DCABES 2008 Proceedings

2008年国际电子商务、工程及科学领域的分布式计算和应用学术研讨会论文集

2008 International Symposium on Distributed Computing and Applications for Business Engineering and Science

Volume I

July 27~31, 2008, Dalian, China

主　编：须文波
Editor in Chief: Wenbo Xu

副主编：刘　丹
Associate Editor: Dan Liu

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 要

随着计算机技术的不断发展，分布式并行以及高性能计算对科学、工程技术、经济管理等领域的重要性日益突出。一年一度的 DCABES 国际会议已经成为该领域有影响的学术会议。2008 年 DCABES 国际会议论文集共收录近 300 篇学术论文，内容涉及：分布式并行计算、网格计算、数值计算、网络技术与信息安全、信息处理、信息管理系统、电子商务、图像处理、Web 技术、无线传感技术、智能计算等。对相关研究领域中的本科高中级学生、研究生、教学及科研人员均有较大的帮助。

图书在版编目（CIP）数据

2008 年国际电子商务、工程及科学领域的分布式计算和应用学术研讨会论文集. 上册：英文 / 须文波主编.
—北京：电子工业出版社，2008. 7
ISBN 978-7-121-07018-1

I. 2… II. 须… III. 分布式计算机—计算机应用—国际学术会议—文集—英文 IV. TP338.8-53

中国版本图书馆 CIP 数据核字（2008）第 096214 号

责任编辑：秦绪军　　潘　娅
印　　刷：北京季蜂印刷有限公司
装　　订：北京季蜂印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：880×1 230　1/16　印张：93.5　　字数：3000 千字
印　　次：2008 年 7 月第 1 次印刷
定　　价：398. 00 元（上、下册）

Design and Implementation of Autonomic Computing System for Server Cluster

Wenjie Liu[1] Yuntao Zhou[2]

1 Department of Software and Theory (School of Computer), Northwestern Polytechnical University, Xi'an, 710072, China

Email：liuwenjie@nwpu.edu.cn

2 Department of Lab and Device Management, Northwestern Polytechnical University, Xi'an, 710072, China

Email：zhouyuntao@nwpu.edu.cn

Abstract

Aiming at the problem that servers cluster cannot rapidly deploys system and maintenance cost is high, on the basis of studying the autonomic computing and analyzing the features of cluster system, this paper designed and implemented an autonomic computing system for servers cluster. The designed system can mask the complexity of hardware and auto-control the cluster system, which realized the software and hardware cooperative work. Agent technique is used to collect the cluster status information and report it to autonomic computing system automatically, which then realizes the system self- check and self-recovery. After describing the system structure and modules functions, the autonomic computing features are described, which can auto-deploying the system and recovering the fault, then reduce the cost and realize the system self-management.

Keywords： Servers cluster; Autonomic Computing; Multi-agent System; Cooperative work; Self-Management

1 Introduction

In the past, when enterprises users build their application system, there are only two kinds of architecture that can be selected. One is based on mainframe; the other is based on client/server cluster. The first kind of architecture has high performance, high flexibility and high availability, but it costs much in buying hardware devices, and there are many functions that are never used, therefore resources are wasted. The second kind of architecture allows users to add hardware devices according their needs, but this kind of system is not real cluster, and it lacks of necessary availability and manageability, which makes users pay much in application upgrading and management.

With the occurrence of network, a new architecture, which has high performance/price comparison, comes into being and becomes the mainstream - distributed cluster architecture. When users want to accomplish their tasks, this architecture provides more computing ability and transparent data access ability, at the same time, realizes the high performance and high reliability.

But one coin has two sides, there are many problems in this kind of structure, such as, after hardware fault occurs, the SPOF (Single Point of Fault) status lasts long, which will take long time to check the cluster system configuration, to find out the fault point, to install software patches or find the difference of hardware/software. So it cannot deal with the increased load. To solve the above problems, higher availability management and more effective hardware utility are needed.

IBM's senior vice president, Paul Horn, proposed the concept of autonomic computing in March 2001. He noted

that autonomic computing system must have four features: self-configuration, self-optimization, self-healing and self-protection. The aim of autonomic computing environment is to make IT system reach the level of RAS (High Reliability, High Availability, and High Serviceability) [1-2]. The core of the concept is to use software to auto-control the complex hardware system therefore reduce the management cost.

This paper takes the servers cluster as management object, designed and implemented an autonomic computing system. By using this system, servers cluster can configure itself, find the hardware fault and recover itself automatically, which realizes the autonomic features and accomplish the aim of self-management.

2 Cluster system analysis

Cluster system contains many homogeneous or heterogeneous computers, which are connected to accomplish specified tasks cooperatively. It provides high performance services continuously. Servers cluster are a group of independent servers, which can be regarded as a single server in the network, and can be managed as a single system. The single system provides the client station with high reliable service. A cluster system contains many servers that share data storage, each server communicates with others across inner LAN. When fault occurs in one of the servers, the application running in this server will be taken over by another server automatically. Severs cluster can provide quite high performance none-stopping service, because each server can take on part of the computing task. As cluster system owns the performance of many servers, the system computing ability will increase also. At the same time, when fault occurs in one of the servers, system can separate this server from others by using special software, realize the new load balance by the load shift mechanism among servers, and simultaneously notify the administrators by signals [3-4].

Practice proves that cooperative work in the cluster has much higher computing ability than mainframe, super computer and fault tolerance system, moreover owns the lower cost.

But there are also disadvantages in cluster system as following [5].

It cannot rapidly deploy software: For large-scale cluster system, to deploy software rapidly on each node of the system needs much time and human power.

It cannot rapidly shift roles: Application requires that each server can shift to another role in different time. For example, in data center, sometimes more web service is required; sometimes more video service is needed. This requires servers to shift rapidly between two services to meet the needs of different time.

Maintenance costs are high: With the increase of nodes count, the fault chances also increase, and system recovery costs become high. How to reduce the time of recovering system and maintenance costs becomes a problem in cluster system.

This paper designed and implemented an autonomic computing system for cluster system, which can solve the above problems and reduce the management cost. The designed system has the following features:

1) Rapidly deploy the system software;
2) Automatically recover system when fault occurs;
3) Optimize system automatically;
4) Protect system when illegal invasion occurs.

3 Design of autonomic computing system

At present, there are two kinds of ways to build cluster system. One is to connect the backup server to the main server, when the main server is failure, backup server will take over all the tasks. The other is to connect multiple servers together, all the servers work cooperatively to do the same task, therefore improve the response time of large -scale application. Also, each server takes on some fault tolerance task, when fault occurs in one server, system will separate that server from others and accomplish new load balance [6]. PC servers usually use two servers to build cluster system.

UNIX system often uses 8 servers to build cluster, and the OpenVMS of Compaq can support 96 servers cluster. The system we designed is aiming at the UNIX server's cluster, by adding a management server to the cluster system to manage the cluster, by adding agents to the servers of cluster to get the health status of servers and communicate with the management server, by using the Ignite-UX to realize the software deployment across network [7-8].

To realize the system self-management and autonomic features, we designed the following system architecture.

3.1 Architecture

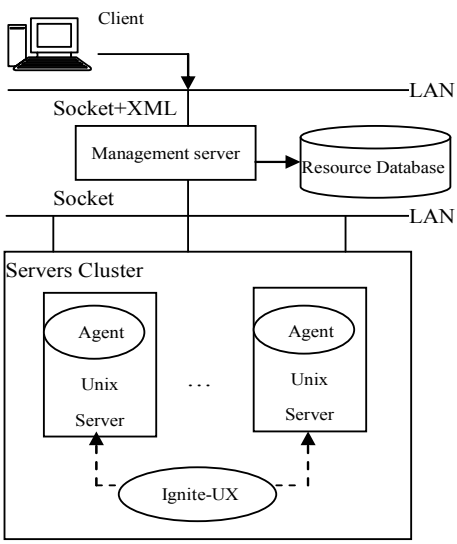

Figure1 Architecture of Autonomic System

The system is divided into three layers, client, management server and servers cluster. The three layers communicate with each other by LAN. Client communicates with management server by socket and XML; management server communicates with cluster system by socket. To communicate and operate the servers in the cluster, Ignite software must be installed on each UNIX server. Ignite-UX uses "pull" and "push" to deploy OS software across network, and can rapidly deploy system in the first time, or copy this configuration to other systems across network. This ability can save the time of the administrators therefore reduce the cost. By this way, rapid cluster system deployment can be achieved. Moreover, an agent is installed on each server to communicate with management server, deal the request and return the result. It is actually a background daemon process, by which management server can get the servers status and send request. It is the core of the autonomic computing system.

Client sends varied requests to management server, such as OS installation, software/patches installation, system backup, system recovery, system hardware status inquiry and so on.

Management server receives the requests and executes different operations according to request type. If the request is to query resource information, management server will get the information from resource database, and return the results with XML format. If the request is to install software on cluster servers, management server will firstly divide the servers of cluster into object servers and Ignite-UX server. For example, 8 nodes cluster can be divided into 7 object servers and one Ignite-UX server. Then management server will use the image file on the Ignite-UX server to deploy the object servers simultaneously. After the installation finished, results will be returned to client and resource database will be updated.

Resource database stores the OS type and version of each node of the cluster, the software and patches version installed in each node, current OS status and disk storage size, etc. The information in the database will be dynamically updated according to the node status.

3.2 Modules Constitution

In the autonomic computing system, the core is the management server. It is the brain of the whole system. It receives the client request, parses the XML data and sends commands to cluster servers, updates the resource database information according to the server status. The realization of this part contains the following modules.

1) Service Control

2) Communication Control

3) Resource Control

4) Events Control

5) Monitor Control

Moreover, agent is also the core module in the system. It is installed on UNIX server, which is to receive commands from management server, execute the commands and return the results.

The autonomic computing system modules constitution is as following:

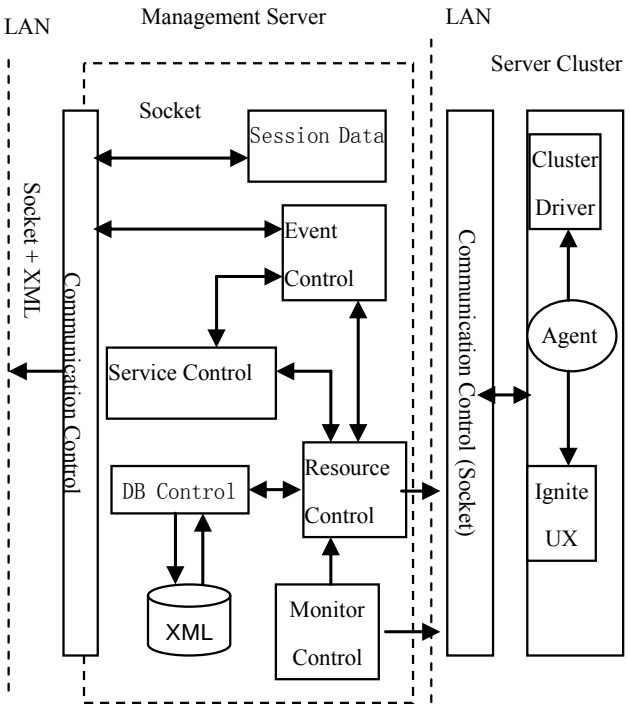

Figure 2 Modules Constitution

Service Control: Service control is the scheduling center, which is monitoring the events from events control module all the time. If new event is received, this module will extract XML data from the event, parse the information and judge the request type. If the request type is to query resource information, then service control will access XML database across resource control module, package the result information and send it to event control module, at last return the result to the client user. If the request type is to operate the servers of cluster, such as system backup or recovery, Unix script execution and etc., service control will call resource control module to parse the XML data to string type, send these strings in bytes to the agents installed in the servers of cluster by socket, then the agents will call the different shell scripts to execute the operations on the actual servers. The result information will be returned to resource control module by socket. After receiving the result from resource control, Service control will package the result information into events, and send events to event control module; finally result information will be returned to the client user across communication control module.

Communication Control: Communication control is to build connection between client and management server, receive the requests from clients, forward the XML data and return the results. To deal the requests of multiple clients, we build a Session Data for each client to save the status, port and IP information. As there are multiple clients in the system, many users may operate one server, for example, one user sends a command of "OS Start", and then the other user sends a command of "OS Stop", after executing the two commands, the server OS is stopped. If the first user wants to know the final status of that server, he should refresh the client GUI manually. To solve the problem, we use polling mechanism to get server status every five seconds, return the information to client, and then refresh the client GUI automatically.

Resource Control: Resource control is to manage the server hardware and status information, including adding data, modifying data, and deleting data from XML database. To assure the validation and integrity of the data, before one client wants to get the server status, the server resources will be locked. After one request is dealt, the server resource will be unlocked.

Events Control: Event control is to manage the events in the system. Each client has a session data in the system to store events information, and the client requests will be sent to management server as different events. All the events from different clients will be queued. Event control will get the event information from session queue and build the events queue. The events that are not dealt will be sent to service control

module, and the events dealt will be sent to communication control module. The result information will be again saved in session data, and then returned to client by communication control module. The events control procedure is as following:

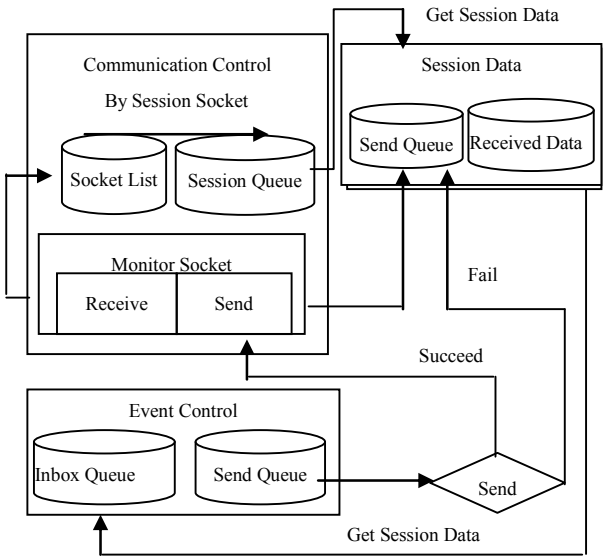

Figure3 Events Control Procedure

As figure3 shows, the Session Data stores the received data from client and the return result in Send Queue. Event control gets the session data into Inbox Queue, and sends the dealt events in Send Queue to Session Data. If succeed to send, the communication control will receive the events by monitoring socket and then send the dealt events to the Send Queue in Session Data. If failed to send, the events that are not dealt will be added into the Send Queue of Session Data directly. This means the socket error may occur. The Communication control builds a socket list and Session Queue to communicate with multiple users.

Monitor Control: Monitor control is to monitor the server's status in the cluster. It receives the report from the agents installed in the servers. If one server status has changed, the agent on it will send new status report to monitor control. The monitor control will update the server status information in XML database and send "Status Change" event to the event control module. At last the event will be return to the client and Client GUI will be updated too.

Agents: The agent on each server is a demon running in background. It is the crucial module, which makes the system own the ability of self-check, self-recovery. If we say the management server is the brain of the cluster system, then the agent is the nerve of each server. It perceives the health status of each server and reports it to management server. It receives the commands from management server and executes them, which is like that the brain tells the arms to stretch out or hold down. The agent is monitoring the request at one port, if new request comes, it will get the information, parse it into different command, and execute the command by cluster communication driver and Ignite-UX. The command type can be OS installation, software backup, patches update and so on. The final realization on each server is across shell scripts provided by UNIX core. Moreover, agent will check the hardware information every 30 seconds, then send health status report to monitor control module. If fault or error occurs in the server, the error information will be returned to management server at once.

3.3 Autonomic Computing Features

The servers in the cluster system accomplish one task cooperatively, providing a high performance environment for end users. If the cluster system owns the autonomic computing features that are, self-configuration, self-optimization, self-healing and self-protection, the system will work more effectively, and the system management will be much easier, therefore system management cost will be reduced [9-10].

The autonomic computing system we designed in this paper is for cluster system to realize self -management. The designed system has the following four features:

Self-configuration: Management server maintains two tables, one stores the server's status information, and the other stores the server's load information. The data in these two tables will be updated according to the real time information provided by monitor control module. When management server finds the workload of one server has exceeded the upper limit threshold value, it will find another server whose workload is normal to replace that server.

Self-healing: Health check is an important function in

cluster system, which provides the automatic error check mechanism. This function is done by monitor control module. After some time, monitor control module will send request to collect hardware information of servers, the agent in each server will receive the request and collect the status information of each server and send health report to monitor control module. If error is found on one server, management server will use the backup image file on Ignite server to recover that server.

Self-optimization: This function is done by monitor control module. It will send the commands of software upgrade and patches update to the cluster system after some time. The servers will download the new software/patches from Ignite server and update themselves, therefore optimize themselves.

Self-protection: In the autonomic computing system, we divide the user into two groups, common user and administrators. Common users cannot execute the operations such as "OS install", "System recovery", "User Management", and so on. Moreover, as to system installation and system recovery, password is requested before operation execution, which can reduce the risk and protect system from unsafe invasion.

4 Summary and future work

On the basis of analysis the problem in current cluster system, this paper designed and implemented an autonomic computing system for servers cluster based on multi-agent technique, which can make the cluster system work more effectively, make the complex system management become much easier, and make the system has the ability to manage themselves. The system designed has been put into practice, which has been proved to realize rapid software configuration, rapid error recovery and rapid application shift. But as the business needs vary quickly, the confirmed factors in system may vary accordingly. So now there is some incomplete design in system, which is the subject to study and improve in the future.

References

[1] ERAPHIN B. CALO and DINESH VERMA, "Toolkit for Policy Enablement in Autonomic Computing", ICAC-04, International Conference on Autonomic Computing. IEEE Computer Society, National Science Foundation, IBM Corporation, SUN Microsystems, April 2004

[2] JEFF O. KEPHART, DAVID M. CHESS, "The Vision of Autonomic Computing", Computer Journal, IEEE Computer Society, January 2003 issue

[3] Yoshihiro Tohma. Incorporating Fault Tolerance into an Autonomic-Computing Environment. IEEE Computer Society Vol. 5, No. 2; February 2004

[4] Zoran Constantinescu. Towards an Autonomic Distributed Computing System. Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA'03). IEEE Computer Society, 2003

[5] Rajkumar Buyya, "High Performance Cluster Computing", Architecture and System, Beijing, Electronic Polytechnic Publishing Company，2001, pp. 15-17

[6] Kephart JO, Walsh WE. An artificial intelligence perspective on autonomic computing policies. In: Verma D, Devarakonda M, Lupu E, Kohli M, eds. Proc. of the 5th IEEE Int'l Workshop on Policies for Distributed Systems and Networks. New York: IEEE Computer Society, 2004. 3-12

[7] CATHERINE H. CRAWFOND and ASIT DAN. EModel: Addressing the Need for a Flexible Modeling Framework in Autonomic Computing. Computer Journal, IEEE Computer Society, January 2002 issue

[8] Tianfield H. Multi-Agent autonomic architecture and its application in E-medicine. In: Liu JM, Faltings B, Zhong N, Lu RQ, Nishida T, eds. Proc. of the IEEE/WIC Int'l Conf. on Intelligent Agent Technolocy (IAT 2003). Los Alamitos: IEEE Computer Society, 2003. 601-604

[9] R. Sterritt, "Towards Autonomic Computing: Effective Event Management", Computer Journal, IEEE Computer Society, January 2003 issue

[10] CATHERINE H. CRAWFOND and ASIT DAN, "E-Model: Addressing the Need for a Flexible Modeling Framework in Autonomic Computing", Computer Journal, IEEE Computer Society, January 2002 issue