CACHE ON DELIVERY

Data Prefetchers And Cache Replacement Interaction

CS 305/341 PROJECT

Vikas Panwar
Megha Baboria
Dhakne Ajay
Pawan Kumar

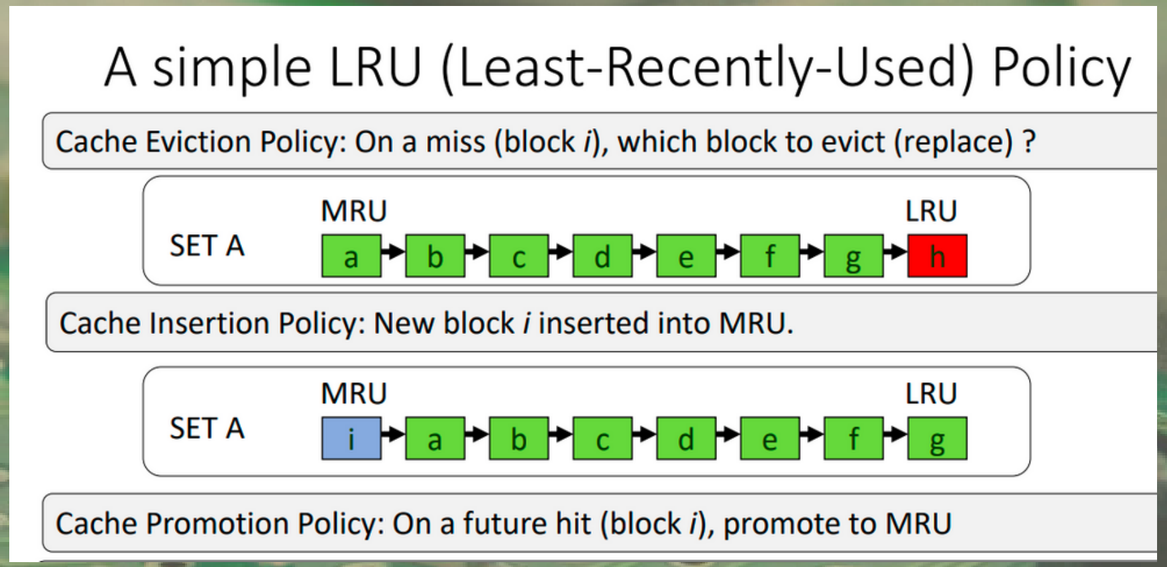
Prof. Biswa

Introduction

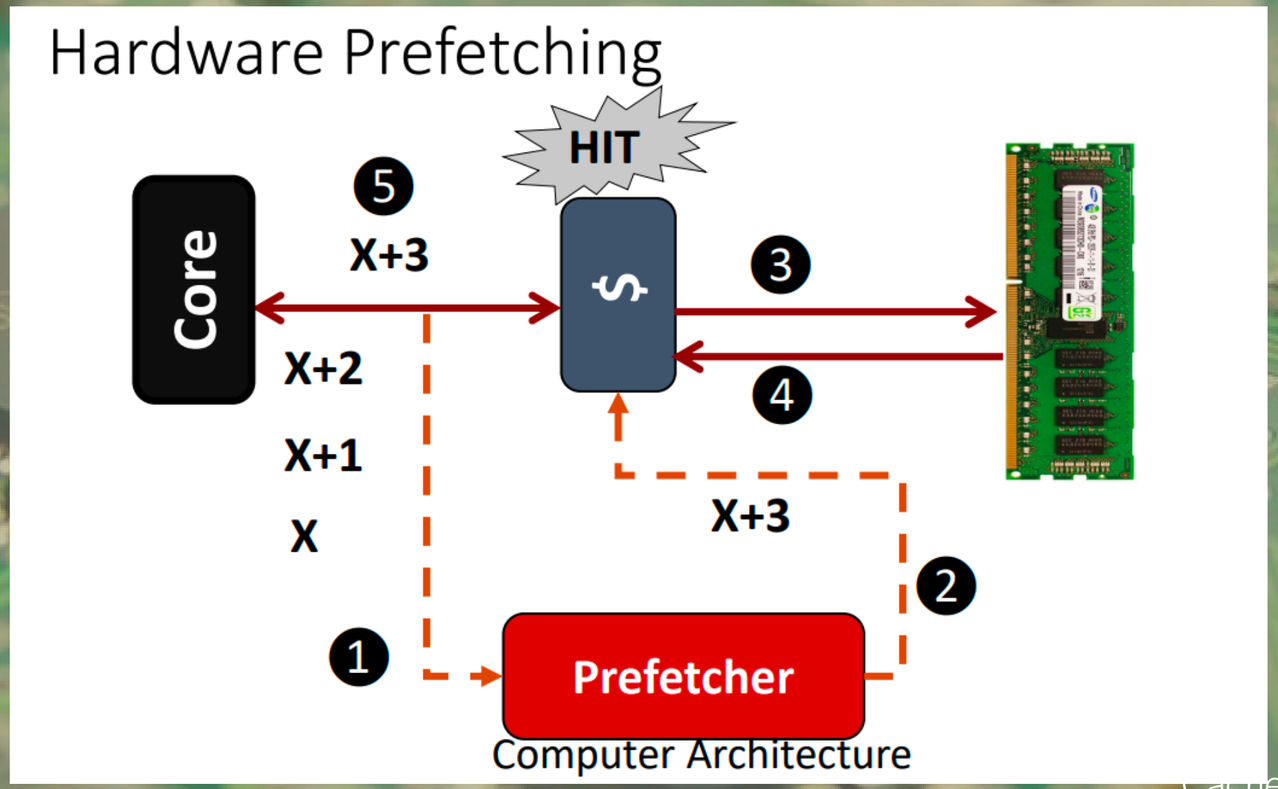
Hardware prefetching and last-level cache (LLC) management are two independent mechanisms to mitigate the growing latency to memory. However, the interaction between LLC management and hardware prefetching has received very little attention.

Introduction

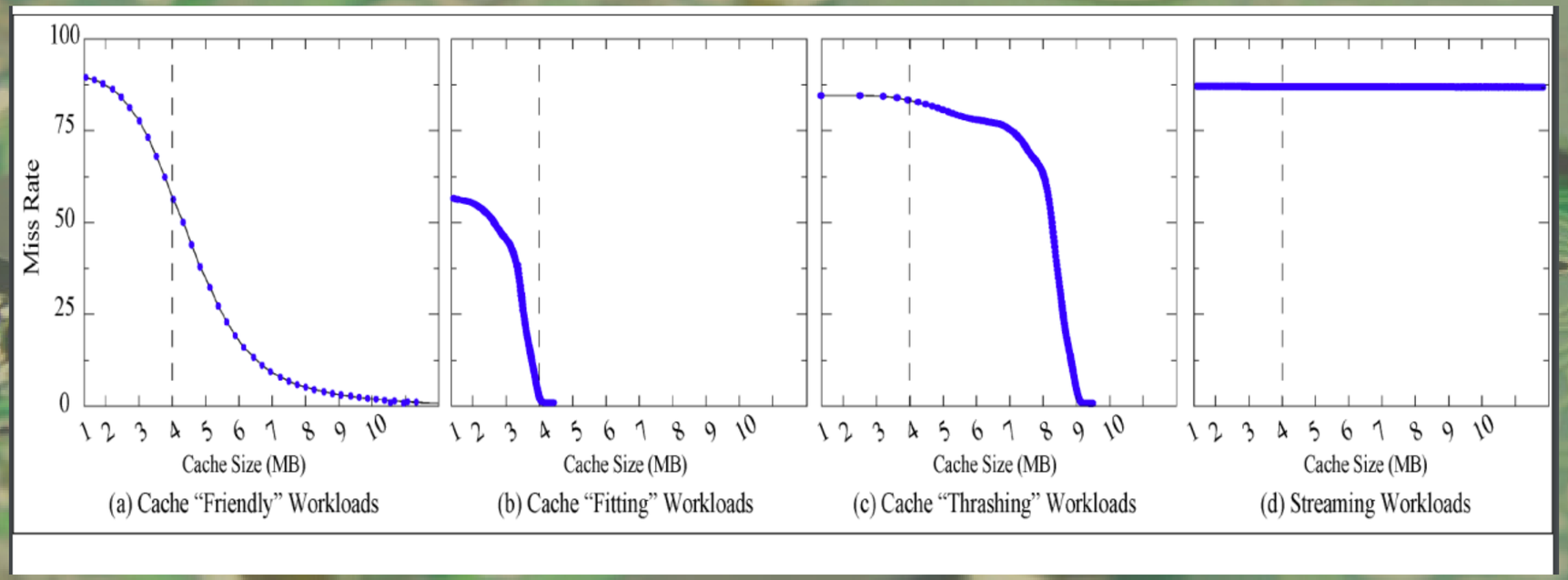
Cache replacement policies were devised to help improve latency issues



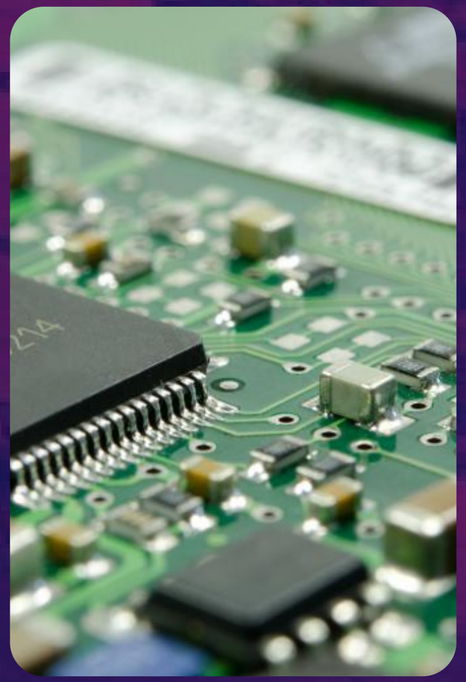
Prefetchers are used to hide the DRAM latency from processor



Cache replacement policies that are not prefetch aware can cause cache pollution



Probem with LRU for diffrent workloads



Cache Replacement Policies

- LRU(baseline policy)
- DRRIP
- Hawkeye
- SHiP
- SHiP++

Hawkeye

- Classifies PCs into cache-friendly and cache-averse.

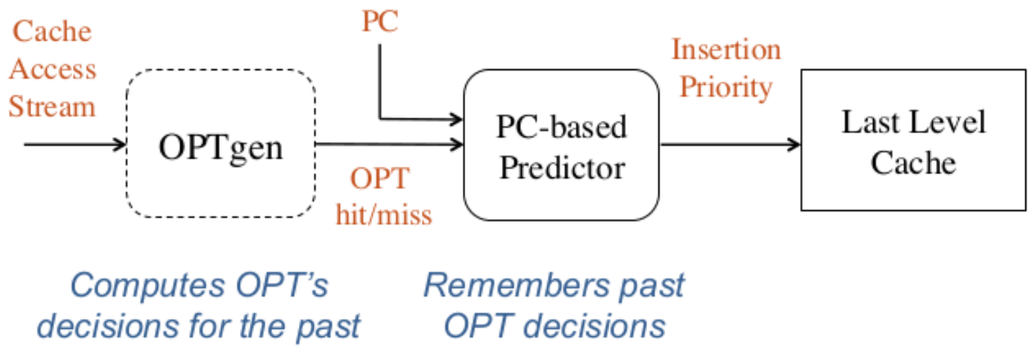
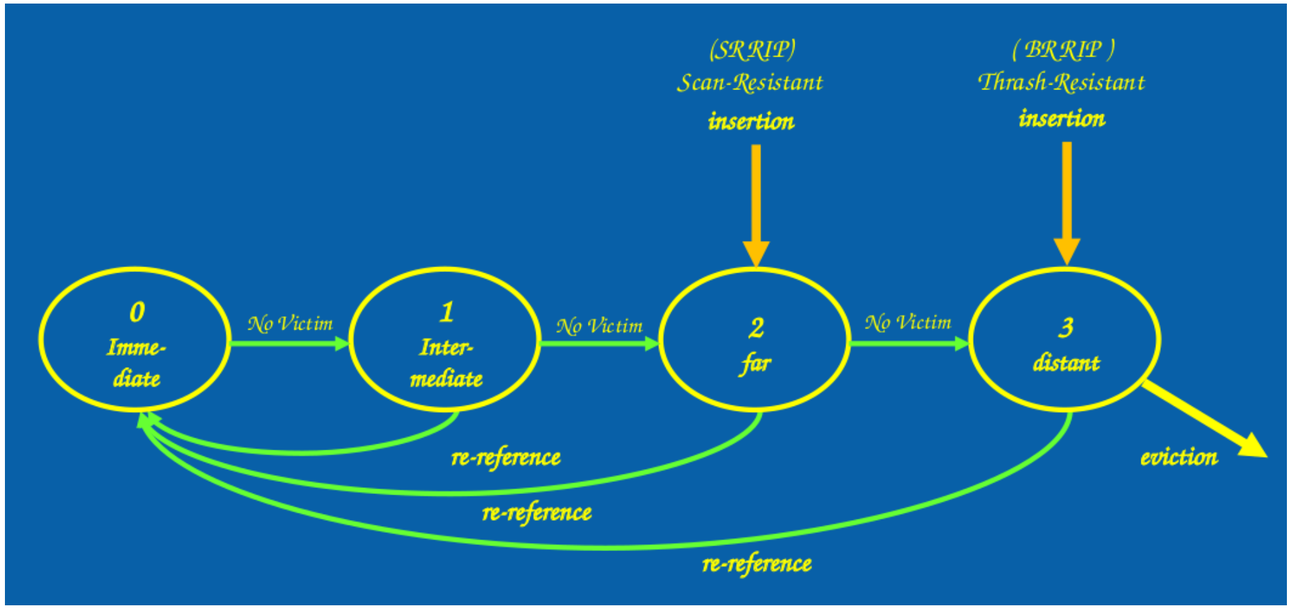


Figure 1: Block diagram of the Hawkeye replacement algorithm.

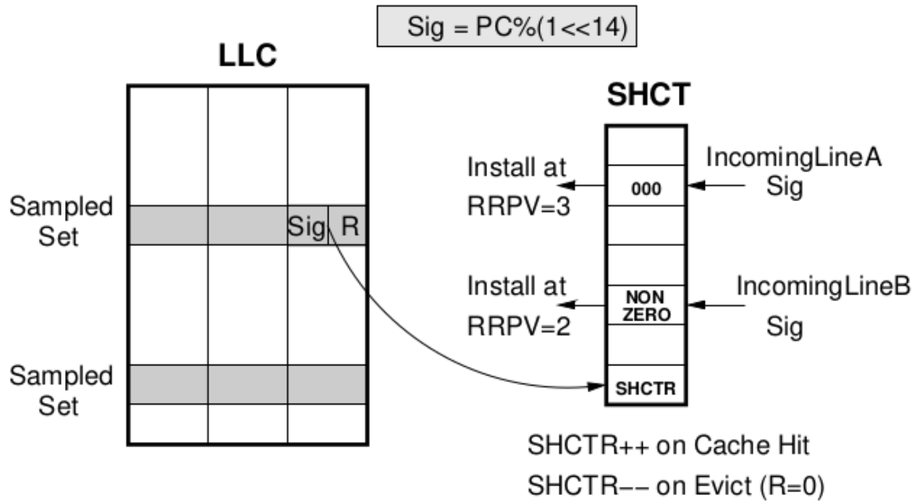
DRRIP

- On a hit update the RRPV of a cache line to 0. On replacements evit a line with rrpv of 3.(2-bit counter)
- Chooses a cache line with RRPV of 3 for replacement.
- Use set dueling to determine SRRP or BRRP.



SHiP

- No cache awareness
- Not prefetch aware

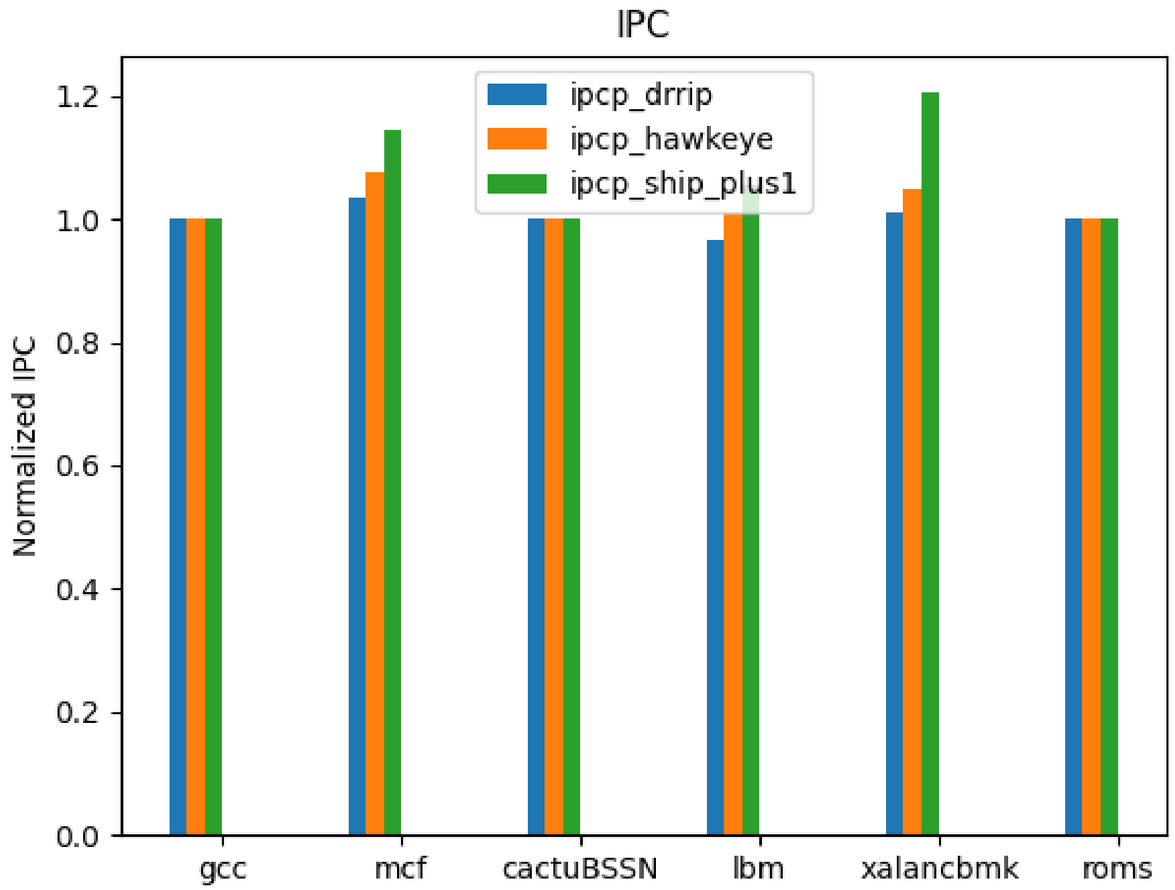


SHiP++ (Improved version of SHIP)

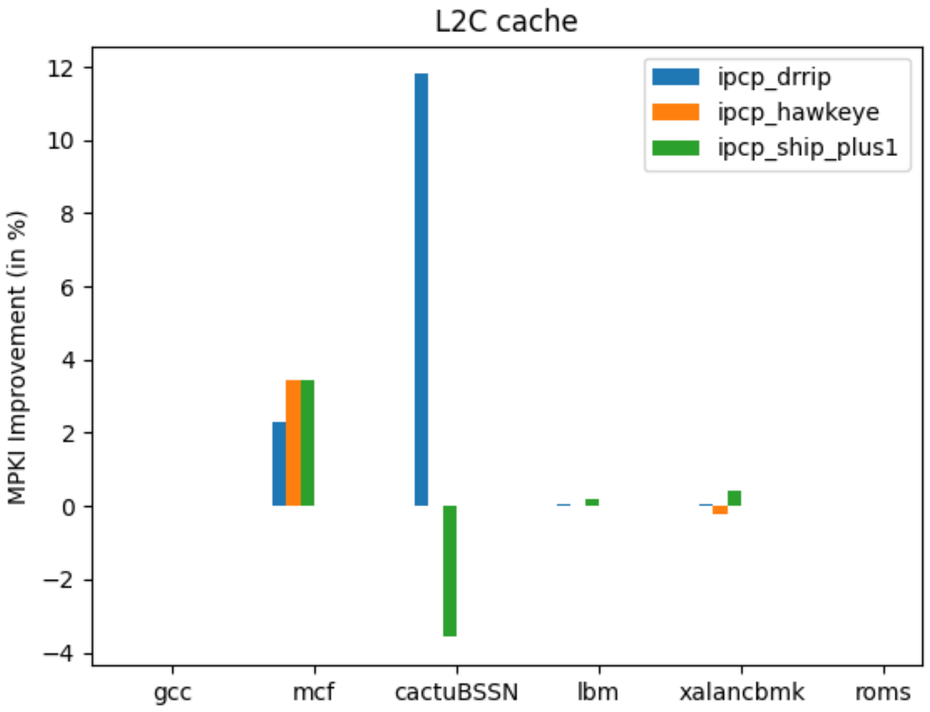
- Cache awareness
- Prefetch aware
- Writeback aware
- Improved SHCT training

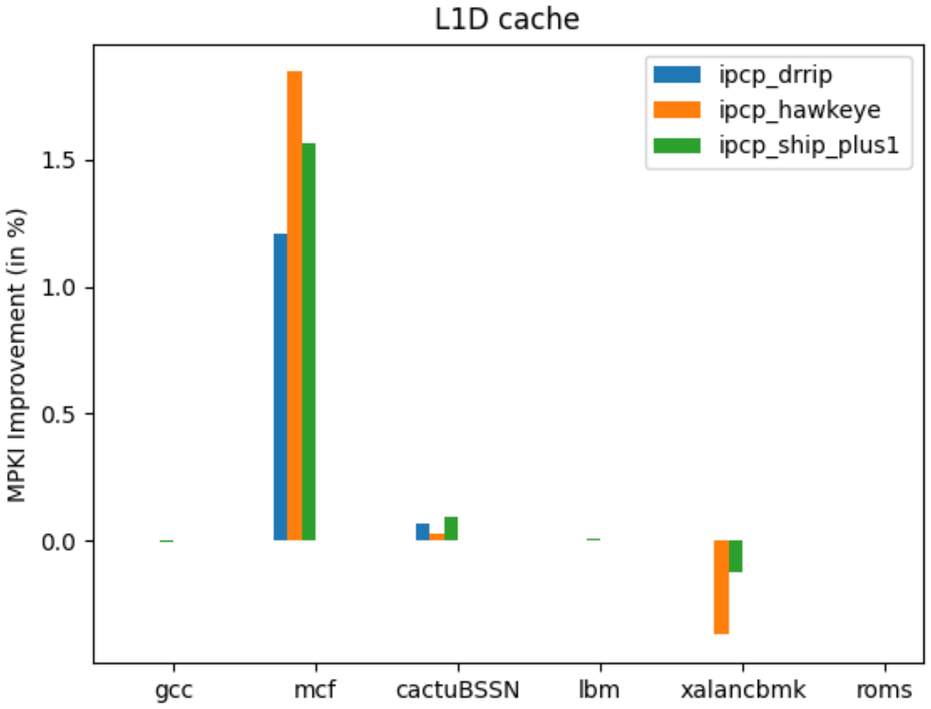
RESULTS

After runnig IPCP prefetcher at L1 and L2 with different cache replacement policies like LRU, SHiP++ and Hawkeye following results were obtained

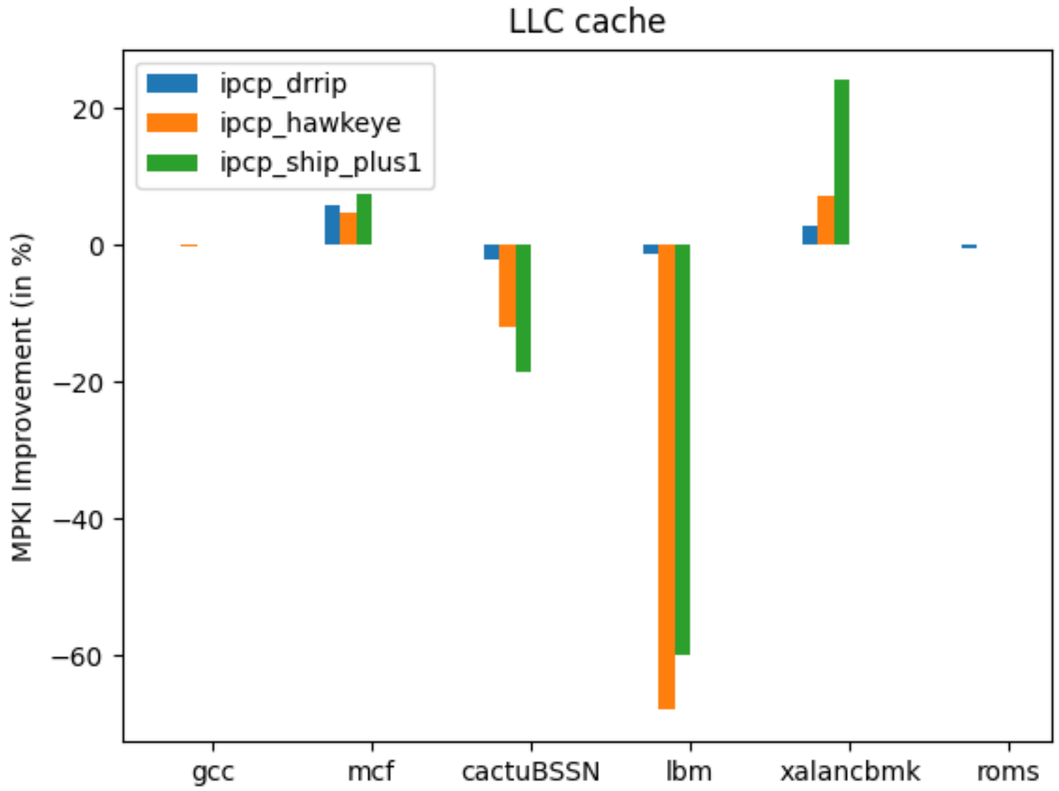


RESULTS

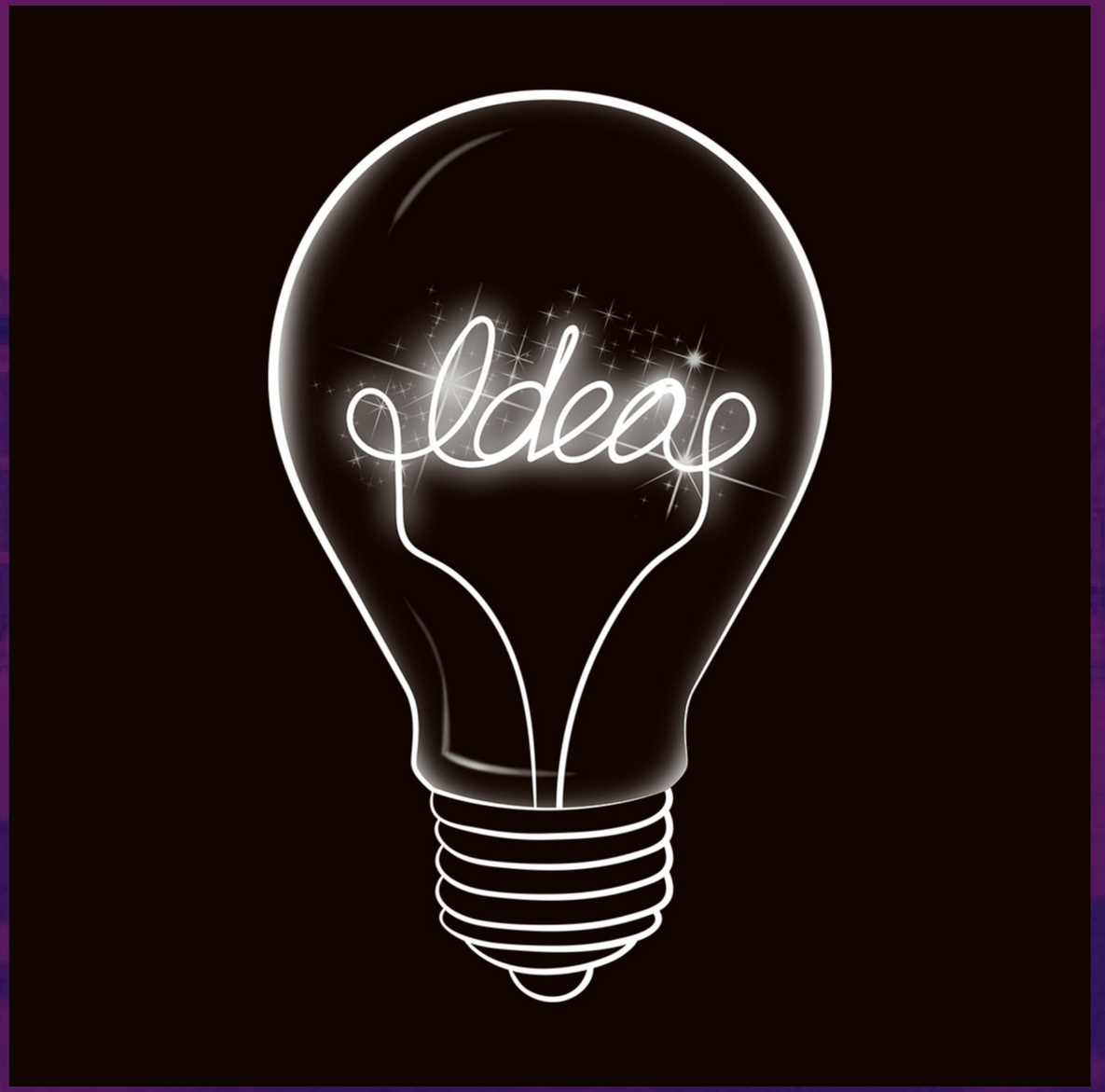

L2C cache
ipcp_drrip
ipcp_hawkeye
ipcp_ship_plus1
MPKI Improvement (in %)
12
10
8
6
4
2
0
−2
−4
gcc
mcf
cactuBSSN
lbm
xalancbmk
roms


L1D cache
ipcp_drrip
ipcp_hawkeye
ipcp_ship_plus1
MPKI Improvement (in %)
1.5
1.0
0.5
0.0
gcc
mcf
cactuBSSN
lbm
xalancbmk
roms

RESULTS


LLC cache
ipcp_drrip
ipcp_hawkeye
ipcp_ship_plus1
MPKI Improvement (in %)
20
0
−20
−40
−60
gcc
mcf
cactuBSSN
lbm
xalancbmk
roms

Improvement Ideas and Suggestions



HERE WE GO!!

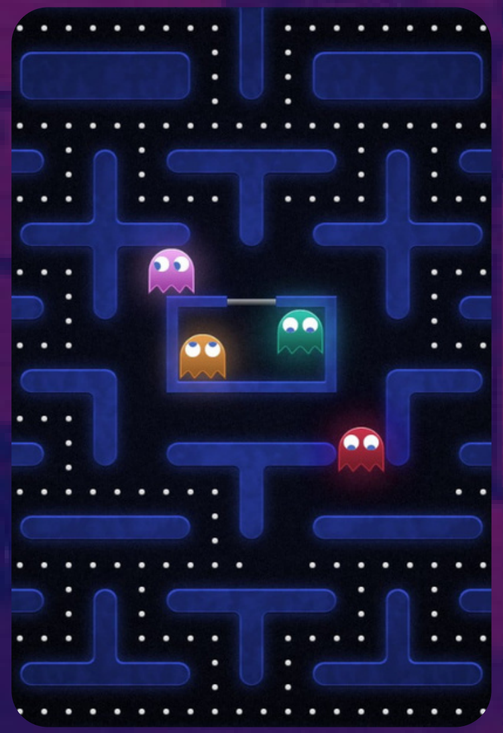
1. Making DRRIP prefetch aware can result in improvemnt in its performance
2. Using a less aggressive prefetcher can improve performance of Hawkeye
3. Inserting Cache Averse lines with a lower RRPV than MAX can result in improvement

HERE WE GO!!

4 Varying alpha - P threshold dynamically may result in better performance, alpha ∈ {P,D}

5 We can get performance gain by augmenting Hawkeye’s predictions to include confidence

Suggestion 1



PACman

Dynamically estimates and mitigates the degree of prefetch-induced cache interference by modifying the cache insertion and hit promotion policies to treat demand and prefetch requests differently.

```
if(cacheMiss && isSDMSet_SRRIP+PACManH[setIndex])
    cnt_SRRIP+PACManH     += 2;
    cnt_SRRIP+PACManHM    -= 1;
    cnt_BRRIP+PACManHM    -= 1;

if(cacheMiss && isSDMSet_SRRIP+PACManHM[setIndex])
    cnt_SRRIP+PACManH     -= 1;
    cnt_SRRIP+PACManHM    += 2;
    cnt_BRRIP+PACManHM    -= 1;

if(cacheMiss && isSDMSet_BRRIP+PACManH[setIndex])
    cnt_SRRIP+PACManH     -= 1;
    cnt_SRRIP+PACManHM    -= 1;
    cnt_BRRIP+PACManHM    += 2;

if(cacheMiss && isFollowerSet[setIndex])
    // Use the policy with the minimum cnt_policy
    int usePolicy = findMinimum();
    Assign RRPV based on usePolicy;
```

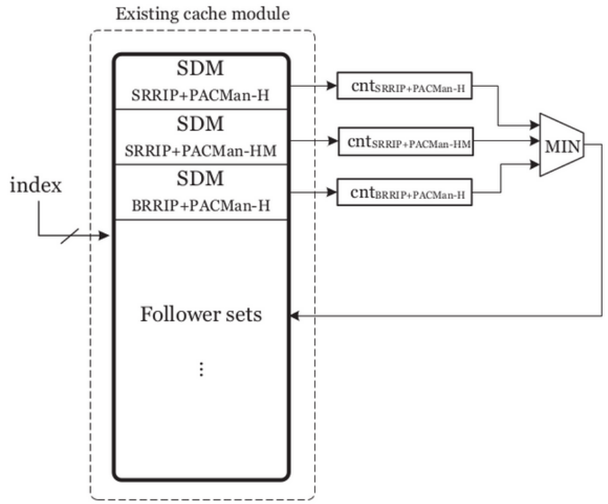
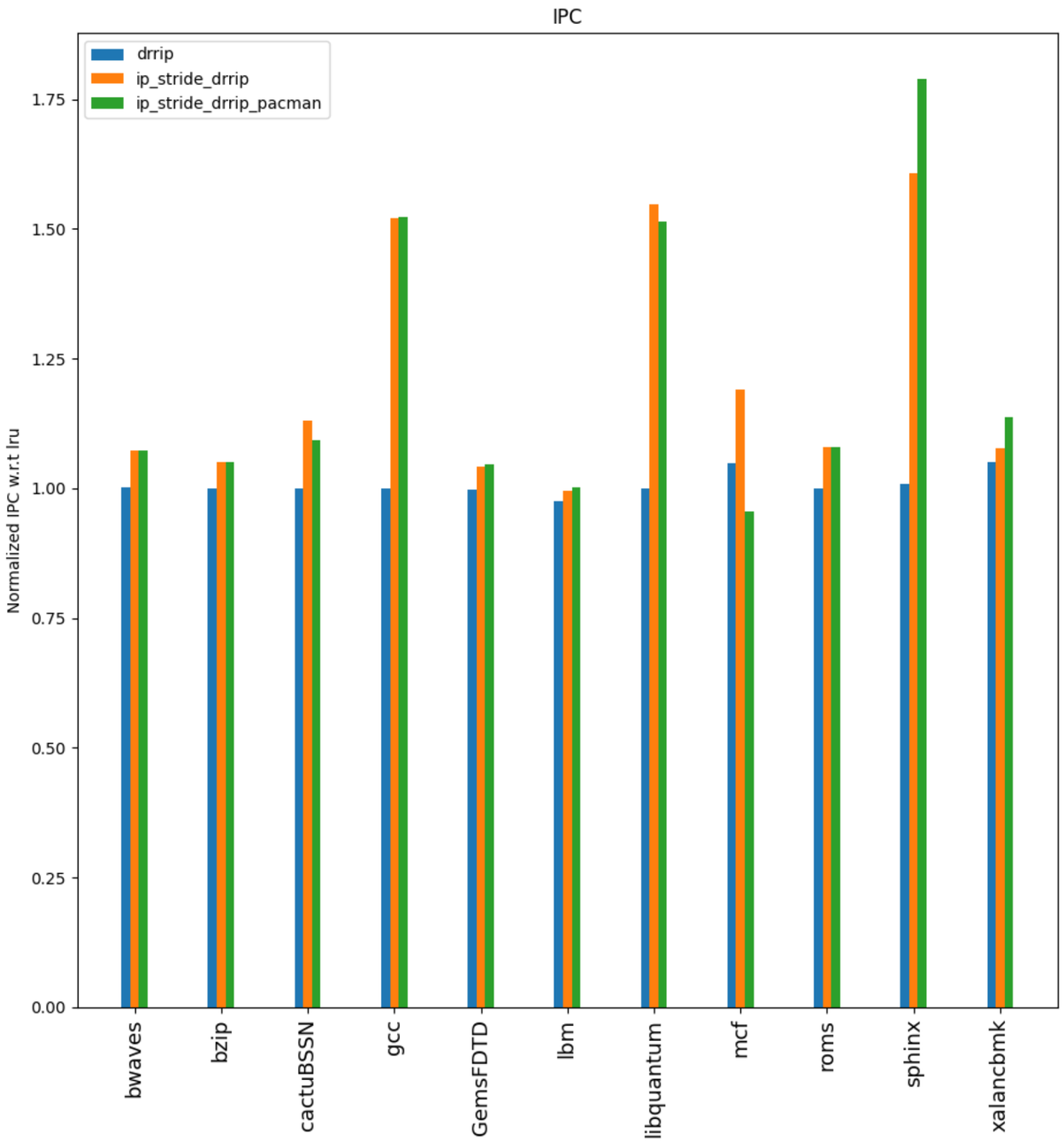


Figure 7: PACMan-DYN Implementation.

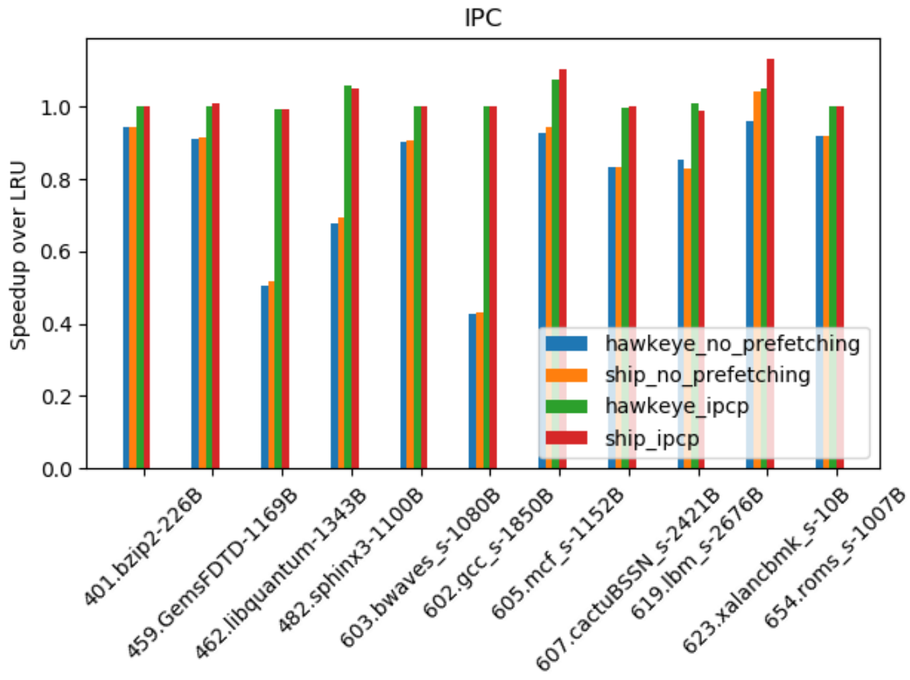
Suggestion 1

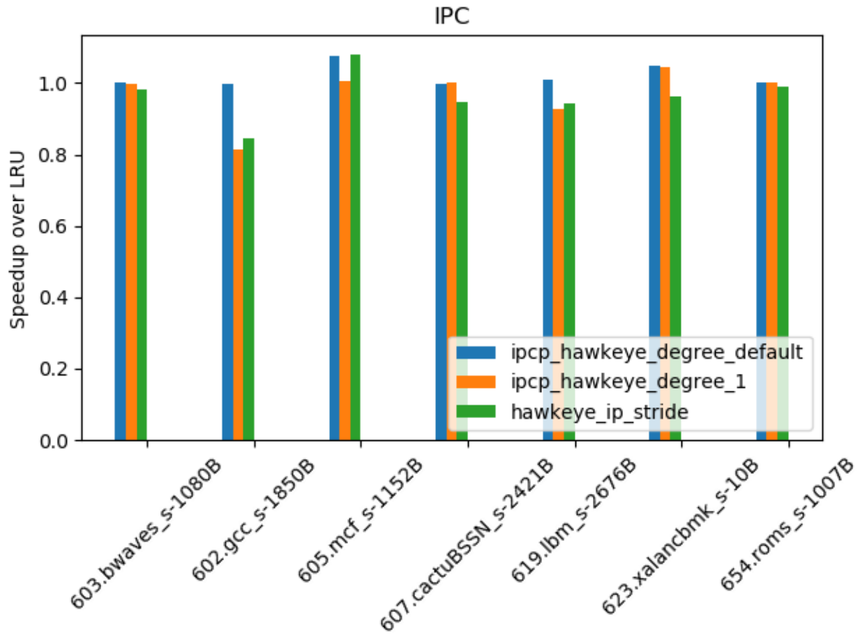
Observed 1.34(approx) speedup after applying pacman compared to 1.24 (approx) without pacman



Suggestion 2

Less agressive prefetcher should result in better performance in hawkeye as per our suggestion.

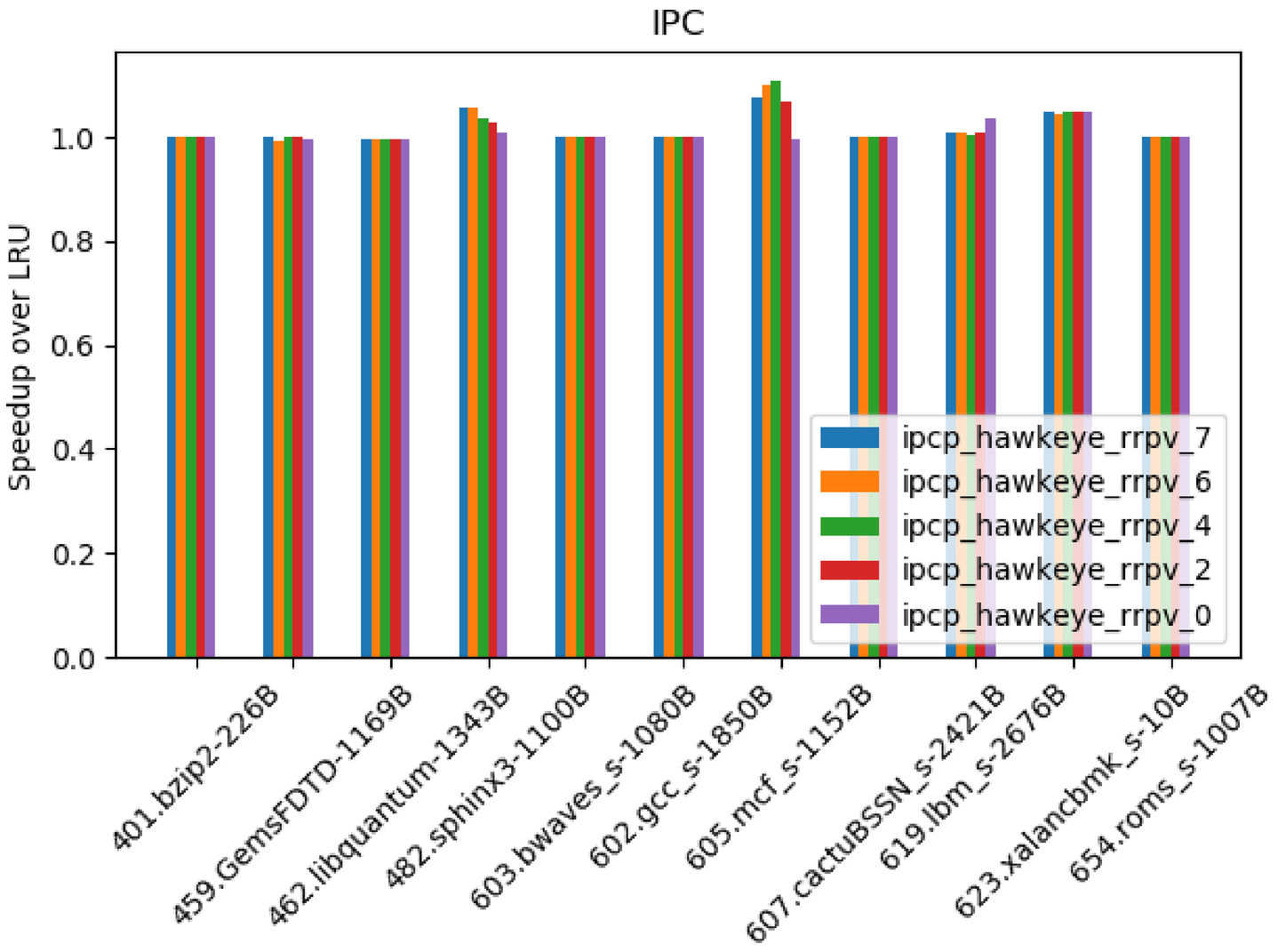




Suggestion 3

NOTE : ipcp_hawkeye_rrpv_x signifies cache averse lines are inserted with rrpv x

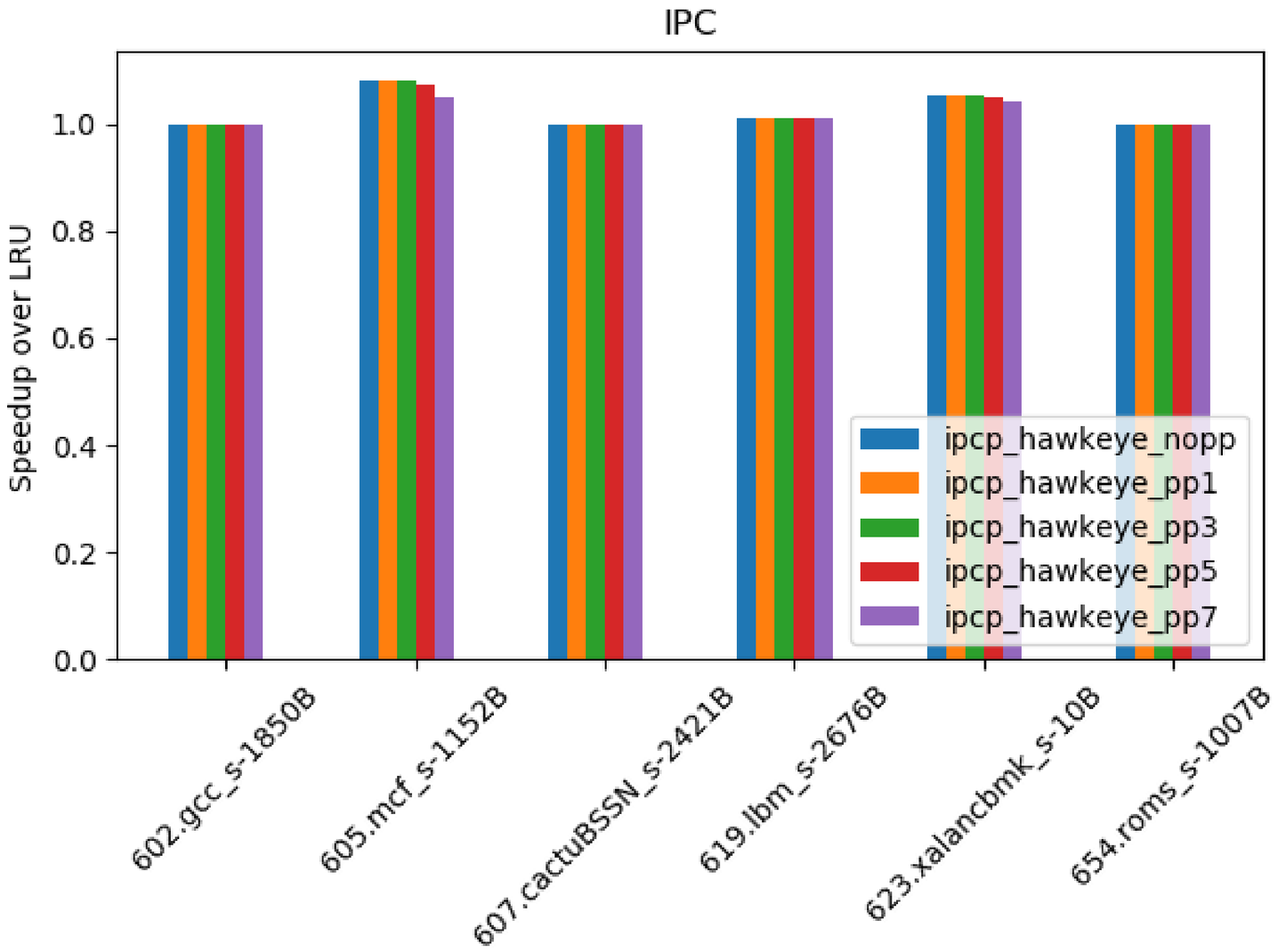
Lower RRPV value resulted in better performance for some of the traces



Suggestion 4

NOTE : ipcp_hawkeye_ppx signifies alpha-P threshold as x*NUMCORE

Did not observe much change in most of the cases, however for some lower threshold resulted in higher IPC

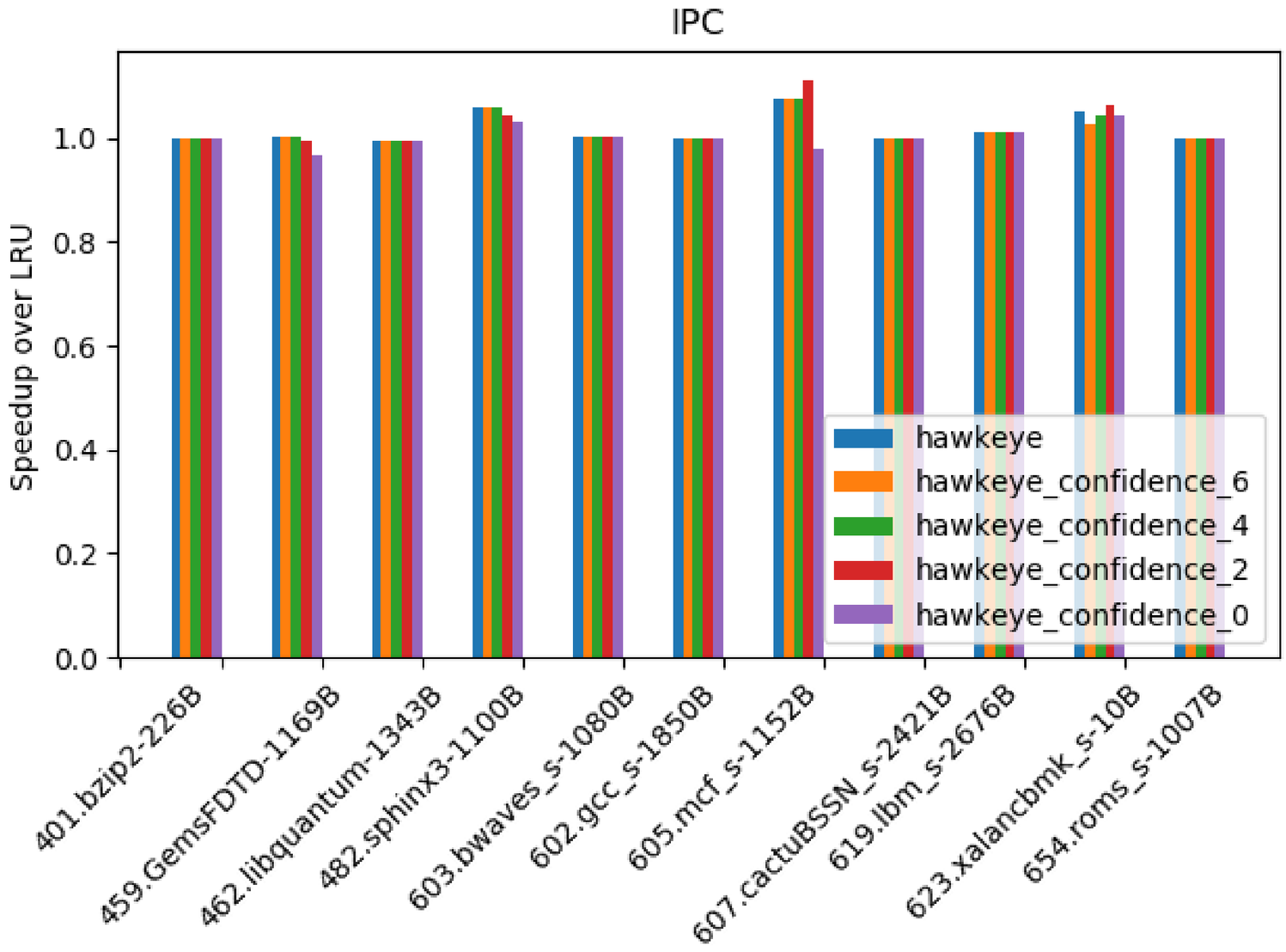


Suggestion 5

```
if(confidence == 0 || confidence == MAX_CONFIDENCE){
    confidence = threshold;
}

if(is_cache){
    confidence ++;
}

else confidence--;

if(confidence >= (MAX_CONFIDENCE+1)/2){
    should be cached
}
else not to be cached
```

Suggestion 5

Did not observe much change in most of the cases, however for some adding confidence resulted in better performance



References :

1. Akanksha Jain, Calvin Lin. Hawkeye: Leveraging Belady’s Algorithm for Improved Cache Replacement

2. Akanksha Jain, Calvin Lin. Back to the Future: Leveraging Belady’s Algorithm for Improved Cache Replacement

3. Samuel Pakalapati, Biswabandan Panda. Bouquet of Instruction Pointers: Instruction Pointer Classifier based Hardware Prefetching

References :

4 Carole-Jean Wu, Aamer Jaleel† Margaret Martonosi, Simon C. Steely Jr., Joel Emer†. PACMan: Prefetch-Aware Cache Management for High Performance Caching

5 Vinson Young, Chia-Chen Chou, Aamer Jaleel, Moinuddin Qureshi. SHiP++: Enhancing Signature-Based Hit Predictor for Improved Cache Performance

Contributions :

Ajay - 190050033	35 % (Contributed to everything +Video Recording + Editing)
Megha - 190050067	30% (Contributed to everything + Video Recording)
Vikas - 190050129	34% (Contributed to everything)
Pawan - 190050080	1 % (Could not contribute due to some family emergency)


Thank You!