Evaluating the Efficacy of the Cloud for Cluster Computation

David Knight, Khawaja Shams, George Chang, Tom Soderstrom
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
{ 818-354-5460, 818-393-0037, 818-393-4979, 818-354-5896 }
{ David.S.Knight, Khawaja.S.Shams, George.W.Chang, Tomas.J.Soderstrom }@jpl.nasa.gov

Abstract—**Computing requirements vary by industry, and it follows that NASA and other research organizations have computing demands that fall outside the mainstream. While cloud computing made rapid inroads for tasks such as powering web applications, performance issues on highly distributed tasks hindered early adoption for scientific computation. One venture to address this problem is Nebula, NASA's homegrown cloud project tasked with delivering science-quality cloud computing resources. However, another industry development is Amazon's high-performance computing (HPC) instances on Elastic Cloud Compute (EC2) that promises improved performance for cluster computation. This paper presents results from a series of benchmarks run on Amazon EC2 and discusses the efficacy of current commercial cloud technology for running scientific applications across a cluster. In particular, a 240-core cluster of cloud instances achieved 2 TFLOPS on High-Performance Linpack (HPL) at 70% of theoretical computational performance. The cluster's local network also demonstrated sub-100 µs inter-process latency with sustained inter-node throughput in excess of 8 Gbps. Beyond HPL, a real-world Hadoop image processing task from NASA's Lunar Mapping and Modeling Project (LMMP) was run on a 29 instance cluster to process lunar and Martian surface images with sizes on the order of tens of gigapixels. These results demonstrate that while not a rival of dedicated supercomputing clusters, commercial cloud technology is now a feasible option for moderately demanding scientific workloads.**

TABLE OF CONTENTS

1. INTRODUCTION 1
2. METHODOLOGY 1
3. SYNTHETIC BENCHMARK RESULTS 2
4. APPLICATION BENCHMARK RESULTS 7
5. CONCLUSIONS 8
REFERENCES 9
BIOGRAPHIES 9

1. INTRODUCTION

While cloud computing made a splash in the consumer web space, certain industries have been relatively hesitant to fully embrace cloud services. The scientific computing community has raised valid concerns about the cloud's ability to perform on large-scale distributed tasks that have long been the domain of dedicated computing clusters. In particular, the constant CPU and network stress imposed by large linear algebra systems has a different usage profile than the more bursty characteristics of web applications.

Early benchmarks of cloud-based clusters revealed lackluster performance largely due to limiting network interconnects and virtual machine overhead [1] [2]. Around this time, NASA's own Nebula cloud project was unveiled in 2009 with an emphasis on increased scientific computing performance and security; areas that were feared ignored by commercial cloud providers [3].

Since then, recent developments in the commercial cloud market have shown a steady investment in making cloud services suitable for scientific computation. In July 2010, Amazon launched their high-performance computing (HPC) Elastic Cloud Compute (EC2) service that provides more powerful instance hardware, reduced virtual machine overhead, and local 10 GigE networking between grouped instances [4]. Using a cluster of these HPC EC2 instances comprised of 7,040 cores, Amazon proceeded to rank in the TOP500 supercomputing list and remains at the number 451 spot as of the June 2011 rankings [5]. Amazon unveiled an upgraded HPC instance type in November 2011 with additional RAM and CPU capacity [6]. Additionally, a third-party vendor acting as a cloud broker started leasing out time on a 30,000 core cluster of EC2 instances at a cost of $1,279 per hour [7].

In light of these developments, a suite of both synthetic and application-level benchmarks were identified for the purpose of testing aspects of cloud computing that are important for running science applications. These benchmarks were then run on Amazon EC2 with HPC instances to test the efficacy of commercial cloud offerings for distributed computing tasks.

2. METHODOLOGY

Naturally, there exists a multitude of different benchmark combinations that could be performed. A chosen set of synthetic benchmarks tests a single machine aspect that could potentially bottleneck distributed applications. Selected application-level benchmarks test a computer cluster as a whole with a representative task. Since application-level benchmarks can be affected by multiple components of a system, results from synthetic benchmarks

978-1-4577-0557-1/12/$26.00 ©2012 IEEE

can inform the results from application-level tests by identifying specific deficiencies.

Synthetic Benchmarks

The synthetic benchmarks fall into three general categories: external network, single instance, and local network. External network benchmarks test the performance of the connection between a user and Amazon's datacenter. Single instance benchmarks test performance of a single component of an instance. Local network benchmarks test the connection between two locally networked instances. Performance aspects being tested and the benchmark tools associated with these aspects are shown in Table 1. Some individual benchmarks are part of the High-Performance Computing Challenge (HPCC), which is described later in this section.

Table 1. Synthetic Benchmark List

Benchmark Type	Benchmark Tool
Internet Throughput	iperf 2.0.5
Internet Latency	BSD traceroute
CPU Cache Throughput	CacheBench (release 5/23/2008)
Mass Storage Throughput	Bonnie++ 1.03e
RAM Throughput	STREAM (HPCC 1.4.1)
Inter-process Latency	MPI Ping-Pong (HPCC 1.4.1)
Inter-process Throughput	MPI Ping-Pong (HPCC 1.4.1)
Inter-node Throughput	iperf 2.0.5

Application Benchmarks

Unlike synthetic benchmarks, application benchmarks stress a machine or cluster as a system and can be negatively impacted by performance bottlenecks in multiple areas. Two application benchmarks were selected: High-Performance Linpack (HPL) and Hadoop image processing.

HPL is a benchmark that measures floating-point computation performance by solving a system of linear equations and is routinely run on supercomputing clusters. Results of HPL are used to rank supercomputers in the TOP500 list. In the case of this work, HPL is run as part of the HPCC.

Hadoop is an open source framework that implements Google's MapReduce algorithm for highly parallel processing. For the benchmark, Hadoop is used to process gigapixel-sized surface images from the both moon and Mars into sets of image tiles. The image processing job taking place is a real-world component of NASA's Lunar Mapping and Modeling Project (LMMP).

High-Performance Computing Challenge (HPCC)

The HPCC is a benchmark suite managed by the University of Tennessee at Knoxville for the purpose of providing an easy to run and consistent method of testing across different computing clusters [8]. Certain synthetic benchmarks and HPL are parts of the HPCC, and results from these tests were gathered by running the challenge as a whole. However, the results of select tests are presented individually.

Hardware Specifications

Tests are conducted using Amazon EC2 cc1.4xlarge HPC instances. These instances use hardware accelerated virtualization to provide lower VM overhead compared to the paravirtualized non-HPC instances. Additionally, a commodity desktop computer is benchmarked as a baseline for the CPU cache throughput, mass storage throughput, and RAM throughput tests to provide context for the cloud instance results. Hardware specifications are shown below in Table 2 and Table 4.

Table 2. Cloud Instance Hardware Details

Instance Type	cc1.4xlarge
CPU	Intel Xeon X5570 (x2)
RAM	23 GB
Network	10 GigE

Table 3. Baseline Hardware Details

Model	Apple iMac 12,2
CPU	Intel Core i7-2600
RAM	8 GB
Network	Gigabit

3. SYNTHETIC BENCHMARK RESULTS

Internet Throughput & Latency

Researchers often have large datasets that need to be transferred to a computing facility for processing. Since cloud computing data centers are physically off-facility for users, cloud providers must support high upload throughput from users' facilities in order to prevent excessive delays when preparing processing jobs.

Tests took place between JPL in Pasadena, CA and Amazon's US West datacenters in San Francisco, CA. Traffic was routed through the CENIC (Corporation for Education Network Initiatives in California) Internet2 connection. TCP parameters were tuned as detailed in Table 4, and iperf 2.0.5 was used to perform data transfer with 10 parallel streams for 10 seconds.

Table 4. TCP Tuning Parameters

Parameter	Value
net.core.rmem_max	8388608
net.core.wmem_max	8388608
net.ipv4.tcp_rmem	4096 87380

	8388608
net.ipv4.tcp_wmem	4096 65536 8388608
net.core.netdev_max_backlog	250000
net.ipv4.tcp_no_metrics_save	1
net.ipv4.tcp_moderate_rcvbuf	1
MTU	9000

With the above configuration, the average bandwidth of the data transfer was *922 Mbps* between JPL and Amazon's US West datacenter. It was likely that the transfer was bottlenecked by the local gigabit Ethernet connection used internally at JPL. This test result indicates that low throughput can possibly be caused by networking within a user's own facilities and is not necessarily restrictions at the cloud.

As seen in the traceroute output in Figure 5, three RTT trials between JPL and Amazon US West resulted in latencies of *15.350, 15.758, and 15.413 ms*.

CPU Cache Throughput

Computational applications that perform large amounts of iteration with tight inner loops are heavily reliant on CPU cache to keep processor pipelines full, allowing a CPU to approach 100% processing load. CacheBench is a single-process program designed to test CPU cache throughput. It is configured to run on a single EC2 instance using a maximum vector length of 2^{33} bytes and on the baseline system with a maximum vector length of 2^{31} bytes. All results are taken from CacheBench's hand-tuned tests (read, write, read-modify-write) that are more robust against varying compiler optimizations.

Table 5. CacheBench EC2 Configuration

Instance Type	cc1.4xlarge
OS	Amazon Linux
Kernel	2.6.35.14-95.38.amzn1.x86_64
Hyper-Threading	Enabled
MTU	1500
C Compiler	gcc 4.4.5

Table 6. CacheBench Baseline Configuration

Model	Apple iMac 12,2
OS	Fedora 16
Kernel	3.1.0-7.fc16.x86_64
Hyper-Threading	Enabled
MTU	1500
C Compiler	gcc 4.6.2

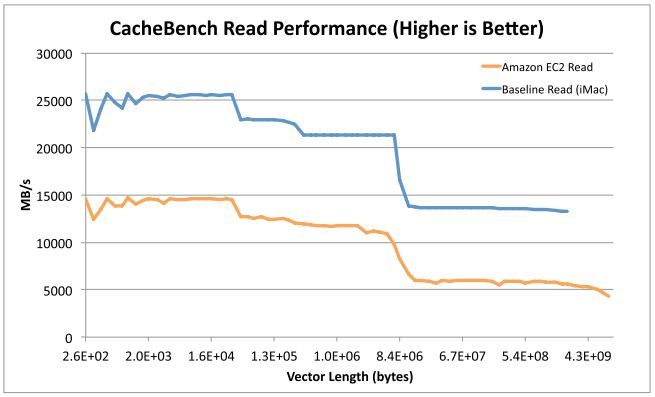

Figure 1 - Performance Curve for CacheBench Read Test

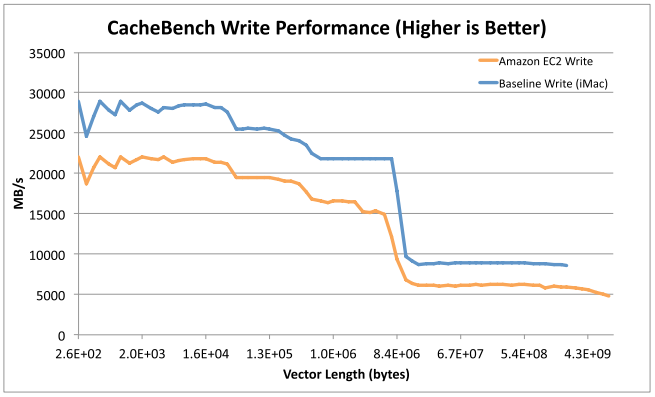

Figure 2 - Performance Curve for CacheBench Write Test

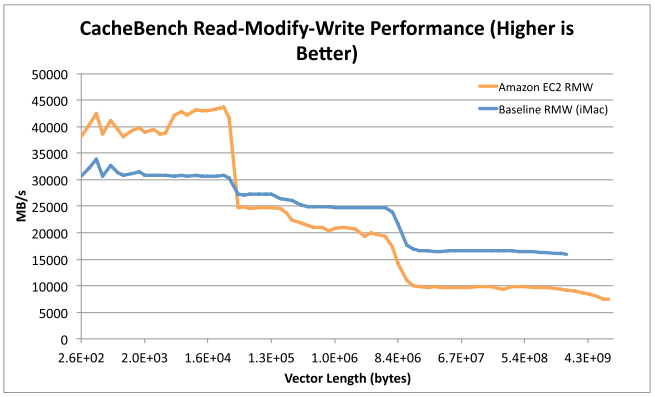

Figure 3 - Performance Curve for CacheBench RMW Test

The newer Intel Core i7-2600 in the baseline system runs at a higher clock frequency (3.4 GHz) than the Intel Xeon X5570 in the Amazon instance (2.93 GHz). Since the benchmark is a single-threaded test, this clock difference is likely responsible for the higher throughput achieved by the iMac in the read and write tests.

Curiously, the higher throughput achieved by the Amazon instance for smaller vector lengths of the RMW test is inconsistent with the rest of the results. This behavior might be due to internal differences between the Linux 2.6.35

kernel used on the Amazon instance and the Linux 3.1.0 kernel used on the baseline system.

Effects of the memory hierarchy on throughput are visible in the results of both systems in Figure 1, Figure 2, and Figure 3. The amounts of L1, L2, and L3 cache are the same in both systems, and throughput drops sharply once the vector length increases beyond data cache boundaries at 32 KB, 256 KB, and 8 MB respectively.

RAM Throughput

RAM has significantly higher throughput and lower latency than mass storage devices, like hard drives, but unlike CPU cache RAM storage is still large enough to hold significant amounts of problem data. Similar to CPU cache, the ability to quickly move data to and from RAM minimizes how much time a processor is kept idle.

The SingleSTREAM section of the HPCC benchmark suite is used to test RAM throughput. Each test of the benchmark reads data from memory and optionally performs an operation on it before writing the value to memory. The copy test performs no floating-point operation on the read data. The scale test multiplies the read data by a constant. The sum test reads two values from memory and adds them together. The triad test reads two values from memory, multiplies one value by a constant, and adds the second value before writing back to memory. Like the name suggests, SingleSTREAM is a single-process benchmark.

Table 7. HPCC/SingleSTREAM Instance Configuration

Instance Type	cc1.4xlarge
OS	CentOS 5.4
Kernel	2.6.18-164.15.1.el5
Hyper-Threading	Disabled
MTU	9000
C/C++ Compiler	icc/icpc 12.1
Fortran Compiler	ifort 12.1
Math Library	Intel MKL
MPI	OpenMPI 1.4.3 (1 process/core)

Table 8. HPCC/SingleSTREAM Baseline Configuration

Model	Apple iMac 12,2
OS	Fedora 16
Kernel	3.1.0-7.fc16.x86_64
Hyper-Threading	Disabled
MTU	1500
C/C++ Compiler	icc/icpc 12.1
Fortran Compiler	ifort 12.1
Math Library	Intel MKL
MPI	OpenMPI 1.4.3 (1 process/core)

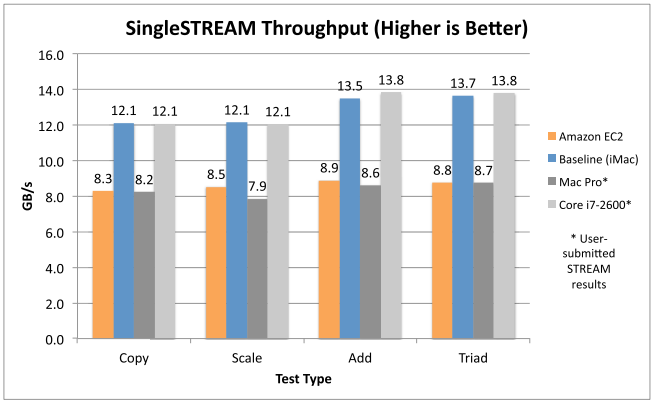

Figure 4 - Plot of SingleSTREAM Throughput Results

On all STREAM tests, the baseline system demonstrates roughly 3.5-4 GBps more throughput. However, it is important to point out that the baseline system's Intel Core i7-2600 uses the newer Sandy Bridge architecture than the Nehalem architecture used by the Amazon instance's Intel Xeon X5570.

For context, two sets of additional STREAM results are included from the STREAM website's user-submission listings: one from a Mac Pro 4,1 and one from a Core i7-2600 system [9] [10]. Despite configuration and OS differences with the user-submission results, the Amazon instance demonstrates comparable throughputs to the Mac Pro that is equipped with a similar Intel Xeon X5550. Likewise, the baseline system shows throughputs similar to the user-submitted Core i7-2600 results. It makes sense that the processor family affects memory throughput due to the fact that both the Nehalem and Sandy Bridge architectures include the integrated memory controller within the CPU.

Considering that the user-submitted STREAM results were obtained without running inside a VM, it demonstrates that the virtualization layer used by Amazon EC2 HPC instances

```
[kssham@sax ~]$ traceroute 50.18.119.228
traceroute to 50.18.119.228 (50.18.119.228), 30 hops max, 40 byte packets
 1  b301-264-edge-v129.jpl.nasa.gov (137.78.29.1)  0.340 ms  0.366 ms  0.421 ms
 2  b171-rtr0-230.jpl.nasa.gov (137.78.12.175)  16.504 ms  16.531 ms  16.561 ms
 3  hpr-riv-hpr--losnettos.cenic.net (137.164.27.193)  20.810 ms  20.838 ms  20.865 ms
 4  lax-hpr2--riv-hpr2-10g.cenic.net (137.164.25.33)  27.768 ms  27.801 ms  27.822 ms
 5  svl-hpr2--lax-hpr2-10g.cenic.net (137.164.25.38)  33.025 ms  33.258 ms  33.375 ms
 6  hpr-amazonpaix--svl-hpr.cenic.net (137.164.26.182)  34.149 ms  32.952 ms  32.990 ms
 7  72.21.222.196 (72.21.222.196)  29.108 ms  22.198 ms  22.138 ms
 8  72.21.222.229 (72.21.222.229)  23.651 ms  20.360 ms  20.530 ms
 9  216.182.236.72 (216.182.236.72)  20.288 ms  15.275 ms  15.242 ms
10  216.182.236.65 (216.182.236.65)  15.702 ms  15.082 ms  15.103 ms
11  216.182.237.85 (216.182.237.85)  15.251 ms  15.308 ms  15.288 ms
12  ec2-50-18-119-228.us-west-1.compute.amazonaws.com (50.18.119.228)  15.350 ms  15.758 ms  15.413 ms
```

Figure 5 - Traceroute Output between JPL and AWS

does not significantly hinder raw memory throughput.

Mass Storage Throughput

Traditional mass storage devices, such as hard drives, are one of the greatest bottlenecks in a computer. Even newer solid state drives have throughputs an order of magnitude slower than RAM. Since scientific computing frequently requires large datasets, slow mass storage access can degrade overall system performance.

Bonnie++ is a program that measures mass storage throughput by writing and reading a series of large files to a device. The mass storage devices themselves were provided through the Elastic Block Store (EBS) service of Amazon Web Services (AWS).

For testing the Amazon instance, a RAID0 array is setup using four 200 GB EBS volumes with mdadm (Linux software RAID). This array is formatted with ext3. Because the physical hardware is abstracted away from the user in a cloud environment, little extra work is needed to provision a higher-throughput RAID array over a single monolithic volume. In order to stress the throughput of the array, Bonnie++ is run in parallel mode with four processes each writing 89 GB of data. This large write size prevents artificial performance inflation from aggressive UNIX I/O caching with RAM.

The baseline system uses a different configuration consisting of a single 776 GB ext3-formatted partition. Bonnie++ is run in parallel mode with four processes each writing 32 GB of data.

Table 9. Bonnie++ Instance Configuration

Instance Type	cc1.4xlarge
OS	Amazon Linux
Kernel	2.6.35.14-95.38.amzn1.x86_64
RAM	23 GB
Hyper-Threading	Enabled
MTU	1500
C Compiler	gcc 4.4.5

Table 10. CacheBench Baseline Configuration

Model	Apple iMac 12,2
OS	Fedora 16
Kernel	3.1.0-7.fc16.x86_64
RAM	8 GB
Hyper-Threading	Enabled
MTU	1500
C Compiler	gcc 4.6.2

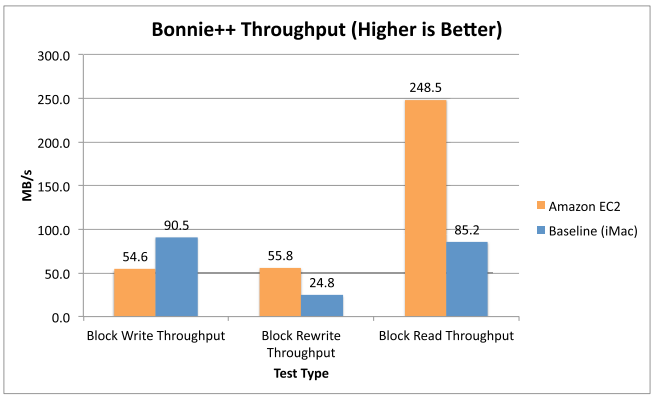

Figure 6 - Plot of Bonnie++ Throughput Results

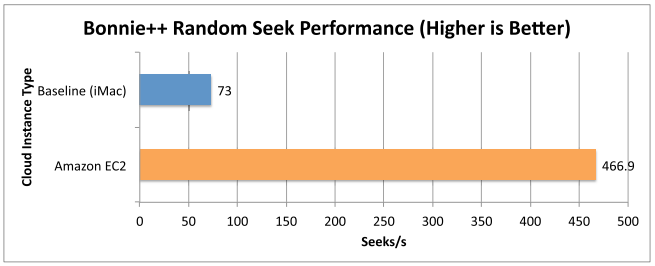

Figure 7 - Plot of Bonnie++ Random Seek Results

The baseline system posts similar read and write throughputs with a significantly lower rewrite throughput. In comparison, the EC2 instance achieves nearly three times as much read throughput as the baseline system but only about 60% of the baseline write throughput. The EC2 instance experiences nearly identical write and rewrite throughputs. Considering that the EC2 instance is equipped with a RAID0 array, the lagging write throughput might indicate a performance bottleneck imposed by virtualization or some other abstraction present in the EBS service.

In the random seek test, the Amazon instance performs over six times as many random seeks per second compared to the baseline system. This performance difference is likely due to higher performing storage devices being used underneath the EBS service.

Inter-Process Latency

One way of thinking about a computing cluster is as a means of operating multiple computers as if they were a single computer. From this perspective, the cluster's local network is similar to a system bus. It is for this reason that the latency between nodes needs to be as low as possible, and the throughput between nodes needs to be as high as possible. Poor network performance slows message passing and data access between nodes.

Latency is measured between Message Passing Interface (MPI) processes by a subtest of the HPCC that performs an MPI ping-pong test between processes using an 8-byte message. The HPCC is run on a 30-node cluster of HPC EC2 instances comprised of 240 cores.

Table 11. HPCC/MPI Ping-Pong Instance Configuration

Instance Type	cc1.4xlarge
OS	CentOS 5.4
Kernel	2.6.18-164.15.1.el5
Hyper-Threading	Disabled
MTU	9000
C/C++ Compiler	icc/icpc 12.1
Fortran Compiler	ifort 12.1
Math Library	Intel MKL
MPI	OpenMPI 1.4.3 (1 process/core)

Table 12. MPI Ping-Pong Latency Results

Maximum RTT	97.066 µs
Average RTT	82.544 µs
Minimum RTT	32.008 µs
Number of Trials	1225

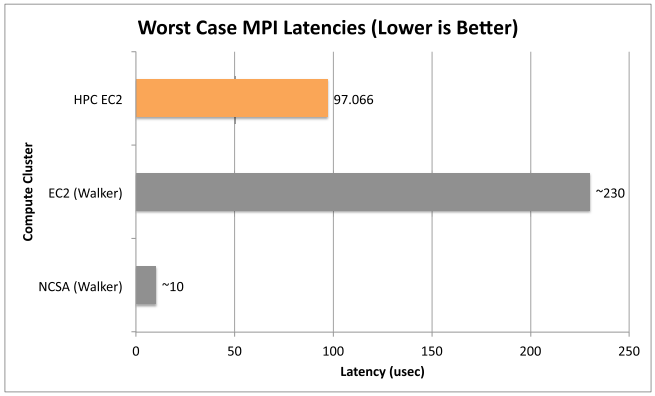

Figure 8 - Latency Comparison with Walker's Tests [1]

Benchmarks performed by Walker in 2008 showed a non-HPC EC2 cloud cluster experiencing worst-case MPI latencies of about 230 µs [1]. These latencies were shown to be consistent between message sizes of 0 to 1 MB. In the same benchmarks, the Abe cluster at the National Center for Supercomputing Applications (NCSA) demonstrated consistent sub-20 µs latency between MPI processes [1].

It should be pointed out that Walker used a different benchmark program, mpptest, and a different distribution of processes across nodes to measure MPI latency. However, it is still valuable to compare worst case latencies that would occur between MPI processes housed on different nodes. While the HPC EC2 cluster tested here exhibits over 100 µs lower worst-case MPI latency compared to the previous non-HPC EC2 cluster, performance still lags behind a dedicated cluster using Infiniband (like the Abe cluster). Higher MPI latencies in the cloud will likely degrade performance when running parallel applications that require large amounts of cross-node data access on a cluster with many nodes.

Inter-process Throughput

Inter-process throughput is measured using the same MPI ping-pong setup as the Inter-process latency benchmark. The primary difference between the two inter-process benchmarks is that a 2 MB message size is used for this throughput test.

Table 13. MPI Ping-Pong Throughput Results

Minimum Throughput	330.631 MBps
Average Throughput	532.414 MBps
Maximum Throughput	2053.515 MBps
Number of Trials	484

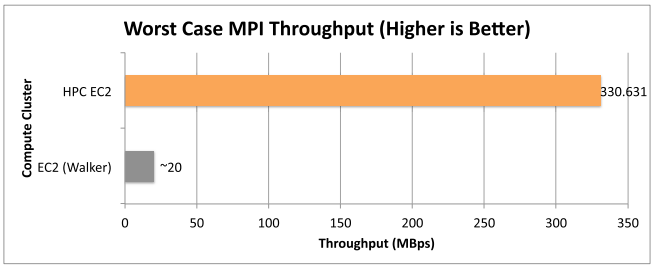

Figure 9 - Throughput Comparison with Walker's Tests [1]

Again, it is valuable to compare the worst-case MPI throughputs between HPC and non-HPC EC2 clusters using results from Walker in 2008. While the maximum message size used by Walker was 1 MB, the non-HPC EC2 cluster asymptotically approached about 20 MBps of inter-process MPI throughput as message size increased [1]. The worst-case throughput experienced on the HPC EC2 cluster was 331 MBps as shown in Table 13. This order of magnitude difference in throughput is likely due to the improved network infrastructure of the HPC EC2 instances (with a promised 10 Gbps of full bisection bandwidth between instances within the same placement group). Both the 10 GigE interconnect and bisection features are not provided for non-HPC instances. The increased MPI throughput between processes will greatly improve the dissemination of problem data.

Inter-node Throughput

Similar to the Internet throughput benchmark, iperf 2.0.5 is used to test throughput between two EC2 nodes within the same placement group. A 1 MB TCP window size is used on both the transmitter and receiver. Tests are conducted using a single stream and 10 parallel streams of data transfer, both with a 10 second transfer duration.

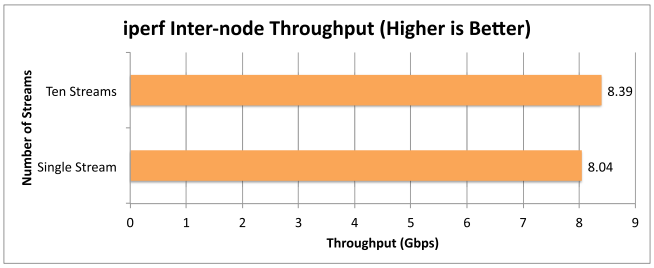

Figure 10 - Inter-node iperf Throughput Results

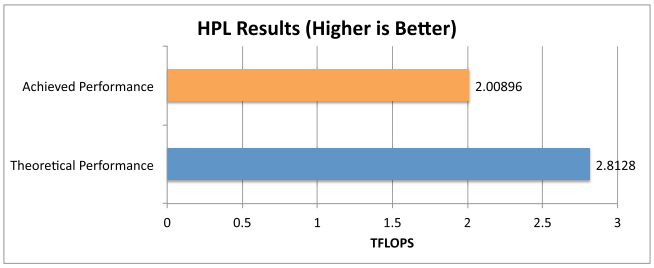

Figure 11 - HPL Computational Performance Results

Even with a single data transfer stream, over 8 Gbps of throughput was achieved between the two instances. Using 10 streams resulted in roughly 5% increased throughput. The small disparity between the results of these two tests indicates that few to no packets were being dropped. The usage of over 80% of the theoretical capacity verifies that instances are connected with 10 Gigabit links.

4. Application Benchmark Results

High Performance Linpack (HPL)

HPL is a benchmark that solves large systems of equations across multiple processes on multiple machines and is a part of the HPCC. MIT's StarCluster 0.92rc2 is used to setup a cluster of 30 EC2 instances composed of 240 cores. The HPL benchmark is run on this cluster using the instance configuration shown in Table 14.

A problem size (*N*) of 221,760 and block size (*NB*) of 168 is used for the HPL test. Values of *P* and *Q* are 10 and 24, respectively. Theoretical cluster performance is calculated by multiplying the maximum number of instructions per cycle (4) by the number of physical cores (240) by core clock frequency (2.93 GHz).

Table 14. HPCC/HPL Instance Configuration

Instance Type	cc1.4xlarge
OS	CentOS 5.4
Kernel	2.6.18-164.15.1.el5
Hyper-Threading	Disabled
MTU	9000
C/C++ Compiler	icc/icpc 12.1
Fortran Compiler	ifort 12.1
Math Library	Intel MKL
MPI	OpenMPI 1.4.3 (1 process/core)

Table 15. HPL Results

HPL_Tflops	2.00896 TFLOPS
HPL_time	3619.03 s
Theoretical Performance	2.8128 TFLOPS
Percent of Peak	71.4%

Results achieved with this 30-node cluster are a significant improvement over earlier HPL tests that used clusters of non-HPC EC2 instances. Due to multiple factors, Napper and Bientinesi achieved sub-40% of peak performance on HPL tests that involved more than one node, and the largest cluster size of 16 nodes achieved less than 10% of peak performance [2]. The Intel-based instances used in these tests had only 7 GB of RAM per node, which limited the HPL problem size. Additionally, it was suspected that slow network interconnects hampered performance as cluster size increased. These insights are consistent with the poor MPI network performance experienced by Walker in 2008 [1]. Most likely, the improved HPL performance witnessed here is attributable to the HPC instances' 10 GigE interconnects and improved hardware specifications. The performance and scalability disparity is too large to be attributed to only OS and library differences.

Hadoop Image Processing

The final benchmark is a real-world application pulled from NASA's Lunar Mapping and Modeling Project (LMMP). LMMP is a dataset collection system that stores lunar surface maps from multiple missions. In order to allow for low-powered computing devices to efficiently view surface maps, full-size surface maps need to be resized to various zoom-levels and subdivided into tiles so that sub-regions can be easily retrieved. This process of resizing and tiling is accomplished using Hadoop.

For benchmark purposes, zoom-level tiles are generated for both a lunar and a Martian surface image. The image of the moon's south pole was captured by NASA's Lunar Reconnaissance Orbiter (LRO) and has a size of 273,052 by 273,052 pixels (~74.5 gigapixels). The Martian image is a mosaic composed of multiple images from the HiRISE project and has dimensions of 162,949 by 224,000 pixels (~36.5 gigapixels). In order to focus the benchmark on the processing and network aspects of the task, distributed storage is setup between RAM disks created on each instance.

Each zoom level of a surface image comprises a single job. The number of zoom levels scales with the size of the original surface image. Processing of higher zoom levels performs less image shrinking and outputs more image tiles. The amount of processing generally scales with zoom level. The highest levels of processing take slightly less time than the second highest levels because at the highest level, image

tiles are only resized and do not need to be passed between nodes.

Table 16. Hadoop Instance Configuration

Instance Type	cc1.4xlarge
OS	Amazon Linux
Kernel	2.6.35.14-95.38.amzn1.x86_64
Hyper-Threading	Enabled
MTU	1500
Hadoop Version	0.20.2
Java Version	Oracle SE 1.6.0_21
Distributed Storage	464 GB (16 GB RAM disk / node)

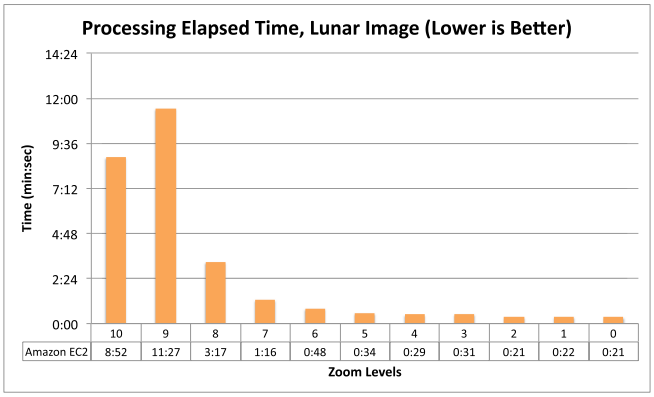

Figure 12 - Elapsed Times for Lunar Image Tiling

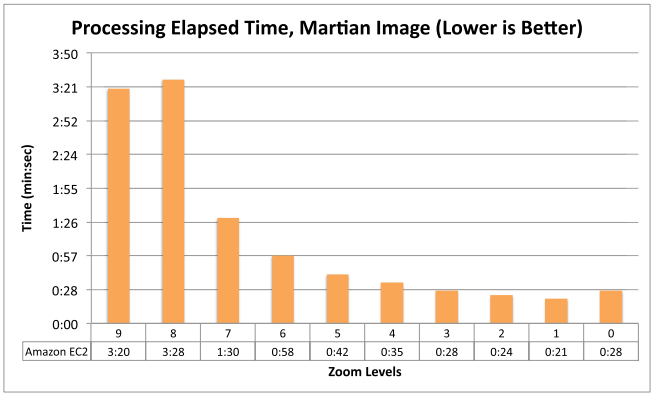

Figure 13 - Elapsed Times for Martian Surface Image Tiling

With fewer than 30 nodes in the cluster, no single job took long than 15 minutes to process. Given the elastic nature of provisioning cloud instances, these processing times could be decreased further by increasing the number of nodes in the cluster.

5. Conclusions

Compared with performance evaluations from a few years ago, the capabilities of commercially available cloud services show promising improvement for running instances in a cluster configuration. Not only has the hardware capacity of instances increased, allowing for more processing to be performed per node, but the local network performance between instances has improved to the point where running real distributed applications across clusters hosted in the cloud is now feasible.

The idea of performing large-scale distributed computation in the cloud is still in its infancy. Vendors will continue to upgrade instance hardware and deploy faster interconnects if demand is present. Additionally, this work only laid out a broad series of benchmarks, and additional work is needed to explore specific system aspects in-depth, such as mass storage I/O, local network performance, and pricing economics.

Even though NASA has supercomputing facilities, like Pleiades at Ames Research Center, NASA has already recognized that there exists a niche for cloud computing services that are highly elastic and quick to provision [3]. As the benchmark results in Section 3 and 4 demonstrated, commercially available cloud services have overcome their initial performance scalability problems. These cloud-based clusters are not meant to compete for the same processing tasks as dedicated supercomputers, but the cloud offers an exciting new tool for scientists to explore modestly sized problems with low barrier to entry.

6. Acknowledgements

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

[1] E. Walker, "Benchmarking Amazon EC2 for high performance scientific computing," *;login: The USENIX Magazine*, vol. 33, no. 5, pp. 18-23, 2008.

[2] J. Napper and P. Bientinesi, "Can Cloud Computing Reach the TOP500?" in Proc. UCHPC-MAW '09, 2009, pp. 17-20.

[3] J. Williams, "NASA Nebula in Action: Cloud Computing Case Examples," *Nebula Cloud Computing Platform*, Aug. 16, 2011. [Online]. Available: http://nebula.nasa.gov/media/uploads/nasa-nebula-in-action.pdf

[4] C. Babcock, "Amazon Launches High Performance Computing Service," *InformationWeek*, July 14, 2010. [Online]. Available: http://www.informationweek.com/news/hardware/utility_ondemand/225800070

[5] TOP500.org, "Amazon EC2 Cluster Compute Instances," *TOP500 Supercomputing Sites*, June 2011. [Online]. Available: http://www.top500.org/system/ranking/10661

[6] Amazon EC2 Team, "Announcing a new Amazon EC2 Cluster Compute Instance (cc2.8xlarge)," *AWS Developer Forums*, November 15, 2011. [Online]. Available: https://forums.aws.amazon.com/ann.jspa?annID=1247

[7] J. Brodkin. "$1,279-per-hour, 30,000-core clutser built on Amazon EC2 cloud," *Ars Technica*, Sept. 21, 2011. [Online]. Available: http://arstechnica.com/business/news/2011/09/30000-core-cluster-built-on-amazon-ec2-cloud.ars

[8] J. Dongarra, "HPC Challenge Benchmark," *Innovative Computing Laboratory: The University of Tennessee*, June 1, 2010. [Online]. Available: http://icl.cs.utk.edu/hpcc/index.html

[9] S. Ludwig, "Re: stream_c results for Mac Pro," *STREAM Mail Archives for 2009*, Oct. 23, 2009. [Online]. Available: http://www.cs.virginia.edu/stream/stream_mail/2009/0009.html

[10] G. Bisseling, "Stream Results," *STREAM Mail Archives for 2011*, June 28, 2011. [Online]. Available: http://www.cs.virginia.edu/stream/stream_mail/2011/0002.html

BIOGRAPHIES

David Knight *is an Application Software Engineer at NASA's Jet Propulsion Laboratory (JPL) where, as a college intern in 2010, he previously developed a WebGL front-end for rendering terrain data from the Lunar Mapping and Modeling Project (LMMP). He received a B.S. in Electrical Engineering from Oregon State University in 2009 and a M.S. in Electrical Engineering from Stanford University in 2011.*

George Chang *is a Senior Software Engineer at NASA's JPL. He developed the surface image tiling approach and corresponding Hadoop implementation for LMMP. He started working at JPL as a college intern, and in his 8 years so far, he has been intimately involved with the data systems, security, and infrastructure of NASA's Deep Space Network (DSN) and LMMP project, in addition to his work for the Mars Exploration Rovers (MER). He constantly tries to infuse new advances in computer technology, such as cloud computing and mobile devices, into his projects. He has a B.S. in Computer Science from Cornell University, a M.S. in Computer Science from Columbia University, and a M.B.A. from UCLA's Anderson School of Management.*

Khawaja Shams *is a member of the Operations Planning Software (OPS) Lab at the NASA JPL. Khawaja develops software that contributes to the operations of a variety of robotic assets including ground, airborne, and waterborne robots, as well as robots on Mars. He leads a variety of software projects, and he serves as the Cognizant Engineer of server side components for the Activity Planning and Sequencing Subsystem (APSS) for the Mars Science Laboratory. Khawaja works closely with the Office of the CIO at JPL to co-lead the efforts to securely deliver the benefits of cloud computing to missions across NASA. He serves as an advisor on the CIO Technology Advisory Board (CTAB) at JPL. Khawaja is currently pursing a Ph.D. in robotics at the University of Southern California, holds a M.S. in Computer Science from Cornell University, and a B.S. in Computer Science from UC San Diego.*

Tom Soderstrom *serves as the IT Chief Technology Officer at NASA's JPL, where his mission is to identify and infuse new IT technologies into JPL's environment. Lately, he has sponsored an effort to bring Cloud technologies to JPL and JPL to the Cloud. He has led remote teams and large scale IT best practices development and change efforts in both small startup and large commercial*

companies, in international venues, and in the US Government arena. Some of the companies has worked for include Telos, enterWorks, User Technology Associates, Digital Island, Exodus, Cable & Wireless, and Raytheon. Tom has been both a frequent producer and consumer of advanced collaboration and engineering tools and practices as his ventures have always included a highly distributed workforce that required advanced collaboration practices.