Monitoring and Analyzing Virtual Machines – Resource Overcommitment Detection and Virtual Machine Classification

Hani Nemati
May 5, 2015

Polytechnique Montréal
Laboratoire DORSAL
Agenda

Motivation

• Why detecting resource overcommitment is important?
• Why we need a virtual machine classification model?

Investigations

• Detecting Resource overcommitment for Infrastructure providers
  • Virtual CPU State Detection
    • CPU Overcommitment
    • Memory Overcommitment
  • Virtual Machine Dissection
  • Virtual Machine Classification

Conclusion and in-progress

References
Motivation

Why detecting resource overcommitment is important?

- **Resource Overcommitment** lets the administrator allocate more resources to virtual machines than the physical host has available.
- Identifying Over-committed and Under-committed hosts
- Impact of Resource Overcommitment:
  - **Cloud User:**
    - Latency in response time of programs inside VMs
  - **Infrastructure Provider:**
    - Increase resource utilization
    - Maximizing profit
Motivation
Why we need a virtual machine classification model?

- Classifying virtual machines based on different resource consuming patterns [1][3]

- Categorizing virtual machines into:
  - CPU-intensive
  - Memory-intensive
  - I/O-intensive
  - Network-intensive
  - Compound

- Advantages:
  - Better capacity planning
  - Increase infrastructure revenue by utilizing host resources
  - Better decision for migration of VMs
Investigations

Qemu threads in Control Flow View

Virtual machine executed into ioctl()

Block I/O? Network? CPU processing?
Investigations

CPU Virtualization with intel-VT-x:

- VMX transition [5]
  - Between Guest and Virtual Machine Manager (VMM)
    - vm_entry, vm_exit
  - Virtual Machine Control Structure (VMCS)

```
vm_exit:
- Save VMCS area for Guest
- Load VMCS area for Host
- Check exit reason

vm_entry:
- Save VMCS area for Host
- Load VMCS area for Guest
```

Handle exit reason
Investigations
Investigations

Resource View for Virtual Machine

VMX Root Disk
VMX Root Network
VMX Root Others
Investigations

vCPU State Detection

• Virtual CPU State Detection:
  • VMX Root and Non-Root State
  • IDLE state
  • Wait state
  • Preempted State

• VMX Root
  • Disk I/O
  • Network
  • Memory
Investigations
Our Architecture and tracepoints used for vCPU state detection:

- sched_wakeup: To detect wait state and VMX Root network and Disk
- kvm_exit: To detect VMX Root state
- kvm_entry: To detect VMX Non-root state
- sched_switch: To detect vCPU thread, Preemption state, and IDLE state
Investigations

VM Resource view and Thread view

• During VMX Root with exit reason 30 (I/O Request)
• Wake up a thread from qemu thread pool ==> Disk request
• Wake up a thread from vhost threads ==> Network request
Investigations

For more information about resource usage:

• Disk I/O:
  • Using Qemu trace points
    • `qemu:thread_pool_submit`, `qemu:thread_pool_complete`, `qemu:bdrv_co_io_em`

• Network:
  • Using Host Kernel Trace
    • `net_if_rx`, `net_dev_xmit`, `sched_switch` and `sched_wakup` for `vhost-$(VM-main-thread)`

• Memory:
  • Written Module that investigate accessed paged for a period of time
# Investigations

## VM statistics for vCPU

![VM statistics for vCPU](image)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>23842</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPUQemu</td>
<td>CPUQemu</td>
<td>CPUQemu</td>
<td>CPUQemu</td>
<td>CPUQemu</td>
<td>CPUQemu</td>
<td>CPUQemu</td>
<td>CPUQemu</td>
<td>CPUQemu</td>
<td>CPUQemu</td>
<td></td>
</tr>
<tr>
<td>vmName</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ValueCPU</td>
<td>3809</td>
<td>0</td>
<td>Int</td>
<td>16:49:49</td>
<td>16:49:50</td>
<td>CPUQemu/23842/ValueCPU</td>
<td>CPUQemu/23842/ValueCPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td>3810</td>
<td>0</td>
<td>Int</td>
<td>16:49:49</td>
<td>16:49:50</td>
<td>CPUQemu/23842/STATUS</td>
<td>CPUQemu/23842/STATUS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPUQemu/23842/vCPU</td>
<td>CPUQemu/23842/vCPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3834</td>
<td>0</td>
<td>Int</td>
<td>16:49:49</td>
<td>16:49:50</td>
<td>CPUQemu/23842/vCPU/2</td>
<td>CPUQemu/23842/vCPU/2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3852</td>
<td>0</td>
<td>Int</td>
<td>16:49:49</td>
<td>16:49:50</td>
<td>CPUQemu/23842/vCPU/3</td>
<td>CPUQemu/23842/vCPU/3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3934</td>
<td>0</td>
<td>Int</td>
<td>16:49:49</td>
<td>16:49:50</td>
<td>CPUQemu/23842/vCPU/0</td>
<td>CPUQemu/23842/vCPU/0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3949</td>
<td>0</td>
<td>Int</td>
<td>16:49:49</td>
<td>16:49:50</td>
<td>CPUQemu/23842/vCPU/1</td>
<td>CPUQemu/23842/vCPU/1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPUQemu/23842/statistics</td>
<td>CPUQemu/23842/statistics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPUQemu/23842/statistics/30</td>
<td>CPUQemu/23842/statistics/30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3945</td>
<td>3426</td>
<td>Long</td>
<td>16:49:49</td>
<td>16:49:50</td>
<td>CPUQemu/23842/statistics/7</td>
<td>CPUQemu/23842/statistics/7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3954</td>
<td>80853</td>
<td>Long</td>
<td>16:49:49</td>
<td>16:49:50</td>
<td>CPUQemu/23842/statistics/1</td>
<td>CPUQemu/23842/statistics/1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Investigations

### VM statistics for Disk and Network

<table>
<thead>
<tr>
<th>Type</th>
<th>Value</th>
<th>Unit</th>
<th>Type</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiskQemu</td>
<td>3651</td>
<td></td>
<td>testU1</td>
<td>3653</td>
<td></td>
</tr>
<tr>
<td>numberOfSubmitted</td>
<td>4056</td>
<td>Int</td>
<td>read</td>
<td>4055</td>
<td></td>
</tr>
<tr>
<td>transfer</td>
<td>4057</td>
<td>Int</td>
<td>STATUS</td>
<td>4058</td>
<td>Int</td>
</tr>
<tr>
<td>latency</td>
<td>4067</td>
<td>Long</td>
<td>totalTransfer</td>
<td>4068</td>
<td>Int</td>
</tr>
<tr>
<td>write</td>
<td>5329</td>
<td></td>
<td>testU1</td>
<td>3653</td>
<td></td>
</tr>
<tr>
<td>numberOfSubmitted</td>
<td>5330</td>
<td>Int</td>
<td>write</td>
<td>5329</td>
<td></td>
</tr>
<tr>
<td>transfer</td>
<td>5331</td>
<td>Int</td>
<td>STATUS</td>
<td>5332</td>
<td>Int</td>
</tr>
<tr>
<td>latency</td>
<td>5340</td>
<td>Long</td>
<td>totalTransfer</td>
<td>5341</td>
<td>Int</td>
</tr>
<tr>
<td>NetQemu</td>
<td>3923</td>
<td></td>
<td>Netif</td>
<td>3924</td>
<td>Int</td>
</tr>
<tr>
<td>STATUS</td>
<td>3925</td>
<td>Int</td>
<td>Netdev</td>
<td>3926</td>
<td>Int</td>
</tr>
<tr>
<td>tNetif</td>
<td>3932</td>
<td>Long</td>
<td>tNetdev</td>
<td>3937</td>
<td>Long</td>
</tr>
<tr>
<td>tNetif</td>
<td>3933</td>
<td>Long</td>
<td></td>
<td>3933</td>
<td></td>
</tr>
</tbody>
</table>
VM statistics for vCPU
Investigations


- **Process Identifier (PID)** and **Process Name** inside the VMs is not accessible from Host tracing.
- **CR3** points to the page directory of process inside the VM.
- **SP** points to the stack of the thread inside the virtual machine.
- **CR3** and **SP** are unique identifiers of threads, but to be more human readable we map it with the process info inside the guest.
Investigations

Virtual Machine Dissection

• **New trace point:**
  - `vcpu_enter_guest` to obtain unique tuple (CR3, SP, IP)
  - If you want **more** information:
    - `lttng_statedump_stack` to obtain thread stack range, thread ID, thread name, hostname

*vcpu_enter_guest:*
- Compare SP with the stack range of all threads retrieved from guest
- Save TID and Thread Name
Investigations
Virtual Machine Dissection

- First, fibo runs then after 1 sec, we run cpu_burn program
Investigations

Use Case – Preemptive Virtual Machine

30 runs of Sysbench to find primes < 10000

Average = 324ms  STD = 5ms

Average = 443ms  STD = 116ms
Investigations

Use Case – Frequent Transition – Memory Overcommitment

- VM1, VM2, and VM3 are Memory intensive
- VM4, and VM5 are CPU intensive

<table>
<thead>
<tr>
<th>VM name</th>
<th>Execution Time(ms)</th>
<th>Freq EPT Violation</th>
<th>EPT Violation Percentage(%) Total Time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1</td>
<td>1329.09</td>
<td>3554</td>
<td>237.4</td>
</tr>
<tr>
<td>VM2</td>
<td>1834.5</td>
<td>18801</td>
<td>260.5</td>
</tr>
<tr>
<td>VM3</td>
<td>1332.4</td>
<td>15288</td>
<td>141.2</td>
</tr>
<tr>
<td>VM4</td>
<td>1169.1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>VM5</td>
<td>1857.8</td>
<td>30</td>
<td>0.2</td>
</tr>
</tbody>
</table>
# Investigations

## Overload Analysis

- Compare overload of vCPU State Builder (VSB) with **multi-level** tracing approach by Mohamad Gebai [2]

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Baseline</th>
<th>VSB</th>
<th>Multi-level</th>
<th>VSB Overhead (%)</th>
<th>Multi-level Overhead (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>File I/O (ms)</td>
<td>233</td>
<td>328</td>
<td>361</td>
<td>28.72</td>
<td>35.29</td>
</tr>
<tr>
<td>Memory (ms)</td>
<td>319</td>
<td>331</td>
<td>344</td>
<td>3.67</td>
<td>7.23</td>
</tr>
<tr>
<td>CPU (ms)</td>
<td>339</td>
<td>352</td>
<td>361</td>
<td>3.72</td>
<td>6.11</td>
</tr>
</tbody>
</table>
Investigations

Demo
Investigations

One More Thing ...
Investigations

Virtual Machine Classification

- **KMeans clustering** is a method for cluster analysis in data mining.
- Aim to partition n VMs into K clusters base on resource consuming patterns.

Metrics: disk_transfer, disk_request, CPU_avg, CPU_request

\[ X = \begin{bmatrix} 30512.80 & 503.4 & 9447.93 & 8845.87 \\ 588.97 & 6.827 & 75537 & 3641.5 \\ 46206.73 & 1383.75 & 4298.9 & 23756.9 \\ 722.22 & 8.78 & 205141.1 & 2716.7 \\ 708.41 & 8.78 & 214826.4 & 3373.6 \\ 714.003 & 7.67 & 255922.2 & 2811.57 \end{bmatrix} \]

\[ Y = \begin{bmatrix} 0.94222657 & 0.36056743 & -1.16197163 & 0.17481621 \\ -0.69031714 & -0.61500025 & -0.51162465 & -0.51364317 \\ 1.7984347 & 2.0901037 & -1.21264046 & 2.14731981 \\ -0.68304747 & -0.61116339 & 0.76373987 & -0.63598022 \\ -0.6838009 & -0.61116339 & 0.8590478 & -0.54908229 \\ -0.68349576 & -0.6133441 & 1.26344907 & -0.62343035 \end{bmatrix} \]
Conclusion and in-progress

Inferences

- vCPU state builder (VSB) lets the Infrastructure provider to detect Over-committed and under-committed hosts.

- Profiling threads inside the Vms.

- Kmeans could cluster VMs based on resource usage pattern

Going Further

- Using kmeans with more metrics for clustering VMs based on resource usage pattern

- Applying supervise machine learning algorithms like SVM to Cluster VMs
References


Questions?

Hani.nemati@polymtl.ca
https://github.com/Nemati
Questions?

Hani.nemati@polymtl.ca
https://github.com/Nemati