PROCEEDINGS OF SCIENCE

The Cloud-Based Sensor Data Warehouse

Wen-Yuan Ku[1]
GIS Center, Feng Chia University
100 Wen0Hwa Rd., Taichung, Taiwan
E-mail: cool@gis.tw

Tien-Yin Chou
GIS Center, Feng Chia University
100 Wen0Hwa Rd., Taichung, Taiwan
E-mail: jimmy@gis.tw

Lan-Kun Chung
GIS Center, Feng Chia University
100 Wen0Hwa Rd., Taichung, Taiwan
E-mail: peter@gis.tw

As advances in technology, sensors have been widely applied to various different fields. Sensor data with time characteristic, therefore after a long period of observation that demand for data storage and analysis are very large. The relational database is the most widely used database architecture that through the normalization process to design the table structure. The tables related to each other through the foreign key fields for data link. The greatest benefit of relational database is easy to manage data. The data after normalization can be avoided through the problem of inconsistent data. But the drawback is that data is stored using row-oriented which all rows in a table are the same. When the data is increasing exponentially such as the need to change table structure it takes more time to consume to restructure data. And, the traditional relation database often faces with the need to expand storage space problem. The most often way to solve this problem is to extend the storage space vertically, however, all data stored on a single server will cause a larger workload on the server.

This paper proposed a method for storing large-scale sensor data which using cloud-based distributed database - HBase. HBase is the Open Source version of the Google BigTable that is different from the row-oriented relational database. HBase used column-oriented paradigm that have highly flexible and more able to meet the needs of a variety of sensor formats. In this paper, we analyzed the data writing and reading performance based on large scale sensor data. The experiment proved that using HBase on large-scale sensor data with very good performance..

International Symposium on Grids & Clouds 2011,
Taipei, Taiwan
March 19-25, 2011

[1] Speaker

1. Introduction

The relational database is the most widely used database architecture that through the normalization process to design the table structure. The tables related to each other through the foreign key fields for data link. The greatest benefit of relational database is easy to manage data. The data after normalization can be avoided through the problem of inconsistent data. But the drawback is that data is stored using row-oriented which all rows in a table are the same. When the data is increasing exponentially such as the need to change table structure it takes more time to consume to restructure data.

The scale of sensor data increases along with time , therefore after a long period of observation, the demand for data storage and analysis are very large. The traditional relation database often faces with the need to expand storage space problem. The most often way to solve this problem is to extend the storage space vertically, however, all data stored on a single server will cause a larger workload on the server.

The cloud computing is considered to solve the large-scale of data analysis and storage problem. One of the characteristics of the cloud computing is high extendibility, flexibility and high fault tolerance. The cloud computing is used in distributed architecture which the data stored in distributed nodes in cloud and it can effectively improve the problem of overloading on a single server.

This research bases on a cloud database – HBase which is designed for supporting large-scale data storage environment. Real world vehicle driving records were used to test HBase under the large-scale data environment, this research can offer the researchers to evaluate the performance and provide future cloud research.

2. Backgroud

The invention of sensors for the nature of human observation, monitoring of the object or biological activities record has great benefits. The current sensing devices commonly used include: rain gauge, geophone, water level gauge, GPS, camera and so on. By sensor data which lets human understanding of the current situation and through the analysis of historical data can also discover information hidden in the vast and valuable information.

Figure1 shows a monitoring system architecture for commonly used front-end set up sensor data over the network back to database of control center. The storage of historical data sensors typically require a considerable amount of storage space, and if a huge amount of data in the analysis of the data need to remove the contents of the timing of this is to build a large database will be an important factor to consider.

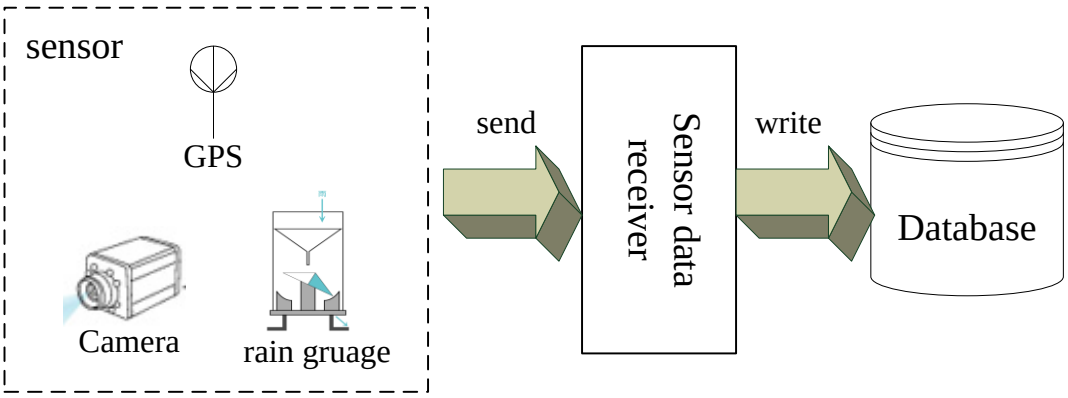

Figure 1 monitoring system architecture

For example, in commercial vehicle fleet management, GPS is installed in vehicles to detect its position, and motion sensor installed to detect whether the vehicle is in an abnormal movements, car hopper was open and other conditions, for instance.

In this paper, we use real world vehicle fleet management record data for testing which data amount of one day by an average of about 1.5 million records. A month will have 45 million records which need about 4G Bytes storage space. One year have 540 million records and need 48G Bytes of data storage space.

In spite of the fact that the cost of a hard drive is relatively low, and it's not a problem to maintain and operate a SAN-based storage server, however, the cumulative amount of data still causes the low accessing efficiency. In the real world fleet management company, they are facing two serious problems of storing huge amount of traffic records:

a. Cannot handle large number of sensor data returned: when the large number of vehicles returned at the same time to the server, the server is very easily to overload.

b. The efficiency of data querying is low: when database records continues to grow, the efficiency of indexing will be low increasingly, and results in a longer query response time.

3. Cloud-Based Database - HBase

In 2006, Google released Bigtable[1] publication which revealed the cloud on Google for the design of database architecture. Bigtable is widely used in Google's cloud-related services, such as: Google Earth, Google Analytics and Google Crawl. Google proved Bigtable is suitable for cloud computing by real applications because of its high availability.

HBase[2] is the open-source version of Bigtable that is a subproject of the Apache Foundation's Hadoop project[3]. HBase implements design of Bigtable that provides a flexible storage of large-scale data and support distributed database system. HBase and Bigtable both are column-oriented model. Its database model used key-value, which has the advantage of high query speed and high compression rate. The structure of storing data to HBase is a multi-dimensional sparse matrix, the key format is:

Value = (rowkey, column key, timestamp) …. (1)

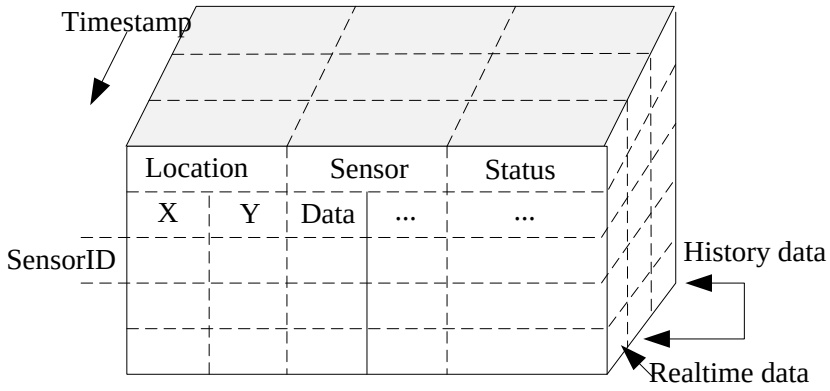

Figure 2 HBase storage

Figure 2 shown HBase storage structure. Each data storage unit on HBase has the property of versioning, we can identify these different data versions by using timestamp.

Table 1 illustrates a table schema of HBase. The indexed key is composed of row key, column key and the timestamp. Each row has a different number of data fields, and it can group a pack of data fields through the column family. The advantage of column-oriented database is to store data into the same storage unit, the efficiency of accessing is increased considerably.

Table 1 HBase table schema

```
RowKey : Key1 {
   Column Family A{
      Column X:
         T1 Value1
         T2 Value2
      Column Y:
         T3 Value3
   }
   Column Family B{
      Column Z:
         T4 Value4
   }
}
```

Table 2 shows the HBase conceptual view, the table has a row, columns and a row key is "Car-01". Each data has a timestamp to identity difference data version. The data row have three

columns include: <Region: >, <Info:Speed> and <Info:Status>. Table 3 shows HBase physical storage view, data will be stored according to row key + column key + timestampe stored respectively.

The major difference between column-oriented database row-oriented is that row-oriented database field structure of each row is fixed, however, the filed structure of each data row can be different, hence, the column-oriented database is more flexible.

Table 2 HBase Conceptual View

RowKey	Timestamp	Column "Region"	Column "Info"	
"Car-01"	T5		"Info:Speed"	"60"
	T4		"Info:Status"	"1"
	T3	"FengYuan"		
	T2	"DaYa"		
	T1	"Taichung"		

Table 3 HBase physical storage view

RowKey	Timestamp	Column "region"	
"Car-01"	T3	"FengYuan"	
	T2	"DaYa"	
	T1	"taichung"	
RowKey	Timestamp	Column "Info"	
"Car-01"	T5	"Info:Speed"	"60"
	T4	"Info:Status"	"1"

4. Design of Sensor Data Structure

There are two major factors to be considered in sensor data schema design, one for row key design another for the columns design [4,5,6]. These two factors will affect the performance of data accessing:

4.1 Row key design

HBase indexed key composed of row key, column key and the timestamp. In order to achieve optimal access that the row key of data will need to consider the structure. The major feature of sensor data is the data containing time information. So, we can design the row key of sensor data by using SensorID and time.

Through this definition, it can meet the requirement of data identification and query. The data content of sensor can be defined by the family column.

[RowKey]: <SensorID>_<YYYYMMDDHHmmss> (2)

4.2 Column design

HBase, which refers to Bigtable, adopts columns-based database architecture. Based on the documentation of HBase, Data row supports any number of columns; the same rows in the table also allow different number of columns. However, research shows that the time out situation may happen when columns number exceed 1000.At the same time, Dana also compared separately with the number of columns of the read / write efficiency estimation. The experimental result shows the number of columns affects the efficiency of datawrite.

Therefore, it is necessary to consider simultaneously that the data design of specific data columns will be updated or not. If there exist the possibility of being updated, we can give a thought to set up a column to store data. On the contrary, if the probability of the data column change is seldom, we can combine data into a column to reduce the frequency toward HBase IO.

Sensor data is provided with important data characteristic; each data of the sensor data is also an independent monitoring data. As long as the data stores into the database, there is little possibility to change it. For this reason, too much column design just makes the frequency of data storing increase on HBase. This outcome is bad for the operation efficiency. Therefore, the construction design of the senor data stores original sensor signal data in a single column to improve the efficiency of read/write.

For instance, Table 4 was the driving record data which sent back from vehicles; there were 9 columns information of the original data, each data used the symbol ’,’to distinguish the contents of the data columns.

Table 4 Vehicle driving data format

carid, datetime, x, y, status, speed, mileage, stay, sat
Car-01, 2010-10-01 12:00, 240012,2400123,80,210410,2,4
Car-02, 2010-10-01 12:40, 250560,2600456,80,210410,2,4
Car-03, 2010-10-01 12:51, 260120,2400013,80,210410,2,4
Car-04, 2010-10-01 12:52, 250000,2600000,80,210410,2,4
…
Car-nn, 2010-10-01 12:*0*0, 251201,2600022,80,210410,2,4

Table 5 illustrates the HBase table schema that divided into two tables: SensorBaseData and SensorLogData. SensorBaseData used for store the basic data of sensor and the table SensorLogData used for store raw data of sensor. Which SensorBaseData designed three columns are Name, LastData, UpdateTime and SensorLogData designed the two columns are SensorID and RawData to store the historical sensor data.

Table 5 HBase Schema Design

```
SensorBaseData
 RowKey [sensorid]{
    Column family [BaseInfo]{
      Column [Name]: Value
      Column [LastData]: Value
      Column [UpdateTime]: Value
    }
 }

SensorLogData
 RowKey [sensorid+time]{
    Column family [Sensor]{
      Column [SendsorID]: Value
      Column [RawData]: Value
    }
 }
```

5. Experiments

5.1 Experiment Environment

In order to test the performance of I/O of HBase with large-scale data, we configured a 3-node Hadoop cluster as shown in Figure 3. Each node was a 4-processor AMD Phenom server running at 2.3GHz with 4G of RAM. All nodes used Ubuntu 9.10, Hadoop framework 0.20.1 and HBase 0.20.3.

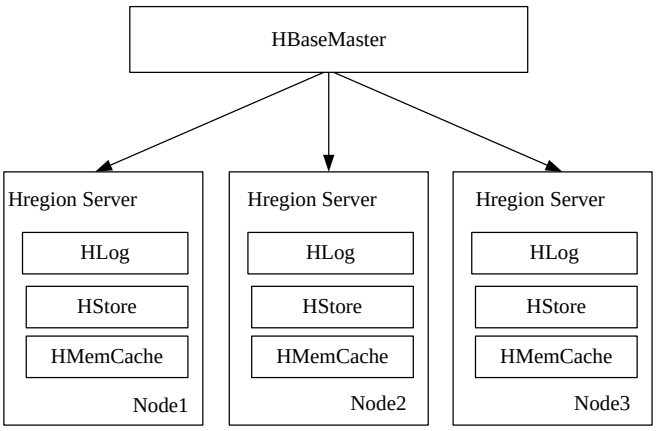

Figure 3 HBase Cluster Setup

In HBase, client sends a data access request to HBase through the HBaseMaster Metadata in order to seek the physical data is located at which HRegionServer[7], when the HRegionServer is located, the data will be accessed by HRegion. Therefore, this way can avoid from overloading of server when the concurrent connections are much.

5.2 Reading test results

In order to test the performance of reading large-scale data in HBase, 400 million records have been imported into HBase. We simulated 50, 100, 200, 300, 400 and 500 clients and read data randomly for lasting 2 minutes, then take the average response time.

Figure 4 shows the experimental test results. When the number of clients was 50, the average response time was 1.9ms. Next, the number of clients is set to 100 and the average response time is 2.1ms. We still change the number of clients, when number of clients is raised to 500 and the average response time is 5.6. After making this change, although when the client number grows from 50 to 500, and the average response time will grows. However, the response time is still able to keep less than 5.5ms. This is on 3 nodes cluster test result, if we increase the number of nodes then overall response time will be better than this test result.

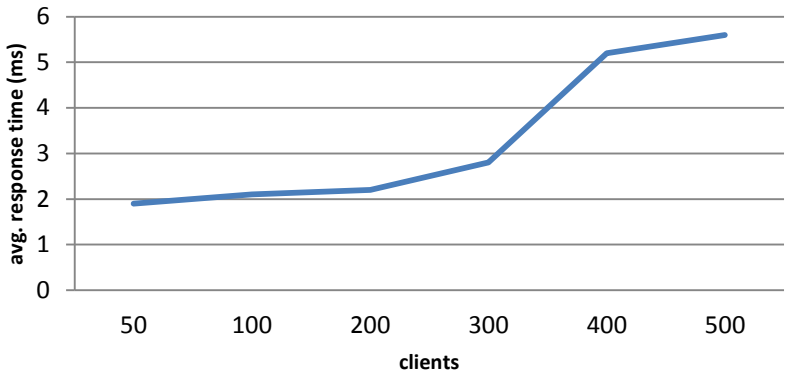

Figure 4. Reading test results

5.3 Writing test results

In order to understand the I/O performance, we carried out the reading test for single family column with increasing number of columns. In every single simulation, 1000 records have been written into 1, 10, 100 and 500 columns respectively. Table 6 shows the test results. Obviously, it took more time to write data into database in multi-columns mode. When the number of column is 1 that written at 914 columns per second, in other word, written at 914 rows per second. When the number of column is 10, the number of written column is almost same(914 columns/sec), but the number of written row will reduced to 1/10(94,9 rows/sec). When the number of column is 500, written performance will reduced to 1.92 rows per second. Therefore, from this test result, we get a conclusion that more number of columns in single record then written time will be longer and the number of columns will affect the write

performance of Hbase. This is beacuse Hbase is a colmun-oriented database, data necesaary to write column by column. This test result is similar with the conclusion of Dana[4].

Table 6 the different number of columns testing results

	# of columns			
Experiment	1	10	100	500
Write Columns/Sec	914	949	900	961
Write Rows/Sec	914	94.9	9	1.92

6. Conclusions

The sensor data has a characteristic of time and daily capacity usage is calculated in GBs of data. The traditional relational database is difficult to carry such large-scale of data access. In addition, a large number of connections connect to relational database that the performance will be significantly decreased. This paper proposes an idea of manage sensor data that used cloud-based database – HBase. We focused on how to design HBase table schema and analyze the performance on data writing and reading.

As HBase using column-oriented architecture, it requires to write data column by column to the database. By experimental test results, the number of columns in the table affects the performance of data accessing. When data row have more columns it will increase the data access time. By our experiment result, it will increase the efficiency of database I/O if data can be converted to XML format and save XML data to single column.

To test the reading efficiency, this paper imported 400 million sensor data into HBase. Through the stress test, we simulated 50 to 500 clients for accessing to the HBase at the same time. By the experimental test results it shows the response time of data reading is less than 6ms in large-scale sensor data. It proves that HBase has a very good reading performance.

References

[1] 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, Google Inc., 2006.

[2] HBase, Web page, http://hadoop.apache.org/hbase

[3] Hadoop, Web page, http://hadoop/apache.org/hadoop

[4] K. Dana, Hadoop HBase Performance Evaluation, http://www.cs.duke.edu/~kcd/hadoop/kcd-hadoop-report.pdf.

[5] A. Rao, S. Zang, HBase-0.20.0 Performance Evaluation, http://cloudepr.blogspot.com/search/label/HBase.

[6] D. Carstoiu, A.Cernian, A. Olteanu, “Hadoop Hbase-0.20.2 performance evaluation”, New Trends in Information Science and Service Science(NISS), 2010 4th International Conference on, 2010, pp84-87.

[7] HBaseArchitecture, http://wiki.apache.org/hadoop/Hbase/HbaseArchitecture