

Scientific Workflows in the Cloud

Gideon Juve and Ewa Deelman

Abstract The development of cloud computing has generated significant interest in the scientific computing community. In this chapter we consider the impact of cloud computing on scientific workflow applications. We examine the benefits and drawbacks of cloud computing for workflows, and argue that the primary benefit of cloud computing is not the economic model it promotes, but rather the technologies it employs and how they enable new features for workflow applications. We describe how clouds can be configured to execute workflow tasks, and present a case study that examines the performance and cost of three typical workflow applications on Amazon EC2. Finally, we identify several areas in which existing clouds can be improved and discuss the future of workflows in the cloud.

1 Introduction

In this chapter we consider the use of cloud computing for scientific workflow applications. Workflows are coarse-grained parallel applications that consist of a series of computational tasks logically connected by data- and control-flow dependencies. They are used to combine several different computational processes into a single coherent whole. Many different types of scientific analysis can be easily expressed as workflows and, as a result, they are commonly used to model computations in many science disciplines [13]. Using workflow technologies, components developed by different scientists, at different times, for different domains can be used together. Scientific workflows are used for simulation, data analysis, image processing, and many other functions.

Gideon Juve
University of Southern California, Marina del Rey, CA e-mail: juve@usc.edu

Ewa Deelman
University of Southern California, Marina del Rey, CA e-mail: deelman@isi.edu

Scientific workflows can range in size from just a few tasks to millions of tasks. For large workflows it is often desirable to distribute the tasks across many computers in order to complete the work in a reasonable time. As such, workflows often involve distributed computing on clusters, grids [18], and other computational infrastructures. Recently cloud infrastructures are also being evaluated as an execution platform for workflows [24, 28].

Cloud computing represents a new way of thinking about how to deploy and execute scientific workflows. On the one hand, clouds can be thought of as just another platform for executing workflow applications. They support all of the same techniques for workflow management and execution that have been developed for clusters and grids. With very little effort a scientist can deploy a workflow execution environment that mimics the environment they would use on a local cluster or national grid. On the other hand, clouds also provide several features, such as virtualization, that offer new opportunities for making workflow applications easier to deploy, manage and execute. In this chapter we examine those opportunities and describe how workflows can be deployed in the cloud today.

Many different types of clouds are being developed, both as commercial ventures and in the academic community. For the purposes of this chapter we are primarily interested in Infrastructure as a Service (IaaS) clouds [3] as these are more immediately useable by workflow applications. Other clouds, such as Platform as a Service (PaaS) and Software as a Service (SaaS) clouds, may provide additional benefits for creating, managing and executing workflow-based computations, but currently there is a lack of systems developed in this area and additional research is needed to determine how such systems can be fruitfully combined with workflow technologies.

2 Workflows in the Cloud

There is some disagreement about what is the killer feature of cloud computing. For many, especially those in the business community, the attractiveness of cloud is due to its utility-based computing model—the idea that someone else manages a set of computational resources and users simply pay to access them. The academic community, however, has had utility computing for quite a long time in the form of campus clusters, high-performance computing centers such as the NCSA [37] and the SDSC [44], and national cyberinfrastructure such as the TeraGrid [48] and the Open Science Grid [38]. Although the availability of commercial clouds may have some impact on the economics of large-scale scientific computing, we do not view economics as the fundamental benefit of cloud computing for science. Instead, we think clouds provide a multiplicity of benefits that are more technological in nature, and that these benefits stem, primarily, from the extensive use of service-oriented architectures and virtualization in clouds. In the following sections we discuss several aspects of cloud computing that are of particular benefit to workflow applications.

2.1 Provisioning

In grids and clusters scheduling is based on a best-effort model in which a user specifies their computation (the number of resources and amount of time required) and delegates responsibility for allocating resources and scheduling the computation to a batch scheduler. Requests for resources (jobs) are immediately placed into a queue and serviced in order, when resources become available, according to the policies of the scheduler. As a result, jobs often face long, unpredictable queue times, especially jobs that require large numbers of resources, or have long runtimes. The allocation of resources and binding of jobs to those resources are fundamentally tied together and out of the user's control. This is unfortunate for workflows because often the overheads of scheduling jobs on these platforms are high, and for a workflow containing many tasks the penalty is paid many times, which hurts performance [28].

In clouds the process is reversed. Instead of delegating allocation to the resource manager, the user directly provisions the resources required and schedules their computations using a user-controlled scheduler. This provisioning model is ideal for workflows and other loosely-coupled applications because it enables the application to allocate a resource once and use it to execute many tasks, which reduces the total scheduling overhead and can dramatically improve workflow performance [28, 45, 46, 41]. Although in clusters and grids, pilot job systems, such as Condor glide-ins [13] aim to simulate resource provisioning, they face limitations imposed by the target systems, for example the maximum walltime a job is allowed to run on a resource.

2.2 On-demand

Cloud platforms allocate resources on-demand. Cloud users can request, and expect to obtain, sufficient resources for their needs at any time. This feature of clouds has been called the "illusion of infinite resources". The drawback of this approach is that, unlike best-effort queuing, it does not provide an opportunity to wait. If sufficient resources are not available to service a request immediately, then the request fails.

On-demand provisioning allows workflows to be more opportunistic in their choice of resources. Unlike tightly-coupled applications, which need all their resources up-front and would prefer to wait in a queue to ensure priority, a workflow application can start with only a portion of the total resources desired. The minimum usable resource pool for workflows containing only serial tasks is one processor. With on-demand provisioning a workflow can allocate as many resources as possible and start making progress immediately.

2.3 Elasticity

In addition to provisioning resources on-demand, clouds also allow users to return resources on-demand. This dual capability, called elasticity, is a very useful feature for workflow applications because it enables workflow systems to easily grow and shrink the available resource pool as the needs of the workflow change over time. Common workflow structures such as data distribution and data aggregation can significantly change the amount of parallelism in a workflow over time [7]. These changes lead naturally to situations in which it may be profitable to acquire or release resources to more closely match the needs of the application and ensure that resources are being fully utilized.

2.4 Legacy Applications

Workflow applications frequently consist of a collection of complementary software components developed at different times for different uses by different people. Part of the job of a workflow management system is to weave these heterogeneous components into a single coherent application. Often this must be done without changing the components themselves. In some cases no one wants to modify codes that have been designed and tested many years ago in fear of introducing bugs that may affect the scientific validity of outputs. This can be challenging depending on the environment for which the components were developed and the assumptions made by the developer about the layout of the filesystem. These components are often brittle and require a specific software environment to execute successfully.

Clouds and their use of virtualization technology may make these legacy codes much easier to run. Virtualization enables the environment to be customized to suit the application. Specific operating systems, libraries, and software packages, can be installed, directory structures required by the application can be created, input data can be copied into specific locations, and complex configurations can be constructed. The resulting environment can be bundled up as a virtual machine image and redeployed on a cloud to run the workflow.

2.5 Provenance and Reproducibility

Provenance is the origin and history of an object [8]. In computational science, provenance refers to the storage of metadata about a computation that can be used to answer questions about the origins and derivation of data produced by that computation. As such, provenance is the computational equivalent of a lab scientist's notebook and is a critical component of reproducibility, the cornerstone of experimental science.

Virtualization is a useful feature for provenance because it allows one to capture the exact environment that was used to perform a computation, including all of the software and configuration used in that environment. In previous work [23] we proposed a provenance model for workflow applications in which virtual machine images are a critical component. The idea behind this model is that, if a workflow is executed using a virtual machine, then the VM image can be stored along with the provenance of the workflow. Storing the VM image enables the scientist to answer many important questions about the results of a workflow run such as: What version of the simulation code was used to produce the data? Which library was used? How was the software installed and configured? It also enables the scientist to redeploy the VM image and create exactly the same environment that was used to run the original experiment. This environment could be used to rerun the experiment, or it could be modified to run a new experiment. These capabilities are enabled by the use of virtualization in cloud computing.

3 Deploying Workflows in the Cloud

Workflow Management Systems such as Pegasus WMS [40, 16], plan, execute, and monitor scientific workflows. Pegasus WMS, which consists of the Pegasus mapper [14], the DAGMan execution engine [12], and the Condor *schedd* [19] for task execution, performs several critical functions. It adapts workflows to the target execution environment, it manages task execution on distributed resources, it optimizes workflow performance to reduce time to solution and produce scientific results faster, it provides reliability during execution so that scientists need not manage a potentially large number of failures, and they track data so that it can be easily located and accessed during and after workflow execution.

An approach to running workflows on the cloud is to build a virtual cluster using the cloud resources and schedule workflow tasks to that cluster.

3.1 Virtual Clusters

Scientific workflows require large quantities of compute cycles to process tasks. In the cloud these cycles are provided by virtual machines. To achieve the performance required for large-scale workflows many virtual machine instances must be used simultaneously. These collections of VMs are called “virtual clusters” [10, 17, 32]. The process of deploying and configuring a virtual cluster is known as contextualization [31]. Contextualization involves complex configuration steps that can be tedious and error-prone to perform manually. To automate this process, software such as the Nimbus Context Broker [31] can be used. This software gathers information about the virtual cluster and uses it to generate configuration files and start services on cluster VMs.

3.2 Resource Management

Having a collection of virtual machines is not sufficient to run a workflow. Additional software is needed to bind workflow tasks to resources for execution. The easiest way to do this is to use some off-the-shelf resource management system such as Condor [35], PBS [5] or Sun Grid Engine [20]. In this way the virtual cluster mimics a traditional computational cluster. A typical virtual cluster is composed of a virtual machine that acts as a head node to manage the other machines in the cluster and accept tasks from the workflow management system, and a set of worker nodes that execute tasks. Configuration of the resource manager and registration of worker nodes with the head node is part of the process of contextualization.

3.3 Data Storage

Workflows are loosely-coupled parallel applications that consist of a set of discrete computational tasks logically connected via data- and control-flow dependencies. Unlike tightly-coupled applications in which tasks communicate by sending message via the cluster network, workflow tasks typically communicate using files, where each task produces one or more output files that become input files to other tasks. These files are passed between tasks either through a shared storage system or using some file transfer mechanism.

In order to achieve good performance, these storage systems must scale well to handle data from multiple workflow tasks running in parallel on separate nodes. When running on HPC systems, workflows can usually make use of a high-performance, parallel file system such as Lustre [36], GPFS [43], or Panasas [27]. In the cloud, workflows can either make use of a cloud storage service, or they can deploy their own shared file system. To use a cloud storage service the workflow management system would likely need to change the way it manages data. For example, to use Amazon S3 [1] a workflow task needs to fetch input data from S3 to a local disk, perform its computation, then transfer output data from the local disk back to S3. Making multiple copies in this way can reduce workflow performance. Another alternative would be to deploy a file system in the cloud that could be used by the workflow. For example, in Amazon EC2 an extra VM can be started to host an NFS file system and worker VMs can mount that file system as a local partition. If better performance is needed then several VMs can be started to host a parallel file system such as PVFS [34, 50] or GlusterFS [25].

4 Case Study: Scientific Workflows on Amazon EC2

To illustrate how clouds can be used for workflow applications we present a case study using Amazon's EC2 [1] cloud. EC2 is a commercial cloud that exempli-

fies the IaaS model. It provides computational resources in the form of virtual machine instances, which come in a variety of hardware configurations and are capable of running several different virtualized operating systems. For comparison we ran workflows on both EC2 and NCSA's Abe cluster [37]. Abe is a typical example of the resources available to scientists on the national cyberinfrastructure. Running the workflows on both platforms allows us to compare the performance, features, and cost of a typical cloud environment to that of a typical grid environment. Figure 1 shows the high-level set up.

4.1 Applications Tested

Three workflow applications were chosen for this study: an astronomy application (Montage), a seismology application (Broadband), and a bioinformatics application (Epigenome). These applications were selected because they cover a wide range of science domains and resource requirements.

Montage [6] creates science-grade astronomical image mosaics from data collected using telescopes. The size of a Montage workflow (Figure 2) 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 mosaic. The resulting workflow contains 10,429 tasks, reads 4.2 GB of input data, and produces 7.9 GB of output data.

Broadband [9] generates and compares seismograms from several high- and low-frequency earthquake simulation codes. Each Broadband workflow (Figure 3) 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 4 sources and 5 sites to generate a workflow containing 320 tasks that reads 6 GB of input data and writes 160 MB of output data.

Epigenome [30] maps short DNA segments collected using high-throughput gene sequencing machines to a previously constructed reference genome using the MAQ software [33]. The workflow (Figure 4) splits several input segment files into small chunks, reformats and converts the chunks, maps the chunks to a 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 to a reference chromosome 21. The workflow contains 81 tasks, reads 1.8 GB of input data, and produces 300 MB of output data.

4.2 Software

All workflows were planned and executed using the Pegasus WMS. Pegasus is used to transform abstract, resource-independent workflow descriptions into concrete, platform-specific execution plans. These plans are executed using DAGMan to track dependencies and release tasks as they become ready, and Condor schedd to run tasks on the available resources.

4.3 Hardware

Table 1: Resource Types Used

Type	Arch.	CPU	Cores	Memory	Network	Storage	Price
m1.small	32-bit	2.0-2.6 GHz Opteron	1/2	1.7 GB	1-Gbps Ethernet	Local disk	\$0.10/hr
m1.large	64-bit	2.0-2.6 GHz Opteron	2	7.5 GB	1-Gbps Ethernet	Local disk	\$0.40/hr
m1.xlarge	64-bit	2.0-2.6 GHz Opteron	4	15 GB	1-Gbps Ethernet	Local disk	\$0.80/hr
c1.medium	32-bit	2.33-2.66 GHz Xeon	2	1.7 GB	1-Gbps Ethernet	Local disk	\$0.20/hr
c1.xlarge	64-bit	2.33-2.66 GHz Xeon	8	7.5 GB	1-Gbps Ethernet	Local disk	\$0.80/hr
abe.local	64-bit	2.33 GHz Xeon	8	8 GB	10-Gbps InfiniBand	Local disk	N/A
abe.lustre	64-bit	2.33 GHz Xeon	8	8 GB	10-Gbps InfiniBand	Lustre	N/A

EC2 provides resources with various hardware configurations. Table 1 compares the resource types used for the experiments. It lists five resource types from EC2 (m1.* and c1.*) and two resource types from Abe (abe.local and abe.lustre). There are several noteworthy details about the resources shown. First, although there is actually only one type of Abe node, there are two types listed in the table: abe.local and abe.lustre. The actual hardware used for these types is equivalent; the difference is in how I/O is handled. The abe.local type uses a local partition for I/O, and the abe.lustre type uses a Lustre partition for I/O. Using two different names is simply a notational convenience. Second, in terms of computational capacity, the c1.xlarge resource type is roughly equivalent to the abe.local resource type with the exception that abe.local has slightly more memory. We use this fact to estimate the virtualization overhead for our test applications on EC2. Third, in rare cases EC2 assigns Xeon processors for m1.* instances, but for all of the experiments reported here the m1.* instances used were equipped with Opteron processors. The only significance is that Xeon processors have better floating-point performance than Opteron processors (4 FLOP/cycle vs. 2 FLOP/cycle). Finally, the m1.small instance type is shown having core. This is possible because of virtualization. EC2 nodes are configured to give m1.small instances access to the processor only 50% of the time. This allows a single processor core to be shared equally between two separate m1.small instances.

4.4 Execution Environment

The execution environment was deployed on EC2 as shown in Figure 5 (left). A submit host running outside the cloud was used to coordinate the workflow, and worker nodes were started inside the cloud to execute workflow tasks. For the Abe experiments Globus [47] and Corral [29, 11] were used to deploy Condor glideins [22] as shown in Figure 5(right). The glideins started Condor daemons on the Abe worker nodes, which contacted the submit host and were used to execute workflow tasks. This approach creates an execution environment on Abe that is equivalent to the EC2 environment.

4.5 Storage

Amazon provides several storage services that can be used with EC2. The Simple Storage Service (S3) [39] is an object-based storage service. It supports PUT, GET, and DELETE operations for un-typed binary objects up to 5 GB in size. The Elastic Block Store (EBS) [2] is a block-based storage service that provides SAN-like storage volumes up to 1 TB in size. These volumes appear as standard block devices when attached to an EC2 instance and can be formatted with standard UNIX file systems. EBS volumes cannot be shared between multiple instances.

In comparison, Abe provides shared storage on a large Lustre [36] parallel file system. This is a very common storage configuration for cluster and grid platforms.

To run workflow applications storage must be allocated for application executables, input data, intermediate data, 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.

For all experiments, executables and input data were pre-staged to the execution site, and output data were not transferred from the execution site. For the EC2 experiments, executables were installed in the VM images, input data was stored on EBS volumes, and intermediate and output data was written to a local partition. For the Abe experiments executables and input data were kept on the Lustre file system and intermediate and output data was written in some cases to a local partition (abe.local experiments) and in other cases to the Lustre file system (abe.lustre experiments).

EBS was chosen to store input data for a number of reasons. First, storing inputs in the cloud obviates the need to transfer input data repeatedly. This saves both time and money because transfers cost more than storage. Second, using EBS avoids the 10 GB limit on VM images, which is too small to include the input data for all the applications tested. We can access the data as if it were on a local disk without packaging it in the VM image. A simple experiment using the disk copy utility ‘dd’ showed similar performance reading from EBS volumes and the local disk (74.6 MB/s for local, and 74.2 MB/s for EBS). Finally, using EBS simplifies our setup by allowing us to reuse the same volume for multiple experiments. When we need

to change instances we just detach the volume from one instance and re-attach it to another.

4.6 Performance Comparison

In this section we compare the performance of the selected workflow applications by executing them on Abe and EC2. The critical performance metric we are concerned with is the runtime of the workflow (also known as the makespan), which is the total amount of wall clock time from the moment the first workflow task is submitted until the last task completes. The runtimes reported for EC2 do not include the time required to install and boot the VM, which typically averages between 70 and 90 seconds [26]. Now this is much less if you use EBS to store images., and the runtimes reported for Abe do not include the time glidein jobs spend waiting in the queue, which is highly dependent on the current system load. Also, the runtimes do not include the time required to transfer input and output data (see Table 4). We assume that this time will be variable depending on WAN conditions. A study of bandwidth to/from Amazon services is presented in [39]. In our experiments we typically observed bandwidth on the order of 500-1000KB/s between Amazon's East Region datacenter and our submit host in Marina del Rey, CA.

We estimate the virtualization overhead for each application by comparing the runtime on `c1.xlarge` with the runtime on `abe.local`. Measuring the difference in runtime between these two resource types should provide a good estimate of the cost of virtualization.

Figure 6 shows the runtime of the selected applications using the resource types shown in Table 2. In all cases the `m1.small` resource type had the worst runtime by a large margin. This is not surprising given its relatively low capabilities.

For Montage the best EC2 performance was achieved on the `m1.xlarge` type. This is likely due to the fact that `m1.xlarge` has twice as much memory as the next best resource type. The extra memory is used by the Linux kernel for the file system buffer cache to reduce the amount of time the application spends waiting for I/O. This is particularly beneficial for Montage, which is very I/O-intensive. The best overall performance for Montage was achieved using the `abe.lustre` configuration, which was more than twice as fast as `abe.local`. This large gap suggests that having a parallel file system is a significant advantage for I/O-bound applications like Montage. The difference in runtime between the `c1.xlarge` and `abe.local` experiments suggests that the virtualization overhead for Montage is less than 8%.

The best overall runtime for Broadband was achieved by using the `abe.lustre` resource type, and the best EC2 runtime was achieved using the `c1.xlarge` resource type. This is despite the fact that only 6 of the 8 cores on `c1.xlarge` and `abe.lustre` could be used due to memory limitations. Unlike Montage the difference between running Broadband on a relatively slow local disk (`abe.local`) and running on the parallel file system (`abe.lustre`) is not as significant. This is attributed to the lower

I/O requirements of Broadband. Broadband performs the worst on m1.small and c1.medium, which also have the lowest amount memory (1.7 GB). This is because m1.small has only half a core, and c1.medium can only use one of its two cores because of memory limitations. The difference between the runtime using c1.xlarge and the runtime using abe.local was only about 1%. This small difference suggests a relatively low virtualization penalty for Broadband.

For Epigenome the best EC2 runtime was achieved using c1.xlarge and the best overall runtime was achieved using abe.lustre. The primary factor affecting the performance of Epigenome was the availability of processor cores, with more cores resulting in a lower runtime. This is expected given that Epigenome is almost entirely CPU-bound. The difference between the abe.lustre and abe.local runtimes was only about 2%, which is consistent with the fact that Epigenome has relatively low I/O and is therefore less affected by the parallel file system. The difference between the abe.local and the c1.xlarge runtimes suggest that the virtualization overhead for this application is around 10%, which is higher than both Montage and Broadband. This may suggest that virtualization has a larger impact on CPU-bound applications.

Based on these experiments we believe the performance of workflows on EC2 is reasonable given the resources that can be provisioned. Although the EC2 performance was not as good as the performance on Abe, most of the resources provided by EC2 are also less powerful. In the cases where the resources are similar, the performance was found to be comparable. The EC2 c1.xlarge type, which is nearly equivalent to abe.local, delivered performance that was nearly the same as abe.local in our experiments.

For I/O-intensive workflows like Montage, EC2 is at a significant disadvantage because of the lack of high-performance parallel file systems. While such a file system could conceivably be constructed from the raw components available in EC2, the cost of deploying such a system would be prohibitive. In addition, because EC2 uses commodity networking equipment it is unlikely that there would be a significant advantage in shifting I/O from a local partition to a parallel file system across the network, because the bottleneck would simply shift from the disk to the network interface. In order to compete performance-wise with Abe for I/O-intensive applications, Amazon would need to deploy both a parallel file system and a high-speed interconnect.

For memory-intensive applications like Broadband, EC2 can achieve nearly the same performance as Abe as long as there is more than 1 GB of memory per core. If there is less, then some cores must sit idle to prevent the system from running out of memory or swapping. This is not strictly an EC2 problem, the same issue affects Abe as well.

For CPU-intensive applications like Epigenome, EC2 can deliver comparable performance given equivalent resources. The virtualization overhead does not seem to be a significant barrier to performance for such applications. In fact, the virtualization overhead measured for all applications is less than 10%. This is consistent with previous studies that show similar virtualization overheads [4, 21, 49]. As such, virtualization does not seem, by itself, to be a significant performance problem for

clouds. As virtualization technologies improve it is likely that what little overhead there is will be further reduced or eliminated.

4.7 Cost Analysis

In this section we analyze the cost of running workflow applications in the cloud. We consider three different cost categories: resource cost, storage cost, and transfer cost. Resource cost includes charges for the use of VM instances in EC2; storage cost includes charges for keeping VM images in S3 and input data in EBS; and transfer cost includes charges for moving input data, output data and log files between the submit host and EC2.

4.7.1 Resource Cost

Each of the five resource types Amazon offers is charged at a different hourly rate: \$0.10/hr for m1.small, \$0.40/hr for m1.large, \$0.80/hr for m1.xlarge, \$0.20/hr for c1.medium, and \$0.80/hr for c1.xlarge. Usage is rounded up to the nearest hour, so any partial hours are charged as full hours.

Figure 7 shows the per-workflow resource cost for the applications tested. Although it did not perform the best in any of our experiments, the most cost-effective instance type was c1.medium, which had the lowest execution cost for all three applications.

4.7.2 Storage Cost

Storage cost consists of a) the cost to store VM images in S3, and b) the cost of storing input data in EBS. Both S3 and EBS use fixed monthly charges for the storage of data, and variable usage charges for accessing the data. The fixed charges are \$0.15 per GB-month for S3, and \$0.10 per GB-month for EBS. The variable charges are \$0.01 per 1,000 PUT operations and \$0.01 per 10,000 GET operations for S3, and \$0.10 per million I/O operations for EBS. We report the fixed cost per month, and the total variable cost for all experiments performed.

We used a 32-bit and a 64-bit VM image for all of the experiments in this paper. The 32-bit image was 773 MB and the 64-bit image was 729 MB for a total fixed cost of \$0.22 per month. In addition, there were 4616 GET operations and 2560 PUT operations for a total variable cost of approximately \$0.03.

The fixed monthly cost of storing input data for the three applications is shown in Table 2. In addition, there were 3.18 million I/O operations for a total variable cost of \$0.30.

Table 2: Monthly storage cost

Application	Volume Size	Monthly Cost
Montage	5GB	\$0.66
Broadband	5GB	\$0.60
Epigenome	2GB	\$0.26

4.7.3 Transfer Cost

In addition to resource and storage charges, Amazon charges \$0.10 per GB for transfer into, and \$0.17 per GB for transfer out of, the EC2 cloud. Tables 3 and 4 show the per-workflow transfer sizes and costs for the three applications studied. Input is the amount of input data to the workflow, output is the amount of output data, and logs is the amount of logging data that is recorded for workflow tasks and transferred back to the submit host. The cost of the protocol used by Condor to communicate between the submit host and the workers is not included, but it is estimated to be less than \$0.01 per workflow.

Table 3: Per-workflow transfer sizes

Application	Input	Output	Logs
Montage	4291 MB	7970 MB	40 MB
Broadband	4109 MB	159 MB	5.5 MB
Epigenome	1843 MB	299 MB	3.3 MB

Table 4: Per-workflow transfer costs

Application	Input	Output	Logs	Total
Montage	\$0.42	\$1.32	< \$0.01	\$1.75
Broadband	\$0.40	\$0.03	< \$0.01	\$0.43
Epigenome	\$0.18	\$0.05	< \$0.01	\$0.23

The first thing to consider when provisioning resources on EC2 is the tradeoff between performance and cost. In general, EC2 resources obey the aphorism “you get what you pay for”—resources that cost more perform better than resources that cost less. For the applications tested, c1.medium was the most cost-effective resource type even though it did not have the lowest hourly rate, because the type with the lowest rate (m1.small) performed so badly.

Another important thing to consider when using EC2 is the tradeoff between storage cost and transfer cost. Users have the option of either a) transferring input data for each workflow separately, or b) transferring input data once, storing it in the cloud, and using the stored data for multiple workflow runs. The choice of which approach to employ will depend on how many times the data will be used, how long the data will be stored, and how frequently the data will be accessed. In general, storage is more cost-effective for input data that is reused often and accessed frequently, and transfer is more cost-effective if data will be used only once. For the applications tested in this paper, the monthly cost to store input data is only slightly more than the cost to transfer it once. Therefore, for these applications, it is usually more cost-effective to store the input data rather than transfer the data for each workflow.

Although the cost of transferring input data can be easily amortized by storing it in the cloud, the cost of transferring output data may be more difficult to reduce. For many applications the output data is much smaller than the input data, so the cost of transferring it out may not be significant. This is the case for Broadband and Epigenome, for example. For other applications the large size of output data may be cost-prohibitive. In Montage, for example, the output is actually larger than the input and costs nearly as much to transfer as it does to compute. For these applications it may be possible to leave the output in the cloud and perform additional analyses there rather than to transfer it back to the submit host.

In [15] the cost of running 1-, 2-, and 4-degree Montage workflows on EC2 was studied via simulation. That paper found the lowest total cost of a 1-degree workflow to be \$0.60, a 2-degree to be \$2.25, and a 4-degree to be \$9.00. In comparison, we found the total cost of an 8-degree workflow, which is 4 times larger than a 4-degree workflow, to be approximately \$1.25 if data is stored for an entire month, and \$2.35 if data is transferred. This difference is primarily due to an underestimate of the performance of EC2 that was used in the simulation, which produced much longer simulated runtimes.

Finally, the total cost of all the experiments presented in this paper was \$149.55. That includes all charges related to learning to use EC2, creating VM images, and running test and experimental workflows.

5 Challenges

In the experiments above we focused on running a workflow application on a single, multi-core virtual instance. There are several challenges that need to be addressed when running workflows on multiple virtual instances. Here we describe the challenges related to data storage, communications, and configurability.

5.1 Lack of Appropriate Storage Systems

Existing workflow systems often rely on parallel and distributed file systems. These are required to ensure that tasks landing on any node can access the outputs of previous tasks that may have executed on another node. It is possible to transfer inputs and outputs for each task separately, however the repeated movement of data is highly inefficient, and time-consuming. In addition, it may be costly in a commercial cloud that charges by the number of bytes transferred. Commercial clouds often deploy structured or object-based storage services that can be utilized by workflow applications. However, these services typically do not provide standard file system interfaces. In order to use these systems the application codes must either be modified to interface with the storage services, or must be wrapped with additional workflow components that know how to do the translation. Another solution is to deploy a temporary shared file system in the cloud as part of a virtual cluster, but the problems with this solution are that it is complex, potentially costly, and requires an additional step to ensure that desired outputs are transferred to permanent storage. A better solution would be a permanent, scalable, parallel file system similar to what existing clusters and grids use. The challenge to this approach is that it is not clear how such a file system would be provisioned and shared among different users within a cloud. In particular, authentication and authorization would be key challenges of such a system.

5.2 Relatively Slow Networks

Most communication in workflows occurs through intermediate files that are written by one task and read by a subsequent task. In a distributed environment these files need to be transferred across the network in order for the workflow to make progress. As such, workflows depend on high-performance networks to achieve good performance. This is especially true for data-intensive workflows. Networks that provide high-throughput, but not necessarily low latency are ideal, however the predominant networking technology employed by existing commercial clouds is Gigabit Ethernet. In comparison, many cluster and grid infrastructures provide interconnects that deliver 10 Gigabit/second or better. In order for clouds to be a viable alternative to clusters and grids as a platform for workflow applications they would need to deploy much faster networks. Making high-performance networking function with OS-level virtualization should be a top priority for future cloud platforms.

5.3 Lack of Tools

Setting up an environment to run workflows in the cloud is a complex endeavor. There is some work in virtual appliances [42], but those are typically designed for

single nodes and not for clusters of nodes. The Nimbus Context Broker [31] can be used to construct virtual clusters and is immensely useful for running workflows in the cloud. More tools are needed to simplify the management of cloud-based environments.

6 Summary and Future Outlook

The benefit of cloud computing for science is not necessarily in its utility computing and economic aspects, which are not new for academic computing. The benefit of clouds is rather in its technological features that stem from service-oriented architecture and virtualization. In the area of scientific workflows clouds have many important benefits. These benefits include the illusion of infinite resources, lease-based provisioning, elasticity, support for legacy applications and environments, provenance and reproducibility, and others.

In our work, we supported the workflow execution on the cloud through the deployment of a Condor-based virtual cluster on top of virtual instance. Workflows can also be made to run across multiple virtual instances, but additional communication of data files need to be supported.

Cloud platforms like Amazon EC2 can be used to execute workflows today, but in the future much work is needed to bring these platforms up to the performance level of the grid. This includes developing cloud storage systems that are appropriate for workflow and other science applications as well as tools to help scientists and workflow engineers deploy their applications in the cloud.

In the future we will see many new cloud-based solutions for workflow applications. For example, we will likely see the development of new management tools that help users run workflows using existing tools in the grid. We may see new workflow systems that are designed with the specific features of the cloud in mind. We will see research on how to deploy workflows across grids and clouds. We may see PaaS and SaaS clouds that are developed exclusively for workflow applications. For example, a PaaS cloud may provide a user-centered Replica Location Service (RLS) for locating files in the cloud and outside, a dynamic Network Attached Storage (NAS) service for storing files used and created by workflows, a transformation catalog service to store and manage application binaries, and a workflow execution service for managing tasks and dependencies. A SaaS cloud for workflows may provide an application-specific portal where a user could enter the details of a desired computation and have the underlying workflow services generate a new workflow instance, plan and execute the computation, and provide access to the results.

Acknowledgements We acknowledge the contributions of Karan Vahi, Gaurang Mehta, Phil Maechling, Benjamin P. Berman, and Bruce Berriman. This work was supported by the National Science Foundation under the SciFlow (CCF-0725332) grant. 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: Amazon web services (aws). <http://aws.amazon.com>
2. Amazon.com: Elastic block store (ebs). <http://aws.amazon.com/ebs>
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A Berkeley view of cloud computing. Tech. rep., UC Berkeley (2009)
4. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., Warfield, A.: Xen and the art of virtualization. In: Proceedings of the 19th ACM symposium on Operating systems principles (2003)
5. Bayucan, A., Henderson, R.L., Lesiak, C., Mann, B., Proett, T., Tweten, D.: Portable batch system: External reference specification. Tech. rep., MRJ Technology Solutions (1999)
6. Berriman, B., Bergou, A., Deelman, E., Good, J., Jacob, J., Katz, D., Kesselman, C., Laity, A., Singh, G., Su, M.H., Williams, R.: Montage: A grid-enabled image mosaic service for the nvo. In: Astronomical Data Analysis Software and Systems (ADASS) XIII (2003)
7. Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.H., Vahi, K.: Characterization of scientific workflows. In: Proceedings of the 3rd Workshop on Workflows in Support of Large-Scale Science (WORKS '08) (2008)
8. Bruneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: Proceedings of the 8th International Conference on Database Theory (2001)
9. Center, S.C.E.: Community modeling environment. <http://www.scec.org/cme/>
10. Chase, J.S., Irwin, D.E., Grit, L.E., Moore, J.D., Sprenkle, S.E.: Dynamic virtual clusters in a grid site manager. In: 12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03) (2003)
11. Corral. <http://pegasus.isi.edu/corral/latest>
12. Dagman (directed acyclic graph manager). <http://cs.wisc.edu/condor/dagman>
13. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems* **25**(5), 528–540 (2008)
14. Deelman, E., Livny, M., Mehta, G., Pavlo, A., Singh, G., Su, M.H., Vahi, K., Wenger, R.K.: Pegasus and DAGMan from Concept to Execution: Mapping Scientific Workflows onto Today's Cyberinfrastructure, pp. 56–74. IOS Press (2008)
15. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: The montage example. In: Proceedings of the 2008 ACM/IEEE conference on Supercomputing (2008)
16. Deelman, E., Singh, G., Su, M.H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G.B., Good, J., Laity, A., Jacob, J.C., Katz, D.S.: Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming* **13**(3), 219–237 (2005)
17. Foster, I., Freeman, T., Keahey, K., Scheftner, D., Sotomayer, B., Zhang, X.: Virtual clusters for grid communities. In: Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06) (2006)
18. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications* **15**(3), 200–222 (2001)
19. Frey, J., Tannenbaum, T., Foster, I., Livny, M., Tuecke, S.: Condor-g: A computation management agent for multi-institutional grids. In: 10th International Symposium on High Performance Distributed Computing (2001)
20. Gentsch, W.: Sun grid engine: Towards creating a compute power grid. In: Proceedings of the 1st International Symposium on Cluster Computing and the Grid (2001)

21. Gilbert, L., Tseng, J., Newman, R., Iqbal, S., Pepper, R., Celebioglu, O., Hsieh, J., Cobban, M.: Performance implications of virtualization and hyper-threading on high energy physics applications in a grid environment. In: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) (2005)
22. Glidein. <http://www.cs.wisc.edu/condor/glidein>
23. Groth, P., Deelman, E., Juve, G., Mehta, G., Berriman, B.: Pipeline-centric provenance model. In: Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science (WORKS '09) (2009)
24. Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B., Good, J.: On the use of cloud computing for scientific workflows. In: Proceedings of the 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES '08) (2008)
25. Inc., G.: Glusterfs. <http://www.gluster.org>
26. Inc, H.: Cloudstatus. <http://www.cloudstatus.com>
27. Inc, P.: Panasas. <http://www.panasas.com>
28. Juve, G., Deelman, E.: Resource provisioning options for large-scale scientific workflows. In: Proceedings of the 3rd International Workshop on Scientific Workflows and Business Workflow Standards in e-Science (SWBES '08) (2008)
29. Juve, G., Deelman, E., Vahi, K., Mehta, G.: Experiences with resource provisioning for scientific workflows using corral. Scientific Programming (to appear)
30. Juve, G., Deelman, E., Vahi, K., Mehta, G., Berriman, B., Berman, B.P., Maechling, P.: Scientific workflow applications on amazon ec2. In: Workshop on Cloud-based Services and Applications in conjunction with 5th IEEE International Conference on e-Science (e-Science '09) (2009)
31. Keahey, K., Freeman, T.: Contextualization: Providing one-click virtual clusters. In: Proceedings of the 4th International Conference on eScience (eScience '08) (2008)
32. Kee, Y., Kesselman, C., Nurmi, D., Wolski, R.: Enabling personal clusters on demand for batch resources using commodity software. In: Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS '08) (2008)
33. Li, H., Ruan, J., Durbin, R.: Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Research* **18**(11), 1851–1858 (2008)
34. Ligon, W.B., Ross, R.B.: Implementation and performance of a parallel file system for high performance distributed applications. In: Proceedings of the Fifth IEEE International Symposium on High Performance Distributed Computing (1996)
35. Litzkow, M., Livny, M., Mutka, M.: Condor - a hunter of idle workstations. In: Proceedings of the 8th International Conference on Distributed Computing Systems (1988)
36. Microsystems, S.: Lustre. <http://www.lustre.org>
37. National center for supercomputing applications (ncsa). <http://www.ncsa.illinois.edu>
38. Open science grid. <http://www.opensciencegrid.org>
39. Palankar, M.R., Iamnitchi, A., Ripeanu, M., Garfinkel, S.: Amazon s3 for science grids: a viable solution? In: International workshop on Data-aware distributed computing (2008)
40. Pegasus workflow management system. <http://pegasus.isi.edu>
41. Raicu, I., Zhao, Y., Dumitrescu, C., Foster, I., Wilde, M.: Falcon: a fast and light-weight task execution framework. In: Proceedings of the 2007 ACM/IEEE conference on Supercomputing (2007)
42. Sapuntzakis, C., Brumley, D., Chandra, R., Zeldovich, N., Chow, J., Lam, M., Rosenblum, M.: Virtual appliances for deploying and maintaining software. In: Proceedings of the 17th USENIX conference on System administration (2003)
43. Schmuck, F., Haskin, R.: Gpfs: A shared-disk file system for large computing clusters. In: Proceedings of the 1st USENIX Conference on File and Storage Technologies (2002)
44. San diego supercomputing center (sdsc). <http://www.sdsc.edu>
45. Singh, G., Kesselman, C., Deelman, E.: Performance impact of resource provisioning on workflows. Tech. rep., University of Southern California, Information Sciences Institute (2005)

46. Singh, G., Kesselman, C., Deelman, E.: A provisioning model and its comparison with best-effort for performance-cost optimization in grids. In: Proceedings of the 16th international symposium on High performance distributed computing (HPDC '07) (2007)
47. Sotomayor, B., Childers, L.: Globus toolkit 4 programming Java services. Elsevier; Morgan Kaufmann (2006)
48. Teragrid. <http://www.teragrid.org/>
49. Youseff, L., Seymour, K., You, H., Dongarra, J., Wolski, R.: The impact of paravirtualized memory hierarchy on linear algebra computational kernels and software. In: Proceedings of the 17th international symposium on High performance distributed computing (2008)
50. Yu, W., Vetter, J.S.: Xen-based hpc: A parallel i/o perspective. In: Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid (CCGrid '08) (2008)

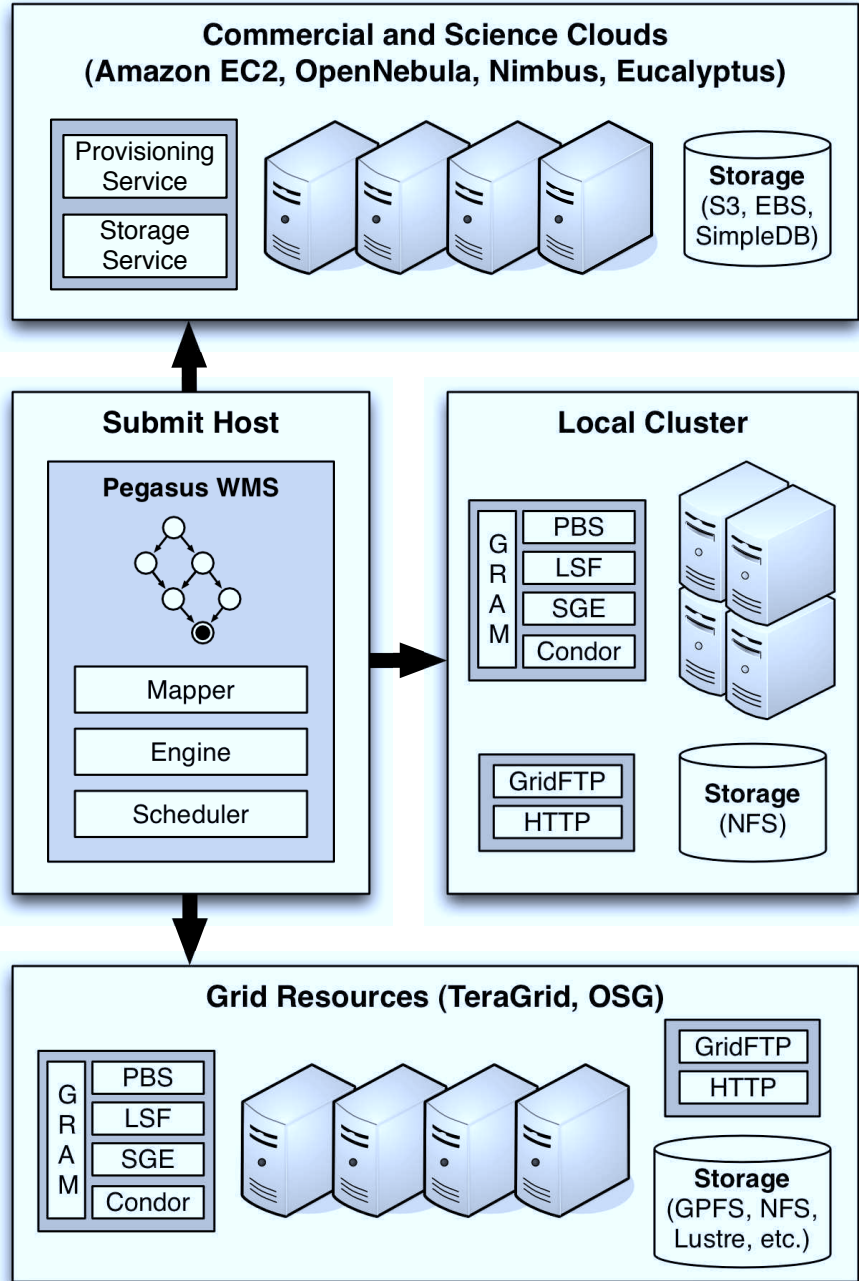


Fig. 1: The Workflow Management in the Context of the Execution Environment.

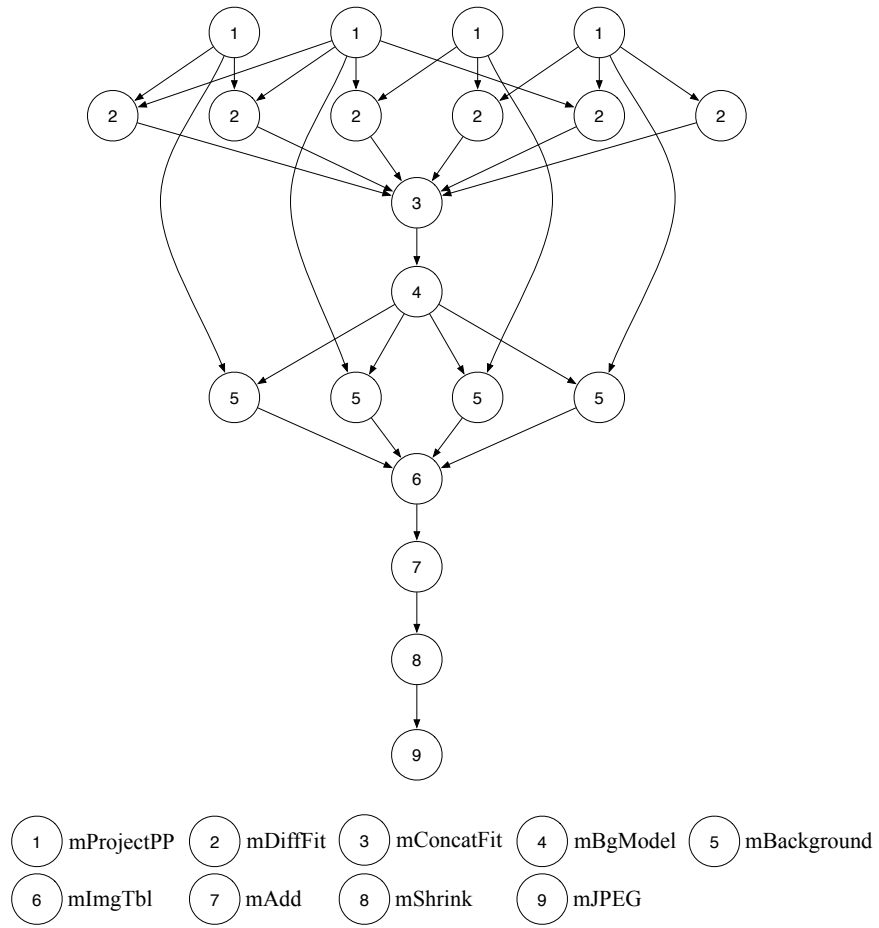


Fig. 2: Montage workflow.

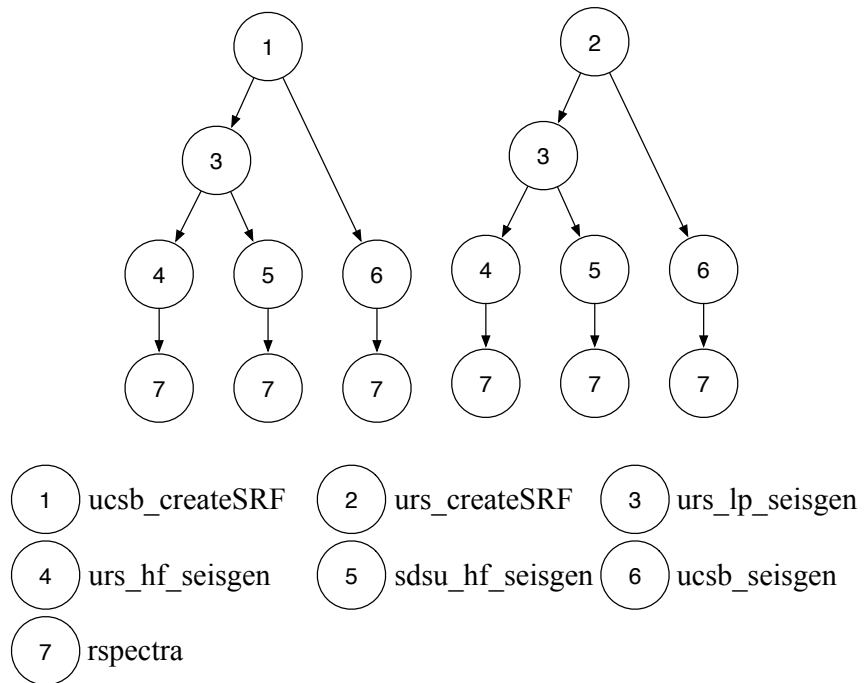


Fig. 3: Broadband Workflow.

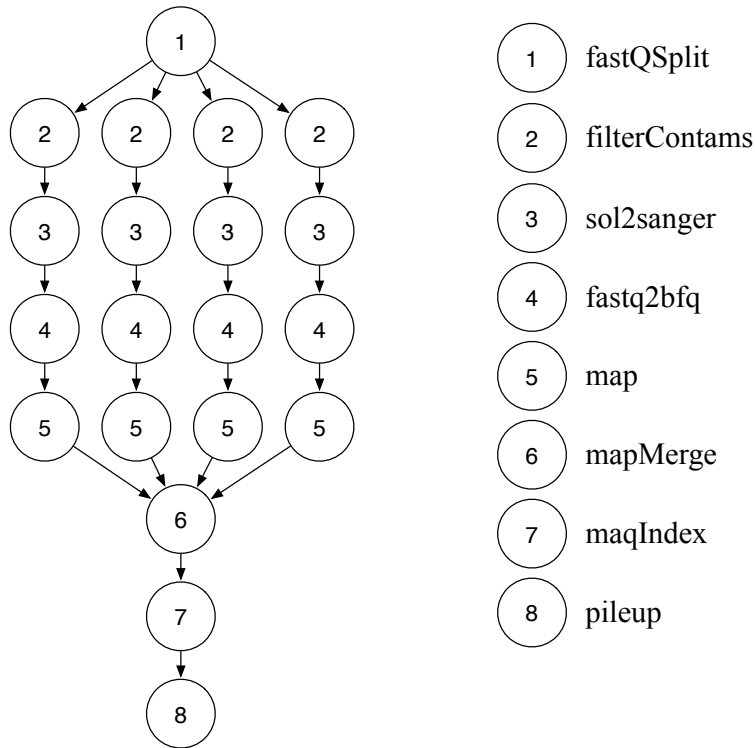


Fig. 4: Epigenome Workflow.

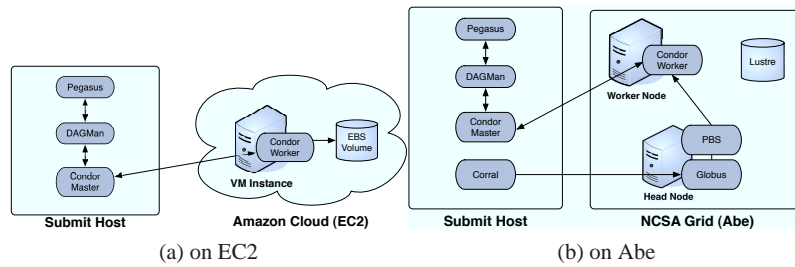


Fig. 5: Execution Environment

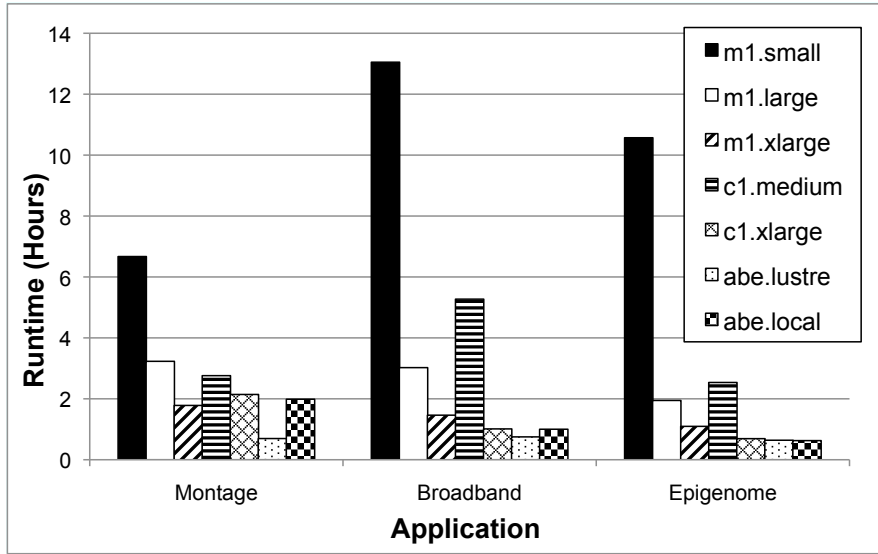


Fig. 6: Runtime Comparison.

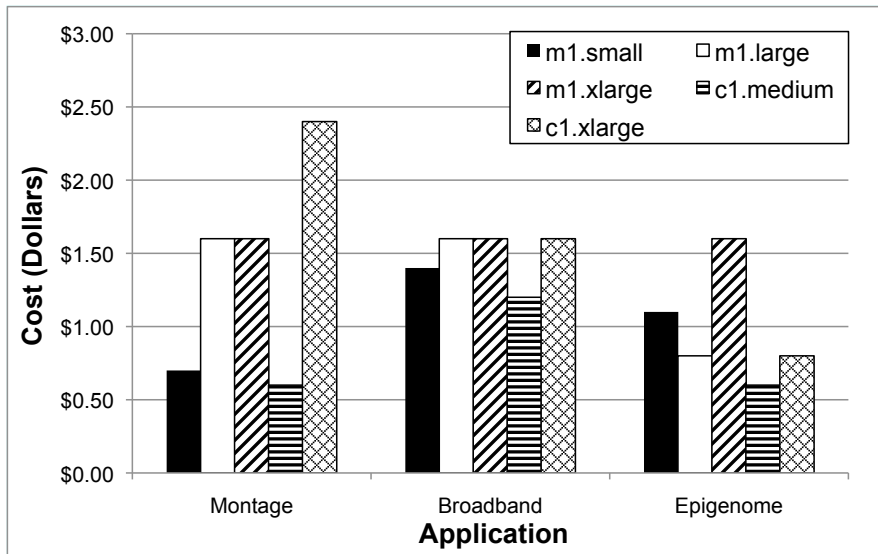


Fig. 7: Resource Cost Comparison.