Adaptive Negotiation for Agent-Based Grid Computing

Weiming Shen and Yangsheng Li
Integrated Manufacturing Technologies Institute
National Research Council Canada
800 Collip Circle
London, Ontario, N6G 4X8, Canada
Phone: +1 519 430-7134
weiming.shen@nrc.ca

Hamada H. Ghenniwa and Chun Wang
Dept. of Electrical and Computer Engineering
University of Western Ontario
1151 Richmond Street
London, Ontario, N6A 5B9, Canada
Phone: +1 519 661-3758
hghenniwa@eng.uwo.ca

ABSTRACT

The Grid concept has recently emerged as a vision for future network based computing, by enabling seamless integration of computing systems and clusters, data storage, specialized networks and sophisticated analysis and visualization software. Intelligent agents can play an important role in helping achieve the Grid vision. Although agents have been applied to computing load balancing for many years, attempts to apply intelligent agents in realizing the Grid vision have been made by academic researchers during the past few years. Due to the highly heterogeneous and complex computing environments, effective load balancing for Grid computing is a very difficult problem, even though intelligent agents are used. In this paper we attempt to highlight major challenges in managing resources in a Grid computing environment and present some of our recent work on adaptive negotiation strategies for agent-based load balancing and Grid computing. The proposed approach is to implement multiple negotiation models/protocols/strategies that can be selected by the system automatically to adapt to computation needs as well as changing computing resource environment.

Keywords

Agents, Grid Computing, Adaptation, Scalability, Negotiation, Load Balancing.

1. INTRODUCTION

Allocation of limited computing resources in an open environment to complete maximum computation tasks is very complex and time consuming. Ideally, researchers and practitioners would spend no time at all deciding which systems to use, where the data resides for a particular application domain, how to migrate the data to the point of computation (or vice versa). Disparate computing resources keep disciplines stratified, so researchers often end up wasting time by replicating work, and this often results in wastage of resource utilization - as a researcher who decides when and where to run a job is often not aware of the loads and priorities of all systems.

The Grid concept is proposed as a means to help address some of these concerns, enabling seamless integration of computing systems and clusters, data storage, specialized networks and sophisticated analysis and visualization software. Like an electrical power grid, the Grid will aim to provide a steady, reliable source of computing power. Intelligent agents can play an important role in helping achieve the Grid vision. Due to the highly heterogeneous and complex computing environments, effective load balancing for Grid computing is a very difficult problem, even though intelligent agents are used. This paper presents some of our recent work on adaptive negotiation strategies for agent-based load balancing and Grid computing.

2. AGENT-BASED GRID COMPUTING AND RELATED WORK

Grid Computing is an exciting buzzword in the computing world today. It is usually defined as "the exploitation of a varied set of networked computing resources, including large or small computers, PDAs, file servers and graphics devices." The networks could be anything from high speed ATM to wireless or modem connections. Exploiting these connected resources could, for example, enable large scale simulations not possible on a single supercomputer, aid computational work of geographically distributed collaborations, simplify remote use of machines, and enable the new dynamic application scenarios.

Although agents have been applied to computing load balancing for many years, attempts to apply intelligent agents in realizing the Grid vision have been made by academic researchers during the past few years. A series of workshops on Agent-Based Cluster and Grid Computing were initiated in 2001 [7] as part of the IEEE/ACM International Conference on Cluster Computing and the Grid. The most interesting work in the literature might be the Agent Grid concept proposed under the DARPA ISO's Control of Agent-Based Systems (CoABS) program [6]. The agent grid is a specific construct or mechanism within that layer for making services and resources available. Another interesting example is ARMS [1].

According to Cao et al. [1], there are two key challenges that must be addressed for Grid computing: Scalability and Adaptability. Our proposed approach described below is to address these two challenges through scalable system architecture and adaptive negotiation techniques.

3. PROPOSED APPROACH

Based on our previous research results on agent-based computing load balancing for a distributed multidisciplinary design optimization (MDO) environment [4], agent-based manufacturing scheduling [5], as well as agent-based e-Marketplace, we have been working on the development of adaptive negotiation strategies for all these similar applications. In this paper, we present how this approach can be applied to complex Grid computing environments.

As introduced in the previous section, we are facing two key challenges for Grid computing: scalability and adaptability. In this paper, we propose a five-layer architecture (Figure 1) to address the scalability issue.

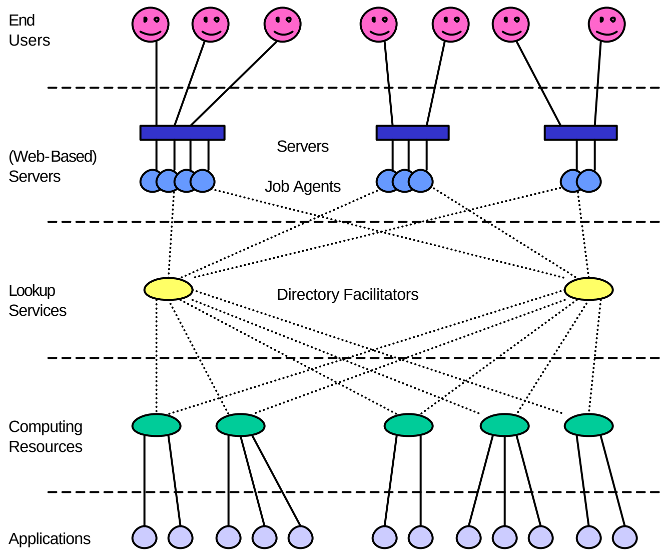

Figure1. Five -Layer System Architecture

As shown in Figure 1, the Applications layer contains all kinds of software applications installed in different computing resources (PCs, workstations, HPC clusters, etc.). Each application is to be wrapped into a software agent.

The Computing Resources layer contains Resource Agents. Each resource agent is used to manage all applications with "one" computing resource which could be a PC, a workstation, or a cluster of computers, and is responsible to schedule this computing resource. This resource agent should have knowledge about the hardware performance of this computing resource, and all applications installed with it as well as the performance of these applications in this particular hardware environment. Case-based learning mechanism will be implemented in this resource agent to keep a case history for all local applications.

The Lookup Services layer contains Directory Facilitators or Yellow Page agents. All resource agents in the Computing Resources layer need to register with these Directory Facilitators or Yellow Page agents.

The (Web-Based) Servers layer includes front-end servers for end users (human users or other software applications) to submit computation jobs. These servers are usually implemented as application servers. When a job is received by the server, a job agent will be created. This job agent will be in charge of finding candidate resource agents through Directory Facilitators, and select a most suitable negotiation protocol, coordinate the negotiation process to finally select resource agents to do the job. The job agent will be dissolved when the job is finished and the results are sent to the end user (the results may also be saved in a secure storage resource). The job agent must be *resource-aware* and be able to express its resource needs to the system, and be able to negotiate for system resources.

The End Users layer includes human end users or other software applications.

The proposed architecture shares some similarities with existing approaches, including commercial technologies like Jini™ network technology (http://www.jini.org/). However, many implementation issues are to be addressed, particularly the security issue. It is evident that the proposed five-layer architecture addresses well the scalability issue. As it can be seen in the following section, our adaptive negotiation approach is well suited to address the adaptability issue.

4. NEGOTIATION FRAMEWORK

4.1 Adaptive Negotiation Approach

In the context of agent-based computing load balancing or Grid computing, negotiation is used to optimize computing resources allocation to computation jobs over time. A negotiation protocol contains the basic rules for the negotiation process and the communication. In addition to using a protocol, each agent will develop and use a negotiation strategy appropriate to the problem to be solved. Clearly, negotiation protocols and strategies will be quite different for different categories of negotiations. Since negotiation involves exchanges of messages, protocols structure what are called *conversations*, defining classes of dialogue. The simplest dialogues are found in contract-net approaches where they are limited to exchanges involving offers, bids, and grants of contract. More complex dialogs are found in human types of negotiations, when trying to change other agent's beliefs. If negotiation protocols govern the exchange of proposals (and perhaps arguments) among agents, negotiation strategies decide the position of a particular agent during the negotiation process.

In order to address the second key challenge, i.e., adaptability, for Grid computing as mentioned in the previous section, we propose to use an adaptive negotiation approach, i.e., the job agent is able to select a suitable (optimal) negotiation protocol based on the specific computation needs, available computing resources and their computing loads. The decision is made by the job agent according to its knowledge using a case based learning/reasoning mechanism, i.e., the job agent learns negotiation strategies based on previous experiences. Each agent involved in the negotiation should be able to adapt to the protocol selected by the job agent (such information is contained in the request messages, e.g., Call for Bid messages).

By implementing this adaptive negotiation approach, the system will also be able to adapt to changes in system state. This is an important problem in Grid computing in which system resource availability may fluctuate. Such fluctuation may result from connection/disconnection of computing resources, human interaction/interruption on the computers, etc. Once scheduled, the job agent must also be notified if the system state has changed in a way that impacts the job (computation tasks) and perhaps enter re-negotiation and re-scheduling. At this point, the job agent may need to adapt to a different set of allocated resources.

It should be noted that a resource agent or a job agent might be involved in several negotiations of different types at the same time, which is usually called combined negotiation. Combined negotiation is another difficult issue, particularly in a multi-agent system implemented with the proposed adaptive negotiation approach. It is being investigated in our research group and will be reported separately.

4.2 Negotiation Models

We have been investigating various negotiation models/ protocols/strategies for our adaptive negotiation framework. In this paper, we introduce four negotiation models that are believed to be useful in agent-based computing load balancing: Contract Net Protocol [2]; Auction Model; Game Theory Based Model; and Discrete Optimal Control Model.

4.2.1 Contract Net Protocol

A modified Contract Net Protocol (CNP) was the only negotiation protocol implemented in the first prototype of our distributed MDO environment [4]. It will also be the most useful negotiation protocol in our adaptive negotiation environment. As mentioned in Section 4.1, the negotiation is done through a job and a selected resource agents (the selection process is done by the job agent checking with one of the Directory Facilitators with their lookup services). In the bidding process for selecting a computer, a cluster, or a group of computers/clusters for the requested computation task (one job agent can have a number of parallel or sequential computation tasks), after receiving the different propositions (bids) from different resource agents, the job agent will select a (or a group) of resource agent(s) to perform the task according to its criteria (e.g., cost criteria or time constraint) and award a contract to it. Other suitable resources are not selected to perform the task, but are recorded as alternatives which may be contacted (negotiated with) in the future in the case of unforeseen situations such as computer failures or any other task delays. This will greatly reduce the re-negotiation / rescheduling time when such unforeseen situations happen. The information about the alternative computing resources will be saved by the job agent together with the information about the selected resources. When the selected resources cannot perform the scheduled tasks due to unforeseen situations, the job agent may negotiate directly with alternative resource agents.

The CNP is the default negotiation protocol to be used the current prototype system in case no other specific protocol is selected.

4.2.2 Auction Model

An auction is a negotiation mechanism for selling indivisible goods (computation resources, in our case) to bidders (application agents). An auction is a one-to-many negotiation, between one seller (resource agent) and many buyers in which the negotiation is reduced into a single variable domain, namely price (e.g., money). An auction negotiates a mutually acceptable solution for the buyer and the seller (it uses market forces to negotiate a clearing price for the item). The auction mechanism sets out rules for bidding, and allocates the computation resource to a certain bidder based on its predefined rule set. The auction literature is rich and varied, and the following is only a brief overview of four major types of auction mechanisms. Note that we assume that there is no further penalty to losing the auction (i.e., the losing bidders do not pay anything).

1. English auction: This is the most common type of auction. It is an open outcry, ascending auction. The auctioneer begins at the seller's reservation price, and solicits progressively higher oral bids from the audience until only one bidder is left. The winner claims the item, at the price it last bid. This auction is strategically equivalent to the Vickrey auction, and carries the additional benefits that it makes bidder reservation prices publicly known and is efficient in the sense that it will give the object to the bidder who values it the most.
2. Dutch auction: This auction is an open outcry, descending auction. The auctioneer begins with a price too high for any buyer to pay, and progressively lowers the price until one bidder calls out, "Mine!" The winner claims the item, at the price it bid. This auction, however, is not necessarily efficient.
3. First-price, sealed-bid auction: In this auction bidders submit a single, irrevocable sealed bid. The bids are opened simultaneously, and the winner is the highest bidder, who claims the item at the price he bid.
4. Vickrey auction: In this auction bidders submit a single, irrevocable, sealed bid. The bids are opened simultaneously, and the winner is the highest bidder, who claims the item at the second-highest bid price. This auction is strategically equivalent to the English auction.

An auction is suited quite well to the automation of the dynamic resource allocation problem for the following reasons:

- The resource allocation optimization problem can be directly mapped into a single variable domain, i.e., price.
- The dynamic structure of the problem has been naturally captured in the multi-agent system architecture and auction

mechanism; by which the solution is dynamically emerge from the current setting of the problem.

- The rules of the negotiation are clearly spelled out in the rules of the auction. Software agents can be easily designed to follow clear rules.
- An auction's mechanism can be inference-proof. When an auction is designed properly, neither the buyer nor the seller will have an incentive to lie or hide their strategies (i.e., the Revelation Principle). Take the English auction with no reservation price as an example:
  - The seller announces its strategy: to sell to the highest bidder at the last price offered.
  - The seller's rules dictate the buyers' strategies. A rational buyer would establish a reservation price and actively bid up to but not beyond that reservation price. A rational buyer's strategy is public knowledge.
  - The price negotiation commences with the first bid; the item will be sold to the highest bidder at the second-highest bidder's reservation price.
  - The strategy is inference-proof; in fact, it depends on inference for its success. If bidders were not aware of the rule for determining the winner, they would not raise the bidding accordingly.

4.2.3 Game Theory Based Model

Negotiation among self-interested agents has been studied from the perspective of game theory. Early research work in this area was carried out by Rosenschein and Zlotkin [3]. In game theory, the global outcome for the system is given in a table showing the results of combined decisions. Each player, however, makes decisions independently.

In case that job agents are competing to get their jobs done or resource agents are competing to do more computation tasks to earn more money (e.g., computing resources are operated by different commercial service providers), Game theory based negotiation protocols may be useful.

When a game theory based protocol is selected, all agents involved in the negotiation process are self-interested. To simplify the explanation, we assume only Rosenschein's protocol of alternating offers [3] are used, individual agent has its own utility function, and they can choose "out" (withdraw form the negotiation). Two scenarios can be implemented:

If a resource agent thinks a job is costly thus against his interest, it could offer a plan which involves other agents in the job before it accepts it. If other agents in the group do not agree with its offer, they can offer their plans to share the responsibility. Every agent has its opportunity to offer a plan. If a plan is accepted by all agents in the group, the negotiation is terminated and the plan is implemented. If all agents choose "out" or no agreement is reached, the job might be simply turned down, or the job agent needs to start a new negotiation process using a different protocol.

When two or many job agents are competing one or several computing resources at the same time. The job agents and resources agents could form a negotiation group. They offer their plans to solve this competition in turn. The agents try to achieve an agreement to work out a schedule using game theory based protocols.

4.2.4 Discrete Optimal Control Model

The proposed discrete optimal control model can be considered as a kind of optimized market model. It consists of three steps: acquiring information, making a decision and announcing a decision. A Marketplace Agent obtains the information about related resource agents (producers or processors in this case) such as the price ($ per hour), tasks (customers) such as the number of the tasks and performance objectives. Then the Marketplace Agent chooses a proper algorithm to make a decision based all available information and its knowledge. Finally, it proclaims the results to other related agents.

In this case, we suppose that there are *m* processors: P_j $(j=1,2,...,m)$, and *n* task T_i $(i=1,2,...,n)$. In order to emphasize the principle, we presume: (1) the tasks are independent to each other; (2) task T_i $(i=1,2,...,n)$ can be run on processor P_j $(j=1,2,...,m)$; (3) preemptions are not allowed; (4) the performance objective is the load balancing. The dynamic system model can be created as follow:

Dynamic equation:

$$\vec{X}_{k+1}=\vec{X}_k+B_k\cdot\vec{U}_k,\ (k=1,2,...,n)$$

$$\vec{X}_1=\vec{X}(1)$$

Measure:
$$\text{Min. } Q=\sum_{k=1}^{n+1}\sum_{j=1}^{m}(x_k^j-\frac{1}{m}\sum_{j=1}^{m}x_k^j)^2$$

where:

$\vec{X}_k=\begin{bmatrix}x_k^1 & x_k^2 & \cdots & x_k^m\end{bmatrix}^T$ is state of the system describing the accumulative work time vector of the processors before the task T_k is loaded. The unit vector $\vec{U}_k=\begin{bmatrix}u_k^1 & u_k^2 & \cdots & u_k^m\end{bmatrix}^T$ $\in\Omega(k)\subset R^m$ is control input vector (decision). $u_k^j=1$ refers to the task T_k run on the processor P_j, vice versa. $B_k=\begin{bmatrix}b_k^1\vec{e}_1 & b_k^2\vec{e}_2 & \cdots & b_k^m\vec{e}_m\end{bmatrix}$ stands for the processing time matrix of the task T_k. b_k^j is the processing time of the task T_k on the processor P_j. $\vec{e}_j$ is the vector which the row *j* is 1 and other rows are 0; $(j=1,2,...,m)$. The processing time of the task T_k on processor P_j usually can be described as b_k^j $(k=1,2,...,n)$, $(j=1,2,...,m)$. Q represents the difference of the run time of each processor. This model is a discrete optimal control model with constraints.

This mechanism can be used to get an optimal solution when the above-mentioned conditions are met. Fore other complex situations, soft computing techniques (Genetic Algorithms, Neural Networks, and Fuzzy Logic, etc.) can be combined with the proposed method to simplify the optimization process.

5. CONCLUSIONS AND PERSPECTIVES

Grid computing and particularly agent-based grid computing are new areas of research with many open issues and challenges. Thompson [6] enumerates a long list of open issues from Agent Grid point of view. This paper reports some of our recent work on adaptive negotiation strategies for agent-based load balancing and Grid computing, without considering many other related issues such as volume of data to be transferred, network bandwidth, traffic, and security, etc. We tried to address the two key challenges for Grid computing, i.e., scalability and adaptability, by applying intelligent agents to computing resource management or load balancing in Grid computing environment. Our research work is still at a preliminary stage, and many detailed implementation issues are to be investigated. The major contributions of this research work include adaptive negotiation and combined negotiation strategies as well as the discrete optimal control model as a novel negotiation model.

It should be noted that the proposed approach can be used not only in agent-based Grid computing environment, but also in other similar resource allocation or scheduling problems, e.g., agent-based manufacturing scheduling and transportation scheduling, as well as agent-based e-Marketplace or e-Business.

6. REFERENCES

[1] Cao, J., Kerbyson, D.J. and Nudd, G.R. Performance evaluation of an agent-based resource management infrastructure for grid computing, in Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid, (Brisbane, Australia, 2001), pp. 311-318.

[2] Davis, R., and Smith, R.G. Negotiation as a Metaphor for Distributed Problem Solving. Artificial Intelligence 20 (1983), 63-109.

[3] Rosenschein, J. S., and Zlotkin, G., Rules of Encounter: Designing Conventions for Automated Negotiation among Computers, (MIT Press, 1994)

[4] Shen, W. and Ghenniwa, H.H., Multidisciplinary Design Optimization: A Framework for Technology Integration, in Proceedings of the First International Conference on MDO, (London, ON, Canada, July 2001), pp. 22-28.

[5] Shen, W. Distributed Manufacturing Scheduling Using Intelligent Agents, *IEEE Expert / Intelligent Systems*, 17(1) (Jan/Feb., 2002), 88-94.

[6] Thompson, C. Characterizing the Agent Grid, Technical Report, Object Services and Consulting, Inc., 1998 [http://www.objs.com/agility/tech-reports/9812-grid.html]

[7] Workshop on Agent based Cluster and Grid Computing, 2001, http://www.cs.cf.ac.uk/User/O.F.Rana/agent-grid/