Providing Automated Detection of Problems in Virtualized Servers using Monitor framework

Gunjan Khanna, Saurabh Bagchi
(Purdue University)
Kirk Beaty, Andrzej Kochut, Norman Bobroff and Gautam Kar
(IBM T. J. Watson Research Center)

Background

- Detection and diagnosis of problems in IT systems is a challenging task
  - Performance, configuration etc.
- Server virtualization is a widely adopted practice
- Virtualization is a way of providing indirection
  - Server Virtualization aims at converting physical machine(s) into virtual server(s) and deploying on top of a hypervisor
- Provides isolation, homogeneity, and increased resource utilization
  - Prevents server sprawl
**Motivation**

- Virtualization Technology (hypervisor) and Management systems have been developed independently
  - Both exercise control over the OS
- Dynamic resource allocation via hypervisor further heightens the problem
- Current management systems only concentrate on system resources
  - Centralized and inflexible
  - Most require management agents to be co-located

**Example Virtual Scenario**

![Virtual Scenario Diagram]

IBM Blade Center

<table>
<thead>
<tr>
<th>ESX Hypervisor</th>
<th>ESX Hypervisor</th>
<th>ESX Hypervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual Server 4</td>
<td>Director Agent</td>
<td>Virtual Server 2</td>
</tr>
<tr>
<td>Virtual Server 3</td>
<td>Director Agent</td>
<td>Virtual Server 1</td>
</tr>
<tr>
<td>Director Agent</td>
<td>Director Agent</td>
<td>Director Agent</td>
</tr>
</tbody>
</table>

IBM Director Server

SAN
Challenges in Addressing Problems in Virtualized Environments

• How to quickly detect the problem arising due to resource contention between co-located virtual machines?
• How to interpret the system metrics from one virtual machine with the metrics from physical machine?
  – System Metrics: CPU, Memory, I/O etc.
• How to correlate the application state information with the system metrics?
• Address error propagation and scalability?

Monitor(s) for Virtualized Environments

Characteristics
• Observes external interactions between system components
• Uses fast matching against a database of rules to detect problems
• Hierarchical topology allowing for correlation and filtering of information
• Can work with black-box components
Rule Framework

- Combinatorial Rules
  - $S_1 \land S_2$

- Temporal Rules
  - 5 different types of temporal rules
  - Expressive enough to form rules for wide range of applications and scenarios
    - Type I:
      \[ S_y = \text{true for } T \in (t_x, t_x + k) \Rightarrow S_y = \text{true for } T \in (t_f, t_f + b) \]
    - Type IV:
      \[ \text{If } S_y = \text{true, and } \forall t \in (t_0, t_0 + a) \; L \leq |V| \leq U \Rightarrow L' \leq |B_y| \leq U' \forall t \in (t_0, t_0 + b) \]
Detecting Problems

• Avoid simultaneous Peak Resource Usage for co-located virtual machines
  – $S_i$ is the state of peak resource usage for machine $i$.
  – $\tau( S_1 \land S_2 \land S_3 )$

• Fault propagation from one physical machine to another
  – WAS sends more messages to a DB2 replica
  – Suitable assertions should be placed

• Correlation of alarms
  – Derived alarms at higher level Monitor(s)

$\exists t = T \text{ s.t. } S_i = \text{true} \Rightarrow A_L \leq A_{LM} \leq A_U \forall t \in (T, T + \alpha)$

Relating the system resources with Application semantics

• Simply looking at system metrics like CPU, memory etc. does not provide a complete view of the system
  – Composite Rules relating the application with system metrics are necessary

• DB2 (Data Base) table contention
  – Response Time $\exists t = T \text{ s.t. } S_j = \text{true} \Rightarrow R_{max} \leq R_{DB} \leq \infty \forall t \in (T, T + \delta)$
  – Open Connections $\exists t = T \text{ s.t. } S_j = \text{true} \Rightarrow L_o \leq O_c \leq U_o \forall t \in (T, T + \delta)$

• WAS (Application Server) resource contention
  – $C_w$ and WAS$_{CPU}$ are application and system metric respectively
  
  If $C_o \leq C_w \leq C_i \forall t \in (T, T + \delta)$
  
  $\Rightarrow L_{CPU} \leq W A S_{CPU} \leq U_{CPU} \forall t \in (T, T + \delta)$
Conclusions

• We show the mismatch between the current management systems and virtualization layer(s)
• We propose a hierarchical management framework for addressing problems faced in a virtualized server system
• Address some of the challenges raised by virtualization
• As a future work we are working on diagnosis of problems in virtualized scenarios

Thank You!
### Differences from existing Management framework(s)

<table>
<thead>
<tr>
<th>Properties</th>
<th>Monitor Framework</th>
<th>Existing Management Framework(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Hierarchical</td>
<td>Centralized</td>
</tr>
<tr>
<td>Adaptability</td>
<td>Re-configurable</td>
<td>Rigid</td>
</tr>
<tr>
<td>Internal Access</td>
<td>Non-Intrusive</td>
<td>Require Agents</td>
</tr>
<tr>
<td>Applicability</td>
<td>Generic in nature</td>
<td>Sensors for specific applications</td>
</tr>
</tbody>
</table>

#### Example Virtualized Server Scenario

- **IBM Director Server**
  - Management Framework
  - Provides discovery of systems and remote management

![Virtualized Server Diagram]

- ESX Hypervisor
- VM 1
- DA
- IBM BladeCenter
- Virtual Servers
- IBM Director Agents
- SAN

7/2/2006  IBM | Dependable Computing Systems Lab