Back to the Future: Leveraging Belady’s Algorithm for Improved Cache Replacement

Akanksha Jain
Calvin Lin
Cache Replacement

• On a cache miss, which line should we evict?
  – Well-studied problem

• Belady provided an optimal solution in 1966
Belady’s Optimal Algorithm (OPT)

• Evict the line that is accessed *farthest in the future*
  – Impractical
Existing Solutions

• Rely on heuristics that make *assumptions about the underlying access pattern*
  – LRU assumes recency-friendly accesses
  – MRU assumes thrashing accesses
  – Recent solutions use more sophisticated heuristics

• Problem: Heuristics don’t work well when their assumptions don’t hold
Example: Matrix Multiplication

A \* B = C

Medium-term Reuse

Long-term Reuse

Short-term Reuse

Hit Rate

LRU, MRU, DRRIP, SDBP, SHiP, OPT
Example: Matrix Multiplication

\[
\text{A} \quad \times \quad \text{B} \quad = \quad \text{C}
\]

Medium-term Reuse

Long-term Reuse

Short-term Reuse

Hit Rate

OPT

B (Long-term)

A (Medium-term)

C (Short-term)
Significant Headroom

We are going to use OPT.
Our Solution: Key Idea

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

• OPT looks arbitrarily far into the future
  – We might need an arbitrarily long history
How far in the future does OPT need to look?

![Graph showing error compared to Infinite OPT (%)]

- **Belady**
- **LRU**

**View of the Future (number of cache accesses):**

- 1×
- 2×
- 4×
- 8×
- 16×

**Error compared to Infinite OPT (%):**

- 70
- 60
- 50
- 40
- 30
- 20
- 10
- 0

**History length not unbounded, but we need to track past 8× cache accesses**
Our Solution

• Hawkeye Cache Replacement (Hawkeye)
  – Hawks can see $8 \times$ farther than the best humans
Hawkeye: Challenges

• Need to look at a long history (8× the cache size)

• Need to efficiently compute the OPT solution for a long history of past references

• New algorithm called OPTgen
  – Online (linear time)
  – Sampling (12KB overhead for 2MB cache)
Hawkeye: Overall Design

Cache Access Stream

OPTgen

Computes OPT’s decisions for the past

PC

OPT hit/miss

PC-based Predictor

Remembers past OPT decisions

Insertion Priority

Last Level Cache
Hawkeye: Overall Design

Training

With OPT, would X be a cache hit or miss?

Address: X

OPTgen

PC

Hit/Miss

PC-based Predictor

Last Level Cache
Hawkeye: Overall Design

**Prediction**

OPTgen → PC-based Predictor → Last Level Cache

Does this PC tend to load cache-friendly or cache-averse lines?

Address: X

Insert X with high or low priority
OPTgen

• Simple online algorithm that reproduces OPT’s solution for the past

• Inspired from Belady’s insight
  – Lines that are reused first have higher priority

• OPTgen also gives higher priority to lines that are reused first
OPTgen

- For each line, we ask if this line would have been a hit or miss with OPT.
OPTgen

- For each line, we ask if this line would have been a hit or miss with OPT.
OPTgen

- For each line, we ask if this line would have been a hit or miss with OPT.

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>C</th>
<th>E</th>
<th>D</th>
<th>D</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
</table>
```

Cache Capacity = 2

Past Behavior

Future

Miss
OPTgen is equivalent to OPT

• 100% accuracy with unlimited history
  – 95.5% accuracy with $8 \times$ history (for SPEC 2006)
OPTgen Algorithm: Insight 1

- OPT hit/miss can be determined at the time of reuse.

Don’t need information about future accesses because they will have lower priority than A.
OPTgen Algorithm: Insight 2

• OPT solution can be reproduced by tracking occupancy rather than cache contents
OPTgen Algorithm: Insight 2

 futuro 

 Past Behavior

 Future

 Cache Capacity = 2

 B

 A

 C

 D

 E

 D

 D

 time
OPTgen Algorithm: Insight 3

• OPT considers both reuse distances and their overlap

E’s reuse distance is smaller than A, but it misses with OPT because it has higher overlap

Cache Capacity = 2
Sampling for OPTgen

- 8× history for a 2MB cache
  - 100K entries (> 0.5MB)

- Set Dueling [Qureshi et al., ISCA 07]
  - Sample the behavior of a few cache sets
  - 64 sampled sets

- Set Dueling with OPTgen
  - Apply OPTgen to 64 sampled sets
  - 2.5K entries (12KB)
  - 95% accuracy in estimating miss rate
Hawkeye: Overall Design

Cache Access Stream

OPTgen

OPT hit/miss

PC-based Predictor

95% accurate

Insertion Priority

Last Level Cache
Evaluation

• Cache Replacement Championship Simulator (CRC)
• Benchmarks: Memory-intensive SPEC 2006
• Hardware Configurations
  – Single Core: Last-level Cache: 16-way, 2MB LLC
  – Multi-core: Shared Cache: 16-way, 4MB & 8MB LLC
• Replacement policies
  – Baseline: LRU
  – DRRIP [2010]
  – SDBP [2011]
  – SHiP [2012]
Miss Reduction

Miss Reduction over LRU (%)

-30 -20 -10 0 10 20 30 40 50 60

DRRIP  SHIP  SDBP

astar  gromacs  lbm  omnetpp  calculix  gcc  leslie  zeus  libq  bzip  h264  tonto  hmmer  xalanc  mcf  soplex  gems  cactus  sphinx3  Mean
Hawkeye outperforms DRRIP, SDBP and SHiP

Mean Miss Reduction over LRU (%) 17.4%
Miss Reduction

Hawkeye does not result in a slowdown for any benchmark
Miss Reduction

Hawkeye’s performance gains are consistent across benchmarks
Miss Reduction

- DRRIP
- SHIP
- SDBP
- Hawkeye
- OPT

Mean Miss Reduction over LRU (%)
Speedup

Speedup (%)

DRRIP  SHiP  SDBP  Hawkeye

Geomean 8.4%
Multi-Core Results

Averaged across 100s of multi-programmed SPEC runs
Hawkeye: Summary

- New goal: learn from the OPT solution
- Not limited to specific access patterns
- Models both reuse distance and demand
Conclusions and Future Work

• Recent trend to view *cache replacement as a prediction problem*
  – Learn from the past to predict the future

• Hawkeye learns from an oracle

• Future Work: More sophisticated predictors to learn the OPT solution for the past