1η ΑΣΚΗΣΗ ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ
Ακ. έτος 2018-2019, 8ο Εξάμηνο, Σχολή ΗΜ&ΜΥ

Ονομ/νυμο: Χαρδούβελης Γεώργιος-Ορέστης
Α.Μ: 03115100
Εξάμηνο: 8ο

PIN

Στα πλαίσια της παρούσας άσκησης χρησιμοποιήσαμε το εργαλείο “PIN” για να μελετηθεί η επίδραση διαφόρων παραμέτρων της ιεραρχίας μνήμης στην απόδοση ενός συνόλου εφαρμογών. Παράλληλα παράχθηκαν πειραματικές μετρήσεις με σκοπό την εξαγωγή χρήσιμων συμπερασμάτων.
Συγκεκριμένα μετήσαμε πως επηρεάζει το cache memory διάφορα μετροπρογράμματα / benchmarks. Στη συνέχεια μελετήθηκε πώς οι μετρικές απόδοσης μεταβάλλονται στον χρόνο.

PARSEC BENCHMARKS

Στα πλαίσια της παρούσας άσκησης χρησιμοποιήσαμε 10 από τα 13 PARSEC benchmarks που προσέφερε η σουίτα που αναφερόταν στην εκφώνηση. Συγκεκριμένα χρησιμοποιήσαμε τα εξής:
1. blackschole
2. bodytrack
3. canneal
4. facesim
5. ferret
6. fluidanimate
7. freqmine
8. raytrace
9.streamcluster
10 .swaptions

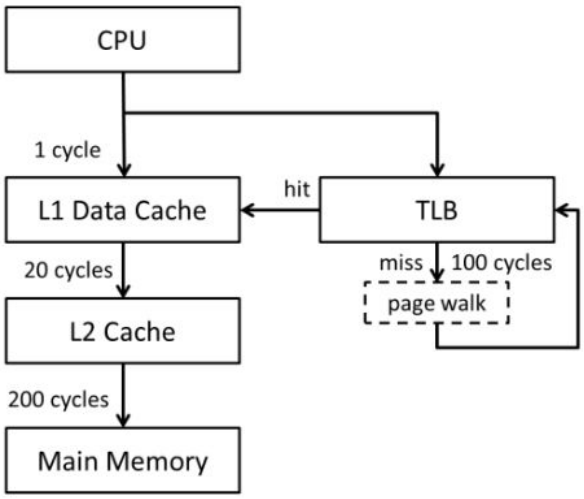
Στον κώδικα των μετροπρογραμμάτων έχουν οριστεί συγκεκριμένες περιοχές του κώδικα οι οποίες παρουσιάζουν μεγαλύτερο ενδιαφέρον προς μελέτη (Regions of Interest, ROI). Τέλος, για την προσομοίωση της εκτέλεσης των εφαρμογών χρησιμοποιήθηκε το pintool simulator.cpp, το οποίο είναι σχεδιασμένο έτσι ώστε να ενεργοποιείται μόνο κατά τη διάρκεια των ROI.

SIMULATION

Για την προσομοίωση εκτέλεσης των παραπάνω εφαρμογών χρησιμοποιήθηκε το αρχείο pintool simulator.cpp που προσομοιώνει σε ένα περιβάλλον με έναν in-order επεξεργαστή την κρυφή μνήμη δύο επιπέδων (L1-Data + L2 caches), και την μνήμη

μετάφρασης διευθύνσεων (TLB). Το simulator.cpp είναι γραμμένο έτσι ώστε να ενεργοποιείται μόνο κατά την διάρκεια των PARSEC ROI[1].

Το simulator pintool προσομοιώνει έναν in-order επεξεργαστή με την inclusive ιεραρχία μνήμης και μετάφραση διευθύνσεων που φαίνεται στο παρακάτω σχήμα. 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 (είτε load είτε store) προκαλούν επιπλέον καθυστερήσεις ανάλογα με το αν οι μεταφράσεις διευθύνσεων βρίσκονται στο TLB και με το πού βρίσκονται τα δεδομένα τους. Θεωρούμε ότι η L1 είναι υλοποιημένη ως Virtually indexed, Physically tagged (VIPT)cache, δηλαδή οι προσβάσεις στο TLB και στην L1cache επικαλύπτονται και πραγματοποιούνται παράλληλα. Αν η ζητούμενη μετάφραση δεν βρίσκεται στο TLB (TLBmiss), τότε εισάγεται μία καθυστέρηση 100 κύκλων που προσομοιώνει την καθυστέρηση ανάκλησης της μετάφρασης διεύθυνσης από την μνήμη (pagewalk), και στη συνέχεια εκτελείται η πρόσβαση στην ιεραρχία μνήμης μέσω της L1 cache.



Το TLB module του simulator χρησιμοποιείται μόνο για την μελέτη της επίδρασης της μετάφρασης διευθύνσεων στην επίδοση των προγραμμάτων όσον αφορά τον χρόνο εκτέλεσης, και όχι για να παρέχει πραγματική μετάφραση διευθύνσεων στον φυσικό χώρο. Δηλαδή, κατά την προσομοίωση το cache module χρησιμοποιεί εικονικές

[1] ROI: Region Of Interest - Η περιοχές των μετροπρογραμμάτων που έχουν οριστεί από εμάς και παρουσιάζουν μεγαλύτερο ενδιαφέρον για μελέτη. Ο simulator.cpp ενεργοποιείται μόνο σε εκείνες τις περιοχές ώστε να εξοικονομήσουμε χρόνο και να μην αναλωνόμαστε σε περιοχές ελάσσονους ενδιαφέροντος.

διευθύνσεις για την πρόσβαση στην μνήμη δεδομένων και τον έλεγχο των cache hits/misses.

Συνολικά, ο αριθμός των κύκλων υπολογίζεται ως:

Cycles = Inst + TLB_Misses*TLB_miss_cycles + L1_Accesses * L1_hit_cycles +
L2_Accesses * L2_hit_cycles + Mem_Accesses * Mem_acc_cycles

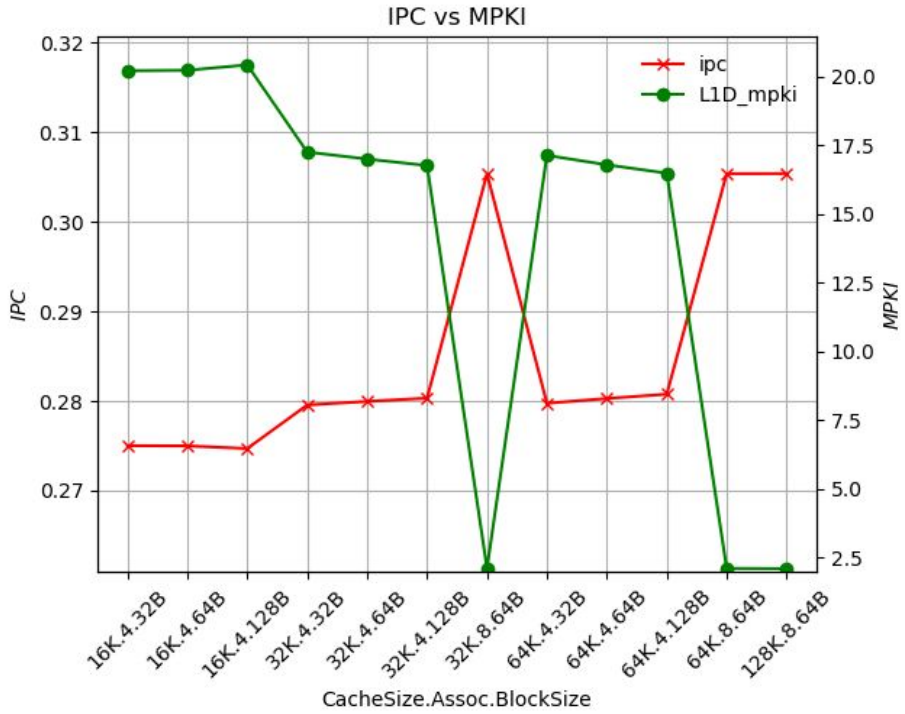
7. Πειραματική αξιολόγηση

7.1.5 Ζητούμενο

L1 Cache

Εδώ διατηρήσαμε τις τιμές της L2 Cache και TLB σταθερές και ίσες με τις τιμές που δίνονται στην εκφώνηση και μεταβάλλαμε τις παραμέτρους της L1 στις τιμές που φαίνονται και στον x άξονα των παρακάτω διαγραμμάτων.
Παρακάτω βλέπουμε όλα τα αποτελέσματα για τα προαναφερθέντα benchmarks.

1. Blackschole

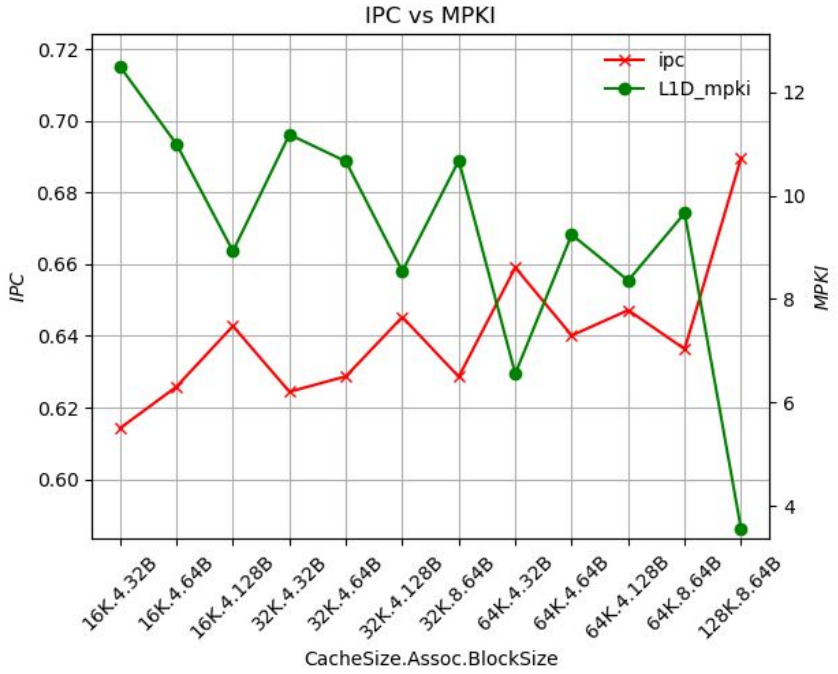


Όσον αφορά το IPC (instructions per cycle) το associativity παίζει μείζονα ρόλο. Συγκεκριμένα όταν γίνεται 8 από 4 (άρα έχουμε λιγότερα conflist misses) υπάρχει πολύ

μεγάλη αύξηση. Με μέγεθος μνήμης 16KB έχουμε μικρότερη απόδοση αλλά μετά τα 32KB η αύξηση της μνήμης δεν έχει αντίκτυπο, όπως και το μέγεθος των blocks.

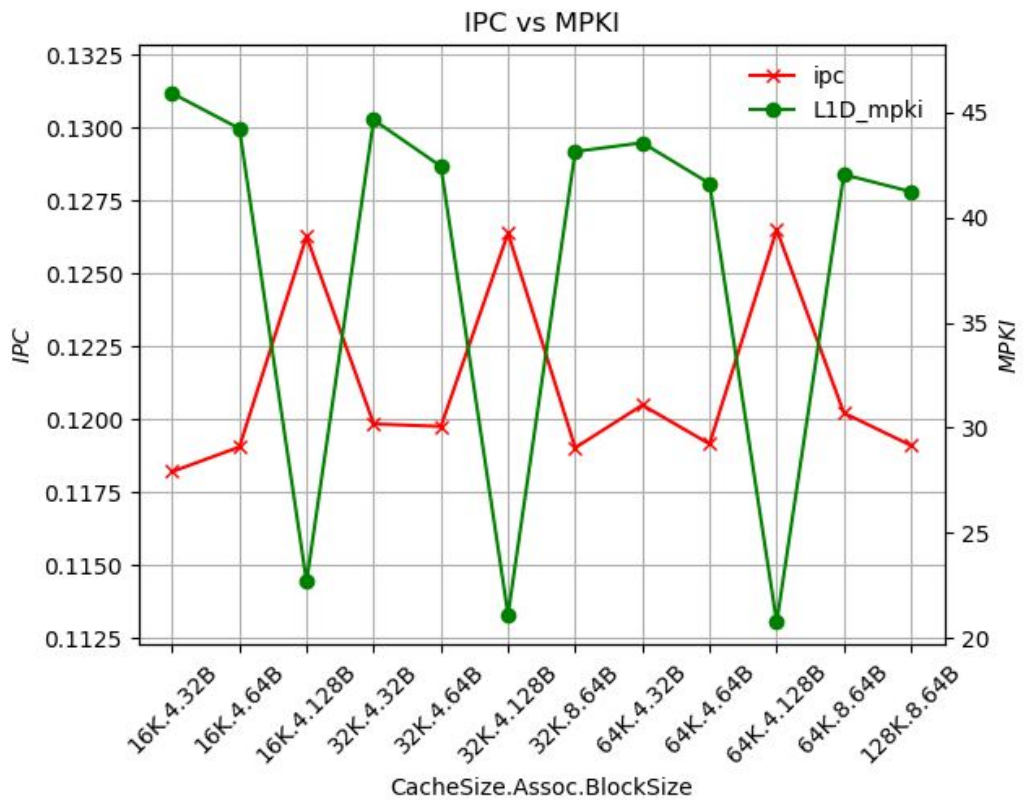
Αντίστοιχα και τα mpki (misses per kilo-instructions) για μνήμη ίση με 16KB έχουμε παραπάνω misses από τα άλλα μεγέθη ενώ τον σημαντικότερο ρόλο παίζει πάλι το associativity όπου λαμβάνει την μικρότερη τιμή του όταν γίνεται 8.

2. Bodytrack



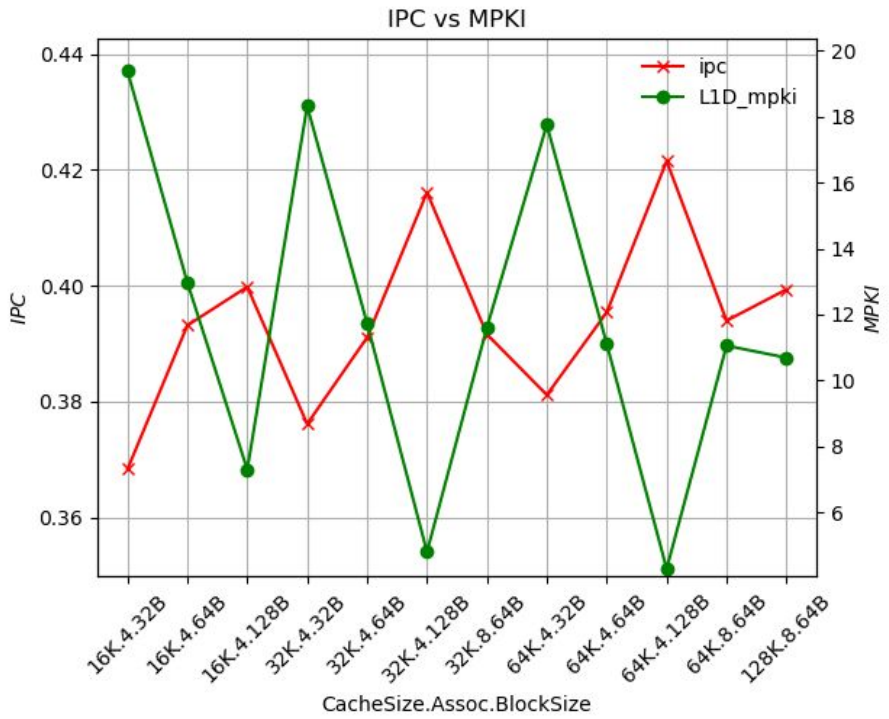
Εδώ βλέπουμε πως το associativity δεν επηρεάζει καθόλου την απόδοση της cache. Σε Σε γενικές γραμμές, όσο αυξάνεται το μέγεθος του block τόσο υπάρχει μικρή αύξηση της απόδοσης.
Πολύ μεγάλη αύξηση της απόδοσης (τόσο στο ipc όσο και στο mpki) έχουμε όταν αυξάνουμε το μέγεθος της μνήμης σε 128 KB.

3. Canneal



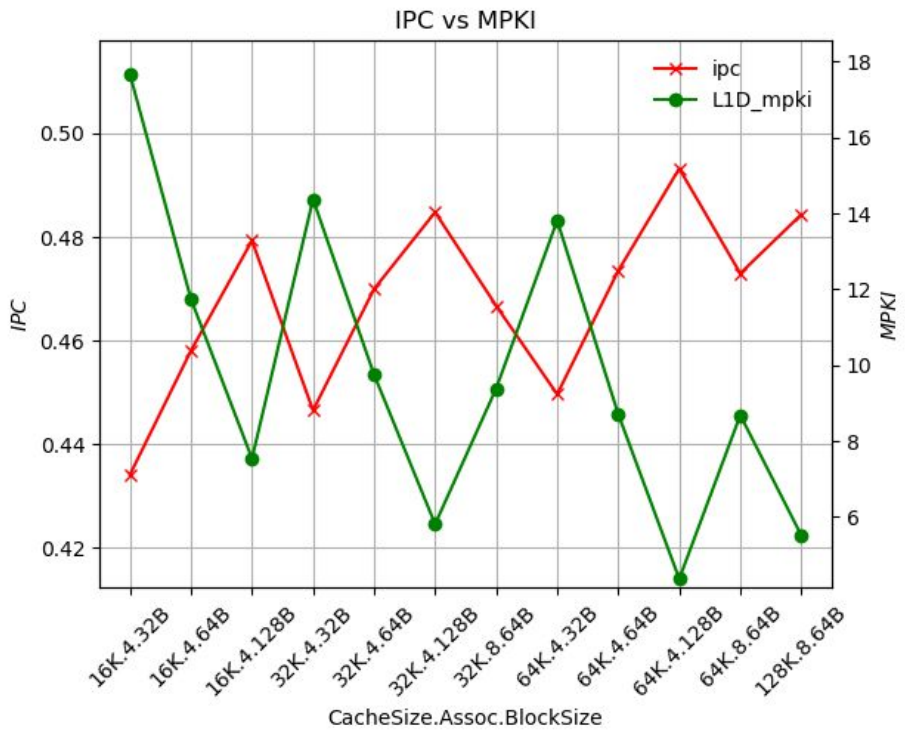
Εδώ παρατηρούμε πως μεγαλύτερη επίδραση της απόδοση έχει το block size. Συγκεκριμένα όταν αυξάνεται από 32 σε 64 υπάρχει μια μικρή βελτίωση, που κορυφώνεται όταν την αυξήσουμε στα 128B.

4. Facesim



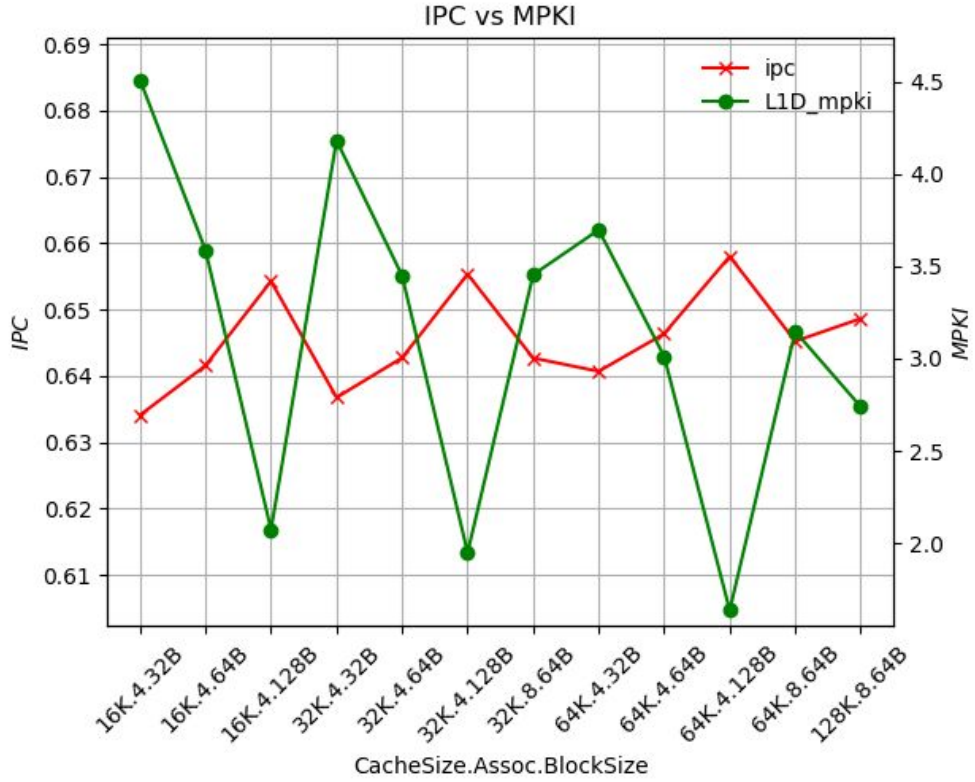
Ομοίως με το προηγούμενο benchmark, μεγαλύτερη επίδραση της απόδοση έχει το block size. Αντίθετα βέβαια με πριν, μεγαλύτερη βελτίωση παρατηρείται όταν αυξήσουμε στα 64 από τα 23 παρά στα 128 από τα 64B.
Μικρό ρόλο παίζει και το μέγεθος της cache καθώς όσο αυξάνεται υπάρχει βελτίωση της απόδοσης. Όσο μεγαλώνει το μέγεθος τόσο μικρότερη βελτίωση παρατηρείται

5. Ferret



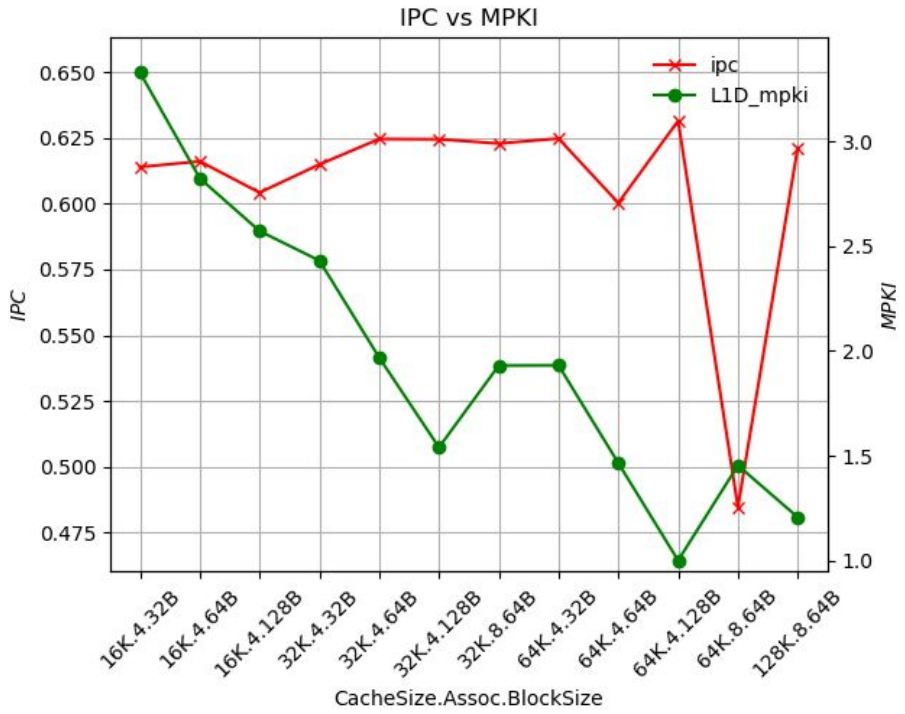
Ομοίως με Facesim

6. Fluidanimate



Ομοίως με τα προηγούμενα benchmark, μεγαλύτερη επίδραση της απόδοση έχει το block size. Μικρό ρόλο (μικρότερο από πριν) παίζει και το μέγεθος της cache.

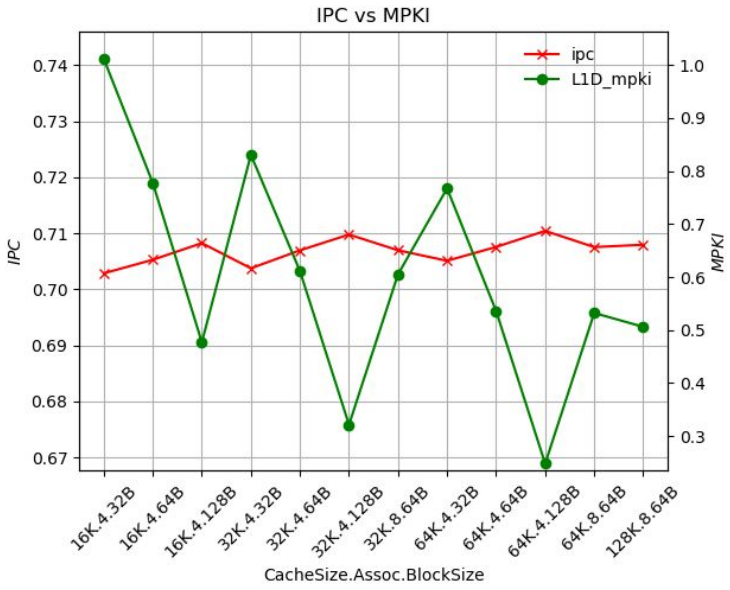
7. Freqmine



Το mpki επηρεάζεται άμεσα και από το μέγεθος του block size καθώς και από το μέγεθος της cache όπου αυξάνεται αφού μειώνονται τα capacity και compulsory misses, με εξαίρεση block size 128 όπου συχνά υπάρχει μια μικρή αύξηση των misses.

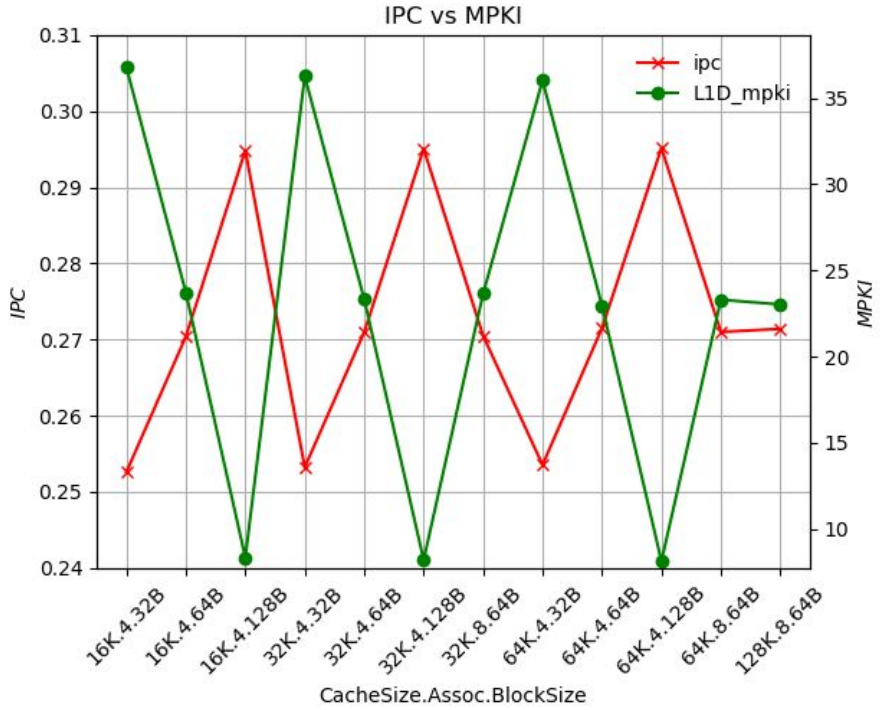
Το ipc στις πρώτες περιπτώσεις παρουσιάζει μικρές αλλαγές με βελτίωση όταν αυξάνεται το μέγεθος της cache στα 32KB. Αντίθετα, όταν πάμε στα 64KB ενώ με 32B block size παραμένει σταθερό, στα 64 υπάρχει μεγάλη πτώση της απόδοσης που αποκαθιστάται όταν αυξήσουμε είτε το μέγεθος της cache, είτε το μέγεθος των block.

8. Raytrace



Η μεταβολή του mpki είναι όμοια με εκείνη του Ferret.
Αντίθετα, δεδομένου της μείωσης του αριθμού των misses, το ipc παραμένει σε μικρές τιμές και εξαρτάται από το μέγεθος της cache και το μέγεθος των blocks.

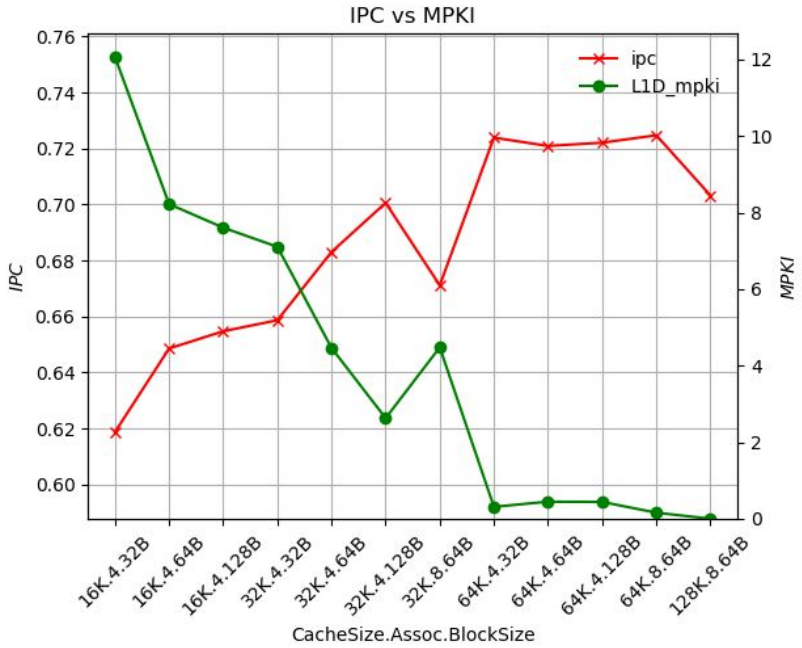
9. Streamcluster



Όπως και στα Ferret και Freqmine, μεγαλύτερη επίδραση της απόδοση έχει το block size ενώ οι άλλες τιμές δεν επηρεάζουν καθόλου. Στη συγκεκριμένη περίπτωση που

όπως βλέπουμε έχουμε πολύ μεγαλύτερο αριθμό misses, οι διακυμάνσεις είναι σαφώς μεγαλύτερες.

10. Swaptions



Εδώ, μέχρι τα 64KB μέγεθος cache υπάρχει μεγαλύτερη βελτίωση όσο αυξάνουμε τόσο το μέγεθος της cache όσο και το blocksize. Από τα 64 και μετά υπάρχει σχεδόν σταθερή βελτίωση. Σε κάθε περίπτωση η αύξηση του associativity μειώνει συνήθως ελαφρώς την απόδοση.

Συμπεράσματα

Συμπερασματικά έχουμε πως αρκετά benchmarks (canneal, facism, ferret, fluidanimate, streamcluster) επηρεάζονται κατά βάση από το block size.

Το associativity παίζει βασικό ρόλο σε μετροπρογράμματα όπως το blacksholes ενώ το μέγεθος της μνήμης σε μετροπρογράμματα όπως το bodytrack και το swaptions.

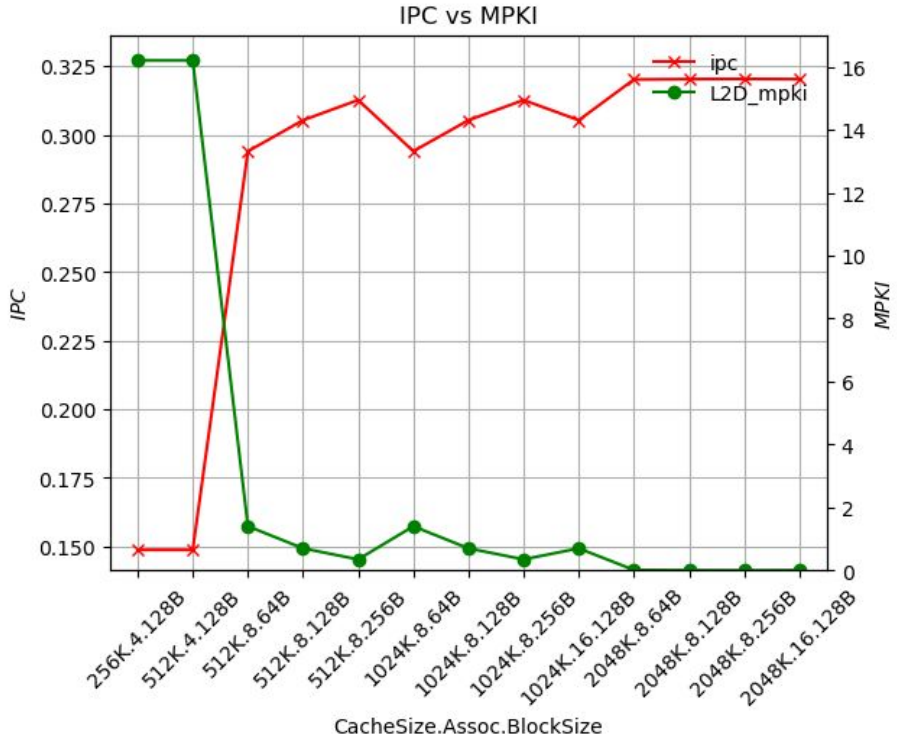
Τέλος, τα freqmine και raytrace δεν επηρεάζονταν έντονα από μεταβολές στην L1 μνήμη.

Συνολικά, μια καλή επιλογή μνήμης που δίνει καλά αποτελέσματα τις περισσότερες φορές θα ήταν με μέγεθος 64KB, 4 way associative και block size 128B. Ικανοποιητικά αποτελέσματα δίνει και η L1 με μέγεθος 128KB, 8 way associative και block size 64B. Από περισσότερο οικονομικές μνήμες, εξίσου ικανοποιητική είναι εκείνη με μέγεθος 32KB, 4 way associative και block size 128B.

L2 Cache

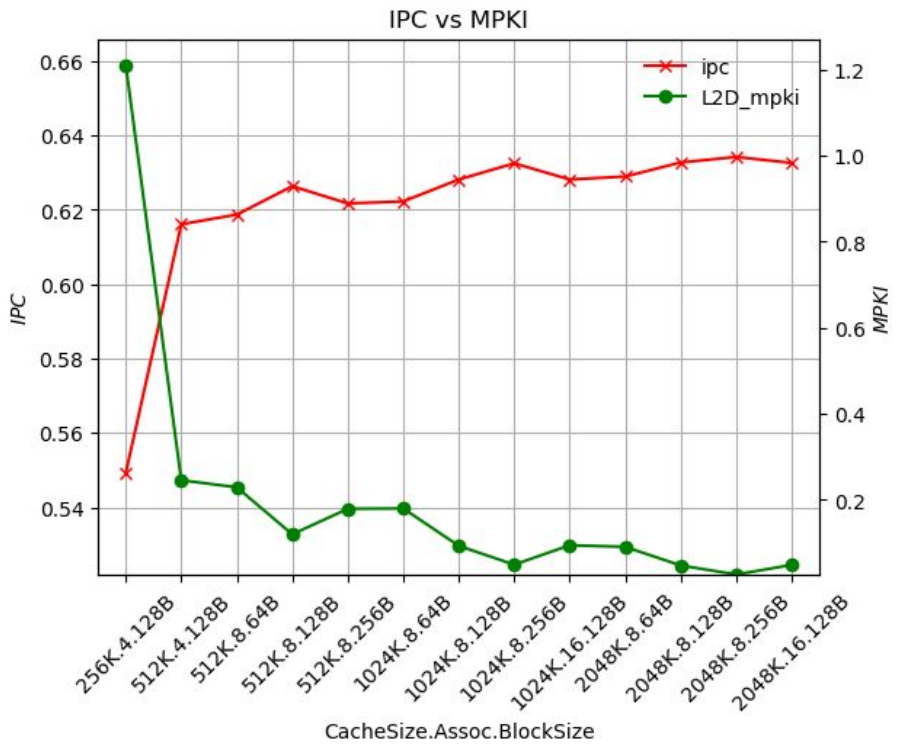
Εδώ διατηρήσαμε τις τιμές της L1 Cache και TLB σταθερές και ίσες με τις τιμές που δίνονται στην εκφώνηση και μεταβάλλαμε τις παραμέτρους της L2 στις τιμές που φαίνονται και στον x άξονα των παρακάτω διαγραμμάτων.

1. Blackschole



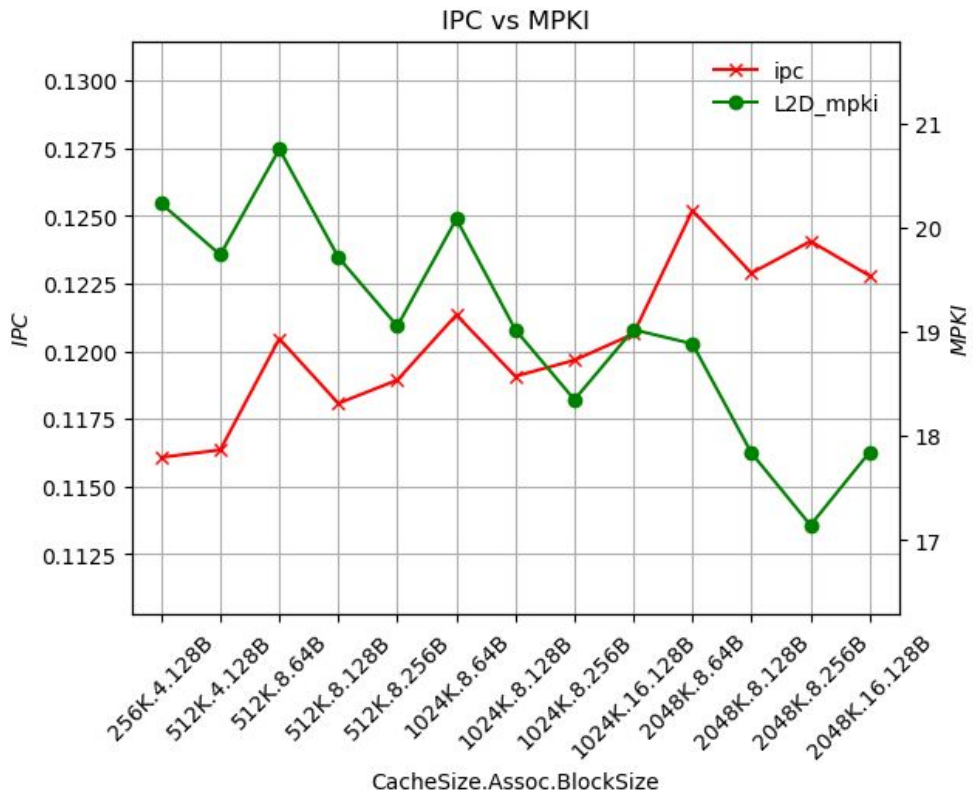
Ο κύριος παράγοντας που επηρεάζει την απόδοση είναι το associativity. Για associativity ίσο με 4 έχουμε πολύ μικρή απόδοση. Μικρό ρόλο παίζει και το μέγεθος των blocks (βελτίωση της απόδοσης όσο αυξάνεται).

2. Bodytrack



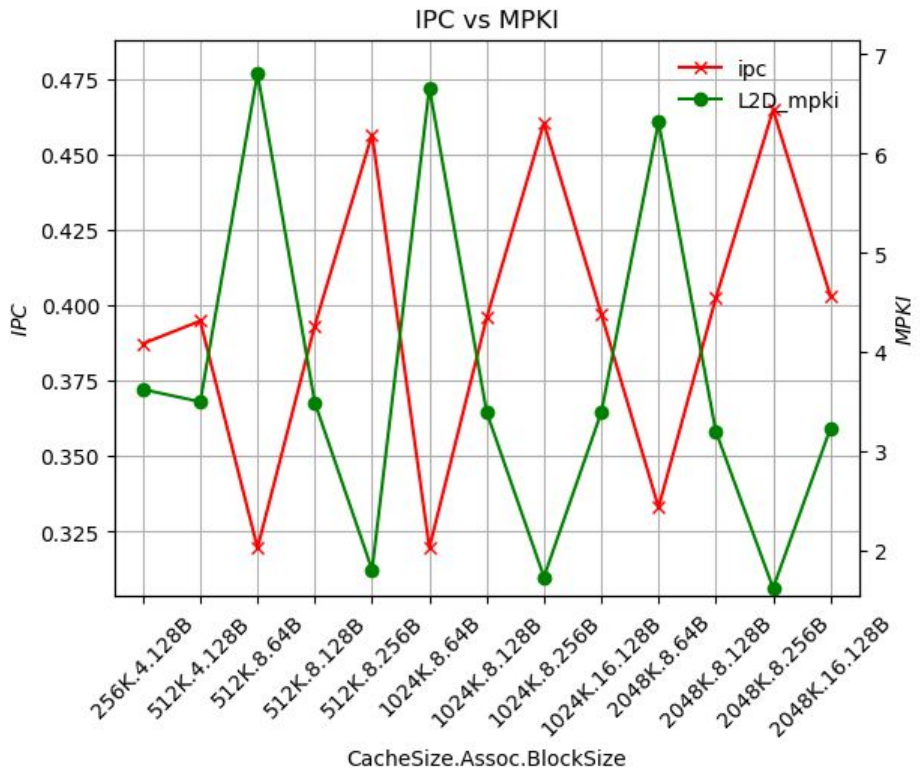
Εδώ σημαντικό ρόλο παίζει το μέγεθος της μνήμης. Για μνήμη 256K έχουμε πολύ μικρή απόδοση ενώ βελτιώνεται με μεγαλύτερη μνήμη.
Από associativity 4 σε 8 υπάρχει μια μικρή αύξηση της απόδοσης ενώ από 8 σε 16 μια μικρή μείωση.

3. Canneal



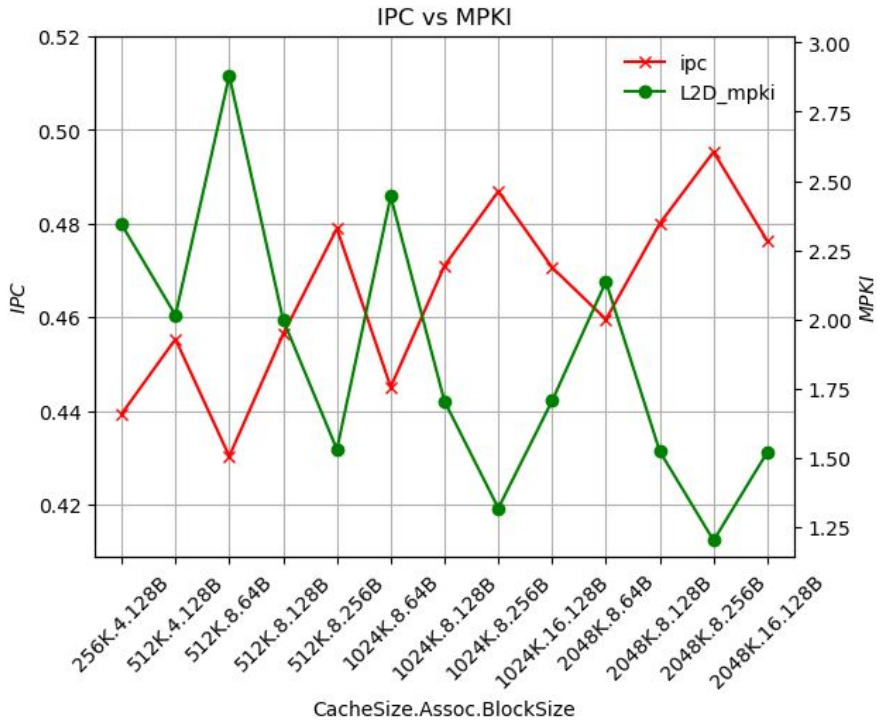
Εδώ όσο αυξάνεται το block size καθώς και το μέγεθος της cache σε γενικές γραμμές μειώνεται το mpki αλλά συχνά μειώνεται και το ipc μαζί. Καλά αποτελέσματα έχουμε για μέγεθος μνήμης 2048KB. Μάλλον ο επικρατέστερος παράγοντας είναι το μέγεθος της μνήμης.

4. Facesim



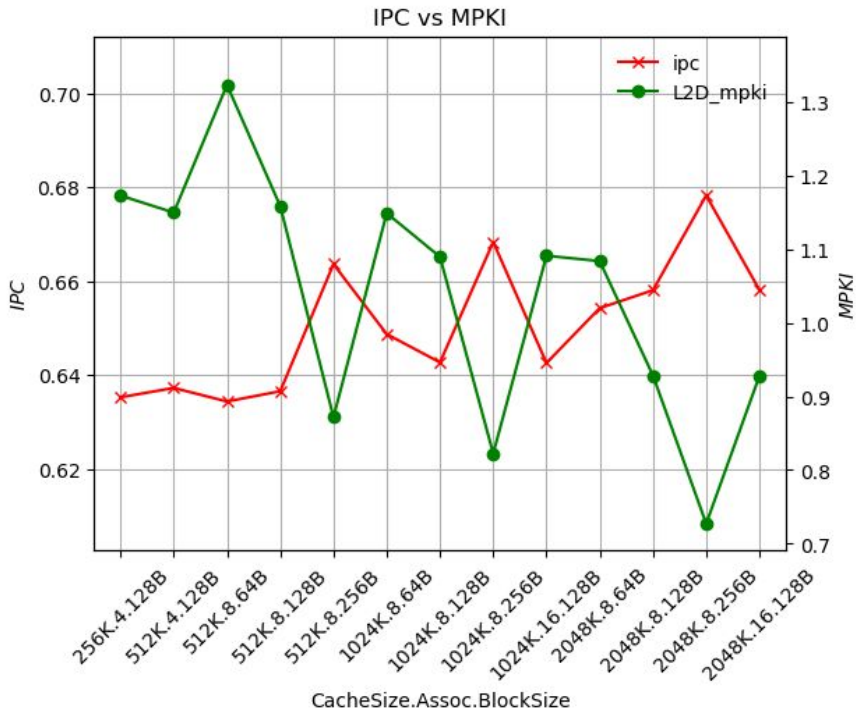
Εδώ ειναι σαφές πως η κυριότερη παράμετρος είναι το block size. Όσο αυξάνεται έχουμε αύξηση και της απόδοσης. Το μέγεθος της cache βελτιώνει ελαφρώς την απόδοση.

5. Ferret



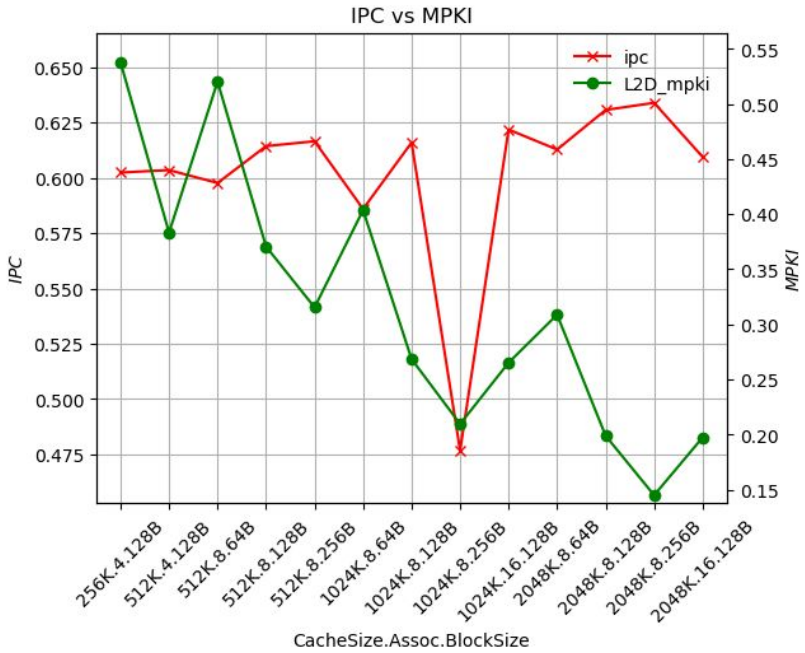
Ομοίως με πριν, η κυριότερη παράμετρος είναι το block size. Όσο αυξάνεται έχουμε αύξηση και της απόδοσης. Ταυτόχρονα, το μέγεθος της cache έχεχι σημαντικότερη επιρροή σε σχέση με πριν.

6. Fluidanimate



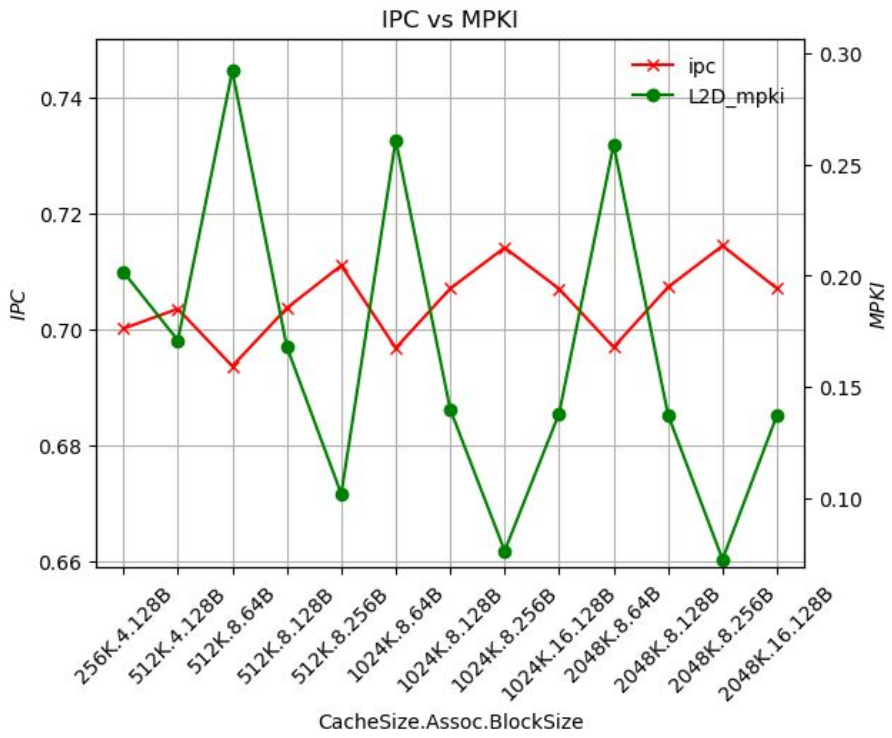
Ομοίως με πριν, η κυριότερη παράμετρος είναι το block size. Όσο αυξάνεται έχουμε αύξηση και της απόδοσης. Η διαφορά είναι πως από block size 64B σε 128B υπάρχει μικρή διαφορά. Ακόμη μιας και έχουμε μικρότερο αριθμό misses η διακύμανση του ipc είναι μικρότερη.

7. Freqmine



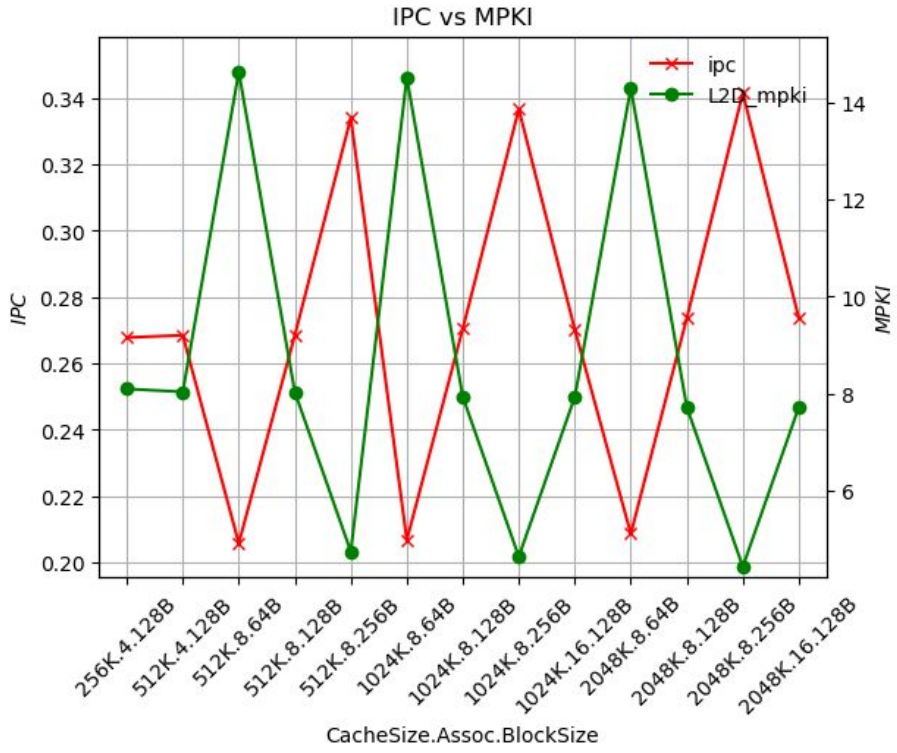
Όσον αφορά το mpki, όσο αυξάνεται το μέγεθος της μνήμης καθώς και το block size μειώνεται. τα misses στην L2-cache.
Το ipc συμπεριφέρεται απρόβλεπτα στο συγκεκριμένο benchmark και συχνά παρουσιάζει συμπεριφορά που δεν συμβαδίζει με τα misses. Σε γενικές γραμμές αυξάνεται με την αύξηση του μεγέθους μνήμης.
Το βέλτιστο αποτέλεσμα έχουμε για μνήμη 2048KB, 8 assoc και block size 256B

8. Raytrace



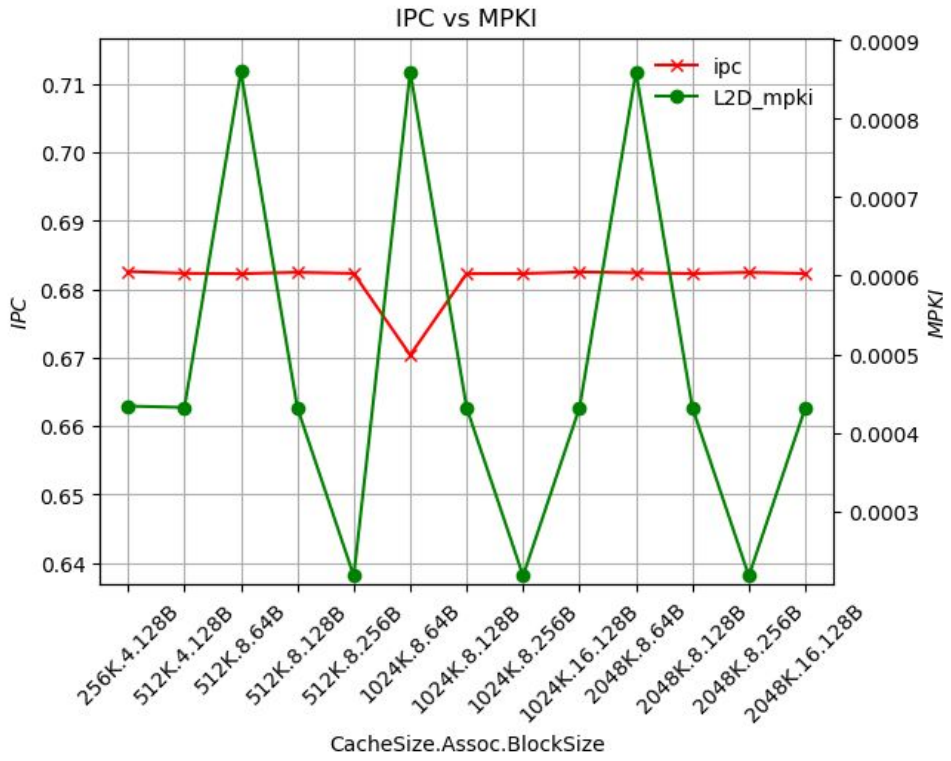
Εδώ εχουμε αποτελέσμα όμοια με το Ferret. Βεβαίως, τα misses είναι σε πολύ μικρό βαθμό οπότε υπάρχει μικρότερη διακύμανση.

9. Streamcluster



Έχουμε όμοια αποτελέσματα με το benchmark facism.

10. Swaptions



Εδώ το mpci μεταβάλετται ακριβώς όπως το steamcluster. Το ipc, δεδομένοθυ πως έχουμε πολύ μικρό αριθμο misses, μένει πρακτικά σταθερό.

Συμπεράσματα

Συμπερασματικά έχουμε πως αρκετά benchmarks (facism, ferret, fluidanimate, raytrace και streamcluster) επηρεάζονται κατά βάση από το block size.

Το associativity παίζει ξανά βασικό ρόλο σε μετροπρογράμματα όπως το blacksholes ενώ το μέγεθος της μνήμης σε μετροπρογράμματα όπως το bodytrack, το canneal και σε μικρότερο βαθμό το freqmine.

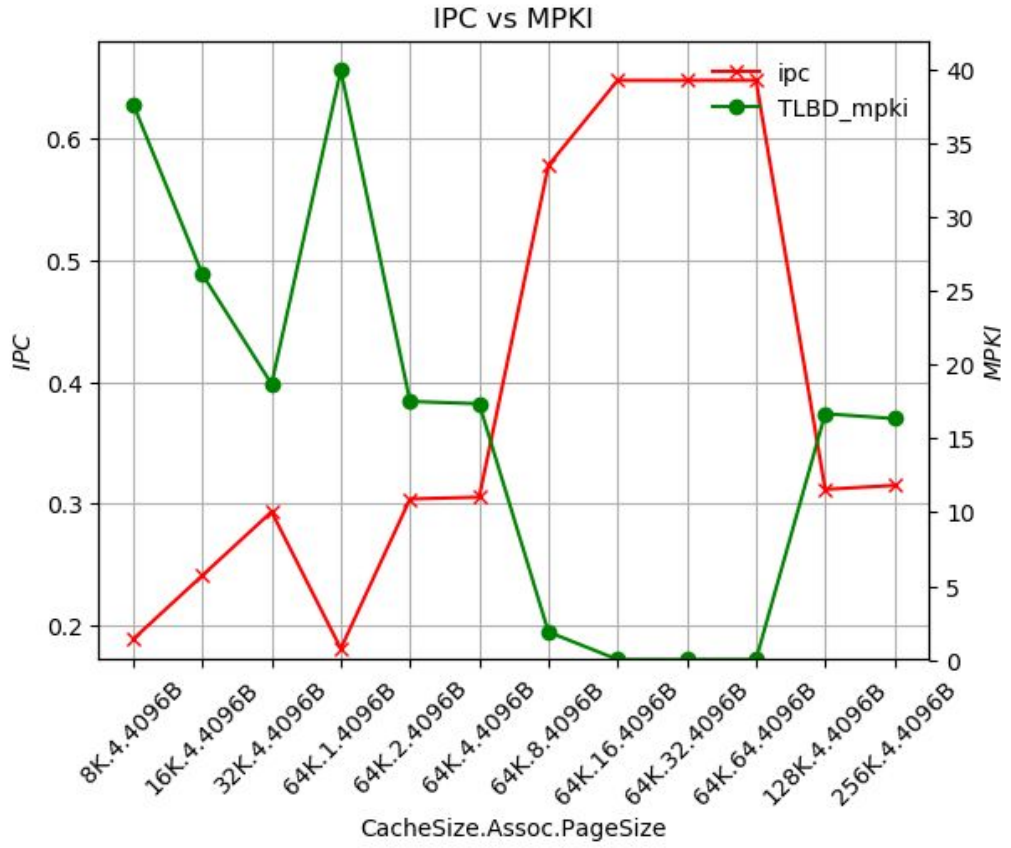
Τέλος, το raytrace δεν επηρεάζονταν έντονα από μεταβολές στην L2 μνήμη.

Συνολικά, μιας και τα περισσότερα προγράμματα εξαρτώνται από το block size, καλές επιλογές μνήμης L2 από τα αποτελέσματα μας φαίνονται εκείνες που έχουν block size 256B.

TLB

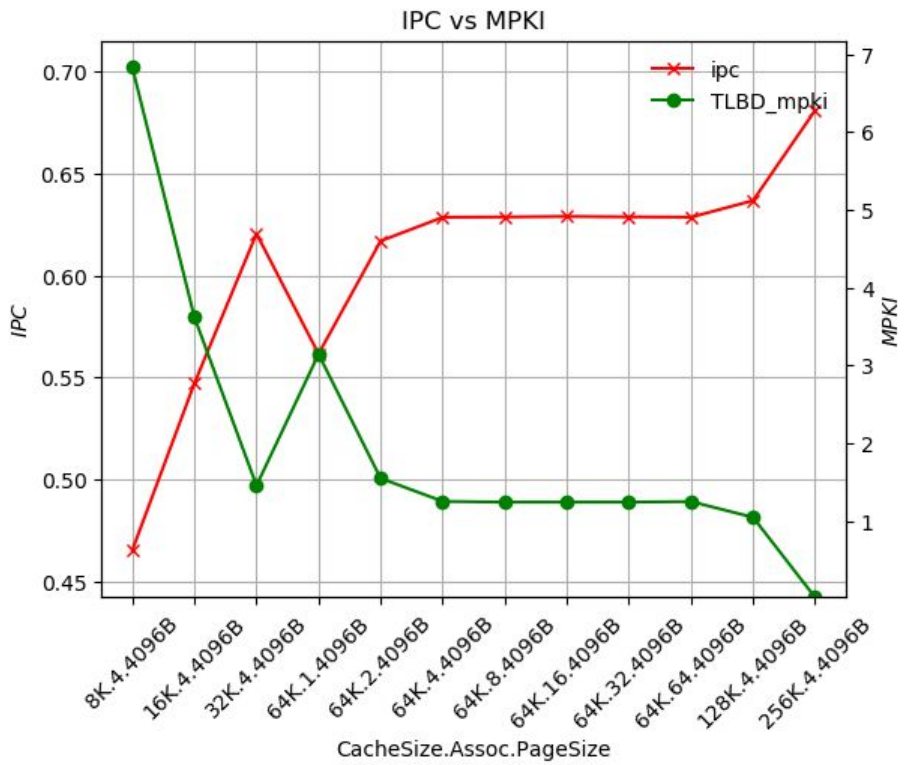
Εδώ διατηρήσαμε τις τιμές της L1 Cache και L2 Cache σταθερές και ίσες με τις τιμές που δίνονται στην εκφώνηση και μεταβάλλαμε τις παραμέτρους της TLB στις τιμές που φαίνονται και στον x άξονα των παρακάτω διαγραμμάτων. Συγκεκριμένα κρατάμε σταθερό το page size και μεταβάλλουμε το associativity και το μέγεθος του TLB.
Παρακάτω βλέπουμε όλα τα αποτελέσματα για τα προαναφερθέντα benchmarks.

1. Blackschole



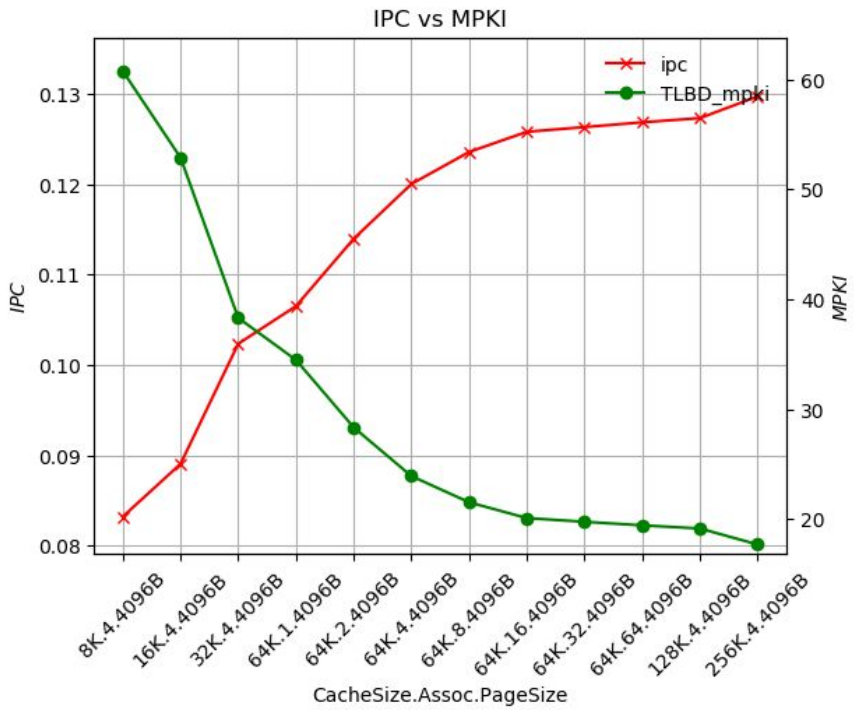
Εδώ βλέπουμε πως όσο αυξάνεται το associativity έχουμε μείωση των misses και αύξηση της απόδοσης. Για associativity από 16 και πάνω μειώνονται τόσο που μεγιστοποιείται το ipc.
Ταυτόχρονα, όσο αυξάνεται το μέγεθος του TLB βελτιώνεται η απόδοση. Όσο μεγαλώνει βέβαια το μέγεθος τόσο μικραίνει η αύξηση ενώ απο 128 σε 256K είναι μηδαμινή.

2. Bodytrack



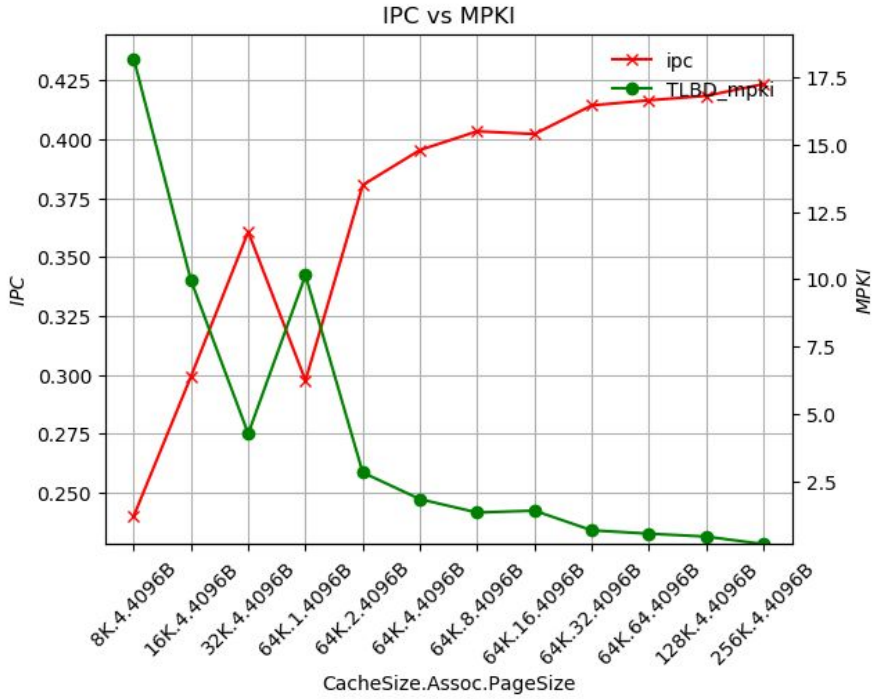
Εδώ βλέπουμε πως η κυριότερη παράμετρος είναι το μέγεθος του TLB, καθώς όσο αυξάνεται υπάρχει σημαντική βελτίωση της απόδοσης.
Το associativity για μικρές τιμές (1 και 2) δίνει λιγότερη απόδοση ενώ για μεγαλύτερο από δύο παραμένει σταθερή.

3. Canneal



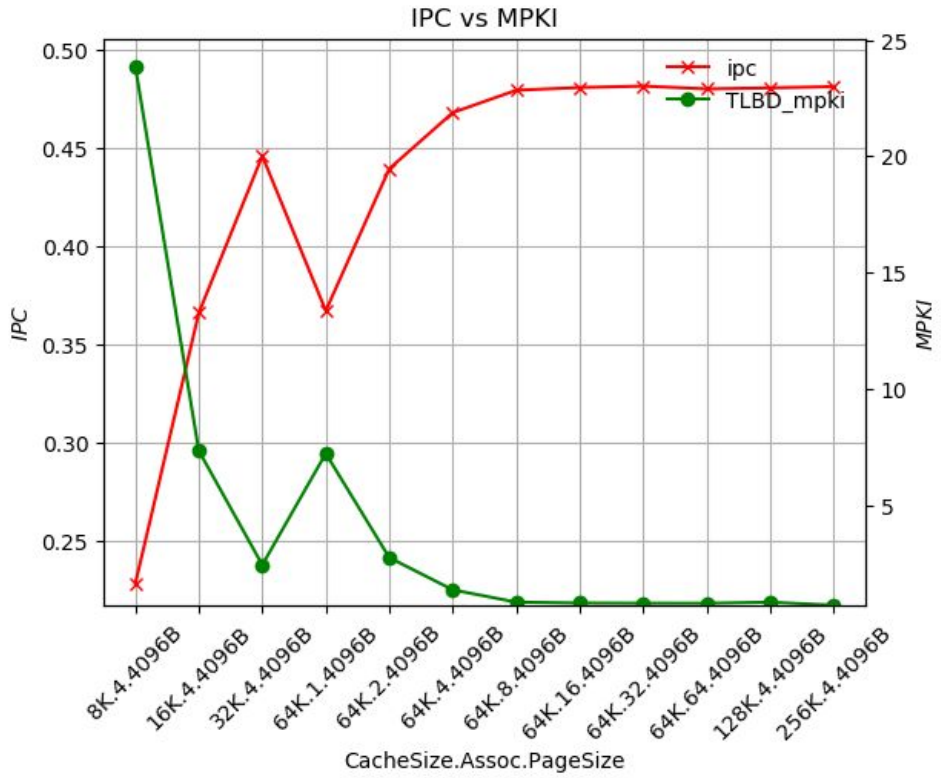
Εδώ τόσο το μέγεθος του TLB όσο και το associativity επηρεάζουν σημαντικά την απόδοση. Μεγαλύτερο ρόλο παίζει σίγουρα το μέγεθος μιας και, σε αντίθεση με το Blackschole, για μεγαλύτερο μέγεθος και μικρότερο associativity έχουμε καλύτερη απόδοση.

4. Facesim



Όμοια με το Bodytrack, όσο το μέγεθος του TLB όσο και το associativity επηρεάζουν την απόδοση. Το associativity για μικρές τιμές (1 και 2) έχει αισθητά μικρότερη απόδοση και για μεγαλύτερο associativity επηρεάζει ελάχιστα.

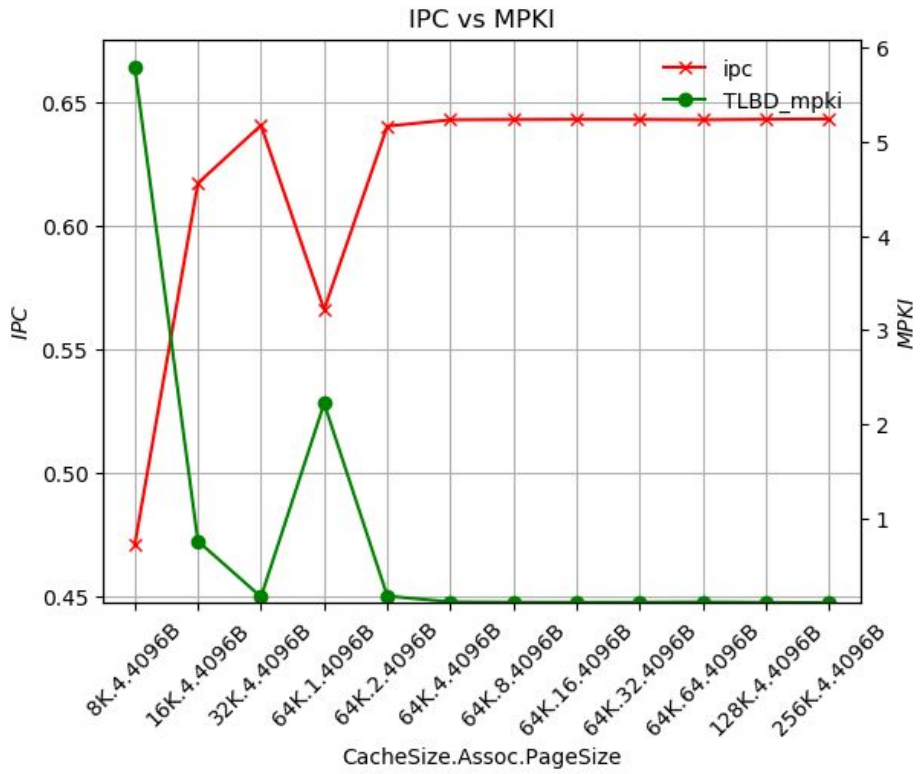
5. Ferret



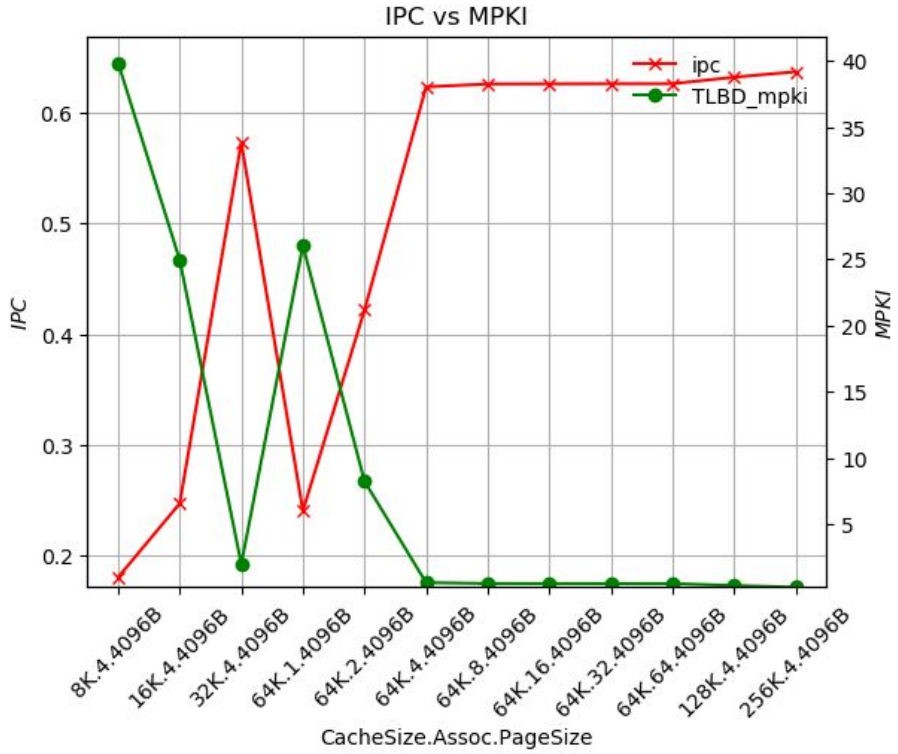
Μέχρι και associativity 8 και μέγεθος 64K η απόδοση βελτιώνεται όσο αυξάνεται κυρίως το μέγεθος του TLB αλλά και το associativity. Για associativity 8 και μέγεθος 64K γτάνει στο peak της απόδοσης και παραμένει σταθερή.

Στα παρακάτω benchmarks βλέπουμε όμοια αποτελέσματα με αυτό με διαφορετικές διακυμάνσεις κυρίως λόγω του διαφορετικού αριθμού λαθών που γίνεται.

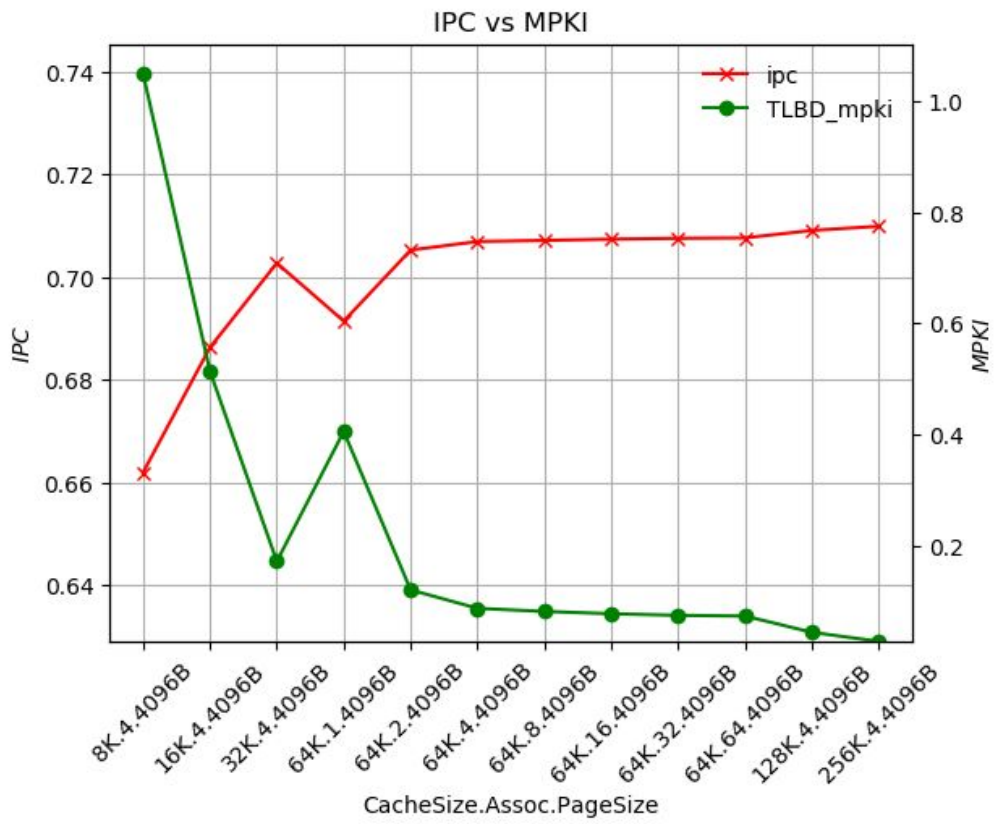
6. Fluidanimate



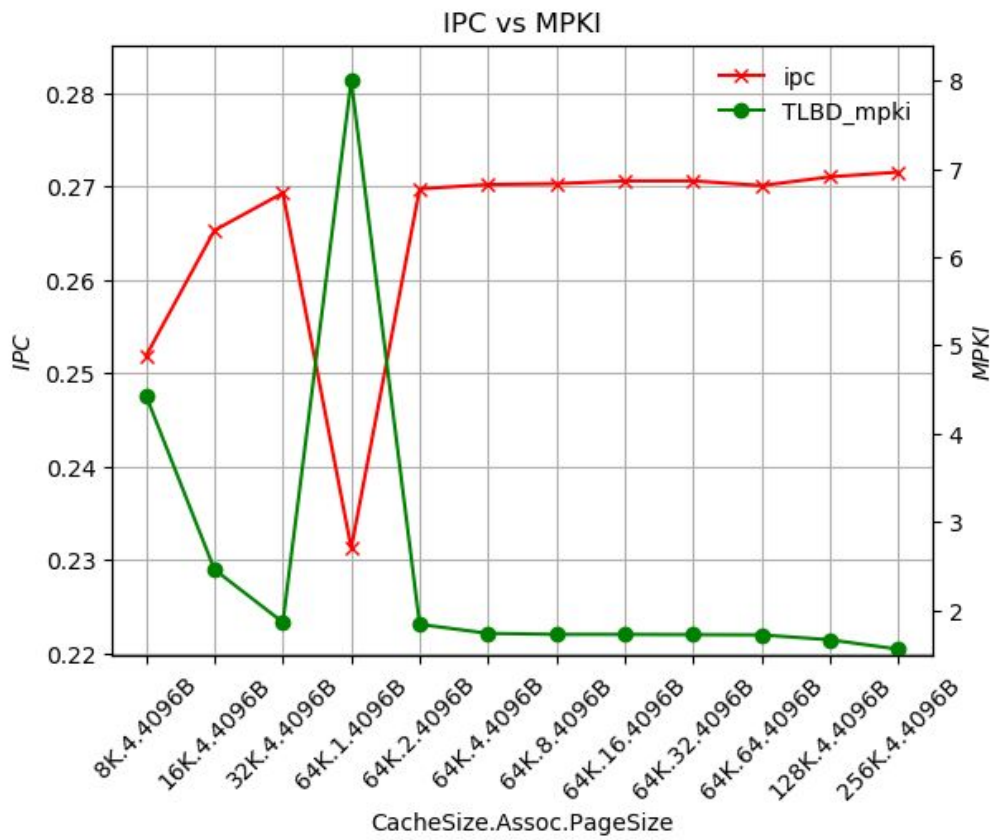
7. Freqmine



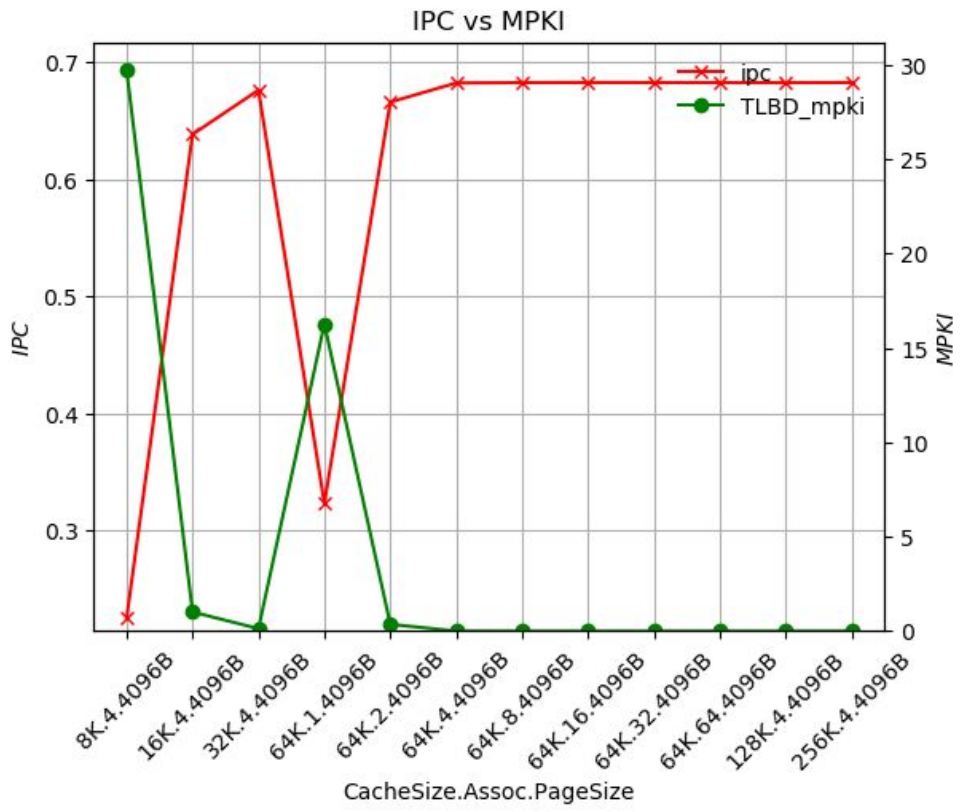
8. Raytrace



9. Streamcluster



10. Swaptions



Συμπεράσματα

Συμπερασματικά όλα τα παραπάνω benchmarks επηρεάζονται από το μέγεθος του TLB καθώς και το associativity. Περισσότερο ρόλο παίζει ίσως το μέγεθος του TLB (πέρα από το blackscholes). Ακόμη, τα ferret, fluidanimate, freqmine, raytrace, streamcluster και swaptions φτάνουν γρήγορα σε μια βέλτιστη τιμή που δεν βελλτιώνεται με αλλαγές στην TLB.

Συνολικά, μια καλή επιλογή μνήμης που δίνει καλά αποτελέσματα τις περισσότερες φορές θα ήταν με μέγεθος 32 entries, 4 way associative, μιας και ειναι φθηνή και δίνει ικανοποιητικά αποτελέσματα σε όλες τις περιπτώσεις.
Καλύτερα δίνει και εκείνη με 64 entries. Τέλος, εκείνη με τα 256 δίνει σίγουρα τα καλύτερα αλλά με ελάχιστες διαφορές από τις προηγούμενες.

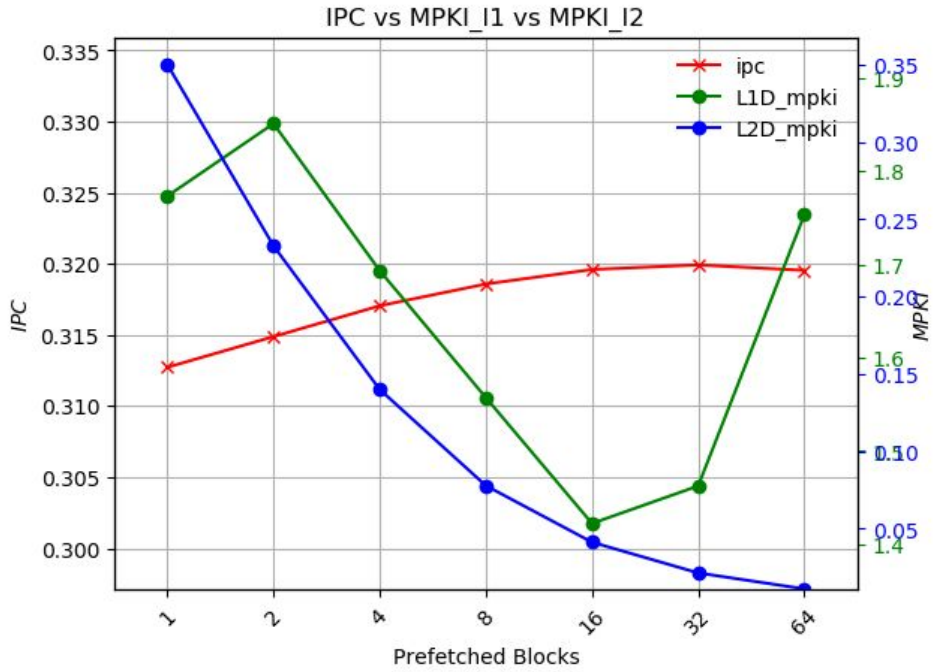
Προανάκληση (prefetching)

Εδώ διατηρήσαμε τις τιμές της L1 Cache και L2 Cache και TLB σταθερές και ίσες με τις τιμές που δίνονται στην εκφώνηση ενώ επεκτάθηκε το πρόγραμμα cache.h που δινόταν όπως φαίνεται παρακάτω, ώστε να ενεργοποιηθεί το prefetching.

```
// 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];
    bool l2find = l2Set.Find(l2Tag);

    if (!l2find) {
        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);
        }
    }
}
    /* ........................... */
}
```

Συγκεκριμένα, μετααβάλλουμε κάθσε φορά τον αριθμό των blocks που κάνει prefetching η μνήμη cahe L2, αλλάζοντας δηλαδή την τομή του L2prf κάθε φορά stiw ενδεικνυόμενες τιμές. Παρακάτω βλέπουμε όλα τα αποτελέσματα για τα προαναφερθέντα benchmarks.

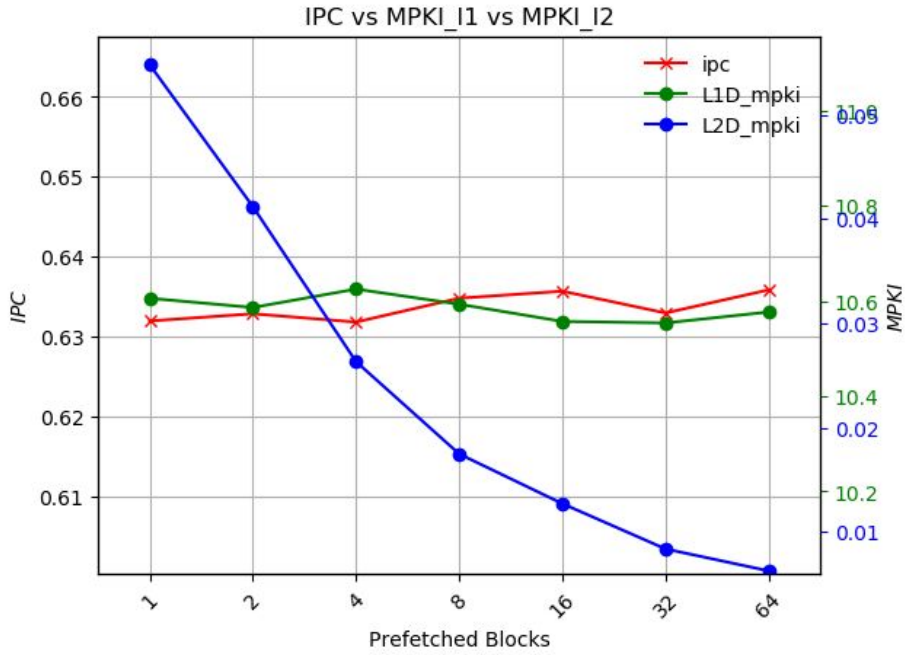
1. Blackschole



Βλέπουμε πως γενικα όσο αυξάνονται τα blocks τόσο καλύτερη απόδοση έχουμε. Το ipc έχει σταθερη πορεία, χωρίς να παρουσιάζει μεγάλες αλλαγες και όταν πάμε στα 64 υπάρχει μια μικρή πτώση. Τα misses της L2 μειώνονται εκθετικά και φτάνουν σε ελάχιστη τιμή για n=64. Τα misses της L1 από την άλλη ενώ μειώνονται γραμμικά για n

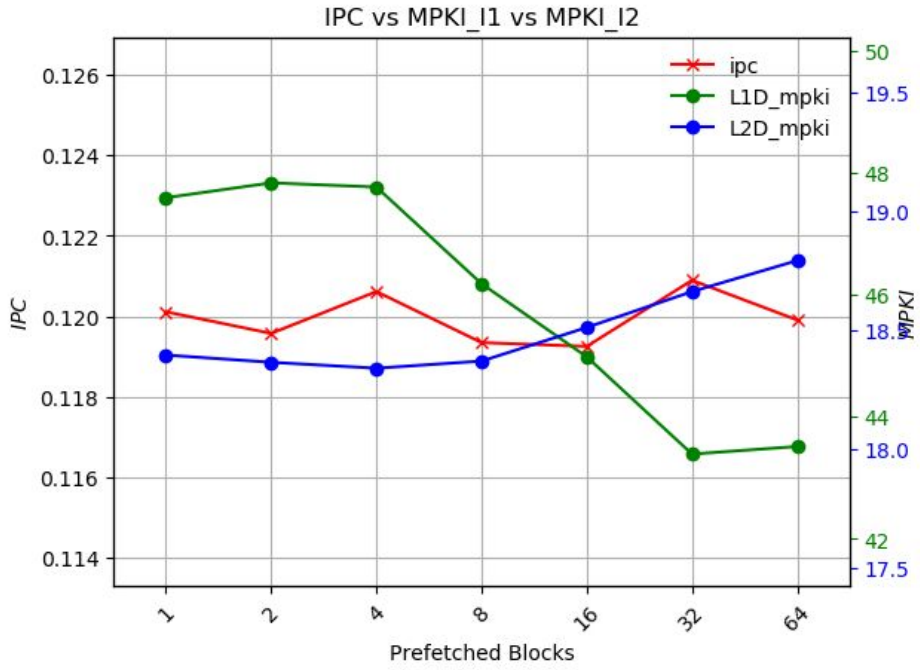
από 2 εώς και 16, παρουσιάζουν αντίθετη συμπεριφορά στις υπόλοιπες τιμές ενώ για n = 64 έχουμε πολύ μεγάλη αύξηση των misses.

2. Bodytrack



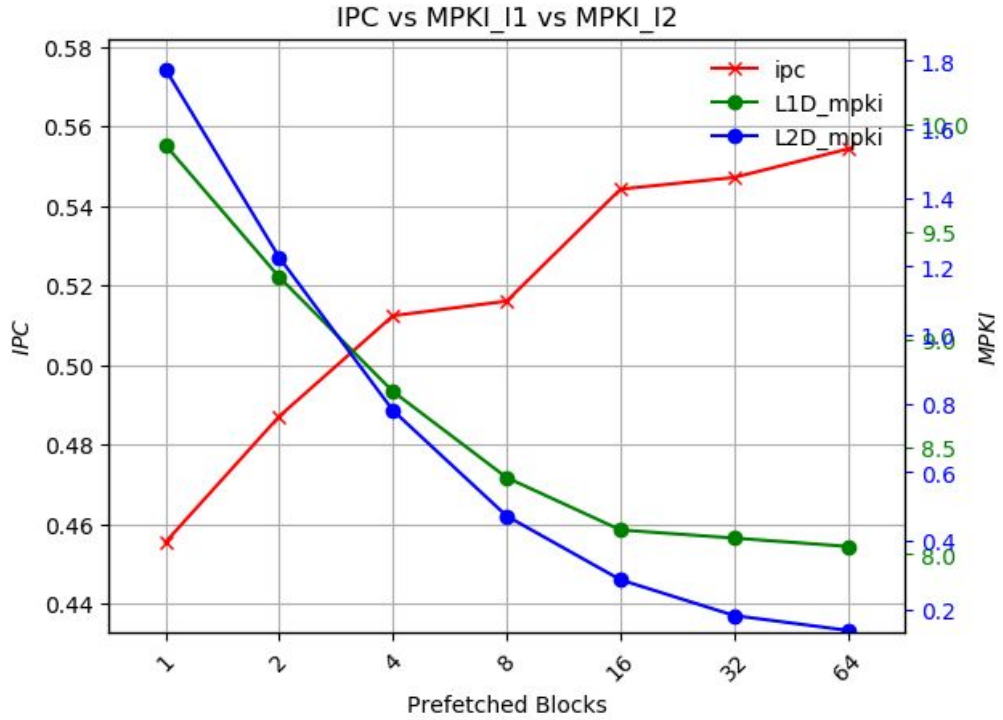
Εδώ τα misses της L2 μειώνονται έντονα καθώς αυξάνονται τα prefetched blocks. Βέβαια βλέπουμε πως είναι πολύ μικρές οι τιμές των misses. Οι υπόλοιπες τιμές δεν επηρεάζονται ιδιαίτερα.

3. Canneal



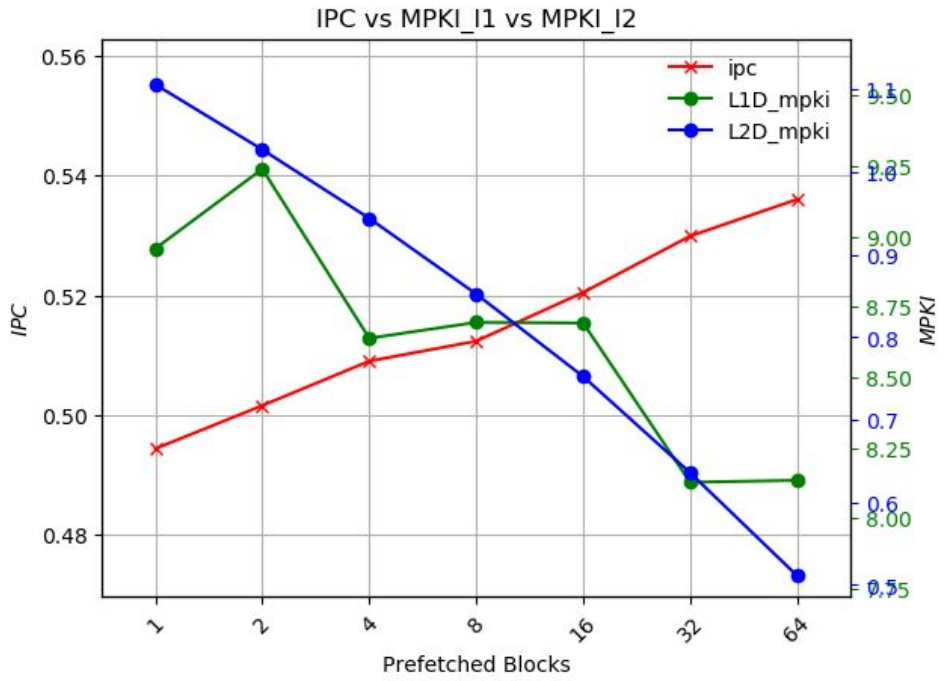
Εδώ βλέπουμε πως το ipc μένει πάνω κάτω σταθερό. Αντίθετα, η απόδοση της L1 μειώνεται ενώ της L2 αυξάνεται καθώς αυξάνονται τα prefetched blocks.

4. Facesim



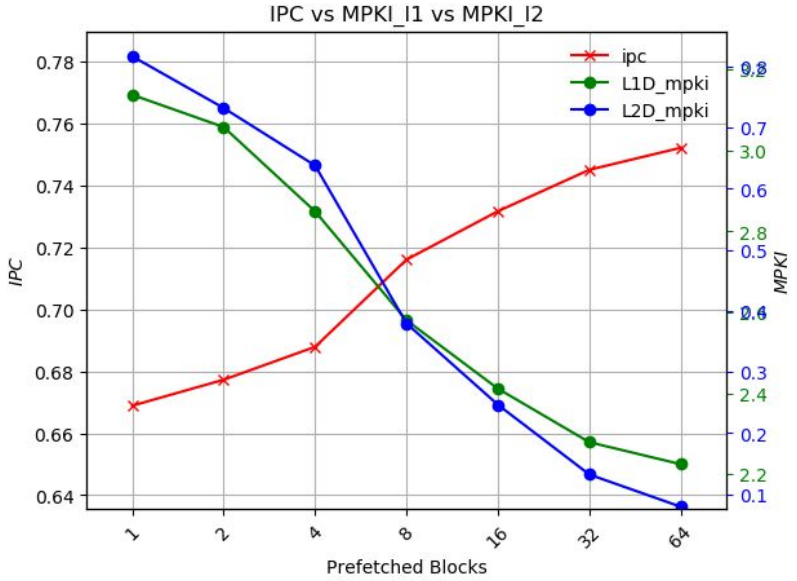
Εδώ φαίνεται ξεκάθαρα πως σε κάθε περίπτωση καθώς αυξάνονται τα prefetched blocks αυξάνεται και η απόδοση.

5. Ferret



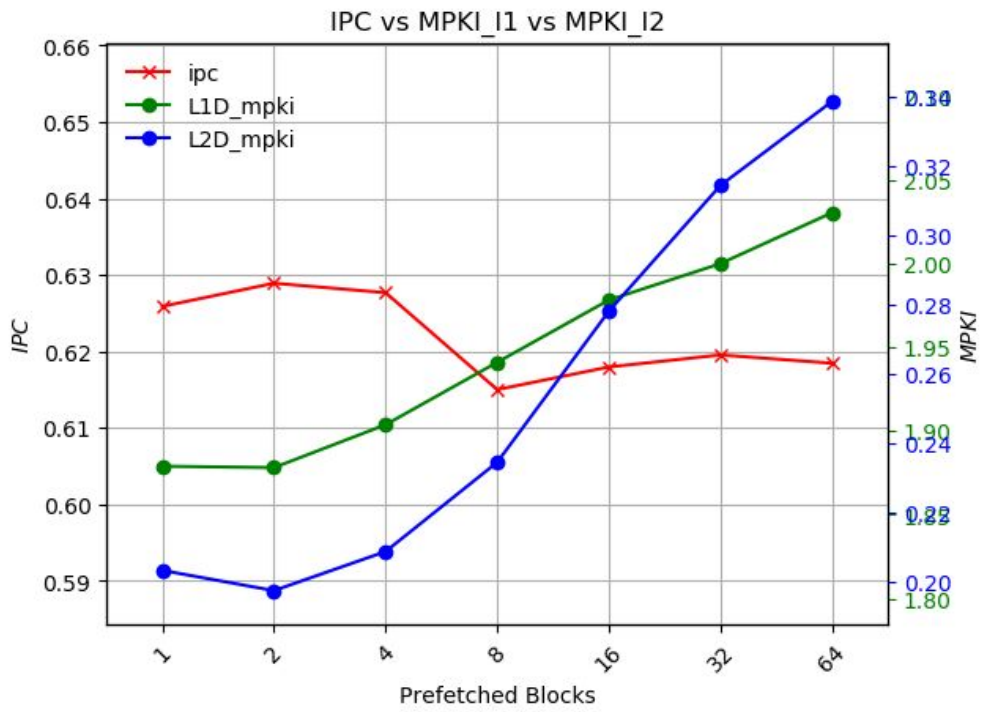
Όμοιως με facesim, με εξαίρεση την L1 για n=1.

6. Fluidanimate



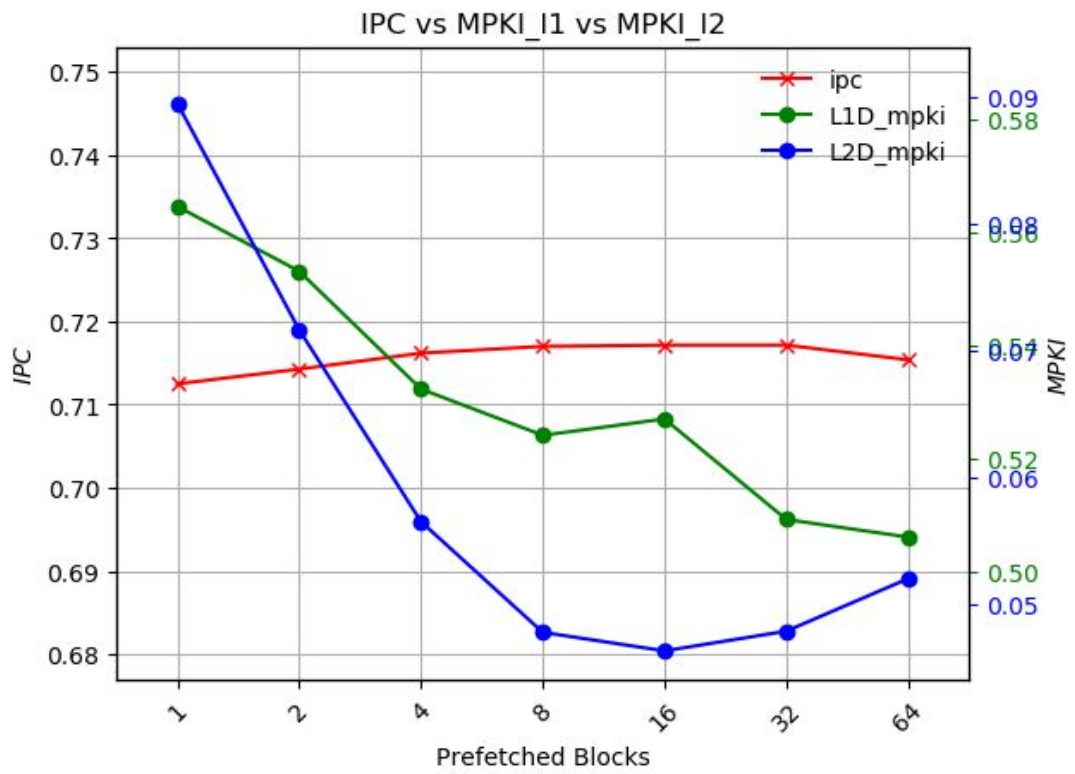
Ομοίως με facesim.

7. Freqmine



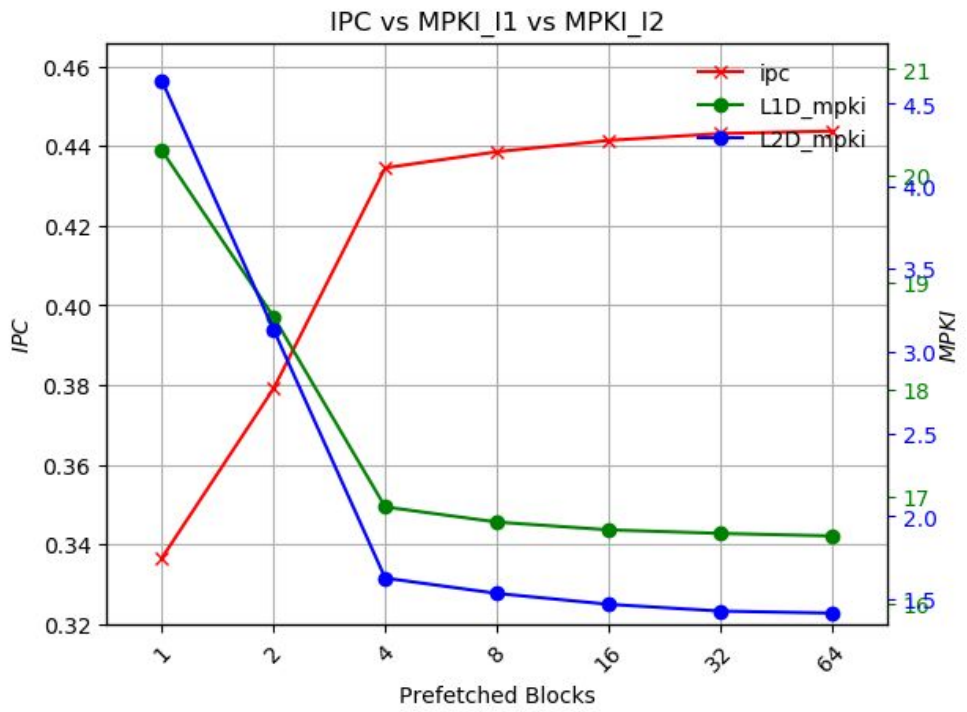
Εδώ βλέπουμε πως οι τιμές mpki και για τις 2 cache μνήμες αυξάνονται καθώς μεγαλώνουν τα prefetched block. Το ipc δεν παρουσιάζει μεγάλες διακυμάνσεις. Παρόλα αυτά είναι ελαφρώς μεγαλύτερο για μικρές τιμές prefetched blocks. Η βέλτιστη τιμή εμφανίζεται για n=2.

8. Raytrace



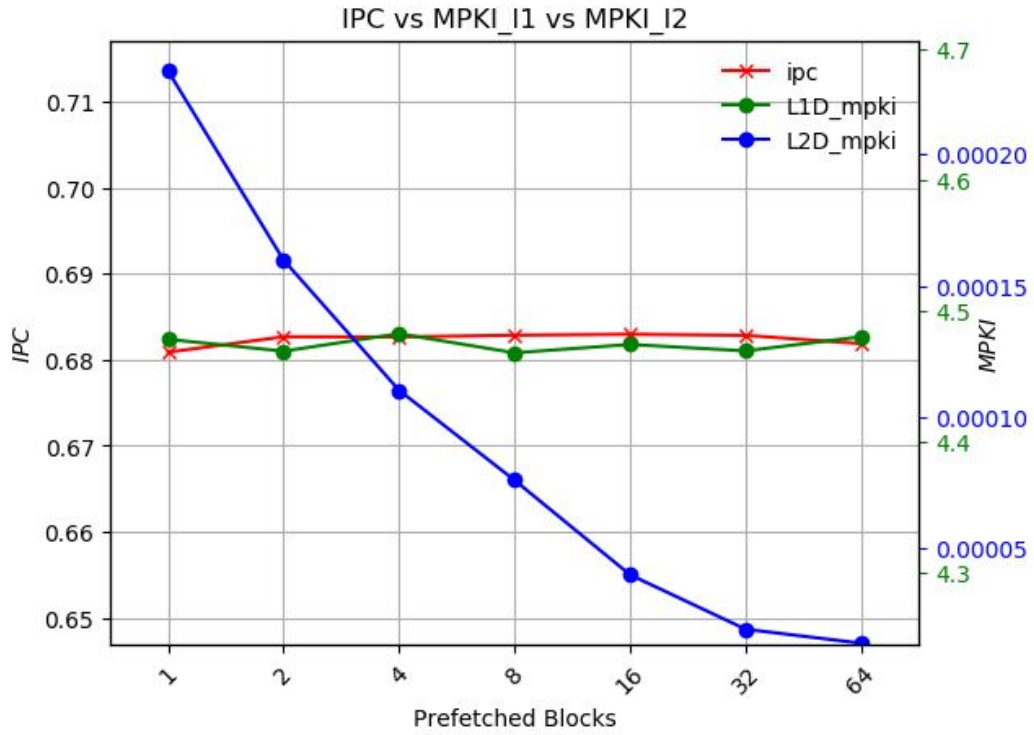
Εδώ τα misses της L2 μειώνονται δραματικά με την αύξηση των prefetched blocks αλλά βρίσκονται σε πολύ μικρές τιμές οπότε δεν έχει μεγάλη επιρροή. Παρατηρείται μια γενικότερη μείωση του mpki και στην L1 ενώ το ipc αυξάνεται είναι ελάχιστα μικρότερη για μικρό αριθμό blocks.

9. Streamcluster



Όμοια με facism, καθώς αυξάνονται τα prefetched blocks αυξάνεται και η απόδοση. Βέβαια, έχουμε πολύ μεγάλη βελτίωση μέχρι και 4 blocks ενώ από εκεί και έπειτα υπάρχει ελάχιστη βελτίωση.

10. Swaptions



Όμοια με bodytrack.

Συμπεράσματα

Συμπερασματικά έχουμε πως τα περισσότερα benchmarks (blacksholes, facism, ferrwet, fluidanimate, streamcluster και λιγότερο το raytrace), αυκάνονται καθώς αυξάνονται τα prefetched blocks.

Τα υπόλοιπα μένουν πάνω κάτω σταθερά με το freqmine να έχει μια ελαφρια πρώση.

Συνολικά μια καλή επιλογή είναι να γίνουν prefetched 32 blocks.

7.2 Μελέτη μεταβολής μετρικών απόδοσης στον χρόνο

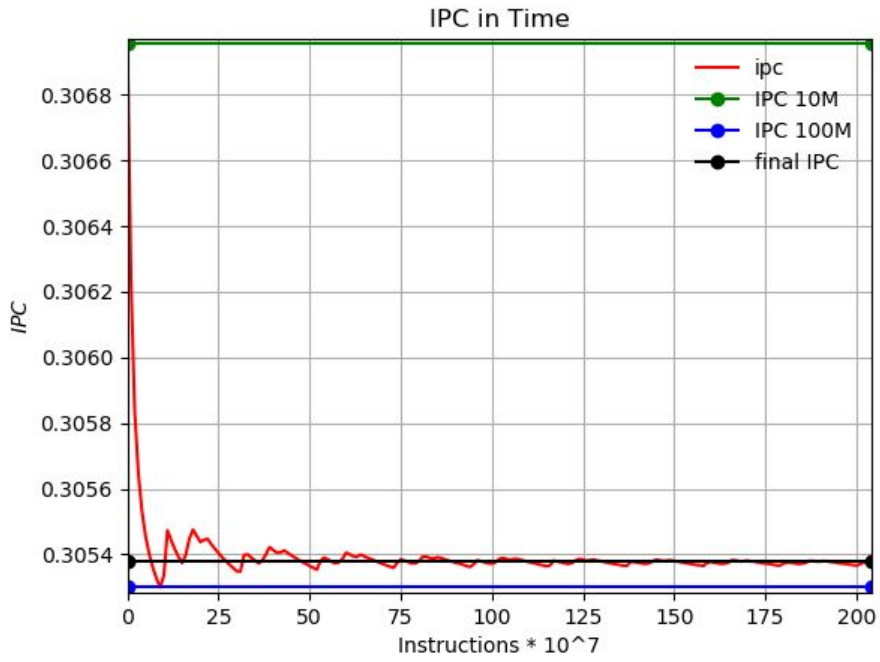
Εδώ επιλέξαμε το εξής configuration:
L1: 32KB, 8-way, blocksize 64B
L2: 1024KB, 8-way, blocksize 128B,
TLB: 64 entries(size) , 4-way, 4096 page size
Write-allocate ιεραρχία, LRU πολιτική αντικατάστασης
L2prf = 0 (0 prefetched blocks στην L2).

Τροποιούμε το αρχείο simulator.cpp όπως φαίνεται παρακάτω ώστε να κρατάμε στατιστικά το ipc κάθε 10 εκατομμύρια εντολές αντί για ολόκληρη την περιοχή ενδιαφέροντος.

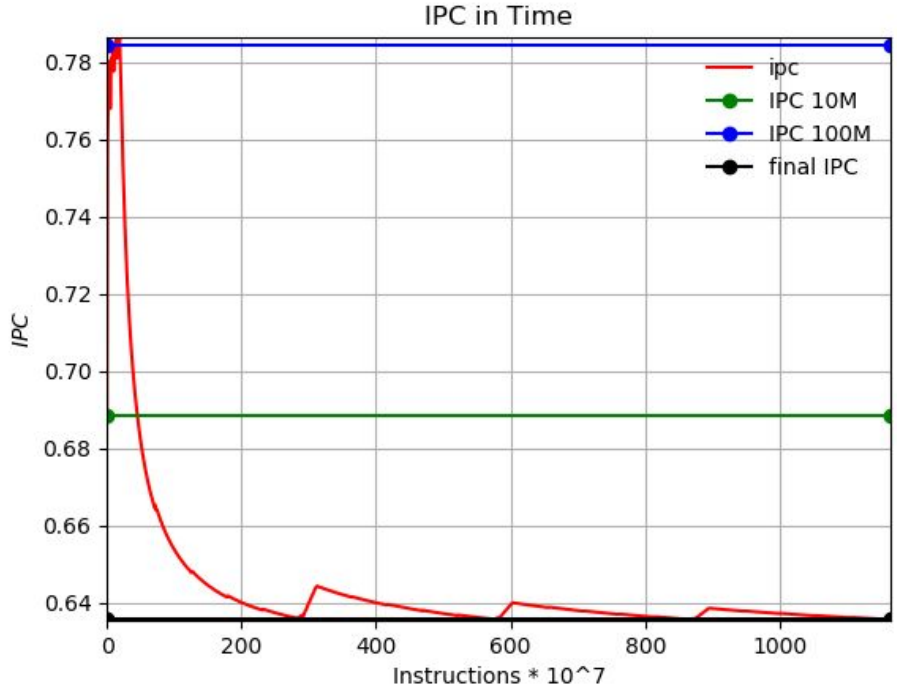
```
VOID count_instruction()
{
    total_instructions++;
    total_cycles++;

    //keep stats per 10M instructions
    if (total_instructions % 10000000==0){
        outFile << "INSTRUCTIONS: " << total_instructions / 10000000 << " M IPC: " << (double)total_instructions / (double)total_cycles << "\n";
    }
}
```

Έτσι έχουμε την δειγματοληψία στο output αρχείο.
Απεικονίζοντας αυτές τις τιμές σε διαγράμματα βλέπουμε πως μεταβάλλεται σε σχέση με τον χρόνο (αριθμό εντολών) εκτέλεσης των benchmarks.
Παράλληλα βλέπουμε το ipc όταν είμαστε στις 10M και στις 100M εντολές και το συγκρίνουμε με το πραγματικό ipc.

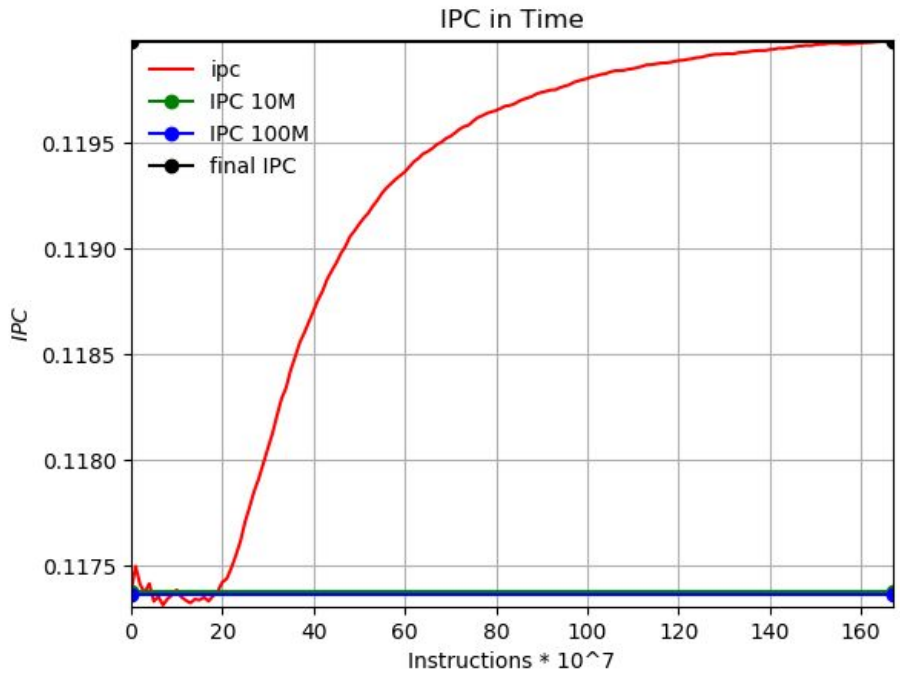
1. Blackschole



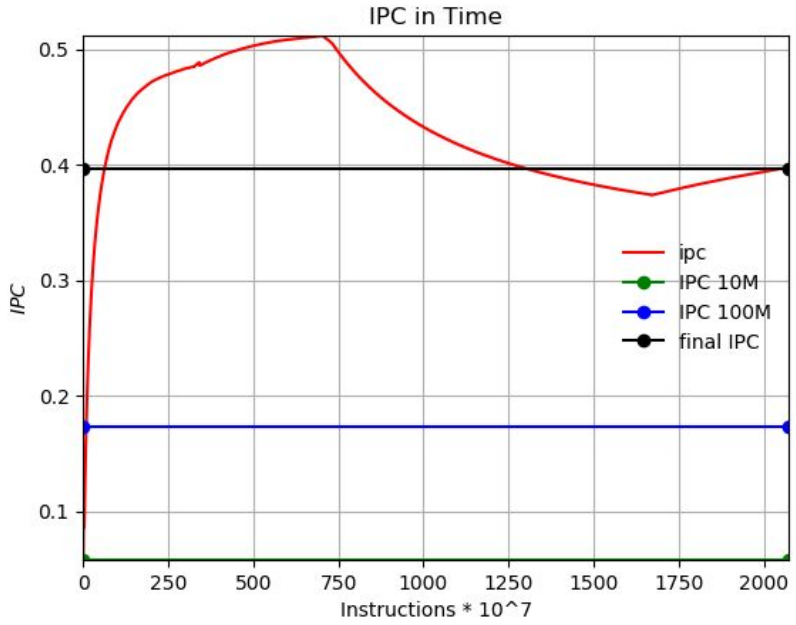
2. Bodytrack



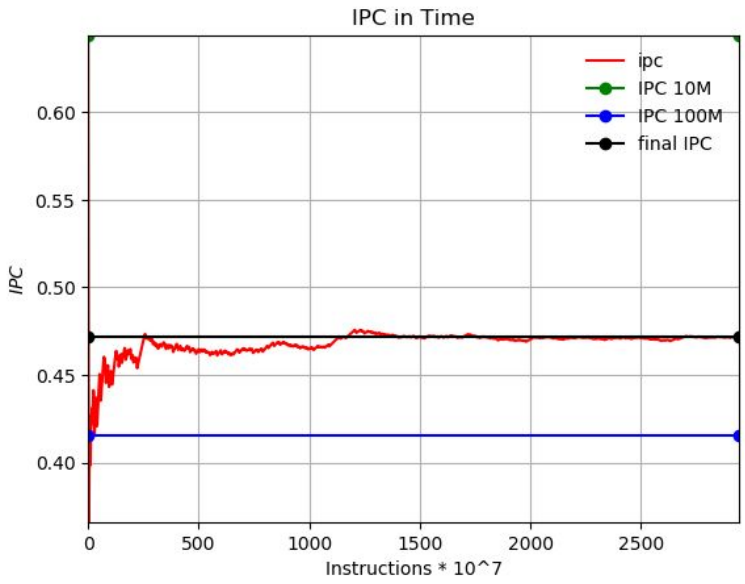
3. Canneal



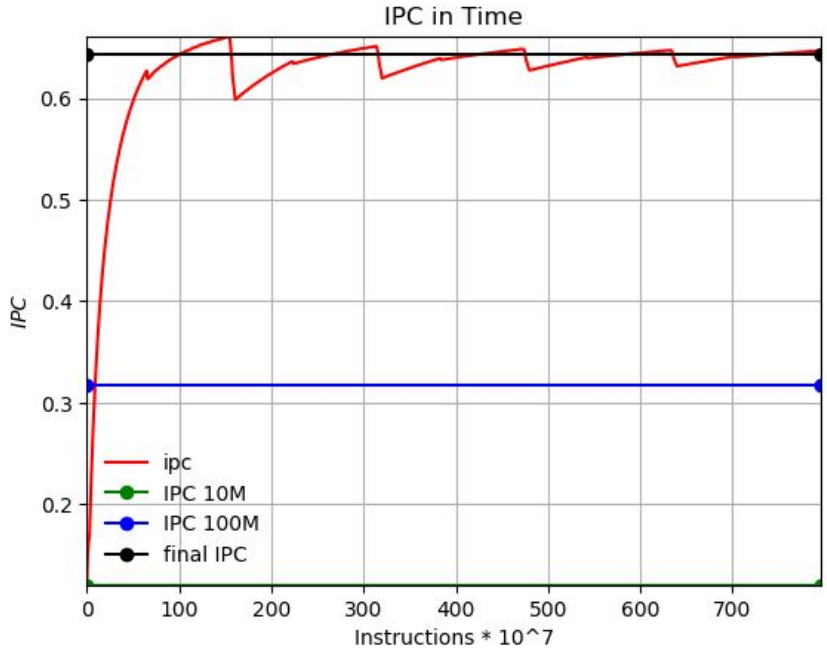
4. Facesim



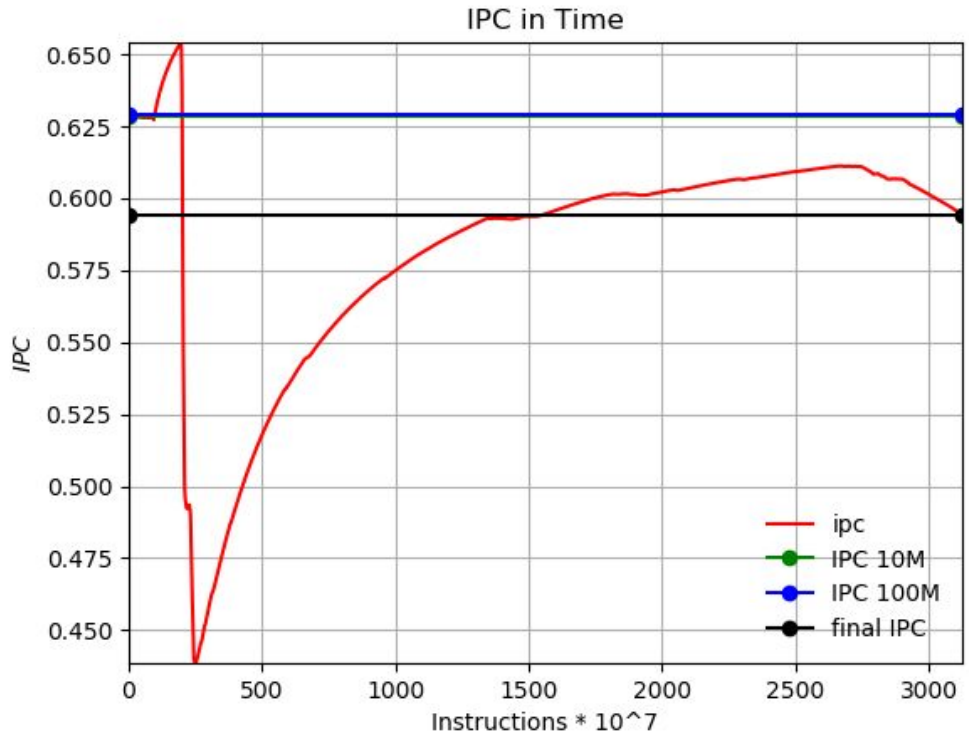
5. Ferret



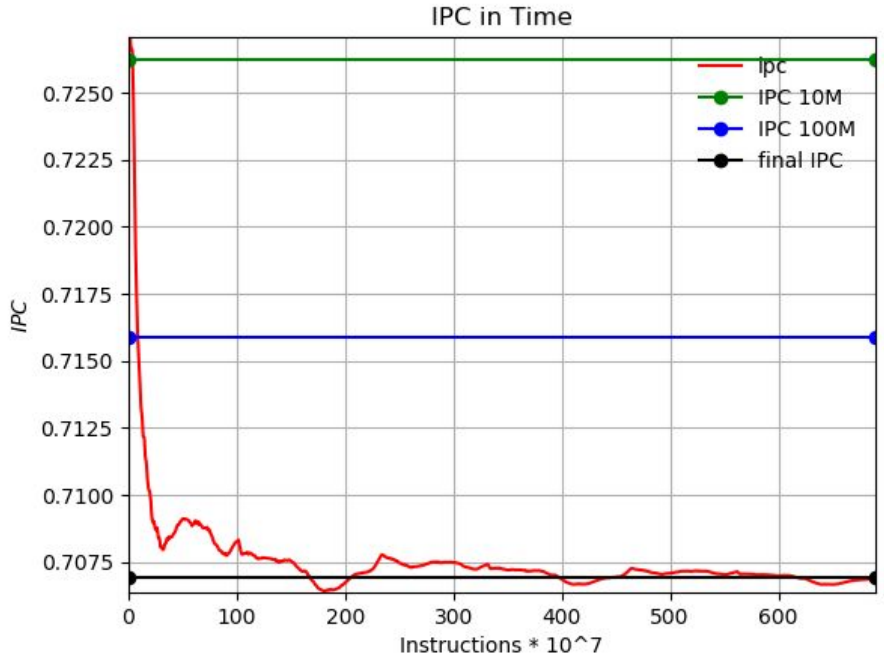
6. Fluidanimate



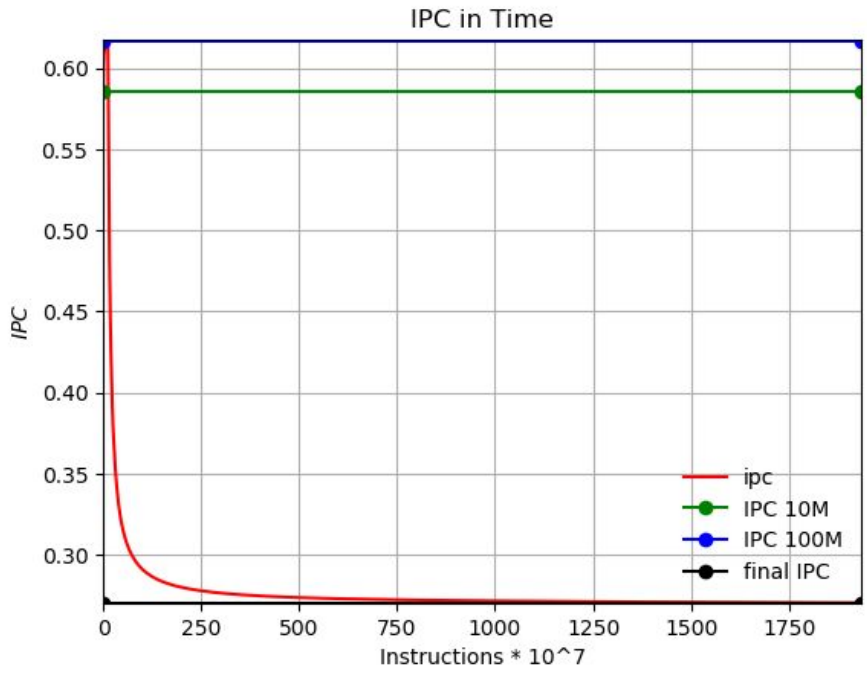
7. Freqmine



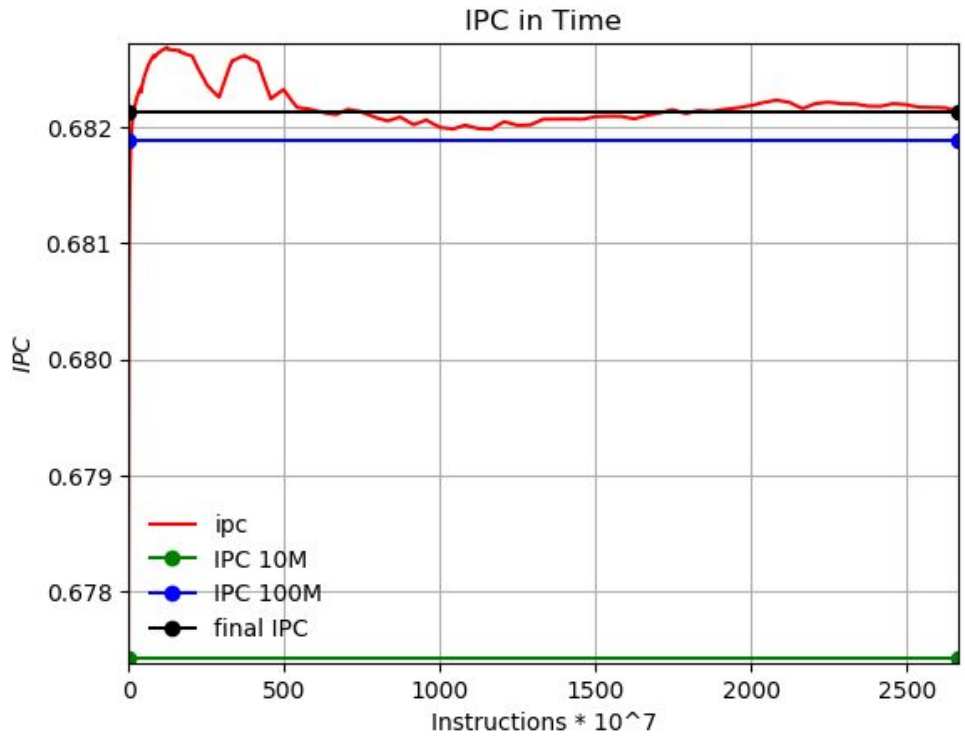
8. Raytrace



9. Streamcluster



10. Swaptions



Απόρροια των παραπάνω είναι πως το IPC στις 10M ή στις 100M δεν είναι συνήθως αντιπροσωπευτικό του πραγματικού IPC. Αυτό είναι εύλογο δεδομένου ότι τα προγράμματα δεν παρουσιάζουν ομοιογένεια και οι πρώτες εντολές δεν είναι ενδεικτικές του συνολικού προγράμματος.

Στα blackscholes, streamcluster και swaptions το IPC στα 100M είναι σχετικά κοντα με την πραγματική τιμή. Αντίθετα η τιμή στα 10M δεν πλησιάζει σχεδόν ποτέ σε κανένα benchmark την πραγματική τιμή.

Συνεπώς είναι προτιμότερο να πάρουμε το πραγματικό IPC και όχι με δειγματοληψία ή να πάρουμε κάποιο δείγμα αφού έχει εκτελεστεί μεγαλύτερο μέρος του προγράμματος που όπως φαίνεται από τις γραφικές παραστάσεις θα ήταν σαφέστατα περισσότερο ενδεικτικό.