



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

1η ΑΣΚΗΣΗ

Αχλάτης Στέφανος-Σταμάτης (03116149)

<sachlatis@gmail.com>

Απρίλιος 2020

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1. Σκοπός της Άσκησης
2. Εργαλεία Pin Tool
3. PARSEC benchmarks
4. Προσομοίωση Ιεραρχίας Μνήμης
5. Καθυστερήσεις (Latencies)
6. Γενικές Παρατηρήσεις
7. Γενικά για το Α Μέρος
8. Έλεγχος Παραμέτρων L1-Cache
9. Έλεγχος Παραμέτρων L2-Cache
10. Έλεγχος Παραμέτρων TLB
11. Prefetching Policy για την L2-cache και έλεγχος παρ
12. Συμπεράσματα Α μέρους
13. Γενικά για το Β Μέρος
14. Έλεγχος Παραμέτρων L1-Cache
15. Έλεγχος Παραμέτρων L2-Cache
16. Έλεγχος Παραμέτρων TLB
17. Συμπεράσματα Β μέρους
18. Παραρτημα
19. Βιβλιογραφία

1.Σκοπός της Άσκησης

Η παρούσα άσκηση έχει ως αντικείμενο τη μελέτη της επίδρασης διαφόρων παραμέτρων της ιεραρχίας μνήμης στην απόδοση ενός συνόλου εφαρμογών. Πιο συγκεκριμένα, θα μελετηθεί η επίδραση των βασικότερων παραμέτρων ιεραρχίας κρυφής μνήμης (cache memory) στην απόδοση 10 διαφορετικών μετροπρογραμμάτων (benchmarks). Οι παράμετροι αυτές είναι το μέγεθος της κρυφής μνήμης (cache size), η συσχετιστικότητα (associativity) και το μέγεθος του μπλοκ της κρυφής μνήμης (cache block size). Στο δεύτερο μέρος της άσκησης θα μελετηθεί η δυναμική συμπεριφορά των εφαρμογών, εξετάζοντας τον τρόπο που οι διάφορες μετρικές απόδοσης μεταβάλλονται στο χρόνο.

2.Εργαλεία PIN TOOL

- Για την πραγματοποίηση της άσκησης χρησιμοποιήθηκε το εργαλείο PIN, το οποίο είναι ένα εργαλείο ανάλυσης εφαρμογών που αναπτύσσεται και συντηρείται από την Intel.
- Η χρήση του PIN δίνει τη δυνατότητα για dynamic binary instrumentation, το οποίο σημαίνει πως εισάγεται δυναμικά κώδικας κατά τη διάρκεια της εκτέλεσης των μετροπρογραμμάτων ανάμεσα στις εντολές της εφαρμογής αυτής έτσι ώστε να συλλεχθούν πληροφορίες σχετικές με την εκτέλεση, όπως ο συνολικός **αριθμός εντολών**, ο συνολικός **αριθμός ευστοχιών στην κρυφή μνήμη** κ.ο.κ.
- Η έκδοση του PIN στην οποία πραγματοποιήθηκε η προσομοίωση είναι η **PIN 98189**, σε **Ubuntu Linux 18.04** με έκδοση **πυρήνα 4.4**.

3.PARSEC benchmarks

Τα μετροπρογράμματα τα οποία χρησιμοποιήθηκαν για την εκτέλεση της άσκησης είναι ορισμένα PARSEC (Princeton Application Repository for Shared- Memory Computers) benchmarks, ανεπτυγμένα από το Princeton University. Αυτή η open-source σουίτα περιέχει 13 μετροπρογράμματα, τα οποία συνοδεύονται από τα κατάλληλα αρχεία εισόδου.

Τα 10 από αυτά χρησιμοποιήθηκαν για την προσομοίωση και αναφέρονται παρακάτω:

1. blackscholes
2. bodytrack
3. canneal
4. facesim
5. ferret
6. fluidanimate
7. freqmine
8. raytrace
9. swaptions
10. streamcluster

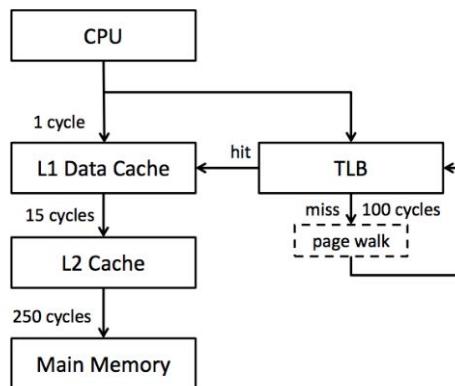
Στον κώδικα των μετροπρογραμμάτων έχουν οριστεί συγκεκριμένες περιοχές του κώδικα οι οποίες παρουσιάζουν μεγαλύτερο ενδιαφέρον προς μελέτη (Regions of Interest, ROI). Τέλος, για την προσομοίωση της εκτέλεσης των εφαρμογών χρησιμοποιήθηκε το pintool

simulator.cpp, το οποίο είναι σχεδιασμένο έτσι ώστε να ενεργοποιείται μόνο κατά τη διάρκεια των ROI.

4.Προσομοίωση Ιεραρχίας Μνήμης

Για τη συγκεκριμένη άσκηση προσομοιώνεται ένας in-order επεξεργαστής, γεγονός που σημαίνει πως όλες οι εντολές εκτελούνται σειριακά από τον επεξεργαστή, ακόμα και αν κάποια επόμενη εντολή είναι “έτοιμη” να εκτελεστεί νωρίτερα από κάποια προηγούμενή της. Η ιεραρχία μνήμης είναι inclusive ως προς τη διεπίπεδη cache που χρησιμοποιείται, όρος που επεξηγείται ακριβώς παρακάτω:

Ας θεωρήσουμε την ιεραρχία μνημών 2-επιπέδων όπου έχουμε τις L1-cache και L2-cache και όπου η L2 cache είναι inclusive. Αυτό σημαίνει πώς όλα τα blocks τα οποία βρίσκονται οποιαδήποτε χρονική στιγμή στην L1-cache θα πρέπει υποχρεωτικά να βρίσκονται και στην L2-cache. Γενικότερα, μια lower level cache λέγεται inclusive μιας higher level cache αν περιέχει όλα της τα blocks. Επιπλέον, προσομοιώνεται μια μνήμη μετάφρασης διευθύνσεων (Translation Lookaside Buffer, TLB), η οποία χρησιμοποιείται για τη μείωση του απαιτούμενου χρόνου πρόσβασης στην τοποθεσία μνήμης χρήστη, αποθηκεύοντας τις πρόσφατες μεταφράσεις της εικονικής μνήμης στην πραγματική μνήμη. Τέλος, θεωρούμε ότι η L1 είναι υλοποιημένη ως Virtually indexed Physically tagged (VIPT) cache, δηλαδή οι πρόσβασεις στο TLB και στην L1 cache επικαλύπτονται και πραγματοποιούνται παράλληλα. Σχηματικά, όλα τα παραπάνω προσομοιώνονται ως εξής:



5.Καθυστερήσεις (Latencies)

Για τον υπολογισμό της επίδοσης των εφαρμογών που χρησιμοποιούνται στις προσομοιώσεις, χρησιμοποιείται ένα απλό μοντέλο, όπου θεωρούμε ότι κάθε εντολή απαιτεί 1 κύκλο για την εκτέλεσή της (IPC=1). Επιπρόσθετα, οι εντολές που πραγματοποιούν πρόσβαση στη μνήμη (load/store) προκαλούν επιπλέον καθυστερήσεις ανάλογα με το αν οι μεταφράσεις διευθύνσεων βρίσκονται στο TLB και με το πού βρίσκονται τα δεδομένα τους. Συνολικά διακρίνουμε τις εξής περιπτώσεις:

- 1.TLB hit: 0 cycles (η πρόσβαση πραγματοποιείται παράλληλα με την L1 cache)
- 2.TLB miss: 100 cycles
- 3.L1 hit: 1 cycle
- 4.L2 hit: 15 cycles
- 5.Main memory access: 250 cycles

Ενώ, ο συνολικός αριθμός των κύκλων υπολογίζεται τελικά ως εξής:

$$\text{Cycles} = \text{Inst} + \text{TLB_Misses} * \text{TLB_miss_cycles} + \text{L1_Accesses} * \text{L1_hit_cycles} + \\ \text{L2_Accesses} * \text{L2_hit_cycles} + \text{Mem_Accesses} * \text{Mem_acc_cycles}$$

6.Γενικές Παρατηρήσεις

Στα πρώτα δύο ερωτήματα μεταβάλλουμε τις παραμέτρους των L1-cache και L2-cache.

Περιμένουμε να δούμε τα παρακάτω σύμφωνα με τη θεωρία που ήδη γνωρίζουμε:

- Αύξηση της χωρητικότητας της cache επιφέρει μείωση των capacity misses και συνεπώς αύξηση της απόδοσης
- Αύξηση του associativity της cache επιφέρει μείωση των conflict misses με αποτέλεσμα επίσης αύξηση της απόδοσης
- Αύξηση του μεγέθους των μπλοκ της cache επιφέρει μείωση των compulsory misses με αμφισβητούμενο αποτέλεσμα ανά εφαρμογή

7.Γενικά για Α Μέρος

Σε αυτό το κομμάτι του εργαστηρίου θα τρέξουμε τις προσομοιώσεις αγνοώντας ότι υπάρχουν ποινές όταν αυξάνουμε το μέγεθος του size και του associativity. Οι προσομοιώσεις χρειάστηκαν αρκετή ώρα για να τρέξουν λόγω του μεγάλου υπολογιστικού φορτίου, και εκτελεστηκαν μαζικά μεσω των script που παρατίθενται στην ενότητα με τα παραρτήματα. Στην συνέχεια εγιναν τα διαγράμματα όπως ακριβώς ήταν και στο προτεινόμενο σκριπτακι από το cslab.

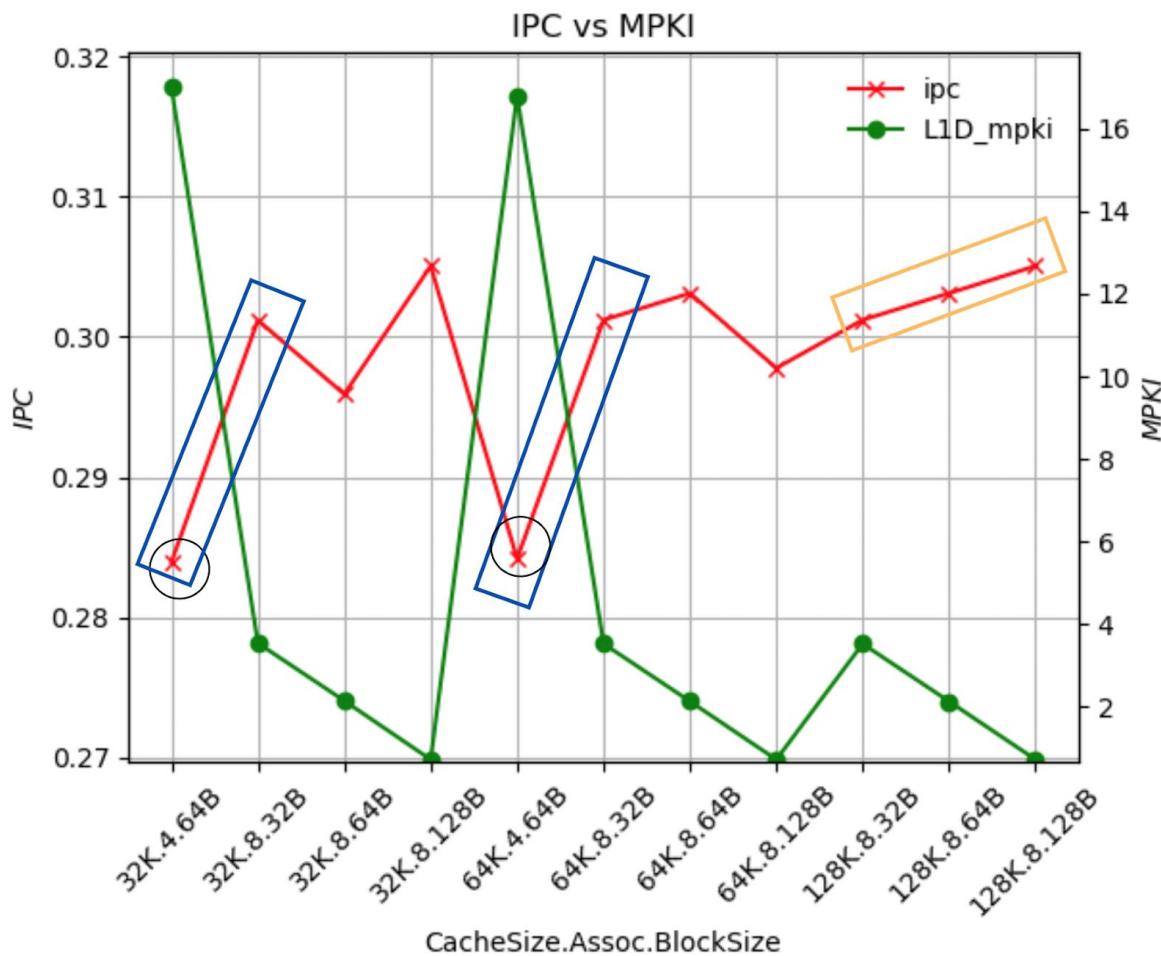
8. Έλεγχος Παραμέτρων L1-cache

Στο ερώτημα αυτό διατηρούμε τις παραμέτρους της L2-cache σταθερές και μεταβάλλουμε τις παραμέτρους της L1-cache. Οι τιμές των παραμέτρων της L2-cache και του TLB γι αυτό το ερώτημα είναι σταθερές και ίσες με:

- L2 size = 1024 KB
- L2 associativity = 8
- L2 block size = 128 B
- TLB size = 64 entr.
- TLB associativity = 4
- TLB page size = 4096 B

Παρακάτω προκύπτουν συγκριτικά τα αποτελέσματα των 10 benchmarks για διάφορους συνδυασμούς των μεγεθών της L1-cache:

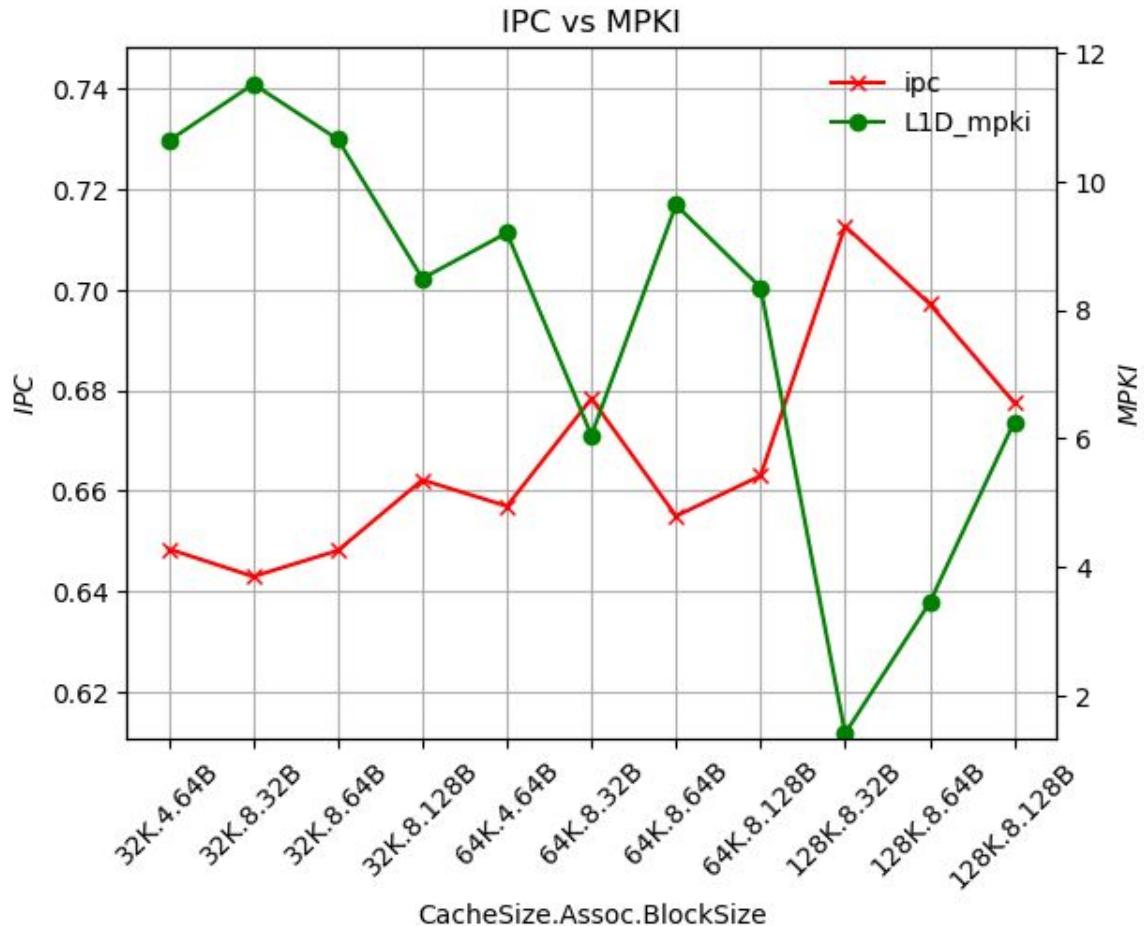
Blackscholes



Παρατηρούμε τα εξής:

- Όσον αφορά το Cache Size: Παρατηρώντας την επισήμανση στο διαγραμμα με μαύρο χρώμα βλέπουμε ότι για σταθερές τις μετικές του assoc και το block Size έχουμε σταθερό ipc ανεξαρτήτως της αύξησης του cacheSize, αυτό εύκολα παρατηρείται ότι ισχύει σε όλο το διάγραμμα. Επομένως η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Παρατηρώντας την επισήμανση στο διάγραμμα με μπλε χρώμα βλέπουμε ότι όταν αυξάνεται το assoc αυξάνεται και το ipc αυτό εύκολα παρατηρείται ότι ισχύει σε όλο το διάγραμμα. Παρατηρούμε λοιπόν πως σε αυτήν την περίπτωση η αύξηση του associativity από 4-way σε 8-way (μείωση των conflict misses) έχει δραστικά αποτελέσματα στα MPKI της L1, αυξάνοντας σε μεγάλο βαθμό τον δείκτη IPC. Επομένως η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το assoc**.
- Όσον αφορά το Block Size: Παρατηρώντας την επισήμανση στο διάγραμμα με πορτοκαλί χρώμα βλέπουμε ότι για δεδομένα cache Size και Assoc έχουμε μια μικρή αύξηση της απόδοσης όσο αυξάνει το block size. Επομένως η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **παρουσιάζει μικρή αύξηση όσο αυξάνει το block size**.

Bodytrack

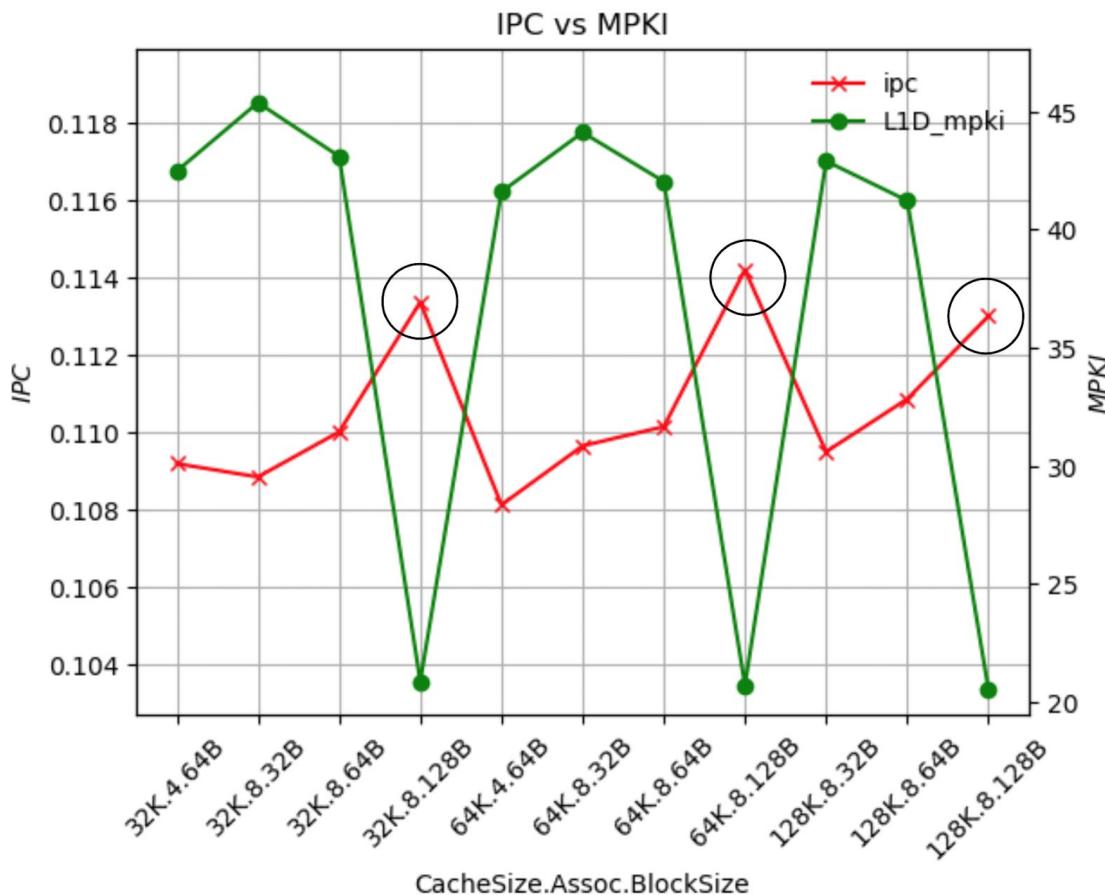


Αρχικά, στο πρώτο σημείο βλέπουμε ότι έχει χαμηλό ipc και αρκετά misses μετά αυξάνοντας το assoc μειώνοντας το block size βλέπουμε ότι έχει αρνητικό αντίκτυπο και στο ipc και στα misses, αλλά από εκεί τόσο στο τρίτο όσο και στο τέταρτο σημείο που αυξάνουμε το block Size βλέπουμε απότομη μείωση των misses και βελτίωση του ipc. Μετά στα σημεία 5,6,7,8 βλέπουμε ότι αυξάνοντας το μέγεθος της cache για να επιτύχουμε καλύτερο αποτέλεσμα πρέπει να μειώσουμε το μέγεθος του block size και επίσης παρατηρούμε ότι το assoc δεν επηρεάζει καθώς το πέμπτο και το έκτο σημείο έχουν την ίδια επίδοση. Μετά βλέπουμε πως αν αυξήσουμε και άλλο το cacheSize πετυχαίνουμε καλύτερο αποτέλεσμα το οποίο χειροτερεύει όσο αυξάνεται το block size.

Επομένως μπορούμε να θεωρήσουμε τα εξείς:

- Όσον αφορά το Cache Size: Όσο αυξάνεται το cache size βελτιώνεται τόσο το ipc όσο και το mpki. Επομένως η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Cache Size**.
- Όσον αφορά το Associativity: Παρατηρούμε ότι η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Για **μικρές τιμές του cache size**, δηλαδή για τις τιμές 32KB και 64KB η αύξηση του block size αυξάνει την επίδοση του συστήματος τοσο στο ipc οσο και στα misses. Ωστόσο, όταν η cache size γίνει μεγαλύτερη της τιμής 128 παρατηρούμε ότι όσο μεγαλώνει το block size τόσο χειρότερα τρέχει το σύστημα. Επομένως, το πως επιδρά το **block size σχετίζεται άμεσα από το cache size**

Canneal

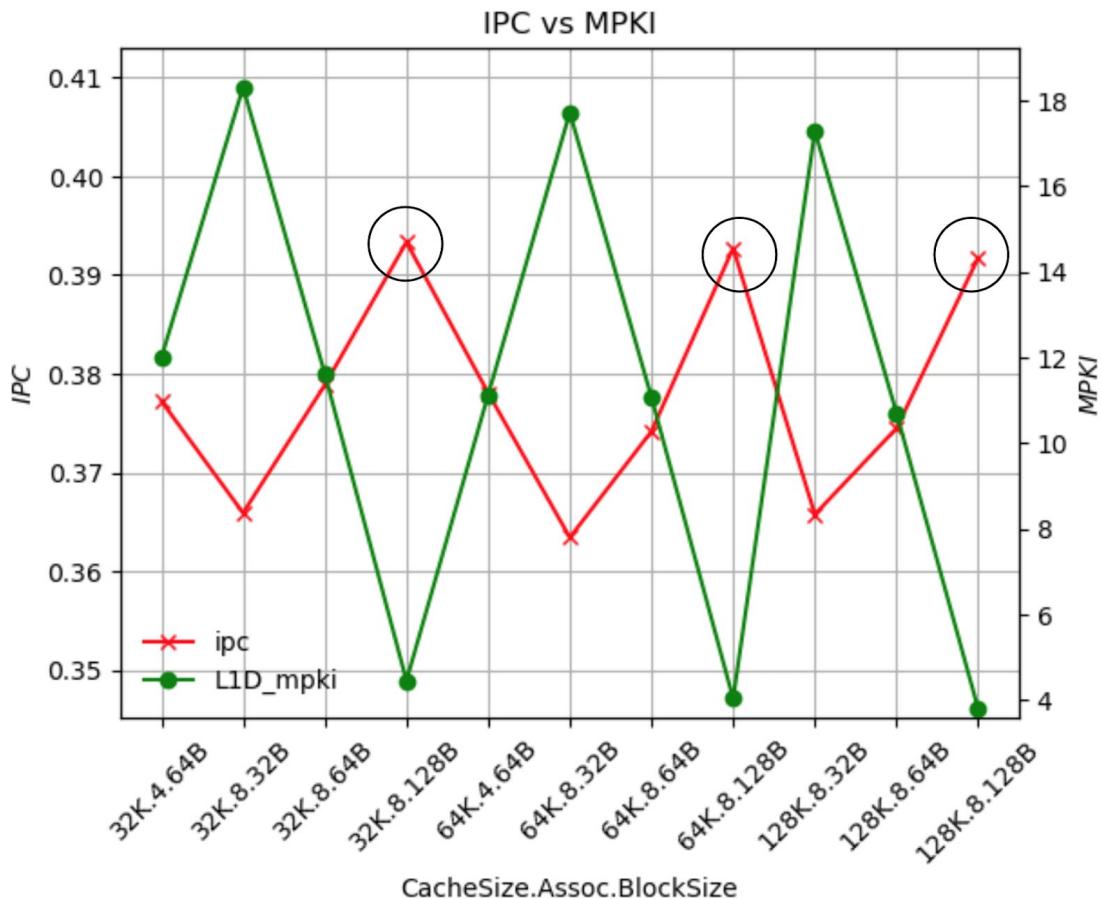


Παρατηρούμε ότι στα σημεία 1,2,3 το ipc είναι περίπου σταθερο αλλά όταν μεγαλώσουμε το block size στο 4 σημείο βλέπουμε μια απότομη βελτίωση του. Αντίστοιχα βλέπουμε να ισχύει το ίδιο για το 5,6,7 σημείο με απότομη βελτίωση της επίδοσης στο 8 και το μοτίβο αυτό ακολουθείται και στα υπόλοιπα σημεία. Αντίστοιχη συμπεριφορά παρουσιάζεται και για τα mpki προφανώς. Λόγω της απότομης αύξησης του ipc στα σημεία 4,8,11 καταλαβαίνουμε ότι η απόδοση αυξάνεται όσο αυξάνεται το block size και λόγω της σταθερά χαμηλής απόδοσης παρά της αλλαγές το cache size και το assoc καταλαβαίνουμε ότι η απόδοση δεν εξαρτάται από αυτά τα δύο.

Επομένως μπορούμε να θεωρήσουμε τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Facesim

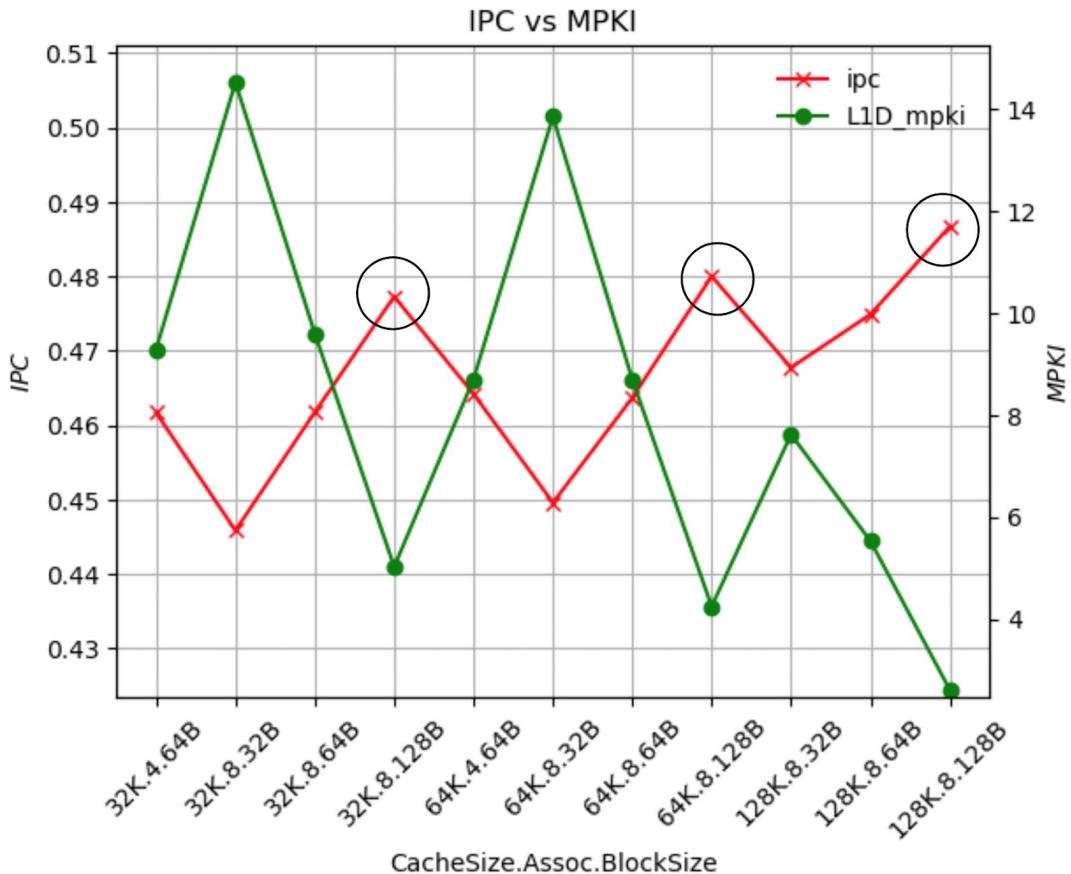


Εδώ έχουμε αντίστοιχη συμπεριφορά με το προηγούμενο benchmark, με μια βασική διαφορά. Σε αυτό το μετροπρόγραμμα η αύξηση της απόδοσης είναι σχεδόν ανάλογη με την αύξηση του block size. Πιο συγκεκριμένα, στα 32bytes έχουμε τη μικρότερη απόδοση, στα 64bytes υπάρχει σημαντική αύξηση και τέλος στα 128bytes σημειώνεται η μέγιστη απόδοση. Όπως και πριν, οι άλλες δύο παράμετροι δεν παίζουν ρόλο στην αλλαγή του IPC.

Επομένως μπορούμε να θεωρήσουμε τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Ferret

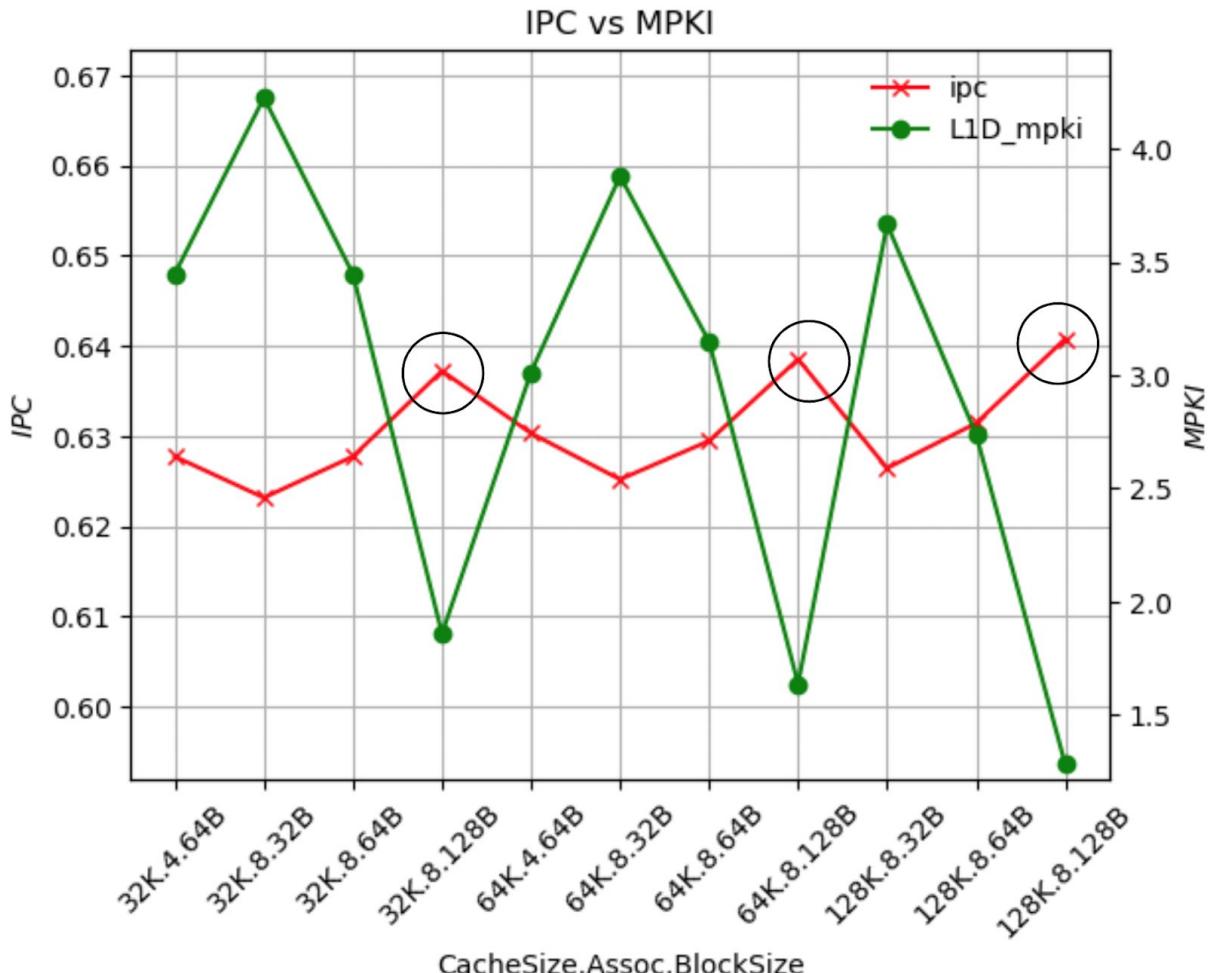


Αυτή η εφαρμογή έχει ακριβώς τα ίδια αποτελέσματα με την ακριβώς προηγούμενη εφαρμογή “Facesim”. Επομένως, για την ανάλυση της μπορείτε να ανατρέξετε στην παρουσίαση της παραπάνω εφαρμογής.

Επομένως μπορούμε να θεωρήσουμε τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Fluidanimate

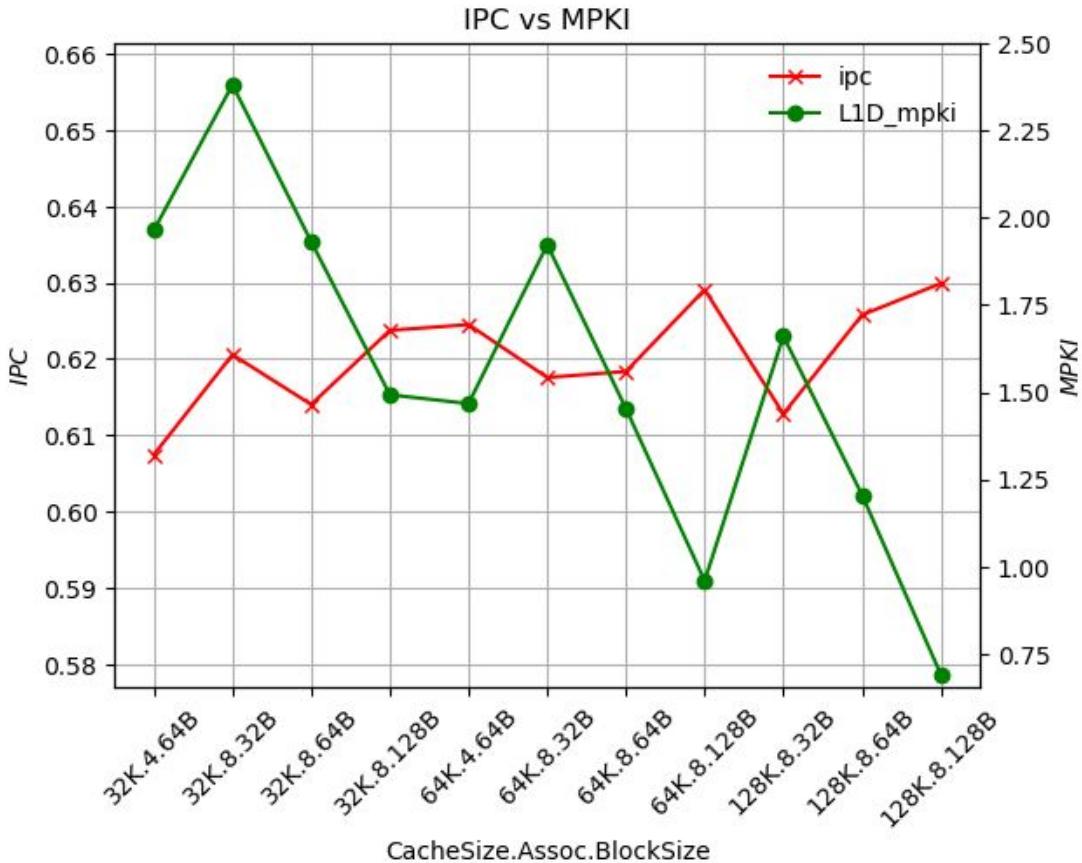


Αυτή η εφαρμογή έχει ακριβώς τα ίδια αποτελέσματα με την ακριβώς προηγούμενη εφαρμογή “Facesim” και “Ferret”. Επομένως, για την ανάλυση της μπορείτε να ανατρέξετε στην παρουσίαση της παραπάνω εφαρμογής.

Επομένως μπορούμε να θεωρήσουμε τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Freqmine



Με γενική επισκόπιση στο διάγραμμα παρατηρούμε ότι η αύξηση τόσο του block size, όσο και του cache size έχει ως αποτέλεσμα τη μείωση των compulsory και των capacity misses και συνεπώς του δείκτη MPKI. Η associativity δεν επηρεάζει αισθητά τον δείκτη MPKI γεγονός που σημαίνει πως δεν υπάρχουν συχνά conflict misses πχ το σημείο 5 και 7 έχει ακριβώς το ίδιο MPKI.

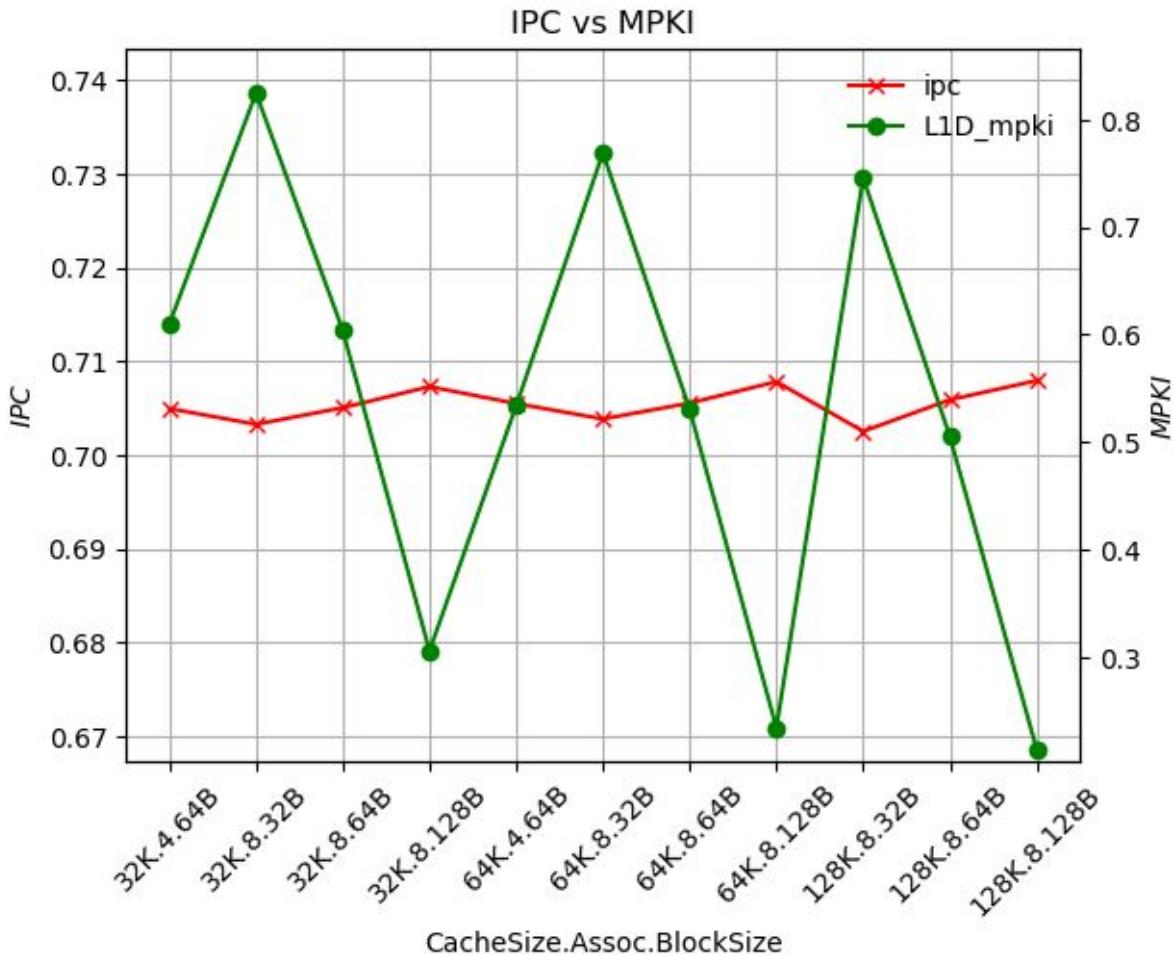
Όσον αφορά το ipc βλέπουμε ότι το assoc δεν επηρεάζει έντονα το αποτέλεσμα του ipc, πχ το σημείο 5 και 7 έχουν σχεδόν ίδια τιμή ipc. Ωστόσο βλέπουμε πως το ipc εξαρτάται τόσο από το cache size όσο και από το block size και γενικά αυξάνεται όσο αυξάνονται αυτές οι δύο παραμετροί.

Επομένως μπορούμε να θεωρήσουμε τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **επιδρά** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η μεταβολή του block size **επιδρά** στην επίδοση του προγράμματος για το συγκεκριμένο input.

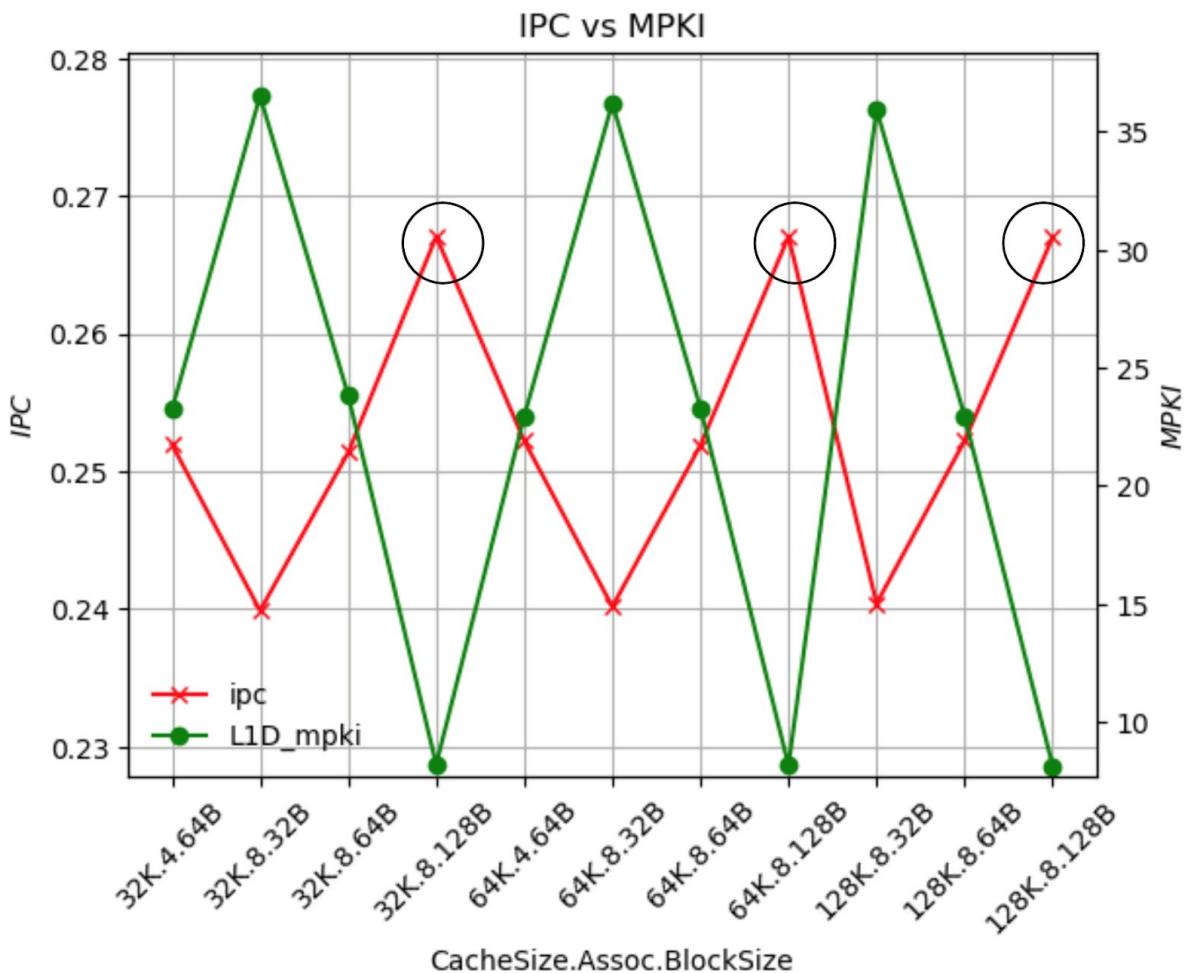
Επομένως για το δεδομένο πρόγραμμα και τη δεδομένη είσοδο θα ήταν πιο λόγο αποδοτική η χρήση μιας μεγάλης cache με μεγαλύτερο block size χωρις να μας ενδιαφέρει τόσο το assoc.

Rtview



Αυτή η εφαρμογή έχει ακριβώς τα ίδια αποτελέσματα με την ακριβώς προηγούμενη εφαρμογή “Facesim” και “Ferret” και “Fluidanimate” όσον αφορά τη διακύμανση των misses (δείκτης MPKI). Ωστόσο, επειδή τα misses ανά εντολή είναι λιγότερα από τις προαναφερθείσες εφαρμογές παρατηρούμε πως το IPC δε μεταβάλλεται σχεδόν καθόλου. Πιο αναλυτικά, σε αυτό το μετροπρόγραμμα η αύξηση των misses είναι σχεδόν ανάλογη με την αύξηση του block size. Πιο συγκεκριμένα, στα 32bytes έχουμε το μεγαλύτερο πλήθος misses στα 64bytes υπάρχει σημαντική μείωση και τέλος στα 128bytes σημειώνεται το μικρότερο πληθος misses. Οπως και πριν, οι άλλες δύο παράμετροι δεν παίζουν ρόλο στην αλλαγή του MPKI. Όσον αφορά το ipc παρατηρούμε ότι παραμένει σταθερός και ανεξαρτητως των παραμετρων. Επίσης ο ipc ειναι ανεξάρητως και από τον MPKI επειδή τα misses ανά εντολή είναι λιγότερα από τις προαναφερθείσες εφαρμογές

Streamcluster



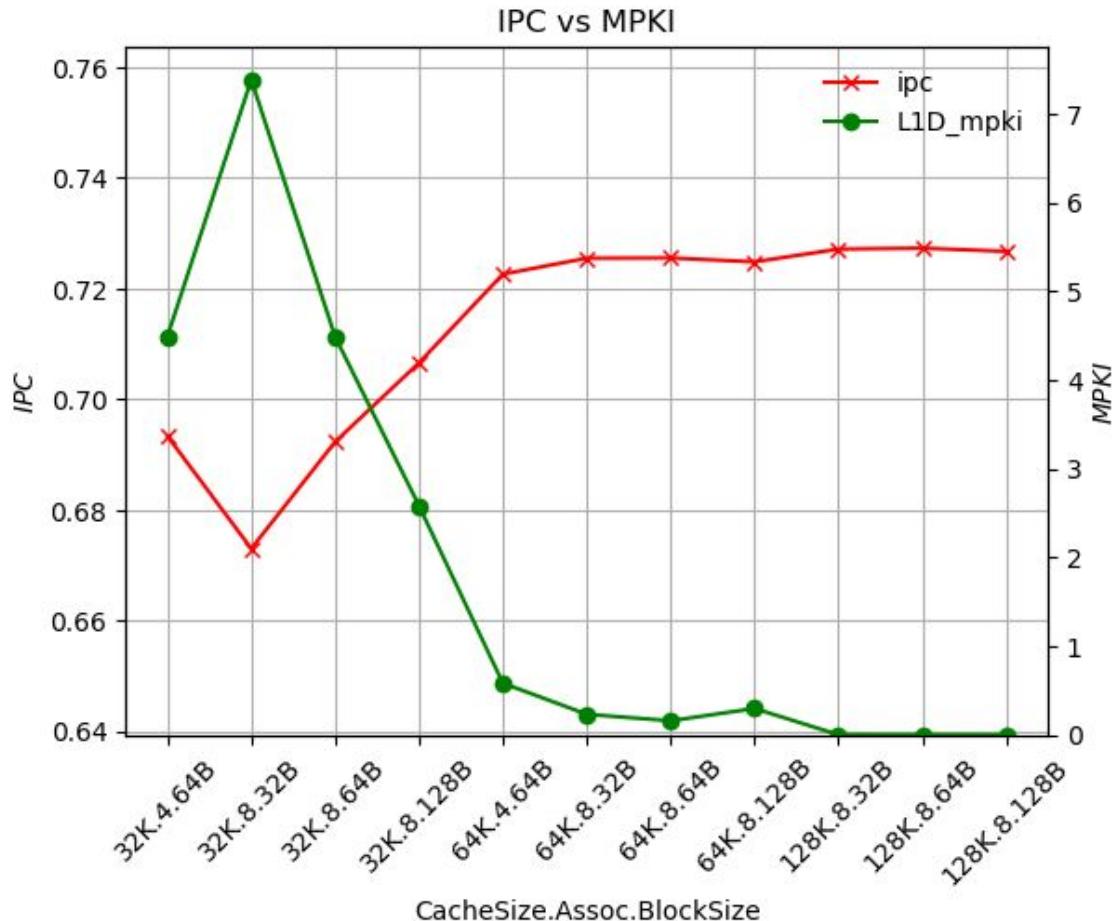
Αυτή η εφαρμογή έχει ακριβώς τα ίδια αποτελέσματα με την ακριβώς προηγούμενη εφαρμογή “Facesim” και “Ferret” και “Fluidanimate”.

Μάλιστα, σε αυτή την περίπτωση επειδή τα misses είναι περισσότερα από των παραπάνω εφαρμογών παρατηρούμε πολύ μεγαλύτερη διακύμανση στον δείκτη IPC. Παρατηρούμε πως στα “Rtview” και “Streamcluster” ενώ ο δείκτης MPKI είναι ποιοτικά ίδιος, η απόλυτη διαφορά στα misses (η μια εφαρμογή έχει 0.5 misses ανά 1000 εντολές μέσο όρο και η άλλη έχει 25 misses ανά 1000 εντολές μέσο όρο) έχει ως αποτέλεσμα διαφορετική συμπεριφορά στο IPC. Αυτό μπορεί να αιτιολογηθεί διαφορετικά πως κάποιες εφαρμογές για δεδομένα inputs έχουν εγγενώς περισσότερα misses από κάποιες άλλες, με αποτέλεσμα να επηρεάζονται οι εντολές ανά κύκλο ρολογιού.

Επομένως μπορούμε να θεωρήσουμε τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Swaptions



Για τις μετρήσεις οταν η cache size είναι ίση με 32KB και έχουμε 8way assoc παρατηρούμε ότι το ipc βελτιώνεται όσο μεγαλύτερο είναι το block size και αντίστοιχα βλέπουμε ότι το mpki μειώνεται όσο αυξάνεται το blocksize.

Επίσης όταν η cache size είναι ίση με 32KB βλέπουμε ότι η assoc δεν επηρεάζει.

Για τις μετρήσεις όταν η cache size είναι μεγαλύτερη των 32KB παρατηρούμε ότι ο δείκτης mpki εκμηδενίζεται και συνεπώς το ipc φτάνει στη μέγιστη τιμή του και είναι ανεξάρτητο του associativity και το block size.

Επομένως για αυτή την εφαρμογή με αυτά τα δεδομένα inputs η καλυτερη στρατηγική θα ήταν να χρησιμοποιήσουμε μια L1 μεγέθους 64KB μη λαμβάνοντας υπόψη τους άλλες δύο παραμέτρους.

Γενικά θα μπορούσαμε να πούμε τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **επιρεάζει θετικά** το ipc με threshold την τιμή των 64KB μετά από την οποία **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size** με threshold την τιμή του cache size των 64KB μετά από την οποία **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.

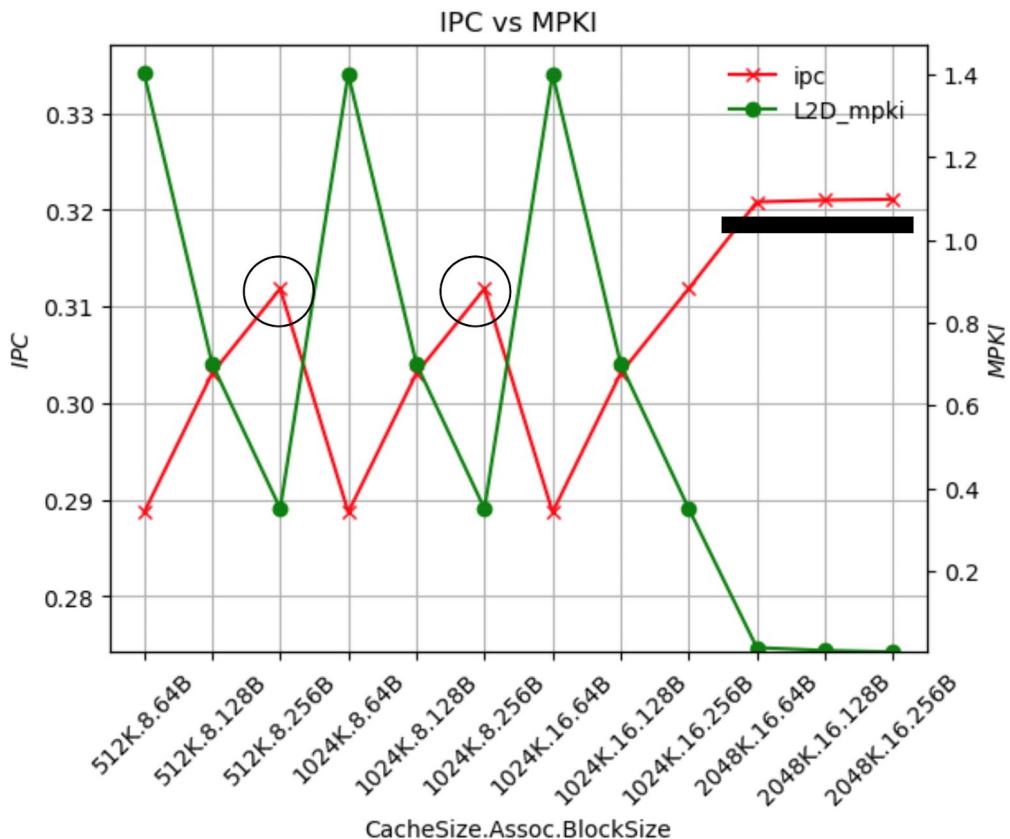
9. Έλεγχος Παραμέτρων L2-cache

Στο ερώτημα αυτό διατηρούμε τις παραμέτρους της L1-cache σταθερές και μεταβάλλουμε τις παραμέτρους της L2-cache. Οι τιμές των παραμέτρων της L1-cache και του TLB γι αυτό το ερώτημα είναι σταθερές και ίσες με:

- L1 size = 32 KB
- L1 associativity = 8
- L1 block size = 64 B
- TLB size = 64 entr.
- TLB associativity = 4
- TLB page size = 4096 B

Παρακάτω προκύπτουν συγκριτικά τα αποτελέσματα των 10 benchmarks για διάφορους συνδυασμούς των μεγεθών της L2-cache:

Blackscholes



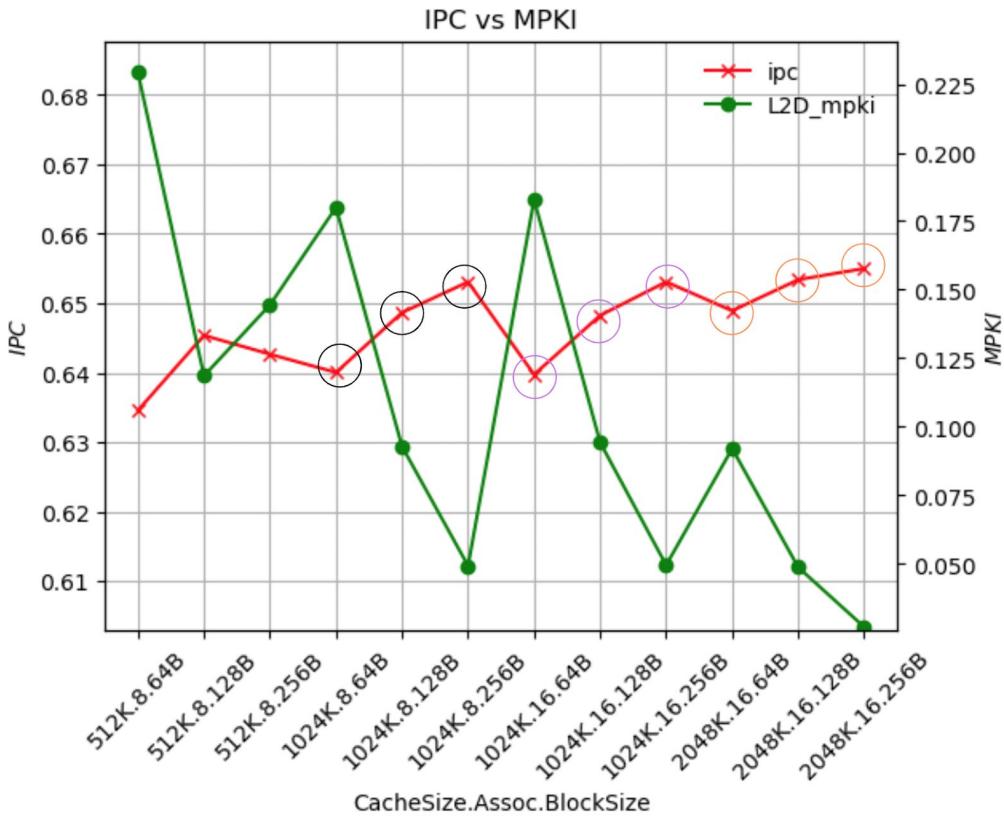
Για τιμές cache size μικρότερες των 2048KB έχουμε ότι η μετρική ipc εξαρτάται από το Block Size και είναι ανεξάρτητη από τους άλλους δύο παράγοντες. Μάλιστα βλέπουμε ότι υπάρχει σχεδόν γραμμική αυξηση του ipc όσο αυξάνει το block size με το ipc να έχει την μικρότερη τιμή του όταν το block size είναι ίσο με 64, να βελτιώνεται όταν έχει την τιμή 128 και να γίνεται βέλιτστος για τιμή ίση με 256. Επίσης αυτή η “περιοδικότητα” των πρώτων 9 σημείων δείχνει φανερά ότι οι παράγοντες του cache size και το assoc δεν επηρεάζουν. Επίσης ο δείκτης mpki παίρνει την μικρότερη τιμή του για block size ίσο με 256 λίγο μεγαλύτερη για 128 και την μικρότερη για 64, ακολουθόντας και αυτος μια σχεδόν γραμμική σχέση.

Για τις μετρήσεις με cache size ίση των 2048KB βλέπουμε ότι το ipc είναι σταθερό και ανεξάρτητο από τις άλλες δύο μετρικές. Για αυτές τις τιμές έχουμε πολύ μικρό και σταθερό mpki

Άρα γενικά βλέπουμε:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επηρεάζει αισθητά** την επίδοση του προγράμματος μέχρι το threshold του cache size=2048 όπου επηρεάζει καθοριστικά και δίνει το βέλτιστο ipc.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size** με threshold την τιμή του cache size των 2048KB μετά από την οποία **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.

Bodytrack



Από τις ομάδες που έχουν επισημανθεί παρατηρούμε ότι όσο αυξάνει το block size τόσο αυξάνει και το ipc. Επίσης από την συγκριση των ομάδων μεταξύ τους παρατηρούμε ότι οι δεξιότερες ομάδες έχουν μεγαλύτερη απόδοση, δηλαδή όσο αυξάνει το cache size αυξάνει και η επίδοση του ipc. Ωστόσο συγκρίνοντας τις δύο δεξιότερες ομάδες μπορούμε να κατανοήσουμε ότι το assoc επηρεάζει λίγο το σύστημα.

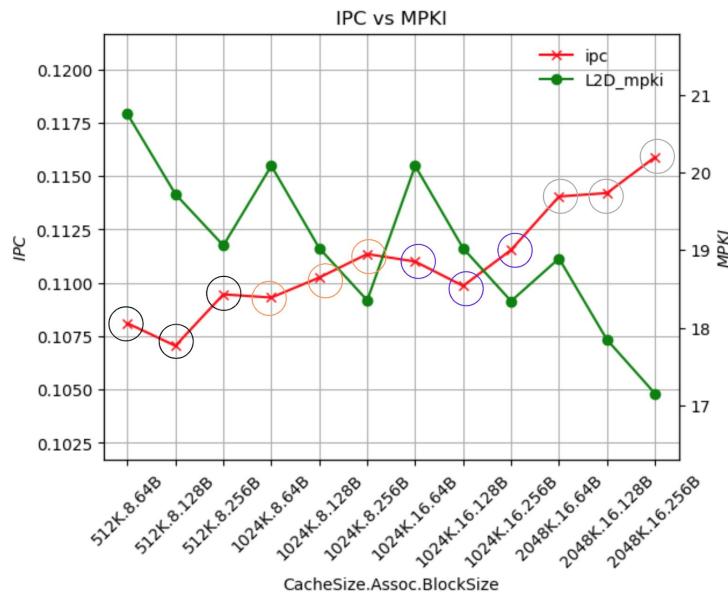
Όμοια βλέπουμε πως και η μετρική του mpki γίνεται μέγιστη για μικρά block size και μικράνει όσο μεγαλώνουν τα block size, επίσης μικράνει όταν μεγαλώνει το cache size και αυτό γίνεται πιο κατανοητό παρατηρώντας τις δυο δεξιότερες κορυφέρ του mpki.

Άρα γι αυτό το πρόγραμμα με αυτά τα δεδομένα για τα υπόλοιπα στοιχεία του και για δεδομένη είσοδο βλέπουμε ότι η καλύτερη επιλογή είναι μια μεγάλη cache με μεγάλο block size.

Άρα γενικά βλέπουμε:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **επιδρά θετικά** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **επιδρά αρνητικά** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size** για τιμές cache size μεγαλύτερες του 1024KB. Για το cache size=512KB βλέπουμε ότι η πρώτη αυξηση του βιηθά και μετά η δεύτερη δρα αρνητικά.

Canneal



Σχόλιο: Στην ανάλυση παρακάτω η πρώτης(αριστεροτερη) ομαδοποίηση αναφέρεται ως πρώτη τριάδα, η δεύτερη ομαδοποίηση ως δεύτερη κλπ

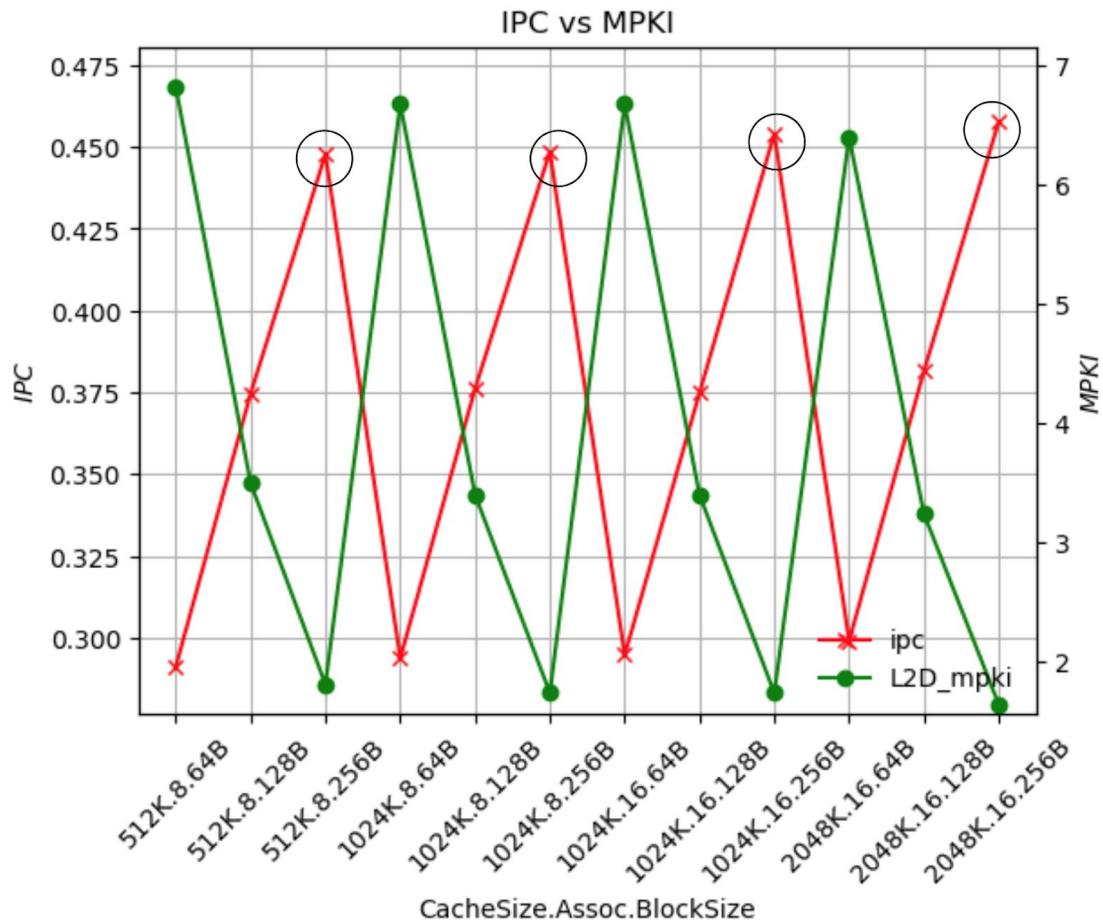
Παρατηρούμε ότι ο ipc εξαρτάται κυρίως από τον παράγοντα του cache size, και του block size. Πιο αναλυτικά βλέπουμε ότι μεταξύ της πρώτης και της δευτερης τριάδας και μεταξύ της τρίτης και της τέταρτης τριάδας παρατηρούμε αύξηση του ipc και μάλιστα αυτή η αύξηση γίνεται ιδιαίτερα έντονη όταν γίνεται 2048KB. Αυτές οι τριάδες μεταξύ τους έχουν διαφορετικό cache size. Επίσης, σε κάθε μία από τις τριάδες βλέπουμε μια αυξητική πορεία του ipc όσο αυξάνει το block size της τριάδας. Επίσης, συγκρίνοντας τη δεύτερη με την τρίτη τριάδα δεν βλέπουμε ουσιαστικές διαφοροποιήσεις επομένως εξάγεται το συμπέρασμα ότι το assoc δεν επηρεάζει το ipc, αφού η δεύτερη και η τρίτη τριάδα διαφοροποιούνται μόνο ως πρός το assoc.

Μελετώντας την άλλη μετρική βλέπουμε ότι παρουσιάζει μια περιοδικότητα ανα τριάδα καταγραφών και όσο μικρότερο είναι το block size τότο μεγαλύτερος είναι ο δείκτης mpki, δηλαδή για 64B έχουμε τον μεγαλύτερο mpki για την εκάστοτε τριάδα για 128 ακόμη μικρότερο και για 256 το μικρότερο της εκάστοτε τριάδας. Επίσης, βλέπουμε πτώση μεταξύ της πρώτης και της δεύτερη τριάδας, η δευτερη και η τρίτη τριάδα είναι σχεδόν ίδια και η τέταρτη παρουσιάζει πτώση σε σχέση με την δευτερη. Αυτό συμβαίνει επειδή μεταξύ πρώτης δεύτερης και τέταρτης έχουμε αύξηση του cache size και μεταξύ της δευτερης και της τρίτης έχουμε ίδιο assoc. Άρα συνολικά ο mpki εξαρτάται από το cache size και το block size (βελτιώνεται όσο οι παράγοντες αυτοί μεγαλώνουν) κια είναι σχεδόν ανεξάρτητος του assoc.

Άρα γενικά βλέπουμε:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **επιδρά θετικά** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **επιδρά θετικά** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input αλλάζει διαρκώς. Για μία τιμή μεγαλύτερη, δηλαδή για 128, δρά αρνητικά και για μία ακόμη, δηλαδή για 256, δρά θετικά.

Facesim



Παρατηρούμε ότι μέσα σε κάθε ομάδα, όπου δηλαδή αυξάνεται το block size έχουμε βελτίωση της απόδοσης.

Συγκρίνοντας τις ομάδες 1-3-4 βλέπουμε ότι η αυξηση του cache size έχει μια μικρή ανεπαισθητη βελιώση στο ipc, αλλά επί της ουσίας δεν έχουμε αξιόλογη βελτίωση.

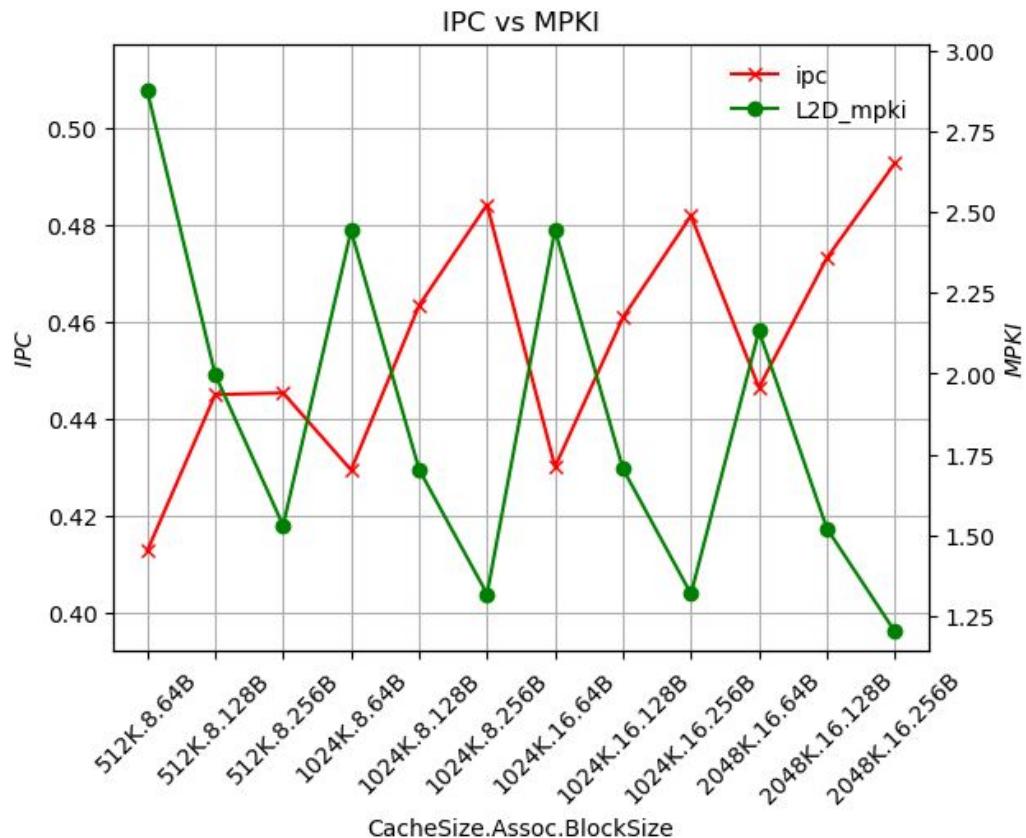
Συγκρίνοντας τις ομάδες 2-3 βλέπουμε ότι η αυξηση του assoc έχει μια μικρή ανεπαισθητη βελιώση στο ipc αλλά επί της ουσίας δεν έχουμε αξιόλογη βελτίωση.

Σχετικά με το mpki παρατηρούμε ότι έχει μια ελάχιστη βελτίωση με την αυξηση του cache size και το assoc, σχεδόν ανεπαισθητη, αλλά καλυτερεύει πολύ με την αύξηση του block size.

Άρα γενικά βλέπουμε:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Ferret



Σχόλιο: πιθανότατα η τρίτη τιμή να έχει υποστεί θορυβωση.

Βλέπουμε ότι κατά την 1-2-4 τριάδα υπάρχει μία σταδιακή αύξηση του ipc άρα υπάρχει μια εξάρτηση από το cache size.

Κατά την 2-3 τριάδα δεν βλέπουμε ουσιαστικές διαφοροποιήσεις και γι αυτό εξάγουμε το συμπέρασμα ότι το assoc δεν επηρεάζει το αποτέλεσμα ή το επηρεάζει στα όρια του επιστητου.

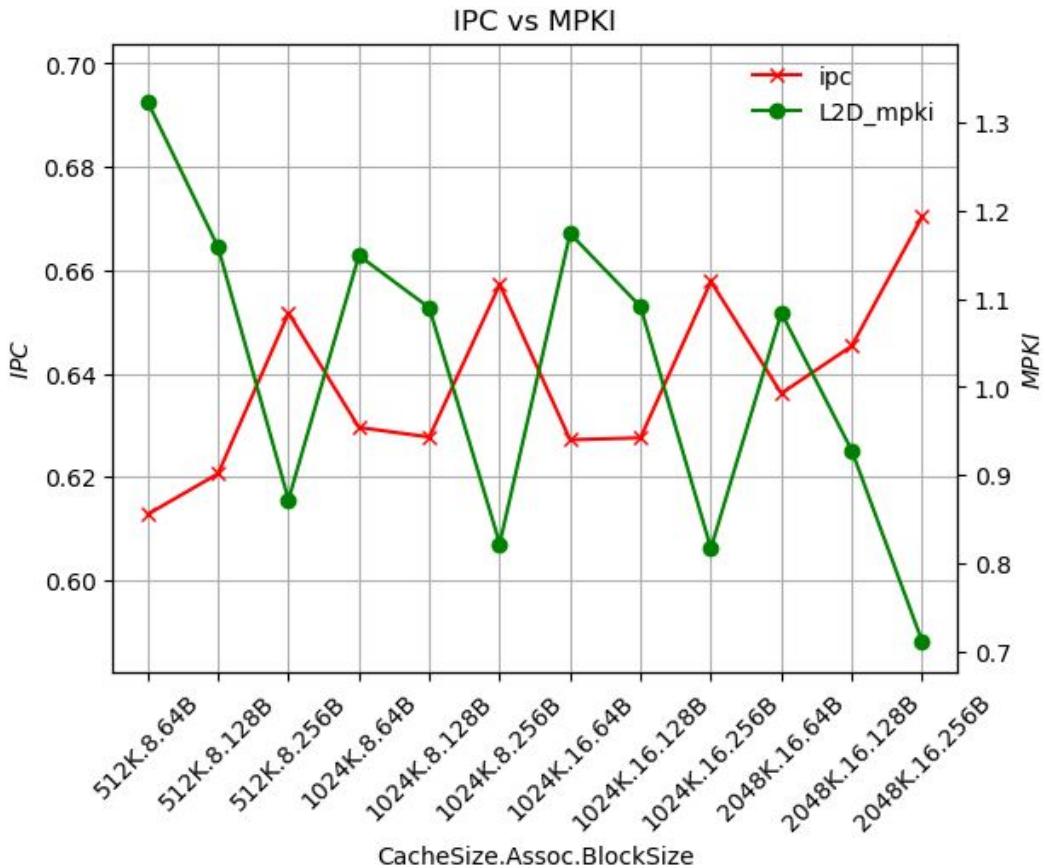
Τέλος μεσα σε κάθε τριάδα βλέπουμε μια έντονη αύξηση του ipc όσο αυξάνει το το block size. Επομένως το αποτέλεσμα εξαρτάται εντονος από το block size.

Σχετικά με τον δείκτη mpki παρατηρούμε ότι ισχύουν ακριβώς τα αντίστοιχα με την προηγούμενη περίπτωση μόνο που τώρα επηρεάζει το αποτέλεσμα και το μέγεθος της cache size.

Άρα γενικά βλέπουμε:

- Όσον αφορα το Cache Size: Η αύξηση του cache size **επηρεάζει θετικά** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**. Εξαίρεση αποτελεί για το cache size 512KB η αύξηση από 128B σε 256B.

Fluidanimate



Βλέπουμε ότι κατα την 1-2-4 τριάδα υπάρχει μία σταδιακή αύξηση του ipc άρα υπάρχει μια μικρή εξάρτηση από το cache size.

Κατα την 2-3 τριάδα δεν βλέπουμε ουσιαστικές διαφοροποιήσεις και γι αυτό εξάγουμε το συμπέρασμα ότι το assoc δεν επηρεάζει το αποτέλεσμα ή το επηρεάζει στα όρια του επιστητου.

Τέλος μεσα σε κάθε τριάδα βλέπουμε μια έντονη αύξηση του ipc όσο αυξάνει το το block size. Επομένως το αποτέλεσμα εξαρτάται εντονος από το block size.

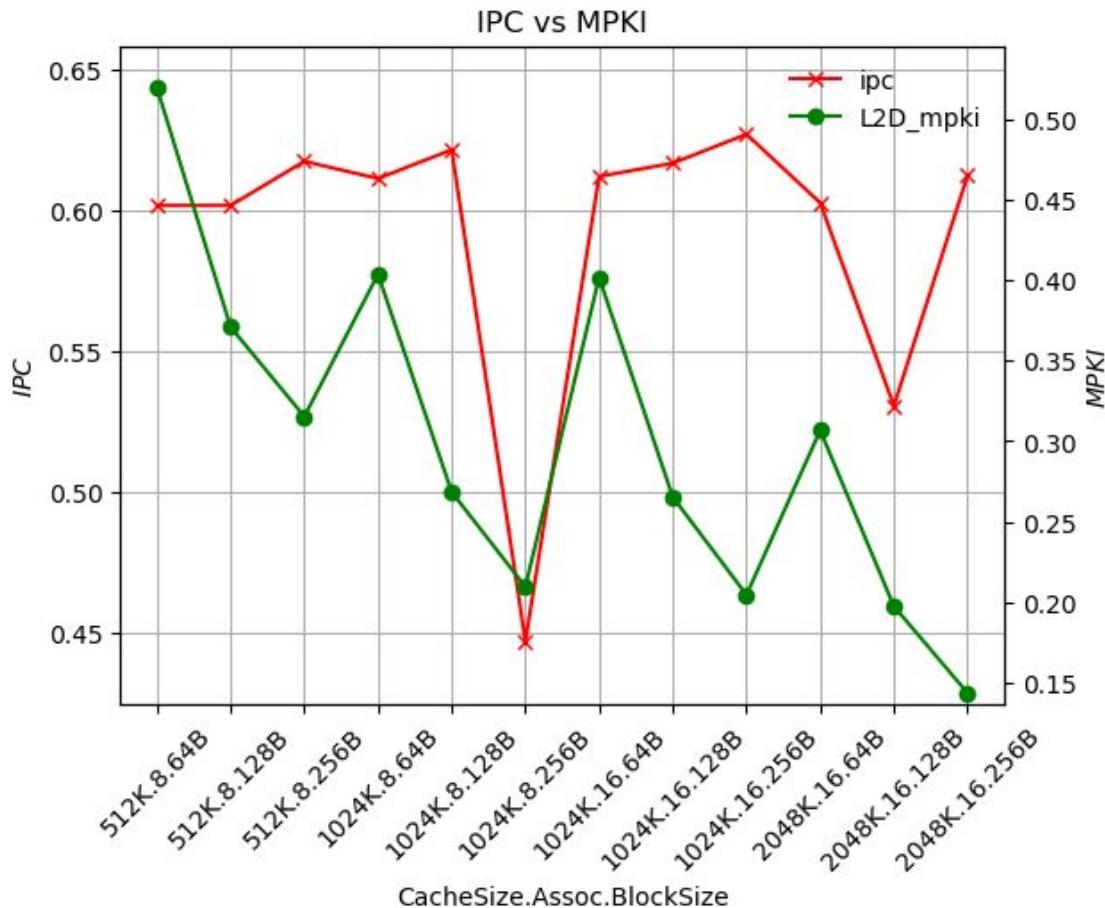
Σχετικά με τον δείκτη mpki παρατηρούμε ότι ισχύουν ακριβώς τα αντίστοιχα με την προηγούμενη περίπτωση μόνο που τώρα επηρεάζει λίγο το αποτέλεσμα και το μέγεθος της cache size.

Επίσης, τα MPKI είναι λιγότερα κατ' απόλυτη τιμή με συνέπεια τα IPC να μην αλλάζει σε τόσο μεγάλο βαθμό όσο των προηγουμένων μετροπρογραμμάτων.

Άρα γενικά βλέπουμε:

- Όσον αφορα το Cache Size: Η αύξηση του cache size **επηρεάζει θετικά** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**. Εξαίρεση αποτελεί για το cache size 512KB η αύξηση από 128B σε 256B.

Freqmine



Θα σχολιάσουμε για αρχή το mpki:

Παρατηρούμε ότι σε κάθε τριάδα όσο αυξάνεται το block size τόσο μειώνεται το mpki. Άρα το mpki καλυτερεύει όταν αυξάνεται το block size. Από την άλλη βλέπουμε ότι συγκριτικά οι τιράδες 1-2-4 δείχνουν ότι όσο αυξάνεται το cache size τόσο βελτιώνεται το mpki. Τέλος η τριάδες 2-3 μοιάζουν πολύ επομένως το assoc δεν επηρεάζει το mpki.

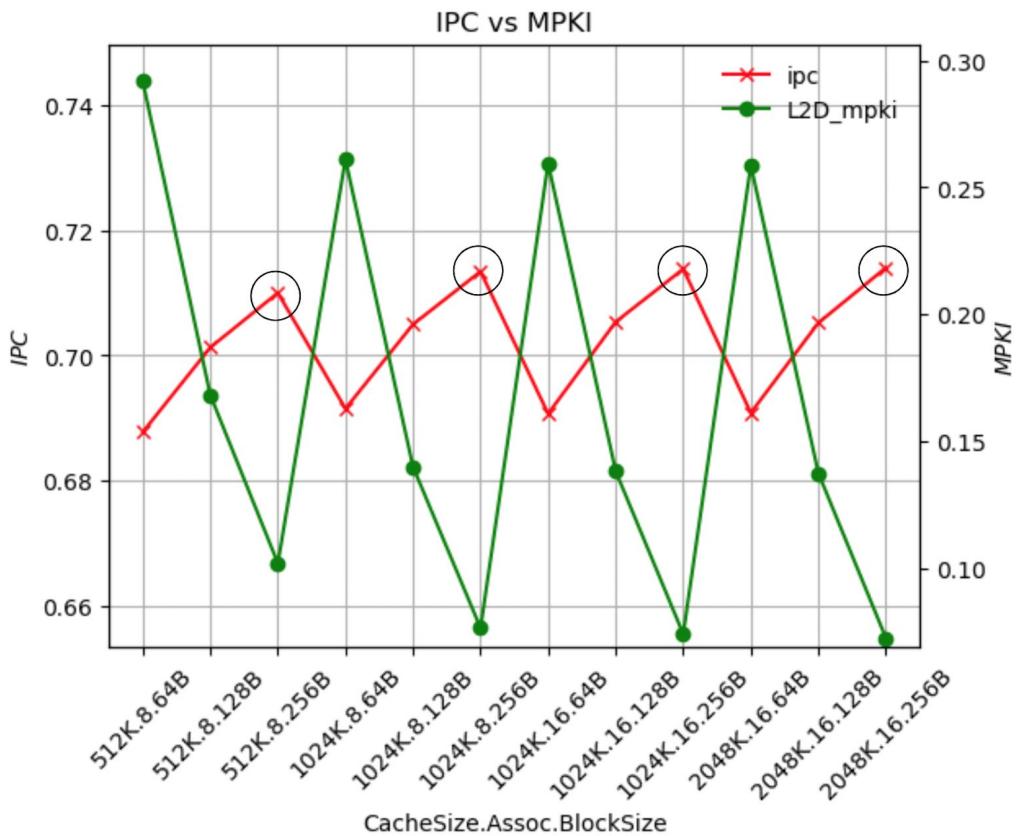
Στα προηγούμενα διαγράμματα έχουμε παρατήρηση ότι η μειωσεί το mpki αντιστοιχεί σε αύξηση του ipc και αντίστροφα κάτι που δεν παρατηρείται εδώ.

Μάλιστα οι απότομες πτώσεις του ipc στην 6 και στην 10 καταγραφεί σε πρώτο επίπεδο φαίνονται αδικαιολόγητες, αλλά η αιτία ύπαρξής τους είναι η κακή συμβατότητα των τιμών της L1 με αυτές τις τιμές. Επομένως η χειρότερη επιλογή θα ήταν να τοποθετησουμε L2 με αυτές τις τιμές.

Άρα γενικά βλέπουμε:

- Όσον αφορά το Cache Size: Η αύξηση του cache size **επηρεάζει θετικά** για την μεταβολή από 512 σε 1024 και **αρνητικά** για την μεταβολή 1024 σε 2048 την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**. Εξαίρεση αποτελούν οι καταγραφές 1024K.8.256 και 2048K.16.128B.

rtview



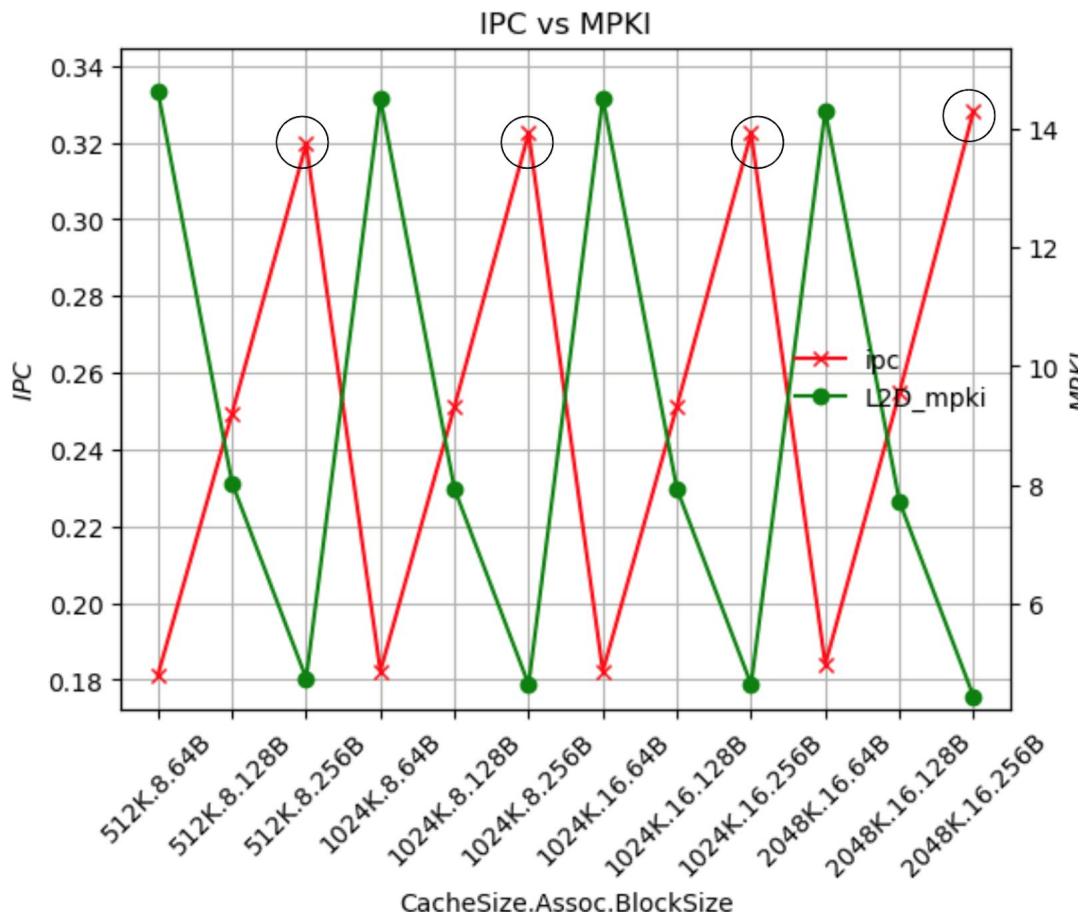
Παρατηρούμε ότι μεταξύ των τριάδων υπάρχει μια περιοδικότητα στον τρόπο που αλλάζει η μετρική ipc και αυτό σημαίνει ότι το αποτέλεσμα δεν εξαρτάται ούτε από το cache size ούτε από το assoc (η σχέση μεταξύ 1-3-4 τριάδας φανερώνει το πως επηρεάζει το ipc η cache size και η σχέση μεταξύ 2-3 το assoc)
Μέσα σε κάθε τριάδα, όπου μεταβάλλεται ο παράγοντας του block size παρατηρούμε ότι όσο μεγαλώνει το block size τόσο καλυτερεύει το ipc.

Σχετικά με το mpki παρατηρούμε ότι ουτε αυτο εξαρτάται από τα μεγέθη cache size και assoc και ότι όσο μεγαλώνει το block size τόσο καλυτερεύει δηλαδή μειώνεται.

Άρα γενικά βλέπουμε:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Streamcluster



Παρατηρούμε ότι μέσα σε κάθε ομάδα, όπου δηλαδή αυξάνεται το block size έχουμε βελτίωση της απόδοσης.

Συγκρίνοντας τις ομάδες 1-3-4 βλέπουμε ότι η αυξηση του cache size έχει μια μικρή ανεπαισθητη βελτίωση στο ipc, αλλά επί της ουσίας δεν έχουμε αξιόλογη βελτίωση.

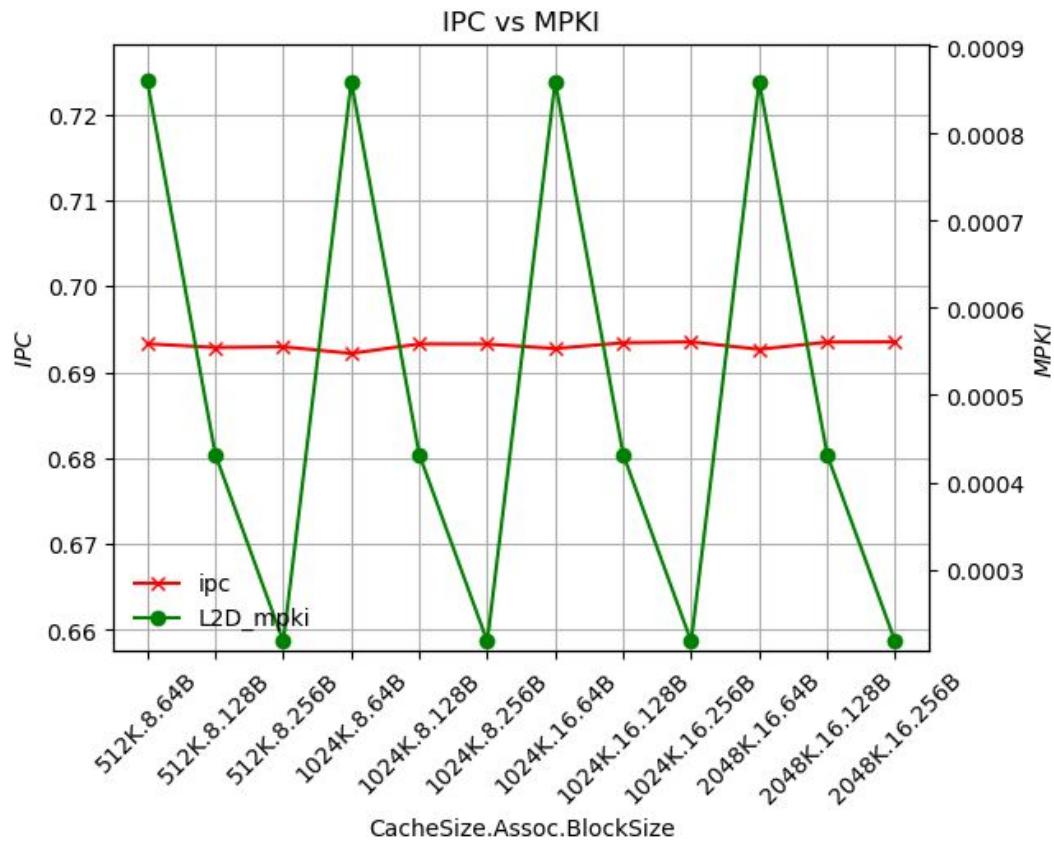
Συγκρίνοντας τις ομάδες 2-3 βλέπουμε ότι η αυξηση του assoc έχει μια μικρή ανεπαισθητη βελτίωση στο ipc αλλά επί της ουσίας δεν έχουμε αξιόλογη βελτίωση.

Σχετικά με το mpki παρατηρούμε ότι έχει μια ελάχιστη βελτίωση με την αυξηση του cache size και το assoc, σχεδόν ανεπαίσθητη, αλλά καλυτερεύει πολύ με την αύξηση του block size.

Άρα γενικά βλέπουμε:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Swaptions



Παρατηρούμε ότι υπάρχουν ελάχιστα misses, max τιμή του είναι 0.0009 misses για 1000 instructions, γεγονός που δεν επηρεάζει την απόδοση του προγράμματος όπως δείχνει ο δείκτης IPC της παραπάνω γραφικής.

Γι αυτό παρατηρούμε ότι ο δείκτης ipc παραμένει σταθερός και ανεξάρτητος από τους παραγοντες cache size, assoc, block size.

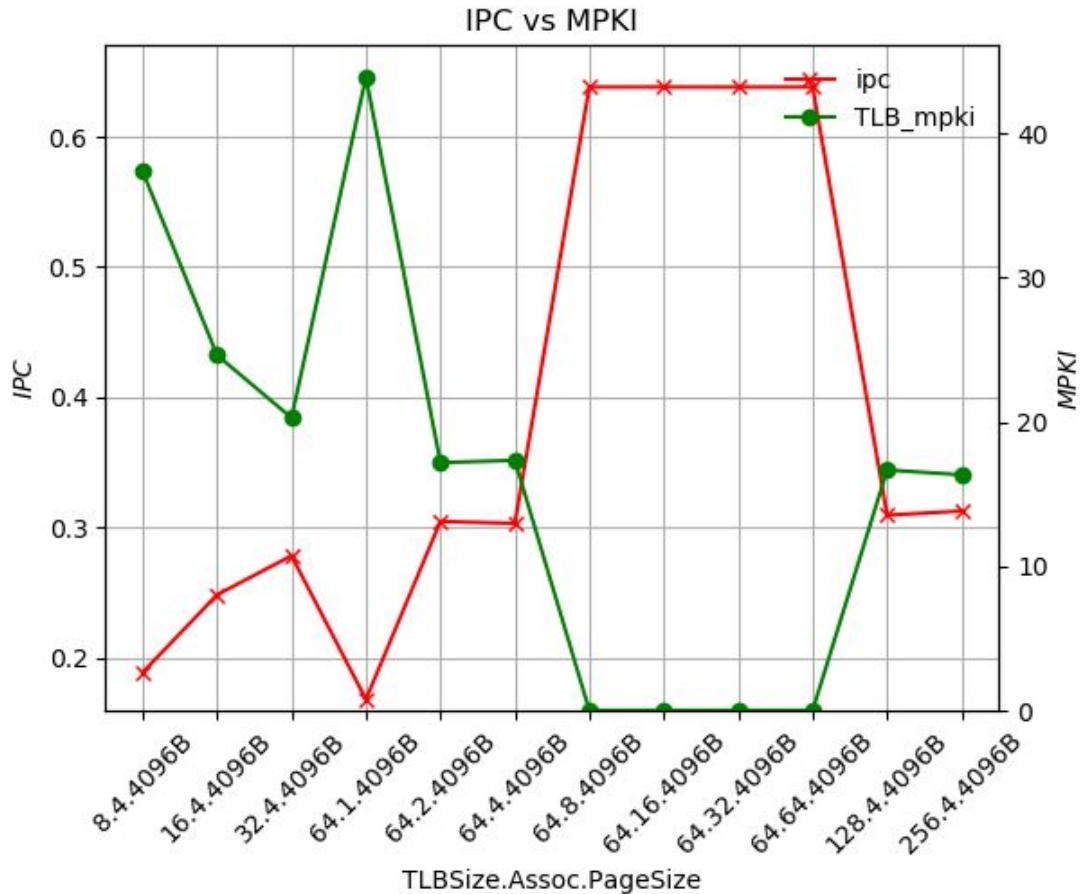
10. Έλεγχος Παραμέτρων TLB

Στο ερώτημα αυτό διατηρούμε τις παραμέτρους των L1-cache και L2-cache σταθερές. Επίσης διατηρείται σταθερό και το μέγεθος σελίδας του TLB και ίσο με 4096 Bytes. Οι τιμές των L1-cache και L2-cache για αυτή τη μελέτη είναι σταθερές και ίσες με:

- L1 size = 32 KB
- L1 block size = 64 B
- L1 associativity = 8
- L2 size = 1024 KB
- L2 associativity = 8
- L2 block size = 128 B

Παρακάτω προκύπτουν συγκριτικά τα αποτελέσματα των 10 benchmarks για διάφορους συνδυασμούς των μεγεθών της TLB

Blackscholes



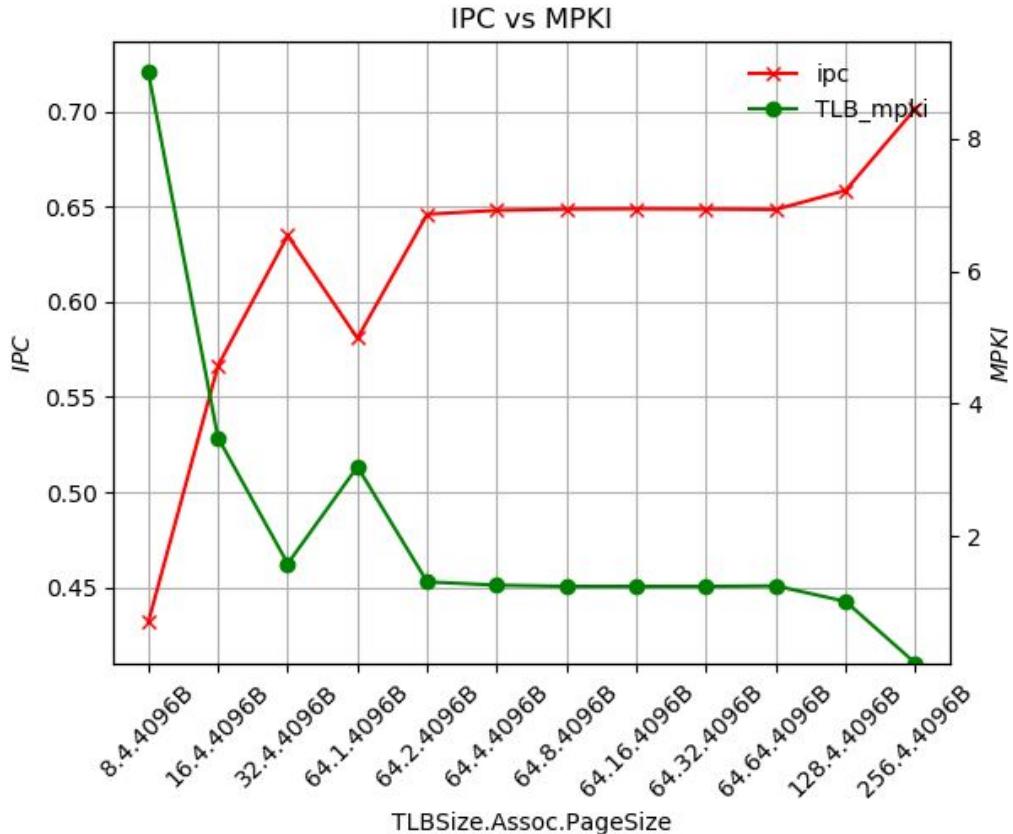
Παρατηρούμε ότι υπάρχει εξάρτηση τόσο από το TLBSize όσο και από το Assoc. Αναλυτικά το tlb size βελτιώνει την απόδοση, αλλά μέχρι ενός σημείου, το οποίο σε αυτή την περίπτωση είναι τα 32 entries. Για την αλλαγή από 32 σε 64 κ.ο.κ. δεν υπάρχει βελτίωση. Επίσης το assoc παρουσιάζει τρία κομβικά σημεία: την τιμή 1 όπου δίνει πολύ κακή τιμή στο ipc, το κατώφλι 4, όπου μεταξύ 1-4 δίνει σταθερά μέτρια τιμή στο ipc και για πάνω από τέσσερα δίνει την καλύτερη τιμή στο ipc αλλα για οποιαδήποτε άλλη αύξηση δεν παρατηρείται καμία βετίωση του ipc. Αξίζει να σημειωθεί ότι για 16-way, 32-way και 64-way έχουμε μηδενισμό των misses με αποτέλεσμα μεγιστοποίηση του IPC.

Ο δείκτης mpki παρουσιάζει συμφωνία με το ipc δηλαδή γίνεται καλύτερος δηλαδή μειώνεται όσο μεγαλώνει το tlb size μέχρι το όριο των 32 ent. και εξαρτάται με αντίστοιχο τρόπο με το assoc όπως και το ipc.

Άρα γενικά βλέπουμε:

- Όσον αφορα το TLB Size: Η αύξηση του tlb size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 32 entr. οπου μετά το οποίο δεν επηρεάζει σχεδόν καθόλου
- Όσον αφορά το Associativity: Για αύξηση του assoc από 1-2 έχουμε βελτίωση της επίδοσης, για αύξηση από 2-4 δεν παρατηρούμε κάποια βελτίωση, για αύξηση από 4-8 παρατηρούμε μια πολύ μεγάλη βελτίωση και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει την επίδοση.

Bodytrack



Παρατηρούμε ότι υπάρχει εξάρτηση τόσο από το TLBSize όσο και από το Assoc.

Αναλυτικά, όσο αυξάνει το TLB Size βελτιώνεται το ipc αυτό φαίνεται στεις πρώτες τρείς καταγραφές αλλα και στις τρεις τελευταίες, όπου με αύξηση του tlb size πετυχαίνουμε αύξηση του ipc.

Επίσης το assoc παρουσιάζει ένα κατώφλι: την τιμή 1 όπου δίνει πολύ κακή τιμή στο ipc, και για τιμές μεγαλυτερες τις μονάδας φαίνεται να μην επιρεάζει καθοριστικά το αποτέλεσμα, αυτό γίνεται κατανοητό στις μετρήσεις για TlbSize = 32/64/128 entries και με associativity ≠ 1-way οπου ενώ αυξάνουμε το assoc έχουμε μία μικρή αυξηση του ipc.

Αξίζει να σημειωθεί πως για 256 entries εκμηδενίζεται το MPKI και η απόδοση λαμβάνει τη μέγιστη τιμή της.

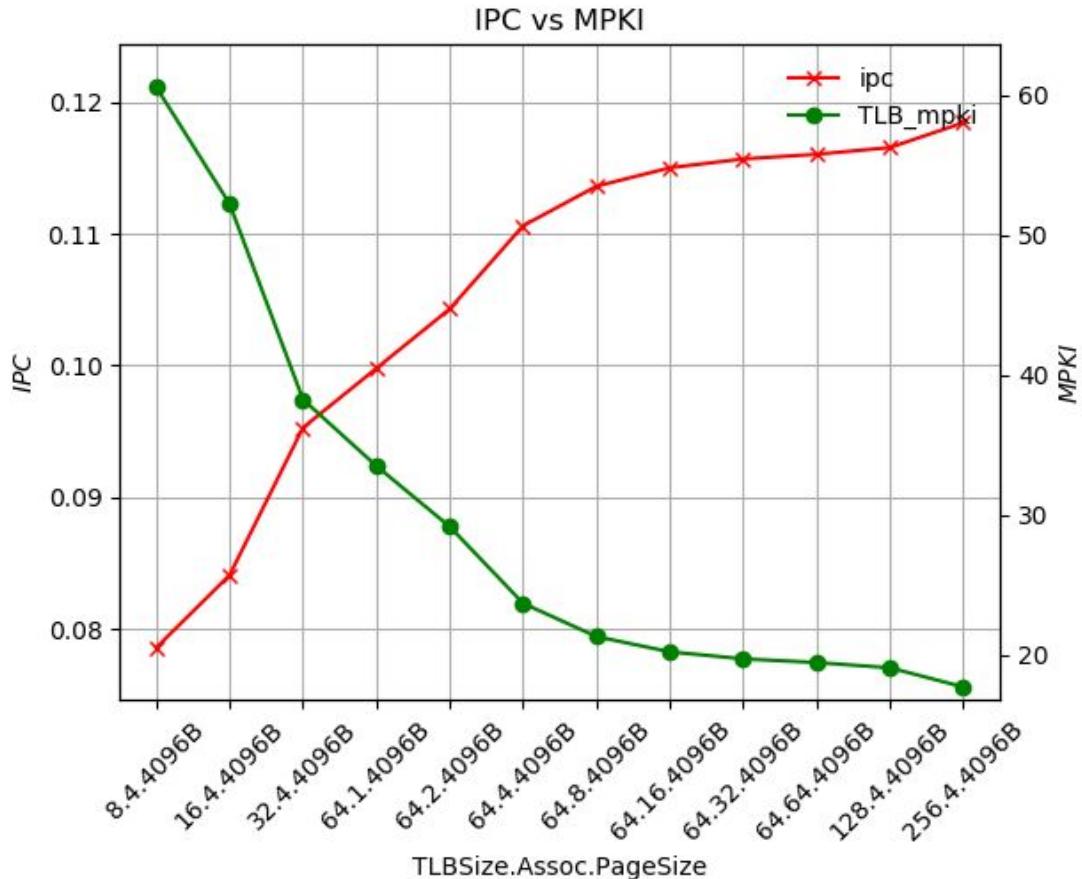
Με βασει αυτα τα συμπεράσματα θα μπορούσαμε να πούμε ότι η ιδανική τιμή του tlb είναι να έχουμε μεγάλο tlb size και assoc πάνω από το κατώφλι του 1.

Ο δείκτης mpki παρουσιάζει συμφωνία με το ipc δηλαδή γίνεται καλύτερος δηλαδή μειώνεται όσο μεγαλώνει το tlb size και εξαρτάται με αντίστοιχο τρόπο με το assoc όπως και το ipc.

Άρα γενικά βλέπουμε:

- Όσον αφορα το TLB Size: Η αύξηση του tlb size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc από 1-2 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Canneal



Παρατηρούμε ότι υπάρχει εξάρτηση τόσο από το TLBSize όσο και από το Assoc.

Αναλυτικά, όσο αυξάνει το TLB Size βελτιώνεται το ipc αυτό φαίνεται στις πρώτες τρεις καταγραφές αλλα και στις τρεις τελευταίες, όπου με αύξηση του tlb size πετυχαίνουμε αύξηση του ipc.

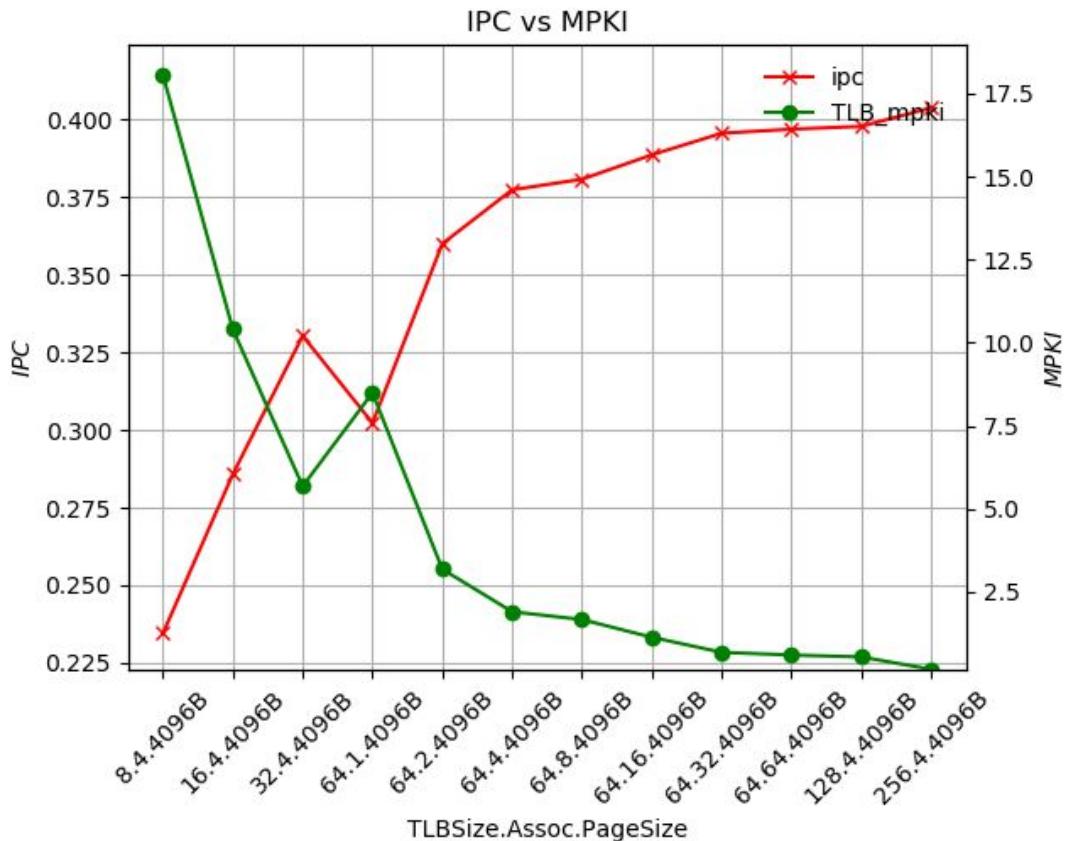
Επίσης το assoc παρουσιάζει ένα κατώφλι: την τιμή 8 κάτω από την οποία τιμή όσο αυξάνουμε το assoc έχουμε αύξηση στο ipc ενώ για τιμές μεγαλύτερες του 8 φαίνεται να μην επηρεάζει καθοριστικά το αποτέλεσμα, αυτό γίνεται φανερό όπου για TlbSize=64 και assoc = 8/16/32/64 που ενώ αυξάνουμε το assoc έχουμε μία μικρή αύξηση του ipc.

Ο δείκτης mpki παρουσιάζει συμφωνία με το ipc δηλαδή γίνεται καλύτερος δηλαδή μειώνεται όσο μεγαλώνει το tlb size και εξαρτάται με αντίστοιχο τρόπο με το assoc όπως και το ipc.

Άρα γενικά βλέπουμε:

- Όσον αφορά το TLB Size: Η αύξηση του tlb size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=8 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Facesim



Παρατηρούμε ότι υπάρχει εξάρτηση τόσο από το TLBSize όσο και από το Assoc.

Αναλυτικά, όσο αυξάνει το TLB Size βελτιώνεται το ipc αυτό φαίνεται στις πρώτες τρείς καταγραφές αλλα και στις τρεις τελευταίες, όπου με αύξηση του tlb size πετυχαίνουμε αύξηση του ipc.

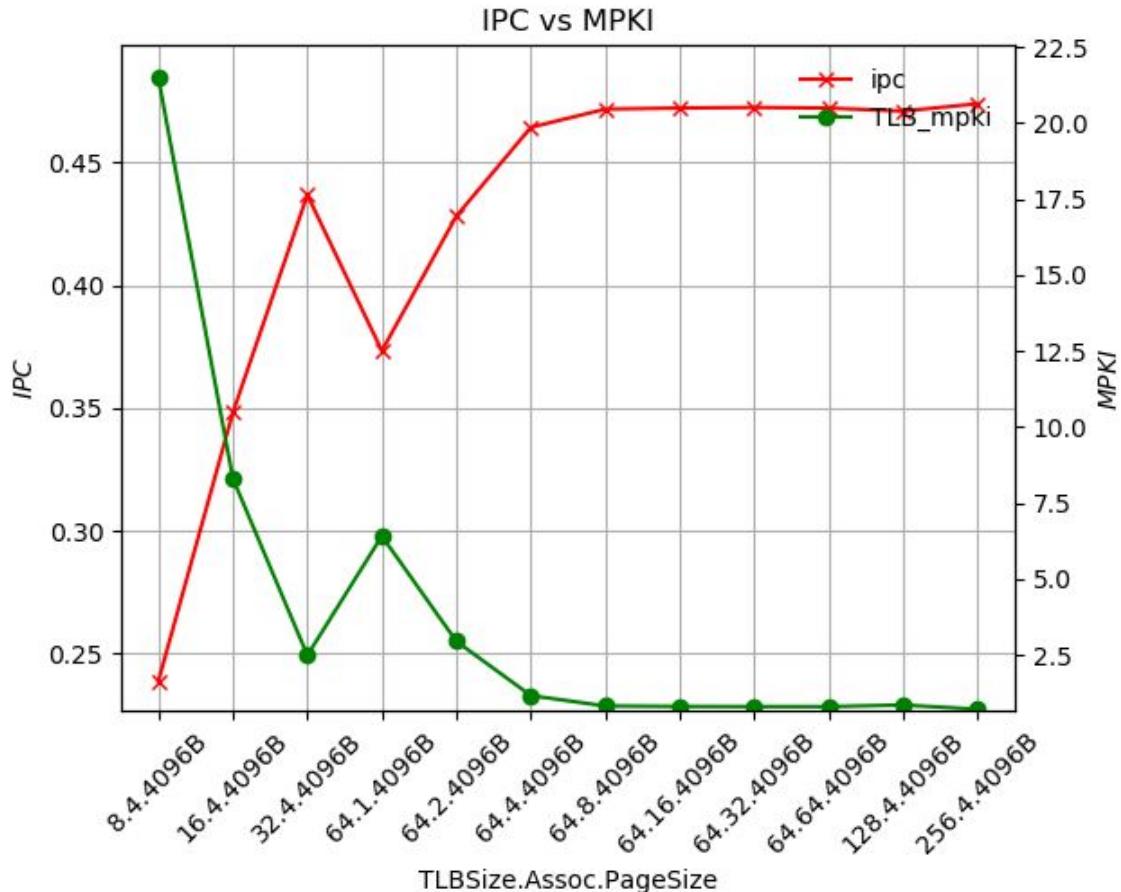
Επίσης το assoc παρουσιάζει τρία κατώφλια: την τιμή 1 όπου δίνει πολύ κακή τιμή στο ipc, το κατώφλι 32, όπου μεταξύ 1-32 όσο το assoc αυξάνει αυξάνει και το ipc και για πάνω από 32 δίνει την καλύτερη τιμή στο ipc αλλα για οποιαδήποτε άλλη αύξηση δεν παρατηρείται καμία βελτίωση του ipc.

Ο δείκτης mpki παρουσιάζει συμφωνία με το ipc δηλαδή γίνεται καλύτερος δηλαδή μειώνεται όσο μεγαλώνει το tlb size και εξαρτάται με αντίστοιχο τρόπο με το assoc όπως και το ipc.

Άρα γενικά βλέπουμε:

- Όσον αφορά το TLB Size: Η αύξηση του tlb size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=8 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Ferret



Παρατηρούμε ότι υπάρχει εξάρτηση τόσο από το TLBSize όσο και από το Assoc.

Αναλυτικά το tlb size βελτιώνει την απόδοση, αλλά μέχρι ενός σημείου, το οποίο σε αυτή την περίπτωση είναι τα 64 entries. Για την αλλαγή από 64 σε 128, από 128 σε 256k.o.k. υπάρχει πολύ μικρή βελτίωση.

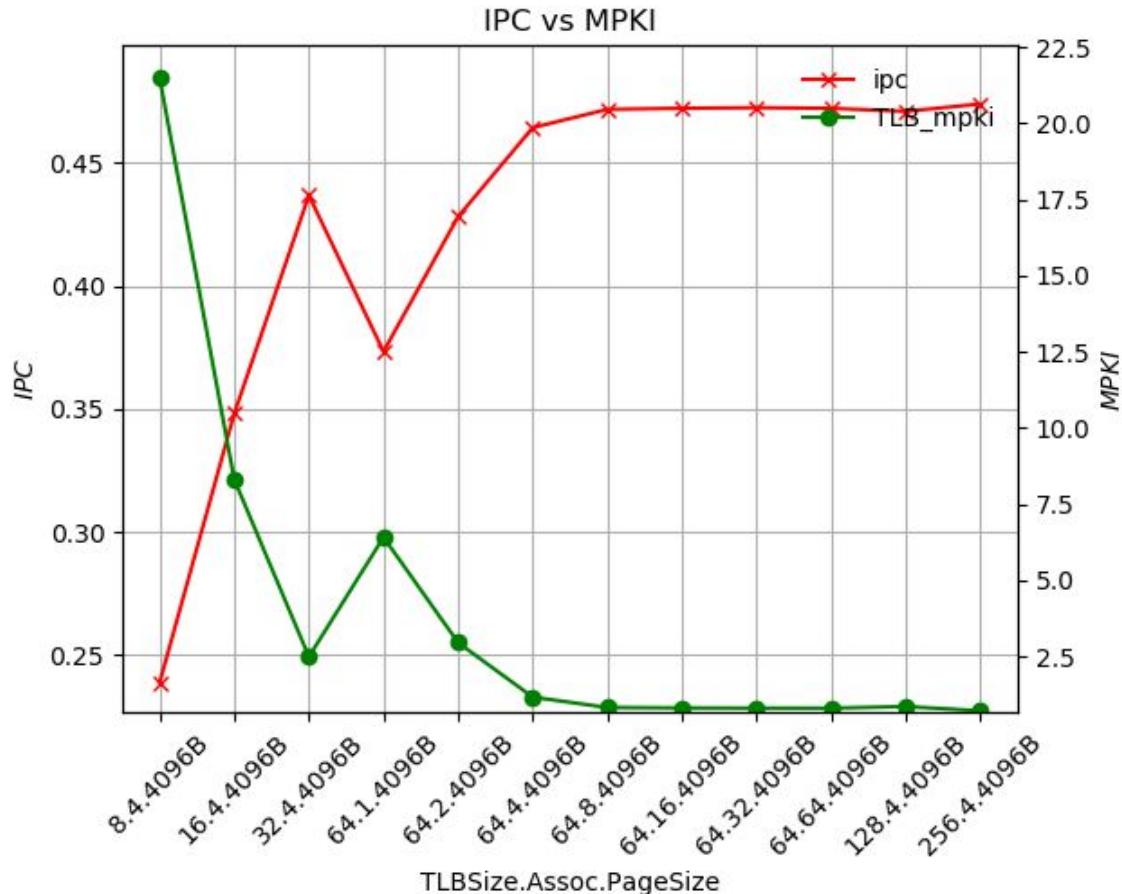
Επίσης το assoc παρουσιάζει τρία κατώφλια: την τιμή 1 όπου δίνει πολύ κακή τιμή στο ipc, το κατώφλι 8, όπου μεταξύ 1-8 όσο το assoc αυξάνει αυξάνει και το ipc και για πάνω από 8 δίνει την καλύτερη τιμή στο ipc αλλα για οποιαδήποτε άλλη αύξηση δεν παρατηρείται καμία βετίωση του ipc.

Ο δείκτης mpki παρουσιάζει συμφωνία με το ipc δηλαδή γίνεται καλύτερος δηλαδή μειώνεται όσο μεγαλώνει το tlb size μέχρι το όριο των 64 ent. και εξαρτάται με αντίστοιχο τρόπο με το assoc όπως και το ipc.

Άρα γενικά βλέπουμε:

- Όσον αφορα το TLB Size: Η αύξηση του tlb size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 64 entr. οπου μετά το οποίο δεν επηρεάζει σχεδόν καθόλου
- Όσον αφορά το Associativity: Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=8 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Fluidanimate



Παρατηρούμε ότι υπάρχει εξάρτηση τόσο από το TLBSize όσο και από το Assoc.

Αναλυτικά το tlb size βελτιώνει την απόδοση, αλλά μέχρι ενός σημείου, το οποίο σε αυτή την περίπτωση είναι τα 64 entries. Για την αλλαγή από 64 σε 128, από 128 σε 256 κ.ο.κ. υπάρχει πολύ μικρή βελτίωση.

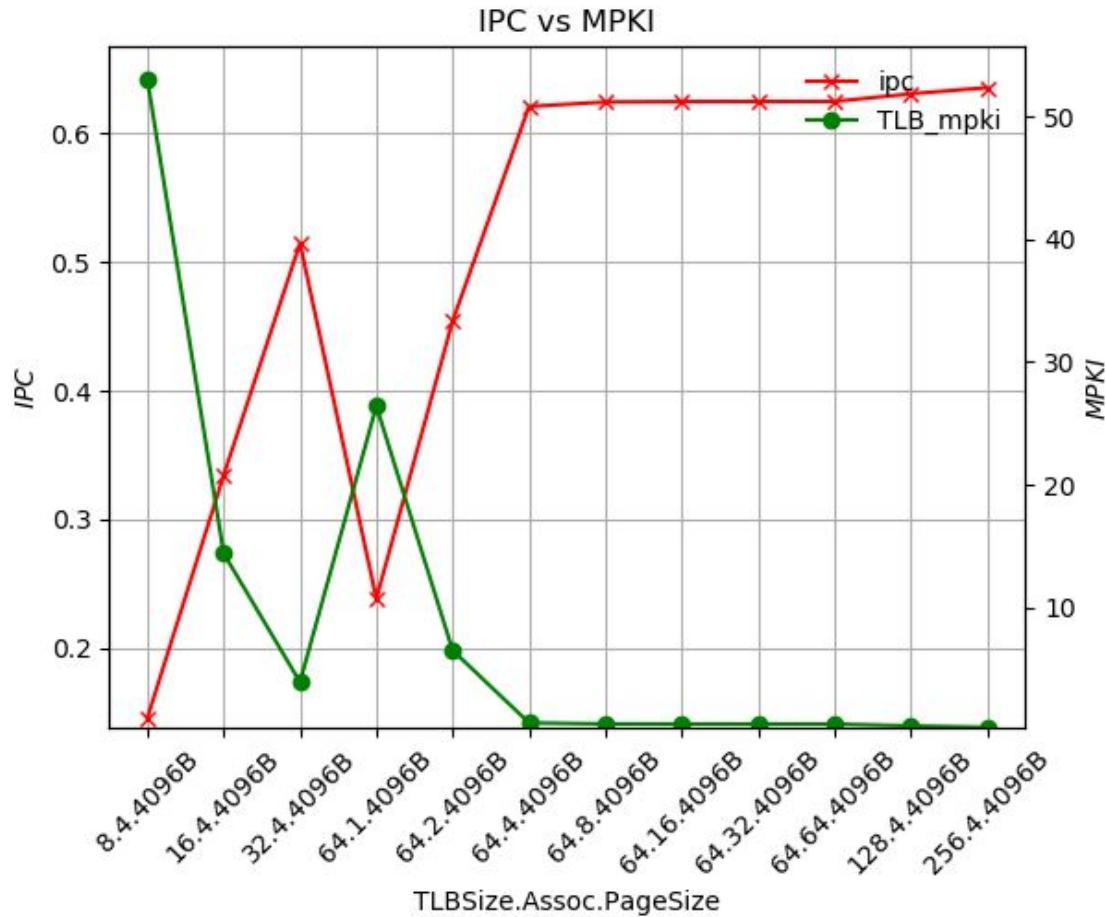
Επίσης το assoc παρουσιάζει τρία κατώφλια: την τιμή 1 όπου δίνει πολύ κακή τιμή στο ipc, το κατώφλι 8, όπου μεταξύ 1-8 όσο το assoc αυξάνει αυξάνει και το ipc και για πάνω από 8 δίνει την καλύτερη τιμή στο ipc αλλα για οποιαδήποτε άλλη αύξηση δεν παρατηρείται καμία βελτίωση του ipc.

Ο δείκτης mpki παρουσιάζει συμφωνία με το ipc δηλαδή γίνεται καλύτερος δηλαδή μειώνεται όσο μεγαλώνει το tlb size μέχρι το όριο των 64 ent. και εξαρτάται με αντίστοιχο τρόπο με το assoc όπως και το ipc.

Άρα γενικά βλέπουμε:

- Όσον αφορά το TLB Size: Η αύξηση του tlb size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 64 entr. οπου μετά το οποίο δεν επηρεάζει σχεδόν καθόλου
- Όσον αφορά το Associativity: Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=8 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Freqmine



Παρατηρούμε ότι υπάρχει εξάρτηση τόσο από το TLBSize όσο και από το Assoc. Αναλυτικά, όσο αυξάνει το TLB Size βελτιώνεται το ipc αυτό φαίνεται στις πρώτες τρεις καταγραφές αλλα και στις τρεις τελευταίες, όπου με αύξηση του tlb size πετυχαίνουμε αύξηση του ipc.

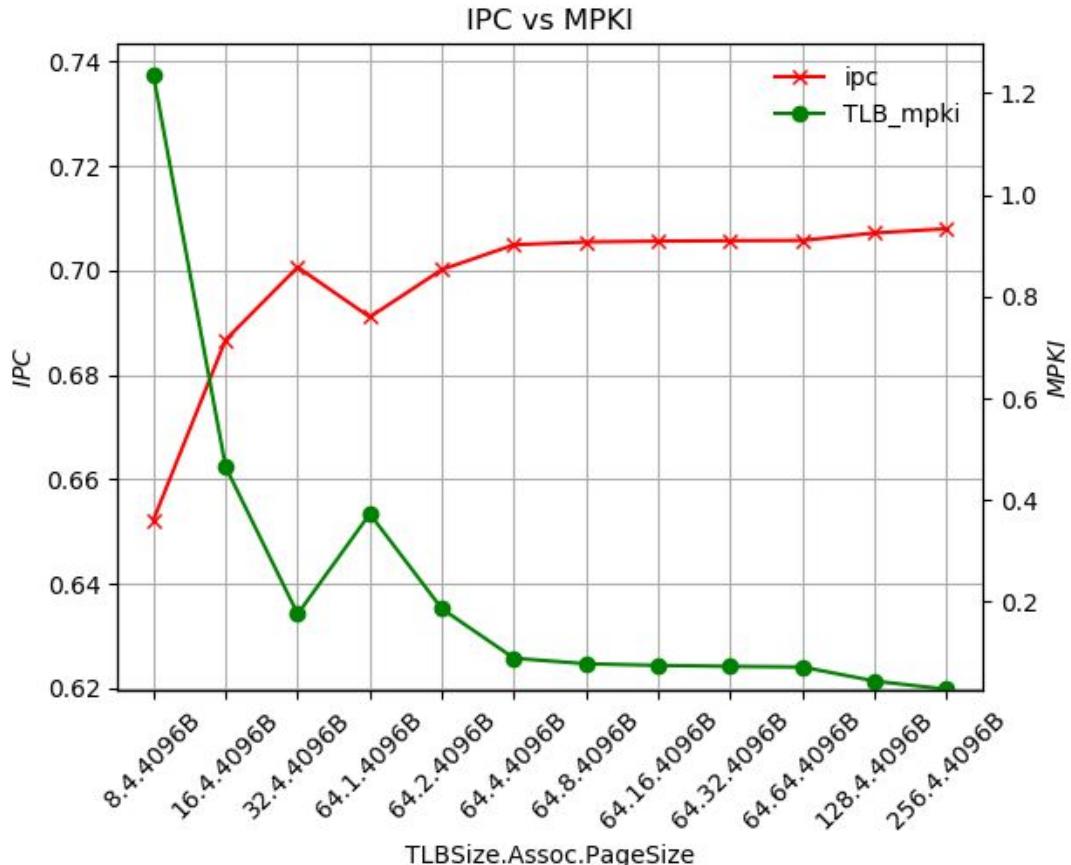
Επίσης το assoc παρουσιάζει τρία κατώφλια: την τιμή 1 όπου δίνει πολύ κακή τιμή στο ipc, το κατώφλι 4, όπου μεταξύ 1-4 όσο το assoc αυξάνει αυξάνει και το ipc και για πάνω από 4 δίνει την καλύτερη τιμή στο ipc αλλα για οποιαδήποτε άλλη αύξηση δεν παρατηρείται καμία βελτίωση του ipc.

Ο δείκτης mpki παρουσιάζει συμφωνία με το ipc δηλαδή γίνεται καλύτερος δηλαδή μειώνεται όσο μεγαλώνει το tlb size κι εξαρτάται με αντίστοιχο τρόπο με το assoc όπως και το ipc.

Άρα γενικά βλέπουμε:

- Όσον αφορά το TLB Size: Η αύξηση του tlb size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

rtview



Παρατηρούμε ότι υπάρχει εξάρτηση τόσο από το TLBSize όσο και από το Assoc.

Αναλυτικά, όσο αυξάνει το TLB Size βελτιώνεται το ipc αυτό φαίνεται στις πρώτες τρεις καταγραφές αλλα και στις τρεις τελευταίες, όπου με αύξηση του tlb size πετυχαίνουμε αύξηση του ipc.

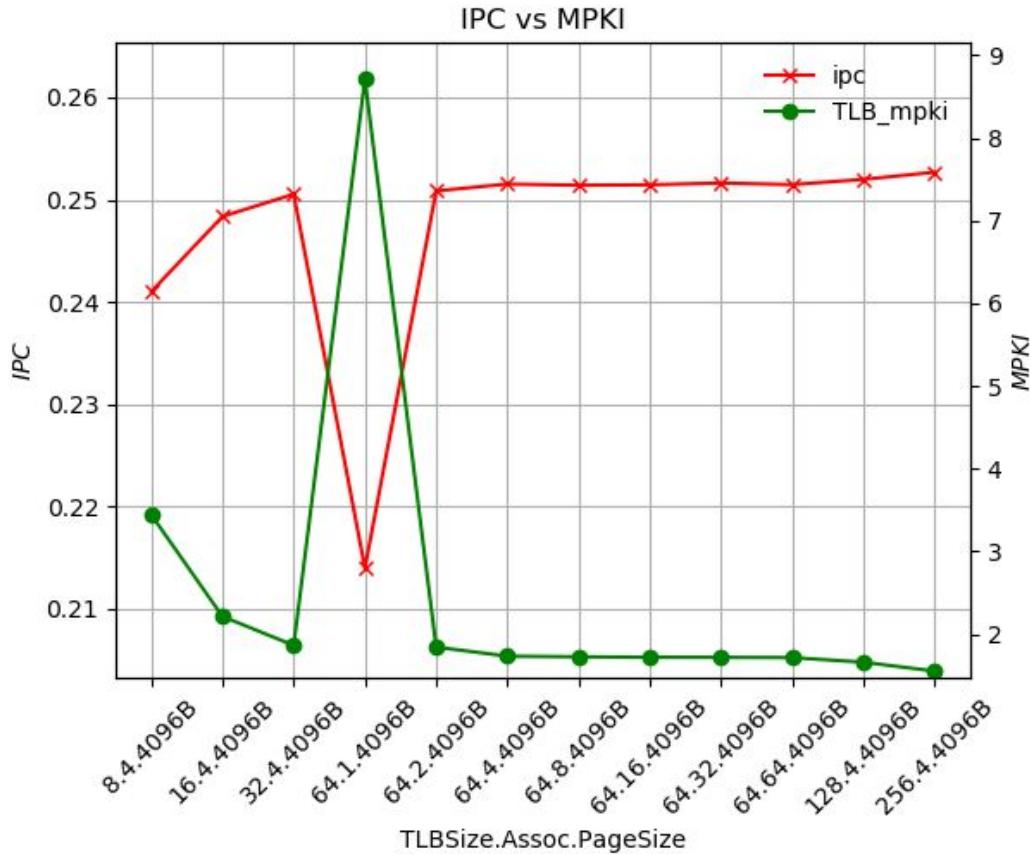
Επίσης το assoc παρουσιάζει τρία κατώφλια: την τιμή 1 όπου δίνει κακή τιμή στο ipc, το κατώφλι 4, όπου μεταξύ 1-4 όσο το assoc αυξάνει αυξάνει και το ipc και για πάνω από 4 δίνει την καλύτερη τιμή στο ipc αλλα για οποιαδήποτε άλλη αύξηση δεν παρατηρείται καμία βελτίωση του ipc.

Ο δείκτης mpki παρουσιάζει συμφωνία με το ipc δηλαδή γίνεται καλύτερος δηλαδή μειώνεται όσο μεγαλώνει το tlb size κι εξαρτάται με αντίστοιχο τρόπο με το assoc όπως και το ipc.

Άρα γενικά βλέπουμε:

- Όσον αφορά το TLB Size: Η αύξηση του tlb size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Streamcluster



Παρατηρούμε ότι υπάρχει εξάρτηση τόσο από το TLBSize όσο και από το Assoc.

Αναλυτικά, όσο αυξάνει το TLB Size βελτιώνεται το ipc αυτό φαίνεται στις πρώτες τρεις καταγραφές αλλα και στις τρεις τελευταίες, όπου με αύξηση του tlb size πετυχαίνουμε αύξηση του ipc.

Επίσης το assoc παρουσιάζει ένα κατώφλι: την τιμή 1 όπου δίνει πολύ κακή τιμή στο ipc, και για τιμές μεγαλύτερες του 1 φαίνεται να μην επηρεάζει καθοριστικά το αποτέλεσμα.

Αξίζει να σημειωθεί πως για 256 entries εκμηδενίζεται το MPKI και η απόδοση λαμβάνει τη μέγιστη τιμή της.

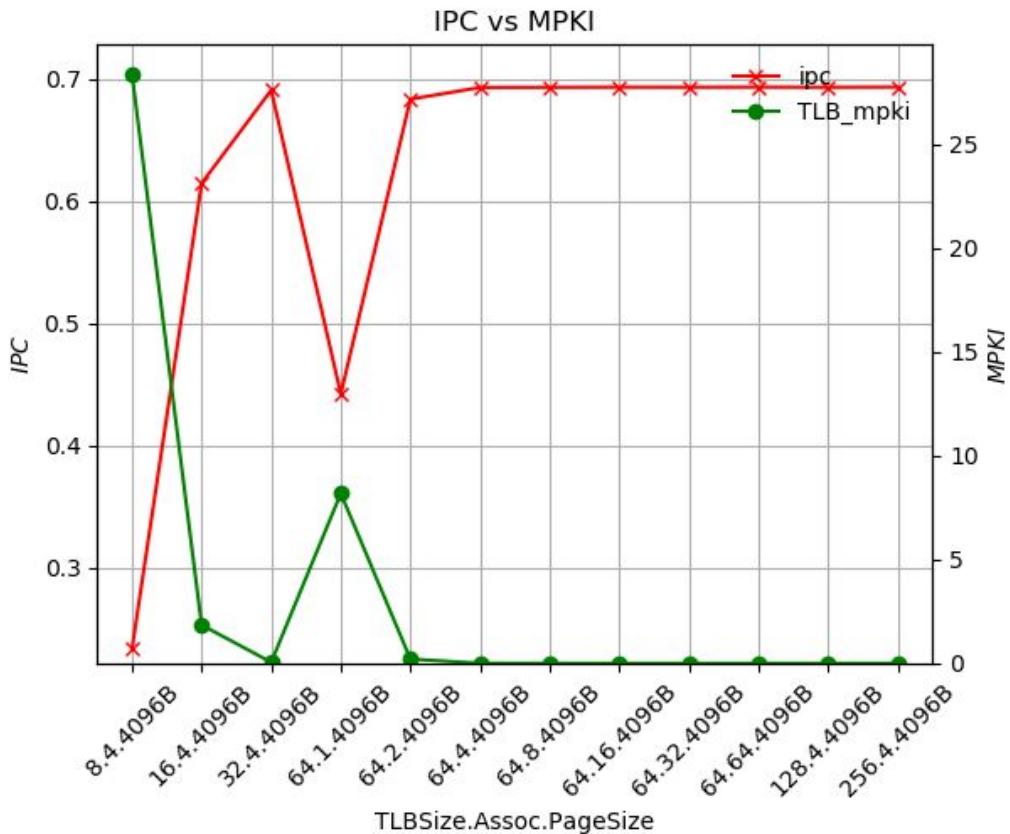
Με βασει αυτα τα συμπεράσματα θα μπορούσαμε να πούμε ότι η ιδανική τιμή του tlb είναι να έχουμε μεγάλο tlb size και assoc πάνω από το κατώφλι του 1.

Ο δείκτης mpki παρουσιάζει συμφωνία με το ipc δηλαδή γίνεται καλύτερος δηλαδή μειώνεται όσο μεγαλώνει το tlb size κι εξαρτάται με αντίστοιχο τρόπο με το assoc όπως και το ipc.

Άρα γενικά βλέπουμε:

- Όσον αφορα το TLB Size: Η αύξηση του tlb size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=2 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Swaptions



Γενικά βλέπουμε:

- Όσον αφορα το TLB Size: Η αύξηση του tlb size **βελτιώνει** την επίδοση του προγράμματος μέχρι το κατώφλι των 32 entr. από εκεί και πέρα δεν επηρεάζει την επίδοση του συστήματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=2 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

11. Prefetching Policy για την L2-cache

Η τεχνική του Cache Prefetching (Προανάκληση στην κρυφή μνήμη) είναι μια τεχνική η οποία χρησιμοποιείται από τους περισσότερους σύγχρονους επεξεργαστές με σκοπό τη βελτίωση της επίδοσης. Η έννοια του prefetching στηρίζεται στη λογική του να φέρνεις δεδομένα ή εντολές από την κύρια μνήμη – η οποία είναι σημαντικά πιο αργή – σε μια τοπική, ταχύτερη μνήμη πριν έρθει η ώρα να χρησιμοποιηθούν ώστε να βρίσκονται ήδη εκεί όταν ζητηθούν από τον επεξεργαστή. Το γεγονός πως τα διάφορα επίπεδα κρυφών μνημών παρουσιάζουν τόσο μεγάλη διαφορά στην ταχύτητα σε σχέση με την κύρια μνήμη καθώς και το γεγονός πως τα προγράμματα είναι σχεδιασμένα έτσι ώστε τα δεδομένα και οι εντολές συνήθως να ζητούνται σειριακά από τον επεξεργαστή κάνουν το prefetching να είναι μια σημαντική τεχνική, η οποία βελτιστοποιεί συνήθως αρκετά το IPC.

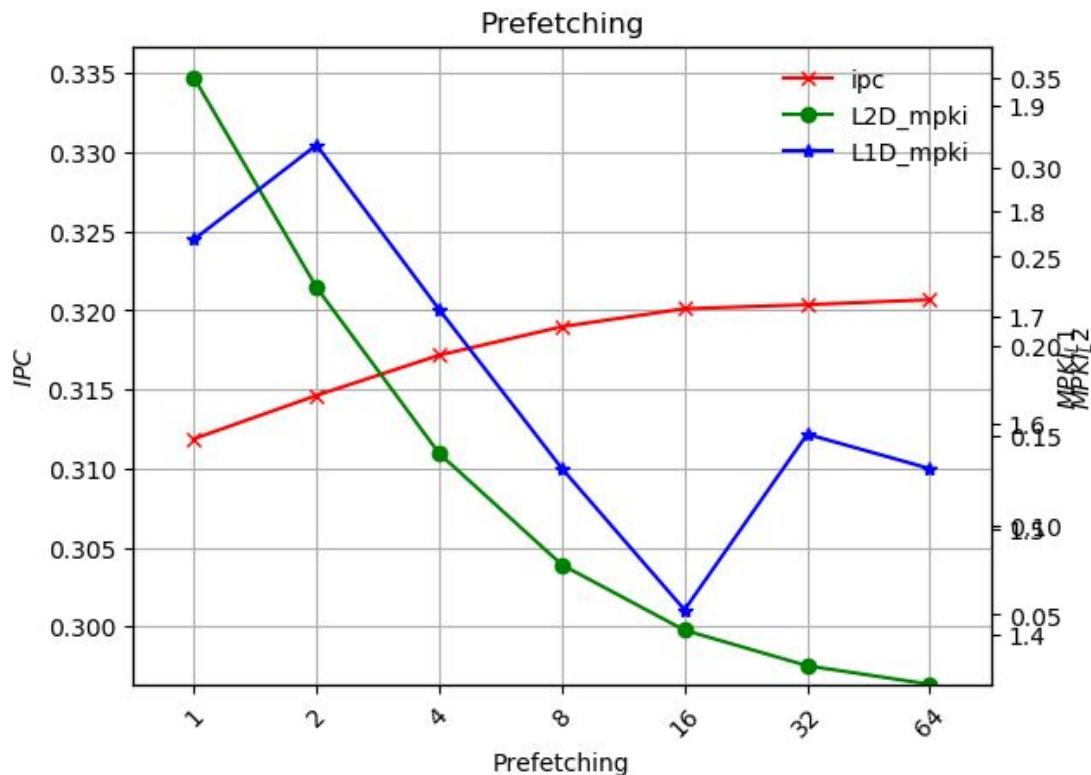
Παρακάτω θα δούμε πως η τεχνική του prefetching επηρεάζει το IPC, καθώς και τους δείκτες MPKI των caches L1 και L2. Για να κάνουμε την παρακάτω μελέτη χρειάστηκε να τροποποιήσουμε τον κώδικα ο οποίος υλοποιεί τις caches (ο νέος τροποποιημένος κώδικας παρουσιάζεται στο παράρτημα στο τέλος της εργασίας). Επιπλέον, το γεγονός πως η L2 είναι inclusive της L1 σημαίνει πώς πρέπει να γίνεται έλεγχος ώστε η L1 να παραμένει κάθε στιγμή subset της L2.

Για τις ανάγκες αυτού του ερωτήματος οι τιμές των παραμέτρων των caches και του TLB που χρησιμοποιήθηκαν σύμφωνα με την εκφώνηση είναι οι παρακάτω:

- | | | |
|-----------------------|-------------------------|--------------------------|
| ➤ L1 size = 32 KB | ➤ L1 block size = 64 B | ➤ L1 associativity = 8 |
| ➤ L2 size = 1024KB | ➤ L2 block size = 128 B | ➤ L2 associativity = 8 |
| ➤ TLB size = 64 entr. | ➤ TLB associativity = 4 | ➤ TLB page size = 4096 B |

Τέλος, η μελέτη έγινε για τιμές 0, 1, 2, 4, 8, 16, 32, 64 και τα αποτελέσματα που παρουσιάστηκαν για τα δέκα benchmarks ομαδοποιούνται και παρουσιάζονται παρακάτω:

Blackscholes

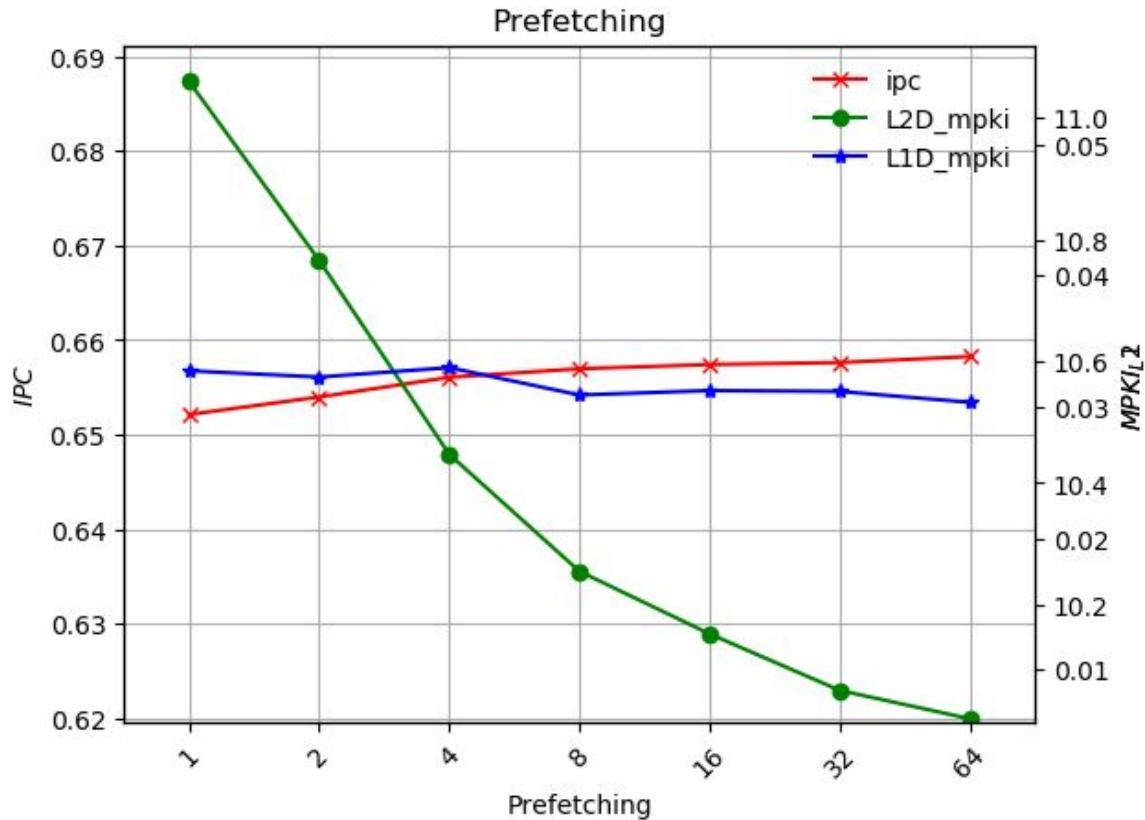


Παρατηρούμε ότι τα misses στην L2 , που σχετίζονται με τον δείκτη MPKI L2, μειώνεται όσο αυξάνουμε τον αριθμό των prefetched blocks.

Παρατηρούμε επίσεις ότι τα misses στην L1, που σχετίζονται με τον δείκτη MPKI L1, αυξομειώνονται κατά τις αλλαγές των prefetching, ωστόσο σε απόλυτο βαθμό είναι πολύ μικρός ο αριθμός τους και γι αυτό επηρεάζουν πολύ λίγο την απόδοση. Πιο αναλυτικά βλέπουμε ότι μεταξύ 2-16 την απόδοση την επιταχύνει το L2 αλλά μεταξύ 16-32 όπου τα misses του L1 αυξάνονται έχουμε μια πιο αργή βελτίωση του ipc, δηλαδή δεν είναι αρκετά μεγάλο το πλήθος των misses έτσι ώστε να αναστείλουν την βελτιωτική πορεία των ipc και απλά την επιβραδύνουν. Αντίστοιχα μεταξύ 32-64 βλέπουμε ότι βελτιώνονται τα misses του L1 και για αυτό παρατηρούμε μια εκ νέου επιτάχυνση της βελτίωσης του ipc.

Ωστόσο είναι ξεκάθαρο ότι καθοριστικό παραγοντα εδώ παίζει η βελτίωση των misses του L2 που είναι πολύ μεγαλύτερα σε απόλυτο αριθμό σε σχέση με τα αντίστοιχα misses του L1 και έτσι έχουμε μια συνεχόμενη βελτίωση του ipc όσο αυξάνεται το prefetching.

Bodytrack

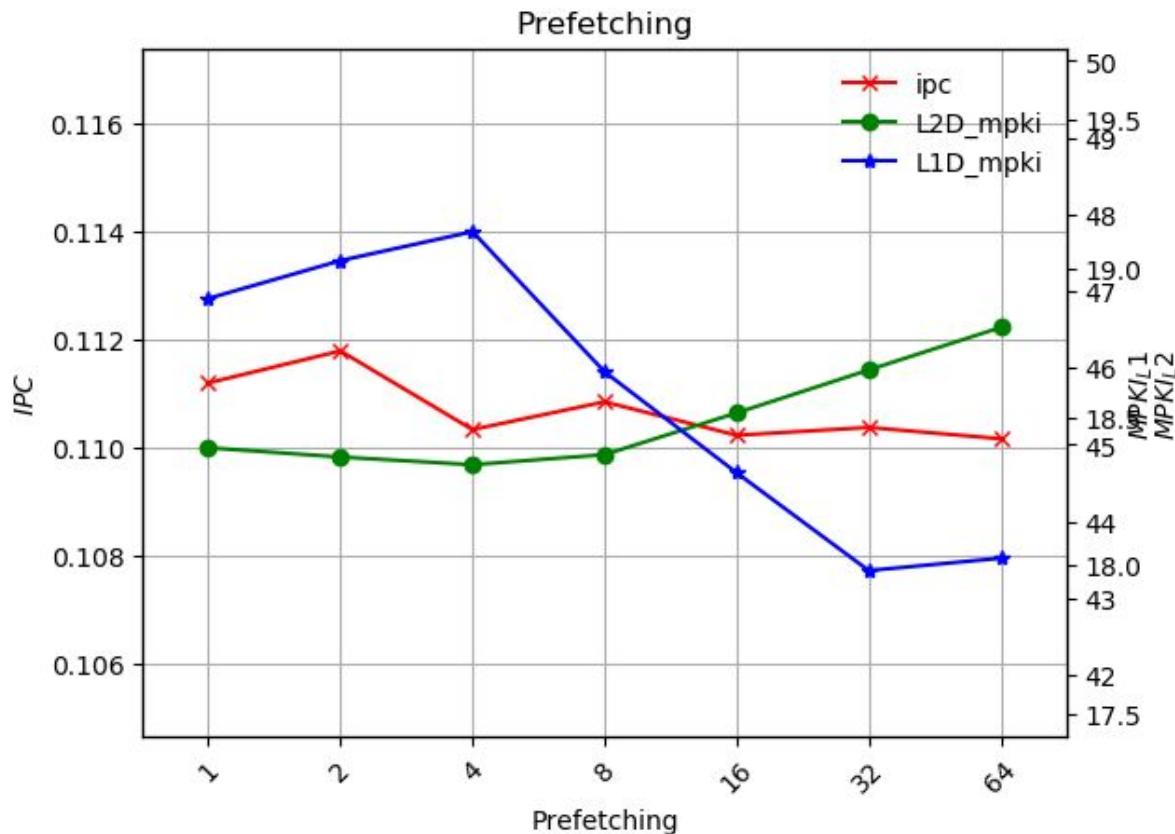


Παρατηρούμε ότι με την αύξηση του prefetching τα misses στην L1 κατα διαστήματα είτε φθήνουν είτε παραμένουν σταθερά.

Επίσης έχουμε διαρκής μείωση των misses στην L2, αλλά είναι πολύ μικρή σε απόλυτο αριθμό, επομένως το ipc καθορίζεται κυρίως από τα misses στην L1.

Έτσι, στα διαστήματα που τα misses της L1 φθίνουν βλέπουμε αντίστοιχη βελτίωση του ipc ενώ στα διαστήματα όπου τα misses της L1 παραμένουν σταθερά βλέπουμε μια πιο αργή βελτίωση του ipc, που οφείλεται στην συνεχής φθίνουσα πορεία των misses της L2, προφανώς αυτή η φθίνουσα πορεία επηρεάζει το ipc και στα διαστήματα που η L1 φθίνει αλλά επηρεάζει σε λιγότερο βαθμό από ότι επηρεάζει η L1.

Canneal



Εδώ παρατηρούμε ότι σε γενικές γραμμές η απόδοση δε μεταβάλλεται σε σχέση με το prefetching, δηλαδή όποια μεταβολή παρατηρείται την βλέπουμε μόνο στο τρίτο δεκαδικό του ipc.

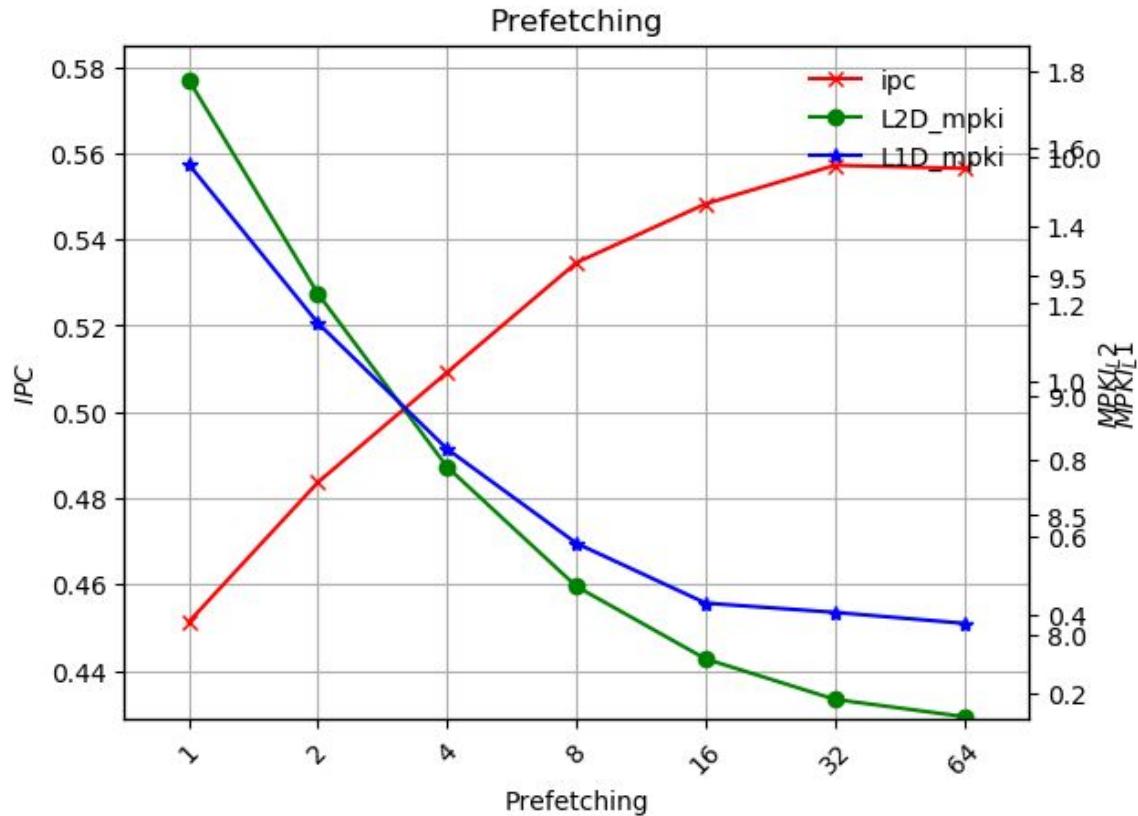
Τα misses της L1 χειροτερεύουν όσο αυξάνει το prefetching μέχρι την τιμή 4 μετά βελτιώνονται μέχρι την τιμή 32 και μετα χειροτερεύουν λίγο για prefetching από 32-64.

Τα misses της L2 στο διάστημα από 1-2 βλέπουμε ότι βελτιώνονται κατα ένα μικρό βαθμό μετα στο διάστημα 2-4 βελτιώνονται κατα ένα μεγαλύτερο βαθμο και από το 4 και μετα χειροτερεύουν πολύ.

Επίσης η απόλυτη τιμή της διαφοράς των misses της L1 και L2 είναι συγκρίσιμη οπότε επηρεάζουν και οι δυο τιμές αλλά το ipc εξαρτάται παραπάνω από το L1 (εξαιτίας της αρχιτεκτονικής).

Έτσι, στο πρώτο διάστημα, 1-2, βλέπουμε ότι ενώ το mpkiL1 βελτιώνεται και το mpkiL2 χειροτερεύει το ipc βελτιώνεται, στο δευτέρου διάστημα όπου το mpkiL1 συνεχίζει να βελτιώνεται αλλά το mpkiL2 χειροτερεύει με γρηγορότερο ρυθμό βλέπουμε ότι το ipc επηρεάζεται και αυτο και χειροτερεύει. Μετά στην συνέχεια που ο ένας δείκτης βελτιώνεται και ο άλλος χειροτερεύει βλέπουμε ότι το ipc παραμένει περίπου σταθερό.

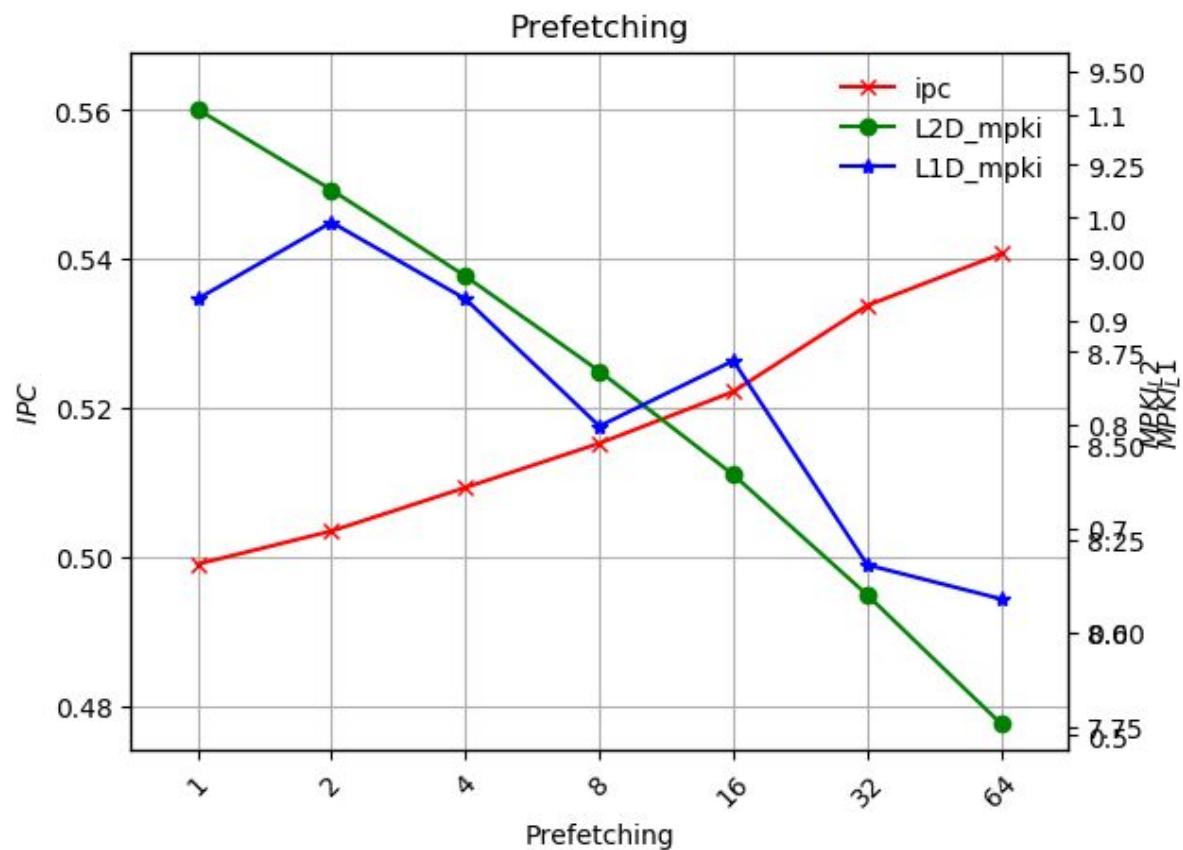
Facesim



Παρατηρούμε ότι και το mpkiL1 και το mpkiL2 φθίνουν, δηλαδή βελτιώνονται όσο αυξάνει το prefetching άρα και το ipc βελτιώνεται όσο αυξάνεται το prefetching.

Επίσης η απόλυτη τιμή της διαφοράς των misses της L1 και L2 είναι συγκρίσιμη οπότε επηρεάζουν και οι δυο τιμές αλλά το ipc εξαρτάται παραπάνω από το L1 (εξαιτίας της αρχιτεκτονικής), ο επιρεασμός κυρίως από το L1 φαίνεται στο διάστημα 32-64 όπου το mpkiL1 φθίνει με αργό ρυθμό και το mpkiL2 φθίνει πιο γρήγορα το ipc βελτιωνεται εν τελει πιο αργά.

Ferret

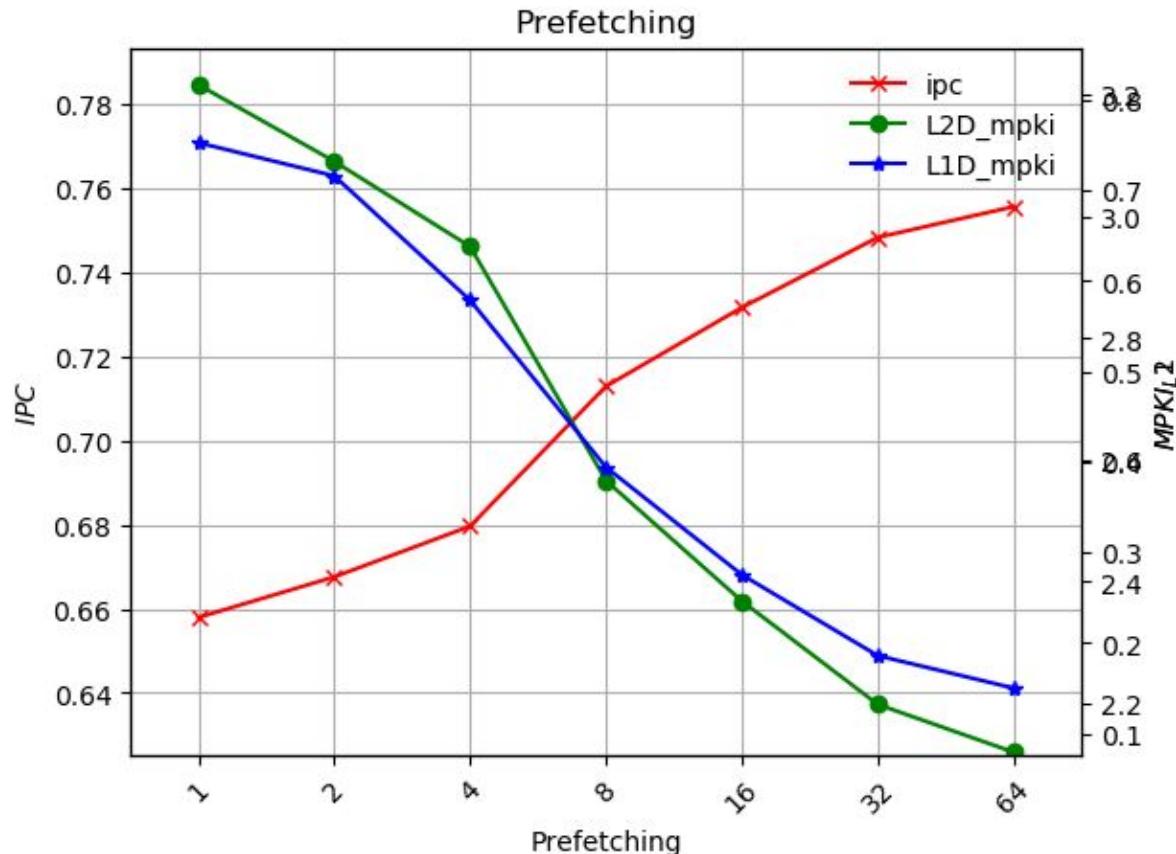


Παρατηρούμε ότι σε απόλυτο βαθμό η διαφορά του mpki L2 είναι μεγαλύτερη, επομένως επηρεάζει παραπάνω το ipc.

Το mpki L2 φθίνει συνεχώς, το mpki L1 σε κάποια διαστήματα φθίνει σε κάποια διαστήματα αυξάνει.

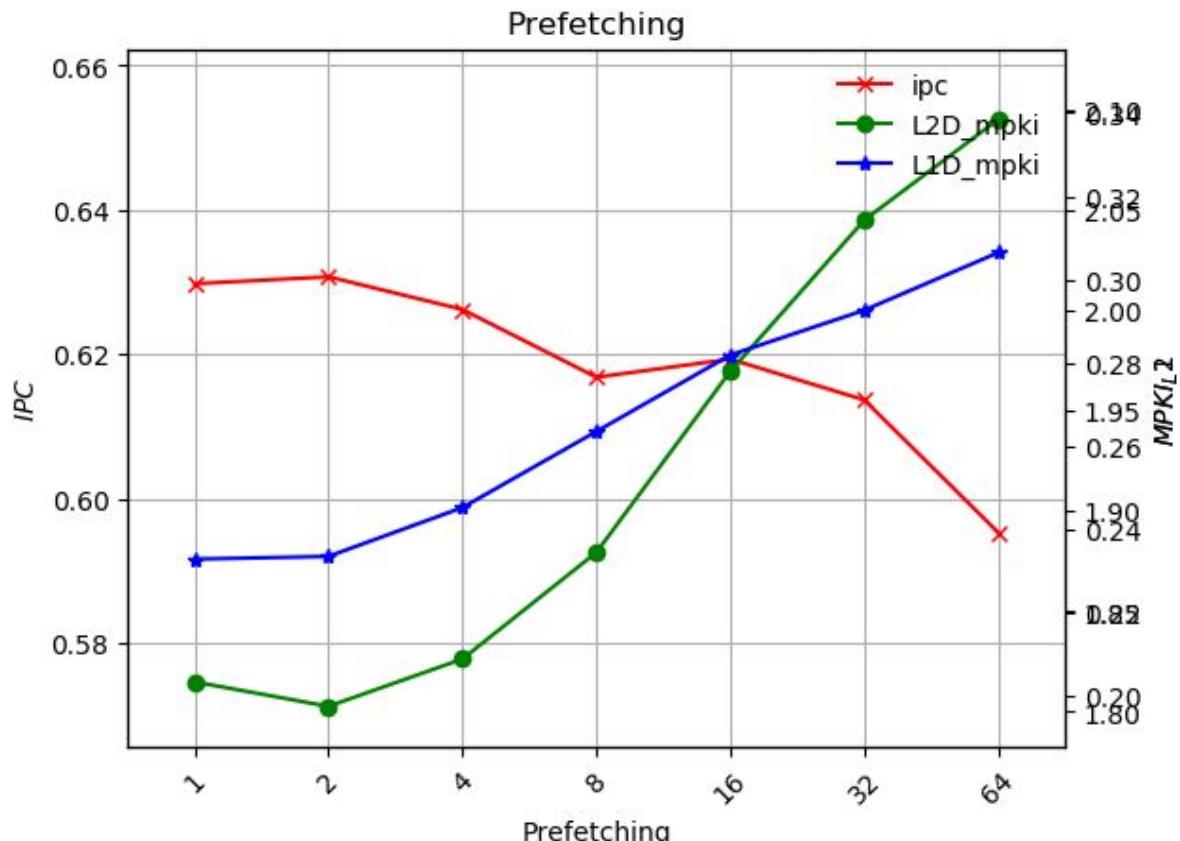
Έτσι, έχουμε ότι το ipc αυξάνεται, δηλαδή βελτιώνεται συνεχώς, αλλά στα αντίστοιχα σημεία που το L1 αυξάνεται δηλαδή χειροτερεύει, το ipc βελτιώνεται με λίγο πιο αργό ρυθμό.

Fluidanimate



Παρατηρούμε ότι και το mpkiL1 και το mpkiL2 φθίνουν, δηλαδή βελτιώνονται όσο αυξάνει το prefetching άρα και το ipc βελτιώνεται όσο αυξάνεται το prefetching.

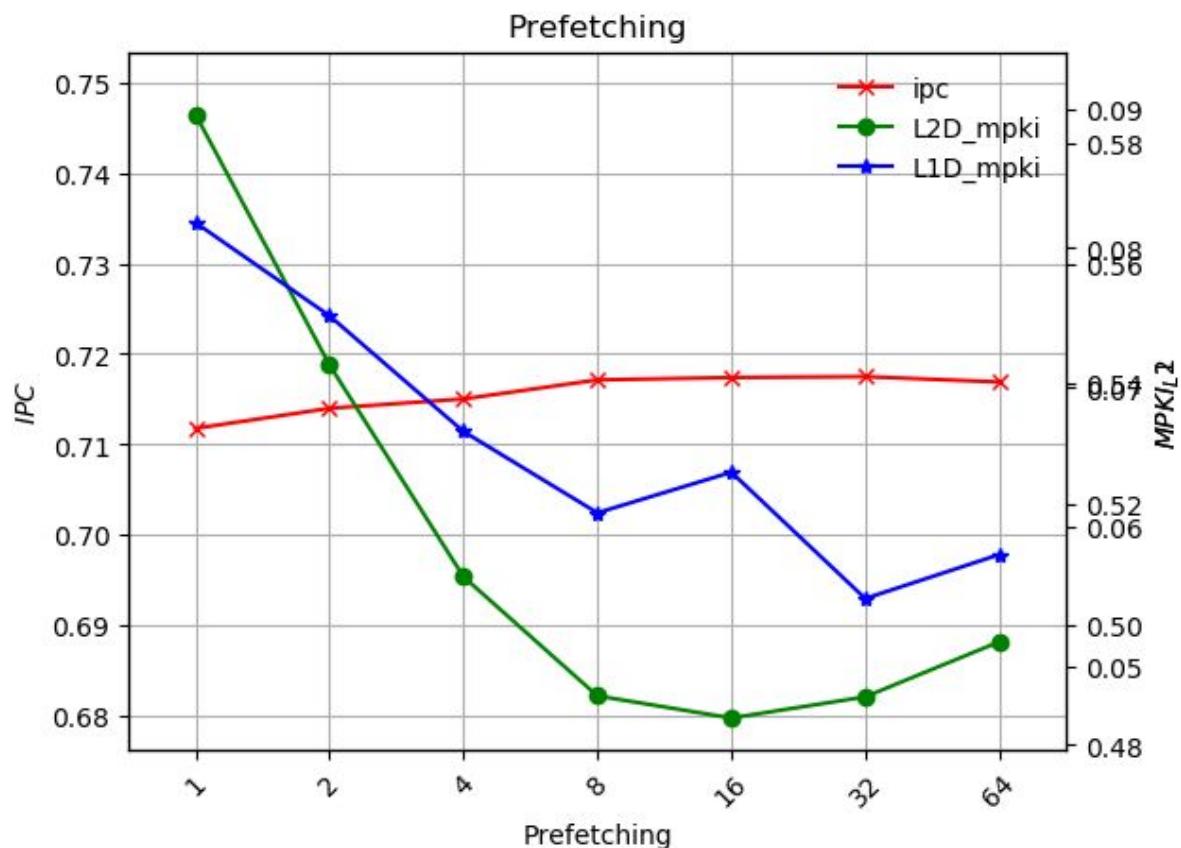
Επίσης η απόλυτη τιμή της διαφοράς των misses της L1 και L2 είναι συγκρίσιμη οπότε επηρεάζουν και οι δυο τιμές αλλά το ipc εξαρτάται παραπάνω από το L1 (εξαιτίας της αρχιτεκτονικής), ο επιρεασμός κυρίως από το L1 φαίνεται στο διάστημα 1-2 όπου το mpkiL1 φθίνει με αργό ρυθμό και το mpkiL2 φθίνει πιο γρήγορα το ipc βελτιώνεται εν τελεί πιο αργά.



Εδώ βλέπουμε ότι τόσο το mpkiL1 όσο και το mpkiL2 χειροτερεύουν με την αύξηση του prefetching, αλλού με γρηγορότερο ρυθμό αλλού με μικρότερο.

Επίσης η απόλυτη τιμή της διαφοράς των misses της L1 και L2 είναι συγκρίσιμη.

Επομένως παρατηρούμε μια διαρκής πτώση του ipc όσο αυξάνεται το prefetching, και μάλιστα στο διάστημα 16-64 όπου το mpkiL2 χειροτερεύει πολύ το ipc χειροτερεύει αρκετά απότομα.



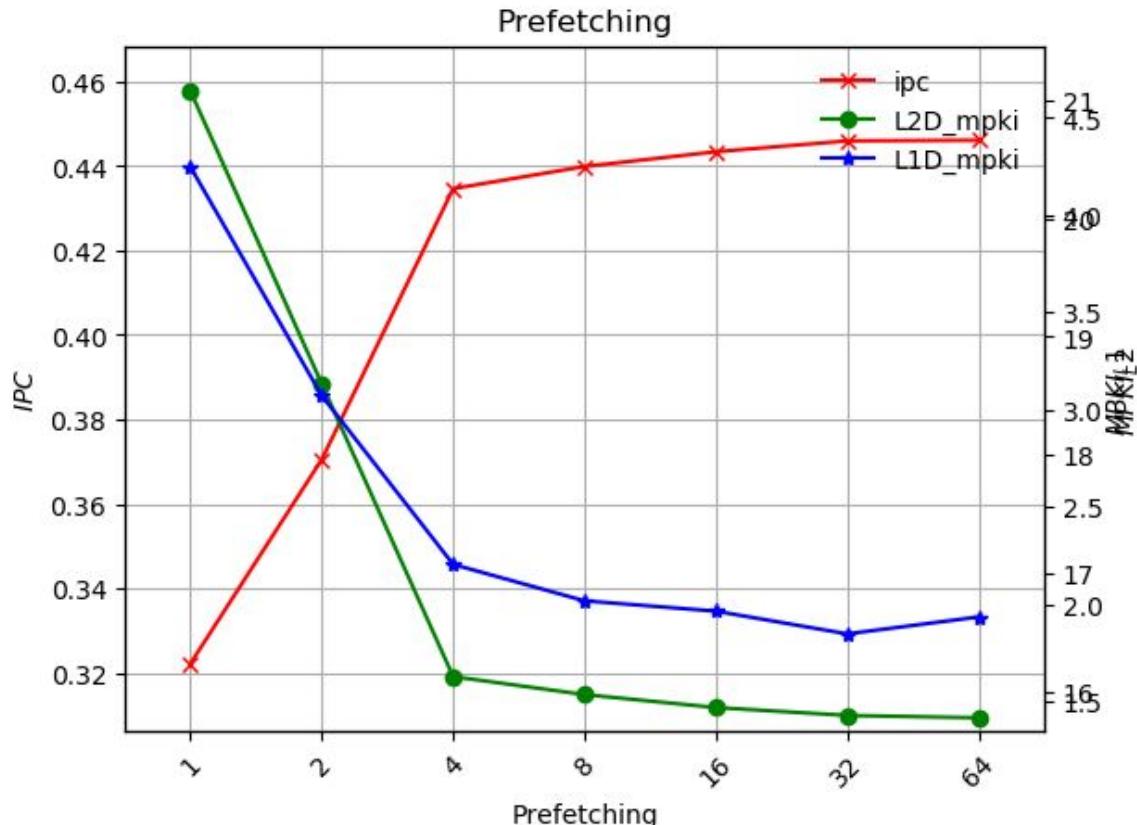
Παρατηρούμε ότι το mpkiL2 φθίνει συνεχώς μέχρι την τιμή 16 όπου μετά αυξάνεται.

Το mpkiL2 φθίνει συνεχώς με εξαίρεση το διάστημα 8-16 και 32-64.

Επίσης η απόλυτη τιμή της διαφοράς των misses της L1 και L2 είναι συγκρίσιμη οπότε επηρεάζουν και οι δυο τιμές αλλά το ipc, αλλά επειδή η απόλυτη τιμή της διαφοράς τους είναι αρκετά μικρή βλέπουμε πολύ μικρές διαφοροποιήσεις του ipc.

Έτσι γενικά το ipc βελτιώνεται μέχρι το prefetching να γίνει ίσο με 16 και μετά παραμένει περίπου σταθερό όπου στο διάστημα 32-64 φθίνει εξ αιτίας τις χειροτερευσεις αμφοτερων mpki.

streamcluster



Παρατηρούμε ότι και οι δυο δεικτες για τα misses παρουσιάζουν αντίστοιχη φθίνουσα πορεία με πιο έντονη πτώση στο διάστημα 1-4 και μετα μια πιο ήρεμη φθινουσα πορεία.

Και οι δυο δεικτες εχουν συγκρίσιμη απόλυτη διαφορά άρα επηρεάζουν σχεδόν εξ ίσου το ipc και έτσι βλέπουμε το ipc να βελτιώνεται πολύ γρήγορα στο διάστημα 1-4 και μετα να βελτιώνεται με ένα μικρότερο βαθμό.

12. Συμπεράσματα Α μέρους

Σχετικά με την L1 Cache

Θα επιχειρήσουμε να χωρίσουμε τα benchmarks σε 4 κατηγορίες ανάλογα με τις παραμέτρους της L1-cache που επηρεάζουν την απόδοση τους:

1. Benchmarks τα οποία επηρεάζονται από το μέγεθος της cache, τα οποία είναι τα εξής:
 - a. Bodytrack
 - b. Freqmine
 - c. Swaptions
2. Benchmarks τα οποία επηρεάζονται από το βαθμό της συσχετιστικότητας, τα οποία είναι τα εξής:
 - a. Blackscholes
3. Benchmarks τα οποία επηρεάζονται από το μέγεθος του block size, τα οποία είναι τα εξής:
 - a. Blackscholes
 - b. Bodytrack
 - c. Canneal
 - d. Facesim
 - e. Ferret
 - f. Fluidanimate
 - g. Freqmine
 - h. Streamcluster
 - i. Swaptions
4. Benchmarks η απόδοση των οποίων παραμένει σταθερή και ανεπηρέαστη από τις παραμέτρους της L1-cache:
 - a. Rtview

Συνοψίζοντας, τα 9 από τα 10 μετροπρογράμματα επηρεάζονται από το cache block size. Επομένως, το συμπέρασμα που προκύπτει είναι πως ο καθοριστικότερος παράγοντας για την πλειοψηφία των μετροπρογραμμάτων είναι το μέγεθος του block. Βέβαια, το συμπέρασμα είναι αρκετά επισφαλές γιατί χρησιμοποιήθηκε δείγμα λίγων μετροπρογραμμάτων. Επίσης πολλά προγράμματα εχουν παρόμοιες επιδόσεις σε σχέση με τα διάφορα inputs.

Σχετικά με την L2 Cache

Θα επιχειρήσουμε να χωρίσουμε τα benchmarks σε 4 κατηγορίες ανάλογα με τις παραμέτρους της L2-cache που επηρεάζουν την απόδοση τους:

1. Benchmarks τα οποία επηρεάζονται από το μέγεθος της cache, τα οποία είναι τα εξής:
 - a. Blackscholes
 - b. Bodytrack
 - c. Canneal
 - d. Ferret
 - e. Fluidanimate
 - f. Freqmine
2. Benchmarks τα οποία επηρεάζονται από το βαθμό της συσχετιστικότητας, τα οποία είναι τα εξής:
 - a. Bodytrack
 - b. Canneal
3. Benchmarks τα οποία επηρεάζονται από το μέγεθος του block size, τα οποία είναι τα εξής:
 - a. Blackscholes
 - b. Bodytrack
 - c. Canneal
 - d. Facesim
 - e. Ferret
 - f. Fluidanimate
 - g. Freqmine
 - h. rtview
 - i. Streamcluster

Και στην περίπτωση της L2-cache παρατηρούμε πως τα περισσότερα benchmarks δίνουν βελτίωση στην απόδοση τους ανάλογα με το block size. Εδώ παρατηρούμε ότι επηρεάζει και το cache size. Επομένως μπορούμε να πούμε ότι καθοριστικά μεγέθοι είναι και το block size και το cache size, οι αύξηση των οποίων δεν οδηγεί πάντα στην βέλτιστη επιλογή καθώς καμία φορά η μειωση οδηγεί στην αύξηση του ipc. Επίσης πολλά προγράμματα εχουν παρόμοιες επιδόσεις σε σχέση με τα διάφορα inputs

Σχετικά με την L2 Cache

Στην περίπτωση της TLB μας δίνεται συγκεκριμένο μέγεθος σελίδας και επομένως ελέγχουμε τις υπόλοιπες δύο παραμέτρους, το TLB size και την Associativity.

Μπορούμε να εντάξουμε τα benchmarks στις εξής κατηγορίες:

1. Benchmarks τα οποία επηρεάζονται κυρίως από το tlb size:
 - a. **Blackscholes**
 - b. **Bodytrack**
 - c. **Canneal**
 - d. **Facesim**
 - e. **Ferret**
 - f. **Fluidanimate**
 - g. **Freqmine**
 - h. **rtview**
 - i. **Streamcluster**
 - j. **Swaptions**
2. Benchmarks τα οποία επηρεάζονται κυρίως από το assoc:
 - a. **Blackscholes**
 - b. **Bodytrack**
 - c. **Canneal**
 - d. **Facesim**
 - e. **Ferret**
 - f. **Fluidanimate**
 - g. **Freqmine**
 - h. **rtview**
 - i. **Streamcluster**
 - j. **Swaptions**
3. Benchmarks τα οποία παρουσιάζουν “ταβάνι” επίδοσης για συγκεκριμένο συνδυασμό και δε βελτιώνονται περαιτέρω:
 - a. **Facesim**
 - b. **Ferret**
 - c. **Fluidanimate**
 - d. **Freqmine**
 - e. **rtview**
 - f. **Streamcluster**
 - g. **Swaptions**

Γενικά παρατηρούμε ότι έχουμε συμμετοχή και των δυο παραμέτρων σε όλα τα benchmarks. Το χαμηλό assoc δίνει καταστροφικά αποτελέσματα στο iρc ωστόσο γενικά για τιμή μεγαλύτερη του 4-8 δεν μεταβάλλει ουσιαστικά το iρc.

Παρατηρούμε ότι τα περισσότερα προγράμματα παρουσιάζουν ένα ταβάνι που δηλώνει πως όσο και αν ανεβαζουμε τις τιμές δεν υπάρχει κάποια σημαντική βετλίωση. Αυτό το συμπέρασμα μας δείχνει ότι γενικά δεν χρειαζόμαστε τεράστια tlb σε size και assoc και ότι μπορούμε να αρκεστούμε σε πιο μικρά με το ίδιο καλή απόδοση. Αυτό το συμπέρασμα εξάγεται τουλάχιστον γι αυτα τα προγράμματα και αυτες τις τιμές L1,L2.

Σχετικά με το Prefetching

Παρατηρούμε εξετάζοντας τα πειραματικά αποτελέσματα πως η καλύτερη επιλογή prefetching κυμαίνεται μεταξύ των τιμών 64 και 32 και αμέσως μετά ακολουθεί η τιμή 16. Θα περίμενε κανείς ότι τα 64 θα έπρεπε να δίνουν το καλύτερο αποτέλεσμα ωστόσο δεν συμβαίνει για όλα τα benchmarks

13. Γενικά για το Β Μέρος

Στο προηγούμενο μέρος του εργαστηρίου ότι η αύξηση σε associativity και size δεν επηρεάζει τους κύκλους του ρολογιού, επομένως ένας χρήστης θα μπορούσε να αυξάνει διαρκώς τόσο το associativity όσο και το size χωρίς κάποια ποινή.

Αυτη η θεώρηση είναι μια υπεραπλούστευση που δεν συναντάται στην πράξη και έτσι στο β μέρος του εργαστηρίου θεωρούμε ότι τέτοιες αυξήσεις έχουν επιπτώσεις στους κύκλους του ρολογιού.

Γι' αυτό για κάθε αύξηση του associativity θεωρούμε ότι υπάρχει αύξηση 1.05 στους κύκλους του ρολογιού ενώ για κάθε αύξηση του size θεωρούμε ότι έχουμε μια αύξηση της τάξης του 1.1 στους κύκλους του ρολογιού. Το κλειδί είναι να συνειδητοποιήσουμε ότι δεν θα αυξηθεί το πλήθος των εντολών για κάθε προσομοίωση επομένως το νες iρc εξαρτάται μόνο από τους νέους κύκλους ρολογιού και γι αυτόν τον λόγο δεν τρέχουμε ξανά τις προσομοιώσεις αλλά κάνοντας τους κατάλληλους μαθηματικούς υπολογισμούς που περιγράψαμε προηγουμένως τροποποιούμε κατάλληλα το κάθε output file από τις προηγούμενες προσομοιώσεις.

Γενικά, το αποτέλεσμα θα είναι αντίστοιχο με το αποτέλεσμα της ιδανικής μελέτης, όπου δηλαδή δεν υπήρχαν ποινές για αύξηση associativity και size, αλλά με μια κατάλληλη προσαρμογή ποινών για κάθε μια από τις αντίστοιχες αυξήσεις. Οπότε συμβολικά θα μπορούσαμε να πούμε ότι ισχύει:

$$\text{πραγμτικό_αποτέλεσμα} = \text{ιδανικό_αποτέλεσμα} + \text{ποινές_αύξησης_size}$$

Επίσης, επειδή δεν αναφέρεται κάτι διαφορετικό θεωρούμε ότι δεν υπάρχει κάποιος επηρεασμός των mrki.

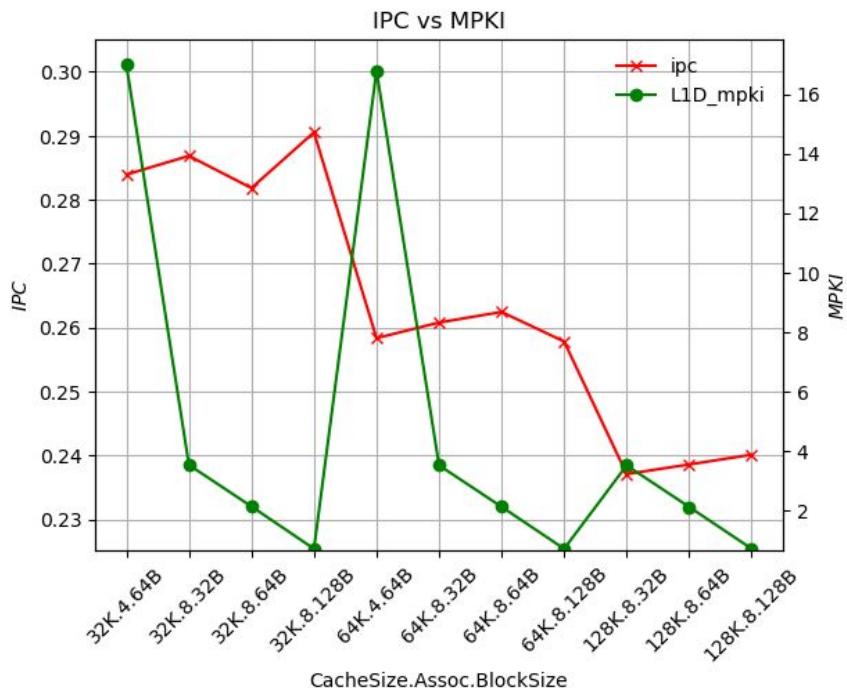
14. Έλεγχος Παραμέτρων L1-cache

Στο ερώτημα αυτό διατηρούμε τις παραμέτρους της L2-cache σταθερές και μεταβάλλουμε τις παραμέτρους της L1-cache, ομώς κάθε φορά που διπλάσιάζουμε το cache size από το αρχικό έχουμε μια αύξηση 1.1 στο πληθος των κύκλων και καθε φορά που αυξάνουμε το associativity από την αρχική τιμή έχουμε μια αύξηση 1.05 στο πλήθος των κύκλων. Οι τιμές των παραμέτρων της L2-cache και του TLB γι αυτό το ερώτημα είναι σταθερές και ίσες με:

- L2 size = 1024 KB
- L2 associativity = 8
- L2 block size = 128 B
- TLB size = 64 entr.
- TLB associativity = 4
- TLB page size = 4096 B

Παρακάτω προκύπτουν συγκριτικά τα αποτελέσματα των 10 benchmarks για διάφορους συνδυασμούς των μεγεθών της L1-cache:

Blackscholes



Στο αντίστοιχο προηγούμενο ιδανικό μοντέλο είχαμε δει ότι:

- Η απόδοση-ipc του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το assoc**.
- Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **παρουσιάζει μικρή αύξηση όσο αυξάνει το block size**.

Τώρα όμως η αύξηση του assoc και του cache size έρχονται με ένα κόστος σε κύκλους, επομένως εξ αρχής μπορούμε να πουμε ότι η αύξηση του cache size είναι άσκοπα δαπανηρη για τις δεδομένες 11 περιπτώσεις του L1 και δεν θα μας δώσει καλύτερα αποτελέσματα άρα περιμένουμε το καλύτερο αποτέλεσμα να βρίσκεται σε κάποιο input με cache size = 32K.

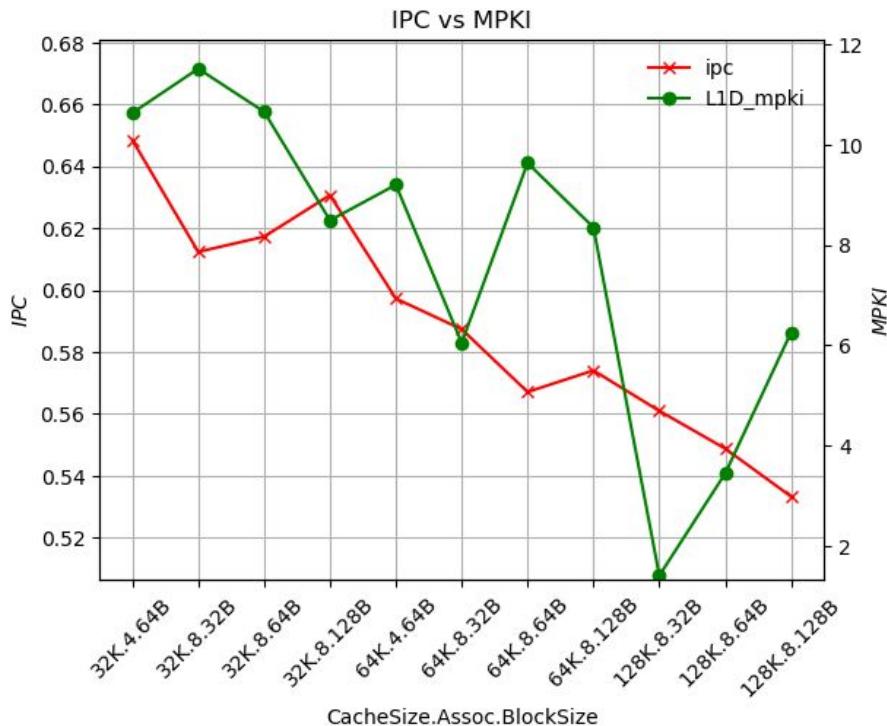
Παρατηρούμε ότι αυτό πετυχαίνεται με μεγαλύτερο assoc, μια τάξης μεγαλύτερο με ποινη 1.05 κυκλων και για το μέγιστο δυνατό block size. Άρα για το input 32K.8.128B

Προφανώς λόγω της ποινής το μέγιστο αυτό ipc είναι χαμηλότερο από το αντίστοιχο για την ιδανική περίπτωση και αυτό οφείλεται στο ότι έχουμε 5% περισσότερους κύκλους και ίδιο πλήθος εντολών.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό βέλτιστο συνδυασμό τιμών παραγόντων!**

Γενικά, βλέπουμε ότι η αύξηση του size δίνει χειρότερα αποτελέσματα επειδη προσφέρει ποινή σε κύκλους και δεν επιβοηθουσι ούτε το ιδανικό σύστημα άρα ούτε και αυτό και η αύξηση του block size δίνει μια μικρή βελτίωση στο ipc.

Bodytrack



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Όσο αυξάνεται το cache size βελτιώνεται τόσο το ipc όσο και το mpki. Επομένως η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Cache Size**.
- Όσον αφορά το Associativity: Παρατηρούμε ότι η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Για **μικρές τιμές του cache size**, δηλαδή για τις τιμές 32KB και 64KB η αύξηση του block size αυξάνει την επίδοση του συστήματος τοσο στο ipc οσο και στα misses. Ωστόσο, όταν η cache size γίνει μεγαλύτερη της τιμής 128 παρατηρούμε ότι όσο μεγαλώνει το block size τόσο χειρότερα τρέχει το σύστημα.

Επομένως, το πως επιδρά το **block size σχετίζεται άμεσα από το cache size**

Αυτό το διάγραμμα έχει ιδιαίτερο ενδιαφέρον γιατί το ipc αυξάνει με το cache size, όμως η αύξηση του cache size δίνει ποινή στο ipc.

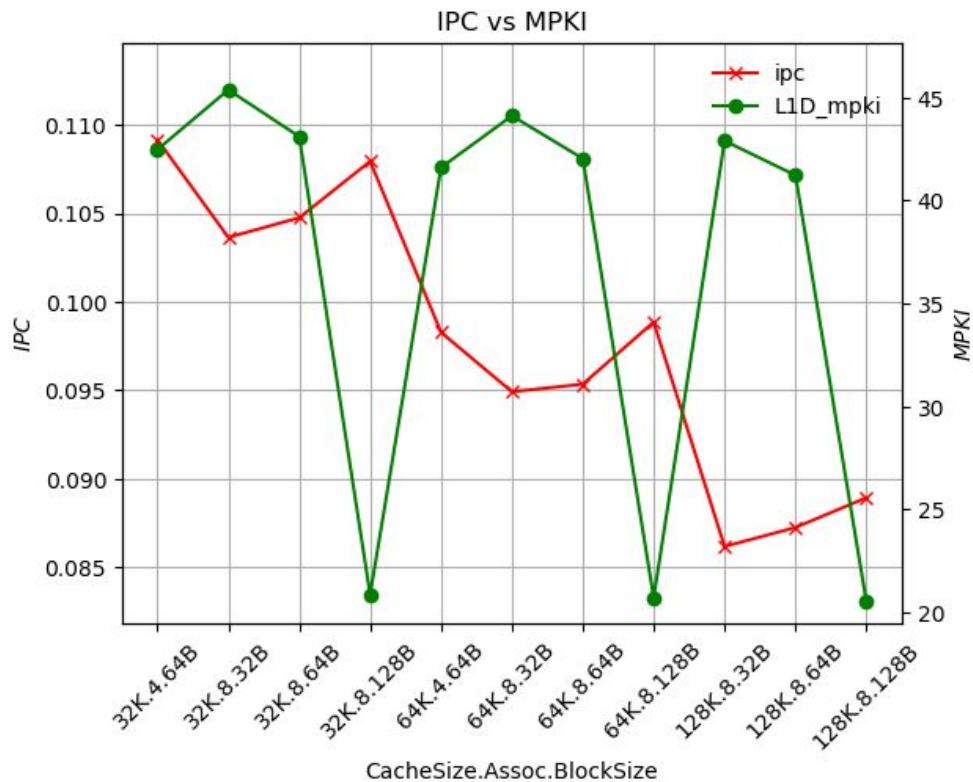
Επίσης για cache size μικρότερα ίσα των 64KB η αύξηση του block size , που δεν επιφέρει κάπιο κόστος στο συστήμα, βοηθάει το αποτέλεσμα. Για τιμές μεγαλύτερες των 128KB δεν χειροτερεύει το αποτέλεσμα.

Έτσι, παρατηρούμε ότι η ποινή του cache size δεν αφήνει να έχουμε βέλτιστη τιμή για μεγάλα cache size και έχουμε βέλτιστη για cache size 32KB, που δεν έχει καμία ποινή και για το μικρότερο assoc που και πάλι δεν έχει καμία ποινή.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων!

Γενικά, παρατηρούμε ότι η αύξηση του size δίνει χειρότερα αποτελέσματα(από την μία βοηθάει αλλά από την άλλη η ποινή της αυξησης είναι πολύ μεγάλη και συνολικά επιφέρει χειρότερο αποτέλεσμα), το assoc δίνει χειρότερο αποτέλεσμα επειδή ουτε πριν βοηθουσε και τώρα δίνει και ποινή ενώ το block size ακολουθεί το ίδιο μοτίβο με πριν.

Canneal



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

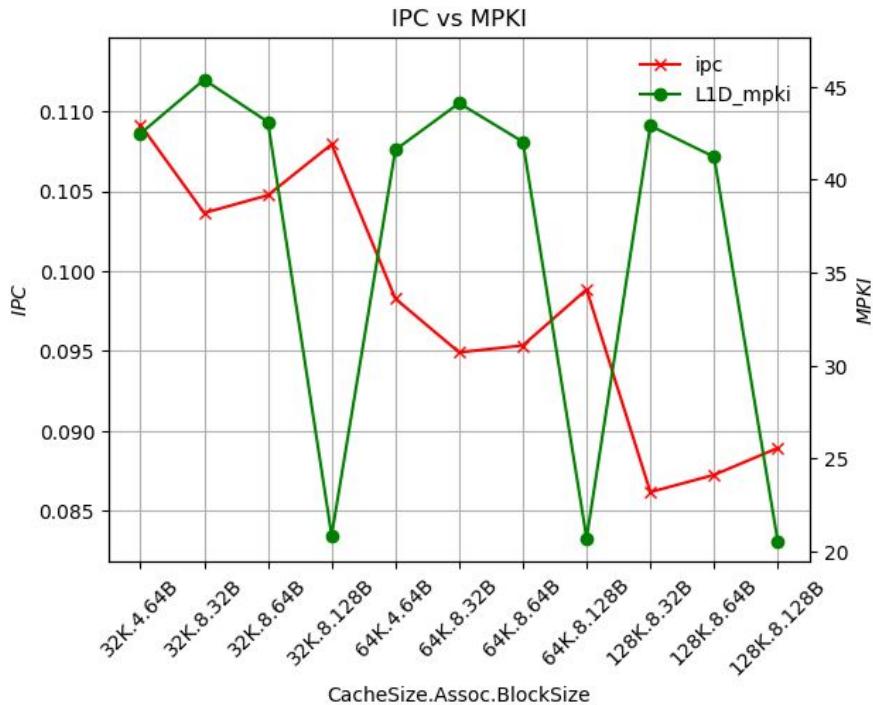
- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Οποιαδήποτε αύξηση size και assoc θα είναι άσκοπη και κοστοβόρα, επομένως περιμένουμε την καλύτερη τιμή για το default size και assoc με το μεγιστο δυνατό block size. Επειδή αυτά τα μεγέθη δεν δίνουν μεγάλο block size (φτάνουν μέχρι 64 από εκφωνηση) ίσως είναι challenging η τιμή για το αμέσως μεγαλύτερο assoc που φτάνει μεγαλύτερο block size 128, δηλαδή το 32K.8.128, αλλά όπως βλέπουμε από την προσομοίωση κερδίζουμε το 32K.4.64 (για λίγο).

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων!

Γενικά, παρατηρούμε ότι η αύξηση του size δίνει χειρότερα αποτελέσματα επειδή ουτε πριν βοηθουσε και τώρα δίνει και ποινή, το assoc δίνει χειρότερο αποτέλεσμα επειδή ουτε πριν βοηθουσε και τώρα δίνει και ποινή ενώ το block size όσο μεγαλώνει τόσο καλύτερα ipc δίνει!

Facesim



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

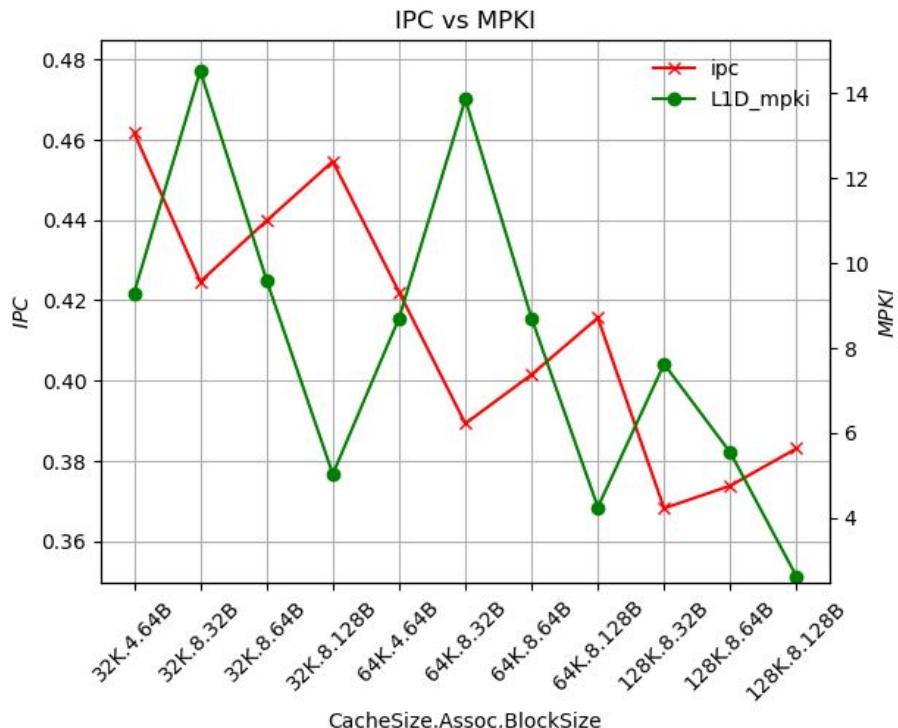
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Επειδή στο ιδανικό παράδειγμα δεν επιδρούσε θετικά τώρα με την προσθήκη ποινης με την αύξηση είναι προφανές ότι θα **χειροτερεύει**
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Για αντίστοιχους λόγους
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Οποιαδήποτε αύξηση size και assoc θα είναι άσκοπη και κοστοβόρα, επομένως περιμένουμε την καλύτερη τιμή για το default size και assoc με το μεγιστο δυνατό block size. Ίσως είναι challenging η τιμή για το αμέσως μεγαλύτερο assoc που φτάνει μεγαλύτερο block size 128, δηλαδή το 32K.8.128, αλλά όπως βλέπουμε από την προσομοίωση κερδίζουμε το 32K.4.64 (για λίγο).

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων!

Ferret



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

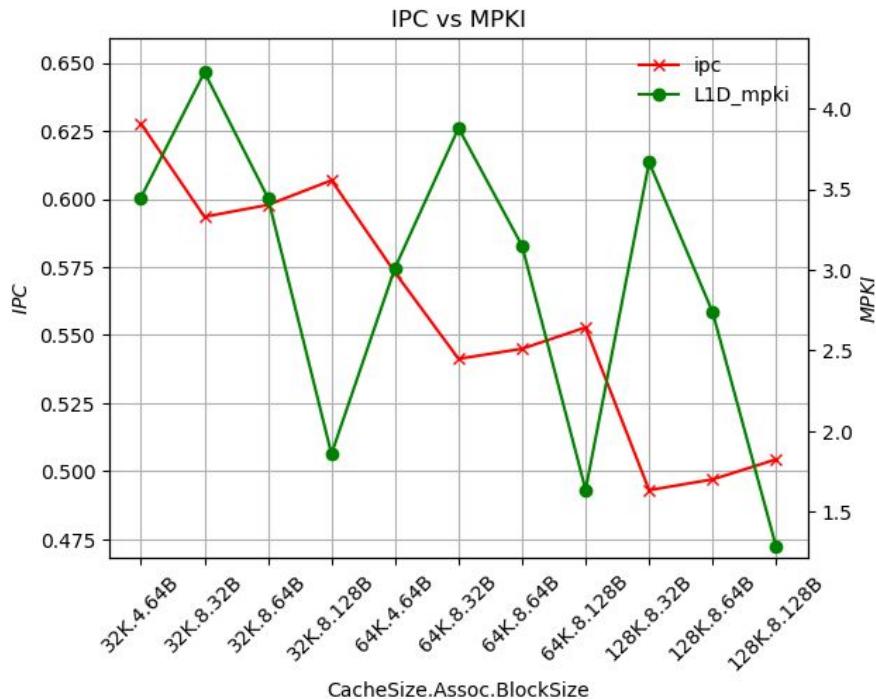
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Επειδή στο ιδανικό παράδειγμα δεν επιδρούσε θετικά τώρα με την προσθήκη ποινης με την αύξηση είναι προφανές ότι θα χειροτερεύει
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Για αντίστοιχους λόγους
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Οποιαδήποτε αύξηση size και assoc θα είναι άσκοπη και κοστοβόρα, επομένως περιμένουμε την καλύτερη τιμή για το default size και assoc με το μεγιστο δυνατό block size. Ίσως είναι challenging η τιμή για το αμέσως μεγαλύτερο assoc που φτάνει μεγαλύτερο block size 128, δηλαδή το 32K.8.128, αλλά όπως βλέπουμε από την προσομοίωση κερδίζουμε το 32K.4.64.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων!

Fluidanimate



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

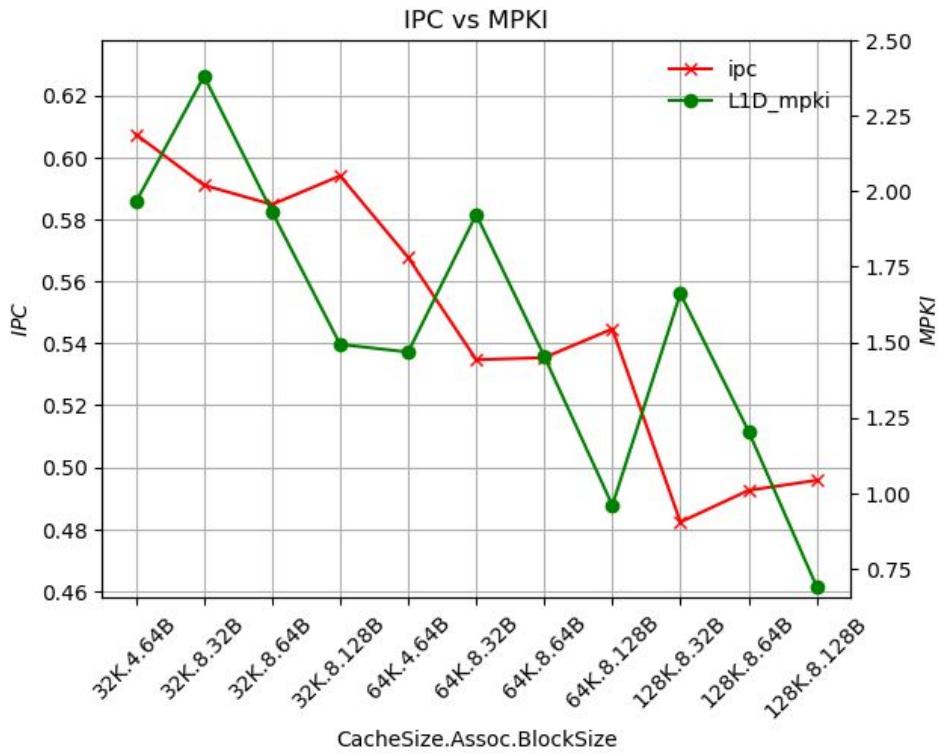
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Επειδή στο ιδανικό παράδειγμα δεν επιδρούσε θετικά τώρα με την προσθήκη ποινης με την αύξηση είναι προφανές ότι θα χειροτερεύει
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Για αντίστοιχους λόγους
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Οποιαδήποτε αύξηση size και assoc θα είναι άσκοπη και κοστοβόρα, επομένως περιμένουμε την καλύτερη τιμή για το default size και assoc με το μεγιστο δυνατό block size. Ίσως είναι challenging η τιμή για το αμέσως μεγαλύτερο assoc που φτάνει μεγαλύτερο block size 128, δηλαδή το 32K.8.128, αλλά όπως βλέπουμε από την προσομοίωση κερδίζουμε το 32K.4.64 (για λίγο).

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων!

Freqmine



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **επιδρά** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η μεταβολή του block size **επιδρά** στην επίδοση του προγράμματος για το συγκεκριμένο input.

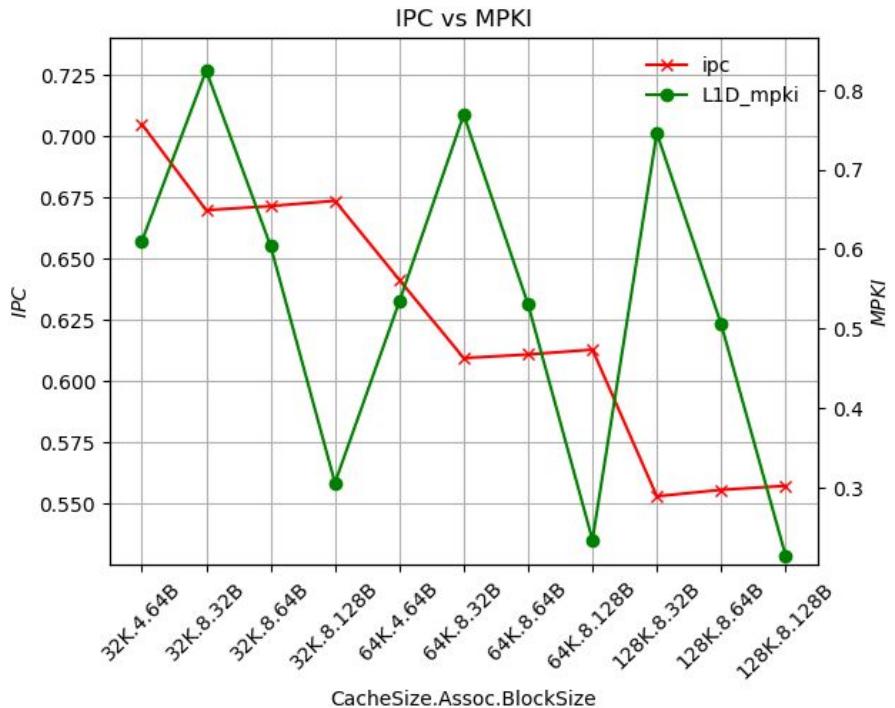
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Επειδή στο ιδανικό παράδειγμα δεν επιδρούσε θετικά τώρα με την προσθήκη ποινης με την αύξηση είναι προφανές ότι θα χειροτερεύει
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Για αντίστοιχους λόγους
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Οποιαδήποτε αύξηση size και assoc θα είναι άσκοπη και κοστοβόρα, επομένως περιμένουμε την καλύτερη τιμή για το default size και assoc με το μεγιστο δυνατό block size. Ισως είναι challenging η τιμή για το αμέσως μεγαλύτερο assoc που φτάνει μεγαλύτερο block size 128, δηλαδή το 32K.8.128, αλλά όπως βλέπουμε από την προσομοίωση κερδίζουμε το 32K.4.64 (για λίγο).

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων!

rtview



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

Το ipc παρατηρούμε ότι παραμένει σταθερός και ανεξαρτητως των παραμετρων.

Άρα περιμένουμε ότι οι αυξήσεις size και assoc θα χειροτερεύουν την απόδοση ενώ η τιμή του block size δεν θα επιδρά στην επίδοση σχεδόν καθόλου.

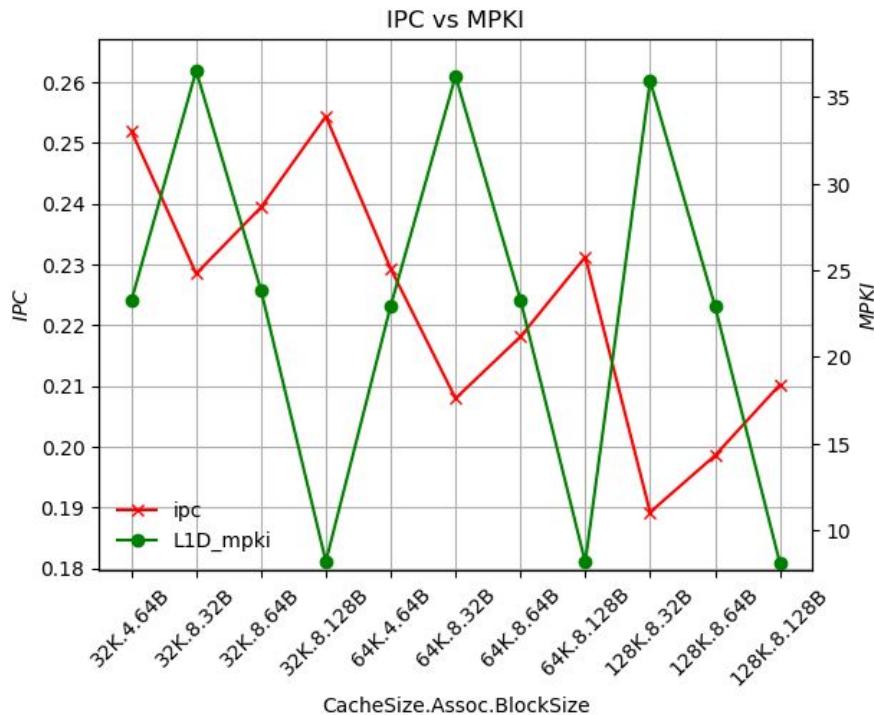
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Επειδή στο ιδανικό παράδειγμα δεν επιδρούν θετικά τώρα με την προσθήκη ποινης με την αύξηση είναι προφανές ότι θα χειροτερεύει
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Για αντίστοιχους λόγους
- Όσον αφορά το Block Size: Η μεταβολή του block size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input. Ίσως να βοηθάει στα όρια του επιστητου το ipc.

Επομένως, οποιαδήποτε αύξηση του size, assoc θα χειροτερέψει το ipc ενώ το block size δεν θα επηρεάσει, άρα είναι αναμενόμενο το καλύτερο ipc να έχει η πρώτη τιμή και πράγματι αυτό ισχύει.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων!

Streamcluster



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

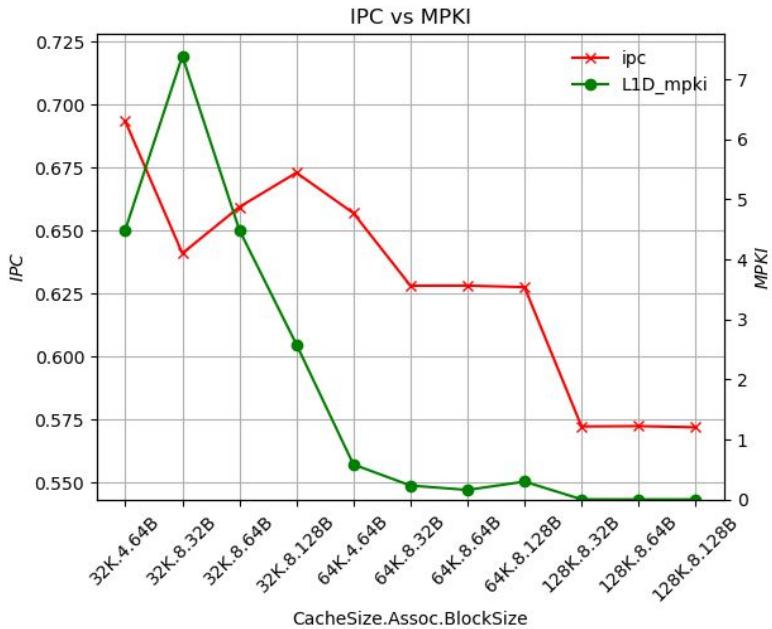
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Επειδή στο ιδανικό παράδειγμα δεν επιδρούσε θετικά τώρα με την προσθήκη ποινης με την αύξηση είναι προφανές ότι θα χειροτερεύει
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Για αντίστοιχους λόγους
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Οποιαδήποτε αύξηση size και assoc θα είναι άσκοπη και κοστοβόρα, επομένως περιμένουμε την καλύτερη τιμή για το default size και assoc με το μεγιστο δυνατό block size. Ίσως είναι challenging η τιμή για το αμέσως μεγαλύτερο assoc που φτάνει μεγαλύτερο block size 128, δηλαδή το 32K.8.128, αλλά όπως βλέπουμε από την προσομοίωση κερδίζουμε το 32K.8.128 (για λίγο)!

Άρα βλέπουμε ότι το ιδανικό και το πραγματικό μοντέλο **συμφωνούν ότι ο βέλτιστος** συνδυασμός είναι ο 32K.8.128!

Swaptions



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **επιρεάζει θετικά** το ipc με threshold την τιμή των 64KB μετά από την οποία **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size** με threshold την τιμή του cache size των 64KB μετά από την οποία **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Επειδή στο ιδανικό παράδειγμα δεν επιδρούσε θετικά τώρα με την προσθήκη ποινης με την αύξηση είναι προφανές ότι θα **χειροτερεύει**
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Για αντίστοιχους λόγους
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size** με threshold την τιμή του cache size των 64KB μετά από την οποία **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.

Οποιαδήποτε αύξηση size και assoc θα είναι άσκοπη και κοστοβόρα, επομένως περιμένουμε την καλύτερη τιμή για το default size και assoc με το μεγιστο δυνατό block size. Ίσως είναι challenging η τιμή για το αμέσως μεγαλύτερο assoc που φτάνει μεγαλύτερο block size 128, δηλαδή το 32K.8.128, αλλά όπως βλέπουμε από την προσομοίωση κερδίζουμε το 32K.4.64 (για λίγο).

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων!

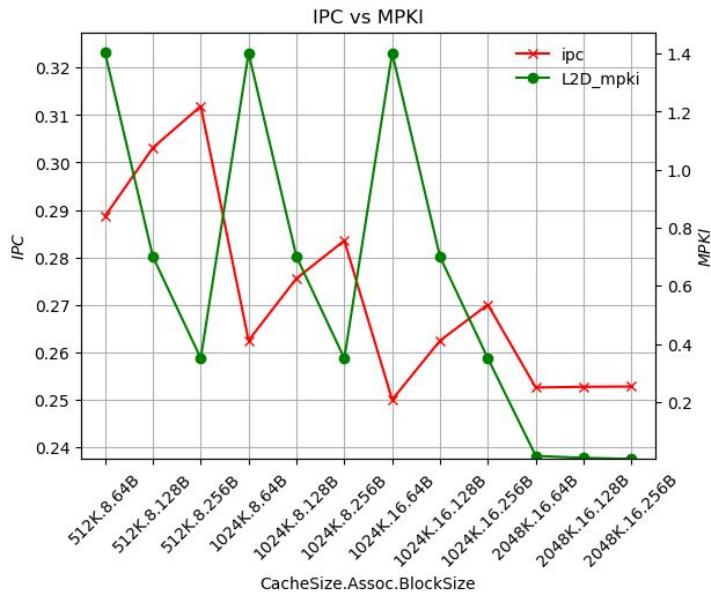
15. Έλεγχος Παραμέτρων L2-cache

Στο ερώτημα αυτό διατηρούμε τις παραμέτρους της L1-cache σταθερές και μεταβάλλουμε τις παραμέτρους της L2-cache, ομώς κάθε φορά που διπλάσιαζουμε το cache size από το αρχικό έχουμε μια αύξηση 1.1 στο πληθος των κύκλων και καθε φορά που αυξάνουμε το associativity από την αρχική τιμή έχουμε μια αύξηση 1.05 στο πλήθος των κύκλων. Οι τιμές των παραμέτρων της L1-cache και του TLB γι αυτό το ερώτημα είναι σταθερές και ίσες με:

- L1 size = 32 KB
- L1 associativity = 8
- L1 block size = 64 B
- TLB size = 64 entr.
- TLB associativity = 4
- TLB page size = 4096 B

Παρακάτω προκύπτουν συγκριτικά τα αποτελέσματα των 10 benchmarks για διάφορους συνδυασμούς των μεγεθών της L2-cache:

Blackscholes



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επηρεάζει αισθητά** την επίδοση του προγράμματος μέχρι το threshold του cache size=2048 όπου επηρεάζει καθοριστικά και δίνει το βέλτιστο ipc.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size** με threshold την τιμή του cache size των 2048KB μετά από την οποία **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.

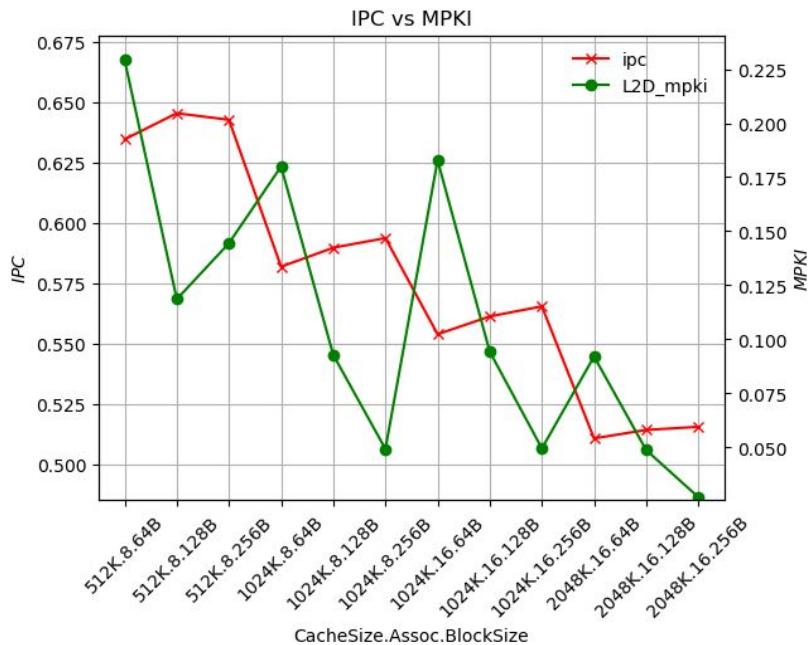
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input ακόμη και για τιμές cache size 2048kb επειδή η ποινή είναι αρκετά αυστηρή. Για τις τιμές μικρότερες του threshold είναι προφανές ότι επιδρά αρνητικά επειδή στο ιδανικό παράδειγμα δεν επιδρούν αισθητά τώρα με την προσθήκη ποινης με την αύξηση είναι προφανές ότι θα χειροτερεύει
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Είναι προφανές ότι επιδρά αρνητικά επειδή στο ιδανικό παράδειγμα δεν επιδρούν αισθητά τώρα με την προσθήκη ποινης με την αύξηση είναι προφανές ότι θα χειροτερεύει
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size** μέχρι το threshold του cache size των 2048KB μετά από την οποία **δεν επιδρά σχεδόν καθόλου** στην επίδοση του προγράμματος για το συγκεκριμένο input.

Δεδομένου ότι το block size βοηθά για μικρές τιμές cache size αναμένουμε από την άνω ανάλυση ότι η βέλτιστη τιμή ipc θα είναι για cache με 512KB.8.256B, πράγμα που επαληθεύεται από το διάγραμμα.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων της cache!

Bodytrack



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

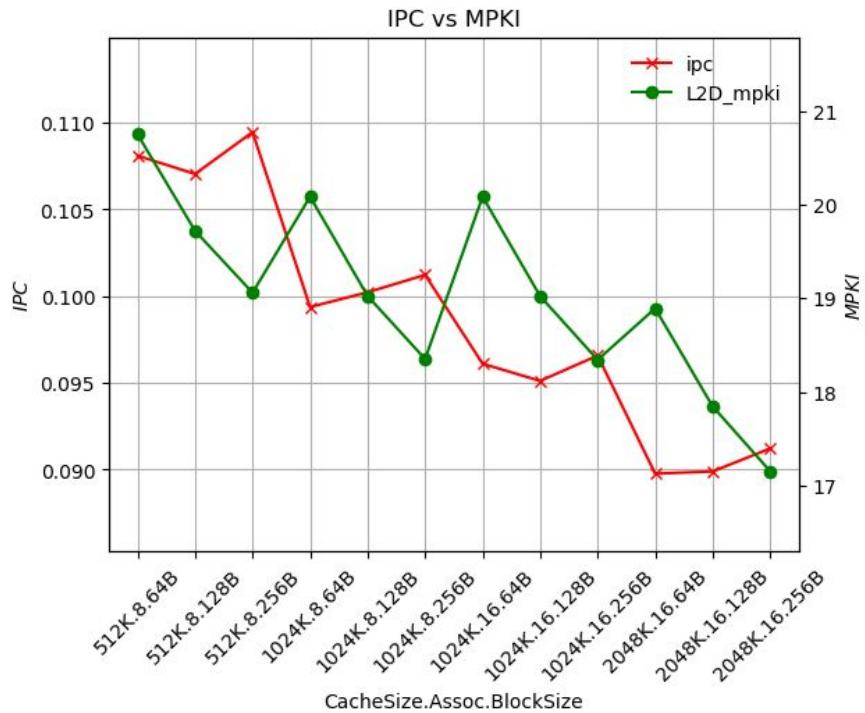
- Όσον αφορά το Cache Size: Η μεταβολή του cache size **επιδρά θετικά** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **επιδρά αρνητικά** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size** για τιμές cache size μεγαλύτερες του 1024KB. Για το cache size=512KB βλέπουμε ότι η πρώτη αυξηση του βοηθά και μετά η δεύτερη δρα αρνητικά.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input. Είναι προφανές ότι επιδρά αρνητικά επειδή στο ιδανικό παράδειγμα επιδρά αρνητικά τώρα με την προσθήκη ποινης με την αύξηση είναι προφανές ότι θα επιδρά ακόμη πιο αρνητικά.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size** για τιμές cache size μεγαλύτερες του 1024KB. Για το cache size=512KB βλέπουμε ότι η πρώτη αυξηση του βοηθά και μετά η δεύτερη δρα αρνητικά.

Δεδομένου ότι το block size για το cache size=512KB βλέπουμε ότι η πρώτη αυξηση του βοηθά και μετά η δεύτερη δρα αρνητικά αναμένουμε από την άνω ανάλυση ότι η βέλτιστη τιμή ipc θα είναι για cache με 512KB.8.128B, πράγμα που επαληθεύεται από το διάγραμμα. Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων της cache!

Canneal



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **επιδρά θετικά** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **επιδρά θετικά** στην επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input αλλάζει διαρκώς. Για μία τιμή μεγαλύτερη, δηλαδή για 128, δρα αρνητικά και για μια ακόμη, δηλαδή για 256, δρά θετικά.

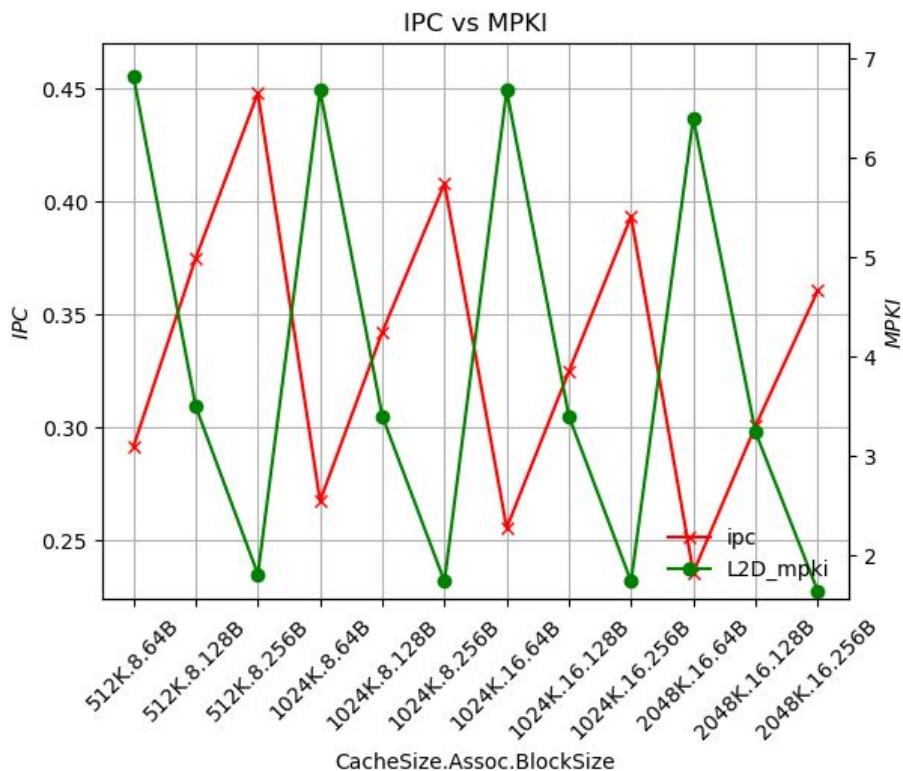
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input αλλάζει διαρκώς. Για μία τιμή μεγαλύτερη, δηλαδή για 128, δρα αρνητικά και για μια ακόμη, δηλαδή για 256, δρά θετικά.

Δεδομένου του συμπεριφοράς του block size με βάση την άνω ανάλυση ότι η βέλτιστη τιμή ipc θα είναι για cache με 512KB.8.256B, πράγμα που επαληθεύεται από το διάγραμμα.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων της cache!

Facesim



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

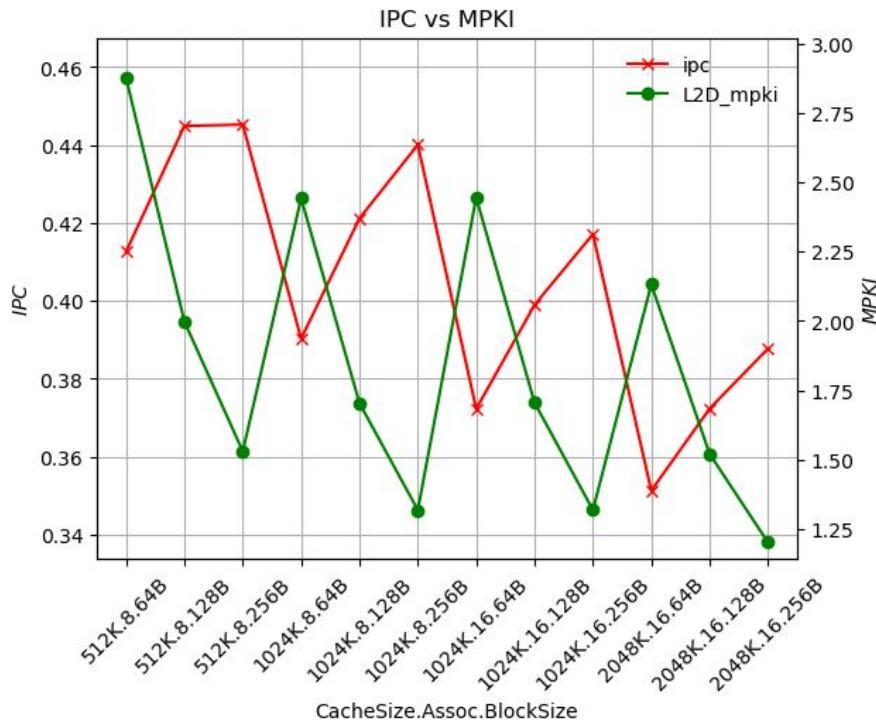
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Δεδομένου του συμπεριφοράς του block size με βάση την άνω ανάλυση ότι η βέλτιστη τιμή ipc θα είναι για cache με 512KB.8.256B, πράγμα που επαληθεύεται από το διάγραμμα.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **έναν ίδιο** βέλτιστο συνδυασμό τιμών παραγόντων της cache!

Ferret



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορα το Cache Size: Η αύξηση του cache size **επηρεάζει θετικά** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**. Εξαίρεση αποτελεί για το cache size 512KB η αύξηση από 128B σε 256B.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

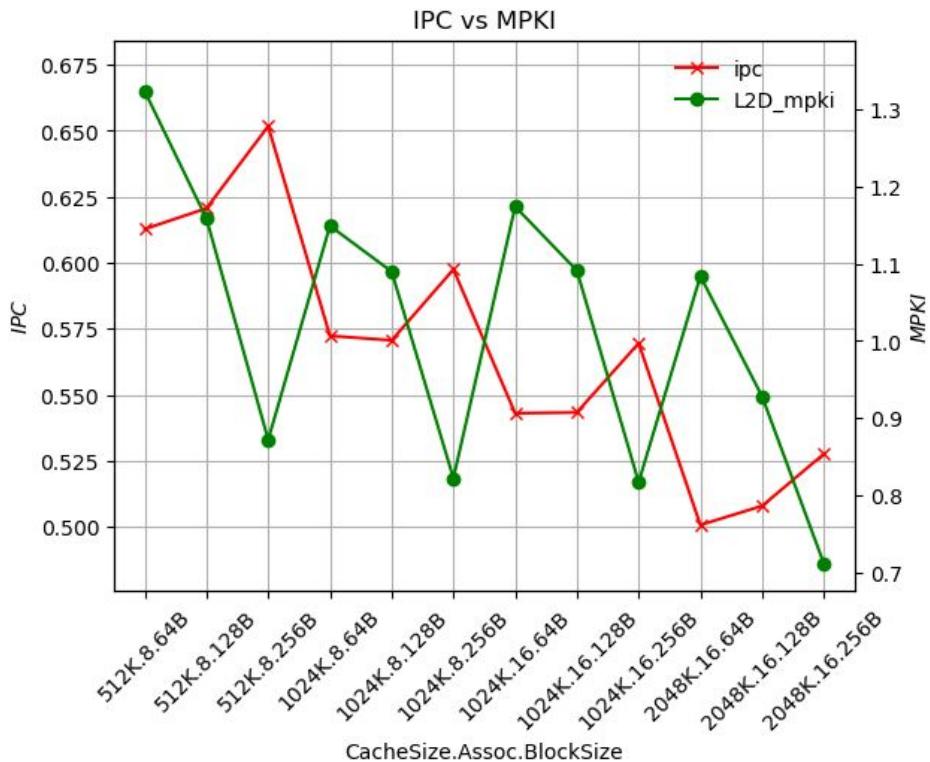
- Όσον αφορα το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή, παρόλο την θετική επίδραση που δίνει στο ιδανικό σύστημα η αύξηση της cache size.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**. Εξαίρεση αποτελεί για το cache size 512KB η αύξηση από 128B σε 256B.

Δεδομένου του συμπεριφοράς του block size με βάση την άνω ανάλυση ότι η βέλτιστη τιμή ipc θα είναι για cache με 512KB.8.256B, ή της τιμής 1024KB.8.256, επειδή η πρώτη τιμή δεν έχει την αναμενόμενη αύξηση που περιγραφεί το block size.

Βλέπουμε ότι η βέλτιστη απόδοση περιγράφεται από το 512KB.8.256B.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων της cache!

Fluidanimate



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η αύξηση του cache size **επηρεάζει θετικά** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

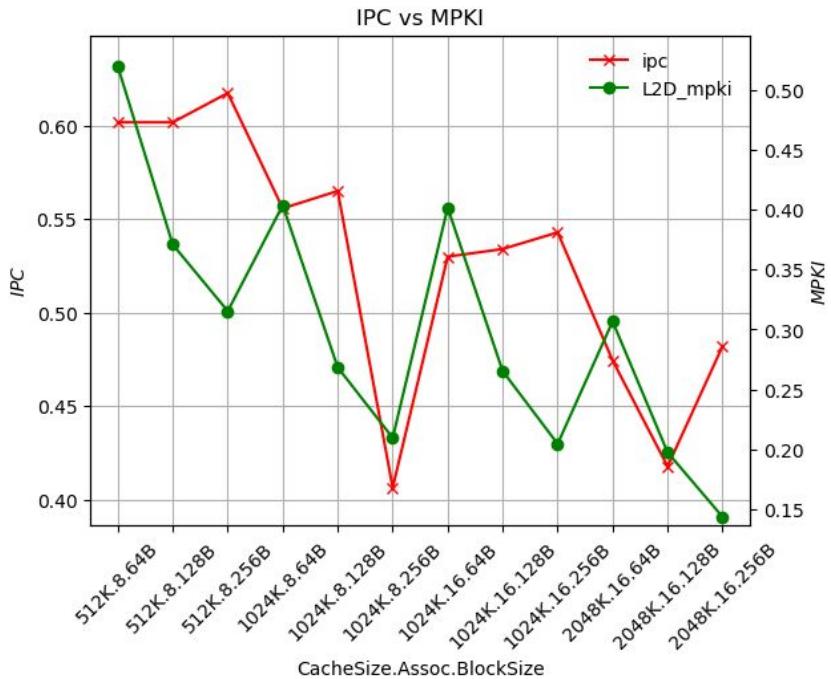
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή, παρόλο την θετική επίδραση που δίνει στο ιδανικό σύστημα η αύξηση της cache size.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Δεδομένου του συμπεριφοράς του block size με βάση την άνω ανάλυση ότι η βέλτιστη τιμή ipc θα είναι για cache με 512KB.8.256B, πράγμα που επαληθεύεται από το διάγραμμα.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **έναν ίδιο** βέλτιστο συνδυασμό τιμών παραγόντων της cache!

Freqmine



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η αύξηση του cache size **επηρεάζει θετικά** για την μεταβολή από 512 σε 1024 και **αρνητικά** για την μεταβολή 1024 σε 2048 την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**. Εξαίρεση αποτελούν οι καταγραφές 1024K.8.256 και 2048K.16.128B.

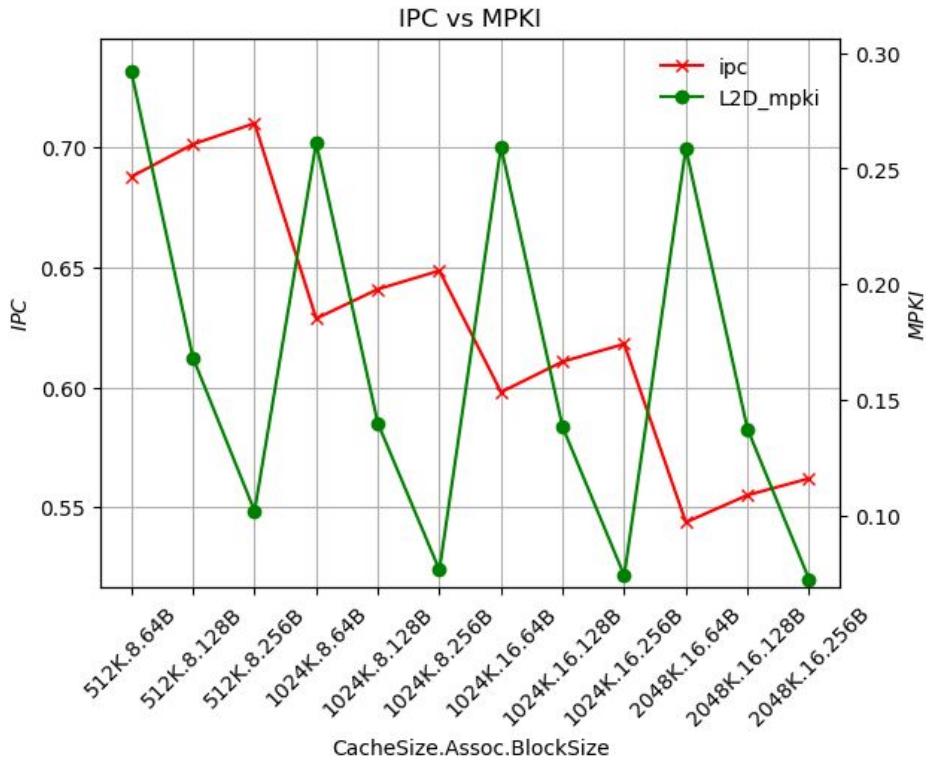
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή, παρόλο την θετική επίδραση στο κομμάτι 512-1024 που δίνει στο ιδανικό σύστημα η αύξηση της cache size.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**. Εξαίρεση αποτελούν οι καταγραφές 1024K.8.256 και 2048K.16.128B.

Δεδομένου του συμπεριφοράς του block size με βάση την άνω ανάλυση ότι η βέλτιστη τιμή ipc θα είναι για cache με 512KB.8.256B, πράγμα που επαληθεύεται από το διάγραμμα.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **διαφορετικό** βέλτιστο συνδυασμό τιμών παραγόντων της cache!

rtview



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **δεν επηρεάζει** την επίδοση του προγράμματος για το συγκεκριμένο input.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

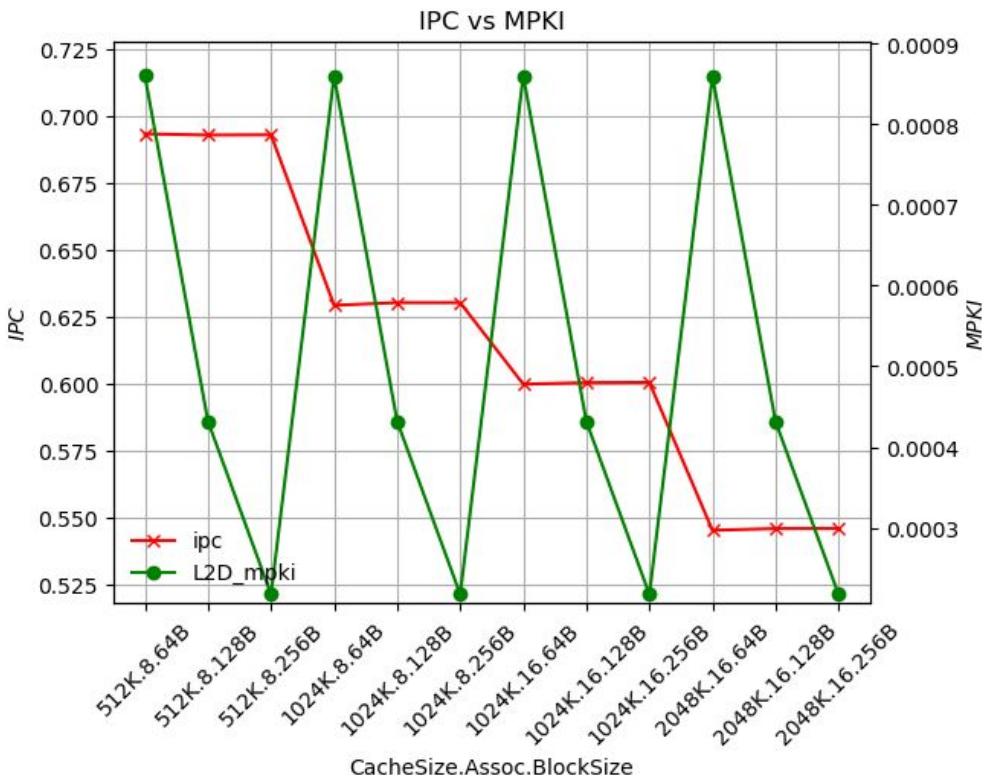
Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **αυξάνεται όσο αυξάνεται το Block Size**.

Δεδομένου του συμπτεριφοράς του block size με βάση την άνω ανάλυση ότι η βέλτιστη τιμή ipc θα είναι για cache με 512KB.8.256B, πράγμα που επαληθεύεται από το διάγραμμα.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **έναν ίδιο** βέλτιστο συνδυασμό τιμών παραγόντων της cache!

Swaptions



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

Ο δείκτης ipc παραμένει σταθερός και ανεξάρτητος από τους παραγοντες cache size, assoc, block size.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το Cache Size: Η μεταβολή του cache size **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Associativity: Η μεταβολή του assoc **χειροτερεύει** την επίδοση του προγράμματος για το συγκεκριμένο input επειδή η ποινή είναι αρκετά αυστηρή.
- Όσον αφορά το Block Size: Η απόδοση του συγκεκριμένου benchmark για το συγκεκριμένο input **παραμένει πρακτικά ανεξάρτητη** για την μεταβολή του block size.

Δεδομένου του συμπεριφοράς του block size με βάση την άνω ανάλυση ότι η βέλτιστη τιμή ipc θα είναι για cache με 512KB.8.64B, που θα είναι περίπου ίση με 512KB.8.128B, που θα είναι περίπου ίση με 512KB.8.256B πράγμα που επαληθεύεται από το διάγραμμα.

Άρα, βλέπουμε ότι εδώ το ιδανικό μοντέλο και το πιο πραγματικό δίνουν **έναν ίδιο** βέλτιστο συνδυασμό τιμών παραγόντων της cache!

16. Έλεγχος Παραμέτρων TLB

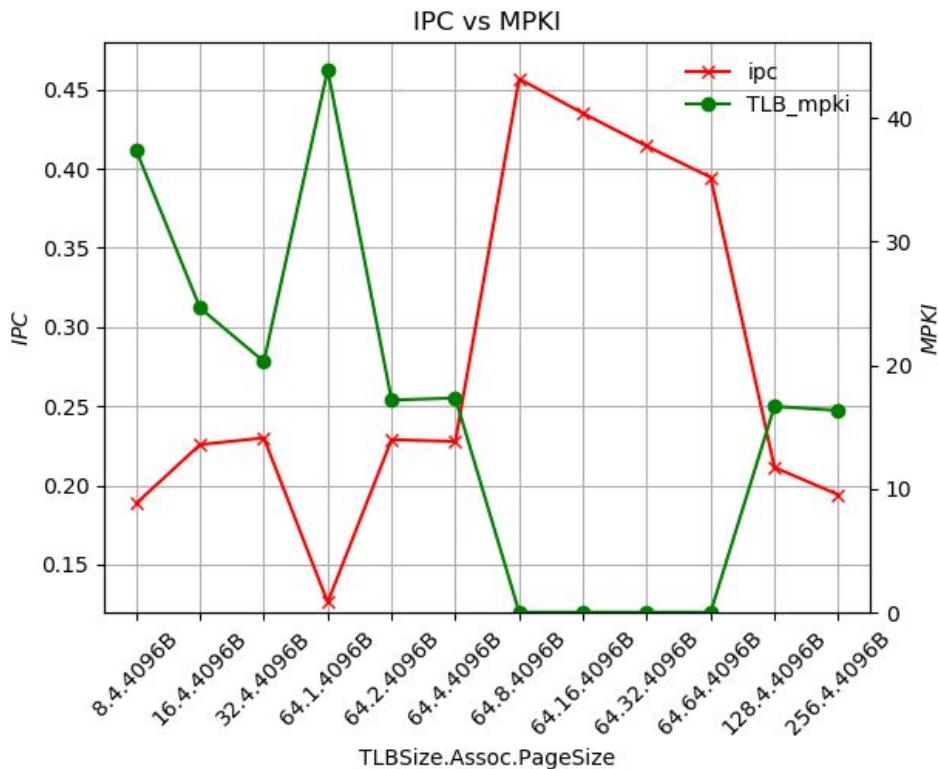
Στο ερώτημα αυτό διατηρούμε τις παραμέτρους των L1-cache και L2-cache σταθερές. Επίσης διατηρείται σταθερό και το μέγεθος σελίδας του TLB και ίσο με 4096 Bytes, ομώς κάθε φορά που διπλάσιαζουμε το cache size από το αρχικό έχουμε μια αύξηση 1.1 στο πληθος των κύκλων και καθε φορά που αυξάνουμε το associativity από την αρχική τιμή έχουμε μια αύξηση 1.05 στο πλήθος των κύκλων. Οι τιμές των L1-cache και L2-cache για αυτή τη μελέτη είναι σταθερές και ίσες με:

- L1 size = 32 KB
- L1 block size = 64 B
- L1 associativity = 8
- L2 size = 1024 KB
- L2 associativity = 8
- L2 block size = 128 B

Παρακάτω προκύπτουν συγκριτικά τα αποτελέσματα των 10 benchmarks για διάφορους συνδυασμούς των μεγεθών της TLB.

Προσοχή: Επειδή η άσκηση δεν έδινε κάποια διευκρίνιση για το τι να κάνουμε αν έχουμε πτώση του assoc από την αρχική τιμή, θεωρώ ότι οι κυκλοί δεν επηρεάζονται.

Blackscholes



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

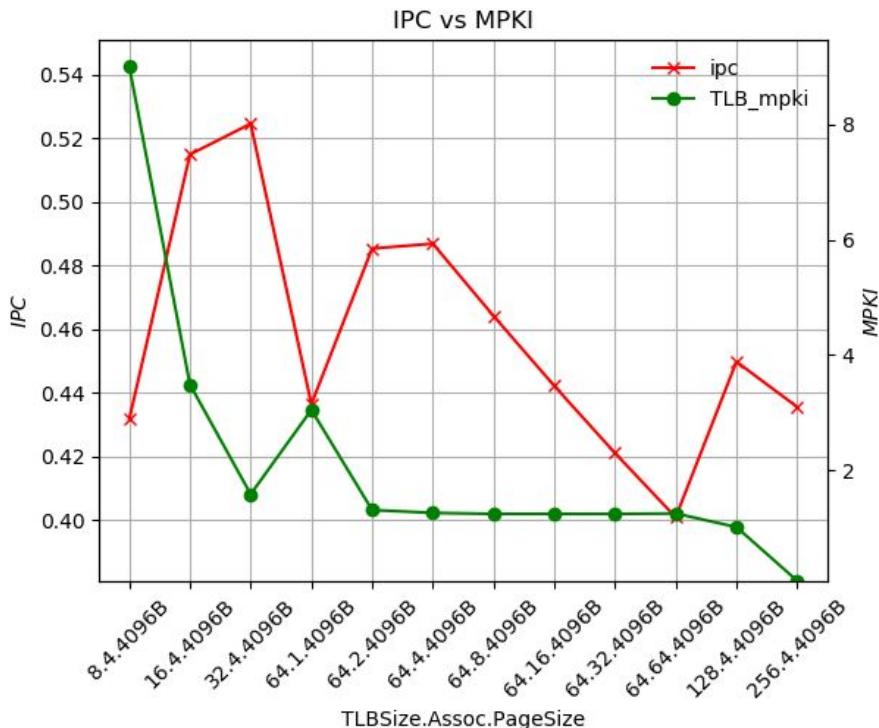
- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 32 entr. οπου μετά το οποίο δεν επηρεάζει σχεδόν καθόλου
- Όσον αφορά το Associativity: Για αύξηση του assoc από 1-2 έχουμε βελτίωση της επίδοσης, για αύξηση από 2-4 δεν παρατηρούμε κάποια βελτίωση, για αύξηση από 4-8 παρατηρούμε μια πολύ μεγάλη βελτίωση και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει την επίδοση.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 32 entr. οπου μετά το οποίο χειροτερεύει την επίδοση.
- Όσον αφορά το Associativity: Για αύξηση του assoc από 1-2 έχουμε βελτίωση της επίδοσης, για αύξηση από 2-4 δεν παρατηρούμε κάποια βελτίωση, για αύξηση από 4-8 παρατηρούμε μια πολύ μεγάλη βελτίωση και από εκεί και πέρα οποιαδήποτε αύξηση του assoc χειροτερεύει την επίδοση την επίδοση.

Παρατηρούμε ότι η βέλτιστη απόδοση δίνεται για 64.8.4096B που **ταυτίζεται** με την βέλτιστη απόδοση που δίνει το ιδανικό μοντέλο.

Bodytrack



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

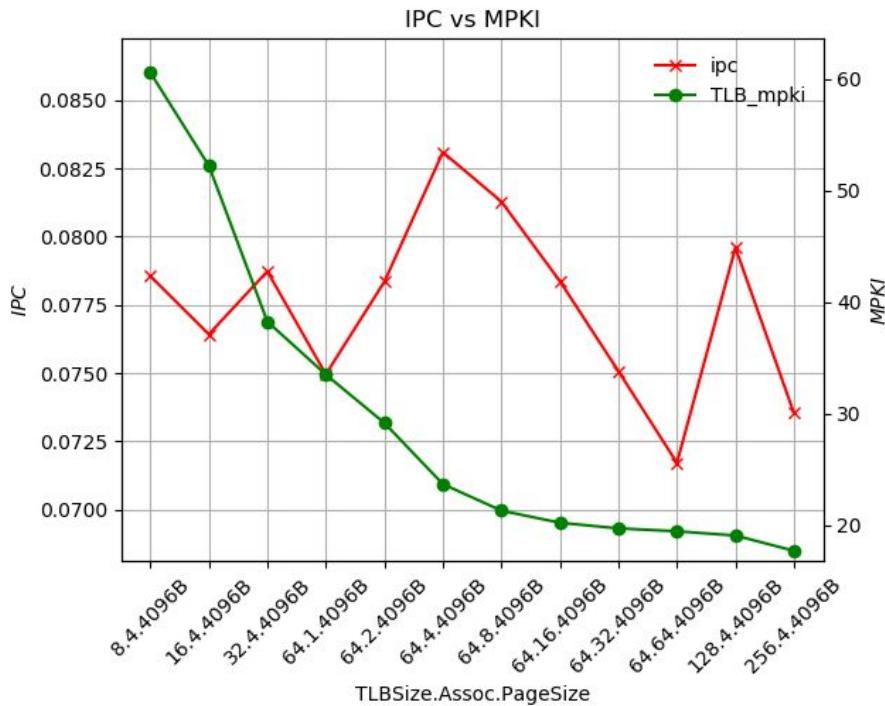
- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc από 1-2 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 32 entr. οπου μετά το οποίο χειροτερεύει την επίδοση, αυτό συμβαίνει καθώς αρχικά κερδίζει η βελτίωση που υπάρχει και στο ιδανικό πρόγραμμα αλλά μετά από τα 32 κερδίζει η ποινή.
- Όσον αφορά το Associativity: Για αύξηση του assoc από 1-2 έχουμε βελτίωση της επίδοσης, για το βήμα 2-4 έχουμε σταθερή βελτίωση αφου ακόμη δεν υπάρχει κάποια ποινή και από εκεί και πέρα οποιαδήποτε αύξηση του assoc χειροτερεύει σημαντικά την επίδοση.

Παρατηρούμε ότι η βέλτιστη απόδοση δίνεται για 32.4.4096B που **δεν ταυτίζεται** με την βέλτιστη απόδοση που δίνει το ιδανικό μοντέλο.

Canneal



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

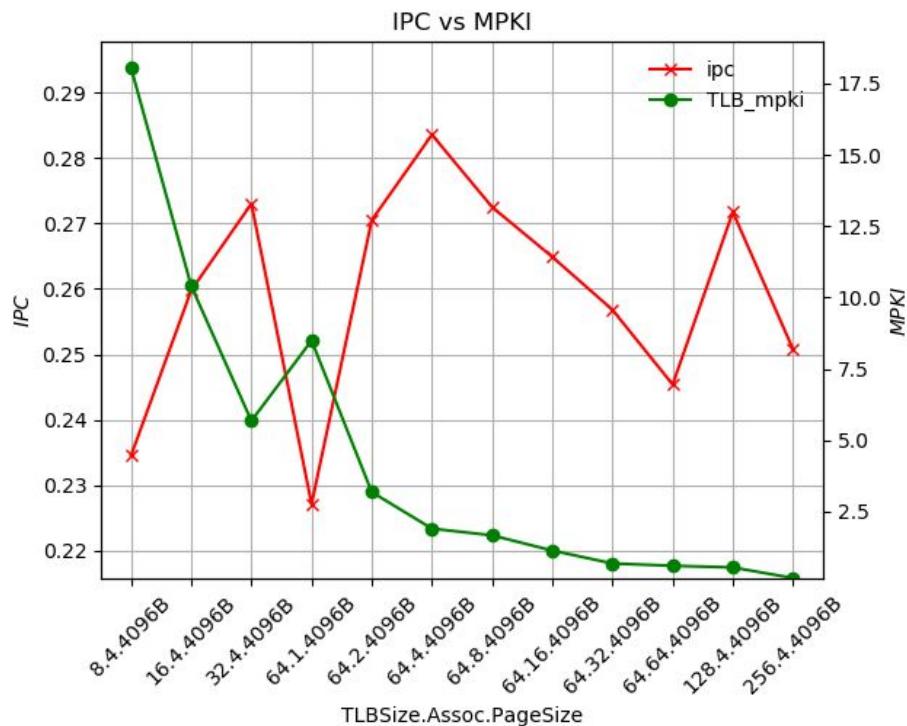
- Όσον αφορα το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=8 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορα το TLB Size: Η αύξηση του cache size **χειροτερεύει** την επίδοση του προγράμματος με εξαίρεση το βήμα από 16entr στα 32entr στο οποίο έχουμε μικρή βελτίωση, αυτό συμβαίνει επειδή μόνο τότε το συνολικό κέρδος που δίνει το ιδανικό συστήμα υπερνικά την ποινή λόγω αύξησης του size.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc **χειροτερεύει** σημαντικά την επίδοση.

Παρατηρούμε ότι η βέλτιστη απόδοση δίνεται για 64.4.4096B που **δεν ταυτίζεται** με την βέλτιστη απόδοση που δίνει το ιδανικό μοντέλο.

Facesim



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

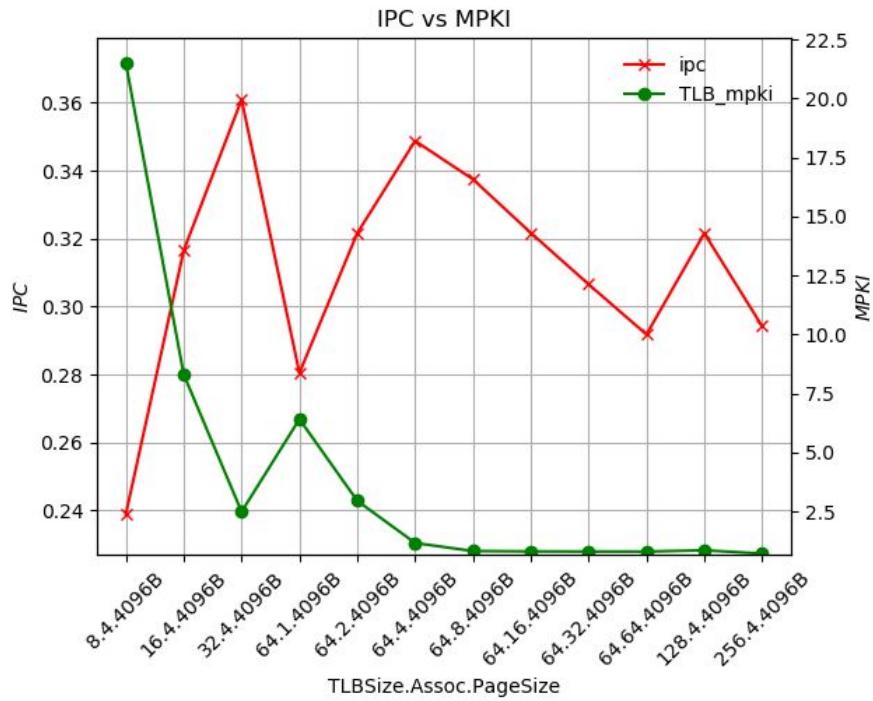
- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=832 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 32 entr. οπου μετά το οποίο χειροτερεύει την επίδοση, αυτό συμβαίνει καθώς αρχικά κερδίζει η βελτίωση που υπάρχει και στο ιδανικό πρόγραμμα αλλά μετά από τα 32 κερδίζει η ποινή.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc χειροτερεύει σημαντικά την επίδοση.

Παρατηρούμε ότι η βέλτιστη απόδοση δίνεται για 64.4.4096B που **δεν ταυτίζεται** με την βέλτιστη απόδοση που δίνει το ιδανικό μοντέλο.

Ferret



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

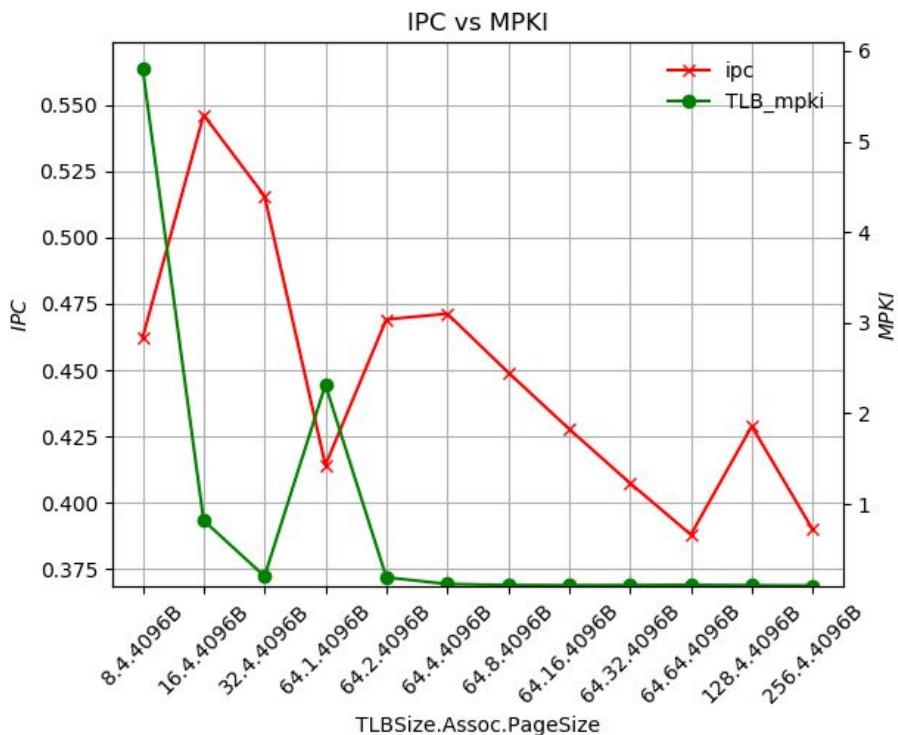
- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 64 entr. οπου μετά το οποίο δεν επηρεάζει σχεδόν καθόλου
- Όσον αφορά το Associativity: Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=8 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 32 entr. οπου μετά το οποίο χειροτερεύει την επίδοση, αυτό συμβαίνει καθώς αρχικά κερδίζει η βελτίωση που υπάρχει και στο ιδανικό πρόγραμμα αλλά μετά από τα 32 κερδίζει η ποινή.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc χειροτερεύει σημαντικά την επίδοση.

Παρατηρούμε ότι η βέλτιστη απόδοση δίνεται για 32.4.4096B που **δεν ταυτίζεται** με την βέλτιστη απόδοση που δίνει το ιδανικό μοντέλο.

Fluidanimate



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

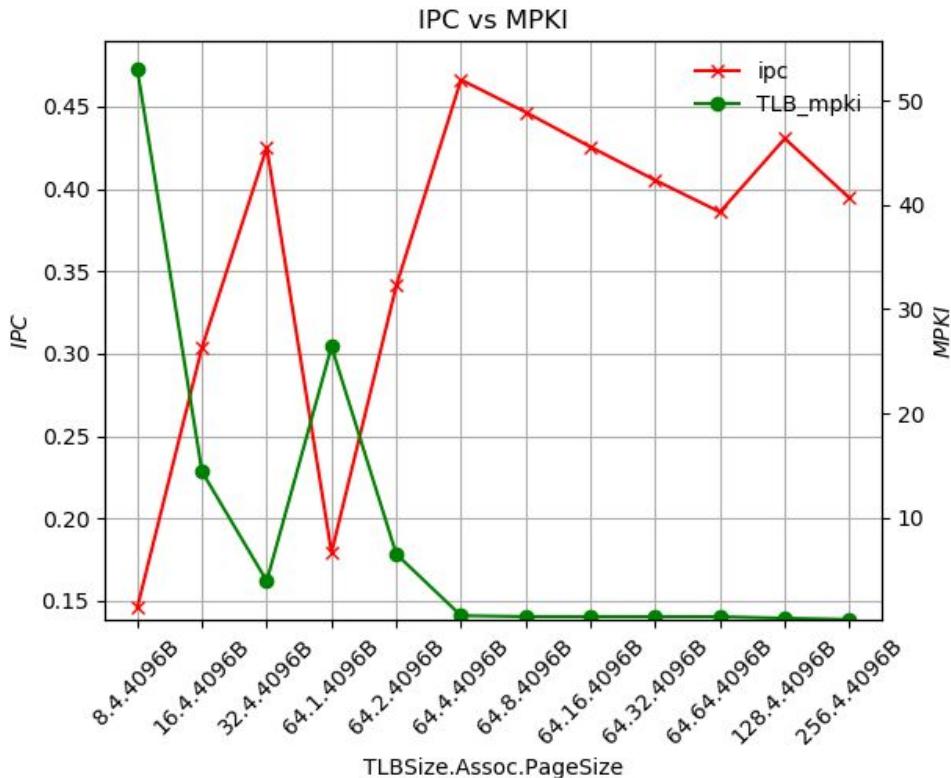
- Όσον αφορα το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 64 entr. οπου μετά το οποίο δεν επηρεάζει σχεδόν καθόλου
- Όσον αφορά το Associativity: Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=8 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορα το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 16 entr. οπου μετά το οποίο χειροτερεύει την επίδοση, αυτό συμβαίνει καθώς αρχικά κερδίζει η βελτίωση που υπάρχει και στο ιδανικό πρόγραμμα αλλά μετά από τα 16 κερδίζει η ποινή.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc χειροτερεύει σημαντικά την επίδοση.

Παρατηρούμε ότι η βέλτιστη απόδοση δίνεται για 16.4.4096B που **δεν ταυτίζεται** με την βέλτιστη απόδοση που δίνει το ιδανικό μοντέλο.

Freqmine



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

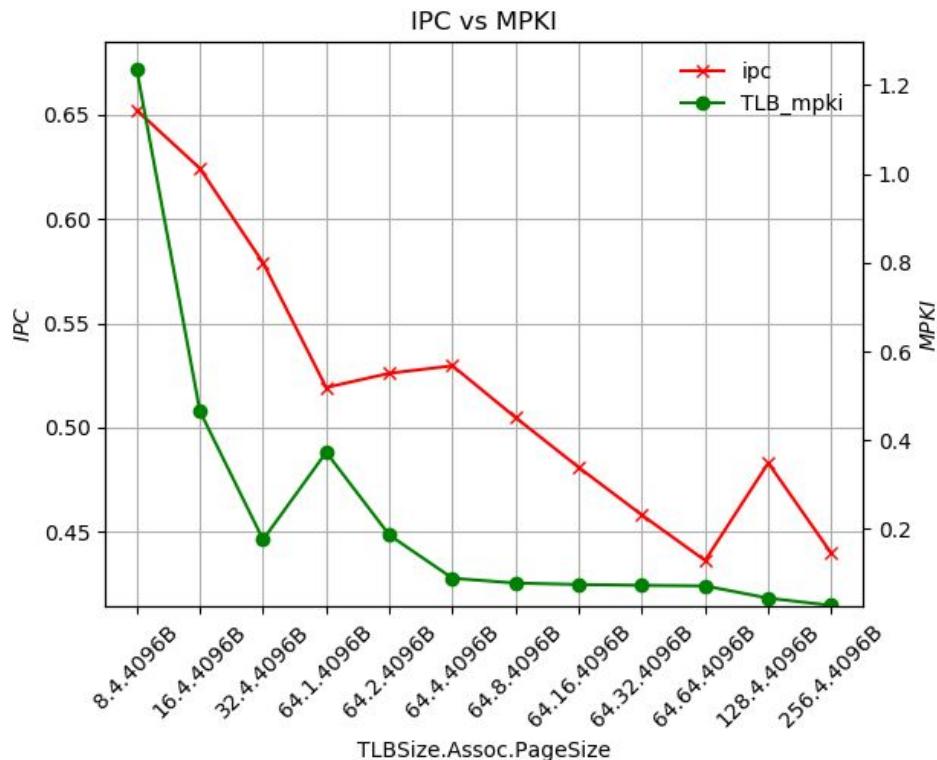
- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το threshold του 64 entr. οπου μετά το οποίο χειροτερεύει την επίδοση, αυτό συμβαίνει καθώς αρχικά κερδίζει η βελτίωση που υπάρχει και στο ιδανικό πρόγραμμα αλλά μετά από τα 64 κερδίζει η ποινή.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc χειροτερεύει σημαντικά την επίδοση.

Παρατηρούμε ότι η βέλτιστη απόδοση δίνεται για 64.4.4096B που **δεν ταυτίζεται** με την βέλτιστη απόδοση που δίνει το ιδανικό μοντέλο.

rtview



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

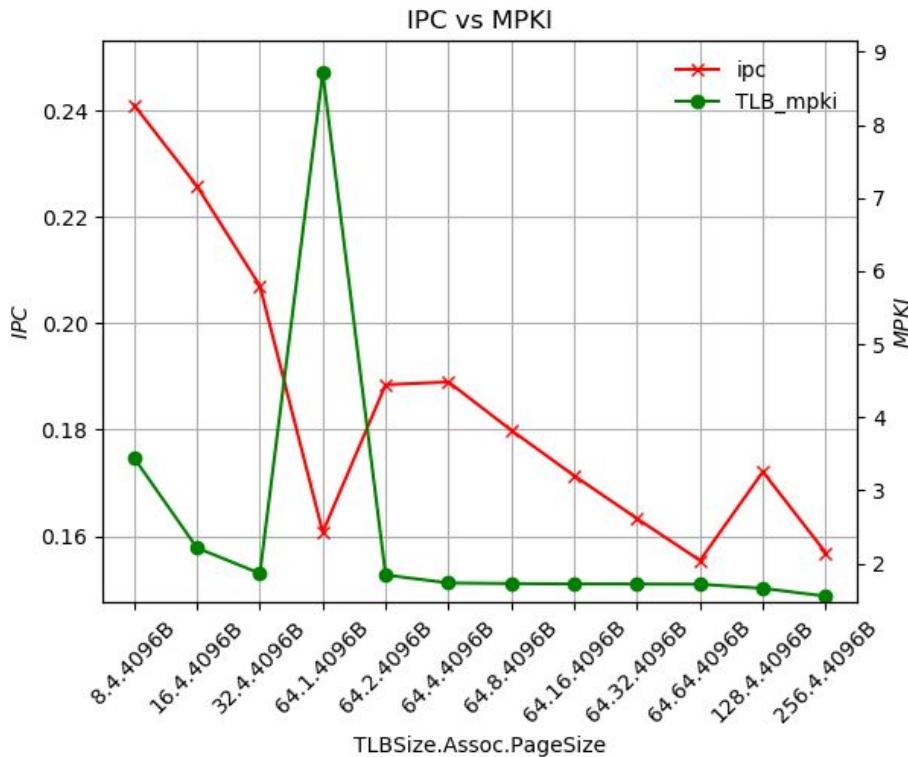
- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το TLB Size: Η αύξηση του cache size **χειροτερεύει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc **χειροτερεύει** σημαντικά την επίδοση.

Παρατηρούμε ότι η βέλτιστη απόδοση δίνεται για 8.4.4096B που **δεν ταυτίζεται** με την βέλτιστη απόδοση που δίνει το ιδανικό μοντέλο.

Streamcluster



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

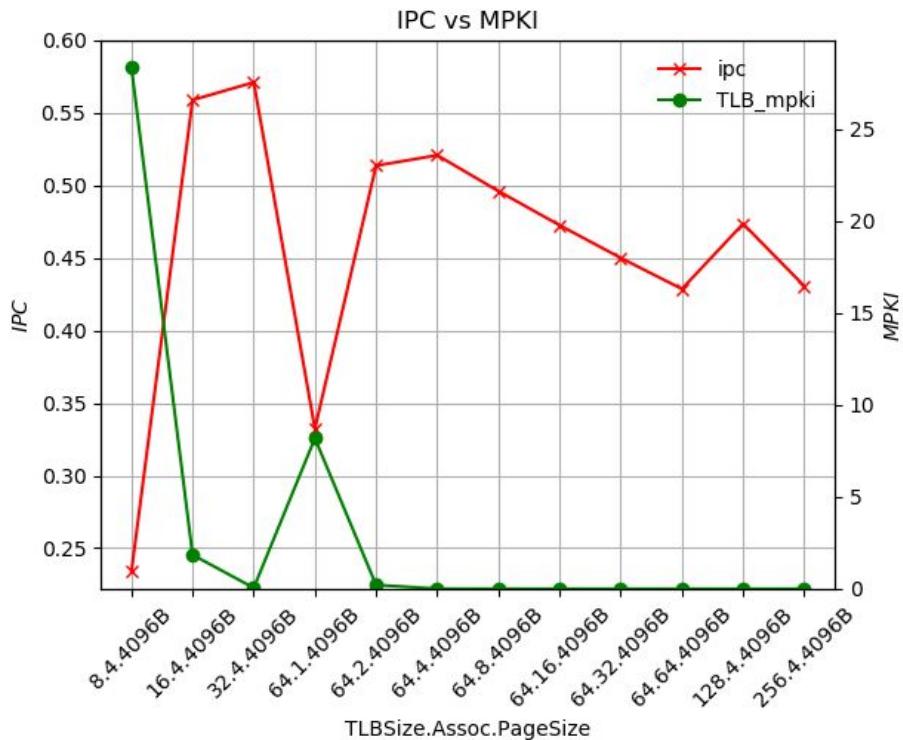
- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=2 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το TLB Size: Η αύξηση του cache size **χειροτερεύει** την επίδοση του προγράμματος, με εξαίρεση την μετάβαση από 32-64 όπου την βελτιώνει.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc χειροτερεύει σημαντικά την επίδοση.

Παρατηρούμε ότι η βέλτιστη απόδοση δίνεται για 8.4.4096B που **δεν ταυτίζεται** με την βέλτιστη απόδοση που δίνει το ιδανικό μοντέλο.

Swaptions



Για το ιδανικό παράδειγμα είχαμε δει τα εξείς:

- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το κατώφλι των 32 entr. από εκεί και πέρα δεν επηρεάζει την επίδοση του συστήματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=2 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc δεν επηρεάζει σημαντικά την επίδοση.

Με βάση αυτά και προσθέτοντας την ποινή που προσθέτει η κάθε αύξηση του size και του assoc μπορούμε να πούμε ότι ισχύουν τα εξής γι' αυτη την περίπτωση:

- Όσον αφορά το TLB Size: Η αύξηση του cache size **βελτιώνει** την επίδοση του προγράμματος μέχρι το κατώφλι των 32 entr. από εκεί και πέρα χειροτερεύει την επίδοση του συστήματος.
- Όσον αφορά το Associativity: Για αύξηση του assoc μέχρι και την τιμή assoc=4 έχουμε βελτίωση της επίδοσης, και από εκεί και πέρα οποιαδήποτε αύξηση του assoc χειροτερεύει σημαντικά την επίδοση.

Παρατηρούμε ότι η βέλτιστη απόδοση δίνεται για 32.4.4096B που **ταυτίζεται** με την βέλτιστη απόδοση που δίνει το ιδανικό μοντέλο.

Συμπεράσματα Β Μέρους

Σχετικά με την L1 Cache

Γενικά παρατηρούμε ότι τα προγράμματα που πετυχαίνουν καλύτερη απόδοση για μεγάλα cache size και assoc δεν καταφέρουν να αποδώσουν τόσο καλά όπως πριν επειδή τα επηρεάζουν οι ποινές που έρχονται με την αύξηση αυτών των δύο παραγόντων.

Από την άλλη τα προγράμματα για τα οποία το ipc τους επηρεάζεται θετικά από την αύξηση του block size καταφέρνουν να μην επηρεαστούν τόσο δραματικά από την ποινή των κύκλων.

Όπως είδαμε και στο συμπέρασμα του πρώτου μέρους επειδή λίγα προγράμματα καταφέρνουν να δουλέψουν καλύτερα με αύξηση του assoc και δεδομένου ότι τώρα η αυξηση του assoc είναι μια επιλογή που χειροτερεύει την επιδοση λόγω της ποινής, συμπεραίνουμε ότι δεν χρειάζεται να μεγαλώνουμε το assoc και ότι πρέπει να το αφήσουμε στην default τιμή των 4 way assoc.

Επίσης, βλέπουμε ότι η αύξηση του μεγέθους έρχεται με πολύ μεγάλο κόστος επομένως δεν είναι λύση και γι αυτό πρέπει να την αφήνουμε στην default τιμή.

Τα προγράμματα που δουλεύουν καλά είναι αυτα που η επίδοση τους βελτιώνεται με την αύξηση του ipc και πάλι καλά για τα περισσότερα προγράμματα ισχύει αυτό.

Ένα πρόβλημα που έχουν τα input του διαγράμματος μας είναι ότι δεν δίνουμε υψηλή τιμή για τα default size και assoc και γι αυτό βλέπουμε ότι όταν η τάση για αυξηση λόγω του block size είναι πολύ υψηλή επιτυγχάνουμε την καλύτερη απόδοση με αύξηση του assoc κατα τον διπλασιασμό του και το μέγιστο δυνατό block size. Άλλα προσοχή αυτο τις περισσότερες φορές δεν συμβαίνει επειδή το assoc βοηθάει αλλά επειδη το block size βοηθαει πολυ και υπερνικά και το κόστος της ποινής λογω της αυξησης του assoc.

Σχετικά με την L2 Cache

Γενικά παρατηρούμε ότι τα προγράμματα που πετυχαίνουν καλύτερη απόδοση για μεγάλα cache size και assoc δεν καταφέρουν να αποδώσουν τόσο καλά όπως πριν επειδή τα επηρεάζουν οι ποινές που έρχονται με την αύξηση αυτών των δύο παραγόντων.

Από την άλλη τα προγράμματα για τα οποία το iρc τους επηρεάζεται θετικά από την αύξηση του block size καταφέρνουν να μην επηρεαστουν τόσο δραματικά από την ποινή των κύκλων.

Όπως είδαμε και στο συμπέρασμα του πρώτου μέρους επειδή λίγα προγράμματα καταφέρνουν να δουλέψουν καλύτερα με αύξηση του assoc και δεδομένου ότι τώρα η αυξηση του assoc είναι μια επιλογή που χειροτερεύει την επιδοση λόγω της ποινής, συμπεραίνουμε ότι δεν χρειάζεται να μεγαλώνουμε το assoc και ότι πρέπει να το αφήσουμε στην default τιμή των 4 way assoc.

Επίσης, βλέπουμε ότι η αύξηση του μεγέθους έρχεται με πολύ μεγάλο κόστος επομένως δεν είναι λύση και γι αυτό πρέπει να την αφήνουμε στην default τιμή.

Τα προγράμματα που δουλεύουν καλά είναι αυτα που η επίδοση τους βελτιώνεται με την αύξηση του iρc και πάλι καλά για τα περισσότερα προγράμματα ισχύει αυτό.

Επομένως τις περισσότερες φορές οι βέλτιστη στρατηγική είναι να αφήσουμε σε default τιμές τα size και assoc και να αυξήσουμε το μέγιστο δυνατό το block size, γι αυτό και τις περισσότερες φορές βλέπουμε ως βέλτιστο συνδυασμό τον 512KB.8.256B.

Σχετικά με το TLB

Το γενικό συμπέρασμα για το tlb χωρίς ποινές αύξησης κύκλου ήταν αυτό:

Γενικά παρατηρούμε ότι έχουμε συμμετοχή και των δυο παραμέτρων σε όλα τα benchmarks.

Το χαμηλό assoc δίνει καταστροφικά αποτελέσματα στο ijc ωστόσο γενικά για τιμή μεγαλύτερη του 4-8 δεν μεταβάλλει ουσιαστικά το ijc.

Έτσι τώρα βλέπουμε ότι τρεις είναι οι καθοριστικές τιμές που θα δίνουν βέλτιστη απόδοση:

- 8.4.4096B(default)
- 32.4.4096B
- 64.4.4096B

Αυτό συμβαίνει γιατί οποιαδήποτε παραπάνω αύξηση του size θα ήταν μη αποδοτική λόγω της πολύ μεγάλης αύξησης των κύκλων, όπως επίσης επειδή βλέπουμε ένα κατώφλι το assoc=4 στην αύξηση του assoc, δηλαδή οποιαδήποτε άλλη αυξηση δημιουργεί αχρειαστο κόστος.

Γενικά τα tlb που δεν είχαν πολύ μεγάλη βελτίωση από τις αυξήσεις των size και assoc καταλήγουν να έχουν βέλτιστη λύση την default (ή κοντά στην default, αν έχουν κάπως μεγαλύτερη βελτίωση).

Συγκριτικά και με τα προηγούμενα συμπεράσματα βλέπουμε ότι το tlb δουλεύει καλά για μεγαλύτερες τιμές από τις default ενω τα L1,L2 όχι. Αυτό δείχνει την ανάγκη να έχουμε ένα κάπως μεγαλό tlb αλλά όχι και πολύ μεγάλο γιατί θα ήταν δυσχρειστο και θα έδινε μεγάλες ποινές σε κύκλους. Δηλαδή το ποσοστό της αυξησης του ijc είναι τόσο μεγάλο που υπερνικά τις ποινές.

18. Παράρτημα

Αρχικά γίνανε κατάλληλες αλλαγές στον cache.h για τις σωστές καθυστερήσεις

```
public:  
    // constructors/destructors  
    TWO_LEVEL_CACHE(std::string name,  
                    UINT32 l1CacheSize, UINT32 l1BlockSize, UINT32 l1Associativity,  
                    UINT32 l2CacheSize, UINT32 l2BlockSize, UINT32 l2Associativity,  
                    UINT32 l2PrefetchLines,  
                    UINT32 l1HitLatency = 1, UINT32 l2HitLatency = 15,  
                    UINT32 l2MissLatency = 250);
```

Μετά γίνανε κατάλληλες αλλαγές στον tlb.h για τις σωστές καθυστερήσεις:

```
public:  
    // constructors/destructors  
    SINGLE_LEVEL_TLB(std::string name,  
                      UINT32 Entries, UINT32 PageSize, UINT32 Associativity,  
                      UINT32 HitLatency = 0, UINT32 MissLatency = 100);  
    ...
```

και μετά έγιναν αλλαγές για την διόρθωση του prefetching στο cache.h

```
cache.h  
~/Desktop/comparch/advcomparch-2019-2020-ex1-helpcode/pintool  
Save  
...  
cycles += _latencies[MISS_L2];  
  
// If L2 is inclusive and a TAG has been replaced we need to remove  
// all evicted blocks from L1.  
if ((L2_INCLUSIVE == 1) && !(l2_replaced == INVALID_TAG)) {  
    ADDRINT replacedAddr = ADDRINT(l2_replaced) << FloorLog2(L2NumSets());  
    replacedAddr = replacedAddr | l2SetIndex;  
    replacedAddr = replacedAddr << L2LineShift();  
    for (UINT32 i=0; i < L2BlockSize(); i+=L1BlockSize()) {  
        ADDRINT newAddr = replacedAddr | i;  
        SplitAddress(newAddr, L1LineShift(), L1SetIndexMask(), l1Tag, l1SetIndex);  
        l1Set = _l1_sets[l1SetIndex];  
        l1Set.DeleteIfPresent(l1Tag);  
    }  
}  
// PREFETCHING  
ADDRINT prefetch_addr = addr;  
for (UINT32 i=0; i < _l2_prefetch_lines; i++) {  
    prefetch_addr += L2BlockSize();  
  
    SplitAddress(prefetch_addr, L2LineShift(), L2SetIndexMask(), l2Tag, l2SetIndex);  
    SET & l2Set = _l2_sets[l2SetIndex];  
    l2Hit = l2Set.Find(l2Tag);  
  
    if(!l2Hit){  
        CACHE_TAG l2_replaced = l2Set.Replace(l2Tag);  
  
        if ((L2_INCLUSIVE == 1) && !(l2_replaced == INVALID_TAG)) {  
            ADDRINT replacedAddr = ADDRINT(l2_replaced) << FloorLog2(L2NumSets());  
            replacedAddr = replacedAddr | l2SetIndex;  
            replacedAddr = replacedAddr << L2LineShift();  
            for (UINT32 i = 0; i < L2BlockSize(); i += L1BlockSize()) {  
                ADDRINT newAddr = replacedAddr | i;  
                SplitAddress(newAddr, L1LineShift(), L1SetIndexMask(), l1Tag, l1SetIndex);  
                l1Set = _l1_sets[l1SetIndex];  
                l1Set.DeleteIfPresent(l1Tag);  
            }  
        }  
    }  
}  
}  
  
return cycles;  
}  
#endif // CACHE_H
```

A μέρος scripts:

L1 outputs:

```
#!/bin/bash

## Modify the following paths appropriately
PARSEC_PATH=/home/avocoder/Desktop/comparch/parssec-3.0
PIN_EXE=/home/avocoder/Desktop/comparch/pin-3.13-98189-g60a6ef199-gcc-linux/pin
PIN_TOOLS=/home/avocoder/Desktop/comparch/dvcomparch-2019-2020-exi-helpcode/pintool/obj-intel64/simulator.so

CMOS_FILE=/home/avocoder/Desktop/cmds_simlarge.txt
outDir="/home/avocoder/Desktop/outputs/"

export LD_LIBRARY_PATH=$PARSEC_PATH/pkg/lib/hooks/inst/amd64-linux.gcc-serial/lib

##export LD_LIBRARY_PATH=/home/avocoder/Desktop/comparch/parssec-3.0/pkg/lib/hooks/inst/amd64-linux.gcc-serial/lib

## Triples of <cache_size>_<associativity>_<block_size>
##CONF=$"16_4_64 32_4_64"

CONF=$"32_4_64 32_8_32 32_8_64 32_8_128 64_4_64 64_8_32 64_8_64 64_8_128 128_8_32 128_8_64 128_8_128"

L2size=1024
L2assoc=8
L2bsize=128
TLBe=0
TLBp=096
TLBa=6
L2prf=0

for BENCH in $@; do
    cmd=$(cat $CMOS_FILE | grep "$BENCH")
    for conf in $CONF; do
        ## Get parameters
        Lsize=$(echo $conf | cut -d'_' -f1)
        Lassoc=$(echo $conf | cut -d'_' -f2)
        Libsize=$(echo $conf | cut -d'_' -f3)

        outFile=$(printf "%s.cache_cslab.L1_%04d_%02d_%03d.out" $BENCH ${Lsize} ${Lassoc} ${Libsize})
        outFile="$outDir/$outFile"

        pin_cmd=$PIN_EXE -t $PIN_TOOLS -o $outFile -Lic ${Lsize} -Lia ${Lassoc} -Llb ${Libsize} -L2c ${L2size} -L2a ${L2assoc} -L2b ${L2bsize} -TLBe ${TLBe} -TLBp ${TLBp} -TLBa ${TLBa} -L2prf ${L2prf} -- $cmd"
        time $pin_cmd
    done
done
```

L1 plots:

```
#!/usr/bin/env python

import sys
import numpy as np

## We need matplotlib:
## $ apt-get install python-matplotlib
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

x_Axis = []
ipc_Axis = []
mpki_Axis = []

for outfile in sys.argv[1:]:
    fp = open(outfile)
    line = fp.readline()
    while line:
        tokens = line.split()
        if (line.startswith(" Total Instructions: ")):
            total_instructions = long(tokens[2])
        elif (line.startswith("IPC")):
            ipc = float(tokens[2])
        elif (line.startswith(" L1-Data Cache")):
            sizeLine = fp.readline()
            l1_size = sizeLine.split()[1]
            bsizeLine = fp.readline()
            l1_bsize = bsizeLine.split()[2]
            assocLine = fp.readline()
            l1_assoc = assocLine.split()[1]
        elif (line.startswith("L1-Total-Misses")):
            l1_total_misses = long(tokens[1])
            l1_miss_rate = float(tokens[2].split('%')[0])
            mpki = l1_total_misses / (total_instructions / 1000.0)

        line = fp.readline()
    fp.close()

    l1ConfigStr = '({0},{1},{2})'.format(l1_size,l1_assoc,l1_bsize)
    print l1ConfigStr
    x_Axis.append(l1ConfigStr)
    ipc_Axis.append(ipc)
    mpki_Axis.append(mpki)

print x_Axis
print ipc_Axis
print mpki_Axis

fig, ax1 = plt.subplots()
ax1.grid(True)
ax1.set_xlabel("CacheSize.Assoc.BlockSize")
ax1.set_xlabel("CacheSize.Assoc.BlockSize")

xAx = np.arange(len(x_Axis))
ax1.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
ax1.set_xticklabels(x_Axis, rotation=45)
ax1.set_xlim(-0.5, len(x_Axis) - 0.5)
ax1.set_ylim(min(ipc_Axis) - 0.05 * min(ipc_Axis), max(ipc_Axis) + 0.05 * max(ipc_Axis))
ax1.set_ylabel("IPC")
lne1 = ax1.plot(ipc_Axis, label="ipc", color="red", marker='x')

ax2 = ax1.twinx()
ax2.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
ax2.set_xticklabels(x_Axis, rotation=45)
ax2.set_xlim(-0.5, len(x_Axis) - 0.5)
ax2.set_ylim(min(mpki_Axis) - 0.05 * min(mpki_Axis), max(mpki_Axis) + 0.05 * max(mpki_Axis))
ax2.set_ylabel("MPKI")
lne2 = ax2.plot(mpki_Axis, label="L1_mpki", color="green", marker='o')

lns = lne1 + lne2
labs = [l.get_label() for l in lns]

plt.title("IPC vs MPKI")
lgd = plt.legend(lns, labs)
lgd.draw_frame(False)
plt.savefig("L1.png", bbox_inches="tight")
```

L2 outputs:

```
1 #!/bin/bash
2
3 ## Modify the following paths appropriately
4 PARSEC_PATH=/home/avocoder/Desktop/comparch/parsec-3.0
5 PIN_EXE=/home/avocoder/Desktop/comparch/pin-3.13-98189-g68a0ef199-gcc-linux/pin
6 PIN_TOOL=/home/avocoder/Desktop/comparch/advcparcmarch-2819-2828-ex1-heipcode/pintool/obj-intel64/simulator.so
7
8 CMDS_FILE=/home/avocoder/Desktop/cmds_simlarge.txt
9 outDir=/home/avocoder/Desktop/output/
10
11 export LD_LIBRARY_PATH=$PARSEC_PATH/pkg/lib/hooks/inst/amd64-linux.gcc-serial/lib/
12
13 ## Triples of <cache_size>-<associativity>-<block_size>
14
15 CONF5="512_8_64 512_8_128 512_8_256 1024_8_64 1024_8_128 1024_8_256 1024_16_64 1024_16_128 1024_16_256 2048_16_64 2048_16_128 2048_16_256"
16
17 Lsize=32
18 Lassoc=8
19 Lbsize=4
20 TLBe=4
21 TLBp=4096
22 TLBa=4
23 L2prf=0
24
25
26 for BENCH in $@; do
27     cmd=$(cat ${CMDS_FILE} | grep "$BENCH")
28     for conf in ${CONF5}; do
29         ## Get parameters
30         L2size=$(echo $conf | cut -d '_' -f1)
31         L2assoc=$(echo $conf | cut -d '_' -f2)
32         L2bsize=$(echo $conf | cut -d '_' -f3)
33
34         outFile=$(printf "%s.dcache_cslab.L2_%04d.%02d.out" $BENCH ${L2size} ${L2assoc} ${L2bsize})
35         outFile="$outDir/$outFile"
36
37         pin_cmd="`SPIN_EXE -t $PIN_TOOL -o $outFile -l1c ${Lsize} -l1a ${Lassoc} -l1b ${Lbsize} -l2c ${L2size} -l2a ${L2assoc} -l2b ${L2bsize} -TLBe ${TLBe} -TLBp ${TLBp} -TLBa ${TLBa} -L2prf ${L2prf} --"
38         time spin_cmd
39     done
40 done
41
```

L2 plots:

```
1#!/usr/bin/env python
2
3 import sys
4 import numpy as np
5
6 ## We need matplotlib:
7 ## $ apt-get install python-matplotlib
8 import matplotlib
9 matplotlib.use('Agg')
10 import matplotlib.pyplot as plt
11
12 x_Axis = []
13 ipc_Axis = []
14 mpki_Axis = []
15
16 for outFile in sys.argv[1:]:
17     fp = open(outFile)
18     line = fp.readline()
19     while line:
20         tokens = line.split()
21         if (line.startswith("Total Instructions: ")):
22             total_instructions = long(tokens[1])
23         elif (line.startswith("IPC")):
24             ipc = float(tokens[1])
25         elif (line.startswith(" L2-Data Cache")):
26             sizeLine = fp.readline()
27             L2_size = sizeLine.split()[1]
28             bsizeLine = fp.readline()
29             L2_bsize = bsizeLine.split()[1]
30             assocLine = fp.readline()
31             L2_assoc = assocLine.split()[1]
32         elif (line.startswith(" L2-Total-Misses")):
33             L2_total_misses = long(tokens[1])
34             L2_miss_rate = float(tokens[2].split('%')[0])
35             mpki = L2_total_misses / (total_instructions / 1000.0)
36
37         line = fp.readline()
38
39     fp.close()
40
41 L2configStr = '{:<15.17.17}B'.format(L2_size,L2_assoc,L2_bsize)
42 print L2ConfigStr
43 x_Axis.append(L2configStr)
44 ipc_Axis.append(ipc)
45 mpki_Axis.append(mpki)
46
47 print x_Axis
48 print ipc_Axis
49 print mpki_Axis
50
51
52 fig, ax1 = plt.subplots()
53 ax1.grid(True)
54 ax1.set_xlabel("CacheSize.Assoc.BlockSize")
55
56 xAx = np.arange(len(x_Axis))
57 ax1.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
58 ax1.set_xticklabels(x_Axis, rotation=45)
59 ax1.set_xlim(-0.5, len(x_Axis) - 0.5)
60 ax1.set_ylim(min(ipc_Axis) - 0.05 * min(ipc_Axis), max(ipc_Axis) + 0.05 * max(ipc_Axis))
61 ax1.set_ylabel("IPC")
62 line1 = ax1.plot(ipc_Axis, label="ipc", color="red", marker='x')
63
64 ax2 = ax1.twinx()
65 ax2.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
66 ax2.set_xticklabels(x_Axis, rotation=45)
67 ax2.set_xlim(-0.5, len(x_Axis) - 0.5)
68 ax2.set_ylim(min(mpki_Axis) - 0.05 * min(mpki_Axis), max(mpki_Axis) + 0.05 * max(mpki_Axis))
69 ax2.set_ylabel("MPKI")
70 line2 = ax2.plot(mpki_Axis, label="L2_mpki", color="green", marker='o')
71
72 lns = line1 + line2
73 labs = [l.get_label() for l in lns]
74
75 plt.title("IPC vs MPKI")
76 lgd = plt.legend(lns, labs)
77 lgd.draw_frame(False)
78 plt.savefig("L2.png", bbox_inches="tight")
```

TLB outputs:

```

1 #!/bin/bash
2
3 ## Modify the following paths appropriately
4 PARSEC_PATH=/home/avocoder/Desktop/comparch/parsec-3.0
5 PIN_EXE=/home/avocoder/Desktop/comparch/pin-3.13-78189-g0a6ef199-gcc-linux/pin
6 PIN_TOOL=/home/avocoder/Desktop/comparch/dvcomparc-2019-2020-ex1-heipcode/pintool/obj-intel64/simulator.so
7
8 CMS_FILE=/home/avocoder/Desktop/cmds_elimlarge.txt
9 outdir=/home/avocoder/Desktop/output/
10
11 export LD_LIBRARY_PATH=$PARSEC_PATH/pkg/libs/hooks/inst/amd64-linux.gcc-serial/lib/
12
13
14 CONFS="8_4_4096 16_4_4096 32_4_4096 64_1_4096 64_2_4096 64_4_4096 64_8_4096 64_16_4096 64_32_4096 64_64_4096 128_4_4096 256_4_4096"
15
16 L1size=8
17 L1assoc=8
18 L1bsize=64
19
20 L2size=1024
21 L2assoc=8
22 L2bsize=128
23
24 L2prf=0
25
26 BENCHMARKS="blackholes bodytrack canneal facesim ferret fluidanimate freqmine rtview streamcluster swaptions"
27
28 for BENCH in $BENCHMARKS; do
29     cmd=$(cat $CMS_FILE | grep "$BENCH")
30     for conf in $CONFS; do
31         ## Get parameters
32         TLBe=$(echo $conf | cut -d '-' -f1)
33         TLBa=$(echo $conf | cut -d '-' -f2)
34         TLBp=$(echo $conf | cut -d '-' -f3)
35
36         outFile=$(printf "%s.dcache_cslab.TLB_%04d_%02d_%03d.out" $BENCH ${TLBe} ${TLBa} ${TLBp})
37         outFile=$outdir/$outFile
38
39         pin_cmd=$PIN_EXE -t $PIN_TOOL -o $outFile -Lic ${L1size} -L1a ${L1assoc} -L1b ${L1bsize} -L2c ${L2size} -L2a ${L2assoc} -L2b ${L2bsize} -TLBe ${TLBe} -TLBp ${TLBp} -TLBa ${TLBa} -L2prf ${L2prf} --
40         $conf
41         time $pin_cmd
42     done
43 done

```

TLB plots:

```

1 #!/usr/bin/env python
2
3 import sys
4 import numpy as np
5 import matplotlib
6 matplotlib.use('Agg')
7 import matplotlib.pyplot as plt
8
9 x_Axis = []
10 ipc_Axis = []
11 mpki_Axis = []
12
13 for outFile in sys.argv[1:]:
14     fp = open(outFile)
15     line = fp.readline()
16     while line:
17         tokens = line.split()
18         if (line.startswith("Total Instructions: ")):
19             total_instructions = long(tokens[2])
20         elif (line.startswith("IPC")):
21             ipc = float(tokens[1])
22         elif (line.startswith("Data TLB")):
23             sizeLine = fp.readline()
24             tlbe_entries = sizeLine.split()[1]
25             bsizeLine = fp.readline()
26             tlbp_size = bsizeLine.split()[2]
27             assocLine = fp.readline()
28             tlba_assoc = assocLine.split()[1]
29         elif (line.startswith("TLB-Total-Misses")):
30             tlb_total_misses = long(tokens[1])
31             tlbp_miss_rate = float(tokens[1].split('%')[0])
32             mpki = tlb_total_misses / (total_instructions / 1000.0)
33
34         line = fp.readline()
35
36     fp.close()
37
38     tlbeConfigStr = '{:.0f}'.format(tlbe_entries,tlba_assoc,tlbp_size)
39     print(tlbeConfigStr)
40     x_Axis.append(tlbeConfigStr)
41     ipc_Axis.append(ipc)
42     mpki_Axis.append(mpki)
43
44     print(x_Axis)
45     print(ipc_Axis)
46     print(mpki_Axis)
47
48     fig, ax1 = plt.subplots()
49     ax1.grid(True)
50     ax1.set_xlabel("TLBSize.Assoc.PageSize")
51
52     xAx = np.arange(len(x_Axis))
53     ax1.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
54     ax1.set_xticklabels(x_Axis, rotation=45)
55     ax1.set_xlim(-0.5, len(x_Axis) - 0.5)
56     ax1.set_ylin(min(ipc_Axis) - 0.05 * min(ipc_Axis), max(ipc_Axis) + 0.05 * max(ipc_Axis))
57     ax1.set_ylabel("IPC")
58     line1 = ax1.plot(ipc_Axis, label="IPC", color="red", marker='x')
59
60     ax2 = ax1.twinx()
61     ax2.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
62     ax2.set_xticklabels(x_Axis, rotation=45)
63     ax2.set_xlim(-0.5, len(x_Axis) - 0.5)
64     ax2.set_ylin(min(mpki_Axis) - 0.05 * min(mpki_Axis), max(mpki_Axis) + 0.05 * max(mpki_Axis))
65     ax2.set_ylabel("MPKI")
66     line2 = ax2.plot(mpki_Axis, label="MPKI", color="green", marker='o')
67
68     lns = line1 + line2
69     labs = [l.get_label() for l in lns]
70
71     plt.title("IPC vs MPKI")
72     lgd = plt.legend(lns, labs)
73     lgd.draw_frame(False)
74     plt.savefig("TLB.png", bbox_inches="tight")

```

pref outputs:

```
1 #!/bin/bash
2
3 ## Modify the following paths appropriately
4 PARSEC_PATH=/home/avocoder/Desktop/comparc/pin-3.13-98189-g69a6ef199-gcc-linux/pin
5 PIN_EXE=/home/avocoder/Desktop/comparc/pin-3.13-98189-g69a6ef199-gcc-linux/pin
6 PIN_TOOL=/home/avocoder/Desktop/comparc/advcmparch-2019-2020-ex1-helpcode/pintool-intel64/simulator.so
7
8 CMDS_FILE=/home/avocoder/Desktop/cmnds_simlarge.txt
9 outDir=/home/avocoder/Desktop/outputspref
10
11 export LD_LIBRARY_PATH=$PARSECPATH/pkgslibs/hooks/inst/amd64-linux.gcc-serial/lib/
12
13 ## Prefetching n_blocks values
14 CONF=1 2 4 8 16 32 64
15
16 Lsize=32
17 Lassoc=8
18 Lbsize=64
19 Lsize=1024
20 Lbsize=1024
21 Lbsize=128
22 TLBsize=64
23 TLBp=4996
24 TLBw=4
25
26 BENCHMARKS="blackscholes bodytrack canneal facesim ferret fluidanimate freqmine rtview streamcluster swaptions"
27
28 for BENCH in $BENCHMARKS; do
29     cmd=$(cat $CMDS_FILE | grep "$BENCH")
30     for conf in $CONF; do
31         ## Get parameters
32         L2prf=$(echo $conf)
33
34         outfile=$(printf "%s.dcache_cslab.PRF_%02d.out" $BENCH ${L2prf})
35         outfile="$outDir/$outfile"
36
37         pin_cmd="PIN_EXE -t $PIN_TOOL -o $outfile -l $Lsize -L $Lassoc -Lb $Lbsize -L2c ${L2size} -L2a ${L2assoc} -L2b ${L2bsize} -TLBe ${TLBe} -TLBp ${TLBp} -TLBa ${TLBa} -L2prf ${L2prf} --"
38         pin_cmd+=" -S $cmd"
39         time spin_cmd
40     done
41 done
```

pref outputs:

```
1#!/usr/bin/env python
2
3 import sys
4 import numpy as np
5
6 ## We need matplotlib:
7 ## $ apt-get install python-matplotlib
8 import matplotlib
9 matplotlib.use('Agg')
10 import matplotlib.pyplot as plt
11
12 x_Axis = []
13 ipc_Axis = []
14 mpkill2_Axis = []
15 mpkill1_Axis = []
16
17 for outFile in sys.argv[1:]:
18     fp = open(outFile)
19     line = fp.readline()
20     while line:
21         tokens = line.split()
22         if (line.startswith("Total Instructions: ")):
23             total_instructions = long(tokens[1])
24         elif (line.startswith("IPC")):
25             ipc = float(tokens[1])
26         elif (line.startswith("L2_prefetching")):
27             l2_prefetching = float(tokens[1])
28         elif (line.startswith("L2_Total_Misses")):
29             l2_total_misses = long(tokens[1])
30             l2_miss_rate = float(tokens[1].split("%")[0])
31             mpkill2 = l2_total_misses / (total_instructions / 1000.0)
32         elif (line.startswith("L1_Total_Misses")):
33             l1_total_misses = long(tokens[1])
34             l1_miss_rate = float(tokens[1].split("%")[0])
35             mpkill1 = l1_total_misses / (total_instructions / 1000.0)
36
37         line = fp.readline()
38
39     fp.close()
40
41     prfConfigStr = '{}'.format(l2_prf)
42     print prfConfigStr
43     x_Axis.append(prfConfigStr)
44     ipc_Axis.append(ipc)
45     mpkill2_Axis.append(mpkill2)
46     mpkill1_Axis.append(mpkill1)
47
48 print x_Axis
49 print ipc_Axis
50 print mpkill2_Axis
51 print mpkill1_Axis
52 print mpkill1_Axis
53
54 fig, ax1 = plt.subplots()
55 ax1.grid(True)
56 ax1.set_xlabel("Prefetching")
57
58 xAx = np.arange(len(x_Axis))
59 ax1.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
60 ax1.set_xticklabels(x_Axis, rotation=45)
61 ax1.set_xlim(-0.5, len(x_Axis) - 0.5)
62 ax1.set_ylim(min(ipc_Axis) - 0.05 * min(ipc_Axis), max(ipc_Axis) + 0.05 * max(ipc_Axis))
63 ax1.set_ylabel("IPC$")
64 line1 = ax1.plot(ipc_Axis, label="ipc", color="red", marker='x')
65
66 ax2 = ax1.twinx()
67 ax2.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
68 ax2.set_xticklabels(x_Axis, rotation=45)
69 ax2.set_xlim(-0.5, len(x_Axis) - 0.5)
70 ax2.set_ylim(min(mpkill2_Axis) - 0.05 * min(mpkill2_Axis), max(mpkill2_Axis) + 0.05 * max(mpkill2_Axis))
71 ax2.set_ylabel("MPK1_L2$")
72 line2 = ax2.plot(mpkill2_Axis, label="L2_mpkill2", color="green", marker='o')
73
74 ax3 = ax1.twinx()
75 ax3.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
76 ax3.set_xticklabels(x_Axis, rotation=45)
77 ax3.set_xlim(-0.5, len(x_Axis) - 0.5)
78 ax3.set_ylim(min(mpkill1_Axis) - 0.05 * min(mpkill1_Axis), max(mpkill1_Axis) + 0.05 * max(mpkill1_Axis))
79 ax3.set_ylabel("MPK1_L1$")
80 line3 = ax3.plot(mpkill1_Axis, label="L1_mpkill1", color="blue", marker='*')
81
82 lns = line1 + line2 + line3
83 labs = [l.get_label() for l in lns]
84
85 plt.title("Prefetching")
86 lgd = plt.legend(lns, labs)
87 lgd.draw_frame(False)
88 plt.savefig("PRF.png", bbox_inches="tight")
```

Mέρος Β:

L1 outputs

```
1 #!/usr/bin/env python
2 from __future__ import division
3 import sys
4 import numpy as np
5 import math
6 L1sizeOLD=32
7 L1assocOLD=4
8 for outFile in sys.argv[1:]:
9     fp = open(outFile)
10    line = fp.readline()
11    while line:
12        tokens = line.split()
13        if (line.startswith("Total Instructions: ")):
14            total_instructions = long(tokens[2])
15        if (line.startswith("Total Cycles: ")):
16            old_cycles = long(tokens[2])
17
18        elif (line.startswith(" L1-Data Cache")):
19            sizeLine = fp.readline()
20            L1sizeNEW = long(sizeLine.split()[1])
21            bsizeLine = fp.readline()
22            l1_bsize = bsizeLine.split()[2]
23            assocLine = fp.readline()
24            L1assocNEW = long(assocLine.split()[1])
25            line = fp.readline()
26
27        sizeDiff=0
28        if(L1sizeNEW/L1sizeOLD!=1):
29            sizeDiff=int(math.log( (L1sizeNEW/L1sizeOLD),2))
30            assocDiff=0
31        if (L1assocNEW/L1assocOLD!=1):
32            assocDiff=int(math.log( (L1assocNEW/L1assocOLD),2))
33
34        forcesPanw=1
35        for x in range(sizeDiff):
36            forcesPanw=forcesPanw*1.1
37        for x in range(assocDiff):
38            forcesPanw=forcesPanw*1.05
39
40        newCycle = long(old_cycles*forcesPanw)
41        fp.close()
42
43        icp = total_instructions / newCycle
44        print(icp)
45
46
47        f = open(outFile,"r+")
48        line = f.readline()
49        while line:
50            if (line.startswith("L2-Total-Accesses: ")):
51                f.write("NEWIPC: ")
52                f.write(str(icp))
53                #print(total_instructions/newCycle)
54            line = f.readline()
```

L1 plots:

```
1  #!/usr/bin/env python
2
3  import sys
4  import numpy as np
5
6  ## We need matplotlib:
7  ## $ apt-get install python-matplotlib
8  import matplotlib
9  matplotlib.use('Agg')
10 import matplotlib.pyplot as plt
11
12 x_Axis = []
13 ipc_Axis = []
14 mpki_Axis = []
15
16 for outFile in sys.argv[1:]:
17     fp = open(outFile)
18     line = fp.readline()
19     while line:
20         tokens = line.split()
21         if (line.startswith("Total Instructions: ")):
22             total_instructions = long(tokens[2])
23         elif (line.startswith("NEWIPC: ")):
24             ipc = float(tokens[1])
25         elif (line.startswith(" L1-Data Cache")):
26             sizeLine = fp.readline()
27             l1_size = sizeLine.split()[1]
28             bsizeLine = fp.readline()
29             l1_bsize = bsizeLine.split()[2]
30             assocLine = fp.readline()
31             l1_assoc = assocLine.split()[1]
32         elif (line.startswith(" L1-Total-Misses")):
33             l1_total_misses = long(tokens[1])
34             l1_miss_rate = float(tokens[2].split("%")[0])
35             mpki = l1_total_misses / (total_instructions / 1000.0)
36
37         line = fp.readline()
38
39     fp.close()
40
41     l1ConfigStr = '{}K.{}.{}B'.format(l1_size,l1_assoc,l1_bsize)
42     print l1ConfigStr
43     x_Axis.append(l1ConfigStr)
44     ipc_Axis.append(ipc)
45     mpki_Axis.append(mpki)
46
47 print x_Axis
48 print ipc_Axis
49 print mpki_Axis
50
51 fig, ax1 = plt.subplots()
52 ax1.grid(True)
53 ax1.set_xlabel("CacheSize.Assoc.BlockSize")
54
55 xAx = np.arange(len(x_Axis))
56 ax1.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
57 ax1.set_xticklabels(x_Axis, rotation=45)
58 ax1.set_xlim(-0.5, len(x_Axis) - 0.5)
59 ax1.set_ylim(min(ipc_Axis) - 0.05 * min(ipc_Axis), max(ipc_Axis) + 0.05 * max(ipc_Axis))
60 ax1.set_ylabel("$IPC$")
61 line1 = ax1.plot(ipc_Axis, label="ipc", color="red", marker='x')
62
63 ax2 = ax1.twinx()
64 ax2.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
65 ax2.set_xticklabels(x_Axis, rotation=45)
66 ax2.set_xlim(-0.5, len(x_Axis) - 0.5)
67 ax2.set_ylim(min(mpki_Axis) - 0.05 * min(mpki_Axis), max(mpki_Axis) + 0.05 * max(mpki_Axis))
68 ax2.set_ylabel("$MPKI$")
69 line2 = ax2.plot(mpki_Axis, label="L1D_mpki", color="green", marker='o')
70
71 lns = line1 + line2
72 labs = [l.get_label() for l in lns]
73
74 plt.title("IPC vs MPKI")
75 lgd = plt.legend(lns, labs)
76 lgd.draw_frame(False)
77 plt.savefig("L1.png", bbox_inches="tight")
```

L2 outputs:

```
1 #!/usr/bin/env python
2 from __future__ import division
3 import sys
4 import numpy as np
5 import math
6 L2sizeOLD=512
7 L2assocOLD=8
8 for outFile in sys.argv[1:]:
9     fp = open(outFile)
10    line = fp.readline()
11    while line:
12        tokens = line.split()
13        if (line.startswith("Total Instructions: ")):
14            total_instructions = long(tokens[2])
15        if (line.startswith("Total Cycles: ")):
16            old_cycles = long(tokens[2])
17
18        elif (line.startswith(" L2-Data Cache")):
19            sizeLine = fp.readline()
20            L2sizeNEW = long(sizeLine.split()[1])
21            bsizeLine = fp.readline()
22            l2_bsize = bsizeLine.split()[2]
23            assocLine = fp.readline()
24            L2assocNEW = long(assocLine.split()[1])
25            line = fp.readline()
26
27        sizeDif=0
28        if(L2sizeNEW/L2sizeOLD!=1):
29            sizeDif=int(math.log( (L2sizeNEW/L2sizeOLD),2))
30        assocDif=0
31        if (L2assocNEW/L2assocOLD!=1):
32            assocDif=int(math.log( (L2assocNEW/L2assocOLD),2))
33
34        foressPanw=1
35        for x in range(sizeDif):
36            foressPanw=foressPanw*1.1
37        for x in range(assocDif):
38            foressPanw=foressPanw*1.05
39
40        newCycle = long(old_cycles*foressPanw)
41        fp.close()
42
43        icp = total_instructions / newCycle
44        print(icp)
45
46
47        f = open(outFile,"r+")
48        line = f.readline()
49        while line:
50            if (line.startswith("L2-Total-Accesses: ")):
51                f.write("NEWIPC: ")
52                f.write(str(icp))
53                #print(total_instructions/newCycle)
54            line = f.readline()
```

L2 plots:

```
1  #!/usr/bin/env python
2
3  import sys
4  import numpy as np
5
6  ## We need matplotlib:
7  ## $ apt-get install python-matplotlib
8  import matplotlib
9  matplotlib.use('Agg')
10 import matplotlib.pyplot as plt
11
12 x_Axis = []
13 ipc_Axis = []
14 mpki_Axis = []
15
16 for outFile in sys.argv[1:]:
17     fp = open(outFile)
18     line = fp.readline()
19     while line:
20         tokens = line.split()
21         if (line.startswith("Total Instructions: ")):
22             total_instructions = long(tokens[2])
23         elif (line.startswith("NEWIPC: ")):
24             ipc = float(tokens[1])
25         elif (line.startswith(" L2-Data Cache")):
26             sizeLine = fp.readline()
27             L2_size = sizeLine.split()[1]
28             bsizeLine = fp.readline()
29             L2_bsize = bsizeLine.split()[2]
30             assocLine = fp.readline()
31             L2_assoc = assocLine.split()[1]
32         elif (line.startswith("L2-Total-Misses")):
33             L2_total_misses = long(tokens[1])
34             L2_miss_rate = float(tokens[2].split('%')[0])
35             mpki = L2_total_misses / (total_instructions / 1000.0)
36
37         line = fp.readline()
38
39     fp.close()
40
41     L2ConfigStr = '{}K.{}.{}B'.format(L2_size,L2_assoc,L2_bsize)
42     print L2ConfigStr
43     x_Axis.append(L2ConfigStr)
44     ipc_Axis.append(ipc)
45     mpki_Axis.append(mpki)
46
47
48 print x_Axis
49 print ipc_Axis
50 print mpki_Axis
51
52 fig, ax1 = plt.subplots()
53 ax1.grid(True)
54 ax1.set_xlabel("CacheSize.Assoc.BlockSize")
55
56 xAx = np.arange(len(x_Axis))
57 ax1.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
58 ax1.set_xticklabels(x_Axis, rotation=45)
59 ax1.set_xlim(-0.5, len(x_Axis) - 0.5)
60 ax1.set_ylim(min(ipc_Axis) - 0.05 * min(ipc_Axis), max(ipc_Axis) + 0.05 * max(ipc_Axis))
61 ax1.set_ylabel("$IPCS$")
62 line1 = ax1.plot(ipc_Axis, label="ipc", color="red", marker='x')
63
64 ax2 = ax1.twinx()
65 ax2.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
66 ax2.set_xticklabels(x_Axis, rotation=45)
67 ax2.set_xlim(-0.5, len(x_Axis) - 0.5)
68 ax2.set_ylim(min(mpki_Axis) - 0.05 * min(mpki_Axis), max(mpki_Axis) + 0.05 * max(mpki_Axis))
69 ax2.set_ylabel("$MPKI$")
70 line2 = ax2.plot(mpki_Axis, label="L2_mpki", color="green", marker='o')
71
72 lns = line1 + line2
73 labs = [l.get_label() for l in lns]
74
75 plt.title("IPC vs MPKI")
76 lgd = plt.legend(lns, labs)
77 lgd.draw_frame(False)
78 plt.savefig("L2.png", bbox_inches="tight")
```

TLB outputs:

```
1 #!/usr/bin/env python
2 from __future__ import division
3 import sys
4 import numpy as np
5 import math
6
7 TLBsizeOLD=8
8 TLBassocOLD=4
9
10 for outFile in sys.argv[1:]:
11     fp = open(outFile)
12     line = fp.readline()
13     while line:
14         tokens = line.split()
15         if (line.startswith("Total Instructions: ")):
16             total_instructions = long(tokens[2])
17         if (line.startswith("Total Cycles: ")):
18             old_cycles = long(tokens[2])
19
20         elif (line.startswith(" Data Tlb")):
21             sizeLine = fp.readline()
22                 TLBsizeNEW = long(sizeLine.split()[1])
23             bsizeLine = fp.readline()
24                 tlb_psize = bsizeLine.split()[2]
25             assocLine = fp.readline()
26                 TLBassocNEW = long(assocLine.split()[1])
27             line = fp.readline()
28
29         sizeDif=0
30         if(TLBsizeNEW/TLBsizeOLD!=1):
31             sizeDif=int(math.log( (TLBsizeNEW/TLBsizeOLD),2))
32         assocDif=0
33         if (TLBassocNEW/TLBassocOLD!=1):
34             assocDif=int(math.log( (TLBassocNEW/TLBassocOLD),2))
35
36         foresPanw=1
37         for x in range(sizeDif):
38             foresPanw=foresPanw*1.1
39         for x in range(assocDif):
40             foresPanw=foresPanw*1.05
41
42         newCycle = long(old_cycles*foresPanw)
43         fp.close()
44
45         icp = total_instructions / newCycle
46
47         f = open(outFile,"r+")
48         line = f.readline()
49         while line:
50             if (line.startswith("L2-Total-Accesses: ")):
51                 f.write("NEWIPC: ")
52                 f.write(str(icp))
53             line = f.readline()
```

TLB Plots:

```
1 #!/usr/bin/env python
2
3 import sys
4 import numpy as np
5 import matplotlib
6 matplotlib.use('Agg')
7 import matplotlib.pyplot as plt
8
9 x_Axis = []
10 ipc_Axis = []
11 mpki_Axis = []
12
13 for outFile in sys.argv[1:]:
14     fp = open(outFile)
15     line = fp.readline()
16     while line:
17         tokens = line.split()
18         if (line.startswith("Total Instructions: ")):
19             total_instructions = long(tokens[2])
20         elif (line.startswith("NEWIPC: ")):
21             ipc = float(tokens[1])
22         elif (line.startswith(" Data Tlb")):
23             sizeLine = fp.readline()
24             tlb_entries = sizeLine.split()[1]
25             bsizeLine = fp.readline()
26             tlb_psize = bsizeLine.split()[2]
27             assocLine = fp.readline()
28             tlb_assoc = assocLine.split()[1]
29         elif (line.startswith("Tlb-Total-Misses")):
30             tlb_total_misses = long(tokens[1])
31             tlb_miss_rate = float(tokens[2].split('%')[0])
32             mpki = tlb_total_misses / (total_instructions / 1000.0)
33
34         line = fp.readline()
35
36     fp.close()
37
38     tlbConfigStr = '{}.{}.{}B'.format(tlb_entries, tlb_assoc, tlb_psize)
39     print(tlbConfigStr)
40     x_Axis.append(tlbConfigStr)
41     ipc_Axis.append(ipc)
42     mpki_Axis.append(mpki)
43
44 print(x_Axis)
45 print(ipc_Axis)
46 print(mpki_Axis)
47
48 fig, ax1 = plt.subplots()
49 ax1.grid(True)
50 ax1.set_xlabel("TLBSize.Assoc.PageSize")
51
52 xAx = np.arange(len(x_Axis))
53 ax1.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
54 ax1.set_xticklabels(x_Axis, rotation=45)
55 ax1.set_xlim(-0.5, len(x_Axis) - 0.5)
56 ax1.set_ylim(min(ipc_Axis) - 0.05 * min(ipc_Axis), max(ipc_Axis) + 0.05 * max(ipc_Axis))
57 ax1.set_ylabel("$IPC$")
58 line1 = ax1.plot(ipc_Axis, label="ipc", color="red", marker='x')
59
60 ax2 = ax1.twinx()
61 ax2.xaxis.set_ticks(np.arange(0, len(x_Axis), 1))
62 ax2.set_xticklabels(x_Axis, rotation=45)
63 ax2.set_xlim(-0.5, len(x_Axis) - 0.5)
64 ax2.set_ylim(min(mpki_Axis) - 0.05 * min(mpki_Axis), max(mpki_Axis) + 0.05 * max(mpki_Axis))
65 ax2.set_ylabel("$MPKI$")
66 line2 = ax2.plot(mpki_Axis, label="TLB_mpki", color="green", marker='o')
67
68 lns = line1 + line2
69 labs = [l.get_label() for l in lns]
70
71 plt.title("IPC vs MPKI")
72 lgd = plt.legend(lns, labs)
73 lgd.draw_frame(False)
74 plt.savefig("TLB.png", bbox_inches="tight")
```

19. Βιβλιογραφία

Για τη συγγραφή της παραπάνω εργασίας χρησιμοποιήθηκαν οι εξής πηγές:

- Computer Architecture: A Quantitative Approach 5th Edition by John L. Hennessy(Author),David A. Patterson (Author)
- [https://en.wikipedia.org/wiki/Cache_\(computing\)](https://en.wikipedia.org/wiki/Cache_(computing))
- <http://parsec.cs.princeton.edu>
- <https://software.intel.com/en-us/articles/pin-a-dynamic-binary- instrumentation-tool>
- <http://www.cslab.ntua.gr/courses/advcomparch/notes.go>