Virtual Platforms

Hypervisor Methods to Improve Performance and Isolation properties of Shared Multicore Servers

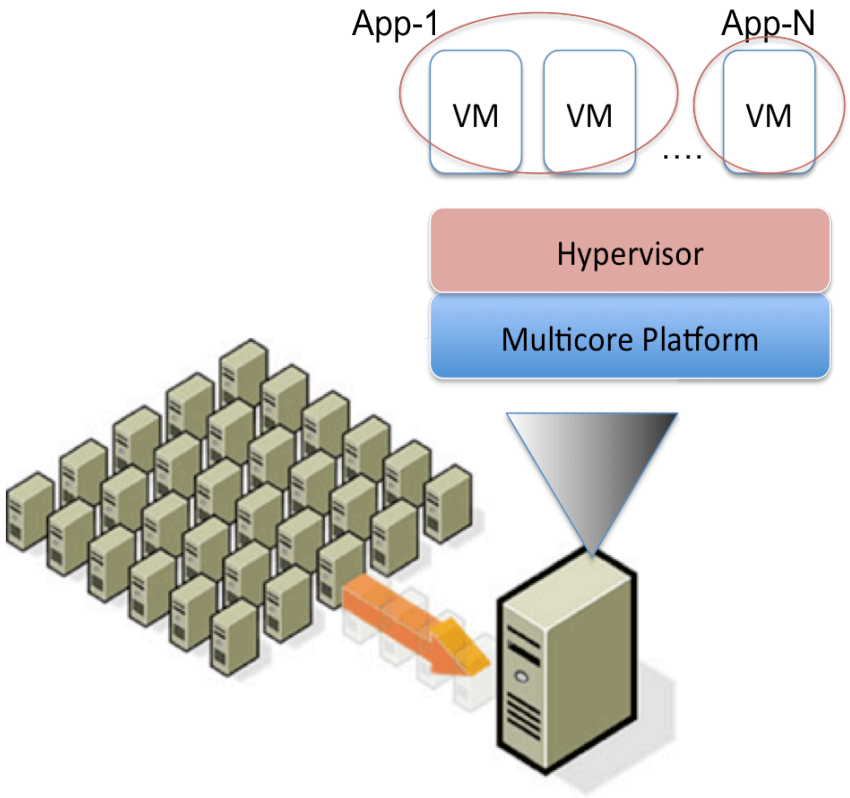
Priyanka Tembey

Ada Gavrilovska, Karsten Schwan

[School of CS, Georgia Tech]

Application Consolidation

- System Virtualization + increasing core-counts: Extreme consolidation
  - Hypervisors allocate CPU and Memory shares to application-VMs
  - Benefits Server resource utilization



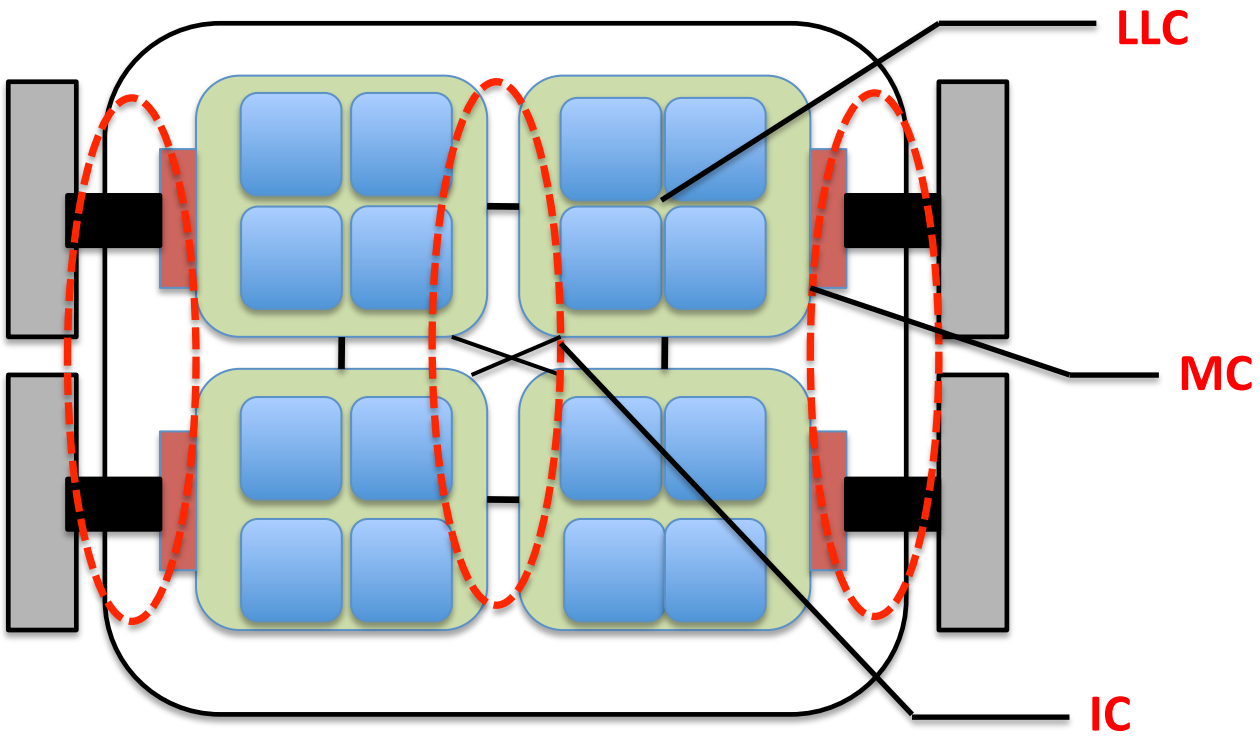
Consolidation: Performance effects

- Hypervisors limited in ability to provide performance isolation
  - Application performance depends on resources beyond CPU, Memory, includes shared resources: *Memory Bandwidth*, I/O*
  - Application resource requirements are *elastic*
  - Shared resources *not easily partitioned* as CPU, Memory in hardware
- Consolidation -> Arbitrary *interference* in shared resource shares which may have detrimental performance implications
  - Some applications more *sensitive* to interference than others

Consolidation: Performance effects

- Hypervisors limited in ability to provide performance isolation
  - Application performance depends on resources beyond CPU, Memory, includes shared resources: *Memory Bandwidth*, I/O*
  - Shared resources *not easily partitioned* as CPU, Memory in hardware
- Consolidation -> Arbitrary *interference* in shared resource shares which may have detrimental performance implications
  - Some applications more *sensitive* to interference than others

Isolation in hardware platforms: State of Art NUMA platforms



- Hardware partitions are coarse and not elastic
- Higher Consolidation levels -> Performance property violations due to arbitrary sharing of memory bandwidth including caches, Memory Controller (MC), Interconnect (IC)

Towards improving isolation using software methods

- Limited hardware support --
  - to create and isolate flexible shares of caches and memory bandwidth
- Can performance isolation of application properties be improved in shared server systems using software methods and how?
  - Also encouraging elasticity
- What are the limits to software methods?

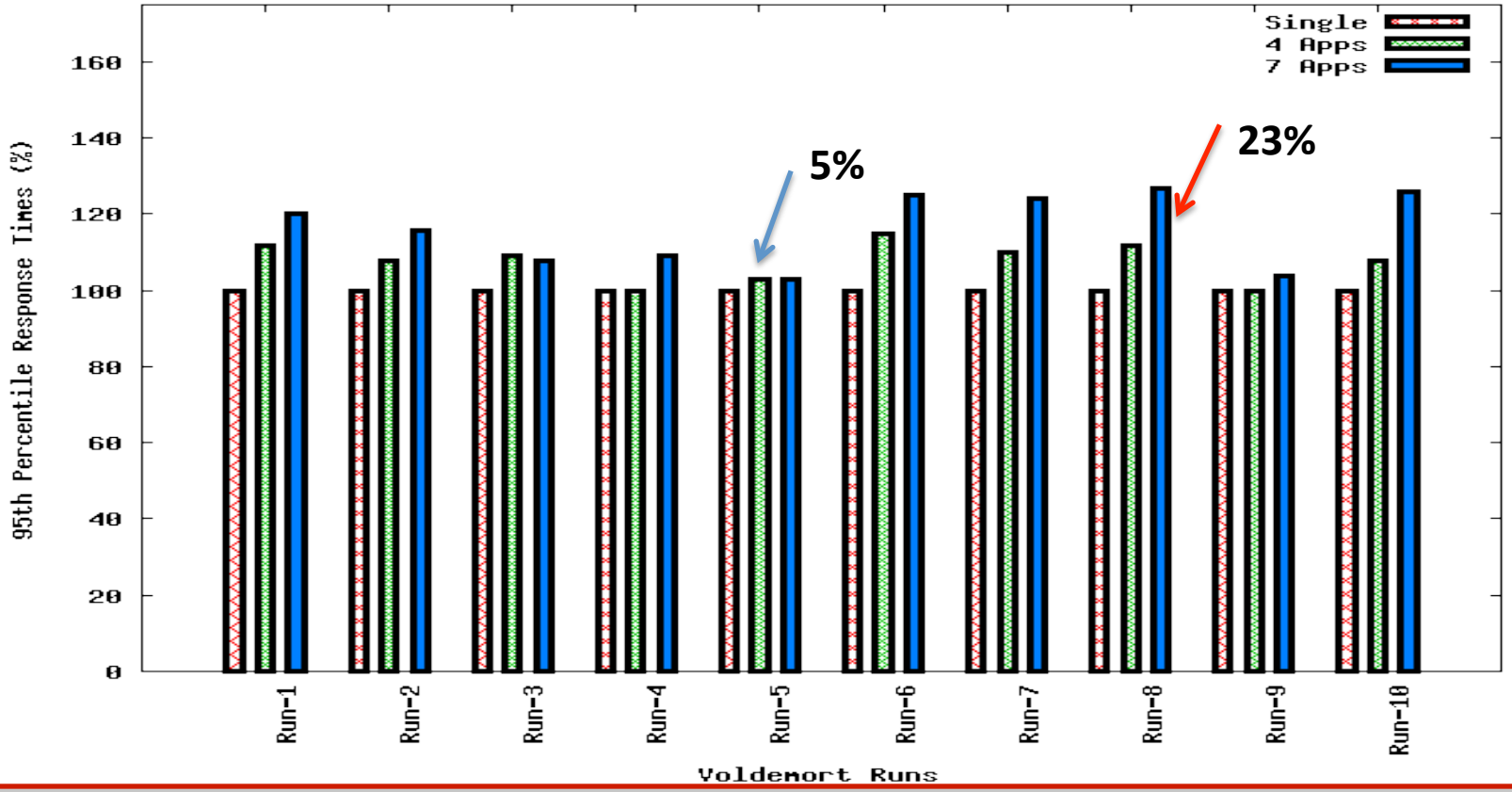
Experimental Motivation

- Hardware: 32 core Westmere Processor with 4 NUMA sockets (8 cores per socket), 32G RAM per NUMA node, 24MB LLC per socket
- Software: Xen 4.1 + Dom0 running 2.6.32 kernel + Guest VMs running Linux 3.0.2

Experimental Motivation

- Applications representative of "Cloud-mix":
  - **Voldemort server + YCSB workload client**: Key-value store used at LinkedIn, supports replication and in-memory backend (Multi-VM application)
  - **Phoenix Shared Memory MapReduce** with Pthreads (HPCA'07)
- Experiment scenarios:
  - **Single** (Baseline)
  - **4-Apps:** Voldemort + 2 Matrix-Mult + 1 WordCount
  - **7-Apps:** Voldemort + 3 Matrix-Mult + 3 WordCount
- Methodology: Run applications choosing distinct startup order for each run (different colocations)

Performance property variation in consolidated platforms: Voldemort



- 95th percentile response times of Voldemort show **unpredictable variation** across runs. Worst-case degradation is 23% (Run-8), **more sensitive** than MMult.
- Some configurations/co-locations better suited. E.g., (Run-5)
- **What hardware/software methods can help provide such isolation as in Run-5?**

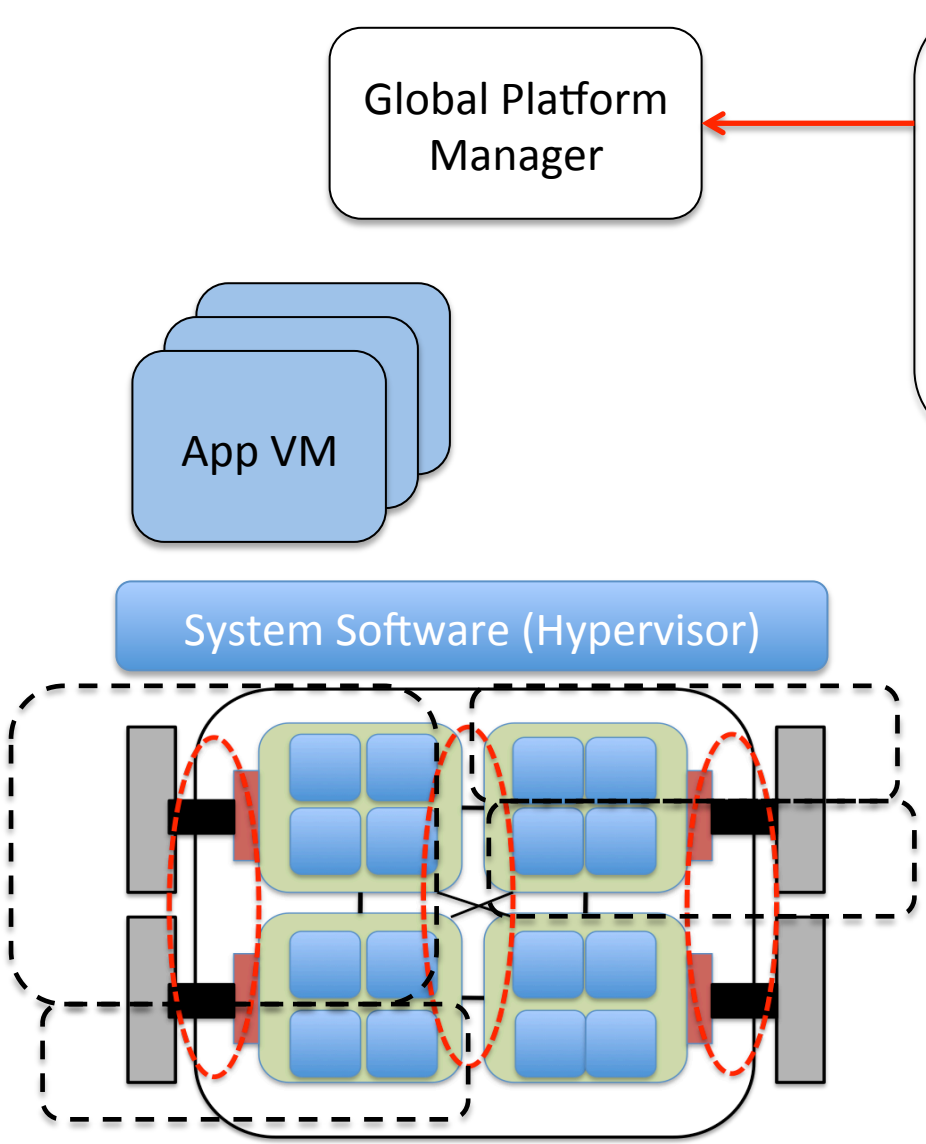
Our contribution: Virtual Platforms

- System-level methods to manage application resources keeping isolation as a first class resource management principal
  - **Online interference models** for shared resource points (Caches, MC, IC)
  - Create, allocate and maintain **Virtual Platforms (VPs) as hypervisor-level commitments of resource shares**
- Implementation in Xen Hypervisor evaluated with enterprise/cloud application mixes
  - Less performance variation, improved performance predictability

Agenda

- Motivation for VP-enabled Hypervisors
- Virtual Platform Architecture
  - Online Interference Model and software architecture
- System Evaluation
- Future Work

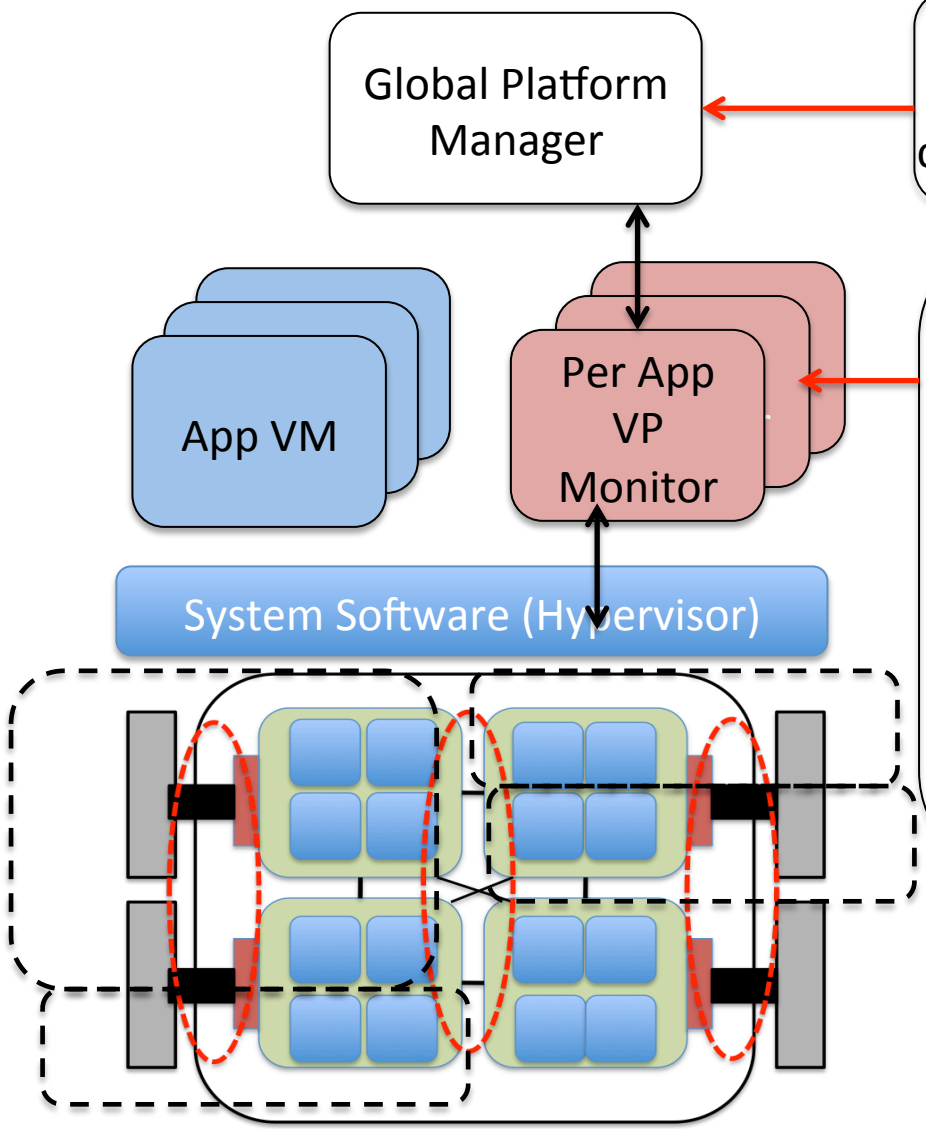
Virtual Platform (VP) Architecture



Global Platform Manager

- Creates App VMs and allocates initial CPU, Memory resource shares to create a VP
- Topology-awareness of VP to Platform resources (colocated VPs knowledge w.r.t LLC, MC, IC)

Virtual Platform (VP) Architecture



Global Platform Manager

- Creates VP monitor per Virtual Platform and communication channels

Per-application VP monitor

- In absence of controls to define shared resource shares, VP monitors track application VM resource usage using black-box techniques (hardware perf. counters)
- How ***intensively*** an application uses CPU, caches and memory bandwidth (MC, IC)?
- How ***sensitive*** an application is to interference at these shared resource types?

Application Performance model

- Perf[k] = *CPU[k] + Memory[k] + Memory_bandwidth[k]* … (1)
- Perf[k] = *CPU[k] + Memory[k]*

  + b.Cache-usage[k]

  + b.Local-Memory-Bandwidth[k]

  + b.(Remote-Memory-Bandwidth, Remote-latency)[k] … (2)

(b: binary variable)

Modeling application resource use intensity

- Why measure how intensively an application uses shared resources?
  - Measure of its "contentiousness" at shared resource points in system
- Approximating resource share use
  - CPU: CPU utilization
  - LLC: Using L2 and L3 miss counters

  (L2-L3)/L2misses per 1000 instructions
  - Memory Bandwidth (MC/IC): L3miss/1000 instructions

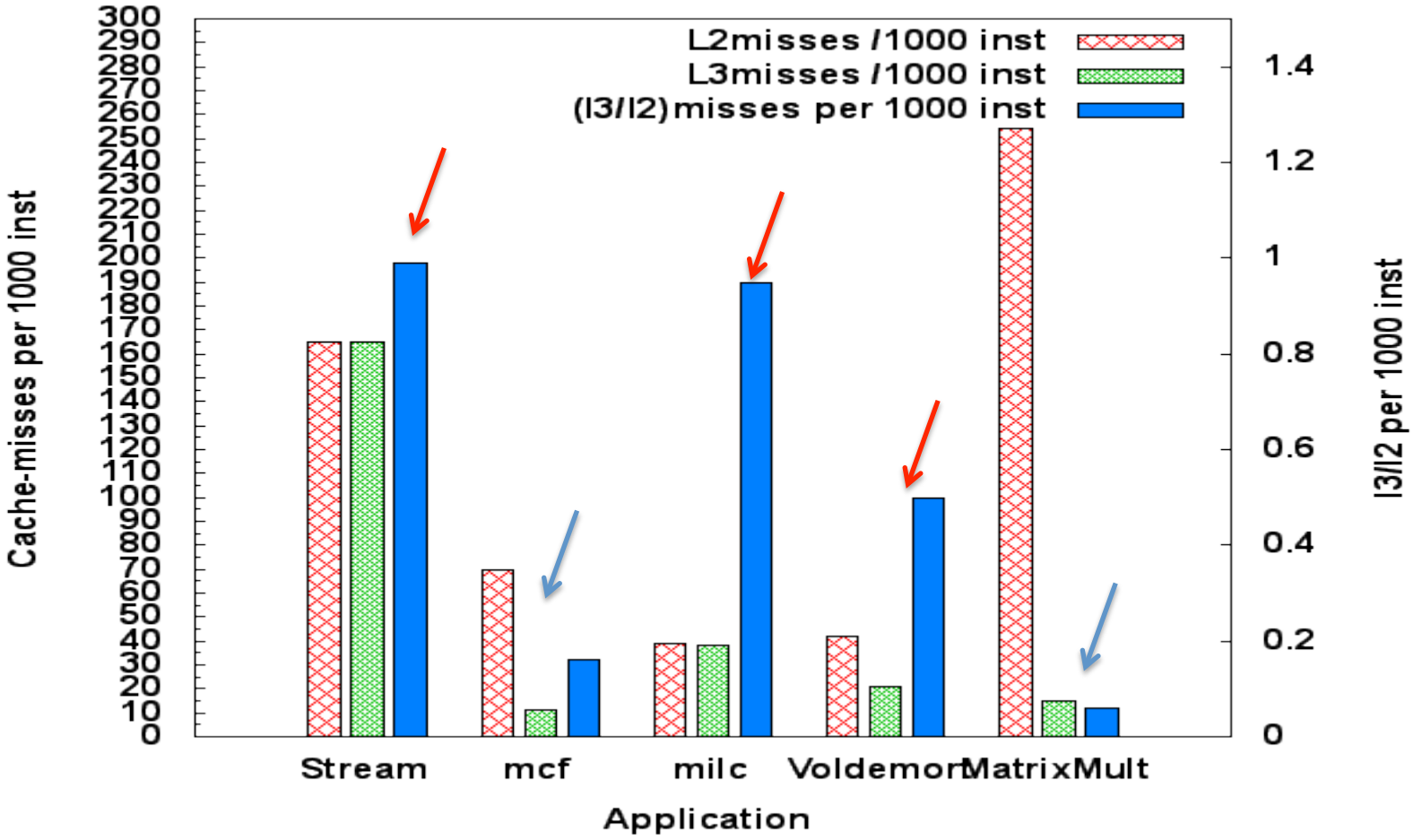
Modeling application resource use intensity – Memory Factor

- Memory factor of an application
  - Approximates application's cache vs. MC/IC use
  - L3/L2 per 1000 instructions: Fraction of L2misses served by memory (or MC/IC)
  - Higher MF: Higher use of MC/IC vs. cache
  - Useful to know which shared resource is more important (Cache or MC/IC)

Application mix characterization

- Memory intensity classes (L3misses/1000inst)
  - <2 (Pugs)
  - >2, <15 (Terriers)
  - >15 (Bulldogs)
- Memory Factor classes (L3/L2 per 1000inst)
  - <0.25
  - >0.25 , <0.6
  - >0.6

Application mix characterization



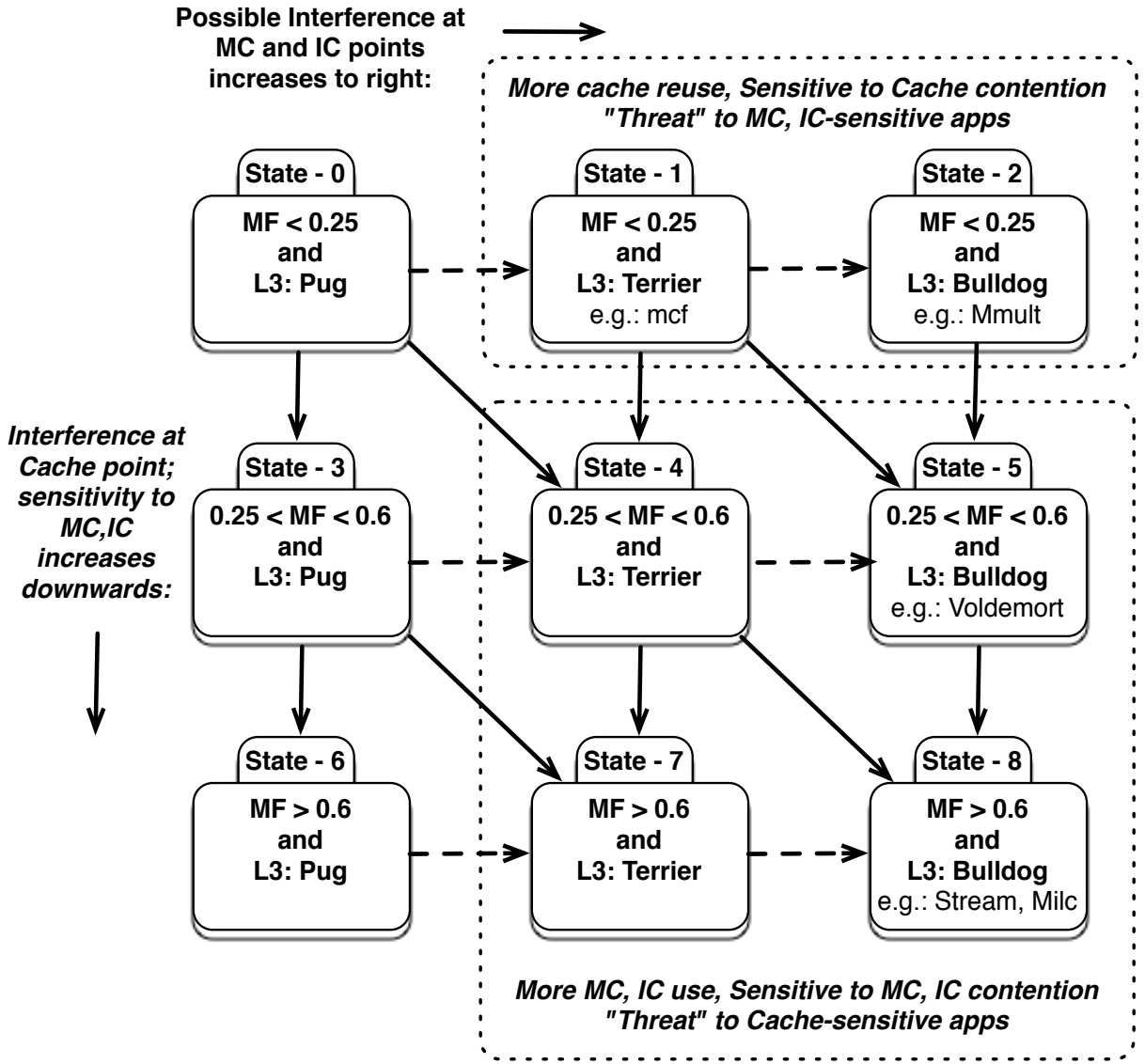
Memory intensive: Stream, Milc, Voldemort
Cache-intensive: Mcf, MatrixMult

Modeling application resource sensitivity

- Why measure sensitivity?
  - Measure of "hurt" caused to application due to contention at particular resource type
  - Hint for which resource share to isolate
  - E.g., A streaming application may be cache-intensive, not cache-sensitive
- Measuring **MC/IC contention sensitivity**:
  - Higher MF (> 0.25) and Terrier/Bulldog: Higher sensitivity to Memory latency and contention E.g: Milc
- Measuring **Cache contention sensitivity**:
  - Lower MF (< 0.25) and Terrier/Bulldog E.g: Mmult, Mcf

Modeling interference

- VP Monitor keeps track of Application states – (MF, L3misses)
- Transition down: LLC interference
- Transition right: Higher memory intensity
- Platform Manager notified on state transition
- Special case (State-8): Monitored by Global Platform Manager



Mitigating interference

- Mitigate interference: Maintain resource shares
- Global Platform Manager uses:
  - VCPU migration amongst NUMA nodes
  - VM page migration across NUMA nodes
  - CPU caps: Indirect control of Memory bandwidth use for highly memory-intensive applications within NUMA node
- Use interference model to guide "better" mitigation actions
  - E.g., Use [VCPU migration + Memory migration] for applications sensitive to MC/IC and remote latency

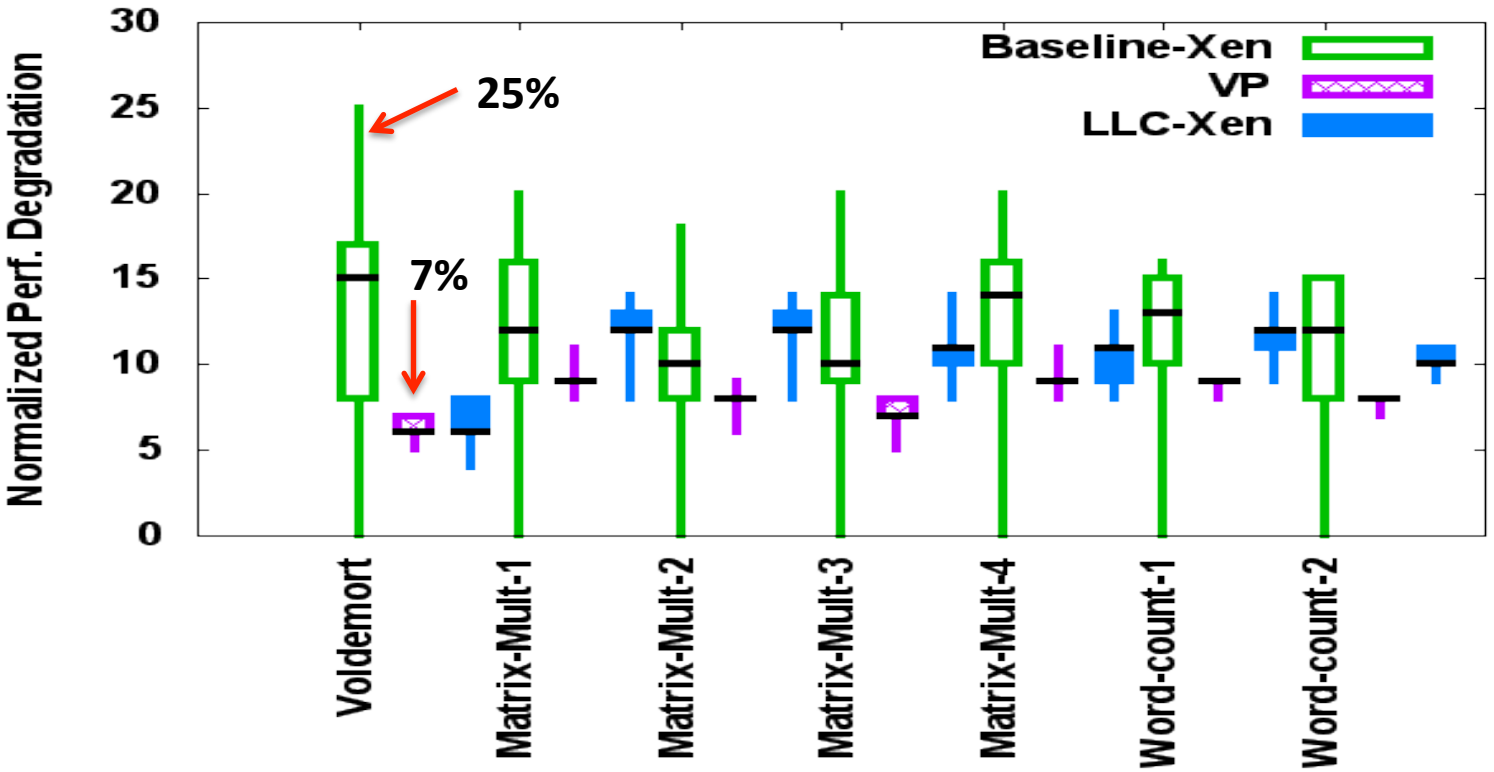
Evaluation

- Application mixes
  - Stream-SPEC (3 Stream + 2 Milc + 2 Mcf + 2 Lbm)
  - Voldemort-MapReduce (1 Voldemort + 3 Matrix-Mult + 3 Wordcount)
  - StreamingServer-MapReduce (1 StreamingServer + 3 MatrixMult + 3 Wordcount)
- 4-VCPU VMs + 4G Memory, no CPU sharing, prefetching disabled
- Why disabled Prefetching?
  - No software control
  - Hard to quantify use of memory bandwidth per application using performance counters

Experiment Methodology

- Baseline-Xen: Each application-mix executed in different startup order to remove colocation bias
  - Observe best performance and normalize other performance values to this case
- VP-Xen: Similar runs with VP-enabled Xen
  - Performance normalized to best performance in Baseline-Xen case
- LLC-Xen: Interference mitigated based only on LLC misses metric, no sensitivity

Voldemort-Mapreduce



- Voldemort has higher MC/IC sensitivity than Matrix-Mult and Wordcount
- Voldemort worst-case times improved almost 3 times when contention caused by Matrix-Mult is mitigated using VP methods
- ~8% overhead in moving applications to "good" configuration (as of the baseline)

Discussion

- VP methods and actions re-allocate resources to better isolate applications irrespective of initial configuration
  - Less performance variation across runs (40-50% worst case performance improvements)
- VP-Xen is more efficient -> "improved ability for extreme consolidation" on shared multicore servers
- Best case performance trails default best case: Motivation for better software/hardware interfaces
  - Model is heuristic-based, more usage counters
    - For practical deployments, models should not be too architecture-dependent (e.g., Memory Controller policies/queues). Generality vs. Specialization trade-offs
  - Interesting to explore software controlled finer-grained per-core prefetch

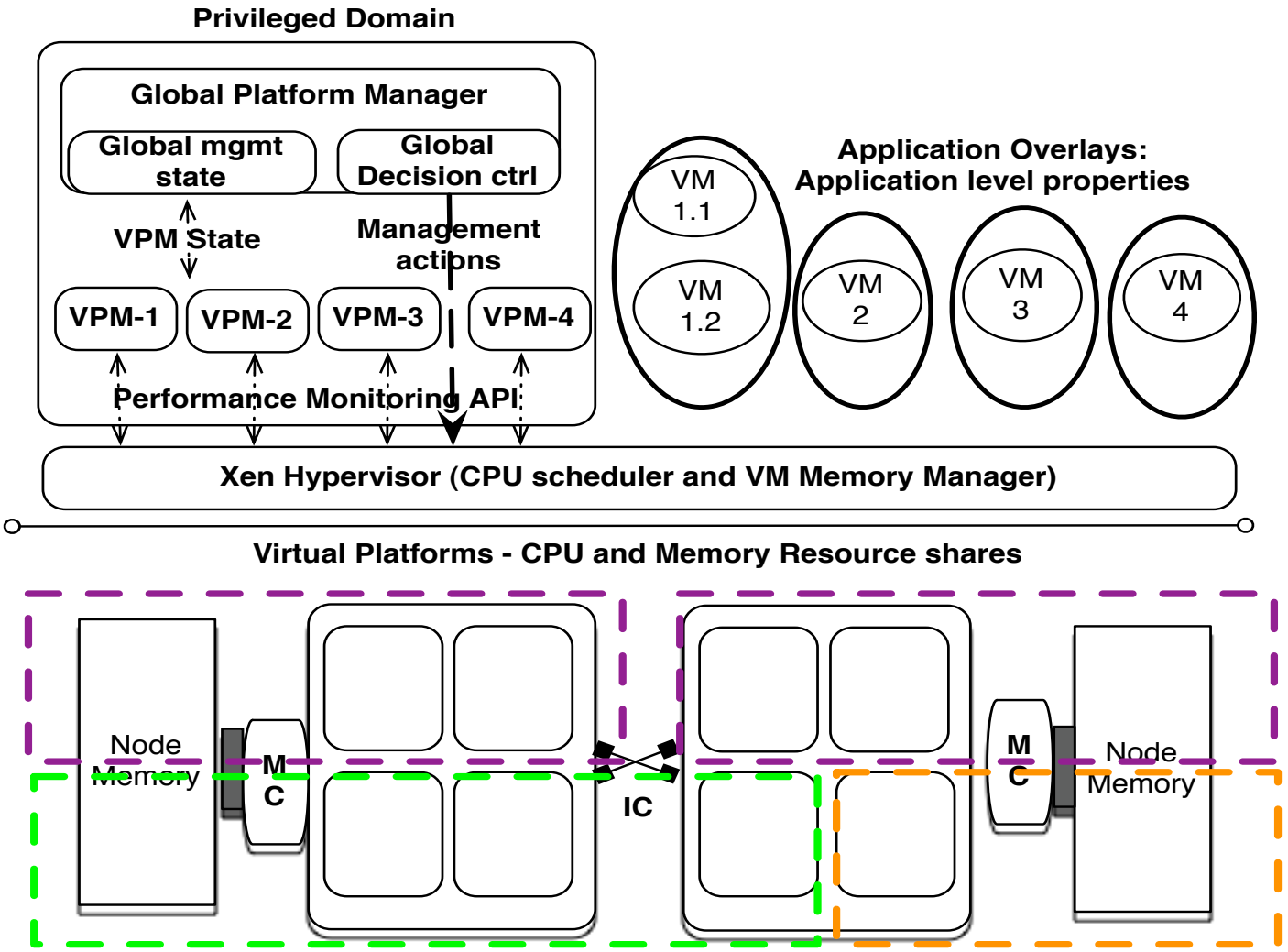
Future Directions

- Multi-dimension resource allocation and arbitration (CPU vs. Cache vs. Memory Bandwidth) in "extreme consolidation" cases
  - Applications with different, elastic resource intensities, sensitivities
  - How many applications can be 'efficiently' packed on to a platform?
    - What are the metrics to consider in making arbitration decisions along multiple resource dimensions?
    - E.g., CPU allocation -> Cache and Mem.Bandwidth use implications
  - Factoring in cost of dynamic management knobs
    - Page migration costs, resources consumed

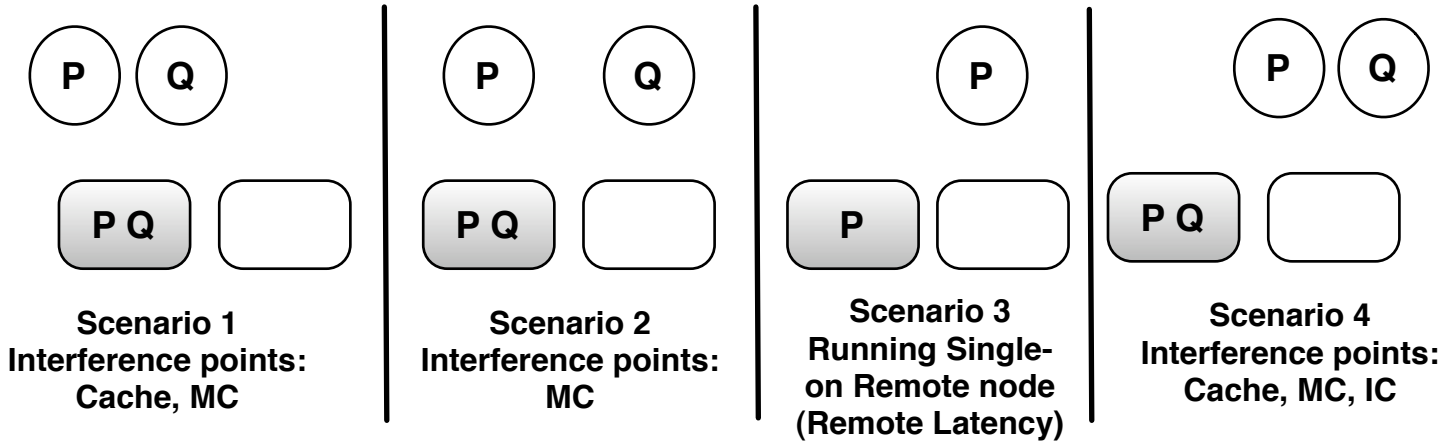
Thank you. Questions?

Georgia Tech

Implementation

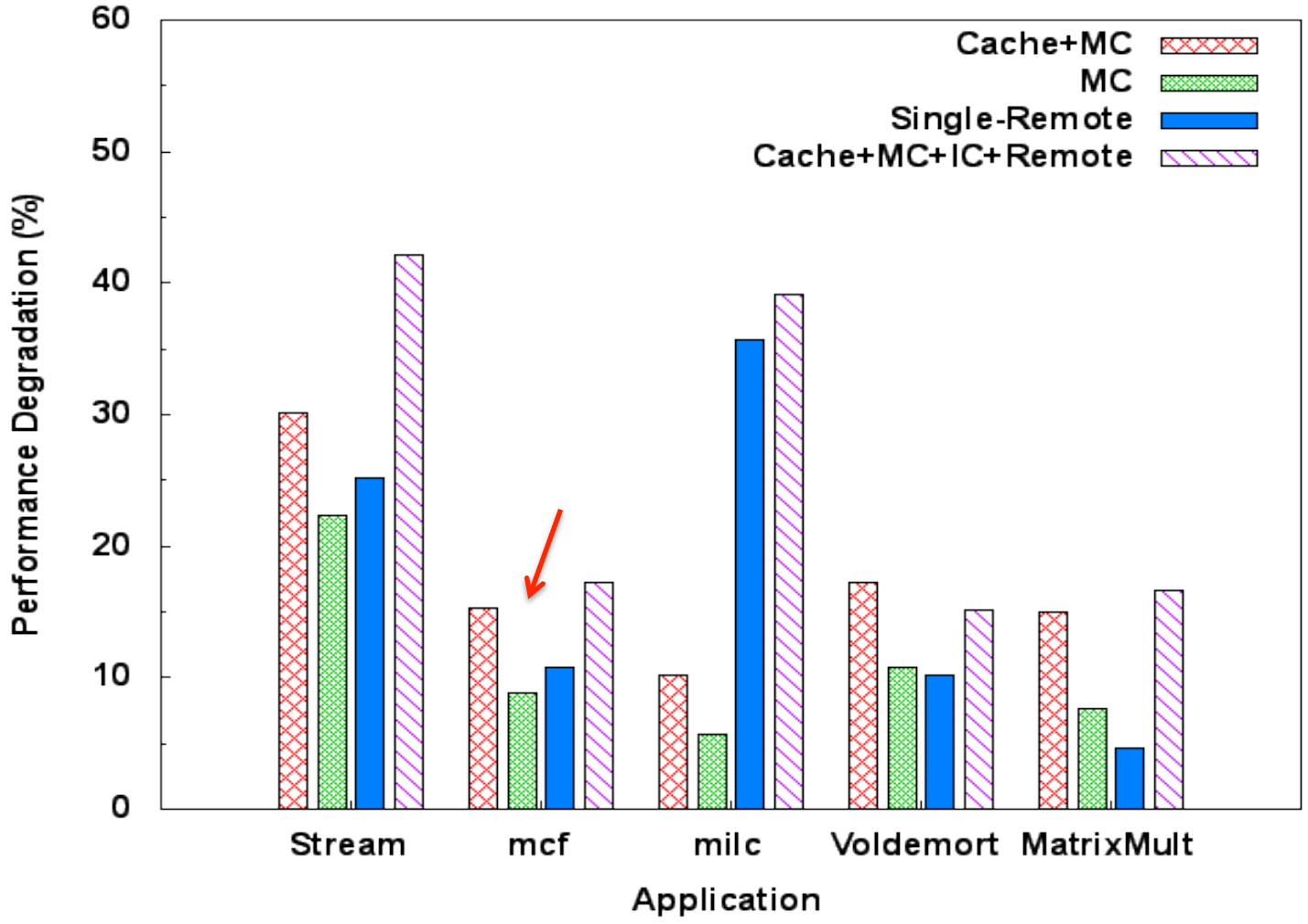


Validating Interference Model

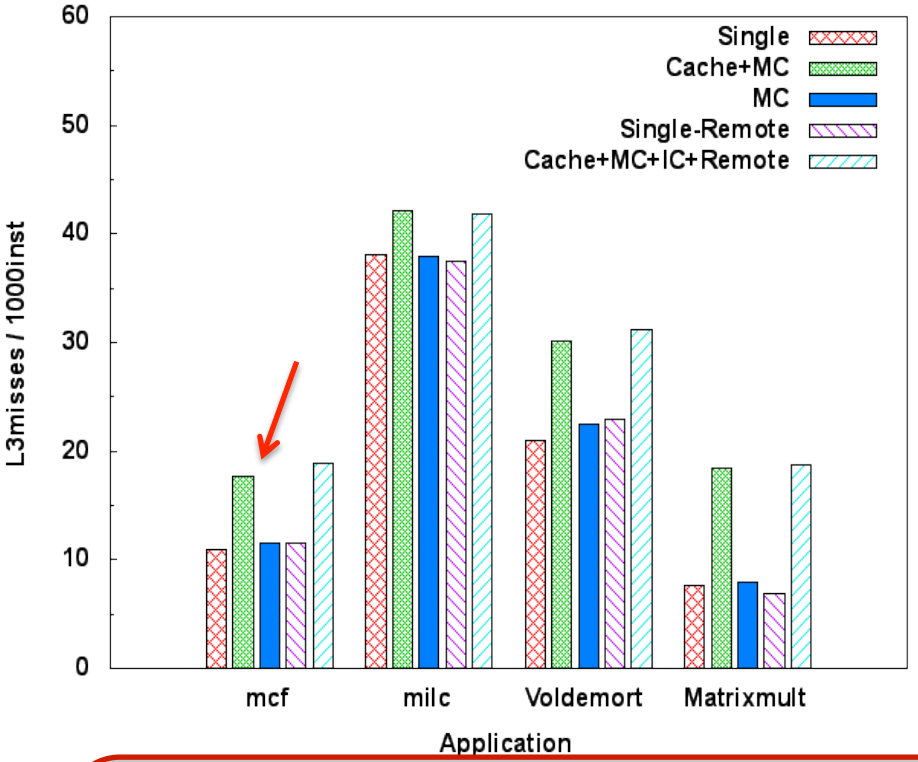


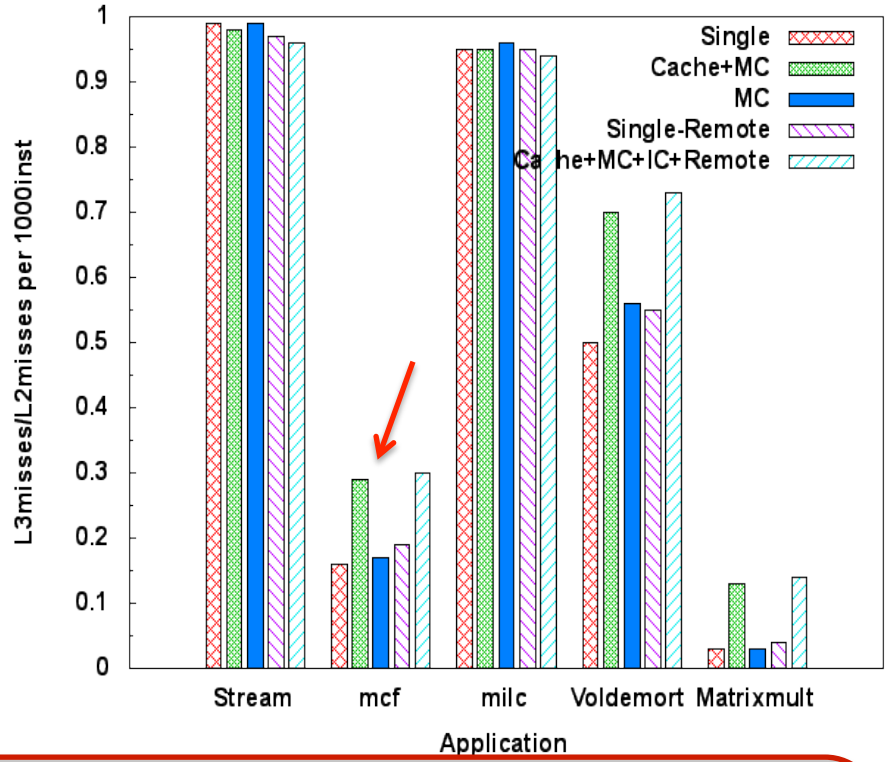
- Need to validate that the state-transition model can detect potential interference at shared resource points
- Experiment with different application colocation scenarios choosing different interference points
  - Isolating each interference point

Performance degradation due to interference



Performance degradation explained by state-transition model





- **Mcf: (Cache)** State-1 to State-5 with increased MF sensitivity and Memory intensity
- **Milc: (Memory)** State-8 application, not much variation at VP-monitor level, needs to be managed at global level
- **Voldemort: (Cache + Memory)** State-5 to State-8

Platform Efficiency metric

- Platform Efficiency (P.E): (Normalized Performance Improvement for all apps / Normalized CPU utilization)
- Higher platform efficiency: Less performance degradation using less platform resources
  - Improved ability for consolidation
- VP-enabled Xen shows consistently higher P.E metric

App Mix	CPU-Util (Xen)	CPU-Util (VP-Xen)	PE (Xen)	PE (VP-Xen)
Stream-SPEC	2376	2336	7.838	8.556
Voldemort-Mapreduce	1920	1900	7.793	8.205
StreamingServer-Mapreduce	1970	1950	7.603	8.064