ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μελέτη και Υλοποίηση Αλγορίθμων Καταμερισμού Διαμοιραζόμενης Μνήμης σε Πολυπύρηνες Αρχιτεκτονικές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Περικλής Β. Βασιλείου

Επιβλέπων : Νεκτάριος Κοζύρης
Αν. Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Μελέτη και Υλοποίηση Αλγορίθμων Καταμερισμού Διαμοιραζόμενης Μνήμης σε Πολυπύρηνες Αρχιτεκτονικές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Περικλής Β. Βασιλείου

Επιβλέπων : Νεκτάριος Κοζύρης
Αν. Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14η Ιουλίου 2009.

.....	.....	.....
Νεκτάριος Κοζύρης Αν. Καθηγητής ΕΜΠ	Δημήτριος Σούντρης Επ. Καθηγητής ΕΜΠ	Γιώργος Οικονομάκος Λέκτορας ΕΜΠ

Αθήνα, Ιούλιος 2009

.....
Περικλής Β. Βασιλείου

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

Περίληψη

Η συνεχής βελτίωση στη διαδικασία κατασκευής των ολοκληρωμένων κυκλωμάτων σε συνδυασμό με τη σταθερή προσπάθεια για αύξηση της συνολικής απόδοσης, έχει οδηγήσει στην ανάπτυξη των πολυπύρηνων αρχιτεκτονικών όπου περισσότεροι από ένας επεξεργαστές τοποθετούνται στο ίδιο chip και μοιράζονται τα επίπεδα της ιεραρχίας μνήμης. Σχεδιαστικές λύσεις που αφορούν τις κρυφές μνήμες και οι οποίες έχουν μελετηθεί εκτενώς στα πλαίσια μονοπύρηνων συστημάτων, έχουν μεταφερθεί αργότερα από τους αρχιτέκτονες υπολογιστών στους Chip MutliProcessors (CMPs). Ένα χαρακτηριστικό παράδειγμα μιας τέτοιας αυτούσιας μεταφοράς, είναι η υιοθέτηση της πολιτικής αντικατάστασης του *ελάχιστα πρόσφατα χρησιμοποιημένου* (LRU) μπλοκ, η οποία είναι ευρέως αποδεκτή ως η καλύτερη πολιτική αντικατάστασης μπλοκ για κρυφές μνήμες μονοπύρηνων επεξεργαστών. Πρόσφατες έρευνες υποδεικνύουν ότι αρκετές παράμετροι διαφέρουν στα CMP συστήματα, δεδομένου ότι η ιεραρχία της κρυφής μνήμης καταμερίζεται μεταξύ των διαφόρων ταυτόχρονα εκτελούμενων νημάτων και επομένως χρίζουν περαιτέρω μελέτης.

Αυτή η διπλωματική εργασία εξετάζει την αποδοτικότητα ορισμένων σχεδίων καταμερισμού της κρυφής μνήμης που έχουν προταθεί, έναντι της απόδοσης της LRU πολιτικής αντικατάστασης. Τα σχέδια αξιολογούνται στην αρχική τους μορφή και έπειτα τα πλεονεκτήματα και τα μειονεκτήματά τους χρησιμοποιούνται ως οδηγός για την πρόταση και τη δοκιμή μερικών τροποποιήσεων.

Λέξεις κλειδιά: δυναμικός καταμερισμός, ιεραρχία μνήμης, πολυπύρηνες αρχιτεκτονικές CMP, διαμοιραζόμενη κρυφή μνήμη, πολιτική αντικατάστασης μπλοκ.

Abstract

The continuous improvement in the chip fabrication process together with the constant effort to increase overall performance, has led to the development of multicore architectures where more than one processors are placed on the same chip and share the levels of the memory hierarchy. Cache design solutions that have been extensively studied in the context of uniprocessor systems, have later been transferred by computer architects to CMPs. A typical example of such a migration is the employment of the Least Recently Used (LRU) replacement policy, which is widely accepted as the best line replacement policy for uniprocessor caches. Recent research has shown that several parameters are different in CMP systems, given that the cache hierarchy is shared among several concurrent threads and therefore need to be further studied.

This diploma thesis examines the efficiency of several proposed cache partitioning schemes compared to the performance of the LRU replacement policy. The schemes are evaluated in their original form and then their advantages and drawbacks are used as a guide for the proposal and testing of some modifications.

Keywords: dynamic partitioning, memory hierarchy, multicore architecture CMP, shared cache, cache replacement policy.

Ευχαριστίες

Η παρούσα διπλωματική εργασία πραγματοποιήθηκε στο Εργαστήριο Υπολογιστικών Συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου, υπό την επίβλεψη του Αναπληρωτή Καθηγητή Νεκτάριου Κοζύρη.

Καταρχήν θα ήθελα να ευχαριστήσω τον καθηγητή μου κ. Νεκτάριο Κοζύρη, για την εποπτεία του κατά την εκπόνηση της εργασίας μου αλλά και για τη συμβολή του με τη διδασκαλία του στη διαμόρφωσή μου ως μηχανικού.

Θα ήθελα επίσης να εκφράσω τις ευχαριστίες μου σε όλα τα μέλη του εργαστηρίου και ιδιαίτερα στο Μεταδιδακτορικό Ερευνητή Κωνσταντίνο Νίκα για τη συνεχή καθοδήγηση και ενθάρρυνση που μου προσέφερε για την ολοκλήρωση της διπλωματικής μου εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω το οικογενειακό και φιλικό μου περιβάλλον και κυρίως τους γονείς μου, η υποστήριξη των οποίων με βοήθησε τόσο στην εκπόνηση της εργασίας όσο και στην ολοκλήρωση των προπτυχιακών μου σπουδών.

Πίνακας Περιεχομένων

Λίστα Σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Γενικά

Η τεχνολογία των υπολογιστών έχει κάνει τεράστια βήματα εξέλιξης μέσα στα περίπου 60 χρόνια από την εμφάνιση του πρώτου ηλεκτρονικού υπολογιστή γενικού σκοπού. Αυτή η γρήγορη βελτίωση προήλθε τόσο από προόδους στην τεχνολογία κατασκευής των chip όσο και από καινοτόμες λύσεις στη σχεδίαση του υλικού των υπολογιστών. Αφενός η εμφάνιση του μικροεπεξεργαστή στα τέλη της δεκαετίας του 1970, έδωσε τη δυνατότητα ενσωμάτωσης όλων των βελτιώσεων της τεχνολογίας των ολοκληρωμένων κυκλωμάτων στους επεξεργαστές. Αφετέρου, η υιοθέτηση γλωσσών προγραμματισμού υψηλού επιπέδου μείωσε την ανάγκη για συμβατότητα με τον αντικειμενικό κώδικα (object code), ενώ η δημιουργία τυποποιημένων και ανεξάρτητων από συγκεκριμένους κατασκευαστές λειτουργικών συστημάτων όπως το UNIX, μείωσαν το κόστος και τον κίνδυνο μετάβασης σε νέες αρχιτεκτονικές και ειδικότερα στις αρχιτεκτονικές RISC.

Από την εποχή της εμφάνισης των αρχιτεκτονικών RISC και μέχρι πρόσφατα, οι αρχιτέκτονες υπολογιστών είχαν επικεντρωθεί σε τρείς βασικές τεχνικές αύξησης της απόδοσης. Η πρώτη και κυριότερη ήταν η αύξηση της συχνότητας των επεξεργαστών, η οποία επιτυγχανόταν με τη διαρκή σμίκρυνση της κλίμακας ολοκλήρωσης και συντελούσε σε άμεση αύξηση της απόδοσης. Η δεύτερη τεχνική ήταν η εκμετάλλευση του παραλληλισμού στο επίπεδο των εντολών, αρχικά με τη διοχέτευση (pipelining) και στη συνέχεια με την ταυτόχρονη προσκόμιση πολλών εντολών ανά κύκλο ρολογιού (multiple instruction issue). Η τρίτη ήταν η ενσωμάτωση κρυφών μνημών στα chip των επεξεργαστών, καταρχήν σε απλές μορφές και ακολούθως υλοποιώντας πιο περίπλοκες οργανώσεις και βελτιστοποιήσεις.

Από το 1986 έως και το 2002, οι προαναφερθείσες βελτιώσεις οδήγησαν σε μια ετήσια αύξηση των επιδόσεων της τάξης του 50%. Όμως από το 2002 και μέχρι σήμερα ο ρυθμός της εξέλιξης μειώθηκε στο 20% το χρόνο για τρείς βασικούς λόγους. Ο πρώτος, σχετίζεται με τους περιορισμούς στη δυνατότητα απαγωγής θερμότητας από τα ως επί το πλείστον

αερόψυκτα chip και συνακόλουθα στην κατανάλωση ρεύματος, που εμποδίζουν την αύξηση της συχνότητας. Ο δεύτερος είναι ο μικρός βαθμός εναπομένοντος παραλληλισμού στην εκτέλεση των εντολών, ενώ ο τρίτος είναι η μικρή βελτίωση στις καθυστερήσεις της ιεραρχίας μνήμης, που την καθιστούν στενωπό της απόδοσης.

Για την αντιμετώπιση των παραπάνω προβλημάτων οι αρχιτέκτονες υπολογιστών στράφηκαν στην ανάπτυξη πολυπύρηνων επεξεργαστών. Οι Chip MultiProcessors (CMPs) επιτυγχάνουν μεγαλύτερη απόδοση από τους μονοπύρηνους επεξεργαστές, προτάσσοντας την αύξηση του αριθμού των πυρήνων έναντι της αύξησης της συχνότητας. Επιπλέον οι πολυπύρηνες αρχιτεκτονικές εστιάζουν στον παραλληλισμό σε επίπεδο νημάτων και όχι σε επίπεδο εντολών, ενώ παρουσιάζουν διαφορετικές απαιτήσεις ως προς την κρυφή μνήμη.

1.2 Πολυπύρηνες αρχιτεκτονικές

Η αξιοσημείωτη πρόοδος που έχει παρατηρηθεί στην τεχνολογία των ολοκληρωμένων κυκλωμάτων έχει καταστήσει ευκολότερη από ποτέ την εισαγωγή πολλαπλών πυρήνων στο ίδιο chip στους σύγχρονους επεξεργαστές. Επιπλέον, οι σχεδιαστές υλικού υπολογιστών κάνουν λόγο για δεκάδες ή και εκατοντάδες πυρήνες στο ίδιο chip τα επόμενα χρόνια. Σε τέτοια υπολογιστικά συστήματα η εκτέλεση φόρτων εργασίας πολλαπλών εφαρμογών είναι μια συνήθης περίπτωση. Ωστόσο έχει διαπιστωθεί ότι ασκούν σημαντική πίεση στο σύστημα μνήμης, φαινόμενο το οποίο αναμένεται να ενταθεί στο μέλλον.

Η μελέτη της απόδοσης των συστημάτων μνήμης στις πολυπύρηνες αρχιτεκτονικές, όπου οι φόρτοι εργασίας πολλαπλών εφαρμογών δημιουργούν ένα περιβάλλον ιδιαίτερα απαιτητικό σε πόρους μνήμης, οδήγησε γρήγορα στο συμπέρασμα ότι πολλές υπάρχουσες σχεδιάσεις είναι ανεπαρκείς. Ειδικότερα, μεγάλη βαρύτητα για τη συνολική απόδοση φαίνεται να έχει η σχεδίαση του τελευταίου επιπέδου της διαμοιραζόμενης κρυφής μνήμης των CMPs. Εντούτοις, τα προβλήματα στη διαμόρφωση και τη διαχείρισή της δεν είναι ακόμα πλήρως κατανοητά και ορισμένες σχεδιαστικές λύσεις που υιοθετούνται προέρχονται από προηγούμενες μονοπύρηνες αρχιτεκτονικές, χωρίς να έχουν δοκιμαστεί επαρκώς.

Συμπερασματικά, τα chip με πολλούς πυρήνες, έχουν καθιερωθεί εδώ και μερικά χρόνια ως ο κοινά αποδεκτός διάδοχος των μονοπύρηνων συστημάτων στην προσπάθεια της βιομηχανίας υπολογιστικών συστημάτων για αύξηση της απόδοσης. Παρόλα αυτά, η αλλαγή φιλοσοφίας στη σχεδίαση των επεξεργαστών έχει δημιουργήσει νέες προκλήσεις,

μία από τις κυριότερες εκ των οποίων είναι η σχεδίαση της ιεραρχίας μνήμης τόσο συνολικά όσο και των επιμέρους συστατικών της. Επιπλέον, η πληθώρα επεξεργαστών που λανσάρονται στην αγορά και οι διαφορετικές προσεγγίσεις που υιοθετούν, υποδεικνύουν ότι οι αρχιτέκτονες υπολογιστών δεν έχουν καταλήξει σε κοινά αποδεκτές σχεδιαστικές προτάσεις.

Στον παρακάτω πίνακα έχουν συγκεντρωθεί ορισμένα αντιπροσωπευτικά παραδείγματα υπερσύγχρονων πολυπύρηνων επεξεργαστών (1), (2), (3), (4) που παράγονται από τις μεγαλύτερες εταιρείες που δραστηριοποιούνται στον τομέα κατασκευής chip επεξεργαστών. Ο πίνακας προσφέρει μια συνοπτική παρουσίαση των σχεδιαστικών τάσεων της βιομηχανίας στους πολυπύρηνους επεξεργαστές, με έμφαση στην ιεραρχία μνήμης. Από τα παραδείγματα εξάγεται εύκολα το συμπέρασμα ότι οι προτάσεις των κατασκευαστών συγκλίνουν σε κάποιους τομείς ενώ σε άλλους είναι τόσες όσοι και οι επεξεργαστές που συγκρίνονται.

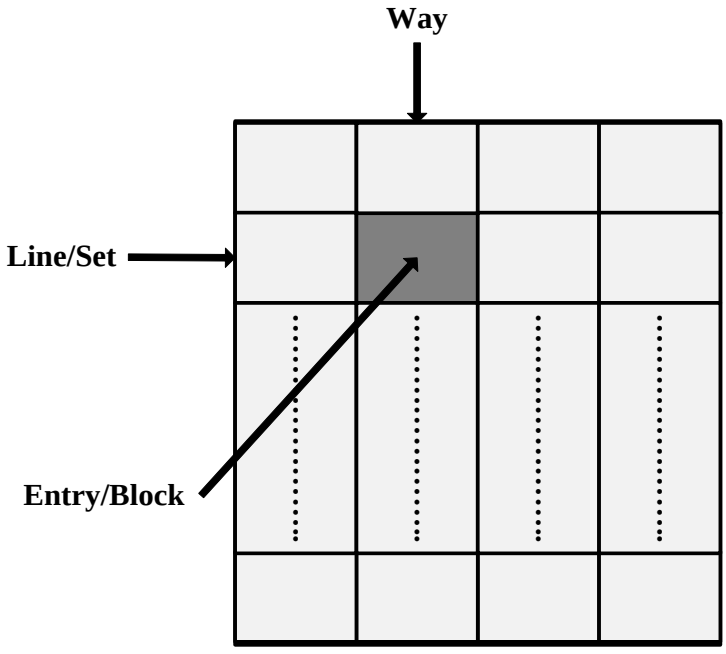
	Intel Dunnington	**Intel Nehalem**	**AMD Istanbul**	**UltraSPARC T2**
Processors per chip	6	4	6	8
Threads per processor	1	2	1	8
Instruction Cache per processor	32 Kbytes	32 Kbytes	64 Kbytes, 2-way associative	16 Kbytes, 8-way associative
Data Cache per processor	32 Kbytes	32 Kbytes	64 Kbytes 2-way associative	8 Kbytes, 4-way associative
On-chip L2 Cache	3 Mbytes per 2 cores shared	256 Kbytes per core	512 Kbytes, 16-way associative per core	4 Mbytes, 16-way associative, shared
L3 Cache	16 Mbytes, shared	8 Mbytes, shared	6128 Kbytes, Maximum 48-way associative	-

Πίνακας 1.1: Ιεραρχία μνήμης σύγχρονων CMPs

1.3 Ορολογία

Σε γενικές γραμμές η εργασία αυτή υιοθετεί την ορολογία που χρησιμοποίησε ο Smith στο (5). Στη συνέχεια παρατίθεται ένα σύνολο ορισμών που καθιστά σαφείς τους όρους που χρησιμοποιούνται σε όλο το κείμενο. Επίσης το σχήμα 1.1 παρουσιάζει ένα παράδειγμα που βοηθά στην καλύτερη κατανόηση του στοιχείου στο οποίο αναφέρεται κάθε όρος.

• **Καταχώρηση/Μπλοκ (Entry/Block):** Μία καταχώρηση είναι ένα σύνολο από λέξεις με κοινό tag, που διαχωρίζονται μόνο από τα λιγότερο σημαντικά bits της διεύθυνσης της μνήμης. Τυπικά έχουν μήκος από 4 έως 8 λέξεις.
• **Γραμμή/Σύνολο (Line/Set):** Μία γραμμή ή ένα σύνολο είναι μια συλλογή από καταχωρήσεις, τα tags των οποίων ελέγχονται παράλληλα κατά την προσπέλαση της κρυφής μνήμης.
• **Δρόμος (Way):** Το πλήθος των δρόμων καθορίζεται από το βαθμό συσχετιστικότητας της κρυφής μνήμης και είναι ίσο με τον αριθμό των καταχωρήσεων που περιέχονται σε κάθε γραμμή.



Σχήμα 1.1: Παράδειγμα μιας συσχετιστικής κρυφής μνήμης 4 δρόμων

Κεφάλαιο 2

Ιεραρχία Μνήμης

2.1 Γενικά

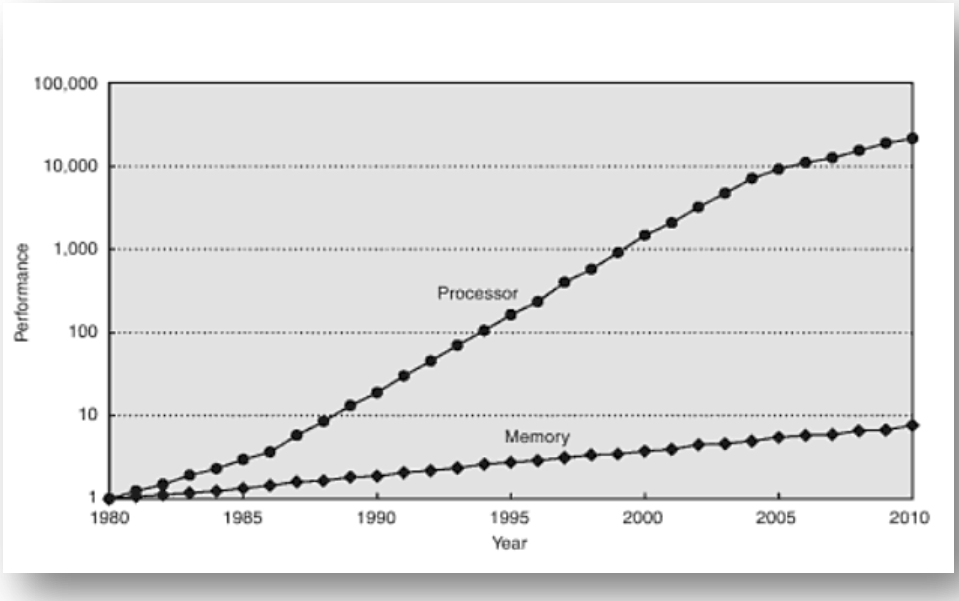
Τα τελευταία χρόνια έχει καταστεί σαφές ότι υπάρχουν αρκετά περιθώρια βελτίωσης της οργάνωσης της ιεραρχίας μνήμης και οι αρχιτέκτονες υπολογιστών αντιμετωπίζουν θεμελιώδη ερωτήματα για τη σχεδίαση και την ενσωμάτωση νέων λύσεων στις πολυπύρηνες αρχιτεκτονικές. Η πρόκληση καθίσταται μεγαλύτερη, καθώς τα σύγχρονα πολυπύρηνα συστήματα πρέπει να παρουσιάζουν βέλτιστη απόδοση σε ένα ολοένα αυξανόμενο φάσμα εφαρμογών, οι οποίες προέρχονται από διάφορα πεδία και έχουν διαφορετικές απαιτήσεις.

2.2 Σημασία της ιεραρχίας μνήμης

Μερικοί από τους πρωτοπόρους στον τομέα της αρχιτεκτονικής υπολογιστών όπως οι A. W. Burks, H. H. Goldstine και J. von Neumann προέβλεψαν πριν από πολλά χρόνια ότι οι προγραμματιστές θα ήθελαν απεριόριστες ποσότητες γρήγορης μνήμης και συγκεκριμένα: «Ιδανικά κάποιος θα επιθυμούσε μία απείρως μεγάλη χωρητικότητα μνήμης τέτοια ώστε οποιαδήποτε συγκεκριμένη …λέξη θα ήταν άμεσα διαθέσιμη…. Είμαστε… υποχρεωμένοι να αναγνωρίσουμε την πιθανότητα κατασκευής μιας ιεραρχίας από μνήμες, κάθε μία από τις οποίες έχει μεγαλύτερη χωρητικότητα από την προηγούμενη αλλά που είναι γρήγορα προσπελάσιμη.» (6).

Σήμερα, η οικονομικά αποδοτικότερη λύση στην απαίτηση για ταχεία μνήμη μεγάλης χωρητικότητας είναι κατά γενική παραδοχή μια ιεραρχία μνήμης η οποία στηρίζει την αποδοτικότητά της στην τοπικότητα των αναφορών στη μνήμη και στην απόδοση ως προς κόστος των διαφόρων τεχνολογιών μνημών. Η αρχή της τοπικότητας των αναφορών τόσο χρονικής όσο και τοπικής, σε συνδυασμό με την κατασκευαστική αρχή ότι ένα μικρότερο κομμάτι υλικού (hardware) μπορεί να καταστεί ταχύτερο από ένα μεγαλύτερο, ουσιαστικά επέβαλαν τη δημιουργία ιεραρχιών μνημών που αποτελούνται από τμήματα μνήμης διαφορετικών μεγεθών, ταχυτήτων και κόστους.

Η σχεδίαση της ιεραρχίας μνήμης υπήρξε κυρίαρχο ζήτημα της αρχιτεκτονικής υπολογιστών για μεγάλη χρονική περίοδο, εξαιτίας της τεράστιας προόδου στην ταχύτητα και την απόδοση των επεξεργαστών. Το χάσμα στην απόδοση ανάμεσα στη μνήμη και τους επεξεργαστές παρουσιάζεται στο σχήμα 2.1. Η αύξηση της απόδοσης των επεξεργαστών κυμαινόταν στο 25% κάθε χρόνο μέχρι το 1986, στη συνέχεια στο 52% μέχρι το 2004 και ακολούθως στο 20% περίπου. Αντίθετα η βελτίωση της απόδοσης της μνήμης παρέμεινε στο ίδιο χρονικό διάστημα στο 7% κάθε χρόνο.



Σχήμα 2.1: Χάσμα ταχύτητας μεταξύ επεξεργαστή και μνήμης (7)

Τα τελευταία χρόνια η σημασία της ιεραρχίας μνήμης υπογραμμίστηκε ακόμη περισσότερο από τις αυξημένες απαιτήσεις σε χωρητικότητα και ταχύτητα μνήμης, που σηματοδότησε η έλευση των πολυπύρηνων επεξεργαστών.

Οι σύγχρονοι επεξεργαστές περιέχουν πολλούς πυρήνες, γεγονός το οποίο καθιστά δυνατή την παράλληλη εκτέλεση πολλών εφαρμογών – νημάτων στο ίδιο chip. Αυτή η αυξητική τάση συνεπάγεται σημαντικά μεγαλύτερο φόρτο εργασίας για την ιεραρχία μνήμης, η οποία πρέπει να υποστηρίξει τις απαιτήσεις σε μνήμη πολλών και συχνά ετερογενών εφαρμογών. Ένα από τα κρισιμότερα σημεία για την επίτευξη υψηλών επιδόσεων σε αυτές τις πολυπύρηνες αρχιτεκτονικές είναι η αποδοτική διαχείριση του τελευταίου επιπέδου της ενσωματωμένης στο chip διαμοιραζόμενης μνήμης. Στόχος είναι η ελαχιστοποίηση των αστοχιών στο επίπεδο αυτό και επομένως η ελαχιστοποίηση των προσβάσεων στην κύρια

μνήμη που βρίσκεται εκτός του chip και η οποία είναι τάξεις μεγέθους πιο αργή από την κρυφή μνήμη.

2.3 Ζητήματα Σχεδίασης της Ιεραρχίας Μνήμης στις Πολυπύρηνες Αρχιτεκτονικές

Οι πολυπύρηνοι επεξεργαστές έχουν δημιουργήσει νέες απαιτήσεις αλλά και ερωτήματα σχετικά με την οργάνωση της ιεραρχίας μνήμης και ιδιαίτερα του διαμοιραζόμενου τμήματος αυτής. Η έντονη ερευνητική δραστηριότητα σε αυτόν τον τομέα της αρχιτεκτονικής υπολογιστών, μαρτυρά την ύπαρξη αρκετών ζητημάτων σχεδίασης που χρίζουν μελέτης και βελτίωσης. Στη συνέχεια εξετάζονται συνοπτικά ορισμένα από αυτά τα ζητήματα.

Καταρχήν μία βασική ιδιαιτερότητα των πολυπύρηνων συστημάτων είναι ότι διάφορες εφαρμογές εκτελούνται παράλληλα, ενώ συγχρόνως το όφελος που αποκομίζει η καθεμία από τη λήψη πρόσθετων πόρων της κρυφής μνήμης δεν είναι το ίδιο. Ακόμα σημαντικότερο είναι το γεγονός ότι οι εφαρμογές παρουσιάζουν διακριτές φάσεις κατά τη διάρκεια της εκτέλεσής τους, κάθε μια από τις οποίες έχει πολύ διαφορετικές απαιτήσεις σε μνήμη. Επομένως ειδικά στις πολυπύρηνες αρχιτεκτονικές, παρουσιάζει ενδιαφέρον ένας δυναμικός καταμερισμός της διαμοιραζόμενης κρυφής μνήμης, με σκοπό τη μεγιστοποίηση της συνολικής απόδοσης του συστήματος.

Όπως αναφέρθηκε, η εισαγωγή κρυφών μνημών στα chip επεξεργαστών αποσκοπεί στην ταχύτερη ικανοποίηση κατά το δυνατόν μεγαλύτερου ποσοστού αιτημάτων μνήμης των εκτελούμενων νημάτων. Μια λύση που συντελεί στη μείωση των αστοχιών είναι ο διαμοιρασμός της κρυφής μνήμης ο οποίος αντιμετωπίζει βασικά προβλήματα, εντούτοις όμως έχει και μειονεκτήματα. Κατά κανόνα, οι διαμοιραζόμενες κρυφές μνήμες παρουσιάζουν υψηλότερους χρόνους πρόσβασης και οι απαιτήσεις εύρους ζώνης του συστήματος αυξάνονται. Από την άλλη, οι ιδιωτικές κρυφές μνήμες έχουν χαμηλότερες καθυστερήσεις στην περίπτωση ευστοχίας, αλλά η χωρητικότητα της κρυφής μνήμης του συστήματος ουσιαστικά μειώνεται όταν χρησιμοποιούνται στη θέση διαμοιραζόμενων μνημών, δεδομένου ότι ένας αριθμός καταχωρήσεων επαναλαμβάνεται σε πολλές από αυτές τις ιδιωτικές κρυφές μνήμες.

Είναι προφανές ότι οι αρχιτέκτονες υπολογιστών επιλέγουν την κατά περίπτωση καταλληλότερη σχεδίαση, συγκρίνοντας τα πλεονεκτήματα και τα μειονεκτήματα ανάλογα με τις απαιτήσεις και τις προδιαγραφές του κατασκευαζόμενου συστήματος. Σήμερα, οι πολυπύρηνοι επεξεργαστές φέρουν στην πλειονότητά τους ιδιωτικές κρυφές μνήμες εντολών και δεδομένων πρώτου επιπέδου (private L1 instruction και data caches), προκειμένου να διατηρηθεί ο χρόνος πρόσβασης σε χαμηλά επίπεδα. Αναφορικά με την κρυφή μνήμη δευτέρου επιπέδου (L2 cache), σε ορισμένες περιπτώσεις αποτελεί το τελευταίο επίπεδο ενσωματωμένης στο chip μνήμης, ενώ σε άλλες το ενδιάμεσο. Στην πρώτη περίπτωση είναι συνήθως διαμοιραζόμενη από όλους τους πυρήνες και χρησιμοποιείται τόσο για εντολές όσο και για δεδομένα, ενώ στη δεύτερη είναι κατακερματισμένη σε ιδιωτικά (για κάθε πυρήνα) ή διαμοιραζόμενα από ομάδες πυρήνων, τμήματα. Τέλος όπου υπάρχει, η κρυφή μνήμη τρίτου επιπέδου είναι διαμοιραζόμενη από όλους τους πυρήνες και χρησιμοποιείται τόσο για εντολές όσο και για δεδομένα (L3 shared cache).

Μια άλλη σημαντική πρόκληση που αντιμετωπίζουν οι σχεδιαστές πολυπύρηνων επεξεργαστών είναι η επιλογή ενός αποδοτικού αλγορίθμου αντικατάστασης μπλοκ της κρυφής μνήμης (cache replacement policy). Δημοφιλείς αρχιτεκτονικές υιοθετούν πολιτικές αντικατάστασης που έχουν μελετηθεί εκτενώς για μονοπύρηνα συστήματα όπως η *τυχαία* (Random) και η πολιτική αντικατάστασης του *ελάχιστα πρόσφατα χρησιμοποιημένου* μπλοκ (LRU). Οι αρχιτέκτονες υπολογιστών έχουν «δανειστεί» τις προαναφερθείσες πολιτικές αντικατάστασης από τους μονοπύρηνους επεξεργαστές, γεγονός που συνεπάγεται ότι αυτές δε λαμβάνουν υπόψη την ταυτότητα του νήματος που προσκόμισε την καταχώρηση στην κρυφή μνήμη ή του νήματος που προκάλεσε την αστοχία, κατά την επιλογή της καταχώρησης που θα απορριφθεί (*threadblind* πολιτική). Το τελευταίο, ενδέχεται να προκαλέσει προβλήματα σε μια πολυπύρηνη αρχιτεκτονική και να συντελέσει σε σημαντική μείωση της απόδοσης.

Πιο συγκεκριμένα σε ένα πολυπύρηνο σύστημα, όταν κάθε πυρήνας χρησιμοποιείται για να εκτελέσει διαφορετικά νήματα, πολλαπλά σύνολα εργασίας ανταγωνίζονται για τον ίδιο χώρο στην κρυφή μνήμη. Η υιοθέτηση μιας *threadblind* πολιτικής αντικατάστασης επιτρέπει τις συγκρούσεις (interference) ανάμεσα στα νήματα, δηλαδή τα στοιχεία που ανήκουν σε ένα νήμα μπορούν να εκδιωχθούν από καταχωρήσεις στοιχείων που ανήκουν σε άλλα νήματα. Ένα χαρακτηριστικό παράδειγμα, κατά το οποίο οι *threadblind* πολιτικές

έχουν πολύ χαμηλή απόδοση είναι η εκτέλεση των εφαρμογών ροής (streaming applications) συγχρόνως με άλλες εφαρμογές. Αυτές οι εφαρμογές παρουσιάζουν μικρό ποσοστό χρονικής τοπικότητας και γεμίζουν την κρυφή μνήμη με στοιχεία που δεν θα προσπελαστούν πάλι στο κοντινό μέλλον, αντικαθιστώντας άλλα που η διατήρησή τους θα ήταν ωφέλιμη. Μια *threadblind* πολιτική αντικατάστασης δεν είναι ικανή να διαγνώσει αυτό το πρόβλημα, συντελώντας έτσι στην υποβάθμιση της απόδοσης.

Ο πλέον παραδοσιακός τρόπος αντιμετώπισης του προβλήματος των διενέξεων (conflicts) μεταξύ των νημάτων, είναι η αύξηση του μεγέθους και του βαθμού της συσχετιστικότητας της κρυφής μνήμης. Οι τεχνικές αυτές που μειώνουν τα ποσοστά αστοχιών χώρου και διενέξεων (capacity και conflict misses αντίστοιχα). Εντούτοις, η αποτελεσματικότητα αυτής της προσέγγισης τίθεται υπό αμφισβήτηση λόγω των περιορισμών χώρου και κατανάλωσης ρεύματος και συνεπώς απαιτείται η εξεύρεση εναλλακτικών λύσεων.

Η εργασία αυτή φιλοδοξεί να συνεισφέρει στην έρευνα για την εξεύρεση λύσεων στο πρόβλημα της αποδοτικής διαχείρισης του τελευταίου επιπέδου της διαμοιραζόμενης κρυφής μνήμης των πολυπύρηνων αρχιτεκτονικών. Στη συνέχεια του κεφαλαίου αυτού παρουσιάζονται οι διαφορετικές πολιτικές αντικατάστασης μπλοκ που χρησιμοποιούνται στις κρυφές μνήμες, καθώς και ορισμένες ερευνητικές προσπάθειες των τελευταίων χρόνων που αποσκοπούν στην επίλυση του προβλήματος.

2.4 Πολιτικές Αντικατάστασης Μπλοκ

Η πολιτική αντικατάστασης μπλοκ είναι μία από τις βασικότερες παραμέτρους κατά τη σχεδίαση της ιεραρχίας μνήμης, διότι η αποδοτικότητα της υιοθετούμενης πολιτικής καθορίζει σε μεγάλο βαθμό τα ποσοστά ευστοχίας στην κρυφή μνήμη. Επιπλέον, η έλευση των πολυπύρηνων αρχιτεκτονικών περιέπλεξε το έργο των σχεδιαστών υλικού, γεγονός που καθιστά την έρευνα στον τομέα αυτό ιδιαίτερα επίκαιρη. Στη συνέχεια εξετάζονται και συγκρίνονται εν συντομία ορισμένες πολιτικές αντικατάστασης μπλοκ.

Μία ιδανική πολιτική αντικατάστασης μπλοκ απορρίπτει κάθε φορά από όλα τα στοιχεία της κρυφής μνήμης εκείνο το οποίο πρόκειται να μην επαναχρησιμοποιηθεί για το μεγαλύτερο διάστημα στο μέλλον. Προφανώς ένας τέτοιος αλγόριθμος είναι μη υλοποιήσιμος αφού δεν υπάρχει δυνατότητα να γνωρίζει κανείς εκ των προτέρων ποιά δεδομένα θα χρειαστεί μια εκτελούμενη εφαρμογή και ποιά χρονική στιγμή. Συνεπώς, αυτή

η ιδανική πολιτική αντικατάστασης έχει νόημα ύπαρξης μόνο στις προσομοιώσεις, για τον υπολογισμό του άνω φράγματος της απόδοσης που μπορεί να επιτύχει οποιαδήποτε άλλη υλοποιήσιμη πολιτική αντικατάστασης.

Η *Least Recently Used* (LRU) πολιτική αντικατάστασης όπως υποδηλώνει και το όνομά της, απορρίπτει πρώτα τα ελάχιστα πρόσφατα χρησιμοποιημένα μπλοκ και είναι ίσως η σημαντικότερη πολιτική μέχρι σήμερα. Αυτός ο αλγόριθμος απαιτεί να διατηρείται πληροφορία σχετικά με το πότε χρησιμοποιήθηκε κάθε μπλοκ, το οποίο κοστίζει σε υλικό (hardware overhead) στην περίπτωση που πρέπει να εξασφαλιστεί ότι ο αλγόριθμος απορρίπτει πάντα το ελάχιστα πρόσφατα χρησιμοποιημένο μπλοκ. Γενικά, μια υλοποίηση αυτής της τεχνικής απαιτεί να διατηρούνται δυαδικά ψηφία ηλικίας (age bits) για κάθε μπλοκ της κρυφής μνήμης. Επίσης κάθε φορά που ένα μπλοκ προσπελαύνεται η ηλικία όλων των άλλων μπλοκ της κρυφής μνήμης αλλάζει. Στην πραγματικότητα η LRU πολιτική συνιστά μια οικογένεια πολιτικών αντικατάστασης για κρυφές μνήμες με πολλές παραλλαγές.

Μία γνωστή περίπτωση από την οικογένεια των LRU πολιτικών αντικατάστασης, είναι η *Pseudo-LRU* (PLRU) πολιτική αντικατάστασης. Για κρυφές μνήμες με μεγάλο βαθμό συσχετιστικότητας (συνήθως μεγαλύτερο από 4 δρόμους), το κόστος υλοποίησης της LRU πολιτικής καθίσταται απαγορευτικό λόγω της ανάγκης ύπαρξης πολλαπλάσιων age bits. Ο PLRU αλγόριθμος στηρίζεται στην παραδοχή ότι ένα πιθανοτικό μοντέλο που διατηρεί μόνο ένα δυαδικό ψηφίο (bit) για κάθε στοιχείο της κρυφής μνήμης απορρίπτει σχεδόν πάντα τα ελάχιστα πρόσφατα χρησιμοποιημένα στοιχεία, γεγονός που συνεπάγεται μια αβεβαιότητα για το κατά πόσο τα στοιχεία που βρίσκονται μέσα στην κρυφή μνήμη έχουν χρησιμοποιηθεί πρόσφατα. Η PLRU πολιτική χρησιμοποιείται για τη δραστική μείωση του κόστους σε υλικό.

Μία ακόμη πολιτική αντικατάστασης η οποία χρησιμοποιείται από πολύ σημαντικούς κατασκευαστές chip όπως η IBM, είναι η πολιτική αντικατάστασης του *μη-πρόσφατα χρησιμοποιημένου μπλοκ* (Not Recently Used – NRU). Η πολιτική αυτή μοιάζει με την LRU πολιτική ως προς το ότι ευνοεί τη διατήρηση στην κρυφή μνήμη μπλοκ που έχουν προσπελαστεί πρόσφατα στο παρελθόν. Ο μηχανισμός λειτουργίας της περιλαμβάνει κάποια bit τα οποία τίθενται όταν ένα μπλοκ τροποποιείται (εγγράφεται) ή όταν γίνεται αναφορά σε αυτή. Σε προκαθορισμένα χρονικά διαστήματα τα bit αυτά τίθενται και πάλι στο μηδέν, συνεπώς μόνο τα μπλοκ που προσπελάστηκαν στο τελευταίο χρονικό διάστημα

είναι σημειωμένα. Ο αλγόριθμος επιλέγει προς αντικατάσταση τα στοιχεία εκείνα, που βάσει των αντίστοιχων bits δεν έχουν προσπελαστεί και δεν έχουν τροποποιηθεί στο τελευταίο χρονικό διάστημα.

Η *First In First Out (FIFO)* πολιτική αντικατάστασης προσομοιώνει μια ουρά στην οποία τα καινούρια μπλοκ εισέρχονται μπροστά, εκδιώκοντας το μπλοκ που βρίσκεται στο πίσω μέρος της ουράς. Οι υλοποιήσεις συχνά χρησιμοποιούν ένα round-robin μετρητή αντικατάστασης για κάθε σύνολο, ο οποίος δείχνει στο μπλοκ της κρυφής μνήμης που πρόκειται να αντικατασταθεί στη συνέχεια. Αυτός ο μετρητής αυξάνεται όταν ένα στοιχείο εισάγεται σε ένα σύνολο, ενώ μια ευστοχία δεν τον μεταβάλλει. Λόγω της απλότητάς της έχει μικρό κόστος υλοποίησης σε υλικό αλλά στις περισσότερες πρακτικές εφαρμογές παρουσιάζει χαμηλή απόδοση. Συνεπώς σπάνια χρησιμοποιείται χωρίς τροποποιήσεις.

Η *τυχαία* πολιτική αντικατάστασης (Random replacement policy) επιλέγει ένα μπλοκ της κρυφής μνήμης προς αντικατάσταση με τυχαίο τρόπο, πιθανόν και με τη χρήση κάποιας βοήθειας από το υλικό. Το μεγαλύτερο πλεονέκτημα της πολιτικής αυτής, είναι το ελάχιστο κόστος σε υλικό που απαιτεί η υλοποίησή της, καθώς δεν υπάρχει ανάγκη παρακολούθησης των προσπελάσεων των μπλοκ της κρυφής μνήμης. Συνήθως αποδίδει καλύτερα από την FIFO πολιτική αντικατάστασης, ενώ όταν εκτελούνται βρόχοι με πολλές προσπελάσεις στη μνήμη αποδίδει καλύτερα και από την LRU πολιτική. Στη γενική περίπτωση έχει χαμηλότερη απόδοση από την LRU πολιτική αντικατάστασης και για το λόγο αυτό δεν απαντάται συχνά σε διαμοιραζόμενες κρυφές μνήμες. Η Sun Microsystems τη χρησιμοποιεί στις ιδιωτικές κρυφές μνήμες εντολών πρώτου επιπέδου του ULTRA SPARC T2.

Τέλος το αντίθετο της LRU πολιτικής, είναι η πολιτική αντικατάστασης του *περισσότερο πρόσφατα χρησιμοποιημένου* μπλοκ (Most Recently Used – MRU), η οποία απορρίπτει πρώτα τα μπλοκ που έχουν χρησιμοποιηθεί πιο πρόσφατα στο παρελθόν. Σε ορισμένες περιπτώσεις, όπως όταν ένα αρχείο προσπελαύνεται κατ’ επανάληψη και με σειριακό τρόπο η MRU πολιτική αντικατάστασης αποδίδει καλά. Επίσης εμφανίζει καλύτερη απόδοση από την LRU πολιτική αντικατάστασης σε περιπτώσεις επαναλαμβανόμενων σαρώσεων μεγάλων συνόλων δεδομένων, ακριβώς επειδή διατηρεί στην κρυφή μνήμη τα παλαιότερα δεδομένα. Ωστόσο αυτό δεν ισχύει για την πλειοψηφία των σύγχρονων αντιπροσωπευτικών εφαρμογών.

2.5 Σχεδιαστικές Προτάσεις για τη Διαχείριση της Διαμοιραζόμενης Κρυφής Μνήμης στις Πολυπύρηνες Αρχιτεκτονικές

Τα CMP συστήματα έχουν καθιερωθεί τα τελευταία χρόνια ως κυρίαρχη σχεδιαστική πρακτική, επειδή καλύπτουν τις ανάγκες για μεγαλύτερη απόδοση χωρίς να αυξάνουν πολύ τη σχεδιαστική πολυπλοκότητα και την κατανάλωση, ενώ δεν επιδεινώνουν περαιτέρω το χάσμα στην ταχύτητα μεταξύ επεξεργαστή και μνήμης. Ωστόσο, η συνεχής αύξηση του αριθμού των πυρήνων που ενσωματώνονται στο ίδιο chip, δημιουργεί νέες προκλήσεις και οι σχεδιαστές υλικού αναζητούν λύσεις σε προβλήματα που δεν υπήρχαν στις μονοπύρηνες αρχιτεκτονικές. Το γεγονός αυτό συνετέλεσε στην παρουσίαση πολλών διαφορετικών προσεγγίσεων οι οποίες σε πολλές περιπτώσεις αφορούν την ενσωματωμένη στο chip κρυφή μνήμη. Ορισμένες από αυτές παρουσιάζονται στη συνέχεια.

2.5.1 Non Uniform Cache Architecture πολιτικές

Οι Lira και συνεργάτες παρουσίασαν στο (8) μια πρωτότυπη προσέγγιση για τη διαχείριση της ενσωματωμένης στο chip κρυφής μνήμης, η οποία στηρίζεται στη διαπίστωση ότι οι καθυστερήσεις επικοινωνίας μεταξύ των λειτουργικών μονάδων πάνω στο chip διαρκώς αυξάνονται. Ειδικότερα, σημείο αφετηρίας υπήρξε η διαρκής επιμήκυνση των αγωγών (σε σχέση με τη συρρίκνωση των chip) που ενώνουν τους πυρήνες με τις κρυφές μνήμες. Οι καθυστερήσεις των καλωδιώσεων (wire delays) έχουν ανεβάσει κατά πολύ το άνω φράγμα καθυστέρησης επικοινωνίας, καθιστώντας αργή την προσπέλαση των μπλοκ. Για το λόγο αυτό, έστρεψαν την έρευνά τους στις ανομοιόμορφες αρχιτεκτονικές κρυφών μνημών (Non-Uniform Cache Architectures – NUCA).

Η κύρια ιδέα στην οποία στηρίζεται το NUCA είναι ο διαχωρισμός της κρυφής μνήμης σε μικρότερα τμήματα (τράπεζες-banks). Το κέρδος είναι ότι καθένα από αυτά τα τμήματα έχει μια μοναδική καθυστέρηση πρόσβασης, η οποία είναι πολύ μικρότερη από την καθυστέρηση που θα είχε μια μη-διαχωρισμένη κρυφή μνήμη. Η απώλεια έγκειται στον επιπλέον χρόνο που χρειάζεται για την εύρεση του τμήματος που περιέχει το μπλοκ που αναζητείται, δεδομένου ότι τα μπλοκ κατανέμονται σε όλα τα διαθέσιμα τμήματα. Το μοντέλο NUCA, ουσιαστικά περιλαμβάνει τέσσερις πολιτικές:

- **Την πολιτική τοποθέτησης (Bank Placement Policy):** αυτή καθορίζει σε ποιά τμήματα δύναται να τοποθετηθεί ένα μπλοκ που προέρχεται από ανώτερο ή κατώτερο επίπεδο της ιεραρχίας μνήμης.
- **Την πολιτική πρόσβασης (Bank Access Policy):** προσδιορίζει τον αλγόριθμο αναζήτησης τμημάτων και τον τρόπο που αυτά προσπελαύνονται (παράλληλα, σειριακά κλπ.)
- **Την πολιτική μετανάστευσης (Bank Migration Policy):** αποφασίζει για το αν και πότε ένα μπλοκ μπορεί να μετακινηθεί από ένα τμήμα σε κάποιο άλλο.
- **Την πολιτική αντικατάστασης (Bank Replacement Policy):** αυτή καθορίζει τη συμπεριφορά της αρχιτεκτονικής NUCA όταν ένα μπλοκ εκδιώκεται από κάποιο τμήμα της κρυφής μνήμης.

Κάθε μία από τις παραπάνω πολιτικές αποτελεί ένα ξεχωριστό και σημαντικό πεδίο έρευνας, που μπορεί να οδηγήσει στην κατασκευή αποδοτικότερων κρυφών μνημών.

2.5.2 Πολιτικές Εισαγωγής Μπλοκ

Οι Qureshi και συνεργάτες στο (9) και (10), εξετάζοντας μονοπύρηνες αρχιτεκτονικές παρατήρησαν ότι σε ορισμένες εφαρμογές πολλά μπλοκ προσκομίζονται στην κρυφή μνήμη και δεν επαναχρησιμοποιούνται πριν εκδιωχθούν από αυτή. Συνεπώς αναζήτησαν κάποια λύση, ούτως ώστε τα μπλοκ αυτά να παραμένουν στην κρυφή μνήμη κατά το ελάχιστο δυνατό χρονικό διάστημα. Η λύση που προτάθηκε ήταν η *πολιτική εισαγωγής δύο τρόπων* (Bimodal Insertion Policy), η οποία εισάγει ένα νέο μπλοκ στην LRU θέση και το προάγει στην MRU θέση μόνο αν προσπελαστεί όσο βρίσκεται εκεί, ενώ εισάγει (ενίοτε) με κάποια μικρή πιθανότητα μπλοκ στην MRU θέση. Η λύση αυτή, συνδυάστηκε με την LRU πολιτικής αντικατάστασης και εισήχθη ένας μηχανισμός εναλλαγής μεταξύ MRU και BIP πολιτικής εισαγωγής, που ονομάστηκε *δυναμική δειγματοληψία συνόλων* (Dynamic Set Sampling – DSS). Ο DSS μηχανισμός, αποφασίζει ποιά πολιτική θα επιλεχθεί για όλα τα σύνολα της κρυφής μνήμης, παρακολουθώντας την απόδοση σε ορισμένα σύνολα-ηγέτες που χρησιμοποιούν παγίως μία από τις δύο πολιτικές. Η συνδυασμένη πολιτική εισαγωγής ονομάστηκε *δυναμική πολιτική εισαγωγής* (Dynamic Insertion Policy - DIP).

Ακολούθως οι Kron και συνεργάτες στο (11), παραμένοντας στο πλαίσιο μονοπύρηνων αρχιτεκτονικών επέκτειναν το σχέδιο που πρότειναν οι Qureshi και συνεργάτες, ούτως ώστε να συνδυάσουν την επιλογή της πολιτικής εισαγωγής των μπλοκ με μία νέα πολιτική

προαγωγής των μπλοκ. Το σχέδιο που παρουσίασαν ονομάστηκε Double-DIP και για την υλοποίησή του διατηρήθηκε ο μηχανισμός του DIP και διπλασιάστηκαν τα σύνολα-ηγέτες. Τα μισά από αυτά χρησιμοποιούνται για τη λειτουργία του DIP και τα υπόλοιπα για την επιλογή της πολιτικής προαγωγής. Οι δύο πολιτικές προαγωγής ανάμεσα στις οποίες πραγματοποιείται η επιλογή, είναι η συμβατική πολιτική προαγωγής στην MRU θέση και μία νέα που ονομάστηκε πολιτική αυξανόμενης κατά βήμα προαγωγής (Single-step Incremental Promotion Policy - SIPP). Όπως υποδηλώνει και το όνομά της, η SIPP δεν προάγει το μπλοκ που θα προσπελαστεί στην MRU θέση αλλά κατά μόνο μία θέση στη στοίβα που διατηρεί η LRU πολιτική.

Οι Jaleel και συνεργάτες στο (12) εξέτασαν τον DIP μηχανισμό για πρώτη φορά σε πολυπύρηνες αρχιτεκτονικές και διαπίστωσαν ότι οι αποφάσεις που λαμβάνει δε συνυπολογίζουν τη διαφορετική συμπεριφορά που παρουσιάζουν τα διάφορα εκτελούμενα νήματα. Συνεπώς πρότειναν την ενήμερη ως προς τα νήματα πολιτική δυναμικής εισαγωγής (Thread-Aware Dynamic Insertion Policy). Η πολιτική αυτή συνιστά επέκταση της DIP πολιτικής με κύρια διαφορά το ότι η απόφαση για την επιλογή της κατάλληλης πολιτικής εισαγωγής (οι επιλογές εξακολουθούν να είναι οι MRU και BIP) λαμβάνεται για κάθε εφαρμογή που εκτελείται σε κάποιον πυρήνα του επεξεργαστή. Ο χώρος αναζήτησης του TADIP είναι ένας N-bit πίνακας με 2^N πιθανές αποφάσεις. Στόχος του TADIP είναι να προσδιορίσει δυναμικά το συνδυασμό των bits που παρουσιάζει την καλύτερη απόδοση για το συγκεκριμένο μίγμα φόρτων εργασίας, με χρήση ενός αλγορίθμου γραμμικής πολυπλοκότητας.

2.5.3 Καταμερισμός Κρυφής Μνήμης

Στο (13) οι Qureshi και Patt συμπέραναν ότι η LRU πολιτική αντικατάστασης, δεν καταμερίζει πάντα αποδοτικά τη διαμοιραζόμενη κρυφή μνήμη. Ειδικότερα διαπίστωσαν ότι σε πολλές περιπτώσεις φόρτων εργασίας πολλαπλών εφαρμογών, η LRU πολιτική ουσιαστικά καταμερίζει την κρυφή μνήμη με βάση τη ζήτηση που παρουσιάζει κάθε εκτελούμενη εφαρμογή. Ωστόσο αυτό δεν ευνοεί απαραίτητα τη συνολική απόδοση. Συνεπώς προχώρησαν στην πρόταση ενός σχεδίου καταμερισμού το οποίο για κάθε εφαρμογή ποσοτικοποιεί και καταγράφει το όφελος από τη δέσμευση περισσότερων πόρων και διαμοιράζει την κρυφή μνήμη σύμφωνα με αυτή την πληροφορία. Το σχέδιο αυτό ονομάστηκε *καταμερισμός της κρυφής μνήμης βάσει χρησιμότητας* (Utility-based Cache Partitioning - UCP).

Στο (14) οι Nikas και συνεργάτες, λαμβάνοντας υπόψη την προσπάθεια των Qureshi και Patt για την πρόταση ενός αποδοτικού σχεδίου καταμερισμού της κρυφής μνήμης, προχώρησαν στην παρουσίαση του δικού τους σχεδίου το οποίο ονόμασαν σχέδιο προσαρμοστικού καταμερισμού με χρήση Bloom φίλτρων (Adaptive Bloom Filter Cache Partitioning scheme – ABFCP scheme). Το ABFCP σχέδιο έχει παρόμοια λογική με το UCP, με τη διαφορά ότι ενσωματώνει σχεδιαστικές λύσεις οι οποίες αντιμετωπίζουν εξαρχής το πρόβλημα του κόστους σε υλικό, προκειμένου να μην απαιτηθούν στη συνέχεια σημαντικές παραχωρήσεις στην ευελιξία και την απόδοση.

Όπως και το UCP, το ABFCP σχέδιο εξετάζεται αναλυτικά σε επόμενο κεφάλαιο και αποτελεί τη βάση πάνω στην οποία μελετώνται τροποποιήσεις προκειμένου η παρούσα εργασία να καταλήξει στην πρόταση ενός αποδοτικού σχεδίου καταμερισμού με χαμηλό επιπλέον κόστος σε υλικό (hardware overhead).

Κεφάλαιο 3

Εργαλεία Προσομοίωσης

3.1 Γενικά

Στην αρχιτεκτονική υπολογιστών ένας προσομοιωτής είναι ένα πακέτο λογισμικού που μοντελοποιεί συσκευές ή συστατικά (components) ενός υπολογιστικού συστήματος, προκειμένου να προβλέψει την έξοδο ή διάφορες μετρικές απόδοσης για μια δεδομένη είσοδο. Ένας τέτοιος προσομοιωτής ενδέχεται να μοντελοποιεί από μόνο έναν επεξεργαστή έως ένα ολόκληρο σύστημα που περιλαμβάνει τυπικά έναν σύνολο από επεξεργαστές, κάποιο σύστημα μνήμης και ορισμένες συσκευές εισόδου/εξόδου.

Η έντονη ερευνητική δραστηριότητα στον τομέα της αρχιτεκτονικής υπολογιστών και τα πλεονεκτήματα που παρέχουν οι προσομοιωτές, συνετέλεσαν στην ανάπτυξη πολλών διαφορετικών υλοποιήσεων. Γενικά οι προσομοιωτές αρχιτεκτονικής υπολογιστών ταξινομούνται σε πολλές διαφορετικές κατηγορίες ανάλογα με το χαρακτηριστικό που ενδιαφέρει. Ειδικότερα:

- **Ακρίβεια:** Οι λειτουργικοί (functional) προσομοιωτές δίνουν μεγαλύτερη προτεραιότητα στην ταχύτητα της προσομοίωσης, επιτυγχάνοντας την ίδια λειτουργικότητα με τα συστατικά του συστήματος που μοντελοποιούν. Ένας προσομοιωτής συνόλου εντολών υπάγεται στην κατηγορία των functional simulators και αποτελεί ένα μοντέλο προσομοίωσης που μιμείται τη συμπεριφορά μιας συσκευής ή ενός συστατικού (component) κάποιου συστήματος παρέχοντας την ίδια λειτουργικότητα και τα ίδια αποτελέσματα, χωρίς ωστόσο να δίνεται ιδιαίτερη έμφαση στην ακρίβεια με την οποία λειτουργεί το μοντέλο εσωτερικά. Για παράδειγμα ένας επεξεργαστής δύναται να υλοποιηθεί σε έναν functional simulator, διαβάζοντας εντολές και διατηρώντας εσωτερικές μεταβλητές που αντιπροσωπεύουν τους καταχωρητές του επεξεργαστή. Από την άλλη, οι προσομοιωτές χρονισμού (timing simulators) δίνουν έμφαση στην ακριβή αναπαραγωγή των χαρακτηριστικών της απόδοσης/χρονισμού των συστημάτων που μοντελοποιούν.

- **Εύρος:** Η προσομοίωση μπορεί να περιλαμβάνει από μόνο έναν επεξεργαστή (micro-architecture simulator) έως ένα ολόκληρο σύστημα (full-system simulator). Συχνά, ένας προσομοιωτής πλήρους συστήματος όπως ο Simics (15), περιλαμβάνει ορισμένους προσομοιωτές συνόλου εντολών για να προσομοιώσει συσκευές όπως οι επεξεργαστές, που αποτελούν τμήμα του συστήματος.
- **Είσοδος:** Οι trace-driven (ή event-driven) προσομοιωτές χρησιμοποιούν κατά τη λειτουργία τους ίχνη ή γεγονότα (traces/events). Τα ίχνη ή γεγονότα είναι εκ των προτέρων καταγεγραμμένες ροές εντολών που αντιστοιχούν σε κάποια συγκεκριμένη είσοδο, επομένως δε χρειάζεται η μοντελοποίηση και προσομοίωση των επεξεργαστικών στοιχείων. Τέτοιοι προσομοιωτές στηρίζονται σε υπάρχοντα εργαλεία για να συλλέξουν τα ίχνη διευθύνσεων μνήμης και να τα καταγράψουν σε κάποιο αρχείο για μετέπειτα χρήση. Αντίθετα οι καθοδηγούμενοι από την εκτέλεση προσομοιωτές (execution-driven simulators), στηρίζονται σε λειτουργικά μοντέλα για να εκτελέσουν το δυαδικό κώδικα μιας εφαρμογής. Οι διευθύνσεις μνήμης που παράγονται από το λειτουργικό μοντέλο τροφοδοτούνται σε πραγματικό χρόνο σε έναν προσομοιωτή συστημάτων μνήμης που διαμορφώνεται μέσα στο λειτουργικό μοντέλο. Γενικά, η trace-driven προσομοίωση είναι πιο γρήγορη και έχει γίνει δημοφιλής για την πραγματοποίηση μελετών σε υπό σχεδίαση συστήματα. Από την άλλη μεριά, απαιτεί συχνά περίπλοκα εργαλεία για την απόκτηση του trace από την προς μελέτη αρχιτεκτονική συνόλου εντολών, καθώς και τη μοντελοποίηση της τελευταίας με ακρίβεια.

Οι προσομοιωτές αρχιτεκτονικής έχουν καταστεί αναγκαία εργαλεία της έρευνας στον τομέα της αρχιτεκτονικής υπολογιστών κυρίως λόγω των παρακάτω πλεονεκτημάτων τους:

- Προσφέρουν τη δυνατότητα αξιολόγησης διαφορετικών σχεδίων υλικού χωρίς να απαιτείται η κατασκευή ακριβών πραγματικών συστημάτων, καθώς και την ευκαιρία προσέγγισης μη-υπαρχόντων υπολογιστικών συστημάτων και συστατικών.
- Επιτρέπουν τη συλλογή λεπτομερών μετρικών απόδοσης. Συχνά μία και μόνο εκτέλεση σε έναν προσομοιωτή δύναται να δημιουργήσει ένα μεγάλο σύνολο δεδομένων απόδοσης.
- Καθιστά πολύ ευκολότερη τη διόρθωση και απομάκρυνση λαθών. Σε πραγματικά συστήματα αυτό συνήθως απαιτεί επανεκκίνηση και επανεκτέλεση του κώδικα προκειμένου να αναπαραχθούν τα ίδια λάθη-προβλήματα. Αντίθετα, πολλοί

προσομοιωτές έχουν ένας πλήρως ελεγχόμενο περιβάλλον και επιτρέπουν στους μηχανικούς λογισμικού να εκτελούν τον κώδικα προς τα πίσω, όταν ανιχνευθεί κάποιο σφάλμα.

Όσον αφορά το θέμα της απόδοσης που παρουσιάζουν οι προσομοιωτές στη μοντελοποίηση των αρχιτεκτονικών, αυτό υπογραμμίστηκε από την αύξηση του μεγέθους και της πολυπλοκότητας των φόρτων εργασίας που χρησιμοποιούνται ως είσοδοι. Η απόδοση ενός προσομοιωτή δύναται να ποσοτικοποιηθεί με χρήση του παράγοντα επιβράδυνσης (slowdown factor). Για τυπικές υλοποιήσεις προσομοιωτών έχουν καταγραφεί παράγοντες επιβράδυνσης από 10 έως 100, ανάλογα με τον τύπο του, την εφαρμογή εισόδου και τη διαμόρφωση της αρχιτεκτονικής που προσομοιώνεται. Στις χειρότερες περιπτώσεις, ένας προσομοιωτής που πραγματοποιεί λεπτομερή προσομοίωση ανά κύκλο ενδέχεται να παρουσιάζει παράγοντα επιβράδυνσης της τάξης του 2000. Η απόδοση ενός προσομοιωτή και η καθυστέρηση που επιφέρει στην εκτέλεση, αποτελούν σήμερα έναν από τους καθοριστικότερους παράγοντες για την επιλογή ή την απόρριψή του.

3.2 Στατιστικές Μέθοδοι

Οι αρχιτέκτονες υπολογιστών, ιδανικά βασίζονται σε λεπτομερείς προσομοιώσεις για να αξιολογήσουν χαρακτηριστικά μελλοντικών αρχιτεκτονικών σχεδίων. Για το λόγο αυτό είναι πολύ σημαντικό να επιλεγούν φόρτοι εργασίας που αντιπροσωπεύουν την αυξανόμενη πολυπλοκότητα των πολλαπλών φόρτων εργασίας που θα κληθούν να εκτελέσουν στην πράξη οι πολυπύρηνοι επεξεργαστές. Ωστόσο, η λεπτομερής μοντελοποίηση της απόδοσης των εμπορικών εφαρμογών αποδεικνύεται ότι είναι δύσκολη υπόθεση.

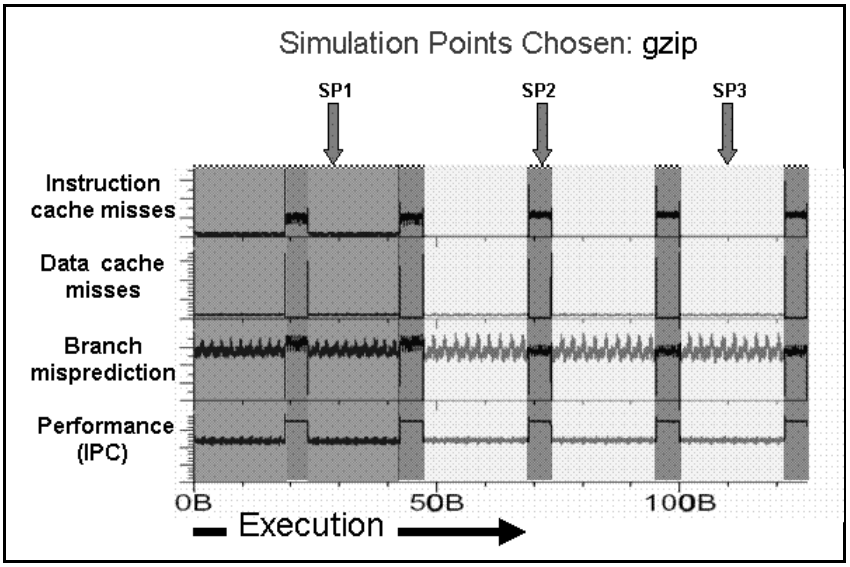
3.2.1 Η Μεθοδολογία του SimPoint

Ο αυξημένος χρόνος πλήρους εκτέλεσης των μετροπρογραμμάτων, ως αποτέλεσμα των μεγαλύτερων ιχνών δεδομένων και των αυξημένων δυναμικών πληθών εντολών, καθιστούν ασύμφορη τη λεπτομερή προσομοίωση τους σε ακριβή μοντέλα (προσομοίωσης). Προκειμένου να επιλυθεί αυτό, οι αρχιτέκτονες επεξεργαστών επιλέγουν μόνο ένα μικρό τμήμα της εκτέλεσης κάθε μετροπρογράμματος για την πραγματοποίηση των προσομοιώσεων. Για το λόγο αυτό έχουν προταθεί στατιστικές μέθοδοι, οι οποίες χρησιμοποιώντας ένα μικρό κομμάτι κάθε εφαρμογής, επιτρέπουν την εξαγωγή ασφαλών

συμπερασμάτων για τη συνολική συμπεριφορά τους εισάγοντας ένα μικρό μόνο σφάλμα. Ωστόσο και η επιλογή αντιπροσωπευτικών τμημάτων των εφαρμογών είναι δύσκολη, διότι η συμπεριφορά του προγράμματος μεταβάλλεται σε συνάρτηση με το χρόνο.

Η μεθοδολογία του SimPoint όπως την περιέγραψαν οι Sherwood και συνεργάτες στο (16), προτείνει έναν τρόπο αναγνώρισης επαναλαμβανόμενων μοτίβων στην εκτέλεση των εφαρμογών και επιλογής αντιπροσωπευτικών τμημάτων αυτών. Ο αλγόριθμος του SimPoint διαιρεί την εκτέλεση του προγράμματος σε διαστήματα με ίσο πλήθος εντολών και τα κατηγοριοποιεί ως προς τη συμπεριφορά που παρουσιάζουν, βάσει της ταχύτητας με την οποία εκτελούνται. Μία *φάση* (phase) της εκτέλεσης είναι μια συλλογή από διαστήματα με παρόμοια συμπεριφορά. Από κάθε φάση επιλέγεται ένα αντιπροσωπευτικό τμήμα το οποίο ονομάζεται *σημείο προσομοίωσης* (simulation point).

Στο παρακάτω σχήμα παρουσιάζεται η πλήρης εκτέλεση του gzip από τη σουίτα Standards Performance Evaluation Corporation (SPEC) CPU2000 (17) και οι τρεις φάσεις εκτέλεσης που προσδιόρισε ο αλγόριθμος ταξινόμησης φάσεων του SimPoint. Το τελευταίο βήμα του αλγορίθμου είναι να επιλεγεί από κάθε φάση εκτέλεσης το διάστημα που είναι πλησιέστερο στο κέντρο αυτής και το οποίο θα θεωρηθεί ως σημείο προσομοίωσης για τη φάση. Στο σχήμα, τα SP1, SP2 και SP3 αντιπροσωπεύουν τα τρία σημεία που επιλέγονται.



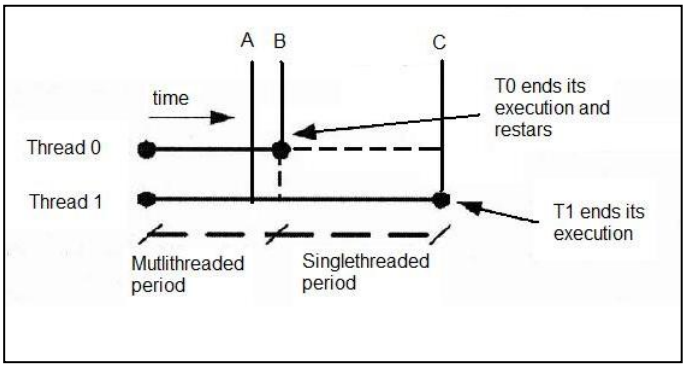
Σχήμα 3.1: Επιλογή σημείων προσομοίωσης για το gzip (18)

Ένας τρόπος για την εξαγωγή τέτοιων σημείων προσομοίωσης είναι η χρήση του εργαλείου PinPoints της Intel (19). Η ακρίβεια της μεθοδολογίας των PinPoints εξαρτάται σε μεγάλο

βαθμό από τον αριθμό των ξεχωριστών φάσεων εκτέλεσης που επιχειρείται να προσδιοριστεί κάθε φορά. Επιπλέον σημεία προσομοίωσης από τα απαραίτητα συντελούν σε μεγαλύτερους χρόνους προσομοίωσης, ενώ αντίθετα πολύ λίγα επιλεγμένα σημεία συνεπικουρούν στην εμφάνιση σημαντικών σφαλμάτων. Όπως προσδιόρισαν οι Arun και Lizy στο (20), παρόλο που τα μετροπρογράμματα της σουίτας SPEC CPU2006 έχουν τουλάχιστον μία τάξη μεγέθους μεγαλύτερα πλήθη εντολών από τα αντίστοιχα της σουίτας SPEC CPU2000, απαιτούν παραπλήσιο αριθμό σημείων προσομοίωσης (ίδιου μεγέθους) για να προσομοιωθούν με ακρίβεια.

3.2.2 Η Μεθοδολογία του FAME

Η έλευση των πολυπύρηνων επεξεργαστών συνετέλεσε στην παρουσίαση ορισμένων νέων μεθοδολογιών και μετρικών για την αξιολόγηση της απόδοσής τους. Μια μεθοδολογία ορίζει μεταξύ άλλων, σε ποιά χρονική στιγμή της εκτέλεσης πρέπει να ληφθούν οι απαραίτητες μετρήσεις, οι οποίες στη συνέχεια θα συνδυαστούν με κατάλληλο τρόπο για να προκύψουν οι μετρικές και τα αποτελέσματα της αξιολόγησης. Υπάρχουν διάφορες τέτοιες μεθοδολογίες που υποδεικνύουν τα σημεία στα οποία πρέπει να ληφθούν μετρήσεις σύμφωνα με κάποιο κριτήριο.



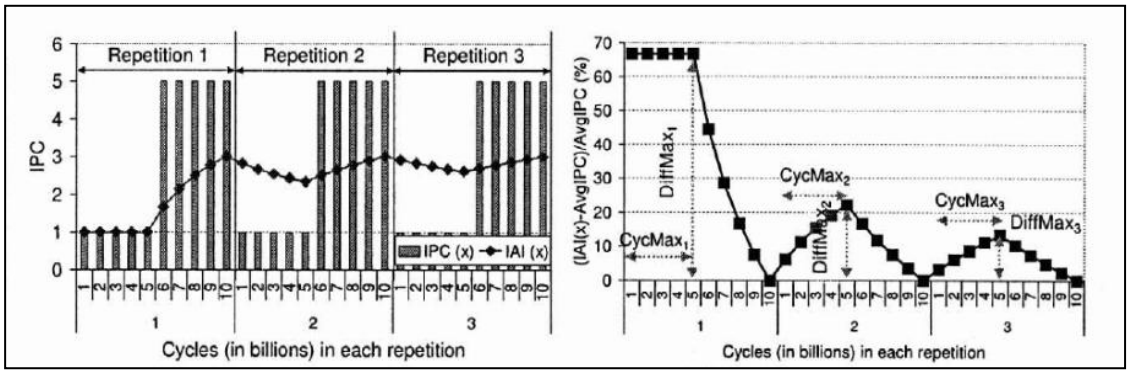
Σχήμα 3.2: Μεθοδολογίες καθορισμού του τέλους της προσομοίωσης

Μέχρι πρόσφατα οι υπάρχουσες μεθοδολογίες καθορισμού της διάρκειας και άρα του τέλους των προσομοιώσεων περιελάμβαναν τη μεθοδολογία του *πρώτου* (First methodology), τη μεθοδολογία του *τελευταίου* (Last methodology) και τη μεθοδολογία του *σταθερού πλήθους εντολών* (Fixed Instructions methodology). Η First μεθοδολογία τερματίζει την εκτέλεση του φόρτου εργασίας όταν οποιοδήποτε εφαρμογή του φόρτου εργασίας ολοκληρώσει την εκτέλεσή της, όπως φαίνεται στο σχήμα 3.2, στο σημείο B.

Αντίθετα η Last μεθοδολογία τερματίζει την εκτέλεση του φόρτου εργασίας μόνο αφότου όλες οι εφαρμογές έχουν εκτελεστεί τουλάχιστον μία φορά (σημείο C στο σχήμα 3.2). Η Fixed Instructions μεθοδολογία τερματίζει την προσομοίωση όταν το πλήθος εντολών που έχουν εκτελεστεί ξεπεράσει ένα κατώφλι, το οποίο ορίζεται συνήθως ως ένας σταθερός αριθμός εντολών επί το πλήθος των εφαρμογών του φόρτου εργασίας (σημείο A στο σχήμα 3.2). Πρόσφατες έρευνες υπέδειξαν ότι οι μετρήσεις που χρησιμοποιούνται στις μετρικές ειδικά για την περίπτωση των πολυνηματικών αρχιτεκτονικών, είναι συχνά μη αντιπροσωπευτικές με αποτέλεσμα οι νέες σχεδιαστικές προτάσεις να μην αξιολογούνται πάντα σωστά.

Οι Vera και συνεργάτες πρότειναν στο (21), τη χρήση μιας μεθοδολογίας διεξαγωγής μετρήσεων για πολυνηματικές αρχιτεκτονικές, την οποία ονόμασαν μεθοδολογία *δίκαιης μέτρησης πολυνηματικών αρχιτεκτονικών* (FAirly Measuring Multithreaded Architectures – FAME). Η FAME μεθοδολογία αποσκοπεί στην κατά το δυνατόν δικαιότερη μέτρηση της απόδοσης των πολυνηματικών αρχιτεκτονικών και στηρίζεται στην επαναληπτική εκτέλεση των εφαρμογών που συνιστούν το φόρτο εργασίας των νημάτων.

Οι ερευνητές χρησιμοποίησαν τη μεθοδολογία του SimPoint για να λάβουν ίχνη εκτέλεσης (execution traces) από τα μετροπρογράμματα της σουίτας SPEC CPU2000. Δεδομένου ότι τα ίχνη σε έναν φόρτο εργασίας πολλαπλών εφαρμογών δεν εκτελούνται με την ίδια ταχύτητα, οι Vera και συνεργάτες συμπέραναν πως είναι απίθανο να ολοκληρώσουν την εκτέλεσή τους συγχρόνως. Για το λόγο αυτό η μεθοδολογία του FAME προσδιορίζει το πλήθος των φορών που πρέπει να εκτελεστεί το κάθε ίχνος προκειμένου να εκπροσωπείται δίκαια στις τελικές μετρήσεις. Ο προσδιορισμός απαιτεί την εκτέλεση κάθε ίχνους σε απομόνωση (χωρίς να εκτελούνται και τα υπόλοιπα παράλληλα) και τη δειγματοληψία του IPC του σε τακτά διαστήματα κατά την εκτέλεση αυτή, όπως φαίνεται στο σχήμα 3.2.



Σχήμα 3.3: Δειγματοληψία IPC σε απομόνωση (21)

Ορίζοντας ένα κατώφλι μέγιστης επιτρεπτής διαφοράς του IPC (Maximum Allowable IPC Variance – MAIV) το ελάχιστο απαιτούμενο πλήθος εκτελέσεων για κάθε ίχνος, δίνεται από τον παρακάτω τύπο:

$$i \geq \left\lceil \frac{(CycleMax_2 - 2 * TotalCycles) * \big(FinalIPC * (1 + MAIV)\big) - InstrMax_2 + 2 * TotalInst}{TotalCycles * \big(FinalIPC * (1 + MAIV)\big) - TotalInst} \right\rceil$$

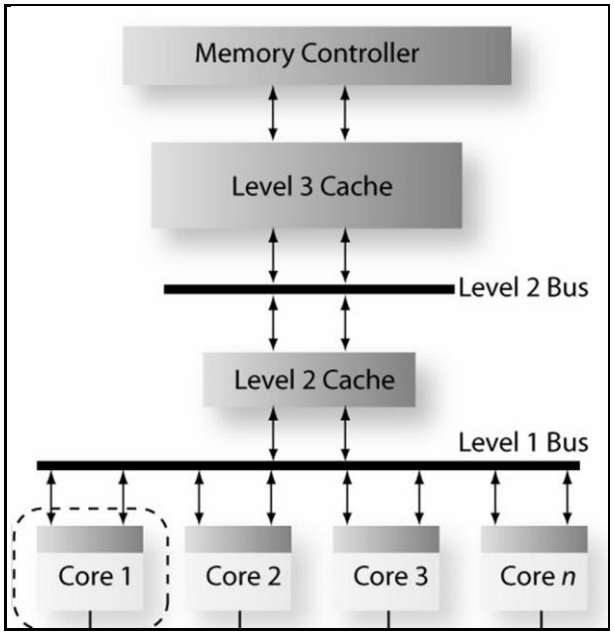
όπου $InstMax_2$ είναι η εντολή της δεύτερης επανάληψης όπου η διαφορά γίνεται μέγιστη ανάμεσα στο μέσο και στο τελικό IPC αυτής της επανάληψης και $CycleMax_2$ είναι ο κύκλος στον οποίο εκτελείται η εντολή αυτή. Επίσης TotalInst και TotalCycle είναι οι συνολικοί αριθμοί εντολών και κύκλων για κάθε επανάληψη.

3.3 Trace-driven Simulation

Στο υποκεφάλαιο αυτό γίνεται μια σύντομη αναφορά στην αρχιτεκτονική που μοντελοποιεί ο προσομοιωτής που αναπτύχθηκε και χρησιμοποιήθηκε για την αξιολόγηση των σχεδίων που μελετήθηκαν και προτάθηκαν. Ο προσομοιωτής ονομάστηκε jsimTracer και αποτελεί τροποποίηση μιας ήδη υπάρχουσας υλοποίησης που προέκυψε από την εργασία των Hornsell και Nikas (14). Όπως διαμορφώθηκε για τους σκοπούς αυτής της εργασίας, ο jsimTracer αποτελεί έναν *trace-driven cycle-level simulator*.

3.3.1 Η Αρχιτεκτονική που Μοντελοποιήθηκε

Η αρχιτεκτονική Jamaica (JAva Machine And Integrated Circuit Architecture) συνιστά ένα μοντέλο ενός CMP, του οποίου τα χαρακτηριστικά ακολουθούν τις σύγχρονες σχεδιαστικές τάσεις στον τομέα κατασκευής πολυπύρηνων συστημάτων. Καταρχήν προτάθηκε από τον Wright (22) και (στην αρχική της μορφή) περιελάμβανε N πυρήνες ενσωματωμένους στο ίδιο chip, οι οποίοι μοιράζονταν μια κρυφή μνήμη δευτέρου επιπέδου. Αργότερα ο Horsnell επέκτεινε το σχέδιο του Wright, προκειμένου να ενσωματώσει στο μοντέλο του επεξεργαστή μια πολυεπίπεδη ιεραρχία κρυφής μνήμης.



Σχήμα 3.4: Η αρχιτεκτονική Jamaica (22)

Η ιεραρχία μνήμης έχει υλοποιηθεί με τέτοιο τρόπο ώστε να εφαρμόζει μια οκνηρή πολιτική δέσμευσης (lazy allocation) της κρυφής μνήμης. Στην περίπτωση μιας αστοχίας στην κρυφή μνήμη, ένα αίτημα αναμεταδίδεται μέσα στο δίκτυο χωρίς να δεσμεύεται κάποιο μπλοκ για να αποθηκεύσει την απάντηση αυτού του αιτήματος. Αυτό συμβαίνει για καθένα από τα διαδοχικά επίπεδα της ιεραρχίας και κανένα μπλοκ δε δεσμεύεται έως ότου ληφθεί μια συναλλαγή απάντησης. Μετά τη λήψη της απάντησης από την κρυφή μνήμη, χρησιμοποιείται η LRU πολιτική αντικατάστασης για να καθορίσει ποιό μπλοκ μνήμης θα εκδιωχθεί, προκειμένου να ελευθερωθεί χώρος για την καινούρια καταχώρηση.

Η αρχιτεκτονική Jamaica παρουσιάζει πολλές ομοιότητες με πολλά ήδη υπάρχοντα CMP συστήματα πολυεπεξεργαστών. Για παράδειγμα οι επεξεργαστές της σειράς Dunnington της Intel (Intel Xeon 7400 Series), έχουν 4 έως 6 πυρήνες καθένας εκ των οποίων έχει ιδιωτικές κρυφές μνήμες πρώτου επιπέδου για εντολές και δεδομένα, μεγέθους 32Kbytes. Κάθε ζεύγος πυρήνων μοιράζεται μέσω ενός διαδρόμου (bus) μια κρυφή μνήμη δευτέρου επιπέδου μεγέθους 3Mbytes, η οποία χρησιμοποιείται για εντολές και δεδομένα. Επιπλέον, υπάρχει ένα τρίτο επίπεδο κρυφής μνήμης ενσωματωμένο στο chip, το οποίο περιλαμβάνει 12 έως 16 Mbytes μνήμης που διαμοιράζονται ανάμεσα σε όλους τους πυρήνες και πάλι με τη χρήση διαδρόμου (bus).

3.3.2 Υλοποίηση του Προσομοιωτή

Η υλοποίηση του προσομοιωτή jsimTracer στηρίχθηκε στην πλατφόρμα προσομοίωσης του jamsim (23), η οποία είναι γραμμένη σε Java. Η πλατφόρμα αυτή υποστηρίζει δύο τρόπους-μοντέλα προσομοίωσης, το λειτουργικό (functional) και την προσομοίωση σε επίπεδο κύκλων (cycle-level simulation). Για τις προσομοιώσεις που πραγματοποιήθηκαν στα πλαίσια αυτής της εργασίας χρησιμοποιήθηκε η cycle-level προσομοίωση και οι τροποποιήσεις που εφαρμόστηκαν αφορούν το αντίστοιχο τμήμα του jamsim προσομοιωτή, δηλαδή τις κλάσεις που μοντελοποιούν τη λειτουργία της ιεραρχία μνήμης σε επίπεδο κύκλων.

Η πρώτη τροποποίηση ήταν η μετατροπή του jamsim από event-driven σε trace-driven simulator. Για το σκοπό αυτό αφαιρέθηκαν τα μοντέλα των επεξεργαστών και αντικαταστάθηκαν από άλλα τα οποία τροφοδοτούνται από αρχεία με ίχνη πρόσβασης στη μνήμη (memory access traces) που παρέχονταν ως είσοδος. Η δεύτερη τροποποίηση αφορά την ενσωμάτωση της μεθοδολογίας FAME στον προσομοιωτή. Συγκεκριμένα, προστέθηκε η δυνατότητα ρύθμισης του προσομοιωτή ώστε να εκτελεί κάθε ίχνος που αντιστοιχεί σε ένα νήμα, τουλάχιστον κατά έναν προκαθορισμένο αριθμό φορών. Το τελευταίο απαιτούσε την επανεκκίνηση των νημάτων που έφταναν στο τέλος της εκτέλεσής τους, όσες φορές ήταν απαραίτητο.

3.4 Μεθοδολογία Πειραματικών Μετρήσεων

Τα ίχνη που καταγράφηκαν και χρησιμοποιήθηκαν ως είσοδος του προσομοιωτή αποτελούν ίχνη πρόσβασης στη μνήμη ενός x86 επεξεργαστή. Για την απόκτησή τους χρησιμοποιήθηκε ο Simics (15) με διαμόρφωση για αρχιτεκτονική x86. Η καταγραφή τους στηρίχθηκε στο Pin (24), καθώς και σε ένα module που αναπτύχθηκε αποκλειστικά για το σκοπό αυτό. Συνολικά η μεθοδολογία που ακολουθήθηκε για την πραγματοποίηση των πειραματικών μετρήσεων συνοψίζεται στα παρακάτω βήματα:

- Επιλογή του full-system simulator (Simics) και της αρχιτεκτονικής του εικονικού συστήματος που θα μοντελοποιούσε αυτός (x86), με κριτήριο το εύρος εφαρμογής στη βιομηχανία.
- Χρήση της μεθοδολογίας των PinPoints (19) στο περιβάλλον του Simics για τον προσδιορισμό αντιπροσωπευτικών τμημάτων εκτέλεσης των μετροπρογραμμάτων

της σουίτας SPEC CPU2006. Τα τμήματα που λήφθηκαν έχουν μέγεθος 1 δισεκατομμύριο εντολές για κάθε μετροπρόγραμμα.

- Συλλογή των ιχνών πρόσβασης στη μνήμη μόνο για τα αντιπροσωπευτικά τμήματα εκτέλεσης που προσδιορίστηκαν νωρίτερα. Η καταγραφή των memory access traces έγινε στο περιβάλλον του Simics και με χρήση ενός module που αναπτύξαμε ειδικά για το σκοπό αυτό. Το module παρακολουθεί την εκτέλεση του μετροπρογράμματος και όταν η εκτέλεση φτάνει σε κάποια από τις αντιπροσωπευτικές περιοχές, ενεργοποιεί το μηχανισμό καταγραφής των προσβάσεων στη μνήμη ο οποίος αποθηκεύει τα ίχνη στο δίσκο. Ο μηχανισμός, απενεργοποιείται όταν η εκτέλεση εξέρχεται του αντιπροσωπευτικού διαστήματος.
- Χρήση της μεθοδολογίας του FAME για τον προσδιορισμό του ελάχιστου πλήθους φορών που έπρεπε να εκτελεστεί το κάθε ίχνος. Στα πλαίσια αυτής της εργασίας ως ανώτατο επιτρεπτό σφάλμα (MAIV) επιλέχθηκε το 5%, το οποίο θεωρήθηκε ικανοποιητικό από άποψη ακρίβειας και συγχρόνως δεν απαιτούσε υπερβολικό αριθμό επαναλήψεων. Η προσομοίωση της απομονωμένης εκτέλεσης έγινε στο περιβάλλον του jsimTracer.
- Εισαγωγή των καταγεγραμμένων ιχνών στον cycle-level προσομοιωτή της ιεραρχίας μνήμης (jsinTracer) σε διάφορους συνδυασμούς προκειμένου να προσομοιωθεί ένα μεγάλο εύρος φόρτων εργασίας πολλαπλών εφαρμογών (multiple application workloads).

Κεφάλαιο 4

Μετροπρογράμματα

4.1 Γενικά

Η κατανόηση των απαιτήσεων σε πόρους μνήμης των σύγχρονων εφαρμογών συνιστά το πρώτο βήμα προς τη σχεδίαση αποδοτικών συστημάτων μνήμης. Επιπλέον οι απαιτήσεις πρέπει να μελετηθούν τόσο ατομικά για κάθε εφαρμογή, όσο και σε περιπτώσεις που πολλές εφαρμογές ανταγωνίζονται για κοινούς πόρους μνήμης σε ένα πολυπύρηνο σύστημα. Οι σουίτες μετροπρογραμμάτων αποτελούν πλέον την de facto μέθοδο μελέτης αυτών των απαιτήσεων αλλά και αξιολόγησης των ερευνητικών ιδεών και υλοποιήσεων στον τομέα της αρχιτεκτονικής υπολογιστών. Οι σουίτες αυτές ανανεώνονται συχνά προκειμένου να συνιστούν ένα αντιπροσωπευτικό παράδειγμα του υπολογιστικού φορτίου και των απαιτήσεων που παρουσιάζουν οι σύγχρονες εφαρμογές.

Μία πολύ δημοφιλής σουίτα μετροπρογραμμάτων είναι η SPEC CPU2006 (25). Πρόκειται για την τελευταία έκδοση της σουίτας SPEC CPU της εταιρείας προτύπων αξιολόγησης απόδοσης (Standards Performance Evaluation Corporation) και αποτελεί σημείο αναφοράς των ερευνητών αλλά και της βιομηχανίας υπολογιστικών συστημάτων για την αξιολόγηση αρχιτεκτονικών σχεδίων. Περιέχει εφαρμογές που πιέζουν ιδιαίτερα τον επεξεργαστή, το υποσύστημα μνήμης και το μεταγλωττιστή ενός συστήματος, με άλλα λόγια τα κυριότερα συστατικά των σύγχρονων συστημάτων που καθορίζουν την απόδοση των εφαρμογών.

Η SPEC σχεδίασε τη σουίτα CPU2006 με σκοπό να παρέχει ένα μέτρο σύγκρισης της υπολογιστικής απόδοσης σε ένα ευρύτατο φάσμα αρχιτεκτονικών και υλικού, χρησιμοποιώντας σε κάθε περίπτωση πραγματικές-υπάρχουσες εφαρμογές. Περιλαμβάνει 12 μετροπρογράμματα ακέραιης αριθμητικής και 17 μετροπρογράμματα αριθμητικής κινητής υποδιαστολής. Όλα τα μετροπρογράμματα έχουν σε γενικές γραμμές πλήθη εντολών (instruction counts) τουλάχιστον μία τάξη μεγέθους μεγαλύτερα από τα αντίστοιχα πλήθη εντολών των μετροπρογραμμάτων της σουίτας SPEC CPU2000 και σημαντικά αυξημένα ίχνη δεδομένων (data footprint). Επιπλέον, το ίδιο ισχύει και για τα μεγέθη συνόλου εργασίας και τα πιο απαιτητικά σε μνήμη από τα μετροπρογράμματα, απαιτούν

περισσότερα από 4 Mbytes κρυφής μνήμης για να εκτελεστούν αποδοτικά. Στο κεφάλαιο αυτό παρουσιάζονται αποτελέσματα από προσομοιώσεις που πραγματοποιήθηκαν με σκοπό τη μελέτη των απαιτήσεων σε μνήμη, 18 μετροπρογραμμάτων της συλλογής SPEC CPU2006.

4.2 Σύντομη Περιγραφή

Στο υποκεφάλαιο αυτό επιχειρείται μια σύντομη περιγραφή της λειτουργίας, των μετροπρογραμμάτων της σουίτας SPEC CPU2006 τα οποία χρησιμοποιήθηκαν στις προσομοιώσεις. Η περιγραφή χωρίζεται σε δύο μέρη, ένα για τα μετροπρογράμματα ακέραιης αριθμητικής και ένα για τα μετροπρογράμματα αριθμητικής κινητής υποδιαστολής, καθώς αυτό το διαχωρισμό ακολουθεί και η SPEC.

4.2.1 SPEC CINT2006

403.gcc

Το 403.gcc είναι βασισμένο στην έκδοση 3.2 του γνωστού C μεταγλωττιστή gcc και στη συγκεκριμένη περίπτωση παράγει κώδικα για έναν επεξεργαστή AMD Opteron. Το πρόγραμμα τρέχει ως μεταγλωττιστής με πολλές από τις σημαίες βελτιστοποιήσεων ενεργοποιημένες. Υπάρχουν εννέα διαφορετικές είσοδοι οι οποίες είναι προεπεξεργασμένα αρχεία κώδικα C και στην περίπτωση αυτή χρησιμοποιείται το 200.i. Όλα τα αρχεία εξόδου είναι αρχεία x86-64 συμβολικής γλώσσας.

429.mcf

Το 429.mcf είναι ένα μετροπρόγραμμα που προέκυψε από το MCF, ένα πρόγραμμα που χρησιμοποιείται για δρομολόγηση μέσων μαζικής μεταφοράς. Ο αλγόριθμος δικτύων simplex που χρησιμοποιείται είναι ειδική έκδοση του αντίστοιχου για την επίλυση προβλημάτων ροής. Το πρόγραμμα είναι γραμμένο σε κώδικα C και χρησιμοποιεί σχεδόν αποκλειστικά ακέραια αριθμητική. Στην έκδοση για τη σουίτα CPU2006, επιλέχθηκαν νέα αρχεία εισόδου με στόχο μεγαλύτερους χρόνους εκτέλεσης. Το 429.mcf απαιτεί περίπου 860MB και 1700MB για 32bit και ένα 64bit μοντέλο, αντίστοιχα.

445.gobmk

Το μετροπρόγραμμα gobmk είναι ένα C πρόγραμμα τεχνητής νοημοσύνης για παιχνίδια, το οποίο παίζει Go και εκτελεί ένα σύνολο εντολών για να αναλύσει θέσεις Go. Η

περισσότερη είσοδος είναι σε μορφή SmartGo, μια ευρέως χρησιμοποιούμενη και τυπική αναπαράσταση των Go παιχνιδιών. Ένα τυπικό τεστ περιλαμβάνει την ανάγνωση ενός σημείου στο παιχνίδι και κατόπιν την εκτέλεση μιας εντολής για την ανάλυση της θέσης. Η έξοδος είναι τυπικά μια ascii περιγραφή μιας ακολουθίας Go κινήσεων.

456.hmmer

Το hmmer μετροπρόγραμμα είναι γραμμένο σε γλώσσα C. Τα Μαρκοβιανά μοντέλα κρυμμένων προφίλ (Profile Hidden Markov Models) είναι στατιστικά μοντέλα ευθυγράμμισης πολλαπλών ακολουθιών, που χρησιμοποιούνται στην υπολογιστική βιολογία για να αναζητούν πρότυπα μέσα σε ακολουθίες DNA. Η τεχνική χρησιμοποιείται για να κάνει αναζήτηση βάσεων δεδομένων, χρησιμοποιώντας στατιστικές περιγραφές μιας οικογένειας ακολουθιών. Χρησιμοποιείται για ανάλυση ακολουθιών πρωτεϊνών.

458.sjeng

Το μετροπρόγραμμα 458.sjeng είναι βασισμένο στο Sjeng11.2, ένα πρόγραμμα που παίζει σκάκι και διάφορες παραλλαγές του. Προσπαθεί να βρει την καλύτερη κίνηση μέσω ενός συνδυασμού αναζητήσεων δέντρων. Πρακτικά, εξερευνεί το δέντρο των παραλλαγών που προκύπτει από μια δεδομένη θέση και για δεδομένο βάθος, επεκτείνοντας τις ενδιαφέρουσες παραλλαγές και απορρίπτοντας αμφισβητήσιμες ή άσχετες. Από αυτό το δέντρο καθορίζεται η βέλτιστη πορεία παιχνιδιού και για τους δύο παίκτες, καθώς επίσης και ένα αποτέλεσμα που απεικονίζει την ισορροπία δυνάμεων μεταξύ των δύο.

462.libquantum

Το libquantum είναι μια βιβλιοθήκη για την προσομοίωση ενός κβαντικού υπολογιστή. Επίσης, μια εφαρμογή του αλγορίθμου παραγοντοποίησης του Shor συμπεριλαμβάνεται στο libquantum. Το μετροπρόγραμμα αναμένει τον προς παραγοντοποίηση αριθμό ως παράμετρο και δίνει ως έξοδο μια συνοπτική εξήγηση αυτού που κάνει καθώς και τους παράγοντες του αριθμού που εισήχθηκε εάν η παραγοντοποίηση ήταν επιτυχής.

464.h264ref

Το 464.h264ref είναι μια υλοποίηση αναφοράς του .264/AVC (Advanced Video Coding) προτύπου συμπίεσης. Το πρότυπο έχει αναπτυχθεί από τις ομάδες VCEG (Video Coding Experts Group) της ITU (International Telecommunications Union) και MPEG (Moving Pictures Experts Group) του οργανισμού ISO/IEC (International Standardization Organization). Ο πηγαίος κώδικας που αποτελεί τμήμα του μετροπρογράμματος της

σουίτας SPEC CPU2006, έχει βασιστεί στην έκδοση 9.3 της υλοποίησης αναφοράς h264avc. Ως είσοδος χρησιμοποιείται μη συμπιεσμένο βίντεο μορφής YUV.

471.omnetpp

Το omnetpp μετροπρόγραμμα διεξάγει προσομοίωση διακριτών γεγονότων ενός μεγάλου δικτύου Ethernet, η οποία βασίζεται στο σύστημα προσομοίωσης διακριτών γεγονότων OMNeT++, ένα γενικό και ανοικτό πλαίσιο προσομοίωσης που προορίζεται κυρίως για προσομοίωση δικτύων επικοινωνιών. Για το φόρτο εργασίας αναφοράς (reference workload), το δίκτυο που προσομοιώνεται είναι το μοντέλο ενός δικτύου κορμού Ethernet ενός campus, με διάφορα μικρότερα LAN κρεμασμένα από κάθε switch του δικτύου κορμού. Η έξοδος αποτελείται από αναλυτικά στατιστικά όπως ο αριθμός πλαισίων που εστάλησαν, παραλήφθηκαν, απορρίφθηκαν κτλ.

473.astar

Το 473.astar προέκυψε από μια φορητή βιβλιοθήκη εύρεσης δισδιάστατων μονοπατιών που βρίσκει εφαρμογή στην τεχνητή νοημοσύνη παιχνιδιών. Υλοποιεί τρείς διαφορετικούς αλγορίθμους αναζήτησης: τον A* για χάρτες με προσπελάσιμα/απροσπέλαστα εδάφη, μια παραλλαγή του για χάρτες με διαφορετικούς τύπους εδαφών και ταχύτητες κίνησης, καθώς και μια υλοποίησή του για γράφους που δημιουργούνται από περιοχές χαρτών με γειτονική σχέση. Η είσοδος είναι ένας χάρτης σε δυαδική μορφή και η έξοδος είναι το πλήθος των δυνατών διαδρομών καθώς και το μήκος τους για επαλήθευση της ορθότητας.

483.xalancbmk

Το xalancbmk μετροπρόγραμμα είναι μια τροποποιημένη έκδοση του Xalan-C++, ενός XSLT επεξεργαστή που είναι γραμμένος σε ένα φορητό υποσύνολο της C++ και προορίζεται για τη μετατροπή αρχείων XML σε HTML, κείμενο ή άλλες μορφές αρχείων XML. Η είσοδος είναι ένα XML κείμενο και μια φόρμα XSL (Stylesheet), ενώ η έξοδος είναι ένα HTML κείμενο.

4.2.2 SPEC CFP2006

410.bwaves

Το 410.bwaves είναι ένα μετροπρόγραμμα γραμμένο σε Fortran 77, που προσομοιώνει αριθμητικά blast waves. Ο αλγόριθμος που έχει υλοποιηθεί κάνει επίλυση των συμπιέσιμων εξισώσεων Navier-Stokes χρησιμοποιώντας τον Bi-CGstab αλγόριθμο, που επιλύει

συστήματα μη συμμετρικών γραμμικών εξισώσεων επαναληπτικά. Το αρχείο εισόδου περιγράφει το μέγεθος του πλέγματος, τις παραμέτρους ροής, τις αρχικές οριακές συνθήκες και τον αριθμό των βημάτων.

416.gamess

Ένα μεγάλο εύρος κβαντικών χημικών υπολογισμών είναι δυνατό να πραγματοποιηθεί με τη χρήση του GAMESS. Το μετροπρόγραμμα 416.gamess είναι γραμμένο σε Fortran και πραγματοποιεί συνεπείς υπολογισμούς πεδίου (Self-consistent field (SCF) computation) του μορίου της κυτοσίνης (cytosine), του νερού και του cu2+, καθώς και του ιόντος του triazolium.

433.milc

Ο κώδικας του milc είναι ένα σύνολο από τμήματα κώδικα, ανεπτυγμένα από το MIMD Lattice Computation (MILC) Collaboration για την πραγματοποίηση προσομοιώσεων της SU(3) lattice gauge θεωρίας τεσσάρων διαστάσεων σε MIMD παράλληλα συστήματα. Στη σουίτα SPEC CPU2006 το 433.milc χρησιμοποιεί τη σειριακή έκδοση του su3imp προγράμματος, η οποία έχει αξία επειδή η απόδοση σε παράλληλα συστήματα βασίζεται στην καλή απόδοση σε αντίστοιχα μη-παράλληλα. Η lattice gauge θεωρία περιλαμβάνει τη μελέτη ορισμένων θεμελιωδών συστατικών της ύλης, τα quarks και τα γλουόνια.

434.zeusmp

Το μετροπρόγραμμα 434.zeusmp βασίζεται στο ZEUS-MP, που χρησιμοποιείται στον υπολογισμό δυναμικής ρευστών και έχει αναπτυχθεί στο εργαστήριο Υπολογιστικής Αστροφυσικής (Laboratory for Computational Astrophysics) του Πανεπιστημίου του Illinois για την προσομοίωση αστροφυσικών φαινομένων. Το φυσικό πρόβλημα που επιλύει το μετροπρόγραμμα είναι η προσομοίωση ενός τρισδιάστατου blast κύματος. Το αυθεντικό ZEUS-MP έχει βασιστεί στο ZEUS-3D και έχει παραλληλοποιηθεί χρησιμοποιώντας τη βιβλιοθήκη μηνυμάτων MPI.

444.namd

Το (C++) μετροπρόγραμμα 444.namd προέρχεται από το NAMD, ένα παράλληλο πρόγραμμα για την προσομοίωση μεγάλων βιομοριακών συστημάτων. Το σύνολο σχεδόν του χρόνου εκτέλεσης αφιερώνεται στον υπολογισμό διατομικών αλληλεπιδράσεων με ένα μικρό σύνολο συναρτήσεων. Αυτό το σύνολο χωρίστηκε από τον κύριο όγκο του κώδικα για να διαμορφώσει ένα συμπαγές μετροπρόγραμμα για τη σουίτα CPU2006.

450.soplex

Το μετροπρόγραμμα 450.soplex βασίζεται στην έκδοση 1.2.1 του SoPlex, που επιλύει ένα γραμμικό πρόγραμμα με χρήση του αλγορίθμου Simplex. Το SoPlex, όπως οι περισσότερες άλλες υλοποιήσεις του αλγορίθμου Simplex, υιοθετεί αλγορίθμους για αραιή γραμμική άλγεβρα, και ιδιαίτερα μια αραιή LU-παραγοντοποίηση και κατάλληλες ρουτίνες επίλυσης για τα προκύπτοντα τριγωνικά συστήματα εξισώσεων.

459.GemsFDTD

Το GemsFDTD λύνει τις εξισώσεις Maxwell σε τρεις διαστάσεις στο πεδίο του χρόνου, χρησιμοποιώντας τη μέθοδο διακριτών διαφορών στο πεδίο του χρόνου (finite-difference time-domain (FDTD) method). Το GemsFDTD είναι ένα υποσύνολο του κώδικα του GemsTD που έχει αναπτυχθεί στο πρόγραμμα γενικών ηλεκτρομαγνητικών επιλυτών (General Electro Magnetic Solvers (GEMS) project). Ο πυρήνας της μεθόδου FDTD είναι ακριβείς προσεγγίσεις, δεύτερης τάξης κεντρικής-διαφοράς των νόμων του Faraday και του Ampere. Η μέθοδος FDTD αναφέρεται επίσης ως σχέδιο Yee.

465.tonto

Το tonto είναι ένα πακέτο λογισμικού κβαντικής χημείας ανοικτού κώδικα. Ο υπολογισμός που πραγματοποιείται στο μετροπρόγραμμα της σουίτας SPEC CPU2006 αφορά το πεδίο της κβαντικής κρυσταλλογραφίας. Τοποθετεί έναν περιορισμό στο μοριακό υπολογισμό μιας κυματικής εξίσωσης Hartree-Fock για να συμφωνεί περισσότερο με πειραματικά δεδομένα διάθλασης ακτινών X. Το αρχείο εισόδου περιέχει την κρυσταλλική δομή, τις θέσεις των ατόμων, τις συναρτήσεις βάσης και πειραματικά δεδομένα διάθλασης ακτινών X. Όταν προσδιοριστεί το τελικό μοντέλο της κυματικής εξίσωσης, τα δεδομένα διάθλασης ακτινών X που υπολογίζονται από αυτή τυπώνονται μαζί με τα πειραματικά δεδομένα για σύγκριση.

4.3 Μελέτη των Μοτίβων Πρόσβασης στη Μνήμη

Η συμπεριφορά που παρουσιάζουν οι εφαρμογές και ειδικότερα τα μετροπρογράμματα της σουίτας SPEC CPU2006 εξαρτάται από τους πόρους της κρυφής μνήμης που δεσμεύουν. Σε προηγούμενες μελέτες ((12), (13) και (14)) τα μετροπρογράμματα που χρησιμοποιήθηκαν στην αξιολόγηση των προτεινόμενων σχεδίων, κατηγοριοποιήθηκαν κατά τον ίδιο ή

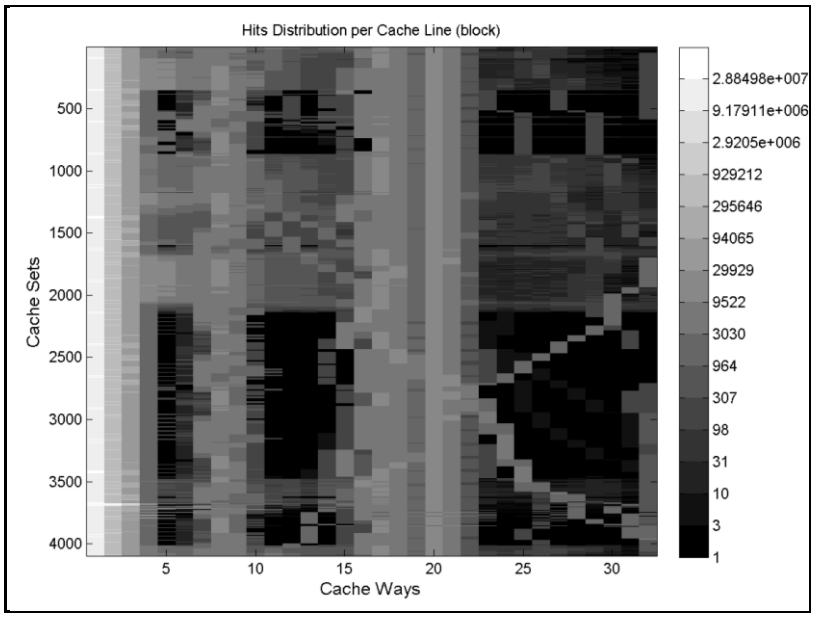
παρόμοιο τρόπο σε ομάδες σύμφωνα με το όφελος σε απόδοση που συνεπάγεται η δέσμευση περισσότερου χώρου της διαμοιραζόμενης κρυφής μνήμης. Ωστόσο, ενώ όλοι οι ερευνητές δίνουν μεγάλη σημασία στην απόδοση των εφαρμογών συναρτήσει των δρόμων της κρυφής μνήμης που καταλαμβάνουν, δεν εξετάζουν τόσο άλλα χαρακτηριστικά των μοτίβων πρόσβασης στην τελευταία.

Στην παρούσα παράγραφο, επιχειρείται μια διαφορετική προσέγγιση των μοτίβων πρόσβασης των εφαρμογών στις *περιοχές* της κρυφής μνήμης. Ο όρος *περιοχές* υποδηλώνει τις μικρότερες μονάδες αποθηκευτικού χώρου στις οποίες υποδιαιρείται νοητά η κρυφή μνήμη, δηλαδή τα μπλοκ της. Για το λόγο αυτό, πραγματοποιήθηκαν προσομοιώσεις για τη μελέτη των απαιτήσεων σε μνήμη των μετροπρογραμμάτων SPEC CPU2006. Οι προσομοιώσεις έγιναν με τη χρήση του Pin και ενός εργαλείου του (Memory) που μοντελοποιεί μια ιεραρχία μνήμης και το οποίο τροποποιήθηκε κατάλληλα για τις ανάγκες της εργασίας.

Πιο συγκεκριμένα το εργαλείο αυτό προσομοιώνει μια κρυφή μνήμη μεγέθους 4 Mbytes με 4096 γραμμές/σύνολα και 32 δρόμους και καταγράφει τις προσβάσεις του επεξεργαστή σε αυτή, καθώς και εάν το αποτέλεσμα ήταν ευστοχία ή αστοχία. Αν ήταν ευστοχία το εργαλείο καταγράφει και τη θέση τοπικότητας (recency position), δηλαδή σε ποια θέση της στοίβας των δρόμων που διατηρεί η LRU πολιτική αντικατάστασης σημειώθηκε. Τα στατιστικά που καταγράφηκαν, χρησιμοποιήθηκαν στα παρακάτω διαγράμματα τα οποία παρέχουν μια σαφή εικόνα της συμπεριφοράς των μετροπρογραμμάτων τόσο ανά δρόμο όσο και ανά σύνολο της κρυφής μνήμης.

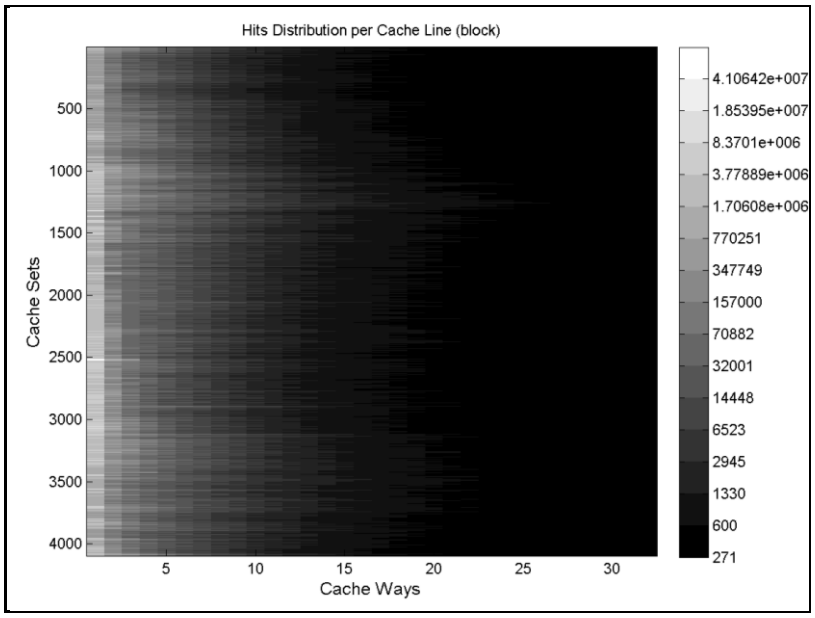
Στο σχήμα 4.1 παρατηρείται ότι το bwaves παρουσιάζει έντονες διακυμάνσεις των προσβάσεων ανά σύνολο και ανά δρόμο της κρυφής μνήμης. Πιο συγκεκριμένα, μέχρι τον τέταρτο δρόμο παρατηρείται μια ομοιόμορφα φθίνουσα πορεία των προσβάσεων, οι οποίες για τους δρόμους 5 και 6 φτάνουν στο ελάχιστο αλλά στη συνέχεια αυξάνονται σημαντικά στους δρόμους 7-10. Αυτό σημαίνει ότι ένα μεγάλο μέρος των δεδομένων που χρησιμοποιεί το bwaves έχει απόσταση επαναχρησιμοποίησης της τάξης του 7-10 (αριθμός accesses), δηλαδή επαναχρησιμοποιείται μία φορά για κάθε 7-10 προσπελάσεις. Το ίδιο φαινόμενο επαναλαμβάνεται ακόμα πιο έντονα στους δρόμους 11-14 (μείωση) και 15-22 (αύξηση). Ωστόσο, αντίστοιχες διακυμάνσεις δεν εμφανίζονται μόνο ανά δρόμο αλλά και ανά σύνολο της κρυφής μνήμης. Από το ίδιο σχήμα φαίνεται ότι για τα σύνολα από 1-400 και από 900-2100 γίνονται περισσότερες προσπελάσεις για τους δρόμους 5-6, 11-14 και 23-31, σε

αντίθεση με το υπόλοιπα όπου στους ίδιους δρόμους υπάρχουν μεγάλες μαύρες περιοχές. Επίσης στο σχήμα 4.1 παρατηρείται ένα μοτίβο επαναχρησιμοποίησης για τα σύνολα 2000 έως 3800, το οποίο μετακινείται από το δρόμο 30 στο 22 (σύνολα 2000-2700) και ανάποδα (σύνολα 2800-3800). Όλα αυτά δείχνουν ότι οι απαιτήσεις της εφαρμογής δεν κατανέμονται ομοιόμορφα στην κρυφή μνήμη.

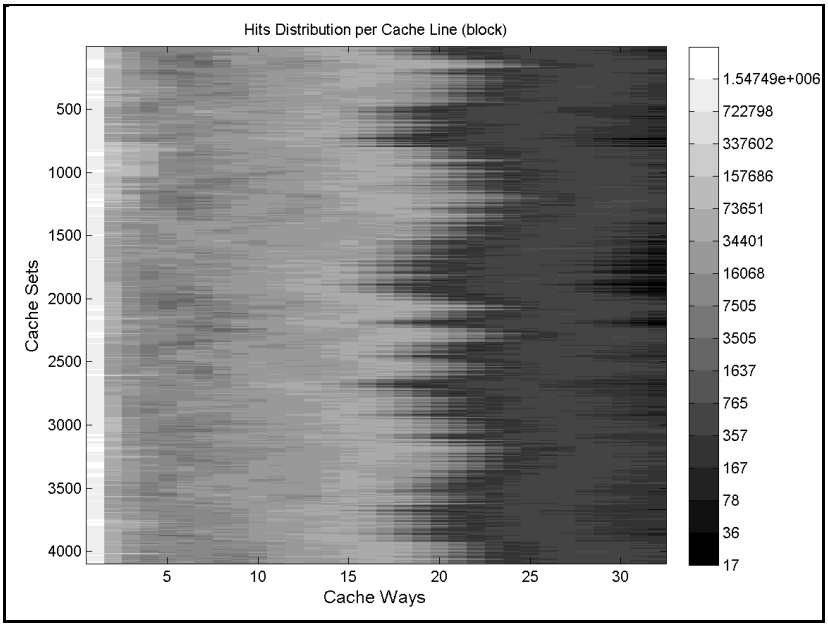


Σχήμα 4.1: Κατανομή ευστοχιών ανά μπλοκ για το bwaves

Το σχήμα 4.2 που αφορά το gobmk, δείχνει ότι το μετροπρόγραμμα αυτό παρουσιάζει μια φθίνουσα τάση στη συχνότητα προσπέλασης των γραμμών καθώς αυξάνεται ο αριθμός των δρόμων. Συνεπώς η LRU πολιτική αντικατάστασης αναμένεται να αποδίδει καλά σε αυτή την περίπτωση. Παρόλο που το διάγραμμα είναι σε γενικές γραμμές ομοιόμορφο, τα σύνολα από το 1000 έως και το 1500 περίπου, καθώς και από το 3100 έως και το 3700 φαίνεται ότι προσπελαύνονται συχνότερα.



Σχήμα 4.2: Κατανομή ευστοχιών ανά μπλοκ για το gobmk

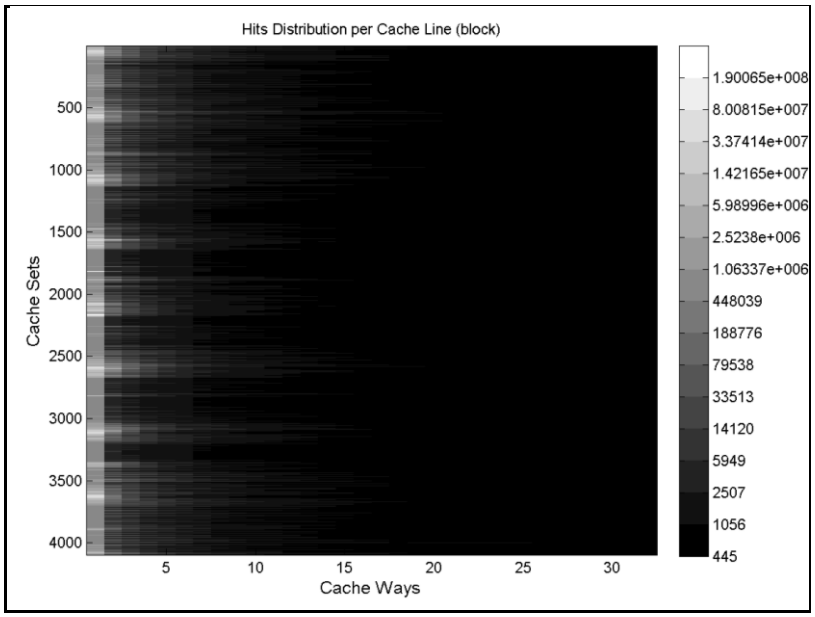


Σχήμα 4.3: Κατανομή ευστοχιών ανά μπλοκ για το h264ref

Το h264ref παρουσιάζει σε γενικές γραμμές σχετικά σταθερή μείωση των προσβάσεων στην κρυφή μνήμη καθώς προχωράμε από τον πρώτο δρόμο προς τον τελευταίο, ενώ οι περισσότερες προσπελάσεις γίνονται στους δρόμους 1 και 2. Όμως ο ρυθμός αυτής της μείωσης δεν είναι ίδιος για όλα τα σύνολα, αφού όπως φαίνεται στο σχήμα για κάποια σύνολα το διάγραμμα «σκουραίνει» από τον 15^{ο} δρόμο ενώ για κάποια άλλα από τον 25^{ο}

και μετά. Με άλλα λόγια για κάποια σύνολα το h264ref επαναχρησιμοποιεί πολλά από τα δεδομένα του κάθε 15 προσπελάσεις και για κάποια άλλα κάθε 25 προσπελάσεις.

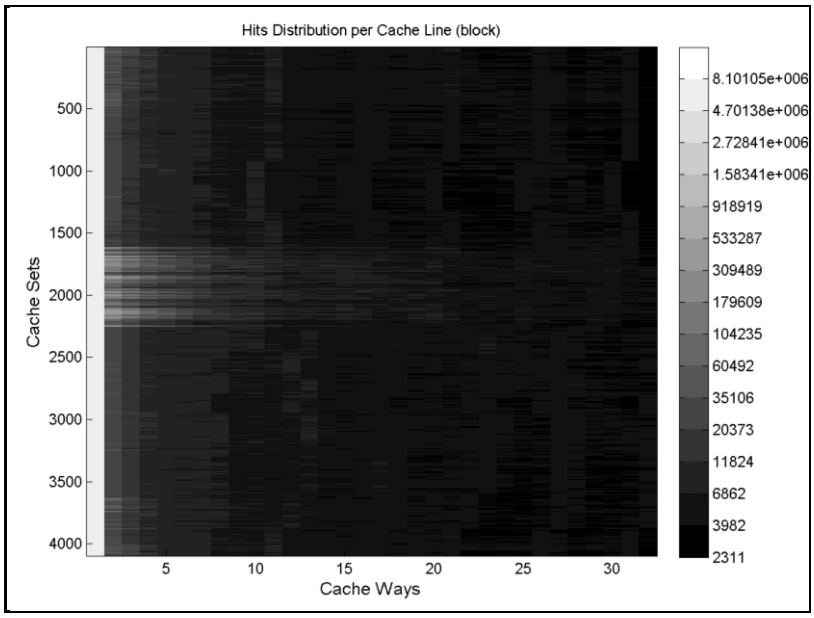
Στο σχήμα που αφορά το sjeng, γίνεται αμέσως αντιληπτό ότι ο ρυθμός πτώσης του πλήθους των προσπελάσεων συναρτήσει των δρόμων της κρυφής μνήμης είναι πολύ μεγάλος. Ωστόσο αυτό που παρατηρείται πολύ έντονα σε αυτή την περίπτωση, είναι η τεράστια διακύμανση των προσβάσεων ανάλογα με το σύνολο της κρυφής μνήμης. Ειδικότερα αν εξετάσουμε το διάγραμμα από πάνω προς τα κάτω, διακρίνουμε εύκολα επαναλαμβανόμενες περιοχές (διαδοχικών συνόλων) όπου το πλήθος των προσπελάσεων είναι σημαντικά αυξημένο για ένα εύρος τουλάχιστον 10 δρόμων. Μάλιστα στους 1-2 πρώτους δρόμους των συνόλων που ανήκουν στις περιοχές αυτές, τα πλήθη των προσπελάσεων είναι τάξεις μεγέθους μεγαλύτερα σε σχέση με των υπολοίπων συνόλων, όπως υποδεικνύει η κλίμακα στα δεξιά του διαγράμματος. Άρα τα περισσότερα δεδομένα του sjeng επαναχρησιμοποιούνται αμέσως ή ανά δύο προσπελάσεις στην κρυφή μνήμη.



Σχήμα 4.4: Κατανομή ευστοχιών ανά μπλοκ για το sjeng

Το soplex γενικά παρουσιάζει ομοιόμορφη συμπεριφορά και χρησιμοποιεί κυρίως τον πρώτο δρόμο της κρυφής μνήμης, όπου το πλήθος των προσπελάσεων είναι τάξεις μεγέθους μεγαλύτερο σε σχέση με τους δρόμους 2 και 3 (που είναι οι αμέσως λιγότερο χρησιμοποιημένοι). Συνεπώς η απόσταση επαναχρησιμοποίησης για τα περισσότερα μπλοκ είναι 1 (επαναχρησιμοποιούνται αμέσως). Ιδιαίτερο ενδιαφέρον παρουσιάζει επίσης η

περιοχή στη μέση περίπου της κρυφής μνήμης, η οποία αποτελείται από λίγο περισσότερα από 500 σύνολα και στην οποία πραγματοποιείται σαφώς αυξημένο πλήθος προσβάσεων.



Σχήμα 4.5: Κατανομή ευστοχιών ανά μπλοκ για το soplex

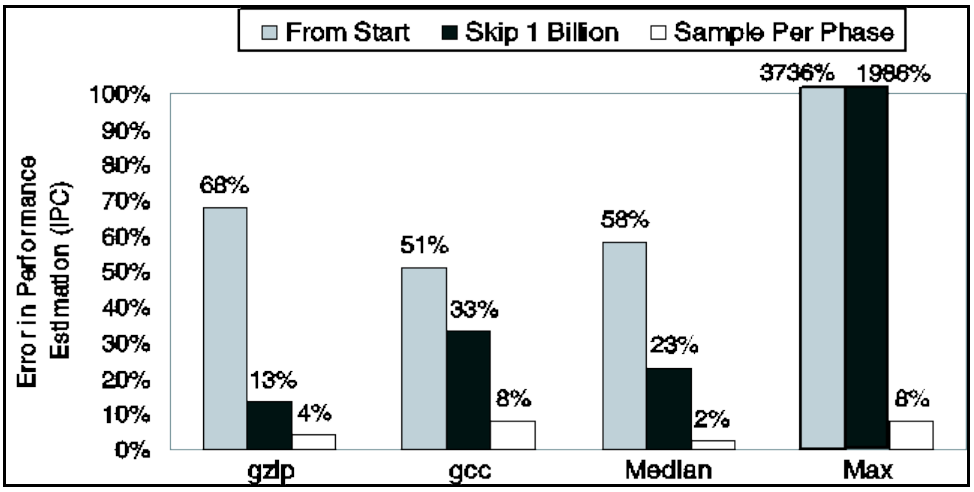
Από τα παραπάνω εξάγεται το συμπέρασμα ότι τα μετροπρογράμματα παρουσιάζουν διαφορετικές απαιτήσεις και συμπεριφορά ως προς τις γραμμές και τους δρόμους της κρυφής μνήμης. Αν οι εφαρμογές αυτές συνδυάζονταν σε μια διαμοιραζόμενη μνήμη, το ζητούμενο θα ήταν ιδανικά η καθεμία να χρησιμοποιούσε διαφορετικό αριθμό δρόμων ανά γραμμή. Η παρατήρηση αυτή είναι σημαντική γιατί υπογραμμίζει την ανάγκη για ανάπτυξη ενός σχεδίου καταμερισμού της κρυφής μνήμης που να διέπεται από μεγάλη ευελιξία.

4.4 Αξιολόγηση της Ορθότητας της Μεθοδολογίας και της Ακρίβειας των Αποτελεσμάτων

Η μεθοδολογία των PinPoints όπως την παρουσίασαν στο (19) οι Patil και συνεργάτες χρησιμοποιείται από πολλούς ερευνητές στο χώρο της αρχιτεκτονικής υπολογιστών. Ο κύριος λόγος για αυτό είναι ότι μειώνει δραστικά το χρόνο προσομοίωσης που απαιτείται για την αξιολόγηση των νέων σχεδίων υλικού εισάγοντας μικρό σφάλμα κατά την επιλογή των αντιπροσωπευτικών τμημάτων.

Το σχήμα 4.6 συγκρίνει την ακρίβεια των αποτελεσμάτων από την προσομοίωση του ίδιου πλήθους εντολών σε τρείς περιπτώσεις, σε σχέση με τα αποτελέσματα της πλήρους

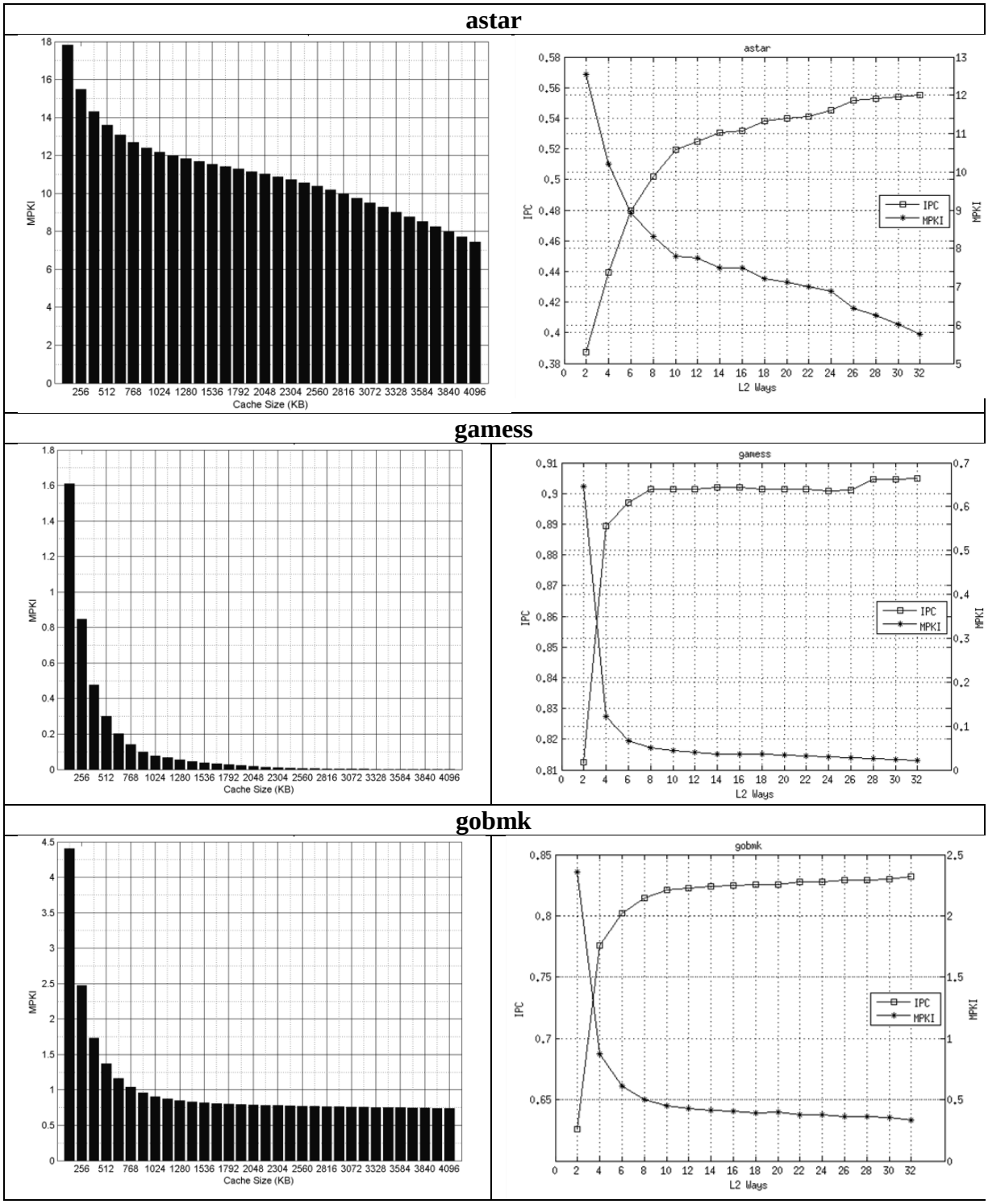
προσομοίωσης μιας εφαρμογής. Στην πρώτη περίπτωση η προσομοίωση ξεκινά από την αρχή της εκτέλεσης, στη δεύτερη ξεκινά μετά από fast-forwarding για ένα δισεκατομμύριο κύκλους, ενώ στην τρίτη περίπτωση προσομοιώνονται τα τμήματα εκτέλεσης της εφαρμογής που έχει επιλέξει το SimPoint. Η στήλη Median αφορά τα αποτελέσματα από όλη τη σουίτα SPEC CPU2000.



Σχήμα 4.6: Ακρίβεια αποτελεσμάτων μεθοδολογίας SimPoint (26)

Σε μια απόπειρα να επιβεβαιωθεί η ακρίβεια των αποτελεσμάτων που παρέχει η προσομοίωση των αντιπροσωπευτικών τμημάτων που προκύπτουν από τη μεθοδολογία των PinPoints, συγκρίνονται στη συνέχεια τα αποτελέσματα δύο διαφορετικών προσομοιώσεων. Ειδικότερα, στη μία περίπτωση οι μετρήσεις έχουν διεξαχθεί με χρήση του Pin και του τροποποιημένου Memory εργαλείου και αφορούν την πλήρη εκτέλεση των μετροπρογραμμάτων. Στη δεύτερη περίπτωση τα αποτελέσματα προέρχονται από τον jsimTracer προσομοιωτή και αφορούν την εκτέλεση μόνο των αντιπροσωπευτικών φάσεων που προσδιορίστηκαν με τη μεθοδολογία των PinPoints.

Στο σχήμα 4.7 παρατίθενται ενδεικτικά κάποια διαγράμματα των Misses Per Kilo Instructions (MPKI) συναρτήσει του μεγέθους της κρυφής μνήμης, όπου καθίσταται σαφής η ομοιότητα της συμπεριφοράς ακόμα και αν τα νούμερα των αστοχιών δε συμφωνούν απόλυτα. Σημειώνεται ότι και στις δύο περιπτώσεις το μέγεθος της κρυφής μνήμης κυμαίνεται από 128 Kbytes/1 δρόμο έως 4096 Kbytes/32 δρόμους, με βήματα των 256 Kbytes/2 δρόμων. Η αύξηση του συνολικού μεγέθους της μνήμης μέσω προσθήκης νέων δρόμων γίνεται για να διατηρηθεί σταθερή η αντιστοίχιση των μπλοκ της κύριας μνήμης σε μπλοκ της κρυφής μνήμης (memory mappings).



Σχήμα 4.7: Σύγκριση αποτελεσμάτων Pin-Memory tool και jsimTracer

Αναφορικά με τα διαγράμματα πρέπει να σημειωθεί ότι μπορεί ο ακριβής ρυθμός των MPKI να είναι διαφορετικός αλλά η συμπεριφορά που παρουσιάζουν οι εφαρμογές συναρτήσει του μεγέθους της κρυφής μνήμης είναι όμοια. Στο astar παρατηρείται μια

σχετικά σταθερή (με μικρές διακυμάνσεις) πτώση των MPKI η οποία είναι πιο απότομη για τους πρώτους 6 δρόμους. Το games αντιθέτως εμφανίζει κάθετη πτώση των MPKI στους πρώτους 4 δρόμους και στη συνέχεια φτάνει με σαφώς πιο αργό ρυθμό σε μηδενικά επίπεδα αστοχιών. Τέλος το gobmk παρουσιάζει και αυτό μεγάλη μείωση των MPKI για τους πρώτους δρόμους, ο ρυθμός της οποίας μειώνεται σταδιακά και στη συνέχεια γίνεται σχεδόν μηδενικός.

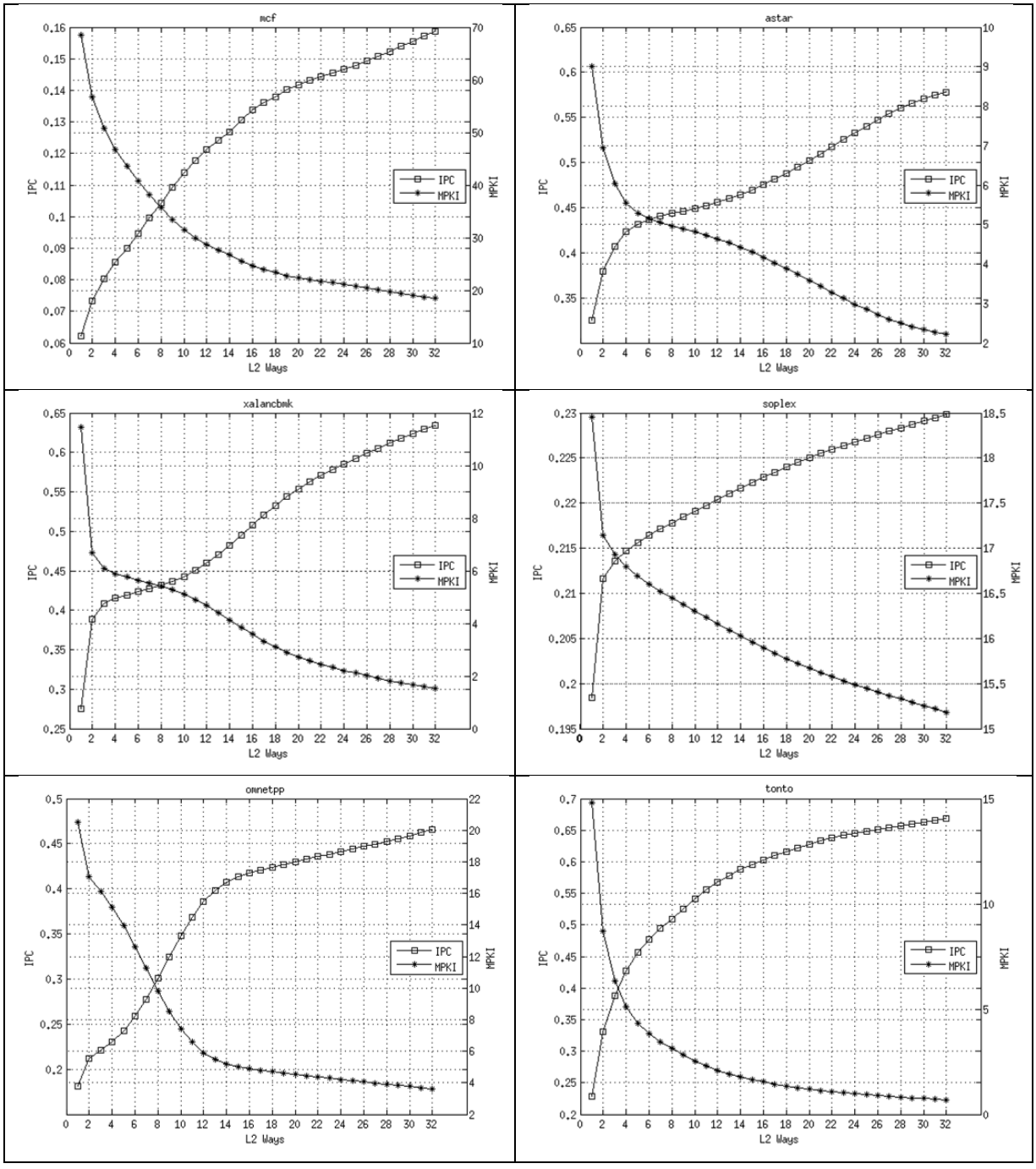
Η μεγάλη ομοιότητα που παρατηρείται στη συμπεριφορά των εφαρμογών στις δύο προσομοιώσεις, αποκτά βαρύνουσα σημασία αν ληφθεί υπόψη ότι στη μία περίπτωση (Pin – Memory tool) εκτελέστηκαν τρισεκατομμύρια εντολές ενώ στην άλλη (jsimTracer – μεθοδολογία PinPoints) εκτελέστηκαν μόνο ένα δισεκατομμύριο εντολές. Τα αποτελέσματα αυτά επιβεβαιώνουν ότι τα σημεία προσομοίωσης που προσδιορίστηκαν με τη μεθοδολογία του PinPoint μπορούν χρησιμοποιηθούν για να προσομοιωθούν και να μελετηθούν οι επιδράσεις της διαθέσιμης κρυφής μνήμης στις εφαρμογές που μελετούνται.

4.5 Κατηγοριοποίηση Εφαρμογών

Οι υπάρχουσες πολιτικές αντικατάστασης και ειδικότερα μία πολιτική αντικατάστασης που ανήκει στην LRU οικογένεια όταν υλοποιείται στα πλαίσια μιας διαμοιραζόμενης κρυφής μνήμης, αντιμετωπίζει τις αστοχίες όλων των εφαρμογών με ομοιόμορφο τρόπο και δεσμεύει το μεγαλύτερο μέρος της κρυφής μνήμης στην εφαρμογή που εμφανίζει την υψηλότερη ζήτηση (με βάση το ρυθμό αιτήσεων). Ωστόσο, δεδομένου ότι δεν ωφελούνται όλες οι εφαρμογές στον ίδιο βαθμό από την εκμετάλλευση των πρόσθετων πόρων της κρυφής μνήμης, η συνολική απόδοση ενδέχεται να είναι χαμηλή. Η παρατήρηση αυτή οδήγησε τους ερευνητές στην κατηγοριοποίηση των εφαρμογών ως προς τις απαιτήσεις τους σε μνήμη, όπως παρουσιάζεται παρακάτω.

Τα σχήματα 4.8, 4.9 και 4.10 που παρατίθενται στη συνέχεια, δείχνουν τη συμπεριφορά 18 μετροπρογραμμάτων της σουίτας SPEC CU2006 συναρτήσει του μεγέθους της κρυφής μνήμης δευτέρου επιπέδου που δεσμεύουν. Το μέγεθος και ο βαθμός συσχετιστικότητας της κρυφής μνήμης μεταβάλλονται από 256 Kbytes/1 δρόμο μέχρι τα 8192 Kbytes/32 δρόμους, με βήματα των 256 Kbytes/1 δρόμου. Αυτό έγινε προκειμένου να διατηρείται αμετάβλητη η αντιστοίχιση των μπλοκ της κύριας μνήμης στα μπλοκ της κρυφής μνήμης. Τα διαγράμματα επίσης απεικονίζουν την εξάρτηση του IPC (Instructions Per Cycle) από το μέγεθος της

κρυφής μνήμης που δεσμεύει η εφαρμογή, προκειμένου να καταστεί σαφής η άμεση σχέση των MPKI με το IPC.



Σχήμα 4.8: Cache friendly μετροπρογράμματα

Cache Friendly Εφαρμογές: Οι εφαρμογές αυτές ωφελούνται όσον αφορά την απόδοση από τη δέσμευση περισσότερων πόρων της κρυφής μνήμης και το πλήθος των αστοχιών που προκαλούν είναι γνησίως φθίνουσα συνάρτηση του μεγέθους της. Στο σχήμα 4.8 για

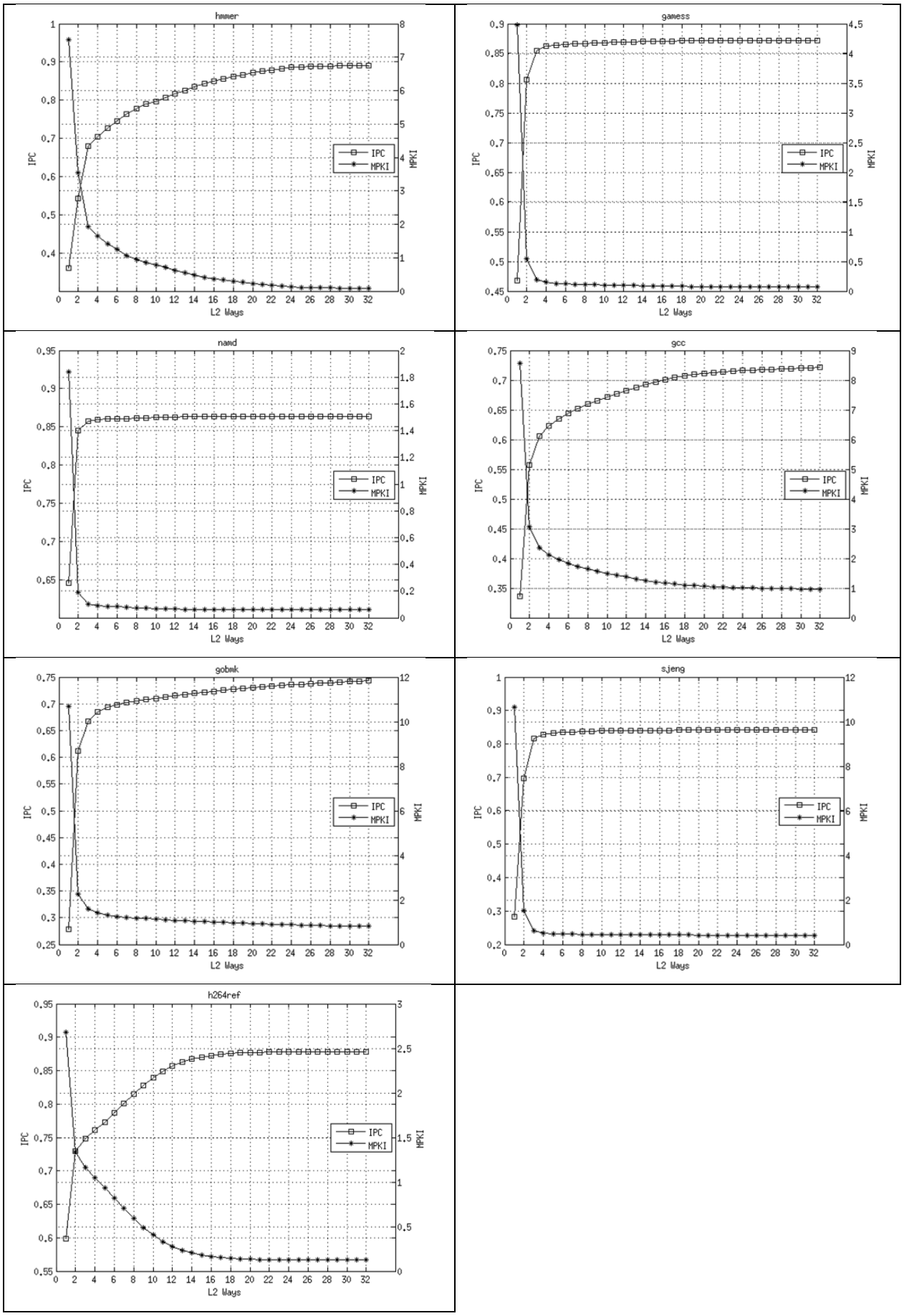
τυπικές cache friendly εφαρμογές όπως το astar, το mcf και το xalancbmk, η καμπύλη των αστοχιών παρουσιάζει συνεχή μείωση με ταυτόχρονη αύξηση του IPC καθώς αυξάνεται το πλήθος των δρόμων της συσχετιστικής μνήμης που δεσμεύει η εφαρμογή.

Παρατηρώντας περισσότερο τα διαγράμματα, φαίνεται ότι το mcf παρουσιάζει μεγάλο πλήθος αστοχιών ανά 1000 εντολές και ακόμη και για μεγέθη κρυφής μνήμης της τάξης των 8 Mbytes συνεχίζει να ωφελείται από την παραχώρηση επιπλέον πόρων. Επίσης το xalancbmk, εμφανίζει κάποια διακύμανση στο ρυθμό μείωσης των αστοχιών, ενώ το ίδιο ισχύει και για το omnetpp. Το μετροπρόγραμμα soplex όπως και το mcf είναι cache friendly εφαρμογή με υψηλά επίπεδα αστοχιών ακόμα και για μεγάλες κρυφές μνήμες (8 Mbytes).

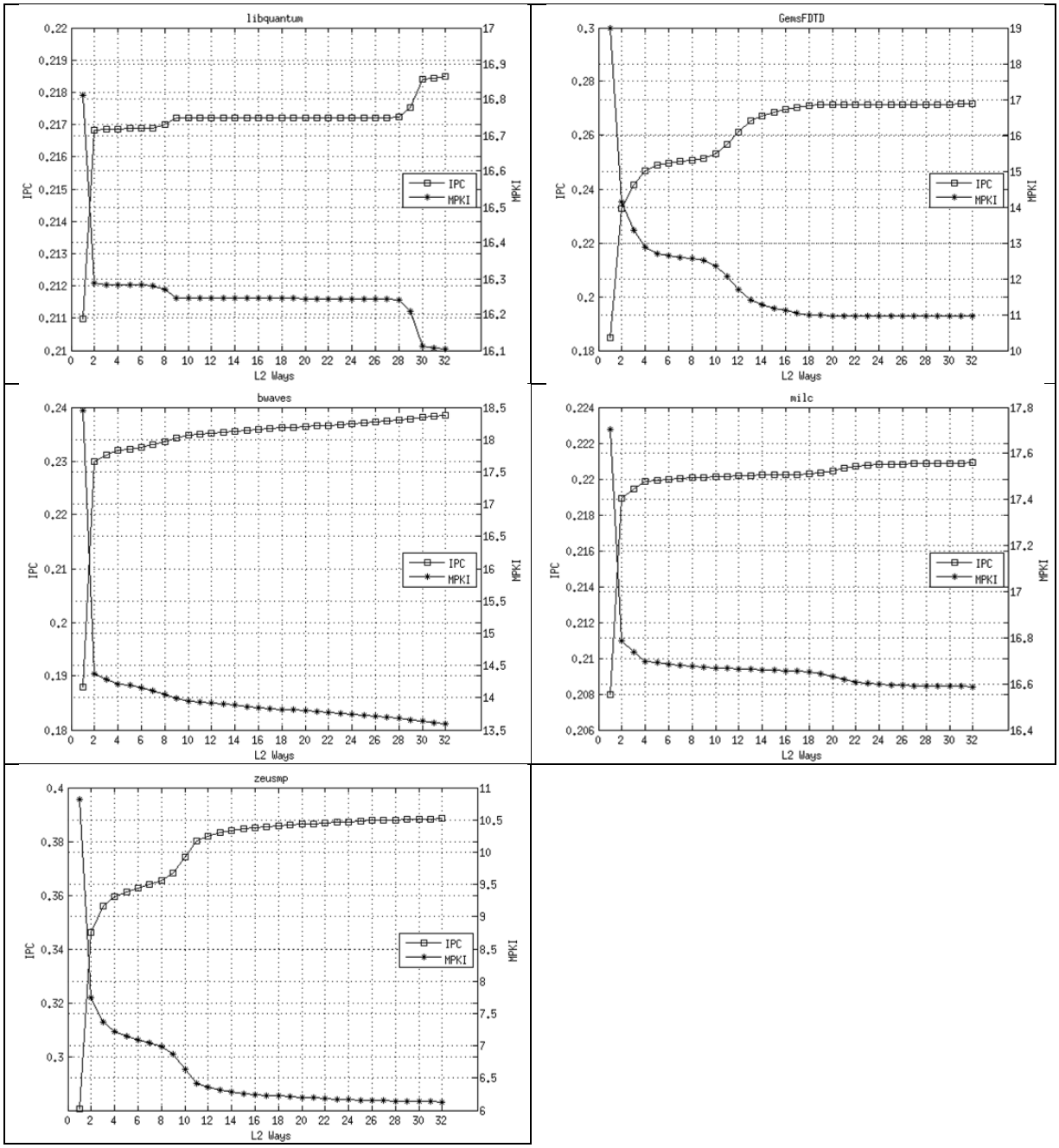
Cache Fitting Εφαρμογές: Στις εφαρμογές αυτές αρκεί ένα υποσύνολο της κρυφής μνήμης προκειμένου να εκτελεστούν αποδοτικά. Ειδικότερα, η καμπύλη των αστοχιών τους παρουσιάζει σε κάποιο δεδομένο μέγεθος κρυφής μνήμης έντονη πτώση έως ότου το πλήθος των αστοχιών φτάσει σε πολύ χαμηλά, σχεδόν μηδενικά επίπεδα. Για το λόγο αυτό θεωρείται ότι οι εφαρμογές αυτές χωρούν (fit) στην κρυφή μνήμη και ονομάζονται cache fitting. Χαρακτηριστικές περιπτώσεις cache fitting εφαρμογών όπως το hmmer, το games και το namd παρουσιάζονται στο σχήμα 4.9.

Από τη μελέτη του σχήματος 4.9 προκύπτει ότι το μετροπρόγραμμα namd με την αύξηση του χώρου της κρυφής μνήμης που του παραχωρείται, παρουσιάζει μια απότομη πτώση των MPKI μέχρι και τους 3 δρόμους και στη συνέχεια ο ρυθμός της πτώσης είναι σχεδόν μηδενικός. Συνεπώς πρόκειται για χαρακτηριστική περίπτωση cache fitting εφαρμογής. Επίσης, το hmmer προκαλεί πολύ λίγες αστοχίες, ειδικά μετά τα 2 Mbytes κρυφής μνήμης επομένως κατατάσσεται κι αυτό στις cache fitting εφαρμογές.

Το μετροπρόγραμμα gamess εμφανίζει ούτως ή άλλως μικρό πλήθος αστοχιών. Μάλιστα όπως φαίνεται από το δεύτερο διάγραμμα του σχήματος 4.9 μια κρυφή μνήμη 1024 Kbytes με 4 δρόμους επαρκεί για να συμβαίνουν λιγότερες από 0,2 αστοχίες ανά χίλιες εκτελούμενες εντολές. Συνεπώς είναι τυπική περίπτωση cache fitting εφαρμογής, όπως και το namd.



Σχήμα 4.9: Cache fitting μετροπρογράμματα



Σχήμα 4.10: Cache thrashing μετροπρογράμματα

Cache Thrashing Εφαρμογές: Οι εφαρμογές αυτές έχουν μεγέθη συνόλων εργασίας μεγαλύτερα από το μέγεθος ολόκληρης της διαθέσιμης κρυφής μνήμης και παρουσιάζουν φτωχή επαναχρησιμοποίηση των πόρων της. Σε αυτή την κατηγορία εντάσσονται και οι εφαρμογές ροής. Οι εφαρμογές αυτές προκαλούν το φαινόμενο thrashing στην περίπτωση της LRU πολιτικής αντικατάστασης αλλά μπορεί να παρουσιάσουν καλύτερη απόδοση με τη χρήση κάποιας άλλης πολιτικής. Όταν μια εφαρμογή αυτής της κατηγορίας, εκτελείται

ταυτόχρονα με κάποια cache friendly ή cache fitting εφαρμογή, είναι προτιμότερο να μειωθούν οι πόροι που της διατίθενται προς όφελος της συνολικής απόδοσης.

Το libquantum είναι τυπική περίπτωση cache thrashing εφαρμογής, καθώς για μέγεθος κρυφής μνήμης μεγαλύτερο από 2560 Kbytes παρουσιάζει υψηλό ρυθμό αστοχιών αλλά δεν ωφελείται σχεδόν καθόλου από τη δέσμευση περισσότερων πόρων της κρυφής μνήμης μέχρι και τα 7680 Kbytes. Η LRU πολιτική αντικατάστασης θα δεσμεύσει μεγάλο μέρος της κρυφής μνήμης για την ικανοποίηση του υψηλού ρυθμού αιτημάτων αυτής της εφαρμογής, υποβαθμίζοντας τη συνολική απόδοση. Αντίστοιχη συμπεριφορά με το libquantum παρουσιάζει και το milc.

Το μετροπρόγραμμα GemsFDTD ωφελείται από την παραχώρηση περισσότερων πόρων της κρυφής μνήμης, για μεγέθη cache έως και λίγο μεγαλύτερα από 4 Mbytes. Ωστόσο, το διάγραμμα των MPKI συναρτήσει του μεγέθους της κρυφής μνήμης παρουσιάζει περιοχές όπου ο ρυθμός μείωσης των αστοχιών είναι πολύ μικρός (στην περιοχή από 5 έως 9 δρόμους) ή μηδενικός (στην περιοχή από 18 έως 32 δρόμους). Συνεπώς, το GemsFDTD χαρακτηρίζεται ως cache thrashing εφαρμογή. Στην ίδια κατηγορία εμπίπτει και το zeusmp του οποίου το διάγραμμα διαφέρει από του GemsFDTD κυρίως στα νούμερα των MPKI. Επιπλέον και το bwaves εντάσσεται στις cache thrashing εφαρμογές, επειδή εξακολουθεί να προκαλεί πολλές αστοχίες ακόμα και όταν του παραχωρείται πολλή μνήμη.

Τέλος είναι κατανοητό ότι τα όρια μεταξύ των κατηγοριών που ορίσθηκαν συχνά είναι δυσδιάκριτα και μια εφαρμογή ενδέχεται να φέρει χαρακτηριστικά από δύο εξ αυτών. Για παράδειγμα το tonto χαρακτηρίστηκε ως cache friendly εφαρμογή αλλά εάν η διαθέσιμη κρυφή μνήμη έφθανε μέχρι και τους 48 δρόμους, πιθανότατα θα κατατασσόταν στις cache fitting εφαρμογές. Παρόλα ταύτα επιχειρήθηκε ο κατά το δυνατόν πιο σαφής χαρακτηρισμός των εφαρμογών.

4.6 Εφαρμογή της FAME Μεθοδολογίας

Η μεθοδολογία του FAME που παρουσιάστηκε στο τρίτο κεφάλαιο εφαρμόστηκε στα ίχνη πρόσβασης στη μνήμη των φάσεων εκτέλεσης που προσδιορίστηκαν με τη μεθοδολογία των PinPoints και καταγράφηκαν με το module που αναπτύχθηκε για τον Simics. Τα αποτελέσματα του τύπου του FAME για τα ίχνη αυτά παρουσιάζονται στον πίνακα που ακολουθεί (4.2).

Benchmark Name	**MAIV(%)**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
astar	24	12	9	7	**6**	5	4	4	4	3
bwaves	8	5	4	3	**3**	2	2	2	2	2
gcc	7	4	3	2	**2**	2	2	2	2	2
GemsFDTD	32	16	11	8	**7**	6	5	4	4	4
gobmk	7	4	3	2	**2**	2	2	1	1	1
h264ref	7	4	3	2	**2**	2	2	1	1	1
hmmer	5	3	2	2	**2**	2	2	2	1	1
mcf	15	8	6	5	**4**	3	3	3	3	2
milc	7	4	3	3	**2**	2	2	2	2	2
namd	28	14	9	7	**6**	5	4	4	3	3
omnetpp	17	9	6	5	**4**	4	3	3	3	2
sjeng	84	42	28	21	**17**	14	12	11	10	9
soplex	10	6	4	3	**3**	3	3	2	2	2
tonto	10	6	4	3	**3**	2	2	2	2	2
xalancbmk	24	12	9	7	**6**	5	4	4	4	3
zeusmp	10	6	4	3	**3**	2	2	2	2	2

Πίνακας 4.2: Πλήθος απαιτούμενων εκτελέσεων συναρτήσει του παράγοντα MAIV για κάθε μετροπρόγραμμα

Στον παραπάνω πίνακα έχει επισημανθεί η στήλη που αντιστοιχεί σε τιμή του παράγοντα MAIV ίση με 5%, επειδή για τις προσομοιώσεις στον jsimTracer, χρησιμοποιήθηκε το πλήθος επαναλήψεων που αντιστοιχεί σε αυτό το περιθώριο σφάλματος.

Κεφάλαιο 5

Δυναμικός Καταμερισμός της Διαμοιραζόμενης Μνήμης

5.1 Γενικά

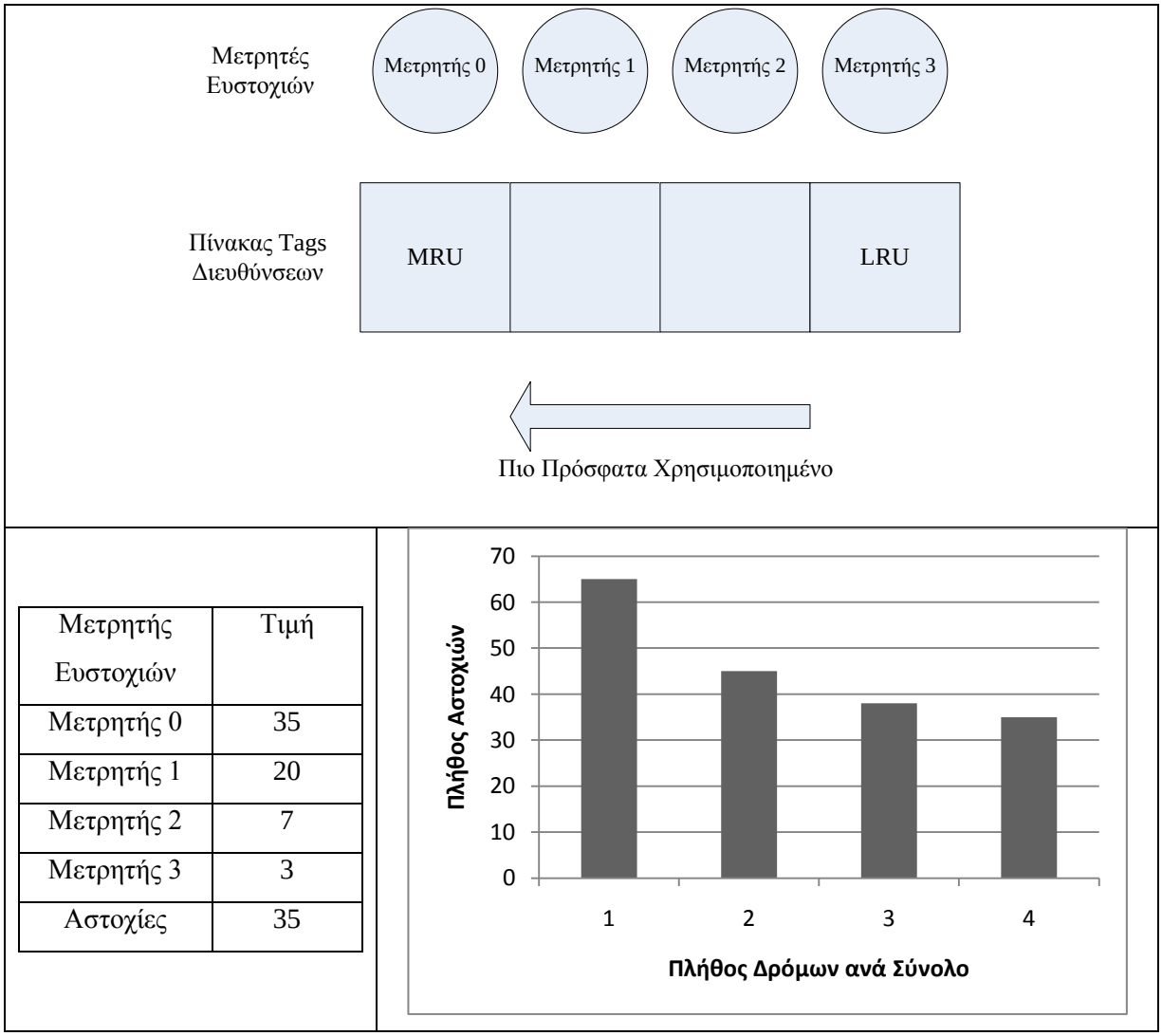
Στο κεφάλαιο αυτό εξετάζονται αναλυτικά τα δύο σχέδια καταμερισμού που αποτέλεσαν το κύριο αντικείμενο μελέτης αυτής της εργασίας και τη βάση για την πρόταση τροποποιήσεων και βελτιστοποιήσεων, ούτως ώστε να διαμορφωθεί ένα νέο αποδοτικό σχέδιο. Τα σχέδια αυτά είναι τα UCP (13) και ABFCP (14) που αναφέρθηκαν για πρώτη φορά στο κεφάλαιο 2.5.3.

5.2 Η Ιδιότητα της Στοίβας

Η πολιτική αντικατάστασης του *ελάχιστα πρόσφατα χρησιμοποιημένου* (LRU) μπλοκ υπακούει στην *ιδιότητα της στοίβας* (stack property), όπως έδειξαν οι Mattson και συνεργάτες στο (27). Σύμφωνα με την ιδιότητα αυτή, μία προσπέλαση η οποία ευστοχεί σε μία κρυφή μνήμη που διαχειρίζεται σύμφωνα με την LRU πολιτική και περιέχει N δρόμους, είναι σίγουρο ότι θα ευστοχούσε σε μια LRU κρυφή μνήμη με περισσότερους από N δρόμους, δεδομένου ότι το πλήθος των συνόλων διατηρείται σταθερό.

Η ιδιότητα της στοίβας είναι πολύ σημαντική, διότι με τη χρήση ενός πίνακα N θέσεων επιτρέπεται ο υπολογισμός της πληροφορίας για τις ευστοχίες και τις αστοχίες για όλες τις περιπτώσεις μιας κρυφής μνήμης με σταθερό πλήθος συνόλων, όπου κάθε σύνολο περιλαμβάνει από 1 έως N δρόμους. Συνεπώς καθίσταται δυνατός ο δραστικός περιορισμός του κόστους σε υλικό που απαιτείται για τον προσδιορισμό της σημασίας που έχει κάθε δρόμος στην αποφυγή των άστοχων προσπελάσεων. Το σχήμα 5.1 δείχνει πώς χρησιμοποιείται η ιδιότητα της στοίβας. Στο παράδειγμα αυτό χρησιμοποιείται μια συσχετιστική κρυφή μνήμη τεσσάρων δρόμων. Το πρόγραμμα που εκτελείται προκαλεί έναν αριθμό ευστοχιών σε κάθε δρόμο της κρυφής μνήμης που καταγράφεται από τους μετρητές, ενώ ένα μέρος των προσπελάσεων (συγκεκριμένα 35) καταλήγει σε αστοχία.

Σύμφωνα με τη ιδιότητα της στοίβας αν η κρυφή μνήμη είχε ένα δρόμο λιγότερο οι αστοχίες θα ήταν 38, αν είχε δύο δρόμους λιγότερους θα ήταν 45 κ.ο.κ.



Μετρητής Ευστοχιών	Τιμή
Μετρητής 0	35
Μετρητής 1	20
Μετρητής 2	7
Μετρητής 3	3
Αστοχίες	35

Σχήμα 5.1: Παράδειγμα χρήσης της ιδιότητας της στοίβας για ην LRU πολιτική αντικατάστασης

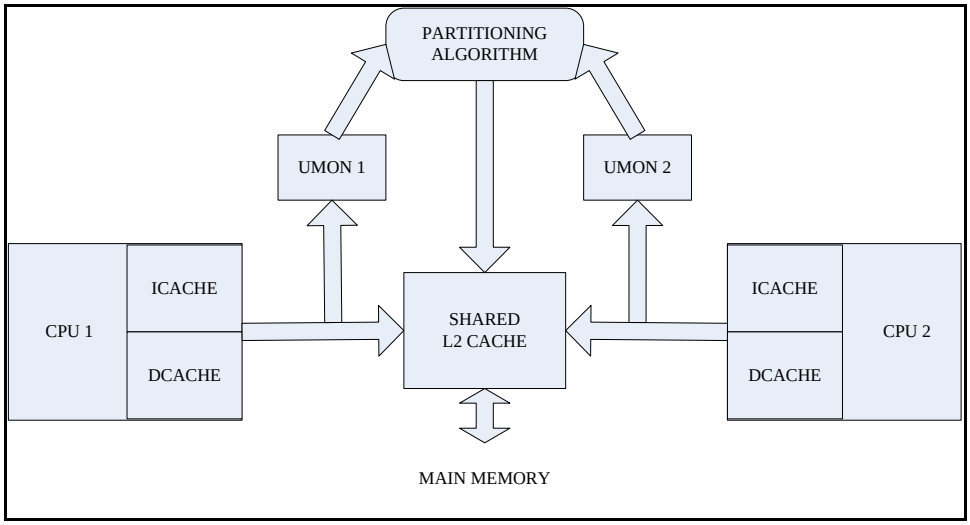
5.3 Το UCP Σχέδιο Δυναμικού Καταμερισμού της Κρυφής Μνήμης

Οι Qureshi και Patt διαπίστωσαν στο (13) πως όταν η LRU πολιτική χρησιμοποιείται σε επεξεργαστές που εκτελούν φόρτους εργασίας πολλαπλών εφαρμογών, διαμοιράζει την κρυφή μνήμη σύμφωνα με το βαθμό ζήτησης που παρουσιάζει κάθε εφαρμογή. Ωστόσο αυτό δεν ευνοεί απαραίτητα τη συνολική απόδοση. Παρατηρώντας αυτό το μειονέκτημα της LRU πολιτικής αντικατάστασης σε συσχετιστικές κρυφές μνήμες, πρότειναν μία

πολιτική καταμερισμού της κρυφής μνήμης σύμφωνα με τη χρησιμότητα (utility) του κάθε δρόμου.

Συγκεκριμένα πρότειναν ένα σχέδιο διαμοιρασμού που καταμερίζει τους πόρους της κρυφής μνήμης ανάμεσα στις εκτελούμενες εφαρμογές σύμφωνα με ένα πιο αξιόπιστο κριτήριο σε σχέση με το βαθμό ζήτησης. Το κριτήριο αυτό είναι η μείωση των αστοχιών που αναμένεται να εμφανίσει η κάθε εφαρμογή όταν δεσμεύεται για αυτή ένα δεδομένο ποσοστό πόρων της κρυφής μνήμης. Το σχέδιο αυτό ονομάστηκε *καταμερισμός της κρυφής μνήμης βάσει χρησιμότητας* (Utility-based Cache Partitioning - UCP).

Η υλοποίηση του μηχανισμού περιλαμβάνει καταρχήν την ανάθεση ενός κυκλώματος παρακολούθησης της χρησιμότητας που ονομάζεται UMON (Utility MONitor) σε κάθε πυρήνα του επεξεργαστή, προκειμένου να προσδιορισθούν τα χαρακτηριστικά των εφαρμογών που εκτελούνται σε αυτούς. Με άλλα λόγια καθένα από αυτά τα κυκλώματα καταγράφει το όφελος που συνεπάγεται για κάθε πυρήνα και συνακόλουθα για κάθε εφαρμογή, η δέσμευση περισσότερων δρόμων της κρυφής μνήμης. Η καταγραφή αυτή στηρίζεται στην ιδιότητα της στοίβας της LRU πολιτικής αντικατάστασης.



Σχήμα 5.2: Το UCP σχέδιο και τα κυκλώματα UMON

Κάθε κύκλωμα UMON περιλαμβάνει ένα δικό του πίνακα με tags που έχει τον ίδιο βαθμό συσχετιστικότητας με τον πίνακα των tags της διαμοιραζόμενης κρυφής μνήμης και χρησιμοποιεί την LRU πολιτική αντικατάστασης. Εντοπίζοντας τις προσβάσεις του κάθε πυρήνα στη διαμοιραζόμενη κρυφή μνήμη, το κύκλωμα UMON ουσιαστικά προσομοιώνει μία κρυφή μνήμη που είναι ιδιωτική για κάθε έναν από αυτούς. Προφανώς η ταυτοποίηση

του πυρήνα που προσπελαύνει την κρυφή μνήμη απαιτεί την εισαγωγή ενός πεδίου ταυτότητας πυρήνα (processor ID) σε κάθε tag της κρυφής μνήμης με μέγεθος $\log_2 P$ bits, όπου P είναι το πλήθος των πυρήνων του επεξεργαστή.

Επιπλέον κάθε UMON κύκλωμα ενσωματώνει ένα σύνολο από N μετρητές, όπου N ο βαθμός συσχετιστικότητας της διαμοιραζόμενης κρυφής μνήμης. Οι μετρητές καταγράφουν το πλήθος των ευστοχιών για κάθε μία από τις θέσεις τοπικότητας που κυμαίνονται από το περισσότερο πρόσφατα μέχρι το ελάχιστα πρόσφατα χρησιμοποιημένο. Όπως εξηγήθηκε και στην παράγραφο 5.2, λόγω της ιδιότητας της στοίβας στην οποία υπακούει η LRU πολιτική επαρκεί ένας πίνακας από tags με N δρόμους, για τον υπολογισμό της πληροφορίας για όλες τις περιπτώσεις που η κρυφή μνήμη περιλαμβάνει από 1 έως N δρόμους.

Η πληροφορία που συλλέγεται από τα κυκλώματα UMON χρησιμοποιείται για τον καταμερισμό που επιτελεί το UCP και ο οποίος γίνεται με βάση τους δρόμους (way-based), δηλαδή επιτρέπει σε κάθε εφαρμογή να καταλαμβάνει ένα συγκεκριμένο ποσοστό των διαθέσιμων δρόμων. Συνεπώς το ποσοστό της κρυφής μνήμης που κατέχει ένα νήμα είναι ίσο με το πλήθος των συνόλων της κρυφής μνήμης, επί τον αριθμό των δρόμων που καταλαμβάνει σε καθένα από αυτά.

Το UCP υλοποιείται τροποποιώντας την LRU πολιτική. Συγκεκριμένα, στην περίπτωση μιας αστοχίας στην κρυφή μνήμη, οι καταχωρήσεις που ανήκουν στην εφαρμογή που προκάλεσε την αστοχία μετρώνται. Αν αυτός ο αριθμός είναι μικρότερος από το όριο που επιβάλλει ο καταμερισμός, τότε εκδιώκεται από την κρυφή μνήμη η λιγότερο πρόσφατα χρησιμοποιημένη καταχώρηση που δεν ανήκει στην εφαρμογή. Στην αντίθετη περίπτωση, επιλέγεται προς αντικατάσταση η λιγότερο πρόσφατα χρησιμοποιημένη καταχώρηση που ανήκει στην εφαρμογή που προκάλεσε την αστοχία.

Ο αλγόριθμος καταμερισμού εκτελείται κάθε πέντε εκατομμύρια κύκλους. Επειδή ο καταμερισμός επιτελείται σε επίπεδο δρόμου (way-based), ο αλγόριθμος αξιολογεί όλες τις πιθανές κατανομές των δρόμων της κρυφής μνήμης ανάμεσα στις ανταγωνιζόμενες εφαρμογές, διαβάζοντας τους μετρητές των ευστοχιών από όλα τα κυκλώματα UMON και υπολογίζοντας το συνολικό αριθμό των ευστοχιών για κάθε περίπτωση. Ο αλγόριθμος αυτός παρέχει μεγάλη ευελιξία διότι έχει τη δυνατότητα να μεταβάλλει τη δέσμευση δρόμων για κάθε εφαρμογή κατά ±N, όπου N ο βαθμός συσχετιστικότητας της κρυφής

μνήμης και επομένως εγγυάται την επιλογή του καταμερισμού που μεγιστοποιεί το συνολικό αριθμό ευστοχιών στο σύστημα. Επίσης εξασφαλίζει ότι κάθε εφαρμογή θα δεσμεύει τουλάχιστον ένα δρόμο.

Στην ιδανική περίπτωση, το UCP σχέδιο περιλαμβάνει έναν πίνακα από tags διευθύνσεων για κάθε μπλοκ της κρυφής μνήμης. Σε αυτή την περίπτωση ο (ιδανικός) αλγόριθμος του UCP σχεδίου, ελέγχει όλες τους δυνατούς τρόπους διαμοιρασμού της κρυφής μνήμης για κάθε σύνολο και υπολογίζει το όφελος ή την απώλεια στην απόδοση για κάθε νήμα ξεχωριστά. Ωστόσο το πλήθος των πιθανών καταστάσεων αυξάνεται εκθετικά όπως φαίνεται και στον πίνακα 5.1. Για το λόγο αυτό οι ερευνητές πρότειναν και δύο επιπλέον αλγορίθμους καταμερισμού με σκοπό τον περιορισμό της πολυπλοκότητας, εκ των οποίων ο ένας ακολουθεί την greedy τεχνική. Παρόλο που οι αλγόριθμοι αυτοί προτάθηκαν προς αντικατάσταση του ιδανικού, τα αποτελέσματα των μετρήσεων που παρουσιάζονται στη συνέχεια προέκυψαν με χρήση του ιδανικού αλγορίθμου.

Πλήθος Πυρήνων	**Πιθανές Περιπτώσεις Καταμερισμού**
2	31
4	6,545
8	15,380,937

Πίνακας 5.1: Πλήθος πιθανών καταμερισμών συναρτήσει του αριθμού των πυρήνων

Ένα ακόμα βασικό πλεονέκτημα που προβάλλει το σχέδιο που προτάθηκε από τους Qureshi και συνεργάτες, έγκειται στο γεγονός ότι το κύκλωμα UMON κάθε πυρήνα προσομοιώνει την κατάσταση της διαμοιραζόμενης κρυφής μνήμης σαν να χρησιμοποιείται μόνο από το συγκεκριμένο πυρήνα. Επομένως η εξαγόμενη πληροφορία η οποία τελικά χρησιμοποιείται για να καθορίσει τον καταμερισμό της κρυφής μνήμης, δεν αλλοιώνεται από τις ανταγωνιστικές προσβάσεις των παράλληλα εκτελούμενων εφαρμογών και συνεπώς μπορεί να χρησιμοποιηθεί με μεγαλύτερη βεβαιότητα.

Ωστόσο, το κόστος σε υλικό του μηχανισμού που πρότειναν οι Qureshi και συνεργάτες είναι πολύ μεγάλο στην ιδανική περίπτωση. Όπως αναλύθηκε και παραπάνω, απαιτείται η ύπαρξη ενός ιδιωτικού πίνακα από tags με τον ίδιο βαθμό συσχετιστικότητας με αυτόν της κρυφής μνήμης, για κάθε μπλοκ της κρυφής μνήμης και για κάθε πυρήνα. Προκειμένου να

υπερκεράσουν αυτό το πρόβλημα εισήγαγαν την ιδέα της *δυναμικής δειγματοληψίας συνόλων* (Dynamic Set Sampling).

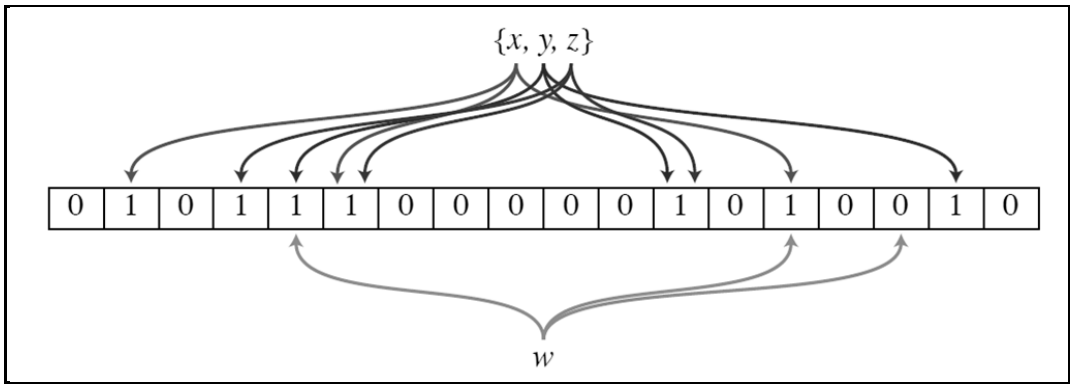
Το DSS στηρίζεται στην παραδοχή ότι η παρακολούθηση ενός μικρού πλήθους συνόλων της κρυφής μνήμης, παρέχει ικανοποιητική πληροφορία για τον καθορισμό του καταμερισμού των δρόμων στις εφαρμογές, για όλα τα σύνολα της κρυφής μνήμης. Πιο συγκεκριμένα οι Qureshi και συνεργάτες κατέληξαν στο συμπέρασμα ότι μπορούν να επιτευχθούν παρόμοια αποτελέσματα στη συνολική απόδοση του συστήματος, όταν κάθε UMON κύκλωμα παρακολουθεί μόνο 32 από τις 1024 γραμμές της κρυφής μνήμης που χρησιμοποιούσαν για τις μετρήσεις τους. Η παραδοχή αυτή συνεπάγεται την εφαρμογή του ίδιου καταμερισμού για όλα τα σύνολα της κρυφής μνήμης. Το γεγονός επιφέρει σημαντική απώλεια ευελιξίας για το UCP σχέδιο.

5.4 Το ABFCP Σχέδιο Δυναμικού Καταμερισμού της Κρυφής Μνήμης

Οι Nikas και συνεργάτες (14) ανέπτυξαν μια άλλη προσέγγιση που δύναται να παρέχει παρόμοια επίπεδα ακρίβειας με το UCP σχέδιο αλλά χωρίς να απαιτεί την παρακολούθηση όλων των tags της κρυφής μνήμης, μειώνοντας κατ' αυτόν τον τρόπο το κόστος της υλοποίησης σε υλικό. Η λύση αυτή στηρίζει σε μεγάλο βαθμό τη λειτουργία της στα Bloom Filters που επινοήθηκαν από τον Burton H. Bloom το 1970.

5.4.1 Τι είναι το Bloom φίλτρο

Το φίλτρο Bloom, είναι μια χωρικά αποδοτική, πιθανοτική δομή δεδομένων που χρησιμοποιείται για να εξετάσει εάν ένα στοιχείο είναι μέλος ενός συνόλου. Έχει πολύ συμπαγή μορφή καθώς η ύπαρξη ενός στοιχείου στο σύνολο υποδηλώνεται από μόνο k bits, όπου k το πλήθος των συναρτήσεων κατακερματισμού (hash functions). Το πλεονέκτημα αυτό συνοδεύεται από το μειονέκτημα της πιθανής ύπαρξης ψευδώς θετικών μελών (ψευδώς αρνητικά δεν υφίστανται). Όσο περισσότερα στοιχεία προστίθενται στο σύνολο, τόσο μεγαλύτερη είναι η πιθανότητα των ψευδών θετικών καταχωρήσεων.



Σχήμα 5.3: Παράδειγμα ενός Bloom φίλτρου με τρείς hash συναρτήσεις

Ένα κενό Bloom φίλτρο είναι ένας πίνακας από m bits, που έχουν όλα τεθεί στο 0. Πρέπει επίσης να έχουν οριστεί k διαφορετικές hash συναρτήσεις, κάθε μια από τις οποίες αντιστοιχεί κάποιο στοιχείο που έχει τεθεί σε μια από τις m θέσεις του πίνακα με μια ομοιόμορφη τυχαία κατανομή. Για να προστεθεί ένα στοιχείο, περνάει από κάθε μια από τις k hash συναρτήσεις για να προσδιοριστούν k θέσεις του πίνακα των οποίων τα bits θα τεθούν στην τιμή 1.

Για να αποφασιστεί αν ένα στοιχείο είναι μέλος του Bloom φίλτρου, το στοιχείο αυτό περνάει από κάθε μια από τις k hash συναρτήσεις για να προσδιοριστούν k θέσεις του πίνακα όπως και στην περίπτωση της εισαγωγής. Εάν οποιαδήποτε από τα bits σε αυτές τις θέσεις είναι 0, το στοιχείο δεν ανήκει στο σύνολο, διότι στην αντίθετη περίπτωση όλα τα bits θα είχαν τεθεί σε 1 κατά την εισαγωγή του. Εάν όλα είναι 1, τότε είτε το στοιχείο ανήκει στο σύνολο, είτε τα bits έχουν τεθεί σε 1 κατά τη διάρκεια της εισαγωγής άλλων στοιχείων (ψευδώς θετικό - false positive). Η πιθανότητα ενός false positive στην περίπτωση που υπάρχει μία (hash) συνάρτηση αντιστοίχισης είναι:

$$\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^{k} \overset{k=1}{\Longrightarrow} 1-\left(1-\frac{1}{m}\right)^{n} \approx 1-e^{n/m}$$

5.4.2 Περιγραφή του ABFCP

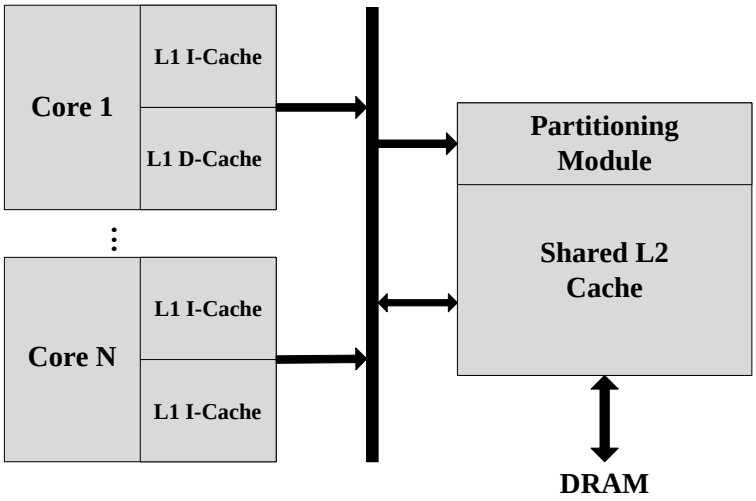
Στην υλοποίηση του ABFCP σχεδίου, ο προτεινόμενος μηχανισμός διαμοιρασμού παρακολουθεί το πραγματικό ποσοστό της κρυφής μνήμης που κατέχει κάθε εφαρμογή ή εναλλακτικά κάθε πυρήνας. Επομένως, όπως και στο UCP απαιτείται ένα πεδίο ταυτότητας πυρήνα (processor ID) για κάθε tag της κρυφής μνήμης με μέγεθος $\log_2 P$ bits όπου P το πλήθος των πυρήνων. Κάθε φορά που ένας πυρήνας εισάγει μια καταχώρηση στην κρυφή

μνήμη, η ταυτότητά του αποθηκεύεται στο αντίστοιχο πεδίο, το οποίο θα χρησιμοποιηθεί στη συνέχεια από τον αλγόριθμο καταμερισμού.

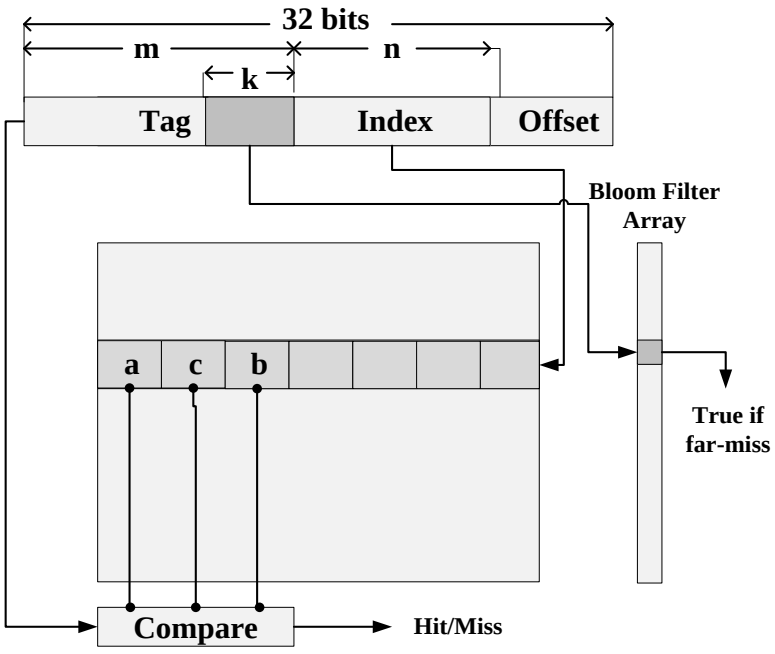
Ο ABFCP μηχανισμός διατηρεί ένα αρχείο των αστοχιών που προκαλούνται στην κρυφή μνήμη από κάθε εφαρμογή, με τη λογική ότι αυτά τα αιτήματα θα μπορούσαν να είχαν ευστοχήσει αν ο αλγόριθμος καταμερισμού της κρυφής μνήμης είχε αποδώσει περισσότερους δρόμους στη συγκεκριμένη εφαρμογή. Οι αστοχίες αυτές ονομάζονται *μακρινές αστοχίες* (Far misses), υπό την έννοια ότι ενδέχεται να απαιτούνται πολύ περισσότεροι από ένας δρόμοι επιπλέον, προκειμένου να αποφευχθούν. Το τελευταίο συνεπάγεται ένα βαθμό αβεβαιότητας ως προς το κατά πόσο η αποτροπή της αστοχίας που ανιχνεύθηκε, είναι δυνατή αλλά και ωφέλιμη για την απόδοση.

Για την υλοποίηση αυτού του αρχείου, προστέθηκε ένα Bloom φίλτρο με 2^k bits σε κάθε γραμμή της κρυφής μνήμης και για κάθε πυρήνα του επεξεργαστή. Όταν ένα tag (μιας διεύθυνσης) εκδιώκεται από την κρυφή μνήμη, τα k λιγότερο σημαντικά bits του χρησιμοποιούνται για να δεικτοδοτήσουν μέσω μίας συνάρτησης κατακερματισμού ένα bit του Bloom φίλτρου. Ακολούθως αυτό το bit τίθεται σε 1 (αληθές) ώστε να υποδηλώνει ότι κάποιο tag που εκδιώχθηκε από την αντίστοιχη γραμμή βρίσκεται εκεί.

Στην περίπτωση μιας αστοχίας στην κρυφή μνήμη, εξετάζεται αν η αντίστοιχη καταχώρηση του Bloom φίλτρου έχει τεθεί στην κατάσταση αληθές και στην περίπτωση αυτή έχει ανιχνευθεί (με κάποια πιθανότητα λάθους) μια μακρινή αστοχία όπως φαίνεται στο σχήμα 5.4 (β). Βάσει των ιδιοτήτων των Bloom φίλτρων που αναφέρθηκαν στο 5.3.1, όταν μία καταχώρηση του Bloom φίλτρου είναι στην κατάσταση ψευδές, είναι απολύτως βέβαιο ότι το tag που αντιστοιχεί σε αυτή δεν εισήχθη στο φίλτρο στο παρελθόν. Αντίθετα, αν η καταχώρηση είναι στην κατάσταση αληθές, υπάρχει η πιθανότητα ψευδώς αληθούς κατάστασης λόγω του προβλήματος που καλείται aliasing. Με άλλα λόγια, ενδέχεται το tag που προκάλεσε την αλλαγή της κατάστασης του συγκεκριμένου bit σε αληθές, να μην είναι το ίδιο με το tag που αναζητήθηκε στο Bloom φίλτρο στη συνέχεια αλλά να είναι «συνώνυμα». Με τον όρο συνώνυμα αναφερόμαστε σε δύο tags των οποίων τα k λιγότερο σημαντικά bits είναι ίδια, οπότε αντιστοιχούν στην ίδια θέση του Bloom φίλτρου και οδηγούν το σύστημα στην ανίχνευση μιας μακρινής αστοχίας χωρίς αυτή να έχει συμβεί.



(α) Επισκόπηση



(β) Κύκλωμα παρακολούθησης των αστοχιών

Σχήμα 5.4: Το ABFCP σχέδιο καταμερισμού της κρυφής μνήμης

Επιπλέον, όταν ανιχνεύεται μία μακρινή αστοχία, δεν είναι δυνατό να προσδιοριστεί με ακρίβεια πόσους ακόμα δρόμους θα έπρεπε να έχει δεσμεύσει αυτή η εφαρμογή προκειμένου να αποφευχθεί η αστοχία. Αυτό οφείλεται καταρχήν στο ότι το πλήθος των αληθών καταχωρήσεων στο Bloom φίλτρο ενδέχεται να ξεπερνά το βαθμό συσχετιστικότητας της κρυφής μνήμης. Εκτός αυτού, η σειρά με την οποία τίθενται τα bits του Bloom φίλτρου δεν καταγράφεται από το μηχανισμό, με αποτέλεσμα να μην υπάρχει

πληροφορία σχετικά με τη σειρά με την οποία τα μπλοκ απορρίπτονται από την κρυφή μνήμη ή πόσο παλιά έγινε αυτό.

Η αβεβαιότητα που εισάγεται από τα false positives, από την έλλειψη της πληροφορίας που αφορά την ηλικία των καταχωρήσεων των Bloom φίλτρων καθώς και από το πιθανά μεγάλο πλήθος αυτών, εκφράστηκε με την εισαγωγή ενός παράγοντα α. Ο παράγοντας αυτός, εκφράζει αριθμητικά την αβεβαιότητα για το πλήθος των μακρινών αστοχιών που επιστρέφουν ως τιμή οι αντίστοιχοι μετρητές, λαμβάνοντας υπόψη το πλήθος των δρόμων που είναι κατειλημμένοι από την εφαρμογή και το συνολικό πλήθος δρόμων που διαθέτει η κρυφή μνήμη. Κατ' αυτόν τον τρόπο προσεγγίζεται το πλήθος των πραγματικών αστοχιών που μπορούν να αποφευχθούν και αυτό χρησιμοποιείται από τον αλγόριθμο διαμοιρασμού.

Επειδή η LRU πολιτική υπακούει στην ιδιότητα της στοίβας, οι ευστοχίες που μετρώνται στις τελευταίες θέσεις τοπικότητας (recency positions) της κρυφής μνήμης που έχει δεσμευτεί για κάθε εφαρμογή θα γίνουν αστοχίες, αν δεσμευτούν λιγότεροι δρόμοι για τη συγκεκριμένη εφαρμογή. Η απόφαση για τον καταμερισμό της κρυφής μνήμης βασίζεται αφενός στη μέτρηση των μακρινών αστοχιών πολλαπλασιασμένη με τον παράγοντα α και αφετέρου στη μέτρηση των ευστοχιών στην ελάχιστα πρόσφατα χρησιμοποιημένη θέση όπως την καθορίζει η LRU πολιτική.

Ο μηχανισμός που διαμοιράζει την κρυφή μνήμη, αλλάζει τον τρόπο λειτουργίας της LRU πολιτικής, ώστε να λαμβάνεται υπόψη το πλήθος των μπλοκ που κατέχει κάθε εφαρμογή σε κάθε γραμμή της κρυφής μνήμης, σε αντιστοιχία με το UCP. Σε περίπτωση αστοχίας, επιλέγεται για αντικατάσταση το ελάχιστα πρόσφατα χρησιμοποιημένο μπλοκ της εφαρμογής που καταλαμβάνει εκείνη τη στιγμή περισσότερους ή ίσους δρόμους με αυτούς που υποδεικνύει ο αλγόριθμος καταμερισμού για αυτή. Διαφορετικά, απορρίπτεται το ελάχιστα πρόσφατα χρησιμοποιημένο μπλοκ ενός άλλου νήματος που καταλαμβάνει εκείνη τη στιγμή περισσότερους δρόμους από όσους καθορίζει ο αλγόριθμος.

Αναφορικά με τον αλγόριθμο καταμερισμού πρέπει να διευκρινιστεί ότι κάθε φορά που εκτελείται έχει τη δυνατότητα να μεταβάλλει τη δέσμευση δρόμων για κάθε εφαρμογή κατά ±1 δρόμους ως προς την τρέχουσα κατάσταση (διαμοιρασμό). Στην αρχή της εκτέλεσης η κρυφή μνήμη κατανέμεται ανάμεσα στις εκτελούμενες εφαρμογές σε ίσα μέρη. Κάθε T κύκλους εκτέλεσης ο αλγόριθμος καταμερισμού εκτελείται και επαναπροσδιορίζει τον αριθμό των δρόμων που δεσμεύονται από κάθε εφαρμογή για όλα τα σύνολα της κρυφής

μνήμης. Το συνολικό κέρδος σε απόδοση για κάθε πιθανό καταμερισμό υπολογίζεται από την πληροφορία που παρέχουν οι μετρητές των ευστοχιών και μακρινών αστοχιών και επιλέγεται αυτός που παρουσιάζει το μεγαλύτερο κέρδος. Σε κάθε περίπτωση εξασφαλίζεται ότι οποιαδήποτε εφαρμογή θα δεσμεύει τουλάχιστον ένα δρόμο.

Στο (14) εξετάστηκε η βέλτιστη περίοδος εκτέλεσης του αλγορίθμου. Καταρχήν η επιλογή μιας πολύ σύντομης περιόδου εκτέλεσης θα συντελούσε στη συλλογή ανεπαρκούς πληροφορίας όσον αφορά τις LRU ευστοχίες και τις μακρινές αστοχίες, συνεπώς θα καθιστούσε δυσχερέστερη τη λήψη ορθών αποφάσεων ως προς τον καταμερισμό των δρόμων της κρυφής μνήμης. Επιπλέον η εκτέλεση του αλγορίθμου προσθέτει υπολογιστικό κόστος στη λειτουργία του ABFCP σχεδίου, το οποίο είναι αντιστρόφως ανάλογο της περιόδου εκτέλεσής του, γεγονός που συνιστά έναν ακόμη λόγο για την αποφυγή επιλογής μικρής περιόδου εκτέλεσης. Συνυπολογίζοντας τους παράγοντες αυτούς, οι Nikas και συνεργάτες κατέληξαν στο ότι μια λογική τιμή είναι οι 1,000,000 κύκλοι.

Τέλος, στο (14) ελέγχθηκε η ευαισθησία του σχεδίου ως προς το μέγεθος των Bloom φίλτρων και προέκυψε το συμπέρασμα ότι η μείωση του δείκτη (index) από 15 σε 5 bits δεν είχε σχεδόν καμία επίπτωση στην απόδοση του μηχανισμού καταμερισμού της κρυφής μνήμης. Το τελευταίο αποδόθηκε σε ένα βαθμό και στην ύπαρξη του παράγοντα α, που περιορίζει τις συνέπειες των false positives. Συνεπώς το κάθε φίλτρο έχει μέγεθος μόνο 32 bits γεγονός που είναι κρίσιμο για την υλοποίηση του σχεδίου καταμερισμού, καθώς μειώνει δραματικά το επιπλέον κόστος σε υλικό (hardware overhead).

Κεφάλαιο 6

Πειραματική Αξιολόγηση των Σχεδίων Καταμερισμού

6.1 Γενικά

Στο κεφάλαιο αυτό παρουσιάζονται τα πειραματικά αποτελέσματα που αφορούν τα UCP και ABFCP σχέδια. Σε προηγούμενες μελέτες, το UCP σχέδιο (13) αξιολογήθηκε με βάση τα μετροπρογράμματα της σουίτας SPEC CPU2000, ενώ το ABFCP σχέδιο (14), (28), αξιολογήθηκε σύμφωνα με μετροπρογράμματα που λήφθηκαν από τις σουίτες Java Grande και NasaHP. Στην εργασία αυτή χρησιμοποιούνται μετροπρογράμματα της σουίτας SPEC CPU2006, καθώς όπως αναφέρθηκε και στο κεφάλαιο 4 είναι πιο καινούρια και αντιπροσωπεύουν καλύτερα τους φόρτους εργασίας των σύγχρονων εφαρμογών.

Οι πειραματικές μετρήσεις πραγματοποιήθηκαν για συστήματα με 2 και 4 πυρήνες και για την παρακάτω διαμόρφωση μνήμης.

Ιδιωτική Κρυφή Μνήμη Πρώτου Επιπέδου (1 Εντολών και 1 Δεδομένων)	
Μέγεθος	32 Kbytes
Βαθμός Συσχετιστικότητας	4 δρόμων
Πλήθος Συνόλων/Γραμμών	128
Μέγεθος Καταχώρησης/Μπλοκ	64 bytes
Διαμοιραζόμενη Κρυφή Μνήμη Δευτέρου Επιπέδου	
Μέγεθος	4 Mbytes
Βαθμός Συσχετιστικότητας	16 δρόμων
Πλήθος Συνόλων/Γραμμών	4096
Μέγεθος Καταχώρησης/Μπλοκ	64 bytes

Πίνακας 6.1: Διαμόρφωση Κρυφής Μνήμης

Η μετρική που χρησιμοποιείται για την παρουσίαση των αποτελεσμάτων και την αξιολόγηση των σχεδίων είναι η *μετρική της επιτάχυνσης με βάρη* (Weighted Speedup metric), η οποία δίνεται από τον τύπο:

$$Weighted\ Speedup = \sum_{i=1}^{N} \frac{IPC_i}{Single\ IPC_i}$$

όπου N το πλήθος των εφαρμογών του φόρτου εργασίας που εκτελείται, IPC_i το πλήθος των εντολών που εκτελούνται ανά κύκλο για την εφαρμογή i όταν εκτελείται ως μέρος του φόρτου εργασίας και Single IPC_i το πλήθος των εντολών που εκτελούνται ανά κύκλο όταν η εφαρμογή εκτελείται μόνη της. Η Weighted Speedup μετρική έχει το πλεονέκτημα ότι εκφράζει δικαιότερα την επίδραση της κάθε εφαρμογής στη μετρούμενη επιτάχυνση, σε σχέση με άλλες μετρικές όπως η *μετρική του αθροίσματος των IPC* (IPC Sum metric).

Στα ραβδογράμματα που παρουσιάζονται για τα σχέδια που αξιολογούνται, κάθε ράβδος εκφράζει ένα λόγο που αντιστοιχεί σε έναν από τους φόρτους εργασίας. Ο λόγος αυτός προκύπτει από τη διαίρεση των τιμών που προσδιορίζονται με τη Weighted Speedup μετρική για το εκάστοτε σχέδιο διαμοιρασμού και για την LRU πολιτική, δηλαδή:

$$Weighted\ Speedup\ Fraction = \frac{(Weighted\ Speedup)_{Partioning\ \ Scheme}}{(Weighted\ Speedup)_{LRU}}$$

για κάθε φόρτο εργασίας. Τέλος κάθε διάγραμμα περιλαμβάνει και μια ράβδο που εκφράζει το γεωμετρικό μέσο (geometric mean) των λόγων όλων των φόρτων εργασίας του ραβδογράμματος. Ο γεωμετρικός μέσος προκύπτει από τον τύπο:

$$Geometric\ Mean = \sqrt[N]{\prod_{i=1}^{N} Weighted\ Speedup\ Fraction}$$

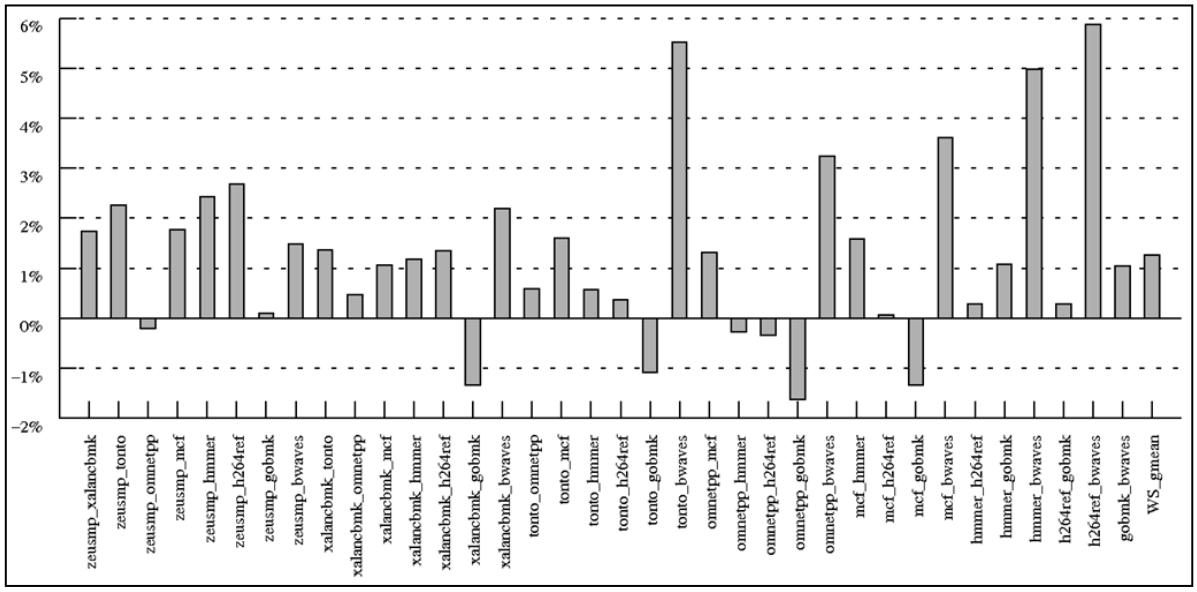
όπου N το πλήθος των φόρτων εργασίας που περιλαμβάνονται στο διάγραμμα.

Το κεφάλαιο ολοκληρώνεται με μία σύγκριση των UCP και ABFCP σχεδίων διαμοιρασμού της κρυφής μνήμης, που οδηγεί στην εξαγωγή ορισμένων χρήσιμων συμπερασμάτων. Τα συμπεράσματα αυτά θα χρησιμοποιηθούν για τη διατύπωση σχεδιαστικών τροποποιήσεων που μπορούν να συντελέσουν δυνητικά στη δημιουργία ενός αποδοτικότερου σχεδίου καταμερισμού της διαμοιραζόμενης κρυφής μνήμης.

6.2 Αξιολόγηση UCP Σχεδίου

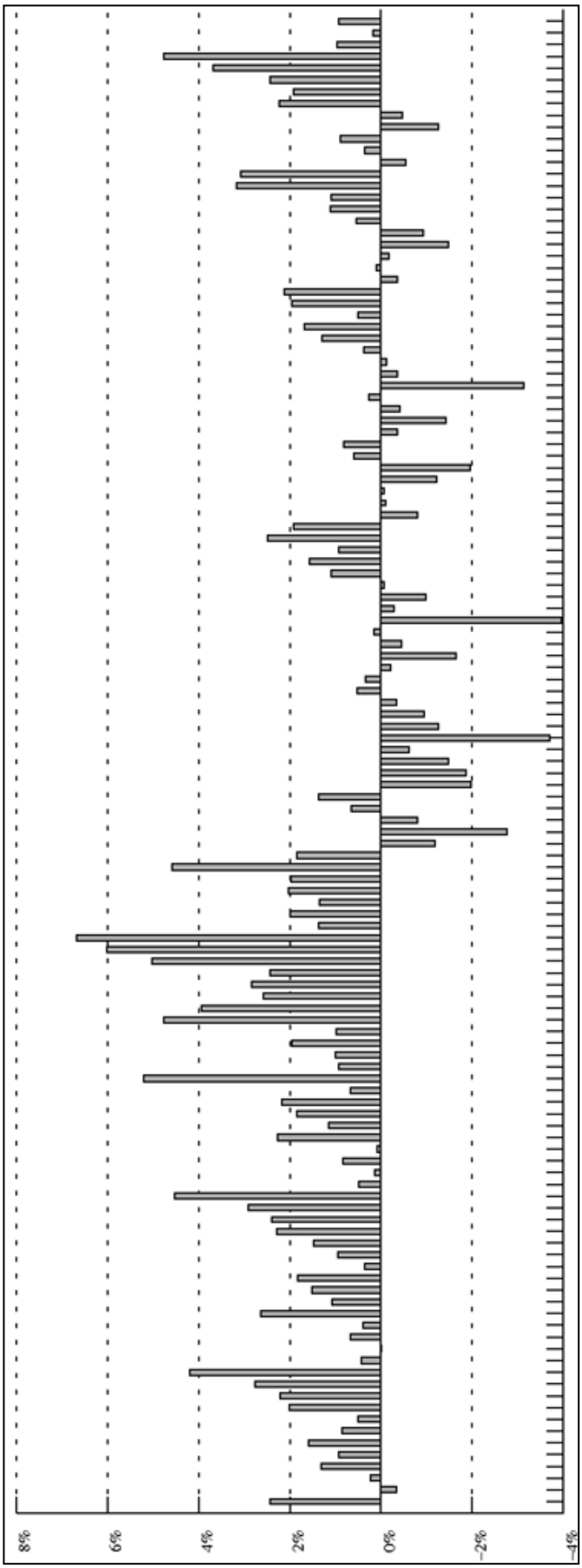
Το UCP σχέδιο αξιολογήθηκε προσομοιώνοντας συστήματα με 2 και 4 πυρήνες που ενσωμάτωναν διαμοιραζόμενη κρυφή μνήμη δευτέρου επιπέδου μεγέθους 4 Mbytes και 16 δρόμων. Τα αποτελέσματα για τις προσομοιώσεις με τους δύο πυρήνες δείχνουν ότι το σχέδιο κατορθώνει να αυξήσει την απόδοση σε σχέση με την LRU πολιτική για την

πλειοψηφία των φόρτων εργασίας. Ειδικότερα παρουσιάζει χειρότερη απόδοση σε σχέση με την LRU πολιτική μόνο για 7 από τους 36 φόρτους εργασίας, από τους οποίους μόνο σε 3 υστερεί της LRU περισσότερο από 1%. Στη χειρότερη περίπτωση παρουσιάζει μείωση 1,7% για το συνδυασμό omnetpp-gobmk. Οι 18 από τους 36 συνδυασμούς εμφανίζουν βελτιωμένη απόδοση τουλάχιστον κατά 1%, ενώ η μέγιστη βελτίωση (6%) επιτυγχάνεται για το συνδυασμό h264ref-bwaves. Κατά μέσο όρο το UCP βελτιώνει την απόδοση ενός συστήματος κατά 1,2 % σε σχέση με την LRU πολιτική στις προσομοιώσεις με 2 πυρήνες.



Σχήμα 6.1: Αποτελέσματα από τις προσομοιώσεις με 2 πυρήνες για το UCP σχέδιο

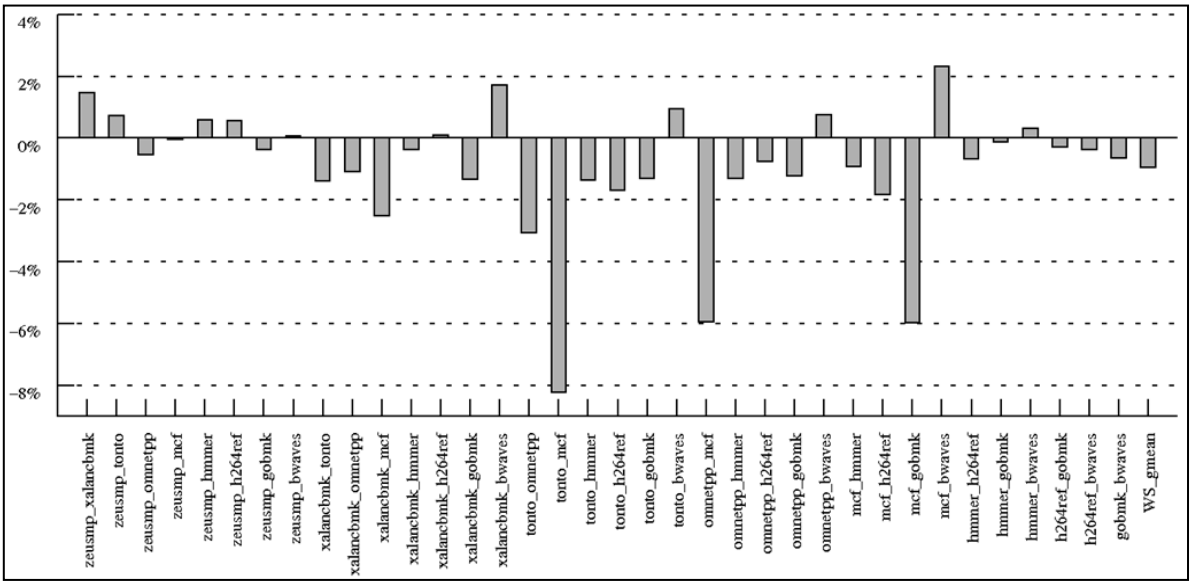
Στις προσομοιώσεις με τους 4 πυρήνες, όπου ο ανταγωνισμός είναι μεγαλύτερος καθώς περισσότερες εφαρμογές διεκδικούν τον ίδιο χώρο της κρυφής μνήμης, το UCP παρουσιάζει παρόμοια συμπεριφορά σε σχέση με τις προσομοιώσεις με τους 2 πυρήνες. Για την πλειοψηφία των περιπτώσεων επιτυγχάνει καλύτερη απόδοση σε σύγκριση με την LRU πολιτική. Συγκεκριμένα για 30 από τους 126 συνδυασμούς μετροπρογραμμάτων παρουσιάζει τουλάχιστον 2% βελτίωση, ενώ η μέγιστη βελτίωση είναι 6,6% και επιτυγχάνεται για τον συνδυασμό bwaves-mcf-omnetpp-zeusmp. Βεβαίως και πάλι υπάρχουν κάποιες περιπτώσεις (37 από τις 126) στις οποίες το UCP επιτυγχάνει χειρότερη απόδοση από την LRU. Στις περισσότερες περιπτώσεις από αυτές η πτώση της απόδοσης είναι λιγότερη από 2%, ενώ η μέγιστη πτώση είναι -4 % για τον συνδυασμό gobmk-hmmer-omnetpp-tonto. Γενικά, κατά μέσο όρο η επίδοση ενός συστήματος που ενσωματώνει το UCP σχέδιο είναι κατά 1,1% καλύτερη από ένα σύστημα που διαχειρίζεται από την LRU πολιτική.



Σχήμα 6.2: Αποτελέσματα από τις προσομοιώσεις με 4 πυρήνες για το UCP σχέδιο

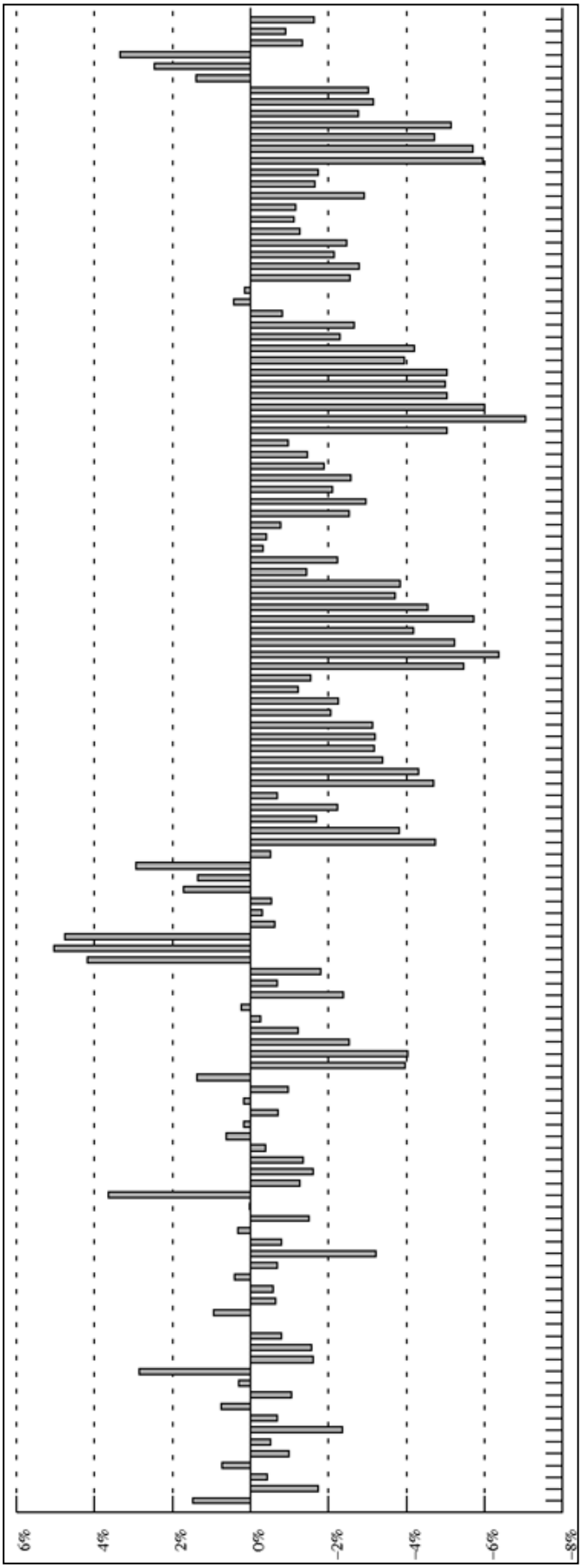
6.3 Αξιολόγηση ABFCP Σχεδίου

Όπως και το UCP έτσι και το ABFCP σχέδιο αξιολογήθηκε προσομοιώνοντας συστήματα με 2 και 4 πυρήνες με την ίδια διαμόρφωση για την κρυφή μνήμη. Από τα αποτελέσματα φαίνεται ότι το ABFCP σχέδιο δε λειτουργεί πολύ καλά στις συγκεκριμένες εφαρμογές καθώς δεν κατορθώνει να αυξήσει την απόδοση σε σχέση με την LRU πολιτική στην πλειοψηφία των περιπτώσεων, όπως συνέβαινε στο (28) και στο (14). Ειδικότερα στις προσομοιώσεις των συστημάτων με 2 πυρήνες, το ABFCP βελτιώνει την απόδοση για 11 από τους 36 φόρτους εργασίας και μόνο στους 3 από αυτούς η βελτίωση ξεπερνά το 1%. Η μεγαλύτερη αύξηση απόδοσης είναι 2,3% και αφορά το συνδυασμό μετροπρογραμμάτων mcf-bwaves. Αντίθετα σε αρκετές περιπτώσεις η πτώση της απόδοσης είναι σημαντική και στη χειρότερη περίπτωση φτάνει το -8,2% για το φόρτο εργασίας tonto-mcf. Ο γεωμετρικός μέσος της αύξησης της απόδοσης είναι αρνητικός και κυμαίνεται στο -1% περίπου.



Σχήμα 6.3: Αποτελέσματα από τις προσομοιώσεις με 2 πυρήνες για το ABFCP σχέδιο

Στις προσομοιώσεις με 4 πυρήνες, για την πλειοψηφία των περιπτώσεων το ABFCP επιτυγχάνει χαμηλότερη απόδοση σε σύγκριση με την LRU πολιτική. Συγκεκριμένα παρουσιάζει βελτίωση για 25 από τους 126 συνδυασμούς μετροπρογραμμάτων, ενώ η μέγιστη βελτίωση είναι 4,8% και επιτυγχάνεται και πάλι για τον συνδυασμό bwaves-mcf-omnetpp-zeusmp. Η μείωση της απόδοσης είναι περισσότερο από 2% σε πολλές περιπτώσεις, ενώ η μέγιστη πτώση είναι -7% για τον συνδυασμό h264ref-hmmer-mcf-tonto. Κατά μέσο όρο η επίδοση ενός συστήματος που ενσωματώνει το ABFCP σχέδιο είναι κατά 1,6% χειρότερη από ένα σύστημα που διαχειρίζεται από την LRU πολιτική.



Σχήμα 6.4: Αποτελέσματα από τις προσομοιώσεις με 4 πυρήνες για το ABFCP σχέδιο

6.4 Σύγκριση των UCP και ABFCP Σχεδίων

Οι Qureshi και συνεργάτες καθώς και οι Nikas και συνεργάτες πρότειναν δύο προσεγγίσεις για το δυναμικό καταμερισμό του τελευταίου επιπέδου της διαμοιραζόμενης κρυφής μνήμης σε πολυπύρηνες αρχιτεκτονικές. Οι προσεγγίσεις αυτές, είχαν ως σκοπό τον ορθολογικότερο καταμερισμό των πόρων της μνήμης με κριτήριο όχι τη ζήτηση που εμφανίζει κάθε εφαρμογή αλλά σύμφωνα με το όφελος που θα αποκομίσει από τη δέσμευση περισσότερων από αυτούς. Και οι δύο προσεγγίσεις, όφειλαν να συμβιβάσουν την ακρίβεια και την πληρότητα της πληροφορίας που συνέλεγαν με το περιορισμένο κόστος σε υλικό.

Η πλέον χαρακτηριστική διαφορά των ABFCP και UCP σχεδίων είναι ότι το πρώτο έχει τη δυνατότητα να καταμερίζει ξεχωριστά τους δρόμους σε κάθε σύνολο, ενώ το δεύτερο διαχειρίζεται την κρυφή μνήμη με ενιαίο τρόπο και εφαρμόζει τον ίδιο καταμερισμό σε όλα τα σύνολά της. Με βάση τα σχήματα (4.1 έως 4.5) που παρουσιάστηκαν στο κεφάλαιο 4 για τα μοτίβα πρόσβασης των εφαρμογών στην κρυφή μνήμη καθώς και τα συμπεράσματα που εξήχθησαν για τον τρόπο χρήσης της κρυφής μνήμης από τις εφαρμογές, αναμενόταν η ευελιξία του ABFCP στον καταμερισμό (κάθε γραμμής) να συντελούσε σε αυξημένη απόδοση έναντι της LRU πολιτικής αντικατάστασης και ενδεχομένως και του UCP σχεδίου.

Εντούτοις τα πειραματικά αποτελέσματα διέψευσαν την προσδοκία αυτή. Από τις προσομοιώσεις που πραγματοποιήθηκαν καθίσταται σαφές ότι το UCP σχέδιο έχει την ικανότητα να ανταποκρίνεται καλύτερα και να διαμοιράζει αποδοτικότερα την κρυφή μνήμη τόσο στις περιπτώσεις με 2 όσο και με 4 πυρήνες. Το γεγονός αυτό οφείλεται πιθανότατα σε δύο κύριους λόγους.

Ο πρώτος λόγος είναι η καθαρότητα της πληροφορίας, η οποία έγκειται στα μετρούμενα μεγέθη που χρησιμοποιούνταν για να αποτυπώνουν το όφελος ή την απώλεια που θα προκαλούσε στη συνολική απόδοση, ένας διαφορετικός καταμερισμός της κρυφής μνήμης σε σχέση με αυτόν που υποδείκνυε η LRU πολιτική αντικατάστασης μπλοκ. Στο UCP σχέδιο, όπου ουσιαστικά γίνεται προσομοίωση μιας ιδιωτικής κρυφής μνήμης για κάθε πυρήνα, η πληροφορία που συλλέγεται είναι απαλλαγμένη από συγκρούσεις μεταξύ των παράλληλα εκτελούμενων νημάτων. Αυτή η καθαρότητα της πληροφορίας αποτελεί πλεονέκτημα που βοηθά το UCP σχέδιο να αποδώσει καλύτερα.

Ο δεύτερος λόγος σχετίζεται με τη δυνατότητα του UCP σχεδίου να μεταβάλλει το πλήθος των δρόμων που αποδίδει σε κάποια διεργασία κατά ένα εύρος ±N (maximum). Αντίθετα το ABFCP μπορεί να μεταβάλλει τη δέσμευση δρόμων για κάθε νήμα μόνο κατά ±1 κάθε φορά που εκτελείται ο αλγόριθμος. Όπως φαίνεται από τα αποτελέσματα ο περιορισμός αυτός στερεί από το ABFCP τη δυνατότητα να ανταποκρίνεται αποδοτικά στις μεταβολές των απαιτήσεων των εφαρμογών κατά τη διάρκεια της εκτέλεσης τους.

Στο επόμενο κεφάλαιο παρουσιάζεται ένα σχέδιο που επιχειρεί να βελτιώσει την απόδοση του ABFCP αυξάνοντας το εύρος μεταβολής της δέσμευσης δρόμων από τα νήματα. Συγκεκριμένα οι πιθανοί διαμοιρασμοί προκύπτουν από τον τρέχοντα (διαμοιρασμό) με αυξομειώσεις κατά ±3 δρόμους για κάθε εφαρμογή. Επίσης, προφανώς διατηρείται η ευελιξία του ABFCP ανά γραμμή της κρυφής μνήμης η οποία είναι χρήσιμη όπως υποδεικνύουν τα διαγράμματα του κεφαλαίου 4.

Κεφάλαιο 7

Μελέτη Πιθανών Τροποποιήσεων

7.1 Γενικά

Η αξιολόγηση των πειραματικών μετρήσεων στο κεφάλαιο 6, οδήγησε στη διατύπωση ορισμένων χρήσιμων συμπερασμάτων. Επιπλέον η σύγκριση των ABFCP και UCP σχεδίων καταμερισμού, συνετέλεσε στην αναγνώριση των χαρακτηριστικών που πρέπει να βελτιωθούν προκειμένου να προκύψει ένα αποδοτικότερο σχέδιο καταμερισμού της κρυφής μνήμης. Με βάση αυτά τα συμπεράσματα και τις παρατηρήσεις αποφασίστηκε η τροποποίηση του ABFCP σχεδίου και το σχέδιο που ενσωματώνει τις σχεδιαστικές προτάσεις περιγράφεται σε αυτό το κεφάλαιο.

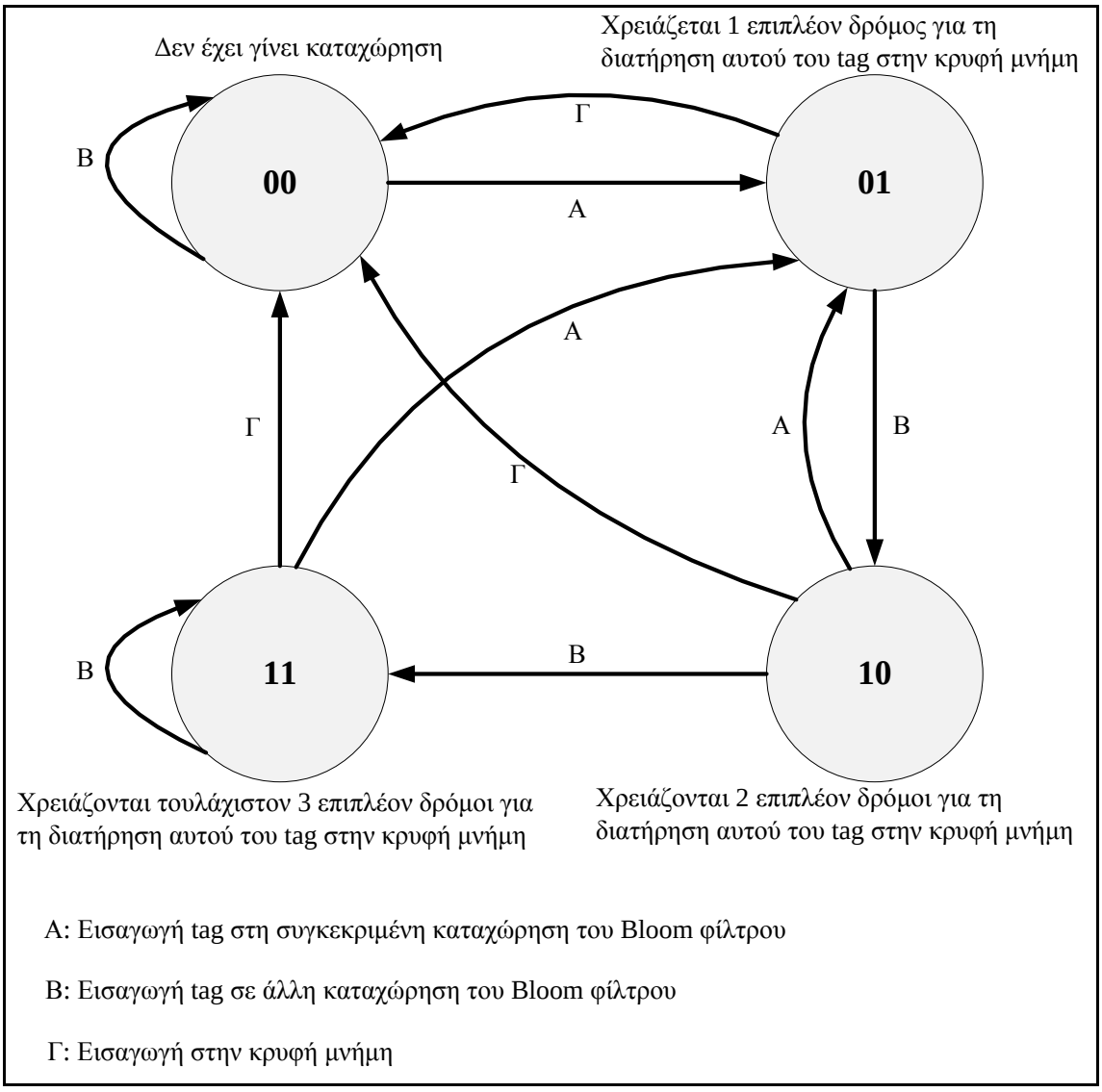
7.2 Περιγραφή ενός Βελτιωμένου Σχεδίου Διαμοιρασμού της Κρυφής Μνήμης

Στο κεφάλαιο 5.3.2 αναλύθηκαν οι λόγοι για τους οποίους οι Nikas και συνεργάτες επέλεξαν την υιοθέτηση Bloom φίλτρων για την καταγραφή των αστοχιών στο ABFCP σχέδιο. Επίσης διαπιστώθηκε πως ένα από τα εγγενή μειονεκτήματα αυτής της επιλογής είναι η απουσία πληροφορίας για το πόσους ακόμα δρόμους θα χρειαζόταν κάθε εφαρμογή για να αποφύγει κάποιες από τις μακρινές αστοχίες. Συγκεκριμένα, η υλοποίηση αδυνατεί να ανιχνεύσει ότι ορισμένες αστοχίες δε θα γίνουν ευστοχίες ακόμα και αν η εφαρμογή δεσμεύσει ολόκληρο το διαθέσιμο χώρο της κρυφής μνήμης.

Το μειονέκτημα αυτό αποτέλεσε το κύριο κίνητρο για την προτεινόμενη τροποποίηση του ABFCP σχεδίου. Το ABFCP χρησιμοποιεί τις μακρινές αστοχίες σαν εκτίμηση του κέρδους σε περίπτωση δέσμευσης περισσότερων δρόμων και τις ευστοχίες στην ελάχιστα πρόσφατα χρησιμοποιημένη (LRU) θέση ως εκτίμηση για το κόστος σε περίπτωση απώλειας ενός δρόμου. Λόγω της σχεδίασης του μηχανισμού οι μεταβολές περιορίζονται στους ±1 δρόμους. Για την χαλάρωση του περιορισμού αυτού προτείνεται η επέκταση των Bloom φίλτρων ώστε να καταστεί δυνατή η αποθήκευση πληροφορίας σε σχέση με την ηλικία των

αστοχιών, δηλαδή με την απόσταση που έχουν διανύσει στην LRU στοίβα από τη στιγμή που εκδιώχθηκαν από την κρυφή μνήμη και εισήχθησαν στο Bloom φίλτρο.

Ειδικότερα, η προτεινόμενη τροποποίηση είναι η προσθήκη κάποιων bit ιστορίας σε κάθε καταχώρηση ενός Bloom φίλτρου. Γενικά, n bits παρέχουν τη δυνατότητα προσομοίωσης 2^n-1 επιπλέον δρόμων, δεδομένου ότι μία από τις καταστάσεις είναι η κενή και οι υπόλοιπες υποδηλώνουν πόσες θέσεις έχει μεταθέσει η LRU πολιτική αντικατάστασης το tag που έχει εκδιωχθεί από την κρυφή μνήμη και βρίσκεται στο Bloom φίλτρο. Η τελευταία κατάσταση περιλαμβάνει όλους τους δρόμους που είναι $\geq 2^n$-1, αφού ο μετρητής που χρησιμοποιείται είναι saturated. Επίσης εκμεταλλευόμαστε και πάλι την ιδιότητα της στοίβας και χρησιμοποιούμε 2^n-1 μετρητές ευστοχιών, οι οποίοι παρέχουν μια εκτίμηση του κόστους που συνεπάγεται η απώλεια αντίστοιχων πόρων.



Σχήμα 7.1: Διάγραμμα καταστάσεων των καταχωρήσεων των Bloom φίλτρων

Στο σχήμα 7.1 παρουσιάζεται ένα παράδειγμα της προτεινόμενης τροποποίησης με n=2 bits ιστορίας. Όπως φαίνεται στο σχήμα 7.1 τα bits βρίσκονται αρχικά στην τιμή 00 και τίθενται σε 01 για μια συγκεκριμένη καταχώρηση όταν ένα στοιχείο εισέλθει στο Bloom φίλτρο, δηλαδή όταν ένα tag που έχει εκδιωχθεί από την κρυφή μνήμη εισέλθει στην καταχώρηση του Bloom φίλτρου. Σε κάθε περίπτωση, όταν γίνεται εισαγωγή κάποιου στοιχείου στο Bloom φίλτρο, τα bits ιστορίας όλων των καταχωρήσεων (πλην αυτής όπου γίνεται η εισαγωγή) που έχουν τεθεί (δηλαδή είναι 01, 10 ή 11) αυξάνονται, προσομοιώνοντας έτσι τη λειτουργία της LRU πολιτικής αντικατάστασης που μετατοπίζει τα μπλοκ προς την LRU θέση κάθε φορά που εισάγει ένα νέο μπλοκ στην MRU θέση της κρυφής μνήμης. Αντιθέτως η καταχώρηση όπου γίνεται η εισαγωγή επιστρέφει στην τιμή 01 αν βρισκόταν σε κάποια άλλη κατάσταση πλην της 00, με τη λογική ότι εισήχθη κάποιο άλλο tag στην ίδια θέση.

Συμπληρώνοντας την περιγραφή του τρόπου με τον οποίο προσομοιώνονται οι επιπλέον δρόμοι της κρυφής μνήμης, επισημαίνεται ότι όταν ένα μπλοκ που έχει εκδιωχθεί στο παρελθόν από αυτή αναζητηθεί από τον μηχανισμό διαμοιρασμού και βρεθεί μέσα στο Bloom φίλτρο, τότε η αντίστοιχη καταχώρηση του Bloom φίλτρου μηδενίζεται. Αυτό συμβαίνει διότι θεωρείται ότι συνέβη μια ευστοχία, δηλαδή το μπλοκ που είχε εκδιωχθεί βρέθηκε σε κάποιον από τους επιπλέον δρόμους που περιλαμβάνει το Bloom φίλτρο και εισήχθη ξανά στην κρυφή μνήμη. Σε αυτή την περίπτωση αυξάνεται ο κατάλληλος μετρητής με βάση την τιμή που έχει η καταχώρηση που προκάλεσε την ευστοχία. Έτσι επιτυγχάνεται ακριβέστερη παρακολούθηση των αστοχιών.

Το προτεινόμενο σύστημα που περιγράφηκε σε αυτή την ενότητα ονομάζεται Adaptive Bloom Filter Cache Partitioning with History και θα αναφέρεται στο εξής ως ABFCPH.

7.3 Υλοποίηση

Για την υλοποίηση και αξιολόγηση του προτεινόμενου σχεδίου επιλέξαμε το μέγεθος n των μετρητών των Bloom filters που αναφέρθηκαν στην προηγούμενη ενότητα να είναι ίσο με 2. Αυτό σημαίνει ότι απαιτούνται τρεις μετρητές ευστοχιών σε κάθε Bloom φίλτρο αλλά και τρεις μετρητές ευστοχιών στις αντίστοιχες LRU θέσεις των συνόλων της κρυφής μνήμης. Η σημασία των μετρητών, δηλαδή η πληροφορία που συλλέγουν φαίνεται στον πίνακα 7.1.

Πληροφορία που Περιέχει κάθε Μετρητής						
Μετρητής	(LRU-2) μετρητής	(LRU-1) μετρητής	LRU μετρητής	(LRU+1) μετρητής	(LRU+2) μετρητής	(LRU+k) μετρητής (k≥3)
Πληροφορία	Αστοχίες που θα προκληθούν από την αποδέσμευση 3 δρόμων	Αστοχίες που θα προκληθούν από την αποδέσμευση 2 δρόμων	Αστοχίες που θα προκληθούν από την αποδέσμευση 1 δρόμου	Ευστοχίες που θα προκύψουν από τη δέσμευση 1 επιπλέον δρόμου	Ευστοχίες που θα προκύψουν από τη δέσμευση 2 επιπλέον δρόμων	Ευστοχίες που πιθανώς θα προκύψουν από τη δέσμευση 3 επιπλέον δρόμων

Πίνακας 7.1: Ρόλος κάθε μετρητή στο ABFCPH σχέδιο

7.4 Κόστος και Πολυπλοκότητα

Το συνολικό επιπλέον κόστος σε υλικό παρουσιάζεται στον παρακάτω πίνακα και συγκρίνεται με το hardware overhead της αρχικής υλοποίησης για την περίπτωση μιας διαμοιραζόμενης συσχετιστικής κρυφής μνήμης 4 Mbytes και 16 δρόμων, σε έναν επεξεργαστή με τέσσερις πυρήνες.

Κόστος Υλικού σε Επεξεργαστή με 4 Πυρήνες (P=4)			
Μονάδα	**Κόστος σε Υλικό (bytes)**	**ABFCP (N=1)**	**ABFCPH (N=3)**
Bloom Φίλτρο	P * Sets * (BF bits/8)	4 * 4096 * 4 bytes = 64 Kbytes	4 * 4096 * 8 bytes = 128Kbytes
Counters	P * 2 * N * Sets	4 * 2 * 1 * 4096 bytes = 32 Kbytes	4 * 2 * 3 * 4096 bytes = 96 Kbytes
Core-id Field	Sets * ($\log_2$P/8) * associativity	4096 * ¼ * 16 bytes = 32 Kbytes	4096 * ¼ * 16 bytes = 32 Kbytes
Σύνολο		128 Kbytes	256 Kbytes
L2 Hardware Overhead		3,125%	6,25%

Πίνακας 7.2: Επιπλέον κόστος σε υλικό

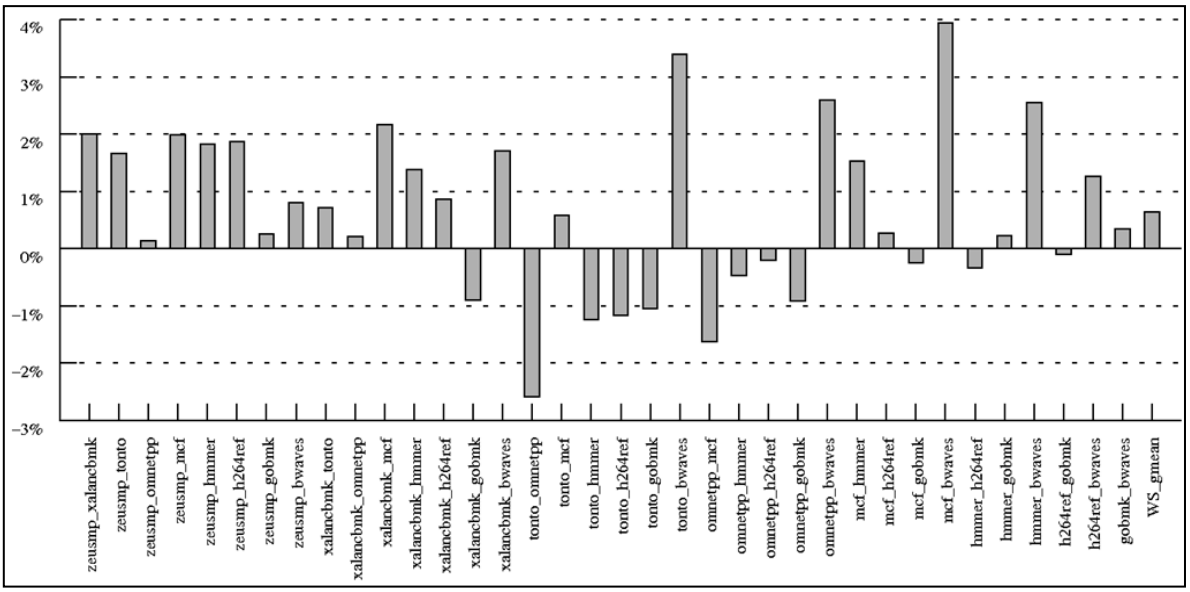
Από τον παραπάνω πίνακα είναι εμφανές ότι ABFCPH έχει διπλάσιο κόστος σε υλικό σε σχέση με την αρχική υλοποίηση και για τη συγκεκριμένη κρυφή μνήμη. Ωστόσο επειδή διαθέτει μεγαλύτερη ευελιξία, καθώς και ταχύτητα προσαρμογής στις μεταβαλλόμενες ανάγκες των εφαρμογών σε πόρους της κρυφής μνήμης, αναμένεται να παρουσιάσει καλύτερη απόδοση.

Τέλος ο αλγόριθμος καταμερισμού του ABFCPH σχεδίου είναι πιο πολύπλοκος από αυτόν του ABFCP, λόγω του ότι πρέπει να εξετάσει περισσότερες περιπτώσεις. Στην παρούσα εργασία προκειμένου να απομονώσουμε τις συνέπειες της επιλογής ενός γραμμικού αλγορίθμου που δεν εγγυάται τη βέλτιστη λύση (global optimum solution), στην αξιολόγηση χρησιμοποιούμε τον ιδανικό αλγόριθμο. Παρόλα αυτά θα μπορούσε σχετικά εύκολα να επεκταθεί ο γραμμικός αλγόριθμος που παρουσιάστηκε στο (14) και να χρησιμοποιηθεί και στο ABFCPH σχέδιο.

7.3 Πειραματική Αξιολόγηση

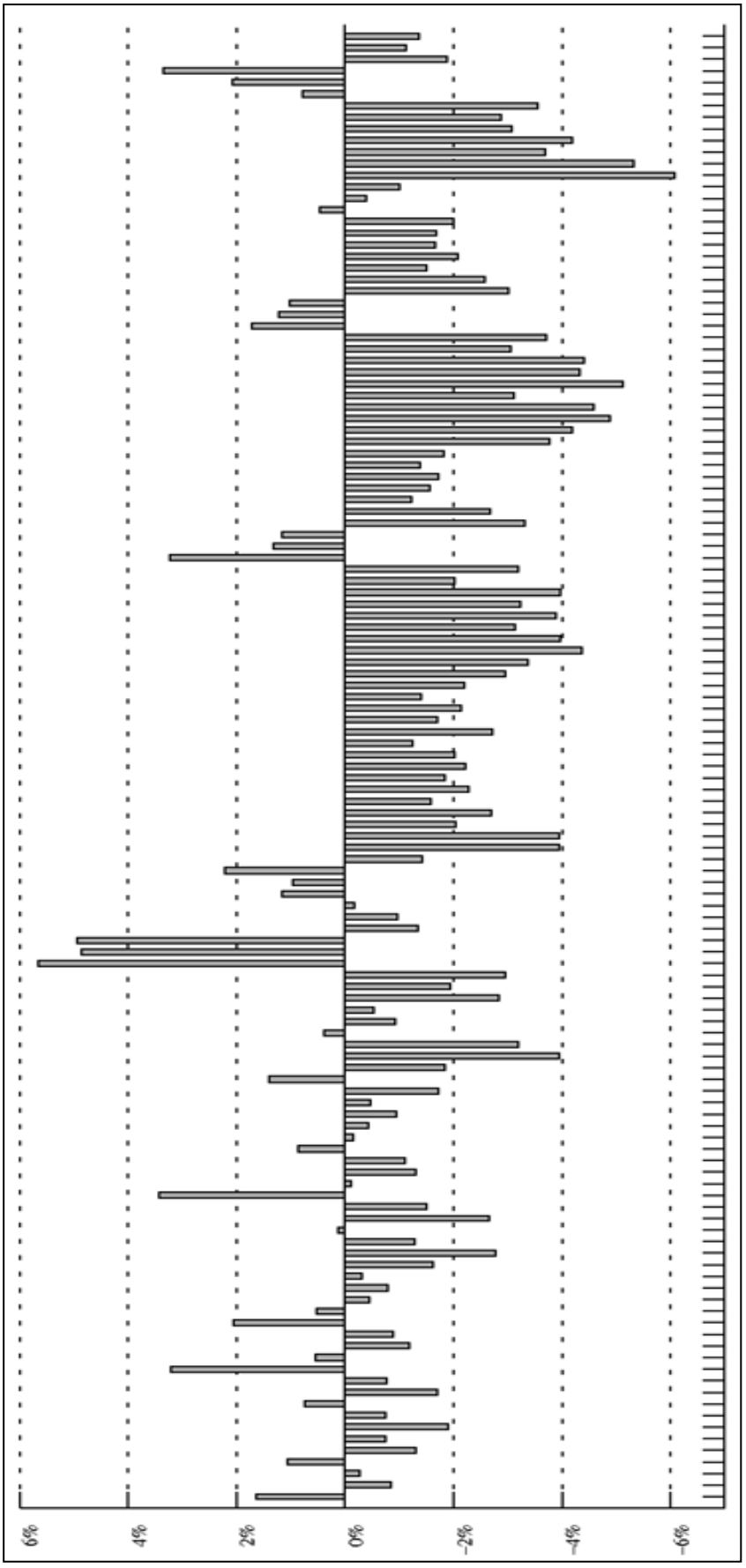
Το νέο σχέδιο αξιολογήθηκε με την ίδια διαμόρφωση συστήματος που χρησιμοποιήθηκε για το UCP και το ABFCP σχέδιο, δηλαδή προσομοιώθηκαν συστήματα με 2 και 4 πυρήνες και ίδια χαρακτηριστικά για την κρυφή μνήμη. Από τα αποτελέσματα φαίνεται ότι το ABFCPH σχέδιο κατορθώνει να βελτιώσει την απόδοση στην πλειοψηφία των περιπτώσεων για τις προσομοιώσεις με τους 2 πυρήνες. Η συνολική του απόδοση κυμαίνεται ανάμεσα σε αυτή του ABFCP και του UCP σχεδίου.

Συγκεκριμένα 24 από τους 36 συνδυασμούς μετροπρογραμμάτων παρουσιάζουν αύξηση της απόδοσης σε σχέση με την LRU πολιτική αντικατάστασης και 14 από αυτούς τουλάχιστον κατά 1%. Η μεγαλύτερη βελτίωση αφορά το mcf-bwaves φόρτο εργασίας και κυμαίνεται στο 4%. Από την άλλη το ένα τρίτο (12/36) των συνδυασμών εμφανίζει κατώτερη απόδοση σε σχέση με την LRU πολιτική αντικατάστασης. Ο συνδυασμός μετροπρογραμμάτων με τη χαμηλότερη απόδοση είναι ο tonto-omnetpp με -2,6%, ενώ η τιμή του γεωμετρικού μέσου είναι 0,6%. Τέλος αξίζει να αναφερθεί ότι το ABFCPH είναι για όλους τους συνδυασμούς (εκτός από 1) καλύτερο από το ABFCP σχέδιο ενώ ξεχωρίζουν περιπτώσεις όπως το omnetpp_mcf όπου η διαφορά είναι της τάξης του 4,4%.



Σχήμα 7.2: Αποτελέσματα από τις προσομοιώσεις με 2 πυρήνες για το ABFCPH σχέδιο

Στις προσομοιώσεις συστημάτων με 4 πυρήνες η γενική εικόνα που παρουσιάζει το νέο σχέδιο είναι κάπως διαφορετική. Για τους περισσότερους φόρτους εργασίας ο λόγος των τιμών της μετρικής του weighted speedup είναι μικρότερος της μονάδας, δηλαδή υπερέχει η LRU πολιτική. Πιο συγκεκριμένα 46 από τους 126 συνδυασμούς μετροπρογραμμάτων παρουσιάζουν πτώση απόδοσης μεγαλύτερη του -2%, ενώ η χειρότερη περίπτωση είναι ο συνδυασμός hmmer-mcf-tonto-xalancbmk με επιδείνωση της τάξης του -6,1%. Αντίθετα η απόδοση είναι αυξημένη για 28 από τις 126 περιπτώσεις και στις 19 από αυτές περισσότερο από 1%. Η καλύτερη απόδοση επιτυγχάνεται για τον bwaves-mcf-omnetpp-tonto συνδυασμό και κυμαίνεται στο 5,7%. Η συνολική εικόνα είναι και πάλι καλύτερη από αυτή που παρουσιάζει το ABFCP σχέδιο και χειρότερη από του UCP. Σε σχέση με το ABFCP, το ABFCPH βελτιώνει την απόδοση μέχρι και 3,2%, όπως συμβαίνει στην περίπτωση του gobmk-hmmer-mcf-tonto. Η τιμή του γεωμετρικού μέσου είναι λίγο χαμηλότερη από -1,3%.



Σχήμα 7.3: Αποτελέσματα από τις προσομοιώσεις με 4 πυρήνες για το ABFCPH σχέδιο

Συμπερασματικά, τα αποτελέσματα δείχνουν ότι προσθέτοντας τη δυνατότητα μεταβολής της δέσμευσης δρόμων κατά ±3 και ταυτόχρονα περιορίζοντας την αβεβαιότητα των μακρινών αστοχιών (λόγω κατάταξης των αστοχιών σε αυτές που απαιτούν 1, 2 ή ≥3 δρόμους) η απόδοση βελτιώνεται. Βεβαίως το νέο σχέδιο εξακολουθεί να μην παρουσιάζει τόσο σταθερή και ομοιόμορφη βελτίωση όσο το UCP, με την έννοια ότι αποδίδει καλύτερα μόνο σε ένα υποσύνολο των φόρτων εργασίας και όχι σε (σχεδόν) όλους. Παρόλα ταύτα, η βελτίωση της απόδοσης είναι αισθητή και σημαντική, συνεπώς αποτελεί ένδειξη ότι έχουν ανιχνευθεί σωστά τα μειονεκτήματα του ABFCP και πρέπει να επιλυθούν. Επιπροσθέτως, επειδή η βελτίωση του σχεδίου συνεπάγεται διπλασιασμό του κόστους υλοποίησης, κρίνεται σκόπιμο να αναζητηθούν μέθοδοι περιορισμού του.

Κεφάλαιο 8

Συμπεράσματα και Μελλοντική Εργασία

8.1 Γενικά

Οι εξελίξεις στη διαδικασία κατασκευής των ολοκληρωμένων κυκλωμάτων οδήγησαν στην καθιέρωση των πολυπύρηνων αρχιτεκτονικών (CMPs). Η πλειοψηφία των πολυπύρηνων συστημάτων σήμερα φέρει πολυεπίπεδη ιεραρχία μνήμης (και ειδικότερα κρυφής μνήμης) με παρόμοια χαρακτηριστικά. Συγκεκριμένα τα CMPs ενσωματώνουν ένα πρώτο επίπεδο ιδιωτικής κρυφής μνήμης εντολών και δεδομένων για κάθε πυρήνα, ενώ τα υπόλοιπα επίπεδα κρυφής μνήμης που περιλαμβάνονται στο chip είναι διαμοιραζόμενα από ένα υποσύνολο ή από το σύνολο των πυρήνων.

Οι αρχιτέκτονες υπολογιστών έχουν διεξάγει εκτεταμένες έρευνες για τη διαμόρφωση της ιεραρχίας των κρυφών μνημών στις μονοπύρηνες αρχιτεκτονικές. Ορισμένες από τις σχεδιαστικές προτάσεις στις οποίες είχαν καταλήξει αυτές οι έρευνες υιοθετήθηκαν στις πολυπύρηνες αρχιτεκτονικές, χωρίς να εξετασθεί επαρκώς η καταλληλότητά τους για τις νέες συνθήκες στις οποίες θα λειτουργήσουν. Χαρακτηριστικό παράδειγμα αυτής της τάσης αποτελεί η *πολιτική αντικατάστασης του ελάχιστα πρόσφατα χρησιμοποιημένου μπλοκ* (LRU). Συνεπώς εγείρονται ερωτήματα σχετικά με την αποδοτικότητα αυτών των επιλογών.

Μία από τις κύριες διαφοροποιήσεις των πολυπύρηνων συστημάτων που πρέπει να ληφθούν υπόψη, είναι η παράλληλη εκτέλεση πολλαπλών ετερογενών εφαρμογών. Μία πολιτική αντικατάστασης όπως η LRU θα αγνοήσει αυτή την παράμετρο και θα διαμοιράσει την κρυφή μνήμη με βάση τη συχνότητα αιτημάτων που παρουσιάζει κάθε νήμα που εκτελείται. Αυτό οφείλεται στο ότι το μπλοκ που θα εκδιωχθεί επιλέγεται χωρίς να συνυπολογίζεται ποιο νήμα το προσκόμισε στην κρυφή μνήμη ή ποιο νήμα προκάλεσε την αστοχία, με αποτέλεσμα τα νήματα με υψηλό ρυθμό προσβάσεων να εκδιώκουν τα μπλοκ των υπολοίπων (νημάτων).

Πρόσφατες έρευνες πρότειναν σχεδιαστικές λύσεις για τον αποδοτικότερο καταμερισμό των διαμοιραζόμενων κρυφών μνημών και την αύξηση της συνολικής απόδοσης. Ωστόσο, κάθε μία από αυτές παρουσιάζει πλεονεκτήματα και μειονεκτήματα και οι ερευνητές δεν έχουν καταλήξει ακόμα σε κοινά αποδεκτές σχεδιαστικές προτάσεις.

8.2 Συμπεράσματα

Στην παρούσα εργασία επιχειρήθηκε η αξιολόγηση ορισμένων σχεδίων καταμερισμού της διαμοιραζόμενης κρυφής μνήμης, ως προς την αποτελεσματικότητά τους στην κατανομή των πόρων ανάμεσα στα παράλληλα εκτελούμενα νήματα. Συγκεκριμένα στο κεφάλαιο 5 περιγράφηκαν τα UCP και ABFCP σχέδια καταμερισμού, που αποτελούν προϊόν εργασίας άλλων ερευνητικών ομάδων και αποτέλεσαν τη βάση της μελέτης. Στο κεφάλαιο 6 παρουσιάστηκαν τα πειραματικά αποτελέσματα που αφορούν αυτά τα σχέδια και τα οποία λήφθηκαν χρησιμοποιώντας μετροπρογράμματα αντιπροσωπευτικά των σύγχρονων εφαρμογών (SPEC CPU2006).

Η αξιολόγηση των αποτελεσμάτων έδειξε ότι το μεν UCP σχέδιο, βελτίωνε την απόδοση στις περισσότερες περιπτώσεις ενώ το ABFCP σχέδιο παρουσίαζε ανομοιόμορφη απόδοση η οποία συνολικά υπολειπόταν αυτής της LRU πολιτικής και του UCP σχεδίου. Ωστόσο, στο κεφάλαιο 4 τα διαγράμματα των μοτίβων πρόσβασης στην κρυφή μνήμη απέδειξαν ότι η ευελιξία σε επίπεδο γραμμών είναι σημαντική και το ABFCP σχέδιο ενσωμάτωνε εξαρχής τη δυνατότητα ξεχωριστής αντιμετώπισης κάθε συνόλου-γραμμής. Συνεπώς θεωρήθηκε δελεαστικότερη η πρόταση λύσεων για τη βελτίωση του ABFCP σχεδίου.

Στο κεφάλαιο 7 περιγράφηκαν οι προτεινόμενες τροποποιήσεις για το ABFCP σχέδιο και αξιολογήθηκε η απόδοσή του νέου μηχανισμού καταμερισμού της κρυφής μνήμης, στο ίδιο πλαίσιο προσομοίωσης με τα προηγούμενα σχέδια. Τα πειραματικά αποτελέσματα έδειξαν βελτίωση σε σχέση με το ABFCP σχέδιο, γεγονός που τεκμηρίωσε ότι η ανίχνευση των αιτιών που οδηγούν σε χαμηλή απόδοση είναι σωστή. Ωστόσο το νέο σχέδιο δεν κατόρθωσε να υπερκεράσει σε απόδοση το UCP σχέδιο καταμερισμού στην έκδοση με τον ιδανικό αλγόριθμο.

Το ABFCPH σχέδιο ενσωμάτωσε δύο βασικές τροποποιήσεις που το διαχωρίζουν από το ABFCP σχέδιο. Πρώτον αύξησε την ευελιξία σε σχέση με την αρχική υλοποίηση επιτρέποντας τη μεταβολή της δέσμευσης δρόμων κατά ±3 κάθε φορά που εκτελείται ο

αλγόριθμος διαμοιρασμού. Δεύτερον μείωσε την αβεβαιότητα που εισάγουν τα Bloom φίλτρα ως προς το πλήθος των δρόμων που απαιτούνται για την αποφυγή των αστοχιών, δεδομένου ότι οι δύο πρόσθετοι μετρητές καταγράφουν το πλήθος των αστοχιών που απαιτούν ακριβώς 1 ή 2 δρόμους επιπλέον (ο τρίτος μετρητής έχει τον ίδιο ρόλο με αυτόν του μετρητή στο ABFCP σχέδιο).

Από την άλλη οι τροποποιήσεις που υιοθετήθηκαν είχαν μία βασική επίπτωση. Η επίπτωση αυτή είναι η αύξηση του κόστους υλοποίησης του νέου σχεδίου σε σχέση με το ABFCP σχέδιο. Συγκεκριμένα το κόστος διπλασιάστηκε και δεδομένου ότι η αρχική υλοποίηση του ABFCP σχεδίου είχε σχετικά αυξημένο hardware overhead (σε σχέση με το UCP για παράδειγμα), κρίνεται σκόπιμο να εξεταστούν τρόποι περιορισμού του.

8.3 Μελλοντική Εργασία

Η παρούσα εργασία ερεύνησε ορισμένα σχέδια διαμοιρασμού της κρυφής μνήμης και την ικανότητά τους να αυξήσουν την απόδοση σε σύγχρονες πολυπύρηνες αρχιτεκτονικές. Σε αυτή την ενότητα πραγματοποιείται αναφορά σε πιθανές κατευθύνσεις για την επέκταση αυτής της έρευνας.

8.3.1 Πολυπύρηνες Αρχιτεκτονικές

Η αυξητική τάση που ακολουθεί ο αριθμός των πυρήνων στα CMPs αναμένεται να συνεχιστεί τα επόμενα χρόνια. Επομένως, παρουσιάζει ιδιαίτερο ενδιαφέρον η εκτενέστερη μελέτη της συμπεριφοράς των σχεδίων καταμερισμού της κρυφής μνήμης που παρουσιάστηκαν σε αυτή την εργασίας καθώς και άλλων, συναρτήσει του πλήθους των πυρήνων. Επιπλέον, η μελέτη αυτή πρέπει να γίνει λαμβάνοντας υπόψη και το κόστος που συνοδεύει την ενσωμάτωση του εκάστοτε σχεδίου.

8.3.2 Πρόταση για τη Μείωση του Κόστους σε Υλικό

Στην αρχιτεκτονική υπολογιστών, η αξιολόγηση οποιουδήποτε αρχιτεκτονικού σχεδίου κρίνεται κυρίως βάσει δύο σημαντικών παραγόντων. Ο πρώτος παράγοντας είναι προφανώς η απόδοση. Ο δεύτερος παράγοντας ο οποίος είναι εξίσου σημαντικός με τον πρώτο, είναι το κόστος σε υλικό. Βασική προϋπόθεση για την υιοθέτηση μιας σχεδιαστικής λύσης είναι να έχει μικρό επιπλέον κόστος σε υλικό ή τουλάχιστον να παρουσιάζει καλή σχέση

απόδοσης ως προς το κόστος του επιπλέον υλικού. Στη συνέχεια περιγράφεται μία πρόταση που έχει σκοπό να θυσιάσει ένα μικρό ποσοστό ευελιξίας, προκειμένου να επιτύχει καλύτερη σχέση απόδοσης προς κόστος για το ABFCP ή το ABFCPH σχέδιο.

Η μελέτη του ABFCP σχεδίου οδηγεί στο συμπέρασμα ότι η μείωση του επιπλέον κόστους σε υλικό μπορεί να επιτευχθεί προχωρώντας στην ομαδοποίηση των γραμμών της κρυφής μνήμης όσον αφορά τα μετρούμενα μεγέθη του μηχανισμού. Δηλαδή, αντί της ύπαρξης ενός Bloom φίλτρου και δύο μετρητών για τις ευστοχίες και τις μακρινές αστοχίες ανά γραμμή της κρυφής μνήμης, δύναται τα παραπάνω να χρησιμοποιούνται για περισσότερες από μία γραμμές. Στη συνέχεια η ομαδοποίηση δύναται να συνδυαστεί με δειγματοληψία. Με άλλα λόγια από κάθε ομάδα που ορίζεται, μόνο μία θα συνεισφέρει στην καταγραφή της πληροφορίας των LRU ευστοχιών και των μακρινών αστοχιών.

Η παραπάνω τροποποίηση συνεπάγεται κάποιο άμεσο κέρδος ως προς το κόστος σε υλικό αλλά έχει επίπτωση στην ευελιξία. Καταρχήν, το επιπλέον κόστος σε υλικό του ABFCP μηχανισμού μειώνεται κατά ένα συντελεστή που πλησιάζει το μέγεθος των ομάδων στις οποίες χωρίζονται οι γραμμές της κρυφής μνήμης. Από την άλλη όμως θυσιάζεται ένα μέρος της ευελιξίας του ABFCP μηχανισμού, καθώς πλέον η κάθε γραμμή δεν καταμερίζεται ανάμεσα στις εφαρμογές ανεξάρτητα αλλά σύμφωνα με τις αποφάσεις που λαμβάνονται συνολικά για την ομάδα στην οποία ανήκει.

Η τροποποίηση συνεπάγεται επίσης μεγαλύτερη πιθανότητα σφάλματος καθώς δεν υπάρχει καμία εγγύηση ότι σε γειτονικές γραμμές της κρυφής μνήμης οι εκτελούμενες εφαρμογές θα παρουσιάζουν την ίδια συμπεριφορά όσον αφορά του απαιτούμενους πόρους μνήμης σε συνάρτηση με το χρόνο. Ωστόσο, για μικρά μεγέθη ομάδας μπορεί να υποτεθεί κάποιος βαθμός ομοιομορφίας στις απαιτήσεις των ανταγωνιζόμενων εφαρμογών για τους πόρους της κρυφής μνήμης. Επιπλέον αποφεύγεται η αύξηση των false positives, καθότι το πλήθος των στοιχείων που εισάγονται στο Bloom φίλτρο παραμένει ίδιο με την αρχική περίπτωση.

Στο σημείο αυτό είναι χρήσιμο να πραγματοποιηθεί μια συνοπτική παρουσίαση του κόστους σε υλικό της υλοποίησης του ABFCPH σχεδίου, καθώς και της σχετικής μείωσης που επιτυγχάνεται με την υιοθέτηση της προτεινόμενης τροποποίησης. Τα μεγέθη υπολογίζονται για συσχετιστική κρυφή μνήμη 4 Mbytes με 16 δρόμους, για την περίπτωση ενός επεξεργαστή με 4 πυρήνες.

Κόστος Υλικού σε Επεξεργαστή με 4 Πυρήνες (P=4), με χρήση ομαδοποίησης-δειγματοληψίας (k γραμμές/ομάδα)				
Μονάδα	**Κόστος σε Υλικό (bytes)**	**ABFCPH (k = 1)**	**k = 4 γραμμές/ομάδα**	**k = 8 γραμμές/ομάδα**
Bloom Φίλτρο	P*Sets* (BF bits/8)*1/k	4*4096*8 bytes= 128 Kbytes	4*4096*8/4 bytes= 32 Kbytes	4*4096*8/8 bytes= 16 Kbytes
Counters	P*2*N*Sets*1/k	4*2*3*4096 bytes= 96 Kbytes	4*2*3*4096/4 bytes= 24 Kbytes	4*2*3*4096/8 bytes= 12 Kbytes
Core-id Field	Sets*($\log_2$P/8)* associativity	4096*¼*16 bytes= 32 Kbytes	4096*¼*16 bytes= 32 Kbytes	4096*¼*16 bytes= 32 Kbytes
Σύνολο		256 Kbytes	88 Kbytes	60 Kbytes
L2 Hardware Overhead		6,25%	2,148%	1,465%

Πίνακας 8.1: Κόστος υλικού με χρήση ομαδοποίησης-δειγματοληψίας

Συμπερασματικά η κύρια διαφορά της τροποποίησης που περιγράφηκε από τη δυναμική δειγματοληψία συνόλων που χρησιμοποιεί το UCP είναι ότι συλλέγει σαφώς περισσότερα δείγματα, επειδή το κόστος των Bloom φίλτρων είναι μικρότερο σε σχέση με το κόστος των κυκλωμάτων που περιλαμβάνει το UCP. Εναλλακτικά μπορεί να ειπωθεί ότι για να μειώσει το κόστος, το UCP σχέδιο λαμβάνει λιγότερα δείγματα με αποτέλεσμα να περιορίζεται στην εφαρμογή ενός ενιαίου καταμερισμού. Συνεπώς η προτεινόμενη τροποποίηση παρέχει τη δυνατότητα να βρισκόμαστε σχεδιαστικά στο μέσο μεταξύ UCP και ABFCP σχεδίου.

Βιβλιογραφία

1. **Intel Corporation.** Intel® Xeon® Processor 7000 Sequence-Technical Documents. [Online] http://www.intel.com/Assets/PDF/prodbrief/7400-prodbrief.pdf.

2. —. Intel® Xeon® Processor 5000 Sequence-Technical Documents. [Online] http://www.intel.com/Assets/PDF/prodbrief/xeon-5500.pdf.

3. **AMD Corporation.** Family 10h AMD Opteron™ Processor Product Data Sheet. [Online]

4. **Sun Microsystems.** UltraSPARC T2™ Supplement to the UltraSPARC Architecture 2007. [Online]

5. **Smith A. J.** Cache Memories. *ACM Computing Surveys.* 1982, pp. 473-530.

6. **Burks, Arthur W., Goldstine, Herman H. and Von Neumann, John.** *Preliminary Discussion of the Logical Design of an Electroinic Computing Instrument.* Institute for Advanced Study, Princeton. New Jersey : s.n., 1946.

7. **Hennessy, J. L. and Patterson, D. A.** *Computer Architecture: A Quantitative Approach.* 4th Edition. s.l. : Morgan Kaufmann.

8. **Lira J., Molina C. and Gonzalez A.** *Analysis of Non_Uniform Cache Architecture Policies for Chip-MultiProcessors Using the Parsec Benchmark Suite.* In Proc. of the 2nd Workshop on Managed Multi-Core Systems (MMCS'09). Washington DC, USA : s.n., 2009.

9. **Qureshi M., Jaleel A., Patt Y., Steely S., Emer J.** *Adaptive Insertion Policies for High Performance Caching.* In Proc. of the 34th International Symposium on Computer Architecture. San Diego, USA : s.n., 2007.

10. **Qureshi M., Jaleel A., Patt Y., Steely S.** *Set-Dueling-Controlled Adaptive Insertion for High-Performance Caching.* In Proc. of the IEEE International Conference on Technologies for Homeland Security. USA : s.n., 2008.

11. **Kron J., Pruno B. and Loh, G.** *Double-DIP: Augmenting DIP with Adaptive Promotion Policies to Manage Shared L2 Caches.* In Proc. of the 35th International Symposium on Computer Architecture (ISCA-35). Beijing, China : s.n., 2008.

12. **Jaleel A., Hasenplaugh W., Qureshi M., Sebot J., Steely S., Emer J.** *Adaptive Insertion Policies for Managing Shared Caches.* In Proc. of the 17th International Conference on Parallel Architectures and Compilation Techniques (PACT). Toronto, Canada : s.n., 2008.

13. **Patt Y. N. and Qureshi K. M.** *Utility-Based Cache Paritioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches.* In Proc. of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. Orlando, Florida, USA : s.n., 2006. pp. 423-432.

14. **Nikas K., Horsnell M. and Garside J.** *An Adaptive Bloom Filter Cache Partitioning Scheme for Multicore Architectures.* In Proc. of the International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation. Samos : s.n., 2008.

15. **Magnusson P.S., Christensson M., Eskilson J., Forsgren D., Hallberg G., Hogberg J., Larsson F., Moestedt A., Werner B.** Simics: A Full System Simulation Platform. *IEEE Computer.* 2002, pp. 50-58.

16. **Sherwood T., Perelman E., Hamerly G., Calder B.** *Automatically Characterizing Large Scale Program Behaviour.* In Proc. of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS). 2002.

17. **Standard Performance Evaluation Corporation.** SPEC CPU2000. [Online] http://www.spec.org/cpu2000/.

18. SimPoint Overview. *SimPoint.* [Ηλεκτρονικό] http://cseweb.ucsd.edu/~calder/simpoint/simpoint_overview.htm.

19. **Patil H., Cohn R., Charney M., Sun A., Karunanidhi A.** *Pinpointing Representative Portions of Large Intel Itanium Programs with Dynamic Instrumentation.* In Proc of the 37th Annual IEEE/ACM International Symposium on Microarchitecture. 2004.

20. **Arun A. J. and Lizy K. J.** *Simulation Points for SPEC CPU 2006.* In Proc. of the 26th International Conference on Computer Design (ICCD). Lake Tahoe, CA, USA : s.n., 2008.

21. **Vera J., Cazorla F.J., Pajuelo A., Santana O., Fernandez E., Valero M.** *FAME: FAirly Measuring Multithreaded Architectures.* In Proc. of the 16th International Conference on Parallel Architectures and Compilation Techniques (PACT). Brasov, Romania : s.n., 2007.

22. **Wright, G.** *A single-chip multiprocessor architecture with hardware thread support.* Department of Computer Science, The University of Manchester. 2001. PhD thesis.

23. **Horsnell M. J.** *Cycle-accurate, distributed chip multiprocessor simulation.* ESPRC Postgraduate Research in Engineering and Physical Sciences (PREP). 2004.

24. **Reddi V.J., Settle A., Connors D.A., Cohn R.S.** *PIN: A Binary Instrumentation Tool for Computer Architecture Research and Education.* In Proc of the 31st Annual International Symposium on Computer Architecture. Munchen, Germany : s.n., 2004.

25. **Standard Performance Evaluation Corporation.** SPEC CPU2006. [Online] http://www.spec.org/cpu2006/.

26. **Calder B., Sherwood T., Hamerly G., Perelman E.** *SimPoint: Picking Representative Samples to Guide Simulation.* 2006.

27. **Mattson, L. R.** *Evaluation Techniques in Storage Hierarchies.* s.l. : IBM, 1970. Journal of Research and Development.

28. **Nikas, Konstantinos.** *An Analysis of Cache Partitioning Techniques for Chip Multiprocessor Systems.* Faculty of Engineering and Physical Sciences, School of Computer Science, University of Manchester. 2008.