# IO Performance Interference among Consolidated n-Tier Applications: Sharing is Better than Isolation for Disks

Chien-An Lai, Qingyang Wang, Josh Kimball, Jack Li, Junhee Park, Calton Pu
Center of Experimental Computer Systems, Georgia Institute of Technology, Atlanta, GA, USA
{calai, qywang, jmkimball, jack.li, jhpark, calton}@cc.gatech.edu

***Abstract*—The performance unpredictability associated with migrating applications into cloud computing infrastructures has impeded this migration. For example, CPU contention between co-located applications has been shown to exhibit counter-intuitive behavior. In this paper, we investigate IO performance interference through the experimental study of consolidated n-tier applications leveraging the same disk. Surprisingly, we found that specifying a specific disk allocation, e.g., limiting the number of Input/Output Operations Per Second (IOPs) per VM, results in significantly lower performance than fully sharing disk across VMs. Moreover, we observe severe performance interference among VMs can not be totally eliminated even with a sharing strategy (e.g., response times for constant workloads still increase over 1,100%). By using a micro-benchmark (Filebench) and an n-tier application benchmark systems (RUBBoS), we demonstrate the existence of disk contention in consolidated environments, and how performance loss occurs when co-located database systems in order to maintain database consistency flush their logs from memory to disk. Potential solutions to these isolation issues are (1) to increase the log buffer size to amortize the disk IO cost (2) to decrease the number of write threads to alleviate disk contention. We validate these methods experimentally and find a 64% and 57% reduction in response time (or more generally, a reduction in performance interference) for constant and increasing workloads respectively.**

## I. INTRODUCTION

Server consolidation research has assumed virtualization technology can provide zero interference for the economical sharing of computing infrastructures such as computing clouds [1] [2]. However, performance interference among consolidated applications have been shown to occur in a variety of systems and applications [3] [4]. Performance interference, appearing in the form of non-monotonic response time, occurs when too many applications are co-located, causing a loss of Quality of Service (Qos) and business revenues. The performance unpredictability associated with consolidation has been recognized as one of the top 10 obstacles to the growth of cloud computing [5].

To prevent the noisy neighbors problem, perfect isolation may appear to be an intuitive solution for eliminating performance interference among the VMs. For example, VMware ESXi allows administrators to limit CPU utilization and IOPs (Input/Output Operations Per Second) for a specific VM to improve performance isolation. Moreover, Gupta et al. introduced ShareGuard to further specify the total number of resources consumed in privileged and driver domains in Xen [6]. In addition, Ye et al. proposed VM-level and core-level cache awareness optimization methods to further enhance performance isolation among VMs [7].

However, contrary to the above intuition, we found evidence showing that the limitations imposed by perfect isolation can outweigh its benefits. Where CPU was the critical resource, an experimental study of VM consolidation [8] showed sharing compared to perfect isolation allowed better utilization of CPU (the critical resource) thus improving overall response time by up to 50%. This counter-intuitive result was called "sharing is better than isolation" [8], which motivated the title of this paper.

The first contribution of this paper is an experimental study of VM consolidation when disk (a representative I/O device) becomes a critical (bottleneck) resource. We show disk isolation performs worse than sharing by detailing the significantly increased overhead disk isolation incurs when it is enforced. A deeper analysis of our experimental results shows that an isolation strategy loses, because it is a "selfish" strategy in which an idle VM prevents the disk from performing any work because of the specified IOPs constraint. In contrast, the sharing strategy wins, because each VM can use the disk when other VMs handle CPU-bound tasks. We explore the differences between disk sharing and isolation using a co-located database scenario. Specifically, we find isolation separates the VMs, so the respective databases' committed records cannot be written synchronously.

Conversely, sharing allows these systems to interleave the writing of committed records, hence optimizing disk I/O (e.g., smaller number of arm movements.)

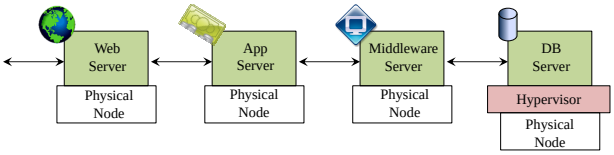
Secondly, we provide an experimental study of several factors related to DB *fsync*, which cause the disk to become the bottleneck. We demonstrate the IO inefficiency of co-located database systems is caused by these systems needing to flush their logs from memory to disk to maintain consistency. As a potential solution to this found performance interference problem, we assess system performance under different database configurations. We found that bigger buffer sizes enabled larger numbers of commit records to be written in a single group commit *fsync*. In addition, decreasing the number of I/O threads for write operations also improved performance isolation by alleviating disk resource contention. More generally, this study shows that virtualization technologies are a relatively immature technology and represent interesting topics for future research.

The rest of the paper is structured as follows. In Section II, we describe our experimental setup, detailing our n-tier application deployment and our testbed. Section III demonstrates the challenges with consolidating n-tier applications in clouds. Furthermore, Section III-A compares different disk allocation strategies, and it shows sharing disk is better than reserving IOPs for consolidated n-tier applications. Section III-B discusses a common disk contention issue, which is brought about by databases' flusing their logs, exhibited by our experimental consolidation scenario. Section III-C proposes changing several database settings for those databases running on co-located VMs in order to improve their resource utilization. Section IV provides an overview of related work in this area. Finally, Section V concludes the paper.

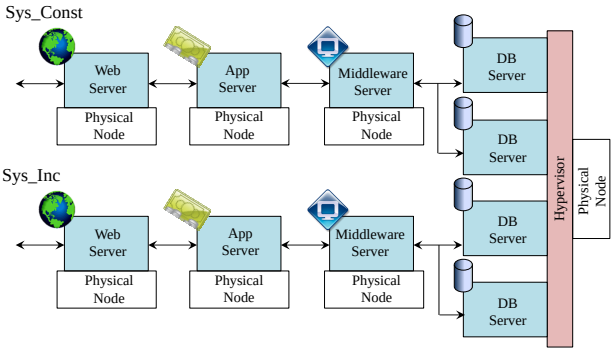
## II. EXPERIMENTAL SETUP

While consolidation in practice may be applied to any type of application, the focus of this paper is n-tier applications with LAMP (Linux, Apache, MySQL, and PHP) implementations. Typically, n-tier applications are organized as a pipeline of servers, starting at web servers (e.g., Apache), through application servers (e.g., Tomcat), and ending with database servers (e.g., MySQL) organized in three tiers or four where an additional layer contains an application, or middleware, for clustering (e.g., C-JDBC). This organization, commonly referred to as n-tier architecture (e.g., 4-tier in Figure 1a), serves many important web-facing applications such as e-commerce, customer relationship management, and logistics. In our experiment, we deploy the popular n-tier application benchmark system RUBBoS [9]. The benchmark includes two kinds of workload modes: browse-only and read/write mixes. In this paper, we focus entirely on the read/write workloads, and specifically, the ratio of write request to all is 10%.

[0]In this paper, we refer to server in the context of computer programs serving client requests. Hardware is referred to as *physical computing node* or *node* for short.




(a) Dedicated deployment of a 4-tier application system with four software servers (i.e., web, application, middleware, and database) and four physical hardware nodes




(b) Consolidated deployment of two 4-tier systems (*Sys_Const* and *Sys_Inc*) with 1/1/1/2 configuration and seven physical hardware nodes in total. The DB server are co-ocated in dedicated VMs on a single shared physical hardware node.

Fig. 1: Example of a dedicated 1a and a consolidated 1b 4-tier application system deployment, presented as mappings of software servers to physical hardware nodes.

At an abstract level, the deployment of n-tier applications in a cloud computing infrastructure can be modeled as a mapping between component severs and physical computing nodes. An application deployment is *dedicated* whenever the number of physical nodes is at least equal to the number of servers as exemplified in Figure 1a. If the number of physical nodes is smaller than the number of servers, the deployment mapping is a *consolidation*, which requires at least two servers to be co-located on a single node (e.g., Figure 1b). In our experiments, we denote the first RUBBOS system as *Sys_Const* and the second RUBBOS system as *Sys_Inc*, as illustrated in the figure. Unless otherwise stated, the default consolidation methodology in this paper is to affiliate (i.e., pin) all of the VMs to different CPU cores to prevent any possible CPU contention in our experiments.

TABLE I: Summary of experimental setup (i.e., hardware, operating system, software, and virtualization environment).

| | |
|---|---|
| CPU | 2 Quad Q9650 3GHz * 4CPU |
| Memory | 16GB |
| HDD | SATA, 7,200RPM, 500GB |
| Network I/F | 1Gbps |
| Web Server | HTTPD-2.2.22 |
| App Server | Apache-Tomcat-5.5.17 |
| Connector | Tomcat-Connectors-1.2.32-src |
| DB Server | MySQL-5.5.19-Linux2.6-i686 |
| Java | JDK1.6.0_27 |
| Monitoring Tools | IOstat and Esxtop |
| Hypervisor | VMware ESXi v5.0.0 |
| Guest OS | RHEL Server 6.3 64-bit |
| Guest OS Kernel | 2.6.32-279.22.1.e16.x86_64 |

For example, in the case when four VMs host individual database servers, each server is pinned to four different cores, but they occupy the same physical node. They only share the same disk as depicted in Figure 1b. Other important characteristics of our experimental testbed are summarized in Table I.

Software configuration, in the context of our work, refers to the server software settings that specify how many software resources such as processes, threads or DB connections are allocated to each server. Software resource is a natural extension of hardware resource (e.g., CPU and network) and inherently controls how much processing can occur concurrently on each tier. Previous research has shown that a delicate balance between concurrency overhead and processing throughput exists. As such, sophisticated configuration algorithms might need to be used in dedicated deployment scenarios [10]. In this paper, both systems are configured with the same practical resource allocation settings, which were derived from our previous experiments for a similar consolidation scenario [8]. These previous experiments validated the specific software resource settings that were required for delivering consistently good overall performance in dedicated n-tier deployments.

## III. EXPERIMENTAL CONSOLIDATION STUDY

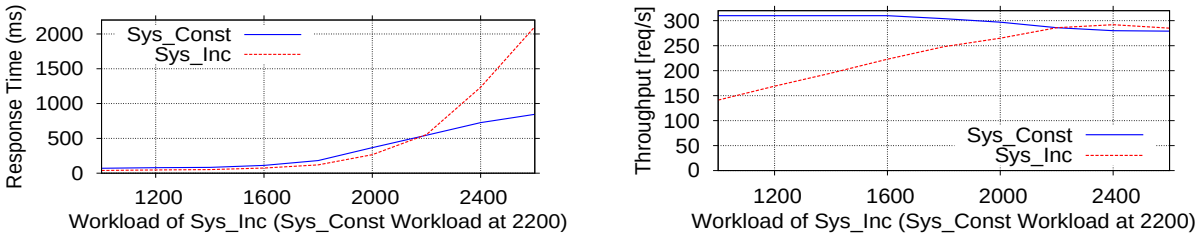
In this section, we experimentally investigate the issues associated with disk isolation in a virtualized environment. Most notably, in a virtualized environment with isolated disk resources, overall system performance can deteriorate because of a push back phenomenon stemming from specific software configuration settings [11] [12]. Concretely, our experimental investigations of co-located VMs with a shared disk reveal that sharing disk resources across VMs reduces the impact of performance interference more than isolating VMs does. This finding is consistent with the CPU performance interference scenario we showed in [8].

In our first experiment, *Sys_Const* has 2,200 concurrent users, while the workload on *Sys_Inc* is increased from 1,000 to 2,600 concurrent users in steps of 200 users. In addition, we adopt a full disk sharing strategy across VMs. The average response time and throughput of *Sys_Const* and *Sys_Inc* are shown in Figure 2a and Figure 2b respectively. The throughput of *Sys_Const* decreases by less than 10%, while throughput rates are largely linear and proportional to the system workload for *Sys_Inc*. Meanwhile, the average response time graph reveals more than a 1,100% increase for *Sys_Const* when the *Sys_Inc* workload increases from 1,000 to 2,600. As the graphs indicate, sharing disk resources lead to severe performance interference. We hypothesize that an isolation strategy would alleviate the disk contention among VMs thus preventing the observed performance interference.

In the following sub-sections, we investigate the possibility of avoiding performance interference entirely by applying a disk isolation strategy across VMs. To test this, we constrain the value of IOPs per VM, and we prevent the hypervisor from distributing disk resources according to its own disk allocation algorithm. Surprisingly, we found that specifying a specific disk allocation actually results in significantly lower performance than a fully-shared disk allocation. We attempt to explain causality by exploring the resource contention that occurs among co-located database systems, such as InnoDB. Because of their ACID properties, these database systems need to flush their log files from memory to disk frequently in order to maintain database consistency. Based on this understanding, we explore possible modifications to software configuration settings that could mitigate these previously mentioned performance penalties. Consequently, our analysis strives to make performance isolation in a virtualized environment more realistic in practice.

### A. Is Isolation better than Sharing?

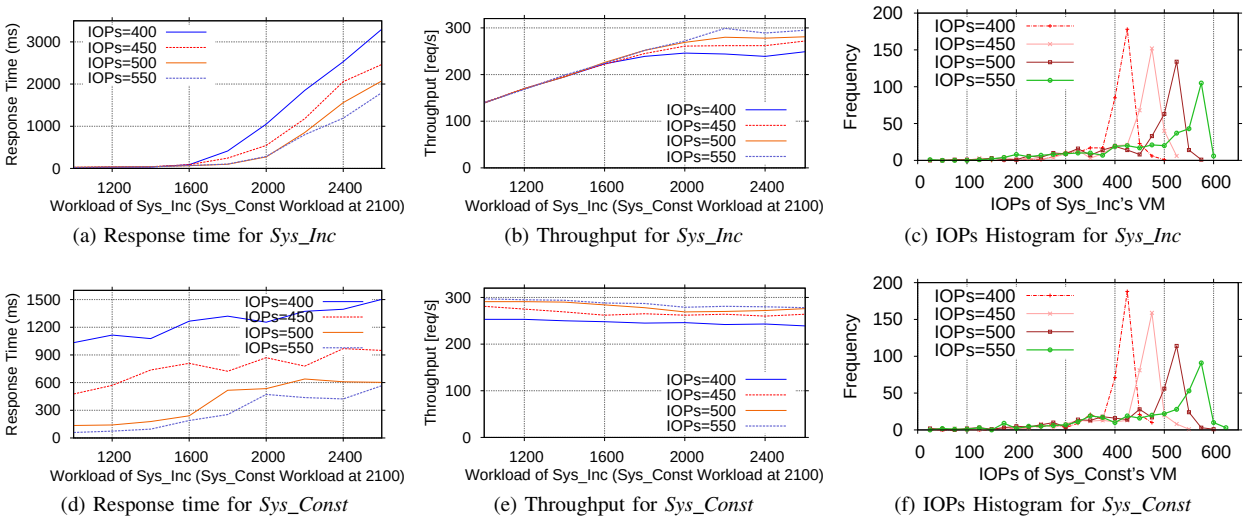
In this subsection, we investigate if performance interference among VMs can be improved by constraining the virtual machines' disk allocation. VMware ESXi 5 allows users to specify the disk IOPs (Input/Output Operations Per Second) for each VM. Methodologically, we consolidate two RUBBoS systems (i.e., *Sys_Inc* and *Sys_Const*) in our experimental testbed, and the IOPs limitation value is set to 400, 450, 500, 550 respectively.




(a) Response time comparison for Sys_Const and Sys_Inc

(b) Throughput comparison for Sys_Const and Sys_Inc

Fig. 2: Performance Interference between *Sys_Const* and *Sys_Inc* with configuration as Figure 1b. When the *Sys_Inc* workload increases from 1,000 to 2,600, *Sys_Const*'s throughput decreases by less than 10%; however, *Sys_Const* exhibits severe performance interference as its response time increases more than 1,100%.




(a) Response time for *Sys_Inc*

(b) Throughput for *Sys_Inc*

(c) IOPs Histogram for *Sys_Inc*

(d) Response time for *Sys_Const*

(e) Throughput for *Sys_Const*

(f) IOPs Histogram for *Sys_Const*

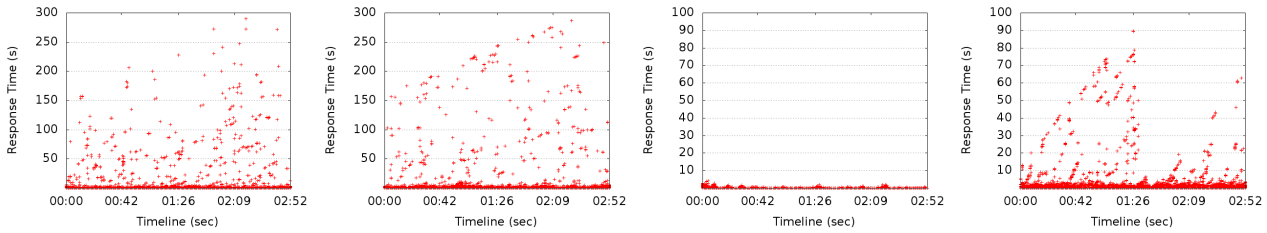
Fig. 3: Performance and IOPs histogram comparison for different IOPs constraint values. Sharing is better than isolation: the most right peaks (IOPs=550) in 3c and 3f show sharing allows a system to have more flexible IOPs when needed, while stricter constraints (IOPs=400) cause high peaks at left and prevent the hypervisor from using the under-utilized disk even if utilization is available.

As the names suggest, *Sys_Const* maintains a constant workload of 2,100 concurrent users, while the *Sys_Inc* workload is increased from 1,000 to 2,600 concurrent users in increments of 200 users.

Figure 3a and Figure 3b show the response time and throughput for *Sys_Inc* for four settings of the IOPs Limit equal to 400, 450, 500, 550 respectively. To be clear, the constraint, as indicated by the value of IOPs, is applied to the co-located DB nodes for both *Sys_Const* and *Sys_Inc*. Surprisingly, we found that the more restrictive the value of IOPs, the more *Sys_Inc's* performance degraded. On the other hand, Figure 3d and Figure 3e show response time and throughput trends for *Sys_Const* for different IOPs configurations. Since we kept the workload on *Sys_Const* constant, the non-constant variations are counter-intuitive; intuition says *Sys_Const*'s response time should have been constant. More surprisingly, these figures show at a glance that strictly constraining the value of IOPs degrades performance more than relaxing the value of IOPs does. Similar to the CPU findings shown in [8], disk sharing is a better strategy than isolation for mitigating potential disk contention.

The histograms, as shown in Figure 3c and Figure 3f, shows variation for different workloads, *Sys_Inc* and *Sys_Const* respectively, and different IOPs Limits (Limit=400, 450, 500 and 550).

When the IOPs Limit is equal to 400, IOPs values range from 350 to 450 with greater frequency than the other scenarios. When the IOPs Limit is set to 500




(a) 1,000 workload with Limit=400 (b) 2,600 workload with Limit=400 (c) 1,000 workload with Limit=550 (d) 2,600 workload with Limit=550

Fig. 4: Response time distribution for *Sys_Const* with different workload of *Sys_Inc* and value of Limit specified above. 4a and 4b exhibit similar distributions because isolation restricts the hypervisor's ability to maximize disk utilization. 4c shows under-utilized systems' hypervisors under a sharing strategy mitigate performance interference well, and comparing with 4b and 4d, sharing is still better than isolation in high workload scenarios.

and 550, peaks in the respective distributions occur at 50 IOPs above the corresponding limit value. These results can be explained by the hypervisor attempting to conform to the specified IOPs Limit value. This explanation also serves as the foundation for the hypothesis that applying stricter IOPs constraints prevents the hypervisor from utilizing the disk more fully–preventing the hypervisor from using the under-utilized disk to mitigate the impact of performance interference in a consolidated environment.

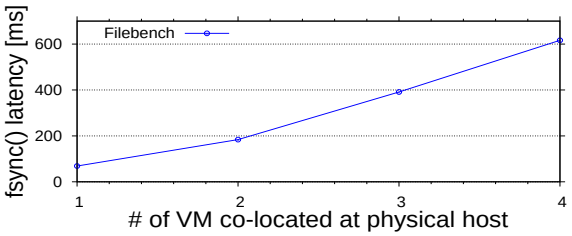
In addition, Figure 4 shows the response time distribution for *Sys_Const*. Figure 4a and Figure 4b shows the distribution with the Limit set to 400 when *Sys_Inc* has 1,000 and 2,600 concurrent users, while Figure 4c and Figure 4d display the response time distributions for the same *Sys_Inc* workloads but with a Limit at 550. Figure 4a and Figure 4b exhibit pretty similar response time distributions, which illustrate the isolation strategy restricting the hypervisor's ability to maximize disk utilization even when a co-located system has a low workload. Figure 4c shows that under a disk sharing allocation, hypervisors of under-utilized systems mitigate performance interference well. Finally, Comparing Figure 4b to Figure 4d demonstrates a sharing strategy is still better than isolation even under a high workload scenario.

### B. How does IO cause severe Performance Interference?

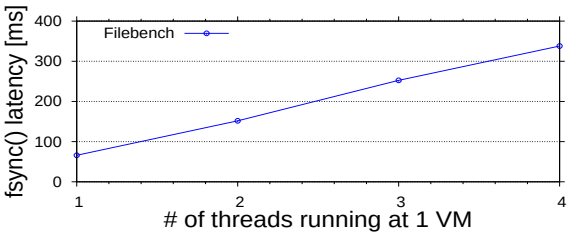
In this section, using file system-dependent database processes as a proxy for more general IO events, we illustrate the reasons why severe performance interference in a consolidated environment occurs. Commit record writing becomes a bottleneck when the number of committed transactions grows, and this has been proven to cause increased database processing time in dedicated environments [13]. ACID compliant storage engines, InnoDB for instance, flush logs from memory to disk frequently to maintain database consistency. To amortize IO cost, current database systems apply *Group Commit* to limit the number of disk writes to scale commit record writing. In the case of InnoDB, binary log (binlog) is used to record database changes such as table creation operations and updates to table data when transactions are committed. To avoid data loss and maintain consistency, InnoDB flushes binlog from memory to disk when it reaches the threshold of *Group Commit*, which is controlled by the parameter *sync_binlog*. In our setting, *sync_binlog* is set to 3, which means InnoDB will synchronize the binlog between memory and disk every third transaction commit. Our hypothesis is that co-located VMs flush data simultaneously and generate disk contention in the cloud.

In this paper, we enable the default InnoDB setting, which uses *fsync()* as the system call to flush log data from memory to disk. First, we investigate *fsync()* latency in a consolidated environment using Filebench, a file system and storage microbenchmark. To test if the consolidated environment experiences latency due to *fsync()*, we vary the number of co-located VMs and run Filebench on each VM simultaneously. Figure 5a shows the latency of *fsync()* for different numbers of co-located VMs. When the number of co-located VMs increases from 1 to 4, the latency of *fsync()* increases more than 600%, which proves our hypothesis that severe performance interference is due to disk contention among database systems in a consolidated environment.

Since the number of write threads for InnoDB is set to 4 by default, we would like to understand whether *fsync()* latency is also affected by the number of write threads. In this experiment, we vary the number of concurrent threads and run Filebench in the dedicated environment. Figure 5b shows the latency of *fsync()* for multiple

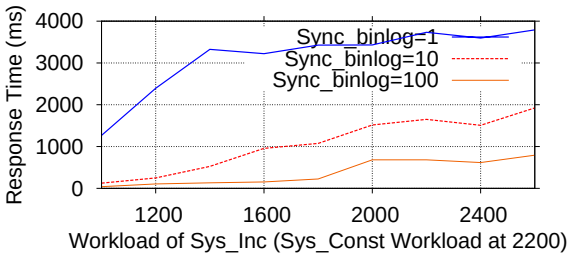



(a) *fsync()* latency vs number of co-located VMs in consolidated environment

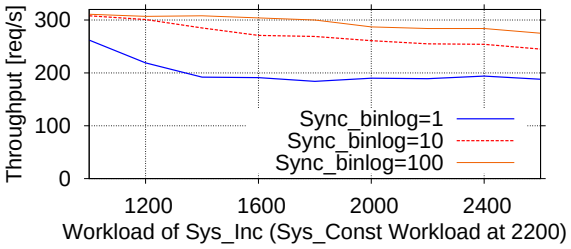



(b) *fsync()* latency vs number of threads in dedicated environment

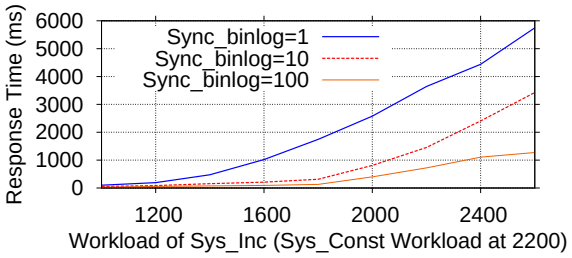
Fig. 5: With more co-located VMs or threads, *fsync()* latency increases due to IO contention.



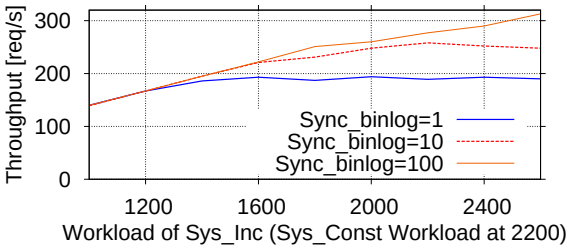

(a) Response time for *Sys_Const*




(b) Throughput for for *Sys_Const*




(c) Response time for *Sys_Inc*




(d) Throughput for *Sys_Inc*

Fig. 6: Performance comparison for *sync_binlog* settings. As InnoDB synchronizes its binlog less often, performance improves. For example, response time of *sync_binlog*=100 is over 400% less than *sync_binlog*=1 at 2,000 workload in 6a. Similar trends show in other graphes.

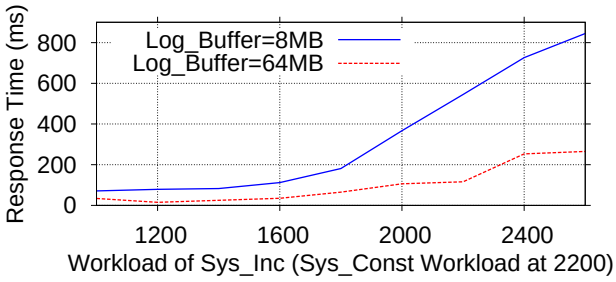
threads, and this figure exhibits a similar trend to the one in the consolidation scenario described above where we increased the number of concurrent threads from 1 to 4. This demonstrates disk contention not only occurs in co-located VMs but also among multiple threads.

Furthermore, by varying the binlog synchronization *sync_binlog* of InnoDB, we find additional evidence to support the hypothesis that flushing log data from memory to disk contributes to performance interference in n-tier applications. Figure 6a and Figure 6b show the response time and throughput of *Sys_Const*, while Figure 6c and Figure 6d display the same metrics for *Sys_Inc*. For different *sync_binlog* settings, 1, 10 and 100 specifically, the performance loss for both workloads improves as InnoDB synchronizes its binlog less often.

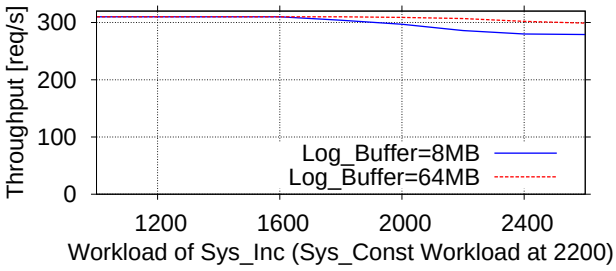
### C. How to mitigate IO Performance Interference?

So far, we have shown that perfect isolation separates the VMs so their commit record writing cannot be synchronized. Conversely, sharing allows commit records to be written interleaved, optimizing disk I/O. In this section, we show that by adjusting the database settings, we further reduce the resource contention among VMs and make an n-tier system less susceptible to performance interference caused by consolidating systems. Because of space limitations, we only present the performance comparison for *Sys_Const*; however, the observations and results related to *Sys_Const* hold for *Sys_Inc*.

In Linux, bigger buffer facilitates larger numbers of commit records to be written in a single group commit *fsync*. Hence, we increase the InnoDB *inn-*

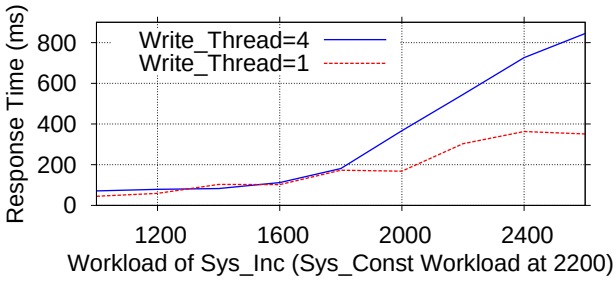



(a) Response Time for *Sys_Const*

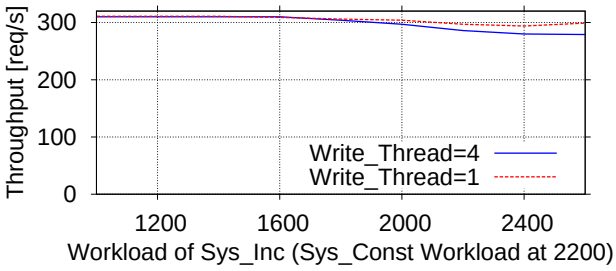



(b) Throughput for *Sys_Const*

Fig. 7: Performance Comparison for different log buffer size. Bigger size enables more records to be written in a single group commit *fsync*, mitigating IO contention.




(a) Response Time for *Sys_Const*




(b) Throughput for *Sys_Const*

Fig. 8: Performance Comparison for different number of threads. Decreasing number of threads alleviates disk contention.

*odb_log_buffer_size* from the default of 8MB to 64MB. In our experiment, we found that increasing the *innodb_log_buffer_size* reduces the number of times the log is flushed. In other words, the database can commit more records with one *fsync()* system call. From Figure 7a and Figure 7b, we observe that the larger log buffer size helps *Sys_Const* keep the throughput constant. Most importantly, the response time for *Sys_Const* is cut in half when the workload of *Sys_Inc* increases from 1,000 to 2,600 concurrent users.

Secondly, we reduce the InnoDB *innodb_write_io_threads* from the default of 4 to 1. Although the InnoDB manual claims that increasing the *innodb_write_io_threads* improves the disk utilization and overall database performance, we found multiple InnoDB write threads actually increases the resource contention among VMs and deteriorates the system performance in a consolidated environment. Figure 8a and Figure 8b show that reducing the number of write threads per VM helps to relieve disk contention, and it improves the response time of *Sys_Const* by a factor of four even when *Sys_Inc* has a high workload (e.g., 2,600 concurrent users).

## IV. RELATED WORK

One of the important features of ACID compliant databases, commit, has been known to increase database processing time because of the incurred disk access time [13]. Remesh et al. [14] proposed the OPT protocol to improve the performance of commit over standard 2PC and 3PC algorithms, while David et al. presented a system that reduces the transaction overhead by replacing the standard redo log with recoverable memory [15]. With Solid State Drives (SSD), transaction throughput has been improved by a factor of 3.5 over traditional hard disk drive systems [16].

With the emergence of cloud computing, server virtualization and consolidation have become active research topics due to the complexity and economics of large-scale computer systems. While analytical approaches assume a linear impact on performance because of consolidation and apply performance prediction and management models accordingly [17], [2], experimental approaches have shown performance interference among consolidated applications in virtualized environments [8], [11]. By comparing standalone virtual machines under different consolidation scenarios, Indrani et al. observed performance variations of 25-65% for database servers and 7-40% for file servers [18].

Cloud computing platforms because of their promise of infinite scalability and high availability provide great opportunities to deploy large scale database services [19], [20]. However, the performance of parallel database servers in cloud environments is still difficult to predict

due to non-trivial scalability characteristics [21]. Barzan et al. [22] discussed these and other challenges, including database service pricing, database tenant isolation and performance optimization, that people encountered when hosting databases in the cloud. Furthermore, Kanemasa et al. found out sharing is better than isolation for CPU scenario [8]. This last paper corroborates our research that attempts to understand the performance interference for different resource contentions in consolidated virtualization environments.

## V. CONCLUSION

This paper provides an in-depth experimental study of issues related to performance interference of cloud-based, consolidated n-tier applications. To summarize, we have demonstrated a sharing-based resource allocation strategy is better than isolation for avoiding IO contention. Using response time as a metric, we investigated performance interference brought about by disk contention. Perfect isolation separates the VMs, so their commit record writing cannot be synchronized, while sharing allows their commit records to be written interleaved, optimizing disk I/O. To mitigate the performance interference among VMs, we found a bigger buffer enables more commit records to be written in a single group commit *fsync*, and decreasing the number of threads alleviates IO contention. Given the importance of VM consolidation to cloud environments, further research in this area is needed to improve our understanding of the performance interference phenomenon and to find solutions that can effectively propel migration of mission-critical applications into clouds.

## ACKNOWLEDGMENT

This research has been partially funded by National Science Foundation by CNS/SAVI (1250260, 1402266), IUCRC/FRP (1127904) , CISE/CNS (1138666), NetSE (0905493) programs, and gifts, grants, or contracts from Singapore Government, and Georgia Tech Foundation through the John P. Imlay, Jr. Chair endowment. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding agencies and companies mentioned above.

## REFERENCES

[1] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *Int. CMG Conference*, 2007, pp. 399–406.

[2] X. Meng, C. Isci, J. O. Kephart, L. Zhang, E. Bouillet, and D. E. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *ICAC*, 2010, pp. 11–20.

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

[4] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, C. Pu, and Y. Cao, "Who is your neighbor: Net i/o performance interference in virtualized clouds," *IEEE T. Services Computing*, vol. 6, no. 3, pp. 314–329, 2013.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," Tech. Rep., 2009.

[6] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforcing performance isolation across virtual machines in xen," in *Middleware*, 2006, pp. 342–362.

[7] K. Ye, X. Jiang, D. Ye, and D. Huang, "Two optimization mechanisms to improve the isolation property of server consolidation in virtualized multi-core server," in *HPCC*, 2010, pp. 281–288.

[8] Y. Kanemasa, Q. Wang, J. Li, M. Matsubara, and C. Pu, "Revisiting performance interference among consolidated n-tier applications: Sharing is better than isolation," in *IEEE SCC*, 2013, pp. 136–143.

[9] "Rubbos: Rice university bulletin board system," http://jmob.ow2.org/rubbos.html.

[10] Q. Wang, S. Malkowski, D. Jayasinghe, P. Xiong, C. Pu, Y. Kanemasa, M. Kawaba, and L. Harada, "The impact of soft resource allocation on n-tier application scalability," in *IPDPS*, 2011, pp. 1034–1045.

[11] S. Malkowski, Y. Kanemasa, H. Chen, M. Yamamoto, Q. Wang, D. Jayasinghe, C. Pu, and M. Kawaba, "Challenges and opportunities in consolidation at high resource utilization: Non-monotonic response time variations in n-tier applications," in *IEEE CLOUD*, 2012, pp. 162–169.

[12] Q. Wang, Y. Kanemasa, J. Li, D. Jayasinghe, T. Shimizu, M. Matsubara, M. Kawaba, and C. Pu, "An experimental study of rapidly alternating bottlenecks in n-tier applications," in *IEEE CLOUD*, 2013, pp. 171–178.

[13] H. Garcia-Molina and K. Salem, "Main memory database systems: An overview," *IEEE Trans. Knowl. Data Eng.*, vol. 4, no. 6, pp. 509–516, 1992.

[14] R. Gupta and J. Haritsa, "Commit processing in distributed real-time database systems," in *Proceedings of 17th IEEE Real-Time Systems Symposium*, 1996.

[15] D. E. Lowell and P. M. Chen, "Free transactions with rio vista," in *SOSP*, 1997, pp. 92–101.

[16] S.-W. Lee, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim, "A case for flash memory ssd in enterprise database applications," in *SIGMOD Conference*, 2008, pp. 1075–1086.

[17] A. Ashraf, "Cost-efficient virtual machine provisioning for multi-tier web applications and video transcoding," in *CCGRID*, 2013, pp. 66–69.

[18] I. Paul, S. Yalamanchili, and L. K. John, "Performance impact of virtual machine placement in a datacenter," in *IPCCC*, 2012, pp. 424–431.

[19] C. Curino, E. P. C. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich, "Relational cloud: a database service for the cloud," in *CIDR*, 2011, pp. 235–240.

[20] A. Arasu, S. Blanas, K. Eguro, M. Joglekar, R. Kaushik, D. Kossmann, R. Ramamurthy, P. Upadhyaya, and R. Venkatesan, "Secure database-as-a-service with cipherbase," in *SIGMOD Conference*, 2013, pp. 1033–1036.

[21] D. Jayasinghe, S. Malkowski, Q. Wang, J. Li, P. Xiong, and C. Pu, "Variations in performance and scalability when migrating n-tier applications to different clouds," in *IEEE CLOUD*, 2011, pp. 73–80.

[22] B. Mozafari, C. Curino, and S. Madden, "Dbseer: Resource and performance prediction for building a next generation database cloud," in *CIDR*, 2013.