Resource Load Balancing Based on Multi-agent in ServiceBSP Model*

Yan Jiang[1], Weiqin Tong[1], and Wentao Zhao[2]

[1] School of Computer Engineering and Science, Shanghai University
[2] Image Processing and Pattern Recognition Institute, Shanghai Jiao Tong University
Shanghai, China
jiangyan2273@hotmail.com, wqtong@mail.shu.edu.cn,
zhaowt1982@hotmail.com

Abstract. Based on ServiceBSP model, a resource load balancing algorithm with Multi-Agent is put forward in this paper which achieves the goal of dynamic load balancing and favorable fault-tolerant. The algorithm calculates the load value according to the status of usage of resources of a node and scheduling tasks relies on the load value, while updating the load information dynamically depending on Multi-Agent. The method also can avoid frequent communications on load information. Furthermore, the paper introduces the function of agents, relations and communications among agents in details. Finally, by comparing response time and distribution of load using proposed method with other available methods such as without no load balancing and load balancing only based on the usage of CPU, the experimental simulation shows that the load balancing based on Multi-Agent possesses superior performance on response time and load balancing.

Keywords: ServiceBSP model , Multi-Agent, Load balancing.

1 Introduction

The problem of load balancing often occurs in some applications of parallel computing. Reasonable load balancing algorithm should be able to improve system throughout and reduce task response time. Many load balancing algorithms designed to support distributed system have been proposed and reviewed in the literature [1][2] [3][4], only a few have been designed or are scalable to support load balancing of all types of resources, which are inclined to cause the occurrence of unbalance of different types of resources in a node(mainly means the computer). In the distributed system, the arrival of task is a dynamic process that sometimes can not be predicted, which indicates that dynamic load balancing is required. Meanwhile, we should know the point that nodes have freedom to join in or leave the queue of providing services. So the method of dynamic load balancing should bear the characteristic of reliability.

* This work is supported by National Natural Science Foundation of China under grant number 60573109 and 90412010 and Shanghai Municipal Committee of Science and Technology under grant number 05dz15005.

Y. Shi et al. (Eds.): ICCS 2007, Part III, LNCS 4489, pp. 42–49, 2007.

ServiceBSP model combines parallel computing model BSP and the concept of service [5]. In the model, we abstract all the resources to services. In the reference [6], the author propounds the ServiceBSP model based on QoS(quality of service) that can satisfy the needs of users. Our method also possesses these advantages.

In this paper, we have developed a load balancing algorithm based on Multi-agent which successfully balances the usage of types of resources. It not only considers the usage ratio of CPU and other types of resources in the precondition of satisfying the needs of users, but also solves the problem of robustness in the process of providing services, avoiding frequent information transfer.

This rest of this paper is organized as follows: ServiceBSP model is shown in Section 2. In Section 3, the load balancing algorithm is proposed and described in details. The application of Multi-agent in load balancing is introduced in Section 4. Section 5 presents experimental simulation results, and Section 6 concludes the paper.

2 ServiceBSP Model

In view of characteristics of distributed system short of providing stable QoS, we advocate ServiceBSP model considering the advantages of BSP model [7][8]. An application is firstly divided into several tasks according to their intrinsic properties because of its loose coupled characteristic and tasks are executed in parallelism.

All the node providing services should publish information of their services to a searchable registry of services description and update it. Such information includes functional performance, physical location, availability and price etc.

Fig.1 shows a superstep of the model.

In the Fig.1, Broker Mediator is responsible to interact with Coordinating Agent. Broker assumes the responsibility to select services satisfying needs of users from the center of service registry while giving consideration to the physical location and

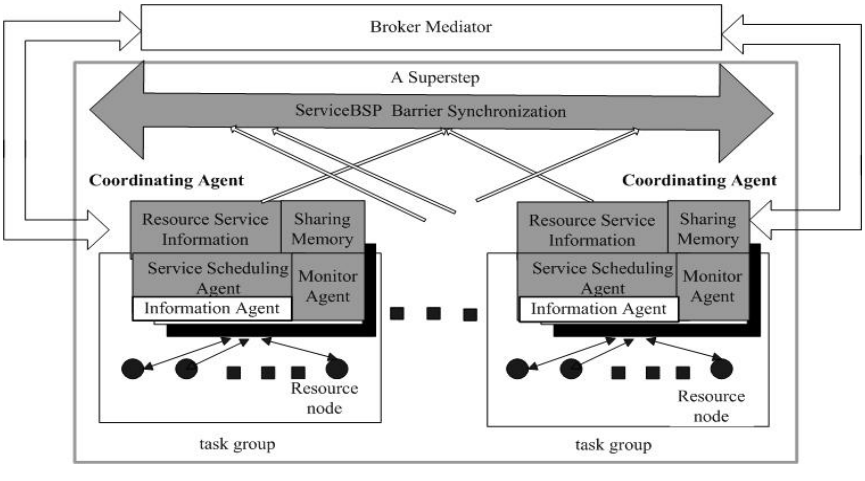

Fig. 1. A superstep of ServiceBSP model

pertinence among services. Then Broker maps the information of selected services to correspondently initialized Coordinating Agent with the help of Mediator.

3 Load Balancing in ServiceBSP Model

3.1 Resource Load Balancing Algorithm

Load balancing is demand driven issue. The dynamic load balancing we are proposing is based on a load rank of nodes, and then Service Scheduling Agent sends the new task to the currently lightest node. In our method, we assess the load of all the nodes relying on dynamic load value. We calculate the value taking such factors into account as CPU, memory resource, number of current tasks, response time, the number of network link etc.

We define Mj that represents the weight of resource j of a node that a task required, and there$\Sigma Mi=1$. In the view of different types of tasks, the weight of Mj varies according to the features of diverse tasks. For example, task with high weight of memory resource emphases the necessity of memory.

Suppose that we only include above referred attributes. A load value of a node owning only one task can be described by follow formula.

$$Vi=M1*Vcpu+M2*Vmem+M3*Vio+M4*Vtasks+M5*Vnetwork+M6*Vresponse \quad (1)$$

Vcpu, Vmem, Vio, Vtasks, Vnetwork, Vresponse respectively represent the usage ratio of CPU, the usage of memory, the number of current tasks, the number of network links, response time. So the load value of a node owning several tasks is V where $V=\Sigma Vi$. Using the current load value V, we can calculate new value when a new task is joining.

The algorithm described above is preferable algorithm with view to the dynamic load balancing of all the nodes in system and types of tasks. In addition, the algorithm is characterized as simplicity and so it would not cause extra cost of system.

3.2 Fault-Tolerant in Load Balancing

We have referred that the nodes have no restrictions to join in or leave the queue of providing services. Therefore ensuring robust and reliable services when we adopt effective method to maintain load balancing is an important issue. Our paper succeeds in solving this issue using Agent technology.

Monitor Agent would discover abrupt change while a node leaves the queue because of fault or other reasons, and then select alternative node ranked as the lightest load from the Resource Information Service. Even with extra time cost, restarting to execute the failed task is preferable to the termination of the task.

4 Application of Multi-agent in Load Balancing

In the Fig.1, we have a general description of Coordinating Agent. The relations among agents and of these agents with other modules in one task group are illustrated in Fig.2.

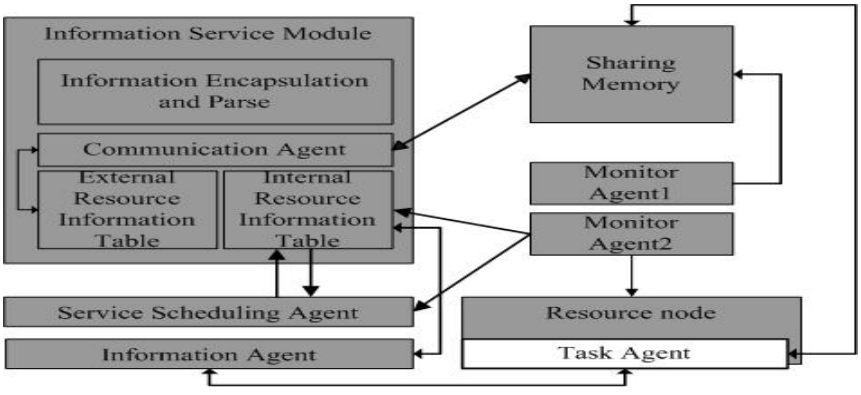

Fig. 2. Relation of Multi-Agent

4.1 Load Balancing Based on Multi-Agent

Fig.2 shows that Coordinating Agent provides such services as node information, monitoring node, communication and scheduling tasks. The interpretation of detailed functions of those modules in Fig.2 is as follows:

External Resource Information Table: The Table stores the information of other Coordinating Agent and makes preparing information of targets for the global communication following a local computation in a superstep.

Internal Resource Information Table: The Table stores the information of all the nodes in a task group including the physical location, logic ID, capability, AgentID, port and load value etc.

Communication Agent: The Communication Agent initializes the interaction with other Coordinating Agent after reading data from Sharing Memory in the end of local computation step. The information sent by Communication Agent can be understood by other Coordinating Agent through Information Encapsulation and Parse Module.

Information Encapsulation and Parse Module: The function of this module is that using XML encodes KQML communication language and describes content in forms of XML documentation after Agent sends message [11]. Similarly, it parses and analyzes the content before Agent receives message.

Task Agent: It is responsible for calculating the dynamic load value for minimizing the frequency of communication between node and Resource Information Service Module. It also reads and writes data in Sharing Memory as representative of node.

Information Agent: The works of receiving the load value of a node and monitoring the information on whether the node is alive or not are assigned to Information Agent. In order to get the valid load value Information Agent should update Internal Resource Information Table timely when monitoring changes.

Service Schedule Agent: It manages to schedule tasks to proper nodes according to the rank of load value by reading the load value from Internal Resource Information Table.

Sharing Memory: Communications in a task group are completed through Sharing Memory rather than the normal direct way. Sharing Memory is also where the data required by global communication are stored.

Monitoring Agent1: It monitors communications occurring in Sharing Memory. For example, Task Agent representative of task A in one node sends data to Sharing Memory, then Monitoring Agent informs Task Agent representative of task B in another node to read these data from Sharing Memory. In addition, it should make sure that a Task Agent has legal right to modify data in Sharing Memory avoiding bringing invalid data.

Monitoring Agent2: Monitoring Agent2 is designed to monitor the status of node for supporting robustness of ServiceBSP model. If nodes encounter abrupt failure or apply to leave the queue of providing services, Monitoring Agent2 would monitor these changes and then update Internal Resource Information Table. Finally, it selects an alternative node with lightest load to execute the discontinued task.

4.2 Agent Communication

In our ServiceBSP model, we suggest two modes of communications among agents [9][10]. One mode concerning agents in the same task group occurs in Sharing Memory. Global communication is attributable to another mode relative to different task groups, meaning direct communication from one point to another point.

The first mode of communication mainly consists of four operations: read data, write data, delete data and modify data. All the operations are supervised by Monitoring Agent.

Up to the point, we have assumed that load value of nodes serves as the criteria for scheduling tasks in order to acquire the goal of load balancing. The load value will update while a new task coming in. The Task Agent of such node that receives a new task assumes the duty to inform the dynamic load value to Internal Resource Information Table immediately, which actually ensures the load balancing.

The second mode of communication adopts the Knowledge Query and Manipulation Language (KQML) to support comparatively massive communications among Coordinating Agents [11]. We argue for an XML encoding of KQML and expressing communications content. It is required that the KQML messages parsed by the Information Encapsulation and Parse Module in Coordinating Agent of target, are separated from XML document, so we can analyze and understand the meaning of these messages.

5 Simulation and Discussion

The experimental simulation is configured with nine task queues, illustrated by the set shown in Table1.These task queues are named *S1*……*S9* and represent different queues with different characteristics including different number of big, middle and small tasks

giving more or less emphasis to CPU or memory. The characteristics of these task queues are found in Table 1. Three figures in parenthesis are representative of the numbers of big, middle, small tasks orderly. The task with higher CPU value represents the one giving more emphasis on CPU than memory, whereas, the task with higher memory value emphasizes memory.

Table 1. Task queues

Queue NO.	Total numbers of tasks	Numbers of tasks with high CPU value	Numbers of tasks with high memory value
S1	(5,5,5)	(5,5,5)	(0,0,0)
S2	(5,5,20)	(4,3,15)	(1,2,5)
S3	(5,10,15)	(5,5,2)	(0,0,13)
S4	(5,15,10)	(4,5,6)	(1,10,4)
S5	(5,20,5)	(3,6,1)	(2,14,4)
S6	(5,20,20)	(1,13,14)	(4,7,6)
S7	(15,5,15)	(0,0,0)	(15,5,15)
S8	(15,10,10)	(10,7,6)	(5,3,4)
S9	(20,5,5)	(16,3,3)	(4,2,2)

We made several measurements of response time of task execution, which can be divided into three categories: load balancing algorithm only based on CPU, load balancing algorithm based on load value, and no load balancing algorithm. Then we contrast the distribution of load of CPU and memory in a task group including five nodes when respectively adopting load balancing based on load value and no load balancing algorithm. The results of simulation are shown in Fig.3 and Fig. 4.

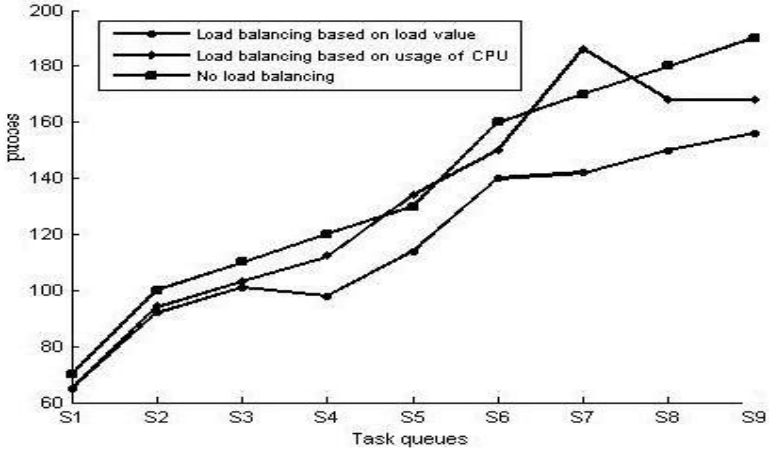

Fig. 3. Contrast of response time

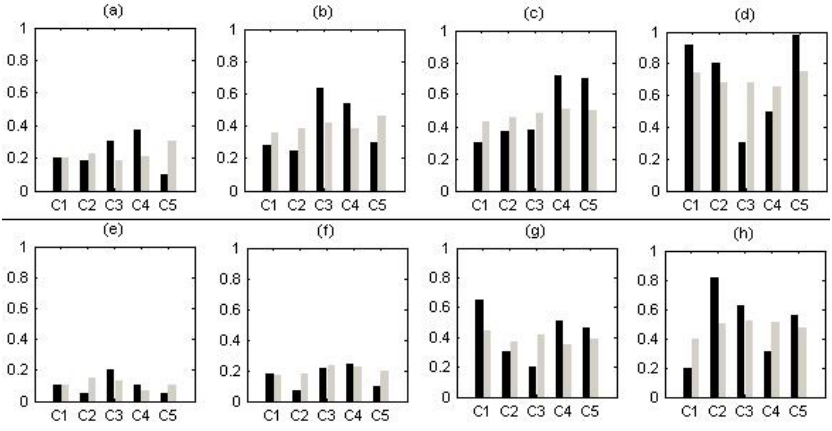

Fig. 4. Contrast of distribution of CPU and memory load

We present results obtained through Fig.3 and Fig.4. Fig.3 illustrates how the load balancing methods performed by contrasting response time. The results obtained in such model with load balancing algorithm based on load value illustrates less response time compared with adopting no balancing algorithm and only based on usage of CPU. Commonly, system with no load balancing algorithm needs longer response time than with load balancing algorithm only based on the usage of CPU. However, if most tasks in task queue with high memory value, sometimes it is better to use no balancing algorithm than the algorithm only based on the usage of CPU.

Fig.4 demonstrates the distribution of memory and CPU load. In the figure, the black and the grey cylinders respectively represent the load distribution in the condition of model with load balancing algorithm based on load value and with no load balancing. (a),(b),(c),(d) represent the distribution of CPU load of a task group including five nodes named C1,C2,C3,C4,C5 in picture while task queues S1,S3,S5,S8 assigning to this task group. (e),(f),(g),(h) present the distribution of memory load. Apparently, a system adopting load balancing algorithm based on load value ensures the balancing distribution of CPU and memory load in five nodes.

6 Conclusion

In this paper, we have proposed a load balancing algorithm based on Multi-Agent in ServiceBSP model. It ensures load balancing and satisfies the needs of users in distributed system. Our method successfully avoids frequent communications between agents when aiming at the goal of dynamic load balancing. The experimental simulation shows that it realizes dynamic load balancing and speed up response time of tasks.

References

1. Wang, Y.T., Morris, R.J.T.: Load sharing in distributed systems. IEEE Trans. Comput., vol. C-34, pp. 204-211, Mar. 1985
2. Fox, G.C.: A review of automatic load balancing and decomposition methods for the hypercube. California Institute of Technology, C3P-385, Nov. 1986
3. Ramamritham, K., Stankovic, J.A., Zhao, W.: Distributed scheduling of tasks with deadlines and resource requirements. IEEE Trans.Comput., pp. 1110-1123, Aug. 1989
4. Baumgartner, K.M., JSling, R.M., Wah, B.W.: Implementation of GAMMON: An efficient load balancing strategy for a local computer system. in Proc. I989 Int. Conf Parallel Processing, vol. 2, Aug. 1989,pp. 77-80
5. Foster, I., Kesselman, C., Nick, J.M., Tuecke, S.: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Downloadable as: http://www.globus.org/research/papers/ogsa.pdf, 2002
6. Zhu, J.Q., Tong, W.Q., Dong, X.J.: Agent Assisted ServiceBSP Model in Grids. GCC2006
7. Valiant, L.G.: A bridging model for parallel computation. Communications of the ACM, 33(8), (1990) 103~111
8. Song, J., Tong, W.Q., Zhi, X.L.: QOS-BASED PROGRAMMING METHOD IN GRID ENVIRONMENT. Computer application and software. Vol 23.No.10
9. Hyacinth S.N.: Software Agents: An Overview, Knowledge Engineering Review. Vol. 11, No 3, pp. 205-244, October/November 1996
10. Michael, R.G., Steven, P.K.: Softwareagents: Communications of the agent. ACM, 37(7):48–53,147, 1994
11. Yannis, L., Tim, F.: A semantics approach for KQML. In Third International Conference on Information and Knowledge Management, November 1994