2η 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 & PINTOOLS

Στα πλαίσια της παρούσας άσκησης χρησιμοποιήσαμε τα εργαλείο “PIN” (όπως και στην πρώτη άσκηση) για να μελετηθεί η επίδραση διαφορετικών συστημάτων πρόβλεψης εντολών άλματος καθώς και η αξιολόγηση τους με δεδομένο το διαθέσιμο χώρο πάνω στο τσιπ.

Για αυτά χρησιμοποιήθηκαν τα δοσμένα pintools cslab_branch_stats.cpp και cslab_branch.cpp ύστερα από κατάλληλη τροποποίηση.
Το πρώτο χρησιμοποιείται για την εξαγωγή στατιστικών σχετικά με τις εντολές αλμάτων που εκτελούνται από την εφαρμογή.
Το δεύτερο χρησιμοποιείται για την αξιολόγηση τεχνικών πρόβλεψης άλματος ενώ για τις εντολές επιστροφής από διαδικασίες προσομοιώνει διαφορετικά μεγέθη στοίβας διεύθυνσης επιστροφής

Οι διάφοροι branch predictors ορίζονται στο αρχείο branch_predictors.η.

BENCHMARKS

Στα πλαίσια της παρούσας άσκησης χρησιμοποιήσαμε ορισμένα από τα SPEC_CPU2006. Συγκεκριμένα χρησιμοποιήσαμε τα παρακάτω 12:

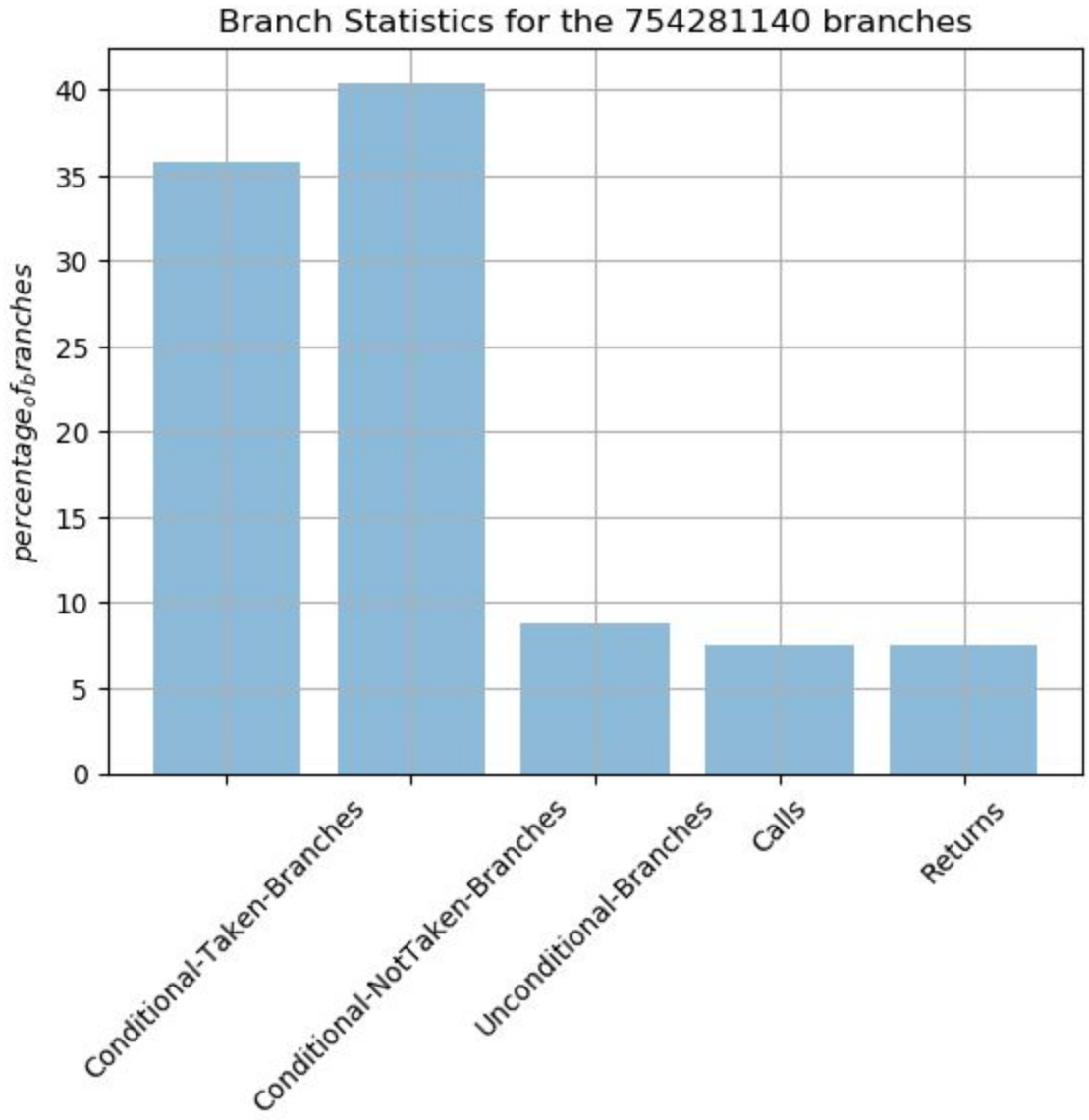
- 403.gcc
- 429.mcf
- 434.zeusmp
- 436.cactusADM
- 445.gobmk
- 450.soplex
- 456.hmmer
- 458.sjeng
- 459.GemsFDTD
- 471.omnetpp
- 473.astar
- 483.xalancbmk

4. Πειραματική αξιολόγηση

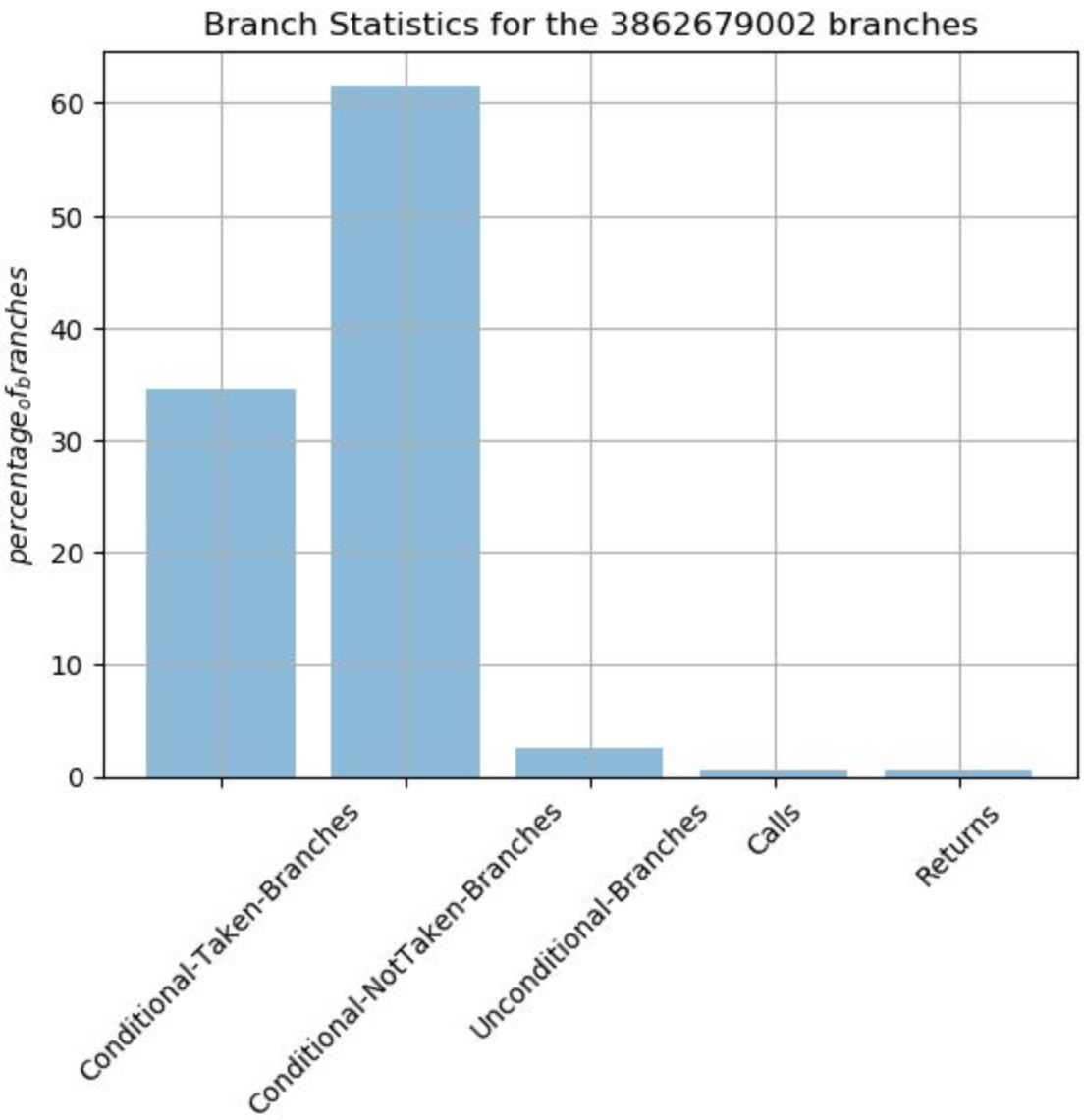
4.1 Μελέτη εντολών άλματος

Εδώ συλλέξαμε στατιστικά για τις εντολές άλματος που εκτελούνται από τα benchmarks. Συγκεκριμένα αναφέρεται στον τίτλο ο συνολικός αριθμός των εντολών άλματος που συναντήθηκαν στο μετροπρόγραμμα αι ύστερα το ποσοστό αυτών που ανήκουν σε κάθε κατηγορία.
Παρακάτω βλέπουμε όλα τα αποτελέσματα για τα προαναφερθέντα benchmarks.

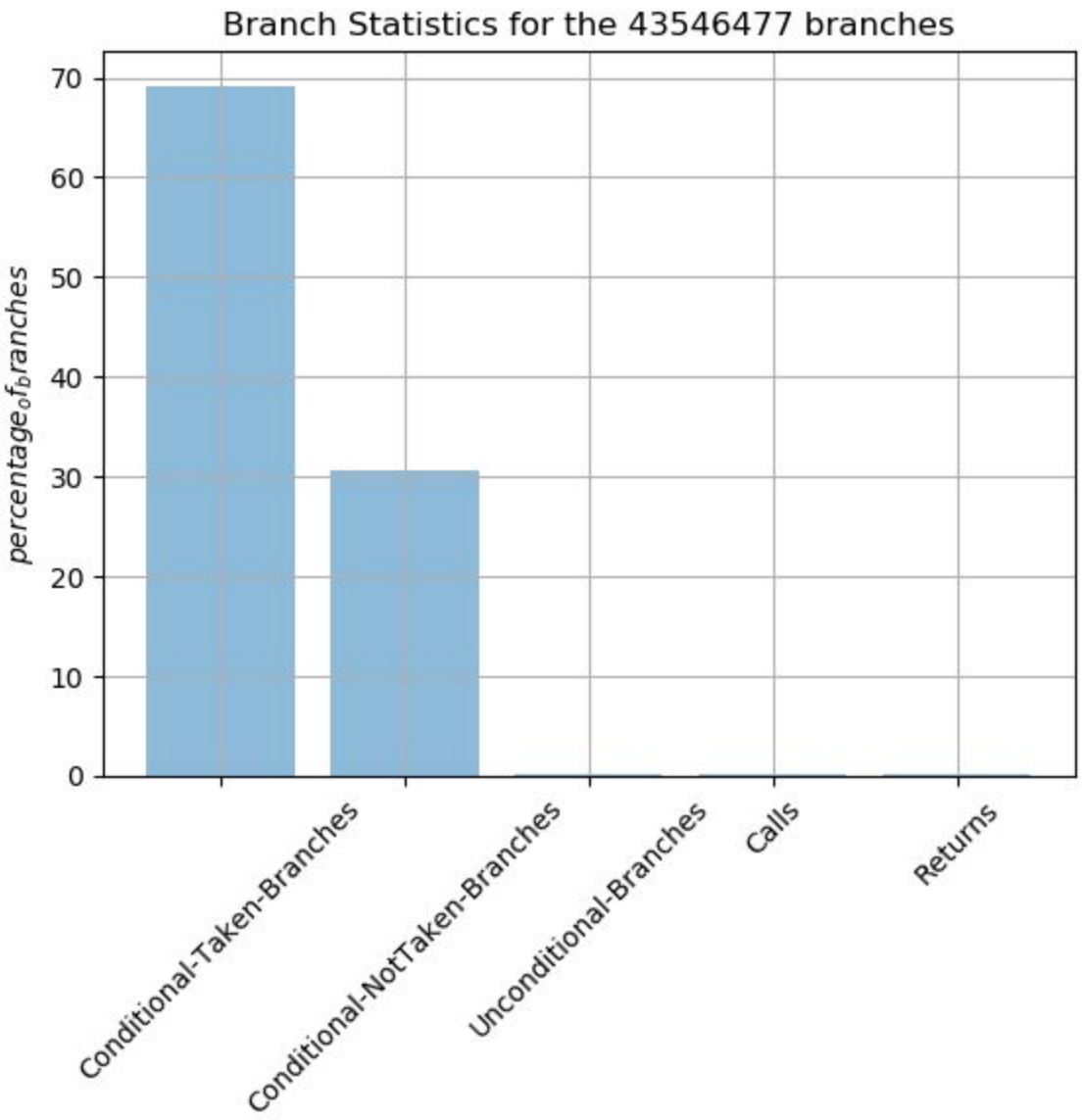
- 403.gcc



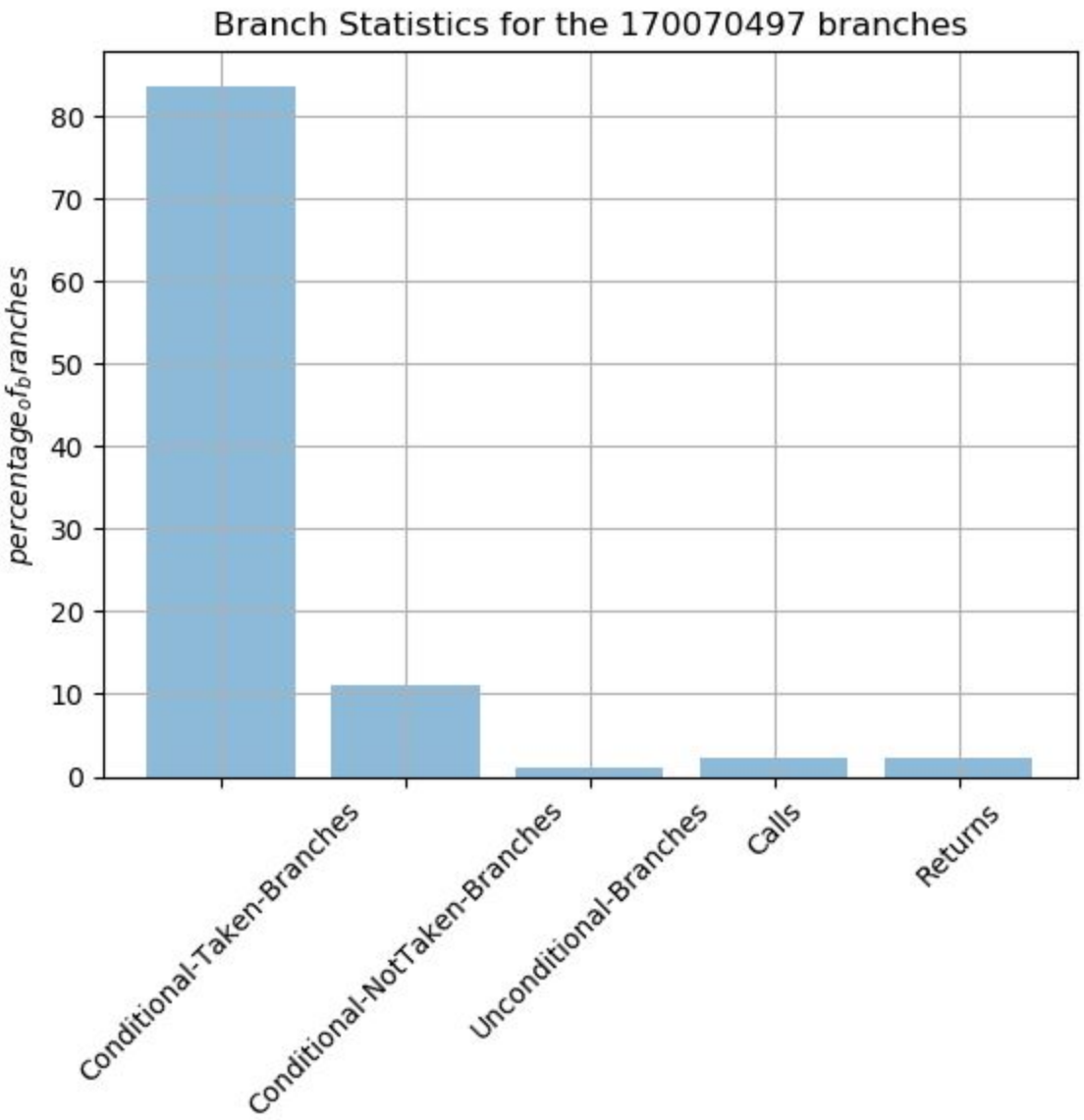
- 429.mcf


Branch Statistics for the 3862679002 branches
percentage of branches
0
10
20
30
40
50
60
Conditional-Taken-Branches
Conditional-NotTaken-Branches
Unconditional-Branches
Calls
Returns

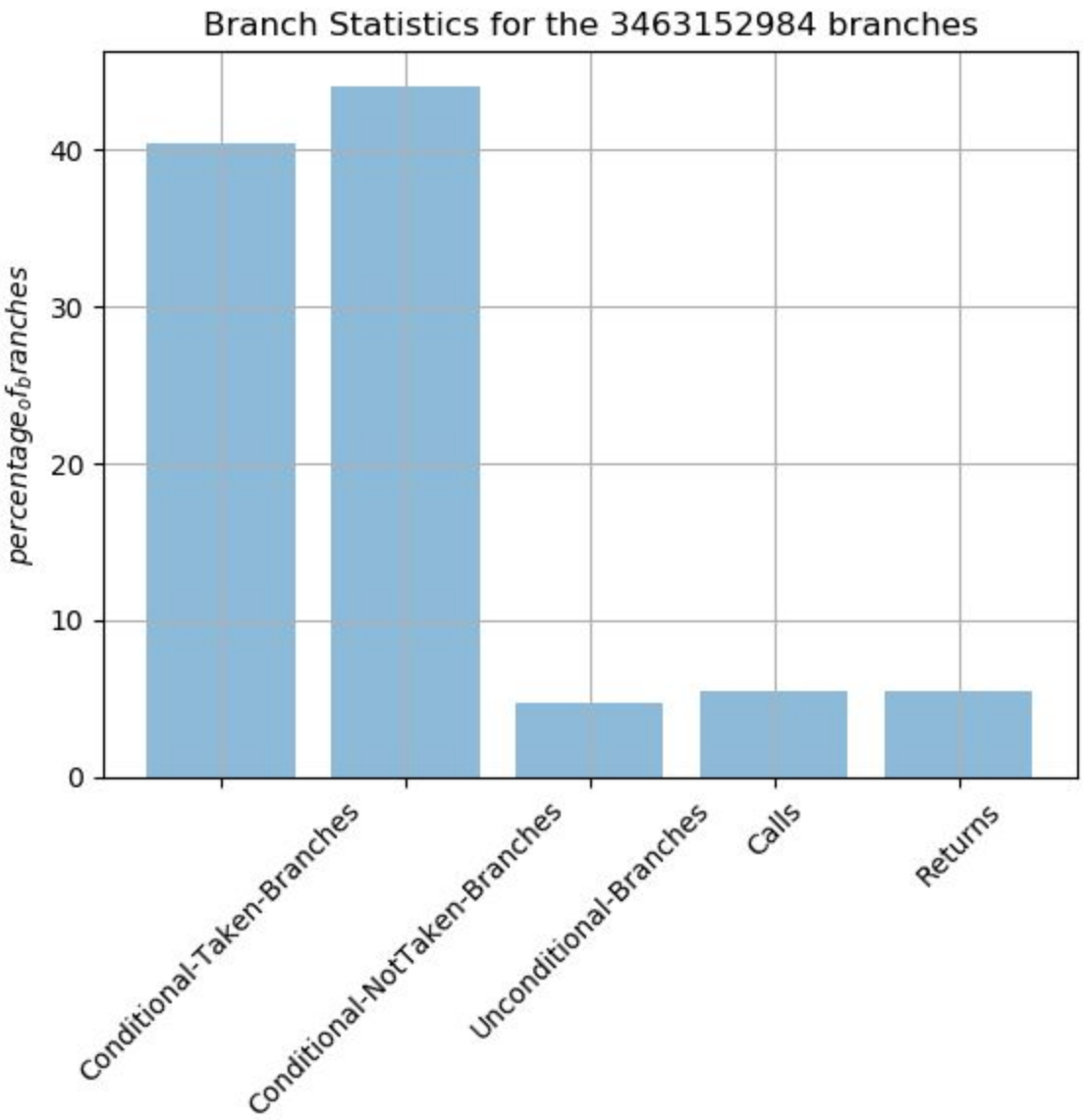
- 434.zeusmp



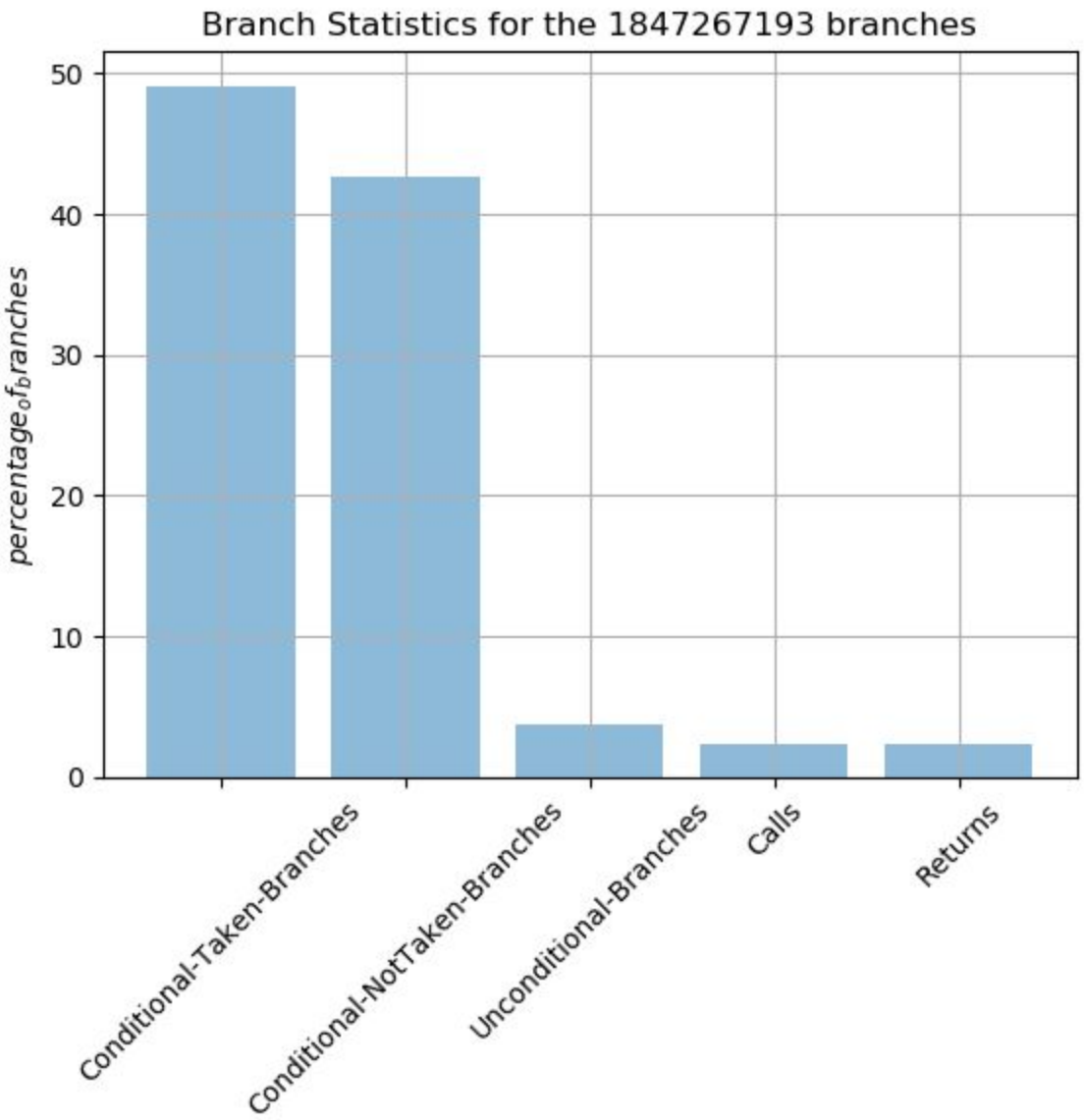
➢ 436.cactusADM


Branch Statistics for the 170070497 branches
percentage of branches
0
10
20
30
40
50
60
70
80
Conditional-Taken-Branches
Conditional-NotTaken-Branches
Unconditional-Branches
Calls
Returns

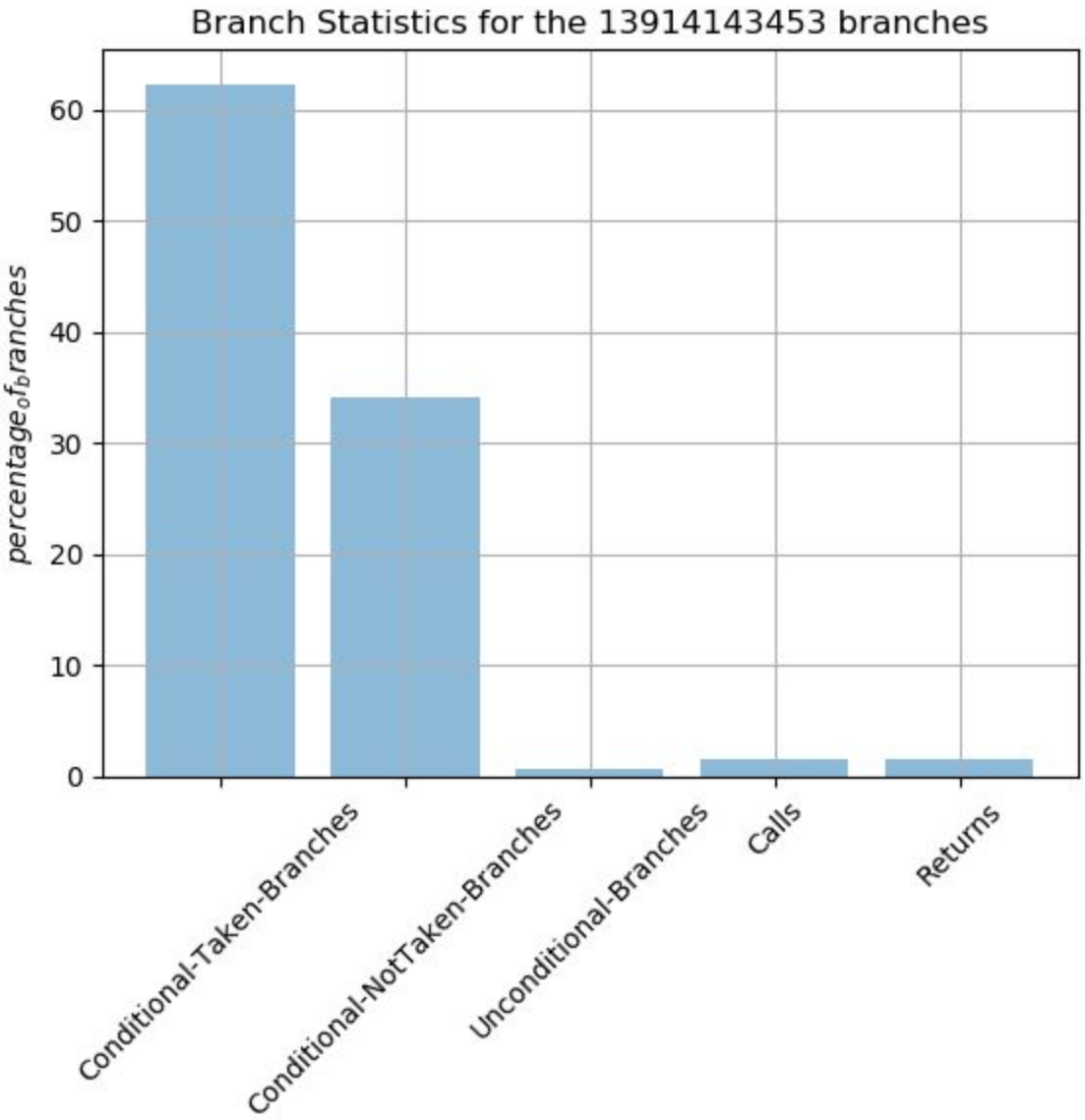
- 445.gobmk


Branch Statistics for the 3463152984 branches
percentage of branches
0
10
20
30
40
Conditional-Taken-Branches
Conditional-NotTaken-Branches
Unconditional-Branches
Calls
Returns

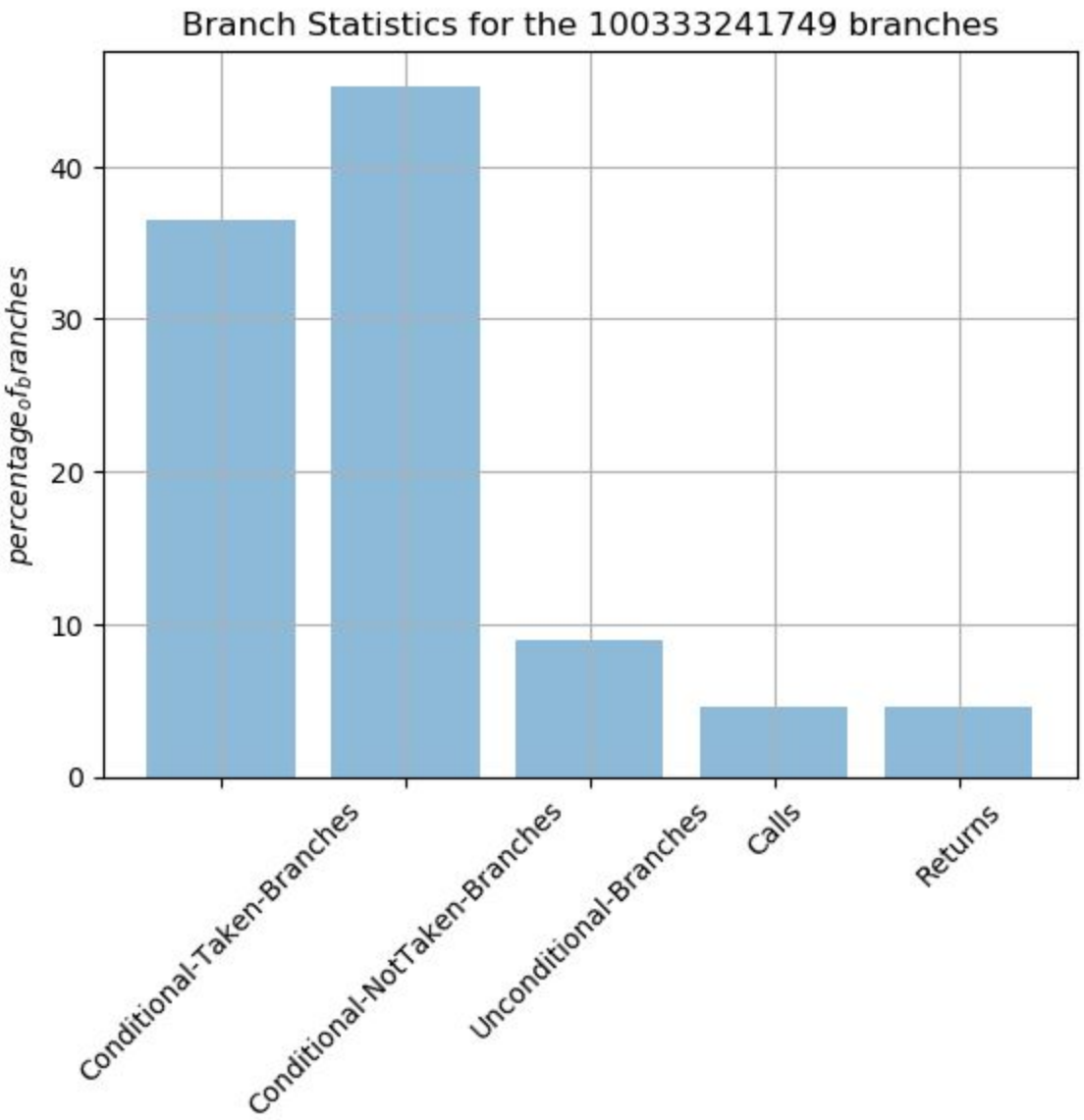
➢ 450.soplex


Branch Statistics for the 1847267193 branches
percentage of branches
0
10
20
30
40
50
Conditional-Taken-Branches
Conditional-NotTaken-Branches
Unconditional-Branches
Calls
Returns

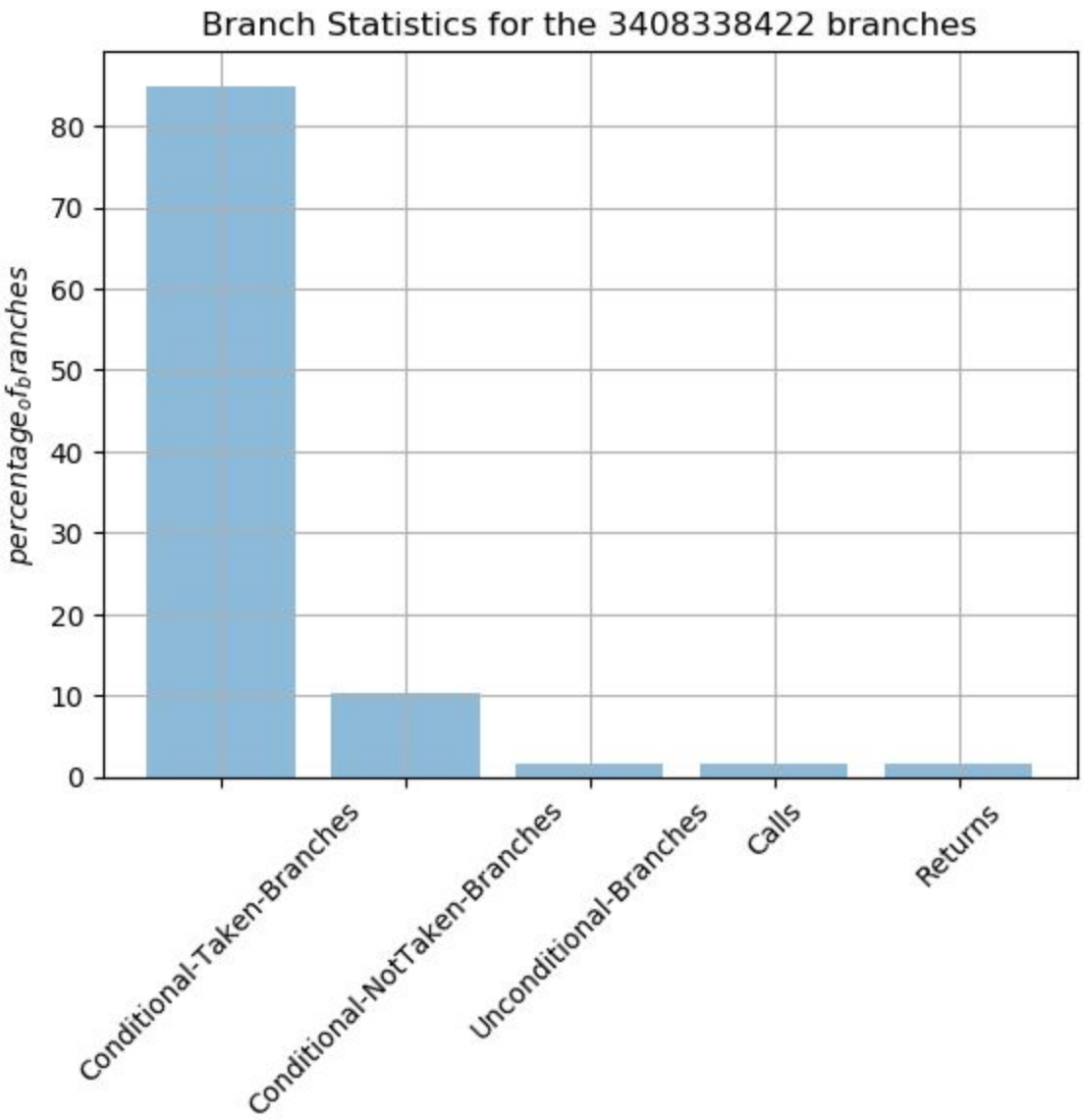
- 456.hmmer


Branch Statistics for the 13914143453 branches
percentage of branches
0
10
20
30
40
50
60
Conditional-Taken-Branches
Conditional-NotTaken-Branches
Unconditional-Branches
Calls
Returns

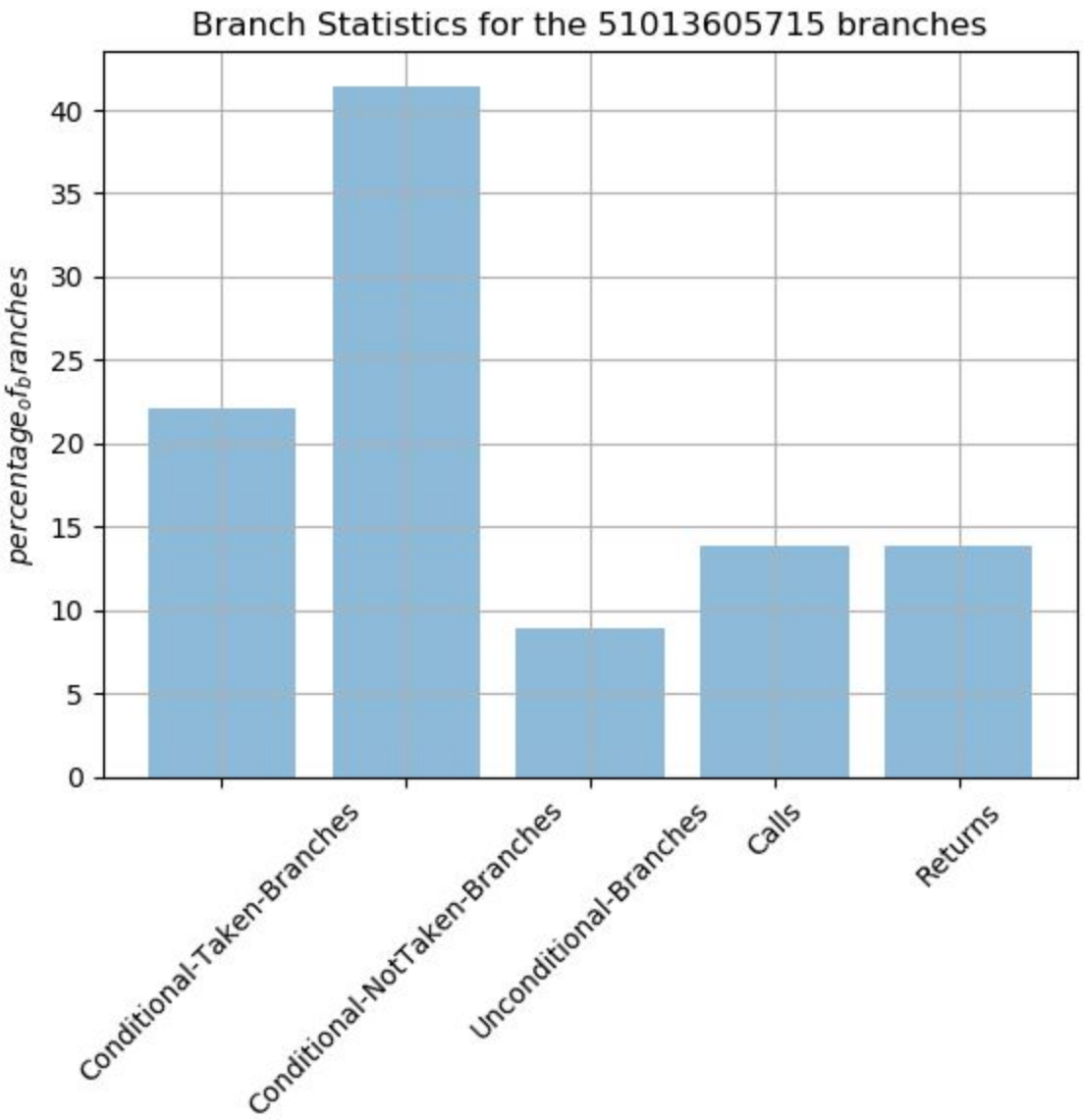
- 458.sjeng



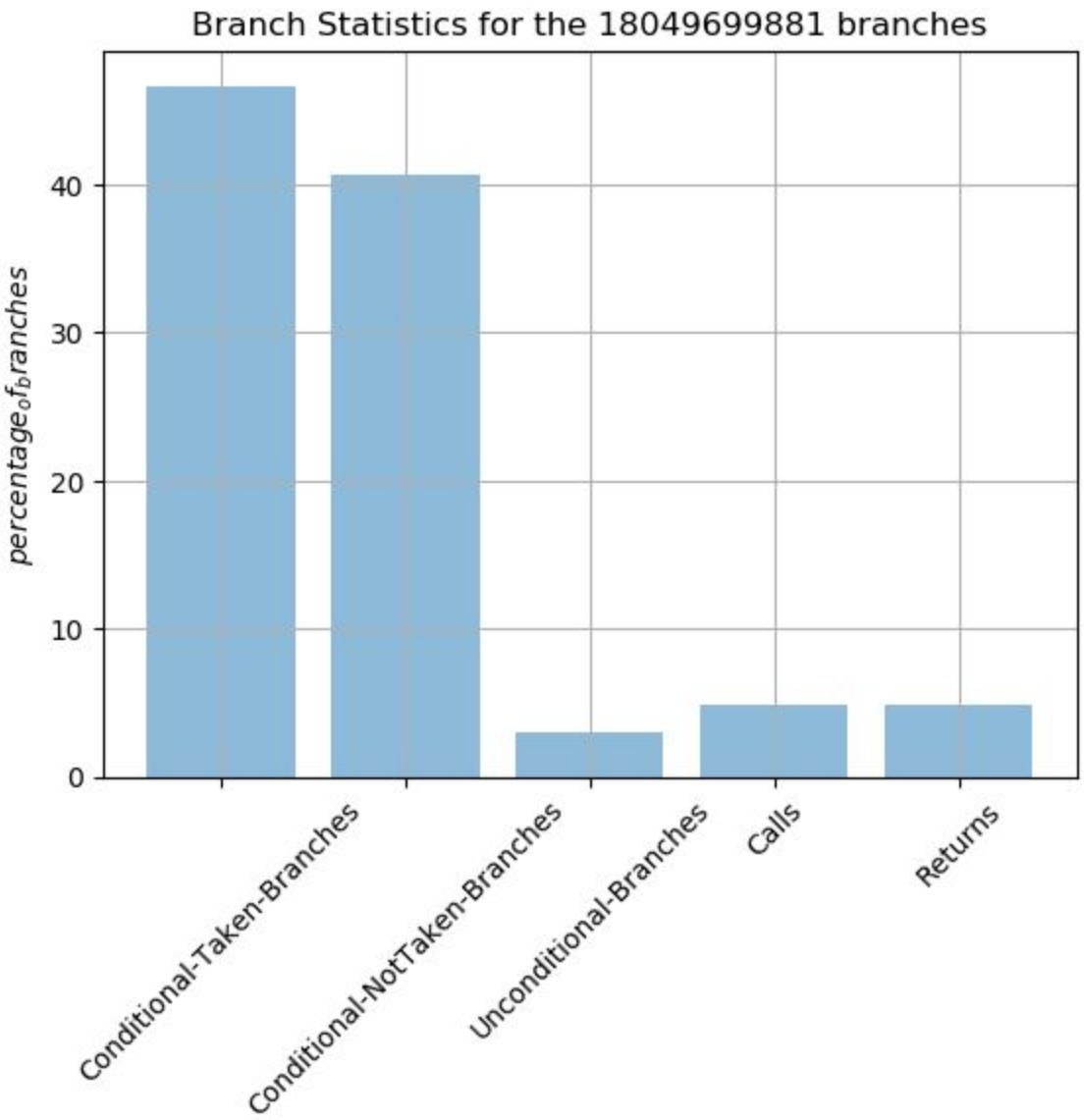
➢ 459.GemsFDTD


Branch Statistics for the 3408338422 branches
percentage of branches
0
10
20
30
40
50
60
70
80
Conditional-Taken-Branches
Conditional-NotTaken-Branches
Unconditional-Branches
Calls
Returns

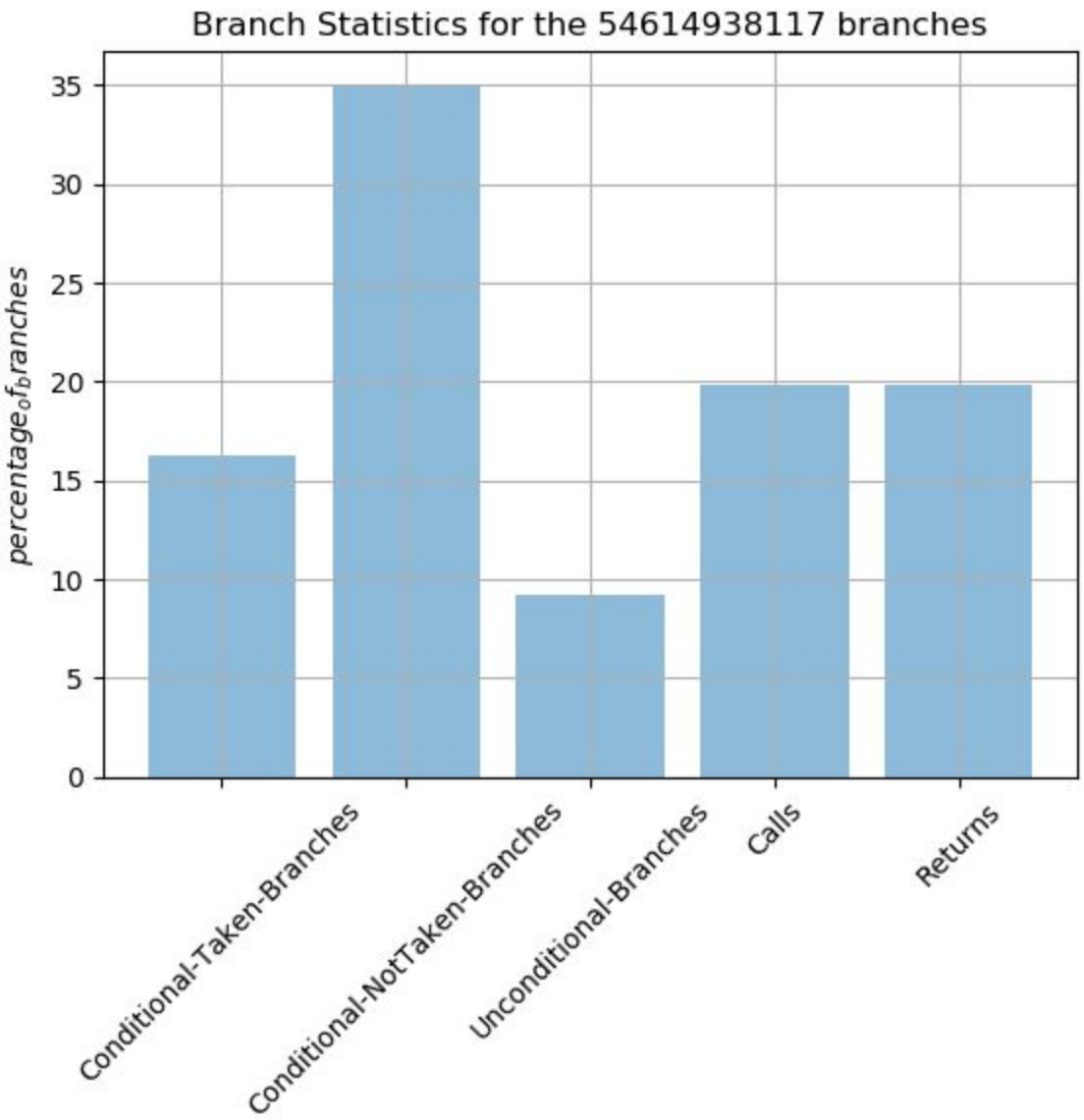
➢ 471.omnetpp


Branch Statistics for the 51013605715 branches
percentage of branches
0
5
10
15
20
25
30
35
40
Conditional-Taken-Branches
Conditional-NotTaken-Branches
Unconditional-Branches
Calls
Returns

➢ 473.astar



➢ 483.xalancbmk



Συμπεράσματα

Καταρχάς βλέπουμε πως σε κάθε benchmark υπάρχει ένα αξιοσημείωτο πλήθος εντολών άλματος. Το μεγαλύτερο ποσοστό αυτών είναι conditional taken και conditional not taken branches ενώ σε πολύ μικρότερο ποσοστό συναντάμε unconditional branches, calls και returns, όπως θα περιμέναμε.
Οι τρεις τελευταίες κατηγορίες σε αρκετά μετροπρογράμματα πλησιάζουν στο 0 (όπως στα mcf, zeusmp, cactusADM, hmmer) ενώ σε ορισμένα έχουν πιο σημαντικό ποσοστό (όπως στα omnetpp, xalancbmk).

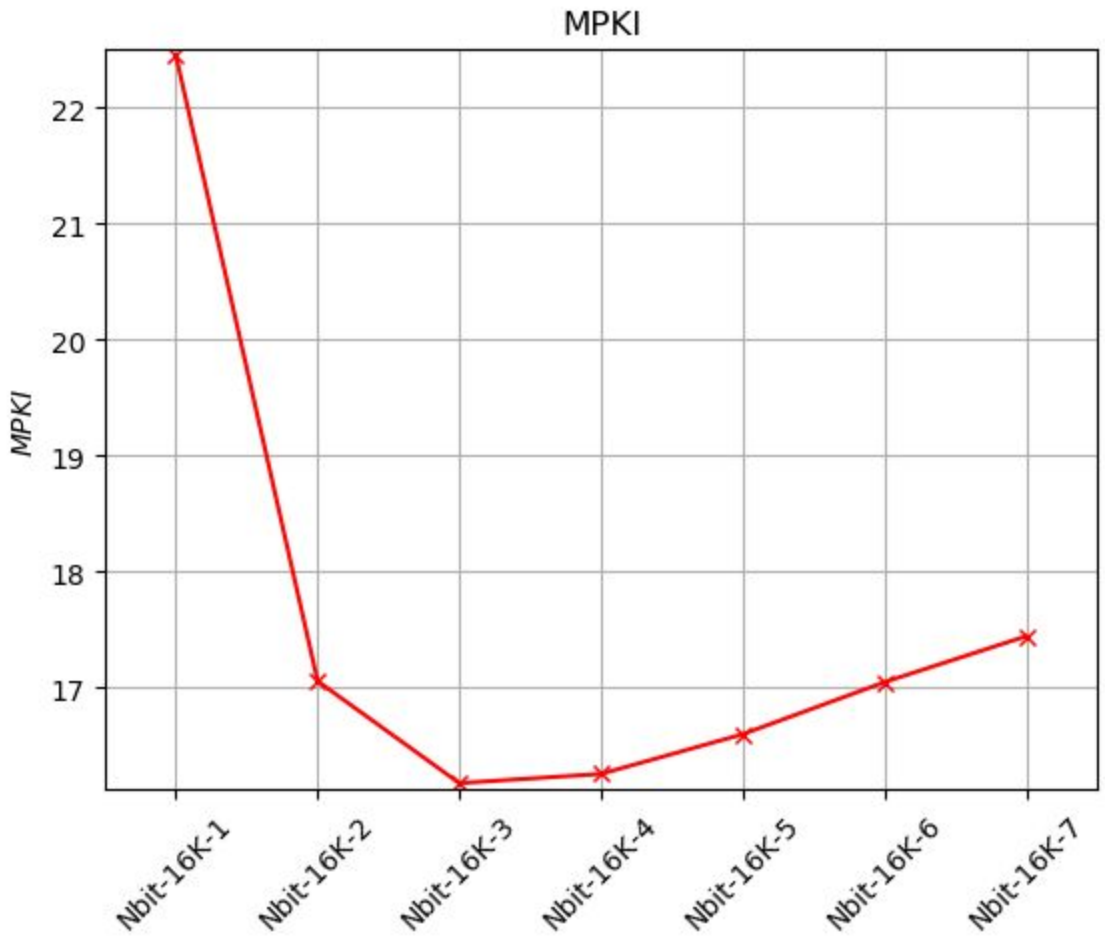
4.2 Μελέτη των N-bit predictors

Εδώ εξετάσαμε τους n-bit predictors και την απόδοση τους σε σχέση με τα BHT entries και το N (και ως αποτέλεσμα το συνολικό μέγεθος σε hardware). Ως μετρικό της απόδοσης έχουμε το direction MPKI (direction Mispredictions Per Thousand Instructions).

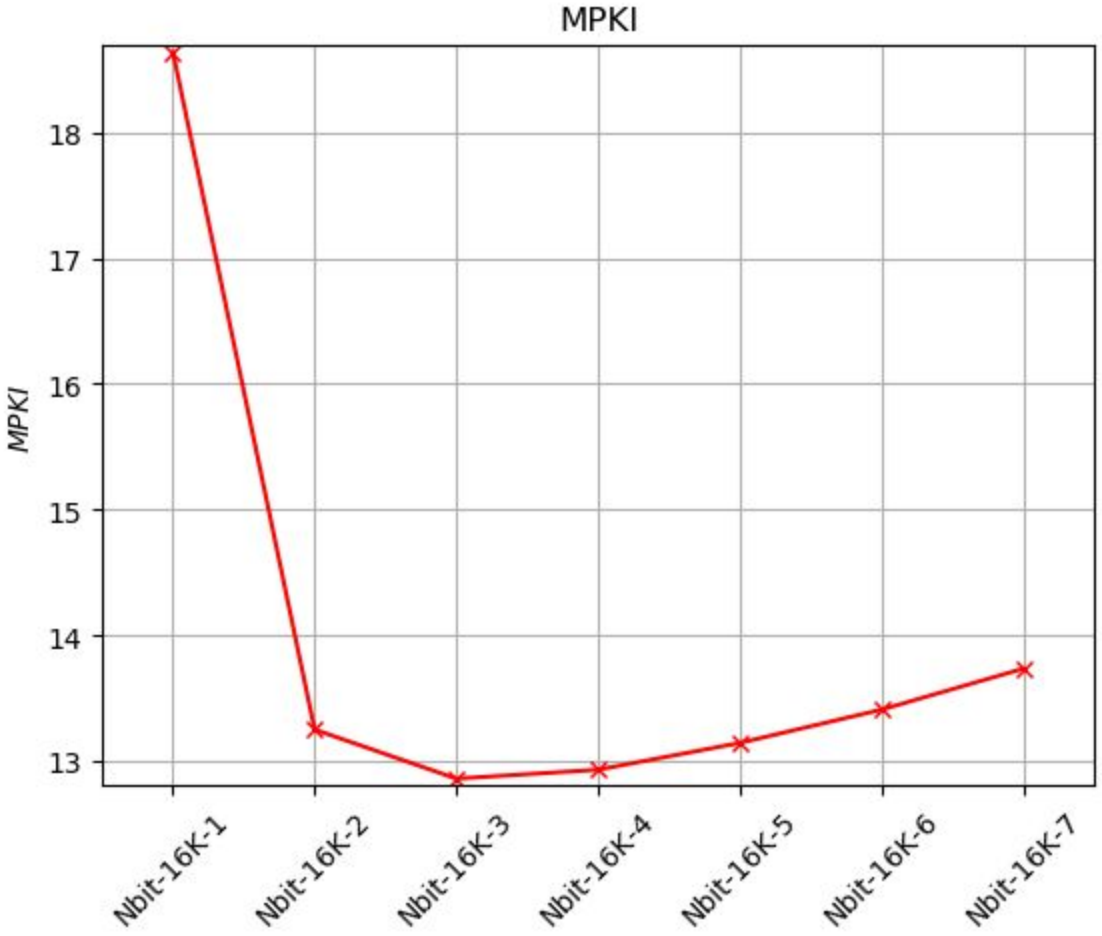
4.2 i

Εδώ διατηρήσαμε το μέγεθος των BHT entries σταθερό και ίσο με 16K και συγκρίναμε τα αποτελέσματα για κάθε benchmark για N κυμαινόμενο από 1 εώς και 7.

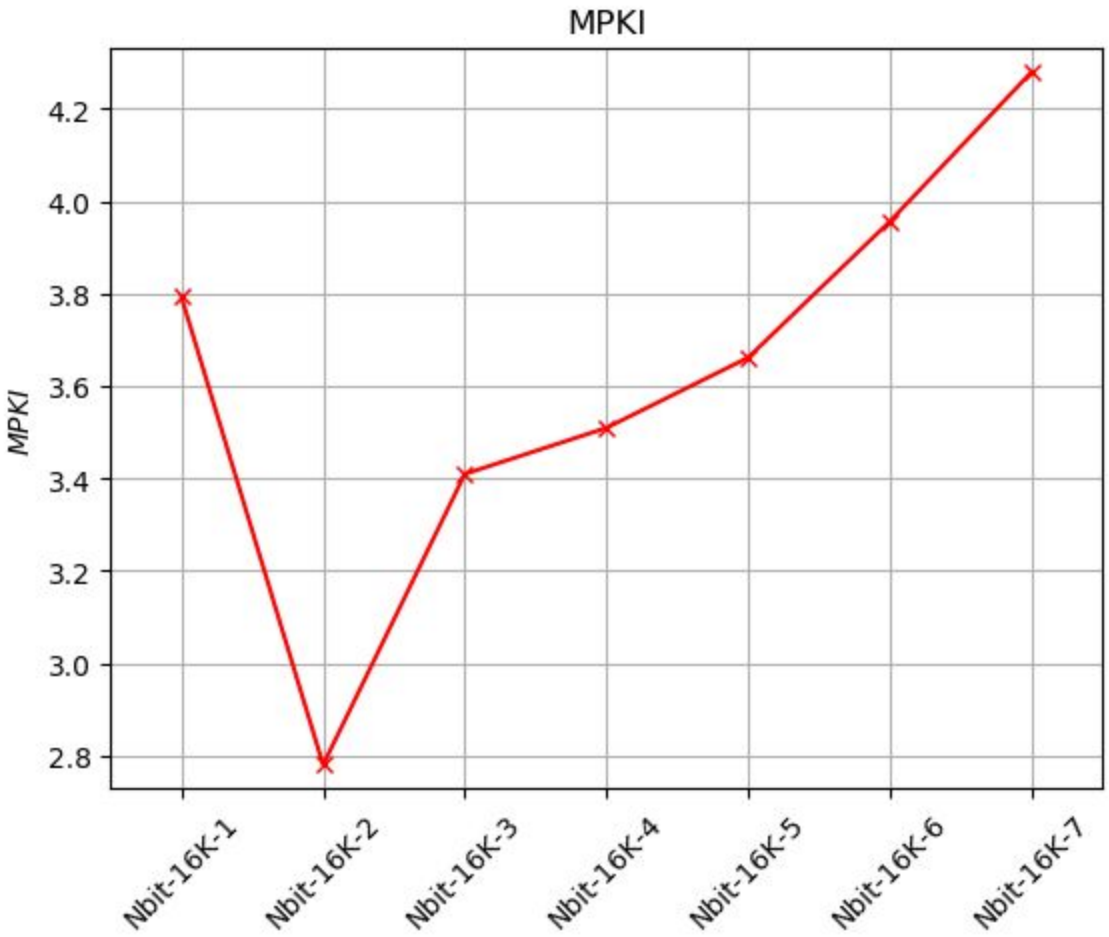
- 403.gcc



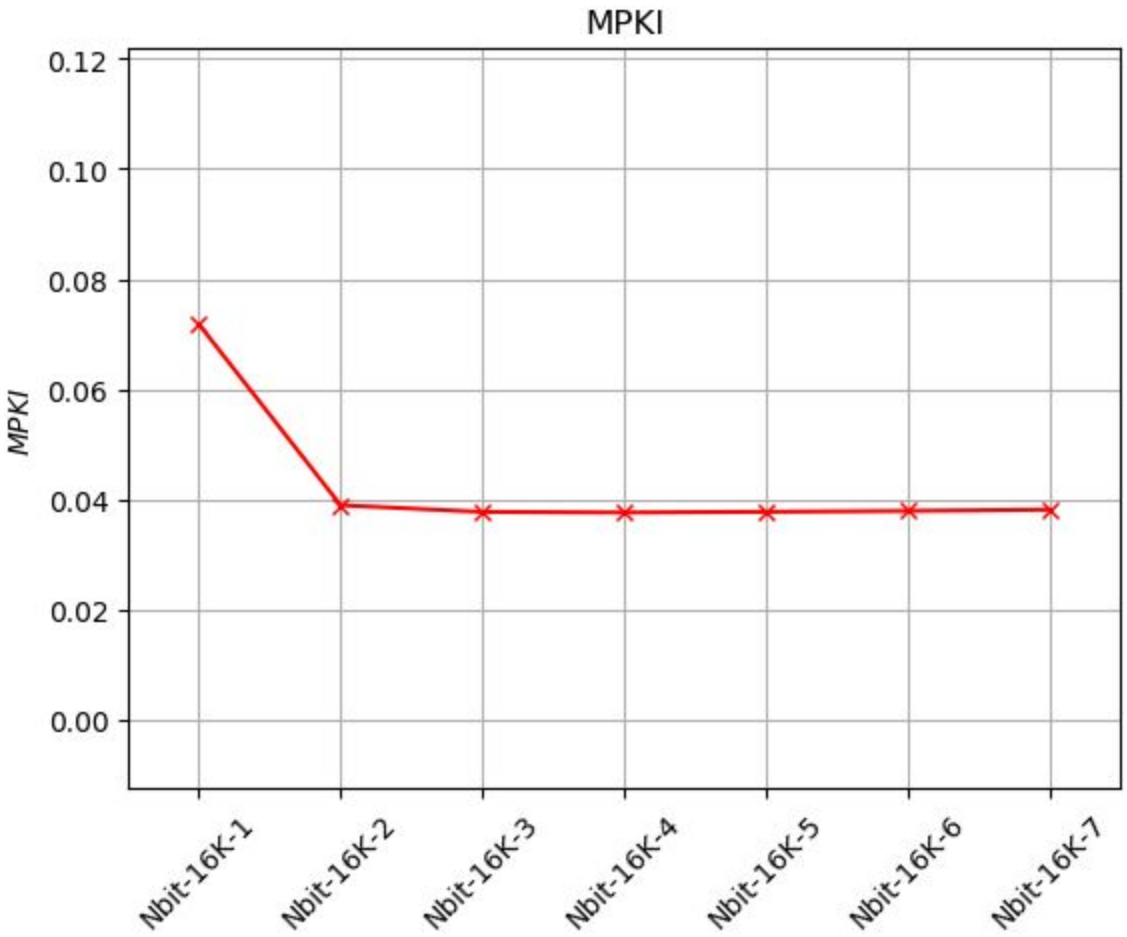
- 429.mcf


MPKI
MPKI
18
17
16
15
14
13
Nbit-16K-1
Nbit-16K-2
Nbit-16K-3
Nbit-16K-4
Nbit-16K-5
Nbit-16K-6
Nbit-16K-7

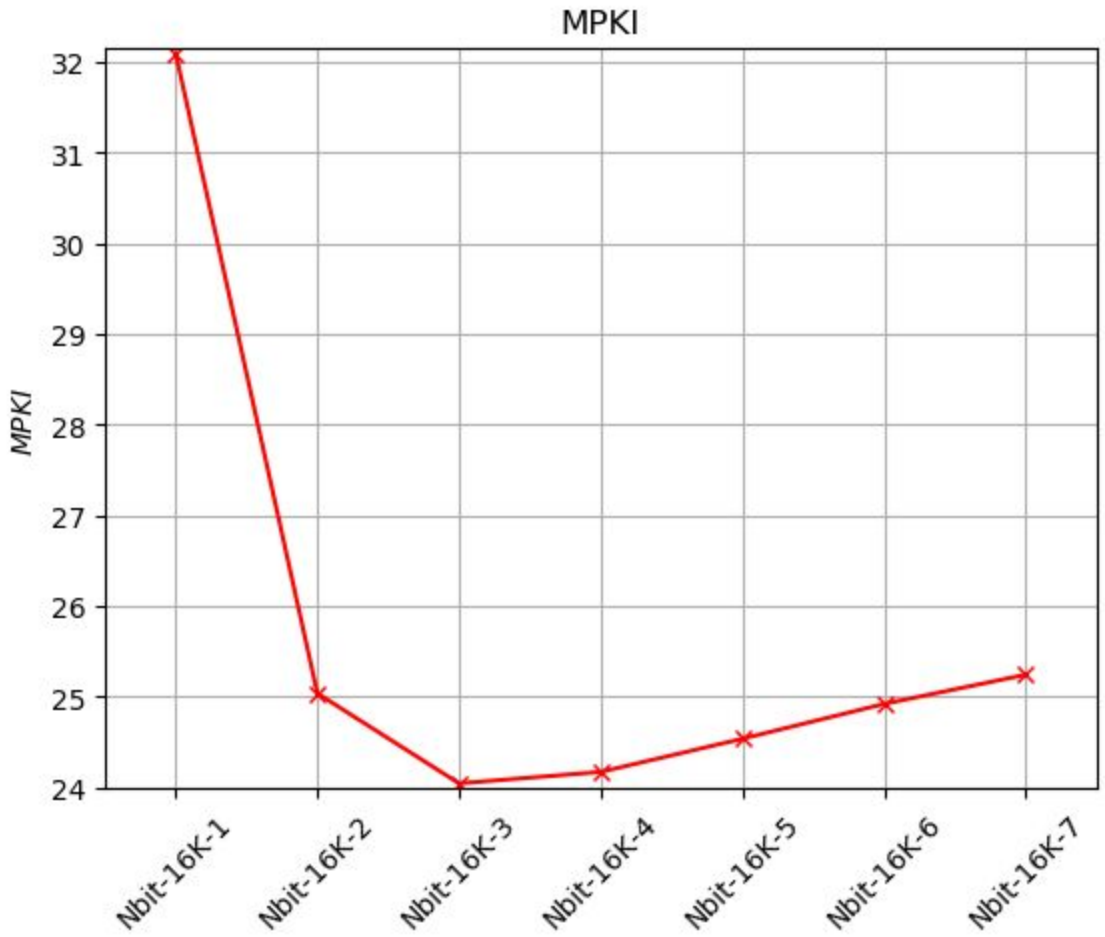
➢ 434.zeusmp



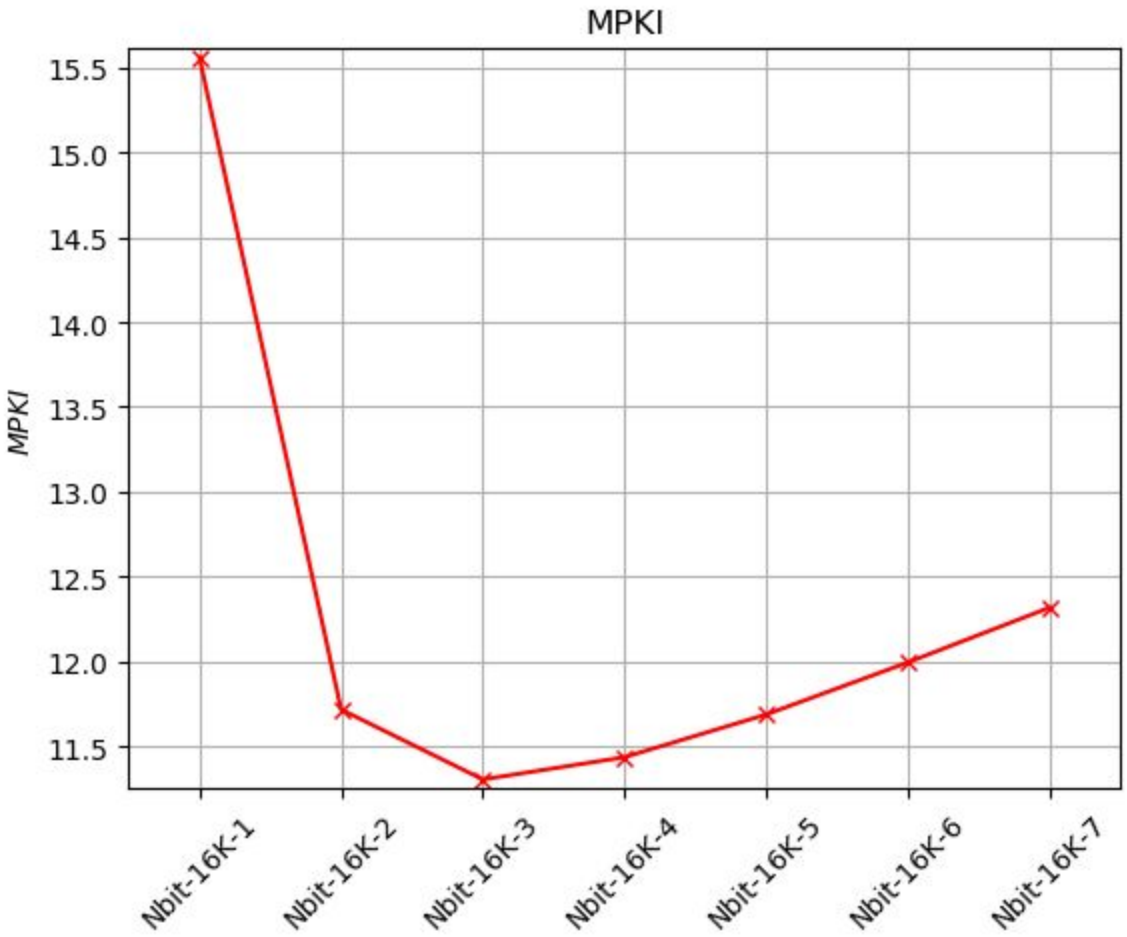
➢ 436.cactusADM


MPKI
0.12
0.10
0.08
0.06
0.04
0.02
0.00
MPKI
Nbit-16K-1
Nbit-16K-2
Nbit-16K-3
Nbit-16K-4
Nbit-16K-5
Nbit-16K-6
Nbit-16K-7

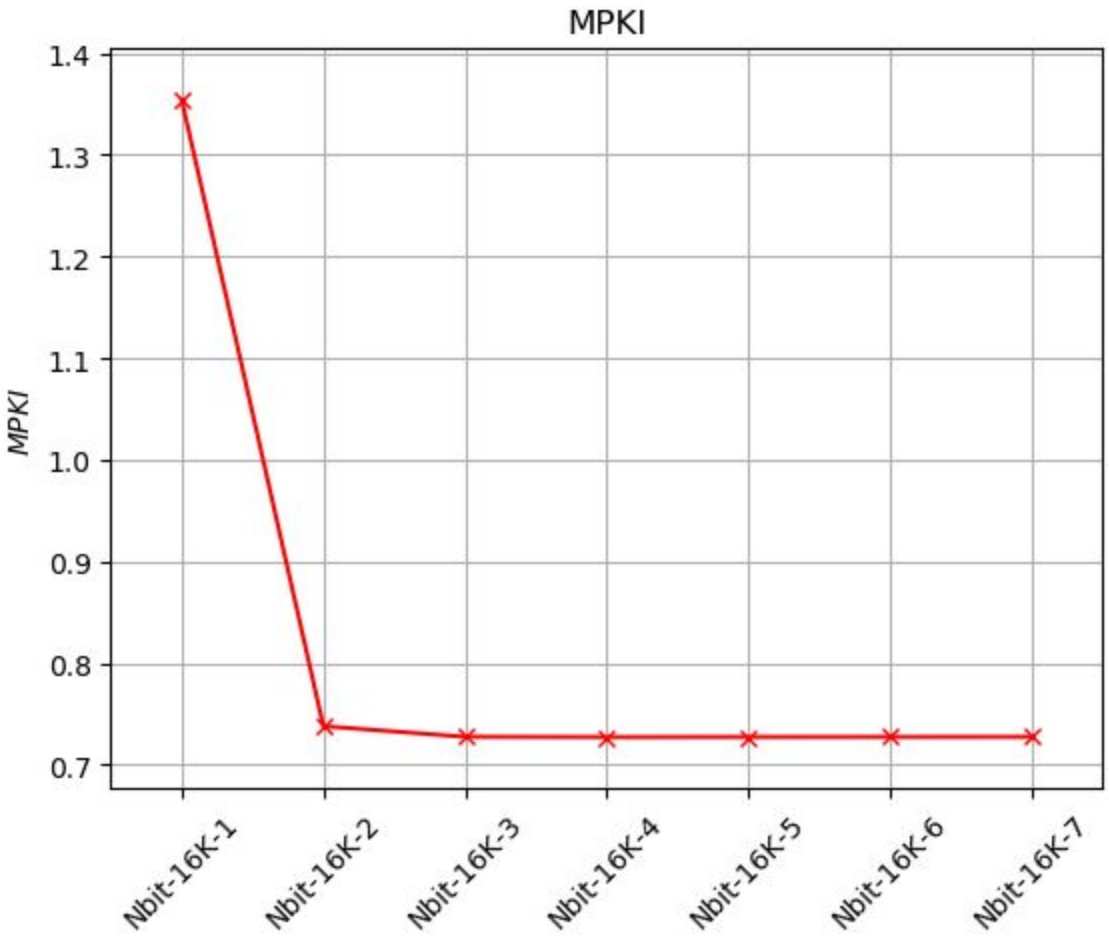
➢ 445.gobmk


MPKI
MPKI
32
31
30
29
28
27
26
25
24
Nbit-16K-1
Nbit-16K-2
Nbit-16K-3
Nbit-16K-4
Nbit-16K-5
Nbit-16K-6
Nbit-16K-7

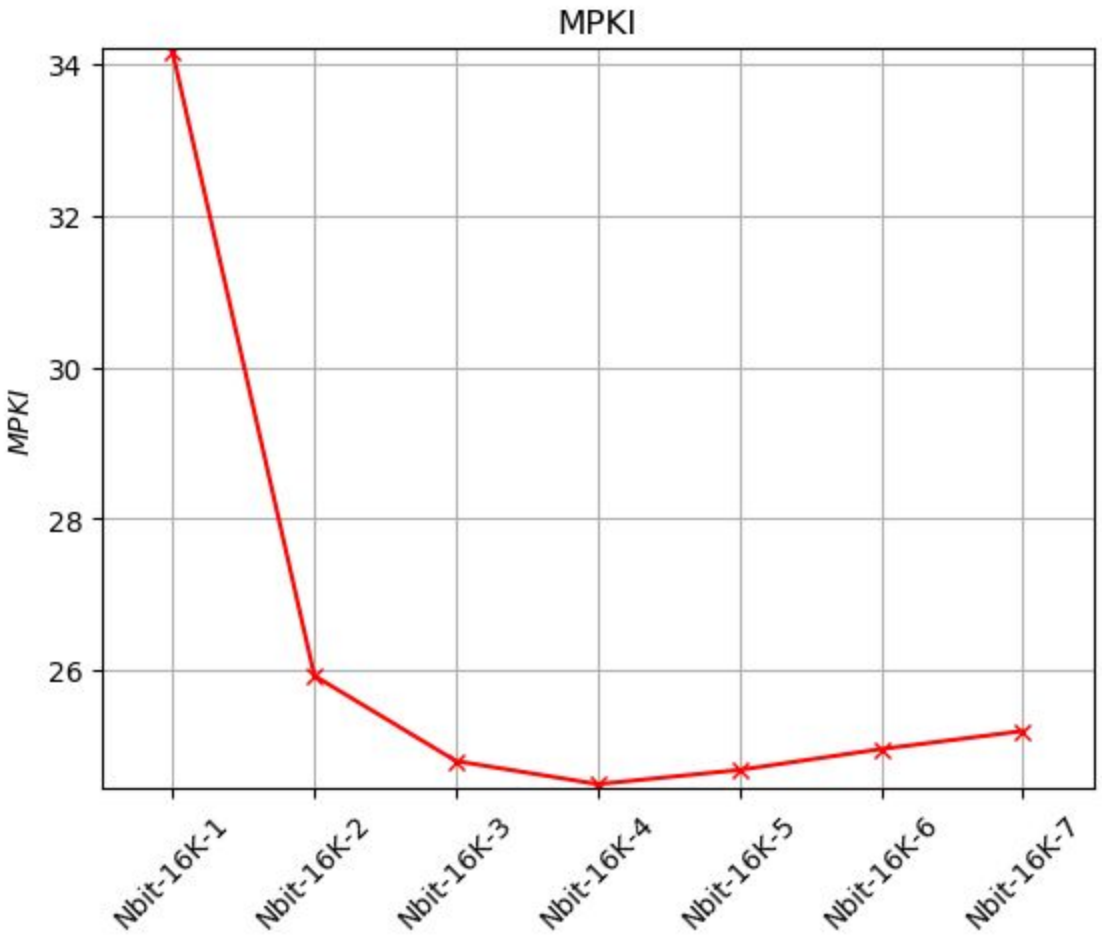
➢ 450.soplex


MPKI
MPKI
15.5
15.0
14.5
14.0
13.5
13.0
12.5
12.0
11.5
Nbit-16K-1
Nbit-16K-2
Nbit-16K-3
Nbit-16K-4
Nbit-16K-5
Nbit-16K-6
Nbit-16K-7

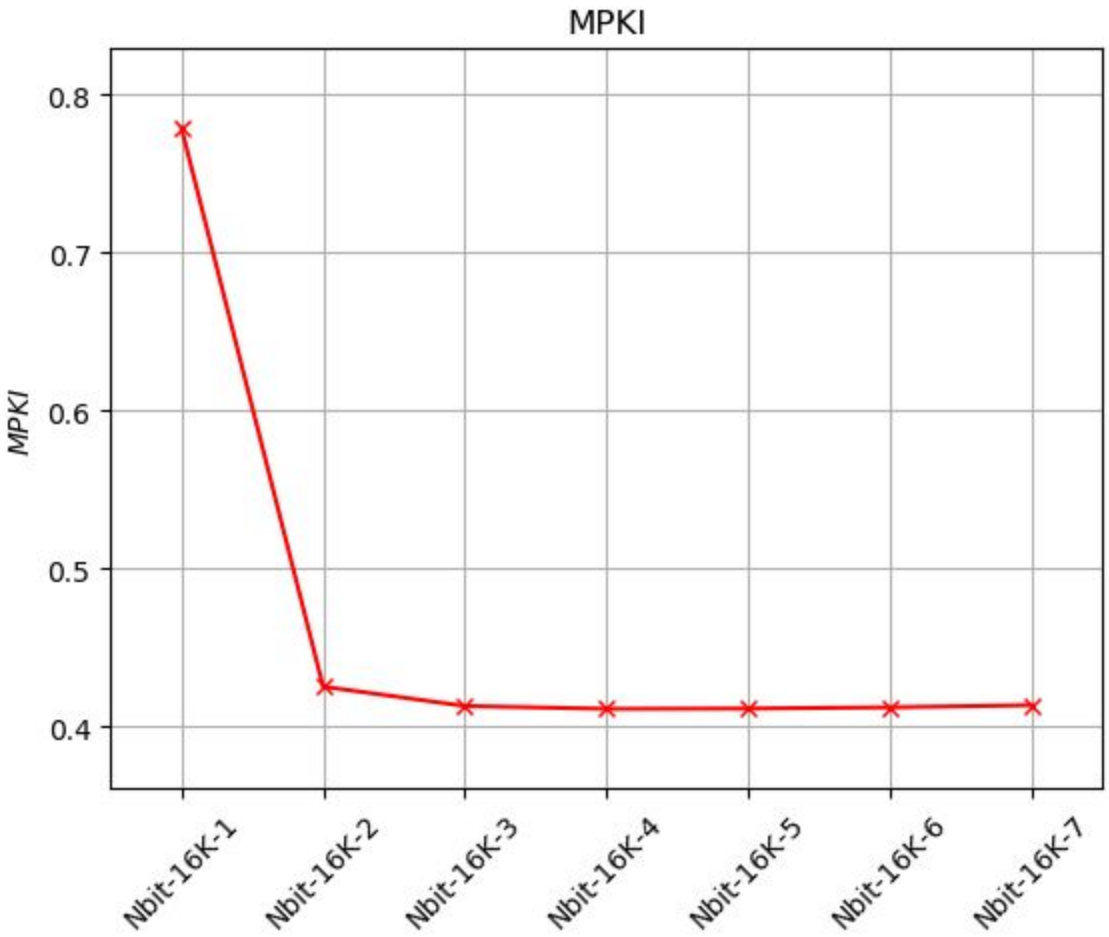
➢ 456.hmmer


MPKI
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
MPKI
Nbit-16K-1
Nbit-16K-2
Nbit-16K-3
Nbit-16K-4
Nbit-16K-5
Nbit-16K-6
Nbit-16K-7

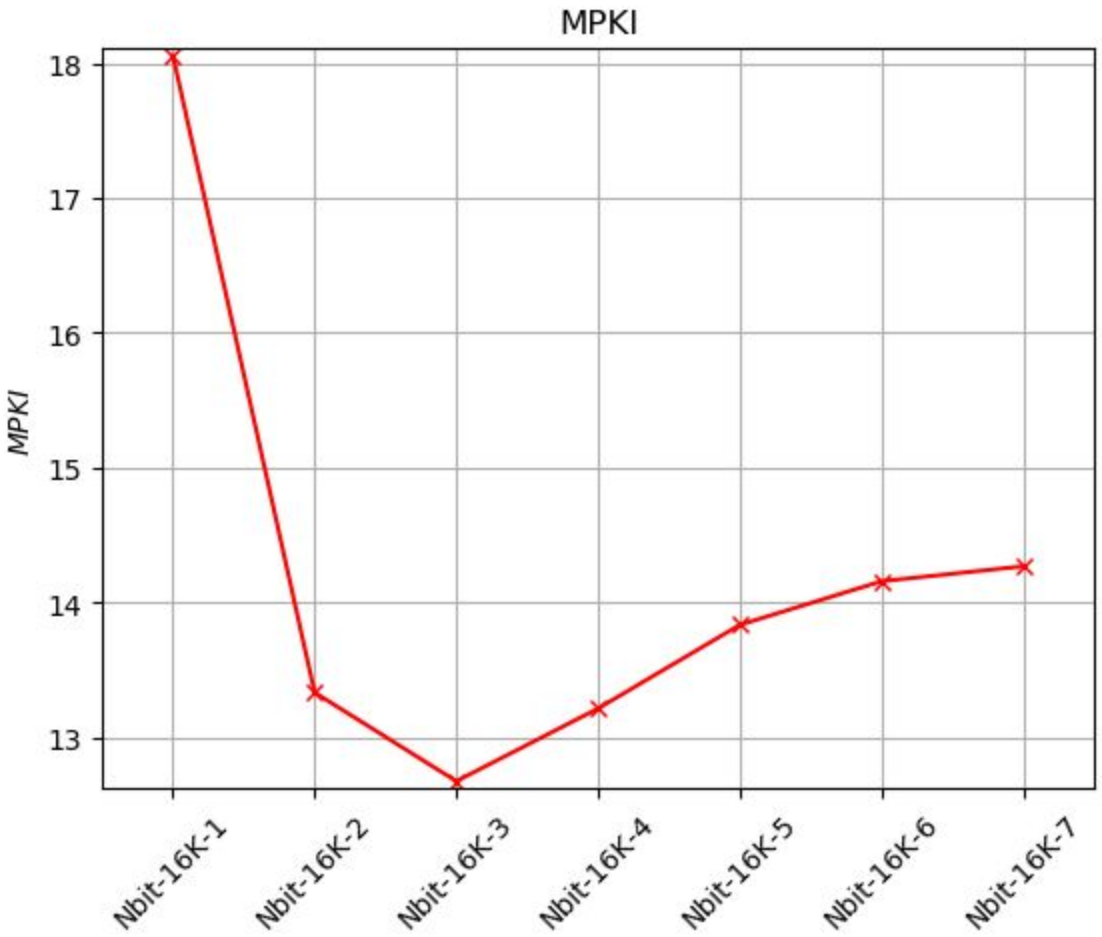
- 458.sjeng



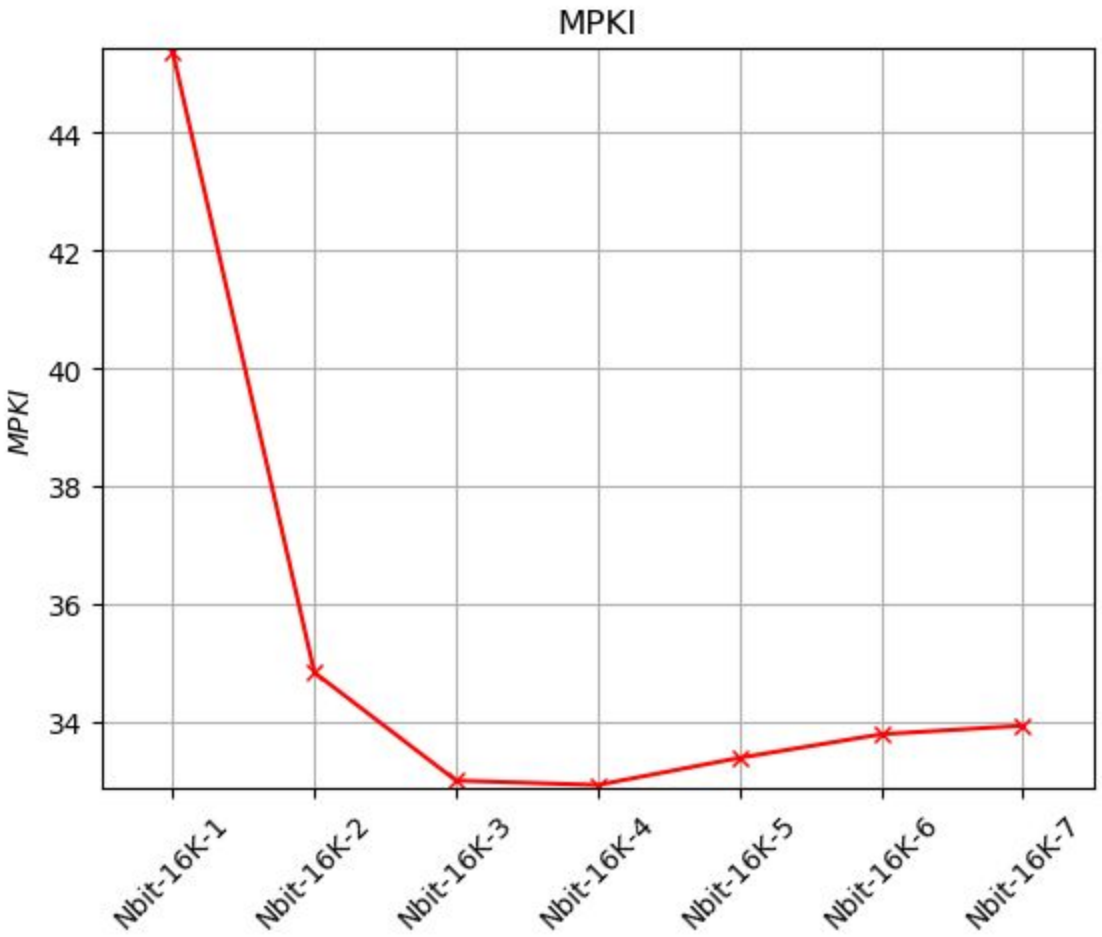
➢ 459.GemsFDTD


MPKI
MPKI
0.8
0.7
0.6
0.5
0.4
Nbit-16K-1
Nbit-16K-2
Nbit-16K-3
Nbit-16K-4
Nbit-16K-5
Nbit-16K-6
Nbit-16K-7

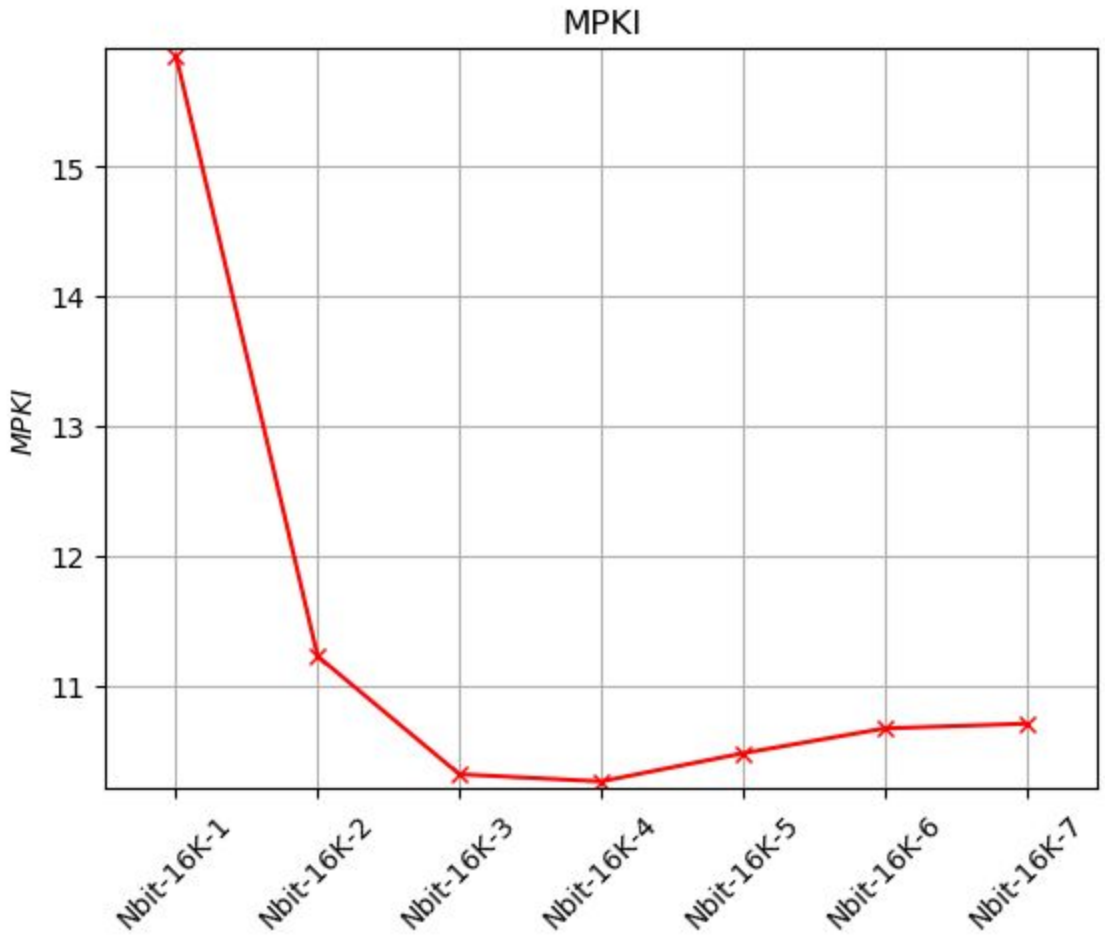
➢ 471.omnetpp


MPKI
MPKI
18
17
16
15
14
13
Nbit-16K-1
Nbit-16K-2
Nbit-16K-3
Nbit-16K-4
Nbit-16K-5
Nbit-16K-6
Nbit-16K-7

➢ 473.astar



➢ 483.xalancbmk



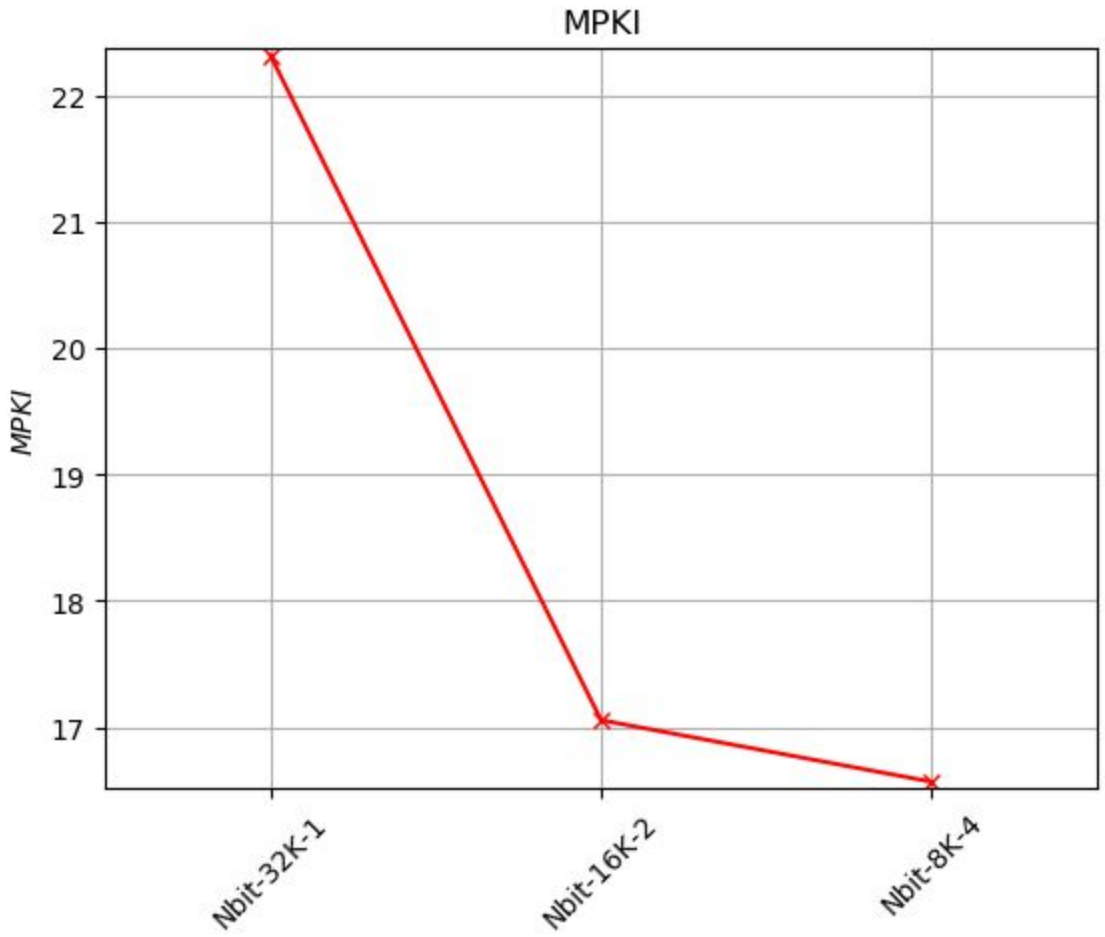
Συμπεράσματα

Παρατηρούμε πως για N=1 παρατηρείται πολύ μεγάλο miss prediction. Πηγαίνοντας για N=2 παρατηρείται η μεγαλύτερη μεταβολή και έχουμε πολύ καλά αποτελέσματα. Συνεχίζοντας για N=3 είτε υπάρχει μια βελτίωση είτε παραμένει σταθερή η απόδοση, με εξαίρεση την περίπτωση του zeusmp που έχουμε έντονη μείωση της απόδοσης. Αυξάνοντας το N πάνω από 3 είτε η απόδοση μένει περίπου σταθερή είτε μειώνεται. Επομένως αν επιλεγαμε θα είχαμε N=2 ή N=3, αναλόγως και την διαφορά τους στην τιμή μιας και για N=3 έχουμε και περισσότερο υλικό hardware.

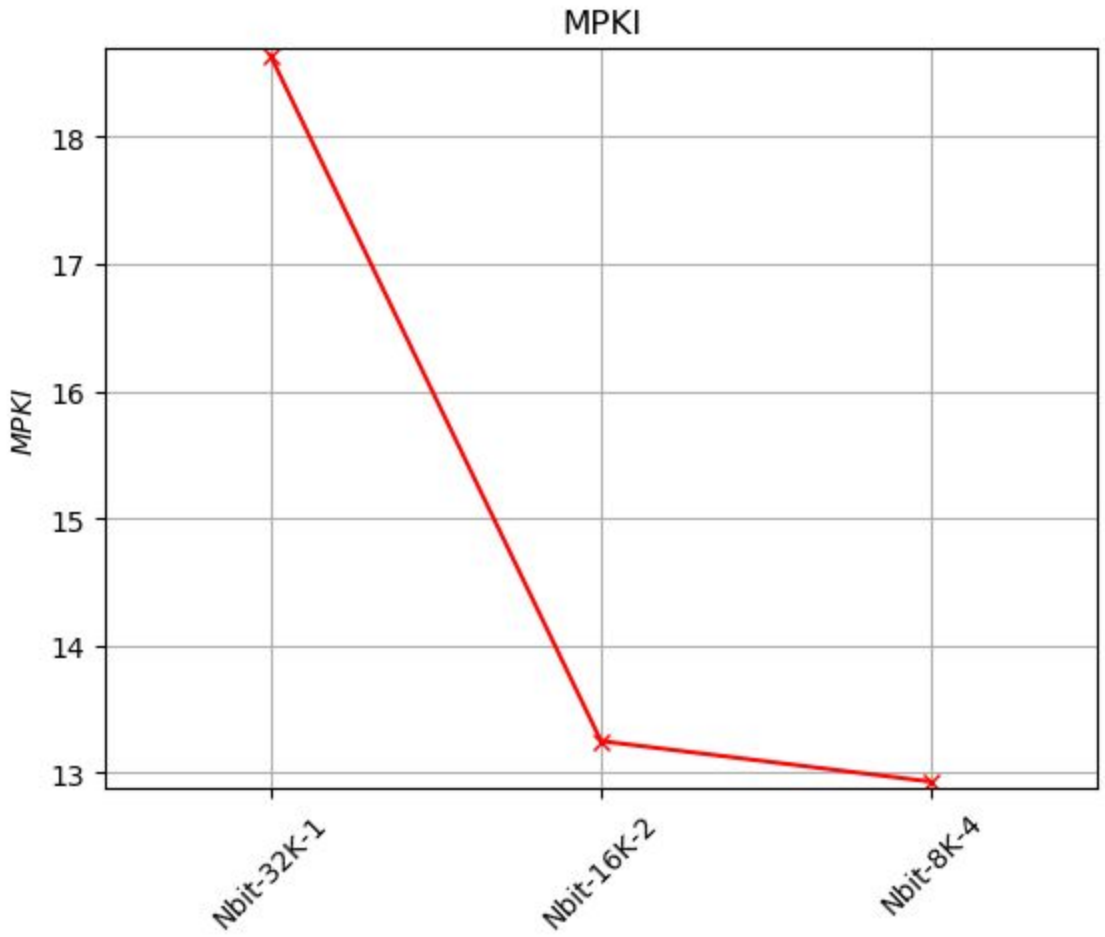
4.2 ii

Εδώ επιθυμούμε να κρατήσουμε το απαιτούμενο hardware σταθερό και ίσο με 32K bits, για N=1,2,4. Έτσι θέτουμε τα BHT entries 32K, 16K και 8K αντίστοιχα. Οι μεταβολές φαίνονται στα παρακάτω διαγράμματα.

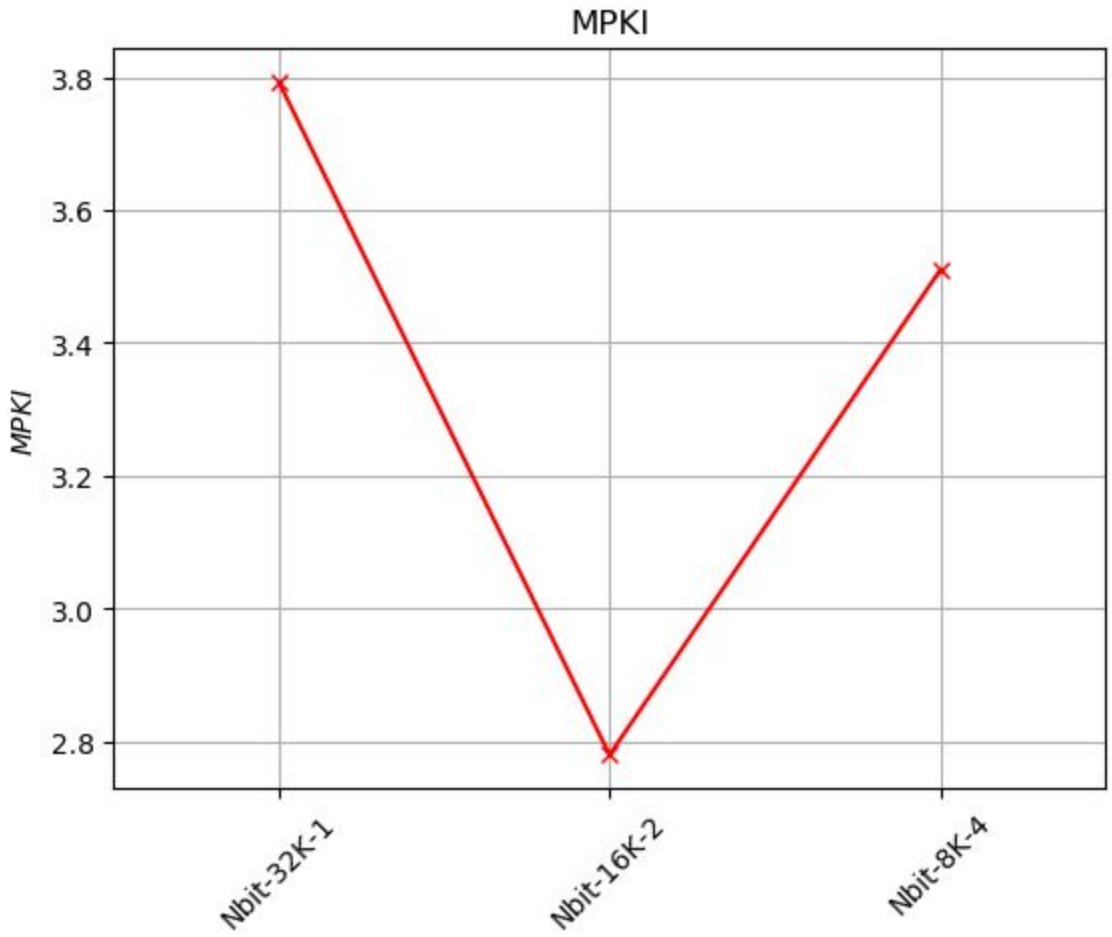
- 403.gcc



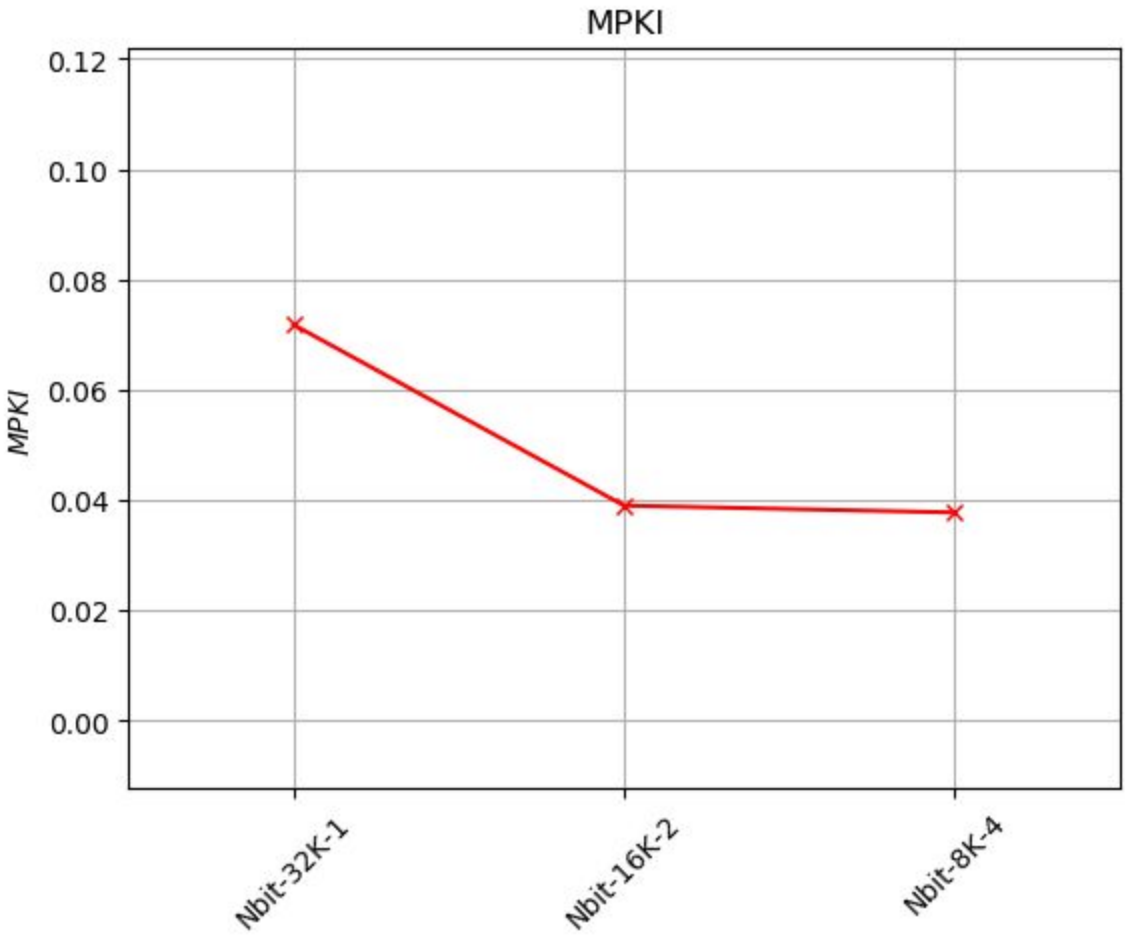
➢ 429.mcf



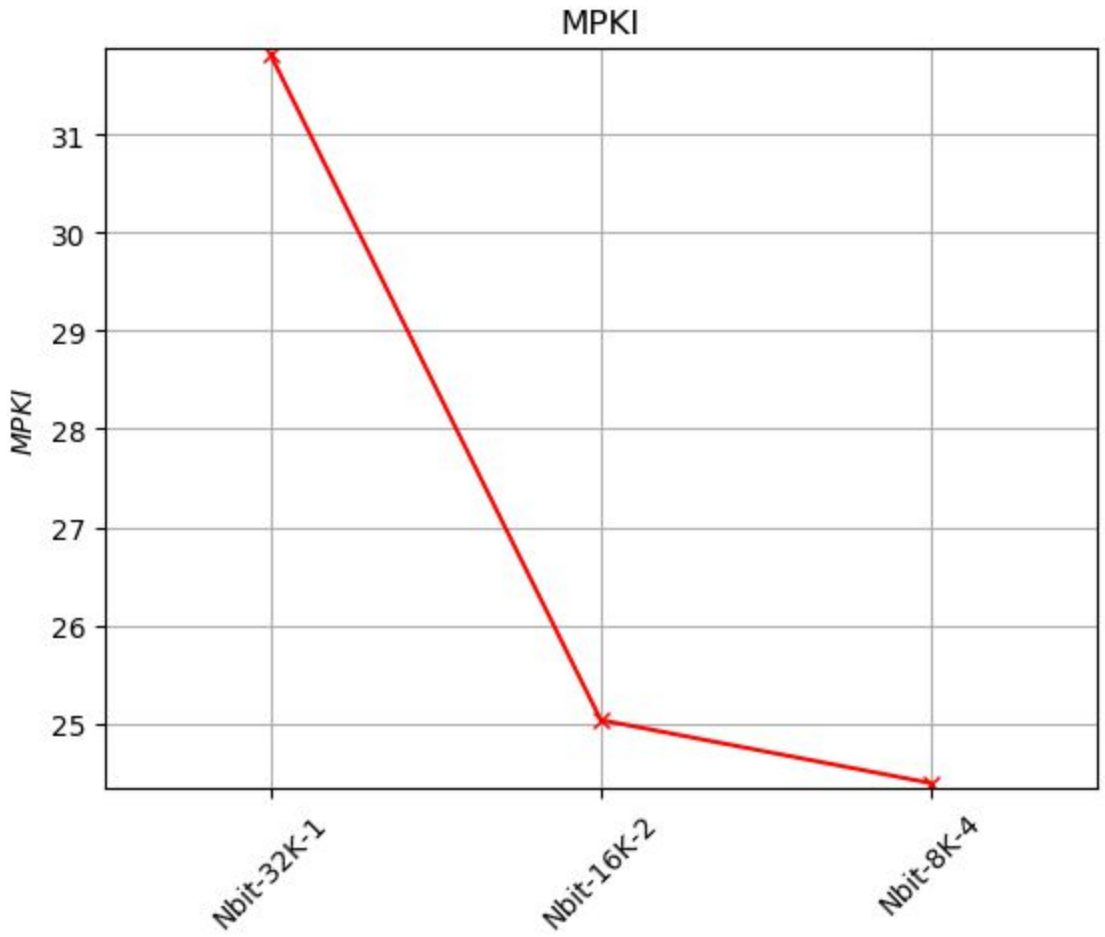
➢ 434.zeusmp



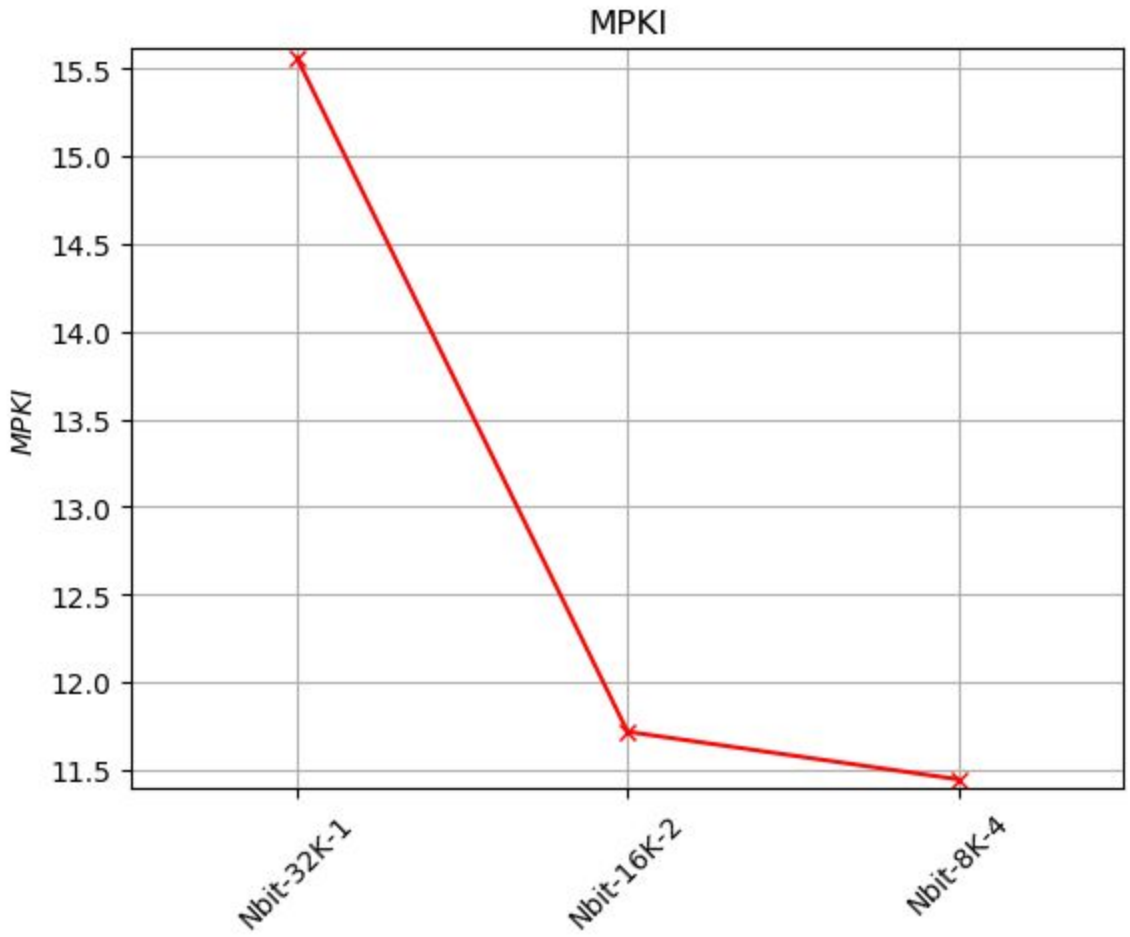
➢ 436.cactusADM


MPKI
0.12
0.10
0.08
0.06
0.04
0.02
0.00
MPKI
Nbit-32K-1
Nbit-16K-2
Nbit-8K-4

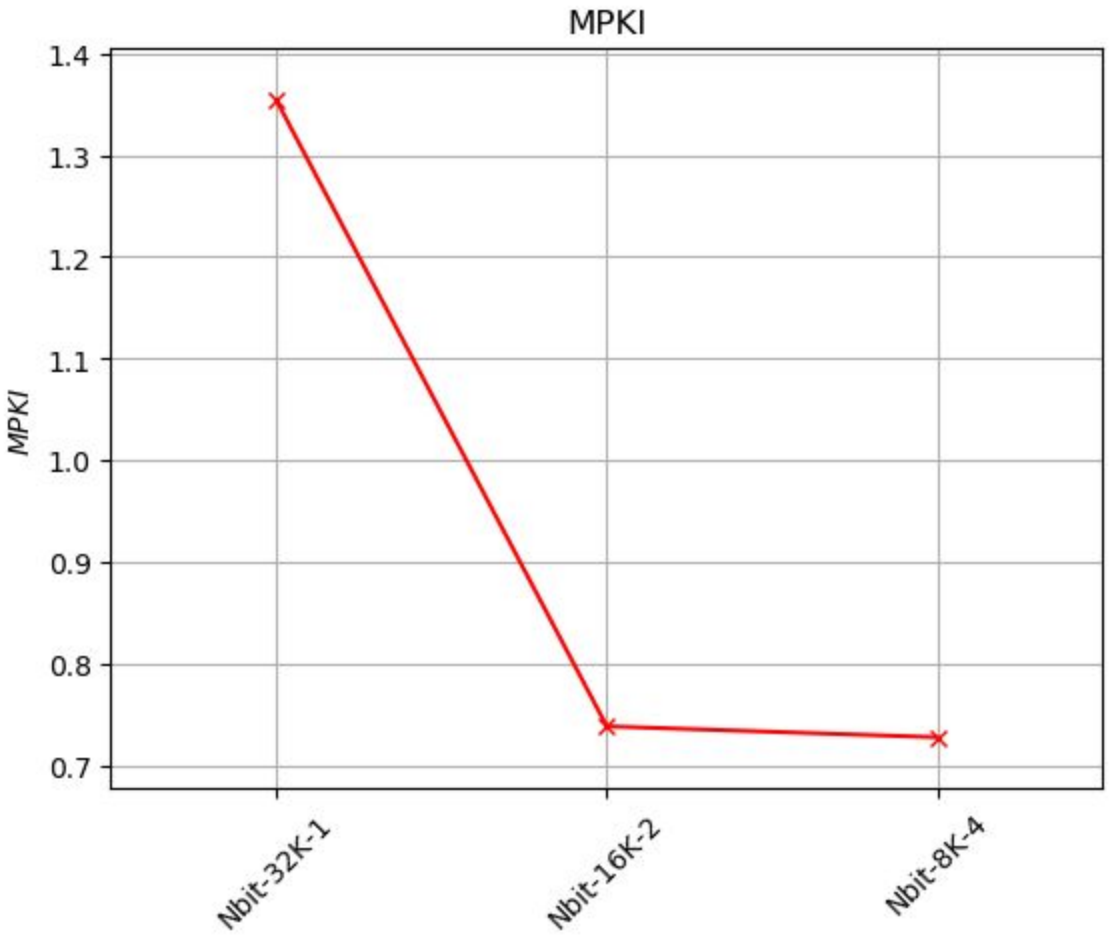
➢ 445.gobmk



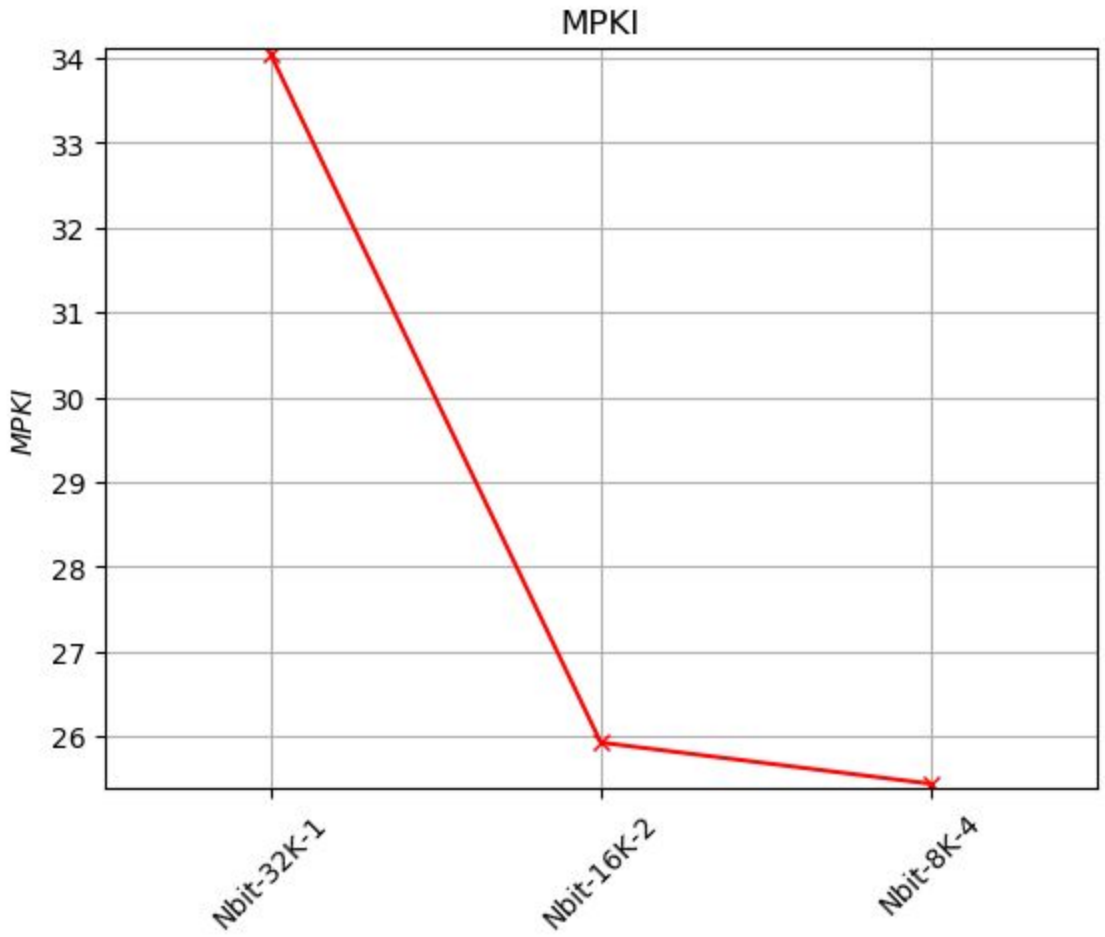
➢ 450.soplex


MPKI
MPKI
15.5
15.0
14.5
14.0
13.5
13.0
12.5
12.0
11.5
Nbit-32K-1
Nbit-16K-2
Nbit-8K-4

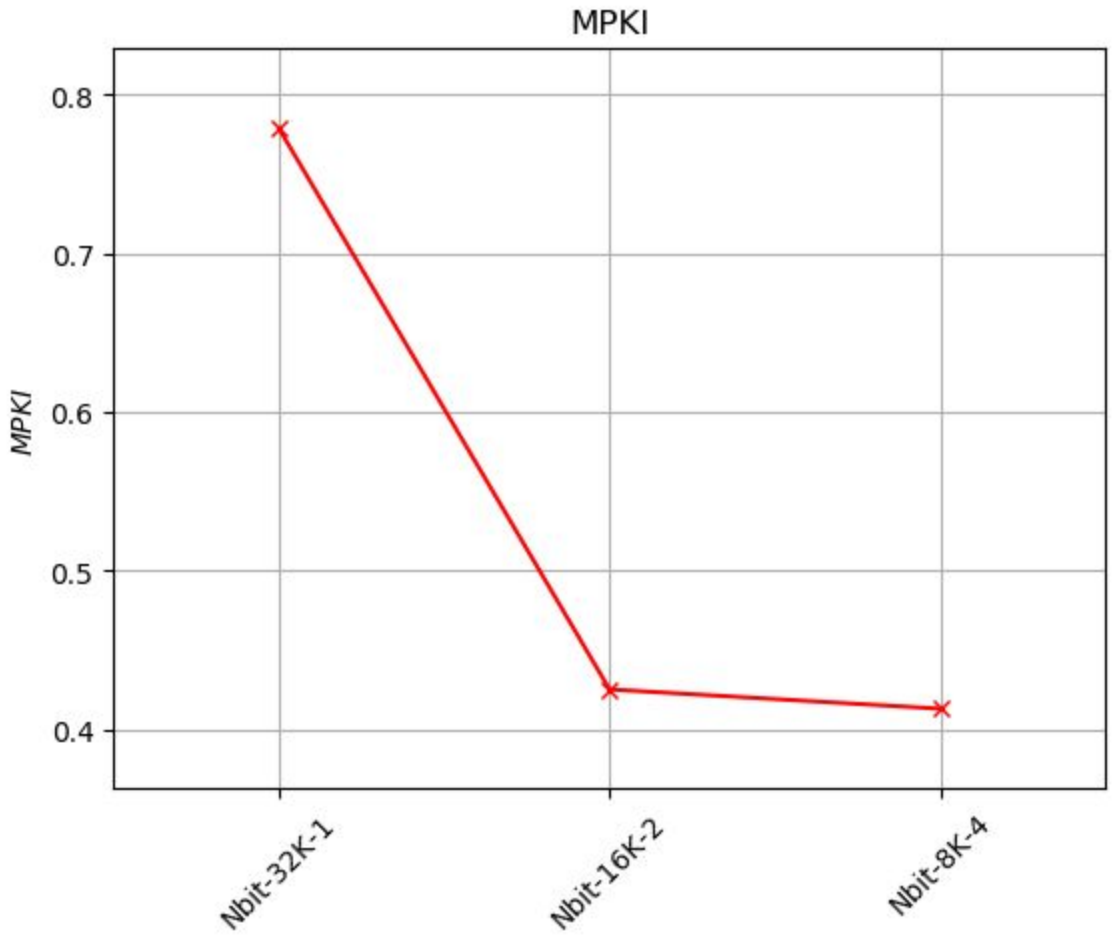
➢ 456.hmmer



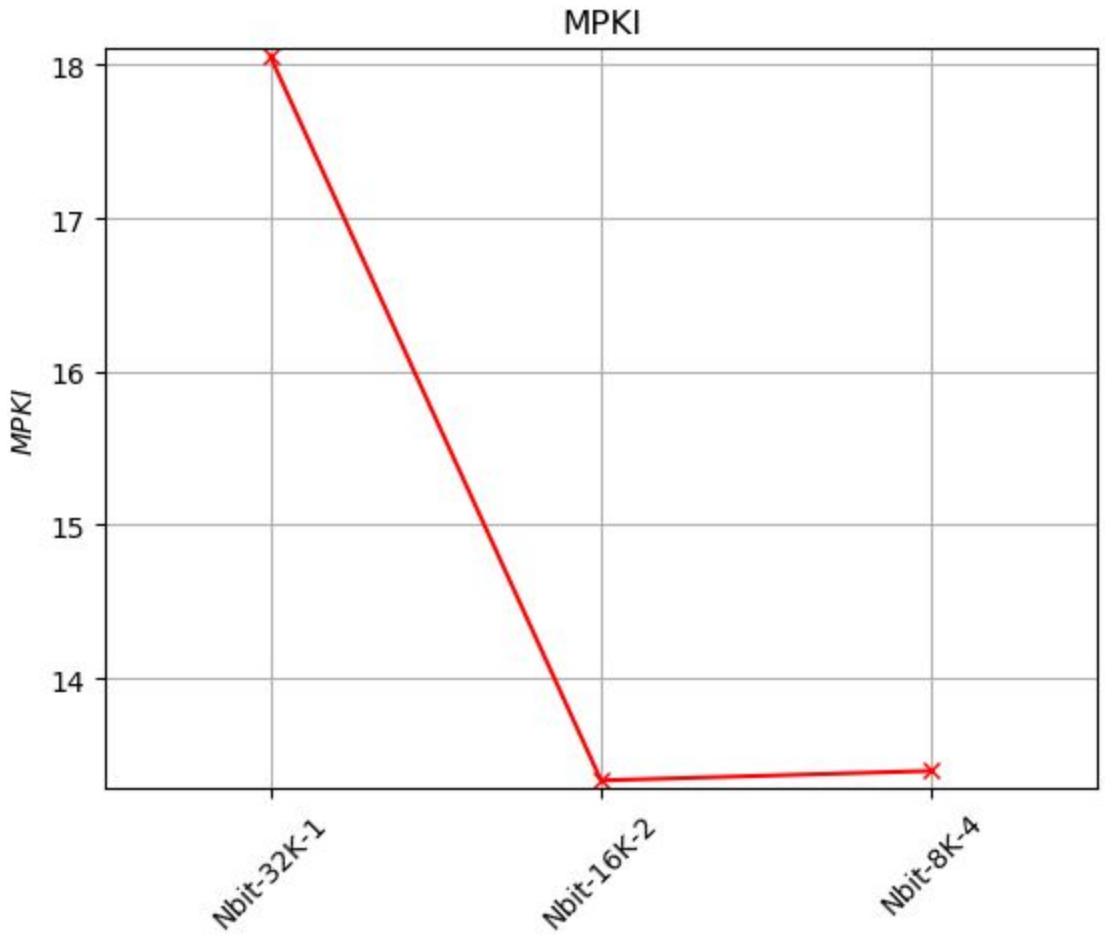
➢ 458.sjeng



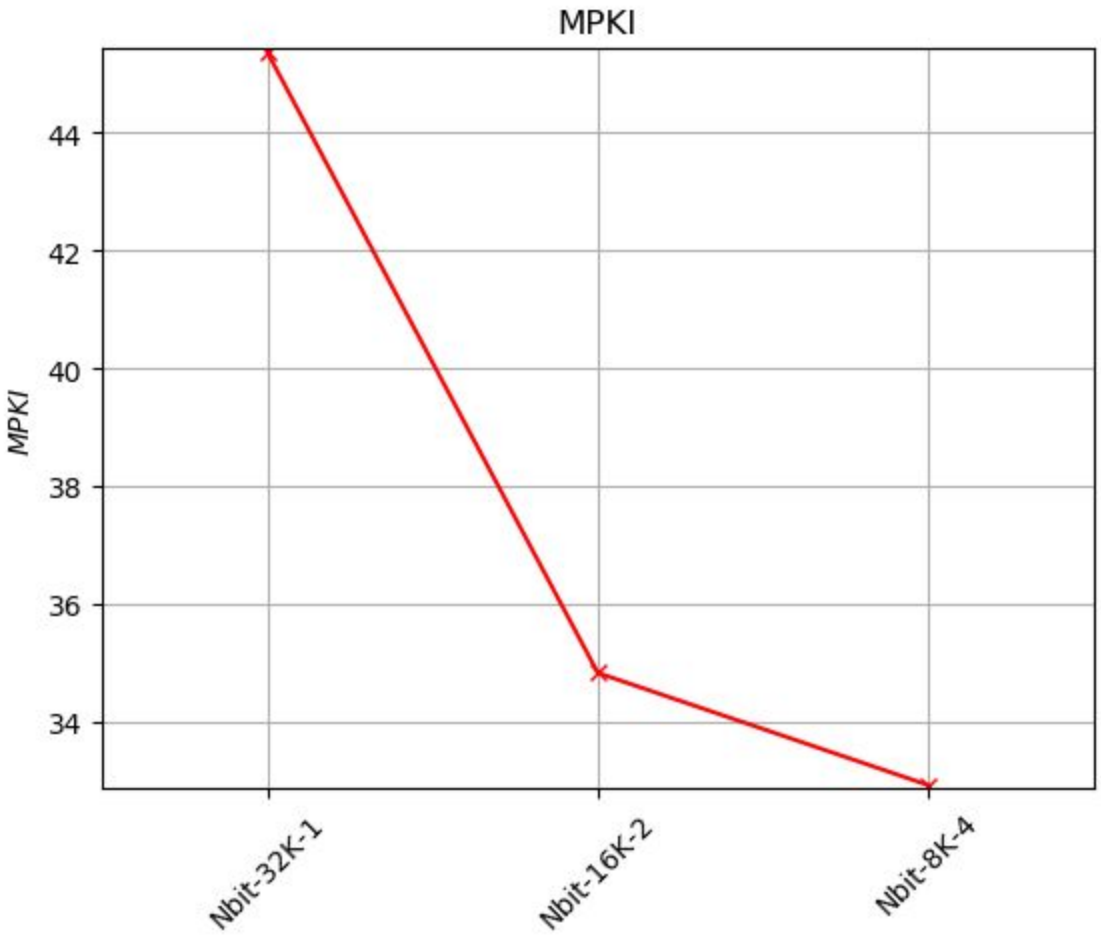
➢ 459.GemsFDTD



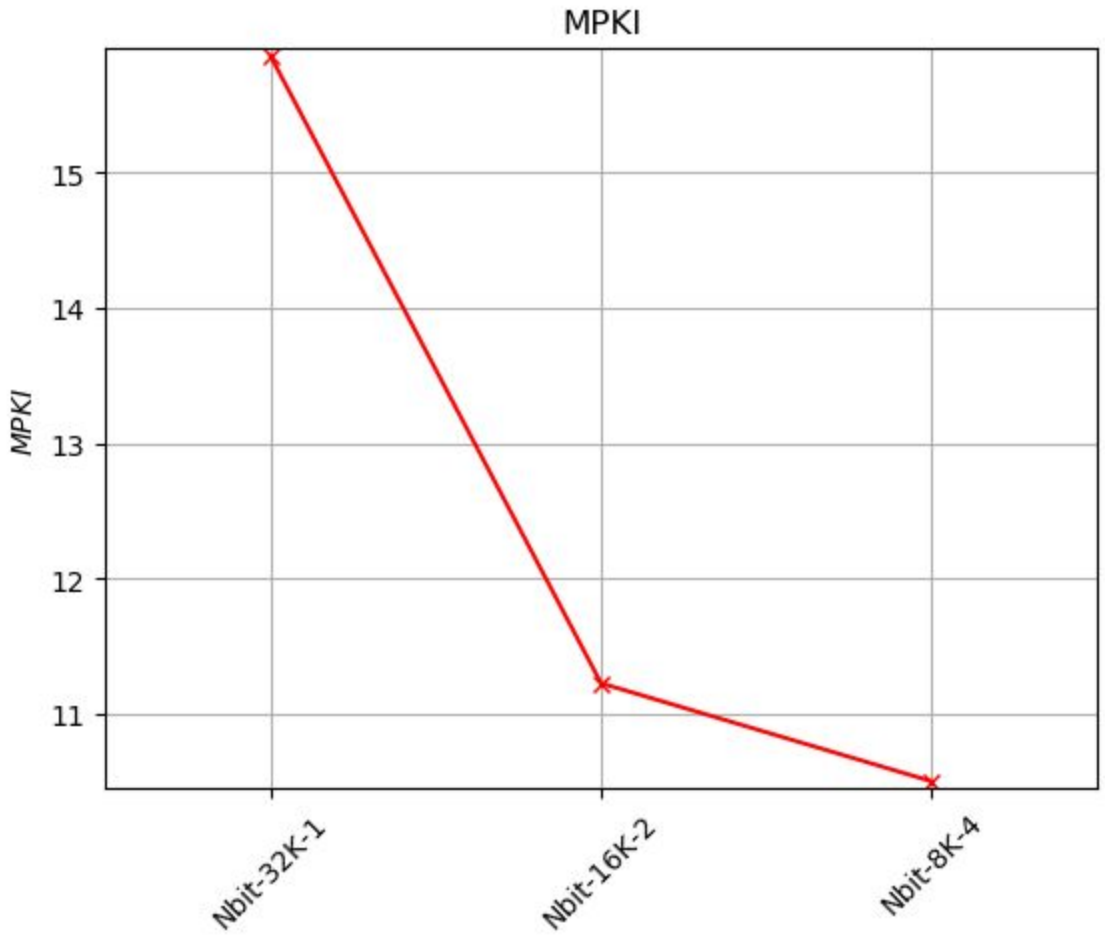
➢ 471.omnetpp



➢ 473.astar



➢ 483.xalancbmk



Συμπεράσματα

Για BHT entries ίσο με 32K και N=1, έχουμε χείριστη απόδοση για κάθε μετροπρόγραμμα. Για 16K και N=2 παρατηρείται μέγιστη βελτίωση της απόδοσης και έχουμε πολύ καλά αποτελέσματα. Για 8K και N=4 στις περισσότερες περιπτώσεις υπάρχει μια ελαφριά βελτίωση της απόδοσης, με εξαίρεση το onetpp που υπάρχει μια ελαφριά χειρότερη και το zeusmp που παρατηρείται έντονη μείωση της απόδοσης,

Ως βέλτιστη επιλογή θα διαλέγαμε τον predictor με 16K BHT entries και N=2 που κατά μέσο όρο για όλα τα μετροπρογράμματα έχει καλύτερη απόδοση.

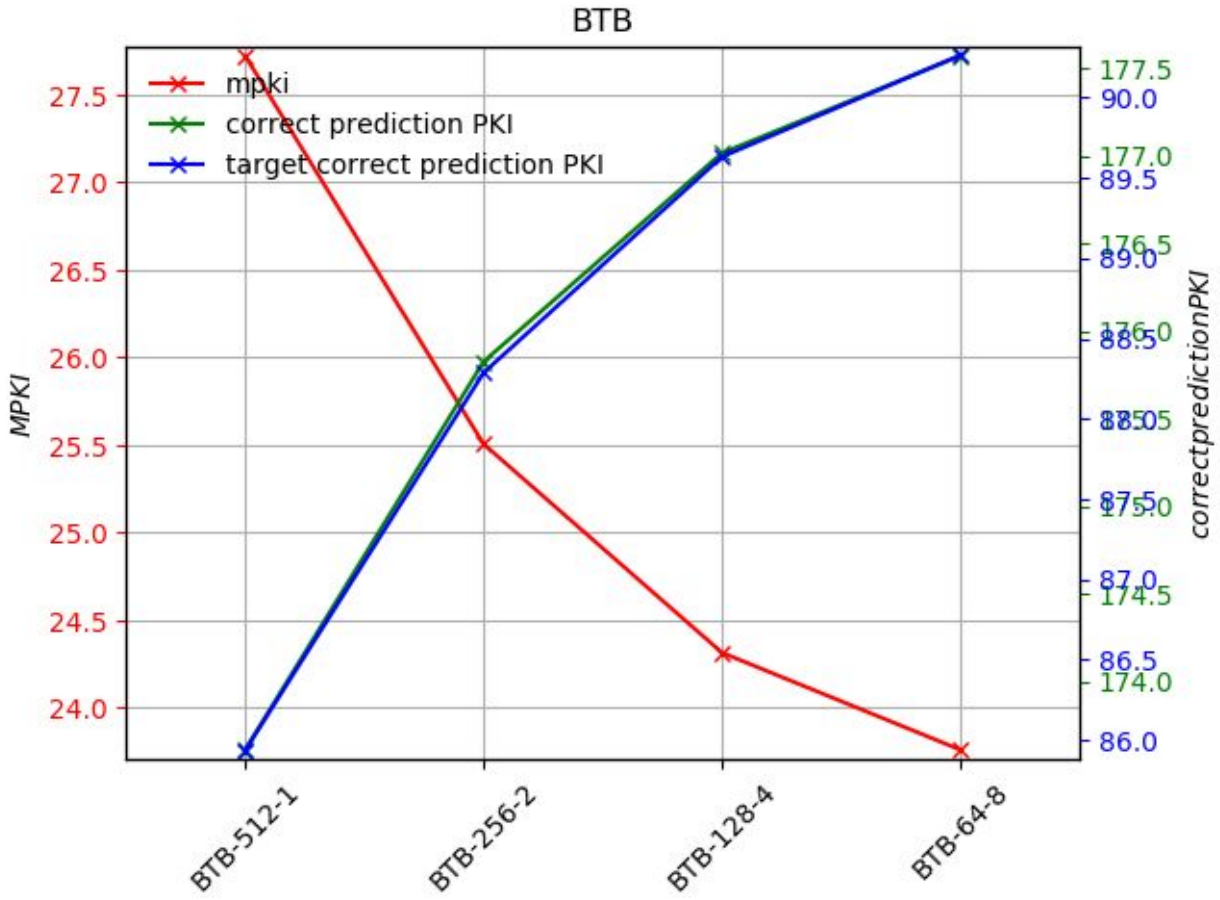
4.3 Μελέτη του BTB

Για αυτό το ερώτημα υλοποιήθηκε ο BTB και μελετήθηκε η ακρίβεια πρόβλεψης του για τις παρακάτω περιπτώσεις:

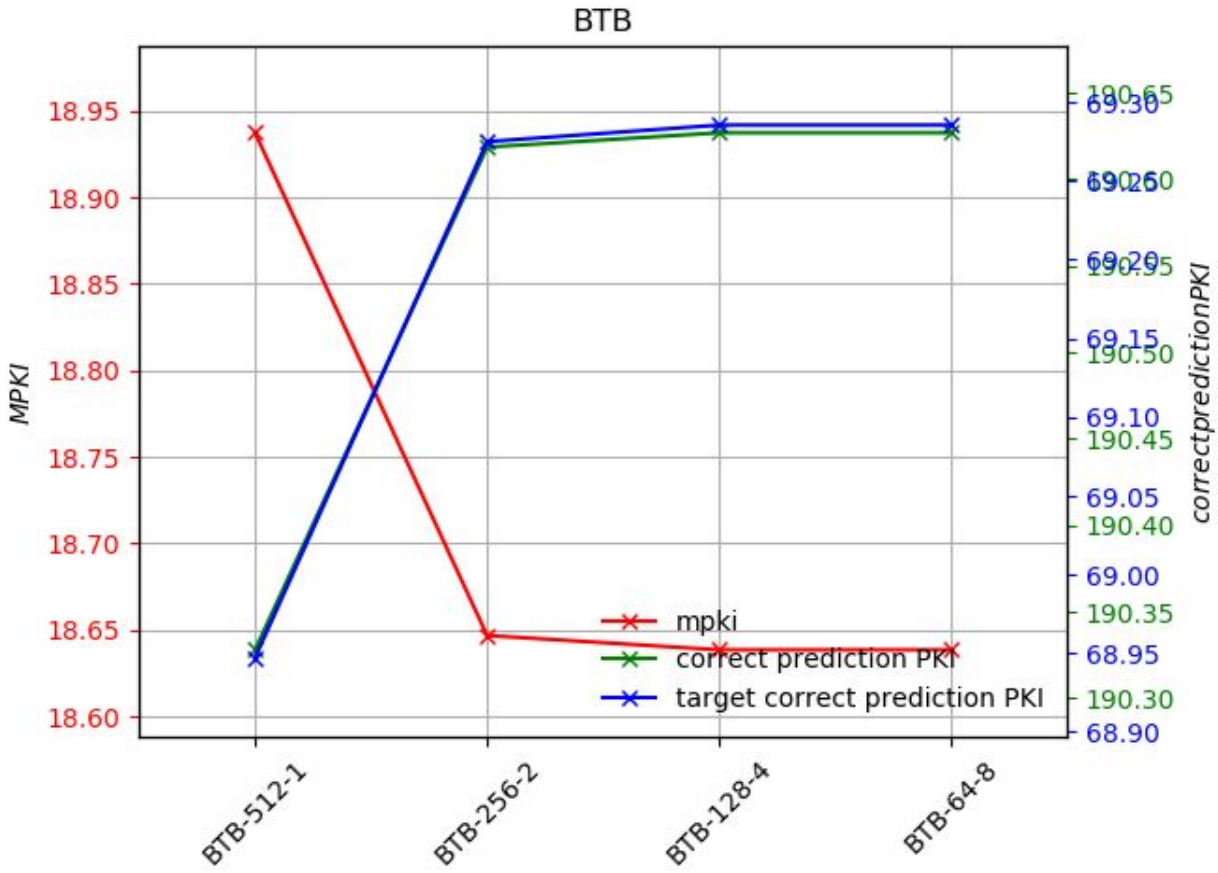
btb entries	btb associativity
512	1
256	2
128	4
64	8

Συγκεκριμένα, δείχνουμε το mpki, δηλαδή το direction misprediction, και αντίστοιχα δείχνουμε το correct prediction αλλά και το target correct prediction, μιας και στους BTB predictors έχουμε 2 περιπτώσεις misses στην περίπτωση direction hit.

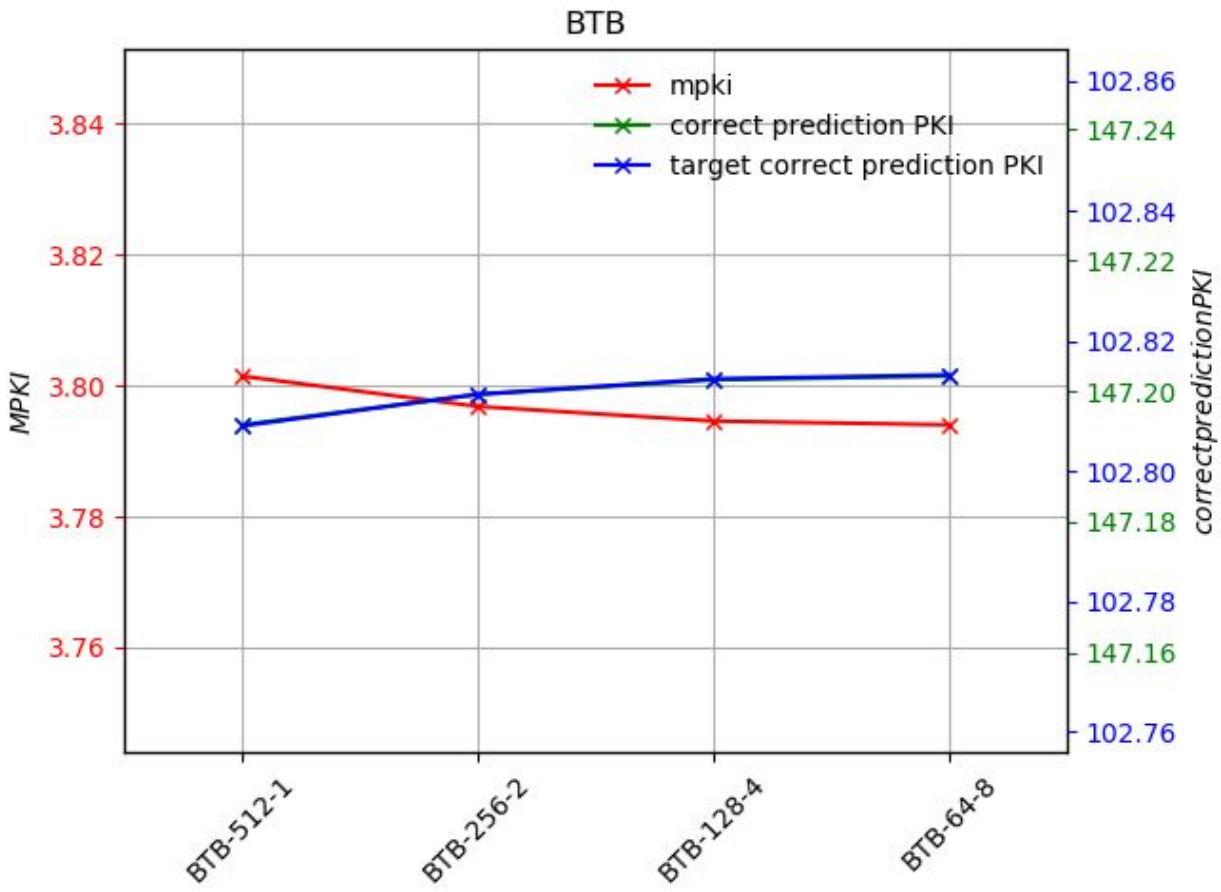
- 403.gcc



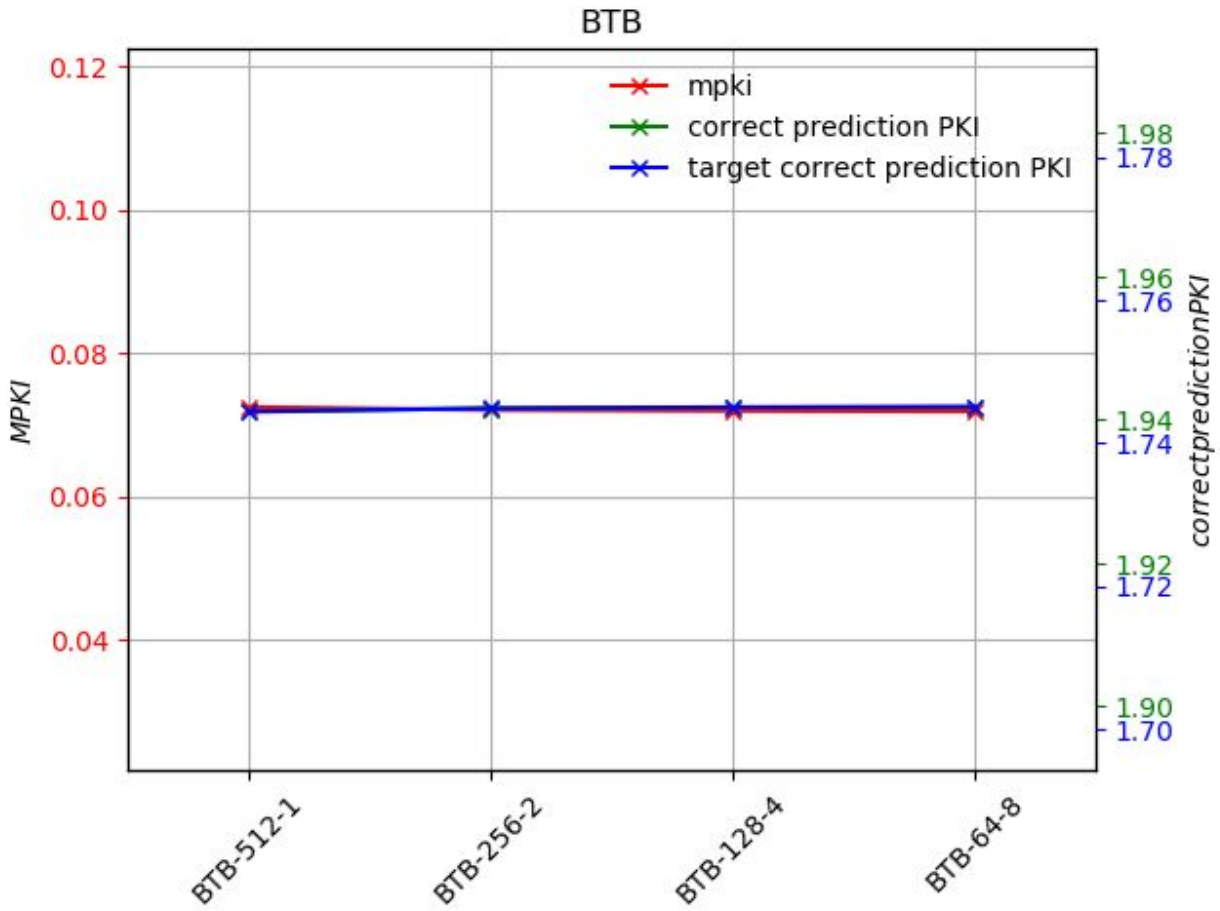
- 429.mcf


BTB
MPKI
correctpredictionPKI
18.95
18.90
18.85
18.80
18.75
18.70
18.65
18.60
190.65
69.30
190.60
69.25
69.20
190.55
69.15
190.50
69.10
190.45
69.05
190.40
69.00
190.35
68.95
190.30
68.90
mpki
correct prediction PKI
target correct prediction PKI
BTB-512-1
BTB-256-2
BTB-128-4
BTB-64-8

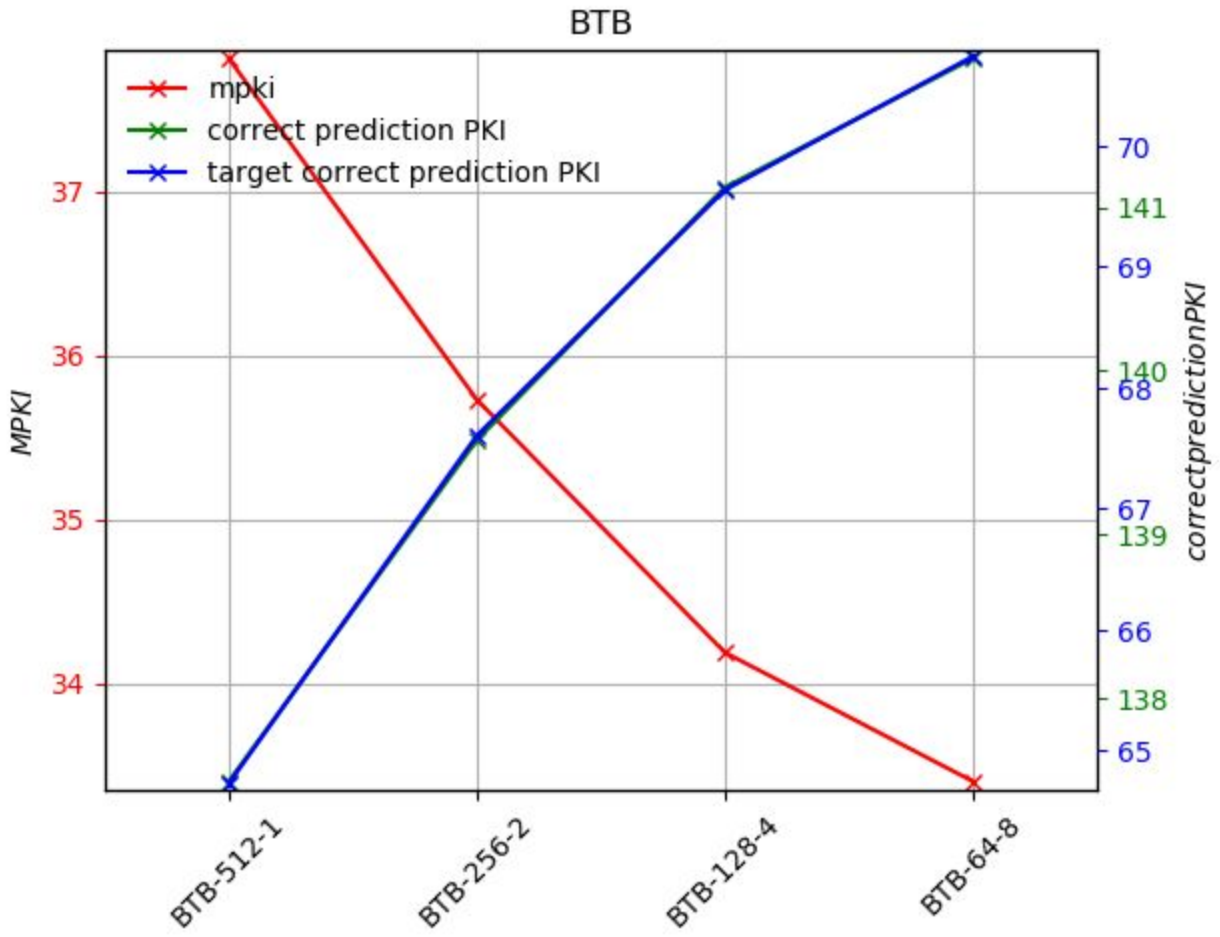
➢ 434.zeusmp



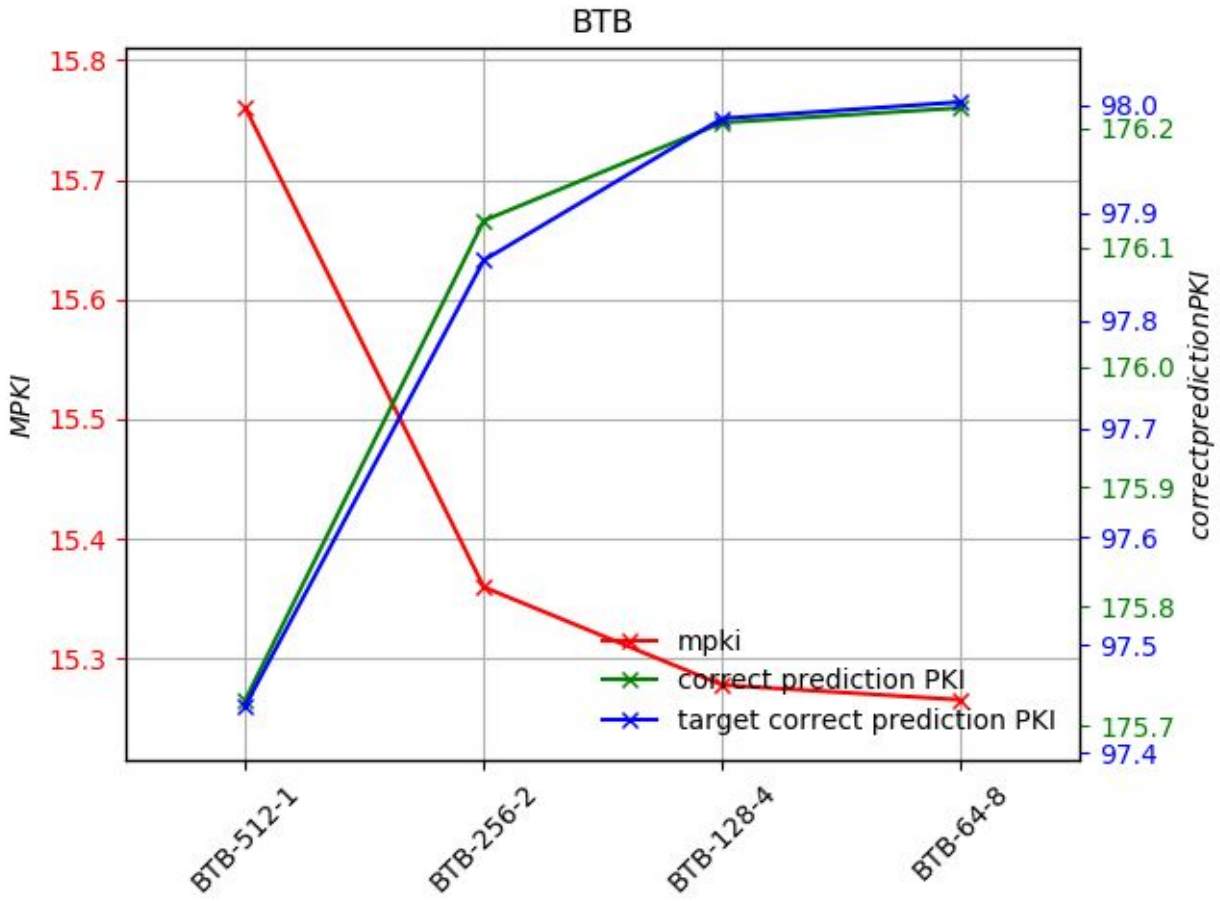
➢ 436.cactusADM


BTB
mpki
correct prediction PKI
target correct prediction PKI
MPKI
correctpredictionPKI
0.12
0.10
0.08
0.06
0.04
1.98
1.78
1.96
1.76
1.94
1.74
1.92
1.72
1.90
1.70
BTB-512-1
BTB-256-2
BTB-128-4
BTB-64-8

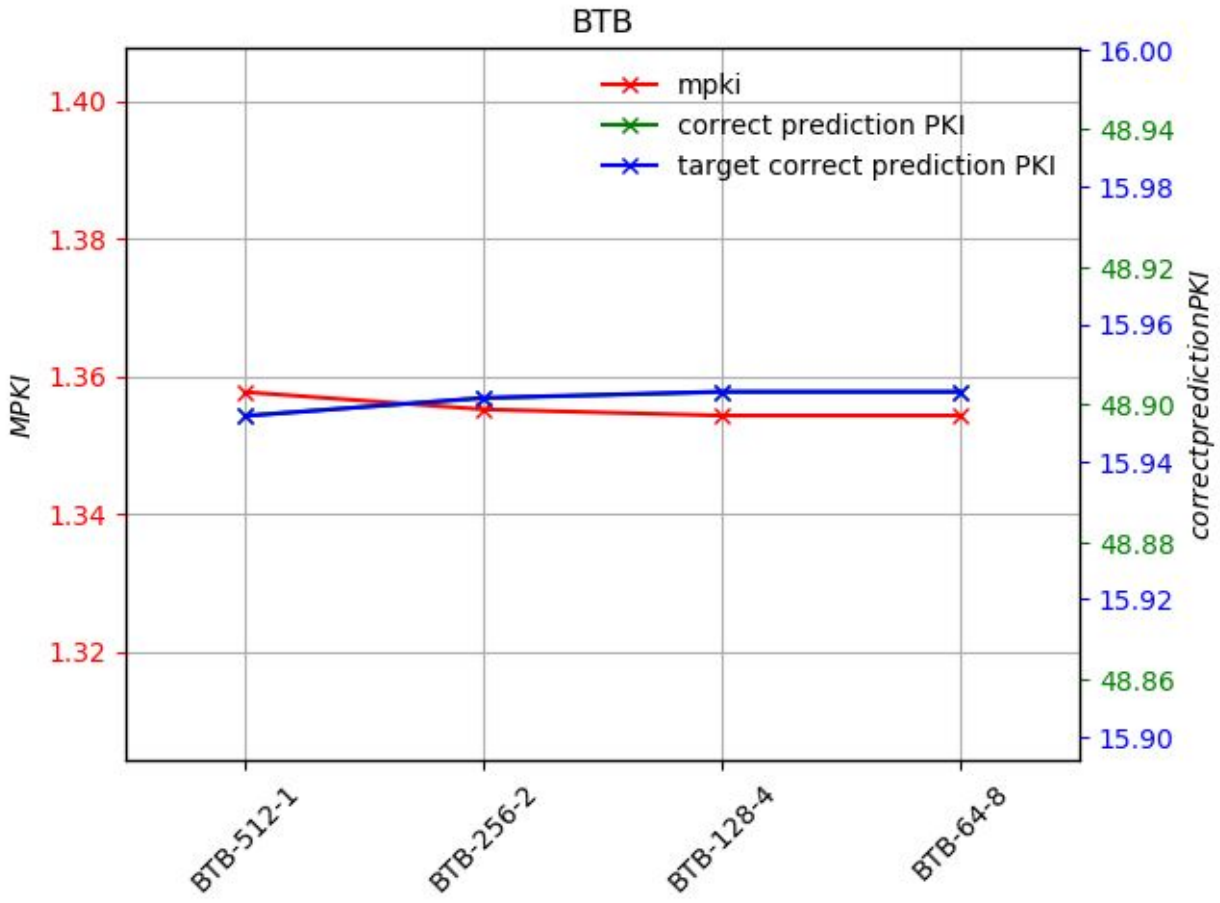
➢ 445.gobmk


BTB
mpki
correct prediction PKI
target correct prediction PKI
MPKI
correctpredictionPKI
37
36
35
34
70
141
69
140
68
67
139
66
138
65
BTB-512-1
BTB-256-2
BTB-128-4
BTB-64-8

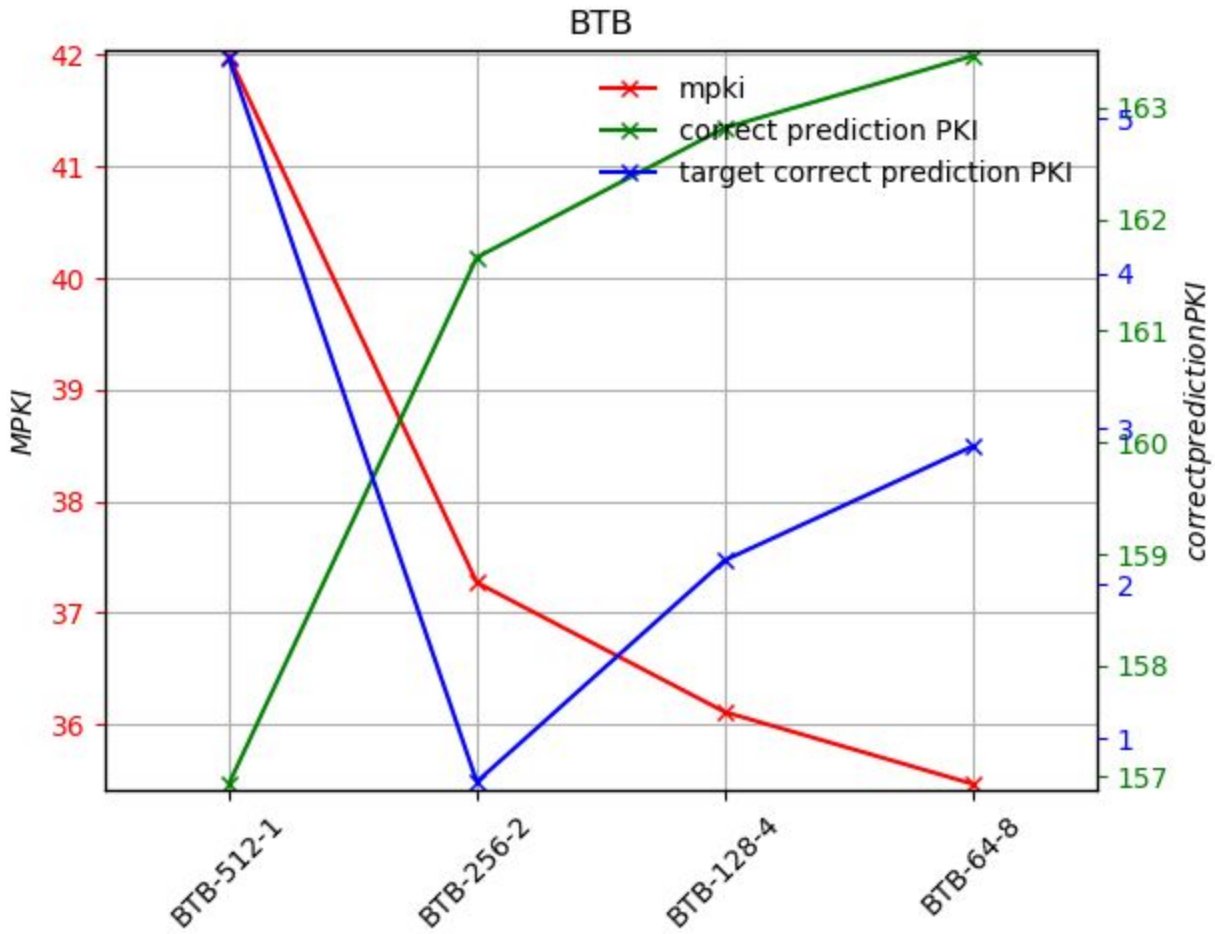
➢ 450.soplex


BTB
15.8
15.7
15.6
15.5
15.4
15.3
MPKI
98.0
176.2
97.9
176.1
97.8
176.0
97.7
175.9
97.6
175.8
97.5
175.7
97.4
correctpredictionPKI
mpki
correct prediction PKI
target correct prediction PKI
BTB-512-1
BTB-256-2
BTB-128-4
BTB-64-8

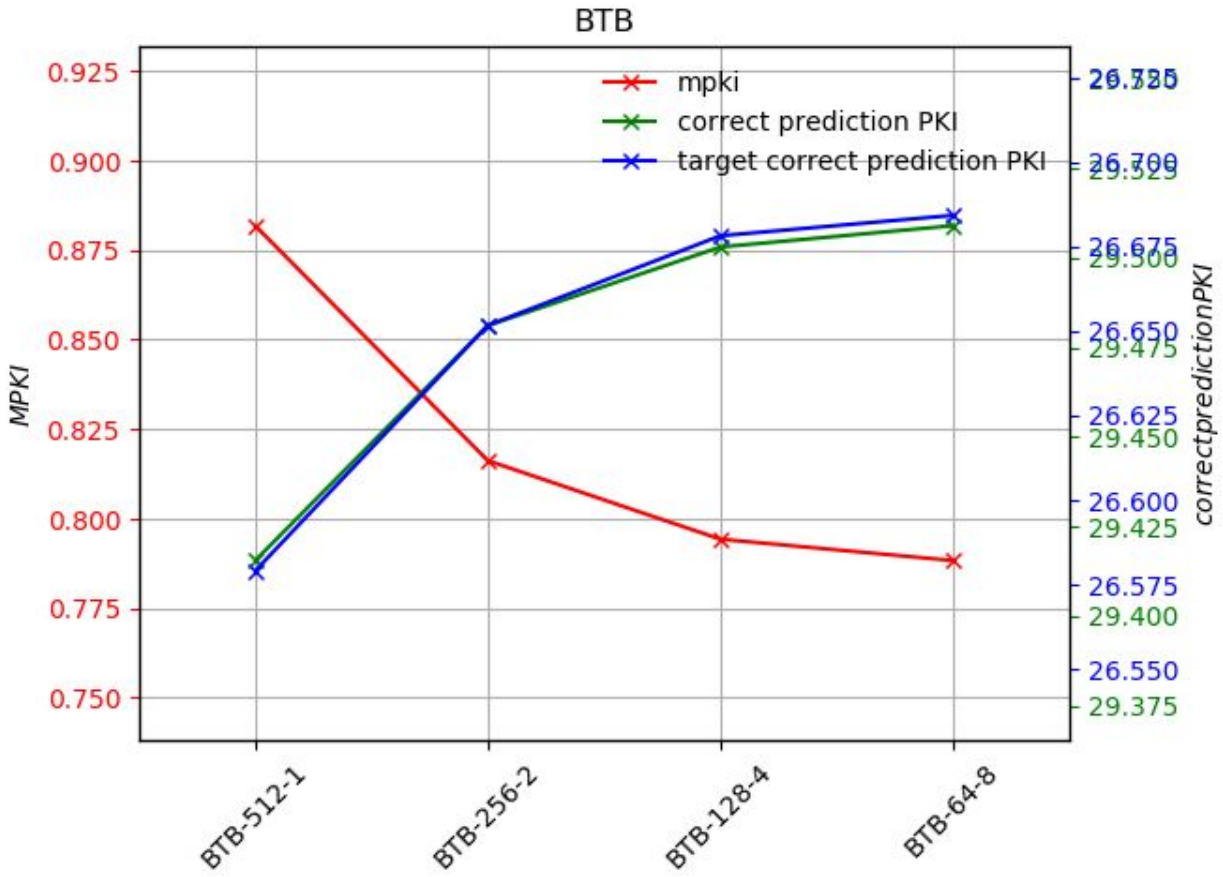
➢ 456.hmmer


BTB
mpki
correct prediction PKI
target correct prediction PKI
MPKI
correctpredictionPKI
1.40
1.38
1.36
1.34
1.32
16.00
48.94
15.98
48.92
15.96
48.90
15.94
48.88
15.92
48.86
15.90
BTB-512-1
BTB-256-2
BTB-128-4
BTB-64-8

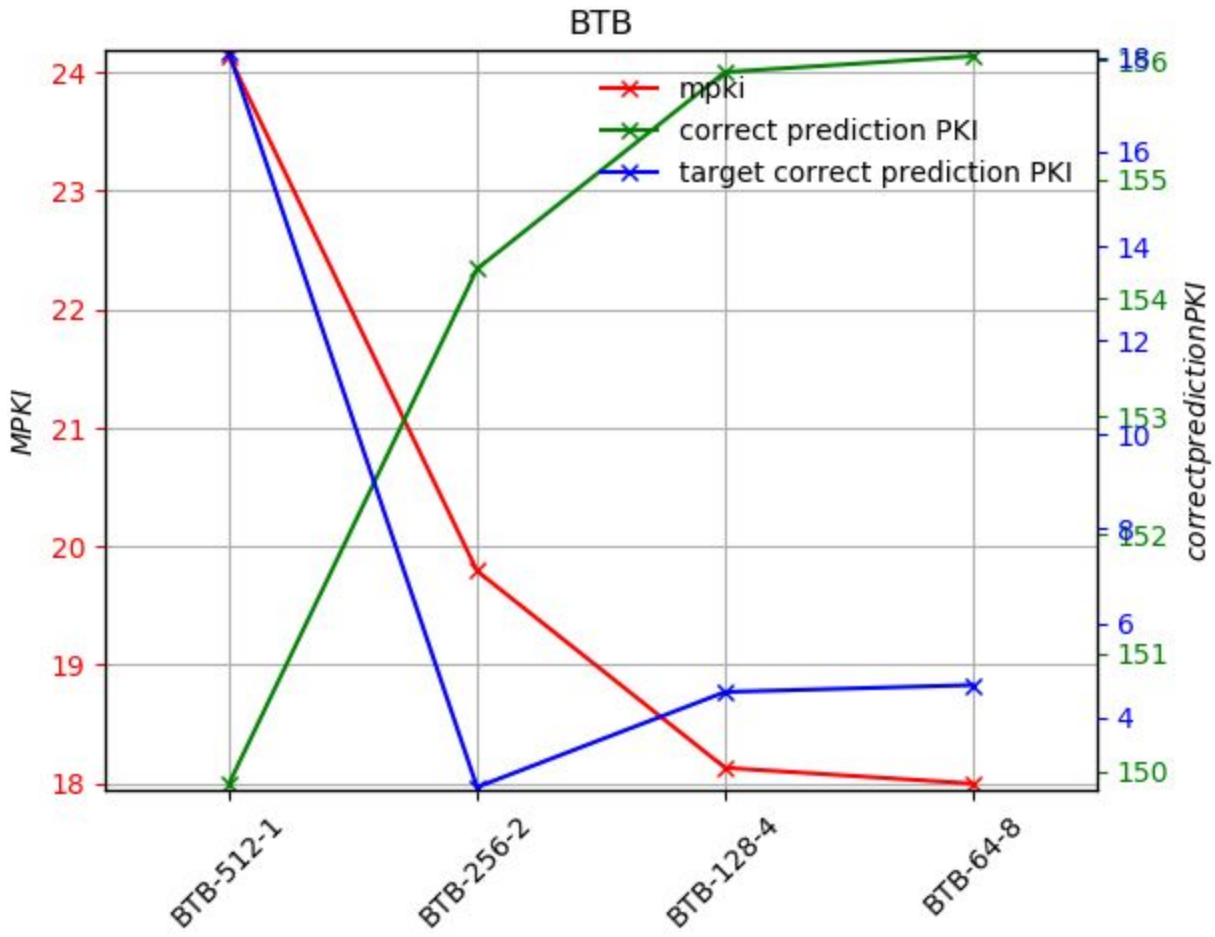
➢ 458.sjeng



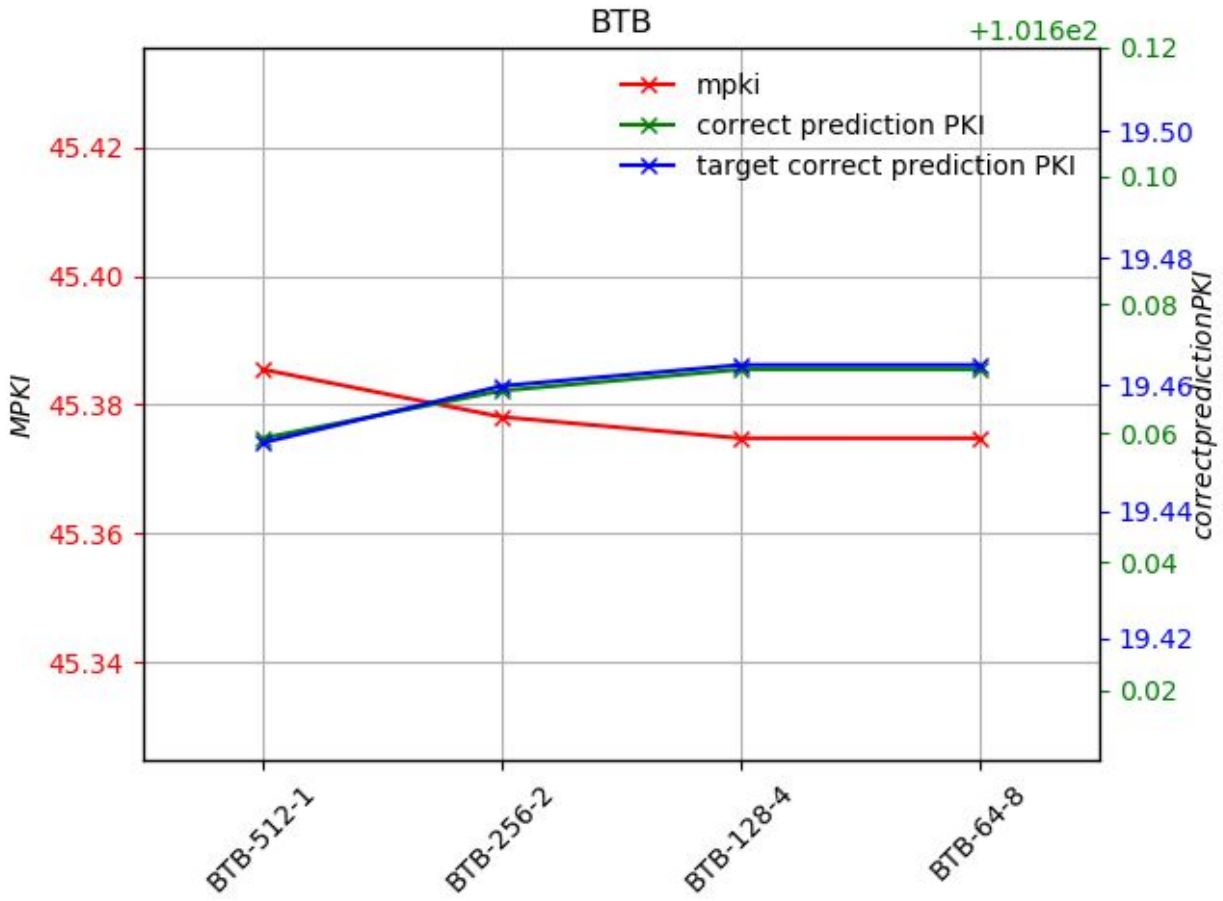
➢ 459.GemsFDTD


BTB
mpki
correct prediction PKI
target correct prediction PKI
MPKI
correctpredictionPKI
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0.750
26.700
26.675
26.650
26.625
26.600
26.575
26.550
29.525
29.500
29.475
29.450
29.425
29.400
29.375
BTB-512-1
BTB-256-2
BTB-128-4
BTB-64-8

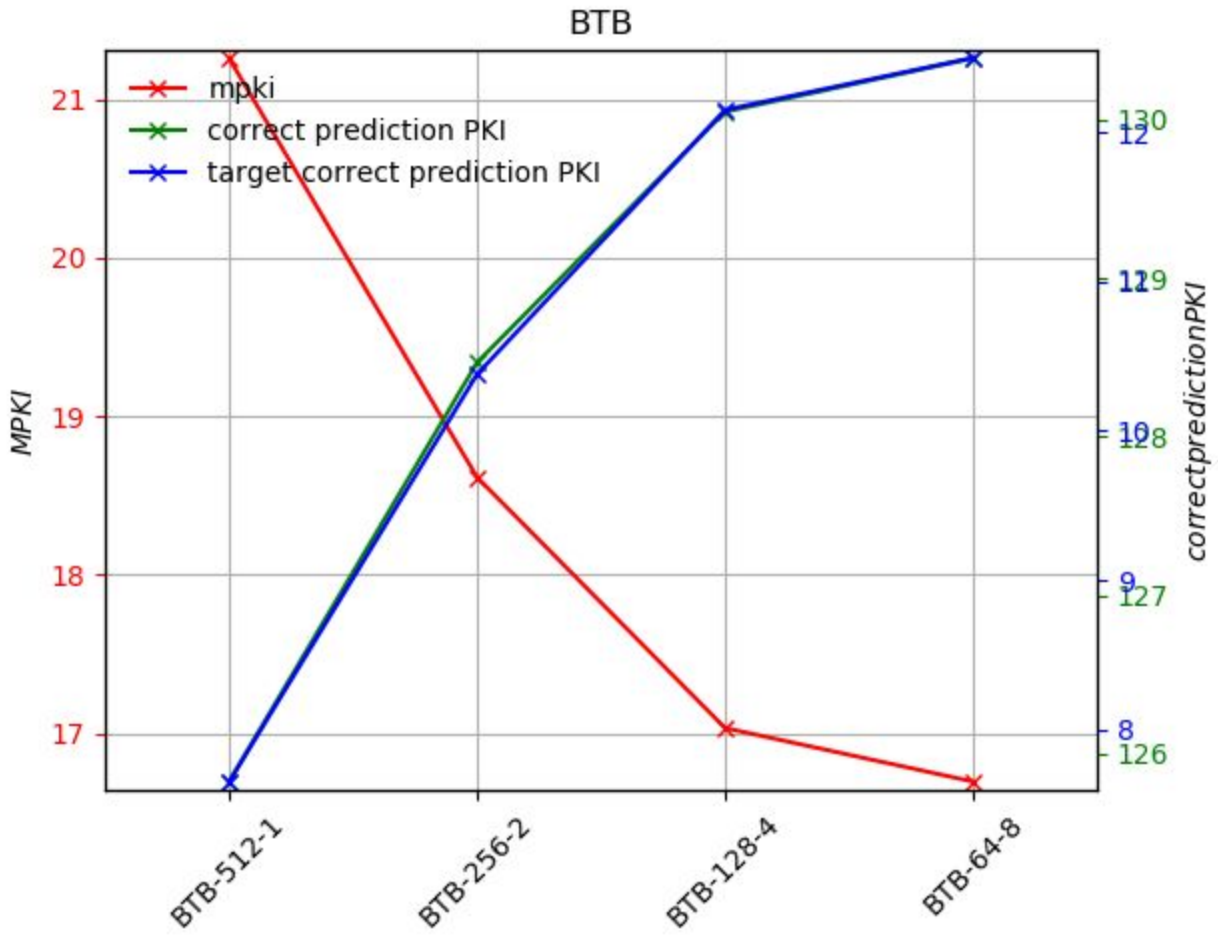
➢ 471.omnetpp


BTB
mpki
correct prediction PKI
target correct prediction PKI
MPKI
correctpredictionPKI
24
23
22
21
20
19
18
16
155
14
154
12
153
10
152
6
151
4
150
BTB-512-1
BTB-256-2
BTB-128-4
BTB-64-8

➢ 473.astar



- 483.xalancbmk



Συμπεράσματα

Αν και σαν μέθοδος παρουσιάζει καλές αποδόσεις, υπάρχουν συχνά σφάλματα λανθασμένου στόχου, αφού ο πίνακας δεν έχει αποθηκευμένο τον στόχο της διακλάδωσης ακόμη και στην περίπτωση που γίνει σωστή πρόβλεψη.

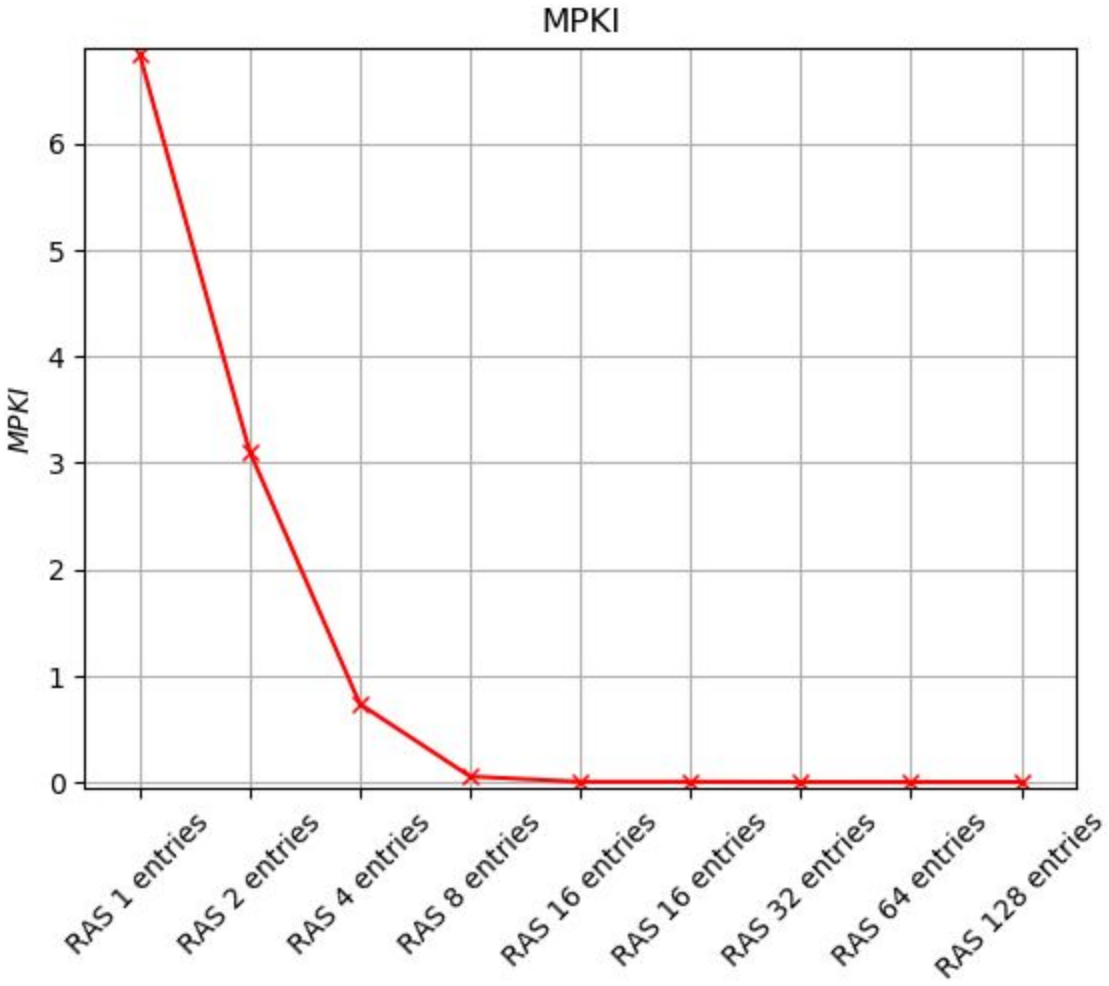
Σε γενικές γραμμές παρατηρούμε πως τα target correct predictions και τα correct predictions ακολουθούν παρόμοιες κατανομές (με εξαίρεση το μετροπρόγραμμα omnetpp) ενώ συνολικά όσο πάμε σε επόμενη περίπτωση (μικρότερο btb entries μεγαλύτερο btb associativity) έχουμε και εκθετικά καλύτερη απόδοση, αφού για να αντικατασταθεί μια εντολή στην ίδια διεύθυνση btb απαιτείται να αποθηκευτούν τόσες νέες εντολές όσο και το associativity.
Βέβαια, βλέπουμε πως μεταβολές δεν είναι και τόσο μεγάλου μεγέθους.

Τελικώς θα επιλέγαμε το BTB-128-4 και ή το BTB-64-8 που παρουσιάζουν μικρές διαφορές στην απόδοση.

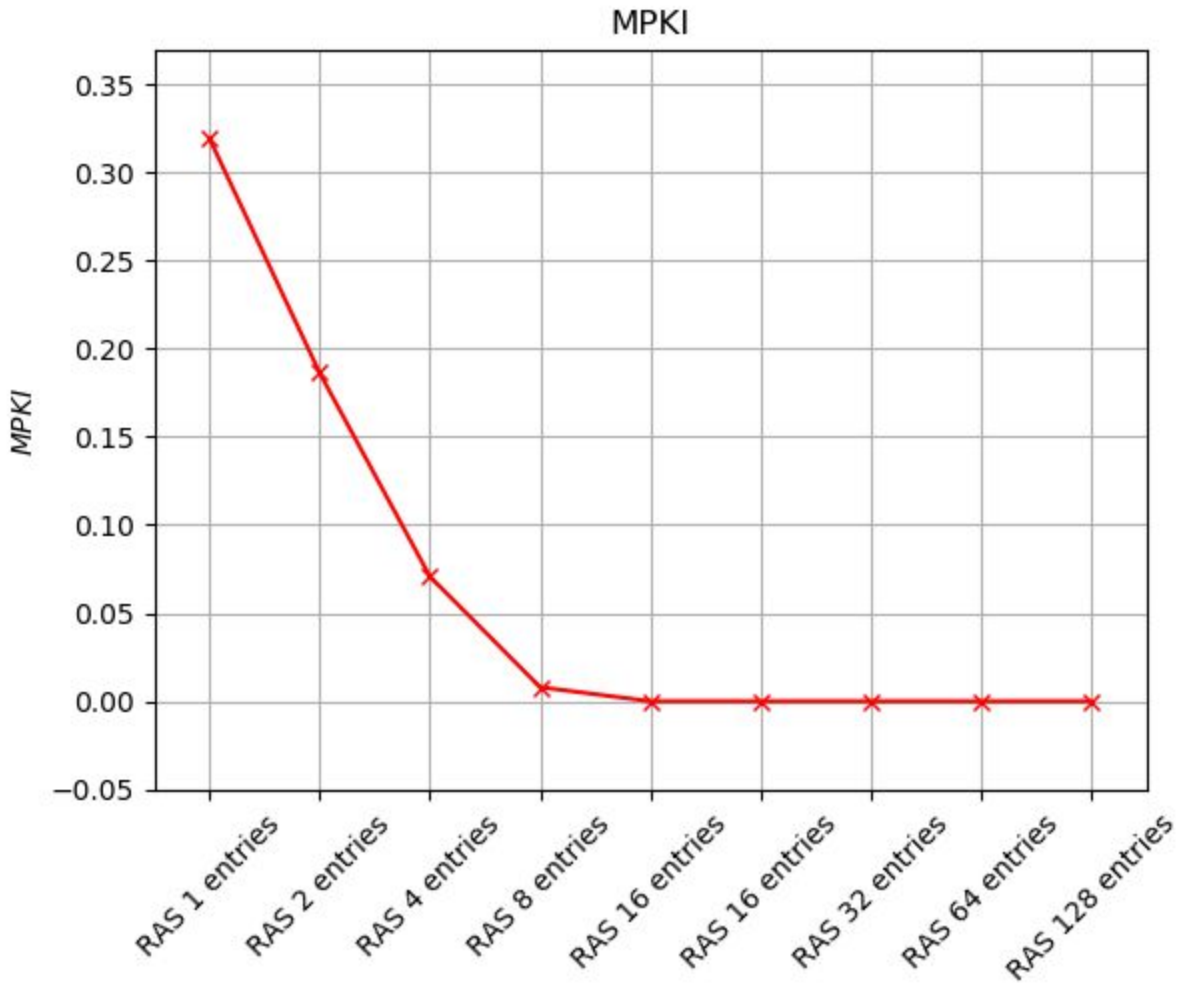
4.4 Μελέτη του RAS

Εδώ χρησιμοποιώντας την υπάρχουσα υλοποίηση της RAS, παρατηρούμε την μεταβολή του ποσοστού αποτυχίας για διαφορετικούς αριθμούς εγγραφών στο RAS (1, 2, 4, 16, 64, 128)

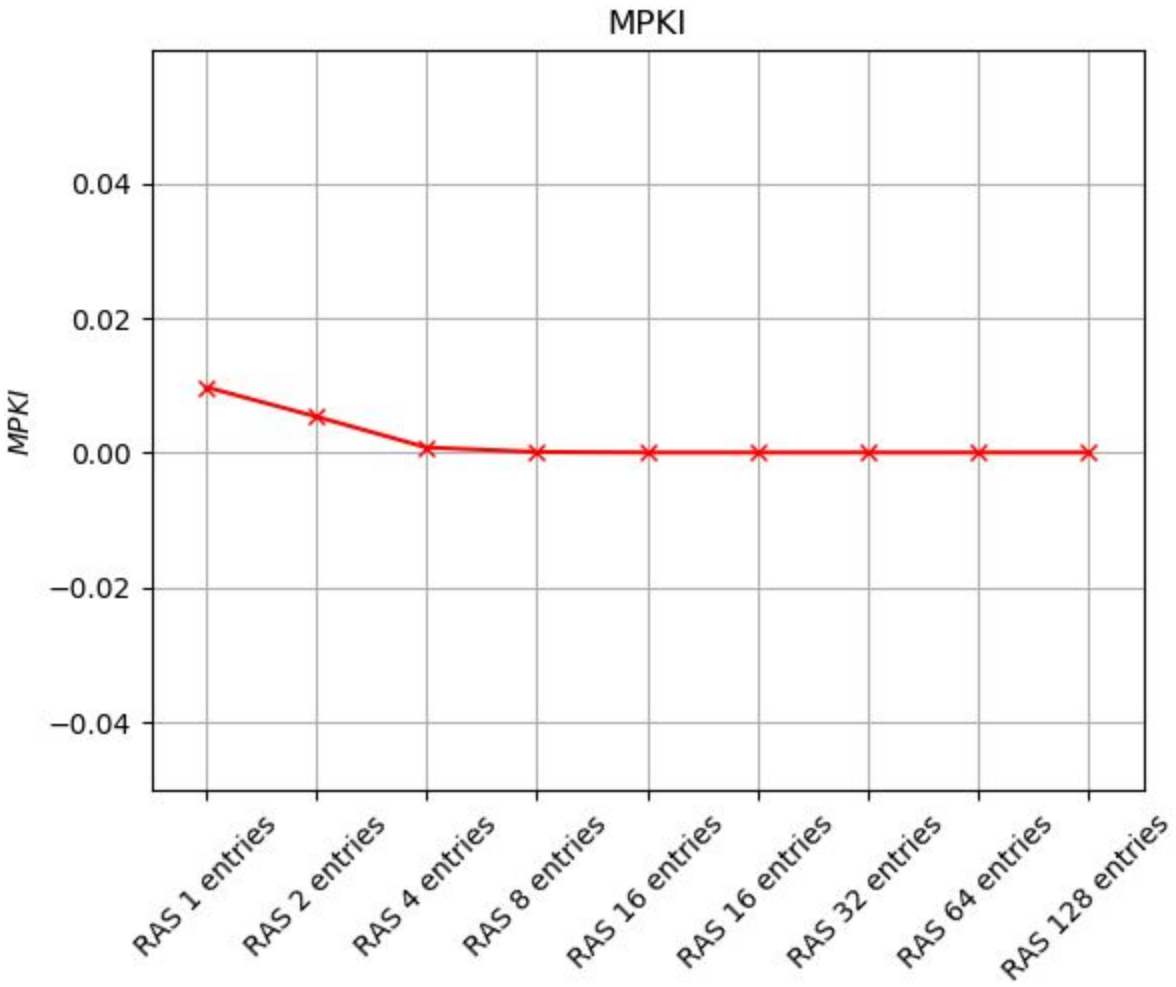
- 403.gcc



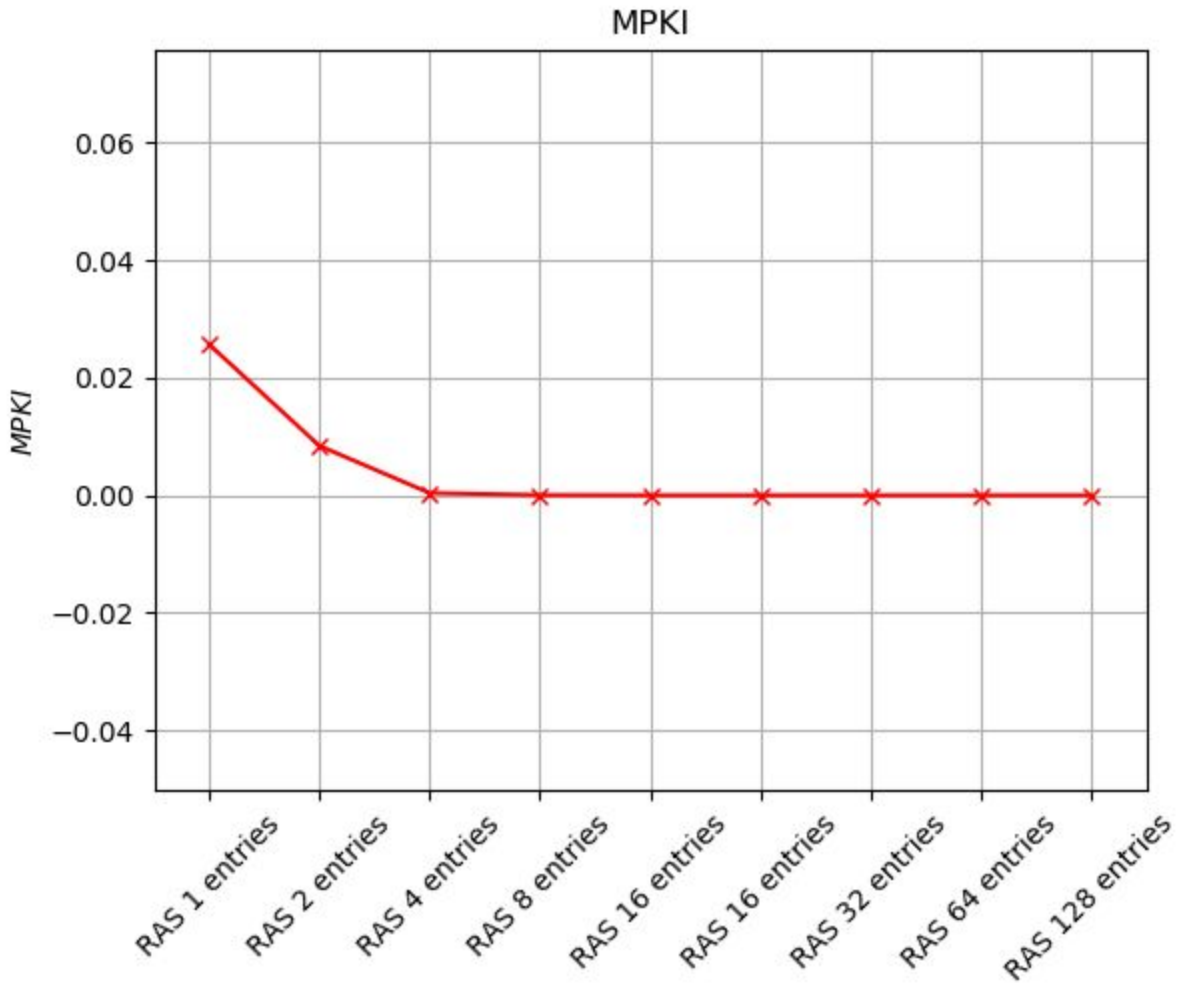
- 429.mcf


MPKI
MPKI
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
RAS 1 entries
RAS 2 entries
RAS 4 entries
RAS 8 entries
RAS 16 entries
RAS 16 entries
RAS 32 entries
RAS 64 entries
RAS 128 entries

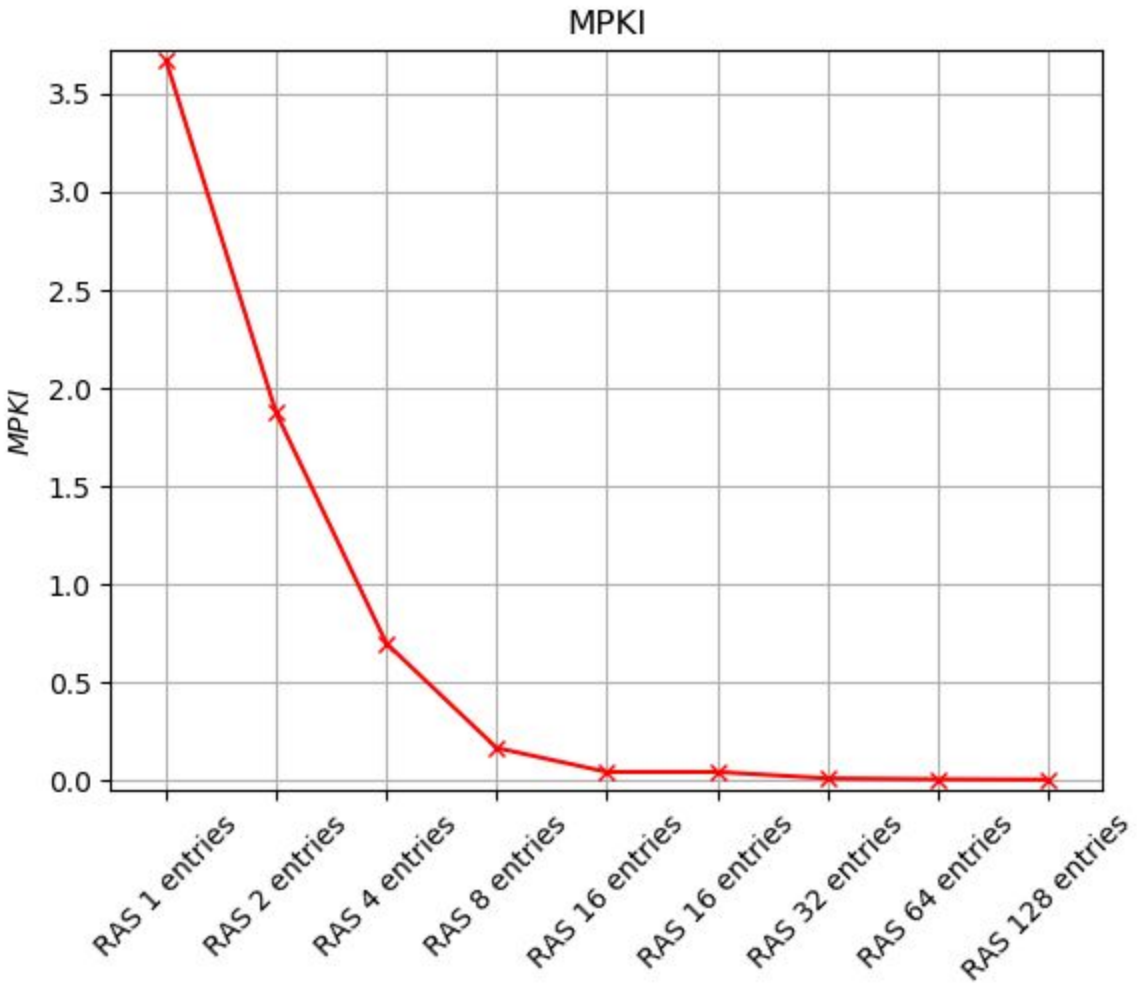
➢ 434.zeusmp



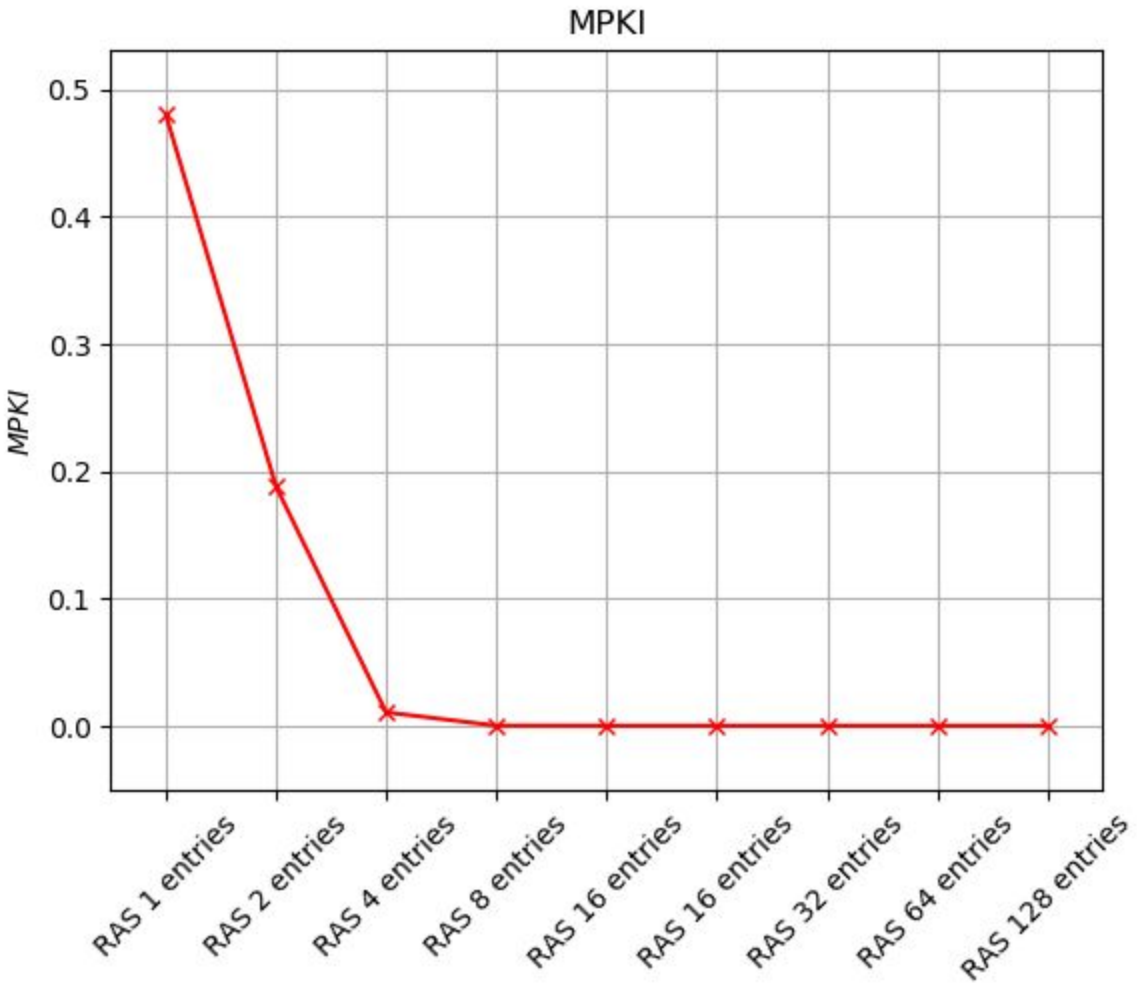
➢ 436.cactusADM


MPKI
MPKI
0.06
0.04
0.02
0.00
−0.02
−0.04
RAS 1 entries
RAS 2 entries
RAS 4 entries
RAS 8 entries
RAS 16 entries
RAS 16 entries
RAS 32 entries
RAS 64 entries
RAS 128 entries

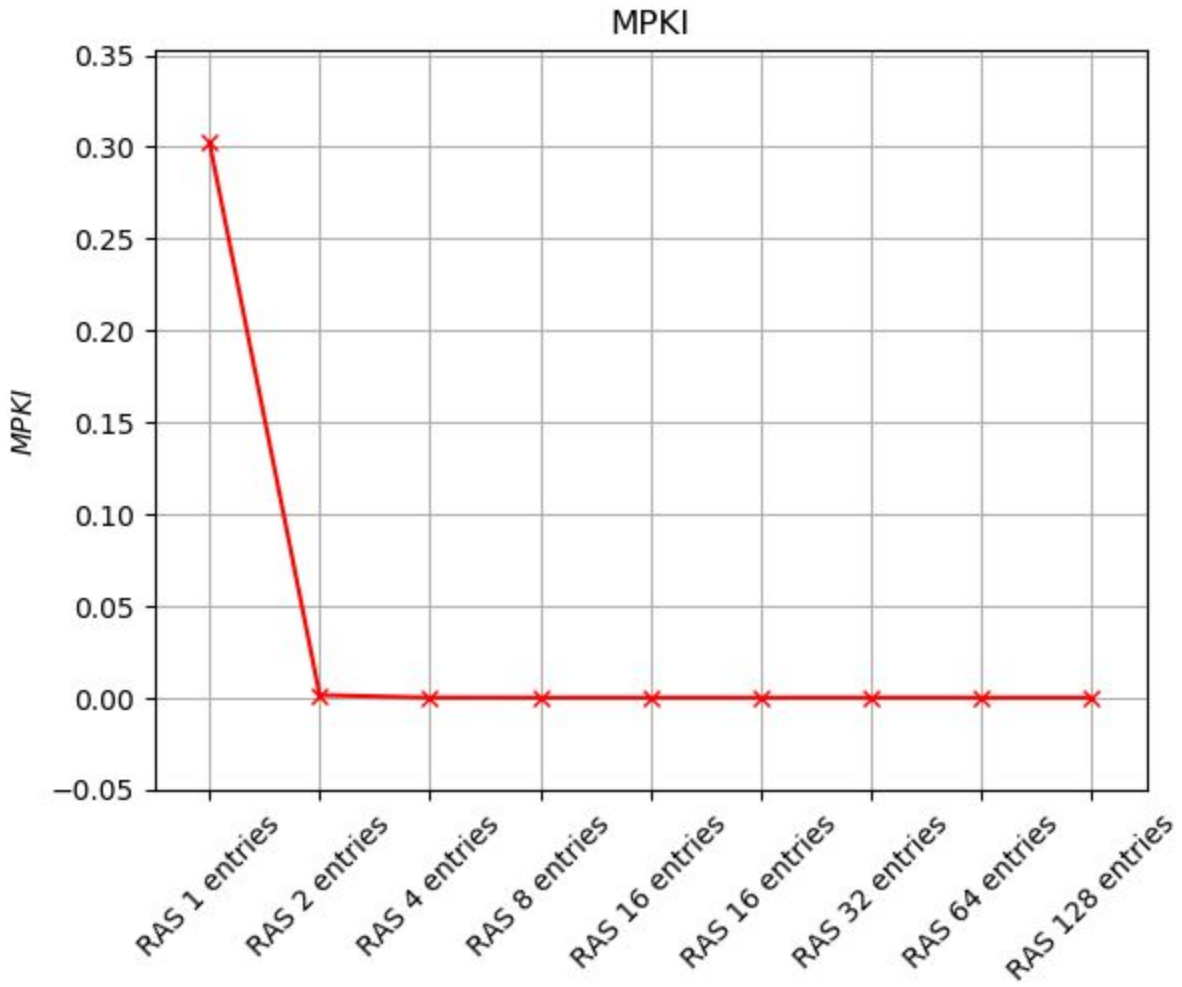
➢ 445.gobmk


MPKI
MPKI
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
RAS 1 entries
RAS 2 entries
RAS 4 entries
RAS 8 entries
RAS 16 entries
RAS 16 entries
RAS 32 entries
RAS 64 entries
RAS 128 entries

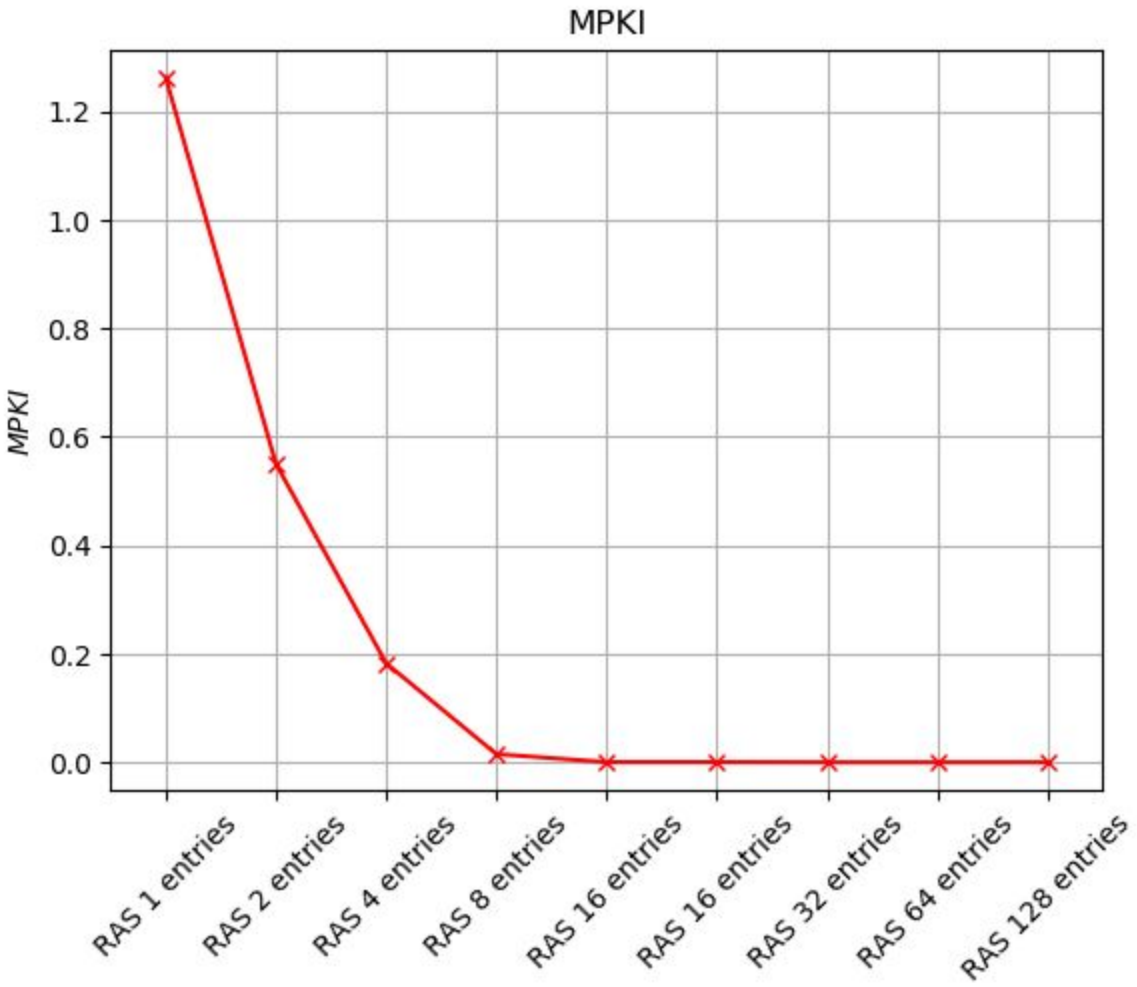
- 450.soplex


MPKI
MPKI
0.5
0.4
0.3
0.2
0.1
0.0
RAS 1 entries
RAS 2 entries
RAS 4 entries
RAS 8 entries
RAS 16 entries
RAS 16 entries
RAS 32 entries
RAS 64 entries
RAS 128 entries

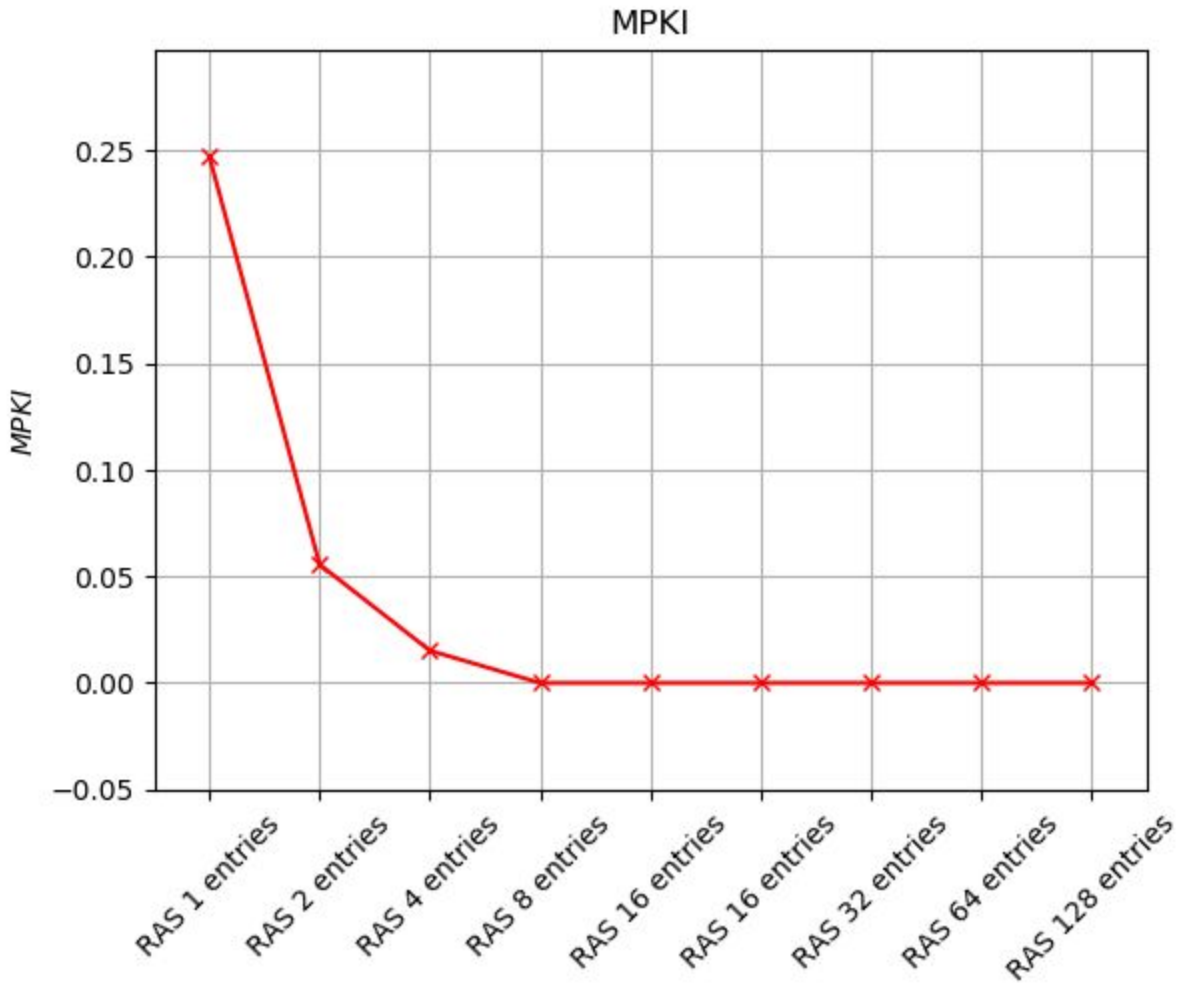
- 456.hmmer


MPKI
MPKI
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
RAS 1 entries
RAS 2 entries
RAS 4 entries
RAS 8 entries
RAS 16 entries
RAS 16 entries
RAS 32 entries
RAS 64 entries
RAS 128 entries

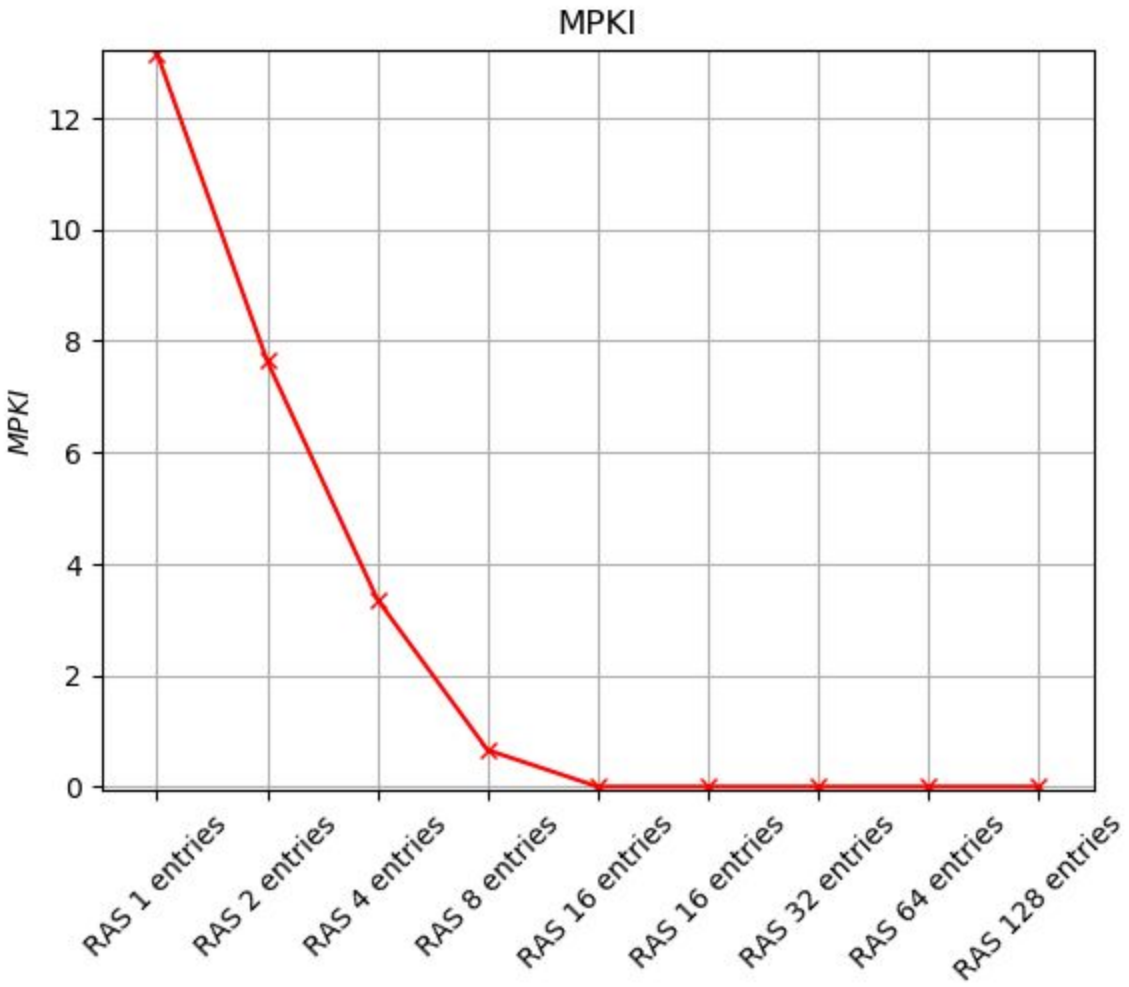
- 458.sjeng



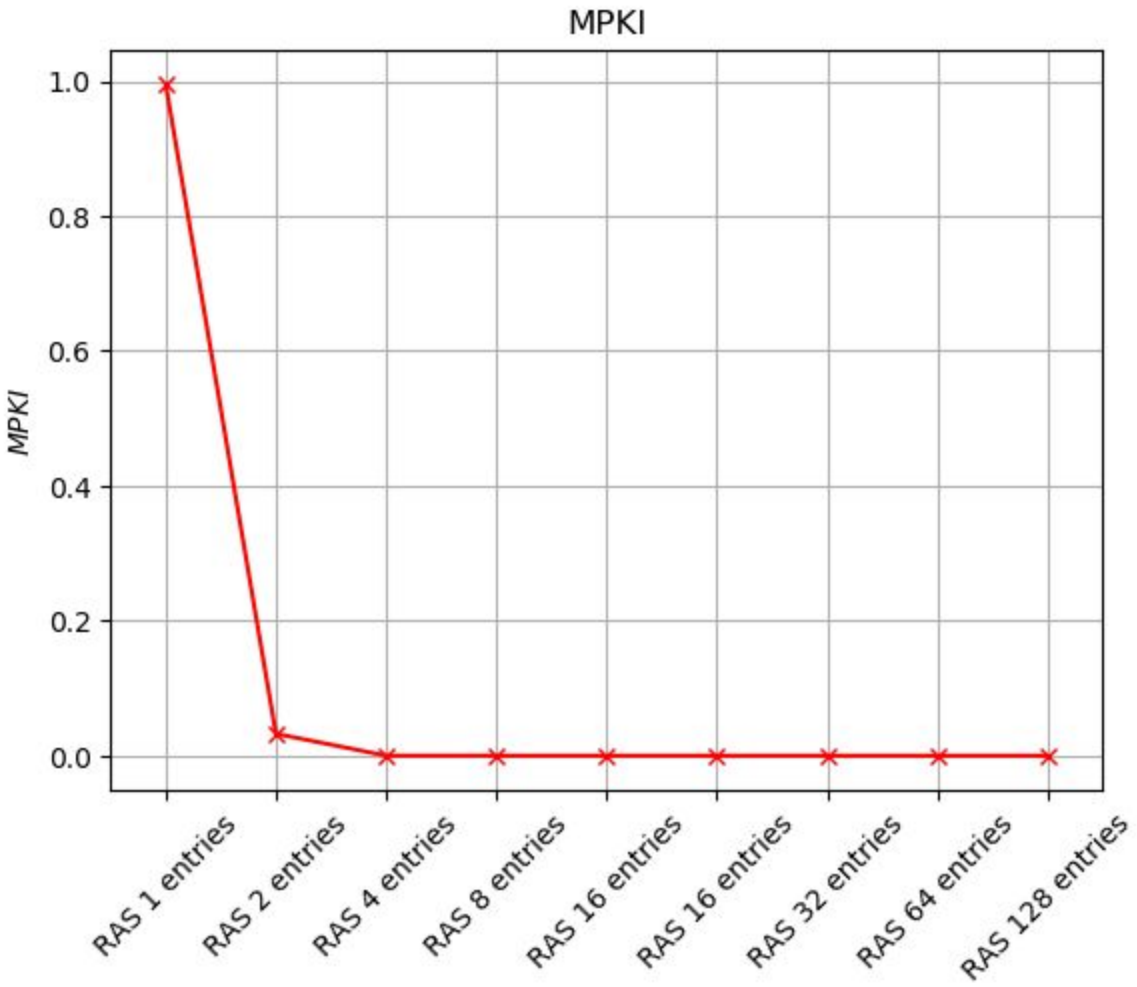
- 459.GemsFDTD


MPKI
MPKI
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
RAS 1 entries
RAS 2 entries
RAS 4 entries
RAS 8 entries
RAS 16 entries
RAS 16 entries
RAS 32 entries
RAS 64 entries
RAS 128 entries

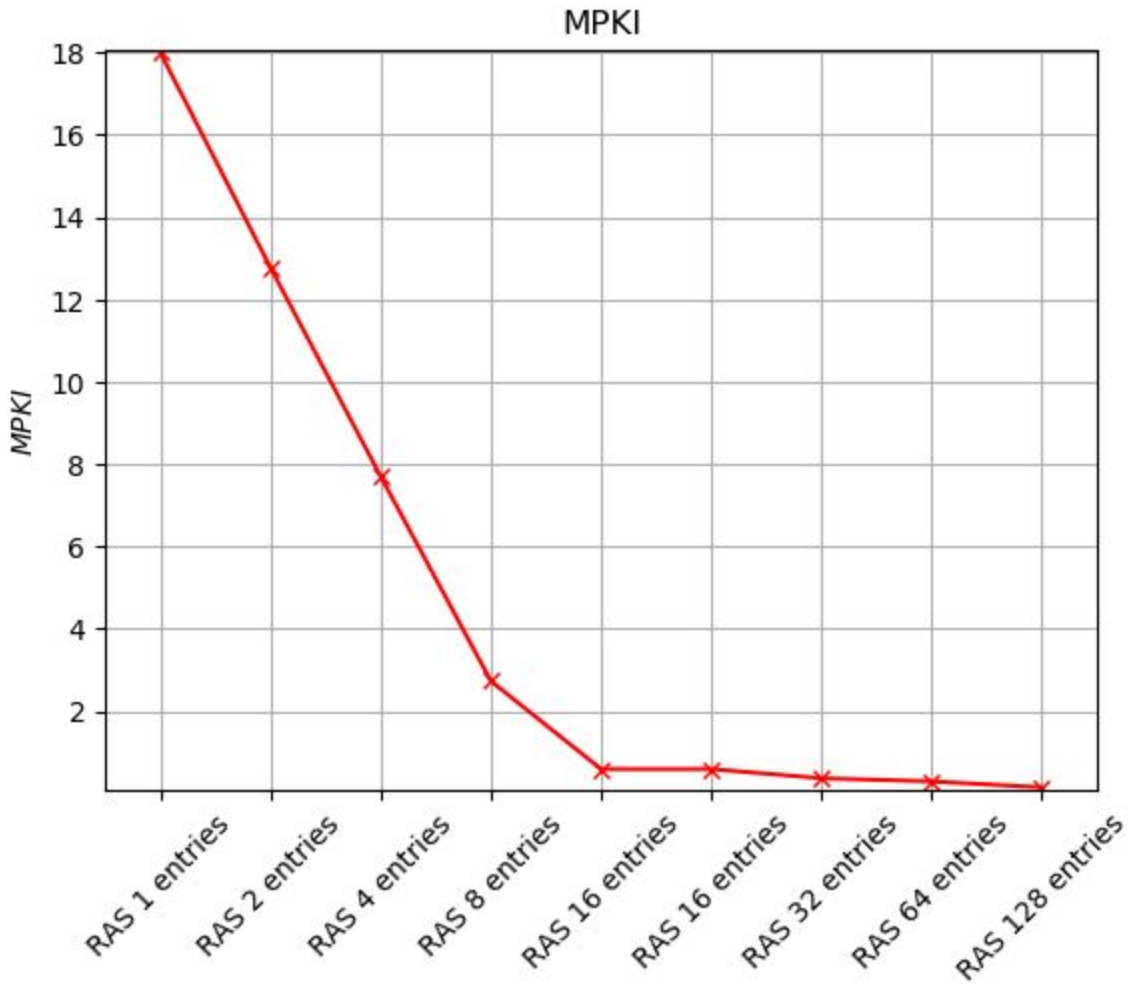
➢ 471.omnetpp


MPKI
MPKI
12
10
8
6
4
2
0
RAS 1 entries
RAS 2 entries
RAS 4 entries
RAS 8 entries
RAS 16 entries
RAS 16 entries
RAS 32 entries
RAS 64 entries
RAS 128 entries

➢ 473.astar



➢ 483.xalancbmk



Συμπεράσματα

Βλέπουμε στα παραπάνω διαγράμματα πως ξεκινάμε με πολύ κακή απόδοση για 1 εγγραφή στο RAS και όσο αυξάνεται ο αριθμός των εγγραφών έχουμε όλο και μικρότερη βελτίωση.
Τελικώς, σε πολλά μετροπρογράμματα από τις 4 εγγραφές ενώ συνολικά από τις 8 και πάνω η απόδοση μένει πάνω κάτω σταθερή. Έτσι θα επιλέγαμε RAS με 4 ή 8 εγγραφές, αναλόγως και την διαφορά στο κόστος.

4.5 Σύγκριση διαφορετικών predictors

Τέλος, θα συγκρίνουμε διαφορετικούς predictors μεταξύ τους, που είτε ήταν ήδη υλοποιημένοι είτε υλοποιήθηκαν με βαση τους περιορισμούς της εκφώνησης. Συγκεκριμένα θα συγκρίνουμε τους παρακάτω:

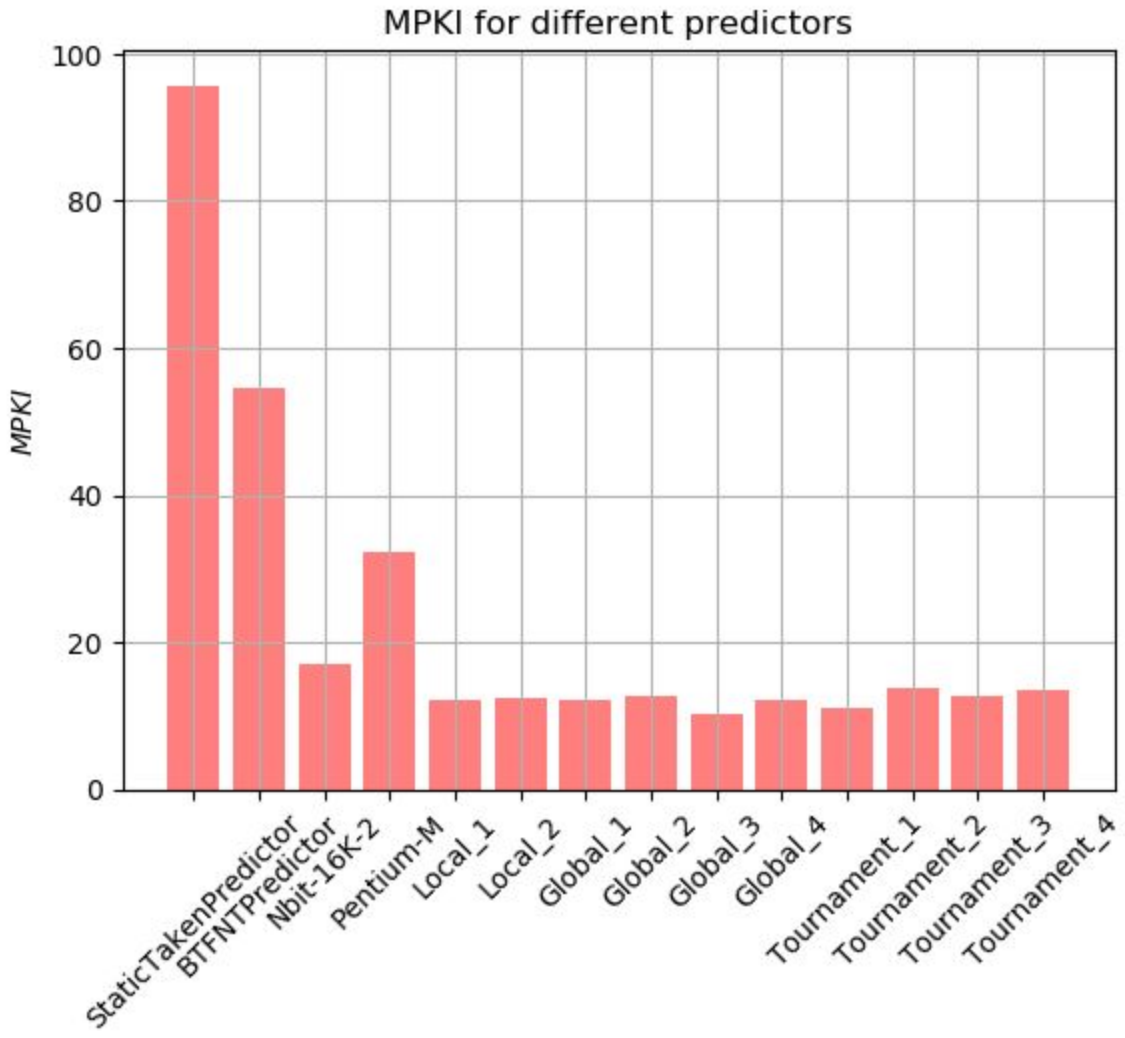
- Static Taken
- Static BTFNT
- Nbit-16K-2 (όπως επιλέχθηκε στο 4.2 ερώτημα)
- Pentium-M predictor
- Local-History two-level predictors
    - Με PHT entries 8192, PHT n-bit counter length = 2, BHT entries = 2048, BHTentry length = 8
    - Με PHT entries 8192, PHT n-bit counter length = 2, BHT entries = 4096, BHTentry length = 4

- Global History two level predictors
    - Με PHT entries = 16384, PHT n-bit counter length = 2, BHR length = 4
    - Με PHT entries = 8192, PHT n-bit counter length = 4, BHR length = 4
    - Με PHT entries = 16384, PHT n-bit counter length = 2, BHR length = 8
    - Με PHT entries = 8192, PHT n-bit counter length = 4, BHR length = 8

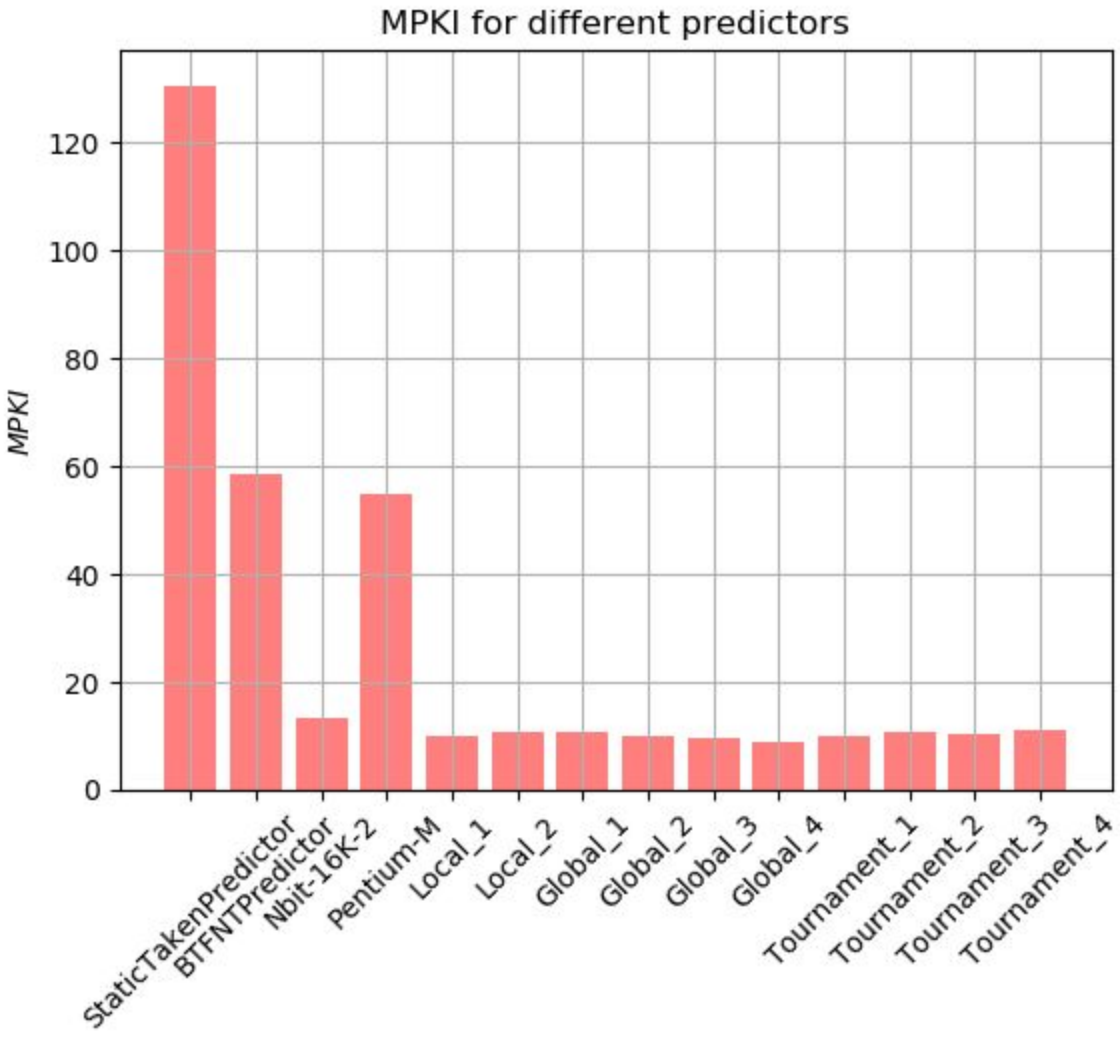
- Tournament Hybrid Predictors με meta-predictor 2-bit με 512 entries και P_0, P_1:
    - GlobalHistory-PHT(8192,2)-BHR(4), LocalHistory-PHT(4096,2)-BHT(2048,4)
    - LocalHistory-PHT(8192,1)-BHT(2048,4), LocalHistory-PHT(4096,2)-BHT(2048,4)
    - GlobalHistory-PHT(4096,4)-BHR(4), Nbit-16K-1
    - LocalHistory-PHT(4096,2)-BHT(2048,4),Nbit-16K-1

Οι predictors Local, Global και Tournament στα διαγράμματα συμβολίζονται αριθμημένα με την σειρά που αναφέρονται παραπάνω για εξοικονόμιση χώρου στα διαγράμματα.

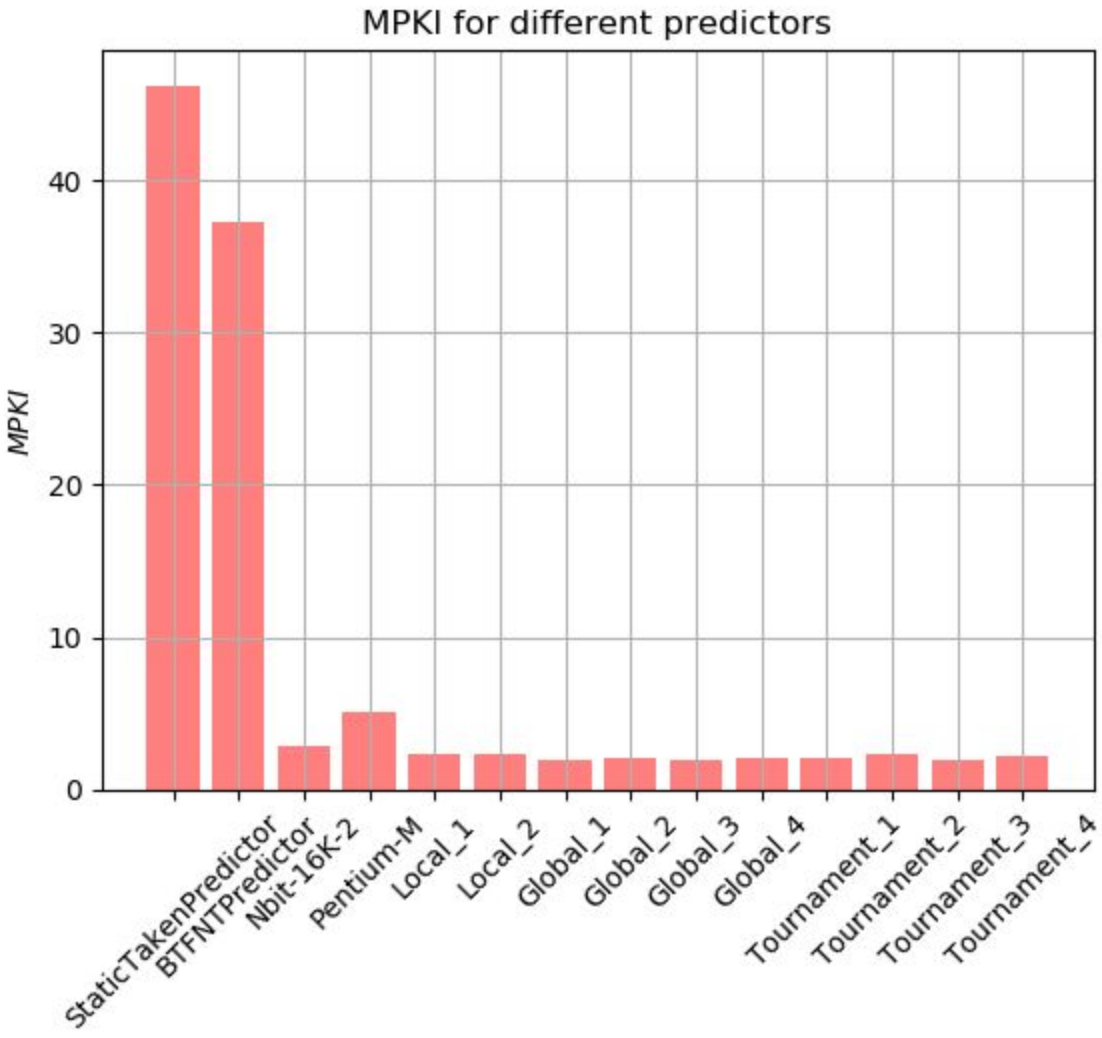
➢ 403.gcc


MPKI for different predictors
MPKI
100
80
60
40
20
0
StaticTakenPredictor
BTFNTPredictor
Nbit-16K-2
Pentium-M
Local_1
Local_2
Global_1
Global_2
Global_3
Global_4
Tournament_1
Tournament_2
Tournament_3
Tournament_4

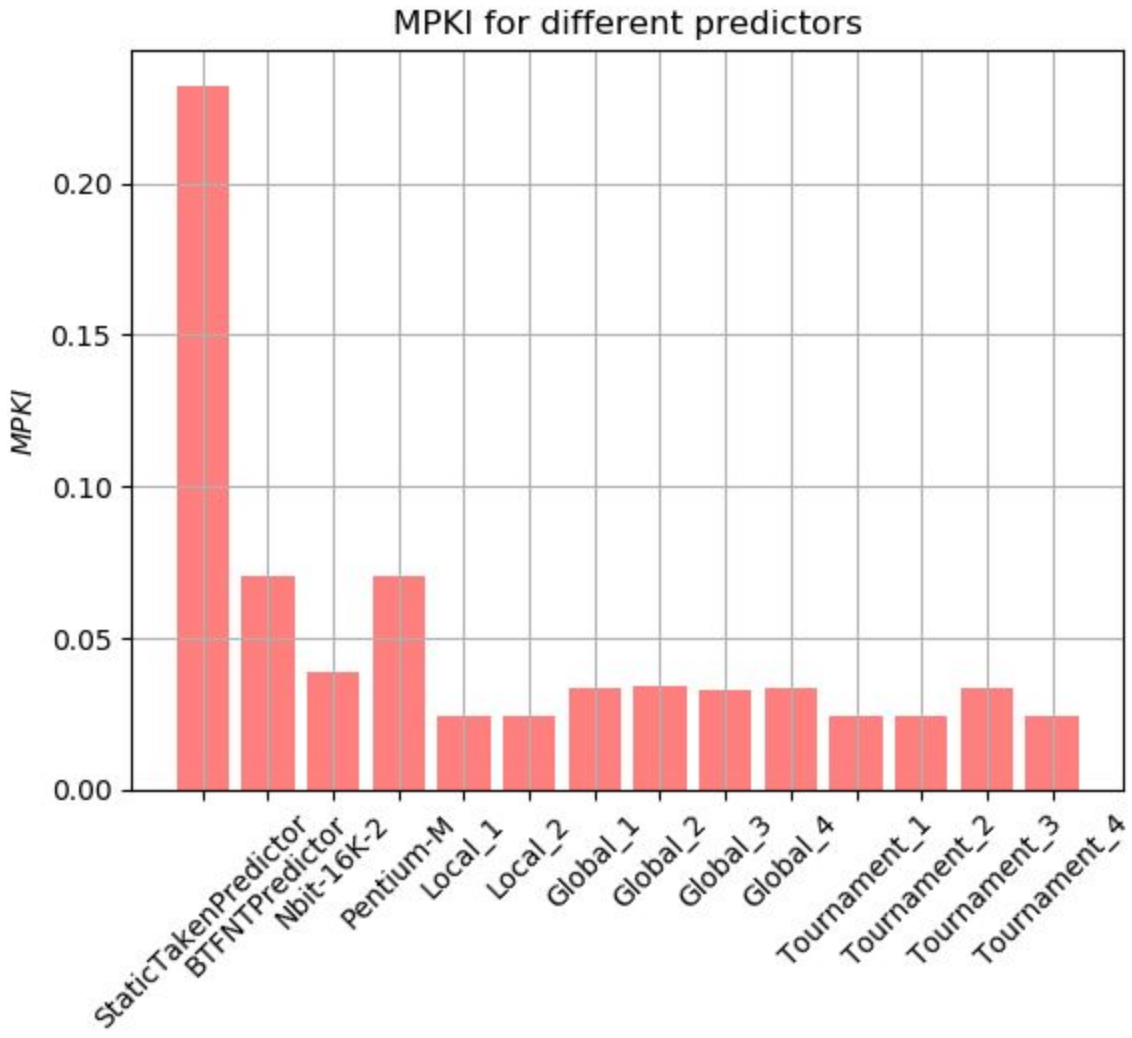
➢ 429.mcf


MPKI for different predictors
MPKI
0
20
40
60
80
100
120
StaticTakenPredictor
BTFNTPredictor
Nbit-16K-2
Pentium-M
Local_1
Local_2
Global_1
Global_2
Global_3
Global_4
Tournament_1
Tournament_2
Tournament_3
Tournament_4

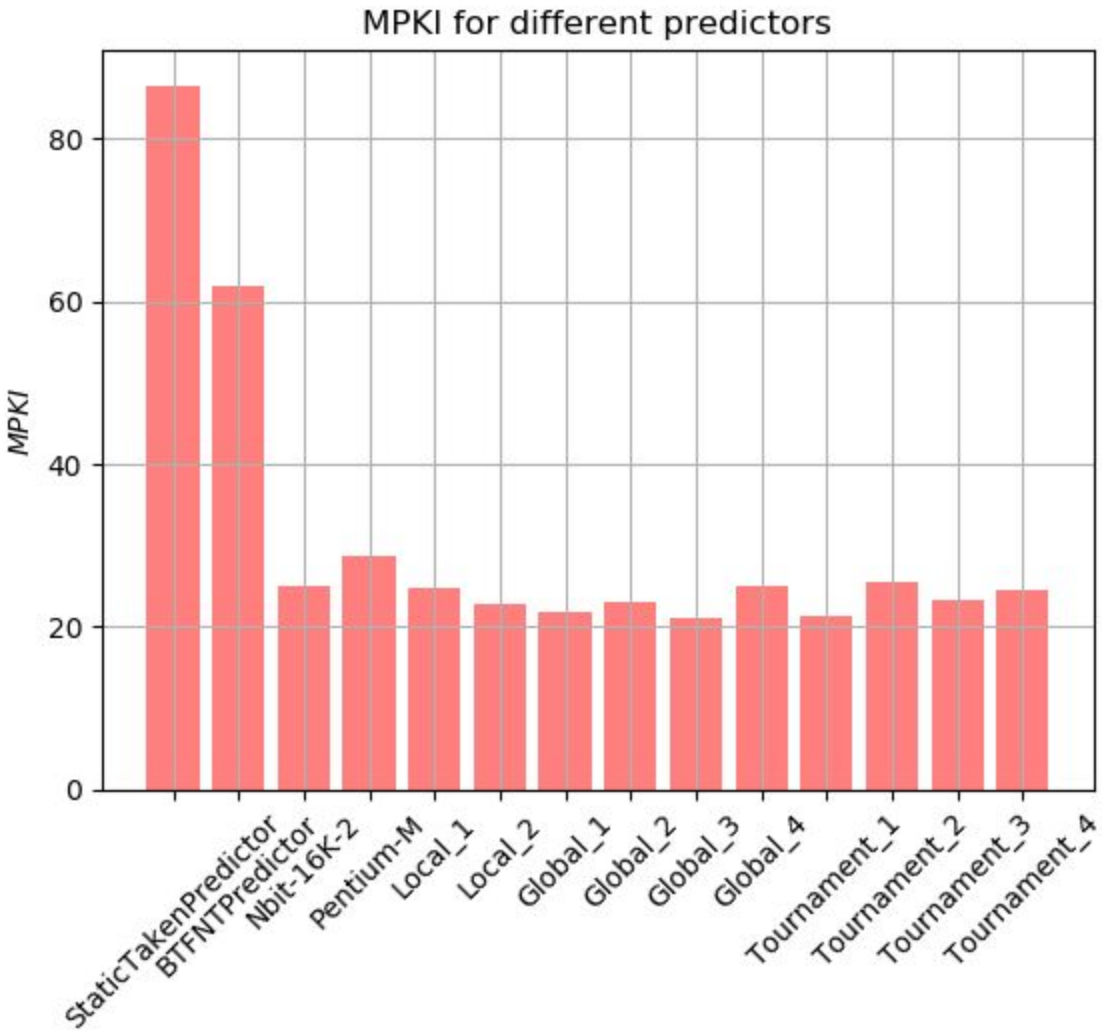
➢ 434.zeusmp



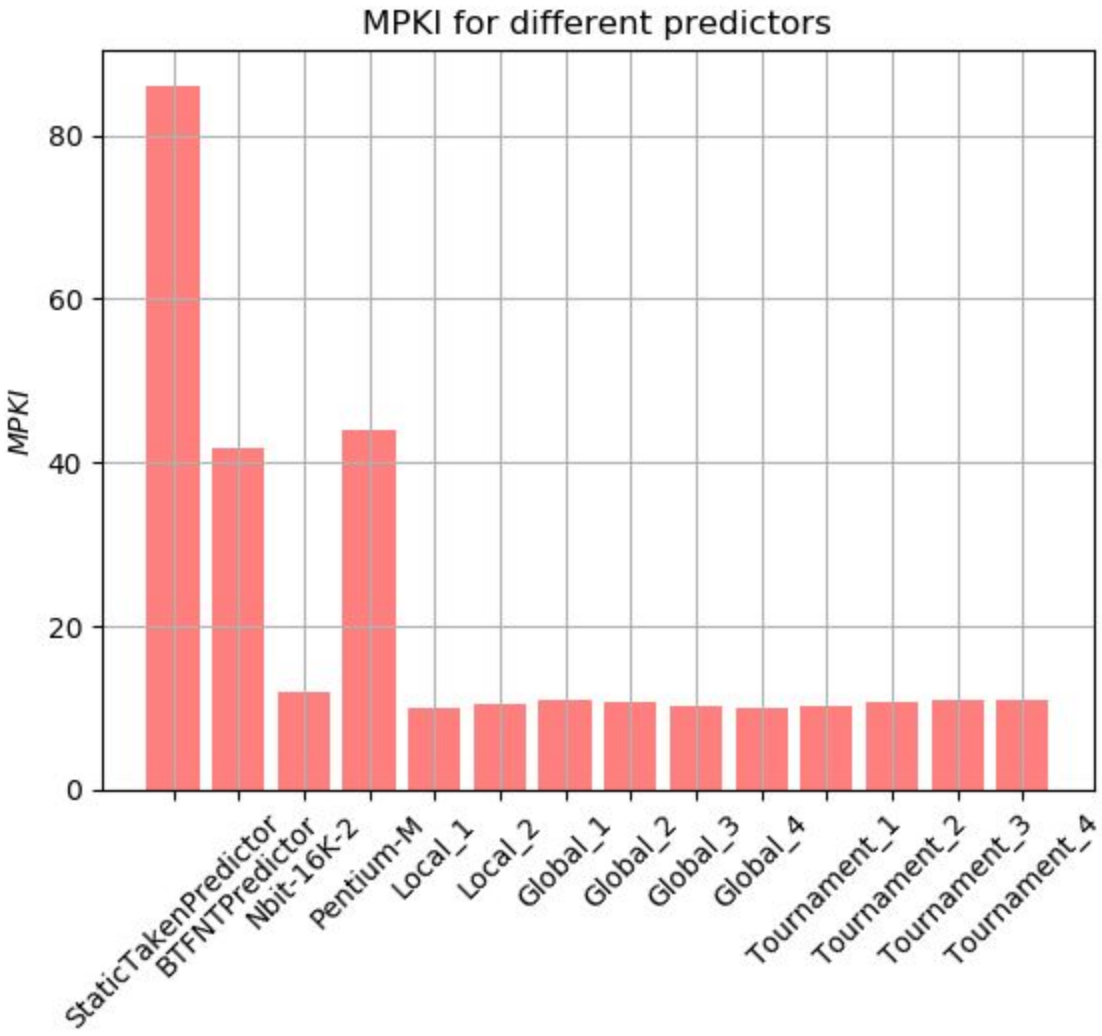
➢ 436.cactusADM


MPKI for different predictors
MPKI
0.20
0.15
0.10
0.05
0.00
StaticTakenPredictor
BTFNTPredictor
Nbit-16K-2
Pentium-M
Local_1
Local_2
Global_1
Global_2
Global_3
Global_4
Tournament_1
Tournament_2
Tournament_3
Tournament_4

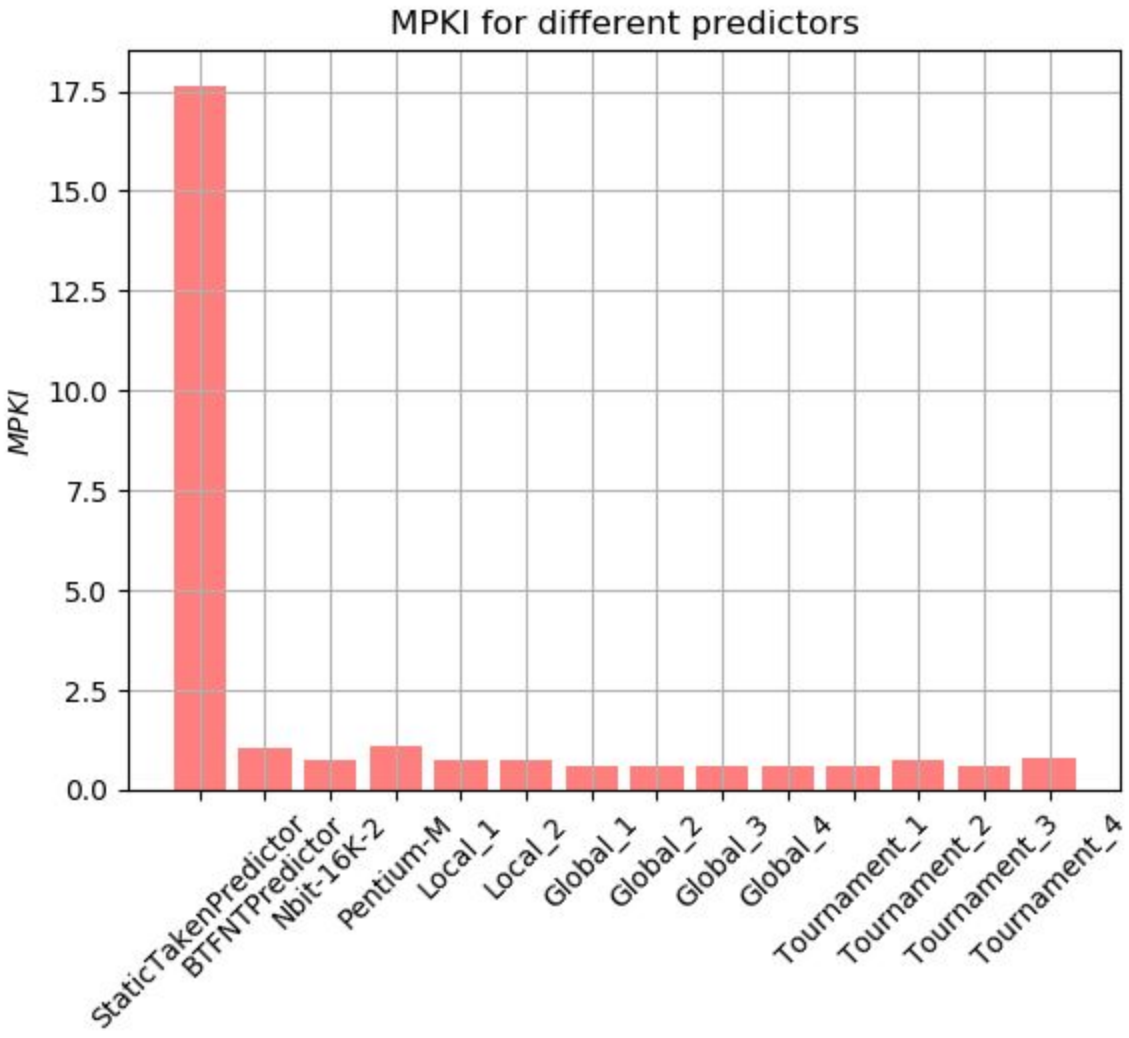
➢ 445.gobmk


MPKI for different predictors
MPKI
0
20
40
60
80
StaticTakenPredictor
BTFNTPredictor
Nbit-16K-2
Pentium-M
Local_1
Local_2
Global_1
Global_2
Global_3
Global_4
Tournament_1
Tournament_2
Tournament_3
Tournament_4

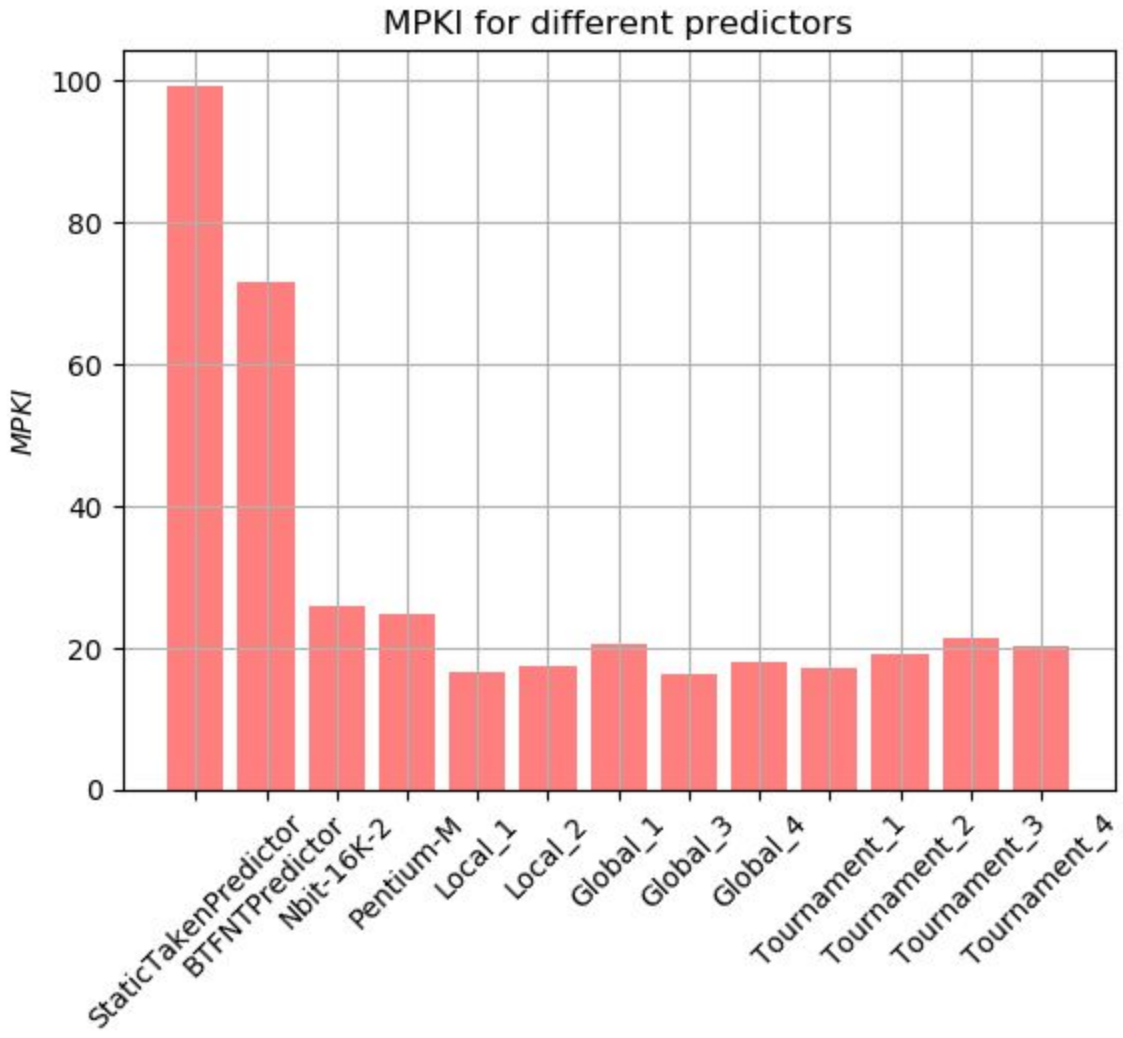
➢ 450.soplex


MPKI for different predictors
MPKI
0
20
40
60
80
StaticTakenPredictor
BTFNTPredictor
Nbit-16K-2
Pentium-M
Local_1
Local_2
Global_1
Global_2
Global_3
Global_4
Tournament_1
Tournament_2
Tournament_3
Tournament_4

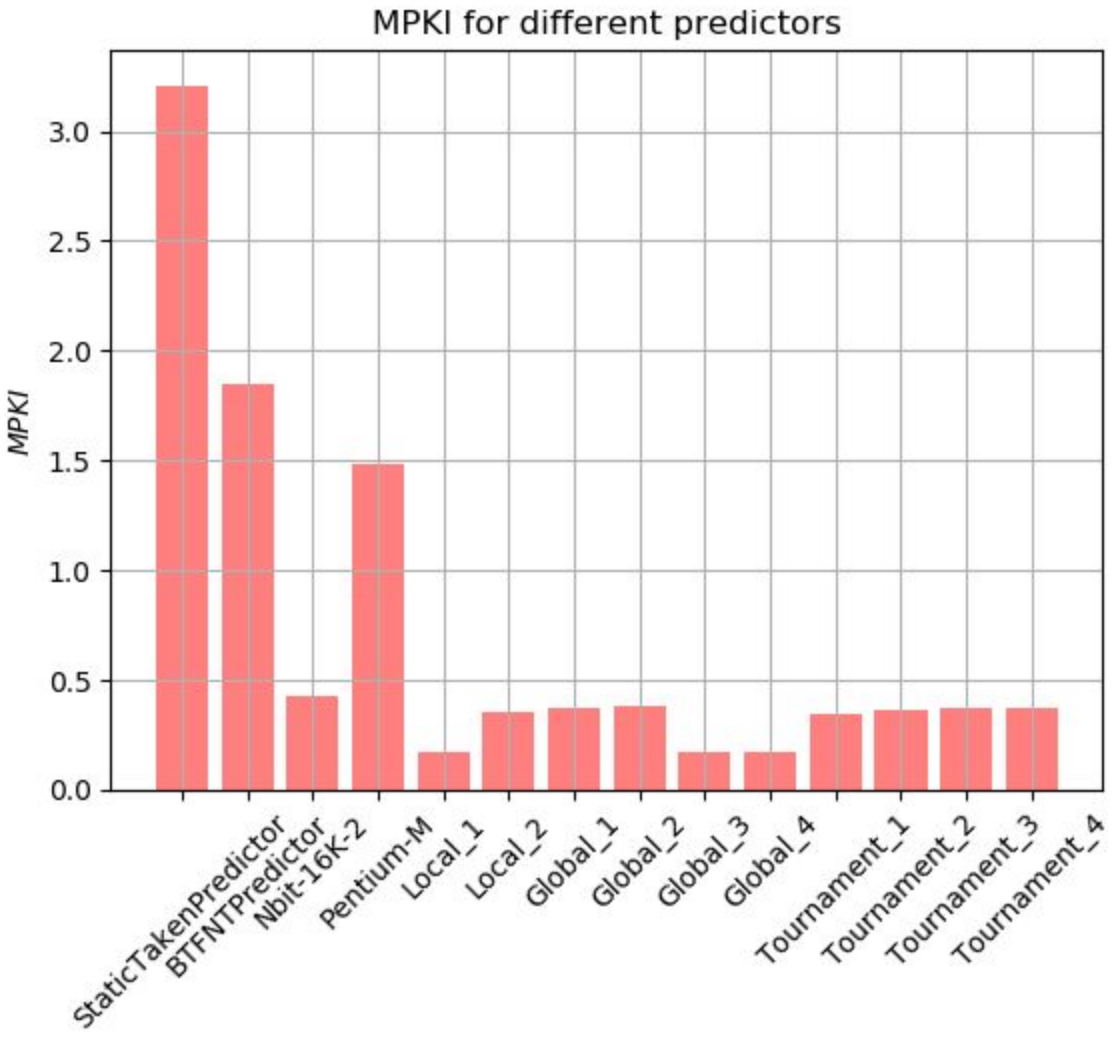
➢ 456.hmmer


MPKI for different predictors
MPKI
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
StaticTakenPredictor
BTFNTPredictor
Nbit-16K-2
Pentium-M
Local_1
Local_2
Global_1
Global_2
Global_3
Global_4
Tournament_1
Tournament_2
Tournament_3
Tournament_4

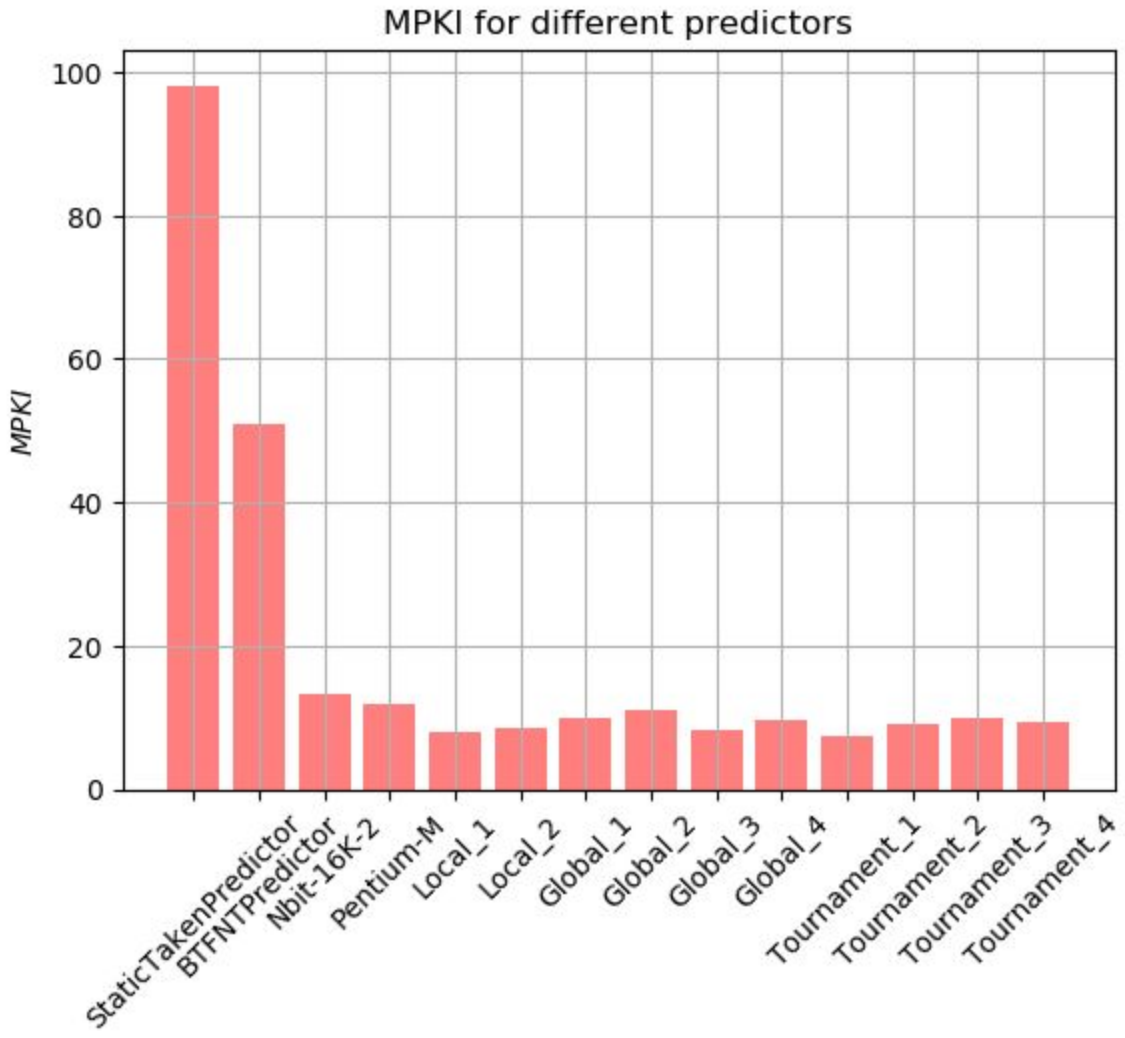
➢ 458.sjeng



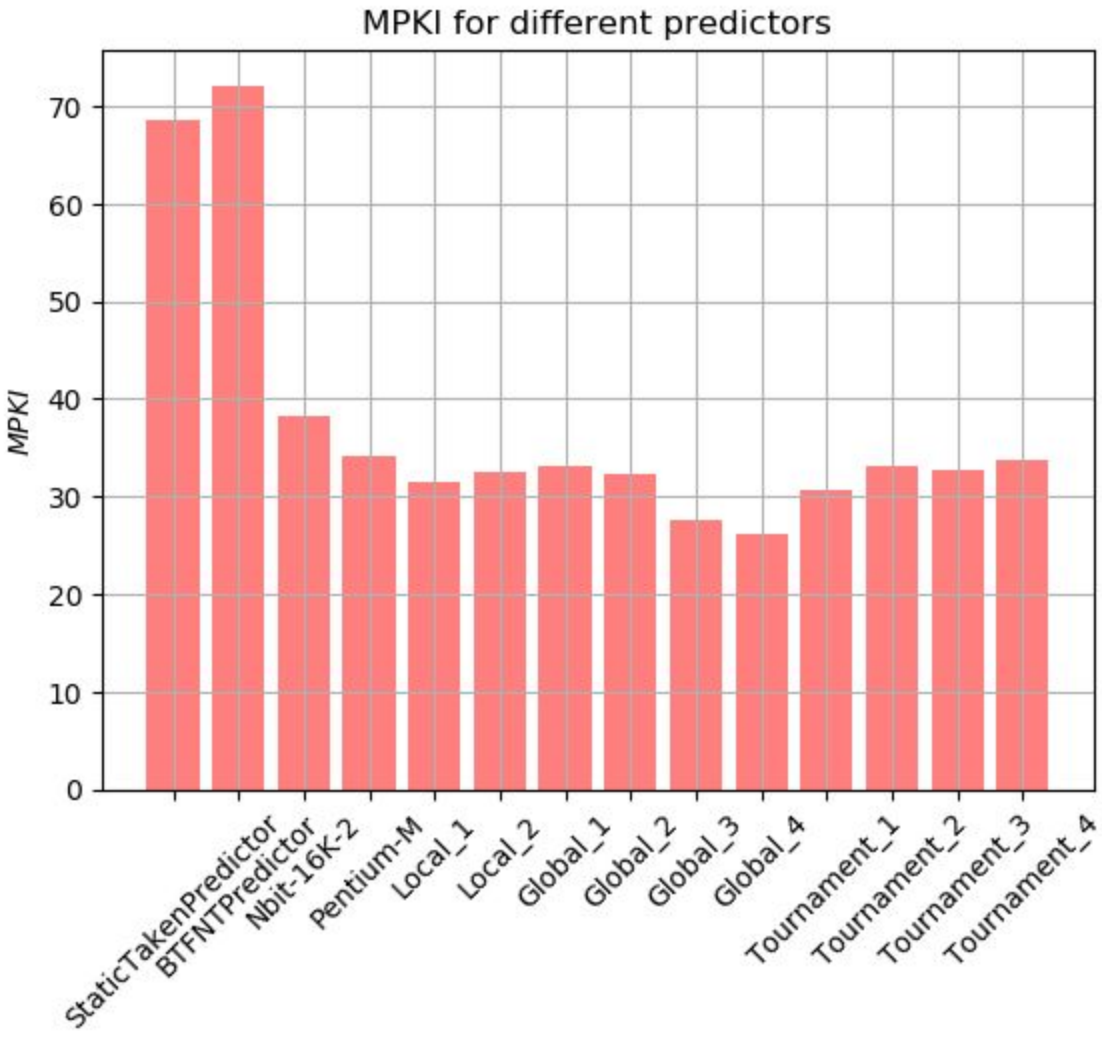
➢ 459.GemsFDTD


MPKI for different predictors
MPKI
3.0
2.5
2.0
1.5
1.0
0.5
0.0
StaticTakenPredictor
BTFNTPredictor
Nbit-16K-2
Pentium-M
Local_1
Local_2
Global_1
Global_2
Global_3
Global_4
Tournament_1
Tournament_2
Tournament_3
Tournament_4

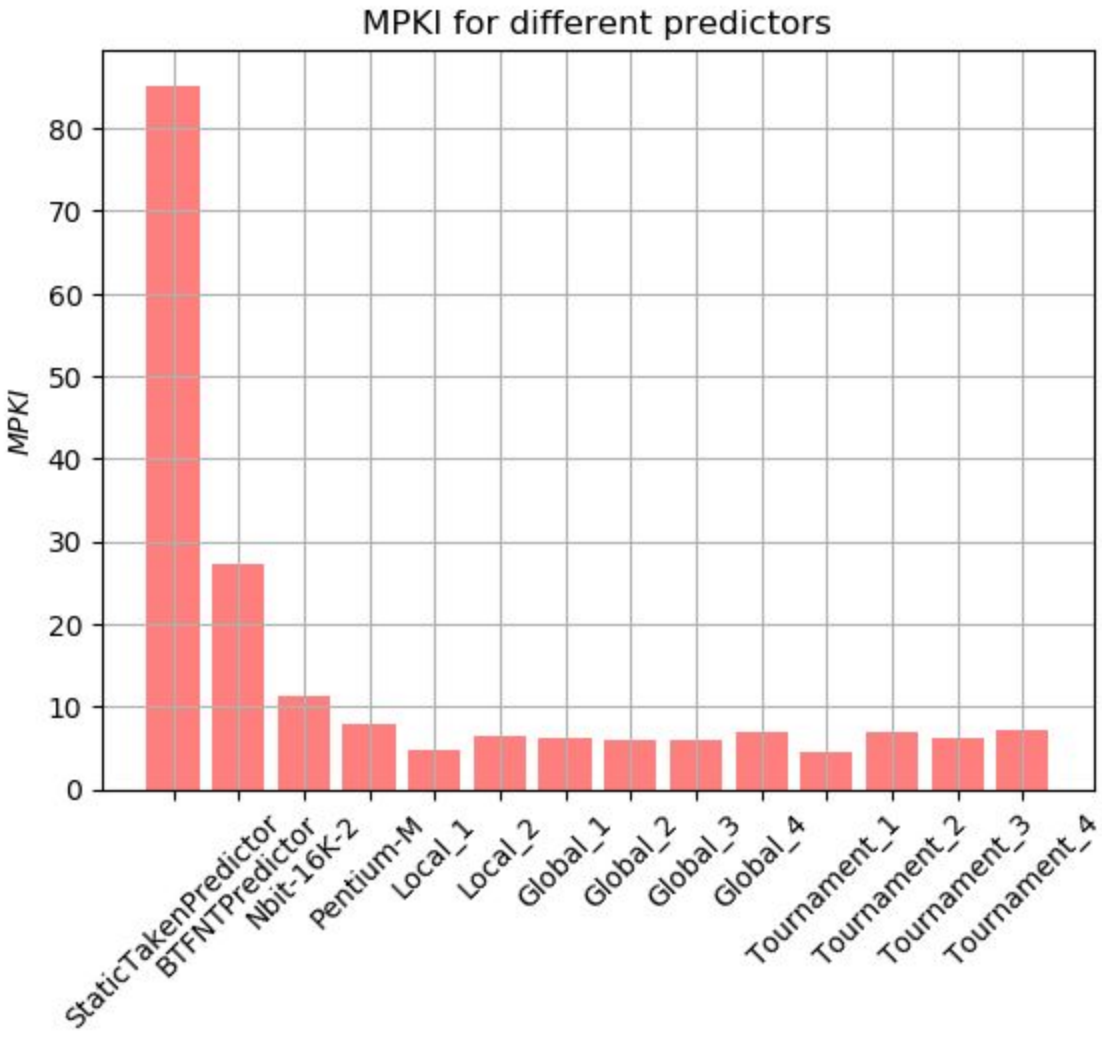
➢ 471.omnetpp


MPKI for different predictors
MPKI
100
80
60
40
20
0
StaticTakenPredictor
BTFNTPredictor
Nbit-16K-2
Pentium-M
Local_1
Local_2
Global_1
Global_2
Global_3
Global_4
Tournament_1
Tournament_2
Tournament_3
Tournament_4

➢ 473.astar

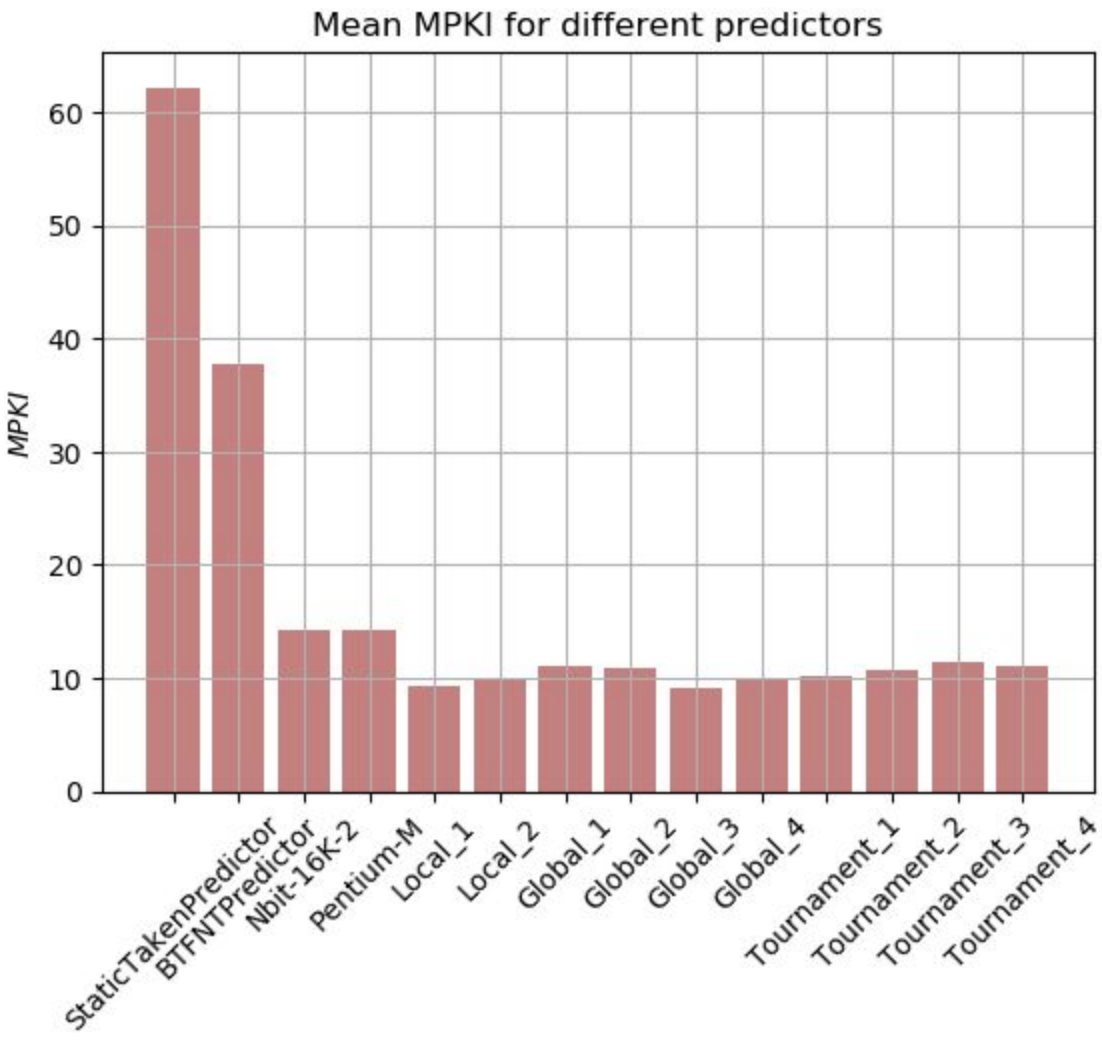


➢ 483.xalancbmk



Συμπεράσματα

Με βάση τα αποτελέσματα των προσομοιώσεων υπολογίστηκε και ο μέσος όρος MPKI των παραπάνω μετροπρογραμμάτων και είναι αυτός που φαίνεται παρακάτω.



Παρατηρούμε πως ο Static Taken Predictor είναι σχεδόν σε κάθε περίπτωση η χείριστη επιλογή predictor (με εξαίρεση στο μετροπρόγραμμα astar όπου έρχεται δεύτερο). Ακόμη ο BNFT predictor έχει πολύ κακή απόδοση στις περισσότερες περιπτώσεις (εξαίρεση έχουμε στο μετροπρόγραμμα hmmer).

Οι υπόλοιποι predictors δεν έχουν μεγάλες διαφορές με εξαίρεση συγκεκριμένα μετροπρογράμματα και συνολικά στο τελικό διάγραμμα μέσου όρου βλέπουμε μικρές διακυμάνσεις. Μικρότερο MPKI και άρα καλύτερη απόδοση έχει ο Global History two-level predictor με 16384 PHT entries, μήκος PHT n-bit counter ίσο με 2 και μήκος BHR ίσο με 8, μιας και παρουσιάζει MPKI ίσο με 9.068, τον οποίο και θα επιλέγαμε.