ARCHITECTURAL PRINCIPALS OF DISTRIBUTED HIGH PERFORMANCE VIRTUALIZATION

by

Andrew J. Younge

Submitted to the faculty of

Indiana University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Indiana University

Jan 2015

INDIANA UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a dissertation submitted by

Andrew J. Younge

This dissertation has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Geoffrey C. Fox, Ph.D, Chair
Date	Judy Qiu, Ph.D
Date	Thomas Sterling, Ph.D
Date	D. Martin Swany, Ph.D

INDIANA UNIVERSITY

As chair of the candidate's graduate committee, I have read the dissertation of Andrew J. Younge in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Geoffrey C. Fox, Ph.D
Chair, Graduate Committee

Accepted for the Department

Department Chair Name, Chair
Computer Science Program

Accepted for the College

Dean Name, Associate Dean
School of Informatics and Computing

ABSTRACT

ARCHITECTURAL PRINCIPALS OF DISTRIBUTED

HIGH PERFORMANCE VIRTUALIZATION

Andrew J. Younge

Department of Computer Science

Doctor of Philosophy

TODO: Compress 250 pages and N years of your life down to a few short paragraphs here.

Scientific computing endeavors have created clusters, grids, and supercomputers as HPC platforms and paradigms. These resources focus on peak performance and computing efficiency, thereby enabling scientific community to tackle non-trivial problems on massively parallel architectures. Meanwhile, efforts to leverage the economies of scale from data center operations and advances in virtualization technologies have created large scale Cloud Infrastrictire. Such Infrastructure-as-a-Service (IaaS) deployments provide mechanisms for handling millions of user interactions concurrently or organizing, cataloging, and retrieving data by allowing users to specify a custom computing environment tailored to their needs. Combining concepts from both supercomputing and clouds will enable users to leverage the performance of

HPC applications with the ease and availably in clouds.

This work proposes to bridge the gap between supercomputing and clouds using a few key aspects. First, we evaluate current hypervisors and their viability to run HPC workloads within current infrastructure. Next, we illustrate a mechanism to enable advanced accelerators such as GPUs in a Virtual Machine that can significantly enhance scientific computing problems. Furthermore, we are also able to support high speed, low latency inter-node communication through the use of InfiniBand within virtual machines. Upon evaluating these newfound features and leveraging the system within the OpenStack environment, we illustrate that virtualzied cloud cyberinfrastructure perform at near-native speeds and support a broad range of scientific computing problems as never before.

ACKNOWLEDGMENTS

TODO: Write acknowledgements section

Thanks, Mom!

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

For years visionaries in computer science have predicted the advent of utility-based computing. This concept dates back to John McCarthy's vision stated at the MIT centennial celebrations in 1961.

> "If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new and important industry."

Only recently has the hardware and software become available to support the concept of utility computing on a large scale.

1.2 Motivation

TBD

Chapter 2

Related Research

In order to accurately depict the research presented in this article, the topics within Cloud computing are reviewed

2.1 Cloud Computing

Cloud computing is one of the most explosively expanding technologies in the computing industry today. However it is important to understand where it came from, in order to figure out where it will be heading in the future. While there is no clear cut evolutionary path to Clouds, many believe the concepts originate from two specific areas: Grid Computing and Web 2.0.

Grid computing [1, 2], in its practical form, represents the concept of connecting two or more spatially and administratively diverse clusters or supercomputers together in a federating manner. The term "the Grid" was coined in the mid 1990's to represent a large distributed systems infrastructure for advanced scientific and engineering computing problems. Grids aim to enable applications to harness the full potential of resources through coordinated and controlled resource sharing by scalable

virtual organizations. While not all of these concepts carry over to the Cloud, the control, federation, and dynamic sharing of resources is conceptually the same as in the Grid. This is outlined by [3], as Grids and Clouds are compared at an abstract level and many concepts are remarkably similar. From a scientific perspective, the goals of Clouds and Grids are also similar. Both systems attempt to provide large amounts of computing power by leveraging a multitude of sites running diverse applications concurrently in symphony. The only significant differences between Grids and Clouds exist in the implementation details, and the reproductions of them, as outlined later in this section.

The other major component, Web 2.0, is also a relatively new concept in the history of Computer Science. The term Web 2.0 was originally coined in 1999 in a futuristic prediction by Dracy DiNucci [4]: "The Web we know now, which loads into a browser window in essentially static screenfulls, is only an embryo of the Web to come. The first glimmerings of Web 2.0 are beginning to appear, and we are just starting to see how that embryo might develop. The Web will be understood not as screenfuls of text and graphics but as a transport mechanism, the ether through which interactivity happens. It will [...] appear on your computer screen, [...] on your TV set [...] your car dashboard [...] your cell phone [...] hand-held game machines [...] maybe even your microwave oven." Her vision began to form, as illustrated in 2004 by the O'Riley Web 2.0 conference, and since then the term has been a pivotal buzz word among the internet. While many definitions have been provided, Web 2.0 really represents the transition from static HTML to harnessing the Internet and the Web as a platform in of itself.

Web 2.0 provides multiple levels of application services to users across the Internet. In essence, the web becomes an application suite for users. Data is outsourced to wherever it is wanted, and the users have total control over what they interact

with, and spread accordingly. This requires extensive, dynamic and scalable hosting resources for these applications. This demand provides the user-base for much of the commercial Cloud computing industry today. Web 2.0 software requires abstracted resources to be allocated and relinquished on the fly, depending on the Web's traffic and service usage at each site. Furthermore, Web 2.0 brought Web Services standards [5] and the Service Oriented Architecture (SOA) [6] which outline the interaction between users and cyberinfrastructure. In summary, Web 2.0 defined the interaction standards and user base, and Grid computing defined the underlying infrastructure capabilities.

A Cloud computing implementation typically enables users to migrate their data and computation to a remote location with minimal impact on system performance [**?**]. This provides a number of benefits which could not otherwise be realized. These benefits include:

- *Scalable* - Clouds are designed to deliver as much computing power as any user needs. While in practice the underlying infrastructure is not infinite, the cloud resources are projected to ease the developer's dependence on any specific hardware.

- *Quality of Service (QoS)* - Unlike standard data centers and advanced computing resources, a well-designed Cloud can project a much higher QoS than traditionally possible. This is due to the lack of dependence on specific hardware, so any physical machine failures can be mitigated without the prerequisite user awareness.

- *Specialized Environment* - Within a Cloud, the user can utilize customized tools and services to meet their needs. This can be to utilize the latest library, toolkit, or to support legacy code within new infrastructure.

- *Cost Effective* - Users finds only the hardware required for each project. This reduces the risk for institutions potentially want build a scalable system, thus providing greater flexibility, since the user is only paying for needed infrastructure while maintaining the option to increase services as needed in the future.

- *Simplified Interface* - Whether using a specific application, a set of tools or Web services, Clouds provide access to a potentially vast amount of computing resources in an easy and user-centric way. We have investigated such an interface within Grid systems through the use of the Cyberaide project [**?**, 7].

Many of the features noted above define what Cloud computing can be from a user perspective. However, Cloud computing in its physical form has many different meanings and forms. Since Clouds are defined by the services they provide and not by applications, an integrated as-a-service paradigm has been defined to illustrate the various levels within a typical Cloud, as in Figure 2.1.

- *Clients* - A client interacts with a Cloud through a predefined, thin layer of abstraction. This layer is responsible for communicating the user requests and displaying data returned in a way that is simple and intuitive for the user. Examples include a Web Browser or a thin client application.

- *Software-as-a-Service (SaaS)* - A framework for providing applications or software deployed on the Internet packaged as a unique service for users to consume. By doing so, the burden of running a local application directly on the client's machine is removed. Instead all the application logic and data is managed centrally and to the user through a browser or thin client. Examples include Google Docs, Facebook, or Pandora.

- *Platform-as-a-Service (PaaS)* - A framework for providing a unique computing

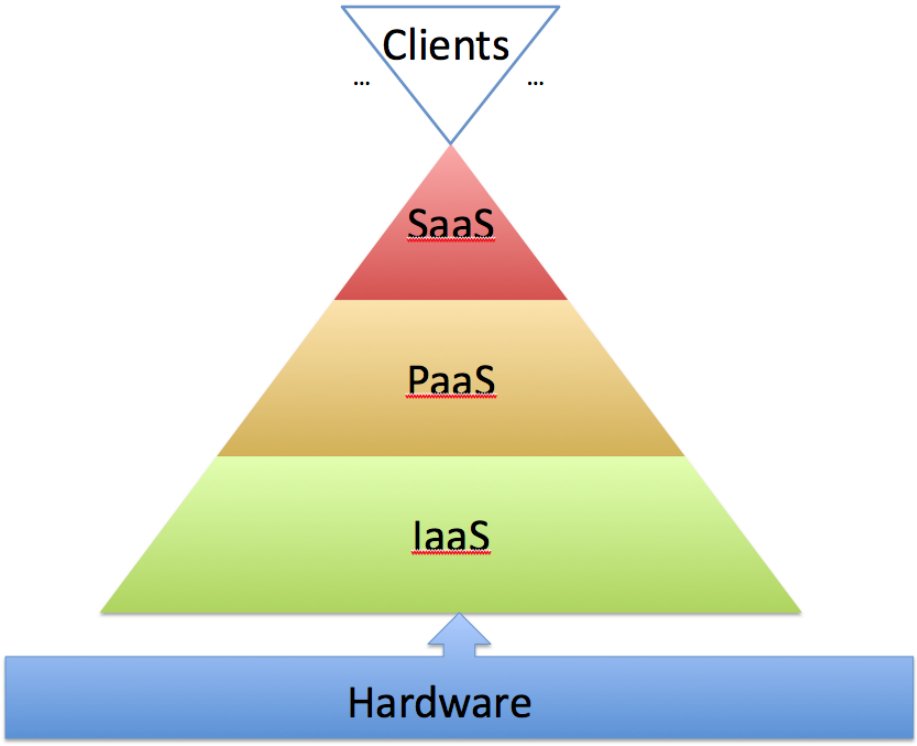

Figure 2.1 View of the Layers within a Cloud Infrastructure

platform or software stack for applications and services to be developed on. The goal of PaaS is to alleviate many of the burdens of developing complex, scalable software by proving a programming paradigm and tools that make service development and integration a tractable task for many. Examples include Microsoft Azure and Google App Engine.

- *Infrastructure-as-a-Service (IaaS)* - A framework for providing entire computing resources through a service. This typically represents virtualized Operating Systems, thereby masking the underlying complexity details of the physical infrastructure. This allows users to rent or buy computing resources on demand for their own use without needing to operate or manage physical infrastructure. Examples include Amazon EC2, Eucalyptus, and Nimbus.

- *Physical Hardware* - The underlying set of physical machines and IT equipment

that host the various levels of service. These are typically managed at a large scale using virtualization technologies which provide the QoS users expect. This is the basis for all computing infrastructure.

When all of these layers are combined, a dynamic software stack is created to focus on large scale deployment of services to users.

2.1.1 Virtualization

There are a number of underlying technologies, services, and infrastructure-level configurations that make Cloud computing possible. One of the most important technologies is the use of virtualization [8, 9]. Virtualization is a way to abstract the hardware and system resources from a operating system. This is typically performed within a Cloud environment across a large set of servers using a Hypervisor or Virtual Machine Monitor (VMM) which lies in between the hardware and the Operating System (OS). From here, one or more virtualized OSs can be started concurrently as seen in Figure 2.2, leading to one of the key advantages of Cloud computing. This, along with the advent of multi-core processing capabilities, allows for a consolidation of resources within any data center. It is the Cloud's job to exploit this capability to its maximum potential while still maintaining a given QoS.

Virtualization is not specific to Cloud computing. IBM originally pioneered the concept in the 1960's with the M44/44X systems. It has only recently been reintroduced for general use on x86 platforms. Today there are a number of Clouds that offer IaaS. The Amazon Elastic Compute Cloud (EC2) [10], is probably the most popular of which and is used extensively in the IT industry. Eucalyptus [11] is becoming popular in both the scientific and industry communities. It provides the same interface as EC2 and allows users to build an EC2-like cloud using their own inter-

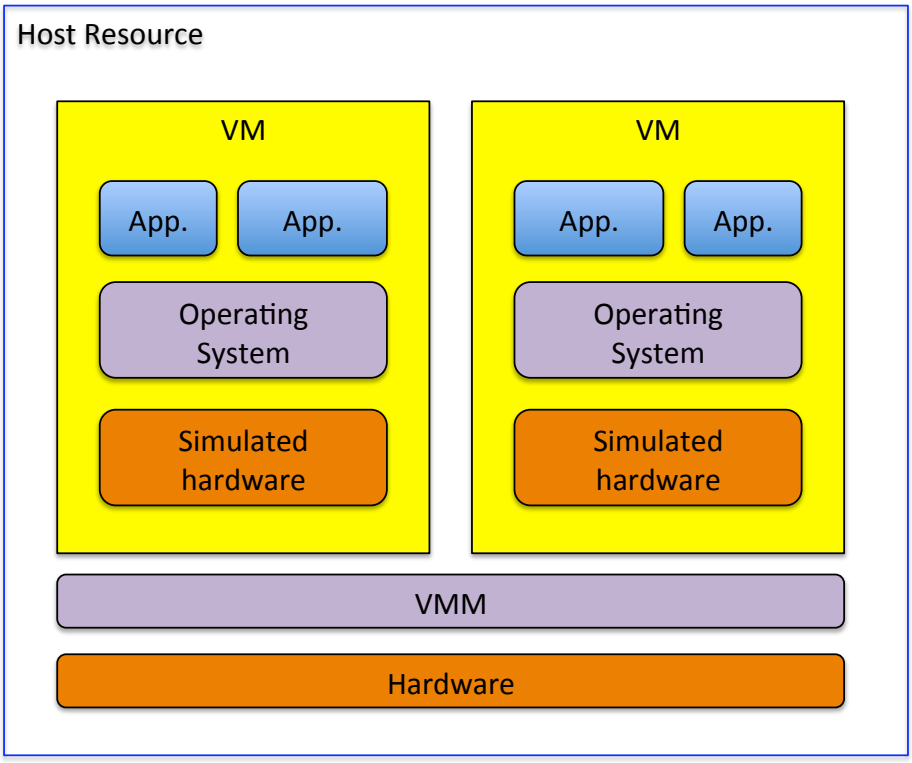

Figure 2.2 Virtual Machine Abstraction

nal resources. Other scientific Cloud specific projects exist such as OpenNebula [12], In-VIGO [**?**], and Cluster-on-Demand [13]. They provide their own interpretation of private Cloud services within a data center. Using a Cloud deployment overlaid on a Grid computing system has been explored by the Nimbus project [14] with the Globus Toolkit [15]. All of these clouds leverage the power of virtualization to create an enhanced data center. The virtualization technique of choice for these Open platforms has typically been the Xen hypervisor, however more recently VMWare and the Kernel-based Virtual Machine (KVM) have become commonplace.

2.1.2 Workload Scheduling

While virtualization provides many key advancements, this technology alone is not sufficient. Rather, a collective scheduling and management for virtual machines is

required to piece together a working Cloud. Let us consider a typical usage for a Cloud data center that is used in part to provide computational power for the Large Hadron Collider at CERN [16], a global collaboration from more than 2000 scientists of 182 institutes in 38 nations. Such a system would have a small number of experiments to run. Each experiment would require a very large number of jobs to complete the computation needed for the analysis. Examples of such experiments are the ATLAS [17] and CMS [18] projects, which (combined) require Petaflops of computing power on a daily basis. Each job of an experiment is unique, but the application runs are often the same.

Therefore, virtual machines are deployed to execute incoming jobs. There is a file server which provides virtual machine templates. All typical jobs are preconfigured in virtual machine templates. When a job arrives at the head node of the cluster, a correspondent virtual machine is dynamically started on acertain compute node within the cluster to execute the job (see Figure BLAH).

While this is an abstract solution, it is important to keep in mind that these virtual machines create an overhead when compared to running on "bare metal." Current research estimates this the overhead for CPU bound operations at 1 to 15% depending on the hypervisor, however more detailed studies are needed to better understand this overhead. While the hypervisor introduces overhead, so does the actual VM image being used. Therefore, it is clear that slimming down the images could yield an increase in overall system efficiency. This provides the motivation for the minimal Virtual Machine image design discussed in Section **??**.

Chapter 3

Analysis of Virtualization Technologies for High Performance Computing Environments

3.1 Abstract

As Cloud computing emerges as a dominant paradigm in distributed systems, it is important to fully understand the underlying technologies that make Clouds possible. One technology, and perhaps the most important, is virtualization. Recently virtualization, through the use of hypervisors, has become widely used and well understood by many. However, there are a large spread of different hypervisors, each with their own advantages and disadvantages. This manuscript provides an in-depth analysis of some of today's commonly accepted virtualization technologies from feature comparison to performance analysis, focusing on the applicability to High Performance Computing environments using FutureGrid resources. The results indicate virtualization sometimes introduces slight performance impacts depending on the hypervisor

type, however the benefits of such technologies are profound and not all virtualization technologies are equal.

3.2 Introduction

Cloud computing [19] is one of the most explosively expanding technologies in the computing industry today. A Cloud computing implementation typically enables users to migrate their data and computation to a remote location with some varying impact on system performance [**?**]. This provides a number of benefits which could not otherwise be achieved.

Such benefits include:

- *Scalability* - Clouds are designed to deliver as much computing power as any user needs. While in practice the underlying infrastructure is not infinite, the cloud resources are projected to ease the developer's dependence on any specific hardware.

- *Quality of Service (QoS)* - Unlike standard data centers and advanced computing resources, a well-designed Cloud can project a much higher QoS than traditionally possible. This is due to the lack of dependence on specific hardware, so any physical machine failures can be mitigated without the prerequisite user awareness.

- *Customization* - Within a Cloud, the user can utilize customized tools and services to meet their needs. This can be to utilize the latest library, toolkit, or to support legacy code within new infrastructure.

- *Cost Effectiveness* - Users finds only the hardware required for each project. This reduces the risk for institutions potentially want build a scalable system,

thus providing greater flexibility, since the user is only paying for needed infrastructure while maintaining the option to increase services as needed in the future.

- *Simplified Access Interfaces* - Whether using a specific application, a set of tools or Web services, Clouds provide access to a potentially vast amount of computing resources in an easy and user-centric way.

While Cloud computing has been driven from the start predominantly by the industry through Amazon [20], Google [**?**] and Microsoft [**?**], a shift is also occurring within the academic setting as well. Due to the many benefits, Cloud computing is becoming immersed in the area of High Performance Computing (HPC), specifically with the deployment of scientific clouds [**?**] and virtualized clusters [**?**].

There are a number of underlying technologies, services, and infrastructure-level configurations that make Cloud computing possible. One of the most important technologies is virtualization. Virtualization, in its simplest form, is a mechanism to abstract the hardware and system resources from a given Operating System. This is typically performed within a Cloud environment across a large set of servers using a Hypervisor or Virtual Machine Monitor (VMM), which lies in between the hardware and the OS. From the hypervisor, one or more virtualized OSs can be started concurrently as seen in Figure 2.2, leading to one of the key advantages of Cloud computing. This, along with the advent of multi-core processors, allows for a consolidation of resources within any data center. From the hypervisor level, Cloud computing middleware is deployed atop the virtualization technologies to exploit this capability to its maximum potential while still maintaining a given QoS and utility to users.

The rest of this manuscript is as follows: First, we look at what virtualization is,

and what current technologies currently exist within the mainstream market. Next we discuss previous work related to virtualization and take an in-depth look at the features provided by each hypervisor. We follow this by outlining an experimental setup to evaluate a set of today's hypervisors on a novel Cloud test-bed architecture. Then, we look at performance benchmarks which help explain the utility of each hypervisor and the feasibility within an HPC environment. We conclude with our final thoughts and recommendations for using virtualization in Clouds for HPC.

3.3 Related Research

While the use of virtualization technologies has increased dramatically in the past few years, virtualization is not specific to the recent advent of Cloud computing. IBM originally pioneered the concept of virtualization in the 1960's with the M44/44X systems [21]. It has only recently been reintroduced for general use on x86 platforms. Today there are a number of public Clouds that offer IaaS through the use of virtualization technologies. The Amazon Elastic Compute Cloud (EC2) [22] is probably the most popular Cloud and is used extensively in the IT industry to this day. Nimbus [**?**, 14] and Eucalyptus [11] are popular private IaaS platforms in both the scientific and industrial communities. Nimbus, originating from the concept of deploying virtual workspaces on top of existing Grid infrastructure using Globus, has pioneered scientific Clouds since its inception. Eucalyptus has historically focused on providing an exact EC2 environment as a private cloud to enable users to build an EC2-like cloud using their own internal resources. Other scientific Cloud specific projects exist such as OpenNebula [12], In-VIGO [**?**], and Cluster-on-Demand [13], all of which leverage one or more hypervisors to provide computing infrastructure on demand. In recent history, OpenStack [**?**] has also come to light from a joint col-

laboration between NASA and Rackspace which also provide compute and storage resources in the form of a Cloud.

While there are currently a number of virtualization technologies available today, the virtualization technique of choice for most open platforms over the past 5 years has typically been the Xen hypervisor [8]. However more recently VMWare ESX [**?**] [1], Oracle VirtualBox [**?**] and the Kernel-based Virtual Machine (KVM) [**?**] are becoming more commonplace. As these look to be the most popular and feature-rich of al virtualization technologies, we look to evaluate all four to the fullest extent possible. There are however, numerious other virtualizaton technologies also available, including Microsoft's Hyper-V [**?**], Parallels Virtuozzo [**?**], QEMU [**?**], OpenVZ [**?**], Oracle VM [**?**], and many others. However, these virtualization technologies have yet to seen widespread deployment within the HPC community, at least in their current form, so they have been placed outside the scope of this work.

In recent history there have actually been a number of comparisons related to virtualization technologies and Clouds. The first performance analysis of various hypervisors started with, unsurprisingly, the hypervisor vendors themselves. VMWare has happy to put out its on take on performance in [**?**], as well as the original Xen article [8] which compares Xen, XenoLinux, and VMWare across a number of SPEC and normalized benchmarks, resulting in a conflict between both works. From here, a number of more unbiased reports originated, concentrating on server consolidation and web application performance [**?**, **?**, 23] with fruitful yet sometimes incompatible results. A feature base survey on virtualization technologies [**?**] also illustrates the wide variety of hypervisors that currently exist. Furthermore, there has been some investigation into the performance within HPC, specifically with InfiniBand performance of Xen [**?**] and rather recently with a detailed look at the feasibility of the

[1] Due to the restrictions in VMWare's licensing agreement, benchmark results are unavailable.

Amazon Elastic Compute cloud for HPC applications [**?**], however both works concentrate only on a single deployment rather than a true comparison of technologies.

As these underlying hypervisor and virtualization implementations have evolved rapidly in recent years along with virtualization support directly on standard x86 hardware, it is necessary to carefully and accurately evaluate the performance implications of each system. Hence, we conducted an investigation of several virtualization technologies, namely Xen, KVM, VirtualBox, and in part VMWare. Each hypervisor is compared alongside one another with base-metal as a control and (with the exeption of VMWare) run through a number of High Performance benchmarking tools.

3.4 Feature Comparison

With the wide array of potential choices of virtualization technologies available, its often difficult for potential users to identify which platform is best suited for their needs. In order to simplify this task, we provide a detailed comparison chart between Xen 3.1, KVM from RHEL5, VirtualBox 3.2 and VMWWare ESX in Figure 2.

	Xen	KVM	VirtualBox	VMWare
Para-virtualization	Yes	No	No	No
Full virtualization	Yes	Yes	Yes	Yes
Host CPU	x86, x86-64, IA-64	x86, x86-64,IA64,PPC	x86, x86-64	x86, x86-64
Guest CPU	x86, x86-64, IA-64	x86, x86-64,IA64,PPC	x86, x86-64	x86, x86-64
Host OS	Linux, UNIX	Linux	Windows, Linux, UNIX	Proprietary UNIX
Guest OS	Linux, Windows, UNIX	Linux, Windows, UNIX	Linux, Windows, UNIX	Linux, Windows, UNIX
VT-x / AMD-v	Opt	Req	Opt	Opt
Cores supported	128	16	32	8
Memory supported	4TB	4TB	16GB	64GB
3D Acceleration	Xen-GL	VMGL	Open-GL	Open-GL, DirectX
Live Migration	Yes	Yes	Yes	Yes
License	GPL	GPL	GPL/proprietary	Proprietary

Figure 3.1 A comparison chart between Xen, KVM, VirtualBox, and VMWare ESX

The first point of investigation is the virtualization method of each VM. Each

hypervisor supports full virtualization, which is now common practice within most x86 virtualization deployments today. Xen, originating as a para-virtualized VMM, still supports both types, however full virtualization is often preferred as it does not require the manipulation of the guest kernel in any way. From the Host and Guest CPU lists, we see that x86 and, more specifically, x86-64/amd64 guests are all universally supported. Xen and KVM both suport Itanium-64 architectures for full virtualization (due to both hypervisors dependency on QEMU), and KVM also claims support for some recent PowerPC architectures. However, we concern ourselves only with x86-64 features and performance, as other architectures are out of the scope of this manuscript. Of the x86-64 platforms, KVM is the only hypervisor to require either Intel VT-X or AMD-V instruction sets in order to operate. VirtualBox and VMWare have internal mechanisms to provide full virtualization even without the virtualization instruction sets, and Xen can default back to para-virtualized guests.

Next, we consider the host environments for each system. As Linux is the primary OS type of choice within HPC deployments, its key that all hypervisors support Linux as a guest OS, and also as a host OS. As VMWare ESX is meant to be a virtualization-only platform, it is built upon a specially configured Linux/UNIX proprietary OS specific to its needs. All other hypervisors support Linux as a host OS, with VirtualBox also supporting Windows, as it was traditionally targeted for desktop-based virtualization. However, as each hypervisor uses VT-X or AMD-V instructions, each can support any modern OS targeted for x86 platforms, including all variants of Linux, Windows, and UNIX.

While most hypervisors have desirable host and guest OS support, hardware support within a guest environment varies drastically. Within the HPC environment, virtual CPU (vCPU) and maximum VM memory are critical aspects to choosing the right virtualization technology. In this case, Xen is the first choice as it supports up

to 128 vCPUs and can address 4TB of main memory in 64-bit modes, more than any other. VirtualBox, on the other hand, supports only 32 vCPUs and 16GB of addressable RAM per guest OS, which may lead to problems when looking to deploy it on large multicore systems. KVM also faces an issue with the number of vCPU supported limited to 16, recent reports indicate it is only a soft limit [24], so deploying KVM in an SMP environment may not be a significant hurdle. Furthermore, all hypervisors provide some 3D acceleration support (at least for OpenGL) and support live migration across homogeneous nodes, each with varying levels of success.

Another vital juxtaposition of these virtualization technologies is the license agreements for its applicability within HPC deployments. Xen, KVM, and VirtualBox are provided for free under the GNU Public License (GPL) version 2, so they are open to use and modification by anyone within the community, a key feature for many potential users. While VirtualBox is under GPL, it has recently also offered with additional features under a more proprietary license dictated by Oracle since its acquirement from Sun last year. VMWare, on the other hand, is completely proprietary with an extremely limited licensing scheme that even prevents the authors from willfully publishing any performance benchmark data without specific and prior approval. As such, we have neglected VMWare form the remainder of this manuscript. Whether going with a proprietary or open source hypervisor, support can be acquired (usually for an additional cost) with ease from each option.

3.4.1 Usability

While side by side feature comparison may provide crucial information about a potential user's choice of hypervisor, that may also be interested in its ease of installation and use. We will take a look at each hypervisor from two user perspectives, a systems administrator and normal VM user.

One of the first things on any system administrator's mind on choosing a hypervisor is the installation. For all of these hypervisors, installation is relatively painless. For the FutureGrid support group, KVM and VirualBox are the easiest of the all tested hypervisors to install, as there are a number of supported packages available and installation only requires the addition of one or more kernel modules and the support software. Xen, while still supported in binary form by many Linux distributions, is actually much more complicated. This is because Xen requires a full modification to the kernel itself, not just a module. Loading a new kernel into the boot process which may complicate patching and updating later in the system's maintenance cycle. VMWare ESX, on the other hand, is entirely separate from most other installations. As previously noted, ESX is actually a hypervisor and custom UNIX host OS combined, so installation of ESX is likewise to installing any other OS from scratch. This may be either desirable or adverse, depending on the system administrator's usage of the systems and VMWare's ability to provide a secure and patched environment.

While system administrators may be concerned with installation and maintenance, VM users and Cloud developers are more concerned with daily usage. The first thing to note about all of such virtualiation technologies is they are supported (to some extent) by the libvirt API [**?**]. Libvirt is commonly used by many of today's IaaS Cloud offerings, including Nimbus, Eucalyptus, OpenNebula and OpenStack. As such, the choice of hypervisor for Cloud developer's is less of an issue, so long as the hypervisor supports the features they desire. For individual command line usage of each tool, it varies quite a bit more. Xen does provide their own set of tools for controlling and monitoring guests, and seem to work relatively well but do incur a slight learning curve. KVM also provides its own CLI interface, and while it is often considered less cumbersome it provides less advanced features directly to users, such as power management or quick memory adjustment (however this is subject to personal

opinion). One advantage of KVM is each guest actually runs as a separate process within the host OS, making it easy for a user to manage and control the VM inside the host if KVM misbehaves. VirtualBox, on the other hand, provides the best command line and graphical user interface. The CLI, is especially well featured when compared to Xen and KVM as it provides clear, decisive and well documented commands, something most HPC users and system administrators alike will appreciate. VMWare provides a significantly enhanced GUI as well as a Web-based ActiveX client interface that allows users to easily operate the VMWare host remotely. In summary, there is a wide variance of interfaces provided by each hypervisor, however we recommend Cloud developers to utilize the libvirt API whenever possible.

3.5 Experimental Design

In order to provide an unaltered and unbiased review of these virtualization technologies for Clouds, we need to outline a neutral testing environment. To make this possible, we have chosen to use FutureGrid as our virtualization and cloud test-bed.

3.5.1 The FutureGrid Project

FutureGrid (FG) [25] provides computing capabilities that enable researchers to tackle complex research challenges related to the use and security of Grids and Clouds. These include topics ranging from authentication, authorization, scheduling, virtualization, middleware design, interface design and cybersecurity, to the optimization of Grid-enabled and cloud-enabled computational schemes for researchers in astronomy, chemistry, biology, engineering, atmospheric science and epidemiology.

The test-bed includes a geographically distributed set of heterogeneous computing systems, a data management system that will hold both metadata and a growing

library of software images necessary for Cloud computing, and a dedicated network allowing isolated, secure experiments, as seen in Figure 3.2. The test-bed supports virtual machine-based environments, as well as operating systems on native hardware for experiments aimed at minimizing overhead and maximizing performance. The project partners are integrating existing open-source software packages to create an easy-to-use software environment that supports the instantiation, execution and recording of grid and cloud computing experiments.

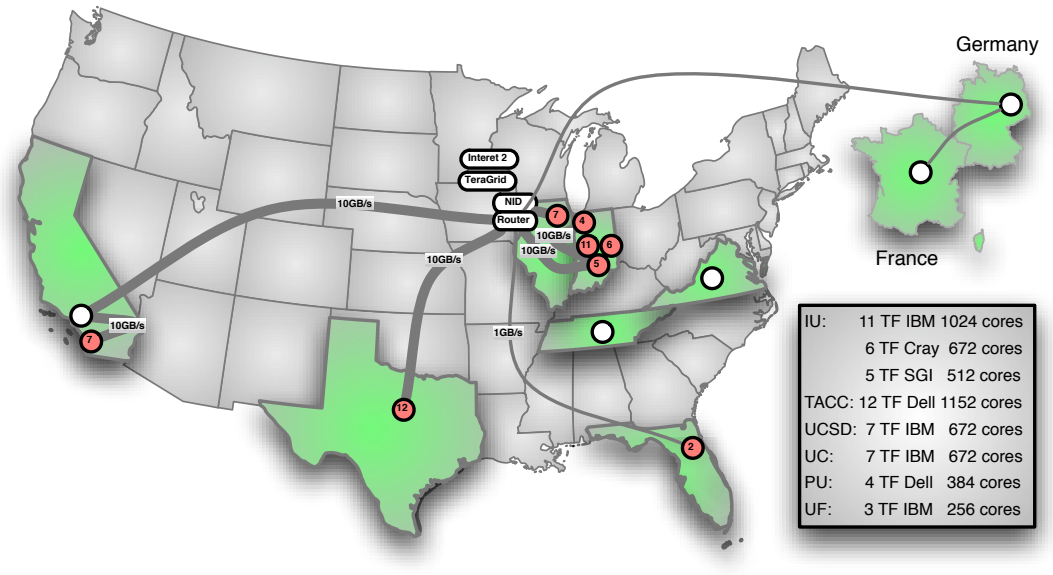

Figure 3.2 FutureGrid Participants and Resources

One of the goals of the project is to understand the behavior and utility of Cloud computing approaches. However, it is not clear at this time which of these toolkits will become the users' choice toolkit. FG provides the ability to compare these frameworks with each other while considering real scientific applications [26]. Hence, researchers are be able to measure the overhead of cloud technology by requesting linked experiments on both virtual and bare-metal systems, providing valuable information that help decide which infrastructure suits their needs and also helps users that want to

transition from one environment to the other. These interests and research objectives make the FutureGrid project the perfect match for this work. Furthermore, we expect that the results gleaned from this manuscript will have a direct impact on the FutureGrid deployment itself.

3.5.2 Experimental Environment

Currently, one of FutureGrid's latest resources is the *India* system, a 256 CPU IBM iDataPlex machine consisting of 1024 cores, 2048 GB of ram, and 335 TB of storage within the Indiana University Data Center. In specific, each compute node of India has two Intel Xeon 5570 quad core CPUs running at 2.93Ghz, 24GBs of Ram, and a QDR InfiniBand connection. A total of four nodes were allocated directly from India for these experiments. All were loaded with a fresh installation of Red Hat Enterprise Linux server 5.5 x86_64 with the 2.6.18-194.8.1.el5 kernel patched. Three of the four nodes were installed with different hypervisors; Xen version 3.1, KVM (build 83), and VirtualBox 3.2.10, and the forth node was left as-is to act as a control for bare-metal native performance.

Each guest virtual machine was also built using Red Hat EL server 5.5 running an unmodified kernel using full virtualization techniques. All tests were conducted giving the guest VM 8 cores and 16GB of ram to properly span a compute node. Each benchmark was run a total of 20 times, with the results averaged to produce consistent results, unless indicated otherwise.

3.5.3 Benchmarking Setup

As this manuscript aims to objectively evaluate each virtualization technology from a side-by-side comparison as well as from a performance standpoint, the selection of

benchmarking applications is critical.

The performance comparison of each virtual machine is based on two well known industry standard performance benchmark suites; HPCC and SPEC. These two benchmark environments are recognized for their standardized reproducible results in the HPC communit, and the National Science Foundation (NSF), Department of Energy (DOE), and DARPA are all sponsors of the HPCC benchmarks. The following benchmarks provide a means to stress and compare processor, memory, inter-process communication, network, and overall performance and throughput of a system. These benchmarks were selected due to their importance to the HPC community sinse they are often directly correlated with overall application performance [**?**].

HPCC Benchmarks

The HPCC Benchmarks [**?**, **?**] are an industry standard for performing benchmarks for HPC systems. The benchmarks are aimed at testing the system on multiple levels to test their performance. It consists of 7 different tests:

- *HPL* - The Linpack TPP benchmark measures the floating point rate of execution for solving a linear system of equations. This benchmark is perhaps the most important benchmark within HPC today, as it is the basis of evaluation for the Top 500 list [27].

- DGEMM - Measures the floating point rate of execution of double precision real matrix-matrix multiplication.

- *STREAM* - A simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel.

- *PTRANS* - Parallel matrix transpose exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.

- *RandomAccess* - Measures the rate of integer random updates of memory (GUPS).

- *FFT* - Measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).

- *Communication bandwidth and latency* - A set of tests to measure latency and bandwidth of a number of simultaneous communication patterns; based on b_eff (effective bandwidth benchmark).

This benchmark suite uses each test to stress test the performance on multiple aspects of the system. It also provides reproducible results which can be verified by other vendors. This benchmark is used to create the Top 500 list [27] which is the list of the current top supercomputers in the world. The results that are obtained from these benchmarks provide an unbiased performance analysis of the hypervisors. Our results provide insight on inter-node PingPong bandwidth, PingPong latency, and FFT calculation performance.

SPEC Benchmarks

The Standard Performance Evaluation Corporation (SPEC) [**?**, **?**] is the other major standard for evaluation of benchmarking systems. SPEC has several different testing components that can be utilized to benchmark a system. For our benchmarking comparison we will use the SPEC OMP2001 because it appears to represent a vast array of new and emerging parallel applications wile simultaniously providing a comparison to other SPEC benchmarks. SPEC OMP continues the SPEC tradition of giving

HPC users the most objective and representative benchmark suite for measuring the performance of SMP (shared memory multi-processor) systems.

- The benchmarks are adapted from SPEC CPU2000 and contributions to its search program.
- The focus is to deliver systems performance to real scientific and engineering applications.
- The size and runtime reflect the needs of engineers and researchers to model large complex tasks.
- Two levels of workload characterize the performance of medium and large sized systems.
- Tools based on the SPEC CPU2000 toolset make these the easiest ever HPC tests to run.
- These benchmarks place heavy demands on systems and memory.

3.6 Performance Comparison

The goal of this manuscript is to effectively compare and contrast the various virtualization technologies, specifically for supporting HPC-based Clouds. The first set of results represent the performance of HPCC benchmarks. Each benchmark was run a total of 20 times, and the mean values taken with error bars represented using the standard deviation over the 20 runs. The benchmarking suite was built using the Intel 11.1 compiler, uses the Intel MPI and MKL runtime libraries, all set with defaults and no optimizations whatsoever.

We open first with High Performance Linpack (HPL), the de-facto standard for comparing resources. In Figure 3.3, we can see the comparison of Xen, KVM, and Virtual Box compared to native bare-metal performance. First, we see that native is capable of around 73.5 Gflops which, with no optimizations, achieves 75% of the theoretical peak performance. Xen, KVM and VirtualBox perform at 49.1, 51.8 and 51.3 Gflops, respectively when averaged over 20 runs. However Xen, unlike KVM and VirtualBox, has a high degree of variance between runs. This is an interesting phenomenon for two reasons. First, this may impact performance metrics for other HPC applications and cause errors and delays between even pleasingly-parallel applications and add to reducer function delays. Second, this wide variance breaks a key component of Cloud computing providing a specific and predefined quality of service. If performance can sway as widely as what occurred for Linpack, then this may have a negative impact on users.

Next, we turn to another key benchmark within the HPC community, Fast Fourier Transforms (FFT). Unlike the synthetic Linpack benchmark, FFT is a specific, purposeful benchmark which provides results which are often regarded as more relative to a user's real-world application than HPL. From Figure 3.4, we can see rather distinct results from what was previously provided by HPL. Looking at Star and Single FFT, its clear performance across all hypervisors is roughly equal to bare-metal performance, a good indication that HPC applications may be well suited for use on VMs. The results for MPI FFT also show similar results, with the exception of Xen, which has a decreased performance and high variance as seen in the HPL benchmark. Our current hypothesis is that there is an adverse affect of using Intel's MPI runtime on Xen, however the investigation is still ongoing.

Another useful benchmark illustrative of real-world performance between bare-metal performance and various hypervisors are the ping-pong benchmarks. These

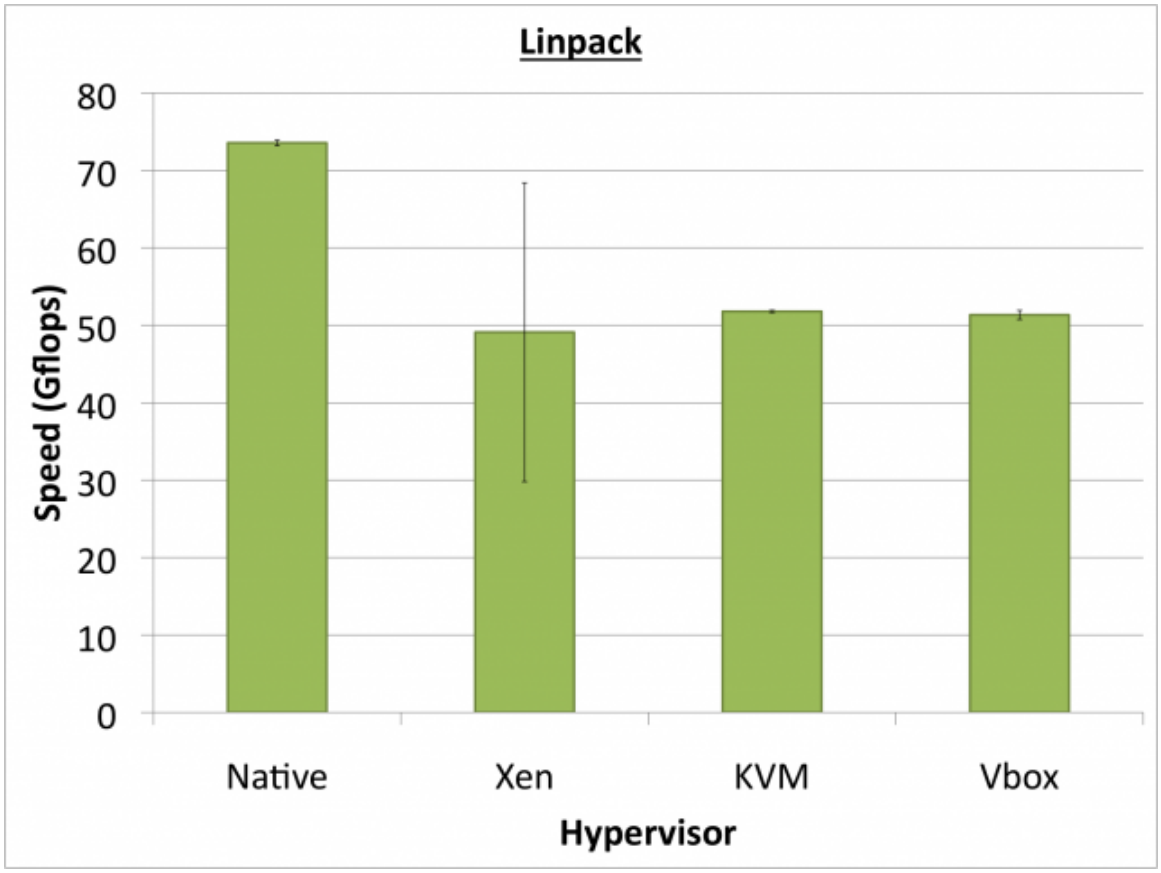

Figure 3.3 Linpack performance

benchmarks measure the bandwidth and latency of passing packets between multiple CPUs. With this experiment, all ping-pong latencies are kept within a given node, rather than over the network. This is done to provide further insight into the CPU and memory overhead withing each hypervisor. From Figure 3.5 the intranode bandwidth performance is uncovered, with some interesting distinctions between each hypervisor. First, Xen performs, on average, close to native speeds, which is promising for the hypervisor. KVM, on the other hand, shows consistent overhead proportional to native performance across minimum, average, and maximum bandwidth. VirtualBox, on the other hand, performs well, in fact too well to the point that raises alarm. While the minimum and average bandwidths are within native performance, the maximum bandwidth reported by VirtualBox is significantly greater than native

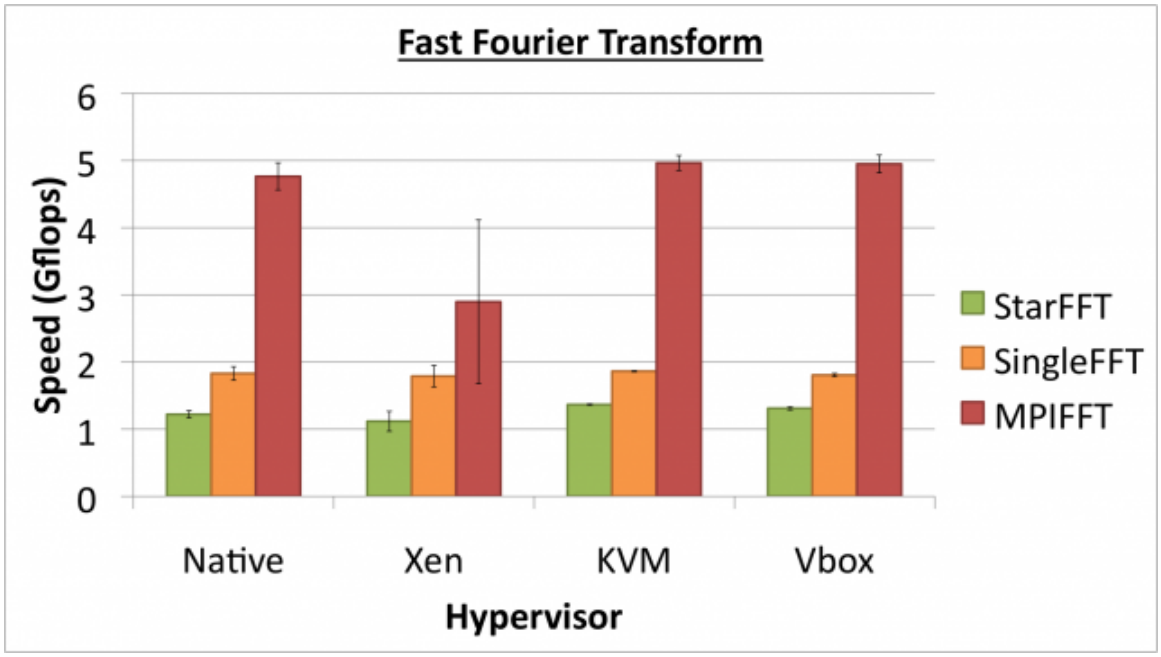

Figure 3.4 Fast Fourier Transform performance

measurements, with a large variance. After careful examination, it appears this is due to how VirtualBox assigns its virtual CPUs. Instead of locking a virtual CPU to a real CPU, a switch may occur which could benefit on the off-chance the two CPU's in communication between a ping-pong test could in fact be the same physical CPU. The result would mean the ping-pong packet would remain in cache and result in a higher perceived bandwidth than normal. While this effect may be beneficial for this benchmark, it may only be an illusion towards the real performance gleaned from the VirtualBox hypervisor.

The Bandwidth may in fact be important within the ping-ping benchmark, but the latency between each ping-pong is equally useful in understanding the performance impact of each virtualization technology. From Figure 3.6, we see KVM and VirtualBox have near-native performance; another promising result towards the utility of hypervisors within HPC systems. Xen, on the other hand, has extremely high latencies, especially at for maximum latencies, which in turn create a high variance

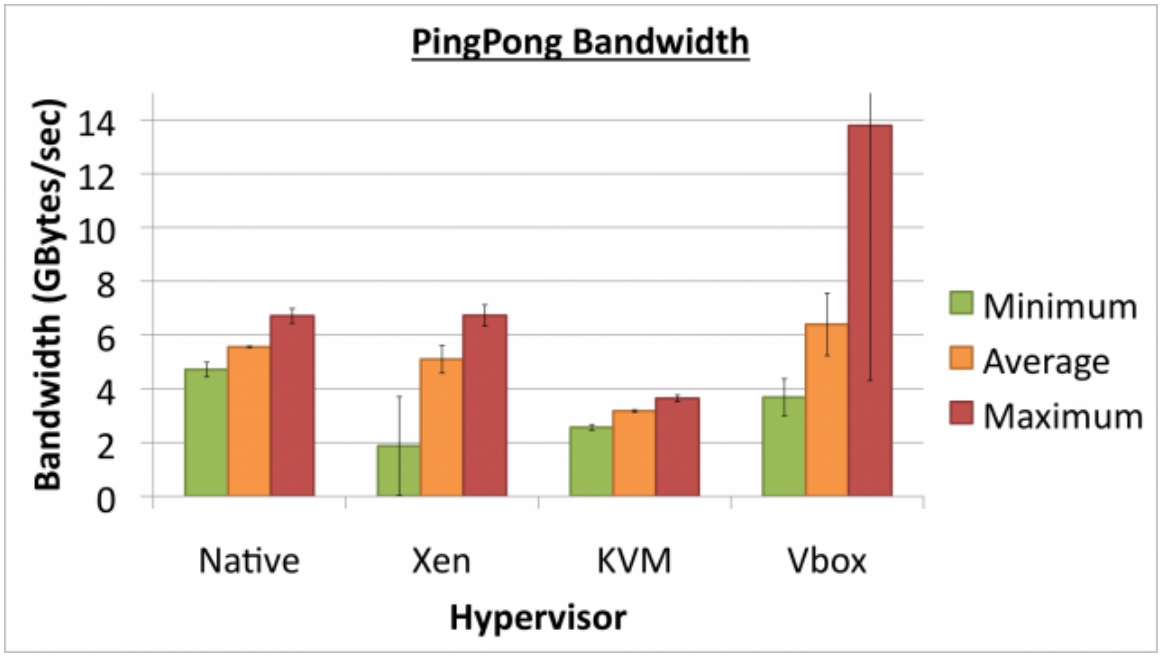

Figure 3.5 Ping Pong bandwidth performance

within the average latency within the VM's performance.

While the HPCC benchmarks provide a comprehensive view for many HPC applications including Linpack and FFT using MPI, performance of intra-node SMP applications using OpenMP is also investigated. Figure 3.7 illustrates SPEC OpenMP performance across the VMs we concentrate on, as well as baseline native performance. First, we see that the combined performance over all 11 applications executed 20 times yields the native testbed with the best performance at a SPEC score of 34465. KVM performance comes close with a score of 34384, which is so similar to the native performance that most users will never notice the difference. Xen and VirtualBox both perform notably slower with scores of 31824 and 31695, respectively, however this is only an 8% performance drop compared to native speeds. Further results can be found on the SPEC website [28].

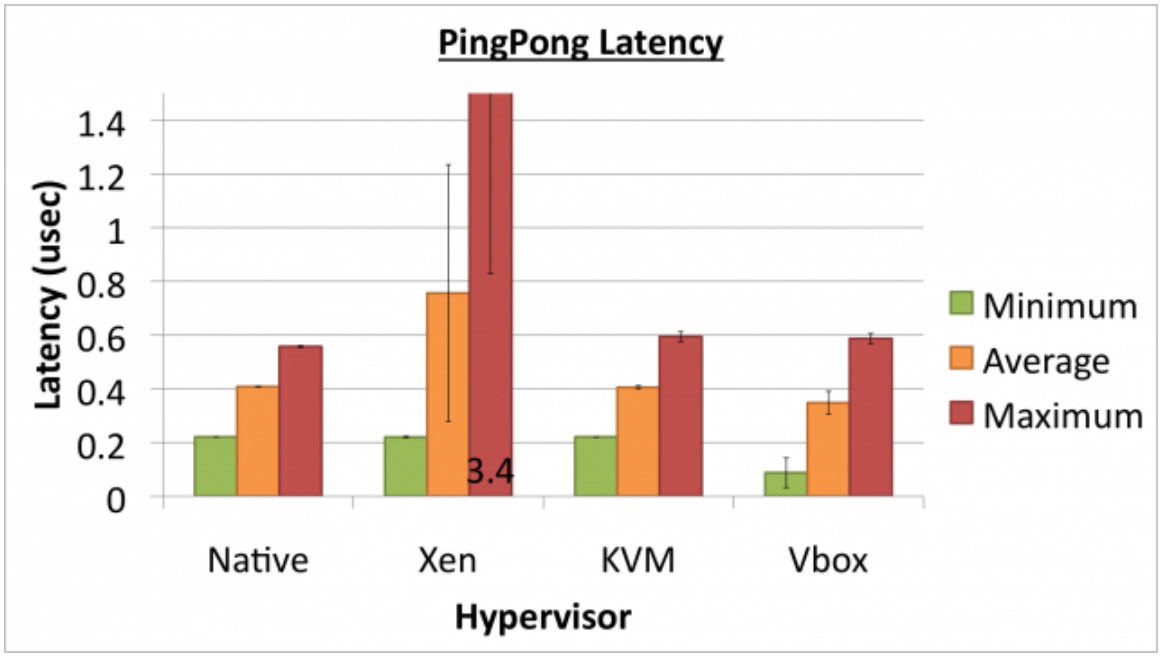

Figure 3.6 Ping Pong latency performance (lower is better)

3.7 Discussion

The primary goal of this manuscript is to evaluate the viability of virtualization within HPC. After our analysis, the answer seems to be a resounding "yes." However, we also hope to select the best virtualization technology for such an HPC environment. In order to do this, we combine the feature comparison along with the performance results, and evaluate the potential impact within the FutureGrid testbed.

From a feature standpoint, most of today's virtualization technologies fit the bill for at least small scale deployment, including VMWare. In short, each support Linux x86_64 platforms, use VT-X technology for full virtualization, and support live migration. Due to VMWare's limited and costly licensing, it is immediately out of contention for most HPC deployments. From a CPU and memory standpoint, Xen seems to provide the best expandability, supporting up to 128 cpus and 4TB of addressable RAM. So long as KVM's vCPU limit can be extended, it too shows promise

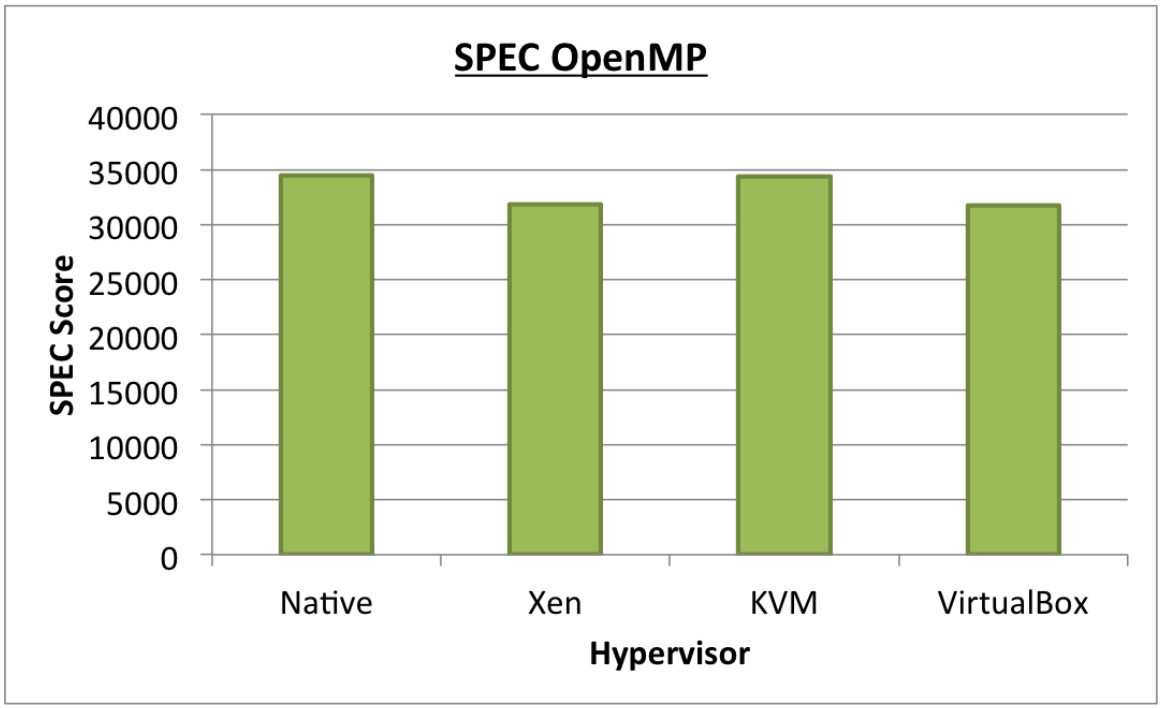

Figure 3.7 Spec OpenMP performance

as a feature-full virtualization technology. One of Virtualbox's greatest limitations was the 16GB maximum memory allotment for individual guest VMs, which actually limited us from giving VMs more memory for our performance benchmarks. If this can be fixed and Oracle does not move the product into the proprietary market, VirtualBox may also stand a chance for deployment in HPC environments.

From the benchmark results previously described, the use of hypervisors within HPC-based Cloud deployments is mixed batch. Figure 3.8 summarizes the results based on a 1-3 rating, 1 being best and 3 being worst. While Linpack performance seems to take a significant performance impact across all hypervisors, the more practical FFT benchmarks seem to show little impact, a notably good sign for virtualization as a whole. The ping-pong bandwidth and latency benchmarks also seem to support this theory, with the exception of Xen, who's performance continually has wide fluctuations throughout the majority of the benchmarks. OpenMP performance

	Xen	KVM	VirtualBox
Linpack	3	1	2
FFT	3	1	2
Bandwidth	2	3	1
Latency	3	2	1
OpenMP	2	1	3
Total Rating	13	8	9

Figure 3.8 Benchmark rating summary (lower is better)

through the SPEC OMP benchmarking suite also shows promising results for the use of hypervisors in general, with KVM taking a clear lead by almost matching native speeds.

While Xen is typically regarded as the most widely used hypervisor, especially within academic clouds and grids, it's performance has shown lack considerably when compared to either KVM or VirtualBox. In particular, Xen's wide and unexplained fluctuations in performance throughout the series of benchmarks suggests that Xen may not be the best choice for building a lasting quality of service infrastructure upon. From Figure 3.8, KVM rates the best across all performance benchmarks, making it the optimal choice for *general* deployment in an HPC environment. Furthermore, this work's illustration of the variance in performance among each benchmark and the applicability of each benchmark towards new applications may make possible the ability to preemptively classify applications for accurate prediction towards the ideal virtualized Cloud environment. We hope to further investigate this concept through the use of the FutureGrid experiment management framework at a later date.

In conclusion, it is the authors' projection that KVM is the best overall choice for

use within HPC Cloud environments. KVM's feature-rich experience and near-native performance makes it a natural fit for deployment in an environment where usability and performance are paramount. Within the FutureGrid project specifically, we hope to deploy the KVM hypervisor across our Cloud platforms in the near future, as it offers clear benefits over the current Xen deployment. Furthermore, we expect these findings to be of great importance to other public and private Cloud deployments, as system utilization, Quality of Service, operating cost, and computational efficiency could all be improved through the careful evaluation of underlying virtualization technologies.

Chapter 4

Evaluating GPU Passthrough in Xen for High Performance Cloud Computing

4.1 Abstract

With the advent of virtualization and Infrastructure-as-a-Service (IaaS), the broader scientific computing community is considering the use of clouds for their technical computing needs. This is due to the relative scalability, ease of use, advanced user environment customization abilities clouds provide, as well as many novel computing paradigms available for data-intensive applications. However, there is concern about a performance gap that exists between the performance of IaaS when compared to typical high performance computing (HPC) resources, which could limit the applicability of IaaS for many potential scientific users.

Most recently, general-purpose graphics processing units (GPGPUs or GPUs) have become commonplace within high performance computing. We look to bridge the

gap between supercomputing and clouds by providing GPU-enabled virtual machines (VMs) and investigating their feasibility for advanced scientific computation. Specifically, the Xen hypervisor is utilized to leverage specialized hardware-assisted I/O virtualization and PCI passthrough in order to provide advanced HPC-centric Nvidia GPUs directly in guest VMs. This methodology is evaluated by measuring the performance of two Nvidia Tesla GPUs within Xen VMs and comparing to bare-metal hardware. Results show PCI passthrough of GPUs within virtual machines is a viable use case for many scientific computing workflows, and could help support high performance cloud infrastructure in the near future.

4.2 Introduction

Cloud computing [29] has established itself as a prominent paradigm within the realm of Distributed Systems [30] in a very short period of time. Clouds are an internet-based solution that provide computational and data models for utilizing resources, which can be accessed directly by users on demand in a uniquely scalable way. Cloud computing functions by providing a layer of abstraction on top of base hardware to enable a new set of features that are otherwise intangible or intractable. These benefits and features include Scalability, a guaranteed Quality of Service (QoS), cost effectiveness, and direct user customization via a simplified user interface [31].

While the origin of cloud computing is based in industry through solutions such as Amazon's EC2 [10], Google's MapReduce [32], and Microsoft's Azure [33], the paradigm has since become integrated in all areas of science and technology. Most notably, there is an increasing effort within the High Performance Computing (HPC) community to leverage the utility of clouds for advanced scientific computing to solve a number of challenges still standing in the field. This can be clearly seen in large-scale

efforts such as the FutureGrid project [34], the Magellan project [35], and through various other Infrastructure-as-a-Service projects including OpenStack [36], Nimbus [37], and Eucalyptus [38].

Within HPC, there has also been a notable movement toward dedicated accelerator cards such as general purpose graphical processing units (GPGPUs, or GPUs) to enhance scientific computation problems by upwards of two orders of magnitude. This is accomplished through dedicated programming environments, compilers, and libraries such as CUDA [39] from Nvidia as well as the OpenCL effort [40]. When combining GPUs in an otherwise typical HPC environment or supercomputer, major gains in performance and computational ability have been reported in numerous fields [**?**, **?**], ranging from Astrophysics to Bioinformatics. Furthermore, these gains in computational power have also reportedly come at an increased performance-per-watt [41], a metric that is increasingly important to the HPC community as we move closer to exascale computing [42] where power consumption is quickly becoming the primary constraint.

With the advent of both clouds and GPUs within the field of scientific computing, there is an immediate and ever-growing need to provide heterogeneous resources, most immediately GPUs, within a cloud environment in the same scalable, on-demand, and user-centric way that many cloud users are already accustomed to [43]. While this task alone is nontrivial, it is further complicated by the high demand for performance within HPC. As such, it is performance that is paramount to the success of deploying GPUs within cloud environments, and thus is the central focus of this work.

This manuscript is organized as follows. First, in Section 2, we discuss the related research and the options currently available for providing GPUs within a virtualized cloud environment. In Section 3, we discuss the methodology for providing GPUs directly within virtual machines. In Section 4 we outline the evaluation of the given

methodology using two different Nvidia Tesla GPUs and compare to the best-case native application in Section 5. Then, we discuss the implications of these results in Section 6 and consider the applicability of each method within a production cloud system. Finally, we conclude with our findings and suggest directions for future work.

4.3 Virtual GPU Directions

Recently, GPU programming has been a primary focus for numerous scientific computing applications. Significant progress has been accomplished in many different workloads, both in science and engineering, based on parallel abilities of GPUs for floating point operations and very high on-GPU memory bandwidth. This hardware, coupled with CUDA and OpenCL programming frameworks, has led to an explosion of new GPU-specific applications. In some cases, GPUs outperform even the fastest multicore counterparts by an order of magnitude [44]. In addition, further research could leverage the per-node performance of GPU accelerators with the high speed, low latency interconnects commonly utilized in supercomputers and clusters to create a hybrid GPU + MPI class of applications. The number of distributed GPU applications is increasing substantially in supercomputing, usually scaling many GPUs simultaneously [45].

Since the establishment of cloud computing in industry, research groups have been evaluating its applicability to science [3]. Historically, HPC and Grids have been on similar but distinct paths within distributed systems, and have concentrated on performance, scalability, and solving complex, tightly coupled problems within science. This has led to the development of supercomputers with many thousands of cores, high speed, low latency interconnects, and sometimes also coprocessors and FPGAs [46, 47]. Only recently have these systems been evaluated from a cloud per-

spective [35]. An overarching goal exists to provide HPC Infrastructure as its own service (HPCaaS) [48], aiming to classify and limit the overhead of virtualization, and reducing the bottlenecks classically found in CPU, memory, and I/O operations within hypervisors [**?**,49]. Furthermore, the transition from HPC to cloud computing becomes more complicated when we consider adding GPUs to the equation.

GPU availability within a cloud is a new concept that has sparked a large amount of interest within the community. The first successfully deployment of GPUs within a cloud environment was the Amazon EC2 GPU offering. A collaboration between Nvidia and Citrix also exists to provide cloud-based gaming solutions to users using the new Kepler GPU architecture [50]. However, this is currently not targeted towards HPC applications.

The task of providing a GPU accelerator for use in a virtualized cloud environment is one that presents a myriad of challenges. This is due to the complicated nature of virtualizing drivers, libraries, and the heterogeneous offerings of GPUs from multiple vendors. Currently, two possible techniques exist to fill the gap in providing GPUs in a cloud infrastructure: back-end I/O virtualization, which this manuscript focuses on, and Front-end remote API invocation.

4.3.1 Front-end Remote API invocation

One method for using GPUs within a virtualized cloud environment is through front-end library abstractions, the most common of which is remote API invocation. Also known as API remoting or API interception, it represents a technique where API calls are intercepted and forwarded to a remote host where the actual computation occurs. The results are then returned to the front-end process that spawned the invocation, potentially within a virtual machine. The goal of this method is to provide an emulated device library where the actual computation is offloaded to another

resource on a local network.

Front-end remote APIs for GPUs have been implemented by a number of different technologies for different uses. To solve the problem of graphics processing in VMs, VMWare [51] has developed a device-emulation approach that emulates the Direct3D and OpenGL calls to leverage the host OS graphics processing capabilities to provide a 3D environment within a VM. API interception through the use of wrapper binaries has also been implemented by technologies such as Chromium [52], and Blink. However these graphics processing front-end solutions are not suitable for general purpose scientific computing, as they do not expose interfaces that CUDA or OpenCL can use.

Currently, efforts are being made to provide a front-end remote API invocation solutions for the CUDA programming architecture. vCUDA [53] was the first of such technologies to enable transparent access of GPUs within VMs by API call interception and redirection of the CUDA API. vCUDA substitutes the CUDA runtime library and supports a transmission mode using XMLRPC, as well as a sharing mode that is built on VMRPC, a dedicated remote procedure call architecture for VMM platforms. This share model can leads to better performance, especially as the volume of data increases, although there may be limitations in VMM interoperability.

Like vCUDA, gVirtuS uses API interception to enable transparent CUDA, OpenCL, and OpenGL support for Xen, KVM, and VMWare virtual machines [54]. gVirtuS uses a front-end/back-end model to provide a VMM-independent abstraction layer to GPUs. Data transport from gVirtuS' front-end to the back-end is accomplished through a combination of shared memory, sockets, or other hypervisor-specific APIs. gVirtuS' primary disadvantage is in its decreased performance in host-to-device and device-to-host data movement due to overhead of data copies to and from its shared memory buffers. Recent work has also enabled the dynamic sharing of GPUs by lever-

aging the gVirtus back-end system with relatively good results [**?**], however process-level GPU resource sharing is outside the scope of this manuscript.

rCUDA [**?**, 55], a recent popular remote CUDA framework, also provides remote API invocation to enable VMs to access remote GPU hardware by using a sockets based implementation for high-speed near-native performance of CUDA based applications. rCUDA recently added support for using InfiniBand's high speed, low latency network to increase performance for CUDA applications with large data volume requirements. rCUDA version 4.1 also supports the CUDA runtime API version 5.0, which supports peer device memory access and unified addressing. One drawback of this method is that rCUDA cannot implement the undocumented and hidden functions within the runtime framework, and therefore does not support all CUDA C extensions. While rCUDA provides some support tools, native execution of CUDA programs is not possible and programs need to be recompiled or rewritten to use rCUDA. Furthermore, like gVirtuS and many other solutions, performance between host-to-device data movement is only as fast as the underlying interconnect, and in the best case with native RDMA InfiniBand, is roughly half as fast as native PCI Express usage when using the standard QDR InfiniBand.

4.3.2 Back-end PCI passthrough

Another approach to using a GPU in a virtualized environment is to provide a VM with direct access to the GPU itself, instead of relying on a remote API. This manuscript focuses on such an approach. Devices on a host's PCI-express bus are virtualized using directed I/O virtualization technologies recently implemented by chip manufacturers, and then direct access is relinquished upon request to a guest VM. This can be accomplished using the VT-d and IOMMU instruction sets from Intel and AMD, respectively. This mechanism, typically called PCI passthrough, uses a mem-

ory management unit (MMU) to handle direct memory access (DMA) coordination and interrupt remapping directly to the guest VM, thus bypassing the host entirely. With host involvement being nearly non-existent, near-native performance of the PCI device within the guest VM can be achieved, which is an important characteristic for using a GPU within a cloud infrastructure.

PCI passthrough itself has recently become a standard technique for many other I/O systems such as storage or network controllers. However, GPUs (even from the same vendor) have additional legacy VGA compatibility issues and non-standard low-level interface DMA interactions that make direct PCI passthrough nontrivial. VMWare has started use of a vDGA system for hardware GPU utilization, however it remains in tech preview and only documentation for Windows VMs is present [51]. In our experimentation, we have found that the Xen hypervisor provides a good platform for performing PCI passthrough of GPU devices to VMs due to its open nature, extensive support, and high degree of reconfigurability. Work with Xen in [**?**] gives hints at good performance for PCI passthrough in Xen, however further evaluation with independent benchmarks is needed when looking at scientific computing with GPUs.

Today's GPUs can provide a variety of frameworks for application programmers to use. Two common solutions are CUDA and OpenCL. CUDA, or the Compute Unified Device Architecture, is a framework for creating and running parallel applications on Nvidia GPUs. OpenCL provides a more generic and open framework for parallel computation on CPUs and GPUs, and is available for a number of Nvidia, AMD, and Intel GPUs and CPUs. While OpenCL provides a more robust and portable solution, many HPC applications utilize the CUDA framework. As such, we focus only on Nvidia based CUDA-capable GPUs as they offer the best support for a wide array of programs, although this work is not strictly limited to Nvidia GPUs.

4.4 Implementation

In this manuscript we use a specific host environment to enable PCI passthrough. First, we start with the Xen 4.2.2 hypervisor on Centos 6.4 and a custom 3.4.50-8 Linux kernel with Dom0 Xen support. Within the Xen hypervisor, GPU devices are seized upon boot and assigned to the xen-pciback kernel module. This process blocks the host devices form loading the Nvidia or nouveau drivers, keeping the GPUs uninitialized and therefore able to be assigned to DomU VMs.

Xen, like other hypervisors, provides a standard method of passing through PCI devices to guest VMs upon creation. When assigning a GPU to a new VM, Xen loads a specific VGA BIOS to properly initialize the device enabling DMA and interrupts to be assigned to the guest VM. Xen also relinquishes control of the GPU via the xen-pciback module. From there, the Linux Nvidia drivers are loaded and the device is able to be used as expected within the guest. Upon VM termination, the xen-pciback module re-seizes the GPU and the devices can be re-assigned to new VMs in the future.

This mechanism of PCI passthrough for GPUs can be implemented using multiple devices per host, as illustrated in Figure 6.1. Here, we see how the device's connection to the VM totally bypasses the Dom0 host as well as the Xen VMM, and is managed by VT-d or IOMMU to access the PCI-Express bus which the GPUs utilize. This is in contrast to other common virtual device uses, where hardware is emulated by the host and shared across all guests. This is the common usage for Ethernet controllers and input devices to enable users to interact with VMs as they would with native hosts, unlike the bridged model shown in the figure. The potential downside of this method is there can be a 1:1 or 1:N mapping of VMs to GPUs only. A M:1 mapping where multiple VMs use a GPU is not possible. However, almost all scientific applications

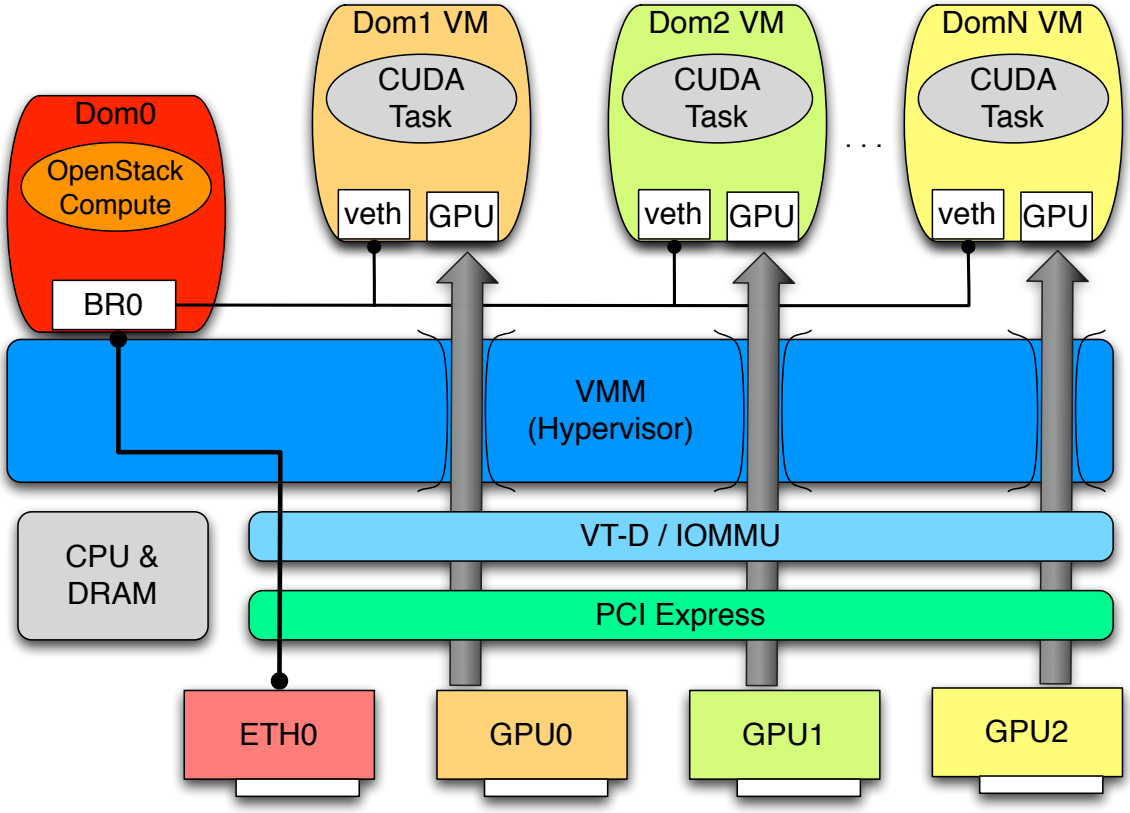

Figure 4.1 GPU PCI passthrough within the Xen Hypervisor

environments using GPUs generally do not share GPUs between processes or other nodes, as doing so would cause unpredictable and serious performance degradation. As such, this GPU isolation within a VM can be considered an advantage in many contexts.

4.4.1 Feature Comparison

Using the GPU PCI passthrough technique described previously has a number of advantages compared to front-end API implementations. First, it allows for an operating environment that more closely relates to native bare-metal usage of GPUs. Essentially, a VM provides a nearly identical infrastructure to clusters and supercomputers with integrated GPUs. This lowers the learning curve for many researchers,

and even enables researchers to potentially use other tools within a VM that might not be supported within a supercomputer or cluster. Unlike with remote API implementations, users don't need to recompile or modify their code, as the GPUs are essentially local to the data. Further comparing to remote API implementations, using PCI passthrough within a VM allows users to leverage any GPU framework available, OpenCL or CUDA, and any higher level programming frameworks such as within Matlab or Python.

Through the use of advanced scheduling techniques within cloud infrastructure, we can also take advantage of PCI passthrough implementation for efficiency purposes. Users could request VMs with GPUs which get scheduled for creation on machines that provide such resources, but can also request normal VMs as well. The scheduler can correctly map VM requirement requests to heterogeneous hardware. This enables large scale resources to support a wide array of scientific computing applications without the added cost of putting GPUs in all compute nodes.

4.5 Experimental Setup

In this manuscript back-end GPU PCI passthrough to virtual machines using the Xen hypervisor is detailed, however proper evaluation of the performance of such method needs to be properly considered. As such, we ran an array of benchmarks that evaluate the performance of this method compared to the same hardware running native bare-metal GPU code without any virtualization. We focus our tests on single-node performance to best understand low level overhead.

To evaluate the effectiveness of GPU-enabled VMs within Xen, two different machines were used to represent two generations of Nvidia GPUs. The first system at Indiana University consists of dual-socket Intel Xeon X5660 6-core CPUs at 2.8Ghz

with 192GB DDR3 RAM, 8TB RAID5 array, and two Nvidia Tesla "Fermi" C2075 GPUs. The system at USC/ISI uses a dual-socket Intel Xeon E5-2670 8-core CPUs at 2.6Ghz with 48GB DDR3 RAM, 3 600GB SAS disk drives, but with the latest Nvidia Tesla "Kepler" K20m GPU supplied by Nvidia. These machines represent the present Fermi series GPUs along with the recently release Kepler series GPUs, providing a well-rounded experimental environment. Native systems were installed with standard Centos 6.4 with a stock 2.6.32-279 Linux kernel. Xen host systems were still loaded with Centos 6.4 but with a custom 3.4.53-8 Linux kernel and Xen 4.2.22. Guest VMs were also Centos 6.4 with N-1 processors and 24GB of memory and 1 GPU passed through in HVM full virtualization mode. Using both IU and USC/ISI machine configurations in native and VM modes represent the 4 test cases for our work.

In order to evaluate the performance, the SHOC Benchmark suite [56] was used to extensively evaluate performance across each test platform. The SHOC benchmarks were chosen because they provide a higher level of evaluation regarding GPU performance than the sample applications provided in the Nvidia SDK, and can also evaluate OpenCL performance in similar detail. The benchmarks were compiled using the NVCC compiler in CUDA5 driver and library, along with OpenMPI 1.4.2 and GCC 4.4. Each benchmark was run a total of 20 times, with the results given as an average of all runs.

4.6 Results

Results of all benchmarks are compressed into three subsections: floating point operations, device bandwidth and pci bus performance. Each represents a different level of evaluation for GPU-enabled VMs compared to bare-metal native GPU usage.

4.6.1 Floating Point Performance

Flops, or floating point operations per second, are a common measure of computational performance, especially within scientific computing. The Top 500 list [27] has been the relative gold standard for evaluating supercomputing performance for more than two decades. With the advent of GPUs in some of the fastest supercomputers today, including GPUs identical to those used in our experimentation, understanding the performance relative to this metric is imperative.

Figure 4.2 shows the raw peak FLOPs available using each GPU in both native and virtualized modes. First, we observe the advantage of using the Kepler series GPUs over Fermi, with peak single precision speeds tripling in each case. Even for double precision FLOPs, there is roughly a doubling of performance with the new GPUs. However its most interesting that there is less than a 1% overhead when using GPU VMs compared to native case for pure floating point operations. The K20m-enabled VM was able to provide over 3 Teraflops, a notable computational feat for any single node.

Figures 4.3 and 4.4 show Fast Fourier Transform (FFT) and the Matrix Multiplication implementations across both test platforms. For all benchmarks that do not take into account the PCI-Express (pcie) bus transfer time, we again see near-native performance using the Kepler GPUs and Fermi GPUs when compared to bare metal cases. Interestingly, overhead within Xen VMs is consistently less than 1%, confirming the synthetic MaxFlops benchmark above. However, we do see some performance impact when calculating the total FLOPs with the pcie bus in the equation. This performance decrease ranges significantly for the C2075-series GPU, roughly about a 15% impact for FFT and a 5% impact for Matrix Multiplication. This overhead in pcie runs is not as pronounced for the Kepler K20m test environment, with near-native performance in all cases (less than 1%).

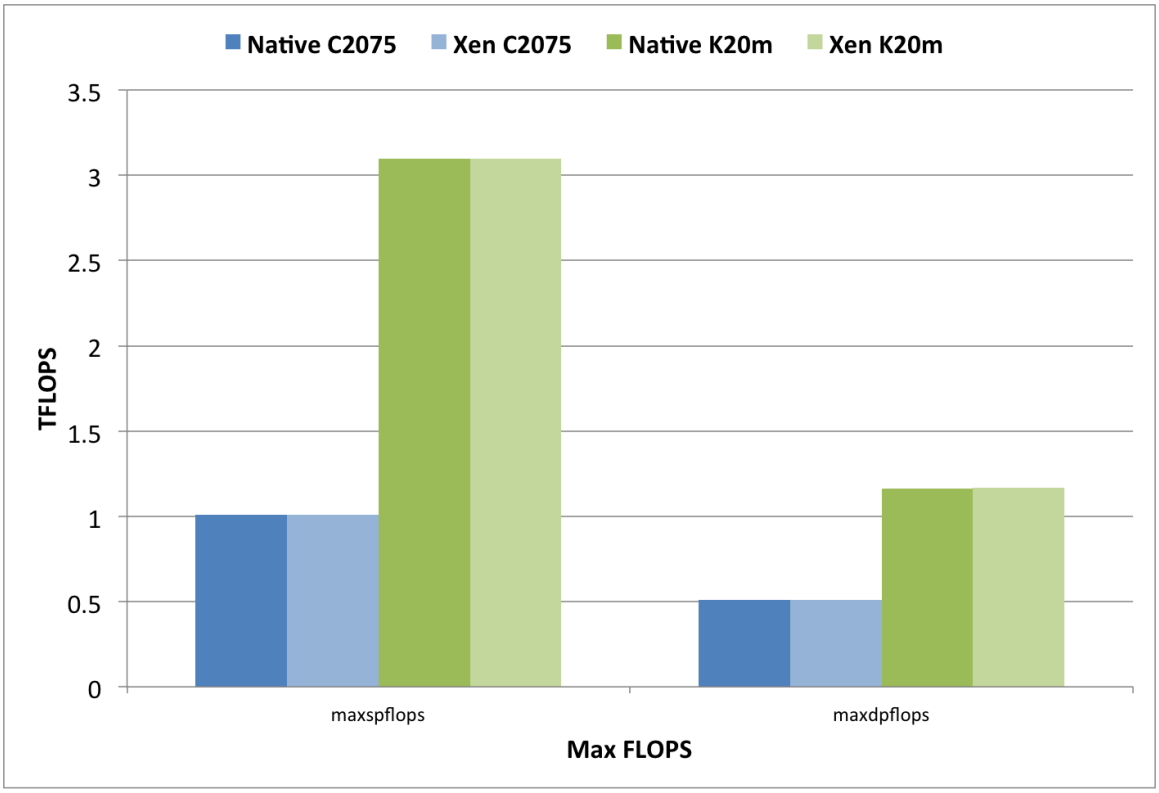

Figure 4.2 GPU Floating Point Operations per Second

Other FLOP-based benchmarks are used to emulate higher level applications. Stencil represents a 9-point stencil operation applied to a 2D data set, and the S3D benchmark is a computationally-intensive kernel from the S3D turbulent combustion simulation program [57]. In Figure 4.5, we see that both the Fermi C2075 and Kepler K20m GPUs performing well compared to the native base case, showing the overhead of virtualization is low. The C2075-enabled VMs experience slightly more overhead when compared to native performance again for pcie runs, but overhead is at most 7% for the S3D benchmark.

4.6.2 Device Speed

While floating point operations allow for the proposed solution to relate to many traditional HPC applications, they are just one facet of GPU performance within scientific

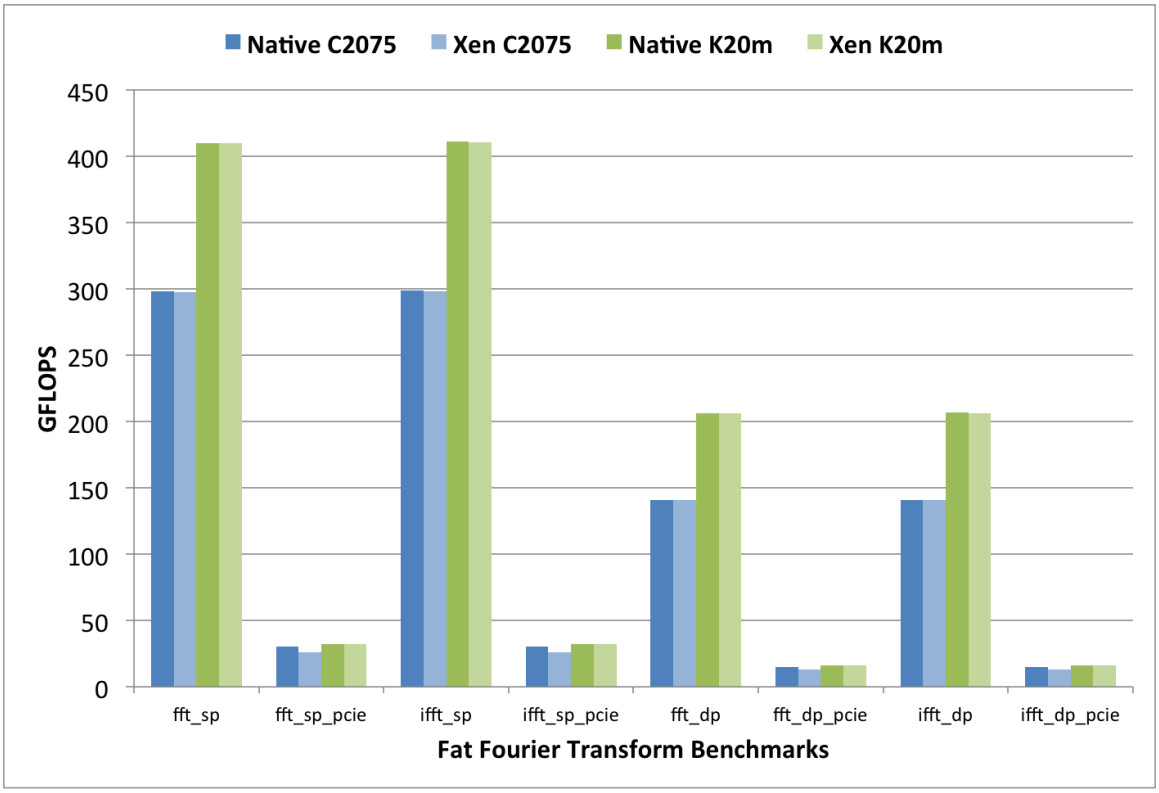

Figure 4.3 GPU Fast Fourier Transform

computing. Device speed, measured in both raw bandwidth and additional benchmarks, provides a different perspective towards evaluating GPU PCI passthrough in Xen. Figure 4.6 illustrates device level memory access of various GPU device memory structures. With both Nvidia GPUs, virtualization has little to no impact on the performance of inter-device memory bandwidth. As expected the Kepler K20m outperformed the C2075 VMs and there was a higher variance between runs with both native and VM cases. Molecular Dynamics and Reduction benchmarks in Figure 4.7 perform again at near-native performance without the pcie bus taken into account. However the overhead observed increases to 10-15% when the PCI-Express bus is considered when looking at the Fermi C2075 VMs.

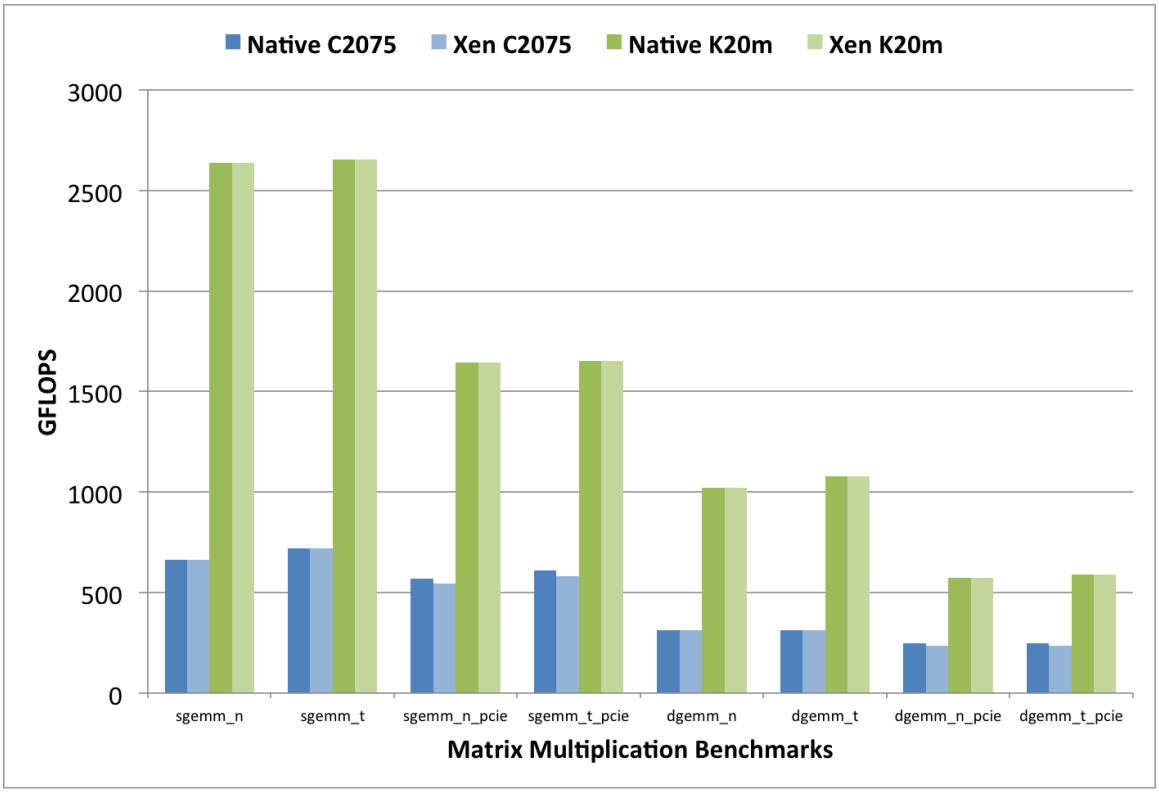

Figure 4.4 GPU Matrix Multiplication

4.6.3 PCI Express Bus

Upon evaluating PCI passthrough of GPUs, it is apparent that the PCI express bus is subject to the greatest potential for overhead, as was observed in the Fermi C2075 benchmarks. The VT-d and IOMMU chip instruction sets interface directly with the PCI bus to provide operational and security related mechanisms for each PCI device, thereby ensuring proper function in a multi-guest environment but potentially introducing some overhead. As such, it is imperative to investigate any and all overhead at the PCI Express bus.

Figure 4.8 looks at maximum PCI bus speeds for each experimental implementation. First, we see a consistent overhead in the Fermi C2075 VMs, with a 14.6% performance impact for download (to-device) and a 26.7% impact in readback (from-

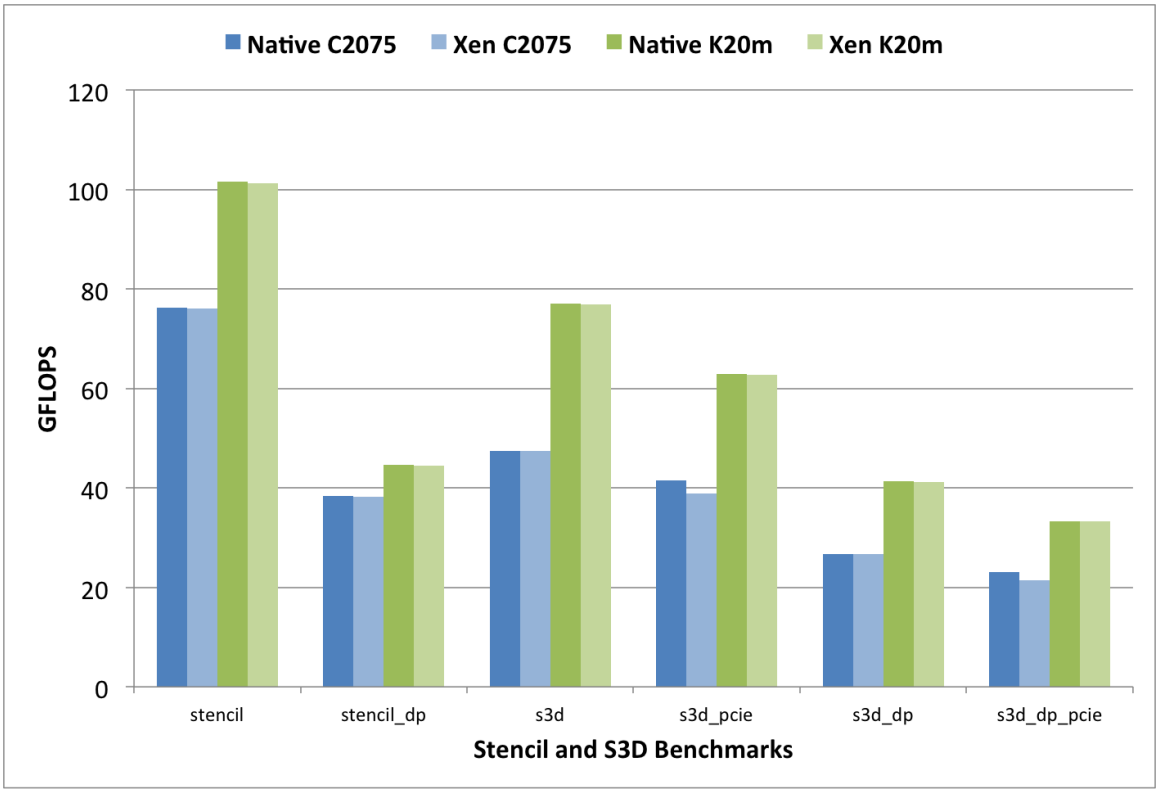

Figure 4.5 GPU Stencil and S3D

device). However, the Kepler K20 VMs do not experience the same overhead in the PCI-Express bus, and instead perform at near-native performance. These results indicate that the overhead is minimal in the actual VT-d mechanisms, and instead due to other factors.

Upon further examination, we have identified the performance degradation within the Fermi-based VM experimental setup is due to the testing environment's NUMA node configuration with the Westmere-based CPUs. With the Fermi nodes, the GPUs are connected through the PCI-Express bus from CPU Socket #1, yet the experimental GPU VM was run using CPU Socket #0. This meant that the VM's PCI-Express communication involved more host involvement with Socket #1, leading to an notable decrease in read and write speeds across the PCI-Express Bus. Such architectural limitations seem to be relieved in the next generation with a Sandy-Bridge CPU ar-

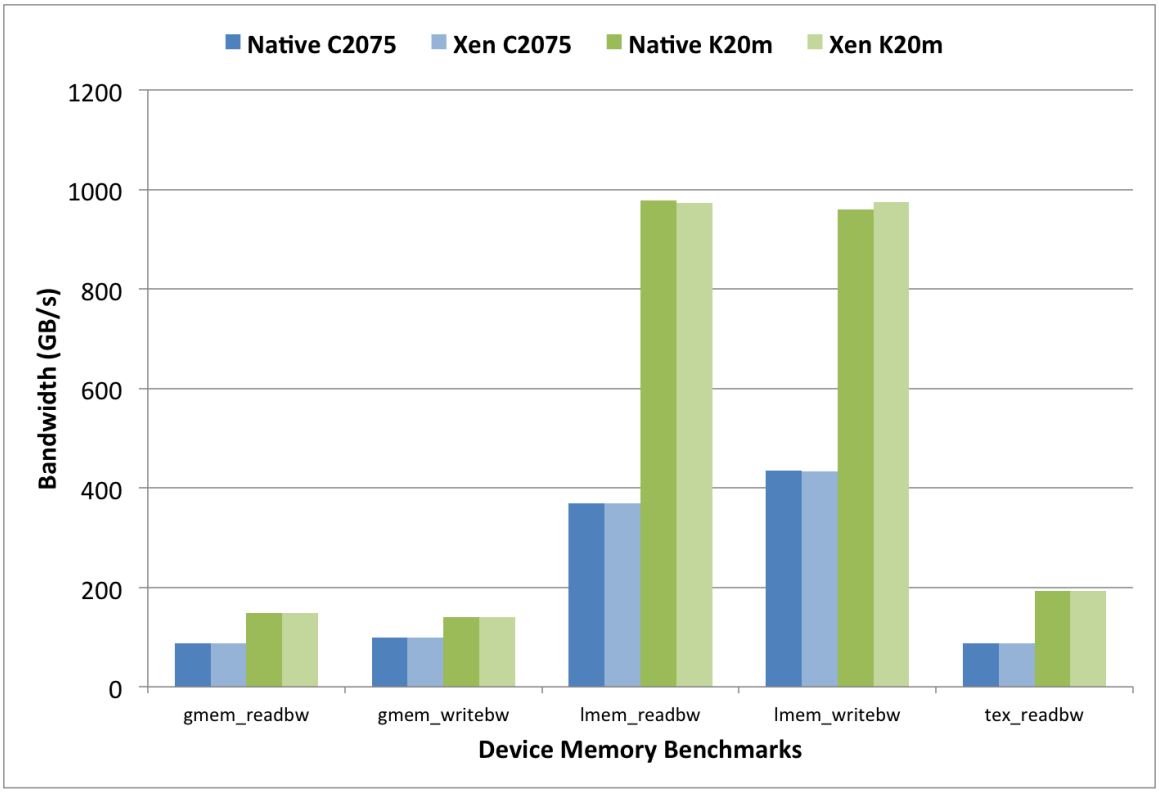

Figure 4.6 GPU Device Memory Bandwidth

chitecture and the Kepler K20m experimental setup. In summary, GPU PCI-Express performance in a VM is sensitive to the host CPU's NUMA architecture and care is needed to mitigate the impact, either by leveraging new architectures or by proper usage of Xen's VM core assignment features. Furthermore, the overhead in this system diminishes significantly when using the new Kepler GPUs by Nvidia.

4.7 Discussion

This manuscript evaluates the use of general purpose GPUs within cloud computing infrastructure, primarily targeted towards advanced scientific computing. The method of PCI passthrough of GPUs directly to a guest virtual machine running on a tuned Xen hypervisor shows initial promise for an ubiquitous solution in cloud

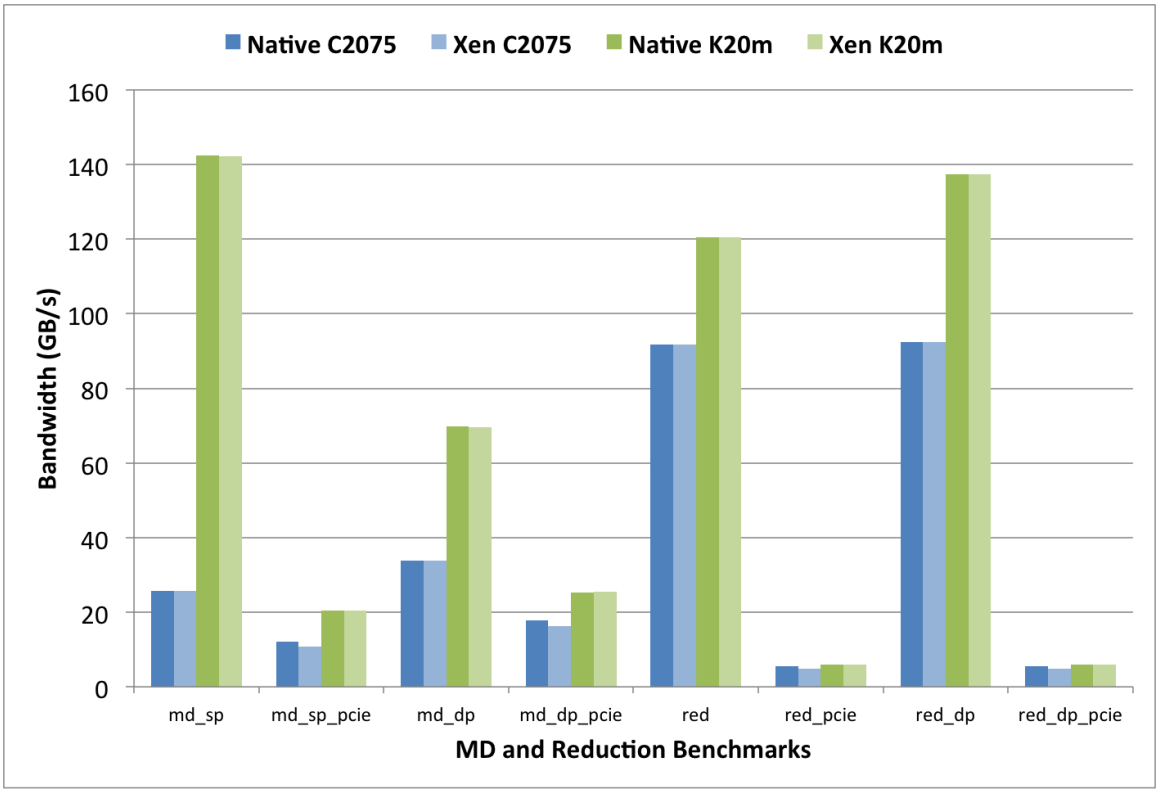

Figure 4.7 GPU Molecular Dynamics and Reduction

infrastructure. In evaluating the results in the previous section, a number of points become clear.

First, we can see that there is a small overhead in using PCI passthrough of GPUs within VMs, compared to native bare-metal usage, which represents the best possible use case. As with all abstraction layers, some overhead is usually inevitable as a necessary trade-off to added feature sets and improved usability. The same is true for GPUs within Xen VMs. The Kepler K20m GPU-enabled VMs operated at near-native performance for all runs, with a 1.2% reduction at worst in performance. The Fermi based C2075 VMs experience more overhead due to the NUMA configuration that impacted PCI-Express performance. However, when the overhead of the PCI-Express bus is not considered, the C2075 VMs perform at near-native speeds.

GPU PCI passthrough also has the potential to perform better than other front-

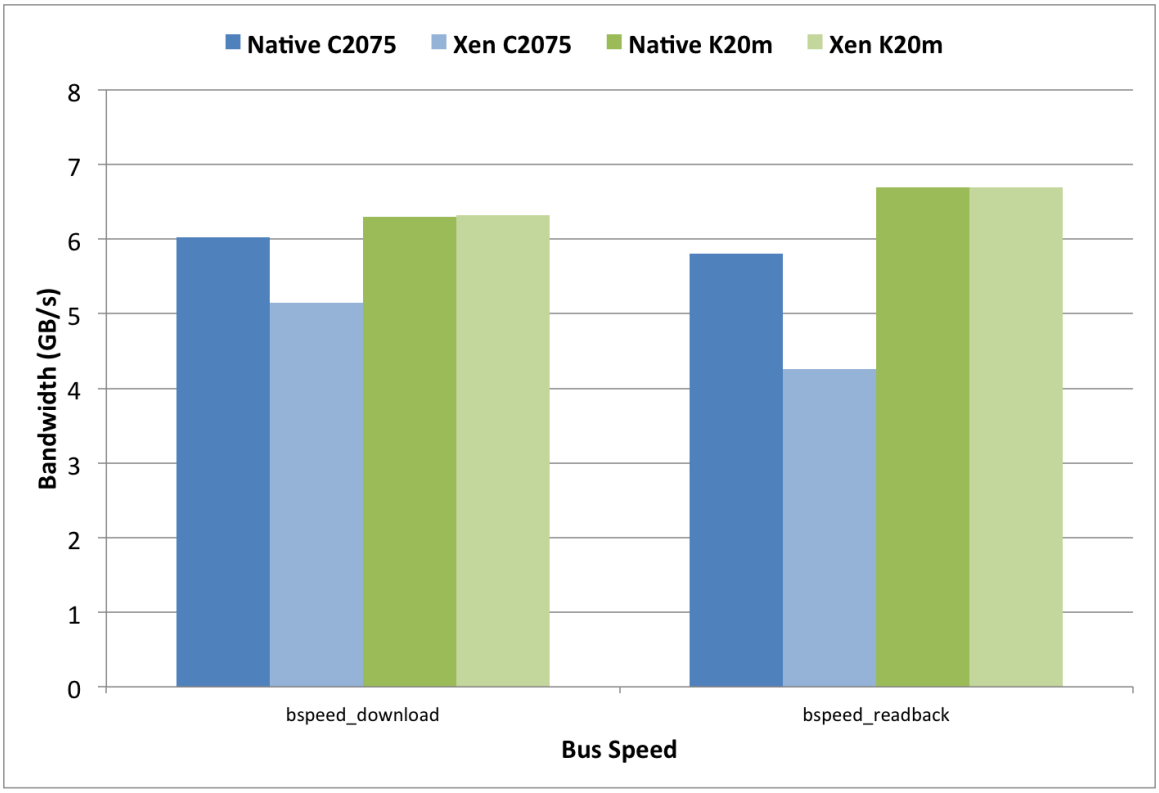

Figure 4.8 GPU PCI Express Bus Speed

end API solutions that are applicable within VMs, such as gVirtus or rCUDA. This is because such solutions are designed to communicate via a network interconnect such as 10Gb Ethernet or QDR InfiniBand [**?**], which introduces an inherent bottleneck. Even with the theoretically optimal configuration of rCUDA using QDR InfiniBand, the maximum theoretical bus speed is 40Gbs, which is comparably less than the measured 54.4Gps real-world performance measured between host-to-device transfers with GPU-enabled Kepler VMs.

Overall, it is our hypothesis that the overhead in using GPUs within Xen VMs as described in this manuscript will largely go unnoticed by most mid-level scientific computing applications. This is especially true when using the latest Sandy-Bridge CPUs with the Kepler series GPUs. We expect many mid-tier scientific computing groups to benefit the most from the ability to use GPUs in a scientific cloud infras-

tructure. Already this has been confirmed in [**?**], where similar a methodology has been leveraged specifically for Bioinformatics applications in the cloud.

4.8 Conclusion and Future Work

The ability to use GPUs within virtual machines represents a leap forward for supporting advanced scientific computing within cloud infrastructure. The method of direct PCI passthrough of Nvidia GPUs using the Xen hypervisor offers a clean, reproducible solution that can be implemented within many Infrastructure-as-a-Service (IaaS) deployments. Performance measurements indicate that the overhead of providing a GPU within Xen is minimal compared to the best-case native use, however NUMA inconsistencies can impact performance. The New Kepler-based GPUs operate with a much lower overhead, making those GPUs an ideal choice when designing a new GPU IaaS system.

Next steps for this work could involve providing GPU-based PCI passthrough within the OpenStack nova IaaS framework. This will enable research laboratories and institutions to create new private or national-scale cloud infrastructure that have the ability to support new scientific computing challenges. Other hypervisors could also leverage GPU PCI passthrough techniques and warrant in-depth evaluation in the future. Furthermore, we hope to integrate this work with advanced interconnects and other heterogeneous hardware and provide a parallel high performance cloud infrastructure to enable mid-tier scientific computing.

Chapter 5

GPU-Passthrough Performance: A Comparison of KVM, Xen, VMWare ESXi, and LXC for CUDA and OpenCL Applications

5.1 Introduction

As scientific workloads continue to demand increasing performance at greater power efficiency, high performance architectures have been driven towards heterogeneity and specialization. Intel's Xeon Phi, and GPUs from both NVIDIA and AMD represent some of the most common accelerators, with each capable of delivering improved performance and power efficiency over commodity multi-core CPUs.

Infrastructure-as-a-Service (IaaS) clouds have the potential to democratize access to the latest, fastest, and most powerful computational accelerators. This is true of both public and private clouds. Yet today's clouds are typically homogeneous

without access to even the most commonly used accelerators. Historically, enabling virtual machine access to GPUs and other PCIe devices has proven complex and error-prone, with only a small subset of GPUs being certified for use within a few commercial hypervisors. This is especially true for NVIDIA GPUs, likely the most popular for scientific computing, but whose drivers have always been closed source.

Given the complexity surrounding the choice of GPUs, host systems, and hypervisors, it is perhaps no surprise that Amazon is the only major cloud provider offering customers access to GPU-enabled instances. All of this is starting to change, however, as open source and other freely available hypervisors now provide sufficiently robust PCI passthrough functionality to enable GPU and other accelerator access whether in the public or private cloud.

Today, it is possible to access GPUs at high performance within all of the major hypervisors, merging many of the advantages of cloud computing (e.g. custom images, software defined networking, etc.) with the accessibility of on-demand accelerator hardware. Yet, no study to date has systematically compared the performance of PCI passthrough across all major cloud hypervisors. Instead, alternative solutions have been proposed that attempt to virtualize the GPU [58] , but sacrifice performance.

In this paper, we characterize the performance of both NVIDIA Fermi and Kepler GPUs operating in PCI passthrough mode in VMWare VSphere, Linux KVM, Xen, and Linux Containers (LXC). Through a series of microbenchmarks as well as scientific and Big Data applications, we make two contributions:

1. We demonstrate that PCI passthrough at high performance is possible for GPUs across 4 major hypervisors.

2. We describe the lessons learned through our performance analysis, as well as the relative advantages and disadvantages of each hypervisor for GPU support.

5.2 Related Work & Background

GPU virtualization and GPU-passthrough are used within a variety of contexts, from high performance computing to virtual desktop infrastructure. Accessing one or more GPUs within a virtual machine is typically accomplished by one of two strategies: 1) via API remoting with device emulation; or 2) using PCI passthrough.

5.2.1 GPU API Remoting

rCUDA, vCUDA, GViM, and gVirtuS are well-known API remoting solutionsFundamentally, these approaches operate similarly by splitting the driver into a front-end/back-end model, where calls into the interposed CUDA library (front-end) are sent via shared memory or a network interface to the back-end service that executes the CUDA call on behalf of the virtual machine. Notably, this technique is not limited to CUDA, but can be used to decouple OpenCL, OpenGL, and other APIs from their local GPU or accelerator.

The performance of API-remoting depends largely on the application and the remoting solution's implentation. Bandwidth and latency-sensitive benchmarks and applications will tend to expose performance bottlenecks more than compute-intensive applications. Moreover, solutions that rely on high speed networks, such as Infiniband, will compete with application-level networking for bandwidth.

5.2.2 PCI Passthrough

Input/Output Memory Management Units, or IOMMUs, play a fundamental roll in the PCI-passthrough virtualization mechanism. Like traditional MMUs that provide a virtual memory address space to CPUs [59], an IOMMU serves the fundamental purpose of connecting a direct memory access (DMA) capable I/O bus to main mem-

ory. The IOMMU unit, typically within the chipset, maps device virtual addresses to physical memory addresses. This process also has the added improvement of guaranteeing device isolation by blocking rogue DMA and interrupt requests [60], with a slight overhead, especially in early implementations [61].

Currently two major IOMMU implementations exist, VT-d and AMD-Vi by Intel and AMD, respectively. Both specifications provide DMA remapping to enable PCI-passthrough as well as other features such as interrupt remapping, hypervisor snooping, and security control mechanisms to ensure proper and efficient hardware utilization. PCI passthrough has been studied within the context of networking [62], storage [63], and other PCI-attached devices; however, GPUs have historically lagged behind other devices in their support for virtual machine passthrough.

5.2.3 GPU Passthrough, a Special Case of PCI Passthrough

While generic PCI passthrough can be used with IOMMU technologies to pass through many PCI-Express devices, GPUs represent a special case of PCI devices, and a special case of PCI passthrough. In traditional usage, GPUs usually serve as VGA devices primarily to render screen output, and while the GPUs used in this study do not render screen out, the function still exists in legacy. In GPU-passthrough, another VGA device (such as onboard graphics built into the motherboard, or a baseboard management controller) is necessary to serve as the primary display for the host, as well as providing emulated VGA devices for each guest VM. Most GPUs also have a video BIOS that requires full initialization and reset functions, which is often difficult due to the proprietary nature of the cards and their drivers.

Nevertheless, for applications that require native or near-native GPU performance across the full spectrum of applications with immediate access to the latest GPU drivers and compilers, GPU passthrough solutions are preferrable to API remoting.

Today, Citrix Xenserver, open source Xen [64], and VMWare ESXi [65], and most recently KVM all support GPU passthrough. To our knowledge, no one has systematically characterized the performance of GPU passthrough across a range of hypervisors, across such a breadth of benchmarks, and across multiple GPU generations as we do.

5.3 Experimental Methodology

5.3.1 Host and Hypervisor Configuration

We used two hardware systems, named Bespin and Delta, to evaluate four hypervisors. The Bespin system at USC/ISI represents Intel's Sandy Bridge microarchitecture with a Kepler class K20 GPU. The Delta system, provided by the FutureGrid project [66], represents the Westmere microarchitecture with a Fermi class C2075 GPU. Table 5.1 provides the major hardware characteristics of both systems. Note that in addition, both systems include 10 gigabit Ethernet, gigabit Ethernet, and either FDR or QDR Infiniband. Our experiments do not emphasize networking, and we use the gigabit ethernet network for management only.

A major design goal of these experiments was to reduce or eliminate NUMA effects (non-uniform memory access) on the PCI passthrough results in order to facilitate fair comparisons across hypervisors and to reduce experimental noise. To this end, we configured our virtual machines and containers to execute only on the NUMA node containing the GPU under test. We acknowledge that the NUMA effects on virtualization may be interesting in their own right, but they are not the subject of this set of experiments.

We use Bespin and Delta to evaluate three hypervisors and one container-based approach to GPU passthrough. The hypervisors and container system, VMWare

Table 5.1 Host hardware configurations

	Bespin	Delta
CPU (cores)	2x E5-2670 (16)	2x X5660 (12)
Clock Speed	2.6 GHz	2.6 GHz
RAM	48 GB	192 GB
NUMA Nodes	2	2
GPU	1x K20m	2x C2075

ESXi, Xen, KVM, and LXC, are summarized in Table 5.2. Note that each virtualization solution imposes its own unique requirements on the base operating system. Hence, Xen's Linux kernel refers to the Domain-0 kernel, whereas KVM's Linux kernel represents the actual running kernel hosting the KVM hypervisor. Linux Containers share a single kernel between the host and guests, and VMWare ESXi does not rely on a Linux kernel at all.

Similarly, hypervisor requirements prevented us from standardizing on a single host operating system. For Xen and KVM, we relied on the Arch Linux 2013.10.01 distribution because it provides easy access to the mainline Linux kernel. For our LXC tests, we use CentOS 6.4 because its shared kernel was identical to the base CentOS 6.4 kernel used in our testing. VMWare has a proprietary software stack. All of this makes comparison challenging, but as we describe in Section 5.3.2, we are running a common virtual machine across all experiments.

Table 5.2 Host Hypervisor/Container Configuration

Hypervisor	Linux Kernel	Linux Version
KVM	3.12	Arch 2013.10.01
Xen 4.3.0-7	3.12	Arch 2013.10.01
VMWare ESXi 5.5.0	N/A	N/A
LXC	2.6.32-358.23.2	CentOS 6.4

5.3.2 Guest Configuration

We treat each hypervisor as its own system, and compare virtual machine guests to a base CentOS 6.4 system. The base system and the guests are all composed of CentOS 6.4 installation with a 2.6.32-358.23.2 stock kernel and CUDA version 5.5. Each guest is allocated 20 GB of RAM and a full CPU socket (either 6 or 8 CPU cores). Bespin experiments received 8 cores and Delta experiments received 6 cores. VMs were restricted to a single NUMA node. On the Bespin system, the K20m GPU was attached to NUMA node 0. On the Delta system, the C2075 GPU was attached to NUMA node 1. Hence VMs ran on NUMA node 0 for the Bespin experiments, and node 1 for the Delta experiments.

5.3.3 Microbenchmarks

Our experiments are composed of a mix of microbenchmarks and application-level benchmarks, as well as a combination of CUDA and OpenCL benchmarks. The SHOC benchmark suite provides a series of microbenchmarks in both OpenCL and CUDA [67]. For this analysis, we focus on the OpenCL benchmarks in order to exercise multiple programming models. Benchmarks range from low-level peak Flops

and bandwidth measurements, to kernels and mini-applications.

5.3.4 Application Benchmarks

For our application benchmarks, we have chosen the LAMMPS molecular dynamics simulator [68], the GPU-LIBSVM [69], and the LULESH shock hydrodynamics simulator [70]. These represent a range of computational characteristics, from computational physics to big data analytics, and are representative of GPU-accelerated applications in common use.

LAMMPS The Large-scale Atomic/Molecular Parallel Simulator (LAMMPS), is a parallel molecular dynamics simulator [68, 71] used for production MD simulation on both CPUs and GPUs [72]. LAMMPS has two packages for GPU support, the USER-CUDA and GPU packages. With the USER-CUDA package, each GPU is used by a single CPU, whereas the GPU package allows multiple CPUs to take advantage of a single GPU. There are performance trade-offs with both approaches, but we chose to use the GPU package in order to stress the virtual machine by exercising multiple CPUs. Consistent with the existing GPU benchmarking approaches, our results are based on the Rhodopsin protein.

GPU-LIBSVM LIBSVM is a popular implementation [73] of the machine learning classification algorithm support vector machine (SVM). GPU-accelerated LIBSVM [69] enhances LIBSVM by providing GPU-implementations of the kernel matrix computation portion of the SVM algorithm for radial basis kernels. For benchmarking purposes we use the NIPS 2003 feature extraction gisette data set. This data set has a high dimensional feature space and large number of training instances, and these qualities are known to be computational intensive to generate SVM models.

Table 5.3 SHOC overheads expressed as geometric means of scaled values within a level, while maximum overheads are expressed as a percentage.

	Bespin (K20)								Delta (C2075)					
	KVM		Xen		LXC		VMWare		KVM		Xen		LXC	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
L0	0.999	1.57	0.997	3.34	1.00	1.77	1.00	1.90	1.01	0.031	0.969	12.7	1.00	0.07
L1	0.998	1.23	0.998	1.39	1.00	1.47	1.00	0.975	1.00	1.45	0.959	24.0	1.00	0.66
L2	0.998	0.48	0.995	0.846	0.999	1.90	0.999	1.20	1.00	0.101	0.982	4.60	1.00	0.01
	Bespin PCIe-only								Delta PCIe-only					
	KVM		Xen		LXC		VMWare		KVM		Xen		LXC	
	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
L0	0.997	0.317	0.999	0.143	0.995	0.981	0.995	1.01	1.04	0.029	0.889	12.7	1.00	0.01
L1	0.998	0.683	0.997	1.39	1.00	0.928	1.01	0.975	1.00	1.45	0.914	20.5	0.999	0.38
L2	0.998	0.478	0.996	0.846	1.00	0.247	1.01	0.133	1.00	0.075	0.918	4.60	1.00	N/A

The GPU-accelerated SVM implementation shows dramatic improvement over the CPU-only implementation.

LULESH Hydrodynamics is widely used to model continuum properties and interactions in materials when there is an applied force [74]. Hydrodynamics applications consume approximately one third of the runtime of data center resource throughout the U.S. DoD (Department of Defense). The Livermore Unstructured Lagrange Explicit Shock Hydro (LULESH) was developed by Lawrence Livermore National Lab as one of five challenge problems in the DARPA UHPC program. LULESH is widely used as a proxy application in the U.S. DOE (Department of Energy) co-design effort for exascale applications [70].

5.4 Performance Results

We characterize GPGPU performance within virtual machines across two hardware systems, 4 hypervisors, and 3 application sets. We begin with the SHOC benchmark suite before describing the GPU-LIBSVM, LAMMPS, and LULESH results. All benchmarks are run 20 times and averaged. Results are scaled with respect to a base CentOS 6.4 system for both systems. That is, we compare virtualized Bespin performance to non-virtualized Bespin performance, and virtualized Delta performance to non-virtualized Delta performance. Values less than 1 indicate that the base system outperformed the virtual machine, while values greater than 1 indicate that the virtual machine outperformed the base system. In cases where we present geometric means across multiple benchmarks, the means are taken over these scaled values, and the semantics are the same: less than 1 indicates overhead in the hypervisor, greater than 1 indicates a performance increase over the base system.

5.4.1 SHOC Benchmark Performance

SHOC splits its benchmarks into 3 Levels, named Levels 0 through 2. Level 0 represents device-level characteristics, peak Flop/s, bandwidth, etc. Level 1 characterizes computational kernels: FFT and matrix multiplication, among others. Finally, Level 2 includes "mini-applications," in this case an implementation of the S3D, a computational chemistry application.

Because the SHOC OpenCL benchmarks report more than 70 individual microbenchmarks, space does not allow us to show each benchmark individually. Instead, we start with a broad overview of SHOC's performance across all benchmarks, hypervisors, and systems. We then discuss in more detail those benchmarks that either outperformed or underperformed the Bespin (K20) system by 0.50% or more.

We call these benchmarks outliers. As we will show, those outlier benchmarks identified on the Bespin system, also tend to exhibit comparable characteristics on the Delta system as well, but the overhead is typically higher.

In Table 5.3, we provide geometric means for each SHOC level across each hypervisor and system. We also include the maximum overhead for each hypervisor at each level to facilitate comparison across hypervisors and systems. Finally, we provide a comparable breakdown of only the PCIe SHOC benchmarks, based on the intuition that PCIe-specific benchmarks will likely result in higher overhead.

At a high level, we immediately notice that in the cases of KVM and LXC, both perform very near native across both the Bespin and Delta platforms. On average, these systems are almost indistinguishable from their non-virtualized base systems. So much so, that experimental noise occasionally boosts performance slightly above their base systems.

This is in sharp contrast to the Xen and VMWare hypervisors, which perform well on the Bespin system, but poorly on the Delta system in some cases. This is particularly evident when looking at the maximum overheads for Xen and VMWare across both systems. In this case, we see that on the Bespin system, Xen's maximum overhead of 3.34% is dwarfed by Delta's maximum Xen overhead of 24.0%. VMWare exhibits similar characteristics, resulting in a maximum overhead of 1.9% in the case of the Bespin system, and a surprising 36.6% in the case of the Delta system. We provide a more in-depth discussion of these overheads below.

Of the Level 0 benchmarks, only four exhibited notable overhead in the case of the Bespin system: bspeed_download, lmem_readbw, tex_readbw, and ocl_queue. These are shown in Figure 5.1. These benchmarks represent device-level characteristics, including host-to-device bandwidth, onboard memory reading, and OpenCL kernel queue delay. Of the four, only bspeed_download incurs a statistically significant

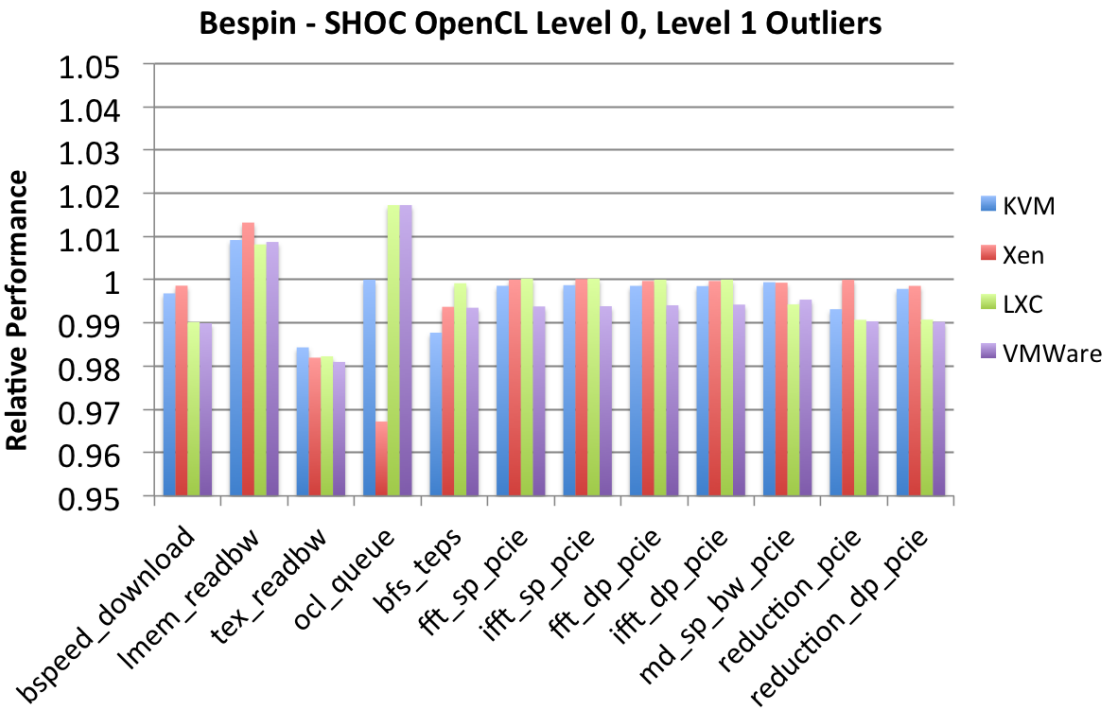

Figure 5.1 SHOC Levels 0 and 1 relative performance on Bespin system. Results show benchmarks over or under-performing by 0.5% or greater. Higher is better.

overhead. The remainder perform within one standard deviation of the base, despite an overhead of greater than 0.5%.

bspeed_download is representative of the most likely source of virtualization overhead, data movement across the PCI-Express bus. The PCIe lies at the boundary of the virtual machine and the physical GPU, and requires interrupt remapping, IOMMU interaction, etc. in order to enable GPU passthrough into the virtual machine. Despite this, in Figure 5.1 we see a maximum of 1% overhead for bspeed_download in VMWare, and less than 0.5% overhead for both KVM and Xen.

The remainder of Figure 5.1 includes a series of SHOC's Level 1 benchmarks, representing computational kernels. This includes BFS, FFT, molecular dynamics, and reduction kernels. Notably, nearly all of the benchmarks exhibiting overhead are the

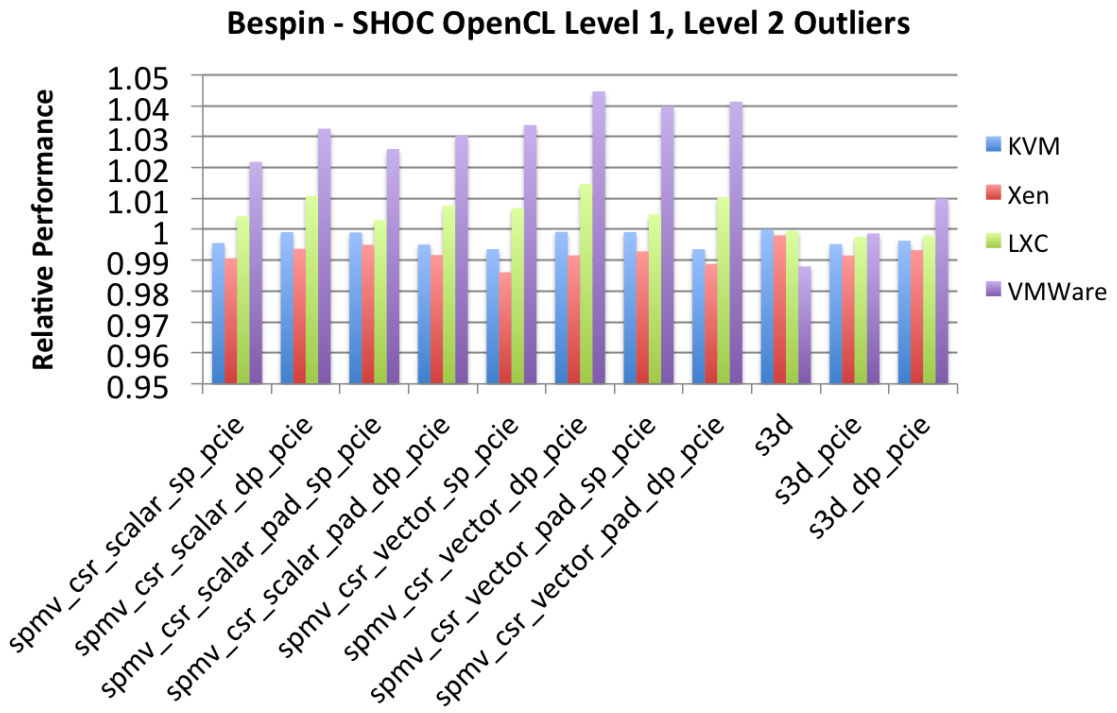

Figure 5.2 SHOC Levels 1 and 2 relative performance on Bespin system. Results show benchmarks over or under-performing by 0.5% or greater. Higher is better.

PCIe portion of SHOC's benchmarks. This is unsurprising, since the Level 0 benchmarks suggest PCIe bandwidth as the major source overhead. Still, results remain consistent with the bspeed_download overhead observed in the Level 0 benchmarks, further suggesting that host/device data movement is the major source of overhead.

In Figure 5.2 we present the remaining SHOC Level 1 results as well as the SHOC Level 2 results (S3D). In these results we begin to see an interesting trend, namely that VMWare consistently outperforms the base system in the Spmv and S3D microbenchmarks on the Bespin system. We believe this to be a performance regression in CentOS 6.4, rather than a unique improvement due to the VMWare ESXi hypervisor. When running the benchmarks on a non-virtualized Bespin system with Arch Linux, the VMWare ESXi performance gains were erased. An interesting finding of

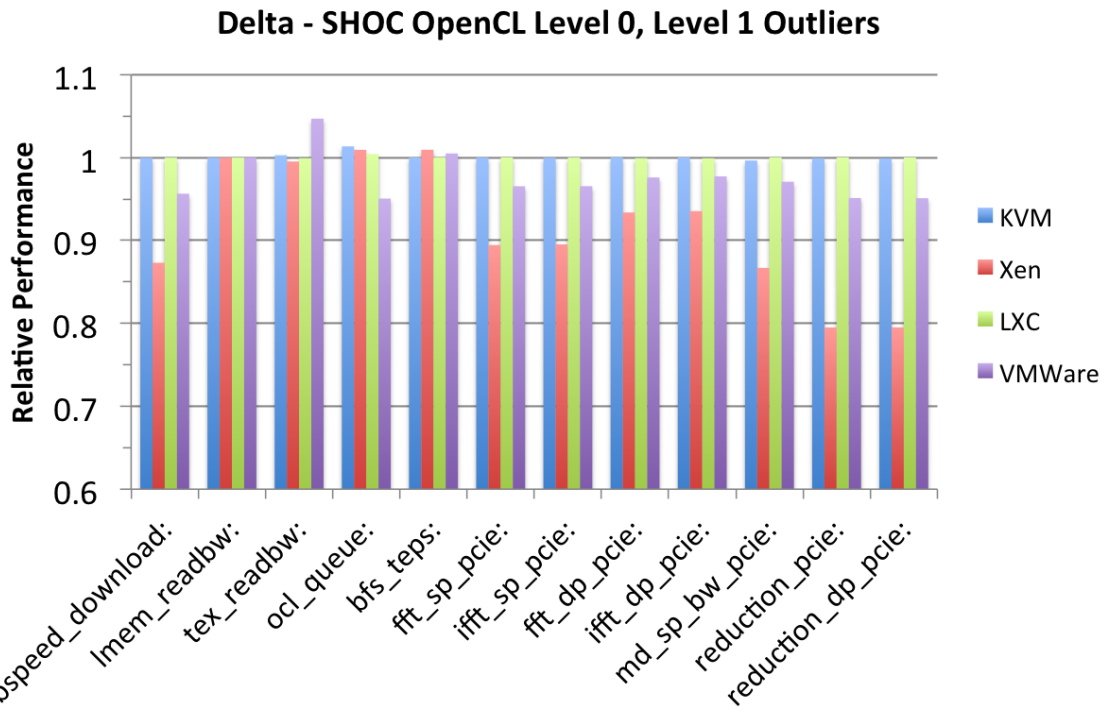

Figure 5.3 SHOC Levels 0 and 1 relative performance on Delta system. Benchmarks shown are the same as Bespin's. Higher is better.

this was that Spmv was unique in this way - no other benchmarks were affected by this performance issue.

Turning to the Delta system, in Figures 5.3 and **??**, we show the same benchmarks for the Delta system as was shown in Figures 5.1 and 5.2. Again, we find that the same benchmarks are responsible for most of the overhead on the Delta system. This is unsurprising, since PCIe was shown to be the source of the bulk of the overhead. A major difference in the case of the Delta system, however, is the amount of overhead. While the Bespin system saw overheads of approximately 1%, Delta's overhead routinely jumps above 35%, especially in the case of the Spmv benchmark for VMWare.

On further examination, we determined that Xen was unable to activate IOMMU large page tables on the Delta system. KVM successfully allocated 4k, 2M, and 1G

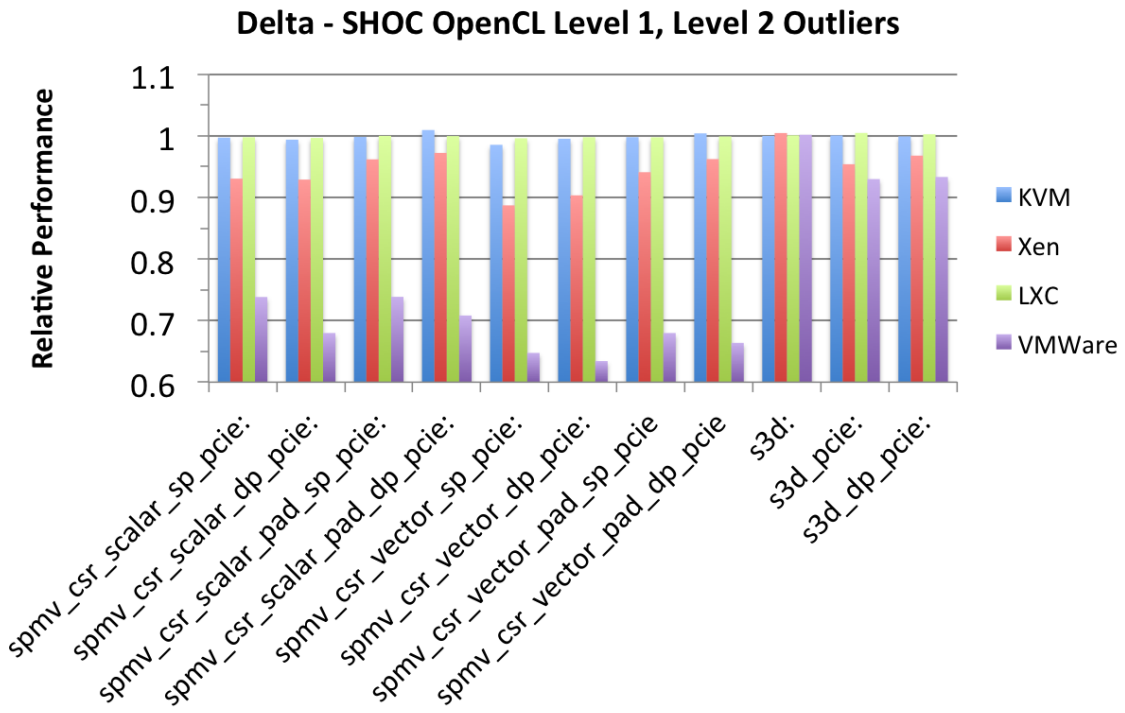

Figure 5.4 SHOC Levels 1 and 2 relative performance. Benchmarks shown are the same as Bespin's. Higher is better.

page table sizes, while Xen was limited to size 4k page tables. The Bespin system was able to take advantage of 4k, 2M, and 1G page sizes on both KVM and Xen. It appears that this issue is correctable and does not represent a fundamental limitation to the Xen hypervisor on the Nehalem/Westmere microarchitecture. While as a closed source product, we have limited insight into VMWare ESXi, we speculate that VMWare may be experiencing a similar issue on the Delta system and not on our Bespin system.

In light of this, we broadly find that virtualization overhead across hypervisors and architectures is minimal. Questions remain as to the source of the exceptionally high overhead in the case of Xen and VMWare on the Delta system, but because KVM shows no evidence of this overhead, we believe the Westmere/Fermi architecture to be suitable for VGA passthrough in a cloud environment. In the case of the Bespin

system, it is clear that VGA passthrough can be achieved across hypervisors with virtually no overhead.

A surprising finding is that LXC showed little performance advantage over KVM, Xen, and VWMare. While we expected near-native performance from LXC we did not expect the hardware-assisted hypervisors to achieve such high performance. Still, LXC carries some advantages. In general, its maximum overheads are comparable to or less than KVM, Xen, and VMWare, especially in the case of the Delta system.

5.4.2 GPU-LIBSVM Performance

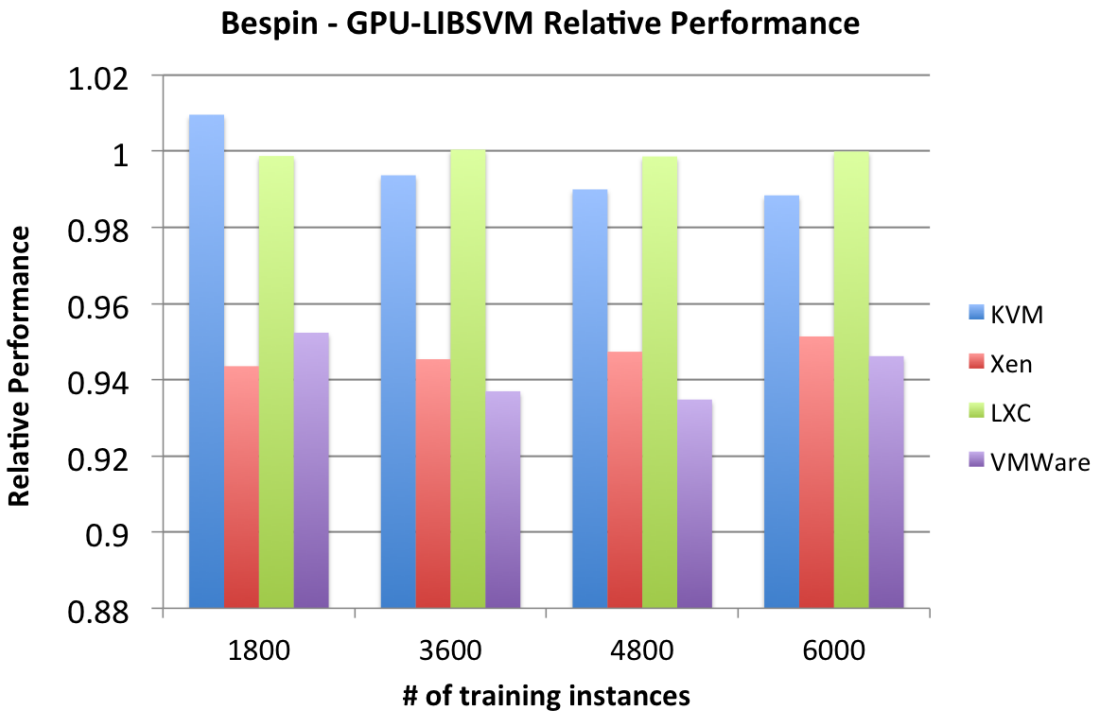

Figure 5.5 GPU-LIBSVM relative performance on Bespin system. Higher is better.

In Figures 5.5 and 5.6, we display our GPU-LIBSVM performance results for the Bespin (K20m) and Delta (C2075) systems. Using the NIPS 2003 gisette data set, we show the performance across 4 problems sizes, ranging from 1800 to 6000 training

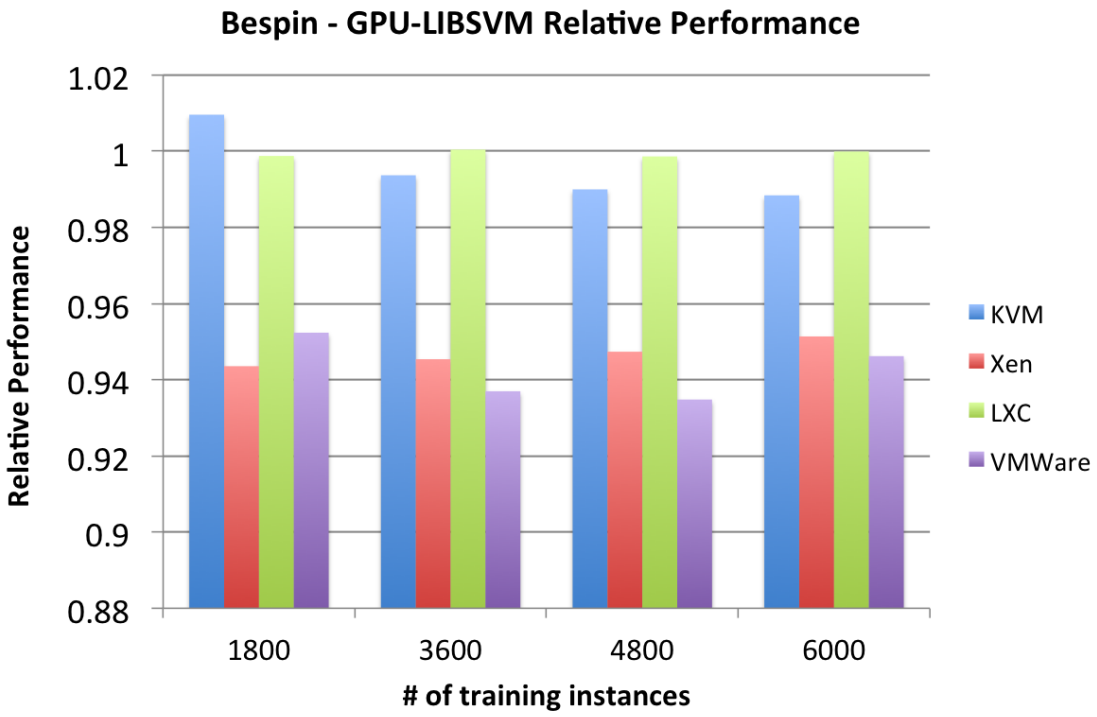

Figure 5.6 GPU-LIBSVM relative performance on Delta showing improved performance within virtual environments. Higher is better.

instances. The NIPS 2003 gisette dataset is a standard dataset that was used as a part of the NIPS 2003 feature selection challenge.

KVM again performs well across both the Delta and Bespin systems. In the case of the Delta system, in fact, KVM, significantly outperforms the base system. We determined this to be caused by KVM's support for transparent hugepages. When sufficient memory is available, transparent hugepages may be used to back the entirety of the guest VM's memory. Hugepages have previously been shown to improve TLB performance for KVM guests, and have been shown to occasionally boost the performance of a KVM guests beyond its host [75]. After disabling hugepages on the KVM host for the Delta system, performance dropped to 80–87% of the base system, suggesting that memory optimizations such as transparent hugepages can substantially improve the performance of virtualized guests under some circumstances. LXC

and VMWare perform close to the base system, while Xen achieves between 72–90% of the base system's performance. We speculate that this may be related to Xen's inability to enabled page sizes larger than 4k.

5.4.3 LAMMPS Performance

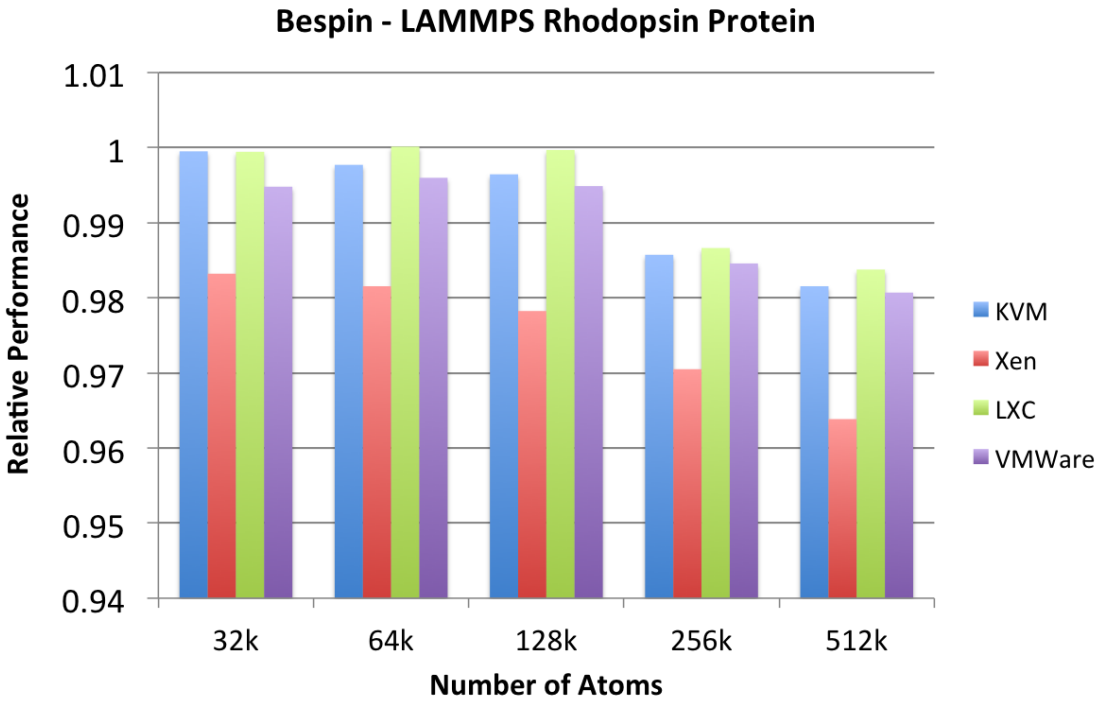

Figure 5.7 LAMMPS Rhodopsin benchmark relative performance for Bespin system. Higher is better.

In Figures 5.7 and 5.8, we show the LAMMPS Rhodopsin protein simulation results. LAMMPS is unique among our benchmarks, in that it exercises both the GPU and multiple CPU cores. In keeping with the LAMMPS benchmarking methodology, we execute the benchmark using 1 GPU and 1–8 CPU cores on the Bespin system, selecting the highest performing configuration. In the case of the Delta system, we execute the benchmark on 1–6 cores and 1 GPU, selecting the highest performing configuration.

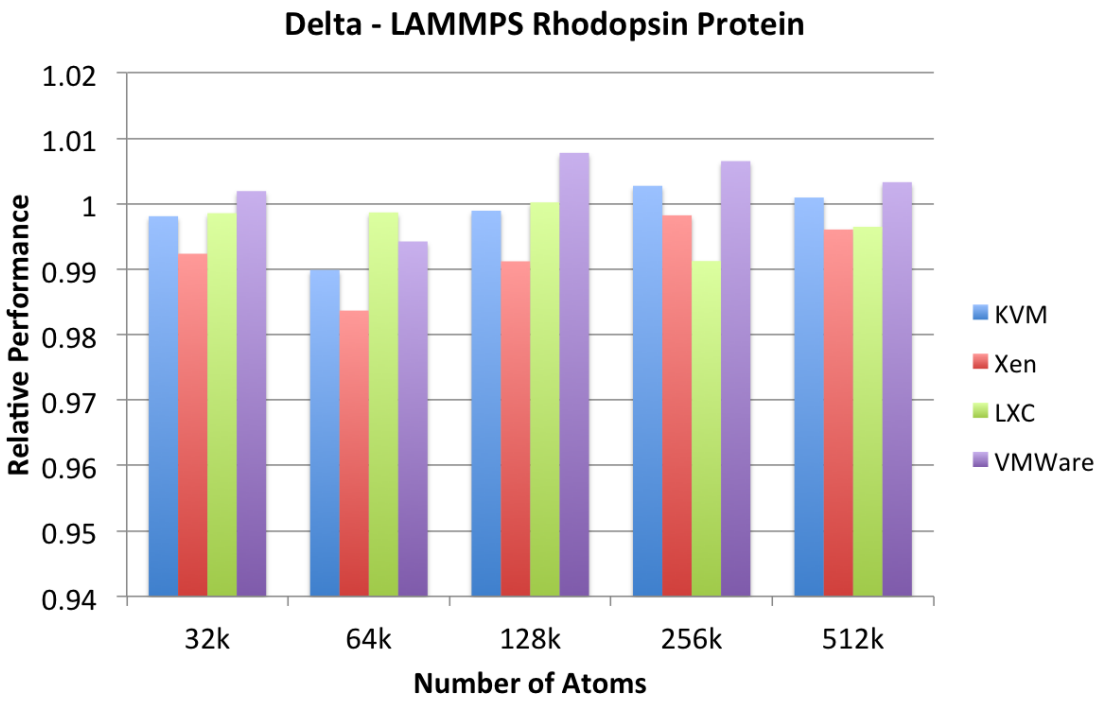

Figure 5.8 LAMMPS Rhodopsin benchmark relative performance for Delta system. Higher is better.

Overall, LAMMPS performs well across both hypervisors and systems. Surprisingly, LAMMPS showed better efficiency on the Delta system than the Bespin system, achieving greater than 98% efficiency across the board, while Xen on the Bespin system occasionally drops as low as 96.5% efficiency.

This performance is encouraging because it suggests that even heterogeneous CPU + GPU code is capable of performing well in a virtualized environment. Unlike SHOC, GPU-LIBSVM, and LULESH, LAMMPS fully exercises multiple host CPUs, splitting work between one or more GPUs and one or more CPU cores. This has the potential to introduce additional performance overhead, but the results do not bear this out in the case of LAMMPS.

5.4.4 LULESH Performance

In Figure 5.9 we present our LULESH results for problem sizes ranging from mesh resolutions of $N = 30$ to $N = 150$. LULESH is a highly compute-intensive simulation, with limited data movement between the host/virtual machine and the GPU, making it ideal for GPU acceleration. Consequently, we would expect little overhead due to virtualization. We show LULESH results only on the Bespin system, because differences in the code bases between the Kepler and Fermi implementations led to unsound comparisons.

While, overall, we see very little overhead, there is a slight scaling effect that is most apparent in the case of the Xen hypervisor. As the mesh resolution (N^3) increases from $N = 30$ to $N = 150$, we see that the Xen overhead decreases until Xen performs on-par with KVM, LXC, and VMWare.

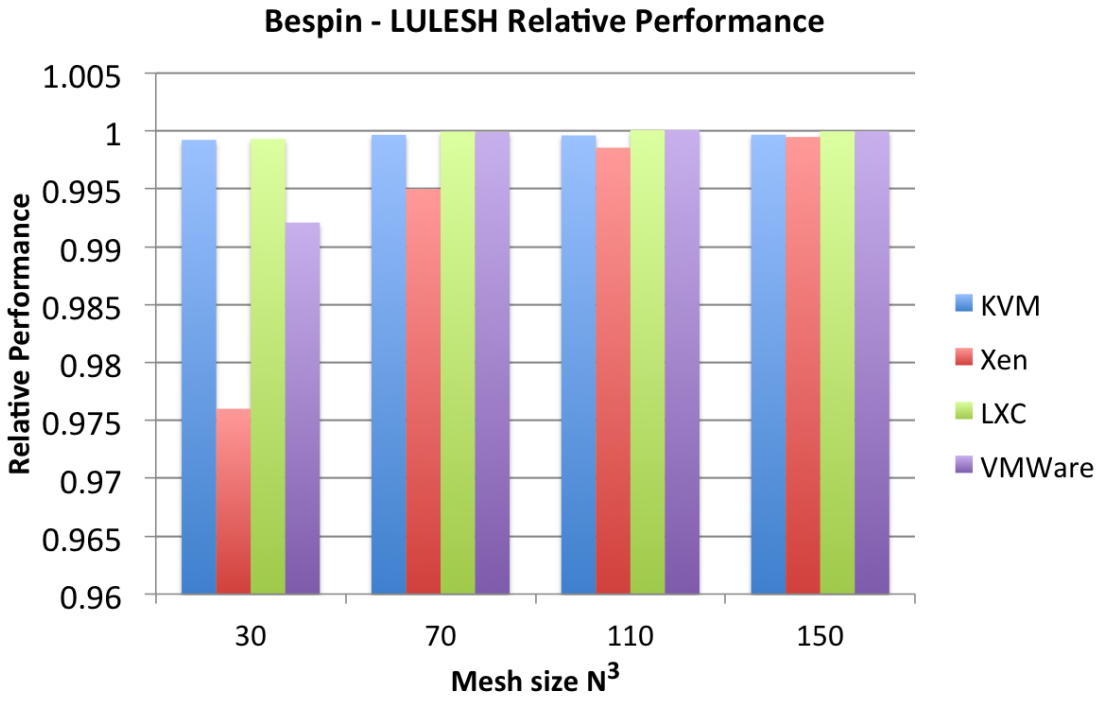

Figure 5.9 LULESH relative performance on Bespin. Higher is better.

5.5 Lessons Learned

Virtualizing performance-critical workloads has always proven controversial, whether the workload is CPU-only [49] or CPU with GPU. From our Westmere results, we can see that this criticism is in part legitimate, at times resulting in a performance penalty of greater than 10%, especially in the case of the VMWare ESXi and Xen hypervisors. We believe much of this to be fixable, especially in the case of the Xen hypervisor.

At the same time, however, we have shown that the Sandy Bridge processor generation has nearly erased those performance overheads, suggesting that old arguments against virtualization for performance-critical tasks should be reconsidered. In light of this, the primary lesson from this study is that VGA-passthrough to virtual machines is achievable at low overhead, and across a variety of hypervisors and virtualization platforms. Virtualization performance remains inconsistent across hypervisors for the Westmere generation of processors, but starting with the Sandy Bridge architecture, performance and consistency increase dramatically. In the case of the Sandy Bridge architecture, even the lowest performing hypervisor, open source Xen, typically performs within 95% of the base case.

This study has also yielded valuable insight into the merits of each hypervisor. KVM consistently yielded near-native performance across the full range of benchmarks. Its support for transparent hugepages resulted in slight performance boosts over-and-above even the base CentOS system in the case of the Delta system.

VMWare's performance proved inconsistent across architectures, performing well in the case of Bespin, and relatively poorly in the case of the Delta system. Because hypervisor configurations were identical across systems, we can only speculate that VMWare's performance is aided by the virtualization improvements offered by the

Sandy Bridge microarchitecture.

The Xen hypervisor was consistently average across both architectures, performing neither poorly nor extraordinarily well in any individual benchmark. Xen and VMWare ESXi are the only two hypervisors from this study that officially support VGA passthrough. As a result, PCI passthrough support in both Xen and VMWare is more robust than KVM. We expect that this advantage will not last long, as commercial solutions targeting PCI passthrough in KVM are becoming common, particularly with regard to SR-IOV and networking adapters.

Linux Containers (LXC), consistently performed closest to the native case. This, of course, is not surprising given that LXC guests share a single kernel with their hosts. This performance comes at the cost of both flexibility and security, however. LXC is less flexible than its full virtualization counterparts, offering support for only Linux guests. More importantly, LXC device passthrough has security implications for multi-GPU systems. In the case of a multi-GPU-enabled host with NVIDIA hardware, both GPUs must be passed to the LXC guest in order to initialize the driver. This limitation may be addressable in future revisions to the NVIDIA driver.

5.6 Directions for Future Work

In this paper we have characterized the performance of 4 common hypervisors across two generations of GPUs and two host microarchitectures, and across 3 sets of benchmarks. We showed the dramatic improvement in virtualization performance between the Fermi/Westmere and the Kepler/Sandy Bridge system, with the Sandy Bridge system typically performing within 1% of the base system. Finally, this study sought to characterize the GPU and CPU+GPU performance with carefully tuned hypervisor and guest configurations, especially with respect to NUMA. Improvements must

be made to today's hypervisors in order to improve virtual NUMA support. Finally, cloud infrastructure, such as OpenStack, must be capable of automatically allocating virtual machines in a NUMA-friendly manner in order to achieve acceptable results at cloud-scale.

The next step in this work is to move beyond the single node to show that clusters of accelerators can be efficiently used with minimal overhead. This will require studies in high speed networking, particularly SR-IOV-enabled ethernet and Infiniband. Special attention is needed to ensure that latencies remain tolerable within virtual environments. Some studies have begun to examine these issues [76], but open questions remain.

Chapter 6

Supporting High Performance Molecular Dynamics in Virtualized Clusters using IOMMU, SR-IOV, and GPUDirect

6.1 Introduction

At present we stand at the inevitable intersection between High Performance Computing (HPC) and clouds. Various platform tools such as Hadoop and MapReduce, among others, have already percolated into data intensive computing within HPC [77]. In addition, there are efforts to support traditional HPC-centric scientific computing applications in virtualized cloud infrastructure. There are a multitude of reasons for supporting parallel computation in the cloud [19], including features such as dynamic scalability, specialized operating environments, simple management interfaces, fault tolerance, and enhanced quality of service, to name a few. The growing importance

of supporting advanced scientific computing using virtualized infrastructure can be seen by a variety of new efforts, including the NSF-funded Comet resource part of XSEDE at San Diego Supercomputer Center [78].

Nevertheless, there exists a past notion that virtualization used in today's cloud infrastructure is inherently inefficient. Historically, cloud infrastructure has also done little to provide the necessary advanced hardware capabilities that have become almost mandatory in supercomputers today, most notably advanced GPUs and high-speed, low-latency interconnects. The result of these notions has hindered the use of virtualized environments for parallel computation, where performance must be paramount.

A growing effort is currently underway that looks to systematically identify and reduce any overhead in virtualization technologies. This effort has, thus far, proven to be a qualified success [49, 79], though further research is needed to address issues of scalability and I/O. Thus, we see a constantly diminishing overhead with virtualization, not only with traditional cloud workloads [80] but also with HPC workloads. While virtualization will almost always include some additional overhead in relation to its dynamic features, the eventual goal for supporting HPC in virtualized environments is to minimize what overhead exists whenever possible. To advance the placement of HPC applications on virtual machines, new efforts are emerging which focus specifically on key hardware now commonplace in supercomputers. By leveraging new virtualization tools such as IOMMU device passthrough and SR-IOV, we can now support the such advanced hardware as the latest Nvidia Tesla GPUs [81] as well as InfiniBand fabrics for high performance networking and I/O [82, 83].

With the advances in hypervisor performance coupled with the newfound availability of HPC hardware in virtual machines analogous to the most powerful supercomputers used today, we see can see the possibility of a high performance cloud in-

frastructure using virtualization. While our previous efforts in this area have focused on single-node advancements, it is now imperative to ensure real-world applications can also operate in distributed environments as found in today's cluster and cloud infrastructures.

Efforts to improve power efficiency and performance in data centers has led to more heterogeneous architectures. That move toward heterogeneity has, in turn, led to support for heterogeneity in the cloud. For example, Amazon EC2 supports GPU accelerators in EC2 [84], and OpenStack supports heterogeneity using flavors [85]. These advancements in cloud-level support for heterogeneity combined with better support for high-performance virtualization makes the use of cloud for HPC much more feasible for a wider range of applications and platforms.

In this paper we discribe background a related work. Then, we describe a heterogeneous cloud platform, based on OpenStack. This effort has been under development at USC/ISI since 2011 [43]. We describe our work towards integrating GPU and InfiniBand support into OpenStack, and we describe the heterogeneous scheduling additions that are necessary to support not only attached accelerators, but any cloud composed of heterogeneous elements.

We then demonstrate running two molecular dynamics simulations, LAMMPS and HOOMD, in a virtual infrastructure complete with both Kepler GPUs and QDR InfiniBand. Both HOOMD and LAMMPS are used extensively in some of the world's fastest supercomputers and represent example simulations that HPC supports today. We show that these applications are able to run at near-native speeds within a completely virtualized environment, demonstrating just small performance impacts that are usually acceptable by many users. Furthermore, we demonstrate the ability of such a virtualized environment to support cutting edge software tools such as RDMA GPUDirect, illustrating how cutting-edge HPC technologies are also possible in a

virtualized environment.

Following these efforts, we hope to ensure upstream infrastructure projects such as OpenStack [**?**, 86] are able to make effective and quick use of these features, allowing users to build private cloud infrastructure to support high performance distributed computational workloads.

6.2 Background and Related Work

Virtualization technologies and hypervisors have been seen widespread deployment in support of a vast array of applications. This ranges from public commercial Cloud deployments such as Amazon EC2 [87, 88], Microsoft Azure [89], and Google's Cloud Platform [90] to private deployments within colocation facilities, corporate data centers, and even national scale cyber infrastructure initiatives. All these support look to support various use cases and applications such as web servers, ACID and BASE databases, online object storage, and even distributed systems, to name a few.

The use of virtualization and hypervisors specifically support various HPC solutions has been studied with mixed results. In [49], it is found that there is a great deal of variance between hypervisors when running various distributed memory and MPI applications, finding that KVM overall performed well across an array of HPC benchmarks. Furthermore, some applications may not may fit well into default virtualized environments, such as High Performance Linpack [79]. Other studies have specifically looked at interconnect performance in virtualization and found the best-case scenario to be lacking [91] with up to 60% performance penalties with conventional techniques.

Recently, various CPU architectures have added support for I/O virtualization mechanisms in the CPU ISA through the use of an I/O memory management unit (IOMMU). Often, this is referred to as PCI Passthrough, as it enabled devices on

the PCI-Express bus to be passed directly to a specific virtual machine (VM). Specific hardware implementations include Intel's VT-d [92], AMD's IOMMU [93] from x86_64 architectures, and even more recently ARM System MMU [94]. All of these implementations effectively look to aid in the usage of DMA-capable hardware to be used within a specific virtual machine. Using these features, a wide array of hardware can be utilized directly within VMs and enable fast and efficient computation and I/O capabilities.

With PCI Passthrough, a PCI device is handed directly to a running (or booting) VM, thereby relinquishing control of the device within the host entirely. This is different from typical VM usage where hardware is emulated in the host and used in a guest VM, such as with bridged ethernet adapters or emulated VGA devices. Performing PCI Passthrough requires the host to seize the device upon boot using a specialized driver to effectively block normal driver initialization. In the instance of the KVM hypervisor, this is done using the *vfio* and *pci_stub* drivers. Then, this driver relinquishes control to the VM, whereby normal device drivers initiate the hardware and enable the device for use by the guest OS.

6.2.1 GPU Passthrough

Nvidia GPUs comprise the single most common accelerator in the Nov 2014 Top 500 List [27] and represent an increasing shift towards accelerators for HPC applications. Historically, GPU usage in a virtualized environment has been difficult, especially for scientific computation. Various front-end remote API implementations have been developed to provide CUDA and OpenCL libraries in VMs, which translate library calls to a back-end or remote GPU. One common use case of this is rCUDA [55], which provides a front-end CUDA API within a VM or any compute node, and then sends the calls via Ethernet or InfiniBand to a separate node with 1 or more GPUs. While

this method is valid, it has the drawback of relying on the interconnect itself and the bandwidth available, which can be especially problematic on Ethernet. Furthermore, as this method consumes bandwidth, it can leave little remaining for MPI or RDMA routines, thereby constructing a bottleneck for some MPI+CUDA applications that depend on inter-process communication.

Recently efforts have been seen to support such GPU accelerators within VMs using IOMMU technologies, with implementations now available with KVM [81], Xen [95] and VMWare [96]. These efforts have shown that GPUs can achieve up to 99% of their bare metal performance when passed to a virtual machine using PCI Passthrough. VMWare specifically shows how the such PCI Passthrough solutions perform well and are likely to outperform front-end Remote API solutions such as rCUDA within a VM [96]. While these works demonstrate PCI Passthrough performance across a range of hypervisors and GPUs, they have been limited to investigating single node performance until now.

6.2.2 SR-IOV and InfiniBand

With almost all parallel HPC applications, the interconnect fabric which enables fast and efficient communication between processors becomes a central requirement to achieving good performance. Specifically, a high bandwidth link is needed for distributed processors to share large amounts of data across the system. Furthermore, low latency becomes equally important for ensuring quick delivery of small message communications and resolving large collective barriers within many parallelized codes. One such interconnect, InfiniBand, has become the most common implementation used within the Top500 list. However previously InfiniBand was inaccessible to virtualized environments.

Supporting I/O interconnects in VMs has been aided by Single Root I/O Virtu-

alization (SR-IOV), whereby multiple virtual PCI functions are created in hardware to represent a single PCI device. These virtual functions (VFs) can then be passed to a VM and used as by the guest as if it had direct access to that PCI device. SR-IOV allows for the virtualization and multiplexing to be done within the hardware, effectively providing higher performance and greater control than software solutions.

SR-IOV has been used in conjunction with Ethernet devices to provide high performance 10Gb TCP/IP connectivity within VMs [97], offering near-native bandwidth and advanced QoS features not easily obtained through emulated Ethernet offerings. Currently Amazon EC2 offers a high performance VM solution utilizing SR-IOV enabled 10Gb Ethernet adapters. While SR-IOV enabled 10Gb Ethernet solutions offers a big forward in performance, Ethernet still does not offer the high bandwidth or low latency typically found with InfiniBand solutions.

Recently SR-IOV support for InfiniBand has been added by Mellanox in the ConnectX series adapters. Initial evaluation of SR-IOV InfiniBand within KVM VMs has proven has found point-to-point bandwidth to be near-native, but up to 30% latency overhead for very small messages [82,98]. However, even with the noted overhead, this still signifies up to an order of magnitude difference in latency between InfiniBand and Ethernet with VMs. Furthermore, advanced configuration of SR-IOV enabled InfiniBand fabric is taking shape, with recent research showing up to a 30% reduction in the latency overhead [83]. However, real application performance has not yet been well understood until now.

6.2.3 GPUDirect

NVIDIA's GPUDirect technology was introduced to reduce the overhead of data movement across GPUs [99, 100]. GPUDirect supports both networking as well as peer-to-peer interfaces for single node multi-GPU systems. The most recent imple-

mentation of GPUDirect, version 3, adds support for RDMA over InfiniBand for Kepler-class GPUs.

The networking component of GPUDirect relies on three key technologies: CUDA 5 (and up), a CUDA-enabled MPI implementation, and a Kepler-class GPU (RDMA only). Both MVAPICH and OpenMPI support GPUDirect. Support for RDMA over GPUDirect is enabled by the MPI library, given supported hardware, and does not depend on application-level changes to a user's code.

In this paper, our GPUDirect work focuses on GPUDirect v3 for multi-node RDMA support. We demonstrate scaling for up to 4 nodes connected via QDR InfiniBand and show that GPUDirect RDMA improves both scalability and overall performance by approximately 9% at no cost to the end user.

6.3 A Cloud for High Performance Computing

With support for GPU Passthrough, SR-IOV, and GPUDirect, we have the building blocks for a high performance, heterogeneous cloud. In addition, other common accelerators (e.g. Xeon Phi [101]) have similarly been demonstrated in virtualized environments. Our vision is of a heterogeneous cloud, supporting both high speed networking and accelerators for tightly coupled applications.

To this end we have developed a heterogeneous cloud based on OpenStack [**?**]. In our previous work, we have demonstrated the ability to rapidly provision GPU, bare metal, and other heterogeneous resources within a single cloud [43]. Building on this effort we have added support for GPU passthrough to OpenStack as well as SR-IOV support for both ConnectX-2 and ConnectX-3 Infiniband devices. Mellanox separately supports an OpenStack InfiniBand networking plugin for OpenStack's Neutron service [102], however the Mellanox plugin depends on the ConnectX-3 adapter.

Our institutional requirements depend on ConnecteX-2 SR-IOV support, requiring an independent implementation.

OpenStack supports services for networking (Neutron), compute (Nova), identity (Keystone), storage (Cinder, Swift), and others. Our work focuses entirely on the compute service.

Scheduling is implemented at two levels: the cloud-level and the node-level. In our earlier work, we have developed a cloud-level heterogeneous scheduler for OpenStack, allowing scheduling based on architectures and resources [43]. In this model, the cloud-level scheduler dispatches jobs to nodes based on resource requirements (e.g. Kepler GPU) and node-level resource availability.

At the node, a second level of scheduling occurs to ensure that resources are tracked and not over-committed. Unlike traditional cloud paradigms, devices passed into VMs cannot be over-committed. We treat devices, whether GPUs or InfiniBand virtual functions, as schedulable resources. Thus, it is the responsibility of the individual node to track resources committed and report availability to the cloud-level scheduler. For reporting, we piggyback on top of OpenStack's existing reporting mechanism to provide a low overhead solution.

6.4 Benchmarks

We selected two molecular dynamics (MD) applications for evaluation in this study: LAMMPS and HOOMD [103, 104]. These MD simulations are chosen to represent a subset of advance parallel computation for a number of fundamental reasons:

- MD simulations provide a practical representation of N-Body simulations, which is one of the major computational *Dwarfs* [105] in parallel and distributed computing.

- MD simulations are one of the most widely deployed applications on large scale supercomputers today.
- Many MD simulations have a hybrid MPI+CUDA programming model, which has often become commonplace in HPC as the use of accelerators increases.

As such, we look to LAMMPS and HOOMD to provide a real-world example for running cutting-edge parallel programs on virtualized infrastructure. While these applications by no means represent all parallel scientific computing efforts (as justified by the 13 Dwarfs defined in [105]), we hope these MD simulators offer a more pragmatic viewpoint than traditional synthetic HPC benchmarks such as High Performance Linpack.

LAMMPS The Large-scale Atomic/Molecular Parallel Simulator is a well-understood highly parallel molecular dynamics simulator. It supports both CPU and GPU-based workloads. Unlike many simulators, both MD and otherwise, LAMMPS is heterogeneous. It will use both GPUs and multicore CPUs concurrently. For this study, this heterogeneous functionality introduces additional load on the host, allowing LAMMPS to utilize all available cores on a given system. Networking in LAMMPS is accomplished using a typical MPI model. That is, data is copied from the GPU back to the host and sent over the InfiniBand fabric. No RDMA is used for these experiments.

HOOMD-blue The Highly Optimized Object-oriented Many-particle Dynamics – Blue Edition is a particle dynamics simulator capable of scaling into the thousands of GPUs. HOOMD supports executing on both CPUs and GPUs. Unlike LAMMPS, HOOMD is homogeneous and does not support mixing of GPUs and CPUs. HOOMD supports GPUDirect using a CUDA-enabled MPI. In this paper

we focus on HOOMD's support for GPUDirect and show its benefits for increasing cluster sizes.

6.5 Experimental Setup

Using two molecular dynamics tools, LAMMPS [103] and HOOMD [104], we demonstrate a high performance *system*. That is, we combine PCI passthrough for Nvidia Kepler-class GPUs with QDR Infiniband SR-IOV and show that high performance molecular dynamics simulations are achievable within a virtualized environment.

For the first time, we also demonstrate Nvidia GPUDirect technology within such a virtual environment. Thus, we look to not only illustrate that virtual machines provide a flexible high performance infrastructure for scaling scientific workloads including MD simulations, but also that the latest HPC features and programming environments are also available in this same model.

6.5.1 Node configuration

To support the use of Nvidia GPUs and InfiniBand within a VM, specific and exact host configuration is needed. This node configuration is illustrated in Figure 6.1. While our implementation is specific to the KVM hypervisor, this setup represents a design that can be hypervisor agnostic.

Each node in the testbed uses CentOS 6.4 with a 3.13 upstream Linux kernel for the host OS, along with the latest KVM hypervisor, QEMU 2.1, and the *vfio* driver. Each Guest VM runs CentOS 6.4 with a stock 2.6.32-358.23.2 kernel. A Kepler GPU is passed through using PCI Passthrough and directly initiated within the VM via the Nvidia 331.20 driver and CUDA release 5.5. While this specific implementation used only a single GPU, it is also possible to include as many GPUs as one can fit

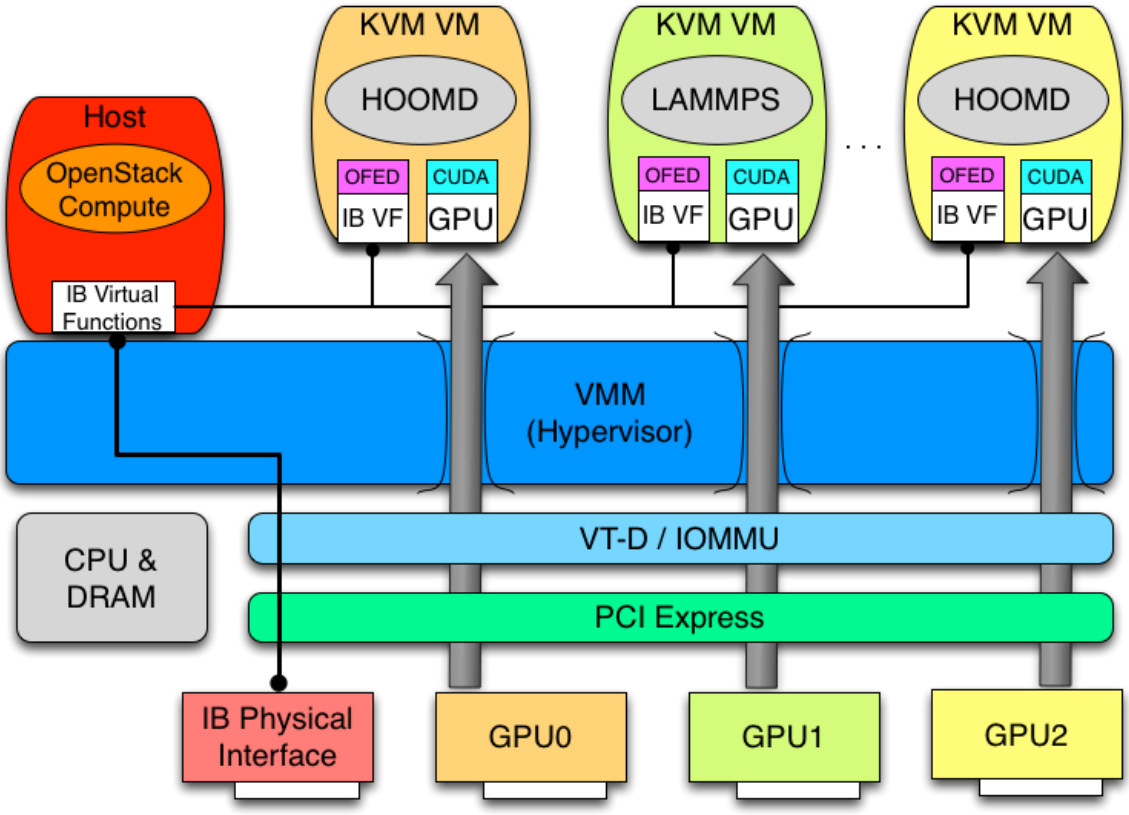

Figure 6.1 Node PCI Passthrough of GPUs and InfiniBand

within the PCI Express bus if desired. As the GPU is used by the VM, an on-board VGA device was used by the host and a standard Cirrus VGA was emulated in the guest OS.

With using SR-IOV, the OFED drivers version 2.1-1.0.0 are used with Mellanox ConnectX-3 VPI adapter with firmware 2.31.5050. The host driver initiates 4 VFs, one of which is passed through to the VM where the default OFED mlnx_ib drivers are loaded.

6.5.2 Cluster Configuration

Our test environment is composed of 4 servers each with a single Nvidia Kepler-class GPU. Two servers are equipped with K20 GPUs, while the other two servers

are equipped with K40 GPUs, demonstrating the potential for a more heterogeneous deployment. Each server is composed of 2 Intel Xeon E5-2670 CPUs, 48GB of DDR3 memory, and Mellanox ConnectX-3 QDR InfiniBand. CPU sockets and memory are split evenly between the two NUMA nodes on each system. All InfiniBand adapters use a single Voltaire 4036 QDR switch with a software subnet manager for IPoIB functionality.

For these experiments, both the GPUs and InfiniBand adapters are attached to NUMA node 1 and both the guest VMs and the base system utilized identical software stacks. Each guest was allocated 20 GB of RAM and a full socket of 8 cores, and pinned to NUMA node 1 to ensure optimal hardware usage. While all VMs are capable of login via the InfiniBand IPoIB setup, a 1Gb Ethernet network was used for all management and login tasks.

For a fair and effective comparison, we also use a native environment without any virtualization. This native environment employs the same hardware configuration, and like the Guest OS runs CentOS 6.4 with the stock 2.6.32-358.23.2 kernel.

6.6 Results

In this section, we discuss the performance of both the LAMMPS and HOOMD molecular dynamics simulation tools when running within a virtualized environment. Specifically, we scale each application to 32 cores and 4 GPUs, both in a native bare-metal and virtualized environments. Each application set was run 10 times, with the results averaged accordingly.

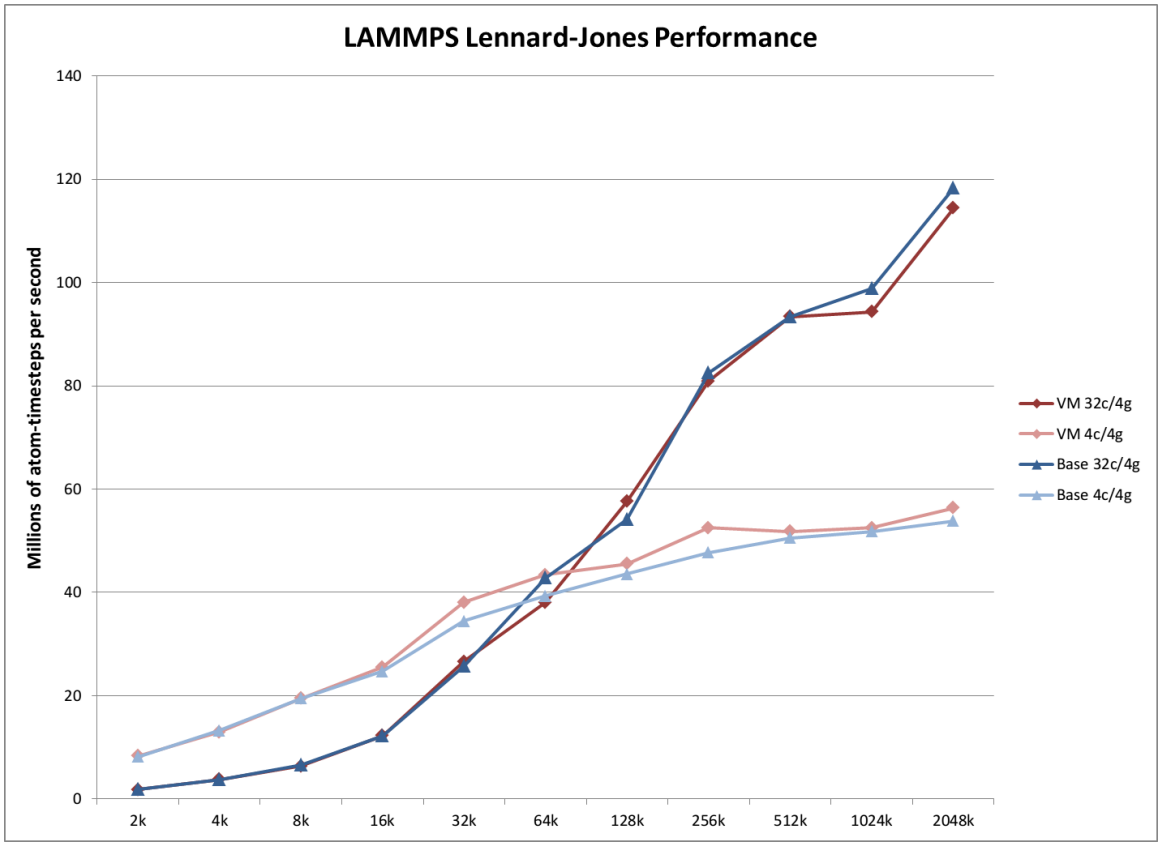

Figure 6.2 LAMMPS LJ Performance

6.6.1 LAMMPS

Figure 6.2 shows one of the most common LAMMPS algorithms used; the Lennard-Jones potential (LJ). This algorithm is deployed in two main configurations - a 1:1 core to GPU mapping, and a 8:1 core to GPU mapping. With the LAMMPS GPU implementation, a delicate balance between GPUs and CPUs is required to find the optimal ratio for fastest computation, however here we just look at the two most obvious choices. With small problem sizes, the 1:1 mapping outperforms the more complex core deployment, as the problem does not require the additional complexity provided with multi-core solution. As expected the multi-core configuration quickly offers better performance for larger problem sizes, achieving roughly twice the performance with all 8 available cores. This is largely due to the availability of all 8 cores

to keep the GPU running 100% with continual computation.

The important factor for this manuscript is the relative performance of the virtualized environment. From the results, it is clear the VM solution performs very well compared to the best-case native deployment. For the multi-core configuration across all problem sizes, the virtualized deployment averaged 98.5% efficiency compared to native. The single core per GPU deployment reported better-than native performance at 100% native. This is likely due to caching effects, but further investigation is needed to fully identify this occurrence.

Another common LAMMPS algorithm, the Rhodopsin protein in solvated lipid bilayer benchmark (Rhodo), was also run with results given in Figure 6.3. As with the LJ runs, we see the multi-core to GPU configuration resulting in higher computational performance for the larger problem sizes compared to the single core per GPU configuration, as expected.

Again, the overhead of the virtualized configuration remains low across all configurations and problem sizes, with an average 96.4% efficiency compared to native. Interestingly enough, we also see the performance gap decrease as the problem size increases, with the 512k problem size in yielding 99.3% of native performance. This finding leads us to extrapolate that a virtualized MPI+CUDA implementation would scale to a larger computational resource with similar success.

6.6.2 HOOMD

In Figure 6.4 we show the performance of a Lennard-Jones liquid simulation with 256K particles running under HOOMD. HOOMD includes support for CUDA-aware MPI implementations via GPUDirect. The MVAPICH 2.0 GDR implementation enables a further optimization by supporting RDMA for GPUDirect. From Figure 6.4 we can see that HOOMD simulations, both with and without GPUDirect, perform very

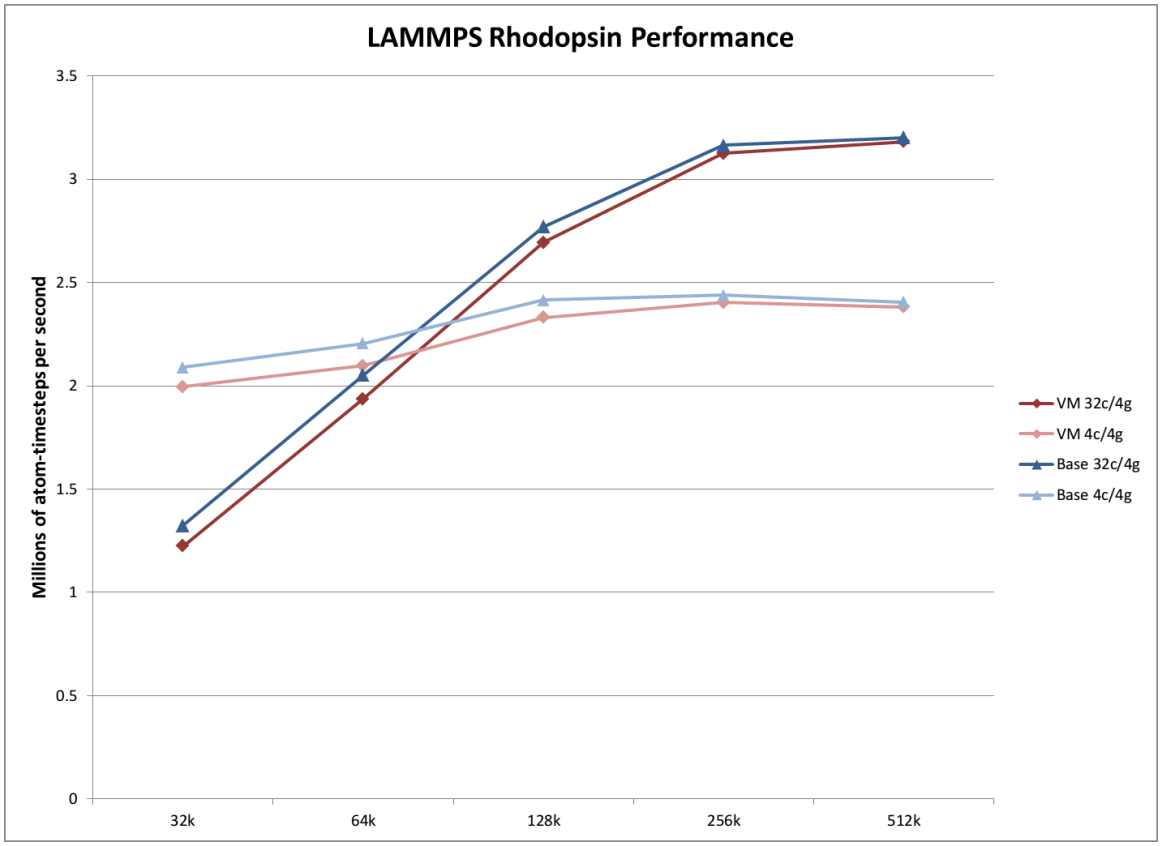

Figure 6.3 LAMMPS RHODO Performance

near-native. The GPUDirect results at 4 nodes achieve 98.5% of the base system's performance. The non-GPUDirect results achieve 98.4% efficiency at 4 nodes. These results indicate the virtualized HPC environment is able to support such complex workloads. While the effective testbed size is relatively small, it indicates that such workloads may scale equally well to hundreds or thousands of nodes.

6.7 Discussion

From the results, we see the potential for running HPC applications in a virtualized environment using GPUs and InfiniBand interconnect fabric. Across all LAMMPS runs with ranging core configurations, we found only a 1.9% overhead between the

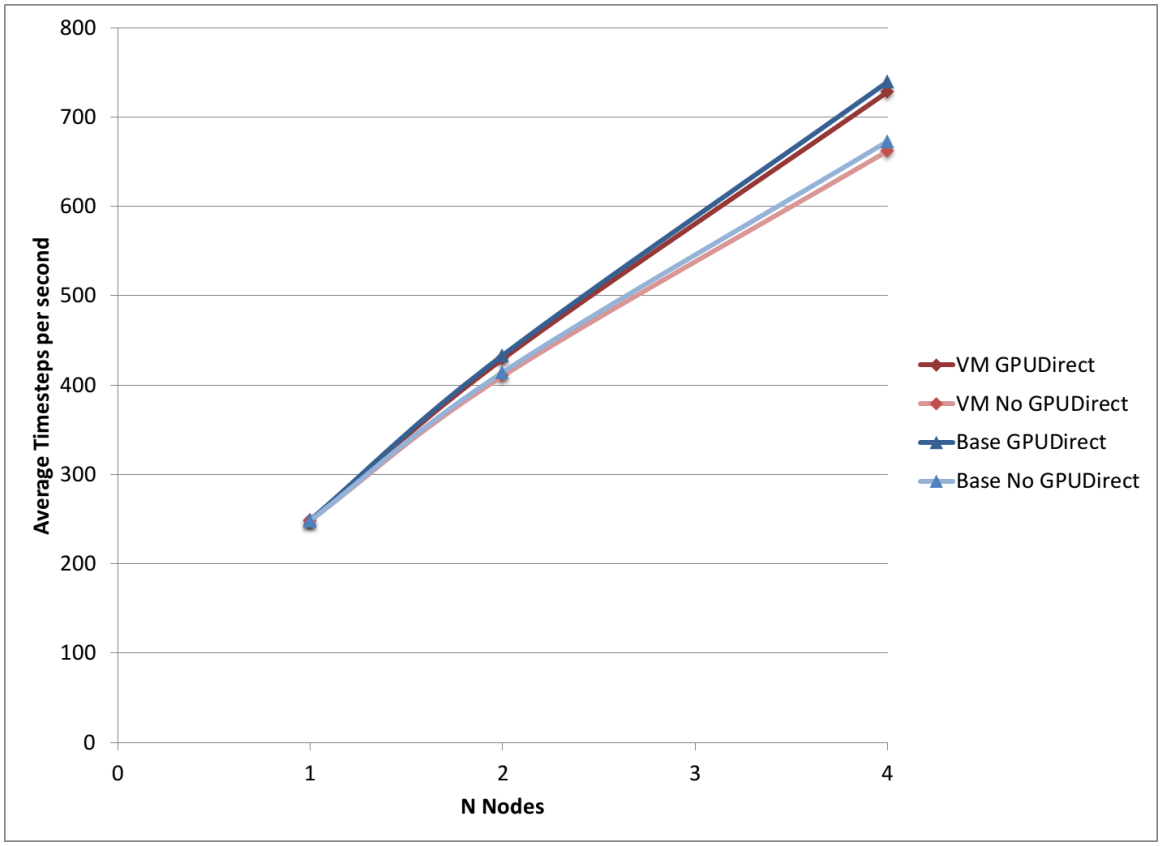

Figure 6.4 HOOMD LJ Performance with 256k Simulation

KVM virtualized environment and native. For HOOMD, we found a similar 1.5% overhead, both with and without GPU Direct. These results go against conventional wisdom that HPC workloads do not work in VMs. In fact ,we show two N-Body type simulations programmed in an MPI+CUDA implementation perform at roughly near-native performance in tuned KVM virtual machines.

With HOOMD, we see how GPUDirect RDMA shows a clear advantage over the non-GPUDirect implementation, achieving a 9% performance boost in both the native a virtualized experiments. While GPUDirect's performance impact has been well evaluated previously [99], it is the author's belief that this manuscript represents the first time GPUDirect has has been utilized in a virtualized environment.

Another interesting finding of running LAMMPS and HOOMD in a virtualized

environment is as workload scales from a single node to 32 cores, the overhead does not increase. These results lend credence to the notion that this solution would also work for a much larger deployment. Specifically, it would be possible to expand such computational problems to a larger deployment in FutureGrid [106], Chameleon Cloud [107], or even the planned NSF Comet machine at SDSC, scheduled to provide up to 2 Petaflops of computational power. Effectively, these results help support the theory that a majority of HPC computations can be supported in virtualized environment with minimal overhead.

6.8 Conclusion

With the advent of cloud infrastructure, the ability to run large-scale parallel scientific applications has become possible but limited due to both performance and hardware availability concerns. In this work we show that advanced HPC-oriented hardware such as the latest Nvidia GPUs and InfiniBand fabric are now available within a virtualized infrastructure. Our results find MPI + CUDA applications such as molecular dynamics simulations run at near-native performance compared to traditional non-virtualized HPC infrastructure, with just an averaged 1.9% and 1.5% overhead for LAMMPS and HOOMD, respectively. Moving forward, we show the utility of GPUDirect RDMA for the first time in a cloud environment with HOOMD. Effectively, we look to pave the way for large-scale virtualized cloud Infrastructure to support a wide array of advanced scientific computation commonly found running on many supercomputers today. Our efforts leverage these technologies and provide them in an open source Infrastructure-as-a-Service framework using OpenStack.

Chapter 7

VirtualCalifornia Earthquate Simulation in SR-IOV InfiniBand enabled Virtual Cloud Infrastructure

7.1 Introduction

TODO Finish!

Chapter 8

Conclusion

8.1 Summary

TBD

8.2 Future Work

TBD

Bibliography

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Intl. J. Supercomputer Applications*, vol. 15, no. 3, 2001.

[2] I. Foster, C. Kesselman *et al.*, "The Physiology of the Grid:An Open Grid Services Architecture for Distributed Systems Integration," Argonne National Laboratory, Chicago, Tech. Rep., Jan. 2002.

[3] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared," in *Grid Computing Environments Workshop, 2008. GCE'08*, 2008, pp. 1–10.

[4] D. DiNucci, "Fragmented future," *AllBusiness–Champions of Small Business*, 1999. [Online]. Available: http://www.cdinucci.com/Darcy2/articles/Print/Printarticle7.html

[5] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic, and S. Zimek, "Web Services Choreography Interface," Jun. 2002, version 1.0. [Online]. Available: http://wwws.sun.com/software/xml/developers/wsci/index.html

[6] D. Krafzig, K. Banke, and D. Slama, *Enterprise SOA: Service-Oriented Architecture Best Practices (The Coad Series)*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2004.

[7] G. von Laszewski, A. J. Younge, X. He, K. Mahinthakumar, and L. Wang, "Experiment and Workflow Management Using Cyberaide Shell," in *Proceedings of the 4th International Workshop on Workflow Systems in e-Science (WSES 09) with 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 09)*. IEEE, May 2009. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/09-gridshell-ccgrid/vonLaszewski-ccgrid09-final.pdf

[8] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. L. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, New York, U. S. A., Oct. 2003, pp. 164–177.

[9] "Vmware esx server," [Online], http://www.vmware.com/de/products/vi/esx/.

[10] Amazon, "Elastic Compute Cloud." [Online]. Available: http://aws.amazon.com/ec2/

[11] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus Open-source Cloud-computing System," *Proceedings of Cloud Computing and Its Applications*, 2008.

[12] J. Fontan, T. Vazquez, L. Gonzalez, R. S. Montero, and I. M. Llorente, "OpenNEbula: The Open Source Virtual Machine Manager for Cluster Computing," in *Open Source Grid and Cluster Software Conference*, San Francisco, CA, USA, May 2008.

[13] J. Chase, D. Irwin, L. Grit, J. Moore, and S. Sprenkle, "Dynamic virtual clusters in a grid site manager," in *12th IEEE International Symposium on High Performance Distributed Computing, 2003. Proceedings*, 2003, pp. 90–100.

[14] K. Keahey, I. Foster, T. Freeman, X. Zhang, and D. Galron, "Virtual Workspaces in the Grid," *Lecture Notes in Computer Science*, vol. 3648, pp. 421–431, 2005. [Online]. Available: http://workspace.globus.org/papers/VW_EuroPar05.pdf

[15] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputer Applications*, vol. 11, no. 2, pp. 115–128, 1997, ftp://ftp.globus.org/pub/globus/papers/globus.pdf.

[16] CERN, "LHC Computing Grid Project," Web Page, Dec. 2003. [Online]. Available: http://lcg.web.cern.ch/LCG/

[17] J. Luo, A. Song, Y. Zhu, X. Wang, T. Ma, Z. Wu, Y. Xu, and L. Ge, "Grid supporting platform for AMS data processing," *Lecture notes in computer science*, vol. 3759, p. 276, 2005.

[18] "CMS," Web Page. [Online]. Available: http://cms.cern.ch/

[19] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[20] "Amazon Elastic Compute Cloud." [Online]. Available: http://aws.amazon.com/ec2/

[21] R. Creasy, "The origin of the VM/370 time-sharing system," *IBM Journal of Research and Development*, vol. 25, no. 5, pp. 483–490, 1981.

[22] "Amazon elastic compute cloud," [Online], http://aws.amazon.com/ec2/.

[23] S. Rixner, "Network virtualization: Breaking the performance barrier," *Queue*, vol. 6, no. 1, p. 37, 2008.

[24] R. Harper and K. Rister, "Kvm limits arbitrary or architectural?" IBM Linux Technology Center, Jun 2009.

[25] "FutureGrid," Web Page, 2009. [Online]. Available: http://www.futuregrid.org

[26] G. von Laszewski, G. C. Fox, F. Wang, A. J. Younge, A. Kulshrestha, and G. Pike, "Design of the FutureGrid Experiment Management Framework," in *Proceedings of Gateway Computing Environments 2010 at Supercomputing 2010*. New Orleans, LA: IEEE, Nov 2010. [Online]. Available: http://grids.ucs.indiana.edu/ptliupages/publications/vonLaszewski-10-FG-exp-GCE10.pdf

[27] J. Dongarra, H. Meuer, and E. Strohmaier, "Top 500 supercomputers," website, November 2013.

[28] R. Henschel and A. J. Younge, "First quarter 2011 spec omp results," Webpage, Mar 2011. [Online]. Available: http://www.spec.org/omp/results/res2011q1/

[29] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," University of California at Berkeley, Tech. Rep., February 2009. [Online]. Available: http://berkeleyclouds.blogspot.com/2009/02/above-clouds-released.html

[30] A. S. Tanenbaum and M. Van Steen, *Distributed systems*. Prentice Hall, 2002, vol. 2.

[31] L. Wang, G. von Laszewski, A. J. Younge, X. He, M. Kunze, and J. Tao, "Cloud Computing: a Perspective Study," *New Generation Computing*, vol. 28, pp. 63–69, Mar 2010. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/10-cloudcomputing-NGC/vonLaszewski-10-NGC.pdf

[32] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[33] "Windows azure platform," [Online], http://www.microsoft.com/azure/.

[34] G. von Laszewski, G. C. Fox, F. Wang, A. J. Younge, A. Kulshrestha, G. G. Pike, W. Smith, J. Vockler, R. J. Figueiredo, J. Fortes *et al.*, "Design of the futuregrid experiment management framework," in *Gateway Computing Environments Workshop (GCE), 2010*. IEEE, 2010, pp. 1–10.

[35] K. Yelick, S. Coghlan, B. Draney, and R. S. Canon, "The Magellan Report on Cloud Computing for Science," U.S. Department of Energy Office of Science Office of Advanced Scientific Computing Research (ASCR), Tech. Rep., Dec. 2011.

[36] OpenStack, "Openstack compute administration manual," 2013.

[37] K. Keahey, I. Foster, T. Freeman, and X. Zhang, "Virtual workspaces: Achieving quality of service and quality of life in the grid," *Scientific Programming Journal*, vol. 13, no. 4, pp. 265–276, 2005.

[38] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The Eucalyptus open-source cloud-computing system," in *Proceedings of Cloud Computing and Its Applications*, October 2008. [Online]. Available: http://eucalyptus.cs.ucsb.edu/wiki/Presentations

[39] C. Nvidia, "Programming guide," 2008.

[40] J. E. Stone, D. Gohara, and G. Shi, "OpenCL: A parallel programming standard for heterogeneous computing systems," *Computing in science & engineering*, vol. 12, no. 3, p. 66, 2010.

[41] S. Hong and H. Kim, "An integrated GPU power and performance model," in *ACM SIGARCH Computer Architecture News*, vol. 38. ACM, 2010, pp. 280–289.

[42] P. Kogge, K. Bergman, S. Borkar, D. Campbell, W. Carson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," 2008.

[43] S. Crago, K. Dunn, P. Eads, L. Hochstein, D.-I. Kang, M. Kang, D. Modium, K. Singh, J. Suh, and J. P. Walters, "Heterogeneous cloud computing," in *Proceedings of the Workshop on Parallel Programming on Accelerator Clusters (PPAC). Cluster Computing (CLUSTER), 2011 IEEE International Conference on.* IEEE, 2011, pp. 378–385.

[44] J. Sanders and E. Kandrot, *CUDA by example: an introduction to general-purpose GPU programming.* Addison-Wesley Professional, 2010.

[45] V. V. Kindratenko, J. J. Enos, G. Shi, M. T. Showerman, G. W. Arnold, J. E. Stone, J. C. Phillips, and W.-m. Hwu, "GPU clusters for high-performance computing," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on.* IEEE, 2009, pp. 1–8.

[46] K. J. Barker, K. Davis, A. Hoisie, D. J. Kerbyson, M. Lang, S. Pakin, and J. C. Sancho, "Entering the petaflop era: the architecture and performance of Road-

runner," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 1.

[47] S. Craven and P. Athanas, "Examining the viability of FPGA supercomputing," *EURASIP Journal on Embedded systems*, vol. 2007, no. 1, pp. 13–13, 2007.

[48] G. Shainer, T. Liu, J. Layton, and J. Mora, "Scheduling strategies for HPC as a service (HPCaaS)," in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*. IEEE, 2009, pp. 1–6.

[49] A. J. Younge, R. Henschel, J. T. Brown, G. von Laszewski, J. Qiu, and G. C. Fox, "Analysis of Virtualization Technologies for High Performance Computing Environments," in *Proceedings of the 4th International Conference on Cloud Computing (CLOUD 2011)*. Washington, DC: IEEE, July 2011.

[50] W. Wade, "How NVIDIA and Citrix are driving the future of virtualized visual computing," [Online], http://blogs.nvidia.com/blog/2013/05/22/synergy/.

[51] S. Long, "Virtual machine graphics acceleration deployment guide," VMWare, Tech. Rep., 2013.

[52] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. T. Klosowski, "Chromium: a stream-processing framework for interactive rendering on clusters," in *ACM Transactions on Graphics (TOG)*, vol. 21. ACM, 2002, pp. 693–702.

[53] L. Shi, H. Chen, J. Sun, and K. Li, "vCUDA: GPU-accelerated high-performance computing in virtual machines," *Computers, IEEE Transactions on*, vol. 61, no. 6, pp. 804–816, 2012.

[54] G. Giunta, R. Montella, G. Agrillo, and G. Coviello, "A GPGPU transparent virtualization component for high performance computing clouds," in *Euro-Par 2010 - Parallel Processing*, ser. Lecture Notes in Computer Science, P. DAmbra, M. Guarracino, and D. Talia, Eds. Springer Berlin Heidelberg, 2010, vol. 6271, pp. 379–391. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-15277-1_37

[55] J. Duato, A. J. Pena, F. Silla, J. C. Fernández, R. Mayo, and E. S. Quintana-Orti, "Enabling CUDA acceleration within virtual machines using rCUDA," in *High Performance Computing (HiPC), 2011 18th International Conference on*. IEEE, 2011, pp. 1–10.

[56] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The scalable heterogeneous computing (SHOC) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*. ACM, 2010, pp. 63–74.

[57] E. R. Hawkes, R. Sankaran, J. C. Sutherland, and J. H. Chen, "Direct numerical simulation of turbulent combustion: fundamental insights towards predictive models," in *Journal of Physics: Conference Series*, vol. 16. IOP Publishing, 2005, p. 65.

[58] J. Duato, A. Pena, F. Silla, R. Mayo, and E. Quintana-Orti, "rCUDA: Reducing the number of gpu-based accelerators in high performance clusters," in *High Performance Computing and Simulation (HPCS), 2010 International Conference on*, June 2010, pp. 224–231.

[59] B. L. Jacob and T. N. Mudge, "A look at several memory management units, TLB-refill mechanisms, and page table organizations," in *Proceedings of the*

Eighth International Conference on Architectural Support for Programming Languages and Operating Systems. ACM, 1998, pp. 295–306.

[60] B.-A. Yassour, M. Ben-Yehuda, and O. Wasserman, "Direct device assignment for untrusted fully-virtualized virtual machines," IBM Research, Tech. Rep., 2008.

[61] M. Ben-Yehuda, J. Xenidis, M. Ostrowski, K. Rister, A. Bruemmer, and L. Van Doorn, "The price of safety: Evaluating IOMMU performance," in *Ottawa Linux Symposium*, 2007.

[62] J. Liu, "Evaluating standard-based self-virtualizing devices: A performance study on 10 GbE NICs with SR-IOV support," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, April 2010, pp. 1–12.

[63] V. Jujjuri, E. Van Hensbergen, A. Liguori, and B. Pulavarty, "VirtFS-a virtualization aware file system passthrough," in *Ottawa Linux Symposium*, 2010.

[64] C. Yang, H. Wang, W. Ou, Y. Liu, and C. Hsu, "On implementation of GPU virtualization using PCI pass-through," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings (CloudCom)*, 2012.

[65] M. Dowty and J. Sugerman, "GPU virtualization on VMware's hosted I/O architecture," *ACM SIGOPS Operating Systems Review*, vol. 43, no. 3, pp. 73 – 82, 2009.

[66] "FutureGrid," Web page. [Online]. Available: http://portal.futuregrid.org

[67] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The Scalable Heterogeneous Computing

(SHOC) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, ser. GPGPU '10. ACM, 2010, pp. 63–74.

[68] "LAMMPS molecular dynamics simulator," http://lammps.sandia.gov/, [Online; accessed Jan. 2, 2014].

[69] A. Athanasopoulos, A. Dimou, V. Mezaris, and I. Kompatsiaris, "GPU acceleartion for support vector machines," in *Proc 12th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2001)*, April 2011.

[70] "LULESH shock hydrodynamics application," https://codesign.llnl.gov/lulesh.php, [Online; accessed Jan. 2, 2014].

[71] S. Plimpton, "Fast parallel algorithms for short-range molecular dynamics," *Journal of Computational Physics*, vol. 117, no. 1, pp. 1 – 19, 1995.

[72] W. M. Brown, "GPU acceleration in LAMMPS," http://lammps.sandia.gov/workshops/Aug11/Brown/brown11.pdf, [Online; accessed Jan. 2, 2014].

[73] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011.

[74] "Hydrodynamics Challenge Problem," https://codesign.llnl.gov/pdfs/spec-7.pdf, [Online; accessed Jan. 2, 2014].

[75] A. Arcangeli, "Transparent hugepage support," http://www.linux-kvm.org/wiki/images/9/9e/2010-forum-thp.pdf, 2010, [Online; accessed Jan. 29, 2014].

[76] J. Jose, M. Li, X. Lu, K. C. Kandalla, M. D. Arnold, and D. K. Panda, "SR-IOV support for virtualization on infiniband clusters: Early experience," in *Cluster*

Computing and the Grid, IEEE International Symposium on. IEEE Computer Society, 2013, pp. 385–392.

[77] S. Jha, J. Qiu, A. Luckow, P. K. Mantha, and G. C. Fox, "A tale of two data-intensive paradigms: Applications, abstractions, and architectures," in *Proceedings of the 3rd International Congress on Big Data*, 2014.

[78] R. L. Moore, C. Baru, D. Baxter, G. C. Fox, A. Majumdar, P. Papadopoulos, W. Pfeiffer, R. S. Sinkovits, S. Strande, M. Tatineni *et al.*, "Gateways to discovery: Cyberinfrastructure for the long tail of science," in *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*. ACM, 2014, p. 39.

[79] P. Luszczek, E. Meek, S. Moore, D. Terpstra, V. M. Weaver, and J. Dongarra, "Evaluation of the hpc challenge benchmarks in virtualized environments," in *Proceedings of the 2011 International Conference on Parallel Processing - Volume 2*, ser. Euro-Par'11. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 436–445.

[80] N. Huber, M. von Quast, M. Hauck, and S. Kounev, "Evaluating and modeling virtualization performance overhead for cloud environments." in *CLOSER*, 2011, pp. 563–573.

[81] J. P. Walters, A. J. Younge, D.-I. Kang, K.-T. Yao, M. Kang, S. P. Crago, and G. C. Fox, "GPU-Passthrough Performance: A Comparison of KVM, Xen, VMWare ESXi, and LXC for CUDA and OpenCL Applications," in *Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD 2014)*. Anchorage, AK: IEEE, 2014.

[82] J. Jose, M. Li, X. Lu, K. C. Kandalla, M. D. Arnold, and D. K. Panda, "Sr-iov support for virtualization on infiniband clusters: Early experience," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE, 2013, pp. 385–392.

[83] M. Musleh, V. Pai, J. P. Walters, A. J. Younge, and S. P. Crago, "Bridging the Virtualization Performance Gap for HPC using SR-IOV for InfiniBand," in *Proceedings of the 7th IEEE International Conference on Cloud Computing (CLOUD 2014)*. Anchorage, AK: IEEE, 2014.

[84] "Aws high performance computing," http://aws.amazon.com/hpc/, last Access Nov. 2014.

[85] "Openstack flavors," http://docs.openstack.org/openstack-ops/content/flavors.html, last Access Nov. 2014.

[86] K. Pepple, *Deploying OpenStack*. O'Reilly Media, Inc., 2011.

[87] S. Hazelhurst, "Scientific computing using virtual high-performance computing: a case study using the amazon elastic computing cloud," in *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*. ACM, 2008, pp. 94–103.

[88] E. Amazon, "Amazon elastic compute cloud (amazon ec2)," *Amazon Elastic Compute Cloud (Amazon EC2)*, 2010.

[89] R. Jennings, *Cloud Computing with the Windows Azure Platform*. John Wiley & Sons, 2010.

[90] "Google cloud platform," https://cloud.google.com/, last Access Nov. 2014.

[91] L. Ramakrishnan, R. S. Canon, K. Muriki, I. Sakrejda, and N. J. Wright, "Evaluating interconnect and virtualization performance forhigh performance computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 2, pp. 55–60, Oct. 2012. [Online]. Available: http://doi.acm.org/10.1145/2381056.2381071

[92] M. Righini, "Enabling intel virtualization technology features and benefits," Intel Corporation, Tech. Rep., 2010.

[93] AMD, "AMD i/o virtualization technology (IOMMU) specification," AMD Corporation, Tech. Rep., 2009.

[94] A. Limited, "Arm system memory management unit architecture specification," ARM Limited, Tech. Rep., 2013.

[95] A. J. Younge, J. P. Walters, S. Crago, and G. C. Fox, "Evaluating GPU Passthrough in Xen for High Performance Cloud Computing," in *High-Performance Grid and Cloud Computing Workshop at the 28th IEEE International Parallel and Distributed Processing Symposium*, IEEE. Pheonix, AZ: IEEE, 05/2014 2014.

[96] L. Vu, H. Sivaraman, and R. Bidarkar, "Gpu virtualization for high performance general purpose computing on the esx hypervisor," in *Proceedings of the High Performance Computing Symposium*, ser. HPC '14. San Diego, CA, USA: Society for Computer Simulation International, 2014, pp. 2:1–2:8. [Online]. Available: http://dl.acm.org/citation.cfm?id=2663510.2663512

[97] J. Liu, "Evaluating standard-based self-virtualizing devices: A performance study on 10 gbe nics with sr-iov support," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, April 2010, pp. 1–12.

[98] T. P. P. D. L. Ruivo, G. B. Altayo, G. Garzoglio, S. Timm, H. Kim, S.-Y. Noh, and I. Raicu, "Exploring infiniband hardware virtualization in opennebula towards efficient high-performance computing." in *CCGRID*, 2014, pp. 943–948.

[99] "NVIDIA GPUDirect," https://developer.nvidia.com/gpudirect, [Online; accessed Nov. 24, 2014].

[100] G. Shainer, A. Ayoub, P. Lui, T. Liu, M. Kagan, C. R. Trott, G. Scantlen, and P. S. Crozier, "The development of mellanox/nvidia gpudirect over infinibanda new model for gpu to gpu communications," *Computer Science-Research and Development*, vol. 26, no. 3-4, pp. 267–273, 2011.

[101] "Getting Xen working for Intel(R) Xeon Phi(tm) Coprocessor," https://software.intel.com/en-us/articles/getting-xen-working-for-intelr-xeon-phitm-coprocessor, [Online; accessed Nov. 24, 2014].

[102] "Mellanox Neutron Plugin," https://wiki.openstack.org/wiki/Mellanox-Neutron, [Online; accessed Nov. 24, 2014].

[103] S. Plimpton, P. Crozier, and A. Thompson, "Lammps-large-scale atomic/molecular massively parallel simulator," *Sandia National Laboratories*, 2007.

[104] J. Anderson, A. Keys, C. Phillips, T. Dac Nguyen, and S. Glotzer, "Hoomd-blue, general-purpose many-body dynamics on the gpu," in *APS Meeting Abstracts*, vol. 1, 2010, p. 18008.

[105] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams *et al.*, "The landscape of parallel computing research: A view from berkeley," Technical Report

UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Tech. Rep., 2006.

[106] G. Fox, G. von Laszewski, J. Diaz, K. Keahey, J. Fortes, R. Figueiredo, S. Smallen, W. Smith, and A. Grimshaw, "Futuregrida reconfigurable testbed for cloud, hpc and grid computing," *Contemporary High Performance Computing: From Petascale toward Exascale, Computational Science. Chapman and Hall/CRC*, 2013.

[107] K. Keahey, J. Mambretti, D. K. Panda, P. Rad, W. Smith, and D. Stanzione, "Nsf chameleon cloud," website, November 2014. [Online]. Available: http://www.chameleoncloud.org/

Appendix A

Appendix

This section provides the formal description for TBD.

A.1 AppendixA

A.1.1 Appendix A1