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

Αξιοποίηση των τεχνολογιών Intel RDT για τη Βελτίωση της Ποιότητας Υπηρεσίας (QoS) σε Πολυεπεξεργαστικά Συστήματα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΔΗΜΗΤΡΑ ΓΙΑΝΤΣΙΔΗ

Επιβλέπων : Γεώργιος Γκούμας
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2018

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

Αξιοποίηση των τεχνολογιών Intel RDT για τη Βελτίωση της Ποιότητας Υπηρεσίας (QoS) σε Πολυεπεξεργαστικά Συστήματα

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΔΗΜΗΤΡΑ ΓΙΑΝΤΣΙΔΗ

Επιβλέπων : Γεώργιος Γκούμας
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 14η Μαρτίου 2018.

.....	.....	.....
Γεώργιος Γκούμας Επίκουρος Καθηγητής Ε.Μ.Π.	Νεκτάριος Κοζύρης Καθηγητής Ε.Μ.Π.	Δημήτριος Σούντρης Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2018

Δήμητρα Γιαντσίδη

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

Περίληψη

Η παροχή ποιότητας υπηρεσίας στα σύγχρονα κέντρα δεδομένων είναι νευραλγικής σημασίας. Μέχρι τώρα, οι πάροχοι υπηρεσιών Υπολογιστικού Νέφους (Cloud Services providers) εξασφαλίζουν και τηρούν τα SLAs (Service-Level-Argeements) των πελατών παρέχοντας απομονωμένη εκτέλεση των επιλεγμένων υπηρεσιών στους επεξεργαστές, δηλαδή απαγορεύοντας την συνεκτέλεση των συγκεκριμένων υπηρεσιών με άλλες. Με άλλα λόγια, τα επεξεργαστικά συστήματα στα κέντρα δεδομένων υποχρησιμοποιούνται σε σημαντικό βαθμό, λιγότερο από 50%, προκειμένου να εξασφαλιστεί η ζητούμενη ποιότητα υπηρεσίας.

Η ανάγκη για απομόνωση της εκτέλεσης μιας υπηρεσίας πηγάζει από τον ανταγωνισμό των εφαρμογών, κατά τη συνεκτέλεση, για τους κοινόχρηστους πόρους του συστήματος, όπως για παράδειγμα είναι η τελευταίου επιπέδου κρυφή μνήμη (Last-Level-Cache). Αυτός ο ανταγωνισμός βλάπτει την βέλτιστη επίδοση και κατ' επέκταση την ποιότητα υπηρεσίας των εφαρμογών. Η διαχείριση αυτού του επιπέδου μνήμης, δηλαδή η επίβλεψη χρήσης και ο διαμοιρασμός τμημάτων της στις εφαρμογές του συστήματος έγινε πρόσφατα εφικτή μέσω των τεχνολογιών Cache-Monitoring και Cache-Allocation που ανέπτυξε η Intel.

Στην παρούσα διπλωματική εργασία μελετάμε και αξιοποιούμε τις παραπάνω τεχνολογίες για την υλοποίηση ενός δυναμικού μηχανισμού με ανάδραση, ο οποίος στοχεύει στην βελτίωση της ποιότητας υπηρεσίας μιας εφαρμογής με υψηλή προτεραιότητα και ταυτόχρονα επιδιώκει την μείωση της χρονικής επιβράδυνσης εφαρμογών με λιγότερη προτεραιότητα. Στόχος μας είναι να προστατεύσουμε την εκτέλεση της υψηλής προτεραιότητας εφαρμογής στην περίπτωση που το σύστημα χρησιμοποιείται πλήρως, δηλαδή για N-το-πλήθος πυρήνες κάθε χρονική στιγμή είναι χρονοδρολογημένα N-το-πλήθος νήματα.

Για την αξιολόγηση του μηχανισμού εκτελούμε συνεκτελέσεις μονονηματικών εφαρμογών από τις σουίτες PARSEC και SPEC2006 στον επεξεργαστή Xeon E5-2630 v4 που υποστηρίζει τις προαναφερθείσες τεχνολογίες. Τέλος, αποδεικνύουμε ότι η χρήση του μηχανισμού αυτού ενδύκνειται ταυτόχρονα σε συνεκτελέσεις με μεγάλη ευαισθησία της εφαρμογής με υψηλή προτεραιότητα ως προς την κρυφή μνήμη αλλά και σε συνεκτελέσεις οι οποίες προκαλούν κορεσμό στο εύρος του διαύλου δεδομένων προς τη μνήμη.

Λέξεις κλειδιά

διαμοιρασμός κοινόχρηστης κρυφής μνήμης, καταμερισμός κοινόχρησης κρυφής μνήμης, ποιότητα υπηρεσίας, τεχνολογία Intel Cache-Allocation, τεχνολογία Intel Cache-Monitoring, δυναμικός μηχανισμός με ανάδραση, πλήρης χρησιμοποίηση επεξεργαστή, κοινόχρηστοι πόροι επεξεργαστή

Abstract

Safeguarding Quality of Service in modern data centers is of great importance. Until now, Cloud Services' providers ensured and satisfied customers' SLAs by providing a fully-isolated execution of the selected applications in multi-processors, namely, forbiding these applications to co-execute with others. In other words, computing systems in data centers are underutilized in a significant degree, less than 50%, so as the required Quality of Service to be ensured.

The need for isolating the execution of an application is a result of the applications' contention, during co-execution, for all shared resources, such as the last level shared cache memory. This contention harms the optimal performance and as consequence, the Quality of Service. The management of this memory hierarchy level, namely, the inspection of use and the cache partitioning among applications is lately feasible via Cache-Monitoring technology as well as Cache-Allocation technology implemented by Intel.

In this Diploma Thesis we study and exploit the above technologies for the implementation of a dynamic mechanism with feedback, which aims to improve Quality of Service of a high priority apllication, and, at the same time, tries to reduce the time latency of other applications with lower priority. Our scope is to protect the execution of the application with high priority in cases of a fully utilised system, that is, for N cores every sigle time, N software theads are scheduled.

For the evaluation of the mechanish we co-execute single-threaded applications of suites PARSEC and SPEC2006 in Intel Xeon E5-2630 v4 processor which supports the aforementioned technologies. Finally, we prove that our mechanism is suitable in co-executions with a cache-sensitive high priority appication as well as in co-executions which causes memory bandwidth saturation.

Key words

common cache sharing, shared cache partitioning, Quality of Service, Intel Cache-Allocation Technology, Intel Cache-Monitoring Technology, dynamic mechanism with feedback, full utilization of a multicore processor, common shared resources

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε στο Εργαστήριο Υπολογιστικών Συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου από τον Ιανουάριο του 2017 έως τον Μάρτιο του 2018.

Καταρχάς, θα ήθελα να εκφράσω τις θερμότερες ευχαριστίες μου στον Επιβλέποντα Καθηγητή μου κ. Γεώργιο Γκούμα για την καθοδήγησή του στην επιλογή ενός ιδιαίτερα ενδιαφέροντος και σύγχρονου θέματος και για τη συμβολή του στην εκπόνηση της παρούσας διπλωματικής εργασίας. Θα ήθελα ιδιαιτέρως να ευχαριστήσω το Μεταδιδακτορικό Ερευνητή κ. Κωνσταντίνο Νίκα για τη καθοδήγησή του και την αδιάκοπη υποστήριξή του καθ' όλη τη διάρκεια της συνεργασίας μας αλλά και για το ενδιαφέρον και τις πολύτιμες συμβουλές του σχετικά με τα επόμενα ακαδημαϊκά μου βήματα. Οι γνώσεις, οι ιδέες του και η υπομονή του, ακόμα και στα πιο τετριμμένα ζητήματα, ήταν πολύτιμες για την ολοκλήρωση της εργασίας. Ευχαριστώ τον Μεταδιδακτορικό Ερευνητή Βασίλη Καρακώστα και τον Υποψήφιο Διδάκτορα Δημήτρη Σιακαβάρα για την βοήθεια που μου παρείχαν στα πλαίσια της συνεργασίας μας.

Ολοκληρώνοντας πλέον ένα σημαντικό κεφάλαιο της ζωής μου, χρωστώ ένα μεγάλο ευχαριστώ στους καθηγητές μου και κυρίως στους κ. Νεκτάριο Κοζύρη και κ. Γεώργιο Γκούμα για τις γνώσεις, την έμπνευση και το ενδιαφέρον που μου καλλιέργησαν μέσα από τις διαλέξεις τους για τον τομέα των Υπολογιστικών Συστημάτων κατά τη διάρκεια των σπουδών μου.

Ευχαριστώ τους αγαπημένους μου φίλους και ιδιαίτερα την Άντα, την Ελένη, την Ελπίδα, την Ευδοκία, τη Μαριάννα και το Φοίβο, για τα υπέροχα φοιτητικά χρόνια που περάσαμε μαζί. Τέλος, το μεγαλύτερο ευχαριστώ ανήκει στην οικογένεια μου για την απεριόριστη εμπιστοσύνη και την στήριξή τους όλον αυτόν τον καιρό.

Δήμητρα Γιαντσίδη,

Αθήνα, 14η Μαρτίου 2018

Περιεχόμενα

Κατάλογος πινάκων

Κατάλογος σχημάτων

Κεφάλαιο 1

Εισαγωγή

1.1 Σύγχρονα Πολυεπεξεργαστικά Συστήματα

Την τελευταία δεκαετία η χρήση πολυπύρηνων επεξεργαστών (Chip Multiprocessors, CMP) έχει πλέον εδραιωθεί σε επεξεργαστικά συστήματα για υψηλή επίδοση, όπως για παράδειγμα στους εξυπηρετητές (servers) των σύγχρονων κέντρων δεδομένων (Data Centers). Σε αντίθεση με τα παλαιότερα μονολιθικά συστήματα, τα οποία αποτελούνταν αποκλειστικά και μόνο από μία επεξεργαστική μονάδα και υλοποιούσαν τεχνικές παραλληλοποίησης μόνο σε επίπεδο εντολών (Instruction Level Parallelism), τα σύγχρονα πολυπύρηνα επεξεργαστικά συστήματα παρέχουν τη δυνατότητα παράλληλης εκτέλεσης πολλαπλών εφαρμογών και ταυτόχρονα εξασφαλίζουν βελτιωμένη απόδοση στο σύστημα. Ωστόσο, σε τέτοια συστήματα προκύπτει το πρόβλημα ανταγωνισμού για τους κοινούς πόρους ([1], [2], [3]), ο οποίος ανταγωνισμός συνεπάγεται σημαντικές επιπτώσεις στην παροχή ποιότητας υπηρεσίας (Quality of Service, QoS). Οι διαμοιραζόμενοι πόροι (shared resources), για τους οποίους ανταγωνίζονται οι συνεκτελούμενες εφαρμογές, περιλαμβάνουν την κοινόχρηστη κρυφή μνήμη (on-chip shared cache memory), το εύρος του διαύλου δεδομένων προς την μνήμη (memory bandwidth), ελεγκτές E/E (I/O controllers), κτλ.. Ο ανταγωνισμός, λοιπόν, για τους κοινούς αυτούς πόρους αποτελεί εμπόδιο στην επίτευξη ντετερμινιστικής και βέλτιστης εκτέλεσης των συνεκτελούμενων εφαρμογών, με αποτέλεσμα να παρατηρείται απρόβλεπτη εκτέλεση και κακή ποιότητα υπηρεσίας.

Ως αποτέλεσμα, η παροχή ποιότητας υπηρεσίας σε ένα περιβάλλον συνεκτελούμενων εφαρμογών αποτέλεσε, και συνεχίζει ακόμα και σήμερα, να αποτελεί σημείο ενδιαφέροντος πολλών ερευνητών ([4], [5], [6], [7], [8], [9], [10]). Απώτερος στόχος των σχετικών ερευνών είναι ο αποτελεσματικός διαμοιρασμός των κοινόχρηστων πόρων του πολυεπεξεργαστή στις εφαρμογές με στόχο την παροχή ποιότητας υπηρεσίας και την ντετερμινιστική τους εκτέλεση. Κατά κύριο λόγο, οι προαναφερθείσες έρευνες επικεντρώθηκαν στον βέλτιστο καταμερισμό των κοινόχρηστων μνημών cache, καθώς, η αποτελεσματική διαχείριση και εκμετάλλευσή τους είναι το κλειδί για την παροχή ποιότητας υπηρεσίας και τη βέλτιστη επίδοση των εφαρμογών.

1.2 Ποιότητα Υπηρεσίας και Υποχρησιμοποίηση των Επεξεργαστών

Η βελτίωση ποιότητας υπηρεσίας είναι πρωταρχικό ζήτημα σε όλα τα σύγχρονα πολυεπεξεργαστικά συστήματα. Είτε πρόκειται για ατομική και προσωπική χρήση υπολογιστικών συστημάτων, είτε για μεγάλες συστοιχίες υπολογιστών (clusters), κέντρα δεδομένων κτλ., οι χρήστες και οι

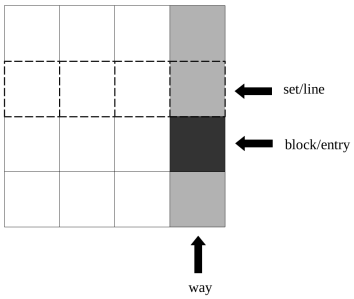
διαχειριστές εκτελούν ταυτόχρονα πολλές εφαρμογές, απαιτώντας την βέλτιστη και, ανεξάρτητη από τις άλλες, εκτέλεσή τους. Μάλιστα, η ταχεία ανάπτυξη των Cloud εφαρμογών, σε συνδυασμό με την τάση για εικονικοποίηση (virtualization), αυξάνει τις υπολογιστικές απαιτήσεις, καθιστώντας την προστασία της ποιότητας υπηρεσίας ζωτικής σημασίας. Για παράδειγμα, κατά την εικονικοποίηση δικτυακών λειτουργιών (Network Function Virtualization) ([11]), όπως π.χ. προώθηση πακέτων (forwarding), τείχη προστασίας (firewalls), δρομολόγηση (routing) κτλ., η παροχή ποιότητας υπηρεσίας, η προβλέψιμη καθυστέρηση των πακέτων (packet latency), καθώς και η καλή απόδοση συστήματος είναι βασικές απαιτήσεις των παρόχων υπηρεσιών Cloud.

Επιπλέον, εκτός από τη βέλτιστη εκτέλεση των εφαρμογών, μία άλλη πρόκληση των πολυεπεξεργαστικών συστημάτων αφορά την ορθολογική και οικονομική εκμετάλλευση τους. Συγκεκριμένα, οι πάροχοι υπηρεσιών Cloud, προκειμένου να εξασφαλίσουν την ποιότητα υπηρεσίας και να τηρήσουν τα SLAs (Server Level Argreements) των πελατών, αναγκάζονται να απομονώνουν σε συγκεκριμένους επεξεργαστές την εκτέλεση κρίσιμων εφαρμογών (latency critical) με εγγύηση μεν, την εξάλειψη του ανταγωνισμού για τους κοινόχρηστους πόρους, με κόστος δε, την υποχρησιμοποίηση του συστήματος ([5], [10]). Για παράδειγμα, οι servers της Google που εξυπηρετούν τις αναζητήσεις των χρηστών στο διαδίκτυο (websearch) έχουν κατά μέσο όρο 30% χρησιμοποίηση ([5]). Με άλλα λόγια, μία συστοιχία 10000 επεξεργαστών εξυπηρετεί ανάγκες για τις οποίες δυνητικά θα επαρκούσαν 3000 επεξεργαστές.

1.3 Ορολογία

Στην εργασία χρησιμοποιούμε την ορολογία που χρησιμοποιούν οι Smith κ.α. ([12]).

- **cache block** : Το cache block είναι ένα σύνολο από λέξεις με κοινό tag που διαχωρίζονται από τα λιγότερο σημαντικά bits της διεύθυνσής της μνήμης. Τυπικά το μέγεθός τους έχει μήκος από 4 έως 8 λέξεις. Στους επεξεργαστές που μας ενδιαφέρουν και χρησιμοποιούμε στην εργασία μας το cache block έχει μέγεθος 64 bytes.
- **cache set/line** : Το cache set/line είναι ένα σύνολο από blocks τα tags των οποίων ελέγχονται παράλληλα κατά την προσπέλαση της κρυφής μνήμης (ίδιο index).
- **way** : Ο αριθμός των ways καθορίζεται από το βαθμό συσχετιστικότητας (assosiativity) και είναι ίσος με τον αριθμό των cache blocks που περιέχονται σε κάθε cache line.



Σχήμα 1.1: Παράδειγμα μιας 4-way set associative cache με 4 sets

1.4 Συνεισφορά της Εργασίας

Στα πλαίσια της παρούσας διπλωματικής εργασίας μελετάμε την τεχνολογία Intel RDT (Resource Director Technology) ([13]), που έχει ενσωματωθεί στους τελευταίας γενιάς επεξεργαστές της Intel, με στόχο την ανάλυση και μελέτη των μειονεκτημάτων της συνεκτέλεσης εφαρμογών σε μεγάλα επεξεργαστικά συστήματα, καθώς επίσης, και την εύρεση κατάλληλου μηχανισμού που εξασφαλίζει την απομόνωση των εφαρμογών, την υψηλή απόδοση και την ποιότητα υπηρεσίας. Όπως αναλύουμε και στη συνέχεια, παρέχονται οι τεχνολογίες Cache Allocation Technology (CAT) και Cache Monitoring Technology (CMT), με τις οποίες καθίσταται δυνατός ο διαμοιρασμός και η επίβλεψη της χρήσης του τελευταίου επιπέδου cache στο chip του επεξεργαστή (Last-Level-Cache, LLC) στους πυρήνες ή/και στις εφαρμογές. Συγκεκριμένα, μελετάμε την περίπτωση όπου στο σύστημα συνυπάρχουν και εκτελούνται ταυτόχρονα μία εφαρμογή που χαρακτηρίζεται ως υψηλής προτεραιότητας και άλλες μικρότερης προτεραιότητας. Στόχος μας είναι η προστασία της εκτέλεσης της υψηλής προτεραιότητας εφαρμογής. Για την επίτευξη αυτού του στόχου, υλοποιούμε έναν δυναμικό μηχανισμό, ο οποίος, εκμεταλλευόμενος τις παραπάνω τεχνολογίες και χωρίς πρωτύτερη γνώση για το είδος των συνεκτελούμενων εφαρμογών, διαχειρίζεται κατάλληλα την κοινόχρηστη μνήμη cache εξασφαλίζοντας όσο το δυνατόν καλύτερη επίδοση για την υψηλής προτεραιότητας εφαρμογή. Ο παραπάνω δυναμικός μηχανισμός υλοποιήθηκε με χρήση του API που μας παρέχει το εργαλείο PQoS.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 Εισαγωγή

Οι υπολογιστικές ανάγκες των σύγχρονων κέντρων δεδομένων απαιτούν επεξεργαστικά συστήματα με όλο και περισσότερους πυρήνες. Η αύξηση των πυρήνων στον επεξεργαστή συνεπάγεται και μεγαλύτερο περιθώριο για ταυτόχρονη εκτέλεση περισσότερων εφαρμογών. Η συνεκτέλεση εφαρμογών (co-execution) συνήθως δημιουργεί ανταγωνισμό για τους κοινόχρηστους πόρους του συστήματος, όπως π.χ. για τις κοινόχρηστες μνήμες cache, και κατ' επέκταση βλάπτει την επίδοση των εφαρμογών και την ποιότητα υπηρεσίας. Όπως έχουμε επισημάνει, στα σύγχρονα κέντρα δεδομένων για να αποφευχθεί το παραπάνω πρόβλημα οι επεξεργαστές υποχρησιμοποιούνται, εκτελώντας αισθητά λιγότερο πλήθος εφαρμογών από το μέγιστο δυνατό και αυξάνοντας έτσι το κόστος λειτουργίας και το συνολικό πλήθος των επεξεργαστικών μονάδων που απαιτούνται για την εκτέλεση όλων των εφαρμογών. Είναι καίριο, λοιπόν, το ζήτημα να εξασφαλιστεί η ποιότητα υπηρεσίας μιας εφαρμογής και παράλληλα να ελαχιστοποιηθούν τα κόστη συντήρησης του κέντρου δεδομένων. Μία ενδεχόμενη λύση αποτελεί η πλήρης χρησιμοποίηση των επεξεργαστών μέσω της συνεκτέλεσης εφαρμογών και η κατάλληλη διαχείριση των κοινόχρηστων πόρων για την προστασία του QoS και την εξασφάλιση της βέλτιστης επίδοσης του συστήματος.

Η χρήση κρυφών μνημών cache (on-chip memory) βελτιώνει κατά κανόνα την επίδοση των εφαρμογών επειδή, διατηρώντας κάποια δεδομένα on chip, μειώνει τις χρονοβόρες και ενεργειακά κοστοβόρες προσβάσεις του επεξεργαστή στην κύρια μνήμη. Στους σύγχρονους επεξεργαστές συνήθως συνυπάρχουν ταυτόχρονα ιδιωτικές (private caches) και διαμοιραζόμενες μνήμες cache (shared caches). Οι ιδιωτικές μνήμες ανήκουν αποκλειστικά σε έναν φυσικό πυρήνα και είναι προσπελάσιμες μόνο από αυτόν. Οι διαμοιραζόμενες μνήμες βρίσκονται στο τελευταίο επίπεδο στην ιεραρχία μνήμης (Last-Level-Cache, LLC) και είναι κοινόχρηστες μεταξύ των επεξεργαστών μιας υπολογιστικής μονάδας. Παρότι οι ιδιωτικές μνήμες εξασφαλίζουν απομόνωση (isolation), είναι μικρές για να εξυπηρετήσουν το συγκριτικά μεγάλο φορτίο των εφαρμογών που τρέχουν στα κέντρα δεδομένων. Από την την άλλη πλευρά, οι διαμοιραζόμενες μνήμες cache έχουν επαρκές μέγεθος ώστε να εξυπηρετούν τις εφαρμογές, όμως, κατά τη συνεκτέλεση αυτών δημιουργείται ανταγωνισμός για αυτόν τον κοινόχρηστο πόρο. Με άλλα λόγια, η αίτηση για δεδομένα μιας εφαρμογής P2 μπορεί να "διώξει" blocks δεδομένων από την LLC που ανήκουν σε μια άλλη εφαρμογή P1. Ως αποτέλεσμα, αυξάνεται το miss-rate της P1, οι προσβάσεις της στην κύρια μνήμη και χειροτερεύει η επίδοσή της. Γενικά, ο ανταγωνισμός για κοινόχρηστους πόρους, ως συνέπεια της συνεκτέλεσης εφαρμογών, αυξάνει το συνολικό miss-rate, την κατανάλωση ενέργειας (energy consumption), προκαλεί κορεσμό στο bandwidth, κακή ποιότητα υπηρεσίας, μη δικαιοσύνη (unfaireness) στο σύ-

στημα και κακή επίδοση.

Οι παραπάνω περιορισμοί οδήγησαν στη μελέτη και ανάπτυξη τεχνικών διαχείρισης της κοινόχρηστης μνήμης cache με στόχο τον κατάλληλο διαμοιρασμό των τμημάτων της στις εφαρμογές (Cache Partitioning, CP). Το CP σε περιβάλλον συνεκτέλεσης παρέχει απομόνωση των εφαρμογών, συμβάλει στην καλύτερη επίδοση της εκτέλεσής τους, στη δικαιοσύνη του συστήματος και στην παροχή QoS.

Οι τεχνικές αυτές μπορούν να διαχωριστούν σε δύο μεγάλες κατηγορίες :

- Software Based Τεχνικές : Συσχετίζουν κάθε εφαρμογή με διάφορα τμήματα της κοινόχρηστης μνήμης cache. Μειονέκτημα αυτής της τεχνικής είναι ότι συνήθως απαιτεί ανάπτυξη λογισμικού σε επίπεδο πυρήνα (kernel level developement) και εισάγει πολυπλοκότητα στο λειτουργικό σύστημα (OS) ή στον επόπτη (hypervisor).

- Hardware Based Τεχνικές : Οι πυρήνες μιας επεξεργαστικής μονάδας συσχετίζονται με τμήματα της κοινόχρηστης μνήμης cache. Μειονέκτημα της τεχνικής αυτής είναι ότι χρειάζεται υποστήριξη από το υλικό (hardware).

Στους σύγχρονους επεξεργαστές ιδιαίτερα διαδεδομένη τεχνική είναι ο χρωματισμός σελίδων (page coloring) ο οποίος εμπίπτει στην κατηγορία των Software Based Τεχνικών και έχει ενσωματωθεί σε διανομές Linux. Επιπλέον, στους επεξεργαστές Opteron AMD υπάρχει κατάλληλο κύκλωμα ελεγκτή της επιπέδου 3 μνήμης cache (Level 3 Cache Controller), ο οποίος ανιχνεύει εφαρμογές που δεν επωφελούνται ουσιαστικά από την μνήμη cache (π.χ. πολλές προσβάσεις σε αυτήν, αλλά χαμηλό hit-rate) και τροποποιεί την πολιτική αντικατάστασης αυτών των cache lines θέτοντας τες ως LRU. Τέλος, στους επεξεργαστές Intel Xeon της γενιάς E5 v3 και πάνω υπάρχει κατάλληλο υλικό που υποστηρίζει το διαμοιρασμό της LLC σε πυρήνες. Ο μηχανισμός αυτός χρησιμοποιεί way-based τεχνική διαμοιρασμού, δηλαδή κάθε πυρήνας έχει πρόσβαση σε συγκεκριμένο αριθμό από ways κάθε set.

2.2 Διαχείριση Κοινόχρηστης Μνήμης Cache μέσω τεχνικών λογισμικού (software)

2.2.1 Χρονοδρομολογητής του Λειτουργικού Συστήματος (OS Scheduler)

Κατά την εκτέλεση πολλών εφαρμογών σε έναν επεξεργαστή δημιουργείται ανταγωνισμός για τους κοινούς πόρους, όπως π.χ. η φυσική μνήμη και η LLC. Ωστόσο, αν και το λειτουργικό σύστημα είναι υπεύθυνο για την δέσμευση σελίδων μνήμης (page frames), δεν μπορεί να διαχειριστεί την κρυφή μνήμη. Επομένως, δεν υπάρχουν μηχανισμοί του OS που μπορούν να μειώσουν την αλληλεπίδραση των εφαρμογών (software threads) στην LLC με αποτέλεσμα cache-misses μιας εφαρμογής να ”διώχνουν” blocks από την LLC που περιέχουν δεδομένα άλλης ταυτόχρονα χρονοδρομολογημένης εφαρμογής. Συνεπάγεται, επομένως, ότι μία εφαρμογή όταν συνεκτελείται με άλλες που έχουν υψηλό miss-rate θα παρουσιάσει χειρότερη επίδοση από την περίπτωση που οι συνεκτελούμενες εφαρμογές έχουν χαμηλό miss-rate.

Στην εργασία [8], οι ερευνητές επεκτείνουν τον χρονοδρομολογητή του Solaris 10 ενσωματώνοντας σε αυτόν έναν δίκαιο ως προς την cache αλγόριθμο (cache-fair). Στόχος τους είναι να διασφαλίσουν την βέλτιστη εκτέλεση μιας εφαρμογής ανεξαρτήτως του τμήματος της LLC που χρησιμοποιεί. Ο αλγόριθμος αυτός ρυθμίζει το κβάντο χρόνου (time slice) των νημάτων της εφαρμογής, δηλαδή ρυθμίζει το χρόνο που αυτά θα τρέχουν στους πυρήνες του επεξεργαστή μέχρι να χρονοδρομολογηθούν νέα. Παράλληλα, ελέγχει το IPC (Instructions Per Cycle) της εφαρμογής και όταν αυτό είναι μικρότερο του ”δίκαιου IPC” (βλ. παρακάτω) αυξάνει το κβάντο χρόνου για τη συγκεκριμένη εφαρμογή, ενώ αντίστοιχα, όταν αυτό είναι πολύ μεγαλύτερο το ελαττώνει. Είναι προφανές ότι αυξάνοντας το κβάντο χρόνου η εφαρμογή αξιοποιεί καλύτερα την LLC αφού τα δεδομένα της δεν ”διώχνονται” από άλλες. Εξηγούμε ότι για την εύρεση του ”δίκαιου IPC” έχουν αναπτυχθεί θεωρητικά μοντέλα, τα οποία στην πράξη προβλέπουν το IPC που θα είχε η εφαρμογή με δίκαιο διαχωρισμό μνήμης χρησιμοποιώντας διάφορες μετρικές, όπως π.χ. τα cache misses ([14]). Η συγκεκριμένη τροποποίηση του χρονοδρομολογητή υλοποιεί μια τεχνική καταμερισμού της LLC στις εφαρμογές και εξασφαλίζει την ποιότητα υπηρεσίας χωρίς τη χρήση ειδικού υλικού στον επεξεργαστή.

Στην εργασία [7], παρουσιάζεται η μεθοδολογία Bubble-Up, η οποία προβλέπει με απόκλιση 1% τη μείωση της επίδοσης μίας εφαρμογής όταν συνεκτελείται με άλλες λόγω ανταγωνισμού για τους διαμοιραζόμενους πόρους στο υποσύστημα της μνήμης (memory subsystem). Χρησιμοποιώντας το Bubble-Up σε 17 production workloads της Google μπορούμε να βρούμε ”έξυπνους” συνδυασμούς εφαρμογών για συνεκτέλεση, διατηρώντας πάντα το QoS των χρονικά ευαίσθητων εφαρμογών και αυξάνοντας τη χρησιμοποίηση των εξυπηρετητών από 50% έως και 90%. Συγκεκριμένα, το Bubble-Up ερευνά την ευαισθησία μιας εφαρμογής ως προς την LLC, δηλαδή πόσο η επίδοση της επηρεάζεται όταν συνεκτελείται με εφαρμογές που ”μολύνουν” την LLC (cache pollution), και την πίεση που η ίδια ασκεί στην LLC, δηλαδή πόσο αυτή επηρεάζει τις συνεκτελούμενες εφαρμογές. Η Bubble-Up παρέχει ένα είδους profiling των εφαρμογών και στοχεύσει στην εύρεση συνδυασμών εφαρμογών που μπορούν να συνεκτελεστούν αποδοτικά. Ωστόσο, εξετάζει τις εφαρμογές αποκλειστικά ως προς τις ανάγκες τους για LLC και δεν λαμβάνει υπόψιν άλλους διαμοιραζόμενους πόρους, οπως π.χ. το DRAM bandwidth.

Άλλες τεχνικές χρονοδρομολόγησης των εφαρμογών με στόχο τη βελτίωση της επίδοσης αφορούν τον σχεδιασμό πολιτικών που επιβάλουν κατάλληλη προτεραιότητα για cache στις εφαρμογές. Παράδειγμα τέτοιας υλοποίησης αποτελεί το CQoS ([2]), το οποίο αρχικά κατηγοριοποιεί τις εφαρμογές ανάλογα με την αλληλεπίδρασή τους με την μνήμη και συγκεκριμένα αναζητά τις ετερογενείς ως προς τις προσβάσεις στη μνήμη εφαρμογές (data structures, memory accesses, transactions, κτλ.). Σε δεύτερη φάση, επιβάλει την κατάλληλη προτεραιότητα σε κάθε εφαρμογή μέσω διάφορων μηχανισμών. Τέτοιους μηχανισμούς αποτελούν (α) ο στατικός ή δυναμικός καταμερισμός των sets της cache με στόχο οι εφαρμογές υψηλής προτεραιότητας να καταλαμβάνουν περισσότερα ways σε κάθε set από ότι οι εφαρμογές χαμηλής προτεραιότητας, (β) η επιλεκτική δέσμευση της cache, κατά την οποία διατηρούνται πληροφορίες για το πλήθος των cache lines που καταλαμβάνουν εφαρμογές συγκεκριμένης προτεραιότητας και βάσει πιθανοτικών μοντέλων επιτρέπει ή απαγορεύει τη δέσμευση cache lines από τις εφαρμογές, και (γ) τις ετερογενείς περιοχές cache που αφορούν την πρόσβαση σε victim caches ή stream buffers.

2.2.2 Χρωματισμός Σελίδων (Page Coloring)

Ο χρωματισμός σελίδων (page ή cache coloring) είναι μια τεχνική κατά την οποία δεσμεύονται κατάλληλες σελίδες (virtual pages) της εικονικής μνήμης (virtual memory) έτσι ώστε να αντιστοιχίζονται σε συνεχόμενα cache line/sets του επεξεργαστή. Στόχος της τεχνικής αυτής είναι να μεγιστοποιηθεί ο αριθμός των συνολικών σελίδων από τις οποίες ο επεξεργαστής αποθηκεύει δεδομένα στην διαμοιραζόμενη μνήμη cache χωρίς να συμβαίνουν conflict misses λόγω αντιστοίχισης πολλών σελίδων στο ίδιο cache line ή cache set. Με το cache coloring, αφενός, παρατηρείται καλύτερη επίδοση μιας εφαρμογής λόγω της μείωσης των προσβάσεων στην κύρια μνήμη και αφετέτου είναι εφικτό όλο το σύνολο των εικονικών διευθύνσεων μιας διεργασίας να αντιστοιχιστεί σε συγκεκριμένα cache lines της LLC, επιτυγχάνοντας με αυτόν τον τρόπο στατικό διαμερισμό της μνήμης cache (static cache partitioning) μεταξύ των διεργασιών.

Το page coloring συνήθως υλοποιείται σε επίπεδο λειτουργικού συστήματος, από χαμηλού επιπέδου κώδικα που αναλαμβάνει τη δυναμική δέσμευση μνήμης και την διαδικασία μετάφρασης/αντιστοίχισης της φυσικής μνήμης στην εικονική μνήμη. Όταν το page coloring υποστηρίζεται από το OS, τα πλαίσια της φυσικής μνήμης (physical memory page frames) "χρωματίζονται" ώστε διαφορετικοί χρωματισμοί να αντιστοιχούν σε διαφορετικά lines στην κρυφή μνήμη. Στην συνέχεια, το MMU (Memory Management Unit) κατά τη δέσμευση (allocation) του εικονικού χώρου διευθύνσεων μιας διεργασίας, επιλέγει είτε να αντιστοιχίσει τις εικονικές σελίδες σε πλαίσια σελίδων φυσικής μνήμης με διαφορετικό χρωματισμό, ούτως ώστε τα δεδομένα που έρχονται από τη μνήμη να μην ανταγωνίζονται το ίδιο cache line/set, είτε στον ίδιο χρωματισμό, έτσι ώστε τα δεδομένα της διεργασίας να "βλέπουν" συγκεκριμένο τμήμα της cache. Επομένως, σελίδες που αντιστοιχίζονται στο ίδιο σετ της cache έχουν το ίδιο χρώμα σελίδας (page color).

Ιστορικά, το page coloring χρησιμοποιήθηκε αρχικά στη μονάδα μετάφρασης εικονικών διευθύνσεων (virtual address translation unit) των επεξεργαστών αρχιτεκτονικής MIPS ([15]) και στη συνέχεια ενσωματώθηκε σε λειτουργικά σύστηματα (Solaris, FreeBSD, NetBSD) ([16]) και στους μεταγλωττιστές ([17]). Τα πλεονεκτήματα αυτής της τεχνικής, στη γενική περίπτωση, είναι ότι υλοποιεί στατικό διαμοιρασμό της μνήμης cache, βελτιώνει την απόδοση όταν συνεκτελούνται διεργασίες και εξασφαλίζει ντετερμινιστική εκτέλεση των προγραμμάτων. Ένα από τα μειονεκτήματά της είναι ότι απαιτεί ανάπτυξη λογισμικού σε επίπεδο πυρήνα και εισάγει πολυπλοκότητα στη Μονάδα Διαχείρισης Μνήμης (OS Memory Management Subsystem). Επιπλέον, στα σύγχρονα λειτουργικά συστήματα που υποστηρίζουν "μεγάλες" σελίδες ή αλλιώς hugepages (μέγεθος σελίδας $\geq$ 2MB), όταν αυτή η δυνατότητα είναι ενεργοποιημένη, το cache coloring δεν εξασφαλίζει ντετερμινισμό κατά την συνεκτέλεση, αφού οι αριθμοί των σελίδων είναι αισθητά μικρότεροι σε πλήθος από την κλασσική περίπτωση στο Linux OS (μέγεθος σελίδας = 4KB) και περισσότερες από πριν χρωματίζονται με το ίδιο χρώμα με αποτέλεσμα να αντιστοιχίζονται στο ίδιο cache set.

2.3 Διαχείριση Κοινόχρηστης Μνήμης Cache μέσω τεχνικών υλικού (hardware)

Η διαχείριση της κοινόχρηστης μνήμης cache στο παρελθόν σε επίπεδο υλικού περιοριζόταν κυρίως στην αντικατάσταση της κλασσικής LRU πολιτικής με άλλες καταλληλότερες που λάμβαναν υπόψιν τις απαιτήσεις και τις ιδιαιτερότητες της εφαρμογής ως προς μνήμη. Σήμερα, η Intel

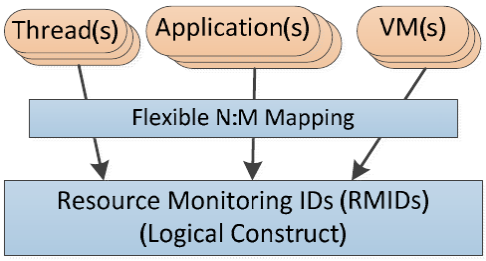
μας παρέχει τη δυνατότητα να διαμοιράσουμε την κοινόχρηστη μνήμη LLC στις εφαρμογές μέσω ειδικών καταχωρητών που ενσωματώνονται στους επεξεργαστές της.

2.3.1 Τεχνολογία Intel RDT

Οι νέοι επεξεργαστές της οικογένειας Intel Xeon E5 v3 υποστηρίζουν την τεχνολογία Intel RDT (Resource Director Technology), η οποία επιτρέπει την επίβλεψη και διαχείριση των κοινόχρηστων πόρων του επεξεργαστή, όπως η LLC. Οι τεχνολογίες αυτές είναι οι Cache Monitoring Technology (CMT), Memory Bandwidth Monitoring Technology (MBM) και Cache Allocation Technology (CAT). Η τεχνολογία CMT επιτρέπει στο λειτουργικό σύστημα, στον επόπτη ή σε οποιονδήποτε διαχειριστή του συστήματος μνήμης να επιβλέπει την χρήση της LLC από τις εφαρμογές που τρέχουν στο σύστημα. Αντίστοιχα, η τεχνολογία MBM επιτρέπει την επίβλεψη του DRAM bandwidth ανά εφαρμογή ή πυρήνα ή και ομάδες αυτών στον επεξεργαστή. Αναφέρουμε ότι στην περίπτωση που υπάρχουν περισσότερες από μία υποδοχές επεξεργαστών (sockets) στην πλατφόρμα είναι δυνατόν να επιβλέπουμε και το bandwdith μεταξύ αυτών. Τέλος, η τεχνολογία CAT δίνει την δυνατότητα σε οποιονδήποτε διαχειριστή του συστήματος να καθορίσει το τμήμα της LLC που μπορεί να χρησιμοποιήσει κάθε πυρήνας, εφαρμογή ή/και ομάδες αυτών.

Οι τεχνολογίες CMT-CAT υποστηρίζονται από το υλικό με την προσθήκη καταχωρητών και με τη χρήση επιπέδων αφαίρεσης (Abstraction Layers) που συσχετίζουν τους λογικούς πυρήνες (hardware threads) με τις εκτελούμενες εφαρμογές.

Συγκεκριμένα, για το CMT, έχει προστεθεί ένα επίπεδο αφαίρεσης μεταξύ των εφαρμογών που τρέχουν στο σύστημα και των λογικών πυρήνων μέσω της χρήσης των RMIDs. Κάθε εφαρμογή ή και συνδυασμός αυτών συσχετίζεται με ένα RMID (Σχήμα 2.1). Επίσης, κάθε λογικός επεξεργαστής ή και ομάδα αυτών μπορεί να συσχετιστεί με ένα RMID. Για κάθε λογικό πυρήνα, μόνο ένα RMID είναι ενεργό κάθε φορά και ο συνολικός αριθμός των διαθέσιμων RMIDs είναι χαρακτηριστικό του επεξεργαστή.



Σχήμα 2.1 Συσχέτιση νημάτων με RMIDs

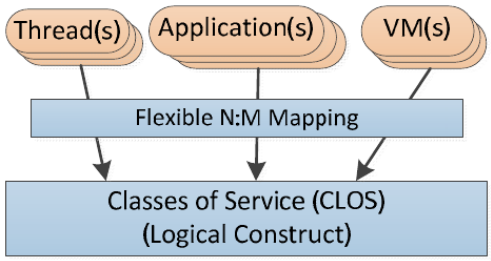
Κάθε λογικός πυρήνας έχει έναν έναν ξεχωριστό καταχωρητή, τον IA32_PQR_ASSOC MSR ή αλλιώς PQR, ο οποίος περιέχει πάντα το ενεργό RMID (Σχήμα 2.4). Για παράδειγμα, όταν μια εφαρμογή χρονοδρομολογηθεί σε έναν πυρήνα ο PQR καταχωρητής ενημερώνεται με το συγκεκριμένο RMID. Το RMID αλλάζει όταν δρομολογηθεί σε αυτόν νέα εφαρμογή (context switch). Η αντιστοίχιση εφαρμογών, νημάτων, εικονικών μηχανών κτλ. (software threads) και RMIDs επιτρέπει στο υλικό να ανακτήσει δεδομένα σχετικά με τη χρήση των κοινόχρηστων πόρων από την εκάστοτε εφαρμογή.

Προκειμένου να γίνουν διαθέσιμες οι τιμές χρησιμοποίησης των πόρων για κάθε εφαρμογή ή για κάθε ομάδα εφαρμογών, προστέθηκαν δύο ακόμα καταχωρητές. Ο πρώτος, IA32_QM_EVTSEL MSR, περιέχει πληροφορίες για το είδος των δεδομένων που θα διαβαστούν. Αυτό επιτυγχάνεται με την αντιστοίχιση του RMID με ένα αναγνωριστικό συμβάντος (Event Code). Το RMID υποδεικνύει την εφαρμογή ενώ το Event Code χαρακτηρίζει το είδος των δεδομένων που αφορούν την εφαρμογή όπως π.χ. τα Misses, το Memory bandwidth ή το LLC Occupancy. Τονίζουμε ότι το RMID δεν

αλλάζει ανάλογα με το είδος της πληροφορίας, αλλά για κάθε είδος δεδομένων υπάρχει ξεχωριστό Event Code. Ο δεύτερος καταχωρητής, IA32_QM_CTR MSR, περιέχει την τιμή των δεδομένων (raw format), καθώς επίσης και συγκεκριμένα bits για να διασφαλιστεί η εγκυρότητα της πληροφορίας.

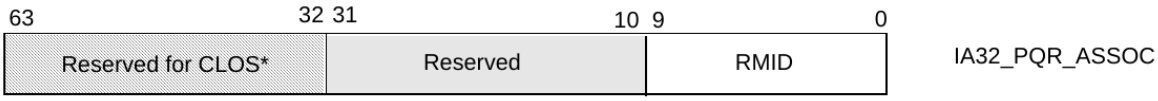
(a) Καταχωρητής IA32_QM_EVTSEL MSR **(b)** Καταχωρητής IA32_QM_CTR MSR

Σχήμα 2.2: Καταχωρητές IA32_QM_EVTSEL και IA32_QM_CTR



Σχήμα 2.3 Συσχέτιση νημάτων με CLOSs

Για την τεχνολογία CAT, υπάρχει ένα επίπεδο αφαίρεσης μεταξύ διεργασιών (software threads) και λογικών πυρήνων (hardware threads), το Class of Service (CLOS). Κάθε PID ή σύνολο αυτών μπορούν να αντιστοιχιστούν σε ένα CLOS. Επίσης, σε κάθε CLOS μπορούν να αντιστοιχιστούν ένας ή και περισσότεροι λογικοί πυρήνες. Αρχικά, δεν υπάρχει καμία ομαδοποίηση των πυρήνων και όλοι ανήκουν στο CLOS 0. Ο PQR καταχωρητής που αναφέραμε στην περιγραφή του CMT περιέχει και την πληροφορία του CLOS (Σχήμα 2.3). Το RMID και το CLOS είναι ανεξάρτητα μεταξύ τους και έτσι η λειτουργία του CMT δεν επηρεάζει ή επηρεάζεται από το CAT. Η τεχνολογία CAT προσφέρει way-based διαμοιρασμό της cache. Αυτό πραγματοποιείται με τη χρήση κατάλληλων bitmasks, οι οποίες ορίζουν το τμήμα της cache το οποίο "ανήκει" σε κάθε CLOS. Οι εφαρμογές που ανήκουν στο συγκεκριμένο CLOS μπορούν να γράψουν δεδομένα μόνο στο συγκεκριμένο τμήμα της cache όμως μπορούν να διαβάσουν ολόκληρη την cache.

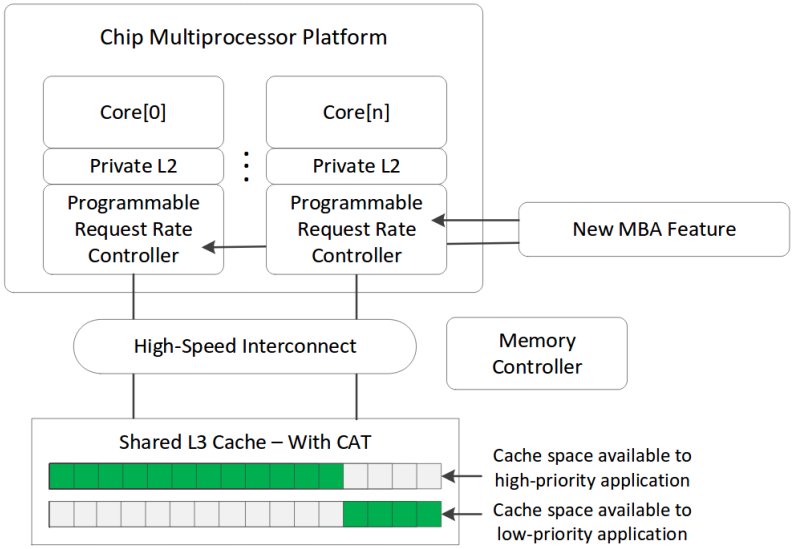


Σχήμα 2.4: Καταχωρητής IA32_PQR_ASSOC MSR

Οι τεχνολογίες αυτές της Intel είναι καθοριστικές στο Cloud και στον τομέα του High Performance Computing, όπου η απομόνωση των πόρων ανά εφαρμογή με στόχο την καλύτερη εκμετάλλευση του συστήματος και την καλύτερη απόδοση είναι μείζονος σημασίας. Επιπλέον, σε συνδυασμό με το CMT είναι εφικτός ο σχεδιασμός κατάλληλου μηχανισμού στο OS ή στον VMM (Virtual Memory Manager) ο οποίος να βρίσκει το βέλτιστο συνδυασμό πυρήνων και τμήματος cache για κάθε εφαρμογή. Ακόμα, βοηθάει στη μελέτη και στο profiling της επίδοσης των εφαρμογών καθώς καθίσταται εφικτή η εύρεση συνάρτησης επίδοσης και μνήμης cache μιας εφαρμογής. Τέλος, ευνοεί τους παρόχους Cloud υπηρεσιών, γιατί μπορούν να υιοθετήσουν πολιτικές χρέωσης με βάση τη πραγματική χρήση των πόρων από τις υπηρεσίες. Συμπληρωματικά με τις παραπάνω τεχνολογίες, υποστηρίζεται η δυνατότητα να διαχωρίσουμε την cache σε τμήματα αποκλειστικά μόνο για εντολές (code) ή δεδομένα (data) μέσω της τεχνολογίας CDP (Code and Data Prioritization).

Τέλος, στη νέα γενιά επεξεργαστών Intel Xeon E5 v4 υποστηρίζονται οι τεχνολογίες για L2 Cache Allocation, που βρίσκει χρήση σε hyperthreading αρχιτεκτονικές, και Memory Bandwidth Allocation (MBA). Πιο αναλυτικά, η τεχνολογία MBA επιτρέπει την ρύθμιση του ρυθμού αιτήσεων

για δεδομένα από τη μνήμη ανά φυσικό πυρήνα. Στόχος της είναι να αποτρέψει τον κορεσμό ή την υπερχησιμοποίηση του DRAM bandwidth από εφαρμογές χαμηλής προτεραιότητας σε βάρος άλλων υψηλότερης προτεραιότητας. Αυτό, σε επίπεδο υλικού, επιτυγχάνεται με την προσθήκη ενός προγραμματιζόμενου ελεγκτή (Programmable Request Rate Controller) μεταξύ κάθε πυρήνα και του διαύλου διασύνδεσης.



Σχήμα 2.5: High-Level Περιγραφή της MBA τεχνολογίας

Η συγκεκριμένη τεχνολογία αξιοποιείται, επεκτείνοντας εννοιολογικά την υποδομή των Class of Services που αναλύσαμε στο CAT και ισχύουν τα εξής :

1. Ο αριθμός των CLOS που υποστηρίζουν CAT μπορεί να είναι διαφορετικός από τον αριθμό των CLOS που υποστηρίζουν MBA. Στην πραγματικότητα, συνήθως, τα CLOS που υποστηρίζουν MBA είναι ένα υποσύνολο των CLOS που υποστηρίζουν CAT. Για παράδειγμα, αν υπάρχουν 16 CLOS που υποστηρίζουν CAT και 8 CLOS που υποστηρίζουν MBA τότε τα 8 πρώτα CLOS, δηλαδή οι CLOS 0, 1, 2, 3, 4, 5, 6, 7 υποστηρίζουν ταυτόχρονα CAT και MBA.

2. Το MBA στην πραγματικότητα ορίζει τη καθυστέρηση (delay) του ρυθμού αιτήσεων για δεδομένα από τη μνήμη (request rate) του επεξεργαστή. Οι τιμές της μπορεί να ακολουθούν γραμμική (linear) ή όχι (non-linear) στοίχιση. Στην μη γραμμική στοίχιση η καθυστέρηση που μπορούμε να επιβάλουμε είναι πάντα δύναμη του 2, ενώ στη γραμμική οι τιμές απέχουν συγκεκριμένη σταθερά.

3. Το MBA ρυθμίζεται ανά φυσικό πυρήνα και όχι ανά λογικό. Εδώ αξίζει να πούμε ότι στην περίπτωση που έχουμε hyperthreading αρχιτεκτονική θα πρέπει να προσέξουμε τι είδους προτεραιότητα έχουν οι εφαρμογές που τρέχουν στους “δίδυμους” πυρήνες (sibling cores). Για παράδειγμα αν διαφορετικές εφαρμογές χρονοδρομολογηθούν στον ίδιον ίδιο φυσικό πυρήνα και ανήκουν σε διαφορετικά CLOS τότε η μεγαλύτερη καθυστέρηση επικρατεί και εφαρμόζεται στον πυρήνα. Άρα αν διαφορετικής προτεραιότητας εφαρμογές χρονοδρομολογηθούν στον ίδιο πυρήνα, σε διαφορετικά hyperthreads, τότε αυτό θα είναι επιζήμιο για την μεγαλύτερης προτεραιότητας εφαρμογή λόγω του περιορισμένου ρυθμού αιτήσεων στη μνήμη (bandwidth throttling) του CLOS της χαμηλότερης προτεραιότητας εφαρμογή.

2.4 Διαχείριση Κοινόχρηστης Μνήμης Cache συνδυάζοντας Τεχνικές Υλικού και Λογισμικού

Η ύπαρξη των νέων τεχνολογιών της Intel που περιγράψαμε στο προηγούμενο κεφάλαιο ανοίγει νέους δρόμους στη μελέτη τεχνικών για εξασφάλιση της ποιότητας υπηρεσίας σε συνδυασμό με την πλήρη χρησιμοποίηση των πόρων ενός πολυεπεξεργαστή κατά την ταυτόχρονη εκτέλεση εφαρμογών. Οι νέες αυτές τεχνικές αξιοποιούν τις τεχνολογίες του Cache Allocation και Cache Monitoring σε συνδυασμό με κάποιον επόπτη-διαχειριστή σε επίπεδο λογισμικού, ο οποίος δυναμικά κατανέμει την μνήμη cache στις εφαρμογές.

Χαρακτηριστική είναι η δουλειά των H. Cook, κ.α. ([10]) οι οποίοι προτείνουν έναν μηχανισμό που εκμεταλλεύεται τεχνικές υλικού για επιμερισμό της LLC σε πειραματικούς επεξεργαστές Intel της σειράς Sandy Bridge x86 και βελτιώνει την χρησιμοποίηση και την κατανάλωση ενέργειας στον επεξεργαστή, ενώ παράλληλα διατηρεί και προστατεύει την εκτέλεση των εφαρμογών (responsiveness).

Στην μελέτη τους παρατηρούν ότι η συνεκτέλεση εφαρμογών χωρίς κάποιο διαμοιρασμό της LLC στις εφαρμογές οδηγεί σε βελτίωση της εξοικονόμησης ενέργειας (10%) και της συνολικής απόδοσης (54%) στον επεξεργαστή συγκριτικά με την περίπτωση στην οποία οι εφαρμογές τρέχουν μόνες τους στο σύστημα (απομονωμένη εκτέλεση). Ωστόσο, σε εφαρμογές υψηλής προτεραιότητας η επίδοση μειώνεται κατά μέσο όρο 6% και σε κάποιες περιπτώσεις έως και 34% σε σχέση με την απομονωμένη εκτέλεση. Ο βέλτιστος στατικός διαμοιρασμός LLC σημειώνει καλύτερη ενεργειακή απόδοση κατά 12%, συνολική επίδοση 60% και μέση χρονική καθυστέρηση της υψηλής προτεραιότητας εφαρμογής 1%. Αν και ο στατικός διαμερισμός της cache επιλύει το αρχικό πρόβλημα, η εύρεση αυτού απαιτεί profiling των εφαρμογών που τρέχουν στο σύστημα, κάτι που είναι χρονοβόρο και πολλές φορές αδύνατο κατά τη διάρκεια της εκτέλεσης. Τους περιορισμούς αυτούς καλείται να αντιμετωπίσει ο δυναμικός αλγόριθμος που υλοποιούν, ο οποίος διαχειρίζεται τα τμήματα της LLC κατά την εκτέλεση και πετυχαίνει την επίδοση του βέλτιστου στατικού διαμερισμού. Παράλληλα αυξάνει και την επίδοση των λιγότερο σημαντικών εφαρμογών κατά 19%. Σχετικά με την κατανάλωση ενέργειας, δεν παρατηρείται μεγάλη απόκλιση με αυτή της στατικής πολιτικής που περιγράψαμε παραπάνω.

Δεδομένου ότι μία εφαρμογή μπορεί να έχει αλλαγές φάσης κατά την εκτέλεσή της και άρα διαφορετικές ανάγκες για LLC, ο μηχανισμός επιχειρεί να ανιχνεύσει τις αλλαγές φάσης με κριτήριο τα LLC misses per kilo-instruction (MPKI). Χαμηλό MPKI δείχνει ότι υπάρχει περιθώριο να μειωθεί το LLC τμήμα της υψηλής προτεραιότητας εφαρμογής χωρίς η μείωση αυτή να βλάψει την επίδοσή της. Σε κάθε αλλαγή φάσης, ο μηχανισμός καλείται να αποφασίσει για το LLC τμήμα που απαιτεί η νέα αυτή φάση. Συγκεκριμένα, όταν ξεκινάει μια εφαρμογή ή αλλάζει φάση ο μηχανισμός της δίνει όσο περισσότερη cache μπορεί, δηλαδή τα N-1 ways της LLC για N way συσχετιστική cache. Στην συνέχεια, σταδιακά μειώνει τα ways του συγκεκριμένου τμήματος cache μέχρις ότου η μείωση αποδειχθεί επιβλαβής για την επίδοση, δηλαδή αυξηθεί το MPKI, όπου τότε παραχωρεί περισσότερη cache. Κάθε φορά η υπόλοιπη cache παραχωρείται στις εφαρμογές παρασκηνίου.

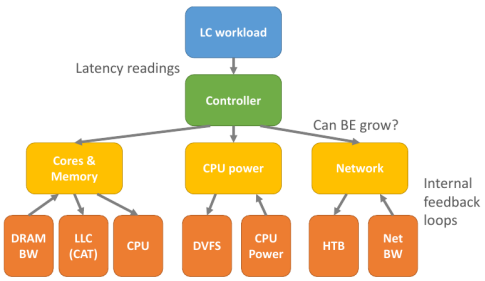
Ο δυναμικός αυτός μηχανισμός, όπως αναφέραμε προηγουμένως, επιτυγχάνει την καλύτερη δυνατή επίδοση των υψηλής προτεραιότητας εφαρμογών με ελάχιστη απόκλιση και βελτιώνει σημαντικά τις υπόλοιπες συνεκτελούμενες.

Παρόλα αυτά, δεν μπορεί πάντα να αναγνωρίσει ορθά τις αλλαγές φάσεις ώστε να διαλέξει το κατάλληλο LLC τμήμα για κάθε φάση. Σε κάποιες από αυτές τις περιπτώσεις, η μη βέλτιστη επιλογή LLC τμήματος έχει μηδαμινή επίδραση στην επίδοση της εφαρμογής με υψηλή προτεραιότητα και βελτιώνει περισσότερο τις λιγότερο σημαντικές εφαρμογές. Ωστόσο, είναι προφανές ότι στις περιπτώσεις εφαρμογών, πολύ ευαίσθητων ως προς την LLC, κακή επιλογή LLC τμήματος επιδρά αρνητικά στην επίδοση σημειώνοντας ωστόσο και πάλι, καλύτερα αποτελέσματα σε σχέση με την πολιτική πλήρους διαμοιραζόμενης cache.

Εξίσου ενδιαφέρουσα είναι και η ιδέα των Lo, κ.α. ([5]) που υλοποιούν τον Heracles, έναν δυναμικό ελεγκτή με ανάδραση ο οποίος αυξάνει την χρησιμοποίηση των επεξεργαστών στα κέντρα δεδομένων χωρίς να βλάπει την ποιότητα υπηρεσίας στις υψηλής προτεραιότητας (Latency Critical, LC) εφαρμογές. Συγκεκριμένα, ο Heracles στοχεύει στην αξιοποίηση των υποχρησιμοποιούμενων πόρων του συστήματος συντονίζοντας και συνεκτελώντας LC εφαρμογές με άλλες, λιγότερο σημαντικές, εργασίες (Best Efford, BE). Ο παραπάνω μηχανισμός βασίζεται στην παρατήρηση ότι κατά την συνεκτέλεση εφαρμογών υπάρχει κακή ποιότητα υπηρεσίας, παραβίαση των SLAs ή/και ελάττωση της υπολογιστικής επίδοσης όταν τουλάχιστον ένας από τους κοινούς και διαμοιραζόμενους πόρους του συστήματος φτάσει σε κορεσμό (π.χ. LLC, εύρος δικτύου, εύρος διαύλου δεδομένων μνήμης, κτλ.). Συνεπώς, ο Heracles στοχεύσει στην αποφυγή του κορεσμού σε κάθε έναν από τους διαμοιραζόμενους πόρους. Η αρχιτεκτονική του περιλαμβάνει έναν ελεγκτή που επιβλέπει κάθε διαμοιραζόμενο πόρο, καθώς επίσης, και έναν κύριο ελεγκτή που λαμβάνει αποφάσεις σχετικές με τη συνεκτέλεση.

Πιο συγκεκριμένα, ο κύριος ελεγκτής επιβλέπει την καθυστέρηση και το φορτίο της LC εφαρμογής και απαγορεύει ή επιτρέπει την συνεκτέλεση των BE εφαρμογών. Επιπλέον, είναι υπεύθυνος να ενημερώσει τους υπο-ελεγκτές για το αν θα πρέπει να παραχωρήσουν περισσότερους πόρους στις BE εργασίες ή να προστατεύσουν την LC εφαρμογή παραχωρώντας σε αυτή επιπλέον πόρους. Στις περιπτώσεις που παρατηρηθεί αύξηση του φορτίου ή παραβίαση των SLAs, ο ελεγκτής επιβάλει τη διακοπή των υπόλοιπων συνεκτελούμενων εφαρμογών και ενημερώνει τους υπο-ελεγκτές να παραχωρήσουν όλους τους πόρους στη LC εφαρμογή. Στη συνέχεια, όταν ομαλοποιηθεί η εκτέλεσή της, ενεργοποιεί ξανά τις BE εργασίες. Χρησιμοποιώντας ως κριτήριο το μέτρο της απόκλισης από τα SLAs, εξετάζει το περιθώριο να παραχωρηθούν ή να παρθούν πόροι από τις BE εργασίες και δίνει ανάλογη εντολή στους υπο-ελεγκτές.

Οι υπο-ελεγκτές βρίσκονται “κάτω” από τον κύριο ελεγκτή (Σχήμα 2.6), στο σύνολο είναι τρεις και διαχειρίζονται το εύρος δικτύου (network bandwidth), τη συχνότητα των πυρήνων και το πλήθος πυρήνων και τμήματος LLC για κάθε εφαρμογή. Ο πρώτος υπο-ελεγκτής αναλμβάνει να βρει τον βέλτιστο συνδυασμό πλήθους πυρήνων και τμήματος LLC μεταξύ των εφαρμογών. Αρχικά όλες οι BE εργασίες καταλαμβάνουν το 10% της LLC και δρομολογούνται σε έναν πυρήνα.



Σχήμα 2.6 Αρχιτεκτονική του Heracles

Το πρόβλημα εύρεσης του κατάλληλου συνδυασμού αποτελεί πρόβλημα δύο διαστάσεων (πλήθος πυρήνων, τμήμα LLC) και επιλύεται με τον αλγόριθμο Gradient Descent (offline analysis). Έτσι, ο υπο-ελεγκτής διακρίνει 2 φάσεις, τις GROW_LLC και GROW_CORES. Ξεκινώντας από τη φάση GROW_LLC, όταν λαμβάνει σήμα επίτρεψης (AllowBEGrowth()) από τον κύριο ελεγκτή και δεν παρατηρείται κορεσμός του DRAM bandwidth παραχωρεί ways στις BE εργασίες. Η παραχώρηση επιπλέον ways σε αυτές θα σταματήσει όταν νέα προσθήκη cache δεν ευνοεί τις BE εφαρμογές. Σε αυτό το σημείο, θα αλλάξει φάση (GROW_CORES) όπου και θα αρχίσει να παραχωρεί περισσότερους πυρήνες σε αυτές. Αν δημιουργηθεί κορεσμός στο bandwidth, αυξάνει το τμήμα cache που καταλαμβάνουν BE εφαρμογές. Ο συνδυασμός πυρήνων και τμήματος LLC που αντιμετωπίζει τον κορεσμό, είναι ο βέλτιστος. Σε περίπτωση που κάποια από αυτές τις αποφάσεις είναι καταστροφική για την επίδοση της LC εφαρμογής ο κύριος ελεγκτής ενημερώνει ότι πρέπει να σταματήσει η παραχώρηση πόρων στις BE εφαρμογές ή/και παύει την εκτέλεσή τους. Με αντίστοιχο δυναμικό τρόπο δρουν οι άλλοι δύο υπο-ελεγκτές για τους πόρους που διαχειρίζονται.

Για την αξιολόγηση του Heracles χρησιμοποιούνται αντιπροσωπευτικές εφαρμογές (latency critical workloads) που εκτελούνται στο κέντρο δεδομένων της Google. Ο μηχανισμός δεν παραβιάζει τα SLAs, αυξάνει τη χρησιμοποίηση των επεξεργαστών από 20% σε 90% με μία μικρή αύξηση στην κατανάλωση ενέργειας στον επεξεργαστή.

Οι Παπαδάκης, κ.α. ([4]) αξιοποιούν τις δυνατότητες των τεχνολογιών CAT-CMT και προτείνουν έναν ελεγκτή (DCP-QoS) ο οποίος εξασφαλίζει την ποιότητα υπηρεσίας μιας υψηλής προτεραιότητας εφαρμογής όταν αυτή συνεκτελείται με άλλες, μικρότερης προτεραιότητας, σε κατάσταση πλήρους χρησιμοποίησης του επεξεργαστή. Συγκεκριμένα, συνεκτελούνται N εφαρμογές σε επεξεργαστή με N πυρήνες (full workload) εκ των οποίων μία είναι η υψηλής προτεραιότητας (High Priority, HP) εφαρμογή και οι υπόλοιπες χαμηλής προτεραιότητας (Low Priority, LP). Στόχος του μηχανισμού είναι να προστατεύσει την επίδοση της HP εφαρμογής αυξάνοντας παράλληλα τη χρησιμοποίηση του συστήματος λόγω συνεκτέλεσης. Για την αξιολόγηση του DCP-QoS λαμβάνονται μετρήσεις και για άλλες δύο πολιτικές, την No-QoS, κατά την οποία δεν επιβάλλεται κανένας διαμοιρασμός της LLC και όλες οι εφαρμογές μοιράζονται την LLC και ανταγωνίζονται εξίσου για αυτή, και την CT-QoS, η οποία επιμερίζει στατικά την LLC ανάμεσα στις εφαρμογές παραχωρώντας 19 ways στην HP εφαρμογή και 1 way στις LP εφαρμογές.

Στην περίπτωση του No-QoS, η HP εφαρμογή σημειώνει μείωση στην επίδοσή της κατά μέσο όρο 59%. Αντίθετα, στην περίπτωση του CT-QoS, η HP εφαρμογή εκτελείται περίπου στο 84% του χρόνου απομονωμένης εκτέλεσης. Σε ό,τι αφορά τις LP εφαρμογές, κατά την πολιτική του CT-QoS, ο χρόνος εκτέλεσής τους μειώνεται περίπου κατά 96% σε σχέση με την απομονωμένη εκτέλεσή τους.

Έτσι, προτείνεται ο μηχανισμός DCP-QoS ο οποίος μοιράζει δυναμικά την LLC στις εφαρμογές, προστατεύοντας την HP εφαρμογή και παραχωρώντας την υπόλοιπη cache στις LP εφαρμογές. Η λογική του DCP-Qos βασίζεται σε τρεις καταστάσεις.

- Αρχική κατάσταση (Cache-Takeover) κατά την οποία παραχωρούνται 19 ways στην HP και 1 way στις LP εφαρμογές. Κάθε χρονικό διάστημα ενός δευτερολέπτου, ο μηχανισμός καλείται να λάβει μία απόφαση.

- Όσο διατηρείται σταθερή η επίδοση αφαιρείται 1 way απο την HP εφαρμογή και παραχωρείται στις υπόλοιπες.

- Όταν η επίδοση γίνει ασταθής, τότε ο μηχανισμός αποδίδει στην HP το προηγούμενο τμήμα cache. Στην επόμενη μέτρηση δύο πράγματα μπορούν να συμβούν :
  - η επίδοση εξακολουθεί να είναι ασταθής. Ο μηχανισμός μεταβαίνει στην αρχική κατάσταση Cache-Takeover.
  - η επίδοση σταθεροποιείται. Ο μηχανισμός μεταβαίνει σε κατάσταση Balanced στην οποία σταματάει ο διαμοιρασμός της LLC και συνεχίζει η εκτέλεση με αυτό το σχήμα καταμερισμού στις εφαρμογές.
- Ο μηχανισμός παραμένει σε κατάσταση Balanced μέχρις ότου η επίδοση γίνει ασταθής. Στην συνέχεια, προσθέτει 1 way στην υψηλής προτεραιότητας εφαρμογή. Αν η προσθήκη του way δεν σταθεροποιήσει την επίδοση τότε πάει στην αρχική κατάσταση Cache-Takeover και επαναλαμβάνει τη διαδικασία.

Το DCP-QoS συγκριτικά με το CT-QoS είναι εξίσου αποτελεσματικό για τις HP εφαρμογές αφού αυτές εκτελούνται στο 80% του χρόνου απομονωμένης εκτέλεσης. Στις LP εφαρμογές πετυχαίνει έως και 5 φορές καλύτερη επίδοση από ότι το CT-QoS.

Στην παρούσα διπλωματική αξιοποιούμε τις τεχνολογίες Intel CMT-CAT στον επεξεργαστή Intel Xeon E5 v3 2660. Αντλώντας ιδέες από τις μελέτες της τελευταίας ενότητας που αναλύσαμε, υλοποιούμε έναν ελεγκτή ο οποίος διαχειρίζεται κατάλληλα την LLC μεταξύ των εφαρμογών στο σύστημα με στόχο να προστατεύσει την εκτέλεση μιας υψηλής προτεραιότητας εφαρμογής και παράλληλα να βελτιώσει το καλύτερο δυνατόν την εκτέλεση των χαμηλότερης προτεραιότητας εφαρμογών.

Κεφάλαιο 3

Συνεκτελέσεις των Εφαρμογών

3.1 Πλατφόρμα Πειράματος και Μετροπρογράμματα (Benchmarks)

Στο παρόν κεφάλαιο καλούμαστε να επιβεβαιώσουμε και να κατηγοριοποιήσουμε τα προβλήματα που δημιουργεί η συνεκτέλεση εφαρμογών σε επεξεργαστές της σειράς Xeon της Intel, καθώς επίσης και να διερευνήσουμε σε πρακτικό επίπεδο τις δυνατότητες που μας παρέχει η τεχνολογία Resource Directory Technology. Ο επεξεργαστής στον οποιο διεξήχθησαν τα πειράματα είναι ο Intel® Xeon® Processor E5-2630 v4 και τα χαρακτηριστικά του αναγράφονται στον ακόλουθο πίνακα :

	Χαρακτηριστικά
Αρχιτεκτονική	Broadwell
αριθμός πυρήνων	10
αριθμός νημάτων	20
βασική συχνότητα	2.20 GHz
LLC	25 MB (inclusive)
assosiativity	20
DRAM Bandwidth	68.3 GB/sec
CMT - CAT	✓
MBA	✗

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

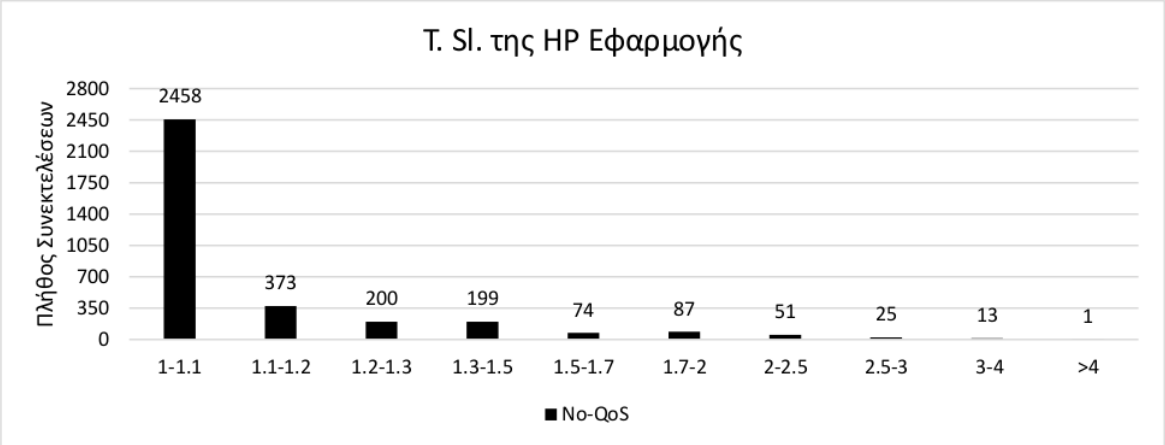
Για την εξακρίβωση και τη μελέτη του προβλήματος της συνεκτέλεσης πολλαπλών εφαρμογών χρησιμοποιούμε μετροπρογράμματα των σουιτών SPEC CPU2006 και PARSEC. Επειδή πολλά χρησιμοποιούνται με παραπάνω από μία είσοδο έχουμε στη διάθεσή μας 59 διαφορετικά μετροπρογράμματα. Αρχικά μελετάμε την απομονωμένη εκτέλεσή τους κατά την οποία η εφαρμογή εκτελείται μόνη της στο σύστημα με στόχο να αντλήσουμε πληροφορίες για τη συμπεριφορά τους. Συγκεκριμένα, με χρήση της τεχνολογίας CMT μετράμε την ευαισθησία των χαρακτηριστικών τους, όπως π.χ. IPC (Instructions Per Cycle), χρόνος εκτέλεσης, LLC Misses, LLC Occupancy, DRAM bandwidth, ως προς διάφορα στατικά τμήματα της LLC. Τα συμπεράσματά μας σε αυτό το στάδιο επιβεβαίωσαν το profiling της εργασίας ([18]), που πραγματοποιήθηκε σε πειραματικό επεξεργαστή διαφορετικής τοπολογίας αλλά ίδιας οικογένειας.

3.2 Χρονική Καθυστέρηση της Υψηλής Προτεραιότητας Εφαρμογής

Στην συνέχεια, δημιουργούμε σχήματα συνεκτελέσεων, συνεκτελώντας ένα μετροπρόγραμμα που χαρακτηρίζουμε ως την υψηλής προτεραιότητας (High Priority, HP) εφαρμογή με εννέα αντίγραφα (instances) του ίδιου ή κάποιου άλλου (Low Priority, LP) με στόχο να ερευνήσουμε την επίδραση αυτών στο μετροπρόγραμμα υψηλής προτεραιότητας που μας ενδιαφέρει. Σημειώνουμε ότι υπάρχει απομόνωση επεξεργαστών, δηλαδή κάθε εφαρμογή είναι προσκολλημένη σε έναν πυρήνα (cpu affinity) και δεν κάνουμε χρήση των hyperthreads. Ο λόγος που εισάγουμε αναγκαστικά τους παραπάνω περιορισμούς είναι γιατί μας ενδιαφέρει να μελετήσουμε τη συνεκτέλεση αποκλειστικά ως προς τον ανταγωνισμό για την LLC χωρίς να υπάρχει ανταγωνισμός για κάποιον άλλο πόρο όπως π.χ. οι πυρήνες ή η αλληλεπίδραση των hyperthreads στις ιδιωτικές caches. Το εν λόγω σχήμα συνεκτέλεσης, κατά το οποίο όλες οι εφαρμογές έχουν εξολοκλήρου πρόσβαση στην LLC, το ονομάζουμε πολιτική No-QoS.

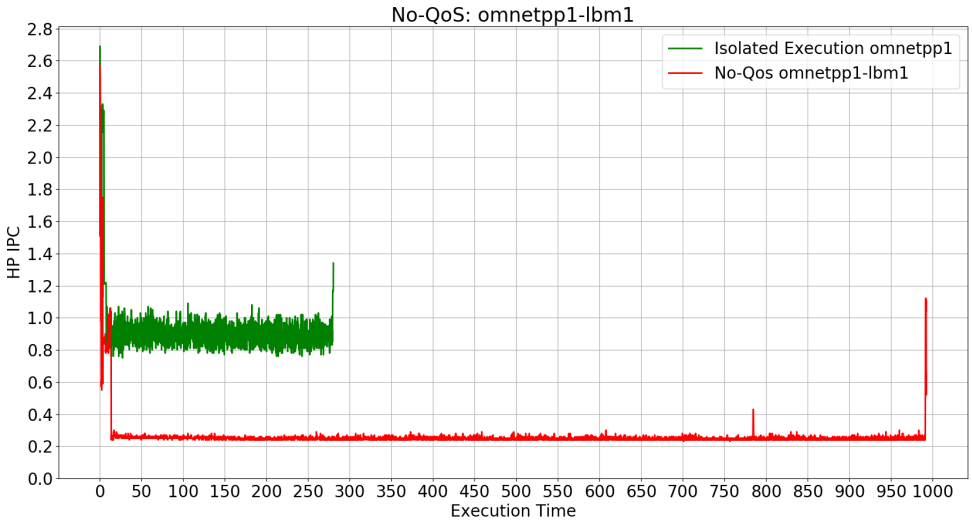
Συνεκτελώντας όλους τους πιθανούς συνδυασμούς ζευγών που προκύπτουν, εξετάζουμε τη σχέση του χρόνου εκτέλεσης της υψηλής προτεραιότητας εφαρμογής των $59 \times 59 = 3481$ συνδυασμών που προέκυψαν, με την απομονωμένη εκτέλεσή της. Συγκεκριμένα, ορίζουμε ως μετρική απόκλισης από την ιδανική περίπτωση, δηλαδή την περίπτωση που η εφαρμογή εκτελείται μόνης της στο σύστημα έχοντας πρόσβαση σε όλη την LLC, τη χρονική καθυστέρηση (Time Slowdown, T.Sl.) που προκύπτει από τον τύπο : $TimeSlowdown = \frac{t_{No-QoS}}{t_{StandAlone}}$, όπου $t_{StandAlone}$, ο χρόνος εκτέλεσης της εφαρμογής υψηλής προτεραιότητας όταν εκτελείται μόνη της στο σύστημα και t_{No-QoS}, ο χρόνος εκτέλεσης της εφαρμογής υψηλής προτεραιότητας όταν εκτελείται με εννέα αντίγραφα κάποιας άλλης εφαρμογής.

Στο Σχήμα 3.1 βλέπουμε την κατανομή των συνεκτελέσεων με βάση την τιμή του T.Sl. της HP εφαρμογής. Παρατηρούμε ότι για το 30% των περιπτώσεων έχουμε σημαντική χρονική καθυστέρηση που κυμαίνεται από 1.1 μέχρι και άνω του 4. Χαρακτηριστικά παραδείγματα μεγάλου και αμελητέου T.Sl. που αναλύουμε στην συνέχεια είναι η συνεκτέλεσή του omntepp1 ως HP με 9 αντίγραφα lbm1 (T.Sl. = 3.6), του astar1 ως HP με 9 αντίγραφα libquantum1 (T.Sl. = 2.3) και του bodytrack1 ως HP με 9 αντίγραφα gromacs1 (T.Sl. = 1.007).



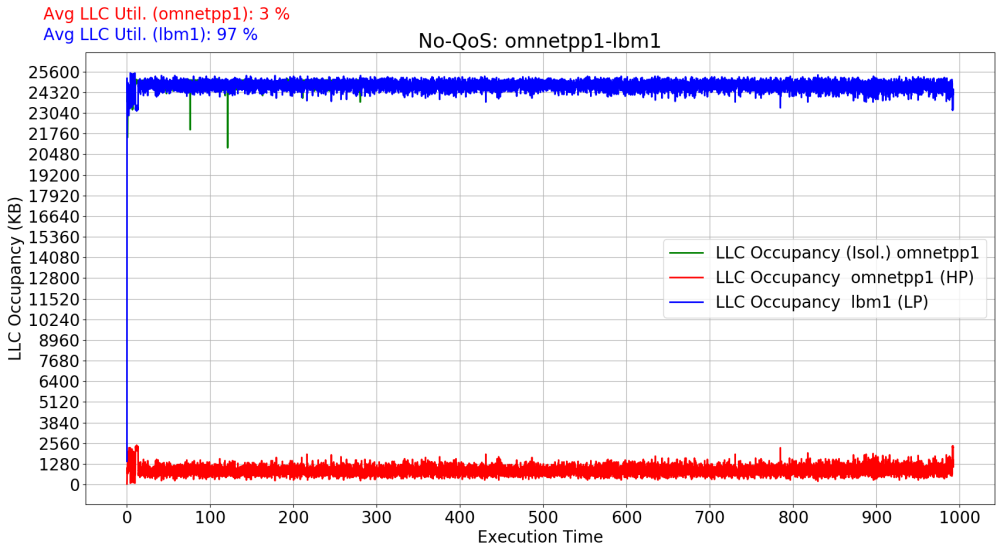
Σχήμα 3.1: Κατανομή workloads με βάση τη χρονική καθυστέρηση κατά τη No-QoS πολιτική

Στο παρακάτω σχήμα αναπαρίσταται το IPC του omnetpp1 ως HP εφαρμογή κατά τη διάρκεια της συνεκτέλεσής του με lbm1 αντίγραφα. Παρατηρούμε, ότι η επίδραση των εννέα LP αντιγράφων lbm1 στο υψηλής προτεραιότητας omnetpp1 είναι σημαντική αφού τριπλασιάζουν σχεδόν τον χρόνο εκτέλεσής του σε σχέση με την απομονωμένη εκτέλεση.



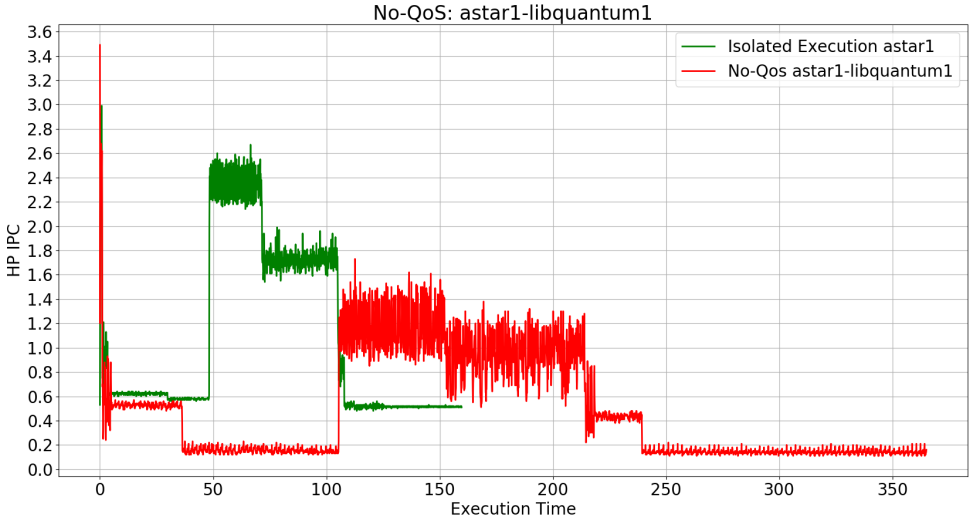
Σχήμα 3.2: Γραφική Παράσταση IPC του omnetpp1 κατά τη συνεκτέλεσή του ως HP με lbm1 ως LP

Στην συνέχεια, παραθέτουμε και τη γραφική παράσταση (Σχήμα 3.3) με τα τμήματα της LLC που δεσμεύουν και χρησιμοποιούν οι εφαρμογές κατά τη διάρκεια της εκτέλεσης. Παρατηρούμε ότι τα εννέα αντίγραφα του lbm1 χρησιμοποιούν σχεδόν όλη την LLC (97%) παραχωρώντας ελάχιστη (3%) στο omnettp1. Δεδομένου ότι, από την ανάλυση των μετροπρογραμμάτων, το omnetpp1 παρουσιάζει μεγάλη ευαισθησία ως προς την LLC, αυτός είναι ο λόγος που δημιουργείται τόσο μεγάλη χρονική καθυστέρηση.

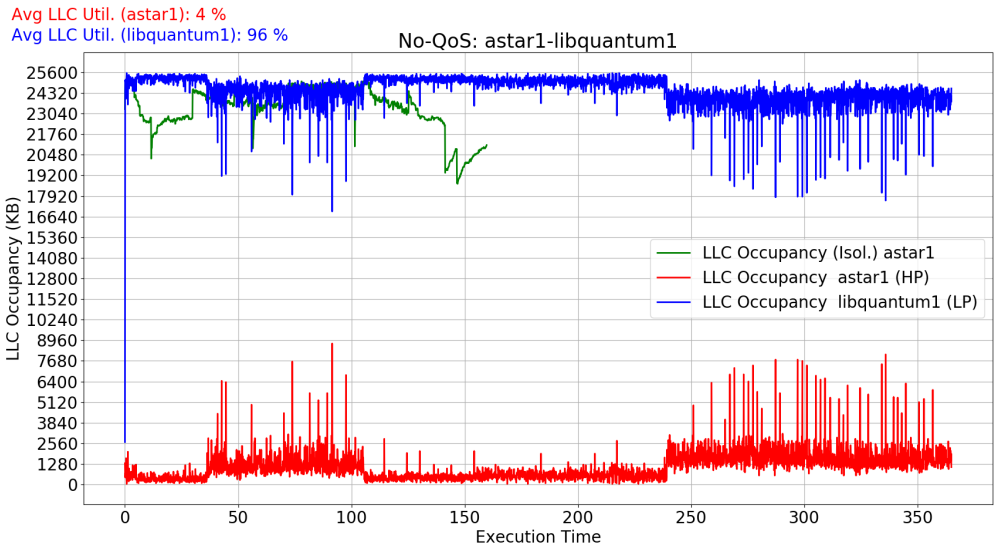


Σχήμα 3.3: Γραφική Παράσταση της χρήσης LLC κατά τη συνεκτέλεση του omnetpp1 ως HP με lbm1 ως LP

Το astar1 είναι ένα μετροπρόγραμμα που χαρακτηρίζεται από πολλές διακριτές διαφορετικές φάσεις. Παρατηρούμε ότι η πρώτη φάση του astar1 δεν απαιτεί μεγάλο LLC τμήμα αφού το IPC κατά τη συνεκτέλεση είναι ελάχιστα μικρότερο από το IPC στην απομονωμένη εκτέλεση, έχοντας LLC λιγότερη από 1 way (Σχήματα 3.4 και 3.5). Ωστόσο, στις επόμενες φάσεις της εκτέλεσης του astar1 παρατηρούμε σημαντική χρονική καθυστέρηση σε συνδυασμό με πολύ χαμηλό ποσοστό χρησιμοποίησης της LLC (4%).



Σχήμα 3.4: Γραφική Παράσταση IPC του astar1 κατά τη συνεκτέλεσή του ως HP με libquantum1 ως LP

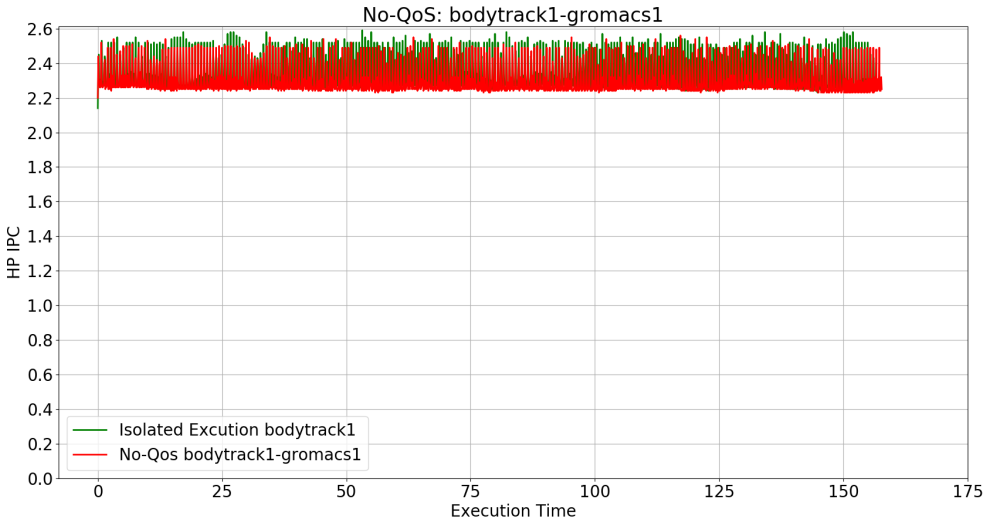


Σχήμα 3.5: Γραφική Παράσταση της χρήσης LLC κατά τη συνεκτέλεση του astar1 ως HP με libquantum1 ως LP

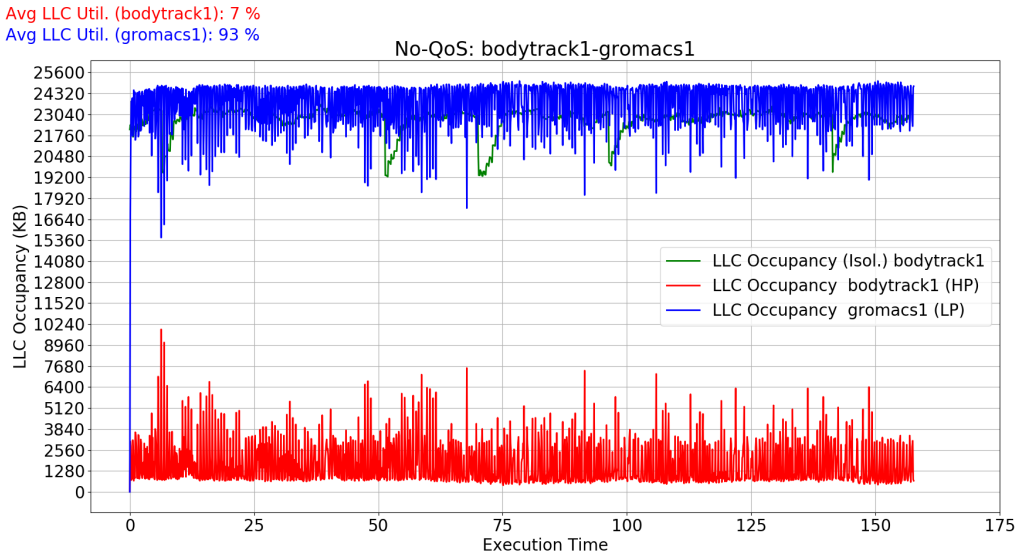
Παρατηρούμε, δηλαδή και πάλι, ότι τα εννέα LP αντίγραφα του libquantum1 χρησιμοποιούν το μεγαλύτερο μέρος της LLC, μη αφήνοντας περιθώρια για το astar1 να εκτελεσθεί ομαλά. Επίσης, το συγκεκριμένο παράδειγμα αποδεικνύει ότι μια εφαρμογή ανάλογα με την υπολογιστική της φάση

αλλά και το φορτίο της έχει κατά την εκτέλεσή της διαφορετικές ανάγκες για LLC.

Στο παράδειγμα που ακολουθεί, θέλουμε να δείξουμε ότι ο ανταγωνισμός για LLC δεν είναι πάντα επιβλαβής για την επίδοση της HP εφαρμογής. Στη συγκεκριμένη περίπτωση, συνεκτελούμε το υψηλής προτεραιότητας bodytrack1 με εννέα αντίγραφα gromacs1. Παρατηρούμε ότι το bodytrack1 με χρησιμοποίηση LLC 7%, δηλαδή κατά μέσο όρο περίπου 1,5 ways = $1,5 \times 1280 = 1920$ MB εκτελείται βέλτιστα σε χρόνο ίδιο με τον χρόνο απομονωμένης εκτέλεσης. Στη συγκεκριμένη συνεκτέλεση σημαντικό ρόλο έπαιξε και η επιλογή του gromacs1 ως την εφαρμογή μικρής προτεραιότητας καθώς, από την ανάλυση που έχουμε κάνει, είναι εφαρμογή με ελάχιστες απαιτήσεις LLC (< 3 ways).



Σχήμα 3.6: Γραφική Παράσταση IPC του bodytrack1 κατά τη συνεκτέλεσή του ως HP με gromacs1 ως LP



Σχήμα 3.7: Γραφική Παράσταση της χρήσης LLC κατά τη συνεκτέλεση του bodytrack1 ως HP με gromacs1 ως LP

Η χρονική καθυστέρηση που παρατηρήσαμε στις συνεκτελέσεις οφείλεται στον ανταγωνισμό για cache και δεν εξαρτάται μονομερώς από τη φύση αυτής καθεαυτής της εφαρμογής, όπως για παράδειγμα, την ευαισθησία της ως προς την cache, αλλά και από τη φύση των συνεκτελούμενων εφαρμογών. Στην συνεκτέλεση του bodytrack1 με εννέα αντίγραφα gromacs1 (Σχήμα 3.6) είδαμε ότι μη έντονες ως προς την cache εφαρμογές δεν ”μολύνουν” την LLC και δεν δημιουργούν σοβαρό ανταγωνισμό για αυτήν. Αντίθετα, στην συνεκτέλεση του astar1 ως HP με libquantum1 ως LP βλέπουμε ότι οι LP εφαρμογές είναι επιθετικές ως προς την cache. Μάλιστα, επειδή η LLC στην συγκεκριμένη αρχιτεκτονική επεξεργαστών (Broadwell) είναι περιεκτική (inclusive), δηλαδή υποχρεωτικά περιέχει πάντα τα στοιχεία των L1 & L2 ιδιωτικών caches, όταν μια εφαρμογή ”διώξει” ένα cache block που ανήκει στην ίδια ή σε κάποια άλλη εφαρμογή, τότε αυτόματα αυτό το cache block γίνεται μη έγκυρο (invalid) και στα ανώτερα επίπεδα κρυφών μνημών. Επομένως, ο ανταγωνισμός για την LLC επηρεάζει και τις ιδιωτικές caches των πυρήνων και κατ’ επέκταση, βλάπτει τον ντετερμινισμό της εκτέλεσης και την επίδοση της HP εφαρμογής.

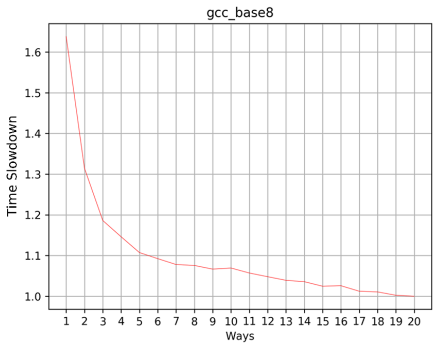
3.3 Κατηγορίες Συνεκτελέσεων

3.3.1 Πολιτικές No-Qos και CT-QoS

Σε αυτή την ενότητα αξιολογούμε και συγκρίνουμε δύο πολιτικές συνεκτέλεσης. Η πρώτη πολιτική, No-QoS, χρησιμοποιήθηκε και στην προηγούμενη ενότητα για την εύρεση της χρονικής καθυστέρησης της HP εφαρμογής και αφορά την συνεκτελέση εφαρμογών χωρίς κανένα καταμερισμό της cache στις εφαρμογές. Κατά συνέπεια, η επίδοση εξαρτάται αποκλειστικά από τη φύση και την αλληλεπίδραση των συνεκτελούμενων εφαρμογών. Σε αυτή την πολιτική, όλα τα συνεκτελούμενα μετροπρογράμματα μοιράζονται ολόκληρη την LLC.

Η χρονική καθυστέρηση που προκαλείται λόγω ανταγωνισμού στην HP εφαρμογή κατά τη συνεκτέλεση δημιουργεί την ανάγκη για απομόνωση του τμήματος της LLC που δεσμεύει η συγκεκριμένη εφαρμογή.

Με βάση τα profiles των εφαρμογών, ο χρόνος εκτέλεσης παραμένει ουσιαστικά ίδιος είτε παραχωρήσουμε στο HP το 95% είτε το 100% της LLC. Για παράδειγμα, όπως βλέπουμε στο Σχήμα 3.8 το T.Sl. για το gcc_base8 είναι μηδαμινό είτε του παραχωρήσουμε τα 19 είτε και τα 20 ways της LLC. Φυσικά, μια εφαρμογή μπορεί να πετυχαίνει βέλτιστη εκτέλεση και με πολύ λιγότερο χώρο στην cache, όμως είναι σίγουρο ότι τα 19 ways επαρκούν ώστε η εφαρμογή να εκτελεστεί σε χρόνο ίδιο με τον χρόνο απομονωμένης εκτέλεσης. Άλλωστε, δεδομένης της συνεκτέλεσης, το 95% της LLC, δηλαδή τα 19 από τα 20 ways, είναι το μεγαλύτερο τμήμα της cache που μπορούμε να παραχωρήσουμε στην HP εφαρμογή.

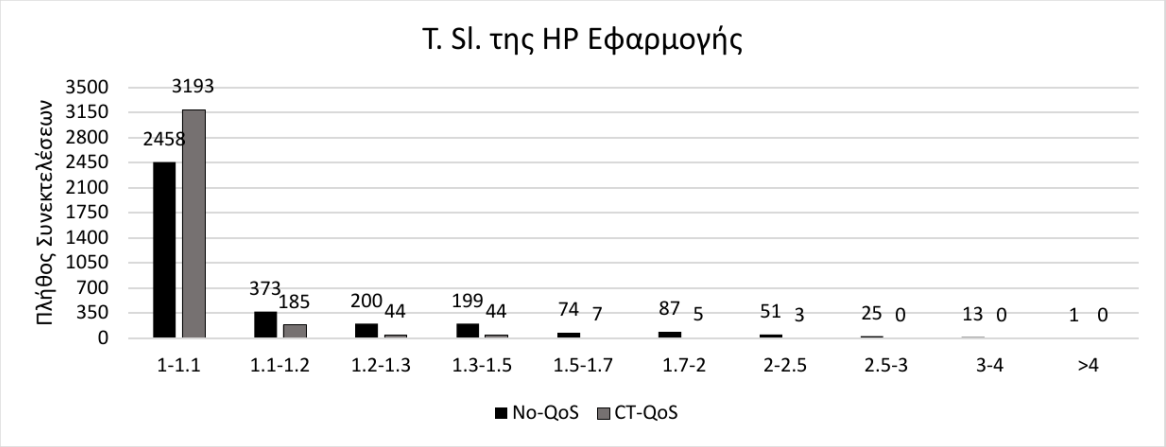


Σχήμα 3.8 Χρονική Καθυστέρηση του gcc_base8 ανά τμήμα LLC (στατική δέσμευση)

Επομένως, δοκιμάζουμε μια δεύτερη πολιτική εκτέλεσης, τη CT-QoS στην οποία κάνουμε στατικό διαμερισμό της LLC ανάμεσα στις εφαρμογές. Πιο αναλυτικά, παραχωρούμε το 95% της LLC,

δηλαδή τα 19 ways κάθε cache set στην υψηλής προτεραιότητας εφαρμογής και το υπόλοιπο 1 way, 5% της LLC, στις χαμηλής προτεραιότητας εφαρμογές. Η πολιτική CT-QoS εξασφαλίζει ότι η υψηλής προτεραιότητας εφαρμογή δεν θα παρουσιάσει μείωση στην επίδοσή της λόγω μη επαρκούς τμήματος LLC. Αναμένουμε γενικά ότι (α) η CT-QoS θα βελτιώσει τη χρονική καθυστέρηση που εμφάνισε η υψηλής προτεραιότητας εφαρμογή κατά τη συνεκτέλεση (No-QoS), αφού είναι πλήρως απομονωμένη ως προς την LLC και, δεδομένης της συνεκτέλεσης, έχει στη διάθεσή της το μεγαλύτερο δυνατό τμήμα της (19 ways) και (β) η CT-QoS, θα άρει την κακή ποιότητα υπηρεσίας λόγω μηδαμινού, πλέον, ανταγωνισμού στην LLC και θα εξασφαλίσει επίδοση ίση με την επίδοση της απομονωμένης εκτέλεσης.

Στο Σχήμα 3.9 βλέπουμε την κατανομή των συνεκτελέσεων βάσει της τιμής του T.Sl. της HP εφαρμογής κατά τις No-QoS και CT-QoS πολιτικές ($\frac{t_{CT-QoS}}{t_{StandAlone}}$) .



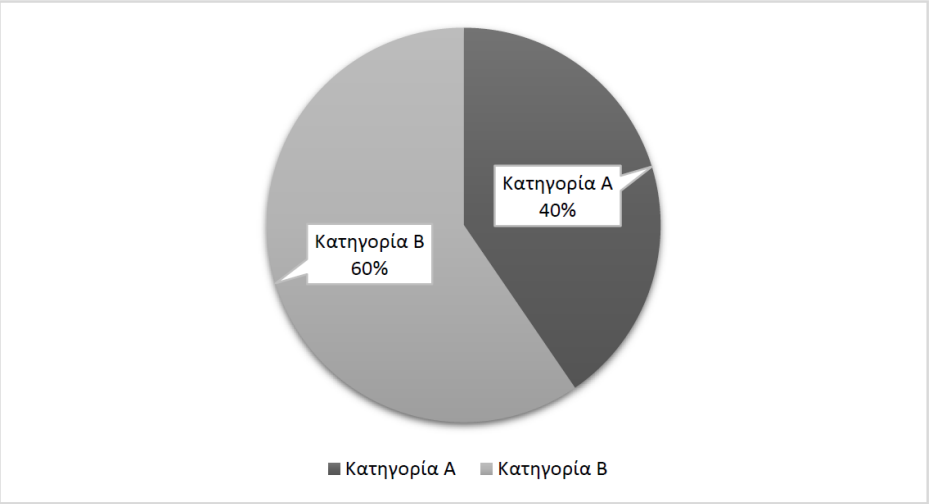
Σχήμα 3.9: Κατανομή workloads με βάση τη χρονική καθυστέρηση κατά τις No-QoS και CT-QoS πολιτικές

Παρατηρούμε ότι η CT-QoS πολιτική βελτιώνει το T.Sl. συγκριτικά με την No-QoS αλλά υπάρχουν περιπτώσεις που δεν το εξαλείφει τελείως. Αυτό οφείλεται ενδεχομένως και στον ανταγωνισμό για άλλους διαμοιραζόμενους πόρους όπως για παράδειγμα το memory bandwidth, τον memory controller, τους hardware prefetchers της LLC, το network bandwidth, και άλλοι, τους οποίους προς το παρόν δεν μπορούμε να διαχειριστούμε. Στόχος της εργασίας μας είναι να μελετήσουμε αποκλειστικά την επίδραση της LLC στη συνεκτέλεση.

Τα συμπεράσματα από την ανάλυση των παραπάνω πολιτικών μας οδηγούν στο σχηματισμό δύο κατηγοριών συνεκτελέσεων (Πίνακας 3.2). Στην κατηγορία Α, εντάσσονται όλοι οι συνδυασμοί που υπακούουν στον διαισθητικό γενικό κανόνα ότι η εκτέλεση μιας εφαρμογής ευνοείται από μεγαλύτερο τμήμα LLC και επομένως, εδώ εντάσσονται όλες περιπτώσεις όπου $t_{CT-QoS} < t_{No-Qos}$, όπου t ο χρόνος εκτέλεσης της HP εφαρμογής κατά την εκάστοτε πολιτική. Στην κατηγορία Β εντάσσονται όλοι οι συνδυασμοί για τους οποίους η πολιτική CT-QoS είναι ανεπιθύμητη, δηλαδή, δεν προσφέρει κάτι έναντι της πολιτικής No-QoS ή ακόμα και χειροτερεύει την επίδοση της HP εφαρμογής, δηλαδή περιλαμβάνονται οι συνεκτελέσεις για τις οποίες $t_{CT-QoS} \geq t_{No-QoS}$.

	Κατηγορίες	
	Κατηγορία Α	Κατηγορία Β
Αριθμός συνεκτελέσεων	1408	2073

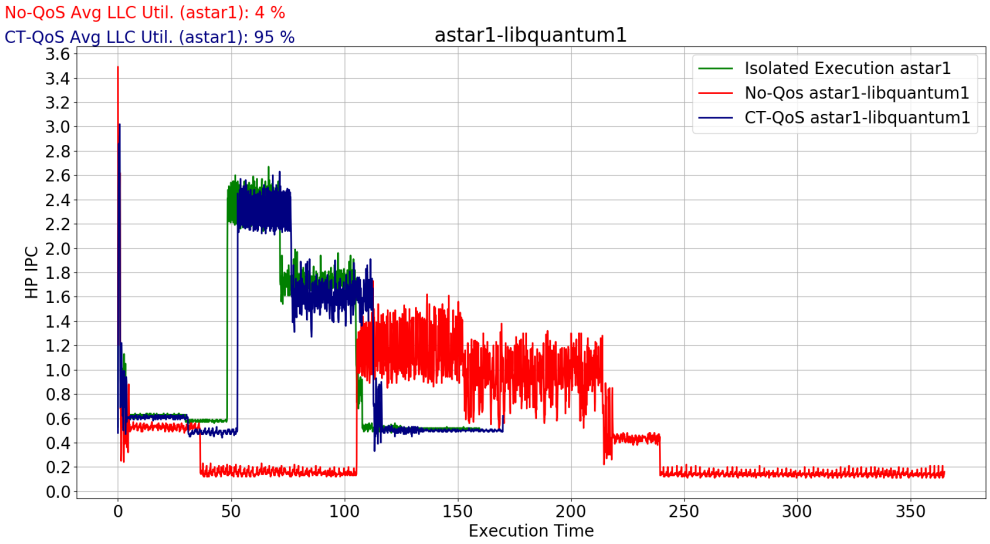
Πίνακας 3.2: Κατηγορίες των συνεκτελέσεων βάσει της σχέσης των χρόνων t_{CT-QoS} και t_{No-QoS}



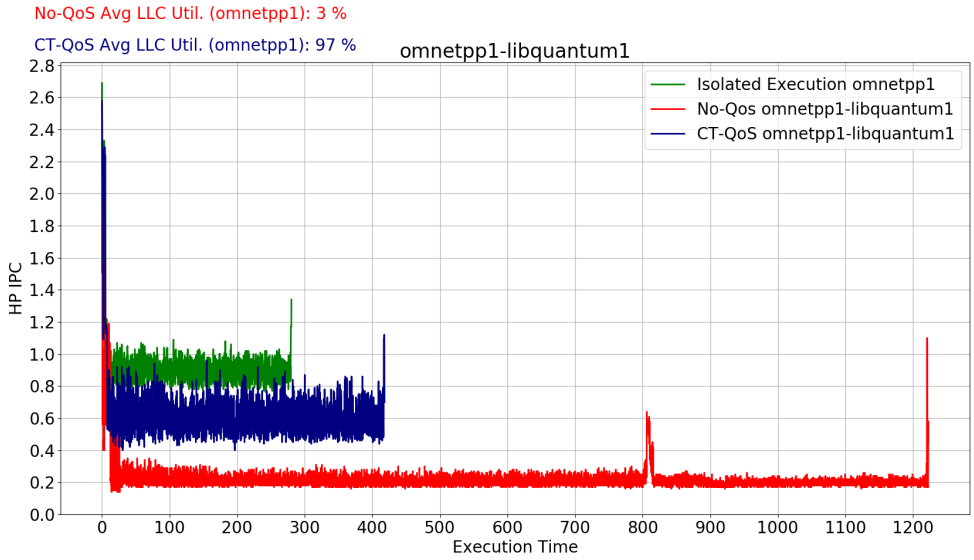
Σχήμα 3.10

3.3.2 Παραδείγματα Συνεκτελέσεων Κατηγοριας Α

Όπως έχουμε αναφέρει, στην κατηγορία Α ανήκουν οι περιπτώσεις στις οποίες η CT-QoS πολιτική ευνοεί την εκτέλεση της εφαρμογής με υψηλή προτεραιότητα. Χαρακτηριστικά παραδείγματα αυτής της κατηγορίας είναι οι συνεκτελέσεις astar1-libquantum1 (T.Sl. = 2.24) και omnetpp1-libquantum1 (T.Sl. = 4.37). Στην πρώτη συνεκτέλεση η CT-QoS πολιτική βελτιώνει την επίδοση του astar1 κατά 53.7% σε σχέση με την επίδοσή του κατά τη No-QoS πολιτική. Αντίστοιχα στη δεύτερη συνεκτέλεση, η επίδοση του omnetpp1 βελτιώνεται κατά 66.8% σε σχέση με την επίδοσή του κατά τη No-QoS πολιτική.



Σχήμα 3.11: Γραφική Παράσταση IPC του astar1 κατά τη συνεκτέλεσή του ως HP με libquantum1 ως LP

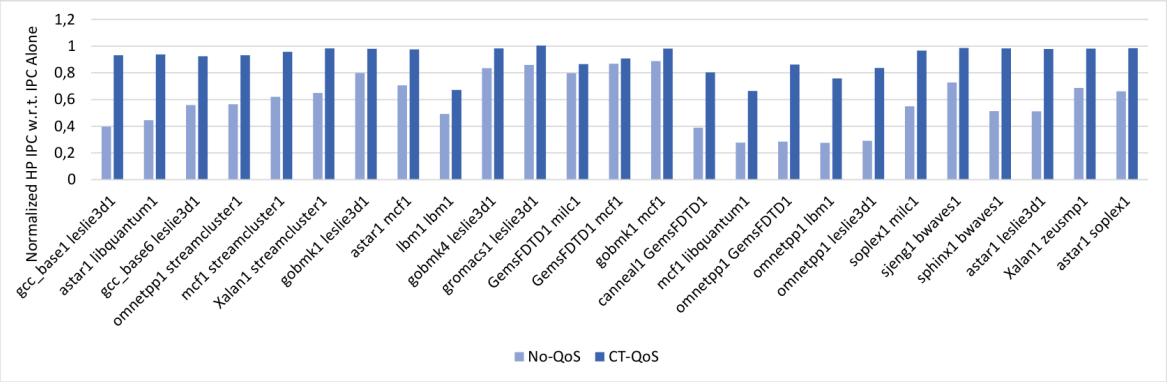


Σχήμα 3.12: Γραφική Παράσταση IPC του omnetpp1 κατά τη συνεκτέλεσή του ως HP με libquantum1 ως LP

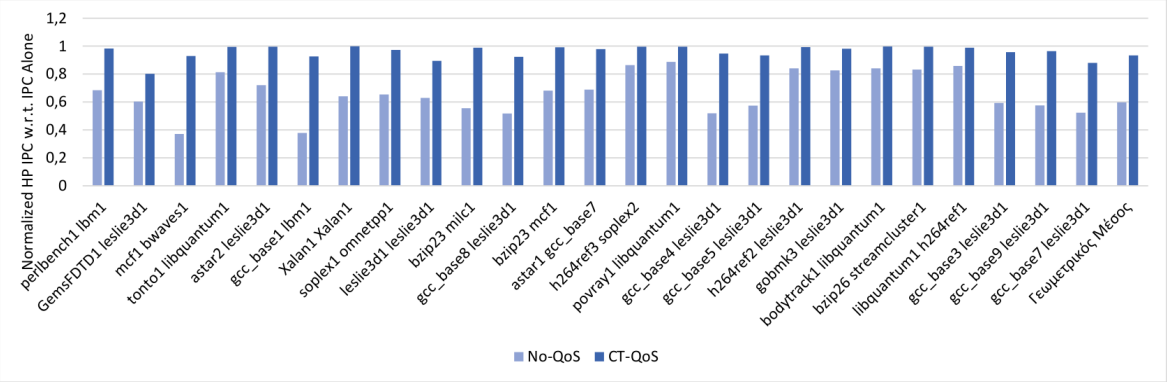
Αναφορικά με την συνεκτέλεση του astar1 ως HP με libquantum1 ως LP, παρατηρούμε ότι κατά τη CT-QoS πολιτική το astar1 σημειώνει μόνο 2% μεγαλύτερο χρόνο εκτέλεσης από το χρόνο της απομονωμένης εκτέλεσής του.

Ακολουθούν τα σχήματα με τη χρονική καθυστέρηση κατά τις πολιτικές No-QoS και CT-QoS 50 συνεκτελέσεων της Κατηγορίας Α. Επιλέξαμε περιπτώσεις από όλες τις τιμές T.Sl. δίνοντας μεγαλύτερη έμφαση σε αυτές με μεγάλη χρονική καθυστέρηση κατά τη No-QoS και μεγάλη βελτίωση κατά τη CT-QoS. Επιπλέον, προσπαθήσαμε να παρουσιάσουμε όσο το δυνατόν μεγαλύτερη ποικιλία workloads ως HP εφαρμογή. Επομένως, θεωρούμε τους 50 αυτούς συνδυασμούς αντιπροσωπευτικούς για το σύνολο των περιπτώσεων της Κατηγορίας Α.

Στα Σχήματα 3.13 και 3.14 αναπαρίσταται η μέση τιμή IPC της HP εφαρμογής κατά τις πολιτικές No-QoS και CT-QoS, όπως μετρήθηκε στους διάφορους συνδυασμούς συνεκτελέσεων, κανονικοποιημένη ως προς το μέσο IPC της εφαρμογής κατά την απομονωμένη εκτέλεση.



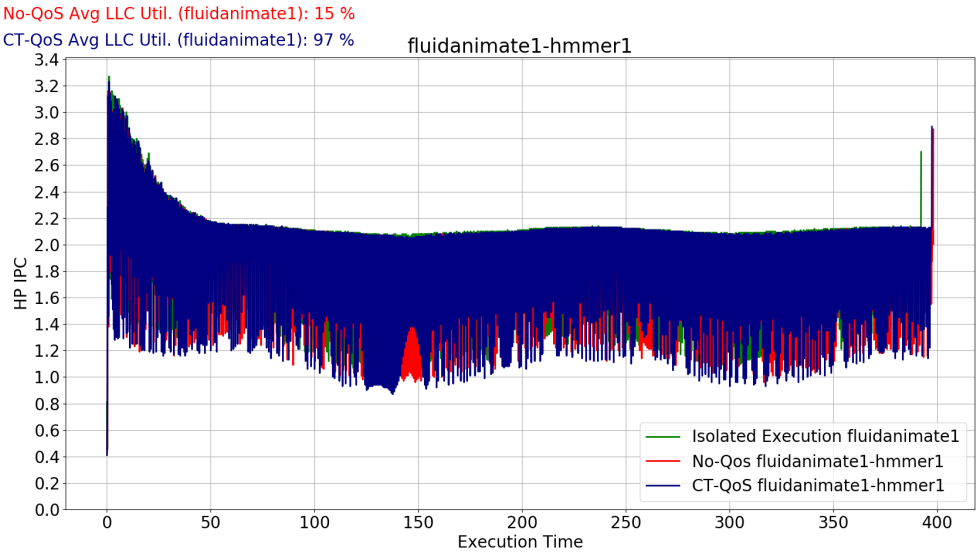
Σχήμα 3.13: IPC HP Εφαρμογής συνεκτελέσεων Κατ. Α κανονικοποιημένη ως προς το IPC κατά την απομονωμένη εκτέλεσής της (I)



Σχήμα 3.14: IPC HP Εφαρμογής συνεκτελέσεων Κατ. Α κανονικοποιημένη ως προς το IPC κατά την απομονωμένη εκτέλεσής της (II)

3.3.3 Παραδείγματα Συνεκτελέσεων Κατηγοριας Β

Στο ακόλουθο παράδειγμα η πολιτική No-QoS είναι ισοδύναμη της CT-QoS. Σε αντίθεση με την Κατηγορία Α, όπου παραχωρώντας στην HP εφαρμογή το μέγιστο δυνατό τμήμα LLC συνεπάγεται και βέλτιστη εκτέλεση, στην συγκεκριμένη περίπτωση, η παραχώρηση μέγιστου τμήματος LLC δεν ενδείκνυται, αφού βλάπτει την εκτέλεση των LP εφαρμογών (Πίνακας 3.3), χωρίς να προσφέρει καλύτερη εκτέλεση στην εφαρμογή υψηλής προτεραιότητας. Για παράδειγμα, για την βέλτιστη εκτέλεση του fluidanimate1 με εννέα αντίγραφα hmmer1 απαιτείται μόνο το 15% της LLC δηλαδή 3 ways.

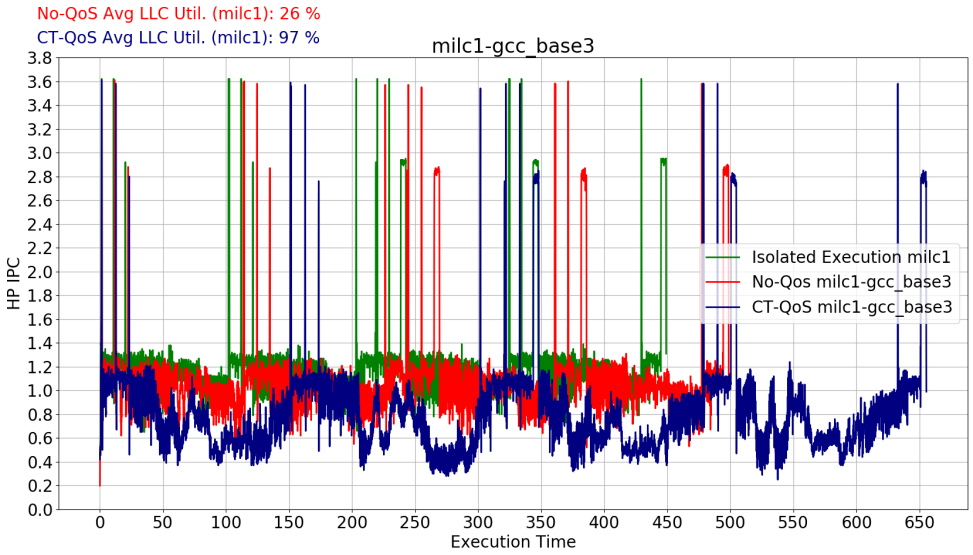


Σχήμα 3.15: Γραφική Παράσταση IPC του fluidanimate1 κατά τη συνεκτέλεσή του ως HP με hmmer1 ως LP

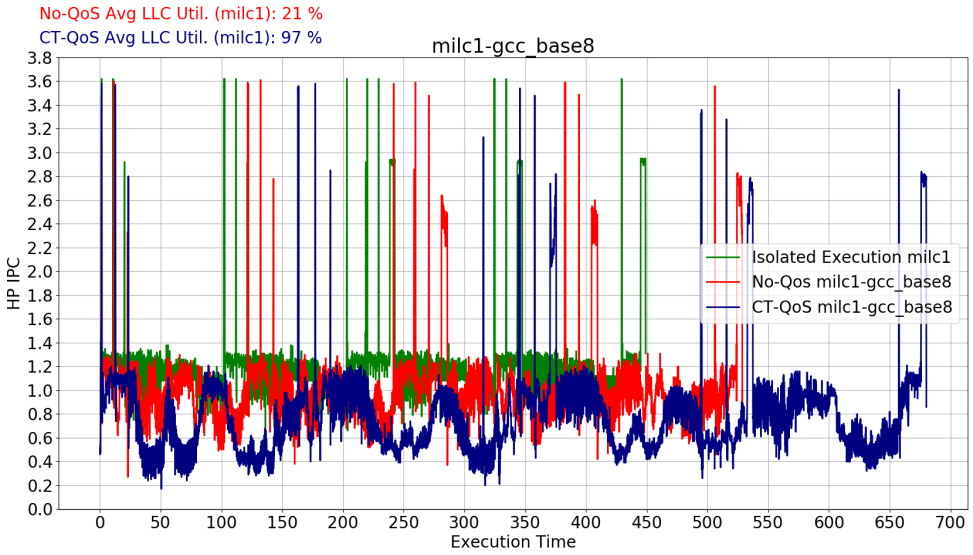
fluidanimate1-hmmer1	Avg IPC των 9 LP hmmer1 αντιγράφων
No-QoS	2.62
CT-QoS	0.66
Ποσοστό Μείωσης IPC	74%

Πίνακας 3.3: Επίδοση των LP hmmer1 κατά την συνεκτέλεσή τους με HP fluidanimate1

Οι συνεκτελέσεις που ακολουθούν παρουσιάζουν ενδιαφέρον καθώς δεν υπακούουν στον γενικό κανόνα που συνδέει την βέλτιση εκτέλεση μιας εφαρμογής με μεγάλο τμήμα LLC, αλλά αντιθέτως, η παραχώρηση του μέγιστου δυνατού τμήματος LLC βλάπτει την επίδοση της υψηλής προτεραιότητας εφαρμογής. Χαρακτηριστικά παραδείγματα αποτελούν οι συνεκτελέσεις των milc1-gcc_base3 και milc1-gcc_base8. Σε αυτές τις περιπτώσεις, η No-QoS πολιτική αποδεικνύεται καλύτερη της CT-QoS με την υψηλής προτεραιότητας εφαρμογή να εκτελείται κατά 32% γρηγορότερα. Και στις δύο περιπτώσεις, παρατηρούμε ότι, κατά τη No-QoS εκτέλεση, η HP εφαρμογή αξιοποιεί περίπου το 20%-25% της συνολικής διαθέσιμης LLC και εκτελείται καλύτερα σε αντίθεση με τη CT-QoS πολιτική που της παραχωρεί το 95%. Επομένως, η πτώση στην επίδοση δεν οφείλεται στο διαθέσιμο χώρο στην LLC.

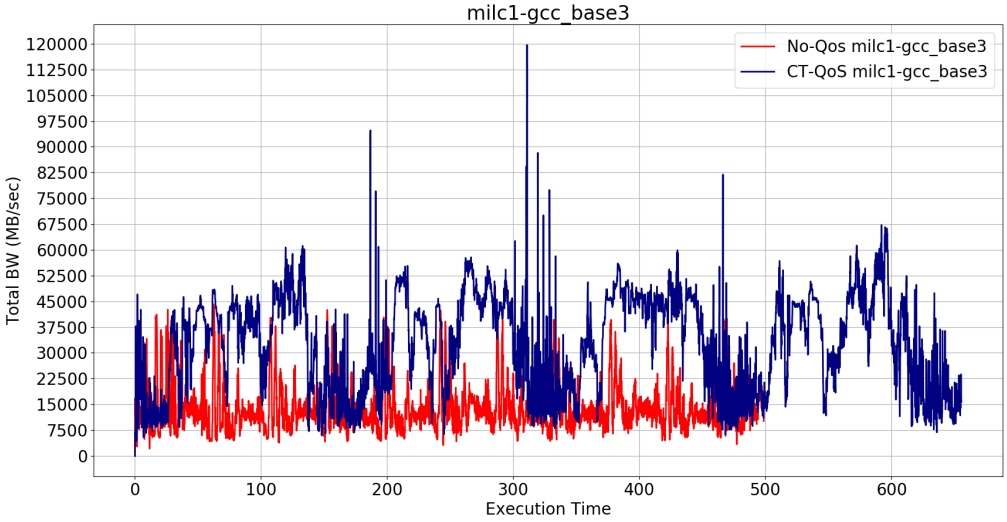


Σχήμα 3.16: Γραφική Παράσταση IPC του milc1 κατά τη συνεκτέλεσή του ως HP με gcc_base3 ως LP

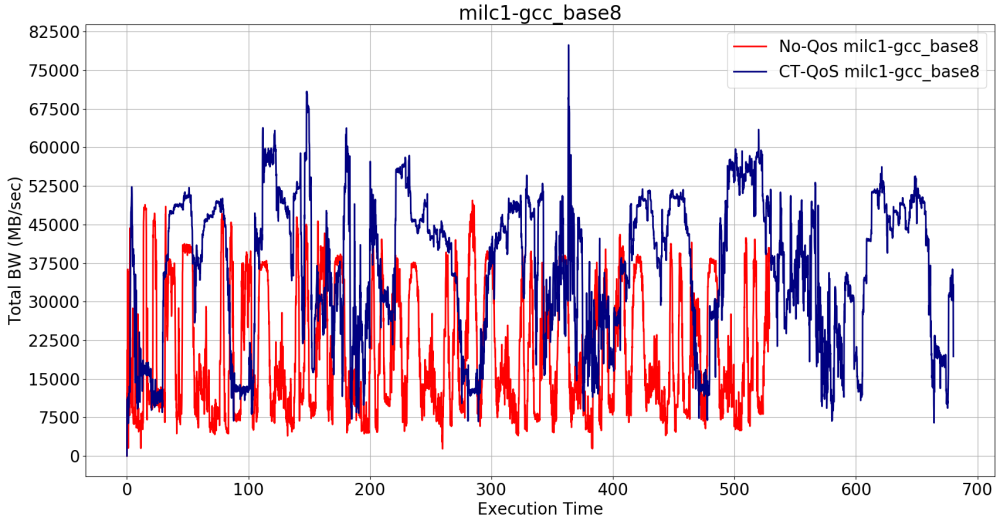


Σχήμα 3.17: Γραφική Παράσταση IPC του milc1 κατά τη συνεκτέλεσή του ως HP με gcc_base8 ως LP

Η χρονική καθυστέρηση που παρατηρείται στην CT-QoS πολιτική οφείλεται στον κορεσμό που συμβαίνει στο DRAM bandwidth. Αναλυτικότερα, η υψηλής προτεραιότητας εφαρμογή λαμβάνει 19 ways και 1 way παραχωρείται στις εννέα άλλες εφαρμογές. Ωστόσο, το 1 way προφανώς δεν καλύπτει τις ανάγκες τους και αναγκάζονται φέρνουν συχνά δεδομένα από τη μνήμη υπερχρησιμοποιώντας τον δίαυλο δεδομένων. Κατά συνέπεια, ενώ το milc1 έχει επαρκές τμήμα LLC καθυστερεί να ικανοποιήσει τα αιτήματά του προς τη μνήμη λόγω κορεσμού στο bandwidth που προκαλούν οι συνεκτελούμενες εφαρμογές. Παραθέτουμε τις γραφικές του συνολικού BW (MB/sec) κατά τη διάρκεια εκτέλεσης και των δύο περιπτώσεων.



Σχήμα 3.18: Γραφική Παράσταση συνολικού bandwidth κατά τη συνεκτέλεση του milc1 ως HP με gcc_base3 ως LP



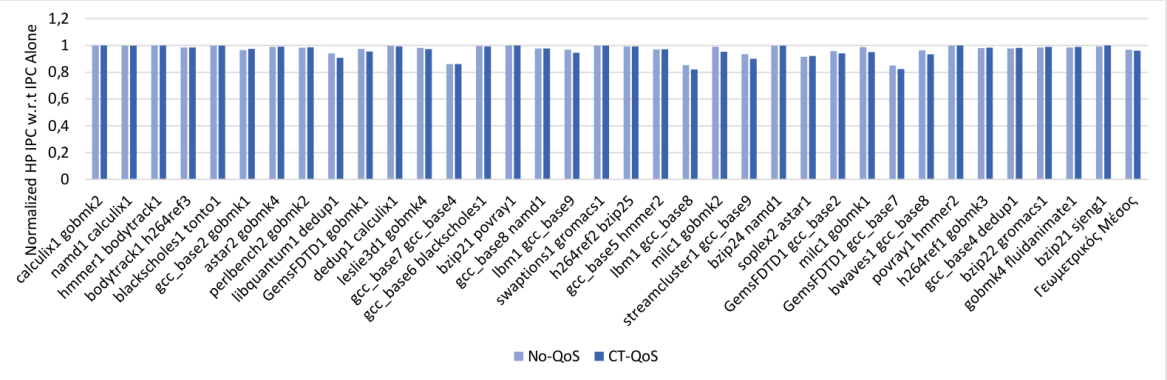
Σχήμα 3.19: Γραφική Παράσταση συνολικού bandwidth κατά τη συνεκτέλεση του milc1 ως HP με gcc_base8 ως LP

Ο πίνακας 3.4 επιβεβαιώνει τα συμπεράσματά μας. Παρατηρούμε ότι ενώ το μέσο bandwidth που χρησιμοποιεί η HP εφαρμογή στη No-QoS πολιτική είναι περίπου 45% μεγαλύτερο από το αντίστοιχο bandwidth στη CT-QoS πολιτική, το συνολικό μέσο bandwidth της CT-QoS πολιτικής είναι σχεδόν διπλάσιο και στις δύο περιπτώσεις. Επομένως, είναι προφανές ότι στην CT-QoS πολιτική οι LP εφαρμογές καταναλώνουν πολύ περισσότερο bandwidth λόγω μη επαρκούς τμήματος LLC.

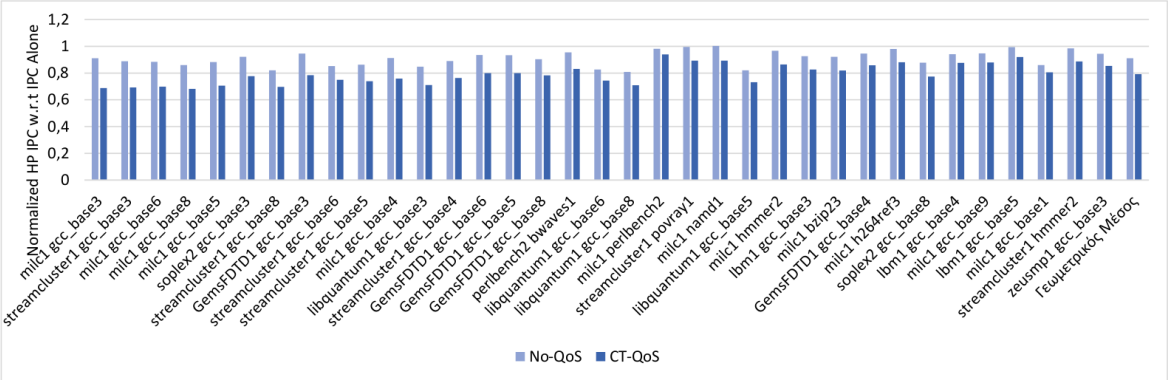
milc1-gcc_base3	Avg Total BW (MB/sec)	Avg HP BW (MB/sec)
No-QoS	13677.76	5228.56
CT-QoS	32486.32	3559.43
milc1-gcc_base8	Avg Total BW (MB/sec)	Avg HP BW (MB/sec)
No-QoS	20152.54	4919.42
CT-QoS	36583.86	3391.66

Πίνακας 3.4: Τιμή DRAM bandwidth ανά πολιτική εκτέλεσης των συνδυασμών milc1-gcc_base3 και milc1-gcc_base8

Αναφέρουμε 70 παραδείγματα της Κατηγορίας Β τα οποία είναι αντιπροσωπευτικά για όλους τους συνδυασμούς της Κατηγορίας Β. Από αυτά, τα 35 παρουσιάζουν ίδιο χρόνο εκτέλεσης στις πολιτικές No-QoS και CT-QoS και τα άλλα 35 εκτελούνται καλύτερα στην No-QoS πολιτική από τη CT-QoS και στα οποία παρατηρούμε κορεσμό στο bandwidth. Στα Σχήματα 3.20 και 3.21 παρουσιάζουμε κανονικοποιημένη την μέση τιμή IPC της HP εφαρμογής ως προς την μέση τιμή IPC της ίδιας εφαρμογής κατά την απομονωμένη εκτέλεσή της.



Σχήμα 3.20: IPC HP Εφαρμογής συνεκτελέσεων Κατ. Β κανονικοποιημένη ως προς το IPC κατά την απομονωμένη εκτέλεσής της (I)



Σχήμα 3.21: IPC HP Εφαρμογής συνεκτελέσεων Κατ. Β κανονικοποιημένη ως προς το IPC κατά την απομονωμένη εκτέλεσής της (II)

3.4 Συμπεράσματα

Η παραπάνω κατηγοριοποίηση αποδεικνύει ότι παρόλο που στην πλειοψηφία των περιπτώσεων ένας στατικός διαμοιρασμός της LLC, όμοιος με αυτόν της CT-QoS πολιτικής, θα έλυνε το πρόβλημα ανταγωνισμού της LLC κατά τη συνεκτέλεση, μια τέτοια πολιτική δεν είναι μονόδρομος, αφού σε συνεκτελέσεις της Κατηγορίας Β λειτουργεί αρνητικά στην επίδοση της HP εφαρμογής. Επιπλέον, ένα από τα μειονεκτήματα της CT-QoS πολιτικής είναι ότι καταστρέφει ουσιαστικά την επίδοση των LP εφαρμογών ακόμα και στις περιπτώσεις που η HP εφαρμογή δεν έχει ανάγκη όλο τον χώρο cache που της παρέχεται. Ενδεικτικά αναφέρουμε το ποσοστό χρονικής καθυστέρησης σε σχέση με τη απομονωμένη εκτέλεση για τις χαμηλής προτεραιότητας εφαρμογές κάποιων περιπτώσεων και των δύο κατηγοριών. Στον Πίνακα 3.5 συμπεραίνουμε ότι η εκτέλεση CT-QoS δημιουργεί σημαντικά μεγάλη επιβράδυνση στις LP εφαρμογές.

Συνδυασμοί	HP Slowdown	LP Slowdown
gromacs1-leslie3d1	98%	36%
GemsFDTD1 leslie3d1	81%	22%
astar1 mcf1	97%	29%
astar1 libquantum1	93%	40%

Πίνακας 3.5: Παραδείγματα επιβράδυνσης LP εφαρμογών στην πολιτική CT-QoS

Μάλιστα, σε πολλές περιπτώσεις μεγαλύτερο τμήμα cache στην HP εφαρμογή δεν προσφέρει ουσιαστική βελτίωση. Σε αυτή την περίπτωση, η εύρεση του απαραίτητου τμήματος LLC είναι σημαντική γιατί οι LP εφαρμογές θα έχουν μεγαλύτερο τμήμα cache από αυτό που τους παραχωρεί η CT-QoS πολιτική (1 way) και επομένως χωρίς να χειροτερεύσει η επίδοση της HP εφαρμογής, θα έχουμε καλύτερη επίδοση των LP εφαρμογών. Όπως αναφέρουμε και στο προηγούμενο κεφάλαιο, η εύρεση της ιδανικής στατικής πολιτικής απαιτεί εξωτερική μελέτη της εφαρμογής (offline profiling) ή πρωτύτερη γνώση του είδους της εφαρμογής. Είναι εύλογη λοιπόν η αναζήτηση εύρεσης ενός αλγορίθμου, ο οποίος δυναμικά να αποφασίζει για το τμήμα μνήμης που θα παραχωρήσει στην HP εφαρμογή με κύρια απαίτηση πάντα την ομαλή και βέλτιστη εκτέλεσή της.

Κεφάλαιο 4

Αλγόριθμος Δυναμικής Παραχώρησης Μνήμης

Μέχρι αυτό το σημείο έχουμε δει και αναλύσει την παρεμβολή και τους περιορισμούς της συνεκτέλεσης στην προστασία της επίδοσης και της παροχής ποιότητας υπηρεσίας της εφαρμογής που μας ενδιαφέρει. Επιπλέον, έχουμε καταλήξει ότι μια στατική πολιτική διαμοιρασμού της LLC, ευνοϊκή ως προς την υψηλής προτεραιότητας εφαρμογή, είναι ακατάλληλη σε κάποιες περιπτώσεις. Δεδομένου ότι δεν γνωρίζουμε πληροφορίες σχετικές με το είδος των εφαρμογών της συνεκτέλεσης, είναι αδύνατον να καθορίσουμε με κάποιο τρόπο ποια από τις δύο πολιτικές της προηγούμενης ενότητας είναι η κατάλληλη για να εφαρμοστεί. Γίνεται, έτσι, επιτακτική η ανάγκη σχεδιασμού και υλοποίησης ενός μηχανισμού ο οποίος, εκμεταλλευόμενος τις τεχνολογίες Intel CMT-CAT, ελέγχει δυναμικά σε χρόνο εκτέλεσης την επίδοση και την πρόσβαση στην LLC μιας εφαρμογής με υψηλή προτεραιότητα, με στόχο να προστατεύσει την εκτέλεσή της σε ένα περιβάλλον πλήρους συνεκτέλεσης. Στις επόμενες ενότητες παρουσιάζουμε, αναλύουμε και αξιολογούμε τον μηχανισμό BW-QoS που υλοποιήσαμε. Επιπλέον, συγκρίνουμε την υλοποίησή μας με την προγενέστερη υλοποίηση DCP-QoS ([4], [18]).

4.1 Δομή και Λειτουργία του Μηχανισμού BW-QoS

Στόχος αυτού του μηχανισμού είναι να προστατεύσει την εκτέλεση της υψηλής προτεραιότητας εφαρμογής όταν αυτή συνεκτελείται με εννέα άλλες. Ο μηχανισμός εξασφαλίζει την απομόνωση της HP εφαρμογής από τις άλλες στην LLC παραχωρώντας της τον κατάλληλο χώρο, εκμεταλλευόμενος την τεχνολογία Intel CAT. Επιπλέον, δρα δυναμικά, δηλαδή αποφασίζει κατά τη διάρκεια της συνεκτέλεσης αν και τι μέρος της LLC θα αποδεσμεύσει ή θα παραχωρήσει στην εφαρμογή με υψηλή προτεραιότητα.

Πιο συγκεκριμένα, ο μηχανισμός μετράει και ελέγχει τις μεταβολές στο IPC της υψηλής προτεραιότητας εφαρμογής για να έχει αίσθηση για την επιδοσή της. Απόκλιση μεγαλύτερη από το 5% της προηγούμενης μέτρησης σηματοδοτεί μη σταθερό IPC. Επιπλέον, είναι απαραίτητο και μετράται το συνολικό bandwidth όλων των πυρήνων στους οποίους έχουν δρομολογηθεί οι εφαρμογές έτσι ώστε ο μηχανισμός να γνωρίζει τη περίπτωση που το bandwidth κορεστεί. Ορίζουμε ως κορεσμό στο bandwidth την κατάσταση κατά την οποία η τιμή του μετρούμενου συνολικού bandwidth ξεπεράσει το κατώφλι των 50 GB/sec.

Επιπλέον, ο μηχανισμός κρατάει και πληροφορίες για το bandwidth της HP εφαρμογής στην προσπάθειά του να "καταλάβει" τις αλλαγές φάσης της. Δεν μας ενδιαφέρουν οι αλλαγές υπολογιστικής φάσης, δηλαδή οι αλλαγές που οφείλονται αποκλειστικά στο είδος των υπολογισμών/εντολών της εφαρμογής και δεν απαιτούν περισσότερη cache, αλλά οι αλλαγές φάσης που υποδεικνύ-

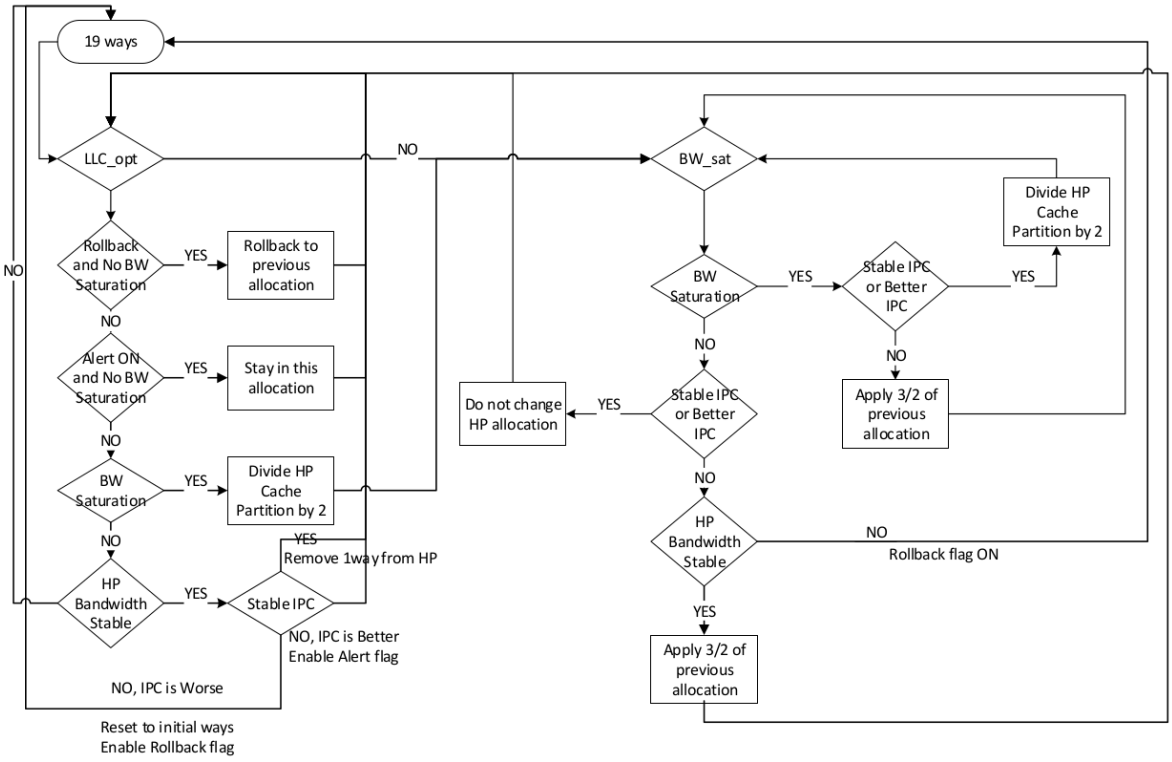
ουν ανάγκη για μεγαλύτερο τμήμα στην LLC. Οι αλλαγές φάσης της HP εφαρμογής που απαιτούν περισσότερο χώρο στην cache χαρακτηρίζονται από μεγαλύτερο συνολικό αριθμό misses και συνεπώς από αύξηση του bandwidth προς την κύρια μνήμη της εφαρμογής. Συγκεκριμένα, ορίζουμε στο μηχανισμό ως αλλαγή φάσης την κατάσταση όπου το bandwidth της υψηλής προτεραιότητας εφαρμογής ξεπεράσει το 130% του γεωμετρικού μέσου των τριών προηγούμενων μετρήσεων. Ειδικά για το bandwidth, χρησιμοποιούμε το ιστορικό μετρήσεων για να έχουμε καλύτερη εικόνα για τη πραγματική τιμή του bandwidth της εφαρμογής σε αυτή τη φάση της. Συνολικά, ο μηχανισμός λαμβάνει μετρήσεις ανά $\Delta t = 1$ sec και στην συνέχεια επιβάλει την απόφασή του για το τμήμα της LLC που θα δεσμεύσει η HP εφαρμογή.

Ο αλγόριθμος που υλοποιούμε (Σχήμα 4.1) διακρίνει δύο καταστάσεις, την κατάσταση LLC_opt και την κατάσταση BW_sat. Αρχικά, η εφαρμογή με υψηλή προτεραιότητα βρίσκεται στην κατάσταση LLC_opt και μεταβαίνει στην κατάσταση BW_sat αν και μόνο αν συμβεί κορεσμός στο bandwidth. Μάλιστα, θα παραμείνει στην BW_sat κατάσταση μέχρις ότου εξαλειφθεί αυτός ο κορεσμός, όπου και στη συνέχεια θα επιστρέψει στην κατάσταση LLC_opt. Καθεμία από τις καταστάσεις LLC_opt και BW_sat συνοδεύεται από συγκεκριμένες ενέργειες με στόχο σε κάθε μέτρηση ο μηχανισμός να μπορεί να ανιχνεύσει τις ανάγκες για LLC της υψηλής προτεραιότητας εφαρμογής και να πάρει την κατάλληλη απόφαση για το τμήμα που θα της αποδώσει στην LLC.

Αρχικά, η εφαρμογή με την υψηλή προτεραιότητα δεσμεύει τα N-1 ways της LLC, δηλαδή στην περίπτωσή μας, 19 ways και στις υπόλοιπες εννιά που συνεκτελούνται αποδίδεται το 1 way. Αν κορεστεί το bandwidth, τότε ο μηχανισμός μεταβαίνει στην κατάσταση BW_sat και αποδεσμεύει το μισό τμήμα της LLC της υψηλής προτεραιότητας εφαρμογής, ώστε να παραχωρήσει χώρο στις υπόλοιπες εφαρμογές και να μειώσει το συνολικό bandwidth. Στη συνέχεια, αν και εφόσον ο κορεσμός έχει αντιμετωπιστεί ο μηχανισμός μεταβαίνει στην κατάσταση LLC_opt. Σε αυτό το σημείο, ελέγχει αν η απότομη μείωση της LLC της HP εφαρμογής χειροτέρευσε την επίδοση της. Αν όχι, τότε μεταβαίνει κανονικά στην κατάσταση LLC_opt χωρίς να αλλάξει προς το παρόν το τμήμα που δεσμεύει στην LLC. Αν, ωστόσο, η επίδοση της υψηλής προτεραιότητας εφαρμογής μειώθηκε, τότε, αν διαπιστώσει αλλαγή φάσης ο αλγόριθμος θα παραχωρήσει 19 ways στην HP εφαρμογή. Διαφορετικά, θα αυξήσει λίγο παραπάνω το τμήμα που δεσμεύει στην LLC παραχωρώντας τμήμα ίσο $\frac{3}{2}$ του τμήματος που είχε στην αμέσως προηγούμενη απόφαση.

Στην κατάσταση LLC_opt υποθέτουμε ότι το bandwidth δεν έχει κορεστεί. Όταν ανιχνευτεί αλλαγή φάσης ο μηχανισμός επιστρέφει στην αρχική κατάσταση (επαναφορά, reset), δηλαδή αποδίδει στην HP εφαρμογή τα 19 ways. Ωστόσο, σε κάθε επαναφορά, είτε ο μηχανισμός βρίσκεται στην κατάσταση LLC_opt είτε στην κατάσταση BW_sat, ενεργοποιείται η επιστροφή (rollback). Έτσι, στην επόμενη μέτρηση θα ελεγχεί αν αυτή η επαναφορά στο αρχικό σημείο βελτίωσε την επίδοση της εφαρμογής. Αν κάτι τέτοιο δεν συνέβη, π.χ. τα 19 ways δεν προσέφεραν καλύτερη επίδοση ή την έβλαψαν, τότε ο μηχανισμός επιστρέφει στην προηγούμενη κατάσταση (επιστροφή). Όσο δεν αλλάζει η φάση και το IPC παραμένει σταθερό, ο μηχανισμός αφαιρεί σε κάθε μέτρηση 1 way από την εφαρμογή με υψηλή προτεραιότητα. Όταν το IPC βελτιωθεί, τότε, ο μηχανισμός ενεργοποιεί την παύση (alert mode) και περιμένει σε αυτό το σχήμα καταμερισμού LLC για 2 μετρήσεις, δηλαδή έως ότου πρακτικά το IPC σταθεροποιηθεί και πάλι.

Ενδιαφέρον παρουσιάζει η περίπτωση όπου δεν παρατηρείται αλλαγή φάσης αλλά το IPC είναι μειωμένο σε σχέση με την προηγούμενη μέτρηση. Στην συγκεκριμένη κατάσταση ο μηχανισμός δεν είναι σε θέση να μπορεί να γνωρίζει αν η επιδείνωση του IPC οφείλεται σε αλλαγή υπολογιστικής φάσης που δεν συνδέεται με απαίτηση για LLC (για αυτό και δεν είδαμε αύξηση στο bandwidth) ή αν η συνεχής αφαίρεση ways ξαφνικά δημιούργησε πρόβλημα. Και σε αυτή την περίπτωση θα κάνουμε επαναφορά στα 19 ways ενεργοποιώντας την επιστροφή. Αν η μείωση στο IPC οφείλεται σε έλλειψη χώρου στην cache, τότε η επαναφορά αποδεικνύεται βελτιωτική και άρα ο αλγόριθμος ξαναξεκινάει από την αρχή. Αν η μείωση στο IPC οφείλεται σε νέα υπολογιστή φάση της εφαρμογής, τότε κατά πάσα πιθανότητα η επαναφορά δεν ωφελεί. Σε κάθε περίπτωση, ο αλγόριθμος θα επιλέξει το σχήμα καταμερισμού cache που προσφέρει το καλύτερο IPC. Παραθέτουμε το συνολικό διάγραμμα ροής της υλοποίησής μας.



Σχήμα 4.1: Διάγραμμα Ροής (FSM) του Μηχανισμού BW-QoS

Ακολουθεί ο Πίνακας Αληθείας 4.1 που διέπει τις αποφάσεις και ενέργειες του αλγορίθμου της υλοποίησής μας. Η δεύτερη στήλη αφορά τη σχέση μεταξύ του IPC_t της συγκεκριμένης χρονικής στιγμής που γίνεται η μέτρηση και του IPC_{t-1} της προηγούμενης μέτρησης. Η τρίτη στήλη αφορά την ύπαρξη ή όχι κορεσμού στο bandwidth και η τέταρτη στήλη αφορά την αλλαγή ή όχι στη φάση της εφαρμογής με υψηλή προτεραιότητα. Στην πέμπτη στήλη αναφέρουμε την απόφαση του μηχανισμού ενώ οι εκτη και έβδομη στήλες δείχνουν την παρούσα (S_t) και την επόμενη κατάσταση (S_{t+1}) στην οποία θα μεταβεί ο μηχανισμός, αντίστοιχα. Το σύμβολο "X" σημαίνει ότι στην συγκεκριμένη συνθήκη το παρόν μέγεθος είναι αδιάφορο και δεν λαμβάνεται υπόψιν. Για παράδειγμα, όταν διαπιστωθεί αλλαγή φάσης ή κορεσμός στο bandwidth η σχέση μεταξύ των IPC της τωρινής και της προηγούμενης μέτρησης μας είναι αδιάφορη, αφού στη μεν αλλαγή φάσης θα κάνουμε επαναφορά, στο δε κορεσμό θα χρειαστεί να μειώσουμε στο μισό το τμήμα της LLC της εφαρμογής με υψηλή προτεραιότητα.

	IPC_t (?) IPC_{t-1}	BW Κορ.	Αλ. Φάσης	Απόφαση	S_t	S_{t+1}
0	=	0	0	-1 way	LLC_opt	LLC_opt
1	>	0	0	όχι αλλαγή, παύση	LLC_opt	LLC_opt
2	<	0	0	επαν., επιστ.	LLC_opt	LLC_opt
3	X	0	1	επαν., επιστ.	LLC_opt	LLC_opt
4	X	1	X	alloc/2	LLC_opt	BW_sat
5	$\geq$	1	X	alloc/2	BW_sat	BW_sat
6	<	1	X	3(allocn)/2	BW_sat	BW_sat
7	$\geq$	0	X	όχι αλλαγή	BW_sat	LLC_opt
8	<	0	1	επαν., επιστ.	BW_sat	LLC_opt
9	<	0	0	3(alloc)/2	BW_sat	LLC_opt

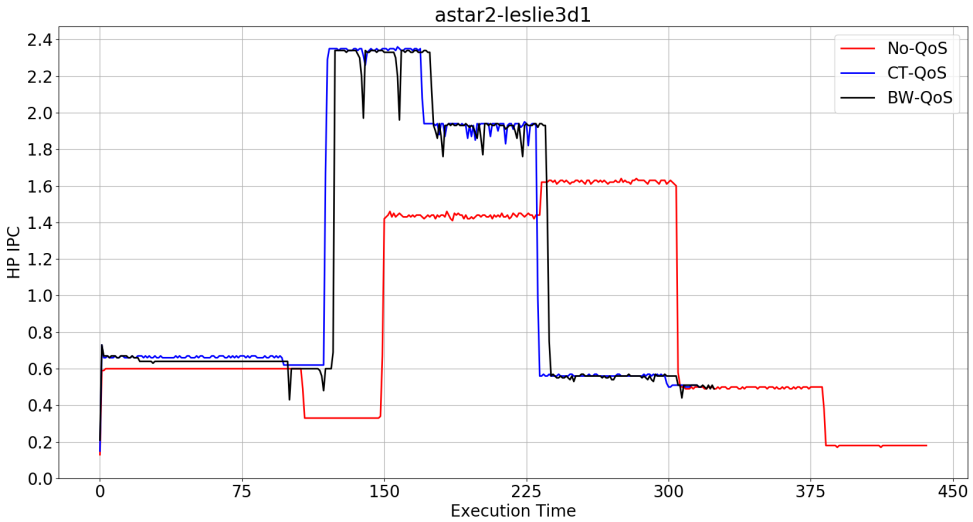
Πίνακας 4.1: Πίνακας Αληθείας Μεταβάσεων και Ενεργειών του Μηχανισμού BW-QoS

4.2 Παραδείγματα Λειτουργίας του Μηχανισμού BW-QoS

Σε αυτήν την ενότητα παραθέτουμε κάποια παραδείγματα που εξηγούν και αναδεικνύουν τη λειτουργία του μηχανισμού BW-QoS που υλοποιούμε.

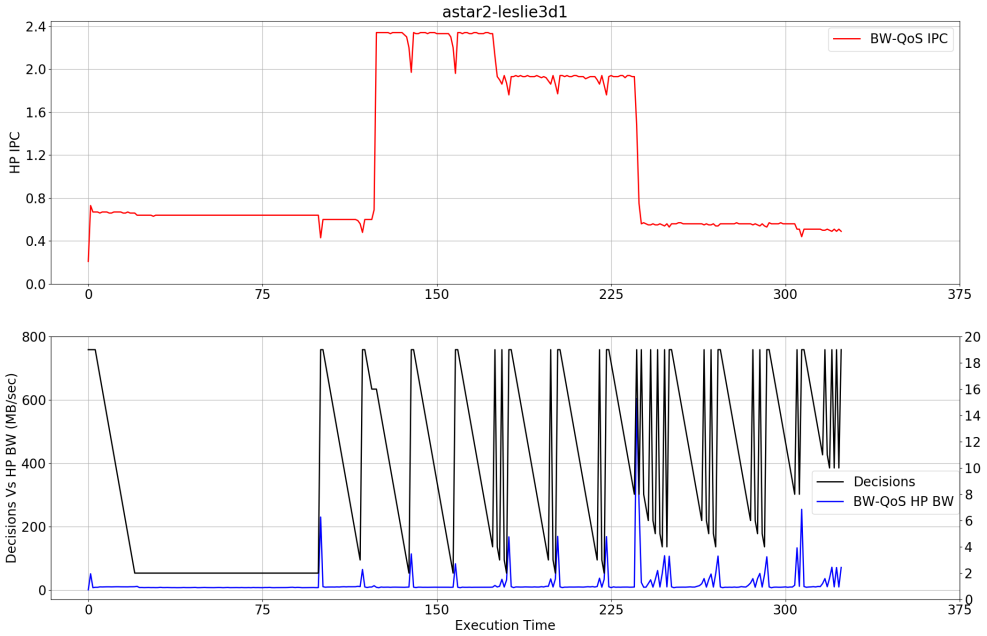
4.2.1 Κατηγορία Α

Παραθέτουμε τη γραφική παράσταση του IPC του astar2 ως HP κατά τη συνεκτέλεσή του με leslie3d1 ως LP για τις 3 πολιτικές συναρτήσει του χρόνου εκτέλεσης. Είναι εμφανής η χρονική καθυστέρηση του astar2 στην No-QoS πολιτική. Την χρονική αυτή καθυστέρηση αντιμετωπίζει ο μηχανισμός BW-QoS.



Σχήμα 4.2: Γραφική Παράσταση IPC του astar2 κατά τη συνεκτέλεσή του ως HP με leslie3d1 ως LP

Στο Σχήμα 4.3 σχεδιάζουμε τα τμήματα LLC που δυναμικά επιβάλει ο μηχανισμός στο astar2 σε συνδυασμό με το IPC και το bandwidth του astar2. Όπως βλέπουμε, ο μηχανισμός ξεκινάει αποδίδοντας 19 ways στο astar2 και όσο παρατηρείται σταθερό IPC και BW, δηλαδή από τη χρονική στιγμή 0 έως τη χρονική στιγμή 100 αφαιρεί σε κάθε απόφαση 1 way.



Σχήμα 4.3: Γραφική Παράσταση IPC και των allocations που επιβάλει ο μηχανισμός στο astar2 κατά τη συνεκτέλεσή του με LP leslie3d1

Το astar2 διατηρεί σταθερή επίδοση ακόμα και με 2 ways στην πρώτη φάση του. Στην συνέχεια, περίπου τη χρονική στιγμή 100, παρατηρούμε αλλαγή φάσης, όπου το bandwidth αυξάνεται στιγμιαία. Ο μηχανισμός πραγματοποιεί επαναφορά και παραχωρεί στο astar2 19 ways. Επειδή τα 19 ways λειτούργησαν βελτιωτικά για την επίδοσή του, ο μηχανισμός δεν γυρίζει στην προηγούμενη κατάσταση των 2 ways, αλλά ξεκινάει ξανά από την αρχή. Παρατηρούμε ότι προς το τέλος της εκτέλεσης το astar2 έχει ασταθές bandwidth. Αυτό εξαναγκάζει το μηχανισμό να επιχειρεί συνεχώς επαναφορές στα 19 ways. Κάποια από αυτές τις μεταβάσεις φαίνεται να είναι αχρείαστες όπως για παράδειγμα τις χρονικές στιγμές 200 και 250, όπου ο μηχανισμός ξαναεπιστρέφει στην προηγούμενη κατάσταση.

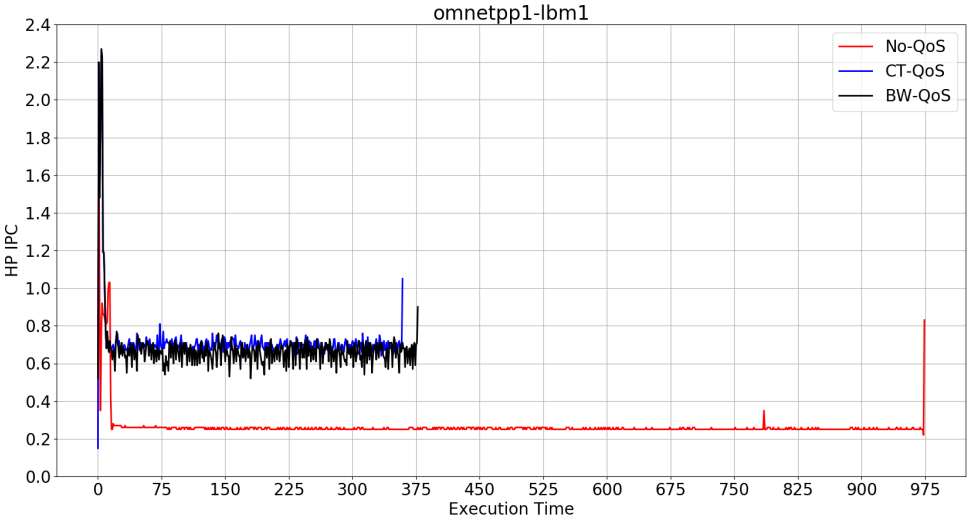
Στον παρακάτω πίνακα σημειώνουμε την επίδοση του astar2 αλλά και των αντιγράφων του leslie3d1 όπως αυτή μετρήθηκε κατά τη διάρκεια εκτέλεσης.

	High Priority (astar2)		Low Priority (9 instances of leslie3d1)
	IPC	LLC (KB)	IPC
No-Qos	0.83	631	1.41
CT-QoS	1.14	19780	0.56
BW-Qos	1.13	13568	1.18

Πίνακας 4.2: Επίδοση των LP leslie3d1 ανά πολιτική εκτέλεσης του συνδυασμού astar2-leslie3d1

Παρατηρούμε ότι το BW-QoS όχι μόνο βελτίωσε την εκτέλεση του astar2 και στην ουσία εξασφάλισε την βέλτιστη εκτέλεσή με 47% λιγότερο χώρο στην LLC σε σχέση με τη CT-QoS πολιτική, αλλά παράλληλα βελτίωσε σημαντικά, σχεδόν κατά το διπλάσιο, την επίδοση των LP εφαρμογών.

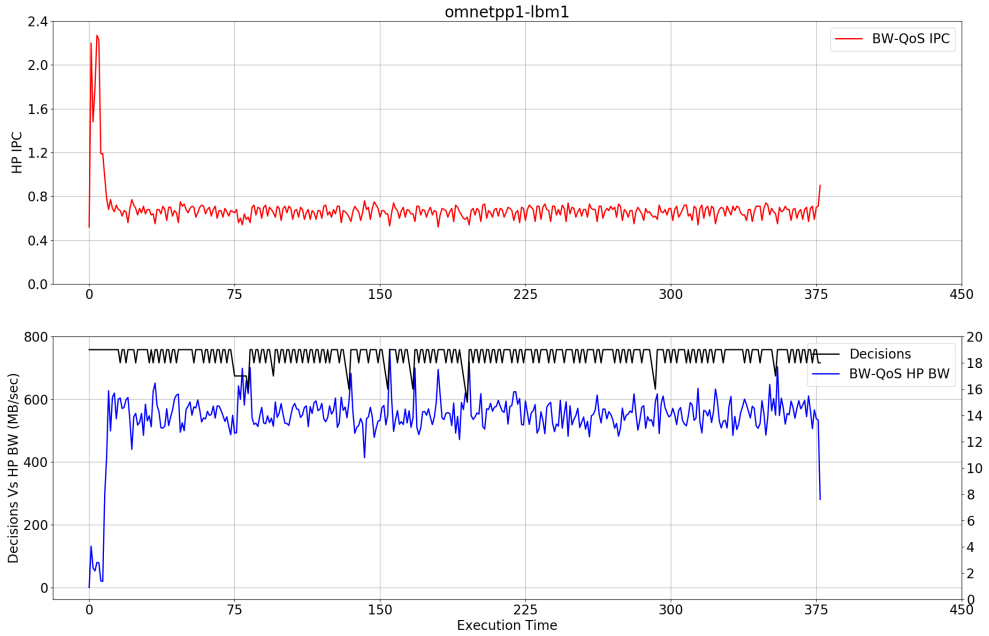
Στο παρακάτω παράδειγμα, ο μηχανισμός αντιλαμβάνεται την ανάγκη του omnetpp1 για μεγάλη cache. Όσες απόπειρες δοκιμάστηκαν για να αφαιρεθούν ways από το omnetpp1 κατά την συνεκτέλεσή του με τα εννέα αντίγραφα lbm1, ”απέτυχαν”. Από το Σχήμα 4.5 βλέπουμε ότι το bandwidth του omnetpp1 παρουσιάζει μεγάλη μεταβλητότητα. Αυτό το χαρακτηριστικό ”δυσκολεύει” το μηχανισμό να καταλάβει τυχόν αλλαγές φάσης. Όσο το bandwidth μεταβάλλεται έντονα συμβαίνουν συχνά επαναφορές, όπως βλέπουμε και στα σχήματα. Τέλος, ο μηχανισμός πετυχαίνει ίδια εκτέλεση με τη CT-QoS πολιτική παραχωρώντας στο omnetpp1 το 93% της cache. Σχετικά με την επίδοση των αντιγράφων (Πίνακας 4.3) lbm με χαμηλή προτεραιότητα δεν παρατηρείται ουσιαστική βελτίωση στην επίδοσή τους.



Σχήμα 4.4: Γραφική Παράσταση IPC του omnetpp1 κατά τη συνεκτέλεσή του ως HP με lbm1 ως LP

	High Priority (omnetpp1)		Low Priority (9 instances of lbm1)
	IPC	LLC	IPC
No-Qos	0.26	998	0.74
CT-QoS	0.71	24137	0.33
BW-Qos	0.68	23707	0.39

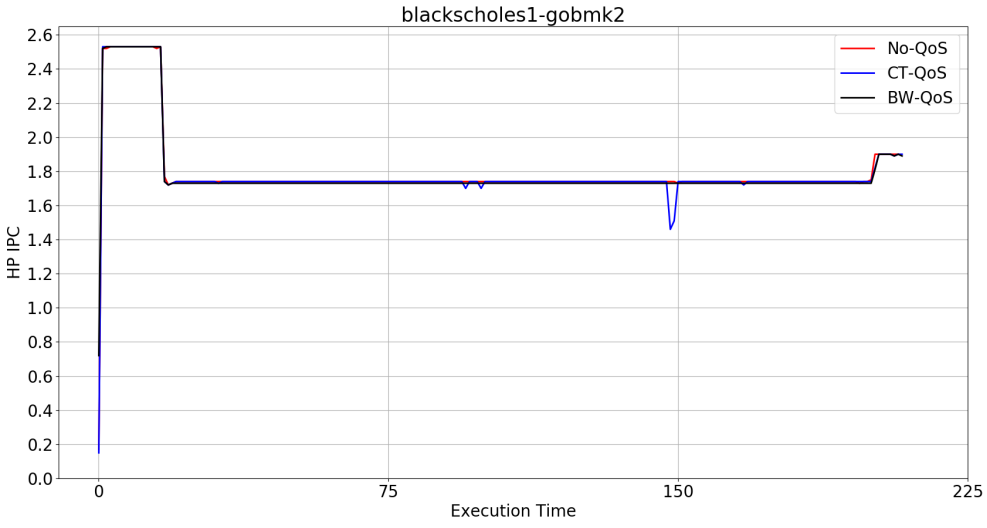
Πίνακας 4.3: Επίδοση των LP lbm1 ανά πολιτική εκτέλεσης του συνδυασμού omnetpp1-lbm1



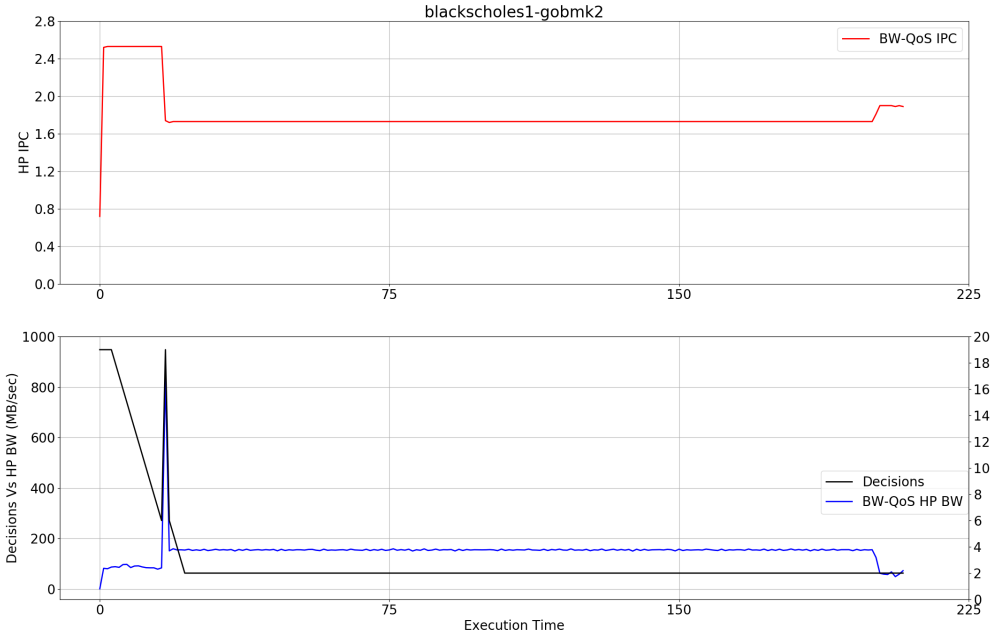
Σχήμα 4.5: Γραφική Παράσταση IPC και των allocations που επιβάλει ο μηχανισμός στο omnetpp1 κατά τη συνεκτέλεσή του με LP lbm1

4.2.2 Κατηγορία B

Το blackscholes1 αποτελείται από 2 φάσεις (Σχήμα 4.6). Ο μηχανισμός κατά την αλλαγή φάσης επιβάλει την αρχική κατάσταση, δηλαδή 19 ways στο blackscholes1 και 1 way στα gobmk2. Ωστόσο, αυτή η ενέργεια δεν αλλάζει την εκτέλεση της HP εφαρμογής και για αυτό επιστρέφει στην προηγούμενη κατάσταση, δηλαδή παραχωρεί 7 ways στο blackscholes1 και 13 στα LP lbm1.



Σχήμα 4.6: Γραφική Παράσταση IPC του blackscholes1 κατά τη συνεκτέλεσή του ως HP με gobmk2 ως LP



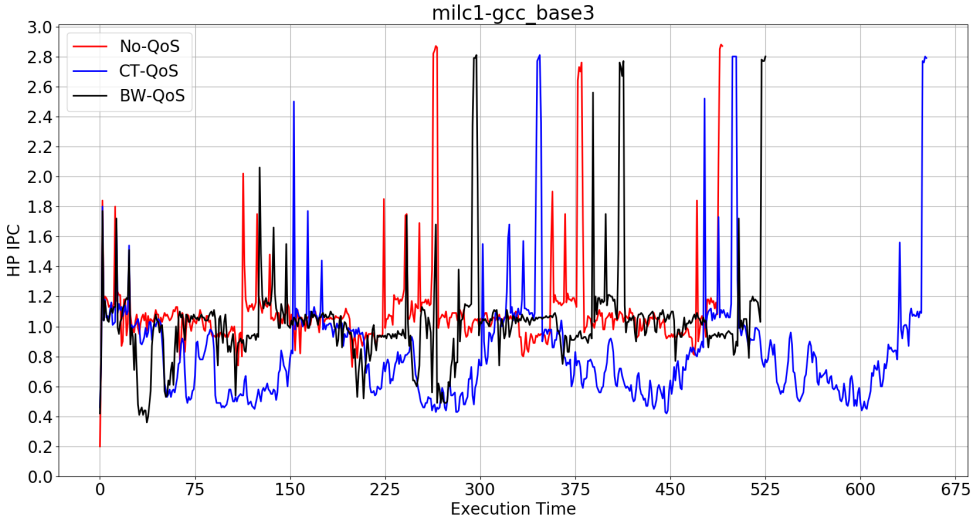
Σχήμα 4.7: Γραφική Παράσταση IPC και των allocations που επιβάλει ο μηχανισμός στο blackscholes1 κατά τη συνεκτέλεσή του με LP gobmk2

Το milc1 στην BW-QoS πολιτική εκτελείται σε χρόνο περίπου ίσο με το 106% του χρόνου εκτέλεσης με τη No-QoS πολιτική και στο 85% του χρόνου εκτέλεσης με τη CT-QoS πολιτική. Στο γράφημα του Σχήματος 4.9 παρατηρούμε ότι τις χρονικές στιγμής 100, 350 και 500 συμβαίνει κορεσμός στο bandwidth και ο μηχανισμός μειώνει το τμήμα της LLC του milc1 στο μισό. Η χρονική απόκλιση του μηχανισμού BW-QoS από τον βέλτιστο χρόνο εκτέλεσης No-QoS οφείλεται στην επιλογή των 19 ways ως σημείο επαναφοράς κατά το reset. Για παράδειγμα στο χρονικό διάστημα 0-100 στο σχήμα βλέπουμε μία μεγάλη πτώση στο IPC του milc1 (περίπου 0.4) την ίδια στιγμή που ο μηχανισμός του αποδίδει 19 ways λόγω κάποιας επαναφοράς. Μάλιστα παρατηρούμε ότι σε όλο εκείνο το διάστημα που το IPC είναι χαμηλότερο του IPC κατά τη No-QoS, δεσμεύει τα 19 ways της cache. Αντίστοιχη συμπεριφορά παρατηρούμε και τη χρονική στιγμή 260-300. Επομένως καταλήγουμε ότι οι επαναφορές στην αρχική κατάσταση των 19 ways αποδεικνύονται μακροπρόθεσμα επιβλαβή για την επίδοση.

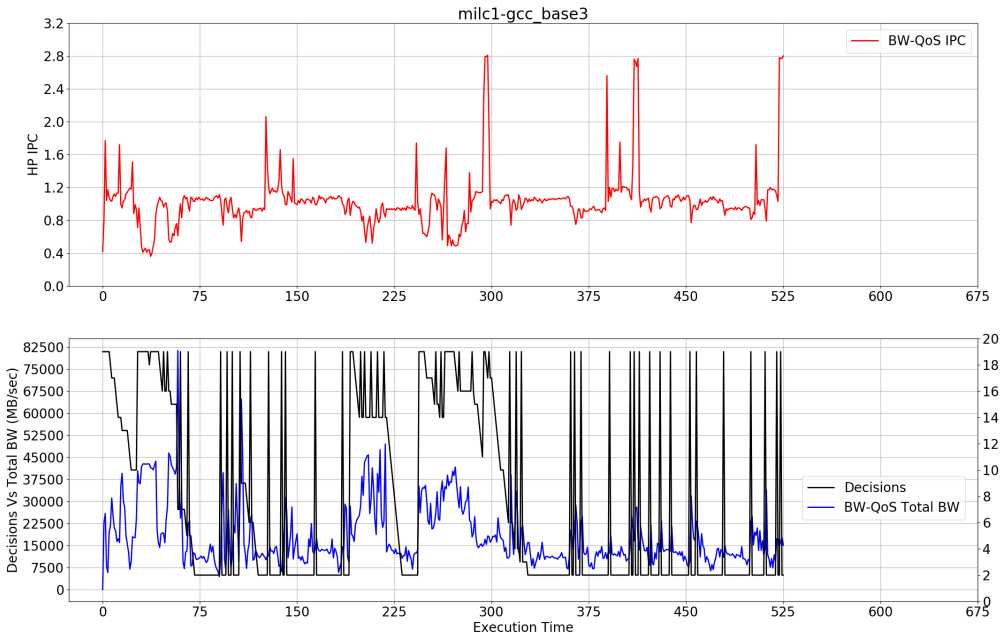
Αναφορικά με τις LP εφαρμογές ο μηχανισμός BW-QoS τετραπλασιάζει την εκτέλεσή τους σε σχέση με την CT-QoS εκτέλεσή.

	High Priority (milc1)		Low Priority (9 instances of gcc_base3)
	IPC	LLC	IPC
No-Qos	1.10	6232	1.34
CT-QoS	0.83	24226	0.33
BW-Qos	1.03	13142	1.25

Πίνακας 4.4: Επίδοση των LP gcc_base3 ανά πολιτική εκτέλεσης του συνδυασμού milc1-gcc_base3



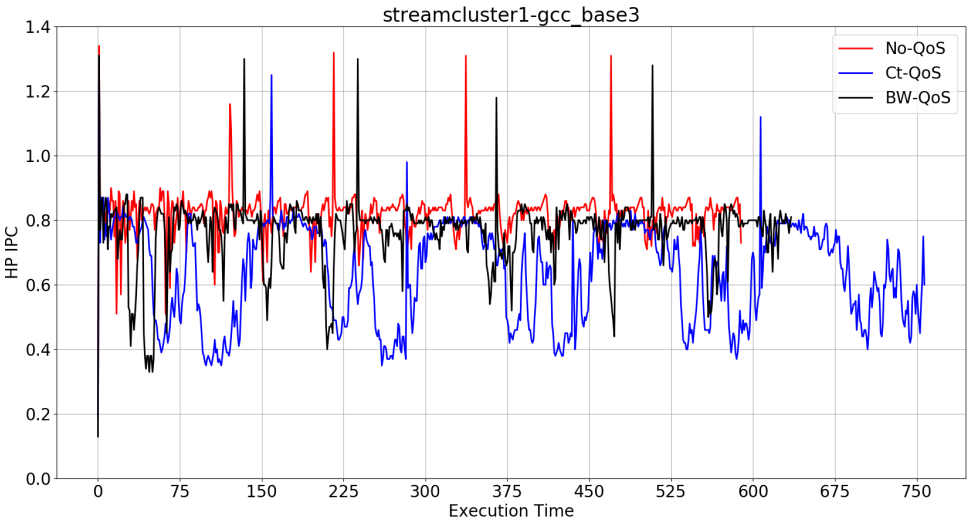
Σχήμα 4.8: Γραφική Παράσταση IPC του milc1 κατά τη συνεκτέλεσή του ως HP με gcc_base3 ως LP



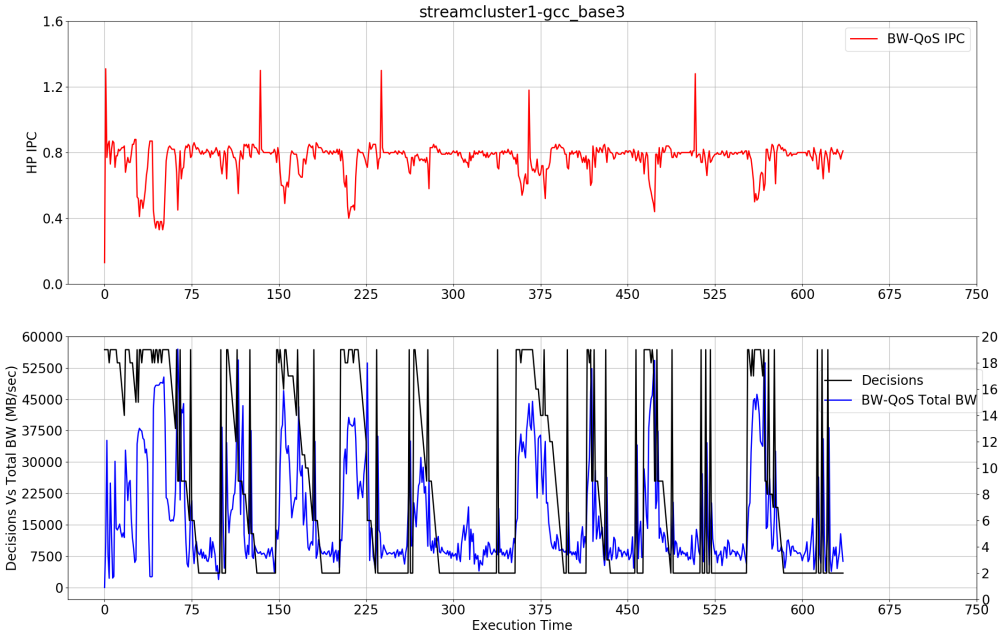
Σχήμα 4.9: Γραφική Παράσταση συνολικού bandwidth και των allocations που επιβάλει ο μηχανισμός στο milc1 κατά τη συνεκτέλεσή του με LP gcc_base3

Όμοια συμπεράσματα παρατηρούμε και στις περιπτώσεις των συνεκτελέσεων streamcluster1-gcc_base3 και milc1-gcc_base8. Για παράδειγμα, στη συνεκτέλεση του streamcluster1 με εννέα αντίγραφα gcc_base3 παρατηρούμε ότι τα διαστήματα κοντά στις χρονικές στιγμές 50 sec, 150 sec, 220 sec, 475 sec, κτλ. υπάρχει αισθητή μείωση του IPC στην υλοποίησή μας BW-QoS σε σχέση με τη No-QoS. Αντίστοιχα με πριν διαπιστώνουμε στα διαγράμματα αποφάσεων ότι τις εκάστοτε χρονικές στιγμές η απόφαση του μηχανισμού είναι να παραχωρήσει στο streamcluster 19 ways της

LLC.Τέλος, στο Σχήμα 4.11 συμπεραίνουμε ότι στις περιόδους που το streamcluster δεσμεύει σχεδόν ολόκληρη την cache το συνολικό bandwidth εξαπλασιάζεται.



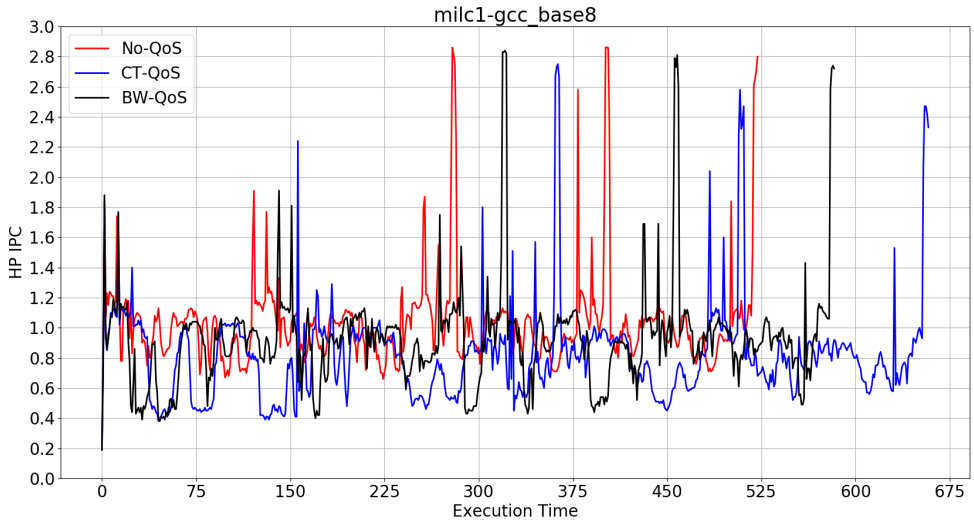
Σχήμα 4.10: Γραφική Παράσταση IPC του streamcluster1 κατά τη συνεκτέλεσή του ως HP με gcc_base3 ως LP



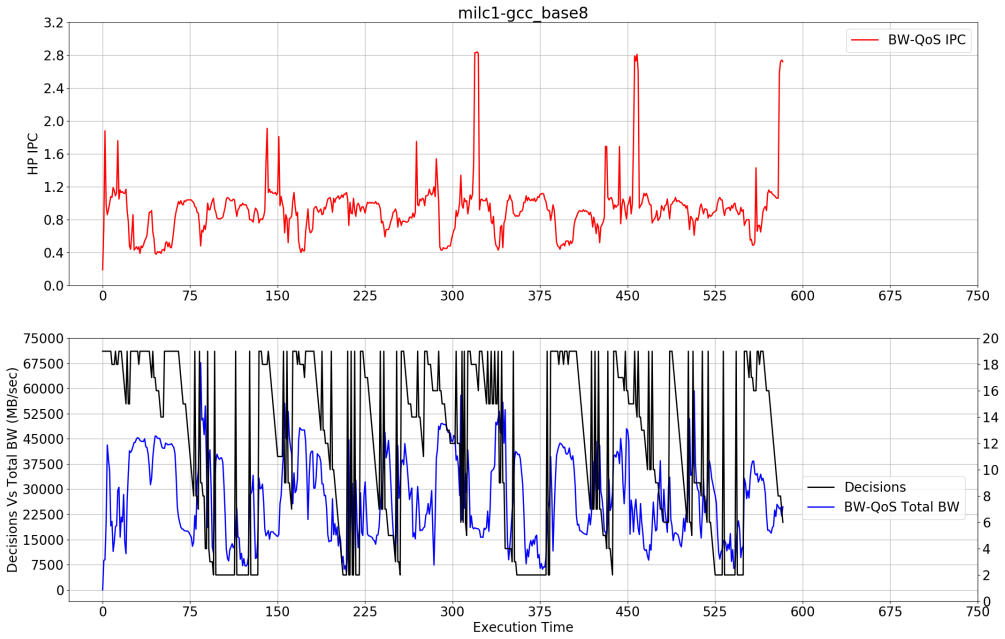
Σχήμα 4.11: Γραφική Παράσταση συνολικού bandwidth και των allocations που επιβάλει ο μηχανισμός στο streamcluster1 κατά τη συνεκτέλεσή του με LP gcc_base3

Αντίστοιχα και στο παράδειγμα της συνεκτέλεσης milc1-gcc_base8 επιβεβαιώνουμε την ακαταλληλότητα του τμήματος των 19 ways της cache ώς σημείο επαναφοράς του μηχανισμού σε αυτές τις περιπτώσεις. Μπορούμε αντίστοιχα να παρατηρήσουμε τις χρονικές περιόδους 0-80 sec και 390-410 sec της εκτέλεσης στις οποίες το IPC του milc1 επιδέχεται μείωση 50% σε σχέση με

το IPC της ίδιας εφαρμογής στη No-QoS εκτέλεση. Παράλληλα μπορεί κάποιος να παρατηρήσει την πολλαπλάσια αύξηση του συνολικού bandwidth των συνεκτελούμενων εφαρμογών αυτές τις χρονικές περιόδους.



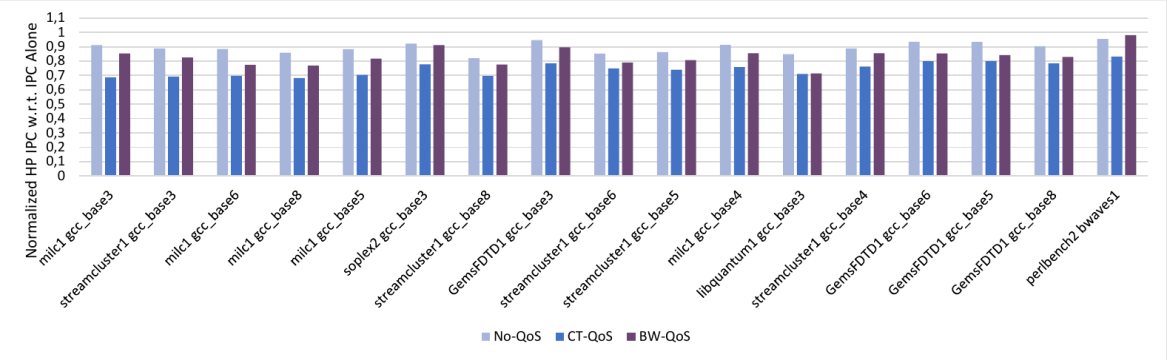
Σχήμα 4.12: Γραφική Παράσταση IPC του milc1 κατά τη συνεκτέλεσή του ως HP με gcc_base8 ως LP



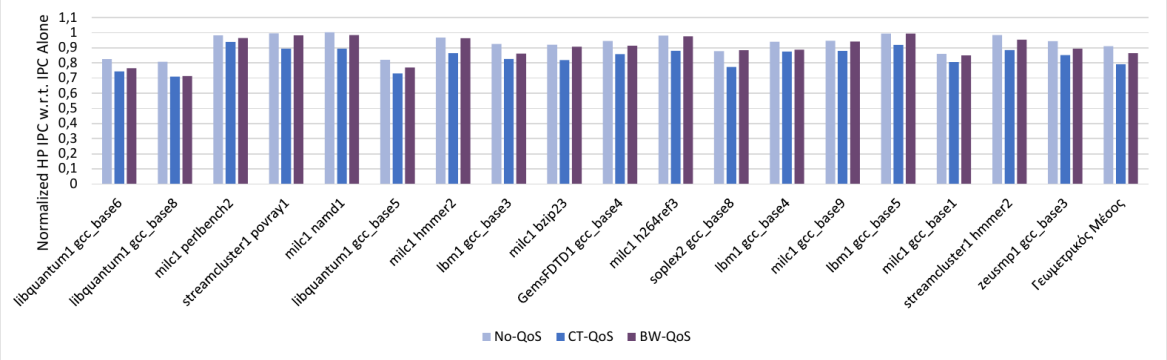
Σχήμα 4.13: Γραφική Παράσταση συνολικού bandwidth και των allocations που επιβάλει ο μηχανισμός στο milc1 κατά τη συνεκτέλεσή του με LP gcc_base8

4.3 Αξιολόγηση του Μηχανισμού BW-QoS σε συνεκτελέσεις Κατηγορίας Β

Τα παρακάτω σχήματα αναπαριστούν το μέσο IPC της HP εφαρμογής στις εκτελέσεις No-QoS, CT-QoS και BW-QoS, κανονικοποιημένο ως προς το μέσο IPC της συγκεκριμένης εφαρμογής όταν αυτή εκτελείται απομονωμένα στο σύστημα. Αναφέρουμε τις 35 από τις 70 περιπτώσεις την Κατηγορίας Β οι οποίες εμφανίζουν κορεσμό στο bandwidth.



Σχήμα 4.14: Αξιολόγηση BW-QoS στα workloads που προκαλούν κορεσμό στο bandwidth (I)



Σχήμα 4.15: Αξιολόγηση BW-QoS στα workloads που προκαλούν κορεσμό στο bandwidth (II)

Παρατηρούμε ότι ο μηχανισμός BW-QoS που υλοποιούμε βελτιώνει (8.8% σε σχέση με CT-QoS) την εκτέλεση των εφαρμογών υψηλής προτεραιότητας στις περιπτώσεις που οι συνεκτελούμενες εφαρμογές προκαλούν κορεσμό στο bandwidth. Ωστόσο, δεν καταφέρνει να επιτύχει την καλύτερη εκτέλεση που προσφέρει η No-QoS πολιτική (απόκλιση περίπου 5.5%). Τέτοιες περιπτώσεις είναι για παράδειγμα οι συνεκτελέσεις των milc1-gcc_base6, GemsFDTD1-gcc_base5, zeusmp1-gcc_base3, libquantum1-gcc_base8, κτλ. Από την ανάλυση που κάναμε προηγουμένως, καταλήγουμε ότι το βασικό μειονέκτημα του μηχανισμού BW-QoS για αυτούς τους συνδυασμούς είναι η επιλογή των 19 ways ως αρχικό σημείο επαναφοράς.

Κεφάλαιο 5

Δειγματοληψία Κατάλληλου Σημείου Επαναφοράς

5.1 Δομή και Λειτουργία του Μηχανισμού BW-QoS-Reset-Sampling

Στον νέο μηχανισμό που προτείνουμε εισάγουμε την έννοια της δειγματοληψίας. Στόχος της είναι η εύρεση ενός καταλληλότερου σημείου επαναφοράς-εκκίνησης του μηχανισμού στις περιπτώσεις που η συνεκτέλεση εμφανίζει κορεσμό στο bandwidth. Δεδομένου ότι το σταθερό σημείο επαναφοράς των 19 ways ενδεικνύεται στις συνεκτελέσεις της Κατηγορίας Α, η δειγματοληψία ενεργοποιείται μόνο όταν ο μηχανισμός ανιχνεύσει κορεσμό στο bandwidth.

Η λειτουργία του αλγορίθμου δεν αλλάζει στις συνεκτελέσεις όπου δεν εμφανιστεί κορεσμός. Στις υπόλοιπες περιπτώσεις, ωστόσο, που αντιλαμβανόμαστε κορεσμό ο μηχανισμός αντί να μειώσει το μισό το τμήμα της cache της εφαρμογής με υψηλή προτεραιότητα, κάνει δειγματοληψία σημείων, δηλαδή αποδίδει διαδοχικά στην HP εφαρμογή 19, 18, 16, 14, 12, 10, 8, 6, 4 και 2 ways. Κάθε ένα από αυτά τα σημεία εφαρμόζεται για ένα καθορισμένο χρονικό διάστημα. Επιπλέον, για κάθε δείγμα μετράει το IPC και χαρακτηρίζει ως σημείο επαναφοράς το δείγμα με το μεγαλύτερο IPC (IPC_phase). Σε μία συνεκτέλεση που έχει εμφανίσει κάποια στιγμή κορεσμό στο bandwidth, στην περίπτωση που δεν έχουμε αλλαγή φάσης, αλλά, το IPC μειωθεί, ο μηχανισμός επιβάλει το αρχικό σημείο που θεωρήθηκε κατάλληλο κατά τη δειγματοληψία. Στην επόμενη μέτρηση συγκρίνει το νέο IPC με την τιμή του IPC_phase και αν είναι χειρότερο ο μηχανισμός πραγματοποιεί νέα δειγματοληψία, διαφορετικά συνεχίζει κανονικά.

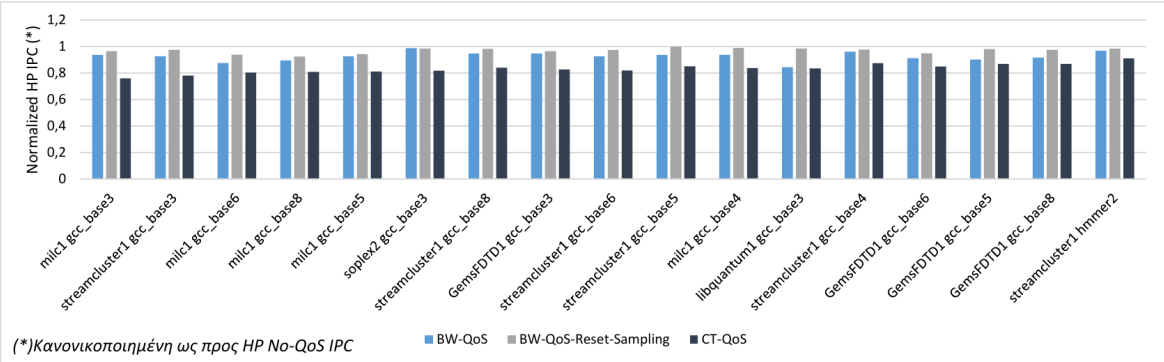
	IPC_t (?) IPC_{t-1}	BW Κορ.	Αλ. Φάσης	Απόφαση	S_t	S_{t+1}
0	=	0	0	-1 way	LLC_opt	LLC_opt
1	>	0	0	όχι αλλαγή, παύση	LLC_opt	LLC_opt
2	<	0	0	επαν.	LLC_opt	LLC_opt
3	X	0	1	επαν. ή δειγμ.	LLC_opt	LLC_opt
4	X	1	X	δειγμ.	LLC_opt	BW_sat
5	$\geq$	1	X	δειγμ.	BW_sat	BW_sat
6	<	1	X	δειγμ.	BW_sat	BW_sat
7	$\geq$	0	X	όχι αλλαγή	BW_sat	LLC_opt
8	<	0	1	δειγμ.	BW_sat	LLC_opt
9	<	0	0	3(alloc)/2	BW_sat	LLC_opt

Πίνακας 5.1: Πίνακας Αληθείας Μεταβάσεων και Ενεργειών του Μηχανισμού BW-QoS-Reset-Sampling

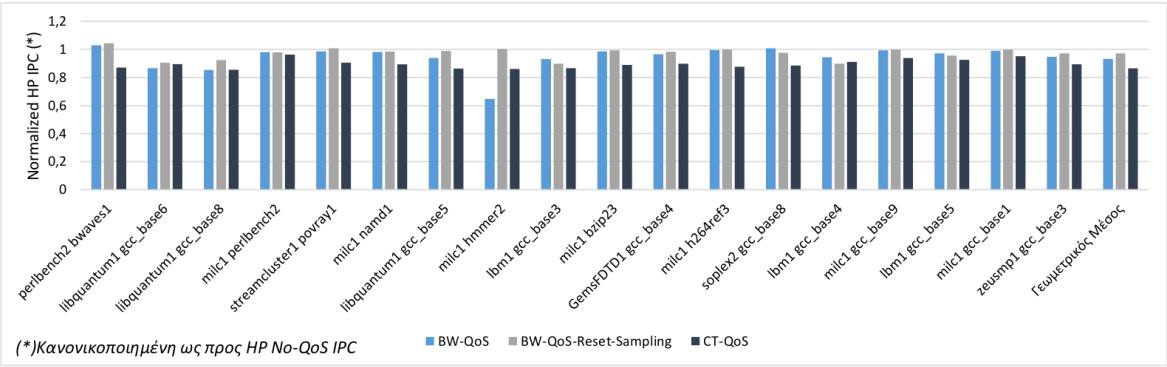
5.2 Σύγκριση Μηχανισμών BW-QoS και BW-QoS-Reset-Sampling

Για τον μηχανισμό BW-QoS-Reset-Sampling επιλέγουμε ως χρονική διάρκεια επιβολής κάθε δείγματος τα 10 msec, δηλαδή 22 Mcycles, και άρα η δειγματοληψία διαρκεί 100 msec. Μάλιστα, διαπιστώνουμε ότι αυτό το χρονικό διάστημα είναι αρκετό για να εκτελεστούν κάποια εκατομμύρια εντολές ούτως ώστε η μετρήσεις για το IPC σε κάθε δείγμα να έχουν νόημα.

Στα παρακάτω διαγράμματα γίνεται εμφανής η καταλληλότητα του μηχανισμού BW-QoS-Reset-Sampling έναντι του μηχανισμού BW-QoS. Οι τιμές του μέσου IPC είναι κανονικοποιημένες ως προς το μέσο IPC της πολιτικής No-QoS που στα συγκεκριμένα workloads είναι και η βέλτιστη. Ο γεωμετρικός μέσος των αποτελεσμάτων BW-QoS-Reset-Sampling (0.97) αποκλίνει κατά 3% από την βέλτιστη εκτέλεση και είναι 4% καλύτερος από το γεωμετρικό μέσο των αποτελεσμάτων BW-QoS (0.93). Ο μηχανισμός BW-QoS-Reset-Sampling βελτιώνει κατά περίπου 12.7% την εκτέλεση της εφαρμογής με υψηλή προτεραιότητα σε σχέση με την CT-QoS πολιτική (0.86).

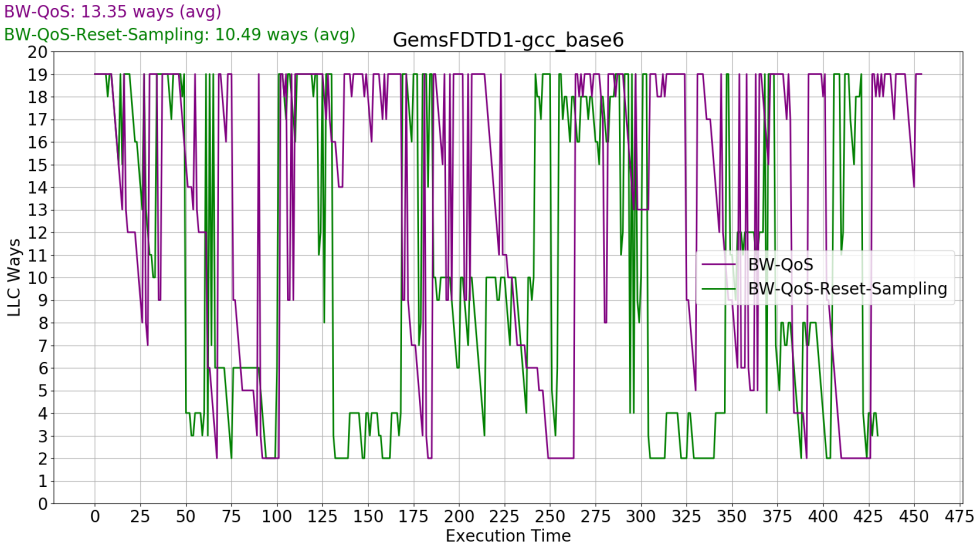


Σχήμα 5.1: BW-QoS και BW-QoS-Reset-Sampling στα workloads που προκαλούν κορεσμό στο bandwidth (I)

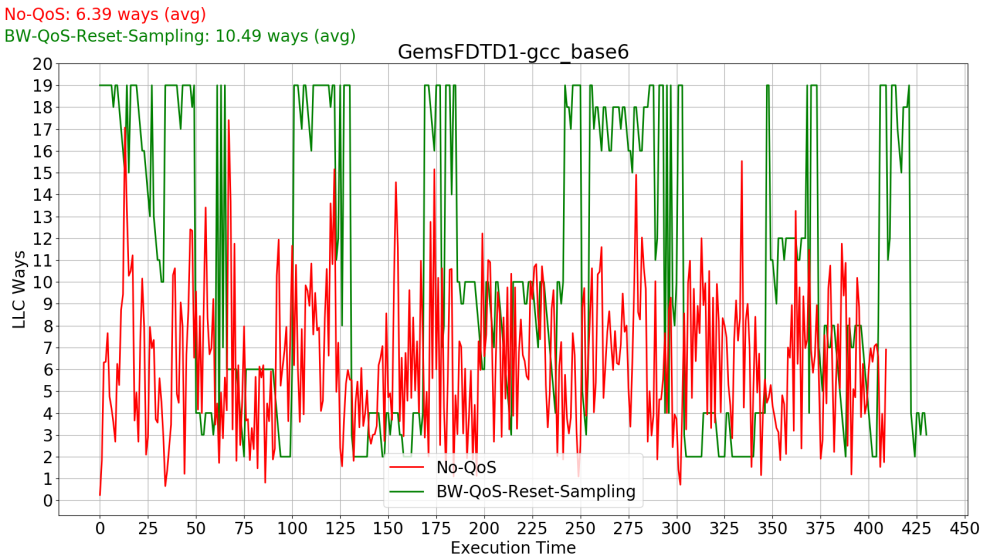


Σχήμα 5.2: BW-QoS και BW-QoS-Reset-Sampling στα workloads που προκαλούν κορεσμό στο bandwidth(II)

Παραθέτουμε ένα παράδειγμα εκτέλεσης του μηχανισμού BW-QoS-Reset-Sampling και συγκεκριμένα τις αποφάσεις που επιβάλει ο μηχανισμός στην HP εφαρμογή σε αντιπαραβολή με τις αποφάσεις του μηχανισμού BW-QoS. Παρατηρούμε ότι ο μέσος αριθμός ways που αποδίδει ο μηχανισμός με τη δειγματοληψία είναι περίπου 3 ways μικρότερος από τον μέσο αριθμό ways που αποδίδει στην εφαρμογή με υψηλή προτεραιότητα ο μηχανισμός BW-QoS που υλοποιήσαμε αρχικά. Μάλιστα, τις χρονικές στιγμές 150 και 230 sec βλέπουμε ότι ο μηχανισμός BW-QoS-Reset-Sampling πραγματοποιεί επαναφορές στο αρχικό σημείο, η εύρεση του οποίου οφείλεται στη δειγματοληψία. Σε αυτό το παράδειγμα ο μηχανισμός BW-QoS-Reset-Sampling βελτιώνει τον χρόνο εκτέλεσης κατά περίπου 5% συγκριτικά με τον μηχανισμό BW-QoS.



Σχήμα 5.3: Παράδειγμα Αποφάσεων BW-QoS και BW-QoS-Reset-Sampling



Σχήμα 5.4: Παράδειγμα Αποφάσεων BW-QoS-Reset-Sampling και No-QoS

Η απόκλιση του BW-QoS-Reset-Sampling από τη βέλτιστη εκτέλεση No-QoS (5%) οφείλεται

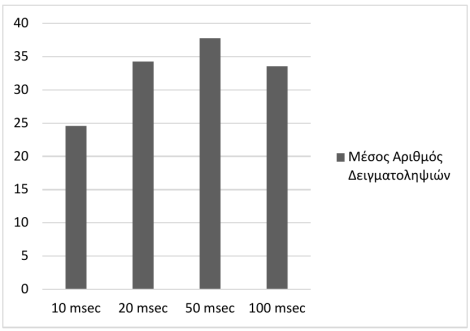
στο γεγονός ότι το σημείο εκκίνησης και σε αυτόν τον αλγόριθμο είναι τα 19 ways. Ωστόσο, δεδομένου ότι δεν γνωρίζουμε το προφίλ των εφαρμογών και ότι οι συνεκτελέσεις της Κατηγορίας Α παρουσιάζουν βέλτιστη εκτέλεση στα 19 ways, η συγκεκριμένη παραδοχή είναι αναγκαστική. Επιπλέον, η δειγματοληψία μπορεί να μην πετυχαίνει πάντα το βέλτιστο τμήμα cache που απαιτείται καθώς αλλαγές φάσης και μεταβατικές περίοδοι της εκτέλεσης που μπορούν να συμβούν κατά τη διάρκεια της δειγματοληψίας, είναι πιθανό να δώσουν εσφαλμένη εντύπωση για την ευαισθησία της εφαρμογής ως προς την cache. Άρα να επιλεχθεί ακατάλληλο σημείο. Οι παραπάνω παρατηρήσεις μπορούν να γίνουν εμφανείς στο Σχήμα 5.4. Συγκριτικά με την εκτέλεση No-QoS ο μηχανισμός BW-QoS-Reset-Sampling παραχωρεί μεγαλύτερο τμήμα cache στο HP GemsFDTD1. Μάλιστα, τη χρονική στιγμή 275 sec βλέπουμε ότι το αρχικό σημείο της φάσης της εκτέλεσης είναι τα 19 ways ενώ κατά τη No-QoS παραχωρούνται κατά μέσο όρο 8-9 ways. Επομένως αυτή η χρονική διαφορά είναι πιθανόν να οφείλεται σε επιλογή μη κατάλληλου σημείου επαναφοράς κατά τη δειγματοληψία.

Στη συνέχεια μελετάμε και αναλύουμε την επίδραση των παραμέτρων του μηχανισμού στην επίδοση της HP εφαρμογής. Οι παράμετροι που θα μελετήσουμε αφορούν τη χρονική διάρκεια της δειγματοληψίας, δηλαδή τη χρονική διάρκεια επιβολής κάθε δείγματος, την παράμετρο σύμφωνα με την οποία ο μηχανισμός αντιλαμβάνεται αλλαγή φάσης (HP_BANDWIDTH) και το κατώφλι συνολικού bandwidth που σηματοδοτεί κορεσμό (BW_LIMIT).

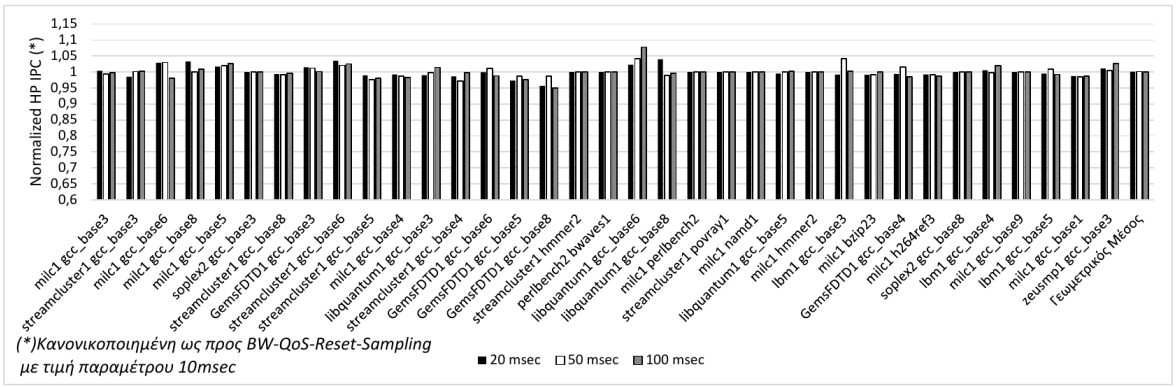
Για την ανάλυση της χρονικής διάρκειας της δειγματοληψίας χρησιμοποιήσαμε τα workloads της Κατηγορίας Β που παρουσίασαν κορεσμό στο bandwidth γιατί μόνο σε αυτές τις συνεκτελέσεις ενεργοποιείται η δειγματοληψία. Για τη μελέτη των παραμέτρων που σχετίζονται με τις μετρήσεις bandwidth επιλέχθηκε ένα αντιπροσωπευτικό υποσύνολο των $50 + 70 = 120$ συνολικών συνδυασμών των Κατηγοριών Α & Β που ισούται με το 25% αυτών, δηλαδή με 30 περιπτώσεις συνεκτελέσεων.

5.3 Μελέτη της Χρονικής Διάρκειας της Δειγματοληψίας

Στο Σχήμα 5.6 παραθέτουμε τα αποτελέσματα της ανάλυσης μας ως προς τη χρονική διάρκεια της δειγματοληψίας. Συγκεκριμένα αναπαριστούμε το μέσο IPC της HP εφαρμογής για κάθε τιμή της χρονικής διάρκειας επιβολής κάθε δείγματος κανονικοποιημένο ως προς το IPC του μηχανισμού με χρονική διάρκεια επιβολής δείγματος ίση με 10 msec. Καθώς ο μηχανισμός δειγματοληψίας ενεργοποιείται μόνο όταν υπάρχει κορεσμός BW, για την μελέτη των παραμέτρων του μηχανισμού χρησιμοποιούμε μόνο τα 35 workloads της Κατηγορίας Β όπου εμφανίζονται αυτά τα φαινόμενα. Δοκιμάζουμε τέσσερις διαφορετικές τιμές, κάθε δείγμα εφαρμόζεται για 10 msec, 20 msec, 50 msec και 100 msec, δηλαδή η χρονική διάρκεια της δειγματοληψίας είναι 100 msec, 200 msec, 500 msec και 1 sec, αντίστοιχα. Διαπιστώνουμε ότι οι διαφορές μεταξύ των περιπτώσεων είναι αμελητέες. Επομένως, όλες οι επιλογές είναι ισοδύναμες αφού ο τελικός γεωμετρικός μέσος είναι περίπου 1 σε όλες τις περιπτώσεις. Ωστόσο, βασιζόμενοι στο μέσο αριθμό δειγματοληψιών ανά τιμή χρονικής διάρκειας κάθε δείγματος (Σχήμα 5.5), επιλέγουμε την τιμή 10 msec ως την καταλληλότερη επειδή με αυτή έχουμε το λιγότερο μέσο αριθμό δειγματοληψιών.



Σχήμα 5.5: Μέσος όρος Δειγματοληψιών ανά τιμή Χρονικού Διαστήματος επιβολής του Δείγματος



Σχήμα 5.6: IPC HP Εφαρμογών ανά τιμή Χρονικού Διαστήματος επιβολής του Δείγματος

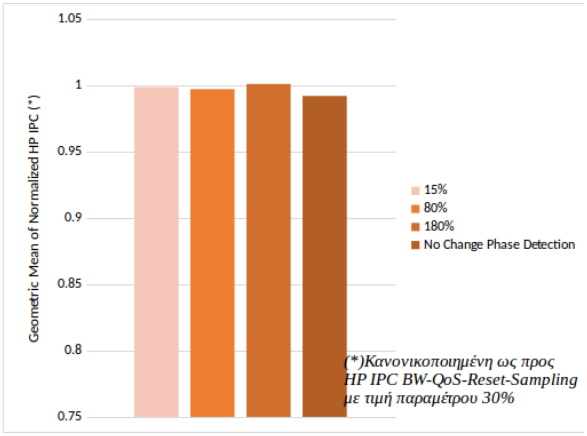
Στις επόμενες ενότητες αναλύουμε την ευαισθησία του μηχανισμού BW-QoS-Reset-Sampling στις παραμέτρους που αφορούν τη διαχείριση του bandwidth. Όπως έχουμε περιγράψει ο μηχανισμός χρησιμοποιεί δύο παραμέτρους για να καταλάβει τον κορεσμό του bandwidth (BW_LIMIT) και την αλλαγή φάσης (HP_BANDWIDTH). Στο αντιπροσωπευτικό δείγμα και των δύο κατηγοριών, επαναλαμβάνουμε τις συνεκτελέσεις μεταβάλλοντας τις τιμές των παραπάνω παραμέτρων.

5.4 Μελέτη στην Παράμετρο του Μηχανισμού που καθορίζει την Αλλαγή Φάσης

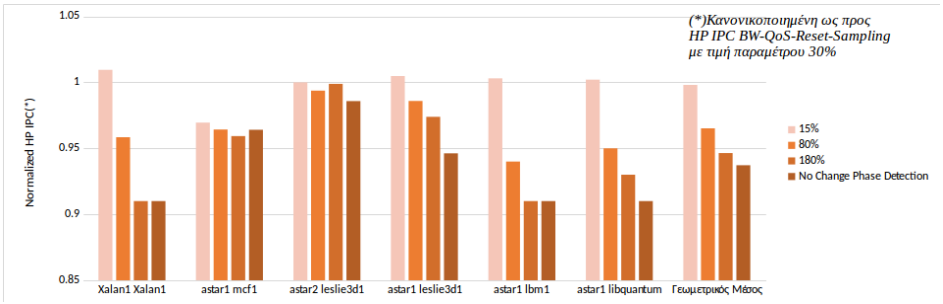
Ως προς την παράμετρο για αλλαγή φάσης δεν παρατηρούμε ουσιαστική ευαισθησία του μηχανισμού (Σχήμα 5.7) και αυτό συμβαίνει για δύο λόγους. Πρώτον, το δείγμα που επιλέχθηκε δεν έχει, παρά μόνο τρεις, περιπτώσεις όπου η εφαρμογή υψηλής προτεραιότητας έχει διακριτές φάσεις. Αυτό επιβεβαιώνεται από την ανάλυση που έχουμε κάνει αλλά και από το γεγονός ότι ακόμα και χωρίς αυτή τη λειτουργία ο αλγόριθμος επιτυγχάνει την επιθυμητή εκτέλεση (βλ. *No Change Phase Detection*). Άλλωστε, όταν το IPC χειροτερεύσει είτε πρόκειται για αλλαγή φάσης είτε όχι ο αλγόριθμος οφείλει να κάνει επανεκκίνηση. Επομένως αφαιρώντας αυτή τη λειτουργία ο αλγόριθμος κάνει επαναφορές στην αρχική κατάσταση μόνο όταν το IPC χειροτερεύει. Εφόσον στις περισσότερες περιπτώσεις (Σχήμα 5.7) δεν παρατηρούμε διαφορές τότε σημαίνει ότι στις περιπτώσεις αυτές είτε δεν υπάρχουν διακριτές φάσεις (π.χ. GemsFDTD1) είτε κατά τις αλλαγές φάσεις το IPC θα μειωθεί ή αν αυξηθεί η μη επαναφορά στο αρχικό σημείο δεν θα προκαλέσει πρόβλημα στην εκτέλεση.

Ως επέκταση της εργασίας θα ήταν σημαντικό να βρούμε περισσότερες περιπτώσεις με διαφο-

ρετικές διακριτές φάσεις. Ωστόσο, στην συγκεκριμένη πλατφόρμα με τις συγκεκριμένες εφαρμογές που εξετάζουμε, παραθέτουμε έξι περιπτώσεις (Σχήμα 5.8) στις οποίες παρατηρούμε ευαισθησία ως προς την παράμετρο HP_BANDWIDTH, η οποία μάλιστα είναι γραμμική. Παρατηρούμε ότι καθώς αυξάνουμε το ποσοστό που θα πρέπει να αυξηθεί το bandwidth της HP εφαρμογής ώστε ο μηχανισμός να θεωρήσει αλλαγή φάσης, το IPC της υψηλής προτεραιότητας εφαρμογής κανονικοποιημένο ως προς το IPC του μηχανισμού με τιμή HP_BANDWIDTH = 30% φθήνει έως και 7%. Ως τιμή της παραμέτρου HP_BANDWIDTH επομένως επιλέγουμε ποσοστό 30%.



Σχήμα 5.7: Γεωμετρικός Μέσος HP IPC για τιμές παραμέτρου Αλλαγής Φάσης



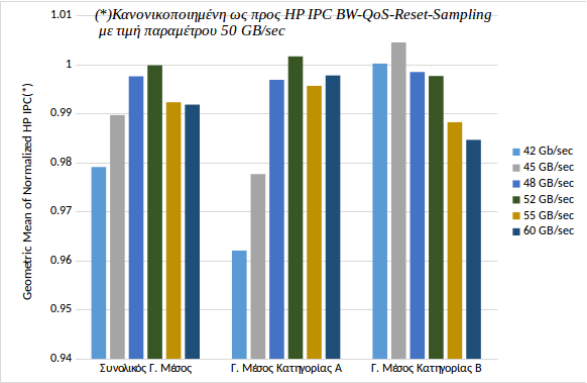
Σχήμα 5.8: IPC HP Εφαρμογών ανά τιμή παραμέτρου Αλλαγής Φάσης

5.5 Μελέτη στην Παράμετρο του Μηχανισμού που καθορίζει τον Κορεσμό στο Bandwdith

Στην συνέχεια, επιλέγοντας βάσει των παραπάνω ποσοστό αλλαγής φάσης 30% εκτελούμε τους συνδυασμούς για τις διάφορες τιμές του ορίου bandwidth (BW_LIMIT) βάση του οποίου μία συνεκτέλεση θεωρείται ότι προκαλεί κορεσμό στο bandwidth ή όχι. Αξιολογούμε το μηχανισμό ως προς εφτά διαφορετικές τιμές, δηλαδή για όριο bandwidth 43, 45, 48, 50, 52, 55, 60 GB/sec. Οι τιμές αυτές αντιστοιχούν στο 65%, 70%, 75%, 80%, 85%, 90% και 95% της ονομαστικής τιμής bandwidth του επεξεργαστή. Περιμένουμε ότι (α) μείωση του ορίου bandwidth θα βλάψει την εκτέλεση των εφαρμογών που ανήκουν στην Κατηγορία A (πολιτική No-QoS χειρότερη από την CT-QoS) και (β) αύξηση του ορίου bandwidth θα βλάψει την εκτέλεση κάποιων περιπτώσεων της Κατηγορίας B.

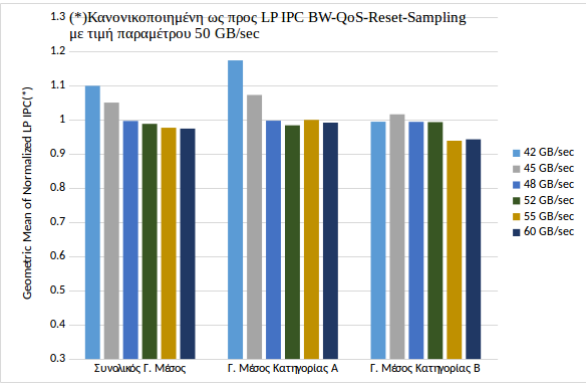
Όσον αφορά το (α) μειώνοντας το όριο bandwidth είναι πιθανό περιπτώσεις που είναι υψηλά ευαίσθητες ως προς cache να θεωρηθούν ότι προκαλούν κορεσμό και άρα να ενεργοποιηθεί ο μηχανισμός δειγματοληψίας έχοντας απρόβλεπτη συμπεριφορά. Στο Σχήμα 5.9, όπου αναπαρίσταται ο γεωμετρικός μέσος των κανονικοποιημένων τιμών του IPC της HP εφαρμογής για τις διάφορες τιμές της παραμέτρου ως προς το IPC της εφαρμογής με τιμή παραμέτρου 50 GB/sec, παρατηρούμε ότι μειώνεται η επίδοση των συνεκτελέσεων της Κατηγορίας Α. Από την άλλη πλευρά, χαμηλή τιμή ορίου ευνοεί τις συνεκτελέσεις της Κατηγορίας Β (β) γιατί ενεργοποιείται χρονικά συντομότερα η δειγματοληψία.

Αξιολογώντας το συνολικό γεωμετρικό μέσο παρατηρούμε ότι καλύτερες εκτελέσεις μας προσφέρουν οι τιμές 48, 50, 52 GB/sec. Στο Σχήμα 5.9 οι γεωμετρικοί μέσοι κάθε κατηγορίας που επιβεβαιώνουν τις αρχικές μας υποθέσεις (α) και (β). Από την ανάλυση αυτή θα επιλέξουμε για τον μηχανισμό την τιμή 50 GB/sec για την τιμή ορίου bandwidth.



Σχήμα 5.9: IPC HP Εφαρμογών ανά τιμή Ορίου Κορεσμού Bandwidth

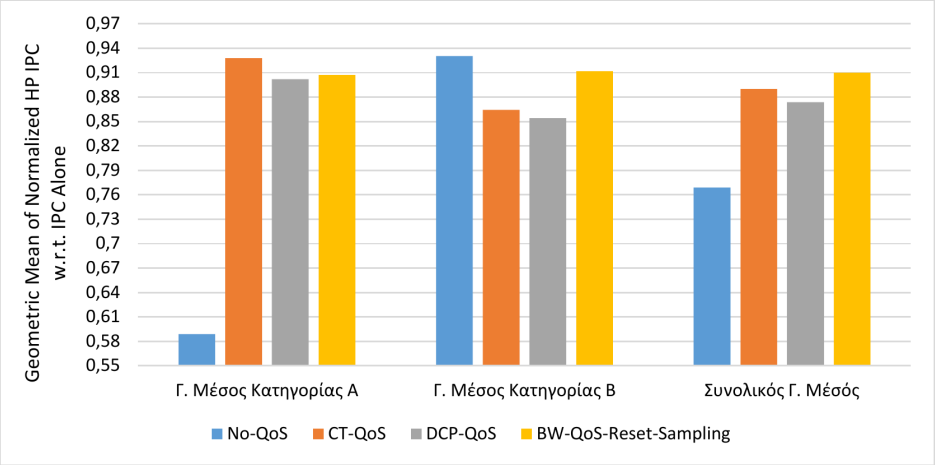
Τέλος παραθέτουμε και τους γεωμετρικούς μέσους της επίδοσης των εφαρμογών χαμηλής προτεραιότητας. Βλέπουμε ότι μικρές τιμές ορίου bandwidth ευνοούν την εκτέλεση των εφαρμογών χαμηλής προτεραιότητας. Αυτό σε συνδυασμό με τα παραπάνω συμπεράσματα επιβεβαιώνει ότι χαμηλές τιμές ορίου bandwidth προκαλούν περισσότερες στο πλήθος δειγματοληψίες που δεν ενδείκνυνται στις περιπτώσεις Κατηγορίας Α. Αντιθέτως, στην Κατηγορία Β δεν θα λέγαμε ότι έχουμε μία σαφή γραμμική συμπεριφορά. Το μόνο σίγουρο είναι ότι οι τιμές 55 GB/sec και 60 GB/sec δεν είναι κατάλληλες αφού σε συνδυασμό με τα παραπάνω δεν ευνοούν ούτε την HP εφαρμογή αλλά ούτε και τις υπόλοιπες LP.



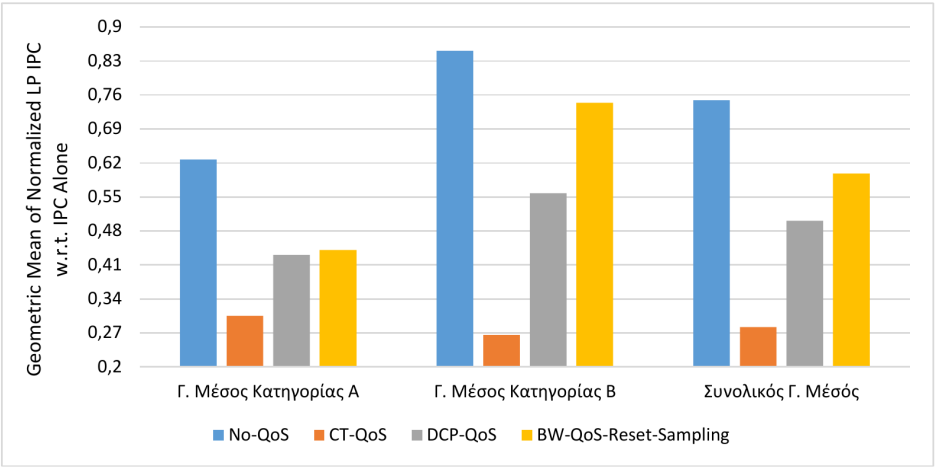
Σχήμα 5.10: IPC LP Εφαρμογών ανά τιμή Ορίου Κορεσμού Bandwidth

5.6 Αξιολόγηση του Μηχανισμού BW-QoS-Reset-Sampling - Τελικά αποτελέσματα

Εκτελώντας τους 120 συνδυασμούς που έχουμε μελετήσει και στις παραπάνω ενότητες, παραθέτουμε τα τελικά αποτελέσματα των No-QoS, CT-QoS, DCP-QoS και BW-QoS-Reset-Sampling πολιτικών συνεκτέλεσης. Ο μηχανισμός BW-QoS-Reset-Sampling βελτιώνει κατά, περίπου, 5% την εκτέλεση της υψηλής προτεραιότητας εφαρμογής σε σχέση με τον μηχανισμό DCP-QoS και περίπου 2.5% σε σχέση με τη CT-QoS εκτέλεση. Οι δύο μηχανισμοί έχουν αμελητέα διαφορά (λιγότερο του 1%) στην προστασία της επίδοσης της HP εφαρμογής για του συνδυασμούς της Κατηγορίας A και τη βελτιώνουν κατά 55% συγκριτικά με τη No-QoS πολιτική. Επιπλέον, στην Κατηγορία A και οι δύο μηχανισμοί εκτελούν κατά μέσο όρο την HP εφαρμογή στο 98% του χρόνου κατά CT-QoS (2% απόκλιση). Ωστόσο, στις συνεκτελέσεις της κατηγορίας B ο μηχανισμός BW-QoS-Reset-Sampling παρέχει 8% καλύτερη εκτέλεση από την DCP-QoS υλοποίηση αποκλίνοντας περίπου 2% από την βέλτιση εκτέλεση του No-QoS. Μάλιστα ο μηχανισμός BW-QoS-Reset-Sampling βελτιώνει κατά 12% τις συνεκτελέσεις της Κατηγορίας B που προκαλούν κορεσμό στο bandwdith σε σχέση με τον DCP-QoS μηχανισμό.



Σχήμα 5.11: Γεωμετρικός Μέσος Τελικών Αποτελεσμάτων για τις HP Εφαρμογές

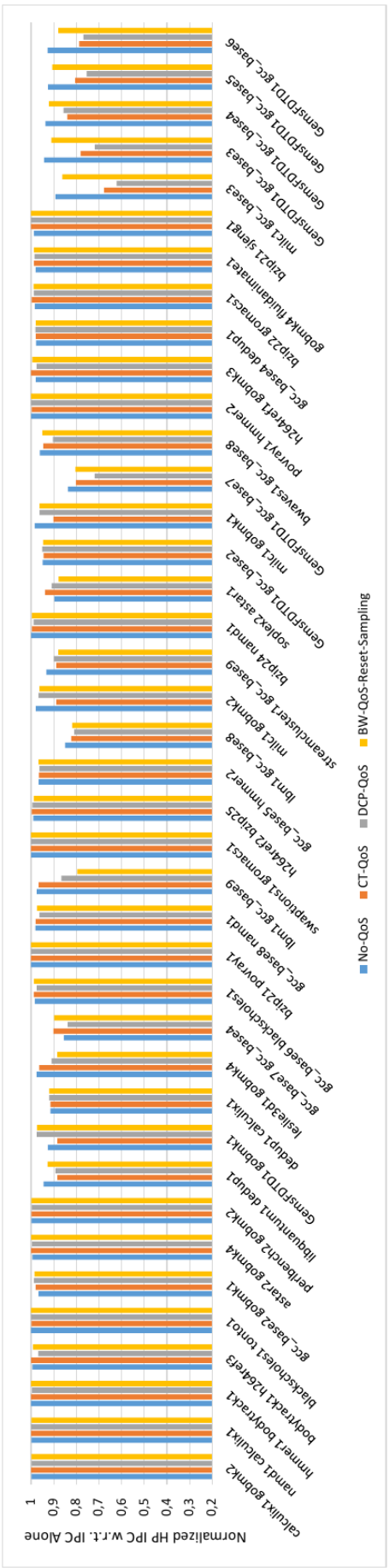


Σχήμα 5.12: Γεωμετρικός Μέσος Τελικών Αποτελεσμάτων για τις LP Εφαρμογές

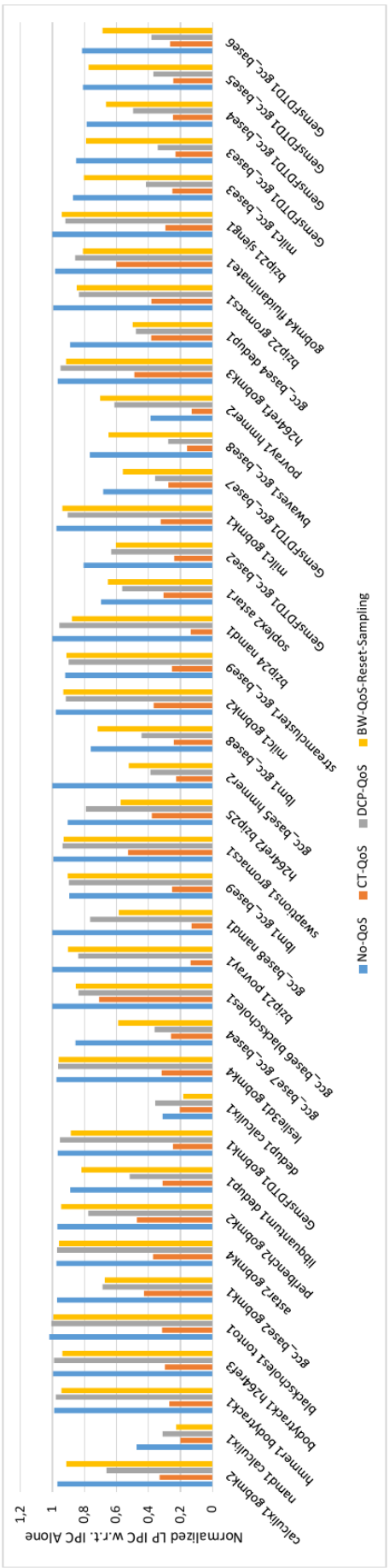
Σχετικά με την επίδοση των εφαρμογών χαμηλής προτεραιότητας με χρήση του μηχανισμού

BW-QoS-Reset-Sampling παρατηρούμε βελτίωση 16% συγκριτικά με την προηγούμενη υλοποίηση, DCP-QoS, και 20% απόκλιση σε σχέση με την καλύτερη εκτέλεση των LP εφαρμογών που προσφέρει η No-QoS πολιτική. Μάλιστα και εδώ βλέπουμε ότι ο μηχανισμός που υλοποιούμε διαχειρίζεται καλύτερα τις περιπτώσεις της Κατηγορίας Β βελτιώνοντας ταυτόχρονα και την HP και τις LP εφαρμογές. Συνολικά παρατηρούμε ότι ο μηχανισμός μας εξασφαλίζει την εκτέλεση της υψηλής προτεραιότητας εφαρμογής με 9% απόκλιση από την απομονωμένη εκτέλεση. Ωστόσο, η απόκλιση αυτή όπως εξηγούμε και σε προηγούμενες ενότητες οφείλεται και στον ανταγωνισμό και διαμοιρασμό άλλων κοινόχρηστων πόρων.

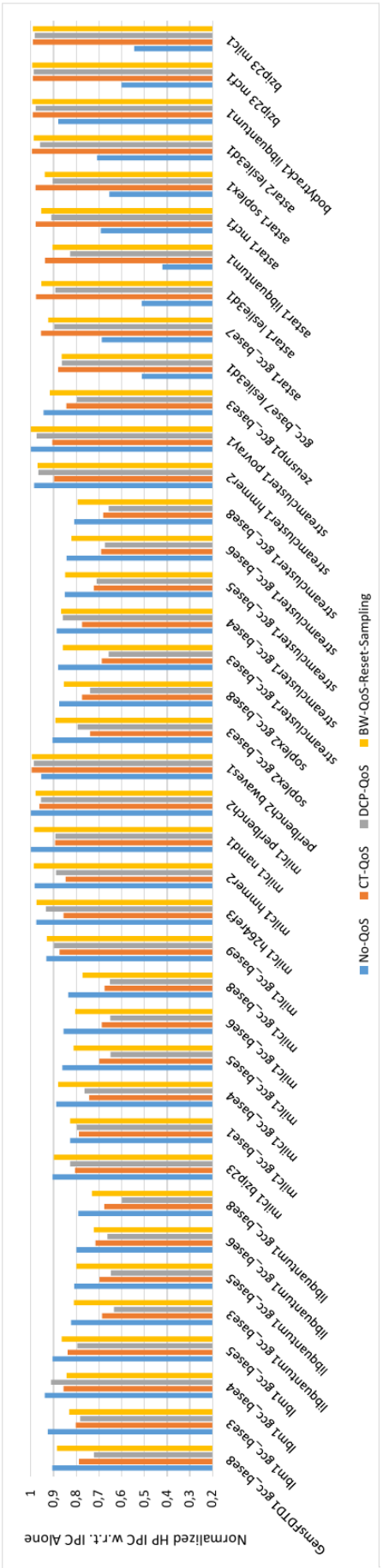
Στη συνέχεια παραθέτουμε αναλυτικά τα διαγράμματα με τις εκτελέσεις μας. Οι τιμές είναι κανονικοποιημένες ως προς το IPC της απομονωμένης εκτέλεσης. Σε κάθε σελίδα παραθέτουμε το διάγραμμα με την κανονικοποιημένη τιμή IPC της εφαρμογής με υψηλή προτεραιότητα και δίπλα το διάγραμμα με τις αντίστοιχες τιμές για τις εφαρμογές χαμηλής προτεραιότητας.



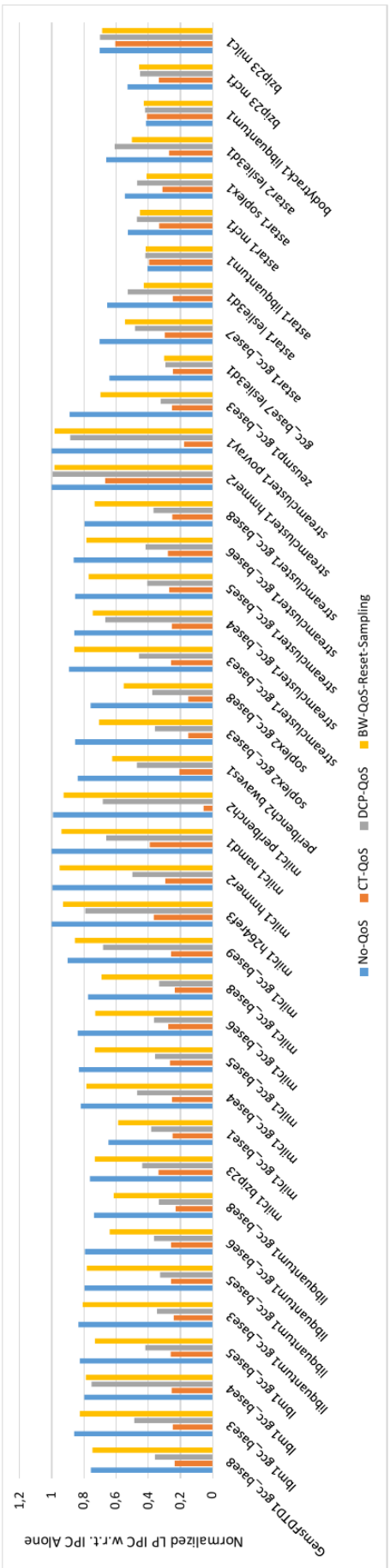
Σχήμα 5.13: Υψηλής Προτεραιότητας (I)



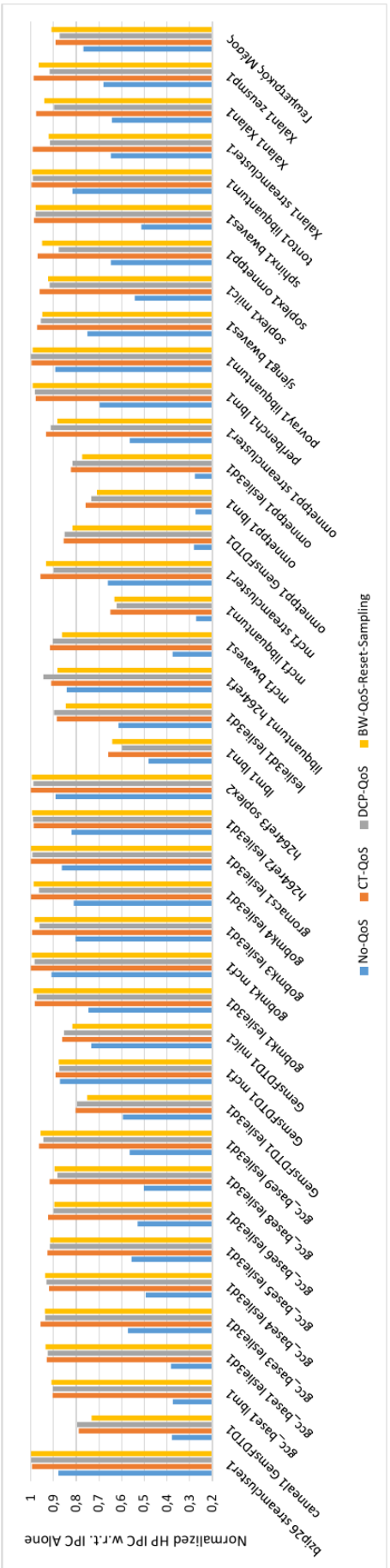
Σχήμα 5.14: Χαμηλής Προτεραιότητας (I)



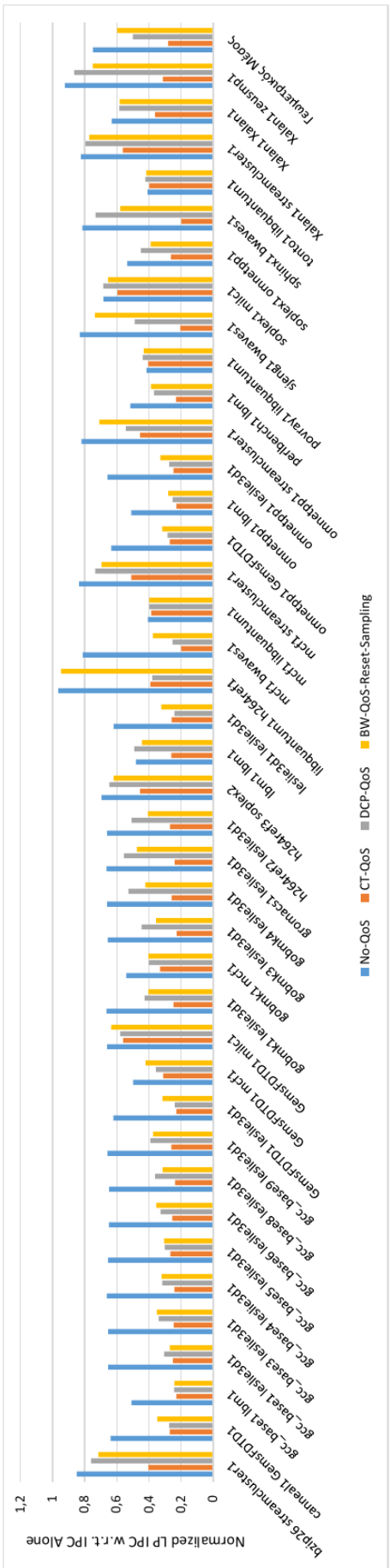
Σχήμα 5.15: Υψηλής Προτεραιότητας (II)



Σχήμα 5.16: Χαμηλής Προτεραιότητας (II)



Σχήμα 5.17: Υψηλής Προτεραιότητας (III)



Σχήμα 5.18: Χαμηλής Προτεραιότητας (III)

Κεφάλαιο 6

Επίλογος

6.1 Σύνοψη και Συμπεράσματα

Στην παρούσα διπλωματική εργασία μελετήσαμε τις δυνατότητες των τεχνολογιών που ανέπτυξε η Intel για την επίβλεψη και τον καταμερισμό της κοινόχρηστης μνήμης cache στους πυρήνες και στις εφαρμογές. Χρησιμοποιώντας τις τεχνολογίες αυτές υλοποιούμε και προτείνουμε ένα μηχανισμό ο οποίος καλείται να προστατεύσει την εκτέλεση μιας υψηλής προτεραιότητας εφαρμογής όταν αυτή συνεκτελείται με άλλες, δηλαδή να βελτιώσει την ποιότητα υπηρεσίας και τη χρησιμοποίηση του επεξεργαστή. Ο μηχανισμός αξιοποιεί κατά βάση μετρικές όπως το IPC και τη χρήση του διαύλου δεδομένων από τους πυρήνες, και λαμβάνει αποφάσεις προκειμένου να προστατεύσει την συνεκτέλεση από τον κορεσμό στο bandwidth παραχωρώντας ταυτόχρονα στην εφαρμογή υψηλής προτεραιότητας επαρκές τμήμα της cache.

Η μελέτη μας χρησιμοποιεί συνεκτελέσεις μονονηματικών εφαρμογών (single threaded processes) των σουιτών SPEC2006 και PARSEC. Αν και στη γενική περίπτωση η εφαρμογή επωφελείται από μεγάλο τμήμα cache, παρατηρήθηκαν συνεκτελέσεις κατά τις οποίες η παραπάνω παραδοχή χειροτέρευε την επίδοση κυρίως εξαιτίας του κορεσμού στο bandwidth. Ο μηχανισμός αντιμετωπίζει αυτές τις περιπτώσεις πραγματοποιώντας δειγματοληψία για την επιλογή του αρχικού σημείου χωρίς να είναι απαραίτητη η ανάγκη για κατάλληλο υλικό που διαχειρίζεται το ρυθμό αιτήσεων προς τη μνήμη από τους επεξεργαστές, δηλαδή την τεχνολογία MBA.

Προτείνουμε επομένως τον μηχανισμό BW-QoS-Reset-Sampling ο οποίος βελτιώνει κατά 55% την επίδοση των ευαίσθητων ως προς την cache εφαρμογών σε σχέση με την επίδοσή τους όταν όλες οι εφαρμογές μοιράζονται και έχουν ισότιμη πρόσβαση σε ολόκληρη την LLC (No-QoS). Επιπλέον, ο μηχανισμός πετυχαίνει βελτίωση (11.5% σε σχέση με τη CT-QoS) στην εκτέλεση των εφαρμογών που δημιουργούν κορεσμό στο bandwidth με απόκλιση 2% από την βέλτιστη εκτέλεσή τους που στην συγκεκριμένη περίπτωση παρέχει η No-QoS πολιτική. Συνολικά στην κατηγορία B, σε σχέση με την CT-QoS, ο μηχανισμός βελτιώνει τον χρόνο εκτέλεσής κατά 5%. Τέλος, ο μηχανισμός παρέχει επίδοση της HP εφαρμογής με απόκλιση 9% από την απομονωμένη εκτέλεσή της.

6.2 Αξιοποίηση της MBA τεχνολογίας

Λίγο πριν την ολοκλήρωση της διπλωματικής εργασίας, αποκτήσαμε πρόσβαση στους καινούργιους επεξεργαστές της Intel και συγκεκριμένα στον Intel 24 Core Xeon Platinum 8160 Server (Πίνακας 6.1), αρχιτεκτονικής Skylake-SP (Purley) , που υποστηρίζει την τεχνολογία MBA. Πραγματοποιήσαμε επιλεκτικά κάποιες συνεκτελέσεις, τα αποτελέσματα των οποίων παρουσιάζονται

στο Σχήμα 6.1. Επισημάνουμε ότι οι επεξεργαστές είναι διαφορετικής αρχιτεκτονικής από τους Broadwell στους οποίους βασιστήκαμε για την ανάπτυξη του μηχανισμού.

	Χαρακτηριστικά
Αρχιτεκτονική	Skylake-SP
αριθμός πυρήνων/νημάτων	24/48
βασική συχνότητα	2.10 GHz
LLC	33 MB (exlusive)
assosiativity	11
CMT - CAT - MBA	✓
MBA delay	linear (step 10%)

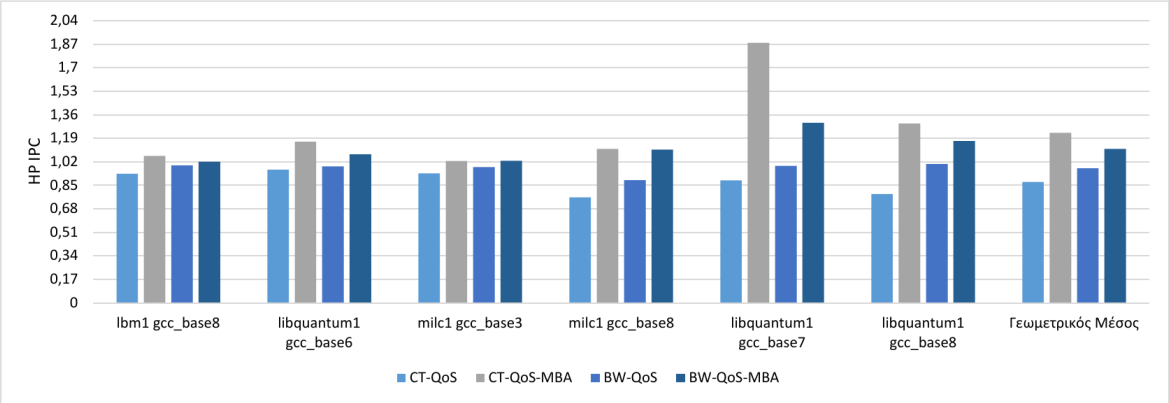
Πίνακας 6.1: Χαρακτηριστικά του επεξεργαστή Intel® 24 Core Xeon® Platinum 8160 Server

Αναφορικά με την παραπάνω αρχιτεκτονική, οι δύο βασικές διαφορές συγκριτικά με την αρχιτεκτονική Broadwell, είναι (α) ότι η LLC πλέον δεν είναι περιεκτική και (β) ότι χρησιμοποιείται MESH δίκτυο διασύνδεσης, δηλαδή τα κομμάτια (slices) της LLC στην υποδοχή (socket) επικοινωνούν μέσω ενός πλέγματος διασύνδεσης και όχι μέσω του κλασσικού δακτυλίου (buffer ring). Έτσι επιταχύνεται η επικοινωνία και η μεταφορά δεδομένων στον επεξεργαστή. Το γεγονός ότι η LLC δεν είναι περιεκτική επηρεάζει τη λειτουργία του μηχανισμού. Πλέον, η απομόνωση του τελευταίου επιπέδου cache μέσω της CAT τεχνολογίας οδηγεί στο σχηματισμό μιας τρίτης ανεξάρτητης και ιδιωτικής, ως προς τους εκάστοτε πυρήνες του CLOS, cache. Είναι επομένως δυνατόν να διατηρείται σταθερό το IPC μιας συνεκτέλεσης επειδή τα δεδομένα υπάρχουν στις L1 & L2 caches και έτσι ο μηχανισμός να μειώνει το τμήμα της L3 cache. Ως αποτέλεσμα, θα διώχνονται στοιχεία από την L3, κάτι που στη συνέχεια της εκτέλεσης θα μπορούσε να στοιχίσει σε όρους επίδοσης.

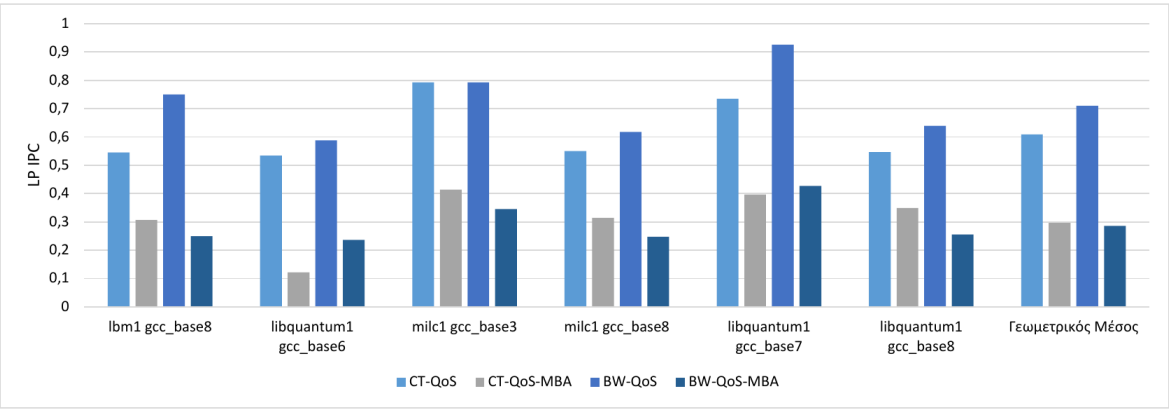
Τα πειράματα που εκτελούμε αφορούν τις εξής πολιτικές συνεκτέλεσεις και είναι κανονικοποιημένα ως προς τον χρόνο εκτέλεσης της No-QoS πολιτικής :

- No-QoS, όπως έχει οριστεί στις προηγούμενες ενότητες.
- CT-QoS, όπως έχει οριστεί στις προηγούμενες ενότητες.
- CT-QoS-MBA, σε αυτή την πολιτική παραχωρούμε τα N-1 ways της cache στην υψηλής προτεραιότητας εφαρμογής και επιβάλουμε 90% καθυστέρηση στο ρυθμό αιτήσεων των πυρήνων στους οποίους έχουν χρονοδρομολογηθεί οι εφαρμογές χαμηλής προτεραιότητας.
- BW-QoS, ο μηχανισμός που υλοποιούμε.
- BW-QoS-MBA, ο μηχανισμός που υλοποιούμε με επιβολή 90% καθυστέρηση στο ρυθμό αιτήσεων των πυρήνων στους οποίους έχουν χρονοδρομολογηθεί οι εφαρμογές χαμηλής προτεραιότητας.

Παρατηρούμε ότι σε κάποιες περιπτώσεις ο μηχανισμός που υλοποιούμε χωρίς τη χρήση MBA παρουσιάζει όμοια αποτελέσματα σε σχέση με την εκτέλεση CT-QoS-MBA που είναι η βέλτιστη σε όλα τα πειράματα. Αυτό μας αποδεικνύει ότι το BW-QoS λειτουργεί σωστά και δεν έχει πάντα ανάγκη την τεχνολογία MBA. Ωστόσο σε συγκεκριμένες περιπτώσεις, όπως για παράδειγμα στη συντεκτέλεση milc1-gcc_base8 που έχουμε αναλύσει, παρατηρούμε ότι ο συνδυασμός του μηχανισμού μας και της MBA τεχνολογίας παρουσιάζει καλύτερη εκτέλεση από την πολιτική No-QoS,



Σχήμα 6.1: IPC HP εφαρμογής με χρήση MBA τεχνολογίας



Σχήμα 6.2: IPC LP εφαρμογών με χρήση MBA τεχνολογίας

δηλαδή αντιμετωπίζεται ο κορεσμός στο bandwidth και παράλληλα αποδίδεται η ελάχιστη δυνατή cache στην εφαρμογή υψηλής προτεραιότητας.

Ωστόσο, αν και η χρήση του MBA βελτιστοποιεί την επίδοση της υψηλής προτεραιότητας εφαρμογής, μειώνει σημαντικά την εκτέλεση των υπόλοιπων εφαρμογών. Είναι και εδώ επομένως ανάγκη η δυναμική διαχείριση της πρόσβασης των εφαρμογών στο bandwidth. Μια τέτοια λογική θα μπορούσε να ενσωματωθεί στον μηχανισμό μας, επεκτείνοντας τη λειτουργία του και στο επίπεδο επίβλεψης και διαμοιρασμού του bandwidth.

6.3 Μελλοντικές Κατευθύνσεις

Η συγκεκριμένη εργασία θα μπορούσε να επεκταθεί σε πολλές κατευθύνσεις. Μια πρώτη επέκταση του μηχανισμού, όπως αναφέρουμε και στην προηγούμενη ενότητα, αφορά την ενσωμάτωση της νέας τεχνολογίας του MBA, ούτως ώστε, να αντιμετωπίζεται επιτυχώς ο κορεσμός στο bandwidth αλλά και να βελτιώνεται ακόμα περισσότερο σε σχέση με την απλή CT-QoS πολιτική η ποιότητα υπηρεσίας. Μάλιστα, αντίστοιχα με τη λογική της δυναμικής εκχώρηση μνήμης LLC θα μπορούσε να επεκταθεί ο μηχανισμός ώστε επίσης δυναμικά να ορίζει και τον περιορισμό του bandwidth των πυρήνων με χαμηλής προτεραιότητας εφαρμογές. Στόχος θα ήταν η πλήρης απομόνωση των κοινόχρηστων πόρων on-chip στις εφαρμογές και η βέλτιστη εκχώρηση πόρων που καλύπτουν τις υπολογιστικές απαιτήσεις της HP εφαρμογής. Μια τέτοια υλοποίηση θα βελτίωνε τη δικαιοσύνη στο σύστημα καθώς οι LP εφαρμογές θα καταλάμβαναν περισσότερους πόρους από

την υλοποίηση του BW-QoS-Reset-Sampling που προτείνουμε.

Επιπλέον, ενδιαφέρον θα είχε η χρήση μιας υψηλής προτεραιότητας πολυνηματικής εφαρμογής με στόχο την ανάδειξη και ορθολογική χρήση των CLOSs, τα οποία αποτελούν επίσης έναν πόρο του συστήματος που δεν είναι απεριόριστος. Πιο αναλυτικά, σε μία τέτοια υλοποίηση θα είχε νόημα η αντιστοίχιση των CLOSs στα νήματα της εφαρμογής και η τμηματοποίηση της LLC σε περισσότερα τμήματα. Ειδικά στην περίπτωση που ο αριθμός των νημάτων είναι μεγαλύτερος του αριθμού των CLOSs ανακύπτει το ερώτημα αντιστοίχισης περισσότερων από ένα νήματα σε ένα CLOSs. Μια τέτοια ανάγκη θα απαιτούσε ενδεχομένως κάποια αρχική ανάλυση της πολυνηματικής εφαρμογής ώστε να διαπιστωθεί η ευαισθησία και η αλληλεπίδραση των νημάτων όταν αυτά ομαδοποιούνται μαζί σε ίδια CLOSs.

Τέλος, δεδομένου ότι η τεχνολογία hyperthreading των επεξεργαστών της Intel πιστεύεται ότι προσφέρει βελτίωση της απόδοσης έως και 30% και σε συνδυασμό με την L2CAT τεχνολογία που περιγράφουμε στην ενότητα 2.3.1 θα ήταν χρήσιμη η επέκταση του μηχανισμού ούτως ώστε αξιοποιούνται και οι “δίδυμοι” (sibling) πυρήνες.

Βιβλιογραφία

[1] Dhruba Chandra, Fei Guo, Seongbeom Kim, and Yan Solihin. Predicting inter-thread cache contention on a chip multi-processor architecture. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*, HPCA '05, pages 340–351, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Ravi Iyer. Cqos: A framework for enabling qos in shared caches of cmp platforms. In *Proceedings of the 18th Annual International Conference on Supercomputing*, ICS '04, pages 257–266, New York, NY, USA, 2004. ACM.

[3] Seongbeom Kim, Dhruba Chandra, and Yan Solihin. Fair cache sharing and partitioning in a chip multiprocessor architecture. In *Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques*, PACT '04, pages 111–122, Washington, DC, USA, 2004. IEEE Computer Society.

[4] Ioannis Papadakis, Konstantinos Nikas, Vasileios Karakostas, Georgios I. Goumas, and Nectarios Koziris. Improving qos and utilisation in modern multi-core servers with dynamic cache partitioning. In *Proceedings of the Joined Workshops COSH 2017 and VisorHPC 2017, COSH/VisorHPC@HiPEAC 2017, Stockholm, Sweden, January 24, 2017.*, pages 21–26. TUM Library, 2017.

[5] David Lo, Liqun Cheng, Rama Govindaraju, Parthasarathy Ranganathan, and Christos Kozyrakis. Heracles: Improving resource efficiency at scale. In *Proceedings of the 42th Annual International Symposium on Computer Architecture*, 2015.

[6] Moinuddin K. Qureshi and Yale N. Patt. Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, pages 423–432, Washington, DC, USA, 2006. IEEE Computer Society.

[7] Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, and Mary Lou Soffa. Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, pages 248–259, New York, NY, USA, 2011. ACM.

[8] Alexandra Fedorova, Margo Seltzer, and Michael D. Smith. 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 '07, pages 25–38, Washington, DC, USA, 2007. IEEE Computer Society.

[9] Andrew Herdrich, Ramesh Illikkal, Ravi Iyer, Don Newell, Vineet Chadha, and Jaideep Moses. Rate-based qos techniques for cache/memory in cmp platforms. In *Proceedings of the 23rd International Conference on Supercomputing*, ICS '09, pages 479–488, New York, NY, USA, 2009. ACM.

[10] Henry Cook, Miquel Moreto, Sarah Bird, Khanh Dao, David A. Patterson, and Krste Asanovic. A hardware evaluation of cache partitioning to improve utilization and energy-efficiency while preserving responsiveness. In *Proceedings of the 40th Annual International Symposium on Computer Architecture*, ISCA '13, pages 308–319, New York, NY, USA, 2013. ACM.

[11] E. Verplanke J. Gasparakis, S.u Ranganath and P. Autee. Deterministic network functions virtualization with intel® resource director technology. White paper, Intel Corporation, 2017.

[12] Alan Jay Smith. Cache memories. *ACM Comput. Surv.*, 14(3):473–530, September 1982.

[13] Andrew Herdrich, Edwin Verplanke, Priya Autee, Ramesh Illikkal, Chris Gianos, Ronak Singhal, and Ravi Iyer. Cache qos: From concept to reality in the intel® xeon® processor e5-2600 v3 product family. In *HPCA*, 2016.

[14] T. J Heller R. Matick and M. Ignatowski. Analytical analysis of finite cache penalty and cycles per instruction of a multiprocessor memory hierarchy using miss rates and queuing theory. *IBM Journal Of Research And Development*, 45(6):819–842, 2001.

[15] George Taylor, Peter Davies, and Michael Farmwald. The tlb slice—a low-cost high-speed address translation mechanism. *SIGARCH Comput. Archit. News*, 18(2SI):355–363, May 1990.

[16] Brian K. Bray, William L. Lunch, and Michael J. Flynn. Page allocation to reduce access time of physical caches. Technical report, Stanford University, Stanford, CA, USA, 1990.

[17] Edouard Bugnion, Jennifer M. Anderson, Todd C. Mowry, Mendel Rosenblum, and Monica S. Lam. Compiler-directed page coloring for multiprocessors. *SIGPLAN Not.*, 31(9):244–255, September 1996.

[18] Ι. Παπαδάκης. Βελτίωση Ποιότητας Υπηρεσίας με Τεχνικές Διαμοιρασμού Κρυφής Μνήμης. Διπλωματική Εργασία, Εθνικό Μετσόβιο Πολυτεχνείο, 2016.

[19] Why Intel added cache partitioning in Broadwell. https://danluu.com/intel-cat/. [Online; accessed 20-January-2018].