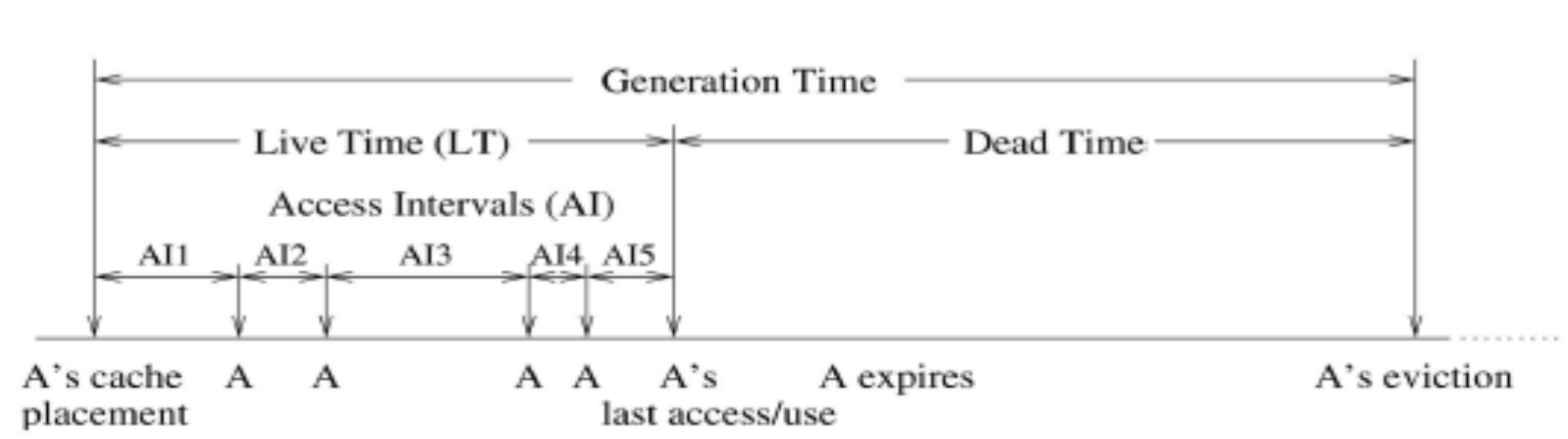
Increasing Cache Efficiency
by
Dead Block Prediction & Elimination

Anjana Mishra, Joshua Linge, Basundhara Dey

UNIVERSITY OF CENTRAL FLORIDA

Dead Block

- Live Time
- Dead Time



Cache Efficiency

- The average fraction of the cache blocks that store live data.

$$E = \frac{\sum_{i=0}^{A \times S - 1} U_i}{N \times A \times S}$$

- Cache efficiency is generally low.

- How can we improve cache efficiency?

Dead Block Predictor

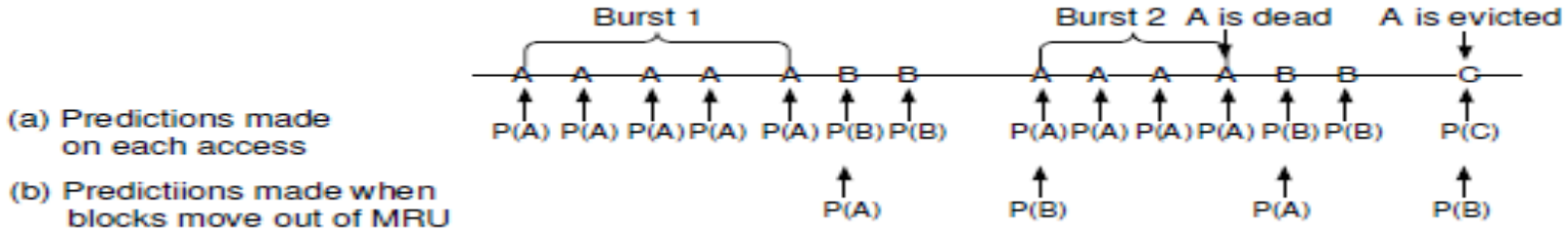
- Predict when a block is dead.
- Identify early during block's dead time.
- Replacing dead bocks with useful data.

Types of Dead Block Predictors

- History based prediction
  - Counting
  - Trace
- Time based prediction
  - Time Keeping

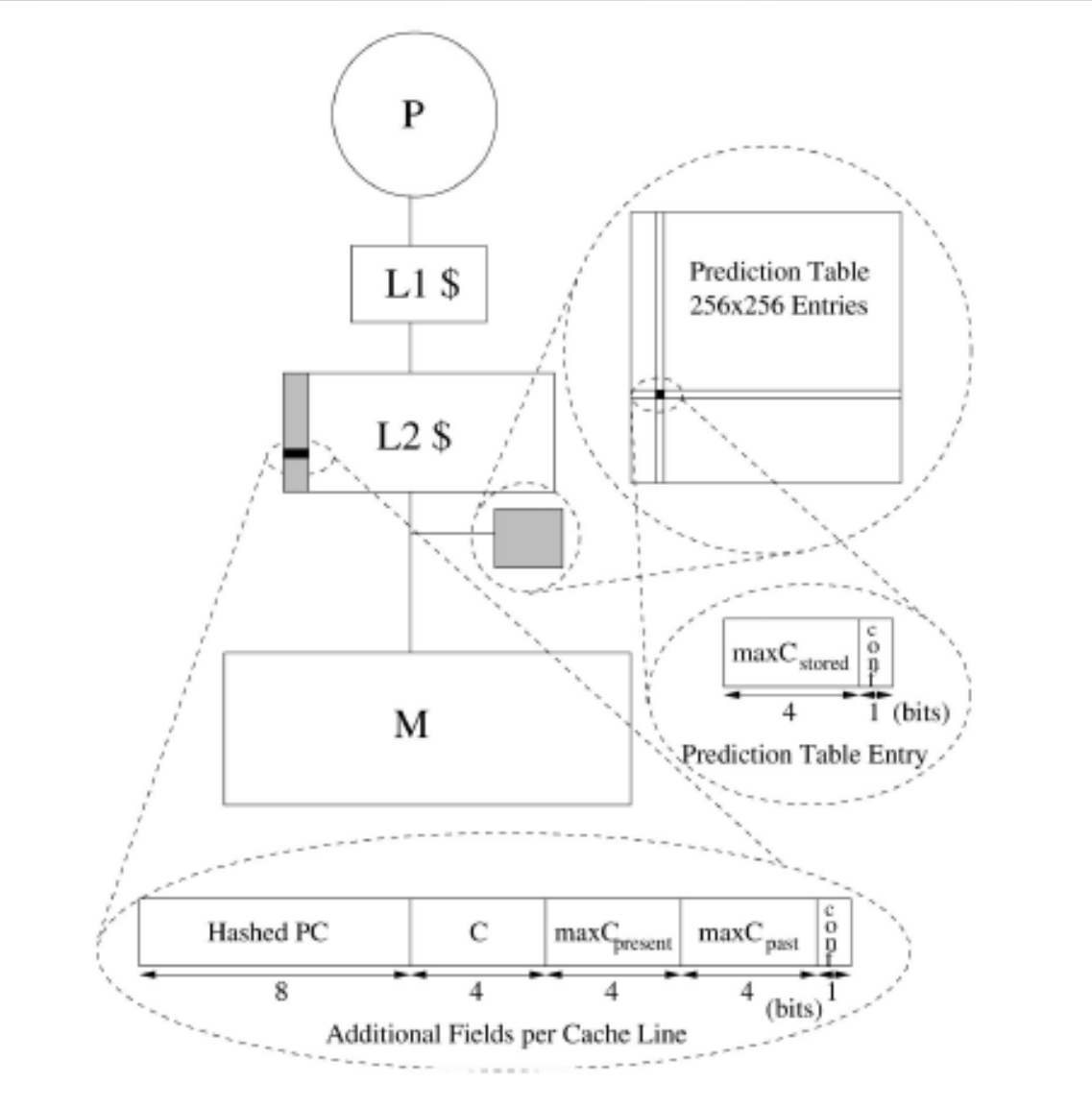
Reference History or Burst History

- Reference
  - Make a prediction after every reference to a cache block.
- Burst
  - Make a prediction only when a block moves out of the MRU position.



Counter Based Cache Replacement

- Relies on counters.
- Helps in-
  - To identify dead lines and replace them early.
- The easiest algorithm that can been used.
  - The Access Interval Predictor (AIP)

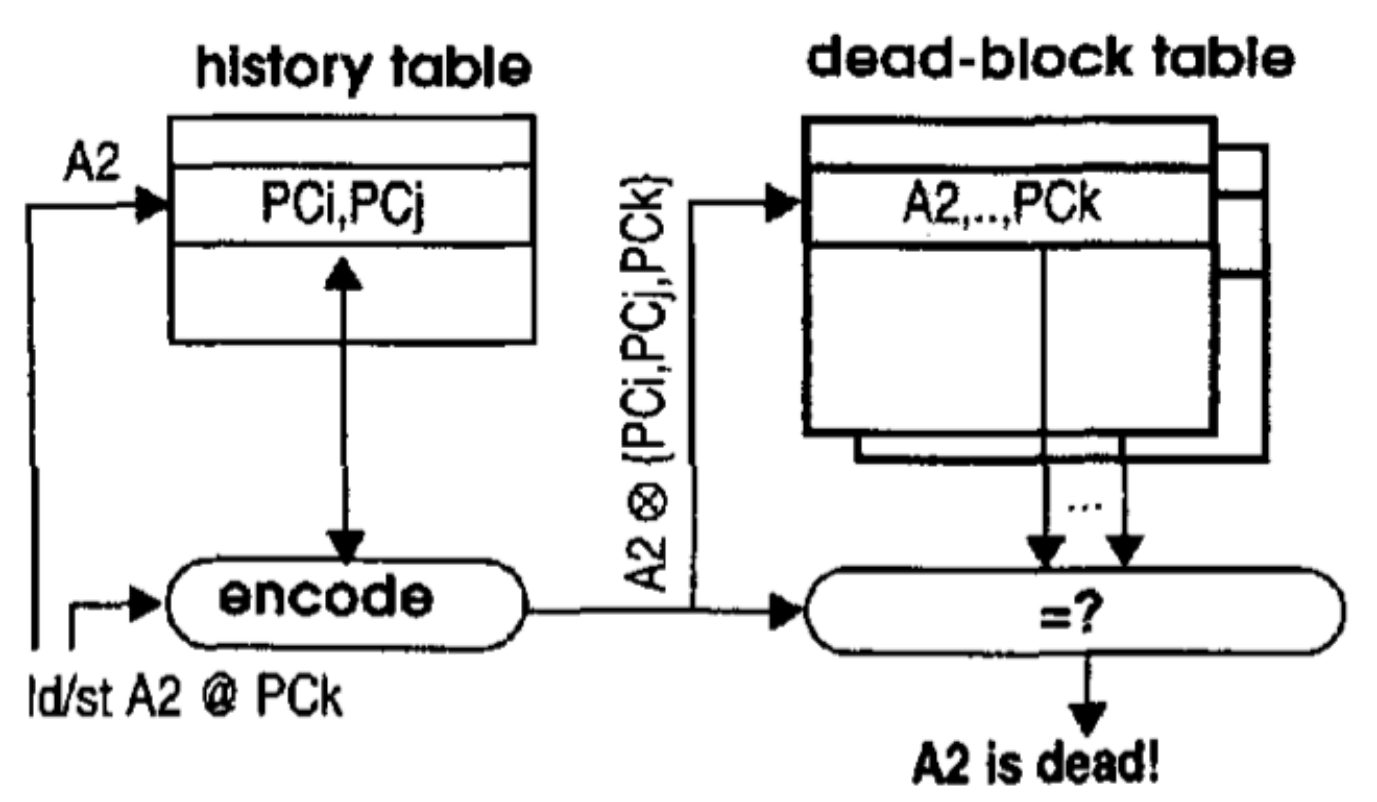


Storage organization and overhead of AIP.

Trace Based Prediction

- Keep track of PCs for each block access.
- Store block traces in a history table.
- When a block is replaced, add its trace to a dead-block table.
- If the dead-block table contains the current block trace, predict dead.

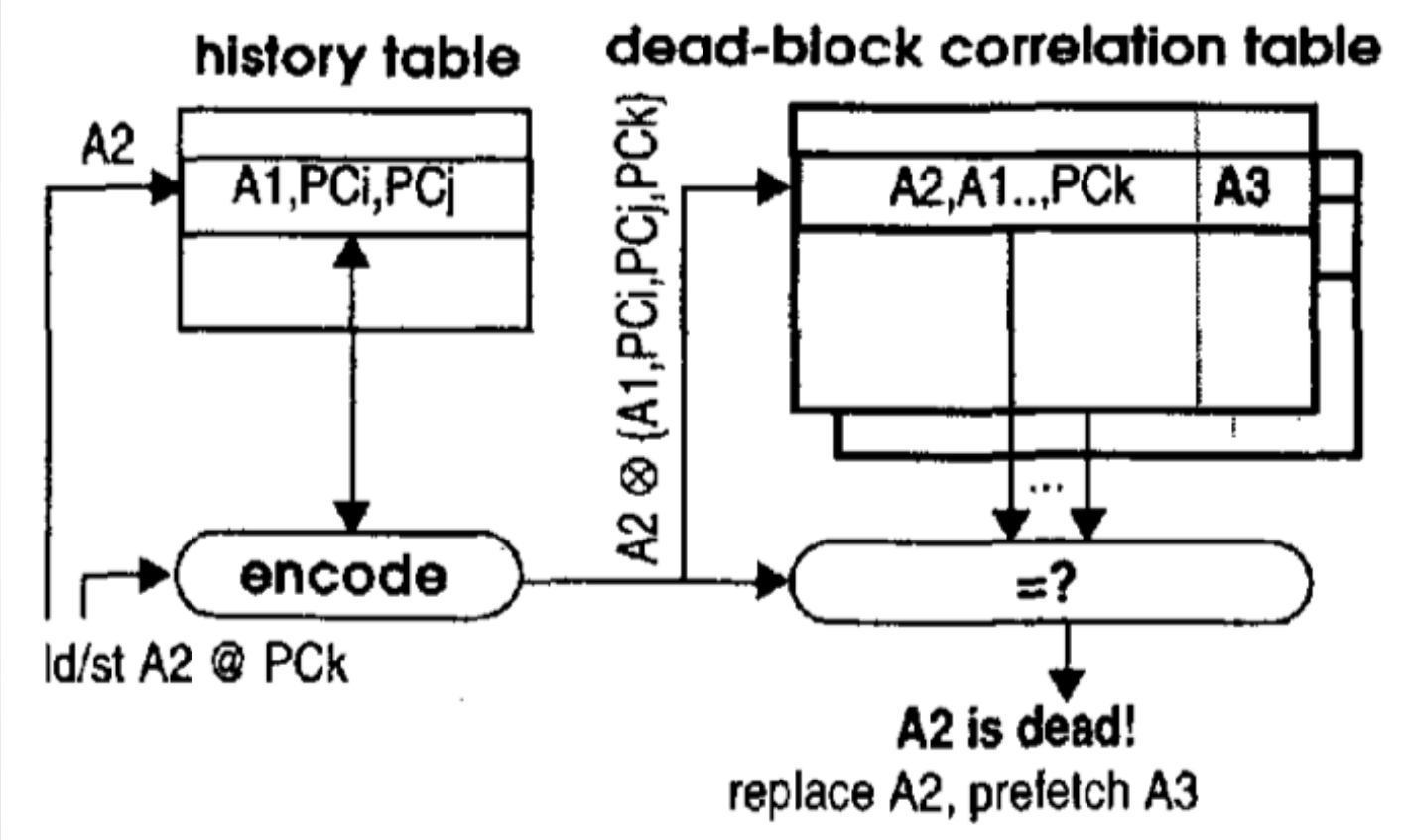
Trace Based Prediction



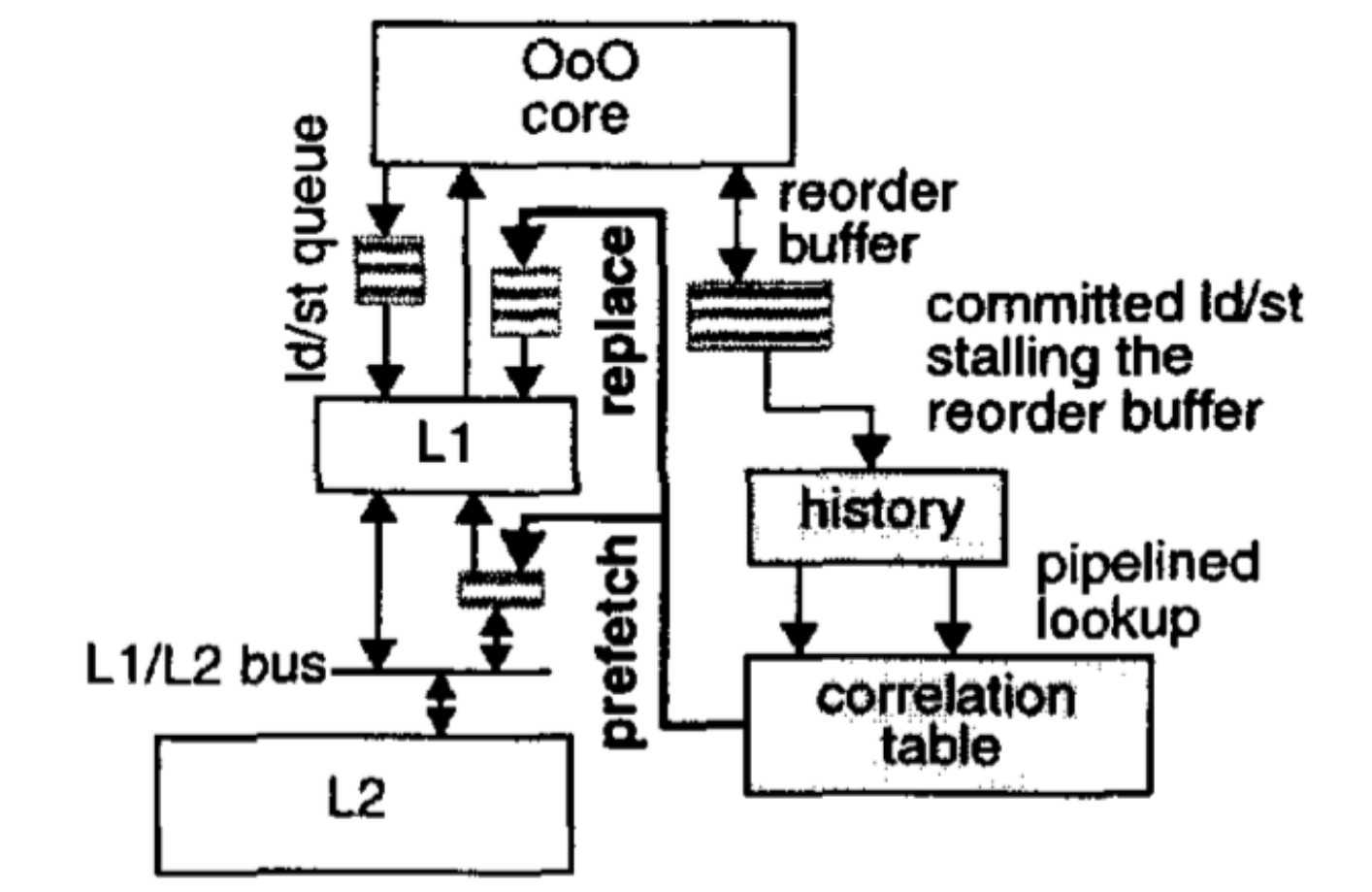
Using the Prediction

- Replace into dead block instead of LRU.
- Bypass the Cache entirely.
- Prefetch into dead blocks.

Trace Based Prefetching



Trace Based Prediction



Simulation

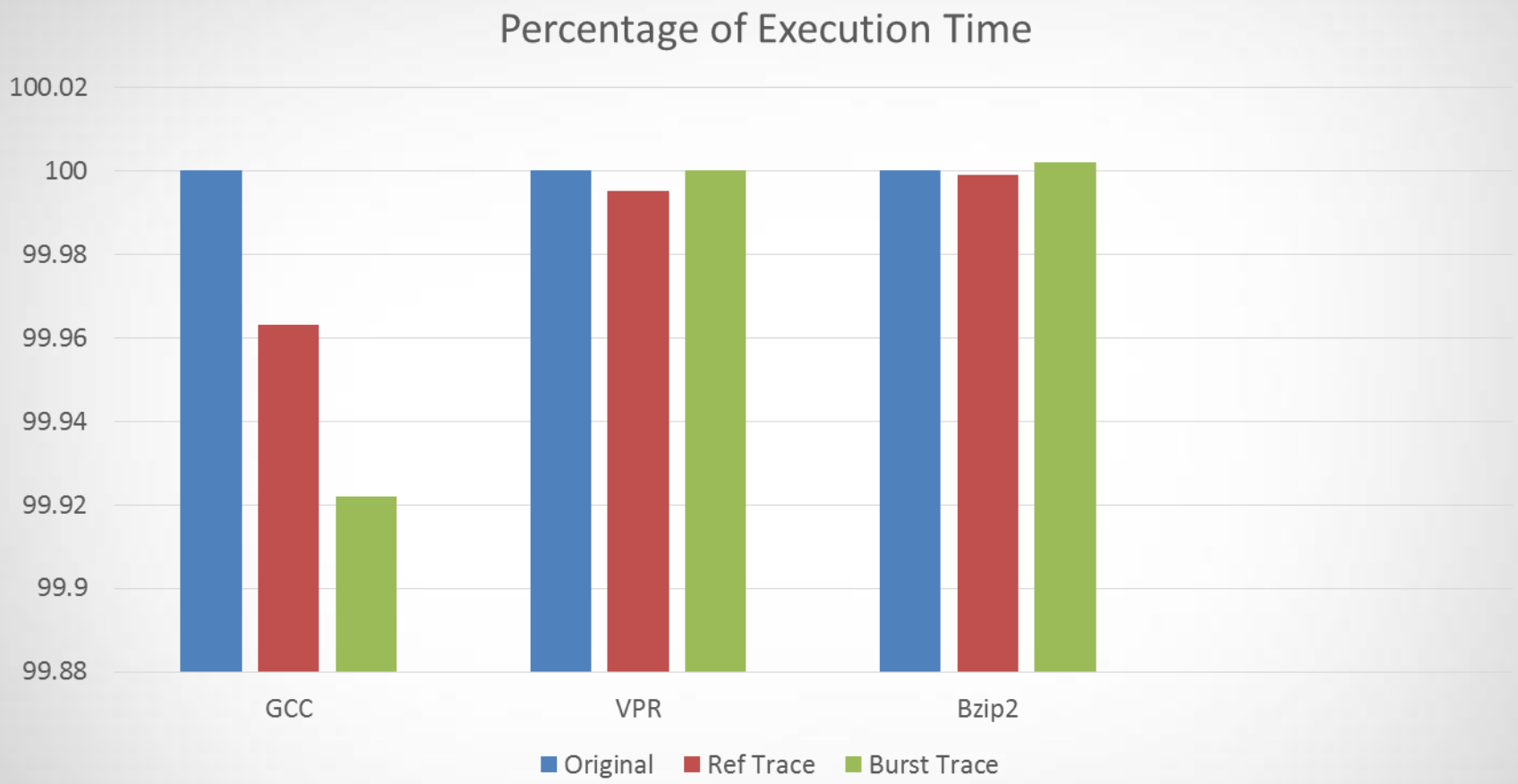
- Simplescalar
  - Width: 4
  - Load Store Queue: 8
  - ROB: 16
  - L1: 16KB, 32B block size, 2-way associative
  - L2: 512KB, 128B block size, 8-way associative
- Benchmarks
  - GCC, VPR, BZIP

Results

Cycles	GCC	VPR	Bzip2
Original	1563170315	103170742417	57734403829
Ref Trace	1562597566	103165823056	57733882196
Burst Trace	1561951766	103170871017	57735542066

	GCC	VPR	Bzip2
Original	100.0%	100.0%	100.0%
Ref Trace	99.963%	99.995%	99.999%
Burst Trace	99.922%	100.0%	100.002%

Graph Plot


Percentage of Execution Time
100.02
100
99.98
99.96
99.94
99.92
99.9
99.88
GCC
VPR
Bzip2
Original
Ref Trace
Burst Trace

Conclusion

- Using a dead block predictor improves cache efficiency.
- Burst history is more predictable than the reference history.
- Trace based dead block prediction and prefetching is effective in increasing the cache efficiency.

Question

How can dead block prediction and removal increase the cache efficiency?

- Dead block removal enables us to fill some live data into the cache and this increases the fraction of cache block that stores live data (refer to the formula in slide 3) and thus the cache efficiency.