International Journal of Science and Technology Education Research Vol. 1(7), pp. 143 - 146, December 2010
Available online http://www.academicjournals.org/IJSTER
ISSN 2141-6559 ©2010 Academic Journals

Full Length Research Paper

Cluster of heterogeneous computers: Using mobile agents for improving load balance

Mohammed A. M. Ibrahim

Department of Information Technology, Faculty of Engineering and Information Technology, Taiz University, Taiz City, P. O. Box: 6030, Republic of Yemen. E-mail: Sabri1966@yahoo.com.

Accepted 05 November, 2010

Due to the increasing interest to achieve high performance computing using cluster, which consists of heterogeneous workstations and/or personal computers (PCs) connected via a fast network. Usually programming such systems is done by applying message-passing libraries like MPI. Due to the lack of a load balancing facility, it is quite possible to overload machines on the network with tasks, and lead parallel applications to frustrated results. In this paper, the mobile agent based approach to improve the load balancing of parallel tasks in heterogeneous computing environment has been proposed. Its benefit has been demonstrated with the experimental results.

Key words: Cluster of heterogenous workstations, mobile agent, load balance, parallel computing.

INTRODUCTION

With the prevalence of powerful workstations and personal computers, and the availability of advanced networking technologies, there has been a great interest in using inter-connected network of workstations and/or personal computers for high performance parallel computing. The cost effectiveness is obvious when the system is compared with dedicated and expensive parallel computer systems. Programming such systems is usually by applying message-passing libraries like parallel visual machine (PVM) or message passing interface (MPI). Parallel programs based on these libraries are built as a collection of communicating processes or threads. To efficiently use such systems, the programmer is responsible for distributing the parallel tasks evenly among the machines in the system. More-over, the programmer can also take dynamic effects like background loads into account. Due to the lack of a load balancing facility (Byers and Mitzenmacher, 2003; Godfrey et al., 2004) it is quite possible to overload machines on the network with tasks.

Nowadays, a new distributing computing model, mobile agent, is of increasing interest. Mobile agents are autonomous software entities that are able to migrate through a heterogeneous network, example, the Internet and to perform tasks on the visited computer nodes (Zhou, 2006; Godfrey and Stoica, 2005; Kunz, 1991; Papastavrou et al., 2000). Based on such systems, we present an approach in this paper to improve the performance of the MPI parallel applications executing in heterogeneous computing environment. We first use the mobile agents to gather load status of each machine in the system, and then the lightweight nodes are selected according to the load information. These machines form the sub-computing system to run the MPI application. The experimental results indicate that it is beneficial to use our approach to improve the load balancing of the cluster.

TYPES OF MOBILE AGENT

Mobile agent is an emerging paradigm that is now gaining momentum in several fields of applications (Jonathan, 1996). A mobile agent corresponds to a small program that is able to migrate to some remote machine, where it is able to migrate to execute some function or collect some relevant data then migrate to other machines in order to accomplish another task. The basic idea of this paradigm is to distribute the processing through the network (Lange and Oshima, 1999; Karnik and Atripathi, 1998a). Several types of applications (Karnik and Tripath, 1998b) are appropriate for mobile agent technology which includes at least one of the following features: data collection, searching and filtering, distributed monitoring, information dissemination, negotiating and parallel processing.

Many mobile agent systems have been developed in the last few years, such as Aglet from IBM, Agent TCL from Dartmouth College, Voyager from object space, Concordia from Mitsubishi, TACOMA from Cornell University, and Odyssey from general magic. Most of the mentioned systems have been implemented on top of Java for its wide spreading but also for its technical advantages: machine independence and strong typing for security Figure 1.

Aglets is the underlying system for our framework, a detailed

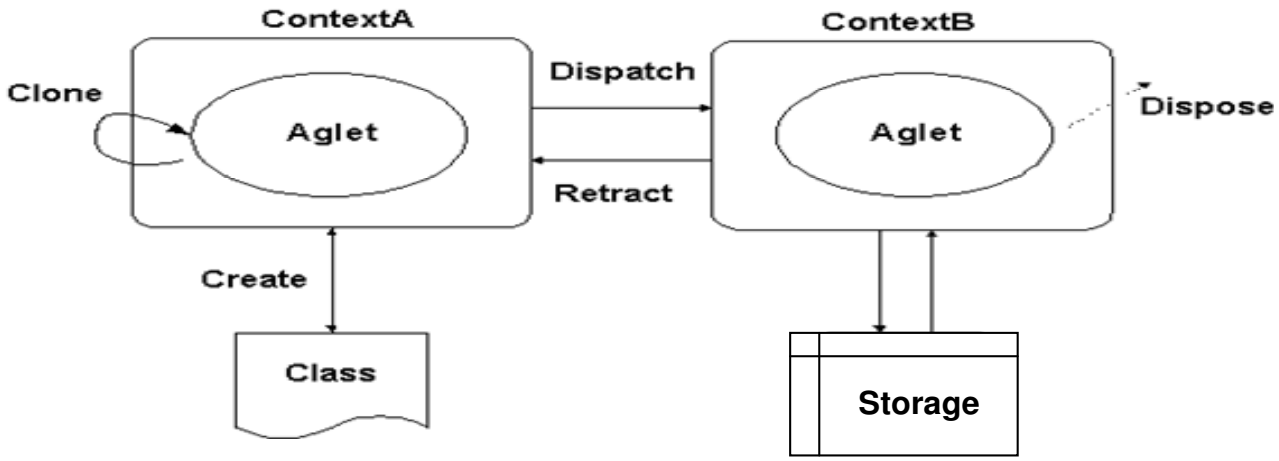

Figure 1. Machine independence and strong typing for security

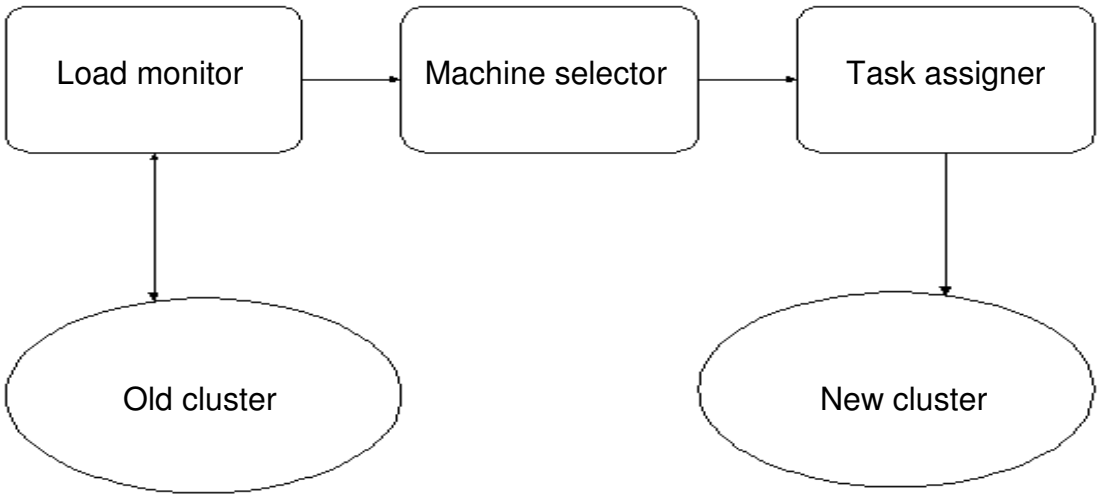

Figure 2. Architecture.

description is presented next. Aglet is probably the most famous platform of mobile agents. It is a very robust platform.

Aglets are Java objects that can move from one host to another. It is possible for them to halt execution, dispatch to a remote host, and restart executing again by presenting their credentials and obtaining access to local services and data. Aglets provide a uniform paradigm for distributed object computing. Aglet has the following characteristics: object-passing, autonomous execution, local interaction, asynchronous, disconnected operation, parallel execution, etc. Using Aglets can ease the development of distributed computing system. The Aglets API defines methods to control mobility, life cycle, travel, itinerary and security.

In a cluster of heterogeneous workstations, MPI parallel programming paradigm is often used to develop parallel applications, but we might obtain frustrated results, because of the imbalance of workload among workstations in the cluster. When starting, MPI processes machines in cluster, some workstations may be heavily loaded, thus cause uneven distribution of workload and increase the execution time of the application.

Based on mobile agent system, we create a framework to improve the load balancing of parallel tasks. It consists of the following components: load monitoring, machine selecting, and application executing (Figure 2). With the help of this approach, the parallel tasks can be running on lightly loaded nodes, whereas it improves the load balancing in the heterogeneous computing system resulting in the improvement of the performance of parallel applications.

LOAD MONITORING

Load information is usually quantifiable by a load index. Usually load values are computed using a manually-specified formula as functions of current and recent utilization levels of various resources. Resources include central processing unit (CPU), memory, disk and network, etc. Different tasks have different requirement of resources. For example, CPU-bound jobs need more CPU cycles and are sensitive to the CPU utilization, rather than network delay. Previous studies have suggested that the run-queue length best describes a workstation's loading, and many dynamic load balancing algorithms have adapted this metric (Byers and Mitzenmacher, 2003; Kremin and Kramer, 2005; Pan et al., 2005).

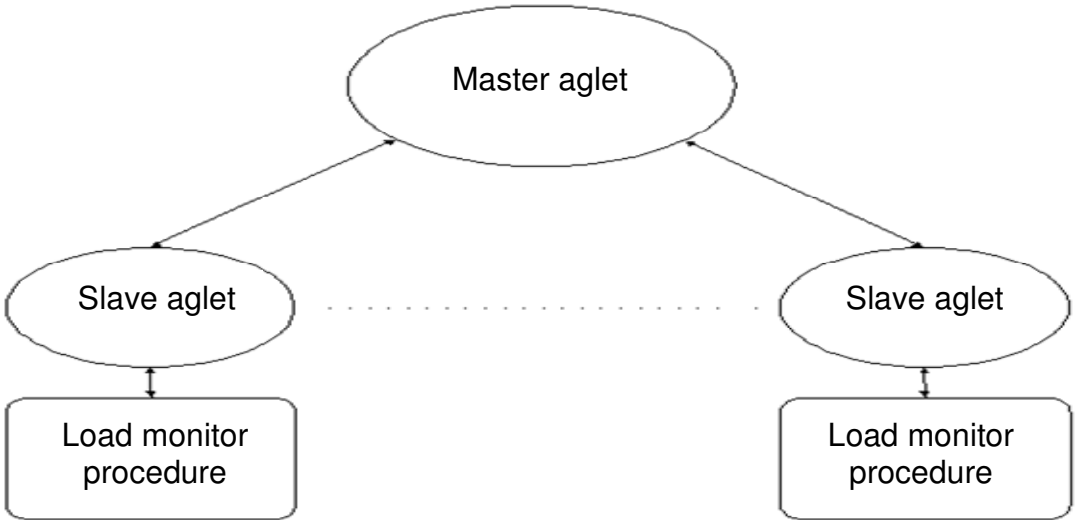

Figure 3. Load-monitoring system.

Based on mobile agent system-Aglet, we construct a load-monitoring system, its architecture is illustrated in Figure 3 and the work going in this architecture as the following steps: First we monitor the load level on every machine in the cluster, master process sends mobile agents to every workstation, where mobile agents call the native methods to measure the load level of the local machine and send it to the master. The master is responsible for selecting lightweight machines to form a new cluster to execute the MPI application.

Secondly, the master process received load information sent by mobile agents on every machine, it would execute the machine-choosing algorithm to select the lowly loaded machines according certain rules, finally form a new cluster on which the MPI application will be executed. To choose the lightly loaded machines, the following machine-choosing algorithm needs to be executed: Determine the threshold, compare the load level of each machine to the threshold.

Thirdly, it forms the new cluster in which the load of every machine is less than the threshold. A fixed threshold value is usually used in a load balancing system, but it does not work very well since the state of the system is changing all the time. For example, the threshold value determined for a system with large amount of tasks will be too large for system with small mount of tasks, resulting in the inefficiency of the load balancing mechanism. Therefore, in our framework adaptive threshold policy has been used so that the threshold can be adjusted as the global system load changes.

$$T = K \times \overline{L}$$

Where T is the threshold; K is a constant multiplier; L is the average workload.

Constant multiplier K reflects the amount that the workload can be exceeded before CPU becomes heavily loaded.

EXPERIMENTAL RESULTS

Here, the experimental results presented the benefits of our method, which uses mobile agent technology to balance the workload within heterogeneous computing system. Our experimental environment is a local area network (LAN) here in our laboratory, it consists of five nodes (Sun workstations, and server with four CPUs), which are connected with fast Ethernet (100Mbit). Aglets (mobile agent platform) are run on every node, based on each node, load monitoring system is executed to compute the load measure of every machine. The lightweight machines (under the threshold) form a new cluster on which we run application.

To test this method using mobile agent technology to improve the load balancing of MPI parallel applications in heterogeneous computing environment, we choose two typical problems: PI calculation and matrix multiplication. These applications have coarse granularity and are easy to implement in cluster environment. Our goals were to employ mobile agent computing model to improve the load balancing of parallel problems by enhancing the scheduling of tasks.

PI calculation can be divided easily into many tasks that executed independently of each other. The number of intervals needs to be a big one in order to increase the computational level of each task. The benchmark was executed with and without mobile agents respectively; the results are shown in Table 1, in which the data are average of five times execution of the program.

The experiments were conducted under both normal use and heavy use of the cluster. In the latter case, there are more heavily loaded machines than in the former case. It is clear to observe from the Table 1 that the performance has been better improved using our method comparing with just using MPI environment. This is true especially the system in heavy use. If machines are heavily loaded, assigning extra tasks will worsen the performance and delay the completion of the program. But if these nodes are excluded from the cluster, only lowly loaded machines are used to run parallel tasks,

Table 1. PI calculation.

Interval number	Normal		Heavy	
	MPI	Mobile agent	MPI	Mobile agent
5000000	0.921	0.692	1.793	1.354
10000000	1.834	1.332	3.475	2.637

Table 2. Matrix multiplication.

Matrix size	Normal		Heavy	
	MPI	Mobile agent	MPI	Mobile agent
250*250	2.056	1.481	4.663	3.648
500*500	19.283	13.072	41.438	32.572

then the load can be balanced in some extent and finally improve the performance.

Matrix multiplication is also a commonly used application in parallel test. Matrix B and blocks of matrix A were sent to every machine in the cluster. Its experimental results are shown in Table 2, including size 300×300 and size 500×500. The benefit of our approach can also be observed from it.

CONCLUSION AND FUTURE WORK

A mobile agent based approach to improve the load balancing of parallel tasks in heterogeneous computing environment is proposed in this paper. Its benefit has been demonstrated with the experimental results.

REFERENCES

Byers J, Mitzenmacher M (2003). Simple load balancing for Distributed hash tables. In 2nd International Workshop on Peer-to-Peer Systems (IPTPS), pp. 80–87.

Godfrey B, Lakshminarayanan K, Surana R (2004). Load balancing in dynamic structured p2p systems. In 23rd Conference of the IEEE Communications Society INFOCOM).

Godfrey B, Stoica I (2005)."Heterogenity and load balance in Distributed Hash Tables", IEEE INFOCOM.

Karnik N, Tripath A (1998a). “Design Issue in Mobile Agent Programming Systems,” in Proceedings of the IEEE Concurrence, Boston, Massachusetts, USA, pp. 52-61.

Karnik N, Atripathi H (1998b). “Agent Server Architecture for Ajanta Mobile-Agent Systems,” in Proceedings of the International Conference Parallel and Distributed Processing Techniques (PDPTA’98), CSREA Press, pp. 63-73.

Kremin A, Kramer J (2005). “Methodical Analysis of Adaptive Load Sharing Algorithms,” IEEE Trans. Parallel Distrib. Syst., 3: 747-760.

Jonathan D (1996). “A Mobile Agent Architecture for Distributed Information Management,” PhD Thesis, University of Southampton.

Kunz T (1991). “The Influence of Different Workload Descriptions on a Heuristic Load Balancing,” IEEE Trans. Software Eng., 17(7): 725-730.

Lange D, Oshima M (1999). “Seven Good Reasons for Mobile Agents,” Commun. ACM, 42(3): 355-395.

Papastavrou S, Samaras G, Pitoura E (2000). “Mobile Agents for World Wide Web Distributed Database Access,” IEEE Transactions on Knowledge and Data Eng., 12(5): 802-820.

Pan Y, Chen D, Guo M, Cao J, Dongarra J (2005). Parallel and distributed processing and application. Lect. Notes Comput. Sci., pp. 724-733

Zhou S (2006). ”A Trace-Driven Simulation Study of Dynamic Load Balancing”, IEEE Trans. Software Eng., 14: 1327-1341.