From the Proceedings of IEEE Cloud 2014

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

John Paul Walters*, Andrew J. Younge†, Dong-In Kang*, Ke-Thia Yao*,
Mikyung Kang*, Stephen P. Crago*, and Geoffrey C. Fox†
*Information Sciences Institute
University of Southern California, Arlington, VA 22203
Email: {jwalters,dkang,kyao,mkkang,crago}@isi.edu
†Pervasive Technology Institute
Indiana University, Bloomington, IN 47408
Email: {ajyounge,gcf}@indiana.edu

***Abstract*—As more scientific workloads are moved into the cloud, the need for high performance accelerators increases. Accelerators such as GPUs offer improvements in both performance and power efficiency over traditional multi-core processors; however, their use in the cloud has been limited. Today, several common hypervisors support GPU passthrough, but their performance has not been systematically characterized.**

In this paper we show that low overhead GPU passthrough is achievable across 4 major hypervisors and two processor microarchitectures. We compare the performance of two generations of NVIDIA GPUs within the Xen, VMWare ESXi, and KVM hypervisors, and we also compare the performance to that of Linux Containers (LXC). We show that GPU passthrough to KVM achieves 98–100% of the base system's performance across two architectures, while Xen and VMWare achieve 96–99% of the base systems performance, respectively. In addition, we describe several valuable lessons learned through our analysis and share the advantages and disadvantages of each hypervisor/GPU passthrough solution.

***Keywords*-virtualization; GPU passthrough; KVM; Xen; VMWare; LXC**

I. 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 GPU passthrough across all major cloud hypervisors. Instead, alternative solutions have been proposed that attempt to virtualize the GPU [1]–[4], but sacrifice performance.

In this paper, we characterize the performance of both NVIDIA Fermi and Kepler GPUs operating in PCI passthrough mode in VMWare ESXi, 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.

II. 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.

A. GPU API Remoting

rCUDA, vCUDA, GViM, and gVirtuS are well-known API remoting solutions [1]–[4]. Fundamentally, 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.

B. 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 [5], an IOMMU serves the fundamental purpose of connecting a direct memory access (DMA) capable I/O bus to main memory. 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 [6], with a slight overhead, especially in early implementations [7].

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 [8], storage [9], and other PCI-attached devices; however, GPUs have historically lagged behind other devices in their support for virtual machine passthrough.

C. 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.

Table I: 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

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 [10], and VMWare ESXi [11], 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.

III. Experimental Methodology

A. 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 K20m GPU. The Delta system, provided by the FutureGrid project [12], represents the Westmere microarchitecture with a Fermi class C2075 GPU. Table I 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 ESXi, Xen, KVM, and LXC, are summarized in Table II. Note that each virtualization solution imposes its own unique requirements

Table II: 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

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 III-B, we are running a common virtual machine across all experiments.

B. 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.

C. 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 [13]. 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.

D. Application Benchmarks

For our application benchmarks, we have chosen the LAMMPS molecular dynamics simulator [14], the GPU-LIBSVM [15], and the LULESH shock hydrodynamics simulator [16]. 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 [14], [17] used for production MD simulation on both CPUs and GPUs [18]. 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 [19] of the machine learning classification algorithm support vector machine (SVM). GPU-accelerated LIBSVM [15] 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. 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 [20]. 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 [16].

IV. 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

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

	Bespin (K20m)								Delta (C2075)							
	KVM		Xen		LXC		VMWare		KVM		Xen		LXC		VMWare	
	Mean	Max	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	0.073	1.00	4.95
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	0.663	0.933	36.6
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	0.016	0.962	7.01
	Bespin PCIe-only								Delta PCIe-only							
	KVM		Xen		LXC		VMWare		KVM		Xen		LXC		VMWare	
	Mean	Max	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	0.995	4.37
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	0.380	0.864	36.6
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	0.869	7.01

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.

A. *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 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 (K20m) 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 III, 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.

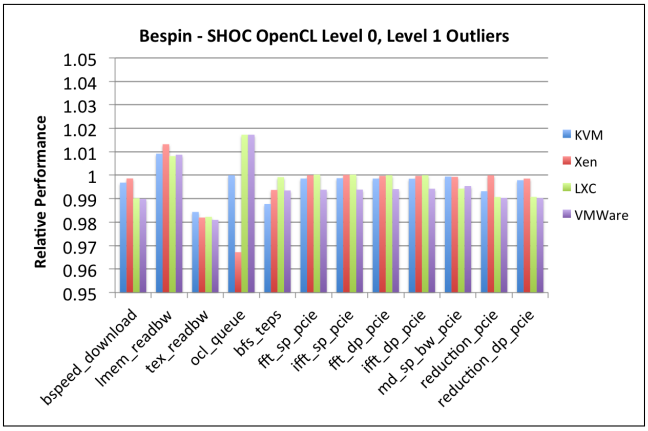

Figure 1: SHOC Levels 0 and 1 relative performance on Bespin system. Results show benchmarks over or underperforming by 0.5% or greater. Higher is better.

There are four level 0 outliers: bspeed_download, lmem_readbw, tex_readbw, and ocl_queue. These are shown in Figure 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 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 bus lies at the boundary of the virtual machine and the physical GPU, and requires interrupt remapping, IOMMU interaction, etc. in order to

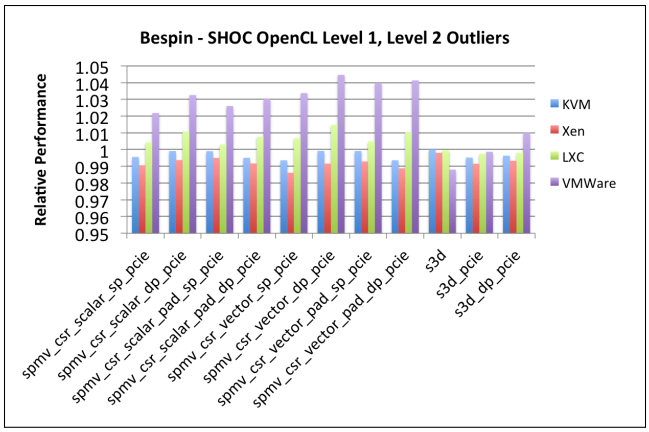

Figure 2: SHOC Levels 1 and 2 relative performance on Bespin system. Results show benchmarks over or under-performing by 0.5% or greater. Higher is better.

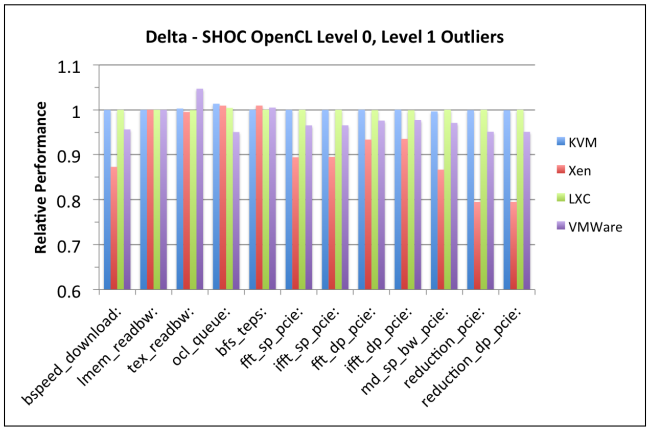

Figure 3: SHOC Levels 0 and 1 relative performance on Delta system. Benchmarks shown are the same as Bespin's. Higher is better.

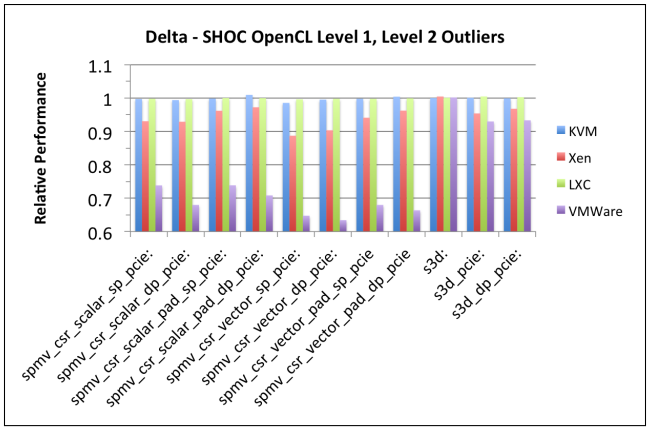

Figure 4: SHOC Levels 1 and 2 relative performance. Benchmarks shown are the same as Bespin's. Higher is better.

enable GPU passthrough into the virtual machine. Despite this, in Figure 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 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 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 2 we present the remaining SHOC Level 1 outliers as well as the SHOC Level 2 (S3D) outliers. In these results we see an interesting trend, namely that VMWare consistently outperforms the base system in the Spmv PCIe microbenchmarks on the Bespin system. Spmv's non-PCIe benchmarks performed comparably across all hypervisors at less than 0.5% overhead.

Turning to the Delta system, in Figures 3 and 4, we show the same benchmarks for the Delta system as was shown in Figures 1 and 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 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. We speculate that VMWare may be experiencing a similar issue on the Delta system and not on our Bespin system. Further study is needed in order to determine with certainty whether pages sizes are contributing to the performance decreases in the case of Xen and VMWare ESXi.

In light of this, we broadly find that virtualization overhead across hypervisors and architectures is minimal for the case of the SHOC benchmarks. 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 GPU passthrough in a cloud environment. In the case of the Bespin system, it is clear that GPU 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.

B. GPU-LIBSVM Performance

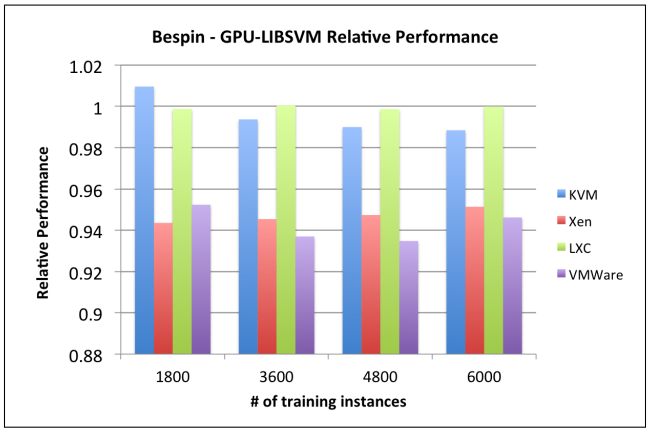

Figure 5: GPU-LIBSVM relative performance on Bespin system. Higher is better.

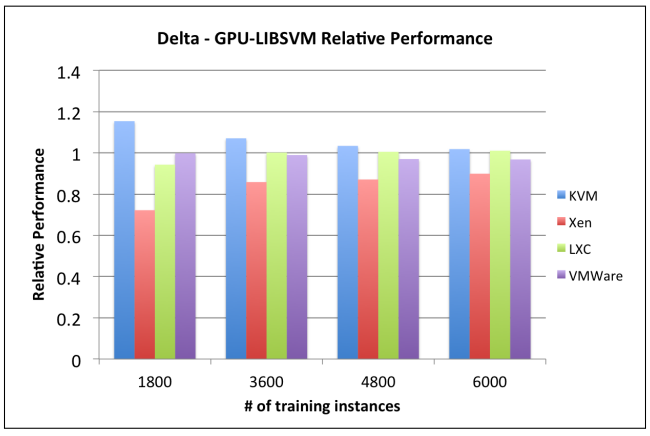

Figure 6: GPU-LIBSVM relative performance on Delta showing improved performance within virtual environments. Higher is better.

In Figures 5 and 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 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 speculate that this may 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 [21]. Nevertheless, further research is needed in order to more precisely characterize the impact of hugepages on virtualized workloads. After disabling hugepages on the KVM host for the Delta system, performance dropped to 80–87% of the base system. LXC and VMWare perform close to the base system, while Xen achieves between 72–90% of the base system's performance.

C. LAMMPS Performance

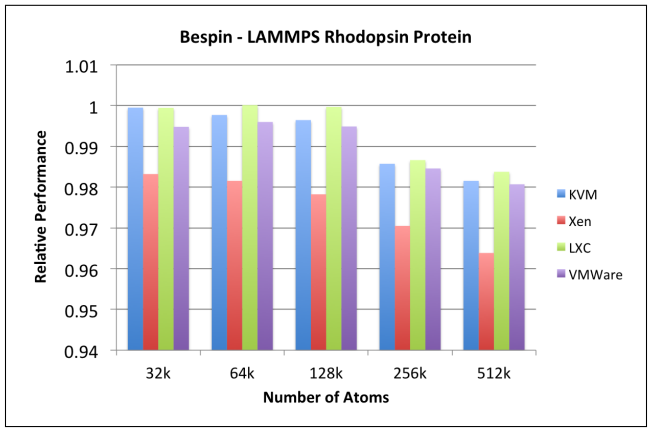

Figure 7: LAMMPS Rhodopsin benchmark relative performance for Bespin system. Higher is better.

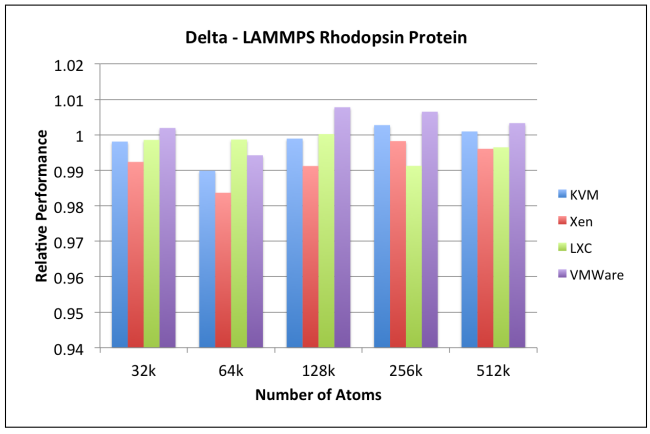

Figure 8: LAMMPS Rhodopsin benchmark relative performance for Delta system. Higher is better.

In Figures 7 and 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.

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. Nevertheless, further research is needed to characterize virtualization overhead on additional heterogeneous applications.

D. LULESH Performance

In Figure 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.

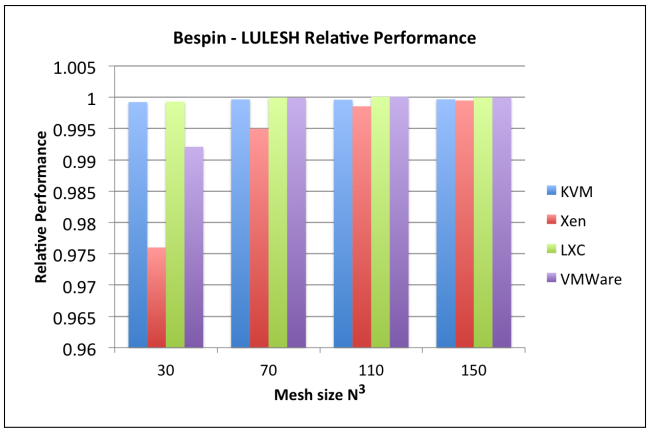

Figure 9: LULESH relative performance on Bespin. Higher is better.

V. Lessons Learned

Virtualizing performance-critical workloads has always proven controversial, whether the workload is CPU-only [22] 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 correctable, 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 GPU 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. While this study was limited to single node performance, KVM's support for SR-IOV, offers a path towards high performance GPU-enabled virtual clusters.

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 GPU 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.

VI. 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 [23], but open questions remain.

References

[1] 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.

[2] 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, June 2012.

[3] V. Gupta, A. Gavrilovska, K. Schwan, H. Kharche, N. Tolia, V. Talwar, and P. Ranganathan, "GViM: GPU-accelerated virtual machines," in *HPCVirt '09 Proceedings of the 3rd ACM Workshop on System-level Virtualization for High Performance Computing*, 2009, pp. 17–24.

[4] 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, Ed. Springer, 2010, vol. 6271, pp. 379–391.

[5] 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.

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

[7] 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 Symp.(OLS)*, 2007, pp. 9–20.

[8] 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.

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

[10] 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.

[11] 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.

[12] G. C. Fox, "The futuregrid project," Web page. [Online]. Available: http://portal.futuregrid.org

[13] 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.

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

[15] 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.

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

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

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

[19] 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.

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

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

[22] 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)*. IEEE, July 2011.

[23] 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.