Scheduling of Scientific Workflows on Data Grids

Suraj Pandey and Rajkumar Buyya
Grid Computing and Distributed Systems (GRIDS) Laboratory
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{spandey,raj}@csse.unimelb.edu.au

Abstract

Selection of resources for execution of scientific workflows in data grids becomes challenging with the exponential growth of files as a result of the distribution of scientific experiments around the world. With more runs of these experiments, huge number of data-files produced can be made available from numerous resources. There is lack of work in optimal selection of data-hosts and compute resources in the presence of replicated files for scientific workflows. Foreseeing this, the thesis work aims at proposing novel workflow scheduling algorithms on data grids with large number of replicated files that incorporates practical constraints in heterogeneous environments such as Grids.

In this paper, we define the workflow scheduling problem statement in the context of data grids, supported by motivating applications; list research issues arising from practical constraints; propose two algorithms for experimenting with the problem; report simulation results obtained as a result of preliminary studies. The results are promising enough to motivate us to research on the problem stated.

1 Introduction

Scientific experiments like the Compact Muon Solenoid (CMS) experiment for the Large Hadron Collider (LHC) at CERN[1], the Laser Interferometer Gravitational-Wave Observatory's (LIGO) science[2] runs, the projects at Grid Physics Network[3] produce data in the scale of petabytes. These experiments are usually represented using directed acyclic graphs (DAG), called workflows, where jobs are linked according to their flow dependencies. The workflow is called compute-intensive when the computational needs of individual jobs are high. Similarly, the workflow is called data-intensive when the data requirements are high.

Optimizing the make-span and cost of execution is the most common objective function of scheduling these scientific applications. The make-span depends on both the communication time involved in transferring the files and the computation time to execute them. Scheduling the jobs to minimize only one objective results in decreased performance. Traditional scheduling algorithms which focus on pull model cannot be just applied as pulling data toward the computation resources incurs high bandwidth utilization cost. This demands a different model where the selection of both data-host and compute-host should be made rather than selecting data-host first and then compute-host or vice versa. In this paper, we define the problem with motivating examples and experiment with two algorithms. We then present the research issues given the primary problem statement.

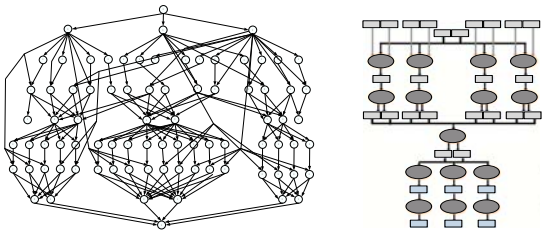
The rest of the paper is structured as follows: Section 2 gives some example applications as motivation, Section 3 outlines the related work, Section 4 gives the model, Section 5 states the problem, Section 6 proposes research issues, Section 7 provides some preliminary results, and Section 8 summarizes the paper with a timeline for the Ph.D. candidature.

2 Motivation

The Laser Interferometer Gravitational-Wave Observatory is a facility dedicated to the detection of cosmic gravitational waves and the harnessing of these waves for scientific research. The LIGO Lab, the LIGO Scientific Collaboration (LSC), and international partners, are proposing Advanced LIGO to improve the sensitivity by more than a factor of 10. Since the volume of space that the instrument can see grows as the cube of the distance, this means that the event rates will be more than 1,000 times greater. Advanced LIGO will equal the 1 year integrated observation time of initial LIGO in roughly 3 hours[1]. One such application workflow is depicted in Figure 1a. The huge amount of data produced will be made available to all of its sites. The selection of data source and compute-host will need to be done in an efficient manner such that the cost and time

[1] http://lhc.web.cern.ch/lhc/LHC_Experiments.htm
[2] http://www.ligo.caltech.edu/advLIGO/
[3] http://www.griphyn.org/

of execution are minimized. Cao et al. [3] have demonstrated a Data Monitoring Toolkit (DMT) with LIGO Data Grid (LDG). We are motivated to design more sophisticated algorithms for scheduling applications like advanced LIGO.



(a) Complex workflow as in LIGO. (b) Balanced workflow as in Atlas.

Figure 1: Example workflow structures.

Magnetic Resonance Imaging (MRI) uses radio waves and a strong magnetic field to provide detailed images of internal organs and tissues. Functional MRI (fMRI) is a procedure that uses MRI to measure the signal changes in an active part of the brain. A typical study of fMRI data requires multiple-stage processes that begin with the preprocess of raw data and concludes with a statistical analysis. Hundreds or even thousands of images are involved in such processes. Population-based atlas [18] creation is one of the major fMRI research activities. Figure 1b shows the workflow structure employing the Automated Image Registration (AIR) and FSL suite for creating population-based brain atlases from high resolution anatomical data [24]. A proper scheduling algorithm is required to select the image files from appropriate data-hosts and execute the jobs in selected compute-hosts.

For executing these application workflows, the data required can be retrieved from several data-hosts as there exist replicas of data files. Among the large number of data-hosts, the set of data-hosts that host the files required by a job should be found such that the repeated transfer of these huge files from one site to another is minimized. The problem of finding the set of data-hosts that provides all the files in optimal time is NP-complete [21]. Data has to be staged as input before any job can be executed. At the end of execution or during the execution, output data is produced that may also be of similar sizes to the input data. These intermediate data should be stored for subsequent jobs requiring them. These new hosts can be treated as a new source of data depending on the policy of retaining or deleting the temporarily data. The total number of data-hosts thus increases as the intermediate output files are stored and the selection problem becomes complex.

The computation requirements of these jobs are also very high. After the set of candidate data-hosts is found, the jobs need to be mapped to the appropriate compute-host for execution. The mapping of these jobs depends on the objective function. Scheduling of the jobs in the workflow primarily focuses on some of the objective functions or combination of them: for example, minimizing the total make-span, minimizing the overall cost of execution and data transfer, deadline and budget. The mapping of the workflow jobs to minimize one of the objective functions is a complex subproblem of the general job scheduling problem as stated in Section 5. The problem becomes complex with the addition of replicated data sets with jobs requiring more than a single file.

3 Related Work

In order to find out the location of replicated files, Replica Location Services (RLS) [5] Local Replica Catalog (LRC) provides information about the data available at the resource. RLS is a distributed replica management system consisting of local catalogs that contain information about logical to physical filename mappings and distributed indexes that summarize the local catalog content. Information about the state of the resources can be obtained via the Monitoring and Discovery Service (MDS) [9]. MDS provides information about the number and type of available resources, processor characteristics and the amount of available memory [23].

The work done by Shankar et al. [17] is most closely related to that of ours. While producing a schedule for the jobs in a workflow, they have proposed a planner that uses the *file_location* table to determine the locations of cached or replicated files. They take the volume of data into account while scheduling and have used cache mechanism for future usage similar to our notion of re-use. However, they do not consider the best location to get the data from. Moreover, the scheduling is performed for cluster management systems and not suitable for heterogeneous environments like Grid where instantaneous data replication, global search is not at all feasible.

Considerable work has been done by Deelman et al. [6, 7] on planning, mapping and data-reuse in workflow scheduling. Their Concrete Workflow Generator (CWG) queries the Transformation Catalog to determine if the components are available in the execution environment and to identify their locations. However, CWG selects a random location to execute from among the returned locations which does not necessarily give the best result. Also, if the input files are replicated at several locations, it selects the source location at random.

Zinn et al. [25] defined the mapping problem as Task Handling Problem (THP). They make strong assumptions by considering intrees and minimal series parallel graphs, unity communication cost and execution cost, which is not the case in heterogeneous environments. It is not always possible to transform workflows to intrees or minimal series parallel graphs.

Task to host assignment heuristic given by the Casanova et al. in the *XSufferage* [4] algorithm targets maximum file re-use by assigning the job to a cluster which already has the required file, provided the file is large compared to the available bandwidth on the cluster network's link. However, they do not consider dependent jobs and replicated files.

Topcuouglu et al. have designed a list scheduling algorithm called *HEFT*. This list scheduling algorithm assigns ranks to the jobs according to both the communication and computation costs and preserves the job execution precedence. We have used a modified version of this algorithm for experimenting with our case. We chose not to test Dynamic Critical Path (DCP) algorithm [11] due to its higher complexity.

Ranganathan et al. [16, 15] have used dynamic replication strategies to improve data access. Significant performance improvement is achieved when scheduling is performed according to data availability while also using a dynamic replication strategy. Locality of access should be leveraged by selecting the appropriate replica. However, replication cannot be done instantaneously given the huge data size and bandwidth constraints.

In Ramakrishnan et al. [14] work, workflow input data is staged dynamically, new data products are generated during execution. They focus on determining which data are no longer needed and when, by adding nodes to the workflow to cleanup data along the way. This minimizes the disk space usage as intermediate files generated are also of similar order to that of the input files.

Venugopal and Buyya [21] have proposed a heuristic, based on the Set Covering Problem (SCP), for selecting compute and data resources for data-intensive bag-of-task applications in an environment where the data is replicated (*SCP-MH*). However, it does not consider storage constraints on the nodes and the dependencies between jobs, and multiple-workflows. The work presented in this proposal incorporates these concerns.

4 Model

4.1 A Workflow Model

A workflow W is represented by a directed acyclic graph $G = (V, E)$, where $V = D \cup C \cup J$, and E represents the edge that maintains execution precedence constraints. Having a directed edge from j_x to j_y means that j_y cannot start to execute until j_x is completed. Having a directed edge from d_x to j_x means that job j_x cannot start to execute until file is transferred or made available to the execution host executing the job. The components are described as follows:

1. A set of jobs $J = \{j_1, j_2, \ldots, j_n\}$
2. A set of files $F = \{f_1, f_2, \ldots, f_n\}$
3. A set of compute-hosts $C = \{c_1, c_2, \ldots, c_n\}$
4. A set of data-hosts $D = \{d_1, d_2, \ldots, d_n\}$

A job j_x requires a set of files $F_x = \{f_1, f_2, \ldots, f_n\}$ to be staged in for execution. In the set of files, we denote f_k^t as temporary file and f_k^f as fixed file to distinguish files produced as a result of execution of a job and those files already hosted by data-hosts, respectively. File f_k^f is hosted by multiple data-hosts.

4.2 Resource Model

A compute resource is a high performance computing platform such as a cluster which has a 'head' node that manages a batch job submission system. Each compute-host has its own storage constrained data-host. There exists data-hosts that are only for storage purposes and do not provide computation. The communication cost between the compute-host and its own data-host is local and thus minimal in comparison to the communication cost between different hosts [22].

Data is organized in the form of datasets that are replicated on the data-hosts by a separate replication process that follows a strategy [2] that takes into consideration various factors such as locality of access, load on the data-host and available storage space. Information about the datasets and their location is available through a catalog such as the Storage Resource Broker Metadata Catalog [13][21].

5 Problem Statement

We now describe the problem of data-host selection and job to resource mapping in the presence of large number of replicated files.

A set of data-hosts that hosts the required files for the jobs in the workflow should be found. The selection of the optimal set of data-hosts in the presence of large number of replicated files for a single job is computationally intensive [21]. The selection is being made by using one of the solutions to the Set-Coverage problem [1]. This selection procedure is compute-intensive.

The mapping of jobs to the compute-resources is an *NP-complete* problem in the general form. The problem is *NP-complete* even in two simple cases: (1) scheduling jobs with uniform weights to an arbitrary number of processors and (2) scheduling jobs with weights equal to one or two units to two processors [20]. There are only three special cases for which there exist optimal polynomial-time algorithms. These cases are (1) scheduling tree-structured job graphs with uniform computation costs on an arbitrary number of processors [10]; (2) scheduling arbitrary job graphs with uniform computation costs on two processors [12]; and (3) scheduling an interval-ordered job graph [8]. However, even in these cases, communication among jobs of the parallel program is assumed to take zero time.

The mapping of jobs in a workflow to the resources in our case is significantly different than the general mapping

problem. Given the replicated files and numerous data-hosts, proper selection of data-hosts and compute-hosts should be made for every job in the workflow such that the dependent jobs will benefit from selection set of its parents. The best case is such that each job gets the optimal data-host set and compute-host pair for execution with the optimal make-span and minimum cost of the workflow. The naive case is when the set is chosen for each job irrespective of their dependencies.

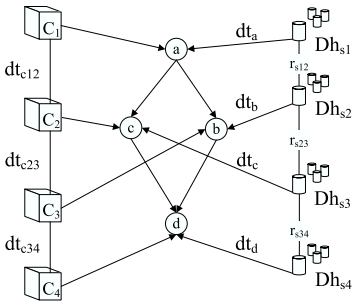


Figure 2: Data-host and Compute-host Selection Problem.

Figure 2 shows a simple workflow with separate compute-hosts and separate data-hosts mapped to each job. Lets first consider the data-transfers occurring due to the selection of compute-host and data-host for job a and job c. Since the jobs are mapped to two different compute-hosts, the intermediate files needs to be transferred from C_1 to C_2 with cost dt_{c12}. Job a has cost dt_a and job c has cost dt_c for transferring data from Dh_{s1} and Dh_{s3}, respectively. Now the optimal solution would select a combination of C_1, C_2, Dh_{s1} and Dh_{s3} to minimize the data transfer cost $\sum (dt_{c12} + dt_a + dt_c)$ and execution time $\sum (C_1 + C_2)_t$. There are several ways to select the data-hosts Dh_{s1} and Dh_{s3} and compute-hosts C_1 and C_2. We describe two techniques. The first way is by considering the proximity of data-hosts in terms of network distance with the set of compute-hosts. The second way is by trying to maximize the co-relation between two set of data-hosts, depicted as r_{s12} in the figure, for some set of jobs that require same set of files.

The first way always searches the entire data-host and compute-host set to find one combination (which is NP-complete) that most closely satisfies the objective function. But it does not take into account that jobs might be sharing the same set of files. Moreover, previous iterations might have already found the data-host and compute-host set combination that can be applied to subsequent set of jobs. For example jobs a and c might be requiring same set of files, so the compute-intensive selection of *candidate* data-host sets and compute-host can be avoided for the second job. The second way would almost always select the same set of data-hosts by trying to maximize their co-relation. This also restricts the compute-host set to within the proximity of the co-related data-host set. When the number of jobs increases, both the compute-host and data-host become overloaded, increasing waiting time of the jobs. Hence a proper selection algorithm should distribute the load and satisfy the objective function. The problem is formally stated in Definition 1.

Definition 1. $DDCSP(D, C, J, F, G, L)$ *Given a set of data-hosts D, a set of compute-hosts C, a set of jobs J, a set of files F, the workflow DAG G and workflow execution time bound L, Dynamic Data-Compute Scheduling Problem (DDCSP) is a problem of finding obtainable mappings of jobs J, where each job j_i may be requiring more than one input file, to compute-hosts C with replicated files in data-hosts D to 1) minimize the make-span of the workflow bounded within L time limits, 2) minimize the total cost of execution , 3) satisfy system and users' constraints.*

6 Research Issues

The scheduling of workflow jobs, as defined in Definition 1, becomes complicated when we consider the following key challenges:

- **Scheduling Policy.** According to the selection of data-hosts and mapping of resources, the workflow's make-span and overall cost changes significantly.
- **Storage Constraints.** Only limited storage capacity is available at resources. As the jobs get executed, the data produced should be either deleted or moved. Storage aware resource scheduling problem is a major area of research. Data providence should be associated with scheduling policy.
- **Fault Tolerance.** If an assigned job fails, the effect gets reflected to the entire workflow unless a fault tolerance mechanism is in place. Pro-active scheduling techniques should be looked into.
- **Replication Policy.** The availability of replicas of data and their locality heavily depends on the replication policy in place. Given the statistical analysis of scientific workflow runs, a dynamic replication policy that can balance the replicas among peers should be formulated.
- **Resource Provisioning.** A service-level-agreement (SLA) based advance reservation can be in place to circumvent the sudden scarcity of resources and thus guarantee access at the scheduled time.
- **Multiple Workflows, Multiple Runs.** Efficient scheduling of multiple workflows at the same time is a research problem. Same workflows can be run many times at varying intervals and places. A proper scheduling algorithm should be designed to incorporate simultaneity and improvidence of users' requests.
- **User QoS.** Users' Quality of Service (QoS) such as budget and deadline. should also be taken into account while scheduling workflows.

7 Preliminary Results

In the course of experimenting with the scheduling algorithms, we have formulated a modified version of *HEFT* algorithm [19], and call it Dynamic Data-Resource-Selection (*DDRS*), as listed in Algorithm 1. Unlike the original *HEFT*, the ranking of the jobs is recalculated in intervals such that the scheduling is based on dynamic statistics of the resources. This changes the job to resource mapping at the time while forming the list of jobs. Jobs are selected in descending order of their ranking for mapping to a resource. The best compute resource is chosen based on *SCP-MH*. We also modified the *HEFT* algorithm similar to *DDRS* and selected compute resource using the earliest finish time.

Algorithm 1 DDRS algorithm.

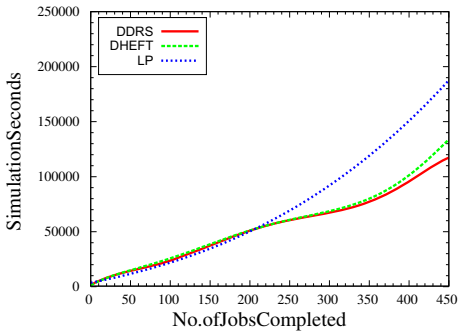
1: **repeat**
2: Set the computation costs of unfinished jobs & communication costs of edges with mean values
3: Compute $rank_u$ for all unfinished jobs by traversing graph upward, starting from the exit job
4: Sort the jobs in a scheduling list by nonincreasing order of $rank_u$ values
5: **for** each job $j_i \in J$ in the sorted list **do**
6: Choose the compute resource c_k using *SCP-MH*
7: **end for**
8: **for all** $j_i \in J$ **do**
9: Assign job j_i to the compute resource c_k
10: **end for**
11: Dispatch all the mapped jobs
12: Wait for the $POLLING\_TIME$
13: Update the ready job list
14: **until** there are unscheduled jobs in the ready list

We also formulated a *Level-Partitioner (LP)* algorithm that schedules jobs one by one according to the dependencies, as listed in Algorithm 2. Instead of maintaining a list of jobs, the algorithm starts dispatching jobs starting from the root. Once the parent job has finished, the child job is ready to be scheduled. So some dependency chains might get scheduled earlier if the jobs in the chain demand less compute and transfer time. For every job, the compute resource is chosen using *SCP-MH*. This algorithm however does not consider the criticality of ordering the paths according to path weights like done in algorithm 1.

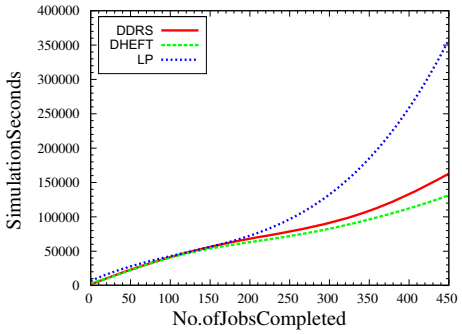
We have used GridSim to simulate the data-intensive environment and evaluate the performance of the scheduling algorithms. We model the data-intensive computing environment based on the European Data Grid Testbed 1 given by Bell, et al. [2] as has been simulated by Srikumar et al. [21]. We refer the reader to this paper for more information on the simulation environment setup. We experiment with two workflows as depicted in Figure 1. Each job has at least two file dependencies from its parents. Both the stage-in and intermediate files are 10MB each in size and are distributed according to *Zipf* distribution among the resources. The compute resources are rated in terms of MIPS (Million Instructions Per Sec) and the make span is reported in simulation seconds.

Algorithm 2 Level-Partitioner algorithm.

1: **repeat**
2: **for** each ready job $j_i \in J$ **do**
3: Choose the compute resource c_k using *SCP-MH*
4: **end for**
5: **for all** $j_i \in J$ **do**
6: Assign job j_i to the compute resource c_k
7: **end for**
8: Dispatch all the mapped jobs
9: Wait for the $POLLING\_TIME$
10: Update the ready job list
11: **until** there are unscheduled jobs in the ready list



(a) Make-span of complex workflows like LIGO.



(b) Make-span of balanced workflows like fMRI-Atlas.

Figure 3: Average Make-span of workflows in 100 iterations.

Figure 3 and Table 1 shows that *DDRS* algorithm performs well in comparison to *DHEFT* and *LP*. For complex workflows like that of LIGO's, significant time reduction (↓) is seen. However, for balanced workflows like that of Atlas, *DHEFT* performs better than *DDRS* and *LP*.

8 Summary and Timeline

In this paper, we have formulated the problem of scheduling job of scientific workflows in the presence of

Table 1: Simulation results averaged over 100 iterations.

#Jobs	Algorithm	LIGO	Time ↓ (Simulation Seconds)	Atlas	Time ↓
450	*DHEFT*	135000	5%	130000	−23%
	DDRS	128000		160000	
	LP	185000	30%	360000	55%

large number of replicated files. Also, we have outlined the research issues arising when physical and practical constraints are considered in the system. With the support of the preliminary results, we believe that more promising results can be obtained from rigorous research efforts. Hence we conclude that the outlined problem can be insighted to Ph.D. research by following the plan as laid out in Table 2.

Table 2: Future research activities and timeline.

Timeline	Description
Jan.'08-Jun.'08	Simulate/implement scheduling algorithms. Work on taxonomy for workflow on data grids.
Jul.'08-Dec.'08	Work on solving constraints with workflows and data grid scheduling.Work on surveying of architectures related to workflows and data grids.
Jan.'09-Jun.'09	Work on **new** ``workflow and data grid architecture´´ and execution models.
Jul.'09-Dec.'09	Work on analytical comparison, verification, performance modeling of work so far. Integrate industry standards & requirements with our work.
Jan.'10-Jun.'10	Develop a prototype system for executing various scientific workflows.Assimilate the testbed and evaluate the system. Finalize thesis.
Ph.D. & Beyond	Give continuity to the work on management of scientific workflows on Grids.

9 Acknowledgment

We thank Srikumar Venugopal, Marco A.S. Netto, Marcos Assunção, Al-Mukaddim Khan Pathan from the University of Melbourne for their encouragement and support in the formulation of this proposal.

References

[1] E. Balas and M. W. Padberg. On the set-covering problem. *Operations Research*, 20(6), 1972.

[2] W. H. Bell, D. G. Cameron, L. Capozza, A. P. Millar, K. Stockinger, and F. Zini. Simulation of dynamic grid replication strategies in optorsim. In *GRID '02: Proceedings of the Third International Workshop on Grid Computing*, UK, 2002. Springer-Verlag.

[3] J. Cao. Enabling LIGO applications on scientific grids. OSG Consortium Meeting, 2005.

[4] H. Casanova, D. Zagorodnov, F. Berman, and A. Legrand. Heuristics for scheduling parameter sweep applications in grid environments. In *Proceedings of the 9th Heterogeneous Computing Workshop*, USA, 2000. IEEE.

[5] A. Chervenak, E. Deelman, I. Foster, and et al. Giggle: a framework for constructing scalable replica location services. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, USA, 2002. IEEE.

[6] E. Deelman, J. Blythe, and et al. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, 1, 2003.

[7] E. Deelman, J. Blythe, Y. Gil, and C. Kesselman. *Grid resource management: state of the art and future trends*, chapter Workflow management in GriPhyN. Kluwer Academic Publishers, USA, 2004.

[8] P. C. Fishburn. *Interval Orders and Interval Graphs–A Study of Partially Ordered Sets*. John Wiley, USA, 1985.

[9] S. Fitzgerald. Grid information services for distributed resource sharing. In *HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*, USA, 2001. IEEE.

[10] T. C. Hu. Parallel sequencing and assembly line problems. *Operations Research*, 9(3), 1961.

[11] Y.-K. Kwok and I. Ahmad. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. *IEEE Trans. Parallel Distrib. Syst.*, 7(5), 1996.

[12] E. Mayr and H. Stadtherr. Optimal parallel algorithms for two processor scheduling with tree precedence constraints. Technical Report Feb1-1, 1996.

[13] A. Rajasekar, M. Wan, and R. Moore. Mysrb & srb: Components of a data grid. In *Proceedings of the 11 th IEEE International Symposium on High Performance Distributed Computing*, UK, 2002.

[14] A. Ramakrishnan, G. Singh, and et al. Scheduling data-intensive workflows onto storage-constrained distributed resources. In *Proceedings of CCGRID*, Washington, DC, USA, 2007. IEEE Computer Society.

[15] K. Ranganathan and I. Foster. Decoupling computation and data scheduling in distributed data-intensive applications. In *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, USA, 2002. IEEE.

[16] K. Ranganathan and I. T. Foster. Identifying dynamic replication strategies for a high-performance data grid. In *Proceedings of the Second International Workshop on Grid Computing*, UK, 2001. Springer-Verlag.

[17] S. Shankar and D. J. DeWitt. Data driven workflow planning in cluster management systems. In *Proceedings of the 16th international symposium on High performance distributed computing*, USA, 2007. ACM Press.

[18] A. W. Toga and J. C. Mazziotta. *Brain Mapping: The Methods(2nd Edition)*. Academic Press, 2002.

[19] H. Topcuouglu, S. Hariri, and M. you Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.*, 13(3), 2002.

[20] J. D. Ullman. Np-complete scheduling problems. *J. Comput. Syst. Sci.*, 10(3), 1975.

[21] S. Venugopal and R. Buyya. A set coverage-based mapping heuristic for scheduling distributed data-intensive applications on global grids. In *Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*, 2006.

[22] S. Venugopal, R. Buyya, and K. Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1), 2006.

[23] J. Yu and R. Buyya. A taxonomy of scientific workflow systems for grid computing. *SIGMOD Rec.*, 34(3), 2005.

[24] J. Yu and R. Buyya. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming Journal*, 2006.

[25] D. Zinn, T. H. Wong, and B. Ludaescher. Scheduling data-intensive workflows in kepler. Presentation, 2007.