000	99.5868	100.0000	100.4132	100.0000	100.0000
	456.hmmer_0	100.0000	100.0000	100.0000	100.0000	99.3464	99.3464
	456.hmmer_1	100.0000	100.0000	100.2299	100.4598	100.0000	100.0000
	458.sjeng	104.9598	106.9705	108.8472	111.7962	101.6949	102.7735
	459.GemsFDTD	150.9294	160.5948	215.5844	238.7755	112.6214	127.4752
	462.libquantum	184.2975	201.2397	280.5785	316.3223	101.9157	104.9618
	464.h264ref_0	100.0000	100.0000	101.0989	100.0000	97.5309	101.2346
	464.h264ref_1	100.0000	100.0000	100.0000	100.0000	100.0000	100.0000
	464.h264ref_2	100.0000	100.4800	100.1600	100.0000	101.0582	101.2346
	465.tonto	100.5917	100.5917	100.0000	100.5917	100.0000	100.0000
	470.lbm	147.7407	154.6169	201.9724	214.9901	109.4092	124.4493
	471.omnetpp	153.5211	161.9718	195.2596	206.5463	101.6393	102.4590
	473.astar_0	109.1346	111.5385	117.3077	119.7115	100.5988	101.1976
	473.astar_1	100.2755	99.4490	100.2755	99.7245	100.0000	100.0000
	482.sphinx3	101.5564	101.5564	101.4212	100.6460	99.8478	99.6956
Parsec	blackscholes_0	99.6377	100.3623	102.9197	102.1898	100.0000	100.9479
	bodytrack_0	100.0000	100.9217	94.3478	94.7826	98.2143	97.1264
	canneal_0	169.3182	174.4318	218.3784	225.9459	117.5573	134.8485
	dedup_0	102.0833	108.3333	104.1667	104.1667	97.6744	102.3256
	facesim_0	114.5009	115.2542	127.6836	131.4501	104.5346	105.7279
	ferret_0	108.9965	108.9965	116.2950	116.6381	101.4675	102.3061
	fluidanimate_0	107.0981	108.5595	114.4050	117.3278	103.0151	104.7739
	freqmine_0	100.7508	102.2523	101.3493	101.6492	101.3865	100.3454
	rtview_0	107.5862	107.5862	112.3288	113.0137	101.2987	104.3290
	swaptions_0	100.0000	100.7979	100.2660	99.7340	100.3344	99.6656
	streamcluster_0	205.9122	207.7703	268.1750	272.8507	125.4417	156.4103
	vips_0	100.6250	101.8750	101.2422	101.2422	99.1935	96.8000

Πίνακας 5.2: Σχέσεις απόδοσης και θέσης της μνήμης.



(a) Sandman (b) Broady

Σχήμα 5.1: Γραφική αναπαράσταση του πίνακα 5.2.

Στον πίνακα 5.2 βλέπουμε όλες τις αποδόσεις για όλα τα benchmark, όλες τις τοπολογίες και όλες τις εκτελέσεις. Όμως για να γίνουν πιο κατανοητά τα αποτελέσματα όσων αφορά την απόδοση των benchmark, παραθέτουμε και γραφική αναπαράσταση των αποτελεσμάτων στο σχήμα 5.1. Στο σχήμα 5.1 (a) βλέπουμε τα αποτελέσματα για τον Sandman, όπου η πρώτη γραφική αφορά την τοπολογία 1 και η δεύτερη την τοπολογία 2, και οι δύο γραφικές έχουν την ίδια κλίμακα για να είναι εύκολη η σύγκριση των αποτελεσμάτων. Στο σχήμα 5.1 (b) βλέπουμε τα αποτελέσματα για την τοπολογία 1 του Broady.

Ας αναλύσουμε τα αποτελέσματά που παίρνουμε από το σχήμα 5.1. Αρχικά παρατηρούμε ότι δεν επηρεάζεται η απόδοση από όλα τα benchmark με τον ίδιο τρόπο από την θέση της μνήμης. Αυτό το γεγονός δεν μας φαίνεται περίεργο, αφού άλλωστε περιμέναμε μια τέτοια συμπεριφορά. Όμως αυτό που μας κάνει εντύπωση είναι ότι υπάρχουν εφαρμογές που επηρεάζονται ελάχιστα από την τοπολογία, όπως είναι η εφαρμογή 401.bzip2_2. Ακόμα παρατηρούμε ότι τα benchmark που έχουν την ίδια εφαρμογή και αλλάζουν τα αρχεία εισόδου τους έχουν παρόμοιες συμπεριφορές. Αυτό το διαπιστώνουμε εύκολα κοιτώντας τις ομάδες εφαρμογών 403.gcc και 416.gamess. Τα παραπάνω ισχύουν εξίσου και για τα δυο μηχανήματα.

Συνεχίζοντας την ανάλυση μας, κάνουμε μια πολύ σημαντική παρατήρηση όσον αφορά την σχέση που έχει η απόσταση της μνήμης και η απόδοση. Εδώ χρειάζεται να επικεντρωθούμε στο μηχάνημα Sandman. Ο τρόπος που επηρεάζεται η απόδοση από την απόσταση δεν είναι ίδιος για όλες τις εφαρμογές. Αυτό θα γίνει πιο κατανοητό με ένα παράδειγμα.

Έστω ότι παρατηρούμε τις εφαρμογές 433.milc και 462.libquantum για το μηχάνημα Sandman, τοπολογίες 1 και 2 και έκδοση remote. Όπως βλέπουμε από το σχήμα μας και στις δύο εφαρμογής η απόδοση επηρεάζονται πάρα πολύ από την θέση της μνήμης. Η μεταβολή της απόστασης από την τοπολογία 1 στην τοπολογία 2 είναι:

$$\triangle D = (\frac{30}{21} - 1) \cdot 100\% = 55\%$$

Άρα η απόσταση αυξήθηκε κατά 55%.

Η μεταβολή της απόδοσης των εφαρμογών μας από μια τοπολογία i σε μια τοπολογία i' δεδομένης μια εκτέλεσης j είναι:

$$\begin{aligned}
&\triangle P_{\text{benchmark}(i \to i')} = \\
&= \triangle P_{\text{benchmark (i', j)}} - \triangle P_{\text{benchmark (i, j)}} = \\
&= (\frac{\text{Total Time (i', j)}}{\text{Total Time}_{local}} - 1) \cdot 100\% - (\frac{\text{Total Time (i, j)}}{\text{Total Time}_{local}} - 1) \cdot 100\% = \\
&= (\frac{\text{Total Time (i', j)}}{\text{Total Time}_{local}} - 1 - \frac{\text{Total Time (i, j)}}{\text{Total Time}_{local}} + 1) \cdot 100\% = \\
&= (\frac{\text{Total Time (i', j)}}{\text{Total Time}_{local}} - \frac{\text{Total Time (i, j)}}{\text{Total Time}_{local}}) \cdot 100\% = \\
&= \frac{\text{Total Time (i', j)}}{\text{Total Time}_{local}} \cdot 100\% - \frac{\text{Total Time (i, j)}}{\text{Total Time}_{local}} \cdot 100\% = \\
&= P_{\text{benchmark}}(\text{i', j}) - P_{\text{benchmark}}(\text{i, j})
\end{aligned}$$

Έτσι για το 433.milc έχουμε:

$$\begin{aligned}
&\triangle P_{\text{433.milc}(1 \to 2)} = P_{\text{433.milc}}(1, \text{remote}) - P_{\text{433.milc}}(2, \text{remote}) = \\
&= (278.2753 - 270.6468)\% = 7.6285\%
\end{aligned}$$

και για το 462.libquantum έχουμε:

$$\triangle P_{\text{462.libquantum}(1\to2)} = P_{\text{462.libquantum}}(1, \text{remote}) - P_{\text{462.libquantum}}(2, \text{remote}) =$$
$$= (316.3223 - 280.5785)\% = 35.7438\%$$

Άρα και οι δυο εφαρμογές επιβραδύνθηκαν κατά 7.6285% και 35.7438% αντίστοιχα.

Παρόλο που η απόσταση της μνήμης αυξήθηκε το ίδιο και στις περιπτώσεις, η μεταβολή της απόδοσης δεν ήταν η ίδια για τα δύο benchmark. Παρόμοιες συμπεριφορές βλέπουμε όταν παρατηρούμε και τα υπόλοιπα benchmark. Δηλαδή παρόλο που η μεταβολή στην απόσταση της μνήμης είναι γραμμική, η μεταβολή της απόδοσης των εφαρμογών δεν είναι γραμμική. Αυτό το γεγονός ήταν το κίνητρο μας για την μοντελοποίηση του προβλήματός μας ως κάτι το οποίο δεν είναι γραμμικό.

Τέλος ας κοιτάξουμε την συμπεριφορά που έχουν τα benchmark μεταξύ των δύο μηχανημάτων. Γενικά τα benchmark παρουσιάζουν παρόμοιες συμπεριφορές και στα δύο μηχανήματα. Δηλαδή εφαρμογές που επηρεάζεται η απόδοση τους από την θέση της μνήμης στο ένα μηχάνημα επηρεάζεται και στο άλλο. Προφανώς αυτός ο επηρεασμός δεν έχει το ίδιο βάρος και στα δύο μηχανήματα.

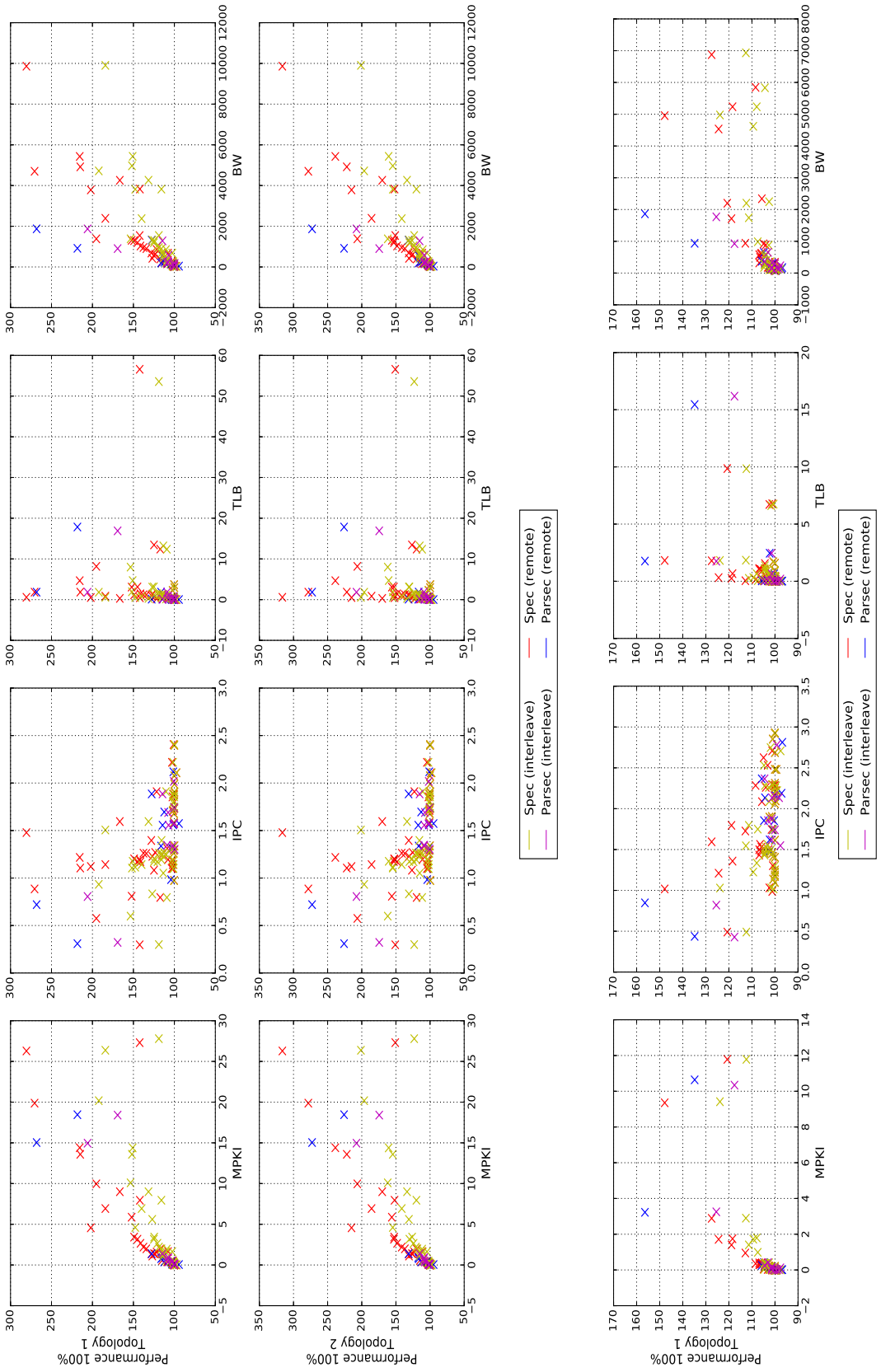
Όμως αυτό δεν συμβαίνει για όλες τις εφαρμογές καθολικά. Μέσα στο σύνολο των benchmark που μελετήσαμε είχαμε και benchmark που επηρεάζεται πολύ η εκτέλεση τους από την αρχιτεκτονική του μηχανήματος στο οποίο τρέχει. Τέτοια χαρακτηριστικά παραδείγματα είναι τα benchmark 462.libquantum και 471.omnetpp. Σε αυτά τα benchmark η μνήμη cache παίζει αρκετά μεγάλο ρόλο. Αυτό μπορούμε να το διαπιστώσουμε πολύ εύκολα κοιτώντας την παράμετρο MPKI στους πίνακες 8.2 και 8.7 για αυτά τα δύο benchmark.

Το επόμενο κομμάτι που θα μελετήσουμε εδώ αφορά την σχέση που έχουν οι παράμετροι των εφαρμογών μας μεταξύ τους, αλλά και με απόδοση. Η κατανόηση αυτών των σχέσεων είναι πολύ σημαντική για να μπορέσουμε να μοντελοποιήσουμε το πρόβλημα, να βρούμε μοτίβα συμπεριφοράς, να ερμηνεύσουμε τα αποτέλεσμα που παρατηρούμε καθώς και άλλα σημαντικά κομμάτια της έρευνας.

Στην ανάλυση μας θα μας βοηθήσει πολύ η γραφική αναπαράσταση όλων των σχέσεων που θα μελετήσουμε. Στα σχήματα 5.2 και 5.3 βλέπουμε την γραφική αναπαράσταση των σχέσεων μας. Τα σχήματα 5.2(a) και 5.3(a) αφορά τις συσχετίσεις για το μηχάνημα Sandman και τα σχήματα 5.2(b) και 5.3(b) αφορά τις συσχετίσεις για το μηχάνημα Broady. Υπενθυμίζουμε ότι, γνωστά μας είναι οι χρόνοι εκτέλεσης για όλες τις τοπολογίες, συνεπώς μας είναι γνωστή και η απόδοση. Ακόμα γνωστά μας είναι και οι παράμετροι του κάθε benchmark για τις τοπικές τοπολογίες. Οπότε μπορούμε να κατασκευάσουμε και να χρησιμοποιήσουμε αυτές τις γραφικές στην περαιτέρω έρευνα μας.

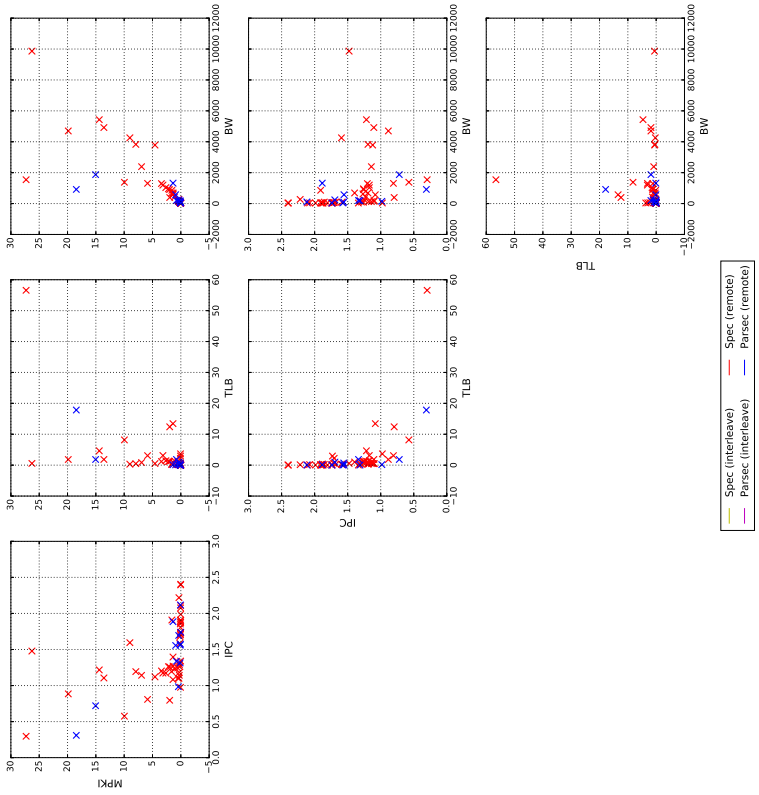
Οι δυο γραμμές από γραφικές από το σχήμα 5.2(a) και η γραμμή από γραφικές από το σχήμα 5.2(b) αφορούν τις συσχετίσεις των παραμέτρων μας με την απόδοση των εφαρμογών μας σε σχέση με την θέση της μνήμης τους. Οι τρεις γραμμές από γραφικές στα σχήματα 5.3(a) και 5.3(b) αφορούν τις συσχετίσεις των παραμέτρων των εφαρμογών για τις τοπικές τοπολογίες. Εδώ κάποιες γραφικές λείπουν, γιατί δεν θα μας έδιναν κάποια έξτρα πληροφορία. Για παράδειγμα δεν υπάρχει η γραφική για την συσχέτιση IPC $\times$ MPKI, όμως υπάρχει η γραφική MPKI $\times$ IPC η οποία είναι ίδια με αυτή που λύπη αλλά με αντεστραμμένους άξονες. Τέλος οι στήλες από γραφικές αναφέρονται πάντα στην ίδια παράμετρο.

Αρχικά ας δούμε κάποια βασικά πράγματα για τις συσχετίσεις που θα μας φανούν πολύ χρήσιμα στην συνέχεια.

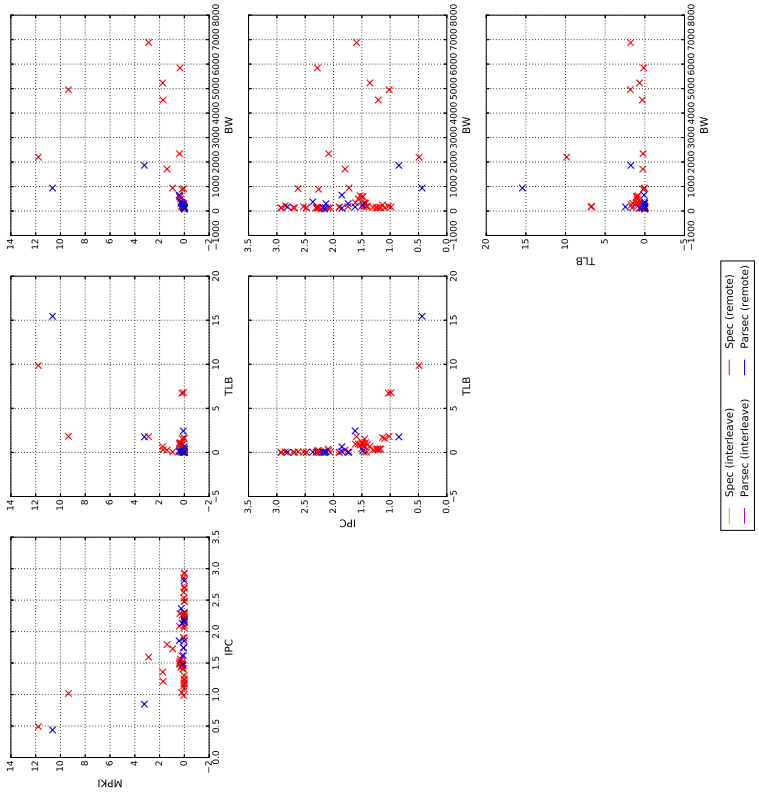


(a) Sandman (b) Broady

Σχήμα 5.2: Συσχετίσεις για την απόδοση των μηχανημάτων.



(a) Sandman



(b) Broady

Σχήμα 5.3: Συσχετίσεις για τις παραμέτρους των μηχανημάτων.

- Αν η γραφική αναπαράσταση μιας συσχέτισης είναι μια οριζόντια ή κάθετη γραμμή, τότε οι δύο ποσότητες συσχετισμού είναι ανεξάρτητες. Αυτή η παρατήρηση είναι αρκετά χρήσιμη γιατί μας απαλλάσσει από τις άχρηστες παραμέτρους που έχουμε μετρήσει.
- Αν η γραφική αναπαράσταση μιας συσχέτιση είναι μια ευθεία γραμμή που έχει μια κλήση α, τότε οι δύο ποσότητες συσχετισμού είναι ανάλογες με παράμετρο α. Αυτή η παρατήρηση είναι αρκετά χρήσιμη καθώς μας επιτρέπει να απαλείψουμε τις περιττές παραμέτρους και να κρατήσουμε μόνο αυτές που μας δίνουν αξιοποιήσιμη πληροφορία.
- Αν η γραφική αναπαράσταση μιας συσχέτισης έχει μορφή σύννεφου, δηλαδή τα σημεία της γραφικής είναι διάσπαρτα στο σχήμα χωρίς κάποια νοητή γραμμή, τότε οι δύο ποσότητες συσχετισμού συνδέονται μη γραμμικά.
- Αν στη γραφική αναπαράσταση μιας συσχέτιση μπορούμε να βρούμε μια νοητή γραμμή, η οποία όμως δεν είναι ευθεία, που μπορεί να αντιπροσωπεύεται από μια συνάρτηση, τότε οι δύο ποσότητες συσχετισμού συνδέονται μέσω αυτής της συνάρτησης.

Έχοντας πλέον μια καλύτερη εικόνα για τις συσχετίσεις ας δούμε τι συμπεράσματα μπορούμε να βγάλουμε από τα αποτελέσματα μας. Αφαιρούμε πλήρως την παράμετρο IMC γιατί βλέπουμε ότι είτε δεν μας δίνουν καθόλου πληροφορίες είτε η πληροφορία που μας δίνουν είναι ήδη γνωστή από άλλες παραμέτρους. Άρα το σύνολο από παραμέτρους που μας δίνουν αξιοποιήσιμη πληροφορία προς το παρόν είναι το {MPKI, IPC, TLB, BW}. Αυτό το σύνολο από παραμέτρους που έχουμε είναι και το ελάχιστο, βάση των μετρήσεων που έχουμε πάρει, καθώς δεν υπάρχει κάποια άλλη συσχέτιση που να μπορούμε να απαλείψουμε είτε για την απόδοση είτε για τις παραμέτρους μεταξύ τους. Αυτό ισχύει και για τα δύο μηχανήματα.

Συνεχίζοντας ας δούμε τις συσχετίσεις που έχει η απόδοση με τις παραμέτρους. Όσα πράγματα θα αναφέρουμε εδώ αφορούν εξίσου και τα δύο μηχανήματα καθώς όλες τις τοπολογίες τους. Εδώ χρειάζεται να κάνουμε μια παρατήρηση: στα σχήματα μας έχουμε τις μετρήσεις και από τις δυο εκτελέσεις μαζί. Οπότε χρειάζεται να δώσουμε προσοχή στην μορφή που έχουν οι γραφικές μας και να ξεχωρίζουμε τις εκτελέσεις την μια από την άλλη.

Ας αρχίσουμε με την παράμετρο MPKI που είναι η πιο εύκολη. Βλέπουμε ότι η απόδοση με το MPKI σχετίζεται μέσω κάποιας συνάρτησης. Η συνάρτηση που προσεγγίζει περισσότερο αυτήν την σχέση είναι η:

$$y = a_1 log(x + 1) + a_2 x + b$$

όπου y είναι η απόδοση και x είναι η παράμετρος μας. Αυτή η σχέση είναι αρκετά εμφανής στο μηχάνημα Sandman και για τις δύο εκδόσεις.

Με παρόμοιο τρόπο μπορούμε να προσεγγίσουμε την παράμετρο BW. Εδώ η σχέση μπορεί να περιγράφει αρκετά ικανοποιητικά από τρεις συναρτήσεις:

$$y = a_1 x + b$$
$$y = a_1 log(x + 1) + b$$
$$y = a_1 log(x + 1) + a_2 x + b$$

Βλέπουμε ότι η τρίτη συνάρτηση είναι ίδια με την συνάρτηση για το MPKI. Όπως θα φανεί και στην συνέχεια αυτές οι τρεις θα μας είναι πολύ χρήσιμες στην έρευνα μας.

Τα πράγματα για τις παραμέτρους IPC και TLB είναι πιο πολύπλοκα. Το IPC εμφανίζει έναν οριζόντιο βραχίονα ενώ το TLB εμφανίζει έναν κάθετο βραχίονα, όμως και στις δύο γραφικές έχουμε σχηματισμούς σύννεφου. Οπότε αυτοί οι δυο παράμετροι φαίνεται να συνδέονται μη γραμμικά με την απόδοση.

Όπως είδαμε σε αυτό υποκεφαλαίο βρήκαμε αρκετά μεγάλη ποσότητα αξιοποιήσιμης πληροφορίας στις μετρήσεις μας. Βρήκαμε βασικές σχέσεις που συνδέουν την απόδοση των εφαρμογών μας με την θέση και την απόσταση της μνήμης στα συστήματα μας. Ακόμα είδαμε ποιες παράμετροι που μετρήσαμε δεν χρειάζονται για την μοντελοποίηση του προβλήματος μας. Τέλος αναλύσαμε με ποίον τρόπο συνδέονται οι παράμετροι μεταξύ τους, γεγονός που θα μας φανεί πολύ χρήσιμο στην συνέχεια της έρευνας μας.

5.2 Υλοποίησεις

Εδώ θα ασχοληθούμε με τις υλοποιήσεις που κάναμε, για να μπορέσουμε να περιγράψουμε την σχέση της απόδοση των εφαρμογών βάση των παραμέτρων τους και της απόστασης της μνήμης. Θα δούμε με ποιον τρόπο παράγουμε τις απαραίτητες συναρτήσεις που θέλουμε για να μπορέσουμε να περιγράψουμε αλλά και στην συνέχεια να προβλέψουμε την παραπάνω σχέση. Τέλος, θα αναλύσουμε την μέθοδο που χρησιμοποιήσαμε για να εκπαιδεύσουμε και να ελέγξουμε τις συναρτήσεις μας.

5.2.1 Παραγωγός συναρτήσεων πρώτης γενιάς

Ένα από τα σημαντικότερα κομμάτια της έρευνας μας ήταν ο τρόπος που παράξαμε τις συναρτήσεις μας. Για να μπορέσουμε να έχουμε ικανοποιητικά αποτελέσματα στην έρευνα μας χρειάζεται να δώσουμε μεγάλο βάρος σε αυτό το κομμάτι. Αυτό προκύπτει από το γεγονός ότι στο τέλος της έρευνας μας θέλουμε να μπορούμε να προβλέψουμε την μεταβολή στην απόδοση των εφαρμογών μας, μέσω των συναρτήσεων που θα κατασκευάσουμε εδώ.

Η παραγωγή των συναρτήσεων χωρίζεται σε δύο στάδια, στους παραγωγούς πρώτης και δεύτερης γενιάς. Ο διαχωρισμός αυτός θα γίνει κατανοητός στην συνέχεια και συγκεκριμένα στο κεφάλαιο 5.3 καθώς εκεί θα αναλύσουμε περαιτέρω τον τρόπο με τον οποίο χρησιμοποιούμε τις συναρτήσεις μας. Εδώ θα ασχοληθούμε με τον τρόπο που παράγουμε τις συναρτήσεις πρώτης γενιάς.

Ο παραγωγός συναρτήσεων πρώτης γενιάς κατασκευάζει τις συναρτήσεις που χρησιμοποιούμε στα πρώτα στάδια της μοντελοποίησης μας. Επειδή τα αποτελέσματα που θα πάρουμε από αυτές τις συναρτήσεις θα καθορίσουν τα επόμενα βήματα της μελέτης μας, χρειάζεται να έχουμε μεγάλη ακρίβεια.

Αρχικά ας δούμε σε ποιες αρχές βασίστηκαν οι συναρτήσεις μας. Οι συναρτήσεις που θέλουμε να κατασκευάσουμε θέλουμε να είναι μη γραμμικές. Αυτό προκύπτει από την ανάλυση που κάναμε για τις παραμέτρους και την επίδοση. Συνεχίζοντας, οι παράμετροι που αποφασίσαμε να αποτελούντες οι συναρτήσεις μας είναι οι $\{$MPKI, IPC, TLB, BW$\}$. Σε κάθε μια παράμετρο αποφασίσαμε να αντιστοιχήσουμε και μια μεταβλητή:

$$X_1 = \text{MPKI},\ X_2 = \text{IPC},\ X_3 = \text{TLB},\ X_4 = \text{BW}$$

Στην συνέχεια χωρίζουμε τους όρους από τους οποίους αποτελούνται οι συναρτήσεις σε βασικούς και σε συσχετίσεις. Πρώτα θα δούμε τους βασικούς όρους των συναρτήσεων. Για τους βασικούς όρους έχουμε κάνει τον εξής διαχωρισμό: ο πρώτος όρος των

συναρτήσεων μας είναι ο κυρίαρχος και όλοι οι υπόλοιποι ονομάζονται δευτερεύοντες. Ο κυρίαρχος είναι αυτός που έχει την περισσότερη βαρύτητα και παίρνει τις περισσότερος μορφές. Ακόμα, κάθε όρος μιας ομάδας θα γίνει μια φορά κυρίαρχος έτσι ώστε να δημιουργήσουμε όλους τους δυνατούς συνδυασμούς. Οι μορφές που παίρνει ο κυρίαρχος όρος είναι οι παρακάτω:

1. $a_1 X_i$
   $a_1 X_i^{n_1}$
2. $a_1 log(X_i + 1)$
   $a_1 log(X_i + 1)^{n_1}$
3. $a_1 log(X_i + 1) + a_2 X_i$
   $a_1 log(X_i + 1) + a_2 X_i^{n_1}$
   $a_1 log(X_i + 1)^{n_1} + a_2 X_i$
   $a_1 log(X_i + 1)^{n_1} + a_2 X_i^{n_2}$

Όπου τα a_i και n_i είναι ελεύθερες μεταβλητές. Οι δευτερεύοντες βασικοί όροι έχουν λιγότερες μορφές, οι μορφές είναι:

1. $a_1 X_i$
2. $a_1 X_i^{n_1}$

Όπου πάλι τα a_i και n_i είναι ελεύθερες μεταβλητές.

Οι συναρτήσεις μας δεν αποτελούνται πάντα από όλες τις παραμέτρους που έχουμε. Δηλαδή μια συνάρτηση που θα κατασκευάσουμε μπορεί να αποτελείται μόνο από τις παραμέτρους MPKI και BW και να μην λαβαίνει υπόψιν τις υπόλοιπες. Αυτή την επιλογή την κάναμε έτσι ώστε να πάρουμε όλο το εύρος από δυνατές συναρτήσεις που μπορούμε να κατασκευάσουμε με όλες τις παραμέτρους. Στον πίνακα 5.3 βλέπουμε όλους τους δυνατούς συνδυασμούς των βασικών παραγόντων.

Size	**Parameters**
1	MPKI
	IPC
	TLB
	BW
2	MPKI, IPC
	MPKI, TLB
	MPKI, BW
	IPC, TLB
	IPC, BW
	TLB, BW
3	MPKI, IPC, TLB
	MPKI, IPC, BW
	MPKI, TLB, BW
	IPC, TLB, BW
4	MPKI, IPC, TLB, BW

Πίνακας 5.3: Όλοι οι δυνατοί συνδυασμοί των βασικών παραμέτρων.

Όπως βλέπουμε έχουμε πολλές ομάδες από διαφορετικές παραμέτρους. Όμως η πολυπλοκότητα δεν σταματάει εκεί, αφού χρειάζεται σε κάθε ομάδα να ορίσουμε τον κυρίαρχο όρο και τους δευτερεύοντες. Για παράδειγμα έστω ότι έχουμε την ομάδα {MPKI, IPC, TLB}, τότε πρέπει κάθε στοιχείο να θεωρηθεί μια φορά κυρίαρχο και να πάρει όλες τις δυνατές μορφές που μπορεί να πάρει ένας κυρίαρχος όρος. Σε αυτό αποτελεί εξαίρεση η πρώτη

μορφή του κυρίαρχου όρου, καθώς όλοι οι παράμετροι θα μας δώσουν τις ίδιες συναρτήσεις. Αυτό όμως, όπως βλέπουμε, δεν ισχύει για τις μορφές 2 και 3. Άρα για το παράδειγμα μας έχουμε:

- *Οποιοσδήποτε όρος κυρίαρχος (Μορφή 1)*: Εδώ δεν μας ενδιαφέρει ποιος είναι ο κυρίαρχος όρος γιατί όλοι οι όροι παράγουν τις ίδιες συναρτήσεις. Ο κάθε όρος έχει δύο δυνατές καταστάσεις. Άρα συνολικά παράγουμε $2^3 = 8$ συναρτήσεις.
- *Κυρίαρχος όρος το MPKI*: Εδώ οι συναρτήσεις που παράγονται εξαρτώνται από το ποιος είναι ο κυρίαρχος όρος.
  - *Μορφή 2*: Ο κυρίαρχος όρος έχει 2 δυνατές μορφές ενώ ο κάθε δευτερεύων έχει 2 δυνατές καταστάσεις. Άρα συνολικά παράγουμε $2 \cdot 2^2 = 8$ συναρτήσεις.
  - *Μορφή 3*: Ο κυρίαρχος όρος έχει 4 δυνατές μορφές ενώ ο κάθε δευτερεύων έχει 2 δυνατές καταστάσεις. Άρα συνολικά παράγουμε $4 \cdot 2^2 = 16$ συναρτήσεις.

Αντίστοιχα πράγματα ισχύουν και όταν είναι κυρίαρχοι όροι τα IPC και TLB. Άρα για το παράδειγμα μας συνολικά παράγουμε:

$$N = 2^3 + \sum_{i=1}^{3} 2 \cdot 2^2 + \sum_{i=1}^{3} 4 \cdot 2^2 = 8 + 3 \cdot (2 \cdot 2^2) + 3 \cdot (4 \cdot 2^2) = 80$$

Το 80 μπορεί να μην φαίνεται μεγάλος αριθμός αρχικά, όμως αυτός ο αριθμός αφορά μόνο μία ομάδα από βασικούς όρους. Λαμβάνοντας υπόψιν όλες τις ομάδες έχουμε:

$$\begin{aligned} N =& 4 \cdot (2 + 2 + 4) + \\ &+ 6 \cdot [2^2 + 2 \cdot (2 \cdot 2) + 2 \cdot (4 \cdot 2)] + \\ &+ 4 \cdot [2^3 + 3 \cdot (2 \cdot 2^2) + 3 \cdot (4 \cdot 2^2)] + \\ &+ 1 \cdot [2^4 + 4 \cdot (2 \cdot 2^3) + 4 \cdot (4 \cdot 2^3)] = \\ =& 4 \cdot 8 + 6 \cdot 28 + 4 \cdot 80 + 1 \cdot 208 = 716 \end{aligned}$$

Αυτές οι 716 συναρτήσεις είναι ο βασικός σκελετός όλων των άλλων συναρτήσεων που θα παράγουμε. Όπως βλέπουμε μόλις υπολογίσουμε το συνολικό πλήθος συναρτήσεων με όλες τις ομάδες ανεβαίνουμε μια τάξη μεγέθους το πλήθος. Το πλήθος των συναρτήσεων αυξάνει υπερβολικά όταν αρχίζουμε να κατασκευάζουμε και συναρτήσεις με συσχετίσεις.

Οι όροι των συσχετίσεων είναι αρκετά πιο πολύπλοκοι σε σχέση με τους βασικούς. Αρχικά δεν μπορούν να έχουν όλες οι συναρτήσεις συσχετίσεις. Συσχετίσεις έχουν οι συναρτήσεις που αποτελούνται από δύο διαφορετικές παραμέτρους και πάνω. Οι συσχετίσεις αφορούν την σχέση που έχουν n παράμετροι μεταξύ τους. Ο βαθμός συσχέτισης μας δείχνει πόσοι όροι εμπλέκονται σε μια συσχέτιση. Ένας όρος συσχέτισης έχει την μορφή:

$$a_i(\prod_{j=1}^{\text{rank}} X_j^{n_{k+j-1}})$$

Όπου τα a και n είναι ελεύθερη παράγοντες, το rank είναι ο βαθμός συσχέτισης και τα X είναι οι παράμετροι μας. Στον πίνακα 5.4 βλέπουμε όλες τις δυνατές συσχετίσεις που μπορούμε να έχουμε από τις παραμέτρους μας.

Το πλήθος των όρων των συσχετίσεων δεν είναι σταθερό για όλες τις συναρτήσεις. Δηλαδή, οι συναρτήσεις μας μπορούν να αποτελούνται από καμιά συσχέτιση, μια συσχέτιση ως έντεκα το πολύ συσχετίσεις. Όταν δεν έχουμε συσχετίσεις στους όρους των συναρτήσεων μας τότε παράγουμε τις συναρτήσεις που έχουμε ήδη δει, μόνο με βασικούς όρους.

Rank	**Correlations**
2	$a_i(X_1^{n_j} \cdot X_2^{n_{j+1}})$
	$a_i(X_1^{n_j} \cdot X_3^{n_{j+1}})$
	$a_i(X_1^{n_j} \cdot X_4^{n_{j+1}})$
	$a_i(X_2^{n_j} \cdot X_3^{n_{j+1}})$
	$a_i(X_2^{n_j} \cdot X_4^{n_{j+1}})$
	$a_i(X_3^{n_j} \cdot X_4^{n_{j+1}})$
3	$a_i(X_1^{n_j} \cdot X_2^{n_{j+1}} \cdot X_3^{n_{j+2}})$
	$a_i(X_1^{n_j} \cdot X_2^{n_{j+1}} \cdot X_4^{n_{j+2}})$
	$a_i(X_1^{n_j} \cdot X_3^{n_{j+1}} \cdot X_4^{n_{j+2}})$
	$a_i(X_2^{n_j} \cdot X_3^{n_{j+1}} \cdot X_4^{n_{j+2}})$
4	$a_i(X_1^{n_j} \cdot X_2^{n_{j+1}} \cdot X_3^{n_{j+2}} \cdot X_4^{n_{j+3}})$

Πίνακας 5.4: Όλοι οι δυνατοί συνδυασμοί από συσχετίσεις.

Οι όροι των συσχετίσεων που μπορεί να έχει μια συνάρτηση εξαρτάται από το μέγιστο βαθμό συσχέτισης που μπορεί να έχει μια συνάρτηση.

Ας χρησιμοποιήσουμε το προηγούμενο παράδειγμα που είχαμε δώσει για τους βασικούς όρους, για να γίνει πιο κατανοητή η δημιουργία συναρτήσεων με συσχετίσεις. Άρα οι πιθανές συσχετίσεις που μπορούμε να έχουμε είναι οι:

- $a_i(X_1^{n_1} \cdot X_2^{n_2})$
- $a_i(X_1^{n_1} \cdot X_3^{n_2})$
- $a_i(X_2^{n_1} \cdot X_3^{n_2})$
- $a_i(X_1^{n_1} \cdot X_2^{n_2} \cdot X_3^{n_3})$

Από αυτές τις συσχετίσεις πρέπει να πάρουμε όλους τούς πιθανούς συνδυασμούς που μπορούν να έχουν. Αυτό το κάνουμε για καλύψουμε όλο το εύρος από πιθανές συναρτήσεις. Για το παράδειγμα μας έχουμε 15 διαφορετικούς συνδυασμούς.

Όπως βλέπουμε το πλήθος από συναρτήσεις αυξάνει εκθετικά μόλις αρχίζουμε να χρησιμοποιούμε τις συσχετίσεις. Αυτό έγινε εμφανές στο παράδειγμα που δώσαμε, καθώς από μια βασική συνάρτηση κατασκευάσαμε 15 νέες συναρτήσεις. Τα πράματα διογκώνονται καθώς αυξάνουν οι παράμετροι. Αν έχουμε τέσσερις παραμέτρους, τότε από μια βασική συνάρτηση παράγουμε 2047 νέες συναρτήσεις. Στην συνέχεια θα δούμε κάποια προβλήματα που χρειάστηκε να αντιμετωπίσουμε με το πλήθος και με την φύση αυτών των συναρτήσεων.

Ας δούμε πρώτα τα προβλήματα που προκύπτουν από την φύση αυτών των συναρτήσεων. Για να μπορέσουμε να εκπαιδεύσουμε μια συνάρτηση χρειαζόμαστε να έχουμε τουλάχιστον τόσα σημεία μετρήσεων όσα και οι ελεύθερες μεταβλητές. Μια από τις πιο πολύπλοκες συναρτήσεις που κατασκευάσαμε είναι η:

$$\begin{aligned} F(X_1, X_2, X_3, X_4) =& a_1 log(X_1 + 1)^{n_1} + a_2 X_1^{n_2} + a_3 X_2^{n_3} + a_4 X_3^{n_4} + a_5 X_4^{n_5} + \\ &+ a_6(X_1^{n_6} \cdot X_2^{n_7}) + a_7(X_1^{n_8} \cdot X_3^{n_9}) + a_8(X_1^{n_{10}} \cdot X_4^{n_{11}}) + \\ &+ a_9(X_2^{n_{12}} \cdot X_3^{n_{13}}) + a_{10}(X_2^{n_{14}} \cdot X_4^{n_{15}}) + a_{11}(X_3^{n_{16}} \cdot X_4^{n_{17}}) + \\ &+ a_{12}(X_1^{n_{18}} \cdot X_2^{n_{19}} \cdot X_3^{n_{20}}) + a_{13}(X_1^{n_{21}} \cdot X_2^{n_{22}} \cdot X_4^{n_{23}}) + \\ &+ a_{14}(X_1^{n_{24}} \cdot X_3^{n_{25}} \cdot X_4^{n_{26}}) + a_{15}(X_2^{n_{27}} \cdot X_3^{n_{28}} \cdot X_4^{n_{29}}) + \\ &+ a_{16}(X_1^{n_{30}} \cdot X_2^{n_{31}} \cdot X_3^{n_{32}} \cdot X_4^{n_{33}}) + b \end{aligned}$$

Όπως βλέπουμε εδώ χρειαζόμαστε 16 σημεία για τα a_i, 33 σημεία για τα n_i και 1 σημείο για το b, άρα συνολικά 50 σημεία τουλάχιστον. Αυτό είναι πρόβλημα, γιατί στην διάθεση

μας έχουμε μόλις 64 σημεία για κάθε τοπολογία και εκτέλεση. Δεν μπορούμε να χρησιμοποιήσουμε όλα τα σημεία που έχουμε στην διάθεση μας γιατί χρειάζεται να φυλάξουμε κάποια για τον έλεγχο.

Συνεχίζοντας ας δούμε το πρόβλημα που είχαμε όσον αφορά το πλήθος των συναρτήσεων. Για να κατασκευάσουμε όλες τις συναρτήσεις που θέλαμε, χρειαζόμασταν να γνωρίζουμε ποιες συναρτήσεις έχουν ήδη φτιάξει. Αυτό μας δημιουργούσε πρόβλημα στο πως θα κατασκευάζαμε τον παραγωγό έτσι ώστε να μην τερματίζεται η λειτουργία λόγω έλλειψης μνήμης. Άρα έπρεπε να κατασκευάσουμε τον παραγωγό έτσι ώστε να είναι μη αναδρομικός. Αυτό σαν διαδικασία είναι πιο πολύπλοκη από ότι φαίνεται, καθώς από την φύση του ένας παραγωγός τέτοιου είδους συναρτήσεων τείνει να είναι αναδρομικός.

Ένα άλλο πρόβλημα που δημιουργούν αυτές οι συναρτήσεις προέρχεται από τον συνδυασμό του πλήθους και της φύσης τους. Από τα πειράματα που εκτελέσαμε είδαμε ότι, αν έχουμε συναρτήσεις μέχρι τρεις παραμέτρους τότε κατασκευάζουμε περίπου ~ 1400 συναρτήσεις. Για να εκπαιδεύσουμε και να ελέγξουμε αυτές τις συναρτήσεις χρειαζόμασταν περίπου ~ 8 ώρες. Αυτός ο υπερβολικά μεγάλος χρόνος προκύπτει από το εξής γεγονός:

- Ο αλγόριθμος εκπαίδευσης που χρησιμοποιήσαμε δουλεύει επαναληπτικά. Εμείς μπορούμε να του ορίσουμε τον μέγιστο αριθμό από επαναλήψεις. Για να πάρουμε όσον το δυνατόν περισσότερες συναρτήσεις μπορούσαμε, είχαμε βάλει αριθμό επαναλήψεων 100000, που είναι πολύ μεγαλύτερος από τον προτεινόμενο. Αυτό είχε σαν αποτέλεσμα οι συναρτήσεις που θα αποτύγχαναν στην εκπαίδευση θα έπαιρναν πολύ χρόνο.
- Ο χρόνος εκπαίδευσης είναι μεγάλος λόγω των ελεύθερων μεταβλητών n_i. Επειδή οι μεταβλητές n_i δεν έχουν κάποια συγκεκριμένη τιμή κατά την εκτέλεση της εκπαίδευσης, σε κάθε κύκλο επανάληψης χρειάζεται να υπολογίζονται συνέχεια οι ποσότητες $X_j^{n_i}$. Αυτό είναι μια αρκετά χρονοβόρα διαδικασία.
- Πολλές συναρτήσεις θα αποτύχουν στην εκπαίδευση. Αυτό συμβαίνει λόγο των πολλών όρων που μπορεί να αποτελείται μια συνάρτηση σε συνδυασμό με τις ελεύθερες μεταβλητές n_i. Εδώ αναφερόμαστε κυρίως σε συναρτήσεις με πολλές συσχετίσεις.

Όπως βλέπουμε πολλά από τα προβλήματα που έχουμε να αντιμετωπίσουμε προέρχονται από τις ελεύθερες μεταβλητές n_i. Για να λύσουμε το πρόβλημα με τις ελεύθερες μεταβλητές n_i, αποφασίσαμε να τις δεσμεύσουμε σε ένα διακριτό εύρος τιμών. Το εύρος τιμών που χρησιμοποιήσαμε είναι το [-2, -1.5, -1, -0.5, 0.5, 1, 1.5, 2]. Εδώ ακολουθήσαμε παρόμοια προσέγγιση με αυτήν που είχε ο Calotoiu και οι συνεργάτες του[18]. Το 0 προφανώς δεν υπάρχει μέσα σε αυτό το εύρος τιμών, γιατί αν υπήρχε θα μας δημιουργούσε συναρτήσεις που ήδη έχουμε κατασκευάσει.

Ακόμα περιορίζουμε τις μορφές που μπορεί να πάρει κάποιος όρος, σε αυτές που έχουν δυνάμεις. Οι νέες μορφές που μπορούν να έχουν οι όροι μας είναι:

- Κυρίαρχος
  1. $a_1 X_i^{n_1}$
  2. $a_1 log(X_i + 1)^{n_1}$
  3. $a_1 log(X_i + 1)^{n_1} + a_2 X_i^{n_2}$
- Δευτερεύοντες
  1. $a_1 X_i^{n_1}$

Όπου τα a_i είναι ελεύθερες μεταβλητές πάλι, n_i παίρνουν τις διακριτές τιμές του εύρους και X_i είναι οι παράμετροι. Αυτός ο περιορισμός έγινε για να μην κατασκευάσουμε διπλές συναρτήσεις, αφού το εύρος μας περιέχει την τιμή 1.

Ας δούμε τώρα πως παίρνουν τιμές οι μεταβλητές n_i μέσα στο εύρος που ορίσαμε. Ας αρχίσουμε με τους δευτερεύοντες βασικούς όρους. Ο κάθε δευτερεύον βασικός όρος παίρνει όλες τις τιμές του εύρους, έτσι ώστε να κατασκευάσουμε όλους τους δυνατούς συνδυασμούς. Συνεχίζουμε με τον κυρίαρχο όρο, ο κάθε υποόρος του κυρίαρχου όρου παίρνει όλες τις τιμές του εύρος, έτσι ώστε να κατασκευάσουμε όλους τους δυνατούς συνδυασμούς που μπορεί να έχει ο κυρίαρχος όρος.

Τέλος, όσον αφορά τις συσχετίσεις, εδώ δεν παίρνει ο κάθε όρος όλες τις δυνατές τιμές του εύρους. Η τιμή που παίρνει ο κάθε όρος μιας συσχέτισης είναι ίση με την τιμή που έχει ο κυρίαρχος όρος στην συνάρτηση. Στην περίπτωση του κυρίαρχου όρου που αποτελείται από δύο όρους, η τιμή που παίρνει η συσχέτιση αυτού του όρου, είναι η τιμή του πρώτου. Για παράδειγμα:

$$\begin{aligned} F(X_1, X_2, X_3, X_4) =& a_1 log(X_1 + 1)^1 + a_2 X_1^2 + a_3 X_2^{0.5} + a_4 X_3^{-2} + a_5 X_4^{0.5} + \\ &+ a_6 (X_1^1 \cdot X_4^{0.5}) + \\ &+ a_7 (X_1^1 \cdot X_2^{0.5} \cdot X_3^{-2} \cdot X_4^{0.5}) + b \end{aligned}$$

Με αυτήν την τεχνική καταφέραμε να αντιμετωπίσουμε:

- Το πρόβλημα που είχαμε με τα ελάχιστα σημεία για την εκπαίδευση.
- Δεν χρειαζόμαστε παραπάνω επαναλήψεις πέρα των προκαθορισμένων για την εκπαίδευση.
- Ο χρόνος εκπαίδευσης μειώθηκε γιατί πλέον δεν υπολογίζουμε σε κάθε επανάληψη τις ποσότητες $X_j^{n_i}$, παρά μόνο μια φορά στην αρχή της εκπαίδευσης.
- Δεν έχουμε συναρτήσεις που αποτυγχάνουν πλέον στην εκπαίδευση.

Όπως βλέπουμε λύσαμε πολλά προβλήματα που είχαμε με αυτήν την τεχνική, όμως διογκώσαμε το πρόβλημα με το πλήθος των συναρτήσεων, αφού πλέον μια συνάρτηση παίρνει πολλές παρόμοιες μορφές.

Για να αντιμετωπίσουμε το νέο πρόβλημα που προέκυψε, αποφασίσαμε να περιορίσουμε το μέγιστο πλήθος από όρους που μπορεί να αποτελείται μια συνάρτηση. Το μέγιστο πλήθος από όρους είναι το πολύ 7. Εδώ ο κυρίαρχος όρος αν αποτελείται από δύο υποόρους τότε μετράει για δύο. Ακόμα κατασκευάσαμε το παραγωγό έτσι ώστε να μην είναι αναδρομικός.

Κάνοντας χρήση όλων των παραπάνω κατασκευάσαμε και ελέγξαμε περίπου ~ 15 εκατομμύρια συναρτήσεις σε περίπου ~ 8 ώρες. Εδώ παραλληλοποιήσαμε κάποια κομμάτια της εκτέλεσης όμως και πάλι τα αποτελέσματα, όσον αφορά το χρόνο εκπαίδευσης, είναι πολύ καλά. Στο παράρτημα Β παραθέτουμε τον κώδικα του παραγωγού πρώτης γενιάς συναρτήσεων, γραμμένο στην γλώσσα προγραμματισμού Python.

5.2.2 Παραγωγός συναρτήσεων δεύτερης γενιάς

Εδώ θα ασχοληθούμε με το επόμενο κομμάτι παραγωγής συναρτήσεων, τον παραγωγό δεύτερης γενιάς. Ο παραγωγός συναρτήσεων δεύτερης γενιάς είναι εξίσου σημαντικός με τον παραγωγό συναρτήσεων πρώτης γενιάς, γιατί είναι αυτός που θα μας δώσει τα τελικά αποτελέσματα στην έρευνα μας.

Για να καταλάβουμε πως λειτουργεί ο παραγωγός δεύτερης γενιάς πρέπει να έχουμε καλή εικόνα για το τι συναρτήσεις κατασκευάζει ο πρώτος παραγωγός. Αυτό συμβαίνει γιατί οι συναρτήσεις που θα κατασκευάσουμε εδώ έχουν άμεση σύνδεση με της συναρτήσεις που κατασκευάσαμε πριν.

Οι συναρτήσεις που κατασκεύαζε ο πρώτος παραγωγός χρησιμοποιούσαν τις παραμέτρους που είχαμε μετρήσει για τις εφαρμογές μας. Όμως δεν λαμβάνει υπόψιν του άμεσα,

στο στάδιο της εκπαίδευσης, τις παραμέτρους της απόσταση της μνήμης και του ποσοστού της μνήμης που έχει έχει ο κάθε κόμβος. Αυτές τις παραμέτρους πριν τις λαμβάναμε υπόψιν έμμεσα. Εδώ θα επεκτείνουμε το μοντέλο για της λαβαίνουμε άμεσα.

Για να το κάνουμε αυτό κατασκευάσουμε δύο νέες παραμέτρους, τις D_0 και D_1. Η D_0 αφορά το ποσοστό της μνήμης που έχουμε στους κόμβους και η D_1 αφορά την απόσταση των κόμβων. Στους πίνακες 5.5 και 5.6 βλέπουμε την κωδικοποίηση που κάναμε σε αυτές τις μεταβλητές. Όπως βλέπουμε η κωδικοποίηση που έχουμε κάνει στις νέες παραμέτρους μας είναι αρκετά απλή, όμως είναι πολύ αποδοτική για της περιγράψει.

Στην συνέχεια θα δούμε πως κατασκευάζουμε τις τελικές συναρτήσεις που θα χρησιμοποιήσουμε. Έστω ότι η συνάρτηση του παραγωγού πρώτης γενιάς:

$$y_i = F_i(X_1, X_2, X_3, X_4)$$

έχει ολοκληρώσει την εκπαίδευση της και έχει περάσει όλα τα κριτήρια που θέλουμε. Τότε η νέα συνάρτηση που κατασκευάζει ο παραγωγός δεύτερης γενιάς είναι:

$$\begin{aligned} Y_i =& y_i + D_0 \cdot (y_i) + D_1 \cdot (y_i) = \\ =& F_i(X_1, X_2, X_3, X_4) + D_0 \cdot [F_i(X_1, X_2, X_3, X_4)] + D_1 \cdot [F_i(X_1, X_2, X_3, X_4)] \end{aligned}$$

D_0	**Numactl**
0.5	interleave
1	remote

Πίνακας 5.5: Κωδικοποίηση για την παράμετρο D_0

D_1	**Machine**	**Topology**
21	Sandman	1
30		2
21	Broady	1

Πίνακας 5.6: Κωδικοποίηση για την παράμετρο D_1

Όπως είδαμε ο τρόπος με τον οποίο κατασκευάζει ο παραγωγός δεύτερης γενιάς τις συναρτήσεις είναι πολύ πιο απλός σε σχέση με της πρώτης γενιάς. Όμως η εξάρτηση μεταξύ τους είναι αρκετά ισχυρή. Εδώ χρειαζόμαστε τριπλάσιο πλήθος από σημεία για να μπορέσουμε να εκπαιδεύσουμε αυτές τις συναρτήσεις, όμως αυτό δεν είναι πρόβλημα γιατί σε αυτό το στάδιο έχουμε περισσότερα σημεία στην διάθεση μας. Τέλος ούτε ο χρόνος εκτέλεσης είναι πρόβλημα, καθώς έχουμε μικρότερο πλήθος συναρτήσεων να ελέγξουμε σε σχέση με πριν.

5.2.3 Η μέθοδος του cross validation

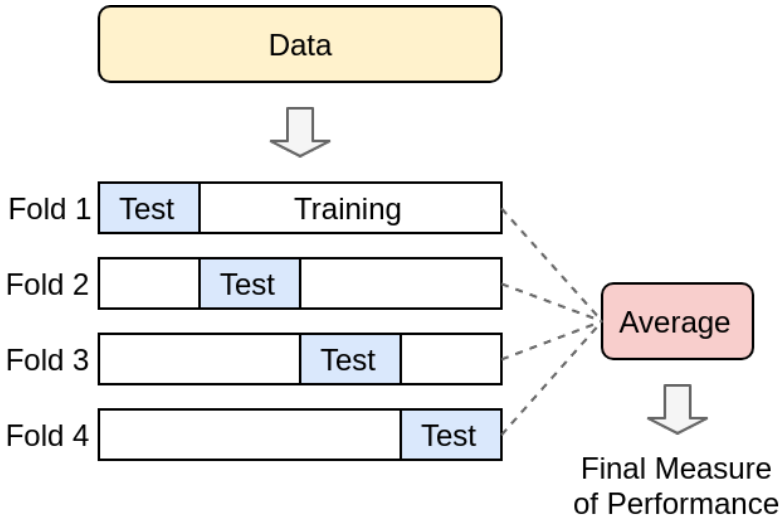
Σε αυτό το υποκεφάλαιο θα δούμε με ποίον τρόπο εκπαιδεύσαμε και ελέγξαμε τις συναρτήσεις μας. Η μέθοδός που αποφασίσαμε να χρησιμοποιήσουμε είναι το cross validation. Αυτή η μέθοδος είναι αρκετά διαδεδομένη στην έρευνα, όταν θέλουμε να ελέγξουμε ένα μοντέλο που έχουμε κατασκευάσει βάσει κάποιον περιορισμένων μετρήσεων για το αν ανταποκρίνεται στην πραγματικότητα. Τη μέθοδο του cross validation την χρησιμοποιήσαμε για να εκπαιδεύσουμε και να ελέγξουμε τις συναρτήσεις που μας κατασκευάζουν και οι δύο παραγωγοί.

Η μέθοδος του cross validation βασίζεται σε μια πολύ απλή και αποδοτική ιδέα. Έστω ότι έχουμε p αντικείμενα. Αυτά τα p αντικείμενα τα χωρίζουμε σε n ομάδες, όπου κάθε ομάδα αποτελείται από k στοιχεία. Ο διαχωρισμός των στοιχειών σε ομάδες γίνεται με τυχαίο τρόπο. Αυτή η μέθοδος του cross validation είναι γνωστή ως k-fold cross-validation[7]. Το κάθε στοιχείο μπορεί να ανήκει μόνο σε μια ομάδα. Γενικά προσπαθούμε να διατηρούμε την σχέση $p = n \cdot k$, όσο περισσότερο είναι δυνατόν έτσι ώστε να υπάρχει ομοιομορφία στην μέθοδο.

Αφού κατασκευάσουμε τις n ομάδες, τις χρησιμοποιούμε για να εκπαιδεύσουμε και να ελέγξουμε τις συναρτήσεις μας. Η διαδικασία με την οποία το κάνουμε αυτό είναι η εξής:

1. Επιλέγουμε την i ομάδα για να κάνουμε τον έλεγχο.
2. Μαζεύουμε όλες τις άλλες $n - 1$ ομάδες για να κάνουμε την εκπαίδευση.
3. Εκπαιδεύουμε την συνάρτηση.
4. Ελέγχουμε την εκπαίδευση που κάναμε χρησιμοποιώντας την i ομάδα.
5. Δίνουμε μια βαθμολογία για το πόσο επιτυχής ήταν ο έλεγχος.

Αυτήν την διαδικασία την επαναλαμβάνουμε για $\forall i$ στο $[0, n]$. Αφού ολοκληρώσουμε τις n επαναλήψεις πλήρως, τότε για κάθε συνάρτηση παίρνουμε τον μέσο όρο των n βαθμολογίων που έχει. Αυτή είναι και η τελική βαθμολογία που έχει η συνάρτηση μας. Στο σχήμα 5.4 βλέπουμε την γραφική αναπαράσταση της μεθόδου.



Σχήμα 5.4: Γραφική αναπαράσταση της μεθόδου cross validation

Ο κύριος λόγος που αποφασίσαμε να χρησιμοποιήσουμε την μέθοδο cross validation, είναι για να αποφύγουμε τις συναρτήσεις που μπορεί να κάνουν overfitting. Όταν λέμε ότι μια συνάρτηση κάνει overfitting σημαίνει ότι, η συνάρτηση που ελέγξαμε έχει εξειδικευτεί πάρα πολύ στην συγκεκριμένη ομάδα μετρήσεων που αν του δώσουμε μια ομάδα ελάχιστα διαφορετική για να την ελέγξουμε τότε θα έχουμε τεράστιο σφάλμα. Προφανώς κάτι τέτοιο δεν το θέλουμε να συμβαίνει, γιατί αν συνέβαινε δεν θα μπορούσαμε να χρησιμοποιήσουμε της συναρτήσεις μας για προβλέπουμε με ακρίβεια την αλλαγή της επίδοση των εφαρμογών που μας ενδιέφεραν.

5.3 Μοντελοποίηση

Εδώ θα δούμε τον τρόπο που μοντελοποιήσαμε το πρόβλημα της πρόβλεψης της απόδοσης των εφαρμογών μας. Αυτό είναι το σημαντικότερο κομμάτι της έρευνας μας, για να αξιολογήσουμε την μεθοδολογία που ακολουθήσαμε.

Αρχικά το πρώτο πράγμα που έχουμε να κάνουμε είναι να μαζέψουμε και να ταξινομήσουμε όλα τα δεδομένα μας. Η ταξινόμηση γίνεται με βάση το μηχάνημα, την τοπολογία και την έκδοση. Υπολογίζουμε για κάθε μη τοπική τοπολογία την απόδοση όλων των εφαρμογών. Στην συνέχεια κατασκευάζουμε τις ομάδες μετρήσεων ως εξής: αντιστοιχούμε με κάθε απόδοση, που έχουμε υπολογίσει, τις παραμέτρους της εφαρμογής από της τοπικές τοπολογίες. Οπότε καταλήγουμε να έχουμε ομάδες που αποτελούνται από στοιχεία της μορφής:

$$\{\text{Performance (i, j), MPKI(local), IPC(local), TLB(local), BW(local)}\}$$

Όπου το i δηλώνει την μη τοπική τοπολογία και το j μας δείχνει τον τρόπο εκτέλεσης. Για το μηχάνημα Sandman έχουμε τέσσερις ομάδες από μετρήσεις ενώ για το μηχάνημα Broady έχουμε δύο.

Αφού έχουμε φτιάξει τις ομάδες από μετρήσεις, χρησιμοποιούμε την μέθοδο του cross validation για να χωρίσουμε την κάθε ομάδα από μετρήσεις σε 4 ομάδες διαχωρισμού με 16 στοιχεία η κάθε μια. Συνεχίζοντας, κατασκευάζουμε τις συναρτήσεις από τον παραγωγό πρώτης γενιάς.

Για κάθε συνάρτηση που έχουμε κατασκευάσει, την εκπαιδεύουμε σε κάθε μια από τις ομάδες διαχωρισμού για όλες τις ομάδες μετρήσεων. Την εκπαίδευση τον συναρτήσεων την κάνουμε με την συνάρτηση curve_fit[8] από το πακέτο scipy.optimize[9] της Python. Έπειτα χρησιμοποιούμε την εκάστοτε ομάδα ελέγχου για να δούμε πόσο καλά μπορεί να την προβλέψει.

Για την βαθμολόγηση της πρόβλεψης χρησιμοποιούμε τις συναρτήσεις r2_score[10] και mean_absolute_error[11] από το πακέτο sklearn.metrics[13] της Python. Η συνάρτηση r2_score μας δίνει την μετρική R^2, η οποία μας δείχνει τον συντελεστή προσδιορισμού. Οι τιμές που παίρνει είναι $(-\infty, 1]$. Όσο πιο κοντά είμαστε στην μεγίστη τιμή (1), τόσο πιο ακριβείς είναι οι προβλέψεις μας. Ουσιαστικά το R^2 μας δείχνει πόσο καλά κάναμε την μοντελοποίηση και πόσο ποσοστό πληροφορίας χάσαμε σε αυτήν. Η συνάρτηση mean_absolute_error μας δίνει το μέσο απόλυτο σφάλμα μεταξύ τις πρόβλεψης και της πραγματική τιμής. Οι τιμές που παίρνει είναι $[0, \infty)$. Όσο πιο κοντά είμαστε στην ελάχιστη τιμή (0), τόσο πιο ακριβείς είναι οι προβλέψεις μας.

Αν μια συνάρτηση έχει βαθμολογία πρόβλεψης:

$$R^2 \geq 0.75 \text{ και } \text{MAE} \leq 15$$

τότε περνάει στην επόμενη φάση. Αποθηκεύουμε όλες τις προβλέψεις από τις συναρτήσεις που περνάνε στην επόμενη φάση σε προσωρινά αρχεία, ανάλογα το μηχάνημα, την τοπολογία και την έκδοση. Αφού ολοκληρώσουν όλες οι συναρτήσεις την εκπαίδευση τους και περάσουν το στάδιο της βαθμολόγησης, τότε αρχίζουμε να μαζεύουμε όλα τα αποτελέσματα από τις προβλέψεις που έχει η κάθε μια συνάρτηση σε κάθε ομάδα διαχωρισμού, για κάθε τοπολογία και κάθε έκδοση. Έπειτα υπολογίζουμε εκ νέου την μετρική R^2. Αν μια συνάρτηση δεν έχει αποτελέσματα για την πρόβλεψη έστω και σε μιας ομάδας διαχωρισμού, επειδή δεν πέρασε τον απαραίτητο έλεγχο, τότε απορρίπτεται πλήρως.

Με την νέα μετρική R^2_{new} ελέγχουμε αν ισχύει η συνθήκη, για κάθε μια συνάρτηση:

$$R^2_{new} \geq 0.8$$

Αν ναι, τότε αυτήν την συνάρτηση την αποθηκεύουμε σε ένα άλλο προσωρινό αρχείο. Αυτές τις συναρτήσεις θα τις χρησιμοποιήσουμε για να κατασκευάσουμε τις συναρτήσεις από τον παραγωγό δεύτερης γενιάς. Όπως βλέπουμε μια συνάρτηση για να φτάσει στο τελικό στάδιο χρειάζεται να περάσει επιτυχώς από αρκετούς έλεγχους. Ο λόγος που το κάνουμε αυτό είναι για να αποφύγουμε το overfitting.

Στην συνέχεια κατασκευάζουμε τις νέες ομάδες μετρήσεων που θα χρησιμοποιηθούν από τις συναρτήσεις του παραγωγού δεύτερης γενιάς. Σε αυτό το στάδιο της εκτέλεσης κάθε μηχάνημα έχει μόνο μια ομάδα. Αυτή η νέα ομάδα είναι μια συγχώνευση των προηγούμενων ομάδων μετρήσεων. Οι νέες ομάδες έχουν την μορφή:

$$\{\text{Performance (i, j), MPKI (local), IPC(local), TLB(local), BW(local)}, D_0, D_1\}$$

Όπου το i δείχνει την μη τοπική τοπολογία και το j μας δείχνει τον τρόπο εκτέλεσης, για όλες τις μη τοπικές τοπολογίες και όλους τους τρόπους εκτέλεσης. Στον πίνακα 4.2 βλέπουμε όλους τους δυνατούς συνδυασμούς για κάθε μηχάνημα, αγνοώντας τις τοπικές τοπολογίες.

Έπειτα χωρίζουμε τις νέες ομάδες μετρήσεων σε νέες ομάδες διαχωρισμού. Για τον διαχωρισμό πάλι χρησιμοποιούμε την μέθοδο του cross validation, όμως εδώ χρειάζεται να προσέξουμε στον τρόπο που χωρίζουμε τις νέες ομάδες. Αν αποφασίσουμε να βάλουμε σε μια ομάδα το benchmark A, τότε πρέπει να βάλουμε στην ίδια ομάδα όλες τις μετρήσεις του από όλες τις μη τοπικές τοπολογίες και εκτελέσεις. Αυτό συμβαίνει γιατί χρειάζεται να υπάρχει ομοιομορφία μεταξύ των ομάδων, δηλαδή δεν πρέπει να κατασκευάσουμε ομάδες που η μια να περιέχει όλες τις interleave εκτελέσεις και οι άλλες να μην έχουν καμία. Για λόγους συνάφειας με τις προηγούμενες ομάδες διαχωρισμού αποφασίσαμε να χωρίσουμε και εδώ τις νέες ομάδες μετρήσεων σε τέσσερις νέες ομάδες διαχωρισμού.

Αφού φτιάξουμε τις νέες ομάδες διαχωρισμού, κατασκευάζουμε τις συναρτήσεις από τον παραγωγό δεύτερης γενιάς, χρησιμοποιώντας τα προσωρινά αρχεία που είχαμε φτιάξει. Έπειτα ακολουθούμε την ίδια διαδικασία με πριν. Εκπαιδεύουμε μια μια τις συναρτήσεις μας για όλες τις ομάδες διαχωρισμού και τις ελέγχουμε με την εκάστοτε ομάδα ελέγχου. Στο στάδιο της βαθμολόγησης χρησιμοποιούμε ακριβώς την ίδια συνθήκη με πριν για να δούμε ποσό καλά μπορεί η συνάρτηση μας να προβλέψει την ομάδα ελέγχου. Αν η βαθμολόγηση της ήταν επιτύχεις, τότε αποθηκεύουμε την πρόβλεψης της σε ένα τελικό αρχείο.

Όταν τελειώσουν όλες οι συναρτήσεις με το στάδιο της βαθμολόγησης, έχουμε πρακτικά ολοκληρώσει την μοντελοποίηση. Το τελευταίο πράγμα που έχουμε να κάνουμε είναι να μαζέψουμε όλες τις προβλέψεις από τις επιτύχεις συναρτήσεις και να τις ταξινομήσουμε βάση του πόσο καλά μπορούν να κάνουν πρόβλεψη. Για να το κάνουμε αυτό χρησιμοποιούμε το R^2.

Όπως είδαμε μια συνάρτηση για να καταλήξει στα τελικά μας αποτελέσματα χρειάζεται να περάσει από πολλά στάδια αξιολόγησης. Έτσι πάρουμε συναρτήσεις που δεν έχουν το πρόβλημα του overfitting και μπορούν αν προβλέψουν αρκετά ικανοποιητικά την απόδοση των εφαρμογών μας.

5.4 Αποτελέσματα

Εδώ θα δούμε τα αποτελέσματα που πήραμε από την μοντελοποίηση που κάναμε στο κεφάλαιο 5.3. Στον πίνακα 5.7 βλέπουμε το πλήθος των επιτυχημένων συναρτήσεων για

μετά από την μοντελοποίηση για τα δύο μηχανήματα και για τις δύο ομάδες από παραμέτρους.

Parameters	Machine	Sucesses
MPKI IPC TLB BW	Sandman	28560
	Broady	410

Πίνακας 5.7: Πλήθους επιτυχημένων συναρτήσεων

Για την αξιολόγηση των αποτελεσμάτων χρησιμοποιήσαμε της μετρικές:

- R^2
- Var (Variance)
- MEA (Mean Absolute Error)
- Min (Min Error)
- Max (Max Error)

Για να υπολογίσουμε τις μετρικές R^2, Var και MEA (Real) χρησιμοποιήσαμε της συναρτήσεις r2_score, explained_variance_score[12] και mean_absolute_error από το πακέτο sklearn.metrics της Python. Τις μετρικές R^2 και MEA τις έχουμε ήδη αναλύσει. Η μετρική Var μας δείχνει την διασπορά της R^2. Για τις μετρικές MEA, Min και Max έχουμε υπολογίσει το πραγματικό και το σχετικό σφάλμα για κάθε συνάρτηση.

Στους πίνακες 5.9 και 5.10 βλέπουμε τα αποτελέσματα που πήραμε από τις 50 καλύτερες συναρτήσεις για τα μηχανήματα Sandman και Broady αντίστοιχα, για τις παραμέτρους MPKI, IPC, TLB και BW. Όλοι οι πίνακες είναι ταξινομημένοι βάση το R^2.

Οι συναρτήσεις που βλέπουμε στους πίνακες 5.9 - 5.12 είναι οι συναρτήσεις που κατασκευάζει ο πρώτος παραγωγός. Αυτό το κάναμε για να μπορέσουν να χωρέσουν τα αποτελέσματα μας σε μια σελίδα. Αν θέλουμε να πάρουμε τις πραγματικές συναρτήσεις μπορούμε να το κάνουμε χρησιμοποιώντας την τεχνική του δεύτερου παραγωγού. Για παράδειγμα η συνάρτηση:

$$\begin{aligned} y =& a_0 \cdot log(x_2+1)^{-1.5} + a_1 \cdot x_2^{-2.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{-0.5} + a_4 \cdot x_4^{-0.5} + \\ &+ a_5 \cdot (x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + b \end{aligned}$$

Γίνεται:

$$\begin{aligned} Y =& a_0 \cdot log(x_2+1)^{-1.5} + a_1 \cdot x_2^{-2.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{-0.5} + a_4 \cdot x_4^{-0.5} + \\ &+ a_5 \cdot (x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + b_0 + \\ &+ D_0(a_7 \cdot log(x_2+1)^{-1.5} + a_8 \cdot x_2^{-2.0} + a_9 \cdot x_1^{1.0} + a_{10} \cdot x_3^{-0.5} + a_{11} \cdot x_4^{-0.5} + \\ &+ a_{12} \cdot (x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + a_{13} \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + b_1) + \\ &+ D_1(a_{14} \cdot log(x_2+1)^{-1.5} + a_{15} \cdot x_2^{-2.0} + a_{16} \cdot x_1^{1.0} + a_{17} \cdot x_3^{-0.5} + a_{18} \cdot x_4^{-0.5} + \\ &+ a_{19} \cdot (x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + a_{20} \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + b_2) \end{aligned}$$

Ας σχολιάσουμε τα αποτελέσματα μας. Αρχικά παρατηρούμε ότι δεν έχουμε τον ίδιο αριθμό από επιτυχημένες συναρτήσεις και στα δυο μηχανήματα. Αυτό το γεγονός είναι αρκετά αναμενόμενο, αφού τα δυο μηχανήματά που χρησιμοποιήσαμε στην έρευνα μας έχουνε σημαντικές αρχιτεκτονικές διαφορές, όπως την LLC και αριθμό από κόμβους.

Συνεχίζοντας όπως βλέπουμε στους πίνακες του κάθε μηχανήματος, παρατηρούμε ότι υπάρχει μεγάλη ομοιότητα στις συναρτήσεις που πέτυχαν με το καλύτερο R^2. Δηλαδή οι κύριοι όροι των καλύτερων συναρτήσεων παραμένουν σχετικά σταθεροί και έχουμε

αλλαγές στις δυνάμεις τους ή στις συσχετίσεις. Αυτό το βλέπουμε σε όλους τους πίνακες των αποτελεσμάτων μας.

Τέλος ας δούμε τις βαθμολογίες που πέτυχαν οι συναρτήσεις μας. Σε όλα τα αποτελέσματα μας έχουμε $R^2 > 0.93$ και MAE (Relative) $< 5\%$, για τις καλύτερες συναρτήσεις. Όμως, αντίστοιχα καλά αποτελέσματα βλέπουμε και για τις υπόλοιπες συναρτήσεις. Ακόμα αξίζει να πούμε ότι και οι υπόλοιπες μετρικές που έχουμε χρησιμοποιήσει είναι πολύ καλές. Στο σχήμα 5.5 βλέπουμε την γραφική αναπαράσταση της καλύτερης συνάρτησης για τις προβλέψεις που έκανε για το κάθε σημείο για το κάθε μηχάνημα.

Στον πίνακα 5.8 βλέπουμε πόσες συναρτήσεις κατάφεραν να περάσουν όλους τους ελέγχους μας σε σχέση με τον πλήθος των παραμέτρων τους. Όπως βλέπουμε από τον πίνακα καμία συνάρτηση δεν κατάφερε να περάσει όλους τους έλεγχους που να αποτελείτε μόνο από μια παράμετρο σε κανένα από τα δύο μηχανήματα. Ακόμα το πλήθος των συναρτήσεων για δύο παραμέτρους ήταν ελάχιστο για το μηχάνημα Sandman ενώ για το Broady δεν είχαμε καμιά επιτυχία. Στον πίνακα 5.11 βλέπουμε τις 20 καλύτερες συναρτήσεις με 3 και 2 παραμέτρους για το μηχάνημα Sandman. Στον πίνακα 5.12 βλέπουμε τις 20 καλύτερες συναρτήσεις με 3 παραμέτρους για το μηχάνημα Broady. Όπως βλέπουμε και στα δύο μηχανήματα έχουμε καταφέρει να πετύχουμε αρκετά υψηλό R^2, όμως έχουμε μεγαλύτερα σφάλματα.

Number of Parameters	**Sandman**	**Broady**
4	27540	386
3	1015	24
2	5	0
1	0	0

Πίνακας 5.8: Ταξινόμηση των επιτυχημένων συναρτήσεων βάση το πλήθος των παραμέτρων τους.

Top	Function	R^2	Var	Real			Relative		
				MAE	Min	Max	MAE	Min	Max
1	$a_0 \cdot log(x_2+1)^{-1.5} + a_1 \cdot x_2^{-2.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{-0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + b$	0.9379	0.9380	6.3635	0.0056	58.2456	4.86	0.01	29.85
2	$a_0 \cdot log(x_2+1)^{-1.5} + a_1 \cdot x_2^{-1.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{-0.5} + a_4 \cdot x_4^{-0.3} + a_5 \cdot (x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5} \cdot x_4^{-0.5}) + b$	0.9356	0.9357	6.5290	0.0082	57.9716	4.99	0.01	30.38
3	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{-0.5} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_4^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{-0.5}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{-0.5} \cdot x_3^{0.5} \cdot x_4^{-0.5}) + b$	0.9337	0.9337	5.9507	0.0717	51.2511	4.34	0.07	36.05
4	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{1.0} \cdot x_3^{0.5}) + b$	0.9332	0.9332	5.8622	0.0523	56.5635	4.22	0.05	30.26
5	$a_0 \cdot log(x_1+1)^{1.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_1^{1.0} \cdot x_2^{1.0}) + a_6 \cdot (x_1^{1.0} \cdot x_2^{1.0} \cdot x_3^{0.5} \cdot x_4^{-0.5}) + b$	0.9323	0.9324	5.6965	0.0130	56.3286	3.97	0.01	32.08
6	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{1.0} \cdot x_3^{0.5} \cdot x_4^{-0.5}) + b$	0.9300	0.9303	5.8957	0.0640	56.9392	4.05	0.06	29.43
7	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_6 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9299	0.9300	6.1335	0.0111	55.1378	4.47	0.01	38.79
8	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9298	0.9299	5.9157	0.0090	55.9132	4.27	0.01	39.33
9	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{2.0} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_2^{-0.5} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{0.5} \cdot x_2^{-0.5}) + a_6 \cdot (x_3^{0.5} \cdot x_1^{0.5} \cdot x_2^{-0.5}) + b$	0.9292	0.9293	5.8549	0.0090	56.4805	4.15	0.01	39.73
10	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{1.5} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_2^{0.5} + a_4 \cdot x_4^{-1.5} + a_5 \cdot (x_1^{0.5} \cdot x_2^{0.5}) + a_6 \cdot (x_3^{0.5} \cdot x_1^{0.5} \cdot x_2^{0.5}) + b$	0.9290	0.9290	5.9231	0.0069	56.6653	4.22	0.01	39.86
11	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{-2.0} + a_5 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_6 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9290	0.9291	5.9856	0.0627	55.9546	4.33	0.06	39.36
12	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_6 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9288	0.9289	6.0157	0.0068	55.8792	4.36	0.01	39.31
13	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_6 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9287	0.9288	6.0847	0.0067	55.6730	4.42	0.01	39.16
14	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{2.0} + a_2 \cdot x_2^{0.5} + a_3 \cdot x_2^{0.5} + a_4 \cdot x_4^{-2.0} + a_5 \cdot (x_2^{0.5} \cdot x_2^{0.5}) + a_6 \cdot (x_3^{0.5} \cdot x_1^{0.5} \cdot x_2^{0.5}) + b$	0.9284	0.9285	5.9958	0.0062	56.6949	4.28	0.01	39.88
15	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{1.0} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_6 \cdot (x_1^{0.5} \cdot x_3^{1.0} \cdot x_4^{-0.5}) + b$	0.9284	0.9288	5.8022	0.0067	56.9388	4.08	0.01	29.87
16	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_2^{-0.5} \cdot x_3^{0.5}) + a_5 \cdot (x_2^{-0.5} \cdot x_3^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{-0.5} \cdot x_3^{0.5}) + b$	0.9284	0.9288	6.2072	0.0159	55.9471	4.45	0.02	33.81
17	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{1.5} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_2^{0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_3^{0.5} \cdot x_1^{0.5}) + a_6 \cdot (x_3^{0.5} \cdot x_2^{0.5}) + b$	0.9282	0.9283	5.8470	0.0043	55.7489	4.15	0.00	39.22
18	$a_0 \cdot log(x_1+1)^{1.0} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{-0.5} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_1^{1.0} \cdot x_2^{-0.5}) + a_6 \cdot (x_1^{1.0} \cdot x_3^{0.5} \cdot x_4^{-0.5}) + b$	0.9274	0.9275	5.9933	0.0175	52.9039	4.16	0.02	35.59
19	$a_0 \cdot log(x_2+1)^{0.5} + a_1 \cdot x_2^{2.0} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_4^{0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_2^{0.5} \cdot x_1^{0.5}) + a_6 \cdot (x_2^{0.5} \cdot x_3^{0.5}) + b$	0.9273	0.9276	5.8974	0.0087	59.1657	4.24	0.01	41.62
20	$a_0 \cdot log(x_4+1)^{-0.5} + a_1 \cdot x_4^{-2.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_2^{-0.5} + a_4 \cdot x_3^{0.5} + a_5 \cdot (x_1^{1.0} \cdot x_2^{-0.5}) + a_6 \cdot (x_4^{-0.5} \cdot x_1^{1.0} \cdot x_2^{-0.5} \cdot x_3^{0.5}) + b$	0.9271	0.9276	5.6686	0.0041	56.1358	3.86	0.00	31.26
21	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{0.5} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{1.0} \cdot x_3^{0.5} \cdot x_4^{-0.5}) + b$	0.9271	0.9272	5.9262	0.0148	55.7978	4.02	0.01	28.83
22	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{0.5} + a_3 \cdot x_3^{1.0} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_1^{0.5} \cdot x_2^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{0.5} \cdot x_3^{1.0} \cdot x_4^{-0.5}) + b$	0.9271	0.9272	6.0797	0.0014	55.4085	4.32	0.00	28.73
23	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{2.0} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_2^{-0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_3^{0.5} \cdot x_2^{-0.5}) + a_6 \cdot (x_3^{0.5} \cdot x_1^{0.5} \cdot x_2^{-0.5}) + b$	0.9270	0.9270	5.9130	0.0092	55.8429	4.20	0.01	39.28
24	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{1.5} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_2^{1.0} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_3^{0.5} \cdot x_1^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + b$	0.9266	0.9266	5.9517	0.0826	57.0058	4.23	0.06	40.10
25	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_2^{0.5} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{0.5} \cdot x_2^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{0.5}) + b$	0.9266	0.9268	5.9591	0.0125	56.3737	4.22	0.01	39.66
26	$a_0 \cdot log(x_3+1)^{1.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_1^{1.0} \cdot x_2^{-0.5}) + a_6 \cdot (x_3^{0.5} \cdot x_1^{1.0} \cdot x_2^{-0.5} \cdot x_4^{-0.5}) + b$	0.9263	0.9265	6.4857	0.0219	59.6524	4.64	0.01	27.86
27	$a_0 \cdot log(x_2+1)^{1.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_3^{0.5} + a_3 \cdot x_4^{-0.5} + a_4 \cdot (x_2^{1.0} \cdot x_1^{0.5}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9263	0.9265	5.8820	0.0510	58.2757	4.19	0.05	41.00
28	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{0.5} + a_3 \cdot x_4^{1.0} + a_4 \cdot (x_1^{0.5} \cdot x_2^{0.5}) + a_5 \cdot (x_2^{1.5} \cdot x_3^{1.0}) + b$	0.9263	0.9270	6.0683	0.0513	55.5962	4.38	0.05	29.45
29	$a_0 \cdot log(x_4+1)^{-2.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_4^{-2.0} \cdot x_2^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_6 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9262	0.9263	5.9071	0.0505	57.8763	4.24	0.05	40.71
30	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{-1.0} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_2^{1.0} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_3^{0.5} \cdot x_1^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + b$	0.9262	0.9264	5.9153	0.0140	56.9823	4.22	0.01	40.09
31	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{-1.0} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_2^{1.0} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_3^{0.5} \cdot x_1^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + b$	0.9262	0.9263	5.9342	0.0145	57.0885	4.24	0.01	40.16
32	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{0.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_2^{-0.5} + a_4 \cdot (x_3^{0.5} \cdot x_1^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + b$	0.9262	0.9263	5.9326	0.0119	57.3634	4.25	0.01	40.35
33	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{0.5} + a_3 \cdot x_4^{0.5} + a_4 \cdot (x_3^{0.5} \cdot x_1^{0.5}) + a_5 \cdot (x_1^{0.5} \cdot x_2^{0.5}) + b$	0.9262	0.9263	5.9541	0.0031	57.0505	4.28	0.00	40.13
34	$a_0 \cdot log(x_2+1)^{0.5} + a_1 \cdot x_2^{2.0} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_2^{0.5} \cdot x_3^{0.5}) + a_6 \cdot (x_2^{0.5} \cdot x_4^{0.5}) + b$	0.9262	0.9264	5.9045	0.0032	59.8103	4.22	0.00	42.08
35	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3 + a_2 \cdot x_1^{0.5} + a_3 \cdot x_2^{1.0} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_3^{0.5} \cdot x_1^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + b$	0.9261	0.9261	5.9423	0.0015	57.1922	4.25	0.00	40.23
36	$a_0 \cdot x_1^{0.5} + a_1 \cdot x_2^{1.0} + a_2 \cdot x_3^{0.5} + a_3 \cdot x_4^{-0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9261	0.9262	5.8909	0.0178	58.1019	4.20	0.01	40.87
37	$a_0 \cdot log(x_4+1)^{-1.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_2^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9260	0.9262	5.9171	0.0212	57.8177	4.23	0.02	40.67
38	$a_0 \cdot log(x_4+1)^{-1.5} + a_1 \cdot x_1^{-2.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9260	0.9262	5.8995	0.0077	58.0165	4.21	0.01	40.81
39	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{-2.0} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_4^{1.0} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_3^{0.5} \cdot x_2^{1.0}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + b$	0.9260	0.9261	5.9452	0.0116	57.2549	4.25	0.01	40.28
40	$a_0 \cdot log(x_2+1)^{1.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_3^{0.5} + a_3 \cdot x_4^{-0.5} + a_4 \cdot (x_2^{1.0} \cdot x_1^{0.5}) + a_5 \cdot (x_2^{1.0} \cdot x_4^{-0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9260	0.9261	5.8582	0.0638	58.2565	4.18	0.06	40.98
41	$a_0 \cdot log(x_4+1)^{-2.0} + a_1 \cdot x_4^{-0.5} + a_2 \cdot x_2^{0.5} + a_3 \cdot x_1^{1.0} + a_4 \cdot x_2^{0.5} + a_5 \cdot (x_1^{0.5} \cdot x_1^{1.0}) + a_6 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9258	0.9260	6.0330	0.0261	57.8472	4.34	0.03	40.69
42	$a_0 \cdot log(x_4+1)^{-0.5} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9258	0.9260	5.9490	0.0345	57.6060	4.26	0.03	40.52
43	$a_0 \cdot log(x_4+1)^{-2.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_1^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9258	0.9260	5.8993	0.0003	58.1952	4.20	0.00	40.94
44	$a_0 \cdot log(x_2+1)^{-0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_3^{0.5} + a_3 \cdot x_4^{0.5} + a_4 \cdot (x_2^{-0.5} \cdot x_1^{1.0}) + a_5 \cdot (x_3^{0.5} \cdot x_4^{-0.5}) + a_6 \cdot (x_2^{-0.5} \cdot x_1^{1.0} \cdot x_3^{0.5} \cdot x_4^{-0.5}) + b$	0.9258	0.9262	6.4212	0.0053	60.4535	4.57	0.01	28.23
45	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{2.0} + a_3 \cdot x_3^{1.5} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{0.5} \cdot x_2^{2.0}) + a_6 \cdot (x_1^{0.5} \cdot x_4^{-1.0}) + b$	0.9258	0.9260	5.8817	0.0121	59.1527	4.26	0.01	31.78
46	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{1.5} + a_3 \cdot x_3^{1.5} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{0.5} \cdot x_2^{1.5}) + a_6 \cdot (x_1^{0.5} \cdot x_4^{-1.0}) + b$	0.9258	0.9260	5.9055	0.0179	58.9526	4.26	0.02	31.34
47	$a_0 \cdot log(x_3+1)^{0.5} + a_1 \cdot x_3^{-2.0} + a_2 \cdot x_1^{0.5} + a_3 \cdot x_2^{0.5} + a_4 \cdot x_4^{-0.5} + a_5 \cdot (x_3^{0.5} \cdot x_1^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{0.5}) + b$	0.9257	0.9258	5.9781	0.0127	56.9058	4.29	0.01	40.03
48	$a_0 \cdot log(x_1+1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{2.0} + a_3 \cdot x_1^{1.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{0.5} \cdot x_2^{2.0}) + a_6 \cdot (x_1^{0.5} \cdot x_1^{1.5}) + b$	0.9257	0.9258	5.9688	0.0013	55.2049	4.28	0.00	31.41
49	$a_0 \cdot log(x_2+1)^{-1.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_3^{-0.5} + a_3 \cdot (x_2^{-1.5} \cdot x_1^{1.0}) + a_4 \cdot (x_2^{-1.5} \cdot x_3^{-0.5}) + a_5 \cdot (x_1^{1.0} \cdot x_3^{-0.5}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5}) + b$	0.9257	0.9259	7.3898	0.0019	62.0470	5.47	0.00	31.41
50	$a_0 \cdot log(x_4+1)^{-1.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_5 \cdot (x_4^{-1.0} \cdot x_1^{0.5}) + a_6 \cdot (x_4^{-1.0} \cdot x_1^{0.5} \cdot x_2^{0.5} \cdot x_3^{0.5}) + b$	0.9256	0.9258	5.9105	0.0016	57.7541	4.21	0.00	40.63

Πίνακας 5.9: Αποτελέσματα του μηχανήματος Sandman για τις παραμέτρους MPKI IPC TLB BW.

Top	Function	R^2	Var	Real			Relative		
				MAE	Min	Max	MAE	Min	Max
1	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_2^{-2.0} + a_2 \cdot x_3^{-0.5} + a_3 \cdot x_4^{1.0} + a_4 \cdot (x_1^{2.0} \cdot x_2^{-2.0}) + a_5 \cdot (x_1^{2.0} \cdot x_3^{-0.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_4^{1.0}) + b$	0.9569	0.9569	1.3283	0.0136	7.4566	1.24	0.01	5.74
2	$a_0 \cdot log(x_4+1)^{1.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_2^{1.0} + a_4 \cdot x_3^{2.0} + a_5 \cdot (x_2^{-1.0} \cdot x_3^{2.0}) + a_6 \cdot (x_4^{1.0} \cdot x_1^{1.5} \cdot x_2^{-1.0}) + b$	0.9521	0.9521	1.3272	0.0006	7.6840	1.23	0.00	6.17
3	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_2^{-0.5} + a_2 \cdot x_3^{-1.0} + a_3 \cdot x_4^{1.5} + a_4 \cdot (x_1^{2.0} \cdot x_2^{-0.5}) + a_5 \cdot (x_1^{2.0} \cdot x_3^{-1.0} \cdot x_4^{1.5}) + a_6 \cdot (x_2^{0.5} \cdot x_3^{-1.0} \cdot x_4^{1.5}) + b$	0.9486	0.9487	1.3720	0.0056	10.1817	1.27	0.01	6.51
4	$a_0 \cdot log(x_4+1)^{1.0} + a_1 \cdot x_4^{-1.0} + a_2 \cdot x_1^{1.5} + a_3 \cdot x_2^{-1.0} + a_4 \cdot x_3^{2.0} + a_5 \cdot (x_2^{-1.0} \cdot x_3^{2.0}) + a_6 \cdot (x_4^{1.0} \cdot x_1^{1.5} \cdot x_2^{-1.0}) + b$	0.9469	0.9469	1.3150	0.0030	9.1410	1.21	0.00	7.35
5	$a_0 \cdot x_1^{1.5} + a_1 \cdot x_2^{-0.5} + a_2 \cdot x_3^{2.0} + a_3 \cdot x_4^{0.5} + a_4 \cdot (x_2^{-0.5} \cdot x_3^{2.0}) + a_5 \cdot (x_2^{-0.5} \cdot x_4^{0.5}) + a_6 \cdot (x_1^{1.5} \cdot x_2^{-0.5} \cdot x_4^{0.5}) + b$	0.9444	0.9445	1.3500	0.0001	9.8247	1.24	0.00	6.28
6	$a_0 \cdot log(x_2+1)^{-0.5} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_3^{2.0} + a_3 \cdot x_4^{0.5} + a_4 \cdot (x_2^{0.5} \cdot x_3^{2.0}) + a_5 \cdot (x_2^{0.5} \cdot x_4^{0.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_1^{1.5} \cdot x_4^{0.5}) + b$	0.9443	0.9445	1.3495	0.0003	9.8980	1.24	0.00	6.33
7	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{-1.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{2.0} \cdot x_3^{-1.5}) + a_6 \cdot (x_3^{-1.5} \cdot x_4^{1.5}) + b$	0.9389	0.9401	1.5404	0.0013	8.4853	1.44	0.00	6.29
8	$a_0 \cdot log(x_2+1)^{-1.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_1^{1.5} + a_4 \cdot x_1^{1.5} + a_5 \cdot (x_1^{1.0} \cdot x_3^{1.5}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_4^{1.5}) + b$	0.9384	0.9386	1.3824	0.0118	14.4185	1.26	0.01	9.22
9	$a_0 \cdot log(x_3+1)^{2.0} + a_1 \cdot x_3^{-1.5} + a_2 \cdot x_1^{-1.0} + a_3 \cdot x_2^{-0.5} + a_4 \cdot x_4^{0.5} + a_5 \cdot (x_3^{2.0} \cdot x_2^{-0.5}) + a_6 \cdot (x_1^{1.5} \cdot x_2^{-0.5} \cdot x_4^{0.5}) + b$	0.9379	0.9385	1.4104	0.0174	10.0110	1.29	0.02	7.42
10	$a_0 \cdot log(x_1+1)^{1.5} + a_1 \cdot x_1^{-0.5} + a_2 \cdot x_3^{-1.0} + a_3 \cdot x_4^{1.0} + a_4 \cdot (x_1^{1.5} \cdot x_2^{-0.5}) + a_5 \cdot (x_3^{-1.0} \cdot x_4^{1.0}) + a_6 \cdot (x_1^{1.5} \cdot x_2^{-0.5} \cdot x_3^{-1.0}) + b$	0.9347	0.9353	1.3258	0.0044	16.2228	1.22	0.00	10.37
11	$a_0 \cdot log(x_1+1)^{1.0} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{2.0} + a_4 \cdot (x_2^{1.0} \cdot x_3^{2.0}) + a_5 \cdot (x_4^{1.0} \cdot x_1^{1.5} \cdot x_2^{1.0}) + b$	0.9339	0.9339	1.3937	0.0260	12.5825	1.28	0.03	9.33
12	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_2^{-2.0} + a_2 \cdot x_3^{-2.0} + a_3 \cdot x_4^{-1.0} + a_4 \cdot (x_1^{2.0} \cdot x_4^{-1.0}) + a_5 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_4^{-1.0}) + b$	0.9332	0.9342	1.3849	0.0249	12.2240	1.27	0.02	10.13
13	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_3^{0.5} + a_3 \cdot x_1^{1.5} + a_4 \cdot (x_1^{2.0} \cdot x_2^{0.5}) + a_5 \cdot (x_1^{2.0} \cdot x_4^{1.5}) + a_6 \cdot (x_2^{0.5} \cdot x_3^{-0.5} \cdot x_4^{1.5}) + b$	0.9318	0.9320	1.4766	0.0269	14.0979	1.37	0.03	11.38
14	$a_0 \cdot log(x_3+1)^{1.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{-1.5} + a_3 \cdot x_4^{1.5} + a_4 \cdot (x_3^{1.5} \cdot x_1^{1.0} \cdot x_2^{-1.5}) + a_5 \cdot (x_1^{1.0} \cdot x_2^{-1.5} \cdot x_4^{1.5}) + b$	0.9317	0.9317	1.4621	0.0184	14.4642	1.34	0.02	9.25
15	$a_0 \cdot log(x_3+1)^{1.5} + a_1 \cdot x_1^{-0.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_2^{1.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{1.0} \cdot x_2^{1.5} \cdot x_4^{1.5}) + a_6 \cdot (x_1^{1.0} \cdot x_2^{-2.0} \cdot x_4^{1.5}) + b$	0.9315	0.9315	1.3692	0.0054	15.3848	1.24	0.01	9.84
16	$a_0 \cdot log(x_3+1)^{2.0} + a_1 \cdot x_1^{-2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{2.0} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{2.0} \cdot x_4^{-1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_4^{-1.0}) + b$	0.9315	0.9328	1.4365	0.0004	11.8455	1.31	0.00	8.62
17	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{-0.5} + a_2 \cdot x_3^{1.0} + a_3 \cdot x_4^{0.5} + a_4 \cdot (x_3^{2.0} \cdot x_4^{0.5}) + a_5 \cdot (x_1^{2.0} \cdot x_3^{1.0} \cdot x_4^{0.5}) + b$	0.9313	0.9313	1.4101	0.0334	9.4848	1.30	0.03	8.38
18	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_4^{1.0} + a_4 \cdot (x_1^{2.0} \cdot x_4^{1.0}) + a_5 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_4^{1.0}) + b$	0.9313	0.9324	1.4169	0.0020	12.2978	1.29	0.00	8.94
19	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{-1.0} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{2.0} \cdot x_3^{-1.0} \cdot x_4^{1.5}) + b$	0.9311	0.9312	1.5126	0.0180	12.0506	1.40	0.02	8.94
20	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{1.5} + a_4 \cdot x_4^{1.0} + a_5 \cdot (x_1^{2.0} \cdot x_3^{1.5} \cdot x_4^{1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_3^{1.5} \cdot x_4^{1.0}) + b$	0.9303	0.9315	1.4030	0.0084	12.0356	1.28	0.01	10.05
21	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{-1.0} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{2.0} \cdot x_4^{-1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_4^{-1.0}) + b$	0.9299	0.9310	1.3999	0.0299	12.2669	1.28	0.03	10.16
22	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{1.5} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{2.0} \cdot x_4^{-1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_4^{-1.0}) + b$	0.9298	0.9310	1.4423	0.0068	12.1521	1.32	0.01	8.79
23	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_2^{-1.5} + a_2 \cdot x_3^{-0.5} + a_3 \cdot x_4^{1.0} + a_4 \cdot (x_1^{2.0} \cdot x_2^{-1.5}) + a_5 \cdot (x_1^{2.0} \cdot x_3^{-0.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_4^{1.0}) + b$	0.9297	0.9299	1.4985	0.0032	12.5632	1.37	0.00	8.50
24	$a_0 \cdot log(x_1+1)^{1.5} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{-0.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{1.5} \cdot x_3^{-0.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_4^{1.5}) + b$	0.9296	0.9308	1.3667	0.0044	12.0961	1.23	0.00	8.18
25	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{-0.5} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{2.0} \cdot x_4^{-1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_4^{-1.0}) + b$	0.9295	0.9305	1.4030	0.0059	12.1554	1.28	0.01	10.07
26	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{1.0} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{2.0} \cdot x_4^{-1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_4^{-1.0}) + b$	0.9292	0.9305	1.4468	0.0198	12.0049	1.32	0.02	8.86
27	$a_0 \cdot log(x_3+1)^{2.0} + a_1 \cdot x_1^{-1.0} + a_2 \cdot x_2^{1.5} + a_3 \cdot x_4^{1.5} + a_4 \cdot (x_3^{2.0} \cdot x_2^{1.5}) + a_5 \cdot (x_1^{2.0} \cdot x_2^{1.5} \cdot x_4^{1.5}) + b$	0.9279	0.9288	1.5360	0.0250	13.8230	1.42	0.02	11.02
28	$a_0 \cdot log(x_2+1)^{-1.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_3^{1.5} + a_3 \cdot x_4^{1.0} + a_4 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{1.5}) + a_5 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_4^{1.5}) + b$	0.9275	0.9275	1.3765	0.0344	16.3304	1.25	0.03	10.44
29	$a_0 \cdot log(x_1+1)^{1.5} + a_1 \cdot x_2^{1.5} + a_2 \cdot x_3^{-1.0} + a_3 \cdot x_4^{1.0} + a_4 \cdot (x_1^{1.5} \cdot x_2^{-0.5}) + a_5 \cdot (x_1^{1.5} \cdot x_2^{-0.5} \cdot x_3^{-1.0}) + a_6 \cdot (x_2^{-0.5} \cdot x_3^{-1.0} \cdot x_4^{1.0}) + b$	0.9260	0.9270	1.4092	0.0138	16.0309	1.29	0.01	10.25
30	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_3^{-1.0} + a_3 \cdot x_1^{1.5} + a_4 \cdot (x_3^{-1.0} \cdot x_4^{1.5}) + b$	0.9250	0.9250	1.6657	0.0144	10.0719	1.53	0.01	7.90
31	$a_0 \cdot x_1^{1.0} + a_1 \cdot x_2^{1.5} + a_2 \cdot x_3^{1.5} + a_3 \cdot x_4^{1.5} + a_4 \cdot (x_1^{1.0} \cdot x_2^{-1.5} \cdot x_3^{1.5}) + a_5 \cdot (x_1^{1.0} \cdot x_2^{-1.5} \cdot x_4^{1.5}) + b$	0.9237	0.9246	1.3945	0.0002	14.0643	1.27	0.00	9.87
32	$a_0 \cdot log(x_1+1)^{1.5} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-1.5} + a_3 \cdot x_3^{-1.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{1.5} \cdot x_3^{-1.5} \cdot x_4^{1.5}) + a_6 \cdot (x_3^{-1.5} \cdot x_4^{1.5}) + b$	0.9234	0.9236	1.5268	0.0051	11.8832	1.40	0.00	9.23
33	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{2.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{1.0} + a_5 \cdot (x_1^{2.0} \cdot x_4^{1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{2.0} \cdot x_4^{1.0}) + b$	0.9233	0.9248	1.4583	0.0070	12.0784	1.33	0.01	9.13
34	$a_0 \cdot log(x_1+1)^{1.5} + a_1 \cdot x_2^{1.5} + a_2 \cdot x_3^{1.0} + a_3 \cdot x_4^{0.5} + a_4 \cdot (x_1^{1.5} \cdot x_2^{1.5}) + a_5 \cdot (x_2^{1.5} \cdot x_3^{1.0} \cdot x_4^{0.5}) + a_6 \cdot (x_1^{1.5} \cdot x_3^{1.0} \cdot x_4^{0.5}) + b$	0.9230	0.9249	1.5340	0.0069	14.8470	1.39	0.01	9.49
35	$a_0 \cdot x_1^{1.0} + a_1 \cdot x_2^{1.5} + a_2 \cdot x_3^{1.5} + a_3 \cdot x_4^{1.5} + a_4 \cdot (x_1^{1.0} \cdot x_2^{-1.5} \cdot x_4^{1.5}) + a_5 \cdot (x_1^{1.0} \cdot x_2^{-1.5} \cdot x_3^{1.5}) + a_6 \cdot (x_1^{1.0} \cdot x_2^{-1.5} \cdot x_3^{1.5} \cdot x_4^{1.5}) + b$	0.9223	0.9230	1.4211	0.0188	14.4905	1.28	0.02	9.26
36	$a_0 \cdot log(x_2+1)^{-1.5} + a_1 \cdot x_2^{1.0} + a_2 \cdot x_1^{1.5} + a_3 \cdot x_3^{1.0} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_2^{-1.5} \cdot x_3^{1.0}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_4^{1.5}) + b$	0.9222	0.9223	1.4917	0.0004	15.9418	1.37	0.00	10.19
37	$a_0 \cdot log(x_4+1)^{1.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_1^{1.5} + a_3 \cdot x_2^{-1.0} + a_4 \cdot x_3^{2.0} + a_5 \cdot (x_2^{-1.0} \cdot x_3^{2.0}) + a_6 \cdot (x_4^{1.0} \cdot x_1^{1.5} \cdot x_2^{-1.0}) + b$	0.9213	0.9214	1.4679	0.0128	13.5659	1.34	0.01	10.06
38	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_2^{1.0} + a_2 \cdot x_3^{1.5} + a_3 \cdot x_4^{1.5} + a_4 \cdot (x_1^{2.0} \cdot x_2^{1.0}) + a_5 \cdot (x_3^{1.5} \cdot x_4^{1.5}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{1.0} \cdot x_3^{1.5} \cdot x_4^{1.5}) + b$	0.9211	0.9211	1.6364	0.0275	13.2129	1.49	0.03	8.94
39	$a_0 \cdot x_1^{1.0} + a_1 \cdot x_2^{-2.0} + a_2 \cdot x_3^{1.5} + a_3 \cdot x_4^{2.0} + a_4 \cdot (x_1^{1.0} \cdot x_3^{1.5}) + a_5 \cdot (x_1^{1.0} \cdot x_2^{-2.0} \cdot x_4^{2.0}) + b$	0.9208	0.9208	1.5073	0.0133	12.2190	1.36	0.01	9.06
40	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{-1.0} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{2.0} + a_5 \cdot (x_2^{1.0} \cdot x_4^{2.0}) + a_6 \cdot (x_2^{1.0} \cdot x_3^{0.5}) + b$	0.9191	0.9194	1.5292	0.0136	13.9788	1.38	0.01	8.94
41	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{-1.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{2.0} + a_5 \cdot (x_2^{1.0} \cdot x_4^{2.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{1.0} \cdot x_3^{0.5}) + b$	0.9191	0.9191	1.5768	0.0108	12.7320	1.44	0.01	9.44
42	$a_0 \cdot log(x_4+1)^{0.5} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_2^{-0.5} + a_3 \cdot x_3^{2.0} + a_4 \cdot (x_2^{0.5} \cdot x_3^{2.0}) + a_5 \cdot (x_2^{0.5} \cdot x_4^{1.5}) + a_6 \cdot (x_3^{0.5} \cdot x_4^{1.5}) + b$	0.9190	0.9192	1.4904	0.0123	11.7873	1.34	0.01	9.77
43	$a_0 \cdot log(x_4+1)^{1.5} + a_1 \cdot x_2^{-1.5} + a_2 \cdot x_3^{-0.5} + a_3 \cdot x_4^{1.0} + a_4 \cdot (x_1^{1.5} \cdot x_2^{-1.5}) + a_5 \cdot (x_1^{1.5} \cdot x_3^{-0.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_4^{1.0}) + b$	0.9188	0.9196	1.3671	0.0047	18.1610	1.22	0.00	12.28
44	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_2^{-2.0} + a_2 \cdot x_3^{-0.5} + a_3 \cdot x_4^{1.5} + a_4 \cdot x_1^{1.5} + a_5 \cdot (x_3^{0.5} \cdot x_4^{1.5}) + b$	0.9184	0.9191	1.6370	0.0187	12.7874	1.51	0.02	9.10
45	$a_0 \cdot log(x_4+1)^{0.5} + a_1 \cdot x_4^{-1.5} + a_2 \cdot x_1^{1.5} + a_3 \cdot x_2^{1.5} + a_4 \cdot x_3^{2.0} + a_5 \cdot (x_2^{0.5} \cdot x_3^{2.0}) + a_6 \cdot (x_4^{0.5} \cdot x_1^{-1.5} \cdot x_2^{-0.5}) + b$	0.9175	0.9177	1.5472	0.0067	10.7255	1.40	0.01	8.89
46	$a_0 \cdot log(x_1+1)^{1.5} + a_1 \cdot x_1^{-1.5} + a_2 \cdot x_2^{-1.5} + a_3 \cdot x_3^{-0.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{1.5} \cdot x_2^{-1.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_4^{1.5}) + b$	0.9167	0.9176	1.4530	0.0512	14.5509	1.31	0.05	10.29
47	$a_0 \cdot log(x_4+1)^{0.5} + a_1 \cdot x_1^{-2.0} + a_2 \cdot x_2^{1.5} + a_3 \cdot x_3^{2.0} + a_4 \cdot x_2^{-0.5} + a_5 \cdot (x_2^{-0.5} \cdot x_3^{2.0}) + a_6 \cdot (x_4^{0.5} \cdot x_1^{1.5} \cdot x_2^{-0.5}) + b$	0.9161	0.9163	1.5428	0.0120	10.9767	1.39	0.01	9.10
48	$a_0 \cdot log(x_4+1)^{0.5} + a_1 \cdot x_1^{-2.0} + a_2 \cdot x_2^{1.5} + a_3 \cdot x_4^{-0.5} + a_4 \cdot x_3^{2.0} + a_5 \cdot (x_2^{-0.5} \cdot x_3^{2.0}) + a_6 \cdot (x_4^{0.5} \cdot x_1^{1.5} \cdot x_2^{-0.5}) + b$	0.9158	0.9161	1.5609	0.0050	10.6318	1.42	0.01	8.81
49	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{1.5} + a_4 \cdot x_4^{2.0} + a_5 \cdot (x_3^{1.5} \cdot x_4^{2.0}) + b$	0.9157	0.9162	1.5937	0.0094	15.1430	1.48	0.01	13.97
50	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_2^{1.0} \cdot x_4^{1.5}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{1.0} \cdot x_3^{0.5}) + b$	0.9143	0.9150	1.3471	0.0054	16.4167	1.21	0.01	10.50

Πίνακας 5.10: Αποτελέσματα του μηχανήματος Broady για τις παραμέτρους MPKI IPC TLB BW.

N	Top	Function	R^2	Var	Real			Relative		
					MAE	Min	Max	MAE	Min	Max
3	1	$a_0 \cdot log(x_1 + 1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{1.0} \cdot x_3^{0.5}) + b$	0.9332	0.9332	5.8622	0.0523	56.5635	4.22	0.05	30.26
	2	$a_0 \cdot log(x_1 + 1)^{0.5} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{2.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + b$	0.9298	0.9299	5.9157	0.0090	55.9132	4.27	0.01	39.33
	3	$a_0 \cdot log(x_1 + 1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{-0.5} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{-0.5}) + a_5 \cdot (x_2^{1.5} \cdot x_3^{1.0}) + b$	0.9284	0.9288	6.2072	0.0159	55.9471	4.45	0.02	33.81
	4	$a_0 \cdot log(x_1 + 1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{1.5} + a_3 \cdot x_3^{1.0} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.5}) + a_5 \cdot (x_2^{1.5} \cdot x_3^{1.0}) + b$	0.9263	0.9270	6.0683	0.0513	55.5962	4.38	0.05	29.45
	5	$a_0 \cdot log(x_2 + 1)^{-1.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_3^{0.5} + a_3 \cdot (x_2^{-1.5} \cdot x_1^{1.0}) + a_4 \cdot (x_2^{-1.5} \cdot x_3^{-0.5}) + a_5 \cdot (x_1^{1.0} \cdot x_3^{-0.5}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5}) + b$	0.9257	0.9259	7.3898	0.0019	62.0470	5.47	0.00	31.41
	6	$a_0 \cdot log(x_3 + 1)^{-0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_2^{-1.5} + a_4 \cdot (x_3^{-0.5} \cdot x_1^{1.0}) + a_5 \cdot (x_1^{1.0} \cdot x_2^{-1.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_1^{1.0} \cdot x_2^{-1.5}) + b$	0.9256	0.9258	7.1433	0.0420	62.3022	5.23	0.04	31.90
	7	$a_0 \cdot log(x_3 + 1)^{-0.5} + a_1 \cdot x_3^{-1.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_2^{-1.5} + a_4 \cdot (x_3^{-0.5} \cdot x_1^{1.0}) + a_5 \cdot (x_1^{1.0} \cdot x_2^{-1.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_1^{1.0} \cdot x_2^{-1.5}) + b$	0.9255	0.9257	7.1845	0.0168	62.2816	5.27	0.02	31.82
	8	$a_0 \cdot log(x_3 + 1)^{-0.5} + a_1 \cdot x_3^{2.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_2^{-1.5} + a_4 \cdot (x_3^{-0.5} \cdot x_1^{1.0}) + a_5 \cdot (x_1^{1.0} \cdot x_2^{-1.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_1^{1.0} \cdot x_2^{-1.5}) + b$	0.9254	0.9256	7.2222	0.0405	62.2752	5.30	0.04	31.77
	9	$a_0 \cdot log(x_2 + 1)^{-1.5} + a_1 \cdot x_2^{2.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{-0.5} + a_4 \cdot (x_2^{-1.5} \cdot x_1^{1.0}) + a_5 \cdot (x_1^{1.0} \cdot x_3^{-0.5}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5}) + b$	0.9254	0.9254	7.3753	0.0213	62.4144	5.50	0.02	32.14
	10	$a_0 \cdot log(x_1 + 1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{2.0} + a_3 \cdot x_3^{1.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{2.0}) + b$	0.9254	0.9256	5.9564	0.0010	54.1081	4.24	0.00	29.90
	11	$a_0 \cdot x_1^{1.0} + a_1 \cdot x_2^{-1.5} + a_2 \cdot x_3^{-0.5} + a_3 \cdot (x_1^{1.0} \cdot x_2^{-1.5}) + a_4 \cdot (x_1^{1.0} \cdot x_3^{-0.5}) + a_5 \cdot (x_2^{-1.5} \cdot x_3^{-0.5}) + a_6 \cdot (x_1^{1.0} \cdot x_2^{-1.5} \cdot x_3^{-0.5}) + b$	0.9254	0.9255	7.4281	0.0466	62.0777	5.50	0.04	31.41
	12	$a_0 \cdot log(x_3 + 1)^{-0.5} + a_1 \cdot x_3^{-0.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_2^{-1.5} + a_4 \cdot (x_3^{-0.5} \cdot x_1^{1.0}) + a_5 \cdot (x_1^{1.0} \cdot x_2^{-1.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_1^{1.0} \cdot x_2^{-1.5}) + b$	0.9252	0.9254	7.0569	0.0666	62.8293	5.14	0.06	32.06
	13	$a_0 \cdot log(x_3 + 1)^{-0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{-1.5} + a_3 \cdot (x_3^{-0.5} \cdot x_1^{1.0}) + a_4 \cdot (x_3^{-0.5} \cdot x_2^{-1.5}) + a_5 \cdot (x_1^{1.0} \cdot x_2^{-1.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_1^{1.0} \cdot x_2^{-1.5}) + b$	0.9252	0.9253	7.4282	0.0329	62.0588	5.50	0.03	31.38
	14	$a_0 \cdot log(x_1 + 1)^{1.0} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{0.5} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{1.0} \cdot x_2^{0.5}) + a_5 \cdot (x_1^{1.0} \cdot x_3^{0.5}) + a_6 \cdot (x_2^{0.5} \cdot x_3^{0.5}) + b$	0.9252	0.9253	6.4279	0.0003	57.1538	4.65	0.00	34.68
	15	$a_0 \cdot log(x_2 + 1)^{-1.5} + a_1 \cdot x_2^{1.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{-0.5} + a_4 \cdot (x_2^{-1.5} \cdot x_1^{1.0}) + a_5 \cdot (x_1^{1.0} \cdot x_3^{-0.5}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5}) + b$	0.9251	0.9252	7.3522	0.0562	62.4104	5.49	0.05	32.17
	16	$a_0 \cdot log(x_1 + 1)^{0.5} + a_1 \cdot x_1^{0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{1.0}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + a_6 \cdot (x_2^{1.0} \cdot x_3^{0.5}) + b$	0.9251	0.9254	6.1895	0.0229	54.5043	4.49	0.02	38.34
	17	$a_0 \cdot log(x_2 + 1)^{-1.5} + a_1 \cdot x_2^{1.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{-0.5} + a_4 \cdot (x_2^{-1.5} \cdot x_1^{1.0}) + a_5 \cdot (x_1^{1.0} \cdot x_3^{-0.5}) + a_6 \cdot (x_2^{-1.5} \cdot x_1^{1.0} \cdot x_3^{-0.5}) + b$	0.9247	0.9247	7.3568	0.0335	62.4095	5.49	0.03	32.17
	18	$a_0 \cdot log(x_3 + 1)^{-0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{1.5} + a_3 \cdot (x_3^{-0.5} \cdot x_1^{1.0}) + a_4 \cdot (x_1^{1.0} \cdot x_2^{-1.5}) + a_5 \cdot (x_3^{-0.5} \cdot x_1^{1.0} \cdot x_2^{-1.5}) + b$	0.9243	0.9245	7.4559	0.0246	62.5162	5.51	0.02	31.38
	19	$a_0 \cdot x_1^{1.0} + a_1 \cdot x_2^{-1.5} + a_2 \cdot x_3^{-0.5} + a_3 \cdot (x_1^{1.0} \cdot x_2^{-1.5}) + a_4 \cdot (x_1^{1.0} \cdot x_3^{-0.5}) + a_5 \cdot (x_2^{-1.5} \cdot x_3^{-0.5}) + a_6 \cdot (x_1^{1.0} \cdot x_2^{-1.5} \cdot x_3^{-0.5}) + b$	0.9243	0.9245	7.4516	0.0507	62.5187	5.51	0.05	31.40
	20	$a_0 \cdot log(x_1 + 1)^{0.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{0.5} + a_3 \cdot x_3^{0.5} + a_4 \cdot (x_1^{0.5} \cdot x_2^{0.5}) + a_5 \cdot (x_1^{0.5} \cdot x_3^{0.5}) + a_6 \cdot (x_1^{0.5} \cdot x_2^{0.5} \cdot x_3^{0.5}) + b$	0.9241	0.9241	6.0750	0.0164	60.2543	4.28	0.01	28.14
2	1	$a_0 \cdot log(x_3 + 1)^{1.0} + a_1 \cdot x_3^{0.5} + a_2 \cdot x_1^{0.5} + b$	0.9052	0.9052	7.1966	0.0337	68.2382	5.09	0.03	33.45
	2	$a_0 \cdot log(x_3 + 1)^{1.0} + a_1 \cdot x_3^{1.5} + a_2 \cdot x_1^{0.5} + a_3 \cdot (x_3^{1.0} \cdot x_1^{0.5}) + b$	0.9023	0.9023	7.6113	0.0158	70.2170	5.58	0.01	35.63
	3	$a_0 \cdot log(x_3 + 1)^{1.5} + a_1 \cdot x_3^{1.0} + a_2 \cdot x_1^{0.5} + b$	0.9022	0.9022	7.6202	0.0232	69.8737	5.58	0.02	34.78
	4	$a_0 \cdot log(x_3 + 1)^{1.5} + a_1 \cdot x_3^{1.5} + a_2 \cdot x_1^{0.5} + b$	0.8946	0.8948	7.7416	0.0154	70.1932	5.67	0.02	35.22
	5	$a_0 \cdot log(x_3 + 1)^{1.5} + a_1 \cdot x_3^{1.5} + a_2 \cdot x_1^{0.5} + a_3 \cdot (x_1^{1.5} \cdot x_3^{0.5}) + b$	0.8849	0.8851	8.2147	0.0385	70.2793	6.04	0.03	36.33

Πίνακας 5.11: Αποτελέσματα του μηχανήματος Sandman για 3 και 2 παραμέτρους.

Top	Function	R^2	Var	Real			Relative		
				MAE	Min	Max	MAE	Min	Max
1	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_2^{-2.0} + a_2 \cdot x_3^{-0.5} + a_3 \cdot x_4^{1.0} + a_4 \cdot (x_1^{2.0} \cdot x_2^{-2.0}) + a_5 \cdot (x_1^{2.0} \cdot x_3^{-0.5}) + a_6 \cdot (x_3^{-0.5} \cdot x_4^{1.0}) + b$	0.9569	0.9569	1.3283	0.0136	7.4566	1.24	0.01	5.74
2	$a_0 \cdot log(x_4+1)^{1.0} + a_1 \cdot x_4^{0.5} + a_2 \cdot x_1^{1.5} + a_3 \cdot x_2^{-1.0} + a_4 \cdot x_3^{2.0} + a_5 \cdot (x_2^{-1.0} \cdot x_3^{2.0}) + a_6 \cdot (x_4^{1.0} \cdot x_1^{1.5} \cdot x_2^{-1.0}) + b$	0.9521	0.9521	1.3272	0.0006	7.6840	1.23	0.00	6.17
3	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_2^{-0.5} + a_2 \cdot x_3^{-1.0} + a_3 \cdot x_4^{1.5} + a_4 \cdot (x_1^{2.0} \cdot x_2^{-0.5}) + a_5 \cdot (x_1^{2.0} \cdot x_3^{-1.0} \cdot x_4^{1.5}) + a_6 \cdot (x_2^{-0.5} \cdot x_3^{-1.0} \cdot x_4^{1.5}) + b$	0.9486	0.9487	1.3720	0.0056	10.1817	1.27	0.01	6.51
4	$a_0 \cdot log(x_4+1)^{1.0} + a_1 \cdot x_4^{-1.0} + a_2 \cdot x_1^{1.5} + a_3 \cdot x_2^{-1.0} + a_4 \cdot x_3^{2.0} + a_5 \cdot (x_2^{-1.0} \cdot x_3^{2.0}) + a_6 \cdot (x_4^{1.0} \cdot x_1^{1.5} \cdot x_2^{-1.0}) + b$	0.9469	0.9469	1.3150	0.0030	9.1410	1.21	0.00	7.35
5	$a_0 \cdot x_1^{1.5} + a_1 \cdot x_2^{-0.5} + a_2 \cdot x_3^{2.0} + a_3 \cdot x_4^{0.5} + a_4 \cdot (x_2^{-0.5} \cdot x_3^{2.0}) + a_5 \cdot (x_2^{-0.5} \cdot x_4^{0.5}) + a_6 \cdot (x_1^{1.5} \cdot x_2^{-0.5} \cdot x_4^{0.5}) + b$	0.9444	0.9445	1.3500	0.0001	9.8247	1.24	0.00	6.28
6	$a_0 \cdot log(x_2+1)^{-0.5} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_3^{2.0} + a_3 \cdot x_4^{0.5} + a_4 \cdot (x_2^{-0.5} \cdot x_3^{2.0}) + a_5 \cdot (x_2^{-0.5} \cdot x_4^{0.5}) + a_6 \cdot (x_2^{-0.5} \cdot x_1^{1.5} \cdot x_4^{0.5}) + b$	0.9443	0.9445	1.3495	0.0003	9.8980	1.24	0.00	6.33
7	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{-1.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{2.0} \cdot x_3^{-1.5}) + a_6 \cdot (x_3^{-1.5} \cdot x_4^{1.5}) + b$	0.9389	0.9401	1.5404	0.0013	8.4853	1.44	0.00	6.29
8	$a_0 \cdot log(x_2+1)^{-1.5} + a_1 \cdot x_2^{1.0} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_3^{1.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{1.0} \cdot x_3^{1.5}) + a_6 \cdot (x_2^{1.5} \cdot x_1^{1.0} \cdot x_4^{1.5}) + b$	0.9384	0.9386	1.3824	0.0118	14.4185	1.26	0.01	9.22
9	$a_0 \cdot log(x_3+1)^{2.0} + a_1 \cdot x_3^{1.5} + a_2 \cdot x_1^{1.5} + a_3 \cdot x_2^{-0.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_3^{2.0} \cdot x_2^{-0.5}) + a_6 \cdot (x_1^{1.5} \cdot x_2^{-0.5} \cdot x_4^{0.5}) + b$	0.9379	0.9385	1.4104	0.0174	10.0110	1.29	0.02	7.42
10	$a_0 \cdot log(x_1+1)^{1.5} + a_1 \cdot x_2^{-0.5} + a_2 \cdot x_3^{-1.0} + a_3 \cdot x_4^{1.0} + a_4 \cdot (x_1^{1.5} \cdot x_2^{-0.5}) + a_5 \cdot (x_3^{-1.0} \cdot x_4^{1.0}) + a_6 \cdot (x_1^{1.5} \cdot x_2^{-0.5} \cdot x_3^{-1.0}) + b$	0.9347	0.9353	1.3258	0.0044	16.2228	1.22	0.00	10.37
11	$a_0 \cdot log(x_4+1)^{1.0} + a_1 \cdot x_1^{1.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{2.0} + a_4 \cdot (x_2^{-1.0} \cdot x_3^{2.0}) + a_5 \cdot (x_4^{1.0} \cdot x_1^{1.5} \cdot x_2^{-1.0}) + b$	0.9339	0.9339	1.3937	0.0260	12.5825	1.28	0.03	9.33
12	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{2.0} + a_3 \cdot x_3^{-2.0} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{2.0} \cdot x_4^{-1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_4^{-1.0}) + b$	0.9332	0.9342	1.3849	0.0249	12.2240	1.27	0.02	10.13
13	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_2^{0.5} + a_2 \cdot x_3^{-0.5} + a_3 \cdot x_4^{1.5} + a_4 \cdot (x_1^{2.0} \cdot x_2^{0.5}) + a_5 \cdot (x_1^{2.0} \cdot x_4^{1.5}) + a_6 \cdot (x_2^{0.5} \cdot x_3^{-0.5} \cdot x_4^{1.5}) + b$	0.9318	0.9320	1.4766	0.0269	14.0979	1.37	0.03	11.38
14	$a_0 \cdot log(x_3+1)^{1.5} + a_1 \cdot x_1^{1.0} + a_2 \cdot x_2^{-1.5} + a_3 \cdot x_4^{1.5} + a_4 \cdot (x_3^{1.5} \cdot x_1^{1.0} \cdot x_2^{-1.5}) + a_5 \cdot (x_1^{1.0} \cdot x_2^{-1.5} \cdot x_4^{1.5}) + b$	0.9317	0.9317	1.4621	0.0184	14.4642	1.34	0.02	9.25
15	$a_0 \cdot log(x_3+1)^{1.5} + a_1 \cdot x_3^{-0.5} + a_2 \cdot x_1^{1.0} + a_3 \cdot x_2^{-1.5} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_3^{1.5} \cdot x_1^{1.0}) + a_6 \cdot (x_1^{1.0} \cdot x_2^{-1.5} \cdot x_4^{1.5}) + b$	0.9315	0.9315	1.3692	0.0054	15.3848	1.24	0.01	9.84
16	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{2.0} + a_3 \cdot x_3^{2.0} + a_4 \cdot x_4^{1.0} + a_5 \cdot (x_1^{2.0} \cdot x_4^{1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{2.0} \cdot x_4^{1.0}) + b$	0.9315	0.9328	1.4365	0.0004	11.8455	1.31	0.00	8.62
17	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{-0.5} + a_2 \cdot x_2^{1.0} + a_3 \cdot x_3^{0.5} + a_4 \cdot x_4^{2.0} + a_5 \cdot (x_2^{1.0} \cdot x_4^{2.0}) + a_6 \cdot (x_2^{2.0} \cdot x_2^{1.0} \cdot x_3^{0.5}) + b$	0.9313	0.9313	1.4101	0.0334	9.4848	1.30	0.03	8.38
18	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{-1.0} + a_4 \cdot x_4^{1.5} + a_5 \cdot (x_1^{2.0} \cdot x_3^{-1.0}) + a_6 \cdot (x_3^{-1.0} \cdot x_4^{1.5}) + b$	0.9311	0.9312	1.5126	0.0180	12.0506	1.40	0.02	8.94
19	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{-1.5} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{2.0} \cdot x_4^{-1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_4^{-1.0}) + b$	0.9303	0.9315	1.4030	0.0084	12.1353	1.28	0.01	10.05
20	$a_0 \cdot log(x_1+1)^{2.0} + a_1 \cdot x_1^{2.0} + a_2 \cdot x_2^{-2.0} + a_3 \cdot x_3^{-1.0} + a_4 \cdot x_4^{-1.0} + a_5 \cdot (x_1^{2.0} \cdot x_4^{-1.0}) + a_6 \cdot (x_1^{2.0} \cdot x_2^{-2.0} \cdot x_4^{-1.0}) + b$	0.9299	0.9310	1.3999	0.0299	12.2669	1.28	0.03	10.16

Πίνακας 5.12: Αποτελέσματα του μηχανήματος Broady για 3 τις παραμέτρους.

(a) Sandman (b) Broady

Σχήμα 5.5: Γραφική αναπαράσταση των καλύτερων συναρτήσεων για τις προβλέψεις που έκαναν.

Κεφάλαιο 6

Μεταφορά Μνήμης

Σε αυτό το κεφάλαιο θα δούμε κάποιες μεθόδους που έχουμε στην διάθεση μας για να μεταφέρουμε την μνήμη μιας εφαρμογής κατά την διάρκεια εκτέλεσης της. Θα δούμε τα βασικά χαρακτηριστικά τους καθώς και πως μπορούμε να εκμεταλλευτούμε τις λειτουργίες τους. Τέλος θα δούμε κάποια από τα αποτελέσματα που πήραμε με την χρήση αυτών των μεθόδων.

6.1 Χαρακτηριστικά των μεθόδων μεταφοράς

Πριν πάμε να δούμε τις συναρτήσεις που χρησιμοποιήσαμε για να κάνουμε την μεταφορά της μνήμης χρειάζεται να δούμε κάποια βασικά χαρακτηριστικά αυτών των συναρτήσεων. Τα χαρακτηριστικά που θα δούμε εδώ δεν είναι εξολοκλήρου κοινά και για τις συναρτήσεις, όμως είναι αρκετά παρόμοια.

Οι δύο συναρτήσεις που χρησιμοποιήσαμε για να πετύχουμε την μεταφορά της μνήμης είναι, η migratepages[3] και η movepages[4]. Αυτές οι δύο συναρτήσεις προέρχονται από την βιβλιοθήκη numaif.h της γλώσσας προγραμματισμού C. Πέρα από την ενσωμάτωση της βιβλιοθήκης, για να μπορέσουμε να χρησιμοποιήσουμε οποιαδήποτε από τις δύο συναρτήσεις χρειάζεται να συνδέσουμε με το -lnuma κατά την διάρκεια της μεταγλώττισης.

Ανάλογα με το πείραμα που θέλουμε να εκτελέσουμε χρησιμοποιούμε και την αντίστοιχη συνάρτηση. Υπεύθυνος για την σωστή χρήση των συναρτήσεων καθώς και για την αρχικοποίηση των απαραίτητων παραμέτρων που χρειάζεται η κάθε συνάρτηση, είναι η δομική μονάδα του μεταφορέα στον επόπτη. Οι δύο συναρτήσεις χρησιμοποιούν διαφορετικές παραμέτρους για να κάνουν τις λειτουργίες τούς. Αυτό το κομμάτι θα το δούμε σε κάθε συνάρτηση ξεχωριστά. Όμως γενικά αυτά που χρειαζόμαστε να ξέρουμε για να τις χρησιμοποιήσουμε είναι: το pid της διεργασίας που θέλουμε να της μεταφέρουμε την μνήμη και την κατάσταση που έχει η μνήμη της την δεδομένη χρονική στιγμή.

Είναι γνωστό ότι η μνήμη μιας διεργασίας, είναι αποθηκευμένη σε διάσπαρτες σελίδες στο σύστημα μας. Η λειτουργία που κάνουν οι συναρτήσεις είναι να μεταφέρουν τις σελίδες από έναν κόμβο του συστήματος σε έναν άλλο. Δηλαδή να μεταφέρουν την μνήμη που υπάρχει ένα κόμβο και να την πάνε σε κάποιον άλλο. Εδώ μπορούμε να έχουμε πολλούς κόμβους αναχώρησης και άφιξης, όχι αναγκαστικά ένα κόμβο. Στην έρευνά μας μεταφέραμε όλοι την μνήμη που είχαν οι εφαρμογές από όλους τους κόμβους σε ένα συγκεκριμένο. Για να το κάνουμε αυτό, πάλι κάναμε χρήση των τοπολογιών, κεφάλαιο 4.2.3.

Και οι δύο συναρτήσεις λειτουργούν με τον ίδιο τρόπο για να μεταφέρουν την μνήμη. Αρχικά μαρκάρουν όλες τις σελίδες που τους έχουμε ζητήσει. Δεν είναι πάντα σίγουρο ότι θα μαρκάρουν όλες τις σελίδες που τους έχουμε ζητήσει είτε γιατί δεν έχουμε κάποιο

δικαίωμα σε κάποιες από αυτές τις σελίδες είτε γιατί πλέον δεν υπάρχουν στην μνήμη. Στην συνέχεια, όταν μπορέσουν, κλειδώνουν αυτές τς σελίδες και δεν επιτρέπουν στην εφαρμογή να γράψει σε αυτές τις σελίδες. Έπειτα αρχίζουν να μεταφέρουν τις σελίδες που έχουν κλειδώσει στον επιθυμητό κόμβο προορισμού. Εδώ πάλι δεν είναι σίγουρο ότι θα μεταφερθούν όλες οι κλειδωμένες σελίδες για τους ίδιους λόγους που είπαμε πριν. Αφού ολοκληρωθεί η μεταφορά, ξεκλειδώνουν τις σελίδες και δίνουν πάλι πρόσβασης στην εφαρμογή, στις σελίδες που μεταφέρθηκαν.

Εδώ χρειάζεται να κάνουμε μια σημαντική παρατήρηση σε σχέση με όσα πράγματα έχουμε δει μέχρι στιγμής στην έρευνα μας. Οι συναρτήσεις migratepages και movepages δεν ενδιαφέρονται για τις προηγούμενες καταστάσεις που βρισκόταν η εφαρμογή μας ούτε και για τις επόμενες. Έστω ότι έχουμε το εξής σενάριο. Έχουμε το benchmark A που το έχουμε ορίσει να παίρνει μνήμη από ένα συγκεκριμένο κόμβο του συστήματος. Θέλουμε να μεταφέρουμε την μνήμη του A σε κάποιον άλλο κόμβο από αυτόν που του έχουμε ορίσει. Έστω ότι έχουμε όλα τα απαραίτητα δικαιώματα για να χρησιμοποιήσουμε οποιαδήποτε από τις δύο συναρτήσεις, έτσι ώστε να μεταφέρουμε την μνήμη. Πριν την μεταφορά αν δεν υπάρχει διαθέσιμη μνήμη για να μεταφέρουμε, οι συναρτήσεις ολοκληρώνουν την λειτουργία τους. Αν είχαμε τότε κάνουμε την μεταφορά και τερματίζουν. Αν η εφαρμογή στην συνέχεια δεσμεύσει νέα μνήμη, σε οποιαδήποτε από τις δύο περιπτώσεις, αυτή θα πάει στο προκαθορισμένο κόμβο και όχι στον κόμβο που είχαμε ζητήσει να γίνει η μεταφορά. Αυτό είναι μια σημαντική διαφορά σε σχέση με το εργαλείο numactl που ενδιαφέρεται για την κατάσταση της μνήμης των εφαρμογών σε όλη την διάρκεια της εκτέλεσης τους.

6.2 Migratepages

Αφού θέσαμε τα βασικά χαρακτηριστικά των μεθόδων μεταφοράς της μνήμης, θα δούμε λίγο πιο αναλυτικά τις λειτουργίες που επιτελεί η κάθε συνάρτηση ξεχωριστά. Ας αρχίσουμε με την συνάρτηση migratepages καθώς έχει πιο απλή λειτουργία. Η συνάρτηση migratepages μεταφέρει όλες τις σελίδες από τους κόμβους αναχώρησης που θα της ζητήσουμε και τις πηγαίνει στους κόμβους προορισμού. Η πλειονότητα των πειραμάτων που εκτελέσαμε αφορούσαν αυτήν την συνάρτηση.

Για να μπορέσουμε να χρησιμοποιήσουμε την migratepages χρειαζόμαστε να έχουμε ελάχιστες πληροφορίες για την διαδικασία της οποίας θα μεταφέρουμε την μνήμη. Η κυριότερη πληροφορία που χρειαζόμαστε είναι το pid της διεργασίας. Αυτό είναι αρκετά εύκολο να το βρούμε μέσο του επόπτη, αφού ο εκκινητής και ο μεταφορέας είναι παιδιά του επόπτη. Οπότε το μόνο που έχουμε να κάνουμε είναι να διασχίσουμε το δέντρο διεργασιών για να βρούμε το σωστό pid. Εδώ γνωρίσουμε πόσο βαθιά χρειάζεται να πάμε στο δέντρο διεργασιών από τον τρόπο που έχουμε κατασκευάσει τον επόπτη.

Το άλλο κομμάτι πληροφορίας που χρειαζόμαστε είναι να γνωρίσουμε σε ποιους κόμβους έχουμε μνήμη που θέλουμε να μεταφέρουμε και σε ποιους κόμβους θέλουμε να την πάμε. Προφανώς ο προορισμός δεν είναι πρόβλημα, γιατί πάντα γνωρίζουμε που θέλουμε να πάμε την μνήμη. Όμως ούτε οι κόμβοι προέλευσης είναι πρόβλημα, γιατί άμα θέλουμε να μεταφέρουμε όλη την μνήμη της εφαρμογής σε ένα κόμβο, τότε μπορούμε να ζητήσουμε να μεταφερθεί η μνήμη από όλους τους άλλους κόμβους σε αυτόν. Όλες αυτές οι πληροφορίες είναι γνωστές από τις τοπολογίες. Οπότε πρακτικά γνωρίζουμε τα πάντα εκ των προτέρων.

6.3 Movepages

Εδώ θα δούμε πως λειτουργεί η συνάρτηση movepages. Η movepages είναι κάπως πιο πολύπλοκη σε σχέση με την migratepages. Όμως έχει μεγαλύτερο εύρος χρήσεων. Η κύρια διαφορά των δύο συναρτήσεων είναι ότι, η movepages ενδιαφέρεται για συγκεκριμένες σελίδες και την θέση τους σε συγκεκριμένους κόμβους ενώ η migratepages είναι πιο ευλύγιστη σε αυτό το θέμα καθώς χρειάζεται μόνο τους κόμβους.

Τα κέρδη και τα προβλήματα που μας δημιουργεί η movepages προέρχονται από το ίδιο γεγονός, ότι ασχολείται με συγκεκριμένες σελίδες. Ας δούμε πρώτα κάποια από τα κέρδη που μπορεί να έχει μια τέτοια προσέγγιση σε σχέση με πριν:

- Έστω ότι ο τοπικός κόμβος του συστήματος έχει γεμίσει η μνήμη του και χρειάζεται να διώξουμε κάποια. Αντί να διώξουμε όλη την μνήμη μιας εφαρμογής, μπορούμε να χρησιμοποιήσουμε την movepages και να μεταφέρουμε μόνο ένα ποσοστό της.
- Έστω ότι παρατηρούμε μια εφαρμογή κάνει πολλές προσβάσεις σε μια απομακρυσμένη σελίδα. Με την movepages μπορούμε να φέρουμε πιο κοντά αυτήν την σελίδα στην εφαρμογή για της αυξήσουμε την απόδοση, αντί να χρειαστεί να μεταφέρουμε όλη την απομακρυσμένη μνήμη.
- Έστω ότι παρατηρούμε μια εφαρμογή που έχει πολλά κομμάτια της μνήμης της ανενεργά. Δηλαδή δεν γράφει σε αυτά αλλά ούτε τα διαβάζει. Με την movepages μπορούμε να φέρουμε μόνο τα ανενεργά κομμάτια και να αφήσουμε τα χρήσιμα κοντά στην εφαρμογή.

Τέτοια και άλλα παραδείγματα μπορούμε να βρούμε για την χρησιμότητα της movepages σε σχέση με την migratepages.

Συνεχίζοντας, ας δούμε κάποια προβλήματα που υπάρχουν όταν μας ενδιαφέρουν συγκεκριμένες σελίδες μια εφαρμογής. Το πρώτο και σημαντικότερο είναι να βρούμε την διεύθυνση της εικονικής σελίδας που μας ενδιαφέρει. Αυτό φαίνεται αρκετά προφανές, όμως η διαδικασία που χρειάζεται να κάνουμε για να βρούμε αυτήν την σελίδα δεν είναι. Ακόμα όταν μας ενδιαφέρουν συγκεκριμένες σελίδες, χρειάζεται να έχουμε και ένα τρόπο για να μπορούμε να τις αξιολογήσουμε. Αυτό είναι πρόβλημα, γιατί η αξιολόγηση πρέπει να είναι μια διαδικασία αρκετά μη απαιτητική έτσι ώστε να μην επηρεάζονται οι μετρήσεις μας. Αυτό πρακτικά δεν μπορεί να γίνει όταν έχουμε πολλές σελίδες και θέλουμε να αξιολογούμε την κάθε μια ξεχωριστά.

Τα πράγματα δεν καλυτερεύουν όταν θέλουμε να μεταφέρουμε ένα ποσοστό της μνήμης. Έστω ότι θέλουμε να μεταφέρουμε ένα συγκεκριμένο ποσοστό της μνήμης για παράδειγμα το 20% σε κάποιο άλλο κόμβο. Ο τρόπος που θα επιλέξουμε αυτό το ποσοστό από την μνήμη ποικίλει. Μπορούμε να επιλέξουμε να μεταφέρουμε τις πρώτες ή τις τελευταίες σελίδες ή ακόμα και να επιλέξουμε τυχαία ποιες σελίδες θα μεταφέρουμε. Προφανώς αν επιλέξουμε τις πρώτες ή τις τελευταίες αυθαίρετα έχουμε πρόβλημα, γιατί δεν γνωρίζουμε πόσο σημαντικές είναι αυτές οι σελίδες για την εφαρμογή. Όπως φαίνεται καλύτερη επιλογή είναι ο τυχαίος τρόπος, όμως αυτός ο τρόπος είναι πολύ πιο χρονοβόρος σε σχέση με τους άλλος δύο. Αν θέλουμε η επιλογή μας να είναι πραγματικά τυχαία και όχι ψευδο-τυχαία, τότε κάθε φορά που επιλέγουμε μια σελίδα πρέπει να την αφαιρούμε από το σύνολο των σελίδων που είναι προς επιλογή και να την βάζουμε στο σύνολο των επιλεγμένων. Αυτό είναι πολύ κοστοβόρα διαδικασία.

Πολλές από τις πληροφορίες που χρειαζόμαστε μπορούμε να τις πάρουμε από τα numa_maps. Όμως η μορφή με την οποία μας παρέχονται αυτές οι πληροφορίες δεν είναι η καταλληλότερη για να αντιμετωπίσουμε τα προβλήματα που είδαμε. Αυτό συμβαίνει

γιατί δεν μας λέει αναλυτικά όλες τις διευθύνσεις των σελίδων που έχει δεσμεύσει μια εφαρμογή, αλλά χρειάζεται εμείς να κατασκευάσουμε την λίστα με όλες τις διευθύνσεις. Αυτό σε συνδυασμό με τον τρόπο επιλογής των σελίδων μας καταναλώνουν αρκετούς πόρους από το σύστημα και δημιουργούν θόρυβο στης μετρήσεις μας.

Όπως είδαμε, πολλά από τα προβλήματα που είχαμε με την movepages δεν αναφέρονται άμεσα σε αυτήν. Όμως προκύπτουν έμμεσα από αυτήν όταν θέλουμε να την χρησιμοποιήσουμε. Για να πειραματιστούμε με την movepages, κατασκευάσαμε ένα πρόγραμμα που διάβαζε το numa_maps, έκανε όλες τις απαραίτητες επεξεργασίες και μας έδινε τις σελίδες που θα μεταφέραμε. Όμως επειδή η επεξεργασία έπαιρνε αρκετό χρόνο δεν μπορούσαμε να πάρουμε άμεσα τις σελίδες που θέλαμε και σε πολλές περιπτώσεις μετά το τέλος της επεξεργασίας αυτές οι σελίδες δεν υπήρχαν πλέον στο σύστημα. Για αυτό αποφασίσαμε να μην προχωρήσουμε σε περαιτέρω πειραματισμούς με την movepages.

6.4 Υλοποιήσεις

Εδώ θα δούμε τις υλοποιήσεις που κάναμε για να πάρουμε πειραματικά αποτελέσματα από την migratepages. Αυτό που θέλαμε να μελετήσουμε είναι ο χρόνος που χρειάζεται η migratepages για να μεταφέρει όλες τις σελίδες που έχει μια διεργασία στην μνήμη, κάποια συγκεκριμένη χρονική στιγμή. Όλες οι υλοποιήσεις που κάναμε αφορούν την δομική μονάδα του μεταφορέα στον επόπτη.

Για να έχουν νόημα τα πειράματα μας χρειάζεται, η εφαρμογή της οποίας θα μεταφέρουμε την μνήμη να έχει όντως δεσμεύσει κάποιο σεβαστό ποσοστό μνήμης. Αν δεν συνέβαινε αυτό τότε, υπάρχει περίπτωση να μεταφέραμε ελάχιστες σελίδες, οπότε να βγάζαμε εσφαλμένα συμπεράσματα για τον χρόνο που χρειάζεται για να γίνει η μεταφορά. Για να το κάνουμε αυτό, χρειάζεται να ορίσουμε μετά από πόσο χρόνο θα πραγματοποιήσουμε την μεταφορά. Αποφασίσαμε για λόγους ομοιομορφίας, αντί να κάνουμε την μεταφορά μετά από κάποιο προκαθορισμένο χρονικό διάστημα, για παράδειγμα μετά από 20 δευτερόλεπτα, να περιμένουμε ένα ποσοστό του συνολικού χρόνου που χρειάζεται η εφαρμογή για να ολοκληρώσει όλη την τοπική της εκτέλεση. Το ποσοστό του χρόνου που επιλέξαμε είναι το 10%.

Ο τρόπος με τον οποίο βρήκαμε το χρονικό διάστημα που πρέπει να περιμένουμε είναι αρκετά απλός. Αρχικά κάθε πείραμα τρέξαμε μια φορά την τοπική τοπολογία. Μέσω των συναρτήσεων που μας παρέχονται από την βιβλιοθήκη sys/time.h[14] της γλώσσας προγραμματισμού C, μετρήσαμε τον συνολικό χρόνο που χρειάστηκε η εκάστοτε εφαρμογή για να ολοκληρώσει την λειτουργία της. Έπειτα υπολογίζαμε πόσο χρόνο χρειάζεται να περιμένουμε:

$$\text{sleep} = \text{Total Time} \cdot 0.1$$

Σε κάθε επόμενο κύκλο λειτουργίας του επόπτη, βάζαμε τον μεταφορέα να κοιμάται το χρονικό διάστημα που είχαμε υπολογίσει πριν. Μόλις ο μεταφορέας ξύπναγε έκανε όλες τις απαραίτητες ενέργειες που χρειαζόταν για να μεταφέρει την μνήμη και τέλος πραγματοποιούσε την μεταφορά.

Παρόλο που πρακτικά ο χρόνος επεξεργασίας για την χρήση της migratepages είναι αμελητέος, αποφασίσαμε να μην τον συμπεριλάβουμε στης μετρήσεις μας. Έτσι μετρήσαμε μόνο τον χρόνο που χρειαζόμαστε για να πραγματοποιήσουμε την μεταφορά. Ακόμα θέλαμε να γνωρίζουμε πόση μνήμη είχε στην διάθεση της η εφαρμογή πριν κάνουμε την μεταφορά. Αυτήν την πληροφορία μπορούμε να την πάρουμε από τον παρακολουθητή. Όμως για να το κάνουμε αυτό, χρειάζεται οι δύο δομικές μονάδες να έχουν συγχρονι-

σμένα ρολόγια. Τον συγχρονισμό όλων τον ρολογιών τον έχει αναλάβει ο επόπτης. Στο τέλους του πρώτου κύκλου επανάληψης ο επόπτης έχει μετρήσει πόσο χρόνο χρειάζεται η εφαρμογή για να ολοκληρώσει την εκτέλεση της τοπικής τοπολογίας της, οπότε υπολογίζει πόσο χρόνο χρειάζεται να κοιμηθούν οι δομικές μονάδες από τον παραπάνω τύπο. Σε κάθε επόμενο κύκλο επανάληψης ο επόπτης θέτει το αρχικό ρολόι του σε μηδέν. Στην συνέχεια κληρονομούν όλες οι δομικές μονάδες αυτό το ρολόι, οπότε πλέον όλες οι δομικές μονάδες οι συγχρονισμένες σε ένα κοινό σημείο αναφοράς. Εδώ λαμβάνουμε υπόψιν μας της καθυστερήσεις που υπάρχουν από τα θέματα συγχρονισμού των διεργασιών.

6.5 Αποτελέσματα

Εδώ θα δούμε τα αποτελέσματα που πήραμε από πειράματα μας για τον χρόνο που χρειάζεται η migratepages για να μεταφέρει όλη την μνήμη μιας εφαρμογής σε ένα κόμβο κατά την διάρκεια της εκτέλεσης της. Τα πειράματα μας τα εκτελέσαμε εξολοκλήρου στο μηχάνημα Sandman και για της δύο μη τοπικές τοπολογίες του.

Χρησιμοποιήσαμε όλα τα μονονηματικά benchmarks που είχαμε δει στο κεφάλαιο 5.1.1. Ακόμα αποφασίσαμε να μελετήσουμε και την συμπεριφορά που είχαν πολυνηματικά εφαρμογές. Αυτές τις εφαρμογές τις πήραμε από την σουίτα Parsec. Επίσης για να στρεσάρουμε όσο περισσότερο μπορούσαμε την migratepages για τις πολυνηματικές εφαρμογές, εκτελέσαμε πειράματα με περισσότερες από μια διεργασίες ταυτόχρονα στο σύστημα μας. Οι μέθοδοι τοποθέτησης των νημάτων στους επεξεργαστές είναι αυτές που είχαμε δει στον επόπτη.

Τα αποτελέσματα που πήραμε τα βλέπουμε στους πίνακες 6.1 - 6.3. Οι πινάκες 6.1 και 6.2 αφορούν τα μονονηματικά και πολυνηματικά benchmarks αντίστοιχα. Ενώ ο πίνακας 6.3 αφορά την εκτέλεση πολλών πολυνηματικά benchmarks ταυτόχρονα στο σύστημα για της μεθόδους τοποθέτησης 1 και 2.

Το MBM (Memory Before Migrate) μας δίνει την μνήμη που έχει η εκάστοτε εφαρμογή πριν κάνουμε το migration. Το TOM (Time Of Migrate) μας δίνει τον συνολικό χρόνο που χρειάστηκε το migratepages για να ολοκληρώσει την μεταφορά της μνήμης, μετρημένο σε δευτερόλεπτα. Το Per (Percentage) μας δίνει τι ποσοστό που χρειάστηκε η migratepages για να ολοκληρωθεί ως προς τον συνολικό χρόνο της εφαρμογής για την τοπική εκτέλεση:

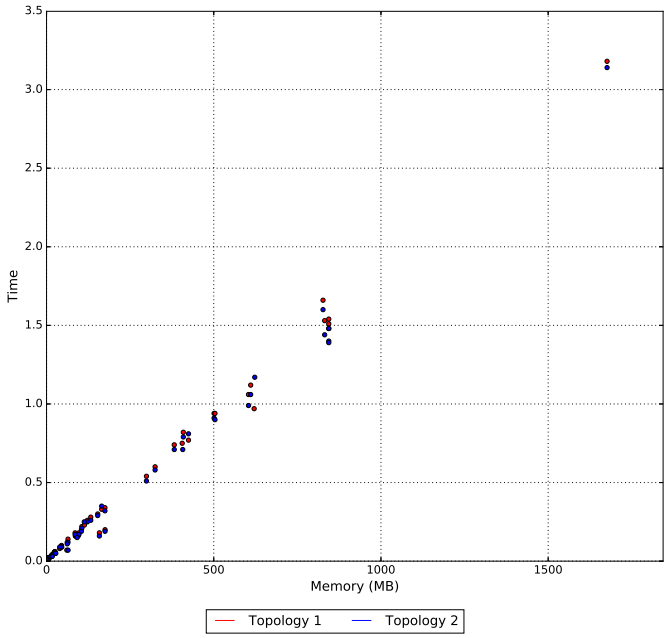
$$\text{Per}(i) = \frac{\text{Time Of Migrate}(i)}{\text{Total Time}_{\text{Local}}} \cdot 100\%$$

Όπου το i μας δείχνει την εκάστοτε τοπολογία.

Ας σχολιάσουμε τα αποτελέσματα που πήραμε. Αρχικά παρατηρούμε ότι έχουμε μεγάλη ποικιλία όσον αφορά την μνήμη που είχαν όλες οι εφαρμογές πριν την μεταφορά. Αυτό μας δίνει την δυνατότητα να βγάλουμε αρκετά γενικά συμπεράσματα για την συμπεριφορά της migratepages στο μηχάνημα Sandman. Συνεχίζοντας παρατηρούμε ο πραγματικός χρόνος που χρειάζεται η migratepages για να κάνει την μεταφορά είναι πολύ λίγος. Στις περισσότερες περιπτώσεις μάλιστα είναι κάτω 1 δευτερόλεπτο. Όμως υπάρχουν και εξαιρέσεις σε αυτό όπως είναι το 429.mcf που χρειάζεται περίπου ~ 3 δευτερόλεπτα. Ακόμα στον σχήμα 6.1 βλέπουμε ότι το μέγεθος της μνήμης που θέλουμε να μεταφέρουμε σε σχέση με τον χρόνο που χρειαζόμαστε για την μεταφορά είναι γραμμικό. Οπότε μπορούμε πολύ εύκολα να προβλέψουμε τον χρόνο που θέλουμε για την μεταφορά αν γνωρίζουμε το μέγεθος της μνήμης που θέλουμε να μεταφέρουμε.

Τέλος είναι άξιο να σημειωθεί ότι σε όλα τα πειράματα που εκτελέσαμε το ποσοστό

του χρόνου είναι της migratepages κάτω από 1.5%. Αυτό ισχύει τόσο για τις μονονηματικές και πολυνηματικές εφαρμογές όσο και για τις πολλαπλές εφαρμογές στο σύστημα. Άρα μπορούμε να συμπεράνουμε οι ο χρόνος που χρειάζεται η migratepages για να μεταφέρει όλοι την μνήμη σε ένα κόμβο δεν επηρεάζεται άμεσα από το πλήθος των διεργασιών που υπάρχουν ταυτόχρονα στο σύστημα μας.



Σχήμα 6.1: Γραφική αναπαράσταση του πίνακα 6.1.

		Topology 1			Topology 2		
Suite	**Benchmark**	**MBM**	**TOM**	**Per**	**MBM**	**TOM**	**Per**
Spec	400.perlbench_0	153.01	0.3	0.1205	152.97	0.29	0.1165
	400.perlbench_1	86.41	0.16	0.1860	86.41	0.16	0.1860
	400.perlbench_2	555.55	0.68	0.5231	557.05	0.62	0.4769
	401.bzip2_0	843.98	1.54	1.1241	843.97	1.48	1.0803
	401.bzip2_1	90.15	0.17	0.3696	93.74	0.16	0.3478
	401.bzip2_2	94.22	0.17	0.2267	94.22	0.16	0.2133
	401.bzip2_3	843.98	1.51	1.0272	843.97	1.4	0.9524
	401.bzip2_4	843.97	1.48	0.9250	843.97	1.39	0.8687
	401.bzip2_5	603.9	1.06	0.9907	603.91	0.99	0.9252
	403.gcc_0	96.94	0.18	0.6207	96.95	0.17	0.5862
	403.gcc_1	60.63	0.07	0.1346	64.41	0.07	0.1346
	403.gcc_2	44.75	0.1	0.2128	44.77	0.09	0.1915
	403.gcc_3	132.23	0.28	0.8750	132.23	0.26	0.8125
	403.gcc_4	158.1	0.18	0.4737	157.96	0.16	0.4211
	403.gcc_5	105.59	0.22	0.4151	105.6	0.21	0.3962
	403.gcc_6	85.41	0.18	0.2609	85.42	0.17	0.2464
	403.gcc_7	174.93	0.2	0.3279	174.9	0.19	0.3115
	403.gcc_8	21.6	0.05	0.2500	21.62	0.05	0.2500
	410.bwaves	831.67	1.53	0.2567	831.66	1.44	0.2416
	416.gamess_0	3.31	0.02	0.0286	3.31	0.02	0.0286
	416.gamess_1	3.52	0.02	0.1111	3.52	0.02	0.1111
	416.gamess_2	5.04	0.01	0.0051	5.04	0.02	0.0103
	429.mcf	1676.13	3.18	0.6723	1676.13	3.14	0.6638
	433.milc	613.1	0.8	0.1408	613.1	0.68	0.1197
	434.zeusmp	500.94	0.94	0.1374	500.94	0.91	0.1330
	435.gromacs	12.71	0.03	0.0044	12.71	0.03	0.0044
	436.cactusADM	620.76	0.97	0.0845	622.61	1.17	0.1019
	437.leslie3d	122.17	0.26	0.0459	122.16	0.25	0.0441
	444.namd	45.04	0.09	0.0159	45.04	0.09	0.0159
	445.gobmk_0	24.44	0.06	0.0659	24.44	0.05	0.0549
	445.gobmk_1	24.77	0.06	0.0260	24.77	0.05	0.0216
	445.gobmk_2	24.43	0.05	0.0439	24.43	0.05	0.0439
	445.gobmk_3	24.43	0.05	0.0556	24.43	0.05	0.0556
	445.gobmk_4	24.52	0.05	0.0413	24.52	0.05	0.0413
	447.dealII	8.09	0.02	0.0045	8.09	0.02	0.0045
	450.soplex_0	113.32	0.23	0.1544	112.76	0.25	0.1678
	450.soplex_1	424.49	0.77	0.6581	424.48	0.81	0.6923
	453.povray	2.67	0.01	0.0041	2.67	0.01	0.0041
	456.hmmer_0	23.39	0.05	0.0248	23.39	0.05	0.0248
	456.hmmer_1	2.97	0.01	0.0023	2.96	0.01	0.0023
	458.sjeng	174.61	0.34	0.0456	174.59	0.32	0.0429
	459.GemsFDTD	826.75	1.66	0.3086	826.75	1.6	0.2974
	462.libquantum	64.18	0.14	0.0289	64.19	0.12	0.0248
	464.h264ref_0	25.11	0.06	0.0659	25.12	0.05	0.0549
	464.h264ref_1	16.05	0.04	0.0563	16.06	0.03	0.0423
	464.h264ref_2	61.87	0.12	0.0188	61.86	0.11	0.0173
	465.tonto	3.56	0.01	0.0059	3.56	0.02	0.0118
	470.lbm	408.99	0.82	0.1592	408.99	0.79	0.1534
	471.omnetpp	164.85	0.33	0.0791	164.84	0.35	0.0839
	473.astar_0	324.57	0.6	0.2885	324.57	0.58	0.2788
	473.astar_1	27.29	0.05	0.0138	27.29	0.05	0.0138
	482.sphinx3	39.44	0.08	0.0104	39.44	0.09	0.0117
Parsec	blackscholes_0	610.41	1.12	0.4118	610.41	1.06	0.3897
	bodytrack_0	4.67	0.02	0.009	4.67	0.01	0.0045
	canneal_0	704.87	0.74	0.4229	704.74	0.68	0.3886
	dedup_0	803.89	0.82	1.5185	803.78	0.7	1.2963
	facesim_0	298.71	0.54	0.1019	298.71	0.51	0.0962
	ferret_0	91.32	0.17	0.0296	91.33	0.15	0.0261
	fluidanimate_0	503.66	0.94	0.1971	503.66	0.9	0.1887
	freqmine_0	381.87	0.74	0.112	381.88	0.71	0.1074
	rtview_0	405.63	0.75	0.2586	407.34	0.71	0.2448
	swaptions_0	0.44	0.01	0.0027	0.44	0.01	0.0027
	streamcluster_0	104.33	0.2	0.0336	104.33	0.19	0.0319
	vips_0	16.7	0.04	0.0233	16.7	0.03	0.0174

Πίνακας 6.1: Αποτελέσματα της migratepages για μονονηματικά benchmarks

		Topology 1			Topology 2		
Benchmark	**Thread**	**MBM**	**TOM**	**Per**	**MBM**	**TOM**	**Per**
blackscholes_0	2	314.45	0.45	0.3041	313.09	0.46	0.3108
	4	216.04	0.25	0.3205	215.49	0.28	0.3590
	8	150.94	0.18	0.3333	150.79	0.21	0.3889
	16	128.27	0.15	0.3409	129.51	0.18	0.4091
bodytrack_0	2	216.04	0.25	0.3205	215.49	0.28	0.3590
	4	150.94	0.18	0.3333	150.79	0.21	0.3889
	8	128.27	0.15	0.3409	129.51	0.18	0.4091
	16	27.36	0.09	0.0726	27.83	0.05	0.0403
canneal_0	2	150.94	0.18	0.3333	150.79	0.21	0.3889
	4	128.27	0.15	0.3409	129.51	0.18	0.4091
	8	27.36	0.09	0.0726	27.83	0.05	0.0403
	16	27.41	0.05	0.0676	27.26	0.04	0.0541
dedup_0	2	128.27	0.15	0.3409	129.51	0.18	0.4091
	4	27.36	0.09	0.0726	27.83	0.05	0.0403
	8	27.41	0.05	0.0676	27.26	0.04	0.0541
	16	27.04	0.07	0.1429	26.59	0.04	0.0816
facesim_0	2	27.36	0.09	0.0726	27.83	0.05	0.0403
	4	27.41	0.05	0.0676	27.26	0.04	0.0541
	8	27.04	0.07	0.1429	26.59	0.04	0.0816
	16	27.02	0.03	0.0750	27.34	0.06	0.1500
ferret_0	2	27.41	0.05	0.0676	27.26	0.04	0.0541
	4	27.04	0.07	0.1429	26.59	0.04	0.0816
	8	27.02	0.03	0.0750	27.34	0.06	0.1500
	16	664.01	0.65	0.4710	664.1	0.68	0.4928
fluidanimate_0	2	27.04	0.07	0.1429	26.59	0.04	0.0816
	4	27.02	0.03	0.0750	27.34	0.06	0.1500
	8	664.01	0.65	0.4710	664.1	0.68	0.4928
	16	586.24	0.53	0.6386	586.77	0.56	0.6747
streamcluster_0	2	27.02	0.03	0.0750	27.34	0.06	0.1500
	4	664.01	0.65	0.4710	664.1	0.68	0.4928
	8	586.24	0.53	0.6386	586.77	0.56	0.6747
	16	537.03	0.47	0.7460	538.54	0.55	0.8730

Πίνακας 6.2: Αποτελέσματα της migratepages για πολυνηματικά benchmarks

			Mode 1						Mode 2					
			Topology 1			Topology 2			Topology 1			Topology 2		
Processes	**Benchmark**	**Thread**	**MBM**	**TOM**	**Per**	**MBM**	**TOM**	**Per**	**MBM**	**TOM**	**Per**	**MBM**	**TOM**	**Per**
2	blackscholes_0	2	224.92	0.44	0.2803	138.16	0.46	0.293	145.66	0.41	0.2628	148.19	0.46	0.2949
		4	257.53	0.35	0.3043	258.13	0.38	0.3304	221.79	0.28	0.2718	221.24	0.31	0.301
		8	212.02	0.28	0.3373	214.86	0.28	0.3373	196.04	0.2	0.2667	196.25	0.26	0.3467
		16	173.34	0.17	0.2615	174.66	0.22	0.3385	174.89	0.18	0.2727	172.29	0.22	0.3333
	bodytrack_0	2	257.53	0.35	0.3043	258.13	0.38	0.3304	221.79	0.28	0.2718	221.24	0.31	0.301
		4	212.02	0.28	0.3373	214.86	0.28	0.3373	196.04	0.2	0.2667	196.25	0.26	0.3467
		8	173.34	0.17	0.2615	174.66	0.22	0.3385	174.89	0.18	0.2727	172.29	0.22	0.3333
		16	27.36	0.07	0.0547	27.38	0.07	0.0547	27.82	0.08	0.063	27.37	0.08	0.063
	canneal_0	2	212.02	0.28	0.3373	214.86	0.28	0.3373	196.04	0.2	0.2667	196.25	0.26	0.3467
		4	173.34	0.17	0.2615	174.66	0.22	0.3385	174.89	0.18	0.2727	172.29	0.22	0.3333
		8	27.36	0.07	0.0547	27.38	0.07	0.0547	27.82	0.08	0.063	27.37	0.08	0.063
		16	26.77	0.08	0.0941	27.86	0.09	0.1059	27.25	0.04	0.0548	27.4	0.07	0.0959
	dedup_0	2	173.34	0.17	0.2615	174.66	0.22	0.3385	174.89	0.18	0.2727	172.29	0.22	0.3333
		4	27.36	0.07	0.0547	27.38	0.07	0.0547	27.82	0.08	0.063	27.37	0.08	0.063
		8	26.77	0.08	0.0941	27.86	0.09	0.1059	27.25	0.04	0.0548	27.4	0.07	0.0959
		16	27.75	0.08	0.1356	27.3	0.05	0.0847	27.93	0.03	0.0577	27.63	0.04	0.0769
	facesim_0	2	27.36	0.07	0.0547	27.38	0.07	0.0547	27.82	0.08	0.063	27.37	0.08	0.063
		4	26.77	0.08	0.0941	27.86	0.09	0.1059	27.25	0.04	0.0548	27.4	0.07	0.0959
		8	27.75	0.08	0.1356	27.3	0.05	0.0847	27.93	0.03	0.0577	27.63	0.04	0.0769
		16	27.32	0.06	0.1364	27.16	0.04	0.0909	27.61	0.03	0.0625	27.15	0.08	0.1667
	ferret_0	2	26.77	0.08	0.0941	27.86	0.09	0.1059	27.25	0.04	0.0548	27.4	0.07	0.0959
		4	27.75	0.08	0.1356	27.3	0.05	0.0847	27.93	0.03	0.0577	27.63	0.04	0.0769
		8	27.32	0.06	0.1364	27.16	0.04	0.0909	27.61	0.03	0.0625	27.15	0.08	0.1667
		16	661.92	0.58	0.3973	666.45	0.66	0.4521	663.96	0.53	0.3813	664.91	0.62	0.446
	fluidanimate_0	2	27.75	0.08	0.1356	27.3	0.05	0.0847	27.93	0.03	0.0577	27.63	0.04	0.0769
		4	27.32	0.06	0.1364	27.16	0.04	0.0909	27.61	0.03	0.0625	27.15	0.08	0.1667
		8	661.92	0.58	0.3973	666.45	0.66	0.4521	663.96	0.53	0.3813	664.91	0.62	0.446
		16	579.39	0.55	0.5556	579.56	0.62	0.6263	573.34	0.44	0.4783	571.94	0.6	0.6522
	streamcluster_0	2	27.32	0.06	0.1364	27.16	0.04	0.0909	27.61	0.03	0.0625	27.15	0.08	0.1667
		4	661.92	0.58	0.3973	666.45	0.66	0.4521	663.96	0.53	0.3813	664.91	0.62	0.446
		8	579.39	0.55	0.5556	579.56	0.62	0.6263	573.34	0.44	0.4783	571.94	0.6	0.6522
		16	547.22	0.46	0.5974	554.55	0.52	0.6753	539.74	0.36	0.5806	540.21	0.52	0.8387
4	blackscholes_0	2	0.04	1.18	0.5728	0.04	1.8	0.8738	0.04	0.79	0.4072	0.04	1.53	0.7887
		4	277.55	0.51	0.375	277.07	0.46	0.3382	313.44	0.36	0.2384	313.76	0.41	0.2715
		8	324.21	0.37	0.296	322.92	0.44	0.352	302.7	0.33	0.2797	302.45	0.39	0.3305
	bodytrack_0	2	277.55	0.51	0.375	277.07	0.46	0.3382	313.44	0.36	0.2384	313.76	0.41	0.2715
		4	324.21	0.37	0.296	322.92	0.44	0.352	302.7	0.33	0.2797	302.45	0.39	0.3305
		8	27.37	0.03	0.0229	27.4	0.05	0.0382	27.33	0.08	0.0625	27.4	0.04	0.0312
	canneal_0	2	324.21	0.37	0.296	322.92	0.44	0.352	302.7	0.33	0.2797	302.45	0.39	0.3305
		4	27.37	0.03	0.0229	27.4	0.05	0.0382	27.33	0.08	0.0625	27.4	0.04	0.0312
		8	27.36	0.1	0.087	27.4	0.12	0.1043	27.41	0.03	0.0366	27.56	0.06	0.0732
	dedup_0	2	27.37	0.03	0.0229	27.4	0.05	0.0382	27.33	0.08	0.0625	27.4	0.04	0.0312
		4	27.36	0.1	0.087	27.4	0.12	0.1043	27.41	0.03	0.0366	27.56	0.06	0.0732
		8	27.48	0.03	0.0435	27.95	0.08	0.1159	26.39	0.03	0.0508	28.14	0.05	0.0847
	facesim_0	2	27.36	0.1	0.087	27.4	0.12	0.1043	27.41	0.03	0.0366	27.56	0.06	0.0732
		4	27.48	0.03	0.0435	27.95	0.08	0.1159	26.39	0.03	0.0508	28.14	0.05	0.0847
		8	658.05	0.65	0.4167	663.19	0.75	0.4808	663.71	0.63	0.4345	664.26	0.66	0.4552
	ferret_0	2	27.48	0.03	0.0435	27.95	0.08	0.1159	26.39	0.03	0.0508	28.14	0.05	0.0847
		4	658.05	0.65	0.4167	663.19	0.75	0.4808	663.71	0.63	0.4345	664.26	0.66	0.4552
		8	596.03	0.64	0.5161	599.9	0.68	0.5484	580.31	0.47	0.4747	581.2	0.63	0.6364
	fluidanimate_0	2	658.05	0.65	0.4167	663.19	0.75	0.4808	663.71	0.63	0.4345	664.26	0.66	0.4552
		4	596.03	0.64	0.5161	599.9	0.68	0.5484	580.31	0.47	0.4747	581.2	0.63	0.6364
		8	548.97	0.41	0.5256	551.97	0.56	0.7179	548.88	0.38	0.4935	551.09	0.56	0.7273
	streamcluster_0	2	596.03	0.64	0.5161	599.9	0.68	0.5484	580.31	0.47	0.4747	581.2	0.63	0.6364
		4	548.97	0.41	0.5256	551.97	0.56	0.7179	548.88	0.38	0.4935	551.09	0.56	0.7273
		8	1358.14	2.14	0.7405	1356.93	2.86	0.9896	1322.73	1.38	0.4381	1318.61	1.92	0.6095
8	blackscholes_0	2	0.04	2.52	0.7347	0.04	2.55	0.7434	0.04	0.89	0.308	0.04	1.67	0.5779
		4	319.77	0.46	0.2644	316.18	0.55	0.3161	0.04	1.05	0.4217	0.04	1.64	0.6586
	bodytrack_0	2	319.77	0.46	0.2644	316.18	0.55	0.3161	0.04	1.05	0.4217	0.04	1.64	0.6586
		4	27.41	0.11	0.057	27.87	0.12	0.0622	27.39	0.08	0.0615	27.4	0.1	0.0769
	canneal_0	2	27.41	0.11	0.057	27.87	0.12	0.0622	27.39	0.08	0.0615	27.4	0.1	0.0769
		4	27.86	0.23	0.211	27.12	0.06	0.055	27.4	0.07	0.0588	27.41	0.16	0.1345
	dedup_0	2	27.86	0.23	0.211	27.12	0.06	0.055	27.4	0.07	0.0588	27.41	0.16	0.1345
		4	697.53	0.93	0.4493	704.85	1.22	0.5894	658.46	0.66	0.4258	662.94	0.74	0.4774
	facesim_0	2	697.53	0.93	0.4493	704.85	1.22	0.5894	658.46	0.66	0.4258	662.94	0.74	0.4774
		4	573.11	0.47	0.3983	589.86	0.64	0.5424	624.41	0.58	0.4915	626.23	0.8	0.678
	ferret_0	2	573.11	0.47	0.3983	589.86	0.64	0.5424	624.41	0.58	0.4915	626.23	0.8	0.678
		4	1383.55	1.22	0.174	1358.35	1.57	0.224	1359.25	1.14	0.1506	1357.6	2.02	0.2668
	fluidanimate_0	2	1383.55	1.22	0.174	1358.35	1.57	0.224	1359.25	1.14	0.1506	1357.6	2.02	0.2668
		4	1385.41	1.46	0.2561	1394.54	1.65	0.2895	1398.86	1.66	0.255	1377.77	1.62	0.2488
	streamcluster_0	2	1385.41	1.46	0.2561	1394.54	1.65	0.2895	1398.86	1.66	0.255	1377.77	1.62	0.2488
		4	301.55	1.12	0.2309	301.55	2.89	0.5959	301.55	0.47	0.1632	301.55	1.57	0.5451

Πίνακας 6.3: Αποτελέσματα της migratepages για πολλά πολυνηματικά benchmarks ταυτόχρονα στο σύστημα

Κεφάλαιο 7

Επίλογος

Με το κεφάλαιο αυτό ολοκληρώνεται η παρούσα διπλωματική εργασία. Στο κεφάλαιο αυτό παρουσιάζουμε ορισμένα γενικότερα συμπεράσματα που βγάλαμε από τη διπλωματική και αναφέρουμε ορισμένες μελλοντικές επεκτάσεις που έχουν ενδιαφέρον.

7.1 Συμπεράσματα

Στο πρώτο κεφάλαιο είχαμε αναφερθεί στους στόχους της διπλωματικής. Ας τους ξαναθυμηθούμε και ας εξετάσουμε κατά πόσο μπορέσαμε να τους πετύχουμε.

- Να βρούμε και να μετρήσουμε κατάλληλες παραμέτρους από τις εφαρμογές μας, που μπορούν να ερμηνεύσουν τη συμπεριφορά τους, ανεξάρτητα από το σύστημα που εξετάζουμε στην εκάστοτε περίπτωση.

Στο κεφάλαιο 5 είδαμε αρκετές παραμέτρους των εφαρμογών που μπορούν να μετρηθούν και στα δύο μηχανήματα που χρησιμοποιήσαμε στην έρευνα μας. Ακόμα οι παράμετροι που είδαμε μπορούν να ερμηνεύσουν αρκετές από τις συμπεριφορές των εφαρμογών μας.

Συμπεραίνουμε ότι οι παράμετροι MPKI, IPC και TLB, που έχουν μετρηθεί με το εργαλείο perf, είναι αρκετά αξιόπιστοι για να περιγράψουν την συμπεριφορά των εφαρμογών μας, σε αντίθεση με την παράμετρο IMC που θα προτείναμε να μην χρησιμοποιηθεί καθώς εμφάνιζε αρκετά προβλήματα η χρήση της. Ακόμα η παράμετρος BW, που έχει μετρηθεί με το εργαλείο pcm-memory.x, είναι εξίσου αξιόπιστη με της προηγούμενες. Οπότε το σύνολο των παραμέτρων {MPKI, IPC, TLB, BW} μπορεί να μετρηθεί το ίδιο καλά και στα δύο μηχανήματα και να ερμηνεύσει την συμπεριφορά των εφαρμογών μας.

- Να μοντελοποιήσουμε αυτές τις εφαρμογές με σκοπό να κατασκευάσουμε μια μέθοδο που θα μπορεί να προβλέψει την μεταβολή της απόδοσης τους σε σχέση με την θέση της μνήμης τους σε ένα NUMA σύστημα.

Στο κεφάλαιο 5 είδαμε ότι μοντελοποιήσαμε όλες τις εφαρμογές μας ως σημεία που έχουν συγκεκριμένες παραμέτρους. Στην συνέχεια πήραμε αυτά τα σημεία και κατασκευάσαμε μια μέθοδο βάσει της οποίας μπορούμε να προβλέψουμε την απόδοση των εφαρμογών μας σε σχέση με την θέση της μνήμης τους.

Συμπεραίνουμε ότι η μοντελοποίηση που κάναμε καθώς και η μέθοδος που αναπτύξαμε για να μπορέσουμε να προβλέψουμε την απόδοση των εφαρμογών μας είναι αρκετά

ικανοποιητικές. Η μέθοδος πρόβλεψης που κατασκευάσαμε για τις καλύτερες συναρτήσεις και για τα δύο μηχανήματα είχαν $R^2 > 0.93$ και MAE (Relative) $< 5\%$. Με αυτόν τον τρόπο μπορούμε να προβλέψουμε την απόδοση των εφαρμογών με μεγάλη ακρίβεια.

- Να μελετήσουμε τους τρόπους που μπορούμε να μεταφέρουμε την μνήμη μιας εφαρμογής μέσα σε ένα NUMA σύστημα.

Στο κεφάλαιο 6 είδαμε κάποιους βασικούς τρόπους που μπορούμε να χρησιμοποιήσουμε για να μεταφέρουμε την μνήμη μιας διεργασίας κατά την διάρκεια εκτέλεσης της. Ασχοληθήκαμε κυρίως με την κλήση Migratepages, όμως όπως είδαμε και η κλήση Movepages είναι εξίσου σημαντική.

Συμπεραίνουμε ότι οι συναρτήσεις για μεταφορά της μνήμης κατά την διάρκεια εκτέλεσης είναι αρκετά χρήσιμες. Αυτές σε συνδυασμό με τις συναρτήσεις πρόβλεψης της απόδοσης μπορούμε να τις συγχωνεύσουμε για να υπολογίσουμε το συνολικό κόστος που θα έχουμε στην απόδοση των εφαρμογών μας. Έτσι μπορούμε και να μεταφέρουμε την μνήμη των εφαρμογών μας άλλα και να γνωρίζουμε από πριν αν η επιλογή που κάναμε για την μεταφορά είναι καλή.

7.2 Μελλοντικές επεκτάσεις

Τελειώνοντας τη διπλωματική εργασία ας δούμε μερικές μελλοντικές επεκτάσεις που μπορούμε να κάνουμε στην έρευνα μας.

- Να πειραματιστούμε και με ασύμμετρα μηχανήματα για να δούμε αν οι μέθοδοι που αναπτύξαμε εδώ ανταποκρίνονται το ίδιο καλά και σε τέτοια μηχανήματα.
- Να μελετήσουμε μηχανήματα μεγαλύτερης κλίμακας.
- Να πειραματιστούμε με περισσότερα ποσοστά της μνήμης πέρα από 50% και 100%.
- Να επεκτείνουμε το μοντέλο που έχουμε για τις προβλέψεις έτσι ώστε να μπορεί να υπολογίσει και το κέρδος όταν μεταφερόμαστε από μια μακρινή τοπολογία σε μια τοπική.
- Να επεκτείνουμε τις συναρτήσεις που κατασκευάσαμε στον παραγωγό δεύτερης γενιάς έτσι ώστε να δέχονται κάποιο όρισμα για την κατάσταση του μηχανήματος, όπως διαθέσιμη μνήμη, αριθμός διεργασιών, ελεύθεροι πόροι και άλλα.
- Να πειραματιστούμε περισσότερο με την κλήση movepages έτσι ώστε να μπορούμε να την χρησιμοποιήσουμε πιο αποδοτικά.
- Να κατασκευάσουμε ένα εργαλείο που όταν θα πραγματοποιούμε την μεταφορά μνήμης μεταξύ δύο κόμβων, ο κόμβος παραλαβής θα γίνεται ο νέος κόμβος δέσμευσης μνήμης για την εφαρμογή.
- Να επεκτείνουμε το εργαλείο numactl έτσι ώστε να να δέχεται σαν όρισμα ένα ποσοστό μνήμης και να επιτελεί την ίδια λειτουργία με την interleave κλήση.

Οποιαδήποτε από αυτές τις μελλοντικές επεκτάσεις θα βοηθήσουν στην καλύτερη λειτουργία του μοντέλου που κατασκευάσαμε καθώς και στην πρακτική του χρήση από κάποιον scheduler που θα οργάνωνε τις διεργασίες σε ένα NUMA σύστημα με σκοπό την αποδοτικότερη εκμετάλλευση όλων των πόρων του συστήματος.

Κεφάλαιο 8

Παράτημα Α

Σε αυτό το παράρτημα της εργασίας παρουσιάζουμε τα αποτελέσματα που πήραμε από τα benchmark. Τα αποτελέσματα αυτά αντιστοιχούν στο κεφάλαιο 5.1.2.

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (2)	IMC (3)
400.perlbench_0	248	0.0355	1.8831	0.3816	43.4390	0.0000	0.0000	5.6490	36.3346
400.perlbench_1	87	0.0950	1.8547	0.2462	92.5139	0.0000	0.0000	47.2093	50.2737
400.perlbench_2	131	0.3313	2.2186	0.1303	270.3549	0.0000	0.0000	131.7528	136.2620
401.bzip2_0	139	0.2252	1.2921	0.3403	100.6323	0.0000	0.0000	57.5005	60.9272
401.bzip2_1	46	0.0810	1.7040	2.2360	57.6362	0.0000	0.0000	32.0041	32.5528
401.bzip2_2	75	0.0499	1.7310	2.9445	45.3138	0.0000	0.0000	25.1449	24.0495
401.bzip2_3	148	0.1850	1.5845	0.2326	98.0005	0.0000	0.0000	58.3637	63.2707
401.bzip2_4	162	0.1479	1.6791	0.2733	87.8578	0.0000	0.0000	51.9131	57.1381
401.bzip2_5	107	0.1815	1.2612	0.5456	85.7989	0.0000	0.0000	52.0961	58.8023
403.gcc_0	29	2.6527	1.1715	1.4278	1029.5819	0.0000	0.0000	509.8559	475.5697
403.gcc_1	52	1.1012	1.2284	1.3422	424.6433	0.0000	0.0000	211.4494	203.2044
403.gcc_2	47	1.3475	1.2687	0.8319	621.7437	0.0000	0.0000	302.7372	288.4387
403.gcc_3	32	1.3732	1.3952	1.0030	687.0416	0.0000	0.0000	336.6475	325.6144
403.gcc_4	38	1.9841	1.2582	1.2108	900.6831	0.0000	0.0000	433.9092	418.3421
403.gcc_5	53	2.1939	1.2619	1.1830	954.3396	0.0000	0.0000	462.1396	451.0957
403.gcc_6	70	6.9281	1.1421	0.8513	2386.9423	0.0000	0.0000	1143.4374	1119.6201
403.gcc_7	61	3.4251	1.2022	1.1658	1292.9105	0.0000	0.0000	635.8749	601.0167
403.gcc_8	20	0.5555	1.2450	1.0705	229.9139	0.0000	0.0000	117.9555	105.2970
410.bwaves	605	9.0010	1.5944	0.3081	4255.8591	0.0000	0.0000	2025.3887	2032.0324
416.gamess_0	79	0.0181	2.1078	0.0109	36.4000	0.0000	0.0000	10.5942	24.4844
416.gamess_1	18	0.0213	1.9031	0.0089	37.2113	0.0000	0.0000	9.3856	21.7828
416.gamess_2	195	0.0153	2.3970	0.0085	35.8220	0.0000	0.0000	11.6862	22.7248
429.mcf	477	27.3083	0.2972	56.5865	1541.2537	0.0000	0.0000	740.5913	734.2012
433.milc	603	19.8729	0.8849	1.8317	4699.4866	0.0000	0.0000	2357.3540	2227.9124
434.zeusmp	690	3.1597	1.1734	3.1270	1194.2250	0.0000	0.0000	564.6733	591.8601
435.gromacs	685	0.0261	1.3392	0.0360	34.2029	0.0000	0.0000	18.0439	17.4590
436.cactusADM	1145	1.3844	1.0824	13.4376	569.6832	0.0000	0.0000	290.4008	254.2036
437.leslie3d	567	7.9483	1.1930	0.4716	3829.3011	0.0000	0.0000	1811.6668	1831.7758
444.namd	568	0.0329	1.7392	0.0093	39.7026	0.0000	0.0000	13.7674	27.3137
445.gobmk_0	91	0.3337	1.1011	0.4840	144.3262	0.0000	0.0000	89.8300	55.0910
445.gobmk_1	231	0.2706	1.1420	0.4810	125.1073	0.0000	0.0000	77.5158	44.9364
445.gobmk_2	115	1.6369	1.1942	0.3134	681.7429	0.0000	0.0000	342.9186	311.1252
445.gobmk_3	90	0.4821	1.1004	0.4760	196.3480	0.0000	0.0000	116.9580	82.1943
445.gobmk_4	121	0.2621	1.1819	0.4315	124.9729	0.0000	0.0000	78.9448	45.8899
447.dealII	452	1.5809	1.9091	0.1704	858.0356	0.0000	0.0000	425.2927	391.0022
450.soplex_0	182	5.8711	0.8079	3.0840	1307.7999	0.0000	0.0000	680.4532	656.9505
450.soplex_1	141	13.5859	1.1051	1.8470	4918.6157	0.0000	0.0000	2383.5215	2390.3774
453.povray	242	0.0198	1.7375	0.1934	33.0305	0.0000	0.0000	4.0464	33.1176
456.hmmer_0	202	0.0258	1.9064	0.0349	48.4126	0.0000	0.0000	9.7275	37.4555
456.hmmer_1	435	0.0197	1.8750	0.0094	30.2147	0.0000	0.0000	3.5909	29.7246
458.sjeng	746	0.4077	1.3115	0.4432	158.5994	0.0000	0.0000	81.2035	76.1690
459.GemsFDTD	539	14.3958	1.2169	4.6500	5427.6558	0.0000	0.0000	2613.9163	2613.9753
462.libquantum	484	26.2881	1.4783	0.5673	9863.0041	0.0000	0.0000	4749.2943	4777.3628
464.h264ref_0	91	0.0216	2.4050	0.0709	35.0228	0.0000	0.0000	31.3335	9.4513
464.h264ref_1	71	0.0209	1.9867	0.0417	35.7786	0.0000	0.0000	17.8163	17.9649
464.h264ref_2	625	0.0326	2.0763	0.1935	46.5670	0.0000	0.0000	22.5808	22.3719
465.tonto	169	0.0558	1.8053	0.0412	51.7725	0.0000	0.0000	28.2299	28.4633
470.lbm	507	4.5687	1.1212	0.5266	3785.8529	0.0000	0.0000	1823.3629	1821.1087
471.omnetpp	443	9.9626	0.5748	8.1747	1380.6427	0.0000	0.0000	679.0666	650.6543
473.astar_0	208	1.9581	0.7959	12.4136	392.1865	0.0000	0.0000	204.2105	177.0991
473.astar_1	363	0.0526	0.9754	3.6654	38.3718	0.0000	0.0000	33.2761	5.2511
482.sphinx3	774	0.0574	1.7121	0.6140	63.6397	0.0000	0.0000	17.9886	35.8984

Πίνακας 8.1: Μηχάνημα: Sandman, Τοπολογία: 0 (local), Σουίτα: Spec

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (2)	IMC (3)
400.perlbench_0	250	0.0327	1.8711	0.3752	43.7261	0.0000	0.0000	6.4500	27.5074
400.perlbench_1	89	0.1005	1.8117	0.2572	105.1734	0.0000	0.0000	38.4880	55.8788
400.perlbench_2	135	0.3388	2.1481	0.1296	353.9850	0.0000	0.0000	153.7018	175.1561
401.bzip2_0	140	0.2230	1.2917	0.3422	108.3639	0.0000	0.0000	49.1452	69.9091
401.bzip2_1	47	0.0789	1.6919	2.2565	65.1529	0.0000	0.0000	20.0474	41.3830
401.bzip2_2	75	0.0487	1.7644	2.9309	47.6807	0.0000	0.0000	11.4999	32.0119
401.bzip2_3	149	0.1822	1.5788	0.2288	109.5220	0.0000	0.0000	50.2938	71.0378
401.bzip2_4	163	0.1456	1.6846	0.2676	96.2104	0.0000	0.0000	41.8347	62.5869
401.bzip2_5	108	0.1775	1.3383	0.5485	94.8172	0.0000	0.0000	43.8652	64.4946
403.gcc_0	41	2.7200	0.8211	1.4720	841.5327	0.0000	0.0000	414.0183	379.2734
403.gcc_1	66	1.1164	0.9637	1.3455	410.3745	0.0000	0.0000	199.4224	184.1576
403.gcc_2	58	1.4677	1.0309	0.8132	620.5496	0.0000	0.0000	297.6388	280.1929
403.gcc_3	41	1.4998	1.1028	0.9947	672.9900	0.0000	0.0000	320.1061	300.1824
403.gcc_4	51	2.1573	0.9656	1.2326	809.7310	0.0000	0.0000	389.1237	367.2110
403.gcc_5	73	2.3555	0.9198	1.1801	836.5420	0.0000	0.0000	399.8975	379.1538
403.gcc_6	129	7.4774	0.6207	0.8657	1391.9128	0.0000	0.0000	664.3720	638.6799
403.gcc_7	91	3.6026	0.8067	1.1972	992.6201	0.0000	0.0000	476.6516	454.4488
403.gcc_8	23	0.5690	1.0850	1.0747	249.3443	0.0000	0.0000	116.0343	104.8417
410.bwaves	1008	10.5779	0.9618	0.3046	2810.8979	0.0000	0.0000	1337.0579	1325.9198
416.gamess_0	77	0.0176	2.0872	0.0110	32.8583	0.0000	0.0000	17.8372	6.6132
416.gamess_1	18	0.0193	2.3731	0.0081	37.2667	0.0000	0.0000	19.0989	7.8533
416.gamess_2	195	0.0151	2.3049	0.0080	35.4985	0.0000	0.0000	17.8635	5.1600
429.mcf	679	26.9724	0.2179	57.6682	1237.7274	0.0000	0.0000	590.8367	562.3377
433.milc	1632	20.2668	0.3344	1.8737	2073.5000	0.0000	0.0000	986.9907	976.4330
434.zeusmp	1001	3.8492	0.8095	3.1616	1057.8693	0.0000	0.0000	513.1895	490.7309
435.gromacs	684	0.0259	1.3219	0.0448	33.8198	0.0000	0.0000	23.2537	1.8462
436.cactusADM	1430	1.6743	0.8689	13.7611	579.4678	0.0000	0.0000	274.0159	277.8031
437.leslie3d	806	10.4364	0.8408	0.4781	3456.2040	0.0000	0.0000	1640.7727	1651.2074
444.namd	570	0.0338	1.7057	0.0094	45.4011	0.0000	0.0000	22.4153	22.6049
445.gobmk_0	93	0.3183	1.0683	0.4818	191.1846	0.0000	0.0000	104.4477	82.4417
445.gobmk_1	236	0.2568	1.1166	0.4796	165.2321	0.0000	0.0000	89.7549	67.7538
445.gobmk_2	124	1.4870	1.1029	0.3155	922.6298	0.0000	0.0000	456.3195	434.3841
445.gobmk_3	93	0.4516	1.0716	0.4781	268.7104	0.0000	0.0000	142.9939	121.1938
445.gobmk_4	124	0.2489	1.1578	0.4284	164.1975	0.0000	0.0000	90.3987	69.0100
447.dealII	550	2.0626	1.5693	0.1715	779.1491	0.0000	0.0000	377.9166	356.6820
450.soplex_0	277	6.0942	0.5354	3.3391	1088.7125	0.0000	0.0000	486.1944	516.8176
450.soplex_1	303	19.1268	0.5216	2.0948	2627.9478	0.0000	0.0000	1234.0014	1255.5525
453.povray	242	0.0199	1.7291	0.1822	32.9198	0.0000	0.0000	1.4917	22.3523
456.hmmer_0	202	0.0225	1.9273	0.0331	47.4683	0.0000	0.0000	10.9639	32.6335
456.hmmer_1	436	0.0182	1.8430	0.0106	33.5859	0.0000	0.0000	1.5251	21.4917
458.sjeng	812	0.3991	1.2111	0.4414	197.5541	0.0000	0.0000	81.7071	103.5558
459.GemsFDTD	1162	15.4279	0.5732	4.5514	3350.7140	0.0000	0.0000	1600.6015	1590.7087
462.libquantum	1358	27.0118	0.5345	0.5899	4559.3975	0.0000	0.0000	2202.7345	2202.3712
464.h264ref_0	92	0.0194	2.4109	0.0691	38.0466	0.0000	0.0000	5.1111	27.4145
464.h264ref_1	71	0.0183	2.0624	0.0414	31.6959	0.0000	0.0000	1.9192	21.7841
464.h264ref_2	626	0.0317	2.1088	0.1923	45.9735	0.0000	0.0000	8.0144	28.8834
465.tonto	169	0.0568	1.8053	0.0400	67.2502	0.0000	0.0000	23.4299	35.0537
470.lbm	1024	5.1410	0.5605	0.5366	2551.5035	0.0000	0.0000	1203.9705	1206.3147
471.omnetpp	865	10.1373	0.2979	8.0137	872.0730	0.0000	0.0000	422.5770	403.0809
473.astar_0	244	1.9950	0.6850	12.4771	407.1061	0.0000	0.0000	191.1598	195.8023
473.astar_1	364	0.0494	0.9789	3.6267	37.1075	0.0000	0.0000	24.8255	3.7383
482.sphinx3	785	0.0670	1.6701	0.6067	63.8292	0.0000	0.0000	35.7346	28.0014

Πίνακας 8.2: Μηχάνημα: Sandman, Τοπολογία: 1 (remote near), Σουίτα: Spec
Numactl: membind

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (2)	IMC (3)
400.perlbench_0	250	0.0344	1.8658	0.3922	45.3491	0.0000	0.0000	5.8955	27.9403
400.perlbench_1	89	0.1012	1.8065	0.2682	106.8444	0.0000	0.0000	38.4321	58.9896
400.perlbench_2	137	0.3366	2.1143	0.1331	355.3588	0.0000	0.0000	154.3669	175.9205
401.bzip2_0	141	0.2213	1.2545	0.3490	108.9594	0.0000	0.0000	50.1071	71.4782
401.bzip2_1	47	0.0773	1.6116	2.2602	63.5458	0.0000	0.0000	20.3591	41.0481
401.bzip2_2	75	0.0481	1.7692	2.9687	46.3579	0.0000	0.0000	11.8839	34.0267
401.bzip2_3	150	0.1826	1.5748	0.2340	111.1159	0.0000	0.0000	51.3116	71.4648
401.bzip2_4	163	0.1465	1.6781	0.2776	96.9551	0.0000	0.0000	42.4590	63.2123
401.bzip2_5	109	0.1764	1.3308	0.5493	96.1255	0.0000	0.0000	45.1999	64.7678
403.gcc_0	43	2.8730	0.7840	1.4338	827.4502	0.0000	0.0000	398.2288	366.0414
403.gcc_1	68	1.1252	0.9398	1.3274	403.3964	0.0000	0.0000	195.7710	182.0141
403.gcc_2	59	1.4614	1.0084	0.8353	608.8766	0.0000	0.0000	290.8688	272.9928
403.gcc_3	42	1.4903	1.0804	0.9914	659.9267	0.0000	0.0000	315.9250	296.4526
403.gcc_4	52	2.1587	0.9410	1.2477	784.2684	0.0000	0.0000	377.0689	354.6845
403.gcc_5	75	2.3494	0.8955	1.1727	815.0910	0.0000	0.0000	392.5520	372.8473
403.gcc_6	130	7.4073	0.6140	0.8695	1374.3619	0.0000	0.0000	658.7806	633.8366
403.gcc_7	93	3.5724	0.7913	1.1836	972.2829	0.0000	0.0000	467.9913	447.2491
403.gcc_8	23	0.5773	1.0686	1.0779	244.8473	0.0000	0.0000	112.6150	100.7658
410.bwaves	1029	10.5538	0.9413	0.3041	2742.4271	0.0000	0.0000	1308.1120	1297.0439
416.gamess_0	78	0.0175	2.1074	0.0109	35.4018	0.0000	0.0000	17.4342	7.4008
416.gamess_1	18	0.0190	2.3341	0.0083	36.8400	0.0000	0.0000	19.2671	5.2265
416.gamess_2	195	0.0151	2.3530	0.0081	33.1659	0.0000	0.0000	17.8175	7.1658
429.mcf	720	26.7698	0.2083	56.6566	1189.2570	0.0000	0.0000	567.3911	534.7227
433.milc	1678	20.1954	0.3257	1.9146	2010.3087	0.0000	0.0000	964.0115	944.5717
434.zeusmp	1055	3.8714	0.7707	3.1431	1003.2800	0.0000	0.0000	488.1457	466.0307
435.gromacs	686	0.0265	1.3383	0.0335	36.1943	0.0000	0.0000	24.1516	2.2767
436.cactusADM	1449	1.7494	0.8581	13.4537	571.0473	0.0000	0.0000	265.4655	276.4133
437.leslie3d	862	10.3223	0.7860	0.4857	3209.1167	0.0000	0.0000	1528.2786	1534.8720
444.namd	570	0.0324	1.6998	0.0094	45.3911	0.0000	0.0000	21.8181	22.0085
445.gobmk_0	94	0.3131	1.0187	0.4851	190.7530	0.0000	0.0000	101.8582	80.6331
445.gobmk_1	239	0.2572	1.0996	0.4832	162.7314	0.0000	0.0000	88.4369	66.5993
445.gobmk_2	128	1.4762	1.0552	0.3112	885.6458	0.0000	0.0000	438.9795	415.9774
445.gobmk_3	94	0.4480	1.0573	0.4713	263.4389	0.0000	0.0000	139.7317	117.2017
445.gobmk_4	125	0.2458	1.1200	0.4290	158.1252	0.0000	0.0000	90.3786	67.9589
447.dealII	556	2.0427	1.5348	0.1710	767.0363	0.0000	0.0000	368.3054	358.0639
450.soplex_0	284	5.7787	0.5210	3.3877	1010.0198	0.0000	0.0000	481.0691	458.4837
450.soplex_1	313	18.7415	0.5048	2.0305	2535.8600	0.0000	0.0000	1200.7522	1222.3531
453.povray	243	0.0199	1.7305	0.0789	33.7724	0.0000	0.0000	1.3353	22.1269
456.hmmer_0	202	0.0232	1.9335	0.0332	48.6700	0.0000	0.0000	10.9418	32.3073
456.hmmer_1	437	0.0185	1.8831	0.0102	31.3281	0.0000	0.0000	1.4443	21.9966
458.sjeng	834	0.4007	1.1796	0.4421	194.5846	0.0000	0.0000	82.4616	100.7216
459.GemsFDTD	1287	15.1734	0.5196	4.5783	3029.7211	0.0000	0.0000	1441.1408	1431.9392
462.libquantum	1531	26.7039	0.4731	0.5995	4088.9468	0.0000	0.0000	1949.6451	1971.8817
464.h264ref_0	91	0.0186	2.3387	0.0702	33.0184	0.0000	0.0000	4.7302	25.2034
464.h264ref_1	71	0.0191	2.0650	0.0418	33.6152	0.0000	0.0000	2.1860	23.2129
464.h264ref_2	625	0.0327	2.1100	0.1926	46.5828	0.0000	0.0000	12.3899	24.1569
465.tonto	170	0.0563	1.7627	0.0397	68.6280	0.0000	0.0000	22.2898	35.4246
470.lbm	1090	5.1956	0.5270	0.5421	2388.4266	0.0000	0.0000	1143.9699	1121.5651
471.omnetpp	915	10.0919	0.2821	8.1106	817.1900	0.0000	0.0000	397.8569	377.7271
473.astar_0	249	1.9107	0.6604	12.4246	401.2646	0.0000	0.0000	173.0525	196.7794
473.astar_1	362	0.0503	0.9844	3.6879	36.5052	0.0000	0.0000	12.3231	15.1360
482.sphinx3	779	0.0502	1.6905	0.6086	56.4924	0.0000	0.0000	16.1116	39.4611

Πίνακας 8.3: Μηχάνημα: Sandman, Τοπολογία: 2 (remote far), Σουίτα: Spec
Numactl: membind

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (2)	IMC (3)
400.perlbench_0	253	0.0360	1.8479	0.3792	39.2258	0.0000	0.0000	26.4449	5.9793
400.perlbench_1	87	0.0973	1.8468	0.2638	97.5575	0.0000	0.0000	54.7554	34.1970
400.perlbench_2	133	0.3255	2.1805	0.1326	315.5918	0.0000	0.0000	156.1430	134.7311
401.bzip2_0	139	0.2174	1.2972	0.3520	101.6150	0.0000	0.0000	61.3443	40.7323
401.bzip2_1	47	0.0789	1.6977	2.2629	59.5720	0.0000	0.0000	40.3011	18.1262
401.bzip2_2	75	0.0485	1.7738	2.9537	45.4777	0.0000	0.0000	30.0144	9.9319
401.bzip2_3	149	0.1789	1.5915	0.2402	104.1090	0.0000	0.0000	63.2788	41.7437
401.bzip2_4	162	0.1414	1.6904	0.2772	91.2887	0.0000	0.0000	56.4796	34.2747
401.bzip2_5	108	0.1778	1.3460	0.5522	83.9247	0.0000	0.0000	57.3503	35.4820
403.gcc_0	35	2.7747	0.9500	1.4678	911.7176	0.0000	0.0000	451.2283	404.4719
403.gcc_1	59	1.1171	1.0747	1.3644	418.0945	0.0000	0.0000	211.2419	184.8795
403.gcc_2	53	1.4278	1.1392	0.8427	621.3616	0.0000	0.0000	303.7821	273.9530
403.gcc_3	37	1.4492	1.2301	0.9944	677.8909	0.0000	0.0000	329.7238	299.7895
403.gcc_4	44	2.1033	1.1097	1.2215	844.6921	0.0000	0.0000	419.9430	384.6875
403.gcc_5	63	2.2934	1.0641	1.2167	892.9982	0.0000	0.0000	434.6176	399.6338
403.gcc_6	98	7.4211	0.8127	0.8726	1743.1302	0.0000	0.0000	845.5030	804.6713
403.gcc_7	76	3.5667	0.9694	1.1785	1112.8489	0.0000	0.0000	534.0516	508.6322
403.gcc_8	22	0.5635	1.1538	1.0778	242.4090	0.0000	0.0000	111.7855	96.7955
410.bwaves	796	9.9817	1.2145	0.3546	3387.5847	0.0000	0.0000	1606.9517	1604.8007
416.gamess_0	77	0.0177	2.1193	0.0123	28.7059	0.0000	0.0000	1.3581	20.9449
416.gamess_1	18	0.0199	2.3602	0.0085	34.4187	0.0000	0.0000	2.5328	20.7017
416.gamess_2	195	0.0154	2.3910	0.0084	32.7296	0.0000	0.0000	1.4667	21.4611
429.mcf	570	27.8502	0.2555	52.6143	1400.2861	0.0000	0.0000	653.8968	666.7853
433.milc	1098	20.4865	0.4911	1.8975	2772.0349	0.0000	0.0000	1344.0955	1312.8247
434.zeusmp	863	3.5826	0.9388	3.1641	1094.9518	0.0000	0.0000	524.1344	513.3135
435.gromacs	685	0.0267	1.3486	0.0270	34.3310	0.0000	0.0000	17.3078	6.3150
436.cactusADM	1304	1.5269	0.9278	14.1628	566.6713	0.0000	0.0000	257.7521	269.4143
437.leslie3d	657	9.4535	1.0293	0.4775	3763.4161	0.0000	0.0000	1789.8324	1798.6522
444.namd	570	0.0345	1.7598	0.0131	45.4180	0.0000	0.0000	21.5781	12.7083
445.gobmk_0	92	0.3372	1.0847	0.4806	170.8593	0.0000	0.0000	90.6430	68.6173
445.gobmk_1	233	0.2730	1.1299	0.4835	147.2500	0.0000	0.0000	78.7573	56.6158
445.gobmk_2	118	1.5921	1.1583	0.3106	817.3502	0.0000	0.0000	402.6825	380.4709
445.gobmk_3	91	0.4797	1.0904	0.4773	237.2038	0.0000	0.0000	123.9665	102.0038
445.gobmk_4	122	0.2640	1.1700	0.4293	144.2716	0.0000	0.0000	79.4775	57.4269
447.dealII	498	1.8889	1.7299	0.1745	821.1662	0.0000	0.0000	394.9293	373.2595
450.soplex_0	230	6.2448	0.6542	3.2508	1187.0626	0.0000	0.0000	558.2981	533.2661
450.soplex_1	214	16.9399	0.7314	2.0205	3516.4283	0.0000	0.0000	1691.9114	1668.9266
453.povray	242	0.0200	1.7486	0.1548	34.1056	0.0000	0.0000	4.3574	17.1074
456.hmmer_0	202	0.0223	1.9327	0.0343	49.5650	0.0000	0.0000	12.7337	24.0555
456.hmmer_1	435	0.0179	1.8949	0.0098	34.3579	0.0000	0.0000	4.1114	17.0784
458.sjeng	783	0.4068	1.2557	0.4458	176.6087	0.0000	0.0000	79.1856	88.3841
459.GemsFDTD	812	15.6808	0.8059	4.6258	4159.0521	0.0000	0.0000	2019.7870	2010.4137
462.libquantum	892	27.1465	0.8209	0.5722	6179.7988	0.0000	0.0000	2949.4847	2952.0631
464.h264ref_0	91	0.0204	2.4203	0.0705	43.4022	0.0000	0.0000	20.6133	9.2710
464.h264ref_1	71	0.0188	2.0887	0.0411	35.9681	0.0000	0.0000	18.0792	5.6246
464.h264ref_2	625	0.0323	2.1129	0.1965	49.2912	0.0000	0.0000	23.2974	12.4081
465.tonto	170	0.0590	1.8048	0.0408	71.0026	0.0000	0.0000	15.2680	36.2457
470.lbm	752	4.9659	0.7583	0.5439	3060.4606	0.0000	0.0000	1428.2898	1450.8088
471.omnetpp	654	10.1985	0.3910	8.0962	1026.8277	0.0000	0.0000	482.8269	507.9263
473.astar_0	227	1.9792	0.7411	12.4242	390.2450	0.0000	0.0000	172.5844	196.2691
473.astar_1	364	0.0503	0.9701	3.6562	37.4025	0.0000	0.0000	17.6703	9.5283
482.sphinx3	783	0.0693	1.7715	0.6071	67.0310	0.0000	0.0000	28.6868	17.1439

Πίνακας 8.4: Μηχάνημα: Sandman, Τοπολογία: 1 (remote near), Σουίτα: Spec
Numactl: interleave

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (2)	IMC (3)
400.perlbench_0	252	0.0366	1.8523	0.3847	42.0793	0.0000	0.0000	26.9203	5.7296
400.perlbench_1	87	0.0980	1.8489	0.2442	96.8680	0.0000	0.0000	56.3708	33.4809
400.perlbench_2	134	0.3277	2.1562	0.1345	314.0626	0.0000	0.0000	158.1372	135.6777
401.bzip2_0	139	0.2219	1.2934	0.3499	101.7541	0.0000	0.0000	63.5145	41.5740
401.bzip2_1	47	0.0781	1.6962	2.2434	57.6682	0.0000	0.0000	36.7532	17.0115
401.bzip2_2	75	0.0488	1.7701	2.9440	40.1318	0.0000	0.0000	31.2309	9.8271
401.bzip2_3	149	0.1830	1.5825	0.2356	104.1784	0.0000	0.0000	65.2065	42.5862
401.bzip2_4	162	0.1457	1.6847	0.2744	89.8094	0.0000	0.0000	56.8335	35.2760
401.bzip2_5	108	0.1817	1.3427	0.5474	88.8192	0.0000	0.0000	57.5989	36.2656
403.gcc_0	36	2.7886	0.9342	1.4665	895.2009	0.0000	0.0000	447.2497	400.4483
403.gcc_1	60	1.1190	1.0569	1.3447	409.9846	0.0000	0.0000	208.1183	181.9705
403.gcc_2	54	1.4390	1.1175	0.8311	615.1483	0.0000	0.0000	298.0891	269.4331
403.gcc_3	37	1.4527	1.2133	0.9910	673.6423	0.0000	0.0000	330.0870	300.3089
403.gcc_4	45	2.1130	1.0902	1.2422	835.4891	0.0000	0.0000	409.8122	378.3858
403.gcc_5	64	2.3033	1.0445	1.1789	878.3826	0.0000	0.0000	426.0022	393.5172
403.gcc_6	99	7.3799	0.8041	0.8610	1740.6101	0.0000	0.0000	838.8227	799.4547
403.gcc_7	77	3.5674	0.9518	1.1819	1099.5712	0.0000	0.0000	529.5308	505.0290
403.gcc_8	22	0.5658	1.1418	1.0790	233.0485	0.0000	0.0000	111.7023	99.3059
410.bwaves	809	10.0360	1.1913	0.3426	3345.0729	0.0000	0.0000	1574.7275	1596.0280
416.gamess_0	77	0.0169	2.1194	0.0110	32.2100	0.0000	0.0000	1.2683	21.2229
416.gamess_1	18	0.0187	2.3529	0.0084	35.5260	0.0000	0.0000	2.8882	22.9112
416.gamess_2	195	0.0154	2.3910	0.0087	30.5762	0.0000	0.0000	1.4241	20.3074
429.mcf	590	27.4261	0.2481	52.5366	1362.8580	0.0000	0.0000	635.6504	650.5364
433.milc	1122	20.4945	0.4800	1.8919	2728.2331	0.0000	0.0000	1334.5901	1280.1980
434.zeusmp	899	3.6389	0.9010	3.1715	1055.6852	0.0000	0.0000	504.5452	495.8921
435.gromacs	687	0.0271	1.3427	0.0398	33.8353	0.0000	0.0000	17.4772	8.4943
436.cactusADM	1322	1.5511	0.9014	13.4735	559.1393	0.0000	0.0000	262.2531	268.1411
437.leslie3d	680	9.6199	0.9950	0.4794	3669.4780	0.0000	0.0000	1735.7215	1744.0875
444.namd	570	0.0349	1.7600	0.0127	39.9214	0.0000	0.0000	27.0274	6.5739
445.gobmk_0	93	0.3346	1.0757	0.4827	166.8180	0.0000	0.0000	90.7500	68.1316
445.gobmk_1	235	0.2706	1.1204	0.4822	143.4088	0.0000	0.0000	78.4779	56.2898
445.gobmk_2	120	1.5645	1.1344	0.3120	812.7461	0.0000	0.0000	393.3768	370.7252
445.gobmk_3	92	0.4743	1.0807	0.4781	234.1010	0.0000	0.0000	123.0717	100.5747
445.gobmk_4	123	0.2616	1.1632	0.4253	146.0396	0.0000	0.0000	78.6878	57.0293
447.dealII	500	1.9031	1.7181	0.1736	813.9205	0.0000	0.0000	394.5001	372.1443
450.soplex_0	237	5.9758	0.6372	3.2576	1154.3307	0.0000	0.0000	541.2533	515.7061
450.soplex_1	218	16.6967	0.7187	1.9795	3458.7006	0.0000	0.0000	1642.8563	1628.1907
453.povray	241	0.0208	1.7575	0.1413	35.3158	0.0000	0.0000	4.5763	17.0918
456.hmmer_0	202	0.0227	1.9345	0.0345	49.8818	0.0000	0.0000	13.6382	25.0469
456.hmmer_1	435	0.0183	1.8944	0.0106	33.2100	0.0000	0.0000	4.6437	17.0296
458.sjeng	798	0.4072	1.2315	0.4446	177.1199	0.0000	0.0000	76.2752	86.8380
459.GemsFDTD	864	15.6858	0.7511	4.5982	3915.9831	0.0000	0.0000	1911.7363	1887.5251
462.libquantum	974	26.9231	0.7523	0.5731	5622.9233	0.0000	0.0000	2699.4205	2701.8360
464.h264ref_0	91	0.0209	2.4190	0.0702	40.9985	0.0000	0.0000	20.4447	10.7364
464.h264ref_1	71	0.0193	2.0856	0.0412	38.3943	0.0000	0.0000	18.0680	8.1154
464.h264ref_2	628	0.0314	2.1086	0.1937	50.8097	0.0000	0.0000	14.0598	21.5836
465.tonto	170	0.0646	1.8027	0.0408	69.0530	0.0000	0.0000	16.3737	38.4258
470.lbm	787	4.9958	0.7249	0.5379	2931.7464	0.0000	0.0000	1372.8094	1395.3987
471.omnetpp	690	10.1975	0.3712	8.1185	979.5028	0.0000	0.0000	459.8163	485.9691
473.astar_0	232	1.9391	0.6971	12.3336	383.6525	0.0000	0.0000	172.4253	194.8036
473.astar_1	361	0.0491	0.9750	3.6465	35.9841	0.0000	0.0000	19.3710	7.7912
482.sphinx3	783	0.0657	1.7724	0.6105	66.8703	0.0000	0.0000	32.1785	20.4001

Πίνακας 8.5: Μηχάνημα: Sandman, Τοπολογία: 2 (remote far), Σουίτα: Spec
Numactl: interleave

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (4)	IMC (5)
400.perlbench_0	202	0.0083	2.2803	0.1954	135.7897	24.3896	32.1300	53.9833	1.9098
400.perlbench_1	63	0.0254	2.5304	0.0366	178.6344	29.7046	48.6530	38.6914	40.3998
400.perlbench_2	101	0.0525	2.8516	0.0715	216.2881	33.9283	64.3071	43.3008	55.9284
401.bzip2_0	128	0.0981	1.4162	0.0317	177.2262	31.0491	52.5754	39.7394	32.6361
401.bzip2_1	41	0.0322	1.8960	0.0210	151.5367	29.3741	38.3615	35.0071	25.0698
401.bzip2_2	64	0.0161	2.0602	0.0174	141.0613	25.5106	34.1058	32.5850	20.2866
401.bzip2_3	135	0.0833	1.7458	0.0252	177.8168	32.7139	53.3477	9.8962	65.4831
401.bzip2_4	143	0.0605	1.9030	0.0206	166.9571	29.8998	48.4576	7.9764	69.5500
401.bzip2_5	99	0.0875	1.4594	0.0290	173.7924	29.2465	50.2969	8.9314	70.7600
403.gcc_0	23	0.1799	1.4516	1.5420	319.5895	50.1209	95.0970	29.5813	112.8317
403.gcc_1	44	0.2870	1.4322	1.1070	320.7942	48.7243	93.4909	27.3724	110.6673
403.gcc_2	41	0.2023	1.4631	0.7535	333.5690	50.1246	99.3427	29.3937	116.8388
403.gcc_3	28	0.1956	1.6137	0.9474	339.6912	49.4061	100.9800	30.6150	118.2907
403.gcc_4	31	0.3175	1.5627	0.9750	492.9028	67.0263	148.6216	48.8213	164.9616
403.gcc_5	44	0.3939	1.4978	1.0559	578.5491	84.6734	186.4305	64.4864	201.6398
403.gcc_6	51	0.3574	1.5293	0.8591	624.6029	92.3171	204.7106	71.8880	216.7855
403.gcc_7	49	0.3650	1.4802	1.0250	600.7245	88.3776	183.4837	67.7973	200.8814
403.gcc_8	18	0.1079	1.4039	0.8910	201.3313	28.8989	50.3128	10.3850	72.0117
410.bwaves	415	0.3528	2.2847	0.1440	5844.8652	866.2014	1942.1360	906.8701	1901.0090
416.gamess_0	61	0.0046	2.6879	0.0117	131.6638	22.0707	30.5611	1.3110	54.5762
416.gamess_1	14	0.0044	2.9284	0.0112	132.6131	23.5657	30.8993	26.4864	30.5679
416.gamess_2	158	0.0041	2.9215	0.0108	131.0494	22.3856	30.9422	24.3197	31.7726
429.mcf	290	11.7832	0.4902	9.8531	2197.8522	410.2149	650.7406	402.4270	656.9338
433.milc	508	9.3554	1.0174	1.8253	4955.7937	42.3465	2233.8024	42.4934	2235.9366
434.zeusmp	440	1.3993	1.7957	0.2518	1712.7238	288.3124	549.7457	263.4071	571.6726
435.gromacs	406	0.0053	2.2206	0.0166	82.1347	22.3977	30.8294	18.6122	30.8178
436.cactusADM	700	0.9471	1.7266	0.0465	938.7463	22.0586	437.4478	18.8846	436.1746
437.leslie3d	316	0.3972	2.0855	0.2190	2340.7679	313.8766	775.7345	374.8020	721.9607
444.namd	459	0.0064	2.1472	0.0134	134.5564	1.3652	52.9234	24.6973	22.5948
445.gobmk_0	85	0.0117	1.1699	0.3828	132.6455	1.3296	37.5880	24.9873	31.1061
445.gobmk_1	215	0.0107	1.2236	0.3935	133.6782	1.2760	42.3400	24.0058	31.2526
445.gobmk_2	105	0.0105	1.3021	0.2778	144.8063	1.2170	34.2286	23.9979	31.1794
445.gobmk_3	84	0.0127	1.1866	0.4062	140.1518	1.5143	44.6410	23.3636	31.0988
445.gobmk_4	114	0.0117	1.2488	0.3871	137.5083	1.1710	45.6375	23.0982	31.0886
447.dealII	377	0.1751	2.2608	0.2029	886.7911	114.1614	293.1966	137.6050	275.9834
450.soplex_0	131	0.0513	1.1364	1.6954	248.8307	27.6585	43.9247	25.2874	43.4940
450.soplex_1	114	1.7461	1.3581	0.6695	5234.9788	25.9533	2398.3917	22.6872	2400.4730
453.povray	184	0.0055	2.2829	0.1071	130.6429	52.1391	0.6465	17.3732	30.8067
456.hmmer_0	153	0.0052	2.4798	0.0118	132.5401	51.2402	0.9893	18.3688	31.1566
456.hmmer_1	329	0.0046	2.4808	0.0105	130.6634	52.6420	0.6489	7.8141	30.8532
458.sjeng	649	0.2155	1.4996	0.4226	191.4231	65.1257	31.1120	14.2811	80.3907
459.GemsFDTD	404	2.8882	1.5944	1.7937	6876.1369	18.9837	3313.1541	17.5806	3339.0680
462.libquantum	262	0.0574	2.6249	0.0740	917.0897	1.4458	87.7970	1.3847	96.3628
464.h264ref_0	81	0.0047	2.7083	0.0170	132.0535	15.4147	30.8586	31.2587	22.5660
464.h264ref_1	65	0.0059	2.2486	0.0139	131.5053	6.0108	30.9597	31.3322	21.3252
464.h264ref_2	567	0.0077	2.2943	0.0205	136.3642	21.1323	32.2319	44.6751	9.9262
465.tonto	131	0.0053	2.3092	0.0243	131.3532	17.1830	31.0388	12.5583	30.9243
470.lbm	454	1.7148	1.2102	0.3136	4534.0411	52.1790	2045.1053	57.1673	2037.7836
471.omnetpp	244	0.2016	1.0310	6.7186	192.1266	25.4167	54.2176	23.0183	56.7648
473.astar_0	167	0.0589	0.9861	6.7801	165.1592	0.6268	64.9353	0.5689	63.5239
473.astar_1	323	0.0113	1.0970	1.5847	131.7394	21.1247	31.2881	15.6874	31.1998
482.sphinx3	657	0.0058	2.0962	0.3837	131.1211	41.8838	13.7340	16.4781	30.9898

Πίνακας 8.6: Μηχάνημα: Broady, Τοπολογία: 0 (local), Σουίτα: Spec

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (4)	IMC (5)
400.perlbench_0	203	0.0075	2.2850	0.1882	136.2185	46.3023	0.8445	20.6164	30.9263
400.perlbench_1	65	0.0256	2.4386	0.0255	198.7261	86.9028	1.3136	50.9238	31.2646
400.perlbench_2	102	0.0500	2.8229	0.0722	256.5883	116.5320	1.8384	84.2691	31.9722
401.bzip2_0	129	0.0988	1.3965	0.0322	190.9302	94.8487	1.3726	62.9862	31.7434
401.bzip2_1	42	0.0325	1.8670	0.0217	158.5551	64.5881	1.0788	30.4431	30.5793
401.bzip2_2	64	0.0167	2.0565	0.0173	144.2490	40.0872	0.9313	26.9323	30.8888
401.bzip2_3	137	0.0841	1.7244	0.0249	193.9323	77.9724	1.2726	63.4499	31.5330
401.bzip2_4	145	0.0611	1.8796	0.0207	178.4584	82.9042	1.2317	61.9747	31.2899
401.bzip2_5	100	0.0891	1.4485	0.0306	188.1588	92.3507	1.4042	64.2239	31.6407
403.gcc_0	24	0.1840	1.4156	1.4344	396.9959	186.1175	3.1408	143.9567	33.1046
403.gcc_1	47	0.2871	1.3588	1.1105	359.3000	163.1924	2.8215	111.9570	32.1594
403.gcc_2	42	0.2070	1.3848	0.7501	421.8315	186.0060	2.7429	139.7119	32.3671
403.gcc_3	28	0.1949	1.5834	0.9449	431.5627	194.9696	2.9911	155.2403	32.0090
403.gcc_4	33	0.3189	1.4832	0.9877	614.9960	269.8421	3.4300	241.1827	32.9858
403.gcc_5	47	0.4037	1.4101	1.0812	711.2384	318.9500	3.8806	289.9066	33.6502
403.gcc_6	54	0.3791	1.4631	0.8423	745.2024	342.4328	3.9400	295.0315	34.0863
403.gcc_7	52	0.3728	1.4113	1.0363	733.5057	336.4100	3.4139	283.9471	33.3702
403.gcc_8	18	0.1083	1.3831	0.8708	223.7663	61.3261	31.2450	43.6206	31.8806
410.bwaves	450	0.3947	2.0905	0.1521	5912.1655	2822.5520	42.4235	2785.5398	41.8657
416.gamess_0	61	0.0046	2.6733	0.0118	131.3310	21.6318	30.6944	15.6097	31.0555
416.gamess_1	14	0.0047	2.9117	0.0109	132.7200	13.9471	30.7879	21.4371	30.4157
416.gamess_2	158	0.0041	2.9216	0.0110	131.2265	23.1675	30.9267	14.9218	30.7054
429.mcf	350	11.7059	0.4115	9.4432	2099.0759	959.7157	47.5241	963.2385	37.9687
433.milc	751	9.2399	0.7036	1.8763	3992.9525	1861.0675	34.8866	1865.4159	34.7515
434.zeusmp	523	1.4172	1.5167	0.2606	94.8759	863.3879	59.9328	861.4317	36.6979
435.gromacs	404	0.0049	2.2563	0.0161	82.1100	1.0805	47.2953	13.0962	30.8392
436.cactusADM	790	0.9452	1.5212	0.0570	1117.7942	499.7634	8.8589	515.5174	4.6925
437.leslie3d	334	0.4734	1.9604	0.2194	2828.3341	1311.7900	64.7466	1376.5151	32.4979
444.namd	459	0.0069	2.1562	0.0136	134.9536	9.2215	47.0415	32.8243	21.7382
445.gobmk_0	85	0.0130	1.1669	0.3937	140.3022	23.3698	30.9609	31.7438	22.5501
445.gobmk_1	215	0.0118	1.2207	0.4015	140.1582	25.4166	30.9364	31.6484	21.1970
445.gobmk_2	106	0.0139	1.2910	0.2808	175.1373	23.4483	30.7453	31.4819	16.7919
445.gobmk_3	84	0.0132	1.1849	0.4063	149.9087	23.4438	30.7936	31.6060	13.5356
445.gobmk_4	114	0.0129	1.2455	0.3960	142.3438	23.9915	30.8814	31.5134	9.6299
447.dealII	392	0.1919	2.1713	0.2099	964.9421	415.3410	33.2434	422.4614	19.5463
450.soplex_0	132	0.0518	1.1312	1.6949	312.7586	51.3442	31.4730	57.6867	22.4093
450.soplex_1	135	1.9775	1.1206	0.6877	4874.8146	2218.0516	37.3382	2210.9429	37.2203
453.povray	184	0.0052	2.2730	0.1117	130.9208	24.6769	30.7387	12.4024	41.4776
456.hmmer_0	152	0.0051	2.5397	0.0125	131.6677	25.3512	31.0982	1.3832	51.8499
456.hmmer_1	329	0.0046	2.4818	0.0103	130.6034	23.9825	30.8739	1.0538	51.7452
458.sjeng	667	0.2193	1.4570	0.4235	88.5368	99.7975	16.2467	72.3490	43.8987
459.GemsFDTD	515	2.9751	1.2504	1.7677	97.8850	3376.7007	36.9170	3377.0321	36.5648
462.libquantum	275	0.1404	2.4845	0.0655	6450.3407	110.0640	31.0232	152.0089	19.0696
464.h264ref_0	82	0.0054	2.6818	0.0168	131.3232	51.9799	0.7394	21.1801	30.7264
464.h264ref_1	65	0.0060	2.2807	0.0136	131.3417	51.7152	0.7306	22.9964	30.8486
464.h264ref_2	574	0.0076	2.2765	0.0205	140.6105	53.7753	0.8343	24.2317	31.0025
465.tonto	131	0.0053	2.3117	0.0243	131.9157	51.1966	0.7397	23.1073	30.8304
470.lbm	565	1.8800	0.9823	0.3234	4750.0096	2131.5502	35.9582	2127.8800	35.7934
471.omnetpp	250	0.2035	1.0076	6.6531	242.4858	52.5254	32.9635	52.8893	32.9753
473.astar_0	169	0.0586	0.9683	6.8600	190.5039	24.2861	46.1079	24.1436	42.8496
473.astar_1	323	0.0118	1.0976	1.5873	132.7819	13.5802	30.9234	16.3531	30.8251
482.sphinx3	655	0.0060	2.0972	0.3818	130.9597	15.5212	30.9506	12.9200	30.9025

Πίνακας 8.7: Μηχάνημα: Broady, Τοπολογία: 1 (remote), Σουίτα: Spec
Numactl: membind

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (4)	IMC (5)
400.perlbench_0	201	0.0074	2.2968	0.1901	134.4216	55.7288	1.2712	25.4917	31.4513
400.perlbench_1	66	0.0248	2.4215	0.0270	110.2171	72.2827	9.9633	45.1200	39.7259
400.perlbench_2	102	0.0586	2.8192	0.0719	127.5000	91.6146	19.5384	63.3310	49.5569
401.bzip2_0	129	0.1064	1.3742	0.0322	104.6081	77.7469	12.7298	48.5230	42.9011
401.bzip2_1	42	0.0326	1.8725	0.0226	92.5027	59.1893	4.6163	30.4407	34.4431
401.bzip2_2	64	0.0167	2.0558	0.0173	86.3166	56.2170	2.6875	28.6159	32.5558
401.bzip2_3	136	0.0896	1.7253	0.0252	106.2117	79.3526	13.1027	48.2478	43.1275
401.bzip2_4	145	0.0653	1.8756	0.0207	100.3143	74.8601	10.3676	45.9269	41.9842
401.bzip2_5	100	0.0949	1.4467	0.0313	103.7153	73.1443	13.5241	45.7969	41.5289
403.gcc_0	24	0.1816	1.4009	1.4711	175.1719	102.0171	62.5942	108.8591	65.1352
403.gcc_1	46	0.2890	1.3888	1.0725	178.6126	86.3711	62.2528	89.5049	62.7798
403.gcc_2	42	0.2137	1.4126	0.7389	185.7826	103.4778	65.5717	103.2690	66.4951
403.gcc_3	28	0.1973	1.5820	0.9541	188.5285	105.4575	67.7318	106.4679	67.9496
403.gcc_4	32	0.3356	1.5067	0.9826	261.2093	153.9744	92.4441	156.0509	91.5572
403.gcc_5	46	0.4157	1.4373	1.1037	305.6748	184.5265	107.5739	186.5202	106.2878
403.gcc_6	53	0.3976	1.4915	0.8075	331.0374	201.5000	119.5698	192.7930	113.4794
403.gcc_7	51	0.3810	1.4283	0.9965	315.3867	194.6020	111.8642	191.7357	109.9082
403.gcc_8	18	0.1054	1.4173	0.8815	145.3906	43.7733	41.2133	44.5056	40.4544
410.bwaves	433	0.3921	2.1654	0.1446	5904.8717	1805.9371	1030.4841	1794.1935	1027.6288
416.gamess_0	61	0.0044	2.6832	0.0113	131.5493	23.1957	30.6264	15.4025	30.6697
416.gamess_1	14	0.0050	2.8806	0.0117	132.6775	21.5200	30.7757	17.3000	30.4521
416.gamess_2	158	0.0039	2.9226	0.0110	130.6513	21.7175	30.8592	15.6808	30.7736
429.mcf	326	11.7006	0.4350	9.6578	2147.1423	652.5350	374.1655	647.6672	372.5217
433.milc	628	9.3930	0.8276	1.8462	4350.1501	1114.7111	948.8909	1113.2227	951.1991
434.zeusmp	490	1.4000	1.6160	0.2538	1822.2074	546.5556	323.9760	568.0419	300.9953
435.gromacs	402	0.0049	2.2706	0.0162	130.7734	1.0537	52.1198	20.5696	30.7942
436.cactusADM	749	0.9804	1.6114	0.0559	1048.4798	296.7336	203.7835	316.6603	180.4439
437.leslie3d	324	0.4504	2.0339	0.2205	1117.2475	821.5344	441.8144	859.5628	409.5077
444.namd	459	0.0066	2.1553	0.0149	83.2898	2.0179	44.7839	35.9220	17.7544
445.gobmk_0	85	0.0119	1.1652	0.4004	82.4112	1.3415	43.7884	31.3642	19.0967
445.gobmk_1	215	0.0112	1.2218	0.4083	82.1944	1.2381	45.0147	31.5322	20.3399
445.gobmk_2	105	0.0114	1.2937	0.2793	82.4160	1.9267	47.3865	31.5889	20.9975
445.gobmk_3	84	0.0133	1.1874	0.4082	129.5605	1.2048	46.9401	31.7681	19.3252
445.gobmk_4	114	0.0104	1.2475	0.3950	131.0571	1.3546	46.4086	31.4777	20.7732
447.dealII	387	0.1918	2.1914	0.2106	923.0257	248.8353	172.8355	271.8257	156.9331
450.soplex_0	132	0.0507	1.1363	1.7003	183.4621	31.3924	37.9108	47.3322	27.9820
450.soplex_1	123	1.9378	1.2444	0.6865	5145.7180	1540.2726	883.8911	1561.0853	865.1695
453.povray	185	0.0032	2.2204	0.1118	130.7308	12.9842	30.4670	14.7384	30.5115
456.hmmer_0	152	0.0053	2.5392	0.0126	131.4625	21.8580	31.1268	31.5155	20.5223
456.hmmer_1	329	0.0045	2.4588	0.0110	130.2064	21.8859	30.8451	3.0698	48.5911
458.sjeng	660	0.2231	1.4736	0.4241	229.3854	77.6429	28.8431	37.3818	68.4486
459.GemsFDTD	464	2.9219	1.3800	1.7858	7162.3776	1928.8173	1432.3974	1980.7498	1457.2720
462.libquantum	266	0.1328	2.6317	0.0698	854.9063	67.0777	68.5924	61.3159	67.4469
464.h264ref_0	79	0.0049	2.7598	0.0170	130.7087	52.9027	0.8088	24.1331	30.9521
464.h264ref_1	65	0.0055	2.2849	0.0140	131.1319	52.4648	0.7198	23.0970	30.7820
464.h264ref_2	573	0.0074	2.2847	0.0199	83.4398	53.9146	1.3766	24.4878	31.6062
465.tonto	131	0.0051	2.2992	0.0239	82.1623	52.0113	0.7535	24.2395	31.0392
470.lbm	500	1.7835	1.0993	0.3209	2053.1060	1244.0710	949.7752	1242.6763	949.5139
471.omnetpp	248	0.2084	1.0136	6.7008	153.8534	28.7384	43.7675	47.4260	44.1481
473.astar_0	168	0.0665	1.0081	6.8056	170.8366	31.2717	37.0179	38.6444	37.6832
473.astar_1	323	0.0114	1.0947	1.5792	132.2219	13.8805	31.0547	25.2701	31.1128
482.sphinx3	656	0.0059	2.0967	0.3844	131.0874	24.1781	30.9566	12.3168	30.8760

Πίνακας 8.8: Μηχάνημα: Broady, Τοπολογία: 1 (remote), Σουίτα: Spec
Numactl: interleave

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (2)	IMC (3)
blackscholes_0	274	0.1233	1.3152	0.0181	182.5693	0.0000	0.0000	87.0538	67.6302
bodytrack_0	230	0.0509	1.5738	0.0247	41.2196	0.0000	0.0000	36.3769	8.5954
canneal_0	185	18.4539	0.3097	17.8401	915.9494	0.0000	0.0000	439.7383	439.7000
dedup_0	48	0.4349	0.9834	0.1676	146.3476	0.0000	0.0000	86.4170	119.0618
facesim_0	531	1.3767	1.8853	0.1326	1317.5911	0.0000	0.0000	611.7619	622.2657
ferret_0	583	0.7720	1.3365	1.8499	198.9131	0.0000	0.0000	96.3512	99.9538
fluidanimate_0	479	0.9114	1.5560	0.1193	572.4374	0.0000	0.0000	285.2364	260.3945
freqmine_0	667	0.1146	1.5601	0.6861	92.9418	0.0000	0.0000	56.7627	26.9271
rtview_0	292	0.3736	1.6951	0.9526	240.9583	0.0000	0.0000	104.8586	114.5331
swaptions_0	376	0.0209	1.7387	0.0083	32.4456	0.0000	0.0000	7.6553	17.0516
streamcluster_0	663	15.0387	0.7195	1.8338	1872.3226	0.0000	0.0000	891.7827	894.4935
vips_0	161	0.0712	2.1178	0.0519	105.4202	0.0000	0.0000	41.9352	40.6793

Πίνακας 8.9: Μηχάνημα: Sandman, Τοπολογία: 0 (local), Σουίτα: Parsec

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (2)	IMC (3)
blackscholes_0	282	0.1502	1.3182	0.0204	178.7111	0.0000	0.0000	77.4010	98.1229
bodytrack_0	217	0.0490	1.6064	0.0236	45.2427	0.0000	0.0000	37.7281	8.8657
canneal_0	404	17.3478	0.1543	15.6092	461.6649	0.0000	0.0000	221.5033	213.2469
dedup_0	50	0.4843	1.3375	0.1688	198.1705	0.0000	0.0000	96.1418	125.1594
facesim_0	678	2.7806	1.4750	0.1312	1101.3529	0.0000	0.0000	543.7070	532.6920
ferret_0	678	0.7870	1.1434	1.8311	184.4959	0.0000	0.0000	91.3561	80.2838
fluidanimate_0	548	1.0419	1.3583	0.1207	601.2348	0.0000	0.0000	290.6766	280.5626
freqmine_0	676	0.1194	1.5528	0.6888	99.4946	0.0000	0.0000	46.5128	49.1723
rtview_0	328	0.4030	1.5124	0.9234	232.2490	0.0000	0.0000	98.6008	118.2670
swaptions_0	377	0.0210	1.7352	0.0081	33.5151	0.0000	0.0000	1.6356	21.4993
streamcluster_0	1778	14.5906	0.2771	1.7914	720.7904	0.0000	0.0000	343.4862	339.1735
vips_0	163	0.0719	2.1046	0.0550	94.6063	0.0000	0.0000	52.6280	44.0108

Πίνακας 8.10: Μηχάνημα: Sandman, Τοπολογία: 1 (remote near), Σουίτα: Parsec
Numactl: membind

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (2)	IMC (3)
blackscholes_0	280	0.1083	1.2613	0.0188	187.4789	0.0000	0.0000	94.6446	76.2285
bodytrack_0	218	0.0495	1.6000	0.0239	42.5414	0.0000	0.0000	37.0656	7.5657
canneal_0	418	17.3237	0.1497	14.9647	441.9525	0.0000	0.0000	216.2172	206.5869
dedup_0	50	0.4900	1.3229	0.2535	187.4467	0.0000	0.0000	108.6530	112.4488
facesim_0	698	2.7013	1.4306	0.1315	1116.7634	0.0000	0.0000	531.9457	521.5343
ferret_0	680	0.7887	1.1360	1.8278	184.3983	0.0000	0.0000	91.7430	81.4000
fluidanimate_0	562	1.0349	1.3223	0.1214	585.6637	0.0000	0.0000	271.2505	289.1270
freqmine_0	678	0.1267	1.5585	0.6880	101.5516	0.0000	0.0000	52.7816	42.3866
rtview_0	330	0.4066	1.5012	0.9303	228.2916	0.0000	0.0000	98.5623	118.6618
swaptions_0	375	0.0207	1.7408	0.0082	32.9515	0.0000	0.0000	1.5373	21.2510
streamcluster_0	1809	14.5775	0.2725	1.7909	706.5734	0.0000	0.0000	348.4065	324.5754
vips_0	163	0.0721	2.1153	0.0568	93.6398	0.0000	0.0000	53.6033	43.7887

Πίνακας 8.11: Μηχάνημα: Sandman, Τοπολογία: 2 (remote far), Σουίτα: Parsec
Numactl: membind

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (2)	IMC (3)
blackscholes_0	275	0.1102	1.3334	0.0186	178.6555	0.0000	0.0000	78.5268	86.2950
bodytrack_0	217	0.0497	1.6017	0.0234	42.2246	0.0000	0.0000	27.4759	17.2237
canneal_0	298	18.1621	0.1978	16.6768	585.5589	0.0000	0.0000	277.8933	289.4528
dedup_0	49	0.4814	1.3268	0.2413	195.5682	0.0000	0.0000	108.0277	102.7606
facesim_0	608	2.1771	1.6354	0.1384	1211.2101	0.0000	0.0000	570.9275	581.2191
ferret_0	630	0.7828	1.2248	1.8388	190.0878	0.0000	0.0000	101.4490	78.4322
fluidanimate_0	513	1.0137	1.4421	0.1340	591.4623	0.0000	0.0000	291.3933	266.9298
freqmine_0	671	0.1197	1.5684	0.6871	102.7873	0.0000	0.0000	35.0801	55.1904
rtview_0	312	0.3736	1.5587	0.9495	237.3928	0.0000	0.0000	104.2038	113.4992
swaptions_0	376	0.0211	1.7372	0.0084	33.7820	0.0000	0.0000	7.0688	17.5409
streamcluster_0	1219	14.7652	0.3978	1.8451	1010.7197	0.0000	0.0000	480.7751	500.8017
vips_0	161	0.0714	2.1155	0.0559	91.5689	0.0000	0.0000	43.9464	43.9220

Πίνακας 8.12: Μηχάνημα: Sandman, Τοπολογία: 1 (remote near), Σουίτα: Parsec
Numactl: interleave

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (2)	IMC (3)
blackscholes_0	277	0.1151	1.2628	0.0191	178.7319	0.0000	0.0000	82.7571	82.0743
bodytrack_0	219	0.0489	1.5388	0.0246	46.7041	0.0000	0.0000	28.7854	15.1529
canneal_0	307	18.0547	0.1927	16.3132	570.4220	0.0000	0.0000	274.2383	284.9023
dedup_0	52	0.4874	1.2622	0.2427	194.4098	0.0000	0.0000	105.7586	95.9346
facesim_0	612	2.1423	1.6237	0.1351	1197.8201	0.0000	0.0000	580.3223	561.4023
ferret_0	630	0.7829	1.2232	1.8405	191.9658	0.0000	0.0000	100.7650	78.0932
fluidanimate_0	520	1.0092	1.4212	0.1307	586.7887	0.0000	0.0000	290.2521	264.3117
freqmine_0	681	0.1229	1.5399	0.7001	103.0981	0.0000	0.0000	60.8058	27.8773
rtview_0	312	0.3761	1.5486	0.9470	237.2506	0.0000	0.0000	106.1320	114.0877
swaptions_0	379	0.0212	1.7249	0.0084	34.7813	0.0000	0.0000	6.8166	17.0642
streamcluster_0	1230	14.7503	0.3944	1.8153	997.8060	0.0000	0.0000	475.3860	496.9892
vips_0	163	0.0718	2.0781	0.0585	94.6793	0.0000	0.0000	44.5092	44.1317

Πίνακας 8.13: Μηχάνημα: Sandman, Τοπολογία: 2 (remote far), Σουίτα: Parsec
Numactl: interleave

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (4)	IMC (5)
blackscholes_0	211	0.0471	1.7409	0.0182	302.8859	48.8436	79.2980	61.7670	77.2775
bodytrack_0	174	0.0142	2.1888	0.0191	135.4918	0.4584	50.5919	0.4403	49.6560
canneal_0	132	10.6394	0.4371	15.4485	937.2699	198.3989	250.5015	204.7156	248.8801
dedup_0	43	0.1351	1.4820	0.1999	249.7820	3.7629	97.7917	3.7983	98.5081
facesim_0	419	0.2552	2.3661	0.0335	372.4565	68.7165	102.5819	72.2515	97.3993
ferret_0	477	0.0804	1.6190	2.4489	169.4788	28.5386	42.2469	30.9643	40.8303
fluidanimate_0	398	0.4158	1.8534	0.0974	649.3264	183.1692	175.2053	158.1336	201.6304
freqmine_0	579	0.0123	1.8532	0.6485	114.7401	0.4577	37.9267	30.6041	8.2710
rtview_0	231	0.1813	2.1336	0.0748	303.0144	79.2883	86.4942	100.7725	64.4577
swaptions_0	299	0.0051	2.1605	0.0208	83.2857	1.0142	53.1449	50.6440	0.5430
streamcluster_0	546	3.2274	0.8471	1.7649	1864.1489	12.1814	837.7236	11.1187	836.3310
vips_0	125	0.0131	2.8120	0.0198	172.8817	46.3563	17.8292	47.4961	17.8395

Πίνακας 8.14: Μηχάνημα: Broady, Τοπολογία: 0 (local), Σουίτα: Parsec

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (4)	IMC (5)
blackscholes_0	213	0.0628	1.7219	0.0182	323.2740	110.6256	33.2858	106.5128	43.4537
bodytrack_0	169	0.0143	2.1288	0.0196	138.4947	2.7159	48.7179	2.6729	45.8692
canneal_0	178	10.4569	0.3322	15.0981	796.9917	336.9600	39.2082	328.6006	34.7268
dedup_0	44	0.1236	1.5929	0.2037	302.6217	106.9302	42.7955	96.9759	38.8186
facesim_0	443	0.2589	2.2294	0.0343	435.4156	143.1755	44.4192	166.4553	32.4700
ferret_0	488	0.0881	1.5607	2.4580	163.4561	48.4738	21.9601	40.2070	32.1906
fluidanimate_0	417	0.4323	1.7602	0.1027	843.0846	379.8194	24.3163	370.4419	34.4182
freqmine_0	581	0.0124	1.8195	0.6438	83.0034	31.0131	19.4700	9.5801	43.6654
rtview_0	241	0.1912	2.0528	0.0759	88.7962	151.8025	21.6090	143.1722	32.9049
swaptions_0	298	0.0056	2.1676	0.0208	83.2362	31.2575	21.4727	23.4907	30.8404
streamcluster_0	854	3.2038	0.5494	1.7635	1321.2113	575.3156	35.3630	566.6384	34.8785
vips_0	121	0.0143	2.8564	0.0231	180.2845	72.2998	6.9854	73.8737	7.6407

Πίνακας 8.15: Μηχάνημα: Broady, Τοπολογία: 1 (remote), Σουίτα: Parsec
Numactl: membind

Benchmark	TT	MPKI	IPC	TLB	BW	IMC (0)	IMC (1)	IMC (4)	IMC (5)
blackscholes_0	211	0.0542	1.7346	0.0184	309.8745	80.3717	58.1711	87.8073	57.7065
bodytrack_0	165	0.0142	2.1815	0.0190	139.3714	36.5596	16.1659	31.5688	24.7715
canneal_0	154	10.5428	0.3730	15.9671	848.8827	209.3774	183.2177	208.4070	178.4734
dedup_0	42	0.1406	1.4829	0.2033	264.0851	54.1007	83.4576	54.2612	84.4321
facesim_0	438	0.2652	2.2429	0.0342	397.3244	114.4566	64.9584	105.7320	82.0010
ferret_0	484	0.0841	1.5891	2.4427	160.2479	25.1975	36.9494	13.4004	58.2466
fluidanimate_0	410	0.4398	1.7881	0.0993	707.3823	266.1590	119.1410	239.3200	143.2233
freqmine_0	585	0.0227	1.8448	0.6441	147.6157	31.1590	34.1441	9.0618	57.3061
rtview_0	234	0.1907	2.1092	0.0762	195.6686	106.6034	64.9152	96.4964	76.4497
swaptions_0	300	0.0050	2.1632	0.0215	83.3083	23.2235	30.8855	31.3905	22.6655
streamcluster_0	710	3.2890	0.6630	1.8033	858.1137	325.5436	373.0473	330.9166	367.8041
vips_0	123	0.0133	2.7505	0.0201	176.2876	29.4966	40.4746	31.4037	41.1087

Πίνακας 8.16: Μηχάνημα: Broady, Τοπολογία: 1 (remote), Σουίτα: Parsec
Numactl: interleave

Κεφάλαιο 9

Παράτημα Β

Εδώ βλέπουμε τον κώδικα του παραγωγού πρώτης γενιάς, γραμμένο στην γλώσσα προγραμματισμού Python.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
import itertools

def function_generator():
    def make_function(x, n, op):
        def Re_num(s):
            ret = ''; j = 0; k = 1
            for i in s:
                if   i == 'a': ret += i+str(j); j += 1
                elif i == 'x': ret += i;        k += 1
                else         : ret += i
            return ret+' + b'

        prod = []
        for i in x:
            prod.append(list(itertools.product([i], n)))
        if op == 2:
            prod[0] = list(itertools.product(prod[0], n))
        prod = list(itertools.product(*prod)); ret = []
        for i in prod:
            if   op == 0:
                j = list(i[0])
                t1 = 'a*'+j[0]+'**('+j[1]+')'
                t2 =      j[0]+'**('+j[1]+')'
                length = len(i)
            if   op == 1:
                j = list(i[0])
                t1 = 'a*log('+j[0]+'+1)**('+j[1]+')'
                t2 =          j[0]+   '**('+j[1]+')'
                length = len(i)
            elif op == 2:
                j = list(i[0][0])+[i[0][1]]
```

```
                t1  = 'a*log('+j[0]+'+1)**('+j[1]+') + a*'
                t1 += j[0]+'**('+j[2]+')'
                t2  =          j[0]+   '**('+j[1]+')'
                length = len(i)+1
            t3 = []
            for j in i[1:]: t3.append(j[0]+'**('+j[1]+')')

            main = (' + a*'.join([t1]+t3))
            ret.append(Re_num(main))

            tt = [t2]+t3
            temp = []
            for j in range(2, len(tt)+1):
                for subset in itertools.combinations(tt, j):
                    temp.append('a*('+(')*('.join(subset))+')')
            for j in range(1, len(temp)+1):
                for subset in itertools.combinations(temp, j):
                    if length+len(subset) >= 8: continue
                    a = Re_num(main+' + '+(' + '.join(subset)))
                    if a != '': ret.append(a)
        return ret

    x = ['x1', 'x2', 'x3', 'x4']
    n = list(np.arange(-2, 2+0.1, 0.5)); n.remove(0)
    n = [str(i) for i in n]
    ret = []
    for i in range(1, len(x)+1):
        for subset in itertools.combinations(x, i):
            ret += make_function(list(subset), n, 0)
            for j in subset:
                t = list(subset); t.remove(j); t = [j]+t
                ret += make_function(t, n, 1)
                ret += make_function(t, n, 2)
    return ret
```

Βιβλιογραφία

[1] “numactl - Control NUMA policy for processes or shared memory.” https://linux.die.net/man/8/numactl, [Online].

[2] “perf: Linux profiling with performance counters.” https://perf.wiki.kernel.org/index.php/MainPage, [Online].

[3] “migrate_pages - move all pages in a process to another set of nodes” http://man7.org/linux/man-pages/man2/migrate_pages.2.html, [Online].

[4] “move_pages (2) - Linux Man Pages” https://www.systutorials.com/docs/linux/man/2-move_pages/, [Online].

[5] “Intel Performance Counter Monitor - A better way to measure CPU utilization” https://software.intel.com/en-us/articles/intel-performance-counter-monitor, [Online].

[6] “numa_maps(5) - Linux man page” https://linux.die.net/man/5/numa_maps, [Online].

[7] “Cross-validation (statistics)” https://en.wikipedia.org/wiki/Cross-validation_(statistics), [Online].

[8] “scipy.optimize.curve_fit” https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.curve_fit.html, [Online].

[9] “Optimization and root finding (scipy.optimize)” https://docs.scipy.org/doc/scipy/reference/optimize.html#optimization-and-root-finding-scipy-optimize, [Online].

[10] “sklearn.metrics.r2_score” https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html, [Online].

[11] “sklearn.metrics.mean_absolute_error” https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html#sklearn.metrics.mean_absolute_error, [Online].

[12] "sklearn.metrics.explained_variance_score"
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.explained_variance_score.html#sklearn.metrics.explained_variance_score, [Online].

[13] "API Reference"
https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics, [Online].

[14] "sys/time.h - time types"
http://pubs.opengroup.org/onlinepubs/7908799/xsh/systime.h.html, [Online].

[15] "waitpid(3) - Linux man page"
https://linux.die.net/man/3/waitpid, [Online].

[16] B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. De Supinski, and M. Schulz, "A regression-based approach to scalability prediction," in Proceedings of the 22nd annual International Conference on Super-computing, 2008, pp. 368–377.

[17] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, January 2011.

[18] A. Calotoiu, D. Beckinsale, C. W. Earl, T. Hoefler, I. Karlin, M. Schulz, and F. Wolf, "Fast multi-parameter performance modeling," in 2016 IEEE International Conference on Cluster Computing (CLUSTER), 2016, pp. 172–181.

[19] G. I. Goumas, K. Nikas, E. B. Lakew, C. Kotselidis, A. Attwood, E. Elmroth, M. Flouris, N. Foutris, J. Goodacre, D. Grohmann, V. Karakostas, P. Koutsourakis, M. L. Kersten, M. Luján, E. Rustad, J. Thomson, L. Tomás, A. Vesterkjaer, J. Webber, Y. Zhang, and N. Koziris, "ACTiCLOUD: Enabling the Next Generation of Cloud Applications," in 37th IEEE International Conference on Distributed Computing Systems (ICDCS), 2017, pp. 1836–1845.

[20] J. L. Henning, "SPEC CPU2006 Benchmark Descriptions," SIGARCH Comput. Archit. News, vol. 34, no. 4, pp. 1–17, Sep. 2006.

[21] C. Lameter, "NUMA (Non-Uniform Memory Access): An Overview," Queue, vol. 11, no. 7, pp. 40:40–40:51, Jul. 2013.

[22] H. Luo, J. Brock, P. Li, C. Ding, and C. Ye, "Compositional model of coherence and NUMA effects for optimizing thread and data placement," in 2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2016, pp. 151–152.

[23] Z. Majo and T. R. Gross, "Memory System Performance in a NUMA Multicore Multiprocessor," in Proceedings of the 4th Annual International Conference on Systems and Storage, ser. SYSTOR '11. ACM, 2011, pp. 12:1–12:10.

[24] P. S. McCormick, R. K. Braithwaite, and W.-c. Feng, "Empirical Memory-Access Cost Models in Multicore NUMA Architectures," Los Alamos National Lab. (LANL), Los Alamos, NM (United States), , 2011.

[25] N. Papadopoulou, G. I. Goumas, and N. Koziris, “A Machine-Learning Approach for Communication Prediction of Large-Scale Applications,” in 2015 IEEE International Conference on Cluster Computing (CLUSTER), 2015, pp. 120–123.

[26] S. Shudler, A. Calotoiu, T. Hoefler, A. Strube, and F. Wolf, “Exascaling your library: Will your implementation meet your expectations?” in Proceedings of the 29th ACM on International Conference on Supercomputing, 2015, pp. 165–175.

[27] C. Su, D. Li, D. S. Nikolopoulos, K. W. Cameron, B. R. de Supinski, and E. A. León, “Model-based, memory-centric performance and power optimization on NUMA multiprocessors,” in 2012 IEEE International Symposium on Workload Characterization (IISWC), 2012, pp. 164–173.

[28] W. Wang, J. W. Davidson, and M. L. Soffa, “Predicting the memory bandwidth and optimal core allocations for multi-threaded applications on large-scale NUMA machines,” in 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), 2016, pp. 419–431.