Meta Scheduling Framework for Workflow Service on the Grids

Seogchan Hwang[1], Jaeyoung Choi[1], and Hyeongwoo Park[2]

[1] School of Computing, Soongsil University,
1-1 Sangdo-5dong, Dongjak-gu, Seoul 156-743, Korea
seogchan@ss.ssu.ac.kr, choi@comp.ssu.ac.kr

[2] Supercomputing Center, Korea Institute of Science and Technology Information,
52 Eoeun-dong, Yusung-gu, Daejun 305-333, Korea
hwpark@supercomputing.re.kr

Abstract. The Globus becomes a standard to construct a Grid and provides core services such as resource management, security, data transfer, information services, and so on. Workflow management becomes a main service as one of the important grid services for grid applications. We propose a Meta Scheduling Framework (MSF) in this paper. The MSF provides an XML-based Job Control Markup Language for describing information and procedures of applications, and a workflow management service for scheduling job flow.

1 Introduction

Grid computing [1] is a new infrastructure to provide computing environments for grand challenge problems by sharing large-scale resources. The Globus toolkit is a standard to construct a Grid and provides essential grid services such as security, resource management, data transfer, information service, and so on. However, it still needs more works and researches to satisfy the requirements of various grid applications. Workflow management is emerging as one of the most important grid services. It is difficult to use the grid resources for general applications because the grid resources have various characteristics such as heterogeneity and dynamic organization. So many research groups have been working on a workflow related projects.

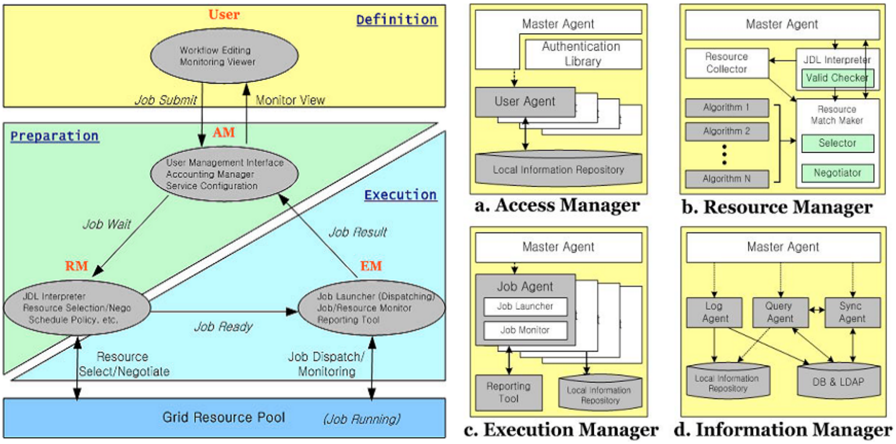
GridFlow [2] is a workflow management system using agent-based resource management and local resource scheduling system, Titan. It focuses on the scheduling of time-critical grid applications in a cross-domain and highly dynamic grid environment using a fuzzy timing technique and performance prediction of application. MyGrid [3] provides a service for integration such as resource discovery, workflow enactment and distributed query processing. It is a research project middleware to support biology environments on a Grid. And Condor [4] provides a workload management system of compute intensive jobs, and a scheduling of dependencies between jobs using DAGMan. This project provides similar functionalities but require their own specific infrastructures.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3036, pp. 445–448, 2004.

In this paper, we introduce a system, called Meta Scheduling Framework (MSF), for grid computational environments. MSF is developing high-level middleware to support process with complexity for general applications. MSF provides a Job Control Markup Language that is able to specify the job flow for general applications which are not developed for grid environments. It also provides a workflow management service, based on Globus, to execute the job, and a graphical user interface to facilitate the composition of grid workflow elements and the access to additional grid resources.

2 Meta Scheduling Framework

The goal of this research is to develop a framework to provide a workflow service to applications using the Globus. To accomplish this, we designed a workflow description language, called Job Control Markup Language (JCML), and a workflow management system. The JCML is designed to describe a process of tasks. And the workflow management system provides services to control the flows of tasks.



a. Access Manager b. Resource Manager c. Execution Manager d. Information Manager

Fig. 1. Meta Scheduling Framework Architecture

Figure 1 shows the architecture of MSF and its major components. A user describes a job flow using the MSF console. The Access Manager (AM) provides services which include user authentication, environment setup, and job submit service. The Resource Manager (RM) provides resource discovery and matching. The Execution Manager (EM) provides job launching, monitoring, and reporting.

MSF consists of three phases; definition, preparation, and execution. During the definition phase jobs are defined to specify a Job Definition List (JDL), which describes a task flow using JCML. In this phase, users connect to the AM for authentication of the MSF and then the AM creates a user proxy for Globus. In the preparation phase, resources are searched and assigned to the matched tasks. The AM creates an agent to provide a proxy service to the user. The agent passes the JDL and traces the status of the job. The RM receives the JDL from the AM and analyzes it. After finding appropriate grid resources for the job, the RM assigns them to tasks and generates a worklist that includes information on activities and their executing order. Finally, during the execution phase the tasks on the worklist are executed, the status is monitored, and the results are reported.

A job description language to specify the flow of an application task has to provide a way to describe various grid environments and task information including execution environments, such as arguments, sequence, data dependency, prefetching, and so on. The JCML is a workflow description language based on the Graph eXchange Language (GXL) [5], which defines a graph-based XML representation for specifying the dependencies among components. The JCML consists of four major elements; Info, Resource, Component, and Dependency.

Info: This element lists the document name, scope, target namespaces, authoring date, and so on.

Resource: This element describes the resources of hardware and software required to execute a job. The hardware includes architecture, CPU, memory, disk, and network bandwidth. The software includes an operating system, installed application, and local scheduler. The Time represents the deadline to be executed.

Component: This element lists all of the task-related information. A Node is an executing program or computer in the workflow. A node is classified into a task and a resource. A task node is an executing program in workflow. And a resource node is an assistant which represents the physical computing resources, such as storage and database, to support the task node. A task node includes execution file, input, output, arguments, and resource configuration, and a resource node includes data location and the access method. A group is a node logically, if it is necessary to refer a series of nodes to a single node according to a job flow logic. And one group can include another group(s).

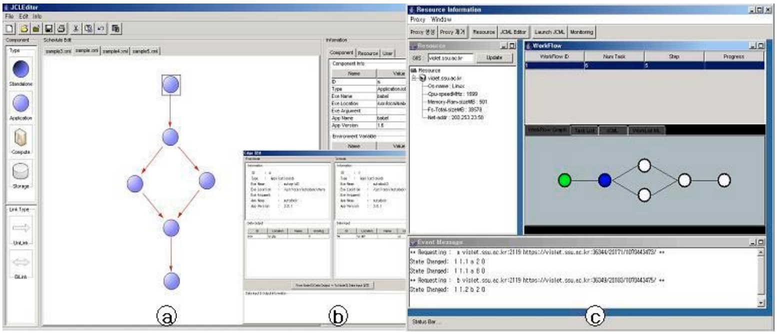
Dependency: This element describes the dependencies of a workflow. Each line is an edge which represents an execution order and a dependency between two objects (node or group). It has two types of links, PriorityOrder and Datalink, which have a direction that expresses a starting point and an end point of linked nodes. The PriorityOrder just represents an execution order between the two linked nodes. The Datalink displays a flow of data which is used for the input or output file of each task.

A workflow management system guarantees that a flow of task will be executed exactly. A job is processed by a workflow management system as follows: interpreting the job, mapping resources, generating a worklist, and scheduling tasks, as shown in Figure 1. A user specifies a JDL to execute a grid application. After interpreting the JDL, the JDL Interpreter searches resources and assigns resources to the job, selected by the Resource Match Maker, which uses an external information service such as NWS [6], and then generates the worklist. The worklist has the execution information of the task and its execution order. Each task consists of activities.

The Job Launcher executes activities according to the worklist. At the workflow system, it is required to transform activities in the worklist to RSL type in order to execute the real task in a local grid scheduler.

3 Implementation and Conclusion

We developed a prototype of MSF using pure Java (JDK 1.4.0) and Java CoG Kit (version 0.9.13) on Globus 2. We also implemented and executed a Virtual Screening on MSF. A Virtual Screening is one of the design methods, called *docking* in Bioinformatics. In order to execute docking, the format of material must be changed. We chose the AutoDock application for this experiment. Figure 2 displays snapshots of the progress windows for AutoDock processing



Resource
violet.ssu.ac.kr
Os-name : Linux
Cpu-speedMHz : 1699
Memory-Ram-sizeMB : 501
Fs-Total-sizeMB : 39578
Net-addr : 203.253.23.50

WorkFlow ID	Num Task	Step	Progress
1	6	5	

```
++ Requesting :  a violet.ssu.ac.kr:2119 https://violet.ssu.ac.kr:36344/20171/1070443473/ ++
State Changed:  1 1.1 a 2 0
State Changed:  1 1.1 a 8 0
++ Requesting :  b violet.ssu.ac.kr:2119 https://violet.ssu.ac.kr:36349/20180/1070443475/ ++
State Changed:  1 1.2 b 2 0
```

Fig. 2. (a) JCML Main Editing Window, (b) Edge Window, (c) Monitoring Console

MSF provides a workflow service for general applications in a Grid environment. So we designed and implemented a workflow description language, JCML, to describe the flow of application including complexity and dependencies, and a workflow management system to execute and monitor the flow. Currently we are working to extend the architecture in order to enhance efficiency and availability and to describe jobs in more detail with JCML. And MSF will support the Globus 3.0, which integrates scientific and enterprise environments based on web service.

References

1. I. Foster and C. Kesselman, ed., The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, (1998).
2. J. Cao, S. A. Jarvis, S. Saini and G. R. Nudd, GridFlow: Workflow Management for Grid Computing, 3rd International Symposium on Cluster Computing and the Grid, (2003), 12-15.
3. R. Stevens, A. robinson and C. Goble, myGrid: personalized bioinformatics on the information grid, Bioinformatics, 19(1), (2003), 302-304.
4. M. Litzkow, M. Livny and M. Mutka, Condor - A Hunter of Idle Workstations, 8th International Conference of Distributed Computing Systems, (1998), 13-17.
5. A. Winter, B. Kullbach and V. Riediger, An Overview of the GXL Graph Exchange Language, Software Visualization. LNCS, Vol. 2269, (2002), 324-336.
6. R. Wolski, N. Spring and J. Hayes, The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, Future Generation Computing Systems, 15(5/6), (1999), 757-768.