

A Data Placement Service for Petascale Applications

Ann L. Chervenak
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
+1(310)822-1511
annc@isi.edu

Robert Schuler
USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
+1(310)822-1511
schuler@isi.edu

ABSTRACT

We examine the use of policy-driven data placement services to improve the performance of data-intensive, petascale applications in high performance distributed computing environments. In particular, we are interested in using an asynchronous data placement service to stage data in and out of application workflows efficiently as well as to distribute and replicate data according to Virtual Organization policies. We propose a data placement service architecture and describe our implementation of one layer of this architecture, which provides efficient, priority-based bulk data transfers.

Categories and Subject Descriptors

D.2.11 [Software Architectures]

General Terms

Management, Performance, Design

Keywords

Data placement, data management, distributed computing, workflow management, data staging.

1. INTRODUCTION

The performance of petascale applications running on resources in a distributed environment can be significantly affected by the placement of data sets. Data sets that are placed on fast storage systems near computational resources where analysis will take place can be efficiently staged into computations and significantly reduce their execution time. By contrast, data sets that must be retrieved from archival storage systems or transferred across wide area networks can delay execution and harm application performance.

Our research examines a variety of policy-driven data placement algorithms for high performance distributed environments. In particular, we are interested in working with petascale applications to provide effective data placement. One set of data

placement policies may be driven by *Virtual Organizations* (VOs). For example, a VO might have a preferred distribution pattern for data sets generated at an experimental source that must be copied to sites where scientists can access the data locally. Hints on preferred data placement can also be provided by workflow management systems and metaschedulers.

Examples of data placement systems that distribute data according to VO policies include existing scientific data management systems such as the Physics Experiment Data Export (PheDEx) [3, 24] system for high-energy physics and the Lightweight Data Replicator (LDR) [16] for gravitational-wave physics. The PheDEx system was developed by the Compact Muon Solenoid (CMS) high energy physics experiment [11]. PheDEx uses agents to automate data distribution among tiers of sites, with the Tier 0 site at CERN containing all data sets, Tier 1 sites containing a large subset of these data sets, etc. PheDEx also uses agents to handle subscription-based and on-demand data distribution activities. The LDR system distributes data sets for the Laser Interferometer Gravitational Wave Observatory (LIGO) project [1, 17] among LIGO sites where analysis will take place. LDR distributes these data sets based on metadata queries performed by LIGO scientists at each site.

Workflow management systems and metaschedulers that manage large distributed computations consisting of many interdependent tasks can also help to drive data placement decisions. These systems often have extensive information about data items that need to be accessed together (for example, files that are staged into or out of particular computational tasks) as well as information about the order in which data items will be accessed (based on knowledge about dependencies among tasks). In our previous work [6], we integrated the Pegasus workflow management system [12, 13] from USC Information Sciences Institute with the Globus Data Replication Service [7], which provides efficient replication and registration of data sets. We demonstrated that using hints from the workflow management system allowed us to reduce the execution time of scientific workflows when we were able to successfully prestage necessary data onto appropriate computational resources.

This initial work has led us to design a general, asynchronous Data Placement Service (DPS) that will operate on behalf of a virtual organization and accept data placement requests from clients that reflect, for example, grouping of files, the order of file requests, etc. We also plan to incorporate configurable policies into the data placement service that reflect the data distribution policies of a particular VO. Our goal is to produce a placement service that works with petascale applications and manages the competing demands of VO data distribution policies, data staging

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Conference '04, Month 1–2, 2004, City, State, Country.
Copyright 2004 ACM 1-58113-000-0/00/0004...\$5.00.

requests from multiple competing workflows, and additional on-demand data requests from other clients.

The remainder of this paper is organized as follows. In the next section, we summarize our recent work using the Pegasus workflow management system and the Data Replication Service to demonstrate the potential performance improvements for workflow execution made possible by intelligent data placement. The next section describes the high-level design of our data placement service. Then we describe the initial implementation of one of the layers of the data placement services that provides efficient, priority-based bulk data transfers. Finally, we discuss related work and future plans.

2. DATA PLACEMENT AND WORKFLOW MANAGEMENT

In previous work [6], we studied the relationship between data placement services and workflow management systems. In particular, our goal was 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. 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 receive 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.

We deployed a simple prototype system that integrates the workflow management functionality of the Pegasus system [12, 13] with a data movement service called the Data Replication Service [8]. Using these existing services for data and workflow management in distributed computing environments, we demonstrated that asynchronous placement of data has the potential to significantly improve the performance of scientific workflows when the data sets can be placed on storage systems before execution begins.

3. DATA PLACEMENT SERVICE DESIGN

Figure 1 illustrates the relationship between a data placement service and one possible client, a workflow management system. In this scenario, the workflow management system coordinates many interdependent tasks in a computational workflow and communicates with the data placement service when data stage-in or stage-out operations are required. The data placement service is then responsible for coordinating data transfers, for example, stage-in of data to a storage system associated with the compute cluster(s) where workflow tasks will be run.

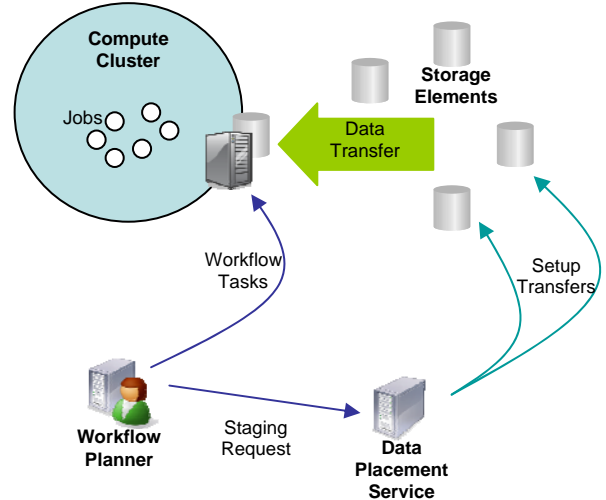


Figure 1: Workflow planner client sends data stage in requests to a data placement service, which initiates data transfer operations from storage elements in the distributed environment to the storage system(s) associated with compute nodes on which the workflow tasks will execute.

Once the placement service receives a client request, it must first determine how much of the requested data sets already exist at the destination storage site. It makes this determination using replica location catalogs that allow discovery of existing data sets. For files that need to be transferred, the placement service also consults replica catalogs to determine the locations of possible source files. The placement service next selects among available source replicas, initiates transfer operations and monitors their progress.

Since the data placement service may have multiple clients issuing competing data staging requests, it must prioritize these requests and issue transfers in priority order based on the available resources in the distributed computing environment.

If a computational task is stalled waiting for a data staging operation that has not yet completed, the computational task or the workflow management system may issue additional client requests to the data placement service to request an increase in the priority of that data staging operation. In this case, the data placement service may increase the priority of the staging request on its internal queue, based on the service's assessment of the relative priorities of outstanding and competing data transfer

requests. These may include staging requests for other running workflows or VO data distribution operations.

The interface for the data placement service will include operations to create, modify, and destroy placement requests; to start and cancel execution of these requests; and to query the state of a placement request. The client issues a *create* placement request command to the data placement service to specify the details of the request, such as the compute cluster(s) on which to stage the data and the sets of files to be staged. As part of this request, the client can indicate a placement policy type that must be interpretable by the data placement service. The client can issue a *modify* placement request command to the data placement service to modify some aspect of an existing request. For instance, the client can request that the priority of an existing placement request be changed. Also, this command could be used to add data items to be added to an existing request. The client issues a *destroy* command to remove an existing placement service request by implicitly cancelling it if it had been started and by permanently deleting all of its state information from the service. A *start* command begins execution of an existing placement request or restarts a request that was suspended due to transient failures. A *cancel* command terminates the activities of an active placement request but retains the state information associated with the request. Finally, a client can issue a *query* command to retrieve some aspect of the request state information. This command may be used by clients, such as jobs in a workflow, to test whether a file transfer is complete or still in-progress.

Figure 2 shows the layered architecture for the data placement service. A variety of policy-based placement algorithms will be supported at the top level, including the ability to maintain a certain number of copies of every data item by placing additional replicas as necessary; subscription-based placement of data; and push- and pull-based algorithms for placing data according to policies such as VO-level distributions of data items.

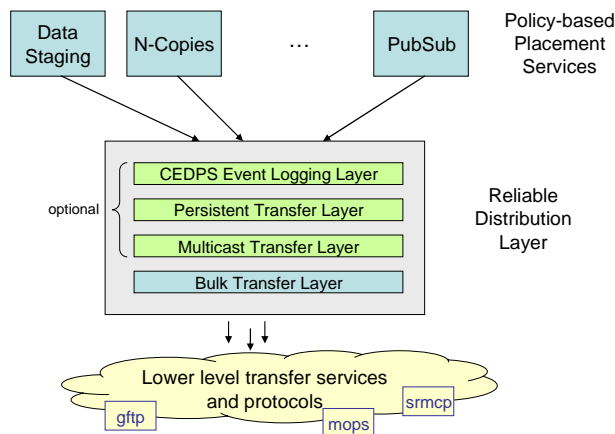


Figure 2: Layered architecture for data placement service

Below the policy-driven data placement layer is a reliable distribution layer, which is responsible for carrying out the placement plan generated at the policy layer. A key component of this layer is the Bulk Transfer Layer, which is responsible for efficient, priority-based data transfers. Above the bulk transfer layer are several optional layers that can add functionality to the reliable distribution of data. These include capabilities such as

multicast; persistence of outstanding requests so that the reliable distribution layer can restart operations after server failures; and use of the general event logging infrastructure being developed by the SciDAC CEDPS project. The reliable distribution layer invokes standard data transfer services to move data, which may include GridFTP, SRM, ftp, http, and others.

4. IMPLEMENTATION

We have implemented the first portion of the Data Placement Service, which was released in October 2007. The implementation focuses on the bulk transfer layer that supports efficient, priority-based data transfers. Over the next year, we will add more layers to the reliable distribution functionality and implement several algorithms for policy-driven data placement functionality.

The Bulk Transfer Service (BuTrS) is a lightweight, simple, efficient utility that supports bulk data transfers with priorities. The BuTrS utility is distributed in a single Java archive and has minimal third-party dependencies. A goal of our implementation is that this layer should be simple to install and require no administrative overhead, since it does not use containers, daemons or back end databases. The service includes a Java API and supports callback style notification of transfer results. The service is designed to be efficient and to support pairwise combinations of transfers between source and destination endpoints. The implementation includes efficient, linear-time algorithms for inserting, deleting, and reprioritizing bulk transfers.

Operations in the API include creating and destroying bulk transfers. The service also allows clients to change the priority of an existing individual data transfer, which may be needed, for example, when execution of a workflow task is stalled waiting for a data staging operation to complete. The initial implementation supports GridFTP third party transfers and most GridFTP transfer operations. The bulk transfer layer also supports retries of failed transfers up to a specified maximum number of retries.

Clients can query the status of outstanding transfers and receive callback notifications when the status of a transfer changes.

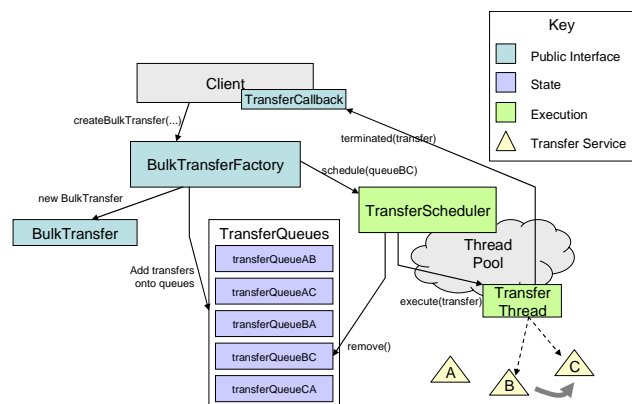


Figure 3: Shows the internal structure of the bulk transfer service.

Figure 3 shows details of the BuTrS service implementation. The service maintains priority-based transfer queues for each source and destination pair. When a client calls a new bulk transfer request, a factory creates a bulk transfer object associated with

that request and adds the associated transfers to the transfer queues. A transfer scheduler removes requests from the priority queues and executes the transfers, providing status to the client via callbacks.

5. 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.

5.1 Data and Computational Scheduling in Grids

Several groups have addressed issues related to the scheduling of data and computations in Grid environments. Two data distribution systems that relate closely to our proposed work are the Physics Experiment Data Export (PheDEx) [3, 24] system for high-energy physics and the Lightweight Data Replicator (LDR) [16] for gravitational-wave physics.

The high energy physics scientific community includes several experiments that will make use of terabytes of data collected from the Large Hadron Collider (LHC) [11] 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 [3], with data generated at the Tier 0 site at CERN and subsets of that data distributed to Tier 1 and Tier 2 sites. The PheDEx system manages data distribution for the Compact Muon Solenoid (CMS) high energy physics experiment [11]. The goal of the system is to automate data distribution processes as much as possible. Data operations include staging data from tape storage into disk buffers; wide area data transfers; 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. 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 Tier 1 and Tier 2 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 [16] distributes data for the Laser Interferometer Gravitational Wave Observatory (LIGO) project [1, 17]. 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 140 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 [5, 9] and the GridFTP data transport protocol [2]. LDR provides a rich set of data management

functionality, including replicating necessary files to a LIGO site. Each LDR site initiates transfers of data collections to the site using a pull model. For each file in a collection, a 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. A transfer daemon initiates data transfers of files on the scheduling queue in priority order, selecting among available data replicas and selecting a source for the data transfer.

The PheDEX and LDR systems both provide asynchronous data placement according to Virtual Organization level policies regarding how data should be distributed as well as respond to subscriptions to data sets and explicit requests for data items. The asynchronous nature of their data placement algorithms strongly influences our approach to data placement.

In [21-23] the authors conducted extensive simulation studies that examined the relationship between asynchronous data placement, replication and job scheduling. 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.

5.2 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 [15]. Scheduling scientific workflows in distributed environments has been recently studied in [4, 18, 25, 27-29]. 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.

5.3 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 [19, 20] 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 [14], several algorithms have been studied for replication of data to maintain high levels of durability. These include a reactive replication algorithm called *Carbonite* [10] that 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* [26] 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.

6. SUMMARY AND FUTURE WORK

We will extend our implementation of the Bulk Transfer Service with additional functionality for policy-driven data placement and reliable distribution. In particular, we plan to work with a variety of application communities, to understand their data placement requirements and implement corresponding policies in the data placement service.

7. ACKNOWLEDGMENTS

This work is funded by the Center for Enabling Distributed Petascale Science (CEDPS) under the Department of Energy's Scientific Discovery through Advanced Computing II program (Grant DE-FC02-06ER25757).

8. REFERENCES

- [1] Abramovici, A., Althouse, W., Drever, R., Gürsel, Y., Kawamura, S., Raab, F., Shoemaker, D., Sievers, L., Spero, R., Thorne, K., Vogt, R., Weiss, R., Whitcomb, S. and Zucker, M. LIGO: The Laser Interferometer Gravitational-Wave Observatory. *Science*, 256. 325-333.
- [2] Allcock, W., 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), 54.
- [3] Barrass, T.A., et al., Software Agents in Data and Workflow Management. in Computing in High Energy and Nuclear Physics (CHEP) 2004, (Interlaken, Switzerland, 2004).
- [4] Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A. and Kennedy, K., Task Scheduling Strategies for Workflow-based Applications in Grids. in CCGrid, (Cardiff, UK, 2005).
- [5] Chervenak, A., E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunst, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, B. Tierney, Giggie: A Framework for Constructing Scalable Replica Location Services. in SC2002 Conference, (Baltimore, MD, 2002).
- [6] Chervenak, A., E. Deelman, M. Livny, M. Su, R. Schuler, S. Bharathi, G. Mehta, K. Vahi, Data Placement for Scientific Applications in Distributed Environments. in 8th IEEE/ACM Int'l Conference on Grid Computing (Grid 2007), (Austin, Texas, 2007).
- [7] Chervenak, A., R. Schuler, C. Kesselman, S. Koranda, B. Moe, Wide Area Data Replication for Scientific Collaborations. in 6th IEEE/ACM Int'l Workshop on Grid Computing (Grid2005), (Seattle, WA, USA, 2005).
- [8] Chervenak, A., 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).
- [9] Chervenak, A.L., 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), 182-191.
- [10] Chun, B.G., Dabek, F., Haeberlen, A., Sit, E., Weatherspoon, H., Kaashoek, M.F., Kubiawicz, J. and Morris, R., Efficient replica maintenance for distributed storage systems. in Proc. of the 3rd Symposium on Networked Systems Design and Implementation, (2006).
- [11] CMS Project. The Compact Muon Solenoid, an Experiment for the Large Hadron Collider at CERN, <http://cms.cern.ch/>, 2005.
- [12] Deelman, E., et al. Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems. *Scientific Programming Journal*, 13. 219-237.
- [13] Deelman, E., et al. Pegasus: Mapping Large-Scale Workflows to Distributed Resources. in Taylor, I., E. Deelman, D. Gannon, M. Shields ed. *Workflows in e-Science*, Springer, 2006.
- [14] Kubiawicz, J., 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).
- [15] Kwok, Y.-K. and Ahmad, I. Static Scheduling Algorithms for Allocating Directed Task Graphs. *ACM Computing Surveys*, 31 (4). 406-471.
- [16] LIGO Project. Lightweight Data Replicator, <http://www.lsc-group.phys.uwm.edu/LDR/>, 2004.
- [17] LIGO Project. LIGO - Laser Interferometer Gravitational Wave Observatory, <http://www.ligo.caltech.edu/>, 2004.
- [18] Mandal, A., Kennedy, K., Koelbel, C., Marin, G., Mellor-Crummey, J., Liu, B. and Johnsson, L. Scheduling Strategies for Mapping Application Workflows onto the Grid. *IEEE International Symposium on High Performance Distributed Computing (HPDC) 2005*.
- [19] Perry, J., et al. QCDgrid: A Grid Resource for Quantum Chromodynamics *Journal of Grid Computing*, 3 (1-2). 113-130.
- [20] QCDGrid Project. QCDGrid: Probing the Building Blocks of Matter with the Power of the Grid, <http://www.gridpp.ac.uk/qcdgrid/>, 2005.
- [21] Ranganathan, K. and Foster, I., 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).

- [22] Ranganathan, K. and Foster, I., Identifying Dynamic Replication Strategies for a High Performance Data Grid. in 2nd IEEE/ACM International Workshop on Grid Computing (Grid 2001), (2001).
- [23] Ranganathan, K. and Foster, I. Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids *Journal of Grid Computing*, 1(1).
- [24] Rehn, J., Barrass, T., Bonacorsi, D., Hernandez, J., Semeniouk, I., Tuura, L. and Wu, Y., PhEDEx high-throughput data transfer management system. in *Computing in High Energy and Nuclear Physics (CHEP) 2006*, (Mumbai, India, 2006).
- [25] Sakellariou, R., Zhao, H., Tsiakkouri, E. and Dikaiakos, M.D. Scheduling Workflows with Budget Constraints. in Gorlatch, S. and Danelutto, M. eds. *Integrated Research in Grid Computing*, CoreGrid series, Springer-Verlag, 2007.
- [26] Sit, E., Haeberlen, A., Dabek, F., Chun, B.G., Weatherspoon, H., Morris, R., Kaashoek, M.F. and Kubiawicz, J., Proactive replication for data durability. in 5rd International Workshop on Peer-to-Peer Systems (IPTPS 2006), (2006).
- [27] Wiczorek, M., Prodan, R. and Fahringer, T. Scheduling of Scientific Workflows in the ASKALON Grid Environment. *SIGMOD Record*, 34 (3).
- [28] Yu, J. and Buyya, R. A Budget Constraint Scheduling of Workflow Applications on Utility Grids using Genetic Algorithms *Proceedings of the Workshop on Workflows in Support of Large-Scale Science (WORKS06)*.
- [29] Zhao, H. and Sakellariou, R. Advance Reservation Policies for Workflows 12th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP), Saint-Malo, France, 2006.