Benchmarking Bare Metal Cloud Servers for HPC Applications

P. Rad[1], A T Chronopoulos[2], P. Lama[2], P. Madduri[2], C. Loader[2]
[1]Department of Computer Engineering, University of Texas at San Antonio
[2]Department of Computer Science, University of Texas at San Antonio
1 UTSA Circle, San Antonio, Texas 78249, U.S.A.

***Abstract*—Cloud Computing is an ever-growing paradigm shift in computing allowing user's commodity access to compute and storage services. As such cloud computing is an emerging promising approach for High Performance Computing (HPC) application development. Automation of resource provision offered by Cloud computing facilitates the eScience programmer usage of computing and storage resources. Currently, there are many commercial services for compute, storage, network and many others from big name companies. However, these services typically do not have performance guarantees associated with them. This results in unexpected performance degradation of user's applications that can be somewhat random to the user. In order to overcome this, a user must be well versed in the tools and technologies that drive Cloud Computing. One of the state of the art cloud systems, is a cloud system that provides bare metal server instances on demand. Unlike traditional cloud servers, bare metal cloud servers are free from virtualization overheads, and thus promise to be more suitable for HPC applications. In this paper, we present our study on the performance and scalability of Openstack based bare metal cloud servers, using a popular HPC benchmark suite. Our experiments conducted at UTSA Open Cloud Institute's cloud system with 200 cores demonstrate excellent scaling performance of bare metal cloud servers.**

Keywords- Bare Metal, Cloud computing, MPI, HPC benchmarks

I. INTRODUCTION

Today there are growing interests among the academic and commercial HPC users to utilize cloud computing as a cost effective alternative for their computing needs. Cloud computing offers the potential of reducing some of heavy upfront financial commitments associated with high performance computing infrastructure, while yielding to faster turnaround times [1]. HPC applications in the cloud can mainly benefit from on-demand elasticity of computing resources, and the pay-per-usage cost model. On the other hand, previous studies have shown that that commodity interconnects and the overhead of virtualization on compute, network and storage performance are major performance barriers to the adoption of cloud for HPC [2,3,4].

Recent efforts towards HPC-optimized clouds, such as Magellan [5] and Amazon's EC2 Cluster Compute [6], are promising steps towards overcoming the performance barriers in the cloud environment. However, these solutions still involve the use of traditional virtualization softwares such as Xen, KVM, etc., which introduce significant performance overheads to HPC applications. In this paper, we present an extensive performance evaluation of a new Cloud technology that offers bare metal servers on demand. The idea of bare metal cloud servers is to give the end users full processing power of the physical server without using a virtual layer. Bare metal cloud servers are single-tenant systems that can be provisioned in minutes, and that allow users to pay by the minute. Subsequently, some of the obstacles of the basic cloud computing approach due to virtualization technology are resolved. First of all, computation intensive applications can request a certain type of server, thus the Service Level Agreement (SLA) is very clear from the providers point. Secondly, users know about the servers they are using and can tune the BIOS and system configurations in order to obtain the highest level of performance. And last but not least, since the server is not shared among multiple tenants, no one can interfere with the performance of the machine [22]. This technology removes the overheads of virtualization, while providing the high availability and elasticity of the cloud.

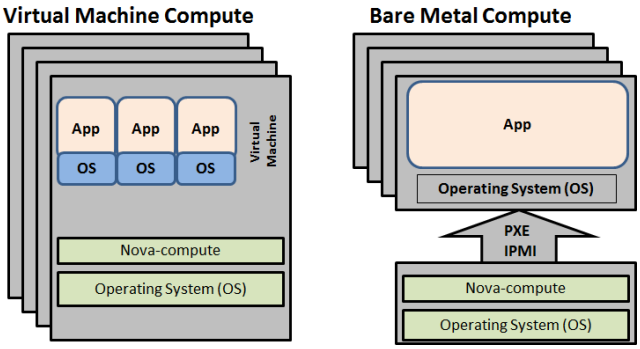

Figure 1. Bare Metal vs. Virtual Machine Compute in Openstack

There are many bare metal resource provisioning frameworks available in market today [23, 24, 25]. These frameworks automate the deployment of operating systems in a datacenter. One of the key challenges of bare-metal provisioning algorithm lies in scheduling the appropriate servers from the datacenter. It is very challenging to optimize the scheduling of heterogeneous resources mainly due to the number of variables involved in making a

decision. In general, it is considered to be a NP hard problem [23]. In this paper, we have evaluated Openstack's new approach of cloud integrated bare-metal provisioning framework plug-in [26]. It is best thought of as a bare metal hypervisor API and a set of plugins which interact with the bare metal hypervisors similar to virtual machines. The main rational behind this approach is to make the cloud Application Programming Interface (API) self-sufficient and enable a single cloud platform that can launch both bare metal and virtual machines. Figure 1 illustrates the difference between bare metal and virtual machine provisioning in a cloud environment managed by Openstack. By default, the bare metal provisioning uses PXE and IPMI in concert to provision and turn on/off machines, but it also supports vendor-specific plugins which may implement additional functionality.

To the best of our knowledge, this is the first paper that evaluates the performance of HPC benchmarks on a bare-metal cloud platform. For performance evaluation, we used UTSA Open Cloud Institute's bare metal cloud servers to run the HPCC (High Performance Computing Challenge) [7] benchmark suite. Bare metal provisioning is enabled by OpenStack Ironic, an integrated OpenStack program which provisions bare metal machines, forked from the Nova baremetal driver. Our results demonstrate excellent scaling performance of bare metal cloud servers with 200 cores.

The remainder of the paper is organized as follows. Sections II and III provide the background, an overview of related work and our approach for cloud-based bare-metal provisioning. Section IV describes our methodology of automating HPC testbed setup in the cloud. Sections V, and VI present a brief introduction to the benchmarks we used and the results of our evaluations. Section VII concludes the paper with directions for future work.

II. RELATED WORK

High performance computing in the cloud has gained significant research attention in recent years [8,9,10,11,18,19,20]. Marathe et al. [10] evaluated the cloud against traditional high performance clusters along turnaround time and cost. Walker [2], followed by several others [3,10,12], conducted the study on HPC in cloud by benchmarking Amazon EC2 [13]. He et al. [14] experimented with three public clouds and compared the results with dedicated HPC systems. These studies show that interconnect latency and virtualization overheads in the cloud environment impose major performance barriers to HPC applications.

In a recent study, Gupta et al. [15] evaluated HPC benchmarks using 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. On the other hand, Containers such as LXC [16] share the physical network interface with its sibling containers and its host. This study showed that thin VM and containers impose a significantly lower communication overhead.

With the advent of state-of-the art cloud technology such as bare metal server provisioning, the performance of HPC applications in the cloud environment is expected to improve further with respect to both computation and communication workloads. Unlike traditional cloud servers, bare metal cloud servers are free from virtualization overheads. However, these emerging cloud platforms have so far not been evaluated extensively for HPC applications.

III. CLOUD ORCHESTRATION WITH BARE METAL CAPABILITY

A. Overview of Bare-Metal Provisioning Frameworks

This section will briefly discuss the bare-metal frameworks prior to our cloud-based bare-metal provisioning.

1) Cobbler is an open source server provisioning system that allows for rapid setup of network installation environments. The cobblerproject was initiated by RedHat and now functions independently [24].
2) Canonical MaaS is an open source bare metal provisioning helps in deploying Ubuntu onto multiple bare-metal machines using (Intelligent Platform Management Interface) IPMI [23].
3) Razor is a bare metal provisioning framework built by Puppet Labs to deploy and configure multiple machines simultaneously [25].
4) Emulab is a network testbed which provides an environment to carry out research in computer networks and distributed systems. To provision bare hardware systems, Emulab takes a user defined network topology in a Network Simulator file and configures the topology.

B. Bare Metal Cloud

Figure 2. shows the architecture of Bare Metal Cloud used in this paper. The main rational behind our approach is to make the cloud scheduler and the cloud Application Programming Interface (API) self-sufficient and make it a single platform that can launch bare metal and virtual machines.

We use cloud based bare-metal provisioning operated by the Open Cloud Institute (OCI) at the University of Texas at San Antonio. It offers significant capacity and similar

design features found in Cloud Computing providers, including robust compute capability and elastic infrastructure design.

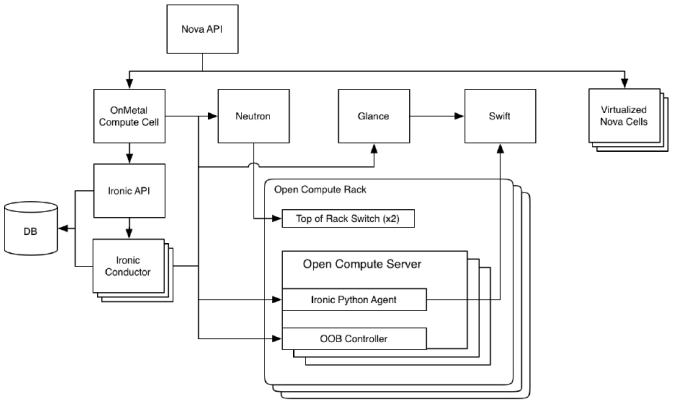

Figure 2. Bare Metal Cloud Architecture

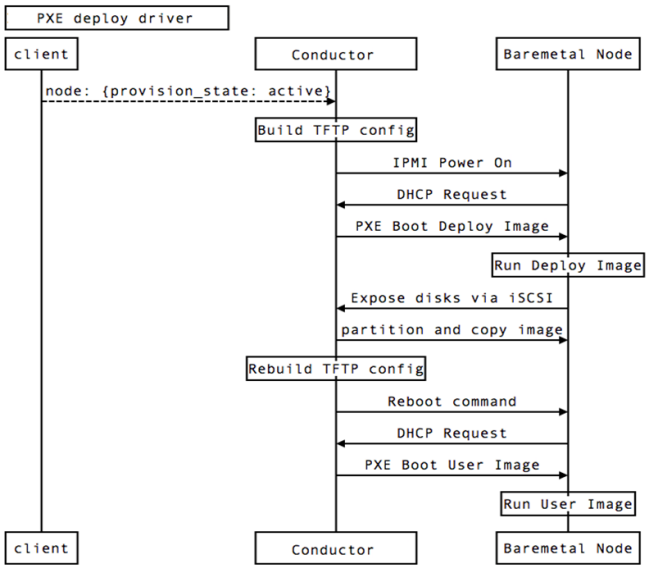

Figure 3. Bare metal provisioning state diagram [28]

As shown in Figure 2, the required steps to boot a bare metal compute node are as follows:

1) Authenticate with keystone
2) Send boot request to Nova API
3) Send boot request to Nova Scheduler to select the host for bare metal deployment
4) Send boot request to the bare metal host
5) Ironic Conductor gets the image from Glance
6) Configure network using Neutron
7) Set deploy request to Ironic API
8) Deploy host – the deployment flow is based on the Ironic driver used.

As an example, the complete deployment flow using the PXE driver is shown in Figure 3.

IV. CLOUD AUTOMATION FOR HPC TESTBED

In order to setup a large scale HPC testbed in the cloud, we developed automation scripts required for the installation, and configuration of OpenMPI and other related software. For this purpose, we used Ansible, an automation engine that automates cloud provisioning, configuration management, application deployment [29]. Ansible allows us to write playbooks and then put a script containing commands to run the playbooks onto the proxy server to distribute the commands to each of the cloud servers as shown in Figure 4.

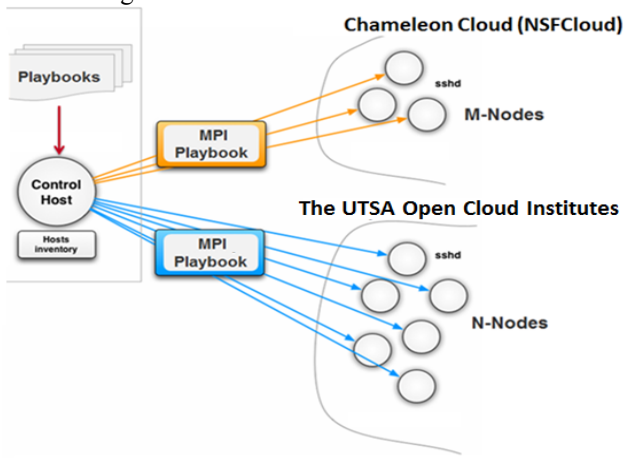

Figure 4. Cloud Automation with Ansible

In future, we plan to run the HPC benchmarking experiments on NSFCloud (Chameleon [30]) using compute nodes with faster processors (intel haswell) and memory (DDR4).
The examples of Ansible playbooks that we developed are as follows.

Playbook for configuring SSH-keys

```
- name: configure SSH on servers
  hosts: servers
  sudo: true
  vars:
  remote_user: root
  tasks:
  - name: Installing required essentials
    apt: name=build-essential state=installed
  - name: Generating SSH Keys
    user: name=root generate_ssh_key=yes
  - name: Obtaining Keys
    fetch: src=~/.ssh/id_rsa.pub dest=~/.ssh/tmp/
recursive=yes
  - name: Copying SSH keys to Machines
    copy: src=~/.ssh/ dest=~/.ssh/ directory_mode
  - name: Adding to the list of authorized_keys
shell:cat~/.ssh/tmp/166.78.164.*/root/.ssh/id_rsa.pub>>
~/.ssh/authorized_keys
```

The first part of the playbook configures the SSH-keys and the communication between machines. We did this by first generating a SSH-key on each machine, then fetched them to the host machine then we copied the contents of each machines public key file to each individual machines authorized_keys file. This ensured that each machine could communicate with each other without a password.

Playbook for installing OpenMPI

```
- name: configure OpenMPI on servers
  hosts: servers
  sudo: true
  vars:
  remote_user: root
  tasks:
   - name: install the required packages for OpenMPI
     action: apt package={{item}} state=installed
     with_items:
     - openmpi-bin
     - openmpi-checkpoint
     - openmpi-common
     - openmpi-doc
     - libopenmpi-dev
```

After SSH-keys are configured, we installed the OpenMPI required packages. We used the apt module in Ansible to make sure those packages were installed.

Playbook for modifying and creating mpi_hosts

```
- name: configure mpi_hosts file on host
  hosts: host
  sudo: true
  vars:
  remote_user: root
  tasks:
   - name: moving inventory to host file
     action: copy src=inventory dest=/root/
   - name: renaming inventory to mpi_hosts
     command: mv inventory mpi_hosts
   - name: editing mpi_hosts file
     command: sed -i '1,2d' mpi_hosts
   - name: continuing editing
     command: sed -i '/[servers]/d' mpi_hosts
```

The final step was to generate a mpi_hosts file (which tells which machines to use). This step was merely copy the inventory file and use sed commands to modify the file. This has allowed for easy configuration of a large number of machines, much faster than a batch script.

V. HPC BENCHMARKS

For performance evaluation, we ran various compute intensive and communication intensive benchmarks from the **HPCC (High Performance Computing Challenge)** benchmark suite. The HPC Challenge benchmark consists at this time of benchmarks such as HPL, Random Access, PTRANS, FFT, DGEMM and Latency Bandwidth. HPL is the Linpack TPP benchmark.

We built the HPCC workload with the ATLAS (Automatically Tuned Linear Algebra Software) math library. ATLAS provides highly optimized Linear Algebra kernels for arbitrary cache-based architectures [17]. ATLAS provides ANSI C and Fortran77 interfaces for the entire BLAS API, and a small portion of the LAPACK API.

HPL
The HPL benchmark measures the ability of a system to deliver fast floating point execution while solving a system of linear equations [32]. Performance of the HPL benchmark is measured in GFLOP/s.

RANDOM ACCESS
The HPC Challenge Random Access benchmark evaluates the rate at which a parallel system can apply updates to randomly indexed entries in a distributed table. Performance of the Random Access benchmark is measured in Giga Updates per second (GUP/s). GUPS (Giga Updates per Second) is a measurement that profiles the memory architecture of a system, and is a measure of performance similar to MFLOPS. The HPCS HPC challenge Random Access benchmark is intended to exercise the GUPS capability of a system, much like the LINPACK benchmark is intended to exercise the MFLOPS capability of a computer. In each case, we would expect these benchmarks to achieve close to the "peak" capability of the memory system. The extent of the similarities between Random Access and LINPACK are limited to both benchmarks attempting to calculate a peak system capability.

PTRANS (parallel matrix transpose)
PTRANS measures the rate of transfer for large arrays of data from multiprocessor's memory. PTRANS exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.

FFT
The HPC Challenge FFT (Fast Fourier Transform) benchmark measures the ability of a system to overlap computation and communication while calculating a very large Discrete Fourier Transform of size m with input vector z and output vector Z [31]. Performance of the FFT benchmark is measured in GFLOP/s. FFT measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).

LATENCY BANDWIDTH
A set of tests to measure latency and bandwidth of a number of simultaneous communication patterns. Latency/Bandwidth measures latency (time required to send an 8-byte message from one node to another) and bandwidth

(message size divided by the time it takes to transmit a 2,000,000 byte message) of network communication using basic MPI routines. The measurement is done during non-simultaneous (ping-pong benchmark) and simultaneous communication (random and natural ring pattern) and therefore it covers two extreme levels of contention (no contention and contention caused by each process communicates with a randomly chosen neighbor in parallel) that might occur in real application.

DGEMM
Measures the floating point rate of execution of double precision real matrix-matrix multiplication.

VI. RESULTS

A. Benchmarking Bare Metal Cloud

The bare metal cloud platform used in this paper was built on a cluster of machines equipped with 10 core Intel® Xeon® E5-2680 v2 2.8Ghz processor, and 32GB RAM. Redundant 10Gbps network connectivity was used to provide high performance access between all nodes.

Table 1. Results of HPL, PTRANS, MPI FFT, and MPI Random Access for Bare Metal Machines

	HPL	PTRANS	MPI FFT	MPI Random Access
cores	Gflops/s	GB/s	Gflops/s	Gup/s
1	1.95E+01	1.923	1.805	0.008665746
20	1.75E+02	4.1364	4.972	0.036206563
40	2.33E+02	5.0882	7.039	0.041778882
80	3.54E+02	6.996	12.06	0.059359527
120	5.81E+02	10.9936	11.655	0.069203492
160	8.03E+02	12.4544	22.004	0.075257348
200	9.49E+02	13.237	22.054	0.079749133

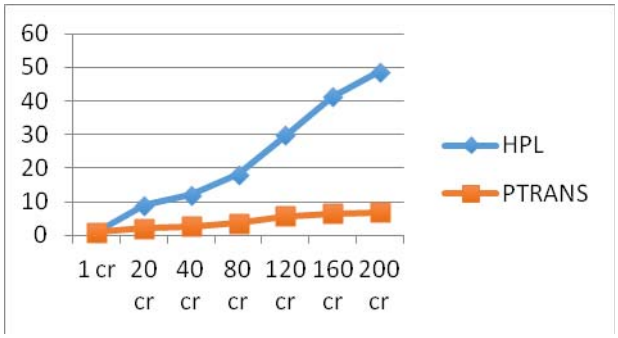

Figure 5. Speedups for HPL and PTRANS for Bare metal machines

As shown in Table 1, the performance of HPL, PTRANS, MPI FFT, and MPI Random Access improve steadily with increasing number of CPU cores used in the bare metal server cluster. The problem size of the benchmarks were determined as follows. For 1 core the dimension N value used is 14000, for 20 cores it is 80000, and for 40, 80, 120, 160 and 200 cores the dimension used is 100000. Figure 5 shows the speedups for HPL and PTRANS benchmarks with increasing number of cores. The results shown in Table 1, and Figure 5 demonstrate that bare metal cloud servers scale well for both compute-intensive and communication-intensive benchmarks. It is mainly due to the fact that the bare-metal servers are single tenant, and they avoid the virtualization overheads by removing the traditional virtualization layer.

The performance results of the remaining HPCC benchmarks are show in Tables 2, 3, and 4. The benchmarks in Table 2 and Table 3 are not parallel. Still, we ran them to see if there is any change in performance by using multiple cores. Table 4 contains bandwidth benchmarks.

Table 2. Results of DEGMM, Random Access, and FFT for single CPU for Bare Metal Machines

	Single DEGMM	Random Access	FFT
cores	Gflops/s	Gup/s	Gflops/s
1	19.728003	0.078908	2.996239
20	10.204755	0.078905	2.0989
40	10.192963	0.078905	2.98469
80	10.259107	0.079012	2.978568
120	10.417615	0.079018	2.965614
160	10.113675	0.053756	1.927949
200	11.421501	0.053756	1.952716

Table 3. Results of Star Random Access and Star FFT for Bare Metal Machines

	Star Random Access	Star FFT
cores	Gups	Gflops
1	0.078729	2.949687
20	0.009831	0.625023
40	0.009828	0.625733
80	0.009839	0.604607
120	0.00984	0.59819
160	0.009867	0.6251
200	0.009867	0.627569

Table 4. Results of Bandwidth for Bare Metal Machines

	Bandwidth (MB/s)		
cores	Ping Pong Min	Natural Ring	Random Ring
20	7936.242195	627.842826	624.633969
40	560.230273	579.754168	215.828013
80	599.44319	553.137582	127.337446
120	581.814954	560.305113	106.077933
160	547.344904	513.213808	98.7737
200	507.554561	524.32077	86.718964

VII. CONCLUSIONS AND FUTURE WORK

Both virtualized and bare metal platforms have different advantages and disadvantages in support of HPC applications. In this paper we evaluated the performance of bare metal cloud for a set of computation and communication benchmarks. Our experimental results show that bare metal cloud system deliver excellent scaling performance. It can provide cloud providers with unprecedented flexibility and capability in building hyper-scale high performance cloud infrastructure required for scientific applications. In future, we plan to set up, and benchmark an HPC cluster with high end nodes from Chameleon Cloud infrastructure (256 GB memory and Haswell Processors). We will also compare the performance of baremetal cloud with equivalent Amazon public cloud servers. Furthermore, we will investigate the performance of storage services combined with computations for the system.

APPENDIX

Shell scripting for installing pre-requisites of ATLAS:

```
#!/bin/sh
sudo apt-get update
sudo apt-get install build-essential -y
sudo apt-get install openmpi-bin openmpi-checkpoint openmpi-common openmpi-doc libopenmpi-dev -y
sudo apt-get update
echo "--- Installing gfortran Compiler ---"
sudo apt-get install gfortran
echo "--- Downloading BLAS ---"
wget http://www.netlib.org/blas/blas.tgz
echo "--- Extracting BLAS ---"
tar -xvzf blas.tgz
cd BLAS
echo "--- BLAS Compilation Started ---"
echo " "
make all
echo " "
echo "--- BLAS Compilation Finished ---"
cd
echo "--- Downloading CBLAS ---"
wget http://www.netlib.org/blas/blast-forum/cblas.tgz
echo "--- Extracting CBLAS ---"
tar -xvzf cblas.tgz
cd CBLAS
echo "#Updating Makefile.in with the appropriate values"
sed -i "/BLLIB =/ s:/.*:$HOME/BLAS/blas_LINUX.a:" Makefile.in
echo "--- Bulding CBLAS ---"
make all
cd
echo "--- Downloading HPCC --- "
wget http://icl.cs.utk.edu/projectsfiles/hpcc/download/hpcc-1.4.3.tar.gz
echo "--- Extracting HPCC ---"
tar -zxvf hpcc-1.4.3.tar.gz
```

ACKNOWLEDGMENT

We gratefully acknowledge the following:
(i) Support by NSF grant CNS-1419165 to the University of Texas at San Antonio; and (ii) time grants to access the Facilities of the Open Cloud Institute of University of Texas at San Antonio.

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] http://icl.cs.utk.edu/hpcc/software/index.html

[8] A. Gupta, O. Sarood, L. V. Kale, and D. Milojicic. Improving hpc application performance in cloud through dynamic load balancing. *In IEEE/ACM Int'L Symposium on Cluster, Cloud and Grid Computing* (CCGrid), 2013.

[9] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema. Performance analysis of cloud computing services for many-tasks scientific computing. *IEEE Transactions on Parallel and Distributed Systems*, 22(6):931–945, 2011.

[10] A. Marathe, R. Harris, D. K. Lowenthal, B. R. de Supinski, B. Rountree, M. Schulz, and X. Yuan. A comparative study of high-performance computing on the cloud. *In Proc. Int'l Symposium on High-performance Parallel and Distributed Computing (HPDC),* 2013.

[11] J. Shafer. I/o virtualization bottlenecks in cloud computing today. In *Proc. Conference on I/O Virtualization*, 2010.

[12] 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.

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

[14] 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.

[15] A. Gupta, L. V. Kale, D. S. Milojicic, P. Faraboschi, R. Kaufmann, ´ V. March, F. Gioachin, C. H. Suen, and B.-S. Lee, “The who, what, why and how of high performance computing applications in the cloud,” in *Proceedings of the 5th IEEE International Conference on Cloud Computing Technology and Science*, ser. CloudCom ’13, 2013.

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

[17] http://math-atlas.sourceforge.net/atlas_install/node6.html

[18] C. A. Lee, “A perspective on scientific cloud computing,” in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC ’10. New York, NY, USA: ACM,2010, pp. 451–459.

[19] A. Thakar and A. Szalay, “Migrating a (large) science database to the cloud,” in *Proceedings of the 19th ACM International Symposiumon High Performance Distributed Computing*, ser. HPDC ’10. New York, NY, USA: ACM, 2010, pp. 430–434. [Online]. Available:http://doi.acm.org/10.1145/1851476.1851539

[20] J. Ekanayake and G. Fox, “High performance parallel computing with clouds and cloud technologies,” *Proceedings of the first International Conference on Cloud Computing*, pp. 20–38, 2010.

[21] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the clouds: A berkeley view of cloud computing,” EECS Department, University of California, Berkeley, *Tech. Rep. UCB/EECS-2009-28*, Feb 2009. [Online]. Available:http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[22] Haider, Aun, Richard Potter, and Akihiro Nakao. "Challenges in resource allocation in network virtualization." *20th ITC Specialist Seminar*. Vol. 18. 2009

[23] Canonical metal-as-a-service. http://maas.ubuntu.com/

[24] Cobbler. http://www.cobblerd.org/

[25] Pubbet lab’s razor. http://puppetlabs.com/solutions/next-generation-provisioning

[26] OpenStack Ironic http://wiki.openstack .org/wiki/Ironic

[27] Paul Rad, Rajendra V. Boppana, Palden Lama, Gilad Berman, and Mo Jamshidi “Low-Latency Software Defined Network for High Performance Clouds”, *the 10th 2015 IEEE International System of Systems Engineering Conference*, May 2015.

[28] P. Querna, https://journal.paul.querna.org/articles/2014/07/02/putting-teeth-in-our-public-cloud/

[29] Ansible. http://www.ansible.com/

[30] NSFCloud Chameleon. https://www.chameleoncloud.org/

[31] Dongarra, J., Luszczek, P. "HPC Challenge: Design, History, and Implementation Highlights,"*Contemporary High Performance Computing: From Petascale Toward Exascale*, Jeffrey Vetter eds. eds. Taylor and Francis, CRC Computational Science Series, Boca Raton, FL, 2013.

[32] A. Petitet and R.C.Whaley and J.Dongara and A.Cleary, “HPL - A Portable Implementation of the High-Performance Linkpack Benchmark for Distributed-Memory Computers, Sept. 10, 2008,” http://www.netlib.org/benchmark/hpl.

[33] Jin, Guohua, et al. "Implementation and performance evaluation of the hpc challenge benchmarks in coarray fortran 2.0." *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*. IEEE, 2011.