Int. J. Communications, Network and System Sciences, **2010, 3, 59-65**
doi:10.4236/ijcns.2010.31008 Published Online January 2010 (http://www.SciRP.org/journal/ijcns/).

Hierarchical Resource Load Balancing Based on Multi-Agent in ServiceBSP Model

Bin CHENG[1,2], Yan JIANG[1], Weiqin TONG[1]
[1]*School of Computer Engineering and Science, Shanghai University, Shanghai, China*
[2]*College of Mathematics, Physics and Information Engineering, Zhejiang Normal University, Jinhua, China*
Email: *cb@shu.edu.cn, jiangyan_2273@163.com, wqtong@mail.shu.edu.cn*
Received September 18, 2009; *revised October* 20, 2009; *accepted November* 26, 2009

Abstract

Based on ServiceBSP model, a hierarchical 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 attributes of resource and scheduling tasks relies on the load value, while updating the load information dynamically depending on Multi-Agent. The method avoids 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 giving regards to 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 [1–4], only a few have been designed, or are scalable to support load balancing of all types of resource and consider the characteristics of different tasks, which are inclined to cause the occurrence of unbalance of different types of resources in a node (mainly mean the computer). In the distributed system, the arrival of task is a dynamic process that sometimes can not be predicted, which indicates that the status of load balancing of nodes is not static and dynamic load balancing is required. Meanwhile, we should know the point that nodes have freedom to choose to join in or leave the queue providing services. So adopting effective method of dynamic load balancing bearing ability of providing reliable resources is a significant research.

ServiceBSP model combines the characteristic of superstep of parallel computing model-BSP and the concept of service in currently popular technology of web service [5]. In the model, we abstract all the resources (mainly computing resources) to services. In the reference [6], the author propounds a ServiceBSP model based on QoS (quality of service) that can satisfy the needs of users.

In this paper, we have developed a load balancing algorithm based on Multi-agent which successfully balances the usage of all the types of resources. The algorithm 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 which would bring extra communication cost.

The 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 hierarchical load balancing is introduced in Section 4. Section 5 present experimental simulation results, and Section 6 concludes the paper.

2. ServiceBSP Model

In view of characteristics of distributed system short of providing stable QoS because of dynamic environment and advantages of BSP model, we advocate the Ser-

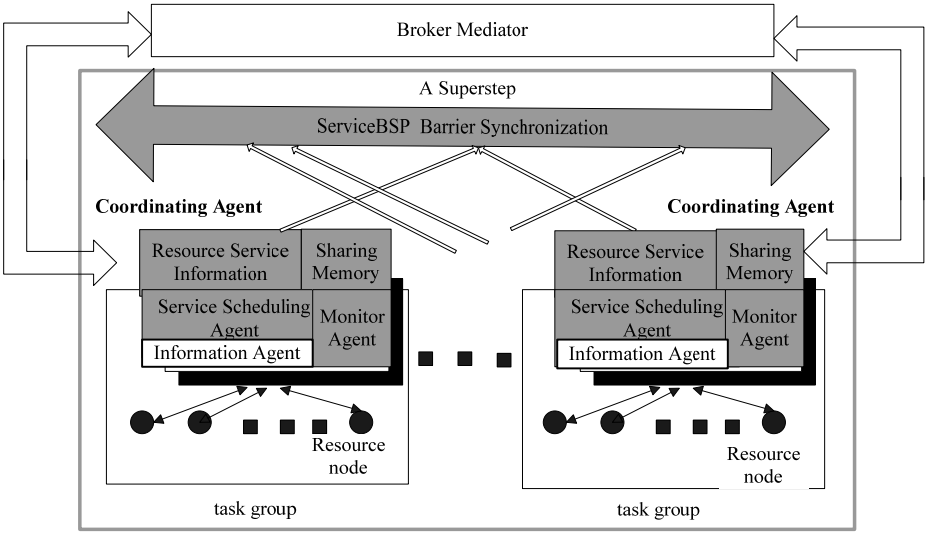

Figure 1. A superstep of ServiceBSP model

viceBSP model which introduces the concept of service to BSP model effectively [5,7]. An application is firstly divided into several tasks according to their intrinsic properties because of its loose coupled characteristic. Tasks are executed in parallel and a little communications occurs between tasks.

All nodes providing services should publish the information of their services to a searchable registry of services description and update it. Such information includes functional performance, physical location, and healthy qualities such as available, and also price.

Figure 1 shows a superstep of the model.

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

3. Resources Load Balancing Algorithm in ServiceBSP Model

Resource Oriented Load Balancing Algorithm (ROLBA) is a dynamic load balancing algorithm, which is based on a load rank of nodes. The load of all nodes is assessed by the dynamic load value which takes such factors into account as CPU, memory, network bandwidth [8–10]. ROLBA is based on the characteristic of the superstep of ServiceBSP. It is that the time of superstep is composed of the time of local job (including the calculating and memory exchange), the time of global communication and the time of barrier synchronization.

ROLBA is divided into 3 stages, like Figure 2. The first stage is information collection, which includes the static information and dynamic information of the node. The information can be used to check the imbalance of the nodes. The second stage is prediction. Predicting the time of local job of every node is based on the information collected in the first stage. Then the distribution of the threads can be made for the nodes. The third stage is thread migration. It is based on the distribution of the threads and the communication cost between the threads.

ROLBA can make node calculate the threshold value by means of investigating the local job finishing time of another nodes and estimate the balance quality itself. The Agent in the node can be used to finish the local job, broadcast the finish time all of the nodes, receive the time information from other nodes and estimate the balance quality of the node. If it is imbalance, the node will send the dynamic information to the load balancer. The responsiveness of Agent simplifies these operations.

4. Application of Multi-Agent in Load Balancing

In the Figure 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 Figure 3.

4.1. Hierarchical Load Balancing Based on Multi-Agent

Load balancing model proposed in this paper is a new method which combines the hierarchical load scheduling policy with Multi-Agent technique. It can realize the higher performance running by the mutual cooperation of Multi-Agent. The agents of each module in the load balancer, the agents in each node and the relationships

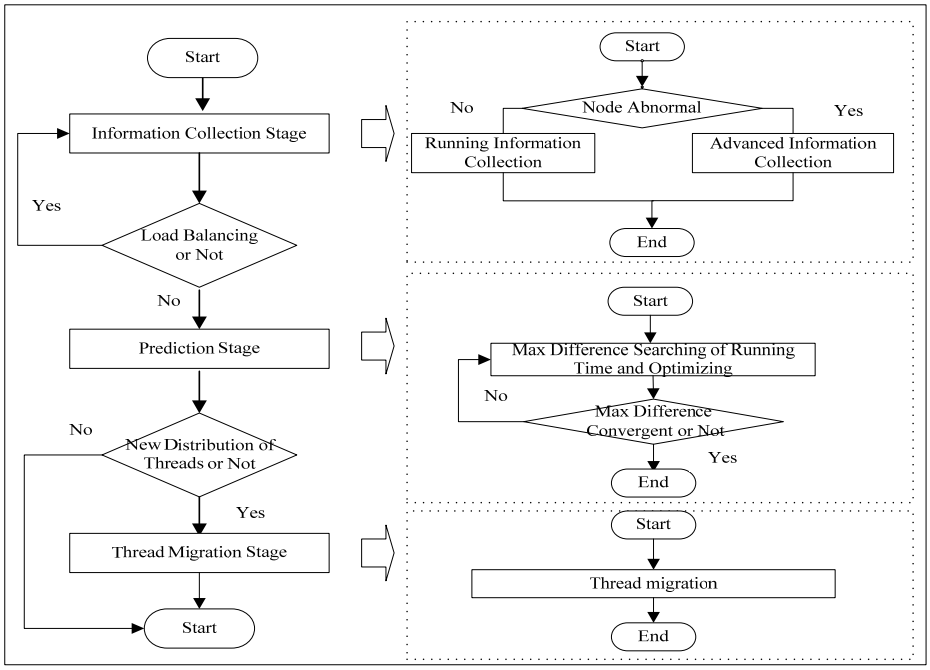

Figure 2. Resource oriented load balancing algorithm.

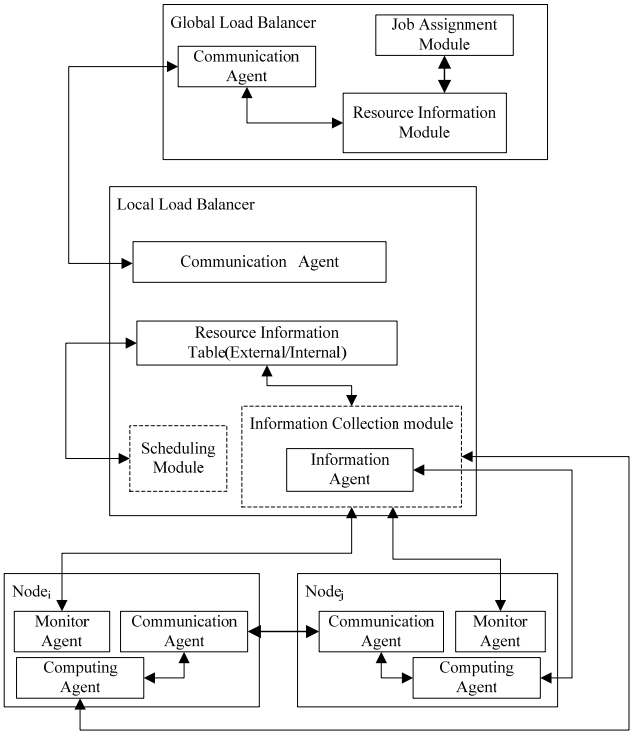

Figure 3. Relation of multi-agent.

among them are expressed in Figure 3. The scheduling process is completed by the cooperation of Global Load Balancer and Local Load Balancer. And the Monitor Agents in each node supply the local information to upper level Information Agent. The interpretation of detailed function in Figure 3 is as follows:

1) Global Load Balancer has Communication Agent. The action class ReceiveDataBehaviour is the main part like Figure 4. It receives the load information of Local Load Balancer by receive(). If the information is inconsistent with the wanted information type of node, replyNotUnderstood() will send the error information. Other-

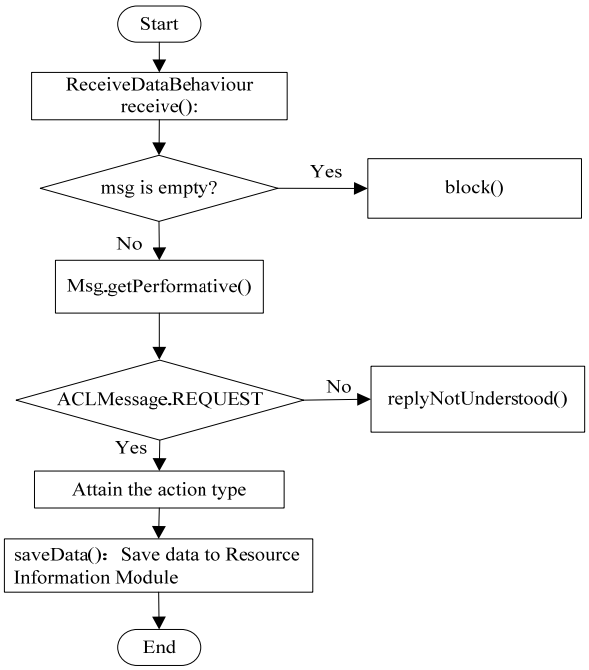

Figure 4. Work flow chart of communication agent in global load balancer.

wise, saveData() stores the received information in Resource Information Module. The information is the foundation to allocate job.

2) Local Load Balancer has Communication Agent and Information Agent.

- Communication Agent sends the load information to the agent in Local Load Balancer similar to the Communication Agent in Global Load Balancer. Besides the action classes ReceiveDataBehaviour, SendDataBehaviour is the main work. ReceiveDataBehaviour gets the data by readData(). And SendDataBehaviour sends these data by SendMessage(), after attaining the Agent list which will receive the information.
- 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. The function principle is simple. If the IP of the node, which has the Information Agent to send information, is in the Internal Resource Information Table, the information of this node can be updated by the received information NodeInfo. Otherwise, the information of this node should be added to the table.
- The Internal Resource Information Table stores the information of all the nodes in the task group including the physical location, logic ID, capability, AgentID, port and load value etc. And the External Resource Information 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.

3) Each node has 3 kinds of Agent. They are Monitor Agent, Communication Agent and Computing Agent.

- Monitoring Agent is designed to monitor the status of node for supporting robustness in ServiceBSP model. If nodes encounter abrupt failure or apply to leave the queue of providing services, Monitoring Agent would gain this information by getStatus() and then update Internal Information Table by sendLoadInfo() and sendActiveStatus(). Finally, it selects an alternative with lightest load for failed node to execute the discontinued task. Figure 7 expresses the flowchart of Monitor Agent.
- Communication Agent broadcasts the time of the local job after the completeness of local computing and receives the time information of other nodes. Two object entities, SendAgent() and ReceiveAgent (), compose the Communication Agent. Their realization process is similar to the Communication Agent in Global/Local Load Balancer.
- Computing Agent calculates $\frac{\sigma}{ave}$, where $\sigma = \sqrt{\frac{\sum_{i=1}^{N}(n-ave)^2}{N}}$ and $ave = \frac{\sum_{i=1}^{N} n_i}{N}$, and estimates whether it is more than the threshold or not. If it is, Local Load Balancer will receive the dynamic information of the job implement situation of this node.

4.2. Agent Communication

In ServieBSP model based on Multi-Agent, we suggest two modes of communications among agents [11]. 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 task in order to acquire the goal of load balancing. The load value will update with 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 Information Table immediately, which actually ensures the load balancing.

The second mode of communication adopts the Knowledge Query and Manipulation Language (KQML)

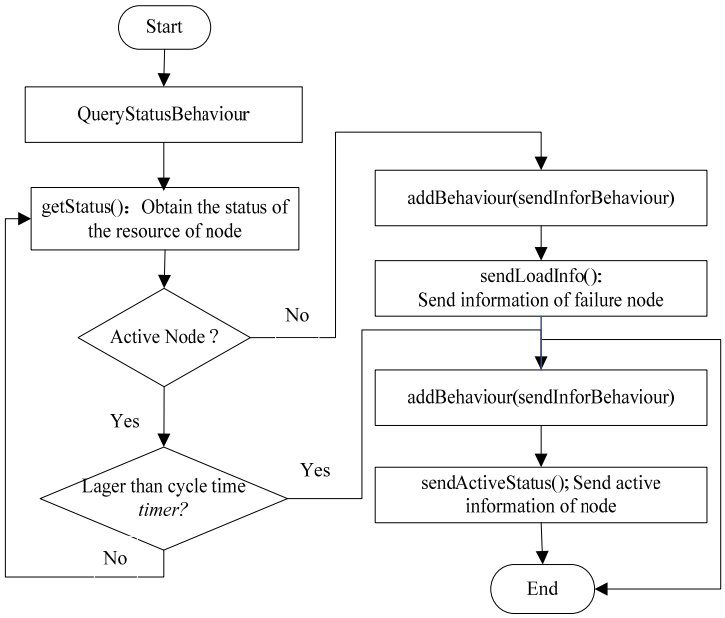

Figure 5. Work flow chart of monitor agent in node.

Table 1. Task queues.

Queue NO.	(bid,mid,small)tasks number	High CPU value	High memory value
S0	(5,5,5)	(5,5,5)	(0,0,0)
S1	(5,5,20)	(4,3,15)	(1,2,5)
S2	(5,10,15)	(5,5,2)	(0,0,13)
S3	(5,15,10)	(4,5,6)	(1,10,4)
S4	(5,20,5)	(3,6,1)	(2,14,4)
S5	(5,20,20)	(1,13,14)	(4,7,6)
S6	(15,5,15)	(0,0,0)	(15,5,15)
S7	(15,10,10)	(10,7,6)	(5,3,4)
S8	(20,5,5)	(16,3,3)	(4,2,2)

[12], a new language and protocol for exchanging information and knowledge, to support comparatively massive communications among Coordinating Agents. The communications among those agents need to know those information concerning AgentID, physical resource ID and corresponding physical address that assist the completion of searching communication targets. We argue for an XML encoding of KQML and expressing communications content. We deal with technologies enabling Agents to manage the document of XML finally. The XML Schema describing syntax principle of KQML is specified for this purpose. 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. The following is an example of a Task Agent sending load value to Information Agent:

(tell
:Sender ID of Task Agent
:Receiver ID of Information Agent
:language KIF
:content (= (weight V) (result))
:In-reply-to L
:ontology Value)

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 *S0,S1…..S8* and represent different queues with different characteristics containing 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. 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.

We made several measurements of response time of task execution, which can be divided into three categories: adopting load balancing algorithm only based on CPU, adopting load balancing algorithm based on load value, and adopting 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

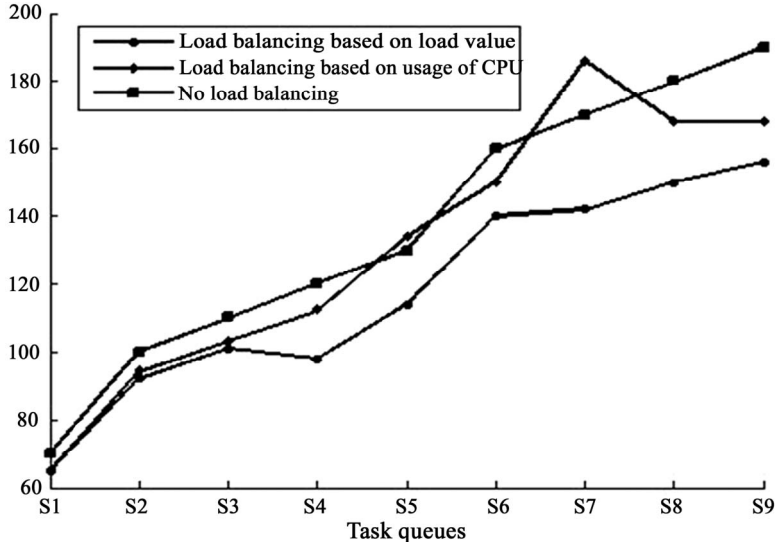

Figure 6. Response time contrast.

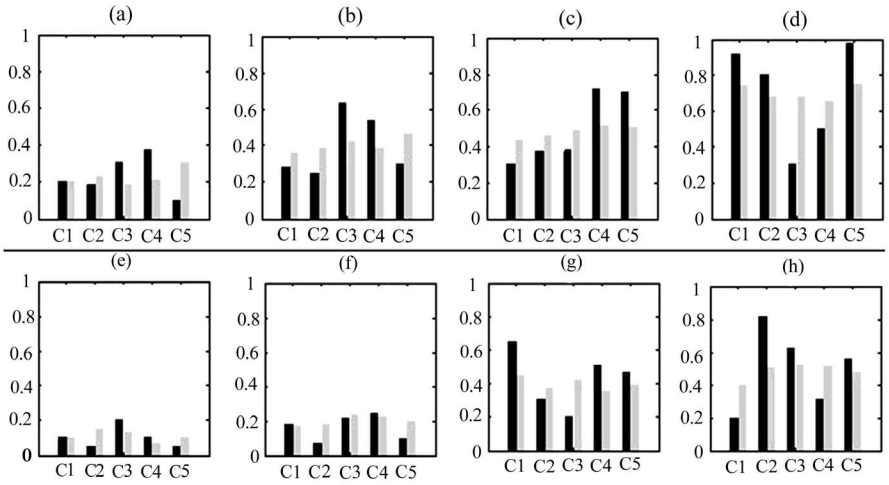

Figure 7. Load of CPU and memory contrast.

balancing algorithm. The results of simulation are shown in Figure 6 and Figure 7.

We present results obtained through Figure 6 and Figure 7. Figure 6 illustrates how the load balancing algorithm 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 commonly. 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 the usage of CPU.

Figure 7 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 in five nodes named C1,C2, C3,C4,C5 in picture while task queues S0,S2,S4,S7 assigning to a task group including five nodes. (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. So it can cut down response time.

6. Conclusions

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, making best use of characteristics of ServiceBSP model. Our method successfully avoids frequent communications between agents attributable to the

technology of Multi-agent when aiming at the goal of dynamic load balancing. The experimental simulation shows that it can solve the problem of load balancing and speed up response tasks.

7. Acknowledgment

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.

8. References

[1] Y. T. Wang and R. J. T. Morris, "Load sharing in distributed systems," IEEE Transactions on Computers, Vol. C-34, pp. 204–211, March 1985.

[2] G. C. Fox, "A review of automatic load balancing and decomposition methods for the hypercube," California Institute of Technology, Vol. 13, pp. 380–385, November 1986.

[3] K. Ramamritham, J. A. Stankovic, and W. Zhao, "Distributed scheduling of tasks with deadlines and resource requirements," IEEE Transactions on Computers, Vol. 38, pp. 1110–1123, August 1989.

[4] K. M. Baumgartner and B. W. Wah, "GAMMON: A load balancing strategy for local computer system with multiaccess networks," IEEE Transactions on Computers, Vol. 38, pp. 1098–1109, August 1989.

[5] Y. Jiang, W. Q. Tong, and W. T. Zhao, "A services selection policy for ServiceBSP model with QoS-aware in grids," International Conference on Convergence Information Technology, ICCIT2007 Conference, IEEE Computer Society, pp. 382–386, September 2007.

[6] J. Q. Zhu, W. Q. Tong, and X. J. Dong, "Agent assisted ServiceBSP model in grids," In the Fifth International Conference of Grid and Cooperative Computing 2006, pp. 17–21, October 2006.

[7] J. Song, W. Q. Tong, and X. L. Zhi, "QoS-based programming method in grid environment," Computer Application and Software, Vol. 23, No. 8, pp. 37–41, August 2006.

[8] K. -J. Liu, X. S. Liu, and C. -S. Zuo, "A task differenced scheduling algorithm(TDSA) on resource's load-balancing," Journal of University of Electronic Science and Technology of China, Vol. 33, No. 5, pp. 562–565, May 2004.

[9] J. Jiang, M. X. Zhang, and X. K. Miao, "Study on load balancing algorithms based on multiple resources," Acta Electronica Sinica, Vol. 30, No. 8, pp. 1148–1152, August 2002.

[10] J. X. Zhou, W. M. Zheng, and G. W. Yang, "Adaptive dual-threshold dynamic load balancing system based on migrating," Vol. 40, No. 3, pp. 121–125, March 2000.

[11] M. R. Genesereth and S. P. Katchpel, "Software agents," Communications of the ACM, Vol. 37, No. 7, pp. 48–53, July 1994.

[12] Y. Labrou and T. Finin, "A semantics approach for KQML-A general purpose communication language for software agents," In the Third International Conference on Information and Knowledge Management, pp. 447–455, November 1994.

