Approach of Dynamic Load Balancing in Network Monitoring

Yong Wang(✉) and Ying Wang

School of Computer Science and Technology,
Guang Dong University of Technology, Guangzhou, China
{wangyong,wangying}@gdut.edu.cn

Abstract. Load balancing is an important technology to support parallel service in network. This paper presents a dynamic load balancing architecture with master/slave model. The architecture is used in a cluster monitoring system in order to use and distribute resources effectively. Dynamic library management mechanism is used to overly manage messages, and allocate tasks using path dispatch strategy. A network monitoring system of enterprise intranet based on web services has been developed which can provide short response time, high efficiency and efficient usage of resources.

Keywords: Web service · Dynamic load balancing · Cluster system

1 Introduction

With the development of automatics, communication and computer technology, network monitoring system is more important for supporting applications through networks. The monitoring system has been through the two phases of total distributed and field bus system, and gradually evolves into a modernized system with the features of digitization, network and intelligence [1]. However, a large scale network monitoring needs to balance the effectiveness of application system [2]. Technology of load balancing has been widely applied in network applications, and also, is researched in supporting network monitoring. Recently, monitor system of server cluster has become a hot topic for research in load balancing area [3, 4]. Many researchers have put forward solutions to some hot issues of CPU, memory and I/O resource [5].

Load balancing is aimed to provide a kind of effective and transparent approaches to improve the service capacity and throughput of servers and networks [6]. Load balancing can accomplish following tasks: solve network congestion, nearby provide service, achieve independent geography; it can provide better access quality, shorter response time, higher efficiency of server and other resources for users, and it can avoid single-point failure at key nodes in networks [7, 8]. Load balancing is a policy that let multi-servers or multi-links to do burdened computing or I/O assignment as a whole, so that network bottlenecks can be eliminated in a cheap way, and networks' flexibility and reliability can be promoted [9].

Z. Pan et al. (Eds.): Transactions on Edutainment XIII, LNCS 10092, pp. 273–280, 2017.
DOI: 10.1007/978-3-662-54395-5_23

There are two types of load balancing technologies, software and hardware approach. There are different advantages and disadvantages between them in service capacity, response-time, etc. [10] The solution of hardware load balancing is more efficient and expensive than software approach [11]. Suitable load balancing strategy can make a series of equipment accomplish tasks as a whole, eliminate or avoid the existing network loading problem of uneven distribution, congested data and bottleneck with long responding time. The load balancing of 2nd, 3rd, 4th and 7th of the OSI reference model respectively requires corresponding load-balancing strategies [12].

There are two considerations in measurements of the difference between good and bad load balancing strategies and their difficult degrees to accomplish: the first is load balancing algorithm; the second is its ability of detecting status of network system. Considering different types, servers, capabilities of service requests, and random selection that leads to uneven load balancing, corresponding load-balancing algorithms (Round Robin, Weighted Round Robin, Radom, Weighted Random, Response Time, Least Connection and Flash DNS) [13–16], which can correctly reflect servers' capacity and networks' statuses, are needed, in order to distribute load to each server more properly. Although there are many load-balancing algorithms that have proper data distribution for servers, without the strategy's detection capacity towards the status of network systems, once a server or a equipment responsible of load balancing and server network has a failure, abundance of service requests may be lost because the load balancing equipment still distribute data to the server; as a result, required uninterrupted availability cannot be achieved in this case [17]. Therefore, good load balancing policy should have the detecting capacity to respond network failure, server system failure and application service failure, such as Ping Detection, TCP Open Detection and HTTP URL Detection [18].

This paper proposes a model of network monitoring based on dynamic load balancing using web service. It adopts master/slave monitoring and dynamic library management mechanism to supervise message transmission and reception, and arrange tasks according to the path assignment policy. In the process of load balancing the system, a dynamic linked list is built among every slave; the threshold for the linked list of dynamic library provides real-time supervision of load on slaves; centralized scheduling helps collect loading information, balance loading policy; loading transmission and linked list management together contribute to overload processing.

2 Network Monitoring System

2.1 Architecture of the System

The main objective of the Network Monitoring System (NetMonitor), which supports dynamic load balancing, is to capture and gather, transmit and query, analyze, comment and diagnose the data of the health of the server in the enterprise LAN; meanwhile, the system is able to calculate the load balance according to the loading of each hosts and systems, and thus improvement of the system resources utilization and reduction of average response time of the tax can be guaranteed maximally. As a result, by refining

and modeling the objective, the following figure shows the architecture of NetMonitor (Fig. 1).

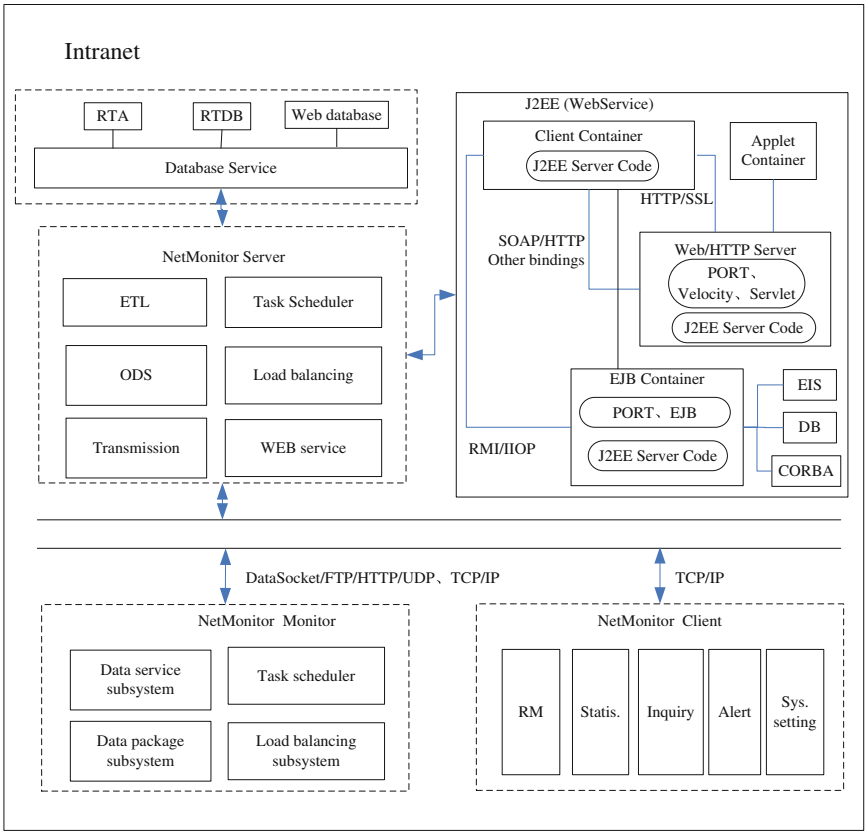

Fig. 1. Architecture of NetMonitor in enterprise intranet.

NetMonitor is developed by using C/S + B/S pattern, and it takes full advantages of both of the patterns. NetMonitor has four main components as follows:

(1) NetMonitor Monitor. NetMonitor Monitor is responsible for collecting every parameter of the performance of CPU, I/O and memory, and it can extract necessary data according to the performance of the supervised object and its parameters; despite its dual roles as a server and a client, the Monitor is also a load balancer. Due to enterprise network security mechanism, the protocol of data transmission should have configurability and high compatibility, such as those support TCP/IP, FTP, HTTP and Data Socket.

(2) NetMonitor Client. NetMonitor Client is useful in on-line monitoring, condition recognition, historical records inquiries, hierarchical alarm and trend prediction. The system can manage permissions to clients due to sever settings, and system administrator can adjust parameters of load balancing of the server by setting them

manually and letting the system auto-regulated due to the system's working state, so that load balancing services may process effectively.

(3) NetMonitor Server. NetMonitor Server is responsible to operations like unpack, classify, extract and process test, control and load balancing data of different levels sent from collector. NetMonitor Server includes data and control services mainly. Data services compose by task scheduling, load balancing and WEB service programs; control services, which are based on J2EE multi-tired architecture of Web Services, compose by transmission, ODS and ETL service programs.

(4) NetMonitor Database Service. NetMonitor Database Service is divided into real-time database, Web database and real time analyzer (RTA). Real time database alleviates pressure of Web database, improves data processing time and throughput capability, and stores operation information from every server; RTA focuses on analyzing and treating data of load balancing, in order to achieve load balance.

2.2 Structure of Monitored Object

The monitored object in the NetMonitor is server which contains CPU, Networking Adapter, I/O devices, sensors, etc. The structure of monitored object is defined below:

```
Server
{   CPU              :   running time
    Network Adapter  :   throughput
    Input device     :   throughput
    Out device       :   throughput
    Sensor           :   used efficiency
}
```

The information of monitored object is dynamic and collected by NetMonitor Monitor. Netmoitor Client is responsible for sending monitoring information to NetMonitor Server.

2.3 Policy of Message Communication

In communication of subsystems, accuracy and instantaneity of message traffic effectiveness of operation of application system and load balancing mechanism. Therefore, considering dynamic change of functions in practical application, dynamic library management mechanism is proposed to optimize system performance. Figure 2 is the flow chart of system message data processing:

System processes communicate through sharing memory and buffer. It has following three developments compare to other similar NetMonitor's communication mechanisms:

(1) Message data can be divided into three types: test data, control data and load balancing data. Test data does not require high instantaneity, and allows delay; load balancing data needs to be classified with high instantaneity, so that leveled message mechanism can be made to ensure effective operation of system message policy.

(2) Inter-process communication (IPC) combines asynchronous communication and synchronous communication mechanisms: asynchronous communication mechanism can reduce waiting time due to synchronization, so it is used to collect test data;

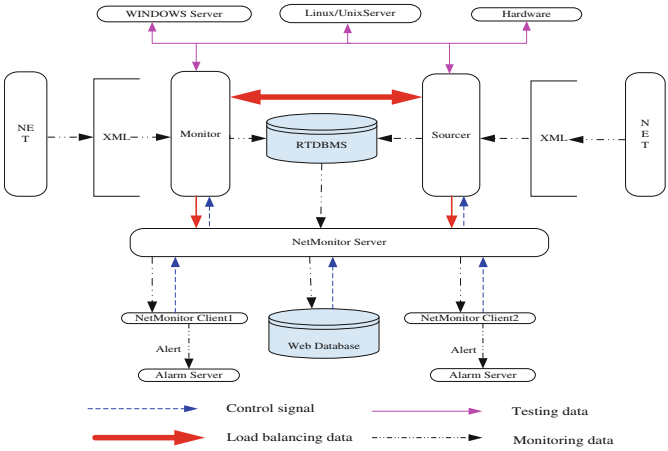

Fig. 2. Data processing model of NetMonitor.

synchronous communication mechanism can assure effective operation of the monitoring data and load balancing data requiring high accuracy and instantaneity.

(3) Aim at the phenomenon of memory-to-memory copy in buffer, message transmission and reception and the server must apply dynamic library management mechanism. Dynamic library management mechanism can allocate, send/receive and free up buffers, and it uses path dispatch instead of data dispatch. To avoid communication bottleneck in dispatch, service processes must deal with time-consuming message acceptation and handling, instead of accepting already dealt message by dispatch processes.

2.4 Policy of Load Balancing

Load balancing is an important measure to use and allocate resources efficiently, through data diffluent and load transfer; it can be normally divided into static load balancing and dynamic load balancing. Static load balancing usually indicates mapping and scheduling problems. Dynamic load balancing, however, continuously allocates workload to each computational nodes, according to real-time loading of each nodes, so low-loaded nodes will be allocated more work. Dynamic load balancing usually includes centralized dynamic load balancing and distributed load balancing [19].

Centralized dynamic load balancing can be applied in compute-intensive tasks with few processes; application of distributed load balancing aims at high-required surge conditions, fine-grained tasks and multi-child processes of cluster system.

The distributed load balancing mechanism has four main strategies:

(1) Message strategy. This strategy uses event-driven centralized information strategy, that is, uses node to send soft interrupt signal to RTA, and RTA accepts and manage global-sloped node events.

(2) Migration strategy. This strategy uses freelist migration strategy, lets RTA preset two changeable thresholds – listused (L_u) and listfree (L_f) – according to servers' different performance index.

Definition 1: Load Factor Host's loading condition and trend in a recent time slice. Smaller value of load factor represents lighter loading. The formula shows below,

$$L(m) = L_u \times \frac{Max\_Load\_List}{Max\_Load} + L_f \times \frac{Load\_List_1 - Load\_List_2}{MAX(Load\_List_1, Load\_List_2)}$$

For example,

(i) $L(m) < L_u$ && $1 - L(m) > L_f$ *When system is underloaded, it requests to accept the task, re-press* $1 - L(m)$ *to add freelist and sort;*
(ii) $L(m) < L_u$ && $1 - L(m) <= L_f$ *When system is suitable, it does not accept nor request to migrate tasks;*
(iii) $L(m) < L_u$ *When system is overloaded, it request to migrate tasks. The system can distribute pressure choosing one or more nodes in freelist, and decrease chosen nodes in value; when the value is smaller than* L_f*, it exits freelist.*

Recently, most of the research focuses on the transmission of the address space of the largest amount of data in the process, and it has total four strategies: Eager (dirty), Copy-on-Reference (COR), Flushing and Precopy. Since this system message policy adopts dynamic library management mechanism, relations among every node can be predicted according to path dispatch, Flushing strategy is applied.

(3) Selection and Positioning strategies. The system – which uses mentioned dynamic library management mechanism to allocate, send, free up and accept information of load balancing, starting or started using freelist mechanism and buffer – can discover overloaded node early, locate migration node precisely and shorten migration time.

3 Performance Analysis and Evaluation of Load Balancing

Average response Time is the main index for performance evaluation of load balancing. Due to the algorithm of the distributed dynamic load balancing in the essay, Rate (j) should be defined first. Rate (j) indicates the maximum weight of the largest resources (I/O, Memory, CPU) of the node j.

Definition 2: The weight of the resources of node j is the ratio of Rate (j) and the largest Rate (j) in the system, which is,

$$W_R(j) = \frac{Rate(j)}{MAX_{I=1}^{p}(Rate(j))},$$

(p indicates the total number of nodes).

Taking an AIX minicomputer and three Windows 2000 Server as a cluster system, the experiment requested CPU time 10 s, and its active applying period is T = 0.1 s, and the average response time changes followed requests arrival rate.

Figure 3 shows that Load Balancing of NetMonitor has low efficiency when requests arrival rate is low, it is even lower than efficiencies of No Load Balancing and Random Strategy; however, as requests arrival rate goes up, NetMonitor Load Balancing's efficiency becomes larger, and it's response time keeps much lower than others'. Reasons are:

(1) The central controlled information alleviates synchronous cost and load of communication network through message hierarchy strategy and buffer dynamic management technology, and it reduces response time and resources waste.
(2) A freelist protecting list improves searching and locating speed, protects tasks migration form jittering, and it reduces system's pressure.
(3) Dynamic library management mechanism avoids freelist from fragmentation and memory-to-memory copy in buffer, and it promotes response time of the system.
(4) The intelligence of NetMonitor Load Balancing allows it to consider real-time trend of servers, if it adds up or decreases loading, so it is suitable for Flushing migration strategy.

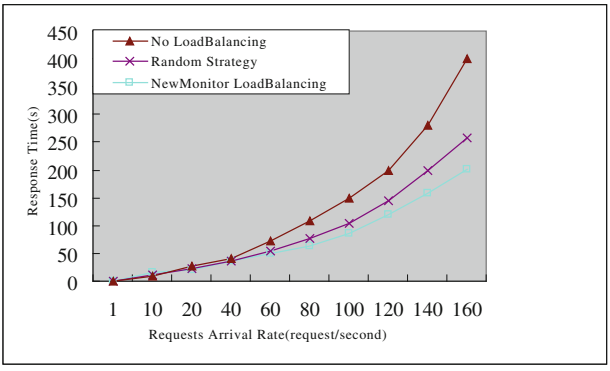

Fig. 3. Response Time (s) vs Requests Arrival Rate (request/second).

4 Conclusion

The experimental data and performance analysis show that the NetMonitor, which supports dynamic load balancing, has powerful monitoring function like analysis, query and diagnosis, and it can use resources effectively, reduce average response time and promote system's loading capacity, so NetMonitor can be used in cluster system of enterprise Intranet.

There are two future works, one will focus on real-time system auto-regulating threshold, and the other will aim at improving and exploring system's expansibility and security.

References

1. Pattanaik, P.A., Roy, S.: Performance study of some dynamic load balancing algorithms in cloud computing environment. In: Spin, pp. 619–624 (2015)
2. Yang, H., Zhang, Y., Zhou, Y., Guo, G.: Algorithm of dynamic load balancing based on transparent computing. Comput. Eng. **32**(13), 133–135 (2006)
3. Yin, A., Wang, B., Hu, X., Yang, W.: MintRoute-HNLB: a novel wireless sensor network routing protocol with load balancing. Comput. Sci. **37**(5), 77–80 (2010)
4. Qiao, Y., Bochmann, G.V.: Using diffusive load balancing to improve performance of peer-to-peer systems for hosting services. In: Chrisment, I., Couch, A., Badonnel, R., Waldburger, M. (eds.) AIMS 2011. LNCS, vol. 6734, pp. 124–135. Springer, Heidelberg (2011). doi: 10.1007/978-3-642-21484-4_15
5. Chen, Y., Liang, G., Li, T.: Load balancing scheme for network intrusion detection system. Comput. Eng. Appl. **47**(7), 116–117 (2011)
6. Yin, A., Wang, B., Hu, X., Tang, Q.: Wireless sensor network routing protocol for load balancing. J. Huazhong Univ. Sci. Technol. (Nat. Sci. Ed.) **38**(1), 88–91 (2010)
7. Wang, P., Huang, Y., Li, K., Guo, Y.: Load balancing degree first algorithm on phase space for cloud computing cluster. J. Comput. Res. Dev. **51**(5), 1095–1107 (2014)
8. Liu, T., Sun, Y.: Clustering algorithm for heterogeneous wireless sensor networks and load balancing based on shortest path. Comput. Sci. **41**(10), 169–172 (2014)
9. Pan, S., Zhang, L., Liu, S.: Adaptive load balancing algorithm based on future load predicting. Syst. Eng. Electron. **37**(6), 1384–1390 (2015)
10. Katyal, M., Mishra, A.: A comparative study of load balancing algorithms in cloud computing environment. Comput. Sci. **6**(1), 25–36 (2014)
11. Randles, M., Lamb, D., Taleb-Bendiab, A.: A comparative study into distributed load balancing algorithms for cloud computing. In: IEEE International Conference on Advanced Information Networking & Applications Workshops, pp. 551–556 (2010)
12. Rathor, V.S., Pateriya, R.K., Gupta, R.K.: Comparative study of load balancing algorithms through virtual machine scheduling in cloud computing environment. Int. J. Comput. Appl. **91**(9), 20–24 (2014)
13. Lin, C.C., Chin, H.H., Deng, D.J.: Dynamic multiservice load balancing in cloud-based multimedia system. IEEE Syst. J. **8**(1), 225–234 (2014)
14. Domanal, S.G., Reddy, G.R.M.: Optimal load balancing in cloud computing by efficient utilization of virtual machines. In: Sixth International Conference on Communication Systems and Networks, pp. 1–4 (2014)
15. Panwar, R., Mallick, B.: A comparative study of load balancing algorithms in cloud computing. Int. J. Comput. Appl. **117**(24), 33–37 (2015)
16. Sharma, R., Kanungo, P.: Dynamic load balancing algorithm for heterogeneous multi-core processors cluster. In: International Conference on Communication Systems and Networks Technology, pp. 288–292 (2014)
17. Tiwari, A., Kanungo, P.: Dynamic load balancing algorithm for scalable heterogeneous web server cluster with content awareness. In: Trendz in Information Sciences & Computing, pp. 143–148 (2010)
18. Guo, C.: A dynamic load-balancing algorithm for heterogeneous web server cluster. J. Comput. Inf. Syst. **8**(13), 5287–5294 (2012)
19. Deng, Y., Lau, R.W.H.: Dynamic load balancing in distributed virtual environments using heat diffusion. ACM Trans. Multimed. Comput. Commun. Appl. **10**(2), 136–156 (2014)