

A Grid Flow Management System for the Problem Solving Environment Based on Intelligent Collaborative Objects

H. M. Chan & T. Hung

Institute of High Performance Computing, 1 Science Park Road, #01-01, The Capricorn, Singapore Science Park II, Singapore 117528, SINGAPORE

chanhm@ihpc.a-star.edu.sg

Abstract: This paper describes a framework for the grid flow management system in a problem solving environment (PSE) based on the Autonomous Manager Grid Service (AMGS). The design of AMGS is based on principles of agency, intelligence, collaboration and co-operation. Using AMGS that carry out tasks on behalf of machines participating in a computational grid, it is possible to build grid flow systems that bring further improvements in task scheduling, resource allocation and reallocation, and some degree of fault tolerance to the overall computational grid. AMGS prototype architecture is implemented in Java using Protégé 2.0.1, JADE and JESS. This research shows the feasibility of an agent-based grid middleware.

1 INTRODUCTION

1.1 Background

Problem-solving environments (PSEs) represent an ongoing and evolving field of research (Fox *et al.*, 2003) with immense potential as a killer application on grid computing. In order to free the computational scientists and engineers from the complexities of grid technologies, so that they can concentrate on solving the problems in their respective domains, Yeo *et al.* (2004) have devised an agent-based grid flow management system (GFMS). This system provides an abstraction of a typical Computational Science & Engineering (CSE) workflow, thereby greatly simplifying the job of a computational scientist when he leverages on the computational powers of the grid.

To further illustrate, a typical CSE workflow consists of modeling, meshing, numerical analysis, and visualization. Models are constructed as an approximation or surrogate for design and analysis. This process may be repeated a few times to fine-tune the design in order to arrive at an optimized solution. It is useful to represent the whole CSE workflow as an abstract grid flow for ease of management and use. The grid flow could be described using a workflow language, such as the Web Service Flow Language (WSFL) as proposed by Leymann (2001). This file can then be read by the GFMS, and processed for execution on the grid resources.

At the heart of the GFMS is a Process Engine – Eisenhower and a federation of Autonomous Manager Grid Service (AMGS) implemented as intelligent collaborative objects or agents. The decision to implement the AMGS as agents is an astute choice. This is because the grid environment resembles a Multi-Agent System (MAS) more closely than other systems such as a type of Distributed Problem Solving (Tveit, 2002). This paper presents the approach taken to design and implement the AMGS conceptualized by Yeo *et al.* (2004).

1.2 Resource Management Issues

In a computational grid environment, resource management is one of the core capabilities that the grid infrastructure must provide. Nabrzycki *et al.* (2004) defines grid resource management

as “the process of identifying requirements, matching resources to applications, allocating those resources, and scheduling and monitoring Grid resources over time in order to run Grid applications as efficiently as possible.”

Resource management is a well researched topic in the field of traditional computing systems (Foster & Kesselman, 2004). Resource managers in the form of batch schedulers, workflow engines, and operating systems have been developed. When applied to the grid, there is a whole new set of issues unique to the grid that have surfaced. These resource management issues arise from: 1) task submission, 2) workload management, 3) on demand access, 4) co-scheduling, and 5) resource broker scenarios. These are further elaborated by Czajkowski *et al.* (2004). In addition, there are also many challenges presented by the security issues associated with grid resource management (Thompson & Jackson, 2004).

To address this, there are several well-known resource management and scheduling solutions such as Condor (Thain & Livny, 2004), Platform’s Load Sharing Facility, Veridian’s Portable Batch System, and IBM’s LoadLeveler. While these systems have addressed the heterogeneous nature of the grid resources, there is still a gap to be bridged between best-effort service and a guaranteed level of service in today’s Grid environment (Foster & Kesselman, 2004).

Schopf (2004) further described three phases in the Grid scheduling approach that covers the various aspects of resource management as defined earlier. The three phases are: 1) resource discovery, 2) information gathering, and 3) job execution. In the resource discovery phase, the steps performed are authorization filtering, job requirement definition and filtering to meet the minimal job requirements. The two steps performed in the second phase are to gather detailed dynamic information on the systems available and select the system on which to execute the job. The final phase is the actual job execution which consists of steps like advanced reservation, job submission, preparation tasks, progress monitoring, job completion, and cleanup tasks.

In this paper, we narrow the scope to focus our efforts on the second phase of Grid scheduling, which involves the actual selection of a best set of resources on which to execute the job. To fulfil the requirements for this phase, we need an efficient grid information service that can provide the necessary dynamic in-

formation on the resources and services available. Thereafter, we need a mechanism for selecting the resources.

1.3 Problem

The problem that is addressed in this paper is: *how to enable efficient resource negotiation and allocation in an agent-based grid workflow?*

More particularly, how to create an architecture that: 1) *uses a mechanism that is based on a market model*, such as an auction mechanism where multiple qualified resources bid in an effort to be compensated for running a job; 2) *is pluggable with respect to allocation algorithms*. This could be achieved through the use of JESS modules (Friedman-Hill, 1998) within the JADE framework (Bellifemine *et al.*, 2000).

The rest of this paper is organized as follows. Section 2 contains the system architecture and detailed descriptions of the different types of agents involved. Section 3 discusses the technologies employed in the implementation of the prototype GFMS. Section 4 describes related work, and finally section 5 contains conclusions of the research presented here with suggested directions for future work.

2 ARCHITECTURE

The architecture of AMGS is based mainly on the JADE framework developed by Bellifemine *et al.* (2000). The JADE framework has been chosen for the following reasons: 1) JADE is a FIPA-compliant agent framework. This implies that the agents developed using this framework would be able to interoperate with agents using other frameworks that are compliant with the FIPA specifications (FIPA, 1997). FIPA provides a set of open specifications and this decision to embrace open standards is in line with the philosophy adopted by Globus, which is the de facto standard for grid middleware. The Globus Alliance has adopted the open source approach to develop the grid middleware, as evidenced in the Globus Toolkit Public License. 2) JADE is implemented using Java, which is a computer programming language geared towards object-oriented programming in distributed heterogeneous environments such as the grid (Bellifemine *et al.*, 2000). 3) The behavior abstraction of the JADE agent model allows the integration of external software such as the usage of JESS as the agent-reasoning engine (Friedman-Hill, 1998). This feature allows the easy replacement of allocation algorithms in the AMGS, thus achieving the pluggable feature for the allocation algorithms of this system.

The PASSI methodology (Cossentino & Potts, 2002) is used in the design of the AMGS. The methodology comprises the construction of five models, namely: 1) system requirements, 2) agent society, 3) agent implementation, 4) code model, and 5) deployment model. As this is only a prototype implementation, the deployment model is not constructed at this stage.

2.1 System Requirements Model

Following the PASSI methodology, there are four phases that guide the construction of the system requirements model. In the domain requirements phase, the domain requirements are first modeled using use case diagrams. Subsequently, the various functionalities are grouped together logically as agents during the agent identification phase as shown in Fig. 1. For the GFMS, we have identified four types of agents, namely *Eisenhower* (the process engine), the *Autonomous Host Manager* (AHM), the

Task Manager Agent (TMA), and the *Local Job Manager Agent* (LJA). Only the Eisenhower and the AHM will interact with the Virtual Organization Information Services (VOIS), which is implemented as a web service (Zang *et al.*, 2004). After identifying the agents required, the interactions among the different agents and their roles are described in the form of a sequence diagram in the roles identification phase. The final phase of task specification describes how the Eisenhower and the federation of AMGS can use its tasks to accomplish their roles.

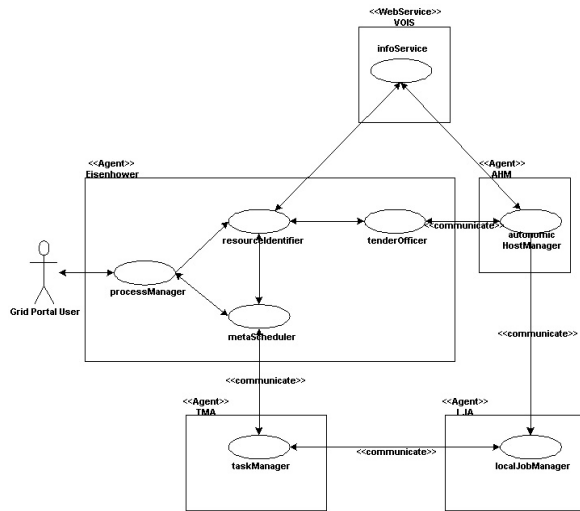


Fig. 1. Use-Case diagram showing the agents identified for GFMS.

2.2 Virtual Organization Information Service (VOIS)

The VOIS is a crucial component of the GFMS as it provides the critical dynamic information on the resources available, thus enabling the process engine to select the system and allocate the resource in the most optimal manner. Fig. 2 shows the architecture of the VOIS. It comprises of three layers, namely the virtual organization (VO) layer, the site layer and the resource layer.

From the bottom up, the resource layer provides the node information service. These are dynamic information concerning the physical node that changes frequently. As the name suggests, the site layer provides the site information service. These are information pertaining to the collection of nodes within a geographical region, such as the NTU campus grid. The information at this layer is less dynamic when compared to the resource layer. In addition, the site information service may serve more than one virtual information service (VIS) in the VO layer. The VO layer provides the information pertaining to the whole virtual organization. In the Singapore context, this is equivalent to the National Grid Pilot Platform (NGPP). The information at the top layer is mostly static information for the whole VO, such as CPU type, total memory, and total hard disk space. This is also the layer that interfaces directly with the users of the VOIS.

As to the type of information stored in the database at the VO layer, there are two models currently being considered, namely the *source data* model and the *metadata* model. In the source data model, all the dynamic information are being transferred and stored in the database at the VO layer for easy access by the information service clients. However, this model puts a strain on the network due to the amount of dynamic information that will be sent up the layers. The second model is the metadata model, where only the metadata from the lower layers is stored in the

VO layer database together with the source data from the VO layer. When the information service client requires information that is only stored as a metadata in the database, the metadata will point the client to the data source which is either at the site layer or the resource layer. While this model may degrade the user experience, it is much more scalable compared to the source data model as it is less demanding on the network bandwidth.

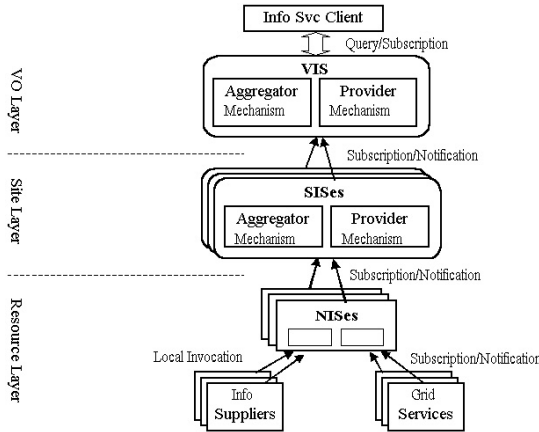


Fig. 2. Diagram showing the architecture of the Virtual Organisation Information Service (VOIS).

2.3 Eisenhower

Eisenhower is the process engine of the AMGS. It receives the grid flow document from the grid portal user, decomposes the grid flow into individual jobs and tenders out the jobs to a host of resources based on a reverse auction model. When the tender closes, it then evaluates the tender returns and decides on the resource that will be awarded the job. At this point, it will spawn off a task manager which will monitor the job execution on behalf of the process engine.

2.4 Autonomous Host Manager

The Autonomous Host Manager (AHM) is the agent that is running on every grid resource. Its role is to bid for jobs offered by Eisenhower on behalf of the grid resource that it is running on. It will also query the information service for dynamic data on the resource layer as part of the bidding process. When it is informed by Eisenhower that its bid was successful, AHM will spawn a Local Job Manager (LJA) to execute the job on the resource, while it continues to listen for more tender requests from Eisenhower.

2.5 Task Manager

The main job of the Task Manager is to monitor the job execution on behalf of Eisenhower. It is also responsible for reporting the status of the job to Eisenhower, which will then report back the status of the grid flow to the grid portal user. In the event that a particular resource breaks down, the Task Manager is able to act autonomously to reassign the job to another resource, thus providing some level of fault-tolerance.

2.6 Local Job Agent

The Local Job Agent is an agent that executes the job on behalf

of the AHM on the grid resource. It receives the job inputs from the AHM and reports back the status and the job outcome to the Task Manager.

3 IMPLEMENTATION

The implementation of AMGS is still work in progress; so far the overall architectural choices and a prototype implementation have been made.

3.1 Implementing AMGS in Java

The Java language was chosen for implementing the AMGS. For the networking needs of the system, Java RMI was used for intra-platform agent communication while CORBA was used for cross-platform agent communication as provided for under the JADE framework.

3.2 Other Tools and Frameworks

Protégé 2000 was used to define the agent ontology for the AMGS. By installing the Beans Generator plug-in for Protégé 2000, the software is able to automatically generate the coding stubs for the various Java beans defined in the ontology.

Finally, JESS provides the expert system shell environment to embed the artificial intelligence needed by the agents in the AMGS to behave in an autonomous fashion.

4 RELATED WORK

The architecture that resembles AMGS most is ARMS (Agent-based Resource Management System) conceptualized by Cao (2002). Like AMGS, ARMS is based on an agent-based approach where an agent is used to make resource allocation decisions based on its in-built agent codes. However, AMGS is different from ARMS in that ARMS achieves the agent autonomy through an integration with the PACE (Performance Analysis and Characteristic Environment) toolkit, which provides ARMS with the capability to predict the grid resources available. AMGS on the other hand uses the market paradigm to achieve the agent autonomy, where the agents act as self-interested entities competing on a market, and the goods being traded are the computational resources.

5 CONCLUSION

In conclusion, this paper proposes AMGS and a market model for implementing the resource negotiation and allocation. The AMGS approach has the potential to enhance the GFMS by optimizing the resource allocation and providing fault-tolerance during the job execution. Further research along this line could address issues of optimization of multiple grid flows.

ACKNOWLEDGMENTS

I would like to thank my IHPC colleague Xiang Wei for the head start discussions on software development using JADE and Pro-

tégé. I also would like to thank Jie Wei, Zang Tianyi, Lei Zhou, Cai Wentong and Stephen Turner for their valuable discussion and interest in this work, as well as my supervisor Terence Hung. This work is supported by the A*STAR Science and Engineering Research Council in the framework of the Science and Engineering Research Grid and Portal project.

REFERENCES

- Bellifemine, F., Poggi, A. and Rimassa, G. 2000. JADE – A FIPA-compliant agent framework. *Proceedings of the 5th International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM)*: 97-108.
- Cao, J. 2002. ARMS: An agent-based resource management system for grid computing. *Scientific Computing*: 135-148.
- Cossentino, M. & Potts, C. 2002. A CASE tool supported methodology for the design of multi-agent systems. *The 2002 International Conference on Software Engineering Research and Practice (SERP'02)*. June 24-27, 2002 – Las Vegas (NV), USA.
- Czajkowski, K., Foster, I. and Kesselman, C. 2004. Resource and Service Management. In Foster, I. and Kesselman, C. (eds.), *The Grid: Blueprint for a New Computing Infrastructure, Second Edition*: 259-283. San Francisco: Morgan Kaufman.
- FIPA – Foundation for Intelligent Physical Agents. 1997. Specifications. Available from <http://www.fipa.org>.
- Foster, I. & Kesselman, C. 2004. The Grid in a Nutshell. In Nabrzyski, J., Schopf, J., and Weglarz, J. (eds.), *Grid Resource Management*: 3-13. Norwell: Kluwer Academic.
- Fox, G., Dongarra, J., Arnold, D., Casanova, H., Catlin, A. C., Haupt, T., Houstis, E., and Rice, J. R. 2003. Problem-Solving Environments. In Dongarra, J. *et al.* (eds.), *Sourcebook of Parallel Computing*: 409-442. San Francisco: Morgan Kaufman.
- Friedman-Hill, E. 1998 Java Expert System Shell (JESS). Available from <http://herzberg.ca.sandia.gov/jess>.
- Leymann, Frank. 2001. *Web Services Flow Language (WSFL 1.0)*. IBM Software Group. Available from <http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf>.
- Nabrzyski, J., Schopf, J., and Weglarz, J. (2004), *Grid Resource Management: State of the Art and Future Trends*. Norwell: Kluwer Academic.
- Schopf, J. 2004. Ten Actions When Grid Scheduling. In Nabrzyski, J., Schopf, J., and Weglarz, J. (eds.), *Grid Resource Management*: 15-23. Norwell: Kluwer Academic.
- Thain, D. & Livny, M. 2004. Building Reliable Clients and Services. In Foster, I. and Kesselman, C. (eds.), *The Grid: Blueprint for a New Computing Infrastructure, Second Edition*: 285-318. San Francisco: Morgan Kaufman.
- Thompson, M. R. & Jackson K. R. 2004. Security Issues of Grid Resource Management. In Nabrzyski, J., Schopf, J., and Weglarz, J. (eds.), *Grid Resource Management*: 53-69. Norwell: Kluwer Academic.
- Tveit, A. 2002. jfipa – an Architecture for Agent-based Grid Computing. *Proceedings of the Artificial Intelligence and the Simulation of Behavior Convention, Symposium on AI and Grid Computing*. London: AISB.
- Yeo, B. K., Hung, T., and Khoo, B. 2004. An Agent-based Grid Flow Management Framework for the Problem Solving Environment (PSE). *Proceedings of GlobusWorld 2004*. January 20-23, 2004. San Francisco (USA).
- Zang, T. *et al.* 2004. The Design and Implementation of an OGSA-based Grid Information Service. *Proceedings of the 2004 IEEE International Conference on Web Services (ICWS 2004)*. July 6-9, 2004. San Diego, California (USA) (accepted paper).