Scalable Cache Miss Handling For High MLP

James Tuck, Luis Ceze, and Josep Torrellas
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

i-acoma group

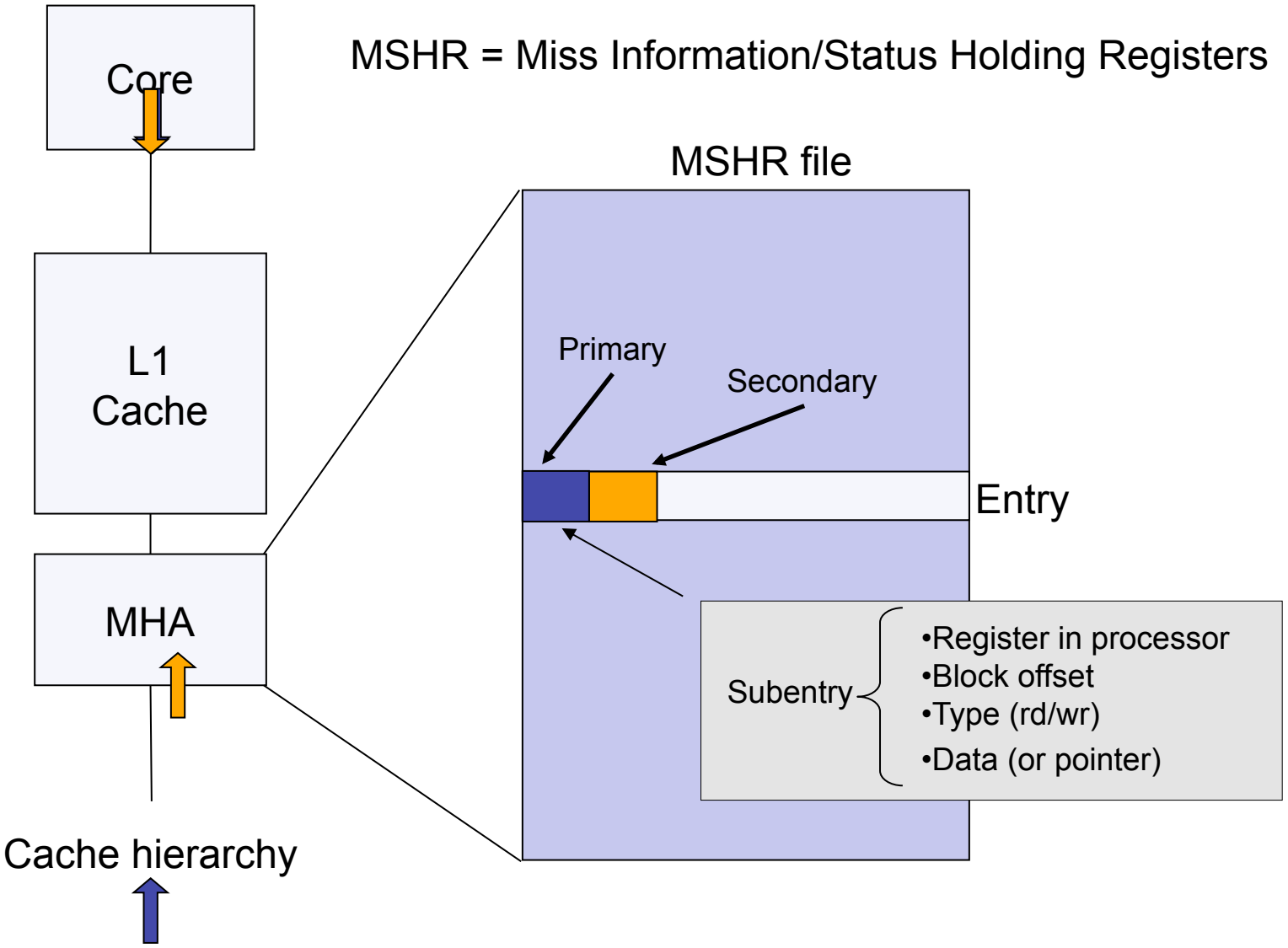
Introduction

- Checkpointed processors are promising superscalar architectures
  - Runahead, CPR, Out-of-order commit, CFP, CAVA
- Deliver high numbers of in-flight instructions
  - Effectively hide long memory latencies
  - Dramatically increase Memory-Level Parallelism (MLP)

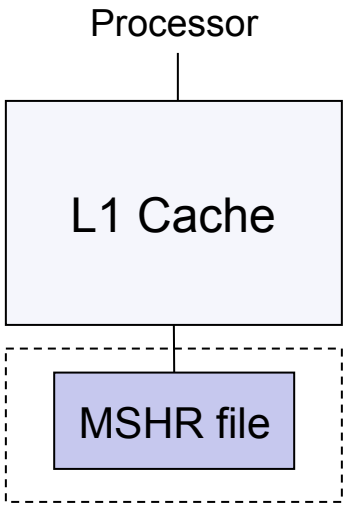
→ Current miss handling structures are woefully under-designed!

Miss Handling Architecture (MHA)

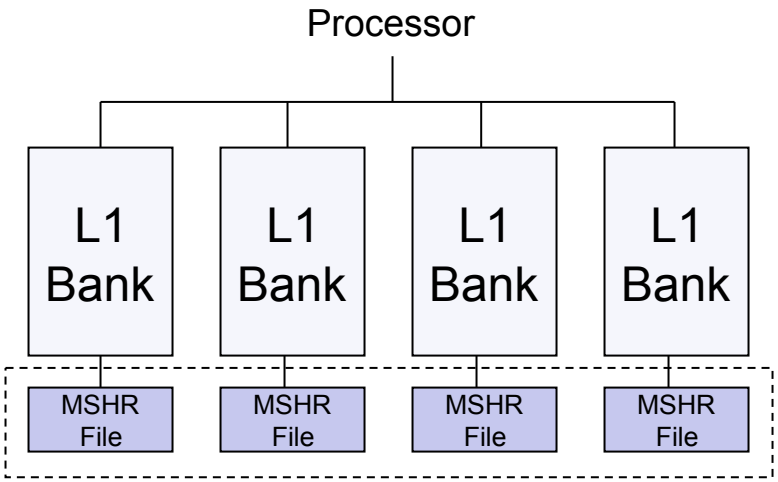
Kroft, ISCA'81
Scheurich & Dubois, SC'88
Farkas & Jouppi, ISCA'94



Background on MHA



Unified MHA



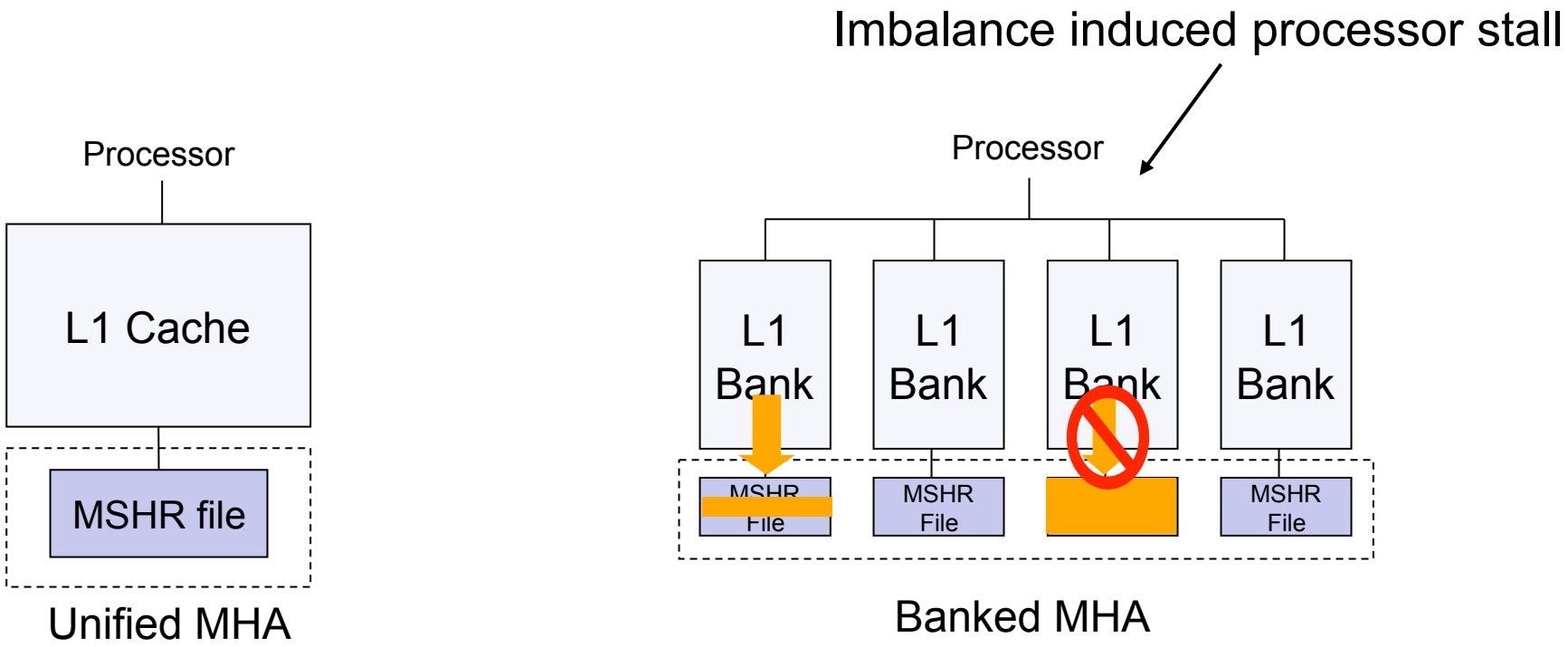
Banked MHA

- Kroft [ISCA'81] proposed first non-blocking cache
  - MSHR file

- Sohi and Franklin [ISCA'91]
  - Evaluated cache bandwidth
  - MSHR file banked with cache

Motivation

- MHAs must support many more misses
- Brute force approach will not do

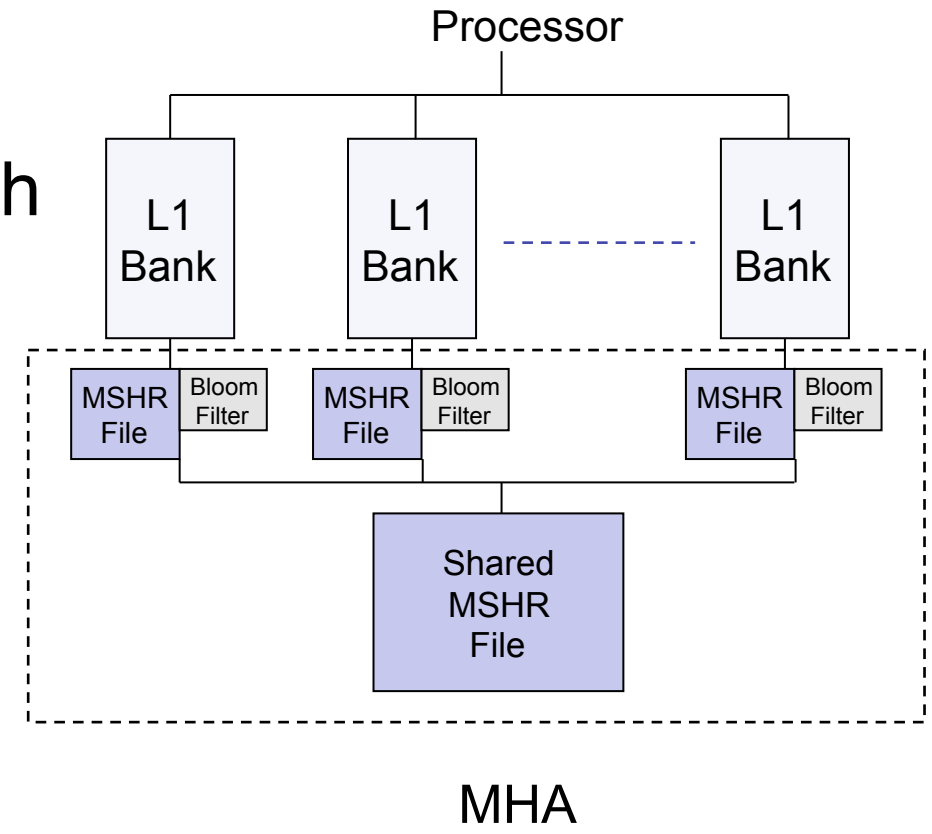


Centralized design has low bandwidth

Banking may cause access imbalance (and lockup) or inefficient area usage

Proposal: Hierarchical MHA

- A small per-bank MSHR file with Bloom filter
  - High bandwidth
- A larger, Shared MSHR file
  - High effective capacity
  - Low lock-up time



Contributions

- Show that state-of-the-art designs are a significant bottleneck
- Propose a Hierarchical MHA to meet high MLP demands
- Thoroughly evaluate on Checkpointed processors with SMT and show
  - Over state-of-the-art, avg. speed-ups of 32% to 95%
  - Over large Unified design, avg. speed-ups of 1% to 18%
  - Performs close to unlimited size MHA

Why not reuse load/store queue state?

- High MLP: need state in LSQ and in MHA
- Could simplify MHA by leveraging complex LSQ
  - Allocate MSHR on primary miss
  - Keep all secondary miss state in LSQ
- Disadvantage of leveraging LSQ
  - Induces additional global searches in the LSQ from the cache side
    - Searches would use MSHR ID or line address---not word address
  - Some checkpointed microarchitectures speculatively retire instructions and discard LSQ state
  - LSQ is timing critical: better not put restrictions on it

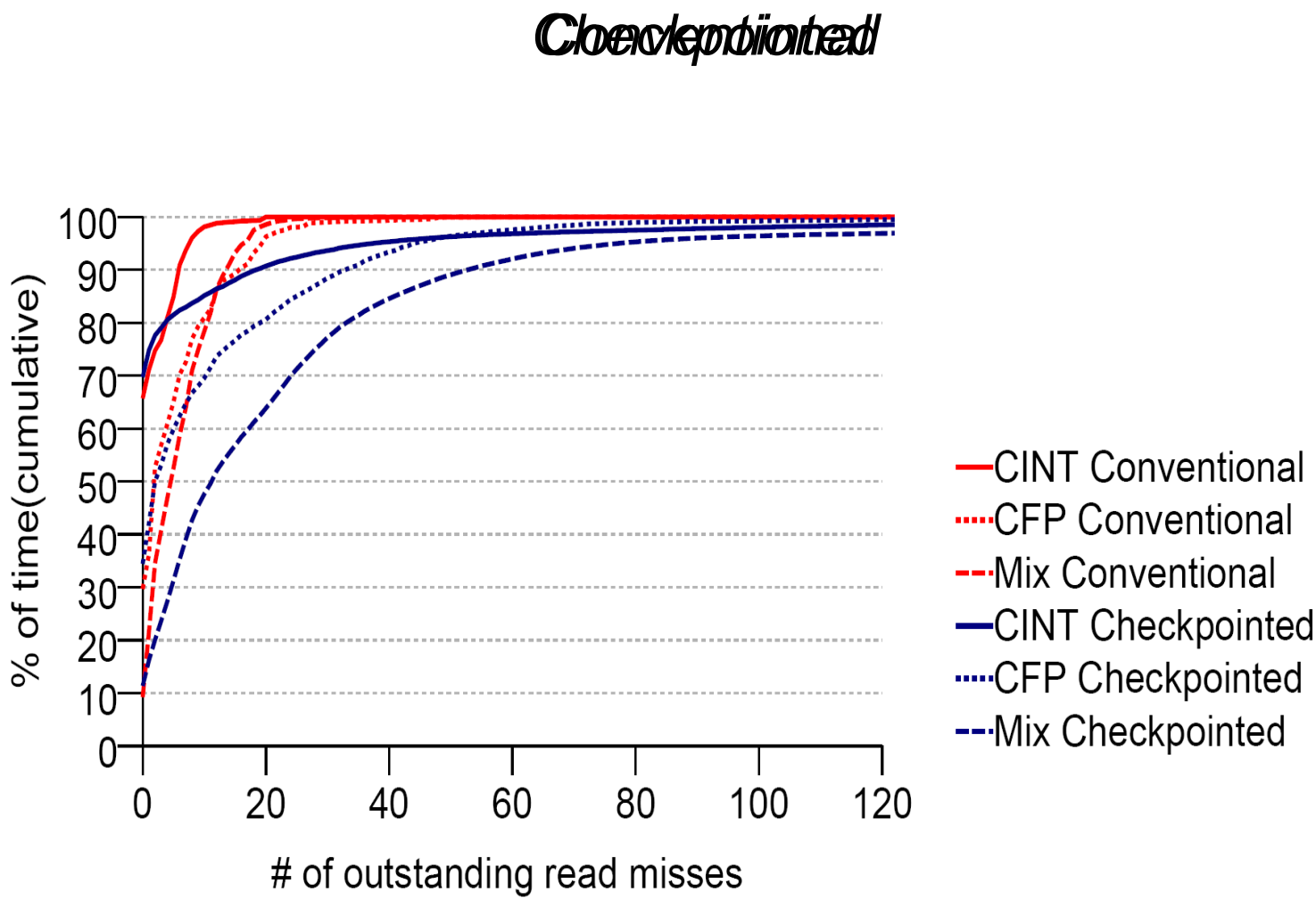
→ We keep primary and secondary miss info in MHA and rely on no specific LSQ design

Outline

- Requirements of new MHAs
- Hierarchical MHA
- Experimental setup and evaluation

Requirements for the new MHAs

- High capacity



Requirements for the new MHAs

- High capacity
- High bandwidth
  - Average increase of 30%

i-acoma group

Requirements for the new MHAs

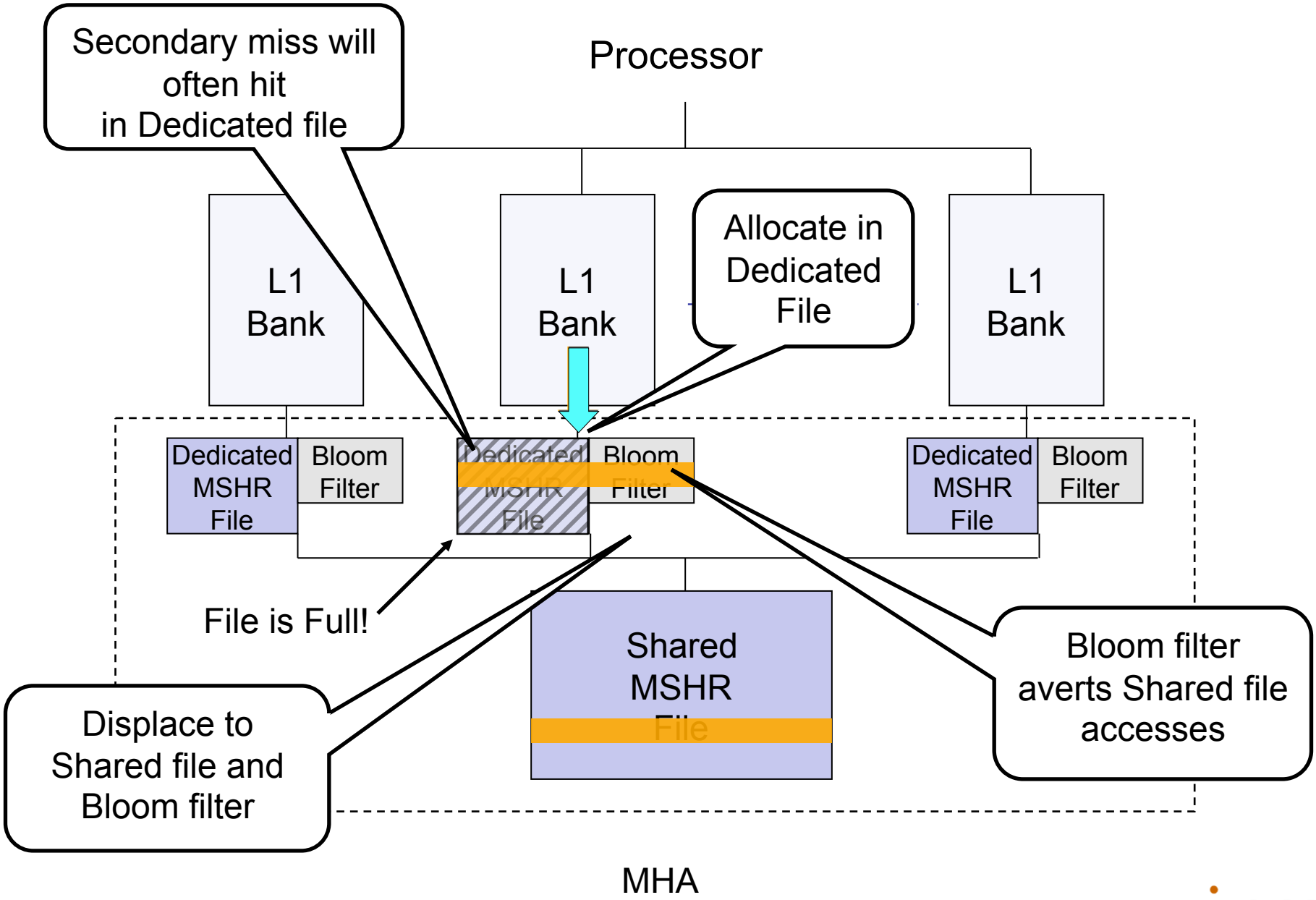
- High capacity
- High bandwidth
  - Average increase of 30%
- Banked MHAs may suffer from access imbalance lockups
  - From 15% to 23% slow down
- Need many entries and subentries
  - 32 Entries (primary misses)
  - 16 to 32 subentries (secondary misses)

→ These are our design goals

Outline

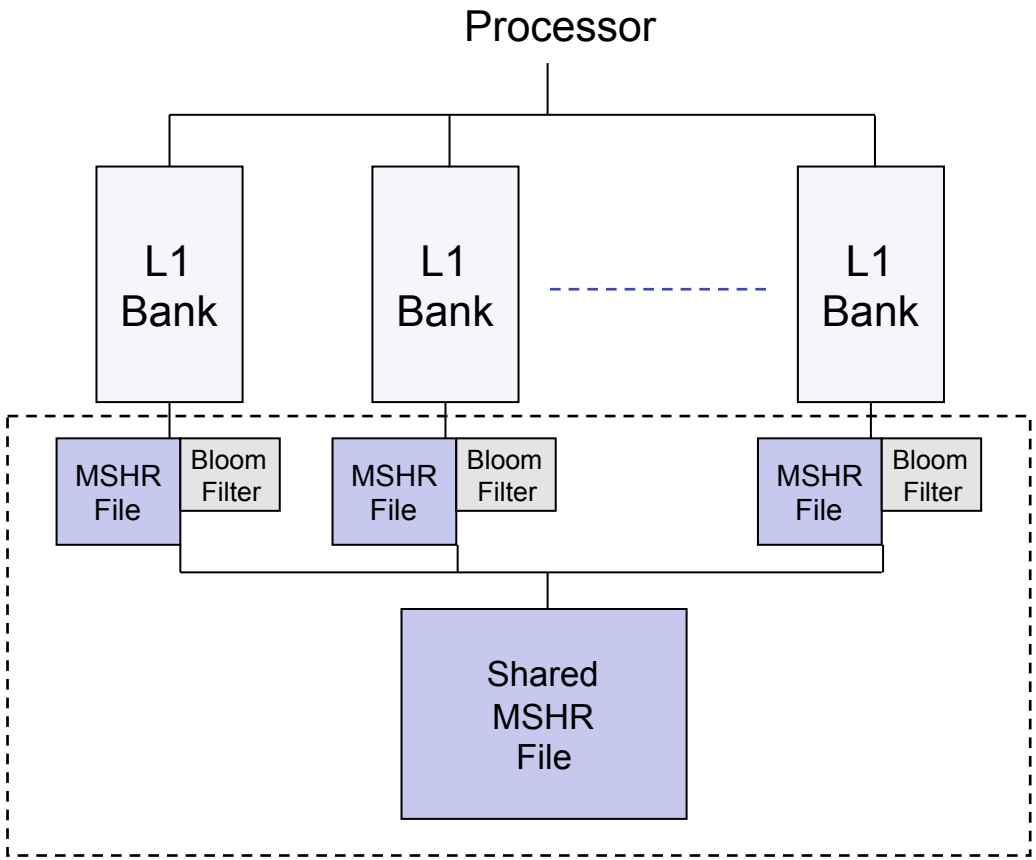
- Requirements of new MHAs
- Hierarchical MHA
- Experimental setup and evaluation

Hierarchical MHA



Hierarchical meets design goals

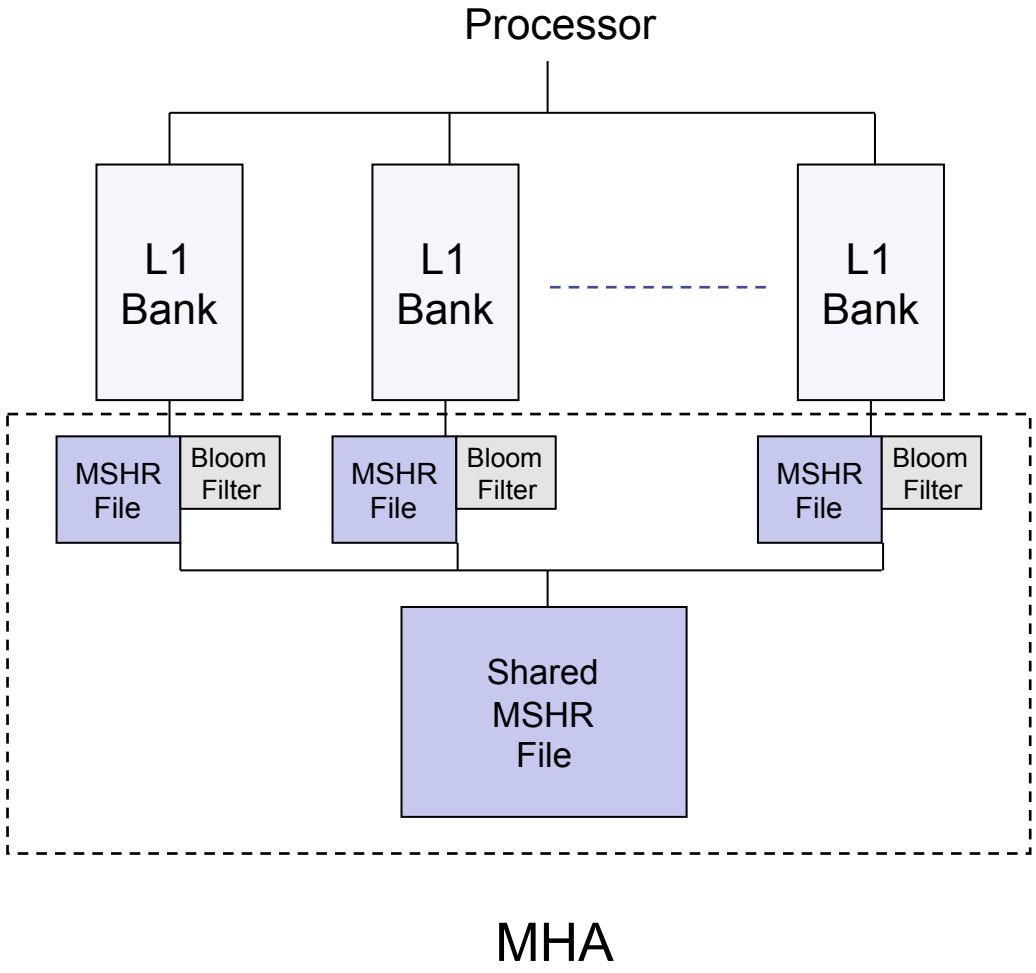
- Infrequent L1 lock-up while using MHA area efficiently
  - Use Shared file for displacements
- High bandwidth
  - Per-bank Dedicated file
  - Allocate in Dedicated file
    - Locality ensures it is in the Dedicated file
  - Bloom filter for Shared file
    - Averts most **useless** accesses to Shared file
    - Prevents a bottleneck at the Shared file



MHA

Overall organization and timing

- Dedicated file
  - Small and fully pipelined
  - Few entries and subentries
- Per bank Bloom filter
  - Accessed in parallel with Dedicated file
  - No false negatives
- Shared file
  - Highly associative and unpipelined
  - Contains many entries and subentries



Outline

- Requirements of new MHAs
- Hierarchical MHA
- Experimental setup and evaluation

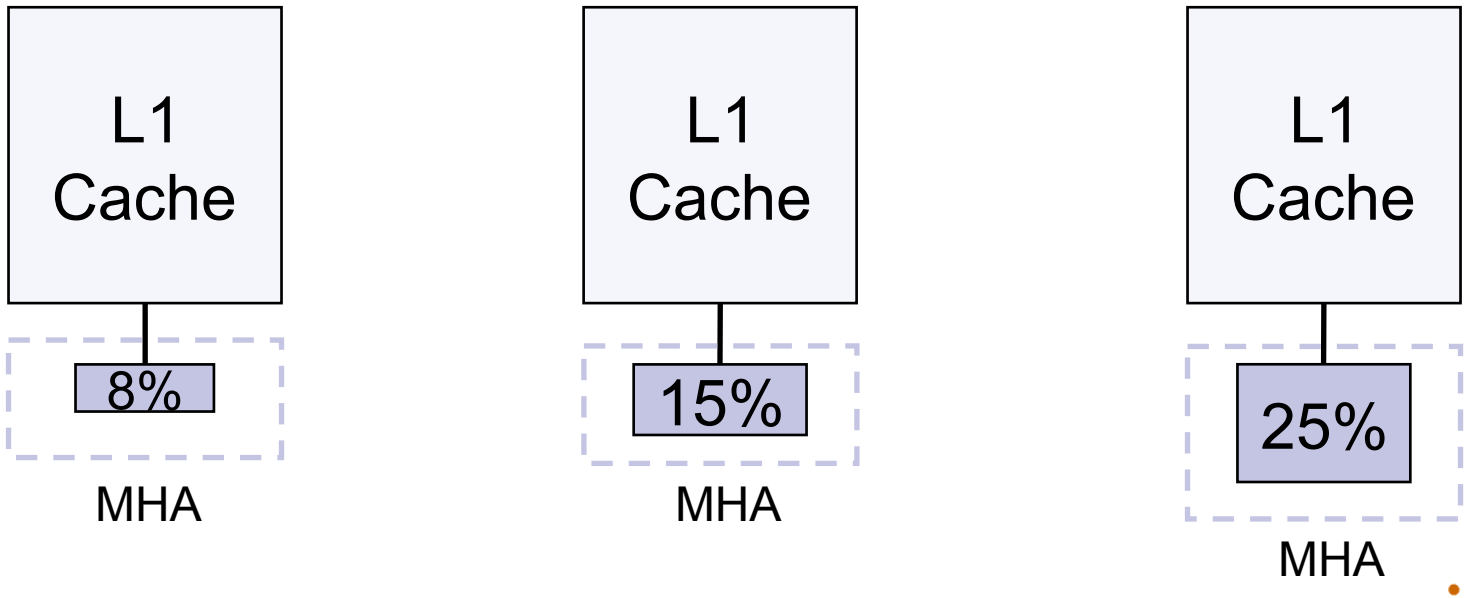
Experimental setup

- 5 GHz processor
- 5-issue, SMT with 2 contexts
  - *Conventional*
  - *Checkpointed*
  - *LargeWindow* (2K entry ROB)
- 32 KB L1 Data Cache
  - 8 banks, 2-way, 64B line, 3 cycle access, 1 port
- Memory bus bandwidth: 15 GB/s
- Workloads: CINT, CFP, Mix

- SESC simulator (sesc.sourceforge.net)

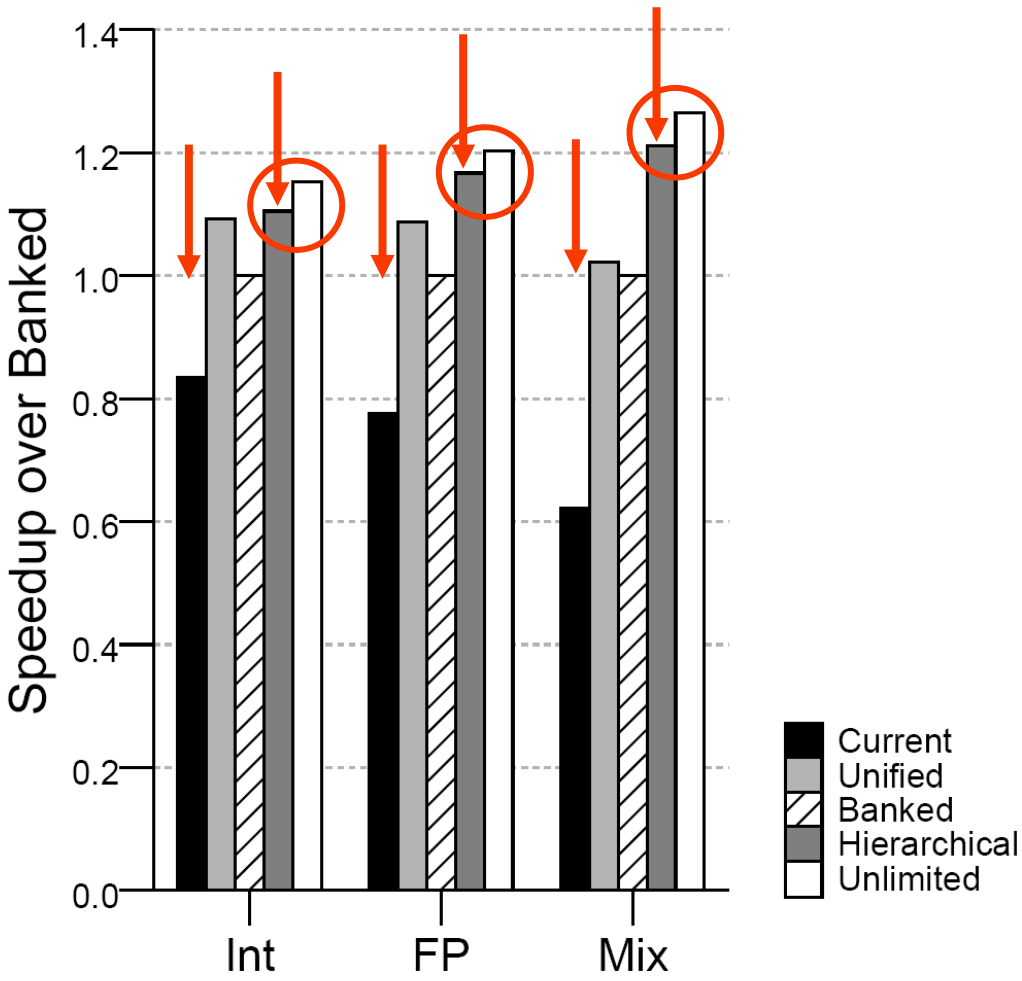
Compare MHAs with the same area

- 8%, 15%, and 25% of L1 cache area
  - Area estimated using CACTI 4.1
  - MSHR structures are fully associative
- Unified, Banked, and Hierarchical at each area
- Current: 8 misses like Pentium 4



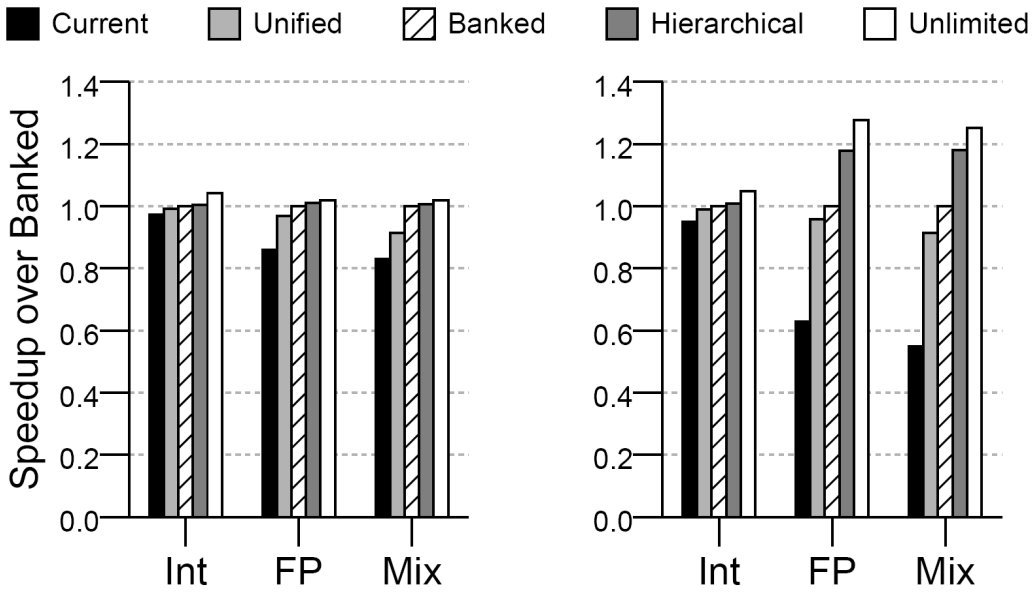
Performance at 15% area for *Checkpointed*

- Current is much worse
- Hierarchical is better than Unified and Banked
  - 1 to 18% over Unified
  - 10 to 21% over Banked
- Hierarchical is very close to Unlimited



Performance at 15% area for other processors

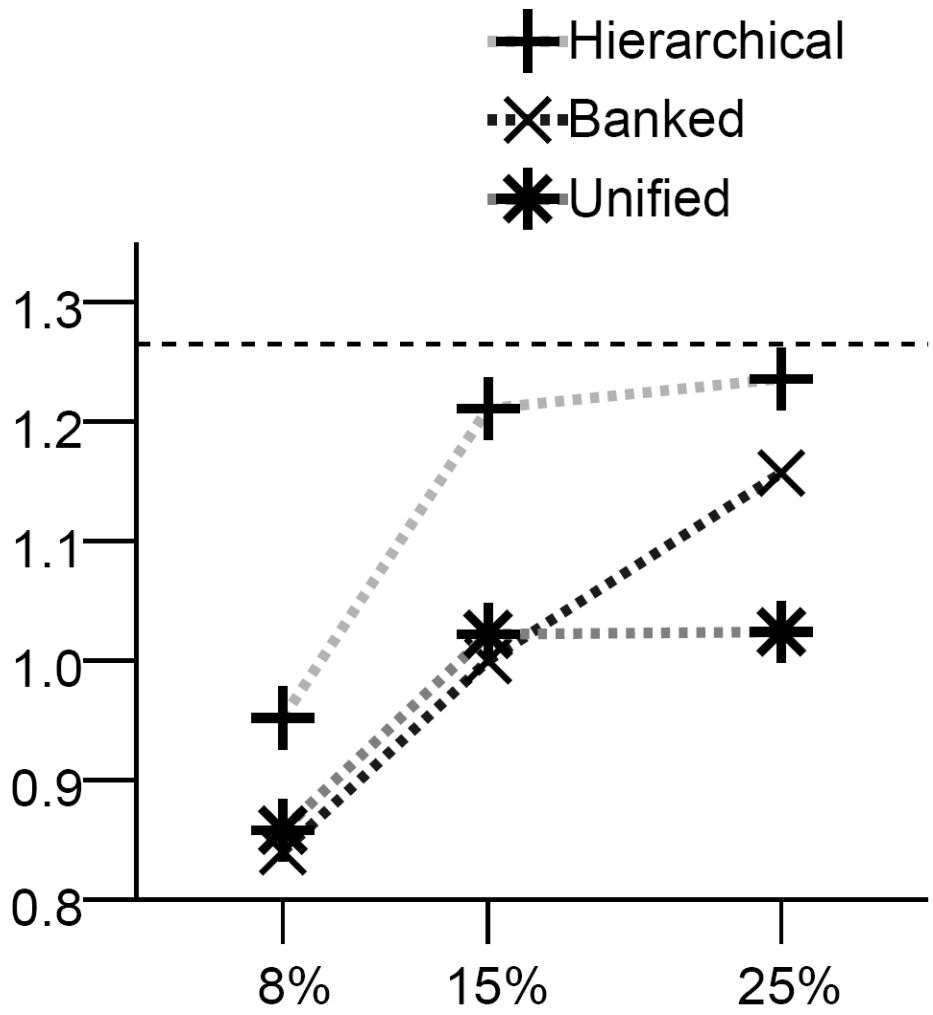
- *Conventional*
  - Less gain across the board
- *LargeWindow*
  - Current bottlenecks the processor
  - Hierarchical outperforms the rest
- Other architectures can leverage this design



Conventional *LargeWindow*

Performance at different area points

- *Checkpointed* running Mixes
- Unified saturates at 15%
- Banked continues to increase as it scales up
- Hierarchical is most efficient for these areas



Characterization

- Bloom filter averts majority of Shared file accesses
  - On average, from 89% to 95%
- Most secondary misses hit in the Dedicated file
- Reasons for displacing an entry from Dedicated
  - No free subentries: 18% to 40%
  - No free entries: 60% to 82%

Conclusions

- State-of-the-art MHA designs are a large bottleneck
  - Hierarchical speeds-up 32% to 95% over state-of-the-art
- Brute force Unified & Banked designs are suboptimal
  - Hierarchical speeds-up 1% to 18% over Unified
  - Hierarchical speeds-up 10% to 21% over Banked
- Hierarchical performs best over a range of areas
- Additional complexity of Hierarchical is reasonable

Questions?

Scalable Cache Miss Handling For High MLP

James Tuck, Luis Ceze, and Josep Torrellas

University of Illinois at Urbana-Champaign

http://iacoma.cs.uiuc.edu

i-acoma group