On the Use of Performance Models in Autonomic Computing

Daniel A. Menascé
Department of Computer Science
George Mason University
Motivation for AC

• “…main obstacle to further progress in IT is a looming software complexity crisis.” (from an IBM manifesto, Oct. 2001).
  – Tens of millions of lines of code
  – Skilled IT professionals required to install, configure, tune, and maintain.
  – Need to integrate many heterogeneous systems
  – Limit of human capacity being achieved
Motivation for AC (cont’d)

• Harder to anticipate interactions between components at design time:
  – Need to defer decisions to run time

• Computer systems are becoming too massive, complex, to be managed even by the most skilled IT professionals

• The workload and environment conditions tend to change very rapidly with time
Multi-scale time workload variation of a Web Server

3600 sec

60 sec

1 sec
Self-Optimizing Systems

- Complex middleware and database systems have a very large number of configurable parameters.

<table>
<thead>
<tr>
<th>Web Server (IIS 5.0)</th>
<th>Application Server (Tomcat 4.1)</th>
<th>Database Server (SQL Server 7.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP KeepAlive</td>
<td>acceptCount</td>
<td>Cursor Threshold</td>
</tr>
<tr>
<td>Application Protection Level</td>
<td>minProcessors</td>
<td>Fill Factor</td>
</tr>
<tr>
<td>Connection Timeout</td>
<td>maxProcessors</td>
<td>Locks</td>
</tr>
<tr>
<td>Number of Connections</td>
<td></td>
<td>Max Worker Threads</td>
</tr>
<tr>
<td>Logging Location</td>
<td></td>
<td>Min Memory Per Query</td>
</tr>
<tr>
<td>Resource Indexing</td>
<td></td>
<td>Network Packet Size</td>
</tr>
<tr>
<td>Performance Tuning Level</td>
<td></td>
<td>Priority Boost</td>
</tr>
<tr>
<td>Application Optimization</td>
<td></td>
<td>Recovery Interval</td>
</tr>
<tr>
<td>MemCacheSize</td>
<td></td>
<td>Set Working Set Size</td>
</tr>
<tr>
<td>MaxCachedFileSize</td>
<td></td>
<td>Max Server Memory</td>
</tr>
<tr>
<td>ListenBacklog</td>
<td></td>
<td>Min Server Memory</td>
</tr>
<tr>
<td>MaxPoolThreads</td>
<td></td>
<td>User Connections</td>
</tr>
<tr>
<td>worker.ajp13.cachesize</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Autonomic Computing

• System that can manage themselves given high-level objectives.
  – High-level objectives can be expressed in term of service-level objectives or utility functions.

• Autonomic computing is inspired in the human autonomic nervous system:
  – “The autonomic nervous system consists of sensory neurons and motor neurons that run between the central nervous system and various internal organs such as the: heart, lungs, viscera, glands. It is responsible for monitoring conditions in the internal environment and bringing about appropriate changes in them. The contraction of both smooth muscle and cardiac muscle is controlled by motor neurons of the autonomic system.” http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/P/PNS.html
  – The autonomic nervous system functions in an involuntary, reflexive manner.
Autonomic Systems

• Self-managing
  – Self-configuring
  – Self-optimizing
  – Self-healing
  – Self-protecting

• Self-* systems
Autonomic Systems

• Self-managing
  – Self-configuring
  – Self-optimizing
  – Self-healing
  – Self-protecting

• Self-* systems
Self-optimizing systems: motivation

Q: How does the response time vary with the number of software threads, $M$?
*Obs:* For each workload level there is an optimal number of threads.
Workload and QoS metrics

Workload:
• transactions
• HTTP requests
• video downloads
• calls to Call Center

QoS metrics:
• response time
• throughput
• availability
• page download time
• revenue throughput
• abandonment rate
Self-optimizing System

Workload:
- transactions
- HTTP requests
- video downloads
- calls to Call Center

QoS metrics:
- response time
- throughput
- availability
- page download time
- revenue throughput
- abandonment rate

Controller

Service Level Objectives
Self-optimizing System

Computer system

Monitor

Controller

Monitor

Workload:
- transactions
- HTTP requests
- video downloads
- calls to Call Center

QoS metrics:
- response time
- throughput
- availability
- page download time
- revenue throughput
- abandonment rate

Service Level Objectives

© 2012 D.A. Menasce. All Rights Reserved.
Self-optimizing System

Computer system

Workload:
- transactions
- HTTP requests
- video downloads
- calls to Call Center

QoS metrics:
- response time
- throughput
- availability
- page download time
- revenue throughput
- abandonment rate

Controller

Analyze & Plan

Service Level Objectives

© 2012 D.A. Menasce. All Rights Reserved.
Self-optimizing System

Computer system

Workload:
• transactions
• HTTP requests
• video downloads
• calls to Call Center

QoS metrics:
• response time
• throughput
• availability
• page download time
• revenue throughput
• abandonment rate

Controller

Service Level Objectives
IBM’s MAPE-K Model for AC
Performance Model-Based Autonomic Computing

state

Value
Performance Model-Based Autonomic Computing

State: e.g., set of configuration parameters

Value: e.g., QoS metric, utility function value

Goal: find state that optimizes the value subject to cost constraints

- State space is large
- Objective function does not have a closed form

Use performance models to compute value at each state.

Use combinatorial search techniques to find near-optimal solution.
Examples of Autonomic Computing Systems

- E-commerce system
- Internet Data Center
- Virtualized Environment
- SOA Software Systems
Examples of Autonomic Computing Systems

• E-commerce system
• Internet Data Center
• Virtualized Environment
• SOA Software Systems

Multi-tier Architecture
Queuing Model Used to Compute QoS Values
Experimental Results

Arrival rate

![Graph showing the arrival rate over time (Controller Intervals). The x-axis represents time in Controller Intervals ranging from 0 to 35, while the y-axis represents the arrival rate in requests per second ranging from 0 to 100. The graph shows an upward trend with a peak around the 15th Controller Interval, followed by a decline.]
Results of QoS Controller

![Graph showing the results of QoS controllers with arrival rate on the x-axis and QoS on the y-axis. The graph compares controlled QoS and uncontrolled QoS.]
Experiment Results

Arrival rate

QoS is not met!
Examples of Autonomic Computing Systems

- E-commerce system
- Internet Data Center
- Virtualized Environment
- SOA Software Systems

Motivation for Dynamic Resource Allocation in Data Centers

• Large data centers host several application environments (AEs)
• AEs run customers’ applications (either online transactions or batch processes) on assigned servers (resources)
• AEs are subject to workloads that exhibit wide and unpredictable variations in their intensities
• Resources need to be redeployed dynamically among AEs to optimize some global utility function
Dynamic Resource Allocation Problem

Application Environment 1

Application Environment 2

. . .

Application Environment M

Server 1

Server 2

. . .

Server N

© 2012 D.A. Menasce. All Rights Reserved.
Dynamic Resource Allocation Problem

Application Environment 1

Application Environment 2

Application Environment M

Server 1

Server 2

Server N
Dynamic Resource Allocation Problem

• The data center has the following characteristics:
  – M Application Environments (batch and online)
  – A fixed number, N, of physical resources (servers)
  – All servers have the same installed software
  – A server can be assigned to any AE
Dynamic Resource Allocation
Two-level Controllers

Global Controller

Decides how many servers to assign to each AE.

Local Controller

Implements Global Controller’s decisions.

Server

Application Environment 1

Application Environment M
Dynamic Resource Allocation Problem – Utility Functions

• The global controller uses a global utility function, $U_g$, to assess the adherence of the overall data center performance to desired service levels (SLAs)

\[
U_g = h(U_1, \ldots, U_M)
\]

\[
U_i = \sum_{s=1}^{S_i} a_{i,s} \times U_{i,s}
\]

\[0 < a_{i,s} < 1\]

\[
\sum_{s=1}^{S_i} a_{i,s} = 1
\]
Utility Function as a Function of the Response Time

\[ U_{i,s} = \frac{K_{i,s} \times e^{-R_{i,s} + \beta_{i,s}}}{1 + e^{-R_{i,s} + \beta_{i,s}}} \]
Utility Function as a Function of the Throughput

\[ U_{i,s} = K_{i,s} \times \left( \frac{1}{1 + e^{-X_{i,s} + \beta_{i,s}}} - \frac{1}{1 + e^{\beta_{i,s}}} \right) \]
Workload Variation for Online AEs

![Graph showing workloads](image-url)
Response Times for Class 1 of AE 1

© 2012 D.A. Menasce. All Rights Reserved.
Response Times for Class 2 of AE 1

![Graph showing response times with control intervals and two curves labeled R1,2 (no controller) and R1,2 (controller).]
Response Times for Class 3 of AE 1

- **Resp. Time (sec)**
  - **Control Interval**
  - **R1,3 (no controller)**
  - **R1,3 (controller)**

© 2012 D.A. Menasce. All Rights Reserved.
Variation of the Number of Servers

![Graph showing variation of the number of servers over control intervals for different configurations: N1 (no controller), N1 (controller), N2 (no controller), N2 (controller), N3 (no controller), and N3 (controller).](image)
Variation of Global Utility

![Graph showing the variation of Global Utility with and without a controller.](image-url)
Examples of Autonomic Computing Systems

• E-commerce system
• Internet Data Center
• Virtualized Environment
• SOA Software Systems

CPU Allocation Problem for Autonomic Virtualized Environments

• Existing systems allow for manual allocation of CPU resources to VMs using CPU priorities or CPU shares.

• Need automated mechanism for the adjustment of CPU shares or the priorities of the virtual machines in order to maximize the global utility of the entire virtualized environment.
CPU Allocation Problem for Autonomic Virtualized Environments (Cont’d)

\[
\begin{align*}
\text{Workload 1} & \quad U_{1,1} \\
\text{Workload 2} & \quad U_{n,1} \\
\ldots & \\
\text{Workload n} & \\
\end{align*}
\]

Virtual Machine 1

Virtual Machine 2

Virtual Machine M

Virtualized Environment
Virtualization Controller Architecture

- CPU
  - disk 1
  - ... disk k

Performance Predictor

Utility Function Computation

Performance Monitor

Autonomic Controller Algorithm

to the VMM resource manager

SLAs

workload
CPU Shares Based Allocation: Workload Variation

Arrival Rate (tps)

Control Interval
CPU Shares Based Allocation: CPU Shares Variation
CPU Shares Based Allocation: Response Time for VM1

Graph showing response time (VM1 Resp. Time (sec)) against control interval with and without a controller.
CPU Shares Based Allocation: Response Time for VM2

no controller

with controller
CPU Shares Based Allocation: Global Utility

with controller

no controller
Examples of Autonomic Computing Systems

• E-commerce system
• Internet Data Center
• Virtualized Environment
• SOA Software Systems

Self-Architecting SOA Software Systems (SASSY)

• SASSY allows domain experts to specify requirements using a high-level visual activity language.
• SASSY generates the initial software architecture optimized to maximize a utility function.
• The running system is constantly monitored and SASSY automatically re-architects the system when needed.
Specify SASs and SSSs

Generation of Base Architecture

Service Directory

QoS Pattern Library

Adaptation Pattern Library

Service Discovery

Develop and Register Services

Develop QoS Architectural Patterns

Develop Software Adaptation Patterns

Domain Experts

Software Engineers

Develop and Register Services

Develop QoS Architectural Patterns

Develop Software Adaptation Patterns

Specify SASs and SSSs

Generation of Base Architecture

Self-Architecting

Service Binding and Coordination Logic Deployment

Near-optimal Architecture

Adaptation Pattern Library

QoS Pattern Library

Service Directory

Service Coordination

Base Architecture

Running system

Service Activity Schema (SAS) + SSSs

© 2012 D.A. Menasce. All Rights Reserved.
SASSY: Run-time Adaptation

- Analyze and Determine Need to Re-Architect
- Monitor Running System
- Architecture of running system
- Plan for Re-architecting
- Execute Software Adaptation Control

KEY:
- ↔ model r/w
- → model read
- → communication
- ← human-computer interaction

© 2012 D.A. Menasce. All Rights Reserved.
Architecture Adaptation
Search Trajectories

© 2012 D.A. Menasce. All Rights Reserved.
Concluding Remarks

- Self-optimization is a key attribute of AC systems.
- Performance models and combinatorial search techniques can be useful in dynamically adjusting a system’s configuration as the workload changes.
Thanks! Questions?

www.cs.gmu.edu/faculty/menasce.html