Shared Resource Partitioning on Chip Multiprocessor (CMP)

Carole-Jean Wu and Margaret Martonosi
Department of Electrical Engineering, Princeton University, Princeton, NJ

Motivation

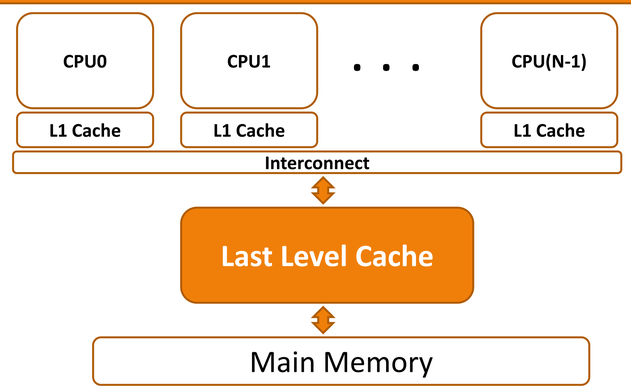
- Multiprogrammed applications sharing CMPs
  - Performance isolation between applications
  - **Resource sharing**
- Scarce on-chip shared resources
  - **Last level cache**
  - Bus bandwidth
- Memory footprints of different applications show different characteristics
  - Inefficient space utilization
  - Poor aggregate throughput

Objective

i. Effective shared resource managements
   - **Improve aggregate throughput**
   - Achieve quality of service (QoS)

ii. Flexibility and Scalability
   - Large-scale chip multiprocessor (CMP) platforms

System Configuration



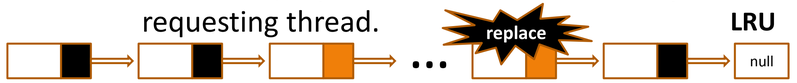
References

[1] R. Iyer, CQos: A Framework for Enabling QoS in Shared Caches of CPU Platforms. ICS'04
[2] K. Nesbit, J. Laudon, and J. Smith, Virtual Private Cache. ISCA'07
[3] P. Petoumenos et. al., Modeling Cache Sharing on Chip Multiprocessor Architectures. IEEE'06

Two Possible Approaches

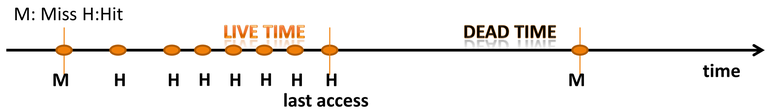
Way-Partitioned Replacement

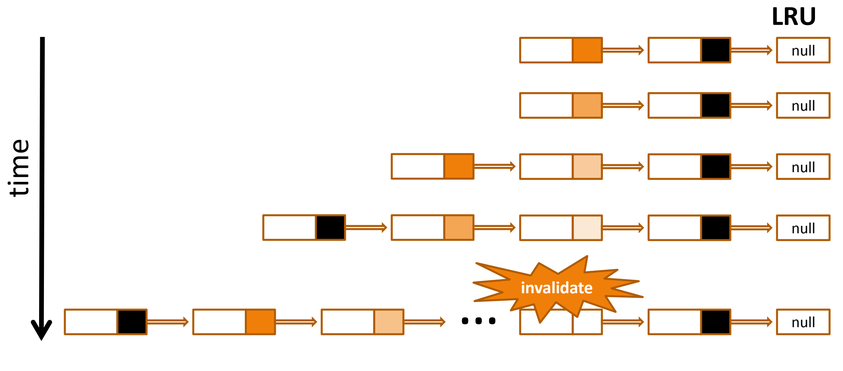
- Statically allocate number of L2 cache ways to threads
  - Replace the least recently used (LRU) line owned by thread i, such that thread i occupies more than its share.
  - Otherwise, replace the LRU line owned by the requesting thread.

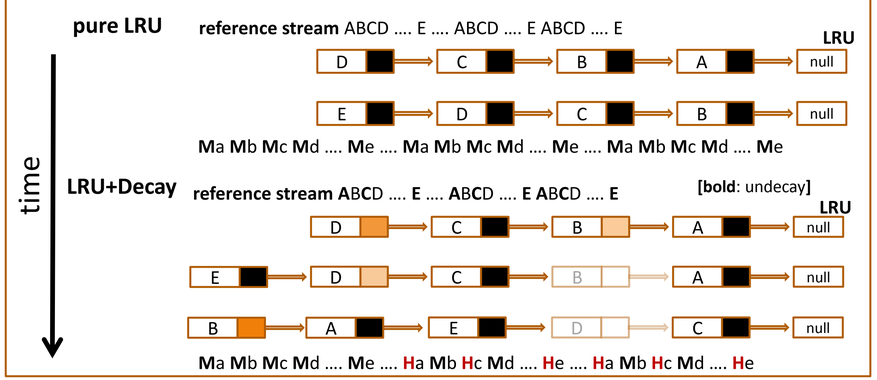


Decay-based Management

- Cache lines decay (invalidate itself) after a certain period of inactive time for replacement
- Restrict the "active ratio" of an application in shared resources



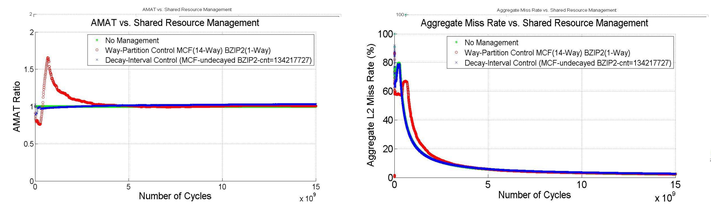




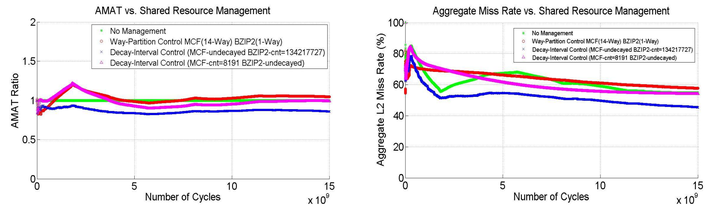
Evaluation

- Baseline: No Resource Management
- Way Partitioned Replacement
- Decay-based Management

No Resource Contention (16MB L2 Shared Cache)



Resource Contention (8MB L2 Shared Cache)



Conclusion

- Way-Partitioned Replacement
  - good performance isolation
  - good QoS
    - **What if…** number of concurrent processors >> number of way set associativities ?
  - same or worse aggregate throughput
- Decay-based Management
  - **better** aggregate throughput
  - **flexible** resource management
  - **scalable** large-scale multiprocessor systems

Ongoing Work

- Dynamically detect and adjust decay intervals based on workload characteristics

Acknowledgement

This work was supported in part by the MARCO/DARPA Gigascale Silicon Research Center (http://www.gigascale.org).