Opportunities and Challenges in Applying Control Theory to Cloud Infrastructure Management Automation

Xiaoyun Zhu, VMware

December 6, 2011
Agenda

Top trends in IT
Key problems in cloud infrastructure management (CIM)
Challenges and opportunities in CIM automation
Top IT trend in 2012

Virtualization is No. 1 in Gartner’s top 10 technology-related trends that will impact enterprise infrastructure in 2012

Key virtualization benefits

- Higher hardware utilization
- Easier deployment
- Elastic capacity
- Better agility via live migration
- Higher availability
- Fault tolerance
- Lower energy cost
Virtualization adoption trends

The **majority** of server deployments during 2010 have been virtual.

Someone turns on **1 VM** every **6 seconds**

Virtual machine installed base will grow **5X** in 3 years (from 11M in 2009 to 58M in 2012)*

By 2012, **50%** of world’s applications running on x86 architecture servers will run on virtual machines*

Average **5.5 vMotions** every second

* [Gartner](https://www.gartner.com/): Dec. 2010 projection
From virtualization to the clouds

**Private cloud**
- Virtualized enterprise data centers
- Internal IT run as a business
- Initial setup & ongoing maintenance cost
- High consolidation ratio via statistical multiplexing

**Public cloud**
- IaaS cloud providers such as Amazon EC2, Windows Azure
- Compute and storage resources offered as utilities
- No capital cost and lower barrier to entry – appealing to SMBs
- Lower consolidation ratio w/ 1-1 virtual-to-physical res. mapping

Key to successful cloud infrastructure management – **Automation**!
Key problems in cloud infrastructure management automation

Resource management
• Performance isolation and differentiation
• VM placement & dynamic load balancing
• Application service level guarantees

Energy management
• Cluster-level power management
• Host-level power management

Performance diagnosis
• Anomaly detection
• Root cause analysis

Performance control & optimization of management software
• Online performance optimization
• Feedback-based performance control
Performance isolation and differentiation

VMware solution

- A rich set of **resource controls** (CPU, memory, disk I/O) [Gulati’11]
  - **Reservation** – minimum *guaranteed* amount of resources – used for admission control
  - **Limit** – upper bound on resource consumption – non-work-conserving
  - **Shares** – relative importance during resource contention

- **Resource pool** hierarchy
  - A logical collection of VMs
  - RP-level resource controls (cpu/memory)

- Hypervisor makes **scheduling** decisions based on VM resource controls and estimated demands
Challenges in performance isolation and differentiation

How to estimate VM resource demands accurately?
• CPU ready time, processor states change, memory access pattern
• Memory ballooned, swapped, etc.

Efficient and fair CPU scheduling
• NUMA-aware scheduling
• SMP vCPU co-scheduling

Scale beyond the current 500 VMs/host limit?
• Overhead in monitoring and accurate tracking
VMware solution – Distributed Resource Scheduler (DRS) [Ji’06]

- **Main tasks**
  - Computes per-VM resource **entitlement** $E_i$ based on resource settings and demands
  - Initial VM **placement**
  - Dynamic **load balancing** via live migration

- **Core algorithm highlights**
  - *Cluster imbalance* $I_C = \sigma(N_h)$ where $N_h = (\sum_i E_i) / C_h$
  - Minimizes $I_C$ with a greedy hill-climbing algorithm
  - Cost-benefit analysis considers migration cost
  - Respects hardware compatibility and logical constraints (affinity & anti-affinity)
  - Allows users to control aggressiveness (target imbalance)

- **Currently supported limits**
  - 32 hosts and 3000 VMs per cluster
Dynamic I/O load balancing

VMware solution – Storage DRS (SDRS) [Gulati’10]

- Improves storage performance via optimized placement of virtual disks
- Balances normalized load using workload models and device models
- Supports 32 LUNs and 9000 virtual disks

<table>
<thead>
<tr>
<th>Initial Configuration</th>
<th>Final Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>9diskLUN</td>
<td>9diskLUN</td>
</tr>
<tr>
<td>6diskLUN</td>
<td>6diskLUN</td>
</tr>
<tr>
<td>3diskLUN</td>
<td>3diskLUN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IOPS</th>
<th>Latency (in ms)</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>4172</td>
<td>16.7</td>
<td></td>
</tr>
<tr>
<td>5631</td>
<td>14.9</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Initial IOPS</th>
<th>Initial Latency (in ms)</th>
<th>Final IOPS</th>
<th>Final Latency (in ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4172</td>
<td>16.7</td>
<td>5631</td>
<td>14.9</td>
</tr>
</tbody>
</table>

- Improves storage performance via optimized placement of virtual disks
- Balances normalized load using workload models and device models
- Supports 32 LUNs and 9000 virtual disks

via Storage vMotion
Cluster-level power management

VMware solution: Distributed Power Management (DPM) \([Holler'09]\)

- Consolidates VMs onto fewer hosts during periods of low cluster utilization
- Powers off idle hosts or powers on standby hosts when load increases
- Works together seamlessly with host power management (HPM)
Challenges in VM load balancing and power management

How to ensure stability and avoid VM ping-ponging?

How to scale to larger clusters? [Gulati’11]
  • Hierarchical – across cluster LB, aggregated cluster metrics, diff. time scales?
  • Distributed – how to get a consistent view of the cluster?
  • Statistical – power of two choices, how to handle resource pools?

Main barrier to adopting DPM
  • Performance degradation during load spikes – host power-on takes time
  • Proactive vs. reactive using demand prediction

Consider correlation between different workloads
  • Need time series based performance stats to estimate correlation – expensive
  • Need efficient algorithm to incorporate correlation information

Cloud scale management
  • Cloud-level vs. VM-level resource controls?
Challenges in application service level guarantees

Existing technologies mainly focus on monitoring and reporting

Automated performance assurance is hard [Padala’09]

- Enterprise applications are often distributed or multi-tiered
- Application performance depends on availability of multiple resources
- Enterprise workloads are time varying and nonstationary
- Potential performance interference among co-hosted applications

Need adaptive and scalable modeling

- Offline modeling may be insufficient due to workload variation and system behavior changes
- Online learning and modeling can be helpful but should be low cost
  - Model accuracy may vary and should be tracked systematically
  - Online change point detection
- Amount of information and no. of potential knobs can be overwhelming
  - >1800 performance stats for a single ESX host
  - Automated metric filtering and model adaptation (paper under submission)
Performance diagnosis

VMware solution – vCenter Operations (VC Ops) [Colbert’11]

• Self-learned dynamic thresholds that define normal behavior of any performance metric (data agnostic)
• Defines performance problem as deviation from normal behavior
• Detects metric-level abnormalities for computing health and generating alerts
VMware solution – vCenter Operations (VC Ops) [Colbert’11]

- Assumes typical problems cause multiple metrics to change values
- Trend average “noise” level for an object and only raises alert when number of abnormalities exceeds noise level

Smart Alerts

- **Early warning**: use total anomalies to predict when a problem is happening, sometimes before users are impacted
- **KPI predictive**: prediction that a KPI might soon go abnormal due to an event occurring that has preceded the KPI going abnormal on previous occasions
- **Fingerprint**: set of metric anomalies matches previously seen problem
Challenges in performance diagnosis

- Correlation may not point to root cause
- Faster detection while reducing false alarms
- Probabilistic (*black-box*) vs. absolute (*white-box*) health models
- System metrics only vs. incorporating application metrics
- Scales beyond the current 1000 hosts/10000 VMs per vCenter limit
Management software performance via online optimization

Software performance a convex/concave function of concurrency

- Tradeoff between higher utilization and less congestion
- Online optimization to find the best concurrency level
- Ex-1: Apache Web server response time vs. MaxClients [Liu’03]
- Ex-2: VMware **High Availability**
  - Overall failover time as a function of failover concurrency
Management software performance via feedback control

Software performance involves two competing concerns

- Ex-2: VMware **vCloud Director** [Zhu’11]
  - Tradeoff between faster update and less sampling overhead
vCloud Director: Average WWTR vs. polling interval

- **Total actual latency**
  \[ A_{total} = \sum_{i=1}^{i=N} a_i \]

- **Total wasted waiting time**
  \[ W_{total} = \sum_{i=1}^{i=N} w_i = \sum_{i=1}^{i=N} (n_i T - a_i) \]

- **Average wasted waiting time ratio (WWTR)**
  \[ W_{ratio} = \frac{W_{total}}{A_{total}} = \frac{\sum n_i}{\sum a_i} T - 1, \text{ where } n_i = \left\lfloor \frac{a_i}{T} \right\rfloor \]

- Average WWTR for 1000 deploy-vApp operations
- Smaller polling interval => faster status update
vCloud Director: Operation throughput vs. polling interval

- Smaller polling interval => Lower operation throughput
Adaptive vs. fixed polling with changing workload

- 19 concurrent clients, each keeps running deploy-vApp and undeploy-vApp operations
- Drop to 6 concurrent clients after a certain time
- Result for undeploy-vApp ops shown
- Fixed polling – 10 sec; adaptive polling – WWTR setpoint = 50%
Management software performance control and optimization

Opportunities

• Critical problem that may cause user pain points and is expensive to fix
• Some problems demonstrate patterns that are well-understood
• Earlier experience in software performance control can be leveraged

Challenges

• Software engineers who own these features may not understand/favor feedback control designs (slightly lower barrier for optimization)
• Modeling effort often not appreciated – most designs are based on intuition
• Implementation details are critical - sensor/actuator selection, sampling interval, noise handling, etc.
• For common performance behavior patterns, develop tools for automated modeling building and controller design that can be easily adopted by software engineers – should be user-friendly with minimum manual tuning


