Performance Evaluation of Video-On-Demand in Virtualized Environments: The Client Perspective

Jagmohan Chauhan
Dept. of Computer Science
University of Saskatchewan
Saskatoon, SK, CANADA S7N 3C9
jac735@cs.usask.ca

Dwight Makaroff
Dept. of Computer Science
University of Saskatchewan
Saskatoon, SK, CANADA S7N 3C9
makaroff@cs.usask.ca

ABSTRACT

Virtualization is a key technology for data centres to implement infrastructure as a service as well as to achieve server consolidation and application colocation. Over the years performance of virtual machine (VM) monitors have improved [8]. Thus, new services are being migrated to these environments. Latency sensitive applications however, are not considered fit for Virtual environments due to high virtualization overhead and potential interference from other VMs. In this paper, we performed measurement-based analysis of the performance impact on VOD server in presence of I/O bound workloads in co-located virtual machines. The focus of our study is on the QOS (quality of service) received by clients; metrics of delay, jitter, packet loss are examined. As expected, the performance of VOD server in a VM suffers severe degradation in presence of heavy disk-I/O bound and outbound network UDP co-located workloads.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems

Keywords

I/O Virtualization, KVM, Virtio, Vhost-Net, video-on-demand

1. INTRODUCTION

Large media server deployments require the use of clusters of servers and complicated content distribution networks. There is also a place for small and medium-sized media servers to be deployed at a lower infrastructure cost for enterprises with more modest viewing population. If some portion of a computer's resources could be dedicated to video server applications, then cloud infrastructures could be deployed to achieve this functionality.

Virtualization allows for such sharing of resources through the creation and management of virtual machines. A given hardware platform thus contains host software (a hypervisor), which creates a simulated computer environment, and a virtual machine (VM), called the guest-OS. Virtualization provides server consolidation, fault isolation, security and migration within a machine and between machines in a cluster or in the cloud. Virtualization necessarily exacts performance penalties, both in resources required to run the hypervisor, and as well as in reduced performance on the virtual machine compared to running native on the physical machine. One of the important criteria of success for any virtualized platform is the ability of the underlying hypervisor to isolate performance of collocated virtual machines.

Over the last few years, hypervisors have evolved, giving better performance isolation for some workloads especially in presence of CPU and memory bound co-located applications [1]. However, virtualization of I/O operations is considered the biggest bottleneck for virtualization technologies today [14]. This raises an important question: Does the current hypervisor implementation provide sufficient performance isolation for latency sensitive applications like gaming and multimedia to guarantee the effectiveness of resource sharing? If not, what changes need to be made to provide these QoS guarantees? This is especially relevant when the applications running on multiple virtual machines of the same physical machine compete for disk/network resources.

In this paper, we investigate the performance measurement and analysis of Video on Demand in a Virtualized cloud using an RTP streaming server, a common approach in resource-constrained server environments [13]. Video on Demand places considerable real time requirements on the servers in terms of content delivery and supporting an acceptable level of QOS at the clients. Barker and Shenoy [2] focused on latency sensitive applications in cloud environments but their emphasis was on the server characteristics and ignored the client perspective. We think that the client side's view is important for two reasons:

- QOS is an important criteria in any multimedia application and it matters the most for clients. Clients simply desire the best QOS possible.
- The relevant behaviour of the server at any point in time will be reflected at the client in terms of packet loss, delay and jitter.

Overall, through our performance analysis we determine whether applications such as VOD can be handled in virtual machine environments and how I/O bound collocated workloads affect client performance. Based on the analysis, we also try to determine if current hypervisors are good at performance isolation or not. The remainder of the paper is structured as follows. We give a brief overview of KVM in section 2. In section 3, we describe related work. The testbed, experiments including methodology is presented in section 4. We present our analysis in section 5. Section 6 summarizes key findings and conclusions.

2. OVERVIEW AND BACKGROUND

In this section, we briefly discuss KVM's architecture and its features as these form the necessary background for our performance analysis ad measurement study. We have chosen KVM because it is the platform of choice for several corporate cloud technologies, including IBM, HP and Linux distribution vendors Red Hat and Ubuntu. In fact, some industry observers[1] claim that KVM is the most promising choice for future deployment of VM technology. Thus, a performance study of KVM for latency sensitive applications can be considered a timely contribution.

Kernel-based Virtual Machine (KVM) [6] represents the latest generation of open source virtualization. Figure 1 shows a KVM-based VM instance.[2] KVM is implemented as a loadable kernel module that converts the Linux kernel into a bare metal hypervisor. KVM takes advantage of hardware CPU virtualization support such as that provided by AMD and Intel to achieve efficient virtualization of CPUs and memory. The kernel module exports a device called /dev/kvm, which enables a guest mode of the kernel (in addition to the traditional kernel and user modes). Devices in the device tree (/dev) are common to all user-space processes. Each process that opens /dev/kvm sees a different map (to support isolation of the VMs). VCPUs run in process execution context as shown in Figure 2 which shows the entire VCPU execution flow.[2]

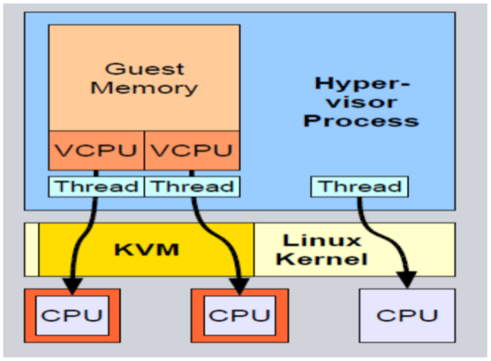


Figure 1: KVM based VM(©Siemens AG, Corporate Technology)

There are a number of mechanisms possible for the implementation of virtual machine network I/O operations. We will explain the differences between the traditional architecture and optimizations provided by alternate architectures.

[1]http://www.zdnet.co.uk/news/cloud/2011/05/20/red-hat-kvm-is-vital-for-the-clouds-future-40092818

[2]adapted from www.linux-kongress.org/2010/slides/KVM-Architecture-LK2010.pdf ©Siemens AG, Corporate Technology)

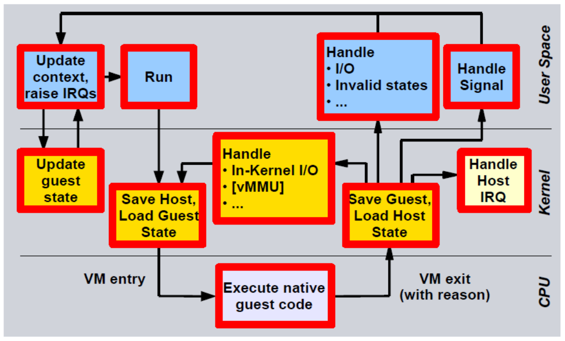


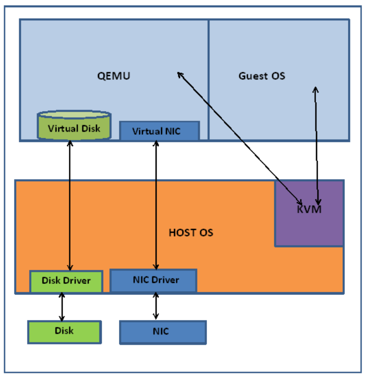
Figure 2: VCPU Execution Flow in KVM (©Siemens AG, Corporate Technology)

For each I/O operation, a VM usually exits from guest VM execution and enters the host kernel mode for each I/O request as shown in Figure 3(a). Then, the I/O operation is redirected to the host user mode (also called VM userspace) and emulated. For instance in case of network virtualization, a NIC is emulated in a VM by the I/O emulation code running in the user mode. The NIC can resemble an existing NIC such as Intel E1000 or be a paravirtualized NIC. To actually transmit and receive Ethernet frames, the I/O emulation code makes use of a point-to-point TAP networking device which has two end points: one in user space and one in kernel space. Thus, the I/O emulation code can send and receive frames by interacting with the user space end of the TAP device. The kernel end of the device can be bridged with a physical NIC to connect to the outside network. From the above description, we can see that network I/O emulation is handled by the emulation code running in user mode. We should note that to enter the user mode, the VM needs to perform a context switch to exit from the guest mode (VM exit). Since VM exits usually have high overhead, it is important to reduce their numbers.

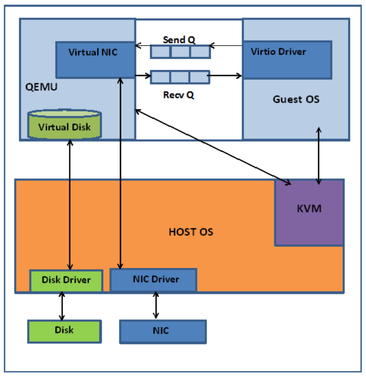
KVM uses a well-architected paravirtualized driver based on *virtio* [11] (Figure 3b). The guest runs a paravirt virtio driver and QEMU emulates the virtio device. The number of context switch from Guest->Host->User and vice/versa are reduced significantly. I/O is buffered in circular send and receive queue. Virtio device drivers are also used for block devices to do disk related operations. In Vhost-net architecture (Figure 3c), the kernel emulates virtio device through vhost-net and the guest runs the paravirt virtio device driver. The context switches take place between kernel and user, eliminating per vmexit context switches.

3. RELATED WORK

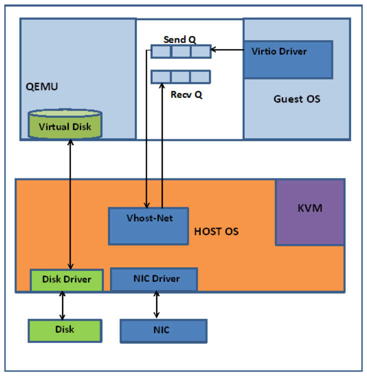
A lot of research has been done regarding the performance of virtualization in different application domains. Padala *et al.* focussed on server consolidation [10]. The authors studied the impact of virtualization on server consolidation and its performance. Wang *et al.* studied the network performance in an Amazon EC2 center [16]. They measured the processor sharing, packet delay, TCP/UDP throughput and packet loss among Amazon EC2 virtual machines and found out that even though the data center network is lightly utilized, virtualization can still cause significant throughput instability and abnormal delay variations.



(a) Traditional Network architecture



(b) Virtio-net architecture



(c) Vhost-net architecture

Figure 3: KVM Network Architecture

There has been lot of focus on the impact of virtualization technologies on high performance computing [4, 5]. These works point out that the current clouds need an order of magnitude in performance improvement to be useful to the scientific community. Nae *et al.* [9] proposed a new hybrid resource provisioning model that uses a smaller and less expensive set of proprietary data centers, complemented by virtualized cloud computing resources during peak hours in their study of MMOG virtualizaton technology.

The work done closest to our investigation is that of Barker and Shenoy [2]. They studied the impact of Xen based virtualization on latency sensitive applications like gaming and multimedia. The results reveal that the jitter and the throughput seen by a latency-sensitive application degrades due to background load from other virtual machines. The degree of interference varies from resource to resource and is most pronounced for disk-bound latency sensitive tasks, which can degrade by nearly 75% under sustained background load. The work was done on Amazon EC-2 cluster and local setup of machines. Our work is different in that we are focusing on only VOD server with a different virtualization technology and taking the client's perspective into account, which has been ignored in previous studies.

4. EVALUATION METHODOLOGY

In this section, we give an overview of our evaluation methodology. We first describe our experimental testbed and explain the scenarios, benchmarks and metrics we used for evaluating VOD performance.

4.1 Experimental Testbed

For all of our experiments we had two machines, one as a VOD server and the other generating client video requests. The host machine running VOD server is an Intel Quad core machine (2.40 GHz, 32 bit) with 3 GB of RAM, 320 GB of hard disk and 100Mbps network connection.The host is installed with latest 3.0 Linux kernel and provides hardware acceleration feature to be used with KVM. KVM is used to create two VM instances on the host. The first VM instance acts as VOD server and second one hosts an interfering workload. Each VM instance has same configuration: 1 VCPU, 512 MB of RAM, 25 GB of disk. The VMs use virtio device drivers for the block devices and network. Ubuntu 11.10 has been installed on the guest virtual machines. The client machine is running 2.4 Ghz Pentium quad core having 2 GB of RAM, 300 GB of hard disk and 1 Gbps network connection. The two machines are part of same LAN which is being served by 8 Gbps network switch. Our experimental settings ensure that neither the client machine or the network links become the bottleneck; thus, we chose the 100 Mbps NIC card on the server. It eliminates any situation where network and NIC at client could have been a bottleneck.

We used the VLC server[3] to act as VOD server. VLC has many features which make it attractive as a VOD server, including the following: ability to encode and decode variety of formats, transcoding on the fly, open source availability. We use OpenRTSP[4] to generate client requests to the VOD server from our client machine, and transmit the video data with a UDP-based RTP stream. OpenRTSP provides QOS statistics after receiving streams, is open source and gives the ability to receive streams over the network without playing them (to save CPU overhead on the client machine). The VOD server contains actual Youtube videos of 4 minutes duration in mp4 format. RTSP is used to establish and teardown the stream sessions. RTSP is used in YouTube mobile services, Real Media and Apple's Quick Time.

4.2 Scenarios

We designed two experimental scenarios:

- All clients accessing the same video: Simultaneous access is a common scenario for any VOD server with unicast delivery of highly popular videos. We expected this kind of scenario to overflow the network buffers at the VOD server because of the VBR nature of the streams, which leads to simultaneous bitrate peaks.

- All clients accessing different videos. This scenario is also common to any VOD server and holds especially true for cold videos. Our expectation in this scenario is that the disk shall become the bottlenecked resource at the VOD server because the requested video sizes

[3]http://www.videolan.org/vlc/
[4]http://www.live555.com/openRTSP/

will not fit in memory and then VOD server has to do a lot of disk I/O to fulfill the needs of different clients.

In all the above scenarios, the main idea was to increase the number of clients till the point we see start to observe issues at clients like packet loss, high delay etc. This is done to calculate the saturating point of the VOD server, which provides our baseline cases. We then reduce the number of clients until the saturating point is not observed any more. We calculate the performance metrics for all clients at this point and this act as our base case.

After establishing the base case, we started to stress the colocated VM with different types of disk and network intensive benchmarks. At the same time, the VOD server was serving the same number of clients as we got in baseline case. This helped us to observe the performance affect on VOD server when it is running along with another VM which is doing some resource intensive job. The impact is measured at the client side with the help of QOS performance metrics which are discussed in the next section.

We used following benchmarks on the colocated VM.

- For the I/O operations stress tests, we use *Filebench*.[5] Filebench is a file system and storage benchmark that allows to generate a large variety of workloads. Unlike typical benchmarks such as bonnie++, it is very flexible and allows to minutely specify (any) applications' behaviour using extensive Workload Model Language. Filebench uses loadable workload personalities to allow easy emulation of complex applications (e.g., mail, web, file, and database servers).

- For network stress testing we used *iperf*,[6] a commonly used network testing tool that creates TCP/UDP data streams and measures connection throughput.

4.3 Performance Metrics

The following metrics are used in our measurement study:

- Packet Loss Percentage: This represents the percentage of packets lost at the VOD server network stack for a stream. Since we are using UDP, a stream with large packet loss has poorer visual quality.

- Delay: This represents the time (milliseconds) between any two consecutive packets for a stream. If packets experience longer delays than required for continuous playback, then either playback will be interrupted, or a startup latency must be introduced. Quantifying the required startup latency is part of future work.

- Jitter: This represents the weighted average of variation in delay as described in the RTP standard [12].

- Delayed packets: This represents the number of packets which are delayed beyond a maximum threshold of 400 ms for a stream.

Although QOS statistics are already implemented in OpenRTSP, we extend it to add the jitter, reordering and number of delayed packet statistics, as well as to write all the QOS stats to a file. The discussed performance metrics give us an idea of the QOS as observed by the clients. In our study, we defined a highest quality stream to have no more than 0.1% packet loss, max delay under 400 milliseconds, a jitter of no more than 50 ms and no reordering. For VOD application, the indicator of good QOS is specified by Tarnai and Telekom [15] and our thresholds for QOS are based on it.

[5]http://sourceforge.net/apps/mediawiki/filebench/index.php
[6]http://sourceforge.net/projects/iperf/

5. EVALUATION

5.1 Scenario 1 (Same Video)

In this scenario, all the clients access the same video over the network from the VOD server. The total bitrate for the video was 3.2 Mbps. The baseline QOS stats are shown in Table 1. The number of clients we were able to support without observing packet loss was 12. The average values shown in the table are the average of all the clients for 5 runs. The max values in stats shows the maximum of all the values observed in 5 runs for all the clients. The video and audio streams are shown separately to get better understanding. We observed that in our base case the value of jitter were less than 1 ms and delay under 80 ms for both audio and video streams in most cases. There was no packet reordering or loss. The average network bandwidth used was around 50 Mbps and occasionally reaching maximum bandwidth of 94-95 Mbps. Jitter shows a high variability in the base case, for both the audio and the video streams. This further indicates instability, but at a very small time scale for the audio.

Metric	**Value**	**Std. Dev.**
Video avg. jitter.	0.23	0.39
Video max. jitter	1.95	
Video avg. max delay	68.17	16.63
Video max. delay	129.17	
Audio avg. jitter.	0.21	0.29
Audio max. jitter	1.51	
Audio avg. max delay	60.19	20.04
Audio max. delay	130.70	

Table 1: Scenario 1 Base Case Performance (ms)

Disk stress test.

Filebench is used for this test. We used the fileserver and webserver workloads from the available workload types. The fileserver workload emulates simple file-server I/O activity, performs a sequence of creates, deletes, appends, reads, writes and attribute operations on a directory tree. 50 threads are used by default. The workload generated is somewhat similar to SPECsfs.[7] The total file size was 1024 MB, which was greater than the size of RAM available to the VM. The web server workload emulates simple web-server I/O activity. It produces a sequence of open-read-close on multiple files in a directory tree plus a log file append. 50 threads are used in our experiments along with 80000 files. This was done to make resulting the file size equal to 1024 MB and comparable to the fileserver workload. We saw more than 100% disk utilization at all times for both workloads. The performance metrics for this test are shown in Table 2. We did not see high impact on any of the average performance metric values. Both the file server and the web server workload had similar results. There were no packet losses. Jitter

[7]http://www.spec.org./

	Fileserver		Webserver	
Metric	Mean	Std. Dev.	Mean	Std. Dev.
Video avg. jitter	0.13	0.09	0.18	0.32
Video max jitter	0.58		2.05	
Video avg. max delay	89.46	45.49	81.9	43.03
Video max delay	288.5		224.5	
Audio avg. jitter	0.15	0.17	0.21	0.26
Audio max jitter	0.99		1.2	
Audio avg. max delay	79.59	44.73	73.36	45.47
Audio max delay	279.3		212.3	

Table 2: Scenario 1 Disk Stress Case Performance (ms)

and delay do not have a correlation, as packets may experience high, uniform delay. Despite a high amount of activity in the colocated VM, the VOD server was able to handle the multimedia traffic. The main reason is that the single video being served was located in physical RAM, eliminating contention on the physical disk. The high degree of variation in the results shows that there is less predictability and though the increase in the average values is not that significant, the relative size and occurrence of outliers does increase.

Network stress test.

As mentioned earlier, *iperf* is used for this test. We did 4 kinds of tests here: TCP Send, TCP Receive, UDP Send, and UDP Receive, where TCP or UDP refers to the type of protocol we used for the iperf tests and send/receive tells the direction of traffic w.r.t. to the VM running iperf. So TCP-Send means the network-collocated VM is sending TCP traffic. In case of TCP, the client sends as much traffic as it can and there is no option to regulate it. In the UDP tests, however, we can specify the bandwidth we want to use. In our case, we specified it as 95 Mbps because this is the throughput we got when we run iperf standalone. The performance metrics for this test are shown in Table 3.

We found some interesting results here. Firstly, the packet loss for UDP-Send is substantially higher than packet loss in TCP-Send cases. When a TCP stream tends to see packet losses, congestion avoidance is used by TCP which reduces the network traffic and the VOD traffic gets a chance to send through packets easily. This causes a loss of throughput in the collocated VM (62 Mbps). With UDP, there is no flow control, and both VMs try to send as much traffic as possible, leading to high packet losses. The throughput in the collocated VM was 82 Mbps. Linux *tc* rate-limiting could have been used to provide isolation among the competing VMs. This mechanism gives each VM its guaranteed rate while fairly dividing slack bandwidth. However, such a mechanism can mitigate, but not totally eliminate interference. In fact, Barker and Shenoy [2] show that network isolation mechanisms in the hypervisor present a trade-off between mean latency and metrics such as jitter and timeouts–dedicated caps yield lower average latency, while fair sharing yields lower timeouts and somewhat lower jitter due to the ability to use unused capacity from bursty background loads.

Secondly, the reason for packet loss in the Send cases is the high number of VM exits because of high IRQs and sharing of common network channel on the host kernel.

Finally, we observe no packet loss in presence of UDP/TCP receive traffic. KVM virtio network architecture maintains separate queues for sending and receiving traffic, accounting for some of the lack of interference. Another reason can be network device drivers in Linux can use NAPI, which switches from interrupt to polling mode during high reception load to reduce the number of interrupts and improve network processing. This leads to fewer VM exits.

5.2 Scenario 2

In this scenario, all the clients access different videos simultaneously. We used the same benchmarks with same settings as in scenario 1 and also perform 5 runs for each configuration. As expected, the disk I/O becomes the bottleneck in this scenario. The videos have different bit rates and hence the network utilization was not very high. We were able to support a maximum of 15 clients with highest QOS. The reason this is higher than Scenario 1 is because of the statistical multiplexing of the VBR streams' bandwidth requirements over time, in which the spikes in bandwidth requirements do not coincide. The total size of the videos was around 450 MB, which was less than the RAM available to the VM. If we increase the number of videos close to physical RAM available, we observed delayed packets in the multimedia streams. The baseline results are shown in Figure 4 and 5. The legend on all remaining bar graphs remains the same as in these figures.

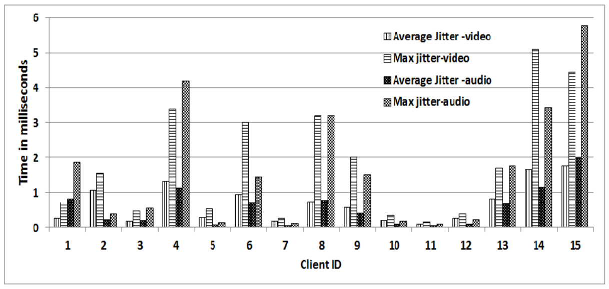


Figure 4: Base config Jitter (Scenario 2)

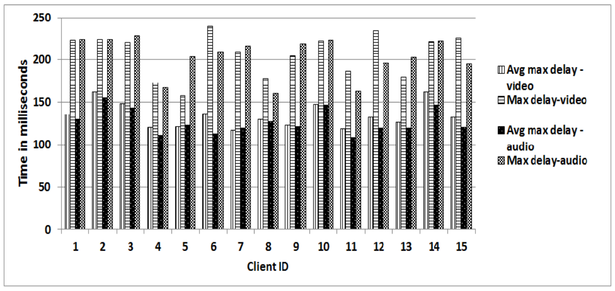


Figure 5: Base config Delay (Scenario 2)

The values for jitter were less than 1 ms for most of the cases, which is encouraging. The delay we observed ranged

Case	Avg pkt loss video %	Avg Jitter video (ms)	Max Jitter video (ms)	Avg Delay video (ms)	Max Delay Video (ms)	Avg pkt loss audio %	Avg Jitter audio (ms)	Max Jitter audio (ms)	Avg Delay audio (ms)	Max Delay audio (ms)
TCP Send	0.45	7.16	7.68	93.53	181	0.09	7.41	8.95	88	192
TCP Recv	0	0.1	0.54	69	96	0	0.13	0.77	53	79
UDP Send	60	1.09	5.41	127	226	26	0.76	2.47	186	302
UDP Recv	0	0.18	0.64	142	291	0	0.33	2.72	140	304

Table 3: Network Stress Case Performance (Scenario 1)

between 100-200 ms; for every stream it was different and did not depend on video bitrate. Each run produced different behaviour for each video, so even for the base case, more client resources would be required to accommodate late packets than in the single video case. The variation between runs was significant, but for QoS guarantees, we are not so concerned about average values, but keeping the maximum delay values observed as small as possible.

For both the fileserver and webserver workloads, we observed very high max and average jitter values for some of the videos (not shown), but on average, the jitter is below 3 ms. The delay results corresponding to the fileserver related tests are shown in Figure 6. High average maximum

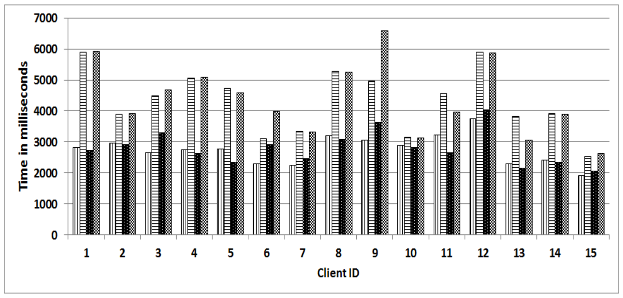


Figure 6: FS-test Delay (Scenario 2)

delay of 2-3 seconds across all runs for nearly every video and max delay of 4-6 seconds was common for the fileserver workload. The webserver workload was even worse (10-20 seconds). The high disk I/O activity leads to high number of VM exits in the collocated VM. This also means a large number of interrupts to the host kernel to service the interrupts whenever the block is ready to be read or written, and a high number of context switches between the VMs. The VOD server workload has high network activity which leads to frequent VM exits. However, these VM exits (causing a host interrupt to send network packets) could not be properly serviced as most of its time is being taken by the disk collocated workload. The result is high jitter and delay, but the delay rarely exceeds the threshold (average 8-12 delayed packets per stream). Startup-latency of a few seconds at the client could be sufficient for this delay, easily available on wired clients. For the webserver workload, more startup-latency would be required.

Figure 7 shows the jitter statistics for TCP send case while 8 shows the delay statistics for UDP send case. There were no packet losses in the TCP send case. This is because with different videos, the network utilization was moderately high at all times and TCP rate control reduced the interference provided by the collocated workload, leaving more resources

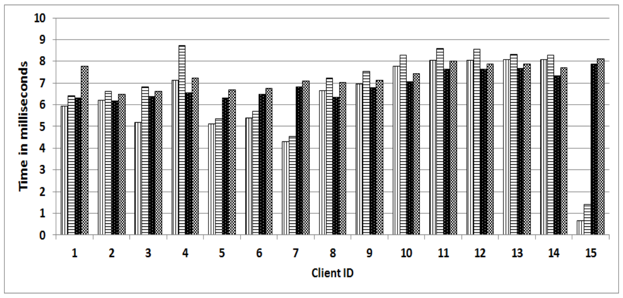


Figure 7: TCP Send Jitter (Scenario 2)

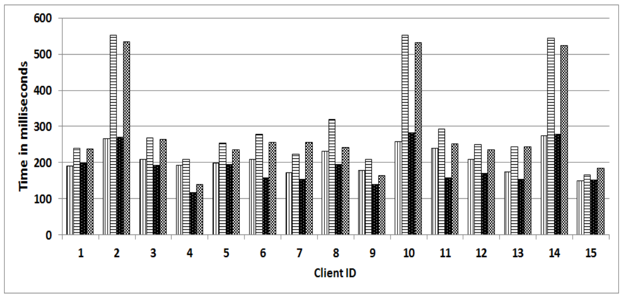


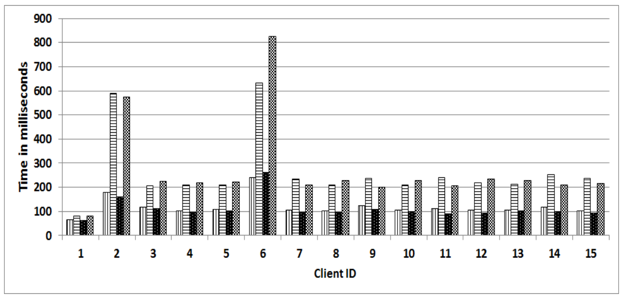
Figure 8: UDP Send Delay (Scenario 2)

for the VOD server. The percentage of packet loss in UDP remains very high, however, and was different for different videos because of its aggressive sending policy. The percentage of packet loss was different for different videos in UDP Send and it ranges from 25-50% for video stream. For audio stream, the percent of packet loss ranged between 10-30%. We observed moderate jitter (6-7 ms) in TCP Send and extremely low jitter values in UDP Send case. This may be because in case of TCP, all the packets reach the destination but with a bit of variable delay. In UDP, many packets do not reach the destination client. As a result we also see a high delay in the UDP Send case (Fig. 9), compared to TCP Send case (result not shown).

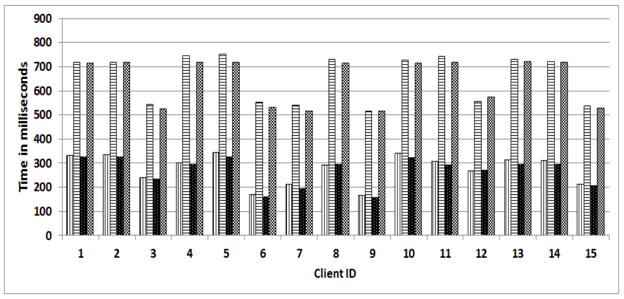
Figures 9(a) and 9(b) are shown to display the difference in delay statistics for TCP and UDP receive case. We did not see any packet losses in TCP or UDP receive test cases as was observed in scenario 1. We also did not observe high jitter and average maximum delay values. Figures 9(a) and 9(b) show high maximum delay, but that was only due to a single delayed packet in each video.

5.3 Other Scenarios

A. Experiments with Vhost-Net: The *vhost* net is a kernel-



(a) TCP



(b) UDP

Figure 9: Network Rx Delay (Scenario 2)

Case	Avg pkt loss video %	Avg Jitter video (ms)	Max Jitter video (ms)	Avg Delay video (ms)	Max Delay Video	Avg pkt loss audio %	Avg Jitter audio (ms)	Max Jitter audio (ms)	Avg. Delay audio (ms)	Max Delay audio (ms)
TCP Send	0.67	7.3	7.8	208	598	0.11	7.5	8.68	196	598
TCP Recv	0.005	0.169	0.67	77.7	112	0.0007	0.23	4.76	69.94	119
UDP Send	67.7	0.4	1.26	132.6	238	27.9	0.39	0.92	208	465
UDP Recv	0.008	0.50	3.38	187	623	0	0.68	3.24	178	600

Table 4: Performance metrics for network stress case in presence of vhost-net (Scenario 1)

level backend for virtio networking. The main motivation for vhost is to reduce virtualization overhead by moving the task of converting virtio descriptors to skbs and back from qemu userspace to the vhost net driver. For virtio-net, this means removing up to 4 system calls per packet: vm_exit for kick, reentry for kick, iothread wakeup for packet, interrupt injection for packet. We tried the single video scenario with vhost-net to see if it improves the performance. We did not see any difference in the scalability of the number of videos. The results for VOD server in presence of collocated VM running UDP/TCP send/recv are shown in Table 4.

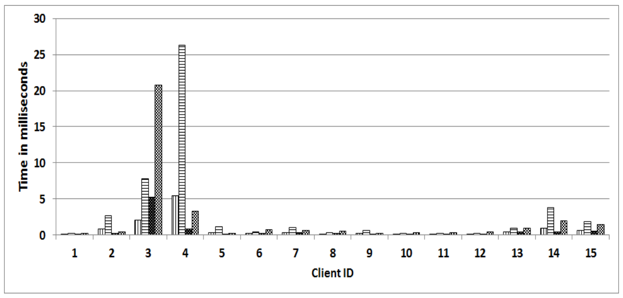
We observed an interesting result that the delay and packet loss are increased in all the cases, especially in the case of TCP. The throughput in colocated VM running iperf has also gone up to 72 Mbps in case of TCP and 85 Mbps in case of UDP, which explains the degradation of performance of VOD in our case. Vhost-net reduces the number of VM exits and hence increases the throughput in collocated VM, leading to more packet losses and high delay in VOD server.
B. colocated Disk workload - different disk/same bus: The disks we used in this experiment were of different configuration. The results are shown in Figures 10(a) and 10(b). The values for jitter, delay are reduced and the average of number of delayed packets is very low (2-3 packets) compared to scenario 1.
C. colocated Disk workload - different disk/different bus: In this experiment, we observed what happens if VOD server and Fileserver disk benchmark are executed on different disks having their own bus. The motivation behind this experiments was to check if KVM can isolate the performance for two completely different workloads on different disk and bus. The disks we used in this experiment were of different configuration. The results are shown in Figures 11(a) and 11(b). In this case, the VOD server was able to support all different videos without any delay and jitter. This is due to hardware isolation removing bus contention, as well as VM isolation efforts in software.

6. CONCLUSIONS AND FUTURE WORK

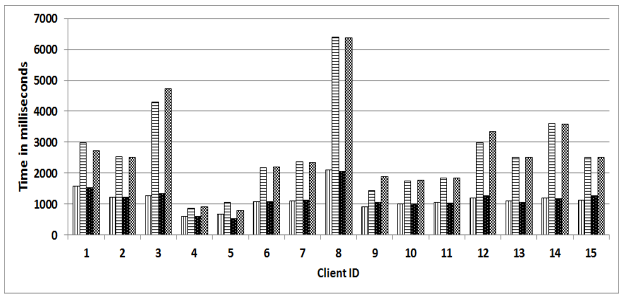
In this paper, we performed an evaluation of VOD servers in virtualized environments and measured the performance impact when such a server runs in parallel with other resource intensive workloads. We found that the VOD performance suffers badly when the collocated VM runs UDP based outbound traffic and it results in packet losses as high as 50-60%. This is expected as there is no congestion or flow control implemented in the collocated VM. For TCP and inbound UDP collocated VM workloads, VOD performs well. High delay and jitter is common when collocated VM runs the disk bound benchmark. However, the average number of delayed packets is relatively rare (10-12) and client buffering capabilities would be sufficient to eliminate this issue.

Software based enhancements like vhost-net can degrade the performance for latency based workload as we saw in our results. So, one has to be careful before using them for any application in cloud environments. Hardware based solutions like Direct I/O[8], VMDQ [3], and SR-IOV [7] can be efficient but have yet to be tested for multimedia applications. Using more resources like different disks can be helpful for multimedia workloads but it defeats the purpose of cloud, which is efficient resource utilization and cost saving. We believe that with hardware assisted solutions I/O Virtualization shall no longer reamin a big issue and many Tier 1 applications including multimedia workloads will be consolidated on virtualized servers. Future work may include experimentation with these hardware-based solutions as well as comparisons with systems that do not use virtual-

[8]http://software.intel.com/en-us/articles/intel-virtualization-technology-for-directed-io-vt-d-enhancing-intel-platforms-for-efficient-virtualization-of-io-devices/

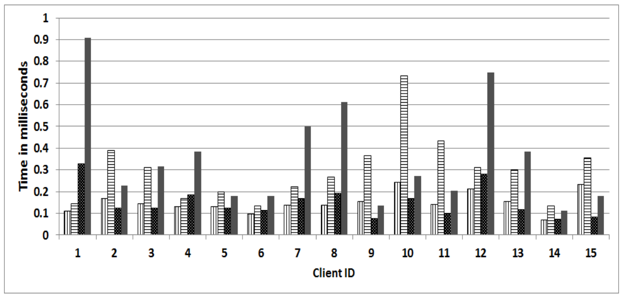


(a) Jitter

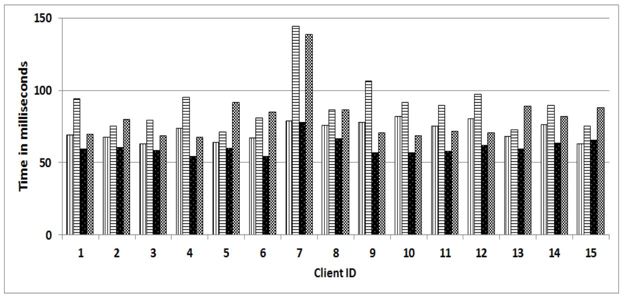


(b) Delay

Figure 10: Different disk/same bus (Scenario 2)



(a) Jitter



(b) Delay

Figure 11: Different disk/different bus - different videos

ization. Tests in a WAN would allow us to determine if the delays at the server are masked by the uncertainty in the network between client and server.

Acknowledgements

The authors would like to acknowledge the financial support of the National Science and Engineering Research Council of Canada, as well as feedback on initial versions of this research from Derek Eager.

7. REFERENCES

[1] S. Appel and I. Petrov. Performance Evaluation of Multi Machine Virtual Environments. In *2010 SPEC Benchmark Workshop*, pages 1–13, Paderborn, Germany, Oct. 2010.

[2] S. K. Barker and P. Shenoy. Empirical Evaluation of Latency-Sensitive Application Performance in the Cloud. In *MMSYS 2010*, pages 35–46, Phoenix, AZ, February 2010.

[3] S. Chinni and R. Hiremane. Virtual Machine Device Queues. Intel Corp White Paper, 2007.

[4] W. Huang, J. Liu, B. Abali, and D. Panda. A Case for High Performance Computing with Virtual Machines. In *International Conference on Supercomputing*, pages 125–134, Queensland, Australia, June 2006.

[5] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. Performance Analysis of Cloud Computing Services for Many-tasks Scientific Computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):931–945, June 2011.

[6] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. KVM: The linux Virtual Machine Monitor. In *Linux Symposium*, pages 225–230, Ottawa, ON, Canada, June 2007.

[7] J. Liu. Evaluating Standard-based Self-virtualizing Devices: A Performance Study on 10 GbE NICs with SR-IOV Support. In *IEEE International Parallel and Distributed Processing Symposium*, pages 1–12, Atlanta, GA, April 2010.

[8] R. McDougall and J. Anderson. Virtualization Performance: Perspectives And Challenges Ahead. *SIGOPS Oper. Syst. Rev*, 44(4):40–56, 2010.

[9] V. Nae, A. Iosup, R. Prodan, and T. Fahringer. The Impact of Virtualization on the Pperformance of Massively Multiplayer Online Games. In *NetGames*, pages 1–6, Paris, France, November 2009.

[10] P. Padala, X. Zhu, Z. Wang, S. Singhal, and K. Shin. Performance Evaluation of Virtualization Technologies for Server Consolidation. Technical Report HPL-2007-59, HP Labs, 2007.

[11] R. Russell. Virtio: Towards a De-facto Standard for Virtual I/O Devices. *SIGOPS Oper. Syst. Rev*, 42(5):95–103, 2008.

[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003. Updated by RFCs 5506, 5761, 6051, 6222.

[13] F. Boronat Seguí, G. Cebollada, J. Carlos, and J. Lloret Mauri. An RTP/RTCP based approach for multimedia group and inter-stream synchronization. *Multimedia Tools Appl.*, 40(2):285–319, Nov. 2008.

[14] J. Shafer. I/O Virtualization Bottlenecks in Cloud Computing Today. In *USENIX WIOV Workshop*, pages 1–7, Pittsburgh, PA, March 2010.

[15] B. Tarnai and M. Telekom. Requirements and Traffic Dimensioning for System Concepts and Architecture. Technical report, OASE/ACCORDANCE, 2010.

[16] G.I. Wang and T. S. Ng. The Impact of Virtualization on Network Performance of Amazon EC2 Data Center. In *IEEE INFOCOM*, pages 1163–1171, San Diego, CA, March 2010.