A Framework for Adaptive Matchmaking in Distributed Computing

K. Sigdel, K. Bertels, B. Pourebrahimi, S. Vassiliadis and L. Shuai

Computer Engineering Laboratory, ITS, TU Delft, The Netherlands
email: [kamana,koen,behnaz,stamatis,psamuri]@ce.et.tudelft.nl

Abstract

One of the major issues of a distributed system is *load balancing* where workloads are distributed throughout the network. For this, it is necessary to assign new tasks to idle resources and reduce the workload for some other overloaded nodes in the same network. This process of assigning workloads to the resources is called matchmaking process. On the continuum from centralized to peer-to-peer mechanisms, we are interested in designing a matchmaking mechanism that enables the network to change its internal matchmaking mechanism from P2P to a more centralized form or vice versa whenever that is required. This will allow the distributed system to adapt itself dynamically to changing conditions and always have the most appropriate matchmaking infrastructure available. This approach boils down to, for instance the idea of multiple matchmakers where the entire system can be partitioned into segments such that every segment has a matchmaker. Agents can then interact with each other in a local neighborhood, and matchmaking in different segments can take place in parallel. Completely centralized or completely localized matchmaking mechanisms each have their efficiencies and deficiencies. This paper defines a framework for an adaptive system architecture that offers a dynamic infrastructure in which the distributed system can manage itself according to the changing circumstances to switch between completely decentralized and completely centralized or any state in between these two.

1 Introduction

The sharing of resources is a main motivation for constructing distributed systems in which multiple computers are connected by a communication network. The problem in such systems is how resource allocation should be done in the case where some resources are lying idle and could be linked with other overloaded nodes in the network. This assumes some kind of matchmaking that is the process of finding an appropriate provider of resources for a requester[15]. Given the variety of approaches for matchmaking, it is important to be able to determine the conditions under which particular mechanisms are more performing than others. A framework that defines a set of criteria can be used to assess the usefulness of a particular mechanism. In this paper, we define a framework allowing an *adaptive* approach of matchmaking that can automatically conform to the most appropriate matchmaking methods for task and resource allocation

depending on its network structure and characterstic. The word *adaptive* here denotes an aggregated and self-operating or self-regulating system that can automatically initiate a modification according to changing circumstances of the system (for instance when computing nodes leave or re-join the network).

The paper is structured as follows: the next section of the paper will discuss the related research. We then describe the matchmaking approach with multiple matchmakers, discuss problems associated with the centralized and p2p mechanism and introduce the need for the adaptive approach. The later portion of the paper will discuss the proposed system model with multiple matchmakers and its target implementation with planetlab.

2 Related Research

Previous research on matchmaking[15] and distributed resource allocation [7] [5][10] proposed different methods for matchmaking [11][14] and resource scheduling (with matchmakers or without matchmakers) that represents both dynamic as well as distributed approach of problem solving. In general, matchmaking mechanism for multi-agent system can be classified into 3 categories: First, market bidding mechanisms where bids and offers are broadcasted to all the agents in the market. Second, matchmaking using broker or middle agent where brokers keep some kind of information directory of the entire system and match accordingly between the service requester and the service provider based on this information. Third, peer-to-peer communication where agents can only interact within a local neighborhood without having any central controlling entities [9][13]. In [3], we defined a model of agents with producers, consumers and matchmakers. Consumers send to the matchmaker the number of tasks they want to delegate and the producers will announce in the same way how much processing time they have available. With this information, the matchmaker matches the consumer requests and producer bids in centralized manner. In this process, it was observed that there is some kind of population size beyond or below which either the matching efficiency remains constant or goes down respectively. This seems to indicate that, when the population size in the network grows, single matchmaker will not be efficient to entertain all the agents and it is necessary to introduce another matchmaker to ensure the matchmaking capacity.

3 Matchmaking with Multiple Matchmakers

Whenever a computing node needs some additional resources to perform its assigned task, the node needs to locate the available resource and make use of it. This assumes some kind of matchmaking enabling collaboration between computing node and the additional resources available. This approach can be either centralized matchmaking [3] or peer to peer [4].

3.1 Problems Associated with Centralized matchmaking

The centralized matchmaking works with one node in the center maintaining central directory of information from requesters(about resource request) and providers(about resource) [3]. Centralized matchmaking has good throughput for certain population size (as the system grows since matching time increases will decrease the throughput) and guarantees that an agent finds the match if it exists and it allows agent to find the best match system wide. At the same time, it has low scalability (it is efficient only for a certain range of population size) because of the bottleneck associated with the central matchmaker when population grows[6][3]. Furthermore, this mechanism has least robustness since the whole system might fail when matchmaker leaves the network or goes down due to hardware or network failure.

3.2 Problems Associated with peer to peer

In P2P system, computing nodes distribute resources via direct exchange between computers without having centralized control or hierarchical organization [12][8]. In P2P, the system employs distributed resources to perform functions in a decentralized manner. Since, it works in a decentralized manner, it has no single point of failure. However, it has low matching rate and will not guarantee to find matches even if it exists in the system. In localized matchmaking, agents only interact within a local neighborhood, that is only a small subset of the entire agents whose addresses are known to that agent. So, peer to peer has low throughput when there is no interaction between nodes' neighborhood.

3.3 Adaptive Approach

As discussed above in sections 3.1 and 3.2, the completely centralized or completely localized matchmaking mechanisms each have their efficiencies and deficiencies. To design a dynamic architecture, the first question that needs to be addressed is to determine the conditions under which either of the above approaches is most efficient and usable. The next step then becomes to define a mechanism that allows the system to restructure itself to the changing states. The connection network of loosely coupled computing nodes can always grows or shrinks as nodes can leave or join the system continuously. Here, we are talking about an open and dynamic system of ***producers***, ***consumers*** and ***matchmakers*** where these agents can come, go and can change its purpose and role whenever required. Consumers are the processing nodes wanting some resources to solve its jobs and producers are the nodes which can provide these resources. Matchmakers are the mediator agents that match the producer's bid with the consumer's request [11][14]. Producers, consumers and matchmakers have a common functional structure which can be enabled/modified among themselves while having their own attributes and methods (for detail see section 4). In this way producer/consumer can become a matchmaker and matchmaker can again returns to 'normal' producer/consumer state. The figures 1, 2 and 3 show the

different views of such system.

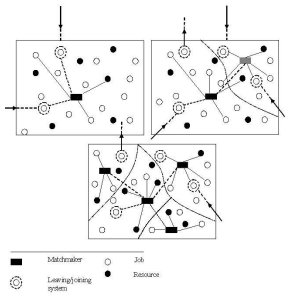

Fig. 1: matchmaking with multiple matchmakers

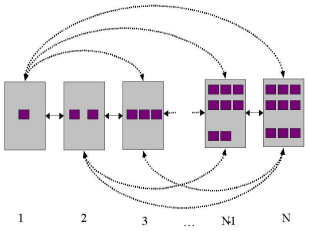

Fig. 2: meta view of the systems

The adaptive approach embraces basically two levels of adaptation viz: ***system level adaptation*** and ***node level adaption***. In system level adaptation, the system adapts according to the changing circumstance of the network and accommodates itself based on it's alternating behavior of growing and shrinking of population size and/or workload etc. For instance, with growth in population size or/and increase in workload, the central matchmaker would not be able to handle all the agents efficiently. One of the way to solve this problem would be to re-structure the system by introducing new matchmakers to reduce the overhead of the central matchmaker. For this a new matchmaker can be defined in a localized manner and a new segment of producers, consumers and matchmaker can be created. This would create a view of the system with segments, each segment having its own matchmaker (see fig 1). Similar mechanism could be applied when workload reduces or/and population size shrinks i.e to combine the segments and avoid matchmakers to increase the throughput. The node level adaptation enables for the system level adaptation. In node level adaptation, nodes can be transformed from one form to another form. When system needs more matchmakers, producer or consumer can be promoted as a matchmaker with little modification. And when these matchmakers are no more required in the system, it can be changed back to producer or consumer.

In addition to the general matchmaking, we also introduce the notion of a meta-matchmaker. Its function is twofold: first, to allow arbitrage between different segments for the remaining requests. Second, it will allow to introduce a self organizing mechanism that will enable the change in number of matchmakers. One of the major issues in adaptive system is the load balancing between segments. In every segment, a matchmaker takes care of finding matches between requesters and producers and when there are some tasks/resources with no matches, the matchmaker should introduce them to other segments. Communication and cooperation between matchmakers can be done in different ways. One possibility is having a flat structure with a meta-matchmaker that manages

the interaction between matchmakers as mentioned above. The other possibility is having a hierarchical structure in which every matchmaker acts as a child for a higher level matchmaker. Every matchmaker can interact with other matchmakers only through its parent or its child matchmaker like a tree structure. The third possibility is considering matchmakers as peers in a pure p2p system so that every peer knows its neighbors. Whenever these peers want to interact with each other, the interaction message will be propagated in the system through the neighboring nodes.

4 System Architecture

The figure 3 shows the system model. In order to introduce self configuration, we propose a system/node architecture which is highly modular with clearly defined interfaces (fig 3). The system contains three types of entities viz: producer, consumer and matchmaker. Each of these entities has its own methods and attributes as shown below:

- **Consumer** ⇒ {resource manager, job manager, request manager, exchange manager}
- **Producer** ⇒ {resource manager, job manager, request manager, exchange manager}
- **Producer** ⇒ {resource manager, job manager, request manager, exchange manager, consumer list, producer list, matchmaking function}

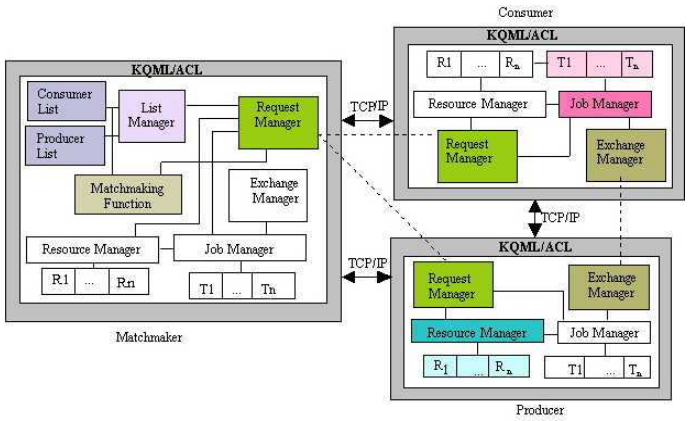

Fig. 3: System Model

Components that are common to producer, consumer and matchmaker are:

- ***Resource Manager:*** It monitors and manages the system resources in each node. System resources can be memory, CPU cycles, power etc. Whenever there are some idle resources in the node then resource manager reports it to the request manager to make an offer to sell that resource.

- ***Job Manager:*** Computing nodes maintain the job queue of all the task to be processed and the queue has its job manager. The job manager manages the task queue that is being processed by the node and asks for additional resources whenever it runs out of resources.
- ***Request Manager:*** It is the responsible for handling all the requests form the job manager. The request can be the request for buying resources (from consumer) or offer of selling the resources (from producer). The request manager receives the requests from job manager and transmits them to matchmaker. It is the only component that can interact with the matchmaker .
- ***Exchange Manager:*** It is the responsible for executing the transactions between consumer and producer. Once a matching has taken place, the exchange manager needs to transfer the data and instruction to the receiving producer or accept data and instruction from the requesting consumer.

In order to become a matchmaker following components need to be enabled:

- ***List manager:*** List manager in matchmaker manages the consumer list and the producer list. Whenever there is a new request for resources or an offer for resources, it updates the producer list and consumer list accordingly. It is also responsible for removing the requests from list when they are timed out or no more available.
- ***Matchmaking Function:*** It is the main matchmaking logic (e.g. firstMatch, MinDifference or MinDistance) for the matchmaker to match between producer and consumer. It continuously scans producer and consumer list to find the appropriate match between them.
- ***Producer List:*** contains the list of the producers with free resources. It contains the address of the producers with the corresponding column containing the type of free resources. (For easiness, this list can be sorted by the type of the resources.)
- ***Consumer List:*** contains the list of the consumers with pending task. It contains the consumer address and its task with the corresponding column containing resource type required for that task.

We intend to implement this architecture in the PlanetLab[2] environment. Planetlab and Globus[1] are the infrastructure platforms that sit on top of Internet and provide an environment for deploying, evaluating and accessing distributed services with geographically distributed resources and users. Planetlab focuses more on network intensive applications while Globus is more oriented towards computational intensive aplications. As we are studying low level interaction rather than real computation we chose Planetlab as an implementation platform.

5 Research Issues

Given the variety of approaches for matchmaking, it is important to be able to determine the conditions under which particular mechanisms are more per-

formant than others. The first issue of the research is to determine the criteria to assess the usefulness of a particular mechanism. These criteria can be:

- ***Scalability***: Scalability requires that an increase in the number of new agents and resources has no noticeable effect on performance nor on administrative complexity.
- ***Dynamic and flexible behavior***: Multi agent systems need to manage problems like increases and fluctuation of the number of agents, changes of task profile and drop-outs of agents. They should be able to determine the most appropriate organizational structure by themselves at run-time (self-building) and to change this structure as their environment changes (adaptivity).
- ***Throughput***: Throughput is defined as the number of tasks that could be allocated to available resources. It is an important element to be considered especially while considering very large scale systems.
- ***Robustness***: It is the ability of the system to remain operational even when minor or major disrupting events occur. This is a vital characteristic of a distributed environment and the matching mechanism should therefore also have this property.

In addition to the criteria, it is also important to establish the dimensions of the matchmaking mechanism on which the criteria should be applied. Examples of such dimensions are population size, task size, task and resources distribution. Beside these, there are some additional research issues that should be addressed:

- Should matchmaking process involve some kind of pecuniary system? If so, what are the implications of the decision making process?
- Should we include differentiation between nodes to have some kind of multilayered network? Certain nodes would then be differentiated in terms of processing resources, memory etc, How does the matchmaking mechanism should be modified to 'automatically' make use of such 'supernode'?
- Should we consider some kind of 'cascade' scheduling approach allowing the supplier of resources to reschedule itself certain tasks to other nodes? What are the implication of such an approach?
- Should the matchmaking function take bandwith into account? How about the communication and computation cost associated with it?

6 Conclusion and Future Work

In this paper, we presented a framework for implementing an adaptive resource management for distributed computing considering the advantages and disadvantages of centralized and localized matchmaking mechanism. The adaptation is both at the node and the system level. Individual nodes are responsible for timely execution of the tasks and can request additional tasks and resources when necessary. At node level, producers/consumers can be transformed to matchmakers and vice-versa. At the system level, an efficient matchmaking is envisioned by providing the capability of introducing(or deleting) matchmakers in order to guarantee a high(both in quality and quantity) number of

matches. Future research will involve the implementation of the proposed framework and experimental tests to study the conditions under which the adaptation (node/system) will take place and how they perform under varying circumstances.

References

1. www.globus.org.
2. www.planet-lab.org.
3. K. Bertels, N. Panchanathan, S. Vassiliadis, and B. Pourebrahimi. Centralized matchmaking for minimal agents. In *Proceedings of the Conference on Parallel and Distributed Computer Systems*, page 9, November 2004.
4. S. Camorlinga, K. Barker, and J. Anderson. Multiagent systems for resource allocation in peer-to-peer systems. In *Proceedings of the winter international synposium on Information and communication technologies*, pages 1–6, 2004.
5. L. Cuihong and K. Sycara. A stable and efficient scheme for task allocation via agent colition formation. In *Algorithms for Cooperative Systems*. World Scientific, 2004.
6. S. Jha, P. Chalasani, O. Shehory, and K. P. Sycara. A formal treatment of distributed matchmaking. In *Agents*, pages 457–458, 1998.
7. J.Modi, H. Jung, M. Tamble, W. Shen, and S. Kulkarni. A dynamic distributed constraint satisfaction approach to resource allocation. In *Proceedings of Autonomous Agents and Multi-Agent Systems Workshop on Distributed Constraint Reasoning*, 2002.
8. K. Kant, R. Iyer, and V.Tewari. On the potential of peer-to-peer computing:classification and evaluation. In *Proceedings. of CCGrid, Berlin, Germany*, 2002.
9. M. Koubarakis. Multi-agent systems and p2p computing: Methods, systems and challenges. In *Proceedings of the 7th International Workshop on Cooperative Information Agents (CIA 2003), Helsinki, Finland*, pages 46–61, August 2003.
10. J. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on computers*, pages 705–717, May 1989.
11. M. Luck, P. McBurney, and C. Preist. A manifesto for agent technology towards next generation computing. *Journal of Autonomous Agents and Multi-Agent System*, pages 203–252, 2004.
12. D. S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu. Peer-to-peer computing, March 2002.
13. E. Ogston and S. Vassiliadis. Matchmaking among minimal agents without a facilitator. In *Proceedings. 5th International Conference on Autonomous Agents*, pages 608–615, May 2001.
14. K. P. Sycara, S. Widoff, M. Klush, and J. Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agent and multiagent system*, pages 173–203, 2002.
15. Z. Zhang and C. Zhang. An improvement to matchmaking algorithms for middle agents. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1340–1347, 2002.