GRID RESOURCE MANAGEMENT (GRM) BY ADOPTING SOFTWARE AGENTS (SA)

T.A.Rama Raju[1], Dr.M.S.Prasada Babu[2]
[1]*Statistician and Researcher JNTUK, Kakinada (India),*
[2]*Professor CS&SE, College of Engineering, Andhra University, Waltair(India).*

ABSTRACT

Grids and Agent communities offer two approaches to open distributed systems. Grids focus on robust infrastructure and services, and Agents on the operation of autonomous intelligent problem solvers. The interests of Grids and Agents converge when efficient management of the services provided by Grids comes to the fore. Cooperation, coordination and negotiation are issues that Grid users and Grid resource providers need to address in order to successfully interoperate. We propose a Grid service platform that dynamically provisions resources for both interactive and batch applications to meet their QoS constraints while ensuring good resource mutualization by adopting software agents (SA). Moreover we believe that relying on software agents for grid resource management (GRM) will allow more flexible, robust and scalable solution.

Keywords: ***Grid, Software Agents, Resource Management, Distributed Systems.***

I. INTRODUCTION

In spite of the success of Grid computing in providing solutions for a number of large-scale science and engineering problems, the problem of GRM remains an open challenge. GRM problems include the management of user requests, their matching with suitable resources through service discovery, and the scheduling of tasks on the matched resources. A high level of complexity arises from the various issues GRM must address: resource heterogeneity, site autonomy and security mechanisms, the need to negotiate between resource users and resource providers, the handling of the addition of new policies, scalability, fault tolerance and QoS, just to cite the most relevant.

Software agents are emerging as a solution for providing flexible, autonomous and intelligent software components. Multiagent Systems (MAS) are collections of software agents that interact through cooperation, coordination or negotiation in order to reach individual and common goals. There are five main dimensions that software computing is presently trying to address and that are shared by the Agent perspective:

- ubiquity

- interconnection
- intelligence (and learning)
- delegation (autonomic computing)
- human orientation.

Agents and MAS meet these requirements, and many computer scientists consider Agent-oriented programming to be the next paradigm in software computing. Grid computing pioneers, as the developers of the 'de facto' Grid middleware Globus Toolkit, agree with agent experts on the advantages that the convergence of the two approaches may bring (see their recent paper: I. Foster, N. R. Jennings and C. Kesselman, (2004). Brain meets brawn: Why Grid and agents need each other. Proc. 3rd Int. Conf. on Autonomous Agents and Multi-Agent Systems, New York, USA)

Software systems have been influencing and controlling all aspects of our daily life. This implies that society has developed an extraordinary addiction on computers. software agents are critically needed to meet the dynamic changes in information technology, which inhabits waste amounts of discrete information that require complex processing under uncertain conditions in order to abstract knowledge and make decisions.

This article presents an overview and deliberation of the most important aspects of the rapidly evolving technology of software agents in order to setting up clear distinctions between many different principals upon which we study and develop agents and multi agent systems.

A software agent is a software program that acts for a user or other program in a relationship of agency, which drives from the Latin agere: an agreement to act on one's behalf. Search "action on behalf of" implies the authority to decide which, if any, action is appropriate.

Attributes of Software Agents

The basic attributes of software agents are that

- Agents are not strictly invoked for a task, but active themselves,
- Agents may reside in wait status on a host, perceiving context,
- Agents may get to run status on a host upon starting conditions,
- Agents do not require interaction of user,
- Agents may invoke other tasks including communication.

The term **"agent"** describes a software abstraction, an idea or a concept, similar to OOP terms such as methods, functions and objects. The concept of an agent provides a convenient and powerful way to describe a complex software entity that is capable of acting with a certain degree of autonomy in order to accomplish tasks on behalf of

its host. But unlike objects, which are defined in terms of methods and attributes, an agent is defined in terms of its behavior.

Agents; those commonly include concepts such as

- Persistence (code is not executed on demand but runs continuously and decides for itself when it should perform some activity)
- Autonomy (agents have capabilities of task selection, prioritization, goal-directed behavior, decision making without human intervention)
- Social ability (agents are able to engage other components through some sort of communication and coordination, they may collaborate on a task)
- Reactivity (agents perceive the context in which they operate and react to it appropriately).
- Learning ability (when designing an agent they developer may furnish it with all the intelligence needed to carry out its assigned roles to achieve specific goals).
- Reasoning (Is a decision-making mechanism, by which an agent decides to act on the basis of the information it receives, and in accordance with its own objectives to achieve its goals.

What makes it an Agent?

An Agent should at least be characterized by possessing the autonomy and pro-activity properties.

Explicitly, to be an agent, it has to work independently without any external intervention, and once initiated, it must plan ahead and take the initiative instead of waiting for external instructions.

Classification of Agents

Due to the multiplicity of properties that agents can possess. As we;; as the tasks they can perform.

Different types of classes of agents. For instance

- Navigation Agents
- Search Agents
- Search and Retrieval Agents
- Management Agents
- Report Agents
- Testing Agents, Development Agents and so on.

Multi-Agent Systems (MAS)

A Multi-Agent System can be viewed as a distributed system that coherently incorporates various discrete and independent problem solving agents.

MAS as a loosely coupled network of software agents that interact to solve problems that is beyond the individual capacities of each problem solver.

In MAS, we focus on the coordinated behavior of a system of individual autonomous agents to work cooperatively by interacting with each other and with their environment, which is usually heterogeneous in nature.

Advantages of MAS

Tsatsoulis and Soh [19] assert that MAS usually inherit the following major advantages:

1. Reliability: MAS are more fault-tolerant and robust.
2. Modularity and scalability: instead of adding new capacities to a monolithic system, agents can be added and deleted without breaking or interrupting the system.
3. Adaptivity: agents have the ability to re-configure themselves to accommodate new changes and faults.
4. Concurrency: agents are capable of reasoning and performing tasks in parallel which in turn provides more flexibility and speeds up computation.
5. Dynamics: agents can dynamically collaborate to share their resources and solve problems.

In addition, Stone and Veloso [11] claim that parallelism is another important advantage in MAS to overcome the limitations imposed by time-bounded reasoning requirements. They also claim the advantage of robustness in MAS that have redundant agents; for instance, "if control and responsibilities are sufficiently shared among different agents, the system can tolerate failures by one or more of the agents." They assume that this feature is absolutely essential for domains that degrade gracefully; for example, "if a single entity, processor or agent, controls everything, then the entire system could crash if there is a single failure." Furthermore, they comment that "although a multi agent system need not be implemented on multiple processors, to provide full robustness against failure, its agents should be distributed across several machines."

Applications of Multi-Agent Systems

Multi-Agent Systems are appropriate in such situations when we have complex designs that incorporate numerous components and remotely distributed agents with different expertise and conflicting interests. Aylett et al. [13] assume that the MAS approach is worth considering for problems that are inherently distributed, either physically or geographically, such that independent processes can be distinctly identified. Examples of such problems include distributed sensor networks, air traffic control systems, decision support systems, and other networked/distributed control systems. They also claim that distribution is not a sufficient factor to use MAS, but there should also be essential requirements for intelligence or adaptivity in the sub-processes that involve explicit reasoning about

behavior. In addition, they recognized the importance of information ownership to existing agents, especially in such situations where information is distributed over different agents and no single agent can access all the information.

Moreover, it would be a trivial solution and insignificant decision to use MAS because we just have complex requirements; not all complex designs require MAS. We usually adopt multi-agent systems when dealing with such situations where different clients/organizations with potentially-conflicting goals and unincorporated information need to interact with each other to effectively achieve their targets and perform their tasks. Furthermore, it should be emphasized that in some situations where a centralized control is essential to a single agent or no parallel actions are possible, we cannot employ the MAS approach.

In this sense, Aylett et al. have demonstrated a number of situations that qualify to adopt MAS, include:

1. Applications that require "inter-connections and inter-operation of multiple autonomous, self-interested existing legacy systems, expert systems, and decision systems".
2. Problems involving distributed autonomous and selfish information sources.
3. Problems requiring solutions based upon "different distributed experts, such as health care provisioning, in which some central agent cannot possibly perform the task without help from other experts".
4. Problems characterized by having "cross organizational boundaries for which an understanding of the interactions among societies and organizations is needed".
5. Problems characterized by a lack of global view for all agents, but having several agents with local views.

II PROPOSED WORK GRID AND AGENT TECHNOLOGIES

We are currently looking at the intermingling of Grid and Agent technologies. We propose an environment based on an agent marketplace, in which agents representing Grid users can detect and meet agents representing Grid services, negotiate with them for the conditions of the service, and execute the tasks on the Grid node corresponding to the service (see Figure 1). Agents allow flexible negotiation, exchanging messages on behalf of the entities they represent. In our system, agents can use an extra information source for negotiations, consisting of a table containing the possible grid configurations ranked by estimated utility. For each possible configuration, this utility table stores the expected time of task completion, given values for parameters used to model the state of the resources (eg available resources, system load, network traffic) and for execution parameters of the task (eg granularity and time between each submission of a single job). The Broker Agent (BA) representing the Grid user can check this utility table to decide which of the bids proposed by Seller Agents (SAs) is the best, and also decide how to prepare the task for the effective execution in order to minimize the time to completion.

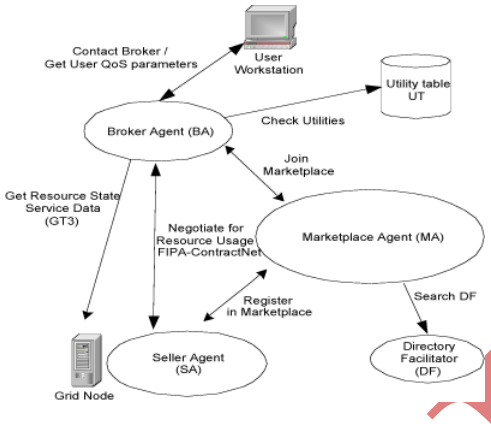

Figure 1: The Overall Architecture.

The negotiation between agents for usage of resources is based on the FIPA-Contract-Net protocol, a common protocol used to dynamically create relationships in response to current processing requirements embodied in a contract. The contract is an agreement between a manager (the Broker) and a contractor (the Seller Agent acting on behalf of a Grid resource), resulting from the contractor successfully bidding for the contract.

The agents are implemented using JADE Framework supporting FIPA standard for agent architectures, and they behave in a coordinated manner by means of the sequential execution of their behaviours (see Figure 2). We use Globus Toolkit 3 as Grid middleware for the interaction between agents and the Grid services and for the effective scheduling of jobs into the available resources.

Figure 2: The Agents operation Sequence Diagram.

Agent Based Grid Computing System

The resource management is central to the operation of a grid. The basic function of resource management is to accept requests for resources from machines within the grid and assign specific machine resources to a request from the overall pool of grid resources for which the user has access permission. A resource management system matches requests to resources, schedules the matched resources, and executes the requests using the scheduled resources.

Resource Management

The basic issues related to meta-computing resource management include the representation and management of knowledge that models data semantics, communication protocols, resource discovery and quality of service (QoS) support [17]. The main issues related to local resource management are multi-processor scheduling, resource allocation and monitoring. Next, we will give a detailed description on how some of these issues are settled in AGCS.AGCS is composed of Local Grid systems. Every Local Grid system is implemented as a Multi-Agent Environment (MAE) multi agent system or other FIFA compliant multi-agent systems, it mainly consists of the following components: RA (Resorce Agent), AMS(Agent Management System), DF(Directory Facilitator), and LGS(Local Grid System).As summarized in [16], there are three different models for grid resource management architecture: hierarchical model, abstract owner model and, computational market/economy model.

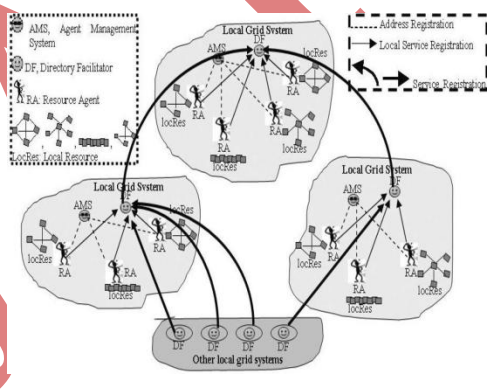

Figure 3: Resource Management Architecture

III. CONCLUSIONS

Establishing grids is an important undertaking in developing scalable infrastructures. Agent-Based Grid computing from the implement point of view. Based on the model, ABGC has been constructed using MAE wish is a multi-agent environment platform.

Due to the very generic nature of the grid computing, we can involve the research on it from different level, such as operating system layer, information layer, knowledge layer, and service-oriented layer. Agent-Based Grid Computing System (ABGCS) focuses on service-oriented layer in terms of current existing running environment. ABGCS will be a useful platform for research on Agent Based Grid Computing.

REFERENCES

[1] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith and S. Tuecke, A resource management architecture for metacomputing systems, in: *Proceedings IPPS/SPDP Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.

[2] N.R. Jennings and M.J. Wooldridge, eds, *Agent technology: foundations, applications, and markets*, Springer-Verlag, 1998

[3] R. Buyya, D. Abramson and J. Giddy, Nimrod/G: an architecture for a resource management and scheduling system in a global computational grid, in: *Proceedings 4th International Conference on High Performance Computing in Asia-Pacific Region*, Beijing, China, 2000.

[4] O.F. Rana and D.W. Walker, The Agent Grid: agent-based resource integration in PSEs, in: *Proceedings 16th IMACS World Congress on Scientific Computation*, Applied Mathematics and Simulation, Lausanne, Switzerland, 2000.

[5] J. Cao, D.J. Kerbyson, and G.R. Nudd, "Performance Evaluation of an Agent-Based Resource Management Infrastructure for Grid Computing", in Proc. of 1st IEEE Int. Symp. on Cluster Computing and the Grid, 311-318, 2001.

[6] K. Krauter, R. Buyya and M. Maheswaran, A taxonomy and survey of grid resource management systems, to appear in: *Software: Practice and Experience*, 2001.

[7] J. Cao, D.J. Kerbyson and G.R. Nudd, Performance evaluation of an agent-based resource management infrastructure for grid computing, in: *Proceedings 1st IEEE/ACM International Symposium on Cluster Computing and the Grid*, Brisbane, Australia, 2001, pp. 311–318.

[8] J.D. Turner, D.P. Spooner, J. Cao, S.A. Jarvis, D.N. Dillenberger, and G.R. Nudd, "A Transaction Definition Language for Java Application Response Measurement", J. Computer Resource Management, 105, 55-65, 2002.

[9] J. Cao, S.A. Jarvis, S. Saini, D.J. Kerbyson, and G.R. Nudd, "ARMS: an Agent-based Resource Management System for Grid Computing", to appear in Scientific Programming, 2002.