An Evaluation of the Cost and Performance of Scientific Workflows on Amazon EC2

Gideon Juve, Ewa Deelman
USC Information Sciences Institute
{gideon,deelman}@isi.edu

Benjamin P. Berman
USC Epigenome Center
bberman@usc.edu

Bruce Berriman
NASA Exoplanet Science Institute, Infrared Processing and Analysis Center, Caltech
gbb@ipac.caltech.edu

Phil Maechling
Southern California Earthquake Center
maechlin@usc.edu

***Abstract*—Workflows are used to orchestrate data-intensive applications in many different scientific domains. Workflow applications typically communicate data between processing steps using intermediate files. When tasks are distributed, these files are either transferred from one computational node to another, or accessed through a shared storage system. As a result, the efficient management of data is a key factor in achieving good performance for workflow applications in distributed environments. In this paper we investigate some of the ways in which data can be managed for workflows in the cloud. We ran experiments using three typical workflow applications on Amazon's EC2 cloud computing platform. We discuss the various storage and file systems we used, describe the issues and problems we encountered deploying them on EC2, and analyze the resulting performance and cost of the workflows.**

1. Introduction

Computational workflows are an important tool for orchestrating complex simulations and data analyses in many different scientific domains. Traditionally, large-scale workflows, which may contain millions of tasks and represent tens of thousands of CPU hours and terabytes of data, have been run on HPC systems such as clusters and grids. With the development of cloud computing, workflow developers are interested in the benefits this new platform offers for workflow applications. Clouds give workflow developers several advantages over traditional HPC systems including: root access to the operating system, which makes it easier to deploy workflows with legacy components by giving developers control over the entire software environment; the use of VM images to capture and store execution environments, which aids in reproducibility and the collection of provenance; and on-demand provisioning, which helps to reduce time-to-solution and improve resource utilization by enabling the resource pool to be adjusted to the changing needs of the workflow over time.

One important question when evaluating the effectiveness of cloud platforms for workflows is: How can workflows best share data in the cloud? Workflows are loosely-coupled parallel applications that consist of a set of computational tasks linked via data- and control-flow dependencies. Unlike tightly-coupled

applications, such as MPI jobs, in which tasks communicate directly via the network, workflow tasks typically communicate through the use of files. Each task in a workflow produces one or more output files that become input files to other tasks. When tasks are run on different computational nodes, these files are either stored in a shared file system, or transferred from one node to the next by the workflow management system. This makes workflows a good fit for clouds because they are not as dependent upon high-speed, low-latency networks as tightly coupled applications.

There are many existing storage systems that can be deployed in the cloud to provide shared storage for workflows. These include various network and parallel file systems, object-based storage systems, and databases. One of the advantages of cloud computing and virtualization is that the user has control over what software is deployed, and how it is configured. However, this flexibility also imposes a burden on the user to determine what system software is appropriate for their application. The goal of this paper is to explore the various options for managing data in the cloud for workflow applications, and to evaluate the effectiveness of the various solutions in terms of cost and performance.

In previous research [13] we:

- Compared the performance of different cloud resource types to determine which types produce the best performance for different workflow applications,
- Compared the performance of cloud resources with typical HPC resources,
- Characterized the impact of virtualization on workflow performance and found it to be less than 10%,
- Analyzed the cost/performance tradeoff of using different cloud resources, and
- Identified and quantified the various costs associated with running workflows in the cloud.

The contributions of this paper are:

- A description of an approach that sets up a computational environment in the cloud to support the execution of scientific workflow applications.
- An overview of the issues related to workflow storage in the cloud and a discussion of the current storage options for workflows in the cloud.
- A comparison of the performance (runtime) of three real workflow applications using six different storage configurations on Amazon EC2.
- An analysis of the cost of running workflows with different storage systems on Amazon EC2.

Our results show that the cloud offers a convenient and flexible platform for deploying workflows with various storage systems. We find that there are many options available for workflow storage in the cloud, and that the performance of storage systems such as GlusterFS [10] is quite good. We also find that the cost of running workflows on Amazon EC2 is not prohibitive for the applications we tested, however the cost increases significantly when multiple virtual nodes are used. At the same time we did not observe a corresponding increase in performance.

The rest of the paper is organized as follows: Section 2 describes the set of workflow applications we chose for our experiments. Section 3 gives an overview of the execution environment we set up for the experiments on Amazon EC2. Section 4 provides a discussion and overview of storage systems (including various file systems) that are used to communicate data between workflow tasks. Sections 5 and 6 provide performance and cost comparisons of different storage systems. Section 7 compares the cost and performance of running workflows on specialized cluster compute nodes versus standard server class nodes. Section 8

evaluates the impact of choosing to run a submit host in the cloud versus outside the cloud. And Sections 9 and 10 describe related work and conclude the paper.

2. Workflow Applications

In order to evaluate the cost and performance of scientific workflows in the cloud we conducted experiments using three real workflow applications: an astronomy application (Montage), a seismology application (Broadband), and a bioinformatics application (Epigenome). These three applications were chosen because they cover a wide range of application domains and a wide range of resource requirements. Table 1 shows the relative resource usage of the applications in three different categories: I/O, memory, and CPU. The resource usage of the applications was determined using a workflow profiler[1], which measures the I/O, CPU usage, and peak memory by tracing all the tasks in the workflow using ptrace [24].

Table 1: Application resource usage comparison

Application	I/O	Memory	CPU
Montage	High	Low	Low
Broadband	Medium	High	Medium
Epigenome	Low	Medium	High

The first application, Montage [16], creates science-grade astronomical image mosaics using data collected from telescopes. The size of a Montage workflow depends upon the area of the sky (in square degrees) covered by the output mosaic. In our experiments we configured Montage workflows to generate an 8-degree square mosaic. The resulting workflow contains 10,429 tasks, reads 4.2 GB of input data, and produces 7.9 GB of output data (excluding temporary data). We consider Montage to be I/O-bound because it spends more than 95% of its time waiting on I/O operations.

The second application, Broadband [26], generates and compares seismograms from several high- and low-frequency earthquake simulation codes. Each Broadband workflow generates seismograms for several sources (scenario earthquakes) and sites (geographic locations). For each (source, site) combination the workflow runs several high- and low-frequency earthquake simulations and computes intensity measures of the resulting seismograms. In our experiments we used 6 sources and 8 sites to generate a workflow containing 768 tasks that reads 6 GB of input data and writes 303 MB of output data. We consider Broadband to be memory-intensive because more than 75% of its runtime is consumed by tasks requiring more than 1 GB of physical memory.

The third and final application, Epigenome [27], maps short DNA segments collected using high-throughput gene sequencing machines to a previously constructed reference genome using the MAQ software [17]. The workflow splits several input segment files into small chunks, reformats and converts the chunks, maps the chunks to the reference genome, merges the mapped sequences into a single output map, and computes the sequence density for each location of interest in the reference genome. The workflow used in our experiments maps human DNA sequences from chromosome 21. The workflow contains 529 tasks, reads 1.9 GB of input data, and produces 300 MB of output data. We consider Epigenome to be CPU-bound because it spends 99% of its runtime in the CPU and only 1% on I/O and other activities.

[1] http://pegasus.isi.edu/wfprof

3. Execution Environment

In this section we describe the execution environment that was used in our experiments. We ran experiments on Amazon's EC2 Infrastructure as a Service (IaaS) cloud [1]. EC2 was chosen because it is currently the most popular, feature-rich, and stable commercial cloud available.

There are many ways to configure an execution environment for workflow applications in the cloud. The environment can be deployed entirely in the cloud, or parts of it can reside outside the cloud. For the majority of this paper we have chosen the latter approach, mirroring the configuration used for workflows on the grid. In this configuration, shown in **Figure 1**, we have a submit host that runs outside the cloud to manage the workflows and set up the cloud environment, several worker nodes that run inside the cloud to execute tasks, and a storage system that also runs inside the cloud to store workflow inputs and outputs. In Section 8 we evaluate a similar architecture where the submit host is also provisioned in the cloud.Software

The execution environment is based on the idea of a *virtual cluster* [4,9]. A virtual cluster is a collection of virtual machines that have been configured to act like a traditional HPC cluster. Typically this involves installing and configuring job management software, such as a batch scheduler, and a shared storage system, such as a network file system. The challenge in provisioning a virtual cluster in the cloud is collecting the information required to configure the cluster software, and then generating configuration files and starting services. Instead of performing these tasks manually, which can be tedious and error-prone, we have used Wrangler [14] to provision and configure virtual clusters for this paper.

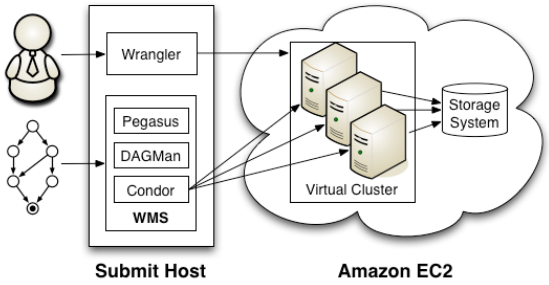

Figure 1: Execution environment on Amazon

All workflows were planned and executed using the Pegasus Workflow Management System [6], which includes the Pegasus mapper, DAGMan workflow engine [5] and the Condor scheduler [18]. Pegasus is used to transform a resource-independent, abstract workflow description into a concrete plan, which is then executed using DAGMan to manage dependencies between tasks, and Condor to manage task execution. With the exception of a few minor modifications to Pegasus, which were required to support Amazon S3 (see Section 4.1), the workflow management system did not require modifications to run on EC2.

To deploy software on the virtual cluster we developed a virtual machine image based on the stock Fedora 8 image provided by Amazon. To the stock image we added the Pegasus worker node tools, Globus clients, Condor worker daemons, and all other packages required to compile and run the selected workflows, including the application binaries. We also installed the Wrangler agent to manage the configuration of the virtual machines, and wrote shell scripts to generate configuration files and start the required services. Finally, we installed and configured the software necessary to run the storage systems that will be described in Section 4. The resulting image was used to deploy worker nodes on EC2.

3.2. Resources

Amazon EC2 offers several different virtual machine configurations called *instance types*. Each instance type is configured with a specific amount of memory, CPUs, and local storage. In our previous work [13] we examined the impact of different instance types on the performance and cost of workflow applications. Here we summarize those results in the form of cost-performance plots.

Figure 2 shows the cost-performance plots for the three example applications on Amazon EC2. Instances labeled m1.* have significant amounts of memory, c1* have powerful cores, and cc1.4xlarge is a cluster instance. Although in most cases these plots do not indicate the specific instance type to choose, they can highlight instance types that should not be used. For example, for Montage (top of Figure 2) the m1.small type can be eliminated from consideration because c1.medium is both faster and cheaper. Similar arguments can be made for m1.large, and c1.xlarge. The other instance types—cc1.4xlarge, m1.xlarge, and c1.medium—are all optimal solutions in the sense that there are no other instance types that are both faster and cheaper. The cc1.4xlarge type is the fastest, c1.medium is the cheapest, and m1.xlarge is faster than c1.medium, but cheaper than cc1.4xlarge. Formally, these three instance types comprise the *Pareto set* for Montage, and are called *Pareto optimal* solutions. Because the Pareto sets for the example applications all contain more than one instance type, there is no "best" choice for these applications. Choosing an instance type from the Pareto set still involves a cost-performance tradeoff based on the user's requirements. Application developers should be aware of the various tradeoffs between different instance types, and benchmark their applications to decide which type meets the requirements of their application, rather than blindly choosing the type with the most power or the lowest hourly rate.

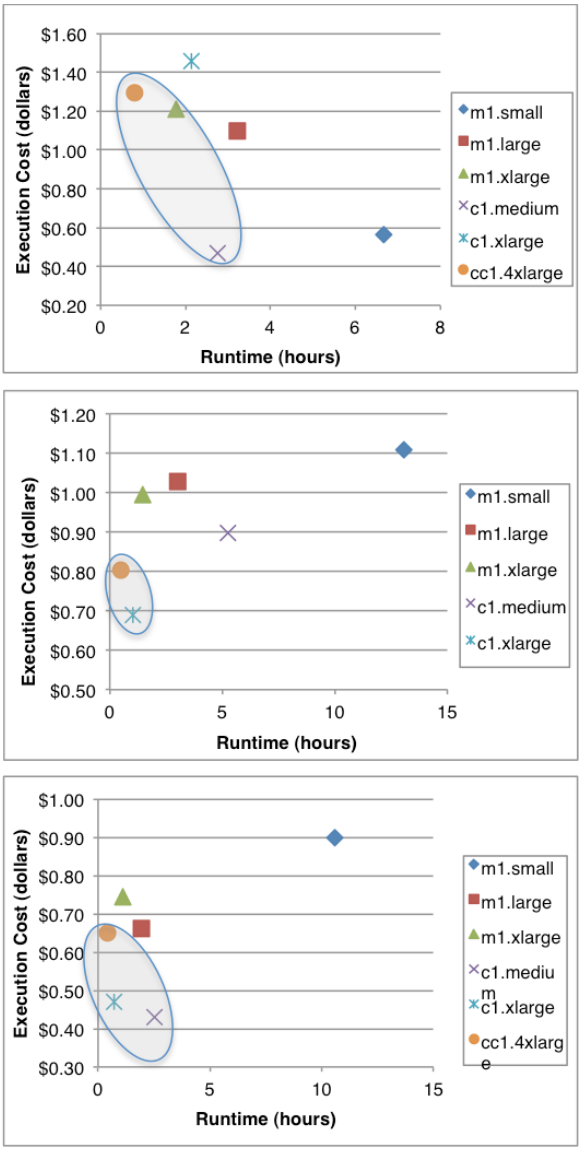

Figure 2: Cost versus performance comparison for Montage (top), Broadband (middle) and Epigenome (bottom). The gray ovals indicate Pareto optimal instance types.

We used the c1.xlarge instance type for the majority of experiments described here. A few experiments were conducted using the cc1.4xlarge "cluster compute" instance type. These experiments and the reasoning behind them are described in Section 7.

An exhaustive survey of all the possible combinations of application, storage system, and instance type is beyond the scope of this study.

3.3. Storage

To run workflows we need to allocate storage for 1) application executables, 2) input data, and 3) intermediate and output data. In a typical workflow application executables are pre-installed on the execution site, input data is copied from an archive to the execution site, and output data is copied from the execution site to an archive. Since the focus of this paper is on the storage systems we did not perform or measure data transfers to/from the cloud. Instead, executables were included in the virtual machine images, input data was pre-staged to the virtual cluster, and output data was not transferred back to the submit host.

Each of the c1.xlarge nodes used for our experiments has 4 "ephemeral" disks. These disks are virtual block-based storage devices that provide access to physical storage on local disk drives. Ephemeral disks appear as devices to the virtual machine and can be formatted and accessed as if they were physical devices. They can be used to store data for the lifetime of the virtual machine, but are wiped clean when the virtual machine is terminated. As such they cannot be used for long-term storage.

Ephemeral disks have a severe first write penalty that should be considered when deploying an application on EC2. One would expect that ephemeral disks should deliver performance close to that of the underlying physical disks, most likely around 100 MB/s, however, the observed performance is only about 20 MB/s for the first write. Subsequent writes to the same location deliver the expected performance. This appears to be the result of the virtualization technology used to expose the drives to the virtual machine. This problem has not been observed with standard Xen virtual block devices outside of EC2, which suggests that Amazon is using a custom disk virtualization solution, perhaps for security reasons. Amazon's suggestion for mitigating the first-write penalty is for users to initialize ephemeral disks by filling them with zeros before using them for application data. However, initialization is not feasible for many applications because it takes too much time. Initializing enough storage for a Montage workflow (50 GB), for example, would take almost as long (42 minutes) as running the workflow using an uninitialized disk. If the node using the disk is going to be provisioned for only one workflow, then initialization does not make economic sense.

For the experiments described in this paper we have not initialized the ephemeral disks. In order to get the best performance without initialization we used software RAID. We combined the 4 ephemeral drives on each c1.xlarge node into a single RAID 0 partition. This configuration results in first writes of 80-100 MB/s, and subsequent writes around 350-400 MB/s. Reads peak at around 110 MB/s from a single ephemeral disk and around 310 MB/s from a 4-disk RAID array. The RAID 0 disks were used as local storage for the systems described in the next section.

4. Storage Options

In this section we describe the storage services we used for our experiments and any special configuration or handling that was required to get them to work with our workflow management system. We tried to select a number of different systems that span a wide range of storage options. Given the large number of network storage systems available it is not possible for us to examine them all. In addition, it is not possible to run

some file systems on EC2 because Amazon does not allow kernel modifications (Amazon does allow modules, but many file systems require source code patches as well). This is the case for Lustre [21] and Ceph [30], for example. Also, in order to work with our workflow tasks (as they are provided by the domain scientists), the file system either needs to be POSIX-compliant (i.e. we must be able to mount it and it must support standard semantics), or additional tools need to be used to copy files to/from the local file system, which can result in reduced performance.

It is important to note that our goal with this work is not to evaluate the raw performance of these storage systems in the cloud, but rather to examine application performance in the context of scientific workflows. We are interested in exploring various options for sharing data in the cloud for workflow applications and in determining, in general, how the performance and cost of a workflow is affected by the choice of storage system. Where possible we have attempted to tune each storage system to deliver the best performance, but we have no way of knowing what combination of parameter values will give the best results for all applications without an exhaustive search. Instead, for each storage system we ran some simple benchmarks to verify that the storage system functions correctly and to determine if there are any obvious parameters that should be changed. We do not claim that the configurations we have used are the best of all possible configurations for our applications, but rather represent a typical setup.

In addition to the systems described below we ran a few experiments using XtreemFS [12], a file system designed for wide-area networks. However, the workflows performed far worse on XtreemFS than the other systems tested, taking more than twice as long as they did on the storage systems reported here before they were terminated without completing. It is possible that these issues with XtreemFS were a result of incompatibilities between XtreemFS and the kernel supplied by Amazon. We were unable to resolve these issues and, as a result, we did not perform a full range of experiments with XtreemFS.

4.1. Amazon S3

Amazon S3 [2] is a distributed, object-based storage system. It stores un-typed binary objects (e.g. files) up to 5 GB in size. It is accessed through a web service that supports both SOAP and REST. Objects in S3 are stored in directory-like structures called *buckets*. Each bucket is owned by a single user and must have a globally unique name. Objects within a bucket are named by *keys*. The key namespace is flat, but path-like keys are allowed (e.g. “a/b/c” is a valid key).

Because S3 does not have a POSIX interface, in order to use it, we needed to make some modifications to the workflow management system. The primary change was adding support for an S3 client, which copies input files from S3 to the local file system before a job starts, and copies output files from the local file system back to S3 after the job completes. The workflow management system was modified to wrap each job with the necessary GET and PUT operations.

Transferring data for each job individually increases the amount of data that must be moved and, as a result, has the potential to reduce the performance of the workflow. Using S3 each file must be written twice when it is generated (program to disk, disk to S3) and read twice each time it is used (S3 to disk, disk to program). In comparison, network file systems enable the file to be written once, and read once each time it is used. In addition, network file systems support partial reads of input files and fine-grained overlapping of computation and communication. In order to reduce the number of transfers required when using S3 we

implemented a simple whole-file caching mechanism. Caching is possible because all the workflow applications used in our experiments obey a strict write-once file access pattern where no files are ever opened for updates. Our simple caching scheme ensures that each file is transferred from S3 to a given node only once, and saves output files generated on a node so that they can be reused as input for future jobs that may run on the node.

The scheduler that was used to execute workflow jobs does not consider data locality or parent-child affinity when scheduling jobs, and does not have access to information about the contents of each node's cache. Because of this, if a file is cached on one node, a job that accesses the file could end up being scheduled on a different node. A more data-aware scheduler could potentially improve workflow performance by increasing cache hits and further reducing transfers.

4.2. NFS

NFS [25] is perhaps the most commonly used network file system. Unlike the other storage systems used, NFS is a centralized system with one node that acts as the file server for a group of machines. This puts it at a distinct disadvantage in terms of scalability compared with the other storage systems.

We deployed NFS on EC2 in two different configurations. The first configuration used a dedicated m1.xlarge node to host the NFS file system. We used m1.xlarge instead of c1.xlarge because it has a comparatively large amount of memory (16GB vs 8GB), which improves cache performance. Using a dedicated node improves performance, but increases cost. In order to evaluate the impact of this on performance and cost, we deployed a second configuration that used one of the c1.xlarge nodes to both host the file system and run compute jobs. In both cases we configured the NFS clients to use the *async* option, which allows calls to NFS to return before the data has been flushed to disk, and we disabled *atime* updates.

4.3. GlusterFS

GlusterFS [10] is a distributed file system that supports many different configurations. It has a modular architecture based on components called *translators* that can be composed to create novel file system configurations. All translators support a common API and can be stacked on top of each other in layers. As a result of these translators, there are many ways to deploy a GlusterFS file system. We used two configurations: *NUFA* (non-uniform file access) and *distribute*. In both configurations nodes act as both clients and servers. Each node exports a local volume and merges it with the local volumes of all other nodes. In the NUFA configuration, all writes to new files are performed on the local disk, while reads and writes to existing files are either performed across the network or locally depending on where the file was created. Because files in the workflows we tested are never updated, the NUFA configuration results in all write operations being directed to the local disk. In the distribute configuration, GlusterFS uses hashing to distribute files among nodes. This configuration results in a more uniform distribution of reads and writes across the virtual cluster compared to the NUFA configuration, but may result in more network I/O.

4.4. PVFS

PVFS [3] is a parallel file system for Linux clusters. It distributes file data via striping across a number of I/O nodes. In our configuration we used the same set of nodes for both I/O and computation. In other words, each node was configured as both a client

and a server. In addition, we configured PVFS to distribute metadata across all nodes instead of having a central metadata server.

Although the latest version of PVFS was 2.8.2 at the time our experiments were conducted, we were not able to run any of the 2.8 series releases on EC2 reliably without crashes or loss of data. Instead, we used an older version, 2.6.3, and applied a patch for the Linux kernel used on EC2 (2.6.21). This version ran without crashing, but does not include some of the changes made in later releases to improve support and performance for small files.

5. Performance Comparison

In this section we compare the performance of the selected storage options for workflows on Amazon EC2. The critical performance metric we are concerned with is the total runtime of the workflow (also known as the makespan). The runtime of a workflow is defined as the total amount of wall clock time from the moment the first workflow task is submitted until the last task completes. The runtimes reported in the following sections do not include the time required to provision and configure the VMs, which typically averages between 50 and 110 seconds to boot the operating system, and 90 to 120 seconds to boot the OS and configure the storage systems [14]. The runtimes also do not include the time required to transfer input and output data. Because the sizes of input files are constant, and the resources are all provisioned at the same time, the file transfer and provisioning overheads are assumed to be independent of the storage system chosen.

In discussing the results for various storage systems it is useful to consider the I/O workload generated by the applications tested. All of the applications generate a large number (thousands) of relatively small files (on the order of 1 MB to 10 MB). The write pattern is sequential and strictly write-once (no file is updated after it has been created). The read pattern is primarily sequential, with a few tasks performing random accesses. Because many workflow jobs run concurrently, many files will be accessed at the same time. Some files are read concurrently, but no file is ever read and written at the same time. These characteristics will help to explain the observed performance differences between the storage systems in the following sections.

When comparing the results for different storage systems, note that the GlusterFS and PVFS configurations that were used require at least two nodes to construct a valid file system, so results with one worker are reported only for S3 and NFS. Also, in addition to the storage systems described in Section 4, we have included performance results for experiments run on a single c1.xlarge node with 8 cores using the local disk from our previous work [13]. These results are shown as a single point in the graphs and are labeled "Local".

5.1. Montage

The performance results for Montage are shown in 3. The characteristic of Montage that seems to have the most significant impact on its performance is the large number (~29,000) of relatively small (a few MB) files it accesses. GlusterFS seems to handle this workload well, with both the NUFA and distribute modes producing significantly better performance than the other storage systems. NFS does relatively well for Montage, beating even the local disk in the single node case. This may be because we used the *async* option with NFS, which results in better NFS write performance than a local disk when the remote host has a large amount of memory in which to buffer writes, or because using NFS results in less disk contention. Surprisingly, the shared NFS

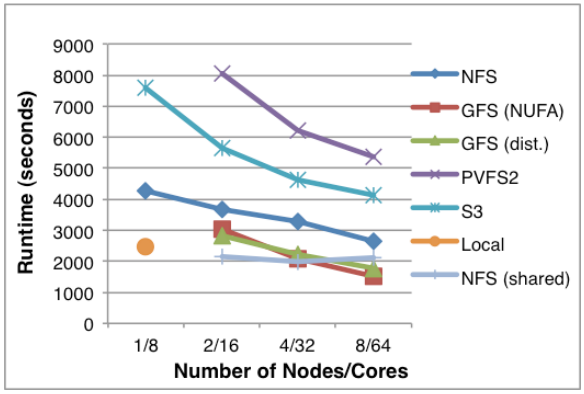

Figure 3: Performance of Montage using different storage systems.

configuration performed better than the NFS configuration using a dedicated node. This is likely a result of the fact that in the shared configuration 1/N of the file accesses are local (where N is the number of nodes), while in the dedicated configuration all accesses are performed over the network, which has larger latencies and lower bandwidth than the local disk. The relatively poor performance of S3 and PVFS may be a result of Montage accessing a large number of small files. As we indicated in Section 4, the version of PVFS we have used does not contain the small file optimizations added in later releases. S3 performs worse than the other systems on small files because of the relatively large overhead of fetching and storing files in S3. In addition, the Montage workflow does not contain much file reuse, which makes the S3 client cache less effective.

5.2. Broadband

The performance results for Broadband are shown in Figure 4. In contrast to the other applications, the best overall performance for Broadband was achieved using Amazon S3 and not GlusterFS. This is likely due to the fact that Broadband reuses many input files, which improves the effectiveness of the S3 client cache. Many of the transformations in Broadband consist of several executables that are run in sequence like a mini workflow. This would explain why GlusterFS (NUFA) results in better performance than GlusterFS (distribute). In the NUFA case all the outputs of a transformation are stored on the local disk, which results in much better locality for Broadband's workflow-like transformations. An additional Broadband experiment was run using a different NFS server (m2.4xlarge, 64 GB memory, 8 cores) to see if a more powerful server would significantly improve NFS performance. The result was better than the smaller server for the 4-node case (4368 seconds vs. 5363 seconds), but was still significantly worse than GlusterFS and S3 (<3000 seconds in all cases). The decrease in performance using NFS between 2 and 4 nodes was consistent across repeated experiments and was not affected by any of the NFS parameter changes we tried. Similar to Montage, the shared NFS configuration performed better than the dedicated configuration, most likely due to the larger proportion of local file accesses. Also similar to Montage, Broadband appears to have relatively poor performance on PVFS, possibly because of the large number of small files it generates (>5,000).

5.3. Epigenome

The performance results for Epigenome are shown in Figure 5. Epigenome is mostly CPU-bound, and performs relatively little I/O compared to Montage and Broadband. As a result, the choice of storage system has less of an impact on the performance of Epigenome compared to the other applications. In general, the performance was almost the same for all storage systems, with S3 and PVFS performing slightly worse than NFS and GlusterFS. Unlike Montage, for which NFS performed better than the local disk in the single

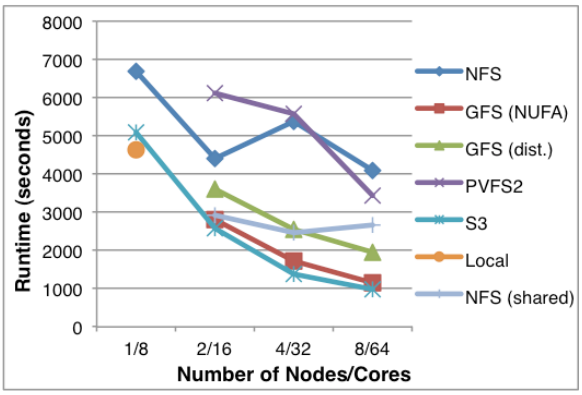

Figure 4: Performance of Broadband using different storage systems.

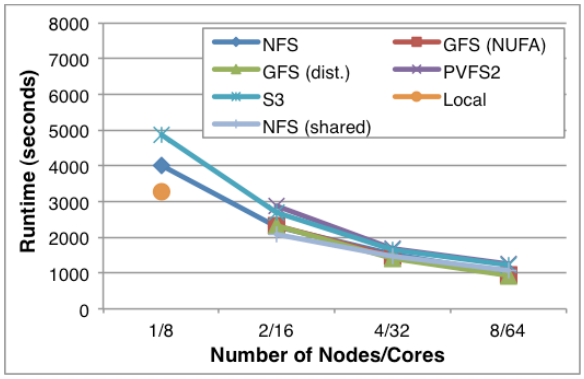

Figure 5: Performance of Epigenome using different storage systems.

node case, for Epigenome the local disk was significantly faster.

6. Cost Comparison

In this section we analyze the cost of running workflow applications using the selected storage systems. There are three different cost categories when running an application on EC2. These include: resource cost, storage cost, and transfer cost. Resource cost includes charges for the use of VM nodes in EC2; storage cost includes charges for keeping VM images and input data in S3 or EBS; and transfer cost includes charges for moving input data, output data and log files between the submit host and EC2.

One important issue to consider when evaluating the cost of a workflow is the granularity at which the provider charges for resources. In the case of EC2, Amazon charges for resources by the hour, and any partial hours are rounded up. One important result of this is that there is no cost benefit to adding resources for workflows that run for less than an hour, even though doing so may improve runtime. Another result of this is that it is difficult to compare the costs of different solutions. In order to better illustrate the costs of the various storage systems we use two different ways to calculate the total cost of a workflow: per hour charges, and per second charges. Per hour charges are what Amazon actually charges for the usage, including rounding up to the nearest hour, and per second charges are what the experiments would cost if Amazon charged per second. We compute per second rates by dividing the hourly rate by 3,600 seconds.

It should be noted that the storage systems do not have the same cost profiles. NFS with a dedicated file server is at a disadvantage in terms of cost because of the extra node that was used to host the file system. This results in an extra cost of $0.68 per workflow for all applications. S3 is also at a disadvantage compared to the other systems because Amazon charges a fee to store data in S3. This fee is $0.01 per 1,000 PUT operations, $0.01 per 10,000 GET operations, and $0.15 per GB-month of storage (transfers are free within EC2). For Montage this results in an extra cost of $0.28, for Epigenome the extra cost is $0.01, and for Broadband the extra cost is $0.02. Note that the S3 cost is somewhat reduced by caching in the S3 client, and that the storage cost is insignificant for the applications tested (<< $0.01).

The resource cost for Montage, Epigenome and Broadband using actual cost, and including extra charges for NFS and S3, is shown in Figure 6. In

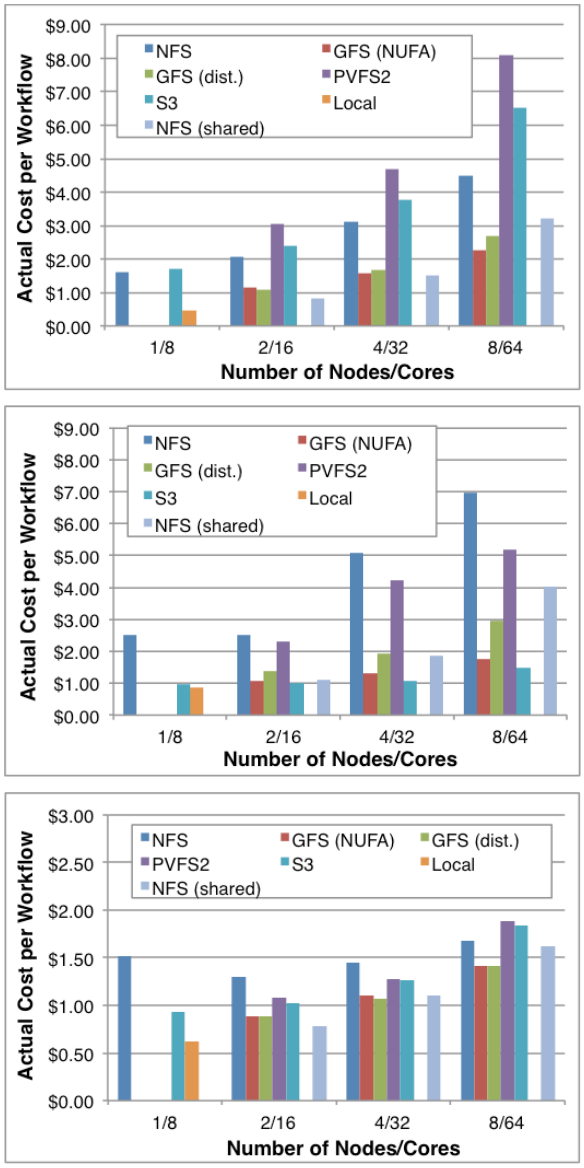

Figure 6: Actual cost of Montage (top), Broadband (middle), and Epigenome (bottom) using different storage systems.

addition to the cost of running on the storage systems described in Section 4, we also include the cost of running on a single node using the local disk from our previous work [13] (Local in the figures). For Montage the lowest cost solution was GlusterFS on two nodes. This is consistent with GlusterFS producing the best performance for Montage. For Epigenome the lowest cost solution was a single node using the local disk. Also notice that, because Epigenome is not I/O intensive, the difference in cost between the various storage solutions is relatively small. For Broadband the local disk, GlusterFS and S3 all tied for the lowest cost.

One final point to make about the cost of these experiments is the effect of adding resources. Assuming that resources have uniform cost and performance, in order for the cost of a workflow to decrease when resources are added the speedup of the application must be super-linear. Since this is rarely the case in any parallel application it is unlikely that there will ever be a cost benefit for adding resources, even though there may still be a performance benefit. In our experiments adding resources reduced the cost of a workflow for a given storage system in only 2 cases: 1 node to 2 nodes using NFS for both Epigenome and Broadband. In both of those cases the improvement was a result of the non-uniform cost of resources due to the extra node that was used for NFS. In all other cases the cost of the workflows only increased when resources were added. Assuming that cost is the only consideration and that resources are uniform, the best strategy is to either provision only one node for a workflow, or to use the fewest number of resources possible to achieve the required performance.

7. Cluster Compute Instance Type

Recently Amazon introduced two new instance types that are designed specifically for high-performance computing. The new "cluster compute" (cc1.4xlarge) and "cluster GPU" (cg1.4xlarge) nodes use 10Gbps Ethernet in comparison with the 1Gbps Ethernet used by the other instance types, and it is possible to ensure that cluster nodes are provisioned "close" to each other (in terms of network locality) in order to minimize network latency. These features

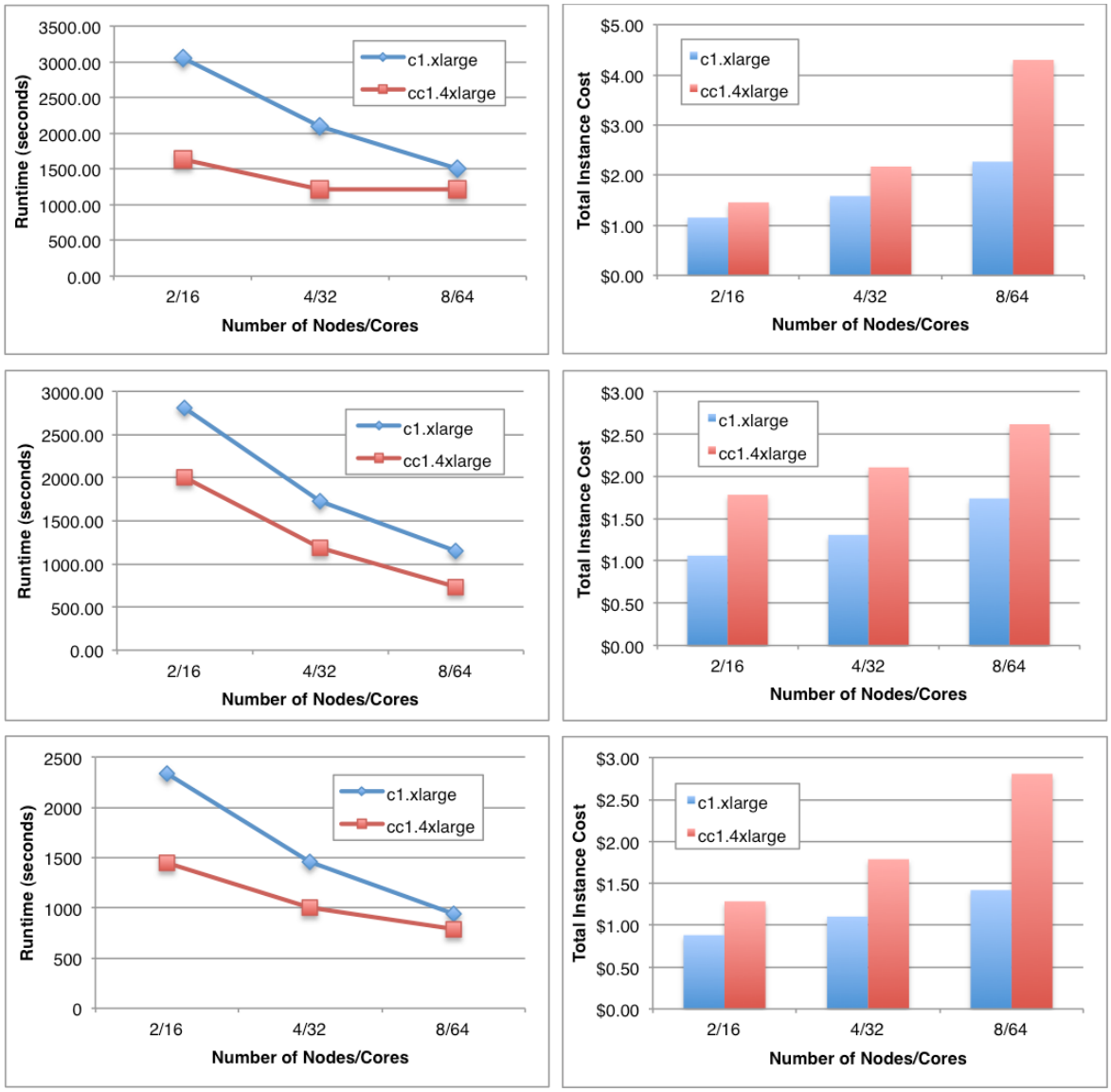

Figure 7: Performance and cost comparison between c1.xlarge and cc1.4xlarge for Montage (top), Broadband (middle), and Epigenome (bottom).

provide significantly better network performance for parallel applications using the new instance types. The new types are also fully virtualized using Xen HVM [15] instead of using paravirtualization. HVM improves application performance by reducing virtualization overhead, and enables the nodes to deploy custom kernels, which is not possible with the older instance types. Both new cluster types have higher-performance, dual, quad-core 2.93 Ghz Intel Xeon "Nehalem" processors. In comparison, most of the c1.xlarge instance types have older, dual, quad-core 2.13 or 2.33 Ghz Intel Xeon processors. The cluster GPU types also include two NVIDIA Tesla "Fermi" GPUs.

The new instance types come at a significantly higher cost than the older instance types. Cluster compute nodes cost $1.60 per hour, while cluster GPU nodes cost $2.10 per hour. In comparison, the c1.xlarge instance type used in the previous storage comparison

Table 2: Performance and cost comparison when switching from c1.xlarge to cc1.4xlarge.

	2 Nodes			4 Nodes			8 Nodes		
	Δ Runtime	Δ Cost	|ΔR/ΔC|	Δ Runtime	Δ Cost	|ΔR/ΔC|	Δ Runtime	Δ Cost	|ΔR/ΔC|
Montage	-46.3%	26.3%	1.76	-41.7%	37.2%	1.12	-19.4%	89.6%	0.22
Broadband	-28.7%	67.7%	0.42	-31.6%	61.0%	0.52	-36.1%	50.5%	0.71
Epigenome	-38.1%	45.7%	0.83	-31.1%	62.1%	0.50	-15.9%	98.0%	0.16

cost $0.68 per hour—58-68% less than the new instance types. In order to offset the increased cost, the cluster compute nodes would have to deliver twice the performance of the c1.xlarge type.

To compare the new, high-performance instance types to the older instance types we ran a set of experiments using the three example workflow applications. Since these applications cannot take advantage of the GPUs provided by the cluster GPU nodes without being modified (which would require significant changes to the application code), we restricted our comparison to the cluster compute (cc1.4xlarge) nodes. We compared the performance of these new nodes to the c1.xlarge results obtained in the previous experiments. Since GlusterFS in the NUFA configuration performed well for all three applications in the previous experiments, we restrict the comparison to that storage system.

The performance and cost comparison are shown in Figure 7. In all cases, the use of cc1.4xlarge nodes resulted in significantly better performance, but significantly increased cost, compared with the c1.xlarge nodes.

The performance improvement decreases as the number of nodes increases due to the limited scalability of the application. Also, the cost curves diverge as a result of weaker performance improvements as the number of nodes increases, so that the cost increases with cc1.4xlarge at a faster rate than with c1.xlarge.

In a few cases, the performance benefit of switching from c1.xlarge to cc1.4xlarge might be worth the increased cost. Table 2 compares the percent change in runtime versus the percent change in cost. In most cases, the improvement in runtime is less than the increase in cost, suggesting that the benefit is not worth the cost. For Montage with 2 or 4 nodes, however, the improvement in performance is greater than the increased cost.

8. Submit Host in the Cloud

In the previous experiments we provisioned worker nodes in the cloud, but used a submit host outside the cloud to manage the workflows. This configuration was used for a number of reasons: it is the way we assumed most people would want to deploy their workflows, it made experiment management and setup easier, and it provided a stable and permanent base from which we could run identical experiments over time. It also required us to provision one less node, which reduced cost. The drawback of this approach is that, although the amount of data transferred between the submit host and the workers is small, the overhead of communication over the WAN could impact the performance of the application.

In order to quantify the impact of submit node placement on performance and cost, we ran a set of experiments where the submit node was provisioned in the cloud and outside. Once again we used the c1.xlarge instance types for worker nodes, and GlusterFS in the NUFA configuration for shared storage. For the experiments were the submit host was provisioned inside the cloud we used the m1.xlarge instance type for the submit host, which costs $0.68 per hour. For the experiments where the submit host was outside the

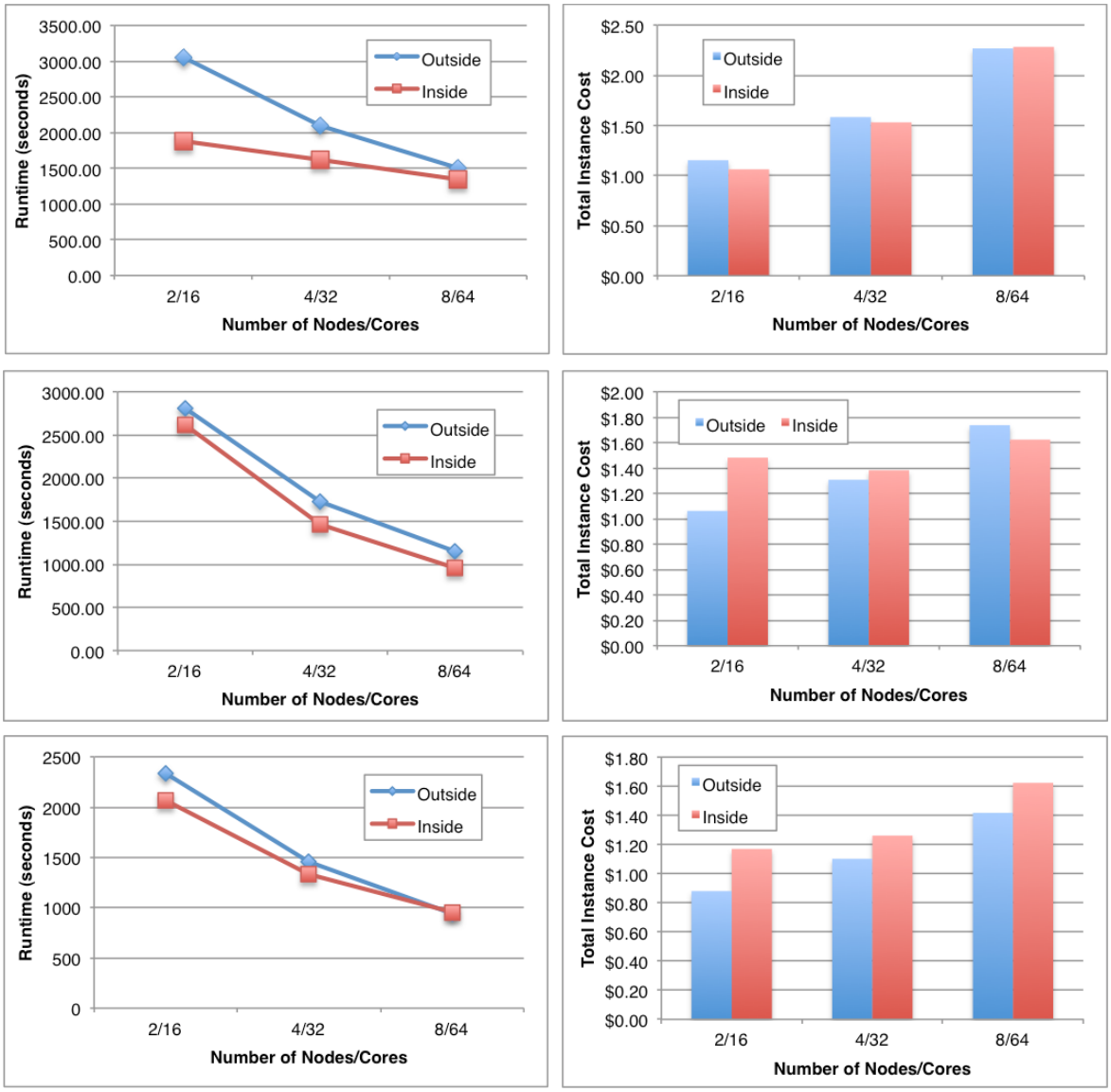

Figure 8: Performance and cost comparison when running the submit host inside the cloud, or outside the cloud for Montage (top), Broadband (middle), and Epigenome (bottom).

cloud, the total cost of the data transfer required was computed in previous work and found to be much less than $0.01 per workflow instance [13].

The results of these experiments are shown in Figure 8. Overall, the location of the submit host does have an impact on the performance of the workflow, with the submit host inside the cloud resulting in somewhat better performance in almost every case. Of all the applications, Epigenome has the least benefit, most likely because there are fewer jobs in the Epigenome workflow and therefore less traffic between the submit host and the worker nodes. Montage has the greatest benefit, which is a result of having the most jobs and the most traffic between the submit host and the workers.

Interestingly, the cost of running with the submit host in the cloud is in many cases less than the cost of running with the submit host outside, despite the fact

Table 3: Performance and cost comparison when switching from a submit host outside the cloud to a submit host inside the cloud

	2 Nodes			4 Nodes			8 Nodes		
	Δ Runtime	Δ Cost	|ΔR/ΔC|	Δ Runtime	Δ Cost	|ΔR/ΔC|	Δ Runtime	Δ Cost	|ΔR/ΔC|
Montage	-38.55%	-8.49%	4.54	-22.66%	-3.44%	6.58	-10.52%	0.66%	15.91
Broadband	-7.00%	39.50%	0.18	-15.42%	5.73%	2.69	-16.94%	-6.56%	2.58
Epigenome	-11.40%	32.90%	0.35	-8.42%	14.47%	0.58	1.81%	14.54%	0.12

that using a submit host inside requires an extra node. This is a result of the decrease in total runtime across all nodes, which offset the cost of the additional node.

Table 3 shows the performance and cost comparison when switching from a submit host outside, to a submit host inside the cloud. In 5 out of 9 cases the change is beneficial, and in 3 of 9 cases the cost actually decreases.

Aside from performance and cost, however, developers of workflow applications should consider application requirements and convenience when deciding upon the location of the submit host. Having a submit host outside the cloud provides a permanent base for storing workflow descriptions, execution logs, data, and metadata that will not be lost when the workflow completes. If a submit host is provisioned in the cloud, then additional measures must be taken to ensure that this information is transferred to a permanent storage location before the submit host is deprovisioned. Other issues, such as the availability of an existing local submit host, the use of multiple clouds, and the combination of local resources with cloud resources would also affect the decision. On the other hand, provisioning submit hosts in the cloud has the advantage of being able to support multiple, large-scale workflows at the same time by enabling the developer to provision a separate submit host for each workflow instance.

9. Related Work

Much previous research has investigated the performance of parallel scientific applications on virtualized and cloud platforms [7,8,11,19,22,23,29,31,32]. Our work differs from these in two ways. First, most of the previous efforts have focused on tightly-coupled applications such as MPI applications. In comparison, we have focused on scientific workflows, which are loosely-coupled parallel applications with very different requirements (although it is possible for individual workflow tasks to use MPI, we did not consider workflows with MPI tasks here). Second, previous efforts have focused mainly on micro benchmarks and benchmark suites such as the NAS parallel benchmarks [20]. Our work, on the other hand, has focused on the performance and cost of real-world applications.

Vecchiola, et al. have conducted research similar to our work [28]. They ran an fMRI workflow on Amazon EC2 using S3 for storage, compared the performance to Grid'5000, and analyzed the cost on different numbers of nodes. In comparison, our work is broader in scope. We use several applications from different domains with different resource requirements, and we experiment with six different storage configurations.

10. Conclusion

In this paper we examined the performance and cost of several different storage systems that can be used to communicate data within a scientific workflow running on the cloud. We evaluated the performance and cost of three workflow applications representing diverse application domains and resource requirements on Amazon's EC2 platform using different numbers of

resources (1-8 nodes corresponding to 8-64 cores) and six different storage configurations. Overall we found that cloud platforms like EC2 can be a practical solution for deploying workflow applications. The performance of EC2 was good enough for many applications, and the cost was within reason for the applications and scenarios studied.

One of the major factors inhibiting storage performance on EC2 is the first write penalty on ephemeral disks. We found that this significantly reduced the performance of storage systems deployed in EC2. This penalty appears to be unique to EC2, so repeating these experiments on another cloud platform may produce better storage system performance given a similar setup.

We found that the choice of storage system has a significant impact on workflow runtime. This is consistent with what we expect for non-cloud environments, so the use of clouds does not appear to be the most significant factor in storage system performance. GlusterFS delivered good performance for all the applications tested and seemed to perform well with both a large number of small files, and a large number of clients. S3 produced good performance for one application, possibly due to the use of caching in our implementation of the S3 client, but it's performance suffered from long latencies to access data on applications with many small files. NFS performed surprisingly well in cases where there were either few clients, or when the I/O requirements of the application were low. Like S3, PVFS performed poorly on workflows with many small files, although the version of PVFS we used did not contain optimizations for small files that were included in subsequent releases.

As expected, we found that cost closely follows performance. In general, the storage systems that produced the best workflow runtimes resulted in the lowest cost. NFS was at a disadvantage compared to the other systems when it used an extra, dedicated node to host the file system, however, overloading a compute node would not have significantly reduced the cost. Similarly, S3 is at a disadvantage, especially for workflows with many files, because Amazon charges a fee per S3 transaction. For two of the applications (Montage, I/O-intensive; Epigenome CPU-intensive) the lowest cost was achieved with GlusterFS, and for the other application (Broadband—memory-intensive) the lowest cost was achieved with S3.

Although the runtime of the applications tested improved when resources were added, the cost did not. This is a result of the fact that adding resources only improves cost if speedup is superlinear. Since that is rarely ever the case, it is better from a cost perspective to either provision one node to execute an application, or to provision the minimum number of nodes that will provide the desired performance. Also, since Amazon bills by the hour, it is more cost-effective to run for long-periods in order to amortize the cost of unused capacity. One way to achieve this is to provision a single virtual cluster and use it to run multiple workflows in succession.

The new cluster compute instance types offered by Amazon EC2 do result in significant performance gains over the older instance types, but in most cases the improvements were not significant enough to offset the increased cost.

We saw that performance gains could be achieved without increasing cost, or with only a small increase in cost, by provisioning a submit host inside the cloud to manage the workflow. However, we have found that using a submit host outside the cloud is more convenient because it provides a permanent base for storing workflow descriptions, execution logs, data, and metadata that would otherwise be lost when the

workflow completes and the submit host is deprovisioned.

Acknowledgements

This work was supported by the National Science Foundation under the SciFlow (CCF-0725332) and Pegasus (OCI-0722019) grants. This research made use of Montage, funded by the National Aeronautics and Space Administration's Earth Science Technology Office, Computation Technologies Project, under Cooperative Agreement Number NCC5-626 between NASA and the California Institute of Technology.

References

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

[2] Amazon.com, Simple Storage Service (S3), http://aws.amazon.com/s3.

[3] P. Carns, W. Ligon, R. Ross, and R. Thakur, "PVFS: A Parallel File System for Linux Clusters," *4th Annual Linux Showcase and Conference*, 2000.

[4] J.S. Chase, D.E. Irwin, L.E. Grit, J.D. Moore, and S.E. Sprenkle, "Dynamic virtual clusters in a grid site manager," *Proceedings of the 12th IEEE International Symposium on High Performance Distributed Computing (HPDC03)*, 2003.

[5] DAGMan, http://cs.wisc.edu/condor/dagman.

[6] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G.B. Berriman, J. Good, A. Laity, J.C. Jacob, and D.S. Katz, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.

[7] C. Evangelinos and C.N. Hill, "Cloud Computing for Parallel Scientific HPC Applications: Feasibility of Running Coupled Atmosphere-Ocean Climate Models on Amazon's EC2," *Cloud Computing and Its Applications (CCA 2008)*, 2008.

[8] R.J. Figueiredo, P.A. Dinda, and J.A.B. Fortes, "A case for grid computing on virtual machines," *23rd International Conference on Distributed Computing Systems*, 2003.

[9] I. Foster, T. Freeman, K. Keahey, D. Scheftner, B. Sotomayer, and X. Zhang, "Virtual Clusters for Grid Communities," *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID06)*, 2006.

[10] Gluster, Inc., GlusterFS, http://www.gluster.org.

[11] W. Huang, J. Liu, B. Abali, and D.K. Panda, "A Case for High Performance Computing with Virtual Machines," *20th annual international conference on Supercomputing (ICS 06)*, 2006.

[12] F. Hupfeld, T. Cortes, B. Kolbeck, J. Stender, E. Focht, M. Hess, J. Malo, J. Marti, and E. Cesario, "The XtreemFS architecture - a case for object-based file systems in Grids," *Concurrency and Computation: Practice & Experience*, vol. 20, no. 17, pp. 2049–2060, 2008.

[13] G. Juve, E. Deelman, K. Vahi, and G. Mehta, "Scientific Workflow Applications on Amazon EC2," *Workshop on Cloud-based Services and Applications in conjunction with 5th IEEE International Conference on e-Science (e-Science 2009)*, 2009.

[14] G. Juve and E. Deelman, "Automating Application Deployment in Infrastructure Clouds," *3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2011.

[15] P. Kärkkäinen and L. Kurth, XenOverview - Xen Wiki, http://wiki.xensource.com/xenwiki/XenOverview.

[16] D.S. Katz, J.C. Jacob, E. Deelman, C. Kesselman, S. Gurmeet, S. Mei-Hui, G.B. Berriman, J. Good, A.C. Laity, and T.A. Prince, "A comparison of two methods for building astronomical image mosaics on a grid," *34th International Conference on Parallel Processing Workshops (ICPP '05)*, 2005.

[17] H. Li, J. Ruan, and R. Durbin, "Mapping short DNA sequencing reads and calling variants using mapping quality scores," *Genome Research*, vol. 18, no. 11, pp. 1851–1858, 2008.

[18] M.J. Litzkow, M. Livny, and M.W. Mutka, "Condor: A Hunter of Idle Workstations," *8th International Conference of Distributed Computing Systems*, 1988.

[19] J. Napper and P. Bientinesi, "Can Cloud Computing Reach the Top500?," *Proceedings of the Workshop on UnConventional High Performance Computing*, 2009.

[20] NASA Advanced Supercomputing Division, NAS Parallel Benchmarks, http://www.nas.nasa.gov/Resources/Software/npb.html.

[21] Oracle Corporation, Lustre parallel filesystem, http://www.lustre.org.

[22] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing," *Proceedings of Cloudcomp 2009*, 2009.

[23] M.R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon S3 for science grids: a viable solution?," *Proceedings of the 2008 international workshop on Data-aware distributed computing (DADC 08)*, 2008.

[24] "ptrace - process trace," *Linux Programmer's Manual*.

[25] R. Sandberg, D. Golgberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network Filesystem," *USENIX Conference Proceedings*, 1985.

[26] Southern California Earthquake Center, Broadband Platform, http://scec.usc.edu/scecpedia/Broadband_Platform.

[27] USC Epigenome Center, http://epigenome.usc.edu.

[28] C. Vecchiola, S. Pandey, and R. Buyya, "High-Performance Cloud Computing: A View of Scientific Applications," *International Symposium on Parallel Architectures, Algorithms, and Networks*, 2009.

[29] E. Walker, "Benchmarking Amazon EC2 for High-Performance Scientific Computing," *Login*, vol. 33, no. 5, pp. 18–23.

[30] S.A. Weil, S.A. Brandt, E.L. Miller, D.D.E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," *7th Symposium on Operating Systems Design and Implementation (OSDI 06)*, 2006.

[31] L. Youseff, R. Wolski, B. Gorda, and C. Krintz, "Paravirtualization for HPC Systems," *Workshop on Xen in High-Performance Cluster and Grid Computing*, 2006.

[32] W. Yu and J.S. Vetter, "Xen-Based HPC: A Parallel I/O Perspective," *8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '08)*, 2008.