GrEASe: Grid Environment based on Agent Services

Antonio Boccalatte, Alberto Grosso, Christian Vecchiola, Sara Fazzari, Silvia Gatto

***Abstract*— This paper presents an agent-based infrastructure for grid computing called GrEASe (Grid Environment based on Agent Services). Grids are typically complex, heterogeneous, and highly dynamic environments, and agent technology can satisfy the basic requirements of this kind of contexts. GrEASe is organized as a two layer structure: the lower one providing the resource independent functionalities and the upper one providing all the grid-specific services. All the features of the grid infrastructure have been modeled with the multi-behavioral agent model of the AgentService programming framework. This platform is also the runtime environment for the multi-agent system associated with each grid node.**

***Index Terms*— Grid Computing, Multi-Agent Systems**

I. Introduction

RESOURCE sharing is nowadays an important issue, not only because it offers many advantages in distributed computing, but also because data sharing is becoming more and more useful in many fields. Resources can be classified in three different groups: data, services, and computational power. By following this classification we can distinguish three types of grids [1]. Data Grids manage huge collections of geographically distributed data, which can be generated in many different ways: data streams are daily sent from satellites for weather forecasts and climatic changes analysis; large collections of data generated from scientific experiments allow geographically distributed researchers to collaborate to the same research project. Service Grids provide services that could not be obtained from a single platform: streaming multimedia services or collaborative applications. Computational Grids provide the aggregate power of a collection of processors spread over the network as a unique, big processor. Grids are an economic and efficient way to compute, since they bring to the end user an incredible set of resources with a relatively low cost.

A Grid infrastructure is a complex and high dynamical environment: multiple, heterogeneous, and distributed resources need to be managed and accessed by means of a uniform interface. Real applications need also a customized interaction according to the different privileges of the users. The depicted scenario can certainly benefit from Agent technology [2]. Agents are autonomous software entities with some level of intelligence; agents work better if they belong to a community such as a multi-agent system (MAS) [3]. Agents act in a distributed manner, cooperate, compete, and negotiate to solve a problem or to perform a task. These features make the agents an interesting technology to implement Grid infrastructures.

In this paper GrEASe, an agent-oriented architecture which provides services in a Grid is described. GrEASe is implemented by the use of the AgentService programming platform [4].

A brief overview on agent technology and multi-agent systems is provided in Section II and how this technology can be applied to grid computing is explained. Section III includes the description of AgentService programming platform. Section IV the describes the features of GrEASe, while Section V presents an interesting use case of such architecture followed by a possible application of GrEASe to a real scenario. Conclusions follow in Section VI.

Manuscript received October 25 2004

A. Boccalatte is with Department of Communication, Computer and Systems Sciences, University of Genova, 16145 Genova Italy (phone: +39-010-353-2812; e-mail: nino@ dist.unige.it).

A. Grosso is with Department of Communication, Computer and Systems Sciences, University of Genova, 16145 Genova Italy (phone: +39-010-353-2284; e-mail: nino@ dist.unige.it).

C. Vecchiola is with Department of Communication, Computer and Systems Sciences, University of Genova, 16145 Genova Italy (phone: +39-010-353-2284; e-mail: nino@ dist.unige.it).

S. Fazzari was with Department of Communication, Computer and Systems Sciences, University of Genova, 16145 Genova Italy.

S. Gatto was with Department of Communication, Computer and Systems Sciences, University of Genova, 16145 Genova Italy.

II. Agents Technology And grid computing

A. Agents and Multi-Agent systems

A software agent is an autonomous software entity able to expose a flexible behavior. Flexibility is obtained by means of reactivity, pro-activity and social ability [3]. Reactivity is the ability to react to environmental changes in a timely fashion while pro-activity is the ability to show a goal directed behavior by taking the initiative. Social ability, that is the ability to interact with peers by means of cooperation, negotiation, and competition, is one of the most important features of agent oriented programming: agents do their best when they interoperate. Interaction is obtained by arranging agents in communities called multi-agent systems (MAS) [3]. MAS are generally decentralized open systems with distributed control and asynchronous computation: they provide a context for agents' activity with the definition of interaction and communication protocols. In addition they are scalable, fault-tolerant, reliable, and designed for reuse.

An abstract architecture specification of a generic multi-agent system has been proposed by the Foundation of Intelligent Physical Agents (FIPA), an international organization that promotes standards for agent technologies. The proposed architecture [6] is implemented by different multi-agent systems and has been taken as reference model in the comparison of different implementations of MAS.

B. *Agents and Grid Computing*

Agent technology has been a useful approach in different contexts: air traffic management [5], biologic systems modeling and simulation [7], workflow management [8], and on-line auction systems [9]. Moreover, different fields of computing have taken advantages from the agent oriented approach such as scheduling systems, collaborative smart agendas [10], information filtering [11], and soft-bots [12]. Agents are reliable components to build more flexible and fail safe systems, since autonomy and reactivity allow recovering from fault conditions. This is certainly necessary in high critical scenarios like air traffic management, but it is also a desirable in the case of grid computing. The social ability, such as cooperation, competition and negotiation, is equally fundamental in grids.

Grids are intrinsically distributed and complex systems, as they may require more than one step to provide a resource to a client. Interactions between nodes can change during time in order to make use of resources available at run time. Each node belonging to a grid needs to keep availability of the resources offering and benefits of a certain degree of autonomy and flexibility. Agent technology has been designed to model high dynamic and complex systems [13] and can fulfill many of the requirements related to the development of a grid infrastructure. By using agent technology, users and administrators of the resulting system can have a more friendly and understandable interface to interact with.

Some projects have already proved that the agent oriented approach could be an interesting solution in the field of Grid Computing.

A4, acronym for "Agile Architecture and Autonomous Agent" is a methodology for grid's resource managing. This approach, described in depth in [14] [15], is based on a flexible architecture, able to rapidly adapt to dynamic environmental changes. Agents are homogeneous and settled in a hierarchical structure, they have capabilities of service discovery and service advertisement.

MyGrid [16] is a Grid project which provides a collaborative environment for biologists working and living in different countries. The architectural design is based on agents and exploits their autonomy and their capability to implement complex interactions through negotiation messages in a generic Agents Communication Language (ACL). MyGrid relies on SoFAR (Southampton Framework for Agent Research) [17], that constitutes the agent oriented infrastructure used by MyGrid.

The Bond Agent System [18] is based on the JADE framework [19] and extends it by providing specific agent behaviours that abstract the concept of grid services.

The agent-oriented approach can take many advantages to field of Grid Computing. In particular it offers a flexible and high level approach that is, at the same time, powerful enough to handle all the different aspects of grid environments. Grids are dynamics by nature and agents have been modeled in order to get aware of the context in which they are situated and to dynamically interact with peers. Agents are also high level interfaces for humans, if compared to objects, and system designers can easily deal with them and organize the entire distributed system in a more clear way. All the presented projects rely on these features of agency and also GrEASe takes benefits from them by the means of the AgentService programming platform.

III. The AgentService Programming Platform

AgentService [4] is a multi-agent system development framework that provides a complete support to agent design, implementation, and management with a full run-time environment for agents scheduling, control and monitoring.

The framework has been developed with an extremely modular architecture in order to be customizable and portable over different architectures and operating systems. Modules cover:

- the storage subsystem (repository of all templates used to create agents in the platform);
- the persistence subsystem;
- the messaging subsystem;
- the logging subsystem.

Additional modules can be loaded into the platform in order to enrich and customize the platform services.

AgentService allows the definition of real, autonomous, and persistent agents. Agents have a multi-behavioral activity and organize their knowledge base in a set of persistent data structures shared among the different activities. Agents are scheduled and executed within the AgentService platform that provides them a set of services as defined by the FIPA2000 specification [6]:

- Agent Management System (AMS);
- Directory Facilitator (DF);
- Message Transport Service (MTS).

The agent model implemented in AgentService is based on the use of behaviors and knowledge. Behaviors include decisions and computational tasks; they dynamically determine the agent activity and influence its state.

Knowledge objects define the agent's knowledge base and consist of a set of shared data structures that can be persisted in order to preserve the agent's state and that are modified by the activity of Behavior objects.

AgentService provides to the developer a set of Agent Programming eXtensions (APX) [20] specifically designed to simplify the development and the implementation of agent oriented applications; they are a set of templates modeling the implementation of agents, behaviors and knowledge, represented as types in a C#-based programming language.

IV. GREASE ARCHITECTURE

A. Overall Overview

GrEASe (Grid Environment based on Agent Services) is an agent oriented infrastructure for grid computing. GrEASe architecture is structured in two layers: the lower one providing the basic services common to every grid, the upper one providing grid-specific functionalities. Both layers have been modeled by using an agent-oriented approach.

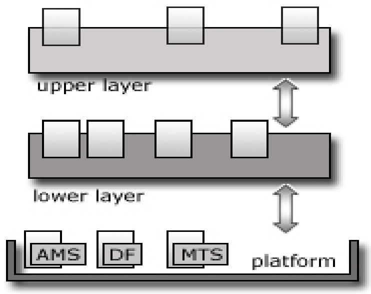

Figure 1 – Structure of a GrEASe node

B. The Lower Layer: Basic Infrastructure

The lower layer provides the basic grid functionalities classified as follows:

- node management: grids are dynamic environments where nodes can subscribe and unsubscribe at run-time. General information about the node status need to be accessed in order to provide the necessary interfaces to monitor and maintain the entire grid;
- resource querying and discovery: nodes can query and find resources by using a distributed dispatching system spread all over the nodes;
- authentication: users that access the resources need to be authenticated, since different policies are applied depending on the identity of the requestor;
- transport services: dispatch and receipt of the information and data.

The design and implementation of the lower layer led to the definition of different types of agents: NodeManager, Dispatcher, Authenticator, and Carrier.

NodeManager is the maintainer of the node and performs all the management operations. Three different behaviours have been designed in order to accomplish all the tasks of the NodeManager:

- node subscription and un-subscription from the Grid;
- monitoring services and information about the status of the node and of the resources;
- resource allocation and monitoring.

Dispatchers agents are spread all over the nodes and implement the resource querying and discovery process: each node has an instance of this type of agent. Dispatcher essentially forwards a request for a resource to the neighbor nodes, and waits for a response; at the same time it handles incoming requests from other dispatcher agents. Dispatcher is critical for performance of the resource search process and can support different search algorithms by simply changing the relevant behaviors.

Carrier agents implement the general file transfer service between nodes: agents inside the node instruct Carrier to send a file or are notified by the Carrier of an incoming file transfer for them. Different protocols (e.g. ftp protocol, or its secure version) can transparently be used to implement file transfer service, by defining the corresponding behaviors and selecting the most fitting ones for the specific context.

Authenticator agents are responsible of the user authentication process. The user profile is evaluated in order to grant:

- access to the grid system;
- access to the specified resource by applying the right policy.

Authenticator implements a two-level authentication strategy: first the credentials provided by the user are checked for the access to the grid system; then the availability of the resource is granted on the basis of the successful validation of the authorization criteria.

C. The Upper Layer: Grid-specific Components

Different types of grids are defined according to the different types of resources they share: processor-cycles, documents and data in general, or services. Therefore, specific requirements need to be fulfilled according to the different grid types. Resources of data-grids should be accessed at the same time by multiple clients. Conversely, in a computational-grid resources can be assigned only to a single client at time since the same processor cycles cannot be shared between multiple users.

The upper layer of the GrEASe architecture takes care of all the peculiar features related to the specific type of the chosen grid. The upper layer is defined by all those agents that strictly interact with the resources belonging to the grid and hosted in the node. For example let's define agents' behaviors according to the requirements of Computing Grids: the submission of a task to a node for computing means not only the transfer of the executable code of the task, but also the transfer of the requested input and output data. In addition, if tasks are not monolithic it may be convenient to monitor their progress. These functionalities can be encapsulated by the implementation of specific run-time behaviors, one for handling the task execution, another for monitoring task progress. A similar approach can be adopted for Data and Service Grids.

V. GREASE IN ACTION

In order to see how GrEASe agents interact to provide a grid service the process of resource querying and discovery will be briefly described.

Figure 2 shows an instance of the AgentService platform running on each grid node (two expanded) that schedules resource agents and infrastructure agents. A client application asks NodeManager of the nearest node by providing user credentials to access the grid. NodeManager forwards the credentials to the co-located Authenticator and waits for feedback. Authenticator verifies the user credentials and in case of success sends an approval message to the user and notifies the NodeManager, which updates the user login status.

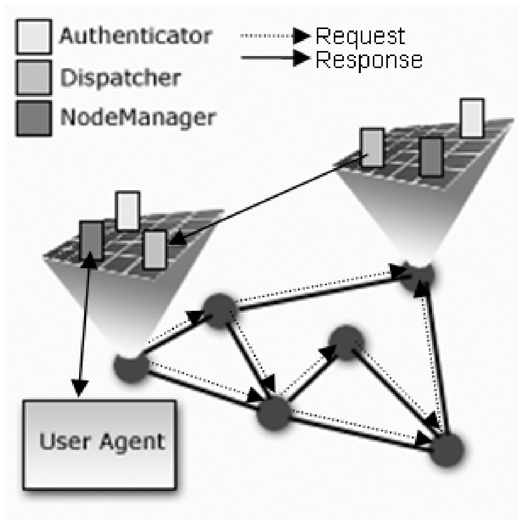

Figure 2 – Resource querying process

Once authenticated the users asks the NodeManager for a given resource. The NodeManager checks the resource availability on its node: if the resource is found it notifies the user, and following the user's confirmation, it instructs Carrier to dispatch the resource; if the resource is not available in the node, NodeManager forwards the request to the co-located Dispatcher who will distribute the request to others Dispatcher agents across the network, according to the selected resource search algorithm. Every Dispatcher reports the query to its NodeManager and the same process described before applies. If no resources are found in the grid, a time-out function associated to the query makes the query inactive. If more than one node answers that the resource is available, the user asks to send only the first answering node. All the messages across the nodes use the address provided by Directory Facilitator.

Knowledge objects used in this process are:

- Grid topology by Dispatcher;
- User credentials by Authenticator;
- Resource availability by NodeManager;
- Logged users by NodeManager.

The architecture of GrEASe is flexible enough to handle all the different scenarios of Grid Computing. An interesting application of GrEASe can be found in the modeling and the simulation of the peer-to-peer (p2p) nets for sharing data: such networks can be considered a sort of Data Grids:

- they provide to the end user a huge volume of data that is spread all over the network;
- the end user access all the data available in the network and the this access is independent from the physical location of data;
- nodes of the net can act either as servers for other nodes or as clients that feed data.

There are some aspects that make Data Grids different from peer-to-peer networks:

- peer-to-peer networks do not implement sophisticated access control techniques and do not have refined user profiles;
- peer-to-peer networks normally provide many different copies of the same data and do not worry about the synchronization of the different copies.

These aspects make peer-to-peer networks only less complex than Data Grids.

By the use of GrEASe it is possible to model each node of the network with an installation of the AgentService platform that runs the agents defined by the GrEASe architecture. The NodeManager will be responsible for the local resources of the node, while the Dispatcher and Carrier will be programmed in order to interact with peers also by using the most known p2p protocols: in this way the nodes of the GrEASe architecture can easily be integrated with the already existing p2p networks. Since peer-to-peer networks normally have simple user policies the Authenticator will provide only the basic functionalities of authentication when needed.

In peer-to-peer networks resources normally refer to simple files and for this reason there is no need to define particular agents that represents the resources inside the platform. The upper layer will be configured with particular agents:

- if the node is attached to an end user the upper layer will require a user-agent that handles the user requirements and control the behavior of the node for the user;
- in the case of the node is intended to measure and monitor the traffic that flows through it a special agent can be designed to track all this kind of information and report it to the user;

The introduction of the agents into the upper layer is rather simple because agents rely on the platform services to perform their activity: by querying the Directory Facilitator can dynamically discover the NodeManager; each agent uses the message based system provided by the MTS to interact with the other agents and they can easily interact with the other "citizens" of the platform once they know the ontology that NodeManager, Dispatcher, Carrier, and Authenticator support. These ontologies are made available to each agent by the platform through the Directory Facilitator.

The architecture provided with GrEASe, the services offered by the AgentService platform, and the approach defined with the agent-oriented paradigm, allow a quick and not difficult implementation of the described example. In fact, designers can better concentrate on the peculiar aspects of the example rather than define the overall infrastructure and the programming model needed to implement the example.

VI. Conclusion and Future Work

Agent technology and in particular agent oriented decomposition has played a key role in the design and the implementation of GrEASe. The division of the tasks to the different types of agents has led to a flexible and customizable architecture. Interaction between agents is done with clean and fixed interfaces defined by the messages they exchange and this allows loose coupling among the different components.

The approach taken with GrEASe is different from the ones adopted by the other similar projects like A4 and the Bond Agent System: while A4 leverages on a hierarchical structure used to organized the resources in the grid, GrEASe adopts a two-level architecture that separates the features common to all the grid types from the features peculiar to the specific grid type. Moreover, GrEASe is not tied, as in the case of the Bond Agent System, to a strong BDI architecture but the use of AgentService allows a more open environment.

The architecture provided with GrEASe, the services offered by the AgentService platform, and the approach defined by the agent-oriented paradigm offer to developers a basic set of functionalities. In particular, the agent oriented approach and the fact that agents live inside a multi-agent system that relies on the services of a platform, are an important abstraction on which the GrEASe architecture is founded. GrEASe exploits the services of AgentService in order to deliver to the developer an high-level tool to model real applications in the field of Grid Computing. A simple example has been discussed in order to show to the user that the approach promoted by GrEASe can be an interesting solution.

Currently GrEASe implements the resource search and delivery in the three common grid types: Data, Computational and Service. The most natural expansion of future development is to allow the interaction between GrEASe and other existing legacy grids. In this case AgentService will be used to shape Interface Agents to access legacy grids in accordance to their individual established rules.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid. Enabling Scalable Virtual Organizations", *International Journal of Supercomputer Applications*, 2001.

[2] N.R. Jennings, and M. Wooldridge "Agent-Oriented Software Engineering", *Proceedings of the 9th European Workshop on Modelling Autonomous Agents in a Multi-Agent World : Multi-Agent System Engineering (MAAMAW-99)*, 1999.

[3] G. Weiss, *Multi-agent Systems – A Modern Approach to Distributed Artificial Intelligence*, G. Weiss Ed., Cambridge, MA, 1999.

[4] A. Boccalatte, A. Gozzi, and A. Grosso, "Una Piattaforma per lo Sviluppo di Applicazioni Multi-Agente", *WOA 2003: dagli oggetti agli agenti – sistemi intelligenti e computazione pervasiva*, Villa Simius, Italy, September 2003.

[5] A. S. Rao, M. P. Georgeff, and D. Kinny, "A Methodology and Modelling Technique for Systems of BDI Agents Agents Breaking Away", *Proceedings of the Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World, (MAAMAW'96)*, published by Springer as Lecture Notes in Artificial Intelligence 1038, 1996.

[6] "FIPA Abstract Architecture Specification", FIPA standard SC00001L, http://www.fipa.org/specs/fipa00001/SC00001L.pdf.

[7] H. Van Dyke Parunak, "Go to the Ant: Engineering Principles from Natural Multi-Agent Systems", Forthcoming in Annals of Operations Research, special issue on Artificial Intelligence and Management Science.

[8] R. Sacile, E. Montaldo, M. Coccoli, M. Paolucci, and A. Boccalatte, "Agent-based architectures for workflow management in manufacturing", SSGRR 2000, L'Aquila I, Aug. 2000.

[9] C. Beam, and A. Segev, "Automated Negotiations: A Survey of the State of the Art", *Wirtschaftsinformatik*, Vol. 37-3, 1997, pp. 263-268.

[10] B. P. C. Yen, "Agent-based Distributed Planning and Scheduling in Global Manufacturing", in *Proc. of the thrid Annual International Conference on Industrial Engineering Thories, Applications and Practice*, Hong Kong, December, 2000

[11] P. Maes, and A. Moukas, "Amalthaea: An Evolving Multi-Agent Information Filtering and Discovery System for the WWW", *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 1, no. 1, 1998, pp. 59-88.

[12] O. Etzioni, and D. Weld, "A Softbot-Based Interface to the Internet", *Communications of the ACM*, 37, 7, 1994.

[13] M. Wooldridge, "Intelligent Agents", in *Multi-agent Systems – A Modern Approach to Distributed Artificial Intelligence*, G. Weiss Ed., Cambridge, MA, 1999, pp. 27-78.

[14] J. Cao, D. P. Spooner, J. D. Turner, S. A. Jarvis, D. J. Kerbyson, S. Saini, and G. R. Nudd, "Agent-Based Resource Management for Grid Computing", in *Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, 2002.

[15] J. Cao, D. J. Kerbyson, G. R. Nudd, "Performance Evaluation of an Agent-Based Resource Management Infrastructure for Grid Computing", in *Proc. of 1st IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid '01)*, Brisbane, Australia, May 2001.

[16] L. Moreau, S. Miles, C. Goble, M. Greenwood, V. Dialani, M. Addis, N. Alpdemir, R. Cawley, D. De Roure, J. Ferris, R. Gaizauskas, K. Glover, C. Greenhalgh, M. Greenwood, P. Li, X. Liu, P. Lord, M. Luck, D. Marvin, T. Oinn, N. Paton, S. Pettifer, M. V Radenkovic, A. Roberts, A. Robinson, T. Rodden, M. Senger, N. Sharman, R. Stevens, B. Warboys, A. Wipat, and C. Wroe, "On the Use of Agents in a BioInformatics Grid", in *Proc. of the Third IEEE/ACM CCGRID'2003 Workshop on Agent Based Cluster and Grid Computing*, Sangsan Lee, Satoshi Sekguchi, Satoshi Matsuoka, and Mitsuhisa Sato ed., Tokyo, Japan, May 2003, pp 653-661.

[17] L. Moreau, N. Gibbins, D. DeRoure, S. El-Beltagy, W. Hall, G. Hughes, D. Joyce, S. Kim, D. Michaelides, D. Millard, S. Reich, R. Tansley, and M. Weal, "SoFAR with DIM Agents: An Agent Framework for Distributed Information Management", in *Proc. Of The Fifth International Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, Manchester, UK, Apr 2000, pp. 369–388

[18] M.A. Khan, S.K.Vaithianathan, K. Sivoncic, and L. Boloni, "Towards an Agent Framework For Grid Computing", *CIPC-03 Second International Advanced Research Workshop on Concurrent Information Processing and Computing*, Sinaia, Romania, 2003.

[19] F. Bellifemmine, G. Rimassa, and A. Poggi, “JADE – A FIPA compliant Agent Framework”, in *Proc. of the 4th international Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents*, London, 1999.

[20] A. Boccalatte, C. Vecchiola, and M. Coccoli, “Agent Programming Extensions relying on a component based platform”, in *Proc. of the 2003 IEEE International Conference on Information Reuse and Integration*, Las Vegas, NV, October 2003, pp. 24-31.