Information Flow and Mirroring in an Agent-Based Grid Resource Brokering System

Maria Ganzha[1], Marcin Paprzycki[1], Michal Drozdowicz[1], Mehrdad Senobari[2], Ivan Lirkov[3], Sofiya Ivanovska[3], Richard Olejnik[4], and Pavel Telegin[5]

[1] Systems Research Institute, Polish Academy of Science, Warsaw, Poland
[2] Tarbiat Modares University, Tehran, Iran
[3] Institute for Parallel Processing, Bulgarian Academy of Sciences, Sofia, Bulgaria
[4] University of Sciences and Technologies of Lille, Lille, France
[5] SuperComputing Center, Russian Academy of Sciences, Moscow, Russia

Abstract. We are developing an agent-team-based Grid resource brokering and management system. One of issues that has to be addressed is team preservation through mirroring of key information. We focused our attention on information generated within the agent team. In this paper we discuss sources of information generated in the system and consider which information should be mirrored to increase long-term survival of the team.

1 Introduction

In our work, to develop an agent-team-based high-level intelligent Grid middleware, we have established when access to information, generated and stored in the team, is needed. For instance, when a team leader (the *LMaster* agent) receives a *Call for Proposals* (*CFP*) message asking about conditions of executing a job, its response (an offer, or a rejection) is to be based on knowledge of the client and of market conditions (see [7, 11]). Such knowledge is to be based on data collected during past interactions with (potential) client(s). Since we assume that a *Global Grid* is a highly dynamic structure [12], in which nodes can crash, it is important to assure that the team knowledge will not be lost if its leader crashes. Therefore, in [8, 13] we have suggested utilization of an *LMirror* agent, which should keep copy of information necessary to prevent team disintegration, in the case of the *LMaster* crash.

The aim of this paper is to point to sources of information useful for an agent team, and to discuss which information should be mirrored and when. To this effect, in the next section, we present a brief overview of the proposed system, followed by arguments for the need of information mirroring. We follow with description of sources of information to be mirrored. In each case we discuss when and where this information should be persisted.

2 System Overview

To discuss the birds-eye view of the system and the two main processes taking place in it (*Worker* joining a team and team accepting a job to be executed) we

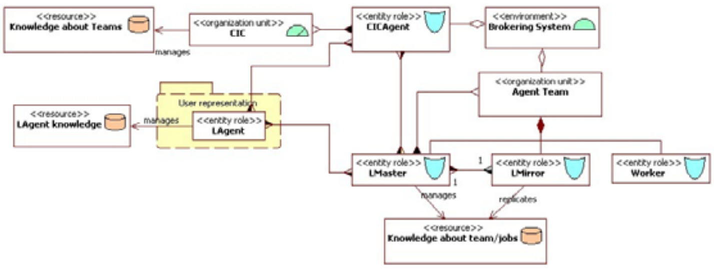

Fig. 1. AML social diagram of the proposed system

will utilize an AML Social Diagram [5] in Figure 1. In our approach, agents work in teams. Each team is supervised by the *LMaster* agent. Agent teams utilize services of the *Client Information Center* (represented by the *CIC* agent), to advertise their resources, and *Workers* they are seeking. Teams consist of the *LMaster*, *Workers*, and the *LMirror* agent (which stores copy of information necessary to persist the team when the *LMaster* crashes). When the *User* is seeking team to execute its job, it specifies job constraints to the *LAgent*. The *LAgent* contacts the *CIC* and obtains the list of teams that can execute its job. Next, it utilizes trust information and the FIPA Contract Net protocol [15] to either find a team, or establish that no team fulfills *User*-specified conditions. When the *User* would like its computer(s) to become *Worker*(s) in a team, it specifies conditions of joining. The *LAgent* interacts with the *CIC* to obtain a list of teams that are seeking *Workers*. Next, it utilizes trust information and the FIPA Contract Net to establish if a team to join can be found. In both cases, trust information is a part of the *LAgent knowledge*. Similarly, when responding to the CFP from an *LAgent*, the *LMaster* utilizes *Team knowledge* (see [11,13]).

3 Need for Information Mirroring

While collecting knowledge by the *LAgent*, and utilizing it in decision making are interesting, we focus on processes taking place within the agent-team. Recall our assumption that in the *Global Grid* any node can disappear at any time and for an unspecified time. Obviously, this applies also to the *LMaster* agent. Let us now see what happens when the *LMaster* becomes unavailable (e.g. due to the Internet access failure), and there is no information mirroring. Observe that the *LMaster* is the *only* agent that *Users* (their *LAgents*) contact. Note that in the case of contract negotiations, there would be no long-term consequences in the case of a short-term disappearance of the *LMaster* (while the team would loose a potential contract, it should be able to manage the team). More serious is the fact that the *LMaster* is the only gateway through which jobs are received and results send back. The *LMaster* is also the only agent that *Workers* know about. In the case of a more permanent disappearance of the *LMaster*, damage could be irreparable; e.g. *Workers* would not be able to send results of completed tasks and/or receive new tasks and would leave the team and not come back (see also [11]). Finally, observe that in this situation all knowledge collected by the *LMaster* becomes unavailable (and may be potentially lost forever).

In response to these challenges we have decided to use data mirroring. In [6, 12, 13] we have proposed that an *LMirror* agent should store information needed to keep the team alive and competitive. Furthermore, in [13] we have described restoring the *LMaster* and the *LMirror* in the case when either one of them crashes. Let us now, consider three issues: (1) what information should be mirrored, (2) when should it be mirrored, and (3) where should it be mirrored.

4 What Should Be Mirrored, When and Where?

4.1 Team Member Data

Let us start from information about members of the team. When a *Worker* joins the team, the *LMaster* obtains the following information:

1. *ID* of the *LAgent* joining the team. We have assumed that each agent utilizing our system has to be registered within it (function of the *CIC*). It is thus possible to check the registration of the potential *Worker*, before accepting it into the team). This can be treated as a mini-security measure that can be extended as the system develops.
2. *Available resources.* In the CFP the *LAgent* includes (ontologically demarcated; [10]) information about resources offered for the team. This information about each *Worker* is to be, among others, used for job scheduling [14].
3. *Details of the contract.* In [13], we have suggested that the contract proposal should contain: (a) specification of the time of availability, and (b) length of contract. However, results of the MiG project (see papers available at [16]) indicate that more complex data-sets can be involved in contract negotiations and thus in the contract.

Out of these three items, the ID of the *Worker* has to be mirrored immediately. Without this information it will be impossible to contact this *LAgent* in the future. The remaining two could be recovered from the *Worker*. However, since information about item (1) is to be send to the *LMirror* regardless, information about items (2) and (3) may be included as well. In this way crashing of the *LMaster* will not disturb team *Worker*s with unnecessary exchanges of messages with the new *LMaster*. Note that the total amount of information mirrored here is proportional to the number of *Workers* in the team and thus should not be very large.

4.2 Job Contracts and Their Execution

In [8, 7, 6] we have specified a limited set of parameters describing the job contract. In the CFP *User* could specify: (a) job start time, (b) job end time, and (c) resource requirements, while the response contained the price proposed by the *LMaster*. Obviously, additional parameters may be added, but this does not affect material presented here. As soon as the contract is signed all data concerning it has to be mirrored. Furthermore, loosing information about a contract would mean that it would either not be completed, or results would have no *User* to be send to (the latter would happen in the case, when information was lost while the task was already being executed by one of the *Workers*). In both cases, the team would rapidly loose its reputation and potential future contracts [11].

Signing a contract means that files pertinent to the job will be send to the *LMaster* and have to be kept and mirrored until job is completed. As soon as the job is completed, its results have to be sent back to the *User*. However, they should be also kept by the *Worker* until it receives a confirmation from the *LMaster* that they were successfully delivered to the customer. At the same time the *LMaster* keeps the second copy of results. In this way we, again, have two copies of sensitive material available in the system.

Information about job completion and successful delivery of results has to be stored. It is to be useful, first, for job scheduling (see, for instance, [14] and references collected there). Second, for trust management (see [11] and section 4.3), and third, for contract disputes. Interestingly, only fact that job has been completed and its results successfully delivered to the customer, has to be mirrored immediately (it describes the current status of the team). Information needed for future job scheduling may be mirrored "later". Loosing data about execution times of some jobs (due to the crash of *LMaster*) does not pose a threat to the existence of the team. Similarly, some information needed for trust management may be lost without damaging the team. Finally, note that scheduling and trust related information may be sent to the team data warehouse instead of the *LMirror* (see section 4.4).

4.3 Trust Information

Thus far we have dealt mostly with information that has to be mirrored immediately. Now we devote our attention to the remaining data generated in the system. In [11] we have considered trust management in the system. As noted above, here, we are interested only in relationships between the *LMaster* and its *Workers*.

Following the discussion presented in [11] we assume that contract between *Worker* and the team is based on paying the *Worker* for availability (not for actual work done). It was also stipulated that *Worker* will have a right to not to be available for a limited number of times during its contract. Here, we can use the "pinging-mechanism" described in [6] to monitor *Worker* availability. Under these assumptions, for each *Worker*, we will collect information how many times (a) it *fulfilled the contract* ($\#fc$) — was available when it was expected to be available; (b) violated the contract ($\#vc$) — was not available when it was supposed to be; and (c) did more than contract required ($\#ac$) — was available even though it could have been gone. This combined data will be stored, for each *Worker*, as a triple ($\#fc, \#vc, \#ac$). While this information is in a cumulative form, we have also access to detailed information about results of each "pinging-procedure," and each assigned and completed job. Since the latter data is not proportional to the number of *Workers*, but to the number of processed jobs, we consider it separately in section 4.4.

Obviously, while the trust-related information has to be mirrored, some loses (e.g. missing the fact that *Worker_31B* successfully fulfilled contract *324_55B_3*) are not a big problem. Therefore, it is enough to update information about trust related issues in a digested format at predefined time-intervals; e.g. once a day. Frequency of updates depends on nature of jobs that a given team is executing and on the type of *Worker* contracts. If a team is executing time-consuming jobs and its members have long-term contracts, then update of trust information can occur much less frequently than in the case when large number of short jobs are completed by a team consisting of *Workers* with short-term contracts. We believe that a good approximation of the right time to send trust information to the *LMirror* is when, on average, each *Worker* has completed one contract. Note that the *LMaster* can adaptively adjust time between updates (responding to changing jobs and contracts of its team).

4.4 Volume Data Collection and Storage

Most information considered thus far was relatively small in volume — of order of the size of the team or of the number of currently contracted jobs. There are however important data sets that grow over time and thus can become very large. First, data generated during contract negotiations (either job execution, or team joining), needed to establish market conditions and results in team behavior adaptivity (see also [6,11]). Second, information about execution time of completed jobs, used for advanced scheduling techniques (see [14] and references collected there for more details). Third, detailed information about behavior of each worker (e.g. history of execution of each job, results of pinging sessions, etc.). This information is crucial not only for trust management, but also to build a realistic and comprehensive economy-based jobs scheduling model.

Let us consider *LAgent–LMaster* negotiations (the remaining cases follow the same general pattern and are omitted for lack of space). Even though in [1,2,3,4] a large variety of contract negotiation mechanisms have been listed, in our work, for both job execution, and joining a team negotiations, we utilize the FIPA Contract Net Protocol [15]. The primary reason is that while allowing for calls for proposals with practically unlimited variety of constraints, it generates the contract (or establishes impossibility of an agreement) within a single round of negotiations. This also reduces the total amount of information that needs to be stored for further use (compare, for instance, with the amount of data generated during any of the multi-round auction mechanisms; [9]). Now, information about a single job contract negotiation consists (in the minimal case) of:

– content of the CFP — *LAgent* ID, hardware and/or software sought; "start time"; time to be contracted / or was it an open-ended contract?
– content of the response — the proposed price; etc.;
– result of the negotiation — success/failure.

Similarly, in the case of negotiation concerning a *Worker* joining the team, at least the following information can be stored:

– content of the CFP — including information about available hardware and software; length of contract and availability,
– content of the response — the proposed price,
– result of the negotiation — did the *Worker* join the team, or not.

Interestingly, in all cases loosing some of the data may decrease competitiveness of the team, but the process is characterized by graceful degradation. Therefore it is possible to perform updates in a digested fashion at times of reduced utilization of the system. However, observe that, as time passes, the total volume of collected information increases rapidly. While it would be possible to store only a specific amount of the most current data, this would reduce utility of data mining techniques (long-term trends may be lost). Therefore, it is important to preserve all historical data and a data warehouse may be a solution.

It should be obvious that in these cases mirroring utilizing the *LMaster* and the *LMirror* pair can be very costly. First, an increasing volume of data would have to be stored by each one of them. Second, large amounts of data would have to be passed from the *LMaster* to the *LMirror*. Naturally, since we utilize single-round negotiations, passing updates in a compressed and digested form would reduce the burden somewhat, but this would help only in a short-term. Finally, recovery of the crashed *LMaster* (or *LMirror*) would be extremely resource consuming. In this case, as a part of the recovery, an entire data warehouse would have to be transferred. For a large team that exists for a long time period (and our assumption is that good teams will be large and will exists for a long time), this would mean necessity to transfer hundreds of gigabytes of data. Such transfer would have to happen each time during recovery of the crashed managerial agent. Since, as suggested in [13], recovery of one crashed managerial agent stops the other one from doing anything else, it also stops *all* managerial activities in the team. Therefore, such approach seems infeasible in a long run.

Considering this, based on the current trend of IT infrastructure outsourcing and specialization, we propose the following solution. The team data warehouse will be outsourced for storage to a team within the system that *specializes* in data warehousing. This is similar to a popular e-commerce scenario, where merchants contract software companies to design, implement, and run infrastructure of their e-stores, while software companies contract storage companies to actually store the data (this solution facilitates quality assurance for data preservation).

Note that, utilization of a data warehouse allows to completely redefine the roles of the *LMaster* and the *LMirror*. Here, the *LMaster* stores information in the contracted "storage facility". Only data needed for team operation, e.g. list of team members, is kept locally (and "mirrored" in the data warehouse; not with the *LMirror*). The role of the *LMirror* is not to store team data, but to store procedures for: (a) checking existence of the *LMaster*, and (b) replacing an *LMaster* in the case when it crashes. In this way the *LMirror* can work as all other *Workers* (it will not be burdened by the mirroring procedures), which may also simplify the overall management and economic structure of the team.

5 Concluding Remarks

The aim of this paper was to discuss issues involved in information mirroring in an agent-based Grid resource management system. We have focused our attention on information generated within the agent team and considered four important cases: (1) team data, (2) job contracts and their execution, (3) trust-related information, and (4) other sources of large volume information. We have established that we deal with two main situations: (a) small-volume data that has to be mirrored immediately, and (b) large volume data that may be mirrored infrequently. Further analysis indicated that collection of large volume data may be best achieved through utilization of a contracted data storage facility. This latter solution is our solution of choice and we plan to utilize it in our system.

Acknowledgments

Work of Maria Ganzha and Michal Drozdowicz was supported from the "Funds for Science" of the Polish Ministry for Science and Higher Education for years 2008-2011, as a research project (contract number N516 382434). Collaboration of the Polish and Bulgarian teams is partially supported by the *Parallel and Distributed Computing Practices* grant. Collaboration of Polish and French teams is partially supported by the PICS grant *New Methods for Balancing Loads and Scheduling Jobs in the Grid and Dedicated Systems.* Collaboration of the Polish and Russian teams is partially supported by the *Efficient use of Computational Grids* grant. Work of Marcin Paprzycki and Sofiya Ivanovska was supported in part by the National Science Fund of Bulgaria under Grants No. D002-146 and D002-115.

References

1. Abraham, A., Buyya, R., Nath, B.: Nature's heuristics for scheduling jobs on computational grids. In: 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), pp. 45–52 (2000)
2. Buyya, R., Abramson, D., Giddy, J., Stockinger, H.: Economic models for resource management and scheduling in grid computing. Concurrency and Computation: Practice and Experience 14(13-15), 1507–1542 (2002)
3. Buyya, R., Abramson, D., Venugopal, S.: The grid economy. Proceedings of the IEEE 93(3), 698–714 (2005)
4. Buyya, R., Giddy, J., Abramson, D.: An evaluation of economy-based resource trading and scheduling on computational power grids for parameter sweep applications. In: Second Workshop on Active Middleware Services (AMS 2000), p. 221. Kluwer Academic Press, Pittsburgh (2000)
5. Cervenka, R., Trencansky, I.: Agent Modeling Language (AML): A Comprehensive Approach to Modeling MAS. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhauser, Basel (2007)
6. Dominiak, M., Ganzha, M., Gawinecki, M., Kuranowski, W., Paprzycki, M., Margenov, S., Lirkov, I.: Utilizing agent teams in grid resource brokering. International Transactions on Systems Science and Applications 3(4), 296–306 (2008)

7. Dominiak, M., Ganzha, M., Paprzycki, M.: Selecting grid-agent-team to execute user-job — initial solution. In: Proc. of the Conference on Complex, Intelligent and Software Intensive Systems, pp. 249–256. IEEE CS Press, Los Alamitos (2007)
8. Dominiak, M., Kuranowski, W., Gawinecki, M., Ganzha, M., Paprzycki, M.: Utilizing agent teams in grid resource management — preliminary considerations. In: IEEE J. V. Atanasoff Conference, pp. 46–51. IEEE CS Press, Los Alamitos (2006)
9. Drozdowicz, M., Ganzha, M., Paprzycki, M., Gawinecki, M., Legalov, A.: Information Flow and Usage in an E-Shop operating within an Agent-Based E-commerce system. Journal of Sibirian Federal University, SFU, Krasnoyarsk (in press)
10. Drozdowicz, M., Ganzha, M., Paprzycki, M., Olejnik, R., Lirkov, I., Telegin, P., Senobari, M.: Ontologies, agents and the grid: An overview. In: Topping, B., Ivanyi, P. (eds.) Parallel, Distributed, and Grid Computing for Engineering. Computational Scientce, Engineering and Technology Series, vol. 21, pp. 117–140. Saxe-Coburg Publications, Stirligshire (2009)
11. Ganzha, M., Paprzycki, M., Lirkov, I.: Trust management in an agent-based grid resource brokering system — preliminary considerations. In: Todorov, M. (ed.) Applications of Mathematics in Engineering and Economics'33. AIP Conf. Proc., vol. 946, pp. 35–46. American Institute of Physics, College Park (2007)
12. Kuranowski, W., Ganzha, M., Gawinecki, M., Paprzycki, M., Lirkov, I., Margenov, S.: Forming and managing agent teams acting as resource brokers in the grid — preliminary considerations. International Journal of Computational Intelligence Research 4(1), 9–16 (2008)
13. Kuranowski, W., Ganzha, M., Paprzycki, M., Lirkov, I.: Supervising agent team an agent-based grid resource brokering system — initial solution. In: Xhafa, F., Barolli, L. (eds.) Proceedings of the Conference on Complex, Intelligent and Software Intensive Systems, pp. 321–326. IEEE CS Press, Los Alamitos (2008)
14. Senobari, M., Drozdowicz, M., Ganzha, M., Paprzycki, M., Olejnik, R., Lirkov, I., Telegin, P., Charkari, N.: Resource management in grids: Overview and a discussion of a possible approach for a agent-based middleware. In: Topping, B., Ivanyi, P. (eds.) Parallel, Distributed, and Grid Computing for Engineering. Computational Scientce, Engineering and Technology Series, vol. 21, pp. 141–164. Saxe-Coburg Publications, Stirligshire (2009)
15. Welcome to the FIPA, `http://www.fipa.org/`
16. Projekt Minimum intrusion Grid, `http://www.migrid.org/MiG/Mig/published_papers.html`