CULTIVATING CLOUD COMPUTING
A Performance Evaluation of Virtual Image Propagation & I/O Paravirtualization

Django Armstrong and Karim Djemame
School of Computing, The University of Leeds, Leeds, U.K.

Keywords: Cloud computing, Performance evaluation, Image propagation, Paravirtualization.

Abstract: Cloud Computing continues to be a rapidly evolving and prevalent paradigm where Quality of Service has a pivotal role to play in guaranteeing performance and provisioning of resources on-demand in a timely fashion. Improvements to the performance of Cloud technology ensure provider profitability and an increased number of applications that can make use of a Cloud where overheads would have otherwise limited usage. This paper presents the results of a quantitative evaluation into the performance overheads of propagating Virtual Machine images to physical resources, at the Infrastructure as a Service layer and then accessing the images, via a Hypervisor's virtual block I/O device. Two Virtual Infrastructure Managers are evaluated: Nimbus and OpenNebula, along side two Virtual Machine Managers: XEN and KVM. Benchmark results demonstrate Nimbus out of the box outperforming OpenNebula and the performance of XEN exceeding KVM in a greater number of benchmark tests. Conclusions are drawn from the results on the suitability of these technologies for data intensive applications and applications requiring highly dynamic resource sets, where making an uninformed decision on what technology to use could prevent an application reaching its full potential.

1 INTRODUCTION

The prevalence of Cloud Computing as a distributed paradigm that can deliver economic, automation and flexibility benefits far beyond more traditional IT infrastructure has and continues to thrust Cloud services into the lime light. Infrastructure as a Service (IaaS) providers play a pivotal role in the Quality of Service (QoS) provisioned in the majority of Cloud architectures comprised of an interchangeable multilayer software stack (Vouk, 2008). Virtualization, as the fundamental resource building block of IaaS, is critical to maintaining acceptable levels of performance to prevent breaches in Service Level Agreements (SLA) and thus increasing the overall profitability of a Cloud (Xiong and Perros, 2009). This provides primary motivation for efficient Hypervisor design and remains a limiting factor in what applications are deployable and can take full advantage of a Cloud. Although research on the topic of Virtualization is not new, it has seen a resurgence of interest in recent years in the problem domain of Cloud Computing (Jun Zhu et al., 2010; Imada et al., 2009; Xu et al., 2009). One such area of interest is Virtual Machine (VM) lifecycle management (Hansen and Jul, 2010; Goiri et al., 2009), where guarantees that a VM will be on-line within a certain timeframe are of importance to the rate at which IaaS can react to changes in demand (Van et al., 2009). Another area of interst is VM I/O (Kesavan et al., 2010), which can have adverse affects on application performance (Dong-Jae Kang et al., 2008).

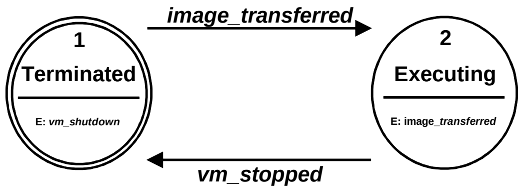


Figure 1: Simplified life cycle states of a non persistent VM.

In the life cycle of a VM managed by an IaaS, overhead is associated with transitions of state illustrated in Figure 1. IaaS requires time to provision resources, transfer data and confirm termination. A trade off exists where at some point reducing these overheads incurs a penalty associated with polling and the contention of resources. In this paper, we focus on analyzing both the overheads of resource acquisition and data access, i.e. propagation of non-persistent VM images to physical resources and the use of paravirtualized I/O device drivers to access the

Armstrong D. and Djemame K..
CULTIVATING CLOUD COMPUTING - A Performance Evaluation of Virtual Image Propagation & I/O Paravirtualization.
DOI: 10.5220/0003388305390550
In *Proceedings of the 1st International Conference on Cloud Computing and Services Science* (CLOSER-2011), pages 539-550
ISBN: 978-989-8425-52-2
Copyright © 2011 SCITEPRESS (Science and Technology Publications, Lda.)

data held within an image after propagation. The aim of our work is to expose the limitation of Cloud technologies through performance evaluation with results that provide insight into the rapidly evolving landscape of Cloud tools in the context of commodity hardware.

We compare via benchmarking two Cloud infrastructure managers Nimbus: (Kate Keahey et al., 2007) and OpenNebula (Sotomayor et al., 2009), alongside the two latest versions of each of the open source Hypervisors: XEN (Barham et al., 2003) and KVM (KVM, 2010), using a combination of synthetic benchmarks. We find that Nimbus can boast a reduced overhead, compared to OpenNebula, when initializing and terminating VMs and that the additional support of a Grid transfer protocol in Nimbus can reduce transfer times by 46%. In addition, we find that KVM guests perform poorly in contrast to XEN in most benchmark tests regarding the write performance to a VM image but this has been partially rectified in a new release of KVM.

We argue that previous work on the subject of hypervisor I/O performance is now outdated due to the pace of development surrounding paravirtualized device drivers. We infer that overheads have the potential to influence Cloud performance and thus the usage patterns of IaaS providers. If predominant Cloud technology is not chosen wisely, it can prevent data intensive applications from reaching their full potential and applications that require highly dynamic resource sets from scaling efficiently. We draw the conclusion that reducing overheads could lead to an increase in the pace of Cloud adoption. A reduction in resource acquisition waiting times enables quicker reaction, by an IaaS, to a changing environment and reduces operating costs via a similar reduction in the number of virtual resources needed to be provisioned for a given quantity of concurrent client requests.

The remainder of the paper has the following structure. Section 2 and 3 outline the Cloud infrastructure managers Nimbus and OpenNebula, and the Hypervisors XEN and KVM, respectively. Section 4 describes the experimental environment. Section 5 presents and evaluates experimental findings. Section 6 reviews the contributions of the paper and critiques related work. Finally, Section 7 presents the conclusion of the paper and future work.

2 VIRTUAL INFRASTRUCTURE MANAGEMENT

An electrical utility provider needs a system in place to provision and monitor resources to ensure service reliability and performance, keeping pace with changes in demand so that consumer usage patterns and high resource contention do not adversely affect the QoS provided. Cloud IaaS providers are no different in that they use Virtual Infrastructure Managers (VIM). Virtual resources need to be brought on and off-line as required and monitored to assess status so that intelligent decisions can be made on how best to use the underlying physical resources for the business objectives of an IaaS provider. VIMs achieve this through a scheduling component that assigns VMs to physical resources with feedback gathered from monitoring services for both the physical and virtual infrastructure. A scheduling component orchestrates with others in the system to: i) assess the needs of a VM, ii) provision a suitable physical resource and iii) transfer a VM image from a central repository over the network via an available network protocol. Once transferred a Hypervisor adapter component is used to execute the VM image on the physical host machine and bring the virtual resource on-line.

2.1 OpenNebula & Nimbus

OpenNebula (Sotomayor et al., 2009) is an implementation of the research being performed and led by Reservoir (Rochwerger et al., 2009), a European Union FP7 funded research initiative in virtualized infrastructure and Cloud computing. It can be used as a Virtualization tool for the manipulation of local virtual infrastructure within datacenter clusters for the creation of private Clouds. Public Cloud support, via a selection of management interfaces, exposes functionality of a remote provider's VM, storage and network resources using de facto standards through a locally accessible portal. Hybrid Clouds, made possible through the combination of public resources and local virtual infrastructure, enable highly scalable application hosting environments. The design of OpenNebula efficiently manages multiple types of workloads with emphasis placed on addressing business requirements and use cases. A black box approach to design facilitates the homogeneous management of VM Operating Systems (OS) and services.

Nimbus (Kate Keahey et al., 2007) builds on past research into Grids by reusing many of the standards and technologies invented by the Grid community, particularly those used in the Globus Toolkit. It provides an upgrade path to the Cloud for organizations using Grids. This is enabled via the features of Nimbus that make use of Grid resources and its integration with familiar resource schedulers, such as PBS, to schedule VMs. Development of Nimbus has placed emphasis on use cases applicable to the needs of the scientific community but many non-scientific applications are still well suited to the virtual infrastructure it provides. The components of Nimbus are modular in design with hypervisor agnostic support enabled via Libvirt and support for Amazon EC2 via a SOAP based API for invocation and securing of off-site resources. Nimbus also supports a Web Service Resource Framework (WSRF) frontend for controlling virtual infrastructure and context broker and agent components enabling automated contextualization of VM images for "one-click" clusters (Keahey and Freeman, 2008).

3 VIRTUAL MACHINE MANAGEMENT

A Virtual Machine Manager (VMM) or Hypervisor, partitions a physical host machine though the use of three generalized techniques: Full Virtualization, Paravirtualization and Hardware Assisted Virtualization (HVM) and is responsible for controlling the life cycle and resource utilization of local VMs. These techniques provide a layer of abstraction away from the underlying physical hardware. The techniques provide a complete virtual hardware environment in which a guest OS can execute in isolation and where resources are multiplexed transparently between concurrently executing OSs.

Full Virtualization (FV) involves the creation of hardware devices purely in software to provide an adequate supply of simulated hardware for a guest Operating System (OS) to run unmodified. This comes at a considerable performance penalty due to the interpretation of hardware operations in the guest (Popek and Goldberg, 1974). Paravirtualization (PV) imitates a device interface using a far more direct path to handle devices inside a VM and can achieve better performance than FV. A downside of this technique is that it requires the installation of altered device drivers into a guest OS. A benefit is that this reduces the amount of time a guest spends accessing the device by relocating execution of critical tasks to its host where such tasks are more performant. HVM of a guest utilizes the additional hardware capabilities of an underlying host and provides the best performance of all the Virtualization techniques. Currently this takes the form of Virtual Machine Extensions (VMX) within the instruction set of a host processor. This accelerates and isolates context switching between processes running in different VMs, increasing computational performance as instructions are directly passed to the host processor without having to be interpreted and isolated by the VMM. Unfortunately this technique comes at the expense of limiting the guest to using the same instruction set as the host. Complete support for HVM of all computer subsystems, i.e. I/O peripherals, has yet to be fully realized in commodity computer hardware. However the performance benefits of HVM I/O have been explored using directed I/O Virtualization extentions (Dong et al., 2009).

3.1 XEN & KVM

Historically XEN has concentrated on the development of PV guests. Recently both VMMs support multiple Virtualization techniques often referred to as Hybrid Virtualization (Nakajima and Mallick Asit K., 2007). Hybrid Virtualization combines the principles of both HVM and PV to obtain near native performance for guest OSs. This however has the disadvantages of both techniques; altered OS device drivers are necessary, along with modern VMX supporting hardware. XEN and KVM both support FV through the utilization of core components from QEMU, a "generic and open source machine emulator and virtualizer" (QEMU, 2010) that provides emulation of other devices besides CPU architectures. When QEMU acts as a machine emulator it provides emulation of different CPU architectures through binary translation. When QEMU is used as a virtualizer in combination with KVM or XEN the features that enable emulation of other devices besides CPUs can be used to virtualize an OS when paravirtualized device drivers do not exist or hardware support for HVM is limited but this approach has an associated performance cost. Before the combination of multiple Virtualization techniques in KVM, the consolidation of an organization's current hardware using KVM was not feasible if its infrastructure did not support the instruction set extensions necessary for HVM guests. KVM conversely provided an excellent foundation for the creation of new virtual infrastructure through a reduction in the number of physical machines required for peak operating demand and thus lowers hardware running and setup costs. XEN on the other hand, with its better support for paravirtualized guests, was more appropriate for deployment onto older hardware and still enabled consolidation.

Comparing XEN and KVM further, the lack of support for fully paravirtualized guests in KVM across all OSs, such as the closed source and proprietary Microsoft Windows, has the potential to reduce performance. Alternatively the costs of porting paravirtualized device drivers (Barham et al., 2003) to these OSs for XEN do not exist. XEN is a more mature Virtualization solution and has been developed for far longer than KVM pertaining to greater stability. However KVM continues to be on the forefront of implementing new Virtualization techniques and utilizes the latest research into HVM. This provides greater performance improvements over the same implementations in software with the downside of requiring state-of-the-art hardware. One such technique introduced recently was hardware page-table walkers that reduce memory address-translation overheads and remove the need for shadow page tables at the memory management unit level, providing memory savings and performance benefits that were not initially available in XEN.

Another non-functional comparative advantage of KVM over XEN is that KVM is less pervasive due to its hypervisor classification. XEN is classified as a Type I hypervisor and KVM as a Type II hypervisor according to (Goldberg, 1972). Type I hypervisors run directly on the host hardware with the guest OS running one level above the hypervisor. In the case of XEN an installed micro-kernel is booted before the administrative guest "Domain0", which is used to configure other guests. On the other hand, Type II hypervisors run from within a conventional OS, where the hypervisor operates as a distinct second software layer and the guests execute above in a third layer. KVM is comprised of a single module probed into a standard Linux kernel. The comparative advantage of this is that a considerably smaller code-base has to be maintained, which lowers the risks of introducing bugs and reduces the amount of code to optimize.

3.2 Paravirtualization of Block I/O Devices

KVM and XEN have different PV architectures for accessing virtual block devices or virtual Hard Disk Drives (HDD) within a guest OS. One of the aims of our work is to evaluate the performance of these devices to ascertain the suitability of virtual infrastructure for data intensive applications.

KVM has adopted Virtio (Russell, 2008), a Linux standard for virtual I/O device driver Application Binary Interfaces (ABI), for PV of block storage. With Virtio the guest OS is "aware" of the virtual environment in which it is running, in essence cooperating with the VMM, attaining higher performance and the benefits of PV. Virtio uses a layer of abstraction and a memory ring buffer as a transport for data between a VM and its host as expressed in Figure 2.

This provides the ability to write generic front-end virtual device drivers and arbitrary back-ends to support different device types for different OSs and VMMs. It removes the need to maintain multiple sets of virtual device drivers for each brand of VMM available.

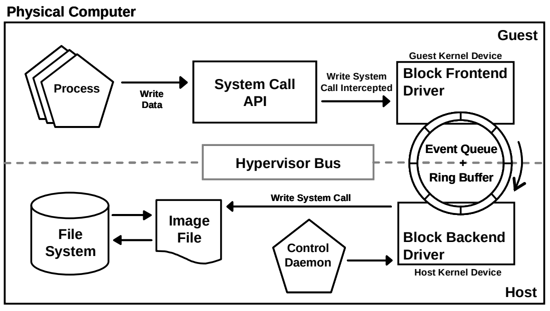


Figure 2: A simplification of virtual block I/O PV.

The XEN approach is very similar to that of KVM as Virtio is based considerably on the works of the XEN developers. XEN supports block devices through a "hypercall" ABI that makes use of an altered version of the Linux blkback device, used for user land access to block devices, named blktap or "block tap". The blktap device is used in combination with a frontend driver embedded in a guest. XEN version 4.0.1 introduces blktap2 the successor to the old blktap1 disk backend driver. Both versions of blktap will be tested to ascertain if there are any performance improvements or regressions from this new implementation. Two versions of KVM of differing maturity will also be tested again to see how performance has changed over time. Paravirtualized support for non-modifiable OSs, such as Windows, has been implemented in both KVM and XEN with emulated PCI devices replacing the ABI in traditional PV. The performance of these devices has been omitted from the scope of this paper but will make an interesting addition in future work.

4 EXPERIMENTAL DESIGN

This section of the paper introduces the testbed architecture, the benchmarks used to assess performance and the experimental methodology that discusses the selection of independent and dependent variables for each of the benchmarks.

4.1 Testbed Architecture

The experimental testbed was comprised of Dell commodity servers. Each server consists of a four core X3360 Intel Xeon CPU, running at the default clock speed of 2.83GHz and a total of 4GB of RAM (four modules of 1GB DDR2 at 800Mhz). In addition, each server utilized a single 3.5 inch Western Digital RE3 250GB SATA HDD (Model: WD2502ABYS), with 16MB of cache and a spindle speed of 7200 RPM.

The machines connect via Gigabit Ethernet using a Broadcom PCI-E NIC (Model: BCM95722). The file system in use on both the physical machines and the VMs is EXT3. VMs were not provisioned concurrently and each instance used all available hardware with the exception of half the available RAM (2GB) of the host. The VM images are contained in raw format and stored on top of the host file system. The OS present on both machines is Centos release 5.4 (Final). The following hypervisor versions are used: KVM versions 83 and 2.6.32.24; XEN versions 3.4.3 and 4.0.1. Two Centos based Linux Kernel versions are used to test the older versions of KVM and XEN: 2.6.18-164.15.1.el5 for testing the performance of the native host and KVM version 83 guests; and 2.6.18-164.11.1.el5xen for testing the performance of XEN Dom0 and XEN 3.4.3 guests. The same vanilla Linux Kernel version 2.6.32.24 is used to test KVM version 2.6.32.24 and XEN version 4.0.1. Version 2.6 of Nimbus and version 2.0.1 of OpenNebula are deployed on the testbed. Globus Toolkit version 4.0.8 is installed enabling support for GSIFTP. The following versions of the benchmarking software, with justifications in the next subsection, are used in the experiments: Bandwidth Monitor NG version 0.6 (BWM-NG, 2010), IOzone version 347 (IOzone, 2010) and Bonnie++ version 1.03e (Bonnie++, 2010).

4.2 Benchmarks

Three benchmarks were used in the performance evaluation. The first a bespoke image propagation benchmark for reviewing the overheads of transferring a VM image between two nodes on a Cloud system. The two other benchmarks were used to expose the performance of VMM block I/O systems.

The image propagation benchmark integrated Bandwidth Monitor NG (BWM-NG) as a tool to measure the input and output of the NIC from where the VM image repository and VIM were resident, referred to in this paper as the 'head node'. The benchmark involved the creation of: i) scripts to amalgamate the commands necessary for a user to provision a VM, ii) verify the VM was on-line and executing and iii) terminate the VM in an automated fashion so that iterations of the experiment could be repeated with relative ease. Each action was timed and compared to the start and end of the data transfer recorded by BWM-NG. This exposed the overheads of each stage of the image propagation, from the head node to the destination node where a VM was deployed. The image propagation benchmark takes a single parameter; the size of image to be transferred. The benchmark was wrapped around the three platforms: OpenNebula using the SCP protocol, Nimbus using the SCP protocol and Nimbus using GSIFTP.

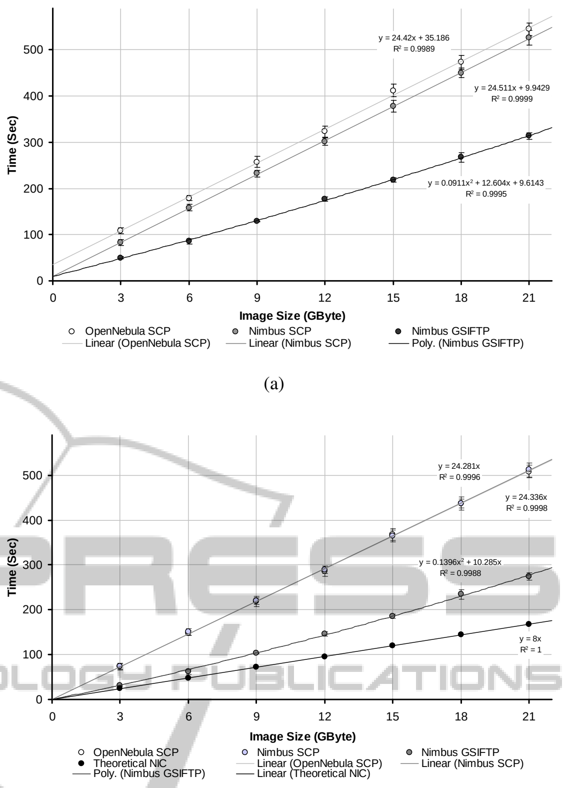
Two synthetic benchmarks: IOzone and Bonnie++, were chosen for the performance evaluation of virtual block I/O devices. These synthetic benchmarks try to encompass all possible parameters that could represent different workloads, such as database software writing small files randomly to file servers reading large files sequentially. IOzone is a benchmark that generates and measures a selection of file operations to analyze the performance of a file system. Bonnie++ is a benchmarking suite with the ability to perform several tests to assess the performance of a file system and underlying HDD. In addition to monitoring the above, Bonnie++ also monitors the CPU overhead of a given test which gives a good indication of the efficiency of the block I/O implementation. Two benchmarks were also chosen to enable the validation of results so that any anomalies can be ruled out as implementation specific issues. A test file size of twice the amount of available RAM was used in both to prevent CPU and RAM caches from biasing the results.

The following IOzone tests, with accompanying definitions, were chosen for the experiments: *Write* – Measures the performance of writing a new file, inclusive of writing file metadata; *Re-write* – Measures the performance of writing to a file that already exists, omitting much of the workload required in writing metadata; *Read* – Measures the performance of reading an existing file; *Re-read* – Measures the performance of reading a recently read file, illustrative of caching affects that can improve performance as reads are satisfied by cache; *Random Read* – Measures performance of reading random locations within a file, indicative of the performance of cache hierarchy and seek latency; *Random Write* – Measures performance of writing to random locations within a file, indicative of the performance of cache hierarchy and seek latency.

The subsequent Bonnie++ tests, with accompanying definitions, were recorded during the experiment: *Sequential Throughput* – The number of blocks of data that can be read or written adjacent to one another in respect to the physical medium, in kilobytes per second; *File Operations* – The number of sequential or random, create or delete file operations per second illustrative of the overheads associated with manipulating file metadata in a file system; *Read File Operations* – The number of sequential or random read operations of file metadata in zero size files per second an indicator of the efficiency of a file system's structure.

4.3 Methodology

The image propagation experiment alters the image size, in the range 3GB to 21GB, in increments of 3GB and records time in seconds. It is difficult to gauge what sizes of image are representative of those used in the real world as the image size is often dependent on the application. We in part have chosen this range as it is representative of a Service Oriented Architecture pathology application we have deployed on our testbed that requires large volumes of data be embedded into the VM image. In addition, the chosen range should provide enough data points for any relationship between the variables to be observed. For the IOzone and Bonnie++ experiments, the software stacks on which the benchmarks run are altered. The software stacks are the native host, KVM hypervisor deployed on the native host, XEN Domain0 or privileged guest that facilitates XEN VMs and XEN DomainU or XEN guest VM. In addition, the newest stable releases of KVM and XEN are also introduced. Depending on which benchmark test is being performed, either the throughput in bytes per second or the number of operations per second is recorded. The percentage CPU time, where applicable in the case of Bonnie++ tests, is also recorded. Ten iterations of each of the benchmark tests are performed and the average result, along with the standard deviation, are presented where possible.



(a)

(b)

Figure 3: Image propagation: Trends.

5 EXPERIMENTAL RESULTS

This section dicusses the results of each of the three benchmarks outlined in the Section 4.2.

5.1 Image Propagation Performance Analysis

The results of the image propagation benchmark provide an interesting insight into how the underlying design and implementation of the image management supervision processes used in each IaaS manager, impact the time it takes for a single iteration of a VM's lifecycle. Figure 3(a) presents these times against an increasing VM image size.

It is clear that Nimbus using GSIFTP as a transfer protocol outperforms both Nimbus and OpenNebula using SCP. Adding appropriate trend lines, interpolating to an image size of zero and reading the appropriate value from the y-axis intercept, provides an indication of the overhead created by the implementation within each IaaS manager, responsible for managing the propagation process of a VM. Figure 3(b) illustrates the theoretical performance of the NIC which neither protocol obtains. It also show that on average GSIFTP transfers data nearly twice as fast or in ∼46% less time than SCP and when using the SCP protocol the transfer time is agnostic of the IaaS manager used.

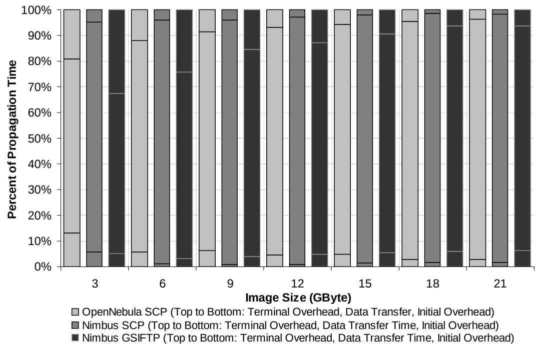


Figure 4: Image propagation: Overheads.

Figure 4 show the results of the time spent waiting to change from one state to another in each of the platforms tested. It can be seen that the majority of the time is spent in a transferring state, waiting for the transfer protocol to complete the movement of data from the image repository to the VM host machine. On the other hand, when transferring smaller images the initial and terminal overheads of the IaaS manager supervising the propagation process become an increased proportion of the total time spent waiting. When comparing the SCP transfer protocol on different IaaS managers Nimbus can be seen to outperform OpenNebula with a considerable reduction in the time spent in the initialization and termination states, specifically ∼3.5 times faster or in ∼72% less time, creating less overhead for identical sizes of image. When Nimbus utilizes the GSIFTP transfer protocol the initialization time remains roughly proportional to the overall propagation time. This is in contrast to the other platforms where the percentage of time spent in the initialization and termination states decrease proportionally with an increase in image size, that is the overheads remain constant. This feature is also illustrated by the polynomial trend line in Figure 3(a).

5.2 Virtual Block I/O Performance Analysis

Block I/O devices are incredibly slow in comparison to the performance of memory and CPU caches in traditional computer systems, with many millions of CPU cycles being wasted to service a single I/O requests when a cache miss occurs. Optimization of the virtual equivalent block I/O, like any leading bottleneck within a system, should be of a high priority and has repercussions for applications that utilize large datasets. The results in this subsection on the benchmarks of IOzone and Bonnie++, divulge the performance of the VMM block I/O devices and thus reveal the state of development, the amount of effort and time assigned to optimization and an indication of the maturity of the Virtualization solutions KVM and XEN.

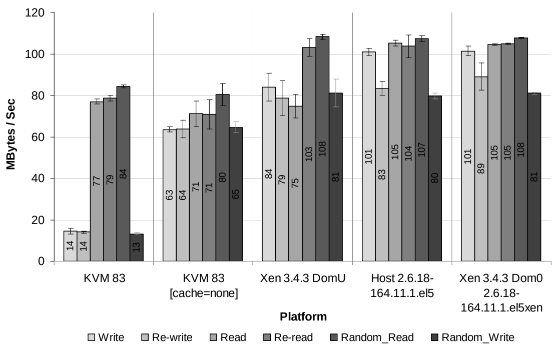


Figure 5: IOzone - record size: 16MB, file: 4GB guest; 8GB host.

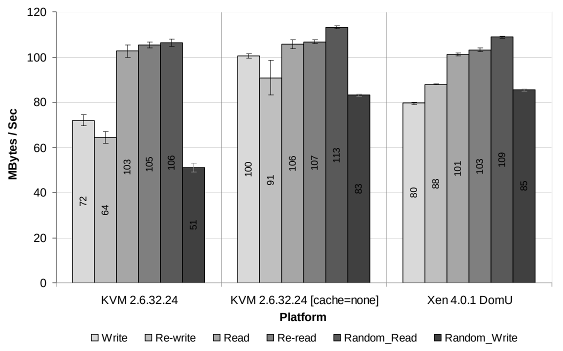


Figure 6: IOzone on 2.6.32.24 - record size: 16MB, file: 4GB guest.

Figure 5 and 6 illustrates the maximum throughput of the virtual block I/O devices for a given platform, where a record size of 16Mbyte is used to minimize CPU usage. It can be seen that out of the box, KVM 83 write performance is incredibly poor in contrast to XEN 3.4.3. KVM 83 on average across all write tests exhibited ∼17% of the throughput of XEN 3.4.3. IOzone read tests demonstrate that KVM 83 performs similarly to XEN 3.4.3 in the initial read test but failed to equal the throughput for the re-read and random read tests attaining ∼77% of the throughput. The tests on the newer versions of KVM and XEN are shown in Figure 6. KVM 2.6.32.24 again shows poor write performance but is a vast improvement over the older version by roughly a factor of 3. The performance of XEN 4.0.1 is on par with the older version of XEN 3.4.3.

After further investigation, the bottleneck was tracked to the caching system of the QEMU back-end used by KVM. By default a write-through cache is deployed for consistency to guarantee storage is committed to the underlying physical device. This has the effect of increasing the amount of bus traffic and additional copy operations needed and consequently reduces the performance of write operations. With this in mind, the benchmarks for KVM were rerun avoiding the use of the cache all together and this demonstrated far superior performance. This time KVM displayed ∼79% of the throughput of XEN. Figure 6 shows the new version of KVM 2.6.32.24 performing on par with and on occasion better than XEN 4.0.1.

Comparing XEN 3.4.3 with the native host platform and accounting for variance, write throughput was on par on all tests other than the initial write test which revealed that XEN 3.4.3 exhibited ∼83% of the throughput. The results of the XEN 3.4.3 initial read test demonstrated similar results with ∼71% of the throughput of the host. The other XEN 3.4.3 read tests were indistinguishable from the host and the performance of the privileged guest "Domain0"

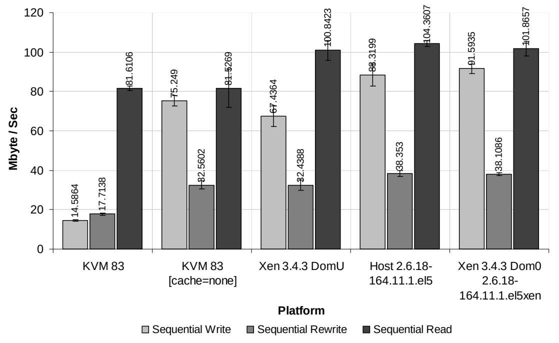


Figure 7: Bonnie++ - Throughput MB/Sec.

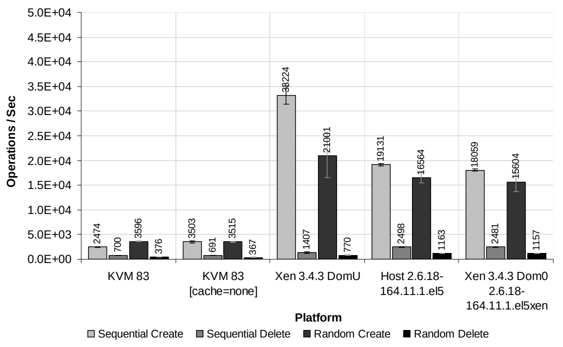


Figure 9: Bonnie++ - File operations per second.

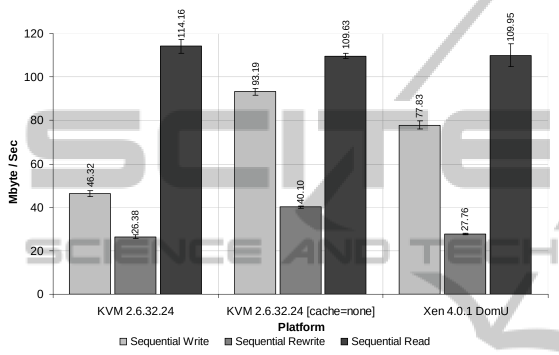


Figure 8: Bonnie++ on 2.6.32.24 - Throughput MB/Sec.

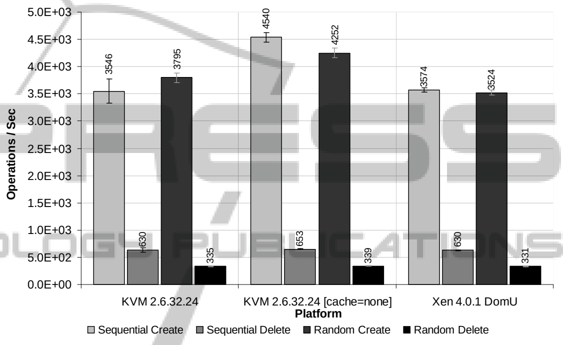


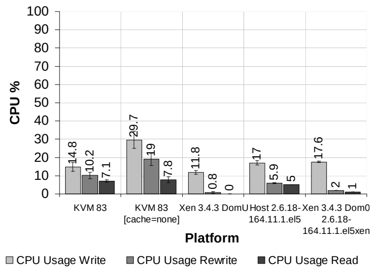
Figure 10: Bonnie++ on 2.6.32.24 - File operations per second.

was equivalent to the native host performance across all tests.

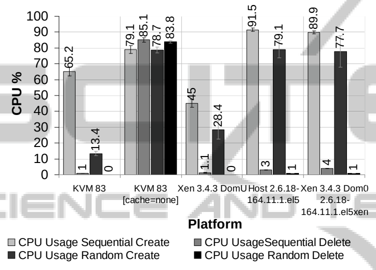
The results of the Bonnie++ benchmark for sequential throughput (Figure 7 and 8) are confirmatory, showing near identical results to IOzone for the sequential write and read tests but oddly show differing results for the sequential re-write test across all platforms. These results show roughly a 50% reduction in throughput compared to IOzone. This could indicate that the sequential re-write test throughput is a formulation of the time to execute two consecutive sequential write operations and has not been accounted for. The discrepancy highlights the need for an additional confirmatory benchmark when running performance evaluation experiments.

While throughput of the virtual block devices provides an indication of performance, the number of operations that can be performed per second give further insight into how efficient the implementations are. Figure 9 and 10 presents the Bonnie++ benchmark results for sequential and random deletion and creation of files respectively. The results show that KVM 83 performs poorly in all cases, more so in the create tests where an order of magnitude less operations can be executed and where turning off the storage cache in KVM has minimal impact on performance. Interestingly XEN 3.4.3 guests outperform the native host. Again it can be seen that the performance of KVM 2.6.32.24 is comparable to XEN 3.4.3 and 4.0.1.

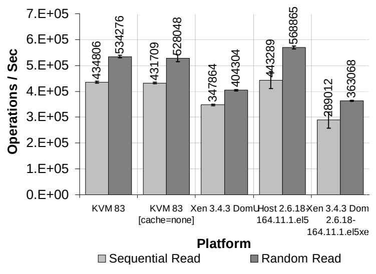
Another indicator of efficient design and implementation is that of CPU usage used when performing I/O related tasks. Applications using CPU resources can become starved by OS subsystems such as block I/O back-ends using excessive CPU time to service requests for accessing I/O devices. Figure 11(a) shows KVM 83 performing poorly on all sequential operations using a greater percentage of CPU to service less requests in comparison to XEN. Interestingly the XEN 3.4.3 guest performs better than the underlying XEN Domain0, exhibiting very little CPU usage if the CPU metric gathered within the XEN guest is to be trusted. Disabling the storage cache of KVM 83 creates an increase in the percentage of CPU time used. In the file operation tests of Figure 11(b) this performance gap is even more prominent with all tests exhibiting CPU usage around 80%. At first glance KVM with storage cache seems to outperform XEN in the percentage of CPU time used to create files randomly but performs far less operations per second. Figure 12(a) demonstrates another performance improvement for KVM 2.6.32.24 over KVM 83 and a slight performance regression from XEN 3.4.3 to 4.0.1. Figure 12(b) reveals a large reduction in CPU usage across the tests for KVM 2.6.32.24 with and without cache and a modest improvement for XEN 4.0.1 over XEN 3.4.3.



(a)



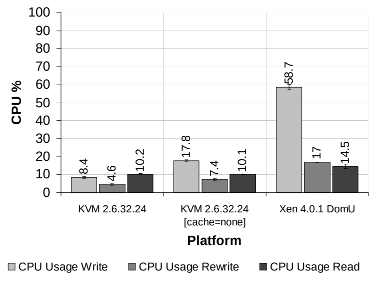
(b)



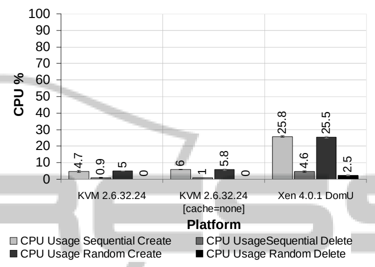
(c)

Figure 11: Bonnie++ - Left-to-Right, Top-to-Bottom: Sequential Throughput CPU Usage, File Operations CPU Usage & Read File Operations.

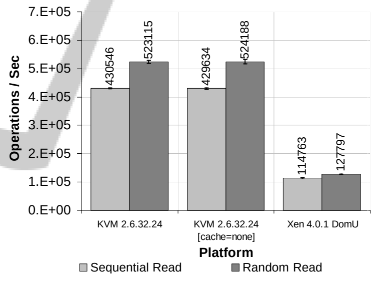
Figure 11(c) presents evidence that KVM 83 guests outperform XEN 3.4.3 guests and that the XEN 3.4.3 privileged guest Domain0 performs worse than the native host when operating on file metadata. This indicates that the cause of the write bottleneck of KVM 83 with storage cache is due to the inefficient manner in which data is transferred from the host physical storage to the guest and not due to the handling of file system metadata. Finally comparing Figure 11(c) and Figure 12(c) another performance regression is illustrated from XEN 3.4.3 to 4.0.1.



(a)



(b)



(c)

Figure 12: Bonnie++ on 2.6.32.24 - Left-to-Right, Top-to-Bottom: Sequential Throughput CPU Usage, File Operations CPU Usage & Read File Operations.

6 RELATED WORK

The main contribution of this paper lies in the findings of a performance evaluation into a specific aspect of Cloud Computing; the management and use of data at the Virtualisation level. Results are presented that provide insight into how Cloud technology can be improved and what technology is best for the needs of a given application. There have been numerous studies on the performance of hypervisors within the literature (Kesavan et al., 2010; Padala et al., 2008; Stantchev, 2009; Yu Liang and Lu, 2008; Xianghua Xu et al., 2008; Jianhua Che et al., 2008; Deshane et al., 2008) and this paper differentiates from these by giving a contemporary performance evaluation within the context of Cloud Computing. We contribute and implement a novel benchmark to evaluate end to end image propagation overheads of any IaaS. (Yigitbasi et al., 2009) present a performance analysis framework for the overheads in resource acquisition and release. However the results of their work are for resources deployed onto Amazon EC2 with the framework left untested on other virtual resource management architectures. (Goiri et al., 2009) describe a proof-of-concept framework for facilitating resource management in service providers in which a performance evaluation of the overhead of creating a VM image is considered not the time to propagate an already existing VM image to a physical resource. (Chen et al., 2009) explore how multicasting file transfers can reduce image propagation overheads.

Previous related works have concentrated on the performance and scalability of CPU virtualisation omitting a performance analysis inclusive of I/O (Padala et al., 2008). An additional publication (Stantchev, 2009) formulates an approach to performance evaluation, using real world workloads on Amazon EC2 and considers the system as a "black box" without finding the root causes of the performance bottlenecks. Another creates a methodology for the collection of virtual I/O metrics (Yu Liang and Lu, 2008) but evaluates the performance of only a single hypervisor. Other publications (Xianghua Xu et al., 2008; Jianhua Che et al., 2008) incorrectly setup a parameter of the IOzone benchmark using a 64MB test file instead of twice the size of available memory, bringing into question the validity of the results obtained. (Deshane et al., 2008) present results that KVM outperforms XEN in IOzone tests. Our experiments have been unable to confirm these results and one can only assume, due to limited information from the paper on the experimental environment used, that again the file size parameter for the IOzone benchmark has been set incorrectly.

7 CONCLUSIONS AND FUTURE WORK

The aspect of performance within QoS has an important role to play in the provisioning of resources in Cloud Computing. This paper presented the findings of a performance evaluation into the management and utilization of data inside an IaaS provider. The implementation of technologies used in virtual infrastructure has been highlighted as influencing the outcome of resource performance and consequently the QoS provisioned to end users, the competitiveness of a provider in the Cloud ecosystem and likely return on investment of services made available. This subsequently makes the selection of technology a decisive decision for any Cloud provider. The outcome of our work provides quantitative evidence that can enhance this decision making process and aid in the prevention of SLAs breaches.

The results of our experiments illustrate that OpenNebula and KVM, relative new comers to the paradigm of distributed systems, perform to a lower standard than Nimbus and XEN. This is being rectified over time as seen with KVM version 2.6.32.24 but, like XEN 4.0.1, is not currently distributed with enterprise grade operating systems such as CentOS 5.4. A general theme has reoccurred throughout our performance analysis: the maturity of a particular technology heavily influences performance. Therefore it could be concluded that the findings of our work advocate mature software solutions due to correlation with improved relative performance, but this is not always the case as seen in XEN version 4.0.1 where a new implementation of the blktap disk backend driver in a mature software solution has introduced performance regression. This puts into context the contemporary feature sets these new technologies provide, which we argue are more appropriate or specific to the usage scenarios of a Cloud environment. As a consequence this trade off between performance and feature set should be factored in when making any decision on whether to use the technology evaluated here in.

The implications of our results draw attention to the impact performance overheads have on the adoption of Cloud technology. OpenNebula for example has a comparatively limited aptitude to react to changes in demand in a stochastic and highly dynamic environment and accordingly Nimbus would be more appropriate in this scenario. Data management services such as Amazon's S3, when considered not economically viable or where the cost of these Cloud services are not completely transparent and vary significantly (Kossmann et al., 2010), could result in data having to be stored and accessed within a local VM image. The lack of KVM 83 guests at writing and reading data could be a limiting factor for applications that access large quantities of data locally and conversely XEN 3.4.1 would be an advisable choice here.

Future work is planned to evaluate the trade off between the features of enhanced image formats such as QCOW2, VMware's VMDK and Microsoft's VHD and the performance of raw images. In addition, the overheads surrounding other popular VIMs, such as Eucalyptus (Eucalyptus, 2010), will be evaluated.

REFERENCES

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164 – 177.

Bonnie++ (2010). Bonnie++ - Benchmark Suite. Website. http://www.coker.com.au/bonnie++/.

BWM-NG (2010). Bandwidth Monitor NG. Website. http://www.gropp.org/?id=projects&sub=bwm-ng.

Chen, Y., Wo, T., and Li, J. (2009). An Efficient Resource Management System for On-line Virtual Cluster Provision. In *IEEE 2009 International Conference on Cloud Computing*.

Deshane, T., Shepherd, Z., Matthews, J., Ben Yehuda, M., Shah, A., and Rao, B. (2008). Quantitative comparison of Xen and KVM. In *Xen Summit*, Berkeley, CA, USA. USENIX Association.

Dong, Y., Dai, J., Huang, Z., Guan, H., Tian, K., and Jiang, Y. (2009). Towards high-quality I/O virtualization. In *ACM International Conference Proceeding Series*, pages 12 – 22, Haifa, Israel.

Dong-Jae Kang, Chei-Yol Kim, Kang-Ho Kim, and Sung-In Jung (2008). Proportional disk I/O bandwidth management for server virtualization environment. In *2008 International Conference on Computer Science and Information Technology*, pages 647 – 53, Piscataway, NJ, USA.

Eucalyptus (2010). Eucalyptus - Elastic Utility Computing Architecture. Website. http://www.eucalyptus.com/.

Goiri, I., Julia, F., Ejarque, J., de Palol, M., Badia, R., Guitart, J., and Torres, J. (2009). Introducing virtual execution environments for application lifecycle management and SLA-driven resource distribution within service providers. In *Proceedings 2009 Eighth IEEE International Symposium on Network Computing and Applications (NCA)*, pages 211 – 18, Piscataway, NJ, USA.

Goldberg, R. (1972). *Architectural Principles for Virtual Computer Systems*. PhD thesis, Harvard University, Cambridge, MA.

Hansen, J. G. and Jul, E. (2010). Lithium: virtual machine storage for the cloud. In *SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing*, pages 15 – 26, New York, NY, USA. ACM.

Imada, T., Sato, M., and Kimura, H. (2009). Power and QoS performance characteristics of virtualized servers. In *Proceedings of the 2009 10th IEEE/ACM International Conference on Grid Computing (GRID)*, pages 232 – 40, Piscataway, NJ, USA.

IOzone (2010). IOzone - Filesystem Benchmark. Website. http://www.iozone.org/.

Jianhua Che, Qinming He, Qinghua Gao, and Dawei Huang (2008). Performance measuring and comparing of virtual machine monitors. In *2008 IEEE/IFIP 5th International Conference on Embedded and Ubiquitous Computing. EUC 2008*, volume 2, pages 381 – 6, Piscataway, NJ, USA.

Jun Zhu, Wei Dong, Zhefu Jiang, Xiaogang Shi, Zhen Xiao, and Xiaoming Li (2010). Improving the Performance of Hypervisor-Based Fault Tolerance. In *Proceedings of the 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, pages 10 – 20, Piscataway, NJ, USA.

Kate Keahey, Tim Freeman, Jerome Lauret, and Doug Olson (2007). Virtual workspaces for scientific applications. In *SciDAC 2007 Conference*, Boston, MA.

Keahey, K. and Freeman, T. (2008). Contextualization: Providing One-Click Virtual Clusters. In *ESCIENCE '08: Proceedings of the 2008 Fourth IEEE International Conference on eScience*, pages 301 – 308, Washington, DC, USA. IEEE Computer Society.

Kesavan, M., Gavrilovska, A., and Schwan, K. (2010). Differential virtual time (DVT): rethinking I/O service differentiation for virtual machines. In *SoCC '10: Proceedings of the 1st ACM symposium on Cloud computing*, pages 27 – 38, New York, NY, USA. ACM.

Kossmann, D., Kraska, T., and Loesing, S. (2010). An evaluation of alternative architectures for transaction processing in the cloud. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*, pages 579 – 590, New York, NY, USA. ACM.

KVM (2010). KVM - Kernel Based Virtual Machine. Website. http://www.linux-kvm.org.

Nakajima, J. and Mallick Asit K. (2007). Hybrid Virtualization - Enhanced Virtualization for Linux. In *Proceedings of the Linux Symposium*.

Padala, P., Zhu, X., Wang, Z., Singhal, S., and Shin, K. G. (2008). Performance Evaluation of Virtualization Technologies for Server Consolidation. Technical report, HP Labs.

Popek, G. and Goldberg, R. (1974). Formal requirements for virtualizable third generation architectures. *Communications of the ACM*, 17(7):412 – 21.

QEMU (2010). QEMU - Open Source Processor Emulation. Website. http://www.qemu.org.

Rochwerger, B., Caceres, J., Montero, R., Breitgand, D., Elmroth, E., Galis, A., Levy, E., Llorente, I., Nagin, K., and Wolfstha, Y. (2009). The Reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development*, 53(4):Online.

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

Sotomayor, B., Rubé andn, Llorente, I. M., and Foster, I. (2009). Virtual Infrastructure Management in Private and Hybrid Clouds. *Internet Computing, IEEE*, 13(5):14 – 22.

Stantchev, V. (2009). Performance evaluation of cloud computing offerings. In *Proceedings of the 2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2009)*, pages 187 – 92, Piscataway, NJ, USA.

Van, H. N., Tran, F. D., and Menaud, J.-M. (2009). SLA-aware virtual resource management for cloud infrastructures. In *Proceedings - IEEE 9th International*

Conference on Computer and Information Technology, CIT 2009, volume 1, pages 357 – 362, Xiamen, China.

Vouk, M. (2008). Cloud computing - Issues, research and implementations. In *2008 30th International Conference on Information Technology Interfaces (ITI)*, pages 31 – 40, Piscataway, NJ, USA.

Xianghua Xu, Feng Zhou, Jian Wan, and Yucheng Jiang (2008). Quantifying performance properties of virtual machine. In *Linux;program testing;software performance evaluation;systems analysis;virtual machines;*, volume 1, pages 24 – 8, Piscataway, NJ, USA.

Xiong, K. and Perros, H. (2009). Service performance and analysis in cloud computing. In *SERVICES 2009 - 5th 2009 World Congress on Services*, pages 693 – 700, Bangalore, India.

Xu, C., Bai, Y., and Luo, C. (2009). Performance evaluation of parallel programming in virtual machine environment. In *NPC 2009 - 6th International Conference on Network and Parallel Computing*, pages 140 – 147, Gold Coast, QLD, Australia.

Yigitbasi, N., Iosup, A., Epema, D., and Ostermann, S. (2009). C-Meter: A framework for performance analysis of computing clouds. In *9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID 2009*, pages 472 – 477, Shanghai, China.

Yu Liang, S. and Lu, X. (2008). An efficient disk I/O characteristics collection method based on virtual machine technology. In *Proceedings - 10th IEEE International Conference on High Performance Computing and Communications, HPCC 2008*, pages 943 – 949, Dalian, China.