Understanding Performance Interference of I/O Workload in Virtualized Cloud Environments

Xing Pu[21], Ling Liu[1], Yiduo Mei[31], Sankaran Sivathanu[1], Younggyun Koh[1], Calton Pu[1]

[1]School of Computer Science, College of Computing, Georgia Institute of Technology, Atlanta, USA

[2]School of Computer Science and Technology, Beijing Institute of Technology, P.R. China

[3]Dept. of Computer Science and Technology, Xi'an Jiaotong University, P.R. China

***Abstract*—**Server virtualization offers the ability to slice large, underutilized physical servers into smaller, parallel virtual machines (VMs), enabling diverse applications to run in isolated environments on a shared hardware platform. Effective management of virtualized cloud environments introduces new and unique challenges, such as efficient CPU scheduling for virtual machines, effective allocation of virtual machines to handle both CPU intensive and I/O intensive workloads. Although a fair number of research projects have dedicated to measuring, scheduling, and resource management of virtual machines, there still lacks of in-depth understanding of the performance factors that can impact the efficiency and effectiveness of resource multiplexing and resource scheduling among virtual machines. In this paper, we present our experimental study on the performance interference in parallel processing of CPU and network intensive workloads in the Xen Virtual Machine Monitors (VMMs). We conduct extensive experiments to measure the performance interference among VMs running network I/O workloads that are either CPU bound or network bound. Based on our experiments and observations, we conclude with four key findings that are critical to effective management of virtualized cloud environments for both cloud service providers and cloud consumers. First, running network-intensive workloads in isolated environments on a shared hardware platform can lead to high overheads due to extensive context switches and events in driver domain and VMM. Second, co-locating CPU-intensive workloads in isolated environments on a shared hardware platform can incur high CPU contention due to the demand for fast memory pages exchanges in I/O channel. Third, running CPU-intensive workloads and network-intensive workloads in conjunction incurs the least resource contention, delivering higher aggregate performance. Last but not the least, identifying factors that impact the total demand of the exchanged memory pages is critical to the in-depth understanding of the interference overheads in I/O channel in the driver domain and VMM.

I. Introduction

Virtualization technology [14, 13] offers many advantages in current cloud computing environments by providing physical resources sharing, fault isolation and live migration. Virtualization allows diverse applications to run in the isolated environments through creating multiple virtual machines (VMs) on a shared hardware platform, and managing resource sharing across VMs by virtual machine monitor (VMM) technology [1]. Although VMMs (hypervisors) have the abilities to slice resources and allocate the shares to different VMs, our measurement study shows that applications running on one VM may still affect the performance of applications running on its neighbor VMs. In fact, the level of interferences mainly depends on the degree of the competition that the concurrent applications running in separate VMs may have in terms of shared resources. We argue that the in-depth understanding of the possible performance interferences among different VMs running on a shared hardware platform is critical for effective management of virtualized cloud, and an open challenge in current virtualization research and development.

In this paper, we study performance interference among different VMs running on the same hardware platform with the focus on network I/O processing. The main motivation for targeting our measurement study on performance interference of processing concurrent network I/O workloads in a virtualized environment is simply because network I/O applications are becoming dominating workloads in current cloud computing systems. By carefully design of our measurement study and the set of performance metrics we use to characterize the network I/O workloads, we derive some important factors of I/O performance conflicts based on application throughput interference and net I/O interference. Our performance measurement and workload analysis also provide some insights on performance optimizations for CPU scheduler and I/O channel and efficiency management of workload and VM configurations. .

The remainder of this paper is structured as follows: in Section II gives an overview of our experimental study, including the background on Xen I/O architecture, our experiment setup and I/O workloads. Section III analyzes the potential I/O interference factors in the cloud environment where different types of network I/O applications are running in isolated VMs on a shared hardware platform. Section IV reports our experimental results and illustrates request throughput and net I/O interference studies based on the network I/O workload characteristics. We conclude the paper with related works and a summary of contributions.

II. Background and Overview

In this section, we provide a brief background of Xen which is the virtualization platform we use in our measurement study. Then we describe the experimental setup, including measurement method and I/O workloads.

A. *Xen I/O Overview*

Xen [1, 2] is a popular open-source x86 virtual machine monitor (VMM) based on virtualization technologies. Recent prevalent virtualization technologies, like full system virtualization adopted in VmWare [15] and para-virtualization

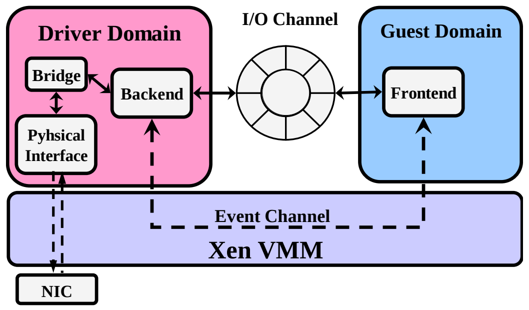

Figure 2. Xen I/O architecture

are both supported by Xen which uses para-virtualization as a more efficient and lower overhead modes of virtualizations. In para-virtualization I/O mode, Xen VMM layer uses asynchronous *hypercall* mechanism to deliver virtual interrupts and other notifications among domains via event channel. A privileged domain called *Domain0* is treated as driver domain hosting unmodified Linux drivers and has the access to hardware devices. Driver domain performs I/O operations on behalf of unprivileged guest domains which are ported to the virtual driver interface from Linux operation system (XenoLinux). Fig. 1 shows the logical components of the latest Xen I/O architecture [5]. The virtual network interface in guest domain is called netend (frontend) acting as the real hardware drivers. In the privileged driver domain, netback (backend) is a counterpart for each netend. Netfront and corresponding netback use "page-flipping" technique to exchange data in the I/O channel by sharing memory pages pointed in the descriptor ring (Fig. 1), no data copy involved. The bridge in driver domain handles the packets from NIC and performs the software-based routine to the destination domain. We notice that netfront, netend and bridge are extra code in the VMs and "page-flipping" is also an extra part in I/O channel.

When a network packet is received by the NIC (RX), the NIC will raise an interrupt to the upper layer. Before the interrupt reaches the driver domain, hypervisor (VMM) handles the interrupt first. Hypervisor will determine whether or not the driver domain has the access to the real hardware. Upon receiving the interrupt, the privileged driver domain starts to process the network packet. It first removes the packet from NIC and sends the packet to the software Ethernet bridge. Then Ethernet bridge de-multiplexes the packet and delivers it to the appropriate netback interface. Netback raises a hypercall to hypervisor, requesting an unused memory page and hypervisor notifies the corresponding guest domain to release a page to keep the overall memory allocation balanced. Netback and netfront exchange the page descriptors by page-remapping mechanism over I/O descriptor ring (Later the data copy is performed). Finally, guest domain receives the packet as if it comes directly from NIC. A similar but reverse procedure is applied to send a packet using the send path (TX), except that no explicit memory page exchange is involved, only the ownership of physical page is transferred instead of the real page. NIC supports direct memory access (DMA) technique handles the target guest memory page directly. We will see that three address remappings and two memory allocation/deallocation operations are used for per packet receive and only two remappings are required for each packet transmit [10, 7, 11].

B. *Testbed Architechture*

We carefully designed our experiments to exercise network I/O traffics and evaluate the performance interference in virtualized cloud environments. All experiments were performed on an IBM ThinkCentre A52 Workstation with two 3.2GHz Intel Pentium 4 CPUs (both have 16KB L1 caches and 2MB L2 caches), 2 GB 400 MHz DDR memory, a Seagate 250 GB 7200 RPM SATA2 disk, and Intel e100 PRO/100 network interface. Client machines were connected by a 1 Gbit/s Ethernet network. The latest version 3.4.0 Xen hypervisor with the most stable Linux Xen Kernel 2.6.18.8-xen [16] are used.

Fig. 2 gives a sketch of the experimental setup used in most of the experiments reported in this paper. In each experiment, two I/O intensive workloads are running in two isolated guest domains respectively (VM1 and VM2) sharing the same physical host via VMM. Each guest domain is allocated with equal resources, where the memory allocations are both 512 MB, the default CPU scheduler (SMP *Credit scheduler*) [4] is configured with equal *weight* value and for each VM the parameter *cap* is 0. The Apache HTTP servers [18] residing in VM1 and VM2 provide the web services, and cache the data in the buffers, no disk readings are involved. Client machines using httperf [9, 17] tool as our HTTP "load generator" are designed to access virtualized servers remotely. They send requests to corresponding virtual server, retrieving a fixed size file: 1 KB, 4 KB, 10 KB, 30 KB, 50 KB, 70 KB or 100 KB. These fixed size files are carefully selected I/O workloads from SPECweb'99 [19], SPECweb'2005 [20] and SPECweb'2009 [21] benchmarks, which are the industry standard to evaluate the web server performance. Each workload is a representative log file size in current data center.

C. *I/O Workloads*

Before measuring the interferences of multiple servers running on one single physical hardware host, we first evaluate the performance of single guest domain in order to get the actual performance results and characteristics of each workload running in our experimental environment, which serve as the *basecase* in the rest of the experiments.

Table I shows the maximum performance results of one guest domain running under the selected SPECweb network I/O workloads. When server becomes saturated at full capacity, the 1 KB and 4 KB file workloads reach 0.5 to 15 times higher request throughput than others respectively, and consume more

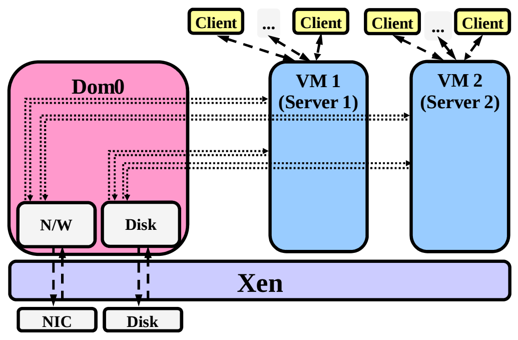

Figure 1. Logical components of virtulized cloud environments. (Web servers reside in VM1 and VM2, *Dom0* is only the driver domain)

TABLE I. MAXIMUM PERFORMANCE OF WORKLOADS IN BASECASE

Workload	Major Resource Used	Throughput (Req/sec)	Net I/O (KB/sec)	Response Time (ms)	CPU (%)
1 KB	CPU	1900	2018	1.52	97.50
4 KB	CPU	1750	7021	5.46	97.46
10 KB	Network	1104	11048	2.36	70.44
30 KB	Network	375	11271	2.52	54.87
50 KB	Network	225	11262	2.7	49.62
70 KB	Network	160	11255	2.84	47.10
100 KB	Network	112	11208	2.08	44.40

than 97.5% CPU resource (approximately, the remaining 2.5% CPU is charged by the system monitor tool), while the network bandwidth utilization is only around 20% and 60% respectively. The web mix performance of these two workloads is limited by CPU resource. Although the achieved request rates are not higher than 1200 req/sec, 10-100 KB workloads saturate the server by consuming all the network bandwidth, which is 10 KB × 1104 req/sec ≈ 30 KB × 375 req/sec ≈ 50 KB × 225 req/sec ≈ 70 KB × 160 req/sec ≈ 100 KB × 112 req/sec ≈ 100 MB/sec. Meanwhile, when VM1 is serving one of these five workloads respectively in each of the five experiments (10-100 KB), the total CPU utilization of driver domain and guest domain is less than 75%. These reveal that 1 KB and 4 KB workloads are CPU bounded and 10-100 KB workloads are network bounded, consistent with the observation made from prior research [3], namely short file workload is CPU bounded and long file workload is network bounded.

III. INTERFERENCE ANALYSIS

In this section we outline the methodology and metrics used for our measurement study.

Let Dom_0, Dom_1 … Dom_n be the virtual machines running on the same host, where Dom_0 is the driver domain. Suppose that Dom_i is serving workloads i, we define the maximum throughput of Dom_i as T_i. We use B_i to denote the maximum throughput of Dom_i in *basecase* scenario where n equals 1, i.e., only driver domain and one guest domain are hosted on the physical machine. Then, the combined normalized throughput scores is defined as follows:

$$Combined\ Score = \sum_{i=1}^{n} \frac{T_i}{B_i}$$

Fig. 3 presents the set of experiments conducted on the setup of two gust domains with each running one of the selected workloads. We measure the normalized throughput scores of different combination of selected network I/O workloads. The base-case throughput used in this experiment is the throughput of workloads in the single gust domain *basecase* (Table I). To present the results clearly, we denote each combination of the two workloads running on VM1 and VM2 in a tuple of two elements, with the first element x denotes xKB file retrieving from VM1, and the second element y is the yKB file retrieving from VM2. For example, the expression (1, 30) says that VM1 is severing 1 KB file workload and VM2 is severing 30 KB file workload. The expression (1, Idle) refers to

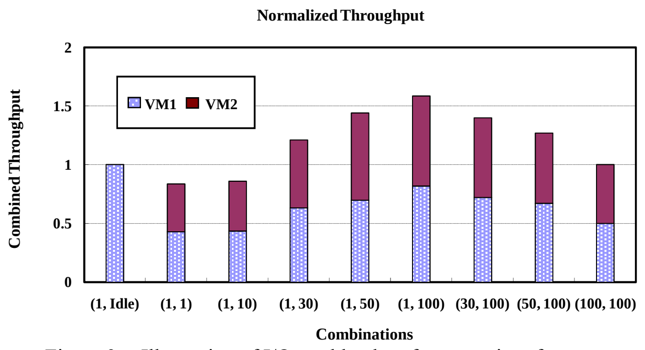

Figure 3. Illustration of I/O workload performance interfernce. (Web servers running in two isolated guest domains)

the case where VM1 is serving 1KB file workload and VM2 is idle. From Fig. 3, we observe some interesting facts regarding performance interferences. First, the combination of workloads (1, 100) achieves the best performance with its combined throughput score of 1.58. Given that 1 KB workload is CPU bounded and 100 KB workload is network bounded, this combination clearly incurs the least resource contention compared to other combinations. Similarly, the workload combinations of (1, 50) is the next best pairing for the similar reason. The workload combinations of (30, 100) and (50, 100) offer better combined throughput than the worst combination of workloads (1, 1) and (1, 10), which incurs highest resource contention. Our observations are consistent with the Xen I/O architecture presented in previous section II. Current Xen I/O mode uses the privileged driver domain to provide better security isolation and fault isolation among VMs. Although this mode could prevent buggy drivers and malicious attacks successfully, the driver domain may easily becomes the bottleneck: when a VM wishes to get accesses to underlying hardware or communicate with others, all the events have to go through driver domain and hypervisor layer. This is supposed to cause control or communication interferences among VMs. The multiplexing/demultiplexing of bridge and I/O channel may incur memory page management interferences, such as packets lost for high latency, fragmentations and increased data coping overhead. In addition, we also believe that the default *Credit scheduler* may have some impacts on the overall performance of running multiple VMs on the same hardware platform. This set of experiments also indicates that although performance interference of network I/O applications in virtualized cloud environments is unavoidable given the Xen I/O architecture and the inherent resource sharing principle across VMs. In-depth understanding of the number of key factors that cause certain types of resource contentions and thus performance interferences is critical for both cloud service providers and cloud consumers.

In addition to the combined throughput ratio scores, we also measure the virtualization specific system-level characteristics using the following eight performance metrics to better understand the performance interference of running multiple network I/O workloads in isolated VM environments on a single hardware platform. These system-level metrics can be best utilized to analyze the resource contentions of network I/O workloads and reveal the intrinsic factors that may have been induced performance interference observed.

TABLE II. SYSTEM-LEVEL CHARACTERISTICS OF I/O WORKLOADS WITH WEB SERVER RUNNING IN VM1 (VM2 IS NOT CREATED).

Workload	CPU (%)	Event (events/sec)	Switch (switches/sec)	Pages Exchange (pages/sec)	I/O Execution (pages/exe)	Driver Domain (*Dom0*)				Guest Domain (VM1)			
						CPU (%)	*Waiting (%)*	*Block (%)*	*Execution (exe/sec)*	*CPU (%)*	*Waiting (%)*	*Block (%)*	*Execution (exe/sec)*
1 KB	97.50	224104	9098	11500	4.93	46.83	4.81	10.54	2433	50.67	38.97	4.76	2535
4 KB	97.46	242216	10525	14900	6.38	46.82	4.77	10.68	2335	50.65	37.72	5.65	2509
10 KB	70.44	279663	15569	16000	7.28	39.7	3.13	13.73	2198	30.74	2.29	23.48	2092
30 KB	54.87	345496	26118	13000	3.87	36.78	1.92	17.78	3023	18.09	1.31	34.32	3777
50 KB	49.62	342584	26981	11000	3.54	34.57	1.22	19.33	3108	15.05	1.14	36.76	3915
70 KB	47.10	341898	27436	10500	3.36	33.41	0.90	19.46	3123	13.70	1.07	37.62	4034
100 KB	44.40	332951	27720	10000	3.17	32.19	0.77	21.04	3150	12.21	1.01	39.08	3962

Using Xen hypervisor system monitor tools, we collect eight system-level characteristics from I/O workloads. These system-level characteristics could reveal the underlying details of I/O performance interference and make VMs behaviors understandable. In our design of evaluation, the following eight different workload characteristics are collected:

- **CPU utilization** (CPU): We measured the average CPU utilization of each VM, including CPU usage of *Dom0*, VM1 and VM2 respectively. The total CPU consumption is the summation of them all.

- **VMM events per second** (Event): VMM adopts asynchronous *hypercall* mechanism to notify the VMs of system events. This metric measures the number of events per second in the event channel.

- **VM switches per second** (Switch): The number of VMM switches per second is collected by this metric. It is an important performance indicator for our study.

- **I/O count per second** (Pages Exchange): This metric captures the number of exchanged memory pages during the period of one execution. This metric reflects the efficiency of I/O processing.

- **I/O count per execution** (I/O Execution): During the period of one execution, the number of exchanged memory pages is captured here. This metric could help to reflect the efficiency of I/O processing.

- **VM state** (Waiting, Block): at any point of time, every domain sharing the same host must be in one of following three states: *execution state*, *runnable state* and *blocked state*. When one VM is currently using CPU, it is in the *execution state*. The metric "Wait" is the percentage of waiting time when a VM is in *runnable state*, or in other words, the VM is in the CPU run queue but do not utilize the physical CPU (*PCPU*). The metric "Block" is the percentage of blocked time when a VM is in *blocked state*, which represents the fact that a VM is blocked on I/O events and not in the run queue.

- **Executions per second** (Execution): This metric refers to the number of execution periods per second for each VM. We divide "Pages Exchange" by the measured value of this metric to calculate the approximate "I/O Execution".

IV. PERFORMANCE INTERFERENCE STUDY

In this section we first provide our measurements of the eight metrics with varying workload rates, from 10%, 20%, 30%, to 100% under the experimental setup of single guest domain running the selected network I/O workloads. This set of measurements is used as the *basecase* scenario to analyze the different workload combinations and understanding the resource contentions displayed by the various combinations of workloads. Then we analyze the throughput performance interferences and net I/O performance interferences in the remaining of this section for network I/O workloads of varying sizes.

A. *Basecase Measurement*

In this set of experiments, we set up our testbed with one VM (VM1) running one of the six selected network I/O workloads of 1 KB, 4 KB, 30 KB, 50 KB, 70 KB and 100 KB for all eight metrics outlined in Section III. Table II shows the results. The value of each I/O workload characteristics is measured at 100% workload rate for the given workload type. Comparing with network-intensive workloads of 30 KB, 50 KB, 70 KB, and 100 KB files, the CPU-intensive workloads of 1 KB and 4 KB files have at least 30% and 60% lower event and switch costs respectively because the network I/O processing is more efficient in these cases. Concretely, for 1 KB and 4 KB workloads, we can see from Table II, driver domain has to wait about 2.5 times longer on the CPU run queue for being scheduled into the *execution state* and the guest domain (VM1) has 30 times longer waiting time. However, they are infrequently blocked for acquiring more CPU resource, especially in the guest domain the block time is less than 6%. We notice that the borderline network-intensive 10 KB workload has the most efficient I/O processing ability with 7.28 pages per execution while the event and switch numbers are 15% larger than CPU-intensive workloads. Interesting to note is that initially, the I/O execution is getting more and more efficient as file size increases. However, with file size of workload grows larger and larger, more and more packets need to be delivered for each request. The event and switch number are increasing gradually as observed in Table II. Note that the VMM events per second are also related to request rate of the workload. Though it drops slightly for the workload of file size 30-100 KB, the overall event number of network-intensive workloads is still higher than CPU-intensive ones. With increasing file size of shorter workloads (1 KB, 4 KB and 10 KB), VMs are blocked more and more frequently. Finally, I/O execution starts to decline when the file size is greater than 10 KB. The network I/O workloads that exhibit CPU bound are now transformed to network bounded as the file size of the workloads exceeding 10 KB and the contention for network resource is growing higher as the file size of the workload increases. In our experiments, the 100 KB workload shows the highest demand for the network resource. These basic system-level characteristics in our *basecase* scenario can help us to compare and understand better the combination of different workloads and the multiple factors that may cause different levels of performance interferences with respect to both throughput and net I/O.

B. *Throughput Interference*

In this subsection, we focus on studying performance interference of running multiple network I/O workloads in isolated VM environments on a shared hardware platform, our testbed setup. We focus on understanding the impact of running different combinations of workloads of different file sizes on the aggregate throughput.

In this group of experiments, we use the representative CPU bounded workload of 1 KB file and the representative network bounded workload of 100 KB file as the main workload combination for our analysis. The reason we choose this combination is primarily because this combination exhibits the best combined throughput score in Table I. This motivates us to understand the key factors that impact on their high combined throughput score.

Fig. 4 shows the set of experiments we have conducted for throughput interference analysis on three combinations of workloads running concurrently on the setup of two VMs: (1, 1), (1, 100) and (100, 100). We use the tuple (1,100) to denote the workload of 1 KB running on VM1 and the workload of 100 KB running on VM2. Also we use the notation of (1, 100)_1KB to denote the measurement of workload 1 KB on VM1 in the graphs. We vary the workload rates in x-axis and measure the different performance metrics and plot them on y-axis. To get a clear overview of curves on the same scale, the rates of different workloads in Fig. 4 were converted into the percentage of maximum achieved throughput in *basecase*. For example, for 100% workload rate of 1 KB workload in Fig. 4, the maximum request rate of 1900 req/sec in Table I is used. Similarly, for 100 KB workload, the 100% workload rate in Fig. 4 means the maximum request rate of 112 req/sec is used

Recall Fig. 3, we have shown that the combinations of workloads that compete for the same resource, either CPU or network, stand a good chance of performance degradation compared to the *basecase* in Table I. However, two workloads that exhibit distinct resource need, such as one CPU bound and the other network bound, running together gets better performance compared to the one VM *basecase*. The following analysis will provide us some in-depth understanding of the key factors that lead to different performance interferences among different combinations of applications running concurrently in multiple VMs

Fig. 4 (a) illustrates the throughput performance of the three workloads combinations: (1, 1), (1, 100), (100, 100). We observe that 1 KB workload in (1, 100) combination reaches around 1560 req/sec, which is twice the achieved maximum 1 KB workload throughput of 790 req/sec in (1, 1) combination, even though it is still 18% lower than 1900 req/sec, the maximum throughput in *basecase*. For (1, 100) combination, 100 KB workload on VM2 achieves the throughput of 85 req/sec at 100% workload rate, which is 76% of maximum throughput of 112 req/sec in *basecase* and 52% higher than the throughput of 56 req/sec for 100 KB workload in the combination of (100, 100). Note that the combination of (100,100) causes high network resource contention and saturates the host at 50% workload rate (see the enhanced chart embedded in Fig. 4).

In order to provide the in-depth understanding of the detailed factors that cause these different levels of throughput interferences, we conduct the rest of the experiments and the results are shown in the remaining figures and plots in Fig. 4. First, let us examine at the combination of two network-intensive workloads of (100, 100).

Fig. 4 (b) shows the numbers of switches per second when varying the workload rates from 10% to 100%. Note that the number of events follows exactly same trend as switches, though the concrete values are different in scale. At 50% workload rate, the combination (100, 100) reaches the maximum switch number, and it starts to drop when the workload rate increases to 60% and it remains almost flat even when the workload rate increases from 60% to 100%. Comparing with other two combinations of workloads, (100, 100) has at least twice higher switch cost than (1,100) and (1, 1) under the peak switch costs. This implies that a main source of overhead and throughput interference for the combination (100, 100) may come from the high event and switch costs in VMM and *Dom0*.

Fig. 4 (c) shows the exchanged memory pages during one CPU execution time. We observe that as the workload rate is getting higher, the combination of (100, 100) has less than four pages per CPU execution duration and it experiences the worst efficiency in I/O pages exchange. Practically, heavy event and switch costs lead to a lot more interrupts that need to be processed, resulting in few I/O pages exchange during each execution cycle.

Fig. 4 (d) measures the *Dom0* CPU utilization with varying workload rates for three combinations of workloads. We observe that *Dom0* is busy for being scheduled for CPU processing on each interrupt. In Fig. 4 (d), (100, 100) combination uses relatively higher but more stable CPU usage around 35% for the workload rate at 50% or higher.

Fig. 4 (e) and Fig.4 (h) present the block time of *Dom0* and block time of two guest domains respectively. For (100,100) combination, the block time of *Dom0* is around 30% for workload rate of 50% or higher, and the block time of guest domains are around 48%, and both are relatively high compared to other two workload combinations. This indicates

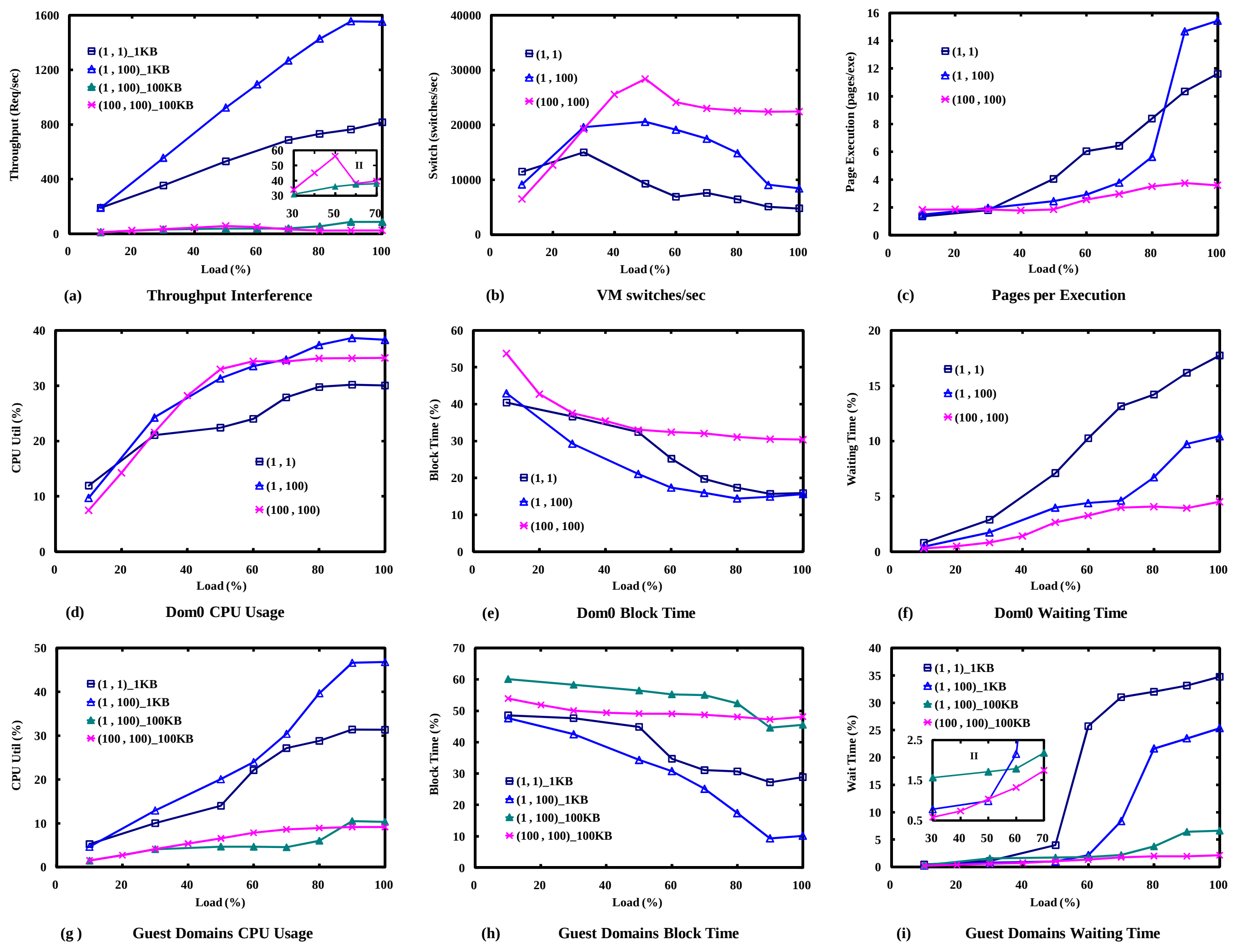

Figure 4. Throughput interference of 1 KB workload running with100 KB workload.

that VMs are frequently blocked for I/O events and waiting for next scheduling.

Fig. 4 (f) and Fig.4 (i) measure the waiting time of *Dom0* and of guest domains. We observe that the combination of (100,100) has the lowest waiting time in both *Dom0* and guest domains. This is mainly due to the high blocking time, as it reveals that the CPU run queues are not crowd and could serve the VMs much faster. In summary, we conclude that due to heavy event and switch costs in (100, 100), VMM and *Dom0* are quite busy to do notifications in event channel, resulting in the fact that driver domain needs more CPU resource and guest domains are free and are waiting for I/O events (see Fig. 4 (g)).

For the combination of workloads (1, 1), we observe from Fig. 4 (b) that it has the lowest event and switch number. However, this workload combination has high contention on CPU resource. Thus, the poor throughput performance shown in Fig.4 (a) should come from other factors. From Fig. 4 (c), we see that the combination (1, 1) processes the I/O communication more efficiently than the combination of (100, 100), it has three times more pages exchange during each execution when the workload rate is high. Before the workload rate reaches 80%, the workload combination of (1,1) has higher pages exchange per execution compared to that of combination (1, 100). Thus, we can infer that the throughput interference of combination (1, 1) may be caused by the fast I/O processing between guest domains and driver domain. This conclusion can be further validated using the experimental results shown in Fig. 4 (f) and Fig. 4 (i). We see that the CPU waiting time of *Dom0* and guest domains are both the longest in comparison, approximately achieving 17% and 34% respectively, i.e., the CPU run queues are crowed waiting for more VCPUs to be put into *execution state*. All the VMs in the combination of (1, 1), including *Dom0*, are acquiring CPU resource. The *Credit scheduler* used in our experiments is configured with equal *weight* for all VMs, i.e., all VMs should be dispatched with the same CPU share, thus we could see that the CPU utilization of *Dom0* and of guest domains have similar trend and share (100% / 3 ≈ 33%) when they all demand for CPU resource. To achieve high throughput, all the VMs in (1, 1) combination need to perform fast I/O page flipping, which results in higher interference in CPU scheduling, and the lower throughput performance shown in Fig. 4 (a) compared to (1,100).

By analyzing the performance interferences in the combinations of (100, 100) and (1, 1), we understand that the frequently I/O memory pages exchange of (1, 1) leads to CPU contention among VMs, and the combination of (100, 100) incurs higher event and switch overheads in VMM and *Dom0*, leads to high level of network contention. In comparison, the combination of (1, 100) founds the balance to achieve higher throughput with increasing workload rates. Concretely, comparing with the combination (1, 1), *Dom0* and VM1 in (1,100) experience *blocked state* infrequently and shorter waiting time to be allocated more CPU resource, finally 50% lower in VM1 (see Fig. 4 (e) and Fig. 4 (h)). Comparing with (100, 100), in (1,100) combination, VM2 is blocked frequently and waiting longer to reduce the event and switch overhead (Fig. 4 (f) and Fig. 4 (i)). Finally, the I/O page exchanges per execution becomes more efficiently under high workload rate (Fig. 4 (c)) and *Dom0* is better utilized.

From this group of experiments, we draw four concluding remarks:

- Due to larger number of packets to be routed per HTTP request, the combination of network-bound workloads leads to at least twice higher event and switch costs in event channel, making driver domain busy for processing I/O interrupts and suffering significantly higher L2 cache misses, while leaving guest domains spending longer time waiting for corresponding I/O events.
- To achieve high throughput, the combination of CPU-bound workloads results in guest domains competing with driver domain for CPU resource to do fast I/O executions, while the switch cost is the lowest, leading to certain level of performance interference due to CPU contention, though the degree of interference is relatively less sever when compared to (100,100) combination.
- The workload combination with least resource competition is the combination of (1,100). This is mainly due to the fact that it alleviates the stress in driver domain by scheduling the extra CPU to *Dom0* and the guest domain (VM1 in our experiments) that is serving CPU-intensive workload and at the same time it increases the block time and waiting time of guest domain (VM2 in our experiments) that is serving network-intensive workload to reduce the switch overhead.
- Interference is highly sensitive to the efficiency of driver domain due to multiple VMs are competing for I/O processing and the switching efficiency in the driver domain.

C. *Net I/O Interference*

Based on our performance analysis in the previous section, we know that (1, 1) and (100, 100) causes throughput interference mainly due to the driver domain demanding for fast I/O processing and the VMM high switch overhead respectively. In comparison (1, 100) reaches better performance by alleviating the stresses in driver domain. In this section we study the net I/O interference in these three combinations of workloads.

Fig. 5 presents the net I/O measurement result for (1,1), (1,100) and (100,100) combinations of workloads. These

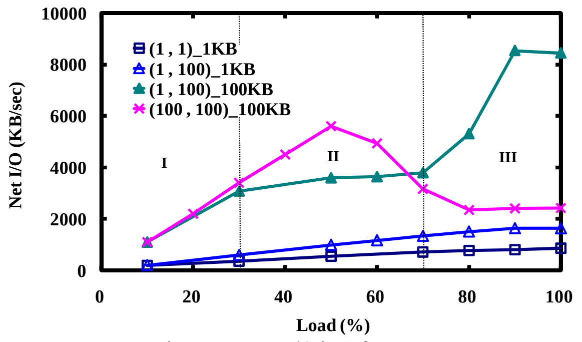

Figure 5. Net I/O interference.

curves are highly correlated with trends of request throughput in Fig. 4 (a), based on the fact that (workload size) × (request rate) ≈ Net I/O. Both 1 KB and 100 KB workloads in (1, 100) achieve higher maximum net I/O than others under high workload rates, i.e., 1 KB × 1560 req/sec ≈ 1634 KB/sec and 100 KB × 85 req/sec ≈ 8550 KB/sec respectively. It is interesting to note that when the workload rate is approximately less than 70%, (100, 100) gains better net I/O performance compared to other combinations (1,1) and (1,100), while between 30% and 70% workload rates (i.e., see the range *II* in Fig. 5) the net I/O of 100 KB workload in (1, 100) combination (i.e., VM2) remains flat, at around 3600 KB/sec, which is close to the current *window size* (i.e., 4 KB) in XenoLinux. It seems that 100 KB workload is "paused" during range *II*. We can understand this phenomenon better by examining the details of 100 KB curves presented in Fig. 4 (g)~ Fig. 4 (i). Within the load range *II* in Fig.5, the block time of both (1,100)_100KB and (1,100)_1KB in Fig. 4 (h) are relatively flat, while (1, 100)_100KB has on average 5% higher block time than (100, 100)_100KB. Meanwhile, the waiting time of (1, 100) in enhanced graph embedded in Fig. 4 (i) is keeping around 2%. Thought, the waiting time of (100, 100) increases three times from 0.5% to 1.7%, it is still the lowest one. Ultimately, (1, 100) stabilizes the CPU usage around 4%, while (100, 100) consumes CPU resource continuously until saturating the NIC.

Despite achieving better request throughput and net I/O in the combination of (1, 100), we notice that 1 KB workload in this combination gets "priority" treatment while leaving 100 KB workload blocked more often and waiting longer. To understand this phenomenon, it is worthwhile to discuss the details of CPU run queues in *Credit scheduler*. The default *Credit scheduler* is configured with equal *weight* for each VM and *cap* is 0, which means it is working in the *work-conserving* mode attempting to share the processor resources fairly. All the VCPUs in the CPU queues are served in the first-in, first-out manner. Additional, when a VM receives an interrupt while it is idle, the VM enters the particular *Boost* state which has a higher priority to be inserted into the head of run queue for the first CPU execution. This mechanism prevents the long waiting time for the latest active VM by preempting the current running VM. However, the even priority shares of processor usage remains a problem for network-intensive workloads here. Consider the combination of (1, 100), because of the file size of 1 KB is the shortest, it could finish each request and enters the idle state faster (most infrequently blocked in Fig. 4 (h)). Finally, hypervisor makes the decision to put the VM1 in the head of run queue frequently. This makes the 1 KB workload have higher priority. Thus, the effects of *Credit scheduler* should be considered in virtualized cloud environments as it is making positive contributions to CPU-intensive workloads while treating network-intensive workloads unfairly with higher processing latency, leading to poor Net I/O performance.

V. RELATED WORK

Virtualization is becoming widely used in cloud environments. Although a fair number of research projects have dedicated to measuring, scheduling, and resource management of virtual machines, there still lacks of in-depth understanding of the performance factors that can impact the efficiency and effectiveness of resource multiplexing and resource scheduling among virtual machines.

Several recent research efforts [12] proposed "direct I/O" to alleviate the pressure in driver domain, with guest domains get direct accesses to hardware. However, direct I/O is not considered in our works, as it lacks of dedicated driver domain to perform fault isolation and live migration. If we want to alleviate the I/O interferences under current Xen I/O model, we should understand in depth about potential interference factors residing in Xen VMM. Yong *et al.* [8] studied the effects of performance interference between two virtual machines hosted on the same hardware platform by looking at system-level workload characteristics. Through subsequent analysis of collected characteristics, they predicted the performance of new application from its workload characteristic values successfully within average error of approximately 5%. However, their studies are no net I/O workload involved, meanwhile only focus on finally results, details of the process are not presented.

VI. CONCLUSIONS

In this paper, we have present our experimental study on the performance interference in parallel processing of CPU and network intensive workloads in the Xen Virtual Machine Monitors (VMMs). We conduct extensive experiments to measure the performance interference among VMs running network I/O workloads that are either CPU bound or network bound. Based on our experiments and observations, we conclude with four key findings that are critical to effective management of virtualized cloud environments for both cloud service providers and cloud consumers. First, running network-intensive workloads in isolated environments on a shared hardware platform can lead to high overheads due to extensive context switches and events in driver domain and VMM. Second, co-locating CPU-intensive workloads in isolated environments on a shared hardware platform can incurs high CPU contention due to the demand for fast memory pages exchanges in I/O channel. Third, running CPU-intensive workloads and network-intensive workloads in conjunction incurs the least resource contention, delivering higher aggregate performance. Last but not the least, identifying factors that impact the total demand of the exchanged memory pages is critical to the in-depth understanding of the interference overheads in I/O channel in the driver domain and VMM.

VII. ACKNOWLEDGMENT

This work is partially supported by grants from NSF CISE NetSE program, CyberTrust program, and grants from IBM faculty award, IBM SUR, and Intel Research Council. The first author performed this research as a visiting PhD student at the Distributed Data Intensive Systems Lab (DiSL), School of Computer Science, Georgia Institute of Technology, supported by China Scholarship Council and School of Computer Science and Technology, Beijing Institute of Technology.

REFERENCES

[1] P. Barham, B. Dragovic, K. A. Fraser, S. Hand, T. Harris, A. Ho, E. Kotsovinos, A. Madhavapeddy, R. Neugebauer, I. Pratt and A. Warfield, "Xen 2002," Technical Report of University of Cambridge, NO. UCAM-CL-TR-553, ISSN 1476-2986, January 2003.

[2] P. Barham, B.Dragovic, K. Fraser, S. Hand, T, Harris, A. Ho, I. Pratt, A. Warfield, P. Barham, and R. Neugebauer, "Xen and the art of virtualization," In Proc. of ACM SOSP, October 2003.

[3] L. Cherkasova, R.Gardner, "Measuring CPU overhead for I/O processing in the Xen virtual machine monitor," In Proc. of 2005 USENIX Annual Technical Conference, Anaheim, CA, USA, 2005.

[4] L. Cherkasova, D. Gupta and A. Vahdat, "Comparison of the three CPU schedulers in Xen," ACM SIGMETRICS Performance Evaluation Review, (PER'2007), Vol. 35, No. 2, pp. 42-51, September 2007.

[5] K. Fraser, S. Hand, R. Neugebauer, I. Pratt, A. Warfield, M. Williamson. "Reconstructing I/O," Tech. Report, UCAM-CL-TR-596, August 2004.

[6] D. Gupta, R. Gardner, and L. Cherkasova, "XenMon: Qos monitoring and performance profiling tool," HP Laboratories Report, NO. HPL-2005-187, October 2005.

[7] D. Gupta, L. Cherkasova, R. Gardner, and A. Vahdat, "Enforing performance isolation across virtual machines in Xen", In Proc. of the ACM/IFIP/USENIX 7th International Middleware Conference (Middleware'2006), Melbourne, Australia, November 2006.

[8] Y. Koh, R. Knauerhase, P. Brett, M. Bowman, Z. Wen and C. Pu, "An analysis of performance interference effects in virtual environments," IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'2007), San Jose, California, USA, April 2007.

[9] D. Mosberger, T. Jin, "httperf—A tool for measuring web server performance," Proc. of Workshop on Internet Server Performance, 1998.

[10] A. Menon, J. Santos, Y. Turner, "Diagnosing performance overheads in the Xen virtual machine environment," In Proc. of the ACM/USENIX Conference on Virtual Execution Environments, June 2005.

[11] A. Menon, A. L.Cox, W. Zwaenepoel, "Optimizing network virtualization in Xen," In USENIX Annual Technical Conference, 2006.

[12] K. Mansley, G. Law, D. Riddoch, G. Barzini, N. Turton, and S. Pope. "Getting 10 Gb/s from Xen: Safe and fast device access from unprivileged domains," In Euro-Par 2007 Workshops: Parallel Processing, 2007

[13] P. Padala, X. Zhu, Z. Wang, S. Singhal and K. Shin, "Performance evaluation of virtualization technologies for server consolidation," HP Laboratories Report, NO. HPL-2007-59R1, September 2008.

[14] R. Rose, "Survey of system virtualization techniques", Technical report, March 2004.

[15] VmWare: http://www.vmware.com.

[16] The XenTM Virtual Machine Moniter: http://www.xen.org.

[17] Httperf: http://www.hpl.hp.com/research/linux/httperfs.

[18] The Apache Software Foundation: http://www.apache.org.

[19] SPECweb99: http://www.spec.org/web99/docs/whitepaper.html.

[20] SPECweb2005: http://www.spec.org/web2005/docs/1.20/run_rules.html.

[21] SPECweb2009: http://www.spec.org/web2009/docs/design/.