A Distributed Re-configurable Grid Workflow Engine

Jian Cao, Minglu Li, Wei Wei, and Shensheng Zhang

Department of Computer Science & Technology, Shanghai Jiaotong University,
200030, Shanghai, P.R. China

{cao-jian, li-ml, wei-wei, sszhang}@cs.sjtu.edu.cn

Abstract. Grid workflow, as a basic service in the grid environment, is a useful tool to help researchers make use of various grid resources to solve scientific problems. It is possible lots of users invoke grid workflow services in a very narrow time interval. Therefore, a centralized grid workflow engine is apt to be a bottleneck. In the paper, a novel grid workflow engine is proposed. It is based on Jini platform and employs Jini services to implement functions of grid workflow engine. By adding or removing enactment services, the grid workflow engine can be reconfigured dynamically. The workflow manager of the engine can allocate requests to proper enactment services according to some load-balancing strategy. The structure and mechanisms of this grid workflow engine are discussed in the paper. A prototype system and testing results are also introduced.

1 Introduction

In order to solve complex scientific problems, geographically distributed and heterogeneous resources should be connected together through the high-speed network. Grid, which provides virtual organization with capabilities to solve the problems cooperatively, is put forward to meets these more and more critical requirements [1]. As a new infrastructure, grid is extending its role as a high performance computing environment and bringing deep impacts on the human life and the society [2].

To make these complex grid applications accessible to the many potential users outside the scientific community is a great challenge. In earthquake science, for example, integrated earth sciences research for complex probabilistic seismic hazard analysis can have greater impact, especially when it can help mitigate the effects of earthquakes in populated areas. In this case, users might also include safety officials, insurance agents, and civil engineers, who must evaluate the risk of earthquakes of certain magnitude at potential sites [3]. A clear need exists to isolate end users from the complex requirements necessary for setting up earthquake simulations and executing them seamlessly over the grid. Workflow has also been used to describe, control the logic and data flows among grid services so that the burdens of the human to proprietary set up the grid applications are released. Because of its importance, grid workflow is regarded as a basic service in grid environment.

Since grid is a global resource sharing platform, the amount of users is tremendous. With more and more critical applications running on grid, the grid workflow systems will suffer heavier and heavier load. Such grid applications must execute hundreds or thousands workflow instances everyday. Therefore, the performance of

the grid workflow system becomes one of the key issues to achieve the success of grid applications. The centralized workflow engine in the grid workflow system is very possible to become a source of system failures and a bottle-neck for the system performance since the load may exceed its design capacity.

In the paper, a novel grid workflow engine is proposed. It is based on Jini platform and employs Jini services to provide functions of grid workflow engine. It can be expanded by adding new enactment services and client request can be allocated to a proper enactment service according to a load-balancing strategy.

The remainder of this paper is organized as follows. Section 2 presents a service based grid workflow engine. Section 3 introduces the structure and main function of workflow manager. Section 4 describes a prototype system and some testing results. Section 5 discusses related work. Finally, Section 6 provides some concluding remarks and gives an introduction to some future work.

2 A Service-Based Grid Workflow Engine

Grid workflow engine plays a central role for workflow enactment. We proposed a service-based grid workflow engine, which is divided into two service layers: management layer and enactment layer (See Fig.1). Both of them are composed of a set of services. The enactment layer as a whole provides those functions which are supported by most workflow engines. But all those functions are implemented as services and for each function, there may be several corresponding services, i.e., all these services provide the same function. The management layer is the core layer of distributed engine, which provides access, management and monitor interfaces to the enactment service layer. It is also composed of several services.

As shown in Fig.1, the management layer and enactment layer will communicate with each other and our implementation is based on Jini network. Jini is a new computing paradigm introduced by Sun Microsystems that can provide a network wide

Fig. 1. A Service-based Grid Workflow Engine

plug-and-play environment [4]. The services are the entities that can be used by a person, a program, or another service within the federation. The infrastructure supports the discovery and joint protocol that enables services to discover and register with lookup services. Jini technology provides us with much functionality to establish a flexible, stable and extensible distributed computing environment.

3 The Structure and Main Functions of Workflow Manager

3.1 The Structure of Workflow Manager

In the management layer, each service wraps a workflow manager. The main functions of each workflow manager includes enactment service proxy caching, load balancing for enactment services, remote event response, enactment service management, persistent data storage and client request dispatching. It can be further divided into several parts (See Fig.2):

Fig. 2. The Internal Structure of Workflow Manager

- **Service Cache Manager (SCM):** to maintain an enactment service cache and be responsible for dynamically discovering enactment service. SCM will cache distributed enactment service proxy in local machine and maintain the content of service cache automatically when it finds any status change of available enactment services. For example, SCM may delete unavailable enactment service record or update the attributes of some enactment service so that all enactment services kept in the cache are available and their information is always correct.
- **Service Manager (SM):** to provide interfaces to manage enactment service remotely. For example, stop or launch a service, modify the attributes of service or service registry information in Jini network.
- **Load-Balancing Manager (LBM):** to be responsible for managing the load distribution within enactment services. LBM dynamically computes and adjusts the load of each enactment service according to some load-balancing strategy in order to keep the whole distributed environment in a balance status.
- **Service Dispatcher (SD):** to receive all the user requests, check the user authority and judge the validity of user requests. Only valid user requests could be accepted and processed. According to the workflow execution strategy and global

load-balancing strategy, SD finds a proper enactment service to handler user request and returns the result to user.

- Remote Event Manager (REM): to be responsible for receiving and processing all events happening during the execution of enactment services.
- Persistence Manager (PM): to store the persistent data during the execution of workflow. Persistent data includes service log, configuration file of workflow manager and all information of enactment service proxy in service cache and so on.

Since management layer provides the uniform access interfaces to the client, the process of accessing, managing and configuring enactment services is simplified.

3.2 System Re-configuration

Since this grid workflow engine is based on services, whenever services join in or be removed from the enactment layer the system is reconfigured. The process that enactment services join in workflow manager is shown in Fig. 3. Firstly, enactment service uses Jini Discovery Protocol to find lookup service in Jini network. When lookup service is discovered, it uploads its own proxy and some attributes' information to lookup service through Jini Service Join Protocol. At the same time, workflow manager obtains the enactment service proxy and check whether there is a new enactment service or not. Therefore whenever new enactment services enters into Jini network, the workflow manager will be notified and then download service proxy and store it into the service cache.

Fig. 3. The Process of Service Joining in Workflow Manager

3.3 Client Requests Handling

The process that workflow manager handles requests from client is shown in Fig 4. When workflow manager receives a request from the client, it passes the requests to the service dispatcher. The service dispatcher determines whether there is a specific service responsible for responding to this type request or not according to a predefined rule. If there is, this related service is invoked. Otherwise, service dispatcher asks load-balancing manager to select a proper service. Load-balancing manager will select a service proxy from the service cache according to a load-balancing strategy and return this service proxy to service dispatcher so that this service can be invoked.

Fig. 4. The Process of Accessing Services through Workflow Manager

3.4 Load Balancing

Currently load-balancing manager uses a simple dynamic load-balancing algorithm to maintain the stability of load distribution in grid workflow system. The load of each enactment service is calculated according to the function:

$$WL = \sum_{i=1}^n P_i W_i$$

where P_i is the value of a enactment service attribute, for example the average response time or the waiting time for handling requests. W_i is the weight of this attribute. Load-balancing manager computes load of each enactment service at regular intervals and selects the service with best performance to handle the request from clients.

4 A Prototype System

We have implemented a prototype system called SGWE (Service-based Grid Workflow Engine), which includes management layer, enactment layer and a management tool. Enactment services mainly include workflow model instantiation, starting a task, submitting a task, accessing work items and renting remote event.

Fig. 5 shows the management tool we developed. In this tool, all available enactment services and executing tasks are listed and monitored. The parameters of each service, such as the number of concurrent users, the max number of concurrent users, CPU usage percent and memory usage percent are also shown. All messages exchanged in different parts of workflow system, such as the execution result of enactment service and the response of remote event are also recorded and shown in this tool.

In order to evaluate the performance of SGWE, experiments were performed to analyze average response time of client request in such a distributed environment. In order to simulate real client access patterns, a request sequence was generated by using a random number generator to produce requests in a given time interval. The results are shown in Fig. 6. Obviously, it can be found that with the increase of the number of enactment service users, the average response time is increasing. However, in the case of SGWE, the average response time is much smaller than that in the case of single enactment service. And the gap between these two cases is increasing with the increase of the number of users, which shows the advantage of distributed workflow system.

Fig. 5. A Management Tool of SGWE

Fig. 6. Experiment Results of Distributed Grid Workflow Engine

5 Related Works

Since grid workflow plays a very important role in grid application, there are many ongoing grid workflow projects. In [5] some well-known grid workflow systems are classified according to their frameworks. But unfortunately, the structure of grid workflow engine is not included in this framework. As far as we know, most grid workflow systems employ a centralized structure, for example, GSFL [6], GridAnt [7] and Condor [8]. In [9], the structure of GeneGrid Workflow is introduced, when a new workflow instance is running, the workflow management factory will generate a workflow manager service instance for this workflow instance. Although workflow manager service instances can be distributed into different machines, this topic is not discussed in this paper.

There are already many distributed business workflow management systems in the market. Although the components of these systems are distributed, their structures can not be changed dynamically. In [10], a goal-driven autoconfiguration tool is introduced. This tool aims to recommend an appropriate system configuration in terms of replicated workflow, application, and communication servers, so as to meet given goals for performance, availability, and performability at low system costs. The basic idea of our system is very similar to theirs. In their system, workflow engine can have several copies and load is distributed to these engines. In our system, the interfaces of

workflow engine are implemented as separate services, which can be deployed independently. It increases the scalability and also flexibility of the whole engine. Besides this difference, they constructed their tool based on CORBA, while in our system, Jini is adopted so that the system can be re-configured more easily with the help of Jini services. In [11], the load-balancing technology for distributed WFMSs is discussed and a workflow load index to measure load level of workflow engines is defined. But it did not discuss how to develop a re-configurable workflow engine.

6 Conclusions and Future Work

In grid environment, it is possible an application has tremendous concurrent users. Since workflow is a basic service in such environment, the heavy load will lead centralized workflow engine become a bottleneck. In this paper, we proposed a distributed re-configurable grid workflow engine. In this architecture, enactment services can be added or removed on demand. Therefore, the whole engine will have high performance and high availability.

Although we have implemented a prototype system, there are many functions waiting to be developed. Future work will include but not limit to:

- (1) To design an algorithm to calculate the load brought by each request so that we can allocate this request to a service with corresponding capability;
- (2) To design more efficient load-balancing algorithm, which can optimize the parameters itself;
- (3) To develop a self-deploying architecture. In this architecture, the engine can find the available resources and when the load is increased, it will deploy an enactment service to this resource. And when the load is decreased it can shut down this service automatically. We have developed the auto-deploying tool for services and we will integrate this tool into the system.

Acknowledgements

This research is supported by Chinese NSF Project (No. 60503041). This work is also partly supported by "SEC E-Institute: Shanghai High Institutions Grid", Chinese Semantic Grid Project (2003CB317005) and Chinese NSF Project (No.60473092).

References

1. Foster, I., Kesselman, C., Tuecke, S., The Anatomy of the Grid: Enabling Scalable Virtual Organization, International Journal of Supercomputer Applications, Vol. 15(3), 2001, pp200-222
2. Foster, I., Kesselman, C., et.al., The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International Journal of Supercomputer Applications, Vol. 15(3), 200-222, 2001

3. Yolanda Gil, Ewa Deelman, Jim Blythe, Carl Kesselman, Hongyuda Tangmunarunkit, Artificial Intelligence and Grids: Workflow Planning and Beyond, IEEE Intelligent Systems, 2004.1, pp26-33
4. Jini(TM) Architecture Specification. http://www.sun.com/software/jini/specs/jini2_0.pdf. Version 2.0 June 2003.
5. Jia Yu and Rajkumar Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, Technical Report, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005
6. Krishnan, S., Wagstrom, P., von Laszewski, G.: GSFL: A Workflow Framework for Grid Services. Technical Report, The Globus Project, <http://www-unix.globus.org/cog/projects/workflow/gsfl-paper.pdf>, 2002
7. von Laszewski, G. K. Amin, M. Hategan, N. J. Zaluzec, S. Hampton, and A. Rossi, GridAnt: A Client-Controllable Grid Workflow System. In 37th Annual Hawaii International Conference on System Sciences (HICSS'04) IEEE Computer Society Press, Los Alamitos, CA, USA, January 5-8, 2004
8. Condor: The Directed Acyclic Graph Manager. <http://www.cs.wisc.edu/condor/dagman/>, 2003
9. David R. Simpson, PV Jithesh, Noel Kelly et al, GeneGrid: A Practical Workflow Implementation for a Grid Based Virtual Bioinformatics Laboratory, Proc. of the UK e-Science All Hands Meeting 2004 (AHM04), 2004, pp. 547-. 554
10. Michael Gillmann, Jeanine Weissenfels, German Shegalov, Wolfgang Wonner, Gerhard Weikum, A Goal-driven Auto-Configuration Tool for the Distributed Workflow Management System Mentor-lite, <http://www.mpi-sb.mpg.de/departments/d5/software/mlite/papers/Demo-SIGMOD-submitted.pdf>, 2004
11. Li-jie Jin, Fabio Casati, Mehmet Sayal, Ming-Chien Shan, Load balancing in distributed workflow management system, Proceedings of the 2001 ACM symposium on Applied computing 2001, Las Vegas, Nevada, United States pp 522 - 530