2. Multi Agent Grid Systems

Michał Wójcik

Gdansk University of Technology,
Faculty of Electronics, Telecommunication and Informatics,
Computer System Architecture Department
e-mail: michal.wojcik@eti.pg.gda.pl

Abstract

This chapter presents an idea of merging grid and volunteer systems with multi agent systems. It gives some basics concerning multi agent system and the most followed standard. Some deliberations concerning such an existing systems were made in order to finally present possibilities of introducing agents into the Comcute system.

Keywords*: Agent, Agent Grid, Comcute, Grid, Multi Agent System, Volunteer System.*

2.1. Agents and Multi Agents Systems

Emerging high-performance networks lead to popularizing distributed computing and introducing various computational paradigms like grid computing and volunteer computing. One of the developing architectures for distributed systems are multi agents systems which are based on autonomic agents.

An agent is a computer system that is situated in some environment, and is capable of autonomous actions in this environment in order to meet its designed objectives. The agent can perceive its environment and act upon it [21].

A Multi Agent System (MAS) is a system which consists of a number of agents. Agents are able to interact, mainly by exchanging messages possibly through some computer network infrastructure. In order to react successfully agents should be able to cooperate, coordinate and negotiate with each other [20].

2.1.1. Agents as Service Providers

A Service Oriented Architecture (SOA) can be regarded as a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains [16]. It needs to be pointed that SOA is not a concrete architecture or not even tool as well as framework. It is a set of guidelines that leads to a concrete architecture.

SOA guides in a process of creating and using business services during their lifecycle. It also provides conditions for the infrastructure which allows

different applications to exchange data and participate in business process irrespective to operating systems or programming languages [15].

Services are the main elements of systems implementing the SOA concept. By dictionaries those are defined as a performance of work by one for another [16]. OASIS additionally provides related ideas:

- the capability to perform work for another,
- the specification of the work offered for another,
- the offer to perform work for another.

It was said that agents exists in some environment. It may as well be environment of some kind of services. Those can be both, Web Services distributed on remote machines connected to the Internet and business services representing company activities mapped into computer system for the sake of simulations and automation. Agents can be treated as autonomous services providers and executors existing in such an environment. Moreover multi agent systems which assume communication and interaction between agents residing in the system, are suitable for this cause.

When one agent is going to invoke a service of another one there is a need for some kind of agreement between them. Such an agreement should be made on the basis of some negotiations and be profitable for both sides. This actions can be described by Service Level Agreement (SLA) which is contractual obligations between a service consumer and a service provider, which can represent guarantees of quality of service (QoS), non-functional requirements of a service consumer and promises of a service provider [10]. An SLA can contain the following components [1]:

- all sides involved into negotiation and execution, those besides contracting sides are supporting third parties such as monitoring, auditing, etc.,
- description of the service specifying functionality delivered under the agreement,
- service level objectives defining the service level of QoS parameters,
- penalty for cases when service provider fails to comply with the contract.

2.1.2. Agent FIPA Standard

As long as agents work in an isolated environment without interactions with external systems there is no need for considering some widely accepted norms and standards. The situation changes when there is a need for an interaction with existing systems, both agent-based and more classical like client-server. A large part of agent systems is projected with a view to cooperation between heterogeneous agents. Agents from different systems can cooperate in order to exchange some information, services or jointly achieve some goals.

The most significant agent standard is the one stated by Foundation For Intelligent Physical Agents (FIPA) which is a part of Institute of Electrical and Electronics Engineers (IEEE). The main goal of FIPA [3] is developing a set of standards concerning cooperation between heterogeneous agents originating form different agent systems. Among all interests in FIPA, those the most important are:

- abstract architecture — in case when a number of systems using different technologies to achieve some functional purposes is going to interoperate, there is a need of defining fundamental elements of these agent systems,
- management — specification for services concerning managing agents [4], some of them are:
    - Directory Facilitator (DF) — a yellow pages[1] service provided to other agents, agent can register in a catalog providing what type of service it is making accessible to other agents or query to find what services are offered by other agents,
    - Agent Platform (AP) — physical infrastructure where agents can be deployed, it consists of the machine with an operating system, an agent support software with agent management components and agents,
    - Agent Management System (AMS) — exerts supervisory control over the Agent Platform, it provides a white pages[2] service by maintaining agents' AID (Agent Identify), each agent has to register with an AMS to get a valid AID,
    - Message Transport Service (MTS) — communication services between agents on different platforms;
- communication — in order to provide understandable communication between heterogeneous agents FIPA proposes a semantic language (SL) [7] for messages recording and ontologies for providing vocabulary for representing knowledge.

For purposes of communication between agents FIPA defines Agent Communication Language (ACL) [6]. Each of messages exchanged between agents consists of fields defining sender, receiver and message type (performativity), where only that last one, defining communicative act, is mandatory. The communicative act [5] is an agent's action detailed by the message content. Those actions describes making requests, querying about inner state and performing negotiations (contact net). For the purposes of ACL messages content expression an SL language was defined [7] which with specified ontology defines syntax and semantics for the message.

2.2. Multi Agent System as a Grid

Grid concepts and technologies were initially developed to enable resource sharing within scientific collaborations. Those collaborations required to share not only databases but also software, computational resources and even some specialized instruments like telescopes and microscopes. Grids can be defined

[1] Terminology from phone directory, yellow pages contain entries concerning business.
[2] Terminology from phone directory, white pages contain entries concerning private persons.

as systems enabling coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [12].

Virtual organization is a set of individuals and/or institutions defined by some sharing rules. Those rules consider sharing computers, software, data, services and other resources based on the resource providers and consumers defining what is shared, who is allowed to share, and the conditions under which sharing occurs [9].

One of the typical and well known grid systems is Globus Toolkit. It allows for resources (data and computational power) management, its state monitoring, inter-nodes communication, providing security mechanisms and failure detection. It provides a set of services, protocols and interfaces supporting in a development of grid applications. Single application deployed in the Globus systems perceives the whole infrastructure as a local resources for which delivery to the specified node, responsible are platform level services [11].

Historically grids were focused on interoperable infrastructure and tools for secure and reliable resource sharing within dynamic and geographically distributed virtual organizations where agent systems have focused on the development of concepts, methodologies, and algorithms for autonomous problem solvers that can act flexibly in uncertain and dynamic environments in order to achieve their aims and objectives [8].

Some researches tries to connect grid and MAS paradigms leading to multi agent grid systems, sometimes simply called agent grids. The agent grid can be described by requirements at two levels: application and functional [14]. The application level defines requirements making it easier to build, maintain, scale, evolve, adapt and survive. Such a systems should be easily adaptable and scalable to large and small sizes and developed (evolved) by groups that do not need to know about each other. The functional requirements defines a unified, heterogeneous distributed computing environment in which computing resources are seamlessly linked. According to it agents can play the roles of applications whose computations can be distributed within the distributed computing environment, resources that can be used within this environment, and infrastructure components of this environment. Moreover agents can be used for performing load balancing, resources wrapping, and services broking.

Because of lack of the autonomy in classics grids, those are mostly predictable units. One of the most important properties of the agents is their autonomy, which could bring unpredictability into the picture. It must be stated here that autonomy does not mean that agent can do whatever its wants, but should be able to act without coordination from the superior unit. Agents must be designed in a such way that they always act on the sake of the whole system and autonomy should be used in means of failures handling and load balancing.

As an example of introducing agent into a grid, the AGrIP system can be mentioned. It introduces an agent environment in order to satisfy two requirements: (a) first, it integrates the resources and makes them available and useful, (b) second, it provides different kinds of agent grid common services [13]. The whole system is based on the MAGE, a multi agent environment for humanized systems which is compatible with the FIPA standard.

The system introduces several agents types required for building grid:

- DF — agent specified by the FIPA standard providing yellow pages service,
- GISA — Grid Information Service Agent contains static and dynamic information about compute resources and network performance between them,
- GRMA — Grid Resource Management Agent, provides capabilities to do remote job start and cancel as well as status checking,
- GSSA — Agent Security Service Agent, provides agent grid security service,
- DMA — Data Management Agent, responsible for access to remote data and its transfer management,
- Agent — a fundamental agent combining one or more service capabilities into a unified and integrated execution model.

AGrIP provides several toolkits using underlying agents on which top applications are built: Information Retrieval Toolkit, Data-Mining Toolkit, Case Base Reasoning Toolkit, Expert System Toolkit, Problem Solving Applications Toolkit, and Distributed Computing Toolkit.

Slightly different approach can be seen in the solution integrating JADE agent based system with the Globus middleware. JADE (Java Agent DEvelopment Framework) is another agent system compatible with the FIPA standard [19]. It tries to solve some issues like: (a) complicated resource brokering and management in existing Grid middlewares, (b) lack of interoperability between individual middlewares, and (c) too high expectations put on the potential user of the grid [18]. As a solution it proposes software agents combined with ontologies. The key functions of the system are: (a) helping the user to contribute its resources to the grid, and (b) helping the user to execute the jog withint the grid.

As a part of the solution, following agent types were introduced:

- LAgent — an agent representing user, provides an intelligent interface between the user and the Brokering System,
- CICAgent — an agent representing Client Information Center (CIC) being a central repository concerning existing agent teams,
- LMaster — an agent representing particular team, responsible for preparing offer and utilizing negotiations with LAgents,
- LMirror — an agent which mirrors/duplicates the LMaster agent in order to keep team functionality in case of LMaster failure.

LAgent searches for an agent team in order to join it as a worker or request some job specified by the user it represents. All the negotiations are based on the FIPA contract net protocol. When the team capable of performing specified job is found, the LAgent sends a binary representation of the job to LMaster, which forwards its to one of the workers.

Worker agent is equipped with a Job Executor module. It allows the worker to execute requested job. There were two concrete implementations prepared: (a) Simple Job Executor executing a job as a normal process within the worker machine and (b) Globs Job Executor passing the job to the Globus system.

Both described systems introduce multi agents systems as a part of grid infrastructure making an use of agents as a service providers. Moreover in both systems, the usage of FIPA standard make it possible to introduce an easy integration with other system and modules.

2.3. Multi Agents System as a Volunteer Application

Volunteer computing is another form of distributed computing which makes an use of a large number of distributed peers connected to the system as volunteers. Typical functionality of such a system is that: (1) a server decomposes some long task into small jobs, (2), volunteers download jobs from the server, (3) jobs are executed, (4) results are returned to the server, (5) the servers composes the results [17].

Most of the volunteer systems are centralized ones and their structure follows the star topology. This can be troublesome in cases when the main server is overloaded or some failure occurs.

A PPCV system is an example of volunteer computing system made with distributed agents. The system is made with distributed volunteers (nodes), where each of them is an agent container, that is a place where agent can residue.

Each of the nodes can have multiple neighbours, where neighbours are nodes with direct communication link between them. In the opposite to standard volunteer systems using star topology this one uses mesh topology which can be presented by complete or non complete graph (depending on how many connections were made during deploying new nodes). The PPCV implements are required volunteer actions, that is: (a) job decomposition, (b) job remote execution, and (c) result composition [22].

System is made with the following agent types:

- Scheduler Agent — responsible for job scheduling by managing particular node and communication with other ones,
- Job Agent — responsible for executing requested jog.

There is not specialized agent or system module responsible for decomposing jobs into smaller ones and sending them to particular agents. Instead of that if there is a need for decomposition, the Job Agent clones itself and negotiate with its copy about part of the job to be done by each of them. If there is still need for decomposition, those two copies can clone themselves independently and so on. This means that jobs passed to the PPCV system must be decomposable. As for the results composition it is made by merging cloned agents. In order to make an use of the great number of available volunteers cloned agents may migrate to another container, make computations and then come back in order to merge. The whole job scheduling inside the PPCV system is decentralized and realized by Scheduler Agents by communication between neighbours which on the basis of the perceived environment (node) make decision about accepting or passing job.

2.4. Possible Usage of Agents in the Comcute System

Above sections show that multi agent systems can be successfully used in high scale distributed systems. One of such a system is, still being in development, Comcute grid. In details it is grid system using computational power of volunteers [2]. It is characterized by the high reliability requirements and easy expansion by attaching new volunteers. In order to meet the requirements the distribution layer is divided into a number of equal peers instead of using one central server. In details the system was divided into four layers:

- Z — layer of the service requester,
- W — contains a number of servers responsible for dividing task and results gathering and verification,
- S — proxy servers between W and I layers, responsible for communication with volunteers,
- I — layer containing volunteers machines available while performing computations.

There are some aspects where agents can be introduced in order to achieve new functionality or possibly improve performance. Typically, in volunteers applications where peers are not autonomic, there is no data size negotiations. Task divider sends specified amount of data and changes sizes of the next parts on the basis of volunteer computation speed. In situation when there would be an agent in I layer representing particular volunteer and an agent in S layer representing task divider there would be a place for data size negotiation depending on the volunteer's system load and how much of its performance it is willing to share. Thank to that agent is perceiving its environment (volunteer's machine system) and its preference concerning performance in order to tune data received from S layer.

Agents implementation would also help in introducing payable services. This would require to implement agents in all the layers. Lets say that the requester from Z layer is willing to pay some amount of money for performing computations in the grid system. Then the grid owner is willing to pay its volunteers for sharing their resources. Firstly an agent from Z layer negotiates with the on from W layer in order to decide what is need to be done, with what performance (how fast) and how much it will be cost. A SLA containing those QoS parameters is done. Then agents from S layer negotiates with those from I layer in order to make a SLA concerning how much of volunteer's performance will be shared and what will be the fee.

In most of the volunteer's systems there is a problem that particular volunteer can disconnect in any time. This causes that some computations are made on the same data by different peers in order to introduce reliability. In situation when there would be introduced fees for sharing computation power, as a part of the SLA there could be established some penalties for disconnecting in the middle of computations of particular data package. This could improve the efficiency of the whole system by reducing a number of backup computations.

In the most complicated form, agents would be introduced in all four layers, but negotiations would be carried on only inside pairs of the Z-W and S-I layers. Because layers W and S belong to the inner system, there is no need for negotiations. That does not mean that there is no place for a SLA agreement in order to provide QoS parameters.

While implementing agents, an usage of standards should be considered. The most commonly used is the FIPA standard, because of its standardization of communication between agents. It allows to express all requited communications acts, like: requests, querying for agents' inner state and negotiations (contract net).

References

1. Alain Andrieux, Karl Czajkowski, Asit Dan, Kate Keahey, Heiko Ludwig, Toshiyuki Nakata, Jim Pruyne, John Rofrano, Steve Tuecke, and Ming Xu. Web Services Agreement Specification (WS-Agreement). https://forge.gridforum.org/projects/graap-wg/, 2012-04-02.
2. Jerzy Balicki and Jarosław Kuchta. Obliczenia rozproszone w systemach komputerowych o architekturze klasy grid. Wydawnictwo Politechniki Gdańskiej, Gdańsk, 2012.
3. FIPA – Foundations for Intelligent Physical Agents. Standard Status Specifications. http://www.fipa.org/repository/standardspecs.html.
4. FIPA – Foundations for Intelligent Physical Agents. FIPA Abstract Architecture Specification, December 2002.
5. FIPA – Foundations for Intelligent Physical Agents. FIPA Communicative Act Library Specification, December 2002.
6. FIPA – Foundations for Intelligent Physical Agents. FIPA Message Structure Specification, December 2002.
7. FIPA – Foundations for Intelligent Physical Agents. FIPA SL Content Language Specification, December 2002.
8. Ian Foster, Nicholas R. Jennings, and Carl Kesselman. Brain meets brawn: Why grid and agents need each other. In Proceedings of the 2005 conference on Towards the Learning Grid: Advances in Human Learning Services, pages 28–40, Amsterdam, The Netherlands, The Netherlands, 2005. IOS Press.
9. Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl., 15(3):200–222, August 2001.
10. Qiang He, Jun Yan, Ryszard Kowalczyk, Hai Jin, and Yun Yang. Lifetime service level agreement management with autonomous agents for services provision. Inf. Sci., 179(15):2591–2605, July 2009.
11. Bart Jacob, Luis Ferreira, Norbert Bieberstein, Candice Gilzean, Jean-Yves Girard, Roman Strachowski, and Seong (Steve) Yu. Enabling applications for grid computing with globus. IBM Corp., Riverton, NJ, USA, first edition, 2003.

12. Carl Kesselman and Ian Foster. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, November 1998.
13. Jiewen Luo and Zhongzhi Shi. Distributed system integration in agent grid collaborative environment. In Integration Technology, 2007. ICIT '07. IEEE International Conference on, pages 373 –378, march 2007.
14. Frank Manola and Craig Thompson. Characterizing the Agent Grid. Technical Report 990623, Object Services and Consulting, Inc., June 1999.
15. Eric Newcomer and Greg Lomow. Understanding SOA with Web Services (Independent Technology Guides). Addison-Wesley Professional, December 2004.
16. OASIS. Reference Model for Service Oriented Architecture. http://www.oasis-open.org/
committees/download.php/16587/ wd-soa-rm-cd1ED.pdf.
17. Luis F. G. Sarmenta. Volunteer Computing. PhD thesis, Massachusetts Institute of Technology, 2001.
18. Mehrdad Senobari, Michal Drozdowicz, Marcin Paprzycki, Wojciech Kuranowski, Maria Ganzha, Richard Olejnik, and Ivan Lirkov. Combining a jade-agent-based grid infrastructure with the globus middleware initial solution. In Proceedings of the 2008 International Conference on Computational Intelligence for Modelling Control & Automation, CIMCA '08, pages 895–900, Washington, DC, USA, 2008. IEEE Computer Society.
19. Telecom Italia Lab. Java Agent DEvelopment Framework Documentation. http://jade.tilab.com/doc/index.html.
20. Michael Wooldridge. Introduction to MultiAgent Systems. John Wiley & Sons, June 2002.
21. Michael J. Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. The Knowledge Engineering Review, 10(2):115–152, 1995.
22. Zhikun Zhao, Feng Yang, and Yinglei Xu. Building P2P Volunteer Computing System Using Agents. In Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on, pages 1 –5, December 2009.