Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud

Keith R. Jackson
and Lavanya Ramakrishnan
Advanced Computing for Science
Lawrence Berkeley National Lab
Berkeley, CA 94720
KRJackson@lbl.gov
LRamakrishnan@lbl.gov

Krishna Muriki
Information Technology
Lawrence Berkeley National Lab
Berkeley, CA 94720
KMuriki@lbl.gov

Shane Canon, Shreyas Cholia, John Shalf
Harvey J. Wasserman, and Nicholas J. Wright
NERSC
Lawrence Berkeley National Lab
Berkeley, CA 94720
SCanon@lbl.gov, SCholia@lbl.gov, JShalf@lbl.gov
HJWasserman@lbl.gov, and NJWright@lbl.gov

***Abstract*—Cloud computing has seen tremendous growth, particularly for commercial web applications. The on-demand, pay-as-you-go model creates a flexible and cost-effective means to access compute resources. For these reasons, the scientific computing community has shown increasing interest in exploring cloud computing. However, the underlying implementation and performance of clouds are very different from those at traditional supercomputing centers. It is therefore critical to evaluate the performance of HPC applications in today's cloud environments to understand the tradeoffs inherent in migrating to the cloud. This work represents the most comprehensive evaluation to date comparing conventional HPC platforms to Amazon EC2, using real applications representative of the workload at a typical supercomputing center. Overall results indicate that EC2 is six times slower than a typical mid-range Linux cluster, and twenty times slower than a modern HPC system. The interconnect on the EC2 cloud platform severely limits performance and causes significant variability.**

I. Introduction

Cloud computing has emerged as an important paradigm for accessing distributed computing resources. Commercial providers such as Amazon, Rackspace, and Microsoft, all offer environments for developing and deploying applications in the cloud. There are many definitions of cloud computing, but some characteristics exist in most definitions, e.g., virtualized environments and on-demand provisioning of compute resources.

The goal of the recently funded DOE Magellan project is to evaluate the ability of cloud computing to meet DOE's computing needs. The project is evaluating existing commercial cloud offerings and technologies. The purpose of this paper is to examine the performance of existing cloud computing infrastructures and create a mechanism for their quantitative evaluation. Our initial work focuses on Amazon EC2 and its performance, which we believe is representative of current mainstream commercial cloud computing services.

Several groups have reported studies of the applicability of cloud-based environments for scientific computing on Amazon EC2 [1], [2], [3], [4]. Various groups have run both standard benchmark suites such as Linpack and NAS [5], [6], [7], [8], [9], and network performance tests [10].

The goal of the work presented here is to build upon these studies by using the NERSC benchmarking framework [11], [12], [13], [14], [15], [16], [17], [18] to evaluate the performance of real scientific workloads on EC2. This framework contains real scientific applications, that are representative of the whole NERSC workload. It includes a diverse range of numerical methods and data-structure representations in the areas of climate, materials science, fusion, accelerator modeling, astrophysics, and quantum chromodynamics. In addition, we also instrument the runs using the Integrated Performance Monitoring (IPM) [19] framework. This allows us to determine, in a non-pertubative manner, the amount of time an application spends computing and communicating using MPI. This information provides insight into which aspects of the underlying architecture are effecting performance the greatest. Additionally, our approach includes a well-documented method for summarizing achieved, application-level performance based on a simple aggregate measure that expresses useful potential of the systems considered.

Previous work has focused solely on the performance of low-level benchmarks on EC2. They show that tightly coupled applications often exhibit poor performance in such an environment. In this work we focus on evaluating the performance of a suite of benchmarks that represent the workload of a typical HPC center. We believe this is an important differentiator of our efforts from previous ones, as we provide a mechanism for the *quantitative* evaluation of exactly which characteristics of an HPC application are important for determining its performance in a cloud environment.

Specifically, we make the following contributions in this paper,

- We provide the broadest evaluation to date of application performance on virtualized cloud computing platforms.
- We describe our experiences with running on Amazon EC2 and the encountered performance and availability

variations.
- We provide an analysis of the impact of virtualization based on the communication characteristics of the application as seen through IPM.
- We summarize the impact of virtualization through a simple, well-documented aggregate measure that expresses the useful potential of the systems considered.

Section II describes related work, section III describes the methods used in this study, including the machines used and the benchmarks run. Section IV discusses the performance of the benchmarks and compares the IPM profile from the EC2 with that of a science-oriented commodity cluster. Section V describes the tools used, and the impediments encountered in attempting to benchmark EC2 performance and Section VI offers our conclusions.

II. Related Work

A number of different groups have conducted feasibility studies of running their scientific applications in the Amazon cloud. In addition, previous work has examined the performance of individual Amazon AWS components, e.g., the simple storage service (S3) [20].

Hazelhurst examines the performance of the bioinformatics application WCD [1]. The performance and storage costs of running the Montage workflow on EC2 are detailed by Deelman et. al. [2]. The High-Energy and Nulclear Physics (HENP) STAR experiment has examined the costs and challenges associated with running their analysis application on the EC2 cloud [3], [21], [4]. In previous work we examined the usefulness of cloud computing for e-Science applications [22], [23].

Standard benchmarks have also been evaluated on Amazon EC2. Napper et. al. examine the performance of the Linpack benchmarks on different EC2 instance types [5]. The NAS benchmarks have been run by Evangelinos et. al. [6] and Masud [7]. Osterman et al. ran a variety of microbenchmarks and kernels [8]. Rehr. et al. show that Amazon EC2 is a feasible platform for applications that don't need advanced network performance [24]. Wang et al [10] study the impact of virtualization on network performance.

This work is unique in examining the performance of a set of applications that represent the typical workload run at a major supercomputing center. The applications chosen represent both the range of science done and the algorithms typical of supercomputing codes. More importantly, by analyzing the running code using IPM we are able to profile the underlying characteristics of the application, and can quantitatively identify the major performance bottlenecks and resource constraints with respect to the EC2 cloud.

III. Methods

A. Machines Used In Study

All results were obtained during normal, multi-user, production periods on all machines.

1) Carver: is a 400 node IBM iDataPlex cluster located at the National Energy Research Scientific Computing Center (NERSC), which is part of Lawrence Berkeley National Laboratory (LBNL). It has quad-core Intel Nehalem processors running at 2.67 GHz, with dual socket nodes and a single Quad Data Rate (QDR) IB link per node to a network that is locally a fat-tree with a global 2D-mesh. Each node has 24 GB of RAM (3 GB per core). All codes compiled on Carver used version 10.0 of the Portland Group suite and version 1.4.1 of Open MPI.

We note here that the Carver machine that we use as part of our suite of machines here is identical, in hardware terms, to the Magellan cloud testbed at NERSC. We hesitate to call Carver itself a cloud, as these experiments were not performed in a virtual environment, but the performance numbers from Carver do represent the best that Magellan is likely to achieve, given the additional overhead induced by virtualization software, resource sharing etc. that a cloud environment is likely to contain.

2) Franklin: is a 9660 node Cray XT4 supercomputer and is also located at NERSC. Each XT4 compute node contains a single quad-core 2.3 GHz AMD Opteron "Budapest" processor, which is tightly integrated to the XT4 interconnect via a Cray SeaStar-2 ASIC through a 6.4 GB/s bidirectional HyperTransport interface. All the SeaStar routing chips are interconnected in a 3D torus topology, where each node has a direct link to its six nearest neighbors. Each node has 8 GB of RAM (2 GB per core). Codes were compiled with the Pathscale (MAESTRO) and Portland Group version 9.0.4 (all others) compilers.

3) Lawrencium: is a 198-node (1584 core) Linux cluster operated by the Information Technology Division at LBNL. Each compute node is a Dell Poweredge 1950 server equipped with two Intel Xeon quad-core 64 bit, 2.66GHz Harpertown processors, connected to a Dual Data Rate (DDR) Infiniband network configured as a fat tree with a 3:1 blocking factor. Each node contains 16 GB of RAM (2 GB per core). Codes were compiled using Intel 10.0.018 and Open MPI 1.3.3.

4) Amazon EC2: is a virtual computing environment that provides a web services API for launching and managing virtual machine instances. Amazon provides a number of different instance types that have varying performance characteristics. CPU capacity is defined in terms of an abstract Amazon EC2 Compute Unit. One EC2 Compute Unit is approximately equivalent to a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor. For our tests we used the m1.large instances type. The m1.large instance type has four EC2 Compute Units, two virtual cores with two EC2 Compute Units each, and 7.5 GB of memory. The nodes are connected with gigabit ethernet. To ensure consistency, the binaries compiled on Lawrencium were used on EC2.

All of the nodes are located in the US East region in the same availability zone. Amazon offers no guarantees on proximity of nodes allocated together, and there is significant variability in latency between nodes.

In addition to the variability in network latency, we also see

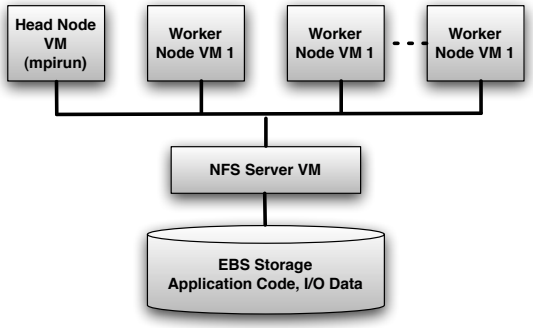


Fig. 1. Virtual Cluster Architecture

variability in the underlying hardware the virtual machines are running on. By examining /proc/cpuinfo we are able to identify the actual CPU type of the un-virtualized hardware. In our test runs, we identified three different CPU's: the Intel Xeon E5430 2.66GHz quad-core processor, the AMD Opteron 270 2.0GHz dual-core processor, and the AMD Opteron 2218 HE 2.6GHz dual-core processor. We have no control over which underlying hardware our virtual machines are instantiated on. Thus, we almost always end up with a virtual cluster running on a heterogeneous set of processors. This hetrogeneneity also meant we were unable to use any of the processor specific compiler options.

Before we could begin our study, we had to address the major differences between the Amazon Web Services environment and that at a typical supercomputing center. For example, almost all HPC applications assume the presence of a shared parallel filesystem between compute nodes, and a head node that can submit MPI jobs to all of the worker nodes. Running these applications in the cloud requires either that the features of a typical HPC environment are replicated in the cloud, or that the application is changed to accommodate the default configuration of the cloud.

For this paper we chose to replicate a typical HPC cluster environment in the cloud by creating virtual clusters [25], [3]. We used a series of Python scripts to configure a file server, a head node, and a series of worker nodes. The head node could submit MPI jobs to all of the worker nodes, and the file server provided a shared filesystem between the nodes. This setup is illustrated in Figure 1.

To implement the shared filesystem, we attached an Amazon Elastic Block Store (EBS) [26] device to the fileserver virtual machine. EBS provides a block level storage volume to EC2 instances that persists independently from the instance lifetimes. On top of the EBS volume we built a standard Linux ext3 file system, that was then exported via NFS to all of the virtual cluster nodes.

B. Applications Used In Study

A supercomputer center such as NERSC typically serves a diverse user community. In NERSC's case the community contains over 3,000 users, 400 distinct projects and is comprised of some 600 codes that serve the diverse science needs of the DOE Office of Science research community. Abstracting the salient performance-critical features of such a workload is a challenging task. However, significant workload characterization efforts have resulted in a set of full application benchmarks that span a range of science domains, parallelization schemes, and concurrencies, as well as machine-based characteristics that influence performance such as message size, memory access pattern, and working set sizes. These applications form the basis for the Sustained System Performance (SSP) metric, which better represents the effectiveness of a system for delivered performance on applications rather than peak FLOP rates [27].

As well as being representative of the DOE Office of Science workload, some of these applications have also been used by other federal agencies as they represent significant parts of their workload. MILC and PARATEC were used by the NSF, CAM by NCAR and GAMESS by the NSF and DoD HPCMO. The representation of the methods embodied in our benchmark suite goes well beyond the particular codes employed. For example, PARATEC is representative of methods that constitute one of the largest consumer of supercomputing cycles in computer centers around the world [11]. Therefore, although our benchmark suite was developed with the NERSC workload in mind, we are confident that it is broadly representative of the workloads of many supercomputing centers today. More details about these applications and their computational characteristics can be found in Ref. [28] (as well as in the references cited here).

The typical problem configurations for these benchmarks are defined for much larger "capability" systems, so we had to construct reduced size problem configurations to target the requirements of mid-range workloads that are the subject of this study. For example, many of the input configurations were constructed for a system acquisition (begun during 2008) that resulted in a 1-PetaFlop peak resource that will contain over 150,000 cores - considerably larger than is possible to run in todays commercial cloud infrastructures. Thus, problem sets were modified to use smaller grids and concomitant concurrencies along with shorter iteration spaces and/or shorter simulation durations. Additionally we also modified the problem configurations to eliminate any significant I/O because I/O performance is beyond the scope of this work.

Next we describe each of the applications that make up our benchmarking suite, describe the parameters they were run with, and comment upon their computation and communication characteristics.

1) CAM: The Community Atmosphere Model (CAM) is the atmospheric component of the Community Climate System Model (CCSM) developed at NCAR and elsewhere for the weather and climate research communities [29], [30]. In this work we use CAM v3.1 with a finite volume (FV) dynamical core and a "D" grid (about 0.5 degree resolution). In this case we used 120 MPI tasks and ran for 3 days simulated time (144 timesteps).

CAM uses a formalism effectively containing two different,

two-dimensional domain decompositions, one for the dynamics that is decomposed over latitude and vertical level and the other for remapping that is decomposed over longitude-latitude. Optimized transposes move data from the program structures between these decompositions. CAM is characterized by relatively low computational intensity that stresses on-node/processor data movement and relatively long MPI messages that stress interconnect point-to-point bandwidth.

2) Gamess: The GAMESS (General Atomic and Molecular Electronic Structure System) code from the Gordon research group at the Department of Energy's Ames Lab at Iowa State University contains various important tools for *ab-initio* quantum chemistry calculations. The benchmark used here calculates the B3LYP DFT energy and gradient for a 43 atom molecule and runs on 64 cores. Gamess is the only benchmark for which no problem size scaling was performed.

GAMESS uses an SPMD approach but includes its own underlying communication library, called the Distributed Data Interface (DDI), to present the abstraction of a global shared memory with one-side data transfers even on systems with physically distributed memory. On the cluster systems included here GAMESS was run using socket communication. On the XT4 an MPI implementation of DDI is used in which only one-half of the processors allocated compute while the other half are essentially data movers. GAMESS is characterized by considerable stride-1 memory access - which stresses memory bandwidth - and interconnect collective performance.

3) GTC: GTC is a fully self-consistent, gyrokinetic 3-D Particle-in-cell (PIC) code with a non-spectral Poisson solver [31] . It uses a grid that follows the field lines as they twist around a toroidal geometry representing a magnetically confined toroidal fusion plasma. The version of GTC used here uses a fixed, 1-D domain decomposition with 64 domains and 64 MPI tasks. The benchmark runs are for 250 timesteps using 10 particles per grid cell (2 million grid points, 20 million particles). Communications at this concurrency are dominated by nearest neighbor exchange that are bandwidth-bound. The most computationally intensive parts of GTC involve gather/deposition of charge on the grid and particle "push" steps. The charge deposition utilizes indirect addressing and therefore stresses random access to memory.

4) IMPACT-T: IMPACT-T (Integrated Map and Particle Accelerator Tracking Time) is an object-oriented Fortran90 code from a suite of computational tools for the prediction and performance enhancement of accelerators. It includes the arbitrary overlap of fields from a comprehensive set of beamline elements, and uses a parallel, relativistic PIC method with a spectral integrated Green function solver. A two-dimensional domain decomposition in the y-z directions is used along with a dynamic load balancing scheme based on domain. Hockneys FFT algorithm is used to solve Poissons equation with open boundary conditions. The problems chosen here are scaled down quite considerably from the official NERSC benchmarks, in terms of number of particles and grid size (4X), 2-D processor configuration (64 cores instead of 256 and 1,024), and number of time steps run (100 instead of 2,000). IMPACT-T performance is typically sensitive to memory bandwidth and MPI collective performance. (Note that although both GTC and IMPACT-T are PIC codes, their performance characteristics are quite different.)

5) MAESTRO: MAESTRO is used for simulating astrophysical flows such as those leading up to ignition in Type Ia supernovae. Its integration scheme is embedded in an adaptive mesh refinement algorithm based on a hierarchical system of rectangular non-overlapping grid patches at multiple levels with different resolution; however, in this benchmark using MAESTRO the grid does not adapt. A multigrid solver is used. Parallelization is via a 3-D domain decomposition in which data and work are apportioned using a coarse-grained distribution strategy to balance the load and minimize communication costs. The MAESTRO communication topology pattern is quite unusual and tends to stress simple topology interconnects. With a very low computational intensity the code stresses memory performance, especially latency; its implicit solver technology stresses global communications; and its message passing utilizes a wide range of message sizes from short to relatively moderate. The problem used was the NERSC-6 "Medium" case (512 X 512 X 1024 grid) on 256 cores but for only 3 timesteps. This problem is more typically benchmarked on 512 cores for 10 timesteps.

6) MILC: This code represents Lattice Computation that is used to study Quantum ChromoDynamics (QCD), the theory of the sub-atomic "strong" interactions responsible for binding quarks into protons and neutrons and holding them together in the nucleus. QCD discretizes space and evaluates field variables on sites and links of a regular hypercube lattice in four-dimensional space time. It involves integrating an equation of motion for hundreds or thousands of time steps that requires inverting a large, sparse matrix at each integration step. The sparse, nearly-singular matrix problem is solved using a conjugate gradient (CG) method and many CG iterations are required for convergence. Within a processor, the four-dimensional nature of the problem requires gathers from widely separated locations in memory. The inversion by CG requires repeated three-dimensional complex matrix-vector multiplications, which reduces to a dot product of three pairs of three-dimensional complex vectors. Each dot product consists of five multiply-add operations and one multiply. The parallel programming model for MILC is a 4-D domain decomposition in which each task exchanges data with its eight nearest neighbors as well as participating in the all-reduce calls with very small payload as part of the CG algorithm. MILC is extremely dependent on memory bandwidth and prefetching and exhibits a high computational intensity.

In this work we use a $32\times32\times16\times18$ global lattice on 64 cores with 2 quark flavors, four trajectories and eight steps per trajectory; this results in over 35,000 CG iterations per run. MILC benchmarking at NERSC uses up to 8,192 cores on a $64^3\times144$ grid with 15 steps per trajectory.

7) Paratec: PARATEC (PARAllel Total Energy Code) performs *ab initio* Density Functional Theory quantum-mechanical total energy calculations using pseudo-potentials, a

TABLE I
HPCC PERFORMANCE

Machine	DGEMM Gflops	STREAM GB/s	Latency μs	Bandwidth GB/s	RandRing Lat. μs	RandRing BW GB/s	HPL Tflops	FFTE Gflops	PTRANS GB/s	RandAccess GUP/s
Carver	10.2	4.4	2.1	3.4	4.7	0.30	0.56	21.99	9.35	0.044
Franklin	8.4	2.30	7.8	1.6	19.6	0.19	0.47	14.24	2.63	0.061
Lawrencium	9.6	0.70	4.1	1.2	153.3	0.12	0.46	9.12	1.34	0.013
EC2	4.6	1.7	145	0.06	2065.2	0.01	0.07	1.09	0.29	0.004

plane wave basis set and an all-band (unconstrained) conjugate gradient (CG) approach. Part of the calculation is carried out in Fourier space; custom parallel three-dimensional FFTs are used to transform the wavefunctions between real and Fourier space.

PARATEC uses MPI and parallelizes over grid points, thereby achieving a fine-grain level of parallelism. The real-space data layout of wave-functions is on a standard Cartesian grid. In general, the speed of the FFT dominates the runtime, since it stresses global communications bandwidth, though mostly point-to-point, using relatively short messages. Optimized system libraries (such Intel MKL or AMD ACML) are used for both BLAS3 and 1-D FFT; this results in high cache reuse and a high percentage of per-processor peak performance.

The benchmark used here is based on the NERSC-5 input that does not allow any aggregation of the transpose data. The input contains 250 Silicon atoms in a diamond lattice configuration and runs for 6 conjugate gradient iterations. More typically NERSC uses a 686-atom system with 20 conjugate gradient iterations run on 1024 cores for benchmarking. A real science run might use 60 or more iterations.

8) HPCC: In addition to the application benchmarks discussed above, we also ran the High Performance Computing Challenge (HPCC) benchmark suite [32]. HPCC consists of seven synthetic benchmarks: three targeted and four complex. The targeted synthetics are DGEMM, STREAM, and two measures of network latency and bandwidth. These are micro-kernels which quantify basic system parameters that separately characterize computation and communication performance. The complex synthetics are HPL, FFTE, PTRANS, and RandomAccess. These combine computation and communication and can be thought of as very simple proxy applications. Taken together these benchmarks allow for the measurement of a variety of lower-level factors that are important for performance, which is why we chose to use them for this work.

C. Evaluation Methodology

At NERSC timing results from these application benchmarks are used to compute the Sustained System Performance (SSP) metric [27], an aggregate measure of the workload-specific *delivered* performance of a computing system. The SSP is derived from an application performance figure, P_i, expressed in units of GFlops per second per core. Given a system configured with N computational cores, the SSP is the geometric mean of P_i over all M applications, multiplied by N, which is the size of the system being considered.

$$SSP = N \left(\prod_{i=1}^{M} P_i \right)^{(1/M)} . \quad (1)$$

The floating-point operation count used in calculating P_i for each of the seven component applications has been predetermined using a hardware performance counter on a single reference system at NERSC, the Cray XT4. The reference counts are combined with the times from other systems to calculate the SSP for those systems.

The SSP is evaluated at discrete points in time and also as an integrated value to give the systems potency, meaning an estimate of how well the system will perform the expected work over some time period. Taken together, the NERSC SSP benchmarks, their derived aggregate measures, and the entire NERSC workload-driven evaluation methodology create a strong connection between science requirements, how the machines are used, and the tests we use.

IV. RESULTS

A. HPC Challenge

The results of running HPCC v.1.4.0 on 64 cores of the four machines in our study are shown in Table I. The DGEMM results are as one would expect based on the properties of the CPUs. The STREAM results show that EC2 is significantly faster for this benchmark than Lawrencium. We believe this is because of the particular processor distribution we received for our EC2 nodes for this test. We had 26 AMD Opteron 270's, 16 AMD Opteron 2218 HE's, and 14 Intel Xeon E5430's, of which this measurement represents an average. The AMD Opteron based systems are known to have better memory performance than the Intel Harpertown-based systems used in Lawrencium. Both EC2 and Lawrencium are significantly slower than the Nehalem-based Carver system, however.

The network latency and bandwidth results clearly show the difference between the interconnects on the tested systems. For display we have chosen both the average ping-pong latency and bandwidth, and the randomly-ordered ring latency and bandwidth. The ping-pong results show the latency and the bandwidth with no self-induced contention, while the randomly ordered ring tests show the performance degradation with self-contention. The uncontended latency and bandwidth measurements of the EC2 gigabit ethernet interconnect are more than 20 times worse than the slowest other machine. Both EC2 and Lawrencium suffer a significant performance degradation when self-contention is introduced. The EC2

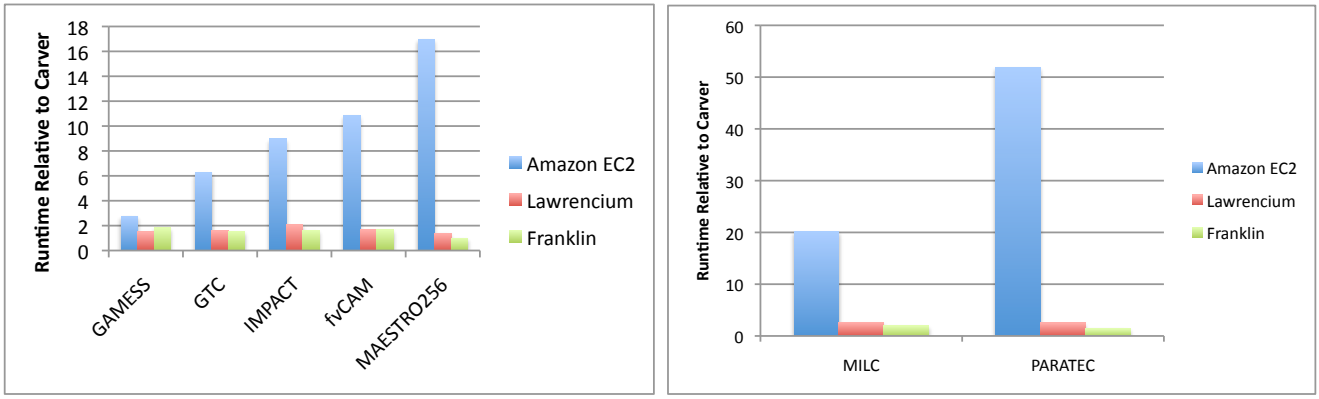


Fig. 2. Runtime of each application on EC2, Lawrencium and Franklin relative to Carver.

latency is 13 times worse than Lawrencium, and more than 400 times slower than a modern system like Carver. The bandwidth numbers show similar trends: EC2 is 12 times slower than Lawrencium, and 30 times slower than Carver.

We now turn our attention to the complex synthetics. The performance of these is sensitive to characteristics of both the processor and the network, and their performance gives us some insight into how real applications may perform on EC2.

HPL is the high-performance version of the widely-reported Linpack benchmark, which is used to determine the TOP500 list. It solves a dense linear system of equations and its performance depends upon DGEMM and the network bandwidth and latency. On a typical high performance computing system today roughly 90% of the time is spent in DGEMM and the results for the three HPC systems illustrate this clearly. However, for EC2 the less capable network clearly inhibits overall HPL performance, by a factor of six or more. The FFTE benchmark measures the floating point rate of execution of a double precision complex one-dimensional discrete Fourier transform, and the PTRANS benchmark measures the time to transpose a large matrix. Both of these benchmarks performance depends upon the memory and network bandwidth and therefore show similar trends. EC2 is approximately 20 times slower than Carver and four times slower than Lawrencium in both cases. The RandomAccess benchmark measures the rate of random updates of memory and its performance depends on memory and network latency. In this case EC2 is approximately 10 times slower than Carver and three times slower than Lawrencium.

Overall the results of the HPCC runs indicate that the lower performing network interconnect in EC2 has a significant impact upon the performance of even very simple application proxies. This is illustrated clearly by the HPL results which are significantly worse than would be expected from simply looking at the DGEMM performance.

B. Applications

Figure 2 shows the relative runtime of each of our test applications relative to Carver, which is the newest, and therefore fastest, machine in our testbed. For these applications, at these concurrencies, Franklin and Lawrencium are between 1.4× and 2.6× slower than Carver. For EC2 the range of performance observed is significantly greater. In the best case, GAMESS, EC2 is only 2.7× slower than Carver. For the worst case, PARATEC, EC2 is more than 50× slower than Carver. This large spread of performance simply reflects the different demands each application places upon the network, as in the case of the compact applications that were described in the previous section. Qualitatively we can understand the differences in terms of the performance characteristics of each of the applications described in Section III-B. PARATEC shows the worst performance on EC2, 52× slower than Carver. It performs 3-DFFT's, and the global (i.e., all-to-all) data transposes within these FFT operations can incur a large communications overhead. MILC (20×) and MAESTRO (17×) also stress global communication, but to a lesser extent than PARATEC. CAM (11×), IMPACT (9×) and GTC (6×) are all characterized by large point-to-point communications, which do not induce quite as much contention as global communication, hence their performance is not impacted quite as much. GAMESS (2.7×), for this benchmark problem, places relatively little demand upon the network, and therefore is hardly slowed down at all on EC2.

Qualitatively, it seems that those applications that perform the most collective communication with the most messages are those that perform the worst on EC2. To gain a more quantitative understanding, we perform a more detailed analysis, which is described in the next section.

C. Performance Analysis Using IPM

To understand more deeply the reasons for the poor performance of EC2 in comparison to the other platforms we performed additional experiments using the Integrated Performance Monitoring (IPM) framework [19], [33]. IPM is a profiling tool that uses the MPI profiling interface to measure the time taken by an application in MPI on a task-by-task basis. This allows us to examine the relative amounts of time taken by an application for computing and communicating, as well as the types of MPI calls made. These measurements

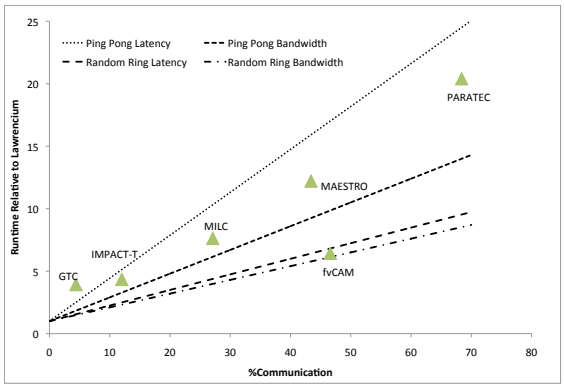


Fig. 3. Correlation between the runtime of each application on EC2 and the amount of time an application spends communicating.

will enable us to determine which particular aspects of the EC2 hardware configuration are most inhibiting performance on an application by application basis in a quantitative manner. Previous measurements with IPM have shown that it has extremely low overhead [33], less that 2%, giving us confidence that by instrumenting the applications with IPM we are not altering their runtime characteristics.

One of the simplest metrics available from IPM is the percentage of the runtime that the application spends communicating (time in the MPI library to be precise). Figure 3 shows the relative runtime on EC2 compared to Lawrencium plotted against the percentage communication for each application as measured on Lawrencium. The overall trend is clear: the greater the fraction of its runtime an application spends communicating, the worse the performance is on EC2.

The principle exception to this trend is fvCAM, where the performance on EC2 is much faster than would be expected from the simple considerations described above. To understand why this is we analyzed the communication characteristics of each of the applications to determine if fvCAM was an anomalous case. To determine these characteristics we classified the MPI calls of the applications into 4 categories: small and large messages (latency vs bandwidth limited) and point-to-point vs collective. (Note for the purposes of this work we classified all messages $<$ 4KB to be latency bound. The overall conclusions shown here contain no significant dependence on this choice.) From this analysis it is clear why fvCAM behaves anomalously; it is the only one of the applications that performs most of its communication via large messages, both point-to-point and collectives. To understand why this causes the performance on EC2 to be faster, consider the HPCC challenge results shown in table I. Compared to Lawrencium the EC2 ping-pong latency is 35$\times$ worse whereas the ping-pong bandwidth is 20$\times$ worse and the random ring latency and bandwidth are 13 & 12$\times$ worse. (Note that to a reasonable approximation, the performance of point-to-point messages will follow the trends of the ping-pong measurements and the performance of collectives will follow those of the random-ring measurements.) Therefore any application that spends a significant amount of its communication performing large messages via point-to-point messages in the latency limit, which in fact is all of the applications here except fvCAM, will be slower, relatively speaking, than one which operates in the bandwidth limit or primarily performs collectives. The slowdown expected for an application that only performs *one* of these four potential modes of communication is also shown in Fig. 3. As is clear from the figure, the faster performance for fvCAM is quite reasonable given its communication pattern.

Thus using quantitative analysis based upon instrumenting our applications with IPM we have explained the reason that each application is slowed down by different amounts when running on EC2.

D. Sustained System Performance

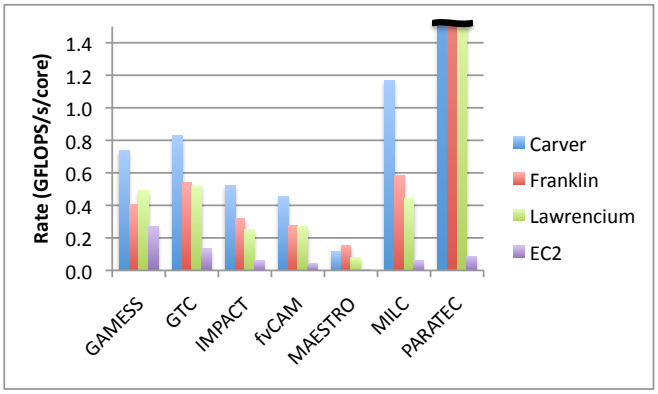
Figure 4a shows the sustained floating point rate per core achieved by each application on each machine. Often this is presented as percentage of peak, but in this case the inhomogeneous CPU environment on EC2 makes that impossible to calculate. The rate achieved by the applications is representative of their computational intensity (ratio of floating point operations to memory operations) as well as the amount of time they spend communicating.

In order to combine these measurements into a single metric that represents the overall performance for this workload we calculate the SSP for these applications on a 3200 core machine. ($N = 3200$ in Equation 1.) We chose 3200 cores as it is the size of Carver, and represents a reasonable mid-size HPC system today. (Ideally, of course, our calculation would be based upon the size of EC2 as that is what we are evaluating here; unfortunately that is not possible.) We note that the particular number of cores used only affects the magnitude of the SSP, not the relative ratios between values for different machines.

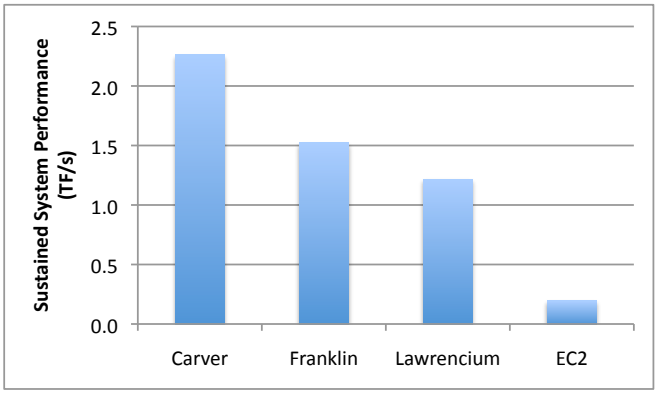
The results of the SSP calculation are shown in Figure 4b. It shows that the sustained performance for Carver is more than 2 TF/s or almost 7% of peak. Franklin, Lawrencium and EC2 are 1.5, 1.2 and 0.2 TF/s respectively. (Note that this value differs from the published one for Franklin because, as described in Section III-B, the problem sizes in this work are drastically reduced.) Thus, for this workload in aggregate, the slowest of our HPC systems, Lawrencium, is six times faster than EC2, and the fastest, Carver is nearly 12 times faster.

V. Discussion

As we saw in the last section, the overall performance of EC2, running this workload, is significantly slower than a typical mid-size cluster like Lawrencium. Our experiments show that this is largely because of the network interconnect available in EC2. This is borne out by the data in Section IV-C, showing that the larger percentage of time an application spends in communication, the worse its overall EC2 performance will be. In addition, the applications communication

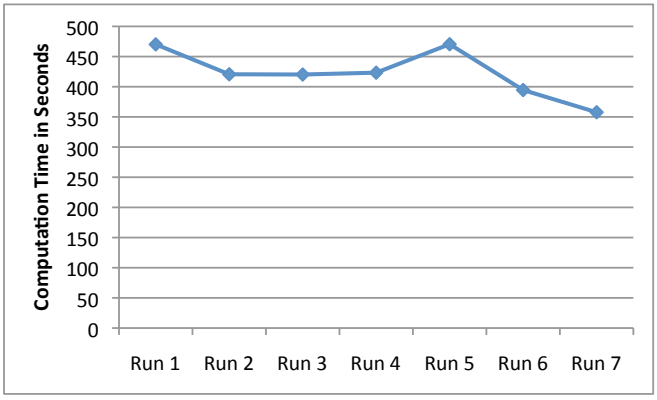


(a) Sustained performance per core for each of the applications on each of the machines. The PARATEC values for Carver, Franklin and Lawrencium are truncated for clarity. They are 4.6, 3.2 and 1.8 Gflop/s/core respectively.

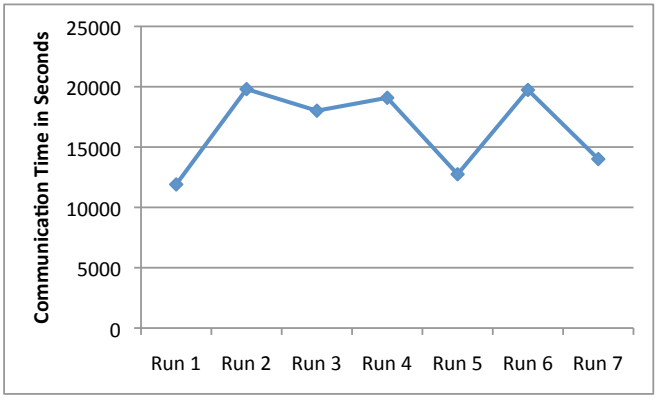


(b) Sustained system performance for Carver, Franklin, Lawrencium and EC2 for the benchmark suite defined in Section III-B.

Fig. 4. Sustained performance a) on a per application and per machine basis b) on a per machine basis only.



(a) PARATEC computation time variability



(b) PARATEC communication time variability

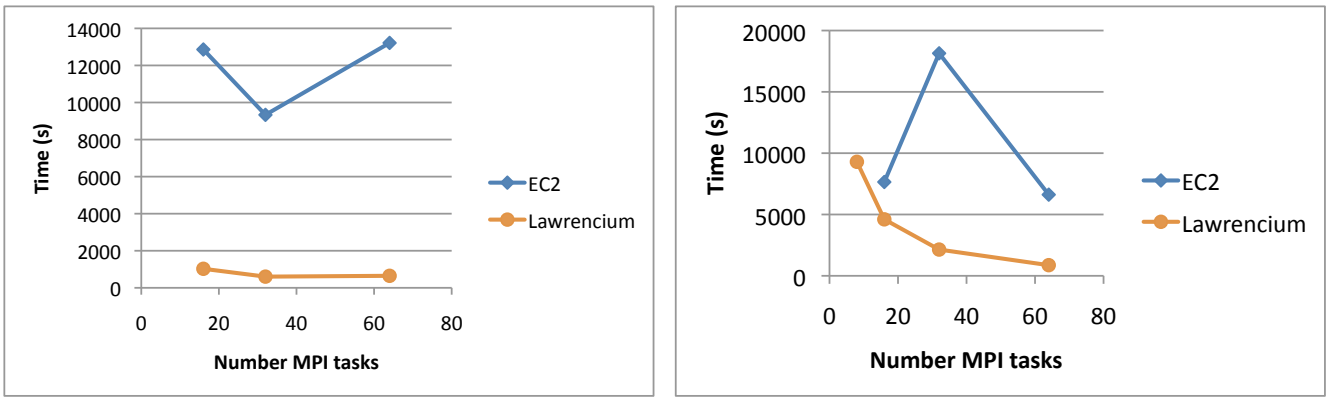
Fig. 5. Paratec performance variability on EC2.

pattern affects how it uses the network interconnect, and will also affect the performance. As seen with PARATEC, all-to-all communications performed to transpose data can severely hamper performance.

While conducting this study, we discovered a significant amount of variability in application performance. One of the sources of this variability in the Amazon cloud environment is the non-homogeneous nature of the systems allocated, as described in Section III-A4. During our testing we saw three different processor types: two AMD Opteron CPUs and one Intel Xeon, with the particular distribution varying from test to test. This heterogeneity makes benchmarking difficult, as it is hard to compare two different runs as a completely different set of processors may be acquired on the next test. In this work our benchmarks are a snapshot of what the performance was with a particular set of resources at a particular time. We note that for application developers this inhomogeneity causes difficulties with performance tuning and load-balancing applications. Another source of variability is introduced by network contention. The switching fabric is shared between all of the EC2 users, and may be heavily contended for.

One last source of variability over which we have no control, nor way of discovering, is if we are sharing the un-virtualized hardware or not. For example, multiple virtual machines may be running on the same physical machine. If one of our nodes is sharing the hardware with another virtual machine that is making extensive use of some hardware component, like the network interface, our performance will suffer.

To study this variability, and understand quantitatively its effect, we conducted seven runs of the PARATEC application using different virtual cluster instances. In Figure 5a we show the time spent in computation for each of these runs as measured using IPM. Overall there is a 30% variability seen in compute time which can be explained by examining the processor distribution acquired for each run. In the first run, which was the slowest, our virtual cluster had 48 of the 2.0GHz AMD 270 processors, and only 2 of the 2.66GHz Intel E5430 processors. On the other hand, run seven spent the least amount of time in computation. For that run we had 18 of the slower AMD 270's, and 26 of the Intel E5430's, and 8 of the 2.66GHz AMD 2218 HE processors. As we anticipated, computation time can vary significantly based on the distribution of processors acquired. In Figure 5b we show the time spent in communication for these runs. As discussed



(a) PARATEC runtime scaling on EC2 and Lawrencium.

(b) MILC runtime scaling on EC2 and Lawrencium

Fig. 6. Paratec and MILC runtime scaling on EC2 and Lawrencium.

in Section IV-B, PARATEC's communication pattern performs particularly poorly on EC2. Communication accounts for approximately 97% of the overall runtime, and accounts for most of the variability in runtime. The difference between the maximum and minimum runtime is 7,900 seconds, or approximately 42% of the mean runtime. This clearly shows the extreme variability in network performance within EC2.

In an attempt to mitigate the effect of the poor performance of the EC2 network (and its variability) we also performed experiments using fewer overall MPI tasks, for the PARATEC and MILC applications on both EC2 and Lawrencium. The results of these are shown in Figures 6a and 6b. The principle observation is that even though we are now using one-half or one-quarter as many nodes the EC2 runtime still shows significant effects due to variability. In fact these are completely dominant, suggesting that unless one is using a loosely coupled application, one is better off running applications exclusively within a single EC2 node. The Lawrencium results show the expected strong-scaling behavior.

We don't explicitly address cost in this work, because it is highly dependent on the specific application and operational requirements of a given collaboration. However there is a direct correlation between cost and performance, and we believe that these performance metrics will prove to be an invaluable tool for computing the true cost of running on the cloud for a given scientific group. This cost will depend on several factors including the size of the application, its need for concurrency, its IO requirements, its fault-tolerance and the general software integration and porting challenges. These are highly site and application dependent.

We also do not address I/O performance in this paper. While we recognize that I/O performance is critical to many HPC applications, we chose to focus this study on computational performance. We expect that future work will examine I/O and Wide Area Network (WAN) performance from a scientific application perspective.

One major lesson of this study was that the mean time between failures (MTBF) of individual nodes in a virtual cluster is significantly higher than in a traditional HPC environment. Traditional MPI based applications are intolerant of node failures and other transient errors. Our experience with the Amazon Web Services environment is that a variety of transient failures can occur, including an inability to access the user-data passed in during image startup, failure to properly configure the network, failure to boot properly, and other performance perturbations, including intermittent virtual machine hangs. While none of these errors occurred frequently, they do in aggregate happen often enough that it becomes a significant barrier to running MPI applications. Approximately one in ten runs would need to be restarted due to some failure.

A common failure that must be handled by the virtual cluster software is resource unavailability. In general, we found that the software creating the virtual cluster cannot assume that it will always acquire all of the requested resources. Allocating 128 or more cores at once is not always practical, and results in indefinite hangs and costly idling of resources if the request cannot be fulfilled. For this reason, we confined our tests to running on no more than 256 cores. Scheduling resources for larger core counts appears to be impractical at this time without moving to higher-cost reservation services, thus severely limiting the ability to run many of the applications common at supercomputing centers.

VI. Conclusions

While cloud computing has proven itself useful for a wide range of e-Science applications, its utility for more tightly-coupled HPC applications has not been proven. In this paper we have quantitatively examined the performance of a set of benchmarks designed to represent a typical HPC workload run on Amazon EC2. Our data clearly shows a strong correlation between the percentage of time an application spends communicating, and its overall performance on EC2. The more communication, the worse the performance becomes. We were also able to see that the communication pattern of the application can have a significant impact on performance. Applications, like PARATEC, with significant global communication perform relatively worse than those with less global communication. Finally we learned that the amount of

variability in EC2 performance can be significant. Variability is introduced by the shared nature of the virtualized environment, by the network, and by differences in the underlying non-virtualized hardware.

VII. Acknowledgements

This work was funded in part by the Advanced Scientific Computing Research (ASCR) in the DOE Office of Science under contract number DE-C02-05CH11231. NJW was supported by the NSF under award OCI-0721397. This research used resources of the National Energy Research Scientific Computing Center, under Contract No. DE-AC02-05CH11231. The authors would like to thank CITRIS, UC Berkeley for Amazon EC2 access.

References

[1] S. Hazelhurst, "Scientific computing using virtual high-performance computing: a case study using the Amazon elastic computing cloud," in *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*. ACM, 2008, pp. 94–103.

[2] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, pp. 1–12.

[3] K. Keahey, R. Figueiredo, J. Fortes, T. Freeman, and M. Tsugawa, "Science clouds: Early experiences in cloud computing for scientific applications," *Cloud Computing and Applications*, vol. 2008, 2008.

[4] K. Keahey, "Cloud Computing for Science," in *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*. Springer-Verlag, 2009, p. 478.

[5] J. Napper and P. Bientinesi, "Can cloud computing reach the top500?" in *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*. ACM, 2009, pp. 17–20.

[6] C. Evangelinos and C. Hill, "Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazons EC2." *ratio*, vol. 2, no. 2.40, pp. 2–34, 2008.

[7] R. Masud, "High Performance Computing with Clouds."

[8] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "An early performance analysis of cloud computing services for scientific computing," *Delft University of Technology, Tech. Rep*, 2008.

[9] E. Walker, "Benchmarking amazon EC2 for high-performance scientific computing," *USENIX Login*, vol. 33, no. 5, pp. 18–23, 2008.

[10] G. Wang and T. E. Ng, "The impact of virtualization on network performance of amazon ec2 data center," in *Proceedings of IEEE INFOCOM*, 2010.

[11] L. Oliker, A. Canning, J. Carter, J. Shalf, and S. Ethier, "Scientific computations on modern parallel vector systems," in *Proc. SC04: International Conference for High Performance Computing, Networking, Storage and Analysis*, Pittsburgh, PA, Nov 6-12, 2004.

[12] L. Oliker, J. Carter, M. Wehner *et al.*, "Leading computational methods on scalar and vector HEC platforms," in *Proc. SC05: International Conference for High Performance Computing, Networking, Storage and Analysis*, Seattle, WA, Nov 12-18, 2005.

[13] J. Carter, L. Oliker, and J. Shalf, "Performance evaluation of scientific applications on modern parallel vector systems," in *VECPAR: High Performance Computing for Computational Science*, Rio de Janeiro, Brazil, July 10-12, 2006.

[14] L. Oliker, A. Canning, J. Carter *et al.*, "Scientific application performance on candidate petascale platforms," in *Proc. IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Long Beach, CA, Mar 26-30, 2007.

[15] T. H. Dunigan Jr., J. S. Vetter, J. B. White III, and P. H. Worley, "Performance evaluation of the Cray X1 distributed shared-memory architecture," *IEEE Micro*, vol. 25(1), pp. 30–40, Jan/Feb 2005.

[16] K. Nakajima, "Three-level hybrid vs. flat MPI on the earth simulator: Parallel iterative solvers for finite-element method," in *Proc. 6th IMACS Symposium Iterative Methods in Scientific Computing*, vol. 6, Denver, CO, Mar 27-30, 2003.

[17] J. Vetter, S. Alam, T. Dunigan, Jr. *et al.*, "Early evaluation of the Cray XT3," in *Proc. IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Rhodes Island, Greece, April 25-29, 2006.

[18] F. Gygi, E. W. Draeger, B. R. de Supinski *et al.*, "Large-scale first-principles molecular dynamics simulations on the BlueGene/L platform using the Qbox code," in *Proc. SC05: International Conference for High Performance Computing, Networking, Storage and Analysis*, Seattle, WA, Nov 12-18, 2005.

[19] D. Skinner, "Integrated Performance Monitoring: A portable profiling infrastructure for parallel applications," in *Proc. ISC2005: International Supercomputing Conference*, Heidelberg, Germany, 2005.

[20] M. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for science grids: a viable solution?" in *Proceedings of the 2008 international workshop on Data-aware distributed computing*. ACM, 2008, pp. 55–64.

[21] K. Keahey, T. Freeman, J. Lauret, and D. Olson, "Virtual workspaces for scientific applications," in *Journal of Physics: Conference Series*, vol. 78. Institute of Physics Publishing, 2007, p. 012038.

[22] L. Ramakrishnan, K. R. Jackson, S. Canon, S. Cholia, and J. Shalf, "Defining Future Platform Requirements for e-Science Clouds," in *Proceedings of the ACM Symposium on Cloud Computing (SoCCi)*. ACM, 2010.

[23] J. Li, D. Agarwal, M. Humphrey, C. van Ingen, K. Jackson, and Y. Ryu, "eScience in the Cloud: A MODIS Satellite Data Reprojection and Reduction Pipeline in the Windows Azure Platform," in *Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2010)*, Atlanta, GA, April 19-23, 2010.

[24] J. Rehr, F. Vila, J. Gardner, L. Svec, and M. Prange, "Scientific computing in the cloud," *Computing in Science and Engineering*, vol. 99, no. PrePrints, 2010.

[25] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, and X. Zhang, "Virtual clusters for grid communities," in *Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid*. Citeseer, 2006, pp. 513–520.

[26] "Amazon Elastic Block Store," http://aws.amazon.com/ebs/.

[27] W. Kramer, J. Shalf, and E. Strohmaier, "The NERSC Sustained System Performance (SSP) Metric," 2005.

[28] K. Antypas, J. M. Shalf, and H. Wasserman, "NERSC-6 workload analysis and benchmark selection process," LBNL, Tech. Rep., 2008.

[29] "CAM3.1," http://www.ccsm.ucar.edu/models/atm-cam/.

[30] "Community Atmopshere Model," http://www.cgd.ucar.edu/csm/models.atm-cam.

[31] W. W. Lee, "Gyrokinetic particle simulation model," *J. Comp. Phys.*, vol. 72, 1987.

[32] "HPCC benchmark web page: http://icl.cs.utk.edu/hpcc/."

[33] N. J. Wright, W. Pfeiffer, and A. Snavely, "Characterizing parallel scaling of scientific applications using IPM," in *The 10th LCI International Conference on High-Performance Clustered Computing*, March 10-12, 2009.