VI Performance Monitoring

Preetham Gopalaswamy – Group Product Manager
Ravi Soundararajan – Staff Engineer

September 15, 2008
Agenda

- Introduction to performance monitoring in VI
- Common customer/partner questions (use cases)
- Tips and Tricks
The VI platform exposes over 150 performance counters.

Using the VI API, counter values can be retrieved for the entire datacenter including hosts and VMs, or just for a user-defined resource pool of hosts and/or VMs.

A counter is uniquely identified by a combination of its name, group and rollup type. It can be represented using a dotted notation: `<group>.<name>.<roll-up>`

e.g. `cpu.usage.min` is the minimum CPU usage in the sample period.

Every counter includes a description and unit of measure.

VMware Developer Center Blog
Performance Metrics Primer

- Use the VI API to ask the server what counters it exposes. A sample script to accomplish this is available on the VMware website.

- The counters are broadly divided into these categories:
  - CPU
  - Management Agent
  - Resource Group
  - Network
  - Disk
  - Memory
  - System

- The rollup options over a sample period are:
  - none (instantaneous value)
  - average (average over the sampling period)
  - maximum (maximum value in the sampling period)
  - minimum (minimum value in the sampling period)
  - latest (last value in the sampling period)
  - summation (sum of the values over the sampling period)
VirtualCenter collects performance metrics from the hosts that it manages and aggregates the data using consolidation algorithms based on MRTG. The algorithm is optimized to keep the database size constant over time.

If the partner application is also aggregating the data, VMware recommends collecting the consolidated data from VC.

Statistics collection levels (range 1-4) define the number of counters collected and aggregated by VC per provider. VMware recommends that normal operation should be Level 1 or 2. Higher values are for debugging i.e. for short periods of time.

Default stat collection periods and how long they are stored are:

<table>
<thead>
<tr>
<th>Interval</th>
<th>Interval Period</th>
<th>Interval Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per day</td>
<td>5 minutes*</td>
<td>1 day*</td>
</tr>
<tr>
<td>Per week</td>
<td>30 minutes</td>
<td>1 week</td>
</tr>
<tr>
<td>Per month</td>
<td>2 hours</td>
<td>1 month</td>
</tr>
<tr>
<td>Per year</td>
<td>1 day</td>
<td>1 year*</td>
</tr>
</tbody>
</table>

(Items with a * next to them can be configured)
Performance Metrics Primer

- The performance statistic collection level and aggregation are extremely configurable.
- Customers can tune the collection level based on the historical interval. Debugging statistics need not be retained for long periods of time. 
  *e.g. Per HBA statistics are important for a week but not a year*
- The aggregation can also be turned off after a particular historical time level.
- Below is an example of a customer configuration

<table>
<thead>
<tr>
<th>Interval</th>
<th>Interval Period</th>
<th>Interval Length</th>
<th>Level</th>
<th>Aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per day</td>
<td>5 minutes*</td>
<td>1 day*</td>
<td>4</td>
<td>Yes</td>
</tr>
<tr>
<td>Per week</td>
<td>30 minutes</td>
<td>1 week</td>
<td>3</td>
<td>Yes</td>
</tr>
<tr>
<td>Per month</td>
<td>2 hours</td>
<td>1 month</td>
<td>2</td>
<td>No</td>
</tr>
<tr>
<td>Per year</td>
<td>1 day</td>
<td>1 year*</td>
<td>1</td>
<td>No</td>
</tr>
</tbody>
</table>
Performance Metrics Primer

- The minimum counter granularity to collect statistics is 20 seconds.
- If information is requested from Virtual Center at a frequency of 5 minutes or lower, that request is passed through directly to the host to get accurate real-time data.
- Virtual Center scalability for statistics is significantly improved in VC 2.5

**Partner quote:**

- VC 2.0 could get Level 4 stats for up to 20 hosts in about 5 minutes.
- VC 2.5 can get the same stats for up to 100 hosts (500 powered-on VMs) in 1.5 minutes
Common Customer Questions

Why can’t I use (r)esxtop? How is it different from the counters?

- I get different numbers from the API v/s esxtop in COS
  - Source of data is the same (VMkernel).
  - Sampling frequencies may differ (esxtop: 5s, VirtualCenter 20s)

- Are there other differences between the metrics?
  - esxtop contains some counters that VC does not (e.g. Disk ACTV)
  - The unit of measure on some counters is different (% vs. ms)

- esxtop has better interval granularity. I will use it all the time.
  - esxtop puts a very high load on the server. It should be used for interactive troubleshooting at best.
  - The API counters are optimized for retrieval and aggregation and provide all the data that is necessary to debug problems.
Common Customer Questions

🤔 How can I validate that my virtual environment is better?

- System administrators are often under pressure to virtualize the datacenter to reduce TCO
- But there is always that nagging question: 
  *Am I doing better than or at least as well as before? How is my system performing?*
Virtual Environment v/s Physical Environment

First, define “better” or “as well as”
- Better CPU utilization?
  1 server at 80% v/s 4 servers at 20% each
- Better memory utilization?
  1 server with 4GB RAM v/s 4 servers with 2GB each
- Lower power consumption?
  Fewer physical servers means less power and less cooling
- More scalable performance?
  4 UP VMs with better throughput than a 4-way native server

How can our counters help?
- Counters are currently limited to resource utilization (cpu, memory, disk, network)
- Collect cpu usage %, memory consumed and compare to the physical
- VMware currently does not expose metrics for application performance or power consumed
Common Customer Questions

❓ When do I need to add another host?

- I now have 30 Virtual Machines running on 3 hosts.
- I have to provision another 5 VMs in the next quarter.
- Can I leverage my existing infrastructure or should I be planning on bringing in another host? Have I already maxed out my current CPU capacity?
CPU capacity

- How do we know we are maxed out?
  - If VMs are waiting for CPU time, maybe we need more CPUs.
  - To measure this, look at CPU ready time.

- What exactly am I looking for?
  - For each host, collect ready time for each VM
  - Compute %ready time for each VM (ready time/sampling interval)
  - If average %ready time > 20% over an extended interval, probe further
CPU capacity

Some caveats on ready time

- Used time ~ ready time: may signal contention. However, might not be overcommitted due to workload variability.

- In this example, we have periods of activity and idle periods: CPU isn’t overcommitted all the time.

(screenshot from VI Client)
Further ready time examination

```
2:01:53pm up 4 days 29 min, 87 worlds; CPU load average: 0.16, 0.16, 0.09
```

```
%CPU(%) : 13.20, 15.55, 10.71, 23.06; used total: 15.63
%CPU(%) : 0 us, 0 sy, 99 id, 0 wa; cs/sec: 90
```

<table>
<thead>
<tr>
<th>ID</th>
<th>GID NAME</th>
<th>%USED</th>
<th>%RUN</th>
<th>%SYS</th>
<th>%WAIT</th>
<th>%RDY</th>
<th>%MLMTD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>idle</td>
<td>337.41</td>
<td>338.34</td>
<td>0.00</td>
<td>0.00</td>
<td>61.66</td>
<td>0.00</td>
</tr>
<tr>
<td>2</td>
<td>system</td>
<td>0.02</td>
<td>0.02</td>
<td>0.00</td>
<td>599.97</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>6</td>
<td>helper</td>
<td>0.02</td>
<td>0.02</td>
<td>0.00</td>
<td>2199.95</td>
<td>0.11</td>
<td>0.00</td>
</tr>
<tr>
<td>7</td>
<td>drivers</td>
<td>0.01</td>
<td>0.01</td>
<td>0.00</td>
<td>1099.93</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>9</td>
<td>console</td>
<td>0.92</td>
<td>0.93</td>
<td>0.00</td>
<td>98.27</td>
<td>0.80</td>
<td>0.00</td>
</tr>
<tr>
<td>14</td>
<td>vmkapiMod</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>200.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>15</td>
<td>vmware-vmkauthd</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>16</td>
<td>cpuBurn-CLONE</td>
<td>50.11</td>
<td>50.03</td>
<td>0.00</td>
<td>495.74</td>
<td>154.20</td>
<td>152.44</td>
</tr>
<tr>
<td>17</td>
<td>fakeDB</td>
<td>1.43</td>
<td>1.44</td>
<td>0.00</td>
<td>697.67</td>
<td>0.93</td>
<td>0.00</td>
</tr>
<tr>
<td>18</td>
<td>Windows 2003 SP</td>
<td>5.10</td>
<td>5.12</td>
<td>0.00</td>
<td>693.01</td>
<td>1.37</td>
<td>0.00</td>
</tr>
<tr>
<td>19</td>
<td>SQL2005</td>
<td>1.63</td>
<td>1.59</td>
<td>0.03</td>
<td>697.16</td>
<td>1.24</td>
<td>0.00</td>
</tr>
<tr>
<td>20</td>
<td>memhog-linux-CL</td>
<td>1.17</td>
<td>1.16</td>
<td>0.02</td>
<td>498.29</td>
<td>0.55</td>
<td>0.00</td>
</tr>
<tr>
<td>21</td>
<td>cpuBurn-CLONE2</td>
<td>1.31</td>
<td>1.29</td>
<td>0.03</td>
<td>698.01</td>
<td>0.70</td>
<td>0.00</td>
</tr>
</tbody>
</table>

High Ready Time

High MLMTD: there is a limit on this VM...

→High ready time not always because of overcommitment
Ready time in VI client

- **High ready time**
- **Limit on CPU**
3 Possible reasons for high ready time

Possible causes

- **CPU over-commitment**
- **Workload variability**
  - A bunch of VMs wake up all at once
  - Note: system may be mostly idle: not always overcommitted
- **Reservation set on VM**
  - 4x2GHz host, 2 vcpu VM, limit set to 1GHz (VM can consume 1GHz)
  - Without limit, max is 2GHz. With limit, max is 1GHz (50% of 2GHz)
  - CPU all busy: %USED: 50%; %MLMTD & %RDY = 150% [total is 200%, or 2 CPUs]

Possible solutions

- VMotion the VM or use DRS to optimize resources
- Change share allocations to de-prioritize less important VMs
- Check CPU limit settings
- More CPUs may be the solution
Common Customer Questions

💡 Will adding more memory to my hosts help?

- I now have 30 Virtual Machines running on 3 hosts.
- The CPU utilization seems to be optimal but applications are a bit sluggish.
- Will adding more memory help solve the problem? Can I find that out by analyzing the performance statistics?
Memory capacity

- **How do we identify host memory contention?**
  - Host-level swapping (e.g., robbing VM A to satisfy VM B).
  - Active memory for all VMs > physical memory on host
    This could mean possible memory over-commitment

- **What do I do?**
  - Check `swapin` (cumulative), `swapout` (cumulative) and `swapused` ("instantaneous") for the host. Ballooning (`vmmemctl`) is also useful.
  - If `swapin` and `swapout` are increasing, it means that there is possible memory over-commitment
  - Another possibility: sum up active memory for each VM. See if it exceeds host physical memory.
Memory capacity

*Increased swap activity may be a sign of over-commitment*

- **Increase in swap activity**
- **No swap activity**
- **Swap in**
- **Balloon & target**
- **Swap out**
- **Consumed & granted**
- **Active memory**
- **Swap usage**

---

*vmware technology exchange*
Troubleshooting memory related problems

### Swapping

![Image](image-url)

6:54:20am up 53 days 19:49, 87 worlds; MEM overcommit avg: 0.98, 1.15, 1.51
PMEM /MB: 4095 total: 272 cos, 175 vmk, 1461 other, 2186 free
VMKMEM/MB: 3735 managed: 224 minfree, 976 rsvd, 2648 ursvd, high state
COSMEM/MB: 9 free: 541 swap_t, 541 swap_f: 0.00 r/s, 0.00 w/s
PSHARE/MB: 5338 shared, 184 common: 5154 saving
SWAP /MB: 1295 curr, 677 target: 0.75 r/s, 0.01 w/s
MEMCCTL/MB: 652 curr, 652 target, 4645 max

<table>
<thead>
<tr>
<th>NAME</th>
<th>MEMSZ</th>
<th>S2TGT</th>
<th>MCTL?</th>
<th>MCTLSZ</th>
<th>MCTLTGT</th>
<th>MCTLMAX</th>
<th>SWCUR</th>
<th>SWTGT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows 2003 SP</td>
<td>1024.00</td>
<td>385.53</td>
<td>Y</td>
<td>0.00</td>
<td>0.00</td>
<td>665.60</td>
<td>119.82</td>
<td>0.00</td>
</tr>
<tr>
<td>SQL2005</td>
<td>2048.00</td>
<td>456.73</td>
<td>Y</td>
<td>0.00</td>
<td>0.00</td>
<td>1331.20</td>
<td>215.91</td>
<td>0.00</td>
</tr>
<tr>
<td>vc server</td>
<td>1024.00</td>
<td>284.19</td>
<td>Y</td>
<td>0.00</td>
<td>0.00</td>
<td>665.60</td>
<td>78.65</td>
<td>0.00</td>
</tr>
<tr>
<td>fakeDB</td>
<td>2048.00</td>
<td>483.75</td>
<td>Y</td>
<td>0.00</td>
<td>0.00</td>
<td>1331.20</td>
<td>203.79</td>
<td>0.00</td>
</tr>
<tr>
<td>memhog-linux-sm</td>
<td>1024.00</td>
<td>376.21</td>
<td>N</td>
<td>0.00</td>
<td>0.00</td>
<td>652.21</td>
<td>518.47</td>
<td>620.36</td>
</tr>
<tr>
<td>memhog-linux-CL</td>
<td>1024.00</td>
<td>347.39</td>
<td>Y</td>
<td>652.21</td>
<td>652.21</td>
<td>652.21</td>
<td>55.10</td>
<td>57.07</td>
</tr>
</tbody>
</table>

- **Memory Hog VMs**
- **MCTL: N - Balloon driver not active, tools probably not installed**
- **Ballooning active**
- **Swapped in the past but not actively swapping now**
- **More swapping since balloon driver is not active**
Is the problem with my network or disk configuration?

- I think that my problems are with my network or disk bandwidth.
- Should I consider reconfiguring my network or perhaps it is my storage network.....
Disk and network capacity

- **Identifying network or disk problems**
  - Check bandwidth of each and compare with expectations
  - Check disk latency and compare with expectations

- **What do I do?**
  - Check requests per sampling interval and bytes transferred/received per sampling interval
  - For disks, check latencies
  - Compare with specs for the network or disk subsystems
SAN Performance Rough Estimation

- From the perspective of a single VMware ESX, roughly:
  Throughput (in MBps) = (Outstanding IOs * Block size in KB) / latency in msec
  
  Effective Link Bandwidth = ~80% of Real Bandwidth
  Effective (2Gbps) = 200 MBps
  Effective (4Gbps) = 400 MBps

- In a clustered Fiber-channel environment:
  Throughput per host = (Effective Link Bandwidth / No. of IO intensive hosts)

- To achieve the effective link bandwidth:
  Latency in msec <= (Outstanding IOs * Block size in KB) / Throughput per host

Source: VMworld '07: IP42 “ESX Storage Performance – A Scalability Study”
Desired Latency Per Host

- Desired Latency in msec ≤ (Outstanding IOs * Block size in KB) / Throughput per host

Example:
- Number of Hosts = 64
- Effective link bandwidth = 400 MBps
- Throughput per host = 400 / 64 = 6.25 MBps
- Desired latency = (32 * 32) / (6.25) = 163.84 msec

<table>
<thead>
<tr>
<th>Workload</th>
<th>Cached Sequential Read</th>
<th>Cached Sequential Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desired latency (msec)</td>
<td>163.84</td>
<td>163.84</td>
</tr>
<tr>
<td>Observed latency (msec)</td>
<td>~310</td>
<td>~163</td>
</tr>
<tr>
<td>Throughput drop ?</td>
<td>YES</td>
<td>NO</td>
</tr>
<tr>
<td>Throughput (MBps)</td>
<td>~270</td>
<td>~400</td>
</tr>
</tbody>
</table>
Disk throughput

SAN cache enabled: High throughput

SAN cache disabled: Poor throughput
Disk capacity – Looking at Disk latency

After enabling the SAN cache, latency is much better.
Common Customer Questions

🤔 So many counters, so little time

- You said earlier that VMware exposes 150 counters
- Well, which ones do I care about?
- Which ones make sense to look at daily? Which ones will give me interesting trends that I should consider?
- Do I care about the rest?
Counters of interest

- **If you are looking at real-time statistics ....**
  - **CPU:** usage (% or MHz), used time, ready time, wait time
  - **Memory:** consumed, active, swapused, swapin, swapout, vmmemctl
  - **Disk:** diskReadLatency, diskWriteLatency, commands, commandsAborted, bytes transferred/received, disk bus resets
  - **Network:** packets transmitted/received

- **Dig deeper if you see issues**
  - For example, on disks
    - deviceLatency, kernelLatency, queueLatency, totalLatency
    - Disk bus resets may signal failing LUNs.
## Counters of interest

<table>
<thead>
<tr>
<th>Counter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu.usage.average</td>
<td>CPU usage (%)</td>
</tr>
<tr>
<td>cpu.used.summation</td>
<td>Used time (ms)</td>
</tr>
<tr>
<td>cpu.ready.summary</td>
<td>Ready to run, no resources available (ms)</td>
</tr>
<tr>
<td>cpu.wait.summation</td>
<td>Blocked waiting (e.g., for I/O) (ms)</td>
</tr>
<tr>
<td>mem.consumed.average</td>
<td>Machine pages taken by VM</td>
</tr>
<tr>
<td>mem.active.average</td>
<td>“Working set” of VM</td>
</tr>
<tr>
<td>mem.swapused.average</td>
<td>“Instantaneous” swapped memory for VM</td>
</tr>
<tr>
<td>mem.swapin.average</td>
<td>Cumulative swapped-in memory for VM</td>
</tr>
<tr>
<td>mem.swapout.average</td>
<td>Cumulative swapped-out memory for VM</td>
</tr>
<tr>
<td>mem.vmmemctl.average</td>
<td>Ballooned memory for VM</td>
</tr>
</tbody>
</table>
## Counters of interest

<table>
<thead>
<tr>
<th>Counter Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disk.commands.summation</td>
<td>Disk commands issued</td>
</tr>
<tr>
<td>disk.usage.average</td>
<td>Disk Bandwidth consumed</td>
</tr>
<tr>
<td>disk.commandsAborted.summation</td>
<td>Disk commands aborted</td>
</tr>
<tr>
<td>disk.busResets.summation</td>
<td>SCSI bus resets</td>
</tr>
<tr>
<td>disk.deviceLatency.average</td>
<td>Latency at the device</td>
</tr>
<tr>
<td>disk.kernelLatency.average</td>
<td>Latency within the vmkernel</td>
</tr>
<tr>
<td>net.usage.average</td>
<td>Network bandwidth consumed</td>
</tr>
<tr>
<td>net.packetsRx.summation</td>
<td>Packets received in sample interval</td>
</tr>
<tr>
<td>net.packetsTx.summation</td>
<td>Packets transmitted in sample interval</td>
</tr>
</tbody>
</table>
Tips and Tricks

- Use view API to monitor inventory
- Use CSV format
- Go multi-threaded
- Statically specify metrics to collect
- Query over small time increments
- Choose correct stats levels
- Historical vs. real-time retrieval (To DB or not to DB)
- Watch your serialization and DB costs
- Optimize your metric gathering code
Tips and Tricks: Serialization and Database costs

▶ How much data are we sending?

- 4-way host, 2 NICs, 1 datastore
  QueryAvailablePerfMetrics → 173 metrics!

- 2-way VM, 1 NIC, 1 datastore
  QueryAvailablePerfMetrics → 99 metrics!

- Assume 4 chars per metric
  ~700B per host, ~400B per VM

- Assume 100 hosts, 1000 VMs
  ~460KB to get 1 data point

- For 12 data points (1 hour of 5-minute stats): 5.4MB
  Things add up, don’t they

- 5.4MB → serialization cost becomes significant
Tips and Tricks: Serialization and Database costs

Sample latency breakdown for a subset of stats

- Single query for a 24 hours of data from a host
- Total query: 1.75s
  - SSL handshake 180ms (~ fixed latency)
  - Server deserialization/transfer: 500ms (scales with # of points selected)
  - DB access 270ms (scales with dataset)
  - call to DB 100ms (~ fixed latency)
  - client deserialization/transfer: 600ms (scales with # of points selected)

Bottom line:
- serialization is important: pick metrics wisely
- As DB grows, its latency becomes significant

(Tools used: wireshark, SQL profiler, logging in SDK code)
Tips and Tricks: Query VC v/s Querying each host

<table>
<thead>
<tr>
<th>Threads</th>
<th>Query through VC(s)</th>
<th>Query directly to host(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>251</td>
<td>242</td>
</tr>
<tr>
<td>2</td>
<td>131</td>
<td>153</td>
</tr>
<tr>
<td>4</td>
<td>81</td>
<td>77</td>
</tr>
<tr>
<td>6</td>
<td>60</td>
<td>70</td>
</tr>
<tr>
<td>8</td>
<td>52</td>
<td>48</td>
</tr>
</tbody>
</table>

- 64 hosts, 1233 powered-on VMs, real-time stats, VIPerl toolkit used
- Querying through VC can be ~ Querying through hosts
  (inventory monitoring easier with VC, though...consider views)
- Different client implementations may yield different results (# threads?)
Tips and Tricks: Writing efficient code

Code that will not scale

```java
pqArray = new PerfQuerySpec[];
for (i = 0; i < 1000; i++) {
    PerfQuerySpec pq = new PerfQuerySpec( ... );
    pqArray[0] = pq;
    PerfEntityMetricBase[ ] pemb = service.queryPerf(perfManager, pqArray);
}
```

*One element array*
Tips and Tricks: Writing efficient code

Code that does it right

```java
pqspecArray = new PerfQuerySpec[];
for (i = 0; i < 1000; i++)
{
    PerfQuerySpec pqs = new PerfQuerySpec( ... );
    pqsArray[i] = pqs;
}
PerfEntityMetricBase[ ] pemb =
    service.queryPerf(perfManager, pqsArray);
```

Remember:
- Collect only what you will use
- Use everything that you collect