Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών

και Μηχανικών Υπολογιστών

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Βελτίωση της Ποιότητας Υπηρεσιών (QoS) σε Πολυεπεξεργαστικά Συστήματα με την χρήση της τεχνολογίας Intel RDT και του Thread Packing

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΕΛΚΟΝ ΧΑΤΣΙΚΙΑΝ

Επιβλέπων : Γεώργιος Γκούμας

Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2020

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών

και Μηχανικών Υπολογιστών

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Βελτίωση της Ποιότητας Υπηρεσιών (QoS) σε Πολυεπεξεργαστικά Συστήματα με την χρήση της τεχνολογίας Intel RDT και του Thread Packing

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΜΕΛΚΟΝ ΧΑΤΣΙΚΙΑΝ

Επιβλέπων : Γεώργιος Γκούμας

Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 7η Οκτωβρίου 2020.

Γεώργιος Γκούμας	Νεκτάριος Κοζύρης	Διονύσιος Πνευματικάτος
Επίκουρος Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2020

Μελκόν Χατσικιάν

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

Περίληψη

Με την ραγδαία ανάπτυξη της τεχνολογίας τα τελευταία χρόνια, τα σύγχρονα πολύ-επεξεργαστικά συστήματα έχουν εδραιωθεί σε πολλά κέντρα δεδομένων, αφού τα συστήματα, αυτά, παρέχουν μεγαλύτερη υπολογιστική ισχύ και αύξηση της επίδοσης των εφαρμογών. Όμως, όταν περισσότερες από μία εφαρμογές συνεκτελούνται σε αυτά τα συστήματα, προκύπτει το πρόβλημα του ανταγωνισμού για κοινόχρηστους πόρους, όπως για παράδειγμα την Last-Level Cache ή το DRAM bandwidth.

Η πλειοψηφία των σύγχρονων κέντρων δεδομένων, προκειμένου να προσφέρουν ποιότητα υπηρεσίας στους πελάτες του και να τηρούν τα συμφωνηθέντα SLAs (Service Level Agreements), οδηγούνται στην υποχρησιμοποίηση των πόρων που έχουν στην διάθεση τους. Συνεπώς, επιλέγουν να εκτελούν μόνο μία εφαρμογή σε κάθε σύστημα, μη χρησιμοποιώντας το μεγαλύτερο μέρος του συστήματος. Γίνεται, λοιπόν, αντιληπτό πως είναι αναγκαία η εύρεση μίας λύσης ώστε να προστατευθεί η επίδοση μίας εφαρμογής σε ένα περιβάλλον συνεκτελούμενων εφαρμογών, ώστε να αυξηθεί η συνολική χρησιμοποίηση του συστήματος.

Ο ήδη υπάρχων μηχανισμός DICER αποτελεί μία εργασία, η οποία προσπαθεί να προστατεύσει την επίδοση μίας εφαρμογής υψηλής προτεραιότητας σε ένα περιβάλλον συνολικά δέκα συνεκτελούμενων εφαρμογών, αξιοποιώντας την τεχνολογία Intel RDT, η οποία προσφέρει την παρακολούθηση της χρήσης αλλά και τον διαμοιρασμό των κοινόχρηστων πόρων. Στην παρούσα διπλωματική εργασία, θα επεκτείνουμε αυτόν τον μηχανισμό, ώστε να αντιμετωπίζει αποτελεσματικότερα τον κορεσμό του DRAM bandwidth και συνεπώς να προστατευθεί περισσότερο η απόδοση της υψηλής προτεραιότητας εφαρμογής. Για την αξιολόγηση του μηχανισμού, χρησιμοποιούμε την σουίτα μετροπρογραμμάτων SPEC CPU2017.

Λέξεις Κλειδιά

κοινόχρηστοι πόροι επεξεργαστή, διαμοιρασμός κοινόχρηστης κρυφής μνήμης, διαμοιρασμός διαύλου δεδομένων, τεχνολογία Intel RDT, τεχνολογία CAT, τεχνολογία CMT, τεχνολογία MBM, τεχνολογία MBA, Thread Packing, δυναμικός μηχανισμός προστασίας επίδοσης

Abstract

In the last few years, with the rapid growth of technology, the modern multi-core systems have been established in many data centers since they provide greater processing power and increase in the performance of applications. However, when more than one application are co-executed in the aforementioned systems, the problem of competitiveness for shared resources occurs among those, as for example the Last-Level Cache or the DRAM bandwidth.

The majority of modern data centers, in order to offer Quality of Service (QoS) to their clients and to serve the agreed SLAs (Service Level Agreements), are led to the underutilization of their systems. Thus, they choose to execute only one application in each system and avoid taking the advantage of their power. The need to find a solution to protect the performance of one application in an environment of co-executed applications, while increasing the utilization of the system, is of primary importance.

The existing algorithm DICER is a previous research work, which tries to protect the performance of a high priority application in an environment of ten co-executed applications, by taking the advantage of the Intel RDT technology, which is used to monitor the use and to divide the shared resources to applications. In this Diploma thesis, we will expand this algorithm, in order to combat more efficiently the saturation of the DRAM bandwidth and therefore, to provide better protection for the performance of the high priority application. For the evaluation of our algorithm, we use the benchmark suite SPEC CPU2017.

Key words

processor shared resources, shared cache partitioning, DRAM bandwidth partitioning, Intel RDT technology, CAT technology, CMT technology, MBM technology, MBA technology, Thread Packing, dynamic mechanism for protection of performance

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε από τον Νοέμβριο 2018 έως τον Οκτώβριο 2020, στο Εργαστήριο Υπολογιστικών Συστημάτων της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου.

Αρχικά, θα ήθελα να ευχαριστήσω τον Επιβλέποντα Καθηγητή μου κ. Γεώργιο Γκούμα για την βοήθειά του στην εκπόνηση της παρούσας διπλωματικής εργασίας. Ακόμη, θα ήθελα ιδιαιτέρως να ευχαριστήσω τον Μεταδιδακτορικό Ερευνητή κ. Κωνσταντίνο Νίκα για τις πολύτιμες συμβουλές του και την καθοδήγησή του, καθώς ήταν ενεργά στο πλάι μου από την αρχή μέχρι το τέλος της διπλωματικής εργασίας. Οι γνώσεις και οι ιδέες του αποτέλεσαν έναν σημαντικό παράγοντα στην επιτυχή εκπλήρωση αυτής της εργασίας. Επίσης, θα ήθελα να ευχαριστήσω τους Μεταδιδακτορικούς Ερευνητές Νικέλα Παπαδοπούλου και Βασίλη Καρακώστα για τις πολύτιμες συμβουλές τους.

Κλείνοντας, πλέον, ένα σημαντικό κεφάλαιο της ζωής μου, θα ήθελα να ευχαριστήσω τους φίλους μου, παλαιούς και νέους, για την υποστήριξη που μου παρείχαν κατά την διάρκεια των φοιτητικών μου χρόνων και για τις αξέχαστες στιγμές που περάσαμε μαζί. Τέλος, το μεγαλύτερο ευχαριστώ ανήκει στους γονείς μου για την απεριόριστη εμπιστοσύνη και την αδιάκοπη στήριξη τους, ακόμη και στις πιο δύσκολες στιγμές.

Μελκόν Χατσικιάν,

7η Οκτωβρίου 2020

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Σύγχρονα Πολυεπεξεργαστικά Συστήματα

Η ραγδαία ανάπτυξη της τεχνολογίας τα τελευταία χρόνια έχει συντελέσει στην εδραίωση των πολυπύρηνων επεξεργαστών στα σύγχρονα επεξεργαστικά συστήματα, όπως για παράδειγμα στους εξυπηρετητές (servers) και στα κέντρα δεδομένων (Data Centers). Σε αντίθεση με τα παλαιότερα επεξεργαστικά συστήματα, τα οποία περιλάμβαναν μόνο μία επεξεργαστική μονάδα και προσέφεραν παραλληλισμό μόνο σε επίπεδο εντολών (Instruction Level Parallelism, ILP), τα σύγχρονα συστήματα δίνουν την δυνατότητα εκτέλεσης πολλαπλών εφαρμογών ταυτόχρονα μέσω των πολλαπλών επεξεργαστικών μονάδων που διαθέτουν. Αναμφίβολα, αυτό έχει συντελέσει στην αύξηση της επεξεργαστικής ισχύος και στην συνολική αύξηση της απόδοσης, όμως το πρόβλημα του ανταγωνισμού των κοινόχρηστων πόρων παρουσιάζεται σε αυτά τα συστήματα ([1], [2], [3]). Ο ανταγωνισμός των εκτελούμενων εφαρμογών για τους κοινόχρηστους πόρους μπορεί να επιφέρει μείωση της απόδοσης και αφορά την διεκδίκηση χώρου στην κοινόχρηστη κρυφή μνήμη (on-chip shared cache memory), το εύρος του διαύλου δεδομένων από και προς την μνήμη (DRAM bandwidth), ελεγκτές E/E (I/O controllers), κτλ..

Ως αποτέλεσμα, ο ανταγωνισμός, αυτός, αποτελεί εμπόδιο για την βέλτιστη απόδοση μίας ή περισσότερων εφαρμογών και για την παροχή ποιότητας υπηρεσίας (Quality of Service, QoS) στα κέντρα δεδομένων. Συνεπώς, το πρόβλημα ανταγωνισμού των κοινόχρηστων πόρων αποτελεί αντικείμενο έρευνας πολλών ερευνητών σήμερα ([4], [5], [6], [7], [8], [9], [10], [11], [12]). Οι έρευνες επικεντρώνονται στην αναζήτηση της βέλτιστης διαχείρισης και τον διαμοιρασμό των κοινόχρηστων πόρων σε περιβάλλον συνεκτελούμενων εφαρμογών, προκειμένου να επιτύχουν καλύτερη απόδοση και να μειώσουν ή εξαλείψουν τον ανταγωνισμό.

1.2 Υποχρησιμοποίηση των Πόρων ενός Συστήματος

Η παροχή συγκεκριμένης ποιότητας υπηρεσιών είναι μείζονος σημασίας για τα σύγχρονα κέντρα δεδομένων, καθώς πολλές εταιρείες απαιτούν οι εφαρμογές τους να εκτελούνται με την βέλτιστη απόδοση ή σε μικρή απόκλιση από αυτήν. Εξαιτίας του προβλήματος του ανταγωνισμού των κοινόχρηστων πόρων, όπως περιγράφθηκε παραπάνω, τα κέντρα δεδομένων υποχρεώνονται να απομονώνουν μία εφαρμογή, ώστε να μην μειώνεται η απόδοσή της. Συνεπώς, σε ένα πολυεπεξεργαστικό σύστημα επιλέγεται να τρέχει μόνο μία εφαρμογή υποχρησιμοποιώντας τους διαθέσιμους πόρους, ώστε να μην παραβιάζονται τα συμφωνηθέντα SLAs (Service Level Agreements) των πελατών. Για παράδειγμα, η Google έχει επιλέξει να χρησιμοποιεί μόνο το 30% των διαθέσιμων πόρων των servers της ([5]), για την αποτελεσματική και γρήγορη εξυπηρέτηση των αιτημάτων των χρηστών κατά την αναζήτηση στις βάσεις δεδομένων της.

Γίνεται, λοιπόν, αντιληπτό πως η βέλτιστη και ανεξάρτητη εκτέλεση μίας εφαρμογής σε ένα περιβάλλον συνεκτελούμενων εφαρμογών, αποτελεί την πρωταρχική απαίτηση των χρηστών, είτε πρόκειται για ατομική είτε για επαγγελματική χρήση.

1.3 Συνεισφορά της Εργασίας

Η παρούσα διπλωματική εργασία αποτελεί συνέχεια της εργασίας των Νίκα κ.α. ([11]) και συγκεκριμένα είναι η επέκταση του μηχανισμού, ώστε να αντιμετωπιστεί αποτελεσματικότερα ο κορεσμός του διαύλου δεδομένων προς και από την μνήμη (DRAM bandwidth saturation).

Ο μηχανισμός DICER (Diligent Cache Partitioning for Efficient Workload Consolidation) χρησιμοποιεί την τεχνολογία Intel Resource Director Technology (RDT) ([13]) και παίρνει δυναμικές αποφάσεις, με σκοπό την προστασία της επίδοσης μίας υψηλής προτεραιότητας εφαρμογής έναντι άλλων χαμηλότερης προτεραιότητας, σε ένα περιβάλλον συνεκτελούμενων εφαρμογών. Η τεχνολογία Intel RDT, απαρτίζεται από το Cache Monitoring Technology (CMT) και το Memory Bandwidth Monitoring (MBM) που επιτρέπουν την επίβλεψη της απόδοσης και των απαιτήσεων των εφαρμογών, καθώς και από το Cache Allocation Technology (CAT) που επιτρέπει την διαχείριση του μεγέθους της Last-Level Cache (LLC) που κατανέμεται σε κάθε εφαρμογή ή πυρήνα.

Στην παρούσα διπλωματική, θα χρησιμοποιηθεί η τεχνική του Thread Packing, προκειμένου να αντιμετωπιστεί ο κορεσμός του διαύλου δεδομένων και

την καλύτερη προστασία της επίδοσης της υψηλότερης προτεραιότητας εφαρμογής. Ο μηχανισμός δυναμικών αποφάσεων που υλοποιούμε βασίζεται στο API που παρέχεται από το εργαλείο PQoS. Η αξιολόγηση του μηχανισμού έγινε με την χρήση της νέα σουίτας μετροπρογραμμάτων (benchmarks) SPEC, τα SPEC CPU2017, αφού αρχικά γίνει ανάλυση των χαρακτηριστικών τους.

Κεφάλαιο 2

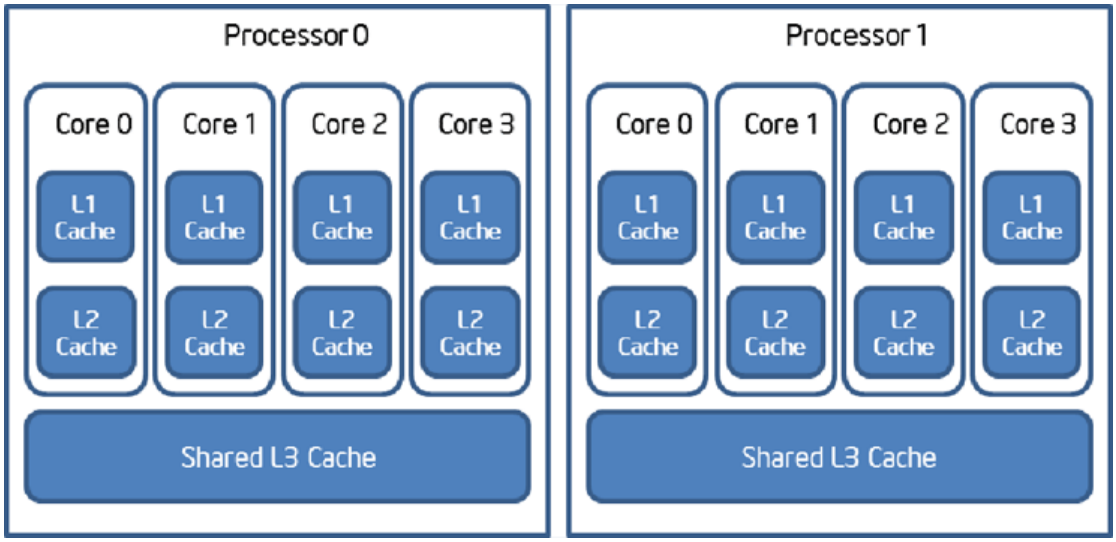
Θεωρητικό Υπόβαθρο

2.1 Εισαγωγή

Οι σύγχρονες εφαρμογές απαιτούν ολοένα και περισσότερη υπολογιστική ισχύ από τα συστήματα, στα οποία εκτελούνται. Συνεπώς, τα κέντρα δεδομένων προμηθεύονται επεξεργαστικά συστήματα με περισσότερους επεξεργαστές για να καλύψουν τις ανάγκες των πελατών τους. Τέτοια συστήματα δίνουν την δυνατότητα στον χρήστη να εκτελέσει πολλές εφαρμογές ταυτόχρονα, κάτι που μπορεί να οδηγήσει στην μείωση της απόδοσης λόγω του ανταγωνισμού για κοινόχρηστους πόρους. Όπως αναφέρθηκε παραπάνω, για να αντιμετωπιστεί αυτό το πρόβλημα, πολλά κέντρα δεδομένων επιλέγουν να υποχρησιμοποιούν τους servers που έχουν στην διάθεσή τους, ώστε να είναι σε θέση να προσφέρουν την πολυπόθητη ποιότητα υπηρεσίας που ζητά ο πελάτης. Αυτό οδηγεί σε αυξημένα κόστη συντήρησης και λειτουργίας, καθώς πολλές φορές ένα πολυεπεξεργαστικό σύστημα χρησιμοποιείται για την εκτέλεση μόνο μίας εφαρμογής. Γίνεται, λοιπόν, αντιληπτό πως είναι μείζονος σημασίας να βρεθεί λύση στο πρόβλημα της συνεκτέλεσης (co-execution) εφαρμογών, ώστε να μειωθούν τα κόστη και να χρησιμοποιούνται αποτελεσματικότερα οι διαθέσιμοι πόροι ενός κέντρου δεδομένων.

Όπως φαίνεται από το σχήμα 2.1, η ιεραρχία μνήμης των περισσότερων σύγχρονων επεξεργαστικών συστημάτων αποτελείται από τρία επίπεδα. Ο σχεδιασμός αυτός, συμβάλλει στην καλύτερη επίδοση των εφαρμογών και στην μείωση της καταναλωμένης ενέργειας των προσβάσεων στην κύρια μνήμη. Στο πρώτο και δεύτερο επίπεδο βρίσκονται οι L1 και L2 caches, οι οποίες είναι προσβάσιμες μόνο από τον φυσικό επεξεργαστή στον οποίο ανήκουν και συνεπώς χαρακτηρίζονται ως 'ιδιωτικές μνήμες'. Λόγω του ότι οι μνήμες αυτές είναι κατά κανόνα μικρές σε μέγεθος, δεν μπορούν να εξυπηρετήσουν όλα τα αιτήματα μίας εφαρμογής για πρόσβαση στην μνήμη, παρόλο που προσφέρουν απομόνωση των δεδομένων μίας εφαρμογής. Αυτό το πρόβλημα αντιμετωπίζεται με την προσθήκη του τρίτου επιπέδου στην ιεραρχία της

μνήμης, την L3 ή Last-Level cache, η οποία είναι πολλαπλάσια σε μέγεθος συγκριτικά με τις L1 και L2 caches και είναι κοινόχρηστη μεταξύ των διαφορετικών φυσικών επεξεργαστικών μονάδων. Η χρήση της LLC βοηθά να παραμένουν περισσότερα δεδομένα 'κοντά' στην επεξεργαστική μονάδα, μειώνοντας τον χρόνο και το ενεργειακό κόστος της πρόσβασης στην κύρια μνήμη (Dynamic Random-Access Memory, DRAM). Όμως, το γεγονός ότι αυτή η μνήμη είναι κοινόχρηστη, ενδέχεται να δημιουργήσει πρόβλημα ανταγωνισμού πόρων σε ένα περιβάλλον συνεκτέλεσης εφαρμογών. Για παράδειγμα, έστω ότι εκτελούνται δύο εφαρμογές P1 και P2 μέσα σε ένα σύστημα. Ένα αίτημα για δεδομένα της P1 τα οποία δεν βρίσκονται στην LLC, μπορεί να 'διώξει' τα δεδομένα της P2, προκείμενου να δημιουργηθεί επαρκής χώρος για την εξυπηρέτηση του αιτήματος της P1. Συνεπώς, αυξάνεται το miss-rate της P2 και μειώνεται η συνολική της επίδοση.



Σχήμα 2.1 : Ιεραρχία μνήμης των σύγχρονων πολυεπεξεργαστικών συστημάτων

Τα τελευταία χρόνια, πολλοί ερευνητές έχουν επικεντρωθεί στο να προσπαθήσουν να βρουν μία λύση ώστε να αντιμετωπιστεί ο ανταγωνισμός για τους κοινόχρηστους πόρους, μέσω του κατάλληλου διαμοιρασμού της κοινόχρηστης μνήμης (Cache Partitioning). Σε ένα περιβάλλον συνεκτέλεσης εφαρμογών, το cache partitioning μπορεί να προσφέρει απομόνωση, αύξηση της επίδοσης και δικαιοσύνη μεταξύ των εφαρμογών για κοινόχρηστους πόρους. Οι τεχνικές που έχουν προταθεί μπορούν να χωριστούν σε δύο κατηγορίες:

- Software-Based Τεχνικές : Αντιστοιχούν τις εφαρμογές σε συγκεκριμένα τμήματα της cache. Κύριο μειονέκτημα αυτής της τεχνικής αποτελεί η εισαγωγή πολυπλοκότητας στο λειτουργικό σύστημα (OS) ή τον επόπτη

(supervisor), καθώς και ότι απαιτείται η ανάπτυξή τους στο επίπεδο του kernel.

- Hardware-Based Τεχνικές : Αντιστοιχούν τους φυσικούς πυρήνες με τμήματα ή ways της cache. Κύριο μειονέκτημα αυτής της τεχνικής είναι ότι απαιτείται υποστήριξη από το hardware και έχει αναπτυχθεί τα τελευταία χρόνια. Συνεπώς, η συμβατότητα των επεξεργαστικών συστημάτων είναι αρκετά περιορισμένη.

2.2 Διαμοιρασμός LLC μέσω Software-Based Τεχνικών

2.2.1 Χρωματισμός Σελίδων (Page Coloring)

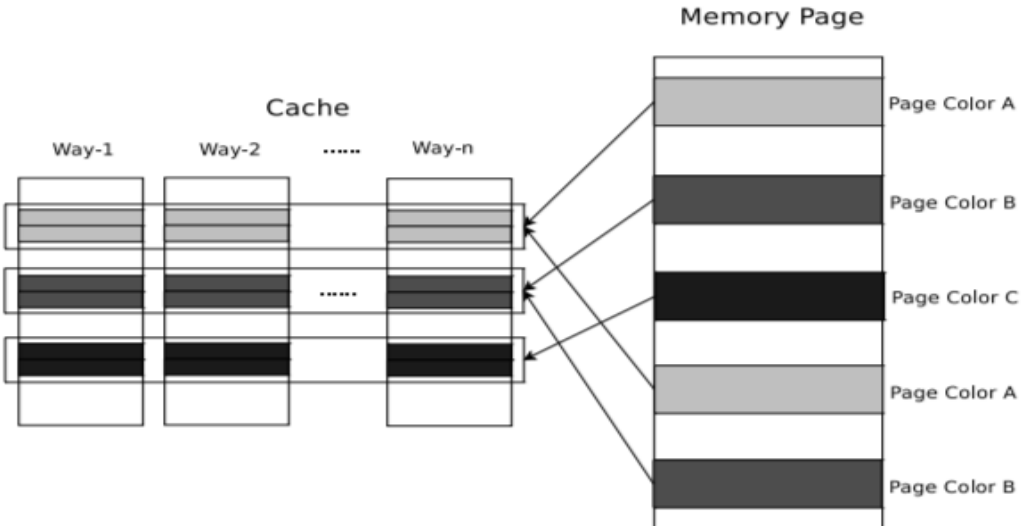
Ο χρωματισμός σελίδων ή αλλιώς page/cache coloring, έχει χρησιμοποιηθεί στο παρελθόν για την μετάφραση των εικονικών διευθύνσεων στους επεξεργαστές αρχιτεκτονικής MIPS ([14]), και έπειτα συμπεριλήφθηκε σε λειτουργικά συστήματα ([15]), όπως Solaris και FreeBSD, καθώς και σε μεταγλωττιστές ([16]).

Κατά την διάρκεια αυτής της τεχνικής, εικονικές σελίδες (virtual pages) της εικονικής μνήμης δεσμεύονται από το λειτουργικό σύστημα, ώστε να αναπαριστούν συνεχόμενα cache lines/sets της μνήμης. Ο στόχος του page coloring είναι να πετύχει απομόνωση των δεδομένων που χρησιμοποιεί η επεξεργαστική μονάδα στην κοινόχρηστη μνήμη cache, μειώνοντας τα conflict misses που συμβαίνουν μεταξύ διαφορετικών επεξεργαστών, οι οποίοι προσπαθούν να τροποποιήσουν τα αποθηκευμένα δεδομένα σε ένα συγκεκριμένο cache block ή cache line. Ως αποτέλεσμα, το page coloring πετυχαίνει την αύξηση της επίδοσης των εφαρμογών σε ένα περιβάλλον συνεκτέλεσης, αφού μειώνει τις προσβάσεις στην μνήμη DRAM και απομονώνει τμήματα της μνήμης LLC ανάμεσα στις διαφορετικές επεξεργαστικές μονάδες.

Η υλοποίηση της τεχνικής του page coloring γίνεται από κώδικα χαμηλού επιπέδου, ο οποίος αναλαμβάνει να δεσμεύσει δυναμικά τον κατάλληλο χώρο στην LLC, καθώς και την μετάφραση των εικονικών διευθύνσεων μνήμης σε φυσικές διευθύνσεις μνήμης. Στην συνέχεια, τα πλαίσια φυσικής μνήμης (physical memory page frames) 'χρωματίζονται' από το λειτουργικό σύστημα, ώστε να αντιστοιχίζονται σε διαφορετικά τμήματα της LLC. Τέλος, το MMU (Memory Management Unit) επιλέγει για το εάν θα αντιστοιχίσει τις εικονικές σελίδες σε διαφορετικά χρωματισμένα πλαίσια φυσικής μνήμης, ώστε οι εφαρμογές να μην ανταγωνίζονται για το ίδιο τμήμα της κοινόχρηστης μνήμης

LLC, ή για το εάν θα τις αντιστοιχίσει στον ίδιο χρωματισμό, ώστε τα δεδομένα μίας εφαρμογής να καταλαμβάνουν έναν συγκεκριμένο χώρο της μνήμης.

Η τεχνική του page coloring προσφέρει ντετερμινιστική εκτέλεση των εφαρμογών και βελτίωση της απόδοσης τους σε ένα περιβάλλον συνεκτελέσεων, μέσω του διαμοιρασμού της κοινόχρηστης μνήμης. Τα βασικό της μειονέκτημα είναι ότι απαιτεί ανάπτυξη κώδικα σε επίπεδο kernel, κάτι που εισάγει επιπλέον πολυπλοκότητα στο σύστημα.



Σχήμα 2.2 : Αντιστοίχιση εικονικών σελίδων στην φυσική μνήμη κατά το page coloring

2.2.2 Χρονοδρομολογητής Λειτουργικού Συστήματος (OS Scheduler)

Όπως περιγράφθηκε παραπάνω, σε ένα περιβάλλον συνεκτελούμενων προγραμμάτων δημιουργείται ανταγωνισμός για τους κοινόχρηστους πόρους. Παρόλο που το λειτουργικό σύστημα είναι αυτό που δεσμεύει τα pages frames, δεν έχει την δυνατότητα να διαχειριστεί τμήματα ή και ολόκληρη την μνήμη LLC.

Οι Mars κ.α. ([7]) παρουσίασαν την μέθοδο Bubble-Up, η οποία κάνει πρόβλεψη με ποσοστό λάθους 1% για την μείωση της απόδοσης μίας εφαρμογής όταν συνεκτελείται με άλλες. Χρησιμοποιώντας, λοιπόν, το Bubble-Up ο χρήστης μπορεί να βρει τους συνδυασμούς εκείνων των εφαρμογών, οι οποίες μπορούν να τρέξουν με μικρότερο ανταγωνισμό για κοινόχρηστους πόρους. Συγκεκριμένα, η εν λόγω μεθοδολογία αναλύει την ευαισθησία μίας εφαρμογής ως προς την LLC, δηλαδή πόσο επηρεάζεται η απόδοσή της όταν συνεκτελείται με άλλες εφαρμογές που χρησιμοποιούν την κοινόχρηστη μνήμη, καθώς και την πίεση που η ίδια εφαρμογή ασκεί στην μνήμη, επηρεάζοντας τις άλλες

εφαρμογές. Ακόμη, υποστηρίζουν πως σε 17 workloads της Google, το Bubble-Up βρίσκοντας τους κατάλληλους συνδυασμούς, αύξησε την χρησιμοποίηση (utilization) των πόρων από 50% μέχρι και 90%. Το μειονέκτημα αυτής της τεχνικής είναι ότι εξετάζει μόνο τις ανάγκες των εφαρμογών ως προς την μνήμη και αγνοεί άλλους κοινόχρηστους πόρους, όπως το DRAM bandwidth.

Οι Fedorova κ.α. ([8]) παρουσίασαν μία επέκταση του χρονο-δρομολογητή του Solaris 10, κάνοντας τον πιο δίκαιο ως προς τον διαμοιρασμό της cache στις εφαρμογές. Ο αλγόριθμος που αναπτύχθηκε στοχεύει στην βέλτιστη απόδοση μίας εφαρμογής, ανεξαρτήτου τους μεγέθους της μνήμης που της έχει δοθεί. Πιο συγκεκριμένα, ο αλγόριθμος ρυθμίζει το κβάντο του χρόνου μέσα στο οποίο, τα νήματα (threads) της εφαρμογής θα είναι ενεργά και θα εκτελούνται μέσα στην επεξεργαστική μονάδα. Όταν το κβάντο τελειώσει, τότε χρονοδρομολογούνται νέα νήματα. Με αυτόν τον τρόπο επιτυγχάνεται η πολυπόθητη απομόνωση των νημάτων διαφορετικών εφαρμογών και αντιμετωπίζεται ο ανταγωνισμός για κοινόχρηστους πόρους, με το κόστος του ότι μία εφαρμογή δεν εκτελείται συνεχόμενα στην διάρκεια του χρόνου. Ακόμη, προκειμένου ο αλγόριθμος να είναι δίκαιος, παρακολουθεί την επίδοση των εφαρμογών μέσω του IPC (Instructions Per Cycle). Όταν το IPC μιας εφαρμογής πέσει κάτω από ένα όριο, τότε ο αλγόριθμος αυξάνει το κβάντο χρόνου της εφαρμογής. Αντίθετα, εάν το IPC είναι μεγαλύτερο από το άνω όριο, τότε μειώνει το κβάντο εκτέλεσης της εφαρμογής και το παραχωρεί σε άλλες.

Τέλος, μία ακόμη τεχνική χρονοδρομολόγησης εφαρμογών είναι το CQoS ([2]). Το CQoS αναλύει και κατηγοριοποιεί τις εφαρμογές ως προς τις απαιτήσεις για την μνήμη cache και έπειτα επιβάλλει την κατάλληλη προτεραιότητα στις εφαρμογές για την πρόσβασή τους στην μνήμη. Συνεπώς, μέσω διάφορων τεχνικών, το CQoS αντιμετωπίζει το πρόβλημα του ανταγωνισμού των εφαρμογών για την κοινόχρηστη μνήμη, όπως ο στατικός ή δυναμικός διαμοιρασμός της.

2.3 Διαμοιρασμός Κοινόχρηστων Πόρων μέσω Hardware-Based Τεχνικών

2.3.1 Τεχνολογία Intel RDT

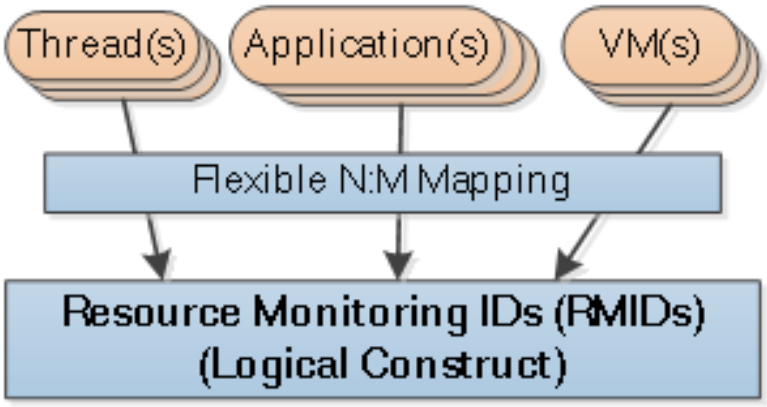
Τα τελευταία χρόνια, η Intel έχει εισάγει την τεχνολογία Intel RDT στους παραγόμενους servers της. Μέσω αυτής της τεχνολογίας γίνεται δυνατό στον χρήστη να παρατηρεί την επίδοση των εφαρμογών σε πραγματικό χρόνο και να κατανείμει τις διαθέσιμους πόρους, όπως την LLC ή το DRAM bandwidth, στους πυρήνες ή τις εφαρμογές. Η τεχνολογία αυτή παρουσιάστηκε πρώτα στους επεξεργαστές της οικογένειας Intel Xeon E5 v3 το 2014.

Η τεχνολογία Intel RDT απαρτίζεται από τα CMT, MBM, CAT και Memory Bandwidth Allocation (MBA). Το CMT επιτρέπει στον χρήστη, επόπτη ή λειτουργικό σύστημα να παρατηρεί την χρήση της LLC για κάθε πυρήνα ή εφαρμογή ξεχωριστά. Ακόμη, το MBM επιτρέπει την επίβλεψη της χρήσης του DRAM bandwidth για κάθε πυρήνα ή εφαρμογή. Επίσης, το MBM δίνει την δυνατότητα της παρακολούθησης του bandwidth μεταξύ διαφορετικών υποδοχέων επεξεργαστών (sockets), σε περίπτωση που υπάρχουν παραπάνω από ένας. Η τεχνολογία CAT δίνει την δυνατότητα στον χρήστη να καθορίσει στατικά ή δυναμικά το τμήμα της LLC που κάθε πυρήνας ή εφαρμογή μπορεί να χρησιμοποιήσει. Τέλος, στους νεότερους επεξεργαστές είναι διαθέσιμη η τεχνολογία MBA, μέσω της οποίας, ο διαχειριστής μπορεί να επιβάλει καθυστέρηση στα αιτήματα προς την μνήμη ενός πυρήνα ή μίας εφαρμογής, προκειμένου να μειώσει το DRAM bandwidth που καταναλώνεται από αυτούς.

Οι τεχνολογίες CMT, CAT και MBM υποστηρίζονται με την προσθήκη επιπλέον καταχωρητών και με την χρήση abstraction layers, τα οποία αντιστοιχίζουν λογικούς πυρήνες με τα εκτελούμενα προγράμματα. Πιο συγκεκριμένα, για την υποστήριξη του CMT και του MBM, έχει προστεθεί ένα abstraction layer μεταξύ των εκτελούμενων εφαρμογών και των λογικών πυρήνων μέσω της χρήσης RMIDs (Resource Monitoring IDs), όπως φαίνεται στο σχήμα 2.3. Σε κάθε εφαρμογή ή ομάδα αυτών, ή σε κάθε πυρήνα ή ομάδα αυτών, συσχετίζεται ένα RMID κατά την αρχικοποίηση από ένα RMID pool. Κάθε λογικός πυρήνας αντιστοιχίζεται μόνο με ένα RMID κάθε φορά. Τέλος, αξίζει να σημειωθεί πως υπάρχει περιορισμένος αριθμός RMIDs σε κάθε επεξεργαστή.

Για την αναγνώριση του RMID που έχει αντιστοιχηθεί σε κάποιον πυρήνα, χρησιμοποιείται ο καταχωρητής με αναγνωριστικό IA32_PQR_ASSOC MSR, ο οποίος περιέχεται σε κάθε πυρήνα. Ο

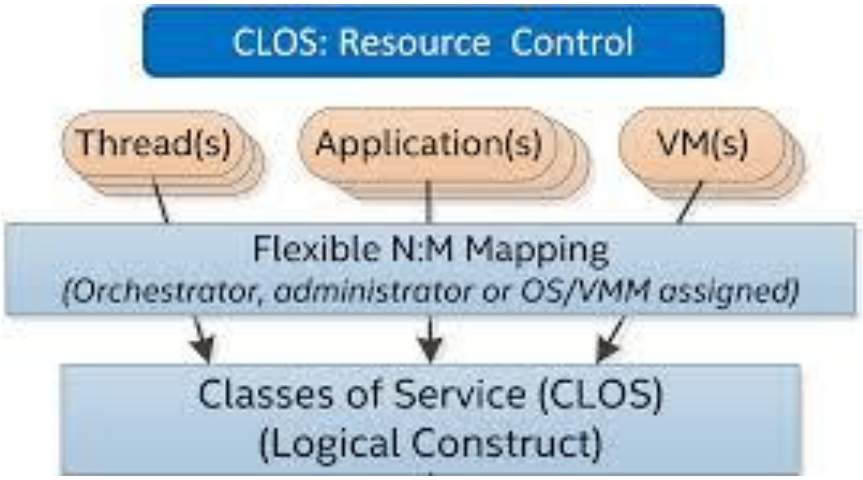
καταχωρητής, αυτός, ενημερώνεται κάθε φορά που αντιστοιχίζεται ένα νέο RMID στον πυρήνα που ανήκει.



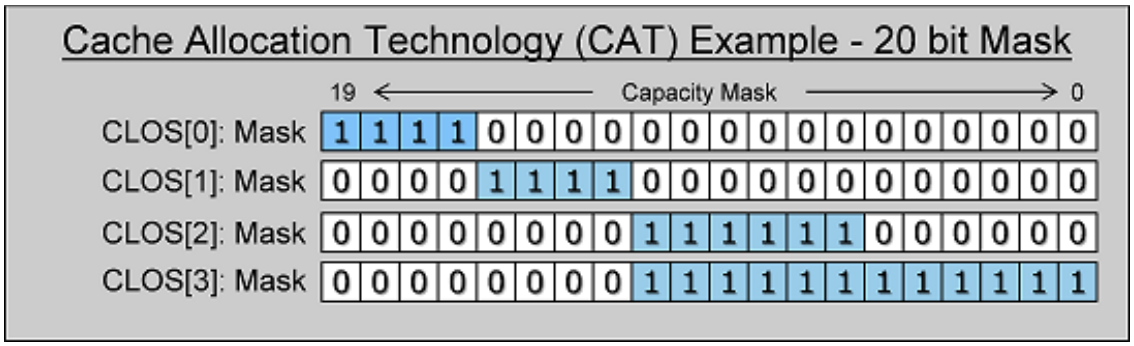
Σχήμα 2.3 : Αντιστοίχιση νημάτων ή εφαρμογών σε RMIDs

Για την λήψη των δεδομένων των μετρήσεων χρησιμοποιούνται δύο επιπλέον καταχωρητές. Ο πρώτος καταχωρητής, IA32_QM_EVTSEL MSR, περιέχει την πληροφορία για τα δεδομένα που αναμένεται να διαβαστούν. Για παράδειγμα, ο χρήστης μπορεί να επιλέξει για το εάν θα λάβει μετρήσει για τα LLC misses, ή το LLC occupancy, ή το DRAM bandwidth ή και όλες τις διαθέσιμες μετρήσεις. Οι τιμές των μετρήσεων περιέχονται στον δεύτερο καταχωρητή, IA32_QM_CTR MSR.

Για τον διαμοιρασμό (allocation) της LLC, χρησιμοποιείται η τεχνολογία CAT, η οποία υποστηρίζεται από το CLOS ή COS (Class of Service), που είναι ένα abstraction layer μεταξύ εφαρμογών και λογικών πυρήνων, όπως φαίνεται στο σχήμα 2.4. Σε κάθε εφαρμογή ή ομάδα αυτών, ή σε κάθε πυρήνα ή ομάδα αυτών, συσχετίζεται ένα CLOS. Κατά την αρχικοποίηση, όλες οι εφαρμογές ή όλοι πυρήνες αντιστοιχίζονται στο CLOS 0, όμως δίνεται η δυνατότητα στον χρήστη να αλλάξει αυτή την αντιστοίχιση στατικά. Η πληροφορία για το ποιο CLOS έχει αντιστοιχηθεί περιέχεται στον καταχωρητή IA32_PQR_ASSOC MSR, ο οποίος όπως προαναφέρθηκε χρησιμοποιείται και για την αποθήκευση του RMID. Πρέπει να τονιστεί πως το RMID και το CLOS είναι ανεξάρτητα μεταξύ τους, και συνεπώς, τα CMT και CAT δεν αλληλοεπιδρούν μεταξύ τους. Τέλος, η τεχνολογία CAT προσφέρει την δυνατότητα στον χρήστη να διαμοιράσει την LLC στατικά ή δυναμικά, επιλέγοντας συγκεκριμένα ways της cache για κάθε εφαρμογή ή πυρήνα, μέσω μίας μάσκας από bits τα οποία αντιστοιχούν σε κάθε way, όπως φαίνεται στο σχήμα 2.5.

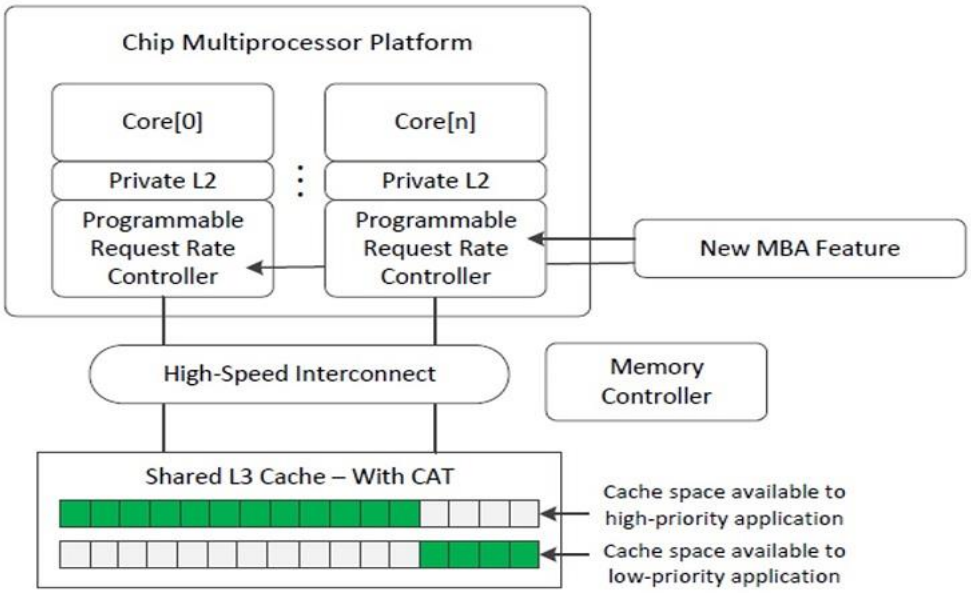


Σχήμα 2.4 : Αντιστοίχιση νημάτων ή εφαρμογών σε CLOS



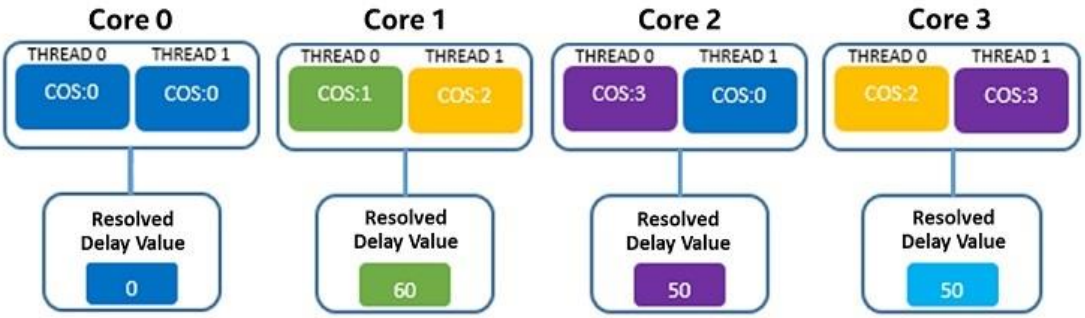
Σχήμα 2.5 : Μάσκα επιλεγμένων ways μίας 20-ways LLC ανά CLOS

Στους νεότερους επεξεργαστές της Intel υποστηρίζεται η τεχνολογία MBA. Το MBA δίνει την δυνατότητα στον χρήστη να ρυθμίσει τον ρυθμό αιτημάτων μίας εφαρμογής ή ενός πυρήνα προς την μνήμη, είτε στατικά είτε δυναμικά. Ουσιαστικά, μέσω της τεχνολογίας MBA, επιβάλλεται χρονική καθυστέρηση στα αιτήματα προς την μνήμη LLC σε κάθε κβάντο του χρόνου και έτσι επιτυγχάνεται η μείωση του DRAM bandwidth που χρησιμοποιούν οι χαμηλής προτεραιότητας εφαρμογές, προστατεύοντας έτσι την επίδοση της υψηλής προτεραιότητας εφαρμογή. Η τεχνολογία, αυτή, υποστηρίζεται από έναν προγραμματιζόμενο ελεγκτή ρυθμού αιτημάτων (Programmable Request Rate Controller) μεταξύ κάθε πυρήνα και LLC (σχήμα 2.6).



Σχήμα 2.6 : Προγραμματιζόμενος ελεγκτής ρυθμού αιτήσεων της τεχνολογίας MBA

Πρέπει, επίσης, να τονιστεί ότι σε συστήματα τα οποία υποστηρίζουν την τεχνολογία hyperthreading και όταν χρησιμοποιούνται και ο λογικός και ο φυσικός πυρήνας με διαφορετικά CLOS, τότε επικρατεί η μεγαλύτερη χρονική καθυστέρηση, όπως φαίνεται στο σχήμα 2.7. Αυτό συμβαίνει διότι το MBA ορίζεται μόνο για τους φυσικούς πυρήνες και όχι για τους λογικούς πυρήνες.



COS	DELAY
0	0
1	60
2	50
3	50

Table at left: Example COS-to-delay value mapping

Resolved Delay Value = Max(Delay(Thread0[CLOS], Thread1[CLOS]))

Σχήμα 2.7 : Επικράτηση μεγαλύτερης τιμής MBA ανάμεσα σε φυσικούς και λογικούς πυρήνες

Τέλος, αξίζει να σημειωθεί πως λόγω της συχνότητας εμφάνισης του προβλήματος ανταγωνισμού κοινόχρηστων πόρων και της ιδιαίτερης σημασίας του, κι άλλες εταιρείες παραγωγής επεξεργαστών έχουν οδηγηθεί στην εισαγωγή τεχνολογίας παρόμοια του Intel RDT στα προϊόντα τους, όπως για παράδειγμα η AMD και η ARM.

2.4 Διαμοιρασμός Κοινόχρηστων Πόρων συνδυάζοντας Software-Based και Hardware-Based Τεχνικές

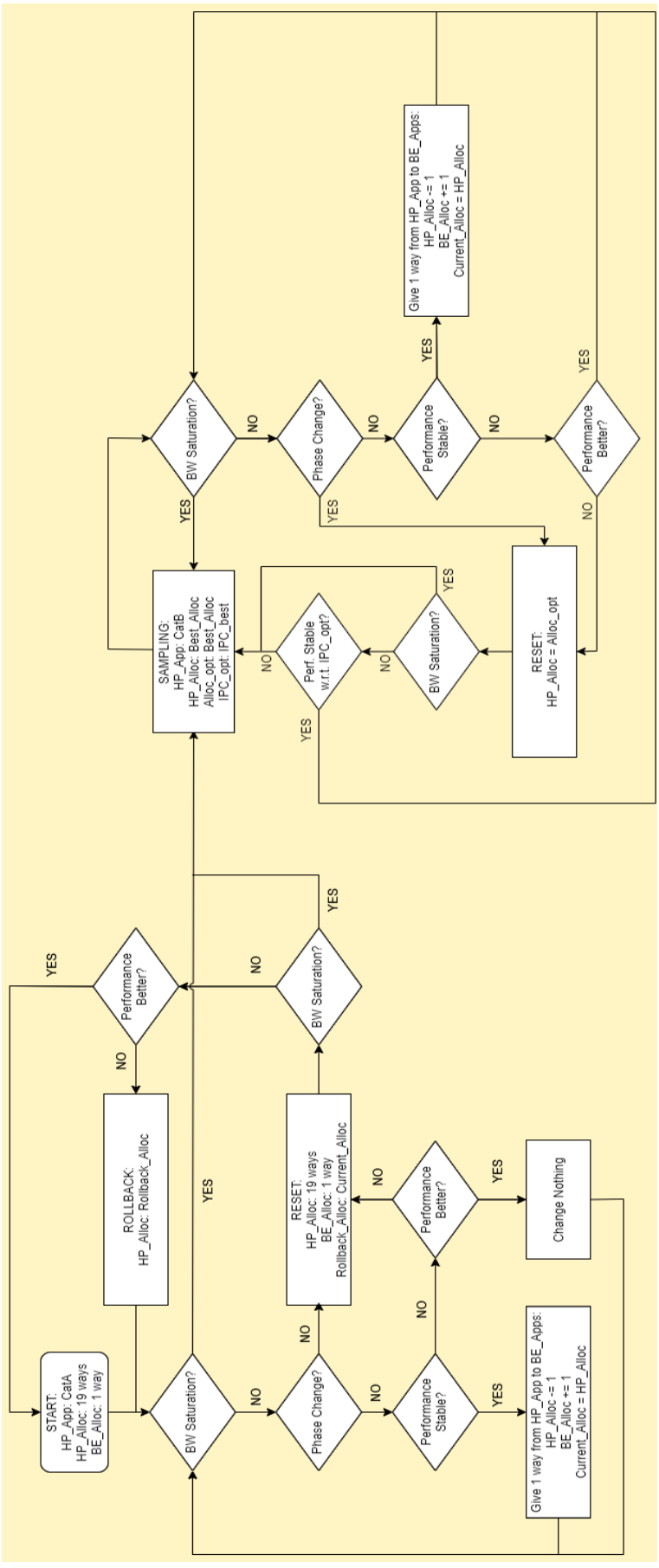
Η ανάπτυξη της προαναφερθέντας τεχνολογίας Intel RDT σε επίπεδο hardware, έχει οδηγήσει πολλούς ερευνητές να την αξιοποιήσουν για να βρουν την βέλτιστη λύση στο πρόβλημα του ανταγωνισμού κοινόχρηστων πόρων. Οι λύσεις που έχουν προταθεί, συνδυάζουν την ύπαρξη ενός επόπτη ή daemon στο επίπεδο λειτουργικού συστήματος, ο οποίος επιτηρεί, αποφασίζει και διαμοιράζει τους κοινόχρηστους πόρους δυναμικά μέσω του Intel RDT.

Μεγάλο ενδιαφέρον παρουσιάζει η εργασία των Lo κ.α. ([5]), οι οποίοι παρουσίασαν τον Heracles, έναν ελεγκτή ο οποίος συντελεί στην αύξηση της χρησιμοποίησης των διαθέσιμων πόρων, ενώ παράλληλα προστατεύει την επίδοση των υψηλής προτεραιότητας εφαρμογών (Latency Critical, LC). Ο Heracles αυξάνει την χρησιμοποίηση των επεξεργαστών συνεκτελώντας LC εφαρμογές με άλλες χαμηλής προτεραιότητας εφαρμογές (Best Effort, BE). Ο μηχανισμός του Heracles επιτηρεί τον κορεσμό των κοινόχρηστων πόρων, όπως LLC ή DRAM bandwidth, καθώς και την απόδοση των LC εφαρμογών και την παραβίαση των SLAs. Σε περίπτωση που αντιληφθεί κάποιο από τα προαναφερθέντα, τότε περιορίζει ή απενεργοποιεί τις BE εφαρμογές, προστατεύοντας έτσι την απόδοση της LC εφαρμογής. Όταν η κατάσταση είναι ομαλή και δεν παραβιάζονται τα SLAs, τότε ο αλγόριθμος προσπαθεί να παραχωρήσει μη απαραίτητους πόρους για την LC εφαρμογή στις BE εφαρμογές. Για την αξιολόγηση του μηχανισμού χρησιμοποιήθηκαν LC workloads που χρησιμοποιούνται στα κέντρα δεδομένων της Google. Τα αποτελέσματα της αξιολόγησης έδειξαν ότι ο Heracles αυξάνει την χρησιμοποίηση των διαθέσιμων πόρων από 20% εώς και 90%, χωρίς να παραβιάζονται τα SLAs.

Μία ακόμη ερευνητική εργασία των Παπαδάκη κ.α. ([4]) εκμεταλλεύτηκε τις τεχνολογίες CMT-CAT, για την ανάπτυξη ενός ελεγκτή με όνομα DCP-QoS. Στην εργασία, αυτή, το σενάριο συνεκτέλεσης περιλαμβάνει την συνεκτέλεσης N εφαρμογών σε N πυρήνες, με 1-1 αντιστοίχιση. Από τις N εφαρμογές, μία χαρακτηρίζεται ως υψηλής προτεραιότητας (High Priority, HP) και οι *N-1*

εφαρμογές ως χαμηλής προτεραιότητας (BE). Στόχος του DCP-QoS είναι να προστατεύσει την επίδοση της HP σε ένα περιβάλλον συνεκτελούμενων εφαρμογών, αυξάνοντας παράλληλα την χρησιμοποίηση των πόρων του συστήματος. Ακόμη, για να γίνει η αξιολόγηση του μηχανισμού λαμβάνονται μετρήσεις για την No-QoS πολιτική, κατά την οποία όλες οι εφαρμογές τρέχουν ελεύθερα χωρίς κανένα περιορισμό ως προς την LLC, και την CT-QoS (Cache Takeover) πολιτική, κατά την οποία δίνονται 19 ways της LLC στην HP εφαρμογή και το εναπομένων 1 ways στις *N-1* BE εφαρμογές. Σύμφωνα με τα αποτελέσματα, το DCP-QoS πετυχαίνει να εκτελούνται οι HP εφαρμογές στο 80% χρόνου απομονωμένης εκτέλεσης (alone execution time), ενώ αυξάνει εώς και 5 φορές την επίδοση των BE εφαρμογών συγκριτικά με την CT-QoS πολιτική.

Τέλος, ο μηχανισμός DICER ([11]), ο οποίος θα χρησιμοποιηθεί και επεκταθεί στην παρούσα διπλωματική εργασία, προσπαθεί να προστατεύσει την επίδοση της HP εφαρμογής και να αυξήσει την χρησιμοποίηση των διαθέσιμων πόρων του συστήματος, δηλαδή μοιράζεται τον ίδιο στόχο με την εργασία ([4]). Επίσης, το σενάριο συνεκτέλεσης παρέμεινε το ίδιο, δηλαδή στο σύστημα εκτελούνται *N* εφαρμογές σε *N* πυρήνες, σε 1-1 αντιστοίχιση. Με παρόμοιο τρόπο με το DCP-QoS, λήφθηκαν μετρήσεις για τις No-QoS και CT-QoS πολιτικές, προκειμένου να αξιολογηθεί ο μηχανισμός DICER συγκριτικά με αυτές. Τα αποτελέσματα της αξιολόγησης του DICER έδειξαν πως ο μέσος όρος της επίδοσης των HP εφαρμογών αποκλίνουν κατά 2% από την CT-QoS πολιτική, ενώ παράλληλα αυξάνει την επίδοση των BE εφαρμογών από 30% σε 43%, για συνεκτελέσεις κατηγορίας CT-F (βλ. παρακάτω). Για συνεκτελέσεις κατηγορίας CT-T (βλ. παρακάτω), ο DICER αποκλίνει κατά 2% από την No-QoS πολιτική σε ότι αφορά τον μέσο όρο της επίδοσης των HP εφαρμογών, ενώ ο μέσος όρος της επίδοσης των BE εφαρμογών αποκλίνει κατά 9% από την αντίστοιχη επίδοση της No-QoS πολιτικής. Στην συνέχεια, παρουσιάζεται το διάγραμμα ροής των καταστάσεων του μηχανισμού DICER (σχήμα 2.8).



Σχήμα 2.8 : Διάγραμμα ροής αποφάσεων μηχανισμού DICER

2.5 Τεχνική Thread Packing – Αντιμετώπιση Κορεσμού DRAM Bandwidth

Το Thread Packing ([17]) αποτελεί μία software-based τεχνική, με την χρήση της οποίας, ο χρήστης μπορεί να μειώσει τον φόρτο εργασίας του συστήματος και την κατανάλωση ενέργειας. Επίσης, μπορεί να χρησιμοποιηθεί για την μείωση του καταναλισκόμενου DRAM bandwidth των νημάτων. Συγκεκριμένα, κατά το Thread Packing, ο χρήστης ορίζει N_T νήματα μίας πολυνηματικής εφαρμογής (multi-threaded application) να εκτελούνται σε N_C πυρήνες, με $N_C < N_T$.

Οι Park κ.α. ([12]) χρησιμοποίησαν την τεχνική του Thread Packing σε συνδυασμό με την software-based τεχνική του Clock Modulation και την hardware-based τεχνολογία MBA, που περιγράφθηκε παραπάνω, με σκοπό να δημιουργήσουν τον μηχανισμό HyPart, ο οποίος περιέχει τους κατάλληλους συνδυασμούς που επιτρέπουν στον χρήστη να μειώσει το DRAM bandwidth που καταναλώνουν οι multi-threaded εφαρμογές. Συνοπτικά, μέσω του Clock Modulation δίνεται η δυνατότητα στον χρήστη να επιλέξει το ποσοστό των κύκλων, στο οποίο ο επεξεργαστής λειτουργεί και εκτελεί εντολές. Το HyPart αποτελείται από 16 x 15 x 10 = 2400 πιθανούς συνδυασμούς των παραπάνω 3 τεχνικών, όπου το 16 αντιστοιχεί στους πυρήνες που χρησιμοποιούνται κατά το Thread Packing, το 15 στις διαφορετικές επιλογές του Clock Modulation και το 10 στις διαφορετικές επιλογές του MBA. Τέλος, σύμφωνα με τα αποτελέσματα της έρευνας, ο μηχανισμός HyPart αποτελείται από μεγαλύτερο δυναμικό εύρος (dynamic range) και παρέχει αρκετά μεγαλύτερο βαθμό λεπτομέρειας (granularity) στον διαμοιρασμό του DRAM bandwidth, συγκριτικά με τις υπόλοιπες τρεις τεχνικές όταν αυτές εφαρμόζονται μόνες τους στο σύστημα.

Στην παρούσα διπλωματική εργασία, θα χρησιμοποιήσουμε την τεχνολογία Intel RDT, στην οποία βασίζεται ο μηχανισμός DICER. Προκειμένου, να επεκτείνουμε τον μηχανισμό ώστε να αντιμετωπίζει αποτελεσματικότερα τον κορεσμό του DRAM bandwidth και να προστατευθεί περαιτέρω η επίδοση της HP εφαρμογής, θα χρησιμοποιήσουμε την λογική της τεχνικής του Thread Packing, εφαρμόζοντας την σε διεργασίες.

Κεφάλαιο 3

Χαρακτηριστικά και Συνεκτελέσεις Μετροπρογραμμάτων

3.1 Παρουσίαση Μετροπρογραμμάτων και Πειραματική Εκτέλεση

Στο παρόν κεφάλαιο, παρουσιάζονται τα μετροπρογράμματα (benchmarks) που χρησιμοποιήθηκαν για την επιβεβαίωση της ύπαρξης του προβλήματος του ανταγωνισμού των κοινόχρηστων πόρων σε ένα πολυεπεξεργαστικό σύστημα, την αξιολόγηση του μηχανισμού DICER και την περαιτέρω βελτίωση του. Πιο συγκεκριμένα, χρησιμοποιήθηκε η πιο πρόσφατη σουίτα benchmarks SPEC, δηλαδή τα SPEC CPU2017([18]). Η συγκεκριμένη σουίτα, περιλαμβάνει δύο κατηγορίες benchmarks, τις εκδόσεις rate και τις εκδόσεις speed. Ακόμη, για πρώτη φορά, η σουίτα SPEC περιλαμβάνει και multi-threaded εκδόσεις benchmarks, πέρα από τις single-threaded. Το σύνολο των διαφορετικών benchmarks και των δύο εκδόσεων ανέρχεται στα 43, ενώ ορισμένα από αυτά περιέχουν περισσότερες από μία εισόδους, οπότε συνυπολογίζοντας τες, το σύνολο γίνεται 55.

Κάνοντας χρήση των τεχνολογιών CMT-CAT-MBM, πραγματοποιήσαμε και λάβαμε μετρήσεις για τις ακόλουθες περιπτώσεις εκτελέσεων των benchmarks:

- Αρχικά, εκτελέσαμε κάθε ένα από τα 55 διαφορετικά benchmarks, μόνο του στο σύστημα, δηλαδή χωρίς να εκτελείται κάποια άλλη εφαρμογή στον server, δίνοντας και τα 25 MB (20 ways) στην διάθεσή του. Την μέτρηση υπό αυτές τις συνθήκες, ονομάζουμε 'Alone'.
- Στην επόμενη φάση, προκειμένου να πραγματοποιήσουμε το profiling του κάθε διαφορετικού benchmark, εκτελέσαμε κάθε ένα από αυτά δίνοντας του διαφορετικά μεγέθη cache. Πιο συγκεκριμένα, εξετάσαμε όλες τις διαφορετικές περιπτώσεις, δίνοντας από 1 εώς 19 ways of associativity, με βήμα 1 way κάθε φορά. Και σε αυτή την φάση των

μετρήσεων, κάθε benchmark εκτελούταν μόνο του, δίχως να τρέχει κάποιο άλλο πρόγραμμα παράλληλα.

- Όλα τα multi-threaded benchmarks εκτελέστηκαν χρησιμοποιώντας μόνο ένα thread, προκειμένου να χαρακτηριστούν ως single-threaded εφαρμογές.
- Τέλος, σημειώνεται πως ο αριθμός δίπλα σε κάθε όνομα του benchmark, υποδηλώνει την εκτέλεσή του με διαφορετική είσοδο.

Ο επεξεργαστής στον οποίο πραγματοποιήθηκαν όλες οι εκτελέσεις και μετρήσεις που θα αναφερθούν παρακάτω και στα επόμενα κεφάλαια, είναι ο Intel® Xeon® Processor E5-2630 v4 και έχει τα ακόλουθα χαρακτηριστικά:

	Χαρακτηριστικά
Αρχιτεκτονική	Broadwell
Sockets	2
Φυσικοί Πυρήνες ανά Socket	10
Λογικοί Πυρήνες ανά Socket	20
Συχνότητα	2.20 GHz
Last-Level Cache (LLC)	Inclusive 25 MB
Ways of Associativity	20
DRAM Bandwidth	68.3 GB/sec
Intel CMT-CAT-MBM	✓
Intel MBA	✗

Πίνακας 3.1 : Χαρακτηριστικά του επεξεργαστή Intel® Xeon® Processor E5-2630 v4

3.2 Επίδοση Benchmarks συναρτήσει του δοθέντος χώρου στην Last-Level Cache

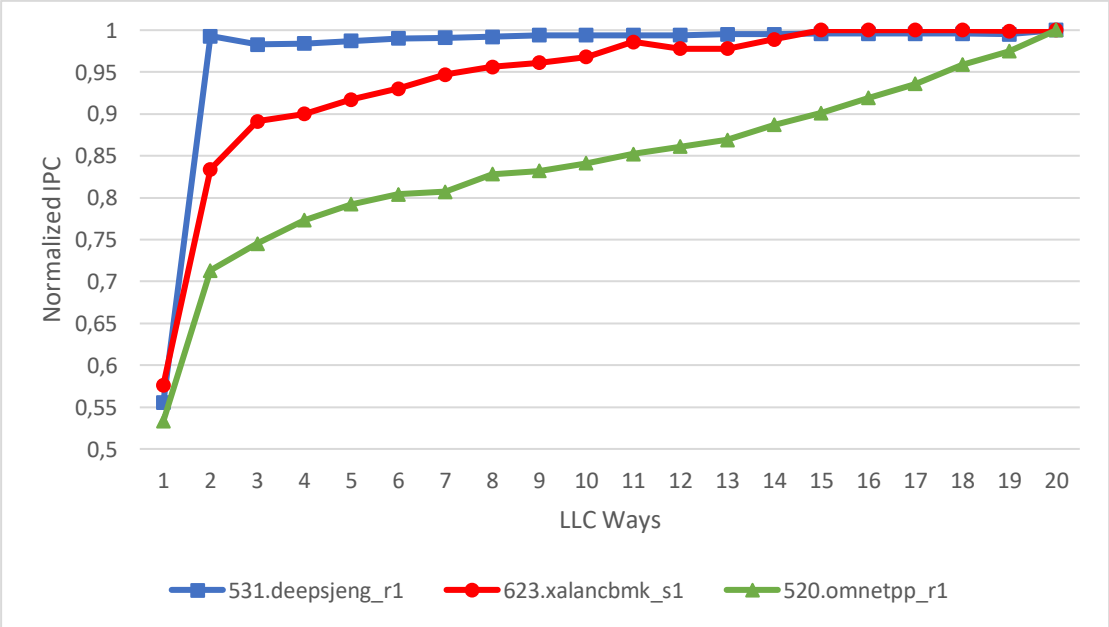
Στο παρόν υποκεφάλαιο, στόχος είναι η εκτίμηση της μείωσης της επίδοσης του κάθε benchmark όταν του δίνεται λιγότερος χώρος στην μνήμη LLC για χρήση. Συνεπώς, εκτελέσαμε κάθε benchmark 20 φορές (i.e. 20 ways of associativity), αυξάνοντας κάθε φορά το διαθέσιμο χώρο της LLC για χρήση. Η επίδοση του benchmark συναρτήσει του χώρου στην LLC, λαμβάνεται από το IPC που πετυχαίνει στο τέλος της εκτέλεσής του. Παρακάτω ομαδοποιούμε τα benchmarks σε 3 κατηγορίες, ανάλογα με την επίδοση που είχαν. Η ομαδοποίηση γίνεται με κριτήριο με το πόσα ways of associativity χρειάζεται κάθε benchmark για να πετύχει το 95% της επίδοσης όταν αυτό εκτελείται με διαθέσιμο χώρο όλη την LLC, δηλαδή και τα 20 ways of associativity και παρουσιάζεται στον παρακάτω πίνακα. Στην πρώτη κατηγορία κατατάσσουμε

τα benchmarks που χρειάζονται από 1 εώς 5 ways, στην δεύτερη κατηγορία αυτά που απαιτούν από 6 εώς 10 ways, ενώ στην τρίτη κατηγορία αυτά που χρειάζονται από 11 εώς 20 ways.

Κατηγορίες	Αριθμός Benchmarks	Ποσοστό (%)
Μικρή απαίτηση για LLC	31	56
Μέση απαίτηση για LLC	11	20
Μεγάλη απαίτηση για LLC	13	24

Πίνακας 3.2 : Κατηγοριοποίηση benchmarks συναρτήσει των απαιτήσεων για LLC για την επίτευξη του 95% της επίδοσης της απομονωμένης εκτέλεσής τους

Στο παρακάτω σχήμα, παρουσιάζεται η γραφική παράσταση του κανονικοποιημένου IPC τριών benchmarks ως προς αυτό της απομονωμένης εκτέλεσης με 20 ways, συναρτήσει του δοθέντος χώρου στην LLC. Το 531.deepsjeng_r1 ανήκει στην κατηγορία των benchmarks που έχουν μικρή απαίτηση για LLC, το 623.xalancbmk_s1 έχει μέση απαίτηση για LLC, ενώ το 520.omntepp_r1 απαιτεί μεγάλο χώρο στην LLC, ώστε να έχει επίδοση μεγαλύτερη του 95% της απομονωμένης εκτέλεσής του. Τέλος, στον πίνακα 3.3 παρουσιάζονται αναλυτικά τα benchmarks με τα απαιτούμενα ways στην LLC, ώστε να αποδώσουν τουλάχιστον στο 95% της απομονωμένης εκτέλεσης τους.



Σχήμα 3.1 : Γραφική παράσταση κανονικοποιημένου IPC ως προς αυτό της απομονωμένης εκτέλεσης, συναρτήσει του δοθέντος χώρου στην LLC

	Benchmark	LLC Ways	Benchmark	LLC Ways
Μικρή απαίτηση για LLC	500.perlbench_r2	2	503.bwaves_r1	3
	503.bwaves_r2	3	503.bwaves_r3	3
	503.bwaves_r4	3	507.cactuBSSN_r1	5
	508.namd_r1	2	510.parest_r1	5
	511.povray_r1	2	521.wrf_r1	2
	525.x264_r1	2	525.x264_r2	2
	525.x264_r3	2	526.blender_r1	4
	527.cam4_r1	4	531.deepsjeng_r1	2
	538.imagick_r1	1	541.leela_r1	2
	544.nab_r1	2	548.exchange2_r1	1
	549.fotonik3d_r1	3	557.xz_r2	4
	600.perlbench_s1	4	600.perlbench_s2	3
	625.x264_s1	2	625.x264_s2	2
	625.x264_s3	2	631.deepsjeng_s1	2
	641.leela_s1	2	648.exchange2_s1	1
	649.fotonik3d_s1 2	2		
Μέση απαίτηση για LLC	500.perlbench_r1	6	502.gcc_r1	9
	502.gcc_r3	10	505.mcf_r1	7
	554.roms_r1	6	605.mcf_s1	8
	623.xalancbmk_s1	8	602.gcc_s1	6
	602.gcc_s2	6	602.gcc_s3	6
	657.xz_s1 6	6		
Μεγάλη απαίτηση για LLC	500.perlbench_r3	15	502.gcc_r2	13
	502.gcc_r4	16	502.gcc_r5	13
	519.lbm_r1	11	520.omnetpp_r1	18
	523.xalancbmk_r1	15	557.xz_r1	17
	557.xz_r3	12	600.perlbench_s3	14
	619.lbm_s1	14	620.omnetpp_s1	18
	657.xz_s2	13		

Πίνακας 3.3 : Απαίτηση χώρου για LLC του κάθε benchmark για την επίτευξη του 95% της επίδοσης της απομονωμένης εκτέλεσης του

3.3 Χρονική Καθυστέρηση της Υψηλής Προτεραιότητας Εφαρμογής

Έχοντας εξάγει και μελετήσει τα αποτελέσματα των χαρακτηριστικών του κάθε benchmark, ομαδοποιήσαμε ορισμένα benchmarks από τα 35 της κατηγορίας rate, που παρουσίασαν παρόμοια χαρακτηριστικά ως προς την μεταβολή του IPC συναρτήσει του χώρου LLC που απαιτούν, όπως περιγράφθηκε στην προηγούμενη υποενότητα, καταλήγοντας να κρατήσουμε 28 από αυτά για τις επόμενες μετρήσεις μας. Ακόμη, επιλέξαμε 7 benchmarks της κατηγορίας speed, με κριτήριο την ανάγκη τους για DRAM bandwidth. Συνεπώς, το σύνολο των benchmarks που χρησιμοποιήθηκαν στις επόμενες ενότητες ανέρχεται σε 35. Η επιλογή των benchmarks έγινε, προκειμένου να μειωθεί ο συνολικός αριθμός των συνεκτελέσεων που προκύπτουν. Παρακάτω παρουσιάζονται αναλυτικά τα benchmarks που επιλέξαμε.

500.perlbench_r1	500.perlbench_r2	500.perlbench_r3	502.gcc_r3
502.gcc_r4	502.gcc_r5	503.bwaves_r1	505.mcf_r1
507.cactuBSSN_r1	508.namd_r1	510.parest_r1	511.povray_r1
519.lbm_r1	520.omnetpp_r1	521.wrf_r1	523.xalancbmk_r1
525.x264_r1	526.blender_r1	527.cam4_r1	531.deepsjeng_r1
538.imagick_r1	541.leela_r1	544.nab_r1	548.exchange2_r1
549.fotonik3d_r1	554.roms_r1	557.xz_r1	557.xz_r3
620.omnetpp_s1	623.xalancbmk_s1	602.gcc_s1	631.deepsjeng_s1
605.mcf_s1	649.fotonik3d_s1	619.lbm_s1	

Πίνακας 3.4 : Παρουσίαση επιλεγμένων benchmarks

Έπειτα, με τα 35 benchmarks που επιλέξαμε, δημιουργούμε όλους τους πιθανούς συνδυασμούς συνεκτελέσεων, δηλαδή 35 x 35 = 1225 συνδυασμοί. Πιο συγκεκριμένα, χρησιμοποιούμε το ίδιο σενάριο συνεκτέλεσης με την εργασία ([11]), κατά το οποίο χαρακτηρίζουμε μία εφαρμογή ως υψηλής προτεραιότητας (HP) και εννέα αντίγραφα της ίδιας ή μίας άλλης εφαρμογής, ως χαμηλής προτεραιότητας (BE). Στόχος της συνεκτέλεσης, είναι να αναδείξουμε το πρόβλημα ανταγωνισμού κοινόχρηστων πόρων, για LLC και DRAM bandwidth. Κατά την συνεκτέλεση, επανεκιννούμε κάθε benchmark μέχρι όλα να τελειώσουν τουλάχιστον μία φορά για λόγους δικαιοσύνης. Το τελικό IPC της κάθε εφαρμογής προκύπτει συνυπολογίζοντας το IPC όλων των φορών που εκείνη τελείωσε την εκτέλεση της. Ακόμη, πρέπει να σημειωθεί πως κάθε

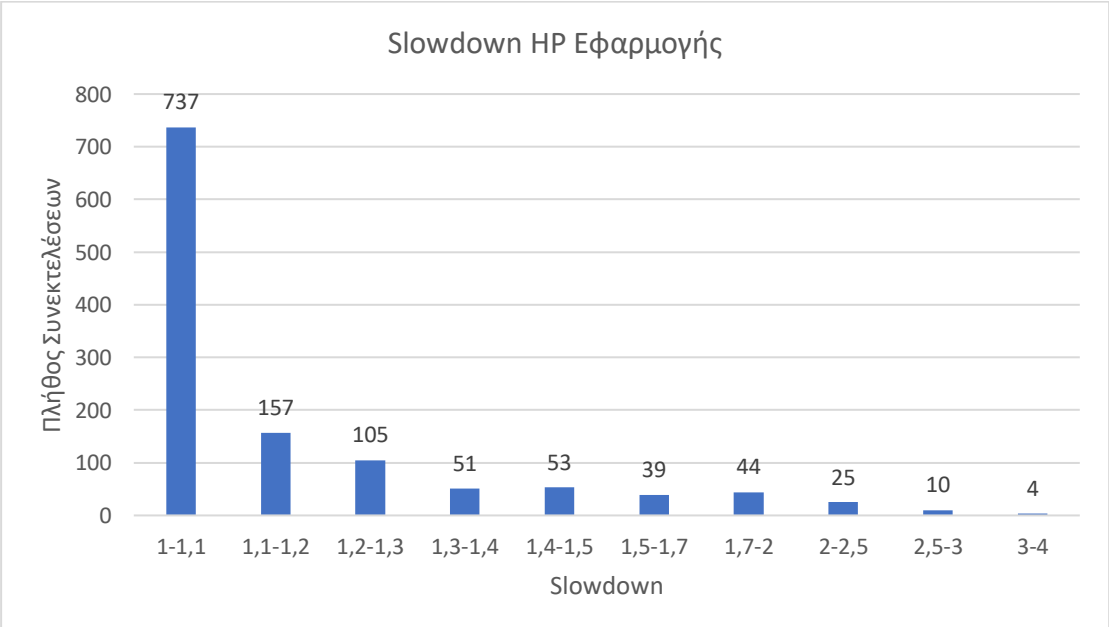
εφαρμογή αντιστοιχίζεται σε έναν και μόνο πυρήνα του ενός socket και δεν υπάρχει επικάλυψη (overlap) μεταξύ αυτών. Τέλος, δεν γίνεται χρήση του hyperthreading, και συνεπώς όλες οι εφαρμογές εκτελούνται στους physical cores.

Στην συνέχεια, μελετάται η επίδραση των BE εφαρμογών στην επίδοση της HP εφαρμογής. Για την αξιολόγηση της χρονικής καθυστέρησης (Slowdown) της HP εφαρμογής, χρησιμοποιούνται δύο πολιτικές εκτέλεσης.

- Unmanaged (UM) πολιτική, κατά την οποία όλες οι εφαρμογές, HP και BEs, δεν έχουν κανένα περιορισμό ως προς τον χώρο που τους αποδίδεται στην μνήμη LLC.
- Cache-Takeover (CT) πολιτική, κατά την οποία δίνονται τα 19 ways of associativity της LLC στην HP εφαρμογή και το εναπομένων 1 way στις 9 BE εφαρμογές.

3.3.1 Πολιτική Unmanaged (UM)

Ως χρονική καθυστέρηση της HP εφαρμογής, ορίζουμε την χρονική απόκλιση, συγκριτικά με την απομονωμένη εκτέλεσή της. Συνεπώς, για την UM πολιτική, ορίζουμε $\text{Slowdown} = \frac{IPC_{StandAlone}}{IPC_{UM}}$, όπου $IPC = \frac{Total\ Instructions}{Total\ Cycles}$, $IPC_{StandAlone}$, η επίδοση της εφαρμογής όταν εκτελείται μόνη της στο σύστημα και IPC_{UM}, η επίδοση της HP εφαρμογής όταν συνεκτελείται με εννέα αντίγραφα εφαρμογών χαμηλής προτεραιότητας κατά την UM πολιτική.



Σχήμα 3.2 : Κατανομή workloads με κριτήριο το Slowdown κατά την UM πολιτική

Στο σχήμα 3.2 παρουσιάζεται η κατανομή των workloads σε σχέση με την τιμή του Slowdown της HP εφαρμογής, κατά την πολιτική UM. Παρατηρούμε ότι κατά την πολιτική UM, για το περίπου 40% των workloads σημειώνεται σημαντική χρονική καθυστέρηση, από 1,1 εώς 4, λόγω του ανταγωνισμού κοινόχρηστων πόρων. Στην συνέχεια, θα παρουσιαστούν τρεις περιπτώσεις συνεκτελέσεων από διαφορετικές κατηγορίες Slowdown.

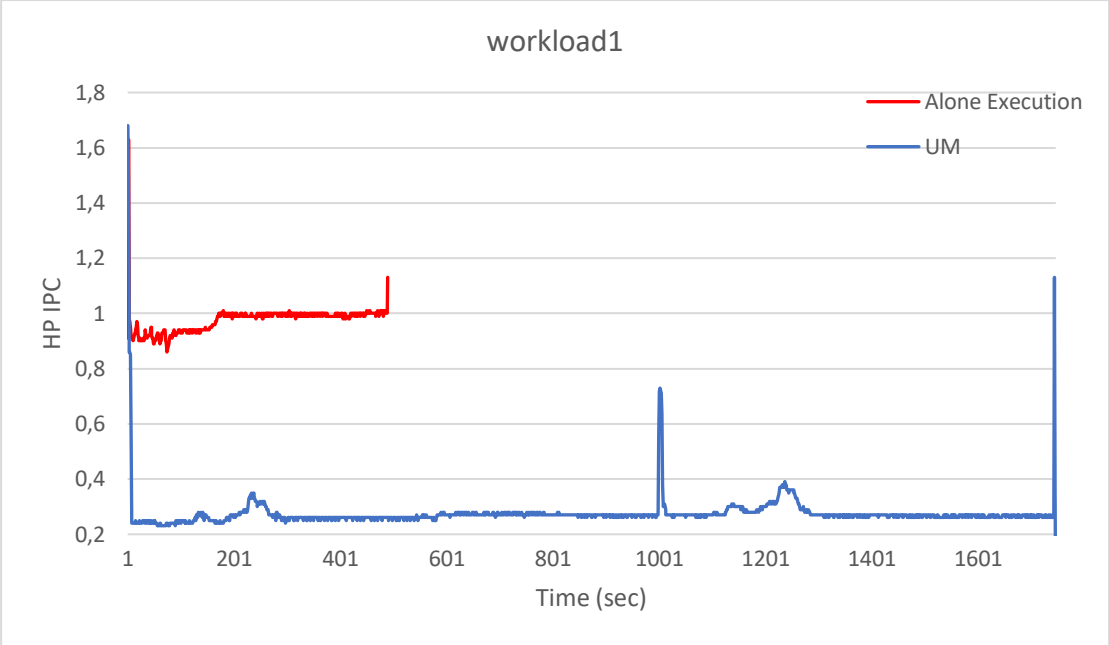
Για την μεγαλύτερη ευκολία του αναγνώστη, στον πίνακα που ακολουθεί, αντιστοιχίζουμε τα workloads που θα παρουσιαστούν στον παρόν, καθώς και στα επόμενα κεφάλαια, σε ονόματα. Για κάθε workload, το πρώτο μέλος αντιστοιχεί στο όνομα της HP εφαρμογής, ενώ το δεύτερο μέλος αντιστοιχεί στο όνομα των εφαρμογών που χρησιμοποιήθηκαν ως BE εφαρμογές.

Workload	Όνομα
520.omnetpp_r1-549.fotonik3d_r1	workload1
549.fotonik3d_r1-549.fotonik3d_r1	workload2
503.bwaves_r1-502.gcc_r3	workload3
557.xz_r1-549.fotonik3d_r1	workload4
549.fotonik3d_r1-502.gcc_r5	workload5
500.perlbench_r3-519.lbm_r1	workload6
520.omnetpp_r1-510.parest_r1	workload7
520.omnetpp_r1-519.lbm_r1	workload8
520.omnetpp_r1-520.omnetpp_r1	workload9
649.fotonik3d_s1-502.gcc_r5	workload10
549.fotonik3d_r1-505.mcf_r1	workload11

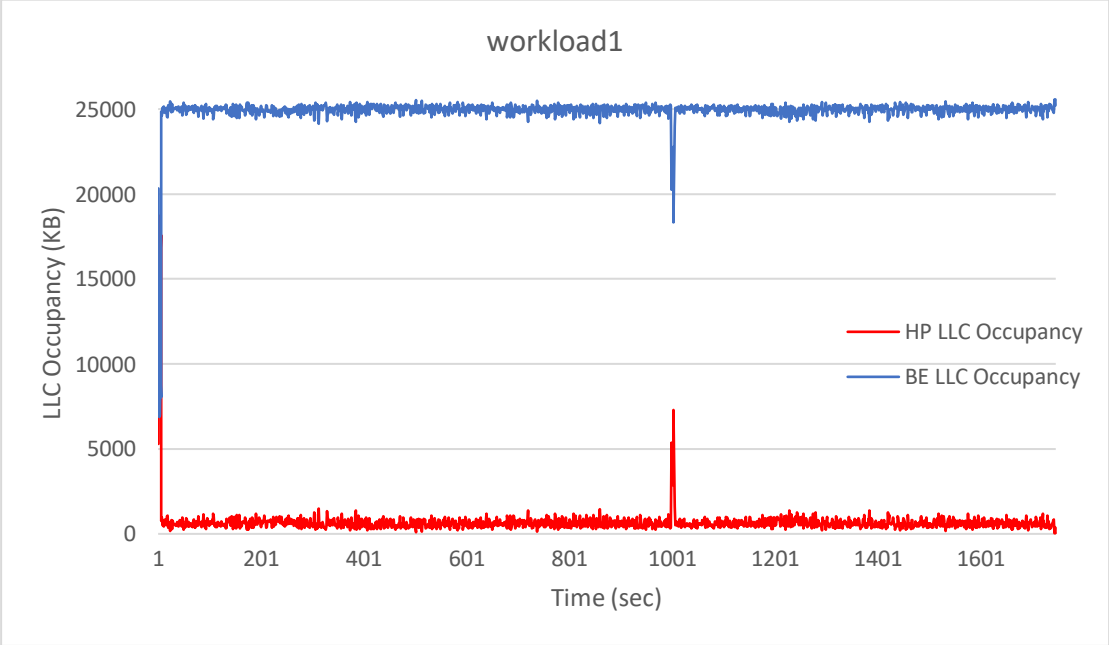
Πίνακας 3.5 : Αντιστοίχιση των workloads σε ονόματα

Μία περίπτωση μεγάλου Slowdown, παρουσιάζεται στο σχήμα 3.3 κατά την εκτέλεση του workload1. Στο συγκεκριμένο workload, η τιμή του Slowdown είναι αρκετά μεγάλη και αντιστοιχεί σε 3,625. Παρατηρούμε ότι η επίδραση των BE εφαρμογών είναι σημαντική στην επίδοση της HP, αφού υπερτριπλασιάζουν το χρόνο εκτέλεσής της. Ακόμη, από το σχήμα 3.4, παρατηρούμε την κύρια αιτία για την μειωμένη επίδοσή της. Κατά την συνεκτέλεση, η HP εφαρμογή λαμβάνει μόνο 600 KB της LLC κατά μέσο όρο, σε αντίθεση με την απομονωμένη εκτέλεσή της, όπου χρησιμοποιεί όλη την LLC. Σύμφωνα με την κατηγοριοποίηση που είχαμε κάνει στη ενότητα 3.2, παρατηρήσαμε πως το 520.omnetpp_r1 χρειάζεται 18 ways της LLC, δηλαδή 23040 KB, για να πετύχει το 95% της επίδοση της απομονωμένης εκτέλεσής της. Συνεπώς, γίνεται αντιληπτό πως η αιτία για την μειωμένη επίδοση της HP

εφαρμογής κατά το workload1, οφείλεται στον μειωμένο χώρο που αυτή λαμβάνει στην LLC, λόγω της επίδρασης των BE εφαρμογών.



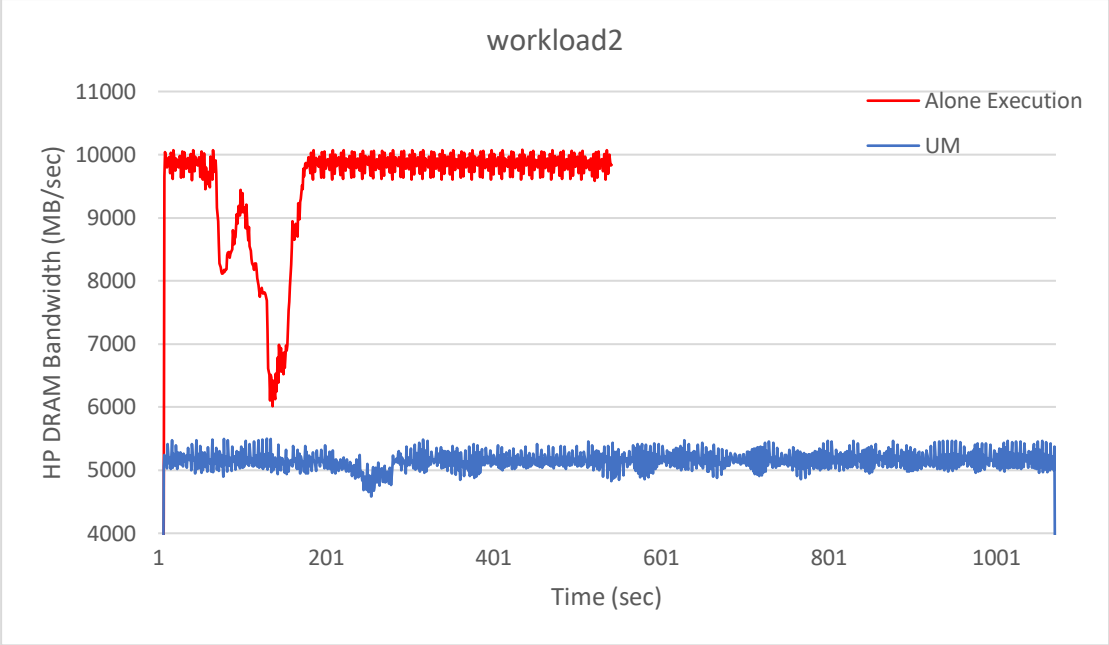
Σχήμα 3.3 : Σύγκριση IPC του 520.omnetpp_r1 κατά την απομονωμένη εκτέλεση και την εκτέλεση του workload1



Σχήμα 3.4 : Χρήση LLC των HP και BE εφαρμογών κατά την εκτέλεση του workload1

Στο σχήμα 3.5, παρουσιάζεται μία μέτρια περίπτωση Slowdown με τιμή 1,98 κατά την εκτέλεση του workload2. Σύμφωνα με την κατηγοριοποίηση που κάναμε στην ενότητα 3.2, το 549.fotonik3d_r1 είναι ένα benchmark, το οποίο χρειάζεται μόλις 3 LLC ways, δηλαδή 3840 KB, για να πετύχει επίδοση κοντά σε αυτήν της απομονωμένης εκτέλεσής του. Κατά την διάρκεια της εκτέλεσης του workload2, η HP εφαρμογή λαμβάνει κατά μέσο όρο 3500 KB, αρκετά κοντά στο μέγεθος που χρειάζεται για να πετύχει το 95% της εκτέλεσής της. Όμως, το 549.fotonik3d_r1 είναι ένα benchmark που παρουσιάζει υψηλή απαίτηση για DRAM bandwidth, όπως φαίνεται στο σχήμα 3.6. Συνεπώς, λαμβάνοντας υπόψιν την μικρή απαίτησή του για LLC όπως προαναφέρθηκε, συμπεραίνουμε πως η επίδοση του 549.fotonik3d_r1 εξαρτάται από την διαθεσιμότητα του DRAM bandwidth. Ακόμη, παρατηρούμε πως κατά την συνεκτέλεση του με 9 αντίγραφα του ιδίου benchmark, το 549.fotonik3d_r1 λαμβάνει λίγο περισσότερο από το μισό DRAM bandwidth της απομονωμένης εκτέλεσής του και αυτή είναι η αιτία για την μειωμένη επίδοσης της HP εφαρμογής.

Σχήμα 3.5 : Σύγκριση IPC του 549.fotonik3d_r1 κατά την απομονωμένη εκτέλεση και την εκτέλεση του workload2



Σχήμα 3.6 : Σύγκριση χρήσης DRAM bandwidth του 549.fotonik3d_r1 κατά απομονωμένη εκτέλεση και την εκτέλεση του workload2

Στην συνέχεια, μία μικρή εώς και αμελητέα περίπτωση χρονικής καθυστέρησης παρουσιάζεται στο σχήμα 3.7, κατά την εκτέλεση του workload3. Η τιμή του Slowdown για την HP εφαρμογή είναι 1,016. Σύμφωνα με την ανάλυση που κάναμε στην ενότητα 3.2, το 503.bwaves_r1, χρειάζεται 3 LLC ways, δηλαδή 3840 KB για να πετύχει το 95% της απόδοσής της απομονωμένης εκτέλεσής του. Κατά την διάρκεια της εκτέλεσης του workload3, η HP εφαρμογή λαμβάνει κατά μέσο όρο 4450 KB, ακόμη περισσότερο χώρο δηλαδή από αυτόν που χρειάζεται για να πετύχει το 95% της επίδοσης. Ακόμη, όπως φαίνεται από το σχήμα 3.8, παρατηρούμε πως το DRAM bandwidth της HP εφαρμογής είναι ελάχιστα αυξημένο συγκριτικά με αυτό της απομονωμένης εκτέλεσής του, το οποίο φαίνεται να επηρεάζει ελάχιστα την επίδοση της HP εφαρμογής.

Σχήμα 3.7 : Σύγκριση IPC του 503.bwaves_r1 κατά την απομονωμένη εκτέλεση και την εκτέλεση του workload3

Σχήμα 3.8 : Σύγκριση χρήσης DRAM bandwidth του 503.bwaves_r1 κατά απομονωμένη εκτέλεση και την εκτέλεση του workload3

3.3.2 Πολιτική Cache-Takeover (CT)

Όπως έχουμε αναφέρει, κατά την πολιτική UM δεν γίνεται κάποιος διαμοιρασμός της LLC και συνεπώς, όλες οι εφαρμογές εκτελούνται χωρίς κανένα περιορισμό ως προς τους διαθέσιμους, σε αυτές, πόρους. Όμως, όπως είδαμε στην προηγούμενη υποενότητα, η πολιτική UM μπορεί να βλάψει από ελάχιστα ως σημαντικά την επίδοση μίας εφαρμογής, που στο δικό μας σενάριο συνεκτέλεσης, αυτή είναι η εφαρμογή υψηλής προτεραιότητας. Συνεπώς, γίνεται αντιληπτή η ανάγκη για την απομόνωση και τον διαμοιρασμό του τμήματος της LLC που λαμβάνει η HP εφαρμογή για την προστασία της επίδοσής της.

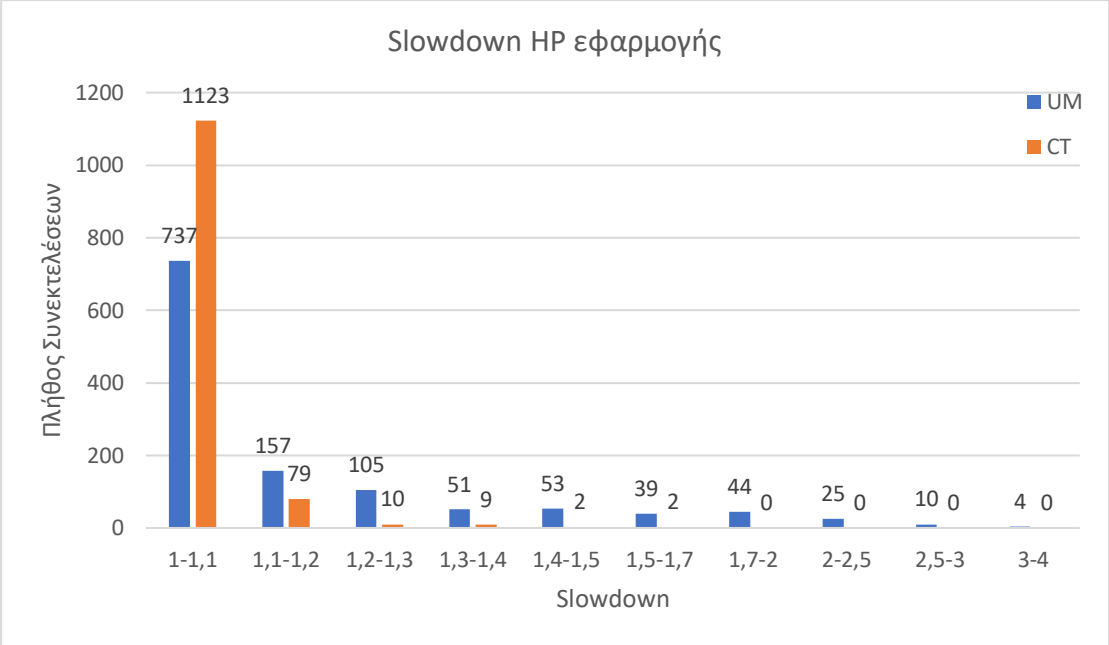
Αντίθετα με την λογική της UM, η πολιτική CT έχει σκοπό να διαφυλάξει την επίδοση της εφαρμογής υψηλής προτεραιότητας. Αυτή η πολιτική αποτελεί τον καλύτερο στατικό διαμοιρασμό της LLC, αφού αποδίδει στην HP εφαρμογή το 95% τις LLC (i.e. 19 ways) και το εναπομένων 5% (i.e. 1 way) στις 9 BE εφαρμογές. Το βασικό μειονέκτημα αυτής της πολιτικής είναι ότι μειώνεται η επίδοση των BE εφαρμογών, καθώς αποδίδουμε το μεγαλύτερο μέρος στην HP εφαρμογή, ανεξαρτήτου εάν αυτό είναι απαραίτητο για την διαφύλαξη της επίδοσής της, όπως για παράδειγμα σε benchmarks που έχουν μικρή ή μέση απαίτηση για LLC που παρουσιάστηκαν στην ενότητα 3.2. Παρ 'όλα αυτά, η CT πολιτική, παραμένει ο καλύτερος τρόπος διασφάλισης της επίδοσης μια εφαρμογής σε ένα περιβάλλον συνεκτέλεσης και για αυτό λαμβάνουμε τις μετρήσεις και για αυτήν την πολιτική. Συνεπώς, αυτό που αναμένουμε από την CT πολιτική είναι ότι θα βελτιώσει την χρονική καθυστέρηση της HP εφαρμογής συγκριτικά με την UM, αφού της αποδίδεται το 95% του μεγέθους της LLC και είναι απομονωμένη από τις υπόλοιπες εφαρμογές.

Με παρόμοιο τρόπο όπως στην ενότητα 3.3.1, ορίζουμε την χρονική καθυστέρηση της CT πολιτικής ως $Slowdown = \frac{IPC_{StandAlone}}{IPC_{CT}}$, όπου $IPC = \frac{Total\ Instructions}{Total\ Cycles}$, $IPC_{StandAlone}$, η επίδοση της εφαρμογής όταν εκτελείται μόνη της στο σύστημα και IPC_{CT} , η επίδοση της εφαρμογής όταν συνεκτελείται με εννέα αντίγραφα εφαρμογών χαμηλής προτεραιότητας, έχοντας στην διάθεση της τα 19 ways τις LLC.

Στο σχήμα 3.9, όπου γίνεται η σύγκριση της χρονικής καθυστέρησης μεταξύ των δύο πολιτικών, UM και CT, παρατηρούμε πως η δεύτερη πολιτική μειώνει τον αριθμό των workloads στα οποία παρατηρείται σημαντική χρονική καθυστέρηση. Συγκεκριμένα, μόνο στο 8% των workloads παρατηρείται Slowdown μεγαλύτερο του 1,1 , σε αντίθεση που το ίδιο ποσοστό αντιστοιχεί σε 40% κατά την UM. Παρόλα αυτά, όπως φαίνεται στο σχήμα, η CT πολιτική δεν εξαλείφει εντελώς το φαινόμενο της χρονικής καθυστέρησης. Αυτό

συμβαίνει, διότι, ο ανταγωνισμός για τις κοινόχρηστους πόρους, όπως το DRAM bandwidth, τους prefetchers της LLC, κτλ., συνεχίζει να υπάρχει κατά την CT πολιτική.

Μάλιστα, σύμφωνα με τα αποτελέσματα των μετρήσεων που λάβαμε, διαπιστώσαμε πως υπάρχουν workloads, κατά τα οποία η CT πολιτική είτε δεν προσφέρει βελτίωση είτε μειώνει την επίδοση της HP εφαρμογής. Για παράδειγμα, το Slowdown του workload5 αυξάνεται από 1,17 χρησιμοποιώντας την UM πολιτική, σε 1,3 χρησιμοποιώντας την πολιτική CT. Ένα ακόμη παράδειγμα αποτελεί το workload10, του οποίου η χρονική καθυστέρηση αυξάνεται από 1,14 σε 1,24 χρησιμοποιώντας την CT πολιτική.



Σχήμα 3.9 : Σύγκριση κατανομής Slowdown κατά τις πολιτικές UM και CT

3.4 Κατηγοριοποίηση Συνεκτελέσεων

Από τις παραπάνω υποενότητες, γίνεται αντιληπτή η ανάγκη κατηγοριοποίησης των workloads με βάση την επίδοση τους κατά τις δύο πολιτικές που περιγράφθηκαν, αφού παρατηρήσαμε πως η επίδοση ορισμένων workloads δεν βελτιώνεται με την χρήση της πολιτικής CT. Συνεπώς, δημιουργούμε δύο κατηγορίες, CT-Favoured και CT-Thwarted. Στην κατηγορία CT-Favoured (CT-F) ανήκουν όλα τα workloads, στα οποία η επίδοση της HP εφαρμογής βελτιώνεται με την χρήση της CT πολιτικής, δηλαδή $IPC_{CT} > IPC_{UM}$. Στην κατηγορία CT-Thwarted (CT-T) κατατάσσουμε τα εναπομείναντα workloads, δηλαδή σε αυτά που η επίδοση της HP εφαρμογής παραμένει ίδια ή μειώνεται με την χρήση της CT πολιτικής συγκριτικά με την UM πολιτική, δηλαδή $IPC_{CT} \leq IPC_{UM}$. Παρακάτω, παρουσιάζεται η κατανομή των workloads στις δύο κατηγορίες.

	Κατηγορίες	
	CT-Favoured	**CT-Thwarted**
Αριθμός workloads	759	466
Ποσοστό (%)	62	38

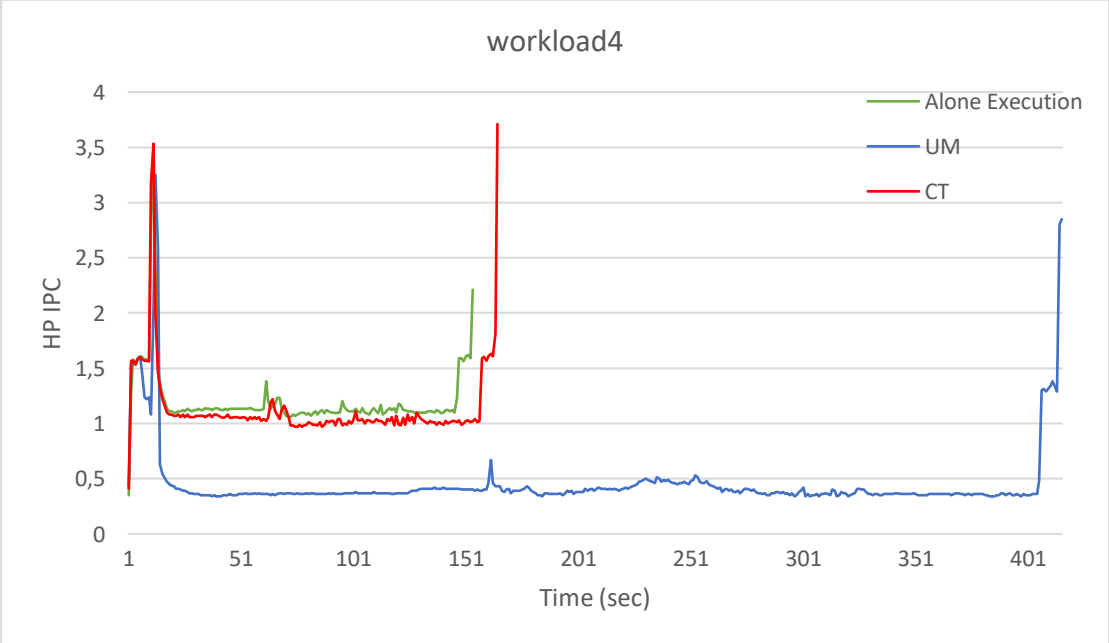
Πίνακας 3.6 : Κατανομή συνεκτελέσεων ανά κατηγορία

Συγκρίνοντας τα παραπάνω αποτελέσματα με τα αποτελέσματα της εργασίας ([11]), στην οποία οι ερευνητές χρησιμοποίησαν την σουίτα SPEC CPU2006, παρατηρούμε πως υπάρχει μεγάλη απόκλιση της κατανομής των workloads στις δύο κατηγορίες. Πιο συγκεκριμένα, σε εκείνη την εργασία, τα workloads κατηγορίας CT-F αποτελούσαν το 40% και τα workloads κατηγορίας CT-T αποτελούσαν το 60%. Αυτή η απόκλιση οφείλεται στην διαφορά των απαιτήσεων και την φύση των benchmarks της σουίτας SPEC CPU2017 και SPEC CPU2006, καθώς σύμφωνα με την εργασία ([19]), οι ερευνητές διαπιστώνουν πως η σουίτα SPEC CPU2017 ασκεί λιγότερη πίεση στην ιεραρχία των μνημών cache συγκριτικά με την προκάτοχο σουίτα.

Ακόμη, κατά την μελέτη των αποτελεσμάτων των συνεκτελέσεων που αναφέρθηκαν παραπάνω, παρατηρήσαμε πως ο κορεσμός του DRAM bandwidth συμβαίνει και στις συνεκτελέσεις κατηγορίας CT-F. Αυτό έρχεται σε αντίθεση με την προηγούμενη εργασία ([11]), σύμφωνα με την οποία, ο κορεσμός του DRAM bandwidth διαπιστωνόταν μόνο σε συνεκτελέσεις κατηγορίας CT-T. Συμπεραίνουμε, λοιπόν, πως στα workloads κατηγορίας CT-F, στα οποία συμβαίνει κορεσμός στο DRAM bandwidth, η επίδοση της HP εφαρμογής περιορίζεται ακόμη παραπάνω. Συνεπώς, η πιθανή καταπολέμηση

αυτού του κορεσμού, θα επέτρεπε την περαιτέρω αύξηση της επίδοσης της υψηλής προτεραιότητας εφαρμογής, συγκριτικά με την CT πολιτική. Στην συνέχεια, παρουσιάζονται αναλυτικά δύο workloads, ένα από την κατηγορία CT-F και ένα από την κατηγορία CT-T.

Όπως έχουμε αναφέρει, στην κατηγορία CT-F ανήκουν τα workloads, των οποίων η επίδοση βελτιώνεται με την χρήση της πολιτικής CT, συγκριτικά με την UM. Χαρακτηριστικό παράδειγμα αυτής της κατηγορίας αποτελεί το workload4, που παρουσιάζεται στο σχήμα 3.10 και τον πίνακα 3.7. Οι τιμές του πίνακα έχουν κανονικοποιηθεί ως προς την απομονωμένη εκτέλεση των benchmarks, ενώ το LLC occupancy έχει κανονικοποιηθεί ως προς τα ways που χρειάζεται η HP για να πετύχει το 95% της επίδοσης της. Από την ανάλυση που είχαμε κάνει στην ενότητα 3.2, γνωρίζουμε πως το 557.xz_r1 χρειάζεται 17 LLC ways, δηλαδή 21760 KB, ώστε η επίδοση του να είναι στο 95% της απομονωμένης εκτέλεσής του. Παρατηρούμε, όμως, ότι κατά την πολιτική UM, η HP λαμβάνει μόνο το 2,35% της LLC κατά μέσο όρο, αφού δεν υπάρχει κάποιος περιορισμός μεταξύ των benchmarks, κάτι που βλάπτει σημαντικά την επίδοσή της. Αντίθετα, η CT πολιτική εξασφαλίζει τον χώρο που χρειάζεται το 557.xz_r1, αφού η μέση χρήση της LLC είναι 111,76% συγκριτικά με την εκτέλεση των 17 ways, και συνεπώς, αυξάνει την επίδοσή του κατά περίπου δυόμισι φορές, σε σχέση με την UM. Ασφαλώς, η βελτίωση της επίδοσης της υψηλής προτεραιότητας εφαρμογής έρχεται υπό το κόστος της μείωσης της επίδοσης των χαμηλής προτεραιότητας εφαρμογών, αφού ο γεωμετρικός μέσος όρος των IPC υποδιπλασιάζεται, όπως φαίνεται από τον πίνακα.

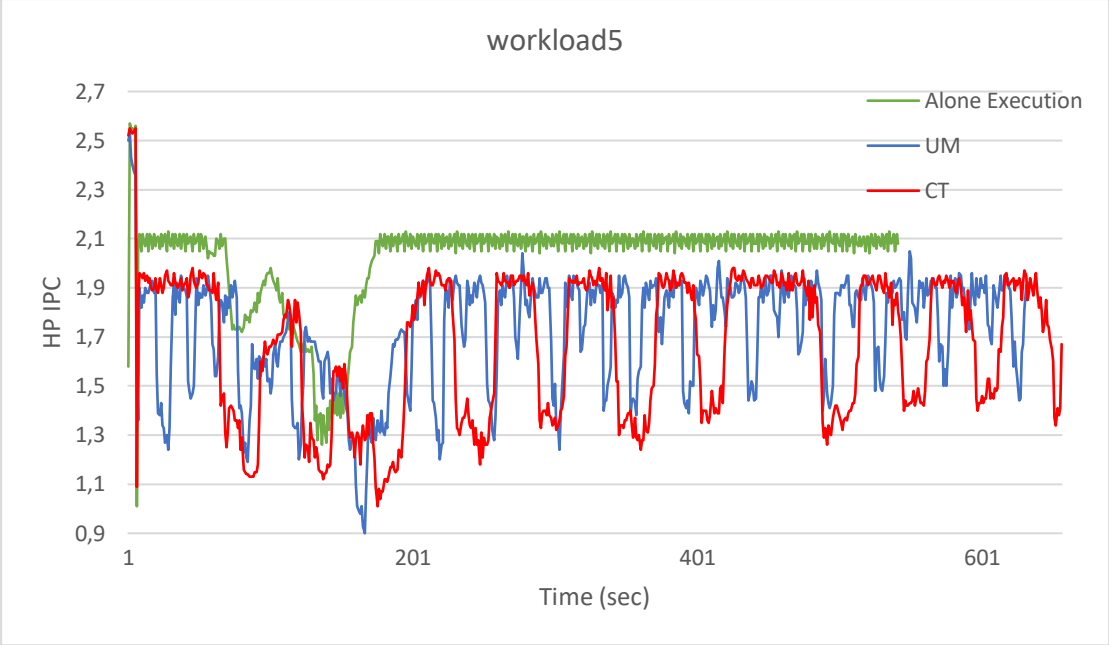


Σχήμα 3.10 : Σύγκριση IPC του 557.xz_r1 κατά την απομονωμένη εκτέλεση του και κατά τις εκτελέσεις του workload4 με τις UM και CT πολιτικές

workload4	UM	CT
HP IPC	0,373	0,942
Avg. BE IPC	0,547	0,234
Avg. HP LLC Occupancy (%)	2,35	111,76

Πίνακας 3.7 : Ληφθέντες μετρήσεις από την εκτέλεση του workload4 κατά τις πολιτικές UM και CT

Στην κατηγορία CT-T έχουμε κατατάξει τα workloads, στα οποία η επίδοση της HP εφαρμογής είτε δεν βελτιώνεται, είτε μειώνεται με την χρήση της CT πολιτικής. Ένα παράδειγμα αυτής της κατηγορίας αποτελεί το workload5, το οποίο παρουσιάζεται στο σχήμα 3.11 και τον πίνακα 3.8. Οι τιμές του πίνακα, εκτός από την μέση τιμή του συνολικού DRAM bandwidth, έχουν κανονικοποιηθεί ως προς την απομονωμένη εκτέλεση των benchmarks, ενώ το LLC occupancy έχει κανονικοποιηθεί ως προς τα ways που χρειάζεται η HP ώστε να πετύχει το 95% της επίδοσης της. Όπως βλέπουμε, η απόδοση του 549.fotonik3d_r1 μειώνεται με την χρήση της CT πολιτικής. Όπως, έχουμε προαναφέρει η απόδοση του συγκεκριμένου benchmark εξαρτάται από την διαθεσιμότητα για χρήση του DRAM bandwidth στο σύστημα. Παρατηρούμε, λοιπόν, πως ενώ με την χρήση της πολιτικής CT και την διασφάλιση ενός μεγαλύτερου χώρου της LLC, η επίδοση της HP εφαρμογής μειώνεται. Αυτό συμβαίνει γιατί εφαρμόζοντας αυτή την πολιτική, οι BE εφαρμογές λαμβάνουν μόνο 1 way της LLC κατά την εκτέλεση τους, το οποίο αυξάνει το DRAM bandwidth που καταναλώνουν. Πιο συγκεκριμένα, δίνοντας μόνο το 5% της LLC στις BE εφαρμογές, αυξάνεται το ποσοστό αστοχιών τους στην μνήμη, έχοντας ως αποτέλεσμα να αυξάνεται το DRAM bandwidth, αφού χρειάζεται να φέρνουν συνεχώς τα απαραίτητα, για την εκτέλεση τους, δεδομένα στην μνήμη. Το φαινόμενο, αυτό, παρουσιάζεται στον πίνακα 3.8, σύμφωνα με τον οποίο, το συνολικό DRAM bandwidth κατά την UM πολιτική είναι 33200 (MB/sec), σε αντίθεση με αυτό της CT που είναι κατά περίπου 20% μεγαλύτερο. Συνεπώς, όταν αυξάνεται το συνολικό DRAM bandwidth που καταναλώνουν οι εφαρμογές, τότε μειώνεται η διαθεσιμότητα του για την HP εφαρμογή. Αυτό επιβεβαιώνεται και από τα χαρακτηριστικά του workload5, αφού το DRAM bandwidth της HP εφαρμογής μειώνεται από την UM στην CT πολιτική κατά 13%, γεγονός που κάνει την CT πολιτική χειρότερη από την UM για αυτό το workload. Τέλος παρατηρούμε, πως παρόλο η επίδοση της HP εφαρμογής μειώνεται, υποτριπλασιάζεται και η επίδοση των BE εφαρμογών με την χρήση της CT πολιτικής.



Σχήμα 3.11 : Σύγκριση IPC του 549.fotonik3d_r1 κατά την απομονωμένη εκτέλεση του και κατά τις εκτελέσεις του workload4 με τις UM και CT πολιτικές

workload5	UM	CT
HP IPC	0,856	0,772
Avg. BE IPC	0,691	0,224
Avg. HP LLC Occupancy (%)	140,63	633,33
Avg. HP Bandwidth (%)	85	72
Avg. Total Bandwidth (MB/sec)	33200	40500

Πίνακας 3.8 : Ληφθέντες μετρήσεις από την εκτέλεση του workload5 κατά τις πολιτικές UM και CT

Κεφάλαιο 4

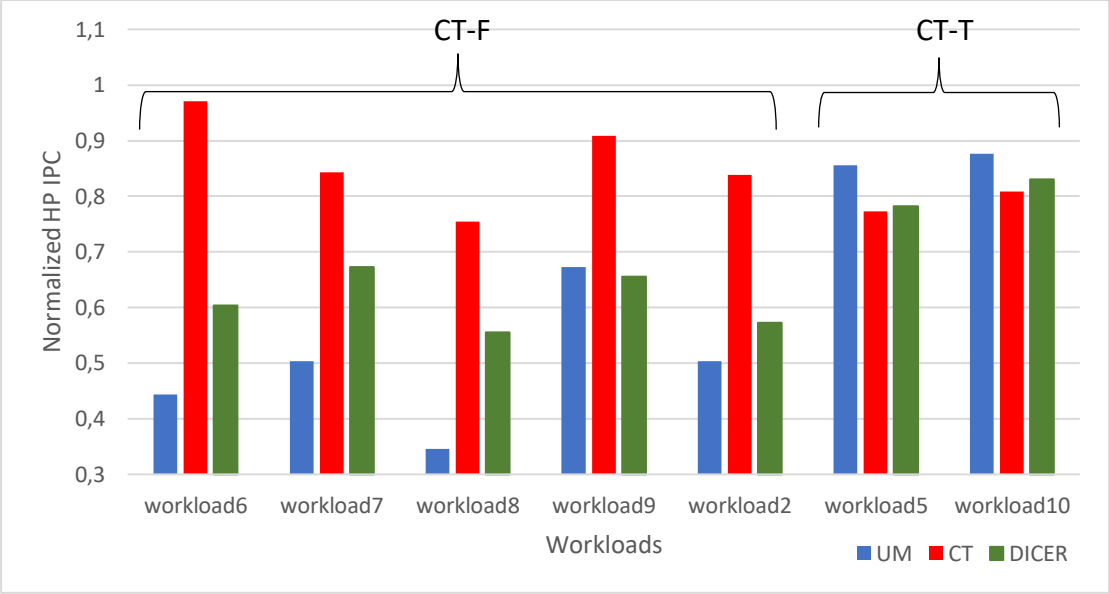
Μηχανισμός DICER-TP

4.1 Κίνητρο της Εργασίας

Έχοντας εκτελέσει όλους τους πιθανούς συνδυασμούς που αναφέρθηκαν στο προηγούμενο κεφάλαιο με τις πολιτικές UM και CT, εκτελέσαμε τα ίδια workloads χρησιμοποιώντας τον μηχανισμό DICER. Από τα αποτελέσματα που λάβαμε, παρατηρήσαμε πως υπήρχαν περιπτώσεις, στις οποίες η επίδοση της HP εφαρμογής είχε μεγάλη απόκλιση από την επίδοση που είχε παρατηρηθεί είτε με την UM, είτε με την CT πολιτική. Στο σχήμα 4.1 παρουσιάζεται το κανονικοποιημένο IPC των HP εφαρμογών ως προς αυτό της απομονωμένης εκτέλεσης τους, για ορισμένες περιπτώσεις workloads, από την κατηγορία CT-F και την κατηγορία CT-T. Παρατηρούμε ότι η επίδοση της HP εφαρμογής χρησιμοποιώντας τον μηχανισμό DICER αποκλίνει αρκετά από την αντίστοιχη επίδοση της CT πολιτικής για workloads κατηγορίας CT-F, ενώ αποκλίνει και από την αντίστοιχη επίδοση με την UM πολιτική για workloads κατηγορίας CT-T. Μάλιστα, παρατηρούμε συγκεκριμένα πως για το workload9, το οποίο ανήκει στην κατηγορία CT-F, ο μηχανισμός DICER πετυχαίνει χειρότερη απόδοση κατά 2% ακόμη και από αυτήν της UM πολιτικής.

Παρατηρήσαμε, λοιπόν, πως η μειωμένη επίδοση της HP εφαρμογής χρησιμοποιώντας τον μηχανισμό DICER, οφείλεται στο γεγονός πως ο μηχανισμός δεν αντιμετωπίζει αποτελεσματικά τον κορεσμό του DRAM bandwidth. Ο DICER προσπαθεί να αντιμετωπίσει έμμεσα τον κορεσμό του DRAM bandwidth μέσω της ανακατανομής των πόρων της LLC. Πιο συγκεκριμένα, όπως φαίνεται στο σχήμα 2.8, σε περίπτωση που το συνολικό DRAM bandwidth ξεπεράσει το κατώφλι των 50 GB/sec, τότε ο μηχανισμός αναγνωρίζει πως συνέβη κορεσμός στο DRAM bandwidth και προσπαθεί να παραχωρήσει όσο το δυνατόν μεγαλύτερο χώρο της LLC στις BE εφαρμογές, δίχως να μειώνεται η επίδοση της HP εφαρμογής. Παραχωρώντας όσα περισσότερα ways της LLC στις BE εφαρμογές, ο μηχανισμός προσπαθεί να

μειώσει τον ρυθμό αστοχιών τους, και συνεπώς το DRAM bandwidth που αυτές καταναλώνουν.

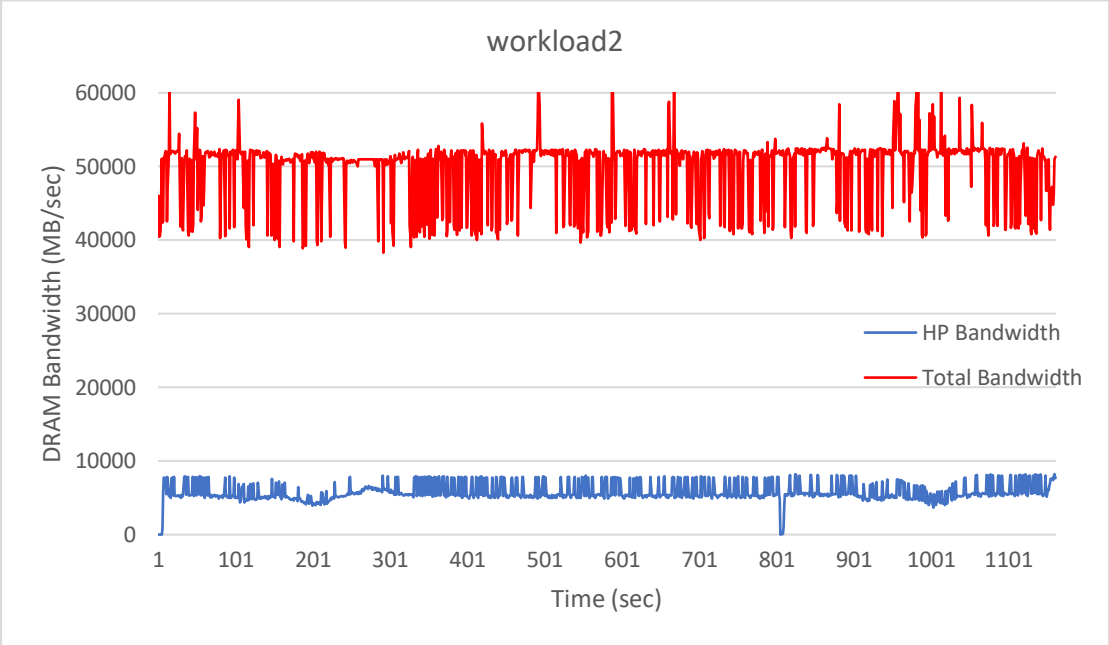


Σχήμα 4.1 : Κανονικοποιημένη επίδοση HP εφαρμογής κατά την εκτέλεση διαφόρων workloads με πολιτικές UM, CT και τον μηχανισμό DICER

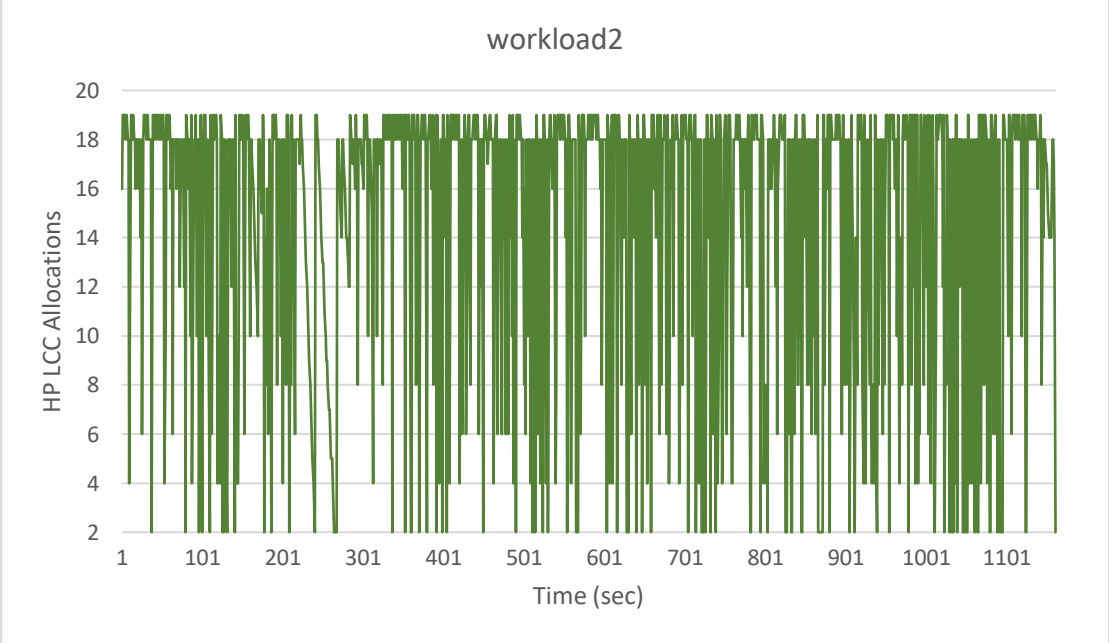
Για την επιβεβαίωση της παραπάνω παρατήρησης, παρουσιάζεται το workload2 από την κατηγορία CT-F και το workload5 από την κατηγορία CT-T. Παρακάτω, παραθέτουμε τις γραφικές παραστάσεις του συνολικού DRAM bandwidth (σχήματα 4.2 και 4.4) και των αποφάσεων που έλαβε ο μηχανισμός DICER σχετικά με τα LLC allocations της HP εφαρμογής (σχήματα 4.3 και 4.5). Παρατηρούμε, λοιπόν, από τα σχήματα του συνολικού DRAM bandwidth των δύο συνεκτελέσεων, πως ο μηχανισμός αδυνατεί να περιορίσει το συνολικό DRAM bandwidth, αφού υπάρχουν συνεχόμενα χρονικά διαστήματα, κατά τα οποία, το DRAM bandwidth παραμένει πάνω από το κατώφλι των 50 GB/sec που προαναφέρθηκε. Από τα σχήματα που απεικονίζουν τα LLC ways που αποδόθηκαν στην HP εφαρμογή, παρατηρούμε πως ο μηχανισμός DICER προσπαθεί συνεχώς να βρει το κατάλληλο σημείο, ώστε να βελτιώσει την επίδοση της HP εφαρμογής. Πιο συγκεκριμένα, βλέπουμε χρονικά διαστήματα που ο μηχανισμός αποδίδει μόλις δύο cache ways στην HP εφαρμογή, αποδίδοντας το μεγαλύτερο μέρος αυτής στις BE εφαρμογές, προκειμένου να μειώσει τα LLC misses τους και συνεπώς, το συνολικό DRAM bandwidth που καταναλώνουν, όμως αδυνατεί να το περιορίσει επιτυχώς.

Γίνεται, λοιπόν, αντιληπτή η ανάγκη της εύρεσης μίας λύσης και η περαιτέρω επέκταση του μηχανισμού, ώστε να αντιμετωπίζεται αποτελεσματικά

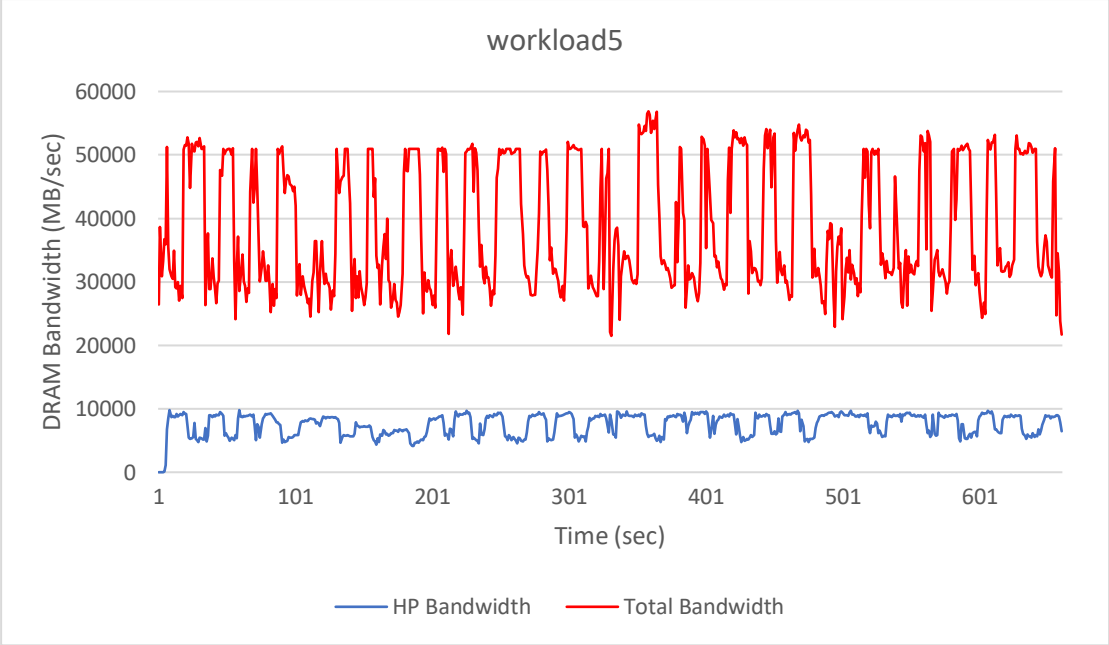
ο κορεσμός στο DRAM bandwidth στις περιπτώσεις των workloads που αυτό δεν είναι δυνατό μέσω της ανακατανομής των πόρων της LLC, προκειμένου βελτιωθεί περαιτέρω η επίδοση της HP εφαρμογής.



Σχήμα 4.2 : Χρήση DRAM bandwidth των εφαρμογών κατά την εκτέλεση του workload2



Σχήμα 4.3 : LLC Allocations της HP εφαρμογής κατά την εκτέλεση του workload2



Σχήμα 4.4 : Χρήση DRAM bandwidth των εφαρμογών κατά την εκτέλεση του workload5

Σχήμα 4.5 : LLC Allocations της HP εφαρμογής κατά την εκτέλεση του workload5

4.2 Ενσωμάτωση Thread Packing – Μηχανισμός DICER-TP

Όπως αναπτύχθηκε στην ενότητα 2.5, κατά την τεχνική του Thread Packing ορίζουμε διαφορετικά νήματα της ίδιας εφαρμογής να εκτελούνται σε λιγότερους πυρήνες από ότι αυτά είχαν αρχικά δρομολογηθεί. Εξαιτίας του ότι οι εφαρμογές που χρησιμοποιούμε για την αξιολόγηση του μηχανισμού είναι μονονηματικές, χρησιμοποιήσαμε την λογική της τεχνικής του Thread Packing, μεταφέροντας διεργασίες αντί για νήματα σε διαφορετικούς πυρήνες.

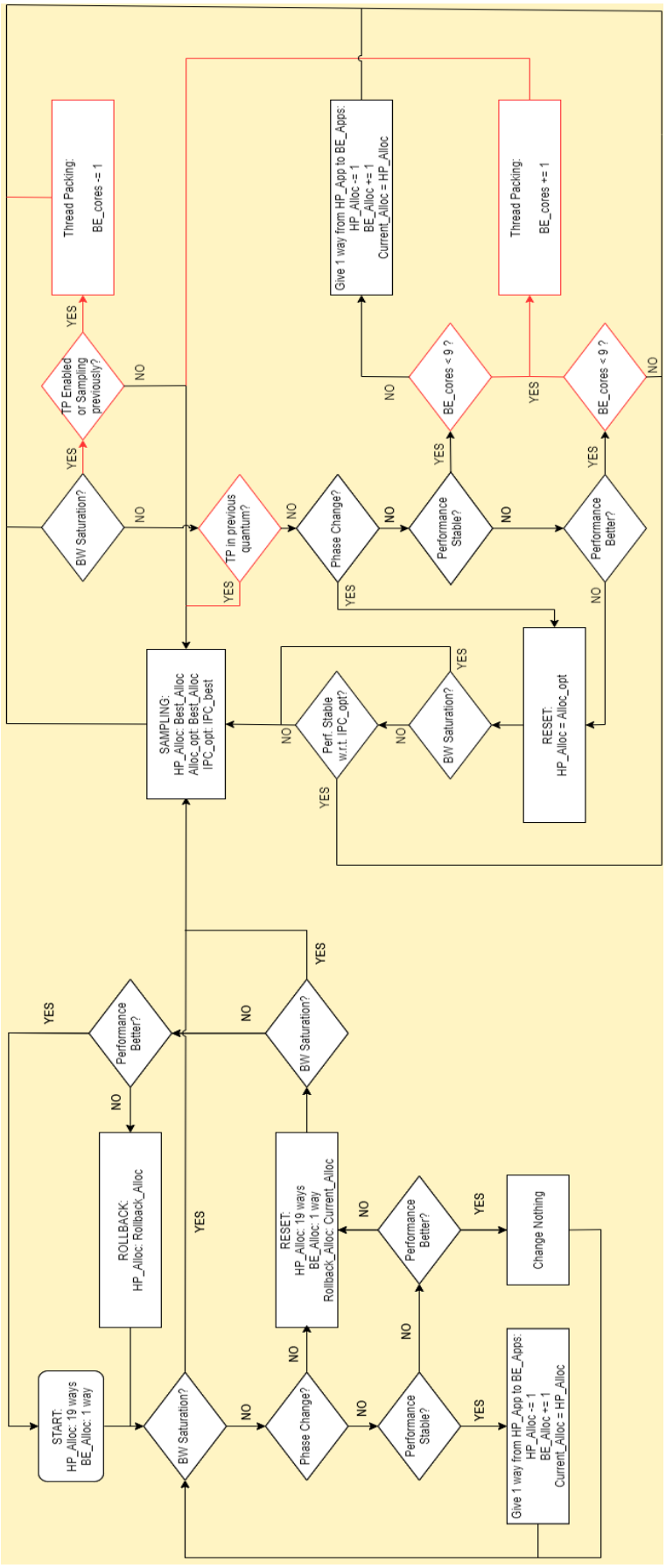
Αρχικά, για την αντιμετώπιση του κορεσμού του DRAM bandwidth, διατηρούμε την διαδικασία της ανακατανομής των πόρων της LLC. Συνεπώς, εάν σε κάποιο κβάντο του χρόνου, ο μηχανισμός αναγνωρίσει κορεσμό στο DRAM bandwidth, θα προσπαθήσει να το επιλύσει δίνοντας όσα περισσότερα ways γίνεται στις BE εφαρμογές, δίχως να επηρεάζεται η επίδοση της HP εφαρμογής. Η απόφαση για την διατήρηση της δειγματοληψίας ως αρχικό μέτρο αντιμετώπισης του κορεσμού του DRAM bandwidth έγινε διότι η μείωση των πυρήνων στους οποίους εκτελούνται οι BE εφαρμογές, βλάπτει σημαντικά την επίδοσή τους. Στην συνέχεια, εάν στο επόμενο κβάντο χρόνου ο μηχανισμός αναγνωρίσει εκ νέου κορεσμό του DRAM bandwidth, τότε αυτό αντιμετωπίζεται μέσω της μείωσης των πυρήνων που οι BE εφαρμογές έχουν στην διάθεσή τους για εκτέλεση.

Πιο συγκεκριμένα, όπως έχουμε περιγράψει προηγουμένως, το σενάριο αποτελείται από 1 HP εφαρμογή και 9 BE εφαρμογές, οι οποίες αρχικά δρομολογούνται σε 10 πυρήνες (1-1 αντιστοίχιση). Εάν ο κορεσμός του DRAM bandwidth δεν αντιμετωπιστεί επιτυχώς με την ανακατανομή των πόρων της LLC, τότε για κάθε συνεχόμενο κβάντο του χρόνου που συνεχίζει να υφίσταται, ο μηχανισμός μειώνει κατά έναν τους διαθέσιμους πυρήνες για τις BE εφαρμογές. Συνεπώς, στην χειρότερη περίπτωση δύο πυρήνες θα είναι ενεργοί κατά την συνεκτέλεση, ένας για την HP εφαρμογή και ένας για τις 9 BE εφαρμογές. Κατά την διαδικασία του Thread Packing, για την διασφάλιση της δικαιοσύνης μεταξύ των BE εφαρμογών ως προς τους διαθέσιμους πυρήνες, ο μηχανισμός δεν ορίζει σε ποιον πυρήνα θα εκτελεστεί κάθε μία από αυτές. Ο μηχανισμός δηλώνει στο λειτουργικό μόνο ποιοι πυρήνες είναι διαθέσιμοι για εκτέλεση και το λειτουργικό είναι υπεύθυνο να τις δρομολογήσει σε αυτούς. Ακόμη, όταν δρομολογηθούν παραπάνω από ένα benchmark στον ίδιο πυρήνα, το λειτουργικό εναλλάσσει με δίκαιο τρόπο αυτές τις διεργασίες, ώστε να εκτελούνται περίπου το ίδιο χρονικό διάστημα. Τέλος, στο πρώτο κβάντο χρόνου, στο οποίο έχει αντιμετωπιστεί ο κορεσμός του DRAM bandwidth, ο μηχανισμός ανακατανέμει τους πόρους της LLC, προκειμένου να βρει το

κατάλληλο LLC allocation, εφόσον οι BE εφαρμογές εκτελούνται σε λιγότερους πυρήνες.

Όταν ο μηχανισμός αναγνωρίσει ότι η επίδοση της HP εφαρμογής είτε παρέμεινε σταθερή είτε βελτιώθηκε, τότε αυξάνει κατά έναν τους διαθέσιμους πυρήνες για τις BE εφαρμογές. Η αύξηση των πυρήνων συνοδεύεται από την διαδικασία της ανακατανομής των πόρων της LLC, προκειμένου να βρεθεί το νέο κατάλληλο LLC allocation για την HP εφαρμογή. Εάν οι διαθέσιμοι πυρήνες στις BE εφαρμογές δεν έχουν μειωθεί, τότε οι αποφάσεις του νέου μηχανισμού παραμένουν ίδιες με αυτές του ήδη υπάρχοντος.

Στο σχήμα 4.6 παρουσιάζεται το ανανεωμένο διάγραμμα ροής του μηχανισμού DICER-TP, μετά την ενσωμάτωση του Thread Packing (TP). Με κόκκινο χρώμα παρουσιάζονται τα νέα στάδια που προστέθηκαν στον μηχανισμό.



Σχήμα 4.6 : Διάγραμμα ροής αποφάσεων μηχανισμού DICER-TP

Κεφάλαιο 5

Πειραματική Αξιολόγηση Μηχανισμού DICER-TP

5.1 Επιλογή Συνεκτελέσεων και Μετρικής για την Αξιολόγηση του Μηχανισμού

Όπως προαναφέρθηκε στην ενότητα 3.3 επιλέξαμε να χρησιμοποιήσουμε 35 benchmarks από τα 55, συνυπολογίζοντας αυτά με τις διαφορετικές εισόδους. Συνεπώς, δημιουργούνται 35 x 35 = 1225 συνδυασμοί συνεκτελέσεων.

Για την αξιολόγηση των αλλαγών που πραγματοποιήσαμε στον μηχανισμό DICER-TP, οι οποίες παρουσιάζονται στις ενότητες 5.2 και 5.3, επιλέξαμε 71 workloads, με κριτήριο την μεγαλύτερη μείωση της επίδοσης της HP εφαρμογής κατά την UM πολιτική για συνεκτελέσεις κατηγορίας CT-F και την μεγαλύτερη μείωση της επίδοσης της HP εφαρμογής κατά την πολιτική CT για συνεκτελέσεις κατηγορίας CT-T. Ακόμη, 44 από τα 71 workloads ανήκουν στην κατηγορία CT-F, ενώ 27 ανήκουν στην κατηγορία CT-T, διατηρώντας την αναλογία των κατηγοριών που παρουσιάστηκε στην ενότητα 3.4.

Για την τελική αξιολόγηση του μηχανισμού DICER-TP επιλέξαμε ένα δείγμα 595 συνεκτελέσεων, από τα οποία τα 378 ανήκουν στην κατηγορία CT-F και τα υπόλοιπα 217 στην κατηγορία CT-T, διατηρώντας την αναλογία των κατηγοριών που παρουσιάστηκε στην ενότητα 3.4. Η επιλογή των 595 συνεκτελέσεων έγινε χρησιμοποιώντας αλγόριθμο τυχαίας επιλογής.

Η μετρική που χρησιμοποιήθηκε για την αξιολόγηση του μηχανισμού, είναι το SLO-Effective Utilization Combined Index (SUCI), όπως ορίζεται στην εργασία ([11]). Η συνδυαστική μετρική SUCI λαμβάνει υπόψιν εάν ο μηχανισμός τήρησε το συμφωνηθέν SLO, καθώς και την χρησιμοποίηση του συστήματος που επετεύχθη. Πιο συγκεκριμένα, η μετρική SUCI ορίζεται ως εξής:

$$SUCI = c_{SLO} * EFU^{\lambda} ,$$

$$\text{όπου} \quad c_{SLO} = \begin{cases} 1, & \alpha\nu \ \frac{IPC^{HP}}{IPC^{HP}_{alone}} \geq SLO \\ 0, & \delta\iota\alpha\phi o\rho\varepsilon\tau\iota\kappa\acute{\alpha} \end{cases},$$

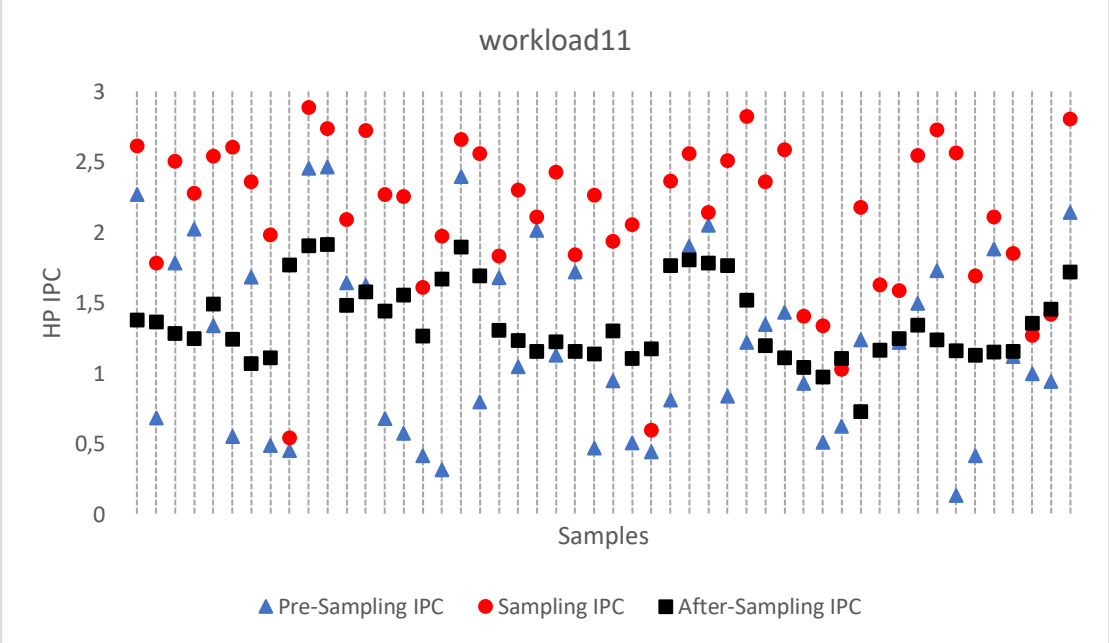
$$EFU = IPC_{norm\_hmean} = \frac{n}{\frac{IPC^{HP}_{alone}}{IPC^{HP}} + \sum_{i=0}^{n-1} \frac{IPC^{BE}_{alone,i}}{IPC^{BE}_{i}}}$$

Η παράμετρος λ της μετρικής SUCI χαρακτηρίζει την βαρύτητα της επίδοσης των εφαρμογών και λαμβάνει τιμές 0,5 , 1 και 2. Πιο συγκεκριμένα, με τιμή λ = 1, οι δύο όροι του SUCI έχουν την ίδια βαρύτητα, δηλαδή η τήρηση του SLO είναι εξίσου σημαντική με την χρησιμοποίηση του συστήματος. Με τιμή λ = 0,5 δίνεται μεγαλύτερη βαρύτητα στην τήρηση του SLO και με τιμή λ = 2 δίνεται μεγαλύτερη βαρύτητα στην χρησιμοποίηση του συστήματος. Η μετρική c_{SLO} εξετάζει εάν ο μηχανισμός τήρησε το συμφωνηθέν SLO. Συνεπώς, η τιμή του SUCI μηδενίζεται στην περίπτωση που δεν επετεύχθη η επίδοση που αναμενόταν. Ακόμη, μετράμε την χρησιμοποίηση του συστήματος με την μετρική Effective Utilization (EFU), η οποία αποτελεί τον αρμονικό μέσο όρο των IPC της HP και των BE εφαρμογών. Στο δικό μας σενάριο έχουμε συνολικά 10 εφαρμογές και συνεπώς n = 10.

Τέλος, πρέπει να σημειωθεί πως τροποποιήσαμε τον τρόπο μέτρησης του IPC των BE εφαρμογών, ώστε να λαμβάνεται υπόψιν και ο χρόνος, κατά τον οποίο αυτές είναι ανενεργές όταν ενεργοποιείται η διαδικασία του Thread Packing.

5.2 Προσδιορισμός Χρονικής Διάρκειας Δειγματοληψίας

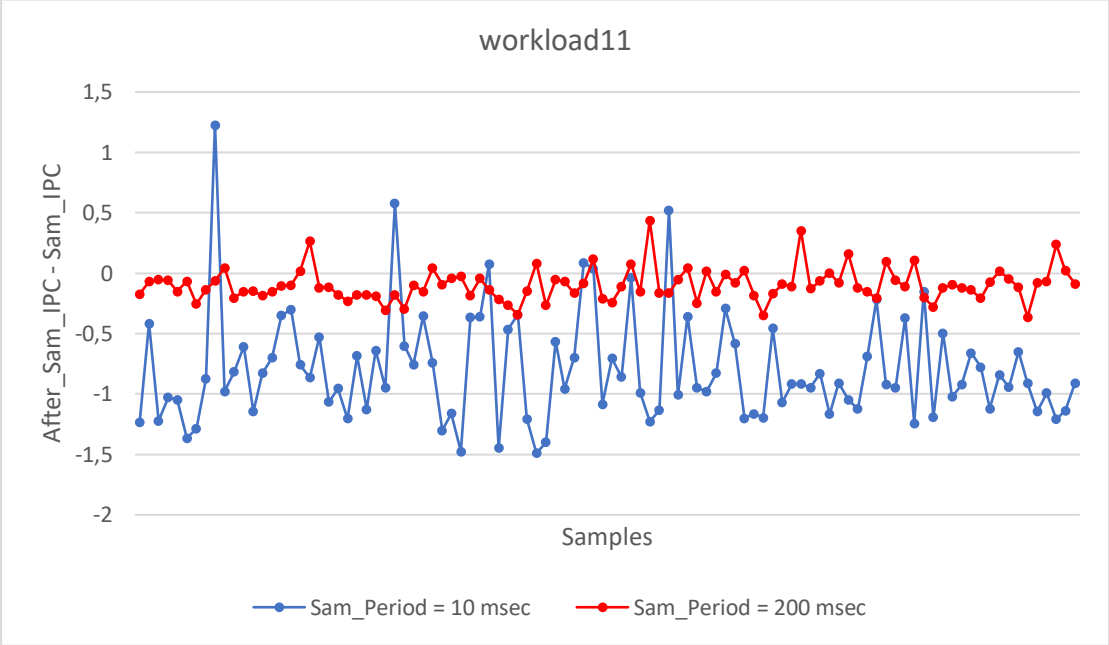
Κατά την διαδικασία της αξιολόγησης του μηχανισμού που αναφέρθηκε παραπάνω, παρατηρήσαμε πως κατά την διάρκεια της δειγματοληψίας, οι τιμές του IPC που λάμβανε ο μηχανισμός απείχαν αρκετά από αυτές πριν και μετά από αυτήν. Ο ήδη υπάρχων μηχανισμός πραγματοποιεί δειγματοληψία, χρησιμοποιώντας 10 διαφορετικά δείγματα, αποδίδοντας 19, 18, 16, 14, 12, 10, 8, 6, 4 και 2 ways στην HP εφαρμογή, με χρονική διάρκεια 10 msec για το καθένα. Συνεπώς, η συνολική διάρκεια της δειγματοληψίας είναι 100 msec. Ένα χαρακτηριστικό παράδειγμα αυτού του προβλήματος, παρατηρείται κατά την εκτέλεση του workload11. Στο σχήμα 5.1 παρουσιάζονται οι τιμές του IPC που μετρήθηκαν πριν, κατά την διάρκεια και έπειτα της δειγματοληψίας για 50 τυχαία επιλεγμένα δείγματα. Παρατηρούμε πως οι τιμές που μετρήθηκαν πριν και μετά την δειγματοληψία, απέχουν αρκετά με την τιμή του IPC που μετρήθηκε κατά την δειγματοληψία και σύμφωνα με αυτήν ο μηχανισμός έλαβε απόφαση σχετικά με το καταλληλότερο memory allocation.



Σχήμα 5.1 : IPC HP εφαρμογής πριν, μετά και κατά την δειγματοληψία σε διαφορετικές χρονικές στιγμές στην εκτέλεση του workload11

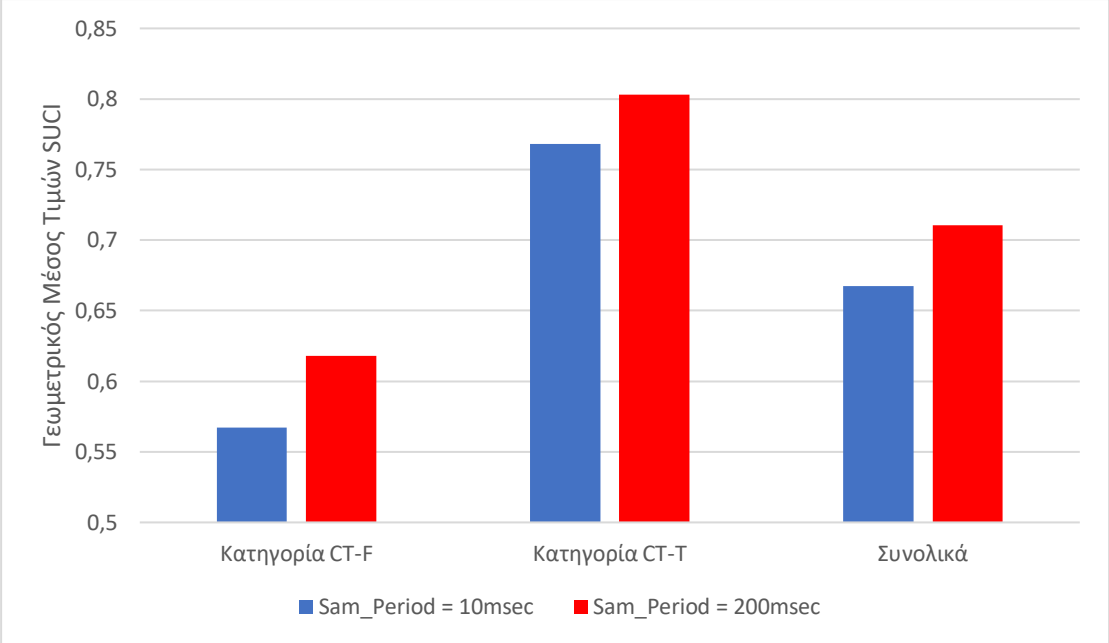
Διακρίναμε πως το αίτιο για το παραπάνω πρόβλημα αποτελούσε η χρονική διάρκεια που λαμβάναμε μετρήσεις για το κάθε δείγμα. Συνεπώς, δοκιμάσαμε διαφορετικές περιόδους εφαρμογής του κάθε δείγματος. Πιο συγκεκριμένα, μειώσαμε τα δείγματα που εφαρμόζονται στην HP εφαρμογή από

10 σε 5, αποδίδοντας σε αυτήν 19, 15, 10, 5 και 2 ways. Ακόμη, λάβαμε μετρήσεις για περίοδο εφαρμογής δείγματος για 10, 50, 100, 200 και 300 msec. Συνεπώς, η συνολική χρονική διάρκεια της δειγματοληψίας είναι 50, 250, 500, 1000 και 1500 msec αντίστοιχα. Από τα αποτελέσματα που λάβαμε, διαπιστώσαμε πως για κάποια workloads επαρκούσε η περίοδος δείγματος των 100 msec, ενώ κάποια άλλα απαιτούσαν περίοδο δείγματος 200 msec. Συνεπώς, επιλέξαμε την περίοδο των 200 msec, με συνολική διάρκεια δειγματοληψίας το 1 sec. Στο παρακάτω σχήμα, γίνεται η σύγκριση μεταξύ της υπάρχουσας χρονικής περιόδου δείγματος, 10 msec, και αυτής που εμείς επιλέξαμε ως καταλληλότερη, 200 msec. Παρατηρούμε πως η διαφορά του μετρούμενου IPC μετά την δειγματοληψία και αυτού κατά την διάρκεια της δειγματοληψίας είναι πολύ μικρότερη για 200 msec συγκριτικά με αυτής των 10 msec.



Σχήμα 5.2 : Διαφορά IPC HP εφαρμογής μετά και κατά την διάρκεια της δειγματοληψίας για διαφορετικές χρονικές περιόδους

Τέλος, στο σχήμα 5.3 παρουσιάζεται ο γεωμετρικός μέσος όρος των τιμών SUCI για λ=1 των 71 συνεκτελέσεων. Παρατηρούμε ότι αυξάνοντας την περίοδο δειγματοληψίας από 10 σε 200 msec, επιτρέπει στον μηχανισμό να λάβει πιο σωστές αποφάσεις σχετικά με την ανακατανομή των πόρων της LLC και να προστατέψει αποτελεσματικότερα την επίδοση της HP εφαρμογής. Συνεπώς, χρησιμοποιώντας την νέα περίοδο δειγματοληψίας, γίνονται λιγότερες παραβιάσεις των SLOs και επιτυγχάνεται υψηλότερη τιμή SUCI ανά κατηγορία αλλά και συνολικά.



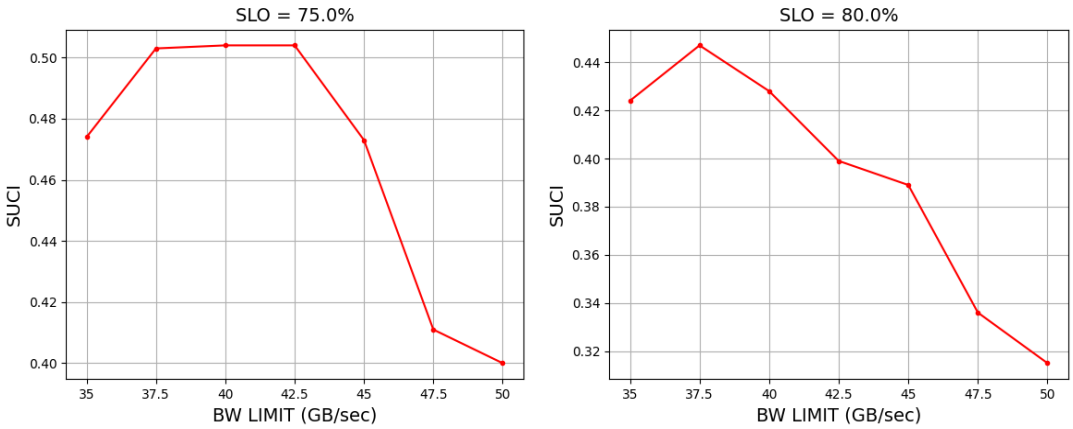
Σχήμα 5.3 : Γεωμετρικός μέσος τιμών SUCI για διαφορετικές χρονικές περιόδους δειγματοληψίας και λ=1

5.3 Προσδιορισμός Παραμέτρου που καθορίζει τον κορεσμό στο DRAM Bandwidth

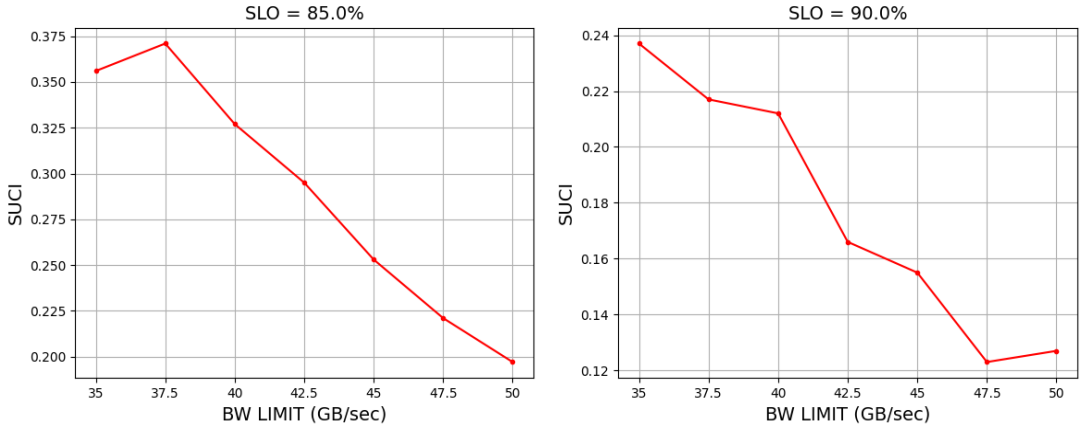
Στην συνέχεια, πραγματοποιήσαμε μετρήσεις σχετικά με το πότε δημιουργείται κορεσμός στο DRAM bandwidth. Σύμφωνα με τον μηχανισμό DICER, ο κορεσμός δημιουργείται όταν το συνολικό DRAM bandwidth όλων των εφαρμογών ξεπεράσει το κατώφλι των 50 GB/sec, το οποίο αντιστοιχεί περίπου στο 75% της ονομαστικής τιμής του διαθέσιμου DRAM bandwidth. Σύμφωνα με τις παρατηρήσεις μας, η επίδοση της HP εφαρμογής επηρεάζεται σε χαμηλότερο όριο από αυτό που προηγουμένως είχε επιλεχθεί. Συνεπώς, δοκιμάσαμε ως παράμετρο αναγνώρισης του DRAM bandwidth τα 35, 37,5, 40, 42,5, 45, 47,5 και 50 GB/sec. Οι τιμές, αυτές, αντιστοιχούν στο 50%, 55%, 60%, 65%, 70% και 75% του ονομαστικού DRAM bandwidth.

Στα παρακάτω σχήματα παρουσιάζεται ο γεωμετρικός μέσος όρος των τιμών του SUCI των 71 workloads για διαφορετικές τιμές SLO και λ=1. Παρατηρούμε πως καθώς αυξάνουμε την τιμή της παραμέτρου που ορίζει τον κορεσμό του DRAM bandwidth, η τιμή του SUCI μειώνεται, καθώς ο μηχανισμός αποτυγχάνει να εξασφαλίσει το SLO. Επιλέγουμε να χρησιμοποιήσουμε την τιμή 37,5 GB/sec, δηλαδή το 55% της ονομαστικής τιμής, ως ένδειξη κορεσμού του DRAM bandwidth στον μηχανισμό

DICER-TP, καθώς παρατηρούμε πως με αυτήν την τιμή πετυχαίνουμε το μεγαλύτερο SUCI για τα περισσότερα SLOs.



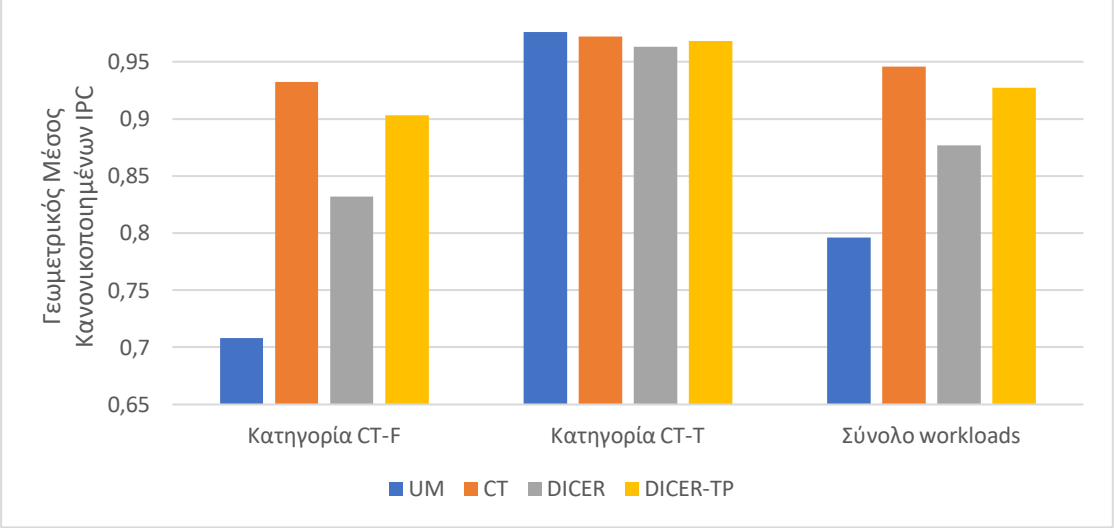
Σχήμα 5.4 : SUCI για SLO=75% και SLO=80% με λ=1



Σχήμα 5.5 : SUCI για SLO=85% και SLO=90% με λ=1

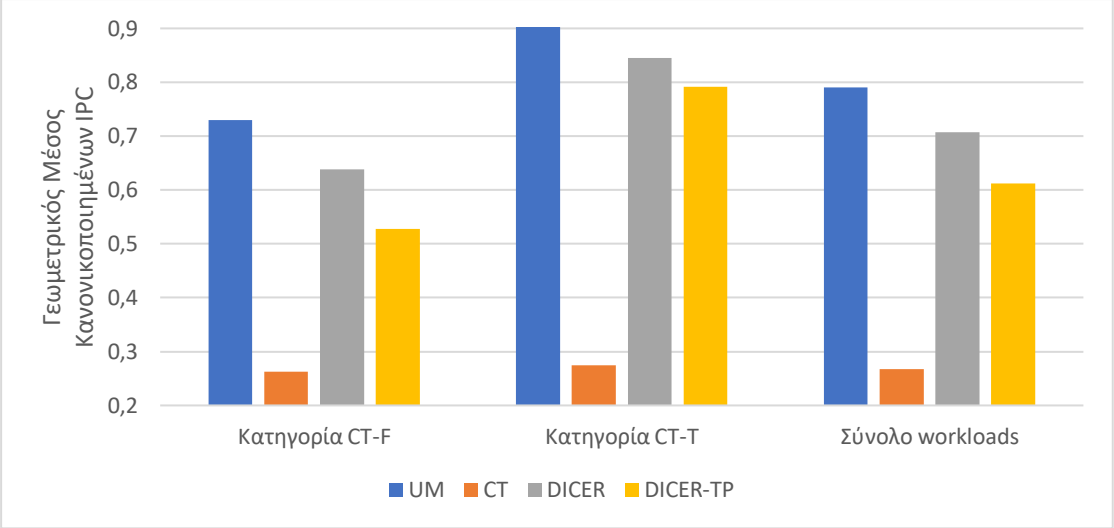
5.4 Αξιολόγηση Μηχανισμού DICER-TP – Τελικά Αποτελέσματα

Έχοντας εκτελέσει τους 595 συνδυασμούς συνεκτελέσεων που αναφέρθηκαν στην ενότητα 5.1, παραθέτουμε τα τελικά αποτελέσματα του μηχανισμού DICER-TP συγκριτικά με τις πολιτικές UM και CT, και τον μηχανισμό DICER. Συνολικά, ο μηχανισμός DICER-TP βελτιώνει την επίδοση των HP εφαρμογών κατά 13% και 5% συγκριτικά με την UM πολιτική και τον μηχανισμό DICER αντίστοιχα, ενώ αποκλίνει κατά 1,9% από την CT πολιτική. Πιο συγκεκριμένα, στην κατηγορία CT-F, ο μηχανισμός DICER-TP παρέχει κατά περίπου 20% και 7% καλύτερη επίδοση στις HP εφαρμογές από την UM και τον DICER αντίστοιχα, ενώ αποκλίνει από την CT κατά 3%. Για συνεκτελέσεις κατηγορίας CT-T, παρατηρείται μία ελάχιστη βελτίωση κατά 0,5% μεταξύ των δύο μηχανισμών, με απόκλιση μικρότερη του 1% από την βέλτιστη εκτέλεση της UM πολιτικής.



Σχήμα 5.6 : Γεωμετρικός μέσος κανονικοποιημένων IPC ως προς την απομονωμένη εκτέλεση των HP εφαρμογών

Η βελτίωση της επίδοσης των HP εφαρμογών που παρατηρείται χρησιμοποιώντας τον μηχανισμό DICER-TP, έρχεται αναμενόμενα υπό το κόστος της επίδοσης των BE εφαρμογών. Από το σχήμα 5.7 παρατηρούμε πως η επίδοση των BE εφαρμογών μειώνεται κατά 9% συγκριτικά με τον μηχανισμό DICER και αποκλίνει κατά 18% από την βέλτιστη εκτέλεση της UM πολιτικής. Όμως, ο μηχανισμός DICER-TP προσφέρει βελτίωση 34% στην επίδοση των BE εφαρμογών συγκριτικά με την πολιτική CT, χρησιμοποιώντας αποτελεσματικότερα τους διαθέσιμους πόρους του συστήματος.

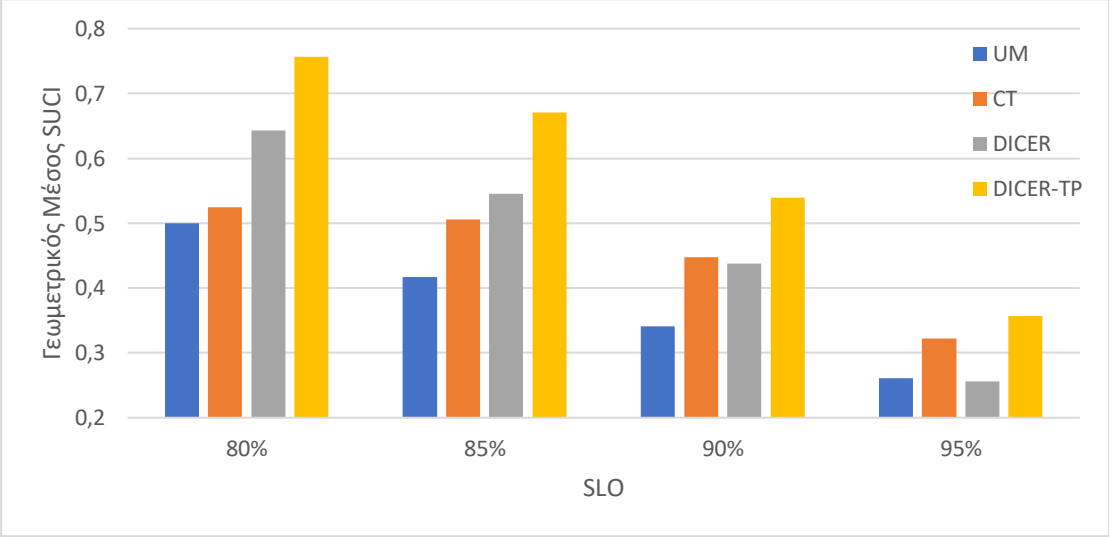


Σχήμα 5.7 : Γεωμετρικός μέσος κανονικοποιημένων IPC ως προς την απομονωμένη εκτέλεση των BE εφαρμογών

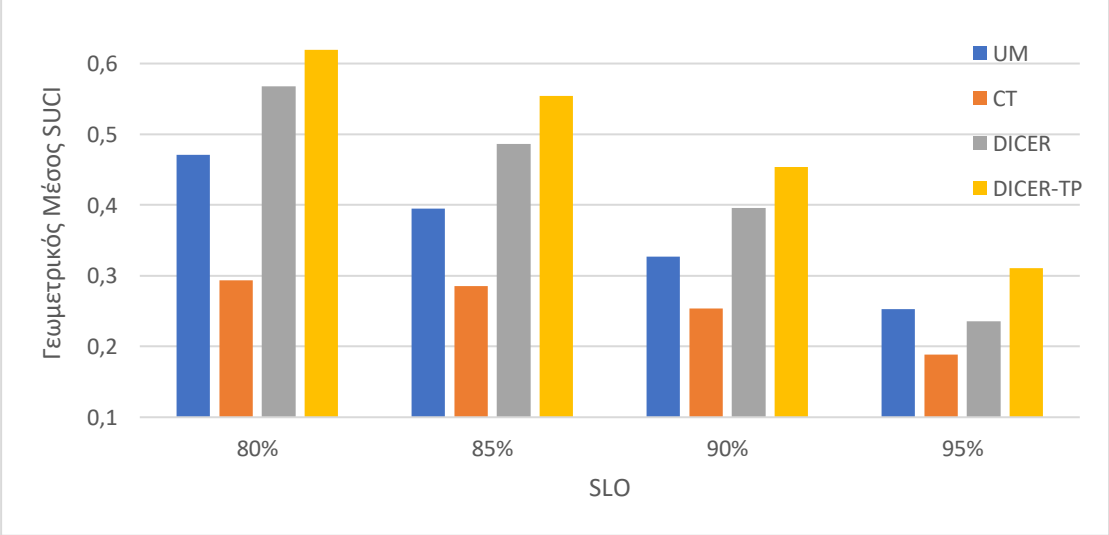
Στην συνέχεια, στα σχήματα 5.8, 5.9 και 5.10, παραθέτουμε τον γεωμετρικό μέσο των τιμών SUCI των συνεκτελέσεων κατά τις πολιτικές UM, CT και των μηχανισμών DICER και DICER-TP για τις διαφορετικές τιμές του λ. Ακόμη, στον πίνακα 5.1 παρουσιάζεται το ποσοστό του συνόλου των workloads, για τα οποία επιτυγχάνονται διαφορετικές τιμές SLO. Όταν δίνεται βαρύτητα στην επίδοση της HP εφαρμογής, δηλαδή για λ=0,5, ο νέος μηχανισμός DICER-TP ξεπερνάει αρκετά τον προκάτοχό του, αφού με την αποτελεσματικότερη καταπολέμηση του ανταγωνισμού για DRAM bandwidth, μπορεί να διασφαλίσει την επίδοση της HP εφαρμογής και συνεπώς, την τήρηση των διαφορετικών SLO, όπως φαίνεται στον πίνακα. Ακόμη, βλέπουμε πως ο μηχανισμός DICER-TP καταφέρνει να τηρήσει τα συμφωνηθέντα SLO ενώ παράλληλα αυξάνει την χρησιμοποίηση του συστήματος, αφού ξεπερνά και τις πολιτικές UM και CT. Για λ=1, δηλαδή όταν θεωρούμε πως η επίδοση της HP εφαρμογής είναι ισάξιας βαρύτητας με την επίδοση των BE εφαρμογών, ο μηχανισμός DICER-TP πετυχαίνει τα καλύτερα αποτελέσματα, ξεπερνώντας για κάθε SLO τον μηχανισμό DICER και τις πολιτικές UM και CT. Τέλος, για λ=2, δηλαδή όταν δίνεται μεγαλύτερη βαρύτητα στην επίδοση των BE εφαρμογών, παρατηρούμε πως παρόλο που ο μηχανισμός DICER-TP διασφαλίζει την επίτευξη του SLO για όλες τις περιπτώσεις, ο γεωμετρικός μέσος των τιμών SUCI είναι μεγαλύτερος κατά την UM και τον DICER, καθώς η επίδοση των BE εφαρμογών λαμβάνει μεγαλύτερη βαρύτητα. Αυξάνοντας την τιμή του SLO, ο μηχανισμός DICER-TP είναι αποτελεσματικότερος από τις δύο πολιτικές και τον DICER, καθώς πετυχαίνει στο να διασφαλίσει το SLO, ενώ διατηρεί την χρησιμοποίηση του συστήματος σε υψηλό επίπεδο.

	Ποσοστό workloads που επιτεύχθηκε το SLO			
SLO (%)	**80**	**85**	**90**	**95**
UM	61,68	52,61	43,87	34,45
CT	97,31	94,45	85,71	64,54
DICER	79,33	69,24	57,31	36,64
DICER-TP	95,46	87,06	72,27	49,92

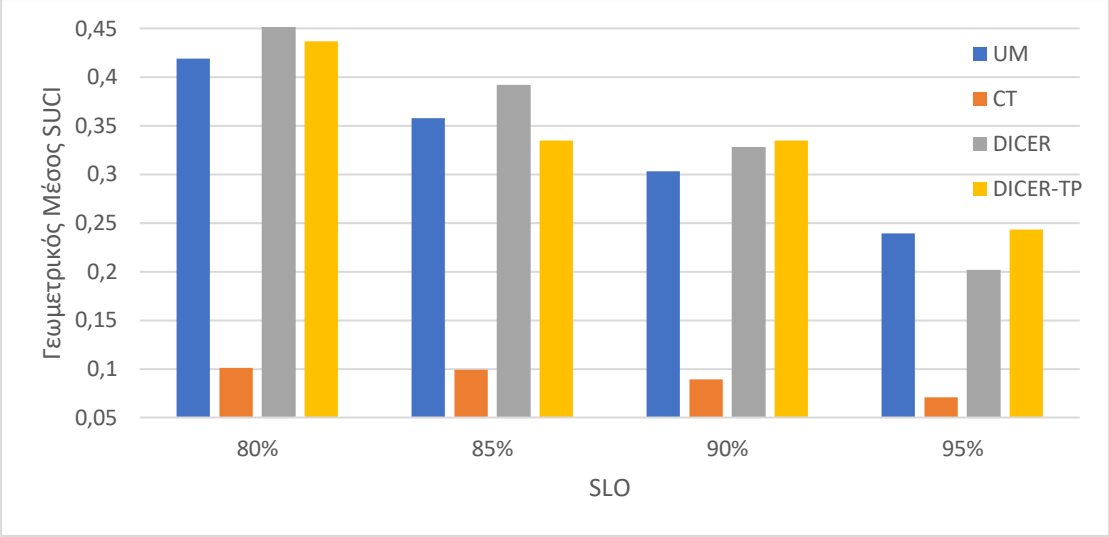
Πίνακας 5.1 : Ποσοστό workloads που επιτυγχάνονται τα SLOs με χρήση UM, CT, DICER και DICER-TP



Σχήμα 5.8 : Γεωμετρικός μέσος τιμών SUCI όλων των workloads για λ=0.5



Σχήμα 5.9 : Γεωμετρικός μέσος τιμών SUCI όλων των workloads για λ=1

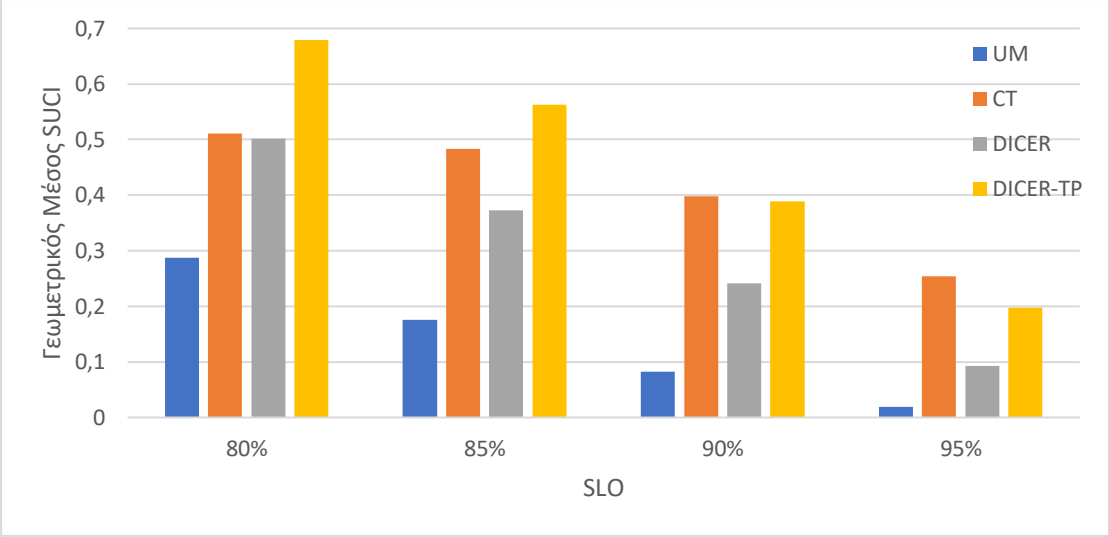


Σχήμα 5.10 : Γεωμετρικός μέσος τιμών SUCI όλων των workloads για λ=2

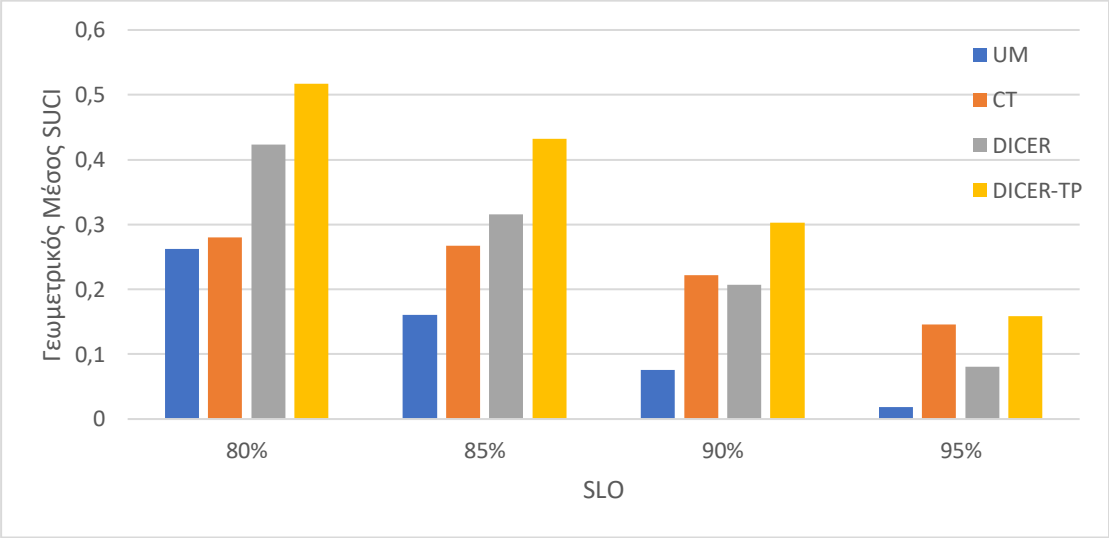
Στην συνέχεια, παρουσιάζονται ο γεωμετρικός μέσος των τιμών SUCI για workloads κατηγορίας CT-F κατά τις πολιτικές UM, CT και τους μηχανισμούς DICER, DICER-TP. Ακόμη, στον παρακάτω πίνακα παρουσιάζεται το ποσοστό workloads κατηγορίας CT-F, για τα οποία επιτυγχάνονται διαφορετικές τιμές SLO. Για λ=0,5 , παρατηρούμε πως για χαμηλές τιμές SLOs, ο μηχανισμός DICER-TP ξεπερνάει την CT πολιτική καθώς αυξάνει την επίδοση των BE εφαρμογών, ενώ για μεγαλύτερες τιμές, είναι λιγότερο αποτελεσματικός καθώς αποτυγχάνει να διασφαλίσει τα SLOs. Για λ=1 και λ=2 και όλες τις διαφορετικές τιμές SLOs, ο μηχανισμός DICER-TP είναι αποτελεσματικότερος συγκριτικά με τις δυο πολιτικές, καθώς και τον μηχανισμό DICER, αφού καταφέρνει να προστατέψει την επίδοση της HP εφαρμογής, ενώ παράλληλα αυξάνει την χρησιμοποίηση του συστήματος.

	Ποσοστό workloads που επιτεύχθηκε το SLO			
SLO (%)	**80**	**85**	**90**	**95**
UM	39,68	25,4	12,17	2,91
CT	96,03	91,8	78,84	53,7
DICER	67,72	52,91	35,98	15,87
DICER-TP	92,96	80,42	58,99	31,75

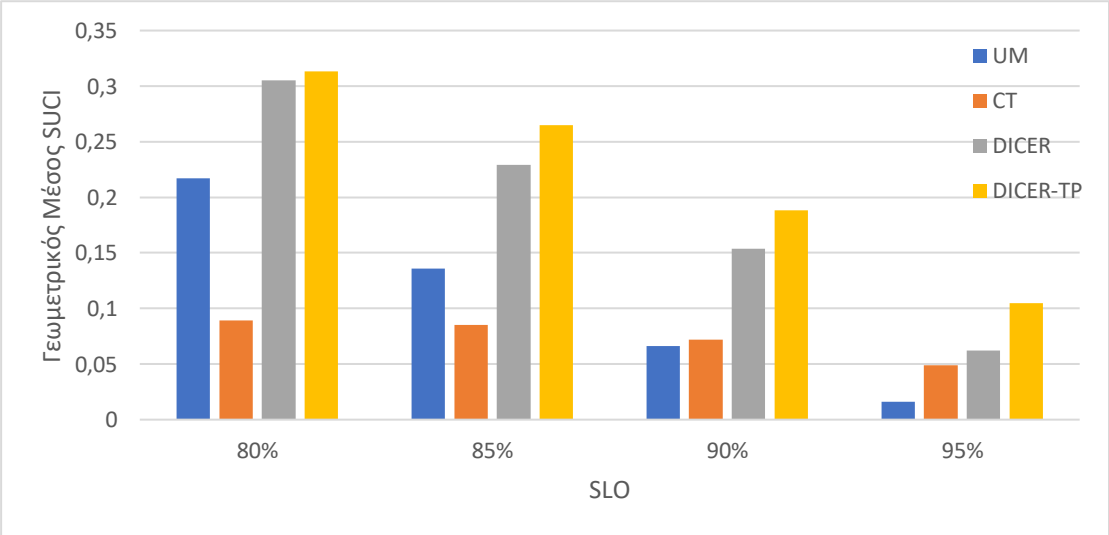
Πίνακας 5.2 : Ποσοστό workloads κατηγορίας CT-F που επιτυγχάνονται τα SLOs με χρήση UM, CT, DICER και DICER-TP



Σχήμα 5.11 : Γεωμετρικός μέσος τιμών SUCI workloads κατηγορίας CT-F για λ=0.5



Σχήμα 5.12 : Γεωμετρικός μέσος τιμών SUCI workloads κατηγορίας CT-F για λ=1

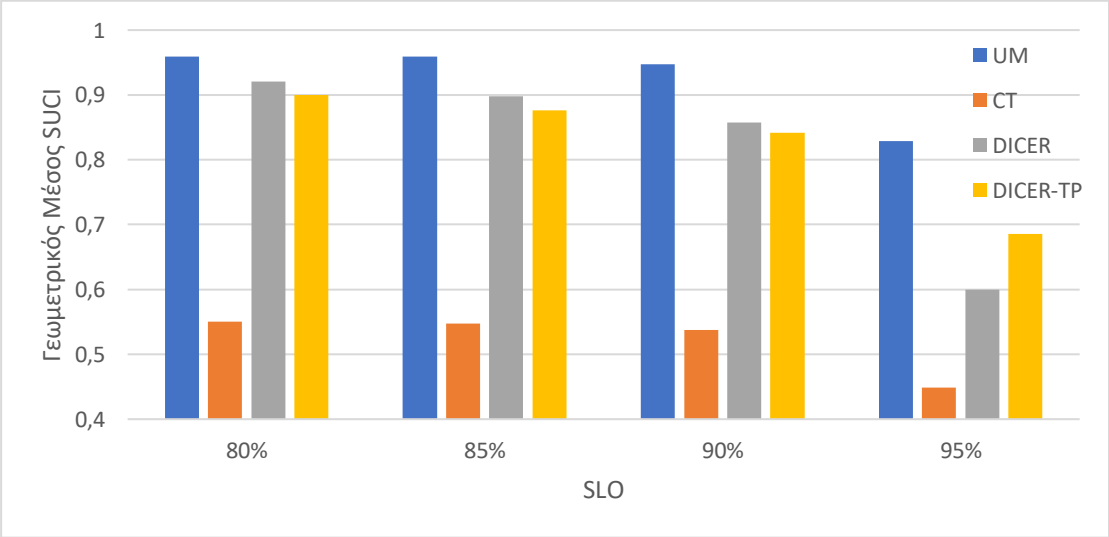


Σχήμα 5.13 : Γεωμετρικός μέσος τιμών SUCI workloads κατηγορίας CT-F για λ=2

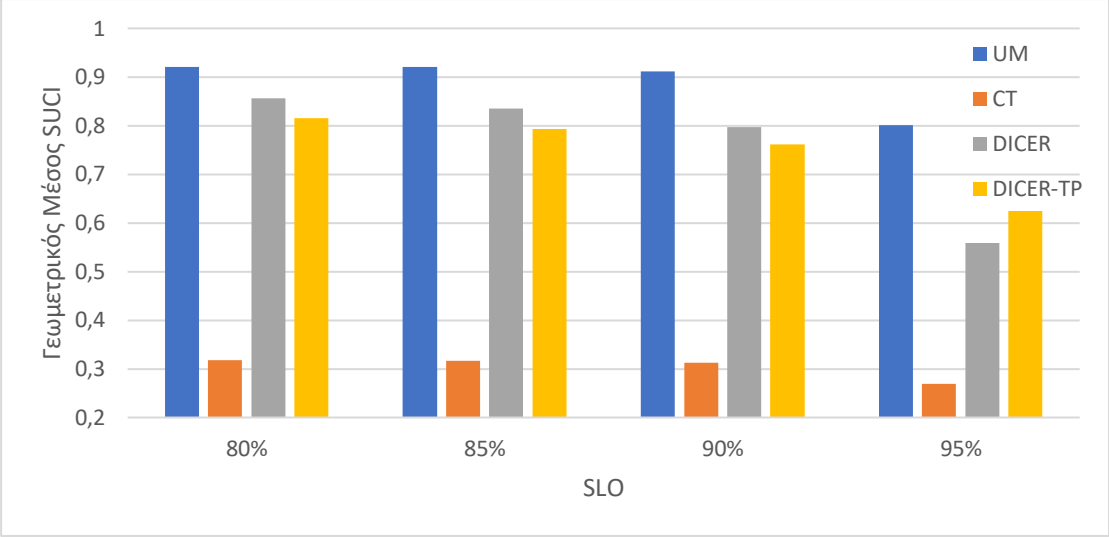
Στην συνέχεια, παρουσιάζονται ο γεωμετρικός μέσος των τιμών SUCI για workloads κατηγορίας CT-F κατά τις πολιτικές UM, CT και τους μηχανισμούς DICER, DICER-TP. Ακόμη, στον παρακάτω πίνακα παρουσιάζεται το ποσοστό των workloads κατηγορίας CT-T, για τα οποία επιτυγχάνονται διαφορετικές τιμές SLO. Παρατηρούμε ότι η UM πολιτική είναι αποτελεσματικότερη για κάθε τιμή λ και SLO συγκριτικά με την πολιτική CT και τους δύο μηχανισμούς, καθώς επιτυγχάνει να διασφαλίσει την επίδοση της HP εφαρμογής, ενώ πετυχαίνει την καλύτερη χρησιμοποίηση του συστήματος, αφού δεν γίνεται κάποιος διαμοιρασμός των κοινόχρηστων πόρων. Ακόμη, συγκρίνοντας τους δύο μηχανισμούς, διαπιστώνουμε πως ο μηχανισμός DICER-TP ξεπερνά τον DICER μόνο για SLO=95%, καθώς επιτυγχάνει αποτελεσματικότερα την διασφάλιση του συγκεκριμένου SLO. Τέλος, η πολιτική CT πετυχαίνει πολύ χαμηλές τιμές SUCI, διότι όπως έχουμε περιγράψει στην ενότητα 3.1, καταστρέφει επί της ουσίας την επίδοση των BE εφαρμογών.

	Ποσοστό workloads που επιτεύχθηκε το SLO			
SLO (%)	**80**	**85**	**90**	**95**
UM	100	100	99,08	89,4
CT	99,54	99,08	97,7	83,41
DICER	99,54	97,7	94,47	72,81
DICER-TP	100	98,62	95,39	81,57

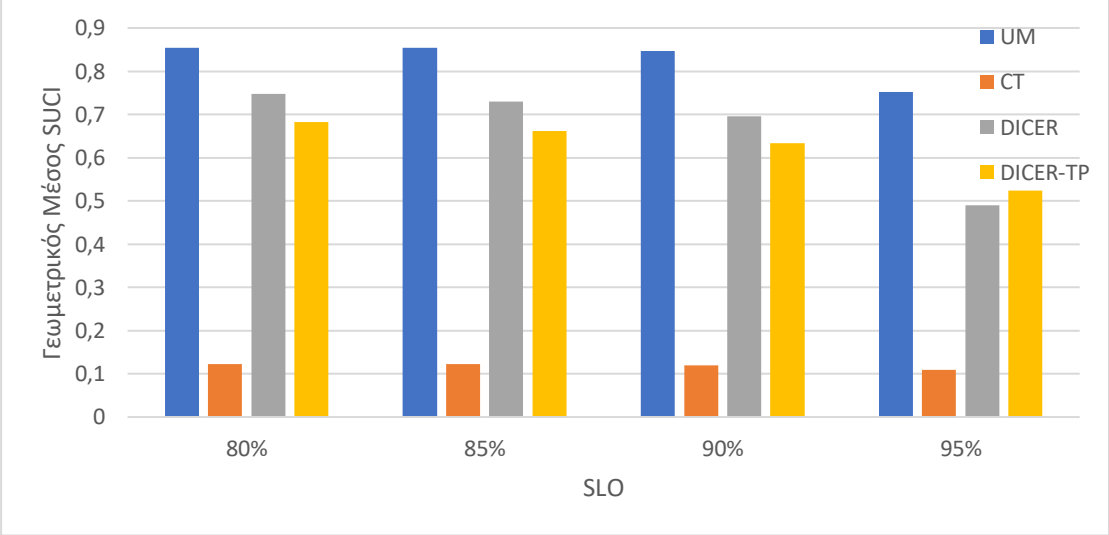
Πίνακας 5.3 : Ποσοστό workloads κατηγορίας CT-T που επιτυγχάνονται τα SLOs με χρήση UM, CT, DICER και DICER-TP



Σχήμα 5.14 : Γεωμετρικός μέσος τιμών SUCI workloads κατηγορίας CT-T για λ=0.5



Σχήμα 5.15 : Γεωμετρικός μέσος τιμών SUCI workloads κατηγορίας CT-T για λ=1



Σχήμα 5.16 : Γεωμετρικός μέσος τιμών SUCI workloads κατηγορίας CT-T για λ=2

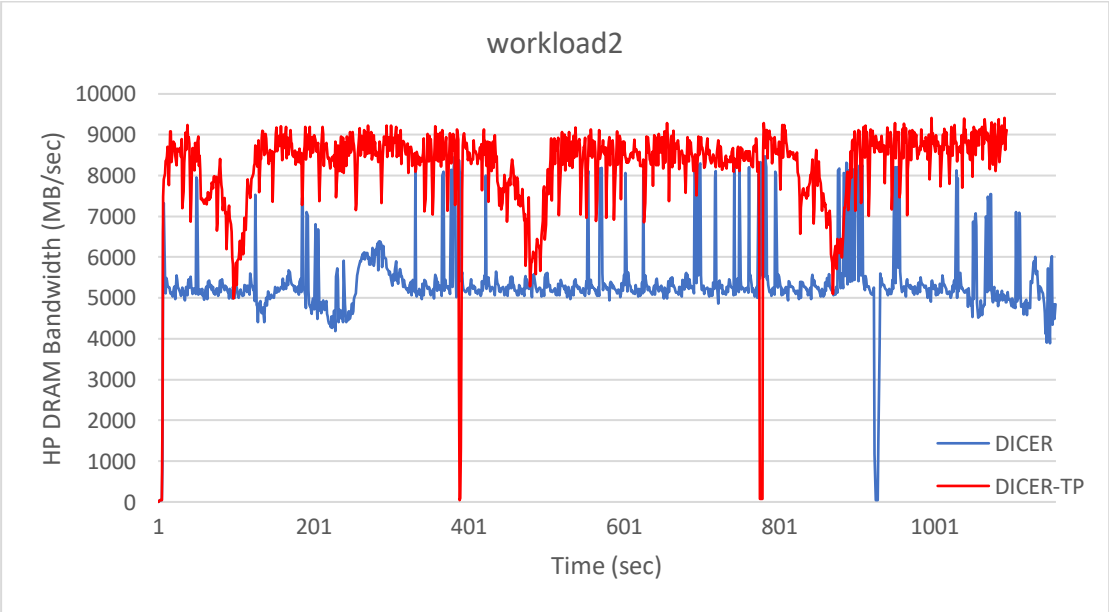
Παρακάτω παραθέτουμε τα στοιχεία της εκτέλεσης του workload2, το οποίο ανήκει στην κατηγορία CT-F και παρουσιάστηκε στα προηγούμενα κεφάλαια. Στον πίνακα 5.4, παρουσιάζονται αναλυτικά τα στοιχεία συνεκτέλεσης, κανονικοποιημένα ως προς αυτά της απομονωμένης εκτέλεσης των benchmarks, εκτός από την τιμή του μέσου συνολικού DRAM bandwidth. Παρατηρούμε πως ο μηχανισμός DICER-TP αυξάνει κατά περίπου 50% την επίδοση της HP εφαρμογής, συγκριτικά με τον μηχανισμό DICER, υπό το κόστος της αναμενόμενης μείωσης της επίδοσης των BE εφαρμογών. Παρατηρούμε, ακόμη, πως με την χρήση του Thread Packing, ο μηχανισμός DICER-TP επιτρέπει στην HP εφαρμογή να κρατήσει μεγαλύτερο μέρος της LLC, για την προστασία της επίδοσης της, σε αντίθεση με τον μηχανισμό DICER. Ακόμη,

παρατηρούμε πως ο μηχανισμός DICER-TP επιτρέπει στην HP εφαρμογή να λάβει κατά 20% περισσότερο DRAM bandwidth συγκριτικά με τον μηχανισμό DICER, αφού μειώνει αποτελεσματικά την κατανάλωση αυτού από τις BE εφαρμογές, όπως παρουσιάζεται και στο σχήμα 5.17.

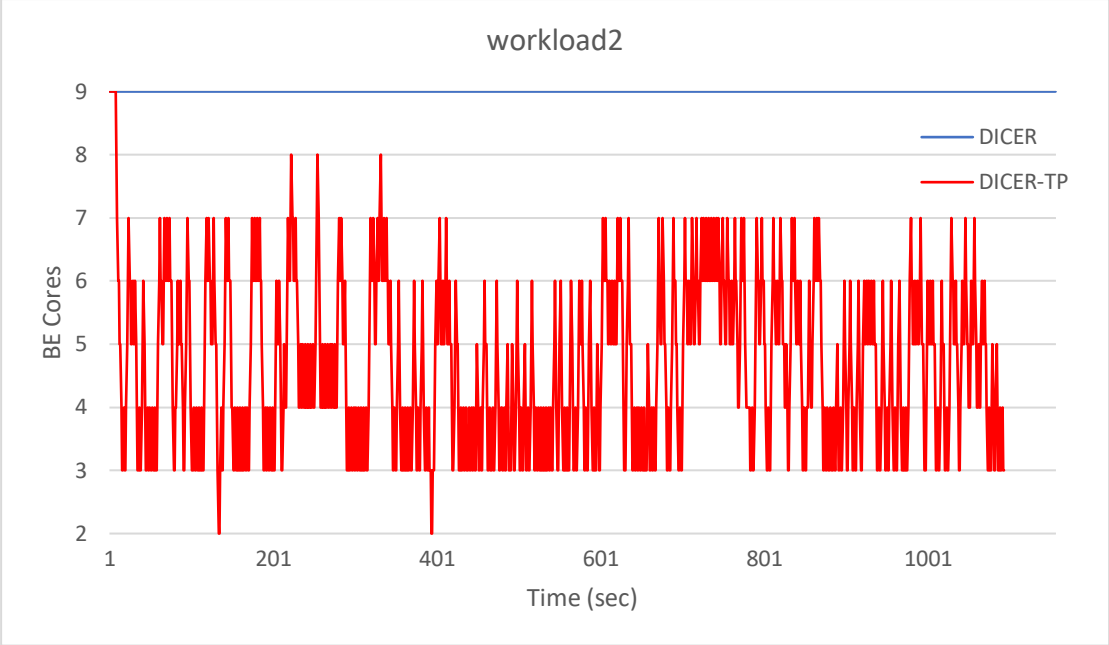
workload2	DICER	DICER-TP
HP IPC	0,572	0,851
Avg. BE IPC	0,464	0,31
Avg. HP LLC Occupancy (%)	57,48	71,08
Avg. HP Bandwidth (%)	57,11	86,61
Avg. Total Bandwidth (MB/sec)	50064,12	36295,56

Πίνακας 5.4 : Ληφθέντες μετρήσεις από την εκτέλεση του workload2 με DICER και DICER-TP

Η λειτουργία του μηχανισμού DICER-TP επιβεβαιώνεται και από το συνολικό DRAM bandwidth, αφού καταφέρνει να κρατήσει το μέσο συνολικό DRAM bandwidth κάτω από το κατώφλι των 37,5 GB/sec, σε αντίθεση με τον μηχανισμό DICER, κατά τον οποίο το μέσο συνολικό DRAM bandwidth ξεπερνά το κατώφλι των 50 GB/sec. Τέλος, στο σχήμα 5.18 παρουσιάζονται οι αποφάσεις του μηχανισμού DICER-TP, σχετικά με την μείωση των διαθέσιμων πυρήνων για την εκτέλεση των BE εφαρμογών. Παρατηρούμε πως ο μηχανισμός DICER-TP μειώνει σταδιακά τους διαθέσιμους πυρήνες, προκειμένου να καταπολεμήσει τον ανταγωνισμό για το DRAM bandwidth και στην συνέχεια προσπαθεί να τους αυξήσει για να βελτιώσει την επίδοση των BE εφαρμογών.



Σχήμα 5.17 : Χρήση DRAM Bandwidth HP εφαρμογής με DICER και DICER-TP κατά την εκτέλεση του workload2

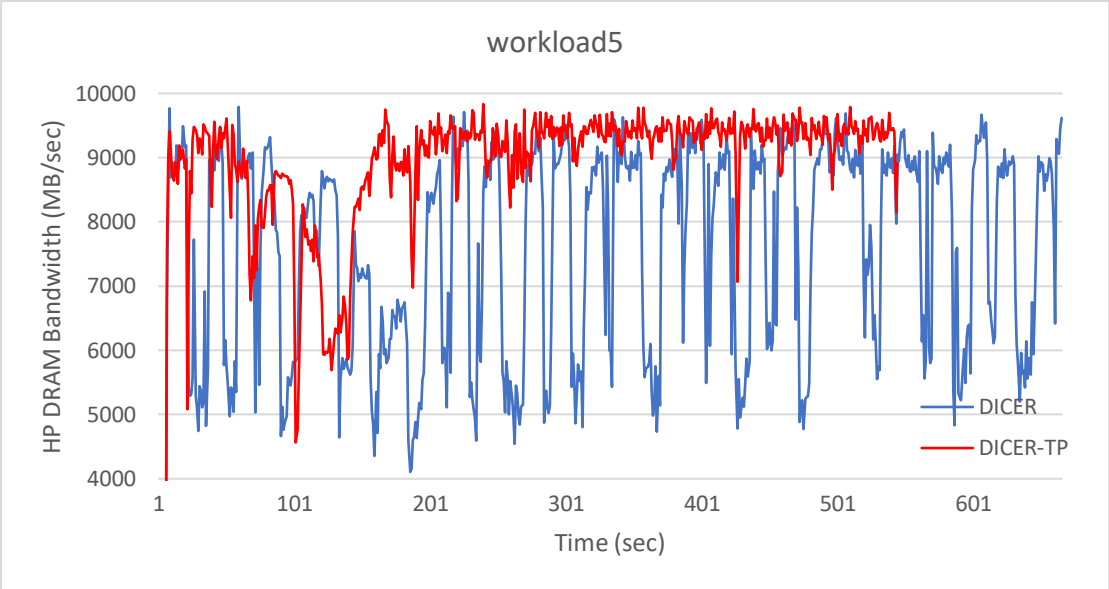


Σχήμα 5.18 : Διαθέσιμοι πυρήνες για την εκτέλεση των BE εφαρμογών με DICER και DICER-TP κατά την εκτέλεση του workload2

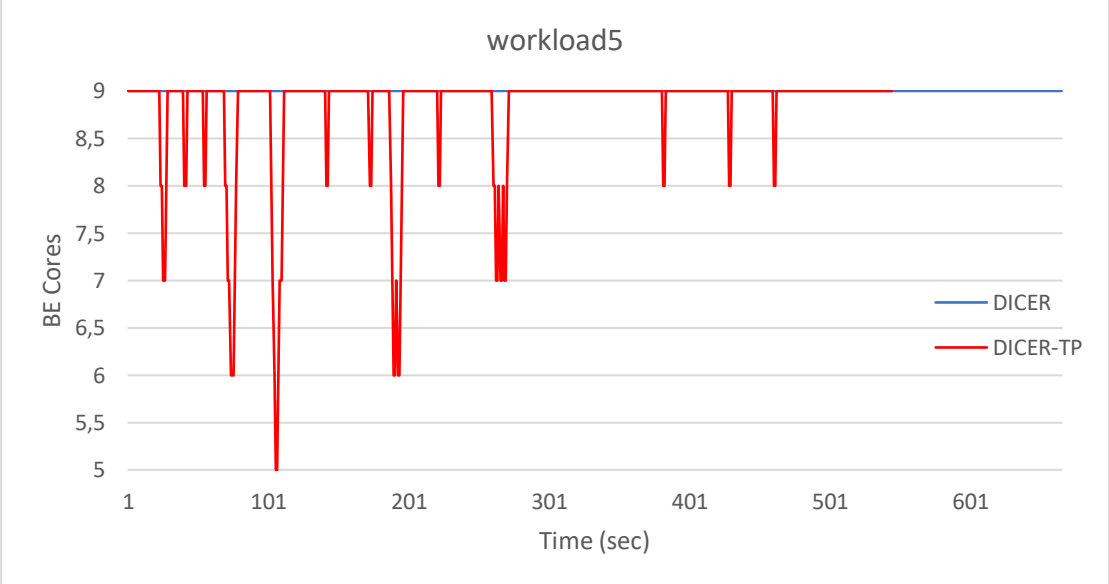
Στην συνέχεια, παρουσιάζεται η συνεκτέλεση workload5, το οποίο ανήκει στην κατηγορία CT-T και αναλύθηκε στα προηγούμενα κεφάλαια. Στον πίνακα 5.5, παρουσιάζονται τα στοιχεία της συνεκτέλεσης κανονικοποιημένα ως προς αυτά της απομονωμένης εκτέλεσης των benchmarks, εκτός από την τιμή του μέσου συνολικού DRAM bandwidth. Παρατηρούμε πως ο μηχανισμός DICER-TP αυξάνει την επίδοση της HP εφαρμογής κατά περίπου 8% συγκριτικά με τον μηχανισμό DICER. Όμως, παρατηρούμε ακόμη πως αυξάνεται και η επίδοση των BE εφαρμογών, κάτι που αρχικά δεν ήταν αναμενόμενο. Παρατηρώντας πιο προσεκτικά τον μέσο χώρο της LLC που αποδίδεται στην HP εφαρμογή, παρατηρούμε ότι οι BE εφαρμογές λαμβάνουν τρεις φορές περισσότερο χώρο με τον μηχανισμό DICER-TP. Ακόμη, από το σχήμα 5.20 παρατηρούμε πως συγκριτικά με το προηγούμενο workload που εξετάσαμε, οι διαθέσιμοι πυρήνες προς τις BE εφαρμογές μειώνονται πολύ λιγότερο. Οι δύο παραπάνω παρατηρήσεις εξηγούν γιατί παρατηρούμε αύξηση και στην επίδοση των BE εφαρμογών. Τέλος, από το σχήμα 5.19, βλέπουμε ότι ο μηχανισμός DICER-TP επιτρέπει στην HP εφαρμογή να κρατήσει το DRAM bandwidth στο επίπεδο που εκείνη χρειάζεται, σε αντίθεση με τον μηχανισμό DICER, κατά τον οποίο παρατηρούνται μεγάλες διακυμάνσεις στην τιμή του.

workload5	DICER	DICER-TP
HP IPC	0,782	0,863
Avg. BE IPC	0,546	0,644
Avg. HP LLC Occupancy (%)	42,35	14
Avg. HP Bandwidth (%)	78,69	93,13
Avg. Total Bandwidth (MB/sec)	36695,61	28693,78

Πίνακας 5.5 : Ληφθέντες μετρήσεις από την εκτέλεση του workload5 με DICER και DICER-TP

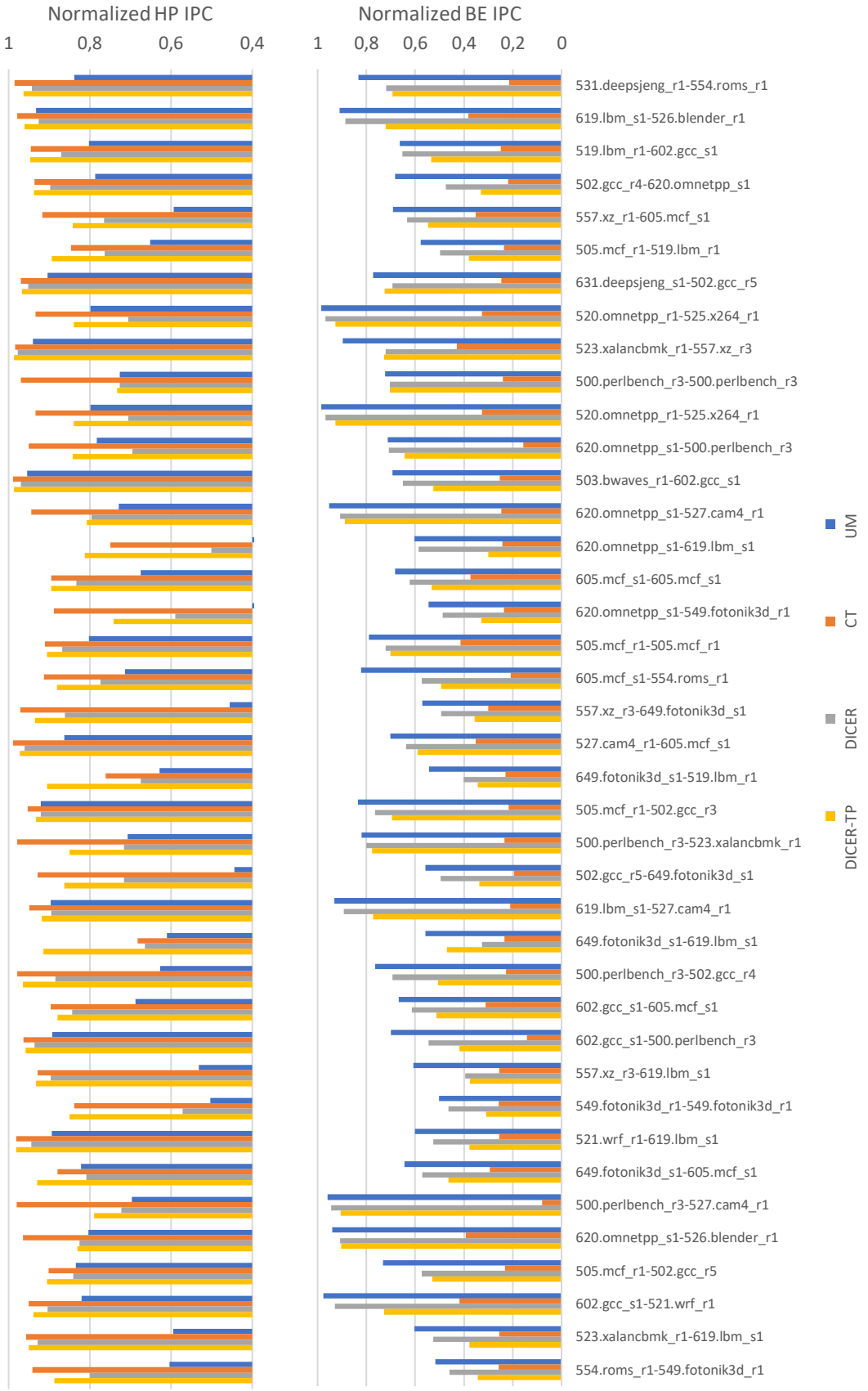


Σχήμα 5.19 : Χρήση DRAM Bandwidth HP εφαρμογής με DICER και DICER-TP κατά την εκτέλεση του workload5

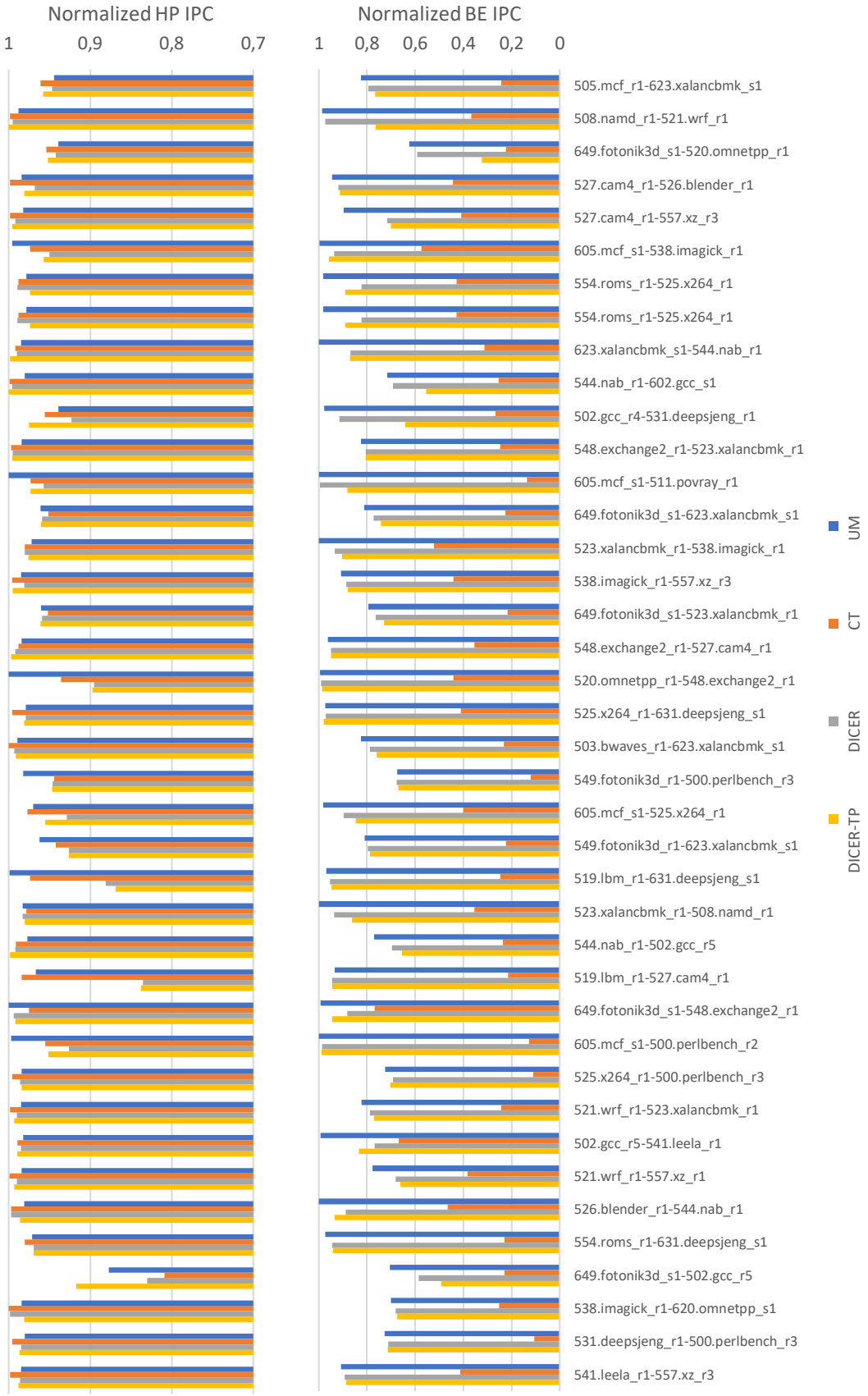


Σχήμα 5.20 Διαθέσιμοι πυρήνες για την εκτέλεση των BE εφαρμογών με DICER και DICER-TP κατά την εκτέλεση του workload5

Τέλος, παραθέτουμε αναλυτικά τις μετρήσεις του IPC της HP εφαρμογής και των BE εφαρμογών για 40 τυχαία επιλεγμένα workloads κατηγορίας CT-F και 40 τυχαία επιλεγμένα workloads κατηγορίας CT-T, όπως αυτές προέκυψαν κατά τις πολιτικές UM και CT και του μηχανισμούς .DICER και DICER-TP. Όλα τα IPC είναι κανονικοποιημένα ως προς το IPC της απομονωμένης εκτέλεσης της κάθε εφαρμογής.



Σχήμα 5.21 : Κανονικοποιημένο IPC HP και BE εφαρμογών κατηγορίας CT-F ως προς την απομονωμένη εκτέλεση τους



Σχήμα 5.22 : Κανονικοποιημένο IPC HP και BE εφαρμογών κατηγορίας CT-T ως προς την απομονωμένη εκτέλεσή τους

Κεφάλαιο 6

Επίλογος

6.1 Σύνοψη και Συμπεράσματα

Στην παρούσα διπλωματική εργασία, στόχος μας ήταν η επέκταση του υπάρχοντος μηχανισμού DICER, ώστε να αντιμετωπίζεται αποτελεσματικότερα ο ανταγωνισμός για DRAM bandwidth μεταξύ των συνεκτελούμενων εφαρμογών. Για την επίτευξη αυτού του στόχου, αξιοποιήσαμε τις τεχνολογίες που έχει αναπτύξει η Intel, οι οποίες μας επιτρέπουν την παρακολούθηση της επίδοσης των εφαρμογών και τον δυναμικό καταμερισμό της μνήμης LLC.

Συνεπώς, δημιουργήσαμε το μηχανισμό DICER-TP, ο οποίος κάνει χρήση της τεχνικής Thread Packing, προκειμένου να μειώσει το DRAM bandwidth που καταναλώνεται από εφαρμογές χαμηλής προτεραιότητας, ώστε να προστατευθεί η επίδοση της υψηλής προτεραιότητας εφαρμογής. Ο μηχανισμός που αναπτύξαμε, μειώνει τους διαθέσιμους πυρήνες, στους οποίους εκτελούνται οι χαμηλής προτεραιότητας εφαρμογές, όταν αναγνωριστεί κορεσμός στο DRAM bandwidth. Έπειτα, όταν ο κορεσμός έχει αντιμετωπιστεί, ο μηχανισμός σταδιακά αυξάνει τους διαθέσιμους πυρήνες, ώστε να αυξήσει την συνολική χρησιμοποίηση του συστήματος.

Για την αξιολόγηση του μηχανισμού DICER-TP, χρησιμοποιήσαμε την σουίτα μετροπρογραμμάτων SPEC CPU2017. Όπως παρατηρήσαμε σε προηγούμενο κεφάλαιο, ο μηχανισμός DICER-TP πετυχαίνει καλύτερα αποτελέσματα SUCI για τα περισσότερα SLO και λ, συγκριτικά με τις πολιτικές UM και CT, και τον μηχανισμό DICER.

6.2 Μελλοντικές Κατευθύνσεις

Η παρούσα εργασία θα μπορούσε να επεκταθεί προς αρκετές κατευθύνσεις. Αρχικά, θα ήταν ιδιαιτέρως χρήσιμο να εξεταστεί η τεχνολογία MBA σε κάποιον server, στον οποίο αυτή η τεχνολογία εκτελείται απροβλημάτιστα. Η αξιοποίηση της τεχνολογίας MBA θα μπορούσε να επιτύχει παρόμοια αποτελέσματα με αυτά που είδαμε στην παρούσα εργασία σε ότι αφορά την προστασία της επίδοσης της υψηλής προτεραιότητας εφαρμογής, όμως θα μπορούσε να αυξήσει την συνολική χρησιμοποίηση του συστήματος, μειώνοντας λιγότερο την επίδοση των εφαρμογών χαμηλής προτεραιότητας. Όπως προαναφέραμε, η χρήση του Thread Packing, συνεπάγεται ότι ορισμένες εφαρμογές δεν εκτελούνται για κάποια κβάντα χρόνου, μειώνοντας αρκετά την επίδοση τους. Συνεπώς, με την χρήση της τεχνολογίας MBA, θα ήταν δυνατό όλες οι εφαρμογές να εκτελούνται ταυτόχρονα, περιορίζοντας την διαθεσιμότητα του DRAM bandwidth προς αυτές.

Στην συνέχεια, μία ακόμη επέκταση του μηχανισμού θα μπορούσε να είναι η προστασία της επίδοσης δύο εφαρμογών υψηλής προτεραιότητας. Συνεπώς, ο μηχανισμός θα πρέπει να είναι σε θέση να κατανείμει κατάλληλα τους διαθέσιμους πόρους μεταξύ των εφαρμογών υψηλής προτεραιότητας και των εφαρμογών χαμηλής προτεραιότητας. Ακόμη, με παρόμοια λογική, θα μπορούσαμε να δημιουργήσουμε ένα τρίτο επίπεδο στην ιεραρχία προτεραιότητας, τις εφαρμογές μεσαίας προτεραιότητας. Σε αυτή την περίπτωση, αρχικός στόχος του μηχανισμού θα είναι η εξασφάλιση παροχής ποιότητας στις εφαρμογές υψηλής και μεσαίας προτεραιότητας και έπειτα η βελτίωση της συνολικής χρησιμοποίησης του συστήματος.

Τέλος, ιδιαίτερο ενδιαφέρον θα είχε η αντικατάσταση των μονονηματικών (single-threaded) εφαρμογών στο σενάριο συνεκτέλεσης, από μία πολυνηματική (multi-threaded) εφαρμογή, κατά την εκτέλεση της οποίας, διαφορετικά νήματα (threads) παρουσιάζουν διαφορετική προτεραιότητα ως προς την εκτέλεσή τους. Συνεπώς, ο μηχανισμός θα πρέπει να κατανείμει τους διαθέσιμους πόρους κατάλληλα ώστε να βελτιωθεί η επίδοση της πολυνηματικής εφαρμογής.

Βιβλιογραφία

1. Chandra D, Guo F, Kim S, Solihin Y. Predicting inter-thread cache contention on a chip multi-processor architecture. In: Proceedings - International Symposium on High-Performance Computer Architecture [Internet]. 2005. p. 340–51. Available from: 10.1109/HPCA.2005.27

2. Iyer R. CQoS: A framework for enabling QoS in shared caches of CMP platforms. Proc Int Conf Supercomput. 2004;257–66.

3. Kim S, Chandra D, Solihin Y. Fair cache sharing and partitioning in a chip multiprocessor architecture [Internet]. Parallel Architectures and Compilation Techniques - Conference Proceedings, PACT. 2004. p. 111–22. Available from: 10.1109/pact.2004.1342546

4. Papadakis I, Nikas K, Karakostas V, Goumas G, Koziris N. Improving QoS and Utilisation in modern multi-core servers with Dynamic Cache Partitioning. In: 2nd Workshop on Co-Scheduling of HPC Applications [Internet]. 2017. Available from: 10.14459/2017md1344298

5. Lo D, Cheng L, Govindaraju R, Ranganathan P, Kozyrakis C. Heracles: Improving resource efficiency at scale. In: Proceedings - International Symposium on Computer Architecture. New York, New York, USA: Institute of Electrical and Electronics Engineers Inc.; 2015. p. 450–62. Available from: 10.1145/2749469.2749475

6. Qureshi MK, Patt YN. Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches. In: Proceedings of the Annual International Symposium on Microarchitecture, MICRO [Internet]. 2006. p. 423–32. Available from: 10.1109/MICRO.2006.49

7. Mars J, Tang L, Hundt R, Skadron K, Soffa M Lou. Bubble-Up: Increasing utilization in modern warehouse scale computers via sensible co-locations [Internet]. Proceedings of the Annual International Symposium on Microarchitecture, MICRO. 2011. p. 248–59. Available from: 10.1145/2155620.2155650

8. Fedorova A, Seltzer M, Smith MD. Improving Performance Isolation on Chip Multiprocessors via an Operating System Scheduler. In: Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques (PACT 2007) [Internet]. Institute of Electrical and Electronics Engineers (IEEE); 2007. p. 25–38. Available from: 10.1109/pact.2007.4336197

9. Herdrich A, Illikkal R, Iyer R, Newell D, Chadha V, Moses J. Rate-based QoS techniques for cache/memory in CMP platforms. In: Proceedings of the International Conference on Supercomputing. New York, New York, USA: ACM Press; 2009. p. 479–88. Available from: 10.1145/1542275.1542342

10. Cook H, Moreto M, Bird S, Dao K, Patterson DA, Asanovic K. A hardware evaluation of cache partitioning to improve utilization and energy-efficiency while preserving responsiveness. In: Proceedings - International Symposium on Computer Architecture. New York, New York, USA: ACM Press; 2013. p. 308–19. Available from: 10.1145/2485922.2485949

11. Nikas K, Papadopoulou N, Giantsidi D, Karakostas V, Goumas G, Koziris N. DICER: Diligent cache partitioning for efficient workload consolidation. In: ICPP: Proceedings of the 48th International Conference on Parallel Processing. 2019. p. 1–10. Available from: 10.1145/3337821.3337891

12. Park J, Park S, Han M, Hyun J, Baek W. HyPart: A hybrid technique for practical memory bandwidth partitioning on commodity servers. In: Parallel Architectures and Compilation Techniques - Conference Proceedings, PACT. New York, NY, USA: Institute of Electrical and Electronics Engineers Inc.; 2018. p. 1–14. Available from: 10.1145/3243176.3243211

13. Herdrich A, Verplanke E, Autee P, Illikkal R, Gianos C, Singhal R, et al. Cache QoS: From concept to reality in the Intel® Xeon® processor E5-2600 v3 product family. In: Proceedings - International Symposium on High-Performance Computer Architecture [Internet]. IEEE Computer Society; 2016. p. 657–68. Available from: 10.1109/HPCA.2016.7446102

14. Taylor G, Davies P, Farmwald M. The TLB slice--A low-cost high-speed address translation mechanism. In: Conference Proceedings - Annual Symposium on Computer Architecture [Internet]. Publ by IEEE; 1990. p. 355–63. Available from: 10.1145/325164.325161

15. Bray BK, Lynch WL, Flynn MJ, Bray BK, Lynch WL, Flynn MJ. Page allocation to reduce access time of physical caches [Internet]. Stanford, CA, United States. 1990.

16. Bugnion E, Anderson JM, Mowry TC, Rosenblum M, Lam MS. Compiler-directed page coloring for multiprocessors. Oper Syst Rev [Internet]. 1996;30(5):244–55. Available from: 10.1145/248208.237195

17. Cochran R, Hankendi C, Coskun AK, Reda S. Pack & Cap: Adaptive DVFS and thread packing under power caps [Internet]. Proceedings of the Annual International Symposium on Microarchitecture, MICRO. 2011. p. 175–85. Available from: 10.1145/2155620.2155641

18. Standard Performance Evaluation Corporation. SPEC CPU 2017 [Internet]. 2019.

19. Navarro-Torres A, Alastruey-Benedé J, Ibáñez-Marín P, Viñals-Yúfera V. Memory hierarchy characterization of SPEC CPU2006 and SPEC CPU2017 on the Intel Xeon Skylake-SP. Mehmood R, editor. PLoS One. 2019;14(8):e0220135. Available from: 10.1371/journal.pone.0220135