Agent-based Negotiation for Resource Allocation in Grid

L. Nassif[1], J. M. Nogueira[1,2], M. Ahmed[3], R. Impey[3], A. Karmouch[4]

[1]Federal University of Minas Gerais (UFMG)
Av. Antônio Carlos 6627 - 31270-010 - Belo Horizonte – MG - Brazil

[2] In sabbatical period at universities of Evry and UPMC/Paris6/LIP6 - France

[3]National Council Research Canada (NRC)
1200 Montreal Road – K1A R06 - Build M50 – Ottawa - Canada

[4]University of Ottawa - SITE
161 Louis Pasteur – K1N 6N5– Ottawa - Canada

{lilian,jmarcos}@dcc.ufmg.br,{Mohamed.Ahmed,Roger.Impey}@nrc-cnrc.gc.ca, karmouch@site.uottawa.ca

Abstract. *Grid technology allows the sharing of resources within groups of individuals or organizations. A job submission in grid initially requires the identification of a list of servers that meet a certain job description. After, it is necessary to select the best server from this list. None of current researches associates the choice of the server with the service delivery conditions. In order to incorporate quality to the grid service it is important to know when the job will finish and what are the cost and quality factors involved. We present here a Multi-Agent System that chooses the best place to run a grid job by making use of negotiation. The prediction of job execution is achieved with case-based reasoning technique and the negotiation flexibility is delimited by resource policies. Our approach models different forms of negotiation, identified as multi-issue, bilateral and chaining negotiations.*

1. Introduction

The grid computing is a large-scale computing infrastructure that allows the sharing of resources among individuals or institutions. Grid middleware such as Globus [Globus 2004] implements core features such as security, information service and data management. Nevertheless, there are some aspects that can be improved in grid using specific tools. We are particularly interested in the resource allocation problem in Grid.

A grid user can submit a job directly to a resource or can use a middleware that chooses a suitable resource for his job. Although some approaches in [Czajkowski et al. 2002][Raman et al. 1998][GRMS 2004] achieve the same goal, that is, to allocate a job to a resource, they lack a negotiation process to give intelligence to the resource allocation problem. This negotiation process should allow users and resources to define the conditions about how the grid service should be delivered. Therefore, the selection of the server is not only a question of mapping job description to resource availability, but also it should take into account the conditions about price, performance and quality of service defined in the negotiation process for each server that is a candidate to run a job.

We present here a middleware for Grid that makes use of agent technology as the main mechanism for grid resource negotiation in the job submission process. The agent technology has features well fitting for distributed communication, and is particularly robust for the negotiation and migration processes. We consider that such characteristics are important fundamentals for a suitable solution to the problem in question. In this paper, we highlight the negotiation function in the job submission process in Grid. The negotiation process in our approach is bilateral (consumers and providers), of multi-issues (price, quality and time schedule) and chaining (network and servers). We complemented our framework with performance prediction and monitoring modules.

The paper is organized as follows: An overview of related work is presented in section 2. In section 3, the middleware is described, highlighting the negotiation processes. We present our final considerations in section 4.

2. Related Work

The work in [Czajkowski et al. 2002] presents the development of a protocol for negotiation called Service Negotiation and Acquisition Protocol (SNAP). SNAP uses three types of Service Level Agreements (SLAs): Task, Resource, and Binding. This work, like ours, considers SLA, but with a different approach. It establishes agreements by using a scheduler community. Our work, on the other hand, treats negotiations in a decentralized fashion by using agents. The SLA representation of SNAP is very superficial, it focus on a weaker form of agreement and no cost model is associated.

Condor [Raman et al. 1998] is a system for compute-intensive jobs. A framework, called Matchmaking, is developed for the purpose of mapping jobs to resources. Although the matching process is very well defined, another Condor method, called Claiming, lacks a more robust way to negotiate. Our work shows how the negotiation process between consumers and providers can be improved.

GRMS [GRMS 2004] is a meta-scheduling system. GRMS does not incorporate SLA guarantees for the service delivery. No prediction for the job time execution is provided. In our solution we can provide guarantees expressed in SLAs.

Works in [Czajkowski et al. 2002][Raman et al. 1998][GRMS 2004] do not present a suitable negotiation process. We believe that it is a way to give flexibility to the job submission by adding guarantees for the service delivery. With negotiation we can associate a job to a faster, cheaper, and higher quality resource.

3. Multi-Agent System

An overview of our middleware architecture, called Mask, is illustrated in figure 1. It consists of negotiation, migration, and interface modules. In the negotiation module are agents and functions involved in the negotiation process. The migration module completes the negotiation module by matching the SLAs established by agents, by deciding the place to run the job, and by submitting the job. The OGSA interface module has functions that allow integrating agents into the grid environment. Mask has different implementations for inter and intra domains. This is represented in the architecture by two different hierarchical levels associated with global and local security questions. Mask has been developing with Jade technology [Jade 2004], a robust environment for agent deployment and it has been integrating with Globus, the de facto standard for grid. The Mask architecture modules are described in the next sub-sections.

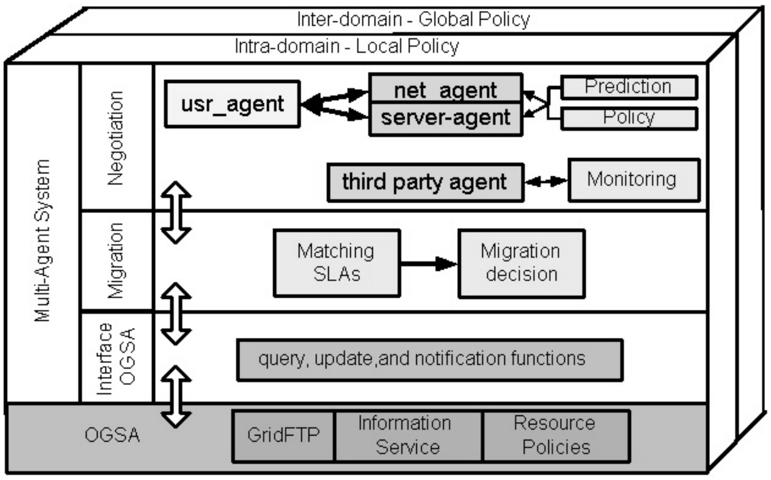


Figure 1. Multi-Agent System Architecture

In the system flow, the user configures the user_agent by providing information about negotiation limits (price, quality, and time schedule), by describing job arguments, and by giving preference among price, quality and time schedule. The user_agent receives a list of candidate servers from the information service component of Globus called Monitoring and Discovery Service (MDS), and selects a group from such list to negotiate with. A negotiation between user_agent and server_agent is started. The server_agent negotiates according to local resource policies. The objective of the negotiation is to establish a SLA. The user_agent should also negotiate with network grid resources in order to transfer data. At the end of the negotiations all SLAs are combined and a group of SLAs is chosen (details in section 3.2). This choice must meet user's preferences concerning price, quality and time schedule. Agents transfer data to the selected server and monitor SLAs. A third-party_agent manages all SLAs.

3.1 Negotiation Module

In the negotiation module there are different forms of negotiation, called bilateral, multi-issue and chaining negotiation. A bilateral negotiation occurs when there are only two parts involved. In a multi-issue negotiation, different aspects are negotiated such as price, quality and time schedule. The negotiation between user_agent and network_agent is also called chaining negotiation. A chaining negotiation occurs after the negotiation between user_agent and server_agent has finished.

3.1.1 User_agent perspective in the multi-issue bilateral negotiation

All negotiations in the system are started by the user_agent perspective. The user_agent selects some servers, which meet job description, to negotiate with. It is defined as RS= (L, Perf, NS, PS, CS) where RS are resources selected to negotiate with; L is the server location related to data location, where L = {LN,RN} for Local Network (LN) or Remote Network (RN), in such a way that, if the server is in the same local network where the data is available, then L=LN; Perf is the machine's performance; NS is the number of SLAs established for the same time interval requested; PS is the percentage of servers to be selected; and CS is the set of candidate servers. The server's selection is done using a utility function. First, it is necessary to associate weights (p1,p2,p3) to characteristics; after, the utility function U = (L*p1 + Perf*p2 + NS*p3) is calculated and the result is sorted; finally, PS*CS servers (with higher utility function from the set) are selected to negotiate with.

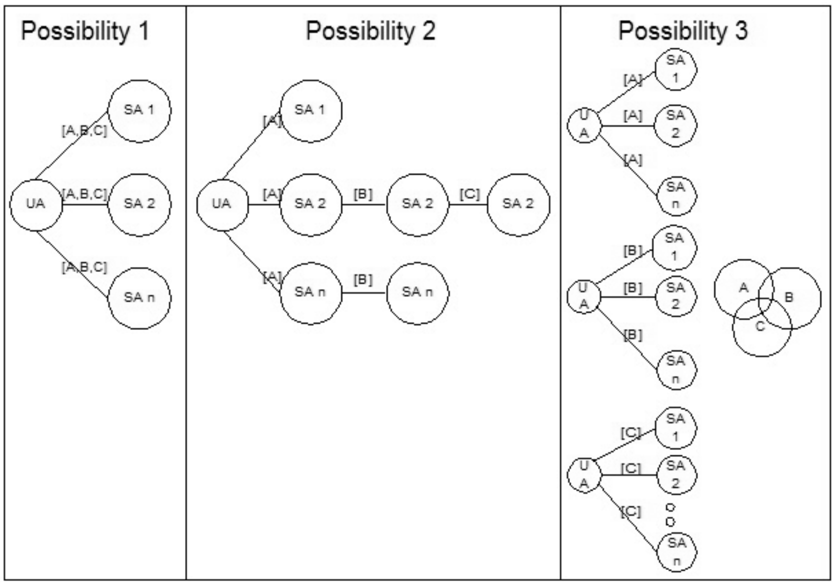


Figure 2. Multi-issue (A,B,C) negotiation between user_agent(UA) and server_agent(SA)

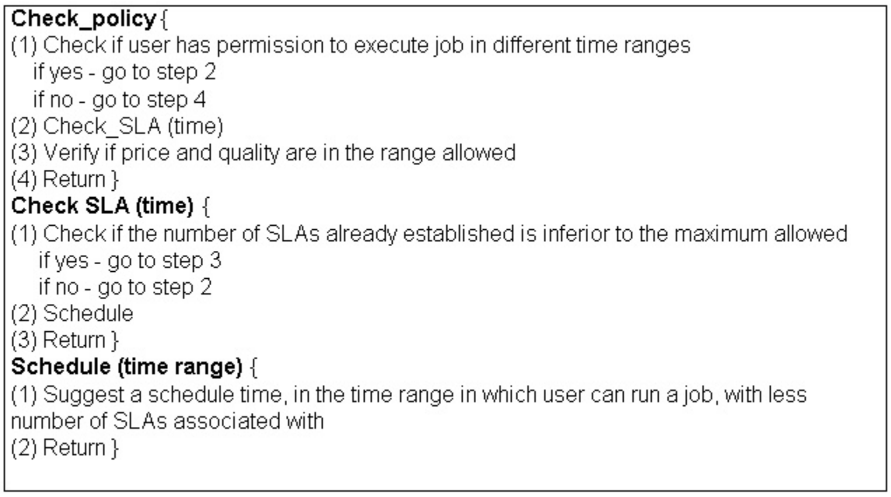
A bilateral negotiation of multi-issues can be performed in many ways (figure 2), such as: 1) negotiate the multi-issue as a package [Zhag and Lesser 2002]; 2) negotiate each issue separately, in sequence, discarding some proposals; and 3) negotiate each issue separately, in parallel, checking intersections of outcomes for each issue. The negotiation of multi-issue as a package (possibility 1) is the most appropriated, since we are considering that the negotiation of one issue can influence the negotiation of others.

User_agent sends proposals to several server_agents and decides about the best counterproposal received back. After, the user_agent proceeds with the chaining negotiation.

3.1.2 Server_agent perspective in the multi-issue bilateral negotiation

From the server_agent perspective, the negotiation is a reaction process. For each proposal sent by the user_agent, the server_agent checks if the proposal can be accepted, otherwise it chooses the best time to run the job from the point of view of the resource, by acting as showing in the algorithm of figure 3.

The schedule problem is here defined by the notation presented in [Pinedo 2002]. The schedule problem has three parameters that represent respectively the number of machines involved, the number of jobs, and the optimization function. In our case, we model the schedule problem as: [1 machine | 1 job | SLA load balance]. This scheduler optimization function intends to balance the workload jobs by choosing the interval with the minimum number of SLAs associated with. The machine can be a cluster head.



```
Check_policy{
(1) Check if user has permission to execute job in different time ranges
    if yes - go to step 2
    if no - go to step 4
(2) Check_SLA (time)
(3) Verify if price and quality are in the range allowed
(4) Return }
Check SLA (time) {
(1) Check if the number of SLAs already established is inferior to the maximum allowed
     if yes - go to step 3
     if no - go to step 2
(2) Schedule
(3) Return }
Schedule (time range) {
(1) Suggest a schedule time, in the time range in which user can run a job, with less
number of SLAs associated with
(2) Return }
```

Figure 3. Resource_agent algorithm skeletons for policy-based negotiation, SLA restrictions and scheduling processes

3.1.3 Network_agent perspective in the multi-issue bilateral chaining negotiation

From the network_agent perspective, a negotiation called chaining, occurs when data transfers are being considered. In this case, there are dependences between the negotiation that happens between user_agent and network_agent and the previous negotiation done between user_agent and server_agent. Such dependences are called of: 1) location precedence; and 2) time and price precedence.

Location precedence: There is location precedence between sites where data to be transferred are located and sites where candidate servers are located. The links that interconnect these points can be represented as a matrix M(Di,Sj)=[Lij] where, Di is a server where data requested by the user are available, Sj is a server selected by the user_agent to negotiate with, and Lij is a link that connects Di to Sj. Lij is sorted, based on the bandwidth, and n elements are selected from this set. Network_agents represent these n elements in order to negotiate with the user_agent in a chaining negotiation.

Time and price precedence: There is time precedence in a chaining negotiation since the time to finish a Network SLA (NSLA) has to be inferior to the time to start a Server SLA (SSLA), so, nsla.finish < ssla.start. There is price precedence in a chaining negotiation since the price to be negotiated has to be inferior to the maximum amount defined by the user minus the price spent in the SSLA, so, nsla.price <= priceMax – ssla.price.

3.1.4 Resource Performance Prediction

The resource performance prediction foresees how long a resource will take to execute a job. During the negotiation process, the server_agent can propose the duration of a job execution based not only in the machine's performance, but also in past cases of running such job at the considered resource. If a job with similar characteristics has already run in the same or similar machine, it is possible to predict the job run time. With prediction, the system estimates how many SLAs are associated to each resource at the same time and improves scheduling process [Setten et al. 2002].

3.2 Migration Module

Migration module is responsible for the decision of "where" to run a job. It is formed by SLA Matching and Migration sub-modules as described below.

SLA Matching Sub-module: The SLA Matching sub-module combines SLAs established by the negotiation module. It matches SLAs that are interconnected and are providing the same service. In order to model the SLA matching procedure, a graph G, directed and acyclic, is used. G=(V,E), where each vertex in V represents a SLA. Each edge denotes the precedence between SLAs in time schedule and accumulates SLA parameters. A node is represented as a tuple <Id,P,Q,Ts,Tf,G> where Id is the SLA identification; P is the price negotiated; Q is a set of quality of service; Ts is the job start time; Tf is the job finish time; and G is a set of guarantees.

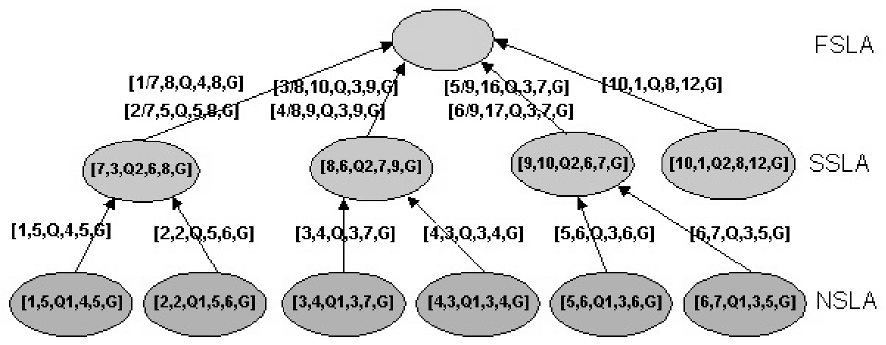


Figure 4. SLA matching graph

The time to start a Final SLA (FSLA) is the time to start the NSLA and the time to finish the FSLA is the time to finish the SSLA, so fsla.start = nsla.start and fsla.finish = ssla.finish. The price of FSLA is the price accumulation for all SLAs interconnected, defined as A(s) = nsla.price + ssla.price. The QoS and guarantee of the FSLA is a set of different QoS and guarantees established in the NSLA and SSLA respectively, where fsla.qos = {(nsla,ssla).qos} and fsla.guarantee = {(nsla,ssla).guarantee}. Figure 4 shows an example of a SLA matching graph. Layers represent FSLA, NSLA and SSLA. The root of the graph represents the Final SLA (FSLA) that will be selected by the migration decision sub-module.

Migration Decision Sub-module: This sub-module makes the decision of "where" to run a job. It takes into account the user preference among price, quality of service or performance. The migration module chooses the best combination of SLAs involved in a service delivery. The migration is done using mobile agents to transport applications and by GridFTP [Globus 2004] to transport data. This sub-module also submits job to the selected resources using globus commands. The SLAs selected are activated in the MDS.

3.3 OGSA Interface Module

This module provides the interface between the Multi_Agent System and the OGSA. It groups the functions associated with the use of grid commands. We are integrating our multi-agent system into the grid environment by the following implementations: 1) Extension of the grid directory service schema by adding SLA attributes. This extension is based in the Web Service Level Agreement (WSLA) definition [Dan et al. 2002]; 2) Definition of an authentication method for agents in grid; and 3) Inclusion of negotiation objects into grid resource access policies. Such policies were defined using the eXtensible Access Control Markup Language (XACML) [Godik and Moses 2003].

4. Conclusions

Intelligent resource allocation is still a challenge in the Grid middleware. Up to now, we have not found another research effort that is considering guarantees for the grid service delivery by exploring the negotiation process in the resource allocation.

We presented here a multi-agent system that addresses these issues with the aim to decide the best place to run a job in a grid environment. Our main contribution is concerning the selection of grid resources using a multi-agent negotiation process. In this negotiation process, we identified and modeled the bilateral, multi-issue and chaining negotiations.The migration process complements the negotiation process. It matches SLAs established by agents described here in a graph notation and makes the migration decision considering the set of SLAs that best fits the user objectives.

We expected to measure the percentage of improvement our system brings to the resource allocation problem in grid. Our future work will concentrate in the implementation of an effective learning mechanism for the prediction module. Prediction can help resource_agents to foresee the resource performance based in past cases. It can also improve SLA confidence and user expectations.

5. Acknowledgements

The first author thanks Prodabel for financial and institutional support.

6. References

Globus (2004), http://www.globus.org.

Czajkowski, K., Foster, I., Kesselman, C., Sander, V., Tuecke, S. (2002).SNAP: A Protocol for negotiation of Service Level Agreements and Coordinated Resource Management. (JSSPP'02), Edinburgh.

Raman, R., Livny, M., Solomon, M. (1998).Matchmaking: Distributed Resource Management for High Throughput Computing. HPDC-7. Chicago.

GRMS (2004), http://www.gridlab.org/WorkPackages/wp-9.

Jade (2004). http://sharon.cselt.it/projects/jade/

Zhang, Xiaogin, Lesser, Victor.(2002) Multi-Linked Negotiation in Multi-Agent System. Proc. AAMAS'02. Bologna, Italy, p.1207-1214.

Pinedo, Michael (2002) Scheduling: Theory, Algorithms, and Systems. Prentice-Hall.

Setten, Makr van, Veenstra, Mettina, Nijholt, Anton (2002) Prediction Strategies: Combining Prediction Techniques to Optimize Personalization. AHÂ'2002, Spain.

Dan, A., Franck, R., Keller, A., King, R., Ludwig, H. (2002) Web Service Level Agreement. (WSLA) Language Specification WSLA.

Godik, S., Moses, T.(2003) eXtensible Access Control Markup Language (XACML).Oasis Standard.