# Research on the design of a scalable hardware platform for data management

**ZHANG Fei[1], LIU Guang-Ming[2]**

[1]College of Computer Science, National University of Defense Technology
[2]National Supercomputer Center in Tianjin
nudtfeizhang1029@163.com, liugm@nscc-tj.gov.cn

**Abstract**

Data management system includes hardware platform and management software. Under the circumstance of explosive growth of data, scalability and processing speed are two important indicators of measurement of hardware platform. Design a highly scalable data management hardware platform, and solve the issues of processing speed and scalability together. Propose two running models of this platform: the Direct Storage Access Model and the Storage Space Mapping Model. The Direct Storage Access Model is more suitable for the computing-intensive data management, while the Storage Space Mapping Model for the I/O-intensive data management.

**Key words:** data management system, scalable hardware platform, the Direct Storage Access Model, the Storage Space Mapping model

## 1. Introduction

Since it went into the information times, especially the appearance of the Internet, "explosion" has become a specific term to describe the growth of data. Under the combined effect of social networking sites, Internet-enabled mobile phones and expanding video surveillance system of governments, etc., the growth speed of global data still keeps accelerating. According to statistics, the total amount of global data is nearly 500 EB. With the growth of data, data management becomes a common issue in all walks of life. It has experienced three stages of development: labor management, file system and database system. Since the emergence of the database technique in 1960s, a variety of hardware platforms and software of data management came into being, such as ORACLE[1], GFS[2], Bigtable[3] and MapReduce[4] in Google, Hadoop[5] in Apache, Cassandra[6] and Hive[7] in Facebook, PNUTS[8] in Yahoo!, etc.. These systems have their own merits and drawbacks[9]. There is no data management system is suitable for all application fields. Nowadays, data management has the following features[10]:

- Large data volume. It is marching towards the EB level from the PB level.
- Big analytic depth. It is changing from general analysis to deep analysis. According to the report of TDWI for data analysis[11], at present, data analysis is not only been restricted in the analysis and survey of the existing data, but is hoped to achieve more analysis and forecasts for the future through analyzing the existing data.
- Generalization of hardware platform. It is the growth of data that leads to the augment of the scale of data management system and the increase of hardware cost. In order to lower the hardware cost, the hardware platform of data manage-

 

ment is converting from high-end servers to large-scale clusters which are composed of general hardware.

The new features of data management put forward new demands for hardware platform. A good hardware platform should have the features of high scalability, fast processing speed and wide applicability. For the purpose of meeting the new demands for hardware platform, a scalable data management hardware platform is designed in this paper. Platform takes the share-nothing structure, and its greatest feature is high scalability. Either processing speed or storage capacity can expand with the growth of data amount or the growth of analysis depth, which results in high adaptability of platform. This platform is very flexible, so it can lower the cost of updating system. Under the architecture of this platform, two running models of this platform are proposed: the Direct Storage Access Model and the Storage Space Mapping Model. They have different characteristics and hardware requirements, can meet different demands of users and are suitable for data management of different kind of data managements.

The structure of this paper is organized as follows: Section one describes the background, purpose and significance of designing this hardware platform. The architecture of this platform is described in Section two. Section three proposes two running models of this hardware platform.

## 2. The architecture

The architecture of hardware platform is the foundation of data management system, and determines its final performance, scalability and availability, etc.. This system consists of Load Balancing Nodes, Management Nodes, Compute Nodes, Index Maintenance Nodes and Storage Network. They are connected by the High-speed Internetwork, and communicate and transmit data with it, as illustrated in Fig. 1. Some core components of this system will be described in graphic detail below.

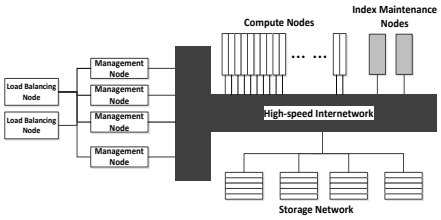


Fig.1 The architecture of platform.

### 2.1 Load Balancing Node

Load Balancing Node evenly distributes tasks to different Management Nodes. The two Load Balancing Nodes are same in case one becomes invalid.

### 2.2 Management Node

Management Nodes take charge of user login, receive users' requests and return results to users. They are also the cores of the system, and supervise and manage system's operation. They distribute tasks to Compute Nodes or Index Maintenance Nodes, and summarize the results returned from them. Aiming to supervise the running conditions of the whole system, Management Nodes must maintain a Node Information Table, as showed in TABLE Ⅰ. Management Nodes uniformly number Compute Nodes and Index Maintenance Nodes, and then find nodes and assign tasks through Node ID. All nodes have three running states: busy, idle and backup. If a node becomes invalid, Management Nodes will open a new backup node to take over its tasks, or assign the tasks in this node to other running nodes.

TABLE I. The Node Infromation Table

| Node Type | Node ID | Running State | Task ID |
|---|---|---|---|
| Compute Nodes | 0 | busy | 52, 45 |
| | 1 | idle | |
| | 2 | busy | 687, 36, 45 |
| | … | … | … |
| | N | backup | |
| Index Maintenance Nodes | N+1 | busy | 34, 15 |
| | N+2 | idle | |
| | … | … | … |
| | N+n | backup | |

Besides supervising the states of nodes, each Management Node also needs a Task Information Table to supervise the running conditions of the tasks managed by it, as showed in TABLE Ⅱ. When Management Nodes receive task from Load Balancing Node, they will distribute it to Compute Nodes and Index Maintenance Nodes by taking some task allocation strategies according to the Node Information Table and the Task Information Table, and create new item in the Task Information Table to record the allocation conditions of tasks. When a Compute Node or an Index Maintenance Node finishes one task, it will submit results to the Management Node which gives this task to it. Management Node will store the results. When all nodes running this task finished, Management Node summarizes them, and returns the final result to user. The Task Information Tables in different Management Nodes are different, while the Node Information Tables are same because it is necessary to distribute tasks. Every time when Management Nodes distribute tasks, all of the rest Management Nodes also receive this instruction, and modify the copies of the Node Information Table according to it. Finally the node distributing task updates the Tables in storage. The Node Information Table and the Task Information Tables are stored in specific position of storage, and Management Nodes just keep their copies.

TABLE II. The Task Information Table

| Task ID | Node ID | Finish or Not | Results |
|---|---|---|---|
| 35 | 2 | No | |
| | 4 | Yes | “ZHANGSAN” |
| 34 | N+2 | No | |
| | N+3 | Yes | “090999FF0F” |
| … | … | … | … |

### 2.3 Index Maintenance Node

For quickening the speeds of querying and processing data, Special nodes are designed to create and maintain indices in this platform. Index directories are stored in specific position of storage, and every Index Maintenance Node keeps their copies. When Management Nodes received user’s requests, they send keywords to the Index Maintenance Nodes to query indices directories. The result of the Index Maintenance Node is the physical addresses in storage of the operands. For example, if the operation is INSERT, the result is the physical address where the data will be inserted in storage.

### 2.4 Compute Node

Compute Nodes are the executive component of data management system to process data, such as INSERT, DELETE, etc.. Each Compute Node also needs a Task Information Table to record the conditions of tasks running on it, as showed in TABLE Ⅲ.

TABLE III. The Task Information Table in Compute Node

| Task ID | Running Condition | Result |
|---|---|---|
| 45 | running | |
| 46 | finished | “BeiJing” |
| … | … | … |

After having received instruction, data and the addresses from Management Nodes, Compute Nodes operate data according to the instruction. There is no data access and message transmission among Compute Nodes.

### 2.5 Storage Network

Storage Network is composed of disk arrays. All users' data and some system's data (e.g. the Task Information Tables, the Node Information Table) are stored in it. Different strategies of fault tolerance and disaster tolerance can be made to meet different demands of users according to different security level.

## 3. The running model

The running model is about how to make platform use hardware resources more fully and work more effectively. According to the features of the architecture, this platform can take two running models: the Direct Storage Access Model and the Storage Space Mapping Model. Two models have their own merits and drawbacks, and are suitable for different applications. Of course, the hardware requirements are different.

### 3.1 The Direct Storage Access Model

The Direct Storage Access Model is similar to the running way of traditional data management hardware platform. After receiving a task, Management Nodes firstly distribute it to the Index Maintenance Nodes to get the storage addresses of operands. Then Management Nodes distribute this task and the addresses of operands to Compute Nodes through some distributing strategies. Compute Nodes process data by reading data from the addresses, and then submit the results to Management Nodes. After all Compute Nodes executing this task finished, Management Nodes summarize the results and return the final result to user, as showed in Fig.2.

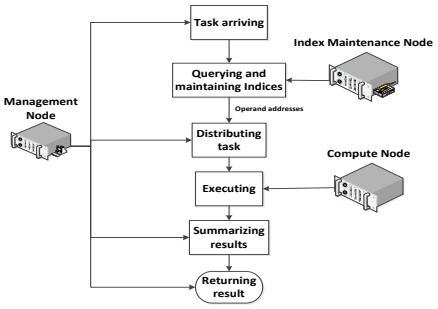


Fig.2 The procedure of the Direct Storage Access Model.

(1) Task arriving. Load Balancing Nodes distribute tasks (e.g. INSERT, QUERY or DELETE, etc.) to Management Nodes. Management Nodes create new item in the Task Information Table for the task received

(2) Querying or maintaining Indices. Management Nodes send instruction and keywords to Index Maintenance Nodes. Index Maintenance Nodes execute tasks according to instruction, return the physical address of keywords in storage, and maintain index directories.

(3) Distributing task. According to the addresses returned by Index Maintenance Nodes and the Node Information Table, Management Node distributes task to Compute Nodes, and records the condition of task distribution in the Task Information Table.

(4) Executing. Compute Nodes create new item for this task in their own Task Information Tables and process data according to the instruction. For example, if the instruction is QUERY, Compute Node matches between the keywords and the data read from the addresses received; if INSERT or DELETE, Compute Nodes insert or delete data in the addresses. Compute Node just stores few recent data reading from storage in memory, and needs to access storage to read data, as showed in Fig. 3. When Compute Node

finishes a task, it submits the results to Management Node distributing this task.

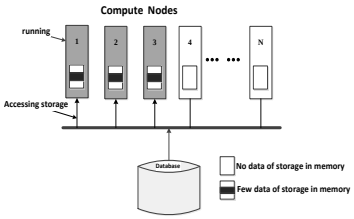


Fig.3 The execution of tasks in Computing Nodes in the Direct Storage Access Model

(5) Summarizing results. Management Node summarizes the results when all Compute Nodes running one task finished, and return the final result to user.

The Direct Storage Access Model is featured by simplicity and low requirements to hardware resources. The allocation strategy of task plays a vital role in the factors affecting the performance of system. Corresponding optimal strategies can be made according to different applications and the performance of hardware platform, such as the loop allocation strategy and the least-priority allocation strategy.

### 3.2 The Storage Space Mapping Model

Compute Node needs large memory in the Storage Space Mapping Model and the storage space is mapped to memories of Compute Nodes (except backup Compute Nodes). When system starts up, it will read all data in storage to corresponding memory. After achieving the addresses of operands, Management Node distributes task to the Compute Nodes which contain theses addresses. Fig. 4 shows the procedure of this model. Compute Nodes do not need to read data from storage when they are processing data, and directly get data from their own memories, which makes full use of the compute resources and improves the efficiency of system.

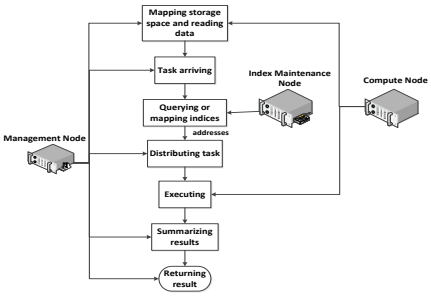


Fig.4 The procedure of the Storage Space Mapping Model

Unlike the Direct Storage Access Model, Compute Nodes need to read data from storage in the Storage Space Mapping Model when system starts up. It is not restricted to read data, but divide storage space into pieces with same size and map them to the memories of all open Compute Nodes. The memory of every Compute Node points to a continuous segment of storage addresses and stores the data of those addresses. Even through there is no data in some of those storage addresses, the memory of Compute Node also needs to reserve places for them, as showed in Fig. 5. Therefore, the minimal number of Compute Node is the result of the storage space divided by the size of memory used to store storage data by Compute Node.

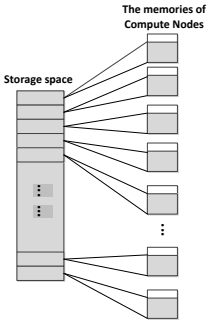


Fig.5 The logic diagram of storage space mapping

Management Nodes need to record the result of mapping storage space for distributing tasks. We can add infor-

mation items (start address and end address of storage space) in the Node Information Table to carry it out, as illustrated in TABLE Ⅳ. At the same time, Compute Node also creates table to record the mapping result. When a Compute Node becomes invalid, Management Node opens a backup Compute Node to take over its tasks, and map its storage space segment to the memory of the new node. Then the new Compute Node read data from this address segment.

TABLE IV. The Node Information Table of the Storage Space Mapping Model

| Node Type | Node ID | Running State | Task ID | Start storage address | End storage address |
|---|---|---|---|---|---|
| Compute Nodes | 0 | busy | 52, 45 | 0000000000 | 000000ffff |
| | 1 | idle | | 0000010000 | 000001ffff |
| | 2 | backup | | | |
| | … | … | … | … | … |
| Index Maintenance Nodes | 0 | busy | 34, 15 | — | — |
| | 1 | idle | | — | — |
| | … | … | … | — | — |

The procedure of the Storage Space Mapping Model is similar to that of the Direct Storage Access Model. They are just different in the phrases of distributing and executing tasks. In the phrase of distributing task, Management Node achieves the IDs of Compute Nodes (i.e. the nodes whose memories contain the addresses of operands) through querying the Node Information Table according to the addresses returned by Index Maintenance Nodes, distributes the task to these Compute Nodes and records the condition of distribution. When Compute Nodes are executing tasks, they do not access data from storage, and directly read data from their own memories. When the data in memory are modified, this Compute Node updates the corresponding data in storage. There is no data access and message transmission among Compute Nodes, as showed in Fig. 6.

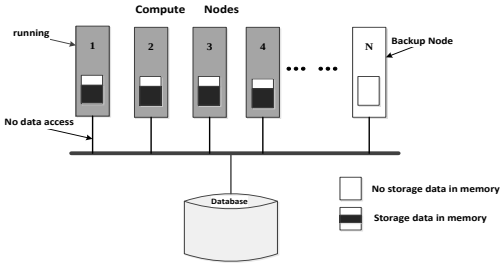


Fig.6 The execution of tasks in the Storage Space Mapping Model

The Storage Space Mapping Model is more complicated and more efficient than the Direct Storage Access Model. However, the Compute Node-invalid overhead is much larger than that of the Direct Storage Access Model. When a Compute Node becomes invalid, Management Node just needs to open a backup Compute Node to take over its tasks in the Direct Storage Access Model, while besides this, the new Compute Node needs to read data of the mapping address segment from storage in the Storage Space Mapping Model. The Storage Space Mapping Model has a higher performance requirement of hardware resources, especially the memory size of Compute Nodes.

## 4. Conclusion

Data management system is divided into hardware platform and management software. Nowadays data are explosively growing. Scalability and the speed of processing data have become two important criterions to evaluate hardware platform. A highly scalable hardware platform of data management is designed in this paper. All nodes in this system are independent and symmetric, so system has high scalability. Scalability determines adaptability to some degree. System can adjust computing ability and storage capacity to different application environments, it, therefore, is characterized by broad adaptability. Management

Nodes, Compute Nodes, Index Maintenance Nodes and disk arrays are easy to expand or shrink, and the original architecture of system will not be changed. Special nodes are used for Maintaining and querying indices which are time-consuming, which remarkably improves the parallelism of system and lower the data-processing time.

Two running models (the Direct Storage Access Model and the Storage Space Mapping Model) are proposed on the basis of the architecture of this platform. These two models have different hardware requirements. The performance and feature of platform under different models also are different. The Direct Storage Access Model needs to access data from storage when executing tasks, so it is more suitable for the computing-intensive applications, while the Storage Space Mapping Model does not need to access data from storage, it, therefore, is more suitable for the I/O-intensive applications.

In future work, we will build a prototype of this platform, and test the performance of two running models in this prototype. At the same time, we will further to find the application fields which are suitable for this platform, and adjust and optimize platform according to different application circumstances.

## 5. References


[1] ORACLE, http://www.oracle.com/index.html.

[2] Sanjay Ghemawat , Howard Gobioff , Shun-Tak Leung, The Google file system, Proceedings of the nineteenth ACM symposium on Operating systems principles, October 19-22, 2003, Bolton Landing, NY, USA.

[3] Fay Chang , Jeffrey Dean , Sanjay Ghemawat , Wilson C. Hsieh , Deborah A. Wallach , Mike Burrows , Tushar Chandra , Andrew Fikes , Robert E. Gruber, Bigtable: A Distributed Storage System for Structured Data, ACM Transactions on Computer Systems (TOCS), v.26 n.2, p.1-26, June 2008.

[4] Jeffrey Dean , Sanjay Ghemawat, MapReduce: a flexible data processing tool, Communications of the ACM, v.53 n.1, January 2010.

[5] Hadoop，http://hadoop.apache.org/，2012.11.18.

[6] Cassandra. http://incubator.apache.org/cassandra/.

[7] A.Thusoo, J. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff and R. Murthy, “Hive – A Warehousing Solution Over a MapReduce Framework,” VLDB, Lyon, France, August 2009, pp. 1626-1629.

[8] B.Cooper, R. Ramakrishnan, U.Srivastava, A. Silberstein, P. Bohannon, H. Jacobsen, N. Puz, D. Weaver and R. Yerneni, “PNUTS: Yahoo!’s Hosted Data Serving Platform,” VLDB, Auckland, New Zealand, August 2008, pp. 1277-1288.

[9] Yingjie Shi , Xiaofeng Meng , Jing Zhao , Xiangmei Hu , Bingbing Liu , Haiping Wang, Benchmarking cloud-based data management systems, Proceedings of the second international workshop on Cloud data management, October 30-30, 2010, Toronto, ON, Canada.

[10] WANG Shan, WANG Hui-Ju, QIN Xiong-Pai, ZHOU Xuan, “Architecting Big Data: Challenges, Studies and Forecasts,” CHINESE JOURNAL OF COMPUTERS, Vol. 34, No.10, pp. 1741-1752, Oct. 2011.

[11] TDWI Checklist Report: Big Data Analytics, http://tdwi.org/research/2010 /08/Big-Data-Analytics.aspx.