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

Andrew J. Younge[1], John Paul Walters[2], Stephen P. Crago[2], Geoffrey C. Fox[1]

[1] School of Informatics and Computing
Indiana Univerisity
Bloomington, IN 47408
{ajyounge,gcf}@indiana.edu

[2] Information Sciences Institute
University of Southern California
Arlington, VA 22203
{jwalters,crago}@isi.edu

Abstract

Cloud infrastructure-as-a-Service paradigms have recently shown their utility for a vast array of computational problems, ranging from advanced web service architectures to high throughput computing. However, many scientific computing applications have been slow to adapt to virtualized cloud frameworks. This is due to performance impacts of virtualization technologies, coupled with the lack of advanced hardware support necessary for running many high performance scientific applications at scale.

By using KVM virtual machines that leverage both Nvidia GPUs and InfiniBand, we show that molecular dynamics simulations with LAMMPS and HOOMD run at near-native speeds. This experiment also illustrates how virtualized environments can support the latest parallel computing paradigms, including both MPI+CUDA and new GPUDirect RDMA functionality. Specific findings show initial promise in scaling of such applications to larger production deployments targeting large scale computational workloads.

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 [20]. 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[11], 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 [26].

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. While some of the first efforts to investigate HPC applications on cloud infrastructure such as in the DOE Magellan project [39] documented many shortcomings in performance, recent efforts have proven to be a qualified success [25, 40], 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 [18] 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 [38] as

VEE '15, March 14–15, 2015, Istanbul, Turkey..

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

well as InfiniBand fabrics for high performance networking and I/O [21, 27].

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 can see the possibility of a high performance cloud infrastructure 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 [4], and OpenStack supports heterogeneity using flavors [7]. 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.

First, we describe the background and related work necessary for high performance virtualization. Then, we describe a heterogeneous cloud platform, based on OpenStack. This effort has been under development at USC/ISI since 2011 [13]. 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 [6, 28] 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.

2. Background and Related Work

Virtualization technologies and hypervisors have seen widespread deployment in support of a vast array of applications. This ranges from public commercial Cloud deployments such as Amazon EC2 [8, 17], Microsoft Azure [19], and Google's Cloud Platform [5] 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 to support various HPC solutions has been studied with mixed results. In [40], 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 [25]. Other studies have specifically looked at interconnect performance in virtualization and found the best-case scenario to be lacking [30] 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 [31], AMD's IOMMU [9] from x86_64 architectures, and even more recently ARM System MMU [23]. 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.

2.1 GPU Passthrough

Nvidia GPUs comprise the single most common accelerator in the Nov 2014 Top 500 List [14] 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 [15], 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. Another mechanism of using GPUs in VMs is hypervisor based virtualization [35].

Recently efforts have been seen to support such GPU accelerators within VMs using IOMMU technologies, with implementations now available with KVM [38], Xen [36, 41] and VMWare [37]. 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[37]. 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.

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 Virtualization (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 [24], 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 [21, 32]. 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 [27]. However, real application performance has not yet been well understood until now.

2.3 GPUDirect

NVIDIA's GPUDirect technology was introduced to reduce the overhead of data movement across GPUs [1, 34]. GPUDirect supports both networking as well as peer-to-peer interfaces for single node multi-GPU systems. The most recent implementation 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.

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 [3]) 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 [6]. In our previous work, we have demonstrated the ability to rapidly provision GPU, bare metal, and other heterogeneous resources within a single cloud [13]. Building on this effort we have added support for GPU passthrough to OpenStack as well as prototyped SR-IOV support for ConnectX-2 and ConnectX-3 Infiniband devices. Mellanox has since added an OpenStack InfiniBand networking plugin for OpenStack's Neutron service [2]. While OpenStack supports services for network-

ing (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 [13]. 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.

4. Benchmarks

We selected two molecular dynamics (MD) applications for evaluation in this study: LAMMPS and HOOMD [10, 29]. 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* [12] 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 [12]), 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.

5. Experimental Setup

Using two molecular dynamics tools, LAMMPS[29] and HOOMD [10], 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.

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 1. While our implementation is specific to the KVM hypervisor, this setup represents a design that can be hypervisor agnostic.

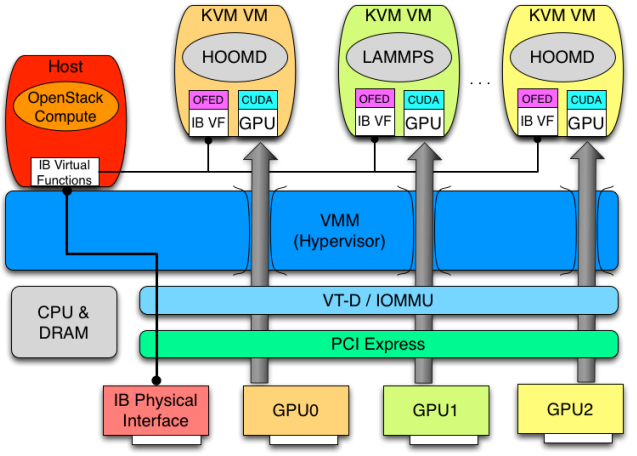

Figure 1. Node PCI Passthrough of GPUs and InfiniBand

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 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.

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. 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.

6.1 LAMMPS

Figure 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

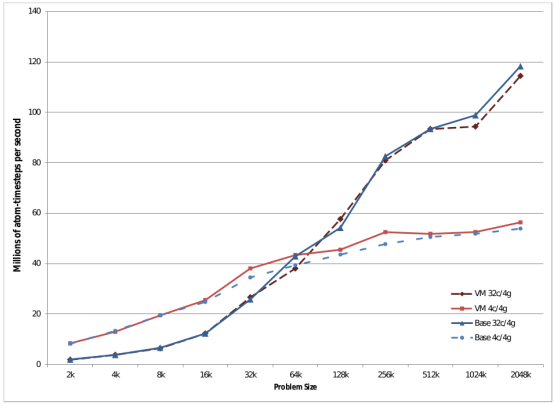

Figure 2. LAMMPS LJ Performance

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 32 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 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. We also see the gap in performance 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

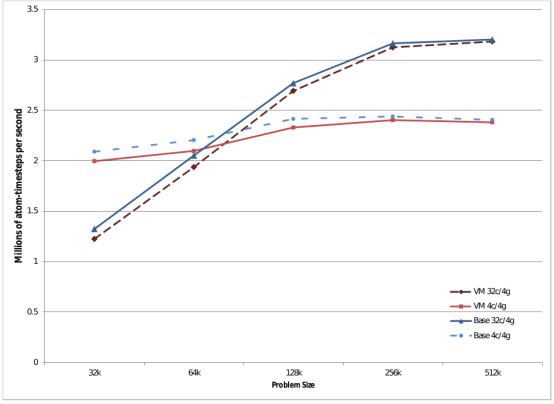

Figure 3. LAMMPS RHODO Performance

a virtualized MPI+CUDA implementation could scale to a larger computational resource with similar success.

6.2 HOOMD

In Figure 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 4 we can see that HOOMD simulations, both with and without GPUDirect, perform very 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.

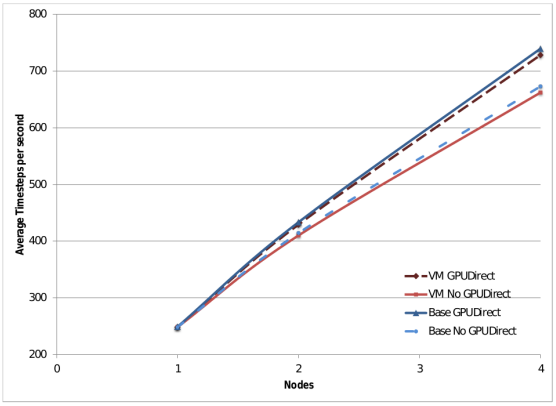

Figure 4. HOOMD LJ Performance with 256k Simulation

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 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 [1], 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, assuming system jitter can be minimized [33]. Specifically, it would be possible to expand such computational problems to a larger deployment in FutureGrid [16], Chameleon Cloud [22], 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.

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.

Acknowledgments

This work was developed with support from the National Science Foundation (NSF) under grant #0910812 to Indiana University and with support from the Office of Naval Research under grant #N00014-14-1-0035 to USC/ISI. Andrew J. Younge also acknowledges support from The Persistent Systems Fellowship of the School of Informatics and Computing at Indiana University.

References

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

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

[3] 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].

[4] Aws high performance computing. http://aws.amazon.com/hpc/, . Last Access Nov. 2014.

[5] Google cloud platform. https://cloud.google.com/, . Last Access Nov. 2014.

[6] Openstack cloud software. http://openstack.org, . Last Access Nov. 2014.

[7] Openstack flavors. http://docs.openstack.org/openstack-ops/content/flavors.html, . Last Access Nov. 2014.

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

[9] AMD. AMD i/o virtualization technology (IOMMU) specification. Technical report, AMD Corporation, 2009.

[10] 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*, volume 1, page 18008, 2010.

[11] 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*, 53 (4):50–58, Apr. 2010. ISSN 0001-0782.

[12] 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, Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, 2006.

[13] 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 *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, pages 378–385. IEEE, 2011.

[14] J. Dongarra, H. Meuer, and E. Strohmaier. Top 500 supercomputers. website, November 2014.

[15] 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*, pages 1–10. IEEE, 2011.

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

[17] 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*, pages 94–103. ACM, 2008.

[18] N. Huber, M. von Quast, M. Hauck, and S. Kounev. Evaluating and modeling virtualization performance overhead for cloud environments. In *CLOSER*, pages 563–573, 2011.

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

[20] 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.

[21] 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*, pages 385–392. IEEE, 2013.

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

[23] A. Limited. Arm system memory management unit architecture specification. Technical report, ARM Limited, 2013.

[24] 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*, pages 1–12, April 2010. .

[25] 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*, Euro-Par'11, pages 436–445, Berlin, Heidelberg, 2012. Springer-Verlag.

[26] 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*, page 39. ACM, 2014.

[27] 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 In-*

ternational Conference on Cloud Computing (CLOUD 2014), Anchorage, AK, 2014. IEEE.

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

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

[30] 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.*, 40(2):55–60, Oct. 2012. ISSN 0163-5999. . URL `http://doi.acm.org/10.1145/2381056.2381071`.

[31] M. Righini. Enabling intel® virtualization technology features and benefits. Technical report, Intel Corporation, 2010.

[32] 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*, pages 943–948, 2014.

[33] S. Seelam, L. Fong, A. Tantawi, J. Lewars, J. Divirgilio, and K. Gildea. Extreme scale computing: Modeling the impact of system noise in multicore clustered systems. In *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pages 1–12, April 2010. .

[34] 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 infiniband—a new model for gpu to gpu communications. *Computer Science-Research and Development*, 26(3-4):267–273, 2011.

[35] Y. Suzuki, S. Kato, H. Yamada, and K. Kono. Gpuvm: why not virtualizing gpus at the hypervisor? In *Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference*, pages 109–120. USENIX Association, 2014.

[36] K. Tian, Y. Dong, and D. Cowperthwaite. A full gpu virtualization solution with mediated pass-through. In *Proc. USENIX ATC*, 2014.

[37] 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*, HPC '14, pages 2:1–2:8, San Diego, CA, USA, 2014. Society for Computer Simulation International. URL `http://dl.acm.org/citation.cfm?id=2663510.2663512`.

[38] 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, 2014. IEEE.

[39] K. Yelick, S. Coghlan, B. Draney, R. S. Canon, et al. The magellan report on cloud computing for science. Technical report, US Department of Energy, 2011.

[40] 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, 2011. IEEE.

[41] 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*, Pheonix, AZ, 05/2014 2014. IEEE, IEEE.