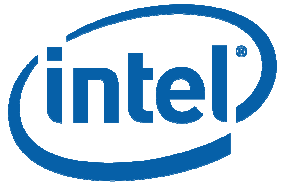


From Chaos to QoS: Case Studies in CMP Resource Management

Hari Kannan, Fei Guo, Li Zhao, Ramesh Illikkal, Ravi Iyer, Don Newell, Yan Solihin, Christos Kozyrakis

Intel Corporation Stanford University North Carolina State University

Outline

- Motivation and Problem Statement
- QoS in Resource Management
  - Platform QoS
  - Case Study: Cache Resource
- Experiments and Evaluation
  - Prototype Implementation
  - Initial Results
- Summary

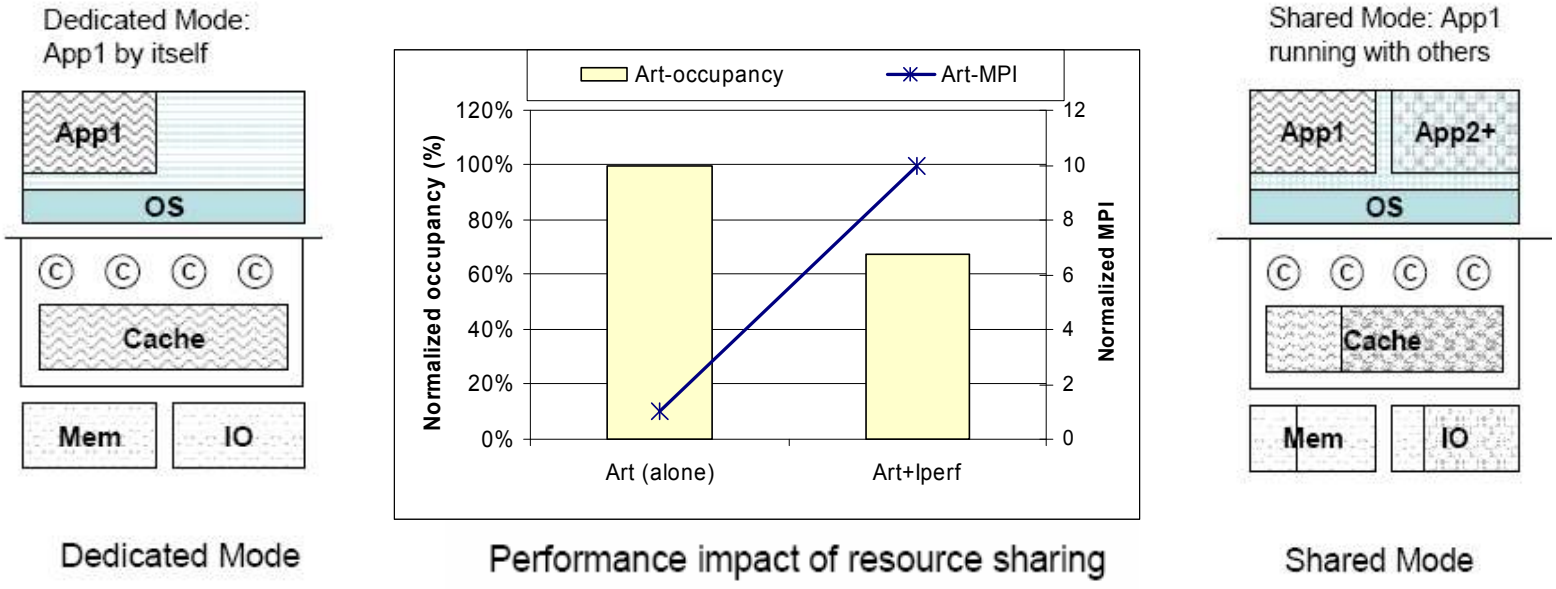
Motivation

- More cores are integrated on the die
  - Multi-tasking becomes more common: multiple applications are running simultaneously
  - Virtualized workloads become mainstream: multiple VMs are consolidated onto the same platform
- Problems in platform resource management
  - Loss of efficiency
    - Disparate Behavior or Disparate Resource Usage of simultaneously-running applications/VMs
  - No fairness or determinism guarantees
    - Cache space or memory bandwidth available is non-deterministic
  - No prioritization
    - Cannot map priority level to platform resource allocation

Problem Statement

- Not all applications are equal - Users have preferences
  - End-users (client) want to treat foreground preferentially
  - End-users (server) want to provide service differentiation
- Priority-based OS scheduling no longer sufficient
  - With more cores, OS will allow high and low priority applications to run simultaneously
  - Low priority applications will steal platform resources from high priority apps → loss in performance & user experience
- Platform has no support for application differentiation
  - No knowledge of user preferences
  - No support for preferential treatment

Example of Resource Sharing Impact



- Choose art as high priority and iperf as low priority
- High priority application suffers 10X more cache misses
  - Iperf has poor cache locality thus thrashes the cache
- Need for the platform to comprehend the priority of applications so that it can allocate hardware resources accordingly

Investigate QoS policies and mechanisms to efficiently manage these shared resources in the presence of disparate applications.

Outline

- Motivation and Problem Statement
- QoS in Resource Management
  - Platform QoS
  - Case Study: Cache Resource
- Experiments and Evaluation
  - Prototype Implementation
  - Initial Results
- Summary

Resource Management

Capitalist

- No management of resources
- If you can generate more requests, you will use more resources
- Grab as you will
- ***E.g. All of today's policies***

Communist/Fair

- Fair distribution of resources
- Give equal share of resources to all executing threads
- Does not necessarily guarantee equal performance
- ***E.g. Partitioning resources for fairness and isolation***

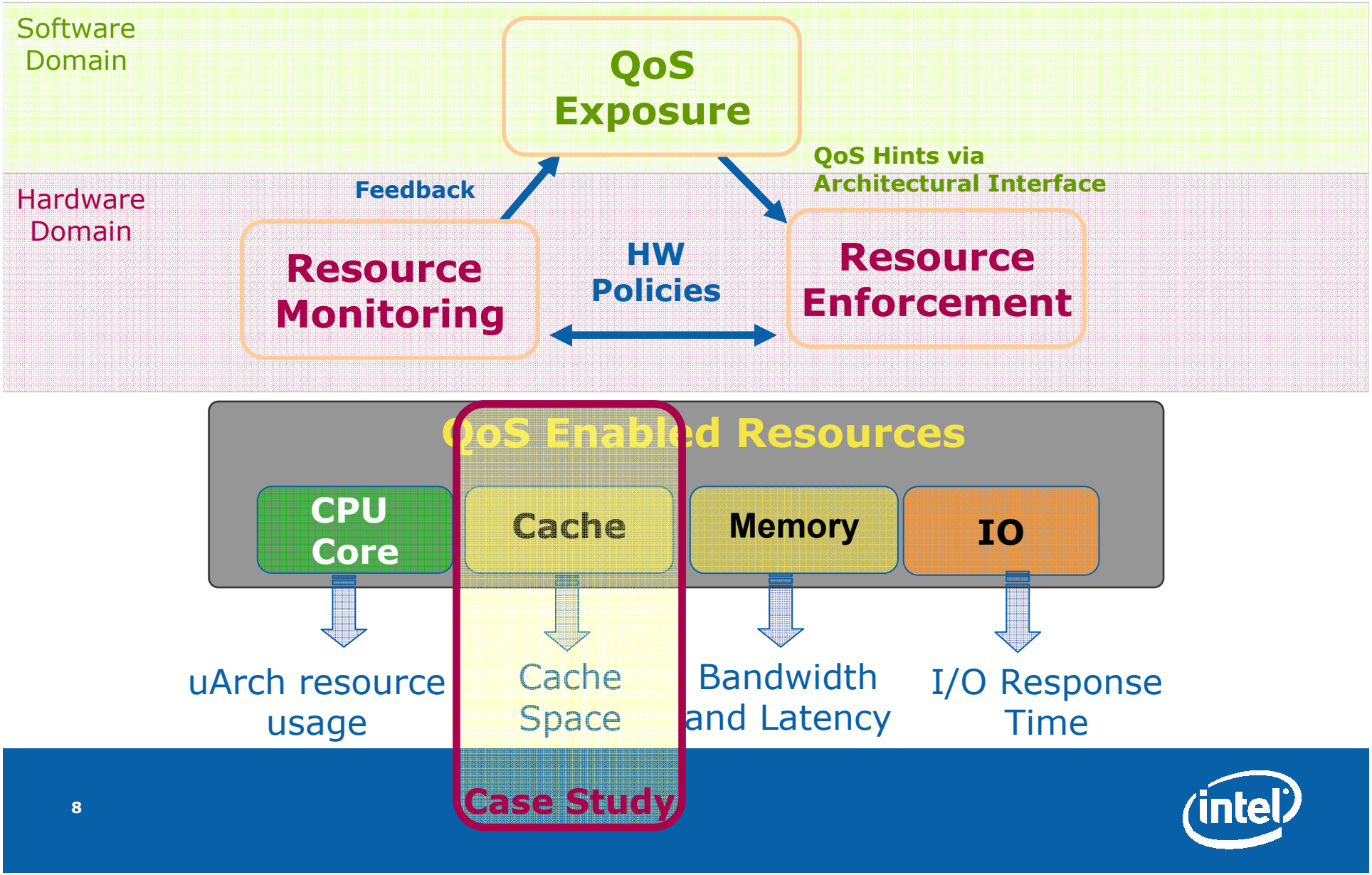
Elitist

- Focus on individual efficiency
- Provide more performance and resources to the VIP
- Limited resources to non-VIP
- ***E.g. Service Level Agreements, Foreground/Background***

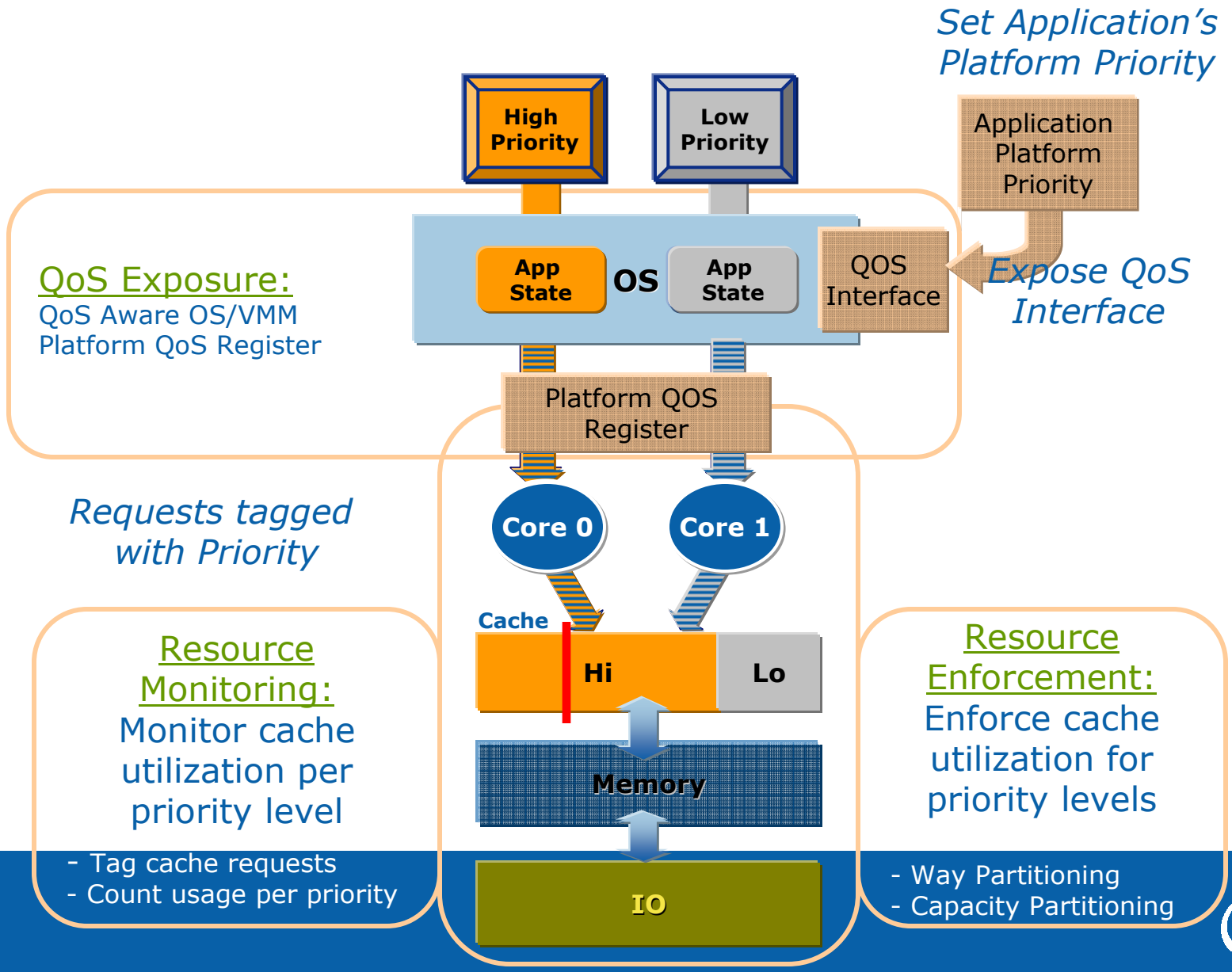
Utilitarian

- Focus on overall efficiency
- Give more resource to those that need it the most, less to others
- ***E.g. Cache-friendly vs. Unfriendly, resource-aware scheduling***

Platform QoS

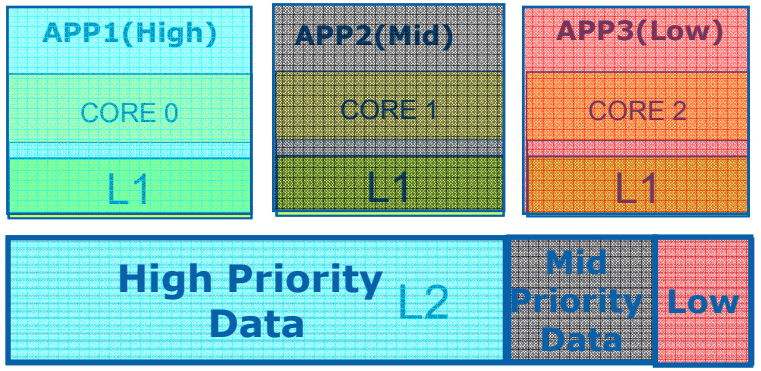


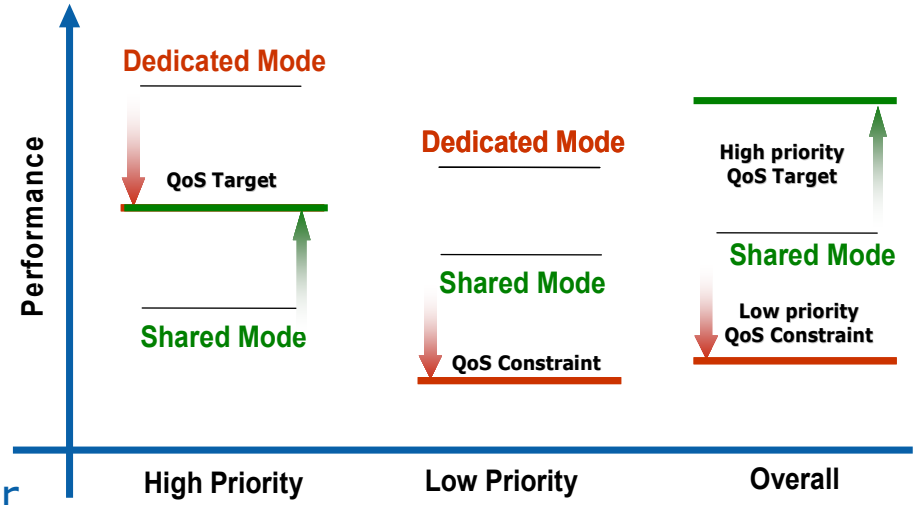
QoS Aware Architecture



Cache QoS Polices & Metrics

- Static
  - N priority levels supported in platform
  - Set a threshold of cache usage for each priority level
- Dynamic
  - Monitor cache space usage & performance metric at frequent intervals
  - Dynamically Adjust based on
    - **QoS Targets:** The extent to which high priority application should be improved
    - **QoS Constraints:** The allowable degradation to low priority application or the overall performance
  - Metrics
    - Resource performance (Miss Rate or **MPI**, etc)
    - Overall performance (IPC, etc)

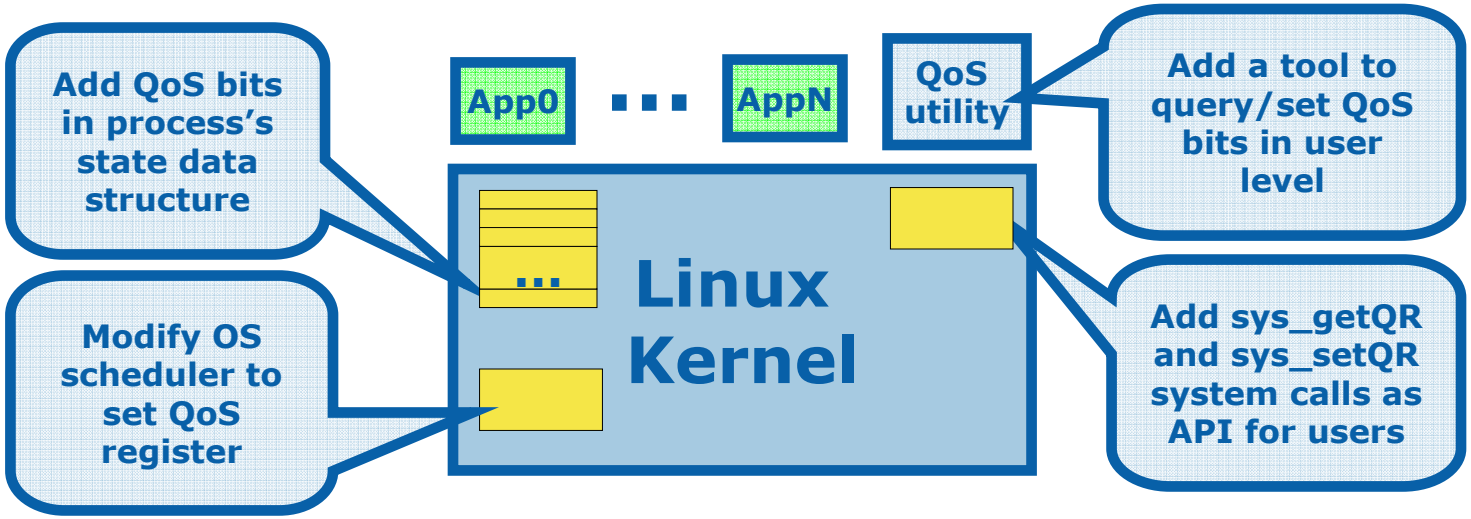




Outline

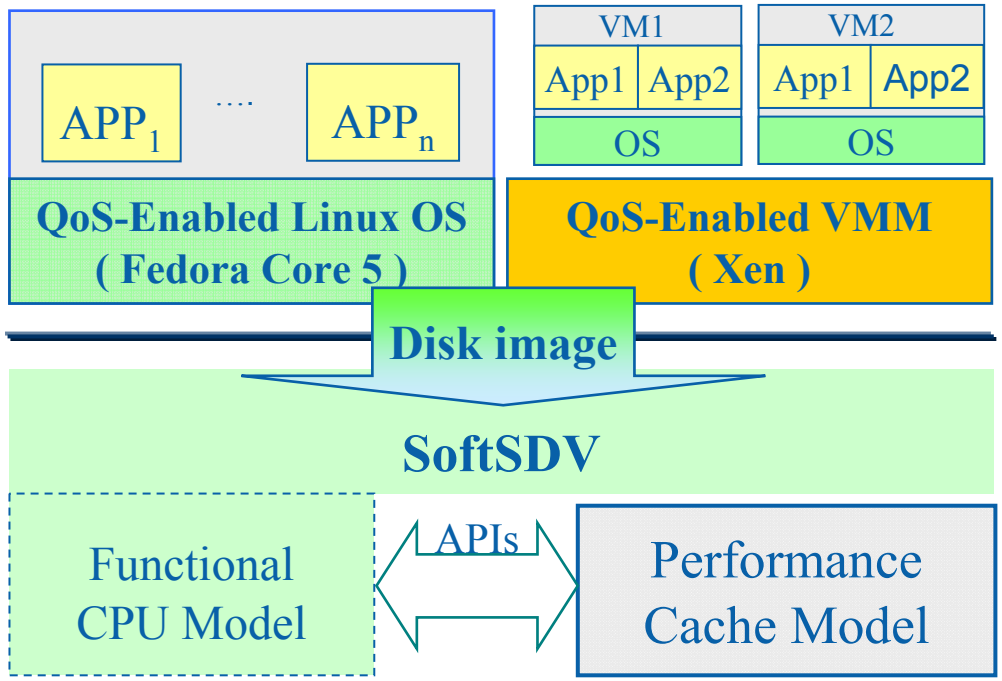
- Motivation and Problem Statement
- QoS in Resource Management
  - Platform QoS
  - Case Study: Cache Resource
- Experiments and Evaluation
  - Prototype Implementation
  - Initial Results
- Summary

Prototype -- QoS aware OS/VMM



- Add QoS bits -- indicate the priority level of the application
- Set QoS register in the platform
  - Special I/O instruction during each context switch
- Priority level management

Simulation Framework



- **Software Prototype**
  - QoS-enabled OS: Fedora Core 5 Linux
  - QoS-enabled VMM: Xen with Suse 9.1 Linux
- **Full system simulation**
  - Employ SoftSDV to functionally model the architecture
  - Employ Casper as a cache simulator
    - Modified to support monitoring and cache space allocation using static/dynamic QoS policies

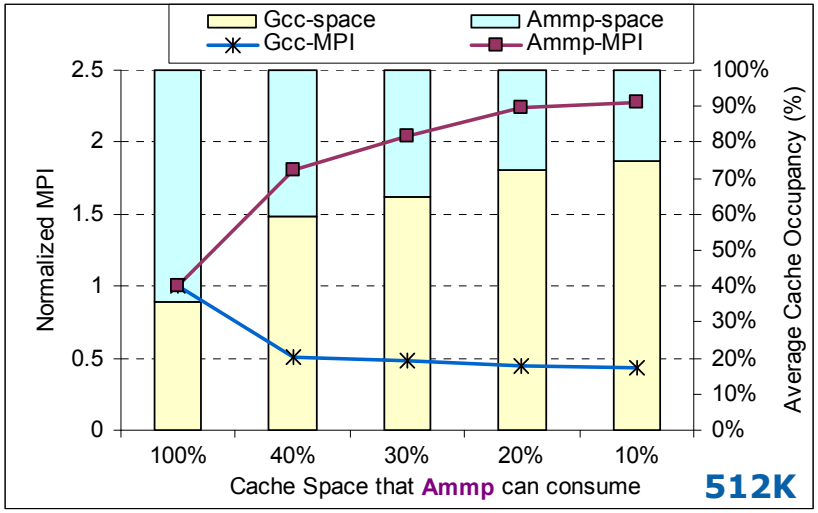
Evaluation Setup

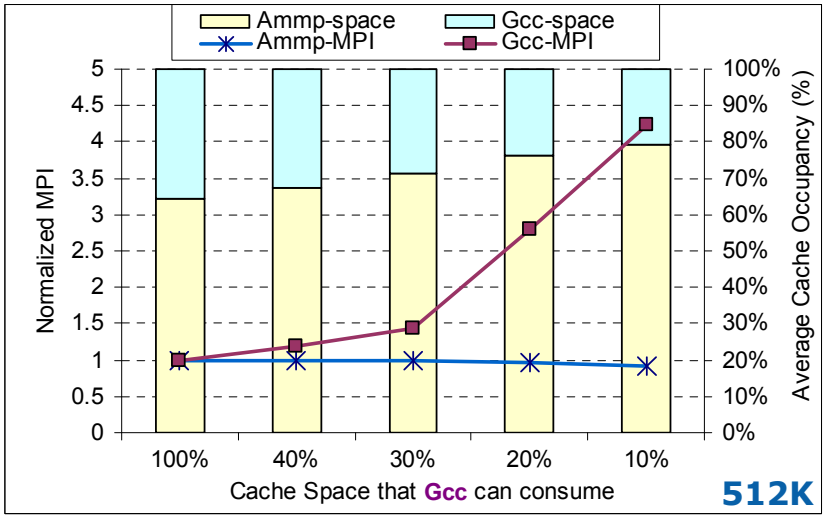
- Simulation configuration

Parameters	Values
Cores	1/2/4
L1 (Private)	Unified, 32KB, 16 way, 64B line, LRU
L2 (Shared)	Unified, 256/512/2048/4096/8192 KB, 16 way, 64B line

- Applications
  - QoS aware OS simulation
    - Spec2000 benchmarks (gcc, ammp, art, applu, mcf, mesa)
  - QoS aware VMM simulation
    - Spec2000 benchmarks (art, swim, mesa, bzip2)
    - Networking benchmark (Iperf)

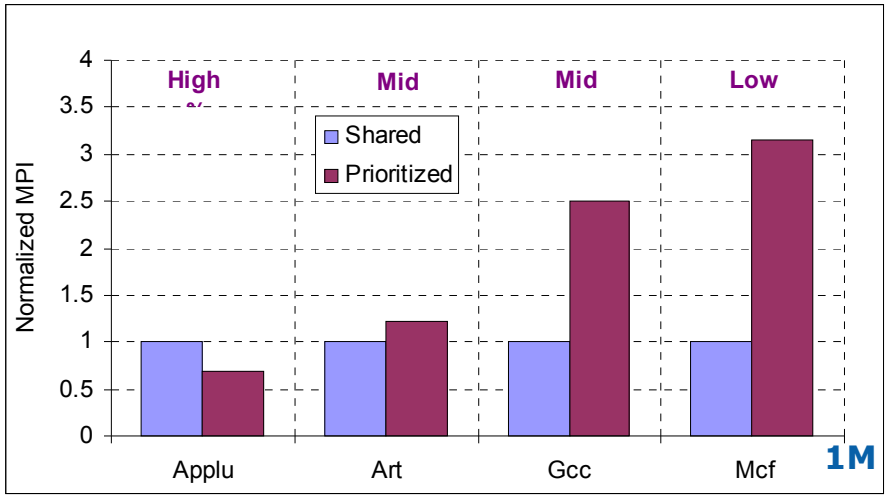
Static Policy on Two-core CMP

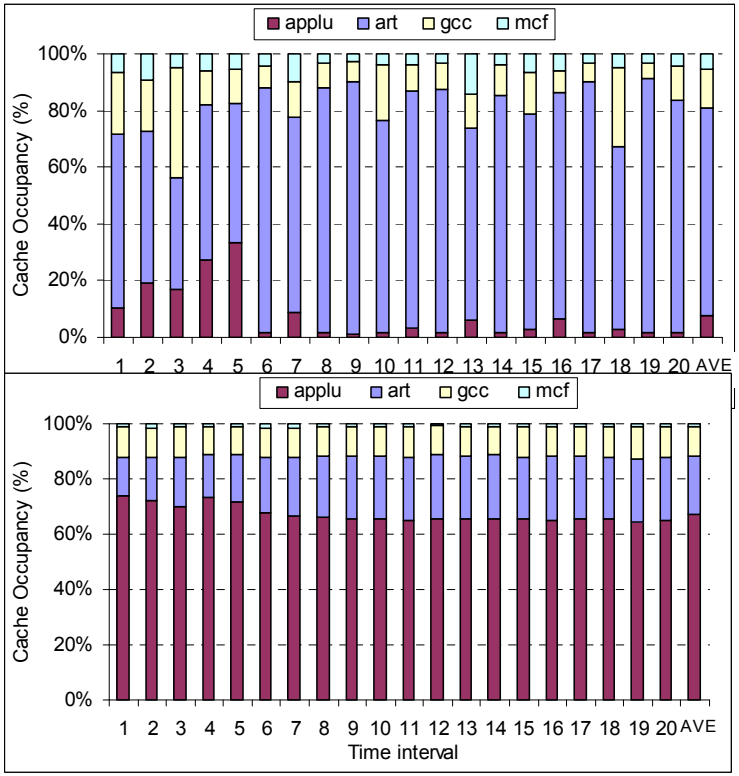




- As we reduce the cache space available for ammp, MPI for gcc is reduced
- Gcc benefits more from cache QoS than ammp
  - Probably because gcc is more sensitive to the cache size around this size range
- Low priority application exceeds its threshold
  - Sharing between applications
  - Capacity partitioning
    - Try to find a victim of low priority if it exceeds its threshold
    - Replace a high priority cache line if set is full of high priority cache lines

Static Policy on Four-core CMP

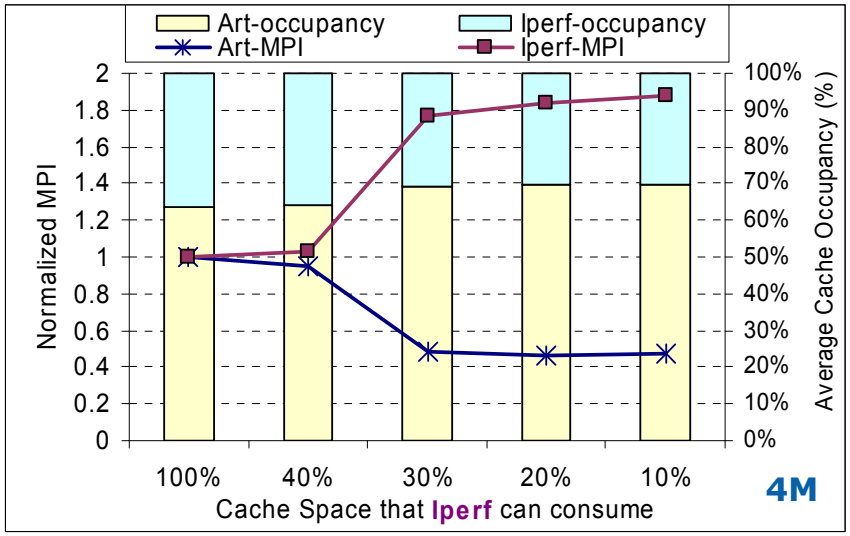


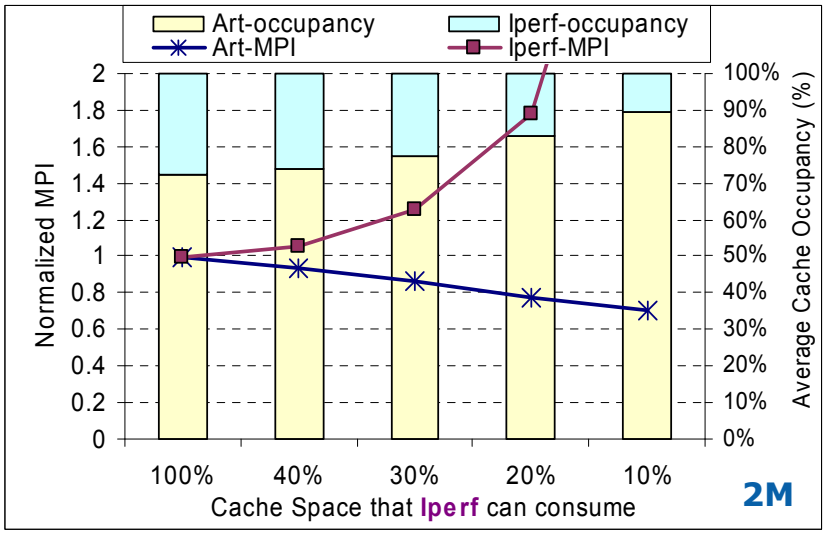


Set threshold for high, mid and low as 100%, 10%, 10%, 0%

- MPI for applu is reduced by 33%, MPI for art, gcc and mcf increases by 21%, 150% and 216% respectively
- Employing cache QoS can efficiently assign a deterministic amount of cache space to various applications.

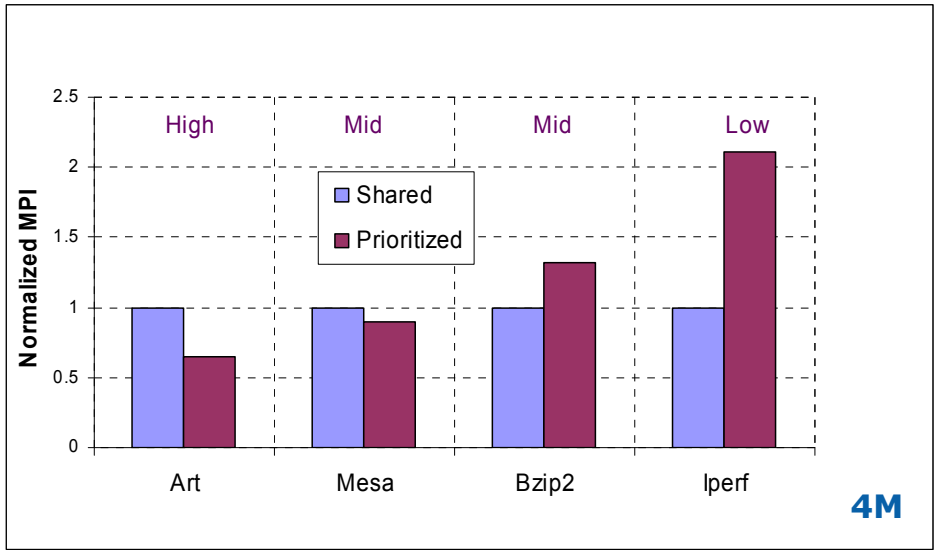
Static Policy on Two-VM CMP





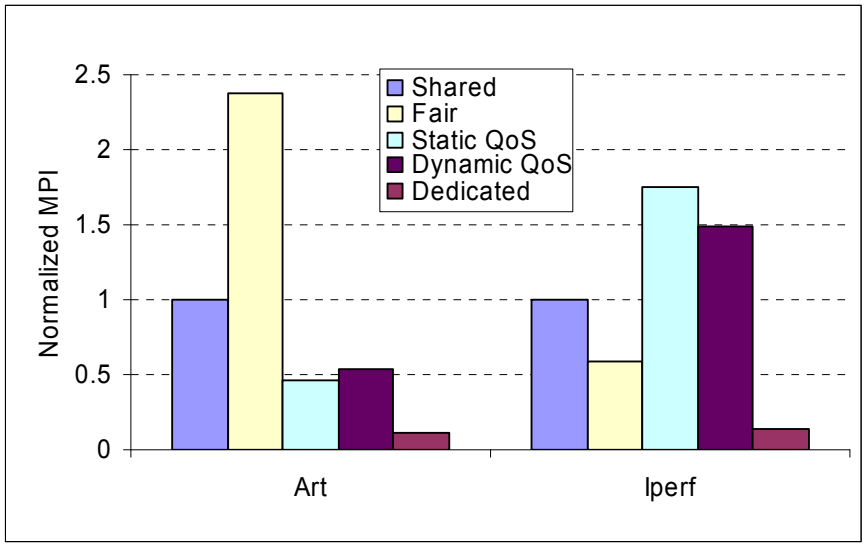
- Iperf (low pri) in Xen's Domain 0, art (high pri) in Domain 1
- 4M cache
  - MPI for art is reduced significantly
- 2M cache
  - MPI for art is reduced lineally
  - This is at the cost of iperf performance
  - The overall MPI increases significantly when iperf is limited to 10% of the 2M cache --> resource management of small caches could adversely affect the system's performance if the lower priority application is not allocated a minimum amount of cache space

Static Policy on Four-VM CMP



- Set threshold for high, mid, low as 100%, 20%, 20%, 10%
- MPI for art is reduced by 45%
- MPI for mesa is reduced by 10% because of the decreased interference from iperf
- Bzip2 gets some degradation
- Iperf sees more than 2x degradation

Comparison of Various Policies



- art as high priority and iperf as low priority
- Shared mode: without prioritization
- Fair mode: each application occupies half of the cache
- Static QoS mode: set threshold for iperf as 10%
- Dynamic QoS mode: QoS target, MPI = 0.5 of the shared mode
- Dedicated mode: application occupies the whole cache

Static and dynamic QoS are efficient to approach the performance improvement bound (the dedicated mode)

Outline

- Motivation and Problem Statement
- QoS in Resource Management
  - Platform QoS
  - Case Study: Cache Resource
- Experiments and Evaluation
  - Prototype Implementation
  - Initial Results
- Summary

Summary

- Motivate QoS-aware platform by showing case studies of CMP cache resource management
- Showed that it is important to provide better determinism in the platform that supports multi-tasking and virtualization
- Described QoS aware architecture and QoS policies
- Developed two software prototypes (QoS aware Linux and QoS aware Xen)
- Simulation results have shown that these techniques efficiently manage the platform resources towards better performance for high priority level applications

Future Works

- Experiment more with dynamic QoS mechanism
  - Detailed Specification of QoS targets & constraints
  - Other algorithms for dynamic schemes
- Study the impact of QoS on more diverse applications (servers, VMs, etc)
- Generalize QoS for all other CMP resources
  - Core, Memory, Interconnect, I/O, etc

Thank You!

Related Work

- *ICS 2004*
  - *Iyer, "CQoS: A Framework for Enabling QoS in Shared Caches of CMP Platforms"*
- *PACT 2004*
  - *Kim, Chandra, Solihin, "Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture"*
- *PACT 2006*
  - Hsu, Reinhardt, Iyer, Makineni, Newell, "*Capitalist, Communist and Utilitarian Policies: Shared Cache as a CMP Resource"*
  - Rafique, Lim, Thottethodi, "*Architectural support for OS-driven CMP cache management"*
- *MICRO 2006*
  - Qureshi, Patt, *"Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches"*
  - *Nesbit*, Aggarwal, Laudon, Smith, "*Fair Queuing CMP Memory Systems"*
  - *Keshavan, et al, "Molecular Caches"*