TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.3, March 2014, pp. 2003 ~ 2013
DOI: http://dx.doi.org/10.11591/telkomnika.v12i3.4568

Agent-based Dynamic Adaptive Cluster Load Balancing Model

Tianshu You, Wenhui Li, Zhiyi Fang*, Chunfei Zhang, Guannan Qu
College of Computer Science and Technology, Jilin University, Changchun, China
*Corresponding author, e-mail: fangzy@jlu.edu.cn

Abstract

In a cluster environment, the identification and the classification of the requested task is a prerequisite for choosing the appropriate load balancing algorithm and improving the processing capacity of the network. As the existing load balancing strategies cannot relate tasks and load balancing algorithm intelligently, this paper present an Agent-based dynamic adaptive load balancing model. Through the identification and classification of the requested task, the model selects corresponding load balancing algorithm to reach the load balancing in cluster. This model has a distinct advantage over other load balancing schedule strategies on response time.

Keywords*: load balancing, agent, cluster, task classification*

1. Introduction

With the rapid development of Internet, the network access and the data traffic is growing exponentially, the load capacity of a single server system cannot meet the actual demand, and using cluster technology to reach load balancing in the server application system provides an important idea to solve this problem. Cluster technology is using general computer hardware and common software to construct a high-performance computing platform and service platform, to solve the problems encountered in large-scale scientific computing, engineering and commercial applications [1]. Load balancing is a server group that is formed by multiple servers in a symmetrical manner, each server in the group has the equivalent status, all of them can provide external services individually, without the aid of other servers. Through a load sharing technology, send the external request evenly to the symmetrical structure server, the server receives the request and responds customers' requests independently [2-3]. The goal of load balancing is to allocate its symmetrical tasks according to the performance of the processor, so as to minimize the execution time of the application, and accurately assess the work load of the server, assign a new requested task to each server [4].

The academia and the industry have already proposed and actualized many load balancing mechanism [5], such as round-robin scheduling (Round Robin), weighted algorithm, the least-connection strategy and the hash table. We have discussed various types of load balancing algorithms in Document [6] however, most load balancing algorithms only designed a single task type; in the document [7], based on the load balancing node's type, the existing methods are divided into four categories, i.e. Client-based method, distributor-based method, server-based method and DNS-based method; documents [8] discussed the advantages and disadvantages of various load balancing algorithms in detail. However, with a long-term analysis and study of the documents, I believe that most scholars mainly concentrated on the load balancing algorithm, but they failed to select the appropriate scheduling strategy according to the type of the task, so how to select the appropriate scheduling strategy according to certain application become the focus of this study.

Agent is a new computer model in the field of artificial intelligence, the main function of Agent is continuity and autonomy. It is able to perceive the changes of outside world as well as changes of its own status, and reacts automatically [9]. Agent is first developed and released by Massachusetts Institute of Technology (MIT), no consensus has been achieved by experts and scholars on the definition of Agent by far, and experts in different fields have different definitions understandings of it and definitions given by them differ too. Wookridge and Jennings proposed

Received August 27, 2013; Revised September 20, 2013; Accepted October 12, 2013

an authoritative definition that was generally recognized by experts and scholars in 1995, which includes two sub-defined [9-10]:

Definition 1 (weak definition): Agent is a software-based or hardware-based computer system; it has characteristics as autonomy, reactivity, social and initiative etc..

Definition 2 (strong definition): Addition to its characteristics that was mentioned in the weak definition, the Agent also has the emotional characteristics of human beings, such as: knowledge, beliefs, obligations, intentions and so on.

Reference [10-12] are using Agent technology in Intelligent Control System, Software Design Pattern and Autonomous and Semiautonomous System.In this paper, based on the fully study of domestic and foreign load balancing theory, and also combine the Agent technology in the field of artificial intelligence, propose an Agent-based dynamic adaptive cluster load balancing model, to select the appropriate load balancing algorithm through the automatic classification and identification of the requested task, so as to achieve the purpose of the dynamic adaptive load balancing of cluster, In this model, according to the actual situation, the requested tasks are mainly divided into WEB request, the video stream processing and parallel and distributed computing. The load balancing algorithm is adopted in response to the above mentioned three types of tasks respectively, i.e. weighted least scheduling algorithm with higher efficiency, adaptive genetic algorithm and general diffusion algorithm.

2. Model Structure

2.1. LVS Cluster Structure

Linux Virtual Server LVS (Linux Virtual Server) can complete load balancing height based on IP layer and content request distribution, and implement these methods in the Linux kernel, constitute a group of servers to be a scalable, highly available network server cluster. LVS is a virtual service with high scalability and stability made of load balancing (LB) server based on Linux operating system and Real Server (RL) that can support any TCP/IP platform [13], in LVS cluster, the server cluster's structure is transparent for users, the network services provided by user access the cluster just like the access to a high-performance, highly available server, the general architecture of the LVS cluster is shown in Figure 1.

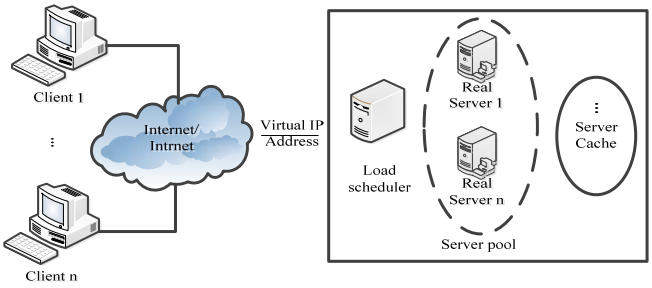

Figure 1. The General Architecture of the LVS Cluster

The architecture consists of three parts:

(1) Load balance: it is the front-end machine to external of the entire cluster; it is responsible for sending the user's request to a group of servers to execute, but from the point of view of the user, the server from one IP address. It can be load balancer based on IP load balancing technology, and can also be the load height based on the content request distribution, as well as a combination of the both;

(2) Server pool: it is a group of servers truly implement users' requests; the implementation service includes Web, MAIL, FTP, VOD (Video on Demand) services and so on;

(3) Backend storage: it provides a shared storage area for the server pool; this makes it easy for the server pool to have the same content, and provides the same services.

The current implementation for the load balancer is basically based on the IP load balancing technology, and the scheduling algorithm implemented in LVS is static [14]. This

paper, with the idea of LVS, proposes a dynamic load balancing scheduling mechanism based on request content.

2.2. Agent-based Dynamic Adaptive Cluster Load Balancing Model

In order to improve the system performance, enlarge the network bearing capacity and shorten respond time ,the structure of Agent-based dynamic adaptive cluster load balancing model is shown in Figure 2, the model includes Task Agent, Master Agent, task management and scheduling Agent, execution Agent, real-time data acquisition Agent and knowledge base share several parts.

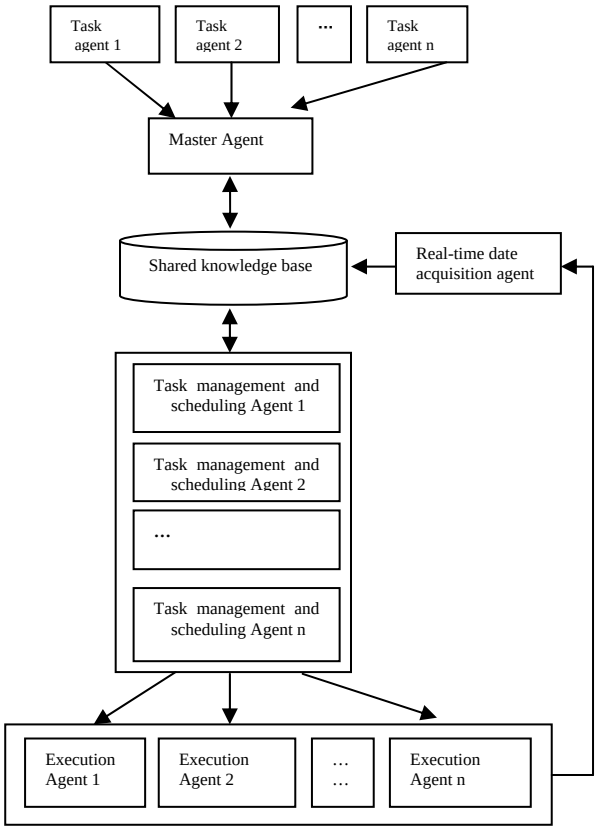

Figure 2. Agent-based dynamic adaptive cluster load balancing model

The basic working principle can be described as follows:

(1) First, when a requested task n arrive at the system, the system generates the corresponding Agent n automatically , and then analyzes the request tasks data packet, collects related data information of the task and transmits to the integrated Agent, follows the situation of network communication at the same time, provides raw data for task management and scheduling Agent.

(2) Integration Agent orderly combine and integrate various kinds of data sent by each task Agent, generates the corresponding task information. Through the preprocessing algorithm of Agent, Integration Agent converts each data into a unified format that can be identified by the system, and stored in the shared knowledge base.

(3) A shared knowledge base is basically a data storage area, like other data storage area, which itself without judge or control functions, just be responsible for recording the data of each node, storage requests information about the tasks and save load balancing scheduling algorithm which adapt to the task; but it also has its own unique characteristics, the data must be stored in specific storage area with certain format and certain data right, in order to provide multi-level information for multi-level load balancing scheduling. The shared knowledge base in

this model includes the following sections: Request task table, load balancing scheduling algorithm table and server load information table.

(4) Task scheduling and management Agent is the core of the whole system, it is the key factor to decide the completion effect of scheduling tasks. It is based on the type of task in the task table in the shared knowledge base through reasoning mechanism to query available scheduling algorithm generates decision-making results in the load scheduling algorithm table and promptly sends to the corresponding execution Agent. When a task scheduling and management agent cannot make decisions, by the communication consultative mechanism between Agent, it can send one task to multiple task scheduling and management Agent at the same time to do distributed decision-making, and finally through preprocessing algorithm transfer the handle information into unified decision-making results, and release to the appropriate execution Agent.

(5) Executive Agent receives directives issued by the task scheduling management Agent, assigns tasks to the appropriate real server to handle based on the load balancing scheduling algorithm and the load information of each server.

(6) Real-time data acquisition Agent is responsible for acquiring real-time load information of the real server, such as CPU utilization, memory utilization, network utilization, hard disk I/O access rate and the current server connection rate. This information is stored in the shared knowledge base for decisions.

In this model, the collaboration between the agents has task-sharing and result-sharing two characteristics. Information communication using the blackboard means of communication, achieved by the shared knowledge base. Task-sharing collaboration feature reflected among similar agents, such as among task Agent, task management and scheduling Agent and Executive Agent. They have the same functional role, when it is needed to play their roles, several Agents with same type will decompose the task, and bear their own separate part of the work, finally, summarize the final results. Results-sharing collaboration feature reflected in Task Agent collecting the information and transfers it to Integrated Agent to handle, after pretreatment of Integrated Agent, transfers to management and scheduling Agent.

2.3. Shared Knowledge Base Construction

Share knowledge base consists of three parts; they are request task table, load balance scheduling algorithm table and server load information table. The following describes various parts of the structure.

(1) Request task table (RTT)

The data in the request task list is consisted by the data acquired by task Agent after analyzing and sorting the request data package, each task request correspond a record in the table. It is the basic to choose which load balancing scheduling algorithm to use in next step, the basic results are shown in Table 1 below:

Table 1. Request Task Table

Task No.	Task Type	Task Name	Task Status
001	1	WEB request	Undone
002	2	Video stream processing	Undone
003	1	WEB request	Undone
004	3	Parallel and Distributed Computing	Undone
…	…	…	…

In this model, the request type of task roughly divided into three categories, namely WEB request, Video stream processing and Parallel and Distributed Computing. In the future, if there is a new type of request to join the system, simply add the related item of algorithm in load balancing scheduling algorithm table.

(2) Load balancing scheduling algorithm table (LBST)

To simplify the operation, this model divides user's requested tasks into three categories, each category uses a fixed dynamic load balancing scheduling algorithm, which uses static written way to achieve. It mainly stores related load balancing scheduling algorithms that can be used for different applications, and can be either static load balancing scheduling

algorithm or dynamic load balancing scheduling algorithm, its basic structure is as follow in Table 2:

Table 2. Load Balancing Scheduling Algorithm Table

Algorithm No.	Algorithm Name	Applicable task type	Algorithm Type	Scheduling Deployment Link
001	Weighted Least scheduling algorithm	1	Dynamic	WLC algorithm realization
002	Adaptive Genetic Algorithm	2	Dynamic	SJG algorithm realization
003	General diffusion algorithm	3	Dynamic	GDA algorithm realization

(3)server load information table(LIT)

Server load information table is mainly used to record the real-time load information of back-end real server, accomplished by real-time data collection Agent, the server load information recorded in this mode includes CPU utilization, memory utilization, network utilization, disk I / O access rate and current server connection rate, the basic structure is shown in Table 3:

Table 3. Server Load Information Table

Node ID	CPU Utilization	Memory Utilization	Network Utilization	Disk I/O	Server Connection Rate
1	0.35	0.24	0.32	0.36	0.10
2	0.32	0.21	0.30	0.37	0.12
3	0.30	0.15	0.31	0.25	0.15
…	…	…	…	…	…

The model table uses XML as the data information carriers, XML can effectively support the Agent access to data sources, such as the XML representation format of server load information is as follows:

```
<?xml version=”1.0” encoding=”utf-8”?>
 <loadinfo>
    <node id=1>
       <cpu>0.35</cpu>
       <memory>0.24</memory>
       <net>0.32</net>
       <disk>0.36</disk>
       <link>0.10</link>
    </node>
    <node id=2>
       <cpu>0.32</cpu>
       <memory>0.21</memory>
       <net>0.30</net>
       <disk>0.37</disk>
       <link>0.12</link>
       </node>
       ……
 </loadinfo>
```

3. Algorithm Design

3.1. Agent Kernel Design

In this paper, the design of Agent kernel uses the mode component structure, and its structure is shown as Figure 3.

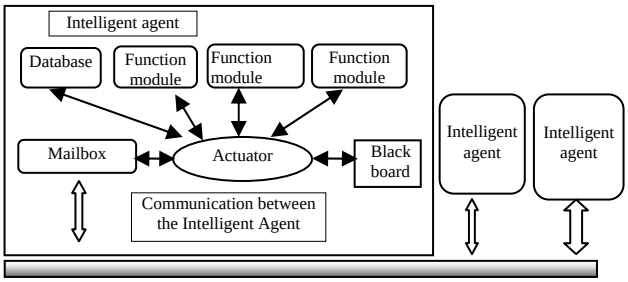

Figure 3. Intelligent Agent Kernel Structure

Each Agent is constituted by a common agent kernel and many functional modules; Agent kernel has internal database, mailbox, blackboard, actuator sections. Among them: Internal database contains the Agent's own information, information of the target collection and model of the world; Mailbox provides the communication between Agent and environment as well as communication between the certain Agent and other Agent; the blackboard provides the communication between various internal functional modules of Agent; the actuator complete the message distribution, the control over the performances of functional module. Each functional module is relatively independent entity, and run parallelly after the executive machine starts, works of different functional modules coordinated through the blackboard, modules like perception, action, reaction, modeling, planning, communication, decision-making are added to Agent in the form of functional module.

In this model, message delivery between multi-Agent uses the blackboard approach. Blackboard's main functions include the following aspects:

(1) storing functional modules, including name of the module, to complete the tasks;
(2) storing intermediate results of calculations, to avoid duplicated;
(3) storage requests between modules to achieve cooperation between modules.
Specific algorithm is shown as follows:

```
public class BlackBoard
{
        private int stub_count;
        private System.Collections.ArrayList itemList;
        public bool addItem(BlackBoardItem item);
        public bool modifyItem (BlackBoardItem oldItem, BlackBoardItem newItem);
        public bool deleteItem(BlackBoardItem item);
        public BlackBoardItem getItem(int stub_id);
        public int getNextStub();
        public string  getModelName(String task);
public class BlackBoardItem
{
            public string fromModelName;
            public string toModelName;
            public int messageType;
            public int stub_id;
            public string content;
}
}
```

In multi-Agent system, the cooperation Engine is needed. cooperation engine is responsible for querying collaborator model, looking for Agent which could cooperative solving the request problem. Agent collaborators model store other Agent model, including duties, skills, resources and goals. Cooperation engine to complete the task of the code is as follows:

```
public class CooperationEngine
{
        System.Collections.ArrayList recognizance;
        public void exec(Task task)
        {
            for (int i=0;i<recognizance.Count;i++)
           {
             AgentModel     model=(AgentModel)recognizance.get(i);
             if (model.canExec(task))
Agent=ServiceLocator.getAgent(model.getName());
        Goal goal=new Goal();
        Goal.add(task);
        sendMsg(Agent,goal);
        }
  }
}
```

3.2. Task Management and Scheduling Agent Design

The core parts of load balance are task management Agent and scheduling Agent, the flowchart of design the load balancing scheduling algorithm is shown in Figure 4.

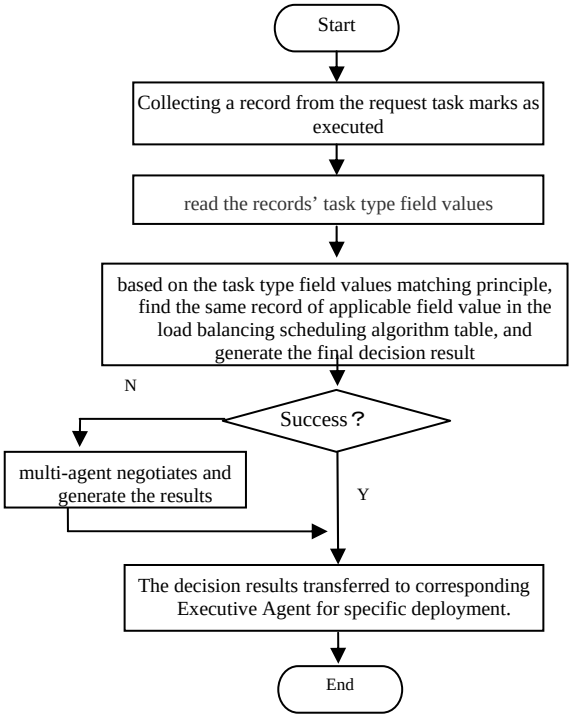

Figure 4. Task Management and Scheduling Agent Internal Scheduling Algorithm Flowchart

Algorithm is executed as follows:

(1) Read one task status from the request task table of share knowledge base and mark it as not executed;

(2) Change the task status of this record as executed;

(3) Read the task type field values of the record (which "1" represents "WEB Request Task", "2" represents "Video Stream Processing Task", "3" represents " Parallel and Distributed Computing Tasks") ;

(4) Based on the task type field value matching principle, find the same record of applicable field value in the load balancing scheduling algorithm table, and generate the final decision result for the Executive Agent to call;

(5) If a single Agent cannot make right decision, then start the multi-Agent negotiation mechanism, work out the decision-result of the task request together;

(6) The decision-result is transferred to the corresponding Executive Agent for specific deployment.

3.3. Real-time Data Acquisition Agent Design

Real-time data acquisition Agent is responsible for interacting with the executive Agents, timely collection of the real server nodes in real time load information, which mainly include CPU utilization, memory utilization, network utilization, disk I/O access rate and the current server connection rate. Then according to the load calculation method to calculate the size of the load node, and decide whether to update the server load information table entries, if found by comparing the acceptable range, then don't do treatment. Acquisition strategy using a polling mechanism, once collected automatically at regular intervals. In the case of server load information changed little, the polling mechanism may waste some system resources. However, if the time intervals set reasonable, real-time information for the server change reflects the more timely. The flowchart of the algorithm is shown in Figure 5.

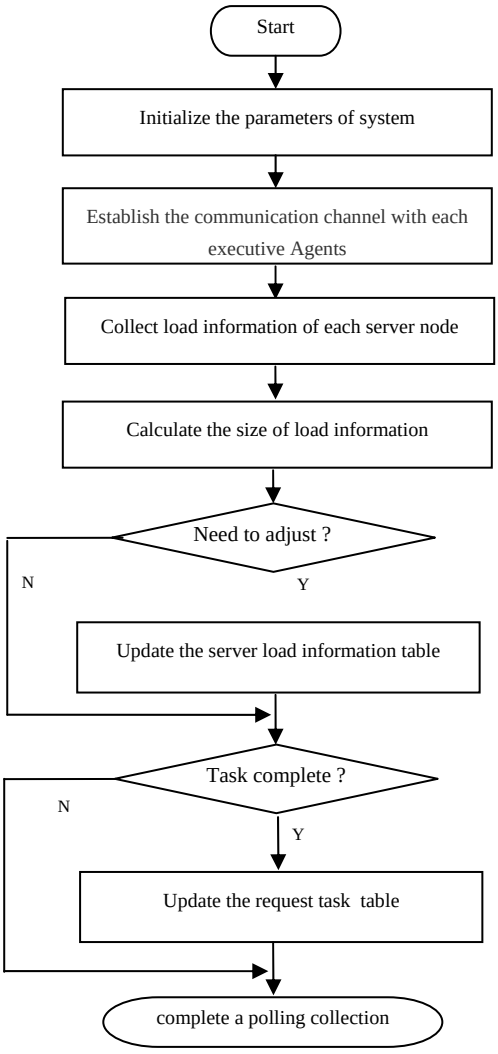

Figure 5. Real-time Data Acquisition Agent Algorithm Flowchart

Algorithm is executed as follows:

(1) initialize the system parameters;

(2) establish communication with each executive Agents;

(3) collect the real load information of each server node;

(4) calculate the size of load information for each server node;

(5) if the load information of server node does not changes or changes within a tolerable range, then do nothing;

(6) otherwise, update the server load information table in the shared knowledge base, and timely interactive with task management and scheduling Agents to adjust timely;

(7) check whether there is a request task has been completed, if so, then update requests task table; otherwise, do nothing.

(8) complete a polling acquisition.

3.4. Knowledge Rule Set and Reasoning Machine Design

In the intelligent system, the definition method of knowledge rule set and the design of reasoning machine are the key parts to achieve its intelligence, in this model the knowledge rule set and reasoning machine will be packaged inside the Agent, as one part of the function module,the knowledge rule set uses production representation to represent the knowledge, and reasoning machine uses forward reasoning algorithms.

(1) Knowledge Representation

The following is the BNF description of the productions:

<Production>:: =<premise> → <conclusion>

<Premise>:: =<simple condition> | <compound condition>

<Conclusion>:: =<facts> | <operation>

<Compound condition>:: =<simple conditions> AND <simple condition> [(AND <Simple condition>) ...]

<Simple conditions> OR <a simple condition> [(OR <simple condition>) ...]

<Operator>:: =<operator name>[(<arguments>...)]

System knowledge rule set includes the integrated data set and rule set two parts.

the structure of General Dataset :: = <Fact id, Fact name, Used by, Active>

which: Fact id : represents the fact code that used for indexing;

Fact name: represents the name of the facts;

Used by :represents which rule is applied to the fact;

Active: represents that the facts are known.

The structure of rule set :: = <Rule name, Premise, Result, Active>

which: Rule name represents name of the rule;

Premise represents the prerequisite of the rule;

Result: represents the rule conclusion or action;

Active: represents the rule is available.

if the rule established then Active = 1 else Active = 0. Set the Active of general data to 0 before reasoning in the system, and set the Active of rule base to 1.

(2) Reasoning Machine Design

Use forward reasoning model to design deduce (), recall (), testify (), remember (), step forward () and other function inference.

Definition 1. Forward reasoning function: function recall (fact: string): Boolean

Continuously choose available rules from the rule set, each time choose an available rule, add the all conclusions of the rule's after pieces that not included in facts to the facts, and expand it.

Definition 2. Function: function recall (fact: string): string

Determine whether a fact of variable fact is in the fact table, if so, return fact, otherwise return nil.

Definition 3. Function: function testify(rele: string): Boolean

Determine whether a rule's front piece that contains all the facts of variable rule is in the facts table, if so return true; otherwise return false.

Definition 4. Function: function remember (new: string): string

Determine whether a fact of variable fact is in the fact table, if so, return nil, otherwise add new to the head of the table and return the fact of new.

Definition 5. Function use then (rule: string): Boolean

Determine whether a rule's after piece that contains all the conclusions of variable rule is in the facts table, if so, return false; otherwise add the conclusions that not included to the facts.

Definition 6. Function step forward (rules: array of string): Boolean

Scanning the rules in the set, if found an available rule in rules, that a rule's front piece that contains all the facts is in the facts table, add all the conclusions of the rule's after piece that are not included in facts to the facts, and return true, if there is no available rule in the rules, return false.

Forward reasoning algorithm framework:

Procedure deduce

Match the rules in the set with the current contents of the database, if succeed, implement the rules:

Otherwise, match with the next rule.

If step forward (rules) then

Begin

Call step forward (rules), add the conclusion of the rule to the database.

call deduce.

End

4. Model Analysis

As we all know, the optimal load balancing algorithm is very difficult to achieve, it is an NP complete problem. Generally, an excellent load-balancing algorithm is only for certain specific request task to achieve maximum effect. Therefore, although there are more researches for different task request on cluster load balancing algorithm, but there is no algorithm is suitable for all different application environments. The load balancing model is based on the idea of request task classification, divide all task requests into three categories, namely WEB-based request, video stream data processing-based request and parallel and distributed computing, and select the most efficient and appropriate load balancing scheduling algorithm for each type of task, achieve the organic integration of a variety of load balancing scheduling algorithms. Fundamentally improve the task response time of the cluster system. But also through the introduction of Agent technology to achieve the dynamic feedback and adaption, provide technical support for dynamic load balancing implementation. The model uses the Multi-Agent collaboration mechanism to achieve load balance of cluster system. Task Agent automatically achieve the task through analyzing the request packet classification, task management and scheduling start Agent reasoning mechanism to lookup load balancing scheduling algorithm table based on the result of classification, find the suitable scheduling algorithm to get decision-result, and handled to the Executive Agent to implement scheduling algorithms specifically, real-time data acquisition Agent can monitor the real-time server load condition, when there is load tilt, it can be dealt with it in time, and meet the basic requirement of the cluster dynamic load balancing algorithm.

For the cluster load balancing scheduling algorithms of different application environments and task types, the current research of domestic and foreign scholars have been more mature, so after fully studying the Intelligent Agent technology, the model will be applied to a variety of load balancing algorithms integration process, it has obvious advantage not only on the response time of the task but also the resource utilization of cluster system over other load balancing scheduling policies.

5. Conclusion

This paper proposes a dynamic adaptive Agent-based load balancing cluster model based on task classification idea and intelligent Agent technology. Through the analysis of task data packet, the requested task is divided into three categories, each category corresponds to an efficient load balancing scheduling algorithm, through Multi-Agent communication and consultation mechanism to achieve load balancing cluster system together, thereby increase the cluster system throughput and reduce the response time of the task, provide a new idea for the integration of a variety of load balancing algorithm. The disadvantage is that this model just divides the task roughly into three categories, and each type of task only set one load balancing

scheduling algorithm fixed, the model is less intelligent. How to divide the request task more carefully and how to choose the optimal algorithm to deployment when each task corresponds to a variety of load balancing algorithms will be the key contents in future.

Acknowledgements

This research is supported by China Jilin Province Natural Science Funds under program of parallelization and dynamic scheduling model of CPU/GPU cooperative high performance computing cluster (201215189).

References

[1] Shijue Zheng, Wanneng Shu. Research on Genetic Algorithm for Load Balancing in VOD Cluster. *Computer Application and Research*. 2007; 20(1): 107-109.
[2] Han Yun, Yu Jiong. An Task Scheduling Algorithm Based on Load Balance. *Microelectronics & Computer*. 2010; 27(8); 201-204.
[3] Honghui Niu, Weijun Chen. A New Framework of Effective Loading-Balancing. *Microcomputer Information*. 2009; 25(1-3): 184-186.
[4] Qin Liu, Julong Lan. Design and Implementation of Dynamic Load Balancing in LVS. *Computer Technology and Its Applications*. 2007; 9(1): 116-119.
[5] A Bestavios, ME Crovella. *Distributed Packet Rewriting and Its Application to Scalable Server Architectures*. Proceedings of the 6th IEEE International Conference on Network Protocols. 2010; 290-297.
[6] Kin C, H Kameda. *Optimal Static Load Balancing of Multi-class Tasks in a Distributed Computer System*. Proc of the 10th International Conference on Distributed Computing Systems. 1990; 562-569.
[7] Carlellin V, Colajanni M. *Algorithm for Load Sharing in Distributed Web-server Systems*. Proceedings of the 19th IEEE International Conference on Distributed Computing Systems. 1999; 528-535.
[8] Carlellin V, Colajanni M. Dynamic Load Balancing on Web-server Systems. *IEEE Internet Computing*. 1999; 3(3): 28-39.
[9] Mingzhu Zhang, Zhenxing Li. Cooperation Mechanism of Wide-area protection System Based on Multi-Agents. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(1): 195-202.
[10] Guangming Zhou. Design of Intelligent Control System Based on Agent. *Electronic Design Engineering.* 2010; 18: 130-132.
[11] Xinjun Mao, Zhiming Chang. Agent-Orient Software Design Patterns. *Computer Engineering and Science*. 2011; 33: 72-78.
[12] Qing-Quan Liu. Coordinated Motion Control of Autonomous and Semiautonomous Mobile Agents. *TELKOMNIKA Indonesian Journal of Electrical Engineering.* 2012; 10(8): 1929-1935.
[13] Chunyong Yang, Shaoping Cheng. Routing Optimization for Network Load Balance Based on Improved Ant Colony Algorithm. *Computer Engineering*. 2010; 36(8): 4-6.
[14] Bingli Guo, Shanguo Huang. An Integrated Routing Scheme with Load Balancing Consideration. *Journal of Beijing University of Posts and Telecommunications.* 2009; 32(4): 1-5.