The Improvement of Auto-Scaling Mechanism for Distributed Database - a case study for MongoDB

Chao-Wen Huang
Telecommunication Laboratories Chunghwa Telecom Co., Ltd.
Taiwan, R.O.C
huangcw@cht.com.tw

Chia-Chun Shih
Telecommunication Laboratories Chunghwa Telecom Co., Ltd.
Taiwan, R.O.C
ccshih@cht.com.tw

Wan-Hsun Hu
Telecommunication Laboratories Chunghwa Telecom Co., Ltd.
Taiwan, R.O.C
ringo0601@cht.com.tw

Bo-Ting Lin
Telecommunication Laboratories Chunghwa Telecom Co., Ltd.
Taiwan, R.O.C
btlin1025@cht.com.tw

Chien-Wei Cheng
Telecommunication Laboratories Chunghwa Telecom Co., Ltd.
Taiwan, R.O.C
cwcheng@cht.com.tw

***Abstract*—In recent years, cloud computing is the most popular topic on the IT industry. The underlying virtualization technologies, that make cloud computing possible, also get more and more attention. Gradually, companies move their services to the virtual host. These services include: desktop virtualization, application virtualization and database virtualization etc. Among these services, database virtualization can improve flexibility, maximize efficiency, lower costs and ease administrative overhead. In this paper, we use on-demand features of cloud computing and sharding features of MongoDB to provide an auto-scaling database virtualization solution that satisfy the service-level agreement (SLA) requirements. First, we apply an auto-scaling mechanism of route server in the database system. The experimental results show that the average response time of auto-scaling DB solution and none-scaling DB solution are 4.3 seconds and 7.1 seconds, respectively. Second, in order to minimize the impact when moving data to a new VM, we also design a shard data migration algorithm for the database system. The auto-scaling DB solution uses the algorithm to determine how many VM should be added and which data should be moved to those added VM.**

Keywords—auto-scaling; nosql; database virtualization; SLA

I. INTRODUCTION

In IT industry, the term "*Big data*" means a collection of a massive volume of both structured and unstructured data. It is so large and complex that it becomes difficult to process using traditional database management tools or data processing applications. In recent years, how to efficiently store, process and query big data becomes an important issue. The traditional databases spend a high amount of processing time to achieve atomicity, consistency, isolation and durability (ACID) properties. In web 2.0 applications, the performance and real-time access of database is more important than ACID properties. One solution for handling such problems is to use NoSQL database.

Carlo Strozzi first used the term "*NoSQL*" in 1998 to name his open-source relational database that did not use the standard SQL interface[1]. A NoSQL database provides a simple, lightweight mechanism for storage and retrieval of data that provides higher scalability and availability than traditional relational databases. Nowadays NoSQL database systems broadly used by major internet companies, such as Google, Amazon, and Facebook, which had different challenges in dealing with huge quantities of data that traditional database solutions could not tackle. The advantages of NoSQL over traditional relational databases, such as elastic scaling, economics and flexible data models, are gaining attention in the enterprise. Often, NoSQL databases are categorized according to the way they store the data and fall under categories such as key-value stores, BigTable implementations, document store databases, and graph databases.

Many discussion and literatures about how to apply NoSQL have been proposed[2][3][4]. Waghmare[5] shows how to improve network scalability using NoSql database, and to discuss about some non-structured databases. It also discusses advantages and disadvantages of Cassandra and how Cassandra is used to improve the scalability of the network compared to RDBMS. Wylie[6] discuss the use of NoSQL databases in a framework that enables the application of computationally expensive models against a real-time network data stream. Liu[7] provides a view of the capabilities of NoSQL applied in global image system. MongoDB is based on document oriented NoSQL database written in C++ language. It is a solution for storage and retrieval of data that provides higher scalability and availability[8]. MongoDB achieves load balance by the strategy of distributing usage on each shard.

A database manager uses the over-provision strategy when setting up a database system. The strategy is to estimate reasonable requests and to prepare enough capacity for users. According our experience, an over-provision strategy

sometimes is inevitable. For example, an over-provision strategy is strongly recommended when building an online pre-ordered system or online ticket order system, since to satisfy SLA requirements is the most important factors. In this paper, we provide a mongos auto-scaling mechanism for such system in order to reduce unused computing resources of mongos instances.

When in initial stage of expanding database (add a new shard) and moving data, user's requests of query and the process of moving data will compete for using database resource under MongoDB architecture. It result in increase of database loading and degression of database performance. Therefore, in reality, a database manager usually expand database when load of database is lighter to avoid degression of database performance. But this way, it cannot fulfill suddenly increasing request. Otherwise, it utilizes the strategy that adds shard one by one until fulfilling SLA requirements and hence results in longer time for database expansion. To solve the above question, we also design a data migration algorithm for the database system. We use VM and MongoDB to realize rapid auto-scaling cloud storage system. MongoDB Sharded Cluster is set up on the VM. When a new shard is added, it merely move data with high-frequency usage to reduce data movement. In addition, we provide an algorithm to calculate the needed shard count instead of adding a shard one at a time, so we can reduce expansion time of the database system.

II. System Architecture and Method

A. System Architecture

The auto-scaling controller system includes seven components and an *Information Storage* as shown in Figure 1. The functional components are introduced as the following :

- *Workload Monitor*: Collect the information and the loading of data node generated by applications.
- *Resource Recorder*: Record which machine is added in the system.
- *Connection Switcher*: Tell users which machines are available and log query information from applications.
- *Workload Analyser*: Analyze the load of the current node and service quality provided by database and then report analysis to Decision Maker.
- *Decision Maker*: Integrate information provided by workload analyser and decide the number of shard to be added and how to migrate the data.
- *Machine Scaler*: The switch interface of Iaas. Control switch of the Iaas according to auto-scaling controller commands.
- *Data Migrator*: Indicate database how to migrate the data.

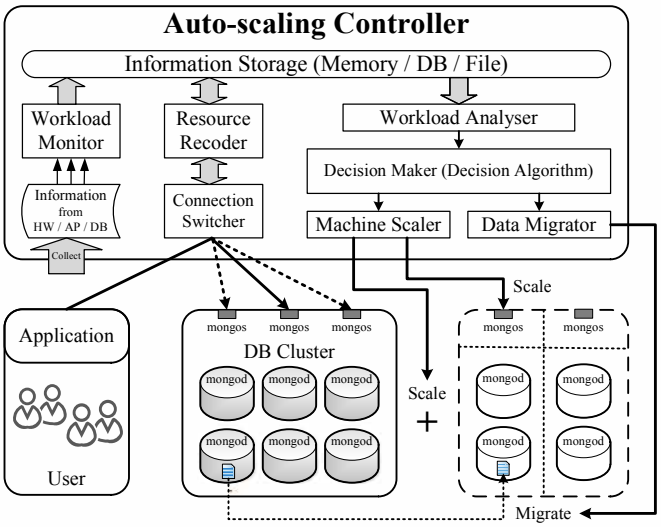
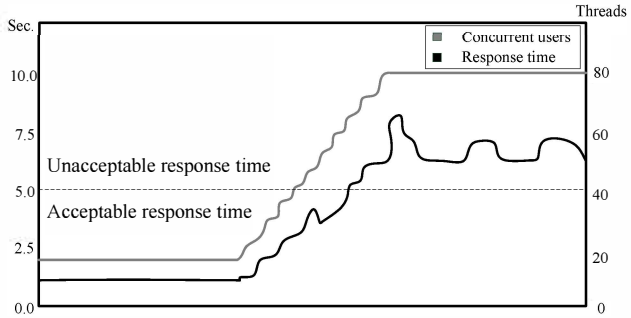


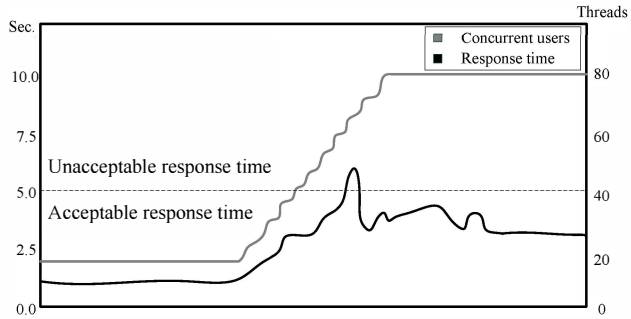
Figure 1. Auto-scaling architecture

B. Introduction of Auto-scaling Route Server

First, we provide a mongos auto-scaling mechanism for the database system in order to reduce unused computing resources of mongos instances. We set up MongoDB on the virtual machine. According to loading of the database system, machines where install mongos service are auto-removed from or auto-added to the cluster.



(a) Non-Scaling DB system



(b) Auto-Scaling DB system

Figure 2. Comparison graph of performance test

Experimental result shows as Figure 2. Figure 2(a) represents non-scaling DB system, and Figure 2(b) represents auto-scaling DB system. The grey line represents amounts of concurrent threads and bold line represents average user response time. We can observe that average user response

time increases with the increasing concurrent threads (from 20 threads to 80 threads). Loading of database also increases quickly. In Figure 2(a), without auto-scaling mechanism average user response time keeps above the unacceptable response time when benchmarking in third step of test. In the other hand, Figure 2(b) shows when system is busy (ex. User response time>Threshold), auto-scaling mechanism adds new route servers to achieve load balance where user response time keeps in acceptable range.

C. *Introduction of Auto-scaling Shard Server*

The purpose of Auto-scaling Shard Server is to decrease the expansion time for scaling database and to reduce the impact on database performance when scaling. To avoid chunks being moved accidently, we disable auto-balance function of MongoDB before applying our migrate algorithm. The whole execution flow is as the following:

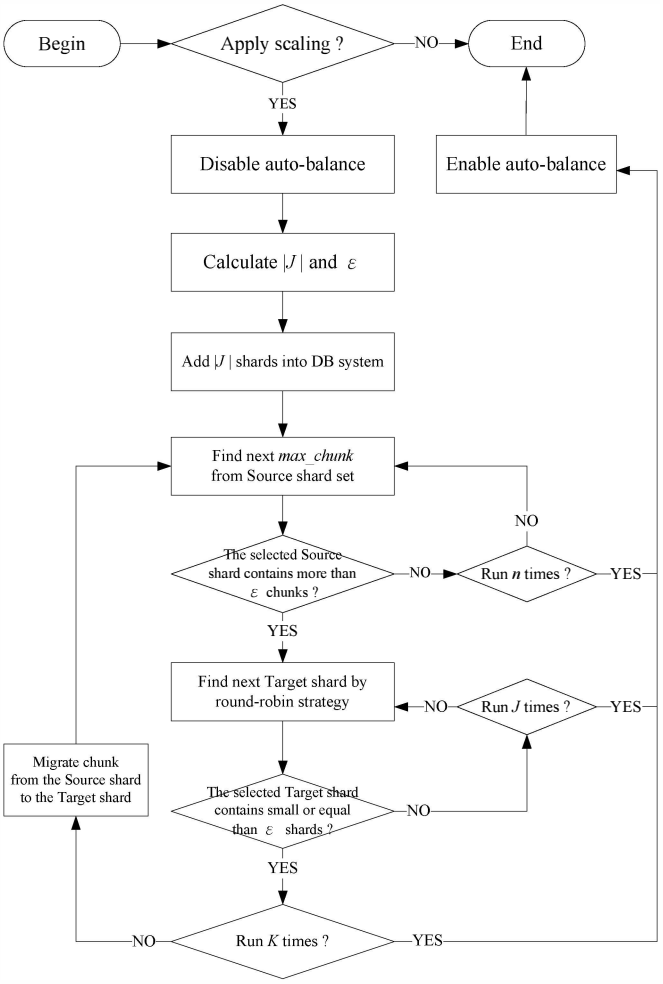


Figure 3. The flowchart of data balance process

The notations in Figure 3 are defined as follows. $|J|$ is the number of shard to be added. ε is the customized threshold. *max_chunk* means the maximum frequency chunk, and K is the max chunks to migrate. Please note that the number of chunks in each shard is uneven after applying our algorithm. In reality, it cannot fully utilize hardware storage. Therefore, we will enable auto-balance functionality to make all chunks distributed evenly in each shard at the last step of the algorithm.

III. CONCUSIONS AND FUTURE WORKS

In this paper, we use VM and MongoDB to realize rapid auto-scaling cloud storage system. We apply a mongos auto-scaling mechanism in the database system. The experimental results show that the average response time reduces about 40%. Moreover, in order to minimize the impact when moving data to a new VM, we also design a shard data migration algorithm for the database system. The auto-scaling DB solution uses the algorithm to determine how many VM should be added and which data should be moved to those added VM. In future work, our attempt is to apply the auto-scaling mechanism of shard on our web server. How to determine optimal parameters mentioned the previous section give rise to a number of new tasks for our future work so that our auto-scaling mechanism for distributed database could be moved towards a more realistic scenario.

REFERENCES

[1] Wikipedia. NoSQL. http://en.wikipedia.org/wiki/NoSQL

[2] J. Han, E. Haihong, G. Le, J. Du, "Survey on NoSQL Database," 6th International Conference on Pervasive Computing and Applications, 2011, pp. 363-366

[3] B. G. Tudorica, C. Bucur, "A comparison between several NoSQL databases with comments and notes," Roedunet International Conference , 2011, pp.1-5

[4] S. Huang, L. Cai, Z. Liu, Y. Hu, "Non-structure Data Storage Technology: A Discussion," Computer and Information Science (ICIS), 2012, pp.482-487

[5] Bhatewara, Ankita; Waghmare, Kalyani, "Improving network scalability using nosql database," International Journal of Advanced Computer Research, 2012, Vol. 2 Issue 6, pp. 488

[6] B. Wylie, D. Dunlavy, W. Davis, J. Baumes, "Using NoSQL databases for streaming network analysis," Large Data Analysis and Visualization (LDAV), 2012, pp.121-124

[7] Z. F. Xiao, Y. M. Liu, "Remote sensing image database based on NOSQL database," Geoinformatics, 2011, pp.1-5

[8] Kristina Chodorow. "scaling MongoDB". O' Reilly Media,January 2011. p13.

[9] Wikipedia. Virtualization. http://en.wikipedia.org/wiki/Virtualization

[10] 10gen. MongoDB. http://www.mongodb.org

[11] 10gen. Sharded Cluster. http://docs.mongodb.org/manual/core/sharded-clusters/

[12] Y. M. Liu, Y. Z. Wang, Y. Jin, "Research on the improvement of MongoDB Auto-Sharding in cloud environment," Computer Science & Education (ICCSE), 2012, pp.851-854