Applied Mathematics & Information Sciences
An International Journal

Data Migration from Grid to Cloud Computing

Wei Chen[1], *Kuo-Cheng Yin*[1], *Don-Lin Yang*[1] *and Ming-Chuan Hung*[2]

[1]Department of Information Engineering and Computer Science, Feng Chia University, 40724 Taichung, Taiwan
[2]Department of Industrial Engineering and Systems Management, Feng Chia University, 40724 Taichung, Taiwan

Received: 2 Jan. 2012; Revised 4 Jun. 2012; Accepted 17 Jul. 2012
Published online: 1 January 2013

Abstract: The advance of information and communication technologies over the last decade has improved the quality of healthcare and medical services tremendously. Especially, the people living in the countryside or remote areas benefit the most from telemedicine and emergency services. Our Health Grid is one of the test beds which provide various health-related Web services associated with mobile devices and physiological data acquisition and analysis instruments. As the number of new applications being developed increases rapidly, the ever-growing volume of collected data and real-time demand of analysis result have driven the architectural migration from Grid to Cloud Computing much sooner than we expected. Our challenge is to make the transition cost effective. This paper describes the data access migration from a relational database to Apache's HBase - one of the cloud databases. Our contribution is to minimize the required change of software for data access. Since the SQL commands of the relational database cannot be used in HBase, various mechanisms for translation and mapping between two sides must be developed. In addition, the services provided by the Web programs in Health Grid are written in various kinds of Web language while HBase does not support the access authority to these Web languages. To reduce the effort of modifying the source code for accessing HBase, we propose the use of Web services as the communication interface between various Web programs and necessary facilities to execute SQL commands in HBase. Although this is a hard engineering work, our results show that the proposed approaches are feasible and cost effective for the development teams at academic institutes. With this preliminary study, our next step is to improve our methods to take advantage of the efficient functions of HBase in processing the large amount of data.

Keywords: Database, Grid, Cloud computing, Healthcare, Web service

1. Introduction

In this information age, the amount of generated data grows exponentially every year. To take on this enormous challenge, a new paradigm shift is needed. One might favor a move from computation-centric to data-centric approach and the other would prefer both of them at the same time. Cloud Computing [1] [2] [3] may be regarded as the most interesting platform to meet the requirement at this time. The cloud connects a vast number of distributed processing nodes to form a virtual processing center. It can support on-demand services with expandable processing power and fast data access via its distributed file management. However, most databases supported by cloud computing platforms are non-SQL, such as GFS [4] of Google, HBase [5] of Apache Hadoop plan, Carssandra [6] and so on. These databases are designed specifically for cloud computing environments, thus they don't support SQL syntax like the relational databases do. Therefore they are called Non-SQL databases. Instead of complex SQL commands, only two simple I/O instructions, put and get, are used to access a full column. The traditional relational databases using SQL commands to access data are not appropriate for cloud computing at this time.

If we move existing systems to the cloud computing environment, we have to face the following three problems regarding the transformation of platform and databases. First, the column oriented database was not developed long enough to provide suitable I/O libraries for different programming languages. Second, most of the system developers are not familiar with column-based databases. When one migrates to the cloud computing platform, all the programs using the SQL syntax have to be changed. It is really a heavy burden to rewrite program code. Third, the column based database just uses put and get instructions. Many powerful and commonly-used SQL commands, such as

* Corresponding author: e-mail: dlyang@fcu.edu.tw

Join, cannot be used in the cloud any more. It is another big challenge for developers.

The main goal of this paper is to find the most effective way for Web based system developers to avoid the above problems. The developers still can use SQL commands to directly access the column based database without rewriting program code. To simplify our presentation, we focus on a specific column based database, HBase of Apache. Here, we propose a SQL-HBase mapping model to facilitate efficient interfaces for developers to directly use SQL commands to access HBase without modifying the program. Combining with the use of Web services, we can access HBase no matter what platform we use.

We investigate and evaluate our SQL-HBase mapping model on a private Health Grid [7] under development. We intend to have a cloud version of Health Grid (called Health Cloud) in the future. Our Health Grid uses an MS SQL Server as the system database and it is implemented with ASP.NET and PHP. In order to migrate to a cloud computing environment, we setup HBase on Ubuntu and transfer the data of MS SQL Server to HBase.

The rest of paper is organized as follows. In Section 2, we review the databases in cloud computing environments. In Section 3, the detail of the proposed method is described. In Section 4, the operation process of SQL-Mapping model is given and the experimental results are shown. Finally, we make a conclusion and present future work in Section 5.

2. Related Work

2.1. HBase

The rapid growth of data requires new information technologies to solve the problem. Apache Hadoop [8] is an open-source project for reliable, scalable, distributed computing and data storage. Hadoop not only has a distributed processing platform but also has a sequential and batched file system, called Hadoop Distributed File System (HDFS). Google had developed a distributed file system, called BigTable to successfully store a large amount of structured data. HBase [9] is a solution similar to BigTable and is developed by the Hadoop team. HBase and BigTable adopt column-oriented approach to store data instead of row-oriented process in the relational database. The advantage of column-oriented access is that a record can have a variable number of columns. HBase takes the advantage of a distributed file system and partitions a table into many portions which are accessed by different servers in order to achieve high performance.

2.2. MapReduce

MapReduce [10] is a software framework introduced by Google and has been used in global communities like Yahoo, Facebook and so on. The superiority of MapReduce is able to store and process a large amount of data by using commodity hardware. However, there is a high threshold for developers to write Map-Reduce programs and the cost of maintenance and reuse is very high too.

Using MapReduce to build systems, the developer first analyzes the problem in parallel manner and finds the parts which can be processed in parallel. In other words, those portions which can be partitioned into small tasks will be written to Map program. Then, one can use a large number of machines to execute Map programs and analyze data in parallel. The results of all Map programs are merged by Reduce program and finally to be output for report.

2.3. SQL in Cloud

The typical works of MapReduce with database systems include Apache's Hive [11], Yale's HadoopDB [12], and Microsoft's SCOPE [13]. [4] proposed a new type of data warehouse named Hive. Hive supports HiveQL which is like SQL. It can insert pre-written MapReduce scripts when executing a query. Through HiveQL and data exploration, query optimization, query compilation of Metastore, users can easily complete Map-Reduce programming and query data. Hive is written by Data Infrastructure Team of Facebook. In Facebook, Hive processes tens of thousands of tables and the amount of report data and ad-hoc analyses per month is over 700TB. However, HiveQL is still slightly different from SQL. For migrating the system developed on RDBMS to Hive and Hadoop, we have to rewrite programs and take a lot of time to make modification and adjustment. Furthermore, very few programming languages support Hive. Java and python are two main Hive programming languages. The transition to use cloud platform is hard for the programmers using other programming languages.

Jing Zhao [6] proposed an algorithm using SQL commands to access structured data and still enjoy the efficiency of cloud computing. The performance of the cloud platforms that can execute SQL commands on traditional relational databases is not satisfactory. On the other hand, the databases suitable for cloud computing don't process structured data well. Jing Zhao et al. got inspiration from MapReduce of Google and proposed the ESQP algorithm. It partitions a job into several tasks for many cloud nodes to execute and divides a database into many units. Every unit is stored in a node in a distributed and replicated manner. In this way, one SQL command is executed in multiple nodes. In order to speed up the process further, the algorithm also adopts pipeline scheduling and other management methods.

The ESQP algorithm performs well in JOIN and non-JOIN SQL commands in distributed file systems. Every SQL command task can be partitioned into subtasks for processing in multi-nodes. However, ESQP just focuses on the read SQL command, while the write command (for example, INSERT) is not included. The database of ESQP has to be partitioned at the beginning and distributed to

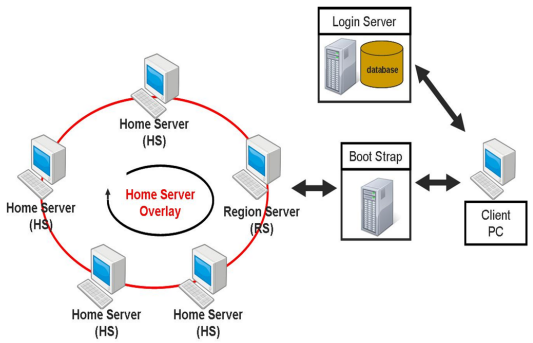

Figure 1 The Architecture of Health Grid

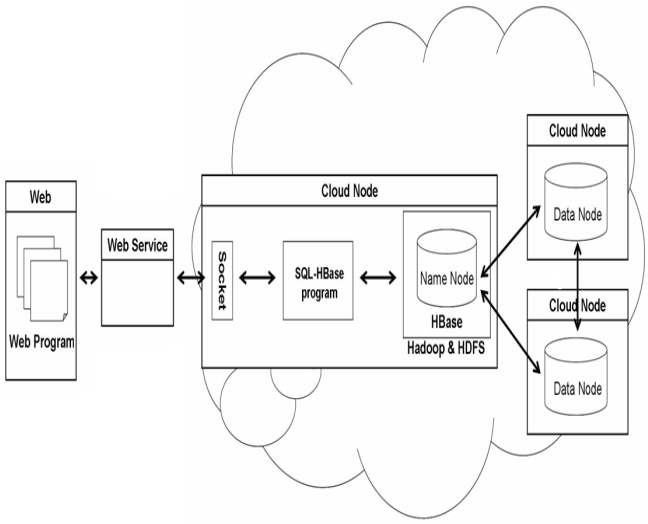

Figure 2 The Architecture of Health Cloud

nodes. It belongs to the static database algorithm and is not suitable for our environment.

3. Proposed Method

Our Health Grid is one of the test beds which provide various health-related Web services associated with mobile devices, physiological data acquisition and analysis instruments. Figure 1 shows the architecture of our Health Grid. When migrating our system from Grid to Cloud, the existing Web programs in Health Grid written in various kinds of Web language cannot be executed in HBase.

We propose a method of SQL-HBase mapping to allow the execution of SQL commands in HBase. The purpose is for the original programs of Health Grid to access HBase without rewrite or considerable modification. This will save a lot of time for system developers.

3.1. Architecture

Figure 2 shows the architecture of our Health Cloud after migrating from Health Grid. The left side of Figure 2 is the Web program environment. It contains various Web programs for clients. The right side is a cloud representing the cloud environment. A cloud database contains as many cloud nodes as needed. Typically, a cluster of such nodes has one name node and multiple data nodes. The Web service, in the middle, is the gateway of data communication and exchange. The name node contains all HDFS metadata and handles Web services.

A Web service is like an external calling function. It can pass parameters to the called function and return a value to the calling function. When a Web program needs to access the database, it packs the SQL command into a string as a parameter and the Web service transmits it to the cloud node running an instance of the name node. After the name node has completed the process, the Web service returns the result to the Web program and finishes the database access. However, the user data need not flow through the name node.

Although Web services can transmit the SQL command to a name node, it still cannot directly access HBase due to privilege and security concerns. Therefore, we need a middleware to translate the SQL command and access HBase. The middleware contains socket programs and SQL-HBase programs written in JAVA as shown in the cloud of Figure 2. A socket program then opens a specific port to listen to the SQL command from Web services. After accepting the SQL command, the socket program establishes a thread to call SQL-HBase programs. The SQL-HBase program translates the SQL command to access HBase. After retrieving HBase, the socket program returns acquired data to Web services. Finally, the Web service returns the result to the calling Web program.

3.2. Relational Database and Column Based Database

Before explaining how the SQL-HBase program accesses HBase, we first introduce the characteristics of a column based database. HBase is one kind of column based databases. The table format has five items as shown in Figure 3.

1. Table name: It is the name of a table. There can be lots of tables in an HBase database. It is the same as a table in a relational database.
2. Rowkey: Rowkey is the key of a row. It is like the primary key (PK) in a relational database.
3. Column family and column label: One column family consists of several column labels. In other words, one column family can record several values. When adding new column labels, the column family can record more values.

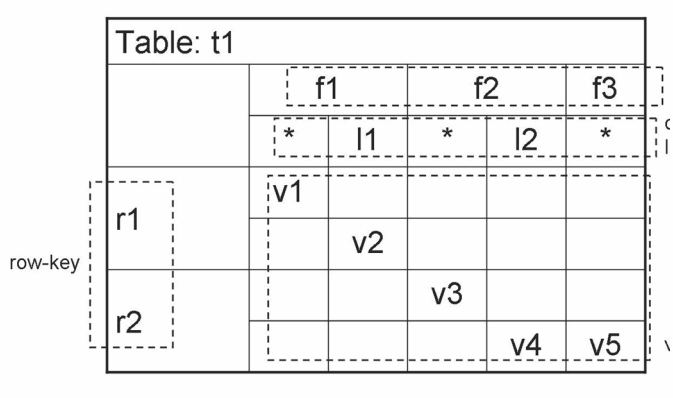

Figure 3 Table format of HBase

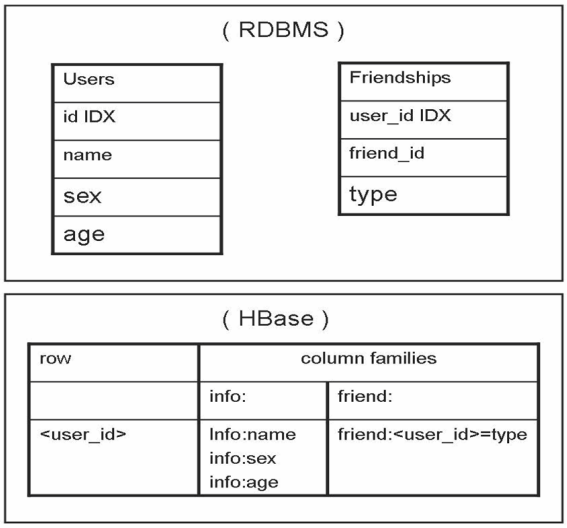

Figure 4 The mapping example of RDBMS and HBase

When reading data from a database, we can get all the column labels from the column family. Column family has to be planned when establishing the database. But column labels can be added as needed. Therefore, column label is not required and becomes an option. In Figure 3, f1:l1 is an example of column label_l1 in the column family_f1 and f1:* is another example of the column family_f1 without any column label.

4. Value: The place to record a value.

The main advantage of the column family and column label is that we can easily add new data into a database as needed. It is easy to increase the required storage space, but also speed up the access of database. We can take advantage of expandable storage space in a cloud environment. If there is a short of storage space, we can add new data nodes to solve the problem. Now, we use a simple example to explain the difference between a relation database management system (RDBMS) and an HBase.

Here, we design an application system of RDBMS to manage the information of friends. Because the relation between a user and his/her friends is one to many, the data have to be partitioned into two tables, Users and Friendships, as shown in Figure 4. These two tables are related by the key id-IDX in the Users table and user_id IDX in the Friendships table. Now we show how to develop the same system in HBase. Since we can add new column labels as needed, the Users and Friendships data can be recorded in the same table and in the same row. We do not need to create a relation for them either. It makes the data input and process more direct. With large data volumes, it is advantageous to use the approach of HBase.

3.3. Analysis of SQL-HBase Mapping

Because the library of HBase supports Java, Java programs can directly access HBase. The SQL-HBase program is written in Java and its main function is to transform SQL commands into the code which HBase can realize.

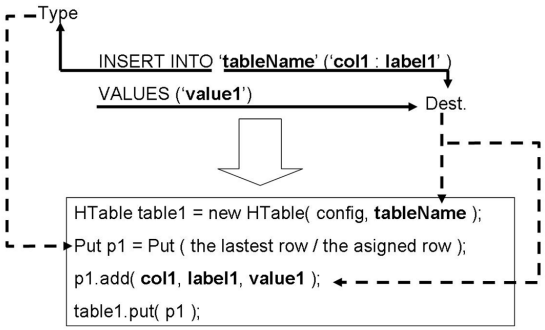

Figure 5 The example of "Insert Into"

Due to space limitations, we only take the SQL command 'Insert into' as an example to illustrate the process of SQL-HBase program. In Figure 5, after accepting the SQL command 'Insert into', we partition it into two parts, Type and Dest. Type records the kind of an SQL command. Dest records which 'table', 'PK value', 'column family and label' to be processed and what the 'where' condition is. In Figure 5, the SQL command consists of table name-'tablename', column family-'col1' and value-'val1'. It lacks of row-key and column label. To execute the 'Insert into' command, the relational database system will append its data after the last record. So SQL-HBase program will automatically access the last row-key and add one as the new row-key. As to the column label, it is optional and can be ignored.

After making sure the presence of five items- table name, row-key, column family, column label and value, the SQL-HBase program follows the code in Figure 5 to

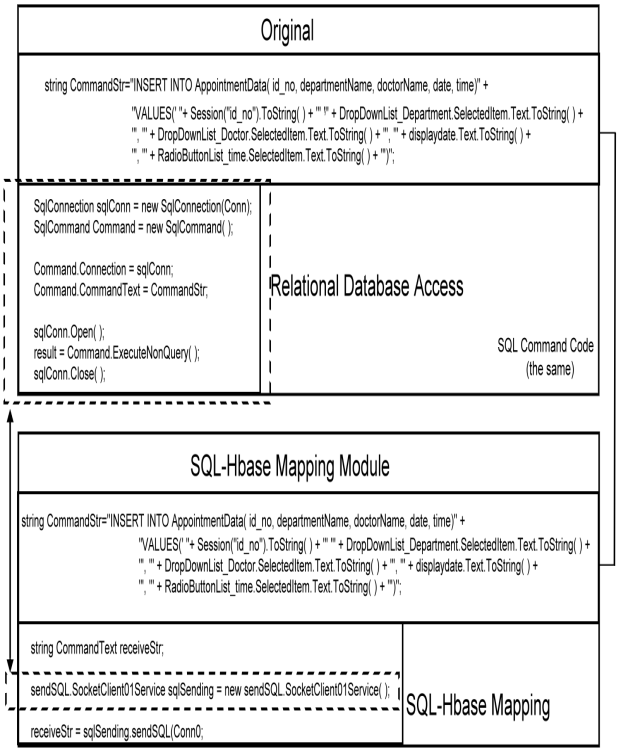

Figure 6 The comparison of program codes

assign table and row-key in order. Then it puts the value to the assigned column family and label. If there is any problem, using the insertion of an existing value as an example, we can use 'try and catch' module to log where it occurs and report the error message for correction later.

4. Discussion and Experiments

To show the correctness and effectiveness of our approach, we will make the comparison between the codes of original and SQL-HBase mapping, discuss the type of mapping, and show performance results.

4.1. Analysis of SQL-HBase Mapping

Figure 6 shows a portion of program code for our Health Grid on the top of the figure. It is the original code which uses ado to access an SQL server. The modified code of SQL-HBase mapping is shown on the bottom of Figure 6. When applying the module of SQL-HBase mapping, the programmer just needs to do two things. One is to use the Web service of this module as below:
SendSQL.SocketClient01Service sqlSending =
new sendSQL.SocketClient01Service ();

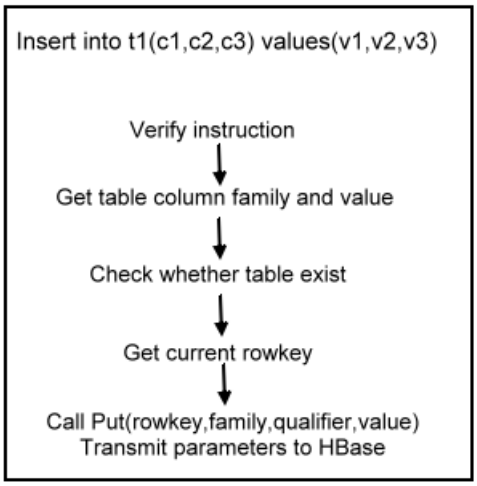

Figure 7 The example of Mapping

The other is to put the original SQL command as a string parameter in the Web service as follows.
string CommandStr;
sqlSending.sendSQL (Conn);

This way, we do not need to rewrite SQL command and can execute the original code to access HBase after setting up the cloud environment. The details of HBase APIs will not be discussed here.

4.2. The Type of Mapping

In this section, we would like to inspect the four SQL commands - Insert, Delete, Update and Select to show how to map them into the API of HBase. Since they are similar, we only depict the Insert due to space limitations. There are four elements - rowkey, column family, column label and value, related to an SQL command. As shown in Figure 7, we will explain how to map the relational database to HBase. SQL command:
Insert into t1 (c1, c2, c3) values (v1, v2, v3)→
Corresponding HBase API: Put (rowkey, family, qualifier, value)

First, we verify the instruction and get parameters. Then, check whether the table name exists in HBase. If not, we use Get.getRow( ) function to get the last rowkey and add one to the last rowkey as the current rowkey. To fill the rest of the put call, the family name is the same as the column name (c1, c2, c3) in the relational database. To reduce the complexity, the qualifier is set as Null. The value is the same as the value in the relational database. The mapping is straightforward, which is the principle rule in our approach.

Figure 8 shows a real example from our mapping process of Health Grid to Health Cloud migration in the experiment.

```
<SQL  INSERT>
INSERT INTO SportData( id_no, inpdate, sport_way, sport_type, sdate, edate, number,
length) VALUES( 'H123342673', '2011/3/22', 3, 4, '2011/3/14', '2011/3/17', 15, 2.5 )

<SQL-HBase Mapping>
HTable table = new HTable( conf, SportData )
Put p = new Put( The Last rowkey+1 )
p.add( id_no, "", 'H123342673' )
p.add( inpdate, "", '2011/3/22' )
p.add( sport_way, "", 3 )
p.add( sport_type, "", 4 )
p.add( sdate, "", '2011/3/14' )
p.add( edate, "", '2011/3/17' )
p.add( number, "", 15 )
p.add( length, "", 2.5 )
table.put( p )
```

Figure 8 A real mapping example of "Insert into"

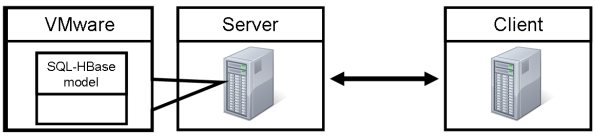

Figure 9 The environment of implementation

4.3. Experimental Results

To show the correctness and effectiveness of our approach, we perform some experiments to evaluate SQL-HBase mapping algorithm.

4.3.1. Experimental Environments

We implemented our approach in two computers as shown in Figure 9. One functions as a server to support the SQL-HBase mapping service. The other functions as a client to invoke requests. The client has Windows XP sp3, Intel Xeon E5520 2.27GHz processor and 8GB RAM. We use IIS 6.0 as the web server and write Web programs in ASP.NET. The server has Windows XP sp3, Intel Core Q9400 2.66 GHz and 2GB RAM. The virtual environment includes Ubuntu 10.04 with 512MB RAM and VMware. The SQL-HBase mapping programs are written in JAVA and run on the virtual environment of the server. The Hbase with version 0.90.3 is also running on the virtual environment of the server. The test dataset, DRUG2006, comes from National Health Insurance Research Database (NHIRData) [14] which is supported by the Bureau of National Insurance of Taiwan. DRUG2006 is a dataset of medicine and contains 25091 records of data.

4.3.2. The Correctness of SQL-HBase Mapping

To verify that the proposed method is correct, we devise an experiment as shown in Figure 10. First, we read data

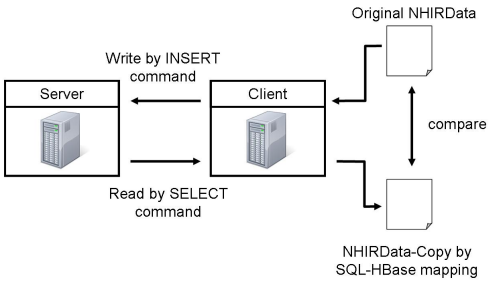

Figure 10 The experiment of correctness

from NHIRData on the client side as the original NHIRData. Then, use 'Insert into' command to store the data into Hbase on the server side through SQL-Hbase mapping. Next, use 'select from' command to get the stored data from Hbase on the server side with a new name of NHIRdata-copy. Then we compare the difference between the original NHIRData and NHIRData-copy. The results show that there is no difference between them.

4.3.3. Performance Evaluation

To evaluate the performance of our approach, we measure the time of transformation by the SQL-HBase Mapping. The experiments are performed to calculate the following time:
(1) A SQL command is transmitted to Web services,
(2) The Web service transmits it to the SQL-HBase Mapping and
(3) The SQL-HBase mapping transforms the SQL command to the syntax of HBase.

We design ten kinds of different SQL commands. They are randomly selected to execute 100, 200, 300 and 400 numbers of times. Then we select 10 samples from the results of each execution. Finally, the average time is computed as shown in Figure 11. The results show that, in average, it takes 0.4 second to test 100 times. In other words, it takes 0.004 extra second to execute one SQL command using our SQL-HBase Mapping. This overhead is much less than the considerable efforts required to rewriting the program code for the existing system to migrate to the cloud environment. Our experiments are performed specifically for the SQL-HBase mapping.

5. Conclusions and Future Work

Using the SQL-HBase mapping approach, our implementation can facilitate the Web-based program to use SQL commands to access HBase effectively. In this way, the existing non-HBase programs can access HBase via Web services without rewriting their programs or making considerable modifications. Because the core of Web services

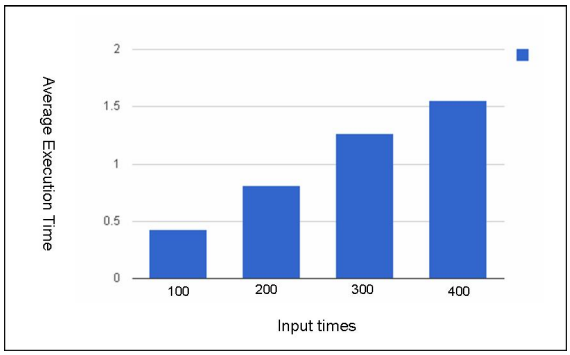

Figure 11 Average execution time of SQL-HBase Mapping

is in the form of XML and most Web programs support XML format, we can use the Web service to achieve cross-platform objective and reduce the effort of migrating existing systems to other new platforms. The experiments show that our approach is feasible and cost effective.

To meet the requirement of our Health Grid to Health Cloud migration, our approach just supports the SQL Commands of 'insert', 'delete', 'update', 'select', basic 'join' and the operators of '+', '<', and '>'. With the experience we gained from this research, we will take more complex SQL commands into consideration in the future. In addition, we would like to enhance the fault tolerance of our modules with some debugging functions. When testing the mapping between two sides, it is very important to trace and pinpoint where the problem is and which side is at fault.

Next, we want to improve the interface for users to complete SQL-HBase mapping more efficiently. Currently, we transform the SQL command line by line. It lacks the resilience on choosing row, column family and label. Furthermore, we have not taken the problem of socket collision into consideration yet. When the number of users increases, how to prevent collision effectively is an important research topic.

Acknowledgement

This research was supported by the National Science Council, Taiwan, under grants NSC 99-2218-E-007-001 and NSC 100-2221-E-035-103.

References

[1] M. Lynch. Amazon elastic compute cloud (Amazon ec2). [Online]. Available: http://aws.amazon.com/ec2/.
[2] IBM. IBM introduces ready-to-use cloud computing. [Online]. Available: http://www-03.ibm.com/press/us/en/pressrelease/22613.wss.
[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, The Google file system, in Proceedings of SOSP'03 29-43 (2003).
[4] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Ning Zhang, Suresh Antony, Hao Liu and Raghotham Murthy, Hive - a petabyte scale data warehouse using Hadoop, Data Engineering (ICDE), IEEE 26th International Conference 996-1005 (2010).
[5] F. Chang, J. Dean, S. Ghemawat, W. Hsieh, D. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber, Bigtable: A distributed storage system for structured data, In Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation 205-218 (2006).
[6] Jing Zhao, Xiangmei Hu and Xiaofeng Meng, ESQP: An efficient SQL query processing for cloud data management, Proceedings of the second international workshop on Cloud data management pp.1-8 (2010).
[7] Wei Chen, Don-Lin Yang, Ming-Chuan Hung, and Jungpin Wu. An Omnipresent Personal Health Management System, Proceedings of 2011 International Conference on Computing and Security 1-7 (2011).
[8] Hadoop. [Online]. Available: http://hadoop.apache.org/.
[9] HBase. [Online]. Available: http://hadoop.apache.org/hbase/.
[10] J. Dean and S. Ghemawat, MapReduce: simplified data processing on large clusters, Communications of the ACM. Vol. 51, 107-113 (2008).
[11] A. Thusoo, J. Sen Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff and R. Murthy, Hive: a warehousing solution over a map-reduce framework, in Proceedings of VLDB'09 922-933 (2009).
[12] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz and A. Rasin, HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads, in Proceedings of VLDB'09 922-933 (2009).
[13] R. Chaiken, B. Jenkins, P. Larson, B. Ramsey, D. Shakib, S. Weaver and J. Zhou, SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets, in Proceedings of PVLDB'08 1265-1276 (2008).
[14] National Health Insurance Research Database, http://nhird.nhri.org.tw/index.htm.

Wei Chen received the B.E. degree from the Department of Information and Computer Engineering at Chung Yuan Christian University, Taiwan, in 2008. Currently, Mr. Chen is working on his M.S. degree in the Department of Information Engineering and Computer Science at Feng Chia University. His research interests are data mining and cloud computing.

Kuo-Cheng Yin received the B.E. degree and M.S. degree from the Department of Information Engineering and Computer Science at Feng Chia University, Taiwan, in 1990 and 1992 respectively. He is now a Ph. D. candidate in the Department of Information Engineering and Computer Science at Feng Chia University. His research interests include data mining and software engineering.

Don-Lin Yang received the B.E. degree in Computer Science from Feng Chia University, Taiwan in 1973, the M.S. degree in Applied Science from the College of William and Mary in 1979, and the Ph.D. degree in Computer Science from the University of Virginia in 1985. He was a staff programmer at IBM Santa Teresa Laboratory from 1985 to 1987 and a member of the technical staff at AT&T Bell Laboratories from 1987 to 1991. Since then, he joined the faculty of Feng Chia University, where he was in charge of the University Computer Center from 1993 to 1997 and served as the Chairperson of the Department of Information Engineering and Computer Science from 2001 to 2003. Dr. Yang is currently a professor at Feng Chia University. His research interests include distributed and parallel computing, image processing, and data mining. He is a member of the IEEE computer society and the ACM.

Ming-Chuan Hung received the B.E. degree in Industrial Engineering and the M.S. degree in Automatic Control Engineering from Feng Chia University, Taiwan, in 1979 and 1985 respectively, and the Ph.D. degree from the Department of Information Engineering and Computer Science at Feng Chia University in 2006. From 1985 to 1987, he was an instructor in the Mechanics Engineering Department at National Chin-Yi Institute of Technology. Since 1987, he has been an instructor in the Industrial Engineering Department at Feng Chia University and served as a secretary in the College of Engineering from 1991 to 1996. Dr. Hung is currently an Associate Professor. His research interests include data mining, CIM, and e-commerce applications. He is a member of the CIIE.