I/O Virtualization Bottlenecks in Cloud Computing Today

Jeffrey Shafer
Rice University
Houston, TX
shafer@rice.edu

ABSTRACT

Cloud computing is gaining popularity as a way to virtualize the datacenter and increase flexibility in the use of computation resources. This type of system is best exemplified by Amazon's Elastic Compute Cloud and related products. Recently, a new open-source framework called Eucalyptus has been released that allows users to create private cloud computing grids that are API-compatible with the existing Amazon standards. Eucalyptus leverages existing virtualization technology (the KVM or Xen hypervisors) and popular Linux distributions. Through the use of automated scripts provided with Ubuntu, a private cloud can be installed, from scratch, in under 30 minutes. Here, Eucalyptus is tested using I/O intensive applications in order to determine if its performance is as good as its ease-of-use. Unfortunately, limitations in commodity I/O virtualization technology restrict the out-of-the-box storage bandwidth to 51% and 77% of a non-virtualized disk for writes and reads, respectively. Similarly, out-of-the-box network bandwidth to another host is only 71% and 45% of non-virtualized performance for transmit and receive workloads, respectively. These bottlenecks are present even on a test system massively over-provisioned in both memory and computation resources. Similar restrictions are also evident in commercial clouds provided by Amazon, showing that even after much research effort I/O virtualization bottlenecks still challenge the designers of modern systems.

Categories and Subject Descriptors

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

General Terms

Measurement, Performance

Keywords

I/O Virtualization, Cloud Computing, Eucalyptus, Storage Performance, Network Performance

1. INTRODUCTION

Virtualization technology has transformed the modern datacenter. Instead of installing applications directly onto physical machines, applications and operating systems are installed into virtual machine images, which in turn are executed by physical servers running a hypervisor. Virtualizing applications provides many benefits, including consolidation — running multiple applications on a single physical machine — and migration — transparently moving applications across physical machines for load balancing and fault tolerance purposes. In this environment, the datacenter becomes a pool of interchangeable computation resources that can be leveraged to execute whatever virtual machine images (applications) are desired.

Once a datacenter is configured to provide generic computation resources, it becomes possible to outsource the physical datacenter entirely to a third-party vendor. Beginning in 2006, Amazon started allowing resources in their datacenters to be rented on-demand through their Elastic Compute Cloud (EC2) service [3]. In this canonical example of *public cloud computing*, customers can access computation resources across the Internet. Virtual machine images can be added or removed on demand. Such a capability is particularly useful for applications that vary greatly in terms of resource requirements, saving clients from the expense of building an in-house datacenter that is provisioned to support the highest predicted load.

Not every application, however, is suitable for deployment to public clouds operated by third party vendors. Medical records or credit card processing applications have security concerns that may be challenging to solve, and many other business applications may require higher levels of performance, quality-of-service, and reliability that are not guaranteed by a public cloud service. Thus, there is a motivation to maintain the administrative flexibility of cloud computing but keep all data behind the corporate firewall. This is referred to as *private cloud computing*. To meet this need, a new open-source framework called Eucalyptus was released in 2008 to allow the creation of private clouds. Eucalyptus implements the same API as Amazon's public cloud computing infrastructure, allowing for application images to be migrated between private and public servers. By maintaining API compatibility, the private cloud can be configured, if desired, to burst images onto the public EC2 systems in peak load situations, but otherwise operate entirely within the private datacenter under normal load.

In this paper, the Eucalyptus framework is tested in a variety of configurations to determine its suitability for applications with high I/O performance requirements, such as the Hadoop MapReduce framework for data-intensive comput-

This paper appeared at the Second Workshop on I/O Virtualization (WIOV '10), March 13, 2010, Pittsburgh, PA, USA.

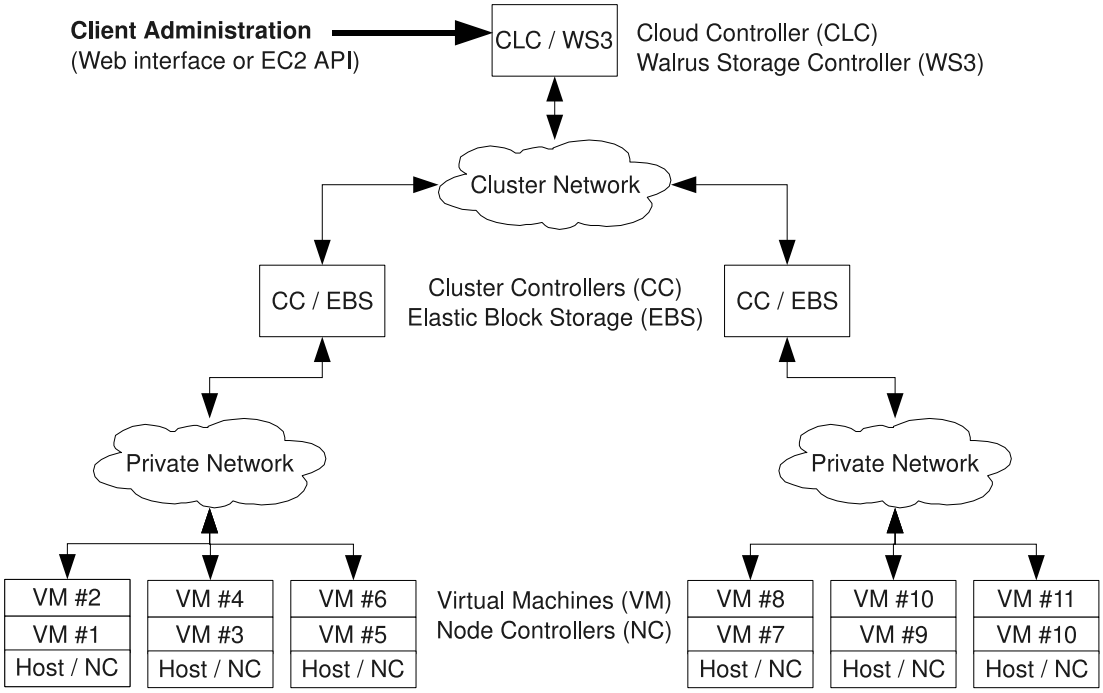

Figure 1: Eucalyptus Cluster Architecture [11]

ing. Experiments were conducted to determine the peak storage and network bandwidth available to applications running in the virtual environment. As tested, bottlenecks in the I/O virtualization framework degrade application performance significantly, preventing the raw disk and network capabilities from being provided to the virtual machines. This motivates continued development and implementation of the current best-practices in I/O virtualization.

2. EUCALYPTUS

Eucalytpus is an open-source cloud computing framework that allows the creation of private clusters in enterprise datacenters [2, 4]. Eucalyptus provides API compatibility with the most popular commercial cloud computing infrastructure — Amazon Web Services (AWS) — which allows management tools to be used in both environments and for computing images to be migrated between clouds as desired. This framework is designed for compatibility across a broad spectrum of Linux distributions (*e.g.,* Ubuntu, RHEL, OpenSUSE) and virtualization hypervisors (*e.g.,* KVM, Xen). It is the key component of the Ubuntu Enterprise Cloud (EUC) product, which advertises that an entire private cloud can be installed from the OS up in under 30 minutes (as confirmed during testing).

The arrangement of a Eucalytpus cluster and its key software services is shown in Figure 1. These services include:

Cloud Controller (CLC) — The cloud controller provides high-level management of the cloud resources. Clients wishing to instantiate or terminate a virtual machine instance interact with the cloud controller through either a web interface or SOAP-based APIs that are compatible with AWS.

Cluster Controller (CC) — The cluster controller acts as a gateway between the CLC and individual nodes in the datacenter. It is responsible for controlling specific virtual machine instances and managing the virtualized network. The CC must be in the same Ethernet broadcast domain as the nodes it manages.

Node Controller (NC) — The cluster contains a pool of physical computers that provide generic computation resources to the cluster. Each of these machines contains a node controller service that is responsible for fetching virtual machine images, starting and terminating their execution, managing the virtual network endpoint, and configuring the hypervisor and host OS as directed by the CC. The node controller executes in the host domain (in KVM) or driver domain (in Xen).

Elastic Block Storage Controller (EBS) — The storage controller provides persistent virtual hard drives to applications executing in the cloud environment. To clients, these storage resources appear as raw block-based devices and can be formatted and used like any physical disk. But, in actuality, the disk is not in the local machine, but is instead located across the network. To accomplish this, virtual machines access EBS storage through block-based disk I/O provided by the hypervisor. This bridges the guest domain with the host domain. In the host domain, a driver converts block-based access into network packets that travel across the private network and reach the remote disk. In Eucalyptus, the non-routable (but lightweight) ATA over Ethernet protocol is used for networked disk access, which requires that the virtual machine and cluster controller be on the same Ethernet segment [1]. EBS data is stored in pre-allocated files on disk, but the same protocol could be used to export entire drives directly across the network.

Walrus Storage Controller (WS3) – Walrus provides an API-compatible implementation of the Amazon S3 (Simple Storage Service) service. This service is used to store virtual machine images and application data in a file, not block, oriented format. The performance of this service was not tested in this paper.

3. PERFORMANCE EVALUATION

In performance-testing the Eucalyptus framework, the goal was to answer two questions. First, how effectively can cloud computing applications access storage resources provided by either local disks or EBS? Second, how effectively can cloud computing applications access network resources? Given the research that has been invested in I/O virtualization in recent years, and the ease-of-installation that was promised by Eucalyptus, the hope was that applications would perform efficiently out-of-the-box.

3.1 Experimental System

To test Eucalyptus, a simplified two-node cluster was used. A front-end node with two network interfaces was connected to both the campus network and a private test network, and a back-end node was connected only to the private network. Both networks ran at gigabit speeds. The private network was configured to support jumbo frames with a 9000 byte MTU to accelerate EBS performance. The ATA over Ethernet protocol used as the transport mechanism behind EBS limits the disk request size to a single Ethernet frame for simplicity, and fragments larger client requests. Thus, using the largest Ethernet frame size possible both uses the disk more efficiently and reduces the protocol overhead in relation to the payload.

The front-end node was equipped with two AMD Opteron processors running at 2.4GHz with 4GB of RAM and a 500GB hard drive. It was configured to run the CLC, CC, EBS, and WS3 services as shown in Figure 1.

The back-end node was equipped with two quad-Core AMD Opteron processors running at 3.1GHz with 16GB of RAM and a 500GB hard drive. These processors support the AMD-V virtualization extensions as required for KVM support in Linux. The back-end node was configured to run the NC service and all virtual machine images.

To provide a performance baseline, the storage and network components were profiled outside of the virtual machine. For storage, the Seagate Barracuda 7200.11 500GB hard drive has a peak read and write bandwidth of approximately 110MB/s, assuming large block sizes (64kB+) and streaming sequential access patterns. For networking, the gigabit Ethernet network has a max application-level TCP throughput of 940Mb/s for both transmit and receive. In an ideal cloud computing system, this performance would be available to applications running inside the virtual environment.

Three different software configurations were evaluated:

Eucalyptus with KVM — In the first configuration, Eucalyptus with the KVM hypervisor was used. This is a default installation of Ubuntu Enterprise Cloud (UEC), which couples Eucalyptus 1.60 with the latest release of Ubuntu 9.10 [11]. The key benefit of UEC is ease-of-installation — it took less than 30 minutes to install and configure the simple two-node system.

Eucalyptus with Xen — In the second configuration, Eucalyptus was used with the Xen hypervisor. Unfortunately, Ubuntu 9.10 is not compatible with Xen when used as the host domain (only as a guest domain). Thus, the CentOS 5.4 distribution was used instead because of its native compatibility with Xen 3.4.2. The guest VM image still used Ubuntu 9.10.

Amazon EC2 — In addition to testing Eucalyptus on a private cloud with both KVM and Xen hypervisors, the Amazon EC2 cloud was also tested. This allows the open-source Eucalyptus framework to be compared against its best-known industry counterpart. Unlike the previous two systems, which were otherwise idle (and should be considered best-case performance), the public Amazon datacenter is a black box where the load averages are a mystery. An EC2 host might have more guests running at the same time, but might be outfitted with more memory, processors, and disks too. This makes a fair comparison against Eucalyptus (running in a controlled environment) difficult. As such, the reported results for EC2 should be considered as average values. Images of Ubuntu 9.10 were prepared and executed on a large-instance EC2 node, which, although more expensive on a per-hour basis, was advertised to have the highest-available I/O quota. A large EC2 instance also reduces the number of other guests running on the host platform. Amazon provides both instance storage (presumed but not guaranteed to be a local disk) and remote disk storage via EBS.

For all virtualized systems, only a single guest domain was used. This represents a use case common in I/O intensive computing applications (such as Hadoop), where a virtualized host is primarily desired for installation convenience (ability to add/remove nodes on demand), not for the purpose of sharing a single machine or disk. Even if multiple guests were present on a single host, they would likely have dedicated disks, and would not share the same disk.

3.2 Test Applications

As a representative storage I/O intensive application that is used on EC2 systems, *Hadoop* was installed in the test virtual machine images. Hadoop is an open-source implementation of the MapReduce programming model for data-intensive computing [5]. Hadoop allows a large pool of loosely-synchronized processors to collaborate on a single application in an efficient way. It accesses data in a sequential streaming manner that is highly conducive to achieving high disk bandwidth. Inside the Hadoop framework (running in Java), a simple synthetic disk reader and writer application was used to measure and report achieved disk bandwidth over 20GB streaming runs. This test size was much greater than the physical memory — and page cache size — of the host system.

In addition to Hadoop, two other microbenchmarks were used. First, the simple *dd* utility was also used to generate storage requests similar to those produced by Hadoop, but without the computation overhead of Java and the rest of the MapReduce framework. When using dd, 20GB tests were conducted using a 64kB block size. Second, the lightweight

VMM	Target	Driver	Bandwidth	Avgrq-sz	Avgqu-sz	% Util
None	Local disk	N/A	111	1024	140	100%
KVM	Local file(*)	SCSI/sparse file	1.3	30	0.9	90%
KVM	Local disk	SCSI/phy	71.5	256	0.57	64%
KVM	Local disk	SCSI/tap	70.0	256	0.58	59%
KVM	Local disk	Virtio/phy	110	1024	60	100%
KVM	Local disk	Virtio/tap	111	1024	60	100%
Xen	Local file(*)	SCSI/pre-alloc file	58.4	995	142	100%
Xen	Local disk	SCSI/phy	65.8	251	0.87	86%
Xen	Local disk	SCSI/tap	66.0	78	30	99%
Xen	Local disk	XVD/phy	102	700	3.0	100%
EC2	Instance	Proprietary	65.8	-	-	-
None	EBS	N/A	65.2	N/A	N/A	N/A
KVM	EBS(*)	SCSI/phy	18.7	N/A	N/A	N/A
KVM	EBS	SCSI/tap	19.9	N/A	N/A	N/A
KVM	EBS	Virtio/phy	20.3	N/A	N/A	N/A
KVM	EBS	Virtio/tap	20.2	N/A	N/A	N/A
Xen	EBS(*)	SCSI/phy	26.5	N/A	N/A	N/A
Xen	EBS	SCSI/tap	19.9	N/A	N/A	N/A
EC2	EBS	Proprietary	43.1	-	-	-

Table 1: DD Write Bandwidth (MB/s) and Disk Access Pattern Measured at Host Domain. Entries marked (*) are Eucalyptus Default Configurations.

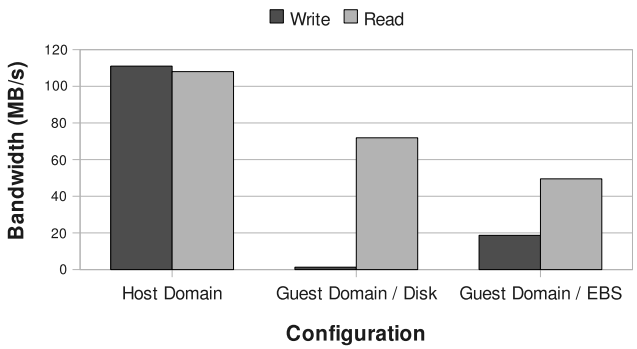

Figure 2: Eucalyptus Storage Bandwidth Overview

netperf utility was used to stress the virtual network with a minimum computation overhead.

3.3 Storage Performance

Eucalyptus storage performance with the KVM hypervisor (*i.e.,* the default Ubuntu Enterprise Cloud configuration) was initially evaluated using the simple dd utility. Three different configurations were used: a local disk installed in the back-end node and accessed directly from the host domain (for comparison purposes), a local disk installed into the back-end node that contains a file that is mapped directly into the guest domain (using the UEC default file option), and a remote disk located across the network that contains a file that is accessed from the guest domain via the EBS architecture described previously.

The results of this performance test are shown in Figure 2. The initial storage bandwidth in the virtualized environment, as compared to the control (non-virtualized) environment, is very poor. When accessing local storage, write bandwidth decreases by 98% and read bandwidth decreases by 38%. When accessing remote (EBS) storage, write bandwidth decreases by 83%, and read bandwidth decreases by 54%. These initial results motivate a deeper investigation into I/O virtualization options available with Eucalyptus.

To investigate the cause of the poor out-of-the-box performance, a new set of tests were conducted with the dd benchmark for a variety of storage I/O virtualization configurations. Several virtual machine monitors (VMMs) were used with Eucalyptus, including none (indicating that only the host domain was used for comparison purposes), KVM (the UEC default), and Xen. Amazon EC2 was also tested as the industry standard for cloud computing. The storage target was either a locally-attached disk (mapped in its entirety into the guest, instead of the file-approach used by default), or a network-attached EBS disk. Several I/O virtualization mechanisms were used, including a fully-virtualized SCSI driver (emulating a LSI Logic 53c895a controller) and a para-virtualized Virtio driver [6, 9]. Both were attached with either a tap or phy interface. Similarly, Xen used either a fully-virtualized SCSI driver or para-virtualized XVD driver. In the case of Amazon EC2, the exact nature of the I/O virtualization mechanism is proprietary.

Several metrics were reported for each configuration. First, the application-level bandwidth (as seen in the guest domain by the dd application) is provided. Next, several disk utilization metrics were measured in the host domain (not the guest domain) by the *iostat* utility to track disk access efficiency after the influence of the I/O virtualization mechanism. These metrics include `avgrq-sz`, the average disk request size measured in 512 byte disk sectors, `avgqu-sz`, the average queue depth measured in disk requests, and percent utilization, the percent of time that the disk had at least one request outstanding. These last two metrics are not recorded for the EBS configuration, where the physical disk

VMM	Target	Driver	Bandwidth	Avgrq-sz	Avgqu-sz	% Util
None	Local disk	N/A	108	512	0.94	96%
KVM	Local file(*)	SCSI/sparse file	71.9	450	1.1	96%
KVM	Local disk	SCSI/phy	70.5	512	0.7	68%
KVM	Local disk	SCSI/tap	56.7	512	0.7	62%
KVM	Local disk	Virtio/phy	76.2	512	0.5	57%
KVM	Local disk	Virtio/tap	76.1	512	0.5	62%
Xen	Local file(*)	SCSI/pre-alloc file	83.1	241	1.6	99%
Xen	Local disk	SCSI/phy	42.8	14	22.4	99%
Xen	Local disk	SCSI/tap	35.3	11	22.0	99%
Xen	Local disk	XVD/phy	94.8	128	2.2	99%
EC2	Instance	Proprietary	113.0	-	-	-
None	EBS	N/A	55.8	N/A	N/A	N/A
KVM	EBS(*)	SCSI/phy	49.5	N/A	N/A	N/A
KVM	EBS	SCSI/tap	46.2	N/A	N/A	N/A
KVM	EBS	Virtio/phy	48.0	N/A	N/A	N/A
KVM	EBS	Virtio/tap	46.3	N/A	N/A	N/A
Xen	EBS(*)	SCSI/phy	51.4	N/A	N/A	N/A
Xen	EBS	SCSI/tap	47.8	N/A	N/A	N/A
EC2	EBS	Proprietary	68.0	-	-	-

Table 2: DD Read Bandwidth (MB/s) and Disk Access Pattern Measured at Host Domain. Entries marked (*) are Eucalyptus Default Configurations.

is on the other side of the network. Further, they are not available in EC2, where the host domain is restricted from customer access.

There are many observations that can be made from inspecting the results shown in Table 1 and Table 2 for write and read tests, respectively. First, the default Eucalyptus configuration for local storage in KVM — mapping a sparse file on local disk into the guest domain — has terrible write performance. Part of the poor performance is due to the overhead of growing the file dynamically during write operations, and part is due to the inefficient disk access size. As measured, each disk write operation averages 15 sectors per access, or 7.5kB. To determine the best-case performance of the target disk at this small access size, the dd microbenchmark was run in the host domain with the same access size. The maximum write bandwidth measured was 15.7MB, compared to a write bandwidth of over 100 MB/s for the same disk using 64kB access sizes. In addition to small access sizes, the disk is not even utilized 100% of the time, degrading performance further when the host waits on I/O requests from the guest. Other tests that used the entire local disk or pre-allocated a file on disk (like Xen) did not exhibit this severe performance degradation. Thus, for any type of data-intensive application (like the target application, Hadoop), sparse file-based access should not be used.

Second, para-virtualized drivers enabled higher bandwidth communication between the guest domain and storage device compared to fully-virtualized devices. This is true regardless of whether the disk was locally attached or accessed across the network. For example, when writing to a locally-attached disk in KVM, the virtio driver was able to achieve full disk bandwidth by maintaining a large queue (60 elements) of large requests (512kB each), thus keeping the disk busy 100% of the time.

Third, although para-virtualized I/O in KVM achieved higher read bandwidth to a local disk than fully-virtualized I/O, the best achieved bandwidth of 76MB/s for a local disk is only 70% of the non-virtualized disk bandwidth of 108 MB/s. The challenge in maintaining high read bandwidth is the synchronous nature of the test application, which does not queue requests but rather sends a single request to disk, waits for the reply, and then issues another request. Because of the increased latency between the guest and host domain, the disk sits idle in KVM for 30-40% of the time. This is a significant difference from writing to the disk. When writing, the para-virtualized driver can immediately acknowledge a synchronous write (allowing the client to continue), aggregate multiple writes in the guest domain, transfer them in a batch to the host domain, and present them in a queue for the disk to commit when ready. When writing, large queues were maintained for the drive to process at all times, keeping the disk 100% busy and operating at high bandwidth.

To overcome this bottleneck in storage reads, asynchronous I/O (AIO) should be used by applications running in the guest domain to provide a queue of read requests instead of a single request [7, 10]. This poses a difficulty in Linux, however, where AIO is only enabled when files are opened using the O_DIRECT mode, which bypasses the page cache and (in a non-virtualized system, at least) does a direct transfer from the device into the user-space buffer. Otherwise, the system transparently reverts to using standard synchronous I/O.

Finally, the performance of writing to EBS storage when accessed inside a guest domain is significantly worse than writing to the same EBS device from inside the host domain. For example, KVM degrades performance by almost 68%. This indicates that the storage virtualization mechanism, not the network transport mechanism, is the bottleneck in this network storage architecture. This bottleneck is less

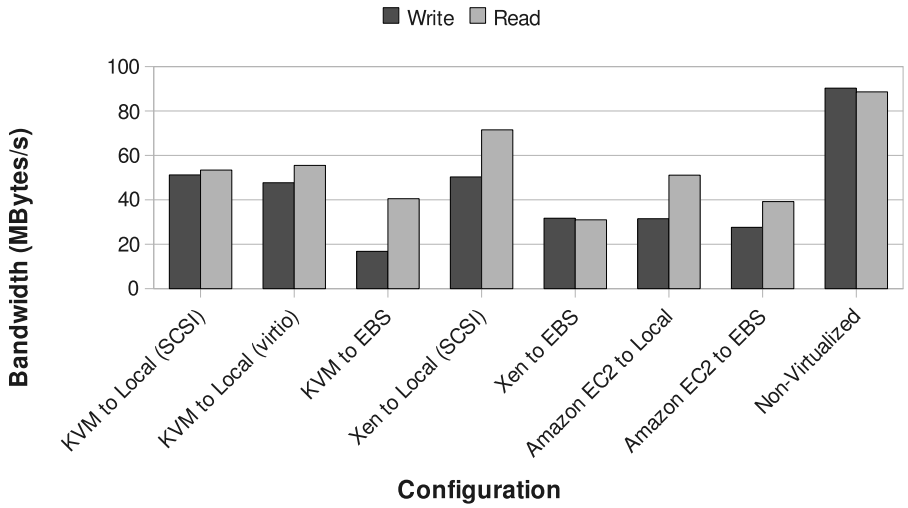

Figure 3: Hadoop Storage Bandwidth

severe when reading from EBS storage, which experiences a performance degradation of approximately 10%.

In addition to using the dd microbenchmark, storage I/O performance in Eucalyptus was also evaluated using Hadoop, a data-intensive computing framework that implements the MapReduce programming model. Inside Hadoop, a simple synthetic reader and writer program was used to test disk bandwidth. The same configurations of local and remote (EBS) storage were evaluated using the KVM and Xen hypervisors for both full (SCSI) and para-virtualized (virtio or XVD) drivers. Results are compared against Amazon EC2 using its default (proprietary) approach, and Hadoop running in a non-virtualized environment. These results are shown in Figure 3.

Several observations can be made from these experimental results. First, the fully-virtualized SCSI and para-virtualized virtio drivers in KVM only achieve 60% of the bandwidth of the non-virtualized Hadoop performance (shown at the right of the figure). Second, accessing network-based EBS storage incurs an additional performance penalty beyond the limitations imposed by moving block data between the virtual machine and the driver domain. Third, any of the Eucalyptus local disk configurations outperform the Amazon EC2 instance, although it is important to remember that the Eucalyptus system was otherwise idle, while the load factor of the Amazon datacenter is unknown.

3.4 Network Performance

In addition to storage bandwidth, the network performance of Eucalyptus on KVM and Xen was also tested. For KVM, both fully virtualized (e1000, the default) and para-virtualized (virtio) drivers were used. Network performance on Amazon EC2 was not tested because such results would be heavily influenced by datacenter network congestion. Network bandwidth was measured between the guest virtual machine running on the back-end node and both the driver domain (Dom0) and front-end node (which is not virtualized). Ideally, the network bandwidth to the driver domain should be several gigabits per second or more, and only be limited by available CPU resources on the heavily over-provisioned system. Further, the ideal external network bandwidth to the front-end node should be limited by the network wire speed. The results of these tests can be seen in Figure 4.

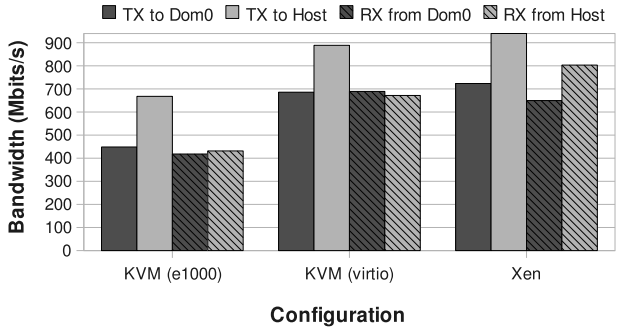

Figure 4: Netperf Network Throughput

Several results can be observed in the experimental results. First, the achieved bandwidth to the driver domain (for both transmit and receive) was unexpectedly low. In fact, bandwidth to the driver domain was lower than bandwidth to a different host, despite the fact that reaching another host should incur much greater latency due to traversing network interfaces at both ends of the network path! The bandwidth never exceeded 700 Mb/s even with para-virtualized drivers, even though bandwidth in this test should be limited only by processor speed, and the processor was idle in excess of 90% of the time. Second, the achieved bandwidth failed to saturate the gigabit NIC to the front-end node, except when transmitting data using the Xen hypervisor. This demonstrates that there is still a large gap between the current performance of these virtualization systems, and the best practices described in current research, which is able to sat-

urate a 10Gb/s Ethernet link from a guest domain [8].

4. CONCLUSIONS

Applications running in the Eucalyptus cloud computing framework suffer from I/O virtualization bottlenecks in KVM and Xen. The default configurations that use fully-virtualized drivers and on-disk files (instead of full devices) perform poorly out of the box. Even using the best default configuration (with the Xen hypervisor), the guest domain can achieve only 51% and 77% of non-virtualized performance for storage writes and reads, respectively. Mapping in full disk devices and using para-virtualized drivers narrows but does not fully close this performance gap, particularly when real applications (such as Hadoop) are used instead of microbenchmarks. Re-writing applications running in the guest domain to use asynchronous I/O could prove useful in providing larger numbers of read requests to the virtual I/O mechanism, and thus maintain higher disk utilization. In the case of data-intensive computing using Hadoop, this would only necessitate modifying the Hadoop framework, not every MapReduce application written for Hadoop. Further, modifying the EBS storage architecture such that the networked disk driver runs inside of the guest domain instead of the driver domain might provide benefits in terms of simplifying the software stack and reducing any mismatch between storage and network virtualization mechanisms. Although this adds some administrative complexity to the guest domain, it has a performance advantage in that the virtualized network interface — not the slower virtualized disk interface — can be used to convey storage data into and out of the guest domain.

5. REFERENCES

[1] AoE (ATA over ethernet). `http://support.coraid.com/documents/AoEr11.txt`, 2009.

[2] Eucalyptus open-source cloud computing infrastructure - an overview. Technical report, Eucalyptus, Inc., August 2009.

[3] Amazon web services. `http://aws.amazon.com`, January 2010.

[4] Eucalyptus community. `http://open.eucalytpus.com`, January 2010.

[5] Hadoop. `http://hadoop.apache.org`, 2010.

[6] Virtio para-virtualized drivers. `http://wiki.libvirt.org/page/Virtio`, 2010.

[7] C. Hellwig. The KVM/qemu storage stack. `http://events.linuxfoundation.org/eus09c4`, 2009.

[8] K. K. Ram, J. R. Santos, Y. Turner, A. L. Cox, and S. Rixner. Achieving 10 gb/s using safe and transparent network interface virtualization. In *VEE '09: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 61–70, New York, NY, USA, 2009. ACM.

[9] R. Russell. virtio: towards a de-facto standard for virtual I/O devices. *SIGOPS Oper. Syst. Rev.*, 42(5):95–103, 2008.

[10] S. R. Seelam and P. J. Teller. Virtual I/O scheduler: a scheduler of schedulers for performance virtualization. In *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*, pages 105–115, New York, NY, USA, 2007. ACM.

[11] S. Wardley, E. Goyer, and N. Barcet. Ubuntu enterprise cloud architecture. Technical report, Canonical, August 2009.