The Who, What, Why, and How of High Performance Computing in the Cloud

Abhishek Gupta,
Laxmikant V. Kale
University of Illinois at
Urbana-Champaign
Urbana, IL 61801, USA
(gupta59, kale)@illinois.edu

Filippo Gioachin,
Verdi March,
Chun Hui Suen,
Bu-Sung Lee
HP Labs, Singapore
firstname.lastname@hp.com

Paolo Faraboschi,
Richard Kaufmann,
Dejan Milojicic
HP Labs
Palo Alto, CA, USA
firstname.lastname@hp.com

***Abstract*—Cloud computing is emerging as an alternative to supercomputers for some of the high-performance computing (HPC) applications that do not require a fully dedicated machine. With cloud as an additional deployment option, HPC users are faced with the challenges of dealing with highly heterogeneous resources, where the variability spans across a wide range of processor configurations, interconnections, virtualization environments, and pricing rates and models.**

In this paper, we take a holistic viewpoint to answer the question – *why* and *who* should choose cloud for HPC, for *what* applications, and *how* should cloud be used for HPC? To this end, we perform a comprehensive performance evaluation and analysis of a set of benchmarks and complex HPC applications on a range of platforms, varying from supercomputers to clouds. Further, we demonstrate HPC performance improvements in cloud using alternative lightweight virtualization mechanisms – thin VMs and OS-level containers, and hypervisor- and application-level CPU affinity. Next, we analyze the economic aspects and business models for HPC in clouds. We believe that is an important area that has not been sufficiently addressed by past research. Overall results indicate that current public clouds are cost-effective only at small scale for the chosen HPC applications, when considered in isolation, but can complement supercomputers using business models such as *cloud burst* and *application-aware mapping*.

***Keywords*-HPC, Cloud, Performance Analysis, Economics**

I. Introduction

Increasingly, some academic and commercial HPC users are looking at clouds as a cost effective alternative to dedicated HPC clusters [1]. *Renting* rather than owning a cluster avoids the up-front and operating expenses associated with a dedicated infrastructure. Clouds offer additional advantages of a) *elasticity* – on-demand provisioning, and b) virtualization-enabled flexibility, customization, and resource control.

Despite these advantages, it still remains unclear whether, and when, clouds can become a feasible substitute or complement to supercomputers. HPC is performance-oriented, whereas clouds are cost and resource-utilization oriented. Furthermore, clouds have traditionally been designed to run business and web applications. Previous studies have shown that commodity interconnects and the overhead of virtualization on network and storage performance are major performance barriers to the adoption of cloud for HPC [1–4]. While the outcome of these studies paints a rather pessimistic view of HPC clouds, recent efforts towards HPC-optimized clouds, such as Magellan [1] and Amazon's EC2 Cluster Compute [5], point to a promising direction to overcome some of the fundamental inhibitors.

Unlike previous works [1–4, 6–9] on benchmarking clouds for science, we take a more holistic and practical viewpoint. Rather than limiting ourselves to the problem – *what* is the performance achieved on cloud vs. supercomputer, we address the bigger and more important question – *why* and *who* should choose (or not choose) cloud for HPC, for *what* applications, and *how* should cloud be used for HPC? In our efforts to answer this research question, we make the following contributions in this work.

- We evaluate the performance of HPC applications on a range of platforms varying from supercomputer to cloud, analyze bottlenecks and the correlation between application characteristics and observed performance, identifying *what* applications are suitable for cloud. (§ III,§ IV)
- We analyze the impact of virtualization on HPC applications and propose techniques, specifically thin hypervisors, OS-level containers, and hypervisor and application-level CPU affinity, to mitigate virtualization overhead and noise, addressing – *how* to use cloud for HPC. (§ V)
- We investigate the economic aspects of running in cloud vs. supercomputer and discuss *why* it is challenging to make a profitable business for cloud providers for HPC compared to traditional cloud applications. We also show that small/medium-scale HPC users are the most likely candidates *who* can benefit from an HPC-cloud. (§ VI)
- Instead of considering cloud as a substitute of supercomputer, we investigate the co-existence of supercomputer and cloud addressing – *how* to use cloud for HPC.(§ VII)

We believe that it is important to consider views of both, HPC users and cloud providers, who sometimes have conflicting objectives: users must see tangible benefits (in cost or performance) while cloud providers must be able to run a profitable business. The insights from comparing HPC applications execution on different platforms is useful for both. HPC users can better quantify the benefits of moving to a

TABLE I: Testbed

Resource	Platform				
	Ranger	Taub	Open Cirrus	Private Cloud	Public Cloud
Processors in a Node	16×AMD Opteron QC @2.3 GHz	12×Intel Xeon X5650 @2.67 GHz	4×Intel Xeon E5450 @3.00 GHz	2×QEMU Virtual CPU @2.67 GHz	4×QEMU Virtual CPU @2.67 GHz
Memory	32 GB	48 GB	48 GB	6 GB	16 GB
Network	Infiniband (1 GB/s)	QDR Infiniband	10GigE internal, 1GigE x-rack	Emulated 1GigE	Emulated 1GigE
OS	Linux	Sci. Linux	Ubuntu 10.04	Ubuntu 10.04	Ubuntu 10.10

cloud and identify which applications are better candidates for the transition from in-house to cloud. Cloud providers can optimize the allocation of applications to their infrastructure to maximize utilization, while offering best-in-class cost and quality of service.

II. Evaluation Methodology

In this section, we describe the platforms which we compared and the applications which we chose for this study.

A. Experimental Testbed

We selected platforms with different interconnects, operating systems, and virtualization support to cover the dominant classes of infrastructures available today to an HPC user. Table I shows the details of each platform. In case of cloud a *node* refers to a virtual machine and a *core* refers to a virtual core. For example, "2 × QEMU Virtual CPU @2.67GHz" means each VM has 2 virtual cores. Ranger [10] at TACC was a supercomputer with a theoretical peak performance of 579 Tera FLOPS[1], and Taub at UIUC is an HPC-optimized cluster. Both use Infiniband as interconnect. Moreover, Taub uses scientific Linux as OS and has QDR Infiniband with bandwidth of 40 Gbps. We used physical nodes with commodity interconnect at Open Cirrus testbed at HP Labs site [11]. The final two platforms are clouds – a private cloud setup using Eucalyptus [12], and a public cloud. We use KVM [13] for virtualization since it has been shown to be a good candidate for HPC virtualization [14].

In case of cloud, most common deployment of multi-tenancy is not sharing individual physical cores, but rather done at the node, or even coarser level. This is even more true with increasing number of cores per server. Hence, our cloud experiments involve physical nodes (not cores) which were shared by VMs from external users, hence providing a multi-tenant environment.

Another dedicated physical cluster at HP Labs Singapore (HPLS) is used for controlled tests of the effects of virtualization (see Table II). This cluster is connected with a Gigabit Ethernet network on a single switch. Every server has two CPU sockets, each populated with a six-core CPU, resulting in 12 physical cores per node. The experiment on the HPLS cluster involved benchmarking on four configuration: physical machines (bare), LXC containers [15], VMs configured with the default emulated network (plain VM), and VMs with pass-through networking (thin VM). Both the plain VM and thin VM run on top of the KVM hypervisor. In the thin VM setup, we enable Input/Output Memory Management Unit (IOMMU) on the Linux hosts to allow VMs to directly access the Ethernet hardware, thus improving the network I/O performance [16].

[1]Ranger was decommissioned in Feb 2013, concurrent with the deployment of a new supercomputer – Stampede at TACC. Ranger was ranked 50 in the November 2012 top500 supercomputer list.

TABLE II: Virtualization Testbed

Resource	Virtualization		
	Phy., Container	Thin VM	Plain VM
Processors in a Node/VM	12×Intel Xeon X5650 @2.67 GHz	12×QEMU Virtual CPU @2.67 GHz	12×QEMU Virtual CPU @2.67 GHz
Memory	120 GB	100 GB	100 GB
Network	1GigE	1GigE	Emulated 1GigE
OS	Ubuntu 11.04	Ubuntu 11.04	Ubuntu 11.04

B. Benchmarks and Applications

To gain insights into the performance of selected platform over a range of applications, we chose benchmarks and applications from different scientific domains and those which differ in the nature, amount, and pattern of inter-processor communication. Moreover, we selected benchmarks written in two different parallel programming environments – MPI [17] and CHARM++ [18]. Similarly to previous work [2, 3, 6, 9], we used NAS Parallel Benchmarks (NPB) class B [19] (the MPI version, NPB3.3-MPI), which exhibit a good variety of computation and communication requirements. Moreover, we chose additional benchmarks and real world applications:

- Jacobi2D – A 5-point stencil kernel to average values in a 2-D grid. Such stencil kernels are common in scientific simulations, numerical linear algebra, numerical solutions of Partial Differential Equations (PDEs), and image processing.
- NAMD [20] – A highly scalable molecular dynamics application and representative of a complex real world application used ubiquitously on supercomputers. We used the ApoA1 input (92k atoms) for our experiments.
- ChaNGa [21] (Charm N-body GrAvity solver) – A cosmological simulation application which performs collisionless N-body interactions using Barnes-Hut tree for calculating forces. We used a 300,000 particle system.
- Sweep3D [22] – A particle transport code widely used for evaluating HPC architectures. Sweep3D is the heart of a real Accelerated Strategic Computing Initiative (ASCI) application and exploits parallelism via a wavefront process. We ran the MPI-Fortran77 code in weak scaling mode maintaining $5 \times 5 \times 400$ cells per processor with 10 k-planes/3 angles per block.
- NQueens – A backtracking state space search problem where the goal is to determine a placement of N queens on an $N \times N$ chessboard (18-queens in our runs) so that

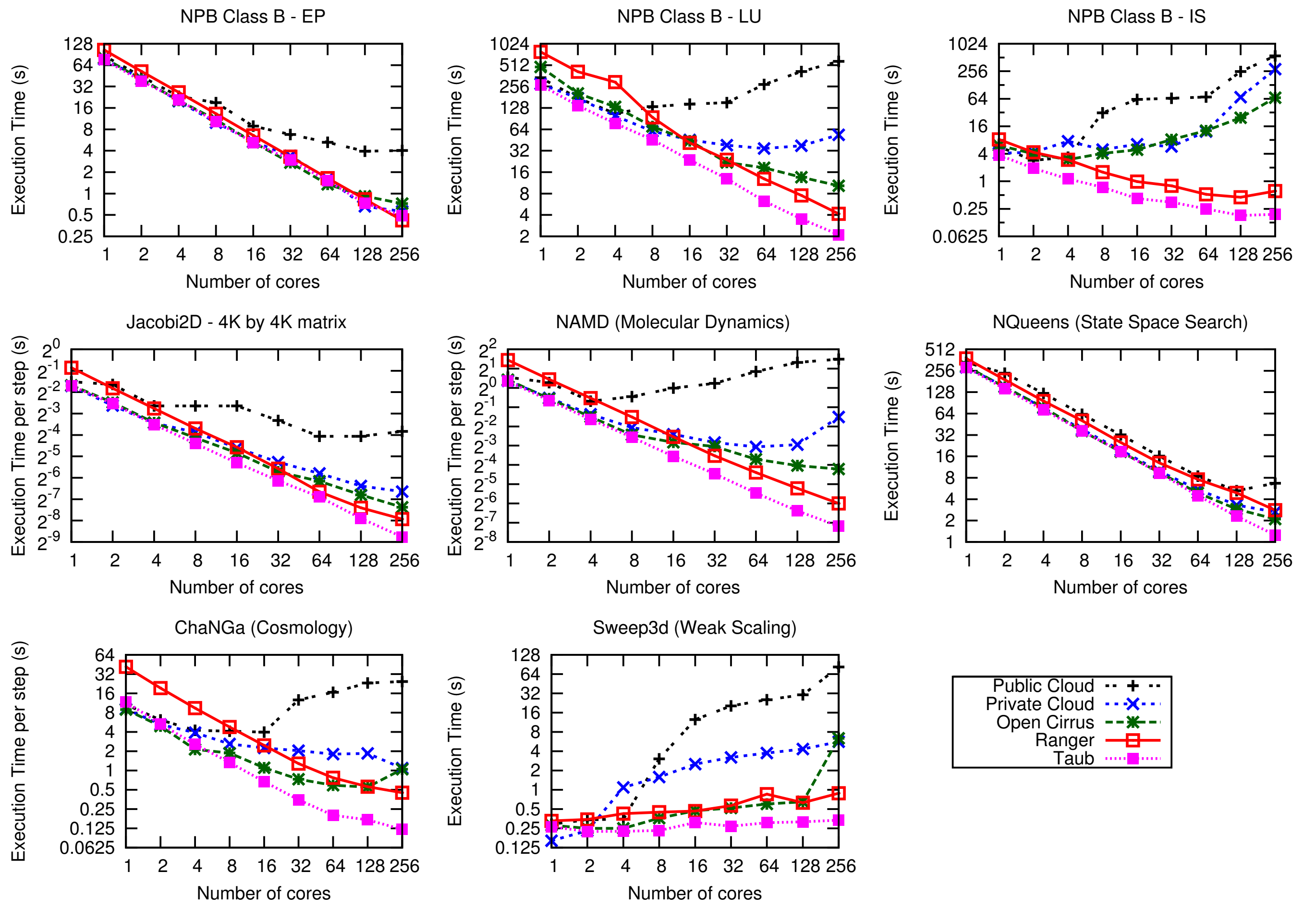


Fig. 1: Execution Time vs. Number of cores (strong scaling for all except Sweep3D) for different applications. All applications scale very well on Taub and Ranger and moderately well on Open Cirrus. On Private and Public Clouds, IS does not scale at all, LU and NAMD stop scaling after 8–32 cores whereas EP, Jacobi2D and NQueens scale well.

no two queens can attack each other. This is implemented as a tree structured computation, and communication happens only for load-balancing purposes.

On Ranger and Taub, we used available MVAPICH2 [23] for MPI and CHARM++ ibverbs layer. On rest of the platforms we installed Open MPI [24] and used net layer of CHARM++.

III. Benchmarking HPC Performance

Figure 1 shows the scaling behavior of our testbeds for the selected applications. These results are averaged across multiple runs (5 executions) performed at different times. We show strong scaling results for all applications except Sweep3D, where we chose to perform weak scaling runs. For NPB, we present results for only Embarrassingly parallel (EP), LU solver (LU), and Integer sort (IS) benchmarks due to space constraints. The first observation is the difference in sequential performance: Ranger takes almost twice as long as the other platforms, primarily because of the older and slower processors. The slope of the curve shows how the applications scale on different platforms. Despite the poor sequential speed, Ranger's performance crosses Open Cirrus, private cloud and public cloud for some applications at around 32 cores, yielding a much more linearly scalable parallel performance. We investigated the reasons for better scalability of these applications on Ranger using application profiling, performance tools, and microbenchmarking and found that network performance is a dominant factor (section IV).

We observed three different patterns for applications on these platforms. First, some applications such as, EP, Jacobi2D, and NQueens, scale well on all the platforms up to 128–256 cores. The second pattern is that some applications such as LU, NAMD, and ChaNGa scale on private cloud till 32 cores, and stop scaling afterwards. These do well on other platforms including Open Cirrus. The likely reason for this trend is the impact of virtualization on network performance (which we confirm below). On public cloud, we used VM instances with 4 virtual cores, hence inter-VM communication starts after 4 cores, resulting in sudden performance penalty above 4 cores. Finally, some applications, especially the NPB IS (Integer Sort) benchmark, fail to scale on the clouds and Open Cirrus. IS is a communication intensive benchmark and involves data reshuffling operations for sorting. Sweep3D also exhibits poor weak scaling after 4 – 8 cores on cloud.

When running experiments on cloud, we observed variability in the execution time across runs, which we quantified by calculating the coefficient of variation (standard deviation/mean) for execution time across 5 executions. Figure 2 shows that there is a significant amount of variability on cloud compared to supercomputer (Ranger) and that the amount of variability increases as we scale up, partially due to decrease in computational granularity. For the case of 256 cores at public cloud, standard deviation is equal to half the mean, implying that on average, values are spread out between $0.5 \times mean$ and $1.5 \times mean$ resulting in low predictability of performance across runs. In contrast, private cloud shows less variability.

One potential reason for the significant performance variation is the use of shared resources. We deliberately chose shared systems, shared at node level, not at the core level, for cloud. Using isolated system would be misleading and likely result in far better performance than what one can get from current cloud offerings. Noise induced by multi-tenancy is an intrinsic component of the cloud, inherent in the fundamental business model of the cloud providers.

Further analysis is required to determine what application and platform characteristics are affecting achieved performance and variability. We present our findings in the next section.

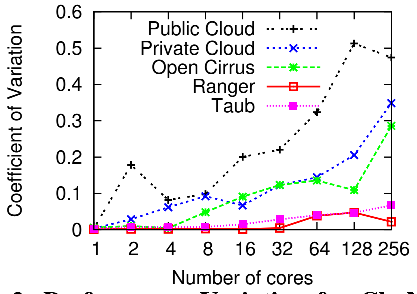


Fig. 2: Performance Variation for ChaNGa

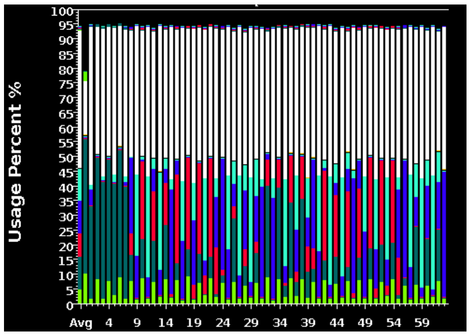


Fig. 3: CPU utilization for execution of Jacobi2D (4K by 4K) on 32 2-core VMs of private cloud. White portion: idle time, colored portions: application functions.

IV. Performance Bottlenecks for HPC in Cloud

We used the Projections [25] tool to analyze the performance bottlenecks on cloud. Figure 3 shows the CPU utilization for a 64-core Jacobi2D experiment on private cloud, x-axis being the (virtual) core number. It is clear that CPU is under-utilized for almost half the time, as shown by the idle time (white portion) in the figure. A detailed time-line view revealed that this time was spent waiting to receive data from other processes. Similarly, for NAMD, communication time is a considerable portion of the parallel execution time on cloud.

Since many HPC applications are highly sensitive to communication, we focused on network performance. Figures 4a–4b show the results of a simple ping-pong benchmark written in Converse, the underlying substrate of CHARM++ [26]. Unsurprisingly, we found that the latencies and bandwidth on cloud are a couple of orders of magnitude worse compared to Ranger and Taub, making it challenging for communication-intensive applications, such as IS, LU, and NAMD, to scale.

While the inferior network performance explains the large idle time in Figure 3, the surprising observation is the notable difference in idle time for alternating cores (0 and 1) of each VM. We traced this effect to network virtualization. The light (green) colored portion at the very bottom in the figure represents the application function `begin_iteration` which initiates inter-processor communication through socket operations (such as `select, recv, send`), and interacts with the virtual network. The application process on core 0 of the VM shares the CPU with the network emulator. This interference (*noise* or *jitter*) increases as the application communicates more over the network. Hence, virtualized network degrades HPC performance in multiple ways: increases network latency, reduces bandwidth, interferes with application process.

We also observed that, even when we used only core 0 of each VM, for iterative applications containing a barrier after each iteration, there was significant idle time on some processes at random times. Communication time could not explain such random idle times. Hence, we used the Netgauge [27] tool for measuring OS noise. We ran a benchmark that performs a fixed amount of work multiple times and records the time it takes for each run (Figure 4c). Each benchmark step is designed to take 1000 microseconds in the absence of noise, but as evident from Figure 4c, a large fraction of steps takes significantly longer time – from 20% up to 200% longer.

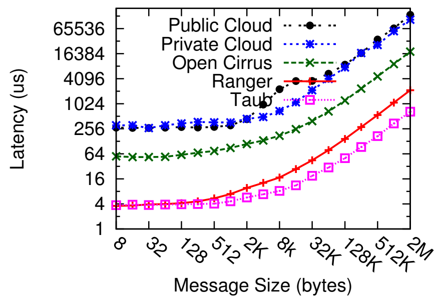
In general, system noise has detrimental impact on performance, especially for bulk-synchronous HPC applications since the slowest thread dictates the speed [28]. Unlike supercomputers, where OS is specifically tuned to minimize noise, e.g., Scientific Linux on Taub, cloud deployments typically run non-tuned operating systems. Clouds have a further intrinsic disadvantage due to the presence of the hypervisor.

V. Optimizing Cloud Virtualization for HPC

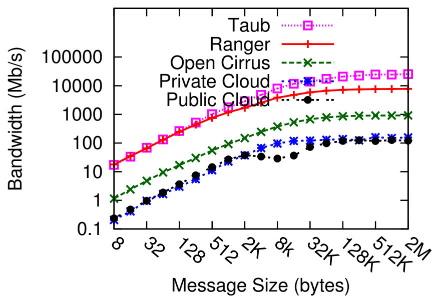
To mitigate the virtualization overhead, we investigate two optimizations: lightweight virtualization and CPU affinity.

A. *Lightweight Virtualization*

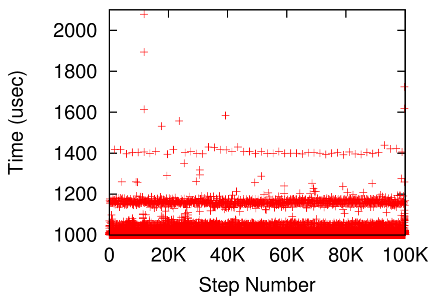
We consider two lightweight virtualization techniques, *thin VMs* configured with PCI pass-through for I/O, and *containers*, that is OS-level virtualization. Lightweight virtualization reduces the latency overhead of network virtualization by granting VMs native accesses to physical network interfaces. In the thin VM configuration with IOMMU, a physical network interface is allocated exclusively to a VM, preventing the interface to be shared by the sibling VMs and the hypervisor. This may lead to under utilization when the thin VM generates insufficient network load. Containers such as LXC [15] share



(a) Latency

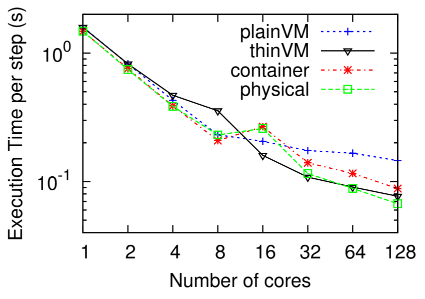


(b) Bandwidth

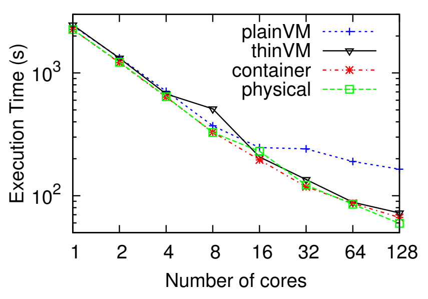


(c) Noise

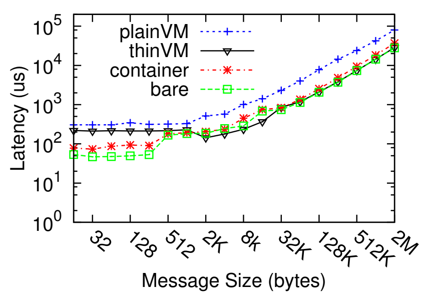
Fig. 4: (a,b) Latency and Bandwidth vs. Message Size on all platforms. Network performance on cloud is off by almost two orders of magnitude compared to Supercomputers. (c) Fixed Work Quantum Benchmark on a VM for measuring OS noise; in a noise-less system every step should take 1000 μs.



(a) NAMD

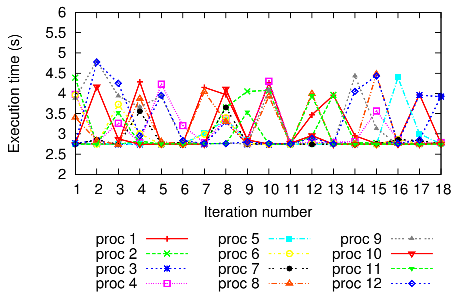


(b) ChaNGa

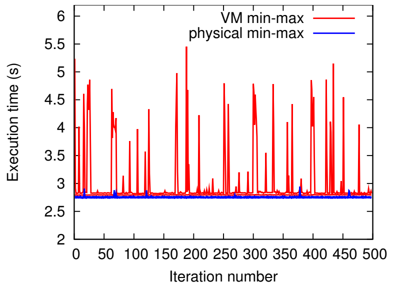


(c) Ping-Pong Test

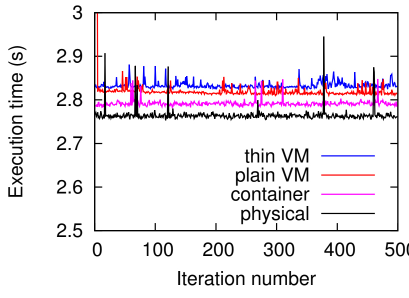
Fig. 5: Impact of Virtualization on Application Performance



(a) Physical Machine (No CPU Affinity)



(b) Thin VM (Application-level Affinity only) and Physical Machine (CPU Affinity)



(c) All Configurations (CPU Affinity)

Fig. 6: Impact of CPU Affinity on CPU Performance

the physical network interface with its sibling containers and its host. However, containers must run the same operating system as their underlying host. Thus, there is a trade-off between resource multiplexing and flexibility offered by VM.

Figure 5 validates that network virtualization is the primary bottleneck of cloud. These experiments were conducted on the virtualization testbed described earlier (Table II). On plain VM, the scalability of NAMD and ChaNGa (Figure 5a–5b) is similar to that of private cloud (Figure 1). However, on thin VM, NAMD execution times closely track that of the physical machine even as multiple nodes are used (i.e., 16 cores onwards). The performance trend of containers also resembles the one of the physical machine. This demonstrates that thin VM and containers impose a significantly lower communication overhead. This low overhead is further validated by the ping-pong test (Figure 5c).

We note that there are other HPC-optimized hypervisors [29, 30]. However, an exhaustive comparison of hypervisors is not our intention. Our goal is to focus on the current state of device virtualization and provide valuable insights to cloud operators.

B. *Impact of CPU Affinity*

CPU affinity instructs the operating system to bind a process (or thread) to a specific CPU core. This prevents the operating systems to inadvertently migrate a process. If all important processes have non-overlapping affinity, it practically prevents multiple processes or threads to share a core. In addition, cache locality can be improved by processes or threads remaining on the same core throughout their execution. However, in the cloud, CPU affinity can be enforced at the *application level*, which refers to binding processes to the virtual CPUs of a VM, and at the *hypervisor level*, which refers to binding virtual CPUs to physical CPUs.

Figure 6 presents the results of our micro-benchmarks with

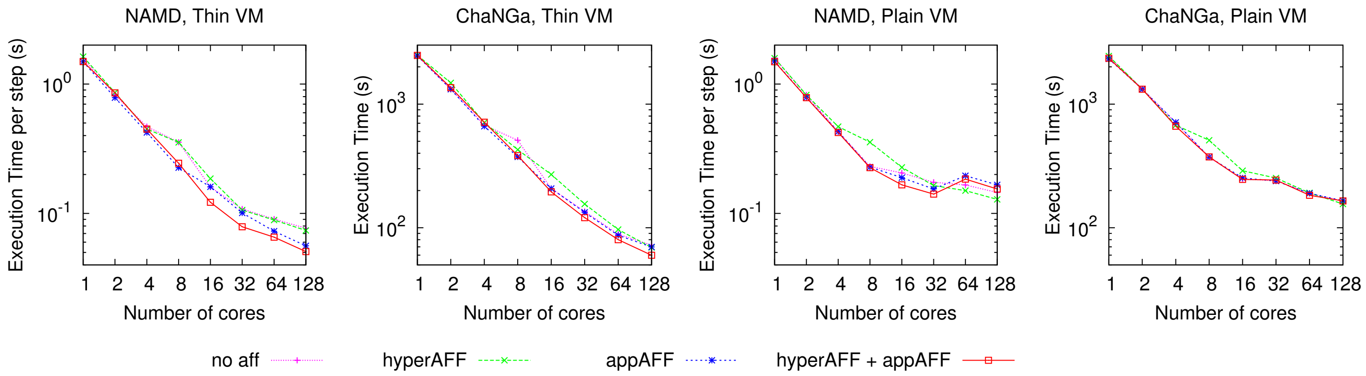


Fig. 7: Application Performance with various CPU Affinity Settings, thin VM and plain VM; legend is at the bottom

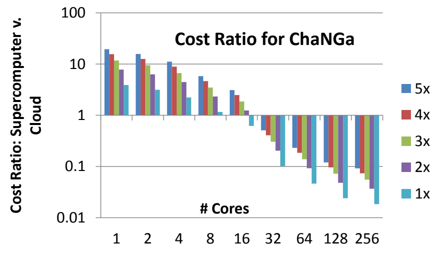
various CPU affinity settings on different types of virtual environments. In this experiment, we executed 12 processes on a single 12-core virtual or physical machine. Each process runs 500 iterations, where each iteration executes 200 millions of $y = y + rand()/c$ operations. Without CPU affinity (Figure 6a), we observe wide fluctuation on the process execution times, up to over twice of the minimum execution time (i.e., 2.7s). This clearly demonstrates that frequently two or more of our benchmark processes are scheduled to the same core. The impact of CPU affinity is even more profound on virtual machines: Figure 6b shows the minimum and maximum execution times of the 12 processes with CPU affinity enabled on the physical machine, while only application-level affinity is enabled on the thin VM. We observe that the gap between minimum and maximum execution times is narrowed, implying that load balance takes effect. However, on the thin VM, we still notice the frequent spikes, which is attributed to the absence of hypervisor-level affinity. Hence, even though each process is pinned to a specific virtual CPU core, multiple virtual cores may still be mapped onto the same physical core. When hypervisor-level affinity is enabled, execution times across virtual cores stabilizes close to those of the physical machine (Figure 6c). In conducting these experiments, we have learned several lessons. Firstly, virtualization introduces a small amount of computation overhead, where the execution times on containers, thin VM, and plain VM are higher by 1–5% (Figure 6c). We also note that it is crucial to minimize I/O operations unrelated to applications to attain the maximum application performance. Even on the physical machine, the maximum execution time is increased by 3–5% due to disk I/O generated by the launcher shell script and its `stdout/stderr` redirection (result not shown due to space limitation). The spikes on the physical machine in Figure 6c are caused by short `ssh` sessions which simulate the scenarios where users log in to check the job progress. Thus, minimizing the unrelated I/O is another important issue for HPC cloud providers to offer maximum performance to their users.

Figure 7 shows the positive impact of CPU affinity on thin VM and plain VM. HyperAFF denotes the execution where hypervisor-level affinity is enabled. Similarly appAFF means application-level affinity is enabled. Significant benefits are obtained for thin-VM, when using both application-level and hypervisor-level affinity compared to the case with no affinity. However, the impact on NAMD on plain VMs is not clear, which indicates that optimizing cloud for HPC is non-trivial. Then the question is why and when should one move to cloud?

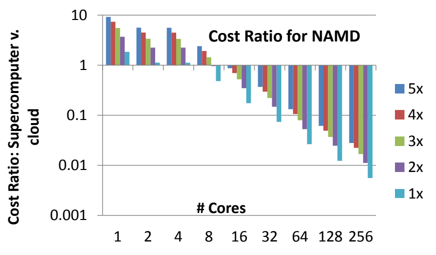
VI. HPC Economics in the Cloud

There are several reasons why many commercial and web applications are migrating to public clouds from fully owned resources or private clouds. Variable usage in time (resulting in lower utilization), trading CAPEX (capital expenditure) for OPEX (operating expenditure), and the shift towards a delivery model of Software as a Service are some of the primary motivations fueling the shift to the cloud in commercial environments. These arguments apply both to cloud providers and cloud users. Cloud users benefit from running in the cloud when their applications fit the profile we described e.g., variable utilization. Cloud providers can justify their business if the aggregated resource utilization of all their tenants can sustain a profitable pricing model when compared to the substantial infrastructure investments required to offer computing and storage resources through a cloud interface.

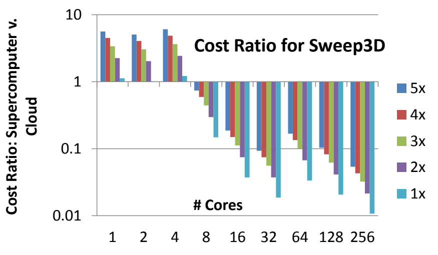
HPC applications are however quite different from the typical web and service-based applications. First, utilization of the computing resources is typically quite high on HPC systems. This conflicts with the desirable properties of high-variability and low average utilization that make the cloud business model viable. Moreover, an HPC cloud user would ideally want a dedicated instance, but for a cloud provider that means that the multi-tenancy opportunities are limited and the pricing has to be increased to be able to profitably rent a dedicated computing resource to a single tenant. As evident from our analysis, the noise caused by virtualization and multi-tenancy can significantly affect HPC applications in terms of performance predictability and scalability. Virtualization is a foundational technology for the cloud to enable improved consolidation and easier management (moving VMs around for performance, support, and reliability), but it needs to be carefully tuned for HPC applications. Secondly, the performance of many HPC applications is very sensitive to the interconnect, as we showed in our experimental evaluation. In particular, low latency requirements are typical for the HPC applications that incur substantial communication. This is in contrast with the commodity Ethernet network (1Gbps today moving to 10Gbps) typically deployed in most cloud



(a) ChaNGa



(b) NAMD



(c) Sweep3D

Fig. 8: Cost ratio of running in cloud and a dedicated supercomputer for different scale (cores) and cost ratios (1x–5x). Ratio>1 imply savings of running in the cloud, <1 favor supercomputer execution.

infrastructures to keep costs small. These limitations constrain the number of HPC application that are a good fit for the cloud: when networking performance is important, we quickly reach diminishing returns of scaling-out a cloud deployment to meet a certain performance target. Depending on the pricing model, if too many VMs are required to meet performance because of lack of scalability, the cloud deployment quickly becomes uneconomical. Finally, the CAPEX/OPEX argument is less clear-cut for HPC users. Publicly funded supercomputing centers typically have CAPEX in the form of grants, and OPEX budgets may actually be tighter and almost fully consumed by the support and administration of the supercomputer with little headroom for cloud bursting. Software-as-a-Service offering are also rare in HPC to date, although that might change in the future. Data movement to and from cloud and the security aspects are other challenges for HPC in cloud. However, they are out of scope of this paper.

So, what are the conditions that can make HPC in the cloud a viable business model for both HPC users and cloud providers? Unlike large supercomputing centers, HPC users in small-medium enterprises are much more sensitive to the CAPEX/OPEX argument. For example, startups with HPC requirements (e.g., simulation or modeling) in general have little choice but to go to the cloud for their resources and buying a supercomputer is not an option. Similarly, small-medium enterprises with growing business and an existing HPC infrastructure may be reluctant to grow on-premise resources in volatile markets and would rather take a pay-as-you-go approach. The ability to take advantage of a large variety of different architectures (with different interconnects, processor types, memory sizes, etc.) can result in better utilization at global scale, compared to the limited choices available in any individual organization. Running HPC applications on the most economical architecture while meeting the performance expectations can result in overall savings for consumers.

To illustrate a few possible HPC-in-the-cloud scenarios, we collected and compared cost and price data of supercomputer installation and typical cloud offering. Based on our survey of cloud prices, known financial situations of cloud operators, published supercomputing costs, and a variety of internal and external data sources [31], we estimate that a cost ratio between 2x and 3x is a reasonable approximate range capturing the differences between a cloud deployment and on-premise supercomputing resources today. In our terminology, 2x indicates the case where 1 supercomputer core-hour is twice as expensive as 1 cloud core-hour. Since these values will continue to fluctuate, possibly in unforeseen ways, we expand the range between 1x and 5x to capture different future scenarios.

Using the performance evaluations for different applications that we presented in Figure 1, we calculated the cost differences of running the application in the public cloud vs. running it in a dedicated supercomputer (Ranger), assuming different per-core-hour cost ratios from 1x to 5x. Figure 8a-c show the cost differences for three applications, where values>1 indicate savings of running in the cloud and values<1 an advantage of running it on a dedicated supercomputer. We can see that for each application there is a scale in terms of the number of cores up to which it is more cost-effective to execute in the cloud vs. on a supercomputer. For example, for Sweep3D, NAMD, and ChaNGa, this scale is higher than 4, 8, and 16 cores respectively. Smaller-scale executions in the cloud are advantageous, but after a certain scale it becomes counterproductive. The break-even point is a function of the application scalability and the cost ratio. However our observation is that there is little sensitivity to the cost ratio and it is relatively straightforward to determine the breakpoint. This is true even for the cost ratio of 1. This might be the artifact of slower processors for the Ranger vs. newer and faster processors in the cloud.

VII. Discussion: Cloud Bursting and Benefits

In the previous sections, we provided empirical evidence that applications behave quite differently on different platforms, and interesting cross-over points appear when taking cost into the equation. This observation opens up several opportunities to optimize the mapping between applications and platforms, and pass the benefits to both cloud providers and end users. In this section, we discuss the case when the dedicated infrastructure cannot meet peak demands and the user is considering “cloud bursting” as a way to offload the peaks to the cloud. In this case, the knowledge of application and platform characteristics and their impact on performance can help answer (1) which applications from a set to burst to cloud, and (2) which cloud to burst to.

Consider (1), a simple allocation scheme may not even find a feasible solution, regardless of the cost. For example, first-come-first-served may exhaust the dedicated resources on

TABLE III: Findings

Question	Answers
Who	Small and medium scale organizations (pay-as-you-go benefits), with applications which result in best performance/cost ratio in cloud vs. other platforms.
What	Applications with less-intensive communication patterns, less sensitivity to noise/interference, small to medium scale
Why	HPC users in small-medium enterprises much more sensitive to the CAPEX/OPEX argument. Ability to exploit a large variety of different architectures (Better utilization at global scale, potential savings for consumers)
How	Technical approaches: Lightweight virtualization, CPU affinity Business models: cloud bursting, hybrid supercomputer–cloud approach with application-aware mapping

cloud-friendly applications, and attempt bursting to the cloud, applications that do not scale and have no chance of meeting the performance target.

Knowledge of application characteristics can also help to answer (2), that is which cloud to select from the several commercially available options, each having different characteristics and pricing rates. For example, for some applications demonstrating good scalability within a given range, it would be cost effective to run on a low-cost ($ per core-hour) cloud. For other communication-intensive applications a higher-cost HPC-optimized cloud would be more effective.

Hence, we propose the co-existence of supercomputer and cloud with a two step methodology – 1) characterize applications using theoretical models, instrumentation, or simulation and 2) intelligently match applications to platforms based on user preferences. In [32,33], we developed a proof-of-concept mapper tool, and demonstrated that such intelligent mapping can reduce cost by 60% while limiting the performance penalty to 10-15% vs. a non optimized configuration.

VIII. Related Work

Walker [2], followed by several others [3,6,7], conducted the study on HPC in cloud by benchmarking Amazon EC2 [34]. The work by He et al. [35] extended the research to three public clouds and real applications and compared the results with dedicated HPC systems. Ekanayake et al. [36] compared applications with different communication and computation complexities and observed that latency-sensitive applications experience higher performance degradation than bandwidth-sensitive applications.

We address these issues by exploring techniques in open-source virtualization, and quantify how close we can get to *physical machine* performance for HPC workloads. There are other recent efforts on HPC-optimized hypervisors [29,30,37].

Perhaps the most comprehensive evaluation of HPC in cloud to date was performed under the US Department of Energy's (DoE) Magellan project [1, 4]. Jackson et al. [4] compared conventional HPC platforms to Amazon EC2 and used real applications representative of the workload at a typical DoE supercomputing center. They concluded that the interconnect and I/O performance on commercial cloud severely limits performance and causes significant variability in performance across different executions. A key take-away of the Magellan project is that it is more cost-effective to run DOE applications on in-house supercomputers rather than on current public cloud offerings. However, their analysis is based on heavy usage (like DoE) which justifies building a dedicated supercomputer, and benefits from economy of scale. Our work looks at similar questions from the perspective of smaller scale HPC users, such as small companies and research groups who have limited access to supercomputer resources and varying demand over time. We also consider the perspective of cloud providers who want to expand their offerings to cover the aggregate of these smaller scale HPC users, for whom an attractive option is to look at a combination of own infrastructure and commercial clouds based on pricing and demand.

Kim et al. [38] present three usage models for hybrid HPC grid and cloud computing: acceleration, conservation, and resilience. However, they use cloud for sequential simulation and do not consider execution of parallel applications on cloud. Napper and Bientinesi [8], Gupta and Milojicic [9,32], Roloff et al. [39], and Marathe et al. [40] performed HPC-cloud cost evaluation. Their conclusions are that different clouds have different performance-cost tradeoffs for different applications.

Our approach in this paper is somewhat different: We take a holistic viewpoint and consider all the different aspects of running in cloud - performance, cost, and business models.

IX. Conclusions, Lessons and Future Work

Through a performance and economic analysis of HPC applications and a comparison on a range of platforms, we have shown that different applications exhibit different characteristics that make them more or less suitable to run in a cloud environment. Table III presents our conclusions.

Although some of the findings are similar to the behavior of early Beowulf clusters, those clusters are quite different from today's clouds: processor, memory, and networking technologies have tremendously progressed. The appearance of virtualization introduces multi-tenancy, resource sharing and several other new effects. We believe our research will help better understand which applications are cloud candidates, and where we should focus our efforts to improve the performance. Next, we summarize the lessons learned from this research and the emerging future research directions.

A hybrid cloud-supercomputer platform environment can outperform its individual constituents. By using an underutilized resource which is "good enough" to get the job done sooner and more cheaply, it is possible to get better performance for the same cost on one platform for some applications, and on another platform for another application. More work is needed to better quantify the "good enough" dimension, as well as the deep ramification of cloud business models on HPC.

Lightweight virtualization is necessary to remove overheads for HPC in cloud. With low-overhead virtualization, the same hardware of a web-oriented cloud infrastructure can be reused for HPC. We envisage hybrid clouds that can support both HPC and commercial workloads. Such a hybrid cloud stack would require proper tuning or VM re-provisioning for HPC applications, which is a fertile topic for future research.

Application characterization for analysis of the performance-cost tradeoffs for complex HPC applications is a non-trivial task, but the economic benefits are substantial. More research is necessary to be able to quickly identify important traits for complex applications such as those with dynamic and irregular communication patterns. A related future direction is to evaluate applications with irregular access patterns and dynamic datasets, such as those arising from 4-D CT imaging, 3-D moving meshes, and Computational Fluid Dynamics (CFD) applications.

Acknowledgments

The authors gratefully acknowledge Lavanya Ramakrishnan of US DoE's Magellan project and Robert Schreiber of HP labs for reviewing this paper and providing valuable feedback.

References

[1] "Magellan Final Report," U.S. Department of Energy (DOE), Tech. Rep., 2011.

[2] E. Walker, "Benchmarking Amazon EC2 for high-performance scientific computing," *LOGIN*, pp. 18–23, 2008.

[3] P. Mehrotra, J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Lazanoff, S. Saini, and R. Biswas, "Performance Evaluation of Amazon EC2 for NASA HPC applications," in *Proceedings of the 3rd workshop on Scientific Cloud Computing*, ser. ScienceCloud '12. New York, NY, USA: ACM, 2012, pp. 41–50.

[4] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright, "Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud," in *CloudCom'10*, 2010.

[5] "High Performance Computing (HPC) on AWS," http://aws.amazon.com/hpc-applications.

[6] C. Evangelinos and C. N. Hill, "Cloud Computing for parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2." Cloud Computing and Its Applications, Oct. 2008.

[7] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, pp. 931–945, June 2011.

[8] J. Napper and P. Bientinesi, "Can Cloud Computing reach the Top500?" ser. UCHPC-MAW '09. ACM, 2009.

[9] A. Gupta and D. Milojicic, "Evaluation of HPC Applications on Cloud," in *Open Cirrus Summit (Best Student Paper)*, Atlanta, GA, Oct. 2011, pp. 22 –26. [Online]. Available: http://dx.doi.org/10.1109/OCS.2011.10

[10] "Ranger User Guide," http://services.tacc.utexas.edu/index.php/ranger-user-guide.

[11] A. I. Avetisyan et al., "Open Cirrus: A Global Cloud Computing Testbed," *Computer*, vol. 43, pp. 35–43, April 2010.

[12] D. Nurmi et al., "The Eucalyptus Open-source Cloud-computing System," in *Proceedings of Cloud Computing and Its Applications*, Oct. 2008.

[13] "KVM – Kernel-based Virtual Machine," Redhat, Inc., Tech. Rep., 2009.

[14] 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," *Cloud Computing, IEEE International Conference on*, vol. 0, pp. 9–16, 2011.

[15] D. Schauer et al., "Linux containers version 0.7.0," June 2010, http://lxc.sourceforge.net/.

[16] "Intel(r) Virtualization Technology for Directed I/O," Intel Corporation, Tech. Rep., Feb 2011, http://download.intel.com/technology/computing/vptech/Intel(r)_VT_for_Direct_IO.pdf.

[17] "MPI: A Message Passing Interface Standard," in *M. P. I. Forum*, 1994.

[18] L. Kale and S. Krishnan, "Charm++: A portable concurrent object oriented system based on C++," in *Proceedings of the Conference on Object Oriented Programming Systems, Languages and Applications*, September 1993.

[19] "NPB," http://www.nas.nasa.gov/Resources/Software/npb.html.

[20] A. Bhatele et al., "Overcoming Scaling Challenges in Biomolecular Simulations across Multiple Platforms," in *IPDPS 2008.*

[21] P. Jetley, F. Gioachin, C. Mendes, L. V. Kale, and T. R. Quinn, "Massively Parallel Cosmological Simulations with ChaNGa," in *IPDPS 2008*, 2008, pp. 1–12.

[22] "The ASCII Sweep3D code," http://wwwc3.lanl.gov/pal/software/sweep3d.

[23] M. Koop, T. Jones, and D. Panda, "MVAPICH-Aptus: Scalable high-performance multi-transport MPI over InfiniBand," in *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, april 2008, pp. 1 –12.

[24] E. Gabriel et al., "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation," in *Proc. of 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, 2004.

[25] L. Kalé and A. Sinha, "Projections : A Scalable Performance Tool," in *Parallel Systems Fair, International Parallel Processing Sympos ium*, Apr. 1993, pp. 108–114.

[26] *The CONVERSE programming language manual*, Department of Computer Science,University of Illinois at Urbana-Champaign, Urbana, IL, 2006.

[27] T. Hoefler, T. Mehlan, A. Lumsdaine, and W. Rehm, "Netgauge: A Network Performance Measurement Framework," in *Proceedings of High Performance Computing and Communications, HPCC'07.*

[28] T. Hoefler, T. Schneider, and A. Lumsdaine, "Characterizing the Influence of System Noise on Large-Scale Applications by Simulation," in *Supercomputing 10*, Nov. 2010.

[29] J. Lange, K. Pedretti, T. Hudson, P. Dinda, Z. Cui, L. Xia, P. Bridges, A. Gocke, S. Jaconette, M. Levenhagen, and R. Brightwell, "Palacios and Kitten: New High Performance Operating Systems for Scalable Virtualized and Native Supercomputing," ser. IPDPS '10, pp. 1–12.

[30] B. Kocoloski, J. Ouyang, and J. Lange, "A case for dual stack virtualization: consolidating HPC and commodity applications in the cloud," ser. SoCC '12, New York, NY, USA, 2012, pp. 23:1–23:7.

[31] C. Bischof, D. anMey, and C. Iwainsky, "Brainware for Green HPC," *Computer Science - Research and Development*, pp. 1–7, 2011. [Online]. Available: http://dx.doi.org/10.1007/s00450-011-0198-5

[32] A. Gupta et al., "Exploring the Performance and Mapping of HPC Applications to Platforms in the cloud," in *HPDC '12*. New York, NY, USA: ACM, 2012, pp. 121–122.

[33] A. Gupta et al, "The Who, What, Why and How of High Performance Computing Applications in the Cloud," HP Labs, Tech. Rep., 2013. [Online]. Available: http://www.hpl.hp.com/techreports/2013/HPL-2013-49.html

[34] "Amazon Elastic Compute Cloud (Amazon EC2)," http://aws.amazon.com/ec2.

[35] Q. He, S. Zhou, B. Kobler, D. Duffy, and T. McGlynn, "Case Study for Running HPC Applications in Public Clouds," ser. HPDC '10. ACM, 2010.

[36] J. Ekanayake, X. Qiu, T. Gunarathne, S. Beason, and G. C. Fox, *High Performance Parallel Computing with Clouds and Cloud Technologies*, 07/2010 2010.

[37] A. Kudryavtsev, V. Koshelev, B. Pavlovic, and A. Avetisyan, "Virtualizing HPC Applications using Modern Hypervisors," in *Proceedings of the 2012 workshop on Cloud services, federation, and the 8th open cirrus summit*, ser. FederatedClouds '12. New York, NY, USA: ACM, 2012, pp. 7–12. [Online]. Available: http://doi.acm.org/10.1145/2378975.2378978

[38] H. Kim, Y. el Khamra, I. Rodero, S. Jha, and M. Parashar, "Autonomic Management of Application Workflows on Hybrid Computing Infrastructure," *Scientific Programming*, vol. 19, no. 2, pp. 75–89, Jan. 2011.

[39] E. Roloff, M. Diener, A. Carissimi, and P. Navaux, "High Performance Computing in the Cloud: Deployment, Performance and Cost Efficiency," in *CloudCom 2012*, 2012, pp. 371–378.

[40] A. Marathe, R. Harris, D. K. Lowenthal, B. R. de Supinski, B. Rountree, M. Schulz, and X. Yuan, "A Comparative Study of High-performance Computing on the Cloud," ser. HPDC '13. New York, NY, USA: ACM, 2013, pp. 239–250.