Large-Reach Memory Management Unit Caches

Abhishek Bhattacharjee

Rutgers University

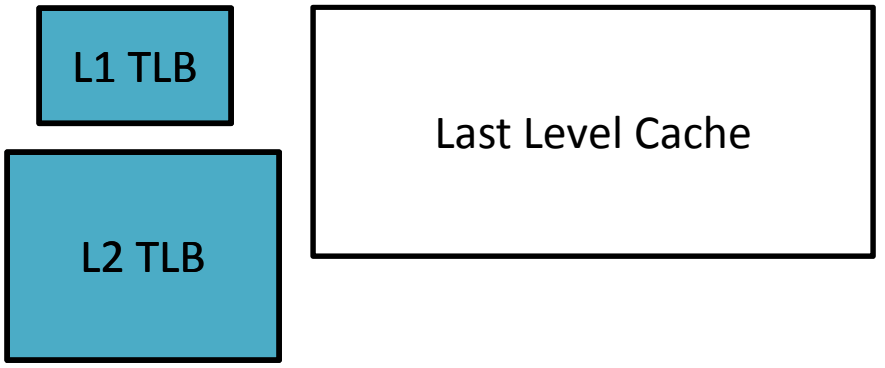
International Symposium on Microarchitecture-46

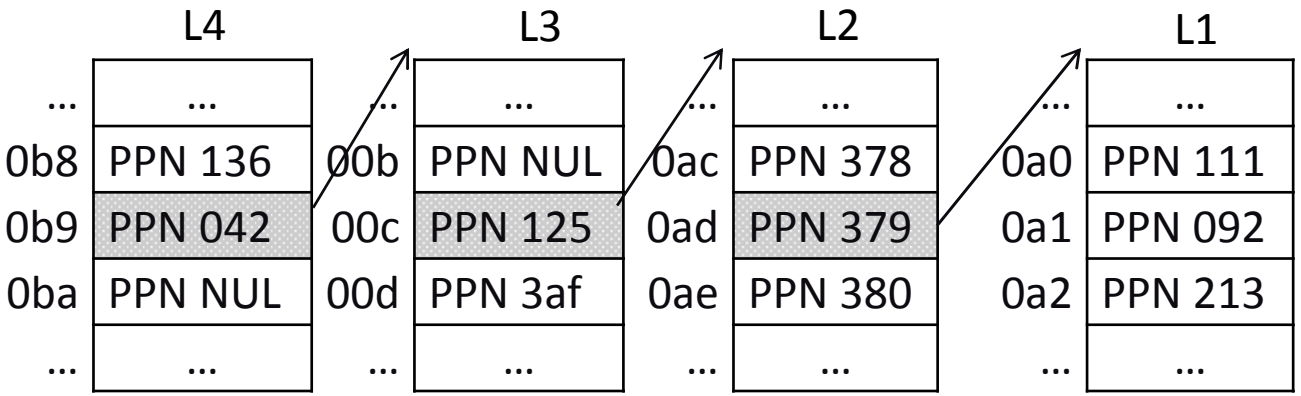
December 2013

Address Translation Primer

On a typical TLB miss:

- x86: 1-4 memory references
- ARM: 1-2 memory references
- Sparc: 1-2 memory references

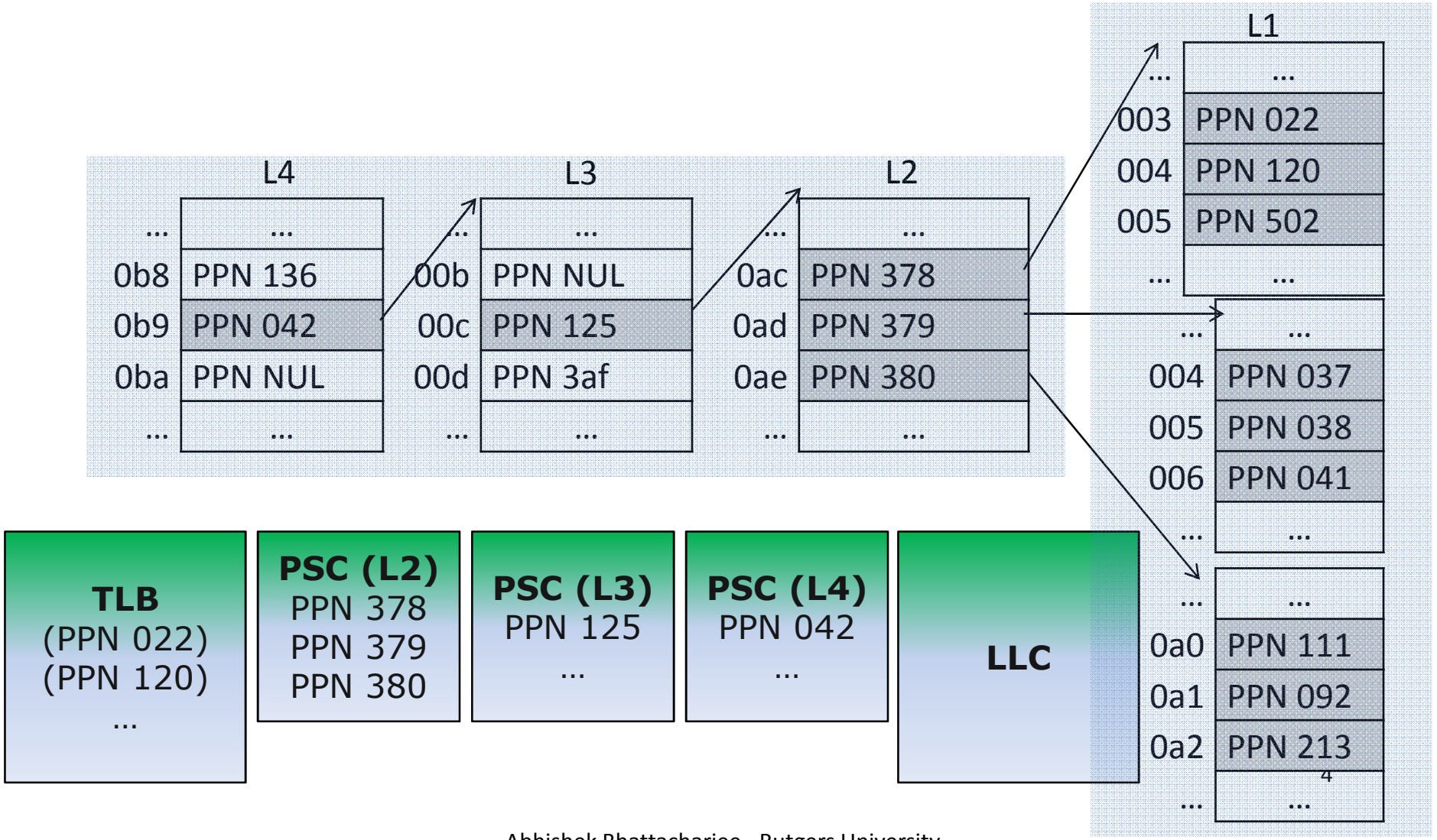




Address Translation Performance Impact

- Address translation performance overhead – 10-15%
  - Clark & Emer [Trans. On Comp. Sys. 1985]
  - Talluri & Hill [ASPLOS 1994]
  - Barr, Cox & Rixner [ISCA 2011]
- Emerging software trends
  - Virtualization 2D walks – 89% overheads [Bhargava et al., ASPLOS 2008]
  - Big data – 5-85% overheads [Basu, Swift, and Hill, ISCA 2012]
- Emerging hardware trends
  - LLC capacity to TLB capacity ratios increasing
  - Manycore/hyperthreading increases TLB and LLC PTE stress

Memory Management Unit Caches Primer

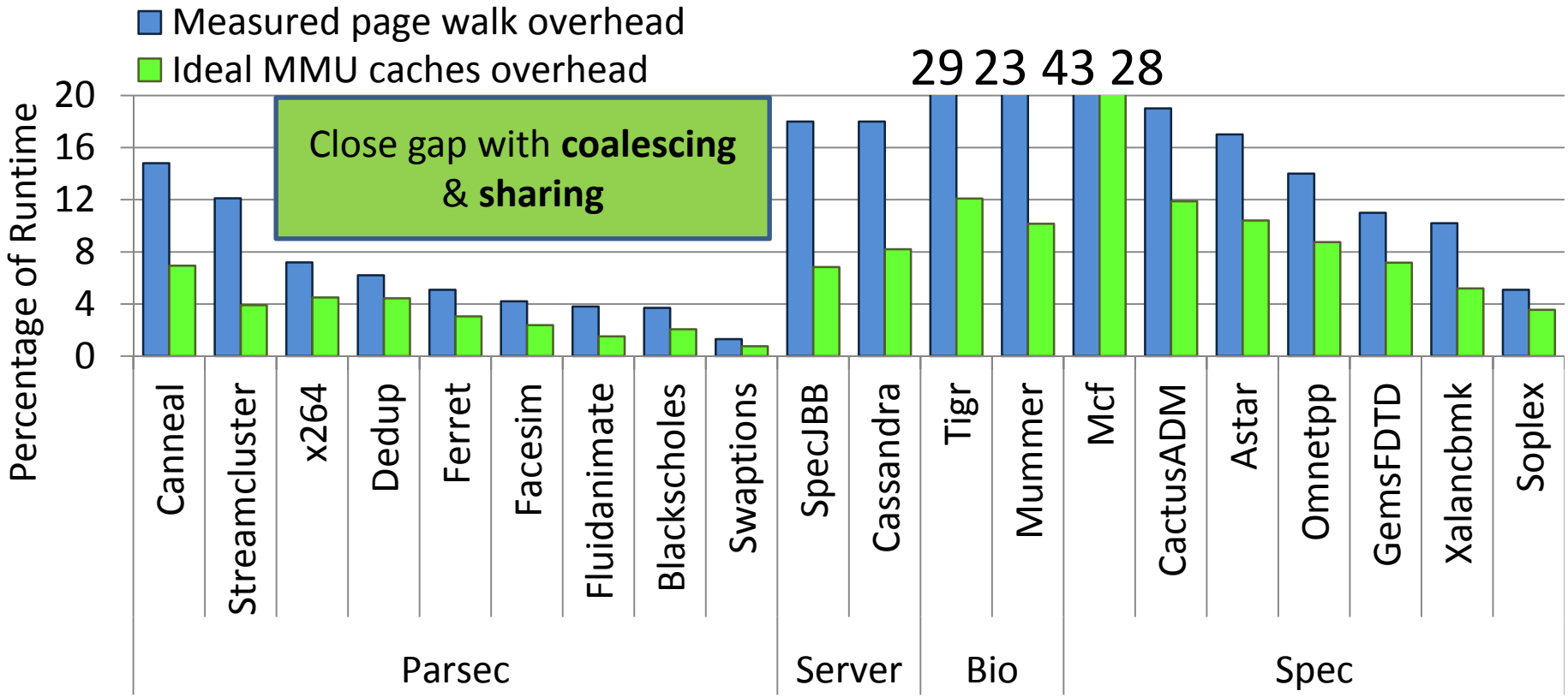


Memory Management Unit Caches

- Intel uses paging structure caches (PSCs)
  - Tagged with virtual address bits
  - Lookup separate PSCs in parallel for longest prefix match
- AMD uses page walk caches (PWCs)
  - Tagged with physical address of page table page
  - Lookups unified PWC sequentially
- Best case – single LLC reference per page table walk
- Past work – Barr, Cox, and Rixner, ISCA 2010

Real-System MMU Cache Performance

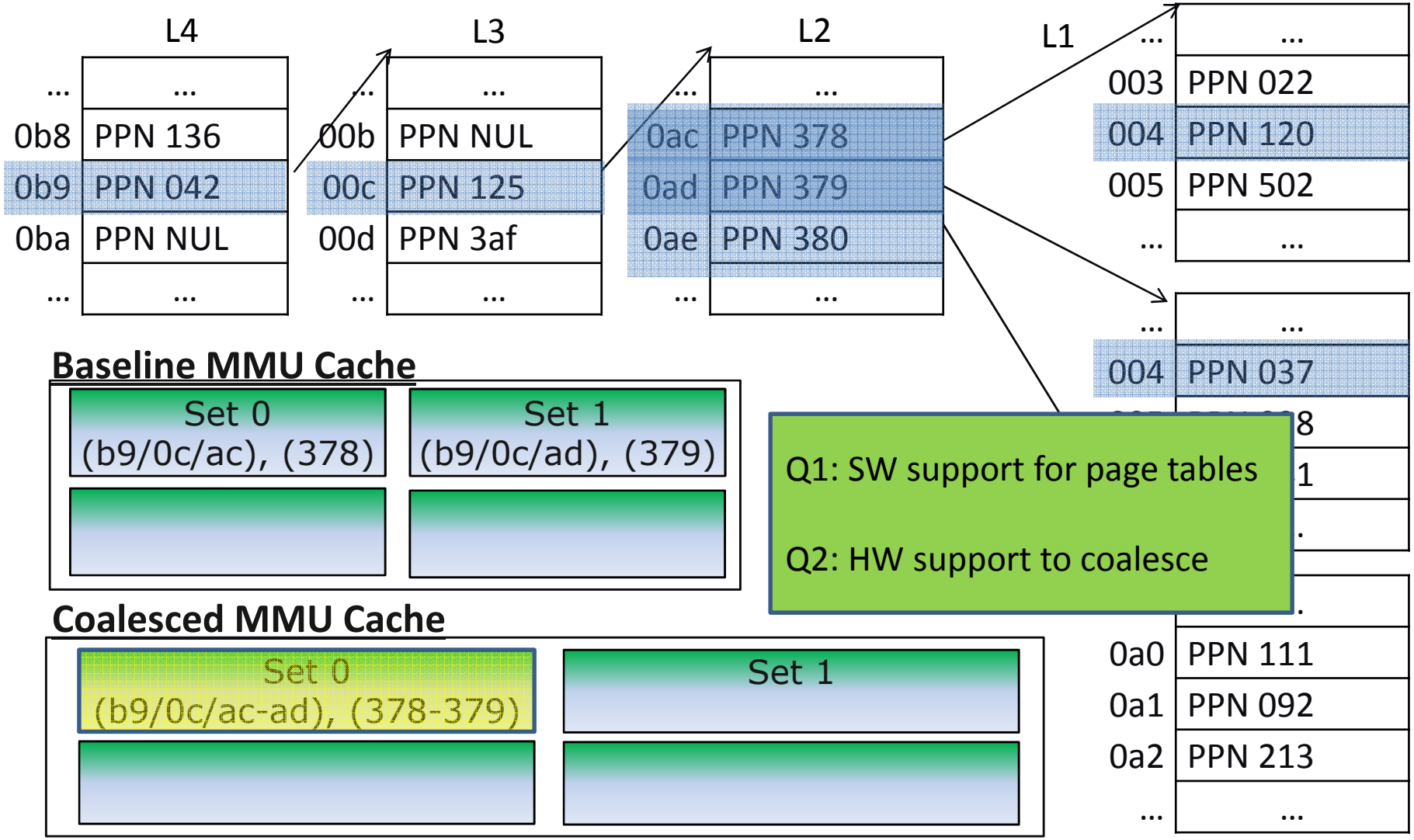
- Intel i7: 8 cores, 8GB memory, 512-entry L2 TLB, and 8MB LLC
- 32-entry L2 PSC, 4-entry L3 PSC, 2-entry L4 PSC



Outline of Techniques

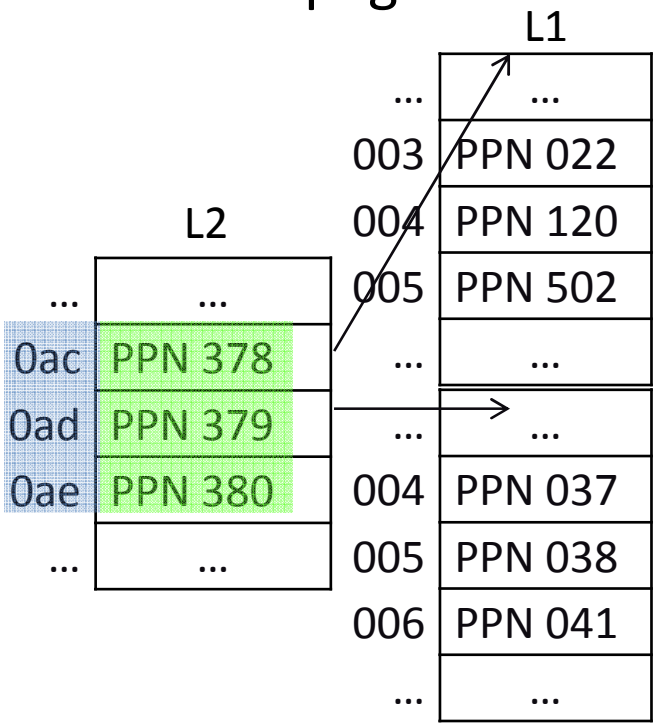
- MMU cache coalescing
  - High-level idea
  - Software support
  - Hardware support
- Shared MMU caches
  - High-level idea
  - Hardware support
- Combining approaches
  - Comparison with optimized TLBs

Coalescing: High-Level Idea



Coalescing: Software Support

- Consecutive L2 PTEs map to consecutive L1 page frames
- Program property – successive L2 PTE use (avg 35)
- OS allocation – successive L1 page frames

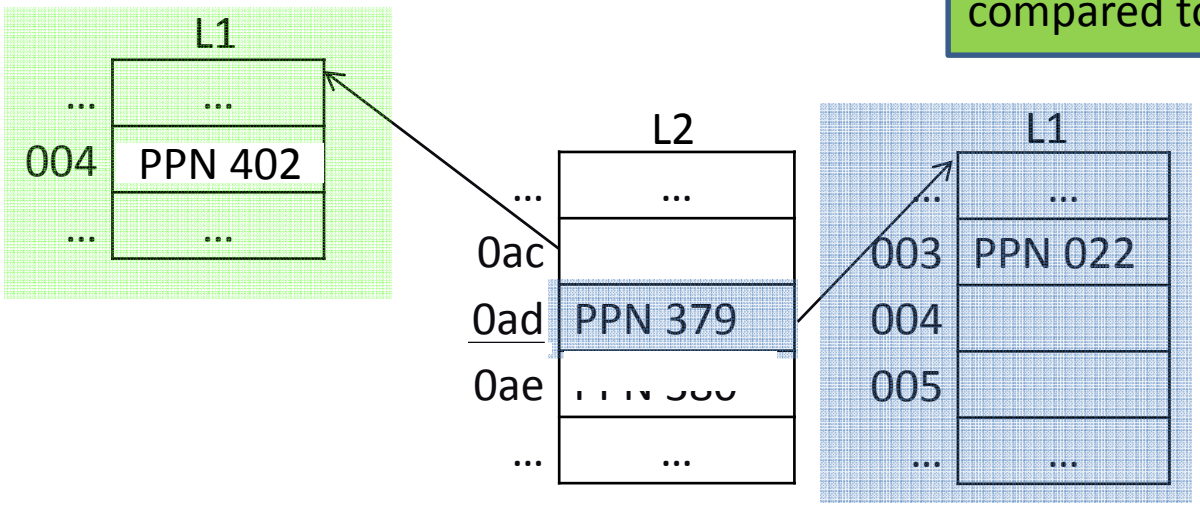


Coalescing: OS Page Table Allocation

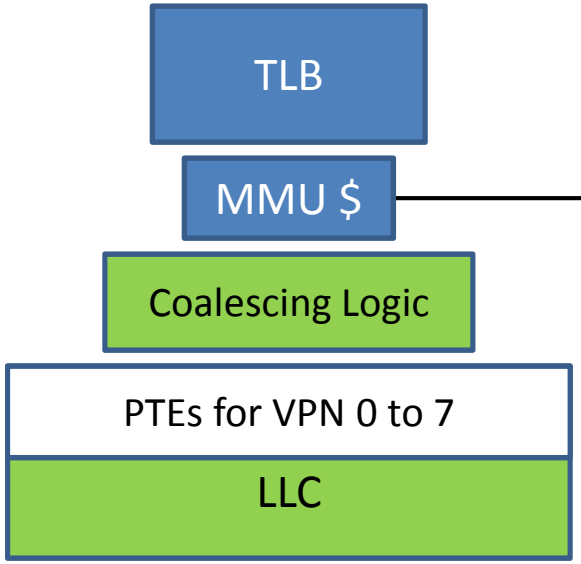
- Reservation-based superpages [Talluri & Hill, ASPLOS '94, Navarro et al., OSDI '02]
- Reserve *n* L1 page frames on L2 page fault
- Use reserved page if coalescible
- Relinquish unused L1 page frames after *m* faults

n of ± 1 and m of 4 ~ 0.1% memory overhead

10-15 mem refs on fault compared to 100s for fault code



Coalescing: Hardware Support



Lookup for PTE: 5 – 0b 1 0 1

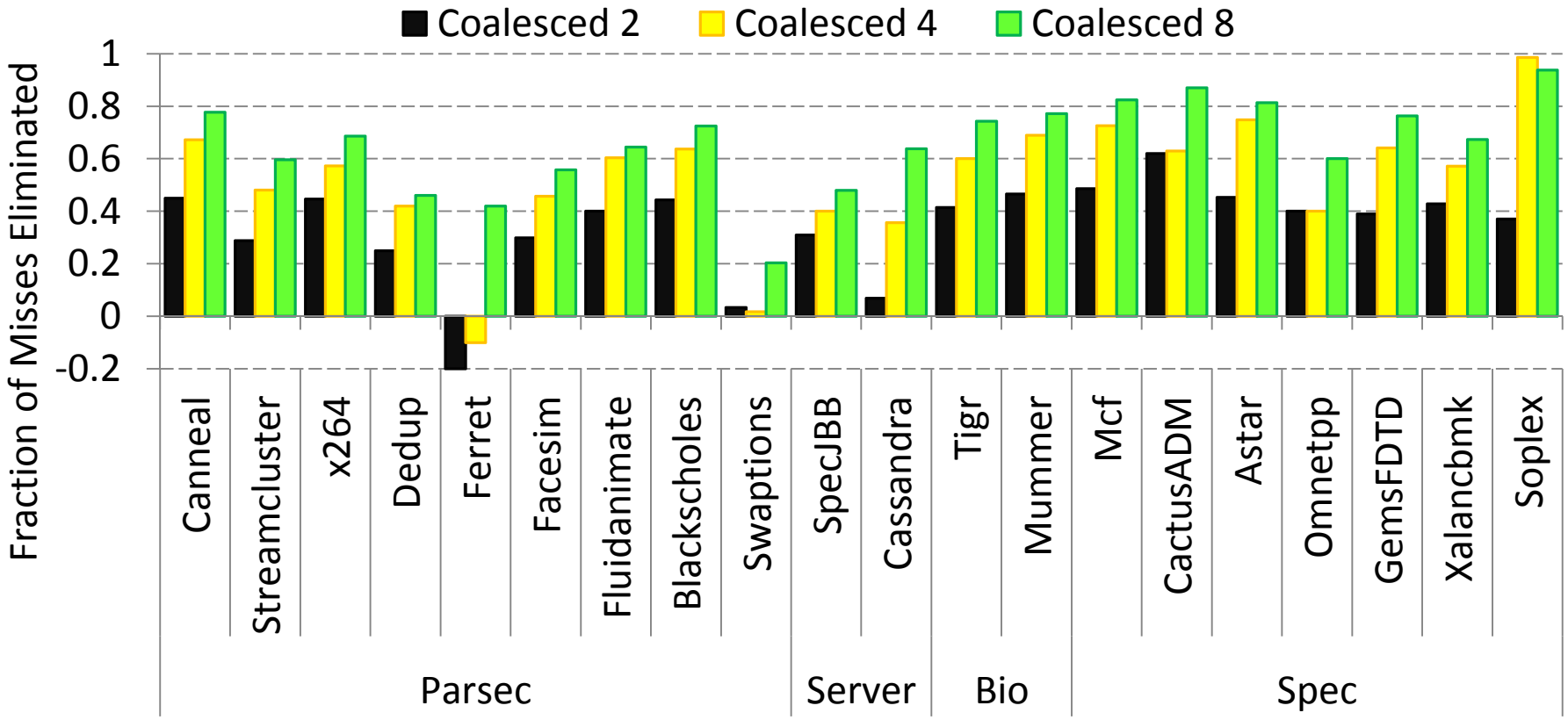
Tag	Valid	Valid	Attribute	Base physical
0b1	1	1	…	0b10110

Combinational logic to calculate physical page

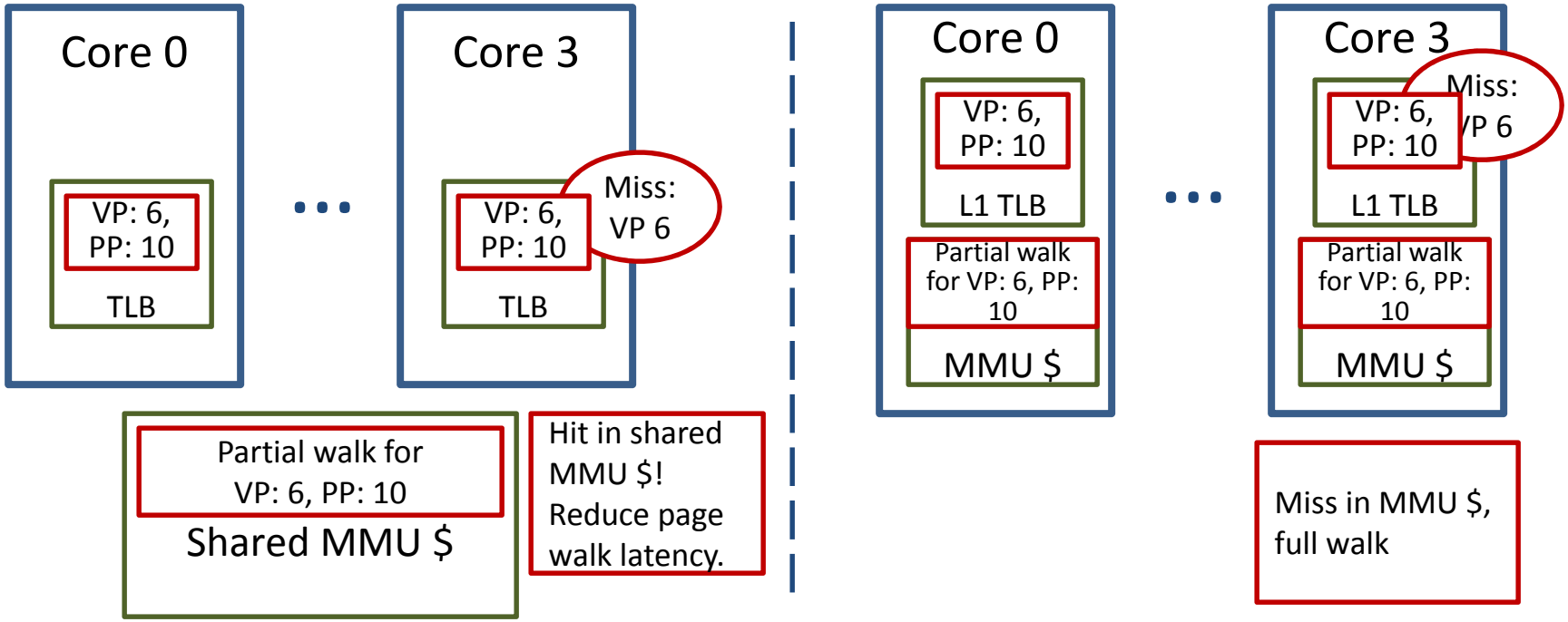
- Hardware complexity
  - No additional ports
  - Coalesce on fill to reduce overhead
  - No additional page walk memory references

Coalesced MMU Cache Miss Eliminations

- Trading off conflict misses for coalescing



Shared MMU Caches



- Parallel applications: reduces cross-core private MMU $ redundancy and cache more total page table entries
- Multiprogrammed sequential applications: better allocates MMU $ entries to different applications as needed

Shared MMU Cache Design Challenges

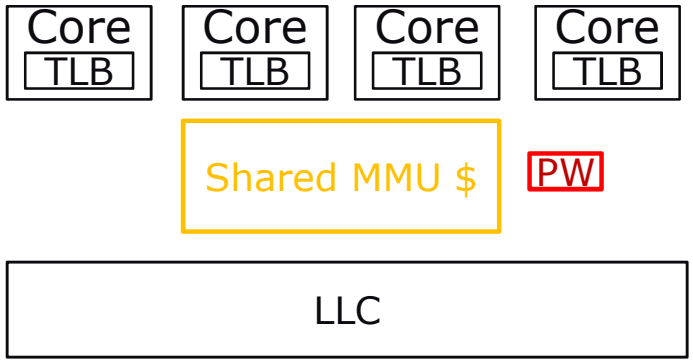
Higher access latencies (CACTI)?

Unlike TLBs, PTEs map different address ranges.

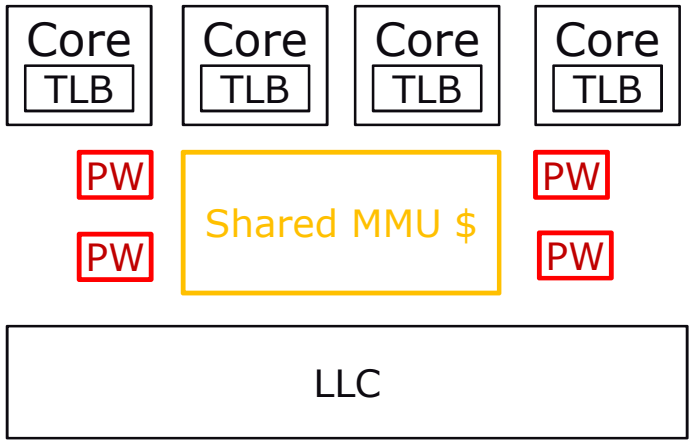
Interaction with TLB optimizations?

Multiple ports?

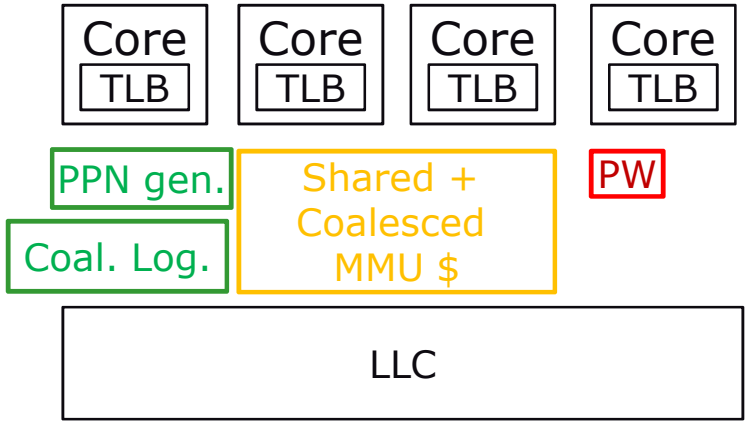
Shared MMU Cache



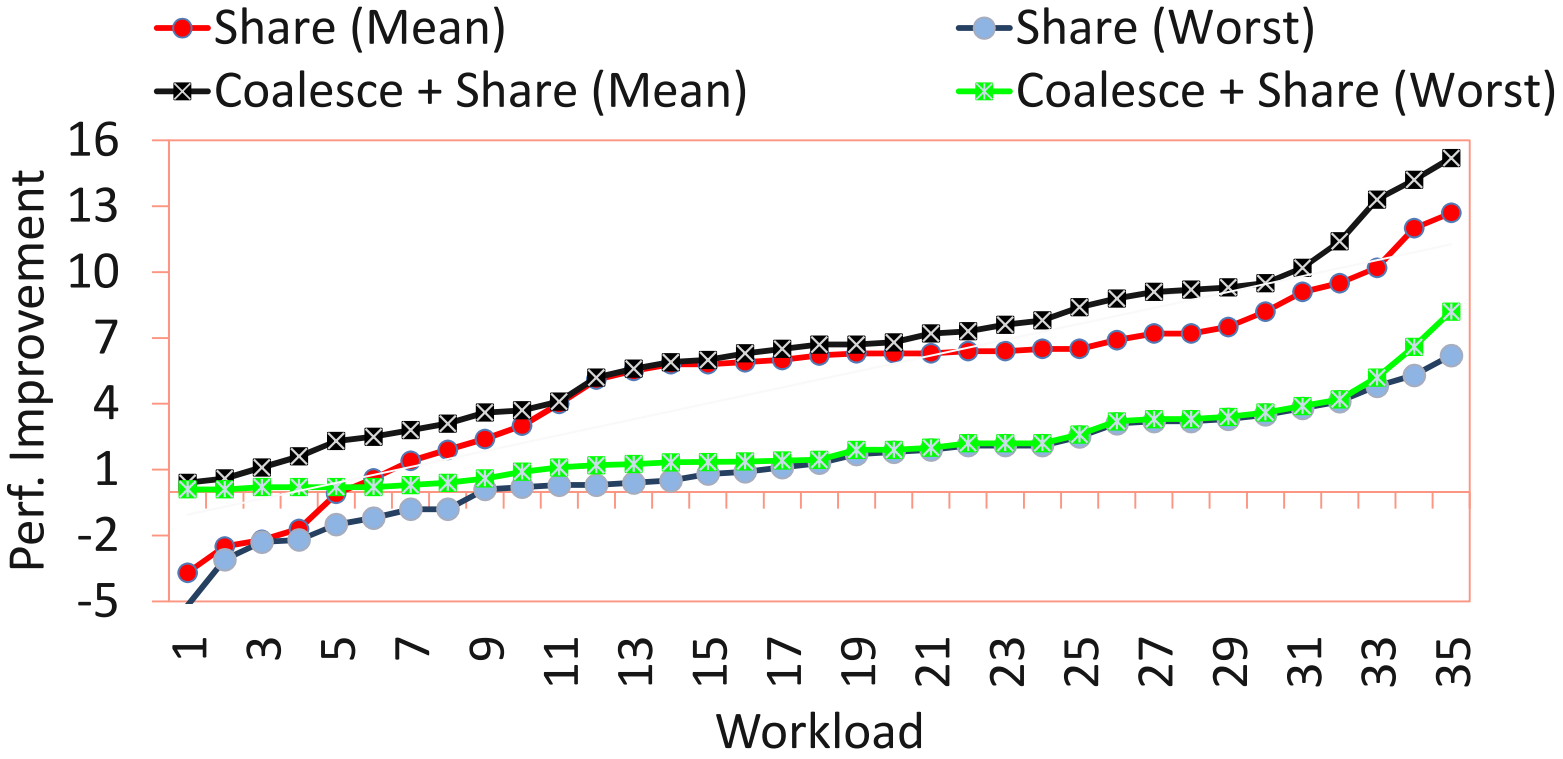
Shared MMU Cache + Multiple Page Walkers



Shared MMU Cache + Coalescing



Shared MMU Cache Results

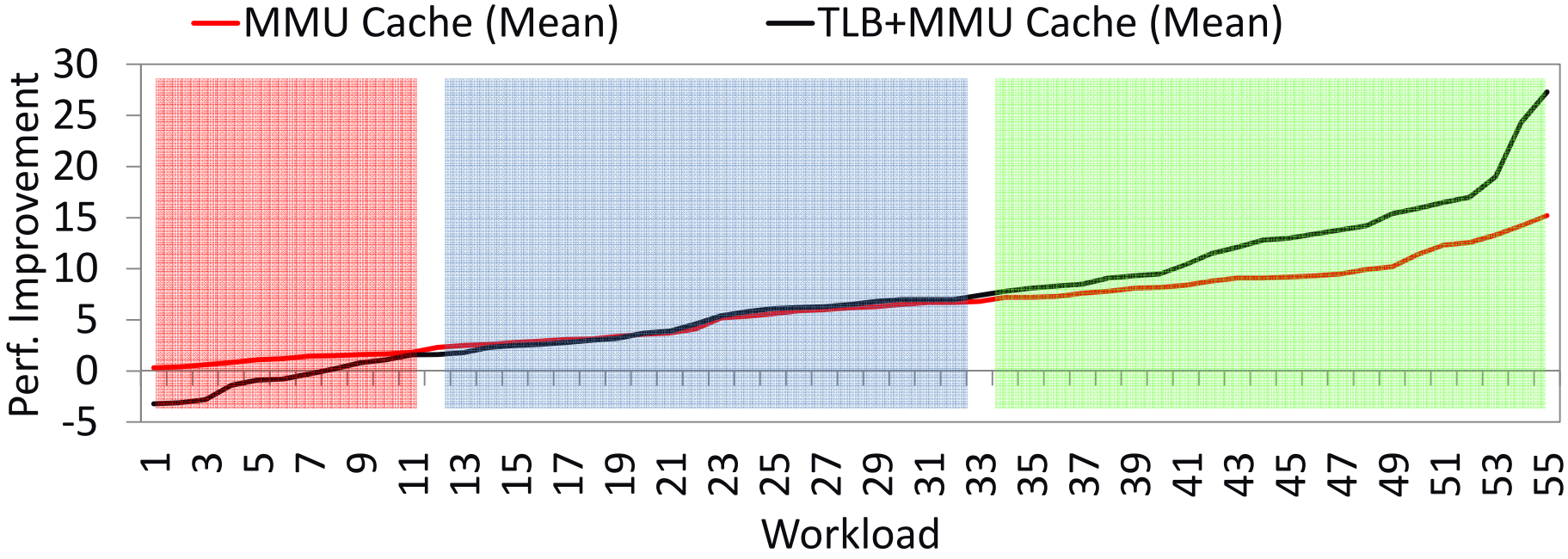


Coalescing + sharing more effective than sharing
Sharing hit rates usually outweigh higher access latency
Multiple ports and PTWs give 1-2% perf on 8-core CMP
Combined approach within 3-5% of ideal case (3x larger MMU cache)

Comparison to TLB Optimizations

Abhishek Bhattacharjee, Daniel Lustig, Margaret Martonosi, "Shared Last-Level TLBs for Chip Multiprocessors", HPCA '11

Binh Pham, Viswanathan Vaidyanathan, Aamer Jaleel, Abhishek Bhattacharjee, "CoLT: Coalesced Large-Reach TLBs", MICRO '12



Shared and coalesced MMU caches provide more predictable performance improvement across workloads

Conclusions

- MMU caches ignored but vitally important for emerging workloads
- Low overhead OS-level and hardware boosts performance
  - Average of 10-12% performance improvements
- Orthogonal design to TLB optimizations and large page support
- Open questions: virtualization? scaling?

Large-Reach Memory Management Unit Caches

Abhishek Bhattacharjee

Rutgers University

International Symposium on Microarchitecture-46

December 2013