A Case for Dual Stack Virtualization: Consolidating HPC and Commodity Applications in the Cloud

Brian Kocoloski Jiannan Ouyang John Lange
{briankoco,ouyang,jacklange}@cs.pitt.edu
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260

ABSTRACT

With the growth of Infrastructure as a Service (IaaS) cloud providers, many have begun to seriously consider cloud services as a substrate for HPC applications. While the cloud promises many benefits for the HPC community, it currently does not come without drawbacks for application performance. These performance issues are generally the result of resource contention as multiple VMs compete for the same hardware. This contention culminates in cross VM interference whereby one VM is able to impact the performance of another. For HPC applications this interference can have a dramatic impact on scalability and performance. In order to fully support HPC applications in the cloud, services need to be available that prevent cross VM interference and isolate HPC workloads from other users. As a means to achieve this goal, we propose a dual stack approach to IaaS cloud services that utilizes multiple concurrent VMMs on each node capable of partitioning local resources in order to provide performance isolation. Each partition can then be managed by a specialized VMM that is designed specifically for either an HPC or commodity environment. In this paper we demonstrate the use of the Palacios VMM, a virtual machine monitor specifically designed for HPC, in concert with KVM to provide a partitioned cloud platform that is capable of hosting both commodity and HPC applications on a single node without interference. Furthermore, our results demonstrate that running KVM and Palacios in parallel allows an HPC application to achieve isolated and scalable performance while sharing hardware resources with commodity VMs.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design

General Terms

Design, Performance

Keywords

Virtual machine monitors; high performance computing; cloud computing

1. INTRODUCTION

Cloud Computing holds great promise for High Performance Computing (HPC), and accordingly a large amount of work has explored the use of current cloud service architectures for running HPC applications [20, 21, 19]. While the allure of cloud based HPC systems is very compelling, there are still a number of issues that prevent the cloud from becoming a truly viable HPC platform. In particular, application performance has been found to suffer from competing workloads [5], randomized layouts and node assignments [10], as well as competing network flows. All of these issues arise from the fact that HPC applications are forced to share and compete for resources along with a wide variety of other commodity applications. Unfortunately, the presence of these competing workloads is critical to the success of the cloud model, which relies on the economics of leveraging shared resources. While this inherent tension has so far acted as a barrier to the deployment of

SOCC'12, October 14-17, 2012, San Jose, CA USA

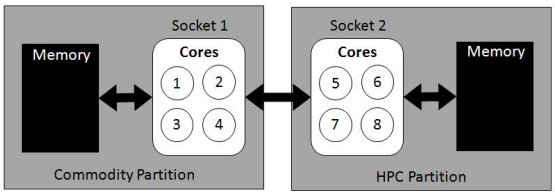


Figure 1: High level overview of node partitioning.

HPC applications in the cloud, we claim that it can be overcome with a dual stack virtualization architecture that is able to accommodate both HPC and commodity users simultaneously on the same hardware.

In this paper we classify HPC applications as large scale and tightly coupled parallel computational kernels that operate in the bulk synchronous parallel (BSP) model. While parallel programming models more suitable to a loosely coupled cloud architectures are emerging for both commodity and HPC environments [7, 4], it is likely that the BSP model will remain the preferred environment for a large class of HPC simulation codes. Therefore, in order for the cloud to become a viable platform for general HPC it must be able to provide an environment suitable for this class of application. Unfortunately, VMM architectures most often used by cloud providers are ill-suited to provide the environment necessary for HPC. This is primarily due to the fact that these architectures are designed for use in commodity environments where goals such as consolidation and resource efficiency are just as, if not more, important than raw performance. While this focus is acceptable for most commodity workloads, it is not acceptable for HPC applications that feature sustained loads requiring full resource utilization as well as large scale distributed synchronization. For these applications, it is much more important that the underlying VMM architecture provide a consistent level of performance, ideally with as little overhead as possible.

In order to avoid the problems associated with competing workloads, many cloud providers have sacrificed efficiency and profitability. Example approaches include the prevention of oversubscribed resources, over provisioning by a wide margin, and providing physically separate infrastructures for HPC specific workloads. While all of these approaches do work to a certain degree, they do so by negating many of the advantages of the cloud model. Ideally, a cloud service provider would be able to use a single infrastructure to serve both commodity and HPC users, while maximizing profits by consolidating resources as much as possible. To reach that goal however requires a cloud architecture that is able to serve both commodity and HPC users equally well while hosting them on the same infrastructure.

In this paper we propose a dual stack approach that utilizes two different VMM architectures in order to allow HPC and commodity VMs to coexist in the same cloud environment in a way that preserves the consolidation capabilities of commodity clouds while still providing the performance necessary for HPC applications. In our proposed approach, a commodity cloud node is partitioned at run time to provide an isolated HPC environment running a specialized VMM architecture. The HPC partition consists of reserved hardware resources that are allocated in a coarse-grained manner and dedicated to the HPC partition. These HPC resources are then managed directly by the HPC VMM without interference from the rest of the system. Figure 1 illustrates a high level view of this partitioning. By assuming resource management duties from the underlying OS, the HPC partition is able to isolate its resources from competing loads on the rest of the system thereby preventing commodity workloads from interfering with HPC applications.

For this work we used (slightly) modified versions of both the Palacios HPC VMM [17] as well as KVM. Our modifications enable both VMMs to separately virtualize disjoint subsets of resources on a physical machine. These modifications allow both Palacios and KVM to execute concurrently on separate local resources, and so partition a cloud node with both HPC (Palacios) and commodity (KVM) zones. Using this approach, we will show that even as commodity workloads saturate a large subset of a machine's resources, the HPC zone remains capable of providing consistent and isolated performance to HPC applications.

While the design and architecture of the Palacios VMM have been described elsewhere [17, 16], we provide an overview of its features for completeness. Palacios provides modular virtualization functionality to x86 based architectures, such as the Cray XT, as well as commodity HPC systems. Our previous work with the Palacios VMM focused on providing a high performance virtualization layer for lightweight kernels (LWKs) and other supercomputing class operating systems, resulting in a demonstration that virtualization can be used on modern supercomputers with minimal overhead. Recently Palacios has been integrated with Linux [16][1], and has been shown to provide an HPC environment capable of outperforming the native Linux user environment [15].

[1]Linux support was introduced with Palacios version 1.3

To better illustrate our proposal we include an example scenario for a dual stack enabled cloud provider. In most cases the default behavior would not change as commodity VMs are hosted in the cloud infrastructure. However, when an HPC workload is introduced the cloud infrastructure is notified that HPC resources are needed. The infrastructure will then reserve a collection of coarse-grained resources, ideally consisting of CPU sockets as well as memory local to those sockets. Reserving these resources will have the effect of removing them from the commodity VMM and OS's resource pools. After a suitable set of resources has been reserved by the system, a specialized HPC VMM environment will be initialized to directly manage those resources. The HPC workload can then be instantiated across the reserved HPC infrastructure where it will run until it completes, at which point the HPC VMM will be shutdown and the reserved resources returned for usage by commodity users.

2. RELATED WORK

Much research has been done on investigating the feasibility of running scientific applications on commodity cloud architectures. Specifically, [10, 20, 19, 21, 13] have all experimented with running performance critical applications and benchmarks on the Amazon EC2 compute cloud. Largely, the results of these experiments have shown that EC2 is unable to provide a completely satisfactory HPC environment, particularly for communication-intensive applications, due to its lack of a high speed interconnect. Of course, the vast majority of commodity workloads neither require nor would significantly benefit from such an interconnect, and so there has been little motivation for their inclusion in commodity cloud architectures. We do not believe, however, that this is a crippling issue for the prospects of HPC in the cloud; if the other technical challenges currently hampering the deployment of HPC applications in the cloud can be overcome, we believe that both the technical and financial motivations for high speed interconnects in cloud architectures will present themselves.

In [5], the authors presented a further technical difficulty with running HPC jobs in commodity clouds in the form of resource contention. Their work presented a resource monitoring and management scheme that adapted to changing resource demands in the cloud with the goal of providing scalability and reliability. In our work, we show that virtualization, whose benefits to reliability are well documented, can directly address the issue of resource contention in commodity clouds. We believe that our approach would be compatible with the schemes presented therein.

Other approaches similar to our work have appeared in system designs such as Disco [6] and Barrelfish [2]. The possibility of using virtual machines to optimize performance and OS behavior on NUMA architectures was explored in Disco as a means of deploying commodity and specialized OS environments on a large scale system. The benefits of the Disco system centered around the fact that global resource management decisions could be centralized inside a small and optimized codebase outside of the full OS kernel. The barrelfish OS took the approach of fully partitioning the entire OS into a distributed system with restricted sharing and coordination using only explicit message passing. The Palacios architecture can be viewed as a compromise between each of these approaches. In contrast to Disco Palacios provides independent management of separate partitions of resources, avoiding the overheads inherent in providing a generalized resource management layer. Conversely, Palacios does not go to the same extremes of Barrelfish in adopting a full share nothing architecture. Palacios does include a large amount of replicated state to siginificantly reduce cross core coordination and synchronziation, but does not require explicit message passing for the small number of cases where cross core communication is required.

3. BACKGROUND

The success of IaaS cloud services is largely the result of the degree to which existing VMM architectures have been able to minimize the overhead associated with resource virtualization. Current virtual environments are capable of providing near native computational and I/O performance, and there is much work being done to fully close the performance gap between native and virtual environments [8, 17]. The elimination of virtualization overhead provides the benefits of cloud based computing to a broader class of applications and users. In particular, HPC users are eager to explore cloud based computation due to the inherent cost of acquiring and maintaining their own local HPC resources, as well as the difficulty in sharing a relatively small number of centralized HPC systems. Cloud based HPC environments promise instantaneous access to computational resources at arbitrary scale with only a fraction of the cost. It is no surprise then that many in the HPC community are eager to exploit the potential of the cloud.

Unfortunately, while current VMM architectures certainly have the potential to deliver acceptable performance for HPC, in commodity cloud environments this potential is difficult to realize. This shortcoming is due to the fact that the underlying reason cloud resources are less expensive is that they are a shared resource hosting multiple users. This multi-tenancy results in resource contention as multiple VM environments access the underlying hardware at the same time. To address this issue, many current cloud environments prevent the oversubscription of hardware resources. However, while this helps, it does not completely eliminate the problem. The resulting contention of both hardware and OS resources results in cross VM

VMM	# of NPFs	% of exits	Avg	Stdev
KVM	3,265,156	52%	8804	5232
Palacios	1,872,017	50%	10876	2685

Figure 2: Behavior of nested page fault (NPF) handler in KVM and Palacios. The 2nd column shows total number of NPFs that occurred, the 3rd column gives the percentage of total VM exits indicated as NPFs, the 4th column shows the average execution time of the NPF handler in CPU cycles, and the 5th column gives the standard deviation of the execution time.

interference, where the behavior of one VM has a negative impact on another locally hosted VM. This interference is especially troublesome for tightly coupled parallel HPC applications, as localized interference has a tendency to propagate throughout the rest of the application due to its highly synchronized behavior.

We claim that the existence of cross VM interference is due to the underlying architecture of the VMM, and exacerbated by the fact that most VMMs are designed to meet the goals of a commodity environment. In these commodity virtualized environments a premium is placed on resource efficiency achieved through server consolidation. As a result, commodity class VMMs are designed to achieve as much consolidation as possible while not overly degrading performance, and also to strive to be a "good citizen" in the system by not monopolizing resources or negatively impacting other processes. This has resulted in many features being added to VMM architectures such as VM page sharing and merging, swapping out pages in the VMM, and on demand page allocation. While these features can easily be disabled for an HPC environment, they place additional requirements on the VMM architecture that cannot be disabled. For example, while the KVM memory system does allow the use of large preallocated pages via the HugeTLBfs mechanism, it contains other features such as a single lock on the VM's memory management layer requiring that operations in the nested page fault handler be serialized.

In contrast the Palacios Virtual Machine Monitor [16] is a virtualization architecture designed explicitly for HPC environments. By focusing on HPC, the Palacios VMM is able to eschew many of the features deemed necessary for commodity environments and instead focus on the requirements needed for scalable HPC performance. In effect Palacios is designed to be a "bad citizen" in the system, in that it takes control over entire segments of the hardware and refuses to share them with other system components. While such an approach might seem heavy handed and inefficient, it is suitable in a dedicated HPC context where scalable performance is critical and interference from other system components can have dramatic effects.

Palacios integrates with Linux via the kernel module interface and so does not require any modifications to the kernel itself. This ensures compatibility with a wide range of kernel versions[2], and for this work we used the stock 2.6.40 kernel distributed with Fedora 15. Virtualized memory is one of the primary sources of overhead in virtual environments, and so Palacios takes particular care in how it manages a VM's memory image. In particular, Palacios uses large contiguous physical memory regions (on the order of Gigabytes) in order to optimize both hardware paging performance as well as software management functions. To achieve this on Linux, Palacios uses the *Hot Removable memory* feature available in recent kernels. Hot removable memory allows for large regions (usually 128MB) of memory to be offlined and removed from the Linux memory management system. However, while this memory is lost to the Linux kernel, it remains active and addressable by the system, so Palacios is able to instantiate its own management layer on top of this offlined memory. This allows Palacios to manage large contiguous memory regions directly, thus avoiding the overheads and contention present in the Linux memory system.

To illustrate the different performance characteristics of commodity and HPC VMM architectures, we instrumented both KVM and Palacios to measure VMM overheads during execution. We then collected results from a set of HPC benchmarks (described in detail in Section 5) running across 4 cores on a single node. In particular, we analyzed the behavior of each VMMs virtualized memory architecture. A summary of the results of this analysis are shown in Figure 2. Several points can be taken away from the results. First, while the average execution time for the nested page fault handler is 25% larger in Palacios, it is much more stable, with a standard deviation that is half that of KVMs exit handler. Second, and most importantly, the total number of nested page faults in Palacios is almost half those of KVM. This means that Palacios' memory layer executes much less frequently than KVM, and while its actual execution is slightly slower it has less variance.

4. DUAL STACK VIRTUALIZATION

The view we present in this paper is that HPC and commodity applications can in fact be effectively consolidated inside a common cloud infrastructure. However, in order to achieve this it is necessary to recognize that the requirements for these classes of applications are significantly different. As a result, a system designed for one application class will not be ideally

[2]Confirmed Versions: 2.6.32 through 3.2

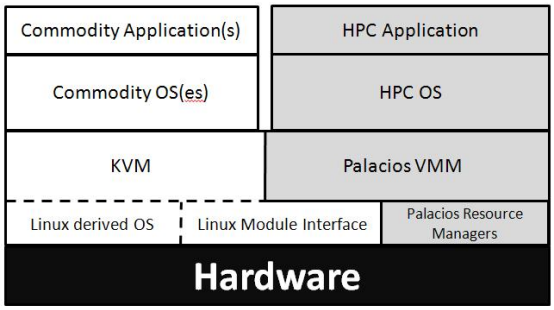


Figure 3: A dual stack architecture

suited for the other. Based on this observation, we claim that a dual stack approach is needed whereby a single system can be partitioned to provide isolated virtual environments for different application classes. A high level example of this approach is shown in Figure 1. In this case, a multi-socket system is partitioned in such a way that an HPC application is executed on one socket in an HPC virtual context, while commodity applications are simultaneously executed on the other socket. The partitioning of the system is done in such a way that memory traffic is restricted as much as possible to the memory contained in the NUMA zone of the local socket. Such an arrangement ensures that cross VM interference is minimized at both the hardware and software layers of the system.

Based on Figure 1, it might appear that such an approach could be easily achieved with currently available systems and software. While it is true that current commodity architectures can be configured in this way, as stated earlier the inherent design characteristics of current commodity VMMs limits the isolation that can be achieved even if it is configured as shown. Conversely, using only an HPC specific VMM would prevent the system from optimizing its behavior to increase consolidation in the commodity zone. In order to effectively perform this partitioning, each zone must have its own underlying system software layer that is capable of providing the optimal behavior. Based on this observation we propose the use of a dual stack approach to virtualization, that relies on two VMMs that handle a specific partition of the machine.

To demonstrate the effectiveness of this approach we have evaluated such a configuration using modified versions of Palacios and KVM. Figure 3 shows the high level architecture of our proposed system. KVM is used to host commodity environments and was modified to restrict it's execution to a subset of cores in the system. We used the Palacios VMM to host HPC VMs executing on dedicated cores and memory managed directly by internal Palacios resource managers. Palacios was also modified to execute on a subset of cores. The modifications required to both KVM and Palacios were minor and confined to the initialization and de-initialization code paths. The changes to KVM consisted of ∼50 lines of C that assigned a configurable CPU mask to a VM's backing kernel threads as they were created. Palacios already implemented CPU masks when creating VMs, so the changes required for it were simply to allow the configuration of those masks by the administrator.

5. EVALUATION

To demonstrate the potential of a dual stack virtualization approach, we ran a set of experiments to compare the dual VMM configuration (Palacios and KVM) against configurations using a single commodity VMM architecture (both KVM and Xen). The goal of this evaluation was to determine the overheads that result when a single commodity VMM is used to co-host commodity and HPC workloads on the same machine. To isolate the overheads to the VMM architecture, the system was configured as shown in Figure 1. This ensured that commodity and HPC applications executed on separate sockets with VM memory backed by each socket's local memory. Therefore, any resulting overheads are the result of contention in the VMM and host OS layers.

Guest Configurations. For each experiment we used a copy of the same VM disk image, based on a Fedora 15 installation. As each VMM supports raw disk images we were able to use the same file for each configuration. The VM executing the HPC benchmarks was configured with 4 cores and 2GB of memory. Each HPC application was configured to use large pages via HugeTLBfs. Conversely the commodity VM image was configured to use a varying number of processor cores to evaluate the effect of increasing commodity workloads. To simulate a commodity workload we conducted a parallel build of the Linux kernel configured to saturate the available cores. The commodity environment was otherwise unmodified from a standard Fedora configuration.

Host configurations. The host environments were similarly configured according to best practices [12], as well as partitioned into both commodity and HPC zones. Each VMM was configured to use large nested pages, though the underlying mechanism for allocating these pages varied according to the VMM architecture. Each guest core was pinned to a dedicated host core in the appropriate performance partition. Memory was also restricted to the local memory of each socket and preallocated using mechanisms specific to each VMM. Both KVM and Palacios used full system virtualization, while Xen was configured to use its optimized Paravirtual extensions (PV-HVM)

Hardware. Each experiment ran on a dedicated Dell R415 server configured with two 6-core Opteron 4174 CPUs and 16GB of memory. The memory layout consisted of 2 NUMA zones equally shared between the processors with memory interleaving disabled. The server was installed with a standard Fedora 15 environment running an unmodified 2.6.40.6-0.fc15.x86_64 kernel.

5.1 Benchmarks

The benchmarks we selected for our evaluation were taken from the Mantevo [11] and Sequoia [1] benchmark sets. These benchmarks are based on actual HPC application architectures and so are designed to exhibit behaviors present in an actual real world workload. We chose these benchmarks to highlight the expected system behavior in a real world scenario.

Mantevo The Mantevo MiniApps from Sandia National Labs consist of a set of skeleton applications that contain the core kernel behavior of large scale HPC applications. While these applications are small, they still exhibit key behaviors of larger scale applications. The four mini-applications we ran from this collection are described below.

- HPCCG: A conjugate gradient solver whose workload is representative of many HPC applications. It was run with a fixed problem size of 100x100x100, and was configured to use OpenMP threads.
- phdMesh: A parallel heterogeneous dynamic mesh application, which exhibits the performance characteristics of the contact search operations in an explicit finite element application. It was executed with the test_mesh.exe binary, with a problem size of 4x6x4, and was configured with MPI support.
- MiniFE: A proxy application for unstructured implicit finite element codes. It was run with a problem size of 100x100x100, and was configured with MPI support.
- MiniMD: A simple proxy for the force computations in a typical molecular dynamics application. It was run with a problem size of 50x50x50, and was configured with MPI support.

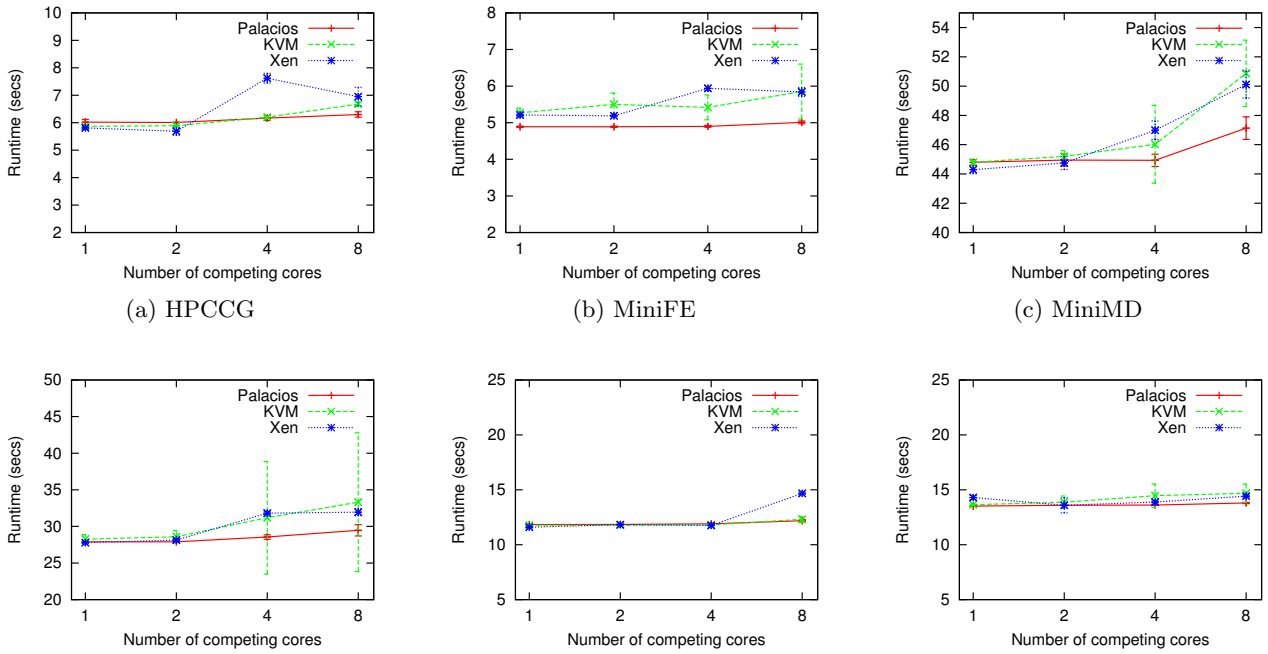
Sequoia The ASC Sequoia Benchmark suite contains a set of real world HPC simulation codes. The two benchmarks that we ran from this package are described further below.

- AMG: An algebraic Mult-Grid linear system solver for unstructured mesh physics packages. It was run with a problem size of 12x12x12, and was configured with MPI support.
- LAMMPS: A classical molecular dynamics simulation code. It was run with a problem size of 20x40x40, and was configured with MPI support.

5.2 Results

For each experiment, we launched exactly one 4 core HPC VM along with varying levels of competing commodity workloads running inside 1-2 separate commodity VMs. All of the benchmarks were executed ten times with each of the following commodity configurations: one 1-core commodity VM, one 2-core commodity VM, one 4-core commodity VM, and two 4-core commodity VMs (8 cores total).

The results for each benchmark's run times are shown in Figures 4 and 5. As can be seen, Palacios typically provides comparable or slightly better performance than KVM and Xen for one or two competing cores. In particular, MiniFE appears to be highly succeptible to cross VM interference when running on either KVM or Xen, but shows substantially better performance on Palacios. As the number of competing cores scales up to four, however, we begin to see separation. For three of the six benchmarks (MiniFE, MiniMD, and phdMesh), we see that Palacios provides a better environment than KVM. Furthermore, the impact of the commodity workload on Xen can be seen in four of the six benchmarks (HPCCG, MiniFE, MiniMD, phdMesh), where Palacios provides significantly better performance. AMG was the only benchmark in which KVM or Xen outperformed Palacios, but the performances were nearly identical. On average, the benchmarks ran 4.37% and 8.68% slower on KVM and Xen respectively than they did on Palacios. These results show that as workloads begin to saturate the commodity cores, Palacios is able to provide greater isolation to the HPC application than either KVM or Xen.



(a) HPCCG (b) MiniFE (c) MiniMD

(d) phdMesh (e) AMG (f) LAMMPS

Figure 4: Results of the isolation experiments. As competing workloads scale up Palacios proves to better isolate the HPC partition than either KVM or Xen

		Mean / Stdev					
VMM	**Competing Cores**	**HPCCG**	**MiniFE**	**MiniMD**	**phdMesh**	**AMG**	**LAMMPS**
Palacios	4	6.17 / 0.09	4.89 / 0.03	44.93 / 0.41	28.54 / 0.31	11.90 / 0.09	13.61 / 0.04
KVM	4	6.20 / 0.04	5.42 / 0.34	46.03 / 2.66	31.18 / 7.70	11.81 / 0.05	14.46 / 1.07
Xen	4	7.62 / 0.18	5.94 / 0.08	46.99 / 0.63	31.81 / 0.24	11.75 / 0.04	13.89 / 0.10

Figure 5: Averages and standard deviations of the isolation experiments with four competing commodity cores, which is a representative cloud workload. Palacios provides better, more consistent performance than KVM and Xen for almost every benchmark

In order to understand how significant the hardware partitioning was to HPC performance, we configured the machine with 2 competing VMs with 4 cores each. This workload configuration forced us to shrink the HPC partition down from an entire socket to a subset of cores on that socket. Although these test cases do not necessarily represent an ideal dual stack workload, as they require the crossing of NUMA boundaries, they nevertheless do give interesting information about the abilities of the VMMs in question to perform when the system is heavily loaded. The results of these experiments show that Palacios is still capable of providing consistent performance in most cases, but in some cases does experience degradation as the result of hardware contention. For each benchmark other than AMG, Palacios was at least 5% faster on average than KVM. Similarly, each benchmark other than LAMMPS performed at least 5% faster on Palacios than on Xen. No benchmark performed better on either KVM or Xen than on Palacios. On average, the benchmarks ran 7.65% and 9.75% slower on KVM and Xen respectively than they did on Palacios.

In addition to exhibiting superior performance on average, the benchmarks running on Palacios exhibited much more consistent performance. This can be seen by examining the standard deviations reported in Figure 5. Compared to their standard deviations on Palacios, the same benchmarks running on Xen experienced standard deviations that were 31%, and 16% higher, for four and eight competing cores respectively, than they did on Palacios. The results are even more drastic for KVM; again compared to Palacios, the same benchmarks experienced average increases in standard deviations of 1122%, and 636%, for four and eight competing cores. As consistency is typically a good indicator of scalability, these results strongly indicate that Palacios will scale much better than either KVM or Xen in the presence of commodity workloads.

6. OPEN ISSUES

While our initial results show the effectiveness of a dual stack approach on a single compute node, there are many open questions that still need to be answered. Currently the largest issue with deploying HPC applications in the cloud is the lack of adequate network infrastructure to handle the communication loads and also to provide reliable bandwidth in the face of competing flows. Any performance improvement provided by a dual stack virtualization approach could easily be overwhelmed by a poorly configured network. Indeed, our own experience has shown that scaling a Palacios hosted HPC application across a dedicated Gigabit switch results in any performance improvement being swamped by delays introduced by the network. While this work does not address this, the networking situation is improving. Cloud providers are beginning to recognize the need for high performance network fabrics, and traditional HPC vendors are beginning to move in this direction as well. This can be seen in the availability of 10 Gigabit networks and other specialized technologies such as RoCEt [18].

Other approaches are also being taken to provide optimized network architectures for cloud services. Novel network architectures such as [9, 14] are being introduced to provide better scalability and performance for cloud infrastructures. Additional work is focusing on providing reservable network paths based on technologies such as optical networks [3]. In parallel with this work are advances in improving virtual I/O performance [8], particularly for networking [22]. We believe that as the networking infrastructure continues to evolve, the capabilities needed to provide reliable and consistent network resources will become available.

Tied closely to the issue of network resources is the problem of how to handle the placement of partitions in our dual stack approach. The layout of a distributed set of HPC applications has significant ramifications on application performance as it determines what the underlying network characteristics will be for any collective communication. We note that while this is a significant issue, it is outside the scope of our work. In addition, it is relevant for any cloud architecture that hosts HPC workloads, and our dual stack approach could easily fit into any proposed scheme with little to no modification.

7. CONCLUSION AND FUTURE WORK

In this paper we have made the case for a dual stack approach to virtualization in the cloud. While there is significant interest in running both HPC and commodity workloads in the cloud, there is currently no unified architecture that satisfies the needs of both classes of users. A dual stack approach would allow a cloud provider to dynamically configure a cloud service at runtime to directly address the needs of both HPC and commodity workloads as they are deployed. We have undertaken a preliminary performance study of such a system configuration using KVM and the Palacios VMM. Our results show that it is possible to partition a single node such that both commodity and HPC applications can execute concurrently without negatively impacting the other's performance. While our evaluations are restricted to a single node, we believe that similar partitioning techniques could be used in other parts of the system to provide a scalable dual stack solution.

Based on our results, we intend to further explore the dual stack approach. In particular, we intend to integrate many of the advanced networking approaches with the Palacios VMM in order to demonstrate the potential scalability of a specialized cloud infrastructure for HPC. We also intend to evaluate the performance of the dual stack approach at large scales, as well as to investigate other mechanisms for alleviating the remaining overheads introduced by Linux.

8. REFERENCES

[1] ASC Sequoia Benchmark Codes `https://asc.llnl.gov/sequoia/benchmarks/`.

[2] Baumann, A., Barham, P., Dagand, P.-E., Harris, T., Isaacs, R., Peter, S., Roscoe, T., Schüpbach, A., and Singhania, A. The Multikernel: A New OS Architecture for Scalable Multicore Systems. In *Proc. 22nd Symposium on Operating Systems Principles (SOSP)* (2009).

[3] Bazzaz, H. H., Tewari, M., Wang, G., Porter, G., Ng, T. S. E., Andersen, D. G., Kaminsky, M., Kozuch, M. A., and Vahdat, A. Switching the Optical Divide: Fundamental Challenges for Hybrid Electrical/Optical Datacenter Networks. In *Proc. 2nd ACM Symposium on Cloud Computing (SOCC)* (2011).

[4] Blelloch, G. E. NESL: A Nested Data-Parallel Language (Version 2.6). Tech. rep., Carnegie Mellon University, 1993.

[5] Brandt, J., Gentile, A., Mayo, J., Pebay, P., Roe, D., Thompson, D., and Wong, M. Resource Monitoring and Management with OVIS to Enable HPC in Cloud Computing Environments. In *Proc, 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (2009).

[6] Bugnion, E., Devine, S., and Rosenblum, M. Disco: Running Commodity Operating Systems on Scalable Multiprocessors. In *Proc. 16th Symposium on Operating Systems Principles (SOSP)* (1997).

[7] Dean, J., and Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM 51*, 1 (Jan. 2008).

[8] Gordon, A., Amit, N., Har'El, N., Ben-Yehuda, M., Landau, A., Schuster, A., and Tsafrir, D. ELI: Bare-metal Performance for I/O Virtualization. In *Proc. 17th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (2012).

[9] Greenberg, A., Hamilton, J. R., Jain, N., Kandula, S., Kim, C., Lahiri, P., Maltz, D. A., Patel, P., and Sengupta, S. VL2: A Scalable and Flexible Data Center Network. In *Proc. ACM SIGCOMM* (2009).

[10] He, Q., Zhou, S., Kobler, B., Duffy, D., and McGlynn, T. Case Study for Running HPC Applications in Public Clouds. In *Proc. 19th ACM International Symposium on High Performance Distributed Computing (HPDC)* (2010).

[11] Heroux, M., et al. Welcome to the Mantevo Project Home Page, `https://software.sandia.gov/mantevo`.

[12] IBM. Best Practices for KVM. *White Paper* (Nov. 2010).

[13] Jackson, K. R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., Wasserman, H. J., and Wright, N. J. Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. In *Proc. 2nd International Conference on Cloud Computing Technology and Science (CLOUDCOM)* (2010).

[14] Kim, C., Caesar, M., and Rexford, J. Floodless in Seattle: A Scalable Ethernet Architecture for Large Enterprises. In *Proc. ACM SIGCOMM* (2008).

[15] Kocoloski, B., and Lange, J. Better Than Native: Using Virtualization to Improve Compute Node Performance. In *Proc. 2nd International Workshop on Runtime and Operating Systems for Supercomputers (ROSS)* (2012).

[16] Lange, J., Dinda, P., Hale, K., and Xia, L. An Introduction to the Palacios Virtual Machine Monitor—Release 1.3. Tech. Rep. NWU-EECS-11-10, Northwestern University, October 2011.

[17] Lange, J. R., Pedretti, K., Dinda, P., Bridges, P. G., Bae, C., Soltero, P., and Merritt, A. Minimal-overhead virtualization of a large scale supercomputer. In *Proc. 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE)* (2011).

[18] Mellanox Technologies. The Case for Infiniband over Ethernet. *White Paper* (2008).

[19] Napper, J., and Bientinesi, P. Can Cloud Computing Reach the Top500? In *Proc. Combined Workshops on Unconventional High Performance Computing Workshop and Memory Access Workshop (UCHPC-MAW)*.

[20] Rehr, J. J., Vila, F. D., Gardner, J. P., Svec, L., and Prange, M. Scientific Computing in the Cloud. *Computing in Science & Engineering 12* (2010), 34–43.

[21] Stantchev, V. Performance Evaluation of Cloud Computing Offerings. In *Proc. 3rd International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP)* (2009).

[22] Xia, L., Cui, Z., Lange, J., Tang, Y., Dinda, P., and Bridges, P. VNET/P: Bridging the Cloud and High Performance Computing Through Fast Overlay Networking. In *Proc. 21st ACM Symposium on High-performance Parallel and Distributed Computing (HPDC)* (2012).