Performance Modeling in Virtualized Environment

Lukáš Marek
Department of Distributed and Dependable Systems
http://d3s.mff.cuni.cz

CHARLES UNIVERSITY PRAGUE
Faculty of Mathematics and Physics
Reading seminar

• Omesh Tickoo, Ravi Iyer, Ramesh Illikkal, Don Newell: **Modeling Virtual Machine Performance: Challenges and Approaches**
  - Intel Labs, Intel Corporation

• Ajay Gulati, Chethan Kumar, Irfan Ahmad: **Modeling Workloads and Devices for IO Load Balancing in Virtualized Environments**
  - VMware Inc.

• ACM SIGMETRICS: Performance Evaluation Review - Volume 37 Issue 3

• All pictures (and some sentences) used in this presentation are taken from papers mentioned above.
Modeling Virtual Machine Performance: Challenges and Approaches

- Performance in virtualized environment

(a) From Dedicated Workloads to Consolidated Workloads
What influences performance?

• Interference with other virtual machines
• Visible (observable)
  ▪ CPU
  ▪ Memory capacity
• Invisible
  ▪ cache space
  ▪ memory bandwidth
  ▪ ...

Goal of the paper

- Using virtualization/consolidation benchmark (vConsolidate) show the performance dependency on resources
- Offline/Online monitoring (modeling) technique to characterize VMs behavior
- Modeling approach for estimating VMs behavior
Measurement platform

- Intel server
  - 16GB memory
  - Two processors
  - 4 cores each
  - Each pair share 4MB cache
- Xen
Benchmarks

- **vConsolidate**
  - VM benchmark
  - Consists of 5 VMs
    - WebBench (Web) - IIS
    - LoadSim (mail) - Exchange
    - SysBench (database) - SQL Server
    - SPECjbb2005 (Java) - BEA JVM
      - Modified – uses ~75% CPU
    - Idle - NA
First measurement - revealing the interference

(c) SpecJBB Performance (1 copy vs 2 copies)
Revealing the interference II

(d) SpecJBB Performance (Alone vs with Sysbench)
VM Performance Modeling Approaches

- Offline modeling
  - workload measured on several platforms alone as well as pairwise with other virtual machines
  - run through core and cache simulators to collect behavioral information
- Online modeling
  - no offline modeling possible
  - gather data and model the application online
The proposed approach

- Specification of key resources for each VM
  - number of cores
  - the core frequency and utilization
  - the cache space at each level
  - the memory frequency and bandwidth
- Take VM as a normal application and estimate its performance in a shared environment
Core Utilization Estimation

• How much CPU time our application consumes in shared environment?
• Monitor a VM’s core utilization when running alone (VMx-Alone-Util)
• Predict a VM’s core utilization on a different consolidated platform (Vmx-Cons-Util)

\[
VM_{x-Const-Util} = \min \left[ VM_{x-Alone-Util}, \frac{VM_{x-Alone-Util} \times PhysicalCPUs}{VM_{all-Alone-Util}} \right]
\]
Estimated results

- specJBB within vConsolidate
- 3 different architectures with different cache sizes
- How was the cache size limited?
- What is the utilization of the whole server?
- What can be the biggest error in estimation?
- Depends on scheduler

### Table 1: SPECjbb VPA Core Utilization

<table>
<thead>
<tr>
<th>specJBB</th>
<th>4MB</th>
<th>2MB</th>
<th>1MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated</td>
<td>115%</td>
<td>124%</td>
<td>126%</td>
</tr>
<tr>
<td>Measured</td>
<td>122%</td>
<td>124%</td>
<td>121%</td>
</tr>
<tr>
<td>Error (%)</td>
<td>-5.58</td>
<td>0.39</td>
<td>3.91</td>
</tr>
</tbody>
</table>
Cache Space Estimation

• How much of the cache will be occupied by the application
  ▪ i.e. How much will increase the cache miss rate when more applications are running together in a same cache.
Cache Space Estimation – step 1

- Profile the execution of a VM by capturing the percentage of time it runs with other VMs on sibling cores (sharing the same cache)
- The probability that VM\textsubscript{y} runs on second core

\[
P(VM_x + VM_y) = \frac{VM_y\text{Util}}{(VM_{all}\text{Cons-Util} - VM_x\text{Cons-Util})}
\]
Estimated results – for step 1

- Measured results +- 2% same

Table 2: Execution Probability of VMs running cores sharing a L2 Cache with SPECjbb

<table>
<thead>
<tr>
<th></th>
<th>specJBB</th>
<th>sysbench</th>
<th>webbench</th>
<th>MMB</th>
<th>Domain-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>4MB</td>
<td>18%</td>
<td>38%</td>
<td>32%</td>
<td>3%</td>
<td>9%</td>
</tr>
<tr>
<td>2MB</td>
<td>37%</td>
<td>19%</td>
<td>31%</td>
<td>3%</td>
<td>10%</td>
</tr>
<tr>
<td>1MB</td>
<td>36%</td>
<td>17%</td>
<td>32%</td>
<td>4%</td>
<td>12%</td>
</tr>
</tbody>
</table>
Cache Space Estimation – step 2

• Estimate the cache space for a VM when it runs with another sibling VM (assuming two CPUs sharing a cache)
   Offline: capture a trace for each VM and then run every pair of traces (for different VMs) through a shared cache model
   Online: not possible with current hardware
Estimated results – for step 2

- Misses per instruction increase

Figure 2: MPI Increase in Pairwise/Consolidated Execution
Cache Space Estimation – step 3

• Put step 1 and step 2 together and create weighted average of these two steps
• Table 4 shows that the estimated CPI values match the measured values with low error (~5%).

Table 3: Overall Cache Contention Effect for Consolidation

<table>
<thead>
<tr>
<th></th>
<th>Est. MPI</th>
<th>Measur. MPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>specJBB</td>
<td>0.0077</td>
<td>0.0085</td>
</tr>
<tr>
<td>sysbench</td>
<td>0.0041</td>
<td>0.0043</td>
</tr>
</tbody>
</table>

Table 4: CPI Estimation for SPECjbb

<table>
<thead>
<tr>
<th></th>
<th>SPECjbb Single VM</th>
<th>specJBB in vCon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated CPI</td>
<td>1.549</td>
<td>2.231</td>
</tr>
<tr>
<td>Measured CPI</td>
<td>1.543</td>
<td>2.234</td>
</tr>
</tbody>
</table>
Modeling Workloads and Devices for IO Load Balancing in Virtualized Environments

- Load balancing VM stores between data storage arrays according to utilization
- VMware (Storage VMotion) supports transparent VM storage migration, without VM stopping

Figure 1: Data migration using Storage VMotion
Goal of the paper

• Workload modeling
  ▪ Assign “L” value to a workload
    • I/O load metric

• Storage device modeling
  ▪ Assign “P” value to a storage device
    • I/O performance metric

• Creating load balance engine
  ▪ Use P and L for load balancing
Workload Modeling

• Storage workload characterization
  ▪ Outstanding IOs
  ▪ IO size
  ▪ read-write ratio
  ▪ average number of outstanding Ios

• Goal
  ▪ Show on simple benchmarks, that for all mentioned characteristics the storage workload overhead varies linearly in respect to IO latency
Benchmarks and tools

- Iometer inside a Microsoft Windows 2003 VM accessing a 4 disk RAID-0
  - Outstanding IOs \{4, 8, 16, 32, 64\}
  - IO size (in KB) \{8, 16, 32, 128, 256, 512\}
  - Read\% \{0, 25, 50, 75, 100\}
  - Random\% \{0, 25, 50, 75, 100\}
  - Variation of all parameters
  - obtain the values of average IO latency and IOPS
- vsccsiStats
  - Allows collection of I/O data mentioned above
• Show on simple benchmarks, that for all mentioned characteristics the storage workload overhead varies linearly
  ▪ (with exceptions)
Outstanding IOs

![Graph showing the relationship between Outstanding IOs and Average IO Latency](image-url)
IO Size

![Graph showing the relationship between IO size and average IO latency for different scenarios. The graph includes lines for scenarios with 8, 16, 32, and 64 IOs, with read and randomness percentages ranging from 25% to 100%.

- 8 IOs, 25% Read, 25% Randomness
- 16 IOs, 50% Read, 50% Randomness
- 32 IOs, 75% Read, 75% Randomness
- 64 IOs, 100% Read, 100% Randomness]
% Read

Average IO Latency (in ms)

8 OIO, 32K, 25% Randomness
16 OIO, 32K, 50% Randomness
32 OIO, 16K, 75% Randomness
64 OIO, 8K, 100% Randomness

% Read

(c)
% Randomness

(d)
Modeling equation

- K1 – K4 are computed values for the purpose of modeling
  - Same for all workloads
- K5 variable depends on storage device parameters
- The characteristic number “L” for workload is numerator of the equation

Based on these observations, we modeled the IO latency (L) of a workload using the following equation:

\[
L = \frac{(K_1 + OIO)(K_2 + IOsize)(K_3 + \frac{\text{read}\%}{100})(K_4 + \frac{\text{random}\%}{100})}{K_5}
\]

(1)
Can we use a fixed set of constants $K_1$, $K_2$, $K_3$ and $K_4$ for all workloads?

How well the modeling works for dynamic workloads where the workload characteristics change over time?
Storage device modeling

• Features such as RPM, average seek delay, etc. are hidden
• IO latency as the main performance metric
• Collect information pairs consisting of number of outstanding IOs and average IO latency observed
  ▪ Again, show a linearity variation
Outstanding IOs

![Graph showing the relationship between Outstanding IOs and Average IO Latency](image)

- 4 disks
- 8 disks
- 12 disks
- 16 disks
Number of disks and latency

- With twice as many disks, you will get half of the latency time
- RAID group is further divided into logical units of storage (LUNs)
Slope and “P” value

- Performance metric “P” = 1/Slope

Figure 8: Device Modeling: different disk types
• $\sigma(N)$ is the variance of the normalized load
• perfect balancing between D1 and D2 is a variant of subset-sum problem which is known to be NP-complete
  ▪ approximation

Algorithm 1: Load Balancing Step

```
foreach device j do
    foreach workload i currently placed on device j do
        $S_+ = L_i$
        $N_j \leftarrow S / \mathcal{P}_j$
    while $\sigma(N) < varianceThreshold$ do
        $D_1 \leftarrow$ Device with maximum normalized load
        $D_2 \leftarrow$ Device with minimum normalized load
        $N_1, N_2 \leftarrow$ PairWiseRecommendMigration($D_1, D_2$)
```
### Evaluation I

<table>
<thead>
<tr>
<th>Workers</th>
<th>OIO</th>
<th>IOSize (KB)</th>
<th>Random %</th>
<th>Read %</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>16</td>
<td>4</td>
<td>100</td>
<td>75</td>
</tr>
<tr>
<td>W2</td>
<td>8</td>
<td>64</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>W3</td>
<td>2</td>
<td>128</td>
<td>90</td>
<td>50</td>
</tr>
<tr>
<td>W4</td>
<td>4</td>
<td>32</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>W5</td>
<td>16</td>
<td>32</td>
<td>100</td>
<td>75</td>
</tr>
</tbody>
</table>

Table 1: Details of Test Workloads.
Evaluation II

<table>
<thead>
<tr>
<th>Workers</th>
<th>Latency (ms)</th>
<th>Throughput (IOPs)</th>
<th>Location</th>
<th>Latency (ms)</th>
<th>Throughput (IOPs)</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>19</td>
<td>800</td>
<td>D_1</td>
<td>17</td>
<td>881</td>
<td>D_2</td>
</tr>
<tr>
<td>W2</td>
<td>13</td>
<td>650</td>
<td>D_2</td>
<td>18</td>
<td>390</td>
<td>D_2</td>
</tr>
<tr>
<td>W3</td>
<td>18</td>
<td>100</td>
<td>D_2</td>
<td>26</td>
<td>70</td>
<td>D_2</td>
</tr>
<tr>
<td>W4</td>
<td>1</td>
<td>4000</td>
<td>D_2</td>
<td>1.7</td>
<td>3600</td>
<td>D_2</td>
</tr>
<tr>
<td>W5</td>
<td>20</td>
<td>760</td>
<td>D_1</td>
<td>13</td>
<td>1200</td>
<td>D_1</td>
</tr>
</tbody>
</table>

Table 2: Workload Migration Recommendation by SRS. Average latency and IO Throughput for workloads W1 through W2 before and after migration.

<table>
<thead>
<tr>
<th>Datastores</th>
<th># Disks</th>
<th>K_5</th>
<th>Latency</th>
<th>IOPs</th>
<th>Total Load (L)</th>
<th>Latency</th>
<th>IOPs</th>
<th>Total Load (L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_1</td>
<td>5</td>
<td>1</td>
<td>19</td>
<td>1560</td>
<td>10472</td>
<td>13</td>
<td>1200</td>
<td>6314</td>
</tr>
<tr>
<td>D_2</td>
<td>10</td>
<td>2</td>
<td>3</td>
<td>4750</td>
<td>3854</td>
<td>6.5</td>
<td>4800</td>
<td>5933</td>
</tr>
</tbody>
</table>

Table 3: Results of Workload Migration Recommendation by SRS. Latency, IOPS and overall load on Datastores D_1 and D_2.
Questions?