Achieving Autonomy Using Alchemi Middleware In Grid Environment
Sanam Kadge\textsuperscript{a}, Gresha Bhatia\textsuperscript{b}

\textsuperscript{a}Student (M.E.)
\textsuperscript{b}Assistant Professor Department of Computer Engineering V.E.S.I.T, Chembur, Mumbai.
\textsuperscript{c}sanam.kadge19@gmail.com

Abstract: Microsoft’s .NET Framework has replaced the Unix systems as the ideal platform for implementing commercial distributed systems for Windows-based platforms for grid computing. Grid computing scales from an enterprise/organization to a global level. The introduction of Grid computing has made distributed systems more complex and unmanageable because of the manual system adopted for the management. The node instability, variation of resource availability, load distribution, heterogeneity of devices and network makes the grid management and configuration error prone and difficult to be performed by human beings. Due to dynamic nature of grid the middleware’s are lacking complete autonomic features. After doing comparative study, we found that Alchemi is more close to autonomy. In this paper we present Alchemi middleware that uses a .NET based framework, provides the runtime machinery and programming environment required to construct enterprise/desktop grids and execute grid applications. We implemented a small desktop grid involving two nodes to study its autonomic behavior. Also since the grid environment needs agent technology, we have tried to understand and map the agent framework integrated into Alchemi middleware. We have discussed the detailed experiment procedure and result analysis.

Keywords: Grid computing, Alchemi, Agent, Autonomy, Manager, Executor.

1 INTRODUCTION

Grid systems require autonomous, flexible behaviors. The dynamic nature of the grid infrastructure, its high scalability and great heterogeneity have turned impracticable its configuration, maintenance and recovery in case of failures solely by human beings. Several recent research projects have recognized the necessity of providing a greater autonomy to grid systems, which is one of the greatest challenges for the new generation of this kind of middleware. [2] This leads to new paradigm of self management through autonomic computing to pervade over the old manual system to begin the next generation of Grid computing.

Self-management of resources can make the system up and running at full pace most of time. Autonomic computing provides such an approach by enabling the design and development of systems/applications that can adapt themselves to meet the requirements without manual intervention. In other terms, grid computing needs to be autonomic Grid middleware provides computing ability and uniform access to resources in heterogeneous environment. A system that is to be classified as an autonomic system should have the following major features [8].

- Self-Protecting: Incorporation of intelligence to recognize and circumvent security threats
- Self-Configuring: Functionality that allows the addition and removal of components or resources without service disruption
- Self-Healing: Capacity to recognize and diagnose deviations from normal conditions and take action to normalize them.
- Self-Optimizing: Ability of the system to monitor its state and performance and proactively tune itself to respond to environmental stimuli

All the major grid middlewares (Globus, condor etc) lacks complete autonomic behavior and things need to be done manually. Alchemi is powerful and flexible grid middleware used to develop desktop grid and enterprise grids. Alchemi grid has three types of components namely the Manager, the Executor, and the user application

Grid environment focus on the infrastructure, tools and applications for reliable and secure resource sharing within dynamic and geographically distributed virtual organizations. In contrast, the agent’s community has focused on autonomous problem solvers that can act flexibly in
uncertain and dynamic environments, we see a convergence of interests, with agent systems requiring robust infrastructure as seen in the figure 1.

Figure 1. Agents supporting Grid Environment

2 AGENT FRAMEWORK

Agent environment is the kernel of Grid computing which is responsible for resources location and allocation, authentication, unified information access, communication, task assignment, agent library and others. Agent based technologies have become more and more widespread during the last few years and the number of practical applications has noticeable increased, especially in the new Grid Computing paradigm. Agents and Grid both need each other. Grid computing can be seen as a multi-agent system facilitating the sharing of compute resources, allowing users to discover and use remote resources. Users are able to submit jobs to remote resources and typically have no explicit control over the resources themselves. [6] Thus, both users and resources can be viewed as autonomous agents, having control of their own behavior. This autonomy gives rise to inherent uncertainty, since an individual cannot predict how another will respond to changing situations.

3 PROPOSED SYSTEM

In this paper we implement a desktop grid using Alchemi middleware and execute the application using .Net framework. A grid is created by installing Executors each machine that is to be part of the grid and linking them to a central Manager component. We execute the application developed in a .Net framework when the grid is established. Thus by implementing the architecture of Alchemi we can observe and study its autonomic behaviour. Also the grid implementation using Alchemi helps us to implement agent framework and map the different agents with Alchemi middleware. The run time components of Alchemi middleware uses .Net framework for execution of grid applications. The different nodes are created as Executors which are connected to a central Manager node. The user application created has API, console interface, Xml representation of data which interact with .Net objects created for the application.(Request object, Response object etc) as shown in figure below
4 IMPLEMENTATION

4.1 Configuration of Manager and Executor nodes
We configure the manager and executor set up through installer interface. In this step we have to provide the installer of manager and executor node with port numbers of each, password for authentication, mode information (Dedicated or non-dedicated).

4.2 Start the Manager node
After providing the details we run the installer for Manager on port no 9000. The necessary tasks are performed by the middleware and the Manager node is started.

4.3 Connecting Executor to Manager to form grid
After starting the manager node we connect the Executor node to the Manager node using the Connect option. Now our Grid nodes are connected to each other and Grid environment is enabled.

4.4 Successful execution of the code or application
We use visual studio 2008 and execute our code written using c# for grid applications. Alchemi.Core.dll is a class library. It is located in the bin directory. It must be referenced from by all your grid applications.

4.5 Disconnect the grid connections and execute the code to test the behavior of Alchemi
In case we disconnect our Executor from Manager node or try to stop Manager node then the connection is broken and grid application gives socket error since grid environment no longer exist. Alchemi notifies connection problems to the user. The manager logs its output and errors to a log file called “alchemi-manager.log.”

5 OBSERVATION SCREENS
The following figure 3 shows that the program runs successfully in the grid environment and displays the following result.
When the grid connection between the manager and executor longer is broken and the program is executed, we get error as shown in figure 4. The manager node cannot connect to the executor node and hence gives the user the following message.

Figure 3. Successful execution of an application when grid environment is enabled.

Figure 4. Socket Error when Manager is stopped or when executor is disconnected

6 RESULT ANALYSIS

6.1 Alchemi middleware supports the following Autonomic Characteristics

- **Self Configuration**: It has interactive windows installer available for Manager node and Executors for making the setup automated. We have seen in the above figure that we can easily provide different details and get easily connected to the grid. We have separate window installer for each Executor node.

- **Self Protection**: Security is due to powerful toolset of .NET framework. Authentication is also supported by use of password. Alchemi uses role based security to authenticate users and authorize execution. The authentication and Authorization is done using X.509 proxy certificate, which is sent to the server side Client is allowed to access various applications provided by the Alchemi .NET-based grid computing framework with single sign-on and full delegation rights.

- **Self Healing**: Poor HBM technique, no efficient mechanism. Since the grid has single manager, its failure can disrupt the services the system may fail.

- **Self Optimization**: Alchemi uses FCFS approach to execute the job. Manager provides different roles to clients (Executors) and delegates the responsibilities to them as per requirement and availability of the resources. A priority for each thread can be explicitly specified when it is created or submitted. Threads are scheduled on a Priority and First Come First Served (FCFS) basis, in that order.

6.2 The following agents are integrated with Alchemi middleware.

- **Manager**: The Manager provides services associated with managing execution of grid applications and their constituent threads. Executors register themselves with the Manager, which in turn monitors their status. Threads received from the User are placed in a pool and scheduled to be executed on the various available Executors.
Executor: The Executor accepts threads from the Manager and executes them. The Executors return completed threads to the Manager which are subsequently collected by the respective users. It performs the task like flexible resource management, flexible deployment under network constraint. It acts as a processor.

User: The API abstracts the implementation of the grid from the user and is responsible for performing a variety of services on the user’s behalf.

The Cross-Platform Manager is a web services interface that exposes a portion of the functionality of the Manager in order to enable Alchemi to manage the execution of grid jobs. Jobs submitted to the Cross-Platform Manager are translated into a form that is accepted by the Manager.

The communication agent: It provides an interface port for Manager and Executor to communicate. Communication between different nodes takes place with FTP protocol.

7 CONCLUSION

Through the implementation of grid environment we discovered that Alchemi allows the user to aggregate the computing power of networked machines into a virtual supercomputer. It provides the ease of deployment and management, interoperability with other grid middleware. Alchemi autonomic features (self-Protecting, self-configuration, self-healing and self-optimization) are evaluated using experiments that clearly indicate that this Adaptive approaches contribute to the grid infrastructure. Agent environment is the kernel of Grid computing which is responsible for resources location and allocation, authentication, unified information access, communication, task assignment, agent library and others. Agents play very important role in grid environment. Grid systems need agent technology and also agent framework supports grid environment as we have seen. Agent environment provides intelligence to Alchemi middleware thus helping it to exhibit autonomic behavior which is becoming necessity of grid systems day by day. Due to its Master Slave configuration healing is poor with Alchemi; future work can be done in this area.

REFERENCES


[6] Ian Foster, Nicholas R. Jennings, Carl Kesselman,” “Brain Meets Brawn: Why Grid and Agents Need Each Other”.


