A Component-Based Performance Comparison of Four Hypervisors

Jinho Hwang
The George Washington University
jinho10@gwu.edu

Sai Zeng and Frederick y Wu
IBM T. J. Watson Research Center
{saizeng, fywu}@us.ibm.com

Timothy Wood
The George Washington University
timwood@gwu.edu

***Abstract*—Virtualization has become a popular way to make more efficient use of server resources within both private data centers and public cloud platforms. While recent advances in CPU architectures and new virtualization techniques have reduced the performance cost of using virtualization, overheads still exist, particularly when multiple virtual machines are competing for resources. We have performed an extensive performance comparison under hardware-assisted virtualization settings considering four popular virtualization platforms, Hyper-V, KVM, vSphere and Xen, and find that the overheads incurred by each hypervisor can vary significantly depending on the type of application and the resources assigned to it. We also find dramatic differences in the performance isolation provided by different hypervisors. However, we find no single hypervisor always outperforms the others. This suggests that effectively managing hypervisor diversity in order to match applications to the best platform is an important, yet unstudied, challenge.**

***Keywords*—Cloud Computing, Hypervisor, Benchmark**

I. Introduction

Virtualization technologies have dramatically changed how businesses run their applications, first by enabling much greater consolidation within private data centers, and more recently as a driving technology behind cloud computing. Whether it is used with public or private resources, virtualization simplifies management, speeds up deployment, and improves resource efficiency. Virtualization enables new features such as performance management and reliability services to be applied without requiring modifications to applications or operating systems. While the overheads of the virtualization layer still prevent it from being used in performance critical domains such as high performance computing, virtualization has become the norm for running a growing set of applications.

The uptake of virtualization has been driven recently by the ease with which Infrastructure as a service (IaaS) cloud platforms can rent users virtual machines (VMs). Many enterprises appreciate the rapid deployment and flexible scalability of these clouds—traits which rely in part on the ease with which virtual servers can be created and adjusted on the fly. However, the main purpose of using virtualization technology is to consolidate workloads so that one physical machine can be multiplexed for many different users. This improves the efficiency of an overall data center by allowing more work to be done on a smaller set of physical nodes [1], and also improves the per-server energy efficiency because even idle servers consume a great deal of energy [2].

While virtualization provides many conveniences, it comes at a cost. The hypervisor which manages the virtualization platform incurs some overhead simply because of the layer of abstraction it must add between a VM and the physical resources it makes use of [3]. Further, since many VMs may run on one physical machine, performance isolation is critical to ensure that competing VMs do not negatively impact one another. For example, a CPU scheduler in the hypervisor must provide a fair amount of time to each VM and prevent a greedy VM from hurting others [4, 5].

There are numerous virtualization platforms ranging from open-source hypervisors such as KVM and Xen, to commercial hypervisors such as VMware vSphere and Microsoft Hyper-V. While the goals of these platforms are the same, the underlying technologies vary, leaving system administrators responsible for picking the ideal virtualization platform based on its performance, features, and price. The choice of hypervisor does not only apply to an enterprise's private data center—different cloud services make use of different virtualization platforms. Amazon EC2, the largest infrastructure cloud, uses Xen as a hypervisor, but Microsoft Azure uses Hyper-V and VMware partners use ESX. Recently, Google launched its own IaaS cloud that uses KVM as a hypervisor [6]. This kind of hypervisor diversity causes new challenges in managing resources due to the different APIs and feature sets supported by each cloud and virtualization platform, but it also promises new opportunities if applications can be carefully matched to the best hypervisor.

The combination of new CPU architectures with embedded virtualization support and advances in hypervisor design have eliminated many of their performance overheads. Despite this, popular hypervisors still exhibit different levels of performance. As a motivating example, we have configured one blade server with four different hypervisors in different disk partitions: Hyper-V, KVM, VSphere, and Xen. We create an identical Ubuntu VM with 1 virtual CPU (VCPU) and 2 GB of RAM under each platform and measure the time required to compile the Linux 2.6 kernel. Even with this straightforward operation, we find significant performance differences under each hypervisor, as illustrated in Figure 1.

While these results suggest that Xen is a poor choice for a hypervisor, our study reveals a more complicated scenario where each hypervisor has different strengths and weaknesses. The result is a far more complicated management decision than many system administrators are currently aware. The virtualization platform (and by extension, cloud platform) best suited for a given application is dependent on both the nature of its resource requirements and on the types of applications it

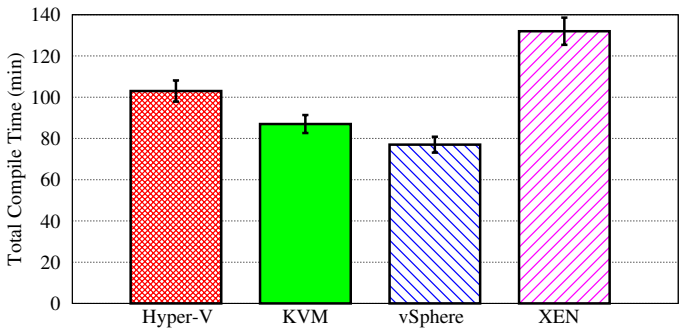


Fig. 1. Linux Compile Workloads

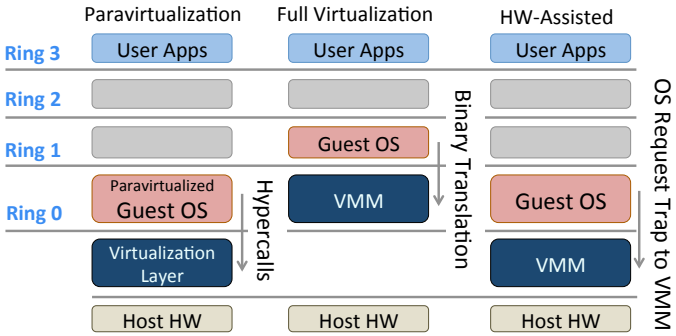


Fig. 2. Different Approaches to Providing the Virtualization Layer

will run alongside. Therefore, situational resource management needs to be considered in highly scalable cloud architecture.

To understand the relative strengths and weaknesses of different hypervisors, in this paper we perform an extensive performance comparison of four popular virtualization platforms. We use a component-based approach that isolates performance by resource such as CPU, memory, disk, and network. We also study the level of performance isolation provided by each hypervisor to measure how competing VMs may interfere with each other [7–10]. Our results suggest that performance can vary between 3% and 140% depending on the type of resource stressed by an application, and that the level of interference caused by competing VMs can range from 2X to 10X depending on the hypervisor. We believe that this study can act as a foundation for driving new research into management systems designed to take advantage of hypervisor diversity to better match applications to the platforms they will run the most efficiently on.

II. Background & Related Work

Virtualization technology provides a way to share computing resources among VMs by using hardware/software partitioning, emulation, time-sharing, and dynamic resource sharing. Traditionally, the operating system (OS) controls the hardware resources, but virtualization technology adds a new layer between the OS and hardware. A virtualization layer provides infrastructural support so that multiple VMs (or guest OS) can be created and kept independent of and isolated from each other. Often, a virtualization layer is called a hypervisor or virtual machine monitor (VMM). While virtualization has long been used in mainframe systems [11], VMware has been the pioneer in bringing virtualization to commodity x86 platforms, followed by Xen and a variety of other virtualization platforms [12, 13].

Figure 2 shows three different approaches to virtualization: para-virtualization (PV), full virtualization (FV), and hardware-assisted virtualization (HVM). Paravirtualization requires modification to the guest OS, essentially teaching the OS how to make requests to the hypervisor when it needs access to restricted resources. This simplifies the level of hardware abstraction that must be provided, but version control between the hypervisor and paravirtualized OS is difficult since they are controlled by different organizations. Full virtualization supports running unmodified guests through binary translation. VMware uses the binary translation and direct execution techniques to create VMs capable of running proprietary operating systems such as Windows [12]. Unfortunately, these techniques can incur large overheads since instructions that manipulate protected resources must be intercepted and rewritten. As a result, Intel and AMD have begun adding virtualization support to hardware so that the hypervisor can more efficiently delegate access to restricted resources [12]. Some hypervisors support several of these techniques; in our study we focus solely on hypervisors using hardware-assisted virtualization as this promises to offer the greatest performance and flexibility.

Our target hypervisors are Hyper-V, KVM, VMware vSphere, and Xen. Each of these hypervisors use different architectures, even when restricted to HW-assisted virtualization mode. The Windows-based Hyper-V has a significantly different architecture than the others which all derive from Linux. While Xen and KVM use open-source modifications of the Linux kernel, VMware uses a custom build with proprietary features [12]. Xen was initially based on the paravirtualization technique, but it now supports HVM as well [13]. However, it still retains a separate management domain (dom0) which controls VMs and can negotiate access to custom block and network drivers. KVM runs as a kernel module, which means it uses most of the features of the linux kernel operating system itself. For example, rather than providing its own CPU scheduler for VMs, KVM treats each VM as a process and uses the default Linux scheduler to allocate resources to them.

A variety of software and operating system aspects can affect the performance of hypervisors and VMs. In particular, how the hypervisor schedules resources such as CPU, memory, disk, and network are critical factors. Each of these resources requires different techniques to virtualize, leading to performance differences in each hypervisor depending on the types of activities being performed. Table I summarizes performance-related elements for each hypervisor.

Both research and development efforts have gone into reducing the overheads incurred by virtualization layers. Prior work by Apparao, et al., and Menon, et al., has focused on network virtualization overheads and ways to reduce them [14, 15]. Others have examined overheads and performance variations caused by different CPU schedulers within the Xen hypervisor [4]. Our prior work attempted to build models of virtualization layer overheads to facilitate the migration from native to virtual environments [16]. Benchmark comparisons between different hypervisors have been performed both by companies such as VMware [17], and by independent tech

TABLE I. FEATURE COMPARISONS OF HYPERVISORS (* DEFAULT OR USED IN THIS PAPER)

Features	Hyper-V	KVM	vSphere	XEN
Base OS	Windows Server	Linux (+QEMU)	vmkernel (linux-based)	Linux (+QEMU)
Latest Release Version	2008 R2	2.6.32-279	5.0	4.1.2
Architecture	Bare-Metal	Bare-Metal (controversial)	Bare-Metal	Hosted (controversial)
Supported Virtualization Technologies	Para-Virtualization, Hardware-Assisted Virtualization*	Para-Virtualization, Full Virtualization, Hardware-Assisted Virtualization*	Full Virtualization, Hardware-Assisted Virtualization*	Para-Virtualization, Full Virtualization, Hardware-Assisted Virtualization*
CPU Scheduling Features	Control with VM reserve, VM limit, relative weight	Linux schedulers (Completely Fair Queuing Scheduler*, round-robin, fair queuing, proportionally fair scheduling, maximum throughput, weighted fair queuing)	Proportional Share-based Algorithm, Relaxed Co-Scheduling, Distributed Locking with Scheduler Cell	SEDF (Simple Earliest Deadline First), Credit*
SMP Scheduling Features	CPU Topology-based Scheduling	SMP-Aware Scheduling	CPU Topology-aware Load Balancing	Work-Nonconserve, Work-Conserve*
Memory Address Translation Mechanism	Shadow Pagetable, Hardware-Assisted Pagetable (Second Level Address Translation)*	Shadow page table, Hardware-Assisted Pagetable*	Eumulated TLB, Shadow Pagetable, Hardware Assisted Pagetable (nested pagetable)*	Direct Pagetable (PV mode), Shadow Pagetable (HVM mode), Hardware-Assisted Pagetable*
Disk Management Features	Fixed disks, pass through disks, dynamic disks*	No-op, anticipatory, deadline, completely fair queue (CFQ)*	Latency-aware Priority-based scheduler, storage DRS	No-op, anticipatory, deadline, completely fair queue (CFQ)*
Network Management Features	TCP offload, large send offload, VM queue	FIFO-base scheduling	Priority-based NetIOC (Network I/O Control), TCP segmemtation offload, netqueue, distributed virtual switch	FIFO-based scheduling

bloggers [18]; however, these studies have tried to simply find the fastest hypervisor, not understand the strengths and weaknesses of each as we do.

III. METHODOLOGY

The methodology for our performance comparison of hypervisors is to drill down each resource component one by one with a specific benchmark workload. The components include CPU, memory, disk I/O, and network I/O. Each component has different virtualization requirements that need to be tested with different workloads. We follow this with a set of more general workloads representative of higher-level applications.

When a VM is created, it is assigned a certain number of virtual CPUs (VCPU). VCPU can represent how many cores this VM can use. However, VCPU does not guarantee a specific physical CPU is dedicated to the VM, rather it represents a flexible assignment of physical to virtual CPUs, which may be further subdivided based on the scheduling weights of different VMs. The CPU scheduling parameters used by the hypervisor can impact the overhead added for handling CPU-intensive tasks. We study cases where VMs are assigned a single VCPU or four VCPUs (the max on our test system).

The hypervisor must provide a layer of indirection between the guest OS and the system's memory to ensure both performance isolation and data integrity. With hardware-assisted virtualization, this mapping is done through Extended Page Table (Intel) or Rapid Virtualization Indexing (AMD) support built into the Memory Management Unit (MMU), which provides a significant performance boost compared to doing memory translation in software [12]. Despite all using this hardware, the hypervisors we compare can all take advantage of it in different ways, leading to varying performance levels.

Disk IO is a common source of overhead in virtualization platforms. If paravirtualization is used, then the IO path between the guest VM and hypervisor can be optimized. With full virtualization, this is not possible, and there is not yet wide support for hardware-assisted disk device virtualization. As a result, the hypervisor must emulate the functionality of the disk device, potentially leading to large overheads. To characterize these overheads, we test a range of IO types and sizes, as well as higher level IO behavior such as that from an email, file, or web server.

Network performance is also a critical source of overhead for many virtualization platforms since many VMs run network-based applications such as web sites. Therefore, two major factors for the performance are network bandwidth (throughput), and latency. Preliminary support for network card-assisted virtualization is being developed [19], but this feature is not standardized enough for our tests. Our network benchmarks stress both low-level network characteristics and web server performance.

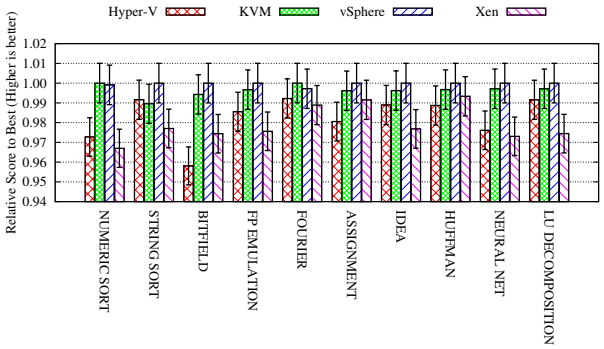
In addition to the component-level tests described above, we run several application level benchmarks. These benchmarks illustrate how the overhead of different components interact to determine overall performance. Finally, we also test scenarios where multiple interfering VMs run simultaneously. Performance isolation is an important goal for the virtualization layer, particularly when used in cloud environments. If the performance isolation fails, customers may complain that the VM performance varies depending on other tenants' usage pattern. In our composite benchmarking experiments, we explore how well the hypervisor schedules or manages the physical resources shared by VMs.

IV. BENCHMARK RESULTS

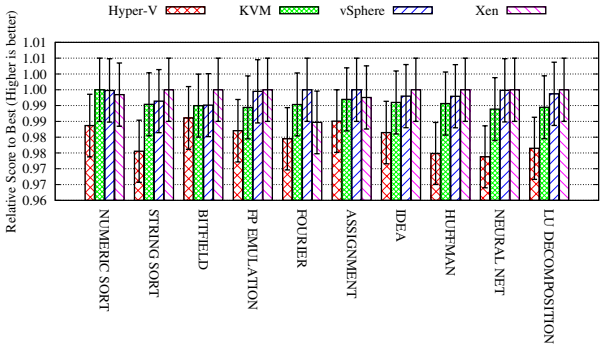
A. Experimental Setup

The goal of our system setup is to provide complete fairness in all the aspects that can affect the system performance.

Hardware Setting: For a fair comparison, the hardware settings are exactly the same for all the hypervisors by using one server machine, which has two 147GB disks that are divided into three paritions. Hyper-V occupies one partition, VMware vSphere occupies one parition, and KVM and Xen share the same linux installation that can be booted using either Xen or the KVM kernel. The machine has Intel(R) Xeon (R) 5160

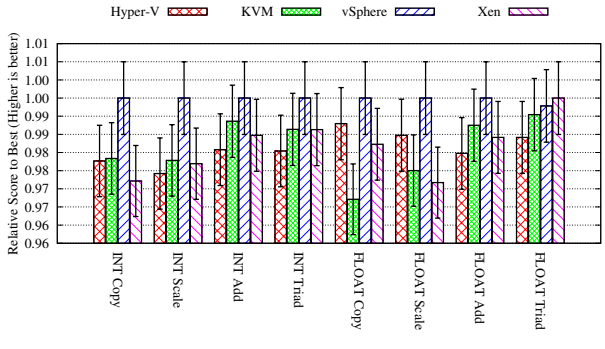


(a) 1 VCPU Case

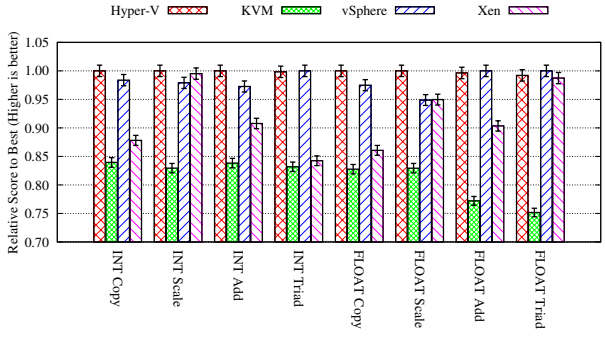


(b) 4 VCPUs case

Fig. 3. Bytemark Benchmark Result (CI = 99%)



(a) 1 VCPU Case



(b) 4 VCPUs case

Fig. 4. Ramspeed Benchmark Result (CI = 99%)

3.00GHz/800MHz four core CPU, 8GB memory, and shared 3MB L2 cache per core (12MB). The disk has LSI logic 1064E SAS 3 GBps controller IBM-ESXS model, and the network is dual broadcom 5708S gigabit ethernet.

Guest OS: The base guest VM OS is Ubuntu 10.04 LTS Lucid Lynx (Linux kernel 2.6.32), 10GB size disk image, and has 2048MB memory assigned. Each hypervisor has this base guest VM with exactly the same environment setup. Interference generator VMs use the same setting with the base guest VM, but it is assigned only 1024MB of memory.

B. Bytemark

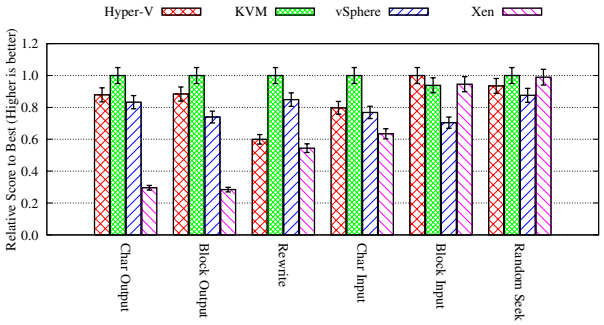
Bytemark [20] is primarily designed to stress the capabilities of the CPU. It tests a *nix machine in different ways, and runs both integer and floating point arithmetic tests. 10 workloads include: 1) numeric sort sorts an array of 32-bit integers, 2) string sort sorts an array of strings of arbitrary length, 3) bitfield executes a variety of bit manipulation functions, 4) emulated floating-point is a small software floating-point package, 5) Fourier coefficients is a numerical analysis routine for calculating series, and approximations of waveforms, 6) assignment algorithm is a well-known task allocation algorithm, 7) Huffman compression is is well-known text and graphics compression algorithm, 8) IDEA encryption is a relatively new block cipher algorithm, 9) Neural Net is a small but functional back-propagation network simulator, 10) LU Decomposition is a robust algorithm for solving linear equations.

As seen in Figure 3, we find that all of the hypervisors perform quite similarly when running these benchmarks. This is because basic CPU operations require no help from the hypervisor at all, allowing them to run at near-native speeds. Increasing the number of VCPUs running the benchmarks from one to four gives similar results for all of the hypervisors, with only minor fluctuations. Confidence interval (CI) is 99% for both cases. CI is calculated with $E = t(\alpha/2) \times s/\sqrt{n}$, where E is the maximum error, α is 1-degree of confidence, $t(\cdot)$ is t-value for a two-tailed distribution, s is the standard deviation, and n is the number of samples. To find $t(\cdot)$, we use Student t distribution table with the corresponding confidence interval. We obtain $\mu \pm E$, where μ is the mean of the samples.

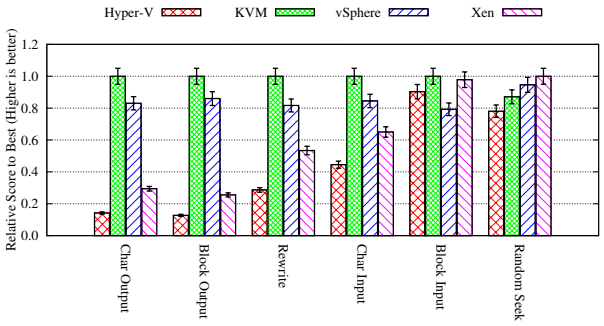
C. Ramspeed

Ramspeed (a.k.a. ramsmp for symmetric multiprocessing) [21] is a tool to measure cache and memory bandwidth. Workloads include integer and float operations. *Copy* simply transfers data from one memory location to another. *Scale* modifies the data before writing by multiplying with a certain constant value. *Add* reads data from the first memory location, then reads from the second, adds them up and writes the result to the third place. *Triad* is a merge of Add and Scale.

The benchmark results shown in Figure 4(a) illustrate the memory access performance of all hypervisors running on a single VCPU are all within 3%. However, Figure 4(b) shows up to a 25% performance difference in FLOAT Triad case for KVM. While all of the other hypervisors exhibit increased performance from the availability of multiple VCPUs (e.g., vSphere rises from 2977 MB/s to 4086 MB/s), KVM performance stays relatively flat (a rise from 2963 MB/s to only 3210 MB/s). This suggests that KVM is not able to fully saturate the memory bus, perhaps because dedicating a full 4 VCPUs to the guest OS causes greater interference with the primary Linux system under which KVM is installed. Notably,

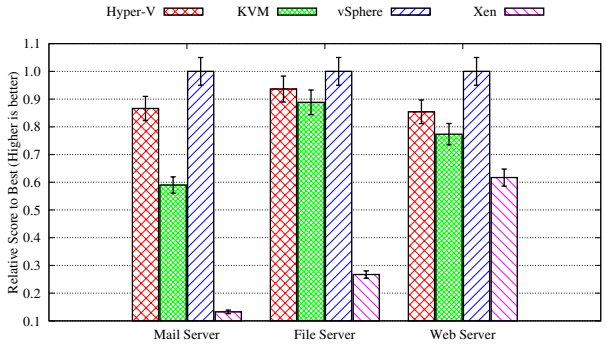


(a) 1 VCPU Case

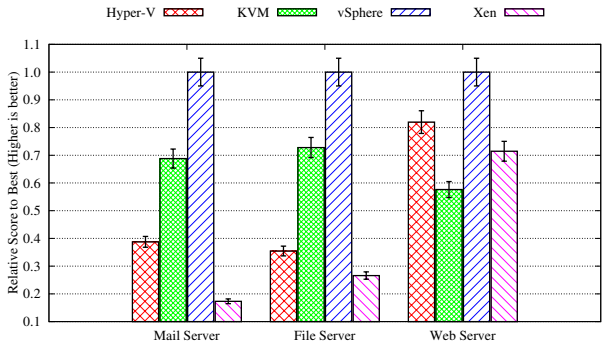


(b) 4 VCPUs case

Fig. 5. Bonnie++ Benchmark Result (CI = 95%)



(a) 1 VCPU Case



(b) 4 VCPUs case

Fig. 6. Filebench Benchmark Result (CI = 95%)

Xen sees a similar, though not as significant performance gap when moving to four VCPUs—not a surprise since both KVM and Xen have heavy weight management systems, requiring a full linux OS to be loaded to run the hypervisor.

D. Bonnie++ & FileBench

Bonnie++ [22] is a disk throughput benchmark. Our system does not have any form of hardware-assisted disk virtualization, so all of the hypervisors must take part in disk operations. As a result, we expect a larger performance difference between the hypervisors in this test. The benchmark results in Figure 5(a) illustrate that while most of the hypervisors have relatively similar performance across the different IO types, Xen has significantly lower performance in the tests involving character level IO. We believe that the performance drop of up to 40% may be caused by higher overhead incurred by each IO request. Surprisingly, KVM beats out all competitors, even though it uses the same QEMU based backend as Xen for disk processing. Figure 5(b) shows that increasing the VCPU count and adding more disk-intensive threads in the benchmark causes a negligible performance gain for Xen, KVM, and vSphere, but surprisingly causes a dramatic decrease in throughput for Hyper-V (from 50.8 Kops/s to only 9.8 Kops/sec for the Char output test). We see this anomalous behavior as long as greater than one VCPU is assigned to the VM, perhaps indicating that Hyper-V's IO scheduler can experience thrashing when too many simultaneous IO requests are issued.

FileBench [23] is a model based file system workload generator that can mimic mail server, file server, and web server workloads. The results in Figure 6 confirm those from Bonnie++ since we again find Xen with lowest performance and also see Hyper-V's performance drop when the number of cores is increased. Xen has the worst performance under the mail server workload, which makes sense because the workload contains a large number of small disk writes. In contrast, the web workload is read intensive, closer to Bonnie++'s random-seek workload in which Xen performed well.

E. Netperf

Netperf [24] is a network performance benchmark that can be used to measure various aspects of networking features. The primary foci are bulk (aka unidirectional) data transfer and request/response performance using either TCP or UDP and the Berkeley Sockets interface. A test is based on the Netperf TCP_STREAM test with a single VCPU. Without defining MessageSize and SocketSize, the maximum throughput per second is measured. Figure 7 shows the throughput per second when using Netperf to transfer data from the test machine using TCP sockets. The network throughput of vSphere is about 22% more than that of Xen. Xen may have more overhead due to the network transmission using the network backend driver located in dom0. This requires more levels of indirection compared to other hypervisors, which in turn affects overall throughput.

F. Application Workloads

We have already demonstrated the Linux kernel compile performance in Figure 1. This illustrated a performance gap between Xen and the other hypervisors as high as 40%. From our component based benchmarks, this difference makes sense and is probably a result of Xen's moderate disk overheads. We test a larger set of applications using the freebench software [25]. As shown in Table II, freebench is an open-source multi-platform benchmarking suite, providing game,

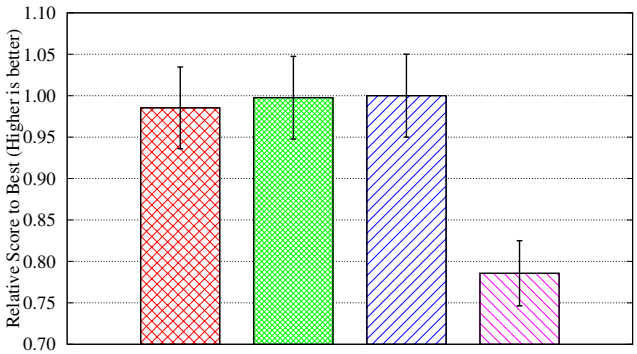


Fig. 7. Netperf Benchmark (CI = 95%)

audio encoding, compression, scientific, HTML processing, photo procesing, encryption, and decompression workloads.

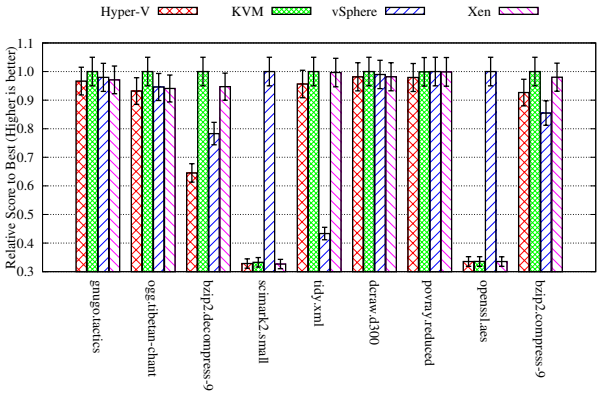
Figure 8(a) shows the benchmark results for 1 VCPU case, while Figure 8(b) shows results with 4VCPUs. The results show significant inconsistencies—the bzip2.decompress, scimark, tidy, and openssl benchmarks all have dramatically different results depending on the hypervisor and number of cores used. While the bzip benchmark includes a modest amount of IO, the others are all dominated by CPU and memory activity. This makes the variability we observe all the more surprising since the CPU and memory benchmarks indicated a relatively consistent ranking of hypervisor speed.

G. Multi-Tenant Interference

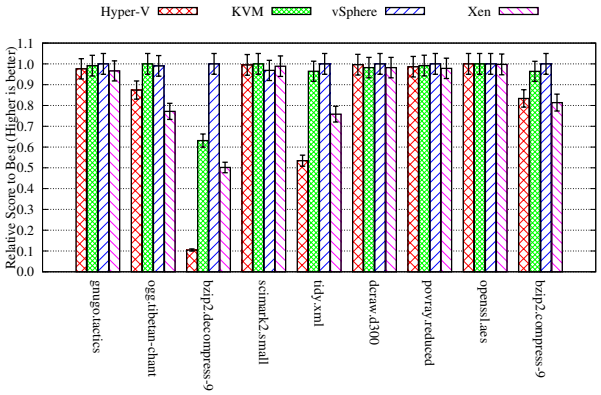
While our previous tests have only considered a single VM running in isolation, it is far more common for each server to run multiple VMs simultaneously. As virtualization platforms attempt to minimize the interference between these VMs, multiplexing inevitably leads to some level of resource contention. That is, if there is more than one virtual machine which tries to use the same hardware resource, the performance of one virtual machine can be affected by other virtual machines. Even though the schedulers in hypervisors mainly isolate each virtual machine within the amount of assigned hardware resources, interference still remains in most of hypervisors [7–10].

As the web server is a popular service in cloud infrastructures, we want to see how its performance changes when other VMs run applications on the same host. In order to see the impact of each component, CPU, Memory, Disk, and network, we measure the HTTP response while stressing each of the resource components with different benchmarks.

Figure 9 shows the impact of interference in each hypervisor. There are four VMs: one VM runs a simple web service being accessed by a client, and the other three are used for interference generators. The experiment is divided into four phases: first a CPU based benchmark is run, followed by memory, disk, and finally a network intensive application. During each phase, all three interfering VMs run the same benchmark workload and we measure the performance impact on the web VM. Note that due to benchmark timing constraints, the start and end of some phases have short periods where no interfering VMs are running. With no interference, all hypervisors have a base web response time of approximately 775 ms.



(a) 1 VCPU Case



(b) 4 VCPUs case

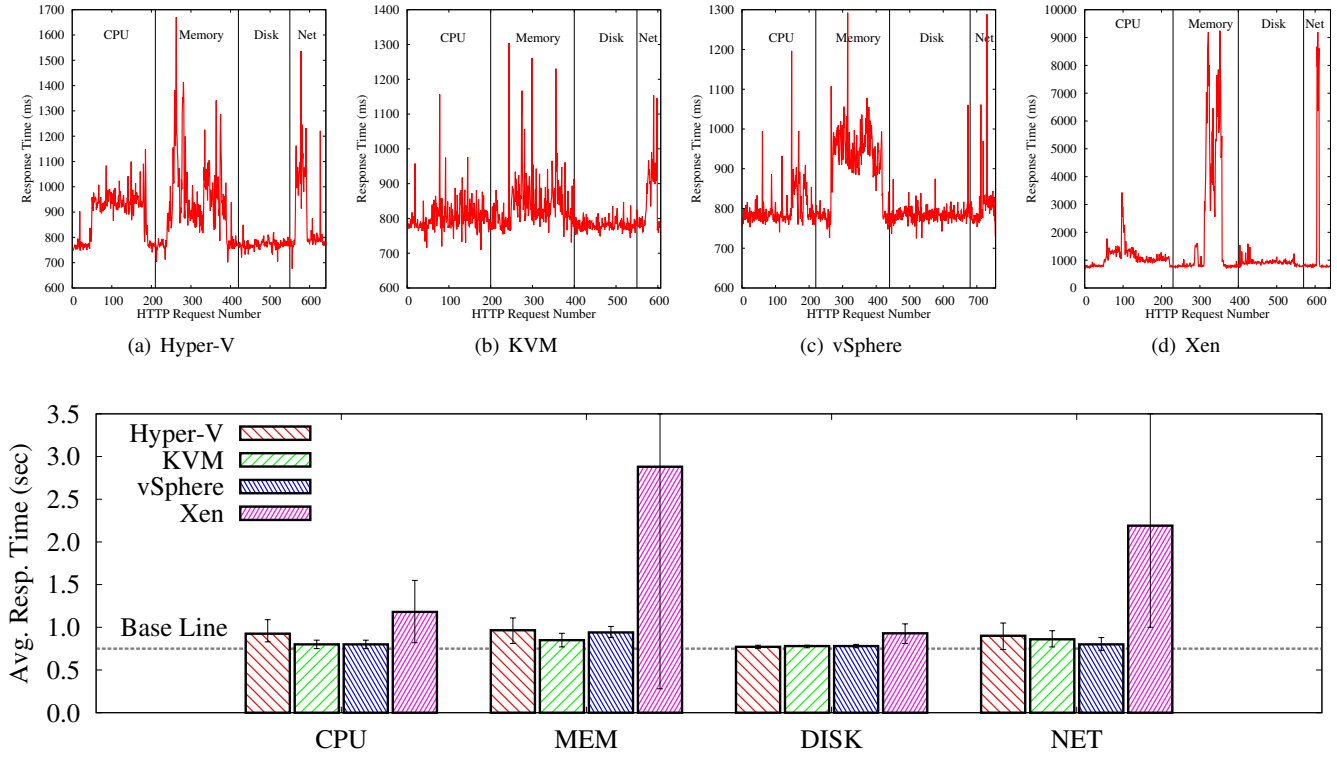
Fig. 8. Freebench Benchmark Result (CI = 95%)

Figure 9(a) illustrates Hyper-V is sensitive to CPU, memory, and network interference. Not surprisingly, the interfering disk benchmarks have little impact on the web server since it is able to easily cache the files it is serving in memory. Figure 9(b) shows the interference sensitivity of KVM; while KVM shows a high degree of variability in response time, none of the interfering benchmarks significantly hurt performance. Figure 9(c) shows the interference sensitivity of vSphere to memory is high, whereas the sensitivity to CPU, disk, and network is very small. Finally, Figure 9(d) shows the interference sensitivity of Xen on memory and network is extremely high compared to the other hypervisors.

Figure 9(e) shows the direct comparison of four hypervisors. A base line shows the average response time without using hypervisors. As we also can see from Figure 7, Xen has inherent network overheads so that other workloads affect more on the application performance.

V. Discussion

Our experimental results paint a complicated picture about the relative performance of different hypervisors. Clearly, there is no perfect hypervisor that is always the best choice; different applications will benefit from different hypervisors depending on their performance needs and the precise features they require. Overall, vSphere performs the best in our tests, not



(a) Hyper-V (b) KVM (c) vSphere (d) Xen

(e) Performance Comparison

Fig. 9. Interference Impact for Web Requests: 4 VMs (1 web server, 3 workload generators) are used. 3 VMs run the same workload at the same time. The workloads run in the sequence of CPU, memory, disk, and network workloads over time span. We can easily identify 4 interference sections from each graph.

TABLE II. FREEBENCH BENCHMARKS

Items	Category	Notes
gnugo.tactics	Game	GNU Go is a free program that plays the game of Go
ogg.tibetan-chant	Audio Encoding	Encoding a song, Tibetan Chant
bzip2.decompress-9	Compression	Compress a file in bzip2
scimark2.small	Scientific	SciMark 2.0 is a Java benchmark for scientific and numerical computing. It measures several computational kernels and reports a composite score in approximate Mflops/s
tidy.xml	HTML processing tool	HTML syntax checker and reformatter
dcraw.d300	Photo Rendering	Dcraw is a photo rendering tool, and become a standard tool within and without the Open Source world
povray.reduced	Persistence of Vision Ray-Tracer	Persistence of Vision Ray-Tracer creates three-dimensional, photo-realistic images using a rendering technique called ray-tracing
openssl.aes	Encryption	OpenSSL AES encryption
bzip2.compress-9	Decompression	Decompress a file in bzip2

surprisingly since VMware's products have been the longest in development and have the largest group of dedicated developers behind them. However, the other three hypervisors all perform respectably, and each of the tested hypervisors has at least one benchmark for which it outperforms all of the others.

In general, we find that CPU and memory related tasks experience the lowest levels of overhead, although KVM experiences higher memory overheads when all of the system's cores are active. Performance diverges more strongly for IO activities, where Xen exhibits high overheads when performing small disk operations. Hyper-V also experiences a dramatic slowdown when multiple cores are dedicated to running small, sequential reads and writes. Xen also suffers in network throughput. It is worth noting that we test Xen using hardware-assisted full virtualization, whereas the hypervisor was originally developed for paravirtualization. In practice, public clouds such as Amazon EC2 use Xen in paravirtualized mode for all but their high-end instance types.

Our application level tests match these results, with different hypervisors exhibiting different overheads depending on the application and the number of cores assigned to them. All of this variation suggests that properly matching an application to the right hypervisor is difficult, but may well be worth the effort since performance variation can be as high as 140%. We believe that future management systems should be designed to exploit this diversity. To do so, researchers must overcome the inherent challenges in managing multiple systems with different APIs, and the difficulty in determining what hypervisor best matches an application's needs. VM interference also remains a challenge for all of the hypervisors tested, and is another area where properly designed management systems may be able to help.

While we have taken every effort to configure the physical systems and VMs running on them identically, it is true that the performance of each hypervisor can vary significantly depending on how it is configured. However, this implies that there may be even greater potential for variability between

hypervisors if they are configured away from their default settings. Thus the goal of our work is not to definitively show one hypervisor to be better than the others, but to show that each have their own strengths and weaknesses.

VI. Conclusion

In this paper we have extensively compared four hypervisors: Hyper-V, KVM, vSphere, and Xen. We show their performance differences and similarities in a variety of situations. Our results indicate that there is no perfect hypervisor, and that different workloads may be best suited for different hypervisors. We believe that the results of our study demonstrate the benefits of building highly heterogeneous data center and cloud environments that support a variety of virtualization and hardware platforms. While this has the potential to improve efficiency, it also will introduce a number of new management challenges so that system administrators and automated systems can properly make use of this diversity. Our results also illustrate how competing VMs can have a high degree of performance interference. Properly determining how to place and allocate resources to virtual servers will remain an important management challenge due to the shared nature of virtualization environments. Our future research is to solve these problems.

References

[1] Vijayaraghavan Soundararajan and Kinshuk Govil, "Challenges in building scalable virtualized datacenter management," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 4, pp. 95–102, Dec. 2010.

[2] Luiz André Barroso and Urs Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[3] Timothy Wood, Ludmila Cherkasova, Kivanc Ozonat, and Prashant Shenoy, "Profiling and modeling resource usage of virtualized applications," in *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, New York, NY, USA, 2008, Middleware '08, pp. 366–387, Springer-Verlag New York, Inc.

[4] Ludmila Cherkasova, Diwaker Gupta, and Amin Vahdat, "Comparison of the three cpu schedulers in xen," *SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 2, pp. 42–51, Sept. 2007.

[5] Diego Ongaro, Alan L. Cox, and Scott Rixner, "Scheduling i/o in virtual machine monitors," in *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, New York, NY, USA, 2008, VEE '08, pp. 1–10, ACM.

[6] Google Compute Engine, "http://cloud.google.com/compute," 2012.

[7] Melanie Kambadur, Tipp Moseley, Rick Hank, and Martha A. Kim, "Measuring interference between live datacenter applications," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Los Alamitos, CA, USA, 2012, SC '12, pp. 51:1–51:12, IEEE Computer Society Press.

[8] Jason Mars, Neil Vachharajani, Robert Hundt, and Mary Lou Soffa, "Contention aware execution: online contention detection and response," in *Proceedings of the 8th annual IEEE/ACM international symposium on Code generation and optimization*, New York, NY, USA, 2010, CGO '10, pp. 257–265, ACM.

[9] Jason Mars, Lingjia Tang, and Mary Lou Soffa, "Directly characterizing cross core interference through contention synthesis," in *Proceedings of the 6th International Conference on High Performance and Embedded Architectures and Compilers*, New York, NY, USA, 2011, HiPEAC '11, pp. 167–176, ACM.

[10] Gang Ren, Eric Tune, Tipp Moseley, Yixin Shi, Silvius Rus, and Robert Hundt, "Google-wide profiling: A continuous profiling infrastructure for data centers," *IEEE Micro*, vol. 30, no. 4, pp. 65–79, July 2010.

[11] Stuart Devenish, Ingo Dimmer, Rafael Folco, Mark Roy, Stephane Saleur, Oliver Stadler, and Naoya Takizawa, "Ibm powervm virtualization introduction and configuration," *Redbooks*, 1999.

[12] VMware, "Understanding full virtualization, paravirtualization, and hardware assist," *VMware White Paper*, 2007.

[13] 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," *ACM SOSP*, 2003.

[14] Padma Apparao, Srihari Makineni, and Don Newell, "Characterization of network processing overheads in xen," in *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, Washington, DC, USA, 2006, VTDC '06, pp. 2–, IEEE Computer Society.

[15] Aravind Menon, Jose Renato Santos, Yoshio Turner, G. (John) Janakiraman, and Willy Zwaenepoel, "Diagnosing performance overheads in the xen virtual machine environment," in *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, New York, NY, USA, 2005, VEE '05, pp. 13–23, ACM.

[16] Jinho Hwang and Timothy Wood, "Adaptive dynamic priority scheduling for virtual desktop infrastructures.," in *IWQoS*. 2012, pp. 1–9, IEEE.

[17] VMware, "A performance comparison of hypervisors," *VMware White Paper*, 2007.

[18] Vmware vs Virtualbox vs KVM vs XEN, "http://www.ilsistemista.net/index.php/virtualization/1-virtual-machines-performance-comparison.html," 2010.

[19] Sunay Tripathi, Nicolas Droux, Thirumalai Srinivasan, and Kais Belgaied, "Crossbow: from hardware virtualized nics to virtualized networks," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, New York, NY, USA, 2009, VISA '09, pp. 53–62, ACM.

[20] Bytemark, "http://www.tux.org/ mayer/linux/bmark.html," 2003.

[21] Ramspeed, "http://www.alasir.com/software/ramspeed," 2009.

[22] Bonnie++, "http://www.textuality.com/bonnie," 2004.

[23] Filebench, "http://sourceforge.net/projects/filebench," 2004.

[24] Netperf, "http://www.netperf.org/netperf," 2012.

[25] Freebench, "http://code.google.com/p/freebench," 2008.