Data Placement for Scientific Applications in Distributed Environments

Ann Chervenak[1], Ewa Deelman[1], Miron Livny[2], Mei-Hui Su[1], Rob Schuler[1], Shishir Bharathi[1], Gaurang Mehta[1], Karan Vahi[1]

[1]*USC Information Sciences Institute*
Marina Del Rey, CA 90292
{annc,deelman,mei,schuler,shishir,gmehta,vahi}@isi.edu
[2]*Computer Science Department, University of Wisconsin Madison*
Madison, WI 53706-1685
miron@cs.wisc.edu

***Abstract*— Scientific applications often perform complex computational analyses that consume and produce large data sets. We are concerned with data placement policies that distribute data in ways that are advantageous for application execution, for example, by placing data sets so that they may be staged into or out of computations efficiently or by replicating them for improved performance and reliability. In particular, we propose to study the relationship between data placement services and workflow management systems. In this paper, we explore the interactions between two services used in large-scale science today. We evaluate the benefits of prestaging data using the Data Replication Service versus using the native data stage-in mechanisms of the Pegasus workflow management system. We use the astronomy application, Montage, for our experiments and modify it to study the effect of input data size on the benefits of data prestaging. As the size of input data sets increases, prestaging using a data placement service can significantly improve the performance of the overall analysis.**

I. Introduction

For data-intensive scientific applications running in a distributed environment, the placement of data onto storage systems can have a significant impact on the performance of scientific computations and on the reliability and availability of data sets. These scientific applications may produce and process terabytes or petabytes of data stored in millions of files or objects, and they may run complex computational workflows consisting of millions of interdependent tasks, including jobs that stage data in and out of storage systems adjacent to computational resources. A variety of data placement algorithms could be used, depending on the goals of the scientific collaboration, or *Virtual Organization* (VO) [17]. For example, a placement algorithm might have the goal of placing data as close as possible to the computational nodes that will execute a scientific analysis, or it might try to store replicas of every data item on multiple storage systems to avoid data loss if any storage system or disk fails. These goals might be considered *policies* of the VO, and a *policy-driven data placement service* is responsible for replicating and distributing data items in conformance with these policies or preferences.

In our previous work we examined issues of reliable data placement [20] where the focus was on utilizing multiple data transfer protocols to deliver data to their destination. We also examined the issues of data management in the context of an individual workflow [14], focusing on the interplay of the data management and computation management components within a single analysis.

In this paper, we are concerned with data placement policies that distribute data in a way that is advantageous for application or workflow execution, for example, by placing data sets near high-performance computing resources so that they can be staged into computations efficiently; by moving data off computational resources quickly when computation is complete; and by replicating data sets for performance and reliability. Effective data placement policies of this type might benefit from knowledge about available resources and their current performance and capacity. Placement services could also make use of hints or information about applications and their access patterns, for example, whether a set of files is likely to be accessed together and therefore should be replicated together on storage systems.

We propose to study the relationship between data placement services and workflow management systems. In particular, our goal is to separate to the extent possible the activities of data placement and workflow execution, so that placement of data items can be largely *asynchronous* with respect to workflow execution, meaning that data placement operations are performed as data sets become available and according to the policies of the Virtual Organization, independently of the actions of the workflow management system. This is in contrast to many current workflow systems, which are responsible for explicitly staging data onto computational nodes before execution can begin. While some explicit data staging may still be required by workflow engines, intelligent data placement on appropriate nodes has the potential to significantly reduce the need for on-demand data placement and to improve workflow execution times.

Based on their knowledge of applications and of expected data access patterns, workflow management systems can provide hints to data placement services regarding the placement of data, including information about the size of data sets required for a workflow execution and collections of data items that are likely to be accessed together. A data placement service can also get hints from the workflow system or from information systems about the availability of computational and storage resources. In addition, the Virtual Organization can provide hints to the placement service on preferred resources or on other policies that might affect the placement of data. Together, these hints will provide important clues regarding where computations are likely to run and where data sets should be stored so that they can be easily accessed during workflow execution.

In this paper, we use a simple prototype system that integrates the workflow management functionality of the Pegasus system [15, 16] with a data movement service called the Data Replication Service [10]. Using these existing services for data and workflow management in distributed computing environments, our goal is to demonstrate that asynchronous placement of data has the potential to significantly improve the performance of scientific workflows.

The paper is organized as follows. First, we describe two existing data placement services for large scientific collaborations and discuss how they might benefit from cooperation between placement services and workflow systems. Second, we describe data placement alternatives for scientific environments in more detail. Third, we describe two existing systems used in large-scale science: Pegasus and the Data Replication Service. We discuss the interplay of functionality that allows these two systems to cooperate to improve overall workflow performance. Next, we present performance results for a Montage application, which generates science-grade mosaics of the sky. These results compare the performance of a workflow that uses asynchronous data placement with one that does on-demand data placement. They demonstrate that such data placement can significantly improve the performance of workflows with large input data sets. We conclude with related work and a discussion of our future plans for work on the integration of data placement and workflow management.

II. Data Placement For Large Scientific Applications

Large scientific collaborations have developed complex systems for management of data distribution and replication. Existing data placement systems include the Physics Experiment Data Export (PheDEx) [6, 32] system for high-energy physics and the Lightweight Data Replicator (LDR) [24] for gravitational-wave physics. In both collaborations, the Virtual Organization has developed policies to distribute and replicate data sets widely so that they will be available to scientists at their individual institutions or near where computations are likely to run and so that the system will provide a high degree of availability.

In parallel with these data placement services, computational workflows have emerged as an important paradigm for large-scale computing. Many sciences are turning to workflow management systems to provide a framework for coupling community codes and applications into large-scale analysis. Until now, data placement and computation placement services have been developed largely independently. However, we argue that these functionalities should not be developed in isolation. Rather, we believe that the next step beyond data distribution and replication is to study the relationship between data placement and workflow management. Our techniques could add value to existing placement services such as PheDEx and LDR by providing additional hints on where data sets should be placed to be used effectively by workflow engines.

For the remainder of this section, we describe the functionality of the PheDEx and LDR systems.

The high energy physics scientific community includes several experiments that will make use of terabytes of data collected from the Large Hadron Collider (LHC) [13] at CERN. The data sets generated by these experiments will be distributed to many sites where scientists need access to the raw and processed data products. The high energy physics community has a hierarchical or tiered model for data distribution [6]. At the Tier 0 level at CERN, the data will be collected, pre-processed and archived. From there, data products will be replicated and disseminated to multiple sites. Tier 1 sites are typically large national computing centers that will have significant storage resources and will store and archive large subsets of the data produced at the Tier 0 site. At the next level, Tier 2 sites will have less storage available and will store a smaller subset of the data. At lower levels of the Tier system, smaller subsets of data are stored and made accessible to scientists at their individual institutions.

The PheDEx system manages data distribution for the Compact Muon Solenoid (CMS) high energy physics experiment [13]. The goal of the system is to automate data distribution processes as much as possible. Data operations that must be performed include staging data from tape storage (for example, at the Tier 0 site) into disk buffers; wide area data transfers (to Tier 1 or Tier 2 sites); validation of these transfers; and migration from the destination disk buffer into archival storage at those sites. The PheDEx system design involves a collection of agents or daemons running at each site, where each agent performs a unique task. For example, there are agents that stage data from tape to disk and agents that perform data transfers. The agents communicate through a central database running on a multi-server Oracle database platform. The PheDEx data distribution system supports three use cases for CMS. First, it supports the initial “push-based” hierarchical distribution from the Tier 0 site at CERN to the Tier 1 sites. It also supports subscription-based transfer of data, where sites or scientists subscribe to data sets of interest, and those data sets are sent to the requesting sites as they are produced. Finally, PheDEx supports on-demand access to data by individual sites or scientists.

The Lightweight Data Replicator (LDR) system [24] distributes data for the Laser Interferometer Gravitational Wave Observatory (LIGO) project [4, 23]. LIGO produces large amounts of data and distributes or places it at LIGO sites based on metadata queries by scientists at those sites. Currently, the collaboration stores more than 120 million files across ten locations. Experimental data sets are initially produced at two LIGO instrument sites and archived at CalTech; they are then replicated at other LIGO sites to provide scientists with local access to data. LIGO researchers developed the LDR system to manage the data distribution process. LDR is built on top of standard Grid data services such as the Globus Replica Location Service [9, 11] and the GridFTP data transport protocol [5]. LDR provides a rich set of data management functionality, including replicating necessary files to a LIGO site. Each LDR site initiates local data transfers using a pull model. A scheduling daemon queries the site's local metadata catalog to request sets of files with specified metadata attributes. These sets of files are called *collections*, and each collection has a priority level that determines the order in which files from different collections will be transferred to the local site. For each file in a collection, the scheduling daemon checks whether the desired file already exists on the local storage system. If not, the daemon adds that file's logical name to a priority-based scheduling queue. Each LDR site also runs a transfer daemon that initiates data transfers of files on the scheduling queue in priority order. The transfer daemon queries replica catalogs to find locations in the Grid where the desired file exists and randomly chooses among the available locations. Then the transfer daemon initiates data transfer operations from the remote site to the local site using the GridFTP data transport protocol.

While the PheDEx and LDR systems provide sophisticated data placement and replication for their communities, these systems could benefit from greater interplay with computational schedulers such as workflow management systems. Hints from workflow systems could potentially influence the placement of data and significantly improve the performance of computational analyses.

III. Classes of Data Placement Services

In this section, we present a broad view of how data placement services could be utilized by scientific applications. There are three broad categories of placement algorithms: those that seek to stage data efficiently into computations; those concerned with staging data out of computational resources; and algorithms designed to provide data reliability and durability.

The first class of data placement algorithms is concerned with staging data *into* computation analyses efficiently. In a large workflow composed of thousands of interdependent tasks, each task that is allocated for execution on a computational node requires that its input files be available to that node before computation can begin. Specific characteristics of data access during workflow execution also need to be taken into account by the data placement service. For example, data items tend to be accessed as related collections rather than individually, and a placement service would ideally place items in a collection together on a storage system to facilitate execution. In addition, data access patterns tend to be bursty, with many data placement operations taking place during the stage in (or stage out) phase of execution. In this paper, we focus primarily on placement services that move data sets asynchronously onto storage systems accessible to computational nodes, ideally before workflow execution begins so that workflow tasks do not need to wait for data transfer operations to complete. This type of data placement is illustrated in Fig. 1. Other placement algorithms try to schedule jobs on or near nodes where data sets already exist, as discussed in Section VII.A.

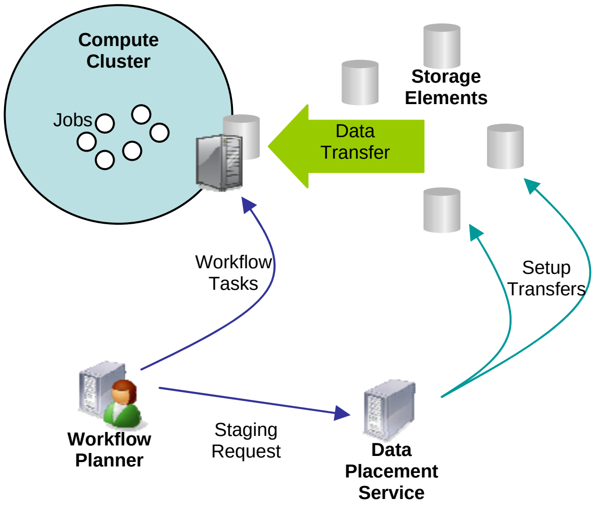

Fig. 1: Workflow planner sends data staging requests to a data placement service, which initiates data transfer operations from storage elements in the distributed environment to the storage system associated with compute nodes on which the workflow tasks will execute.

In practice, staging data *out* of computational resources efficiently may also be a significant challenge for scientific applications. When these applications run large analyses on distributed resources (e.g., on the Open Science Grid [2], which provides a number of diverse distributed resources to scientific collaborations), the individual nodes that run computational jobs may have limited storage capacity. When a job completes, the output of the job may need to be staged off the computational node onto another storage system before a new job can run at that node. Thus, a data placement service that is responsible for moving data efficiently off computational nodes can have a large impact on the performance of scientific workflows.

A third set of data placement algorithms is concerned with the maintenance of data to provide high availability or durability, i.e., protection from data failures. These placement algorithms replicate data to maintain additional copies to protect against temporary or permanent failures of storage systems. For example, a placement service of this type might create a new replica of a data item whenever the number of accessible replicas falls below a certain threshold. These replication algorithms might be reactive to failures or might proactively create replicas. Several of these algorithms are described in Section VII.C.

IV. The Pegasus Workflow Management System

Pegasus, which stands for Planning for Execution in Grids [15, 16], is a framework that maps complex scientific workflows onto distributed resources such as the TeraGrid [3], the Open Science Grid, and others. Pegasus relies on the Condor DAGMan [18] workflow engine to launch workflow tasks and maintain the dependencies between them. Pegasus enables scientists to construct workflows in abstract terms without worrying about the details of the underlying cyberinfrastructure or the particulars of the low-level specifications required by the cyberinfrastructure middleware (Globus[19] and Condor [25]). As part of the mapping, Pegasus automatically manages data generated during workflow execution by staging them out to user-specified locations, by registering them in data catalogs, and by capturing their provenance information.

Sometimes workflows, as structured by scientists, are not tuned for performance. Additionally, given that at the time of the workflow generation, the eventual execution resources are not known, it is impossible to optimize the runtime of the overall workflow. Since Pegasus dynamically discovers the available resources and their characteristics, and queries for the location of the data (potentially replicated in the environment), it improves the performance of applications through: data reuse to avoid duplicate computations and to provide reliability, workflow restructuring to improve resource allocation, and automated task and data transfer scheduling to improve overall workflow runtime. Pegasus also provides reliability through dynamic workflow remapping when failures during execution are detected.

Currently, Pegasus schedules all the data movements in conjunction with computations. However, as the new data placement services are being deployed within the large-scale collaborations, workflow management systems such as Pegasus need to be able to interface and efficiently interact with the new capabilities.

V. THE DATA REPLICATION SERVICE

For asynchronous data placement, we use the Data Replication Service (DRS) [10]. The function of the DRS is to replicate a specified set of files onto a storage system and register the new files in appropriate replica catalogs. DRS builds on lower-level Grid data services, including the Globus Reliable File Transfer (RFT) service, which provides reliable multi-file transfer requests, and the Replica Location Service (RLS), a distributed registry for replicated data items. The operations of the DRS include *discovery*, identifying where desired data files exist on the Grid by querying the RLS; *transfer*, copying the desired data files to the local storage system efficiently using the RFT service; and *registration*, adding location mappings to the RLS so that other sites may discover newly created replicas. Throughout DRS replication operations, the service maintains state about each file, including which operations on the file have succeeded or failed. Fig. 2 illustrates the basic operation of the Data Replication Service.

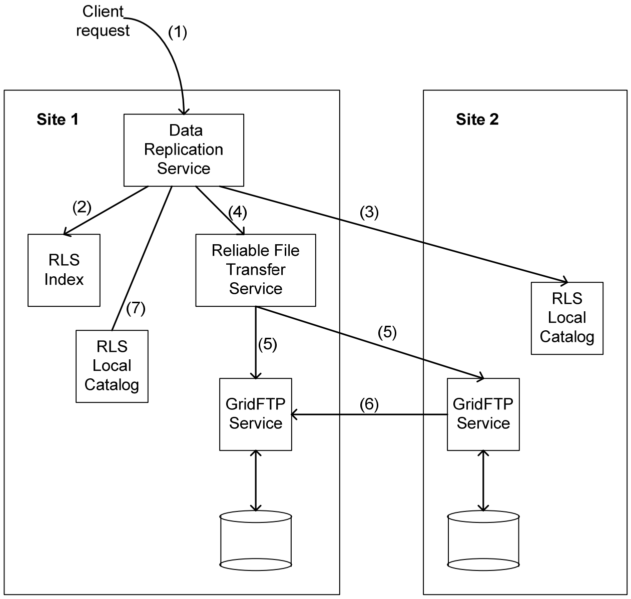

Fig. 2 Operation of the Data Replication Service

When a client request for a data replication operation arrives (1), the Data Replication Service first queries a Replica Location Service Index node to determine the location(s) of the requested files in the Grid. The RLS Index returns a pointer to an RLS local catalog at site 2, and the DRS next queries that catalog (3) to determine the physical location of the desired file(s). Next, the DRS issues a file transfer request to the RFT Service (4), which initiates a third-party transfer operation (5) between GridFTP servers at the source and destination sites. After the file transfer operation is complete (6), the DRS registers the new replica in its local RLS catalog (7).

VI. WORKFLOW PERFORMANCE USING ASYNCHRONOUS DATA PLACEMENT

We combined the functionality of the Data Replication Service (DRS) for data placement with that of the Pegasus system for workflow management. The goal is to demonstrate that data-intensive workflows may execute faster with such asynchronous data placement than with on-demand staging of data by the workflow management system.

Our data placement is performed based on an explicit knowledge of which files will be used during the workflow execution. We issue requests to DRS to move these files to a storage system associated with the cluster where workflow execution will take place. This data movement takes place asynchronously with respect to the execution of the workflow.

When the Pegasus workflow management system is launched, it detects the existence of these data sets by querying a replica catalog. If the data items are available on the storage system associated with the computational cluster where the workflow will run, then Pegasus accesses the data sets via symbolic links to that storage system. Thus, Pegasus avoids explicitly staging the data onto computational resources at run time.

A. *The Montage Workflow*

For these experiments, we used the workflow for the Montage astronomy application. Montage [1, 7] is an application that constructs custom science-grade astronomical image mosaics on demand. Montage is used as a key component of eleven projects and surveys worldwide to generate science and browse products for dissemination to the astronomy community. Fig. 3 shows the structure of a small Montage workflow. The figure only shows the graph of the resource-independent abstract workflow. The executable workflow will contain data transfer and data registration nodes in addition to those shown in the figure.

The levels of the workflow represent the depth of a node in a workflow determined through a breadth-first traversal of the directed graph. At each level of the workflow, a different set of executables that performs specific image processing functions is invoked. The inputs to the workflow include the input images in standard FITS format (a file format used throughout the astronomy community), and a "template header file" that specifies the mosaic to be constructed. The workflow can be thought of as having three parts, including reprojection of each input image to the coordinate space of the output mosaic, background rectification of the reprojected images, and coaddition to form the final output mosaic.

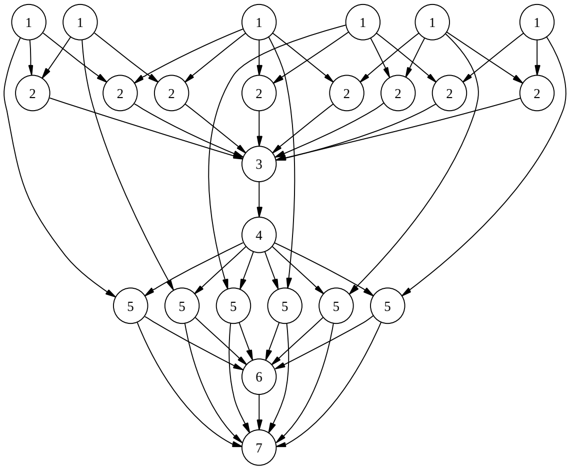

Fig. 3: A small Montage workflow.

To experiment with data-intensive workflows, we varied the sizes of the Montage input data set. In addition to default input file sizes in the range of Kbytes, we staged in an additional input file for each job in the first level of the Montage workflow. We set the size of these additional files to 2 Megabytes and to 20 Megabytes in two different experiments. We expected that the performance results would show greater benefits for more data-intensive workflow executions.

B. *Performance Comparison*

The workflows for our experiments ran on a cluster with up to 50 available compute nodes, where each node is a dual Pentium III 1GHz processor with 1GByte of RAM running the Debian Sarge 3.1 operating system. The data sets are staged onto a storage system associated with the cluster from a GridFTP server on the local area network.

In our experiments, we compared the time it takes to asynchronously prestage data versus letting the Pegasus workflow management system stage the data explicitly as part of the workflow execution. The graphs below show the time to stage data using the Data Replication Service; the running time of the workflow when the data are already prestaged by DRS, requiring no additional data movement by Pegasus; and the running of the workflow when Pegasus manages the data staging explicitly. In order to facilitate comparisons, we also show the sum of the DRS data staging time and the Pegasus execution time for prestaged data, which would correspond to sequential invocation of these two services.

In our experiments, we modified the data granularity of the Montage workflow from the default size. Our hypothesis was that asynchronous data placement would be more advantageous for workflows that were data-intensive. To test this hypothesis, we experimented with three input sizes for the files required by the Montage workflow. Table 1 shows the total number of files used as input to each workflow execution, where additional files are used to simulate more data-intensive workflows.

Table 1 shows the total number of files used as input to each workflow execution, where additional files are used to simulate more data-intensive workflows.

Table 2 shows the total input size for each degree of the Montage workflow that we ran. Although we increase the input data size for the workflow, the computational run time of the workflow remains the same as for the default input size, because the additional input files are ignored for the purposes of computation.

TABLE 1

NUMBER OF INPUT FILES

Degree Square of Montage Mosaic	**Number of input files for workflow execution**		
	Default input size	**With additional 2MB files**	**With additional 20MB files**
1	50	95	95
2	166	318	318
4	648	1258	1258

TABLE 2

TOTAL INPUT SIZE FOR WORKFLOWS

Degree Square of Montage Mosaic	**Total input size for workflow execution**		
	Default	**With additional 2MB files**	**With additional 20MB files**
1	91 MBytes	182 MBytes	993 MBytes
2	307 MBytes	612 Mbytes	3.31 GBytes
4	1.2 GBytes	2.4 GBytes	13.2 GBytes

Fig. 4 through Fig. 6 show the performance of the Montage workflow with three input sizes for mosaic degrees of 1, 2 and

4. The line in each graph labeled “DRS” shows the time for the Data Replication Service to do asynchronous placement of the input files onto the storage system associated with the cluster. The line in each graph labeled “Pegasus with Prestaged Data” shows the execution time for the workflow under Pegasus using the data sets that have been prestaged by DRS. The line labeled “DRS + Pegasus” shows the sum of these times for data staging and execution with pre-staged data, as a measure of the worst-case performance if these two operations occur sequentially. Finally, the line labeled “Pegasus with Data Staging” shows the performance of the Montage workflow when the Pegasus system stages the data as part of the workflow execution. Pegasus stages data through explicit data transfer tasks in the workflow. These tasks are placed by DAGMan (Pegasus’ workflow engine) in the local Condor queue and eventually released for execution. Thus, these data movement tasks incur the overheads of the Condor queuing and scheduling system. As the number of data transfer jobs increases, these overheads also increase, and the execution time of workflows using Pegasus to perform data staging may exceed the execution time for the combination of DRS data staging and Pegasus execution with prestaged data, as can be seen most notably in Fig. 6.

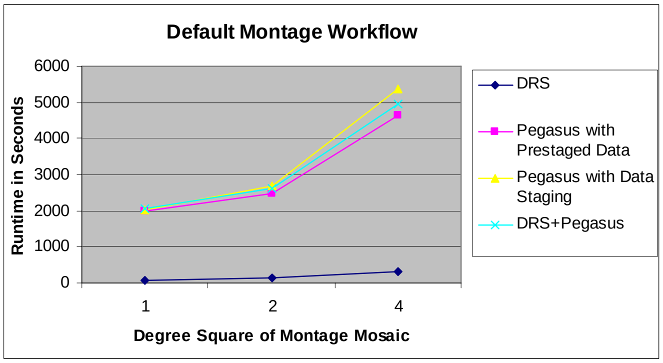

Fig. 4 Shows the execution times for a Montage workflow with the default input size for mosaic degrees of 1, 2 and 4. Different lines show the time for prestaging data with DRS; the execution time for the workflow if data sets have been prestaged; the combined execution time for DRS staging and Pegasus execution; and finally the execution time for a Montage workflow in which Pegasus explicitly stages in data at run time.

Fig. 4 shows the performance for the default input size for the Montage workflow. The graph shows that there is a small advantage for prestaging data for this workflow, where a total of about 1.2 GBytes are prestaged by DRS. Workflow execution time is reduced approximately 8% for the combined DRS staging plus Pegasus execution, compared to the case where Pegasus explicitly stages data.

Fig. 5 also shows a small advantage for prestaging data for the Montage workflow. Here, we increase the total input data size by transferring an additional 2 MByte file into every job at the first level of the Montage workflow. With a mosaic degree of 4, the total input data size is about 2.4 Gbytes, approximately double the size for the experiment in Fig. 4.

As expected, for the most data intensive of the workloads we measured, the advantages of prestaging data using the Data Replication Service are significant, as shown in Fig. 6. For this experiment, we transferred an additional 20 MByte file into every job at the first level of the Montage workflow. With a total input data size of 13.2 GBytes for a mosaic of 4 degree square, the combination of prestaging data with DRS followed by workflow execution using Pegasus improves execution time approximately 21.4% over the performance of Pegasus performing explicit data staging as part of workflow execution. When data sets for this workflow are completely prestaged by DRS before workflow execution begins, the workflow execution time is reduced by over 46%.

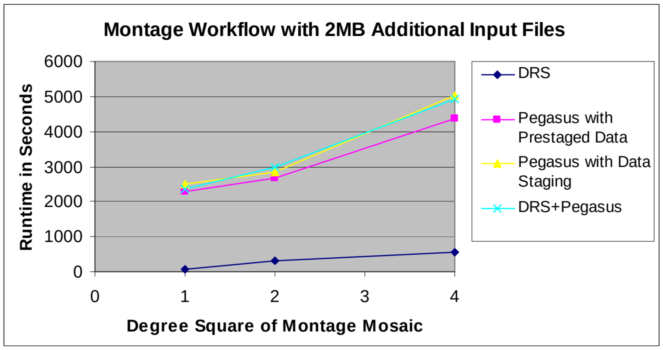

Fig. 5 Shows execution times for a Montage workflow with larger input sizes, where additional 2 MByte files are staged in for each job at the first level of the workflow.

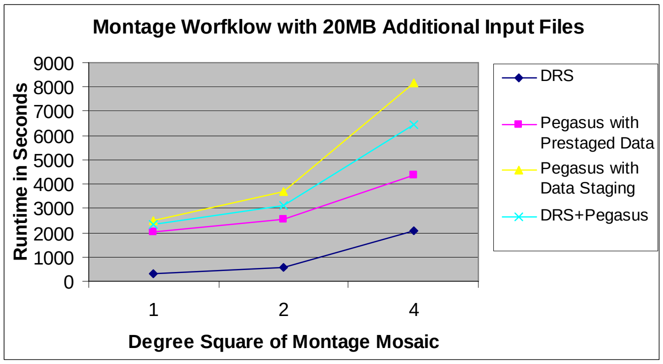

Fig. 6 Shows execution times for a Montage workflow with an additional 20 MByte file staged in for each job run as part of the workflow. For this more data intensive workflow, it is more advantageous to perform asynchronous data placement using DRS.

As described above, additional overheads are incurred when Pegasus performs data staging tasks explicitly as part of its workflow execution. These overheads increase with the number of input files being transferred. Additionally, data-centric services such as DRS can potentially tune the parameters of data transfers to be optimal for particular sets of data, particular sizes of data and network conditions. This type of tuning would be hard to achieve in general-purpose workflow management systems.

The above performance numbers demonstrate the potential advantages of combining data placement services with workflow management systems for data-intensive scientific applications. While these performance studies only address the problem of efficiently prestaging input files for a computation, we also plan to study the problems of staging data out of computational nodes as well as the relationship between data placement for computational efficiency and for availability/reliability.

VII. Related Work

There has been extensive work on data placement and replication for a variety of distributed file systems and distributed storage systems. Here we focus particularly on work related to policy-driven data placement in large, wide-area distributed systems.

A. Data and Computational Scheduling in Grids

Several groups have addressed issues related to the scheduling of data and computations in Grid environments. We have already discussed the PheDEx and LDR systems, which focus primarily on data distribution for high energy and gravitational wave physics applications.

In [29-31] the authors conducted extensive simulation studies that examined the relationship between asynchronous data placement and replication and job scheduling. They examined a variety of combinations of job scheduling and data scheduling. The data scheduling policies keep track of the data set usage and replicate popular datasets. The authors concluded that scheduling jobs where data sets are present is the best algorithm and that actively replicating popular data sets also significantly improves execution time.

One difference between this work and our proposed approach is that the authors assume that the jobs being scheduled are independent of one another. We propose to study the interplay between more complex analyses composed of many interdependent jobs. Thus, we consider costs associated with managing the entire workflow, for example, moving intermediate data products to where the computations will take place and co-allocating possibly many data products so that the workflow can progress efficiently. Additionally, the approach they propose is reactive in the sense that it examines the past popularity of data to make replication decisions, whereas our approach is proactive and examines current workflow needs to make data placement decisions.

B. Workflow Scheduling in the Context of Data Management

Directed Acyclic Graphs (DAGs) are a convenient model to represent workflows, and the extensive literature on DAG scheduling is of relevance to the problem of workflow scheduling [22]. Scheduling scientific workflows in distributed environments has been recently studied in [8, 26, 33, 35-37]. In the majority of these works, the aim is to minimize the workflow execution time, with the assumption that data scheduling is included as part of the computational scheduling decisions.

C. Data Placement and Replication for Durability

Data placement services may also enforce policies that attempt to maintain a certain level of redundancy in the system to provide highly available or durable access to data. For example, a system where data sets are valuable and expensive to regenerate may want to maintain several copies of each data item on different storage systems in the distributed environment. The UK Quantum Chromodynamics Grid (QCDGrid) project [27, 28] is a virtual organization that maintains several redundant copies of data items for reliability. Medical applications that preserve patient records could also benefit from placement services that maintain multiple copies of data items. Such placement services monitor the current state of the distributed system, and if the number of replicas of a data item falls below the threshold specified by V.O. policy, the placement service initiates creation of additional replicas on available storage nodes.

In the Oceanstore global distributed storage system [21], several algorithms have been studied for replication of data to maintain high levels of durability. These include a reactive replication algorithm called *Carbonite* [12] that models replica repair and failure rates in a system as the birth and death rates in a continuous time Markov model. To provide durability, the replication rate must match or exceed the average rate of failures. Carbonite creates a new replica when a failure is detected that decreases the number of replicas below a specified minimum.

The Oceanstore group has also proposed a proactive replication algorithm called Tempo [34] that creates replicas periodically at a fixed low rate. Tempo creates redundant copies of data items as quickly as possible using available maintenance bandwidth and disk capacity. Tempo provides durability for data sets comparable to that from the reactive Carbonite algorithm using a less variable amount of bandwidth, thus helping to keep maintenance costs predictable.

The work presented in this paper does not directly address data placement policies for durability, but we plan to include this functionality in the future.

VIII. Summary and Future Work

We are interested in understanding the relationship between existing data placement services and workflow management systems used today in data-intensive scientific applications. In particular, we believe that by separating to the extent possible the activities of data placement and workflow execution, we can significantly improve the performance of scientific workflows. We presented experimental results for the execution of several astronomy workflows. For each Montage workflow, we compared the execution time using a workflow management system that explicitly stages data into the workflow at runtime with the execution time for a system that uses a data placement service to prestage data sets onto storage resources. Using the Data Replication Service for data placement and the Pegasus workflow management system to execute these workflows, we demonstrated that this separation of data placement and workflow execution has the potential to significantly improve the performance of workflows that have large input data sizes.

This work represents a first step towards understanding the interplay between community-wide data placement services and community workflow management systems. In addition to the work presented here that focuses on staging data into workflows efficiently, we also plan to study placement services that move data sets produced by workflow execution off computational nodes in a timely and reliable way. This efficient staging out of data will allow workflows to execute efficiently on nodes that have limited storage. Finally, we are interested in placement services that replicate data for performance and reliability reasons and their relationship to workflow management systems.

Acknowledgments

This work was supported in part by the National Science Foundation under the grants CNS 0615412 and OCI 0534027 and by the Department of Energy's Scientific Discovery through Advanced Computing II program under grant DE-FC02-06ER25757. The authors would like to thank the Montage team: Bruce Berriman, John Good, and Daniel Katz for their helpful discussions and the use the of the Montage codes and 2MASS data.

References

[1] "Montage."http://montage.ipac.caltech.edu

[2] "Open Science Grid."www.opensciencegrid.org

[3] "TeraGrid."http://www.teragrid.org/

[4] A. Abramovici, W. Althouse, et al., "LIGO: The Laser Interferometer Gravitational-Wave Observatory," *Science*, vol. 256, pp. 325-333, 1992 1992.

[5] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster, "The Globus Striped GridFTP Framework and Server," in *SC05 Conference*, Seattle, WA, 2005.

[6] T. A. Barrass, et al., "Software Agents in Data and Workflow Management," in *Computing in High Energy and Nuclear Physics (CHEP) 2004*, Interlaken, Switzerland, 2004.

[7] G. B. Berriman, et al., "Montage: A Grid Enabled Engine for Delivering Custom Science-Grade Mosaics On Demand," in *SPIE Conference 5487: Astronomical Telescopes*, 2004.

[8] J. Blythe, et al., "Task Scheduling Strategies for Workflow-based Applications in Grids," in *CCGrid*, Cardiff, UK, 2005.

[9] A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B, Schwartzkopf, H, Stockinger, K. Stockinger, B. Tierney, "Giggle: A Framework for Constructing Sclable Replica Location Services," in *SC2002 Conference*, Baltimore, MD, 2002.

[10] A. Chervenak, R. Schuler, C. Kesselman, S. Koranda, B. Moe, "Wide Area Data Replication for Scientific Collaborations " in *Proceedings of 6th IEEE/ACM Int'l Workshop on Grid Computing (Grid2005)*, Seattle, WA, USA, 2005.

[11] A. L. Chervenak, N. Palavalli, S. Bharathi, C. Kesselman, R. Schwartzkopf, "Performance and Scalability of a Replica Location Service," in *Thirteenth IEEE Int'l Symposium High Performance Distributed Computing (HPDC-13)*, Honolulu, HI, 2004.

[12] B. G. Chun, et al., "Efficient replica maintenance for distributed storage systems," in *Proc. of the 3rd Symposium on Networked Systems Design and Implementation*, 2006.

[13] CMS Project, "The Compact Muon Solenoid, an Experiment for the Large Hadron Collider at CERN, http://cms.cern.ch/," 2005.http://cmsinfo.cern.ch/Welcome.html/

[14] E. Deelman, et al., "What Makes Workflows Work in an Opportunistic Environment?," *Concurrency and Computation: Practice and Experience*, vol. 18, 2005.

[15] E. Deelman, et al., "Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems," *Scientific Programming Journal*, vol. 13, pp. 219-237, 2005.

[16] E. Deelman, et al., "Pegasus: Mapping Large-Scale Workflows to Distributed Resources," in *Workflows in e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds.: Springer, 2006.

[17] I. Foster, et al., "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International Journal of High Performance Computing Applications*, vol. 15, pp. 200-222, 2001.

[18] J. Frey, et al., "Condor-G: A Computation Management Agent for Multi-Institutional Grids.," *Cluster Computing*, vol. 5, pp. 237-246, 2002.

[19] Globus, "www.globus.org," 2006

[20] T. Kosar and M. Livny, "A Framework for Reliable and Efficient Data Placement in Distributed Computing Systems," *Journal of Parallel and Distributed Computing*, vol. 65, pp. 1146-1157, 2005.

[21] J. Kubiatowicz, et al., "OceanStore: An Architecture for Global-Scale Persistent Storage," in *9th Int'l. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, 2000.

[22] Y.-K. Kwok and I. Ahmad, "Static Scheduling Algorithms for Allocating Directed Task Graphs.," *ACM Computing Surveys*, vol. 31, pp. 406-471, 1999.

[23] LIGO Project, "LIGO - Laser Interferometer Gravitational Wave Observatory, http://www.ligo.caltech.edu/," 2004.http://www.ligo.caltech.edu/

[24] LIGO Project, "Lightweight Data Replicator, http://www.lsc-group.phys.uwm.edu/LDR/," 2004.http://www.lsc-group.phys.uwm.edu/LDR/

[25] M. Litzkow, et al., "Condor - A Hunter of Idle Workstations," in *Proc. 8th Intl Conf. on Distributed Computing Systems*, 1988, pp. 104-111.

[26] A. Mandal, et al., "Scheduling Strategies for Mapping Application Workflows onto the Grid.," in *IEEE International Symposium on High Performance Distributed Computing (HPDC)* 2005

[27] J. Perry, et al., "QCDgrid: A Grid Resource for Quantum Chromodynamics " *Journal of Grid Computing*, vol. 3, pp. 113-130, June 2005 2005.

[28] QCDGrid Project, "QCDGrid: Probing the Building Blocks of Matter with the Power of the Grid, http://www.gridpp.ac.uk/qcdgrid/," 2005.http://www.gridpp.ac.uk/qcdgrid/

[29] K. Ranganathan and I. Foster, " Identifying Dynamic Replication Strategies for a High Performance Data Grid," in *2nd IEEE/ACM International Workshop on Grid Computing (Grid 2001)*, 2001.

[30] K. Ranganathan and I. Foster, "Decoupling Computation and Data Scheduling in Distributed Data Intensive Applications," in *Eleventh IEEE Int'l Symposium High Performance Distributed Computing (HPDC-11)*, Edinburgh, Scotland, 2002.

[31] K. Ranganathan and I. Foster, "Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids " *Journal of Grid Computing*, vol. 1(1), 2003 2003.

[32] J. Rehn, et al., "PhEDEx high-throughput data transfer management system," in *Computing in High Energy and Nuclear Physics (CHEP) 2006*, Mumbai, India, 2006.

[33] R. Sakellariou, et al., "Scheduling Workflows with Budget Constraints," in *Integrated Research in Grid Computing*, S. Gorlatch and M. Danelutto, Eds.: CoreGrid series, Springer-Verlag, 2007.

[34] E. Sit, et al., "Proactive replication for data durability," in *5rd International Workshop on Peer-to-Peer Systems (IPTPS 2006)*, 2006.

[35] M. Wieczorek, et al., "Scheduling of Scientific Workflows in the ASKALON Grid Environment," *SIGMOD Record*, vol. 34, 2005.

[36] J. Yu and R. Buyya, "A Budget Constraint Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms," in *Proceedings of the Workshop on Workflows in Support of Large-Scale Science (WORKS06)*

[37] 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