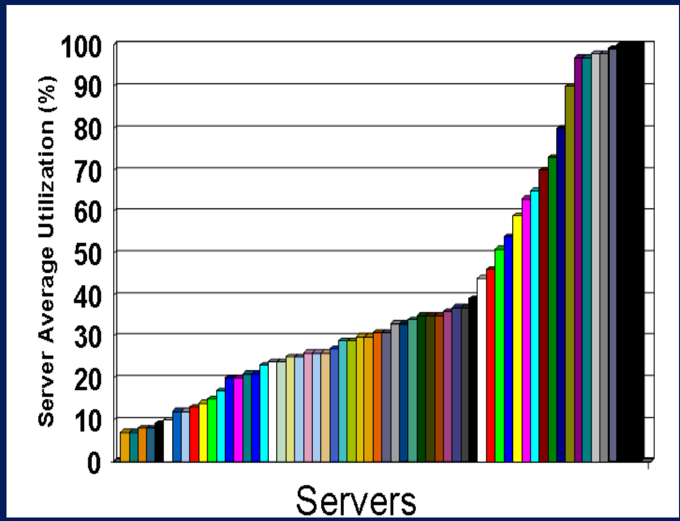
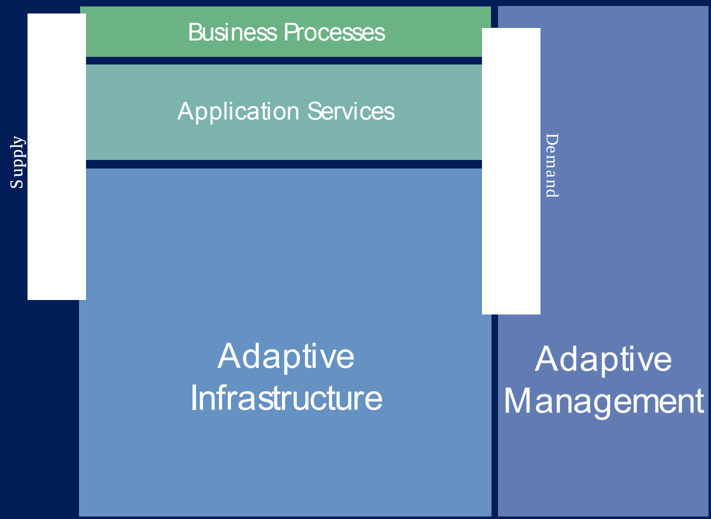
An Automated Approach for Supporting Application QoS in Shared Resource Pools

Jerry Rolia, Lucy Cherkasova, Martin Arlitt, and
Vijay Machiraju
HPLabs

Introduction

- Adaptive enterprise and automation – are key research topics on HP agenda.
- Resource pools are computing environments that offer virtualized access to shared resources:
  - Clusters of servers
  - Racks of blades
  - SMPs
- When these environments used effectively, they can:
  - align the use of capacity with business needs (*flexibility*),
  - lower infrastructure costs (via *resource sharing*),
  - lower operating costs (via automation).
- We designed the Quartermaster capacity manager service for automatically managing such pools: it implements a trace-based approach and provides answers to classic problems:
  - how much capacity is needed?
  - which workloads should be assigned to each resource?
  - what is the performance impact of workload scheduler and/or policy settings?





Problem statement

- Develop method that provides each hosted *application with the QoS* it requires while making effective use of the resource pool.

- *General approach*
  - Define range of acceptable application QoS
  - Relate application QoS to target utilization of resource allocation
  - Partition allocations across two Classes of Services (CoS 1 and Cos2) to manage resource access QoS:
    - CoS 1: *guaranteed access* to resources
    - CoS 2: *access with some pre-specified probability*
  - Model resource access using capacity manager
    - Assign applications to resources such that the application QoS requirements are met.

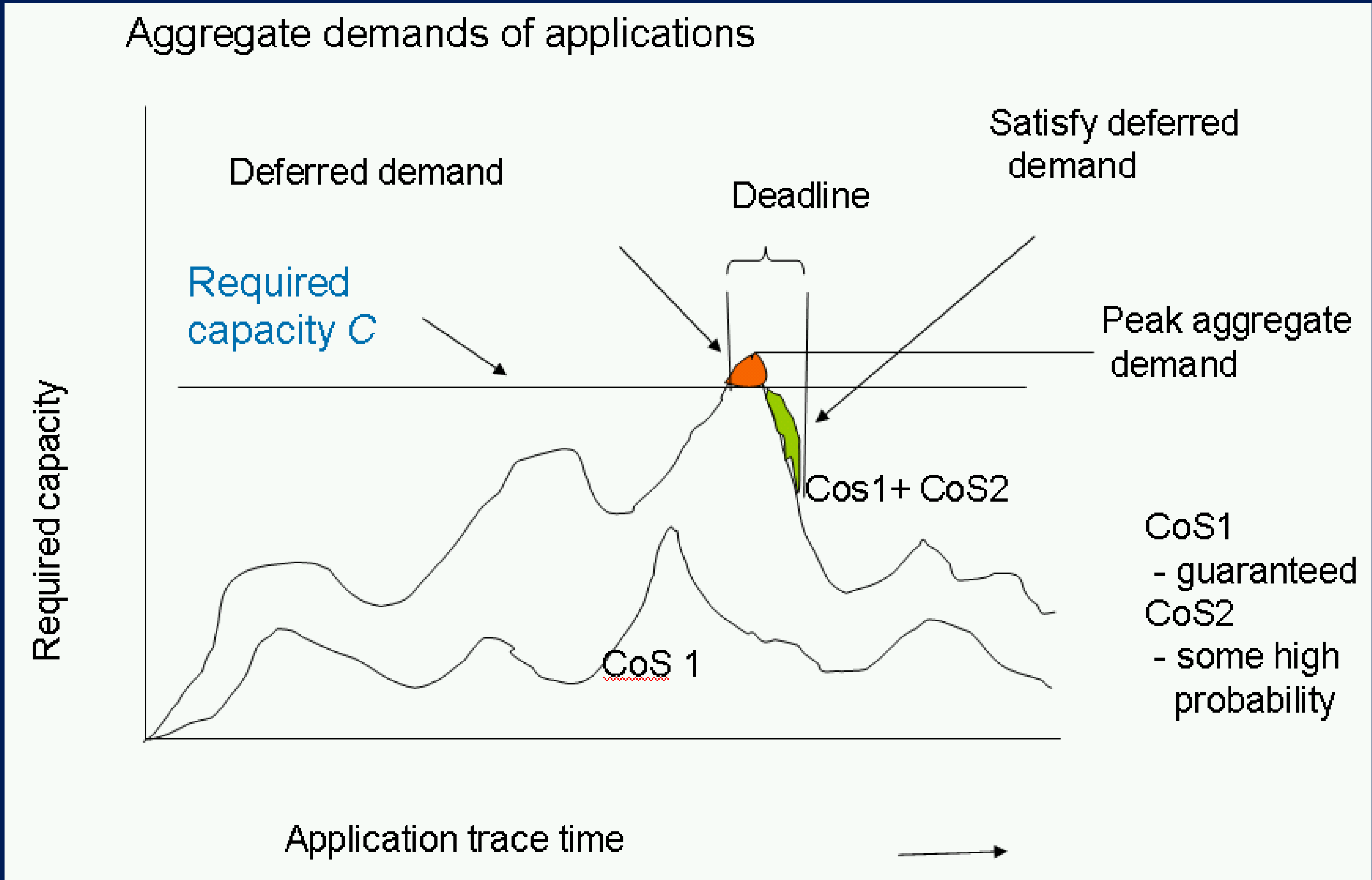
Approach

- There are existing tools (like HP Workload Manager or IBM Enterprise Workload Manager) that aim to support the resource management tasks in such resource pools.
- *However,* the process of setting and adjusting parameters in these tools for managing required capacities is still a manual process. For example:
  - *each resource in the pool has a scheduler* that monitors workloads' demands and dynamically varies the allocation of capacity,
  - *the scheduler can implement, at least, two priorities*, where
    - all demands associated with *highest priority* satisfied first,
    - the remaining capacity is used to satisfy the demands of the *next priority*.
- Question: how to split workload demands between these two priority classes to satisfy range of acceptable application QoS.

Trace-Based Approach and Resource Access QoS

- Each application workload is characterized using several weeks to several months of demand observations (e.g. one observation every 5 min).
- We designed and implemented the Quartermaster capacity management tool that has an optimized search method that supports consolidation (e.g. tight packing) and load leveling (e.g. load balancing) exercises.
- This tool supports capacity planning across two different Classes of Services (*CoS 1* and *CoS 2*) in such a way that
  - demands associated with CoS 1 are *guaranteed* and
  - demands associated with CoS 2 are *offered with* an operator specified resource *access probability* θ .
- This way, it allows the controllable overbooking of the available capacity for exploiting the benefits of statistical multiplexing of application demands while providing the desirable application QoS.

Resource Access QoS



Application QoS

- The relationship between acceptable application QoS and system resource usage is complex.
- For interactive applications (like web servers), the responsiveness is important.
- To optimize the application response time it can be desirable *to support the utilization of assigned resource allocation at a given level* (e.g. 50%).
- This goal can be achieved by controlling the relationship between demand and allocation using a *burst factor n*, i.e. workload resource allocation is *n* times its recent demand.
- Another angle: allocations are adjusted using periodic utilization measurements: *mean values hide the bursts of demand* within the interval. Typically, workloads with higher variation of demands require a higher allocation scaling factor (i.e higher burst factor).

Application QoS (cont.)

- We employ empirical approach that aims to find an acceptable range for the burst factor and application QoS objectives.
- Stress testing exercise is used to estimate the application performance under different burst factor.
- We search for a value of burst factor n_{ideal} >=1 that supports desirable application responsiveness, as well as n_{ok} : $1 \le n_{ok} \le n_{ideal}$ that offers adequate responsiveness (not as good as “ideal” but still acceptable).
- These values for a burst factor bound lower and upper values for the *utilization of an allocation*:

$$U_{low} = 1/\, n_{ideal} \qquad\qquad U_{high} = 1/\, n_{ok}$$

Thus the utilization of allocation must remain in the range

(U_{low}, U_{high})

where U_{high} is acceptable but not ideal.

Application QoS (cont.)

- We aim to partition an application’s workload demands across two classes of service, CoS 1 and CoS 2, to ensure that the application’s burst factor remains within an acceptable range, i.e. that the utilization of allocation is kept within the desirable range (U_{low} , U_{high}).
- Let *p* be a fraction of *peak demand D* for the CPU attribute of application workload that is associated with CoS 1.
- Value *p x D* gives a *breakpoint* for the application workload that is associated with CoS 1:
  - all demand that less or equal to *p x D* is placed in CoS 1, and
  - the remaining demand is places in CoS 2.
- When system has enough capacity, each application gets access to all capacity it needs, and utilization of allocation is U_{ideal} .
- When demands exceed supply, the demands associated with CoS 1 are guaranteed to be satisfied. However, demands associated with CoS 2 are not guaranteed and offered with access probability θ . This could lead to a smaller allocation (than ideal) and to higher utilization of allocation U_{high}.

Computation of Breakpoint

- To ensure that the allocation utilization remains within the desirable range (U_{low} ,U_{high}), the range of allocations must be between $A_{ideal} = D \times n_{ideal}$ and $A_{ok} = D \times n_{ok}$.
- So the allocation for the lower but acceptable QoS offered to the applications is:

$$A_{ok} = A_{ideal} \times p + A_{ideal} \times (1 - p) \times \theta.$$

Solving this equation for p, we get:

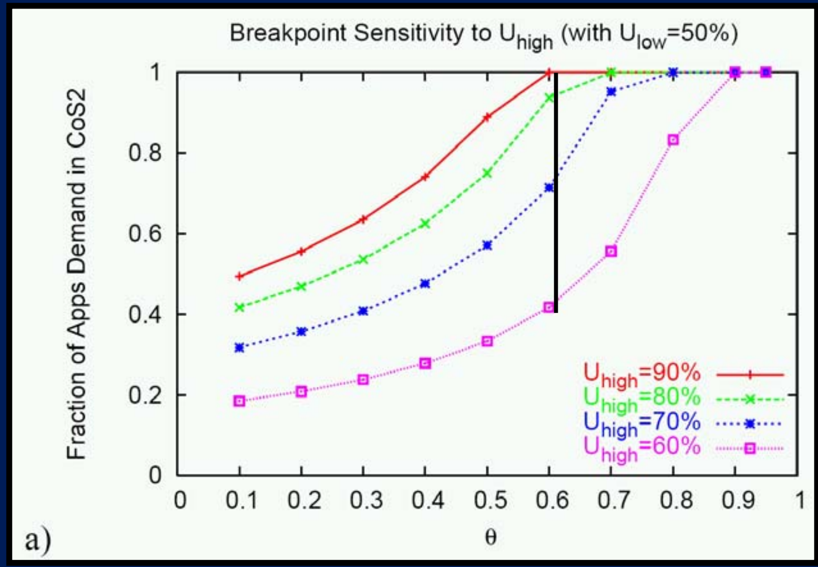
$$p = \frac{\frac{n_{ok}}{n_{ideal}} - \theta}{1 - \theta},$$

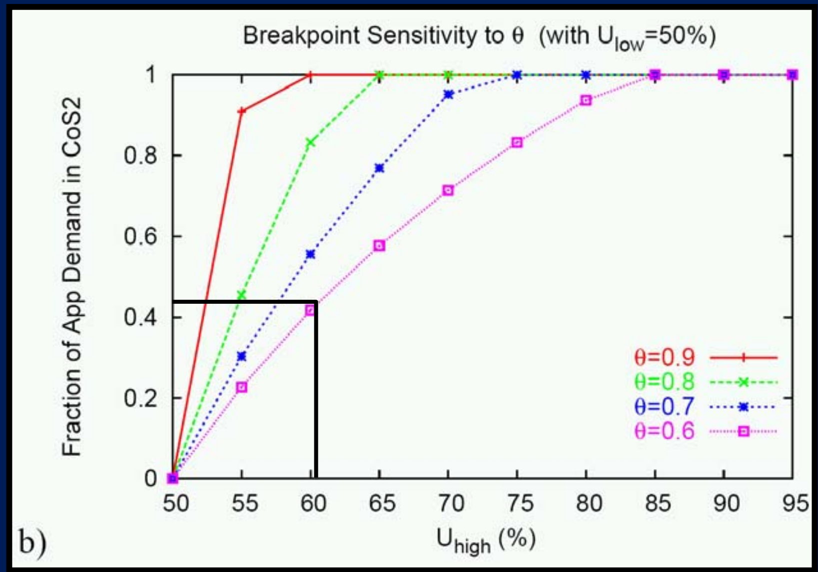
where $1 \geq \theta > 0$.

- This breakpoint *p* is a scheduler parameter that we automatically compute.

Case Study

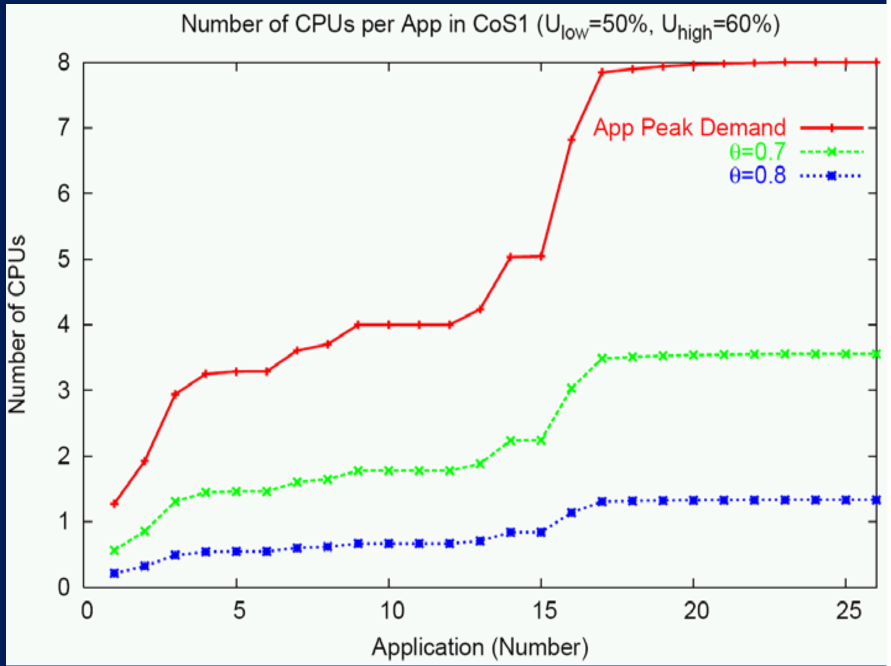
- The top Figure shows general relationship between resource access probability for COS 2, the allocation utilization, and the fraction of application's peak demand associated with CoS 2.
  - Even for a low access probability of 0.6, 40% to100% of workload demand can be put in CoS 2.
- The bottom Figure shows similar results: Quartermaster capacity manager can automatically map the desirable allocation utilization and selected value θ to the recommended breakpoint value.





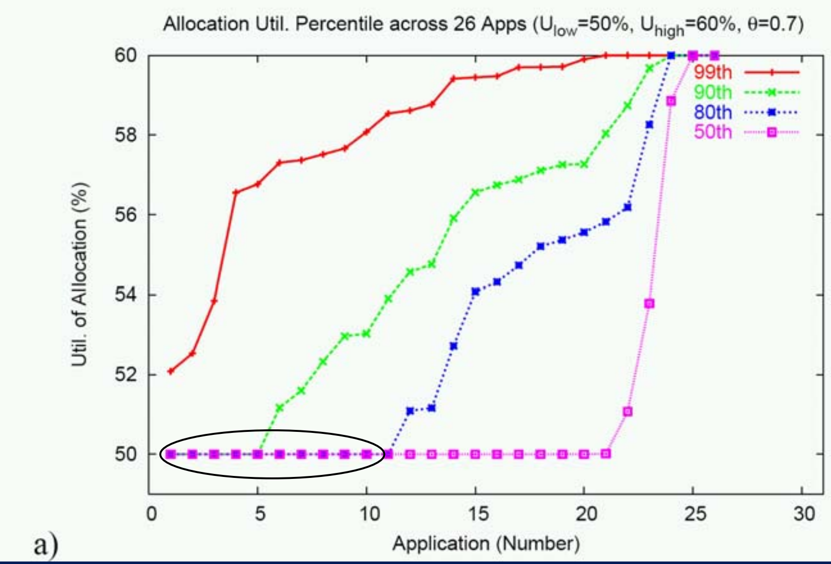
Case Study

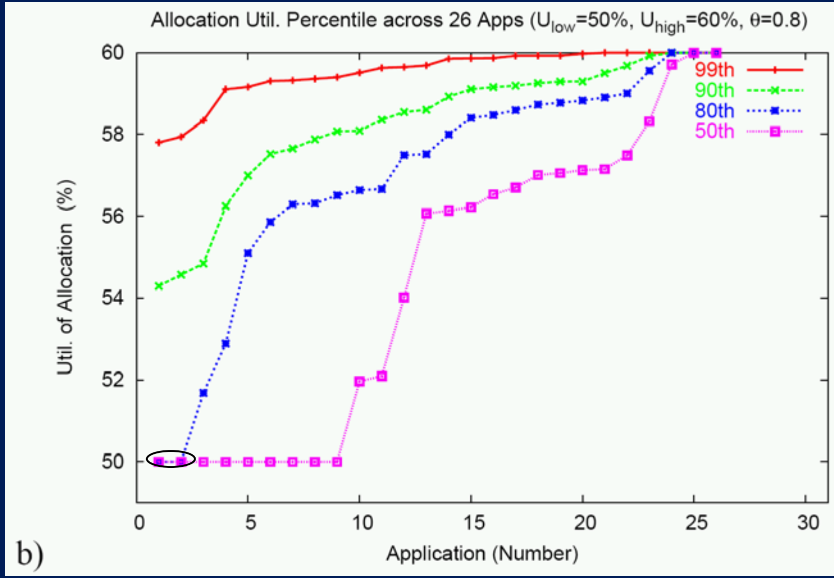
- The Figure illustrates the proposed approach using the 26 applications of a large order entry system.
- In this scenario, the application utilization of allocation is in the range (0.5, 0.6).
- The Figure shows
  - the peak number of CPUs needed by each application,
  - how many CPUs must be provisioned using the guaranteed CoS 1 under different θ (0.7 and 0.8) for CoS 2.
- As expected, higher value for θ increases the use of the shared portion of resource pool, that supports higher resource overbooking, and leads to the higher resource utilization in the pool.



Case Study (cont.)

- These Figures provide insight into how the selected breakpoint affects each of 26 applications
- The breakpoint is chosen based on the application *peak* demands.
- The top Figure shows that 11 of 26 applications spend 80% of their time at 50% utilization of allocation, i.e. at the low end of its utilization allocation U_{low}.
- The bottom Figure shows that only 2 of 26 applications spend 80% of their time at 50% utilization of their allocation.
- Increasing the access probability for CoS 2, allows us to put more of the application demand in CoS 2, while at the same time putting the application at a greater risk of operating closer to the higher end of its utilization allocation U_{high}.





Conclusion and Future Work

- We presented a method for selecting application workload specific scheduling parameters for shared resource pool environments.
- The approach lets the application owner specify application QoS requirements using a range of a desirable allocation utilization for the CPU demand attribute.
- This range along with the resource access QoS determine how much of the applications' demands must be associated with a guaranteed Class of Service CoS 1 and how much can be put in a second class CoS 2 with a given resource access probability.
- Future work includes developing a better understanding of scheduler behavior on allocations, its impact on capacity management at long timescales, and application QoS.