Multi-Agent Support for Internet-Scale Grid Management

B.J. Overeinder, N.J.E. Wijngaards, M. van Steen, and F.M.T. Brazier
Department of Computer Science, Faculty of Sciences, Vrije Universiteit Amsterdam,
de Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{bjo,niek,steen,frances}@cs.vu.nl

Abstract

Internet-scale computational grids are emerging from various research projects. Most notably are the US National Technology Grid and the European Data Grid projects. One specific problem in realizing wide-area distributed computing environments as proposed in these projects, is effective management of the vast amount of resources that are made available within the grid environment. This paper proposes an agent-based approach to resource management in grid environments, and describes an agent infrastructure that could be integrated with the grid middleware layer. This agent infrastructure provides support for mobile agents that is scalable in the number of agents and the number of resources.

1 Introduction

Computational grids are wide-area (Internet-scale) distributed environments that differ from conventional distributed computing by their focus on large-scale resource sharing, innovative applications, and high-performance orientation (Foster et al., 2001).

In a grid architecture, four levels of management can be distinguished: fabric, connectivity, single resource, and collective multiple resources. The fabric layer typically constitutes computational resources, storage resources, network resources, and code repositories. The connectivity layer deals with easy and secure communication by providing single sign on, delegation, integration with various local security solutions, and user-based trust relationships. The resource layer is concerned with individual resources, and the two primary classes of resource layer protocols are information protocols and management protocols. The collective multiple resources layer provides directory services, co-allocation, scheduling, and brokering services, monitoring and diagnostics services, data replication services, grid-enabled programming systems, workload management systems and collaboration frameworks (problem solving environments), etc.

In particular, coordinating collective resources is a complex high-level task that integrates the multiple resources into a wide-area distributed system. Many of the services in this layer can be effectively facilitated by applying *multi-agent systems*. Co-allocation, scheduling, and brokering services, monitoring and diagnostic services, workload management and collaboration frameworks, community authorization servers, community accounting and payment services, and collaborative services are processes that require intelligence, autonomy, and social capabilities: all qualities that are characteristic to intelligent agents (Wooldridge and Jennings, 1995).

A distinct problem is *scalability* in Internet-scale distributed systems like the Grid. In this paper, scalability refers to scalability of the wide-area distributed computing infrastructure and services, not of applications. Services such as resource management, co-allocation, and scheduling must deal with scalability problems that appear in large-scale, wide-area distributed systems (Wijngaards et al., 2002). Centralized (client-server) approaches have scalability problems as there is one central authority coordinating the activities. Hierarchical approaches already direct to a scalable solution, but peer-to-peer interaction strategies as embraced by agent technologies seems to be the most promising approach to provide scalable and adaptive services.

This paper presents a multi-agent infrastructure, called AgentScape, that can be employed for an agent-based approach to integrate and coordinate distributed resources in a computational grid environment. In particular, scalability, heterogeneity, and interoperability are discussed in perspective to a grid environment and the proposed multi-agent infrastructure.

2 Background

A number of initiatives to apply agents in computational grids have been initiated in recent years. Manola and Thompson (1999) present an overview of different perspectives to grid environments and describe DARPA's Control of Agent-Based Systems (CoABS) agent grid. In the CoABS Grid, a number of application level and functional requirements hold. Specifically, applications are considered to have multi-year lifetimes, evolving and changing requirements, are adaptable and scalable, and allows for system management without explicitly monitoring all components all the time. Practically, agent

technology is expected to help to provide more reliable, scalable, survivable, evolvable, adaptable systems, and help to solve data blizzard and information starvation problems. From a functional point of view, the CoABS Grid knows not only about agents, but also about their computational requirements, and about available computational (and other) resources. Hence, the CoABS Grid provides a unified, heterogeneous distributed computing environment in which computing resources are seamlessly linked.

Bradshaw et al. (2001) remark that "cyberspace" is currently a lonely, dangerous, and relatively impoverished environment for software agents. Consequently, most of today's agents are designed for "solitary, poor, nasty, brutish, and short" lives of narrow purpose in a relatively bounded and static computational world. They argue that focusing greater attention to making the environment in which agents operate more capable of sustaining various types of agents and collaboration groups, would simplify some of these problems. The CoABS Grid provides the infrastructure for large-scale integration of heterogeneous agent frameworks. The CoABS Grid capabilities have been extended by integrating the NOMADS agent environment for strong mobility and safe execution and the KAoS framework for policy-based management of agent domains to support long-lived agents and their communities (Bradshaw et al., 2001).

A good example of an agent grid is presented by Rana and Walker (2000). They identify the need to combine problem-specific problem solving environments (PSEs), facilitating interoperability between various tools and specialized algorithm each PSE supports. An agent based approach to integrate services and resources for establishing multi-disciplinary PSEs is described, in which specialized agents contain behavioral rules, and can modify these rules based on their interaction with other agents and with the environment in which they operate.

Another type of application of agents in distributed or parallel computing is typically with master-slave computations in wide-area distributed environments (Ghanea-Hercock et al., 1999). In these systems, large computations are initiated under control of a coordinating agent that distributes the computation over the available resources by sending mobile agents to these resources. In this perspective it is in some way similar to the Condor system or the SETI@home experiment, which also incorporate coordination and distributing the computation of the available resources. Essentially, the added value of the distributed computing agent systems is similar to the agent grid: coordination and seamless integration of the available distributed resources.

Principal idea behind a grid infrastructure is resource sharing and providing services. The introduction already outlined the four levels of management that can be determined in the process of sharing resources over a wide-area network. With respect to providing services in a grid environment, Foster et al. (2002) presented an Open Grid Services Architecture that addresses the challenges to achieve various qualities of service when running applications on top of different native platforms. The architecture builds on concepts and technologies from the Grid and Web services communities. The architecture defines a uniform exposed service semantics; defines standard mechanisms for creating, naming, and discovering transient Grid service instances; provides location transparency and multiple protocol bindings for service instances; and supports integration with underlying native platform facilities. The Open Grid Services Architecture also defines interfaces and associated conventions, mechanisms required for creating and composing sophisticated distributed systems, including lifetime management, change management, and notification. Service bindings can support reliable invocation, authentication, authorization, and delegation, if required. The resource sharing mechanisms are complementary to the Grid services, but can be incorporated to implement a service-oriented architecture.

3 AgentScape: A Scalable Multi-Agent Infrastructure

AgentScape is a middleware layer that supports large-scale agent systems. The rationale behind the design decisions are (i) to provide a platform for large-scale agent systems, (ii) support multiple code bases and operating systems, and (iii) interoperability with other agent platforms. The consequences of the design rationale with respect to agents and objects, interaction, mobility, security and authorization, and services are presented in the following subsections.

3.1 The AgentScape Model

The overall design philosophy is "less is more," that is, the AgentScape middleware should provide a minimal but sufficient support for agent applications, and "one size does not fit all," that is, the middleware should be adaptive or reconfigurable such that it can be tailored to a specific application (class) or operating system/hardware platform.

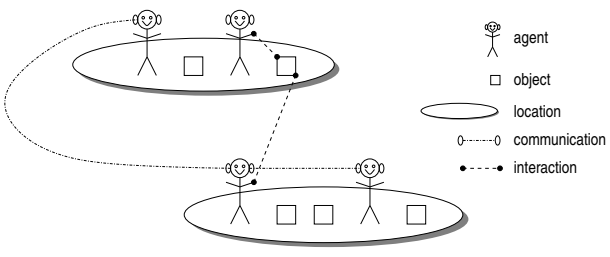

Figure 1: AgentScape conceptual model.

Agents and *objects* are basic entities in AgentScape. A *location* is a "place" in which agents and objects can reside (see Fig. 1). Agents are active entities in AgentScape that interact with each other by message-passing communication. Furthermore, agent migration in the form of weak mobility is supported (Picco, 2001). Objects are passive entities that are only engaged into computations reactively on an agent's initiative. Besides agents, objects, and locations, the AgentScape model also defines *services*. Services provide information or activities on behalf of agents or the AgentScape middleware.

Scalability, heterogeneity, and interoperability are important principles underlying the design of AgentScape. The design of AgentScape includes the design of agents, objects and services, interactions, migrations, security and authorization, as well as the agent platform itself. For example, scalability of agents and objects is realized by distributing objects according to a per-object distribution strategy, but not agents. Instead, agents have a public representation that may be distributed if necessary.

The basic idea in the AgentScape model is that most of the functionality is provided by the *agent interface* implementations such that the middleware (or the agent representation of the middleware) can be designed to perform basic functions. This approach has a number of advantages. First as the middleware must provide basic functionality, the complexity of the design of the middleware can be kept manageable and qualities like robustness and security of the middleware can be more easily asserted. Additional functionality can be implemented in the agent-specific interface implementation (see Fig. 2).

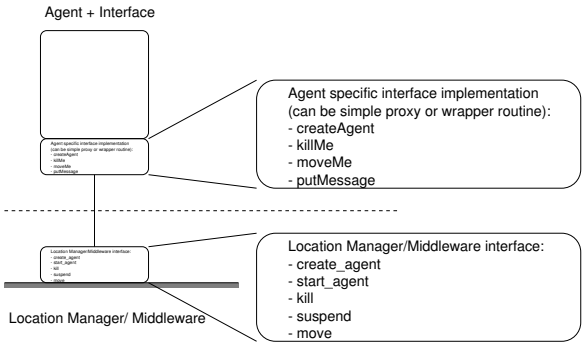

Figure 2: The AgentScape interface model.

Agent-agent interaction is exclusively via message-passing communication. Asynchronous message passing has good scalability characteristics with a minimum of synchronization between the agents. Tuple spaces also provide a mechanism for communication that does not enforce synchronization between the communicating partners, but also cannot enforce the actual receipt of the information.

Agent migration between locations is based on weak mobility. The *state* of the agent is captured (e.g., the variables referenced by the agent) but not the *context* of the agent (e.g., stack pointer and program counter).

3.2 An AgentScape Architecture

The four basic concepts agents, objects, locations, and services are further implemented in the AgentScape architecture. Agents and objects are supported by *agent servers* and *object servers* respectively. Agent servers provide the interface and access to AgentScape to the agents that are hosted by the agent server. Similarly, objects servers provide access to the objects that are hosted by the object server. A location is a closely related collection of agent and object servers, possibly on the same (high-speed) network, on hosts which are managed in the same administrative domain.

Depending on the policy or resource requirements, one agent can be exclusively assigned to one agent server, or a pool of agents can be hosted by one agent server. The explicit use of agent servers makes some aspects in the life cycle model of agents more clear. An active agent is assigned to, and runs on a server; a suspended agent is not assigned to an agent server. In this model, starting a newly created, or activating an existing suspended agent, is similar, and some design decisions of the agent life cycle can be simplified.

The use of agent and object servers is transparent to the agents. Hence, agent servers do not belong to the AgentScape model from the agent perspective. However, an agent could ask the middleware to determine on which agent server the agent runs.

The *AgentScape Operating System* (AOS) forms the basic fundament of the AgentScape middleware. An overview of the AgentScape architecture is shown in Fig. 3. The AOS offers a uniform and transparent interface to the underlying resources and hides various aspects of network environments, communication, operating system, access to resources, etc. The AgentScape API is the interface to the middleware. Both agents and services (e.g., resource management and directory services) use this interface to the AOS middleware.

The design of the AgentScape Operating System is *modular*. The AOS kernel is the central active entity that coordinate all activities in the middleware. The modules in the AOS middleware provide the basic functionality. Below a brief overview of the most important modules is given. The life-cycle module is responsible for the creation and deletion of agents. The communication module implements a number of communication services, e.g., similar to UDP, TCP, and streaming, with different qualities-of-service. Support for agent mobility is implemented in the migration module. The location service associates an agent identifier with an address (or contact-point). There are also location services for objects and locations. The security architecture is essential in the design of AgentScape, as it is an integral part of the middleware. Many components in the middleware have to request authentication or authorization in order to execute

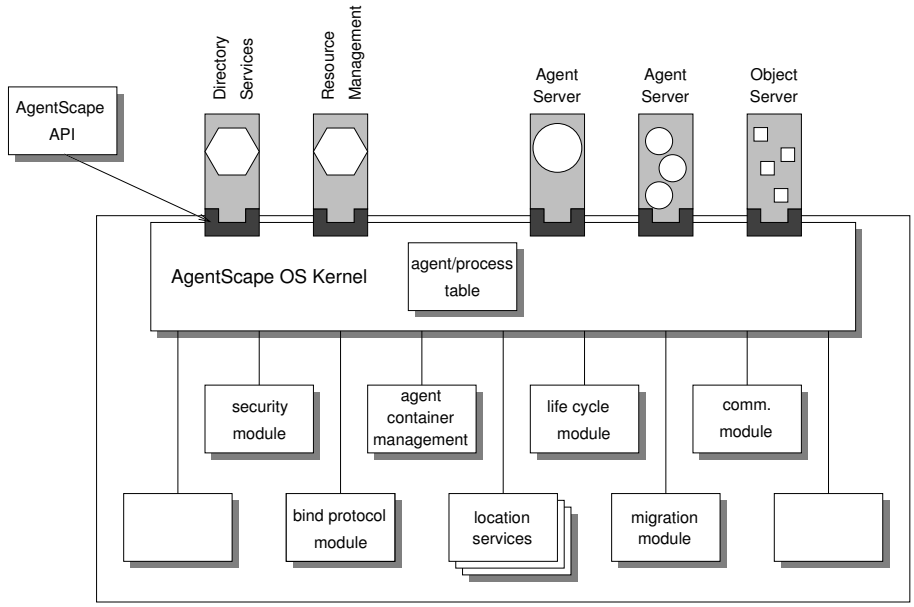

Figure 3: An AgentScape middleware architecture.

their tasks.

In AgentScape, interoperability between agent platforms can be realized in two ways. First by conforming to standards like FIPA or OMG MASIF. These agent platform standards define interfaces and protocols for interoperability between different agent platform implementations. For example, the OMG MASIF standard defines agent management, agent tracking (naming services), and agent transport amongst others. The FIPA standard is more comprehensive in that it defines also agent communication and agent message transport, and even defines an abstract architecture of the agent platform. A second approach to interoperability is realized by reconfiguration or adaptation of the mobile agent. This can be accomplished by an agent factory as described by Brazier et al. (2002), which regenerates an agent given a blueprint of the agent's functionality and its state, using the appropriate components for interoperability with the other agent platform.

3.3 AgentScape Prototype

The current prototype implementation of the AgentScape architecture provides the following basic functionality: creation and deletion of agents, communication between agents and middleware, and weak migration of agents. The AgentScape Operating System kernel and some basic services are implemented in the programming language Python, while the agent server is implemented in Java. As a proof of concept, the middleware not only supports agents written in different programming languages, but its components are implemented in different programming languages.

4 Supporting Agent-Based Grid Management

Resource management is one of the central components of wide-area distributed computing systems like a grid architecture. There have been various projects focused on grid computing that have designed and implemented resource management systems with a variety of architectures and services. Krauter et al. (2002) describe a comprehensive taxonomy for resource management architectures. The design objectives and target applications for a Grid motivate the architecture of the resource management system.

Decentralized, peer-to-peer interaction, resource trading, and machine learning are typically application areas where multi-agent systems are a potentially effective solution. Based on the definition of, for example, the Open Grid Services Architecture (Foster et al., 2002), multi-agent systems can be integrated with grid environments to provide services such as resource management. To integrate multi-agent systems, middleware support for agents should also be integrated with the grid environment.

For the interoperability of AgentScape and a grid environment, or more specifically a grid middleware, two levels of interoperability are important: *runtime system* and *middleware level*.

First, interoperability, extensibility and adaptivity at the runtime system level is incorporated in the agent interface, that is, code associated with an agent's implementation. Agents can provide their own implementation of a runtime system which makes calls on the middleware. This makes the agents adaptable to different environments and extensible if other runtime services are required. The interfaces can be loaded dynamically, and after migration

of the agent to another platform, a platform specific interface implementation can be bound to the agent. This flexibility makes the agent highly adaptive and extendible.

Second, interoperability and adaptivity at the middleware level is provided by the component-based design of the incorporated functionality of the middleware. That is, the components that implement the required functionality have a clearly defined interface, and can be replaced with other implementations. For example, the standard communication module that is included with AgentScape, can be replaced by a communication module supported by the grid middleware. This is also according the Globus design philosophy where existing or proprietary technologies can be incorporated in the Globus system (Foster et al., 2001).

Agent-based methods for coordination and control mechanisms in heterogeneous distributed system is a new and active research area. Minsky and Ungureanu (2000) formulate the requirements for agent-based coordination and control: (i) coordination policies need to be enforced; (ii) the enforcement needs to be decentralized; (iii) coordination policies need to be formulated explicitly; and (iv) it should be possible to deploy and enforce a policy incrementally. Minsky and Ungureanu describe a law-governed interaction mechanism that satisfy these principles, and can be used as a model for an agent-based approach to coordination and control of resource management system.

5 Summary and Future Work

Computational grids are often used for computationally intensive applications. Grids are currently expanding to Internet-scale sizes. Management issues, including task allocation and resource management, becomes a very important issue. Agent-based approaches may facilitate the management of these large-scale grids. Unfortunately, almost all of the current agent-based systems are not developed for large-scale environments; CoABS is a notable exception.

AgentScape, a large-scale distributed agent system, designed to support heterogeneity and interoperability, facilitates extensibility: it is relatively easy to build agent environments "on top of" AgentScape. AgentScape is also relatively easily adapted to different (lower-level) operating systems and network infrastructures. As such, AgentScape can be relatively easily integrated with other environments and support agent-based approaches to grid resource management.

Future work is further development of the AgentScape prototype. An elaborate management system will be incorporated to deal with performance, security, fault-tolerance, and accounting. Other research issues are scalable services for agents, such as name, location, and directory services. Agent-based scheduling and resource allocation algorithms have to be developed and evaluated on the AgentScape middleware.

References

J. M. Bradshaw, N. Suri, A. J. Cañas, R. David, K. Ford, R. Hoffman, R. Jeffers, and T. Reichherzer. Terraforming cyberspace. *Computer*, 34(7):48–56, July 2001.

F. M. T. Brazier, B. J. Overeinder, M. van Steen, and N. J. E. Wijngaards. Agent factory: Generative migration of mobile agents in heterogeneous environments. In *Proceedings of the ACM Symposium on Applied Computing (SAC 2002)*, Madrid, Spain, March 2002.

I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. The physiology of the Grid: An open Grid services architecture for distributed systems integration. `http://www.globus.org/research/papers/-ogsa.pdf`, January 2002.

I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal on High Performance Computing Applications*, 15 (3):200–222, Fall 2001.

R. Ghanea-Hercock, J. C. Collis, and D. T. Ndumu. Co-operating mobile agents for distributed parallel processings. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 398–399, Seatle, WA, April 1999.

K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience*, 32(2):135–164, February 2002.

F. Manola and C. Thompson. Characterizing the agent grid. `http://www.objs.com/agility/tech-reports/990623-characterizing-the-agent-grid.html`, June 1999.

N. Minsky and V. Ungureanu. Law-governed interaction: A coordination and control mechanism for heterogeneous distributed systems. *ACM Transactions on Software Engineering and Methodology*, 9(3):273–305, July 2000.

G. P. Picco. Mobile agents: An introduction. *Microprocessors and Microsystems*, 25(2):65–74, April 2001.

O. F. Rana and D. W. Walker. 'The Agent Grid': Agent-based resource integration in PSEs. In *Proceedings of the 16th IMACS World Congress on Scientific Computing, Applied Mathematics and Simulation*, Lausanne, Switzerland, August 2000.

N. J. E. Wijngaards, B. J. Overeinder, M. van Steen, and F. M. T. Brazier. Supporting Internet-scale multi-agent systems. *Data and Knowledge Engineering*, 2002. (To appear).

M. J. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2): 115–152, 1995.