Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp533-539)

Grid Workflow for Decision Resource Scheduling

Mingsheng Hu[1 2],Jianjun Zhang[3],Xueguang Chen[1]
(Institute of Systems Engineering, Huazhong University of Science and Technology, 430074, P.R.China)1
(Institute of Software Science, Zhengzhou Teachers College, 450044, P.R.China) 2
(Department of computer science,Xuchang College, 461000, P.R.China)3

Abstract: With the advent of Grid and application technologies, scientists and engineers are building more and more complex decision support applications to process large data sets, and scheduling distributed decision resources. Such decision resource scheduling processes require means for composing and executing complex workflows. Therefore, many efforts have been made towards the development of workflow management systems for decision resource scheduling on Grid environment. In this paper, we investigates the emerging need for Grid workflow to aid decision support on the Grid and proposes a Grid workflow for decision resource scheduling that is a Petri net-based graph model and incorporates a reasoning system designed to perform such a task.

Key-Words: Decision Resource Scheduling; Workflow; Grid

1 Introduction

Workflow is concerned with the automation of procedures whereby files and data are passed between participants according to a defined set of rules to achieve an overall goal. A workflow management system defines, manages and executes workflows on computing resources. Imposing the workflow paradigm for application composition on Grids offers several advantages such as:

· Ability to build dynamic applications which orchestrate distributed resources.

·Utilization of resources those are located in a particular domain to increase throughput or reduce execution costs.

· Execution spanning multiple administrative domains to obtain specific processing capabilities.

· Integration of multiple teams involved in managing of different parts of the experiment workflow – thus promoting inter-organizational collaborations.

The Web-based Decision Support Systems (DSS) have made information sharing on the Internet possible, but they cannot meet the decision-maker's needs in the heterogeneous, autonomic, dynamic and distributed decision support environment, because they only link web pages and lack global mechanism to manage and coordinate decision support resources on the Internet. Grids have emerged as a global cyber-infrastructure for the next-generation of DSS applications by integrating large-scale, distributed and heterogeneous resources. It will improve DSS greatly, and bring profound revolution to DSS theory and its application [1].

In order to support complex resource scheduling, distributed decision resources such as data, applications, and knowledge-based database need to be orchestrated while managing the application workflow operations within Grid environments [2].

The existence of description languages, semantics, service publishing methods and Grid middleware have up to now enabled interoperability between decision resources available on the Grid environment. However, there is an emerging need for the ability to dynamically discover how available services, resources and data could be utilized not only to process a task, but to achieve the desired outcome in the most suitable manner. In this paper, we investigates the emerging need for Grid workflow to aid decision support on the Grid and proposes a Grid workflow for decision resource scheduling that is a Petri net-based graph model and incorporates a reasoning system designed to perform such a task.

2 Grid Workflow Requirements

Just like the Web service technologies aim to do, the Grid workflow specification should allow specific activities implemented by individual services to be exported as activities of the workflow. It should also allow the exported activities to trigger a chain of other activities. Current technologies such as WSFL address this issue effectively. Hence, we try to incorporate these features presented by WSFL into the Grid

Services Flow Language. Furthermore, the activities exported in such a manner should also be described in the same manner as the service itself. In this sense, the specification should be rich enough to describe the workflow such that the WSDL for the workflow entity (henceforth referred to as a workflow coordinator) can be auto-generated from the specification. The workflow coordinator must be able to handle the methods that have been dynamically exported as a composition of the various activities of the workflow, in such a way that clients can access them using the same standard tools that they use to deal with the individual services. This is an important requirement for recursive composition of services.

Web services define their workflow in such a way that the workflow engine has to intermediate at each step of the application sequence, as shown in Figure 1. This is because most of the current sets of workflow technologies are designed for business to business communication, where there may only be a moderate level of data transmission across the Web services. As a result, the workflow engine does not end up being a real bottleneck. However, for Grid based services, exchanging large amounts of data is the norm. Having a central workflow engine relay the data between the services would be a bad idea in this case. The workflow specification needs to be able to allow communication between the services as depicted in Figure 2. OGSA adds extensions to WSDL in order to address Grid-specific needs. It addresses communication between Grid services using *notificationSources* and *notificationSinks*, which allow services to carry out asynchronous delivery of messages between each other.

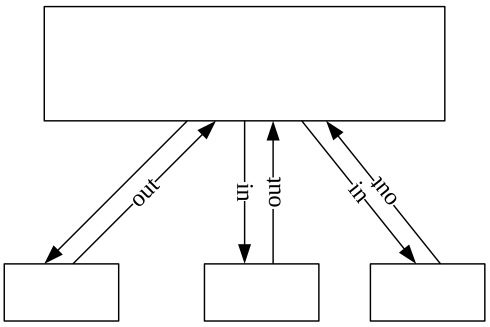

Fig. 1. Web Services Workflow Model

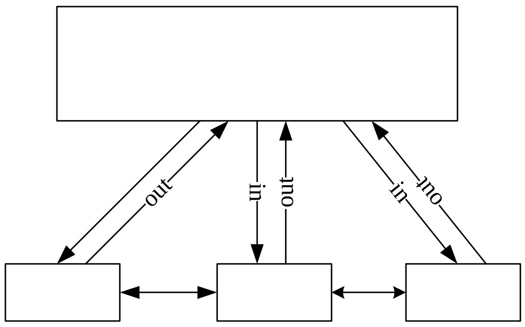

Fig. 2. Grid Services Workflow Model

3 decision resource scheduling process Based on Grid workflow

The Grid workflow described in this paper incorporates the interoperability of Grid web services. An example is given of a decision support for multiple travel-related businesses. The initiating business is the travel agency company. The Travel Agency has internal services for managing customers’ accounts and credit card numbers. However, the travel agency uses other third-party vendors to realize the hotel reservation and car rental reservations. The Hotel Reservation and Car Rental companies register their offerings as web services in a distributed registry, such as a UDDI registry. The Travel Agency uses these registry services as a part of its internal workflow. In addition, the Travel Agency has a partnership with an on-line publishing company that publishes the finalized itineraries. In this case, the travel agency has a static connection with the partner organization and is able to access services directly over a shared network connection. Problems occur in this domain when the online companies update or remove their service offerings. This requires a Grid workflow manager system that supports a methodology for process or workflow oriented service specification. This way would allow workflow developers to specify the process sequence and message exchange between local and distributed services. The specification approach must support both functional and nonfunctional concerns. Therefore when the process or services change, the specification can be updated and the supporting architecture automatically reconfigured [3].

Decision resource scheduling process Based on Grid workflow, in a general sense, consists of five steps as illustrated in Figure 3.

(1) Decision resource Discovery: Identify characteristics, configuration, capability, and

suitability of Decision resource. This discovery can occur on services in UDDI registry or locally registered component-based services.

(2) Decision resource Capturing: Save the Decision resource characteristics in the service-oriented data model and then interact with Decision resource to establish their configuration and access cost. It creates a broker Decision resource list and the Decision resource performance data as predicted through the measurement and extrapolation methodology.

(3) Scheduling Process: The scheduling flow manager selects an appropriate scheduling algorithm for component decision resource depending on the user's requirements (deadline and budget limits, and optimization strategy—cost, cost–time, time, and conservative-time).

(4) Process Capturing: When the resource Scheduling Process has finished, the Decision resource returns it to the broker's receptor agent. It aids in predicting the job consumption rate for making scheduling decisions.

(5) Agent Self-Configuration and Deployment: application layer agents access the integrated data model and configure themselves for workflow enactment in the Grid environment. At the end, the agent returns updated Decision resource data back to the user entity.

In this process, scheduling decisions are made dynamically at runtime and are driven and directed by the end-users requirements. While a conventional cost model often deals with software and hardware costs for running applications, an economy model primarily charges the end user for services that they consume based on the value they derive from it. Pricing policies are based on the demand from the users and the supply of Decision resource is the main driver in the competitive, economic market model. Therefore, a user competes with other users and a resource owner with other resource owners.

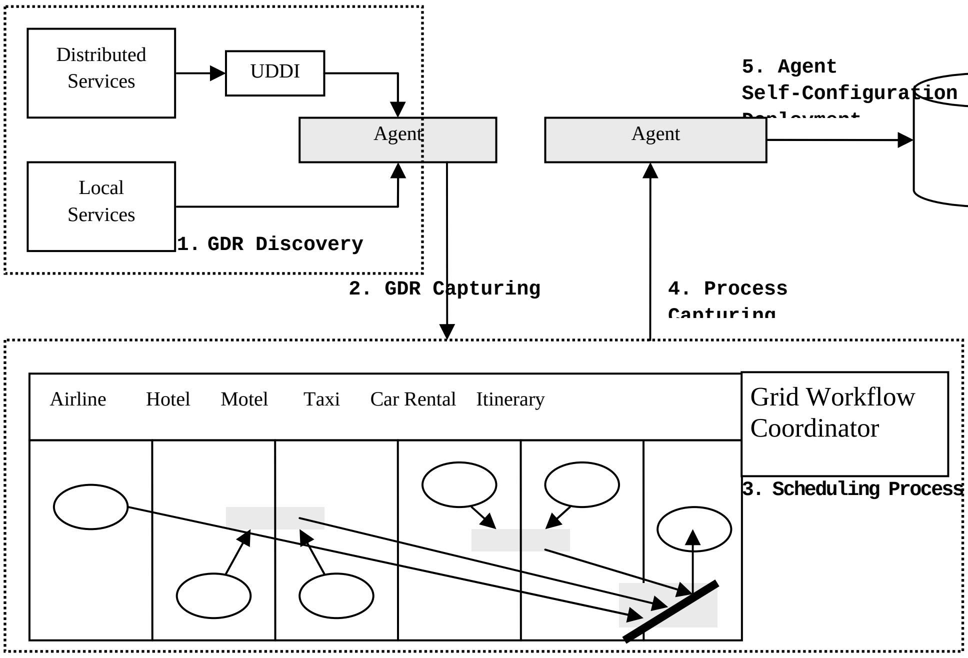

Fig. 3. Decision resource scheduling process based on Grid workflow

4 Grid workflow orchestrations

In order to enable the user to compose complex decision support applications on distributed heterogeneous and unreliable decision resources within Grid environments, the concept of Grid workflows has emerged. It describes patterns of control and dataflow between resource, including human actors participating in interactions. Several techniques have been established in the Grid community in order to define the workflow of Grid jobs. A very promising approach for this purpose is the use of graphs. Graphs offer a very intuitive way of modeling abstract workflows and can be handled easily even by non-expert users. The main limitation of graphs is the fact that they may become very huge if the workflow is too complex.

The OGSA working group defines a workflow as a pattern of process interaction, not necessarily corresponding to a fixed set of processes. All such interactions may be between services residing within a single data center or across a range of different platforms and implementations anywhere. The orchestration of workflows describes the ways in which these processes are constructed from Web Services and other processes, and how these processes interact. Two main preconditions must be fulfilled in order to enable the orchestration of workflows: an adequate description of the components for deciding which components are functional to solve the problem, and a suitable workflow model for defining how the components should interact within that workflow [4].

The component model of the system supports legacy command line programs as well as OGSI Grid Services in a mixed way. The command line programs may be wrapped by simple shell scripts to fulfill the component specification, which requires a specific format of the command line parameters in order to enable asynchronous communication between the programs using file IO. The Grid Services are extended Web Services according to OGSI specification.

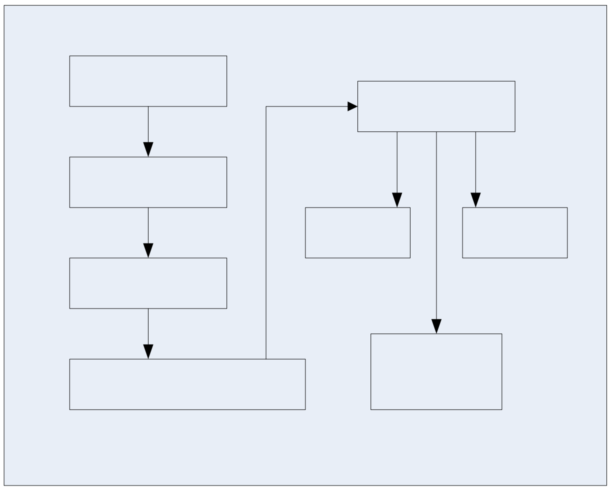

Fig. 4. Architecture of Grid workflow orchestrations

The distinct characteristic is to use the grid workflow as the foundation of the grid service composition and scheduling, which can benefit from the workflow architecture in the composition, visualization, verification, scheduling, execution and monitoring (Figure 4). So the architecture of grid workflow orchestrations shares the components of grid workflow architecture. The paper focuses on the method addressing semantic issues and the composition of grid service, and the algorithm improving scheduling efficiency when the composite grid service execution in the grid environment.

The architecture can be divided into two primary parts. The first part is the left component of Figure 4 which comprises “Semantic processor”, “Composition processor” and “Grid workflow model constructor”, and deals with the grid service composition. The second part is the right component of Figure 4 which deals with the grid service scheduling through the grid workflow. According to the architecture, it is very convenient to visualize and verify the composition and to schedule the grid services.

For specifying the interactions between the components, it is mandatory to have an adequate workflow model and a corresponding workflow description language. We detected the following main requirements within the context of a Grid computing architecture:

• The user must be able to define the Grid workflow on an abstract level without knowledge about the infrastructure.

• The Grid workflow model should be mostly universal (e.g., Turing complete) in order to cover a broad range of workflow patterns.

• At the same time, the Grid workflow model must be simple and easy to use.

• It should support the modeling of data as well as control flow.

• Due to the dynamic and unreliable nature of GDRs, dynamic workflows are required that may change their structure during runtime.

Petri nets are well established as a model for the analysis and the design of complex discrete systems. Here, we extended this model to be usable directly for the automatic execution of workflows on common Grid middleware as well. Therefore, the abstract Petri net elements are linked to component descriptions that themselves can be mapped onto real Grid resources.

The Petri net model is a powerful modeling tool. A very complex activity can often be quickly and easily translated to a Petri net representation. Petri net models have proved effective in net analysis for deadlock detection and behavior trends in asynchronous systems, thus for determining the correctness and efficiency of proposed systems. A Petri net model consists of a set of places and a set of transitions. The places and transitions are connected by a set of directed arcs. A transition is said to be enabled if there are enough tokens in each of the input places as specified by the arcs connecting the input places to the transition. An enabled transition can fire if the other conditions associated with the transition are satisfied. We model above algorithms and verify their correctness with Petri nets. In the models, the places (denoted by circles) correspond to the states and the transitions (denoted by lines) represent the actions related to the states. For the purpose of intuition, we express the coordinator states with corresponding messages. Petri nets allow the graphical definition of arbitrary workflows with only few basic graph elements – just by connecting data (files, parameters) and software components (command line programs, Grid Service method calls). Petri nets belong to a special class of directed graphs. The type of Petri nets we introduce here, are Petri nets with individual tokens (colored Petri nets) and constant arc expressions which are composed of places, denoted by circles (○), transitions, denoted by boxes (□), arcs from places to transitions (○→□), arcs from transitions to places (□→○), individual and distinguishable objects that flow through the net as tokens (●), an initial marking that defines the tokens which each place contains at the beginning, and an expression for every arc that denotes an individual object. A place p is called input place (output place) of transition t if an arc from p to t (from t to p) exists. A brief introduction to the theoretical aspects of colored Petri nets can be found. Petri nets are suitable to describe the sequential and parallel execution of tasks with or without synchronization; it is possible to define loops and the conditional execution of tasks.

A Petri net is an abstract and formal modeling tool for representation and analysis of scheduling processes. It is able to represent most coordination problems, easy to use and understand, and shares common properties, such as bounded-ness, liven-ness, deadlock-freeness, proper termination and completeness. It can model systems' events, conditions and the relationships among systems. The occurrence of these events may change the state of the system, causing some of the previous conditions to cease holding and other conditions to begin to hold. In most cases, the workflow within Grid jobs is equivalent to the dataflow, i.e., the decision when to execute a software component is taken by means of availability of the input data. Therefore, the tokens of the Petri net represent real data that is exchanged between the software components. In this case, we use Petri nets to model the interaction between software resources represented by software transitions, and data resources represented by data places. In other cases, however, the workflow should be independent from the dataflow, if you want to synchronize several activities – and in addition to the data places and software transitions behave to introduce control places and control transitions. Control transitions evaluate logical conditions. In the case that a software transition is linked to a command line program, the tokens on the output places contain the exit status of the process (e.g., done, failed), whereas in the case of Grid Services, the tokens store the output parameter returned by the method call.

5 Coordinate Scheduling Strategies based on workflow

Given the complexity of the grid workflow execution, we have designed a decentralized scheduling system which supports just in-time planning and allows the decisions of the resource allocation to be made and changed at run-time.

We believe that decentralized scheduling architecture is more efficient over the centralized scheduling for complex workflow processing, which handles all tasks by one scheduler. In our system, every task has its own scheduler called task manager (TM) which implements a scheduling algorithm and handles the processing of the task, including resource selection, resource negotiation, task dispatcher and failure processing. The lifetimes of TMs, as well as the whole workflow execution, are controlled by a workflow coordinator (WCO).Each TM has its own monitor which is responsible for monitoring the health of the task execution on the remote node. Every TM maintains a resource group which is a set of resources that provided services required for the execution of an assigned task. TMs and WCO communicate through a event service server (ESS).

In the system, the behaviors of task managers and workflow coordinator are driven by events. A task manager is not required to handle communication with others and only generates events according to task's processing status. At the same time, the task managers take actions only according to occurred events without concern for details of other task managers [5].

The event notification is based on subscription-notification model. WCO and TMs just subscribe the events of interest after activation, and then they can do whatever they want. When a subscribed event occurs, they will be informed.

The benefit of the event-driven mechanism is it provides a loosely-coupled control; hence the design and development of the system are very flexible.

Interaction sequence in Workflow Scheduling is illustrated in Figure 5. System Firstly, WCO needs to register to ESS and subscribe the events of task execution status. And then WCO activates task managers of first level tasks, in the example, only one TM1. After TM1 finishes the preprocessing of the task execution, it sends a message to ESS saying "I am executing the task". ESS informs the WCO and WCO activates TMs of children tasks of TM1, namely TM2 and TM3 in this example. The inputs of the task of TM2 and TM3 rely on the output of the task of TM1, so TM2 and TM3 register to ESS and listen to its output events. Once TM1 identifies a suitable resource, it submits task to that resource. As soon as TM1 knows the output of the task, it informs TM2 and TM3 through ESS. If all input data of the task of TM2 and TM3 are ready, TM2 and TM3 reports execution status to ESS, and then proceeds to the execution of their tasks. After WCO receives the notification of the execution of TM2 and TM3, WCO will activate their children task managers, so they can begin to listen to their inputs and prepare task execution. This process will be continued until the end of workflow execution.

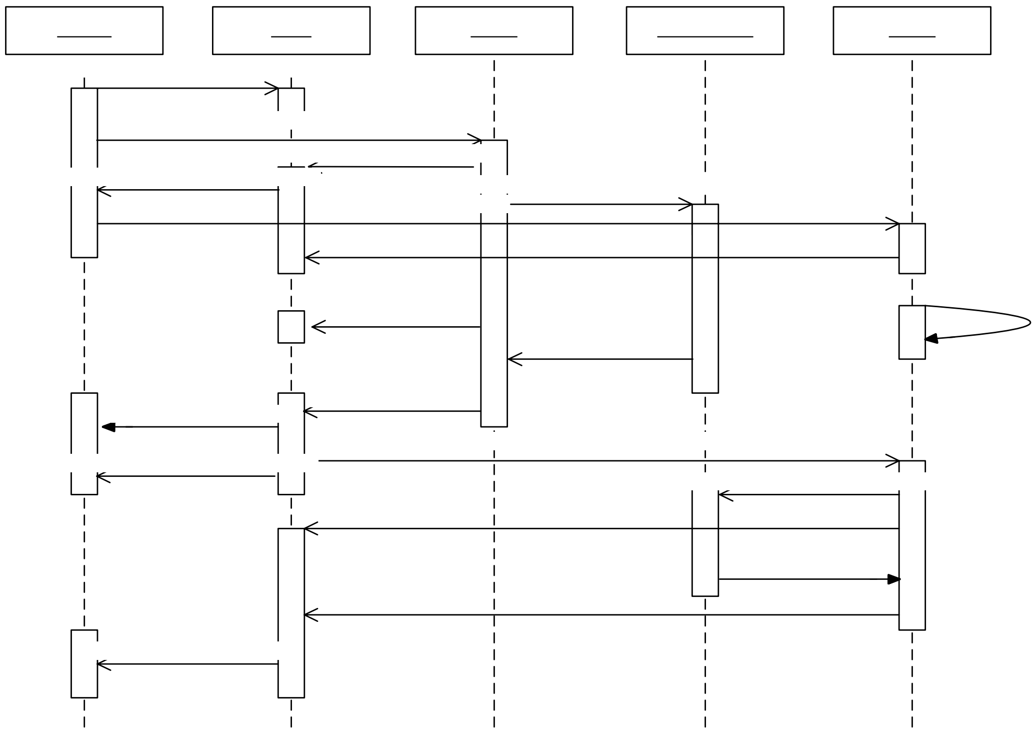
Fig. 5. Interaction sequence in Workflow Scheduling System

6 Conclusions and Future Work

There still are broad areas of application and enhancement that are possible in the Grid Workflow. This is still a work in progress, and the language will continue to evolve depending upon the requirements of the Grid community. The innovation in respect to former work in this domain is the incorporation of a Petri net-based workflow model for orchestrating Grid Service method calls as well as legacy command line applications within a single workflow. This approach allows the definition of arbitrary workflows, including conditions and loops, with only three different abstract graph elements: transitions, places, and arcs. Grid Services and legacy command line programs can be integrated easily to a single, loosely coupled Grid application, regarding the dataflow as well as the control flow.

Our future work will focus on workflow execution optimization. We will be extending resource allocation algorithms to support optimal and QoS (Quality of Service) requirements based scheduling, using computational economy. In addition, we will assist its users in composing powerful grid workflows by means of a self-adapting expert system.

References:

[1] Xueguang Chen a study on the framework of Grid based Decision Support Systems proceedings of 2003 International Conference on Management Science & Engineering Atlanta

WCO

register task status eve

act

[2] Mingsheng Hu,Xueguang Chen,Zhijuan Jia. Decision Resource Management and Scheduling on the Grid. Greece:1st WSEAS International Symposium on GRID COMPUTING, 2005:3-5

[3] Mingsheng Hu,Xueguang Chen. AGBODSS□Agent Grid-Based Open Decision Support System. WSEAS transacyions on information science and applications[J],2005,8:2-8

[4] Falk Neubauer, Andreas Hoheisel , Joachim Geiler , Workflow-based Grid applications. Future Generation Computer Systems,2006, 22:6–15

[5] Mingsheng Hu, Application of PBS Based on Grid Computing. MICROCOMPUTER & ITS APPLICATIONS, China, Vol.24 No.6, 2005, P.7-10