A Different Approach to Resource Management for Distributed Systems *

S. Groh
Munich University of Technology
Department of Computer Science
80290 Munich (Germany)

M. Pizka
Munich University of Technology
Department of Computer Science
80290 Munich (Germany)

Abstract *This paper presents a new concept for a distributed resource management. The management is performed by a multi agent system and based on a graph rewriting system. Every activity in the distributed system is managed by one agent, which is the basis for a management that adapts to each application. The cooperation among all agents achieves the management of the whole system and is the foundation for a fair and efficient resource mapping for all applications, regardless of whether the applications are executed in parallel or not. The management overhead is reduced by many different realizations for the agents and the possibility to transform an agent from one realization to another which is formally described by a graph rewriting system.*

Keywords: resource management, multi agent system, application adapted management, graph rewriting system

1 Distributed Resource Managements

In the past years the computing power of workstations has grown enormously. But also the demand on computing power rises. One main problem is, that the demand rises in some areas faster than the development. Therefore parallel programs are necessary to use more than one CPU at the same time. The drawback of parallel programs are the very high costs for parallel hardwares. A solution to minimize these costs are the Intranets of the companies, which are set up in many companies with high throughput rates. Upon closer examination it is obviously, that most time a workstation in a cluster is idle. Therefore the goal should be to use these available computation power for high computing parallel application. The consequences for the resource management is to support all services, which are needed by existing software, to support new services for distributed systems, like file sharing, and in addition to that support efficient services for parallel systems, like distributed shared memory or load balancing.

In many existing operating systems for workstations, like Solaris and HP-UX, the first steps are taken. All of them support network file systems (NFS) or network information services (NIS). Additional services are added by servers like distributed shared memory servers (DSM) or distributed light weight processes (activities) servers [1, 2, 3, 4].

The remaining question is which services should be added to these servers to reach the goal of a distributed resource management. Many strategies are known for mapping or for load balancing. Also many different strategies for distributed shared memory are developed. But not all strategies are usable with all applications. A strategy may be efficient for one application, but for another it is absolutely inefficient [5, 6]. One way to answer this question is to integrate the resource management into the application. This means the programmer

*This work is sponsored by the German Research Council (DFG)

of an parallel application has to take care about the distributed system and optimize the application for a specific workstation cluster. The disadvantage of this approach is the increasing cost for software development because of the necessary hardware knowledge and the lacking portability. Another way is to have strategies in the servers, which are efficient for the average application. The consequence is the possible lack of performance for certain applications. The aim is to have a resource management that adapts to a application, but which is nevertheless portable and usable in cooperation with other applications.

2 The basic concept

A parallel application is in this context an application which is made of many light weight processes (activities) using shared memory. The number of activities changes during runtime and the communication among the activities is high. The applications are fully portable, which means, that no explicit definitions for mapping is included in the application.

This approach tries to delete the boundary between application and resource management, without making the system unportable. The key is a high level programming language, that allows to describe the parallel problem without considering a certain hardware. Every job which can be done in parallel is marked. The communication is done via shared objects or via method invocations. Based on this information the compiler tries to find the best realization for each parallel part. This is done according to the distributed hardware on which the system should run. The information, how the realization should or could be, is given to special managers, later on called *manager agents*. A manager is responsible for the realization and uses the information from the compiler to do its job.

In order to get the necessary knowledge about the system the manager cooperates with other managers. The functionality of a manager is determinized by a graph rewriting system. As described in section 3 a manager has a couple of rewriting rules to manage a set of objects. The information about the system is stored distributed in all managers and exchanged via cooperation.

3 The formal background

A graph rewriting system is a powerful tool to describe the distributed system and the behavior of the management system. This is done by representing all objects by nodes and dependency information by edges. Additional information is included by attributes, which are added to the nodes and to the edges. The management functionality is represented by rewriting rules and additional strategies. A detailed description of the graph rewriting system can be found in [7].

The next step is to build a running management system on the base of the formal description. The management system adopts from the distributed environment the distributed realization. Every management decision has to be made distributed and not centralized to avoid a possible bottleneck. With this in mind a communication and cooperation between the now distributed management units has to be established. The information included in the graph has to be distributed among all management units and reduced to the important points (depending on the management units) to avoid the duplication of information and by this to save resources. All this can be done by building management agents as described in the next sections.

4 Multi Agent Systems

The concept of agents was first mentioned in the 80-ties. Afterwards more and more disciplines of computer science used this term, like artificial intelligence, cooperation between two humans with the help of agents, or distributed systems in general. Because of these many fields of applications no common defi-

nition exists, but the following attributes are nearly common for all fields [8, 9]. The first requirement for agents is the mobility. The multi agent system defines an environment in which an agent is able to move nearly free and is not bound to one specific location. The second requirement is its autonomy. It is able to decide things regarding its functionality on its own. Other requirements are its flexibility, the possibility to cooperate and communicate as well as its scalability.

The term flexibility means, that the agent is able to flexibly react to its environment. It has to adapt to the situation in order to fulfill its function in the best way by the help of the environment. To be able to do so, it has to cooperate and communicate with the environment, especially with other agents. The autonomous decisions are made by communicating and trading with other agents and by trying to find the best solution for all agents. Special strategies exist to solve possible conflicts during the trading process. The scalability is necessary to have the possibility to increase and decrease the number of agents during a period of time, to be able to adapt the number of agents to the unprocessed jobs.

5 Resource Management based on Manager Agents

5.1 Task of a Manager Agent

The graph rewriting system describes the state of the distributed system. Each manager agent is responsible for a part of this graph. The distribution is done according to the activities of the system. Each manager agent is responsible for only one activity. It is generated before the activity is generated. During the generation, information is passed from the generating manager agent to the new manager agent concerning the activity, that has to be generated as well as the status of the system. This means the new manager agent gets all information it needs from the complete graph. After its generation the newly created manager is responsible for all resource needs of its activity or in other words, it is primely responsible for the management of a subgraph, which includes its activity. This means in detail, that one of its jobs is to organize the hardware resources, like memory and CPU time for the activity. This includes the task to look for better resources for the activity in the system. Another of its jobs is to manage the higher level resources of its activity. A higher level resource is for example a shared memory object. Because of the distribution many management decision about the realization of such objects have to be made. To manage the communication between another activity and its own activity is also the task of the manager agent. The complete definition of its task is as follows: the agent is responsible for all resources on all levels of abstraction which are needed by the agent's activity to fulfill its job.

As mentioned above the manger agent is customized for its activity. It is the only manager agent, which has the complete subgraph with all attributes. Therefore it is a specialist for the resource demands of its activity. It knows exactly how often its activity will use shared objects, how much CPU power it will need and how important its activity really is for the whole application. During the runtime the agent collects information about the behavior of the activity and updates the attributed in its subgraph.

The manager agent tries to find the best realization for its activity with the help of its information. This is done by evaluation strategies given with each rewriting rules. The intensity of its efforts to find the best rewriting rule depends on the activity. Maybe it tries to make its efforts by saving resources for its own realization and therefore only tries to find a less than optimal rule (for example for very small activities), or it makes more efforts and uses more resources for the finding process in order to find a optimal solution. A more detailed description is given in section 5.3.

5.2 Cooperation among Manager Agents

Beside of the information included in the subgraph, the manager agent also needs information about the environment of the subgraph. Therefore it has to communicate with other manager agent to get an overview about the rest of the graph. To minimize the communication costs a special communication protocol is established between the agents. The foundation for these protocol is a force model. A more detail description can be found in [10].

Beside of the information flow also a cooperation between the manager agents is necessary. Each manager agent tries to find a optimal realization for its thread. The result of this effort is an executable rewriting rule. But in many cases this rule involves objects which are under control of other manager agents. Therefore each manager agent has to cooperate and negotiate with other manager agents. To avoid conflicts between the manager agents a hierarchy is defined on them. The root of the whole system is the manager agent of the first activity in the system. One of its first tasks is to collect necessary information about the system to be able to build the starting graph. Furthermore it is responsible for all following manager agents and for all hardware resources in the system. This sounds, as if the problem of the information bottle neck of the servers is now concentrated in this manager agent, but the responsibility does not mean that this agent really has all the information. It only means that in the case of conflicts it has the last word, or in other words, it is able to solve every possible conflict in the system by deciding which rewriting rule has to be performed.

Another possibility is to generate or use an existing helping manager agent. A helping manager agent is a special form of a manager, which has no own activity for which it is responsible. It only has delegated tasks. The aim is to concentrate the informations of several subgraphs, which does not fit directly together in one manager agent. A very simple example for helping manager agents is a load facility. According to the hierarchy of the manager agents it is possible, that two manager agents are realized on one workstation, which only shares the root manager as their parent, but are themself located far away from the root of the graph. In such a case the communication and cooperation among these manager agents is optimized by establishing a helping manager agent, which is responsible for both subgraphs and therefore is able to make decision concerning both manager agents without involving all manager agents of the whole hierarchy.

With the help of the hierarchy the conflict solving strategy is very simple. If a conflict occurs, the manager agent who is responsible for the conflict resource has to solve it. All other manager agent have to agree to this decision and make their decisions according to this new situation. If a manager agent is not able to solve a conflict, its parent manager agent is responsible to solve the conflict.

5.3 The Realization of Manager Agents

Finally there is the question of how to realize a manager agent. The task of a manager agent is to manage all resources of a activity, by evaluation rewriting rules. As mentioned in section 2 the number of activities in a distributed environment with many parallel applications is very high. In consequence the number of manager agents is also very high. Therefore the realization of the manager agents is a critical point. If we assume that every manager agent is also an active thread, the system would spend more time for managing the resources than for solving problems of the applications. But this is not necessary. The customization of the manager agents makes it possible to realize them in many different ways. As already mentioned in section 5.1 it depends on the application and activity how extensive a manager agent tries to find the best solution. Therefore the realization range for manager agents is wide. Very small manager agents may just be realized as small function calls or additional commands added to the application. On the

other end of the realization range there is the possibility to implement a manager agent as a thread. Which kind of realization is chosen depends on the rewriting rules.

As mentioned in section 3 the basis for the management is a graph rewriting system. Every manager agent represents a subset of rewriting rules. The size and the abilities of the manager agent depend on the subset. Furthermore every single rule requires resources from the system in two ways: on the one hand resources for the evaluation of the strategy and on the other hand resources for the transformation, i.e. the execution of the rule. The subset of rules which is given to a manager agent at generation fixes the amount of resources needed by the manager agent. The decision which real hardware resources are given to the manager agent is made by the generating manager agent. In order to make the whole management process flexible and to reduce the management overhead it is also possible to change a manager agent during runtime. This includes changing its realization and its rule subset. This can be done by the parent manager agent, according to the hierarchy. The precondition for a transformation of a manager agent is that the parent manager agent has the possibility to do such a transformation or, in other words, has a rewriting rule for the transformation.

6 Conclusion

The main focus of this paper was the presentation of a manager agent system for the resource management in distributed systems. The advantage of manager agents over servers in current distributed operating systems is the knowledge about the running application. The manager agents are customized to the application but not restricted to the application. Each manager agent is able to get all information about the system including information about other applications that it needs for the best possible management of the resources. To reduce the management overhead it is possible to reduce the resource requirements of the manager agents, but this reduction is reversible. If a more detailed management is needed, a manager agent can be transformed by giving it more resources. The whole resource management consists of a set of cooperating manager agents.

References

[1] B. Bryant et. al. An Introduction to Mach 3.0's XMM Subsystem. Technical report, OSF Research Institute, June 1993.

[2] Bryan Ford and Jay Lepreau. Evolving Mach 3.0 to a migrating thread model. In *Proceedings of the Winter 1994 USENIX Technical Conference and Exhibition*, pages 97–114, January 1994.

[3] Paul J. Roy. Unix file access and caching in a multicomputer environment. In *Proceedings of the Usenix Mach III Symposium*, pages 21–37, 1993.

[4] Andrew S. Tanenbaum. *Distributed Operating Systems.* Prentice–Hall International, 1995.

[5] John B. Charter, Dilip Khandekar, and Linus Kamb. Distributed Shared Memory: Where We Are and Where We Should Be Headed. In *Proceedings of the HotOS-V*, May 1995.

[6] K.Murray et. al. Experiences with Distributed Shared Memory. Technical Report TCU/SARC/1993/3, City University, September 1993.

[7] Sascha Groh. Designing an efficient resource management for parallel distributed systems by the use of a graph replacement system. In *Proceedings of the PDPTA'96*, pages 215–225, August 1996.

[8] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

[9] C. G. Harrison, D. M. Chess, and A. Kershenbaum. Mobile agents: Are they a good idea? Technical report, IBM Research Division, 1996.

[10] Sascha Groh and Jürgen Rudolph. On the efficient distribution of a flexible resource management. In *Proc. of EuroPDS'97*, June 1997.