Benchmarking a Virtualization Platform

Vijayaraghavan Soundararajan Banit Agrawal Bruce Herndon Priya Sethuraman Reza Taheri
R&D Performance, VMware Inc.
Email: {*ravi,banit,bherndon,psethuraman,rtaheri*}*@vmware.com*

Abstract— **Traditional benchmarking of new architectures often involves running a workload on a single system with a single OS. In such a setup, the objective is typically to stress a single resource (e.g., CPU) and produce a single number used to characterize the performance of the system. Newer benchmarks have extended this paradigm by testing the performance of distributed systems like Hadoop clusters or cloud-style workloads such as big data analytics. These benchmarks are invaluable for evaluating physical systems, but have numerous drawbacks and gaps when used to evaluate virtualized systems. A virtualized system must be evaluated in an end-to-end way beyond what is done for physical systems, with measurements and workloads for the hypervisor platform, the application, and the management layer that supports virtualization-related services like live migration or rapid VM provisioning.**

In this paper, we describe an end-to-end approach to evaluating virtualization platforms. We begin with a discussion of traditional virtualization benchmarking, involving comparing performance on real systems vs. performance on virtualized systems. We next describe three benchmarks that we have developed specifically for virtualized environments. We discuss a system-level benchmark, VMmark, which incorporates workloads that simultaneously stress a number of different resources (CPU, memory, and IO). We then outline two additional benchmark suites designed for measuring virtualization management performance and application performance: VCBench and ViewPlanner. For each workload, we describe how it was developed, what it tests, and the key insights it has provided in terms of optimizing virtualized platforms. We conclude with a discussion of next-generation virtualization benchmarks.

Keywords— Virtual Machine management, cloud computing, datacenter management, management workload, benchmarking

I. INTRODUCTION

Virtualization has become the backbone of enterprise datacenters as well as cloud computing infrastructures. As a result, virtualization is increasingly serving as the backend for not only enterprise use cases, but also consumer-facing applications [37] and even high-performance computing [38]. Moreover, previous work has shown [27,28] that the value in virtualization is not just in server consolidation, but in the wide range of services built on top of virtualization for easing system management, for example, live migration, automated load balancing, snapshots, and disaster recovery. These services have also made applications like virtual desktop infrastructure practical by allowing efficient centralized management of desktops. The end result is that performance of a virtualized system is not just the performance of the hypervisor or the application, but also the performance of the management piece.

In this paper, we present a set of benchmarks designed to test the entire virtualization stack. We argue that benchmarking a virtualization platform is very different from benchmarking a physical system. In a physical system running conventional applications, micro-architectural components like caches, TLBs, and NUMA controllers are stressed. In a virtualized environment, however, the resource contention that occurs is exacerbated because of the increased number of Virtual Machines (VMs) competing for resources on the hardware. Moreover, as features are added to improve performance, if these features cannot be exposed to the end user or to the administrator with reasonable performance, then the value of the feature is diminished. In essence, we argue that benchmarking is required at the hypervisor layer, the application layer, and the management layer.

The contributions of this paper are as follows. First, we discuss the use of heavy-duty Tier-1 applications, in particular, database transaction processing, to stress all system components at once and identify bottlenecks and tuning opportunities. We then describe the design of three benchmark suites specifically intended for measuring the performance of virtualized systems: VMmark, which measures hypervisor and consolidation performance; ViewPlanner, which measures interactive performance of desktop applications in a virtualized environment; and VCBench, which measures the performance of the management layer. We then describe some examples of how they have been used in virtualization environments and what sorts of insights we have gleaned from them. We then briefly describe how these workloads can be used in assessing next generation use cases and applications.

The structure of this paper is as follows. In section II, we give background on virtualization platforms and the terminology used in this paper. In section III, we describe the design of our benchmarks. In section IV, we show some of the

978-1-4799-6454-3/14/$31.00 ©2014 IEEE

insights we have gained while running these benchmarks on real systems. In section V, we briefly describe related work. We conclude in section VI.

II. VIRTUALIZATION PLATFORM ARCHITECTURE

For this paper, we consider the virtualization platform to consist of all of the components from the physical hardware up to the virtualization management controller. The hypervisor runs directly on physical hardware and abstracts physical details from the VMs running on top of it. Applications run inside the "guest" OS within the VMs. A virtualization management controller is responsible for monitoring the status of the hardware, hypervisors, and the VMs and for issuing user operations (like powering on VMs) to the hypervisors and VMs. The management server is also responsible for other services such as automated load balancing and high availability by moving VMs across hosts according to policies. In order to perform operations on VMs, management agents run on top of the hypervisor. These agents take commands from the management controller and perform them on the appropriate VMs, or communicate between hypervisors for cross-hypervisor operations like load balancing. More details are available in Soundararajan, *et al* [27].

As previous work has shown, for virtualized environments, performing higher-level services like snapshots, disaster recovery, or load balancing can often be a workload over and above the applications running within the VMs themselves [27,28]. Thus, to truly evaluate the effectiveness of a virtualization platform, we must measure the performance of the VMs as well as the management operations.

In this paper, we use VMware vSphere [32] as our virtualization platform, consisting of the ESX hypervisor and the vCenter management controller. We also utilize VMware View as our VDI deployment [41]. While some of the benchmarks we describe are written for vSphere using publicly available APIs, they can be adapted to other platforms in a straightforward manner.

III. BENCHMARK DESCRIPTIONS

In this section, we discuss the rationale and design of the benchmarks we use for evaluating virtualized platforms. We first describe a commercially available benchmark that represents a typical Tier-1 application workload. Next, we describe three benchmark suites that we have developed specifically for virtualized systems: VMmark, to measure hypervisor performance; VCBench, to measure virtualization management performance; and ViewPlanner, to measure end-user responsiveness in virtual desktop infrastructure.

A. Tier-1 Applications: virtual vs. native

When evaluating a virtualization platform, the first goal is typically to make sure that an individual application running on virtual hardware performs as well as when it runs on physical hardware, the so-called virtual vs. native comparison. Tier-1, or mission-critical, applications pose a special challenge to virtualized servers in that these applications have strict performance requirements, both in throughput and response time, compounded by the fact that they stress all components of the system: CPU, memory, storage, networking, contention management routines, etc. One cannot, for example, avoid a storage bottleneck by allocating more memory and caching more of the dataset, since memory itself is a resource under pressure.

Examples of Tier-1 applications are databases, CRM, ERP, and corporate collaboration/email. For this paper we limit ourselves to database on-line transaction processing (OLTP) applications, which exhibit high CPU usage, heavy storage load in terms of IOPS, extensive use of the memory for caching disk blocks, and a moderate level of networking I/O. OLTP applications have a high operating system content, so if the hypervisor runs kernel code slower than it runs user code, this is exposed. They also make heavy use of the contention management constructs such as spinlocks, making them a great tool for studying the hypervisor scheduler. To flesh out bottlenecks, we intentionally run the benchmark in a configuration that puts a demanding load on the system.

B. VMmark

The goal of VMmark is to create a meaningful measurement of virtualization consolidation across a wide range of hardware platforms. It consists of application workloads and infrastructure workloads. For the purposes of this paper, we focus on the application workload component. Eight VMs comprise the application workload, including a web2.0 application as well as popular multi-tier workloads like email and databases (see Table 1). The applications are based on surveying virtualization usage patterns across a wide range of enterprises. An idle VM is included as a proxy for easily-virtualized, lightly-loaded VMs. Although any single workload may only stress a subset of the underlying hardware resources (e.g., CPU and memory), the full collection of workloads stresses the resources in a more comprehensive way by exercising all components of the system when run together.

TABLE 1: VMMARK APPLICATION WORKLOADS

Workload Type	**Workload Generator**	**OS**
Database Multi-tier	DVDStore2/ MySQL, Apache	4VMs SUSE SLES 11
Web 2.0/App	OLIO/MySQL, Tomcat	2 VMs SUSE SLES 11
Email Server	MS Exchange/ Loadgen	MS Windows Server 2008
Idle System	Idle System	MS Windows Server 2008

One intent of VMmark is to focus measurements on the performance of the hypervisor and the underlying hardware, rather than measure the effects of application or OS level tunings. Thus, we strictly limit the application and operating system tunings allowed to prevent software level tunings from obscuring the characterization of the underlying platform. This is a departure from commonly-used benchmarks from industry-sponsored consortia such as SPEC [29,30].

The unit of scalability for VMmark is defined as the collection of eight virtual machines and is referred to as a *tile*. The total number of tiles that a physical system can accommodate gives a coarse-grained measure of that system's consolidation capacity. Tiles are relatively heavyweight objects that cannot by themselves capture small variations in system performance. Each workload within a tile is constrained to execute at less than full utilization of its VM. However, the performance of each workload can vary to a degree with the speed and capabilities of the underlying system. For instance, disk–centric workloads might respond to the addition of a fast disk array with a more favorable score. These variations can capture system improvements that do not warrant the addition of another tile. However, the workload throttling will force the use of additional tiles for large jumps in system performance. When a tile is added, workloads in existing tiles might measure lower performance (required to be above an SLA), but the aggregate score including the new tile should increase if the system has not been overcommitted. The result is a flexible benchmark metric that provides a measure of the total number of workloads that a particular system can support as well as the overall performance level within the VMs.

Workload metrics of each tile are computed and aggregated into a score for that tile. This aggregation is performed by first normalizing the different performance metrics such as MB/s and database commits/s with respect to a reference system. Then, a geometric mean of the normalized scores is computed as the final score for the tile. The resulting per-tile scores are then summed to create the final metric. The geometric mean is chosen in part to model similar metrics in other SPEC benchmarks [29][30].

C. VCBench

VCBench is intended to model the management activities of enterprise customers and cloud providers. For example, for cloud use cases, it is important to be able to provision a large number of VMs in a short period of time; this requires tuning host performance as well as tuning the management server doing the orchestration. We have tried with VCBench to create a workload that accurately represents the types of actions performed in virtualized environments and also acts as a stress test to determine the maximum capability of the management stack. VCBench is not a cloud benchmark *per se*: instead, it implements operations that a cloud manager like OpenStack [24] or vCloud Director [40] might call via APIs, so a cloud management benchmark could be built on top of it.

VCBench generates load by attaching a set of vSphere API [43] clients to the virtualization manager (in this case, the vCenter server). VCBench is configurable in terms of operation mix and load level, but is typically used in two primary modes: the first mode is a so-called light load, in which we issue approximately 100 management operations per minute. This is more than most datacenters perform, but allows us to anticipate realistic near-term increased demand. Moreover, it allows us to evaluate the performance of the API itself, of increasing importance as API-driven automation within datacenters becomes more prevalent. With more management operations, the entire stack is exercised, from the management server down to the hardware running the management agents. In addition to the light load, we also utilize a heavy load of over 500 operations per minute to try to saturate the management server. VCBench can be configured to run for any length of time, but we have found that running it in alternating light and heavy loads over a period of 12 hours provides sufficient data to assess management layer performance.

To measure performance, we collect the average latency and throughput of each operation, the 95th percentile latency and throughput for each operation, and the overall latency and throughput. For measuring the resource efficiency of the management stack, we also collect resource usage statistics. The overall throughput number represents the VCBench score, while the other metrics are helpful for performance debugging.

In order to faithfully represent operations that customers would do in practice, our operation mix and the frequency of operations is based on data collected from numerous virtualized datacenters. The most common operations along with their relative frequencies in the virtualized datacenter are shown in . We are continuously updating these ratios and operations in the presence of new features and changing datacenter use cases. For example, in recent years, automated provisioning through self-service portals has become more popular, so provisioning operations like cloning have increased in frequency. The operation mix and frequency can be varied simply by editing a run list.

TABLE 2: VCBENCH TYPICAL OPERATIONS AND RELATIVE FREQUENCIES.

Operation	**Relative frequency**	**Operations/minute (light)**
Power on VM	8	40
Power off VM	8	40
Clone VM	2	10
Migrate VM	8	40
Remove VM	2	10
Create Snapshot	1	5
Delete Snapshot	1	5
Reconfigure VM	4	10

D. ViewPlanner

In virtual desktop infrastructure (VDI), per-user desktops are replaced with thin clients that connect to backend servers where the desktop VMs are hosted. A remote display protocol transfers keystrokes and mouse clicks from the user to the VM, and then transfers display updates back to the thin client. The goal of the ViewPlanner benchmark is determine if a user is getting a "local desktop experience" from a remote VM, despite the fact that the VM is remote and is presumably sharing hardware resources with VMs from other users. Achieving this goal in a virtualized environment is feasible for two reasons: first, desktop applications are not usually CPU-bound and are typically hosted on machines with much faster server-class CPUs rather than standard desktop CPUs. Second, the backend servers are typically using faster, enterprise class storage (like SSDs with sub-millisecond latencies rather than inexpensive 5400 RPM drives). An end-to-end VDI deployment also includes a piece that brokers VM communication between end-users and the virtualization manager, and this provides another test of the virtualization management subsystem. In this paper, we focus on the end-user application component of ViewPlanner which is freely available to download at the reference link [47].

ViewPlanner simulates the operations of typical office users running commonly-used business applications such as the Microsoft Office suite, Adobe Reader, Firefox, Windows Media player, and a compression utility. The full list is shown in Table 3. Within these applications, common desktop user operations are performed, including opening, saving, and closing files, minimizing and maximizing windows, browsing the web and reading email. ViewPlanner deploys VMs that act as clients. These clients interact with desktop VMs and collect interactive performance data, which is uploaded and synthesized into a Quality-of-Service (QoS) metric by the ViewPlanner controller. A "think time" is inserted between operations to better mirror realistic use patterns. To achieve repeatability while simulating the varying usage patterns across users, each user performs the same actions within an application, but the order of applications per user is randomized.

1) Design challenges

We encountered two main challenges when designing ViewPlanner: measuring timing accurately within the guest, and developing an appropriate QoS metric.

Accurate Timing. For timekeeping inside the guest, timer interrupts are delivered to the virtual desktop at a pre-specified rate. In a highly-consolidated environment, clock skew can occur if timer interrupts are not delivered in time by the hypervisor or if the timer interrupts are not processed by the guest in time; thus we cannot rely on guest timers to keep track of wall clock time. Instead, we make use of virtualized hardware performance counters using rdpmc [33]. These counters are exposed to the guest by the hypervisor and provide a true sense of the elapsed time per VM. The overhead of these calls is about 50 microseconds (150K cycles), minimal enough for an interactive workload.

TABLE 3: VIEWPLANNER APPLICATIONS AND THEIR OPERATIONS

APPLICATION	OPERATIONS
FIREFOX	["OPEN", "CLOSE"]
EXCEL	["OPEN", "COMPUTE", "SAVE", "CLOSE", "MINIMIZE", "MAXIMIZE", "ENTRY"]
WORD	["OPEN", "MODIFY", "SAVE", "CLOSE", "MINIMIZE", "MAXIMIZE"]
ADOBEREADER	["OPEN", "BROWSE", "CLOSE", "MINIMIZE", "MAXIMIZE"]
IE_APACHEDOC	["OPEN", "BROWSE", "CLOSE"]
POWERPOINT	["OPEN", "RUNSLIDESHOW", "MODIFYSLIDES", "APPENDSLIDES", "SAVEAS", "CLOSE", "MINIMIZE", "MAXIMIZE"]
OUTLOOK	["OPEN", "READ", "RESTORE", "CLOSE", "MINIMIZE", "MAXIMIZE", "ATTACHMENT-SAVE"]
7ZIP	["COMPRESS"]
VIDEO	["OPEN", "PLAY", "CLOSE"]
WEBALBUM	["OPEN", "BROWSE", "CLOSE"]

Another aspect of timing is making sure to capture user-experience latency accurately. For this, we focus on the display protocol, which is critical for good user experience. It must be adaptive and aware of the underlying network conditions to provide similar user experience irrespective of network latency and bandwidth. The challenge to client-side benchmarking is that the end-user only receives display updates when a particular workload operation completes, and we must make use of the same display channel to detect the start and end of the operation and measure the latency; if we use a different channel, it may get out of sync with the display channel. Our approach is to place a watermark in a non-interfering/disjoint location that does not interfere with the application updates. This watermark is transmitted along with screen data to the client, and ViewPlanner can detect it and use it for timing.

QoS metric development. For our QoS metric, we consider 95% latency thresholds. To avoid the QoS metric being dominated by short-running operations, we divide all View Planner operations into two main groups: Group A, which consists of very fast running or interactive operations; and Group B, with relatively slower operations such as open/save file. To meet satisfactory user expectations, we define that for Group A, 95% should finish within T1 seconds, and for Group-B, 95% should finish in T2 seconds. We have performed several user studies to find when the user expectations are satisfactorily met for Group-A & Group-B

operations, and our empirically chosen values for T1 and T2 are one second and six seconds, respectively.

IV. SAMPLE BENCHMARK USE CASES

In this section, we give a few simple examples of how we have used these benchmarks to understand tradeoffs in system design. We start first with the OLTP and then move on the benchmarks that we designed to assess the entire platform.

A. OLTP: virtual vs. native

Initial virtualization studies focused on the gap between virtual machine performance to native performance ratio, and helped drive improvements such as CPU virtualization extensions [1] and extended or nested page tables for MMU virtualization [3]. The *de facto* performance expectation, therefore, is that virtual machines perform nearly on par with physical. The goal of our OLTP workload is to assist with such a comparison by assessing single VM performance. As pointed out in section III.A, we will focus on OLTP applications as the representative of Tier 1 applications using a fair-use version of TPC-C. We run our tests on a single VM that had as many virtual CPUs as the number of physical CPUs and used up all the memory exposed by the hypervisor.

The server is a 4-socket system with Intel E7-4870 processors and 512GB of memory. In order to separate basic performance overheads from any possible SMP scaling issues on the hypervisor, we run most of our tests with only 2 cores active per socket and hyperthreading disabled. Storage consists of two EMC CX4-960 arrays with 62 SSD drives, capable of at least 260K IOPS. The guest operating system is RHEL 6.1, and the DBMS is Oracle 11gR1. We use a benchmark based on the TPC-C workload, although the configuration did not conform to the full TPC-C specification, and our results should not be compared to published results; nor should they be considered indicative of absolute performance of Oracle DBMS. We increase the load to the point of CPU saturation, as is common in this type of benchmarking [22]. We show the transaction rate, number of users, ratio to native, and IOPS in Table 4. At CPU saturation, the OLTP workload is performing at around 80% of native.

TABLE 4: PERFORMANCE METRICS FOR OLTP BENCHMARK

Metric	Native	ESX 4.1	ESX 5.0
Users	125	160	160
Trans/sec	10.6K	8.4K	8.4K
Ratio to Native	1.0	0.8	0.8
IOPS	69K	54K	54K
Idle CPU left	1.4%	2.6%	1.8%

We repeated the experiment with a 32-vCPU VM on 32 cores. The results in Figure 1 show that scaling continues fairly evenly up to 32 virtual CPUs and 157K IOPS. Hence, the SMP scaling of the hypervisor is near linear.

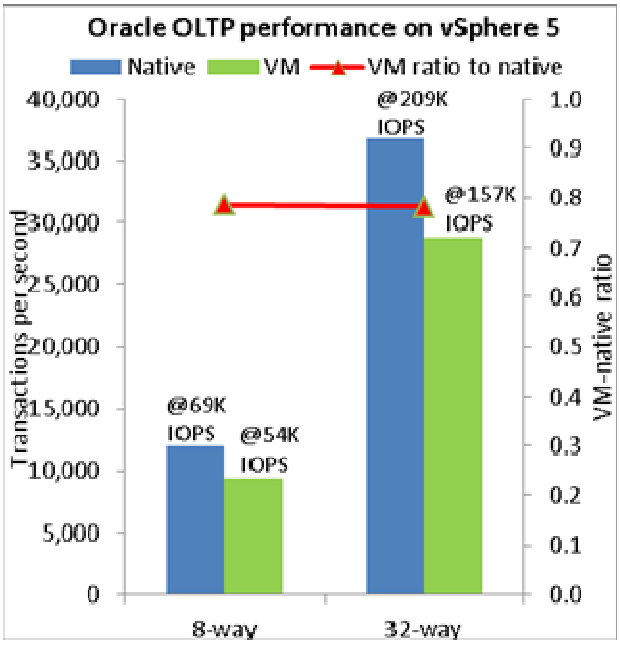

Figure 1: Near-linear scaling from 8-way to 32-way for OLTP workload.

Although a 20% overhead for the heaviest workload we know is not unacceptable, it is quite high given the advent of processor support for virtualization, and the progress made in 10 years of developing hypervisors for the server market. In the next section, we examine the sources of the 20% overhead.

1) Page address translation costs in a virtual environment

Using hardware performance counters, we are able to examine the 20% increase in cycles/transaction in more detail. Looking at the cycle breakdown, we make a number of observations.

1. The number of executed instructions per transaction in user mode in the guest is the same for native and virtual configurations. In other words, virtualization does not cause a growth in the code path in user mode.
2. Similarly, there is little change in the kernel-mode path length inside the guest. However, there is a ~5% increase in cycles/transaction attributable to time in the hypervisor. This increase makes sense: each I/O operation has to first go through the virtual device drivers inside the guest, and then be processed by the physical device drivers in the hypervisor kernel.
3. We see about 3% in the virtual machine monitor (the methodology to retrieve this is beyond the scope of this paper, but is described by Buell *et al* [9]).
4. We see an increase of 12-13% in CPI.

The most surprising observation is the large increase in CPI. We studied the detailed behavior of the hardware using processor-specific tools that collect data from built-in hardware event counters (see Table 5).

TABLE 5: HARDWARE STATISTICS FOR OLTP WORKLOAD

Hardware counter	Native	Virtual
Thread Util	99%	97%
unhalted core cycles	2,998K	3,668K
Thread CPI	2.10	2.38
Path length	1,429K	1,541K

More-detailed TLB hardware counters (not shown here) indicate that about half the growth in CPI is due to the time spent traversing the two-level paging structures during TLB miss processing. An explanation of the two-level TLB is beyond the scope of this paper [8][13]. In brief, modern X86 microprocessors provide features that make it much easier to manage the translation of the guest's linear addresses to actual machine addresses. However, this comes at the cost of having to traverse both the guest's logical-to-physical page tables and the hypervisor's physical-to-machine page tables at TLB miss time. This additional overhead accounted for about half the growth in the CPI, possibly even more once we consider the additional pressure on the hardware caches caused by the traversal of a second set of page tables.

As this brief study demonstrates, the single-VM evaluation of an enterprise workload can yield understanding of areas of improvement for current hypervisor software and virtualization hardware. We next move on to VMmark for a slightly higher-level evaluation.

B. VMmark

VMmark takes the benchmarking of a virtualization platform beyond a single-VM. It studies a situation that in some sense has no analogy in physical systems: multiple VMs running on the same base platform. As a result, the types of insights we gain are different from those of a single VM benchmark. While VMmark has been helpful in finding a variety of issues ranging from storage block alignment on a SAN to CPU scheduling, for space concerns, we discuss three experiments that help explore the value of such a benchmark: tracking scalability of hardware when running large numbers of virtual machines, measuring performance per kilowatt for virtual machines connected to high-end storage, and discovering the impact of NUMA scheduling on VM performance.

1) Tracking hardware scalability for VMs

VMmark is used to help measure hypervisor performance. As more virtualization-related features are introduced into modern processors, and as hypervisors evolve to take advantage of such features, we expect the VMmark score to track these changes. In Figure 2, we show the trend in VMmark scores over several years, reflecting improvements with different processor generations. The scores reflect an earlier version of the benchmark with a slightly different application mix, but similar results hold for the most recent version and the most recent hardware.

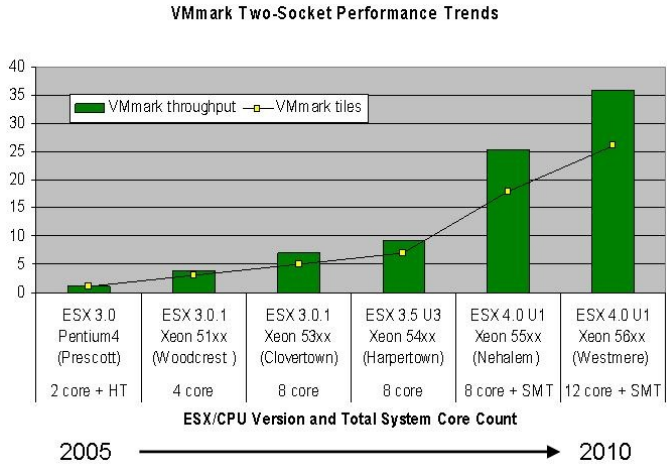

Figure 2: Evolution of VMmark scores across hardware and hypervisor generations. The big jump from 2008 to 2010 reflects the introduction of EPT support in Nehalem.

As Figure 2 shows, from 2006 until 2010, VMmark scores for typical two-socket x86 servers have increased by roughly a factor of 30. This increase has tracked and quantified the increase in processor core counts, memory capacity and speed, hardware-assist technologies for virtualization, and widespread improvements in hypervisor performance. We see a jump from an 8-core Harpertown system to an 8-core Nehalem system. This jump occurs for two reasons: first, Nehalem introduced extended-page table (EPT) support in the processor, reducing virtualization overheads; moreover, Nehalem introduced SMT, which provides an additional boost. From Nehalem to Westmere, the main change was a die shrink and increased cores per socket. The hypervisor is able to match these improvements and the result is a higher VMmark score.

2) Performance per Kilowatt

Our second sample use of VMmark involves comparing storage technologies. In addition to measuring system throughput through tile scores, VMmark also includes a facility to measure performance per kilowatt. In Figure 3 we show the results of running different numbers of application tiles with per-host local storage vs. a shared SAN. As the figure indicates, in this particular configuration, the application performance of local storage actually matches that of a SAN, while Figure 4 shows that despite similar performance, the SAN is consuming significantly more power. The tradeoff, of course, is that a SAN is likely to have features that aid the virtualization management subsystem, like native snapshots or fast cloning or virtual disks.

3) NUMA Migrations

Our final example illustrates that NUMA migrations can impact application performance in a multi-VM environment. While testing a new generation of hardware with VMmark, some tiles started underperforming and not reaching their SLAs. As a result, the aggregate VMmark score was lower for this generation of hardware than for a previous one.

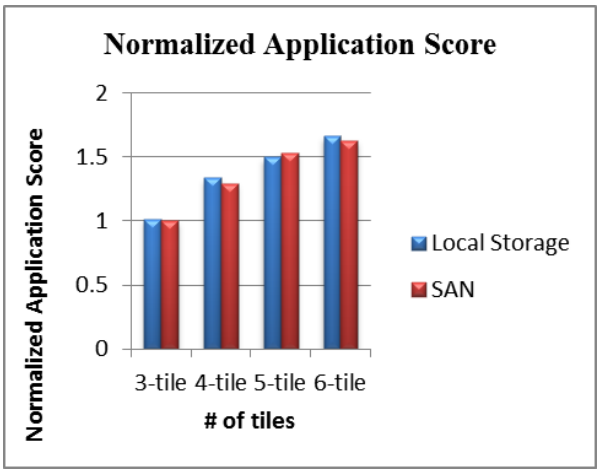

Figure 3: Normalized VMmark application score comparing Local storage vs. SAN. Local Storage provides approximately the same performance as a SAN.

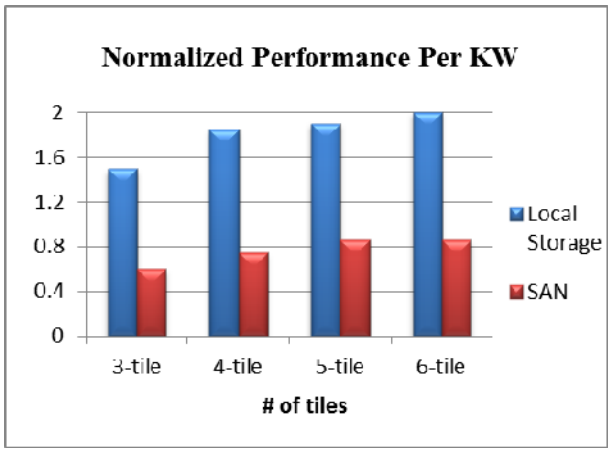

Figure 4: Performance per kilowatt, normalized to a reference architecture. While the VMmark score is the same between local storage and SAN, the power usage statistics show that SAN consumes significantly more power for the same performance.

Ultimately, the issue was traced to a hypervisor bug causing excessive NUMA page migrations, which consumed sufficient memory bandwidth that a number of workloads were getting resource-starved. A single-VM benchmark would not have exposed this behavior because there would be no NUMA migrations and no contention for memory resources among different VMs.

C. VCBench

1) Server consolidation

Our OLTP single-VM studies and VMmark studies have focused on the performance of the hypervisor. With improvements in hardware-assist technologies and corresponding improvements in support for such technologies, current CPUs are able to handle many more VMs per core, and the consolidation ratio has increased dramatically since virtualization has become mainstream.

VCBench complements these two benchmarks: while increased VM density is obviously desirable for the hypervisor, it places increased stress on the management layer in a variety of ways. For a given host, the management agents will have to collect more information and will thus require more CPU cycles. This results in more network bandwidth per host and also more update processing on the management server. More VMs per core can also result in stress on the resource management subsystem when load balancing occurs: the bin packing problem becomes more difficult.

To illustrate this point, in Table 6, we show the performance of VCBench as we change the number of VMs per host from 10 to 125, normalized to the number of hosts (since the number of hosts in the two experiments is different: 1000 in the first, and 32 in the second). We see an increase in latency for various operations like power on and clone. There are several reasons: with more densely-packed hosts, the resource management computations required to choose a good initial placement for the VM can become more complicated, since there are likely fewer free slots in which the VM can be placed. Remove VM has no such need, so its performance is nearly equivalent across setups. Second, for operations like clone, the underlying network and disk bandwidth is now shared among more operations, increasing the latency of each individual operation. In order to mitigate the effects of each of these, it is important to overprovision the IO resources and also increase the CPU and memory shares available to the management agents running on each host. A third reason for this increase is a bit more subtle. With more concurrent operations to a given host, there is more contention in the filesystem in order to create per-virtual machine swap files and file handles for tracking newly-created virtual machines. While a hypervisor is typically intended to perform mostly virtualization-related activities, these types of mainstream OS activities must also be done well for the full spectrum of virtualization-related gains to be achieved.

TABLE 6: VCBENCH OPERATION LATENCIES FOR DIFFERENT CONSOLIDATION RATIOS, NORMALIZED TO THE NUMBER OF HOSTS IN EACH ENVIRONMENT. (UNITS: SECONDS/OPERATION/# HOSTS)

Operation	**Normalized Latency/operation**	
	10 VMs/host	**125 VMs/host**
Power on VM	0.081	0.94
Power off VM	0.01	0.15
Clone VM	0.029	0.18
Migrate VM	0.047	0.59
Remove VM	0.071	0.053
Create Snapshot	0.012	0.13
Delete Snapshot	0.010	0.056
Reconfigure VM	0.021	0.19

Regarding server consolidation, there is also an interesting interplay with power management alternatives. Current power management solutions either reduce power during runtime [25] or move VMs around so that hosts can be powered off [34]. In the former case, the CPU has to be woken up from sleep states in order to process management operations, while in the latter case, the increased number of VMs per host potentially incite the same problems we have described above. Thus, the expected rate of virtualization management operations may influence the choice of power management policy.

2) Impact of fast storage

To again illustrate the complexity of tradeoffs in a virtualized environment, consider the VMmark results for using local storage vs. SAN. While the VMmark scores are very similar, the performance per watt is significantly worse for the SAN case, suggesting that local storage is a better choice. However, if the SAN supports native snapshot or native clone capabilities [42], then management-related provisioning activities are likely to be quite a bit faster for the SAN vs. local storage. To illustrate this point, we run VCbench using a simulated high-bandwidth storage device on a host (to mimic a high-speed SAN) vs. a realistic local storage device. Not surprisingly, provisioning time is significantly improved in the high-bandwidth storage case (2.5x). These gains are for VMs with small (4GB) disks—with larger disks, the expected gains may be larger. Thus, depending on how frequently VMs need to be created, destroyed, or snapshotted, perhaps the SAN is better choice. Again, it is matter of assessing overall system price, power, and performance, and not just hypervisor performance.

D. ViewPlanner

By focusing on end-user performance in a multi-VM environment, ViewPlanner complements VMmark and has revealed a number of interesting insights with respect to optimizing performance on virtualization hardware.

1) CPU Virtualization Modes

Similar to previous work [1,2], we have found situations in which software virtualization primitives outperformed hardware. In one case, we noticed that we spent significantly more CPU cycles in Microsoft PowerPoint application on virtual hardware than on physical hardware. This was clearly a case in which there was a substantial penalty due to the virtualization overhead. We compared performance while running the guest VM in binary translation (BT) [1] mode vs. hardware-assist. We found that when we used BT, we did not see the high CPU usage, suggesting that we were incurring some overhead in the virtualization layer. Upon detailed hypervisor level profiling, we found that in WinXP VMs, task priority register (TPR) accesses are frequent, and this incurs a significant penalty at the virtualization layer. BT mode optimizes such accesses out. Thus, for this application in a Windows XP VM, BT provided better performance than hardware-assisted virtualization.

2) Dynamic Virtual Interrupt Coalescing

In some cases, using ViewPlanner, we observe that a guest OS does not take advantage of a hypervisor optimization. One example is with dynamic interrupt coalescing. When running 1080p HD video and a custom 3D graphics workload at high frame rates, we noticed a performance issue: at the client side, we saw 25 fps (acceptable user experience) for some period of time, and then we saw a sudden drop down to 10 fps (unacceptable user experience). This bimodal behavior continued indefinitely. Looking further, we found that this performance issue is related to interrupt coalescing.

Virtual interrupt coalescing is useful in improving CPU efficiency for high throughput workloads because the guest VM receives a set of interrupts instead of receiving individual interrupts, thus avoiding a significant penalty on every interrupt in order to exit from kernel to user world. While this is a very useful technique to avoid extra CPU costs for both network and IO packet processing, we have found that out-of-the-box, Windows does not benefit from interrupt coalescing in some scenarios. After sending a packet, Windows waits for a completion interrupt to be delivered before sending the next packet. The hypervisor will send the transmit completion interrupt to the guest under two conditions: either when a set number of packets have been sent due to the interrupt coalescing optimization, or when there is just one packet and a timeout has been reached. These two scenarios were occurring (the latter due to large packets), thus explaining the bimodal packet distribution. To fix this, we modify a registry setting in Windows which instructs Windows not to wait for the completion interrupt even when sending larger packets. In this way, we allow the guest OS to not block the packet transmit. As a result, the VM is able to take advantage of interrupt coalescing, and the CPU load is reduced accordingly, resulting in significantly improved video playback performance.

3) Virtual Platform Scaling

An important question in VDI is how to maximize the number of virtual desktops without degrading the user experience. To answer this question, we run ViewPlanner while increasing consolidation from 4 single-CPU VM desktops per core to 8 desktops per core. We use the Intel Sandy Bridge processor (dual socket) with 8 CPU cores at 3.3 GHz frequency and then measured the response time of the operations. We then calculate the Group A 95% and Group B 95% response times. The results are shown in Figure 3. We see that response time slowly increases as we increase the number of VMs per core. At 7 VMs/core, the Group-A 95% response time is failing the QoS criteria by a slight margin, while Group-B is passing the six-second threshold criteria. We thus conclude that for this CPU architecture, we can run nearly 7 users per core while maintaining an acceptable user experience.

V. RELATED WORK

Benchmarking a virtualized system is different from a physical system in part because of the multitude of workloads

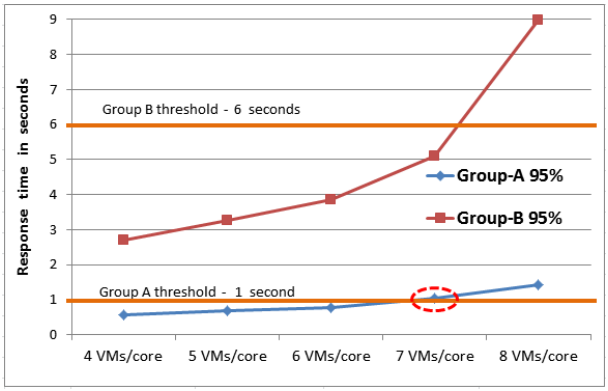

Figure 3: Group-A and Group-B 95% response time as the VMs/core are increased on a Sandy Bridge system.

that run on a single physical platform. Simply saturating a server by executing a single legacy bare-metal benchmark within a VM may provide some insight into hypervisor overheads, but ignores the effect of resource sharing and contention between VMs.

Previous work on performance of virtualized environments has typically focused on single-node virtualization with limited treatment of Tier-1 applications. McDougall, *et al* [20] present an overview of improvements in single-node virtualization performance with processor generations. Adams and Agesen [2] also study single-node virtualization performance, comparing binary translation vs. initial hardware support for virtualization. They show cases in which binary translation outperforms hardware-assisted virtualization but conclude that hardware extensions are indeed addressing the correct sources of overhead. Similar to our work, Menon, *et al* [50] study virtualization overheads using the Xen Virtual Machine environment. However, their work was performed with single-vCPU VMs on single- and dual-core systems with 4K pages and a simple networking workload, while we use modern, virtualization-aware hardware and a more complex workload.

There has been a great deal of work on analyzing applications running in cloud environments, but these have not specifically focused on the virtualization backbone. Ferdman *et al* [11] describe CloudSuite, which studies emerging scale-out workloads and compares the architectural behavior such as cache misses, off-chip bandwidth, etc. with traditional benchmarks, while the Yahoo Cloud Serving Benchmark (YCSB) [10] is primarily focused on data serving workloads. While not specifically cloud benchmarks, Tarasov *et al* [49] study network-attached storage (NAS) benchmarks as VM workloads. In addition to providing an additional set of VM workloads, they also observe that virtualization increases the randomness in I/O operations since the I/O is interleaved among VMs.

One example of early attempts to study multi-VM performance is vConsolidate from Intel [17]. It contains a tile-like unit known as a CSU (Consolidated Stack Unit), and includes mail, web, and database VMs. Intel acknowledges that vConsolidate was intended as a test suite rather than as a benchmark suite and has stopped maintaining it. SpecVirt2010 [29] and SpecVirt2013 [30] are SPEC-sanctioned variants of VMmark-like workloads. In contrast to VMmark, however, SPEC allows tuning of applications and OSes as well as the hypervisor platform itself, making the comparison of different hypervisors potentially difficult. IBM has a similar effort called Virtualization Grand Slam [12] for benchmarking their POWER-based virtualization platform. In contrast to the per-tile throughput metric in VMmark, Georges *et al* [48] study per-VM throughput and define total normalized throughput and average normalized reduced throughput. In some cases, these per-VM throughput metrics yield different results from VMmark for the same benchmark, indicating different system-wide tradeoffs.

We are not aware of any other benchmarks that attempt to model virtualization management as a workload like VCBench. Soundararajan, *et al* [27], study the management workload in enterprise environments and describe the impact of management operations on the data plane. Follow-on work [28] studies the control plane and shows how it can significantly add to overheads in self-service provisioning environments. Neither work attempts to formulate a workload.

For user experience monitoring, there have been some out-of-band techniques [45] to estimate remote responsiveness, but these techniques fall short when the out-of-band techniques go out of sync and give false latencies, unlike our in-band approach. Other approaches include analyzing screen updates and attempting to automatically detect pertinent events [46]. The work that comes closest to ViewPlanner is from Jiang *et al* [18] where they show micro-architectural behavior of desktop workloads and compare the same with traditional benchmarks. While this work provides good results on architectural events, no user experience measurement framework is presented, nor do any findings that can help optimize virtual systems for desktops.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we have shown that benchmarking virtualized systems is very different from benchmarking physical systems. In a virtualized environment, the performance of the hypervisor should be matched by performance in upper-level management software tiers to provide the best end-user experience. We have developed 3 benchmark suites to assist with this evaluation: VMmark, for measuring core hypervisor performance; VCBench, for measuring the management tier, and ViewPlanner, for measuring interactive user experience in a virtual desktop environment. We have shown that hardware changes that may benefit lower layers, like CPU virtualization extensions that improve server consolidation, may need to be balanced against the additional overhead of managing more densely-packed hosts. For a more traditional native-to-virtual comparison, we have seen that while the latest virtualization-support improves end-to-end performance, the gap between native and virtual performance has increased. Moreover, end-to-end benchmarks like View Planner can expose interesting

interactions between hypervisors optimizations (like virtual interrupt coalescing) and guest OSes.

There are a number of areas for future work. From a holistic perspective, as new features get added to virtualization platforms (for example, SSD-based caching), we continue to update our benchmarks to incorporate such changes and make sure they improve performance without adverse effects elsewhere in the stack. Another example is evaluating CPU features like Advanced Vector Extensions (AVX) [15], which help floating-point performance and may also help the compute-intensive portion (e.g., SSL encryption) of various management agents, thereby improving the performance of management operations. We are also examining "next-generation" workloads like Hadoop, which are increasingly being virtualized. For benchmarks like ViewPlanner, which mimic end-user computing applications that are increasingly being accessed from mobile devices, we may need to examine our measurement techniques to incorporate touch-based UIs instead of just keyboard/mouse. We are also continuing to examine the correlation between micro-architectural events (like cache misses and TLB misses) and the performance of our benchmarks—as the software is increasingly optimized, such issues may become more prominent. Finally, rather than considering benchmark scores in isolation, it might be interesting to speculate on creating a composite score that combines that outputs of each benchmark and provides a single metric to assess the overall performance of the virtualization platform.

VII. ACKNOWLEDGMENTS

We would like to thank Jennifer Anderson, Vikram Makhija, Rajit Kambo, Bruce McCready, Adarsh Jagadeeshwaran, and the ViewPlanner and VMmark teams for their significant contributions to this work.

VIII. REFERENCES

[1] Adams, K., *et al*. A Comparison of Hardware and Software Techniques for x86 Virtualization. In Proceedings of ASPLOS XII (San Jose, CA, October 21-25, 2006). 2-13.

[2] Agesen, O. Software and Hardware Techniques for x86 Virtualization, 2009

[3] Agesen, O. "Heed the Rise of the Virtual Machines", Programming Language Design and Implementation, June 2012, Beijing, China.

[4] Ahmad, F. *et al*. Joint Optimization of Idle and Cooling Power in Data Centers While Maintaining Response Time. In Proceedings of ASPLOS XV (Pittsburgh, PA, March 13-17, 2010). 243-256.

[5] Amazon EC2. http://aws.amazon.com/ec2.

[6] Apache Hadoop. http://hadoop.apache.org/

[7] Armbrust, M., *et al*. Above the Clouds: A Berkeley View of Cloud Computing. University of California at Berkeley Technical Report No. UCB/EECS-2009-28.

[8] Bhargava, R., *et al*. Accelerating two-dimensional page walks for virtualized systems. In Proceedings of the 13th ASPLOS, March 2008.

[9] Buell, J., *et al*. Methodology for Performance Analysis of VMware vSphere under Tier-1 Applications, in VMware Tech Journal, 2013

[10] Cooper, B., *et al*. Benchmarking cloud serving systems with YCSB. In Proceedings of the 1st ACM SOCC. June 2010.

[11] Ferdman, M., *et al*. "Clearing the clouds: a study of emerging scale-out workloads on modern hardware." ACM SIGARCH. Computer Architecture News. Vol. 40. No. 1. ACM, 2012.

[12] IBM. Virtualization Grand Slam. http://www.ibmsystemsmag.com/ibmi/trends/whatsnew/It-s-a-Grand-Slam/

[13] Intel. Intel® 64 and IA-32 Architectures Software Developer's Manual. http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-manual-325462.pdf

[14] Intel Inc., First the Tick, Now the Tock: Intel® Microarchitecture (Nehalem), http://www.intel.com/content/www/us/en/architecture-and-technology/microarchitecture/intel-microarchitecture-white-paper.html

[15] Intel. Intel Instruction Architecture Extensions. http://software.intel.com/en-us/intel-isa-extensions

[16] Intel. Intel Nehalem. http://www.intel.com/p/en_US/products/server/processor/xeon7000

[17] Intel. Intel.vConsolidate. http://www.intel.com/pressroom/archive/releases/2007/20070417gloc1.htm

[18] Jiang, T., *et al*. "Micro-architectural characterization of desktop cloud workloads." IISWC 2012.

[19] Lim, K., *et al*. Understanding and Designing New Server Architectures for Emerging Warehouse-Computing Environments. In Proceedings of ISCA 2008 (Beijing, China, June 21-25, 2008). 315-236.

[20] McDougall, R., *et al*. Virtualization performance: perspectives and challenges ahead. SIGOPS Oper Syst. Rev. 44, (December 2010), 40-56.

[21] Microsoft Inc., Windows Performance Toolkit, http://msdn.microsoft.com/en-us/library/hh162945.aspx

[22] Mohan, C. Impact of recent hardware and software trends on high performance transaction processing and analytics. Performance Evaluation, Measurement and Characterization of Complex Systems, Lecture Notes in Computer Science, Volume 6417. 85-92.

[23] Nelson, M., *et al*. Fast Transparent Migration for Virtual Machines. In Proceedings of USENIX '05 (Anaheim, CA, April 10-15, 2005).

[24] OpenStack. OpenStack Cloud Software. http://www.openstack.org

[25] Ranganathan, P. Recipe for Efficiency: Principles of Power-aware Computing. CACM, Volume 53, No. 4. 60-67

[26] Real World technologies, Intel Sandy Bridge Microarchitecture, http://www.realworldtech.com/sandy-bridge/

[27] Soundararajan, V., *et al*. The impact of management operations on the virtualized datacenter. In Proceedings of the 37th ISCA. June 2010.

[28] Soundararajan, V., *et al*. Revisiting the Management Control Plane in Virtualized Cloud Computing Infrastructure. In IISWC 2013. (Portland, OR, September 22-24, 2013). 143-152.

[29] SPEC. SpecVirt2010. http://www.spec.org/virt_sc2010/

[30] SPEC. SpecVirt2013. http://www.spec.org/virt_sc2013/

[31] Transaction Processing Performance Council. TPC-C. http://www.tpc.org/tpcc

[32] VMware.VMware vSphere. http://www.vmware.com/products/vsphere

[33] VMware Inc., Timekeeping in VMware Virtual Machines, http://www.vmware.com/vmtn/resources/238

[34] VMware. VMware Distributed Power Management Concepts and Use. http://www.vmware.com/files/pdf/DPM.pdf

[35] VMware. VMware DRS. http://www.vmware.com/products/drs/

[36] VMware. VMware High-Availability. http://www.vmware.com/products/high-availability

[37] VMware. VMware Horizon Workspace. http://www.vmware.com/products/horizon-workspace/

[38] VMware. VMware Serengeti. http://www.vmware.com/hadoop/serengeti.html

[39] VMware. VMware Site Recovery Manager. http://www.vmware.com/products/site-recovery-manager

[40] VMware. VMware vCloud Director. http://www.vmware.com/products/vcloud-director/overview.html

[41] VMware. VMware View. http://www.vmware.com/products/horizon-view

[42] VMware. VMware vStorage APIs for Array Integration. http://communities.vmware.com/docs/DOC-14090

[43] VMware. vSphere API Documentation. https://www.vmware.com/support/developer/vc-sdk/

[44] Yahoo! http://www.yahoo.com

[45] Yang, S., *et al*. "The Performance of Remote Display Mechanisms for Thin-Client Computing", Proc. of the USENIX ATC, 2002.

[46] Zeldovich N. *et al*. "Interactive performance measurement with VNCplay", Proceedings of the USENIX ATC, 2005.

[47] VMware. VMware View Planner. http://www.vmware.com/products/view-planner

[48] Georges, A., and Eeckhout, L. Performance metrics for consolidated servers. In Proc. of HPCVirt (2010)

[49] Tarasov, *et al.* Virtual machine workloads: The case for new benchmarks for NAS. In Proc. of FAST 2013.

[50] Menon, *et al*. "Diagnosing Virtual Machine Overheads in the Xen Virtual Machine Environment." In Proceedings of VEE 2005 (Chicago, IL).