

Q-Clouds: Managing Performance Interference Effects for QoS-Aware Clouds

Ripal Nathuji
Aman Kansal
Alireza Ghaffarkhah

Presented by Joshua Davis

Motivation and Background

- Cloud computing
 - Off load processing and storage
 - Charged per resource or time unit
 - No Quality of Service (QoS) guarantees
 - Cloud might not meet the demands of the customer
 - Cloud resources shared among customers
 - Virtual Machines
 - Contention can result in performance issues

Motivation and Background

- Example: Cache contention
- Running alone: level until saturates LLC
- With co-runners: fast and significant time increases

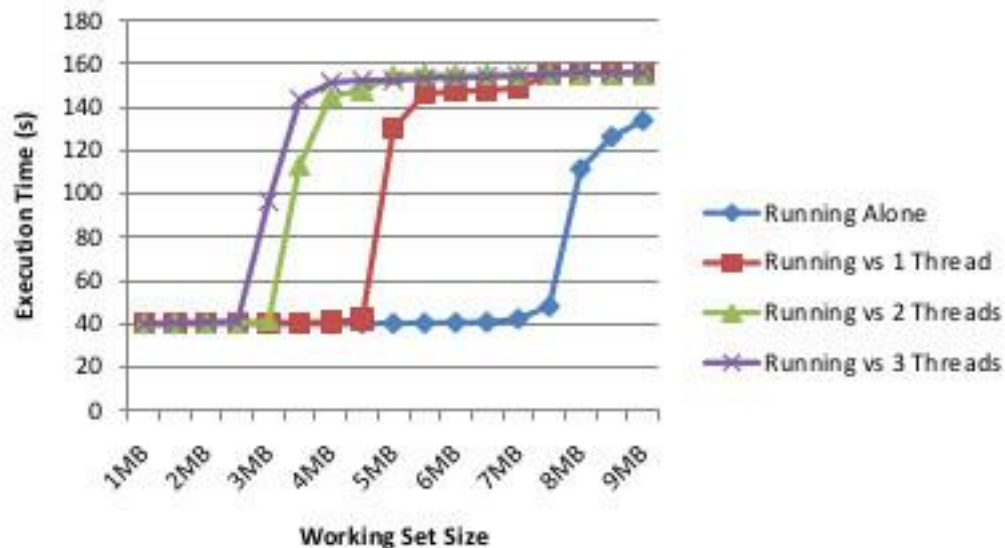


Figure 1. Performance impact of cache interference on consolidated VMs.

Motivation and Background

- Solution: tune performance to the level the customer would see if they were running alone on the system
- Q-Clouds: “A QoS-aware control framework”
 - Allocates resources in a fair way between customers, resulting in an acceptable QoS level

Q-Clouds System

- Change resource allocation to meet the various customers' Service Level Agreements (SLAs)
 - Applications perform the same as if the customer were alone on the hardware
- MIMO (multiple-input multiple-output) closed-loop feedback model
 - Feedback from applications
 - “Interference relationships”
 - “Q-states” specify QoS level of applications

Q-Clouds System

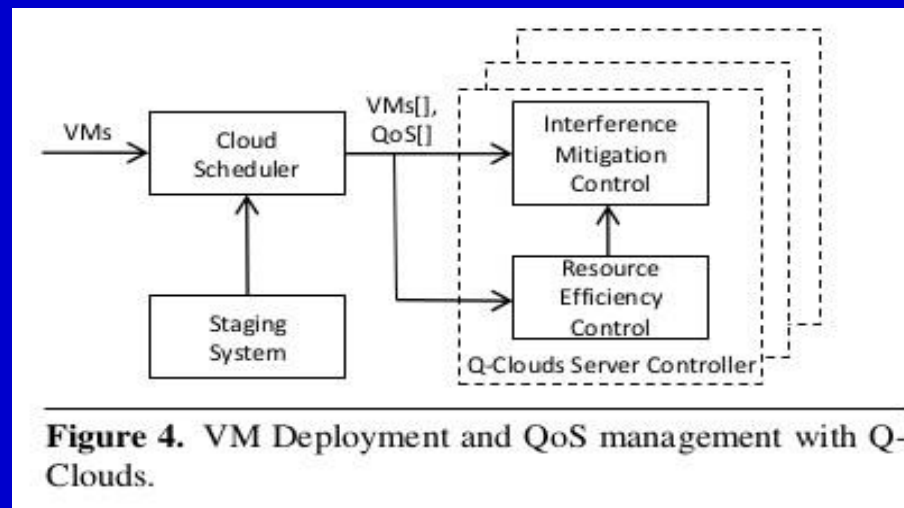
- Interference on multi-core processors: QoS not tied to resources available
- Best way to implement QoS: Guarantee app. performance, charge for app. Performance
 - Charge as if the app. were running without contention
 - When interference occurs, adjust resource allocations to maintain QoS level
 - How to implement this?

Q-Clouds System

- Q-Clouds: QoS in the face of interference
 - Head room: unallocated resources given to an app. to prevent falling below QoS performance
 - Q-States: higher level of QoS to apps. that are willing to pay for it, when unused head room exists

Q-Clouds

- Q-Clouds Management Architecture
 - Cloud Scheduler: Place VMs on servers according to resource requirements



Q-Clouds System

- Q-Clouds Management Architecture
 - First watch the VM on a *Staging Server* to see how it would run without contention, then Cloud Scheduler can place on appropriate server
 - The resource needs observed on the Staging Server also determine \$\$\$
 - Interference Mitigation Control
 - Subsystem on each server
 - Change resource allocations to keep VMs running at the same level as they were on the Staging Server

Q-Clouds System

- Q-Clouds Management Architecture
 - Resource Efficiency Control
 - Increase QoS for VMs with Q-State levels when there is extra (unused) headroom
 - Tune Interference Mitigation Control to comply with the QoS changes determined (new Q-State for a VM)
- How to map resource allocation to QoS?
 - MIMO, feedback loops.

Q-Clouds System

- Q-Clouds MIMO
 - Input: control of resource allocations
 - This is the system itself, so already available
 - Output: VM performance (QoS values)
 - Requires feedback from applications
 - But each application might have its own QoS metric
 - They expect the applications to provide QoS data
 - QoS data used in staging area and during run-time adjustments of resources such, as assignment of Q-States
 - MIMO analyzes performance WRT process interactions

Q-Clouds System

- Q-states allow processes to run at a higher QoS (performance) level if: a) the customer paid for it, and b) there are extra resources available (in the headroom)
- Only bump up QoS past base SLA level if every task is running \geq its acceptable minimum, otherwise use some of that extra headroom to help a struggling task

Experiment

- Considered three interference effects:
 - Memory bus contention
 - Last level cache (LLC) contention
 - Prefetching (instructions and data)
- Control interference by capping VM Virtual Processors (VPs)

Experiment

- Since controlling interference by limiting VP function, want to test with CPU-bound benchmarks
 - SPEC CPU2006 benchmark suite
- Four applications on one quad-core processor
- Selected 5 benchmarks from the set, tested every combination of 4

Experiment

Table 1. SPEC CPU2006 workload mixes.

Benchmark	Workload Mix				
	1	2	3	4	5
436.cactusADM	X	X	X	X	
437.leslie3d	X	X	X		X
459.GemsFDTD	X	X		X	X
470.lbm	X		X	X	X
471.omnetpp		X	X	X	X

Experiment

- Dual socket server
- Ea. socket quad-core Nehalem processor
- 18 GB RAM
- Total: 36 GB RAM, eight cores
- Virtualization system: Windows Server 2008 with Hyper-V

Experiment

- Q-Clouds runs in the hypervisor ('root partition')
 - Watches CPU related performance counters of the VMs
 - Adjusts VP resource allocations
- MATLAB code for the System Controller functional block of Q-Clouds
 - Queries hypervisor for QoS information, adjusts VP caps in response

Evaluation

- App. From Figure 1 shown here. Note that capping CPU resources linearly increases execution time
- Running four at a time causes performance to degrade faster
- WSS relevant

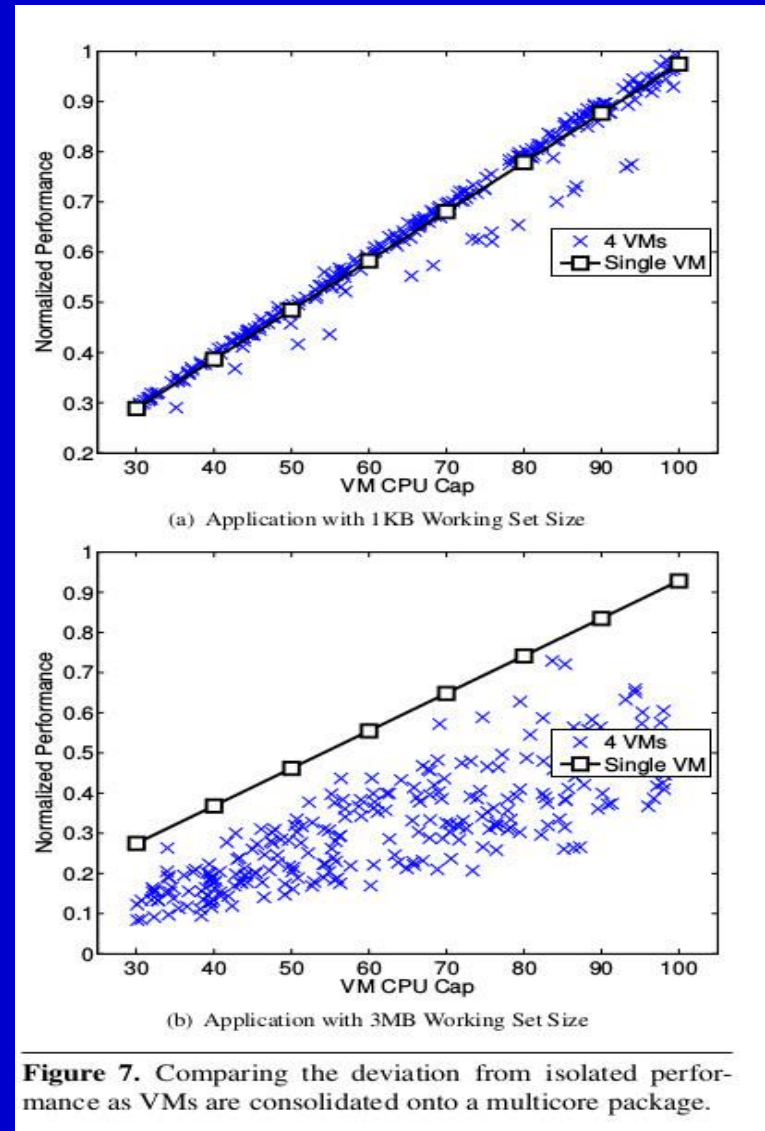


Figure 7. Comparing the deviation from isolated performance as VMs are consolidated onto a multicore package.

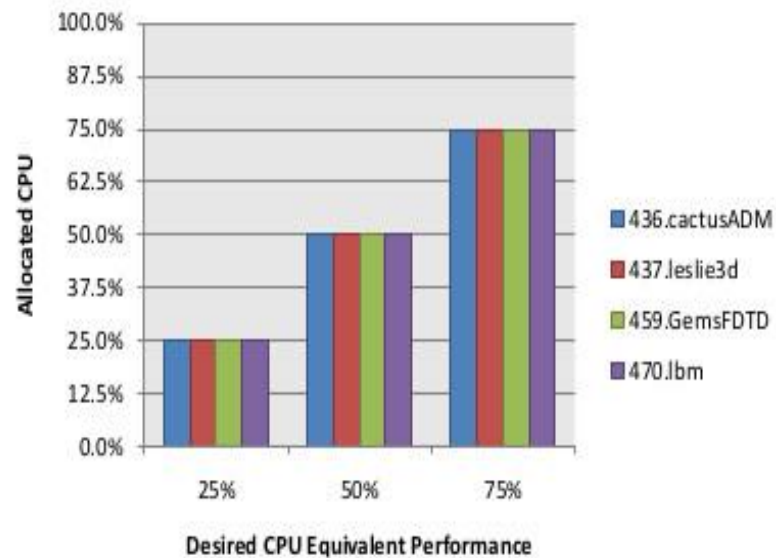
Evaluation

- What's that tell us? That we can model interference and make a MIMO model from application performance feedback
- Various MIMO models available with different benefits and drawbacks

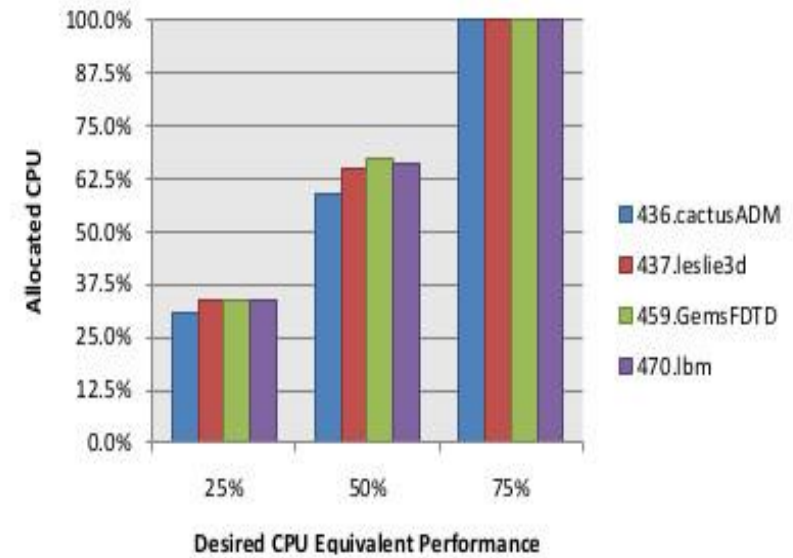
Evaluation

- Back to the point of all this: “Meeting QoS Requirements with Q-Clouds”
- Must meet the non-contention QoS specified in the SLA. In the example, the test process set specifies QoS by processor resources available to it
- Compare performance to the case where the system does not allocate resources for QoS, to test the system
- 3 test SLA levels: require 25%, 50%, 75% of CPU

Evaluation

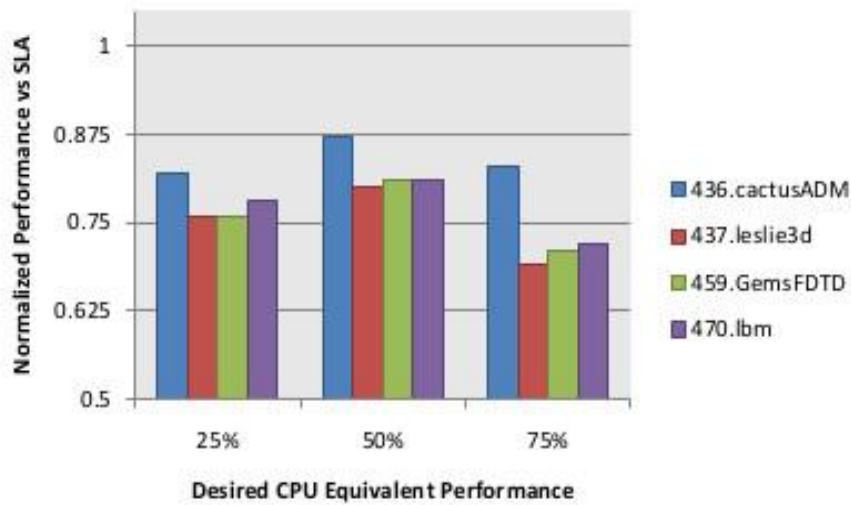


(b) Default Resource Allocation

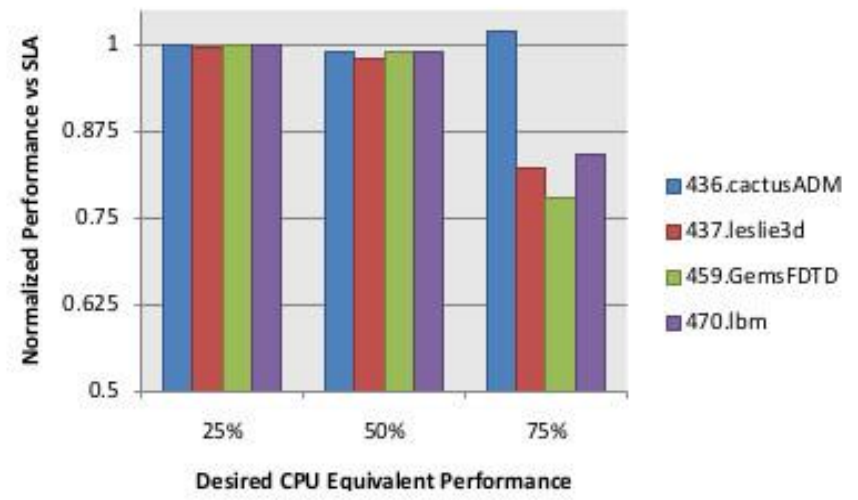


(b) Q-Clouds Resource Allocation

Evaluation



(a) Default Performance



(a) Q-Clouds Performance

Evaluation

- Without Q-Clouds, contention is significant and nobody gets their desired QoS
- With Q-Clouds, the 25% and 50% CPU allocation instances are great. But at the 75% level, the system runs out of resources (headroom) and contention results in degradation

Evaluation

- Other test sets (mixes of the benchmark programs) show similar results. Q-Clouds improves performance as long as there is headroom available
- When the test is extended to include Q-States functionality, it is found that the system is able to implement it successfully, again if there is sufficient headroom

Evaluation

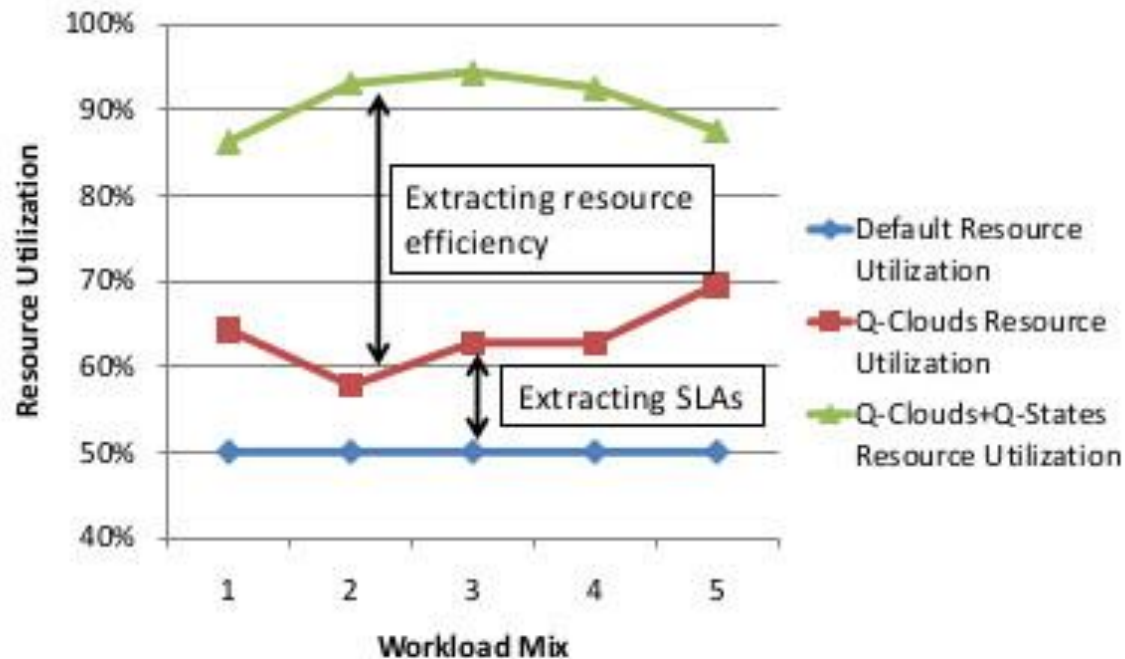


Figure 13. Resource utilization comparisons between default systems, Q-Clouds servers, and Q-Clouds with Q-state enabled resource efficiency optimization.

Conclusion

- With the cloud comes the need for cloud-aware scheduling to address performance limiting factors unique to this environment
- Q-Clouds can theoretically ensure processes a particular QoS level, if the processes know the QoS metric(s) that is(are) important to them

Questions

- The Q-Clouds system relies on QoS feedback provided by the application. Is there a way around this, so that any application could be handled by Q-Clouds?

References

R. Nathuji, A. Kansal, and A. Ghaffarkhah. Q-Clouds: Managing performance interference effects for QoS-aware clouds. Microsoft Research.