Agent-Based Middleware Architecture for Workflow in Grid Portals

Sangkeon Lee[1], Jaeyoung Choi[1], and Keumwon Cho[2]

[1] School of Computing, Soongsil University,
1-1 Sangdo-dong, Dongjak-gu, Seoul 156-743, Korea
seventy9@ss.ssu.ac.kr, choi@ssu.ac.kr
[2] Supercomputing Center, Korea Institute of Science and Technology Information,
52 Eoeun-dong, Yusung-gu, Daejun 305-306, Korea
ckw@kisti.re.kr

Abstract. In this paper, we propose an agent-based middleware architecture for the workflows used in Grid portal. We combined workflow model, workflow management system, job distribution, parameter scheduling, and service inteface in the middleware architecture. The workflow model consists of three layers. The agent-based middleware architecture consists of five agents and three types of communication protocols. Users can design a lightweight, flexible and effective middleware, which can be used to construct a workflow-based Grid portal.

1 Introduction

Grid portals[1][2] are widely used to construct virtual scientific research environments to integrate various types of software such as simulation software, analysis software and visualization software. A scientific research may require iterative execution of the experiments with experimental data or change of parameters. The number of iteration can be larger than billion times. To process billions of experiments on computing resources, Grid computing is necessary. Scientists design their research with workflows[3][4], which integrates the software and experimental data with a flow of research scenario. Therefore, virtualization, resource scheduling[5], data management, security management, information integration, ontology processing[6], high-speed network, and other facilities should be integrated to construct a Grid portal.

However, construction of e-Science environments using Grid portal is not simple because Grid computing requires setting and managing tens of middleware and integrating them as a single system. The most difficult problem is that those middleware are designed separately, so integrating the heterogeneous middlewares causes many troubles.

Therefore, integrating complex layers between Globus Toolkit[7] and user interface into a single design is required and design of the integrated system must take care of all matters such as it's workflow model, workflow management, job distribution, parameter scheduling[8], and the service oriented architecture.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 972–975, 2006.

In this paper, we propose an agent-based middleware architecture for workflow design in e-Science environments. This architecture uses a service-oriented workflow model, which is developed by our prior research, called Meta Services[9], which integrates functionalities mentioned above. Meta Services provide an approach to abstract and map a workflow for a service by overriding workflow's attributes with the service's parameters.

By combining Meta Services with an agent-based middleware architecture, we designed and implemented a lightweight, flexible, and effective middleware architecture which can be used to construct workflow-based Grid portals.

2 Agent-Based Middleware Architecture for the Workflow Model

The Agent-based middleware architecture is shown in Figure 1. It consists of five major agents : AM (Access Manager), OM (Ontology Manager), SM (Service Manager), RM (Resource Manager), and EM (Execution Manager). Our workflow model has three layers: SM on service layer, RM on flow layer, and EM on task layer. Description of services, flows, and tasks is stored as XML files and managed by OM. Finally, AM is required to handle security such as authentication and access control.

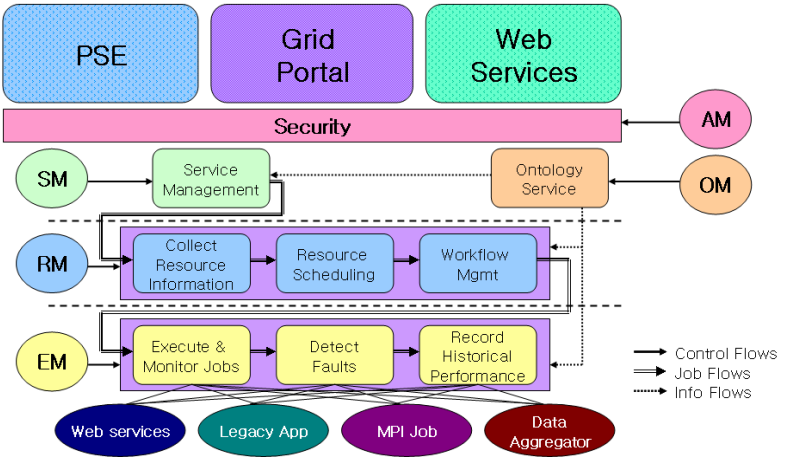

Fig. 1. Agent-based Middlleware Archtecture

2.1 Communication Among Agents

There are three types of communication protocols in the agent-based middleware architecture: admin protocol, user protocol, and agent protocol. An administrator has an administrator's authority, and he can connect or manage SM, RM, and EM. Users can connect to OM through a GUI editor, and they can browse, search, and compose their works such as tasks, flows, and services depending on

their authorities. After a workflow scenario is prepared as a service by searching or composing, then a user submits the scenario to SM. Then, SM processes the overriding of workflow parameter and submits the changed flow to RM. After this process, RM analyzes the flow and distributes tasks to EMs, which connected to the RM. If a task is allocated to an EM, then the EM executes the task when a processor is available. Execution of flow and task is also performed with a user's authority, and it doesn't interrupt other user's process.

2.2 Distribution of Flow and Task

In our agent-based middleware architecture, an EM has a parameter named 'number of processing element'. Its value reflects how many tasks can be executed on the EM at the same time. By default the value sets with a number of processor in which the EM is installed. An administrator can change it using SET_PE operation. EM can also cooperate with other Grid job distributors by reducing its concurrency using SET_PE operation. EM has a waiting queue whose size is multiple of processing element. When an EM is started, it registers itself to a RM, and the RM maintains a list of EMs with information such as the number of processors and the size of waiting queue.

Each RM has a priority queue which queuing list of users connected to AM. When a user is appended to the list, a flow queue is created for the user. If an EM's waiting queue is freed, RM fills the EM's waiting queue with one of the activatable tasks from the flow queue of a user who has the highest priority. After the task is finished, the task owner's flow queue updates the list of activatable tasks.

2.3 Parameter Scheduling

Instead of appending prefix or postfix to data or parameter, parameter scheduling is performed by separation of tasks' working directory. When a task is executed, EM creates a temporary working directory /msftmp/⟨*username*⟩/⟨*flow*⟩/⟨*username*⟩/⟨*task id*⟩ under a local user's home directory permitted by the gridmap file of hosts where the EM is executed. RM allocates a flow ID to the flow when it is submitted and task ID is assigned in a flow description stored in OM. When the task is completed its execution, the data file marked as an outputfile in the flow description is transferred to the next task's working directory. Therefore, even if a user doesn't override the name of an input or an output file, file corruption by conflict file names never occurred.

3 Conclusion and Future Work

In this paper, we present an agent-based middleware architecture for workflow-based Grid portals. The architecture contains essential functionalities for Grid portals. The functionalities include workflow, service interface, job distribution, and parameter scheduling. Our agent-based architecture consists of five agents and each agent has its own role over the corresponding workflow model. By using

this middleware model, users can design a lightweight, flexible, and effective middleware which is used to construct workflow-based Grid portals.

We have remodeled our pervious middleware MSF[10] using the agent-based middleware architecture and have implemented a prototype. We tested the middleware with simple scenarios which are used in Bio-grid portal for drug discovery. Workflow editor, management console, more complex workflow pattern, and performance optimization are required to use our middleware practically. So we are going to design and implement this features and continuously upgrade our middleware architecture.

Acknowledgement

"This work was supported by the National e-Science Project, funded by the Korean Ministry of Science and Technology (MOST)."

References

1. Allen, G., (ed.): Enabling applications on the Grid - a GridLab overview. Intl. Journal on High Performance Computing Applications. **17 4** (2003) 449–466
2. Stevens, R., Robinson, A., Goble, C.: myGrid: personalized bioinformatics on the information grid. Bioinformatics. **19 1** (2003) 302–304
3. Cao, J., Jarvis, S., Saini, S., Nudd, G.: GridFlow: Workflow Management for Grid Computing. 3rd International Symposium on Cluster Computing and the Grid. (2003) 12–15
4. Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M., Vahi, K.: Pegasus Mapping Scientific Workflows onto the Grid. In Proceedings of across Grid EU Conference. (2004)
5. Litzkow, M., Livny, M., Mutka, M.: Condor - A Hunter of Idle Workstations. 8th International Conference of Distributed Computing Systems. (1998) 13–17
6. Wroe, C., Stevens, R., Goble, C., Roberts, A., Greenwood, M.: A suite of DAML+OIL Ontologies to Describe Bioinformatics Web Services and Data. International Journal of Cooperative Information Systems special issue. **12 2** (2003) 197–224
7. Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. Intl. J. Supercomputer Applications. **11 2** (1997) 115–128
8. Abramson, D., Sosic, R., Giddy, J., Hall, B.: Nimrod: A tool for performing parametised simulations using distributed workstations. In Proc. of the 4th IEEE Symposium on High Performance Distributed Computing. **8** (1995)
9. Lee, S., Choi, J.: Meta Services: Abstract a Workflow in Computational Grid Environments. Lecture Notes in Computer Science, Vol. 3516. Springer-Verlag, Berlin Heidelberg New York (2005) 916–919
10. Hwang, S., Choi, J., Park, H.: Meta Scheduling Framework for Workflow Service on the Grids. Lecture Notes in Computer Science, Vol. 3036. Springer-Verlag, Berlin Heidelberg New York (2004) 445–448