Revisiting Performance Interference among Consolidated n-Tier Applications: Sharing is Better than Isolation

Yasuhiko Kanemasa*, Qingyang Wang†, Jack Li†, Masazumi Matsubara*, and Calton Pu†
*System Software Laboratories, FUJITSU LABORATORIES LTD., Kawasaki, Japan
{kanemasa, matz}@jp.fujitsu.com
†College of Computing, Georgia Institute of Technology, Atlanta, GA, USA
{qywang, jack.li, calton}@cc.gatech.edu

***Abstract*—Performance unpredictability is one of the major concerns slowing down the migration of mission-critical applications into cloud computing infrastructures [4]. An example of non-intuitive result is the measured n-tier application performance in a virtualized environment that showed increasing workload caused a competing, co-located constant workload to decrease its response time [12]. In this paper, we investigate the sensitivity of measured performance in relation to two factors: (1) consolidated server specification of virtual machine resource availability, and (2) burstiness of n-tier application workload. Our first and surprising finding is that specifying a complete isolation, e.g., 50-50 even split of CPU between two co-located virtual machines (VMs) results in significantly lower performance compared to a fully-shared allocation, e.g., up to 100% CPU for both co-located VMs. This happens even at relatively modest resource utilization levels (e.g., 40% CPU in the VMs). Second, we found that an increasingly bursty workload also increases the performance loss among the consolidated servers, even at similarly modest utilization levels (e.g., 70% overall). A potential solution to the first problem (performance loss due to resource allocation) is cross-tier-priority scheduling (giving higher priority to shorter jobs), which can reduce the performance loss by a factor of two in our experiments. In contrast, bursty workloads are a more difficult problem: our measurements show they affect both the isolation and sharing strategies in virtual machine resource allocation.**

***Keywords*-application co-location; cloud; consolidation; experimental study; n-tier; performance; interference; RUBBoS.**

I. Introduction

Server consolidation (or simply *consolidation*) through hardware virtualization technology is a fundamental technique for achieving economical sharing of computing infrastructures as computing clouds. Consolidated servers run on the same physical node in dedicated Virtual Machines (VMs) to increase overall node utilization, which increases profit by reducing operational costs such as power consumption. Unfortunately, the effectiveness of consolidation seems limited in practice, as shown by reportedly low utilization levels in production data centers [5], [19]. One of the recognized problems is the performance unpredictability in virtualized environments (ranked fifth in the top 10 obstacles for growth of cloud computing [4]). Performance interferences among consolidated applications have been demonstrated for a variety of concrete systems and applications [3], [8], [11], [15].

A representative example of unexpected phenomena in consolidated environments is the non-monotonic response time variations [12], where measured response time *decreases* with *increasing* workload at high CPU utilization levels. This kind of surprising result illustrates the interesting challenges and opportunities in consolidated application performance research. In this paper, we describe detailed measurements to study two factors that affect consolidated application performance significantly: (1) consolidated server specification of virtual machine resource availability, and (2) burstiness of n-tier application workload.

One of the common assumptions made in consolidation research is that the hypervisor is able to provide perfect isolation of CPU allocated to each virtual machine (VM), e.g., see [10], [20]. Since previous work such as [12] have shown this assumption to be valid only partially, it became important to investigate the behavior of hypervisor CPU schedulers, particularly how the specification of VM resource allocation strategies affect performance isolation among the VMs. As a concrete example of this assumption: with 2 VMs sharing a CPU, an isolation specification of 50% allocation for each VM would eliminate the performance interference problem (In this paper *performance interference* is used synonymously with ”performance loss due to interference”). Perhaps unsurprising to many practitioners, our experiments show that such a “disjoint” (isolation) VM CPU allocation strategy can result in large response time degradation even under relatively low utilization in the consolidated system (e.g., starting at 40% of assigned CPU).

The first contribution of the paper is an experimental confirmation that the performance loss due to interference in a consolidated system can be improved by a “sharing” allocation strategy. In the above 2-VM example, instead of 50% allocation for each VM, the sharing allocation allows up to 100% of CPU for each VM. This appears to be an anti-intuitive result, since the sharing allocation seems to prevent the hypervisor scheduler from providing resource isolation among the consolidated VMs. A deeper analysis of our experimental results shows that the disjoint allocation loses because it is a "selfish" strategy in which an idle VM prevents the CPU from doing any work because of its own reservation. In contrast, the sharing allocation strategy wins by letting each VM use the CPU during other VM's idle time, which is particularly effective at higher utilization levels when VMs are likely to have queued requests. As a potential solution to the performance interference problem found, we measured the system performance (e.g., response time) under the *cross-tier-priority (CTP) based scheduling* [22]. We found that CTP can reduce the performance loss due to interference, up to a factor of two. These experiments show that improved scheduling could lead to significant reduction and therefore represents interesting topics of future research.

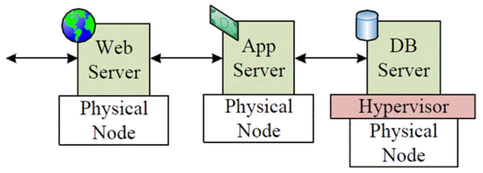
The second factor that may cause performance interference during consolidation is bursty workload, a well-known challenge of web-facing applications [2], [14]. The second contribution of the paper is a detailed experimental study of the impact of bursty workloads on the performance of consolidated applications. Our measurements show that bursty workloads degrade consolidated system response time at normally considered safe CPU utilization of 70% overall. Furthermore, the negative impact of bursty workloads is insensitive to CPU allocation of VMs, affecting both the isolation and sharing strategies.

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. Subsequently, Section III compares several CPU limiting configurations and shows sharing is better than isolation for consolidated n-tier applications. Section IV shows the impact of bursty workload on the performance interference. Section V provides an overview of related work in this area. Finally, Section VI concludes the paper.

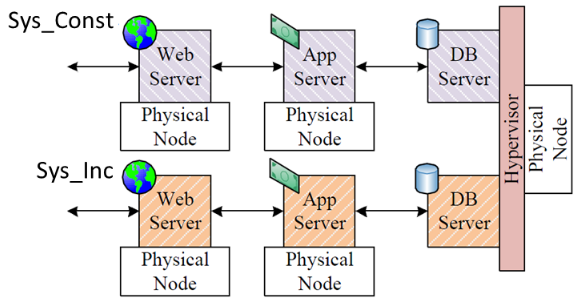
II. Experimental Setup

While consolidation in practice may be applied to any type of application, the focus of this paper are n-tier applications with LAMP (Linux, Apache, MySQL, and PHP) implementations. Typically, n-tier applications are organized as a pipeline of servers[1], starting from web servers (e.g., Apache), through application servers (e.g., Tomcat), and ending in database servers (e.g., MySQL). This organization, commonly referred to as n-tier architecture (e.g., 3-tier in Figure 1a), serves many important web-facing applications. In our experiments, we use the popular n-tier application benchmark system RUBBoS [1]. Due to space constraints, we solely focus on results obtained with a browsing-only workload set. Our default experiment ramp-up and run-times are 180 and 300 seconds, respectively.

[1]In this paper *server* is used in the sense of computer programs serving client requests. Hardware is referred to as *physical computing node* or *node* for short.



(a) Dedicated deployment of a 3-tier application system with three software servers (i.e., web, application, and database) and three physical hardware nodes.



(b) Consolidated deployment of two 3-tier systems (`Sys_Const` and `Sys_Inc`) with one server per tier and five physical hardware nodes in total. The DB servers are co-located in dedicated VMs on a single shared physical hardware node.

Figure 1: Example of a dedicated (a) and a consolidated (b) 3-tier application system deployment, presented as mappings of software servers to physical hardware nodes.

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

CPU	Quad Xeon 2.27GHz * 2 CPU (HT)
Memory	12GB
HDD	SAS, 10,000RPM, 300GB * 2 (RAID1)
OS	RHEL Server 5.3 (Tikanga), 32-bit
OS Kernel	2.6.18-128.el5PAE
Web Server	HTTPD-2.0.54
App Server	Apache-Tomcat-5.5.17
Connector	Tomcat-Connectors-1.2.28-src
DB Server	MySQL-5.0.51a-Linux-i686-glibc23
Java	JDK1.6.0_23
Monitoring Tools	Esxtop
Hypervisor	**VMware ESXi v5.1.0**
Guest OS	RHEL Server 5.3, 32-bit
Guest OS Kernel	2.6.18-8.el5

At an abstract level, the deployment of n-tier applications in a cloud computing infrastructure can be modeled as a mapping between component servers 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. In contrast, 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 physical node (e.g., Figure 1b). In our experiments, we denote the first RUBBoS system as `Sys_Const` and the

Table II: Configurations of major software resources

Tier	Parameter name	Configuration name	
		Liberal-SR	**Conservative-SR**
Apache	MaxClients	340	120
	ThreadsPerChild	170	60
	WorkerConnection PoolSize	100	25
Tomcat	maxThreads	240	60
DB conncetions	total #	96	16
	# in each Servlet	12	2

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) both VMs to the same CPU core (i.e., the maximum processing power totally allocated to the VMs is 2266MHz) and limit both virtual CPUs to 1133, 1360, 1586 or 2266MHz (i.e., 50%, 60%, 70% or 100% of 2266MHz) with a reservation of 0.00MHz and normal shares (i.e., both VMs have the same priority). Other important characteristics of our experimental testbed are summarized in Table I.

Software configuration, in the context of our work, refers to software settings that specify how many *software resources* such as processes, threads or DB connections in servers are allocated. Both systems are configured with practical resource allocation settings that are derived from our previous experiments with the same consolidation scenario [12]. The two configurations of software resource allocation adopted for the systems are summarized in Table II. Both of the two configurations were validated in the previous experiments to achieve good total performance if used with the dedicated n-tier deployment.

In the following sections, we use the notation $Sys\_Const[SRconf1], Sys\_Inc[SRconf2], Limit = X\%$ to show an experimental configuration of `Sys_Const` and `Sys_Inc` with software resource configurations *SRconf1* and *SRconf2* in Table II respectively, and the maximum CPU allocated to each VM is limited to *X%* of the core clock.

III. Performace Interference: Sharing Is Better Than Isolation

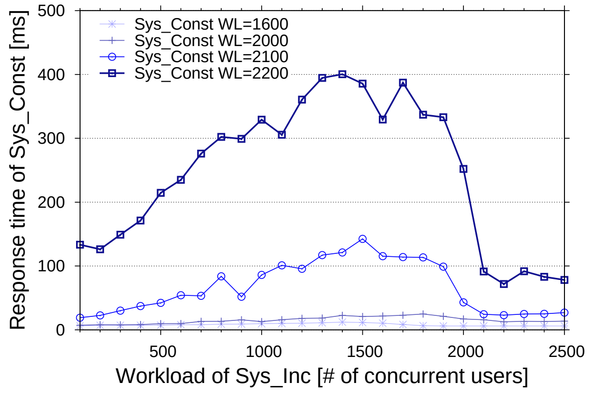
As shown in our previous study, the non-monotonic response time variations are mainly caused by a hypervisor's CPU scheduling artifacts and appear in consolidation scenarios under high CPU utilization [12]. In this section our further experimental investigations reveal that, in order to reduce the impact of the performance interference, sharing on resource allocation is better than isolation in a virtualized infrastructure. In Subsection III-A we show the disjoint allocation (splitting CPU fifty-fifty between two systems) cannot remove the performance interference. Then, in Subsection III-B we compare several configurations of CPU sharing (concretely "*Limit*" of CPU on VMware hypervisor) to show that a highly shared configuration is better than a humbly shared configuration to reduce the impact of performance interference. Finally, in Subsection III-C we show our previously proposed cross-tier-priority based scheduling is successfully applied and effective as a solution to mitigate the impact of the performance interference.

A. Can we Avoid Performance Interference by Removing CPU Sharing?

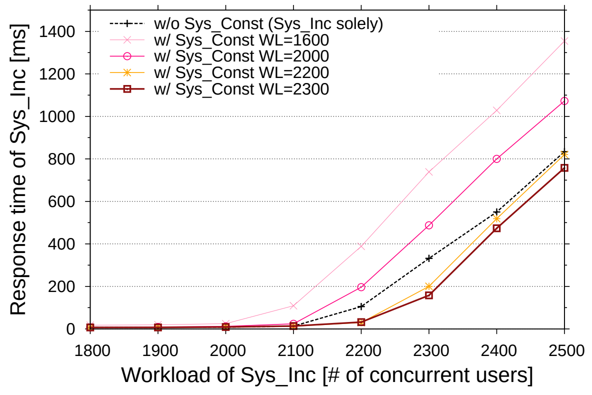
In Figure 2 we show a set of experimental results with the disjoint CPU allocation strategy (*Limit*=50%) to reveal whether the performance interference caused by consolidated n-tier systems can be avoided with the isolation-oriented configuration. In this configuration, the processing power of a CPU core is split fifty-fifty (i.e., 1133MHz each) and assigned to each VM of the consolidated two n-tier systems `Sys_Const` and `Sys_Inc`, which are deployed as shown in Figure 1b. In all experiments in this paper, the amount of workload on `Sys_Const` is always kept constant while the amount of workload on `Sys_Inc` is increased monotonically up to 2,500 concurrent users in steps of 100 users as labeled on the x-axis. Here the response times of each system under varying workloads on `Sys_Const` are separately shown in Figure 2a (`Sys_Const`) and Figure 2b (`Sys_Inc`). Figure 2b also contains the response times of `Sys_Inc` executed on a dedicated deployment which is shown in Figure 1a (i.e., Only `Sys_Inc` receives a workload while `Sys_Const` receives no workload).

Figure 2a shows that the response time degradation also appears in this case contrary to the motivation (i.e., isolation) of such a configuration. Furthermore, the increment starts from low workload on the consolidated `Sys_Inc` until the `Sys_Inc`'s workload exceeds `Sys_Const`'s amount. This result, combined with the knowledge revealed in our previous research [12], shows that the disjoint allocation is unable to prevent the disruption of job-execution on `Sys_Const`'s VM caused by the low priority on CPU scheduling, which gives higher priority to the VM with averagely lower CPU utilization. This disruption causes job processing to be delayed and results in the large degradation of overall response time in the system.

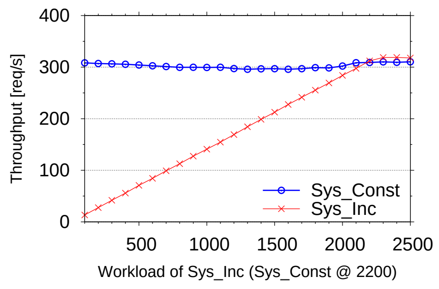
On the other hand, what needs to be emphasized in Figure 2b is that `Sys_Inc` achieves the best response time with the highest workload on the consolidated system among the five cases in the graph. Comparing the two cases with workload 1600 and 2300 on the consolidated `Sys_Const`, surprisingly, `Sys_Inc`'s response time is much better under a higher workload of 2300 than under the lower workload of 1600. In other words, `Sys_Inc` consolidated with `Sys_Const` under WL=2300 provides short response time to a 10% higher number of concurrent users than that consolidated with `Sys_Const` under WL=1600. This is contrary to the intuitive understanding that performance decreases as the total workload on a node increases. Even more surprisingly, `Sys_Inc` consolidated



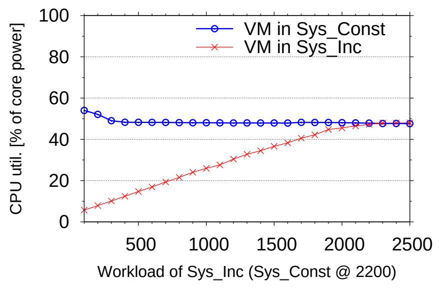
(a) *Response time of* `Sys_Const` *according to its own workload. The disjoint allocation cannot avoid the performance interference caused by a consolidated system, which starts under low workload on the consolidated system and appears as the non-monotonic response time variations.*



(b) *Response time of* `Sys_Inc` *according to the workload on the consolidated* `Sys_Const`. *Contrary to intuitive understanding,* `Sys_Inc` *with high workloads (2200 and 2300) on the consolidated* `Sys_Const` *achieves better performance than that with lower workloads (1600 and 2000) on the consolidated side and even better than that without* `Sys_Const`.



(c) *Throughput of each system.*



(d) *CPU utilization of the VM in each system.*

Figure 2: Each response time (a)(b), throughput (c), and each VM's CPU utilization (d) of the two consolidated 3-tier systems `Sys_Const` and `Sys_Inc` which are deployed as shown in Figure 1b. The workload of `Sys_Const` is kept constant, while the workload of `Sys_Inc` is increased monotonically as show in the x-axis. The *Limit* values of both VMs are 50% of CPU clock speed (that is, disjoint allocation). `Sys_Const` [Conservative-SR], `Sys_Inc` [Liberal-SR], *Limit*=50%.

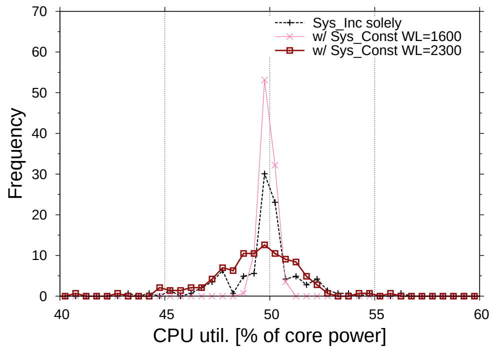
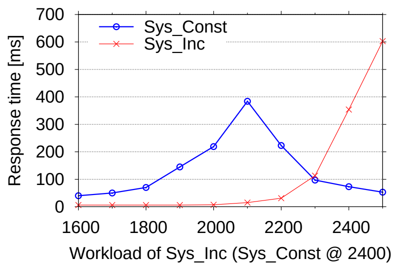


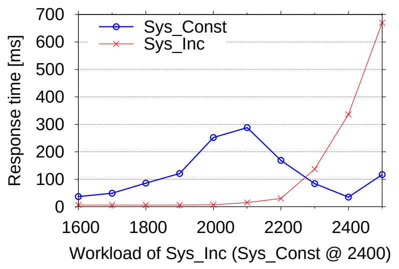
Figure 3: Histogram of CPU utilization on the VM in `Sys_Inc` at workload=2200. This comparison shows that the CPU capping for `Sys_Inc`'s VM is relaxed when the workload of the consolidated side is high at 2300. The flexible allocation of more CPU results in the shorter response time than other cases in Figure 2b.

with `Sys_Const` WL=2300 achieves slightly better response time than that on a dedicated deployment.

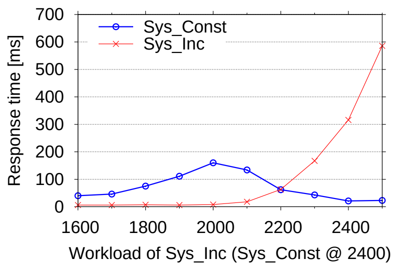
In order to investigate the reason why such an inversion phenomenon happens, the histograms of CPU utilization in the cases are compared in Figure 3 (the workload on `Sys_Inc` is at 2200). These CPU utilization data are collected by Esxtop every 2 seconds during a 300 second run-time (totally about 150 samples in an experiment). The histograms show that `Sys_Inc` is nearly capped at 50% of CPU power when it is executed on a dedicated deployment, while it is more strictly capped in the case consolidated with `Sys_Const` WL=1600 (i.e., lower workload than that on `Sys_Inc`). These are natural results of the hypervisor's CPU scheduling to keep the *Limit*=50% configuration. In the case consolidated with `Sys_Const` WL=2300, on the other hand, it is observed that the CPU capping for `Sys_Inc` is relaxed, that is, it can utilize more than 50% CPU power frequently. The combination of the following two factors can better explain this result: 1) `Sys_Inc`'s VM and `Sys_Const`'s VM have the same CPU scheduling priority since their workloads are similar, 2) The hypervisor needs to relax the capping in order to keep a fair CPU allocation



(a) Limit=60%



(b) Limit=70%



(c) Limit=100%

Figure 4: Comparing the response time of the consolidated `Sys_Const` and `Sys_Inc` with three *Limit* values of 60% (a), 70% (b), and 100% (c). The workload of `Sys_Const` is kept constant at 2,400 while the workload of `Sys_Inc` is increased monotonically from 1,600 to 2,500. This comparison shows that the performance interference can be mitigated with a highly shared configuration (100%). `Sys_Const` [Liberal-SR], `Sys_Inc` [Conservative-SR].

Table III: Average CPU utilization of each VM during 5 minutes run-time of the experiments with workload `Sys_Const`=2400 and `Sys_Inc`=2100, which correspond the high peaks in Figure 4a, 4b and 4c.

Limit	VM in `Sys_Const` [%]	VM in `Sys_Inc` [%]
60%	51.63	48.20
70%	51.57	48.29
100%	51.91	47.58

between the two VMs under the congested CPU utilization situation.

B. CPU Sharing can Mitigate the Performance Interference

The previous subsection shows that the disjoint allocation of CPU allocation between the VMs on two consolidated n-tier systems cannot remove the performance interference caused by the consolidated system. Instead, it results in large response time degradation due to its limited flexibility on CPU allocation which is strictly capped at a certain amount. Next in this subsection, we compare three configurations of CPU sharing with *Limit*=60%, 70% and 100%[2] to show that resource sharing works well to mitigate the impact of the performance interference in the consolidated deployment scenario. In fact, the fully-shared configuration with *Limit*=100% achieves better performance than other humbly shared configurations with *Limit*=60% and 70%.

Figure 4 shows the response time of each system in the three configurations with *Limit*=60%, 70% and 100%. In these experiments, the amount of workload on `Sys_Const` is constant at 2,400 concurrent users, while the amount of workload on `Sys_Inc` is increased monotonically from 1,600 to 2,500 concurrent users. The figures show at a glance that all the three cases have non-monotonic variations of response time, which peak around workload 2000 or 2100 on `Sys_Inc`. Since we keep the amount of workload on `Sys_Const` constant, the non-monotonic variations are

[2]On the hypervisor, we specify 1360MHz, 1586MHz, and 2266MHz respectively as the *Limit* values.

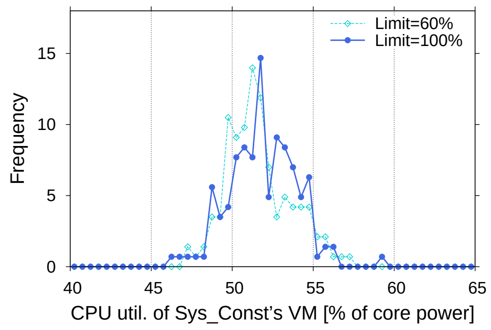


Figure 5: Comparing the histograms of CPU utilization on the VM in `Sys_Const` (workload=2400) consolidated with the VM in `Sys_Inc` (workload=2100) under two different *Limit* values 60% and 100%. With *Limit*=100%, the VM in `Sys_Const` can use over 50% CPU power more frequently than 60% case, and it achieves better response time as compared in Figure 4a and 4c.

contrary to the intuitive understanding that the response time of `Sys_Const` should be kept constant. As explained in the previous subsection (and also in our previous research [12]), these non-monotonic response time variations are caused by performance interference between consolidated systems which is a result of the hypervisor's CPU scheduling. In the CPU scheduling, a system which on average a higher workload than the other consolidated system gets lower CPU allocation priority compared to the consolidated one.

In comparison to the results in Figure 2a, Figure 4 clearly shows that the response time degradation only appears at relatively high workload on the consolidated `Sys_Inc` in this shared CPU configuration. Furthermore, simply judging based on the small samples, it seems like a large *Limit* value always leads to small response time degradation. As shown in Table III, however, the average CPU utilization of both VMs under workload `Sys_Const`=2400 and `Sys_Inc`=2100 with each *Limit* value do not have any differences. To figure out the reason for the performance differences among the three configurations, we compare the histograms of CPU utilization on `Sys_Const`'s VM with two *Limit* values 60% and 100% in Figure 5. Comparing

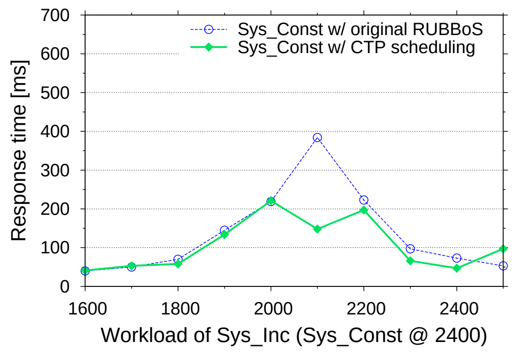


Figure 6: Cross-tier-priority based scheduling can mitigate the performance interference caused by a consolidated VM. These experiments correspond to those in Figure 4a, that is, `Sys_Const` [Liberal-SR] with constant workload at 2400, `Sys_Inc` [Conservative-SR], *Limit*=60%.

these two cases, *Limit*=100% case records CPU utilization at the range between 53% and 55% more frequently than *Limit*=60% case. This occurs because the configuration can utilize more than 60% CPU power when the workload of the consolidated VM is instantaneously low.[3] This result denotes that, with higher sharing configuration, VMs can utilize the CPU more flexibly and can assign the extra CPU power to mitigate the impact of performance interference caused by a consolidated VM.

C. Another Approach of Reducing Performance Interference with CTP Scheduler

So far we have shown that the performance of an n-tier system can be significantly interfered by its consolidated systems in a non-trivial way and the magnitude of interference depends on some system level factors such as CPU sharing/limiting strategies. The root cause of the performance interference is due to the "unfair" resource (e.g., CPU) contention between consolidated VMs when resource utilization is high. To solve this problem, one possible solution is to change the current hypervisor scheduling algorithm and make efforts to provide perfect isolation among VMs. We will leave it as one of our future research.

In this section, we show another practical approach to make an n-tier system more robust to the performance interference cause by consolidated systems. Our previous work [22] introduced a technique named *cross-tier-priority based scheduling* (CTP scheduling) which can mitigate the large response time fluctuations of an n-tier system under high hardware resource utilization. Here we show that the CTP scheduling also works well in the case of the performance interference caused by resource contention among VMs.

[3]It should be noted that the granularity of recording CPU utilization in the results is 2 seconds since it is the minimum value of the Esxtop command. So, comparing with the result of the *Limit*=60% case, it is reasonable to consider that the 53-55% are the results of aggregation among instantaneous values of CPU utilization higher than 60% and low values.

CTP scheduling is essentially an extension of applying the shortest-job-first scheduling policy for a web server [9], [16] in the context of n-tier systems. It categorizes transaction[4] types in an n-tier system into several groups according to the total service time one transaction takes in the bottleneck tier. Transactions with short service times (i.e., light transactions) are granted with a higher priority to execute while transactions with long service times (i.e., heavy transactions) are granted with a lower priority.[5] By giving higher priority to light transactions during transaction execution, the CTP scheduling prevents light transactions from being involved in the long waiting time for heavy transactions during transient saturation of hardware resources in the bottleneck tier. Overall, CTP scheduling can decrease the influence caused by transient saturation of a hardware resource and achieve stable, good response time of the system. In our VM consolidation scenario, the CPU resource of a VM is frequently limited by the consolidated VM due to resource contention; thus applying CTP scheduling can mitigate the negative interference from the consolidated VM and achieve a more stable response time.

Figure 6 shows the comparison between the response time of the original RUBBoS and the modified RUBBoS with CTP scheduling during transaction execution. At workload=2100 on `Sys_Inc` in the figure, CTP scheduling decreases the high peak of response time caused by the performance interference from the consolidated system.

IV. Bursty Workloads Increase Performance Interference

In this section, we show how the burstiness of the workload on a consolidated system can cause performance interference and impacts the non-monotonic response time variations. Our detailed experimental observation reveals that the response time of even an under-utilized n-tier system can be largely degraded when the workload on a consolidated system becomes bursty. This fact suggests that it is quite difficult to achieve stable response times in a consolidated deployment scenario on a virtualized infrastructure since the burstiness of workload on a consolidated system is unavailable or unpredictable in many cases.

Bursty workloads are very common in real world n-tier web-facing applications. For instance, the popular term *Slashdot effect* describes a phenomenon where a web page linked by a popular blog or media site suddenly experiences a huge increase in web traffic [2]. Mi et al. [14] proposed a bursty workload generator which takes into account the Slashdot effect. The bursty workload generator uses one

[4]A transaction services an entire web page requested by a client and may consist of multiple interactions between different tiers.

[5]Whether a transaction is heavy or light is application level knowledge; the operating system in each individual server cannot distinguish heavy transactions from light transactions, thus traditional OS-level shortest-job-first scheduling policy does not improve performance here. See [22] for more details.

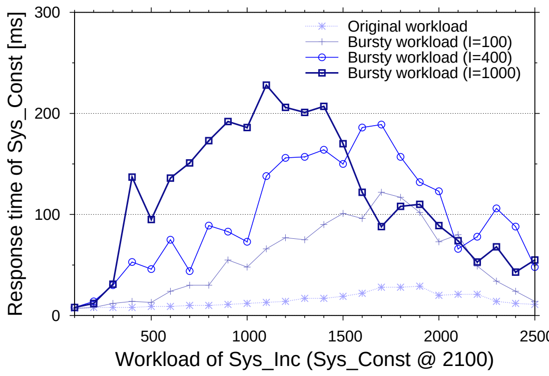


Figure 7: Response time of `Sys_Const` (workload=2100) with various burstiness levels of workload on the consolidated `Sys_Inc` under fully shared configuration (*Limit*=100%). The performance interference becomes larger as the burstiness of workload on the consolidated system becomes larger. `Sys_Const` [Liberal-SR], `Sys_Inc` [Liberal-SR], *Limit*=100%.

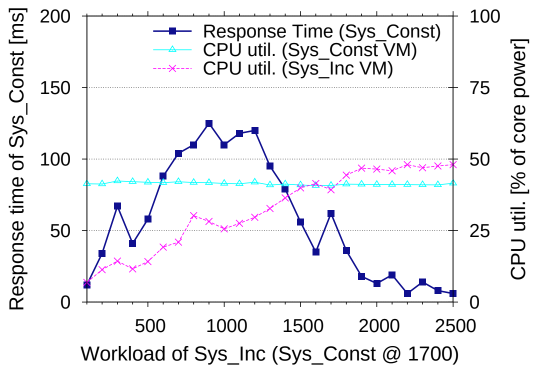


Figure 8: Response time of `Sys_Const` (workload=1700) consolidated with `Sys_Inc` (bursty workload: *I*=1000) in the disjoint CPU allocation. The performance interference from bursty workload on the consolidated system is unavoidable even under unsaturated CPU utilization in the disjoint CPU allocation scenario. `Sys_Const` [Liberal-SR], `Sys_Inc` [Liberal-SR], *Limit*=50%.

parameter to characterize the intensity of the traffic surges: *index of dispersion*, which is abbreviated as *I*. The larger the *I* is, the longer the duration of the traffic surge.[6] In this subsection, we use the bursty workload generator (with *I*=100, 400, and 1000) to investigate the non-monotonic response time variations under the condition that the workload of a consolidated system is bursty.

Figure 7 shows the response time of `Sys_Const` with the original RUBBoS workload (constant at WL=2100) consolidated with `Sys_Inc` which has bursty workload with various *I* values (also *I*=1, that is, the case of original RUBBoS workload). Here, the CPU sharing configuration is *Limit*=100% on each VM since it has the minimum response time increment in previous experiments as shown in Figure 4. The response time of `Sys_Inc` is omitted due to space constraints.

Figure 7 includes two significant facts to understand the impact of bursty workload on the performance interference. First, as the burstiness of the workload on a consolidated `Sys_Inc` increases, the larger the response time degradation on `Sys_Const` becomes. It means that the non-monotonic response time variations depend not only on the amount of workload and the CPU sharing configuration but also on the burstiness of the workload on the consolidated systems. Second, as the workload burinstess increases on a consolidated `Sys_Inc`, the peak of non-monotonic response time variations of `Sys_Const` shifts to a lower workload. When the burstiness increases to a certain level, the variations of workload are recognized as the periods with high workload and the other periods with low workload by the CPU scheduler of the hypervisor. Thus, in the periods recognized as high workload, the VM is assigned low priority on CPU allocation by the scheduler.

[6]The burstiness level of the original RUBBoS workload generator is *I*=1.

Finally, one more result with the *Limit*=50% configuration (i.e, fifty-fifty CPU splitting) is shown in Figure 8, so as to show the performance interference from bursty workload on a consolidated system is unavoidable even under unsaturated CPU utilization in the disjoint CPU allocation scenario. Here, the burstiness of workload on `Sys_Inc` is solely *I*=1000 and the amount of workload on `Sys_Const` is 1700. In this case, surprisingly, `Sys_Const` is degraded to twentyfold response time by the bursty workload on the consolidated system even though the VM in `Sys_Const` still remains the assigned CPU power (the CPU utilization of the VM is about 40% of core power, i.e., 80% relative utilization) and also the total CPU utilization of the core is far from saturation (70% overall).

V. Related Works

VM consolidation in cloud environments has become a very active research in recent years due to its practical interest.

Analytical approaches that assume linear consolidation performance have been proposed for performance prediction and management under VM collocation scenario (e.g., [6], [7], [13], [18]). For example, Ferreto et al. [6] apply linear programming to improve consolidated application performance through dynamic VM migration. Similarly, most papers that model and solve consolidation as a bin-packing optimization problem assume linear consolidation. Clearly, our experimental study of consolidation performance does not invalidate these good results, but our work helps to delimit the applicability of such results that assume linear consolidation performance.

Experimental approach that measures the performance interference between consolidated applications in virtualized environment has been studied before. For example, Padam et al. [3] have characterized and analyzed server consolidation benchmarks. Their experiments have shown that the performance of any workload suffers considerable loss when it is run in a consolidated environment. Aritra et al. [17] investigated the problem of data center consolidation with the goal of minimizing the total costs of running a data center; they built a tool called CloudBridge by combining hardware and software consolidation. Curino et at. [5] use a non-linear optimization algorithm to find the consolidation strategy, and then evaluated their optimization algorithm through measurements of benchmarks and real database usage traces. Their measurements found a very close match between their optimized consolidation solution and the measured results, even for relatively high CPU utilization levels (30% average and the 95% percentile at near 60%). To the best of our knowledge, their work is the first experimental paper that claims low performance interference in their measurements of production clouds at realistic CPU levels, which represents a significant evolution from other measurement-based papers on performance interference. Consequently, this paper corroborates our research that aims at providing an extensive evaluation of consolidation performance in diverse scenarios.

VI. Conclusion

We have investigated the sensitivity of consolidated n-tier application performance, particularly response time, with respect to two factors: (1) consolidated server specification of virtual machine resource availability, and (2) burstiness of n-tier application workload. With respect to scheduling isolation, our measurements show that the usual assumption of good hypervisor scheduling isolation (e.g., 50-50 even split of CPU between 2 VMs) may not yield the best overall performance. In fact, a sharing strategy (e.g., two co-located VMs each allowed 100% of CPU) enabled higher overall CPU utilization and better response time as well as throughput. The explanation is that isolation resulted in unused CPU due to reservation, and sharing enabled better utilization of CPU when one of the VMs became idle. As a potential solution to the performance interference among VMs, we studied cross-tier-priority (CTP) based scheduling [22] and found CTP effective in reducing performance loss by a factor of two.

The second part of our research studied the impact of workload burstiness on consolidated application performance. Our measurements show that sufficiently bursty workloads can degrade the consolidated application response time at relatively low overall CPU utilization of 70%. Furthermore, the negative impact of bursty workloads is insensitive to the CPU allocation strategy of VMs, affecting both the isolation and sharing strategies. Our experimental study shows that multiple factors (CPU allocation and workload burstiness) have non-trivial negative impact on the performance of consolidated applications. The experimental evidence corroborates the current belief that it is unlikely a simple solution will address consolidated application performance interference problem completely. Given the importance of consolidation for improved cloud utilization, it is clear that further research in this area is needed to improve our understanding of the performance interference phenomenon and find solutions that can effectively reduce its negative impact.

References

[1] RUBBoS: Rice University Bulletin Board Benchmark. http://jmob.objectweb.org/rubbos.html.

[2] S. Adler, "The SlashDot effect: An analysis of three internet publications," *Linux Gazette Issue*, Vol. 38, 1999.

[3] P. Apparao, R. Iyer, X. Zhang, D. Newell, and T. Adelmeyer, "Characterization & analysis of a server consolidation benchmark.," in *VEE 2008*.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, *et al.*, "A view of cloud computing," *Communications of the ACM*, 53.4 (2010): 50–58.

[5] C. Curino, E. P. Jones, S. Madden, and H. Balakrishnan, "Workload-aware database monitoring and consolidation," in *ACM SIGMOD 2011*.

[6] T. C. Ferreto, M. A. Netto, R. N. Calheiros, and C. A. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, 27.8 (2011): 1027-1034.

[7] Z. Gong, X. Gu, "Pac: Pattern-driven application consolidation for efficient cloud computing," in *MASCOTS '10*.

[8] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforcing performance isolation across virtual machines in Xen," in *Middleware 2006*.

[9] M. Harchol-Balter, B. Schroeder, N. Bansal, and M. Agrawal, "Size-based scheduling to improve web performance," *ACM Trans. on Computer Systems*, 21.2 (2003): 207-233.

[10] D. Jayasinghe, C. Pu, T. Eilam, M. Steinder, I. Whally, and E. Snible, "Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement," in *IEEE SCC 2011*.

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

[12] 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*.

[13] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via VM multiplexing," in *ICAC 2010*.

[14] N. Mi, G. Casale, L. Cherkasova, and E. Smirni, "Injecting realistic burstiness to a traditional client-server benchmark," in *ICAC 2009*.

[15] X. Pu, L. Liu, Y. Mei, S. Sivathanu, Y. Koh, and C. Pu, "Understanding performance interference of I/O workload in virtualized cloud environments," in *IEEE CLOUD 2010*.

[16] B. Schroeder, A. Wierman, M. Harchol-Balter, "Open versus closed: a cautionary tale," in *NSDI '06*.

[17] A. Sen, A. Garg, A. Verma, and T. Nayak, "CloudBridge: on integrated hardware-software consolidation," *ACM SIGMETRICS Performance Evaluation Review*, 39.2 (2011): 14–25.

[18] U. Sharma, P. Shenoy, D. F. Towsley, "Provisioning multi-tier cloud applications using statistical bounds on sojourn time," in *ICAC 2012*.

[19] B. Snyder, "Server virtualization has stalled, despite the hype," in *InfoWorld*, 2010.

[20] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise datacenters," in *WWW 2007*.

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

[22] Q. Wang, Y. Kanemasa, M. Kawaba, C. Pu, "When average is not average: Large response time fluctuations in n-tier systems," in *ICAC 2012*.