

Storage Management in Virtualized Cloud Environment

Sankaran Sivathanu¹, Ling Liu¹, Mei Yiduo^{1,2}, and Xing Pu^{1,3}

¹College of Computing, Georgia Institute of Technology

²Xi'an Jiaotong University, China

³Beijing Institute of Technology, China

Abstract—With Cloud Computing gaining tremendous importance in the recent past, understanding low-level implications of the cloud infrastructure becomes necessary. One of the key technologies deployed in large Cloud infrastructures namely the Amazon EC2 for providing isolation and separate protection domains for multiple clients is *virtualization*. Therefore, identifying the performance bottlenecks in a virtualized setup and understanding the implications of workload combinations and resource configurations on the overall I/O performance helps both the cloud providers in managing their infrastructure efficiently and also their customers by means of better performance.

In this paper we present the measurement results of detailed experiments conducted on a virtualized setup focusing on the storage I/O performance. We categorize our experimental evaluation into four components, each of which presenting some significant factors that affect storage I/O performance. Our experimental results can be useful for cloud application developers to tune their applications for better I/O performance and for the cloud service providers to make more effective decisions on resource provisioning and workload scheduling.

I. INTRODUCTION

The cloud computing paradigm is the backbone of various Internet services that take an ever increasing fraction of time people spend on computers today. By hosting data and applications in a server farm (i.e. a data center) and providing access through a thin-client, such services/applications enable ubiquitous access to data besides the ability to harness massive computing power even from portable devices. Cloud computing allows customers to only pay for the computing resources they need, when they need them.

In order to enable support such cloud-based applications in a cost-effective manner and to lower the barrier to entry for such applications, a somewhat recent business model that has emerged is that of a service provider hosting third-party applications in a shared data center (e.g. Amazon's EC2 platform). A key technology that enables such cost-efficient resource sharing in such data centers is *virtualization*; by providing isolation and separate protection domains for multiple third party applications, virtual machines [10], [4] enable server consolidation, and consequently significant savings in the cost of managing such data centers.

Unfortunately, one complexity that arises with virtualization is that it becomes harder to provide performance guarantees and to reason about a particular application's performance, because the performance of an application hosted on a VM is now a function of applications running in other VMs hosted on the same physical machine. Also, it may be challenging to harness the full performance of the underlying hardware, given the additional layers of indirection in virtualized resource management. While many virtual machine platforms provide good CPU isolation and minimal CPU overheads [10], [4], we find that when it comes to I/O performance, virtual machines result in some interesting and non-obvious interactions and side-effects.

In this paper, we perform a detailed experimental analysis of the I/O performance observed in a virtualized environment. Through a series of systematic experiments on the Xen [4] VMM platform, we show that the storage I/O performance observed by an application inside a VM depends on various aspects such as which disk partition the virtual machine uses, the CPU utilization of other VMs on that system, the nature of I/O performed by other VMs on the hosted system and the block size used for performing disk I/O. In many cases, the performance difference between the best choice and the average (eg. default) choice is as high as 8% to 35%. We also evaluate the costs of using a virtualized environment compared to a dedicated physical system, in terms of I/O performance.

The rest of the paper is organized as follows: In section II we present our detailed experiment results and their analysis, including the technical insights we learned through this measurement study. We outline the related research in section III and conclude in section IV

II. EXPERIMENTS AND ANALYSIS

In this section we present our empirical results in four groups based on their impact on the cloud service providers, application developers and the customers. We start with presenting results that provide insights on how to provision and place virtual storage space in physical devices. We then present the effects of workload interactions across multiple VMs that use a shared physical resource. We finally provide

results showing the performance impact of I/O block sizes for different workloads some of the differences between using a physical machine versus a VM.

A. Experimental Setup

We conducted all our experiments on a 2.4GHz Intel Quad-core machine with 4 GB RAM and two Seagate 7400 RPM SATA hard disks of 250 GB each. We used Xen hypervisor [4], version 3.3.1 and created two guest domains Dom_1 and Dom_2 . We used physical disk volumes for every partitions in each of the guest domains by default. We used a variety of monitoring tools such as *Xenmon* [6], *Xentop*, *iostat*, *blktrace* and *btt*. In cases where we record the block-level trace using UNIX *blktrace* utility, we used separate physical disks for recording the trace – this avoids most of the disk-level interference caused by the tracing activity. Each of the guest domains were allocated a single dedicated processor core as it reflects typical enterprise setting. Depending on the nature of experiments, we varied the guest domains’ RAM size from 128MB to 1GB each.

B. Virtual Disk Provisioning and Placement

Choosing the right location at the right disk: Our first set of experiments are focused on identifying the impact of provisioning and placement policies for virtual disks on the overall I/O performance.

In our experiment we have a single virtual machine running on a physical host. We test the performance of ‘Postmark’ benchmark [11] with a file size of 5KB in two different virtual disk size settings viz. 4GB and 40GB. Postmark benchmark is reminiscent of mail server workload with various configurable parameters like file size, block size, read/write ratio, etc.,. The 4GB and 40GB virtual disks are separate partitions from a single physical disk of size 250GB. The total data footprint of our workload is 150 MB, and the total number of files is 30000 spread across 500 subdirectories. As the file placement policy aims to co-locate the files within only a single directory, the files from 500 subdirectories are spread across the entire virtual disk by the file system. Figure 1(a) and Figure 1(b) show the distribution of read accesses in both cases. We used Linux *blktrace* utility to collect the block level traces and plotted them in the context of the entire physical disk. When the virtual disk size is 4GB, the data (and hence the accesses) is confined to the 4GB (~141GB-145GB) space. However when the virtual disk size is 40GB, even a small data footprint of 150MB is spread across the entire 40GB (~196GB-236GB) space. Figure 1(d) and Figure 1(e) show the corresponding throughput profile of each virtual disk respectively. The disk I/O throughput for 4GB setting is at 2.1 MS/sec on average, whereas for 40GB, it is about 60% lesser on 1.3 MB/sec. As the disk seek distance is vastly reduced when the virtual disk size is small, the throughput achieved by a 4GB virtual disk is significantly better than that of a 40GB disk for the

exactly same workload. When disproportionate amount of disk space, with respect to data footprint of the workload, is allocated for a user, some file system allocation policies or application decisions might lead to data being spread across a larger footprint on disk.

Next, we present some insights regarding *placement* of virtual disks in a physical storage system. In the set of experiments reported in this paper we consider different cases within a single disk. However, the principles discussed here are applicable to large scale storage arrays as well. We start with the configuration of one VM per host and then extend to multiple VMs per host. The goal of these experiments is to identify the impact of placement of virtual disks on both the I/O performance of each virtual machine and the overall throughput achieved by the physical disk device. We consider four cases: (i) a file-backed virtual disk (which is common in most desktop virtualization setting), where a large file is created in the host’s file system and it is mounted as a disk in the virtual machine, and then virtual disks backed by physical partitions which are located in (ii) outer, (iii) middle (mid) and (iv) inner zones of a disk. We ran Postmark benchmark with varying file sizes in each of these settings and collected the storage I/O bandwidth values. Figure 1(c) shows the bandwidth achieved by each of these virtual disk configurations. We observe that the file-backed disk performs consistently worse than the other cases. This is because the virtual disk is essentially a file in the host file system and every time a virtual machine accesses its disk, the meta data for that file in the host file system need to be accessed as well. Therefore, meta-data access is doubled for most of the disk accesses by the virtual machine – one within the virtual machine, and the other by the host file system. We also observe that, although there is not much variation across different configurations for small file sizes, the performance varies significantly as the file size increases. This is because as the file size gets larger, the sequentiality is increased and the subtle differences between various regions within the disk are clearly shown. In contrast, for smaller files, the accesses are mostly random and the disk seek overhead amortizes smaller differences in accessing various disk regions. In addition, we observe that for large files (sequential accesses), the performance is best when the virtual disk is placed in the outer-most zone. This is because the outer-most zones have larger tracks and sequential accesses benefit from a single large track read. As we move to the inner zones, the track size decreases and the sequential accesses would incur more and more track switches, reducing the I/O performance.

Next, we try to understand the case of more than one virtual disks per physical host. The virtual disks may either be from a single VM or multiple VMs. We compare different cases of placing the two virtual disks and identify the best setting with respect to overall I/O performance. Figure 1(f) shows the average response time of Postmark benchmark with file size averaging to 10KB. We bypassed the file

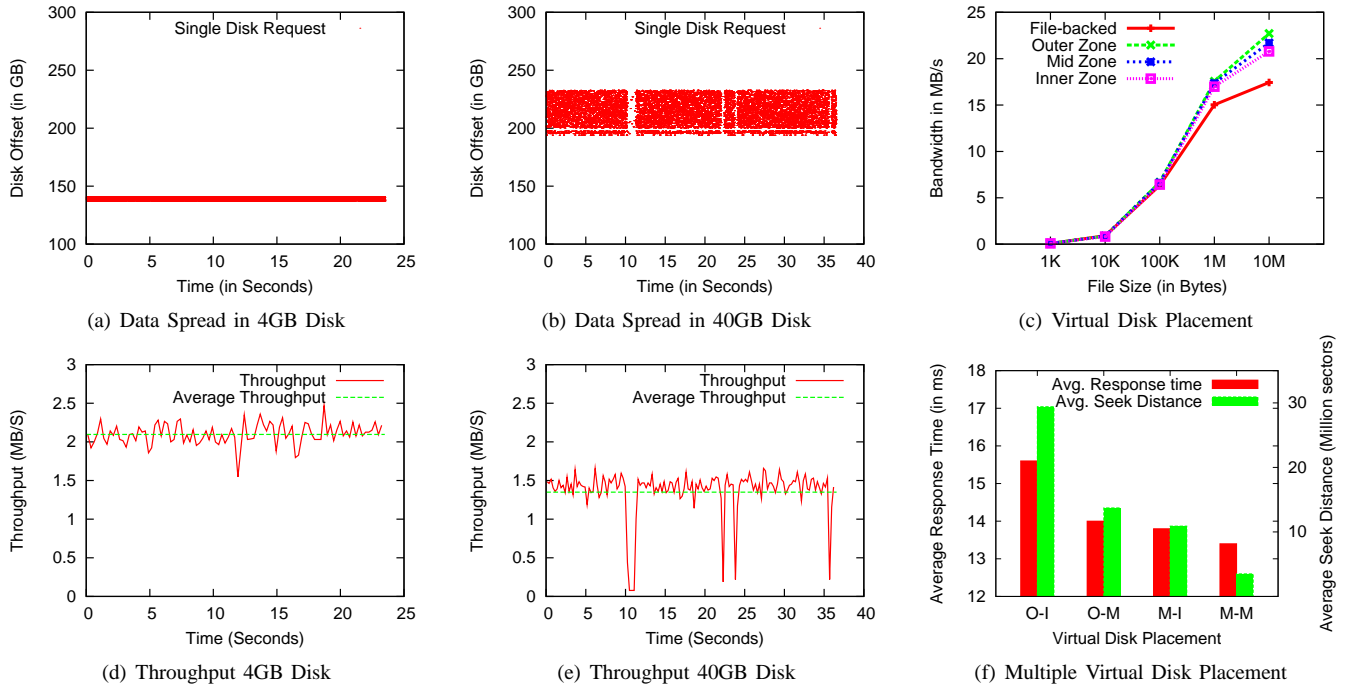


Fig. 1. Provisioning and Placement of Virtual Disks

system cache in order to understand isolated disk behavior for different configurations by using ‘O_DIRECT’ flag for the Postmark transactions. The six different configurations, with respect to placement of the two virtual disks, are: (i) One virtual disk in outer zone(smaller LBNs) and other in inner zone(larger LBNs)(O-I), (ii) One virtual disk in inner zone and other in middle zone(M-I), (iii) one virtual disk in middle zone and other in outer zone(O-M), (iv) both virtual disks in middle zone(M-M), (v) both virtual disks in inner zone, and (vi) both virtual disks in outer zone. We show the first four cases in our results as the other two results resemble the (M-M) configuration. The secondary Y-axis shows the average seek distance of the requests reaching the physical disk. This value is obtained by running blktrace utility in the physical disk device. This result shows that when the two virtual disks are co-located, they provide the best performance when compared to other placement combinations. This is because the seek-distance is reduced when the virtual disks are collocated, leading to faster seeks.

Remarks: These experimental results deliver four important observations: (i) Cloud service providers can improve their storage efficiency by practicing thin provisioning of their storage resources. (ii) Customers of cloud storage can purchase storage based on their expected workload footprint rather than just the price. This is important because, customers may tend to purchase more disk space than what they need, especially when the price differential is not significant. (iii) It is beneficial to place sequentially accessed virtual disk in the outer zone of the disk to achieve better performance (Eg. VMs that host movies, large OS distributions, etc.). (iv) When a cloud service provider needs to place multiple virtual disks in a single physical storage, those virtual disks whose

accesses are temporally dependent on each other should be co-located for best performance. By temporal dependency, we refer to the cases where each of the two virtual disks are owned by one or more VMs such that the applications running on them are closely related or causally dependent on each other in the sense that both applications might issue disk requests fairly simultaneously.

C. Workload Interference

Choosing right workload for the right VM : Our next set of experiments are focused on understanding the interactions between different types of workloads running in multiple VMs hosted on a single physical machine. It is important to study such interactions because these are common cases in a typical infrastructure cloud setting, where VMs hosted in a physical machine serves to different customers. Therefore, it is important that a specific load from one VM should not adversely affect other applications running on other VMs hosted in the same physical machine.

We dedicate this group of experiments to the understanding of two types of contention, as both affects the overall I/O performance: (i) one VM running CPU intensive application and another VM running disk intensive application; and (ii) both VMs running disk intensive applications. We first run a CPU intensive workload (a batch of floating point operations in loop) and a disk intensive workload (Postmark benchmark) in two different VMs, Dom_1 and Dom_2 , and track the performance of the disk workload in Dom_2 . Intuitively, one might expect a drop in performance of the I/O workload when there is another VM running a CPU intensive workload, given that the Dom_0 CPU is time-shared between all the domains. However our results show a different phenomenon. Fig-

Freq. Governor		Reqs/s	Blks/s	CPU Util	CPU Freq(MHz)
‘Ondemand’	Idle CPU	4756	38042.5	64.7	1400
	Busy CPU	5169	41344.8	63.8	2400
‘Performance’	Idle CPU	5166	41314.2	62.9	2400
	Busy CPU	5163	41302	63.2	2400

TABLE I
EFFECT OF DYNAMIC FREQUENCY SCALING ON I/O PERFORMANCE

ure 2(a) shows bandwidth achieved by Postmark benchmark with varying file sizes under two cases: (1) Dom_1 is idle and Dom_2 runs postmark benchmark, and (2) Dom_1 runs CPU intensive task and Dom_2 runs postmark benchmark. Counter-intuitively, the Dom_2 I/O performance in case-2 is better than that of case-1. We initially suspected it to be Xen’s VM scheduling anomaly. We repeated the same experiments under a different Xen scheduler, namely SEDF, instead of using the default CREDIT scheduler. However, the results did not change significantly. In order to isolate this behavior with respect to workloads, we chose synthetically generated purely sequential workload. Figure 2(b) shows similar behavior across different CPU weight assignments for purely sequential file read. The ratio in the X-axis represents the weight ratio of virtual CPUs of the two VMs, Dom_1 and Dom_2 . We conjecture that this behavior is observed for sequential workloads across different Xen schedulers.

We further investigated the experiments by a Xen monitoring tool-Xenmon [6] and recorded the ‘CPU Cycles/Execution’ metric, which shows the number of CPU cycles spent for a particular execution unit. Figure 2(c) shows the CPU cycles/Execution for both case (1) and case (2). We observe that case (1), where the Dom_1 is idle, incurs more cycles per execution than case (2), where Dom_1 is running a CPU intensive task.

We eventually found that this behavior is because of a power optimization technique used in the CPU, called *Dynamic Frequency Scaling*. It is a technique where the frequency of the CPU (and hence the power consumed) is increased or decreased dynamically based on CPU load. For an Intel Quad-Core processor and Linux kernel version 2.6.18, the dynamic frequency scaling is enabled by default. However, this policy is configurable by means of choosing CPU frequency scaling ‘governors’ in the Linux kernel, and the options available in Linux kernel are *performance*, *powersave*, *ondemand* and *Conservative*. Linux allows the user to set a particular CPU frequency scaling policy based on the power and performance requirements. ‘ondemand’ is the default option in Linux kernel 2.6.18, which means the CPU frequency is scaled dynamically based on load. The ‘performance’ option is used to run the CPU at high frequency always and the ‘powersave’ option runs the CPU in lowest frequency always. The ‘conservative’ is similar to ondemand with slightly more bias towards either performance or power. Therefore, the fact that Dom_2 I/O performance in case (2) (busy CPU in Dom_1) out-performs that of case (1) (idle CPU in Dom_1) is annotative: when CPU is idle in Dom_1 , the overall CPU load on the physical host is less with just the I/O workload in Dom_2 . Therefore the CPU

is ‘under-clocked’ i.e., the CPU frequency is reduced, this results in the Postmark benchmark running in a lower frequency. When Dom_1 runs a CPU intensive task, the overall CPU load is increased significantly, thereby switching the CPU to highest frequency automatically, thanks to dynamic frequency scaling. Though we have allocated separate cores to each of the VMs, the frequency scaling policy is applied to all cores equally. Therefore, the difference between case (1) and case (2) essentially represents the performance difference between the case when Dom_2 CPU runs in highest frequency and the case when Dom_2 CPU runs in lower frequency. In Figure 2(a) the dynamic frequency scaling phenomenon is less significant for smaller file sizes because random accesses on disk lead to larger seek delays, which amortizes delays due to reduced CPU frequency. Table I shows the throughput and CPU utilization values for the default ‘Ondemand’ policy and the ‘Performance’ policy along with the corresponding CPU frequency values. This set of results further testifies our measurement analysis.

We next measure the effect of running two I/O intensive workloads on multiple VMs hosted on a single physical machine. For these experiments, we use Postmark benchmark and synthetically generated combinations of sequential and random reads. Figure 2(d) shows the achieved bandwidth of postmark benchmark for various file sizes in three different configurations viz. (i) one instance of postmark in a single VM and other VM remains idle, (ii) two VMs each run one instance of postmark and their virtual disks reside in same physical disk, and (iii) two VMs each run one instance of postmark and their virtual disks reside in different physical disks. We observe that case (ii) performs the worst among all three configurations as the sequentiality in both Postmark instances is affected when both virtual disks(which reside in a single physical disk) are accessed alternatively. Interestingly there is roughly 24% difference in bandwidth for 10MB file size between cases (i) and (iii) even though each of the two VMs in case (iii) is running its virtual disks in separate physical disks (and with separate processor cores). This can be viewed as a scheduling overhead between the two virtual machines. As all I/O requests go through Dom_0 ’s block device driver, when one of the VMs is scheduled, the requests in the other VM waits for their turn and its disk might skip the sector to be read. This may lead to rotational delays even in the case of pure sequential workloads, hence accounting for the reduced bandwidth.

In order to understand the effect of different I/O schedulers in Dom_0 on I/O Performance in guest domains, we run synthetically generated sequential workloads in each of the two VMs whose virtual disks reside in two different physical

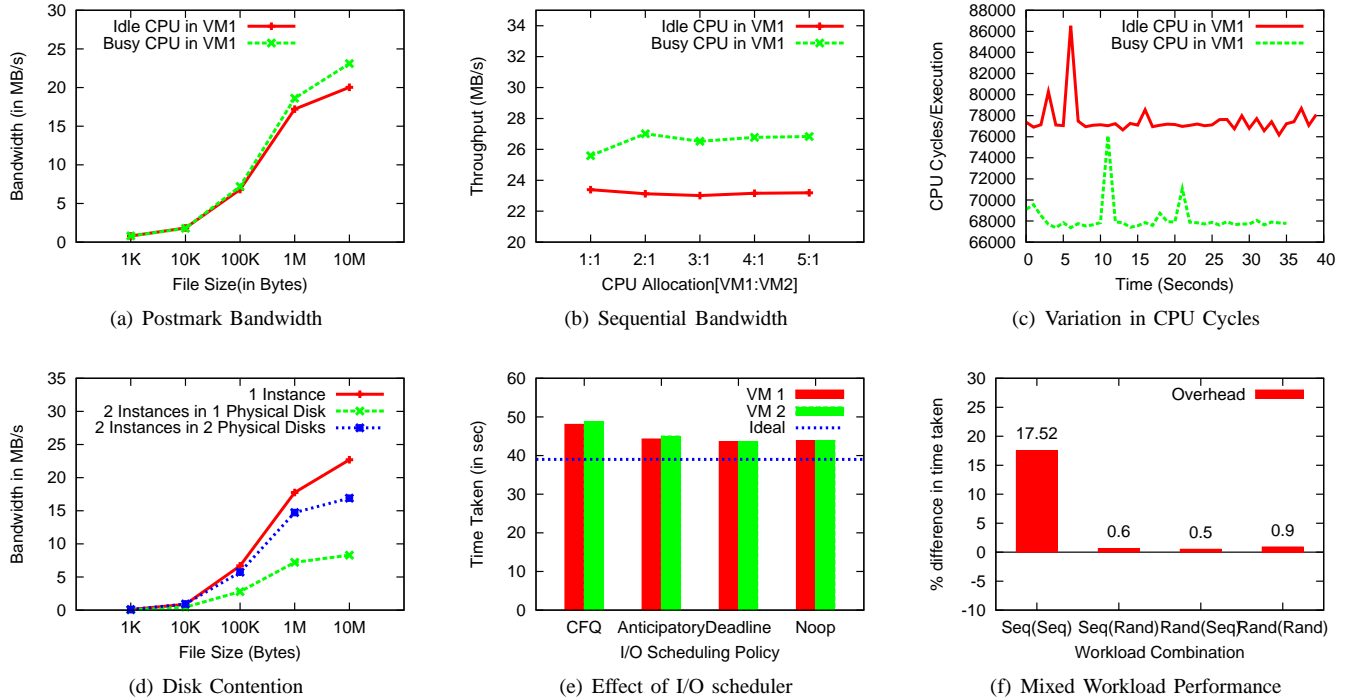


Fig. 2. Resource Contention w.r.t. to workloads

disks. Figure 2(e) shows the time taken for reading 100,000 blocks of data from a large file sequentially in the granularity of 4KB blocks. The ‘ideal’ bandwidth shown in Figure 2(e) is the bandwidth recorded when a single process reads 200,000 blocks of data. The comparison against this ideal bandwidth value shows the scheduling overhead when running two virtual machines, each serving half of this amount of requests. Figure 2(e) shows that the Deadline scheduler and the Noop scheduler (no scheduling) perform slightly better than the Completely Fair Queued scheduler (CFQ) for this particular workload. However, we observe that the performance does not change significantly across scheduling policies and the perceived overhead is inherent when alternating I/Os between multiple virtual machines.

Figure 2(f) shows the performance overhead when running different types of I/O workload in difference combinations. We consider sequential and random reads and study four combinations of them. Every x-axis points is in the form *foreground(background)*. For example, Seq(Rand) represents the performance overhead suffered by a foreground sequential workload when a background random workload runs simultaneously. The y-axis represents the percentage difference in time compared to running just the corresponding foreground workload (without any background workloads). We observe that when two sequential workloads run simultaneously, there is close to 18% performance difference compared to running a single sequential workload. Note that the foreground and background workloads are sent to different physical disks since we set up each VM with its own physical disk in this set of experiments. However, other combinations of sequential and random workloads induce very less overhead.

This is because of the contention in one of the Dom_0 's queue structures: two sequential workloads would populate the queue faster when compared to other workloads and therefore some of the sequential requests would have to wait before entering the queue.

Remarks: Through this group of experiments, we make three observations, which are useful for both cloud service providers and their consumers. (i) Most of the infrastructure cloud systems employ Dynamic Frequency Scaling technique to conserve power. Therefore, strategies for co-locating workloads can be critical for achieving maximum performance and at the same time conserving power. (ii) Cloud service providers should avoid co-locating VMs with two sequential I/O workloads on a single physical machine, even when each of them goes to separate physical disks.

D. Impact of I/O block size

Choosing right block size for the right workload: In this group of experiments, we analyze the significance of I/O block size of VMs on the overall I/O performance of the system. In our experiments we run synthetically generated random workload of varying spatial locality between accesses. Our policy of defining the locality factor is as follows: *Once a block is read, it is likely that ‘x’ more blocks out of the next ‘y’ contiguous blocks are read in the near future*. In all experiments, the total amount of data read is configured to be the same. We read 16,000 blocks of data from a 1GB file. In ‘No_Locality’, we read 16,000 blocks of data randomly, each 4KB in size. The remaining four cases, denoted as ‘Locality_x’, represent the workload where we have a set of random accesses and between every such access,

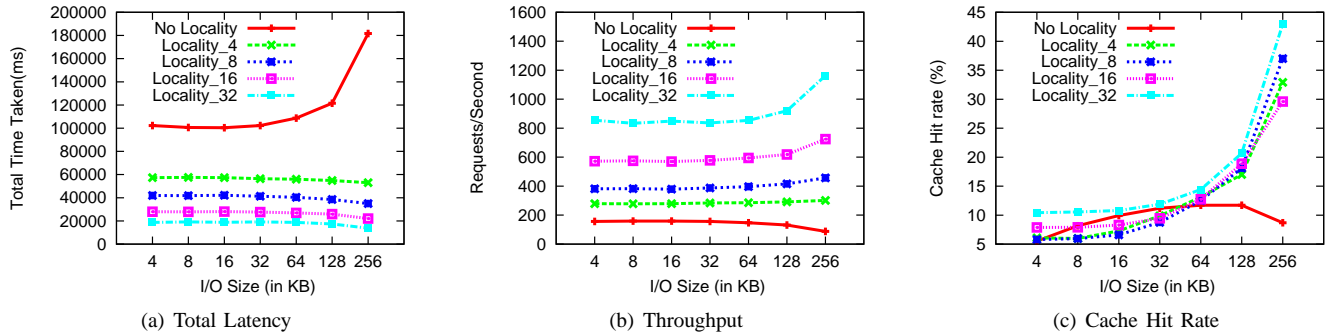


Fig. 3. Effect of I/O size on Performance

there are ‘x’ more block accesses within the neighboring 256 blocks.

Figure 3(a) shows the difference in total time taken by the workload with respect to varying I/O block sizes for different locality factors. Clearly, as the block size increases, we see that the time taken for workloads with spatial locality constantly decreases. However, when there is no locality in the workload, the total time taken raises sharply as the block size increases. Figure 3(b) shows similar behavior when we measure the throughput as the block size increases from 4KB to 256KB. The requests served per second increases for workloads with locality. This is because, when there is spatial locality in the workload, reading a large amount of data in one request helps in placing more data in page cache and future requests to neighboring blocks are served from the page cache. However, when the block size is small, even requests to neighboring blocks are treated as individual random requests, thus reducing performance. When there is no locality of data, reading in larger block sizes reduces the performance because irrelevant data are read into the page cache, leading to cache pollution and wasteful I/O bandwidth.

Figure 3(c) shows the cache hit rate for each of the five configurations discussed above. It is interesting to note that for workloads with locality, there is a steep increase in cache hit rate, while the slope of increase in throughput and decrease in latency is much lesser comparatively. This shows that even though many requests are served from cache, there is a penalty to be paid for reading large amount of data in every request in the form of storage bandwidth consumed.

Remarks: This group of experiments shows the impact of I/O block size, with respect to spatial locality of I/O workloads, on the overall performance of the system. Cloud application developers can use this result to tune their application’s I/O block size based on the spatial locality.

E. Virtual Machine vs. Physical Machine

The Pros and Cons of using a virtual machine: Our final set of experiments are aimed at identifying additional overhead introduced by using a virtual machine when compared to its physical machine equivalent for different types of I/O workloads. For these experiments we carefully configure the VM and the physical machine resources such that the comparison is fair. In addition to using the same amount

of hardware resources, we used the same operating system (Ubuntu 8.04 Server), kernel version and the I/O scheduler in both cases. We used CFQ (Completely Fair Queued) scheduler for both physical machine and *Dom₀*. The guest domains’ I/O scheduler is set to ‘noop’ so that they do not schedule the requests themselves before they are queued in *Dom₀*.

First, we run Postmark benchmark on both physical machine and its equivalent virtual machine and record the bandwidth achieved in both cases. Figure 4(a) shows the bandwidth measurements (MB/sec) of postmark benchmark under various file sizes. The bandwidth achieved by the physical machine is not much different from that of the VM for small file sizes (1KB, 10KB). As the file size increases there is increasing drop in bandwidth in virtual machine bandwidth, when compared to physical machine bandwidth. This is because, when the file size is small and when they are spread across many directories, the Postmark transactions are seen by the disk as random accesses. As the file size increases, the accesses become more and more sequential. Therefore, the overhead induced by virtualization is amortized by the disk seek overhead in the case of small file sizes. As the file size grows, the disk seek overhead is reduced, thus projecting the virtualization overhead.

In order to understand the virtualization overhead in the case of large file size (or sequential access), we picked up a constant file size of 10MB and ran the Postmark benchmark with varying block sizes. Figure 4(b) shows the bandwidth of Postmark workload running in a physical machine versus a VM. As the block size reaches 1MB, the virtual machine achieves performance similar to that of a physical machine. This is because, a small and fixed overhead is introduced in the virtual machine for every system calls/requests. This can be attributed to the extra layers introduced by virtualization technique. Therefore as the block size increases, the same amount of work is done by issuing lesser number of requests, thus reducing virtualization overhead. Figure 4(c) shows performance of postmark benchmark with 10MB file size, when two instances of the benchmark are running in two VMs simultaneously. We compare the performance of running two instances in a single disk versus running them on two different disks for both physical machine and virtual

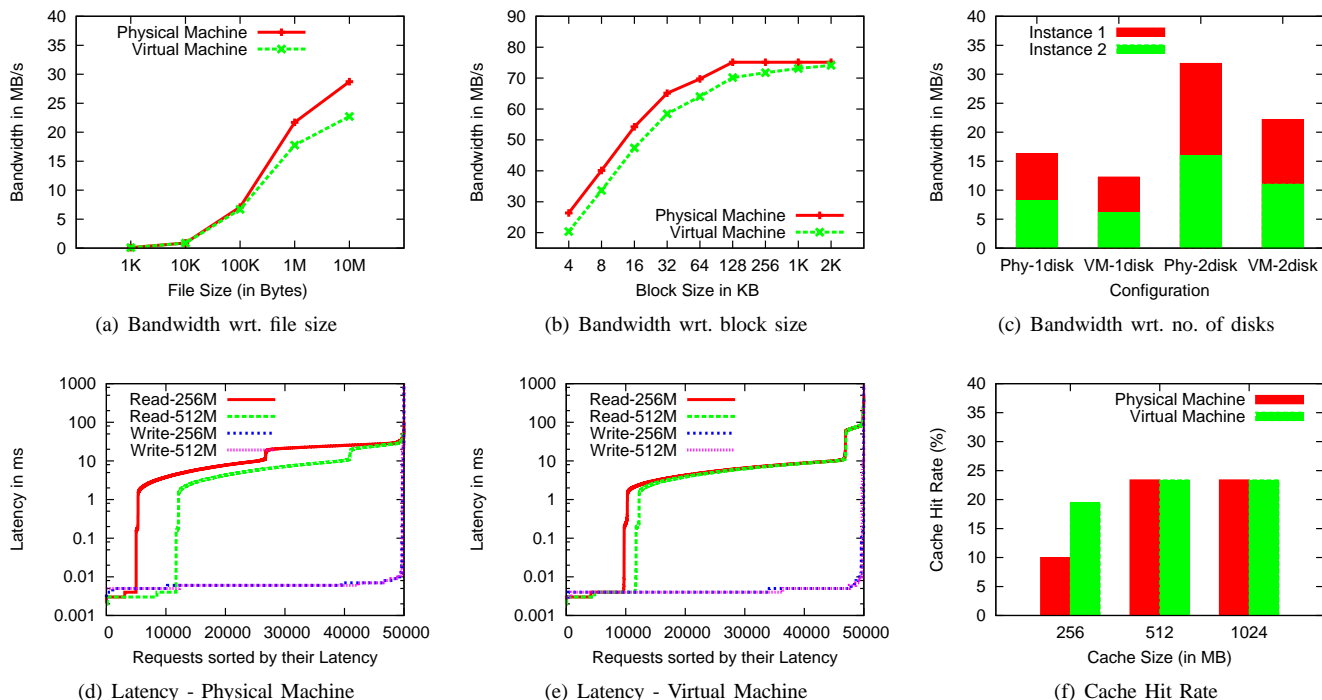


Fig. 4. Comparison of Physical machines versus Virtual machines

machine. Overall we observe that there is a 32% difference in bandwidth between physical machine and its virtual machine equivalent both in the case of single disk and two separate disks.

Finally, we show that virtual machine performs better than a physical machine in some specific cases resulting from an inherent design advantage of virtualization technology called Cache Isolation. In a physical machine, the cache space is generally not partitioned equally across processes because one process may use the cache space better than the other. On the flip-side, there is a chance that one process may be unduly affected by another process, such as when the other process uses the cache aggressively than the other. For example, when there are two processes, one doing large amount of writes and other process reading a set of files frequently, the cache space may be aggressively filled by the write process leaving no space for read cache. This is not the case for two VMs because the cache space is well divided between the two VMs. Therefore it is beneficial to run two workloads competing for cache space in two different VMs than on a single VM or a single physical host. We confirm this hypothesis by conducting a set of experiments that capture the latency, cache hit rate and overall bandwidth. For these experiments, we run synthetic read workload that models Zipf distribution of accesses in the foreground and run a background sequential write workload. We tailored our foreground workload such that the blocks are generally accessed randomly and 20% of the blocks are accessed 80% of the time. Cache size of ‘x’ MB refers to ‘x’ MB of memory in the case of physical machine and partitioned as ‘x/2’ MB of memory in each of the two VMs. Figures 4(d)

and 4(e) show the CDF of latency values of foreground read for two different cache sizes. The latency CDF of physical machine shows a sudden spike in latency(cache miss) at around 9% of requests for 256MB cache and around 22% of requests for 512MB cache size. This shows that the write workload consumes undue amount of cache space leaving lesser space for the foreground read workload. When this is compared to the performance in virtual machine setup, where even for aggregate cache size of 256 MB, around 19.5% of requests are hit in cache. The footprint of our read workload is 150 MB and background write workload is 400 MB. Therefore at 512MB of cache, the workload achieves top performance –any more increase in cache size no longer have any effect on the overall performance. For cache size of 512MB, both the read and write workload reside in the cache without affecting each other. Figure 4(f) shows precise cache hit rate for different cache sizes. These results show that virtualization provides better cache isolation between two processes running in two separate virtual machines, when compared to running them in a single physical system.

Remarks: In our final set of experiments we derive two observations which will be useful for cloud customers and service providers: (i) Virtualization overhead is more felt in sequential workloads accessed through smaller block sizes than random workloads. The cloud customers can estimate the expected performance based on the characteristics of the workload they deploy. (ii) Cloud service providers can practice static partitioning of memory across multiple VMs hosted in a single physical machine, whenever strict performance isolation is important across different customers.

III. RELATED WORK

In this section we provide some of the earlier work on virtual machine I/O performance study. The case of optimized performance in virtual environments has been studied in the past and most of them focus on sharing and scheduling of CPU [7], [16], [8], partitioning and sharing memory [17], [14], and networking [5], [13], [12]. However, IO optimization in the storage layer [15], [1], [9] has relatively been studied less. Most of these focus on effect of disk I/O on CPU [7] or effect of workloads on disk performance in virtual machine [1] or on I/O scheduling issues and impact of different scheduling policies on the I/O performance [9]. Nevertheless, there are no papers that study the interference of multiple virtual machines purely in the disk layer to the best of our knowledge.

Provisioning and placement issues in enterprise storage servers have been studied in the past [2], [3] which is useful for large scale storage providers. But our experiments target the customers of cloud storage also in providing them insights into how much space they need to purchase for their typical usage. Researchers have presented the effect of CPU and the I/O performance in the perspective of CPU scheduling [7], [8], but the side-impacts of CPU power optimization policies on the I/O performance have not been studied before.

IV. CONCLUSION

We presented a measurement study of I/O performance in a virtualized environment. Our experimental design and measurements are focused on identifying and understanding the set of factors that have performance implications on efficient storage management in a virtualized cloud environment. This paper makes three unique contributions. First, our experiments and analysis shows the significance of careful disk provisioning and placement on I/O performance. Second, our measurement study on workload interference, especially taking into consideration the dynamic frequency scaling feature deployed widely in modern processors. Third, we have shown the implications of virtualization on different types of workloads.

We conjecture that the insights gained from this measurement study will be useful for cloud service providers in maximizing their resource utilization and overall system performance at the same cost. Application developers for cloud environments can also benefit from our study by making more informed decisions about optimal I/O block size for their applications based on the workload characteristics.

V. ACKNOWLEDGEMENTS

We acknowledge the partial support by grants from NSF CISE NetSE program, NSF CISE CyberTrust program, IBM faculty award, IBM SUR grant, and a grant from Intel Research Council.

REFERENCES

- [1] Irfan Ahmed, Jennifer.M Anderson, Anne M. Holler, Rajit Kambo, and Vikram Makhija. An analysis of disk performance in vmware esx server virtual machines. In *Proceedings of the Sixth Annual Workshop on Workload Characterization*, Austin, Texas, October 2003.
- [2] G. A. Alvarez, E. Borowsky, S. Go, T. H. Romer, R. A. Becker-Szendy, R. A. Golding, A. Merchant, M. Spasojevic, A. C. Veitch, and J. Wilkes. Minerva: An automated resource provisioning tool for large-scale storage systems. *ACM Trans. Comput. Syst.*, 19(4):483–518, 2001.
- [3] E. Anderson, M. Hobbs, K. Keeton, S. Spence, M. Uysal, and A. Veitch. Hippodrome: Running circles around storage administration. In *Proceedings of the First USENIX Conference on File and Storage Technologies (FAST 2002)*, pages 175–188, Monterey, CA, January 2002. USENIX Association.
- [4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pages 164–177, Bolton Landing, NY, October 2003. ACM SIGOPS.
- [5] Vineet Chadha, Ramesh Illiikkal, Ravi Iyer, Jaideep Moses, Donald Newell, and Renato J. Figueiredo. I/o processing in a virtualized platform: a simulation-driven approach. In *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*, pages 116–125, New York, NY, USA, 2007. ACM.
- [6] Ludmila Cherkasova and Rob Gardner. Measuring cpu overhead for i/o processing in the xen virtual machine monitor. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 24–24, Berkeley, CA, USA, 2005. USENIX Association.
- [7] Ludmila Cherkasova and Rob Gardner. Measuring cpu overhead for i/o processing in the xen virtual machine monitor. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 24–24, Berkeley, CA, USA, 2005. USENIX Association.
- [8] Ludmila Cherkasova, Diwaker Gupta, and Amin Vahdat. Comparison of the three cpu schedulers in xen. *SIGMETRICS Perform. Eval. Rev.*, 35(2):42–51, 2007.
- [9] Boucher Dave and Chandra Abhishek. Does virtualization make disk scheduling pass? In *Hotstorage '09: SOSP Workshop on Hot Topics in Storage and File systems*. ACM, 2009.
- [10] A. S. Kale. VMware: Virtual machines software. www.vmware.com, 2001.
- [11] J. Katcher. PostMark: A new filesystem benchmark. Technical Report TR3022, Network Appliance, 1997. www.netapp.com/tech_library/3022.html.
- [12] Aravind Menon, Jose Renato Santos, Yoshio Turner, G. (John) Janakiraman, and Willy Zwaenepoel. Diagnosing performance overheads in the xen virtual machine environment. In *VEE '05: Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, pages 13–23, New York, NY, USA, 2005. ACM.
- [13] Diego Ongaro, Alan L. Cox, and Scott Rixner. Scheduling i/o in virtual machine monitors. In *VEE '08: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 1–10, New York, NY, USA, 2008. ACM.
- [14] Martin Scwidefsky, Hubertus Franke, Ray Mansell, Himanshu Raj, Damian Osisek, and JonHyuk Choi. Collaborative memory management in hosted linux environments. In *Proceedings of the Linux Symposium, Volume 2*, Ottawa, Canada, July 2006.
- [15] Jeremy Sugarman, Ganesh Venkitachalam, and Beng-Hong Lim. Virtualizing i/o devices on vmware workstation's hosted virtual machine monitor, 2001.
- [16] Ananth I. Sundararaj, Ashish Gupta, and Peter A. Dinda. Increasing application performance in virtual environments through runtime inference and adaptation. In *In Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 47–58, 2005.
- [17] C. A. Waldspurger. Memory resource management in VMware ESX server. In *OSDI'02: Fifth Symposium on Operating Systems Design and Implementation*, pages 181–194, New York, NY, USA, 2002. ACM Press.