Triangel: A High-Performance, Accurate, Timely On-Chip Temporal Prefetcher

Sam Ainsworth (University of Edinburgh), Lev Mukhanov (Queen Mary University of London)

Background and Motivation.

- On-chip Temporal prefetchers use Markov tables stored in repurposed parts of the cache to store data address access patterns and prefetch it when required.
- Target workloads? Why not covered by other prefetchers…
- ARM introduced CRC (Correlating Miss Chaining). Few details.
- The details are only available for the Triage prefetcher [MICRO 2019] and Triage-ISR [ToC 2022].

PC: A->B->C

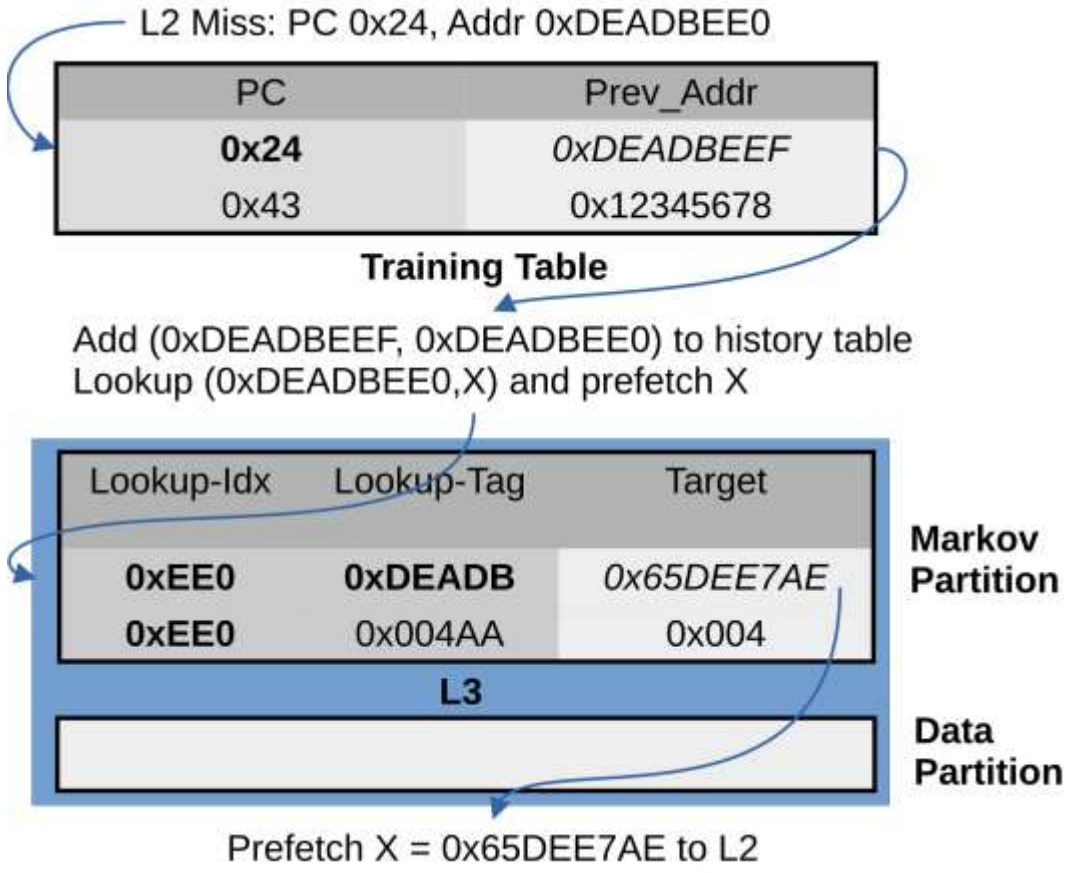
A (appears) -> B (prefetched)

Markov table

Index	Target
A	B
B	C

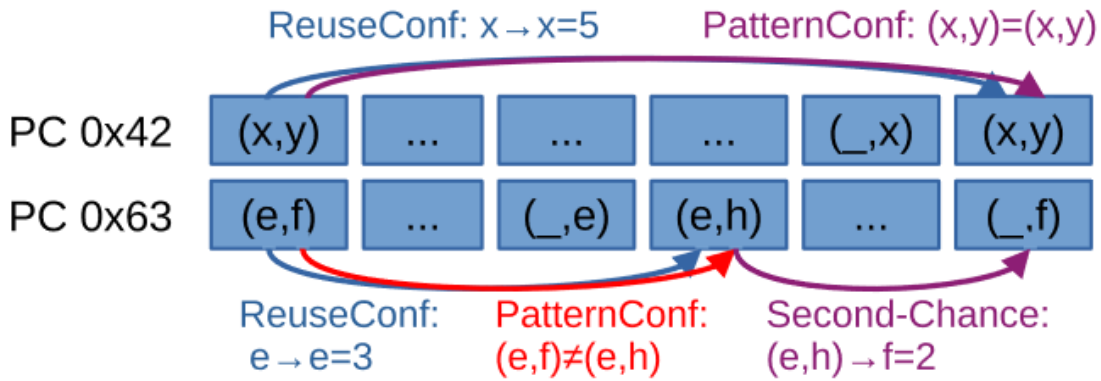
Triage.

- Prefetch happens at the L2 level (misses and tagged-pf hits).
- Training table contains PCs and the last accessed address (Prev_Addr).
- Markov table stores address pairs (A->B, B->C). It is placed in a partition of L3.
- In case of a cache miss or the prefetch hit, the Markov table is updated.
- The Markov table is not indexed by PC but is trained by PC-local streams.
- We infer a large set of implications about Triage's design that are suboptimal or impossible to implement, and fix them – see our paper or backup slides!

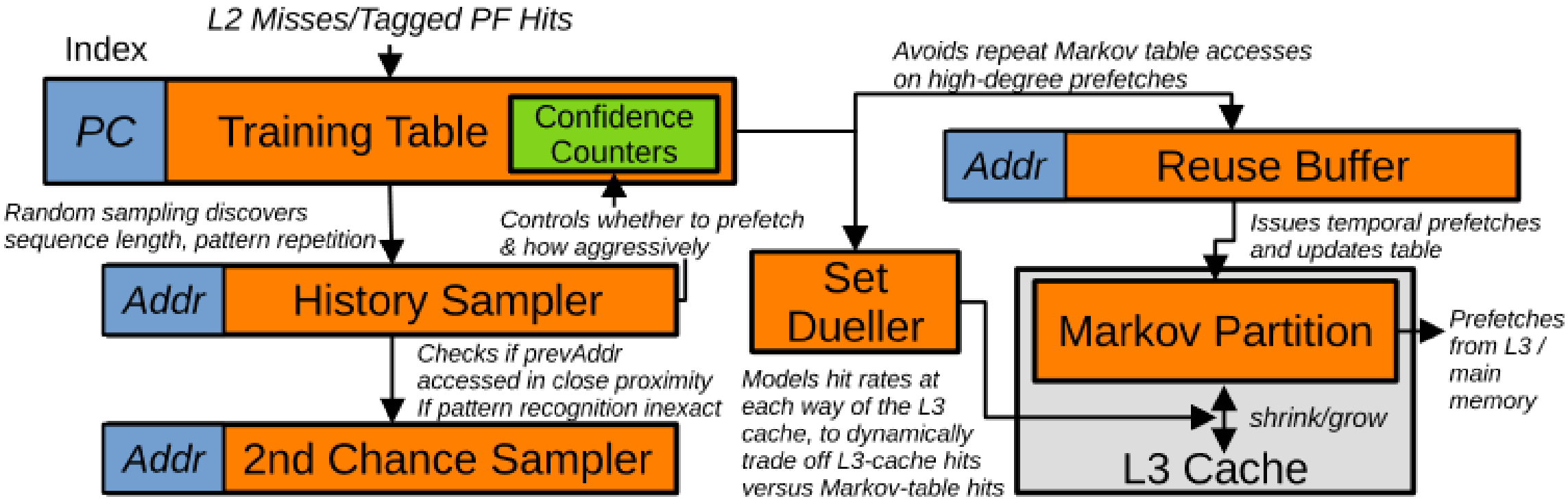


Triangel.

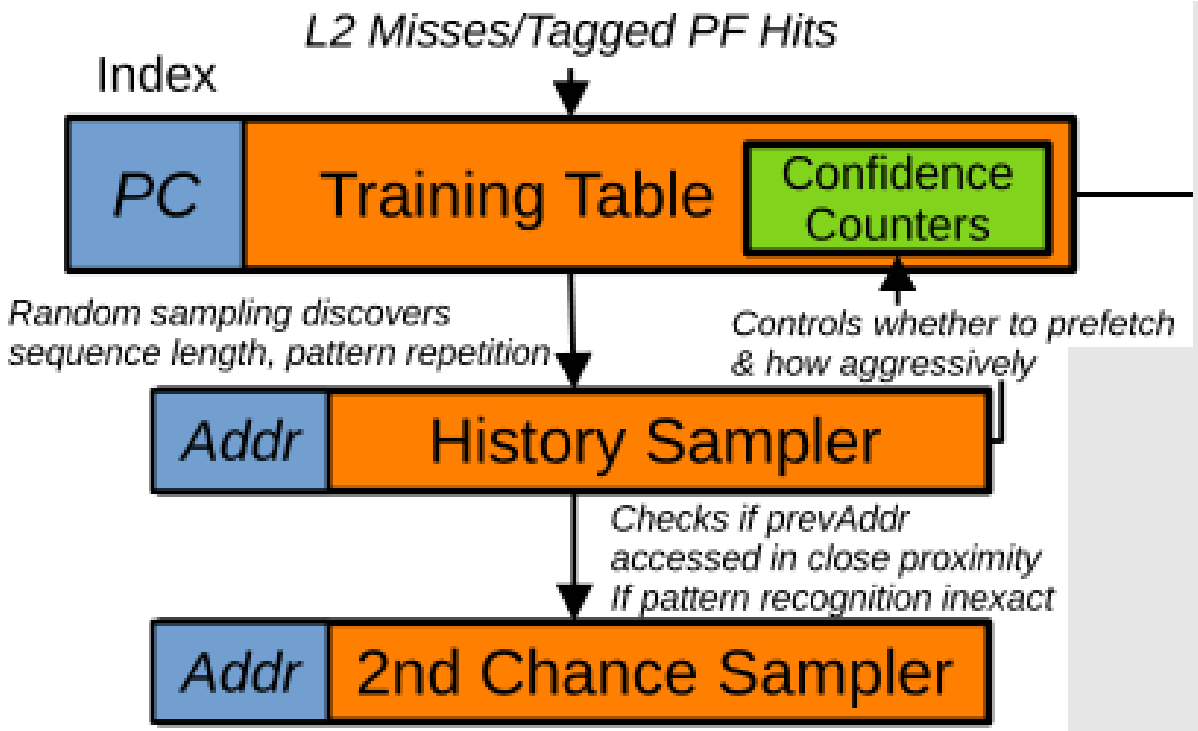
- Not all address patterns repeat. Triage can't tell if it sees a repeating pattern or not, so its prefetches are inaccurate.
- When Triage is accurate, it isn't aggressive enough, so is not timely, and not all memory latency is hidden.
- Can we have the best of both worlds? YES – by History Sampling to work out for each PC, whether there is an accurate pattern that fits in the Markov table.
- If the pattern reliably repeats (PatternConfidence) and fits in the Markov Table (ReuseConfidence), we store it in the Markov Table.
- If we are very confident in the pattern (HighPatternConfidence), we prefetch aggressively, at high Degree and Lookahead, to improve timeliness.
- Prefetches can be good even if the pattern doesn't repeat perfectly, if they are still used before eviction from the L2



Triangel


L2 Misses/Tagged PF Hits
Index
PC
Training Table
Confidence Counters
Avoids repeat Markov table accesses on high-degree prefetches
Addr
Reuse Buffer
Random sampling discovers sequence length, pattern repetition
Controls whether to prefetch & how aggressively
Issues temporal prefetches and updates table
Addr
History Sampler
Set Dueller
Markov Partition
Prefetches from L3 / main memory
Checks if prevAddr accessed in close proximity If pattern recognition inexact
Models hit rates at each way of the L3 cache, to dynamically trade off L3-cache hits versus Markov-table hits
shrink/grow
Addr
2nd Chance Sampler
L3 Cache

Triangel's Samplers



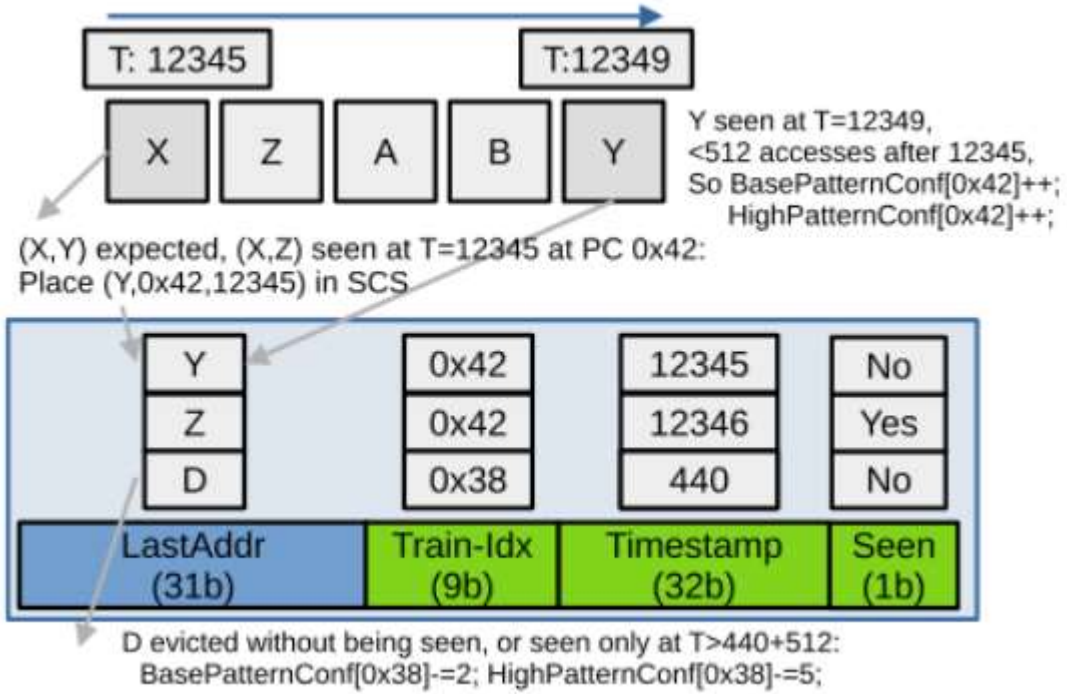
History Sampler

- We randomly sample (to fit a full history in a tiny 512-entry table) the entries the Training Table would have sent to the Markov Table if it were prefetching.
- If the target is identical next time we see an index, and that is within a time limit where it would fit inside the L3 Markov table, great!

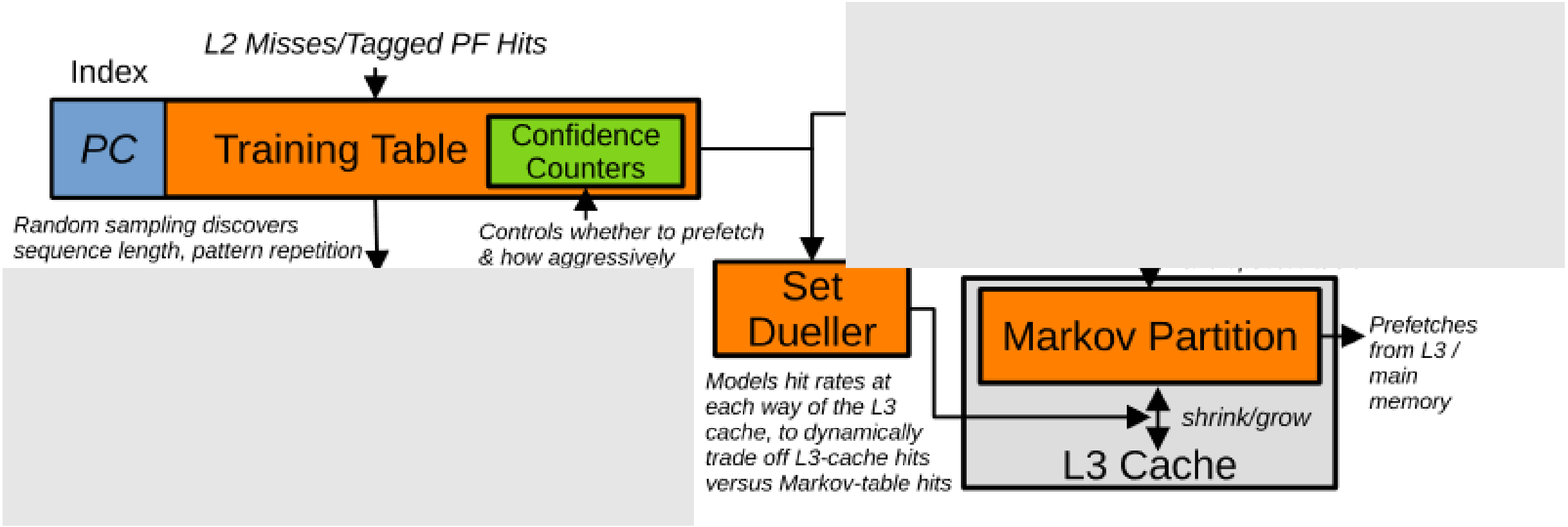
Access **If** Addr-Tag == *Training[PC]*.LastAddr[0]
&& Train-Idx == *&Training[PC]*
Replace **If** Random_Chance(*Training[PC]*.SampleRate)

Second-Chance Sampler

- If the pattern doesn't match, the prefetch stream might still be good if the resulting prefetch would still be used before being evicted from the cache.
- So we place the expected address and associated PC into a small buffer with a timestamp – if it is accessed within a small timeframe, confidence is increased, otherwise it is decreased.

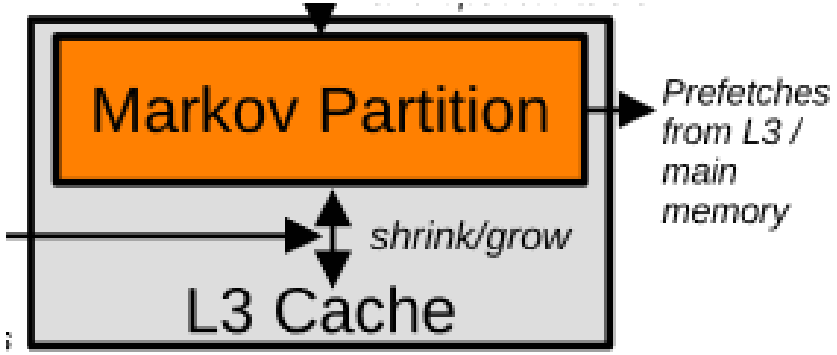


Markov-Table Sizing: the Set Dueller



Markov-Table Sizing

- Triage(ISR) uses a Bloom Filter that increases the size of the Markov Partition while there are still unique addresses.
- This oversizes the Markov partition if the L3 cache is more useful, causing lots of extra DRAM traffic through capacity misses.

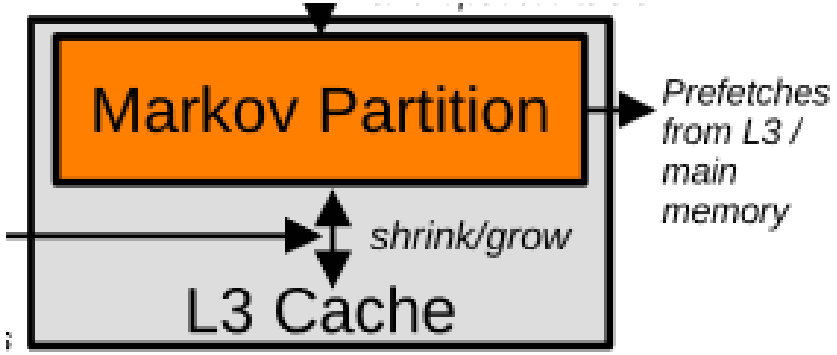


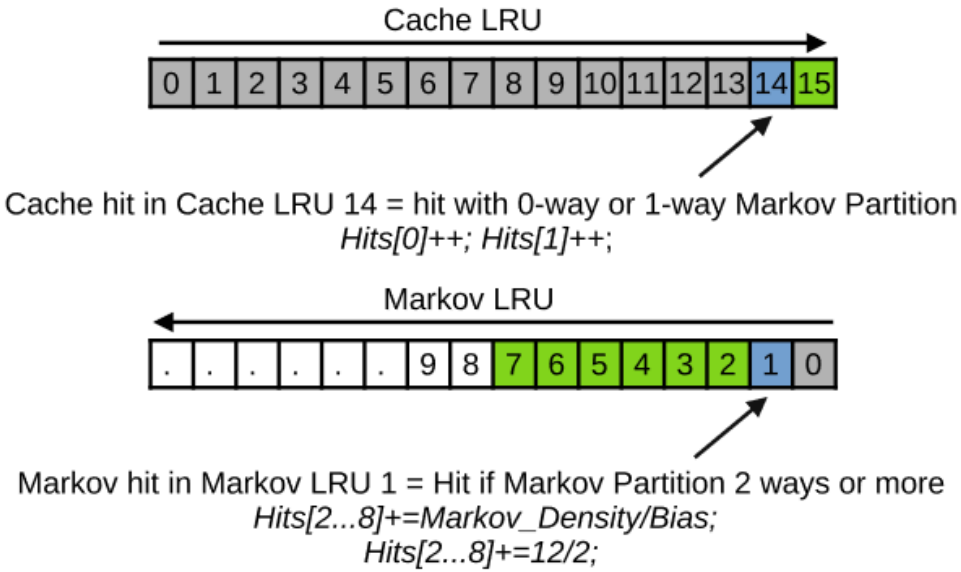
THE UNIVERSITY OF EDINBURGH

Queen Mary University of London

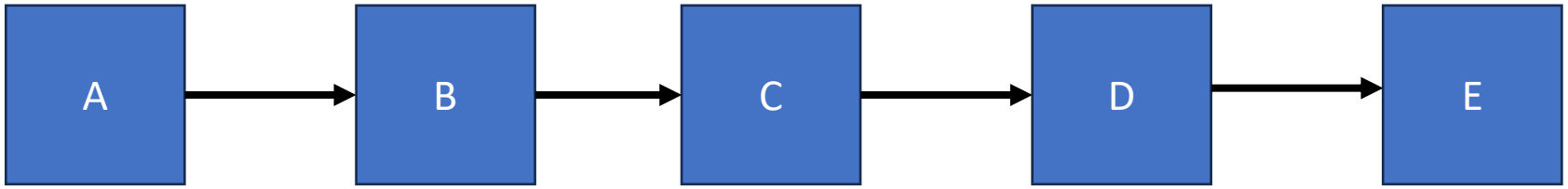
Markov-Table Sizing

- Triangel uses a Set-Dueller, trained on the filtered output of the Samplers
- It stores a full Markov Table and Full Cache for 64 random sets – and works out for every possible partition which would be the highest joint hit rate.
- More tricks than this, to consider every outcome while storing only two alternatives, and to handle the Markov table being denser than cache… see the paper!



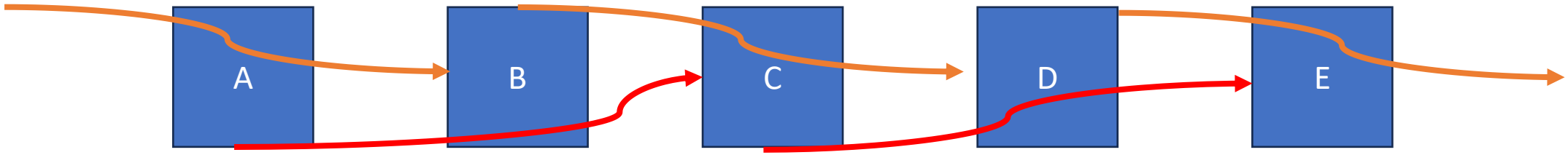


High-Lookahead Prefetching



- Each sequential lookup is slow, and fails to hide Main-Memory Latency

High-Lookahead Prefetching



- Offset the index-tag pairs by one to increase timeliness without harming coverage!
- Can be implemented with a shift register in each training-table entries, to offset the Markov-table pairs stored.

Combine with High-Degree Prefetching when Accurate (According to Samplers)

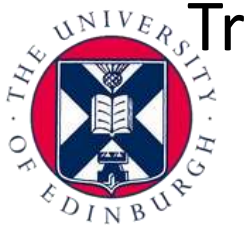
- Aggressively issue 4 prefetches per miss/pf-hit if >5/6 chance of useful prefetch (according to highPatternConfidence in the Samplers)
- Note the chained lookup with lots of redundant Markov-table lookups: use a Metadata Reuse Buffer to avoid L3-Markov-Table accesses and latency!

Experimental Setup

- We implement in gem5
- Open source – download and reuse our Triage and Triangel implementations from Github (link in paper)!
- Evaluate on the 7 most irregular workloads from SPEC CPU2006, as with Triage paper.

Core	5-Wide, out-of-order, 2GHz
Pipeline	288-Entry ROB, 120-entry IQ, 85-entry LQ, 90-entry SQ, 150 Int / 256 FP registers, 4 Int ALUs, 2 Mul/Div, 4 FP/SIMD, 2 R/W Ports
Branch	64KiB MPP-TAGE
L1 ICache	64KiB, 4-way, 2-cycle hit lat, 16 MSHRs
L1 DCache	64KiB, 4-way, 4-cycle hit lat, 16 MSHRs, deg-8 stride pf
L2 Cache	512KiB, 8-way, 9-cycle hit lat, 32 MSHRs
L3 Cache	2MiB/core, 16-way, 20-cycle hit lat, 36 MSHRs
Memory	LPDDR5_5500_1x16_BG_BL32
OS	Ubuntu 22.04

TABLE II: Core and memory experimental setup.



Queen Mary University of London

Results

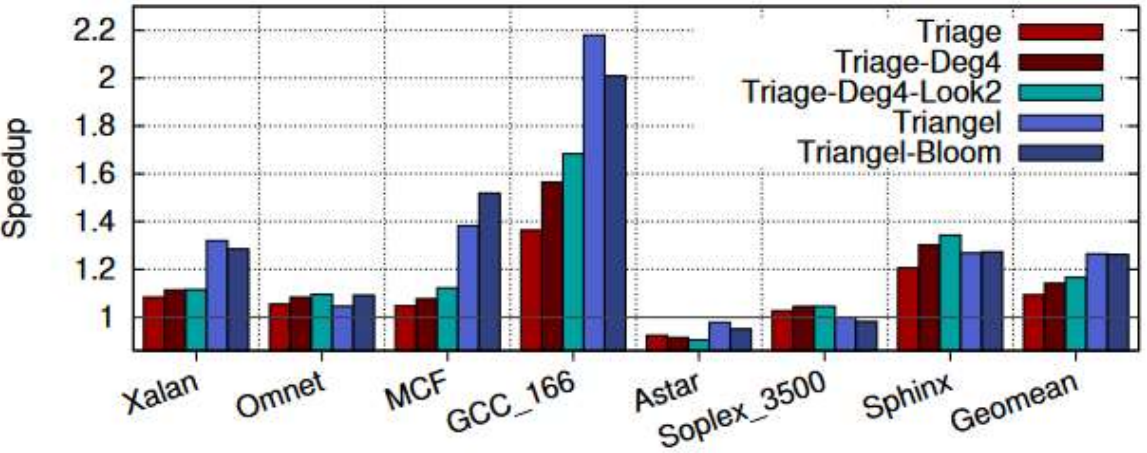


Fig. 10: Speedup

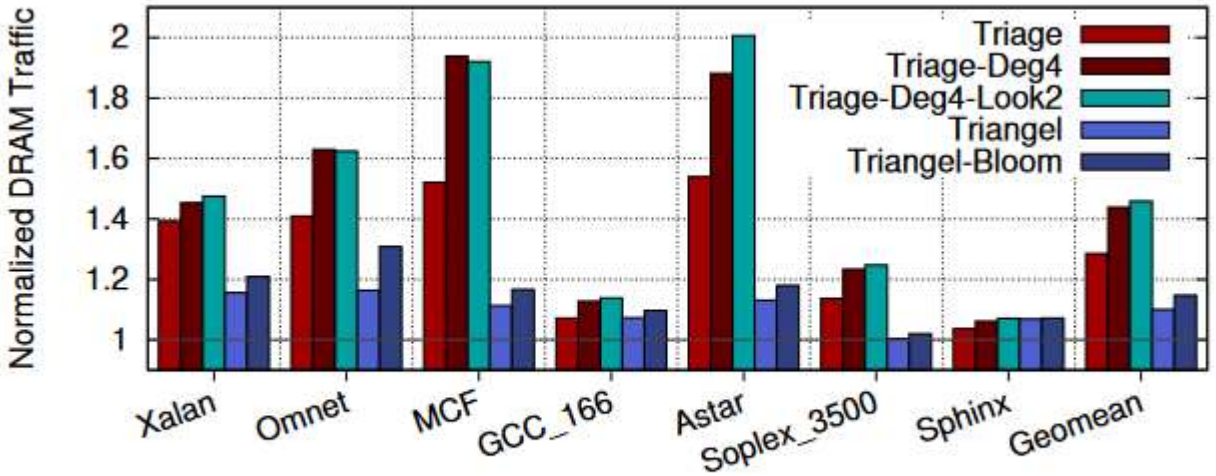


Fig. 11: Normalized DRAM Traffic (lower is better).

Triangel is faster and has lower DRAM traffic than Triage regardless of Triage's configuration, because Triangel can selectively switch off entirely or become aggressive or not based on sampling!

Triangel also benefits from its Set Dueller, which reduces traffic without harming performance compared to a Bloom-Filter Strategy

Accuracy and Coverage

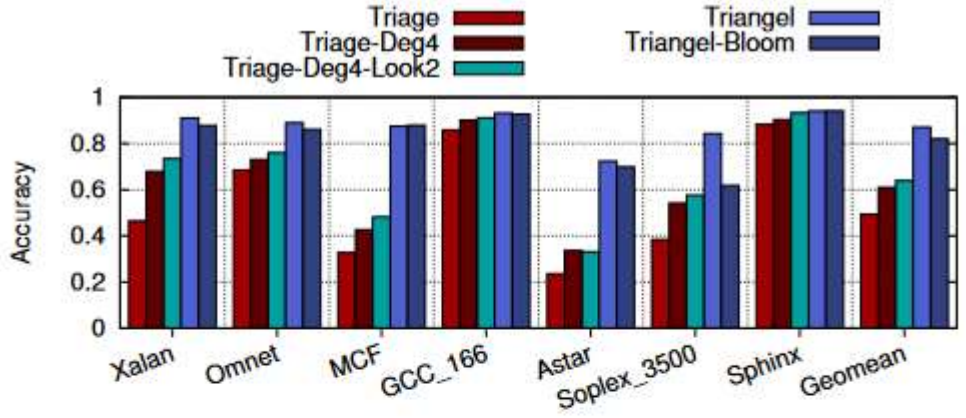


Fig. 12: Accuracy (prefetched lines used before L2 eviction).

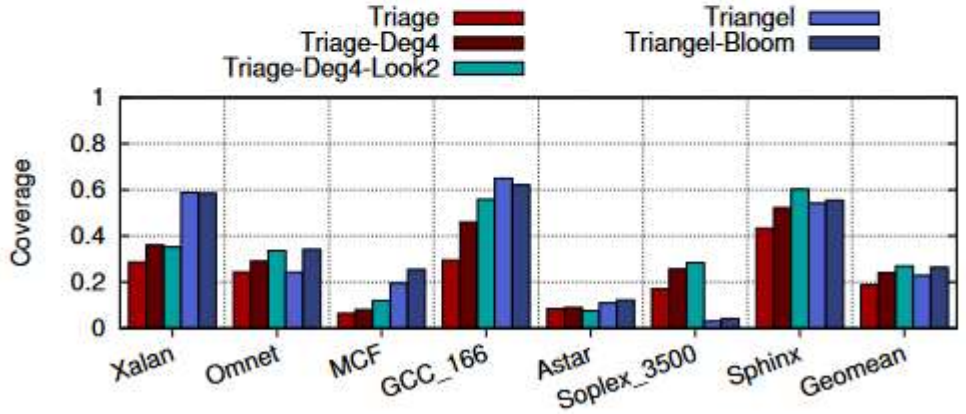


Fig. 13: Coverage (CPU L2-cache demand misses eliminated from baseline).

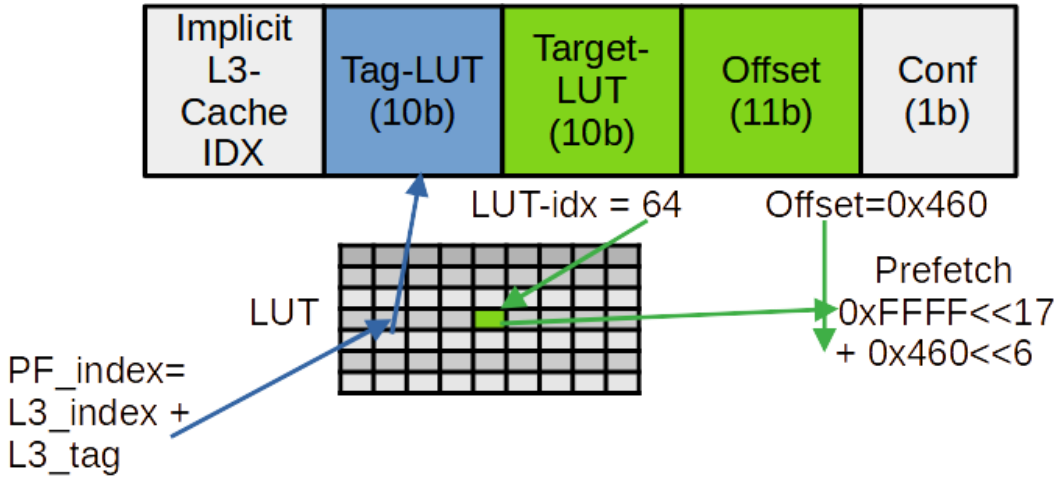
- Triangel is more accurate but typically lower coverage, precisely because it knows when to switch off!
- Triangel can sometimes be higher coverage (e.g. MCF) because it only stores good prefetches, increasing capacity.
- Triangel will sometimes switch itself off with good prefetches (Omnet) when it works out it's not worth the cost in L3 Misses from the partitioning (via its Set Dueller)

Triangel: A High-Performance, Accurate, Timely On-Chip Temporal Prefetcher

Sam Ainsworth (University of Edinburgh), Lev Mukhanov (Queen Mary University of London)

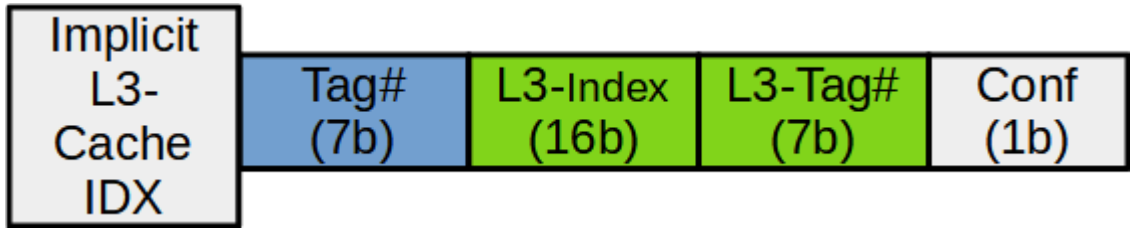
Triage. Lookup table and meta data format.

- The lookup and prefetch addresses are compressed to fit in 32 bits.
- The lookup address: cache set(11b)+ Inverse_LUT[Tag](10b)
- The prefetched address: LUT[LUT-idx(10b)] + offset(11b)
- The lookup has presumably 1024 entries (10-bit direct-index lookup). It is likely stored separately from the cache.
- The inverse LUT access requires the L3 index to be precisely 11 bits, otherwise the otherwise unrelated PF_target offset must grow beyond 11 bits.
- This table assumes common upper address bits, and has poor accuracy even with a typical amount of physical-address fragmentation.



Triage-ISR.

- The Triage-ISR removes the lookup table.
- 7-bit hash is stored instead of the full tag being retrieved via lookup table.
- The address: 16-bit index of the L3 cache + 7-bit hash for the remaining tag.
- 22 bits – not enough to uniquely identify an address in memory, so in reality can only be used for prefetch targets still in the L3, not from DRAM…
- 7 hash bits also not enough – high false positive rate with implied high associativity

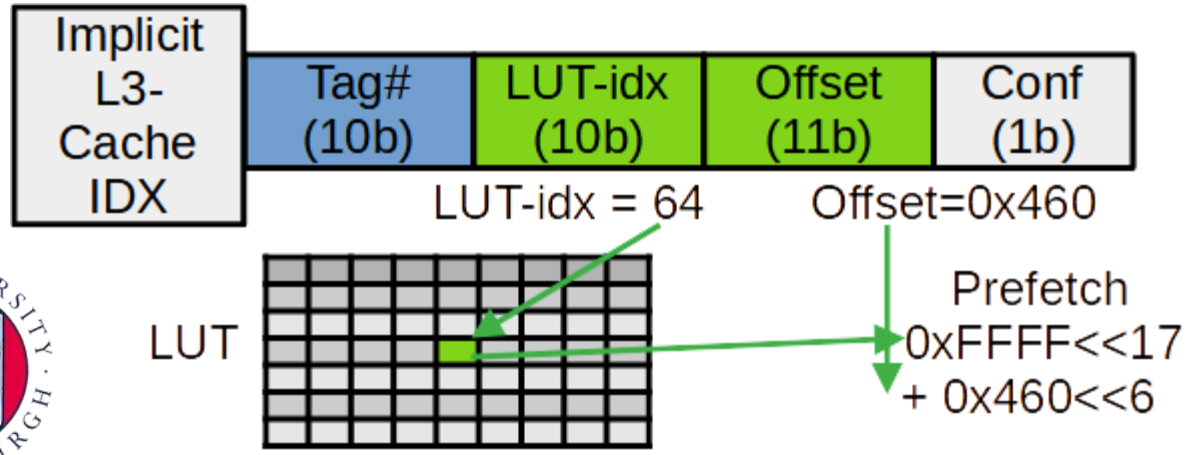


THE UNIVERSITY OF EDINBURGH

Queen Mary University of London

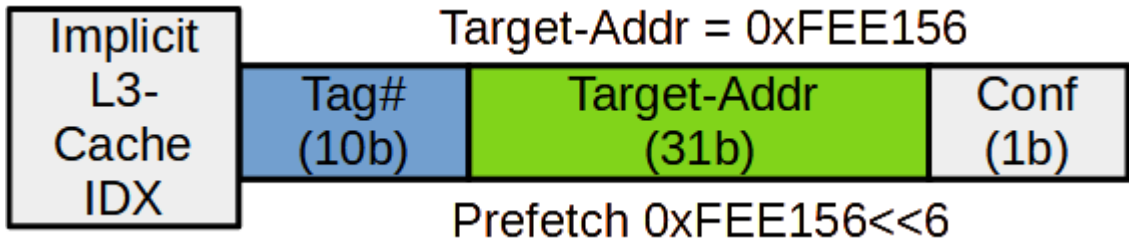
An implementable, practical Triage

- Our new implementable Triage baseline takes the best of both: removes the inaccurate, L3-size-limiting reverse lookup table for the index, using Triage-ISR Hashing instead, but uses a lookup table for the prefetch target to allow DRAM prefetches.



Triangel's Metadata

- Triangel stores a full address for the target, as this achieves higher overall performance despite increasing Markov-table entries to 42 bits and dropping density, due to superior accuracy



THE UNIVERSITY OF EDINBURGH

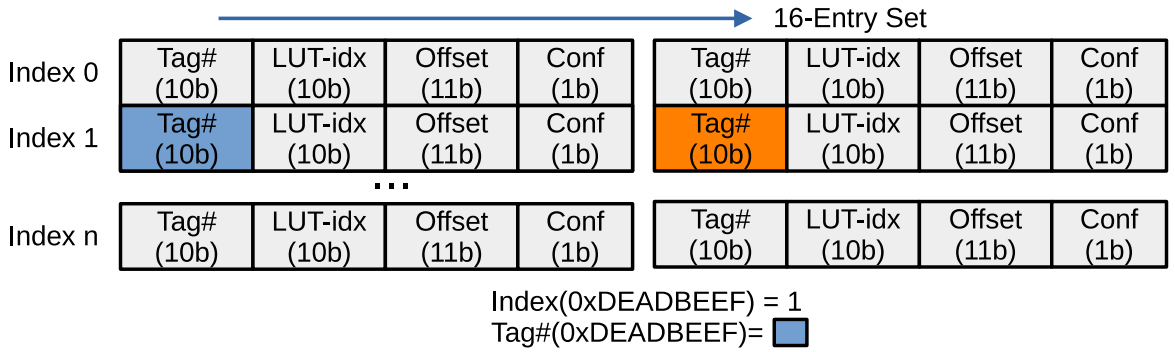
Queen Mary University of London

Fixing Triage's Associativity and Indexing.

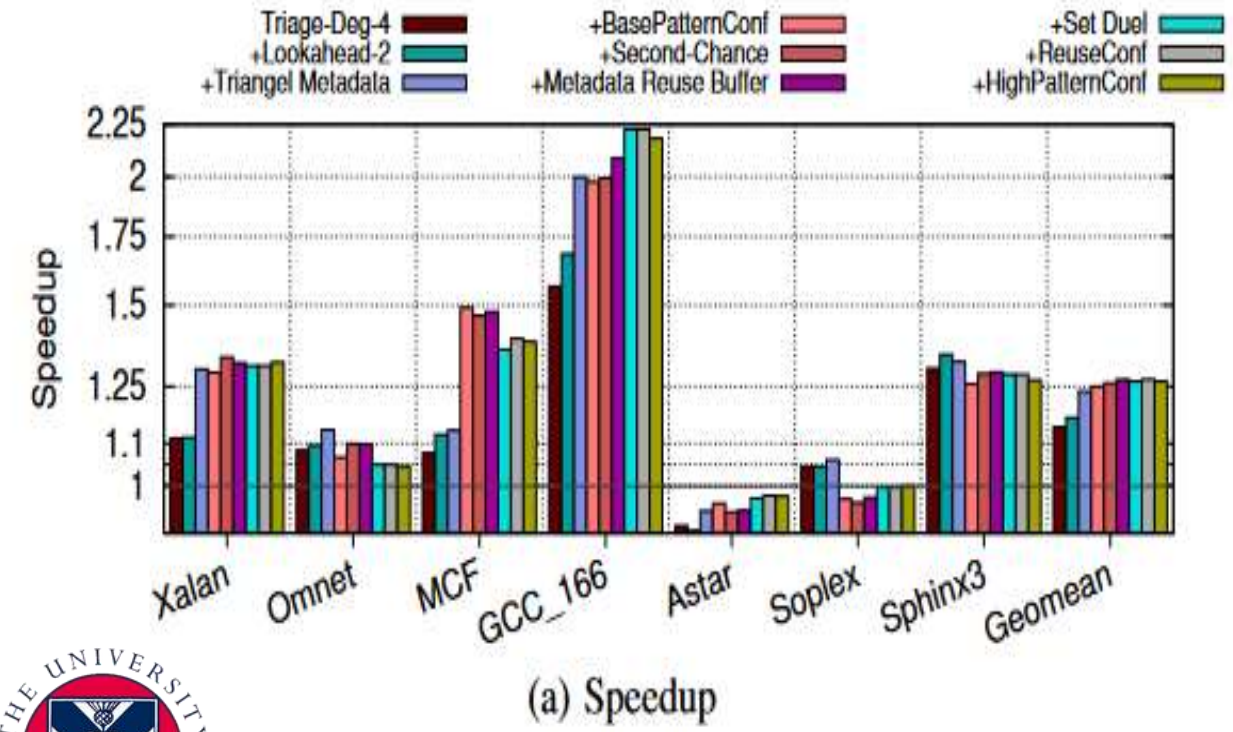
- 16 entries per 64 bytes cache line at L3.
- Tags are compressed *inside* the L3's cache lines, as there are too many to fit inside the L3's tags
- For an 8-way partition of the L3 cache, there are 128 possible tags that could match inside 8 cache lines.
- Thus, we have 160 cycles to access all compressed tags against 20 cycles assumed in paper.
- To address this, we introduce a second indexing policy and to choose a Sub-Set (cache line/way within the L3 cache's set):

  Index=Tag#%Partition_ways_Markov

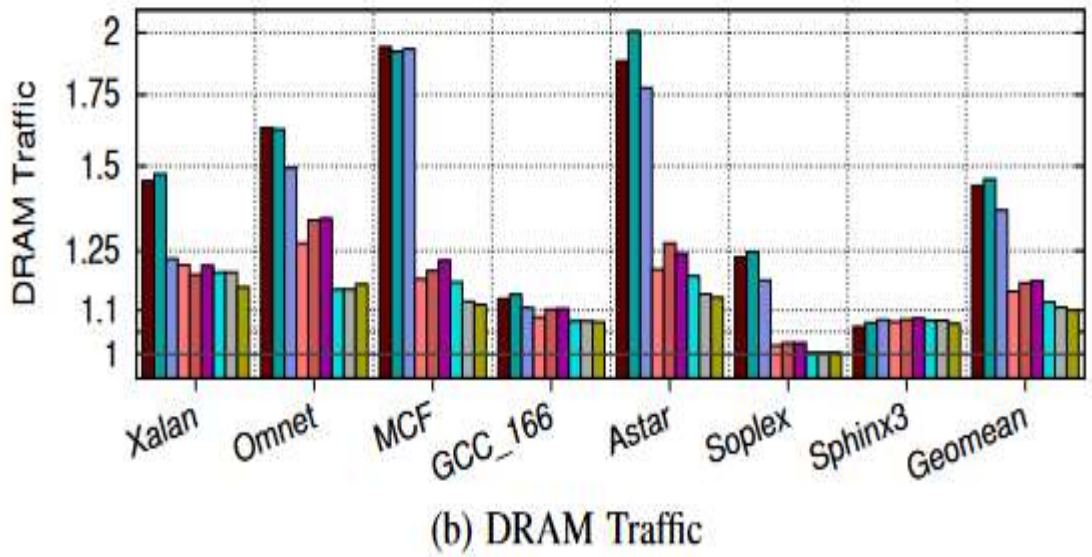
- This means that only one cache line must be accessed for a Markov-table lookup, that the Markov table is always 16-way set associative (12-way for Triangel), but that Markov-table entries must be moved every time the partition size changes, so that they are within the correct Sub-Set.



Backup Results: Triangel Benefits from lots of features



(a) Speedup



(b) DRAM Traffic

Fig. 20: Impact of progressively adding individual features to form Triangel, starting with Triage Degree 4.

Backup Results: Triangel saves energy through only storing and issuing good prefetches, and through removing redundant lookups on high-degree prefetches (Metadata Reuse Buffer)

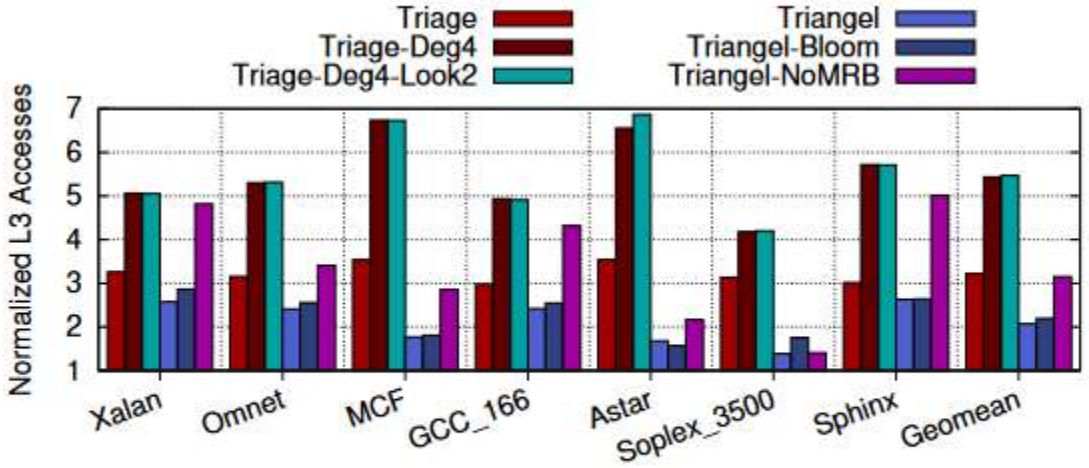


Fig. 14: Normalized L3-cache traffic, including Markov-table accesses and L3 data accesses (lower is better).

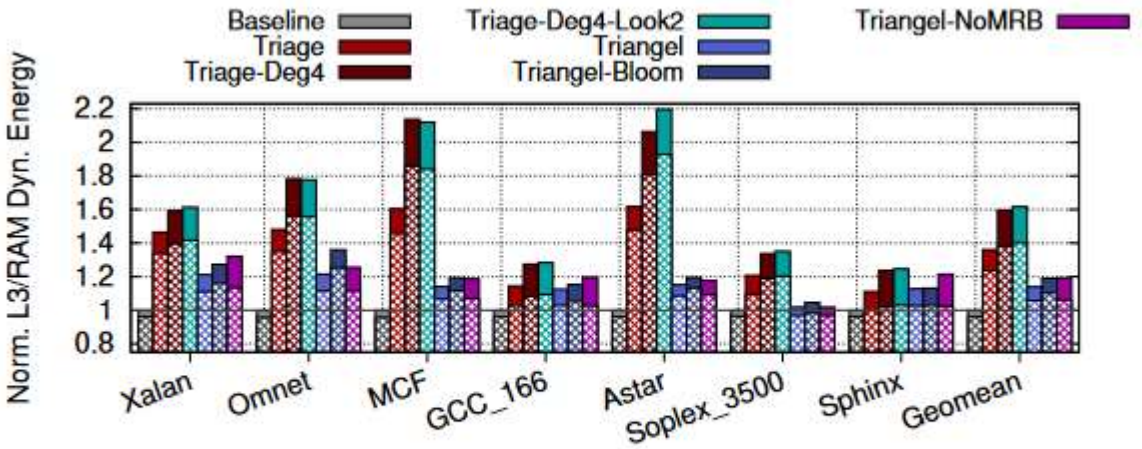


Fig. 15: Normalized combined DRAM+L3 dynamic energy (lower better). Hashed bars represent the DRAM proportion.

THE UNIVERSITY OF EDINBURGH

Queen Mary University of London

Backup Results: Triangel handles Multiprogrammed workloads well because of a lack of excess traffic

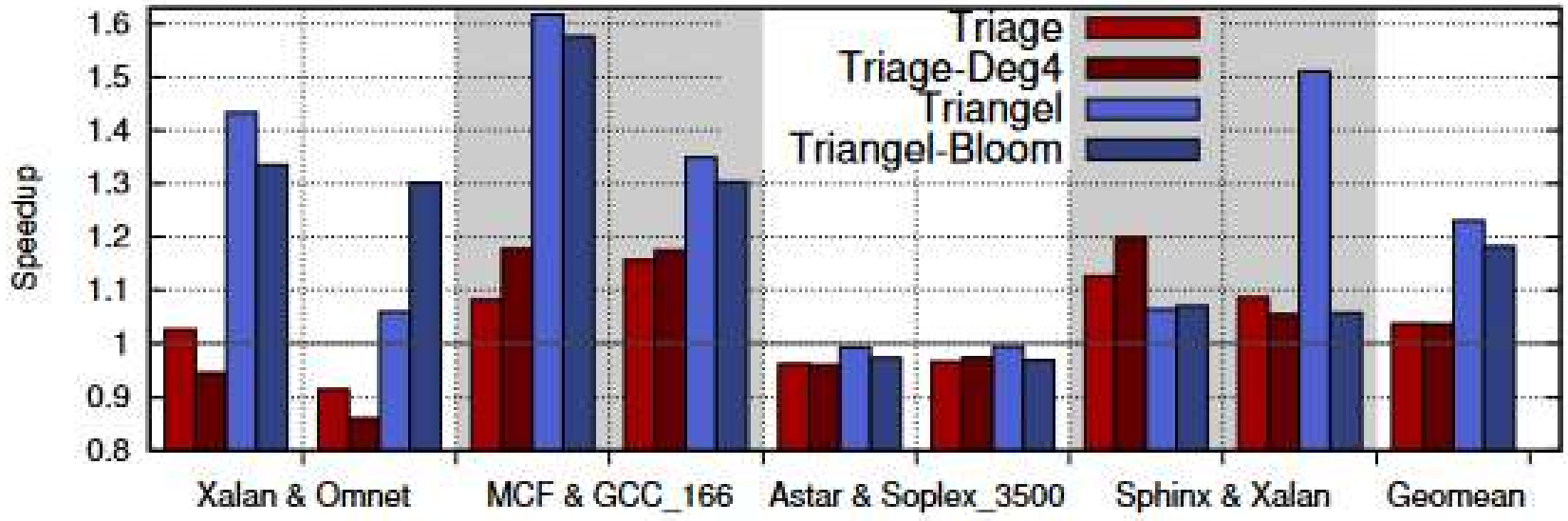


Fig. 16: Multiprogrammed-workload speedup. Workloads paired and prefetched simultaneously on two cores.

Backup Results: Triangel knows when to switch itself off

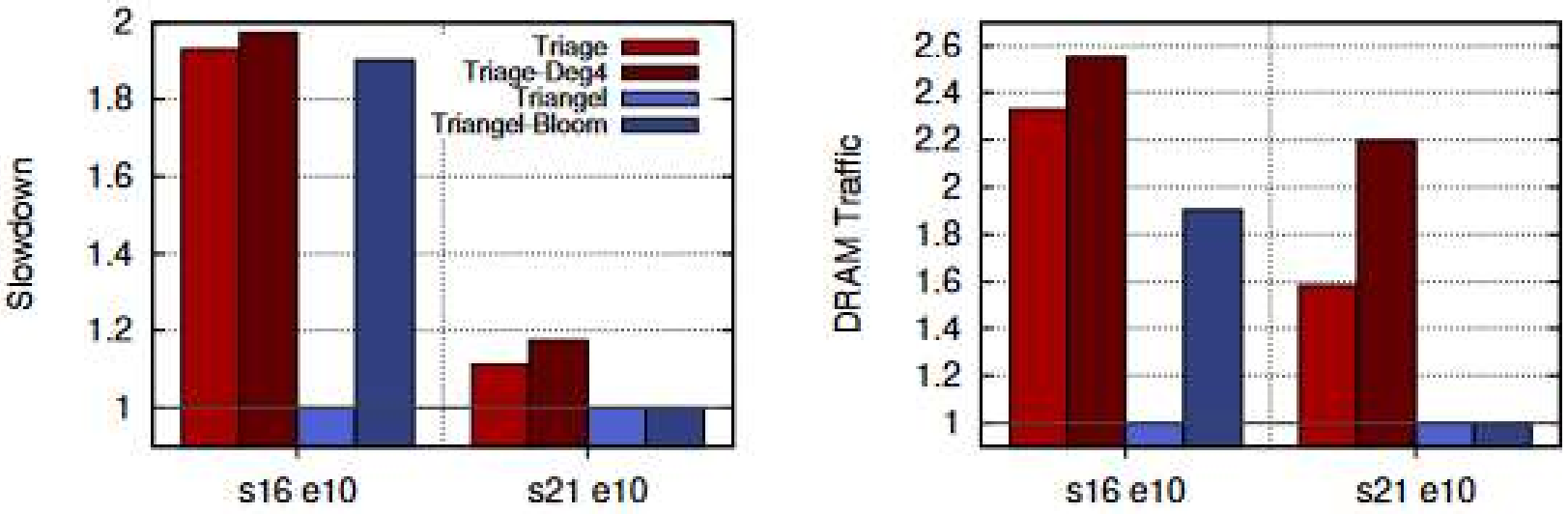


Fig. 17: *Slowdown* and DRAM traffic for Graph500 search.

THE UNIVERSITY OF EDINBURGH

Queen Mary University of London

Backup Slides: Triage's LUT indexing is inaccurate because it doesn't have tags, and upper address bits of physical addresses don't see enough sharing.

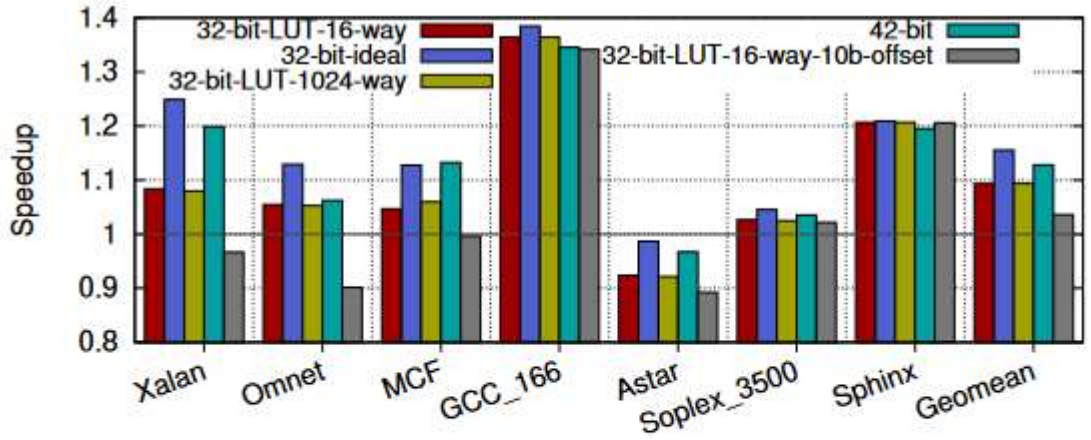


Fig. 18: Performance of Triage with different Markov-table entry sizes, and Lookup-Table configurations in cases where the Markov-table entry is 32-bits and so requires lookup to reconstruct a full prefetch target (first is default).

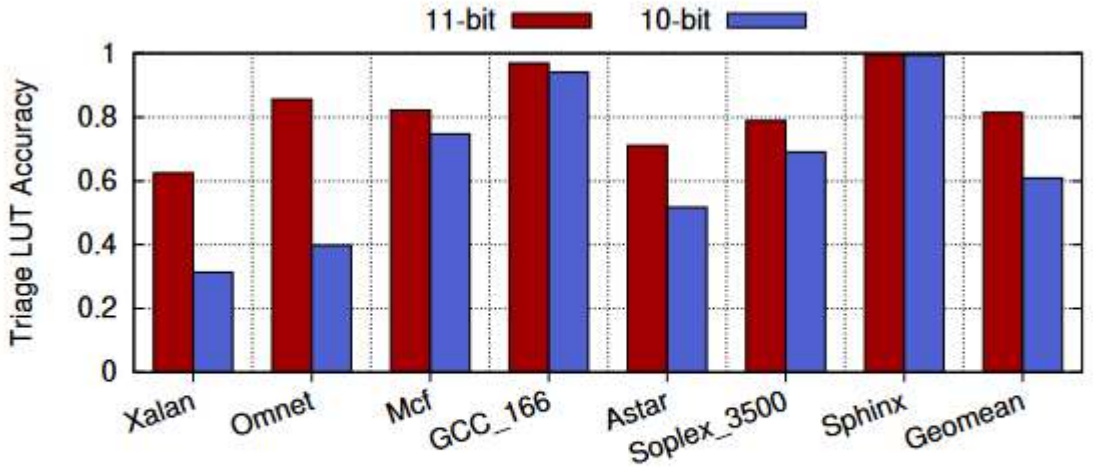


Fig. 19: Accuracy of Triage's LUT with varying number of offset bits (11-bit default). When capacity is exhausted, we see wrong prefetches. Triangel avoids its use.

THE UNIVERSITY OF EDINBURGH

Queen Mary University of London