TEXAS
The University of Texas at Austin

Hawkeye: Leveraging Belady's Algorithm for Improved Cache Replacement

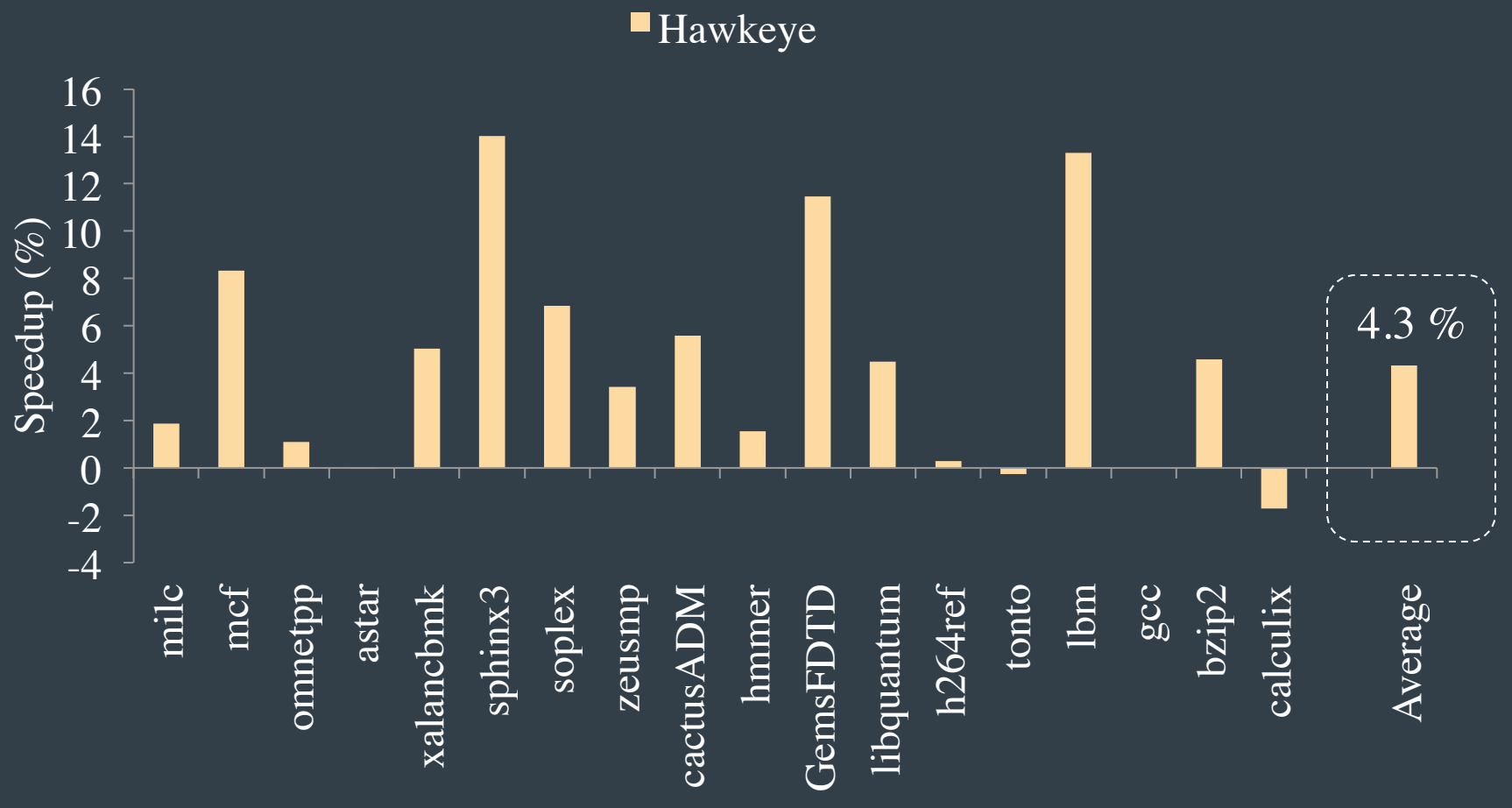
Akanksha Jain
Calvin Lin

June 25, 2017

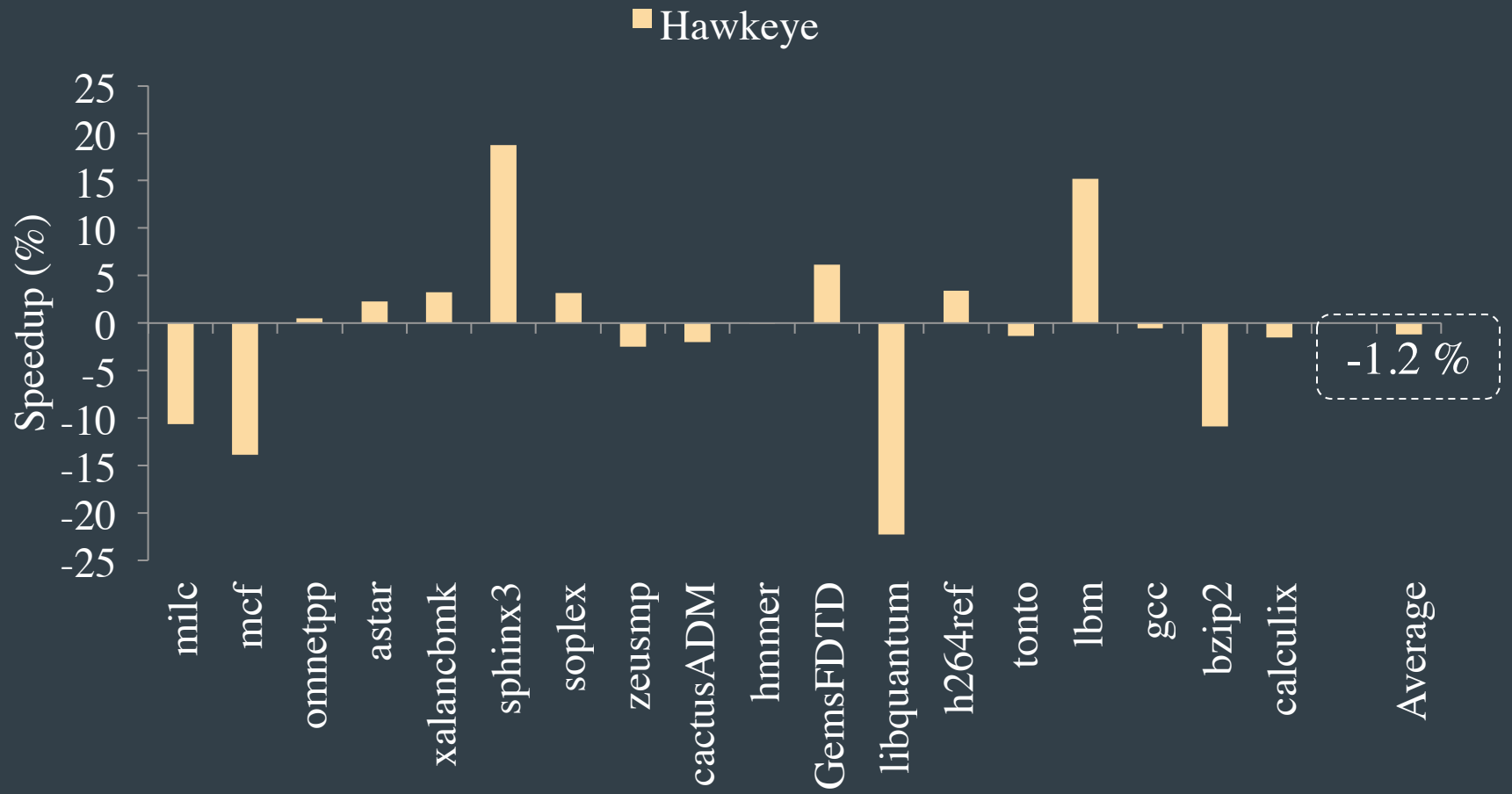
Our Strategy

- Based on the Hawkeye Cache [ISCA '16]

Performance Improvement (Config 1)



Performance Improvement With Prefetcher (Config 2)



Today's Talk

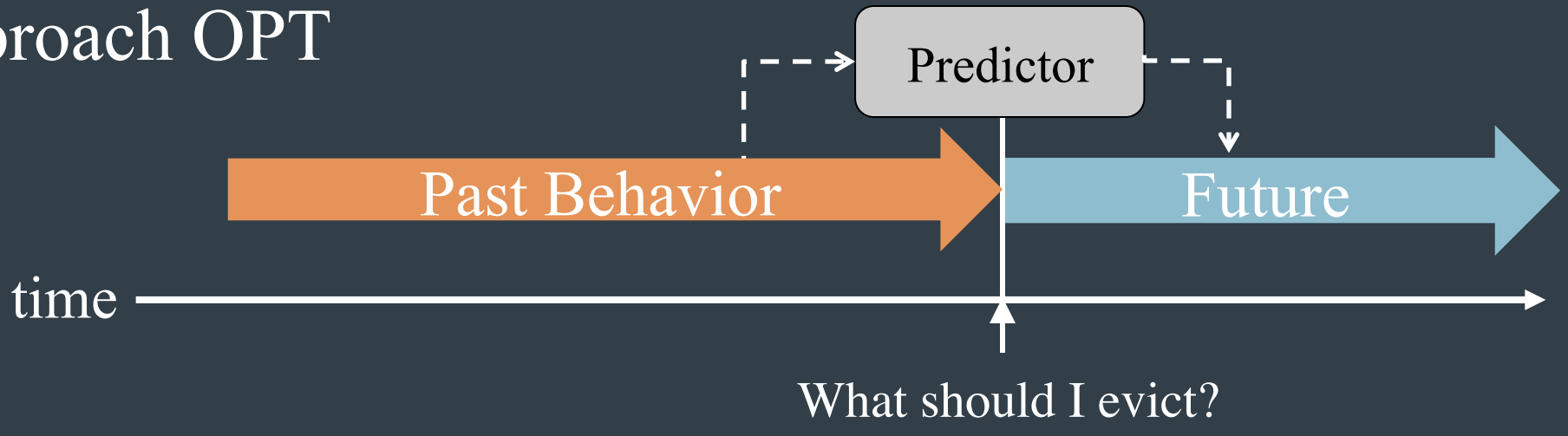
- Understand why Hawkeye does poorly with prefetches
- Suggest solutions to address these issues

The Hawkeye Cache

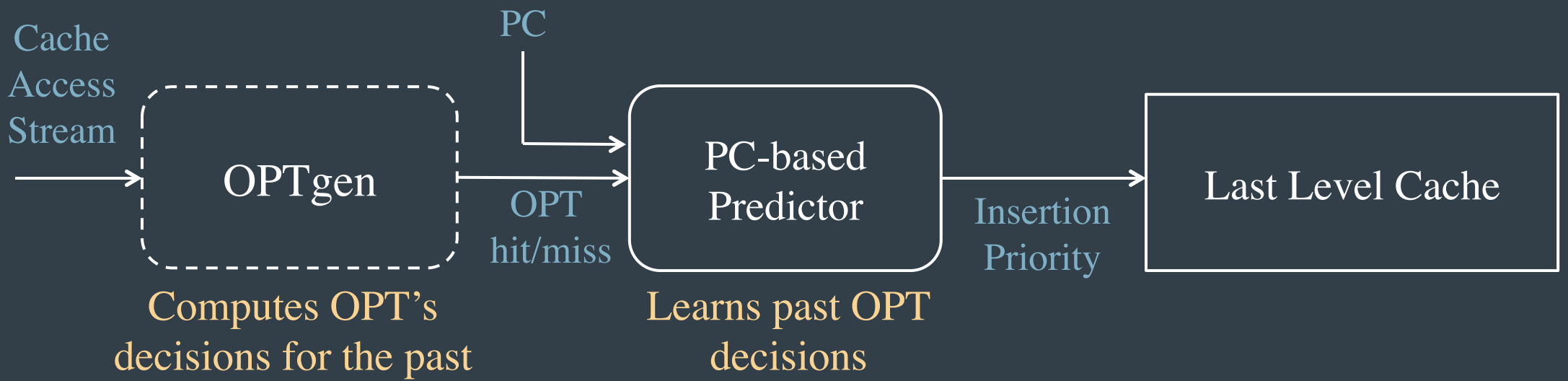
- Based on Belady's optimal solution
  - Evict the line that is re-used furthest in the future
  - Provably optimal, but requires future knowledge

The Hawkeye Cache

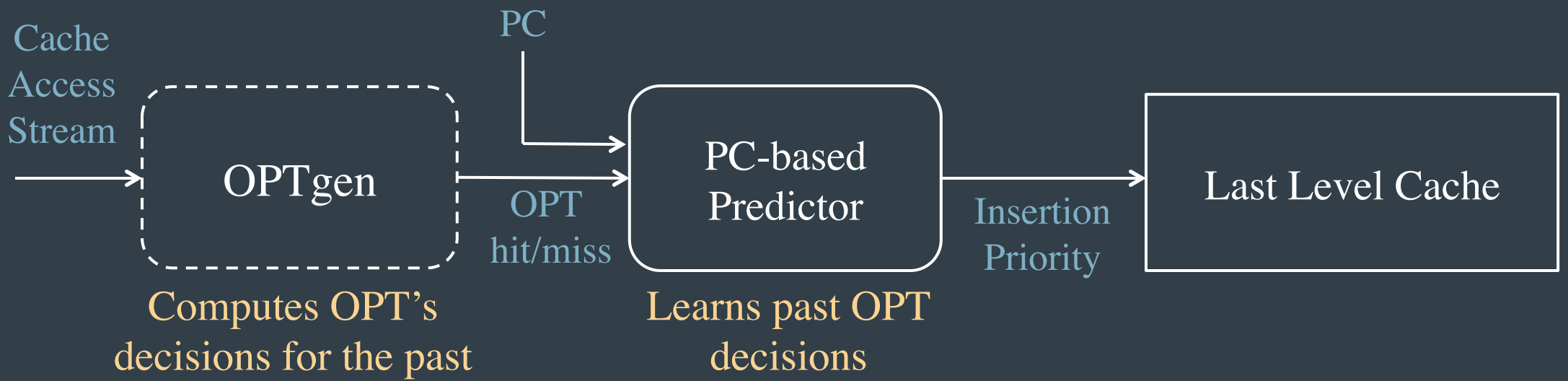
- We cannot look into the future
- But we can apply the OPT algorithm to past events to learn how OPT behaves
- If past behavior predicts the future, then this solution should approach OPT



The Hawkeye Cache



The Hawkeye Cache



How should OPTgen deal with prefetches?

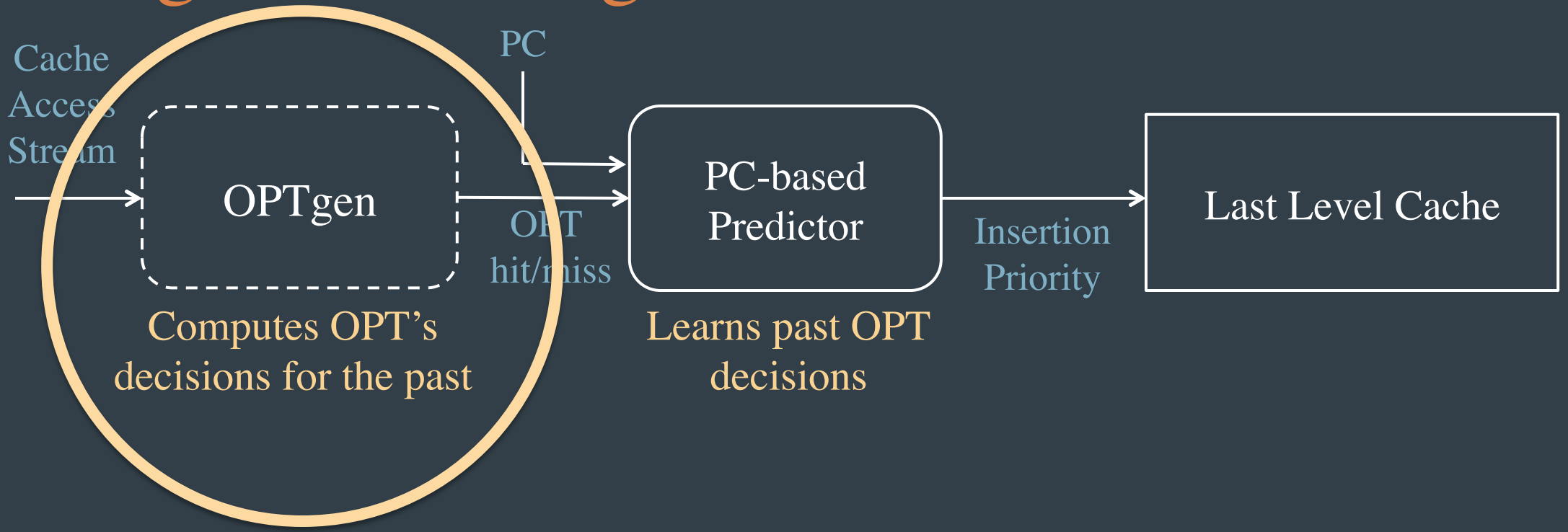
How should the predictor deal with prefetches?

Prefetch-Aware Hawkeye

- How should OPTgen deal with prefetches?
  - Key challenge: redundant prefetches and inaccurate prefetches

- How should the predictor deal with prefetches?
  - Separate predictors for demand and prefetches

Background: OPTgen



Background: OPTgen

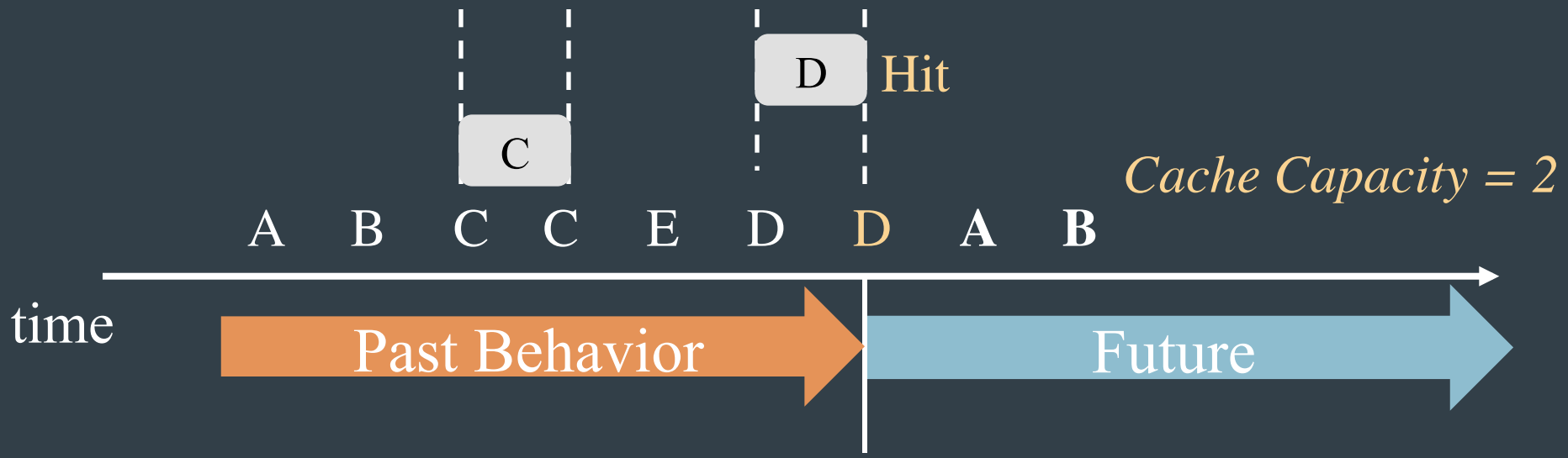
- Linear time algorithm that reproduces OPT's solution for the past
  - 100% accuracy with no resource constraints (assuming no prefetches)
  - 95.5% accurate with sampling (assuming no prefetches)
- Each line's demand on the cache is represented by a liveness interval
- Lines are allocated cache capacity in the order that they are reused

Background: OPTgen

- For each re-use, would this line have been a hit or miss with OPT?
- Is there room for C?

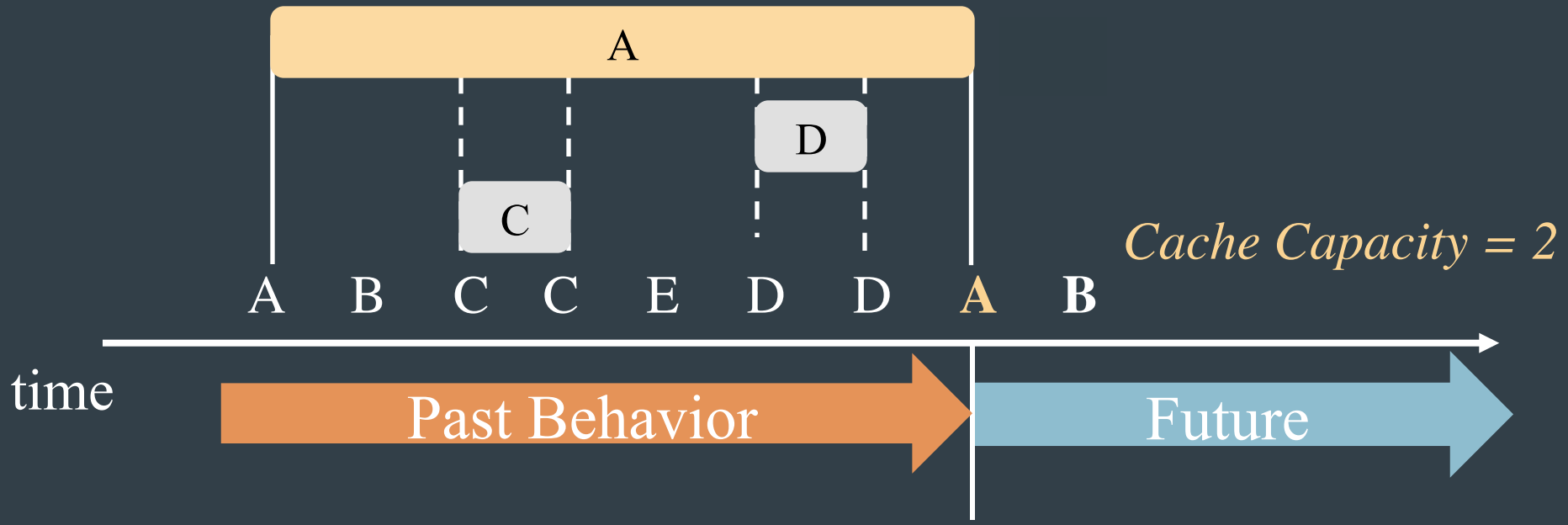
Background: OPTgen

- For each re-use, would this line have been a hit or miss with OPT?
- Is there room for D?



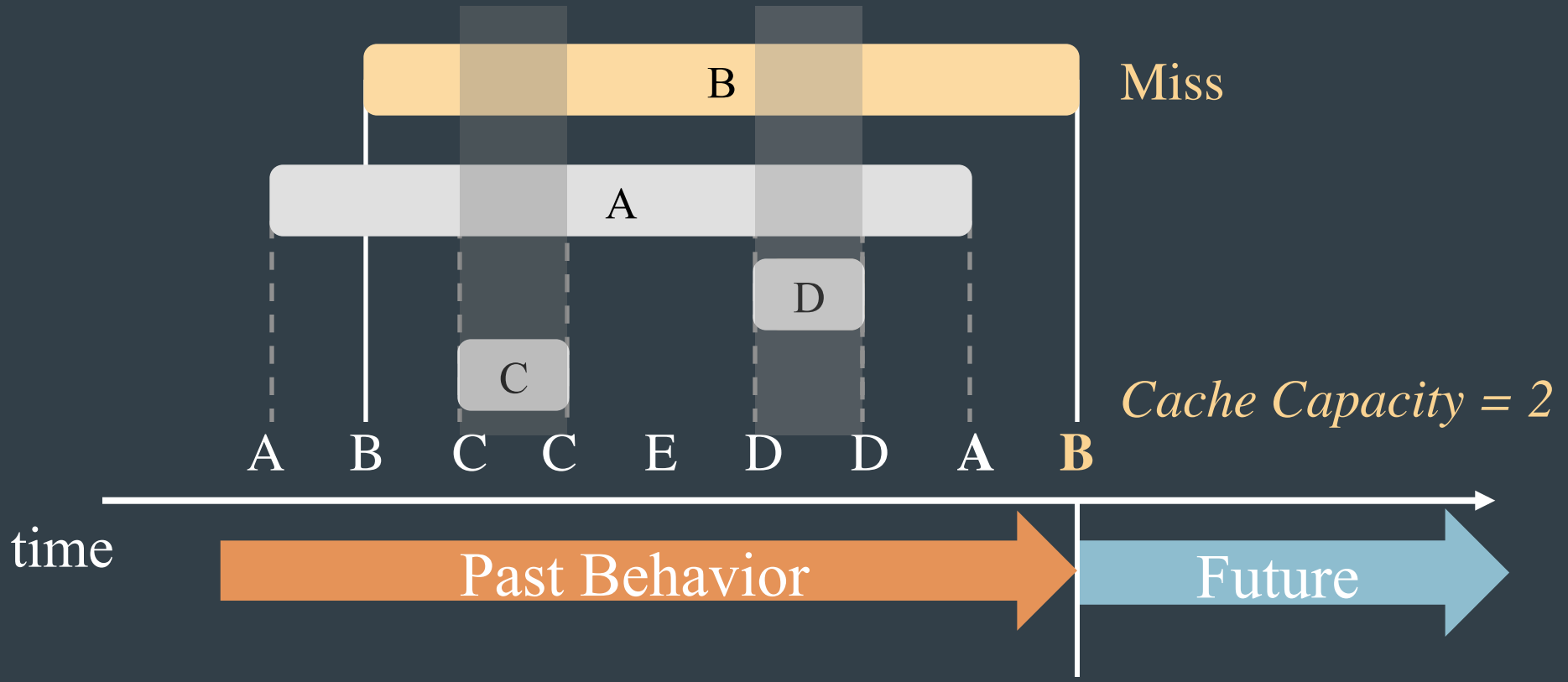
Background: OPTgen

- For each re-use, would this line have been a hit or miss with OPT?
- Is there room for A?



Background: OPTgen

- For each re-use, would this line have been a hit or miss with OPT?

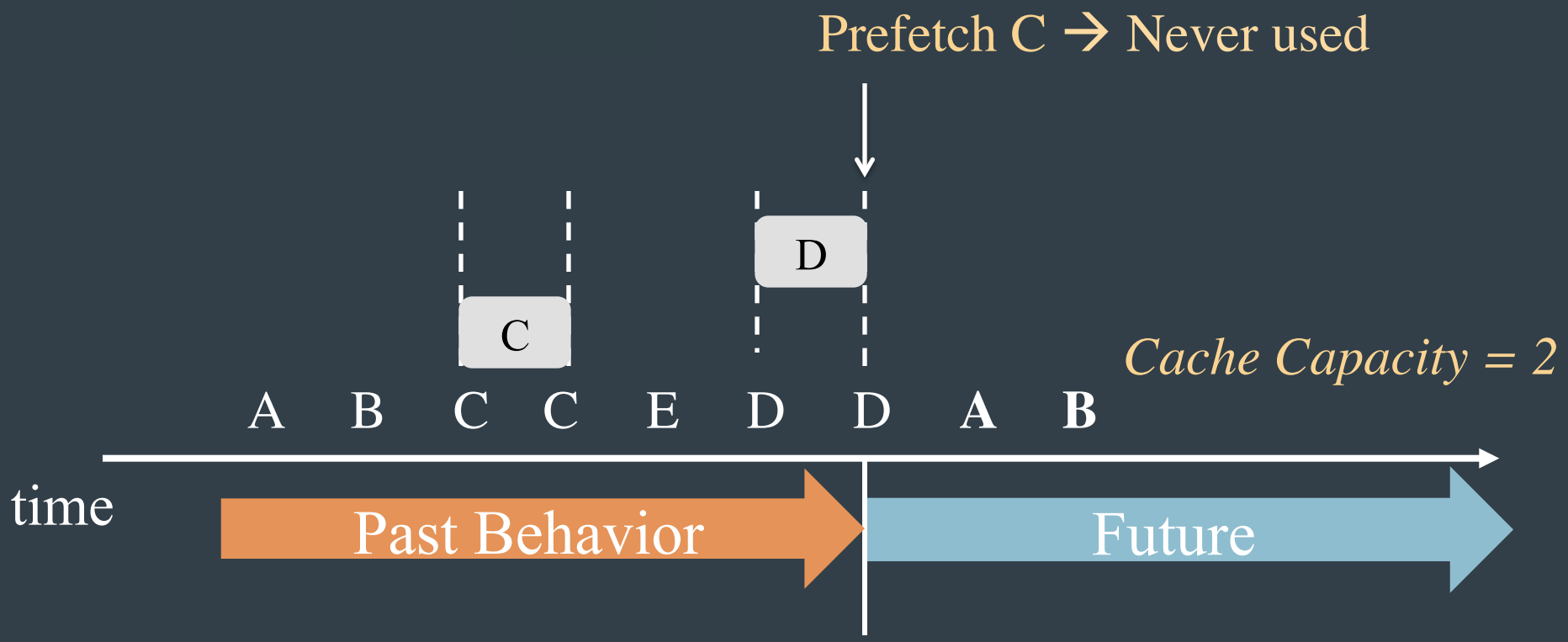


Liveness Intervals

- What defines liveness intervals?
- In the absence of prefetching, endpoints are demand accesses
  - D-D intervals (reuse of demand accesses)
- In the presence of prefetching, four kinds of intervals possible
  - D-D, D-P, P-D, P-P intervals
  - Original solution did not distinguish among these

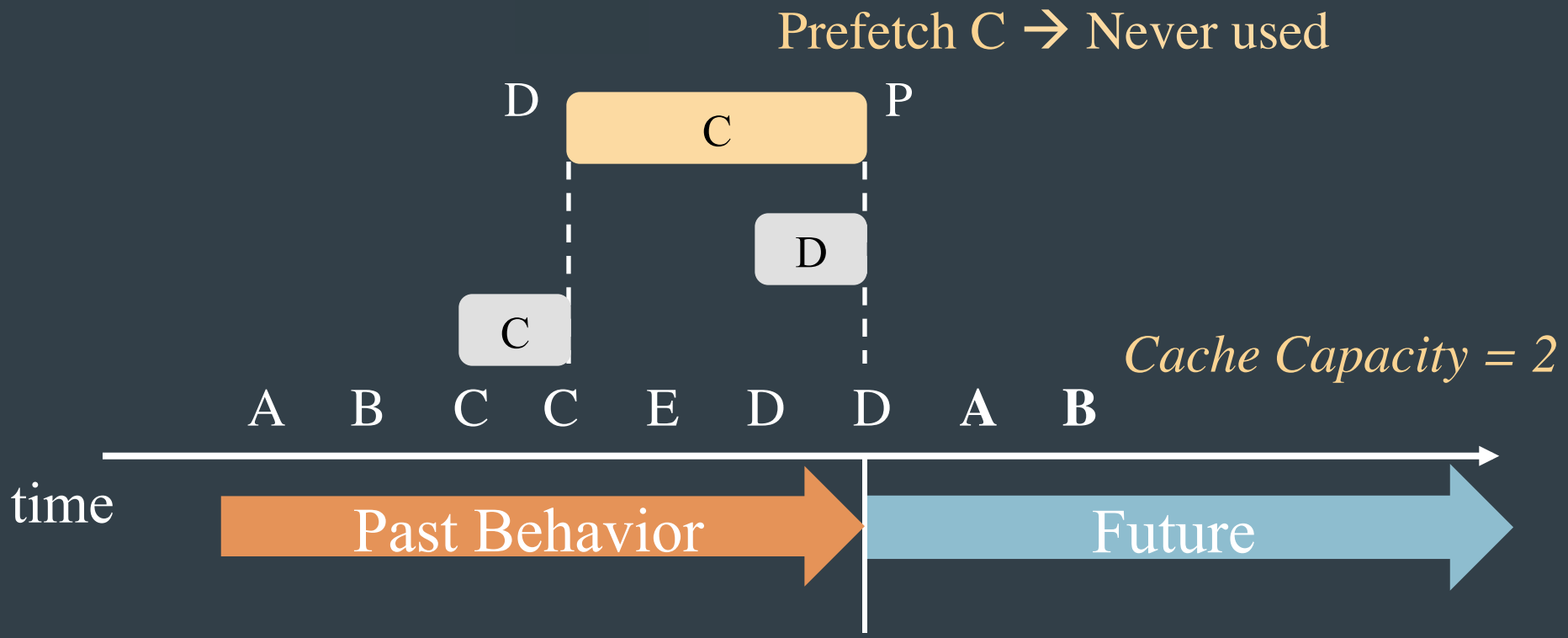
OPTgen With Prefetches

- What if we had an inaccurate prefetch to C?



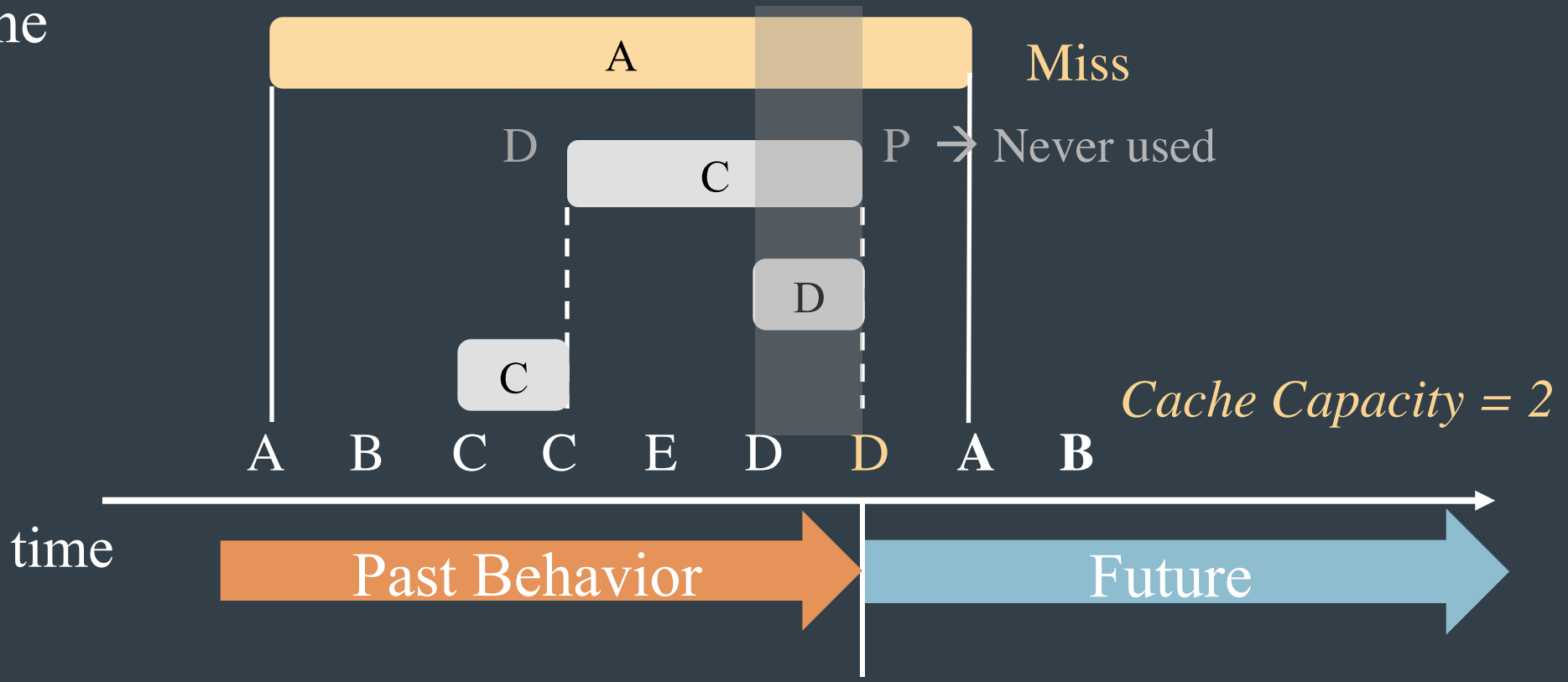
OPTgen With Prefetches

- What if we had an inaccurate prefetch to C?



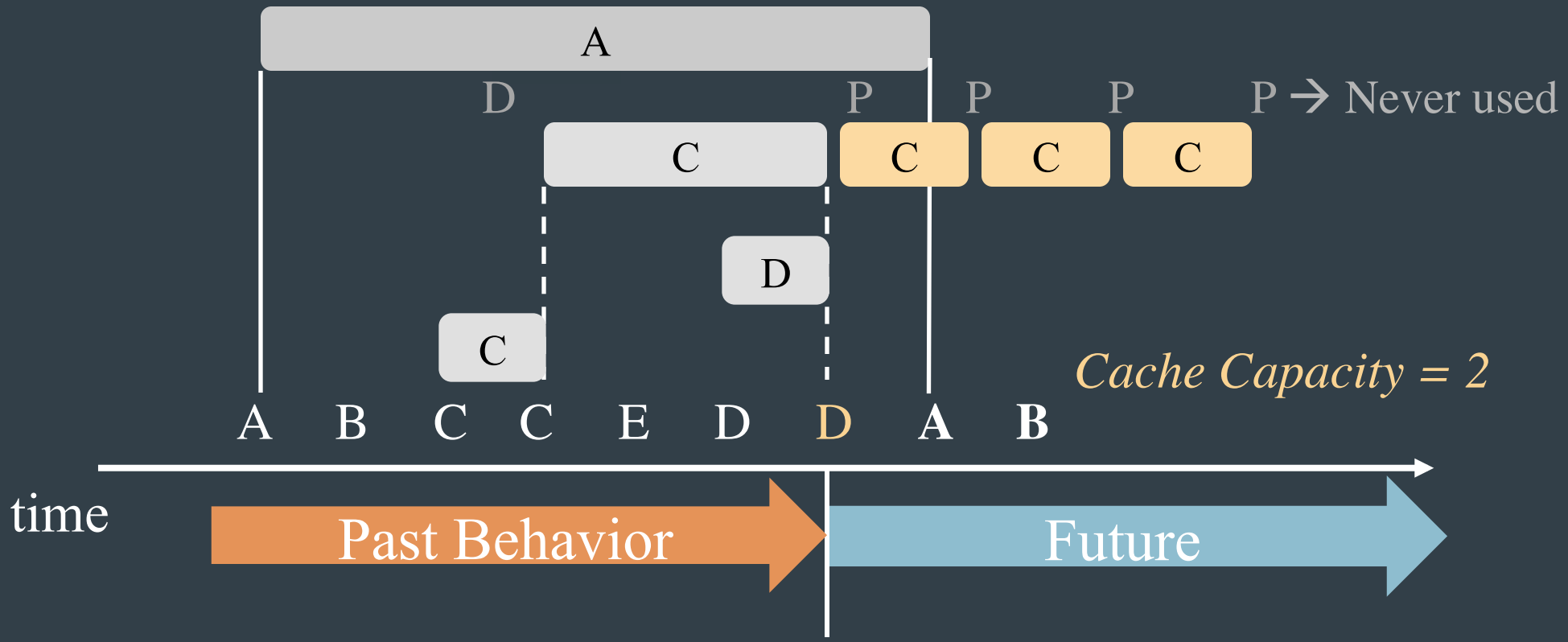
OPTgen With Prefetches

- D-P intervals are undesirable because they create unwanted demand on the cache



OPTgen With Prefetches

- P-P intervals also create unwanted demand on the cache without resulting in hits



OPTgen With Prefetches

- One Solution: Only consider liveness intervals that end with demand accesses
  - D-D intervals (demand reuse)
  - P-D intervals (useful prefetch)

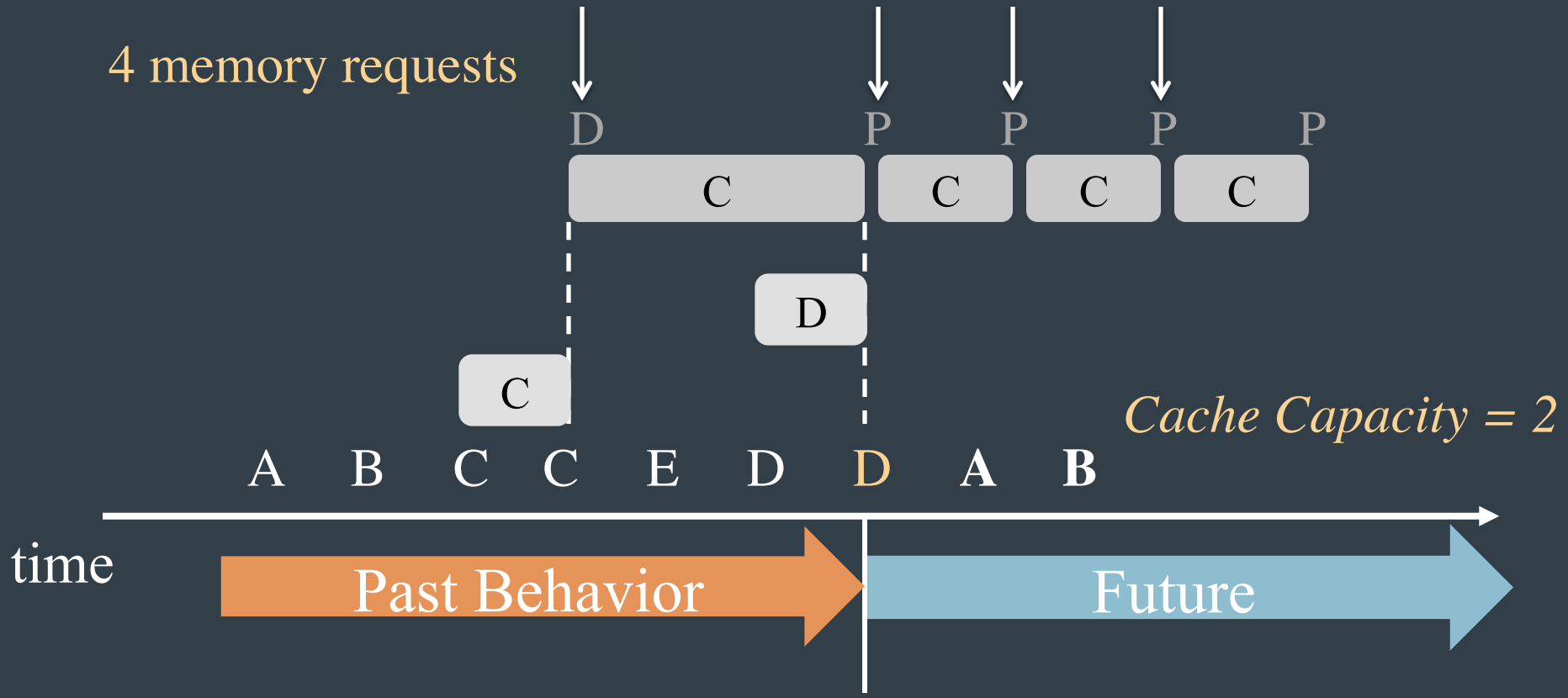
- Ignore liveness intervals that end with a prefetch
  - D-P intervals (redundant prefetch, potentially inaccurate)
  - P-P intervals (redundant prefetch, potentially inaccurate)

Complication

- Ignoring redundant prefetches (*-P intervals) maximizes cache efficiency at the expense of memory traffic

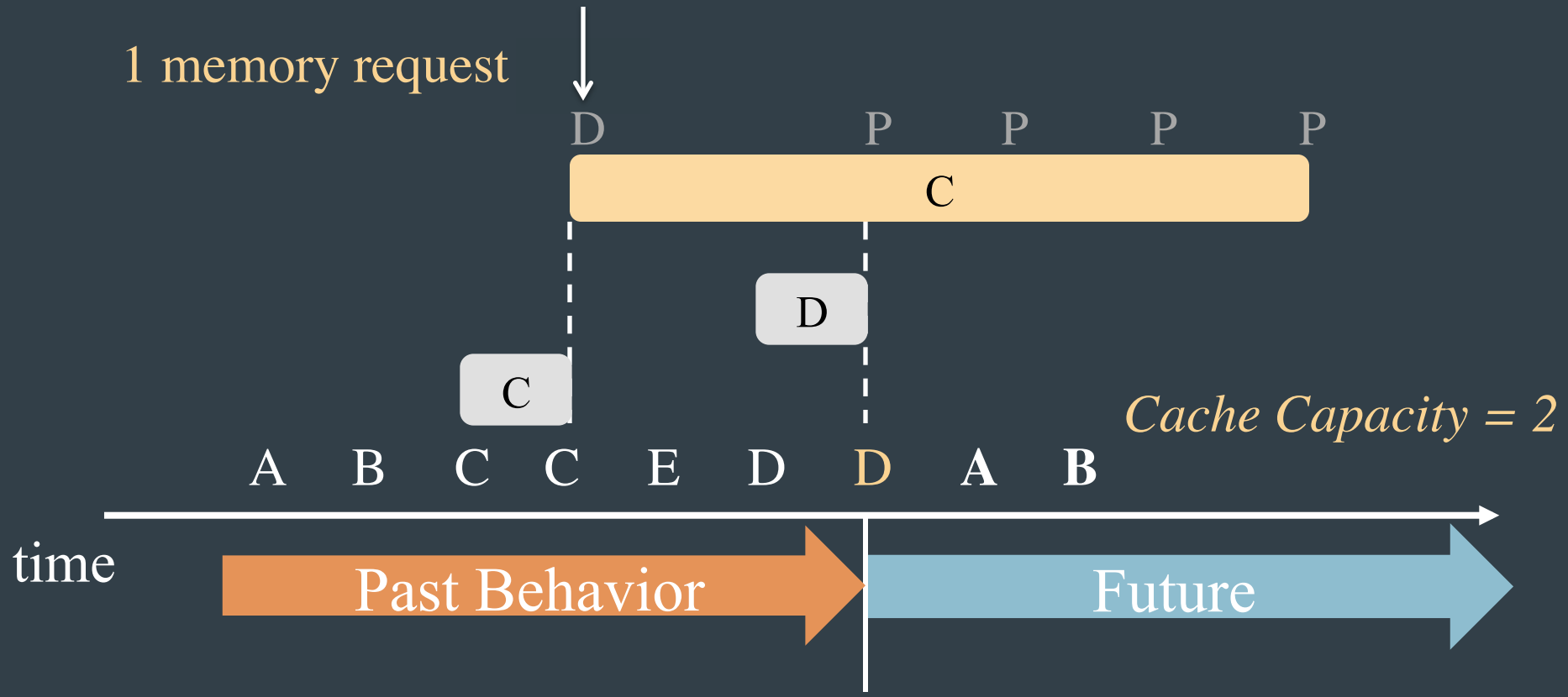
Memory Traffic Overhead

- Ignoring *-P intervals generates additional memory traffic



Memory Traffic Overhead

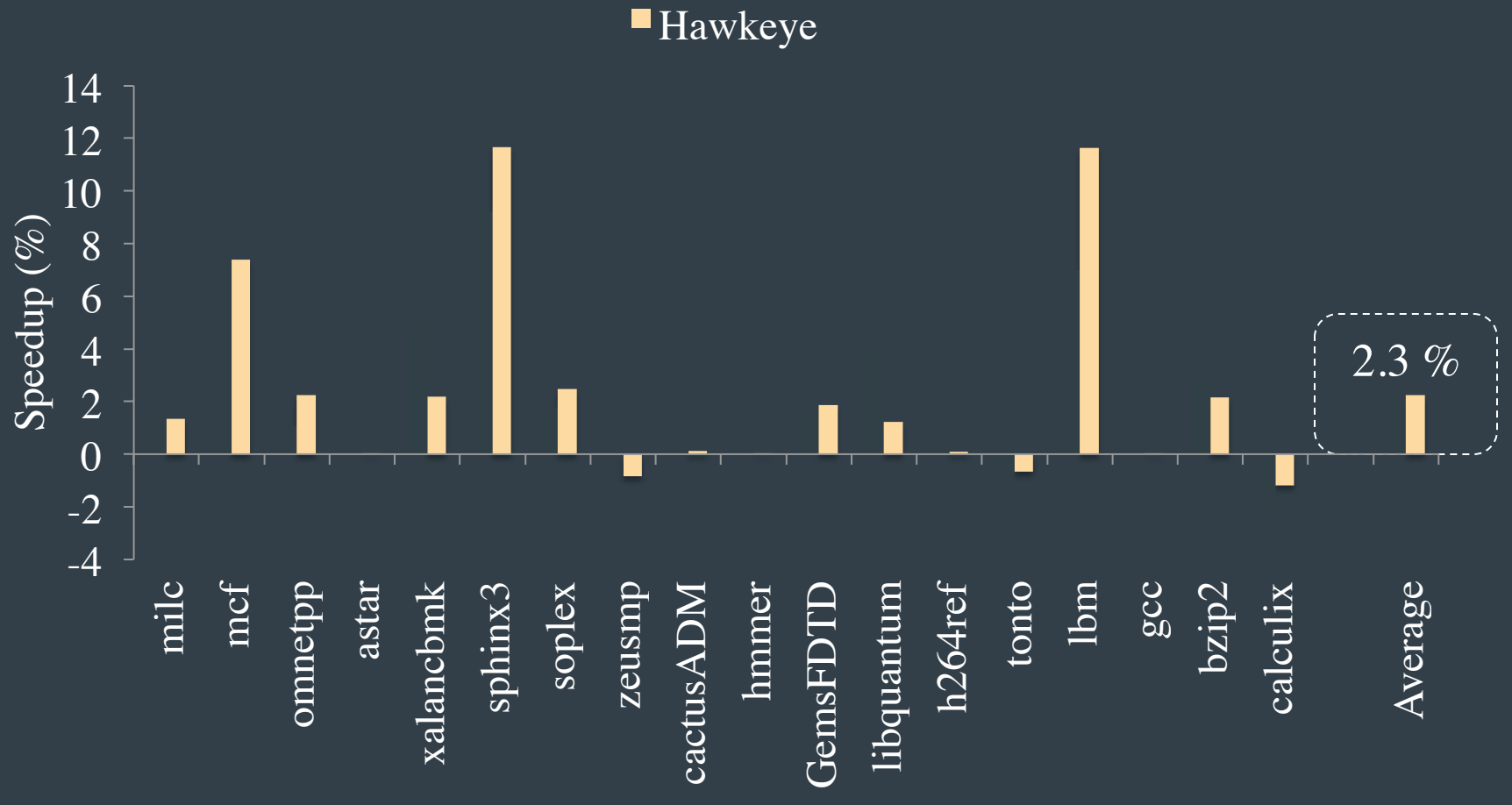
- Caching *-P intervals reduces memory traffic



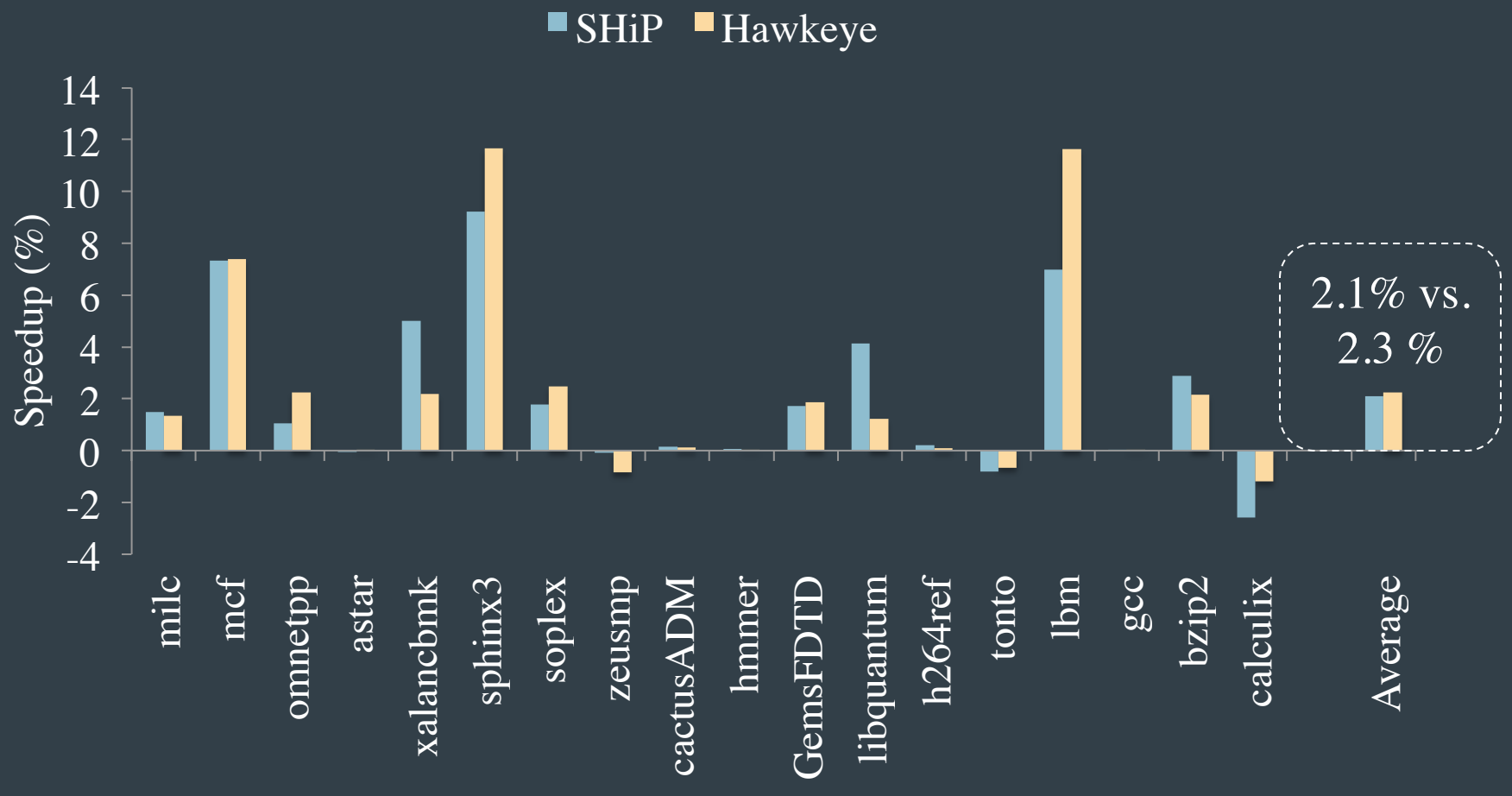
Complication

- Ignoring redundant intervals maximizes cache efficiency at the expense of memory traffic
- Trade-off between cache efficiency and memory traffic
- Our Solution: Ignore long *-P intervals, but cache short *-P intervals

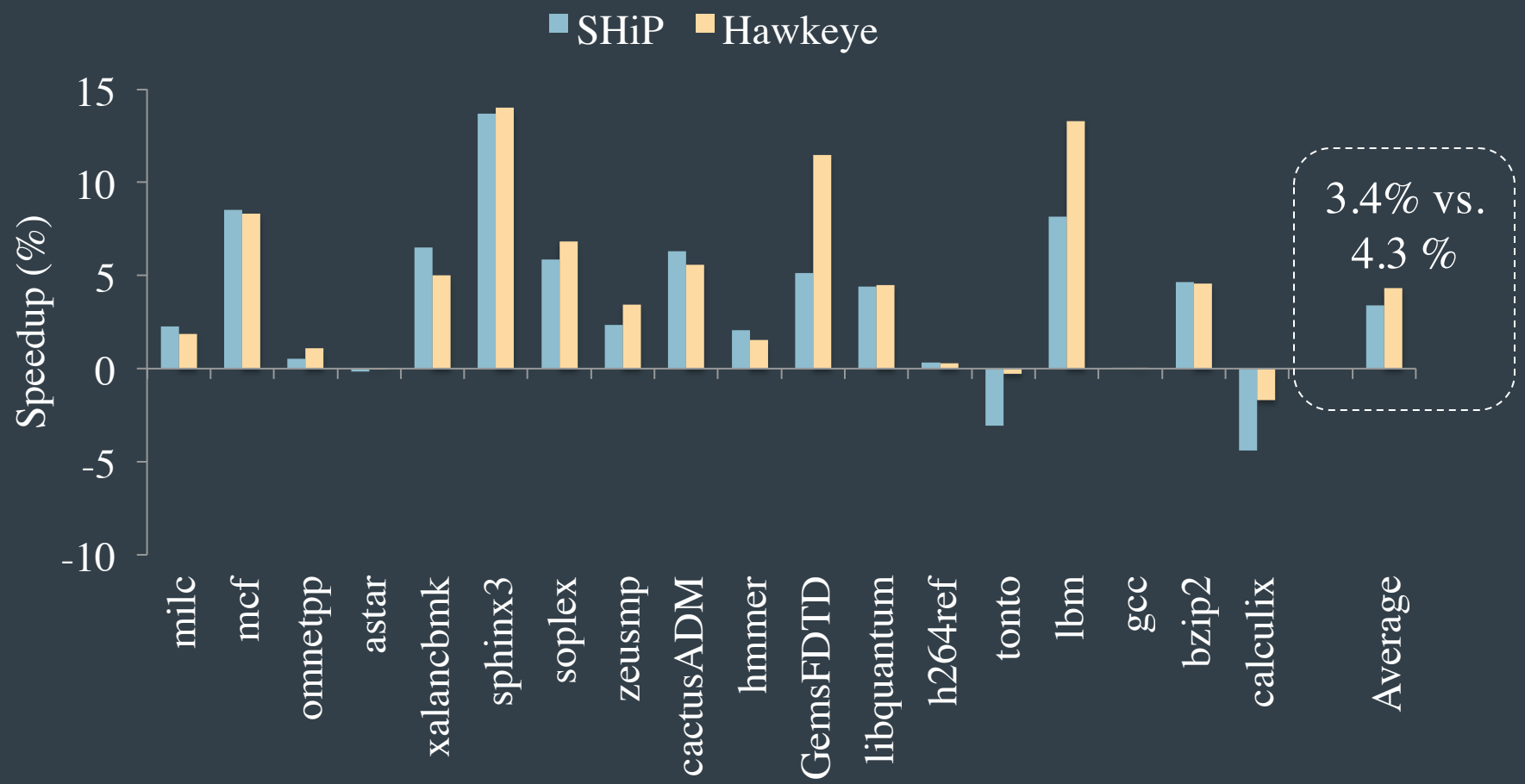
Performance Improvement With Prefetcher (Config2)



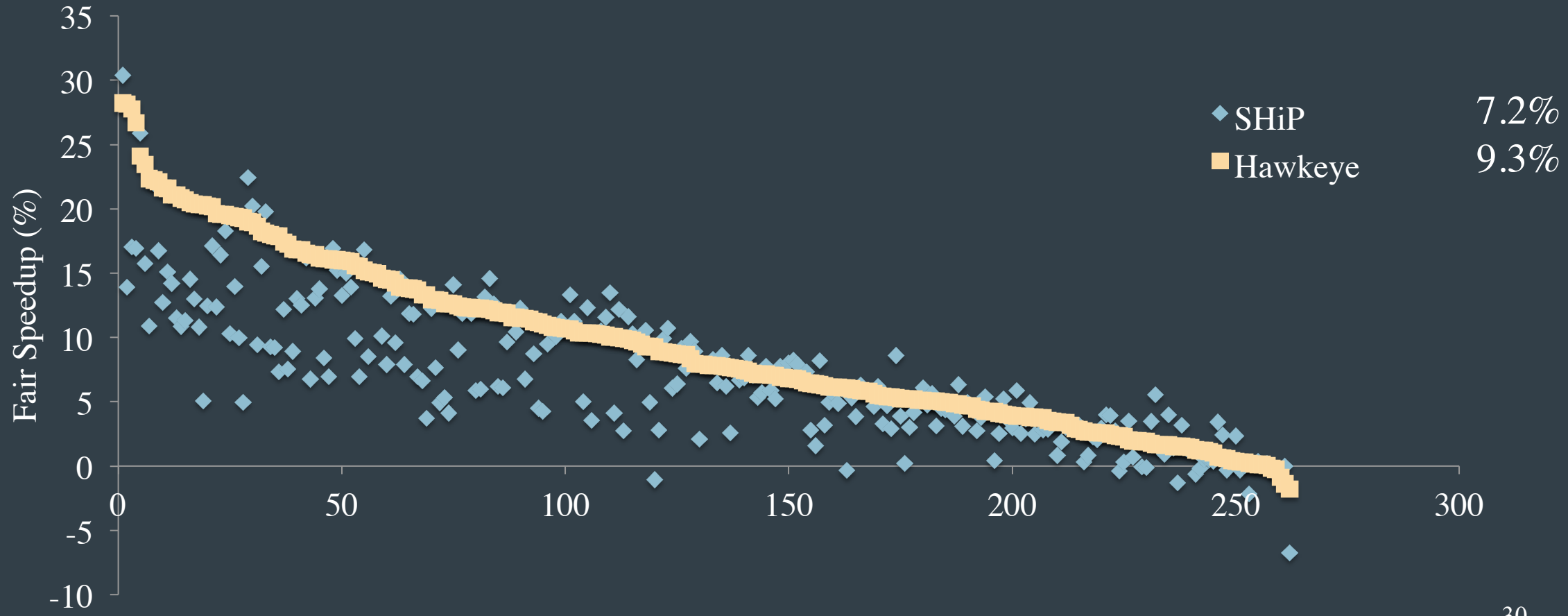
Performance Improvement With Prefetcher (Config2)



Performance Improvement (Config 1)



Multi-Core Results (Config 3)



Conclusions

- The Hawkeye Cache
  - Performs well in the absence of prefetching
  - It matches state-of-the-art solutions in the presence of prefetching

Conclusions

- Hawkeye doesn't really do better than SHiP in the presence of prefetching

- What is the optimal caching solution in the presence of prefetching?
  - If we know the answer, then a Hawkeye-like solution should do well

Thank You

- Questions?