A Scientific Review on Virtual Machine Performance

Chris Weber\textsuperscript{1} (100216595), Azhar Saiyed\textsuperscript{2} (100464089), Maaz Kamani\textsuperscript{3} (100453151), Pirasanth Sivalingam\textsuperscript{4} (100460273)  
Faculty of Business and IT  
University of Ontario Institute of Technology  
Oshawa, Canada

\textsuperscript{1}Christopher.weber@uoit.net, \textsuperscript{2}Azhar.saiyed@uoit.net, \textsuperscript{3}Maaz.kamani@uoit.net, \textsuperscript{4}Pirasanth.sivalingam@uoit.net

Abstract — Virtualization has become such a hot topic in information technology lately. Many businesses are consolidating their resources into single virtualized systems which saves them money. However this cost savings comes a decrease in performance. This paper provides a scientific review of performance studies completed on virtualized environments. We first analyze the results and provide a survey of performance studies completed comparing benchmark results of various hypervisor's such as VMware, ESX, KVM, VirtualPC and Qemu. We then discuss the performance issues that the host operating system has on virtual environments and based on these results provide a recommendation for the best host operating system to use when running virtual machines.

Keywords - performance; virtualization; hypervisor; XEN; comparison; VMware; VirtualBox; QEMU; KVM

I. INTRODUCTION

One of the hottest topics these days in the area of information technology is that of virtualization. Technology has advanced to the point where we have begun to virtualize all aspects of our information systems from servers to personal computers to networks, hard drives and I/O devices. However the issue that arises is that of performance and at what levels these virtualized systems can perform in comparison to their physical counterparts.

The field of performance relating to virtualization is a vast one that spans many different subject areas and technologies. For that reason we have chosen to focus this review on the performance of operating systems in virtualized environments.

For this study, we address several previous works conducted which measure performance levels of various hypervisors and the inherent performance issues involved as they relate to both the host operating system and the guest operating systems involved.

The rest of this paper is organized as follows: Section 2 provides background information for some of the terms related to virtualization that are discussed in this paper. Section 3 presents a survey of several benchmark works completed and the performance results obtained from these experiments. Section 4 discusses the impacts that virtualization has on the host operating system. Finally Section 5 provides some conclusions and future works to be studied in the area of virtual machine performance.

II. BACKGROUND

A hypervisor, also known as a virtual machine monitor (VMM) is the program or system that runs on an operating system and provides the virtualized environment from which virtual machines or virtual operating systems are run. Examples of hypervisors include VMware, VirtualBox, XEN, KVM, QEMU and VirtualPC.

A host operating system is the one installed physically on the machine. It is the one on which the hypervisor is installed and run from. The guest operating system is the virtual operating system which is run from the hypervisor. Therefore the layers within a virtualized system involve the hardware and system resources at the lowest layer. Then the host operating system followed by the hypervisor and finally the guest operating system is the highest layer in this virtualized hierarchy [1].

III. HYPervisor BENCHMARK RESULTS

With regard to hypervisor performance, there exist several comparison studies [2] [3] [4] [5]. We address these works in the following sections and the results that were obtained.

In the first study [2], the authors measure and analyze the performance of two open source virtual machine monitors or hypervisors; XEN and KVM using benchmarks; Linpack, LMbench and IOzone. The study provided a quantitative and qualitative comparison of both the open source virtual machine monitors.

KVM is an example of Full-Virtualization technology which provides an emulation of the
An interface to guest OS’s which don’t realize they are living in a virtual machine environment and requires no change to these guest OS [6] while XEN is an example of Para-Visualization technology which provides a virtual machine abstraction that is similar but not identical to underlying hardware and thus requires some modifications for porting into virtual machines [7].

In the first test of this study the authors measured the overhead of CPU virtualization using Linpack 10.0.2 and memory virtualization using LMbench 2.5. The performance disparities of virtualized environment were compared against the native environment. All the experiments were based on native Fedora core 8 based Linux 2.6.24.4 (Fedora8) as the host OS and XEN 3.1 and KVM-60 as the virtual machine monitors. The authors ran the Linpack test and compared the float-point computing power of virtualized Fedora8 in XEN and KVM against native Fedora8. Virtualized Fedora8 against virtualized Windows XP in XEN and KVM. Figure 1 below summarizes these Linpack results.

These Linpack results shows that the maximal (GFLOPS) of virtualized Fedora8 in KVM is about 83% of that of native Fedora8 and in XEN it’s about 97% of native. This shows that the processing power of KVM on float-point computing is not as good as XEN because KVM needs to check whether every executing instruction is an interrupt, page fault, I/O or common instruction to decide exiting from or staying in guest mode. This wastes a lot of time in KVM, while in XEN it allows the guest OS to execute the float-point computing instructions on physical CPU to save the solving time. XEN is a relatively more mature virtual machine monitor compared to KVM and thus overhead of CPU virtualization is negligible. In addition the results for virtualized Windows XP are better than virtualized Fedora8 in XEN but not in KVM. The reason for this is that XEN has a few enhancement packages for Windows as a guest OS and so virtualized Windows XP has a better performance than virtualized Fedora8 in XEN.

To measure the overhead of memory virtualization the authors benchmarked virtualized Fedora8 in XEN and KVM against native Fedora8 using LMbench tests. They focused on the latency of system operations and context switch and the bandwidth of memory reading and writing.

From the results in Table 1 it can be seen that XEN and KVM display slower fork, exec, sh and pipe performance compared to native Fedora8 while KVM shows worse results than XEN. The process creation and execution need to update the hardware page table so these operations will always trap into and validated by XEN or rebuild the shadow page tables and map guest virtual to host physical memory in KVM to cause awful performance of both.

Table II shows context switch times between different numbers of processes with different working set sizes. XEN shows a faster context switch than KVM except when the working set size is 16k with more than two processes. This is related mostly to cache effects and context switch overhead for larger working set sizes.
The results of memory reading in 3 environments are similar except of KVM which shows a big reduction in bandwidth of memory writing shown in Figure 2. KVM-60 lacks the support of extended page tables (EPT) and optimizations to MMU virtualization and employs a method of memory virtualization resembling full-virtualization, which needs to pre-scan the code that will be executed to replace those sensitive privilege instructions. This is the reason for low performance of KVM. XEN uses Para-virtualization and guest OS’s realize themselves being virtualized and batch update requests to reduce the overhead when entering the hypervisor, so XEN can obtain almost the same performance as native Fedora8.

The read and write bandwidth results of IOzone benchmarks are shown in Figure 3 and 4 respectively. In the read bandwidth test, virtualized Fedora8 in XEN with VT-d showed significantly greater performance than just the native Fedora8. The same metrics in XEN and KVM without VT-d results in 6-7 times less performance compared with native Fedora8. The results for the write bandwidth test were similar to the read test but with smaller numbers. As the VT-d embeds the remapping hardware of DMA and IRQ in the North Bridge chipset, the guest OS can directly control I/O devices and avoid trapping into the hypervisor for each I/O request. With VT-d opened, KVM doesn’t make any advance in I/O bandwidth because it achieves virtual I/O’s processing with QEMU which simulates the I/O operations and ignores the hardware advantages. This shows the importance of hardware support with software virtualization.

The authors of [3] conducted tests to compare two of the more popular hypervisor options – VMware and VirtualBox. Their experiment tested the processing speed and the network response time of Ubuntu and Windows XP in both VMM environments. The tests measured the floating point operational performance using LINPACK [8] and the network bandwidth performance using Iperf [9]. The results of both tests are shown in Fig. 5 and Table III.

Fig. 5 and Table III show that Ubuntu on VMware has the best performance in both tests. Of more significance though is that the network performance of XP on VirtualBox is significantly faster than XP on VMware. One possible reason for this is due to the implementation of the network support modules in VirtualBox being better refined than VMware. As we will see in [4], KVM network performance severely lags behind VMware and XEN due to the more primitive nature of the KVM network modules [10]. This could be the same in VMware compared to VirtualBox as VMware may not have refined their network virtualization modules very well yet.

In the third study [4], the authors carried out quantitative performance analysis of the three leading hypervisors XEN, Kernel-based Virtual Machine (KVM) and VMware. The benchmarking test measured Overall Performance, Performance Isolation and Scalability of the virtual machines which were running on the above mentioned hypervisors.

The first test conducted in this study was on the application. The result of the test is shown in Table IV. It shows that in the CPU test XEN had the best performance which was almost identical to native Linux. KVM and VMware were close with 94% and
Table IV. Summary of Application Test Results [4]

<table>
<thead>
<tr>
<th></th>
<th>Linux</th>
<th>XEN</th>
<th>KVM</th>
<th>VMware</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1.00</td>
<td>0.99</td>
<td>0.94</td>
<td>0.96</td>
</tr>
<tr>
<td>Memory</td>
<td>1.00</td>
<td>0.71</td>
<td>0.51</td>
<td>0.68</td>
</tr>
<tr>
<td>Fork</td>
<td>1.00</td>
<td>1.13</td>
<td>3.38</td>
<td>1.08</td>
</tr>
<tr>
<td>Gzip compression</td>
<td>1.00</td>
<td>2.38</td>
<td>1.59</td>
<td>1.81</td>
</tr>
<tr>
<td>LAME Compilation</td>
<td>1.00</td>
<td>2.18</td>
<td>0.91</td>
<td>3.91</td>
</tr>
<tr>
<td>LAME Encoding</td>
<td>1.00</td>
<td>1.02</td>
<td>1.11</td>
<td>1.10</td>
</tr>
</tbody>
</table>

96% performance compared to native Linux respectively. XEN did the best again in the Memory test with 71% performance, while KVM had the worst performance with only about half of that of native Linux. For the Fork Test the authors ran a program that loops endlessly, creating new child processes. KVM performed the worst again, consuming more than three times the resources as native Linux, while VMware consumed almost the same amount of resources as native Linux. In the Gzip test the authors compressed a 1 GB file in which KVM performed the best while XEN performed the worst. In the LAME Compilation XEN performed best while using double the resources and KVM and VMware used almost 7 times and 4 times more resources respectively. For the last LAME encoding test, XEN performed almost identical to native Linux. KVM and VMware performed almost identical with 111% and 110% respectively.

The second test carried out was the disk intensive test using IOzone Benchmark [11] which generates and measures a variety of file operations including read and write. According to results in Figure 6 and 7 below, XEN had the worst read and write performance, while KVM and VMware had higher performance. The reason for better performance by KVM and VMware may be disk caching.

The third test conducted was the network I/O intensive test using Netperf benchmark and the results are documented in Figure 8 and 9. The results show that KVM lagged significantly behind XEN and VMware. The reason for this is that XEN and VMware have been out for a long time and have a long history of development. KVM is relatively new and its network modules are still in development so it hasn’t joined Intel VT-d technical support [12].

The authors performed a performance isolation comparison of the 3 hypervisors and ran several tests such as a CPU Intensive Test, Memory Bomb test, Fork Bomb test, Disk Intensive test, and Server Transmit and Receive Data test. The results of these tests are shown in Table V.

The first test carried out by authors under the Performance Isolation Comparison was CPU Intensive Test where the CPU usage was stressed with a tight loop containing both integer and floating point operations. All the 3 hypervisor performed well on the test. The CPU load on all three virtual machine never reached to 100%.

The authors conducted Memory Bomb test which loops constantly and touching memory. VMware reported significant degradation of 12.8% while KVM and XEN misbehaved but still continued to report 100% good response.
Table V. Summary of stress test results [4]

<table>
<thead>
<tr>
<th></th>
<th>XEN</th>
<th>KVM</th>
<th>VMware</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>Stress</td>
<td>Normal</td>
</tr>
<tr>
<td>CPU</td>
<td>0</td>
<td>0.02</td>
<td>0</td>
</tr>
<tr>
<td>MEMORY</td>
<td>0</td>
<td>DNR</td>
<td>0</td>
</tr>
<tr>
<td>FORK</td>
<td>0.02</td>
<td>DNR</td>
<td>0</td>
</tr>
<tr>
<td>Disk Intensive</td>
<td>2.61</td>
<td>15.67</td>
<td>0.01</td>
</tr>
<tr>
<td>NETWORK Sender</td>
<td>0.02</td>
<td>0.43</td>
<td>0.14</td>
</tr>
<tr>
<td>NETWORK Receiver</td>
<td>0.03</td>
<td>0.12</td>
<td>0</td>
</tr>
</tbody>
</table>

The authors also ran a Fork Bomb test where they ran a program that loops, creating new child process. All 3 virtual machines misbehaved and produced no results.

For the Disk Intensive Test, IOzone was used again and ran threads of it each running an alternating read and write pattern. The results were quite amazing. The misbehaving XEN showed degradation of 15% and the well behaved XEN shows 2.6% degradation. Both KVM and VMware showed no degradation under normal conditions but under stress they showed 26% and 40% degradation. The reason proposed for XEN performing well was its hardware access model, a specialized device driver VM that could be written to ensure quality of service guarantee.

The results for Server Transmitting test were mixed. For this test the authors started 4 threads and each constantly sent 60K sized packets over UDP to external receiver. Both KVM and XEN performed great with less than 1% degradation under normal and misbehaving condition while VMware showed a substantial degradation of 48% under the misbehaving conditions.

For the Server receiving test 4 threads were started and each one constantly read 60K packets over UDP from an external receiver. Both KVM and XEN performed great with 100% good response while on XEN there was very little degradation. It was suspected that the reason for minute degradation in XEN may be the result of the network connection dropping the incoming packets.

The last test done in [4] was a virtual machine scalability test which determines the hypervisors ability to run more virtual machines without loss of performance. In this test the authors ran the Sysbench benchmark software and tried to scale a single guest OS, running on top of the three hypervisors; XEN, KVM and VMware. The results show that XEN had an advantage when the number of VM’s is increased. Figure 10 show the average throughput per VM while Figure 11 shows the total throughput. KVM did the worst but the difference between KVM and others becomes smaller and smaller when the number of VMs are increased. VMware reached its peak at 4 VMs and the order of performance from higher to lower was XEN, VMware and KVM. According to the results XEN and KVM had excellent scalability.

Possibly the most interesting performance benchmark study conducted is [5]. The authors pit what are known as thin hypervisors against thick hypervisors and compare the performance results against bare metal environments to see how the virtual systems perform.

A bare metal environment (Fig. 12.A) is one in which the host operating system runs directly on the hardware which is the case in non-virtualized environments and is the way personal computers run. A thick hypervisor (Fig. 12.C) refers to a software interface that lies between the host operating system and the virtual (guest) operating system. This is the most common form of virtualization where the VM’s are run from software such as VMware, VirtualBox or another VMM, which in turn lies on the operating system which is run on the hardware. The third environment discussed in this study is thin hypervisors. A thin hypervisor (Fig. 12.B) is one in which the VMM software is run directly on the hardware rather than on top of another operating system.

![Figure 10. Sysbench: Average throughput [4]](image1)

![Figure 11. Sysbench: Total throughput [4]](image2)
The authors conducted this test by comparing the thin hypervisors: XEN, ESXi and KVM against the thick hypervisors of VMware Player and VirtualBox. They conducted tests to measure the performance of each type of virtualization in terms of:

- Local memory bandwidth
- Local disk bandwidth
- Network disk bandwidth
- Network web interface bandwidth
- CPU integer processing
- CPU floating point processing

From the diagram in Fig. 12, we would expect that bare metal (a) would be the fastest system since the operating system is running on the hardware itself. We would then expect that thin hypervisors (b) would be the next fastest since the hypervisor itself runs directly on the hardware and we would expect thick hypervisors to be the worst performer because there is an added layer between the hardware and the hypervisor, which should theoretically slow performance. However, the results shown in Table VI challenge these beliefs.

In Table VI, we see that in every benchmark test performed, the thick hypervisors (VMware and VirtualBox) outperformed the thin VMM's. Of equal importance is that in 7/9 test conducted, VirtualBox outperformed VMware. This was equally surprising since VMware has long since been the industry standard in virtualization and has been around longer than VirtualBox, which would lend itself to being more robust.

The most surprising results of all, however, come when we compare VirtualBox to the bare metal system. For the purpose of this discussion, it is important to reiterate that the bare metal system is a real, physical system that is running an operating system in a non-virtualized environment.

When we compare the two, we see that in four of the tests, VirtualBox was able to outperform the bare metal system. In every write operation conducted (RAM, local disk and network disk), we see VirtualBox perform at 112-192% of the bare metal system.

The reason for this, however, makes the results less surprising. A virtual system may employ newer or better buffering algorithms for communicating with I/O interfaces. This explains why VirtualBox was able to perform better than the bare system [5].

A more challenging result to comprehend and explain is that VirtualBox also outperformed the bare metal system in the Dhrystone2 test. Dhrystone2 tested the CPU performance and the number of integer operations/second that could be performed. It is odd that VirtualBox won here because the bare system has direct access to the CPU whereas VirtualBox interfaces with the CPU through two levels of software before reaching the hardware (Fig. 2). A possible explanation for this is that VirtualBox has some well-refined implementations in terms of its hardware interfacing modules.

After seeing the above two results where

<table>
<thead>
<tr>
<th>VM environment</th>
<th>Host OS</th>
<th>RAM W Mb/s</th>
<th>RAM R Mb/s</th>
<th>Local W Mb/s</th>
<th>Local R Mb/s</th>
<th>Net W Mb/s</th>
<th>Net R Mb/s</th>
<th>Web R Mb/s</th>
<th>Dhrystone 2 (Billions of operations/s)</th>
<th>Whetstone (Millions of operations/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bare Metal</td>
<td>Rhel-32</td>
<td>641</td>
<td>532</td>
<td>519</td>
<td>536</td>
<td>10.3</td>
<td>54</td>
<td>80.2</td>
<td>6.0</td>
<td>1,085</td>
</tr>
<tr>
<td>Thin Hypervisor</td>
<td>XEN</td>
<td>541</td>
<td>370</td>
<td>231</td>
<td>385</td>
<td>5.8</td>
<td>7.3</td>
<td>10.8</td>
<td>5.7</td>
<td>537</td>
</tr>
<tr>
<td></td>
<td>ESXi</td>
<td>148</td>
<td>217</td>
<td>134</td>
<td>210</td>
<td>8.3</td>
<td>5.4</td>
<td>6.8</td>
<td>5.2</td>
<td>518</td>
</tr>
<tr>
<td></td>
<td>KVM_redhat64</td>
<td>282</td>
<td>368</td>
<td>272</td>
<td>346</td>
<td>11.1</td>
<td>5.7</td>
<td>15.2</td>
<td>5.7</td>
<td>530</td>
</tr>
<tr>
<td>VMware player</td>
<td>Win7-32</td>
<td>135</td>
<td>194</td>
<td>134</td>
<td>195</td>
<td>8.3</td>
<td>7</td>
<td>10.7</td>
<td>5.3</td>
<td>535</td>
</tr>
<tr>
<td></td>
<td>Win7-64</td>
<td>152</td>
<td>180</td>
<td>144</td>
<td>175</td>
<td>9.1</td>
<td>7.8</td>
<td>15</td>
<td>5.7</td>
<td>518</td>
</tr>
<tr>
<td></td>
<td>Win08-64</td>
<td>149</td>
<td>201</td>
<td>164</td>
<td>198</td>
<td>7.1</td>
<td>4.9</td>
<td>17.5</td>
<td>5.6</td>
<td>518</td>
</tr>
<tr>
<td></td>
<td>Redhat-64</td>
<td>60</td>
<td>265</td>
<td>76.4</td>
<td>278</td>
<td>8.5</td>
<td>9</td>
<td>13</td>
<td>5.3</td>
<td>524</td>
</tr>
<tr>
<td></td>
<td>Fedora 13-64</td>
<td>169</td>
<td>203</td>
<td>150</td>
<td>188</td>
<td>9.3</td>
<td>8.1</td>
<td>15.8</td>
<td>5.6</td>
<td>510</td>
</tr>
<tr>
<td>VirtualBox</td>
<td>Win7-32</td>
<td>634</td>
<td>461</td>
<td>1000</td>
<td>195</td>
<td>8.9</td>
<td>5.1</td>
<td>5.4</td>
<td>10.6</td>
<td>535</td>
</tr>
<tr>
<td></td>
<td>Win7-64</td>
<td>814</td>
<td>438</td>
<td>524</td>
<td>350</td>
<td>9.7</td>
<td>7.9</td>
<td>19</td>
<td>8.8</td>
<td>518</td>
</tr>
<tr>
<td></td>
<td>Win08-64</td>
<td>595</td>
<td>321</td>
<td>699</td>
<td>227</td>
<td>7.5</td>
<td>5.3</td>
<td>16.7</td>
<td>8.7</td>
<td>518</td>
</tr>
<tr>
<td></td>
<td>Redhat-64</td>
<td>629</td>
<td>416</td>
<td>232</td>
<td>496</td>
<td>11.6</td>
<td>3.7</td>
<td>4</td>
<td>9.1</td>
<td>821</td>
</tr>
<tr>
<td></td>
<td>Solaris 64 bit</td>
<td>346</td>
<td>319</td>
<td>326</td>
<td>206</td>
<td>9</td>
<td>7</td>
<td>7.9</td>
<td>8.5</td>
<td>852</td>
</tr>
<tr>
<td></td>
<td>Fedora 13-64</td>
<td>542</td>
<td>329</td>
<td>716</td>
<td>206</td>
<td>9.9</td>
<td>8.2</td>
<td>15.8</td>
<td>10.1</td>
<td>518</td>
</tr>
</tbody>
</table>
VirtualBox bested the bare metal system, another question is then raised regarding VMware’s implementation. Since VMware is the industry standard and has been around the longest, we would expect it to also outperform the bare metal system in the write and Dhrystone2 tests, however this was not the case.

IV. PERFORMANCE IMPACTS OF HOST OS

As a result of running a virtual system, the host OS experiences its own performance issues in terms of CPU, memory, graphics and the hard disk drive. In order to run virtual machines, it is necessary for the computer to have the physical resources required by the running virtual machine as well as enough resources for the host itself. In addition to this the computer will also need another 10% in overhead [14]. For these reasons it is imperative that the host OS be one that can effectively manage its hardware resources efficiently in order to perform virtualization and maintain host platform stability to its maximum potential.

In the study of host operating systems, Windows 7, Vista and XP were compared. In multiple tests, evidence is shown that Windows 7 dominates in many aspects of hardware management. It is important to note that Windows 7 and Vista produce similar results in many of the hardware management tests shown in Figure 13. This shows that both Operating Systems are built upon the same core architecture. In terms of graphics, results from Cinebench 10 show that Windows 7 and Windows Vista perform 6.15 % better than Windows XP (Fig. 9A). In regards to the CPU, performance is managed well amongst all operating systems however for aspects such as memory management, the PCMark2002 [15] test illustrates that it is done best with Windows Vista.

When it comes to hard disk management, Windows 7 does it better than the other two in comparison. Not only is this the case but when it comes to running a virtual machine on the host operating system, there is a difference between Windows 7, Vista and XP, with Windows 7 allocating resources more efficiently and effectively (Fig. 13B).

The tests of hardware management conclude that Windows 7 and Vista would be the better choice as a host operating system. When testing how the operating systems manage tasks like compression and coding, all do fairly well with only small differences, XP being at the bottom, however when it comes to Video encoding, Vista does the worst of the three. Notably through all tests, Windows 7 had the best performance of the three tested OS’s, signifying that when it comes to managing CPU and memory it is the superior and should be regarded as the primary choice for the host OS in virtualization.

The virtual systems have many platforms that can be used to run virtual machines on a computer. The

Figure 13. Performance of the host OS running virtualized environments [17]
virtual system can impact the host OS because it uses resources from the host OS itself. The amount of strain on the host OS depends on the applications running on the virtual machine. The performance impact can range from 10% to 35% or more.

The comparison of different types of hypervisor software; VMware, QEMU, VirtualBox, and VirtualPC were tested with Linux as the host OS. In the four tests performed fifty time each, the software that was used to perform the benchmark was 7z, Matrix, IOBench and NetBench. 7z, software for decompressing files, showed that all four virtual machines were slower than the native OS. VMware was the best performer with a 15% drop in comparison while QEMU performed the worst in comparison to the native environment.

The second test was performed using Matrix benchmark. This software evaluates the computational efficiency of the computer. These results were very similar to the previous test where VMware was the best and QEMU was the worst with a 30% performance drop compared to the native environment.

In the third test, the application IOBench heavily impacted the virtual environment. VMware performed 30% slower than the host OS. Virtual Box and VirtualPC performed nearly twice as slow and QEMU performed the worst at almost five times slower than the native environment.

The last software, NetBench, tested the network performance of the virtual machines. Surprising to the previous tests, QEMU was the fastest virtual machine after the native OS with a performance speed of 97.60 mbps.

Overall, virtual machines are good in some aspects, but not all. Virtual machines can really be an alternative to native computing with respect to application and network based applications. However, I/O based applications really use many resources of the virtual machines, thus not being feasible.

V. CONCLUSIONS AND FUTURE WORK

We presented a survey of several benchmarking studies conducted and generalized the results from these papers. Also discussed was the impact that virtualization has on the host operating system of a computer.

Future work in this area is as follows. A benchmarking study should be completed which analyzes the performance of hardware virtualization in comparison to software virtualization. In section IV we discussed how virtual machines running on a multicore system did not impact the performance of the host OS because the host was running on one core and the guest OS on another. However a further area of study would be determining the performance impacts a virtual OS has on the host OS in a uniprocessor system since both OS’s would be running from the same core and would need to be interleaved. Finally, an additional comprehensive benchmark test should be conducted which conducts performance comparisons at three separate levels: the host OS, the hypervisor and the guest OS. The goal of such a study would be to determine which combination of the three would result in the best performance for the host and guest operating systems.

VI. REFERENCES


