Determining Overhead, Variance & Isolation Metrics in Virtualization for IaaS Cloud

Bukhary Ikhwan Ismail
Devendran Jagadisan
Mohammad Fairus Khalid
MIMOS BERHAD
Contents

1. Introduction
2. Testing Methodologies
3. Setup & Tools
4. Results
5. Discussion
6. Summary
• What is IaaS
  – Commodity base resources
  – User self service consumable resource
  – Consist of
    • Resources (Storage/Compute/Network)
    • Management Software

• IaaS Nature
  – Distributed / Inter-related
  – Virtualization - KVM
Objective

• Understanding of behavior of VM in KVM hypervisor
• Benchmark the performance of VM
  – Overhead / Isolation / Variances
• Determine the performance factor
• Understanding design decision
Methodologies

- Benchmark VM’s
  - Processor, Memory, Storage and Network.

- Guest OS Behavior (Micro Level)
  - Overhead
    - Virtualization Overhead
  - Variance
    - The Effects of more VM resides in a single physical
  - Isolation or Fairness
    - Obeservation of VM in isolated container.

- Application Specific Behavior (Macro Level)
  - Overhead/variance/Isolation of Java & MySQL
• Benchmark tools
• Open source / free
• Results can be comparable with others
• Specific tool for each component
Setup & tools

libraries
• KVM 85
• KMOD-KVM -2.6.3.0.1
• Libvirt-0.6.5
• QEMU-KVM 10.6.0

Network Setup
• NIC capacity – 1Gbps
• Switch – 1Gbps
• Head node - NAT/forwarding
## Setup & tools

<table>
<thead>
<tr>
<th>Host</th>
<th>Guess VM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Processor</strong></td>
<td>4 Cores Intel Xeon CPU E5405 @ 1.99GHz</td>
</tr>
<tr>
<td><strong>Mainboard</strong></td>
<td>DellPrecision WorkStation T5400</td>
</tr>
<tr>
<td><strong>Chipset</strong></td>
<td>Intel 5400 Chipset Hub</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>2 x 4096 MB 667MHz</td>
</tr>
<tr>
<td><strong>Disk</strong></td>
<td>750GBHitachi HDS72107</td>
</tr>
<tr>
<td><strong>Graphics</strong></td>
<td>nVidia Quadro FX 1700,</td>
</tr>
<tr>
<td><strong>OS</strong></td>
<td>CentOS 5.3 64bit</td>
</tr>
<tr>
<td><strong>Kernel</strong></td>
<td>Kernel:2.6.18-128.4.1.el5 (x86_64)</td>
</tr>
<tr>
<td><strong>File System</strong></td>
<td>EXT3</td>
</tr>
</tbody>
</table>
Results - Overhead

Processor

<table>
<thead>
<tr>
<th>MIPS</th>
<th>Host</th>
<th>Guest</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6139</td>
<td>6077</td>
</tr>
</tbody>
</table>

Memory

<table>
<thead>
<tr>
<th>MB/s</th>
<th>Host</th>
<th>Guest</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2579</td>
<td>2282</td>
</tr>
<tr>
<td></td>
<td>2367</td>
<td>2323</td>
</tr>
<tr>
<td></td>
<td>2366</td>
<td>2328</td>
</tr>
</tbody>
</table>

Categories:
- int Add
- int Copy
- int Scale
Results - Overhead

Storage

<table>
<thead>
<tr>
<th></th>
<th>4GB Write</th>
<th>4GB Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>179</td>
<td>1296</td>
</tr>
<tr>
<td>Guest</td>
<td>96</td>
<td>355</td>
</tr>
</tbody>
</table>

MB/s
Results - Overhead

Network Throughput

<table>
<thead>
<tr>
<th>Mb/s</th>
<th>TCP_STREAM</th>
<th>UDP_STREAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>941</td>
<td>852</td>
</tr>
<tr>
<td>800</td>
<td>232</td>
<td>405</td>
</tr>
<tr>
<td>600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Network Latency

<table>
<thead>
<tr>
<th>TPS</th>
<th>Between hosts</th>
<th>between guests in single host</th>
<th>between guests across node</th>
</tr>
</thead>
<tbody>
<tr>
<td>15,000</td>
<td>10,623</td>
<td>3,692</td>
<td>2,866</td>
</tr>
<tr>
<td>10,000</td>
<td>10,623</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5,000</td>
<td>10,623</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>10,623</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results – Overhead Applications

Java Sun Flower

- Rendering Time
  - Seconds

SQLite

- Time
  - Seconds

Host

Guest
Results – Variance & Fairness

**CPU Variance**

<table>
<thead>
<tr>
<th>MIPS</th>
<th>1 guest</th>
<th>2 guest</th>
<th>4 guest</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1792</td>
<td>1720</td>
<td>1373</td>
</tr>
</tbody>
</table>

**CPU Fairness**

<table>
<thead>
<tr>
<th>MIPS</th>
<th>1st guest</th>
<th>2nd guest</th>
<th>3rd guest</th>
<th>4th guest</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1396</td>
<td>1370</td>
<td>1394</td>
<td>1334</td>
</tr>
</tbody>
</table>
Results – Variance & Fairness

Memory Variance

![Memory Variance Chart]

- **int Add**: 2081 MB/s (1 Guest), 1169 MB/s (2 Guests), 611 MB/s (4 Guests)
- **int Copy**: 2122 MB/s (1 Guest), 1186 MB/s (2 Guests), 618 MB/s (4 Guests)
- **int Scale**: 2125 MB/s (1 Guest), 1164 MB/s (2 Guests), 601 MB/s (4 Guests)

Memory Fairness

![Memory Fairness Chart]

- **int Add**: Instance 1: 600 MB/s, Instance 2: 608 MB/s, Instance 3: 621 MB/s, Instance 4: 613 MB/s
- **int Copy**: Instance 1: 608 MB/s, Instance 2: 607 MB/s, Instance 3: 633 MB/s, Instance 4: 625 MB/s
- **int Scale**: Instance 1: 592 MB/s, Instance 2: 578 MB/s, Instance 3: 619 MB/s, Instance 4: 613 MB/s
Results – Variance & Fairness

**Iozone Disk Variance**

- **4GB Write**
  - 1 Guest: 96 MB/s
  - 2 Guest: 47 MB/s
  - Variance: 49 MB/s

- **4GB Read**
  - 1 Guest: 355 MB/s
  - 2 Guest: 8 MB/s
  - Variance: 347 MB/s

**Iozone Disk Fairness**

- **4GB Write**
  - 1st Guest: 47.18 MB/s
  - 2nd Guest: 50.96 MB/s

- **4GB Read**
  - 1st Guest: 8.49 MB/s
  - 2nd Guest: 9.03 MB/s
Results – Variance & Fairness

Tiobench Disk Variance

<table>
<thead>
<tr>
<th></th>
<th>1 Guest</th>
<th>2 Guest</th>
</tr>
</thead>
<tbody>
<tr>
<td>64MB Write</td>
<td>17</td>
<td>23</td>
</tr>
<tr>
<td>64MB Read</td>
<td>3</td>
<td>39</td>
</tr>
<tr>
<td>256MB Write</td>
<td>23</td>
<td>31</td>
</tr>
<tr>
<td>256MB Read</td>
<td>52</td>
<td>131</td>
</tr>
</tbody>
</table>

Tiobench Disk Fairness

<table>
<thead>
<tr>
<th></th>
<th>1st Guest</th>
<th>2nd Guest</th>
</tr>
</thead>
<tbody>
<tr>
<td>64MB Write</td>
<td>30</td>
<td>16</td>
</tr>
<tr>
<td>64MB Read</td>
<td>76</td>
<td>3</td>
</tr>
<tr>
<td>256MB Write</td>
<td>39</td>
<td>23</td>
</tr>
<tr>
<td>256MB Read</td>
<td>202</td>
<td>60</td>
</tr>
</tbody>
</table>
Results – Variance & Fairness

Java Variance

- Rendering time
  - 1 Guest: 36.95
  - 2 Guests: 38.35
  - 4 Guests: 45.03

SQLite Variance

- Time to complete
  - 1 Guest: 114
  - 2 Guests: 121
  - 4 Guests: 136
Results – Variance & Fairness

Network Throughput Variance

Network Latency Variance

www.mimos.my © 2009 MIMOS Berhad. All Rights Reserved.
Results – Variance & Fairness

Network TCP Fairness

Network UDP Fairness
## Discussion

<table>
<thead>
<tr>
<th>ELEMENTS</th>
<th>OVERHEAD</th>
<th>VARIANCE</th>
<th>FAIRNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1%</td>
<td>GOOD</td>
<td>20%</td>
</tr>
<tr>
<td>Memory</td>
<td>5%</td>
<td>GOOD</td>
<td>70%</td>
</tr>
<tr>
<td>Iozone_Write</td>
<td>46%</td>
<td>FAIR</td>
<td>46%</td>
</tr>
<tr>
<td>Iozone_Read</td>
<td>72%</td>
<td>BAD</td>
<td>98%</td>
</tr>
<tr>
<td>Tiobench_Write</td>
<td>1087%</td>
<td>V.BAD</td>
<td>35%</td>
</tr>
<tr>
<td>Tiobench_Read</td>
<td>657%</td>
<td>V.BAD</td>
<td>1200%</td>
</tr>
<tr>
<td>Java</td>
<td>1%</td>
<td>GOOD</td>
<td>21%</td>
</tr>
<tr>
<td>Sql</td>
<td>11%</td>
<td>GOOD</td>
<td>26%</td>
</tr>
<tr>
<td>Net Tcp</td>
<td>-</td>
<td>N/A</td>
<td>49%</td>
</tr>
<tr>
<td>Net Udp</td>
<td>N/A</td>
<td>39%</td>
<td>FAIR</td>
</tr>
<tr>
<td>Network Latency</td>
<td>N/A</td>
<td>32%</td>
<td>GOOD</td>
</tr>
</tbody>
</table>
Discussion

• CPU
  – Good results for all metrics.

• Memory
  – Shows bad variance.

• Disk
  – disk I/O bottleneck.
    • High overhead
    • very bad distribution of variance and fairness.
    • Tiobench it gives fluctuate results.
  – Disk performs better in Para-virtualization

• Application
  – Application specific benchmark result, does not reflect bad CPU, memory and storage performance.
Discussion

• Overhead
  – CPU & Memory gives low overhead
  – Provisioning decision based on Bare Metal Vs. Virtualization
  – Application specific scenarios (Grid/Rendering/Web)

• Variance
  – More load (VM’s) affects overall performance

• Fairness/Isolation
  – Behavioral understanding of KVM hypervisor
Summary

• What we gain?
  – KVM specific behavior
  – Important role of VM scheduling
  – Storage plays important role

• Future Work
  – Different benchmarking methodology
  – Focus on values attain from testing
  – Determine performance improvements weight age (%)
    • Hardware choice
    • Software stacks (virtualization/libraries)
    • Host OS (Kernel&TCP parameters tweaking /OS choices
    • Different I/O Scheduler
Summary

- Hypervisor specific results – KVM (Virtio devices)
- Investment on hardware or software stack?
- Which area contribute to increase the performance and larger capacity?
- Variance & isolation issues, makes scheduling policy a crucial component.
THANK YOU