Run-time Resource Management in SOA Virtualized Environments

Danilo Ardagna, Raffaela Mirandola, Marco Trubian, Li Zhang

Amsterdam, August 25 2009
SOI Run-time Management

- SOI = SOA + virtualization

- Goal: flexible solution for accessing component based service applications on demand

- High workload fluctuations: need for run-time resource provisioning

- Our work: determine the optimum capacity allocation for multiple Virtual Machines
Outline

• Problem statement

• Reference system

• Optimal capacity allocation problem

• Experimental results

• On going work and conclusions
Problem statement

- SOI workload can vary by orders of magnitude within the same business day
- Such variations cannot be accommodated by separating design and run-time point of view:
  - run-time resource provisioning and exploit lower level (OS) mechanisms
  - dynamically allocate resources on the basis of short-term demand estimates
  - meet end user’s QoS requirements while adapting the SOA environment
  - the infrastructure configuration is updated periodically according to a workload prediction
Virtualization of resources

- Hardware resources (CPU, RAM, ecc...) are partitioned and shared among multiple **virtual machines** (VMs)

- The virtual machine monitor (VMM) governs the access to the physical resources among running VMs

- Performance isolation and security
Virtualization of resources

- Service and platforms heterogeneity
- Need to predict efficiently the performance of such systems at runtime
- Optimal capacity allocation with strict time constraints without adding a significant overhead
Virtualized Service Center

- **L1 resource manager:**
  - the set of physical servers to use
  - VMs to physical servers allocation
  - load balancing across multiple VMs
  - perform admission control

- **L2 resource manager:**
  - capacity allocation
  - frequency scaling, ...
Virtualized Service Center

- App
- App
- App
- OS
- OS
- OS
- VM
- VM
- VM
- Virtual Machine Monitor
- Monitor & Workload predictor
- Resource Allocator
- \( \mu_k \)
- \( \lambda_k \)
- \( \phi_k \)
- SMP - Multicore system
Revenues are a function of average response times

Per request revenue

\[ m_k = -u_k / \bar{R}_k \]

Average response time soft-constraint

Average response time

\[ E[R_k] \]
• VMM modelling: GPS (Generalized Processor Sharing) scheduling

• The VMM allocates the CPU core to competing VMs proportionally to the scheduling normalized weights $\phi_k$, $\sum_{k \in K} \phi_k = 1$ that each VM has been assigned
• Under GPS, the server capacity devoted to class $k$ VM at time $t$ (if any) is:

$$\frac{\phi_k}{\sum_{k' \in \mathcal{C}(t)} \phi_{k'}}$$

• Requests within each class are executed either in a First-Come First-Serve (FCFS) or a Processor Sharing (PS) manner

• Under FCFS: service time has an exponential distribution with mean $\mu_k^{-1}$

• Under PS: service time follows a general distribution with mean $\mu_k^{-1}$
GPS Bounding technique

- Approximation: each multi-class single-server queue is decomposed into multiple independent single-class single-server queues with capacity greater than or equal to $\phi_k$. 
GPS Bounding technique

- Approximation: each multi-class single-server queue is decomposed into multiple independent single-class single-server queues with capacity greater than or equal to $\phi_k$

- Under these hypotheses, an upper bound of VMs average response time can be evaluated as:

$$E[R_k] = \frac{1}{\mu_k \phi_k - \lambda_k}$$
Capacity Allocation Optimization Problem

- The Service Provider objective is to maximize the revenues from SLAs which are given by:

\[
\sum_{k \in K} U_k \left(E[R_k]\right) \lambda_k = \sum_{k \in K} \left( m_k \frac{\lambda_k}{\mu_k \phi_k - \lambda_k} \right) + \sum_{k \in K} u_k \lambda_k
\]

- The decision variables of our model are \( \phi_k \) which determine the capacity devoted for executing class \( k \) VM on a specific core.
Capacity Allocation Optimization Problem

P1) \[
\max \sum_{k \in K} m_k \frac{\lambda_k}{\mu_k \phi_k - \lambda_k}
\]

\[
\sum_{k \in K} \phi_k \leq 1 \quad (1)
\]
\[
\lambda_k < \mu_k \phi_k \quad (2)
\]
\[
\phi_k \geq 0 \quad (3)
\]

• The objective function is concave:

\[
H = Diag \left( \frac{2m_k \mu_k \lambda_k}{(\mu_k \phi_k - \lambda_k)^3} \right)
\]
Global Optimum Solution

- Efficient ad-hoc solution with complexity $O(|K|)$ through the Karush Kuhn-Tucker conditions which is based on the following Theorem
Global Optimum Solution

- Efficient ad-hoc solution with complexity $O(|K|)$ through the Karush KuhnTucker conditions which is based on the following Theorem

**Theorem 1.** Let $k$ be an arbitrary VM class. The optimum solution of problem P1) is given by:

$$
\phi_k = \frac{1}{\mu_k} \left( \sqrt{\frac{m_k \mu_k \lambda_k}{m_k \mu_k \lambda_k}} \left( \frac{\mu_k \phi_k}{\mu_k} - \frac{\lambda_k}{\lambda_k} \right) + \lambda_k \right), \quad \forall k \neq \bar{k}
$$

where:

$$
\phi_{\bar{k}} = \frac{\lambda_{\bar{k}}}{\mu_{\bar{k}}} + \frac{1 - \sum_{k \in K} \frac{\lambda_k}{\mu_k}}{\sum_{k \in K} \sqrt{\frac{m_k \mu_k \lambda_k}{m_k \mu_k \lambda_k}}}
$$
Algorithm Performance

- All tests have been performed on a Intel Core Duo E6850 3 GHz workstation

- The number of request classes $|K|$ has been varied between 10 and 100

- Service demands and $|m_k|$ were uniformly generated in the interval [0.1, 1] second and [0.2, 2], respectively

- Revenues obtained by using a single CPU for one hour varies between 1$ and 10$ according to the current commercial fees (e.g., Sun Utility Computing, Amazon Elastic Cloud)

- $R_k$ was proportional to the demanding time of class $k$ requests, we set $R_k = 10/\mu_k$
| $|K|$ | Optimization Time |
|-----|-------------------|
| 10  | 24                |
| 20  | 30                |
| 40  | 60                |
| 80  | 181               |
| 100 | 220               |

Capacity Allocation Solution Execution Time (s)
Validation in a Prototype Environment

- Goals:
  - evaluate the quality of the GPS approximation
  - evaluate the effectiveness of our capacity allocation algorithm in a real testbed

- The physical machine hosting the VMs is based on two Intel Xeon Woodcrest 3GHz dual core (four physical core overall) with 2 GB RAM

- The VMM is VMware ESX Server 3.0.1, while VMs run Linux Fedora Core 6.0

- VMWare parameters for resource settings:
  - limit, i.e., a cap on the consumption of CPU time, measured in MHz
  - reservation, i.e., a certain number of CPU cycles reserved for the execution of the VM, measured in MHz;
  - share, i.e., a priority for the execution of the VM expressed by a number between 1 and 10,000

- In our experimental setup we did not set any limit, the reservation was set equal to 0 MHz for each VM and the shares were set proportionally to VMs' $\phi_k$ values

- VMs were constrained to run on a single physical core with 256 MB of RAM reservation
Validation in a Prototype Environment

- The experimental framework is based on a workload generator and a micro-benchmarking Web service:
  - custom extension of the Apache JMeter 2.3.1 workload injector
  - Web service application is hosted within the Apache Tomcat 5.5 application server, designed to consume a fixed CPU time

- The service demand is generated according to a log-normal distribution where $C=4$ and the incoming workload varied in the range 0.16 and 10 req/sec
The error reduces as the utilization of the physical machine increases

When the utilization of the physical machine is about 90-95%, the average percentage error is around 30%

With the current practice of server consolidation, the aim is to increase the CPUs utilization and 80% has been reached also on x86 server farm
Validation in a Prototype Environment

- Two single tier applications supporting a gold and a bronze request class hosted on a single core are considered ($|m_{\text{gold}}| = 10 \cdot |m_{\text{bronze}}|$)
- The gold class workload is increased while the bronze class workload is kept constant
Conclusion and ongoing work

- Resource allocation policy which dynamically allocates resources among competing virtual machines
- The capacity allocation problem has been modeled as a non-linear problem which has been optimally solved
- Introduce more accurate but still fast approximated solution techniques for performance evaluation of virtualized environments
- Experimental evaluation of operating system based VMM like OpenVZ or Virtuozzo
- Standard benchmarks and real applications