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

Abhishek Gupta, Laxmikant V. Kale, Filippo Gioachin, Verdi March, Chun Hui Suen, Bu-Sung Lee, Paolo Faraboschi, Richard Kaufmann, Dejan Milojicic

HP Laboratories
HPL-2013-49

Keyword(s):

High Performance Computing; Cloud; Applications; Supercomputers; Performance Evaluation; Economics

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 (or not choose) cloud for HPC, for what applications, and how should cloud be used for HPC? To this end, we perform a comprehensive performance evaluation of a set of benchmarks and complex HPC applications on a range of platforms, varying from supercomputers to commodity clusters, both in-house and in the cloud. Having identified the performance bottlenecks in cloud, we demonstrate that alternative lightweight virtualization mechanisms such as thin VMs and OS-level containers, and hypervisor and applicationlevel CPU affinity can greatly lower the overhead and noise of virtualization. Next, we discuss the economic aspects of HPC in the cloud, which we believe is an important area that has not been sufficiently addressed by past research. Finally, we explore the benefits of an intelligent matching of an HPC application to the best deployment platform within a given set, discussing how we can combine a cloud deployment with a supercomputer. Overall results indicate that current public clouds are cost-effective only at small scale ? 4 to 16 cores for the chosen HPC applications, when considered in isolation, but can complement (and co-exist with) supercomputers using models such as cloud burst and application-aware mapping to achieve significant cost benefits.

External Posting Date: July 21, 2013 [Fulltext] Approved for External Publication
Internal Posting Date: July 21, 2013 [Fulltext]

The Who, What, Why and How of High Performance Computing Applications 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
(gioachin, verdi.march,
chun-hui.suen, francis.lee)@hp.com

Paolo Faraboschi,
Richard Kaufmann,
Dejan Milojicic
HP Labs
Palo Alto, CA, USA
(paolo.faraboschi, richard.kaufmann,
dejan.milojicic)@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 (or not choose) cloud for HPC, for *what* applications, and *how* should cloud be used for HPC? To this end, we perform a comprehensive performance evaluation of a set of benchmarks and complex HPC applications on a range of platforms, varying from supercomputers to commodity clusters, both in-house and in the cloud. Having identified the performance bottlenecks in cloud, we demonstrate that alternative lightweight virtualization mechanisms such as thin VMs and OS-level containers, and hypervisor and application-level CPU affinity can greatly lower the overhead and noise of virtualization. Next, we discuss the economic aspects of HPC in the cloud, which we believe is an important area that has not been sufficiently addressed by past research. Finally, we explore the benefits of an intelligent matching of an HPC application to the best deployment platform within a given set, discussing how we can combine a cloud deployment with a supercomputer. Overall results indicate that current public clouds are cost-effective only at small scale – 4 to 16 cores for the chosen HPC applications, when considered in isolation, but can complement (and co-exist with) supercomputers using models such as *cloud burst* and *application-aware mapping* to achieve significant cost benefits.

***Keywords*-High Performance Computing, Cloud, Applications, Supercomputers, Performance Evaluation, Economics**

I. Introduction

Setting up a dedicated infrastructure for HPC is a complex endeavor that requires a long lead time, high capital expenditure, and large operational costs. Increasingly, academic and commercial HPC users are looking at cloud computing as a cost effective alternative with the potential of reducing some of these heavy upfront financial commitments, while yielding to faster turnaround times [1]. By moving HPC applications to the cloud, additional advantages come in form of *elasticity*, which reduces the risks caused by under-provisioning, and reduces the underutilization of resources caused by over-provisioning. Also, the built-in virtualization support in the cloud offers an alternative way to support flexibility, customization, security, migration and resource control to the HPC community.

Despite these advantages, it still remains unclear whether, and when, HPC in the cloud can become a feasible substitute or complement to supercomputers. Traditionally, clouds have been designed for efficient execution of business and web applications which have very different demands than HPC 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 [5] and Amazon's EC2 Cluster Compute [6], point to a promising direction to overcome some of the fundamental inhibitors.

HPC clouds rapidly expand the application user base and the available platform choices to run HPC workloads: from in-house dedicated supercomputers, to commodity clusters with and without HPC-optimized interconnects and operating systems, to resources with different degrees of virtualization (full, CPU-only, none), to hybrid configurations that offload part of the work to the cloud. HPC users and cloud providers are faced with the challenge of choosing the optimal platform based upon a limited knowledge of application characteristics, platform capabilities, and the target metrics such as, QoS, cost.

Building upon previous research [1–5, 7–10], we start with the hypothesis that the cost/performance-optimal execution platform varies from supercomputer to cloud depending upon application characteristics, such as sensitivity to network performance and parallel scaling. To validate our hypothesis, we present a comparison of various platforms (see Table I) commonly available to HPC users, using microbenchmarks and real-world applications.

Unlike previous works on benchmarking clouds for science, we take a more holistic and practical viewpoint and go beyond performance comparison to explore techniques for reducing the performance gap between native and virtualized environment. Rather than limiting ourselves to the problem –

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

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? To answer this question, we leverage the results of the performance benchmarking, investigation of performance bottlenecks on cloud, the co-relation between application characteristics and observed performance, and the analyses of economic aspects of HPC in cloud. Also, instead of considering cloud as a substitute of supercomputer, we investigate the co-existence of supercomputer and cloud. We provide a proof-of-concept of this approach and the associated benefits through an intelligent tool based on application characterization and smart mapping.

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 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. In our terminology, the term *cloud provider* refers to public cloud providers that offer various flavors of platforms with a pay-per-use business model.

The contributions of this paper are summarized as follows.

- We analyze the performance of HPC applications on a range of platforms varying from supercomputer to cloud, study performance bottlenecks and identify *what* applications are suitable for cloud. (§ IV,§ V)
- 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 performance overhead and noise, addressing – *how* to use cloud for HPC. (§ VI)
- 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. (§ VII)
- We demonstrate that rather than running all the applications on a single platform (in-house or cloud), a more cost-effective approach is to leverage multiple platforms (dedicated and in the cloud) and use a smart application-aware mapping of applications to platforms, addressing – *how* to use cloud for HPC.(§ VIII)

The remainder of the paper is organized as follows. Section II discusses related work and Section III explains our evaluation methodology. Section IV presents evaluation results, Section V analyzes the bottlenecks in cloud and Section VI highlights techniques for optimizing clouds for HPC. Section VII introduces a few usage scenarios with a focus on cost and business aspects and Section VIII recommends how to find a good mapping of applications to platforms for a "cloud bursting" scenario. Finally, section IX covers insights and lessons learned, and leads to conclusions and future work directions in Section X.

II. Related Work

Walker [2], followed by several others [3,7,8], conducted the study on HPC in cloud by benchmarking Amazon EC2 [11]. The work by He et al. [12] extended the research to three public clouds and real applications and compared the results with dedicated HPC systems. Ekanayake et al. [13] 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 existing techniques in open-source virtualization, such as OS-level containers and thin virtualization, and we quantify how close we can get to *physical machine* performance for communication and I/O intensive HPC workloads.

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, 5]. 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. [14] discuss autonomic management of scientific workflow applications on hybrid infrastructures such as those comprising HPC grids and on-demand pay per use clouds. They 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 [9] also performed some cost evaluation and concluded that cloud cannot compete with supercomputers based on the metric $/GFLOPS, since memory and network performance is insufficient to compete with existing scalable HPC systems. In our earlier work [10], we also studied the performance-cost tradeoffs of running different applications on supercomputer vs. cloud.

Our approach in this paper is somewhat different: our hypothesis is that the optimal platform depends upon application characteristics and performance requirements. We consider benchmarks and applications which differ in computation and communication characteristics, and investigate what applications better map to which platform.

III. 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 for the purpose of this study. We believe that these platforms cover the major 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, "2QEMU VirtualCPU@2.67GHz" means each VM has 2 virtual cores. Ranger [15] at TACC is a supercomputer and Taub at UIUC is an HPC-optimized cluster. Both use Infiniband [16] as interconnect. Moreover, Taub uses scientific Linux as operating system and has QDR Infiniband with bandwidth of 40 Gbps. We used physical nodes with commodity interconnect at Open Cirrus testbed at HP Labs site [17]. The final two platforms are clouds – a private cloud which was setup using Eucalyptus [18], and a public cloud. Both these clouds use KVM [19] for virtualization. We chose KVM since it is considered by past research as one of the best candidates for HPC virtualization [20].

To get maximum performance from virtual machines, we avoided any sharing of physical cores between virtual cores. 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. Our cloud experiments involve nodes which shared physical cores with other applications from multiple users, since default VM allocation policy was round robin and only half of total capacity was available to us, hence addressing multi-tenancy.

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

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 in this cluster has two CPU sockets, each populated with a six-core CPU. Thus, each HPLS server has 12 physical cores. The experiment on the HPLS cluster involved benchmarking on four configuration: physical machines (bare), LXC containers [21], virtual machines configured with the default emulated network (plain VM), and virtual machines 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 virtual machines to directly the Ethernet hardware, thus improving the network I/O performance [22].

B. Benchmarks and Applications

The choice of suitable benchmarks and applications is critical to analyze the validity of our hypothesis. To gain insights into the performance of selected platform over a range of applications, we chose benchmarks and applications from different scientific domains and which differ in nature, amount, and pattern of inter-processor communication. Moreover, we selected benchmarks written in two different parallel programming environments - MPI [23] and CHARM++ [24]. Similarly to previous work [2, 3, 7, 10], we used NAS Parallel Benchmarks (NPB) class B [25] (the MPI version, NPB3.3-MPI), which exhibit a good variety of computation and communication requirements, and are widely used.

In addition, we chose some additional benchmarks and real world applications:

- Jacobi2D – A kernel which performs 5-point stencil computation to average values in a 2-D grid. Such stencil computation is very commonly used in scientific simulations, numerical algebra, and image processing.
- NAMD [26] – 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 [27] (Charm N-body GrAvity solver) – A highly scalable application used to perform collisionless N-body simulation. It can be used to perform cosmological simulation and also includes hydrodynamics. It uses Barnes-Hut tree to calculate forces between particles. We used a 300, 000 particle system for our runs.
- Sweep3D [28] – A particle in transport code which is widely used for evaluating high performance parallel

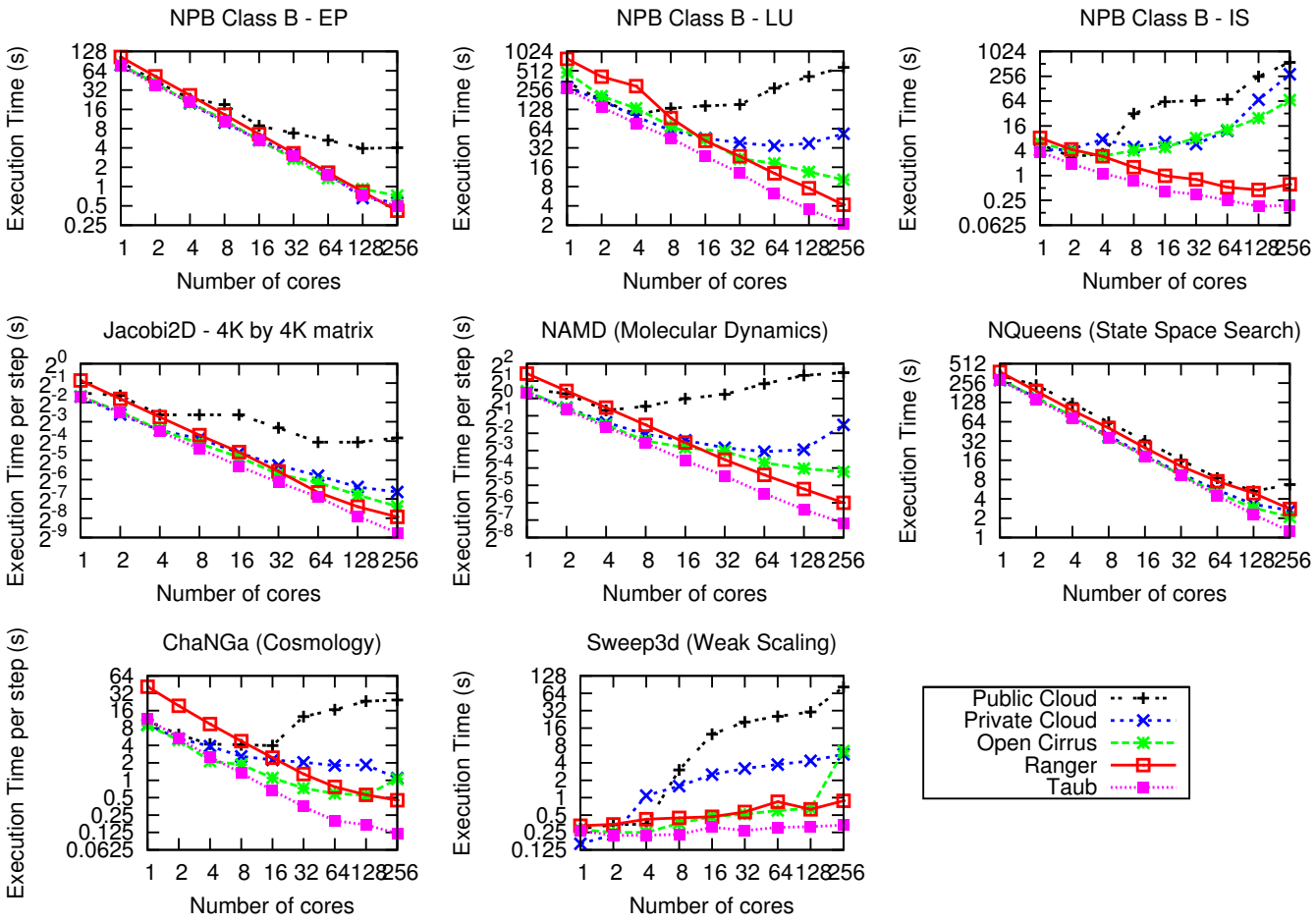


Fig. 1: Execution Time vs. Number of cores (strong scaling for all except Sweep3D) for different applications. All applications scale very well on Taub and Ranger and moderately well on Open Cirrus. On Private Cloud, IS does not scale at all, LU and NAMD stop scaling after 32 cores whereas EP, Jacobi2D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

architectures [29]. Sweep3D is the heart of a real Accelerated Strategic Computing Initiative (ASCI) application and exploits parallelism via a wavefront process. We used the code written in Fortran77 and MPI. We ran Sweep3D 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 by N chessboard (18-queens in our runs) so that no two queens can attack each other. This is implemented as a tree structured computation, and communication happens only for load-balancing purposes.

We used the available installation of MVAPICH2 [30] for MPI and ibverbs implementation of CHARM++ on Ranger and Taub. On rest of the platforms we installed Open MPI [31] and used net layer build of CHARM++.

IV. 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 the platforms and found that network performance is a dominant factor (section V). While all applications scale well on Ranger and Taub, 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 and permutation operations for sorting. Sweep3D

also exhibits poor weak scaling after 4–8 cores on cloud. Other communication intensive applications such as LU, NAMD and ChaNGa also stop scaling on private cloud around 32 cores, but do reasonably well on Open Cirrus, because of the impact of virtualization on network performance (which we confirm below). The remaining applications, EP, Jacobi2D, and NQueens, scale well on all the platforms up to 256 cores except the public cloud where most applications suffer a hit above 4 cores. On public cloud, we used VM instances with 4 virtual cores, hence there is no inter-VM communication up to 4 cores, resulting in good parallel scaling and a sudden performance penalty as the communication starts happening across VMs.

When running experiments on cloud, we observed variability in the execution time across runs. To quantify the amount of variability on cloud and compare it with a supercomputer, we calculated the coefficient of variation (standard deviation/mean) for execution time of ChaNGa 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 emphasize that we deliberately chose shared systems, shared at node level or higher granularity, not at the core level, for cloud. Results from 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. One can of course get rid of that through dedicated instances, but that will break the fundamental business model of the cloud providers, and will reflect on the cost to the point that it quickly becomes uneconomical.

Clearly, more analysis is required to determine what application and platform characteristics are degrading performance on cloud and introducing variability. In the following subsection we present our findings.

V. Performance Bottlenecks for HPC in Cloud

We used the Projections [32] 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 HPC applications are highly sensitive to communication latency, we focused on network performance. Figures 4a–4b shows the results obtained by a simple ping-pong

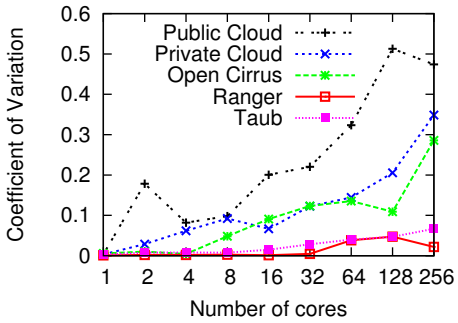


Fig. 2: Performance Variation for ChaNGa

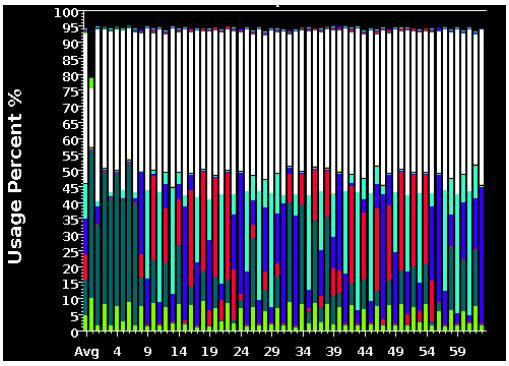
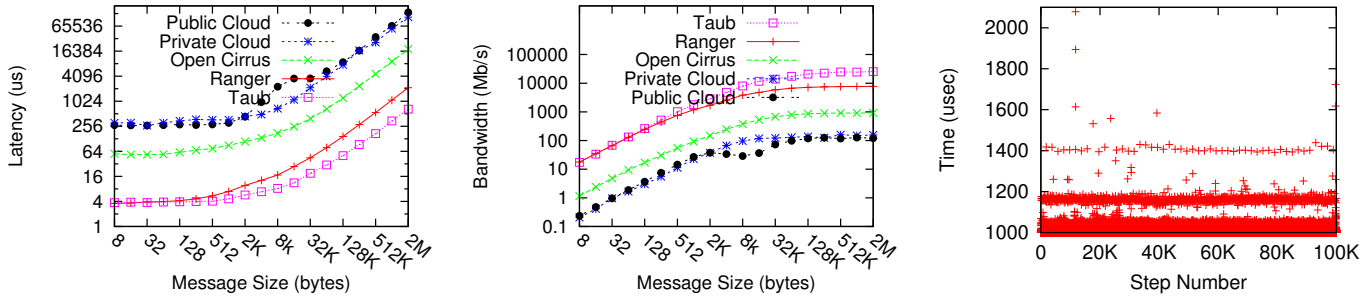


Fig. 3: CPU utilization for execution of Jacobi2D (4K by 4K) on 32 2-core VMs of private cloud: white portion shows idle time while colored portions represent application functions.

benchmark written in Converse, the underlying substrate of CHARM++ [33]. 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, NAMD and ChaNGa, to scale.

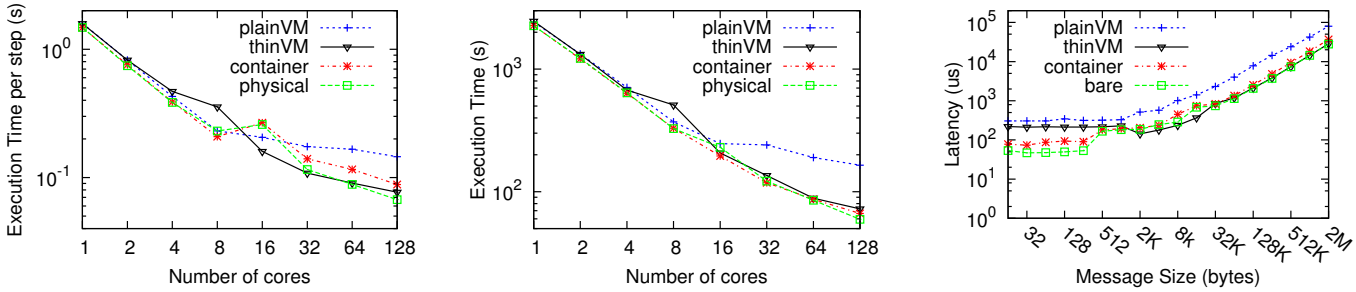
While the inferior network performance explains the large percentage of idle time in Figure 3, the surprising observation is the notable difference in idle time for alternating cores (0 and 1) of each VM. We traced this effect to network virtualization. The light (green) colored portion at the very bottom in the figure represents the application function `begin_iteration` which initiates inter-processor communication and 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 virtual network driver emulator, and this interference (sometimes called *noise* or *jitter*) increases as the application communicates more over the network. Hence, network virtualization has multiple negative effects on HPC application performance: it increases network latency, it decreases bandwidth, and it decreases application performance by interfering with the application processes.

We also observed that, even when we only used core 0 of each VM, for iterative applications containing a barrier after each iteration, there was a lot of idle time on some processes. Our first hypothesis was communication time, but that alone could not explain the significant amount of idle



(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

time. Hence, we used the Netgauge [34] tool for measuring operating system noise. We ran a benchmark (Figure 4c) that performs a fixed amount of work multiple times and records the time it takes for each run. Each benchmark step is designed to take 1000 microseconds in absence of noise, but as we can see from the figure, interference from the OS and the hypervisor results in a large fraction of steps taking significantly longer time – from 20% up to 200% longer.

In general, system noise has detrimental impact on performance, especially for HPC applications that frequently synchronize using barriers since the slowest thread dictates the speed [35]. Unlike supercomputers, which come with an OS specifically designed to minimize noise, e.g., Scientific Linux on Taub, cloud deployments typically run non-tuned operating systems, and have a further intrinsic disadvantage due to the presence of the hypervisor which increases noise. This represents an additional impediment to HPC in the cloud.

VI. Optimizing Cloud Virtualization for HPC

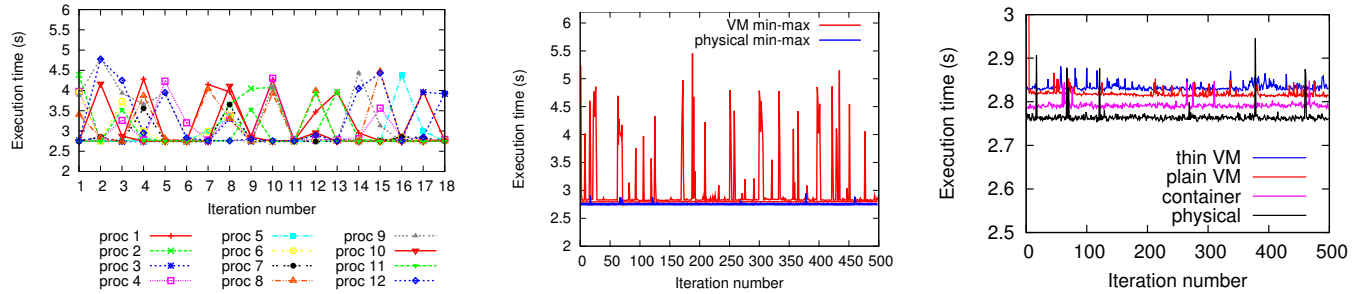
To mitigate the overhead of cloud platform, we investigate two optimization techniques for cloud: lightweight virtualization and CPU affinity.

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 virtual machines native accesses to physical network interfaces. In the thin VM configuration with IOMMU, a physical network interface is allocated exclusively to a thin 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 [21] share 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).

It should be noted that it is not our intention in this study to exhaustively compare multiple hypervisors. Our goal is to concentrate on the current state of device virtualization and provide valuable insights to cloud operators. We are aware of other research in the virtualization and networking communities to improve virtual drivers offering 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

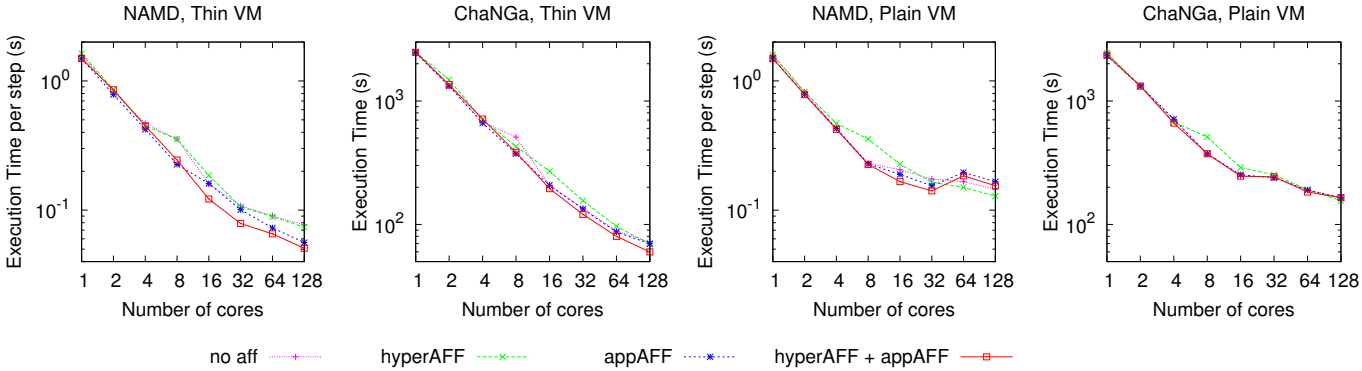


Fig. 7: Application Performance with various CPU Affinity Settings, using thin VM and plain VM; legend is at the bottom

and multi-tenancy. While HPC will of course benefit from that, we consider it orthogonal to the findings of this paper.

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 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 whereby users logging 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. In this figure, hyperAFF denotes the execution where hypervisor-level affinity is enabled and appAFF denotes the case when application-level affinity is

enabled. We see significant benefits, especially for thin-VM, when using both application-level and hypervisor-level affinity compared to the case when no affinity is used. However, the impact on NAMD running on plain VMs is not clear, which suggests that optimizing cloud for HPC is non-trivial. Hence, there remains a gap between the expectations of HPC users and cloud offerings, then the question is why and when should one move to cloud?

VII. 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 for OPEX, 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, sensitivity to CAPEX. 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 of all, utilization of the computing resources is typically quite high on HPC systems. Queue-based approach to scheduling always makes sure that a supercomputer is kept busy 24x7. This conflicts with the desirable properties of high-variability and low average utilization that make the cloud business model viable, as we describe above. In other words, 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. Then, 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 cloud infrastructure.

As we showed above, 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. Again, 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.

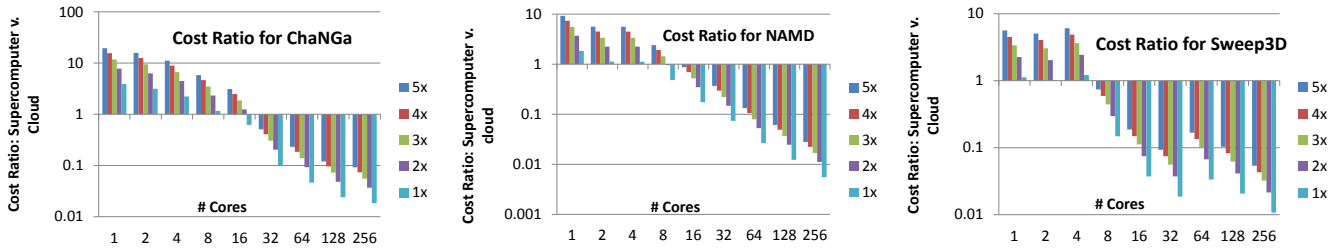
So, what are the conditions that can make HPC in the cloud a viable business model for both HPC users and cloud providers? Unlike large supercomputing centers, HPC users in small-medium enterprises are much more sensitive to the CAPEX/OPEX argument. For example, startups with HPC requirements (e.g., 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. Unfortunately, while cloud pricing is readily available, it is quite difficult to obtain reliable cost information. In some cases, like for cloud operators, the sensitivity of the information is intimately tied to the business model and is treated as proprietary and closely guarded. In other cases, like for supercomputing center, the complexity of accounting for and disentangling the per-node administrative, support and software licensing components makes it very difficult to compute the per-node total cost of ownership.

Based on our survey of cloud prices, known financial of cloud operators, published supercomputing costs, and a variety of internal and external data sources [36], we estimate that a cost ratio between 2x and 3x[1] is a reasonable approximate range capturing the differences between a cloud deployment and on-premise supercomputing resources today. Of course, these values will continue to fluctuate, possibly in unforeseen ways, so 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 for an HPC user 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. For example, Figure 8a-c show the cost differences

[1]To clarify our terminology, 2x indicates the case where 1 supercomputer core-hour is twice as expensive as 1 cloud core-hour.



(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.

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. We can see that smaller-scale executions in the cloud are advantageous, but after a certain scale it becomes counterproductive. This observation is consistent with the use of clouds for smaller runs (or peak burst) and for testing. 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.

In practice, HPC users will need to run a portfolio of applications and distribute them among dedicated resources and cloud. In that case, the techniques and analyses methodology we present in this work, are going to be very important to determine which of the application are the most profitable candidates for cloud execution.

VIII. Discussion: Cloud Bursting and Benefits

In the previous sections, we provided empirical evidence that our hypothesis holds: 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 its impact on performance can help answer (1) which application to burst to cloud, and (2) which cloud to burst to.

To further explore (1), let's consider a practical scenario where an organization needs to run a set of HPC applications, each requiring a performance guarantee (e.g., needing to meet a deadline), and has access to in-house, dedicated, HPC optimized resources that cannot meet the aggregated demand. Under a given performance constraint, how do we find a mapping of the application set to the available resources that makes best use of the dedicated HPC-optimized resources, minimizes the cost of bursting to the cloud, and meet the performance targets?

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 cloud-friendly applications, and attempt bursting to the cloud, applications that do not scale and have no chance of meeting the performance target. An intelligent mapping algorithm should be aware of application characteristics and understand that application which scale poorly on cloud should be allocated to dedicated resources first to maximize the utilization of an optimized supercomputer infrastructure. We believe that these techniques and associated tools for automating the mapping of applications to platforms will become increasingly important in the future.

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 applications that are more sensitive to networking performance, selecting a higher-cost HPC-optimized cloud, such as the offering that has Infiniband or 10G Ethernet. An intelligent mapper can provide guidance in selecting the best platform, thus enabling the end users to concentrate on application development by leaving the burden of platform choices to a semi-automated tool.

Figure 9 shows the conceptual architecture of the mapping tool. We start from an HPC application, and through off-line characterization extract an application *signature* capturing the most important dimensions such as communication profiles and problem size. Subsequently, given a set of applications to execute and a set of target platforms, we define a set of heuristics to map the applications to the platforms that optimize *parallel efficiency* (static mapping in Figure 9). Parallel efficiency (E) is a crucial metric and a concept central to our approach towards application characterization. It is defined as: $E = S/P$ where P is the number of processors, and Speedup (S) is defined as: $S = T_s/T_p$ where T_s is the sequential

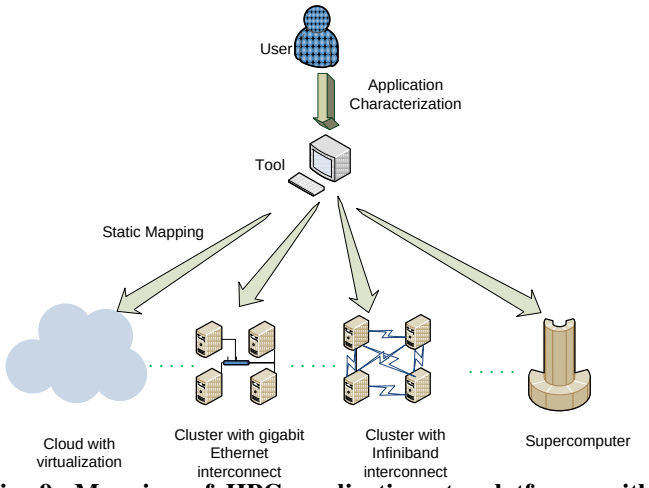


Fig. 9: Mapping of HPC applications to platforms with varying resources (e.g., different processor types and speed, interconnection networks, and virtualization overhead)

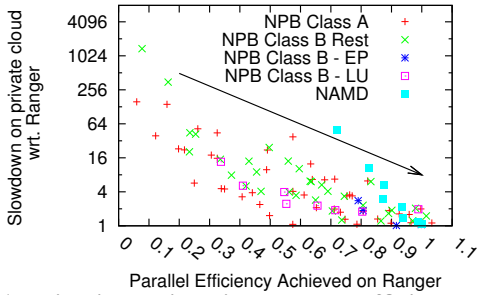


Fig. 10: Applications with high parallel efficiency are good candidates for cloud.

execution time and T_p is the parallel execution time. To study its impact, in Figure 10 we plotted the slowdown caused by moving an application from supercomputer to cloud vs. the parallel efficiency achieved on supercomputer. This was done for all applications run across different number of processors.

From Figure 10, we see the trend that applications which achieve high parallel efficiency on supercomputer suffer less slowdown when moved to cloud compared with applications with low parallel efficiency. Applications which achieved less than 90% parallel efficiency on Ranger suffered serious degradation on cloud implying that only applications with high parallel efficiency are possible candidates for execution on cloud. Hence, performance degradation on cloud will be less for applications with high computational granularity (hence low communication to computation ratio and high parallel efficiency). For mapping purposes, we focus on those application characteristics which can contribute to difference in application's expected parallel efficiency across different platforms. From the performance analysis, it is clear that communication time (and more precisely the communication-to-computation ratio) is a major contributor.

Using the application signature and platform characteristics, we can estimate relative application performance, execution cost and other relevant metrics of the available platforms and recommend the best platform for a given application instance based on user preferences. For each platform, the algorithm estimates the communication to computation ratio, calculates parallel efficiency, scales it by sequential performance, and normalizes it. Subsequently, it calculates normalized cost and recommends the best platform based on user preferences. For estimating communication time (T_{comm}) of an N-byte message, we can use the formula $T_{comm} = \alpha + N \times \beta$ where α is the per-message cost (startup cost) and β is the per-byte cost (inversely proportional of bandwidth).

Such a tool can be used to provide mapping recommendations under various scenarios: to maximize performance under budget constraints, to minimize cost under performance guarantees, or to consider an application set as a whole instead of individual application instances. For example, we can minimize cost of a set of applications under a hard allocation limitation while providing performance guarantees.

To demonstrate the potential impact of such a tool and provide a proof-of-concept, we evaluate the results obtained by a simple mapper based on the characterization mentioned above. It is not our intention in this paper to research accurate techniques for parallel performance prediction of complex applications. Our goal is to quantify the benefits of smart mapping to develop an understanding and foundation for HPC in cloud, which can promote further research towards additional techniques for more accurate results and complex applications. With this goal. we consider an application set with well-understood computation and communication patterns and available combination of two platforms: supercomputer (Ranger) and Eucalyptus cloud. From now on, we use "cloud" for Eucalyptus cloud (typical hypervisor-based resources), and "supercomputer" for Ranger.

A. Cost with Performance Guarantees

The question that we explore here is whether we can reduce the cost of an application execution while maintaining the desired level of performance. The Mapper estimates the number of processors required to achieve the same performance on cloud (typically larger than the number needed on supercomputer), and recommends the cost-optimal platform.

Figure 11 shows the results for various supercomputer/cloud pricing ratios. The application suffix is the number of processors (when run on supercomputer); for Jacobi, we consider multiple problem sizes, that is input matrix dimensions (e.g. size 1k $\times$ 1k). The target performance is that obtained by running the application on supercomputer. We normalized results with respect to execution on supercomputer, so that a normalized cost of 1 means that it is optimal to run the application on supercomputer for that pricing ratio, for example, that happens to all IS instances, regardless of the pricing ratio. The pricing ratio where the normalized cost $\neq$ 1 is when the mapper recommends cloud as the optimal platform. Thus, we can deduce the pricing ratio at which the optimal platform shifts from supercomputer to cloud for each application. We see that this cross-over point occurs at pricing ratio of 1, 2, and 4, for different applications. Another important observation is that savings vary for different applications;

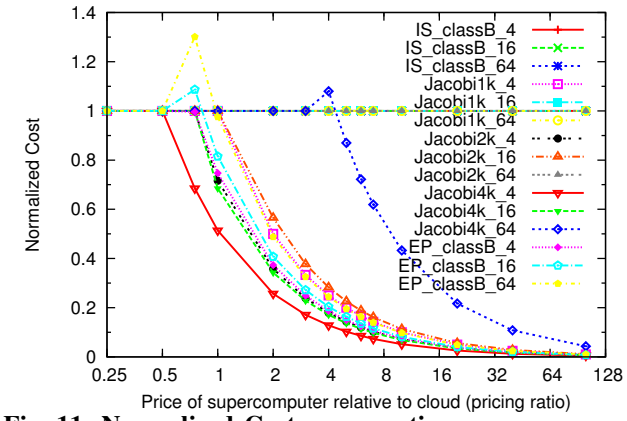


Fig. 11: Normalized Cost vs. execution on supercomputer vs. pricing ratio for FixedPerfMapper

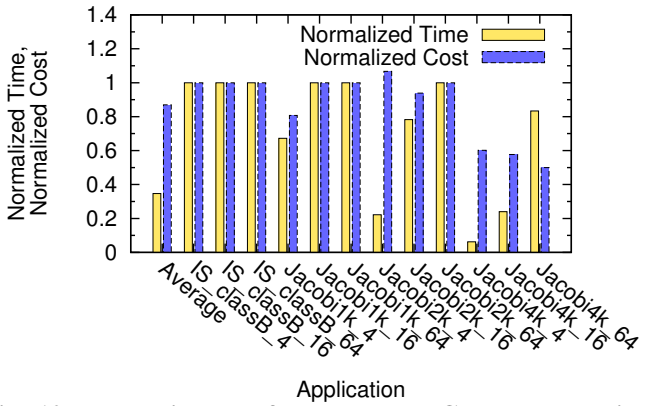


Fig. 12: Normalized Performance and Cost vs. execution on supercomputer for FixedCostMapper

for example, maximum savings were obtained for Jacobi4k_4 running on cloud. This observation is pivotal to intelligently map a group of applications under some constraints (e.g., a fixed SU budget) as discussed earlier.

The IS benchmark contains all-to-all collective communication as its dominant communication pattern. Using a theoretical estimation for this collective primitive is non-trivial since different MPI implementations use different broadcast algorithms which may even change with message sizes. Hence, we benchmarked the `MPI_Alltoall` for relevant message sizes for different processor counts on our platforms, and used these values in the mapper to predict parallel efficiency. Scientific applications are typically long-running, and are executed many times with different inputs, but same problem size. Such one-time benchmarking can be useful in those cases.

B. Performance with Constrained Budget

Another common situation is an HPC user with hard budget constraints wanting to find the execution platform that maximizes performance. We can use the Mapper by constraining the budget of each application, for example with the cost of its execution on supercomputer, and using a fixed-cost mapping algorithm to estimate the number of processors that fit the budget, while taking into account the parallel efficiency drop as we scale. The Mapper recommends the platform which would provide best performance, as shown in Figure 12, where we summarize the achieved performance benefits under budget constraints. We can see that with marginal performance penalty, significant cost benefits can be attained. A subtle point is that cost is affected by run-time since longer running means larger cost. Hence most supercomputer-friendly applications will still execute on supercomputer.

IX. Lessons Learned

We summarize here some important insights we found.

A hybrid cloud-supercomputer platform environment can outperform its individual constituents. While it may sound obvious that an application should be run on the environment it was initially developed and tuned for, there might be an underutilized resource which is "good enough" to get the job done sooner cheaper. Based on the study of the performance-cost tradeoffs, it is possible to get better performance for the same cost on one platform for some applications, and on another platform for another application. Hence, a hybrid environment can potentially perform better when catered to a wide class of HPC applications. 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 important to remove overheads for HPC in cloud. We have shown that giving VMs a more direct access to I/O (network and storage) reduces the performance gap between cloud and bare metal. For the HPC applications used, we have shown that the computational overhead of virtualization is negligible, and that much more important is the communication overhead of virtualized networking. We believe that a promising research direction is to use thin virtualization and container-based virtualization for HPC. In particular, since the same hardware of a web-oriented cloud infrastructure can be reused, this approach shows the potential for building hybrid clouds that can support both HPC and commercial workloads. Such a hybrid cloud stack would however require proper tuning or VM re-provisioning for HPC applications, which is a fertile topic for future research.

Application characterization in the HPC-cloud space is challenging but the benefits are substantial. As expected, identification of the main application characteristics which influence performance-cost tradeoffs for complex HPC applications is a non-trivial task. However, it is critical for the identification of an application's suitability for cloud. More research is necessary to be able to quickly identify important traits for complex applications such as those with dynamic and irregular communication patterns. Once the application is characterized, it is possible to determine if there is an economic benefit to execute it in the cloud and up to what scale specifically.

X. Conclusions and Future Work

Through a performance 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. Applications with non-intensive communication patterns are good candidates for cloud deployments. For communication-intensive applications, supercomputers remain the optimal platform, largely due to the overhead of network virtualization in the cloud.

Although the findings are similar to the behaviour of early Beowulf clusters, those clusters are quite different from todays clouds: processor, memory, and networking technologies have tremendously progressed. The appearance of virtualization between hardware and the OS introduces multi-tenancy, resource sharing and several other new effects. We believed it was time to repeat these experiments. The fact that they generate similar results (to the extent they do), while not shocking, is valuable information for the community and helps to better understand which applications are cloud candidates, and where we should focus our efforts to improve the cloud performance.

In addition to evaluating the suitability of HPC applications in cloud and identifying performance bottlenecks, we also suggested techniques for improving performance in cloud, specifically CPU affinity and alternative lightweight virtualization mechanisms. We also dived into economic aspects of HPC in cloud and showed that there is significant cost-saving potential in using hybrid platform environments and intelligent mapping of applications to available platforms. Finally, we described how we could automate the mapping using a combination of application characteristics, platform parameters, and user preferences. In the future, we plan to enhance the techniques to generate better "application signatures" and smarter mapping.

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] "Magellan - Argonne's DoE Cloud Computing," http://magellan.alcf.anl.gov.

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

[7] 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.

[8] 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.

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

[10] 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

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

[12] 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.

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

[14] 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.

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

[16] Infiniband Trade Association, "Infiniband Architecture Specification, Release 1.0," Tech. Rep. RC23077, October (2004).

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

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

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

[20] 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.

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

[22] "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.

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

[24] 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.

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

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

[27] 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.

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

[29] Y. Yoon, J. C. Browne, M. Crocker, S. Jain, and N. Mahmood, "Productivity and Performance through Components: the ASCI Sweep3D Application: Research Articles," *Concurrency and Computation: Practice and Experience*, vol. 19, no. 5, pp. 721–742, 2007.

[30] 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.

[31] 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.

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

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

[34] 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*.

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

[36] 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