Volume: 08, Issue: 05 Pages: 114-119 (2017) Special Issue

Efficacious Management of Grid Resources by Grid Venture

M. Janaki
Assistant Professor, Department of Computer Science,
Dr. Umayal Ramanathan College for Women, Karaikudi.
mjanaki81@gmail.com

ABSTRACT

Grids have emerged as a global cyber-infrastructure for the new-generation of e-Science applications by integrating large-scale, distributed and heterogeneous resources. Grid Computing enables scientists and engineers to build more and more complex applications to manage and process large data sets, and execute scientific experiments on distributed resources. Such applications include complex workflows of resources involved. Therefore, more efforts are taken to develop several workflow management systems for Grid computing. In this paper, we propose a taxonomy called Grid Venture that captures and classifies various approaches for building and executing workflows on Grids. The taxonomy not only focuses on the design and engineering similarities and differences in Grid workflow systems, but also identifies the areas that need further research.

Keywords: **grid computing, resource management, scheduling, taxonomy, workflow management.**

I. INTRODUCTION

Grids have emerged as a global cyber-infrastructure for the new-generation of e-Science applications by integrating large-scale, distributed and heterogeneous resources. Scientific communities, such as high energy physics, geo physics, astronomy and bioinformatics, are utilizing Grids to share, manage and process large data sets. In order to support complex scientific experiments, distributed resources such as computational devices, data, applications, and scientific instruments need to be orchestrated while managing the application workflow operations within Grid environments [2].

Workflow deals with the automation of procedures to pass files and data between users according to a defined set of rules to achieve an overall goal [3]. A workflow management system defines, manages and executes workflows on computing resources. Applying the workflow paradigm for Grid resources management offers several advantages such as:

- Ability to build dynamic applications which orchestrate distributed resources.
- To reduce execution costs.
- To obtain specific processing capabilities.
- Integration of multiple teams involved in managing of different parts of the experiment workflow

In the recent years, several Grid workflow systems have been developed for defining, managing and executing scientific workflows. In order to enhance the way of our understanding of the field, we propose a taxonomy that primarily (a) captures architectural styles and (b) identifies
design and engineering similarities and differences between them. There are a number of proposed taxonomies [5][7][10] for distributed and heterogeneous computing, However, none of these focuses on distributed workflow managements. This taxonomy named Grid Venture provides an in-depth understanding of building and executing workflows on Grids.

The rest of the paper is organized as follows: Section 2 presents the taxonomy that classifies approaches based on major functions and architectural styles of Grid workflow systems. In Section 3, we provide a detailed survey of several selected Grid workflow systems and the mapping of the proposed taxonomy to the systems. We conclude in Section 4 with a discussion and identification of areas that need further work.

II. GRID VENTURE TAXONOMY

This taxonomy captures and classifies approaches of workflow management in the context of Grid computing. It consists of five elements of a Grid workflow management system: (a) workflow design, (b) information retrieval, (c) resource scheduling, (d) fault tolerance and (e) data transfer. In this section, we look at each element and its taxonomy in detail. The fig 1 shows the five key components of grid venture taxanomy.

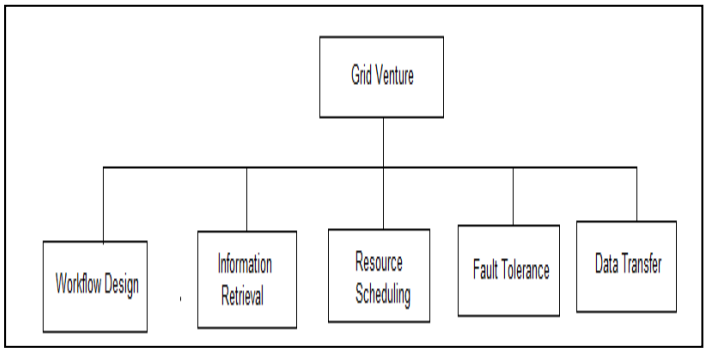


Fig : 1 Workflow Design

Workflow design includes four key factors, namely (a) workflow structure, (b) workflow model, (c) workflow composition system, and (d) workflow QoS (Quality of Service) constraints.

2.1.1 Workflow structure

A workflow is composed of multiple tasks connected according to their dependencies. The workflow structure, also referred as workflow pattern [6], indicates the temporal relationship between these tasks. Figure 4 shows the workflow structure taxonomy. In general, a workflow can be represented as a Directed Acyclic Graph (DAG) [8] or a non-DAG. In DAG-based workflow, workflow structure can be classified as sequence, parallelism, and choice. Sequence can be defined as an ordered series of tasks, with one task starting after a previous task has completed. Parallelism represents tasks which are performed simultaneously, rather than serially. In choice control pattern, a task is selected to execute at run-time when its associated conditions are true. In addition to all patterns found in a DAG-based workflow, a non-DAG workflow also contains the iteration structure in which sections of workflow tasks in an iteration block are allowed to be repeated. Iteration is also known as loop or cycle. The iteration structure is quite frequently used in scientific applications, where one or more tasks need to be executed repeatedly [9].

2.1.2 Workflow Model

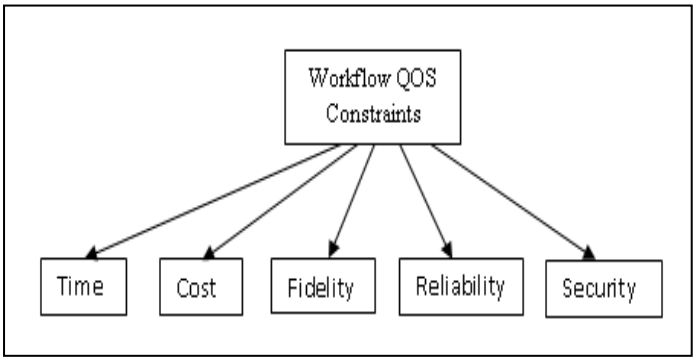
Workflow Model (also called workflow specification) defines a workflow including its task definition and structure definition. As shown in Figure 6, there are two types of workflow models, namely abstract and concrete. They are also referred to as abstract workflows and concrete workflows [12]. In an abstract model, a workflow is defined in an abstract form in which the workflow is provided without referring to specific Grid resources for task execution. An abstract model provides a flexible way for users to define workflows by hiding low-level implementation details. Tasks in an abstract model are portable and can be mapped onto any suitable Grid services at run-time by using suitable discovery and mapping mechanisms. Using abstract models also eases the sharing of workflow descriptions between Grid users [11]; in particular it benefits the participants of Virtual Organizations (VOs).
In contrast, a concrete model binds workflow tasks to specific resources. Concrete models are referred to as executable workflows. In some cases, a concrete model may include tasks acting as data movement to transfer data in and out of the computation and application movement to transfer computational code to a data site for large scale data analysis. Given the dynamic nature of the Grid environment, it is more suitable for users to define workflow applications in abstract models. A full or partial concrete model can be generated just before or during workflow execution according to the current status of resources. Additionally, in some systems [14], every task in a workflow is concretized only at the time of task execution. However, concrete models may be used by some end users who want to control the execution sequence [15].

2.1.3 Workflow Composition System

Workflow composition systems are designed for enabling users to assemble components into workflows. They need to provide a high level view for the construction of Grid workflow applications and hide the complexity of underlying Grid systems. Workflow composition may be either User-directed or automatic systems. *User-directed* composition systems allow users to edit workflows directly, whereas *automatic* composition systems generate workflows for users automatically. In general, users can use workflow languages for *language-based modeling* and the tools for *graph-based modeling* to compose workflows. Within language-based modeling, users may express workflow using a *markup* language such as Extensible Markup Language (XML). Graph-based modeling allows graphical definition of an arbitrary workflow through a few basic graph elements. It allows users to work with a graphical representation of the workflow. Users can compose and review a workflow by just clicking and dropping the components of interest. It avoids low-level details and hence enables users to focus on higher levels of abstraction at application level [16]. The major modeling approaches are *Petri Nets*, *UML* (Unified Modeling Language).

2.1.4 Workflow QoS Constraints

In a Grid environment, there are a large number of similar or equivalent resources provided by different parties. Grid users can select suitable resources and use them for their workflow applications. These resources may provide the same functionality, but optimize different QoS measures. In addition, different users or applications may have different expectations and requirements. Therefore, it is not sufficient for a workflow management system to only consider functional characteristics of the workflow. QoS requirements such as time limit (deadline) and expenditure limit (budget) for workflow execution also need to be managed by workflow management systems. Users must be able to specify their QoS expectations of the workflow at the design level. Then, the actions conducted by workflow systems using run-time must be chosen according to the initial QoS requirements.



Workflow QoS constraints includes five dimensions: *time*, *cost*, *fidelity*, *reliability* and *security*. Time is a basic measure of performance. For workflow systems, it refers to the total time required for completing the execution of a workflow. Cost represents the cost associated with the execution of workflows including the cost for managing workflow systems and usage charge of Grid resources for processing workflow tasks. Fidelity refers to the measurement related to the quality of the output of workflow execution. Reliability is related to the number of failures for execution of workflows. Security refers to confidentiality of the execution of workflow tasks and trustworthiness of resources.

2.2 Information Retrieval

A Grid workflow management system does not execute the tasks itself, but it merely coordinates the execution of the tasks by the Grid resources. To map tasks onto suitable resources, information about the resources has to be retrieved from appropriate sources [10]. There are three dimensions of information retrieval: static information, historical information and dynamic information. Static information refers to information that does not vary with time. It may include infrastructure-related (e.g. the number of processors), configuration-related (e.g. operating system, libraries), QoS-related (e.g. flat usage charge), access-related (e.g. service operations), and user-related information (e.g. authentication ID). Generally, static information is utilized by Grid workflow management systems to preselect resources during the initiation of the workflow execution.

As Grid resources are not dedicated to the owners of the workflow management systems, the Grid workflow management system also needs to identify dynamic information such as resource accessibility, system workload, and network performance during execution time. Unlike static information, dynamic information reflects the status of the Grid resources, such as load average of a cluster, available disk space, CPU usage, and active processes. It also includes task execution information and market related information such as resource price.

Historical information is obtained from previous events that have occurred such as performance history and execution history of Grid resources and application components. Generally, workflow management systems can analyze historical information to predict the future behaviours of resources and application components on a given set of resources. Historical information can also be used to improve the reliability of future workflow execution.

2.3 Workflow Scheduling

Casavant et al. [19] categorized task scheduling in distributed computing systems into 'local' task scheduling and 'global' task scheduling. Local scheduling involves handling the assignment of tasks to time-slices of a single resource whereas global scheduling involves deciding where to execute a task. According to this definition, workflow scheduling is a kind of global task scheduling as it focuses on mapping and managing the execution of inter-dependent tasks on shared resources that are not directly under its control.

The workflow scheduler needs to coordinate with diverse local management systems as Grid resources are heterogeneous in terms of local configuration and policies. Taking into account users' QoS constraints is also important in the scheduling process so as to satisfy user requirements. In this section, we discuss workflow scheduling taxonomy from the view of (a) scheduling architecture, (b) decision making, (c) planning scheme, (d) scheduling strategy, and (e) performance estimation as shown in Fig 3.

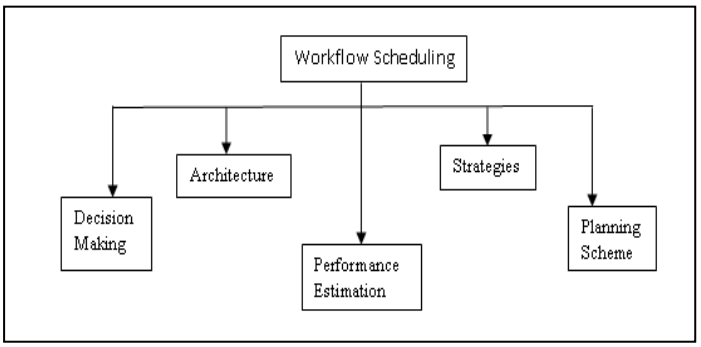


Fig 3: Key Factors of Workflow Scheduling

2.3.1 Scheduling Architecture

The architecture of the scheduling infrastructure is very important for scalability, autonomy, quality and performance of the system [17]. Three major categories of workflow scheduling architecture are centralized, hierarchical and decentralized scheduling schemes. In a *centralized* workflow enactment environment, one central workflow scheduler makes scheduling decisions for all tasks in the workflow. The scheduler has the information about the entire workflow and collects information of all available processing resources. It is believed that the centralized scheme can produce efficient schedules because it has all necessary information [13].

Unlike centralized scheduling, both *hierarchical* and *decentralized* scheduling allow tasks to be scheduled by multiple schedulers. Therefore, one scheduler only maintains the information related to a sub-workflow. Thus, compared to *centralized* scheduling, they are more scalable since they limit the number of tasks managed by one scheduler. However, the best decision made for a partial workflow may lead to suboptimal performance for the overall workflow execution. Moreover, conflict problems are more severe Scheduling Architecture. For *hierarchical* scheduling, there is a central manager and

multiple lower-level sub-workflow schedulers. This central manager is responsible for controlling the workflow execution and assigning the subworkflows to the low-level schedulers.

2.3.2 Decision Making

There is no single best solution for mapping workflows onto resources for all workflow applications, since the applications can have very different characteristics. It depends to some degree on the application models to be scheduled. In general, decisions about mapping tasks in a workflow onto resources can be based on the information of the current task or of the entire workflow and can be of two types, namely local decision and global decision [18]. Scheduling decisions made with reference to just the task or sub-workflow at hand are called local decisions whereas scheduling decisions made with reference to the whole workflow are called global decisions.

2.3.3 Planning Scheme

A planning scheme is a method for translating abstract workflows to concrete workflows. Schemes for the schedule planning of workflow applications can be categorized into either static scheme or dynamic scheme. In a static scheme, concrete models have to be generated before the execution according to current information about the execution environment and the dynamically changing state of the resources is not taken into account. In contrast, a dynamic scheme uses both dynamic information and static information about resources to make scheduling decisions at run-time.

Static schemes, also known as *full-ahead* planning, include *user-directed* and *simulation-based* scheduling. In user-directed scheduling, users emulate the scheduling process and make resource mapping decisions according to their knowledge, preference and/or performance criteria. In simulation-based scheduling, the 'best' schedule is achieved by simulating task execution on a given set of resources before a workflow starts execution. The simulation can be processed based on static information or the result of performance estimation.

Dynamic schemes include *prediction-based* and *just in-time* scheduling. Prediction-based dynamic scheduling uses dynamic information in conjunction with some results based on prediction. It is similar to simulation-based static scheduling, in which the scheduler is required to predict the performance of task execution on resources and generate a near optimal schedule for the task before it starts execution. However, it changes the initial schedule dynamically during the execution. Rather than making a schedule ahead, just in-time scheduling [19] only makes scheduling decision at the time of task execution. Planning ahead in Grid environments may produce a poor schedule, since it is a dynamic environment where utilization and availability of resources varies over time and a better resource can join at any time.

2.3.4 Scheduling Strategy

In general, scheduling workflow applications in a distributed system is an NP-complete problem [10]. Therefore, many heuristics have been developed to obtain near-optimal solutions to match users' QoS constraints. We categorize strategies of major scheduling approaches into performance-driven, market-driven and trust-driven. Performance-driven strategies try to find a mapping of workflow tasks onto resources that achieves optimal execution performance such as minimize overall execution time. Market-driven strategies employ market models to manage resource allocation for processing workflow tasks. They apply computational economy principle and establish an open electronic marketplace between workflow management systems and participating resource providers.

Workflow schedulers act as consumers buying services from the resource providers and pay some notion of electronic currency for executing tasks in the workflow. The tasks in the workflow are dynamically scheduled at run-time depending on resource cost, quality and availability, to achieve the desired level of quality for deadline and budget. Unlike the performance-driven strategy, market-driven schedulers may choose a resource with later deadline if its usage price is cheaper. Trust-driven schedulers select resources based on their trust levels. By using trust-driven approaches, workflow management systems can reduce the chance of selecting malicious hosts, and non-reputable resources. Therefore, overall accuracy and reliability of workflow execution will be increased.

2.3.5 Performance Estimation

In order to produce a good schedule, estimating the performance of tasks on resources is crucial, especially for constructing a preliminary workflow schedule. By using performance estimation techniques, it is possible for workflow schedulers to predict how tasks in a workflow or sub-workflow will behave on distributed heterogeneous resources and thus make decisions on how and where to run them. There are several performance estimation approaches: simulation, analytical modeling, historical data, on-line learning, and hybrid. simulation approaches [14] provide resource simulation environments to emulate the execution of tasks in the workflow prior to its actual execution. In analytical modeling a scheduler predicts the performance of tasks in workflow on a given set of resources based on an analytic metric.

The historical data approach relies on historical data to predict the task's execution performance. The historical data related to a particular user's application performance or experience can also be used in predicting the share of available of resources for that user while making scheduling decisions based on QoS constraints. The on-

line learning approach predicts task execution performance from on-line experience without prior knowledge of the environment's dynamics. In certain conditions, these approaches could be used together in a hybrid approach for generating performance evaluation of workflow tasks.

2.4 Fault Tolerance

In a Grid environment, workflow execution failure can occur for various reasons: the variation in the execution environment configuration, non-availability of required services or software components, overloaded resource conditions, system running out of memory, and faults in computational and network fabric components. Grid workflow management systems should be able to identify and handle failures and support reliable execution in the presence of concurrency and failures. Workflow failure handling techniques can be divided into two different levels, namely task-level and workflow-level.

Task-level techniques mask the effects of the execution failure of tasks in the workflow, while workflow-level techniques manipulate the workflow structure such as execution flow to deal with erroneous conditions. Task-level techniques have been widely studied in parallel and distributed systems. They can be catalogued into retry, alternate resource, checkpoint/restart and replication. The retry technique is the simplest failure recovery technique, as it simply tries to execute the same task on the same resource after failure. The alternate resource technique [18] submits failed task to another resource.

The checkpoint/restart technique moves failed tasks transparently to other resources, so that the task can continue its execution from the point of failure. The

replication technique runs the same task simultaneously on different Grid resources to ensure task execution provided that at least one of the replicas does not fail.

Workflow-level techniques include alternate task, redundancy, user-defined exception handling and rescue workflow. The alternate task technique executes another implementation of a certain task if the previous one failed, while the redundancy technique executes multiple alternative tasks simultaneously. The user-defined exception handling allows the users to specify a special treatment for a certain failure of a task in workflow. The rescue workflow technique ignores the failed tasks and continues to execute the remainder of the workflow until no more forward progress can be made.

2.5 Data Transfer

For Grid workflow applications, the input files of tasks need to be staged to a remote site before processing the task. Similarly, output files may be required by their children tasks which are processed on other resources. Therefore, the intermediate data has to be staged out to the corresponding Grid sites. Some systems require users to manage intermediate data transfer in the workflow specification, rather than providing automatic mechanisms to transfer intermediate data. We categorize approaches of automatic intermediate data movement into centralized, mediated and peer-to-peer.

Basically a centralized approach transfers intermediate data between resources via a central point. Centralized approaches are easy to implement and suit workflow applications in which large-scale data flow is not required. In a mediated approach, rather than using a central point, the locations of the intermediate data are managed by a distributed data management system [17]. Mediated approaches are more scalable and suitable for applications which need to keep intermediate data for later use.

A peer-to-peer approach transfers data between processing resources. Since data is transmitted from the source resource to the destination resource directly without involving any third-party service, peer-to-peer approaches save the transmission time and reduce the bottleneck problem caused by the centralized and mediated approaches.

Thus, they are suitable for large-scale intermediate data transfer. However, there are more difficulties in deployment because they require every Grid node to be capable of providing both data management and movement service. In contrast, centralized and meditated approaches are more suitable to be used in applications such as bio-applications, in which users need to monitor and browse intermediate results. In addition, they also need to record them for future verification purposes.

III. CONCLUSION

We have presented a taxonomy for Grid workflow management systems name Grid Venture Taxanomy. The taxonomy focuses on workflow design, workflow scheduling, resource management, fault management and data transfer. At the execution level, the workflow scheduling must be able to map the workflow onto Grid resources to meet users' QoS requirements. Therefore, the role of market-driven strategies will become increasingly important, currently being ignored in most Grid workflow management systems. Trust-based scheduling is another approach to improve QoS in open distributed systems such as Grid and peer-to-peer. It is impossible to make an optimal scheduler without knowledge of estimated time of task execution. Given the dynamic nature of Grid environments, fault tolerance should be fully supported by Grid workflow management systems. However, most fault handling techniques have not been developed or implemented in many Grid workflow systems, especially at the workflow execution level. It is hard for a workflow management system to survive in real Grid environments without robust fault handling techniques.

REFERENCES

[1] W.M.P. van der Aalst and K.M. van Hee, Workflow Management: models, methods, and Systems. MIT Press, Cambridge, Mass., USA, 2002.
[2] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski and A.P. Barros, Workflow Patterns. URL http://tmitwww.tm.tue.nl/research/patterns/ [December 2004].
[3] J. H. Abawajy. Fault-Tolerant Scheduling Policy for Grid Computing Systems. In 18th International Parallel and Distributed Processing Symposium (IPDPS'04), Santa Fe, New Mexico, IEEE Computer Society (CS) Press, Los Alamitos, CA, USA, April 26-30, 2004; 238-244.
[4] D. Abramson, J. Giddy, and L. Kotler. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? In 14th International Parallel and Distributed Processing Symposium (IPDPS 2000), Cancun, Mexico, IEEE CS Press, Los Alamitos, CA, USA, May 1-5, 2000.
[5] M. Addis, J. Ferris, M. Greenwood, P. Li, D. Marvin, T. Oinn, and A, Wipat. Experiences with e-Science Workflow Specification and Enactment in Bioinformatics, In UK e-Science All Hands Meeting 2003, IOP Publishing Ltd, Bristol, UK, 2003; 459-467.
[6] G. Allen, K. Davis, K. N. Dolkas, N. D. Doulamis, T. Goodale, T. Kielmann, A. Merzky, J. Nabrzyski, J. Pukacki, T. Radke, M. Russell, E. Seidel, J. Shalf, and I. Taylor. Enabling Applications on the Grid – A GridLab Overview. International Journal of High Performance Computing Applications (JHPCA), Special Issue on Grid Computing: Infrastructure and Applications, SAGE Publications Inc., London, UK, August 2003.
[7] J. Almond and D. Snelling. Unicore: Secure and Uniform Access to Distributed Resources via the World Wideeb. White Paper, October 1998, http://www.fz-juelich.de/zam/RD/coop/unicore/whitepaper.ps
[December 2004].
[8] I. Altintas, A. Birnbaum, K. Baldridge, W. Sudholt, M. Miller, C. Amoreira, Y. Potier, and B. Ludaescher. A Framework for the Design and Reuse of Grid Workflows, International Workshop on Scientific Applications on Grid Computing (SAG'04), LNCS 3458, Springer, 2005.
[9] K. Amin and G. von Laszewski. GridAnt: A Grid Workflow System. Manual, February 2003, http://wwwunix. globus.org/cog/projects/gridant/gridant-manual.pdf [December 2004].
[10] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, S. Weerawarana. Business Process Execution Language for Web Services Version 1.1, 05 May 2003, http://www-128.ibm.com/developerworks/library/ws-bpel/ [Feb 2005]
[12] D. A. Bacigalupo, S. A. Jarvis, L. He, and G. R. Nudd. An Investigation into the application of different performance techniques to e-Commerce applications. In Workshop on Performance Modelling, Evaluation and Optimization of Parallel and Distributed Systems, 18th IEEE International Parallel and Distributed Processing Symposium (IPDPS), Santa Fe, New Mexico, IEEE CS Press, Los Alamitos, CA, USA, April 26-30, 2004.
[13] R. Bastos, D. Dubugras, and A. Ruiz. Extending UML Activity Diagram for Workflow Modeling in Production Systems. In 35th Annual Hawaii International Conference on System Sciences (HICSS'02), Big Island, Hawaii, IEEE CS Press, Los Alamitos, CA, USA, January 07 -10, 2002.
[14] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski. The GrADS Project: Software Support for High-Level Grid Application Development. International Journal of High Performance Computing Applications(JHPCA), 15(4):327-344, SAGE Publications Inc., London, UK, Winter 2001.
[15] I. Brandic, S. Benkner, G. Engelbrecht, and R. Schmidt, Towards Quality of Service Support for Grid Workflows, First European Grid Conference (EGC 2005), Amsterdam, The Netherlands, Feb 2005.
[16] T.D.Braun, H. J. Siegel, N. Beck, L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, and B. Yao. A Taxonomy for Describing Matching and Scheduling Heuristics for Mixed-Machine Heterogeneous Computing Systems. In 17th Symposium on Reliable Distributed Systems. West Lafayette, IN. IEEE CS Press, Los Alamitos, CA, October 1998: 330-335.
[17] R. Buyya, D, Abramson, and J. Giddy. Nimrod/G: An Architecture of a Resource Management and Scheduling System in a Global Computational Grid, HPC Asia 2000, Beijing, China, IEEE CS Press, Los Alamitos, CA, USA, May 14-17, 2000; 283-289.
[18] R. Buyya, D. Abramson, and J. Giddy. A Case for Economy Grid Architecture for Service-Oriented Grid Computing. In 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), San Francisco, California, USA , IEEE CS Press, Los Alamitos, CA, USA, April 2001.
[19] R. Buyya and S. Venugopal. The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report. In 1st IEEE International Workshop on Grid Economics and Business Models, GECON 2004, Seoul, Korea, IEEE CS Press, Los Alamitos, CA, USA, April 23, 2004; 19-36.
[20] J. Cao, S. A. Jarvis, S. Saini, D. J. Kerbyson and G. R. Nudd. ARMS: An Agent-based Resource Management System for Grid Computing. Scientific Programming. Special Issue on Grid Computing, 10(2):135-148, IOS Press, Amsterdam, Netherlands, 2002.