Autonomic Computing

Daniel A. Menascé
Department of Computer Science
George Mason University
Credits

• My former Ph.D. students:
  – Mohamed Bennani, Oracle
  – Ron Dodge, West Point
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.cachessize</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
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

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
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

Monitoring

Service Level Objectives
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

Service Level Objectives

Controller

Parameter Change
Examples of Autonomic Computing Systems

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

• E-commerce system
• Internet Data Center
• Virtualized Environment
Multi-tier Architecture
arriving requests

E-commerce System

Service Demand Computation (2)

Workload Analyzer (3)

QoS Controller Algorithm (5)

Performance Model Solver (4)

QoS Controller

completing requests

QoS goals

arriving requests

completing requests
E-commerce System

arriving requests

$U_i$

Service Demand Computation
(2)

QoS Controller Algorithm
(5)

Workload Analyzer
(3)

Performance Model Solver
(4)

QoS Controller

completing requests

$X_0$

QoS goals
E-commerce System

$D_i = \frac{U_i}{X_0}$

QoS Controller

$U_i$

Service Demand Computation

QoS Controller Algorithm

Performance Model Solver

Workload Analyzer

© 2007 D.A. Menasce. All Rights Reserved.
arriving requests

E-commerce System

Service Demand Computation
(2)

QoS Controller Algorithm
(5)

Workload Analyzer
(3)

Performance Model Solver
(4)

QoS Controller

completing requests

QoS goals

arriving requests

QoS Controller

(1) (2) (3) (4) (5) (6) (7) (8) (9) (10) (11) (12)
E-commerce System

arriving requests

Service Demand Computation
(2)

QoS Controller Algorithm
(5)

Workload Analyzer
(3)

Performance Model Solver
(4)

QoS Controller

(1) (6)

(12) (7)

(8)

(10)

(11)

(9)

QoS goals

© 2007 D.A. Menasce. All Rights Reserved.
E-commerce System

arriving requests

Service Demand Computation

(2)

Workload Analyzer

(3)

QoS Controller Algorithm

(5)

Performance Model Solver

(4)

QoS goals

completing requests

QoS Controller

arriving requests

completing requests
Controller Interval

reconfiguration commands

$i$-th interval

measurements from servers in e-commerce site.

$(i+1)$-th interval

measurements from servers in e-commerce site.

controller algorithm
Self-Optimization Approach

- The space of configuration points is searched using combinatorial search techniques.
- Each point has a QoS value computed through an analytic performance model.
Self-Optimization Approach

- The space of configuration points is searched using combinatorial search techniques.
- Each point has a QoS value computed through an analytic performance model.
Self-Optimization Approach

- The space of configuration points is searched using a combinatorial search technique.
- Each point has a QoS value computed through an analytic performance model.

\[ QoS = f(\vec{W}, c_1, c_2, \ldots, c_m) \]
A Queuing Model is Used to Compute QoS Values
Combined QoS Metric

\[ QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X \]

\( w_R, w_P, \) and \( w_X \) are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.
QoS Metric

\[ QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X \]

\(w_R\), \(w_P\), and \(w_X\) are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.

\(\Delta QoS_R\), \(\Delta QoS_P\), and \(\Delta QoS_X\) are relative deviations of the response time, throughput, and probability of rejection metrics with respect to their desired levels.
QoS Metric

\[ QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X \]

\(w_R\), \(w_P\), and \(w_X\) are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.

\(\Delta QoS_R\), \(\Delta QoS_P\), and \(\Delta QoS_X\) are relative deviations of the response time, throughput, and probability of rejection metrics with respect to their desired levels.

The QoS metric is a dimensionless number in the interval \([-1, 1]\).
QoS Metric

\[ QoS = w_R \times \Delta QoS_R + w_P \times \Delta QoS_P + w_X \times \Delta QoS_X \]

- \( w_R, w_P, \) and \( w_X \) are relative weights that indicate the relative importance of response time, throughput, and probability of rejection.

- \( \Delta QoS_R, \Delta QoS_P, \) and \( \Delta QoS_X \) are relative deviations of the response time, throughput, and probability of rejection metrics with respect to their desired levels.

The QoS metric is a dimensionless number in the interval \([-1, 1]\).

\[ \text{If all metrics meet or exceed their QoS targets, } QoS \geq 0. \]
Response Time Deviation

\[ \Delta QoS_R = \frac{R_{max} - R_{measured}}{\max(R_{max}, R_{measured})} \]

- = 0 if the response time meets its target.
- > 0 if the response time exceeds its target.
- < 0 if the response time does not meet its target.

- \( \Delta QoS_R \leq 1 - \left( \sum_{i=1}^{K} D_i \right) / R_{max} < 1 \)

- \(-1 < -(1 - R_{max} / R_{measured}) \leq \Delta QoS_R \)
Probability of Rejection Deviation

\[ \Delta QoS_P = \frac{P_{\text{max}} - P_{\text{measured}}}{\max(P_{\text{max}}, P_{\text{measured}})} \]

- 0 if the probability of rejection meets its target.
- > 0 and ≤ 1 if the probability of rejection exceeds its target.
- < 0 and ≥ -1 if the probability of rejection does not meet its target.
Throughput Deviation

\[
\Delta QoS_X = \frac{X_{measured} - X^*_\text{min}}{\max(X_{measured}, X^*_\text{min})}
\]

- \( X^*_\text{min} = \min(\lambda, X_{\text{min}}) \)
- = 0 if the throughput meets its target.
- > 0 and < 1 if the throughput exceeds its target.
- < 0 and > -1 if the throughput does not meet its target.
Prototype Configuration

QoS Controller

Database Server

Web Server

Application Server

100 Mbps Hub

Workload Generator

Workstation

TPC-W site
Experiment Results

Arrival rate

![Graph of Arrival Rate vs Time (Controller Intervals)]
Results of QoS Controller

![Graph showing the comparison between Controlled QoS and Uncontrolled QoS.](Graph)

- **Controlled QoS**
- **Uncontrolled QoS**

Arrival Rate (req/sec)

© 2007 D.A. Menasce. All Rights Reserved.
Experiment Results

Arrival rate

QoS is not met!
Examples of Autonomic Computing Systems

• E-commerce system
• Internet Data Center
• Virtualized Environment
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 (cont’d)

• 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

Decides how many servers to assign to each AE.

Implements Global Controller’s decisions.
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)

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

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

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

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

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

![Graph showing workload variation for online AEs](image-url)
Response Times for Class 1 of AE 1

![Graph showing response times for Class 1 of AE 1. The graph compares response times with and without a controller. The x-axis represents control intervals, and the y-axis represents response time in seconds. The graph includes two lines: one for R1,1 (no controller) and another for R1,1 (controller).](image-url)
Response Times for Class 2 of AE 1

[Graph showing response times for R1,2 with and without a controller.]
Response Times for Class 3 of AE 1

![Graph showing response times for different control intervals with and without a controller.](image-url)
Variation of the Number of Servers

![Graph showing variation of the number of servers with control intervals and controller/No controller scenarios.]
Variation of Global Utility

Control Interval

Ug

No Controller

With Controller
CPU Utilization for Servers in AE 1

![Graph showing CPU utilization with and without a controller.](image-url)
Examples of Autonomic Computing Systems

• E-commerce system
• Internet Data Center
• Virtualized Environment
Autonomic Virtualized Environments

• OSs (guest OSs) run in user mode and the VMM runs in supervisor mode

• Advantages of virtualization:
  – Increased security and reliability
  – Reduced cost
  – Better maintenance
  – Easy support for legacy systems
  – Load balancing

• Virtual machines are allocated the CPU based on their priority or assigned CPU shares

• Examples: VMWare, Sun Zones, Xen, IBM LPARs.
**Autonomic Virtualized Environments**

- OSs (guest OSs) run in user mode and the VMM runs in supervisor mode

- Advantages of virtualization:
  - Increased security and reliability
  - Reduced cost
  - Better maintenance
  - Easy support for legacy systems
  - Load balancing

- Virtual machines are allocated the CPU based on their priority or assigned CPU shares

- Examples: VMWare, Sun Zones, Xen, IBM LPARs.
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)

– Use **utility functions**: maps performance metrics and SLAs to a scalar.

– Each service class of any given VM has a utility function that depends on the type of the class, the value of the performance metric for the class, and on the SLA for the class.

– The total utility function of a VM is a function of the utility functions of all its service classes.

– The global utility function is a function of the utility functions of all VMs.
CPU Allocation Problem for Autonomic Virtualized Environments (Cont’d)

\[
\begin{align*}
\text{Workload 1} & \quad U_{1,1} \\
\text{Workload 2} & \quad U_{n,1} \\
\vdots & \quad \text{Virtual Machine 1} \\
\text{Workload n} & \quad U_1 \\
\end{align*}
\]

\[
\begin{align*}
\text{Virtual Machine 2} & \quad U_2 \\
\vdots & \quad U_g \\
\text{Virtual Machine M} & \quad U_M
\end{align*}
\]

![Graph showing utility vs. response time](image)
Performance Model-Based Autonomic Computing

state
Performance Model-Based Autonomic Computing

state

(30,45,25) -> (25,50,25) -> (25,60,15) -> (25,45,30)
(35,30,35) -> (15,60,25) -> (15,50,35) -> (35,15,50)
(35,25,40) -> (15,70,15) -> (15,45,40) -> (30,25,45)
Performance Model-Based Autonomic Computing

state

utility

(30,45,25) 0.2

(25,50,25) 0.3

(25,60,15) 0.6

(25,45,30) -0.2

(35,30,35) 0.5

(15,60,25) 0.8

(15,50,35) 0.1

(35,15,50) -0.1

(35,25,40) 0.6

(15,70,15) 0.9

(15,45,40) 0.95

(30,25,45) 0.5

© 2007 D.A. Menasce. All Rights Reserved.
Performance Model-Based Autonomic Computing
Performance Model-Based Autonomic Computing
Performance Model-Based Autonomic Computing

```plaintext
state

(30,45,25) 0.2
(25,50,25) 0.3
(15,60,25) 0.8
(35,30,35) 0.5
(15,70,15) 0.9
(35,25,40) 0.6
(15,45,40) 0.95
(25,60,15) 0.6
(15,50,35) 0.1
(35,15,50) -0.1
(25,45,30) -0.2
(30,25,45) 0.5
```
Performance Model-Based Autonomic Computing
State: CPU share or priority allocated to each VM: 
(a₁, …, aₘ)

The utility function at each state is computed with the help of a predictive queuing network model. The best state to move to is determined using combinatorial search techniques.
Virtualization Controller Architecture

- CPU
- Performance Predictor
- Autonomic Controller Algorithm
- Utility Function Computation
- Performance Monitor

To the VMM resource manager

SLAs

workload
Priority Based CPU Allocation: Workload Variation

Out of synch peaks and valleys to force change of CPU priorities.
Priority Based CPU Allocation: Variation of VM Priority
Priority Based CPU Allocation: Response Time for VM1
Priority Based CPU Allocation: Response Time for VM2

![Graph showing response time for VM2 with and without controller](Image)
Priority Based CPU Allocation: Global Utility

Control Interval

Virtual Environment Global Utility

with controller

no controller
CPU Shares Based Allocation: Workload Variation

![Graph showing CPU shares based allocation with workload variation over control intervals and arrival rates in tps.](image-url)
CPU Shares Based Allocation: CPU Shares Variation
CPU Shares Based Allocation: Response Time for VM1

![Graph showing CPU response times with and without controller](image-url)
CPU Shares Based Allocation: Response Time for VM2

![Graph showing response times with and without controller](image-url)
CPU Shares Based Allocation: Global Utility

Control Interval

Virtual Environment Global Utility

with controller

no controller
Concluding remarks

• Autonomic computing is an area of growing importance due to its potential to deal with complexity.

• Other techniques used:
  – Machine learning
  – Control theory
Bibliography


Thanks! Questions?

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