Data Intensive Distributed Computing:

Challenges and Solutions for Large-Scale Information Management

Tevfik Kosar
State University of New York at Buffalo (SUNY), USA

Information Science
REFERENCE

Managing Director: Lindsay Johnston
Senior Editorial Director: Heather Probst
Book Production Manager: Sean Woznicki
Development Manager: Joel Gamon
Development Editor: Hannah Abelbeck
Acquisitions Editor: Erika Gallagher
Typesetters: Milan Vracarich, Jr.
Cover Design: Nick Newcomer, Greg Snader

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com

Library of Congress Cataloging-in-Publication Data

Data intensive distributed computing: challenges and solutions for large-scale information management / Tevfik Kosar, editor.
p. cm.
Includes bibliographical references and index.
Summary: “This book focuses on the challenges of distributed systems imposed by the data intensive applications, and on the different state-of-the-art solutions proposed to overcome these challenges”--Provided by publisher.
ISBN 978-1-61520-971-2 -- ISBN 978-1-61520-972-9 (ebk.) 1. Expert systems (Computer science) 2. Computer systems. I. Kosar, Tevfik, 1974-
QA76.76.E95D378 2012
006.3--dc22
2010006730

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter 7
A Survey of Scheduling and Management Techniques for Data-Intensive Application Workflows

Suraj Pandey
The Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia

Rajkumar Buyya
The University of Melbourne, Australia

ABSTRACT

This chapter presents a comprehensive survey of algorithms, techniques, and frameworks used for scheduling and management of data-intensive application workflows. Many complex scientific experiments are expressed in the form of workflows for structured, repeatable, controlled, scalable, and automated executions. This chapter focuses on the type of workflows that have tasks processing huge amount of data, usually in the range from hundreds of mega-bytes to petabytes. Scientists are already using Grid systems that schedule these workflows onto globally distributed resources for optimizing various objectives: minimize total makespan of the workflow, minimize cost and usage of network bandwidth, minimize cost of computation and storage, meet the deadline of the application, and so forth. This chapter lists and describes techniques used in each of these systems for processing huge amount of data. A survey of workflow management techniques is useful for understanding the working of the Grid systems providing insights on performance optimization of scientific applications dealing with data-intensive workloads.

DOI: 10.4018/978-1-61520-971-2.ch007

INTRODUCTION

Scientists and researchers around the world have been conducting simulations and experiments as a part of medium to ultra large-scale studies in high-energy physics, biomedicine, climate modeling, astronomy and so forth. They are always seeking cutting-edge technologies to transfer, store and process the data in a more systematic and controlled manner as the data requirements of these applications range from megabytes to petabytes. Thus, to help them manage the complexity of execution, transfer and storage of results of these large-scale applications, the use of a Workflow Management Systems (WfMS) is in wide practice (Yu & Buyya, 2005).

Scheduling and managing computational tasks of a workflow were the main focus of WfMS in the past. With the emergence of globally distributed computing resources and increasing output data from scientific experiments, scientists began to realize the necessity of handling data in conjunction with computational tasks. Scientific workflows were then modeled taking into account the flow of data. However, even with a plethora of techniques and systems, many challenges remain in the area of data management related to workflow creation, execution, and result management (Deelman & Chervenak, 2008; Gil et al., 2007).

Some challenges for managing data-intensive application workflows are:

- High throughput data transfer mechanisms
- Massive, cheap, green and low latency storage solutions and their interfaces
- Composition of scientific applications as workflows
- Multi-core technology and workflow management systems
- Standards for Interoperability between workflow systems
- Globally distributed data and computation resources

In this chapter, we classify and survey techniques that have been used for managing and scheduling data-intensive application workflows to meet the challenges listed above. The classification is based on techniques that take into account data, storage, platform and application characteristics. We sub-divide each general heading into more specific techniques. We then list and describe several work under each sub-heading. Most systems use a combination of existing techniques to achieve the objectives of an application workflow.

The rest of the chapter is organized as follows. In next section, we present previous studies that focused more on systems side of Grid workflows and Data Grids along with their taxonomy. We then describe the terms and definitions used in this chapter followed by an abstract model of a WfMS and its component responsible for data and computation management. In the rest of the chapter, we present the survey. We finally conclude identifying some future trends in management of data-intensive application workflows.

RELATED WORK

Over the last few years, we can find much work being done on data-intensive environments and workflow management systems. We list taxonomies for Data Grid Systems and Workflow management Systems that present the grounds for our survey.

Venugopal, Buyya, & Ramamohanarao (2006) proposed a comprehensive taxonomy of data Grids for distributed data sharing, management and processing. They characterize, classify and describe various aspects of architecture, data transportation, data replication and resource allocation, and scheduling for Data Grids systems. They list the similarities and differences between Data Grids and other distributed data-intensive paradigms such as content delivery networks, peer-to-peer networks, and distributed databases.

Yu & Buyya (2005) proposed taxonomy of workflow management systems for Grid computing. They characterize and classify different approaches for building and executing workflows on Grids. They present a survey of representative Grid workflow systems highlighting their features and pointing out the differences. Their taxonomy focuses on workflow design, workflow scheduling, fault management and data movement.

Bahsi, Ceyhan & Kosar (2007) presented a survey and analysis on conditional workflow management. They studied workflow management systems and their support for conditional structures such as *if, switch and while*. With case studies on existing WfMS, they listed the differences in implementation of common conditional structures. They show that the same structure is implemented in completely different ways by different WfMS. A system or a user can define explicit conditions in the structure of a workflow to manage the data flow across resources and between tasks for data-intensive application workflows.

Yu, Buyya, & Ramamohanarao (2008) listed and described several existing workflow scheduling algorithms developed and deployed in various Grid environments. They categorized the scheduling algorithms as either best effort based or Quality of Service (QoS) constraint based scheduling. Under best-effort scheduling, they presented several heuristics and meta-heuristics based algorithms, which intend to optimize workflow execution times on community Grids. Under QoS constraint based scheduling algorithms, they examined algorithms, which intend to solve performance optimization problems based on two QoS constraints, deadline and budged. They also list some of the techniques we have explicitly described for data-intensive workflows in this chapter.

Kwok & Ahmad (1999) surveyed different static scheduling techniques for scheduling application Directed Acyclic Graphs (DAGs) onto homogeneous platforms. In their model, tasks are scheduled onto multiprocessor systems. The model also assumes that communication is achieved solely by message passing between processing elements. They proposed taxonomy that classified the scheduling algorithms based on their functionality. Their survey also provides examples for each algorithm along with the overview of the software tools for scheduling and mapping.

TERMS AND DEFINITIONS

In this section, we define the terms *data-intensive*, *scientific workflow* and *workflow scheduling* as applicable for scientists working on distributed, heterogeneous, large-scale platforms such as Grids and Clouds.

Data-Intensive

A data-intensive computing environment consists of applications that produce, manipulate, or analyze data in the range of hundreds of megabytes (MB) to petabytes (PB) and beyond (Moore, Prince, & Ellisman, 1998). A data-intensive application workflow has higher data workloads to manage than its computational parts. In other words, the requirements of resource interconnection bandwidth for transferring data outweigh the computational requirements for processing tasks. This, as a consequence, demands more time to transfer and store data as compared to task execution of a workflow. It is common to characterize the distinction between data-intensive and compute-intensive by defining a threshold for the Computation to Communication Ratio (CCR). Applications with lower values of this ratio are distinctly data-intensive in nature.

Scientific Workflow

Standard application components of scientific, data-intensive applications can be combined to process the data in a structured way in contrast to executing monolithic codes (Deelman et al., 2003).

The application is represented as a workflow structure, which consists of tasks, data elements, control sequences and their dependencies. According to Zhao et al. (2008), scientific workflow management systems are engaged and applied to the following aspects of scientific computations: 1) describing complex scientific procedures, 2) automating data derivation processes, 3) high performance computing (HPC) to improve throughput and performance, and 4) provenance management and query.

Workflow Scheduling

In simple terms, a process of mapping of tasks in a workflow (or an entire workflow) to compute resources for execution (preserving dependencies between tasks) is termed as scheduling of workflows. Once the workflow is instantiated in the form of a DAG, middleware technologies, such as Pegasus (Deelman et al., 2005), Gridbus Workflow Management System (Yu & Buyya, 2004) and so forth, are used to schedule the jobs described in the nodes of the DAG onto the specified resources in their specified order. The objectives of scheduling a workflow can vary from application to application. Most often, a data-intensive application workflow is scheduled to optimize the data-transfer time/cost, storage space, total execution time or a combination of these.

Resource Broker

A resource broker is an intermediate entity that acts as a mediator between Grid resources and end users. It performs resource allocation and/or scheduling, and manages execution of applications on behalf of one or multiple users. For instance, the Grid Service Broker (Venugopal, Buyya, & Winton, 2006) developed as part of the Gridbus Project, mediates access to distributed resources by discovering resources, scheduling tasks, monitoring and collating results.

ABSTRACT MODEL OF A WORKFLOW MANAGEMENT SYSTEM

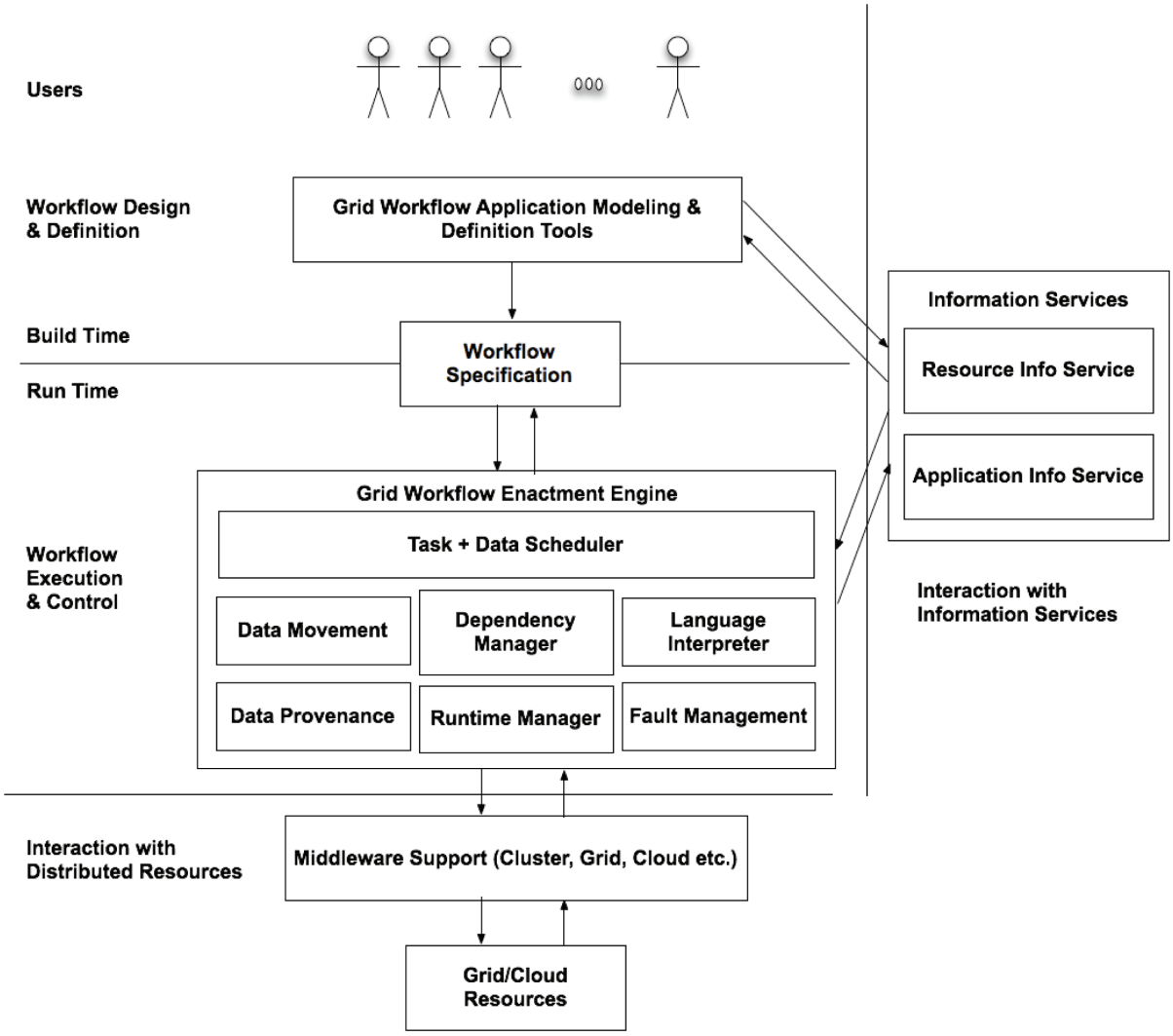
Figure 1 shows the architecture of a Grid workflow system based on the workflow reference model (Hollingsworth, 1994) proposed by Workflow Management Coalition (WfMC) (www.wfmc.org) in 1994. We have extended it to include components that manage data in addition to tasks.

Yu et al. (2005) have described the abstract model in detail, but without the data-centric components. The *build time* and *run time* borders separate the functionality of the design to defining and executing tasks, respectively. At the core of the *run time*, we propose components to actively process both data and tasks equally, different from the model presented by Yu et al. (2005), where data was not given as high priority as tasks.

The scheduler, that forms the core of the engine, handles data flow schedules on top of task schedules. For example, if a workflow is modeled such that the data transfer tasks are separate from computation tasks, the scheduler may apply a different scheduling policy to the data transfer tasks. Similarly, when there is no distinction between these tasks, the scheduler may prioritize data transfers between certain tasks over computation depending on the structure of the workflow, scheduling objectives, and so forth.

We propose to add a *data provenance* (also referred to as lineage and pedigree) manager component to the architecture. It keeps the record of data entities associated with the tasks in a workflow. The scheduler may interact with this component for determining specific data flow paths between tasks and distributed resources. For example, when a workflow is executed a number of times, previously produced data may exist that could be reused. In such cases, intermediate data transfer may not be scheduled for some tasks. Similarly, the scheduler may take reference of provenance data to create/dissolve data transfer and data cleanup tasks for storage aware sched-

Figure 1. An abstract model of a workflow management system



uling. Simmhan, Plale, & Gannon (2005) have surveyed and described systems using provenance for data-intensive environments in greater detail.

We envision each component in the core architecture to handle data as a first class citizen as also proposed by Kosar & Livny (2004). Data movement component, in particular, should be smart enough to overlap data transfer tasks with computation so that wait-times for data-availability is minimized. Data-transfer tasks could be prioritized for different tasks. Similarly, fault tolerance policies should be capable of handling frequent failures of data transfer tasks. Scheduling steps heavily depend on the capability of data movement and fault tolerance components for data-intensive applications as the repercussions of failure of data transfer tasks can affect the performance of the entire workflow. Different from generic WfMS models, a higher and more sophisticated coordination mechanism is required between these components for handling data-intensive application workflows.

New models for IT service delivery (e.g. Clouds Computing) are emerging. Workflow systems

should be capable of interacting with these types of service oriented architectures so that it can better utilize the storage and compute facilities provided by them for optimized data delivery, storage and distributed access. Access and security policies may different than existing Grid infrastructures when resources are from centralized data centers.

SURVEY

In this section, we characterize and classify key concepts and techniques used for scheduling and managing data-intensive application workflows. As shown in Figure 2, we have classified the techniques into seven major categories: (a) data locality, (b) data transfer, (b) data-footprint, (c) granularity, (d) model, (e) platform, (f) miscellaneous technologies. In this section, we describe each of these categories and their branches in detail.

Data Locality

In data-intensive computing environments, the amount of data involved is huge. Transferring data between computing nodes takes significant amount of time depending on the size of data and network capacity between participating nodes. Hence, most scheduling techniques target on optimizing data transfers by exploiting the locality of data. These techniques can be classified into: (a) spatial clustering, (b) task clustering, and (c) worker centric.

Spatial Clustering

Spatial clustering creates a task workflow based on the spatial relationship of files in the input data set. In spatial clustering, clusters of jobs are created based on spatial proximity, each job then assigned to a cluster, each cluster to a grid site and during the execution of the workflow, all jobs scheduled belonging to the cluster to the same site (Meyer, Annis, Wilde, Mattoso, & Foster, 2006). It improves data reuse and reduces total number of file transfers by clustering together tasks with high input-set overlap. These clustered tasks are scheduled to the resource with the maximum overlap of input data. This reduction benefits the Grid as a whole by reducing traffic between the sites. It also benefits the application by improving its performance.

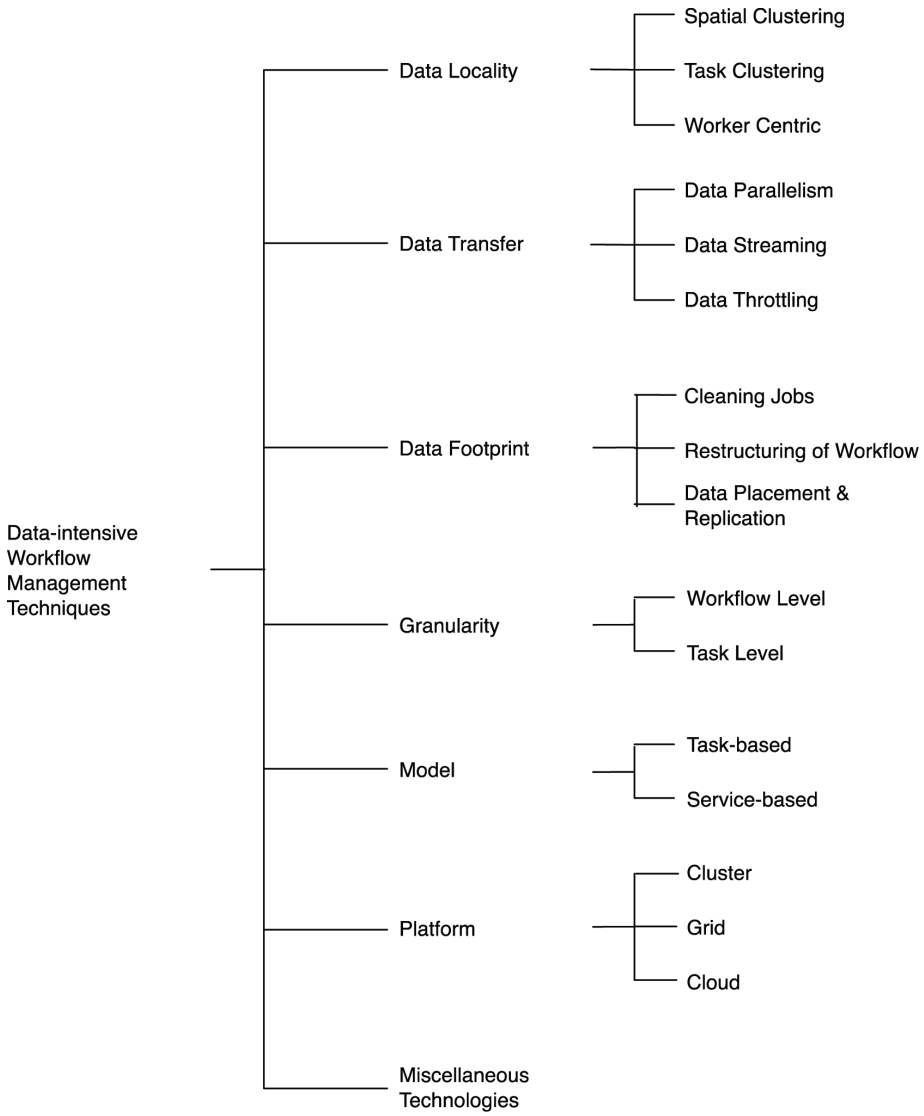
Meyer et al. (2006) presented a generalized approach to planning spatial workflow schedules for Grid execution based on the spatial proximity of files and the spatial range of jobs. They proposed *SPCL* (for "spatial clustering") algorithm that takes advantage of data locality through the use of dynamic replication and schedule jobs in a manner that reduces the number of replicas created and the number of file transfers performed when executing a workflow. They evaluated their solution to the problem using the file access pattern of an astronomy application that performs *coaddition* of images from the Sloan Digital Sky Survey (SDSS) (*SDSS Project*, 2000).

Brandic, Pllana & Benkner (2006) developed QoS-aware Grid Workflow Language (QoWL), by extending the Business Process Execution Language (BPEL) that allows users to define preferences regarding the execution *location affinity* for activities with specific security and legal constraints. Using QoS parameters that directs the WfMS to restrict the movement of sensitive and proprietary data to only agreed domains is very important for certain kinds of applications. A set of QoS-aware service-oriented components is provided for workflow planning to support automatic constraint-based service negotiation and workflow optimization.

Task Clustering

With task clustering, small tasks are grouped together as one executable unit such that the overhead of data movement can be eliminated. Task clustering groups tasks so that the intermediate

Figure 2. Classification of management techniques for data-intensive application workflows



files produced by each task in the group remains in the same computing node the grouped task was submitted to. Other tasks in the same group can now access the file locally. This scheme reduces the need to transfer the intermediate output files in case the tasks in the group were scheduled to different computing nodes. Clustering also eliminates the overhead of running small tasks.

Singh, Kesselman, & Deelman (2005) explored approaches for restructuring of workflows so that the dependencies in the workflow graph can be reduced. They group independent jobs at the same level into clusters. Their task clustering does not imply that the tasks in a group is scheduled to one processor or executed sequentially. They show workflow performance using clustering with centralized (single submit host) and distributed (multiple submit hosts) job submission. In the centralized submission, the whole workflow is submitted and executed using a single submit host. In order to increase the dispatch rate of jobs for execution, their distributed job submission strat-

egy has a central manager, multiple submit hosts and worker nodes. The workflow is restructured with multiple clusters at each level. The number of clusters at each level is equal to the number of submit hosts in the pool. The schedulers on the submit hosts then try to find suitable nodes for the submitted jobs.

Pandey et al. (2009) used task clustering to schedule data-intensive tasks for a medical application workflow. They clustered tasks based on their execution time, data transfer and level. If tasks were having high deviation and value of average execution time, they were executed without clustering. Tasks with lower deviation and value of execution time were clustered together. They showed that clustering tasks for data-intensive application workflows has better makespan than scheduling the workflow without clustering, mainly attributed to the decrease in file transfers between tasks in the same cluster.

Worker Centric

Worker centric approaches exploit locality of interest present in data-intensive environments. Ko, Morales, & Gupta (2007) presented an algorithm where one global scheduler, upon receiving a request from a worker (computation node), calculates the weight of each unscheduled task and chooses the best task to assign to the requesting worker. The weight calculation procedure takes into account the set of files already present at the worker's site and additional files required by the worker for the task. This scheme exploits locality of file access, and thus minimizes both the number of files that need to be transferred as well as prefers workers that accessed the same files in the past. They proposed both deterministic and randomized metrics that can be used with worker-centric scheduling and found that metrics considering the number of file transfers generally gave better performance over metrics considering the overlap between a task and a storage. They experiment with traces of *Coadd* (*SDSS Project*, 2000).

Data Transfer

Researchers have proposed several mechanisms for transferring data so that data transfer time is minimized. These techniques are: (a) data parallelism, (b) data streaming, and (c) data throttling.

Data Parallelism

Data Parallelism denotes that a service is able to process several data fragments simultaneously with a minimal performance loss. This capability involves the processing of independent data on different computing resources. Glatard, Montagnat, Lingrand, & Pennec (2008) designed and implemented a workflow engine named *MOTEUR*. They propose algorithms that combine well-defined data composition strategies and fully parallel execution. They adopted the Simple Concept Unified Flow Language (SCUFL) as the workflow description language for conveniently describing data flows. In their system, tasks and data are scheduled such that most data sets are processed by independent computing resources, but by preserving the precedence constraints. They evaluated the system using a medical imaging application run on the EGEE (Enabling Grids for E-Science EU IST project, http://www.eu-egee.org) grid.

Data Streaming

In data streaming, real-time data generated through simulation or experiment is delivered in an asynchronous, high-throughput, low-latency and robust way to data analysis and storage machines. Bhat et al. (2007) investigated data streaming for executing scientific workflows on the Grid. They proposed the design, implementation and experimental evaluation of an application level self-managing data streaming service that enables efficient data transport to support Grid-based scientific workflows. The system provides adaptive buffer management mechanisms and proactive QoS management strategies based on

model-based online control and user-defined policies. They showed that online data streaming could have significant impact on the performance and robustness of the data-intensive application workflow applications in Grids. They used a fusion simulation workflow consisting of long-running coupled simulations to evaluate the data streaming service and its self-managing behaviors.

Bhat, Parashar, & Klasky (2007) investigated reactive management strategies for in-transit data manipulation for data-intensive scientific and engineering workflows. Their framework for in-transit manipulation consists of processing nodes in the data path between the source and the destination. Each node is capable of processing, buffering and forwarding the data. Each node processes the data depending on its capabilities and the amount of processing still remaining. The data is dynamically buffered as it flows through the node. Eventually the processed data is forwarded until it reaches the sink. The choice between forwarding and further processing is dependent upon the network congestion. They used application level online controllers for high throughput data streaming.

Korkhov et al. (2007) & Afsarmanes et al. (2002) proposed Grid-based Virtual Laboratory AMsterdam (VLAM-G), a data-driven WfMS. Their system uses Globus services (*Globus Project*, 1996) to allow data streams to be established efficiently and transparently between remote processes composing a scientific workflow. The execution engine initiates 'point-to-point' data streams between workflow components allowing intermediate data to flow along the workflow pipeline, without requiring local storage. They use unidirectional, typed streams to ensure that proper connection can be established. Control and monitoring communication is not transmitted on such typed streams. They model the system such that all the resources needed for data stream driven distributed processing have to be made available (e.g. by advance reservation) simultaneously in contrast to the scenario where Grid resources join and leave anytime.

Data Throttling

Data throttling is a process of describing and controlling when and at what rate data is to be transferred in contrast to moving data from one location to another as early as possible. In scientific workflows with data-intensive workload, individual tasks may have to wait for large amounts of data to be delivered or produced by other tasks. Instead of transferring the data immediately to a task, it can be delayed or transferred using lower capacity links so that the resources can be dedicated to serve other critical tasks.

Park & Humphrey (2008) identified the limitation of current systems in that there is no control available regarding the arrival time and rate of data transfer between nodes. They designed and implemented new capabilities for higher efficiency and balance in Grid workflows by creating a data-throttling framework that regulates the rate of data transfers between the workflow tasks via a specially created QoS-enabled GridFTP server. Their workflow planner constructs a schedule that both specify when/where individual tasks are to be executed, as well as when and at what rate data is to be transferred. The planner allows a workflow programmer/engine to specify the requirements on the data movement delay. This delay helps keep a balance between execution time of workflow branches by eliminating unnecessary bandwidth usage, resulting in more efficient execution.

DAGMan (Directed Acyclic Graph MANager) (*DagMan*, 2002) is a workflow engine under the Pegasus (Deelman et al., 2005) WfMS. It supports job and data throttling using parameters. Pegasus uses DAGMan to run the executable workflow. In DAGMan a "prescript" and a "postscript" step, associated with each workflow job, are responsible for transferring input files and deleting output files, respectively. It controls the number of prescripts that can be concurrently (across all

jobs) started using the MAXPRE parameter. This serves as a convenient workflow-wide throttle on the data transfer load that the workflow manager can impose on the Grid from the submit host.

Data Footprint

Workflow systems adopt several mechanisms to track and utilize the data footprint of the application. These mechanisms can be classified into: (a) cleaning jobs, (b) restructuring of workflow, (c) data placement & replication.

Cleaning Jobs

Cleaning jobs are introduced in the workflow to remove the data from the resources once its no longer needed. When applications require large amount of data storage, tasks in the workflow can only be scheduled to those compute resources that can provide temporary storage large enough to hold the input and output files the tasks need. Scheduling decisions should take into consideration the storage capability of the compute resource for all tasks with data-intensive workloads.

Singh et al. (2007) presented two algorithms for reducing the data footprint of workflow type applications. The first algorithm adds a cleanup job for a data file when that file is no longer required by other tasks in the workflow or when it has already been transferred to permanent storage. Given the possibility of data being replicated on multiple resources, the cleanup jobs are made on a per resource basis. The algorithm is applied after the executable workflow has been created, but before it is executed. The second algorithm is an improvement in terms of the number of cleanup jobs and dependencies it adds to the workflow. As the workflow engine has to spend considerable amount of time in managing job execution for every added job or dependency, the authors design the algorithm to reduce the number of cleanup tasks at the possible cost of workflow footprint. This is achieved by adding at most one cleanup node per computational workflow task in contrast to one cleanup job for every file required or produced by tasks mapped to the resource as done in the first algorithm. They reduce data footprint but as a consequence the workflow execution time increases as a result of the increased number of workflow levels.

Ramakrishnan et al. (2007) proposed an algorithm for scheduling data-intensive application workflows onto storage-constrained resources. Their algorithm first takes into account disk space availability in resources and then prioritizes resources depending on performance. The algorithm starts by identifying all resources that can accommodate the data files needed for a task to be scheduled. If no resource is available that satisfies the space requirement of any ready task, the algorithm halts. It then tries to allocate the task to the resource, which can achieve the earliest finish time (data transfer time and execution time) for the task. Finally it cleans up any unnecessary data file remaining in the resource.

Restructuring of Workflows

The structure of the workflow defines the data footprint. Restructuring of workflows is a transformation of the workflow structure such that it influences the way input/output data is placed, deleted, transferred or replicated during the execution of the workflow. Task clustering, workflow partitioning are common ways to restructure workflows. Tasks can be clustered and dependencies re-defined in such a way that data transfer is minimized, data re-use is maximized, storage resources and compute resources have well-balanced load and so forth.

Singh et al. (2007) defined workflow restructuring as the ordering or sequencing of the execution of the tasks within the workflow. They restructure the workflow primarily to reduce the data footprint of the workflow. They introduce dependencies between stage-in tasks and the previous-level computational tasks. This prevented

multiple data transfers from occurring at the same time as soon as tasks become ready.

Pegasus (Deelman et al., 2005) has the capability to map and schedule only portions of the entire workflow at a given time, using partitioning techniques. Deelman et al. (2005) demonstrate the technique using level-based partitioning of the workflow. The levels refer to the depth of the tasks in the workflow. In their Just-in-time planning algorithm (Deelman et al., 2004), Pegasus waits (using DAGMan) to map the dependent workflow until the preceding workflow finishes its execution. Original dependencies are maintained even after partitioning. They also investigate partition-level failure recovery. When resources fail during execution, the entire task is retried and new partitions are not submitted to that resource.

Duan, Prodan, & Fahringer (2006) proposed an algorithm for partitioning a scheduled workflow for distributed coordination among several slave enactment engine services. They incorporated the algorithm in the ASKALON distributed workflow Enactment Engine (Duan et al., 2005). Their purpose of workflow partitioning was to minimize the communication between the master and the slave engines that coordinates the individual partitions of the entire workflow. The partitioning algorithm is based on a graph transformation theory. Partitioning reduced the number of workflow activities and, therefore, the job submission and management latencies and eliminated the data dependencies within partitions. However, the algorithm was used for compute intensive scientific workflows with large numbers of small sized data dependencies. In contrast to Pegasus (Deelman et al., 2005), which partitions the workflow before the scheduling phase, they partition the workflow after scheduling. This results in reduced overheads for job submissions and aggregated file transfers.

Data Placement and Replication

Data placement techniques try to strategically manage placement of data before or during the execution of a workflow. Data placement schedulers can either be coupled or decoupled from task schedulers. Replication of data onto distributed resources is a common way to increase the availability of data. Replication also occurs when scientists download and share the data for experimental purposes, in contrast to explicit replications done by workflow systems. In data-intensive applications, replication may or may not be feasible. Schedulers make the decision of data placement and replication based on the objectives to be optimized. If data analysis workloads have locality of reference, then it is feasible to cache and replicate data at each individual compute node, as high initial data movement costs can be offset by many subsequent data operations performed on cached data (Raicu, Zhao, Foster, & Szalay, 2008).

Kosar et al. (2004) presented Stork, a scheduler for data placement activities in the Grid. They propose to make data placement activities a first class citizen in the Grid. In Stork, data placement is a full-fledged job and decoupled from computational jobs. Users describe the data placement job explicitly in the *classads*. DAGMan (*DagMan*, 2002), a workflow scheduler for Condor, uses Stork for managing these data placement jobs. It manages the dependencies between Condor and Stork jobs as defined by the dependencies in a DAG (Couvares, Kosar, Roy, Weber, & Wenger, 2007). Under Stork, data placement jobs are categorized into three types. *Transfer* jobs are for transferring a complete or partial file from one physical location to another. *Allocate* jobs are used for allocating storage space at the destination site, allocating network bandwidth, or establishing a light-path on the route from source to destination. *Release* jobs are used for releasing the corresponding resource, which was allocated before.

Chervenak et al. (2007) studied the relationship between data placement services and workflow management systems for data-intensive applications. They propose an *asynchronous* mode of data placement in which data placement opera-

tions are performed as data sets become available and according to the policies of the virtual organization and not according to the directives of the WfMS. The WfMS can however assist the placement services on placement of data based on information collected during task executions and data collection. Their approach is proactive as it examines current workflow needs to make data placement decisions rather than depending on the popularity of data in the past. They evaluated the benefits of pre-staging data using the data replication service versus using the native data stage-in mechanisms of the Pegasus WfMS (Deelman et al., 2005). Using the Montage astronomy example, they conclude that as the size of data sets increases, pre-staging data increases the performance of the overall analysis.

Shankar & DeWitt (2007) presented architecture for Condor in which the input, output and executable files of jobs are cached on the local disks of machines in a cluster. Caching can reduce the amount of pipeline and batch I/O that is transferred across the network. This in turn significantly reduces the response time for workflows with data-intensive workloads. With caching enabled, data-intensive applications can reuse the files and also be able to compare between old and new versions of the file. They presented a planning algorithm that takes into account the location of cached data together with data dependencies between jobs in a workflow. Their planning algorithm produces a schedule by comparing the time saved by running jobs in parallel with the time taken for transferring data when dependent jobs are scheduled on different machines. By executing the BLAST (http://blast.ncbi.nlm.nih.gov.gov/) application workflow they showed that storing files on the disks of compute nodes significantly improves the performance of data-intensive application workflows.

Ranganathan & Foster (2001, 2002, 2003) conducted extensive studies for identifying dynamic replication strategies, asynchronous data placement and job and data scheduling algorithms for Data Grids. Their replication process at each site periodically generates new replicas for popular datasets. For dataset placement scheduler they define three algorithms: *Data-DoNothing*- no active replication takes place, *DataRandom*- popular datasets are replicated to a random site on the Grid, *DataLeastLoaded*- popular datasets are replicated to a least loaded neighboring site. They proposed to decouple data movement from computation scheduling, also known as asynchronous data placement. This provides opportunity for optimizing both data placement and scheduling decisions, also simplifying the design and implementation of the Data Grid system. They concluded through simulations on independent jobs that scheduling jobs to locations that contain the data they need and asynchronously replicating popular data sets to remote sites achieves better performance than coupled systems.

Granularity

Workflow schedulers can make scheduling decisions based on either: (a) task level, or (b workflow level.

Task level schedulers map individual tasks to compute resources. The decision of resource selection and data movement is based on the characteristics of individual task and its dependencies with other tasks.

Workflow level schedulers map the entire workflow rather than a set of available tasks to compute resources. A workflow's compute and storage requirements guide the scheduler to make a decision on resource selection and data movement.

Blythe et al. (2005) compared several task-based and workflow-based approaches to resource allocation for workflow applications. In their workflow-based approach, the entire workflow is mapped a priori to the resources to minimize the makespan of the whole workflow. The mapping is changed according to the changing environment, if necessary. The mapping of the jobs does not imply scheduling all the jobs ahead of time.

They use a local search algorithm for workflow allocation based on generalized GRASP procedure (Greedy randomized adaptive search) (Feo & Resende, 1995). The final schedule is chosen after an iterative and greedy comparison between alternative schedules. On each iteration, task to resource is mapped based on the minimum margin of increase to the current makespan of the workflow if the task was to be allocated to that resource. This approach is based on the *min-min* (Braun et al., 2001) heuristic. They noticed that during large file transfers, resources spent significant time waiting for all the files to arrive before they could start executing the scheduled jobs. They proposed a *weighted min-min* heuristic that takes into account the idle times of all the resources if a job were to be scheduled to a resource. Based on the weighted sum of the idle times and estimated completion time, a job is mapped to the resource that gives the minimum weighted sum. The step is repeated until all the jobs have been mapped. Due to the pre-mapping, the workflow-based approach could pre-position the data to the known destination by transferring a large file immediately after it is created. In the task-based approach, transfers could not begin until the job is scheduled which happened only after its parent was scheduled. They also simulated the impact of inaccurate estimates of transfer times for data-intensive application workflows. They show that the performance of task-based approach degrades rapidly with increasing uncertainty in comparison to workflow-based approach. Based on these facts, they conclude that workflow-based approaches perform better for data-intensive applications than task-based approaches.

Model

Workflow scheduling model depends on the way the tasks and data are composed and handled. They can be classified into two categories: (a) task-based, and (b) service-based.

Task Based

Task based approaches mention data dependencies explicitly. The workflows are generally complex in structure. Optimizations used by most systems are simple in nature. The WfMS has greater control over the data flow as it can define data placement, cleanup and transfer tasks separately from the workflow tasks. DAGMan (*DagMan,* 2002), Pegasus (Deelman et al., 2005), GridAnt (Laszewski, Amin, Hategan, Hampton, & Rossi, 2004), GrADS (Berman et al., 2005), and GridFlow (Cao, Jarvis, Saini, & Nudd, 2003) are some of the workflow systems that support task based approaches. These have been described individually in preceding sections.

Service Based

Service based approaches, also referred to as *meta computing*, wrap application codes into standard interfaces. Such services are hidden from the users and only invocation interface is known. Various interfaces such as Web Services (Alonso, Casati, Kuno, & Machiraju, 2003) or gridRPC (Nakada et al., 2007) have been standardized (Glatard et al., 2008). In this model, the application is described separately from the data. Data is declared as parameters to the service. In this approach, workflows are generally simple in structure. In contrast to task based approaches, workflow systems use complex optimizations. This model is useful when an application workflow is to be repeatedly executed over a large number of varying data sets. Instead of replicating the task for each data set, service based model has the ability to define different data composition strategies over the input data of a service. Kepler system (Ludäscher et al., 2006), the Taverna workbench (Oinn et al., 2004) and the Triana workflow manager (Taylor, Wang, Shields, & Majithia, 2005), are some of the service based workflow systems.

The myGrid project (http://www.mygrid.org.uk/) has developed a comprehensive loosely cou-

pled suite of middleware components specifically to support data-intensive in-silico experiments represented as workflows, using distributed resources. The main tool is the Taverna workbench (Oinn et al., 2004). Taverna allows for the automation of experimental methods through the integration of varying services, including WSDL-based single operation web services, into workflows. It uses FreeFluo (FreeFluo, 2003) as a workflow enactment engine that facilitates intermediate data transfers and service invocations. Workflows are represented using the Simple Conceptual Unified Flow Language (SCUFL). A workflow graph consists of processors, each of which transforms a set of data inputs into a set of data outputs. Using SCUFL, implicit iteration over incoming data sets can be carried out based on user specified strategy. Users can use the Thread property to specify the number of concurrent instances that can send parallel requests to the iteration processor for handling simultaneous processing. This can help reduce the service wait time as workflow engine can send data at the time when the service is still working on previously sent data.

Kepler (Ludäscher et al., 2006) provides support for web service-based workflows. Using an extension of PTOLEMY II (Buck, Ha, Lee, & Messerschmitt, 2002), it uses an actor-oriented design approach for composing and executing scientific application workflows. Computational components are termed as actors, which are linked together to form a workflow. A director represents the interaction between these components. It specifies and mediates all inter-actor communication, separating workflow orchestration and scheduling from individual actor execution. Two of the directors (namely, Synchronous Data Flow (SDF) and Process Networks (PN)) work primarily by controlling the sequencing of actors according to the data availability, to preserve the order of execution of the workflow. The *WebService* actor provides a simple plug-in mechanism to execute any WSDL defined web service. An instantiation of the actor acts as a proxy for the web service being executed and links to other actors through its ports. Using this component, any application that can be deployed as a remote service, can be used as a Kepler component (Jaeger et al., 2005).

Kalyanam, Zhao, Park, & Goasguen (2007) proposed a web service-enabled distributed data-driven workflow system on top of the TeraGrid (http://www.teragrid.org) infrastructure. The workflow system is based on an existing data management architecture that provides easy access to scientific data collections via the TeraGrid network. It leverages JOpera (Pautasso, 2005), an open-source workflow engine that integrates web services into a processing pipeline. Users can construct data-driven workflows using local or TeraGrid data and computation resources. Their system helps automate the operations such as data discovery, movement, filtering, computationally intensive data processing and so forth, by organizing them as a pipeline so that researchers can execute applications with minimal user interaction.

Brandic, Pllana & Benkner (2008) presented a service-oriented environment, named as Amadeus, for QoS-aware Grid workflows. For data-intensive application workflows QoS parameters may be defined for data-transfer time, reliability, storage requirements, cost and so forth. It allows users to specify QoS constraints at workflow composition, planning and execution stages. Various QoS-aware service components are provided for workflow planning to support automatic constraint-based service negotiation and workflow optimization.

Platform

Data-intensive application workflows could be executed in different resource configuration and environments (e.g. Cluster, Data Grids, Clouds etc.) depending on the requirements of the application.

Clusters are generally composed of homogeneous processors and are under a single domain. For data-intensive applications, clusters provide a viable platform for low cost and enhanced per-

formance. When the data produced and stored are local and not globally shared, cluster based platforms is more feasible than Grids or Clouds.

Data Grids are globally distributed resources for volunteering computing designed for data intensive computing. Data is generated and/or used in research labs distributed globally, giving rise to sharing and re-use. Data grids are feasible for large-scale experiments that are a result of worldwide collaboration of resources and scientists.

Clouds are emerging model for centralized but highly available and powerful infrastructure. Large-scale storage and computation is provided by data-centers. Computing power is achieved by using virtualization technology. Data-intensive applications can highly benefit from using services provided by Clouds as compared to Data Grids and Clusters when factors such as scalability, cost, performance and reliability are important.

In the past, scientific workflows were generally executed on a shared infrastructure such as TeraGrid (http://www.teragrid.org), Open Science Grid (http://www.opensciencegrid.org), and dedicated clusters. In such systems, file system is usually shared for easy data movement. However, this can be a bottleneck for data-intensive operations (Zhao, Raicu, & Foster, 2008).

Deelman, Singh, Livny, Berriman, & Good (2008) presented a simulation-based study of costs involved when executing scientific application workflows using Cloud services. They studied the cost performance tradeoff of different execution and resource provisioning plans, and storage and communication fees of Amazon S3 in the context of an astronomy application Montage. They showed that for a data-intensive application with a small computational granularity, the storage costs were insignificant as compared to the CPU costs. They concluded that cloud computing is cost-effective solution for data-intensive applications.

Broberg, Buyya, & Tari (2008) introduced MetaCDN, which uses 'Storage Cloud' resources to deliver content to content creators at low cost but with high performance (in terms of throughput and response time). Data could be delivered to tasks in a workflow using tools provided by CDN.

In our work with data-intensive application workflows, we studied the performance characteristics of a brain Image Registration workflow (IR) (Pandey et al., 2009). We executed the application on an experimental Grid platform, Grid'5000 (Cappello & Bal, 2007), and profiled each task's execution and data flow. We were able to decrease the makespan of the workflow significantly by using Grid resources. We also used partial data retrieval technique to retrieve data from distributed storage resources while scheduling data-intensive application workflows (Pandey & Buyya, 2011). We proposed static and dynamic heuristics that incorporated the retrieval techniques. We experimented with two synthetic and one real data-intensive application workflow (IR workflow). Executions were done using Virtual Machines (VM) connected through a simulated network environment. Experimental results showed that retrieving data from multiple sources significantly improves the time taken to download data to the execution sites. Cumulative effect thus decreased the total makespan of all the workflows.

Ramakrishnan & Reed (2008) studied the impact of varying resource availability on application performance. They applied *performability* analysis (i.e., a measure of the system's performance in the event of failures) at two levels - computational resources and the network, to obtain the application workflow's overall execution time, given the failure level of resources. They used these values to estimate task completion times during each iteration of the workflow-scheduling algorithm. Their *HYBRID* approach, which takes resource failure and repair into account, performs better than the approach that does not take failures into account, when the failure-to-repair rates increase. Through simulation results, they concluded that the joint analysis of performance and reliability should improve dynamic workflow scheduling

and fault tolerance strategies required for Grid and cloud environments.

Miscellaneous

In this Section, we list some technologies that have been used for enhancing the performance of data-intensive application workflows.

Semantic Technology

The myGrid project (http://www.mygrid.org.uk/) exploits semantic web technology to support data-intensive bioinformatics experiments in a grid environment. The semantic description of services in RDF and OWL is used for service discovery and matchmaking. Kepler (Ludäscher et al., 2006) is a data-driven workflow system (as described under the sub heading "Service Based" workflows), which allows semantic annotations of data and actors, and can support semantic transformation of data.

Database Technology

GridDB (Liu & Franklin, 2004) is a grid middleware based on a data centric model for representing workflows and their data. It uses database to store *memo* and *process* tables that store the inputs and outputs of a program that has completed, and process state of executing programs, respectively. It provides functional data modeling language (FDM) for expressing the relationship between programs and their inputs and outputs.

Shankar, Kini, DeWitt, & Naughton (2005) have pointed out the advantages of tightly coupling workflow management systems with data-manipulation for data-intensive scientific programs. They also presented a language for modeling workflows that is tightly integrated with SQL. Data products from workflows are defined in relational format. They use SQL for invocation and querying of programs.

FUTURE DIRECTIONS

Most workflow systems in the past focused on performance of tasks rather than data management. The reason might have been due to cluster management systems and shared storage space. But with globally distributed resources, it is a must these systems take into account the data flow management along with computational tasks. Composition of workflows that is scalable thus remains a challenge. Distributed coordinated execution of globally distributed scientific workflows can then be possible without much hurdle.

Requirements of data-intensive applications can be specified using QoS parameters at all levels of a WfMS. To meet QoS requirements of e-Research application workflows, we need technologies that support (a) QoS-based scheduling of e-Research application workflows on distributed resources, (b) mechanisms for formulating, negotiating and establishing service level agreements (SLA) with resource providers and (c) SLA-based allocation and management of resources. Specifically, we need to:

- Define an architectural framework and principles for the development of QoS-based workflow management and SLA-based resource allocation systems,
- Develop QoS-based algorithms for scheduling e-Research workflow applications,
- Develop SLA-based negotiation protocols and resource allocation algorithms.

With the advent of virtualization technologies, Cloud storage systems, content delivery networks (CDN) and so forth, it is likely that big scientific projects will start using services provided by third parties for storing and processing application data. As companies such as Amazon, IBM, and Google are innovating the use of their huge data centers for commercial use as Cloud services, data-intensive applications may leverage their utilities and not depend on conventional, error-prone, costly and

unreliable solutions (Buyya, Yeo and Venugopal, 2008). However, due to higher usage and access costs of these commercial services, small-scale scientific projects may still need to rethink of deploying their application on Clouds.

CONCLUSION

In this chapter, we classified and surveyed techniques for managing and scheduling data-intensive application workflows. Under each classification, there were several specific techniques that workflow systems used for executing data-intensive application workflows on globally distributed resources. We listed and described each such work in detail. We found that most systems focused on minimizing data transfers and optimally structuring model of execution to subdue the effect of large data requirements of most scientific data-intensive applications. We also found that many systems used a combination of techniques we listed to achieve higher scalability, fault tolerance, lower costs and increase performance. A single technique alone would not suffice to minimize the effect of increasing data processing requirements of scientific applications. Due to the lack of standardization and interoperability, many of the systems were developed in isolation. As a result, techniques for managing data for data-intensive workflows were mixed and duplicated. Nevertheless, scientific community has been able to successfully achieve the goals of all scientific projects with promising results, where PetaByes of data play a major role. This was only possible due to the seamless effort put on for the development of workflow management systems that manages data and tasks for most scientific applications.

ACKNOWLEDGMENT

This work is partially supported through Australian Research Council (ARC) Discovery Project grant. We also thank Ivona Brandic from University of Vienna, Austria; Marcos Assunção, Srikumar Venugopal, Rajiv Ranjan and Marco A.S. Netto from The University of Melbourne, Australia for their valuable comments.

REFERENCES

Afsarmanesh, H., Belleman, R. G., Belloum, A. S. Z., Benabdelkader, A., Brand, J. F. J., & van den Eijkel, G. B. (2002). Vlam-g: A grid-based virtual laboratory. *Science Progress*, *10*(2), 173–181.

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2003). *Web services - Concepts, architectures and applications*. Springer.

Bahsi, E. M., Ceyhan, E., & Kosar, T. (2007). Conditional workflow management: A survey and analysis. *Science Progress*, *15*(4), 283–297.

Berman, F., Casanova, H., Chien, A., Cooper, K., Dail, H., & Dasgupta, A. (2005). New grid scheduling and rescheduling methods in the grads project. *International Journal of Parallel Programming*, *33*(2), 209–229. doi:10.1007/s10766-005-3584-4

Bhat, V., Parashar, M., & Klasky, S. (2007). Experiments with in-transit processing for data intensive grid workflows. In *GRID* (pp. 193-200). IEEE.

Bhat, V., Parashar, M., Liu, H., Kandasamy, N., Khandekar, M., & Klasky, S. (2007). A self-managing wide-area data streaming service. *Cluster Computing*, *10*(4), 365–383. doi:10.1007/s10586-007-0023-x

Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., et al. (2005). Task scheduling strategies for workflow-based applications in grids. In *CCGRID '05: Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05) - Volume 2* (pp. 759–767). Washington, DC: IEEE.

Brandic, I., Pllana, S., & Benkner, S. (2006). An approach for the high-level specification of qos-aware grid workflows considering location affinity. *Science Progress*, *14*(3/4), 231–250.

Brandic, I., Pllana, S., & Benkner, S. (2008). Specification, planning, and execution of qos-aware grid workflows within the Amadeus environment. *Concurrent Computing: Practice and Experience*, *20*(4), 331–345. doi:10.1002/cpe.1215

Braun, T. D., Siegel, H. J., Beck, N., Boloni, L. L., Maheswaran, M., & Reuther, A. I. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *Journal of Parallel and Distributed Computing*, *61*(6), 810–837. doi:10.1006/jpdc.2000.1714

Broberg, J., Buyya, R., & Tari, Z. (2008, August). *MetaCDN: Harnessing 'storage clouds' for high performance content delivery* (Tech. Rep. No. GRIDS-TR-2008-11). GRIDS Lab: The University of Melbourne.

Buck, J., Ha, S., Lee, E. A., & Messerschmitt, D. G. (2002). *Ptolemy: A framework for simulating and prototyping heterogeneous systems* (pp. 527–543). Norwell, MA: Kluwer Academic Publishers.

Buyya, R., Yeo, C. S., & Venugopal, S. (2008). Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *HPCC '08: Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications*, (pp. 5-13). Washington, DC: IEEE.

Cao, J., Jarvis, S. A., Saini, S., & Nudd, G. R. (2003). Gridflow: Workflow management for grid computing. In *CCGRID '03: Proceedings of the 3st International Symposium on Cluster Computing and the Grid* (pp. 198–205). Washington, DC, USA.

Cappello, F., & Bal, H. (2007). Toward an international "computer science grid". In *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid* (pp. 3–12). Washington, DC: IEEE.

Chervenak, A., Deelman, E., Livny, M., Su, M.-H., Schuler, R., Bharathi, S., et al. (2007, September). Data placement for scientific applications in distributed environments. In *Proceedings of the 8th IEEE/ACM International Conference on Grid Computing (Grid 2007)*. Austin, TX: IEEE.

Couvares, P., Kosar, T., Roy, A., Weber, J., & Wenger, K. (2007, January). Workflow management in condor. In Workflows for e-Science (pp. 357–375). London, UK: Springer. doi:10.1007/978-1-84628-757-2_22

DagMan. (2002). *Online*. Retrieved from http://www.cs.wisc.edu/condor/dagman/

Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., et al. (2004). Pegasus: Mapping scientific workflows onto the grid. In *European Across Grids Conference* (vol. 3165, pp. 11–20). Springer.

Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., & Vahi, K. (2003). Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, *1*(1), 25–39. doi:10.1023/A:1024000426962

Deelman, E., & Chervenak, A. (2008). Data management challenges of data-intensive scientific workflows. In *CCGRID '08: Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)* (pp. 687–692). Washington, DC: IEEE Computer Society.

Deelman, E., Singh, G., Livny, M., Berriman, B., & Good, J. (2008). The cost of doing science on the cloud: The montage example. In *SC '08: Proceedings of the 2008 ACM/IEEE Conference on Supercomputing* (pp. 1–12). Piscataway, NJ: IEEE.

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., & Kesselman, C. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Science Progress*, *13*(3), 219–237.

Duan, R., Fahringer, T., Prodan, R., Qin, J., Villazon, A., & Wieczorek, M. (2005, February). *Real world workflow applications in the Askalon grid environment.* In European Grid Conference (EGC 2005). Springer Verlag.

Duan, R., Prodan, R., & Fahringer, T. (2006). Runtime optimisation of grid workflow applications. In *GRID* (pp. 33-40). IEEE.

Feo, T. A., & Resende, M. G. (1995, March). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*(2), 109–133. doi:10.1007/BF01096763

FreeFluo. (2003). *Online*. Retrieved from http://freefluo.sourceforge.net/

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., & Gannon, D. (2007). Examining the challenges of scientific workflows. *Computer*, *40*(12), 24–32. doi:10.1109/MC.2007.421

Glatard, T., Montagnat, J., Lingrand, D., & Pennec, X. (2008). Flexible and efficient workflow deployment of data-intensive applications on grids with moteur. *International Journal of High Performance Computing Applications*, *22*(3), 347–360. doi:10.1177/1094342008096067

Globus Project. (1996). *Online*. Retrieved from http://www.globus.org/

Hollingsworth, D. (1994). *The workflow reference model. (Tech. Rep. No. TCOO- 1003)*. Workflow Management Coalition.

Jaeger, E., Altintas, I., Zhang, J., Ludäscher, B., Pennington, D., & Michener, W. (2005). A scientific workflow approach to distributed geospatial data processing using web services. In *SSDBM'2005: Proceedings of the 17th International Conference on Scientific and Statistical Database Management* (pp. 87–90). Berkeley, CA: Lawrence Berkeley Laboratory.

Kalyanam, R., Zhao, L., Park, T., & Goasguen, S. (2007). A web service-enabled distributed workflow system for scientific data processing. In *FTDCS '07: Proceedings of the 11th IEEE International Workshop on Future Trends of Distributed Computing Systems* (pp. 7–14). Washington, DC: IEEE.

Ko, S. Y., Morales, R., & Gupta, I. (2007). New worker-centric scheduling strategies for data-intensive grid applications. In Cerqueira, R., & Campbell, R. H. (Eds.), *Middleware* (*Vol. 4834*, pp. 121–142). Springer. doi:10.1007/978-3-540-76778-7_7

Korkhov, V., Vasyunin, D., Wibisono, A., Belloum, A. S. Z., Inda, M. A., & Roos, M. (2007). Vlam-g: Interactive data driven workflow engine for grid-enabled resources. *Science Progress*, *15*(3), 173–188.

Kosar, T., & Livny, M. (2004). Stork: Making data placement a first class citizen in the grid. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing* Systems *(ICDCS'04)* (pp. 342–349). Washington, DC: IEEE.

Kwok, Y. K., & Ahmad, I. (1999). Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys*, *31*(4), 406–471. doi:10.1145/344588.344618

Laszewski, G. V., Amin, K., Hategan, M., Hampton, N. J. Z. S., & Rossi, A. (2004, January). Gridant: A client-controllable grid workflow system. In *37th Hawaii International Conference on System Science (HICSS'04)* (pp. 5–8). IEEE.

Liu, D. T., & Franklin, M. J. (2004). Griddb: A data-centric overlay for scientific grids. In *VLDB '04: Proceedings of the Thirtieth International Conference on Very Large Data Bases* (pp. 600–611). VLDB Endowment.

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., & Jones, M. (2006). Scientific workflow management and the kepler system: Research articles. *Concurrency and Computation*, *18*(10), 1039–1065.

Meyer, L., Annis, J., Wilde, M., Mattoso, M., & Foster, I. (2006). Planning spatial workflows to optimize grid performance. In *SAC '06: Proceedings of the 2006 ACM Symposium on Applied Computing* (pp. 786–790). New York, NY: ACM.

Moore, R., Prince, T. A., & Ellisman, M. (1998). Data-intensive computing and digital libraries. *Communications of the ACM*, *41*(11), 56–62. doi:10.1145/287831.287840

Nakada, H., Matsuoka, S., Seymour, K., Dongarra, J., Lee, C., & Casanova. (2007, June). *A GridRPC model and API for end-user applications.* GridRPC Working Group of Global Grid Forum.

Oinn, T., Addis, M., Ferris, J., Marvin, D., Senger, M., & Greenwood, M. (2004, November). Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics (Oxford, England)*, *20*(17), 3045–3054. doi:10.1093/bioinformatics/bth361

Pandey, S, & Buyya. R. (2011, in press). Scheduling Workflow Applications based on Multi-Source Parallel Data Retrieval in Distributed Computing Networks, *The Computer Journal*.

Pandey. S, Voorsluys, W., Rahman, M., Buyya, R., Dobson, J., Chiu, K. (2009, November) A Grid Workflow Environment for Brain Imaging Analysis on Distributed Systems. *Concurrency and Computation: Practice and Experience*, *21*(16), 2118-2139.

Pautasso, C. (2005). Jopera: An agile environment for Web service composition with visual unit testing and refactoring. In *VLHCC '05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing* (pp. 311–313). Washington, DC: IEEE Computer Society.

Raicu, I., Zhao, Y., Foster, I. T., & Szalay, A. (2008). Accelerating large-scale data exploration through data diffusion. In *DADC '08: Proceedings of the 2008 International Workshop on Data-Aware Distributed Computing* (pp. 9–18). New York, NY: ACM.

Ramakrishnan, A., Singh, G., Zhao, H., Deelman, E., Sakellariou, R., Vahi, K., et al. (2007). Scheduling data-intensive workflows onto storage-constrained distributed resources. In *CCGrid '09: Proceedings of the 7th IEEE Symposium on Cluster Computing and The Grid* (pp. 14–17). Brazil: IEEE.

Ramakrishnan, L., & Reed, D. A. (2008). Performability modeling for scheduling and fault tolerance strategies for scientific workflows. In *HPDC '08: Proceedings of the 17th International Symposium on High Performance Distributed Computing* (pp. 23–34). New York, NY: ACM.

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

Ranganathan, K., & Foster, I. (2003, March). Simulation studies of computation and data scheduling algorithms for data grids. *Journal of Grid Computing*, *1*(1), 53–62. doi:10.1023/A:1024035627870

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

SDSS Project. (2000). *Online*. Retrieved from https://www.darkenergysurvey.org

Shankar, S., & DeWitt, D. J. (2007). Data driven workflow planning in cluster management systems. In *HPDC '07: Proceedings of the 16th International Symposium on High Performance Distributed Computing* (pp. 127–136). New York, NY: ACM.

Shankar, S., Kini, A., DeWitt, D. J., & Naughton, J. (2005). Integrating databases and workflow systems. *SIGMOD Record*, *34*(3), 5–11. doi:10.1145/1084805.1084808

Simmhan, Y. L., Plale, B., & Gannon, D. (2005, September). A survey of data provenance in e-science. *SIGMOD Record*, *34*(3), 31–36. doi:10.1145/1084805.1084812

Singh, G., Kesselman, C., & Deelman, E. (2005, September). Optimizing grid-based workflow execution. *Journal of Grid Computing*, *3*(3-4), 201–219. doi:10.1007/s10723-005-9011-7

Singh, G., Vahi, K., Ramakrishnan, A., Mehta, G., Deelman, E., & Zhao, H. (2007). Optimizing workflow data footprint. *Science Progress*, *15*(4), 249–268.

Taylor, I., Wang, I., Shields, M., & Majithia, S. (2005). Distributed computing with triana on the grid: Research articles. *Concurrency and Computation*, *17*(9), 1197–1214. doi:10.1002/cpe.901

Venugopal, S., Buyya, R., & Ramamohanarao, K. (2006). A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Computing Surveys*, *38*(1). doi:10.1145/1132952.1132955

Venugopal, S., Buyya, R., & Winton, L. (2006). A grid service broker for scheduling e-science applications on global data grids: Research articles. *Concurrency and Computation*, *18*(6), 685–699. doi:10.1002/cpe.974

Yu, J., & Buyya, R. (2004). A novel architecture for realizing grid workflow using tuple spaces. *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing*.

Yu, J., & Buyya, R. (2005). A taxonomy of scientific workflow systems for grid computing. *SIGMOD Record*, *34*(3), 44–49. doi:10.1145/1084805.1084814

Yu, J., Buyya, R., & Ramamohanarao, K. (2008). Workflow scheduling algorithms for grid computing. In *Metaheuristics for Scheduling in Distributed Computing Environments* (*Vol. 146*, pp. 173–214). Berlin, Germany: Springer. doi:10.1007/978-3-540-69277-5_7

Zhao, Y., Raicu, I., & Foster, I. (2008). Scientific workflow systems for 21st century, new bottle or new wine? In *SERVICES '08: Proceedings of the 2008 IEEE Congress on Services - Part I* (pp. 467–471). Washington, DC: IEEE.