A Model of Storage I/O Performance Interference in Virtualized Systems

Giuliano Casale
Dept. of Computing
Imperial College London
London, SW7 2AZ, UK
g.casale@imperial.ac.uk

Stephan Kraft*
SAP Research
Belfast, UK
stephan.kraft@sap.com

Diwakar Krishnamurthy
Dept. of ECE
University of Calgary
Calgary, AB, Canada
dkrishna@ucalgary.ca

***Abstract*—In this paper, we propose simple performance models to predict the impact of consolidation on the storage I/O performance of virtualized applications. We use a measurement-based approach based on tools such as *blktrace* and *tshark* for storage workload characterization in a commercial virtualized solution, namely VMware ESX server. Our approach allows a distinct characterization of read/write performance attributes on a per request level and provides valuable information for parameterization of storage I/O performance models. In particular, based on measures of quantities such as the mean queue-length seen upon arrival by requests, we define simple linear prediction models for the throughput, response times, and mix of read/write requests in consolidation based only on information collected in isolation experiments for the individual virtual machines.**

***Keywords*-Performance modeling; Virtualization; Storage**

I. Introduction

In virtualized systems, estimates of I/O contention for a given Virtual Machine (VM) placement configuration can support management and consolidation decisions. However, modeling the performance of disk requests is very challenging due to the interaction of the I/O flows issued by several VMs. In this paper we derive simple measurement-driven models to predict throughputs and response times of disk requests in an environment where multiple consolidated VMs can share a storage server. In particular, we describe a method that only needs input obtained when VMs run in isolation on a virtualized server, thus the effects of consolidation are estimated before actually consolidating the workloads. Such a method therefore obviates the need to collect data for different VM consolidation scenarios.

There are several important issues that need to be addressed while modeling virtualized I/O environments. Queueing models for I/O consolidation should distinguish between reads and writes as they affect system performance in very different ways. Ganger and Patt [1] introduce three classes of requests based on how individual request response times influence system performance. An I/O request is considered *time-critical*, if the generating thread blocks until the request is completed, e.g. a process is halted until a synchronous read request has been completed. *Time-limited* requests must be completed within a given amount of time otherwise they will become time-critical, for example file system read-aheads. Requests that do not require waiting times of the submitting process are classified as *time-noncritical*. Such disk I/O requests must be completed to maintain stable copies of nonvolatile storage, for example background flushes of asynchronous writes. Time-noncritical requests can indirectly impact performance when interfering with the completion of more critical requests, thus model definition is complicated by the interaction between these different workload behaviors.

Finally, our experience shows that disk requests service times can have significant temporal correlations leading to bursty workloads [2]. The batched submission of disk requests can result in an extreme behaviour of the arrival patterns where large amounts of requests are condensed into a short time period. As a result, we record large values (i.e. > 0.35) for the autocorrelation function of response times for single VMs running on a Virtual Machine Monitor (VMM). In these experiments the utilization of the storage device is low and therefore response times should be a close approximation of I/O request service requirements. The correlation of arrival and service times complicates accurate performance predictions using analytical models.

The remainder of the paper is organized as follows. Section II gives an overview of related work and Section III introduces the reference system for our study. Our analysis of consolidated workload traces and the batched submission of requests is illustrated in Section IV. The proposed modeling approach is presented in Section V. Section VI offers summary and conclusions.

II. Related Work

A large amount of I/O research literature is concerned with scheduling algorithms for disk I/O in virtualized environments. The main challenges regarding scheduling are to provide fair access to the shared storage resource for all in-VM applications, while maintaining performance isolation, i.e. disk accesses by one application should not affect the I/O performance of another [3]. Performance isolation in the

*Stephan Kraft is also affiliated with Queen's University, Belfast, UK.

The work of G. Casale has been supported by the Imperial College Junior Research Fellowship scheme. The work of S. Kraft has been partially funded by the InvestNI/SAP VIRTEX project.

presence of resource contention is studied in [4]. The authors consolidate different types of workloads, i.e. CPU bound and disk bound, and derive mathematical models to predict relative performance compared to a normalized performance score. Closer to our work, [5] derives a mathematical model to predict disk I/O throughputs when moving from a native system to an isolated VMware ESX server environment. The work in [6] measures interference of consolidated workloads on an ESX server using histograms, but do not offer a modeling approach. More recently, [7] characterizes disk workloads and performance metrics in a consolidated virtualized environment. Contrary to our work they do not consolidate by placing multiple workloads on the same physical device, i.e., LUN, but consolidate multiple LUNs into a single RAID group.

Queueing models are a popular tool to model the performance of shared resource environments. A shared server environment is modeled as a time-domain queueing model with Generalized Processor Sharing scheduling in [8] in order to compute and assign resource shares. In [9], layered queueing networks are used to model multi-tier applications hosted in consolidated server environments. Recently, [10] proposes an iterative model training technique based on artificial neural networks for dynamic resource allocation in consolidated virtualized environments. While some of the work above captures coarse grained disk requirements in the model in order to predict effects of resource allocation changes on performance of consolidated applications, none specifically tries to predict fine grained disk I/O request performance degradation due to workload consolidation.

III. Experimental Environment

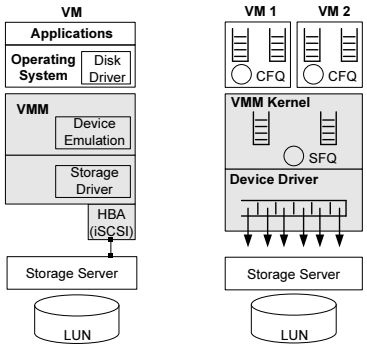
In this section we introduce our reference system, measurement tools used for workload characterization, as well as the considered workload configurations.

A. Reference System

We conduct our study on an AMD-based enterprise server with 4 quad-core processors containing a total of 16 CPU cores each clocked at 2.21GHz. The system is equipped with 68GB of RAM and is connected to an OpenFiler [11] storage server via the iSCSI protocol and 1 GBit Ethernet. The storage server manages a SATA-II hardware RAID controller, which in turn manages a 15 disc RAID 5 array.

On the server we host the virtualization platform VMware ESX Server 3i - 3.5.0 [12], which accesses the storage device through a software iSCSI host bus adapter (HBA). The virtualized environment consists of multiple Debian 5.0.1, kernel 2.6.26-2, guest VM systems, each with identical configurations of 1 virtual CPU, 500MB RAM, and 50GB of "virtual" hard disk formatted as an *ext3* file system.

The virtual disks are represented by a large file and can be thought of as a linear array made up of units of space, i.e. logical blocks. In the remainder of this paper the term



(a) Storage Subsystem (b) Storage Queues

Figure 1. I/O Architecture and Storage Queues of the Reference System.

"block" always refers to logical block units, rather than the storage device's physical block units. In our system virtual disk files of all VMs are stored on a single physical storage device (LUN) and access the LUN through the same HBA.

Disk I/O requests issued from a VM consist of one or multiple contiguous blocks for either reads or writes. Once an application running inside the VM submits a request, the request first goes to the disk driver of the VM operating system as shown in Figure 1 (a). The driver processes the request and forwards it to the VMM, where it is trapped and may undergo further optimization operations before being issued to the LUN via the storage driver and the iSCSI HBA [5].

On their way through the previously described layers of the storage subsystem, requests can be queued multiple times as illustrated in Figure 1 (b). Hence latencies of requests may be affected by multiple queueing delays. Furthermore, requests may undergo optimizations such as splitting, merging, and reordering by schedulers managing the various queues shown in Figure 1 (b). Such operations are used to optimize disk access patterns, e.g., by merging multiple requests for small amounts of contiguous blocks to fewer requests for large amounts of contiguous blocks. In virtualized environments these operations can have a significant impact on disk request performance, as scheduling policies at VM and VMM level may impair each other [13].

In our environment the Debian guest VMs are configured to use the completely fair queueing (CFQ) [14] scheduler, which has per default 64 internal queues to maintain and keep disk I/O requests [15]. The in-VM scheduler optimizes disk access for the ext3 file system, which is configured at the default block size of 4kB. Hidden from the operating system of the VM, the VMM conceptually comprises two sets of queues, namely the VMM kernel queues and the device driver queue. The VMM kernel maintains a queue of pending requests per VM for each target SCSI device [16], controlled with a fair-share (FS) [17] scheduler. Fair-share algorithms are work-conserving and schedule shared resources between competing requests by allocating capacity based on proportional weights. In our environment all

requests are assigned equal resource shares. The VMM formats virtual disk files in the *virtual machine file system* [12] and performs its own splitting, merging and reordering processing on disk requests that are queued at the VMM kernel.

Furthermore, the server maintains a device driver queue for each LUN, which controls the *issue queue length* defined as the number of pending requests the server can have at the storage device at a time [18]. Virtualized servers typically allow to configure the issue queue length for each LUN, which we maintain at the default specification of 32.

B. Measurement Tools

This section describes the monitoring tools we have used to collect disk I/O traces in order to quantify the performance of disk requests. Traces are captured at two system layers: the virtualization and the storage server. Our traces comprise a time stamp for each request issue and completion, a flag that indicates whether a request was a read or write, the logical block address (LBA) pertaining to the request, and the number of blocks accessed. We note that we calculate per request response times as the time difference between completion and issue events in the trace.

Measurements on the virtualized server are obtained inside VMs with the block layer I/O tracing mechanism *blktrace* [19]. The tool allows us to record traces of disk request *issue* and *completion* events, as they are recorded from the VM operating system kernel. In order to monitor the I/O request trace submitted from the VMM to the storage server, we use the network protocol analyzer *tshark* [20] to intercept iSCSI network packets. The tool is installed at the storage server and only collects the headers of iSCSI packets, which enables us to extract operational codes and control information without handling iSCSI payloads.

C. Workload Generation

We conduct our study using emulated disk workloads of mail, web, and file server type applications. Workloads are generated with FileBench [21], a framework for measuring and comparing file system performance. We maintain the default workload specification and use the recommended parameters for small configurations (50000 files in the initial file set). The presented measurements are gathered during a series of benchmarking experiments, each consisting of 15 runs of 300s length. We report results as the means over all 15 iterations or based on a representative run. Experiments are conducted with a single VM running in isolation and with two VMs on the same server in consolidation. Specifically, we consolidate one web and one mail server, denoted Web+Mail, one file and one mail server, denoted File+Mail, and one file and one web server, denoted File+Web.

IV. Workload Consolidation Study

We use the consolidation scenario Web+Mail as an example to illustrate how the reference system submits consolidated workloads to the storage device. Section IV-A uses monitoring data from the storage server for a static analysis of the arrival instant queue lengths seen by read and write requests. Section IV-B presents a dynamic analysis to show how arrival queues change over time.

Table I
Accuracy of Read/Write Mix Predictions for Consolidated Workloads.

	Measurement (%)		***Prediction (%)***			
Workload	*Total: VM1+VM2*		*Total: VM1+VM2*			
	R	W	R	Δ	W	Δ
File+Mail	0.47	0.53	0.49	0.03	0.51	0.03
File+Web	0.72	0.28	0.74	0.03	0.26	0.08
Web+Mail	0.58	0.42	0.63	0.10	0.36	0.13

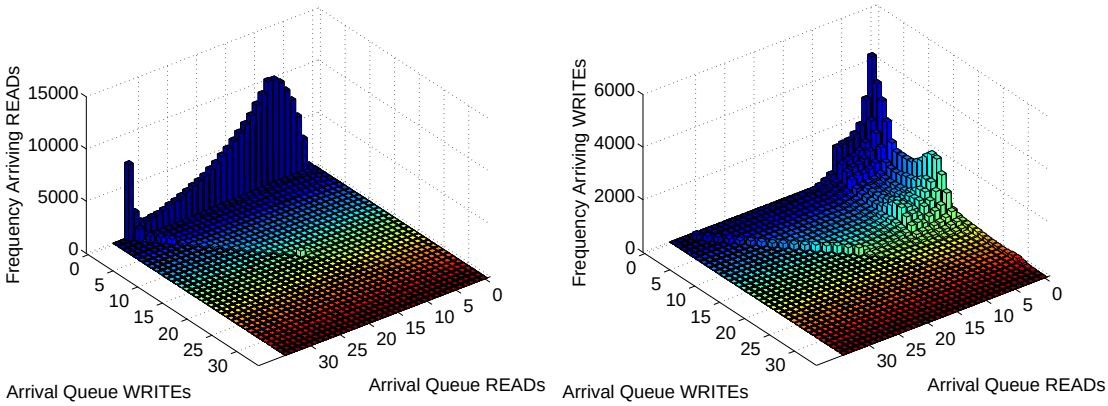
Table II
Accuracy of Throughput Predictions for Consolidated Workloads.

	Measurement (cmd/s)		***Prediction (cmd/s)***			
Workload	*Total: VM1+VM2*		*Total: VM1+VM2*			
	R	W	R	Δ	W	Δ
File+Mail	293	336	286	0.03	307	0.09
File+Web	448	176	397	0.12	146	0.17
Web+Mail	382	280	348	0.09	222	0.21

A. Static Analysis of Arrival Queue Lengths

The I/O workload of the storage device consists of a mixture of read and write requests. Due to the mixed nature of consolidated workloads, ideally the system must minimize situations where time-noncritical write requests may interfere with time-critical reads. We observe how the arrival queue length at the storage server is partitioned between read and write requests, to study how our reference systems submits requests of different types. Figure 2 (a) shows that a majority of arriving read requests only find reads ahead of them. This suggests that the VMM assigns a form of priority shares for read requests, e.g. similar to original UNIX systems (System 7) that used a disk request scheduler giving read requests non-preemptive priority. Mostly the utilization of the storage server is low, since only a small fraction of the 32 available connections from the VMM to the storage server are being used. However, there is a significant number of instances where the maximum amount of available connections are in use indicating high utilization periods. Interestingly, we also see a significant number of cases where the arrival queue contained 16 writes.

Figure 2 (b) shows more variability in the arrival queue lengths for write requests. On most occasions arriving write requests find a small number of reads, as well as other write requests ranging in $[1; 16]$. These observations suggest that a maximum of 16 write requests are batched from the VMM to the storage device. However, there also are noticeable frequencies where arriving write requests see large queues of read requests. These cases could indicate situations where batches of write and read requests are submitted within a short time interval.



(a) Arrival Queue as Seen by Arriving Reads (b) Arrival Queue as Seen by Arriving Writes

Figure 2. Arrival Queue Lengths From a Two VM Consolidation Experiment.

Table III
ACCURACY OF RESPONSE TIME PREDICTIONS FOR CONSOLIDATED WORKLOADS.

	Measurement (ms)				***Prediction (ms)***							
Workload	*VM1*		*VM2*		*VM1*				*VM2*			
	R	W	R	W	R	Δ	W	Δ	R	Δ	W	Δ
File+Mail	35.0	6.90	45.5	7.90	36.3	0.04	6.91	0.003	37.6	0.17	12.2	0.55
File+Web	40.3	9.45	29.6	7.19	33.6	0.17	10.9	0.16	32.1	0.08	11.1	0.54
Web+Mail	33.0	7.97	28.0	7.03	28.2	0.15	12.9	0.62	30.9	0.11	8.48	0.21

B. Time-varying Arrival Queue Lengths

To better understand the dynamic behavior of arrival queues, we randomly chose a 1s interval and illustrate how the queue at the storage server changes over time. Figures 3 (a) and (b) show the time-varying arrival queue for read and write requests, respectively. We find further confirmation that arrival patterns of both request types are extremely bursty. While sizes of read request bursts roughly vary in ranges of $[5; 32]$, write requests appear to be scheduled in batch sizes $<= 16$. Furthermore, the system seems to batch read and write requests independently. Even though we see some occasions where a burst of read requests finds a few scattered write requests and vice versa, largely it either submits a burst of reads, or a burst of writes.

V. MODELING APPROACH

In this section, we discuss modeling the consolidation of virtual machines in ESX server. We assume detailed information about each VM to be available from the isolation experiments, and we focus on the prediction of the expected I/O performance of the VMs in consolidation. We describe here the consolidation of two VMs, the extension to the case with 3 or more VMs seems viable and will be addressed in future work.

Our performance prediction methodology considers the following performance metrics:

- T_i^R: mean throughput of VM i's read requests in isolation
- $T_{i,j}^R$: mean throughput of VM i's read requests in consolidation with VM j
- $X_{i,j}^R$: throughput of read requests, originating from any VM, in the consolidation of VMs i and j

A similar notation T_i^W, $T_{i,j}^W$, and $X_{i,j}^W$ is used for write requests. We also introduce the following derived quantities:

- $T_i = T_i^R + T_i^W$: mean throughput of VM i in isolation
- $T_{i,j} = T_{i,j}^R + T_{i,j}^W$: mean throughput of VM i in consolidation with VM j
- $\alpha_i^R = T_i^R/T_i$: relative throughput fraction of read requests for VM i in isolation
- $\alpha_{i,j}^R = T_{i,j}^R/T_{i,j}$: relative throughput fraction of VM i's read requests in consolidation with VM j
- $\beta_{i,j}^R = (T_{i,j}^R + T_{j,i}^R)/(T_{i,j} + T_{j,i})$: mix of read requests, originating from any VM, in the consolidation of VMs i and j

Similar quantities α_i^W, $\alpha_{i,j}^W$, and $\beta_{i,j}^W$ are defined for write requests. In addition, for response times we define indexes C_i^R, $C_{i,j}^R$, C_i, $C_{i,j}$ with similar meaning of the corresponding indexes for throughputs. We now propose three classes of linear estimators for request throughputs, mix, and response times in consolidation.

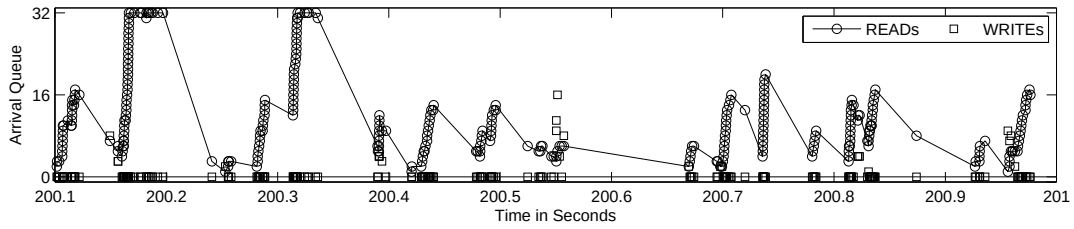
A. Approximation of Consolidation Mixes

Let us first introduce the following linear predictor for the consolidation mixes $\alpha_{i,j}^R$ and $\alpha_{i,j}^W$.

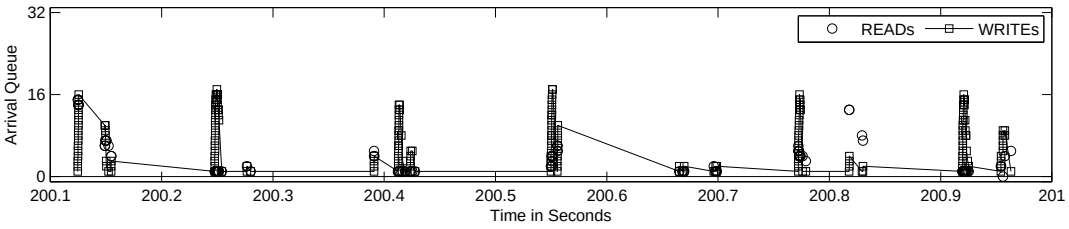
$$\beta_{i,j}^R \approx \alpha_i^R \left(\frac{T_i}{T_i + T_j}\right) + \alpha_j^R \left(\frac{T_j}{T_i + T_j}\right) \quad (1)$$

$$\beta_{i,j}^W \approx \alpha_i^W \left(\frac{T_i}{T_i + T_j}\right) + \alpha_j^W \left(\frac{T_j}{T_i + T_j}\right) \quad (2)$$

Justification. Let T the length of the period the system was observed and denote with n_i^R (resp. n_i^W) the number of read (resp. write) requests completed by the system in T. Then by definition $X_i^R = n_i^R/T$, $X_i^W = n_i^W/T$, and thus $\alpha_i^R = n_i^R/(n_i^R + n_i^W)$, $\alpha_i^W = n_i^W/(n_i^R + n_i^W)$. Using a similar



(a) Arrival Queue as Seen by Arriving Reads



(b) Arrival Queue as Seen by Arriving Writes

Figure 3. Time-Varying Analysis of Arrival Queue Lengths From a Two VM Consolidation Experiment.

notation for consolidation we get $\alpha_{ij}^R = (n_{i,j}^R + n_{j,i}^R)/(n_{i,j} + n_{j,i})$, $\alpha_{ij}^W = (n_{i,j}^W + n_{j,i}^W)/(n_{i,j} + n_{j,i})$. Then

$$\beta_{i,j}^R = \left(\frac{n_{i,j}^R + n_{i,j}^W}{n_{i,j}^R + n_{i,j}^W + n_{j,i}^R + n_{j,i}^W}\right)\left(\frac{n_{i,j}^R}{n_{i,j}^R + n_{i,j}^W}\right) + \left(\frac{n_{j,i}^R + n_{j,i}^W}{n_{i,j}^R + n_{i,j}^W + n_{j,i}^R + n_{j,i}^W}\right)\left(\frac{n_{j,i}^R}{n_{j,i}^R + n_{j,i}^W}\right)$$

where we can identify the terms $\alpha_{i,j}^R = n_{i,j}^R/(n_{j,i}^R + n_{j,i}^W)$ and $\alpha_{j,i}^R = n_{j,i}^R/(n_{j,i}^R + n_{j,i}^W)$. Consider now the approximation $\alpha_i^R \approx \alpha_{i,j}^R$, which assumes the incoming workload from VM i to be the same in isolation and consolidation. This approximation is accurate if the arrival process of VM i is loosely dependent on the overheads of consolidation, a situation that we have seen verified with good accuracy very often. Using this approximation we get

$$\beta_{i,j}^R = \left(\frac{n_{i,j}}{n_{i,j} + n_{j,i}}\right)\alpha_i^R + \left(\frac{n_{j,i}}{n_{i,j} + n_{j,i}}\right)\alpha_j^R = \left(\frac{T_{i,j}}{T_{i,j} + T_{j,i}}\right)\alpha_i^R + \left(\frac{T_{j,i}}{T_{i,j} + T_{j,i}}\right)\alpha_j^R$$

where the last passage follows by first scaling numerator and denominators by T. The final formula is obtained by approximating the throughput ratios in consolidation by the ratios of T_i and T_j in isolation. This corresponds to the assumption that a common overhead factor c_{oh} exist for the two VMs such that

$$\left(\frac{T_{i,j}}{T_{i,j} + T_{j,i}}\right) = \left(\frac{c_{oh}T_i}{c_{oh}T_i + c_{oh}T_j}\right) = \left(\frac{T_i}{T_i + T_j}\right)$$

The justification for $\beta_{i,j}^W$ follows in a similar way.

B. Approximation of Consolidation Throughputs

Using a similar justification as the one for read/write mixes, we define the following linear estimators for the total throughputs of read and write requests in consolidation

$$X_{i,j}^R \approx T_i^R\left(\frac{T_i}{T_i + T_j}\right) + T_j^R\left(\frac{T_j}{T_i + T_j}\right) \quad (3)$$

$$X_{i,j}^W \approx T_i^W\left(\frac{T_i}{T_i + T_j}\right) + T_j^W\left(\frac{T_j}{T_i + T_j}\right) \quad (4)$$

C. Approximation of Consolidation Response Times

Finally, we outline our ongoing efforts towards modeling response times of read and write requests in consolidation. Our approach involves using the response times and arrival queue-lengths collected with the *blktrace* and *tshark* tools to estimate the expected response times in consolidation. Let S_i^R and S_i^W be the estimated service times of read and write requests in isolation. Assuming first-come first-served as an approximation of the scheduling policy at the disk, we use the following mean-value analysis (MVA) [22] estimates $S_i^R \approx C_i^R/(1 + A_i^R)$, where $1 + A_i^R$ is the queue-length seen upon arrival by a read request at ESX including the arriving job itself. Equivalently, the arrival queue-length can be considered at the storage if ESX data is not directly available, since we found that the difference between the two measurements is typically negligible. Then we estimate the expected response times for a read request in consolidation as

$$C_{i,j}^R \approx C_i^R + S_j^R A_j^R + \left(\frac{max\_write\_conn}{max\_conn}\right) S_j^W A_j^W \quad (5)$$

where max_conn is the maximum number of available iSCSI connections to the storage and max_write_conn is the maximum number allocated to write requests. Following our empirical observations, we have set for ESX server $max\_conn = 32$ and $max\_write\_conn = 16$. This approximation relies on the idea that a read request coming from VM i finds ahead a number of read and writes requests from VM j similar to the isolation experiment for VM j. The choice of using A_j^R and A_j^W is a pessimistic one,

since we assume that requests coming from VM i may find batches of reads of mean size A_j^R or batches of writes of mean size A_j^R ahead. Choosing other queue-length metrics, such as the time-averaged queue-length, result in overly-optimistic predictions that ignore the batching. Finally, the scaling factor $(max\_write\_conn/max\_conn)$ accounts for the fact that writes can compete with reads only for a fraction of the available connections. Similarly, for write requests we define the following approximation

$$C_{i,j}^W \approx C_i^W + S_j^R A_j^R + \left(\frac{max\_write\_conn}{max\_conn}\right) S_j^W A_j^W \tag{6}$$

Finally, analogous expressions are defined for the response times of VM j, i.e., $C_{j,i}^R$ and $C_{j,i}^W$.

D. Evaluation

Tables I, II, III report approximation errors for three independent consolidation experiments obtained by the mail, web, and file server workloads generated by FileBench. The results for mix and throughput predictions are fairly accurate, with absolute relative errors compared to measurement in the range $[0.03, 0.13]$ for the request mix and $[0.03, 0.21]$ for throughput. Notice that read requests have errors generally lower than write requests; this is a positive property since predicting the performance of read requests, which are mostly synchronous, is much more relevant than predicting the performance of write requests, which are mostly asynchronous and thus do not impact much on the response times of the applications in the VMs.

The results for response times in Table III indicate that our heuristic for response time predictions is highly-effective for read requests, which are the most important to predict; the absolute relative errors are in the range $[0.04, 0.17]$ for reads. Conversely, our model is still insufficient for predicting write request performance, where our model is overly pessimistic. We conjecture that this is due to the fact that ESX seems to actively issue write requests to the disks in periods where the system is less congested with reads. This is shown in Figure 2(b), where we notice that a write request rarely finds more than about $5 - 10$ read requests, even though Figure 2(a) shows that such periods are frequent. Our model does not account for such behavior, hence we plan to further refine it in future work along this direction.

VI. Conclusion

In this paper, we have provided a number of insights on the behavior of consolidated storage workloads in a commercial virtualized solution, VMware ESX server. Our measurements based on the *blktrace* and *tshark* tools support the idea that modeling batching is fundamental for response time predictions in consolidation, whereas throughput and request type mixes can be effectively approximated by linear combinations of the same metrics obtained with isolated virtual machine experiments before consolidating.

In future work, we intend to further refine our models trying to define a better model for the interaction between reads and writes in consolidation, as outlined in Section V-D, and to assess the effectiveness of our approach on consolidation of three or more VMs.

References

[1] G. R. Ganger and Y. N. Patt, "The process-flow model: Examining i/o performance from the system's point of view," in *SIGMETRICS*. ACM, 1993, pp. 86–97.

[2] A. Riska and E. Riedel, "Disk drive level workload characterization," in *USENIX*, 2006, pp. 97–102.

[3] W. Jin, J. S. Chase, and J. Kaur, "Interposed proportional sharing for a storage service utility," *ACM PEVA*, vol. 32, no. 1, pp. 37–48, 2004.

[4] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen, and C. Pu, "An analysis of performance interference effects in virtual environments," in *ISPASS*. IEEE, 2007, pp. 200–209.

[5] I. Ahmad, J. M. Anderson, A. M. Holler, R. Kambo, and V. Makhija, "An analysis of disk performance in VMware ESX server virtual machines," in *WWC-6*. IEEE, 2003, pp. 65–76.

[6] I. Ahmad, "Easy and efficient disk I/O workload characterization in VMware ESX server," in *IISWC*. IEEE, 2007, pp. 149–158.

[7] A. Gulati, C. Kumar, and I. Ahmad, "Storage workload characterization and consolidation in virtualized environments," in *VPACT*, 2009.

[8] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," in *IWQoS*. Springer, 2003, pp. 381–398.

[9] G. Jung, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, and C. Pu, "Generating adaptation policies for multi-tier applications in consolidated server environments," in *ICAC*, 2008, pp. 23–32.

[10] S. Kundu, R. Rangaswami, K. Dutta, and M. Zhao, "Application performance modeling in a virtualized environment," in *HPCA*, 2010, pp. 1–10.

[11] "openfiler," http://www.openfiler.com.

[12] "vmware," http://www.vmware.com.

[13] D. Boutcher and A. Chandra, "Does virtualization make disk scheduling passé?" *SIGOPS OSR*, vol. 44, no. 1, pp. 20–24, 2010.

[14] J. Axboe, "Linux block IO - present and future," in *Linux Symposium*, 2004, pp. 51–61.

[15] "RHEL 5 IO tuning guide," http://www.redhat.com/docs/wp/performancetuning/iotuning/index.html.

[16] A. Gulati, I. Ahmad, and C. A. Waldspurger, "PARDA: Proportional allocation of resources for distributed storage access," in *FAST*. USENIX, 2009, pp. 85–98.

[17] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *SIGCOMM*. ACM, 1989, pp. 1–12.

[18] "Storage queues and performance," http://communities.vmware.com/docs/DOC-6490.

[19] "blktrace-linux man page," http://linux.die.net/man/8/blktrace.

[20] "tshark manpage," http://www.wireshark.org/docs/man-pages/tshark.html.

[21] "Filebench," http://www.solarisinternals.com/wiki/index.php/FileBench, 2010.

[22] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains*. 2nd ed., John Wiley and Sons, 2006.