The Cost of Doing Science on the Cloud: The Montage Example

Ewa Deelman[1], Gurmeet Singh[1], Miron Livny[2], Bruce Berriman[3], John Good[4]
[1] *USC Information Sciences Institute, Marina del Rey, CA*
[2] *University of Wisconsin Madison, Madison, WI*
[3] *Infrared Processing and Analysis Center & Michelson Science Center, California Institute of Technology, Pasadena, CA*
[4] *Infrared Processing and Analysis Center, California Institute of Technology, Pasadena, CA*

Abstract

Utility grids such as the Amazon EC2 cloud and Amazon S3 offer computational and storage resources that can be used on-demand for a fee by compute and data-intensive applications. The cost of running an application on such a cloud depends on the compute, storage and communication resources it will provision and consume. Different execution plans of the same application may result in significantly different costs. Using the Amazon cloud fee structure and a real-life astronomy application, we study via simulation the cost performance tradeoffs of different execution and resource provisioning plans. We also study these trade-offs in the context of the storage and communication fees of Amazon S3 when used for long-term application data archival. Our results show that by provisioning the right amount of storage and compute resources, cost can be significantly reduced with no significant impact on application performance.

1. Introduction

Over the years the research community has developed a wide spectrum of funding and usage models to address the ever growing need for processing, storage, and network resources. From locally owned clusters to national centers and from campus grids to national grids, researchers combine campus and federal funding with competitive and opportunistic compute time allocations to support their science. In some cases, research projects are using their own clusters or pool their resources with other communities (for example in the Open Science Grid (OSG) [1]), or they apply for compute cycles on the national and international cyberinfrastructure resources such as those of the TeraGrid [2] or the EGEE project [3]. Each of these solutions requires a different level of financial commitment and delivers different levels of service. When a project purchases a cluster, this cluster may be expensive but it is fully dedicated to the needs of the project. When joining the OSG, a project contributes some of their resources to the overall collaboration while being able to tap into the capabilities provided by other members of the community. The resource provider still has control over their own resources and may decide on how to share them with others. Providers can also potentially gain the capacity contributed by other members of the collaboration. This system works on the principle that not all the resources are needed at the same time, and when a project does not need their own resources, these cycles are made available to others in the broader collaboration.

Another model of computing is delivered by the TeraGrid, which is a national-level effort to provide a large-scale computational platform for science. Instead of funding individual clusters for individual science projects, it pools together the financial resources of the National Science Foundation to deliver high-performance computing to a broad range of applications. Research projects can apply for allocations of compute cycles that allow them to execute jobs on particular clusters or across the TeraGrid resources. However, the quality of service is not routinely guaranteed on the TeraGrid. Although reservations [4], and "urgent computing" [5] are becoming available, an application may not be able to obtain the necessary resources when they are needed (for example, advance reservations generally require one week advance notice).

A new dimension to the research computing landscape is added by the cloud computing business model [6]. Based on the economy of scale and advanced web and networking technologies, cloud operators such as Amazon [7] and Google [8] aim to offer researchers as many resources as they need when they need them for as long as they need them. Cloud providers charge applications for the use of their resources according to a fee structure. In addition to supporting on-demand computing, clouds, which use virtualization technologies, enable applications to set up and deploy

a custom virtual environment suitable for a given application. Cloud-based outsourcing of computing may be attractive to science applications because it can potentially lower the costs of purchasing, operating, maintaining, and periodically upgrading a local computing infrastructure.

In this paper we ask the question: given the availability of clouds, how can an application use them in a way that strikes the right balance between cost and performance. In particular we examine the cost of running on the cloud in the context of an astronomy application Montage [9], which delivers science-grade mosaics of the sky to the community composed of both professional and amateur astronomers. We want to find out what it would mean for a project such as Montage to rely on the cloud, such as the one provided by Amazon [7] to: 1) handle sporadic overloads of mosaic requests, 2) provide resources for all its computations, and 3) support both computation and long-term data storage. Finally we also ask a domain question: how much would it costs to compute the mosaic of the entire sky on the cloud.

The rest of the paper is organized as follows: Section 2 describes the application that motivated this work, Section 3 describes the Amazon computational model we use for the experiments. Section 4 refines the goals of this work, Sections 5 and 6 give an overview of the simulator used in the studies and present the results. Related work is shown in Section 7. Section 8 concludes the paper.

2. Motivating Application

Montage is a general engine for computing mosaics of input images [9]. The input images for the mosaics are taken from image archives such as the Two Micron All Sky Survey (2MASS) [10], Sloan Digital Sky Survey (SDSS) [11], and the Digitized Sky Surveys at the Space Telescope Science Institute (STScI, http://www.stsci.edu/resources/). A service, hosted at the Infrared Processing and Analysis Center (http://www.ipac.caltech.edu/) provides Montage-based mosaics on demand. The input to the service is the region of the sky whose mosaic is desired, the size of the mosaic in terms of square degrees, and other parameters such as the image archive to be used etc [12]. The input images are first reprojected to the coordinate space of the output mosaic, the reprojected images are then background rectified and finally coadded to create the final output mosaic. Figure 1 shows the structure of a small montage workflow. The tasks in the workflow are depicted by the vertices in the graph and the edges represent the data dependencies between the tasks in the workflow. The numbers in the vertices represent the level of the task in the workflow. The tasks that are not data dependent on other tasks are designed level one. The level of any other task is one plus the maximum level of any of its parent tasks. For the montage workflow, all the tasks at a particular level are invocations of the same routine operating on different input data. For example, the tasks at level one are invocations of the routine mProject which reprojects an input image to the scale defined in an input file called the template header file. This header file is used by all the tasks at level one. The reprojected images produced by the tasks at level one are further processed by the tasks at level two as indicated by the edges between the tasks at the two levels.

Montage is a data-intensive application. The input images, the intermediate files produced during the execution of the workflow and the output mosaic are of considerable size and require significant storage resources. The tasks on the other hand have a small runtime of at most a few minutes. Section 6.3 quantifies the communication to computation ratio of the montage workflow. As a result of these characteristics, it is desirable to run the montage application in a resource rich environment where the availability of storage resources can be assured.

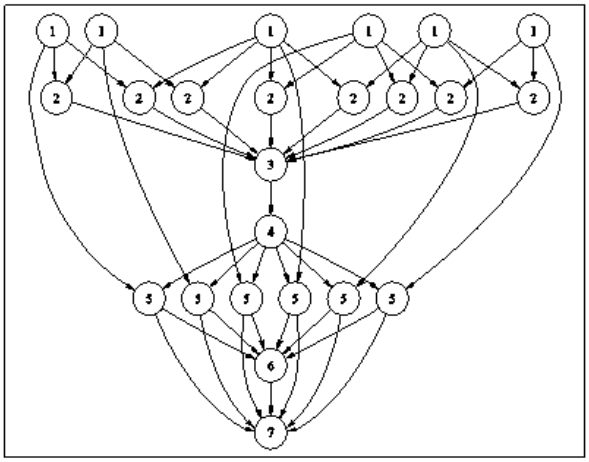

Figure 1. Montage workflow.

The Montage mosaic engine (montage.ipac.caltech.edu) was funded by NASA's Earth Sciences Technology Office, and is maintained by IRSA. By design, the engine preserves the calibration and astrometric fidelity of the input images, rectifies background radiation to a common level

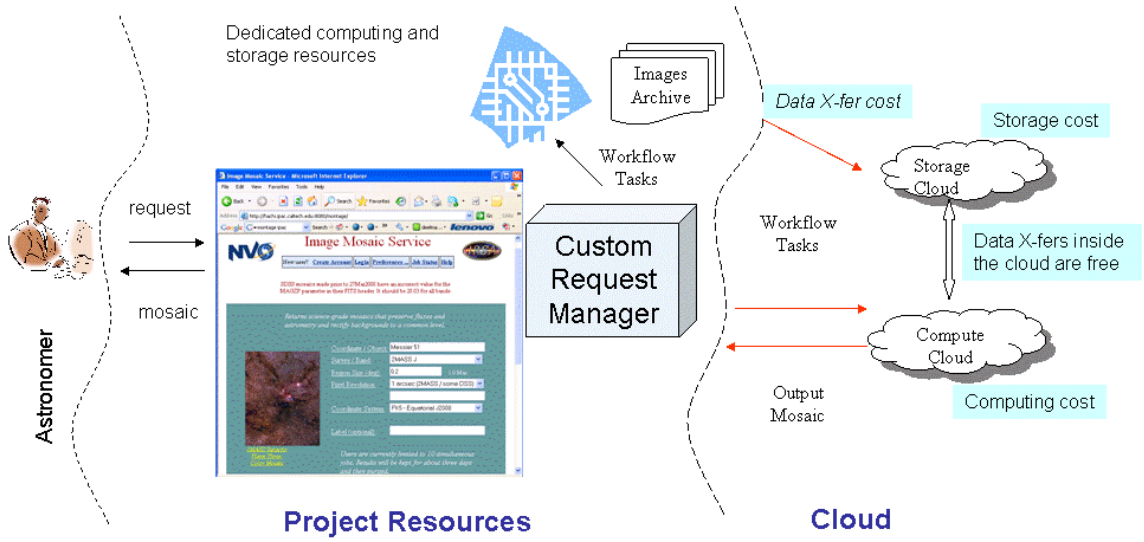

Figure 2. Cloud Computing for a Science Application such as Montage.

across the mosaic, supports all World Coordinate System (WCS) projections, and is portable across all common *nix platforms. The same code can be run on desktops, clusters, grids and supercomputers. There have been over 300 downloads of the source code through a click-wrap license issued by Caltech.

3. Computational and Cost Models

We picked the Amazon services [7] as the basic model. Amazon provides both compute and storage resources on a pay-per-use basis. In addition it also charges for transferring data into the storage resources and out of it. As of the writing of this paper, the charging rates were:

- $0.15 per GB-Month for storage resources
- $0.1 per GB for transferring data into its storage system
- $0.16 per GB for transferring data out of its storage system
- $0.1 per CPU-hour for the use of its compute resources.

There is no charge for accessing data stored on its storage systems by tasks running on its compute resources. Even though as shown above, some of the quantities span over hours and months, in our experiments we normalized the costs on a per second basis. Obviously, service providers charge based on hourly or monthly usage, but here we assume cost per second. The cost per second corresponds to the case where there are many analyses conducted over time and thus resources are fully utilized.

In this paper, we use the following terms: application—the entity that provides a service to the community (the Montage project), user request—a mosaic requested by the user from the application, the cloud—the computing/storage resource used by the application to deliver the mosaic requested by the user.

Figure 2 illustrates the concept of cloud computing as could be implemented for the use by an application. The user submits a request to the application, in the case of Montage via a portal. Based on the request, the application generates a workflow that has to be executed using either local or cloud computing resources. The request manager may decide which resources to use. A workflow management system, such as Pegasus [13], orchestrates the transfer of input data from image archives to the cloud storage resources using appropriate transfer mechanisms (the Amazon S3 storage resource supports the REST and HTTP transfer protocol [14]). Then, compute resources are acquired and the workflow tasks are executed over them. These tasks can use the cloud storage for storing temporary files. At the end of the workflow, the workflow system transfers the final output from the cloud storage resource to a user-accessible location.

While the above gives a high-level description of the overall process, we present three different implementation models that correspond to different execution plans for using the cloud storage resources. In order to explain these computational models we use

the example workflow shown in Figure 3. There are seven tasks in the workflow numbered from 0 to 6. Each task takes one input file and produces one output file except for task 6 that takes three input files.

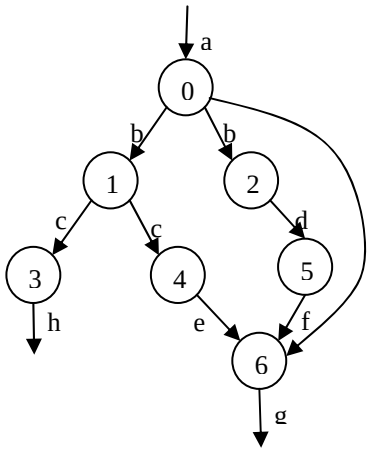

Figure 3. An example workflow.

We explore three different data management models:

- **Remote I/O (on-demand)**: For each task we stage the input data to the resource, execute the task, stage out the output data from the resource and then delete the input and output data from the resource. This is the model to be used when the computational resources used by the tasks have no shared storage. For example, the tasks are running on hosts in a cluster that have only a local file system and no network file system. This is also equivalent to the case where the tasks are doing remote I/O instead of accessing data locally.
- **Regular**: When the compute resources used by the tasks in the workflow have access to shared storage, it can be used to store the intermediate files produced in the workflow. For example, once task 0 (Figure 3) has finished execution and produced the file *b*, we allow the file *b* to remain on the storage system to be used as input later by tasks 1 and 2. In fact, the workflow manager does not delete any files used in the workflow until all the tasks in the workflow have finished execution. After that files *g* and *h* which are the net output of the workflow are staged out to the application/user and then all the files *a – h* are deleted from the storage resource. As mentioned earlier this execution mode assumes that there is shared storage that can be accessed from the compute resources used by the tasks in the workflow. This is true in the case of the Amazon system where the data stored in the S3 storage resources can be accessed from any of the EC2 compute resources.
- **Dynamic cleanup**: In the regular mode, there might be files occupying storage resources even when they have outlived their usefulness. For example file *a* is no longer required after the completion of task 0 in Figure 3 but it is kept around until all the tasks in the workflow have finished execution and the output data is staged out. In the dynamic cleanup mode, we delete files from the storage resource when they are no longer required. This is done by Pegasus by performing an analysis of data use at the workflow level [15]. Thus file *a* would be deleted after task 0 has completed, however file *b* would be deleted only when task 6 has completed. Thus the dynamic cleanup mode reduces the storage used during the workflow and thus saves money. Previously, we have quantified the improvement in the workflow data footprint when dynamic cleanup is used for data-intensive applications similar to Montage [16]. We found that dynamic cleanup can reduce the amount of storage needed by a workflow by almost 50%.

4. Study Goals

The main focus of the paper is to examine the tradeoffs of different execution and resource provisioning plans for providing science services to a community using cloud computing. We pose several questions related to the main aim:

Question 1: Assume that an application has a set of resources available to them but sometimes it needs more resources than it has, so it reaches out to the cloud from time to time to meet the additional demands. In this case, the application will provision a set of resources from the cloud and bring in the data and stage the results back to a location where the user can access it. The question is how many processors to provision in order to optimize application performance while minimizing the monetary cost.

Question 2: Assume that an application has very limited computational resources and wants to rely on the cloud resources to provide the necessary computing power. Also assume that the application provisions a certain amount of resources over a period of time to sustain the expected computational load. That set of resources requested is assumed to be larger then the needs of any single computation. Thus the requests can run at their full level of parallelism. Here the cost is measured only as the cost of the resources used by a single request. We assume that the

application would incur the cost of the resources over time and would need to decide how much to charge the user for a given request. In this case we are only calculating the cost of this request to the application and not the premium the application may decide to charge on top of it.

Here we also examine two situations:
Question 2a: Given that the application has a local data archive and just wants to farm out the computing, the question is, how much each user request will cost?

Question 2b: Assume that the application relies fully on both the compute and storage services on the cloud. Here besides the cost of a particular request, we also determine how many requests it would take to make the cost of storing the data on the cloud worthwhile. The issue is that if you have only a few requests, the cost of storing large amounts of data over time can be prohibitively expensive, so for a small number of requests, it may be cheaper to stage data to the cloud on demand.

Question 3: Finally, we answer a totally application-focused questions: 1) how much money it would cost to generate the mosaic of the entire sky? This sky mosaic based on 2Mass data can be created by combining 3,900 plates (mosaics) in three frequency bands, each of 4 degrees square; 2) if one calculates a mosaic, how long does it make financial sense to store it on the cloud rather than recomputed it on demand.

5. Simulator Description

In order to answer the questions raised in the previous section, we performed simulations. No actual provisioning of resources from the Amazon system was done. Simulations allowed us to evaluate the sensitivity of the execution cost to workflow characteristics such as the communication to computation ratio by artificially changing the data set sizes. This would have been difficult to do in a real setting. Additionally, simulations allow us to explore the performance/cost tradeoffs without paying for the actual Amazon resources or incurring the time costs of running the actual computation. The simulations were done using the GridSim toolkit [17]. Certain custom modifications were done to perform accounting of the storage used during the workflow execution.

We used three Montage workflows in our simulations:

1. Montage 1 Degree: A Montage workflow for creating a 1 degree square mosaic of the M17 region of the sky. The workflow consists of 203 application tasks.
2. Montage 2 Degree: A Montage workflow for creating a 2 degree square mosaic of the M17 region of the sky. The workflow consists of 731 application tasks.
3. Montage 4 Degree: A Montage workflow for creating a 4 degree square mosaic of the M17 region of the sky. The workflow consists of 3,027 application tasks.

These workflows can be created using the mDAG component in the Montage distribution. The workflows created are in XML format. We wrote a program for parsing the workflow description and creating an adjacency list representation of the graph as an input to the simulator. The workflow description also includes the names of all the input and output files used and produced in the workflow. The sizes of these data files and the runtime of the tasks were taken from real runs of the workflow and provided as additional input to the simulator.

We simulated a single compute resource in the system with the number of processors greater than the maximum parallelism of the workflow being simulated. The compute resource had an associated storage system with infinite capacity. The bandwidth between the user and the storage resource was fixed at 10 Mbps. Initially all the input data for the workflow are co-located with the application. At the end of the workflow the resulting mosaic is staged out to the application/user and the simulation completes. The metrics of interest that we determine from the simulation are:

1. The workflow execution time.
2. The total amount of data transferred from the user to the storage resource.
3. The total amount of data transferred from the storage resource to the user.
4. The storage used at the resource in terms of GB-hours. This is done by creating a curve that shows the amount of storage used at the resource with the passage of time and then calculating the area under the curve.

6. Results

Question 1: Cost of running sporadic computations on the cloud.

Here we examine how best to use the cloud for individual mosaic requests. We calculate how much would a particular computation cost on the cloud,

given that the application provisions a certain number of processors and uses them for executing the tasks in the application. We explore the execution costs as a function of the number of resources requested for a given application. The processors are provisioned for as long as it takes for the workflow to complete. We vary the number of processors provisioned from 1 to 128 in a geometric progression. We compare the CPU cost, storage cost, transfer cost, and total cost as the number of processors is varied. In our simulations we do not include the cost of setting up a virtual machine on the cloud or tearing it down, this would be an additional constant cost.

Montage 1 Degree Square

The Montage 1 degree square workflow consists of 203 tasks. Figure 4 shows the execution costs for this workflow. The most dominant factor in the total cost is the CPU cost. The data transfer costs are independent of the number of processors provisioned. The Figure shows that the storage costs are negligible as compared to the other costs. The Y-axis is drawn in logarithmic scale to make the storage costs discernable. As the number of processors is increased, the storage costs decline but the CPU costs increase. The storage cost declines because as we increase the number of processors, we need them for shorter duration since we can get more work done in parallel. Thus we also need storage for shorter duration and hence the storage cost declines. However, the increase in the CPU cost far outweighs any decrease in the storage costs and as a result the total costs also increase with the increase in the number of provisioned processors. The graph shows the storage costs with (Storage Costs (C)) and without cleanup (Storage Costs) as described in Section 3. The storage costs with cleanup are slightly less than the storage costs with cleanup. The total costs shown in the Figure are computed using the storage costs without cleanup. The total cost with cleanup is very similar and virtually indistinguishable in the figure if drawn. The total costs shown in the graphs are aggregated costs for all the resources used.

Based on Figure 4, it would seem that provisioning the least amount of processors is the best choice, at least from the point of view of monetary costs (60 cents for the 1 processor computation versus almost 4$ with 128 processors). However, the drawback in this case is the increased execution time of the workflow. Figure 4 (bottom) shows the execution time of the Montage 1 Degree workflow with increasing number of processors. As the Figure shows, when only one processor is provisioned leading to the least total cost, it also leads to the longest execution time of 5.5 hours. The runtime on 128 processors is only 18 minutes. Thus a user who is also concerned about the execution time, faces a trade-off between minimizing the execution cost and minimizing the execution time.

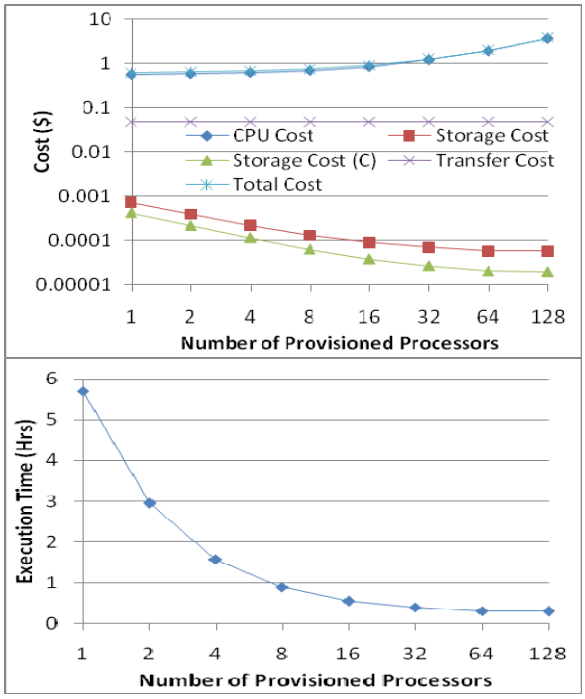

Figure 4. Execution Costs and Execution time for Montage 1 Degree Workflow.

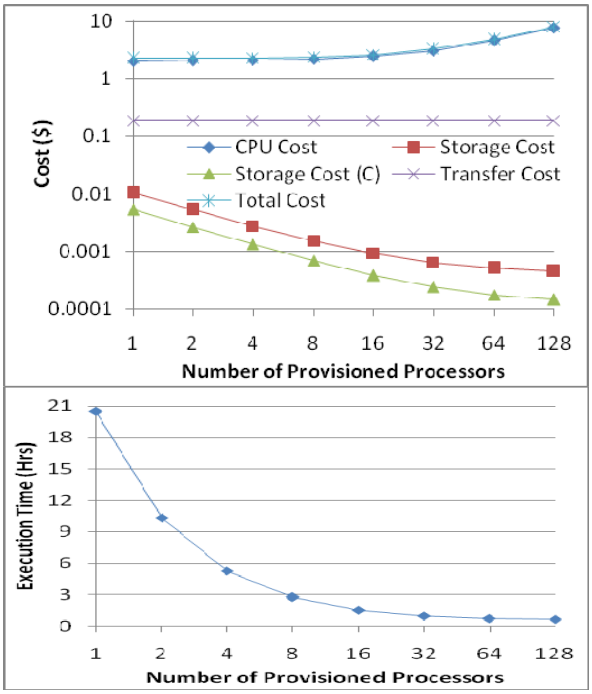

Figure 5. Execution Costs and Execution time of Montage 2 Degree Workflow.

Montage 2 Degree Workflow

Figure 5 shows similar results for the Montage 2 degree workflow as for the Montage 1 degree workflow. The total cost is an increasing function of the number of the allocated processors while the execution time is a decreasing function of the number of allocated processors. The Montage 2 degree workflow consists of 731 tasks. At the extremes, the cost of running the workflow on 1 processor is $2.25 with a runtime of 20.5 hours whereas running the same workflow on 128 processors results in a runtime of less than 40 minutes with a cost of less than $8.

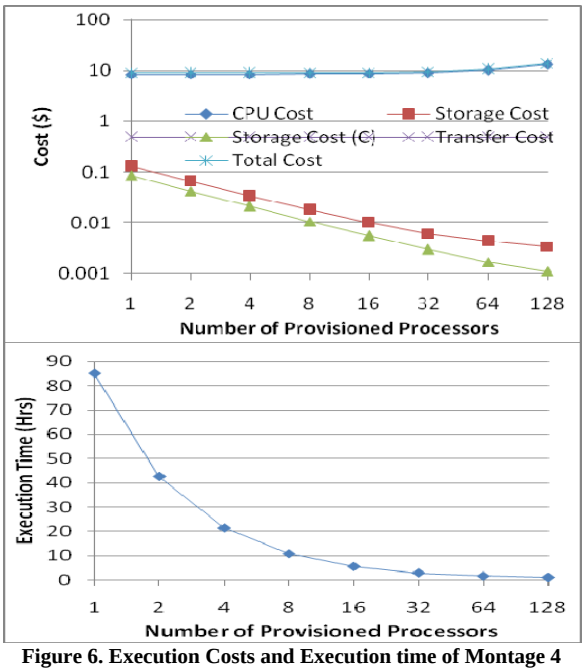

Figure 6. Execution Costs and Execution time of Montage 4 Degree Workflow.

Montage 4 Degree Workflow

Figure 6 show similar results for the Montage 4 degree workflow as for the Montage 2 degree and Montage 1 degree workflow. The Montage 4 degree square workflow consists of 3,027 application tasks in total. In this case running on 1 processor costs $9 with a runtime of 85 hours; with 128 processors, the runtime decreases to 1 hour with a cost of almost $14. Although the monetary costs do not seem high, if one would like to request many mosaics to be done, as would be in the case of providing a service to the community, these costs can be significant. For example, providing 500 4-degree square mosaics to astronomers would cost $4,500 using 1 processor versus $7,000 using 128 processors. However, the turnaround of 85 hours may be too much to take by a user. Luckily, one does not need to consider only the extreme cases. If the application provisions 16 processors for the requests, the turnaround time for each will be approximately 5.5 hours with a cost of $9.25, and thus a total cost of 500 mosaics would be $4,625, not much more than in the 1 processor case, while giving a relatively reasonable turnaround time.

Question 2a: Cost of relying on the cloud for all computing needs

Here we examine the issue of the cost of user requests for scientific products when the application provisions a large number of resources from the cloud and then allows the request to use as many resources as it needs. The application is in this scenario responsible for scheduling the user requests onto the provisioned resources. In this case, since the processor time is used only as much as needed, we would expect that the data transfer and data storage costs may play a more significant role in the overall request cost. As a result, we examine the tradeoffs between using three different data management solutions: 1) remote I/O, where tasks access data as needed, 2) regular, where the data are brought in at the beginning of the computation and they and all the results are kept for the duration of the workflow, and 3) cleanup, where data no longer needed are deleted as the workflow progresses. In the following experiments we want to determine the relationship between the data transfer cost and the data storage cost and compare it to the overall execution cost.

Figure 7 (top) shows the amount of storage used by the workflow in the three modes in space-time units. The least storage used is in the remote I/O mode since the files are present on the resource only during the execution of the current task. The most storage is used in the regular mode since all the input data transferred and the output data generated during the execution of the workflow is kept on the storage until the last task in the workflow finishes execution. Cleanup reduces the amount of storage used in the regular mode by deleting files when they are no longer required by later tasks in the workflow.

Figure 7 (middle) shows the amount of data transfer involved in the three execution modes. Clearly the most data transfer happens in the remote I/O mode since we transfer all input files and transfer all output files for each task in the workflow. This means that if the same file is being used by more than on job in the workflow in the remote I/O mode the file may be transferred in multiple times whereas in the case of regular and cleanup modes, the file would be transferred only once.

The amount of data transfer in the Regular and the Cleanup mode are the same since dynamically removing data at the execution site does not affect the data transfers. We have categorized the data transfers into data transferred to the resource and data transferred out of the resource since Amazon has different charging rates for each as mentioned in Section 3. As the figure shows, the amount of data transferred out of the resource is the same in the Regular and Cleanup modes. The data transferred out is the data of interest to the user (the final mosaic in case of Montage) and it is staged out to the user location. In the Remote I/O mode intermediate data products that are needed for subsequent computations but are not of interest to the user also need to be stage-out to the user-location for future access. As a result, in that mode the amount of data being transferred out is larger than in the other two execution strategies.

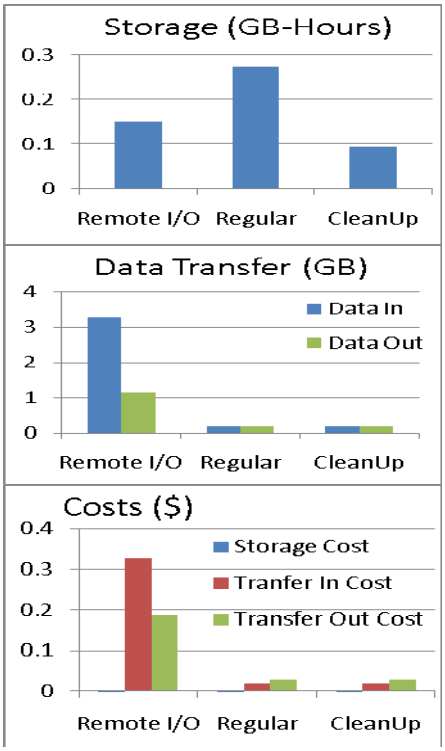

Figure 7. Data Management Metrics for the Montage 1 degree Workflow.

Figure 7 (bottom) shows the costs (in monetary units) associated with the execution of the workflow in the three modes and the total cost in each mode. The storage costs are negligible as compared to the data transfer costs and hence are not visible in the figure. The Remote I/O mode has the highest total cost due to its higher data transfer costs. Finally, the Cleanup mode has the least total cost among the three. It is important to note that these results are based on the charging rates currently used by Amazon. If the storage charges were higher and transfer costs were lower, it is possible that the Remote I/O mode would have resulted in the least total cost of the three.

Figure 8 and Figure 9 show the metrics for the Montage 2 and 4 degrees workflow respectively. The cost distributions are similar for all the workflows and differ only in magnitude as can be seen from the figures.

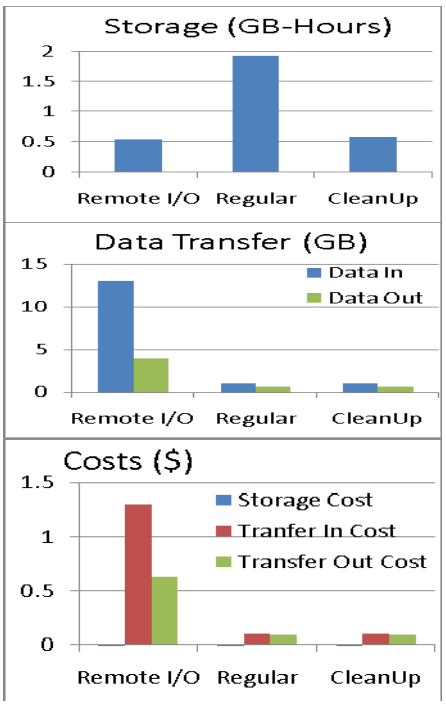

Figure 8. Data Management Metrics for the Montage 2 degree Workflow.

The total cost shown in Figures 7-9 does not include the CPU cost of running the workflow tasks on Amazon EC2 resources. Figure 10 compares the CPU cost of these workflows with the other costs shown in earlier figures (aggregated and shown as DM, Data Management costs in Figure 10). As the figure shows, the CPU cost is slightly higher than the data management costs for the remote I/O execution mode. The CPU cost is invariant between the three execution modes (Remote I/O, Regular, CleanUp) shown in the earlier figures.

In these experiments we ignore limitations on the granularity of Amazon fee structure in time and assume the least possible granularity i.e $ per Byte-

seconds for storage, $ per Bytes for transfers and $ per CPU-second for compute resources.

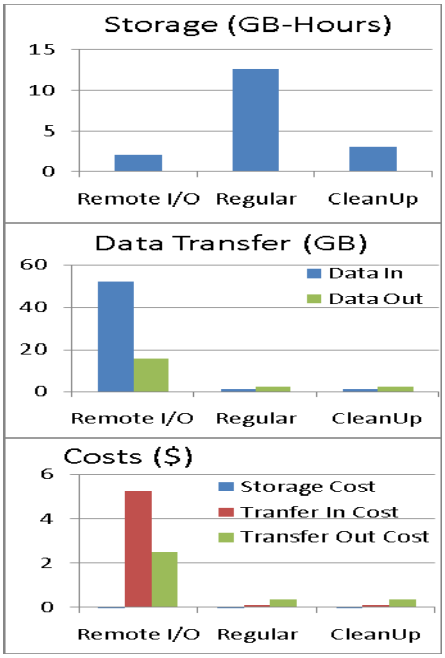

Figure 9. Data Management Metrics for the Montage 4 degree Workflow.

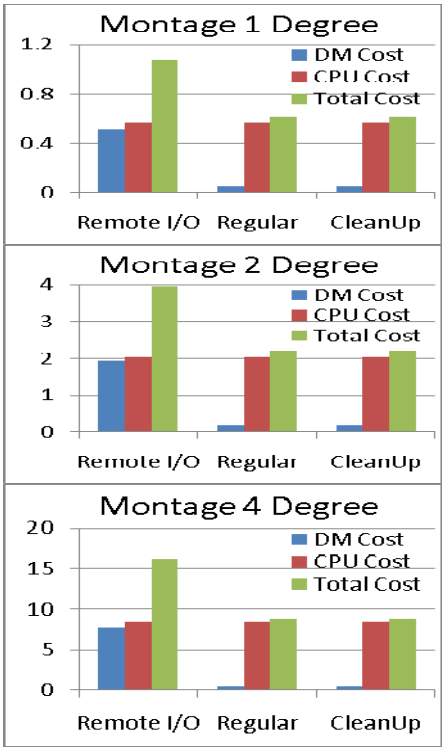

Figure 10. The CPU and other costs of the execution of Montage Workflows.

If we compare these results with the costs we observed in the case of provisioning a fixed amount of resources for the duration of the workflow request (Question 1), we see that these costs are significantly different. For example, the cost of running the 4 degree square Montage workflow on 128 processors is $13.92 in the provisioned case, whereas the workflow which is charged only for the resources used is only $8.89. The maximum parallelism of that workflow is 610. This shows that CPU utilization can be low in the provisioned case.

Impact of the Communication to Computation Ratio on the Cost of the Request

Obviously, Montage is only one of a number of scientific applications that can potentially benefit from cloud services. Here we also explored the costs of applications that would have different communication (data) to computation ratios (CCR). The CCR of a workflow is defined as follows. Let $F = \{f_1, f_2,\ldots,f_k\}$ be the set of files used or produced in the workflow and let $s(f_i)$ denotes the size of file i in bytes. Let $V = \{v_1, v_2, \ldots, v_n\}$ be the set of task in the workflow and let $r(v_i)$ denotes the runtime of task v_i in seconds on a standard reference CPU. Let B be reference bandwidth in bytes per second. Then the CCR of a workflow (V, F) is

$$CCR = \frac{\sum_{f \in F} s(f) \Big/ B}{\sum_{v \in V} r(v)}$$

The CCR of the Montage workflow computed by this equation and based on a bandwidth B of 10 Mbps is shown in the table below.

Workflow	CCR
Montage 1 Degree	0.053
Montage 2 Degree	0.053
Montage 4 Degree	0.045

For the set of experiments described in this section, we change the CCR of the Montage workflows by appropriately scaling the file sizes in the workflow. For example, let CCR_d be the desired CCR and CCR_r be the real CCR of the workflow. Then we multiply each file size by CCR_d/CCR_r to get the desired CCR.

Figure 11 shows the execution costs for the Montage 1 Degree workflow with changing CCR. For these experiments, we provision 8 processors for as long as required to execute the workflow. 8 processors were chosen since they represent a reasonable compromise

between the execution cost and execution time as seen in Section 6.1.

As the CCR increases, we see that the storage costs, both with and without cleanup increase. The transfer costs which were constant earlier also increase due to the increase in the size of the data. The workflow execution time increases since it takes longer to stage in the input data. The CPU cost also increase due to increase in the execution time of the workflow. As a result, the total cost is also an increasing function of the CCR.

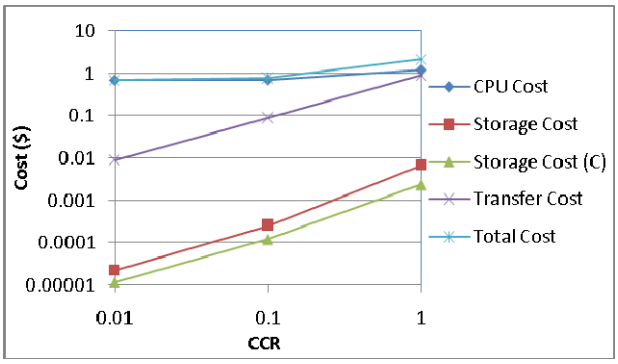

Figure 11. Execution costs of Montage 1 Degree workflow with changing CCR.

The results for the Montage 2 and 4 degree workflows are similar to the 1 degree workflow and hence not shown here. From these experiments we can see that the transfer and storage costs increase in proportion to the increase in CCR or even higher (for the storage costs). Thus it seems that it may be beneficial to pre-store all the input data in the cloud in order to reduce the transfer costs as the applications become more data-intensive. In the next section, we examine the use of the cloud for data archival purposes.

Question 2b: Cost of running and storing data on the cloud

In this section we answer the question of benefit of relying on the cloud to do computing and to store the large datasets required to do science. In the case of Montage, one of important datasets is the 2Mass data (2 Micron all sky survey http://www.ipac.caltech.edu/2mass/releases/allsky/), which contains images of the entire sky in three different bands. The size of entire data set is 12 Terabytes. If we were to store the entire collection on the cloud, the application would benefit from low data access latencies (for input data) and the cost of these accesses would be zero. However, the cost of storing the data can be significant and equal to 12,000 × $0.15 = $1,800 per month. The cost of producing a 2 degree square mosaic when the input data are already available in the cloud is $2.12 which includes a $2.03 CPU cost (Figure 10) and $0.09 data management cost as storage charges for the temporary files during the execution run and the transfer charges for transferring the final mosaic to the user. The cost of the mosaic that has to bring in the data from outside the cloud is $2.22 (Figure 10). Thus in order to be able to overcome the storage costs, users would need to request at least $1,800/($2.22-$2.12) = 18,000 mosaics per month. These figures to do not include the initial cost of transferring the data to the cloud, which would be an additional $1,200 at ($0.1 per GB). This cost would only be incurred once and would need to be amortized over time. A possibly better solution is to pre-stage some popular data sets. This would require application developers to analyze their requests patterns and where possible discover trends.

Question 3: Cost of large-scale science on the cloud

In this section, we examine the cost of creating the mosaic of the entire sky. This can be done by making a complete set of mosaics covering every region of the sky (with some overlap). Roughly it would translate to about 3,900 4-degree-square mosaics or about 1,734 6-degrees-square mosaics.

The cost of creating a 4 degrees square mosaic in regular mode was $8.88 (Figure 10). Thus the total cost would be 3,900 x $8.88 = $34,632. If we assume that the input data is already archived in the cloud, then the execution cost of the 4 square degree mosaic is $8.75 leading to a total cost of 3,900 x $8.75 = $34,145.

Another interesting question is whether it makes economic sense to archive the generated popular mosaics in the cloud instead of always generating them on demand from the basic input data. For the 1 degree Montage workflow the CPU cost was 56 cents (Figure 10). This cost can be totally saved by just storing the mosaic which had a size of (173.46 MB) and serving it again when another similar request is received. For the cost of 56 cents, this mosaic can be stored for 21.52 months assuming storage charges of $0.15 per GB-month. Similarly the size of the 2 square degree mosaic is 557.9 MB and the CPU cost for creating it was $2.03 cents. For this cost, the mosaic can be stored for 24.25 months. Similarly the size of the 4 square degree mosaic is about 2.229 GB and the CPU cost for creating it is $8.40. At this cost, the mosaic can be stored for 25.12 months. Thus in summary, if it is likely that the same request would be repeated with the

next two years, then it would make economic sense to store the generated mosaic instead of recomputing it. Therefore, it would be cost effective to save popular mosaics of the sky (areas such as those around Orion for example) in the cloud.

7. Related Work

There has been many proposals for Grid systems that operate using market mechanisms such as Grace [18], Spawn [19], Tycoon [20] among others. However, Amazon Web Services is among the first providers that have made computational and storage resources commercially available on pay per use basis on a production level. IBM has a cloud computing initiative underway called Blue Cloud [21]. There are other storage providers that cater to niche markets such as Nirvanix [22] that optimizes storage for media files. Some recent work have focused mainly on the performance issues related with these services [23]. There has not been much work on the classification and quantification of the execution costs of scientific applications on these systems.

Cost-aware execution of workflow structured applications have been addressed previously [24, 25]. The model in [24] however, is a futures-based resource market model whereas in the case of Amazon services, all the resources are available for immediate occupancy and there is no concept of advance reservations [25]. Previously, we have explored the performance cost tradeoff while provisioning resources for workflows [26]. However, in that study we did not take storage resources into consideration. Cost-based scheduling of scientific applications has also been addressed in [27]. In [27], the model is of service providers that undertake to execute individual tasks in the application at different prices. In our case we allocate computational resources from providers. Individual tasks are executed by transferring the executable program to the compute resource and then invoking it using the proper arguments. Thus, resource providers are agnostic to the tasks being executed on their resources and only charge for the occupancy of their resources.

The question of what resources to provision has also been investigated in earlier works. In [28] the optimal size of the resource request to make is considered in order to minimize the workflow completion time. The cost of the resources however is not taken into consideration. In a number of earlier works [29, 30] the size of the resource request is optimized so that the sum of the wait time to get the requested resource and the run time of the application on the resource is minimized. With the advent of cloud computing that provides resources on demand these issues become irrelevant as there is no wait time involved to get the resources. It is possible that as the demand for the cloud resources increases, it might increase beyond supply and the resource providers would then have to deal with admission control issues [31, 32]. In these cases, when there are advance purchase discounts, the completion time of the workflow would be an important criterion to consider.

8. Conclusions

Cloud computing offers a new business model for supporting computations and provides a new option for scientific applications to have on-demand access to potentially significant amounts of compute and storage resources. Using the Montage application and the Amazon EC2 fee structure as a case study, we showed that for a data-intensive application with a small computational granularity, the storage costs were insignificant as compared to the CPU costs. Thus it appears that cloud computing offers a cost-effective solution for data-intensive applications.

Clouds are still in their infancy--there are only a few commercial [7, 21, 22] and academic providers [33]. As the field matures, we expect to see a more diverse selection of fees and quality of service guarantees for the different resources and services provided by clouds. It is possible that some providers will have a cheaper rate for compute resources while others will have a cheaper rate for storage and provide a range of quality of service guarantees. As a result, applications will have more options to consider and more execution and provisioning plans to develop to address their computational needs.

In this paper, we explored only one aspect of using cloud computing for science, examining the tradeoffs of different workflow execution modes and provisioning plans for cloud resources. Many other aspects of the problem still need to be addressed. These include the startup cost of the application on the cloud, which is composed of launching and configuring a virtual machine and its teardown, as well as the often one-time cost of building a virtual image suitable for deployment on the cloud. The complexity of such an image depends on the complexity of the application. We also did not explore other cloud issues such as security and data privacy. The reliability and availability of the storage and compute resources are also an important concern. According to Amazon

sources, the targeted availability of the S3 storage system is 99.9% [34] which is also verified by independent studies [23]. However, when the system goes down, as it did twice in the first 7 months of 2008, the possible impact on the applications can be significant. Due to the mainly commercial nature of cloud computing, there are expectations and penalties resulting from any violation of the user-provider contract are clearly spelled out [34]. These and other issues such as scalability of the new computing paradigm are still open questions.

Acknowledgments

This work was funded by the National Science Foundation under Cooperative Agreement OCI-0438712 and grant # CCF-0725332. This research made use of Montage, funded by NASA's Earth Science Technology Office, Computation Technologies Project, under Cooperative Agreement Number NCC5-626 between NASA and the California Institute of Technology. Montage is maintained by the NASA/IPAC Infrared Science Archive.

References

[1] "Open Science Grid,"www.opensciencegrid.org.
[2] "TeraGrid,"http://www.teragrid.org/.
[3] "Enabling Grids for E-sciencE (EGEE),"http://www.eu-egee.org/.
[4] "TeraGrid Resource Reservations," www.teragrid.org/userinfo/resource_reservation.php.
[5] "Special Priority and Urgent Computing Environment," http://spruce.teragrid.org.
[6] G. Lawton, "Moving the OS to the Web," *Computer*, vol. 41, pp. 16-19, 2008.
[7] "Amazon Web Services," http://aws.amazon.com,http://aws.amazon.com.
[8] "Google App Engine,"http://code.google.com/appengine/.
[9] "Montage Project," http://montage.ipac.caltech.edu.
[10] The Two Micron All Sky Survey., "http://www.ipac.caltech.edu/2mass," -+*.
[11] Sloan Digital Sky Survey., "http://www.sdss.org/."
[12] "Image Mosaic Service," http://hachi.ipac.caltech.edu:8080/montage/.
[13] E. Deelman, G. Mehta, et al., "Pegasus: Mapping Large-Scale Workflows to Distributed Resources," in *Workflows in e-Science*, I. Taylor, E. Deelman, et al., Eds.: Springer, 2006.
[14] "REST vs SOAP at Amazon," http://www.oreillynet.com/pub/wlg/3005?wlg=yes.
[15] A. Ramakrishnan, G. Singh, et al., "Scheduling Data - Intensive Workflows onto Storage-Constrained Distributed Resources," in *CCGrid* 2007.
[16] G. Singh, K. Vahi, et al., "Optimizing workflow data footprint," *Scientific Programming*, vol. 15, pp. 249-268, 2007.
[17] R. Buyya and M. Murshed, "GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing," *Concurrency and Computation: Practice and Experience*, vol. 14, pp. 1175-1220, 2002.
[18] R. Buyya, D. Abramson, et al., "The grid economy," *Proceedings of the IEEE*, vol. 93, pp. 698-714, 2005.
[19] C. A. Waldspurger, T. Hogg, et al., "Spawn: a distributed computational economy," *IEEE Transactions on Software Engineering*, vol. 18, pp. 103-117, 1992.
[20] K. Lai, B. A. Huberman, et al., "Tycoon: a Distributed Market-based Resource Allocation System," Technical Report, Hewlett-Packard Laboratories, Palo Alto, CA, September2004.
[21] "IBM Blue Cloud," http://www-03.ibm.com/press/us/en/pressrelease/22613.wss.
[22] "Nirvanix," http://www.nirvanix.com.
[23] M. Palankar, A. Onibokun, et al., "Amazon S3 for Science Grids: a Viable Solution," in *4th USENIX Symposium on Networked Systems Design & Implementation (NSDI'07)*, 2007.
[24] G. Singh, C. Kesselman, et al., "A Provisioning Model and its Comparison with Best-Effort for Performance-Cost Optimization in Grids," in *HPDC* 2007, pp. 117-126.
[25] H. Zhao and R. Sakellariou, "Advance Reservation Policies for Workflows," in *12th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, Saint-Malo, France, 2006.
[26] G. Singh, C. Kesselman, et al., "Performance Impact of Resource Provisioning on Workflows," USC http://www.cs.usc.edu/Research/TechReports/05-850.pdf 05-850, 2005.
[27] J. Yu, R. Buyya, et al., "Cost-based Scheduling of Scientific Workflow Applications on Utility Grids," in *1st IEEE International Conference on e-Science and Grid Computing*, 2005.
[28] R. Y. Huang, H. Casanova, et al., "Automatic Resource Specification Generation for Resource Selection," in *Super Computing Conference*, Reno, 2007.
[29] W. Cirne and F. Berman, "Using Moldability to Improve the Performance of Supercomputer Jobs," *JPDCg*, vol. 62, pp. 1571-1601, 2002/10 2002.
[30] A. B. Downey, "Using Queue Time Predictions for Processor Allocation " in *Proceedings of the Job Scheduling Strategies for Parallel Processing* Springer-Verlag, 1997 pp. 35-57
[31] G. Singh, C. Kesselman, et al., "Adaptive Pricing for Resource Reservations in Shared Environments," *Grid* 2007.
[32] A. Sulistio, K. H. Kim, et al., "Using Revenue Management to Determine Pricing of Reservations," in *e-Science and Grid Computing, IEEE International Conference on*, 2007, pp. 396-405.
[33] "Nimbus Science Cloud," http://workspace.globus.org/clouds/nimbus.html.
[34] "Amazon SLA," http://www.amazon.com/gp/browse.html?node=379654011.