Future Generation Computer Systems 25 (2009) 747–755

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An agent architecture for managing data resources in a grid environment

María S. Pérez[a,*], Alberto Sánchez[a], Jemal H. Abawajy[b], Víctor Robles[a], José M. Peña[a]

[a] DATSI. FI. Universidad Politécnica de Madrid, Spain

[b] Deakin University, Victoria, Australia

ARTICLE INFO

Article history:
Received 20 January 2006
Received in revised form
2 July 2008
Accepted 22 July 2008
Available online 31 July 2008

Keywords:
Multiagent system
Cooperation
Parallel file system
Data grid

ABSTRACT

The agent paradigm has been successfully used in a large number of research areas. MAPFS, a parallel file system, constitutes one successful application of agents to the I/O field, providing a multiagent I/O architecture. The use of a multiagent system implies *coordination* and *cooperation* among its agents. MAPFS is oriented to clusters of workstations, where agents are applied in order to provide features such as caching or prefetching. The adaptation of MAPFS to a grid environment is named MAPFS-Grid. Agents can help to increase the performance of data-intensive applications running on top of the grid.

This paper describes the conceptual agent framework and the communication model used in MAPFS-Grid, which provides the management of data resources in a grid environment. The evaluation of our proposal shows the advantages of using agents in a data grid.

1. Introduction

Several agent-based applications have been built in diverse areas such as business [20], electric management [19,7], control [2,3], networks [11] or in general, industrial applications [17,18].

Agents provide several concepts, which allow software engineers to design applications in a way closer to the human thought. Furthermore, agents offer useful characteristics to deal with complex and dynamic environments.

Nevertheless, if our objective is applying the agent technology to a field, like *operating systems*, we find some difficulties. In fact, there are important limitations when agents are applied to operating systems. These can make complex the success combination of these two fields. The most important ones are:

- The agent paradigm interacts with the infrastructure at a higher level than the operating system.
- Efficiency is a very strict requirement in the case of the development of a component of the operating system, and specifically a file system. Agent technology introduces an abstraction layer and, thus, it involves a loss of efficiency.

Nevertheless, these limitations can be avoided, since the agent paradigm distinguishes clearly between agent theory, which provides concepts related to the agent field, and agents architectures, which offer specific solutions and implementations.

MAPFS is a successful application of the agent theory in the development of a parallel file system [26]. The same philosophy is applied to the design of MAPFS-Grid [27], the adaptation of this parallel file system to data grids. Our proposal is to demonstrate that MAPFS-Grid takes advantage of the use of a multiagent system as a conceptual framework in its design and development.

The outline of this paper is as follows: Section 2 introduces the MAPFS-Grid system and describes the related work. Section 3 describes the generic structure of an agent in MAPFS-Grid. Section 4 analyzes the MAPFS-Grid cooperation model and describes the communication features of MAPFS-Grid. Section 5 shows the implementation and evaluation of MAPFS, in order to measure the influence of agents in the management of data resources. Finally, Section 6 summarizes our conclusions and suggests future work.

2. Problem statement and related work

2.1. MAPFS-Grid overview

MAPFS-Grid [27,30,31] is a generic framework for increasing the performance of I/O operations in grid environments. Thus, MAPFS-Grid is a suitable approach for data-intensive applications executed on data grids.

MAPFS-Grid makes use of MAPFS [26], a high-performance parallel file system for clusters of workstations. MAPFS (*Multi Agent Parallel File System*) has been developed in the Universidad Politécnica de Madrid since 2003. The main contribution of

* Corresponding address: DATSI. FI. Universidad Politecnica de Madrid, Facultad de Informatica, Campus de MontegancedoBoadilla del Monte, 28660 Madrid, Spain. Tel.: +34 91 336 73 80; fax: +34 91 336 73 73.

E-mail addresses: mperez@fi.upm.es (M.S. Pérez), ascampos@fi.upm.es (A. Sánchez), jemal@deakin.edu.au (J.H. Abawajy), vrobles@fi.upm.es (V. Robles), jmpena@fi.upm.es (J.M. Peña).

0167-739X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2008.07.011

MAPFS is the conceptual use of agents to offer new properties to applications, with the aim of increasing their adaptation to dynamic and complex environments.

The feasibility of the combination between MAPFS and MAPFS-Grid is due to the fact that grid environments are composed of different and heterogeneous resources, clusters being one of the most used, because of its good relation power vs. cost. Thus, it is possible to improve the grid data operations through parallel accesses into the clusters resources. MAPFS distributes data stripes over all the nodes of a cluster. On the other hand, MAPFS-Grid allows heterogeneous servers connected by means of a wide-area network to be used as data repositories, by storing data in a parallel way through all the clusters and individual nodes which compose the grid.

The heterogeneity of grid environments makes the application of parallelism difficult. In fact, since every resource of the grid can be composed of several components (e.g., clusters of workstations), it is necessary to optimize the I/O performance of every resource before tackling the global I/O optimization. Therefore, MAPFS-Grid provides two levels of software parallelism in a grid:

(1) The high level provides parallelism among the grid storage elements, that is, *inter-storage element parallelism*. These clusters can be heterogeneous and belong to different virtual organizations. The only requirement is that the user application must have permission on the final storage elements.
(2) The low level provides parallelism among the set of nodes of each cluster, that is, *intra-cluster parallelism*. This is made through the *Parallel Data Access Service (PDAS)* of MAPFS-Grid, which allows parallel I/O operations to be made in a cluster environment, providing access to the MAPFS file system.

Both levels are integrated and cooperate with the aim of providing an enhanced I/O bandwidth. Fig. 1 shows the double parallelism of MAPFS-Grid. The inner level is achieved only if the storage element is a cluster of nodes. The outer level is provided among a set of storage elements, at grid level.

MAPFS is based on a multiagent architecture, named MAPFS_MAS, which provides support to the main MAPFS subsystem (MAPFS_FS) in three different areas:

- Access to the information: This feature is the main task of MAPFS_MAS. Data is stored in I/O nodes (a set of disks distributed among several server nodes). Two different kinds of agents are used for providing this capability: *Extractor agents* are responsible for invoking parallel I/O operations and *distributor agents* distribute the workload to extractor agents.
- Caching and prefetching services: MAPFS takes advantage of the temporal and spatial locality of data stored in servers. A cache has a copy of the most recently used data in a storage device, which is faster than the original storage device. However, using a cache causes an important coherence problem. Inside MAPFS_MAS, there is a set of agents which manage this feature. These agents are named *cache agents*. They are responsible for using a cache coherence protocol and control data transfer between both storage devices.
- I/O optimizations: MAPFS takes advantage of different I/O optimizations techniques, such as caching and prefetching or use of hints. Hints are structures known and built by the file system, which are used for improving the read and write routines performance. In MAPFS, hints can be determined in two ways: (i) they can be given by the user, that is, the user application provides the necessary specifications to the file system for increasing the performance of the I/O routines, and (ii) they can be built by the MAPFS multiagent subsystem. This last feature is performed by *hints agents*. A case study of the use of hints is described in [29].

Files are stored finally in several servers, which constitute the server-side of the underlying architecture. The grouping of servers from a logical point of view in MAPFS is denominated *storage group* [28]. These groups take the role of data repositories and can be built applying several policies, trying to optimize the access to all the storage groups. We refer the reader to [28] for a more detailed description of the concept of storage group.

As previously mentioned, the use of a multiagent system implies coordination among their agents. The main goal of the agents cooperation is the interaction among such agents to achieve a common objective in a distributed system.

2.2. Related work

Nowadays, most of the frameworks are influenced by their environment. Indeed, the environment conditions affect their performance in a dynamic way. For this reason, the use of the agent technology is being widely used, since this paradigm is characterized by its adaptation to changing and dynamic environments. The agent paradigm is usually implemented on distributed systems.

A very important characteristic of agents is the cooperation. The agents cooperation can be made through a set of steps [16]:

- It is necessary to specify the *goals* of all the agents, that is, the descriptions of the desired state of the agents "world" or environment.
- Every agent must perform a set of *actions* in order to modify its state. Besides, *plans* containing precise instructions to achieve goals or objectives must be built.
- Every agent must have scheduled a set of *events*.
- According to this scheduling, the agent must run the plan.
- The *cooperation* is achieved using *shared plans*, that is, sharing the scheduling.

In a complex system, the interaction of several agents is required and, thus, a mechanism of communication between agents is necessary. For achieving agents communication and interoperability, it is necessary to use:

- A common language;
- common ideas about the knowledge agents interchange;
- capacity for interchanging this information.

With the aim of standardizing this way of communication, a common or standard language is used. KQML (Knowledge Query Manipulation Language) [6,1,8], is one of the most known agent communication languages. This language is composed of a set of messages, known as *performatives*, which are used for specifying agent communication elements. In [22], Labrou and Finin widely describe the KQML reserved performatives. Some of them are used in MAPFS, which takes advantage of agent properties to increase the I/O performance.

The idea of using agents to access data is not an innovative idea. Nowadays, a great number of agents platforms are deployed for accessing web databases. The web popularity has created the need for developing Web Distributed Database Management Systems (DBMSs), obtaining simple data distribution, concurrency control and reliability. However, DBMSs offer limited flexibility, scalability, and robustness. Some suggestions propose the use of agents to solve this problem [32,25].

With regard to file management, several approaches have been developed. MESSENGERS [4] is a system based on mobile agents used for the development and deployment of distributed applications, named *messengers*. This system is composed of a set of daemons distributed in every node. They are used for managing received agents, supervising their execution and planning where agents must be sent.