AGRID - Agent Based Grid System

Uygar Gümüş
Institute of Science and Technology
Istanbul Techical University
Maslak, İstanbul, Turkey
gumusuy@itu.edu.tr

Prof. Dr. Nadia Erdoğan
Computer Eng. Department
Electrical-Electronics Faculty
Istanbul Techical University
Maslak, İstanbul, Turkey
nerdogan@itu.edu.tr

ABSTRACT

This paper presents the design and implementation of an agent-based grid system (AGrid) that provides clients with a distributed execution environment for sharing of processing power resources. AGrid combines favorable aspects of two different areas of distributed computing, namely grid computing and agent technology. During the design phase, system stability and robustness has been a primary concern. The framework builds on various types of agents that are defined and implemented to handle different issues of grid computing. Each type of agent acts according to protocols that define the interaction and coordination between agents and describe actions required for the management of the grid. This paper describes in detail the procedures followed for connection and disconnection of clients and workers, task assignment, task execution and result delivery.

Categories and Subject Descriptors

D.1.3 [**Concurrent Programming**]: Distributed programming; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Design

Keywords

Agent systems, mobile agents, grid computing, computational grids, JADE.

1. INTRODUCTION

Distributed computing has become one of the popular research topics in computer science. Especially, very high speed Internet connections and new networking structures enable promising research to be conducted in this field. This paper, presents a new agent-based grid system, which combines two different areas of distributed computing, namely grid computing and agent technology.

Grid computing is a model for wide-area distributed and parallel computing across heterogeneous networks, aiming to reach breakthrough computing power at low cost [3]. Grids are hardware and software infrastructures that enable the sharing, distribution and collective use of heterogeneous resources [8]. These resources may be secondary storage, processing power or output data of any specific input output device. The grid system we present focuses on sharing of processing power resources. Reliability and stability are important specifications of grids systems. Data security and trustworthiness of calculation results are very important. Grid systems need mechanisms to manage the grid infrastructure as to ensure these issues.

Agents are encapsulated and autonomous software and hardware systems, which execute an assigned task by communicating and collaborating with other actors at the same or different physical environments [6]. Main attributes of agent systems are flexibility and autonomy. In traditional agent systems, generally no single agent controls the system. Each agent has limited information about the problem and limited capability to solve the problem. Agents build a virtual organization using their communication capabilities and solve the problem by combining the insufficient capability of each agent through intensive cooperation. In the context of grid computing, mobile agents are usually employed in resource discovery, job scheduling, job deployment, task execution and result collection [7].

This paper presents an agent-based grid system (AGrid) for sharing of processing power resources. In AGrid, the framework builds on various types of agents that are defined and implemented to handle different issues of grid computing. Some agents handle job scheduling and job deployment, while others execute jobs assigned to them and produce results. In the design phase, we determined the actors of the system, specifying their tasks and responsibilities. After associating each actor with an agent type, protocols were developed for each role of agents. These protocols define in detail the interaction and coordination between agents and describe actions required for the management of the grid. Methods of connection, disconnection, task assignment, task executing and result delivery are declared. In addition, an efficient and fast messaging infrastructure is developed for effective agent communication. This paper presents the agent types, their responsibilities in the grid systems and the communication protocols between agents during the life cycle of the grid. Protocols on task assignment, job scheduling and result collection as handled by agents are also described in detail. The grid was developed in a systematical manner and up to the final version, three prior versions were implemented, detecting and making design decisions to fix problems of the prior version each time. Figure 1 depicts the final grid architecture.

AGrid is compatible with the Foundation for Intelligent Physical Agents (FIPA) standards [5]. The use of autonomy and flexibility features of agents in the grid design has resulted in a stable and robust grid structure.

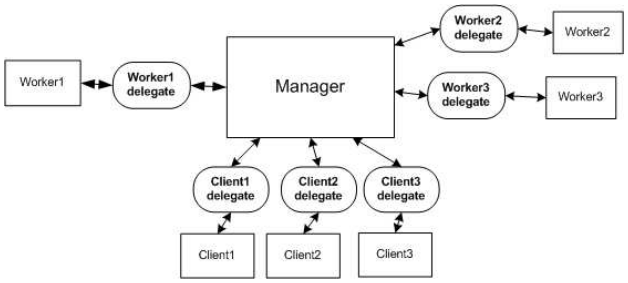

Figure 1: Final grid architecture.

The rest of the paper continues with Section 2, which briefly summarizes recent work in agent based grid computing. Next, Section 3 focuses on design and implementation issues of the system, describing the system components and runtime protocols. In Section 4, system performance is evaluated and, finally, Section 5 concludes the paper.

2. RELATED WORK

Agent based grid system has been a popular area in computer science in recent years. Athanaileas et al. argued about adding mobility support to grid systems using mobile agents [1]. This system, named GridSBAP, is build on OSGA platform and mainly focused on adding mobility feature to the grid systems. GridSBAP uses FIPA ACL standards for agent communication.

AgentScape, an agent supported Internet based grid, is developed by B. J. Overeinder et al.[10] It supports large scale agent system. This system defines its own communication protocols for agents and a resource management system for the grid. AgentScape can adapt to other communication standards using an extra layer which transforms messages in to the native format of AgentScape.

Fukuda and Smith introduced UWAgent, a grid system middleware for Java based on mobile agents [7]. Just like AGrid, UWAgent is not only an agent based grid system but also a middleware that meets management needs of distributed computing. However,UWAgent defines its own communication standards for mobile agents, which is not fully compliant with FIPA standards.

Also Poggi, Michele and Turci, worked on extending JADE framework in order to support grid computing [9].

3. AGENT BASED GRID SYSTEM

AGrid is an agent based infrastructure for distributed and parallel computing. Currently AGrid can be used to create a grid system to share processing power and task executing on remote platforms. The system is implemented on JADE (Java Agent Development Framework) which is developed by Telecom Italia SpA. JADE is a distributed runtime environment on which mobile agents can live, communicate and run parallel tasks via behaviours. JADE also supports graphical user interfaces that can be used for debugging, monitoring, logging and management of the agent system. In addition, JADE is compliant with FIPA specifications, which enables the agents to communicate and cooperate with other agent systems which are also compliant with the FIPA standards.[4].

As stated in the introduction section, multi agent systems and grid systems are two different branches of distributed systems with different perspectives on distributed computing. In this work, a hybrid of these two different approaches is implemented. In this section, we will present the components of the system and describe in detail the protocols between the agents.

3.1 System Participants

In AGrid, four different types of agents cooperate to provide a distributed computing environment:

- `Manager agent` which is in charge of general grid management,
- `Worker agents` which execute jobs assigned to them and produce results,
- `Client agents` which use the grid to run their tasks,
- `Delegate agents` which help the manager agent via coordinating the interaction and the communication between client or worker agents and the manager agent.

Figure 2 depicts the hierarchical relation between agent constituents of the grid.

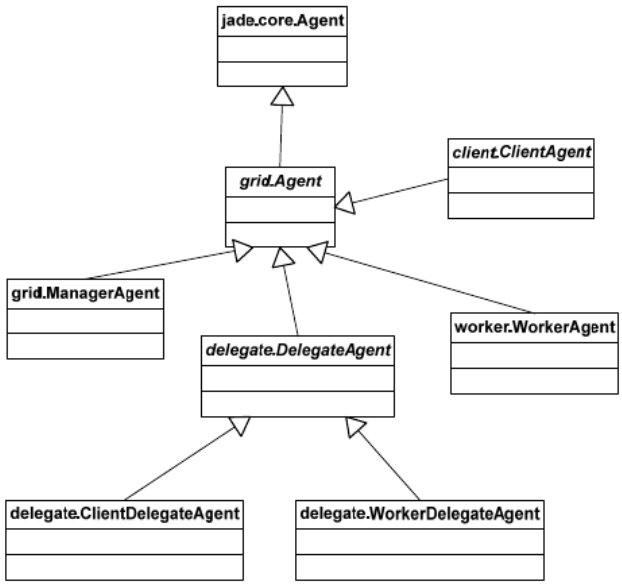

Figure 2: Hierarchical relation between agents.

The following sections give detailed information for each type of agent.

3.1.1 Manager Agent

AGrid has a central management system. There exists a manager agent which controls communication channels, task assignment and result collection issues. Classical agent systems generally do not contain a central management organization [11]. However, a computational grid system usually needs a management structure to control the entire grid [2]. Central management usually becomes a bottleneck; therefore the manager agent in AGrid conveys some of its tasks to delegate agents as to decrease its work load. The manager agent has the following responsibilities:

- Keeps the record of each connected client and worker agent.
- Creates a delegate for each worker and client agent that connects to the system.
- Ensures safely disconnection of participant agents.
- Coordinates reassignment of a task in the case of improper termination/exit of worker agents.
- Terminates task execution in the case of corresponding client agent exits the system.

3.1.2 Client Agents

Client agents are users of AGrid that connect to the grid system to receive service. They connect to the system in order to have their local tasks be executed on remote hosts which can lend their processing power. After a client connects to the grid, it sends the required task information to the delegate agent with which it is associated and waits for the result. It may disconnect while the computation is going on and may later collect the results

3.1.3 Worker Agents

These agents connect to the system to share any of their resources. Currently, AGrid only supports computing power based resources. When a worker agent connects to the system, it provides the manager agent information about its resources that are available and it is willing to share. If the participation request to the grid is accepted, the manager agent creates a delegate agent for the worker, which handles all further communication/ interaction of the worker agent with the grid system.

3.1.4 Delegate Agents

Delegate agents reside on the node where the manager agent is present and handle all communication and protocol implementation between the agents they are representative of and the manager agent. Over the life time of a computation, each agent that participates in the process needs to communicate frequently with grid management. Delegate agents are representatives of those agents and they act as addressees and coordinate the interaction, in order to minimize the heavy work load of the manager agent. As the manager and the delegates communicate locally, communication overhead of the manager is reduced significantly. Two types of delegate agents are created:

Client delegates: One is created for each connected client agent to handle the coordination between grid management and the client.

Worker delegates: One is created for each connected worker agent. Delegate agents plan and organize task assignment, task result collection and handle monitoring issues for their associated worker agent.

3.1.5 Tasks

In AGrid, a task is a set of computations designed to solve a certain problem. Tasks can be highly specialized and may require the target platform where they will be executed to carry certain attributes. During initial system registration, each worker agent provides information about its attributes, in the form of a description of its the computational features, to its delegate. The manager and delegate agent cooperate to select the proper worker agent for the task, according to the attributes requested. Actually, the manager agent consults worker delegates not only to match the requested features of a task with those of worker agents, but also to locate capable worker agents whose schedule is suitable to accept the task. The task runs its abstract "execute" method at the worker platform to which it is deployed. AGrid supports every kind of computational task which can be employed using the Java language. Tasks indicate the attributes they require by an abstract "properties" method. When the manager agent receives a task assignment request through a client delegate agent, it checks the requested attributes and matches the task with a suitable worker agent. After a worker agent is assigned a task/group of tasks, it executes each task via the execute method.

3.2 Protocols

Each agent in AGrid system must execute certain protocols during its lifetime in the grid. These protocols define the behaviour of the agent according to the role it carries in the system. This section introduces these protocols.

3.2.1 Agent Connection Protocols

JADE framework assigns an identifier number (AID – Agent Identifier) to each connected agent. Since the manager agent needs to keep the record of all the agents in the distributed environment, AID is inadequate for AGrid. The manager agent needs to know the type and the properties of each connected agent. Also, the manager agent has to decide whether to allow an agent to connect to the system or not. Therefore, a connection protocol for worker and client agents is defined. There is no need to for a connection protocol for delegate agents, since they are automatically created by the manager agent when any client or worker is connected to system.

There are some minor differences between the connection protocols of client and worker agents. Thus the protocols will be presented separately for each.

3.2.2 Connection Protocol for Client Agents

Client agents connect to the system through the following procedure:

- Client agent sends a "register message" to the manager agent.
- Manager checks if the client has already connected to the grid before. If the client is requesting to connect for the first time, the manager creates a client delegate agent for the client and pairs them. Otherwise, it runs the reconnection protocol.
- The client delegate agent sends an "accepted message" to the client agent which it represents.
- Client agent sends back an "info message" to its delegate agent. The body of this message contains client information.
- Consequently, the client agent is connected to system and is ready to issue tasks.

Client agents can temporally disconnect after transmitting a task execution request. The system allows clients to reconnect later and receive the results.

3.2.3 Connection Protocol for Worker Agents

Worker agents connect to the system with a very similar protocol to the client agents' protocol. However, the grid system does not allow the workers to reconnect. The details of the protocol are as follows:

- Worker sends a "register message" to the manager agent.
- Manager checks if the worker agent has connected to the grid before. If the worker is trying to connect for the first time, the manager creates a worker delegate agent for the worker and pairs them. Otherwise, a delegate already exists. Delegate sends "accepted message" to the worker.
- Worker sends an "info message" to the delegate agent.
- Thus, the worker agent connects to the grid system and is ready for task assignment and execution.

3.2.4 Task Assignment Protocol

Task assignment procedure and recovery from probable errors are vital issues for grid systems. During the design phase of AGrid, special care has been taken to develop an effective and flexible task assignment protocol in order to minimize runtime errors as much as possible. The details of the protocol are as the following.

- A client agent reports its identification information during the connection protocol. The "info message" which it sends to its delegate contains tasks which are to be processed in the grid.
- Each client delegate keeps track of tasks issued by its client agent. It maintains two task lists: a list that contains pending tasks and another that contains previously assigned and running tasks. Initially, all tasks requested to be executed in the info message are added to pending task list. Afterwards, tasks are moved to the running tasks list after they are assigned to worker agents.
- Client delegate agent checks the pending tasks list periodically and sends task assignment requests to the manager agent.
- The manager agent consults worker delegates in order to locate a free and suitable worker agent that meets the requirements of the task. Worker delegate accepts the request if the worker agent satisfies the task requirements and is currently available, as it is constantly informed about the status of the worker. Otherwise, the request is rejected.
- If the manager can locate an appropriate worker, it sends a "task execute request" message to the worker's delegate agent.
- The delegate agent forwards the request message to its worker agent.
- Worker agent accepts the task and begins to execute it.
- Worker delegate sends "confirm message" to the manager agent.
- Worker delegate moves the task from pending tasks list to running tasks list.

3.2.5 Monitoring Work Load of the Worker

One of the important parts of the task assignment is monitoring the work load of the machines where worker agents are located. Manager agent should assign tasks to non-busy workers. Each worker agent measures its work load periodically. If the state of the worker changes, it informs its delegate agent. In the initial versions of the system, the manager agent kept two separate lists for free and busy workers. However, as this approach caused too many synchronization problems, we decided to have each worker delegate to keep track of the workload state of its worker agent, in the final version.

3.2.6 Agent Disconnection Protocols

During the lifetime of the grid, client and worker agents may be in either connected or disconnected state. For a robust, stable grid system, the manager agent needs to be aware of the states in which client and worker agents are. It is clear that not noticing the disconnection of an agent has a negative effect on the stability of the grid system. It may result in the loss of some tasks or in the assignment of tasks to worker agents that no longer exist. Consequently, the grid system will have to run error recovery protocols. Therefore, one of the responsibilities of the manager agent is to detect disconnected agents. Hence, a flexible protocol for disconnection is defined on account of increasing the stability of the system.

As delegate agents are created and destroyed by the manager agent itself, there is no need for a disconnection protocol for them. Disconnection protocols for the worker and client agents are different. As stated in the connection protocol, while client agents can temporarily disconnect, a worker agent's disconnection is permanent. Agents periodically send an "alive message" to their delegates in order to indicate that they are still connected to the grid system. If a delegate agent does not receive this type of a message from its pair agent for a certain period of time, it decides that the participant agent has quit the system. In the normal case, a participant agent is expected to inform its delegate agent about its disconnection by sending an " disconnect message".

3.2.7 Disconnection Protocol for Client Agents

Two disconnection protocols are defined for client agents; one for temporary disconnection and one for permanent disconnection.

Temporary disconnection protocol is the following:

- A client delegate decides that the client agent has quit the system because either the client has sent an "disconnect message" or an "alive message" has not been received from the client for a certain period of time.
- Client delegate stops checking alive messages and waits for a "reconnect message from the client agent.

A client agent may decide to permanently leave the system before the computation it has requested completes. In this case the following protocol is executed:

- A client agent sends a "quit message" to its delegate agent.
- The delegate agent informs the manager agent about the disconnection of the client agent.

- The manager agent locates the delegate agents of the workers which are running tasks on behalf of the quitting client and informs them of the situation.
- Each worker delegate sends a message to its worker agent, requesting it to stop task execution.
- Each worker agent terminates its task execution thread and sends a "worker stopped" message to its delegate agent.
- After receiving "worker stopped" messages from every worker delegate, the manager agent destroys the client delegate agent and deletes information records about the client agent.

3.2.8 *Reconnection Protocol for Client Agents*

As stated before, a client agent may disconnect after submitting its tasks and, after a while, it may reconnect to the system in order to receive the results. Even though the reconnection protocol is very similar to the initial connection protocol, there are some major differences between them. In the reconnection protocol, the client agent does not send identification information as this information is already present in the system. The corresponding protocol is the following:

- Client agent sends "register message" to the manager agent
- The manager agent checks if this agent has connected to the system before.
- If the client has connected before, manager sends "accept reconnect message" to the delegate agent of the client.
- Client delegate agent sends a message to the client which indicates that its reconnection request has been accepted.
- Client delegate sends the results of the completed tasks.
- Client delegate starts to wait for "alive messages", that are periodically sent by the client agent in order to show that it is still connected to the system
- The client agent starts to wait for the result of its tasks.

4. TESTS AND ASSESSMENT

We have not yet carried out extensive experiments to observe and assess the performance of AGrid under various workloads and with varying number of participating workers. However, for a preliminary assessment, we have implemented a parallel matrix multiplication algorithm. Since multiplying an mXn matrix with an nXp matrix results in an mXp matrix, mXp tasks are created to compute the resultant matrix. We used 5X6 and 6X7 matrices for the test, which required a total number of 35 tasks. We ran several instances to observe the effect of the increasing number of clients and workers. The minimum, maximum and average calculation time were evaluated.

In the first test run, only one client was introduced and the number of the workers were varied between 1 to 100. As seen in the Figure 3, the working time decreases as the number of workers increases. After the number of the workers reaches nearly 40, working time stays constant, as expected.

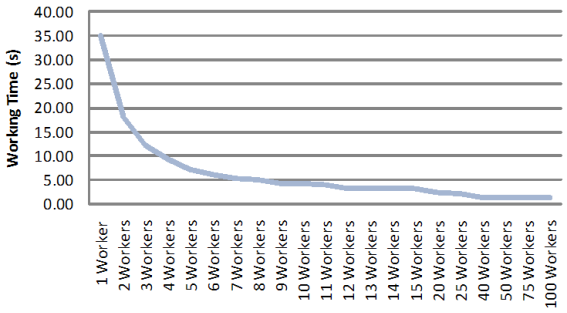

Figure 3: Results of first test.

In the second test run, the number of workers were kept constant while the number of the clients were increased. The test results for 15 workers is given in the Figure 4. As the number of clients increases, the number of the tasks also increases. As seen in the figure, total working time for the calculation increases as the number of independent tasks increases.

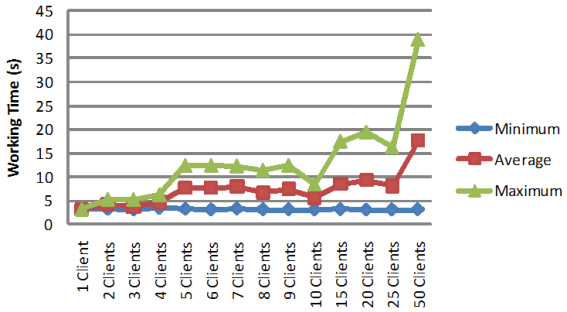

Figure 4: Results of second test.

5. CONCLUSIONS

AGrid is an agent based infrastructure for distributed and parallel computing. It combines the favorable aspects of grid computing and agent technology, producing a robust yet flexible execution environment. The system is implemented on JADE and is fully FIPA compliant. Agents with dedicated roles make up the framework. Client agents are users of AGrid that connect to the grid system to receive service. Worker agents connect to the system to share any of their resources. AGrid has a centralized control, with a manager agent in charge of general grid management. Delegate agents cooperate with the manager agent, reducing its workload significantly. Agent coordination and cooperation through well designed protocols has resulted in a robust, stable grid execution environment.

Even though experiments we have carried out to evaluate the performance of the system are not yet adequate, the results are promising. We have observed that system response time is within acceptable borders and the system is scalable, capable of serving large numbers of clients.

Currently, work is going on to enhance the system, to spot bottlenecks to optimize execution time. Our future work will

be on new protocols for dynamic load balancing to adapt to changing computation needs and changing computing resource environments. It is a fact that large amounts of data that accumulate on the manager agent may overload it, resulting in inefficient scheduling of tasks. Currently, work is continuing on a new version of task assignment policy, where market based algorithms are employed. Auctions are held to determine worker agents which can provide a requested service. These algorithms are carried out by specialized agents, thus decreasing the workload of the manager agent.

6. REFERENCES

[1] T. E. Athanaileas, N. D. Tselikas, G. V. Tsoulos, and D. I. Kaklamani. An agent-based framework for integrating mobility into grid services. In *MOBILWARE '08: Proceedings of the 1st international conference on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, pages 1–6, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[2] K. F. N. T. Bart Jacob, Michael Brown. *Introduction to Grid Computing*. IBM Corp., Riverton, NJ, USA, 2005.

[3] F. R. L. Cicerre, E. R. M. Madeira, and L. E. Buzato. Structured process execution middleware for grid computing: Research articles. *Concurr. Comput. : Pract. Exper.*, 18(6):581–594, 2006.

[4] T. T. G. R. Fabio Bellifemine, Giovanni Caire. *JADE Programmer's Guide*. Telecom Italia S.p.A., 2007.

[5] FIPA. *FIPA ACL Message Structure Specification*. Foundation for Intelligent Physical Agents, 2002.

[6] I. Foster, N. R. Jennings, and C. Kesselman. Brain meets brawn: Why grid and agents need each other. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 8–15, Washington, DC, USA, 2004. IEEE Computer Society.

[7] M. Fukuda and D. Smith. Uwagents: A mobile agent system optimized for grid computing. In *GCA*, pages 107–113, 2006.

[8] M. Li and M. Baker. *The grid core technologies*. John Wiley & Sons, 2005.

[9] A. Poggi, M. Tomaiuolo, and P. Turci. Extending jade for agent grid applications. In *WETICE '04: Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 352–357, Washington, DC, USA, 2004. IEEE Computer Society.

[10] O. W. Van, B. J. Overeinder, N. J. E. Wijngaards, M. V. Steen, and F. M. T. Brazier. Multi-agent support for internet-scale grid management. In *AISB'02 Symposium on AI and Grid Computing*, pages 18–22, 2002.

[11] M. Wooldridge. Agent-based software engineering. *Software Engineering. IEE Proceedings- [see also Software, IEE Proceedings]*, 144(1):26–37, 1997.