Virtualization and the Metrics of Performance & Capacity Management

Has the world changed?

Mark Preston
Agenda

• Reality Check …….  
• General Observations  
• Traditional metrics for a non-virtual environment  
• How virtualization scheduling/dispatching of shared environments can change the paradigm  
• Case Study  
• Conclusions  
• Discussion
Reality Check ……

• Application users don’t care about system utilization as long as response time is met

• Consumers purchase units of computing to support functionality, not CPU cycles
  
  • Internal – Clusters, SAN, Networking
  
  • Cloud - Ex: AWS; “One EC2 Compute Unit provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. This is also the equivalent to an early-2006 1.7 GHz Xeon processor”

• Virtualized environments exhibit a range of latency & neighborhood sensitivity

• Shared environment interference effects are real

• Consumers of IT are realizing that in a virtual world utilization based capacity planning is suspect, if not useless in some cases
Reality Check ……

• System building blocks incur latency
• Virtualization is not a free technology (adds another layer)
• In the drive to reduce enterprise cost host are over-subscribed
• Over-subscribed host and shared resources (Multi-CPU, I/O consolidation) require scheduling, which incurs overhead
• The application team doesn’t have control of the neighborhood
• Let’s reword the opening Reality Check

  • Application admin’s don’t care about system utilization as long as response time is met
General Observations

• The neighborhood is affected by many factors
  • Hypervisor, Over-subscription, Load synchronicity
  • Underlying HW virtualization (AMD-V, Intel VT-x, srIOV, mrIOV)

• Some classes of applications are better suited for virtualization than others
  • Financial trading systems, Infrastructure Monitoring – Less compatible
  • Transaction processing – Depends on workload
  • Web facing – Highly compatible

• Virtualization is best suited for dissimilar applications

• Multi-threaded applications (databases) prefer bare metal

• Intra-tier sensitivity may produce unexpected application effects
  • A Virtualized DB, sensitive to virtualization latency, may result in low Web tier utilization and “Server is too busy” messages

• Cloud adds both on-site and connection latency
Traditional Metrics

• CPU Utilization
  • CPU is generally considered to be a proxy for all other performance/capacity vectors
    • Memory
    • Network
    • I/O
  • Assumption - Interference effects from other workloads do not occur for the CPU vector

• Free Memory
  • An indication of goodness
  • Virtualization can over-commit memory
  • ccNUMA can add latency - local L1 cache vs. ‘remote’ inter-socket L2/L3 cache

• Storage Throughput
  • Typically specified as IOPs, MB/s
  • Assumption - Interference effects from other workloads do not occur for the CPU vector
How Does Virtualization Change Things?

- Shared-occupancy host environments exhibit not-easily-visible interference effects
- Resource scheduler / dispatcher design sharpens this interference effect
- CPU busy and storage throughput at the guest level is no longer a reliable indicator of workload performance
- Usage and throughput metrics are increasingly decoupled
Virtualization adds layers and latency

Virtualization
Scheduling, Load Balancing & Bandwidth Management

Increasing Latency

2 vCPU/Mem vFS, vFA, vNIC
2 vCPU/Mem vFS, vFA, vNIC
4 vCPU/Mem vFS, vFA, vNIC
4 vCPU/Mem vFS, vFA, vNIC

12 vCPUs
Over Subscription

pCPU pCPU pCPU pCPU
pCPU pCPU pCPU pCPU

Cache Cache Cache Cache

pMemory

Switch SAN

© Copyright 2011 | First Data Corporation
Each Layer Incurs Latency

Designs, Lessons and Advice from Building Large Distributed Systems
“Numbers Everyone Should Know”
Jeff Dean Google Fellow
Virtualization exhibits the classic latency curve

Source: VMware ESX 3: Ready Time Observations - Feb 2004
Storage also follows the traditional latency curve
Capacity Planning with Traditional Metrics

%CPU < 60-70% ..... The world is happy! ..... I think

Traditional allocation of resources

Produces an ESX host with apparently significant capacity
Case Study

- Our application provides:
  - Infrastructure discovery
  - Event correlation
  - Service assurance
- Deployed across entire enterprise
- Composed of multiple tiers / components
  - Collect data across thousands of monitored resources
  - Ingest and processing of events
  - Console functionality for operation centers/admins
- Some components exhibit synchronous behavior
- Reporting elements can be resource hogs
- Users characterized application as “Unusable”
Environment Capacity Audit

**No Problem Found**

<table>
<thead>
<tr>
<th>Host</th>
<th>CPU Utilization</th>
<th>Memory Utilization</th>
<th>Storage Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESX Host A</td>
<td>16-20%</td>
<td>~12.5%</td>
<td>30MB/s</td>
</tr>
<tr>
<td>ESX Host B</td>
<td>19-25%</td>
<td>~28%</td>
<td>3MB/s</td>
</tr>
</tbody>
</table>
| ESX Host C    | ~10%            | ~12.5%             | 22MB/s for App A
|               |                 |                    | kB/s for App of Interest            |
Performance Analysis

Significant waiting for resources

Traditional utilization metrics indicate significant capacity

Virtualization metrics indicate significant waiting for CPU resources

%Rdy – Ready-to-Run but no physical CPU resource free

5% - Could be sign of trouble

10% - There is a problem

10-15% - In this case the application was unusable
## Environment Performance Audit

<table>
<thead>
<tr>
<th>Host</th>
<th>CPU Utilization</th>
<th>Memory Utilization</th>
<th>Storage Throughput</th>
<th>CPU %Rdy Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESX Host A</td>
<td>16-20%</td>
<td>~12.5%</td>
<td>30MB/s</td>
<td>10%</td>
</tr>
<tr>
<td>ESX Host B</td>
<td>19-25%</td>
<td>~28%</td>
<td>3MB/s</td>
<td>35-55%</td>
</tr>
<tr>
<td>ESX Host C</td>
<td>~10%</td>
<td>~12.5%</td>
<td>➢ 22MB/s for App A</td>
<td>22-35%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>➢ kB/s for App of Interest</td>
<td></td>
</tr>
</tbody>
</table>
Resolution

• Starting point
  • Performance team brought in after many months of application team trying to make application ‘usable’
  • Existing infrastructure - ESX 3.5 + vmware tools + multi-core ESC host + SAN
  • Followed a structured use-case driven performance test, analysis, recommendations process

• Initial relatively short timeframe solution
  • ESX 4.1 + vmware tools + same multi-core ESX host + SAN
  • Result – better but still judged “Not Usable”

• Final solution
  • ESX 4.1 + vmware tools + new 24-core based server cluster + new dedicated VM neighborhood layout + dedicated one-to-one vCPU-to-pCPU + load-balanced SAN
  • Although one step away from a physical environment, virtualization still provides ‘ility’ benefits
Performance issues resolved

%Rdy reduced to less than 5%!

CPU utilization has virtually quadrupled!

%CPU, a dependent metric, has increased to acceptable levels.
Latency can be cumulative

- From an overall service perspective the impact of %Rdy is application, neighborhood and path dependent.
- From a host perspective the impact of %Rdy can be summation dependent.
- For sequential task the cumulative delay of %Rdy is most likely unacceptable.

Careful latency driven selection and layout of VMs is critical to service performance.
Summary

• Old metrics can be misleading
  • Low CPU utilization may be an indication of significant waiting for CPU resources
  • Low storage throughput may be an indication of significant waiting for SAN resources

• Review your application for suitability
  • Financial trading systems, Infrastructure Monitoring – Less compatible
  • Transaction processing – Depends on workload
  • Web facing – Highly compatible

• Adopt /augment new metrics for virtualized environments
  • CPU – For vmware ESX control for %Rdy
  • Storage – Latency is unique to class, architecture and technology of storage

Similar to BTM, Control for Latency