

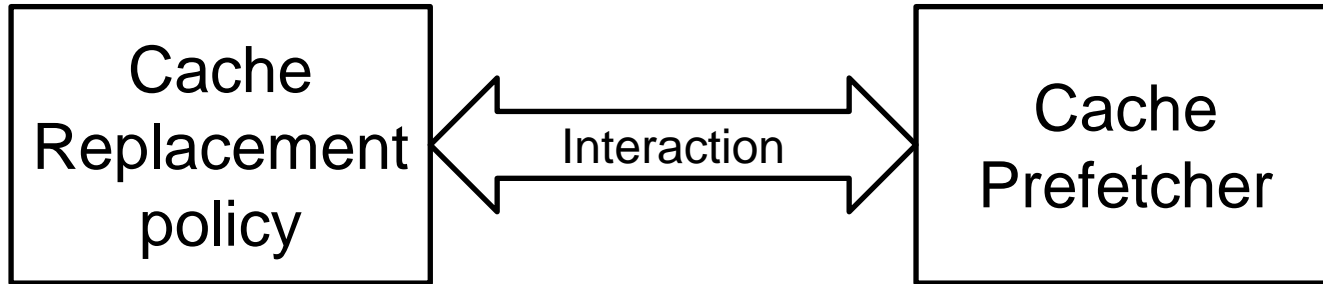
CS-773 Project checkpoint 2

# Microarchitecture interactions

Sumon Nath  
Profetcher(#7)  
sumon@cse.iitb.ac.in

# Background

---



# Previous ideas

---

- Prefetch aware cache management(PACMan<sup>1</sup>)
  - Targets replacement policy only
- Kill the PC(KPC<sup>2</sup>)
  - Not flexible
- Harmony<sup>3</sup>
  - Targets replacement policy only

[1] Jain, Akanksha, and Calvin Lin. "Rethinking belady's algorithm to accommodate prefetching." In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 110-123. IEEE, 2018.

[2] Kim, Jinchun, Elvira Teran, Paul V. Gratz, Daniel A. Jiménez, Seth H. Pugsley, and Chris Wilkerson. "Kill the program counter: Reconstructing program behavior in the processor cache hierarchy." *ACM SIGPLAN Notices* 52, no. 4 (2017): 737-749.

[3] Wu, Carole-Jean, Aamer Jaleel, Margaret Martonosi, Simon C. Steely Jr, and Joel Emer. "PACMan: prefetch-aware cache management for high performance caching." In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 442-453. 2011.

# Proposed idea

---

- Fine grained approach: identify interactions
- Dynamically tune replacement policy & prefetcher
- Across the cache hierarchy
- Flexible: works with any prefetcher & replacement policy

# Interactions- cache eviction

---

- Positive interaction

Useful block evicts useless block

- Negative interaction

Useless block evicts useful block

\*Useless prefetch block evicts useless block

- Neutral interaction

other cases

# Ways to identify interactions

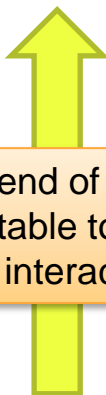
---

- Predictor based ~ not precise
- Future access based ~ precise

# Identification of interactions

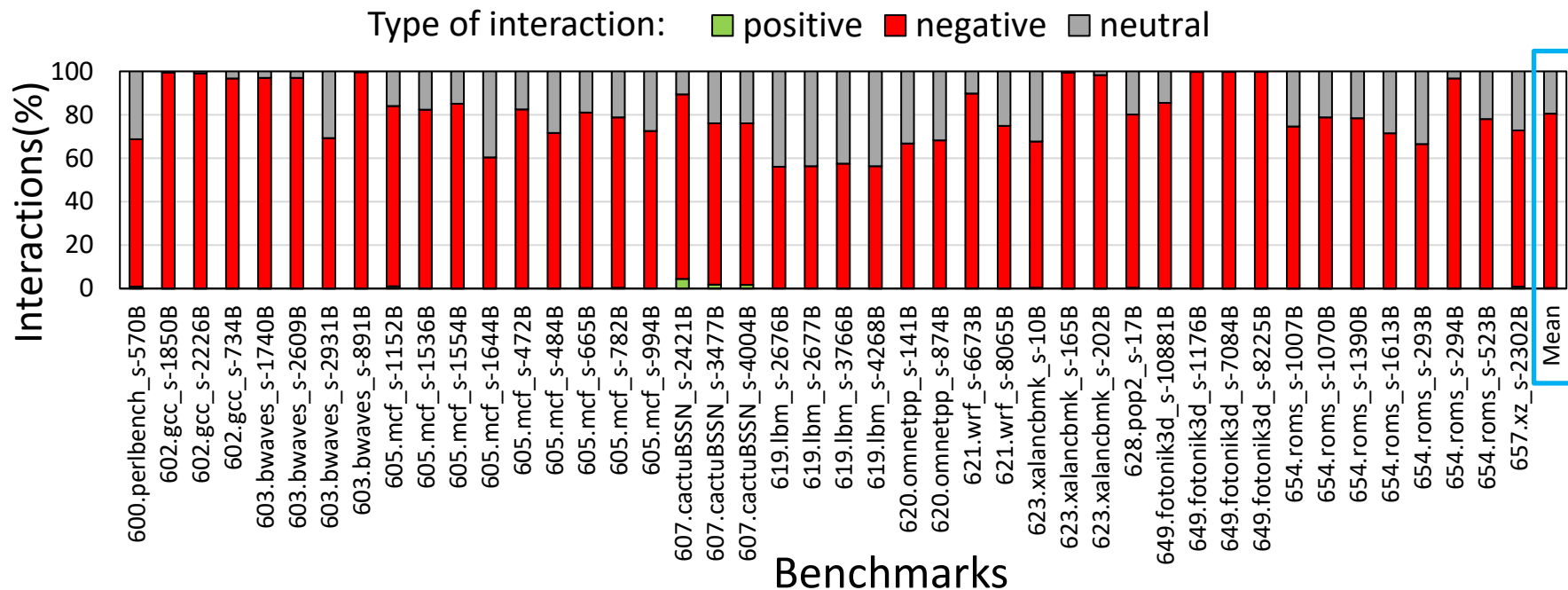
---

- Table which stores the interaction as well as all cache accesses.

- 
- At the end of epoch
  - Parse table to record interaction

Is_eviction	Line_1	Type_1	Line_2	Type
1	A(evicts)	P	X(evicted)	C
0	X(access)			
0	X(access)			
1	Y(evicts)	C	Z(evicted)	P

# Interactions distribution

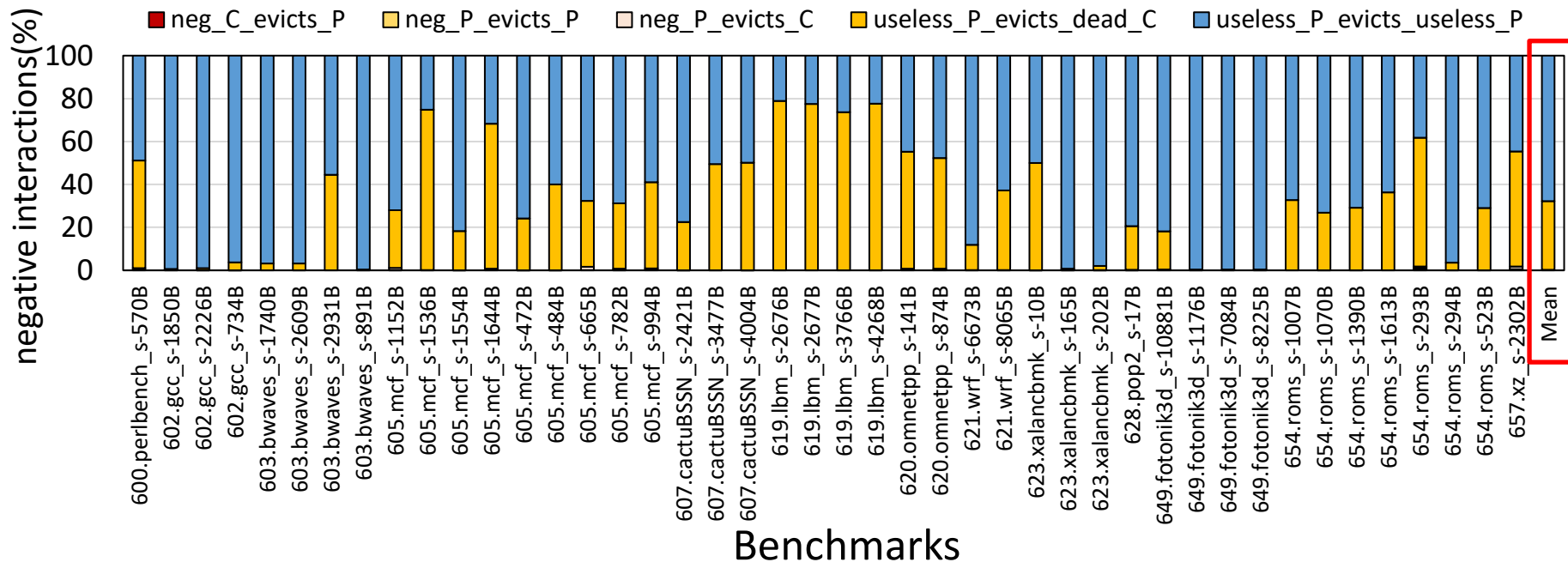


- Prefetcher used at L1, L2 : **IPCP**
- Replacement policy used at LLC: **LRU**

- Negative interactions: ~80%



# Breakdown of negative interactions



- 98% : Useless Prefetch blocks evicting useless blocks
- **Inaccurate prefetches**

# Checkpoint - 1

# Ideal performance improvement

---

- What if all negative interactions are eliminated?
- Upper bound performance improvement

# Methodology

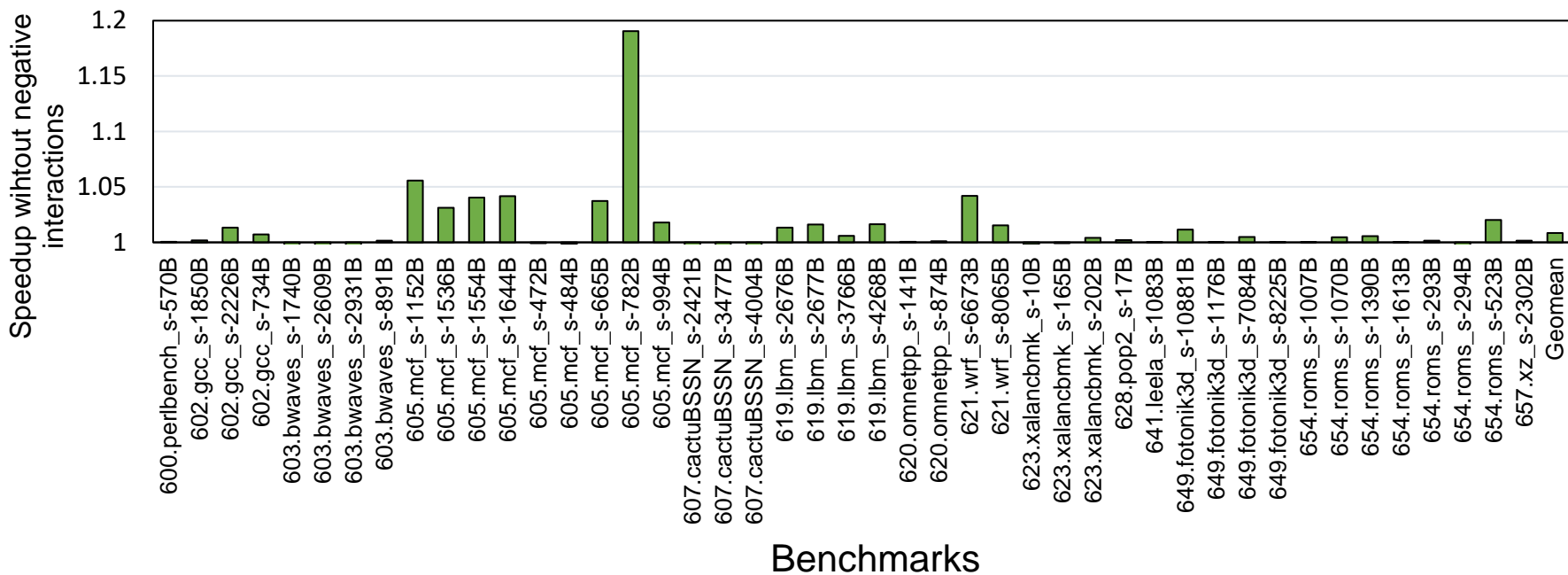
---

- *Additional penalty due to the negative interactions per LLC miss ( $P$ )*

$$= \frac{\text{avg miss latency with prefetcher} - \text{avg miss latency without prefetcher}}{\text{avg MSHR occupancy}}$$

- *Fraction of llc misses suffer from penalty ( $F$ )*  
 $= \text{number of llc misses} \times \text{frac. Negative interactions}$
- $\text{cpu cycles}(\text{no neg.}) = \text{cpu cycles}(\text{with neg.}) - F \times P$

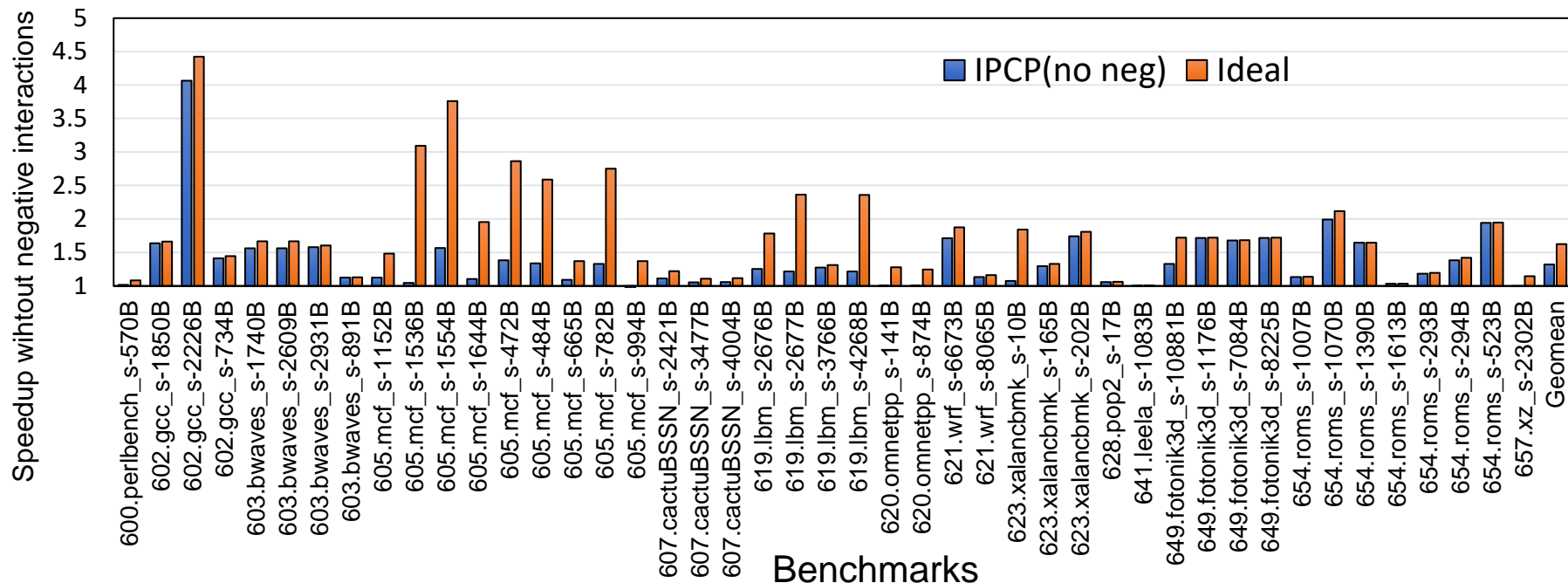
# Performance improvement



Baseline: IPCP prefetcher

**0.8%** speedup with no negative interaction at LLC

# Performance improvement



- Baseline: No prefetcher

- IPCP: 30%
- IPCP(no neg) : 31%
- Ideal L1D : 62%

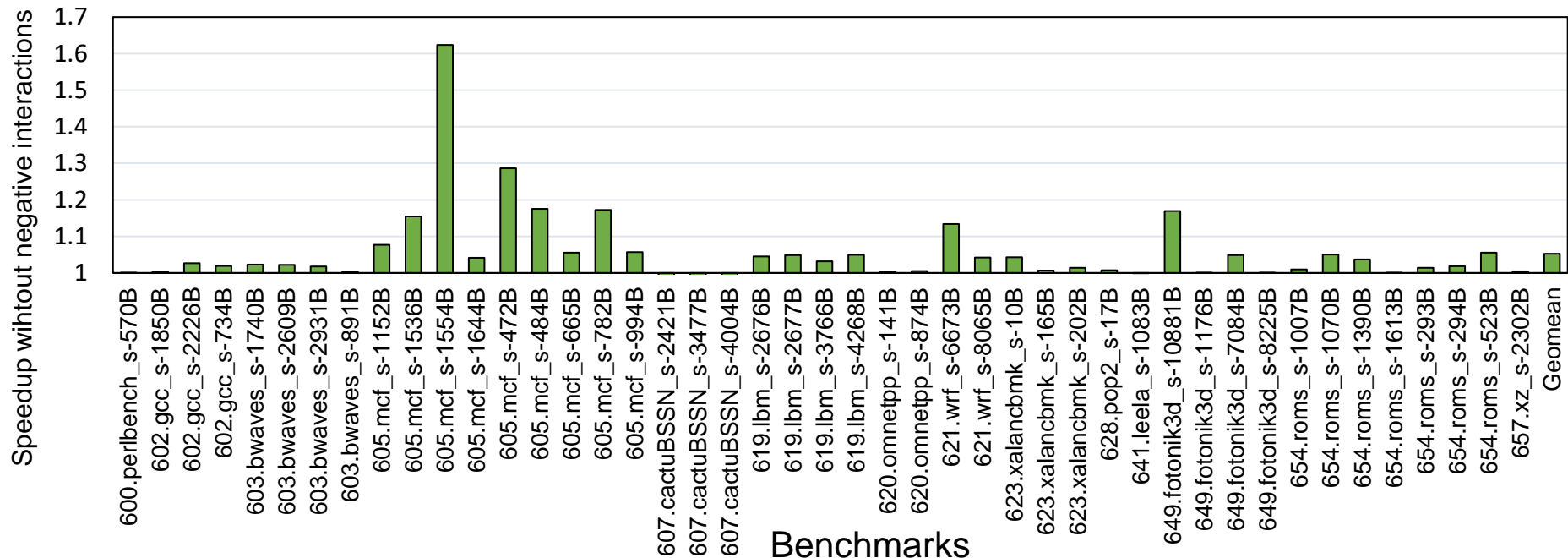
# Low DRAM bandwidth

---

Servers – Multicore system

- **32** cores
  - DRAM BW per channel – **6400 MT/s**
  - **4** available channels
  - 1 channel for 8 cores
  - On average per core DRAM BW
- **800 MT/s**

# Performance improvement(low DRAM BW)

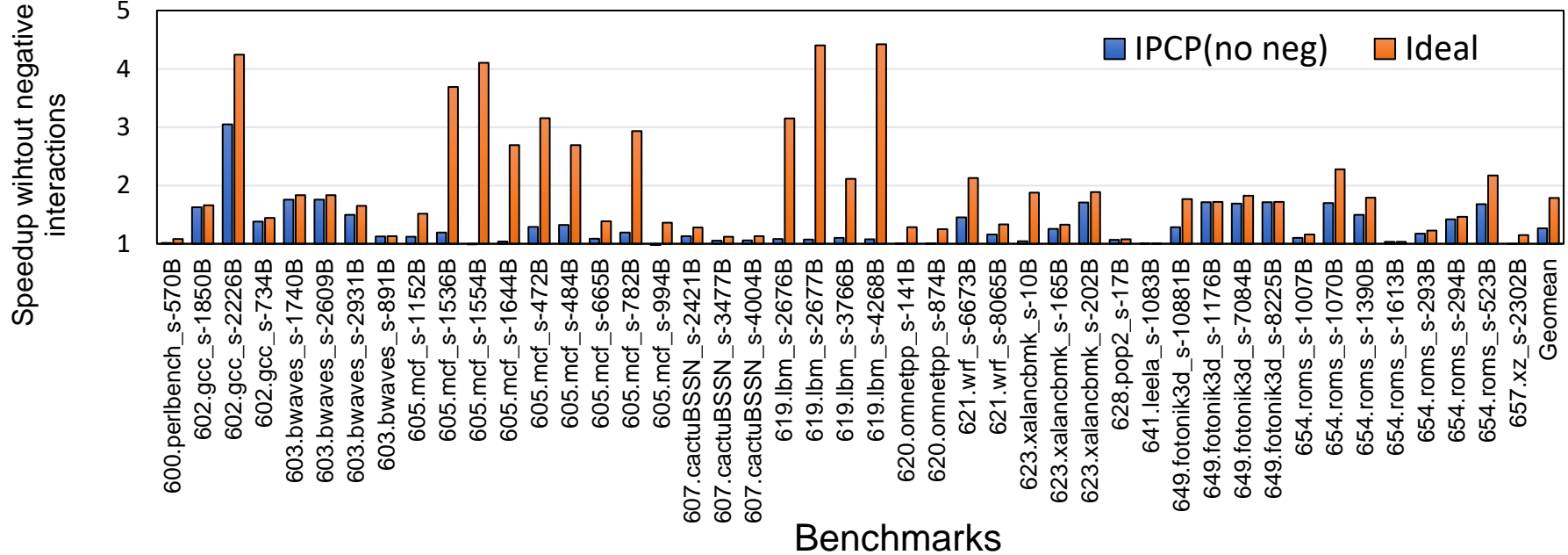


- Baseline: IPCP prefetcher

**5.2%** speedup with no negative interaction at LLC



# Performance improvement(low DRAM BW)



- Baseline: No prefetcher

- IPCP(with neg): 20%
- IPCP(no neg) : 26%
- Ideal L1D: 78%

# Checkpoint - 2

# What next?

---

- Run **experiments** with multiple prefetcher and replacement policy combination
- **Practical solution** with performance improvement close to ideal improvement

Thank You