

## Integration of Grid and Agent Systems to Perform Parallel Computations in a Heterogeneous and Distributed Environment

<sup>1</sup>Ghulam Ali, <sup>1</sup>Noor Ahmed Shaikh, <sup>2</sup>Zubair A. Shaikh

<sup>1</sup>Department of Computer Science, Shah Abdul Latif University, Khairpur, Pakistan

<sup>2</sup>National University of Computer and Emerging Sciences, Karachi, Pakistan

---

**Abstract:** Grid is the infrastructure that provides environment for secure resource sharing and problem solving through geographically distributed networks. Grid system lacks the autonomy and flexible behavior. Mobile Agents are the autonomous programs that live and roam in the distributed environment, sense and act in the environment to achieve the goal. Agent systems lack the robust infrastructure. Most of the toolkits that exploit the Grid systems are operating system dependent. So there is need to integrate both technologies in a platform independent framework to get full benefits in distributed environment. This paper proposes an effective framework that integrates both Grid systems and Agent systems to perform parallel computations in complete heterogeneous and distributed environment where autonomous and intelligent applications can easily be build and deployed. It performs dynamic resource sharing and management. The framework is platform independent that works over Linux, UNIX and Windows operating systems.

**Key words:** Software Agents, Parallel and Distributed Computing, Resource Management, Grid Architecture.

---

### INTRODUCTION

Both technologies, i.e. Grid and Agents, are emerging and common in the way that they both are being explored in the distributed environment. Both have their advantages and limitations when work as community. As a community Grid has limitations such as Grid system can not anticipate and diagnose the changes to the state, Trust measuring after authentication of a user. In the same way Agent system has also limitations such as Agent framework does not provide support for secure interaction, Agent systems provide limited Knowledge base and use many assumptions. So there is a need to complete the deficiencies of both technologies so that both systems may become more efficient. Grid must have flexibility and decentralized decision making capabilities whereas Agent requires a robust distributed platform for scheduling and management of the tasks to be carried out. There is need to integrate both technologies to get true and full benefits of both technologies (Foster *et al.*, 2004; FIPA, 2001).

#### 2. Grid systems and Parallel Virtual Machines:

The term Grid is taken as analogy to the Electric power grid that provides easy, reliable & transparent access to electricity, irrespective of its source. It facilitates in such a way that just plug any compatible device and enjoy its benefits (Oram, 2001). Keeping in view the reliability, easy accessibility and transparency of the Electric power grid, the computer scientists started to work on the development of the similar design called Computational Power Grid. The actual motivation behind computational grid was to solve complex scientific applications that require more resources under a single administration. A Grid facilitates to select, share and aggregate the physically distributed resources to solve large scale complex problems. Grid handles and manages all services such as security, resource sharing and management, scheduling, quality of services, etc.

A Grid can be viewed as an integrated computational and collaborative where user interact with the Grid application (at high level) to solve the problems through underlying distributed Grid resources that handle resource discovery, scheduling, and processing tasks (Foster and Kesselman, 1999).

To ensure the availability of the resource in the heterogeneous environment, there is need to continuous monitor the system changes. A number of Grid systems have been developed to resolve this kind of problems.

---

**Corresponding Author:** Ghulam Ali, Department of Computer Science, Shah Abdul Latif University, Khairpur, Pakistan  
E-mail: ghulam.ali@salu.edu.pk

The most common toolkit is Globus (Czajkowski *et al.*, 1998). Globus Toolkit is a technology that facilitates peoples to share computing power, databases, etc. The toolkit consists of information infrastructure, security software, management of data and resources at various levels, fault detection, portability, communication, etc. The environment and the architecture of every organization is different. The Globus Toolkit was conceived to remove 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. Globus is supported by UNIX, Linux while Microsoft Windows support is under development.

The concept of PVM (Parallel Virtual Machine) is also used to enhance computation power. The difference between Grid and the PVM is that in Grid the number of participating nodes is not fixed and resources dynamically join the virtual network, while the number of computational resources in PVM is fixed (Mirza *et al.*, 2003).

### **3. Agent Systems:**

“Software agents are computational systems that live in inhabit some complex dynamic environment; sense and act autonomously in that environment to achieve the task for which they are designed” (Maes, 1990). Where as a mobile agent works on behalf of a user on networks and is skilled of performing its duties and assigned tasks autonomously from host to host (Karnik and Tripathi, 1998; Shaikh and Mallah, 2005). The migrating agent is capable to carryout and roams along with its characteristics; execution state and program code as it may be legitimated and resume its execution on the destination site on arrival (Piszcz, 1998).

Mobile agents are very much beneficial in the networked environment. Most common benefits are: Reduce the network load, Overcome network latency, Encapsulate protocols, Execute asynchronously and autonomously, adapt dynamically, robust and fault tolerance, and heterogeneous (Danny, 1999).

### **4. Proposed Architecture:**

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

- a) Resuming of tasks (by using software agents) after a CPU has returned back to its idle state.
- b) All the communication and the execution of tasks are handled by software agents.
- c) 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 Methodology:**

We have recommended the usage of Master-Slave design pattern for the development of the Agent Based Grid Computing Framework. Server actually manages the grid, whereas the nodes are dynamic. The Master agent has the logic for the distribution of processes into tasks and re-compilation of the results. 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. The slave code will report back to the master after finishing the task assigned to it. The messaging is supported through the agents framework.

To cater the resume support for the task, during task migration, as agents move from one host to another, some constructs will also be available, to the programmer. JNI (JAVA Native Interface) and API (Application Programming Interface) are two possible solutions. We opted API for resuming the state of a task.

This is just a pseudo code to demonstrate the usage of the framework and its support for resuming the task support.

#### **4.3 Framework Components:**

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

##### **i. 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.

##### **ii. Resource Manager:**

This component (agent) will keep track of the agents available on Grid: their location and status. It will also cater the request for idle agent from task completed.

##### **iii. 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.

##### **iv. Agents Manager:**

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

##### **v. Statistical Analyser:**

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

#### **5. Application Exploiting IBM Aglets:**

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 (Ali Ghulam, 2007; Ali *et al.*, 2007).

##### **For Small Matrices:**

- Master agent analyses the matrices and partitions 1<sup>st</sup> matrix into block of rows and 2<sup>nd</sup> 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

Fig.5 and fig.6 show that 12 agents have been created and how two different agents, Agent[AB(1,2)] and Agent[AB(3,3)], are performing calculations and place result in the resultant matrix.

##### **For Large Matrices:**

- Master agent analyses the matrices
- Partitions 1<sup>st</sup> 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

The proposed algorithm for master logic:

```
Master {
void Initialize () {
//do initialization here.
}
void CreateSlave(){
for i=0;i<n;i++
{
//CreateGridThread ( Slave)
//start the thread.
}
}
void ReceiveMessage(Message msg){
//compile the result from the msg
}
}
void Main (void){
//read in Input file
Initialize();
CreateSlave();
//waitWhileDone;
//output the result on OUTPUT file
}
```

Fig. 1: Master logic algorithm

The proposed slave algorithm will be

```
Slave{
Void DoTask()
{
//performs computation

//when done send message to
master.
}
```

Fig. 2: Slave logic algorithm

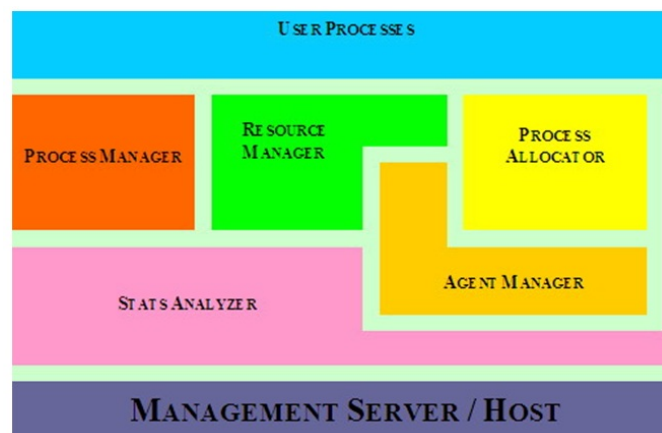


Fig. 3: Components of the Proposed Framework

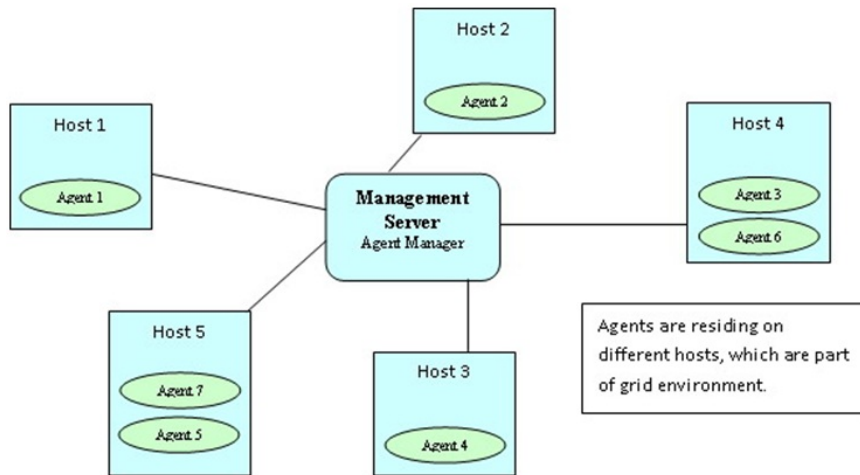


Fig. 4: Agents residing on hosts

$$(AB)_{1,2} = \sum_{r=1}^2 a_{1,r}b_{r,2} = a_{1,1}b_{1,2} + a_{1,2}b_{2,2}$$

$$(AB)_{3,3} = \sum_{r=1}^2 a_{3,r}b_{r,3} = a_{3,1}b_{1,3} + a_{3,2}b_{2,3}$$

Fig. 5: Task of agents to perform calculations in Matrix Multiplications

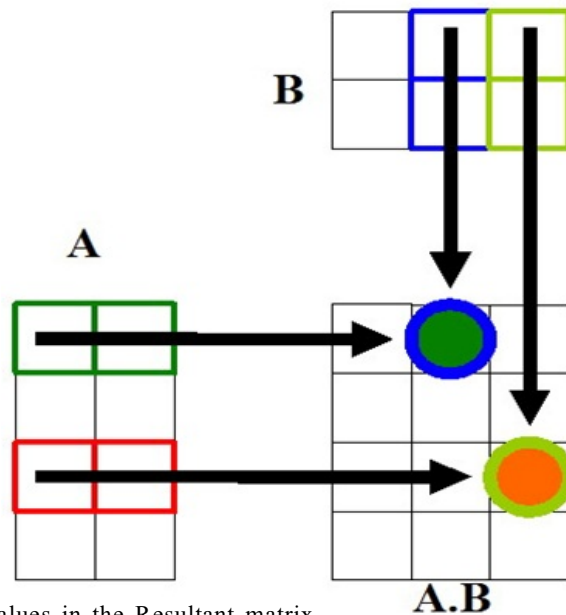
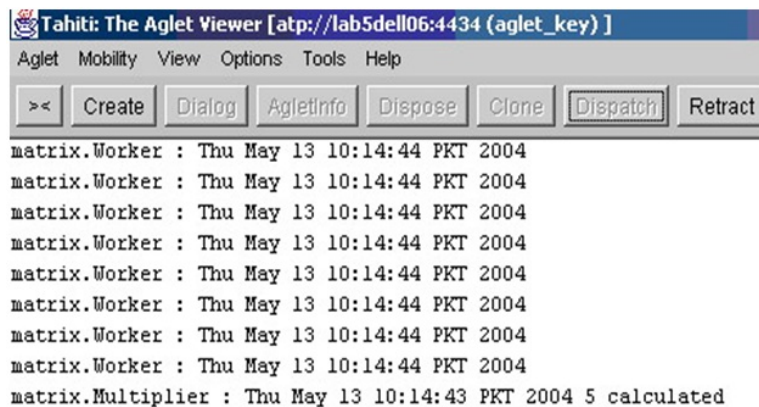


Fig. 6: Agents placing values in the Resultant matrix

We deployed Tahiti Server (implemented by IBM Aglets) that supports mobility, to demonstrate the idea of using agents to divide and solve computational problems (Danny, 1999). The following figures depict the creation of Agents which is showing the division of the Matrix multiplication.

**6. Conclusion and Discussions:**

The key purpose for proposing this Agent based Grid Architecture is twofold, i.e. a) To make the Grid Computing environment more efficient by exploiting Software Agents and b) To exploit platform independence.



**Fig. 7:** Agents' creation on "Tahiti" server

We developed a Cost-effective framework where Software Agents and Grid Systems are integrated and work in a distributed and heterogeneous environment to solve parallel computing. It supports resource management and sharing. The framework runs over all operating systems. The framework's all components are handled by agents. To test the functionality of the framework we implemented an application of Matrix Multiplication. Java based IBM's Aglets Agent Toolkit was exploited for Agents migration and security. We have shown the results that show the success of the framework.

The framework has provided a test bed for future ideas in distributed and parallel computing. The main focus during research was on agents. It can be tested and even extended for more complex computations where large Grid systems may be installed and agents may run over it.

Furthermore, in the applications like Matrix Multiplications, order determines the problem complexity as it affects the problem size. When matrix order will be increased, the problem size will also be increased. Hence the problem complexity increases. The performance against problem size should be measured. Result must be taken, analyzed and compared when more computing nodes are attached to the Grid.

## REFERENCES

- Ali, G., Z.A. Shaikh, M.J. Nisar, 2007. "A Decentralized and Fault-tolerant FIPA Compliant Agent Framework based on .NET", presented in UITCS ACM TECHNICAL CONFERENCE NOV – 2007 at Usman Institute of Technology, Karachi on, 23-24.
- 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.
- Czajkowski, K., I. Foster, N. Karonis and S. Tuecke, 1998. "A Resource Management Architecture for Meta computing Systems", Proceedings of the 4<sup>th</sup> International Workshop on Job Scheduling Strategies for Parallel Processing, Orlando, Florida, USA.
- Danny, B., 1999. Lange, Mitsuru Oshima, "Seven Good Reasons for Mobile Agents", Communications of the ACM, 42(3): 88-89.
- Danny, B., 1999. Lange and Mitsuru Oshima, "Programming and Deploying JAVA™ Mobile Agents with Aglets™".
- FIPA Interaction Protocols Specifications, Available at FIPA's official website <http://www.fipa.org>.
- Foster, I., N.R. Jennings, C. Kesselman, 2004. "Brain meets brawn: why grid and agents need each other", Autonomous Agents and Multiagent Systems (AAMAS): 8-15.
- Foster, I. and C. Kesselman, 1999. (editors), "The Grid: Blueprint for a Future Computing Infrastructure", Morgan Kaufmann Publishers, USA.
- 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.
- Karnik, N.M., A.R. Tripathi, 1998. "Design issues in mobile agent programming systems", IEEE Parallel & Distributed Technology, 6(3): 52-61.

Mirza, M.U., Z.A. Asim-ur-Rehman Shaikh, O.A. Khan, 2003. "A parallel distributed environment for Pakistan", 7<sup>th</sup> International Multi Topic Conference, 2003. INMIC – 391- 395.

Maes, Pattie, 1990. "Designing Autonomous Agents", Cambridge, MA: MIT Press, 1990.

Oram, A., 2001. (editor), "*Peer-to-Peer: Harnessing the Power of Disruptive Technologies*", O'Reilly Press, USA.

Piszc, Alan, 1998. "A Brief Overview of Software Agent Technology", The MITRE Corporation, McLean, VA 22102.

Shaikh, Z.A., G.A. Mallah, 2005. "A Platform Independent Approach for Mobile Agents to Monitor Network Vulnerabilities", WSEAS Transactions on Computers ISSN 1109-2750, published in.