A Quantitative Analysis of High Performance Computing with Amazon's EC2 Infrastructure: The Death of the Local Cluster?

Zach Hill [#1] and Marty Humphrey [#2]

[#] *Department of Computer Science, University of Virginia*
151 Engineer's Way, P.O. Box 400740, Charlottesville, VA 22904-4740, USA
[1] `zach.hill@email.virginia.edu`
[2] `humphrey@cs.virginia.edu`

Abstract—**The introduction of affordable infrastructure on demand, specifically Amazon's Elastic Compute Cloud (EC2), has had a significant impact in the business IT community and provides reasonable and attractive alternatives to locally-owned infrastructure. For scientific computation however, the viability of EC2 has come into question due to its use of virtualization and network shaping and the performance impacts of both. Several works have shown that EC2 cannot compete with a dedicated HPC cluster utilizing high-performance interconnects, but how does EC2 compare with smaller departmental and lab-sized commodity clusters that are often the primary computational resource for scientists? To answer that question we have run MPI and memory bandwidth benchmarks on EC2 clusters with each of the 64-bit instance types to compare the performance of a 16 node cluster of each to a dedicated locally-owned commodity cluster based on Gigabit Ethernet. Our results show that while EC2 does experience reduced performance, it is still viable for smaller-scale applications.**

I. Introduction

The introduction of virtualized remote on-demand computing resources, Amazon's Elastic Compute Cloud (EC2) [1] in particular, for reasonable prices has made many people question how computing should be delivered in the future. Will it be cheaper to simply lease time on remote resources rather than purchasing and maintaining your own? The emerging infrastructure-provider segment has been generally focused on business users and hosting web applications and services, but some researchers have begun to look at the cloud as a viable solution for scientific computing as well [4][6][7][10][12][14].

Utilizing cloud services for scientific computing opens up a new capability for many scientists: on-demand clusters with no queuing front-end and a nearly-zero cost of entry. This allows scientists to satisfy their high-performance computing needs without resorting to entering their job in a queue with an hour or day long wait and the debugging headaches associated with queue-based systems. While this is certainly an exciting prospect, we must recognize that cloud resources are not equivalent in processing power and network capability to custom-built high-performance systems such as the large clusters at the nation's supercomputing centers. Others have shown that EC2 simply cannot compete in terms of raw network performance with dedicated high-performance clusters, but many scientists do not have access to the supercomputing centers and rely on smaller-scale (tens of nodes) clusters to perform their basic research. Many others, faced with the long queue delays of large HPC system, choose to do primary development, testing, and debugging on smaller clusters before switching to large systems. EC2 resources are particularly attractive for this purpose because of the pay-as-you-go model and complete flexibility in the software stack. Thus, the question remains: Does EC2 provide good enough performance to compete with commodity clusters?

To answer these questions we have quantitatively evaluated EC2's performance for the most common type of scientific application: the MPI program. We have run some benchmarks of the memory systems and networks to compare Amazon's performance to that of our university's IT-managed clusters. The purpose of the benchmarks is to evaluate EC2's specific performance capabilities relevant to a commodity cluster. We do not intend to generalize these results to virtualized environments in general nor to other cloud offerings.

Our results show that EC2 competes reasonably with a commodity, Gigabit Ethernet-based, cluster in terms of memory performance and MPI performance. While we agree that the network performance of EC2 is not equivalent to that of a high-performance dedicated cluster with Myrinet or Infiniband interconnects, in terms of latency in particular, it does provide enough performance to be useful to a large class of scientific users which use the small to mid-sized departmental and lab commodity clusters often found in academia.

The rest of this paper will be organized as follows: the Related Work section will discuss other evaluations of the performance of EC2 and virtualized environments in general; the Evaluation Setup section will describe how the EC2 cluster was setup and configured as well as the configuration of the commodity clusters used as baselines; the Evaluation section will present our results from the various performance benchmarks; the Discussion section will address some observations made during this work as wells problems encountered and possible future work; finally, we give our conclusions on using EC2 clusters for MPI-based HPC computation.

II. Related Work

Edward Walker's work [12] examines the feasibility of using EC2 for HPC, but compares it to high-end compute clusters at NCSA. This comparison pits EC2 against high-end clusters utilizing Infiniband interconnects. His work focuses on network latency and bandwidth in particular and unsurprisingly finds that the NCSA cluster has network latency that is more than an order of magnitude lower than that seen at EC2. We are focusing our evaluation on comparing EC2-based clusters to commodity clusters utilizing Gigabit Ethernet interconnects as would more likely be found in departmental and research-lab sized systems.

C. Evangelinos and C. Hill performed an evaluation of running couple ocean-atmosphere simulations on EC2 [6]. Their work focuses on 32-bit applications and as such they only examine two out of the five EC2 instance types. We have focused primarily on 64-bit platforms as this used for the majority of scientific computing. We also compare the performance of the various EC2 instance types against each other to get a larger picture of EC2's offerings.

Others have evaluated EC2 and its associated blob-storage, the Simple Storage Service (S3) [2], for applicability in data intensive eScience workloads [4][7]. We do not consider the I/O component in our evaluation and instead focus on CPU and network bound workloads based on message passing. We also do not primarily consider the question of cost-effectiveness of cloud-based clusters over locally-owned clusters. While we do feel that a performance-per-dollar comparison is valuable when deciding how to implement infrastructure, we leave that for future work as it can be very application specific and may rely on the economics found in a specific application setting or institution.

General evaluations of the performance of virtualized systems such as Xen [3] have been studied extensively and Xen has been shown to impose negligible overheads in both micro an macro benchmarks [13][14]. However, these were evaluations of Xen itself in a controlled cluster environment whereas we are evaluating Amazon's specific implementation and customizations of Xen and their overall product offering including networking. Amazon's multiplexing of physical resources and networks introduces sources of performance limiters not found in these previous works.

III. Evaluation Setup

Our experimental setup included clusters composed of each EC2 64-bit instance type as well as our local 64-bit cluster. For creating and managing our EC2 clusters we utilized an existing project which provided Python scripts to handle most operations. We describe that project and the specifics of the EC2 instance types below. We also describe the configuration of our local resources that we used to compare the EC2 results against.

A. EC2 Overview

Amazon's EC2 service has become the standard-bearer for Infrastructure-as-a-Service (IaaS) providers. It is the most popular and provides many different service levels. Machine instances are available in the following configurations:

TABLE I
AWS Instance Type Specifications

Type	CPU	Mem (GB)	Disk (GB)	I/O Speed	Cost / Inst-Hr.
M1.Small 32bit, 1 core	1* ECU	1.7	160	Mod	$0.10
M1.Large 64bit, 2 cores	2* ECU	7.5	850	High	$0.40
M1.XLarge 64bit, 4 cores	2* ECU	15	1690	High	$0.80
C1.Medium 32bit, 2 cores	2.5*ECU	1.7	350	High	$0.20
C1.XLarge 64bit, 8 cores	2.5* ECU	7	1690	Mod	$0.80

*1 ECU (EC2 Compute Unit) is equivalent to 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor capacity [6]. ** Indicates Linux pricing only, Windows Server prices are higher.

Amazon utilizes a customized version of Xen to host the AMIs. Instance operating systems available are: Windows Server 2003, Red Hat Linux, Fedora Core, openSUSE, Gentoo, Oracle Enterprise Linux, Ubuntu, and Debian Linux. While each instance has storage associated with it, the local instance storage is not persistent across instance termination, so other solutions, such as Amazons Simple Storage Service (S3) or Elastic Block Store (EBS) are required. However, for this work we did not to utilize either of those services since we are only evaluating the network and processing capabilities of EC2 rather than the storage capabilities.

In terms of service agreement terms, EC2's service agreement states an availability level of 99.95% and Amazon will credit your account if it falls below that during a 365 day period. They also claim an internal network bandwidth of 250 Mbps regardless of instance type, although this is not included in the official instance type specification.

After working to build our own Amazon Machine Images (AMIs) we found a project called ElasticWulf [5], which in addition to providing a pair of AMIs that have multiple MPI libraries (MPICH2, LAM, and OpenMPI) and OpenMP already installed, also included scripts to start, stop, and monitor a cluster of EC2 machines. ElasticWulf requires only the installation of the Amazon command line tools and API libraries. It includes the basics to get a cluster up and running that has a shared NFS directory as well as the MPI runtime, various Python MPI libraries and tools, and Ganglia for monitoring the cluster via the web. The specific AMI's that it uses are based on Fedora Core 6 and are 64-bit. The AMI numbers are: ami-e813f681 and ami-eb13f682 for the master node and the worker nodes respectively. ElasticWulf also works with any AMI, not just the supplied ones by modifying the configuration files to utilize different image files. There are also alternative 32-bit images provided although they do not include the NFS share and Ganglia. Our specific configuration was using MPICH2, *mpicc* version 1.0.6

specifically, on top of Fedora Core 6 and using *gcc* 4.1.2 for the x86_64 architecture.

For our MPI tests we constructed a cluster of each of the instance types which are 64-bit. We feel that while the 32-bit instances may be more cost-effective 64-bit is the standard for scientific computations as well as providing substantially more memory, which is often a large performance enhancer for CPU-bound applications. For the memory bandwidth tests we utilized a single node of each instance type.

B. Local Resources

To give context and a baseline to the EC2 performance numbers we ran the same benchmarks on our locally owned cluster run by the university's IT organization. The Dogwood cluster is composed of nodes with 64-bit Intel Xeon processors has the following specifications: 2 Physical CPUs Intel Xeon EMT 3.00GHz with Intel's "HyperThreading" technology (CPU Family 15, model 4, stepping 3), so the operating system sees four logical CPUs; 2MB L2 cache per CPU, 3GB RAM SDRAM, 800MHz FSB, and Gigabit Ethernet interconnect. The *mpicc* version was 1.0.8p1 and *gcc* was version 4.1.2. The platform was x86_64 64-bit and the MPI library used was MPICH2 1.0.8p1 [10].

The other locally-owned resource we use for comparison is Camillus. Camillus is a 64-bit dual-CPU Intel Xeon E5345 2.4 GHz Quad-Core machine (8 cores total) with shared 8MB L2 cache, a 1333 MHz FSB, 16GB DDR2 RAM, and a Gigabit Ethernet NIC. We use this machine for comparison in the memory-bandwidth benchmarks since it represents typical modern processors with a multiple cores which share caches.

IV. Evaluation

To evaluate the performance of EC2 clusters against that of a small commodity cluster we created clusters of each EC2 instance type and benchmarked their performance using the STREAM memory bandwidth benchmark [3] and then Intel's MPI Benchmark version 3.2[8]. We did not utilize a CPU ALU-op performance benchmark because Xen's CPU performance has been studied extensively and in scientific applications which require a cluster the performance limiter is usually the interconnect or the memory bandwidth on each node since CPU performance has increase much more rapidly than either network or memory performance.

A. Memory Bandwidth

Many scientific applications involve operations on large amounts of data stored in memory. Thus, it is important to evaluate the memory-bandwidth of the EC2 instance types in order to see how they compare to non-virtualized resources. In this case we did not directly compare an EC2 instance's performance to the performance of the same CPU in a non-virtualized environment. Others have examined the performance overhead of virtualization [13][14] on performance, and part of the abstraction of EC2's "Cloud" paradigm is that the underlying resources can change arbitrarily, so no single processor can be designated as the EC2 standard.

The results of running STREAM with array length of 64 million integers follow in Tables II and III.

TABLE II

STREAM Results: Single Thread CPU Memory Bandwidth

Machine Type	1 Thread Bandwidth in GB/s			
	Copy	Scale	Add	Triad
M1.Large	2.058	1.777	1.868	1.725
M1.XLarge	2.551	2.394	2.434	2.178
C1.Medium	2.865	2.852	3.114	3.097
C1.XLarge	2.849	2.840	3.126	3.120
Camillus	2.834	2.830	3.171	3.160
Dogwood	2.493	2.414	2.928	2.923

The performance of the EC2 nodes is similar to Camillus and the Dogwood nodes, which are both modern processors although there is an advantage seen in the processors with higher clock speeds—Dogwood nodes and the high-CPU EC2 instances.

TABLE III

STREAM Results: N threads (cores) CPU Memory Bandwidth.

Machine Spec	N Threads Bandwidth in GB/s				N (Threads)
	Copy	Scale	Add	Triad	
M1.Large	3.244	3.186	3.564	3.508	2
M1.XLarge	3.748	3.936	3.717	3.714	4
C1.Medium	4.241	4.494	4.840	4.796	2
C1.XLarge	4.807	4.788	5.149	5.161	8
Camillus	4.653	4.661	4.895	5.007	8
Dogwood	2.280	2.288	2.617	2.621	4*

**Dogwood is a dual socket Xeon processor with HyperThreading not 4 physical cores. These results are only for comparison.*

It is worth noting here that we are running 64-bit programs, not 32-bit as Evangelinos and Hill did. Thus, there are some discrepancies in the bandwidth numbers.

The Dogwood results for multiple threads are difficult to interpret because of the use of HyperThreading which makes the operating system see each physical CPU as two CPUs. Due to operating system limitations it is not possible to only run the test with two threads and still guarantee that they are run on distinct physical cores. We ran the benchmark using a single thread for each logical CPU to give an accurate comparison with the other platforms.

However, we see that the EC2 nodes perform as well or better than both Camillus and the Dogwood nodes in most of the tests, and we can expect reasonable to good memory bandwidth performance even thought the machines are virtualized.

B. Intel MPI Benchmarks v3.2

The most useful measure of the performance of a cluster is how well it performs on the specific code that it is being used for. Since a large proportion of scientific applications utilize

MPI for inter-process communication, we have tested the various EC2 instance types using the Intel MPI Benchmarks (IMB) version 3.2[14]. We present here the results for the IMB-MPI1 suite of benchmarks which evaluate the MPI v1 specification. We believe that the common operations in most applications are covered in this suite and that the extensions provided in the MPI 2 specification while useful, would not paint a significantly different picture of the relative performance of EC2 instances compared to our local cluster. Thus, we have omitted those results for both brevity and clarity.

Each of the benchmarks presented here measures the average latency for messages passed of a given size. Each data point is an average of multiple runs (1000 for the smaller data points up to 32K and 10 for the 4 MB messages). These averages are reported by the IMB code itself. We do not present the minimums and maximums for sake of clarity.

There are three classes of benchmark: Single Transfer, Parallel Transfer, and Collective. The PingPong and PingPing benchmarks (Figures 1 - 4) compose the Single Transfer class while SendRecv and Exchange (Figures 5 - 8) compose the Parallel Transfer class with the remaining benchmarks (Figures 8 – 16) composing the Collective class. More information about each benchmark including the specific communication patterns in each can be found on the Intel MPI Benchmarks website [14].

Each of the benchmarks was run such that only a single MPI process was run on each node. Thus, even though some EC2 instance types have multiple virtual cores we did not match the number of MPI processes to the number of cores on the virtual machine because we are focusing on the network performance and co-location of MPI processes on nodes would not measure that accurately. Also, we were interested to see if requesting larger node types might reduce the possibility of being co-located with another user's instances and thus reduce or eliminate contention for the I/O system, including the network.

In the following figures we show the results of running each benchmark on a cluster of 16 nodes of each EC2 64-bit instance type as well as our local cluster. The Single Transfer benchmarks (Figures 1 - 4) utilize only 2 nodes in each cluster for their measurements, but all other utilize the full 16 nodes. We conducted runs of the benchmarks using 4 and 8 nodes of each cluster as well but found that the results were not significantly different than for those run with 16 nodes.

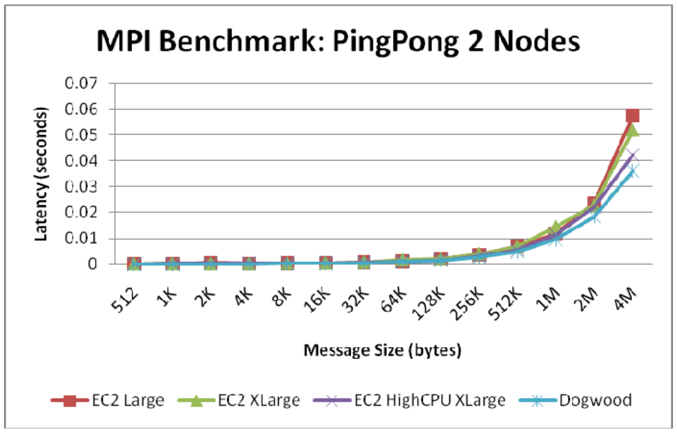

Fig. 1. Average Latency of PingPong

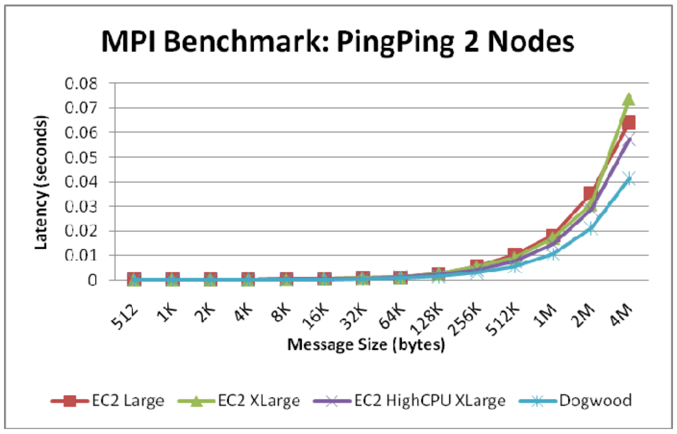

Fig. 2. Average Latency of PingPing

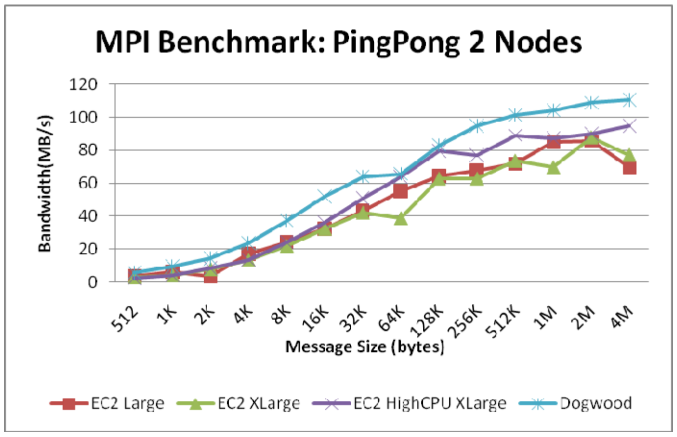

Fig. 3. Bandwidth of PingPong

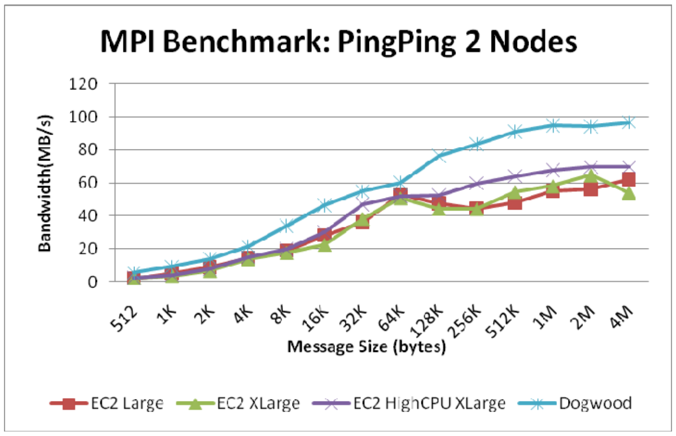

Fig. 4. Bandwidth of PingPing

In the Single Transfer benchmarks (Figures 1 - 4) we see that the EC2 instance type is not a great determiner of performance. The message latency of the EC2 clusters, while not as low as that of the Dogwood cluster, is well below an order of magnitude lower and is generally below 2X higher.

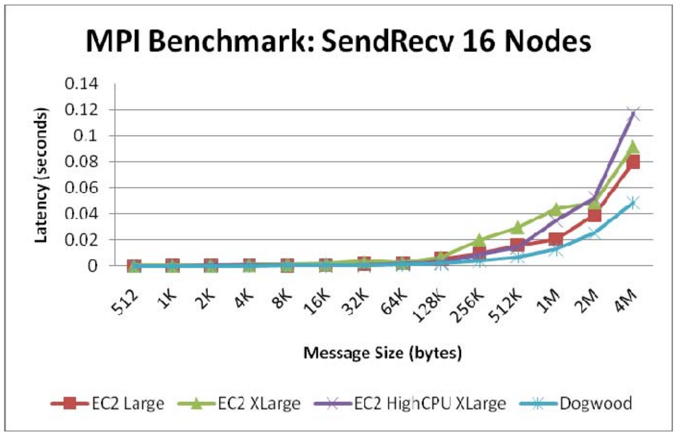

Fig. 5. Average Latency of SendRecv

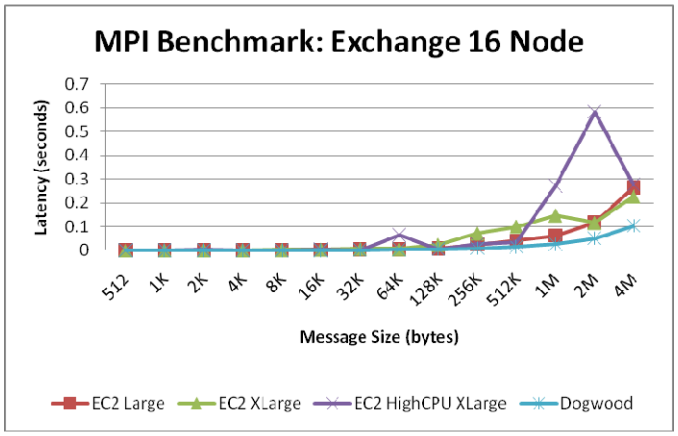

Fig. 6. Average Latency of Exchange

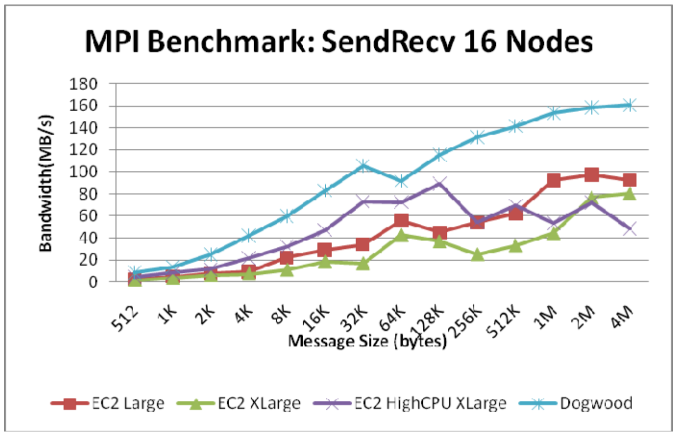

Fig. 7. Bandwidth of SendRecv

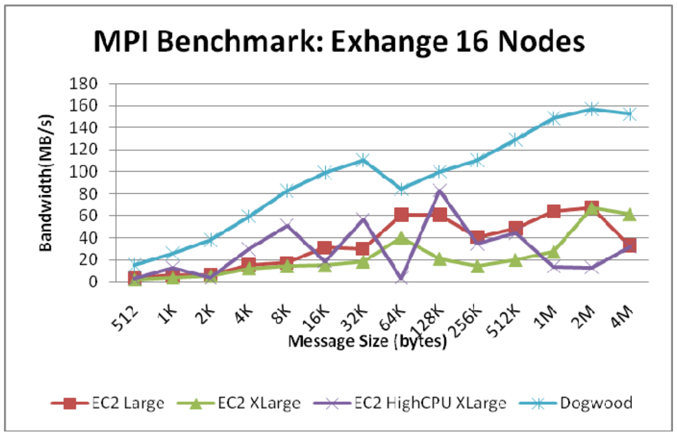

Fig. 8. Bandwidth of Exchange

The Parallel Transfer benchmarks (Figures 3 & 4) show that again all EC2 clusters trail the local cluster, Dogwood, in terms of latency, but not by large margins. The spike seen in Figure 4 at the 2MB message size for the EC2 HighCPU XLarge instance-type cluster is interesting in that we do not expect to have I/O performance isolation issues in this instant type due to the fact that it occupies 8 cores which would fill a dual-CPU quad-core server. Thus, this abnormality could be either due to network conention, most likely, or evidence that performance isolation between hosted VMs is still an issue even when utilizing 8 virtual cores. Note that the results are averaged over 6 runs as opposed to the 20 that are normal used. This indicates that the benchmark experienced difficulty completing all the repititions which most likely indicates network problems, although we are looking into this issue and how it might be related to the specific types of operations which were being conducted. The most difficulties tended to be seen in Gather and Scatter or similar tests.

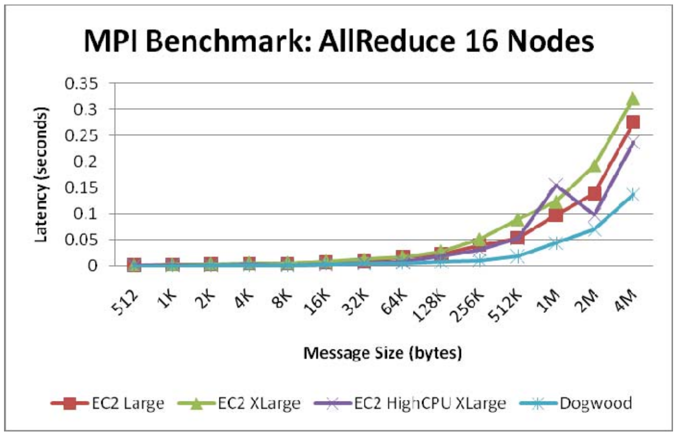

Fig. 9. Average Latency of AllReduce

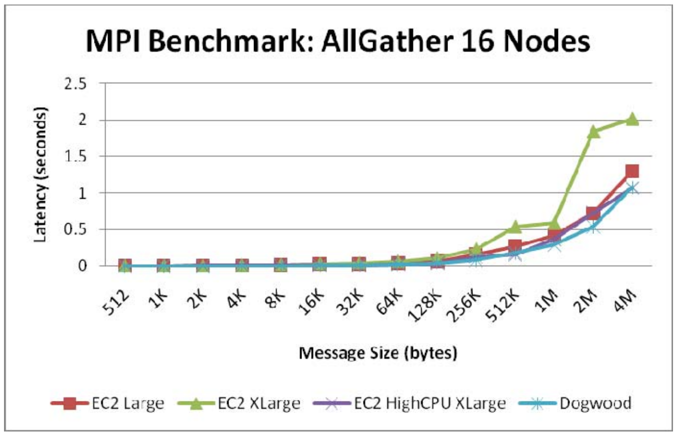

Fig. 12. Average Latency of AllGather

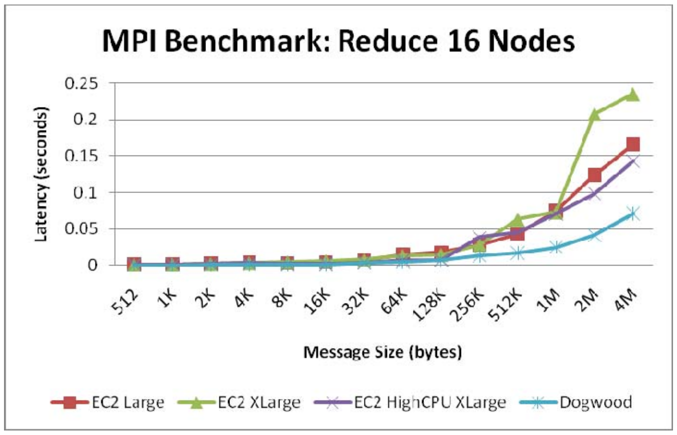

Fig. 10. Average Latency of Reduce

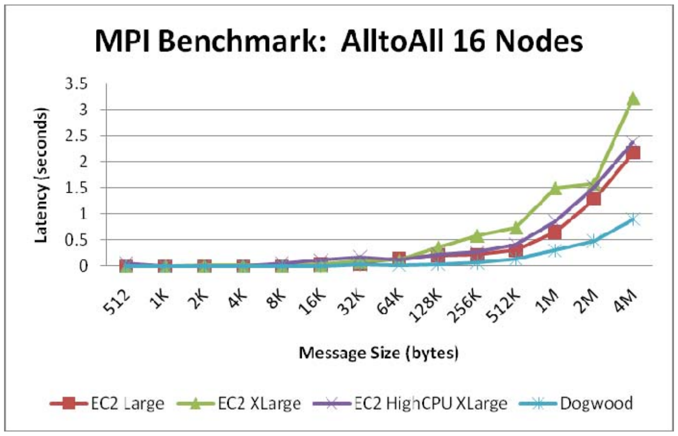

Fig. 13. Average Latency of AlltoAll

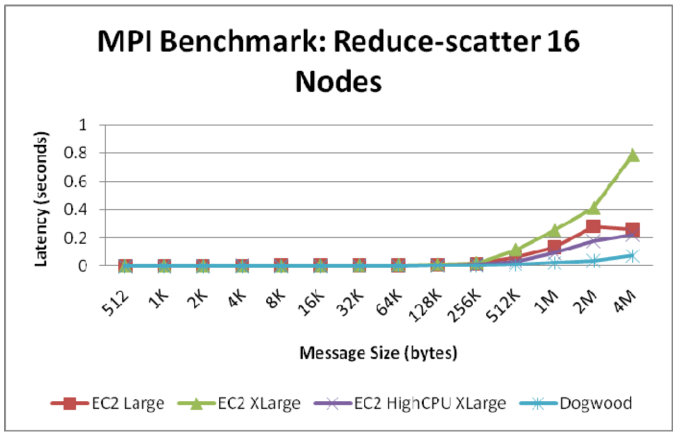

Fig. 11. Average Latency of Reduce-scatter

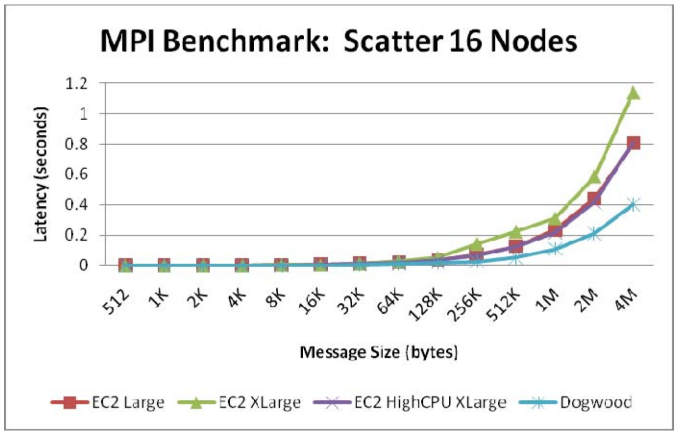

Fig. 14. Average Latency of Scatter

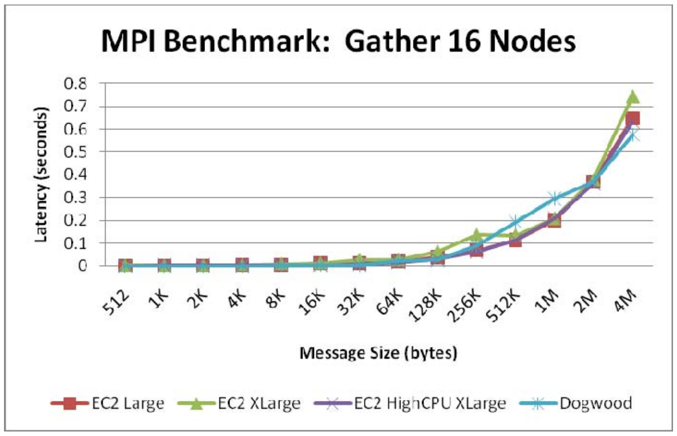

Fig. 15. Average Latency of Gather

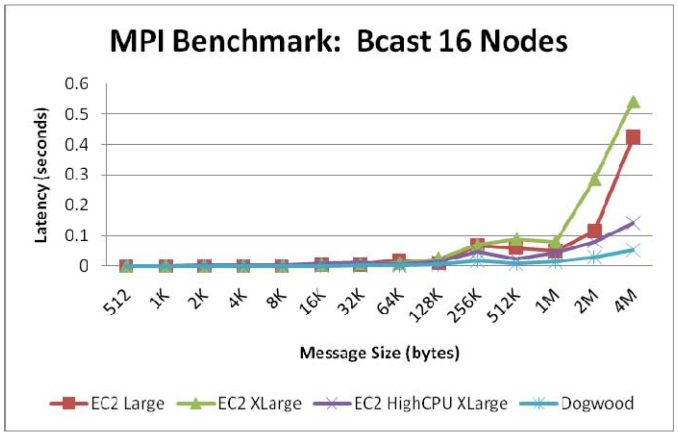

Fig. 16. Average Latency of Bcast

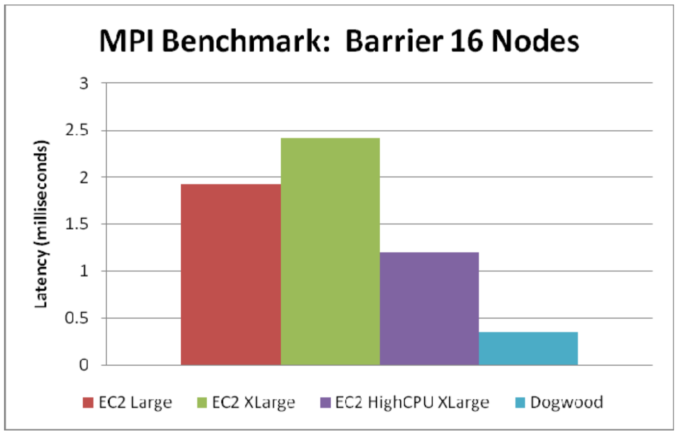

Fig. 17. Average Latency of Barrier

The results for the Collective class of benchmarks generally show that the local Dogwood cluster still has a distinct advantage in network latency performance, and that while the EC2 "High-CPU X-Large" instance type usually has the highest performance of the EC2 clusters it is not a significantly better performer despite its virtual CPUs clock rate advantage. Also of note in all the results is that the "XLarge" EC2 instance type was almost always the worst performer despite having the same virtual clock rate as the "Large" instance type and twice as many virtual cores. While we did not expect the core count to directly impact the MPI results, because we didn't schedule multiple processes on a single node, we did hope to see that requesting more cores would improve performance by reducing the possibility of a co-hosted instance from another user interfering with the I/O of our instances. This did not appear to be the case, however.

V. Discussion

There are several influences on performance which are of concern in virtualized environments such as EC2. These include: cache behavior, buffer-copy costs, and I/O sharing between instances. Cache behavior is critically important to the performance of programs, and good cache management can result in significant performance gains. The difficulty in a virtualized environment is that it is not clear how the caches are shared and whether cache-pollution is possible from other VMs. This is particularly the case for multi-core CPUs with shared L2 caches such as most Intel processors. Since the caches are not explicitly controlled by software (either the application or the operating system) this cannot be controlled by the VMM. Thus, in a pathological case where another VM instance is running on the same physical machine and is using a lot of the shared cache a user may see significantly slower performance than he would on a dedicated machine.

One hypothesis for improved performance is to simply pay for a larger instance in EC2 even if the actual cores are not needed. The larger instances occupy more of the CPUs and cores than the smaller instances do, and thus might reduce the likelihood of co-hosting VMs. For example, the C1.XLarge instance type has 8 cores, which might be a single physical machine with two quad-core processors dedicated solely to that instance. Thus, even if you don't need more cores, by reserving them you may be able to stabilize performance since other VMs will not be hosted on the same physical machine. However, this cannot be guaranteed in the future as physical machines will have more cores and thus co-hosting my again take place even for large instance types. Unfortunately, our results do not show an improvement in performance by using instance-types with more cores.

A similar problem exists with the I/O subsystem of a physical machine. It necessarily must be multiplexed across the instances being hosted. Thus, one VM's I/O performance may affect that of another user. It is not clear whether Amazon has addressed this in their version of the Xen hypervisor, but this would require controlling I/O request routing to the hardware in the host operating system. The same hypothesis for the cache behavior may work for this case as well. By reserving larger instances it may be possible limit the amount of external interference that your VM instance can receive, but this must be evaluated independently from this work.

VI. Conclusions

The emergence of EC2 and other cloud resource hosting platforms has enabled scientists to create clusters of machines on-demand and use them for small to medium scale computational science problems. We showed that while EC2 clusters are not the highest performers, they do provide reasonable performance which when coupled with their low cost and ease of use may provide an attractive alternative to dedicated clusters.

EC2 is not the best platform for tightly-coupled synchronized programs with frequent but small communication between nodes. The high latency reduces performance. However, the reasonable bandwidth available between nodes suggests that less frequent but quite large data exchanges are acceptable and thus redundant computation may be a way to extract extra performance.

In all, EC2 is not a high-performance system which will replace specialized clusters any time soon, but it does offer on-demand capabilities which are very useful for debugging and smaller scale computations without having to deal with queuing systems and their associated wait times. Using pre-existing tools we were able to create and setup a cluster within minutes and using only three shell scripts. This is the beauty of EC2, its configurability and ease of use. We believe it would make a suitable small scale cluster for research groups, labs, and departments.

References

[1] (2009) Amazon EC2 website. [Online]. Available: http://aws.amazon.com/ec2

[2] (2009) Amazon Simple Storage Service website. [Online]. Available: http://aws.amazon.com/s3

[3] P. Barham, B. Dragovic, K. Frasier, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. "Xen and the Art of Virtualization," in *Proc. SOSP'03*, 2003. P. 164-177

[4] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. "The Cost of Doing Science on the Cloud: The Montage Example". in *Conference on High Performance Networking and Computing, Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, 2008, paper No. 50.

[5] (2009) Elasticwulf Project website. [Online]. Available: http://code.google.com/p/elasticwulf

[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 *Proc. CCA'08*, 2008.

[7] G. Hoffa, M. Guarang, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good. "On the Use of Cloud Computing for Scientific Workflows," in *Proc. 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science* (SWBES), of *e-Science'08*, December 2008.

[8] (2009) Intel MPI Benchmarks v3.2 website. [Online]. Available: http://www.intel.com/cd/software/products/asmo-na/eng/219848.htm

[9] (2009) LAM MPI website. [Online]. Available: http://www.lam-mpi.org/

[10] (2009) MPICH2 website. [Online]. Available: http://www.mcs.anl.gov/research/projects/mpich2/

[11] (2009) STREAM Project website. [Online]. Available: http://www.cs.virginia.edu/stream/

[12] E. Walker, "Benchmarking Amazon EC2 for High-Performance Scientific Computing", *;Login: The Usenix Magazine*, Vol. 33, No. 5., 2008

[13] L. Youseff, R. Wolski, B. Gorda, and C. Krintz. "Evaluating the Performance Impact of Xen on MPI and Process Execution in HPC Systems," in *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, 2006, p. 1

[14] L. Youseff, R. Wolski, B. Gorda, and C. Krintz. "Paravirtualization for HPC Systems". Workshop on XEN in HPC Cluster and Grid Computing Environments (XHPC), in *Proc. ISPA'06*, December 2006, LNCS, vol. 4331. p. 474-486.