International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.3, No.1, January 2014 DOI:10.15693/ijaist/2014.v3i1.28-35

Trans2Cloud: Cloud Database Converter

Han-Chuan Hsieh, Jiann-Liang Chenand Chien-Hsiang Chen
Department of Electrical Engineering
National Taiwan University of Science and Technology, Taipei 106, Taiwan

***Abstract*—Cloud Computing has attracted much as an effective virtualization technology because of its efficiency for handling massive data and its flexibility for managing Computing and storage resources.A cloud database may be an NoSQL database, which is used by Key-Value tables in a database, or a relational database. This study proposes the use of a HadoopDB cloud relational database system to avoid changing the data structure.This architecture is a hybrid of MapReduce and relational database technologies. The HadoopDB is built in Amazon EC2, and a database converter, called Trans2Cloud, is designed to execute the data migration process. Trans2Cloud is used for distributed the migration and verification data in the Hadoop system. Trans2Cloud also supports any JDBC-compliant (Java Database Connectivity) database server. HadoopDB performs data migration and query analysis in different relational databases. Performance benchmarks are used to compare data migration between the Trans2Cloud and Sqoop systems.**

Index terms -HadoopDB, Cloud Database, Amazon EC2, Cloud Computing.

I. INTRODUCTION

With the rapid growth of users and services of Web applications, increased use of information technology (IT) resources and network bandwidth is inevitable. Rapid accumulation of data over time can result in storage problems, reduced efficiency of query data, and overall deterioration of the application system. Particularly, the many database queries required to execute statements or statistics programs can have a huge impact. The general practice is to establish a database snapshot of the timing synchronization in the original database, not only to provide a database backup, but also so that the statements and statistics program can be used to query the snapshot database without affecting the primary database operations. Although this practice reduces the burden of major database queries, the large accumulation of data over time can degrade system performance. Cloud Computing architectures can reduce the problem of poor system performance caused by information overload. In recent years, Cloud Computing has been extensively studied as a means of optimizing limited Computing resources and storage space. By linking terminal equipment to a network, services can be provided over the network, and users can pay for computing resources and storage space according to their usage. Therefore, renting enterprise cloud system architecture avoids the need to invest in IT equipment and to upgrade the equipment every few years. The main purpose is to enable enterprises to avoid unnecessary investment in IT equipment and services byproviding the flexibility to adjust resources according to demand. Three popular cloud models include software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). The SaaS enables users to access the applications via a network. The PaaS provides users with the resource needed to enable applications. The IaaS is used for processing, storage, networking, and other fundamental Computing resources. The cloud database, which has a pivotal role in the information system database, can be categorized into two classes. One is a non-relational database as proposed by Google BigTable, Facebook, Cassandra, Apache HBase and Amazon SimpleDB [1-4]. These cloud databases, known as NoSQL databases, do not support the SQL syntax of the query, and the data structure is based on key-value store. Another class of cloud databases is a relational database such as Microsoft SQL Azure, Yale University HadoopDB and Amazon RDS (Relational Database Service). Since most Web application systems use a relational database, a relational database system is needed for future data migration to the cloud systems or use operations but is less problematic compared to a NoSQL database system. Additionally, since HadoopDB supports all compatible JDBC traditional relational databases, porting a HadoopDB database to the cloud system is an effective approach.

Cloud Computing is an important trend, and most businesses are expected to convert their original Web application systems to cloud systems. Re-developing a cloud database system is bound to be relatively expensive and time-consuming. Although transplanting original application systems to the cloud system can substantially reduce re-development time and cost, the Web application system migration to the cloud is expected to encounter some problems, e.g., migrating to a different database and ensuring that the content is correct. Therefore, this study proposes Trans2Cloud, a database converter for cloud database computing. The remainder of this paper is organized as follows. Section 2 surveys the related background knowledge on this subject. Section 3 describes the proposed Trans2Cloud system. Section 4 introduces the implementation issues and the results of a performance analysis. Finally, conclusions are given in Section 5.

II. BACKGROUNDKNOWLEDGE

This subsection willintroduce the concepts of proposeddefinition and virtual technologiesof Cloud Computing, and the following subsections describethe studies evaluate the cloud database in terms of system performance query and Apache Sqoop as a migration tool that designed for database efficientlytransferring.

A. Cloud Computing

According to the definition proposed by NIST, Cloud Computing has five essential characteristics, three service models and four deployment types [5].

1. Cloud Computing Essential Characteristics: Cloud Computing has five characteristics: on-demand self-service, broadband network access, resource pooling, rapid elasticity and measured Service.

2. Cloud Computing Service Models: Three popular models include SaaS, PaaS, and IaaS. The SaaS enables users to access applicatoins viaa network. The PaaS all resources that users require to enable applications. The IaaS provides users with fundamental Computing resources, including processing, storage, networking.

3. Cloud Computing Deployment Models: The physical services for a private cloud and community cloud that coexist in the same premises are located at the organization itself or offsite. That is, the physical servers are located at the third party. The four deployment models defined by NIST are private cloud, community cloud, public cloud and hybrid cloud.

B. Cloud Virtualization

This subsection introduces the main virtual technologies of Cloud Computing.

1. Citrix Xen: Xen is an open-source para-virtualized virtual machine monitor (VMM) currently under development by the University of Cambridge Computer Laboratory, UK. Xensource, which was acquired by Citrix on October, 2007, can run an instance of the operating system. Xen, which is a VMM for x86-64, IA-32, ARM and other architectures, allows a guest operating system to execute on the same computer hardware. It also provides VMs with Xen Computing Platform (XCP) software installed for Cloud Computing. Xen can use entities of different specifications of Physical Block Device (PBD) to Virtual Disk Image (VDI) and provides Virtual Block Device (VBD) to the VM as needed.

2. Kernel-based Virtual Machine: The Kernel-based Virtual Machine (KVM) is full virtualization for Linux on x86 hardware, including Intel VT or AMD-V. The KVM enables VMs to run using Linux or Windows disk images. The two execution modes in a standard Linux system are user mode and kernel mode. The KVM includes a guest mode, which executes QEMU-I/O guest code. Kernel mode handles any exit guest mode caused by a normal I/O. Additionally, although the user mode provides a normal I/O for the user, it cannot use special instructions. When the virtual CPU is running, the guest mode is added to the Linux kernel and joins the existing kernel mode and user mode. The VMs are created by opening a device node (/dev/kvm) under KVM. This device node can be used by the user-space library to create and run VMs.

C. Cloud Database

The database for a web application system is very important, even if the cloud system is identical. Therefore, the choice of cloud database systems is also very important. Major service operators use cloud database systems, including Google BigTable [1], Facebook Cassandra, Hive [2,6], and Yahoo PNUTS [7]. In academia, HadoopDB at Yale University includes Hadoop MapReduce in DBMS. Although each cloud database system is unique, their common characteristics are high availability, high fault tolerance, flexible expansion, and capability to run in heterogeneous environments.

Choosing a suitable system is difficult because studies of cloud databases apply different methods of measuring and comparing cloud database performance, including Yahoo Cloud the Serving Benchmark (YCSB) [8] or Benchmarking cloud-based data Management Systems [9]. These studies evaluate the cloud database in terms of system performance query which is considered in addition to Return on Investment (ROI) in a cloud database such as a Service-ROI [10]. However, to transplant the old database system to the cloud database systems,another consideration is compatibility of the data structure, i.e., whether the cloud database can conform to demand, which is an important factor in stored data structures.

1. BigTable: To cope with the huge amounts of information generated by the Internet, Google has developed a BigTable distributed data storage system with capacity reaching the Petabye level. BigTable is used by Google for search engine website indexing and for Google Earth and Google Finance. Because these products must store large amounts of information, both storage capacity and data processing speed are very important. Figure 1 shows the BigTable operations. As the number of servers increases, the number of data read/write cycles increases. When the number of servers reaches 500, the read/write speed reaches 1 million, which enables storage of a huge amount of BigTable data with very good system performance.

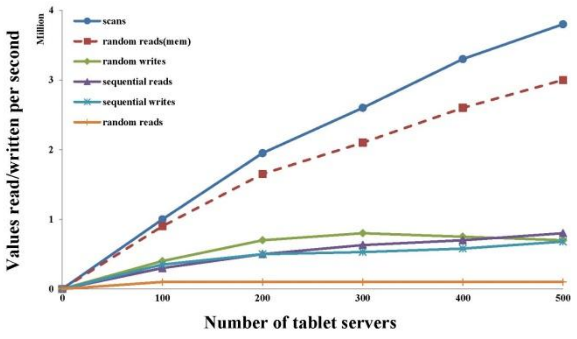


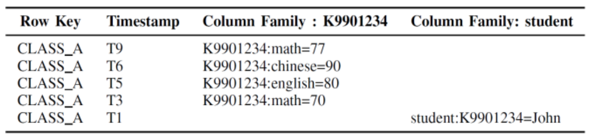
Figure 1. BigTable operation performance

BigTable column-oriented, common relational database (Oracle, MySQL) differs in that its data type is limited to this byte array type. Since the Key-Value map is used to store data, the data storage formats are as follows.

(row; family : column; timestamp) → byte[] data

Row key saving column key is a string. Timestamp is a 64-bit integer used to store each column, which is part of a Column Family Groups. Table 1 shows how another row of timestamps is recorded in the multiple Column Family storage structure.

Table 1. BigTable example: school database



Row Key	Timestamp	Column Family : K9901234	Column Family: student
CLASS_A	T9	K9901234:math=77	
CLASS_A	T6	K9901234:chinese=90	
CLASS_A	T5	K9901234:english=80	
CLASS_A	T3	K9901234:math=70	
CLASS_A	T1		student:K9901234=John

Table 1 shows how the scores for the students are recorded in each class. For example, in the Row key name of the "CLASS_A", a Column Family called "student information" is used to record the class. Additional data are stored in the inside keys Column. The student number corresponding to the student number value is the name of the student. Another Column Family has a name based on the number of students, so, for each student number, there is a group Column Family to record this student subject matter in Table 1 for student number "K9901234". For example, in the records of the results for the three subjects in this Column Family, two math documents are recorded. Therefore, after removing the math results, if did not specify the timestamp will return the last data, so each data version number seen timestamp function.

2. HadoopDB: The HadoopDB [11] framework developed by Yale University combines DBMS and MapReduce technologies to link multiple databases using Hadoop and MapReduce parallel processing data query operation. HadoopDB fault tolerance characteristics are applicable in heterogeneous environments, and, as long as the databases support JDBC link, it can be used as the HadoopDB layer of DBMS. Figure 2 shows the HadoopDB architecture, and the major components are as follows.

   - Database Connector: The database connector, which is a software interface linked to the database, inherits the Hadoop Input Format category. A database connector in each MapReduce queries the SQL and links to the database through the JDBC driver. It can link to any database to support JDBC in a SQL query. The query results are based on key-value type.
   - Catalog: This component, which is the main information in the record database, includes the location information of the database, JDBC driver class, the location of the data backup and information on how to perform the split, etc., the Catalog is the XML file format stored in HDFS (Hadoop Distributed the Filesystem) for JobTracker and TaskTracker applications.
   - Data Loader: The major functions of the Data Loader are loading data and re-cutting data into multiple blocks. Finally, when all blocks are loaded as a single database node, the component consists mainly of the Global Hasher and Local Hasher. Global Hasher with Hadoop obtain information file and bases on the number of nodes split into multiple parts to HDFS. After the Local Hasher copies the file system of each node from HDFS, the file size of each node is sorted by block size and cut into smaller blocks for storage.
   - SMS (SQL to MapReduce to SQL) Planner: The function component is used to convert the SQL syntax to MapReduce job. To perform this task, it parses the input SQL. *Via* Map and Reduce, it then splits the input into multiple MapReduce jobs before performing the task.

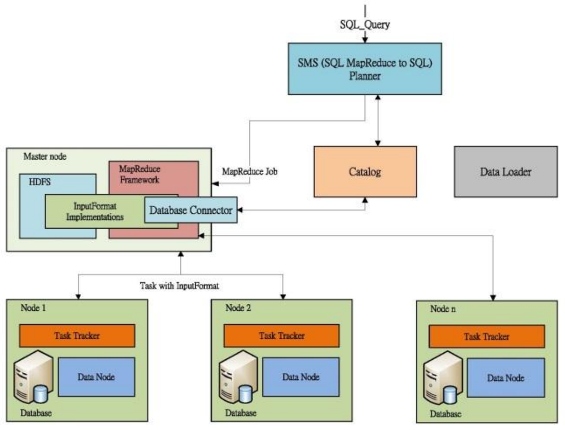


Figure 2.HadoopDB Structure

D. Apache Sqoop System

The main function of Apache Sqoop, which is an open source tool [12], is to provide the relational database and Hadoop HDFS data migration so that users can analyze and manage data by using Hadoop MapReduce [13]. The Sqoop system merges current data into a text file or sequence file format in the default text file.

The Sqoop system is operated through the JDBC database connection. Therefore, before execution, the JDBC driver must put the Sqoop system in the lib data folder. Sqoop uses the MapReduce operation to perform data migration. The default is to perform four Mappers, each of which produces a data file. As additional Mapper data are generated, the data are divided into additional files. Figure 3 shows how the Squoop system imports data to the HDFS architecture.

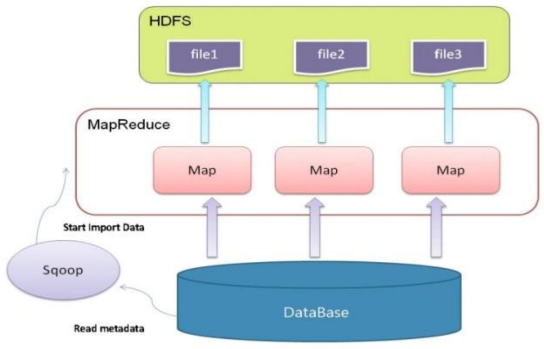


Figure 3. Sqoop Database Converter

III. PROPOSED TRANS2CLOUD SYSTEM

Migration of a heterogeneous database interface program is needed to enable an interface between different database features. In addition to the connection between two different databases, it can also be based on the destination database data type. This chapter describes the research design used in the data migration interface program state to convert the data type of the source of database information. This interface program supports all JDBC-compliant databases, and the Hadoop distributed processing platform performs heterogeneous database data migration. The next section describes the HadoopDB destination database and the operation of the program and the program architecture design of this interface.

A. Trans2Cloud System Operations

Figure 4 shows the five steps of a Trans2Cloud operation.

1. Read Profile: This step obtains the source database, the destination database connection method, and the program execution set. This information is stored in the HadoopDB program memory. Since each data node has a database, HadoopDB.xml profile includes several database connections and data nodes. The dbtrans.properties profile is the connection of the source database, which records tables to be transferred and enables validation of the data set.

2. Export Data: In the read profile steps, the source connection is stored in the program memory. Therefore, this step establishes a connection to the Source and begins exporting table data to HDFS. Each table exports a text file and an export data format u—v field delimiter. Consistent data types are exported to HDFS and cut. Since HadoopDB has a distributed database architecture, each Data Node has its own database server. The cutting is performed to disperse information to each Data Node. Another way to harmonize data is to import data into the destination database and convenient to do data type conversion.

3. Cutting Data: Since profile data are based on pre-existing program memory, this step determines if there is more than one location in the destination database. If the database has multiple locations, this step cuts previously exported text files into multiple text files based on the number of destination database locations.

4. Import Data: This step first processes the text file into the destination database. Based on the number of the destination databases using Hadoop distributed processing capability and the import data, this step converts the data type for each field based on the destination table.

5. Data validation: This step determines whether the destination database has been imported and whether the destination table and the source data table field content. Then the comparisions and statistics will output a file, this file records the number of table of each destination and total items listed in this file, and another file compares the information field and the primary key value. When the data migration is complete, users can view the correct destination database items and content. The assumptions content, based on the primary key value, determines why the error occurred and corrects the error.

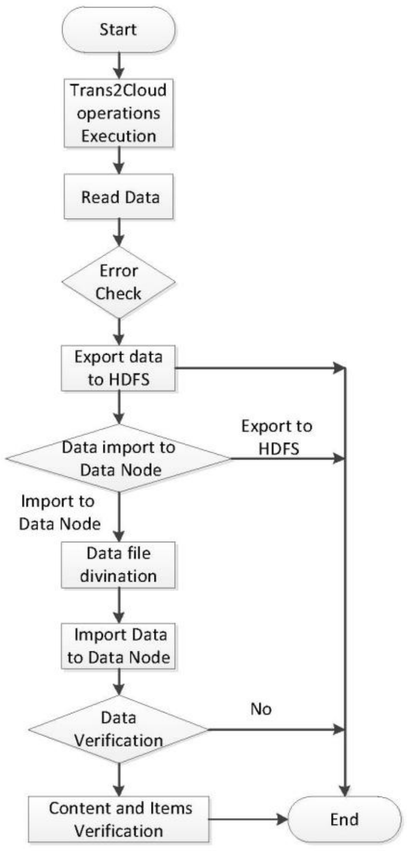


Figure 4. Trans2Cloud System Operations

B. System Design

Figures 5-6 show the class diagram and the sequence diagram for the system design stage. Table 2 also lists the functional modules.

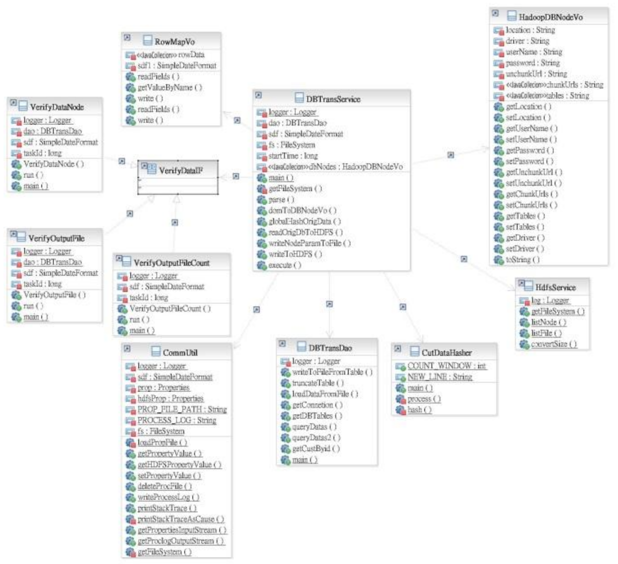


Figure 5. Trans2Cloud Class Diagram

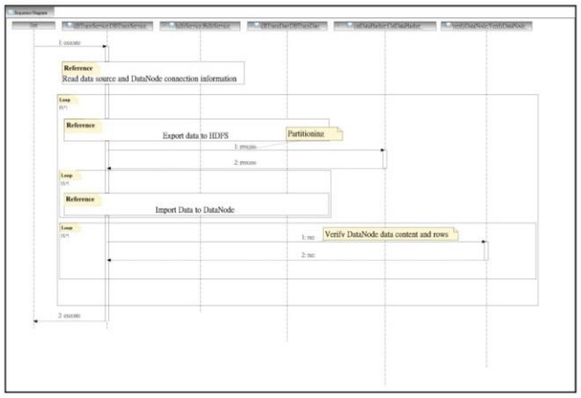


Figure 6. Trans2Cloud Sequence Diagram

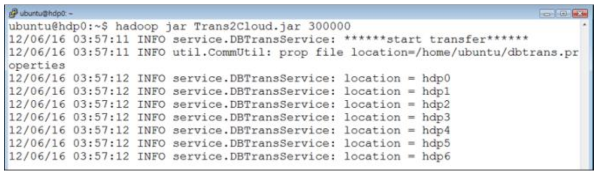
Table 2.Function Modules

Class	Name Description
DBTransService	Main program
DBTransDao	Access database program C
CutDataHasher	Partition program
HadoopDBNodeVo	Store Hadoop Data Node connection information
HdfsService	Access HDFS program
CommonUtil	Common program
RowMapVo	Value Object
VerifyDataIF	Verify data interface
VerifyDataNode	Implement verify data interface(verify data node data)
VerifyOutputFile	Implement verify data interface(verify file data)
VerifyOutputFileCount	Implement verify data interface(verify rows)

C. Trans2Cloud Functional Test

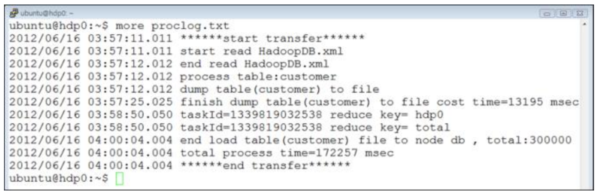
Trans2Cloud system is based on the JAVA programming language. The execution environment is performed with Hadoop versions 0.19-0.20. The execution instructions hadoop jar Trans2Cloud.jar [args1 "[args1] is used to migrate the items.

First, the tables to be migrated to the "products" are set up. The primary key field prod_id A to set validation data parameters "is_verify" Y ", and the data migration operation is performed as shown in Fig. 7. Figure 8 shows the system operation for processing records. Upon completion of the operation, the final output is sent to a file containing the search for import to each data node and for final transfer of items. Suppose that, in the migration process, the source data be altered, resulting in inconsistent data in this file will be the content of the inconsistency displayed.If, during the migration process, an alteration of the source data results in inconsistent data, this file shows the content of the inconsistency.



```
ubuntu@hdp0:~$ hadoop jar Trans2Cloud.jar 300000
12/06/16 03:57:11 INFO service.DBTransService: ******start transfer******
12/06/16 03:57:11 INFO util.CommUtil: prop file location=/home/ubuntu/dbtrans.pr
operties
12/06/16 03:57:12 INFO service.DBTransService: location = hdp0
12/06/16 03:57:12 INFO service.DBTransService: location = hdp1
12/06/16 03:57:12 INFO service.DBTransService: location = hdp2
12/06/16 03:57:12 INFO service.DBTransService: location = hdp3
12/06/16 03:57:12 INFO service.DBTransService: location = hdp4
12/06/16 03:57:12 INFO service.DBTransService: location = hdp5
12/06/16 03:57:12 INFO service.DBTransService: location = hdp6
```

Figure 7. Trans2Cloud Data Migration Operation



```
ubuntu@hdp0:~$ more proclog.txt
2012/06/16 03:57:11.011 ******start transfer******
2012/06/16 03:57:11.011 start read HadoopDB.xml
2012/06/16 03:57:12.012 end read HadoopDB.xml
2012/06/16 03:57:12.012 process table:customer
2012/06/16 03:57:12.012 dump table(customer) to file
2012/06/16 03:57:25.025 finish dump table(customer) to file cost time=13195 msec
2012/06/16 03:58:50.050 taskId=1339819032538 reduce key= hdp0
2012/06/16 03:58:50.050 taskId=1339819032538 reduce key= total
2012/06/16 04:00:04.004 end load table(customer) file to node db , total:300000
2012/06/16 04:00:04.004 total process time=172257 msec
2012/06/16 04:00:04.004 ******end transfer******
ubuntu@hdp0:~$
```

Figure 8. Trans2Cloud Process Records

IV. Heterogeneous Database Migration and Performance Analisis

A. Data Migration

The feasibility of the proposed Trans2Cloud system is evaluated in three cases:

1. Performance analysis of HadoopDB in different data migration
2. Performance analysis of HadoopDB in original database query
3. Performance analysis of Trans2Cloud Sqoop data migration

In a well-developed e-commerce business, accurate customer and order information are essential. Therefore, migration of data in a customer order database is often necessary. The database contains four tables: customer information sheet, product data sheets, the orders table and the order details table. Figure 9 shows the relationships among the tables. Table 3 shows the test data.

Table 3. Test Data

Data type	Data number
Customer data	1,060,300
Products data	1,395,244
Orders data	1,049,998
Order products data	2,100,354

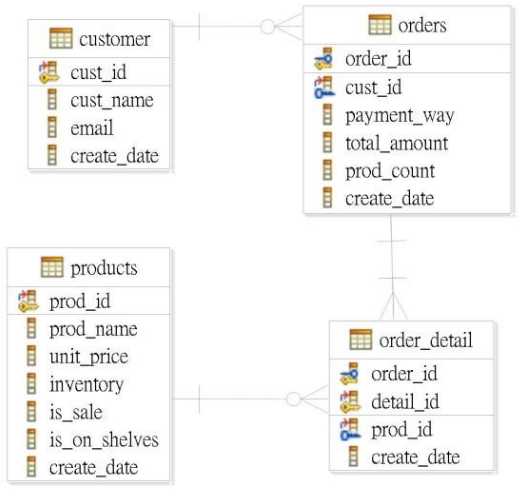
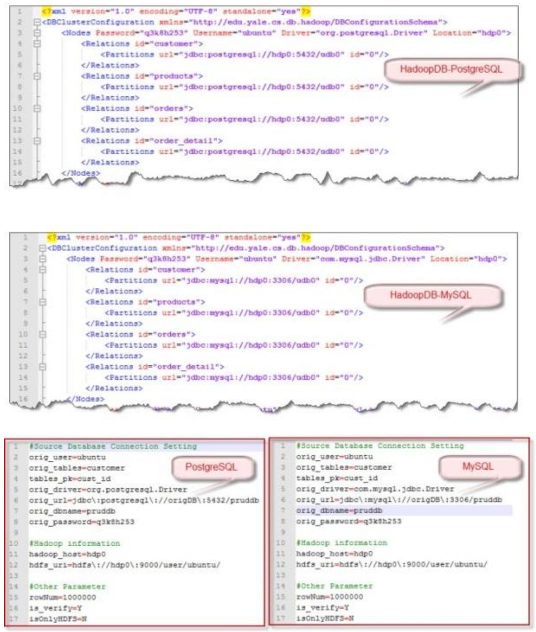


Figure 9. Test Cases

As described earlier in the Source Data Node database PostgreSQL and MySQL database performance analysis, the "dbtrans.properties" and "HadoopDB.xml" profiles are used to establish two sets of profiles are shown in Fig. 10 for different combinations of experiments.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DBClusterConfiguration xmlns="http://edu.yale.cs.db.hadoop/DBConfigurationSchema">
    <Nodes Password="q3k8h253" Username="ubuntu" Driver="org.postgresql.Driver" Location="hdp0">
        <Relations id="customer">
            <Partitions url="jdbc:postgresql://hdp0:5432/udb0" id="0"/>
        </Relations>
        <Relations id="products">
            <Partitions url="jdbc:postgresql://hdp0:5432/udb0" id="0"/>
        </Relations>
        <Relations id="orders">
            <Partitions url="jdbc:postgresql://hdp0:5432/udb0" id="0"/>
        </Relations>
        <Relations id="order_detail">
            <Partitions url="jdbc:postgresql://hdp0:5432/udb0" id="0"/>
        </Relations>
    </Nodes>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DBClusterConfiguration xmlns="http://edu.yale.cs.db.hadoop/DBConfigurationSchema">
    <Nodes Password="q3k8h253" Username="ubuntu" Driver="com.mysql.jdbc.Driver" Location="hdp0">
        <Relations id="customer">
            <Partitions url="jdbc:mysql://hdp0:3306/udb0" id="0"/>
        </Relations>
        <Relations id="products">
            <Partitions url="jdbc:mysql://hdp0:3306/udb0" id="0"/>
        </Relations>
        <Relations id="orders">
            <Partitions url="jdbc:mysql://hdp0:3306/udb0" id="0"/>
        </Relations>
        <Relations id="order_detail">
            <Partitions url="jdbc:mysql://hdp0:3306/udb0" id="0"/>
        </Relations>
    </Nodes>
```

```
#Source Database Connection Setting
orig_user=ubuntu
orig_tables=customer
tables_pk=cust_id
orig_driver=org.postgresql.Driver
orig_url=jdbc\:postgresql\://origDB\:5432/pruddb
orig_dbname=pruddb
orig_password=q3k8h253

#Hadoop information
hadoop_host=hdp0
hdfs_uri=hdfs\://hdp0\:9000/user/ubuntu/

#Other Parameter
rowNum=1000000
is_verify=Y
isOnlyHDFS=N
```

```
#Source Database Connection Setting
orig_user=ubuntu
orig_tables=customer
tables_pk=cust_id
orig_driver=com.mysql.jdbc.Driver
orig_url=jdbc\:mysql\://origDB\:3306/pruddb
orig_dbname=pruddb
orig_password=q3k8h253

#Hadoop information
hadoop_host=hdp0
hdfs_uri=hdfs\://hdp0\:9000/user/ubuntu/

#Other Parameter
rowNum=1000000
is_verify=Y
isOnlyHDFS=N
```

Figure 10. Database Setting

B. *Performance Analysis*

This section describes the research and analysis of data migration to HadoopDB, and each step is described in detail below.

1. Analysis of export performance data to the HDFS, Trans2Cloud, and Sqoop.
2. Analysis of transfer performance data source to invalidated data content in HadoopDB.
3. Analysis of transfer performance data source to validated data content in HadoopDB.

In each step, the obtained data are recorded as three times the average amount. Figure 11 shows the first step, which is exporting different data sources to HDFS for performance comparison. Table 4 shows the data rows. Figure 12 shows the step for Trans2Cloud and Sqoop data transfer performance. Although Sqoop also uses the Hadoop platform for distributed processing, the Trans2Cloud outperforms Sqoop when only one Mapper is set.

The second stage analyzes performance in transferring the data source to invalidated data in HadoopDB. Figure 13 shows the Data Node of HadoopDB with a different database for analysis. The results show that PostgreSQL is slower than MySQL. Therefore, MySQL is superior to PostgreSQL for data import and for frequent writing.

The third stage analyzes performance in transferring a data source to validated data in HadoopDB. The data source is identical to that in the second stage, and HadoopDB Data Node is used with a different database for analysis as shown in Fig. 14. PostgreSQL is slower than MySQL, but the gap is smaller than that for invalidated data. The PostgreSQL outperforms MySQL in validating data for queries and in making frequent comparisons, so the data node with PostgreSQL is superior for frequent queries.

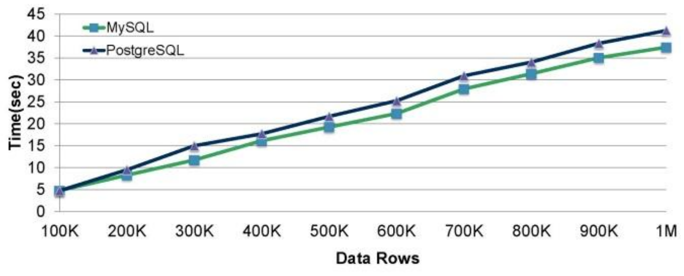


Figure 11. Different Source DB to HDFS

Table 4. Different Source DB to HDFS Data Records

	Rows									
Data Source	100K	200K	300K	400K	500K	600K	700K	800K	900K	1000K
PostgreSQL	4(sec)	9(sec)	14(sec)	17(sec)	21(sec)	25(sec)	30(sec)	34(sec)	38(sec)	41(sec)
MySQL	4(sec)	8(sec)	11(sec)	16(sec)	19(sec)	22(sec)	27(sec)	31(sec)	35(sec)	37(sec)

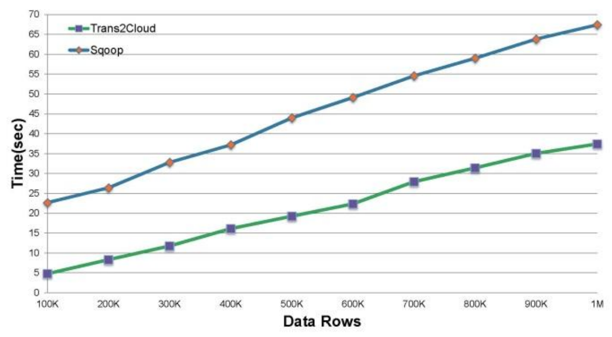


Figure 12. Trans2Cloud and Sqoop Performance Comparison

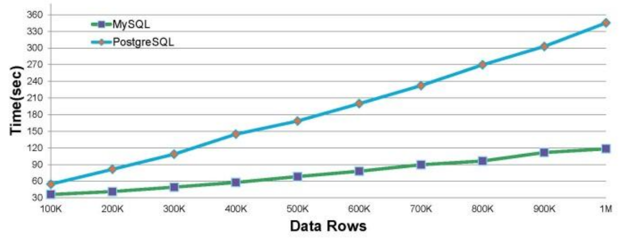


Figure 13. Identical DB to Data Node (Invalidated Data)

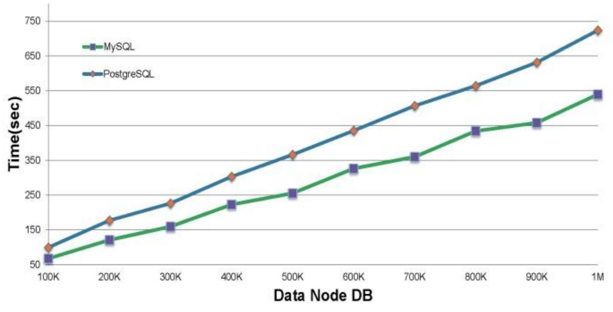


Figure 14. Identical Source DB to Data Node (Validated Data)

C. *HadoopDB query Performance Analysis*

After data migration to HadoopDB, this section describes the results of a HadoopDB query performance comparison between an aggregate query and a JOIN query. Each query is described below.

1. Aggregate query: The query format is as follows.

selectsum(total_amount), cust_id
fromordersgroupbycust_id;

Since this query contains a "Group By" syntax, performing a group aggregation requires this syntax to query the orders table, i.e., the total amount consumed by each customer. Comparisons of aggregate queries (Fig. 15) show that the query speed of HadoopDB with PostgreSQL is better than that with MySQL. However, when the amount of data is less than 500,000, the query speed of the original database is better than that of HadoopDB. Until the amount of data exceeds 500,000, HadoopDB is clearly better than the original database.

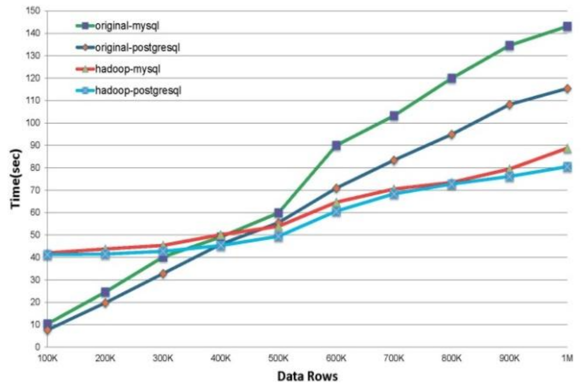


Figure 15. Aggregation Query Performance

2. JOIN query: The query format is as follows.

selectsum(a.total_amount), a.cust_id, b.cust_name; b.emailfromordersajoincustomerbona.cust_id = b.cust_idgroupby
a.cust_id, b.cust name,b.email;

Since the query syntax reduces effectiveness, the query time is longer than that for the previous syntax. The syntax for the orders table and customers table is used to conduct a joint inquiry, and the output contains the results of the total amount of customer information and consumer. The JOIN query analysis in Fig. 16 shows that the query is more time-consuming than the previous one, but the results are similar when the amount of data exceed 500,000. The HadoopDB query performance is only slightly better than that of the original database, which indicates that, when the amount of data is not very large, HadoopDB does not improve performance. The relatively poor performance of HadoopDB may explain why it allocates work to the Data Node in each query.

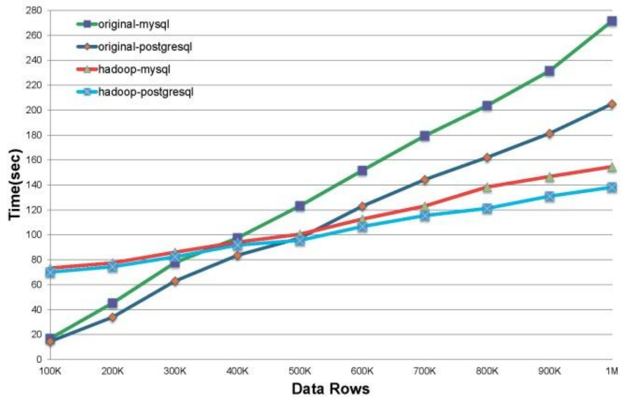


Figure 16. JOIN Query Performance

V. CONCLUSION

This study proposes a Trans2Cloud system for converting a database for use in Cloud Computing. The relational database was migrated to HadoopDB in five steps. This Trans2Cloud system can also verify the results of the entire data migration. The experiments to test the feasibility and the performance of the proposed Trans2Cloud system showed that HadoopDB with PostgreSQL performs the Join and aggregate queries more efficiently compared to HadoopDB with MySQL. However, data migration issues arise because MySQL is faster than PostgreSQL. Another observation was that, when the number of data is lower than 500,000, the default HadoopDB query is faster than HadoopDB with MySQL and PostgreSQL. When the number of data items reaches 1 million, HadoopDB is faster than the original database. Therefore, when the amount of data is large, the database is suitable for use as a HadoopDB cloud database but also reduces processing performance. In addition to open source cloud databases, cloud database systems have been developed by many manufacturers, including Oracle, IBM, Mircrosoft and other manufacturers. These manufacturers have also launched cloud versions of their database systems. Therefore, the performance of future Trans2Cloud Data Migration systems is expected to be improved by the following capabilities automatic table creation, migration to different cloud database systems, correction of incorrect information, and adjustment of the number of suitable Mappers.

ACKNOWLEDGMENTS

The authors would like to thank the National Science Council of the Republic of China, Taiwan for financially/partially supporting this research under Contract No. NSC 100-2218-E-011 -019 -MY3 and NSC 100-2218-E-011 -018 -MY3.

REFERENCES

[1] 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 Proc. of the 7th Conference on USENIX Symposium on Operating Systems Design and Implementation, pp.205-218, 2006.
[2] Lakshman and P. Malik,"Cassandra: a decentralized structured storage system," ACM SIGOPS Operating Systems Review, pp.35-40, April 2010.
[3] E. Sciore,"SimpleDB: a simple java-based multiuser system for teaching database internals," In Proc. of the 28th SIGCSE Technical Symposium on Computer Science Education, pp.561-565, 2007.
[4] Abouzeid, K. BajdaPawlikowski, D. Abadi, A. Silberschatz and A. Rasin,"HadoopDB: an architectural hybrid of MapReduce and DBMS technologies for analytical workloads," VLDB Endowment, pp.922-933, Aug. 2009.
[5] P. Mell and T. Grance,"The NIST Definition of Cloud Computing," NIST Sep. 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf
[6] 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 Endow., pp.1626-1629, Aug. 2009.
[7] 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 Endow., pp.1277-1288, Aug. 2008.
[8] B. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan and R. Sears,"Benchmarking Cloud Serving Systems with YCSB," In Proc. of the 1st ACM symposium on Cloud Computing, pp.143-154, June 2010.
[9] Y. Shi, X. Meng, J. Zhao, X. Hu, B. Liu and H. Wang,"Benchmarking Cloud-based Data Management Systems," In Proc. of the 2nd international workshop on Cloud data management, pp.47-54, Oct. 2010.
[10] V. Mateljan, D. Cisic and D. Ogrizovic,"Cloud Database as a Service (DaaS)-ROI," In Proc. of the 33rd MIPRO international convention, pp.1185-1188, May 2010.
[11] Abouzied, K. Bajda-Pawlikowski, J. Huang, D. J. Abadi and A. Silberschatz,"HadoopDB in action: building real world applications," In Proc. of the SIGMOD International Conference on Management of Data, pp.1111-1114, June 2010.
[12] D.P. Pham, C.F. Lin and E. Jou,"Database backed by cloud store for on-premise applications," In Proc. of the IEEE 13th International Conference on High Performance Computing and Communications, pp.708-713, Sep. 2011.
[13] M. J. Hsieh, C. R. Chang, L. Y. Ho, J. J. Wu, P. Liu,"SQLMR : A Scalable Database Management System for Cloud Computing," In Proc. of IEEE International Conference on Parallel Processing, pp.315-324, Sep. 2011.

Authors Profile

Han-Chuan Hsieh was received a **B.S.** degree in Electrical Engineering from National Taipei University of Technology (NTUT) in 1998, and an **M.S.** degree in Communication Engineering from Tatung Institute of Technology, Taipei, Taiwan in2008. He is currently a Ph.D. candidate in Department of Electrical Engineering of National Taiwan University of Science and Technology (NTUST). His major interests are in **4G/B4G** Network, Software-Defined Networking and Internet of Things.

Jiann-Liang Chen was born in Taiwan on December 15, 1963. He received the **Ph.D.** degree in Electrical Engineering from National Taiwan University (NTU), Taipei, Taiwan in 1989. Since August 2008, he has been with the Department of Electrical Engineering ofNational Taiwan University of Science and Technology(NTUST), where he is a professor now. His current research interests are directed at cellular mobility management and personal communication systems.

Chien-Hsiang Chen received the **M.S.** degree in Electrical Engineering from the National Taiwan University of Science and Technology (NTUST), Taipei, in 2012. He is currently with the Chunghwa Telecom Inc. His research interests include parallel and distributed system and Cloud Computing.