Performance Overhead Among Three Hypervisors: An Experimental Study using Hadoop Benchmarks

Jack Li, Qingyang Wang, Deepal Jayasinghe, Junhee Park, Tao Zhu, Calton Pu
School of Computer Science at the College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
{jack.li, qywang, deepal, jhpark, tao.zhu, calton}@cc.gatech.edu

***Abstract*—Hypervisors are widely used in cloud environments and their impact on application performance has been a topic of significant research and practical interest. We conducted experimental measurements of several benchmarks using Hadoop MapReduce to evaluate and compare the performance impact of three popular hypervisors: a commercial hypervisor, Xen, and KVM. We found that differences in the workload type (CPU or I/O intensive), workload size and VM placement yielded significant performance differences among the hypervisors. In our study, we used the three hypervisors to run several MapReduce benchmarks, such as Word Count, TestDSFIO, and TeraSort and further validated our observed hypothesis using microbenchmarks. In our observation for CPU-bound benchmark, the performance difference between the three hypervisors was negligible; however, significant performance variations were seen for I/O-bound benchmarks. Moreover, adding more virtual machines on the same physical host degraded the performance on all three hypervisors, yet we observed different degradation trends amongst them. Concretely, the commercial hypervisor is 46% faster at TestDFSIO Write than KVM, but 49% slower in the TeraSort benchmark. In addition, increasing the workload size for TeraSort yielded completion times for CVM that were two times that of Xen and KVM. The performance differences shown between the hypervisors suggests that further analysis and consideration of hypervisors are needed in the future when deploying applications to cloud environments.**

I. Introduction

The recent growth in cloud environments has accelerated the advancement of virtualization through hypervisors; however, with so many different virtualization technologies, it is difficult to ascertain how different hypervisors impact application performance and whether the same performance can be achieved for each hypervisor. Many different hypervisors (both open-source and commercial) exist today, each with their own advantages and disadvantages. This introduces a large number of new and challenging research questions. Some past work has focused on virtualization overhead of a single hypervisor on a particular application or microbenchmark [4]–[6]. Other work [11], [12], [15] has been aimed to provide a quantitative performance comparison between different hypervisors using microbenchmarks.

In this paper we try to provide an in-depth performance analysis of two commonly used open-source hypervisors (i.e., Xen and KVM) and a commercial hypervisor (referred to as CVM). Through our experiments, we aim to distinguish performance similarities and differences among the three hypervisors when running Hadoop MapReduce benchmarks. MapReduce has become an integral component of data-intensive applications in the cloud and many companies are running MapReduce applications in virtualized environments. Also, since the recent trend has been to colocate virtual machines to lower costs and increase utilization, in our experiments, we run a Hadoop cluster with all nodes residing in one physical host. To analyze our experiments, we run system monitoring tools in each virtual machine and at the host level to capture system performance. We also validate our Hadoop benchmark results with microbenchmarks from Filebench.

Through our large scale experiments, we observed that for some CPU-bound MapReduce benchmarks (such as Word Count), the performance between the three hypervisors was negligible with the highest percent difference being 15%; however, we found that for I/O-bound benchmarks, there were significant performance variations among the three hypervisors. For example, the commercial hypervisor is 46% faster at the data generation benchmark TestDFSIO Write than KVM, but 49% slower in the disk and CPU intensive TeraSort. These macro-benchmark results were verified using I/O microbenchmark results collected using FileBench. In addition, we found that that the three hypervisors showed different and interesting trends when adding more virtual machines to the same physical host and also adding additional workload size to the benchmarks.

In this paper, we do not deal with the speedup of Hadoop MapReduce or Hadoop benchmarks. Instead, the contribution of this paper is to provide some useful cases to help decide when to use which hypervisor for a particular workload. Generally, this study shows that virtualization technologies are a relatively immature technology and significantly more experimental analysis will be necessary in order for them to become truly suitable for mission-critical applications. Also, it is important to understand the characteristics about an application (e.g. is it CPU intensive or I/O intensive) in order to better determine the right virtualization platform to use.

The remainder of this paper is structured as follows. Section II provides an overview of the experimental setup, the different hypervisors used, and the benchmarks used in our experiments. In Section III we study the hypervisor's performance under different MapReduce benchmarks, and Section IV contains the results when scaling out VMs in a

978-0-7695-5006-0/13 $26.00 © 2013 IEEE
DOI 10.1109/BigData.Congress.2013.11

IEEE computer society

single physical host and changing VM placement. Related work is summarized in Section V, and Section VI concludes the paper.

II. Background

In this section, we briefly discuss our experimental approach. In Section II-A, we give an overview of our system. Section II-B has a brief description of each hypervisor used in our experiments, and in Section II-C, we provide a brief overview of the benchmarks run in this paper.

A. Overview of Experiments

Table I provides an overview of physical and virtual configurations we use for our experiments. We have selected the exact same virtual machine configurations across all three hypervisors (e.g., OS, Memory, disk). In terms of hypervisor configuration, we have setup each hypervisor and virtual machine using best practice guides. We use the Elba toolkit [1], [7], [8] to automatically generate and run the experiment as well as collect the results. During the runtime of each benchmark, with the exception of the CVM host, we run the system monitoring tool collectl [2] to collect system statistics in both the physical machine and each of the guest VMs with one second granularity. A proprietary monitoring tool was run on the CVM host with two second granularity.

TABLE I: Hardware and VM Configuration.

Physical Machine	
Processor	2 X Intel(R) Xeon(R) @ 2.27GHz (Quad)
Memory	16GB
Cache	2 X 8MB
Operating System	Debian Linux/Fedora 16
Virtual Machine	
Virtual CPUs	1
Memory	2GB
Disk Space	50GB
Operating System	Fedora 16

B. Hypervisors

We use three different hypervisors in our experiment: a commercial hypervisor, Xen, and KVM. There are many different commercial virtualization technologies each of which aim to provide similar or better performance and features as open-source hypervisors; however, most of them have licensing and copyright issues which prevent publications of performance and comparison data. Thus, in our study, we selected one of the commonly used commercial hypervisor (hereafter CVM). Xen is an open-source virtual machine monitor which is structured with the Xen hypervisor as the lowest and most privileged layer. Above this layer are one or more guest operating systems, which the hypervisor schedules across the physical CPUs. Xen can work both in para-virtualized or HVM mode; in the first, the guest operating system must be modified to be executed. Through paravirtualization, Xen can achieve very high performance. The HVM mode offers new instructions to support direct calls by a para-virtualized guest/driver into the hypervisor, typically used for I/O or other so-called hyper calls. KVM (also known as Linux KVM) is a full virtualization solution for Linux on x86 hardware containing virtualization extensions (Intel VT or AMD-V). It consists of a loadable kernel module, kvm.ko, which provides the core virtualization infrastructure, and a processor specific module, kvm-intel.ko or kvm-amd.ko. KVM requires a modified version of qemu, a well-known virtualization software.

C. Benchmarks

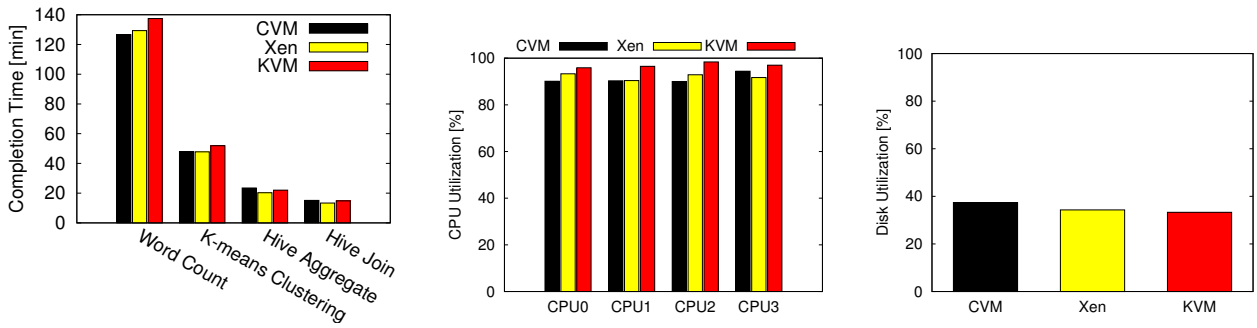
We run an unmodified Hadoop implementation of the MapReduce framework. Our Hadoop cluster comprises of 4 nodes that all reside in the same physical machine. Each node has one virtual core pinned to a different physical core, allotted 2 GB of memory, 50 GB of disk space, and is set to run at most 2 map tasks or 2 reduce tasks. We run 3 different Hadoop benchmarks found in version 1.0.2 including TestDFSIO Write/Read, and TeraSort and 3 benchmarks within the HiBench suite, namely, Word Count, K-Means Clustering, and Hivebench. TestDFSIO is a read and write benchmark that stresses the Hadoop Distributed File System (HDFS). It is mainly used to test the I/O speed of the cluster. TeraSort is used to sort a specified size of 100-byte records. The TeraSort benchmark is CPU, network, and I/O intensive and is usually considered a representative benchmark for real MapReduce workloads. In the TestDFSIO Write benchmark, the dataset that was generated was 10 GB with a replication factor of 3. Each of these benchmarks was run ten times and the average was used as the result of the experiments. The I/O-microbenchmark, Filebench, was also used to validate some of our macrobenchmark results. Filebench was run on a single VM with the same configuration as the Hadoop node. During Filebench experiments, only the VM running the microbenchmark was turned on, with all other VMs in the physical host turned off.

III. Experimental Results

We ran six different Hadoop MapReduce benchmarks in our experiments. Three of the benchmarks (TestDFSIO Write/Read and TeraSort) can be found in the latest Hadoop distributions. The last three benchmarks (Word Count, K-Means Clustering and Hivebench) were run from the HiBench benchmark suite [3]. We characterize these six benchmarks into three categories: CPU-bound, I/O-bound and both CPU and I/O-Bound benchmarks. CPU-bound benchmarks use a significant amount of CPU and the performance is limited due to a lack of CPU power. Similarly, I/O-bound benchmarks use a significant amount of disk (also significant CPU but only in terms of I/O Wait). CPU and I/O-Bound benchmarks use a significant amount of CPU and I/O, with the CPU being utilized both for computation and waiting for I/O.

A. CPU-Bound Benchmarks

We ran several CPU-bound benchmarks within the HiBench benchmarking suite including Word Count, K-Means Clus-



(a) CPU-Bound Benchmark Completion Times. The performance difference is small.

(b) Word Count Average CPU at VM Level. The CPU for each VM is heavily saturated.

(c) Word Count Average Disk Utilization at VM Level. The disk utilization for the VM is low.

Fig. 1: CPU-Bound Benchmark Results and Word Count Statistics. The performance difference for these benchmarks as seen in Figure 1(a) is negligible between the different hypervisors. A representative benchmark, Wordcount, shows high CPU utilization and low disk utilization during the job as seen in Figure 1(b) and 1(c).

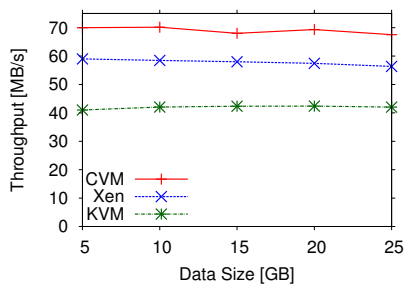
tering, and Hivebench. As previously mentioned, the CPU-benchmarks are characterized by a high CPU percentage throughout the experiment with most of the CPU coming from the user component. We found in all CPU-bound benchmarks that the performance impact when using different hypervisors was negligible. Figure 1(a) shows the completion times for the four mentioned CPU-bound benchmarks. As shown, the performance difference between the hypervisors for each benchmark is not very significant with the highest percentage difference being 15.7% for Hive Aggregate between the commercial hypervisor and Xen. Figure 1(b) shows the average CPU utilization during Word Count at the VM level. The benchmark shows very high saturation of the CPU for each hypervisor. Figure 1(c) shows the average disk utilization during Word Count also at the VM level. The disk is being utilized at an average of less than 40% for each hypervisor, but this utilization is not high compared to the I/O-bound benchmarks in the next section. The CPU and disk utilization statistics for the K-Means Clustering and Hivebench experiments are similar to Word Count and are omitted.

B. I/O-Bound Benchmarks

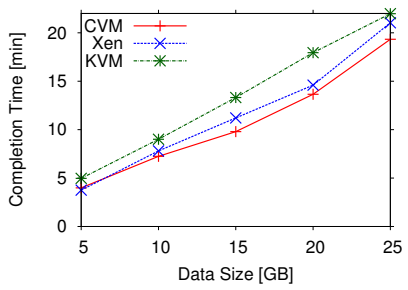
We ran TestDFSIO Write on a 4 node Hadoop cluster for each hypervisor and varied the write size to HDFS from 5GB to 25GB with 5GB increments. Figure 2(a) shows the disk write throughput for the hypervisors recorded at the host level. CVM has the fastest write throughput for all experiment sizes at an average of 69.0 MB/s while KVM has the slowest write throughput for all sizes at an average of 42.0 MB/s (a percent difference of 48.6%). These results directly correlate to the completion times for the experiment as shown in Figure 2(b). The average percent difference for TestDFSIO Write to generate the data for CVM and KVM is 24.2%. The difference in write throughput and completion times between the three hypervisors is significant. We found that the performance variation is due to the efficiency of request merging when processing a large number of write requests to disk. Figure 2(c) shows the number of write requests merged per second at the VM level. I/O request merging increases disk efficiency by combining requests servicing similar disk regions into one request. When I/O requests are merged at the VM level, it means there will be less privileged calls to the kernel. We see in this figure that CVM is able to merge a greater number of write requests to the disk than the other hypervisors. We ran a disk write benchmark within Filebench to validate the TestDFSIO Write results and are shown in Figure 3. We see a similar trend in the microbenchmark that we saw for the host level write throughput in Figure 2(b), but the CVM throughput for the microbenchmark is much higher than Xen and KVM.

We ran two I/O-bound benchmarks from Hadoop, namely TestDFSIO Write and TestDFSIO Read. The TestDFSIO benchmarks are used to test the health of the Hadoop Distributed File System by measuring total file system throughput. The I/O-bound benchmarks are characterized by high disk utilization and low CPU utilization (for the user component).

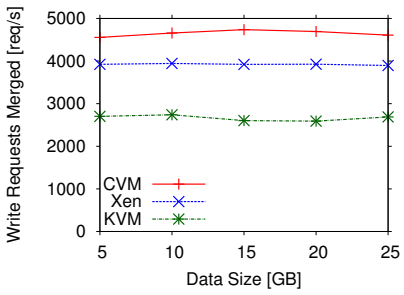
Figure 4(a) shows the read throughput at the host level for I/O-bound benchmark TestDFSIO Read. We ran TestDFSIO Read with the same setup as with TestDFSIO Write using a 4 node Hadoop cluster for each hypervisor and varied the experiment size from 5GB to 25GB with 5GB increments. For the read-intensive benchmark TestDFSIO Read, we see a reverse in the trend as in the TestDFSIO Write scenario. KVM reads data at an average throughput of 23.2 MB/s for the four data sizes while CVM is only able to read at an average throughput of 6.8 MB/s (a 109% difference). This read throughput confirms the completion times seen for the TestDFSIO Read benchmark shown in Figure 4(b). We looked into the VM level system statistics to diagnose CVM's poor read performance. Figure 5(a) and Figure 5(b) shows the average CPU Used % and CPU Disk Wait % at the VM level, respectively. Overall, the CPU usage for the VMs is quite high and most of the CPU being used is for I/O; however, if we look at the physical core utilizations for these VCPUs at the host level as seen in Figure 5(c)), we see that only KVM is heavily utilizing the physical CPUs. CVM's physical CPU usage is near 0% for all four cores yet the VMs report



(a) Disk Write Throughput Recorded at Host Level. CVM is the fastest with an average throughput of 69.0 MB/s and KVM is the slowest with an average of 42.0 MB/s.



(b) TestDFSIO Write Completion Times. The completion times match with the trend seen in throughput as shown in subfigure 2(a).



(c) Write Requests Merged Per Second at VM Level. CVM is able to merge more write requests and thus make less system calls to the hypervisor.

Fig. 2: TestDFSIO Write Results and Disk Statistics. Figure 2(a) shows the write throughput at the host level for each experiment size. The write throughput remains relatively constant for each hypervisor which yields a linear trend in completion times as the write size increases as shown in Figure 2(b). CVM performs the best and KVM performs the worst when writing to HDFS and this performance difference is due to more efficient write request merging in CVM compared to Xen and KVM as seen in Figure 2(c).

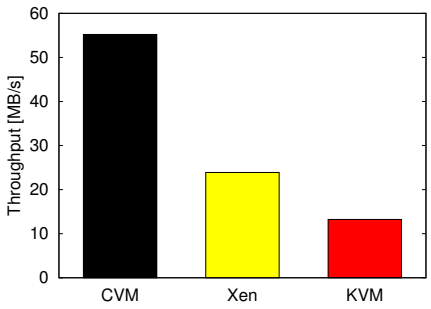


Fig. 3: Filebench Write Microbenchmark. The write microbenchmark results support the disk write throughput trend seen in TestDFSIO Write.

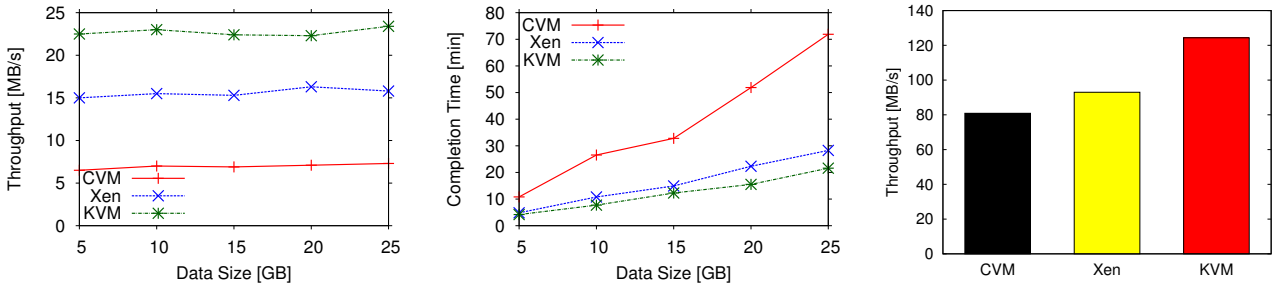
heavy utilization of its VCPUs. We believe the disparity here is due to a disk bottleneck and inefficiencies of the CVM dealing with numerous read requests going to the hypervisor. We suspect that a system call in the commercial hypervisor is locking the system and blocking subsequent privileged calls to the hypervisor from the VMs. This is an ongoing work that we will address in future papers. We also ran a disk read microbenchmark within Filebench to validate the TestDFSIO Read results and are shown in Figure 4(c). We see that the read microbenchmark results also has a matching trend with the TestDFSIO Read disk read throughputs in Figure 4(a).

C. CPU and I/O-Bound Benchmarks

TeraSort is both a CPU and disk intensive benchmark. Each worker has to read data from disk, perform the sort (where intermediate data gets written back to the disk), sends it to the reducer node where the reducer again performs an aggregate sort and writes the final result back to disk. We ran TeraSort with the same previous setup as in TestDFSIO Read/Write with 4 slave nodes with a sort size ranging from 5GB to 25GB in 5GB increments. The completion times for TeraSort are shown in Figure 6(a). We found that for a combination of heavy read and write-intensive I/O and CPU used for sorting, Xen performed the best and was able to sort the data the quickest. CVM had the worst performance for all sort sizes while KVM's performance was slightly worse than Xen's performance.

Figure 6(c) shows the completion times for each phase of the TeraSort job. During the map phase, each node reads data from disk and sorts it. We see that Xen and KVM complete this phase the fastest and CVM the slowest which is similar to the results found in the TestDFSIO read benchmark (KVM was fastest and CVM was slowest); however, for TestDFSIO Read, KVM was faster than Xen, but in this map phase Xen is faster than KVM. We believe this difference is due to the added CPU needed to compute the sort on the data; whereas Xen can use dom0 to offload some of the disk read which frees up CPU for the VMs in Xen to perform the sort, the KVM VMs must use its own CPU to read and perform the sort. In the next shuffle and sort phase, we see that Xen was again the fastest followed by KVM and then CVM. Generally in the shuffle phase, data is moved around the different nodes and sorted again which consumes both network and CPU. In our setup, all the nodes reside on the same physical machine so the network activity needed to do the shuffle is actually disk activity inside the physical machine. We can see here that again Xen is able to perform this phase the fastest due to dom0 support. KVM performs this step faster than CVM because it can read from disk faster. During the reduce phase, we see that Xen and CVM actually take around the same time to complete this phase (Xen is a little faster than CVM) while KVM takes the longest time. These results also parallel to the TestDFSIO Write results. During the reduce phase, all the data is in the reducer VM and the reducer VM performs one final sort and writes the data back to disk. Although CVM was able to write faster than Xen in the TestDFSIO Write benchmark and Filebench microbenchmark, here Xen is a little faster than CVM. The distinction here is that dom0 is able to free up CPU so that the reducer VM can use more CPU to perform the sort.

In TestDFSIO Write, Xen was found to be slower than

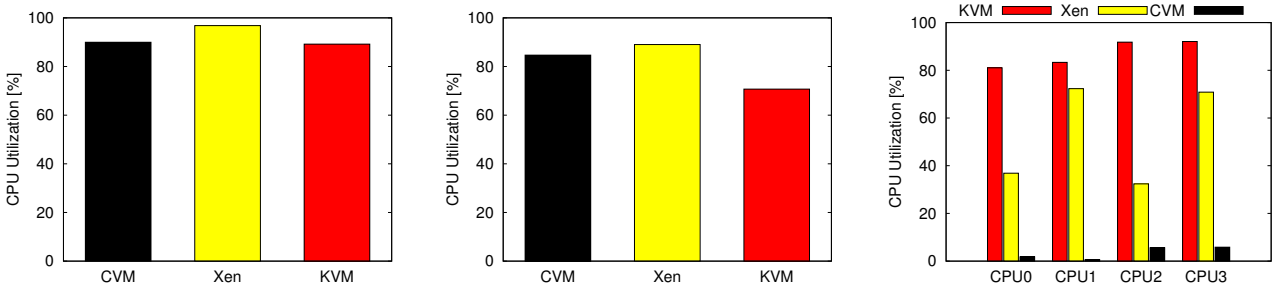


(a) Disk Read Throughput Recorded at Host Level. KVM's average throughput is 23.2 MB/s while CVM's average throughput is 6.8 MB/s.

(b) TestDFSIO Read Completion Times. KVM has the fastest completion time at all data sizes for TestDFSIO.

(c) Filebench Read Microbenchmark. The microbenchmark results confirm the throughput trend seen in TestDFSIO Read.

Fig. 4: TestDFSIO Read Results and Microbenchmark Measurement. Figure 4(a) shows that the read throughput for each hypervisor is relatively constant. CVM's throughput is less than 33% of KVM's throughput and this difference can be seen in the completion times in Figure 4(b). The Filebench read microbenchmark results in Figure 4(c) mirror the trend seen in the TestDFSIO Read benchmark.



(a) CPU Utilization Recorded at VM Level

(b) CPU Wait Utilization Recorded at VM Level

(c) Pinned CPU at Host Level

Fig. 5: TestDFSIO Read CPU data from VM and Host. To diagnose the poor read performance of CVM, we look at the average CPU utilizations at the VM level as seen in Figure 5(a). The VMs for each hypervisor are using a significant amount of CPU and Figure 5(b) reveals that the majority of this CPU is spent waiting for I/O; however, at the hypervisor level, the actual CPU for CVM shows very low utilization while Xen and KVM exhibit moderate CPU utilization as seen in Figure 5(c). Our hypothesis is that there is I/O-read scheduling contention within the CVM hypervisor which prevents it from performing well.

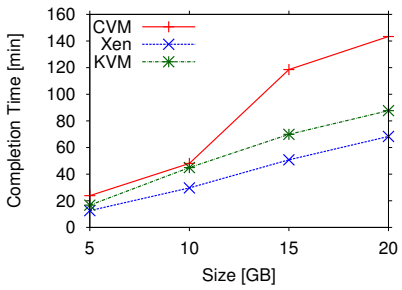
CVM. From our experiments, we see that Xen's dom0 uses a considerable amount of CPU to help process I/O requests. Figure 7 shows the CPU utilizations for the unpinned cores at the host level. The unpinned cores are the four cores remaining in the physical machine that are not tied to a virtual machine. The CVM and KVM hypervisors are not utilizing the unpinned cores, but for Xen hypervisor, dom0 is using a significant amount of CPU from the unpinned cores. We found that over 90% of this CPU in each core for Xen was in the I/O Wait component, which means Xen was using unpinned cores to process I/O. This CPU usage by dom0 makes the throughput of each VM in Xen for the read/write-intensive TeraSort workload much faster.

IV. Increasing Processing Parallelism and VM Placement

We found several interesting phenomenon when we ran our benchmarks. More specifically, each hypervisor exhibited different behavior when we tried to increase the processing parallelism in the Hadoop cluster by co-locating more virtual machines within the same physical host. For read and write benchmarks, increasing parallelism degrades performance. For both CPU and I/O-intensive workloads, increasing parallelism may lead to improved performance, but increasing parallelism too much may also degrade performance. Also, when we changed the VM colocation strategy by adding a new physical machine, we found that it is better to spread the VMs in the cluster to as many different physical machines as possible.

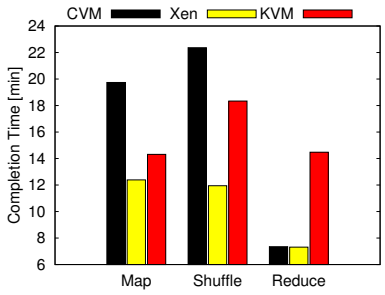
A. Increasing Processing Parallelism Through Colocation

When VMs are added to the same physical host in I/O-intensive benchmarks, the performance of these benchmarks degrades significantly. Figure 8(a) and Figure 8(b) shows the completion times for a 10GB TestDFSIO Write and TestDFSIO Read experiment when increasing the number of virtual machines from 1 to 4 on the same physical host. Each VM has the same specifications as the others used in the experiments in this paper. We see that for the two I/O-bound benchmarks, each hypervisor achieves the fastest completion time for the data write and data read with only one VM in the cluster. In fact, the performance of each hypervisor degrades significantly when increasing the number of VMs from 1 to 2, with CVM degrading much worse than the other two. For example, when increasing the number of VMs from 1 to 2, CVM's completion time for TestDFSIO Write nearly doubles, and its completion time for TestDFSIO Read increases six times. Interestingly,



(a) TeraSort Completion Times. CVM has the longest completion time while Xen has the shortest completion time. As data size increases, CVM degrades the most compared to the other two hypervisors.

(b) TeraSort Total Amount of Data Read and Written. The amount of intermediate data generated that is being written and read is significantly greater than the original sort size. This intermediate data degrades CVM's performance the most in TeraSort.



(c) 10GB TeraSort Duration of Each Phase. CVM spends most of the time in the map (heavy in disk reads) and shuffle (heavy in both disk reads and writes). KVM spends a lot of the time during shuffle and reduce (heavy in disk writes).

Fig. 6: TeraSort Results and Analysis. Figure 6(a) shows the completion times for TeraSort and reveals that Xen performs the best while CVM performs the worst. Figure 6(b) shows all the data that is read and written for each sort size. A significant amount of data is read from HDFS which makes CVM very slow as the size of the sort increases. Figure 6(c) shows the duration of each phase during a 10GB sort. CVM spends most of its time during the Map and Shuffle phases which is heavily dominated by reading data from HDFS. On the other hand, KVM spends relatively less time compared to CVM during the Map phase, but takes much longer during the reduce phase which is spent writing the sorted data back to HDFS.

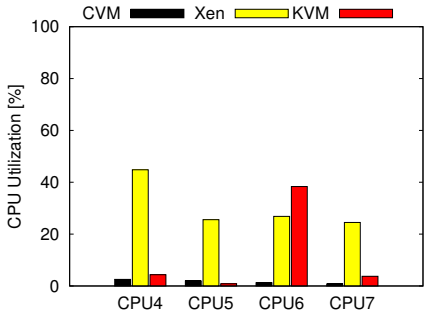


Fig. 7: TeraSort Unpinned CPU Utilization. Xen uses a considerable amount of CPU from its unpinned cores. A majority of this CPU is spent processing disk operations.

when increasing from 3 to 4 VMs in TestDFSIO Write, each hypervisor's throughput remains the same.

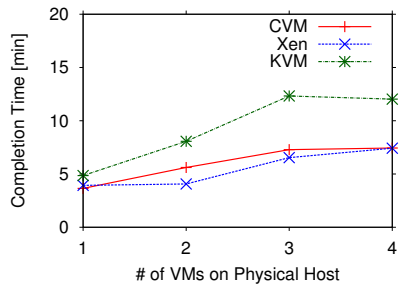
For TeraSort, we see a different trend. Figure 8(c) shows the completion time for a 10GB TeraSort when increasing the number of virtual machines from 1 to 4 on the same physical host. The performance when increasing the number of virtual machines from one to two actually improves performance for each hypervisor. This improvement occurs because when there is only one VM in the cluster for TeraSort, the bottleneck is on CPU resources to perform the sort, not an I/O bottleneck. When a second slave node is added, the extra computing power shifts the CPU bottleneck to disk. When you increase the number of VMs to three, you see that the I/O contention starts to take over for CVM and KVM, with CVM's completion time dramatically increasing. Xen, through the help of dom0, is able to maintain the same performance and does not suffer from the same I/O contention. Furthermore, when the number of VMs for CVM and KVM increase to four, the performance improves again. The reason for this improvement may seem strange but can be explained in Figure 9. The figure shows the average and total read and write throughput for CVM when increasing the number of VMs from 2 to 4. In Figure 9(a), the average read and write throughput decreases for each VM as the number of VM increases; however, the total read and write throughput as shown in Figure 9(b) shows a different trend. The total read and write throughput decreases from 2 to 3 VMs, but actually increases once the fourth VM is added. This occurs because the extra throughput from the fourth VM actually is greater than the throughput degradation lost adding the fourth VM.

B. Changing VM Placement

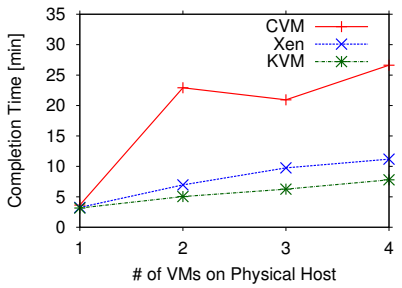
To alleviate some of the disk contention issues encountered by CVM for I/O-bound benchmarks, we propose to separate out VMs into multiple physical hosts as possible. Figure 10(a) shows the completion times for TestDFSIO Write and Read on two different node placement configurations. The first placement has two VMs co-located on a single physical host while the second placement has two VMs, but each VM being on a separate physical host. We see that when we choose not to co-locate the performance improves for write and significantly improves for the read case. In Figure 10(b), we have the same experiment but with four VMs. The first scenario has all four VMs on the same physical host, while the second scenario has two VMs each on two physical hosts. Similarly, we see that performance improves for both read and write with the read case being much better. Although these results don't suggest co-locating virtual machines when running I/O-intensive benchmarks, the better solution is to find ways to minimize the disk contention for these types of benchmarks within each hypervisor or for the actual application.

V. Related Work

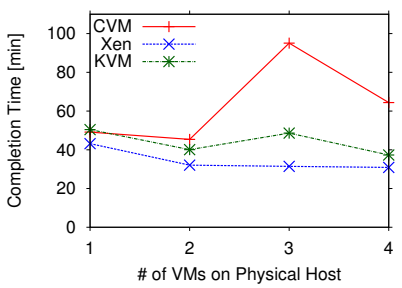
The increasing popularity of virtualization and cloud computing has spawned very interesting research on private and



(a) TestDSFIO Write 10GB Completion Times for 1-4 VMs. As processing parallelism increases, the performance degrades.

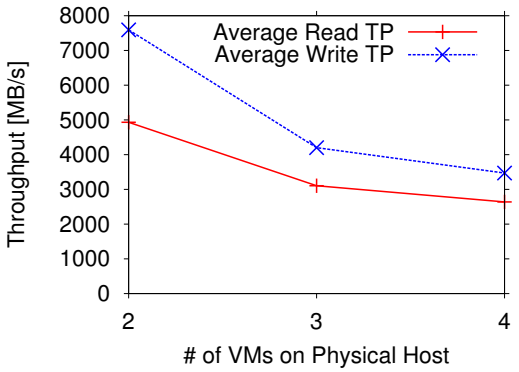


(b) TestDSFIO Read 10GB Completion Times for 1-4 VMs. As processing parallelism increases, the performance degrades.

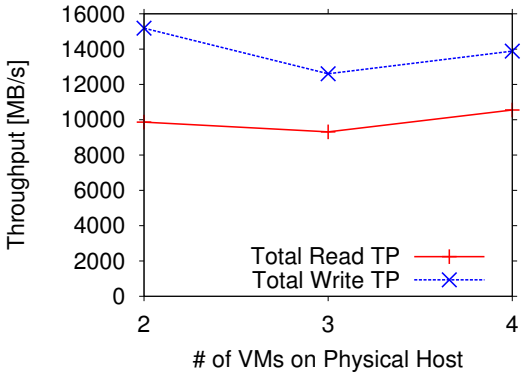


(c) TeraSort 10GB Completion Times for 1-4 VMs. The addition of the second VM increases performance by alleviating a CPU bottleneck.

Fig. 8: Effects of Adding VMs on the same Physical Host on TestDFSIO Write/Read and TeraSort. For TestDFSIO Write and Read as shown in Figure 8(a) and 8(c), as more VMs are added to the same physical host, the completion time for the experiment increases. For TeraSort in Figure 8(c), when increasing beyond 2 VMs, I/O scheduling contention makes the completion times increase for CVM and KVM, but Xen deals with the I/O contention quite well and the completion times do not increase as the number of VMs increases.



(a) Average Read and Write Throughput decreases from 2 to 4 VMs



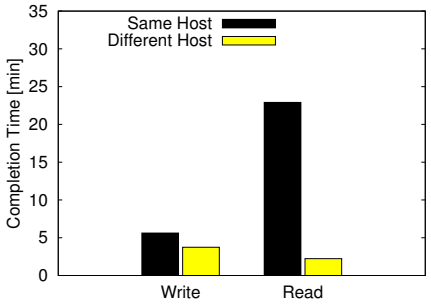
(b) Total Read and Write Throughput increases from 3 to 4 VMs

Fig. 9: The Average and Total Read and Write Throughput Using the Commercial Hypervisor. Increasing the number of VMs decreases the average read and write throughput per VM, however the total throughput increases from 3 to 4 VMs due to the extra throughput from the fourth VMs being greater than the degradation.

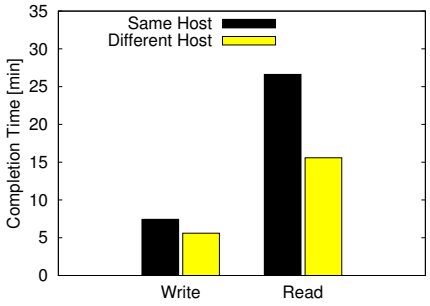
public clouds. Barham et al. [9] benchmarked Xen against VMware Workstation and User-Mode Linux. They show that Xen outperforms them on a range of micro-benchmarks and system-wide tests. Clark et al. [10] repeated this performance analysis of Xen in [9] and confirmed the high performance of Xen claimed in [9]. Also they compared Xen on x86 with IBM zServer and found that the former has better performance than the latter. Interestingly, in our study we observed complex results, where different hypervisor show different performance characteristics for varying workloads.

VMware [11] provides a quantitative and qualitative comparison of VMwareESX Server 3.0.1 and open-source Xen 3.0.3. In CPU tests, such as SPECcpu2000 and Passmark, the Xen hypervisor's overhead is twice as much as VMwareESX Server's overhead. Xen hypervisor performs extremely poorly in the Netperf tests having only 3-6 percent of the native performance while VMware ESX Server delivers near native performance. Xen argues that it is not fair to compare a commercial product with a code base in [11], so they perform an evaluation of the performance and scalability of VMware ESX Server versus XenSource's XenEnterprise [12], both of which are commercial products. Using the same benchmark as in [11], XenEnterprise performs as well as ESX in most benchmark tests. XenEnterprise even outperforms in a couple of areas such as compile time, Netperf and Passmark memory operations. Also, XenEnterprise scales well to utilize multiple CPUs to allow a wide variety of workloads to be run.

In an another study, Padala et al. [13] concentrate on comparing Xen's and OpenVZ's performance when used for consolidating multi-tiered applications. Their experimental results show that Xen incurs higher overhead than OpenVZ does and average response time can increase by over 400% in Xen and only 100% in OpenVZ as the number of application instances increases. This can be explained due to Xen having higher L2 cache misses than OpenVZ. Meanwhile, Adams et al. [14] compare software VMM (binary translation) with hardware-assisted VMM. They show that software and hardware VMMs both perform well on compute-intensive workloads. However,



(a) Using Two VMs Each On Its Separate Physical Host



(b) Using Four VMs, Two Each on Two Separate Physical Hosts

Fig. 10: Effects of Changing VM Placement Using the Commercial Hypervisor. Minimizing the number of VMs on a physical machine for CVM drastically decreases the completion times for TestDFSIO Read and slightly improves the completion times for TestDFSIO Write.

if workloads include progressively more privileged operations such context switches, memory mapping, I/O, interrupts and system calls, both VMMs suffer overheads while software outperforms hardware.

Deshane et al. [15] focus on three aspects of benchmarking Xen and KVM: overall performance, performance isolation, and scalability. They illustrate that Xen has excellent scalability, while KVM has substantial problems with guests crashing when it hosts more than 4 guests. KVM outperforms Xen in isolation. In overall performance test, Xen has a better performance than KVM on a kernel compile test, while KVM outperforms Xen on I/O-intensive tests. Camargos et al. [16] analyze the performance and scalability of six virtualization technologies (KQEMU, KVM, Linux-VServer, OpenVZ, VirtualBox and Xen) for Linux. They find Linux-Server delivers little or even no overhead in all test. In all these experiments, Xen, KVM and VirtualBox perform well. The experimental result of OpenVZ is disappointing and they suggest that KQEUM should be avoided in production systems.

VI. Conclusion

In this paper, we ran experiments on a commercial virtual machine monitor, Xen and KVM, using MapReduce. We verified our results using the Filebench micro-benchmark. We analyzed the performance results for each experiment and found similarities and significant performance variations between the different hypervisors. For example, for CPU-bound benchmarks, we found that there was negligible performance differences among the hypervisors; however, for I/O-bound benchmarks and benchmarks that were both CPU and I/O intensive, there were considerable performance differences. The commercial hypervisor was found to be better at disk writing, while KVM was better for disk reading. Xen was better when there was a combination of disk reading and writing with CPU intensive computations. Increasing processing parallelism for I/O-intensive workloads is detrimental to performance, but may increase performance in I/O and CPU-intensive workloads.

More generally, this work enhances the understanding of hypervisors when running MapReduce workloads. In addition, experiment results show the need to better study the differences among hypervisors in order to better determine which hypervisor to use for user applications.

References

[1] The Elba project. http://www.cc.gatech.edu/systems/projects/Elba/.
[2] Collectl http://collectl.sourceforge.net/.
[3] S. Huang, J. Huang, J. Dai, T. Xie, B. Huang. The HiBench benchmark suite: Characterization of the MapReduce-based data analysis. In *ICDEW '10*.
[4] A. Menon, J.R. Santos, Y. Turner, G. Janakiraman, W. Zwaenepoel. Diagnosing performance overheads in the xen virtual machine environment In *VEE'05*.
[5] A. Ranadive, M. Kesavan, A. Gavrilovska, K. Schwan. Diagnosing performance overheads in the xen virtual machine environment In *VEE'05*.
[6] A. Kudryavtsev, V. Koshelev, B. Pavlovic, A. Avetisyan. Virtualizing HPC applications using modern hypervisors. In *FederatedClouds'12*.
[7] G. Swint, G. Jung, C. Pu, A. Sahai. Automated Staging for Built-to-Order Application Systems. In *NOMS 2006*, Vancouver, Canada.
[8] S. Akhil, C. Pu, G. Jung, Q. Wu, W. Yan, G. Swint. Towards Automated Deployment of Built-to-Order Systems. In *DSOM 2005*.
[9] P. Barham , B. Dragovic , K. Fraser , S. Hand , T. Harris , A. Ho , R. Neugebauer , I. Pratt and A. Warfield. Xen and the Art of Virtualization. In Proceedings of the 9th ACM Symposium on Operating Systems Principles, pages 164-177, 2003.
[10] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne and J. Matthews. Xen and the Art of Repeated Research. In Proceedings of the 2004 USENIX Annual Technical Conference, pages 135-144, 2004.
[11] VMware Inc. A Performance Comparison of Hypervisors. Technical report, VMware Inc., 2007.
[12] XenSource. A Performance Comparison of Commercial Hypervisors. Technical report, XenSource, Inc., 2007.
[13] P. Padala, X. Zhu, Z. Wang, S. Singhal, and K. G. Shin. Performance Evaluation of Virtualization Technologies for Server Consolidation. Technical report, HP Labs, 2007.
[14] K. Adams and O. Agesen. A Comparison of Software and Hardware Techniques for x86 Virtualization. In ASPLOS, Pages 2 -13, 2006.
[15] T. Deshane, Z. Shepherd, J. Matthews, M. Ben-Yehuda, A. Shah and B. Rao. Quantitative Comparison of Xen and KVM. Xen Summit, 2008.
[16] F. Laudares Camargos, G. Girdard, and B. des Ligneris. Virtualization of Linux Servers: A Comparative Study. In Proceedings of the 2008 Linux Symposium, volume 1, pages 63-76, 2008.