Sensors & Transducers, Vol. 174, Issue 7, July 2014, pp. 123-128

www.sensorsportal.com

Sensors & Transducers

http://www.sensorsportal.com

Implementation of Massive Real-time Database System Using Network Sensors and Sector Operation

Q. D. Sun, Y. P. Qiu, Y. F. Shao and W. Y. Yan
School of Electronic and Electrical Engineering, Shanghai Second Polytechnic University,
2360 Jinhai Road, Shanghai, 201209, China
E-mail: qdsun@sspu.edu.cn, ypqiu@sspu.edu.cn, yfshao@sspu.edu.cn, wyyan@ sspu.edu.cn

Received: 14 March 2014 /Accepted: 30 June 2014 /Published: 31 July 2014

Abstract: In order to improve the stability of the massive database system and its capabilities of data processing and services, this paper proposed a distributed information server cluster. The N-level distributed architecture based on the network sensors was used to form a massive database system for real-time collection and query. Due to support by the network sensors, all components in this architecture have the functionality of plug and play. To efficiently schedule the tasks of storing gathered data and querying information, a dynamic and self-adaptive scheduling algorithm based on task sensors was introduced in the application server. The task sensor collects the load status from the common processes in various information servers by information collection processes in the application server and sends them to the scheduler in the same server, which dispatches the tasks of data storing into the most appropriate information server. Furthermore, an appropriative database system based on sector read-write directly was presented to improve the access speed, which is almost 25 times than that of database based on SYBASE or ORACLE. The practice shows that the system developed by this strategy has good flexibility and efficiency. *Copyright © 2014 IFSA Publishing, S. L.*

Keywords: Information server cluster, N-level distributed architecture, Network sensor, Task sensor, Task scheduling, Massive real-time database, Sector segmentation.

1. Introduction

As a massive real-time data collection system, its real-time capability should be considered chiefly, that is to say it can not lose the gathered data even when it is in any abnormal situations, for example need to replace storage medium or backup data. It is almost impossible when applying the single information server in the system. If this system is also required to provide the query service, the database system with the form of single information server will be endured more load pressure. Additionally, the mass amount of data should be taken into account. It is very difficult to satisfy the demands of the writing of gathered data and real-time query of information if using general design method for such a system.

Therefore, to solve the problems mentioned above, this paper presents the concept of information server cluster, and applies the advanced N-level distributed architecture based on network sensors to provide the distributed writing method for data collection and parallel operation mode for information query [1-4]. In order to assuring the real-time capability of system and agility of maintenance, a dynamic and self-adaptive algorithm based on task sensor has been introduced in this system to schedule and optimize the tasks among various information servers [5].

This paper is organized as follows. In section II, we give the N-level architecture of distributed computing application with network sensors. Section III introduces the concept of distributed information server cluster and presents its architecture. Section IV advances the organized tactic of appropriative quick

database system based on hard disk sector read-write. Section V applies those theories proposed in foregoing sections to a massive real-time database system. Section VI gives the conclusion of this paper.

2. Architecture of Distributed Computing Based on Soft Sensors

2.1. Architecture

The leading architecture of distributed computing application is N-level one [6], which divides the application into three independent logical parts: Display component, Logic component and Data component. The reason why it is called N-level is that the number of application servers (logic component) is unlimited. Here we can also construct its architecture as shown in Fig. 1.

In Fig. 1, the whole application system is composed of Display component, Logic component, Data component and Network. If the Data component uses the ODBC supporting database, such as Oracle, Sybase, SQL Server, DB2 or Informix etc, then it will make programming easy. The key task in the system is the harmony and cooperation of Display component (i.e. the User terminal in Client) and Logic component (i.e. the Application service in the Server).

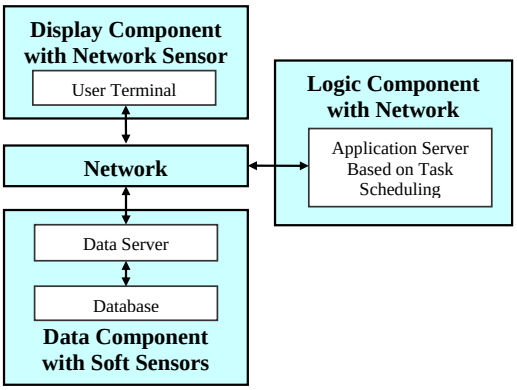

Fig. 1. N-level architecture of distributed computing.

2.2. Functions of Soft Sensors

Soft Sensor is a special procedure based on communication protocol such as WinSock in windows, to detect the status of the component in the network. If the detection object is the connection status, then it is called the Network Sensor. If the detection object is the busy status of data component, then it is called the Task Sensor.

2.2.1. Network Sensor

Not only did the connections between various components build by the general network, but also by the network sensors based on WinSock in our system. Thus, once a component disconnect from or connect into the system, the network sensor in application server will detect them immediately.

2.2.2. Task Sensor

In a massive real-time database system, there are many equally data components to process the information storing and querying usually. The tasks of these data components need a dispatcher in the application server to scheduling their tasks. The scheduling principle of the dispatcher is busy one doing less and idle one doing more. The status of busy or idle of data component needs a task sensor to collect it and feedback to the dispatcher in the application server.

2.3. Thin-client Realization

Due to applying the distributed computing, the Client/Server needs no longer too big, too fast CPUs and too large capacity memory. User can optimize the application program to reduce the complexity of tasks to an acceptable degree by using of distributed object if need be. So, the Client can be real thin-client.

3. Implementation of Information Server Cluster

3.1. Information Server Cluster

The information server cluster (ISC) is such a system that several information servers work with parallel mode in the system, and each of them plays equal role to tidy and store the data from collectors and provide various services to query terminals, and also every one can be moved away or hung in the system at any moment, that is plug and play (PnP).

Although there are several information servers in the cluster, but as far as data collectors and query terminals are concerned, they should be an entirety or a whole component.

3.2. Data Structure of Information Server State

The work state of information server in the cluster is determined by the state data structure stored in the information server. The state data structure is defined as follows:

```
Struct InfoSvrState
{
    Int nNdx;
    Int nState;
    Bool bWritable;
    Bool bIsBackSvr;
},
```

where nNdx is the serial number of information server and it is exclusive in the cluster. nState is the logical online state of information server. When nState=0, the information server is offline and cannot be operated for reading or writing. If nState=1, the information server is online and accepted reading and writing. bWritable is the state of readable or writable of information server. When bWritable=TRUE, the information server is read-writable. If bWritble=FALSE, the information server is of read-only. bIsBackSvr is the mark of whether the information server is a backup server or not. When bIsBackSvr =TRUE, the information server is one of backup server and is of read-only, and bWritable can not be TRUE. If bIsBackSvr =FALSE, the information server allows reading and writing, but the state of read-write depends on the value of bWritable.

Each information server holds itself state information. When an information server was connected into the cluster and the service program started up, its state information firstly sent to the application server to inform that it has been in physical online. But it is logical online or offline, we need to check the value of nState, which allows be modified by information service program.

3.3. Distributed Computing and Network Sensor

To want information server cluster operating as normal demands, it is far from enough that just having itself, and it should be supported by other hardware and software systems.

3.3.1. Distributed Computing

The implementation of parallel processing of information server cluster depends on the distributed computing. Its architecture is shown as Fig. 2. There is an application server in the system, which can combine several independent information servers as a whole component and schedule their parallel works. Both the pending data from collectors and the query instructions from user terminals are dispatched to various servers in the cluster by all task sensors and the task scheduler based on the dynamic and self-adaptive algorithm in the application server.

3.3.2. PnP Based on Network Sensor

In the basis of general network, the connection between information server cluster and application server is also built by network sensor based on communication protocol such as WinSock in windows. Thus, once the one or several servers in cluster disconnect from or connect into the system, the network sensor in application server will detect them immediately and assure dispatching stored data equally to each server and sending query instructions to all connected servers, i.e. realizing PnP.

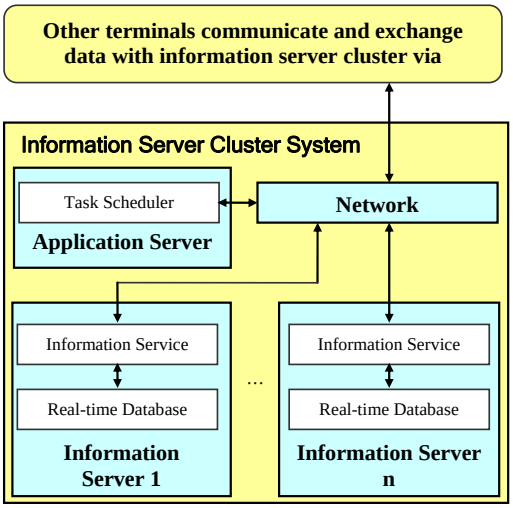

Fig. 2. Architecture of information server cluster based on network sensors.

3.4. Task Scheduling and Task Sensor

The task scheduling is one of top challenges in parallel distributed computing. Its general form and several limited forms are all NP-complete problems. Accompanying with widely using of the large-scale parallel distributed processing system especially the network workstation cluster system, it becomes a key problem to improve the utilizing rate of resources how the efficient scheduling tactic can be designed to balance the loads in various node computers [5, 7-10].

This paper uses a dynamic centralized self-adaptive scheduling tactic (shown as in Fig. 3) [5], which collects the load status from the common processes in various information servers by task sensor (TS) in the application server and sends them to a scheduling process in the same server to dispatch the tasks of data storing into the most appropriate information server.

The TS does not participate in decision-making of task scheduling but is responsible for gathering load status and transferring the instructions from query terminals. When the number of node computers located in parallel platform are great than an enactment threshold, the network sensor in the application server will automatically create a TS, which is responsible for gathering the load status of its own node computers and sending them to the scheduling process wholly to decrease the access times to application server by every node computer and lighten the system spending. When all of its node computers are removed, the TS will be destroyed too.

3.5. Pros and Cons of ISC

There are quite a lot of advantages in the ISC, which can provide information servers much flexibility for their maintenance, data backup and query of backup data.

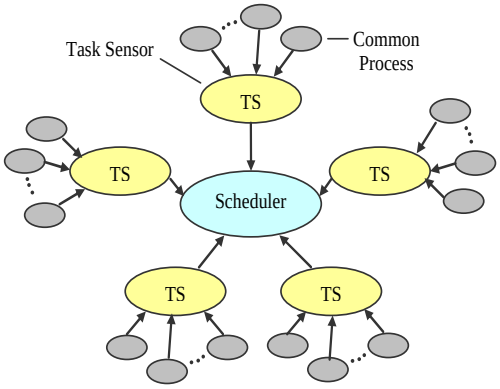

Fig. 3. Architecture of task scheduler based on task sensors.

3.5.1. Easy Maintenance

Since the ISC supports PnP, if there is an information server out of order and needed to repair, it can be taken off the system immediately and also can be put in this system when repaired.

3.5.2. Easy Backup

It is unnecessary to prepare backup hardware and software sedulously in such an ISC. There are two reasons supporting the point of view. First, the current process of data writing is, namely, backup. When the store medium (hard disk or re-write DVD) is full, you can take away from the system and change a new one put in. Thus, the data in the full medium has become the backup data. Second, the data in the real-time database usually has certain time validity, the data outdated can be circularly covered with new data. As enough as the capacity of store medium in the existent ISC, therefore, the data covering method can be applied to data writing process. Here, the backup process will be no longer needed.

3.5.3. Easy Query for Backup Data

The PnP function of ISC based on network sensor can assure the backup information server linking to the system optionally just like a general information server. This method can easily solve the query problem of backup data. But is must be considered that you may make out a distinctive mark for backup information server distinguishing from general information server to avoid the system covering its data.

3.5.4. Real-time Assuring p

The parallel processing for multi-servers can assure system's real-time capability of massive data collection and information query. In principle, if the data capacity increases rapidly, the speed of data writing and inquiring can still be ensured only through adding some information servers, which will be detected by the network sensor automatically.

3.5.5. Some Disadvantages

Although ISC has many advantages, its disadvantages are also obvious. On one hand, the cost of hardware of system may be raised due to increasing the number of servers. However, using the technology of distributed computing in the system can lower the requirement of single server's capability. From this point of view, the part of raising cost of hardware can be compensated. On the other hand, the capacity of communication in the system will be increased because of distributed computing. But this issue can be solved by improving the network speed.

4. Design Scheme of Appropriative Quick Database System

The key component in information server is the database system, which provides the data writing and information querying. Generally, the common database system can be simply operated and easily managed but has a slower speed. Meanwhile, the notable characteristics of massive real-time database system are having large amount of data to be treated and higher degree of real-time, and thus, the common database can not satisfy those demands. In this case, it looks like especially important if there is an appropriative quick database system (AQDS) organized by an exceptive method to fit those demands.

4.1. Specializing Database System

The universality of database system is a critical factor impacting the speed of database. To be fit for certain purpose, an appropriative database does not consider many additional cases. Therefore, it will be simple and easy to operate and have strong rule. Thereby, the access speed of database system will be improved.

4.2. Sector Operating Instead of General Data Read-Write

Generally, the common database system is just a file management system. Normally, the speed of file read-write is much slower than that of directly operating to sectors of hard disk. Therefore, it is determinant of improving the speed of database system that using sector read-write orderly instead of file management.

4.2.1. Hard Disk Layout of Information Server

Usually, the hard disk of information server can be segmented into several logical disks as shown in Table 1. The disk C is used to store some service programs and common system information, while

other disks are reserved to store the data directly with sector read-write.

Table 1. Hard disk layout of information server.

Logical disk	Stored data
C	System data files such as information service programs, state information file and so on.
⩾D	Sector read-writing information and index data.

4.2.2. Sector Segmentation of Hard Disk

In order to store data conveniently and set up index and improve the speed of data writing and information querying, we treat the logical disks below D (include D) as shown in Fig. 4. A logical disk can be segmented into some blocks. A block is composed of some sectors. A sector also can be segmented into some units, which is a minimum cell for storing data. To improve the access speed, the hard disk operation is based on block read-write, and thus it will decrease the read-write times of hard disk to extend its using life.

In Fig. 4, we set: 1 Unit = 32 Bytes, 1 Sector = = 16 Units = 512 Bytes, and 1 Block = 4096 Sectors = = 65536 Units = 2 Mbytes. The most significant bit (MSB) of the first byte of a Unit is used to stand for the validity of this Unit. If the MSB is 0, then the Unit is invalid and can not be used to store the data, otherwise the Unit is valid to be stored data.

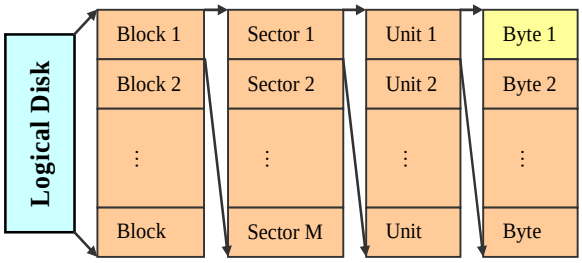

Fig. 4. Sector segmentation of hard disk.

4.2.3. Creating Subsection Index

The purpose of index for database is to improve the query speed. It is quite difficult to build the index for whole data in the massive database due to its very bulky data, and building index has a big impact on the writing speed to store gathered data. Therefore, we do consider the subsection index, which is a method of classification and indexing. In this method, the data are classified firstly according to some characteristics of data, and then the data in the same class are indexed.

For example, the indexed records are the information about telephone number. Firstly, we do consider the regional code of telephone number as the classification number. Then those records in the same class are indexed. When in query, we should search the class first then the detail telephone number. In this way, the speed of building index will be improved greatly, and also the query speed will be enough fast.

4.2.4. Building Distributed Computing for AQDS

As mentioned above, the distributed computing mode is used for multi information servers. To augment the capacity of system is just to increase the number of information server.

5. Implementation of Massive Real-time Database System

5.1. Architecture of Massive Real-time Database System

A massive real-time database system can be described by the architecture shown as in Fig. 5.

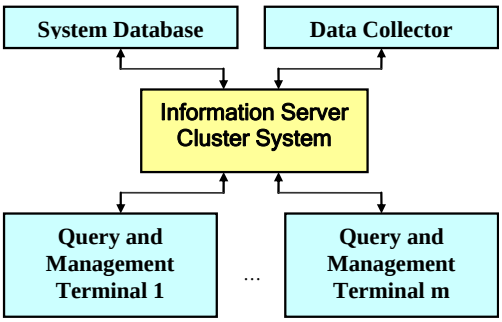

Fig. 5. Architecture of massive real-time database system.

The key module of whole system is the information server cluster (ISC) system, and the other modules in system such as a (or some) data collector(s), some query and management terminals and a system database are connected into the Network layer within the ISC system. The data collector is responsible for real-time data gathering, the query and management terminal is used to manage the system and query or search the information, and the duty of system database, which can be implemented by general database system such as SYBASE or ORACLE, is depositing those settled data system required. The ISC system, which is an AQDS with distributed computing style, is in charge of storing the data collected by data collectors. Within the ISC system, the application server based on virtual sensors and task scheduler is the centre for harmonizing and organizing the whole system's communication and distributed computing. The data gathered by collectors being sent to which information server and the information required by the query terminal getting from which information server is organized by the application server. Namely, the application server

enables that the information servers are transparent to the data collectors and the query terminals.

5.2. An Application and Performance Comparison with Common Database

This organized tactic had been used to develop a real-time information system in Shanghai Public Security Bureau and obtaining a great achievement. The application system had many advantages, such as high access speed, strong real-time capability, agility configuration and easy maintenance. In this system, the number of bytes of unit L was set as 32. In principle, a record can be stored into a unit, but a long record can be permitted to store into several continuous units. The information capacity of half year in this system was billions units. In such a mount of data, querying the data within a half month period was only needed about 4 seconds, but the time of querying same data from a database based on SYBASE or ORACLE was round 100 seconds. Comparing with general database, it is obvious that the information querying speed from the AQDS based on this tactic has been improved quite a lot.

In addition, to enable the system having a more flexible configuration when being actualized, the service procedure regarding to the data collecting can be separated from the application server and installed into the data collector, while the data collector copies a system data. Thus, even only one data collector and one information server remain in the system, the data collecting system can still run to farthest ensure no real-time data being lost.

6. Conclusions

This paper proposed a distributed information server cluster to improve the stability of the massive database system and its capabilities of data processing and services. It also used the network sensors based N-level distributed architecture, in which all components have the functionality of plug and play, to form a massive database system for real-time collection and query. In the application server, a dynamic and self-adaptive scheduling algorithm based on task sensors, which collect the load status from the common processes in various information servers, was introduced to efficiently schedule the tasks of storing gathered data and querying information. That is to say, the scheduler will dispatch the tasks of data storing into the most appropriate information server according to its load state. Furthermore, an appropriative database system applied the technique of sector read-write directly was presented to improve the access speed, which is almost 25 times than that of database based on SYBASE or ORACLE. The application shows that the system developed by this design method has good flexibility and efficiency.

Acknowledgements

This research project was supported by the Leading Academic Discipline Project of Communication and Information System of Shanghai Second Polytechnic University Grant (No. XXKZD1302).

References

[1]. E. Stoyanov, A. MacWilliams, D. Roller, Architecture for Distributed Component Management in Heterogeneous Software Environments, T. Sobh (ed.), *Springer*, 2007.

[2]. S. Kirn, O. Herzog, P. Lockemann, O. Spaniol, International Handbooks on Information Systems - Multiagent Engineering (Architectural Design), *Springer*, 2006.

[3]. P. McBrien, Design of distributed applications based on the OSI model, *Lecture Notes in Computer Science*, Vol. 1250, 1997, pp. 361-373.

[4]. P. K. Rajesh, B. Magesh Babu, Web-based collaborative conceptual design: an XML approach, *International Journal of Advanced Manufacturing Technology*, Vol. 38, No. 5-6, 2008, pp. 433-440.

[5]. X. R. Wu, A Dynamic and Self-adaptive Task Scheduling Mechanism, *Computer and Modernization*, No. 148, 2007, pp. 65-67.

[6]. S. Gallagher, S. Herbert, PowerBuilder 6.0 Unleased, *Sams Publishing*, 1996.

[7]. X. H. Kong, W. B. Xu, J. Sun, Research on Distributed Multiprocessor Scheduling Based on the Ant Colony Algorithm, *Computer Engineering & Science*, Vol. 29, No. 3, 2007, pp. 63-65, 83.

[8]. A. Z. Liu, J. Z. Wang, H. L. Jia, S. Z. Wang, L. Y. Chen, Genetic algorithm for mobile Agent tasks scheduling, *Computer Applications*, Vol. 27, No. 11, 2007, pp. 2830-2833.

[9]. Y. Zhang, X. H. Zhang, Research of multiprocessors scheduling policy based on improved ant colony algorithm, *Computer Engineering and Applications*, Vol. 43, No. 35, 2007, pp. 74-76.

[10]. X. N. Tong, W. N. Shu, Z. M. Li. Load balancing and task scheduling of heterogeneous multiprocessor system, *Optics and Precision Engineering*, Vol. 15, No. 12, 2007, pp. 1969-1973.