IBM Research

Design and Implementation of the Blue Gene/P Snoop Filter

Valentina Salapura
Matthias Blumrich
Alan Gara

The 14th International Symposium on High-Performance Computer Architecture
HPCA '08, February 18, 2008

System optimization for the multicore era

- Scaling up the system performance
  - Increasing the number of processors on chip
- Keeping the communication cost between the processors low
  - Data sharing for unified memory space and shared address space
  - Coherence protocols implemented
- Coherence comes at a cost
  - Designing a coherent multiprocessor is one of the most challenging tasks
  - System performance is reduced due to coherence overhead
- Goals
  - Reduce performance loss due to coherence traffic
  - Reduce power consumption for high-power cache access

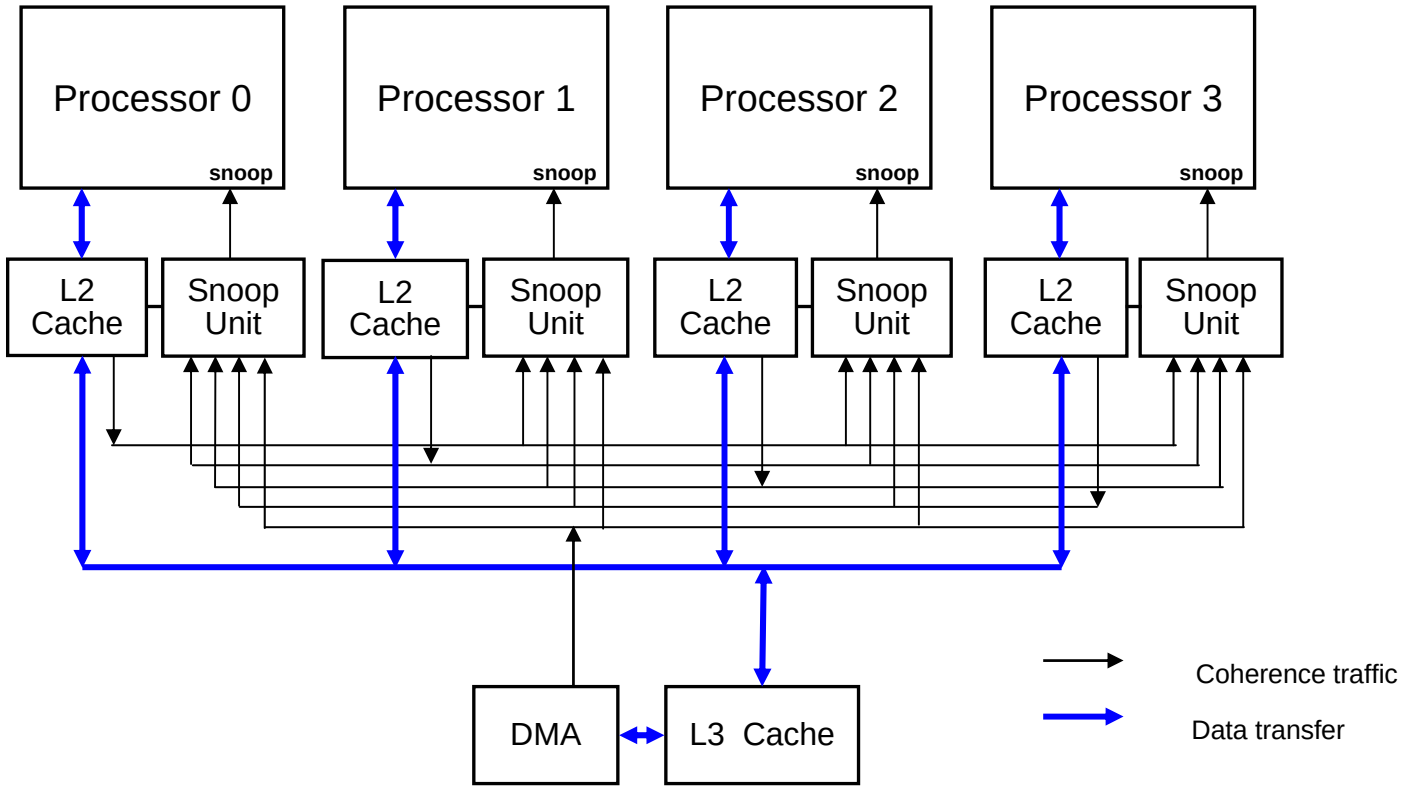
A new 4-way chip multiprocessor node design

- 4-way SMP based on PowerPC 450
  - Next-generation IBM embedded microprocessor
- PPC450 represents first IBM 4xx embedded processor with coherence support
  - Write through policy
  - Snoop on write requests to invalidate cache lines
- Internal architecture closely related to PPC440
  - Single copy of single ported L1 data cache tag array reduces area
  - Snoop traffic shares tag port with data cache access
- Design driven by technical and non-technical considerations
  - Area and power limit ability to increase cache subsystem complexity
  - NRE and time to market considerations limit

Coherence revisited

- Coherence establishes consistent, shared view of memory despite the presence of local caches
  - Accomplished by ensuring updates from one core are visible to readers on other cores
  - Invalidate cache contents in remote caches when locally modified
- Coherence is important for data sharing
  - Locks and shared data resources must be synchronized
  - Allow applications to efficiently share data
- Coherence represents overhead for non-shared data
  - But in properly tuned applications, most data is *not* shared
    - Cost of data transfer would diminish application performance
    - Many code transformations to increase locality of reference
  - Key goal: avoid paying for synchronizing unshared data
- Eliminate useless coherence actions
  - Increases performance (remove unnecessary lookups, reduced cache interference)
  - Reduces power (and energy)

Snoop filtering of unnecessary coherence requests



Multi-component snoop filtering

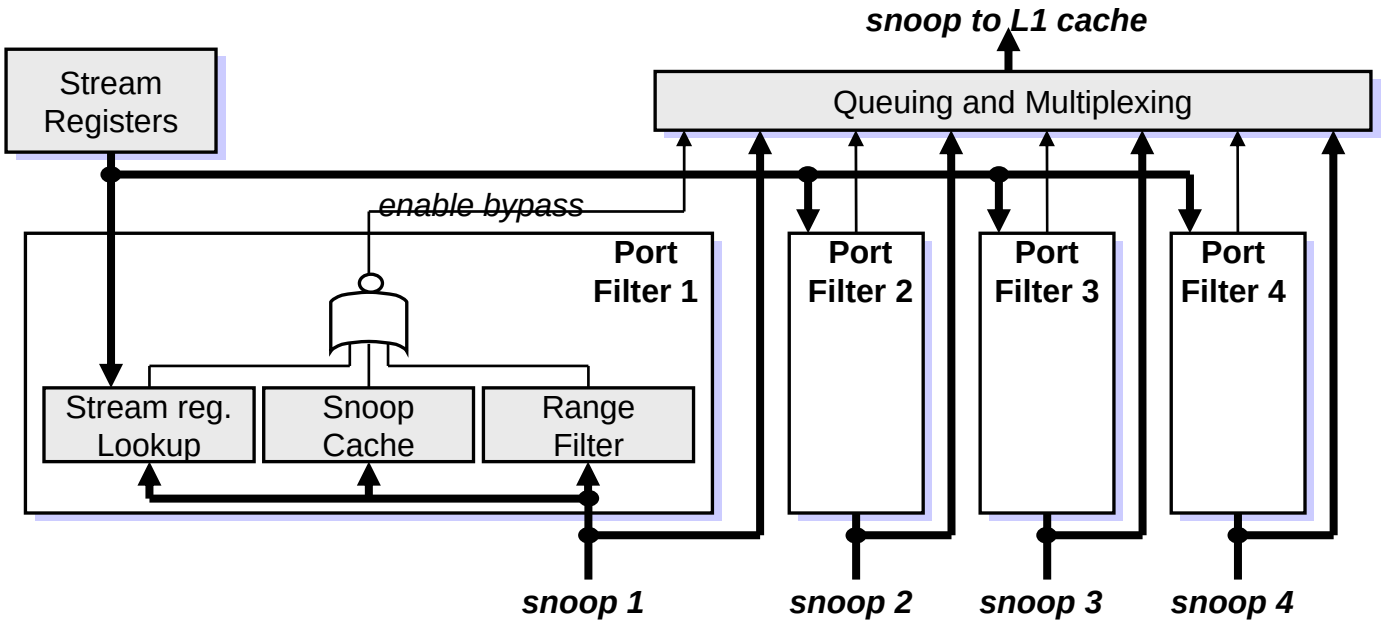
- Key to snoop filter effectiveness is high filtering rate
- Correctness requirement is to *never* miss a necessary snoop request
- Different applications and different data classes have distinct characteristics
  - Streams of contiguous data
  - Repeated accesses to the same data addresses
  - Some regions are known to not be shared
- Novel multi-component snoop filter
  - matches individual snoop filters to specific access patterns

Multi-component filtering: Matching filtering techniques to data access patterns

- Stream registers
  - Contiguous data areas
  - Adaptive to cache arbitrarily sized contiguous regions with a single register
  - Stream registers track strided and sequential streams
- Snoop caches
  - Cache of recently executed snoop requests
  - Multiple requests to same line do not have to cause multiple snoop lookups
  - Snoop caches track locality
- Range filter
  - Identify regions of known non-shared data
  - Configured by software

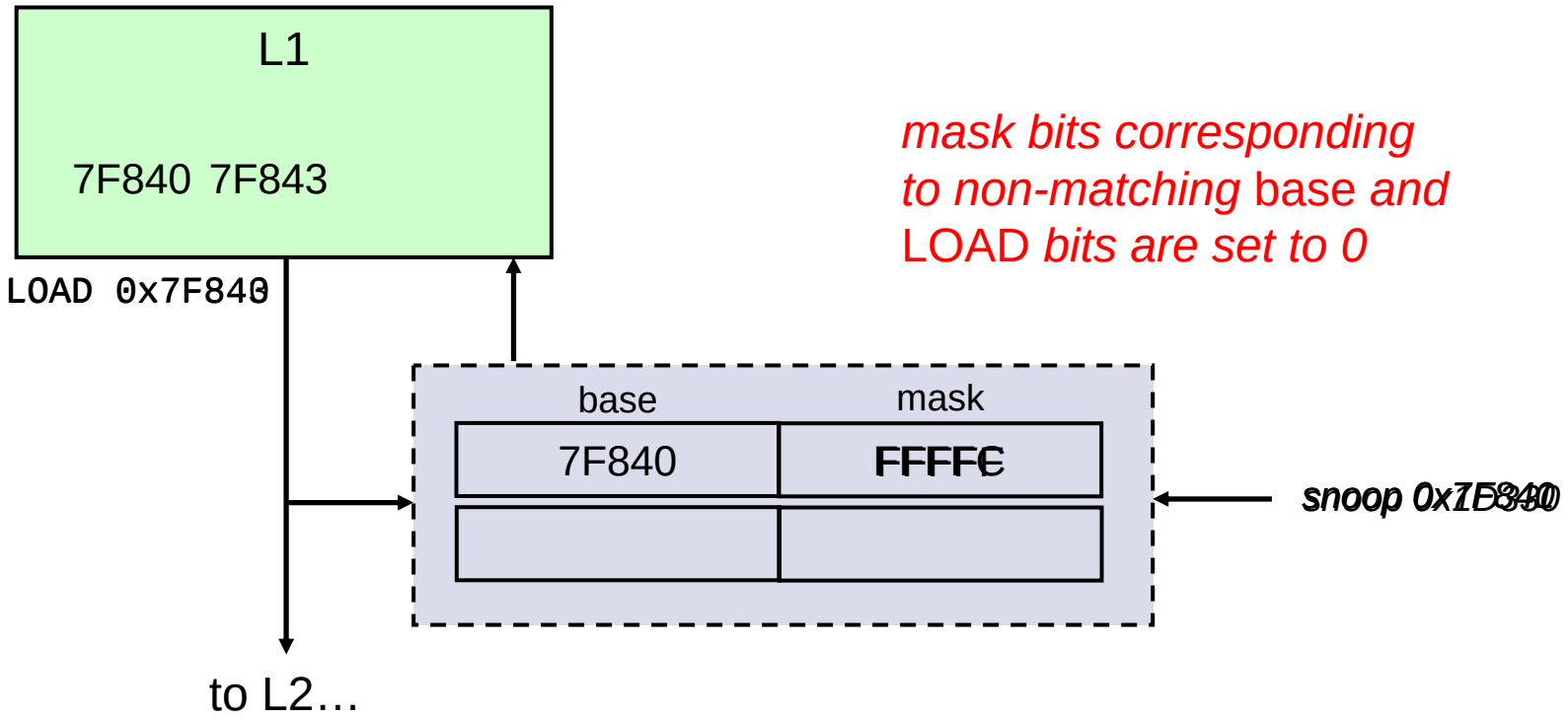
Multi-component port filters

- Scalability requires multiple, simultaneous lookups for each cache
  - Different sharing patterns
- Port filters capture sharing with a specific remote cache
  - Port filters can be replicated to service multiple incoming requests simultaneously ⇒ scalable
    - No communication necessary between filters



Stream registers

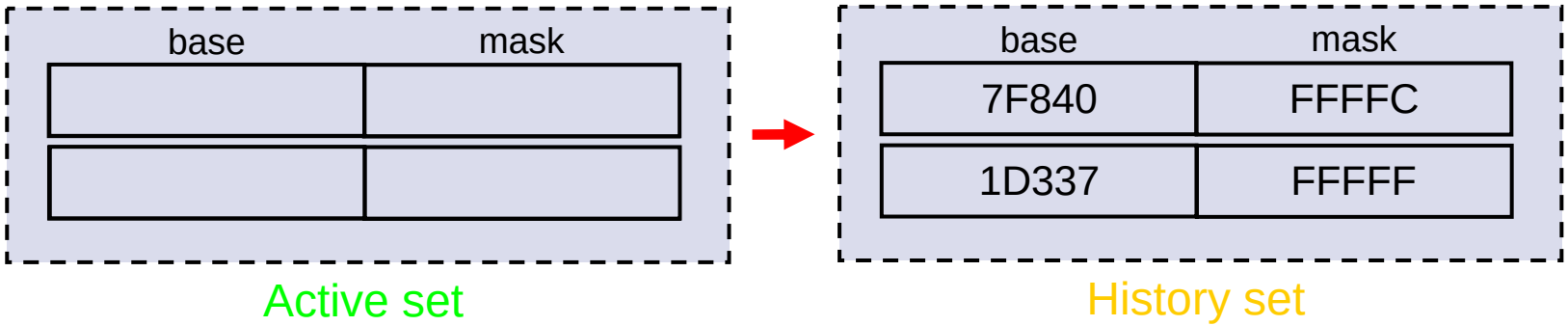
- Tracks a superset of data that are in the cache
  - Captures when a line *might be* in the cache
  - Stream register state is only modified by L1 activity
- Each stream register is intended to capture an address stream
  - Multiple registers for multiple streams



mask bits corresponding to non-matching base *and* LOAD *bits are set to 0*

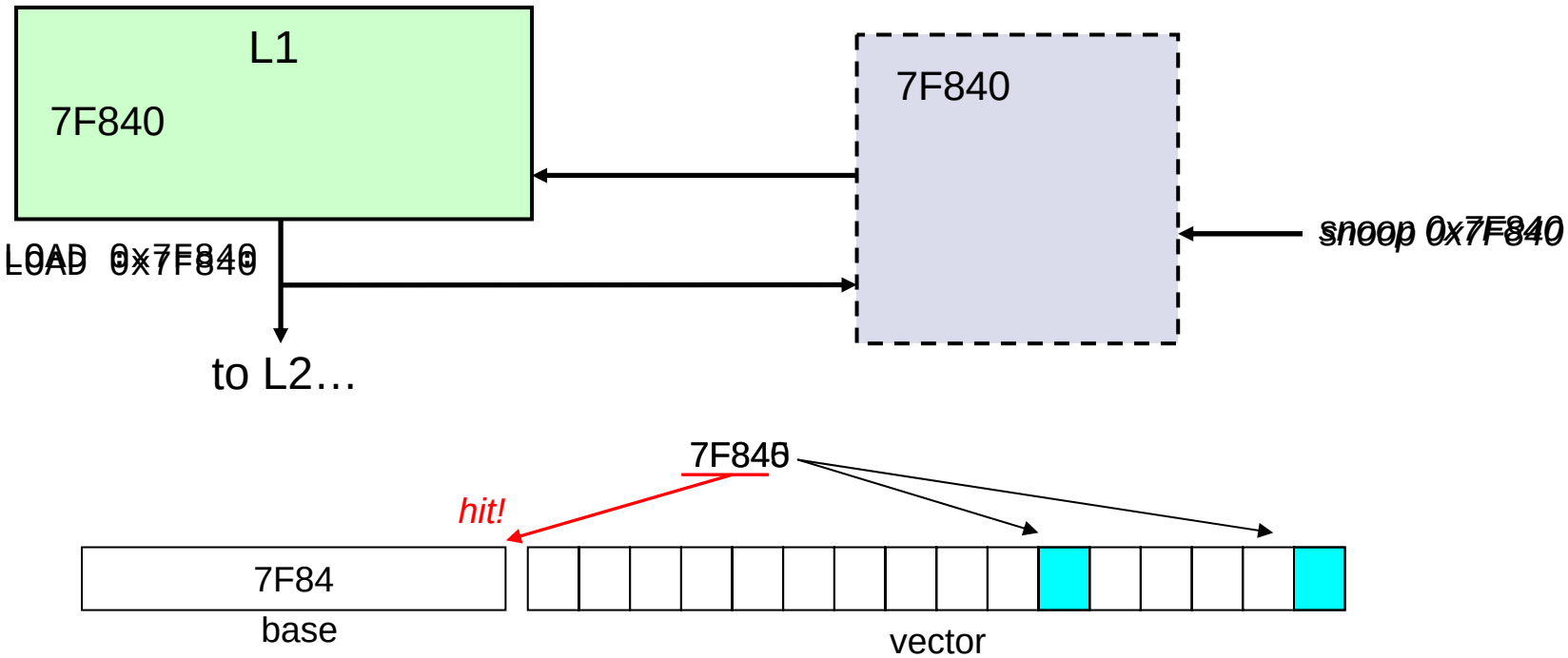
Stream registers: cache wrapping

- Stream registers become less discriminating as addresses are added
  - Addresses cannot be subtracted from stream registers
  - Solution: periodically reset the registers
    - Safe to do only when *all* of the addresses they are tracking are no longer in L1



Snoop cache

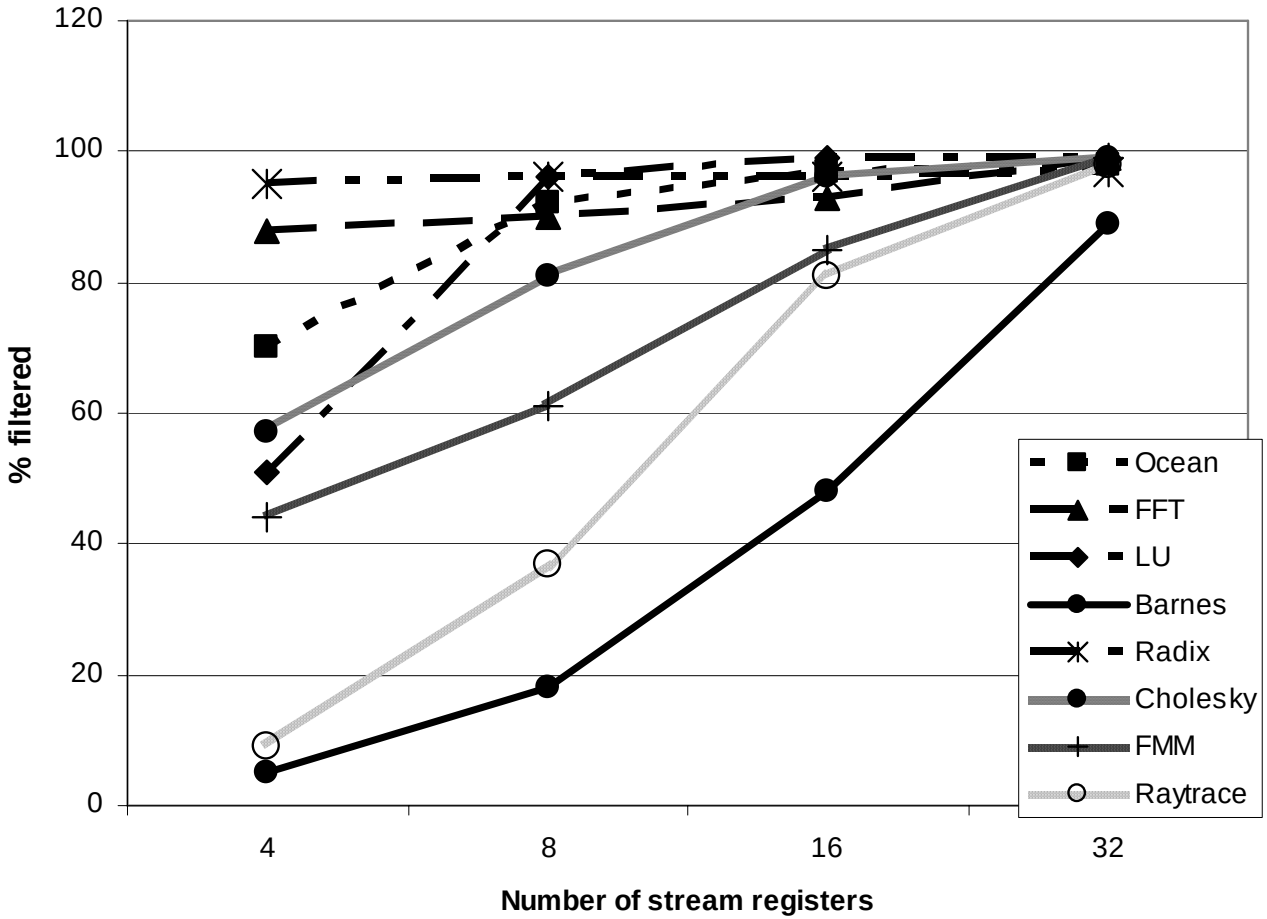
- If a cache line was invalidated, no need to invalidate it again
  - A hit in the snoop cache guarantees that the line is *not* in L1
  - Store L1 snoop addresses in the snoop cache, and evict LOAD addresses
  - Snoop cache state may be modified after each lookup



Evaluation methodology

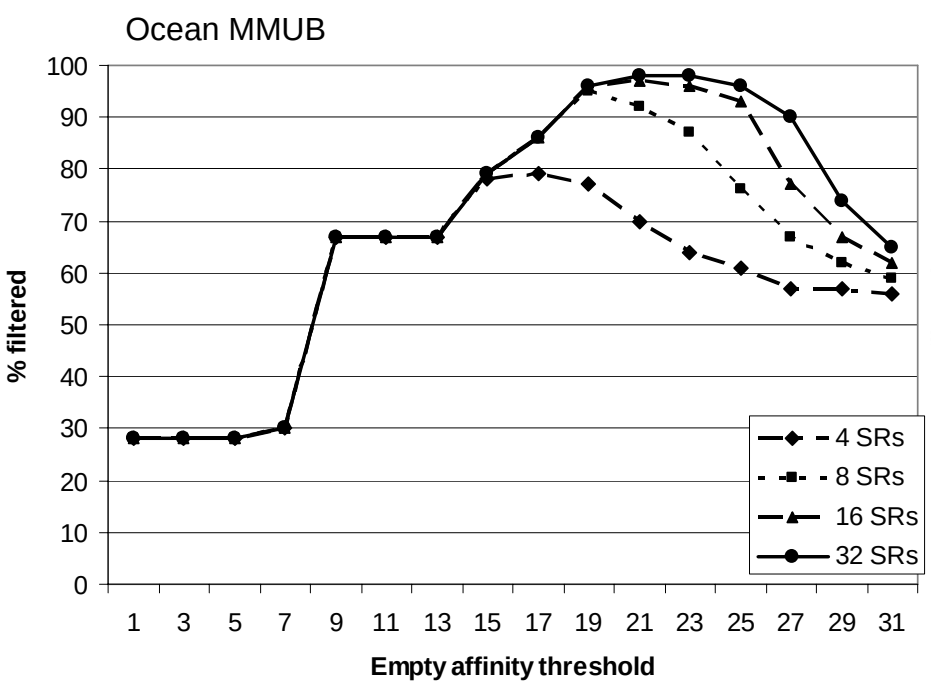
- Trace-based evaluation and hardware measurements
  - Trace-based simulation during design to study design-tradeoffs
  - Hardware measurements confirm accuracy of modeling and performance benefits
- Multiprocessor traces
  - Used Augmint to generate traces
  - SPLASH-2 benchmarks and other applications
- Custom simulator to study filter effectiveness
  - Modeled the PowerPC 450 L1 data cache
  - Memory accesses processed sequentially, with their effect applied immediately
- Evaluated various snoop filters
  - More details on sizing in: “Improving the Accuracy of Snoop Filtering Using Stream Registers”, MEDEA Workshop in conjunction with PACT 2007 Conference 2007

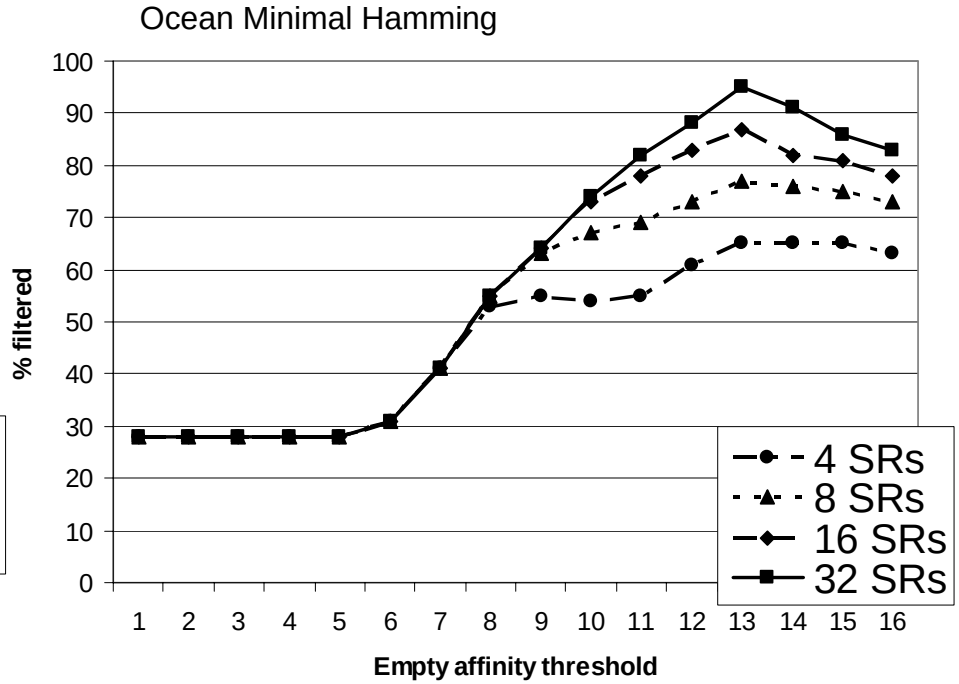
Varying the number of stream registers



Stream register update policy

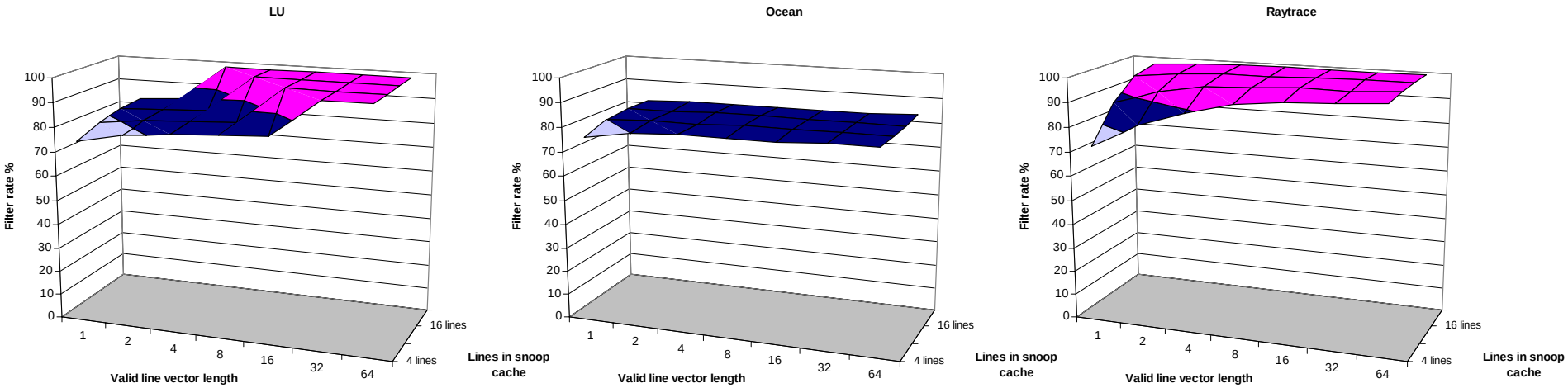
- Affinity: Matching stream registers to memory accesses
  - Minimal Hamming Distance – captures strided accesses thru memory
  - Most Matching Upper Bits (MMUB) – captures contiguous regions of accesses
- Selecting the “Empty Affinity” threshold at which to start a new stream
  - If too low, streams not well captured and too many streams are started.
  - If too high, all streams merged and filters force passing of wide range of accesses.





V. Salapura et al., Design and Implementation of the Blue Gene/P Snoop Filter
HPCA ’08

Snoop cache size

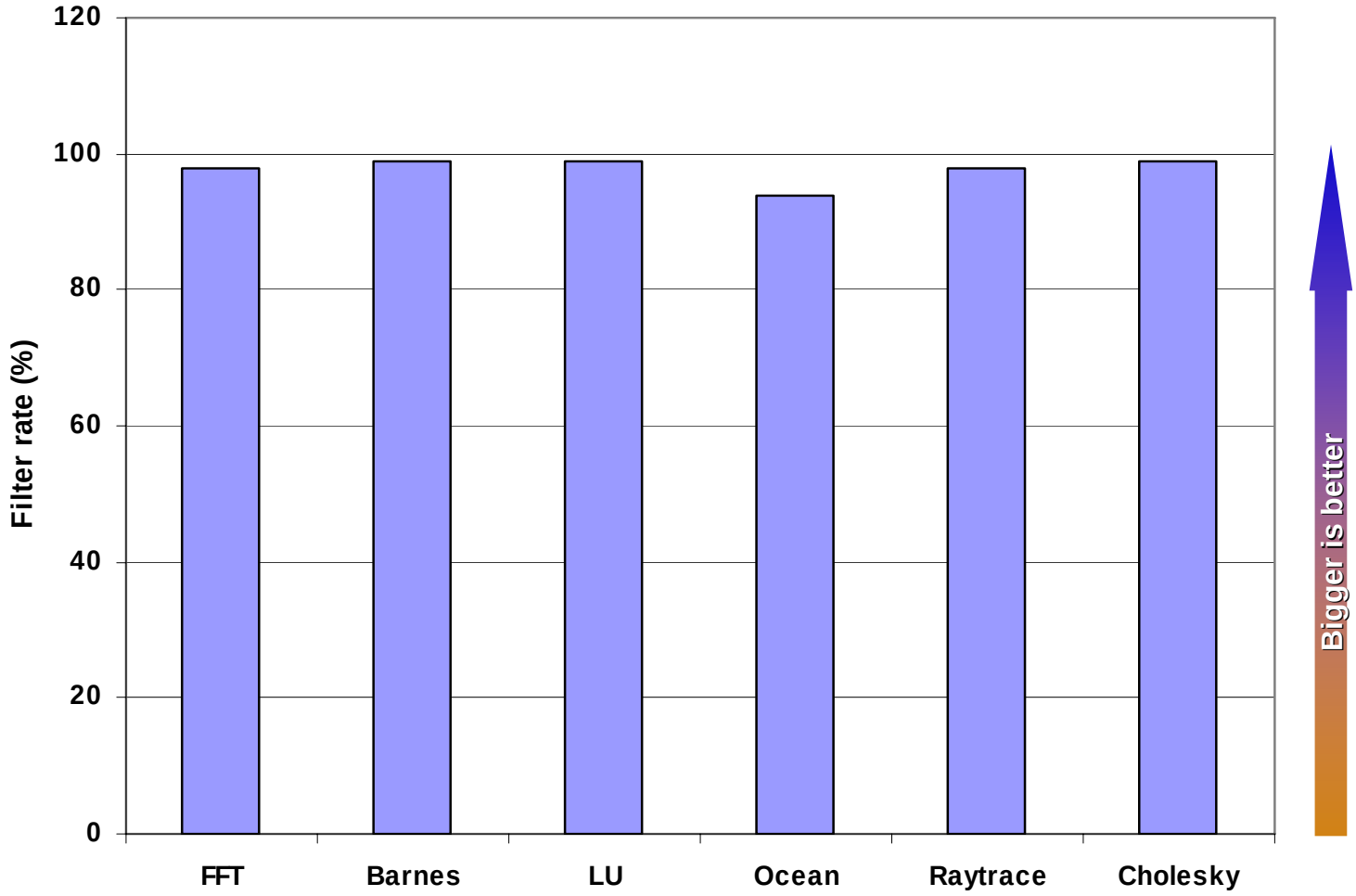


- Different filtering rate depending on the benchmark
- Larger caches advantageous for some applications, less important for others

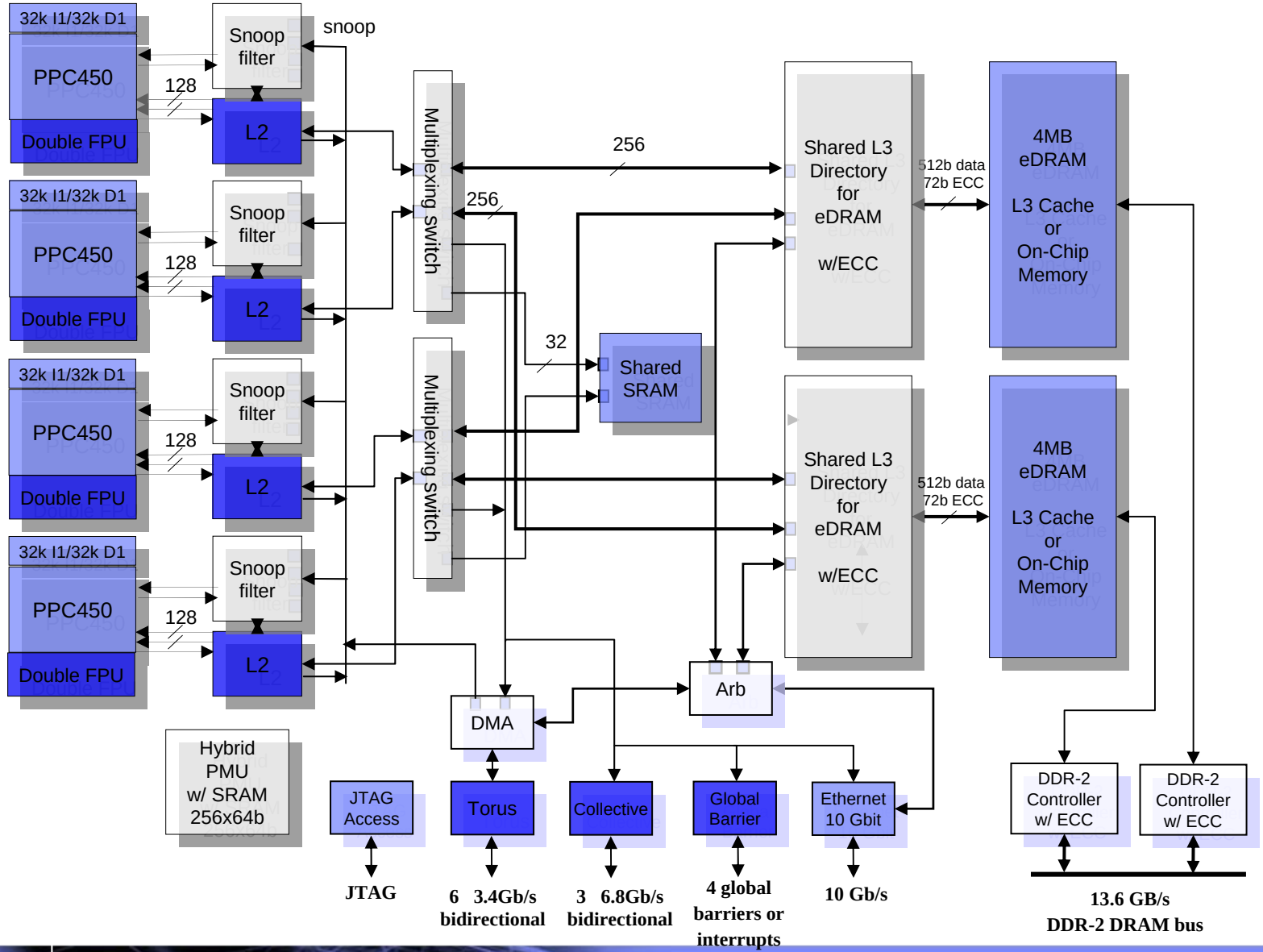
Combined snoop filter

Simulation results

snoop caches: 8 lines, 32 cache lines
8 stream registers

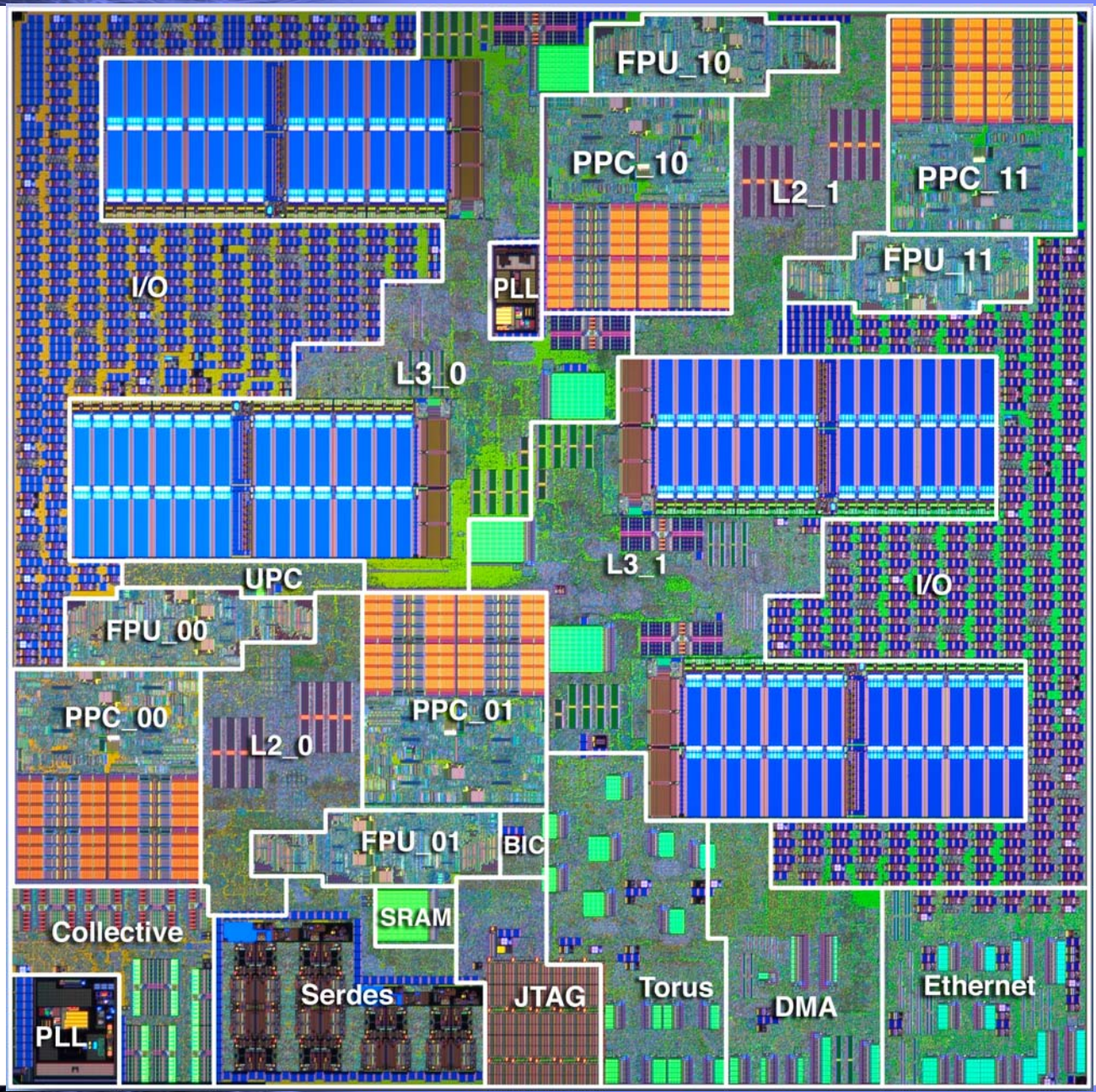


System-level view

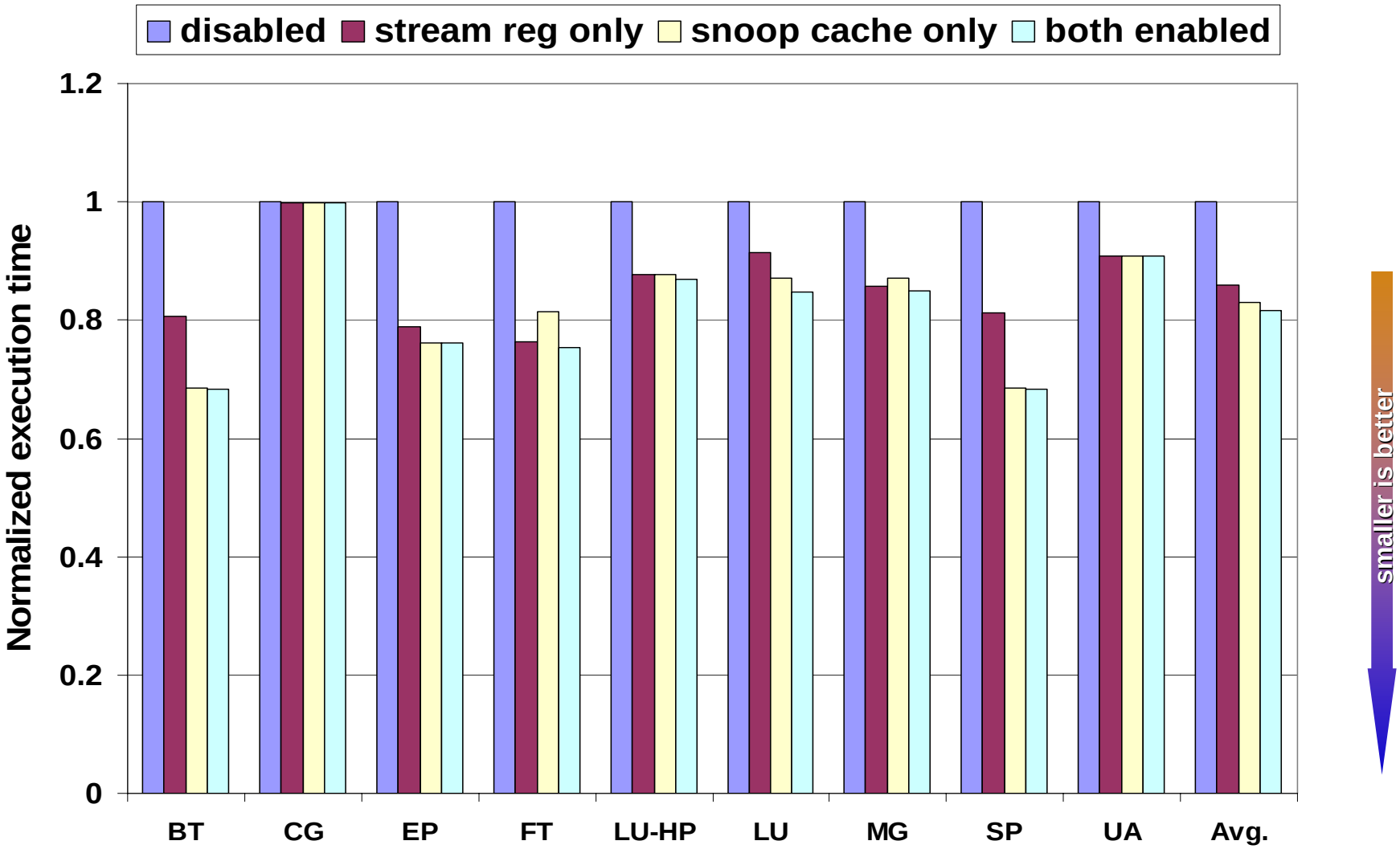


BlueGene/P ASIC

- IBM Cu-08 90nm CMOS ASIC process technology
- Die size 173 mm^2
- Clock frequency 850MHz
- Transistor count 208M
- Power dissipation 16W



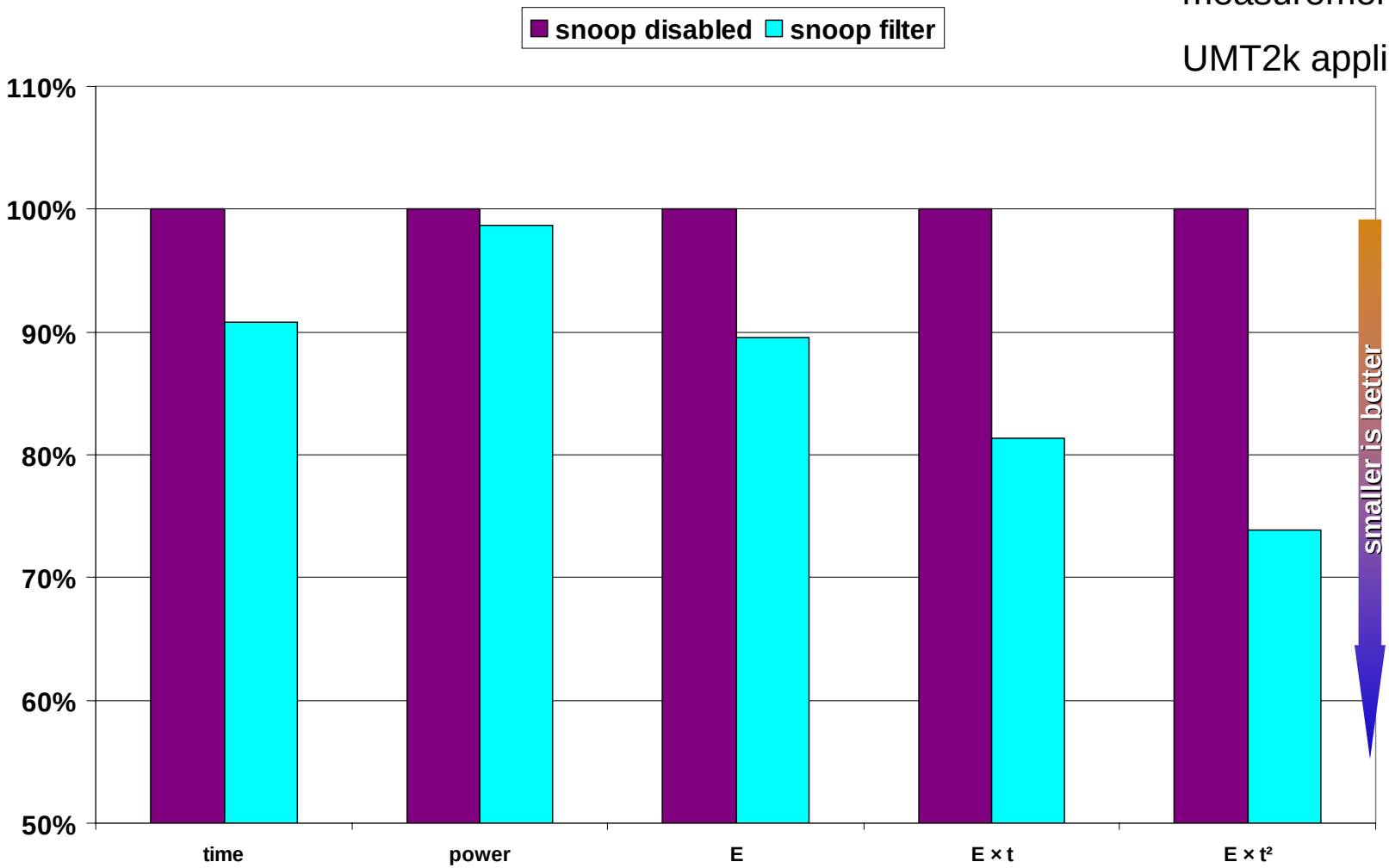
Hardware measurements: snoop filter efficiency



Snoop filtering improves power and performance

Actual hardware measurements

UMT2k application



Conclusion

- System level optimization is key to delivering on CMP promise
  - Coherence traffic problematic in terms of performance and power
- Multi-component filtering to capture data sharing and use patterns
  - Capture both temporal locality and streamed memory accesses
  - Capture region-based sharing
- Port filtering
  - Scalable, as port filters can be replicated to service multiple incoming requests simultaneously
    - No communication necessary between filters
- Stream registers
  - Adaptive filter to capture streams corresponding to contiguously accessed memory regions
- Snoop filtering highly effective solution for multicore systems
  - Filters up to 99% of all coherence requests
  - Efficient hardware implementation
  - Significant performance improvement
  - Power efficient

BlueGene on the Web

www.research.ibm.com/bluegene

The Blue Gene/P project has been supported and partially funded by Argonne National Laboratory and the Lawrence Livermore National Laboratory on behalf of the United States Department of Energy under Subcontract No. B554331.