A Security Model for Mobile Agent in Grid Environment

K.MuthuManickam
Senior Lecturer
Department of Computer Science and Engineering
Arunai Engineering College
Tiruvannamalai- 606 60, Tamilnadu, India

ABSTRACT
Grid is a system that coordinates resources that are not subject to centralized control and provides environment for secure resource sharing through geographically distributed networks. Grid system lacks the autonomy and flexible behaviour. Mobile agents are autonomous programs that live and roam in the distributed environment, sense and act in the environment to achieve the goal. We propose a communication model that integrates both Grid system and Mobile Agent to perform parallel computations with secured communication in a complete heterogeneous and distributed environment. This framework is platform independent.

Categories and Subject Descriptors
[**IBM's Aglets**]: An *aglet* is a mobile agent. All aglets are derived from an abstract class called Aglet. Aglets use an event driven approach to mobile agents that is analogous to the Java library Applet class. Each aglet implements a set of event handler methods that define the aglets behaviour.

General Terms
Parallel computing, Distributed computing, Grid Computing, public key encryption.

Keywords
Grid Architecture, Resource Management, Mobile Agent, Parallel Virtual Machines.

1. INTRODUCTION
Both Grid system and mobile agent technology are common on the way that they both are being explored in a distributed environment. Both have their advantages and limitations when work as community or group. As a community, Grid has limitations such as Grid system can not anticipate and diagnose the changes to the state. In the same way Agent system has also limitations such as Agent framework does not provide support for secure interaction and use many assumptions. So there is a need to integrate both technologies to get true and full benefits of both technologies (Foster et al., 2004, FIPA, 2001).

2. GRID SYSTEMS AND PRALLEL VIRTUAL MACHINES
Grid computing is a kind of new technology which has been known since 1990s. The idea is to provide computational resources similarly to the way we get power supply. When you want power supply, you may connect your devices to the power grid; when you want computational resources, you may connect your device to the computational grid. The actual motivation behind computational grid was to solve complex scientific applications that require more resources under a single administration.

A Grid does not rely on high speed networks and is more available; they can be composed of computers spread around the world, interconnected by Internet. A number of Grid systems have been developed to monitor system changes to ensure the availability of the resource in the heterogeneous environment. The most common toolkit is Globus. This toolkit is more or less targeted towards the Linux platform.

The Globus Toolkit is a very rich set of tools to establish a Grid computing environment. It consists of information services, security, data management, execution management, fault detection, portability, communication, etc. The environment and the architecture of every organization are different. The Globus Toolkit was conceived to remove the obstacles that prevent seamless collaboration (Foster, 2006). Most of the Grid applications are implemented using Globus Toolkit, with no modification simply by linking with a Grid-enabled version of an appropriate programming library.

The concept of Grid has evolved from another similar paradigm called PVM (Parallel Virtual Machine). The difference between Grid and the PVM is that in Grid, the numbers of resources dynamically join the virtual network, while the number of computational resources in PVM is fixed (Mirza et al., 2003).

3. AGENT SYSTEMS
Mobile agents are also an emerging technology. A mobile agent is a program that can migrate from one computer to another for remote execution on behalf of a user on networks. The migrating agent is capable to carryout and roams along with its characteristics; execution state and program code and resume its execution on the destination site on arrival. Most common benefits of mobile agents are: reduce network load, overcome network latency, execute asynchronously, adapt dynamically, robust and fault tolerance, and heterogeneous (Danny. 1999).

The main problem in providing security to mobile agents is the fact that the execution environment does not belong to the user who has created the agent. As such after eh agent is launched, the user does not have any control over the agent. A security solution is therefore required which would guarantee the security of agents code and its personal data.

4. PROPOSED ARCHITECTURE

We propose a system where agent system will be explored on the top Grid systems that will provide security, autonomy, dynamic behaviour and robust infrastructure. The key features of the proposed Agent based Grid Architecture are:

- Resuming of tasks (by using software agents) after a CPU has returned back to its idle state. All the communication and the execution of tasks are handled by software agents.
- Providing security to agents personal (confidential) data. Support of task migration is provided by our architecture due to the introduction of agents. It handles fault tolerance by maintaining multiple copies of the task.

The architecture is actually a modification of Globus Toolkit where agents are introduced. In this way we reduced the communication overhead and provided support for task migration for resource utilization. To avoid operating systems dependency we selected Java as implementation tool. Java is feasible for both perspectives:

a) It is platform independence,
b) Most of the agent frameworks are developed in Java. Our recommendation is to use JAVA as the core programming language for implementing the Agent Based Grid Computing Framework.

Due to platform independence our modified framework enables various platforms to integrate in forming a Grid Computing environment.

4.1. Approach

We have recommended the use of Master-Slave design pattern for the development of the Agent based Grid Computing Framework. In our framework there is a single server and it is fixed. The server actually manages the grid, whereas the nodes are dynamic. It is the responsibility of the server to manage the entire grid. The master agent has the logic for the distribution of processes into tasks and recompilation of the results. This logic is provided by the user. The tasks are then sent to nodes with idle CPU cycles in order to utilize the free CPU cycles.

4.2. Programming Model

We propose a programming model that is a derivation of Master-Slave pattern. Master class has all the controlling logic. It divides the problem into small computable tasks which are called slaves. One Master class, but one or more slave files can exist. During the execution, the master thread will create slave threads, which will be executing the slave code. These threads will be mapped to the Agents. When a call to create a slave thread is made, a new task will be created and assigned to any available agent, which will execute it.

It is assumed that the platform where the Agents execute the task can eavesdrop on the agents data and communication hence confidentiality is required for both. It is also assumed that the platforms would not collude to compromise the data. An agent platform may support multiple locations or meeting places where agents can interact. Figure 1, which depicts the movement of an agent among several agent platforms.

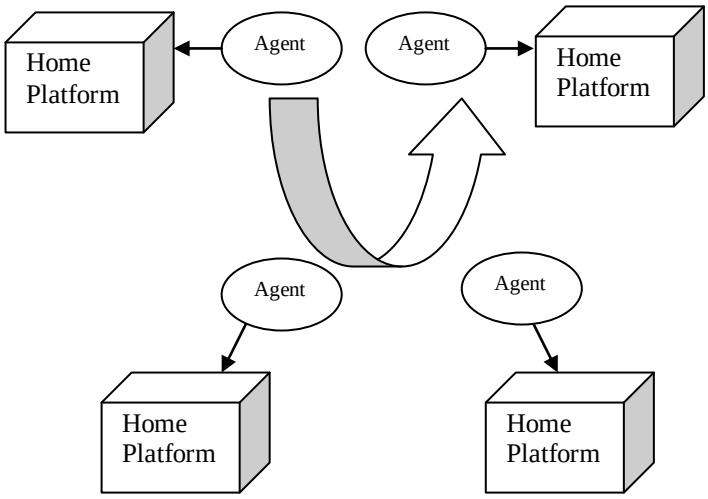

Fig. 1 Agent system Model

In conventional Client-Server systems the data is usually encrypted with the data owner's key to keep it confidential and when the data is needed in communication it is .first decrypted and then again encrypted using the public key of communication partner or the session key used during the communication. In an agent environment this is not an acceptable solution as the data is at one moment unencrypted and accessible by the host (untrusted host on which the agent resides). In our scheme, the data is first encrypted using the encryption key of the agent. At the moment data must be exchanged to another party, the data is again encrypted, but this time with the encryption key of the communicating partner. A decryption process then follows where the decryption key of the agent is used, such that the overall result is encrypted data, which can only be deciphered by the communicating party. This solution is referred as E-E-D. The process is depicted in figure 2 below:

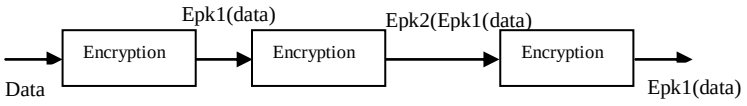

Fig. 2 Confidentiality in agent communication.

A necessary condition for an encryption algorithm to be used as E-E-D is:

$$DSK1 (EPK2 (EPK1 (M))) = EPK2 (M) \quad (1)$$

Where, PK1 and PK2 are the public keys of the agent and communicating party respectively. SK1 and SK2 are their corresponding private keys. It is assumed that there are more than one secret keys generated by the agent corresponding to different types of data. Initially the data to be encrypted is stored at the users computer and in order to encrypt it, the user .first generates a key pair for the agent depending on type of data. The user generates a large random prime p and a generator α of the multiplicative group Z* p of the integers modulo p. The user selects a random integer a1, 1<=a1<= p - 2, and computes :

$$y1 = \alpha a1 \bmod p \quad (2)$$

The agent's public key is (p, α, y1) and its private key is a1.
The user encrypts the data (represented by parameter m) as follows. He first selects a random integer k1, 1<=k1 <= p – 2 and computes:

$$\gamma 1 = \alpha k1 \bmod p;\ \delta 1 = my1\ k1 \bmod p \quad (3)$$

The cipher text is c1 = (γ1, δ1). This is stored in the agent and can be run on any platform at any host. At the moment that the agent needs to give the personal data to another entity in the system, the following process is started. The agent collects the communicating partner's (from here on called Bob) public key y2, which is formed in the same way as the user's public key (y2 = αa2 mod n). Bob's private key is a2. It must be noted here that in order to fulfill equation (1), Bob must use the same generator and prime number for generating its key pair as the user. The agent encrypts the cipher text c1 using Bob's public key y2, by the following computations:

$$\gamma 2 = \alpha k2 \bmod p;\ \delta 2 = \delta 1 y2\ k2 \bmod p \quad (4)$$

Where, k2, 1 <= k2 <= p - 2, is an integer chosen at random by Bob. The second cipher text c2 is then formed by the pair (δ2, γ1). It is now possible to decrypt it once using the agent's private key:

$$m' = (\gamma 1 - a1)\ \delta 2 \bmod p = my2\ k2 \bmod p \quad (5)$$

The result is an encryption of m based on PK2, e.g. y2. This is sent to Bob, who can decrypt according to the normal AlGamal decryption:

$$m = (\gamma 2 - a2)\ m' \bmod p \quad (6)$$

Decryption of m' (6) should occur at different place than the one where E-E-D operation took place as decryption of m' results in the plain text.

In order to adopt the above encryption scheme we propose a model for communication which confirms to agent system model.

5. FRAMEWORK COMPONENTS

Following are the major components of the Architecture. Each component will works as an agent.

a) *Process Manager*
The job of the process manager (agent) is to manage the user process. It will maintain a data structure similar to the PCB. Major responsibilities will include Process creation, Process termination, Task creation, Task termination, Process and Task Scheduling.

b) *Resource Manager*
This component (agent) will keep track of the agents available on Grid: thier location and status. It will also cater the reqeust for idle agent from task completed.

c) *Process Allocator*
The purpose of this component (agent) is to assign a task to an agent. It will take a task from Process Manager and an agent from Resource Manager, and then assign that agent the task. After assignment it will send the agent for execution.

d) *Agents Manager*
Agent Manager (agent) will be responsible for creating and destroying agents. It will also maintain an agent pool.

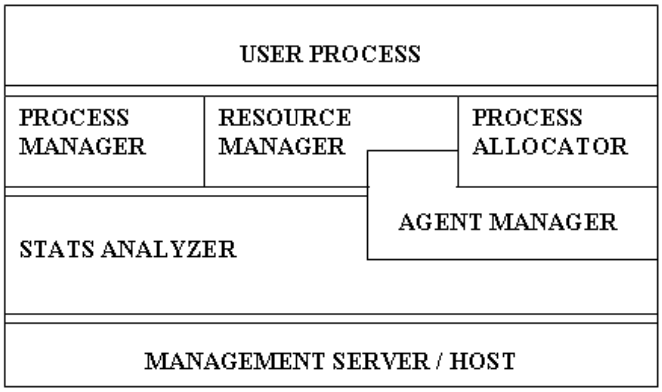

Fig. 3 Components of the Framework

e) *Statistical Analyser*
This component (agent) will interact with the other components and collect the statistic from them. Statistical analyser will also be responding to the queries regarding the statistics of the grid.

5.1 Application exploiting mobile agent

Matrix multiplication has been selected as a test case because it falls in the category of distributed and parallel applications. Two different approaches have been used to handle the problem. This application was also tested on ACENET to prove its support to distributed problem solving. It was Windows based agent platform and does not support LINUX.

For Small Matrices:

- Master agent analyses the matrices and partitions 1 matrix st into block of rows and 2nd into block of columns
- Master creates agents equal to the number of resultant matrix's cells and assign tasks to the agents
- Agents migrate to the idle nodes, or receives message, if already there
- Perform calculations there
- Bring result to the Master agent
- Master agent places the result at specific location in the resultant matrix

For Large Matrices:

- Master agent analyses the matrices
- Partitions 1st matrix into block of rows
- Master creates agents equal to the number of rows of the matrix one
- Broadcast in full the second matrix to the agents
- Assign tasks to the agents
- Agents migrate to the idle nodes, or receives message, if already there
- Each agent uses its block of the first matrix to multiply with the whole of the second matrix
- Place result to the specific location in the resultant

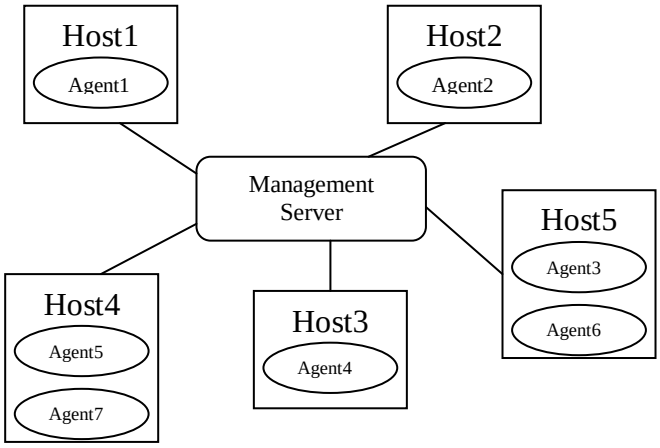

Fig. 4 Agents residing on hosts

Agents (Data agents) are residing on different hosts, which are part of grid environment. Data Agent Manager will keep tracks of all data agents.

5.2. Sample Application Exploiting IBM Aglets

We deployed Tahiti Server (implemented by IBM aglets) to demonstrate the idea of using agents to divide and solve computational problems. The following figures depict the creation of Agents which is showing the division of the Matrix multiplication.

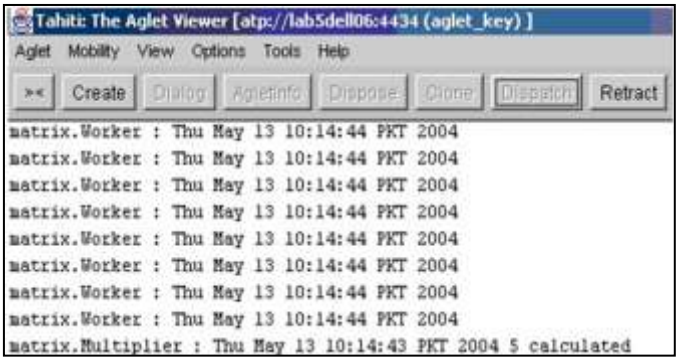

Fig. 5 Agents Creation on Tahiti server

Fig. 6 Retracting Agents on remote server

6. CONCLUSION

The key purpose of proposing this Agent based Grid architecture is three fold:

a. To make the Grid computing environment more efficient by exploiting software agents
b. To provide secured mobile agent communication
c. To exploit platform independence

In this paper we have proposed a cost-effective framework where software agents and Grid systems are integrated and work in a distributed and heterogeneous environment to solve parallel computing. To test the functionality of the framework we proposed an application of matrix multiplication. The main focus is on agents. We have explicitly tried to provide data privacy to an agent. So that the data possessed by the agents is secured at all times when it is executing at any of the untrusted hosts.

7. REFERENCES

[1] Ian Foster (2002). What is the Grid? A Three Point Checklist

[2] Luis Ferreira, Viktors Berstis, Jonathan Armstrong, Mike Kendzierski, Andreas Neukoetter, MasanobuTakagi, Richard Bing-Wo, Adeeb Amir, Ryo Murakawa, Olegario Hernandez, James Magowan and Norbert Bieberstein (2003). Introduction to Grid Computing with Globus

[3] Danny B. Lange and Mitsuru Oshima(1998) Programming and Deploying JAVA™ Mobile Agents with Aglets™

[4] Ian Foster and Carl Kesselman. Globus:A Metacomputing Infrastructure Toolkit

[5] Ali Ghulam, Shaikh Zubair Ahmed, and Shaikh Noor Ahmed, 2009. "The Design and Implementation of an Agent Framework to Support Distributed Problem Solving", published as proceedings European Computing Conference, Greece, September 2007, later published as Chapter (Mulyi-agent Systems) in Springer Verlag, 347-354.

[6] Czajkowski, K., I. Foster, N. Karonis and S. Tuecke, 1998. "A Resource Management Architecture for Meta computing Systems", Proceedings of the 4 International Workshop on Job Scheduling Strategies for Parallel Processing, Orlando, Florida, USA.

[7] Foster, I., 2006. "Globus Toolkit Version 4: Software for Service-Oriented Systems" IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779: 2-13.

[8] Karnik, N.M., A.R. Tripathi, 1998. "Design issues in mobile agent programming systems", IEEE Parallel & Distributed Technology, 6(3): 52-61.

[9] Mobile Agent Security, National Institute of Standards and Technology, Special Publication 800-19, August 1999.Wayne Jansen and Tom Karygiannis. http://csrc.nist.gov/publications/nistpubs/800-19/sp800- 19.pdf

[10] Count ermeasures for Mobile Agent Security, Computer Communications, Special Issue on Advanced Security

[11] Techn iques for Network Protection, Elsevier Science BV November 2000. Wayne Jansen. http://csrc.nist.gov/groups/SNS/mobile_security/documents /mobile_agents/ppcounterMeas.pdf

[12] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 31(4):469–72, 1985.