International Journal of Emerging Technologies in Engineering Research (IJETER)
Volume 6, Issue 10, October (2018) www.ijeter.everscience.org

Distributed Database for Water Supply Data Based on Hbase

Adlene Ebenezer [1], Gurpartap Singh [2], Adarsh Malav [3], Kishan Agarawalla [4]
[1] Asistant Professor, Computer Science department, SRM Institute of Science and Technology, Chennai,India
Computer Science department, SRM Institute of Science and Technology, Chennai,India

Abstract – Managing massive water supply data is a typical big data application because water supply systems generate millions or billions of records every single day. To guarantee the safety and sustainability of water supply systems, massive water supply data need to be processed and analysed quickly to make real-time decisions. Traditional solutions typically use relational databases to manage water supply data. However, relational databases cannot efficiently process and analyse massive water supply data when the data size increases significantly. In this paper, we show how water supply data can be managed by using HBase, a distributed database maintained by Apache. Our system consists of clients, zookeeper, HBase database, Hadoop Distributed File System (HDFS). We show's how HBase's parameters can be tuned to improve the efficiency of our system.

Index Terms – water supply data; HBase; database

1. INTRODUCTION

Everyone needs adequate amount of water in their day to day life. This need is fulfilled by water supply systems. There are water supply boards who supply water to every house hold. They have to keep record of every consumer. This record includes consumers personal details and their monthly consumption of water. Water supply board employees take reading of water consumption meter monthly from every consumer. Bills are generated accordingly. This huge amount of data is updated to database periodically. Such amount of data needs a lot of storage space. This is a good example of bigdata.

The water supply data processed by water supply boards are typical 4Vs data (data with characteristics of volume, variety, variety, velocity, and veracity) which are difficult to process, query, and analyse within a tolerable time. Traditionally water supply data are stored in relational database, where maintenance can rise due to data size and real time processing. These problems can be solved by introducing big-data storage systems into this fields.

Such huge amount of water supply data can be stored using cloud computing, distributed file systems and data management technologies such as Google File System [1], Hadoop Distributed File System (HDFS) [2] and Apache HBase [3]. Technologies like MapReduce [4] and Spark [5] make it possible to process water supply data in real-time.

Frameworks like Hadoop would not work for real-time queries as it can perform only batch processing. Instead, we propose a platform that uses apache HBase distributed database to store water supply data. HBase is an open-source, non-relational distributed database. It runs on top of Hadoop Distributed File System (HDFS) and provide BigTable like capabilities to Hadoop. HBase can perform faster read and write operations on larger datasets. It provides low input/output latency.

2. APACHE HBASE

Apache HBase is a distributed database based on BigTable. HBase is built on top of HDFS. MapReduce framework is implementation framework of HBase. Column family is basic unit of HBase data table. This family consist of one or more columns. In Figure 1 bill shows column family of a HBase data table which contains columns total and balance. There is a row key for every row which identifies data for related row. It is used for retrieving records in HBase. Frequently visited columns are placed in same column family to reduce query time. We can dynamically update data columns in column family according to the needs of applications.

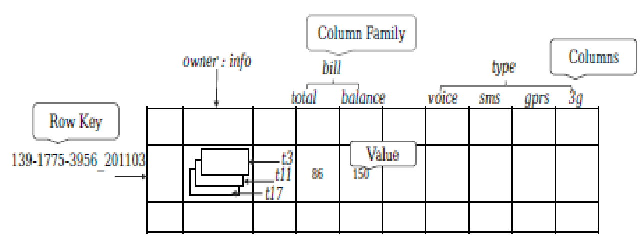

Figure 1 HBase storage system

The architecture of HBase consists of three major components: client, a master server and multiple region servers. Region servers contain tables and these regions are divided vertically into stores by column families. These are saved as files in HDFS. Master server is a cluster manager which assign regions to region servers with the help of Apache Zookeeper. It handles load balancing of regions across region servers

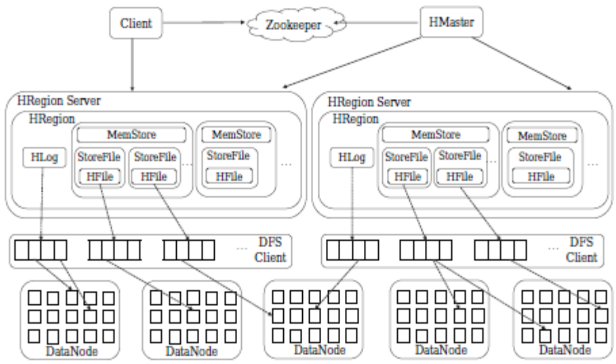

Figure 2 Architecture of HBase

3. PORPOSED MODELLING

The This system is set for managing massive data which comes from power supply board. System consists of client and data processing platform. Water supply boards upload data through clients. This data is stored and analyse by server using platform.

A. Data Format

Water boards provide data in different data formats. It is very laborious task to manage such data. Therefore, standardization of data format is very important. It is easy to store and process data in standardize format. It can improve speed and efficiency of system. In this system data in standardize format is stored. We can perform run-time queries for such data. So, this can solve data exchange problem and make it possible to share data among water supply boards and our database.

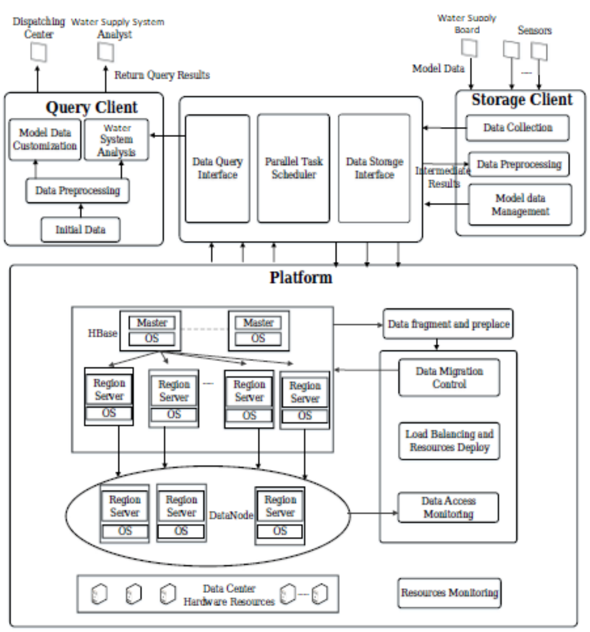

Figure 3 System design

B. Client

HBase database have clients that helps in storing and processing data stored in data base. This system contains two types of clients: query clients and storage clients. Storage clients calls platform-side storage interface to store data provided by water supply boards. Query clients answer queries submitted by data analysts and water supply board. These clients search for data in platform-site interface. This data is then converted into standardized format and returned to user. These clients are deployed in offices of water supply board.

C. Platform

The platform is the core part of our system and is responsible for storing and analysing data. The platform consists of database, status monitor, data migration, data fragmentation, and other modules, which are introduced in the following

- *Database*.

The database is responsible for storing and managing data. We use HBase to store and manage these data. When storing power data, the data should be split and stored across different computers. However, improper data fragmentation may increase the overhead of querying. Our data fragmentation strategy takes the graph partitions of the power grid network into account.

- *Status monitor*.

The status monitor collects real time CPU, memory, network, and disk I/O information of the servers in our platform, and sends the data to the workload balance controller. The traditional systems typically use open-source monitor software to acquire the status information, but the open-source software itself may introduce considerable running overhead. Instead of using the open-source softwares, our status monitor collects the server status by calling the operating system APIs.

- Workload balance controller.

To determine the time to migrate data, the workload balance controller detects each server's collected status. System adopts a straightforward approach based on threshold detection method. When the extent of workload unbalance reaches the threshold, data migration will start.

Data migration. When the servers' workloads are unbalanced, the data migration module generates a migration plan based on the server status. It then transfers data among servers according to the migration plan. The traditional data migration strategies are time-costly, because they need to redistribute all the data. To address this problem, system first calculates the number of data fragments that need to be migrated and then generates a corresponding data migration plan.

Water supply data consists of static and run-time data. Data containing details of consumers is uploaded to database only once, it is static data. Run-time data consists of details about monthly consumption of water and bills generated. This data needs to be uploaded frequently. This system uses HBase database to store data. The run-time data are generated in an incremental manner, which has low value density and large quantity. Figure shows how data is stored in database in form of tables.

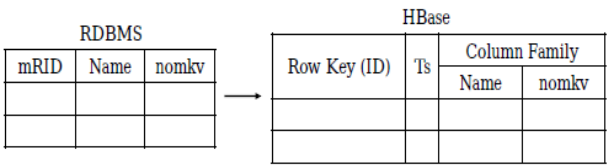

Figure 4 storing data on HBase

4. RESULTS AND DISCUSSIONS

We compare our system with a MySQL-based water supply database system on the 15-server cluster. In our experiment, we use eight water supply datasets sizes vary from 1 million tuples to 128 million tuples. We also perform a query that selects the error information of station 1 issued in January 2016 (select * from net where station id=1) on the datasets, and measure the query response time of our system and the MySQL-based system. We can see that our system is four times faster than the MySQL-based system (0.53 s vs 2.31 s) on the 1-million-tuple dataset, and 2.3 times faster than the MySQL-based system (73.82 s vs 173.86 s) on the 128-million-tuple dataset. Our system stores the error information in a column family; thus, it can efficiently respond to a query. The performance gap between our system and the MySQL-based system is smaller when the dataset is larger. When the dataset is larger, the data will be distributed on more servers, so the communication overhead cannot be ignored. Nevertheless, our system is faster than the existing MySQL-based system when we perform queries on large datasets.

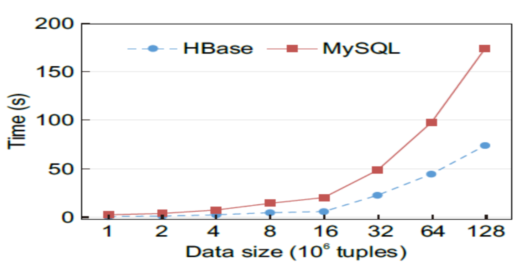

Figure 5 Comparison with SQL-based system

a) SCALABILITY

The experimental results illustrate that the storage performance is significantly improved when the number of storage servers increases. The running time is reduced by half when there are three storage servers. However, when the number of servers increases further, the performance improves slightly due server communication.

Furthermore, the number of clients has a great impact on system performance. Given the advantage of the distributed storage architecture, the write speed improves significantly when the number of clients increases.

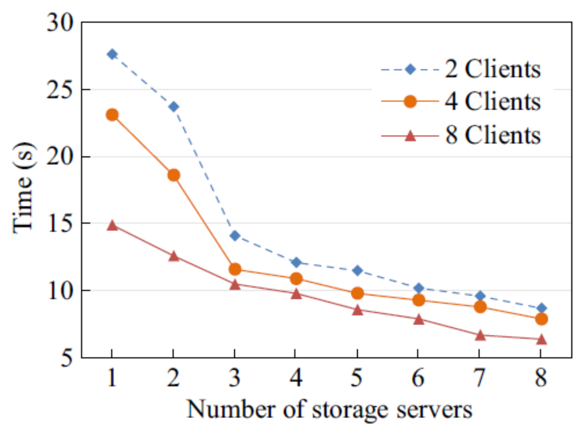

Figure 5 Scalability

5. CONCLUSION

In this paper, we introduce our system, which uses HBase to store water data. The system consists of HBase database, status monitors, data migration modules, and data fragmentation modules. We introduce the designs of the modules and evaluate their performance through experiments.

REFERENCES

[1] S. Ghemawat, H. Gobioff, and S. T. Leung, The Google file system, ACM SIGOPS Ope. Syst. Rev., vol. 37, no. 5, 2003
[2] Welcome to Apache Hadoop. http://hadoop.apache.org/, 2017.
[3] Apache HBase–Apache HBase Home, http://hbase.apache.org/, 2017.
[4] J. Dean and S. Ghemawat, MapReduce: Simplified data processing on large clusters, Commun. ACM, vol. 51, no. 1, 2008.
[5] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, Spark: Cluster computing with working sets, in Proc. 2nd USENIX Conf. Hot Topics in Cloud Computing, Boston, MA, USA, 2010, p. 10.