Case Study for Running HPC Applications in Public Clouds

Qiming He
Open Research, Inc.
Qiming.He@openresearchinc.com

Shujia Zhou*, Ben Kobler, Dan Duffy, Tom McGlynn
NASA Goddard Space Flight Center
{shujia.zhou, benjamin.kobler, daniel.q.duffy, thomas.a.mcglynn}@nasa.gov

ABSTRACT

Cloud computing is emerging as an alternative computing platform to bridge the gap between scientists' growing computational demands and their computing capabilities. A scientist who wants to run HPC applications can obtain massive computing resources 'in the cloud' quickly (in minutes), as opposed to days or weeks it normally takes under traditional business processes. Due to the popularity of Amazon EC2, most HPC-in-the-cloud research has been conducted using EC2 as a target platform. Previous work has not investigated how results might depend upon the cloud platform used. In this paper, we extend previous research to three public cloud computing platforms. In addition to running classical benchmarks, we also port a 'full-size' NASA climate prediction application into the cloud, and compare our results with that from dedicated HPC systems. Our results show that 1) virtualization technology, which is widely used by cloud computing, adds little performance overhead; 2) most current public clouds are not designed for running scientific applications primarily due to their poor networking capabilities. However, a cloud with moderately better network (vs. EC2) will deliver a significant performance improvement. Our observations will help to quantify the improvement of using fast networks for running HPC-in-the-cloud, and indicate a promising trend of HPC capability in future private science clouds. We also discuss techniques that will help scientists to best utilize public cloud platforms despite current deficiencies.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Performance evaluation

General Terms

Performance

*This author is also affiliated with University of Maryland, Baltimore County(szhou@umbc.edu).

Keywords

Cloud Computing, High-Performance Computing, Benchmarks

1. INTRODUCTION

Scientists are often interested in problems that are too small for supercomputers like IBM Blue Gene series, but too big for desktop PC or typical facility computers. Traditional computing resource provisioning does not provide a timely and economical solution to satisfy their needs, mostly due to operational inefficiency such as prolonged procurement and installation processes and high maintenance cost.

Cloud computing [23][1] is emerging as an alternative computing platform to bridge the gap between scientists' growing computational demands and their local computing capabilities. Recent years have seen an increasing adoption of cloud computing in a wide range of scientific disciplines, such as high-energy and nuclear physics, bioinformatics, astronomy and climate research [6]. Cloud computing is also an excellent collaboration tool that allows scientists to share information globally, and replicate others' work in the cloud that contains identical application, datasets and environmental settings.

In this paper, we focus on evaluating the technical capability of current public cloud computing platforms, and their suitability for running scientific applications, especially High Performance Computing (HPC) applications. We have built upon the work of previous researchers in compiling a suite of HPC benchmarks and port them into three public clouds.

Evaluating platforms' capabilities and benchmarking results, we note that current public clouds do not seem to be optimized for running scientific applications. Many public cloud platforms have slow network connections between virtual machines (VM), which often becomes a bottleneck for running some HPC applications. However, in contrast to the results obtained by previous researchers using EC2 [28][21], we have obtained some satisfactory results from a public cloud with relatively faster and more consistent networks.

While current HPC systems in the cloud are far less powerful than dedicated supercomputers, our work provides evidence that fast interconnection will significantly improve the scalability and performance of a cloud-based HPC system. We believe this work will show a promising trend of increasingly powerful HPC capability in future science clouds with faster

networks, like NASA's Nebula [22] and DOE's Magellan [7].

For scientists who wish to port their applications into a current public cloud, it is essential for them to understand its HPC capability before deploying their applications. Due to suppliers' unspecified implementations, the performance delivered by a cloud HPC system could be quite different from their in-house systems, even though two systems have similar processors, memory and cluster size.To understand a cloud's HPC capability quantitatively, scientists can choose benchmarks close to the nature of their applications (e.g., solving linear equation, Monte Carlo simulation) and create a cloud image including both, so that they can quickly benchmark HPC capability of any size cluster before running time-consuming applications packed in the same image. In order to avoid inter-process communication over slow network and best utilize increasingly popular multi-core processors, scientists may also need to consider changing their programming paradigm and/or find application best suited to run in the cloud. Cloud computing allows users to install any software (as root) to support their programming. For example, we updated *gcc* from 4.1 (in base OS) to 4.4 to have OpenMP supported from the compiler. Such flexibility is not normally available to end-user in a dedicated HPC system.

Some HPC providers allocate computing resources for users' proposed research work. Their HPC facilities may not grow at the same pace as ever-growing computational demands, or could be limited by local power supply. Instead of rejecting users' applications, they may consider redirecting users' requests to a public cloud as an alternative platform. This will not only allow HPC capability to grow more economically, but may also reduce 'time-to-solution' from user perspective. For example, dedicated HPC systems normally use job queue [3]. In peak hours, user submitting a job request may have 25% chance to have their cluster ready in 100 seconds, and 75% chance in 1000 seconds. From our experience, we can have entire cloud cluster provisioned in less than 5 minutes, and thus save a considerable amount of time which would otherwise be spent to request a cluster from a job queue.

This paper is organized as follows: in section 2, we define our scope and approach in our case study; in section 3, we detail our experiment setup including benchmarks, applications and specifications for public cloud platforms; in section 4, we present experiment results and our analysis; in section 5, we conclude this paper with some forward-looking remarks.

2. BACKGROUND AND METHODOLOGY

According to NIST definition [23], *cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.....* However, cloud computing means different things to different people, ranging from using web-based email to externalizing data storage. In this paper, we choose IAAS (Infrastructure-As-A-Service) type of cloud where we have full control of a VM (guest OS) to install applications and customize computing environment. Our focus is primarily on running HPC (i.e., computing-intensive) applications in the cloud.

Most public cloud computing suppliers allow users to create an arbitrary size cluster of cloud servers. For HPC-like applications, users would like to choose the 'highest' cloud server instances which are normally equipped with state-of-the-art multi-core processors and a large amount of memory. However, cloud computing relies on virtualization technology which may introduce unknown performance overhead to HPC applications. Some researchers have shown that modern VM technology can deliver near-native performance, e.g.,[29] shows that VM-based HPC system poses no statistically significant overhead. As suspected by other researchers [28][21][11], among other technical factors, (slower) network connection is the primary source for HPC performance degradation in Amazon EC2 cloud.

In this paper, in order to perform a quantitative and comparative case study, we adopt a suite of benchmarks used by previous researchers, and deploy them to three public clouds. We anticipate benchmarking results from clouds with different settings will highlight the bottleneck of running HPC applications in the cloud.

3. EXPERIMENTS

For each cloud computing platform, our experiment starts with launching a server instance using Linux as a base OS (64-bit if available), and install supporting software and libraries onto the base OS, e.g., MPICH2. We upload and compile benchmarks and applications, and change application and environment configurations accordingly, e.g., password-less ssh access. We use the imaging tool supplied by the platform to create a server image stored in the cloud. When running benchmarks and applications, we launch multiple server instances from the image to form a computing cluster.

3.1 Benchmarks and Applications

3.1.1 NPB

NAS Parallel Benchmark (NPB) is designed by NASA/NAS to evaluate the performance of parallel supercomputers. The benchmark is derived from Computational Fluid Dynamics (CFD) applications. NPB benchmarking in EC2 and NCSA supercomputers has been reported in [28]. We choose to run the same class benchmark (i.e., class=B) using NPB-3.3.

3.1.2 HPL

HPL (High Performance LINPACK) is a well-known TOP500 benchmark using a work load that is characteristic of some scientific applications. HPL benchmarking in EC2 cloud has been reported in [21]. Like [21], we implement HPL-2.0 [13] using GotoBlas-1.10 [10]. We use hpl-calculator [14] to determine benchmark parameters (i.e., N, NB, P and Q) to best utilize system memory and maximize the use of threads on each multi-core cloud server.

3.1.3 CSFV

In addition to above classical benchmarks, we choose to port a full-size (with 110K lines of code) parallel computing application into the cloud. Our work is in line with [6] effort

to run climate models in the cloud. We choose a NASA climate and numerical weather prediction application known as Cubed-Sphere-Finite-Volume (CSFV) Dynamic Core [2]. CSFV is a MPI-based Fortran program compiled by Intel Fortran compiler (*ifort*). At current numerical weather prediction resolution 25km, the CSFV scales at an ideal rate within a dedicated SGI Altix cluster of 512 CPUs. In our experiment, for the sake of proof-of-concept, we choose 50km resolution and set the length of simulation to 6 hours.

3.2 Public Clouds

We have investigated a few public cloud computing platforms and selected three platforms for this paper, based on both technical and economical reasons. Table 1, 2 and 3 show technical capabilities of selected platforms. In order to maximize server performance and minimize the impact of unpredictable processor sharing, on each cloud platform, we choose the 'highest' possible server instance with maximum number of cores denoted by (*). Table 4 shows theoretical peak performance (RPeak) per server instance on three platforms.

Server instance	RAM(GB)	Cores	Disk(GB)	Price/hr
Small	1.7	1	160	$0.085
Large	7.5	2	850	$0.34
Extra Large	15	4	1690	$0.68
Double Extra Large	34.2	4	850	$1.20
Quad Extra Large	68.4	8	1690	$2.40
High-CPU Medium	1.7	2	350	$0.17
High-CPU Extra Large*	7	8	1690	$0.68

Table 1: EC2 Linux server specification and price

Server instance	RAM(GB)	Core	Disk(GB)	Price/hr
0.5G	0.5	1	30	$0.095
1G	1	1	60	$0.19
2G	2	1	120	$0.38
4G	4	3	240	$0.76
8G*	8	6	480	$1.52

Table 2: GoGrid Linux server specification and price

Server instance	RAM(GB)	Core	Disk(GB)	Price/hr
Small	1	2	8	N/A
Medium	1.8	4	18	N/A
Large*	3.6	8	38	N/A

Table 3: IBM Linux server specification

Server	CPU (GHz)	Hyper-visor	Processor/ node	Core/ Processor	RPeak (GFLOPS)
EC2	2.33	Xen	2	4	74.56
GoGrid	3.00	Xen	1	6	72.00
IBM	2.93	?	2	4	93.76

Table 4: Cloud Server RPeak

3.2.1 Amazon EC2 Cloud[5]

Amazon EC2 is the most popular cloud computing platform, and has been the target platform for numerous academic and commercial applications. It uses dual-socket quad-core Intel Xeon processors E5345@2.33GHz, which is 2-3 years older compared to other two platforms in our test. Like other researchers [28][21], we choose High-CPU-Extra-Large instead of High-memory-Quad-Extra-Large with the same number of cores.

3.2.2 GoGrid Cloud[9]

GoGrid sever uses Intel Xeon processors E5459@3GHz. We suspect GoGrid uses high-end servers with 8-socket and quad-core processors (up to 32 cores). However, due to its unspecified server provisioning mechanism, not all cores are made available for users' requests. Unlike the other two 8-core-per-node platforms, GoGrid decides to provide up-to six-core-per-node. A server's price is proportional to the memory size (not to the number of processors).

3.2.3 IBM Cloud[15]

IBM is providing a cloud computing platform, currently in beta and free. It uses Intel's new Nehalem processor X5570@2.93GHz. However, only 32-bit Linux OS is available at this writing.

3.2.4 Other Clouds

Other public cloud platforms offer similar or inferior IAAS from technical and pricing perspectives, e.g., Rackspace (a.k.a., Mosso)[24] cloud server has two Quad-core operating at 2GHz, but user can only access 4-core at a time. The documented network throughput, depending on server instance, can be slightly better than 100Mbps; Flexiscale [8] provides up to 4 cores per server but only 10Mbps network. We exclude other 'managed-hosting' cloud platforms where user has to reserve and pay for a server instance for at least a month, because it does not provide an economic advantage of using clouds for 'ad-hoc' computing tasks.

4. RESULT AND ANALYSIS

4.1 Single server instance performance

The objective here was to evaluate single cloud server instance performance without inter-process communication over the network. This focuses attention on the virtualization overhead. Modern VM technology should allow user to achieve near-native performance in the cloud, unless cloud suppliers implement some resource management (e.g., processor sharing) that affect the application negatively. We run NPB-OMP (OpenMP-based shared-memory communication with threads) on three cloud platforms and compare our results with that of NCSA 'Abe' system as reported by [28]. We find that the performance delivered by single cloud server instance from all platforms is comparable to NCSA supercomputer as shown in Figure 1[1], that means virtualization technology does not add significant overhead to HPC performance. IBM has overall the best performance because of the use of the latest processor.

4.2 Cloud network

Unlike processor-type and memory-size, a cloud server's networking capacity (throughput, latency, packet-loss-ratio, etc) is not clearly specified on most public cloud platforms. Some researchers have to take a 'probing' approach to understand networking characteristics of cloud platforms, e.g., [11] provides some in-depth analysis for TCP/UDP/IP traffic profiling in EC2. In this paper, our goal is to use simple 'probing' tools, at both IP and MPI messages level, to highlight networking differences among public clouds, which will help us

[1]We do not observe large variations across multiple runs of the same benchmark suite. For the sake of clarity and comparison with others' work, we choose to present the result from one instance.

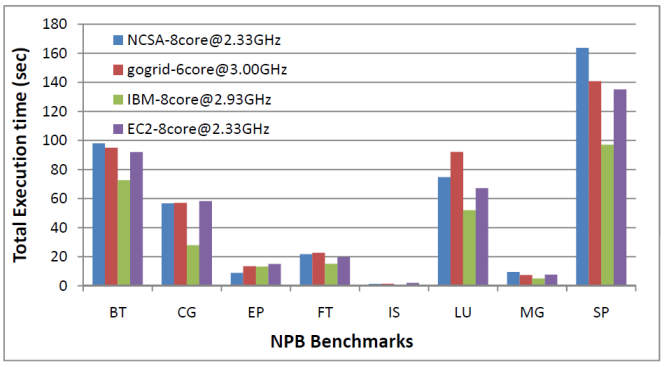

Figure 1: NPB OMP on public clouds

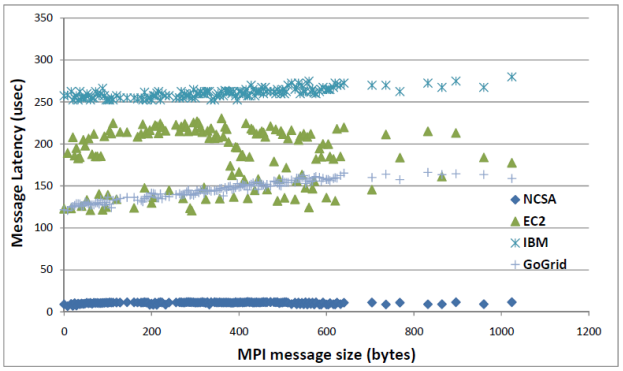

Figure 2: MPI message latency

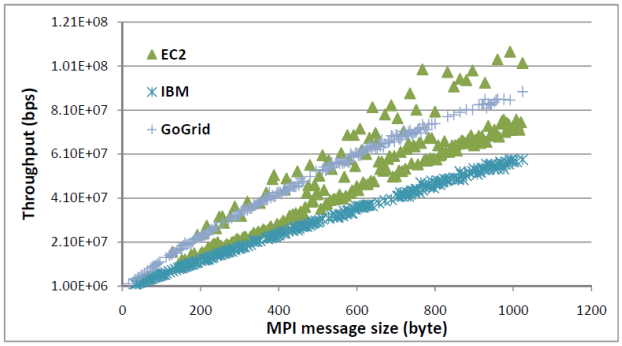

Figure 3: MPI message throughput

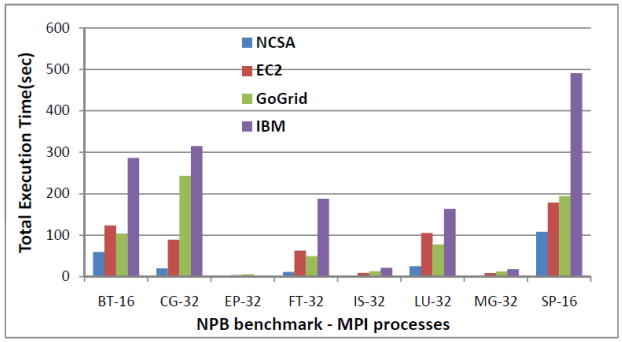

Figure 4: NPB-MPI benchmark results

to understand the network impact on running HPC applications in our case study. At TCP/IP level, we use *iperf* [17] with TCP windows size of 16KB and obtain median TCP/IP throughput on three platforms as shown in Table 5. Our EC2's measurement is similar to the result obtained by [11]. Therefore, we believe EC2 and GoGrid use 1Gbps and IBM uses 100Mbps Ethernet. EC2 may implement some unspecified resource or security management policy in its network, and thus result in slower network performance. It is worth noting that EC2 also has a unique network characteristics, i.e., no two servers are guaranteed to be provisioned in the same subnet, even though they are requested in a back-to-back fashion. It normally takes 3-4 multi-hop for any IP packet traveling between any EC2 server instances [6][11]. Both GoGrid and IBM simply put server instances in the same subnet, confirmed by *traceroute*. At MPI message

Cloud	EC2	IBM	GoGrid
Bandwidth(Mbps)	750	100	1000
Multi-hop	Y	N	N

Table 5: Network characteristics probed by *iperf* and *traceroute*

level, we use *mpptest* [20] tool to measure MPI message latency and throughput among 32 MPI processes distributed on multiple nodes. Similar results can be obtained by other tools like IBM's pingpong [16]. We choose *mpptest* to compare our results with that of NCSA [28]. As shown in Figure 2, the end-to-end delay of all clouds are 1-2 orders of magnitude worse than that of NCSA. The IBM cloud has the highest latency, mostly due to its slow networks. EC2 cloud has the largest variation, that means application-level MPI messages share the same characteristics of network-level messages as reported by [11], which may lead to unpredictable and undesirable behavior of MPI-based applications. Figure 3 shows MPI message throughput in three cloud platforms. The GoGrid network outperforms other two clouds in terms of average latency and variation.

4.3 MPI-based benchmarks and applications

4.3.1 NPB

We run NPB-MPI benchmark on three clouds and compare our results with NCSA as shown in Figure 4. Comparatively speaking, IBM cloud underperforms others although it has the latest processor and the maximum number of cores per node. This provides some insights on the fact that networking plays an important role in the MPI-based parallel computing. Seemingly, GoGrid and EC2 deliver approximately the same performance. However, due to the special requirement of NPB and GoGrid's 6-core-per-node server, in order to set up a cluster and assign 16/32 MPI processes to 16/32 processors, we have to launch 3 GoGrid instances (vs. 2 in EC2) and 6 instances (vs. 4 in EC2) respectively. This means more inter-process communications are carried out over the network in the GoGrid cluster. Therefore, it is reasonable to anticipate better results from GoGrid if we were able to use 8-core-per-node servers.

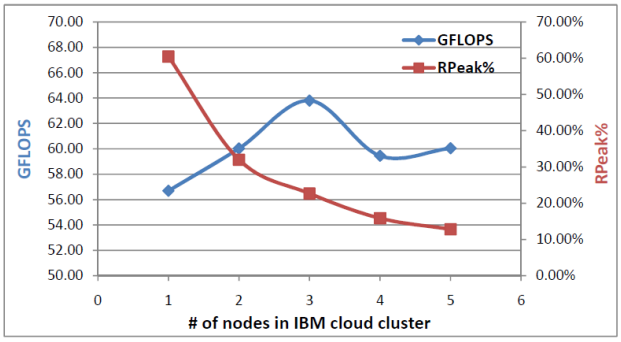

Figure 5: HPL benchmark reulks in the IBM cloud

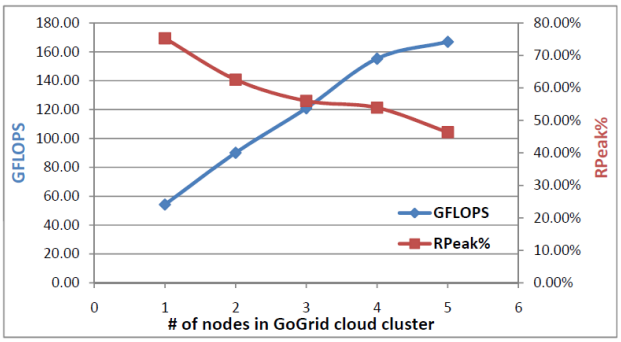

Figure 6: HPL benchmark results in the GoGrid cloud

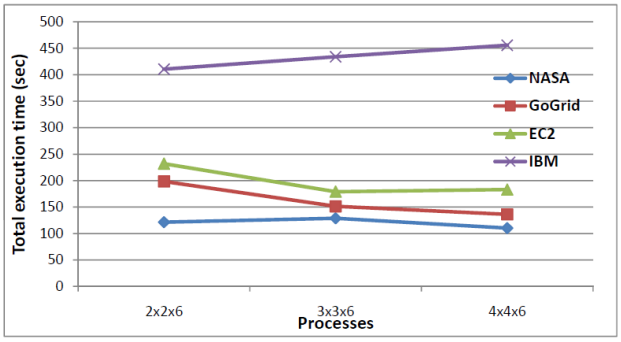

Figure 7: CSFV benchmark in public clouds vs. in NASA system

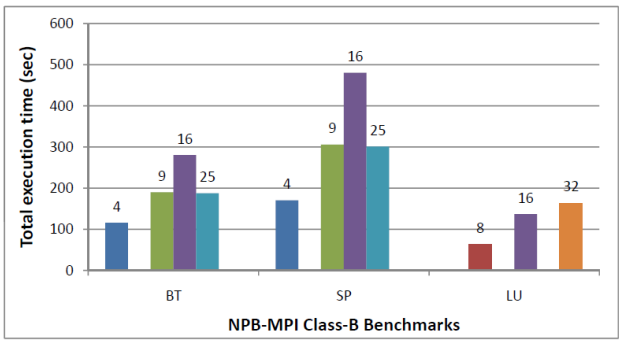

Figure 8: NPB-MPI benchmark results in the IBM cloud

4.3.2 *HPL*

We ran HPL in the GoGrid and the IBM cloud and focus on the overall FLOPS delivered by increasingly large clusters. HPL benchmarking results on EC2 has been reported in [21] Figure 1, which shows exponentially decreased performance in terms of percentage of theoretical peak (RPeak%), i.e., the system does not scale out of a single box.

The HPL benchmarking results we obtained in the IBM cloud is similar to that of EC2. As shown in Figure 5, the system stops scaling up after 3 nodes join the cluster. RPeak% decays nearly exponentially.

The HPL benchmarking results in GoGrid make us optimistic about running HPC applications in the cloud. As shown in Figure 6, when the cluster increases from 1 to 5, the overall system FLOPS increase monotonically from 55 to 165 GFLOPS. RPeak% is still decaying, but in a nearly linear fashion from 75% to 40%.

4.3.3 *CSFV*

CSFV can be decomposed to NxNx6 MPI processes. In our experiment, we firstly choose N=2 and launch a small cluster with 24 cores in three clouds. We also run oversubscribed cases, i.e., run 3x3x6 and 4x4x6 MPI processes in the the same cluster. We compare the total execution time with that from NASA dedicated HPC system. As shown in Figure 7, IBM cloud performance does not scale satisfactorily. GoGrid outperforms EC2 in all cases. Oversubscription will improve GoGrid's performance, and make it underperform NASA system only by 10-20% while EC2 underperforms NASA systems by 50+%.

4.4 Discussions

In this section, we discuss other technical and non-technical factors that may affect HPC applications in the cloud.

4.4.1 *Parallel programming paradigm*

In previous sections, our case studies include pure MPI (via distributed-memory communication) and pure OpenMP (via shared-memory communication) implementations. In order to overcome network problems posed by current public clouds, one may consider combining these two parallel programming paradigms. We use NPB-MZ benchmark to demonstrate the effectiveness of using hybrid MPI+OpenMP [19] to improve HPC application scalability in the cloud. Firstly, we use the IBM-cloud (the one with the slowest network) and NPB-MPI benchmark to show that a pure MPI implementation does not scale up from a single server. We chose BT, LU and SP benchmarks in order to compare with results from hybrid approach. Due to the special requirements of NPB-MPI benchmarks, we choose 4, 9, 16 and 25 MPI processes for BT and SP, and 8, 16 and 32 MPI processes for LU respectively. Figure 8 shows performance starts degrading when the cluster size is larger than a single node which can hold up to 8 MPI processes (without oversubscription).

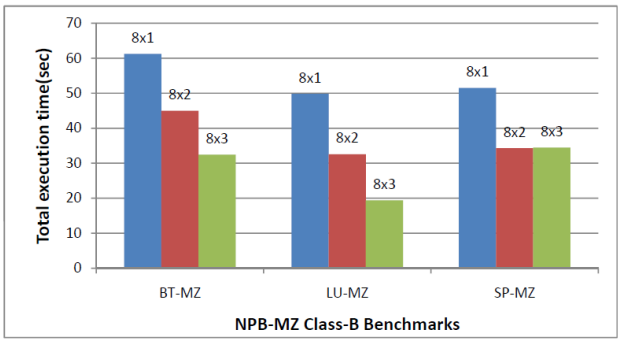

Figure 9: NPB-MZ benchmark results in the IBM cloud

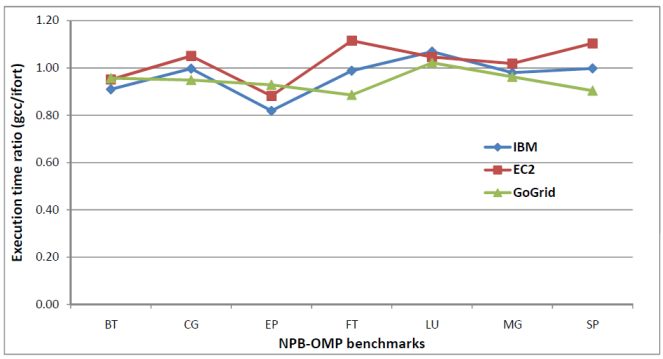

Figure 10: compiler difference

The hybrid OpenMP+MPI implementation uses two levels of parallelization. OpenMP is applied to fine-grained intra-zone parallelization and MPI is used for coarse-grained inter-zone parallelization. NPB-MZ benchmarks (with the size similar to NPB-MPI) scale out up to 3 nodes as shown in Figure 9 where MPI is used between nodes and each MPI task has 8 OpenMP threads.

4.4.2 Open source vs. commercial software

Software license issue has not been clearly defined in the cloud. Free and Open Source Software (FOSS) are not only extremely popular in scientific communities but also technically capable to support HPC applications, e.g., Linux as an OS, *gcc/gfortran* as compilers, OpenMPI/MPICH as runtime libraries.

In order to understand the performance difference between using FOSS vs. commercial software, we perform a case study using GNU *gfortran* and Intel *ifort* compiler. We choose NPB-OMP benchmark to minimize network impact in our case study. We use '-opemp -O3' option for *ifort* and '-fopenmp -O3' for *gfortran* respectively, and other compiler options are out-of-box. We choose 32-bit CentOS-5 for IBM, and 64-bit CentOS-5 for both GoGrid and EC2. From Figure 10, there is no significant performance difference using commercial and FOSS compiler for all (Fortran) benchmarks in all three clouds.

4.4.3 Cloud-friendly applications

Despite current deficiencies in public clouds, some types of application can still benefit greatly from a large of number of processors provisioned in the cloud, e.g., applications with less inter-process communications like 'Embarrassingly Parallel' (EP) benchmark in Figure 1 and Figure 4 where little or no performance degradation is observed. EP represents a wide range of applications that demand more computation but less communication capabilities, such as Monte Carlo simulation. For these types of 'cloud-friendly' applications, one can easily find cloud servers that have more memory and newer processors than his/her in-house HPC systems, and thus achieve better performance in the cloud.

4.4.4 Cloud computing economics

Unlike many dedicated HPC systems where FLOPS is the goal and the only goal for optimization, cloud computing can be formalized as a multi-objective optimization problem under both technical and economic constraints. Many new terminologies have been introduced from economics perspective, such as FLOPS per-dollar or per-watt [1][21]. While the entire cloud computing economics topic is quite beyond the scope of this paper, we do notice that, as far as HPC is concerned, the system performance does not scale at the same rate as the price rises. For example, we run NPB-MPI benchmark using a GoGrid cluster that consists of the same number of 1G and 4G servers. 4G cluster only outperforms 1G cluster by 30%, but the price is quadrupled. Some user may prefer a solution that is 30% slower but 4X cheaper. Cloud computing platforms make it possible for users to choose the most efficient computing resources suited for their application and budget. One of its advantage (over other on-demand computing like grid computing on VM) is to make computation as a commodity, i.e., one can buy computation like buying books online, and can have root access to a large number of VMs instantly without installing special software. It is best-suited for moderate-size computing tasks arising from spontaneous and ad-hoc (research) projects.

5. RELATED WORK

In the past decade or so, numerous HPC benchmarks have been studied extensively in either dedicated systems [4] [18] or grid computing environments [26]. Recently, cloud computing has attracted many researchers to re-run well-known benchmarks in the cloud to evaluate its viability as an alternative HPC platform [27][25][28][21]. In contrast to results of eariler related work which are mostly obtained from EC2 platform, our results from non-EC2 clouds are new. Actually, we find that EC2 is not an ideal platform to run HPC applications due to its 'unique' networking characteristics aforementioned. This, in part, explains some pessimistic conclusions (for running HPC in the cloud) drawn by other researchers.

HPC benchmarks used by this paper and other researchers have relatively small and portable code base. There are few literature reports on running special-purpose and production-size application in the cloud. We have demonstrated that cloud VMs are well-suited for porting large-size applications, at both compilation- and run-time.

6. CONCLUSION AND FUTURE WORK

From our research, some clouds are performing better than others, but we do not have a cloud worth recommending for running serious HPC applications at this point, because most of public clouds are optimized for running business applications instead of HPC. However, once cloud vendors consider HPC as in-scope, we believe they can make their clouds more HPC-friendly, by making slight changes to their existing infrastructure, e.g., GoGrid could open up eight or more cores per-node; IBM could upgrade their network to use commodity Gigabit NICs and switches, etc. New science clouds platforms will be equipped with much better processing and networking capabilities, e.g., DoE Magellan cloud will use Infiniband [7], Penguin Computing's HPC-As-A-Service will use either GigE or Infiniband [12]. That said, we see a promising trend for HPC-in-the-cloud.

Our cluster size for this preliminary study is ∼10 nodes. In the future, we would like to deploy HPC applications to a cloud cluster with 100+ or even more nodes. Currently, there some technical difficulties to scale HPC-in-the-cloud to supercomputing size. For example, some vendor requests special approval for user to launch 20+ servers; other vendor only provides limited number of IPs in a subnet; To deploy an application to a larger cloud cluster, we also have to automate our configuration process to deal with issues like dynamic IP on each node. In the future, we would also like to report our benchmarking results on other cloud platforms when they become available, e.g., NASA's Nebula. Most research on HPC is conducted in dedicated systems. As cloud computing gains wider acceptance, we anticipate some research has to be revisited, considering new technical and economical constraints imposed by cloud computing.

7. ACKNOWLEDGMENT

This work is supported in part by NASA SBIR contract NN09CD71P and NNX10CD82P. The authors would like to thank IBM Systen Technology Group for partial support.

8. REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. Technical report, UC Berkeley, 2009.

[2] Cubed-sphere finite-volume dynamic core(fvcore). `http://sivo.gsfc.nasa.gov/cubedsphere.html`.

[3] J. B. Daniel Nurmi and R. Wolski. QBETS: Queue bounds estimation from time series. In *JSSPP*, pages 76–101, 2007.

[4] J. J. Dongarra, P. Luszczek, and A. Petitet. The LINPACK benchmark: past, present and future. *Concurrency and Computation Practice and Experience*, 15(9):803.820, 2003.

[5] Amazon EC2 cloud. `http://aws.amazon.com/ec2`.

[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. In *Cloud Computing and Its Applications*, October 2008.

[7] M. Feldman. HPCwire: DOE labs to build science clouds. `http://www.hpcwire.com/specialfeatures/cloud\_computing/news/DOE-Labs-to-Build-Science-Clouds-64189872.html`.

[8] Flexiscale cloud. `http://www.flexiscale.com`.

[9] Gogrid cloud. `http://www.gogrid.com`.

[10] Gotoblas. `http://www.tacc.utexas.edu/tacc-projects`.

[11] T. S. E. N. Guohui Wang. The impact of virtualization on network performance of amazon ec2 data center. In *INFOCOM*, 2010.

[12] Hpc as-a-service. `http://www.penguincomputing.com/POD/HPC_as_a_service`.

[13] High-performance LINPACK. `http://www.netlib.org/benchmark/hpl`.

[14] Hpl-caculator. `http://hpl-calculator.sourceforge.net`.

[15] IBM cloud. https://www-949.ibm.com/cloud/developer/login.jsp.

[16] IBM MPI benchmark. `http://www.intel.com/software/imb`.

[17] iperf. `http://sourceforge.net/projects/iperf`.

[18] H. Jin, M. Frumkin, and J. Yan. The OpenMP implementation of NAS parallel benchmarks and its performance. *NASA Ames Research Center,. Technical Report NAS-99-011*, 1999.

[19] E. L. Lusk and A. Chan. Early experiments with the openmp/mpi hybrid programming model. In *IWOMP*, pages 36–47, 2008.

[20] mpptest. `http://www.mcs.anl.gov/research/projects/mpi/mpptest/`.

[21] J. Napper and P. Bientinesi. Can cloud computing reach the top500? In *UCHPC-MAW '09: Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*, pages 17–20. ACM, 2009.

[22] NASA NEBULA. `http://nebula.nasa.gov`.

[23] NIST definition of cloud computing. `http://csrc.nist.gov/groups/SNS/cloud-computing/index.html`.

[24] Rackspace cloud. `http://www.rackspacecloud.com`.

[25] J. Rehr, F. Vila, J. Gardner, L. Svec, and M. Prange. Scientific computing in the cloud. *Computing in Science and Engineering*, 99(1), 5555.

[26] A. Snavely, G. Chun, H. Casanova, R. F. V. der Wijngaart, and M. A. Frumkin. Benchmarks for grid computing: a review of ongoing efforts and future directions. *SIGMETRICS Perform. Eval. Rev.*, 30(4):27–32, 2003.

[27] V. Stantchev. Performance evaluation of cloud computing offerings. In *2009 Third International Conference on Advanced Engineering Computing and Applications in Sciences*, page 187.192, 2009.

[28] E. Walker. Benchmarking amazon ec2 for high-performance scientific computing. *;LOGIN:*, 33(5), 2008.

[29] L. Youseff and et. al. Evaluating the performance impact of xen on mpi and process execution for hpc systems. In *VTDC '06: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, 2006.