Operating System Management of Shared Caches on Multicore Processors

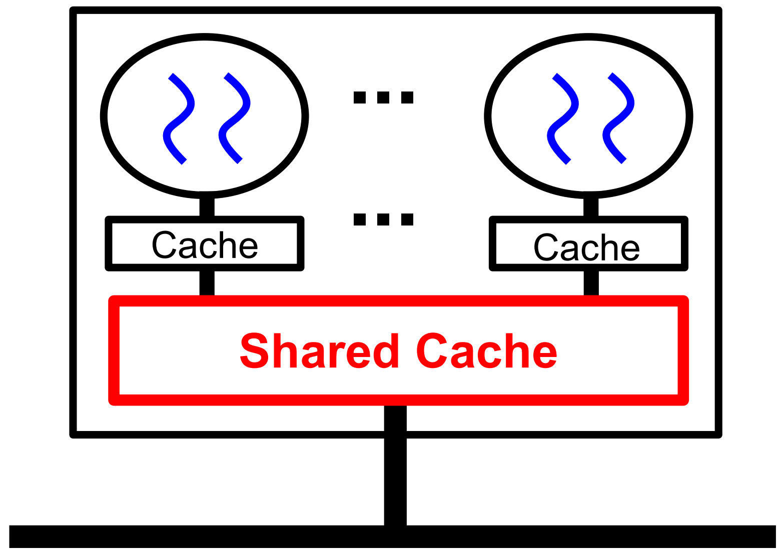
-- Ph.D. Thesis Presentation --

Apr. 20, 2010

David Tam

Supervisor: Michael Stumm

Multicores Today



Multicores are Ubiquitous

- Unexpected by most software developers
- Software support is lacking (e.g., OS)

General Role of OS

- Manage shared hardware resources

New Candidate

- **Shared cache**: performance critical
- Focus of thesis

Thesis

OS should manage on-chip shared caches of multicore processors

Demonstrate:

- Properly managing shared caches at OS level can increase performance

Management Principles

1. Promote sharing
   - For threads that share data
   - Maximize major advantage of shared caches
2. Provide isolation
   - For threads that do not share data
   - Minimize major disadvantage of shared caches

Supporting Role

- Provision the shared cache online

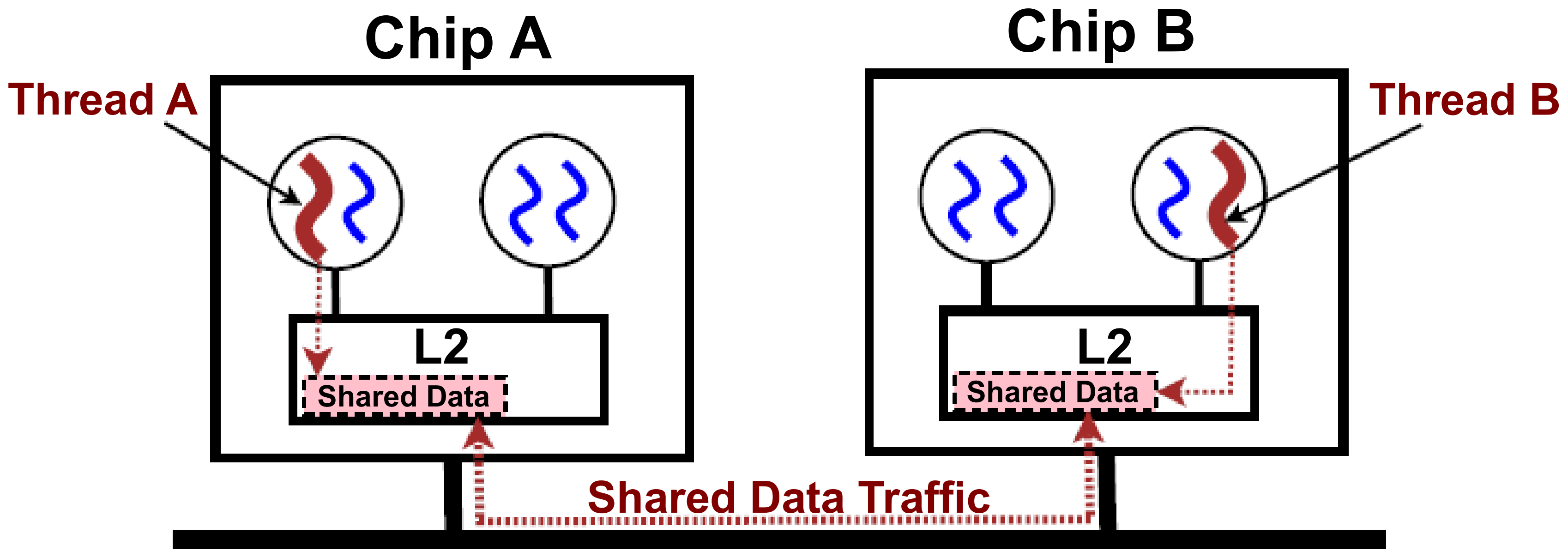
#1 - Promote Sharing

Problem: Cross-chip accesses are slow

Solution: Exploit major **advantage** of shared caches:
Fast access to shared data

OS Actions: Identify & localize data sharing

View: Match software sharing to hardware sharing



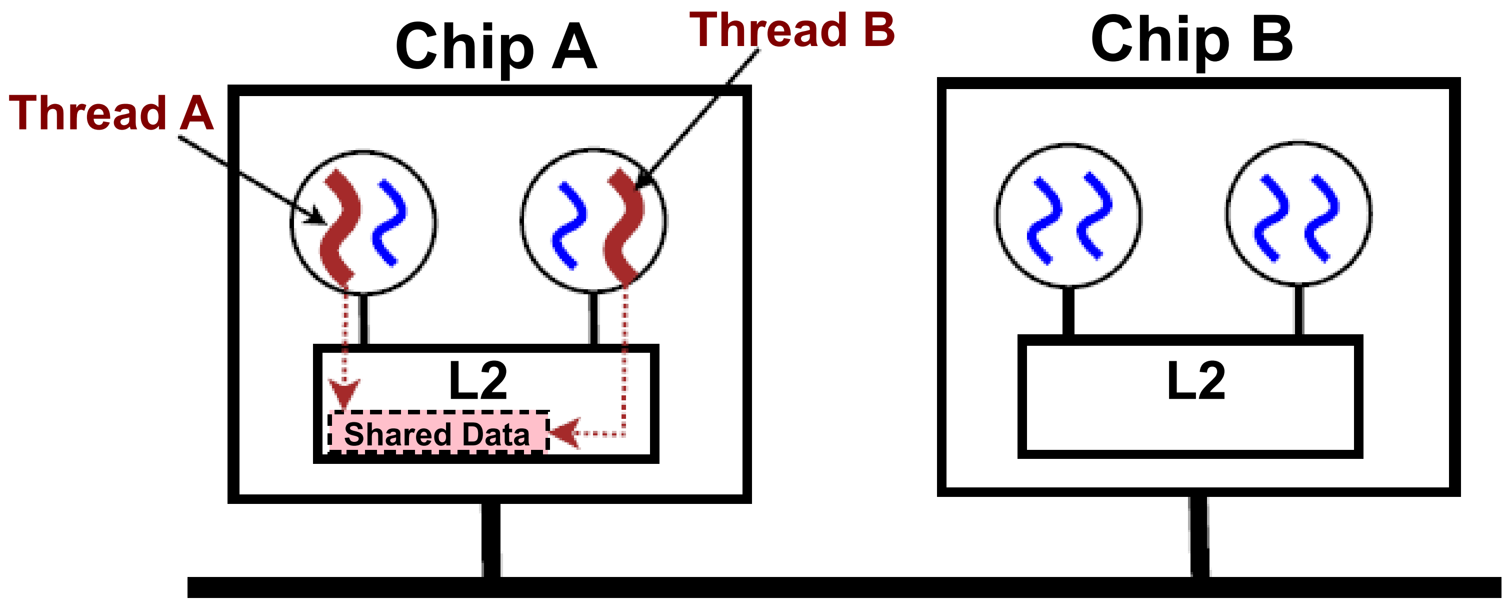
#1 - Promote Sharing

Problem: Cross-chip accesses are slow

Solution: Exploit major **advantage** of shared caches:
Fast access to shared data

OS Actions: Identify & localize data sharing

View: Match software sharing to hardware sharing



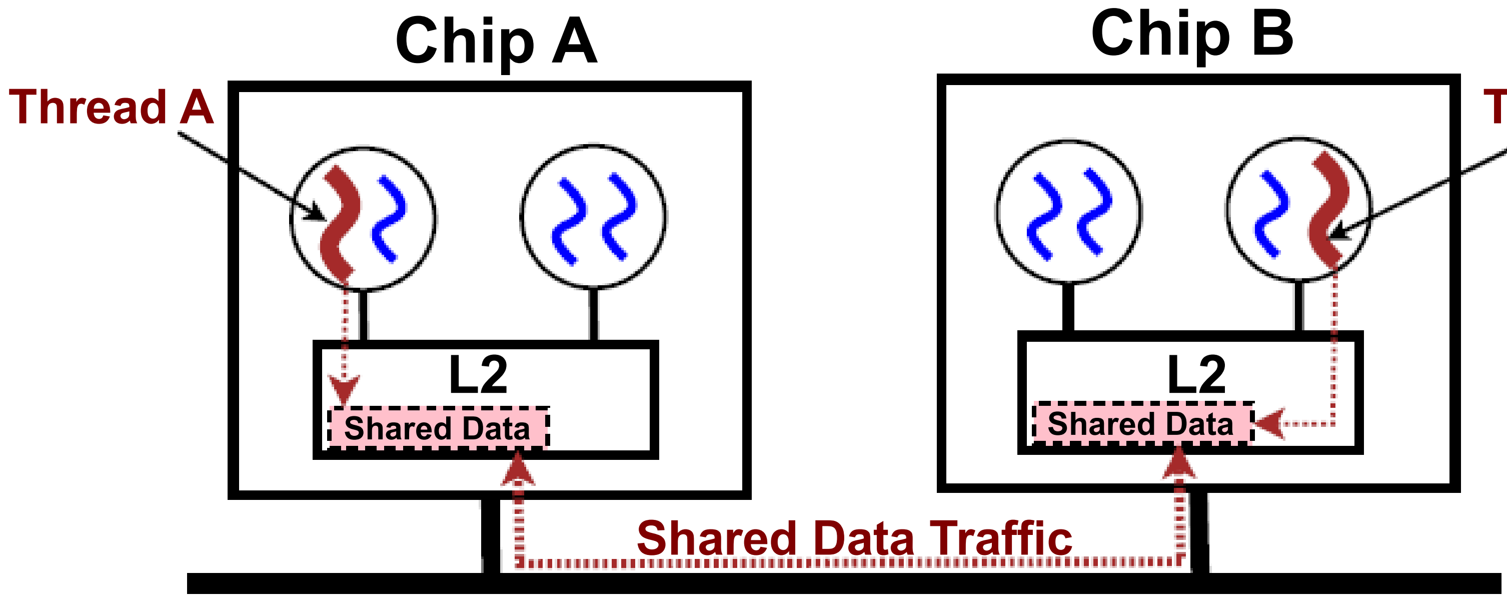
Thread Clustering [EuroSys'07]

Identify Data Sharing

- Detect sharing online with hardware performance counters
  - Monitor remote cache accesses (data addresses)
  - Track on a per-thread basis
  - Data addresses are memory regions shared with other threads

Localize Data Sharing

- Identify clusters of threads that access same memory regions
- Migrate threads of a cluster onto same chip



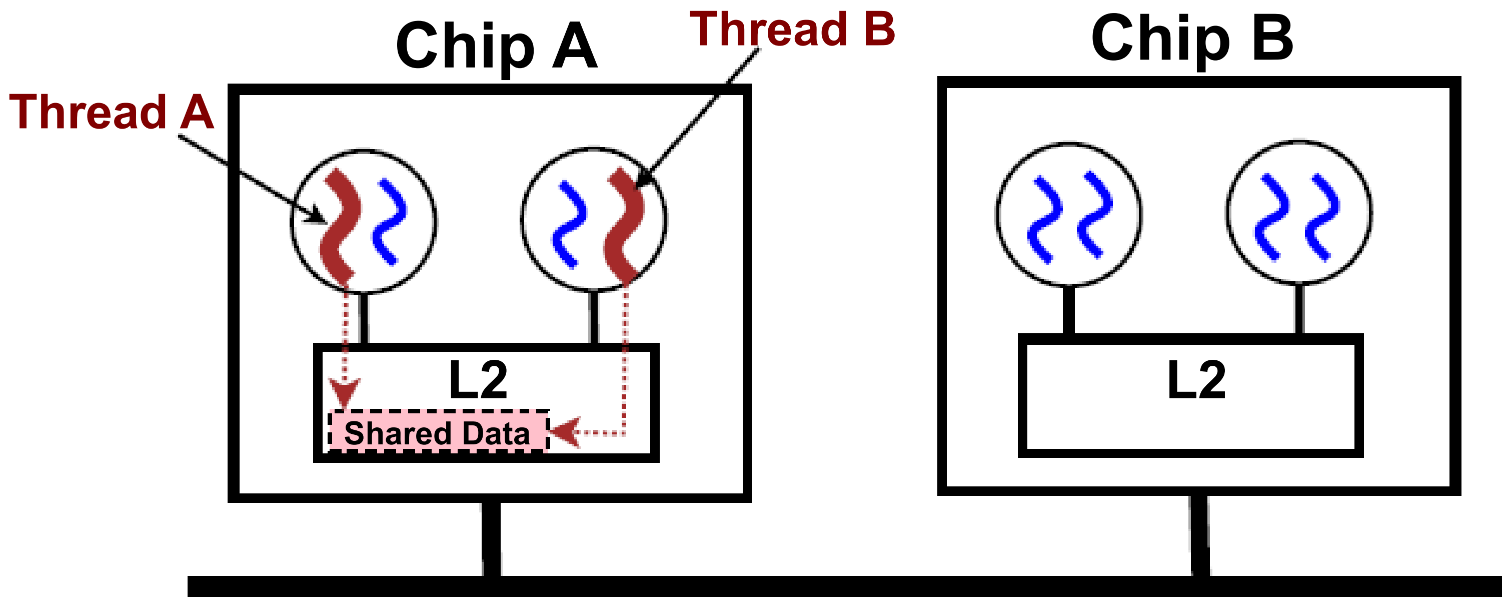
Thread Clustering [EuroSys'07]

Identify Data Sharing

- Detect sharing online with hardware performance counters
  - Monitor remote cache accesses (data addresses)
  - Track on a per-thread basis
  - Data addresses are memory regions shared with other threads

Localize Data Sharing

- Identify clusters of threads that access same memory regions
- Migrate threads of a cluster onto same chip

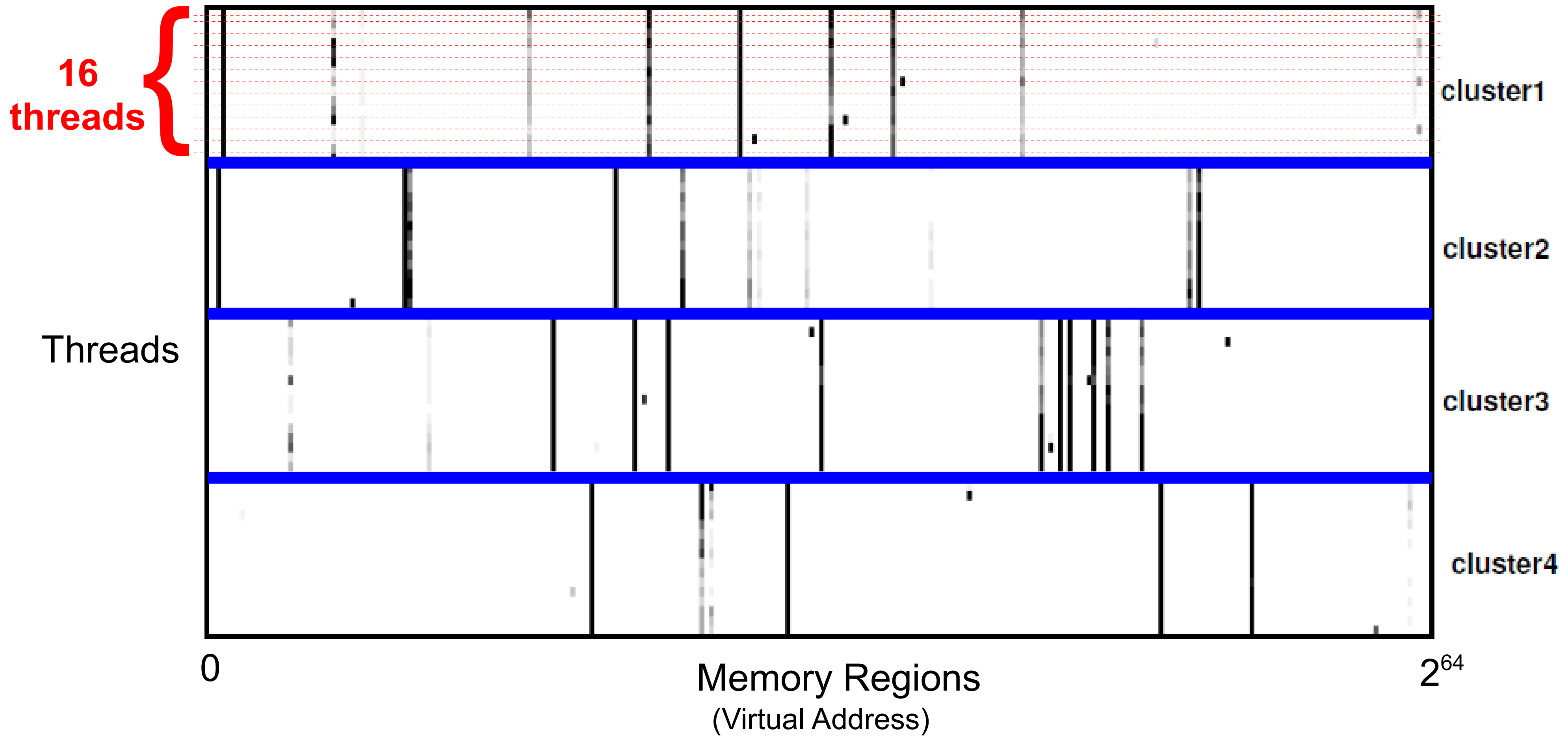


Visualization of Clusters

- SPECjbb 2000
  - 4 warehouses, 16 threads per warehouse
- Threads have been sorted by cluster for visualization

Sharing Intensity

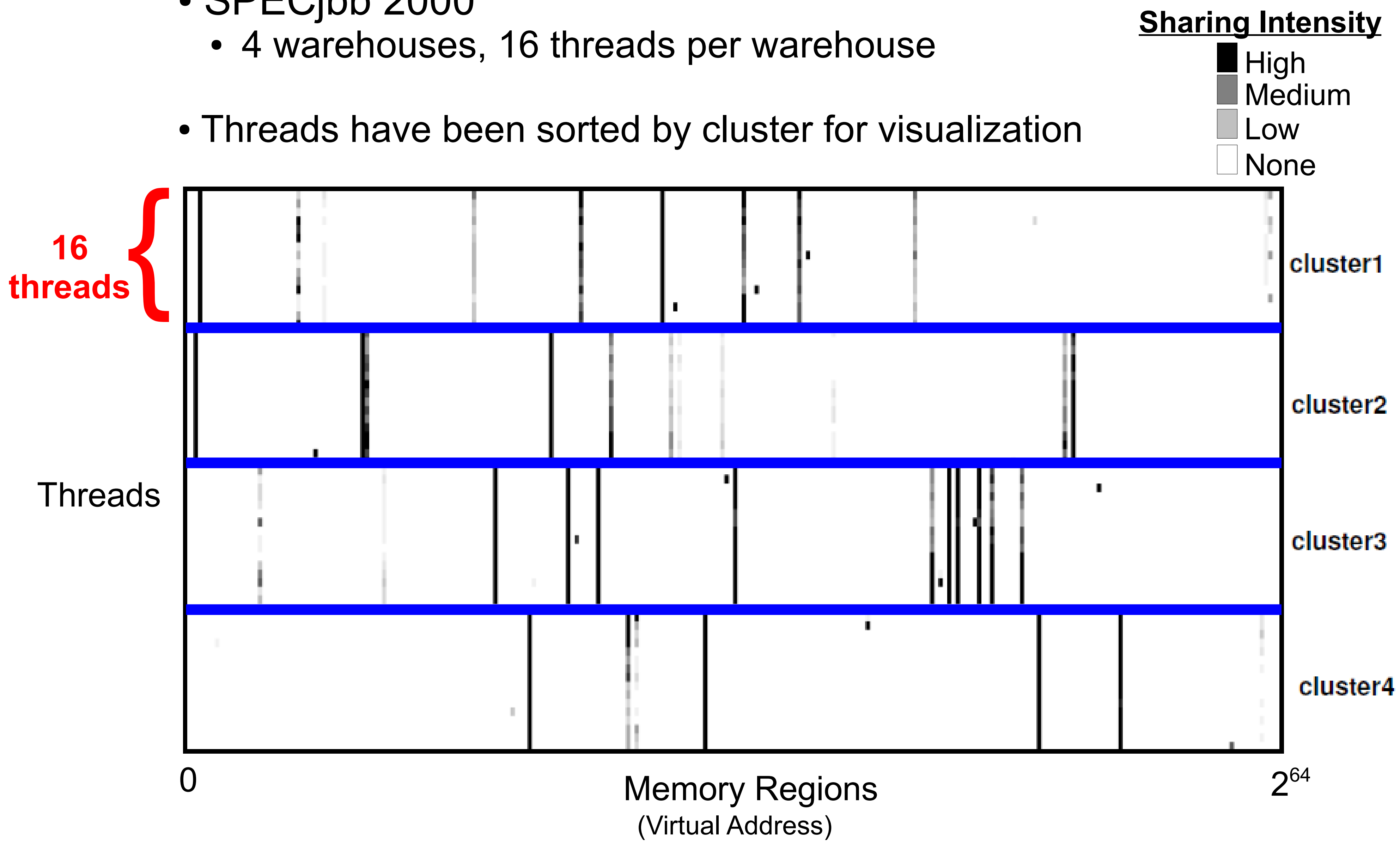
■ High
▨ Medium
□ Low
□ None



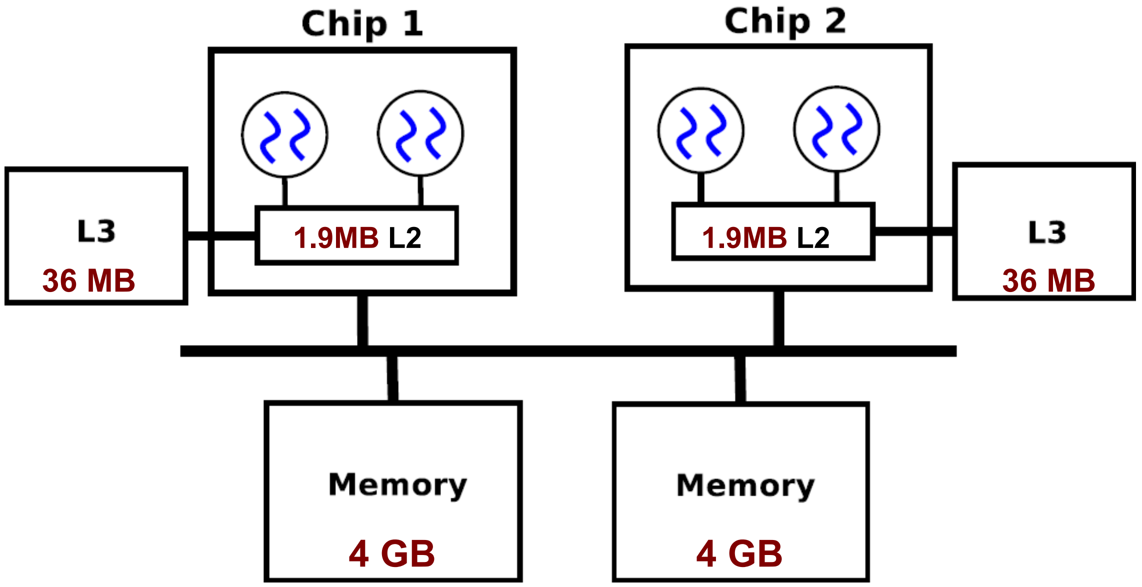
Visualization of Clusters

- SPECjbb 2000
  - 4 warehouses, 16 threads per warehouse

- Threads have been sorted by cluster for visualization



Performance Results



- Multithreaded commercial workloads
  - RUBiS, VolanoMark, SPECjbb2k
- *8-way* IBM POWER5 Linux system
  - 22%, 32%, 70% reduction in stalls caused by cross-chip accesses
  - 7%, 5%, 6% performance improvement
- *32-way* IBM POWER5+ Linux system
  - 14% SPECjbb2k potential improvement

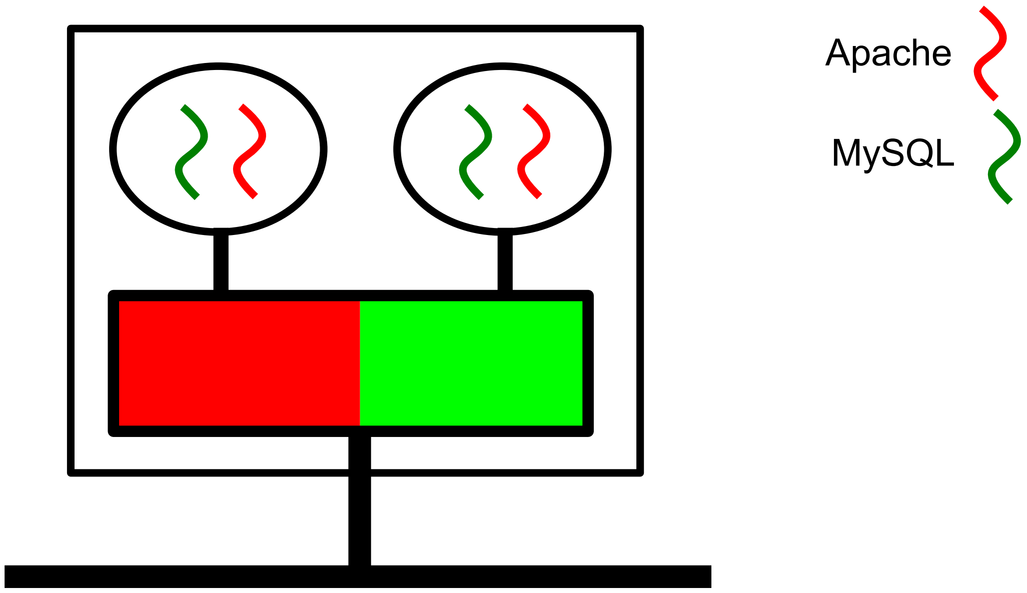
#2 – Provide Isolation

Problem: Major **disadvantage** of shared caches
Cache space interference

Solution: Provide cache space isolation between applications

OS Actions: Enforce isolation during physical page allocation

View: Partition into smaller private caches



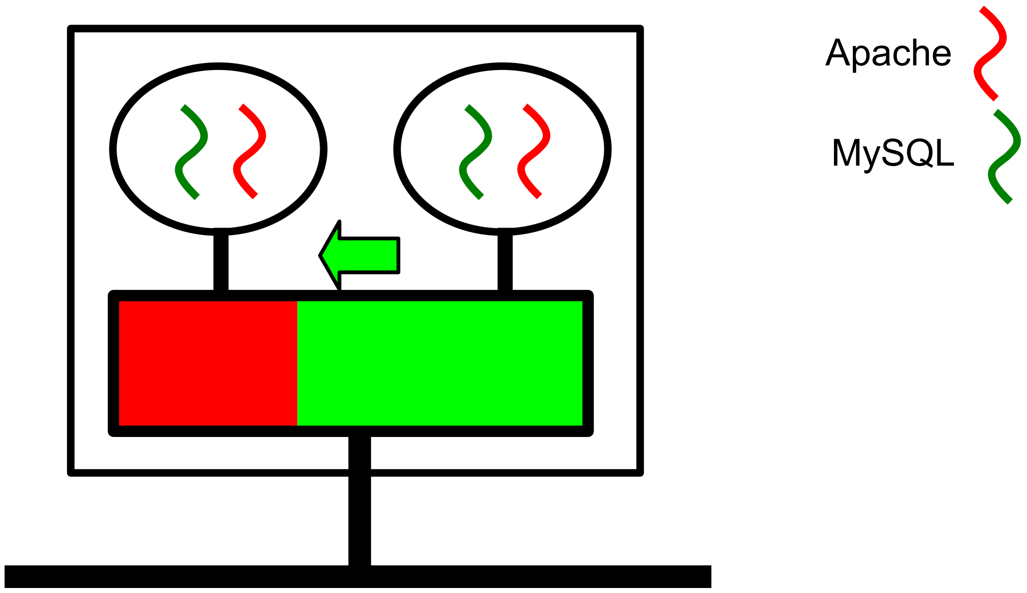
#2 – Provide Isolation

Problem: Major **disadvantage** of shared caches
Cache space interference

Solution: Provide cache space isolation between applications

OS Actions: Enforce isolation during physical page allocation

View: Partition into smaller private caches



#2 – Provide Isolation

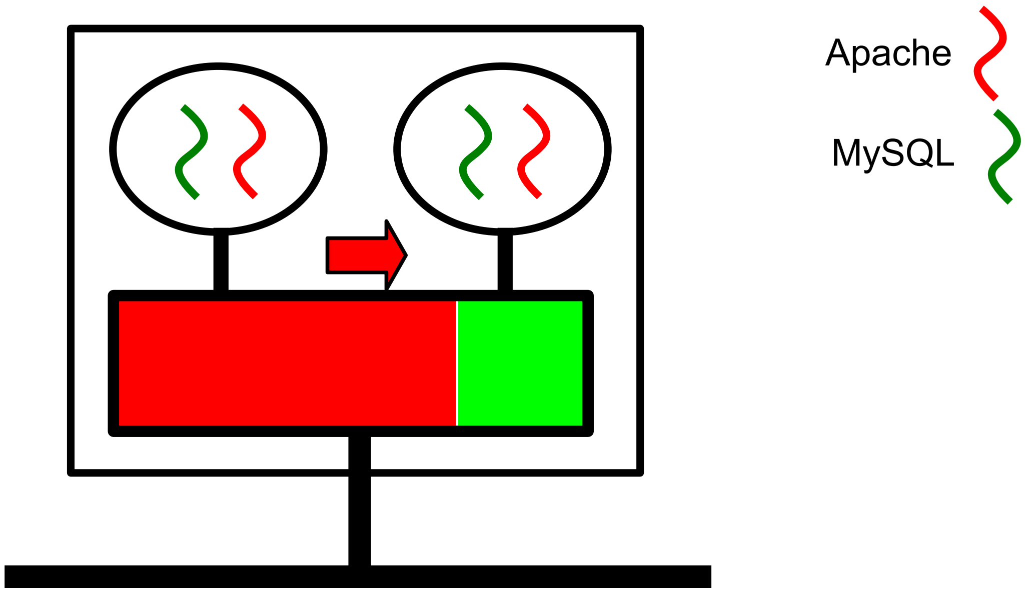
Problem: Major **disadvantage** of shared caches

Cache space interference

Solution: Provide cache space isolation between applications

OS Actions: Enforce isolation during physical page allocation

View: Partition into smaller private caches



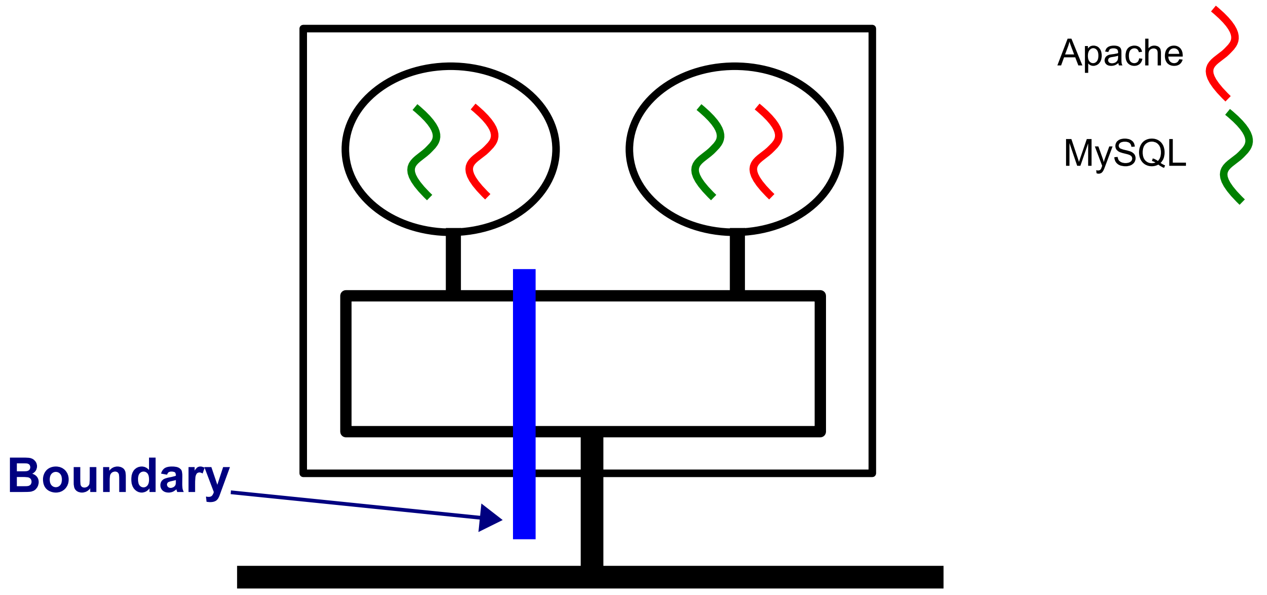
#2 – Provide Isolation

Problem: Major **disadvantage** of shared caches
Cache space interference

Solution: Provide cache space isolation between applications

OS Actions: Enforce isolation during physical page allocation

View: Partition into smaller private caches



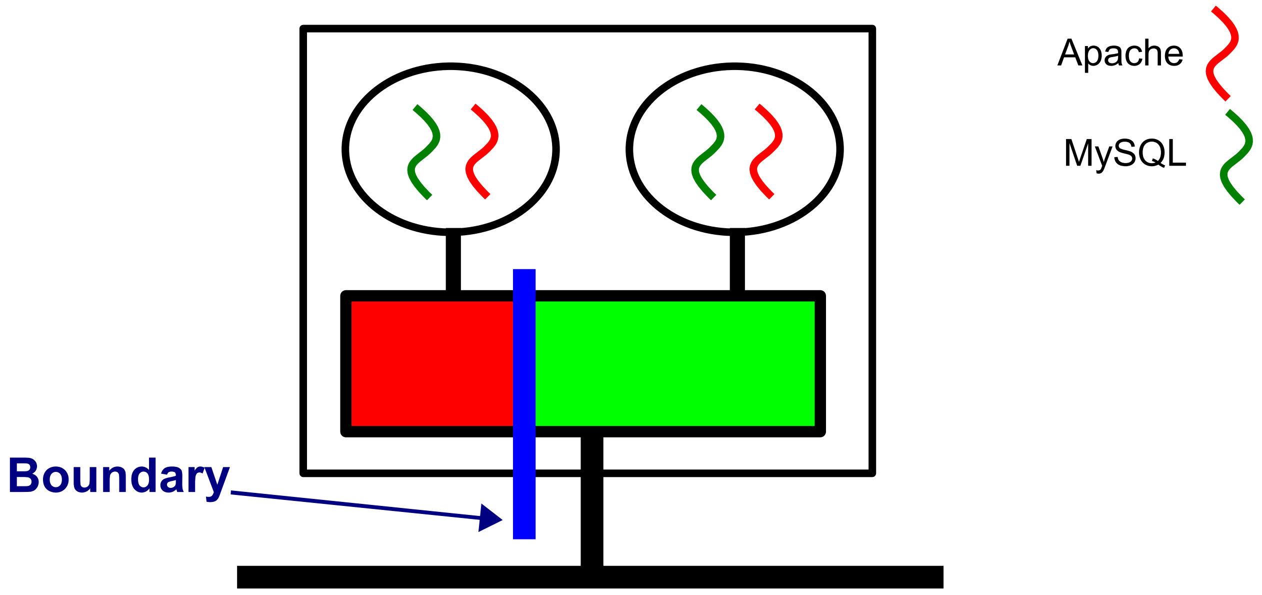
#2 – Provide Isolation

Problem: Major **disadvantage** of shared caches
Cache space interference

Solution: Provide cache space isolation between applications

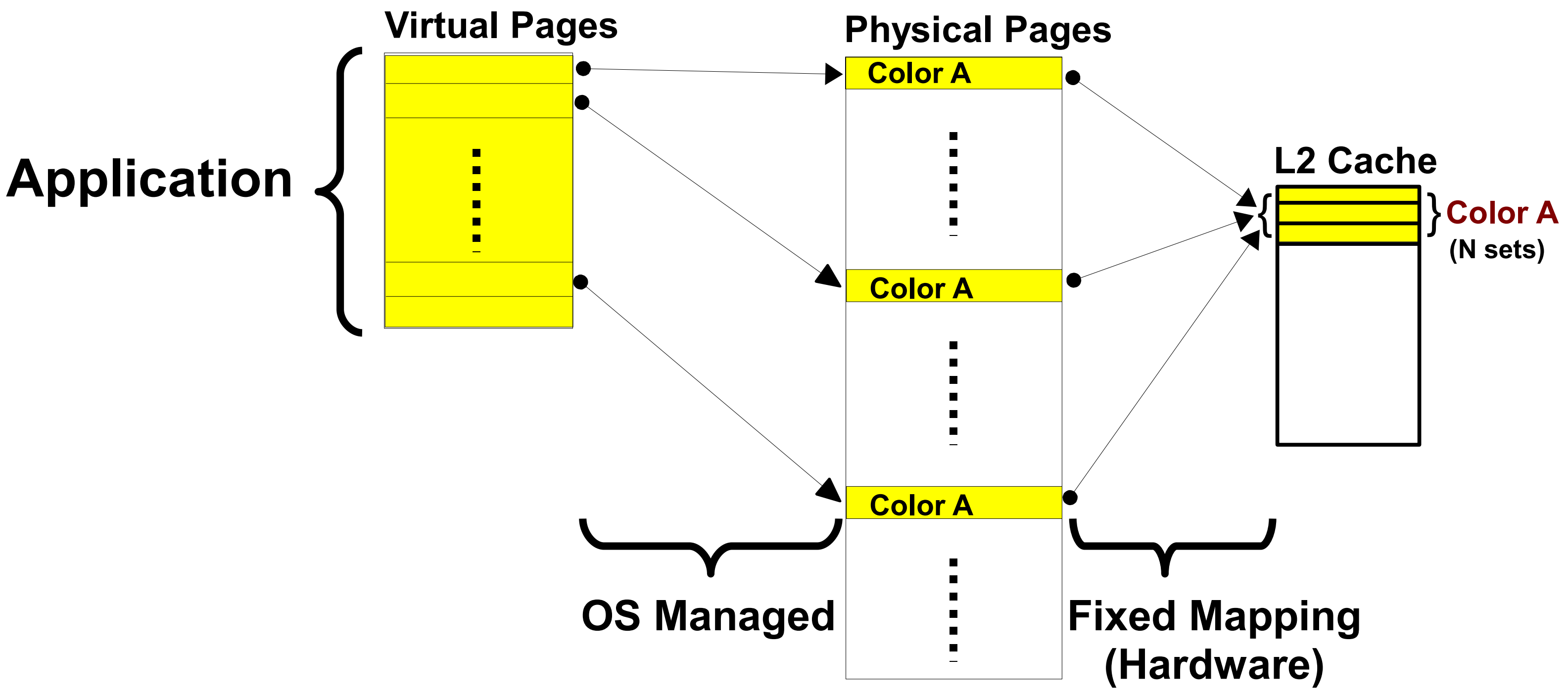
OS Actions: Enforce isolation during physical page allocation

View: Partition into smaller private caches



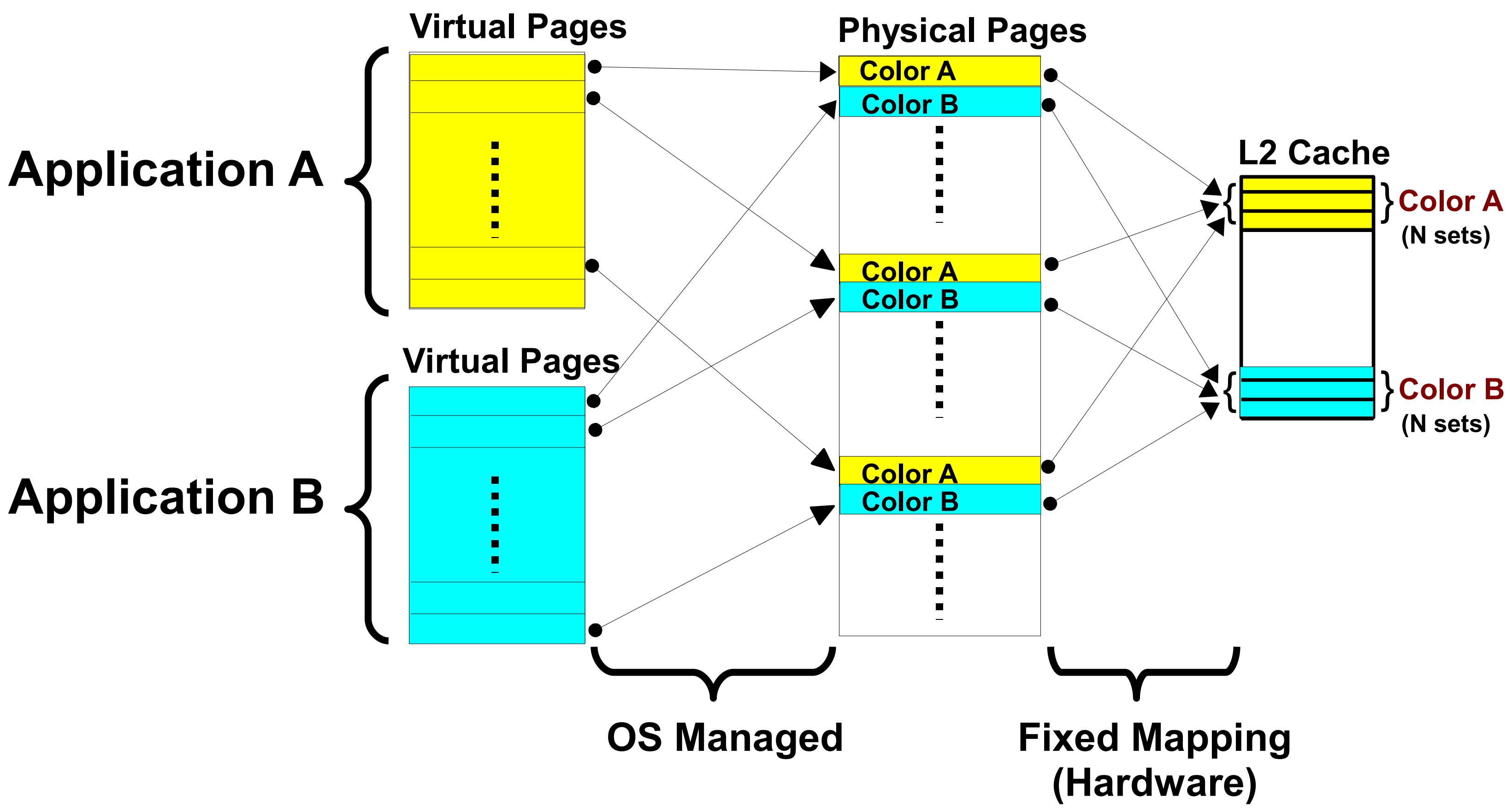
Cache Partitioning [WIOSCA'07]

- Apply page-coloring technique
- Guide physical page allocation to control cache line usage
- Works on existing processors

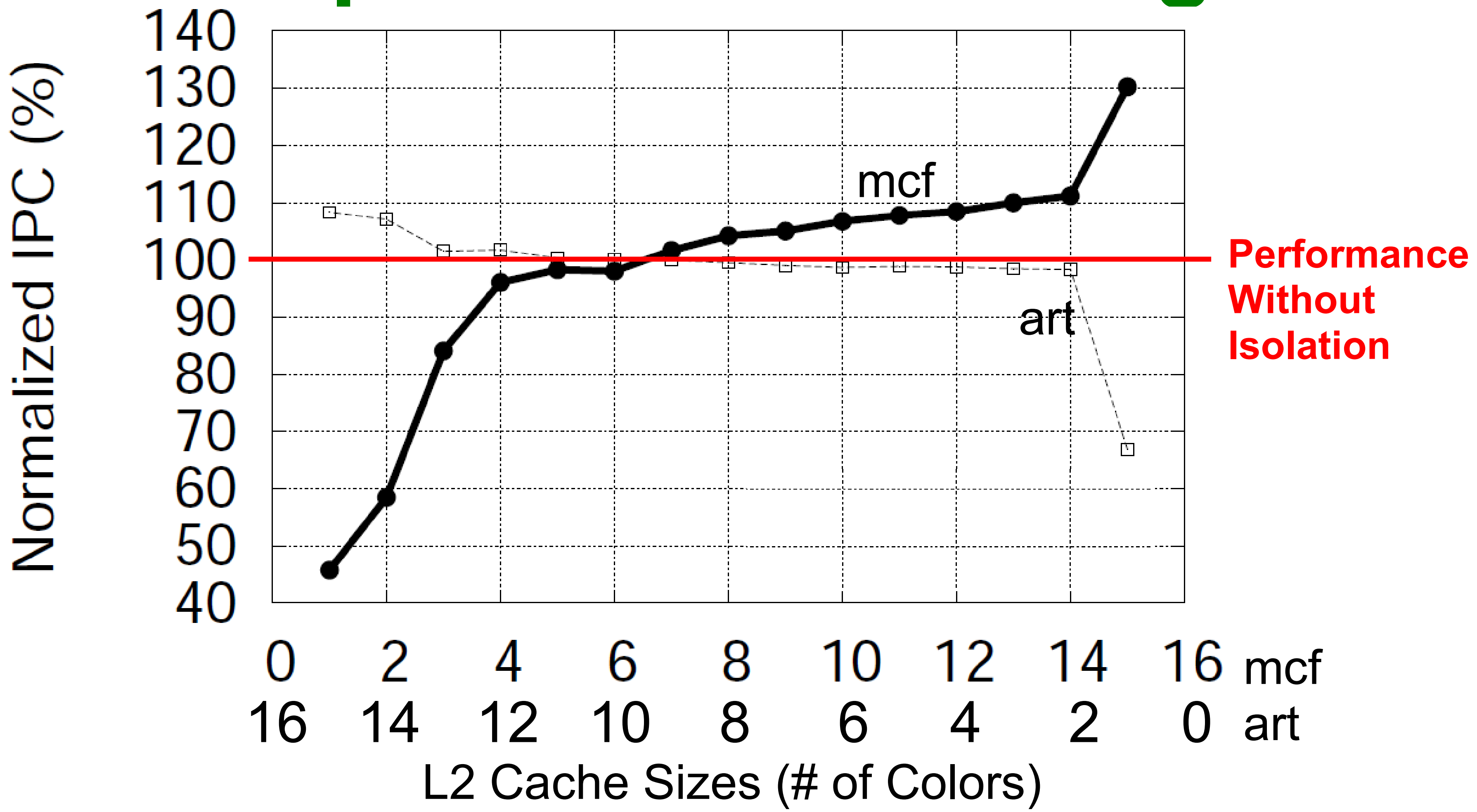


Cache Partitioning [WIOSCA'07]

- Apply page-coloring technique
- Guide physical page allocation to control cache line usage
- Works on existing processors



Impact of Partitioning



Performance of Other Combos

- 10 pairs of applications: SPECcpu2k, SPECjbb2k
  - 4% to 17% improvement (36MB L3 cache)
  - 28%, 50% improvement (no L3 cache)

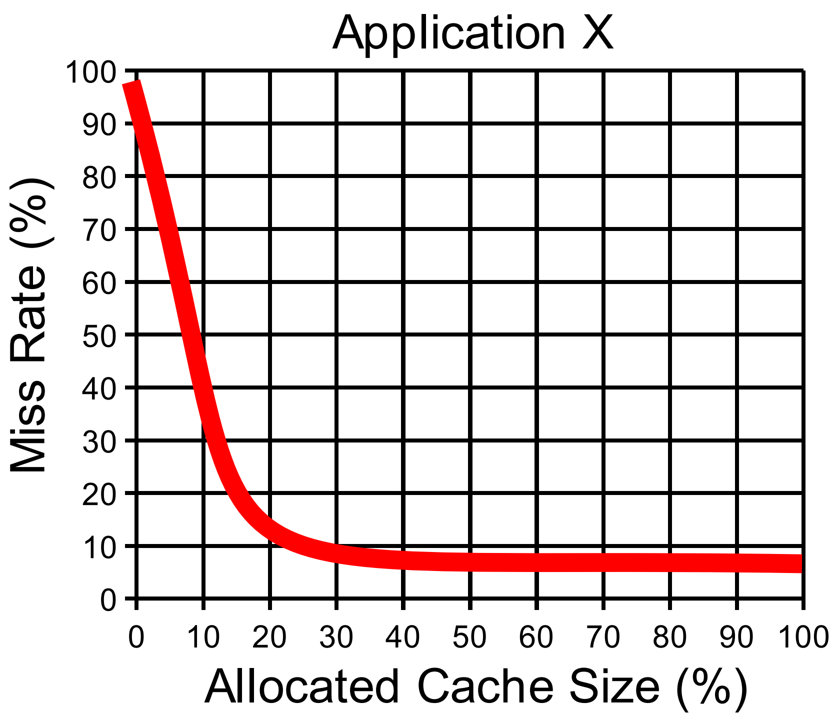
Provisioning the Cache

Problem: How to determine cache partition size

Solution: Use L2 cache miss rate curve (MRC) of application

Criteria: Obtain MRC rapidly, accurately, online, with low overhead, on existing hardware

OS Actions: Monitor L2 cache accesses using hardware performance counters



RapidMRC [ASPLOS'09]

Design

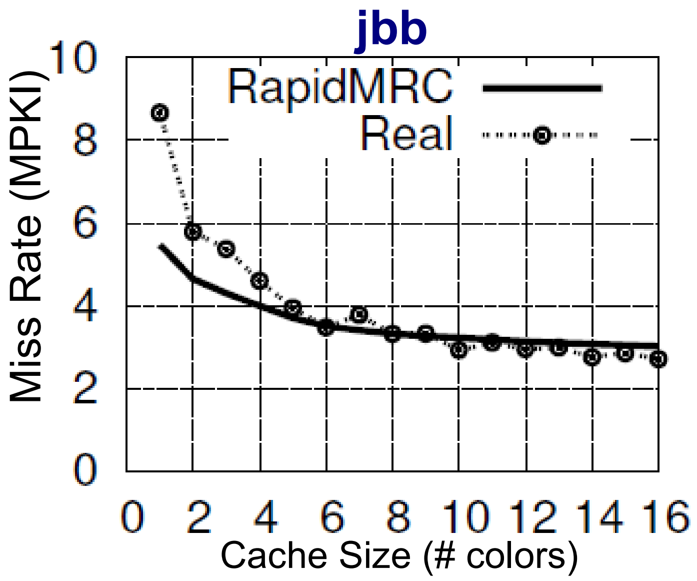
- Upon every L2 access:
  - Update *sampling* register with data address
  - Trigger interrupt to copy register to trace log in main memory
- Feed trace log into Mattson's stack algorithm [1970] to obtain L2 MRC

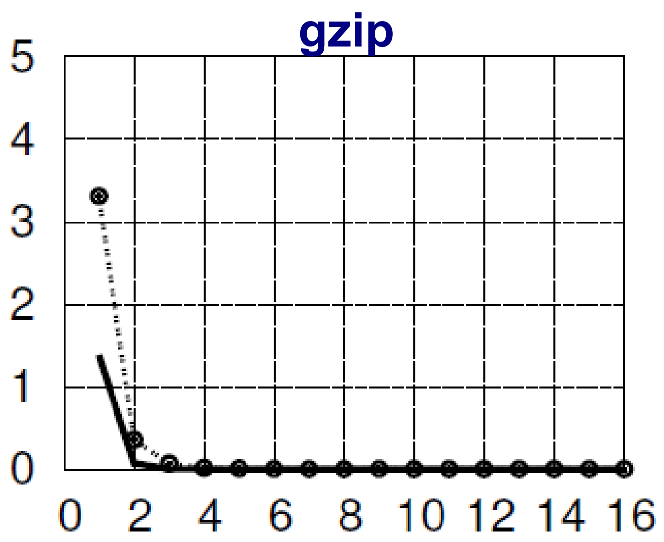
Results

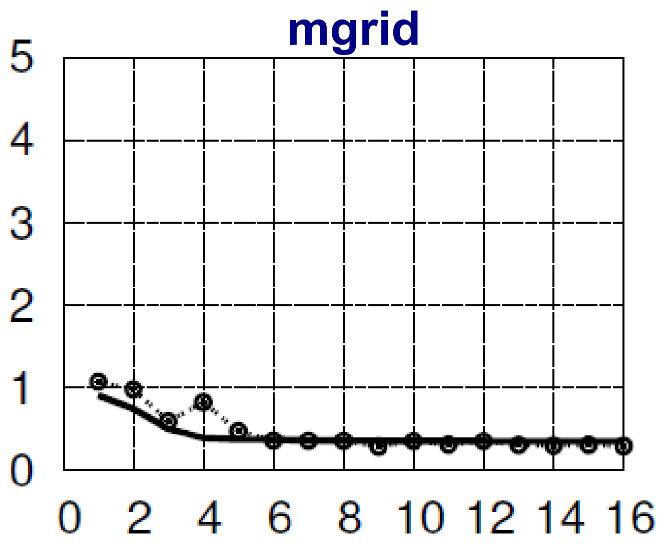
- **Workloads**
  - 30 apps from SPECcpu2k, SPECcpu2k6, SPECjbb2k
- **Latency**
  - 227 ms to generate online L2 MRC
- **Accuracy**
  - Good, e.g. up to 27% performance improvement when applied to cache partitioning

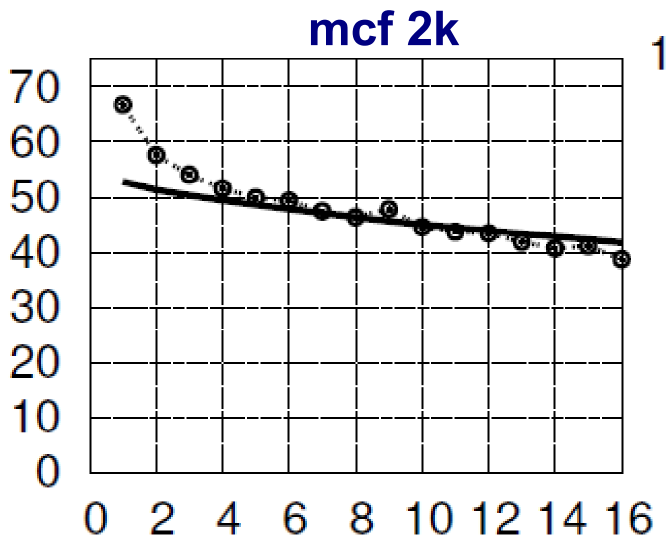
Accuracy of RapidMRC

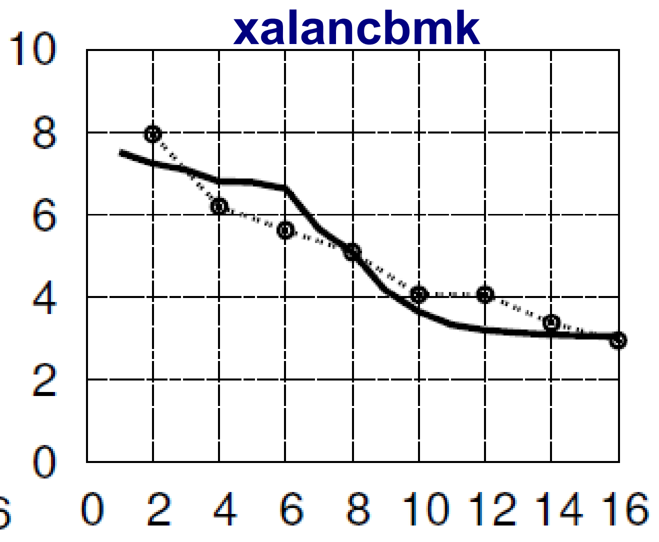
- Execution slice at 10 billion instructions

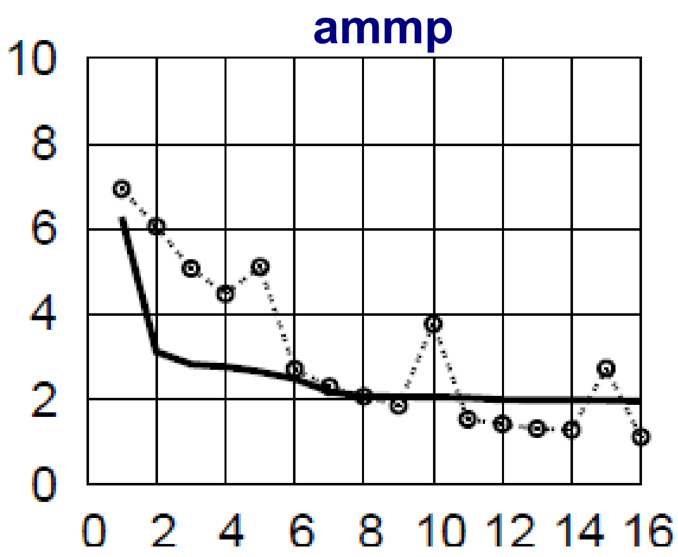




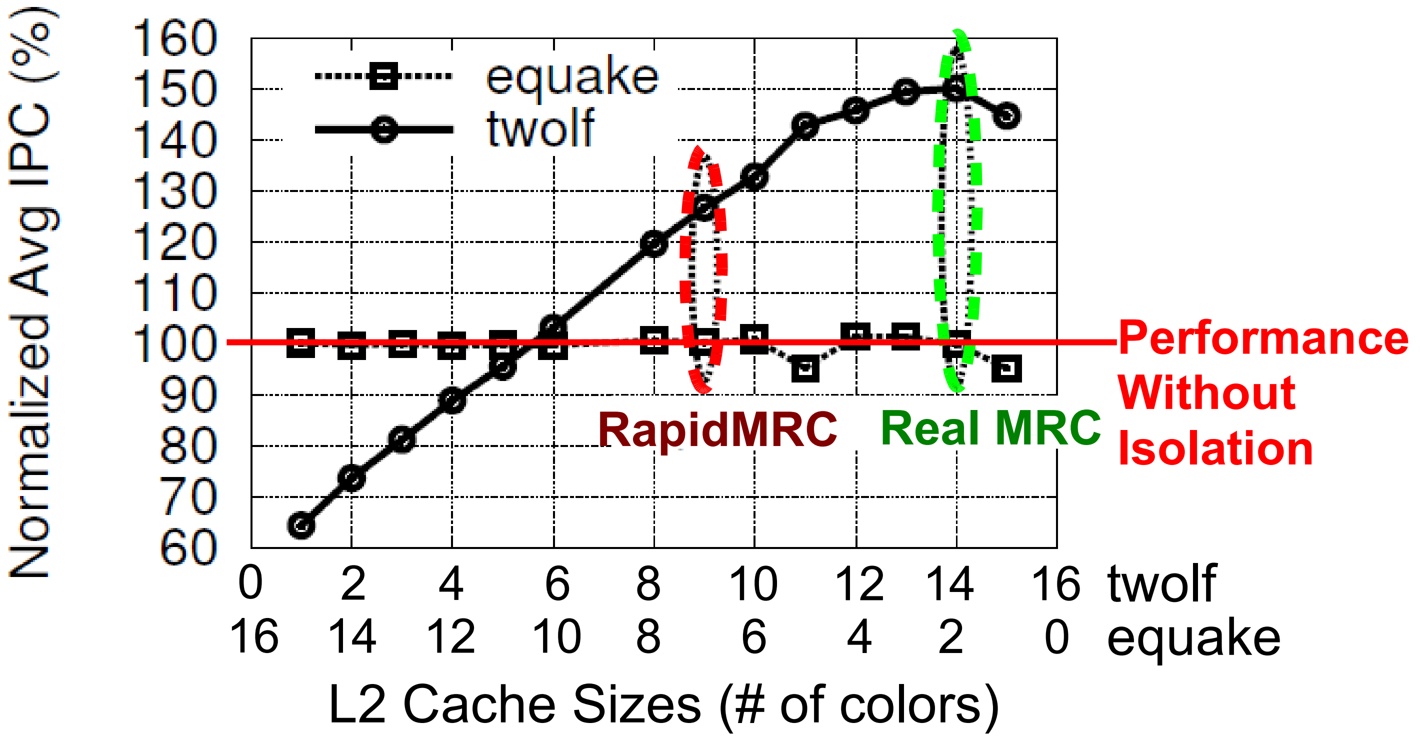








Effectiveness on Provisioning



Performance of Other Combos Using RapidMRC

- 12% improvement for vpr+applu
- 14% improvement for ammp+3applu

Contributions

On commodity multicores, first to demonstrate

- **Mechanism**: To detect data sharing online & automatically cluster threads
- **Benefits:** Promoting sharing [EuroSys'07]

- **Mechanism:** To partition shared cache by applying page-coloring
- **Benefits:** Providing isolation [WIOSCA'07]

- **Mechanism:** To approximate L2 MRCs online in software
- **Benefits:** Provisioning the cache [ASPLOS'09]

...all performed by the OS.

Concluding Remarks

Demonstrated Performance Improvements

- **Promoting Sharing**
  - 5% – 7% SPECjbb2k, RUBiS, VolanoMark (2 chips)
  - 14% potential: SPECjbb2k (8 chips)

- **Providing Isolation**
  - 4% – 17% 8 combos: SPECcpu2k, SPECjbb2k (36MB L3 cache)
  - 28%, 50% 2 combos: SPECcpu2k (no L3 cache)

- **Provisioning the Cache Online**
  - 12% – 27% 3 combos: SPECcpu2k

OS should manage on-chip shared caches

Thank You

24-9=15 slides

Future Research Opportunities

Shared cache management principles can be applied to other layers:

- Application, managed runtime, virtual machine monitor

Promoting sharing

- Improve locality on NUMA multiprocessor systems

Providing isolation

- Finer granularity, within one application [MICRO'08]
  - Regions
  - Objects

RapidMRC

- Online L2 MRCs
  - Reducing energy
  - Guiding co-scheduling

- Underlying Tracing Mechanism
  - Trace other hardware events