國立臺灣大學資訊學院電機工程學系碩士班

碩士論文

Graduate Institute of Electrical Engineering College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

雲端計算平台於資料庫上的應用

Cloud Computing Platform for Applications on Database

陳敬翔

CHEN, JING-SIANG

指導教授：郭斯彥 博士

Advisor: Sy-Yen Kuo, Ph.D.

中華民國 100 年 6 月

June, 2011

國立臺灣大學碩士學位論文
口試委員會審定書

雲端計算平台於資料庫上的應用

Cloud Computing Platform for Applications on Database

本論文係陳敬翔君（學號 R98921026）在國立臺灣大學電機工程學系完成之碩士學位論文，於民國 100 年 06 月 25 日承下列考試委員審查通過及口試及格，特此證明。

口試委員：郭斯彥（簽名）
（指導教授）

雷欽隆　陳英一

陳俊良　劉宗榮

系主任　顏嗣鈞（簽名）

誌謝

本篇論文能順利圓滿完成，首先要感謝指導教授郭斯彥教授這兩年來的指導與教誨，並提供了完善的研究環境、資源讓學生能夠完成這兩年碩士班的學業和論文。

起初在進入碩士班開始參與的國科會「物件互聯網之 RFID 關鍵技術及系統研發」計劃前，我對 RFID 的概念和技術非常的陌生，但經由博士班張騰文學長非常細心且不厭其煩的教導、陪伴下，讓我逐漸了解了 RFID 中介軟體層的重要性以及 RFID 技術可以在不同面向上應用之情境，也因此在 2009 年暑期代表實驗室參與中國上海復旦大學的 RFID CJK 研討會議。雖然事後張騰文學長因其他原故必須休學離開，但是他卻成為我在碩士生涯中最佳的典範，亦開拓了我在 RFID 中所應用的情境感知有了紮實的了解。

接著在我邁入碩士二年級時參與了另一項台大資電中心的「雲端運算於老人照護之應用整合」計劃，並且轉由紀廷運學長帶領，開始進入下一個階段的學習旅程。紀廷運學長是位非常認真而且也願意一個步驟一個步驟地教導你邁向計劃正確的方向。然而在參與計劃的這段期間，紀廷運學長循序漸進的讓我了解日本東京大學所發展的 Live E!計劃以及資料庫的相關應用技術，同時我也逐漸知道傳統資料庫的限制與雲端可以給予我們在限制上的突破，配合了我在先前在一年級所學習到的 RFID 情境感知知識，和學長一同順利研製出一套屬於情境感知所使用的雲端資料庫。

在漫長的參與計劃和研究過程中，靠著非常多人的幫忙和協助才得以順利完成計劃和研究。在這裡要特別感謝胡永立學長、陳立勝學長、鄒耀東學長、劉晉廷學弟、莊育瑄學妹、王凡老師實驗室的李淑芬同學；胡永立學長、陳立勝學長與鄒耀東學長在我撰寫論文階段，給予我極大的幫助以及提供了許多方向與建議，每當遇到問題時總是熱心的提供解決方針並分享寶貴的經驗，同時也提供了許多重要的資料與文獻；劉晉廷學弟、莊育瑄學妹與李淑芬同學在我進行論文最後衝

刺的階段，給予我很多很多很重要的鼓勵與支持，也常常在他們最繁忙的時候給予我最熱切的協助。真的非常謝謝你們，沒有你們的幫助也不會有這篇論文的產出！

再來還要感謝在我在台科大就學時期所參與的第四屆微軟學生大使的企業實習裡，一直支持我到現在的林昱佑、林子昂、蔡鎮宇、戚守為、詹欣芸、姜沛晴、以及其他第三屆到第五屆的各位。因為有著一同奮鬥、完成各種挑戰的經歷，所以我相信有著無人可擋的情誼存在著。在這段期間也謝謝你們包容我的任性，以及我所提出的各種無理的要求，同時也適時的讓我了解到我在各個方面還可以更加的邁進與堅持。我相信我們的友誼將會長久存在！

接著感謝在實驗室裡一同奮鬥的陳世軒、陳沛汝、李庭宇、林春薰與顏新晨同學，還有陳義雄、金迺安、張耕力、周怡廷、高宗永、吳長勳、黃聿凱學弟，總是在我失意的時候帶給我歡樂，讓枯燥的研究生活充滿歡笑。以及雖然畢業了但依然真誠的給我提點的梁容豪、翁榮鴻、王維成、王睿、黃敏純、尤佳良、張天寶學長。還有博士班的陳威良、陳彥旭、簡嘉宏、游家牧、費爾德、王富平、劉頂立、卓啟翔、梁毓珊、何智祥、謝博全、謝長泰、田謹維、張廷勛、梁漢文、林義堯、陳永維、陳凱學長姊們，你們總能解答我心中的疑惑，讓我的學習百尺竿頭更進一步！

當然，沒有父母在背後無怨無悔的支持和栽培，是不可能有今日的我。有著父母提供的良好、衣食無缺的成長環境，才能讓我有機會接受完整的教育和獲得正面的人格發展。在這邊要將此篇論文獻給我最愛的家人，和給予我支持的朋友們。

陳敬翔 于國立臺灣大學電機工程學研究所

中華民國一百年七月

中文摘要

情境感知（Context-Aware）是現代計算機科學領域主要研究的議題。由於嵌入式計算技術不斷的進步，小型的感測儀器已逐漸深入了生活之中。但是，這些大量且連續產生的感測資料卻漸漸的無法被現今常見的關聯式資料庫（RDBMS，Relational Database Management System）所承受。置於雲端運算平台的資料庫系統其可擴充性雖然可以解決此一問題，但是雲端運算（Cloud Computing）平台分散式檔案系統（DFS，Distributed File System）的特性，使得使用者務必要重新學習新的資料庫管理方法。

本論文提出了一項置於雲端運算平台的關聯式資料庫管理系統，讓使用者可以透過 JDBC 驅動程式與 SQL 結構化查詢語言來操作資料庫。同時可以妥善利用雲端運算平台所提供的優勢，資料庫將會有更佳的彈性與可靠程度。

關鍵字：情境感知、雲端運算、Hadoop、HDFS、Map／Reduce、關聯式資料庫、JDBC、CloudBase、SQL。

ABSTRACT

Context-Aware is one of the main focuses in the research field of modern computer science. While embedded computing technology had improved, micro-sensor devices have gradually gained their presence in our lives. However, the commonly seen RDBMS (Relational Database Management System) nowadays cannot handle the large quantities of continuously generated sensor data. The extendibility of databases on Cloud Computing platform Hadoop can solve this problem, but the feature of Cloud Computing, DFS (Distributed File System), requires the users to learn new ways of managing databases.

This thesis not only presents a RDBMS on Cloud Computing platform which shows the user how to manage the database with JDBC driver and SQL (Structured Query Language), but also explains how to use the advantages of Cloud Computing platform to improve the elasticity and reliability of database.

Keywords: Context-Aware, Cloud Computing, Hadoop, HDFS, Map/Reduce, RDBMS, JDBC, CloudBase, SQL.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

1.1 Research Background And Motivation

Context-Aware is one of the major focuses in the research field of modern computer science. While embedded computing technology had improved, micro-sensor devices such as RFID or Sensor Network, have gradually gained their presence in our lives. Information about objects or the environment gathered by the sensor instruments offers all kinds of follow-up services, such as digital family, mobile services, etc. However, the continuously generated, large amounts of sensor data need to be stored and managed in real time. The task is very crucial.

The application of database is indispensable for modern information industry. It separates enterprise material and application program so that the application program can access data from the same sources. Besides, DBMS (Database Management System) provides various services for users to manage the database. Among them, RDBMS is one of the most widely used database systems. It manages the data with mathematics concepts of Set and Algebra. We can do simple operations on the database through SQL provided by RDBMS. Users do not even need to know how data are stored.

The RDBMS databases which are OSS (Open-source software) such as MySQL, PostgreSQL, etc. have the advantages of high reliability, high security, high availability, and low cost. Therefore, the database systems have been widely used in middle or small scale websites. As the mobile internet devices, Wire/Wireless Sensor Network and IPv6

(Internet Protocol version 6) become more and more popular these days, we need larger database to store the ever-increasing amounts of data. Not only do we need large storage space, we also hope that data server can withstand large data flow when saving the data. The databases will require enough reliability and security as well. Although we can always choose a larger or more competent database server to solve the storage space and loading problems, the time and energy used on transferring data and the procurement cost will not be cost effective.

Cloud computing is a growing trend in information technology. It comes from distributed computing and grid computing. The main purpose of Cloud Computing is to make use of separated data resources on the Internet effectively, to increase the efficiency of large data operation, which is often very complex, and to shorten the computing response time [1]. With flexible, diverse, and virtualized basic structure, Cloud Computing offers demand-oriented and dynamic allocated computing resources. In addition, the supplying process is completely automated. Not only does it meet the security requirements, it also comes up with the best way of allocating resources according to different amount of data being processed, so that large amounts of complex data can be computed instantaneously and accurately. Therefore, Cloud Computing will become an effective solution to accessing and operating large amounts of data.

The original purpose of “Live E!” (IEEE 1888 Standard) was to solve modern environmental problems. For example, to set up information gathering platforms about global warming or environmental conservation, and to share the environmental data collected from mini-weather stations in each region through the Internet. However, the improved version of “Live E!” can be extended and become a platform used for

collecting Context-Aware data. It combines TCP/IP (Transmission Control Protocol/Internet Protocol) communication agreement, and all data from the sensor devices can be gathered by “Live E!” platform. In the end the data will be stored and compiled into XML (eXtensible Markup Language) files, which can used on application services later.

1.2 Research Intention

Although the current “Live E!” platform focuses on data collection and information dissemination, its fundamental storage system is still PostgreSQL database, and it is not highly scalable. Therefore, it is not fully enough to store the large amounts of context-aware data. If it integrates with a highly scalable Cloud Computing platform, the problem may be solved, but Cloud Computing platform has limitations from its distributed file systems, so it is not possible to directly integrate with PostgreSQL relational database.

The purpose of this study is to rebuild a highly scalable data storage system based on SQL structured query language syntax and Cloud Computing platform. Therefore, the flexibility and reliability of the databases can be improved. It allows users, information analysts or back-end application developers to easily and effectively store huge amounts of context-aware data through Cloud Computing and its storage technology.

Chapter 2 Related Works

2.1 “Live E!” Data Collection Platform

The original purpose of setting up “Live E!” data collection platform was to solve modern environmental problems and serve as a weather information collection platform for global warming and environmental conservation. “Live E!” integrates micro-weather stations from different regions, and constructs a clustering weather information collection and sharing system through the Internet. Due to the fact that low-cost micro-weather stations can be densely distributed in cities, high-level weather surveillance can be implemented.

After “Live E!” was improved, it can be expanded into a context-aware information platform. Through TCP/IP agreement, all data gathered by the sensors will go to “Live E!” platform. “Live E!” not only has sensory messages with high intensity and high accuracy, it also compiles all gathered data into the commonly used XML format to provide application services for future use.

There are three main interfaces of “Live E!” data collection platform that researchers and developers can use: Data Retrieval Interface, Profile Registration Interface and Data Upload Interface. Through the three interfaces, it enables data collection and information distribution. The structure is as shown in Figure 1.

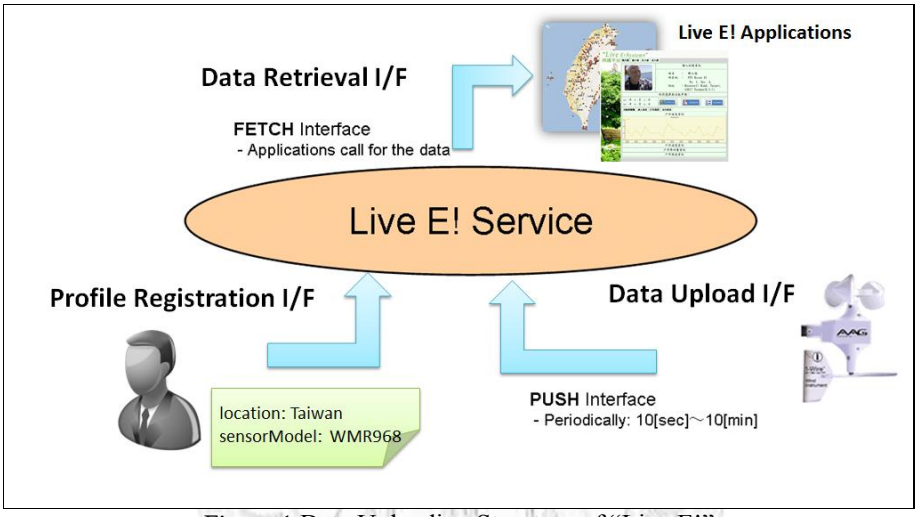

Figure 1 Data Uploading Structure of “Live E!”

Firstly, “Live E!” will change the upload installation into a legitimate data provider through Profile Registration Interface. Then Data Upload Interface will upload the instantaneous sensing data, such as that of nursing care, medical personnel, context-aware information and environmental conditions to “Live E!” platform. Lastly, the application services will retrieve integrated XML files from “Live E!” and offers important services by means of analyzing or mining.

The detailed structure of “Live E!” is as shown in Figure 2. “Live E!” uses PostgreSQL relational database to store the Profile information, which is defined by the user, and the uploaded data from the Context-Aware environments. However, as the Internet getting more and more popular, the data we hope to upload to the “Live E!” Data Upload Interface is getting larger and larger. Not only do we need larger storage space, we also hope that “Live E!” services can withstand large data flow when database is being accessed. Meanwhile, the data stored in the database should also be secure and reliable.

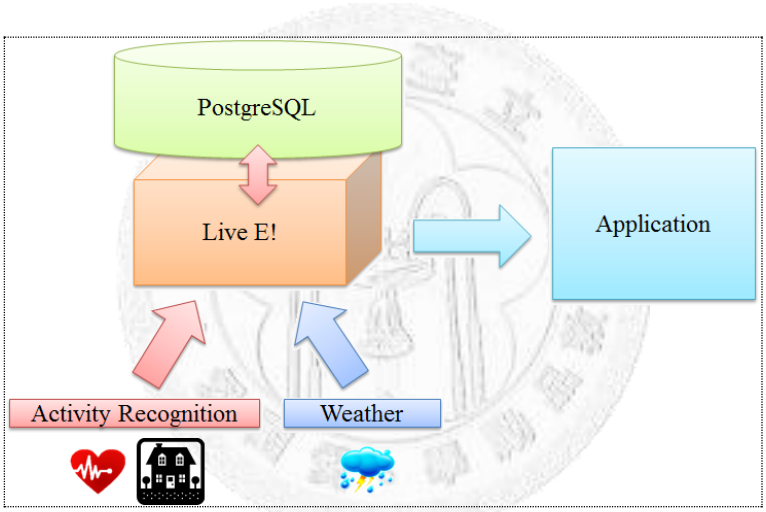

Figure 2 “Live E!” Information Exchange Structure

Take health care for the elderly as an example; “Live E!” can be used to collect physiological and environmental information from the elderly, and then create a medical healthcare platform which has two interfaces: one for users, the other for medical personnel. In the User Interface, the elderly can upload physiological information such as blood pressure and heartbeat every day and check the latest living conditions and statistics about his activities. The graphic is as shown in Figure 3.

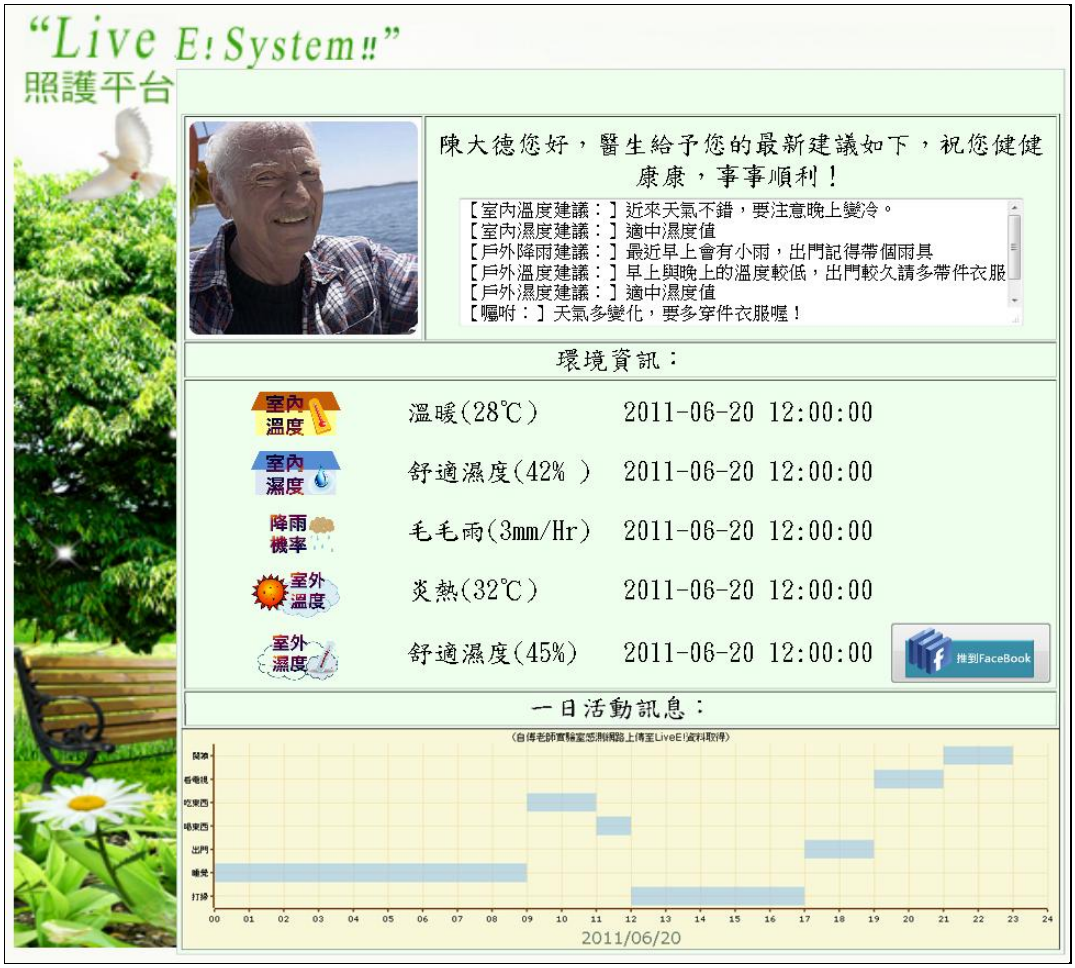

Figure 3 Healthcare Platform --User Interface

Then, the healthcare platform offers a simple sharing function. Users can upload their latest conditions to social network site Facebook with just one click. It helps friends and families to familiarize with the health conditions of elderly people and enhances his level of social attention. As it is shown in Figure 4, the platform analyzes data obtained from “Live E!”, reconstructs them into colloquial descriptions and imports them to social network site Facebook, all within one click.

Figure 4 Status Sharing Function of Social Network Facebook

In addition, Medical Personnel Interface will provide more detailed information about the elderly, such as environmental conditions both indoors and outdoors, daily lives of the elderly and different physiological conditions, etc. Medical personnel can therefore provide long-distance healthcare assistance. Figure 5 shows the interface of how medical personnel can give the elderly useful advices according to their condition.

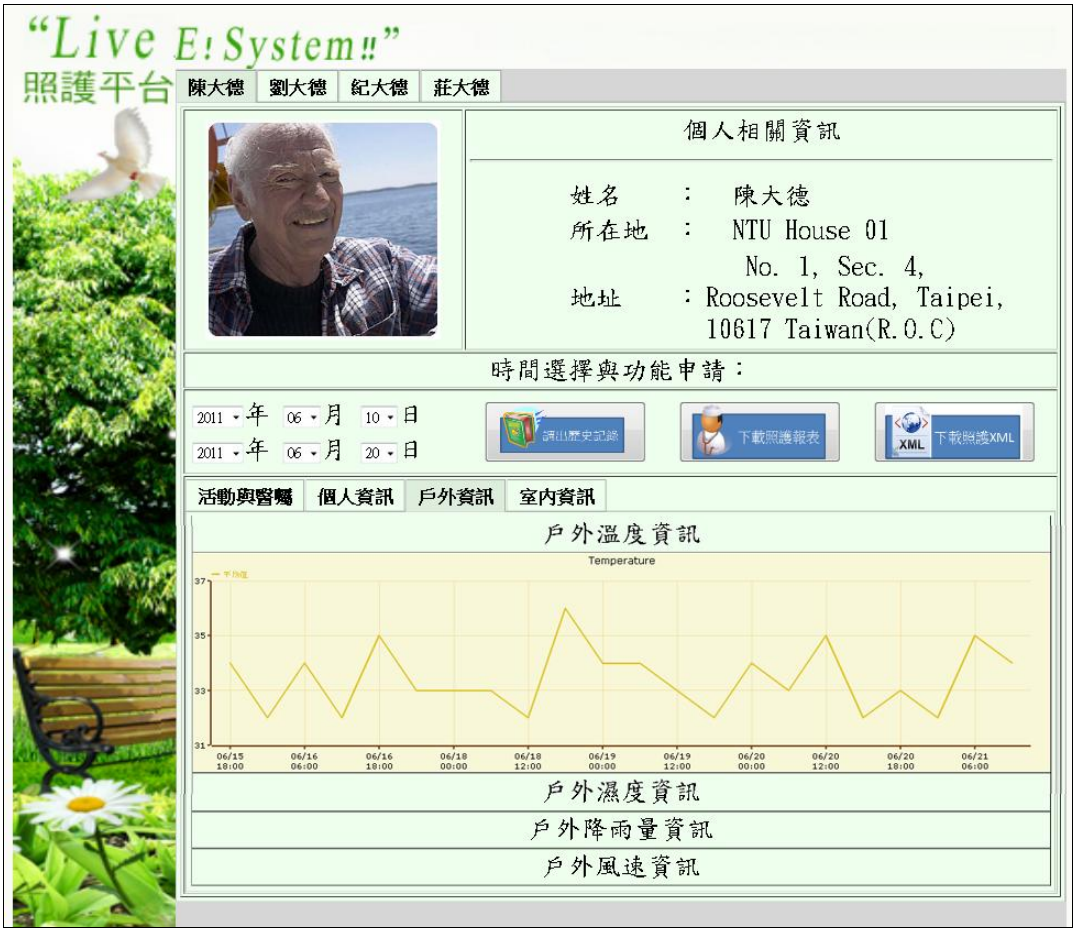

Figure 5 Healthcare Platform --Medical Personnel Interface

Meanwhile, in order for the medical personnel to have a deeper understanding of the long-time physiological conditions about the elderly or environmental changes, and to provide deeper diagnosis and assessment, the healthcare platform also provides healthcare reports for them to download. Figure 6 explains that medical personnel only need to choose the time intervals for the required data and the information within that interval will be exported into a diagram for medical use.

Figure 6 Healthcare reports provided by the Healthcare Platform

Lastly, if any other application services need to access information about the elderly, programmers can download XML files from the healthcare platform and accomplish further service development after the administrator has authorized the approval. Figure 7 illustrates that extended services can supplement current insufficient healthcare platform and provide more complete support in the fields of data mining, etc. and find out the relationship between certain diseases, weather or daily activities.

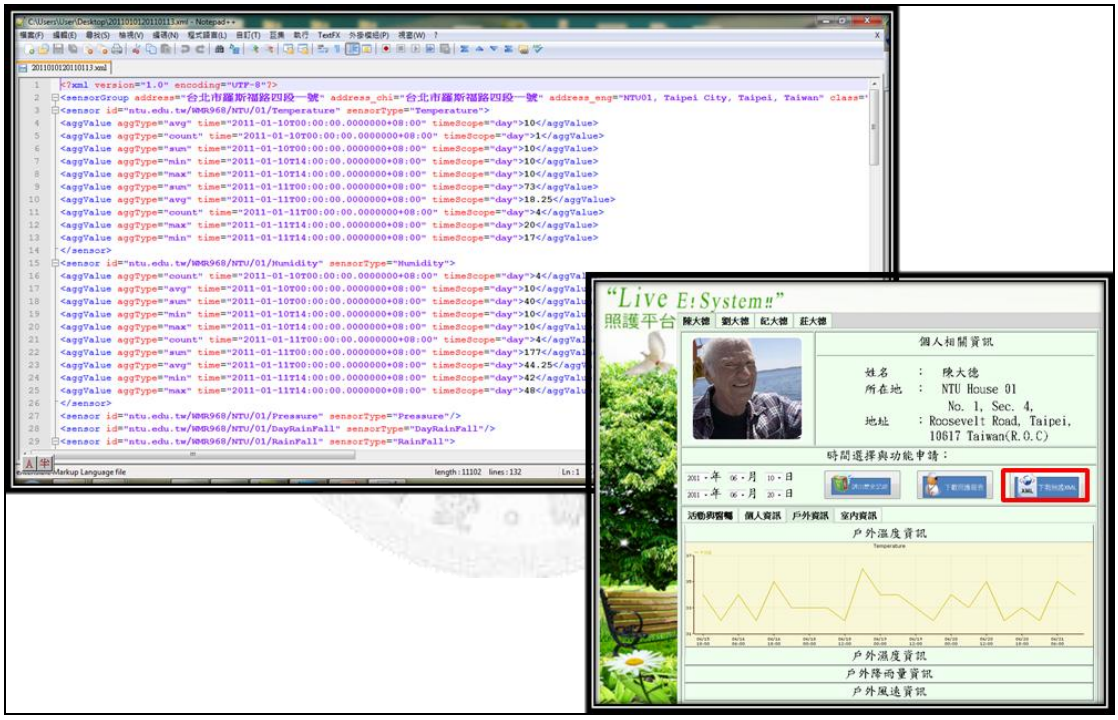

Figure 7 XML files provided by the Healthcare Platform

2.2 Cloud Computing Platform

2.2.1 Cloud Computing

Essentially, Cloud Computing is a new application of Distributed Computing. The basic concept is that a large process is automatically divided into numerous sub-processes through the Internet. A large system consisted of several multi-servers will send back the results to end-users after searching and computational analysis. Generally speaking, Cloud Computing can be divided into three layers [1]:

A. Infrastructure as a Service (IaaS): By implementing usage-based billing, IaaS provides resource services of hardware, software and other devices to software applications. IaaS can dynamically expand or decrease the available range according to requirements of applications, and also make full use of the resources and avoid waste. There are two typical cases: Amazon EC2 (Elastic Cloud Computing) Service and S3 (Simple Storage Service). Both cases have public schema of billing operation and storage function. In addition, Eucalyptus is also a kind of Cloud Computing which offers interfaces compatible to Amazon EC2. Users can set up a Cloud Computing environment with business transaction functions based on their individual business models.

B. Platform as a Service (PaaS): Platform as a Service (PaaS) offers a highly integrated environment which can be used to set up, test and distribute user applications. Developers usually have to work on scalable, built-in software. For example, Google App Engine allows users to establish their web application

system in powerful scalable Google App systems.

C. Software as a Service (SaaS): According to usage-based billing model, users can access specific software system through the Internet. For example, Force.com from Salesforce is an online Customer Relationship Management (CRM) service, and LiveMesh of Microsoft shares files and folders through multiple devices simultaneously. Although Cloud Computing provides three kinds of services as listed above, there is still room for improvement regarding the standards of interfaces. Due to this current situation, several problems occur when different Cloud Computing systems connect with one another. Therefore, Cloud suppliers need to put in additional investments on research, development and manufacturing of interfaces. With the development of different Cloud Computing systems, interactions among them have become more and more frequent; a unified standard will therefore be required in the future.

By applying Cloud Computing, associated applications are no longer limited to personal computers or regional internet servers. They can be stored in servers accessed through the Internet. Even when regional computational node or servers are crashed, normal users can still access software application system through the Internet. In addition, through flexible, diverse, and virtualized basic structure, Cloud Computing can provide demand-oriented and dynamic allocated operation resources for different workload. The fully automated supplying procedure is secure; it can also be adjusted to meet users' needs automatically. The advantages are as follows:

A. High computational and storage capability: Users can use computational services at any time or place. Cloud Computing services have high-volume data storage capacity which users can use to process large amounts of data.

B. Flexible configuration: Users can order services, applications and resources according to their personal requirements without having to deal with the details of software and hardware.

C. High Cost Point value: Users only require low-cost hardware to utilize Cloud Computing services and operations. It reduces the maintenance cost and improves the utilization ratio. Moreover, Cloud Computing has better computational ability than normal large host computers.

D. High reliability and security: With data being processed and backup in the Cloud, which is controlled strictly by specialized technicians, the execution environment for applications has high stability and fault tolerance.

E. Dynamic scalability: It can dynamically add nodes for servers and computational resources, and enhances computational performance and the process of fault tolerance.

Cloud computing has high computational capabilities. The most influential factor is the technique of resource allocation management of Distributed Computing in low level system architecture, such as job scheduling and load balancing. The scheduling algorithm of Hadoop consists of three phases: Hadoop On Demand (HOD), Capacity Scheduler and Fair Scheduler. HOD is managed by Torque/Maui of Linux. Capacity

Scheduler is proposed by Yahoo! for Multiple Queue processing. Fair Scheduler is presented by Facebook with the focus on balancing the resources of Small Job. Different tasks should have different processing methods. Only with proper measures, resources of the whole structure can be fully utilized and reach its best performance.

2.2.2 MapReduce

Currently the MapReduce structure from Google is the most highly-valued structure in terms of the distributed programming model in Cloud Computing [2]. MapReduce simplifies the distributed programming model and enhances the response speed of distribution and integration in distributed programming. The design model for MapReduce is to separate the programs into map functions and reduce functions. As stated below:

$$\text{Map}(K_1, V_1) \rightarrow \text{List}(K_2, V_2)$$

$$\text{Reduce}\big(K_2, List(V_2)\big) \rightarrow \text{List}(V_2)$$

Firstly, through map functions, the data will be broken into irrelevant blocks and sent to the servers in distributed environment to do parallel computing. A map function will processes a key/value pair to generate a set of intermediate key/value pairs. Then a reduce function will combine the ones with the same intermediate key with all the other relevant intermediate values, and produce the output Key/Value pairs, which will be the computing result. Therefore, after map function and reduce function, the unprocessed data will become precise and accurate. Figure 8 explains how MapReduce structure works in a distributed environment. MapReduce programs will be handled by Master and Worker. Master will assign map and reduce functions to different Workers. First the data will be separated into blocks and delivered to the Worker for map functions, and to be processed and stored there. The Workers in charge of reduce functions then access the output of the map functions remotely, and after running the reduce functions, the

final result will be compiled and sequenced.

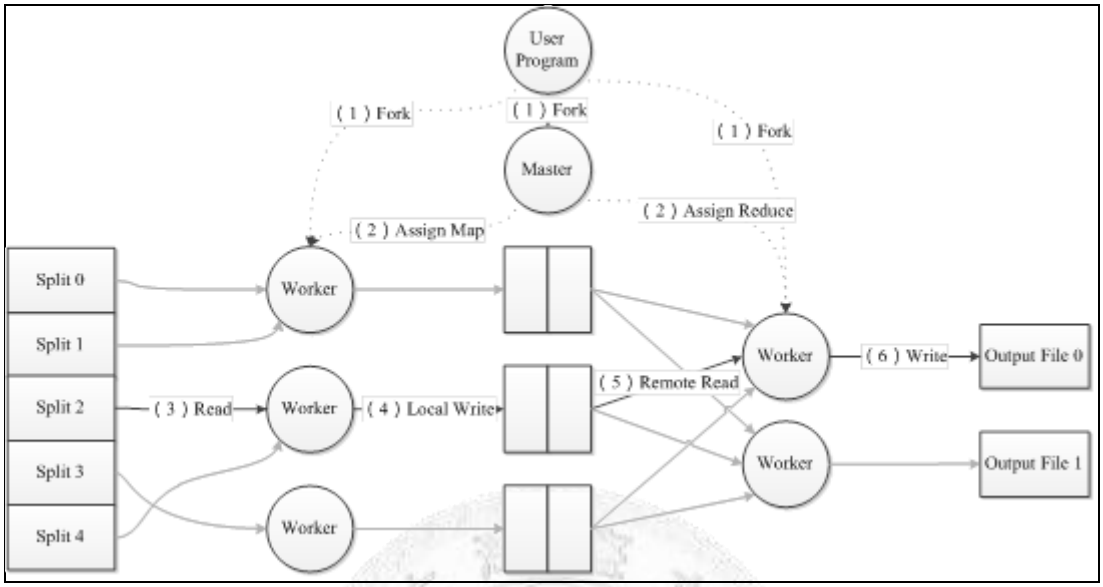

Figure 8 MapReduce Structure（Source From [2]）

2.2.3 Google File System

As for distributed resource and file management, Google has also offered a file system structure, called Google File System [3]. It has high fault tolerance, and can be used in low cost hardware devices and to handle the reading and writing for large amounts of data, as it is shown in Figure 9. Google File System is a Master/Slave structure. The master servers are used as Namenode, which maintain and manage the operation of the file system, such as creating or deleting files. Other slave servers used to store the file blocks and record the metadata blocks, are used as Datanode. The communication protocols of Namenode, Datanode and client are layered on top of the TCP/IP protocol. When a client writes in the data, the local host will cache them first. As the cached data arrive at the predetermined blocks, the client will inform a Namenode. The Namenode will then decide the appropriate storage blocks in Datanodes

and send information of the location and the corresponding blocks back to the client; the client can therefore write and read the data effectively.

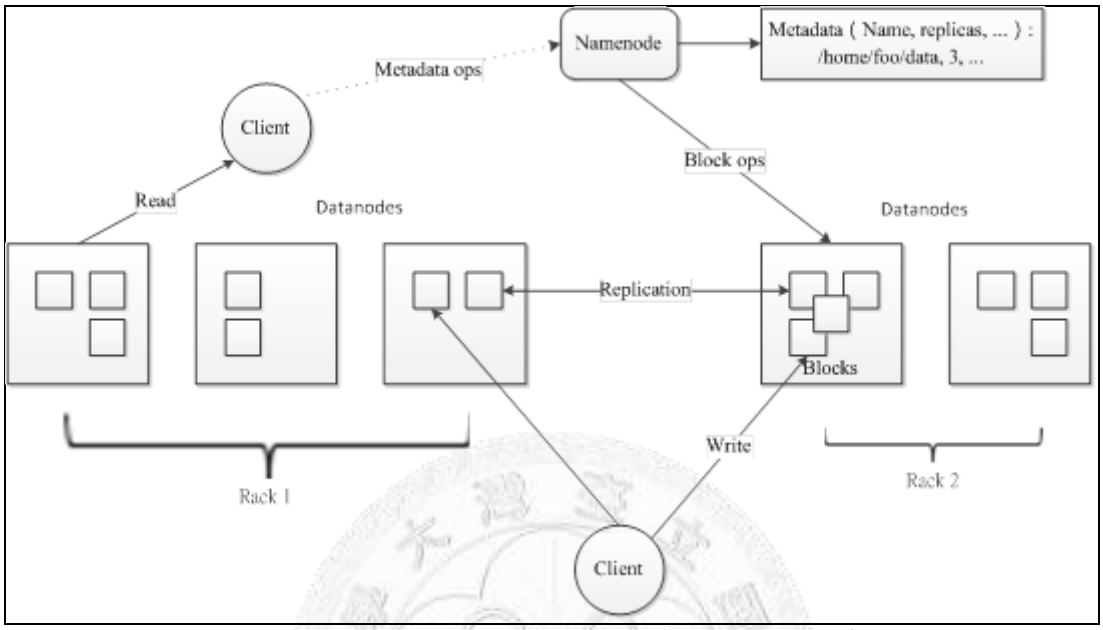

Figure 9 Google File System Structure（Source From [2]）

Google File System has many characteristics that are suitable for processing large amounts of data. Google File System uses streaming data access, and it is Write-Once-Read-Many. The files cannot be changed once they are set up. Besides, the files are stored in blocks and in distributed storage system, and it performs offsite backup; therefore the accuracy of the data can be assured. Because Google File System has taken the location of the data into account, the process of data computing can be performed at the location of the data. The efficiency is much higher than moving data to the computational nodes, and it also improves the service quality of data computing as a whole.

2.3 Cloud Database

2.3.1 Hbase

HBase is the data storage system on Hadoop platform. Bigtable, a distributed and structuralized data storage system presented by Google [4], is an effective mechanism which stores large amounts of data in a distributed environment. It is object-oriented and operates on HDFS (Hadoop Distributed File System). In HBase, the structure of data storage is similar to tables. The data is organized into multidimensional tables in rows and columns. It can be extended easily and also very practical. The tables in HBase can be divided into several parts, stored or deployed to different computing nodes, which are formed by several blocks and files in HDFS. Relational Database is good for data changes, such as to create, to update or to delete, etc. These operations are performed mainly in memories. As for analyzing large amounts of data, when the data scatter in many nodes, relational database is not suitable for use. Therefore, unlike classical relational database, HBase in many ways simplifies the operations that consume memory resources. For example, there is only one index for all the tables; there is no “join” function, and it does not support SQL syntax and only deals with byte arrays, so that it can perform instant access to large amounts of distributed data.

Because of the limits from HBase, traditional database developers face many disadvantages, such as they require extra training to understand different table structures, and that is why there are other Cloud databases which support SQL structures, such as Hive [5] and CloudBase [6].

2.3.2 Hive

Hive, initially developed by Facebook and later given to Apache as a subproject of Hadoop, is built on top of Hadoop. In Hive, tables set up by the users will be organized into files and stored in HDFS. Through the tools provided by Hive, you can easily analyze large amounts of database files in HDFS. Hive uses HQL (Hive Query Language), a structuralized query language similar to SQL, so the users familiar with SQL can search the data in Hive, too. The search queries from the users will be broken down into Map/Reduce tasks and the results will be replied quickly. However, the database is built within HDFS, so it has to comply with the restriction of "Write-Once-Read-Many," which means none of the files can be altered once they are created, so HQL does not support uploading, inserting and deleting of any files. The system structure of Hive is shown in Figure 10:

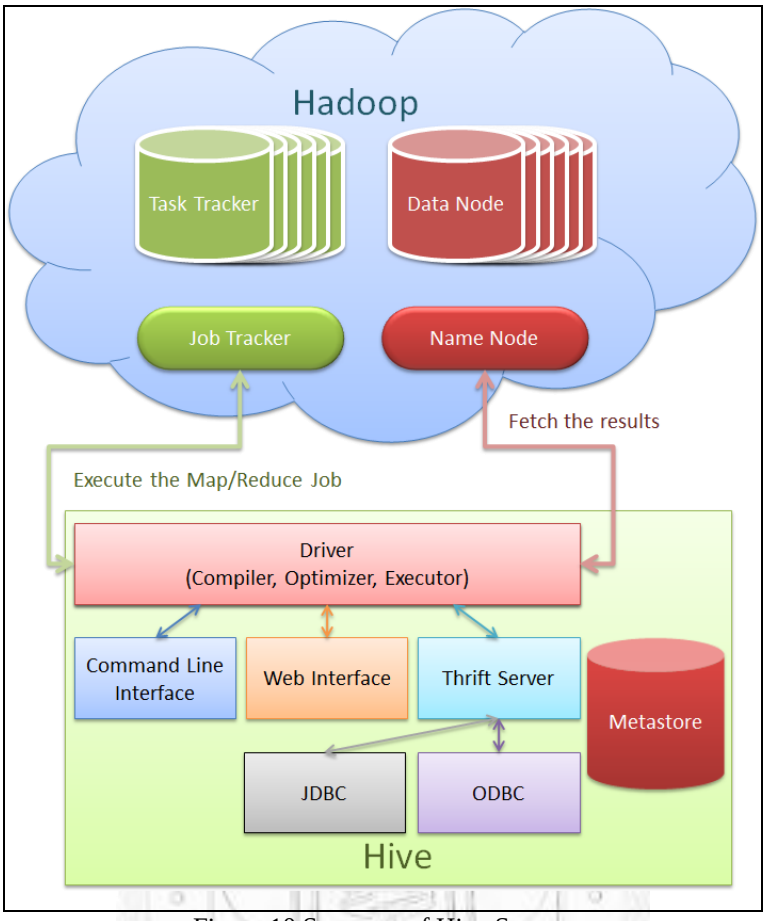

Figure 10 Structure of Hive System

2.3.3 CloudBase

CloudBase is created and maintained by Tarandeep Singh, Prathibha Deshikachar and some others from Business.com. It aims to help the industry use SQL and through JDBC driver to search the Terabyte-level data in distributed storage systems. Just like Hive, CloudBase is also built on Hadoop, but only lighter and easier to use.

2.3.4 Comparison

CloudBase and Hive both provide interfaces similar to SQL, which enables the SQL users to operate the database easily, and because the database search queries will be transformed into Map/Reduce processes, they can be completed quickly. However, both systems are confined to HDFS and can only search the data within the database; thus the main point of this thesis is how to redesign the database and add the functions of inserting, updating and deleting data.

Among all the Hadoop databases supporting SQL, CloudBase supports the widest range of database query languages. It is easy to install and does not take up much space. To use CloudBase, you only need to set up the routes and execute, and because it contains JDBC drivers, it allows all the JDBC's managing application programs, such as Squirrel, to connect with it, and enables the end users to connect and access the data in CloudBase more easily.

By contrast, Hive provides many additional tools, and it is more complicated and inconvenient to build. Users need to manually create certain space and different tools specifically for Hive in HDFS. For example, extra downloading tools need to be implemented to install web interfaces or JDBC drivers. If you simply want to use JDBC and do not want a too-complicated database with SQL interface, CloudBase should be an easy solution.

The comparison chart for HBase, Hive and CloudBase is shown in Table 1, and this thesis will focus on SQL relational database system in CloudBase database and how it can be improved in terms of data storage.

Table 1 Comparison of HBase, Hive and CloudBase

	File size	Levels of Support for SQL	Barriers for SQL users	Database Extension	Table Extension	SQL integration level
HBase	large （around 40MB）	none	high	high	high	low
Hive	large （around 50MB）	medium	low	high	none	medium
CloudBase	small （around 4MB）	high	low	high	none	high

Chapter 3 Middleware Design on Cloud Database

There are many restrictions on efficiency and reliability when large volumes of data are uploaded to a single database node. If "Live E!" changes in to the architecture of Cloud database as shown in Figure 2, the level of scalability can be improved. Furthermore, all the data which are stored on the Hadoop cluster can be scheduled to backup on different data nodes. Therefore, the reliability of data stored on Cloud can increase dramatically.

In this thesis, we design a middleware to improve the ability of "Live E!". In this middleware as shown in Figure 11, system can use the access function which is offered by HDFS to implement simple access methods. Furthermore, this middleware can extend the functions of CloudBase and redefine the basic operations of database: INSERT, UPDATE, and DELETE. Because HDFS has the feature (or limitation) of "write-once-read-many", the direct method of processing the data is to read data from HDFS and store them in the memory, and then execute the operations of INSERT, UPDATE, and DELETE. Finally, the processed data will be written to the HDFS once. The access functions of HDFS are as follows:

$$\text{Memory} \leftarrow \text{Read File From HDFS}(\textit{Source File Path})$$

$$\text{HDFS} \leftarrow \text{Write File From Memory}(\text{Processed Files}, \text{Destination File Path})$$

As it is illustrated in Figure 11, when application executes a query to middleware (Tag 1), it will connect to the CloudBase actively (Tag 2). Then the middleware will

also execute the query request from application on CloudBase. CloudBase would analyze and dispatch the Map and Reduce tasks to Hadoop cluster (Tag 3). After finishing the querying operation of MapReduce tasks, the results will be sent back to the application.

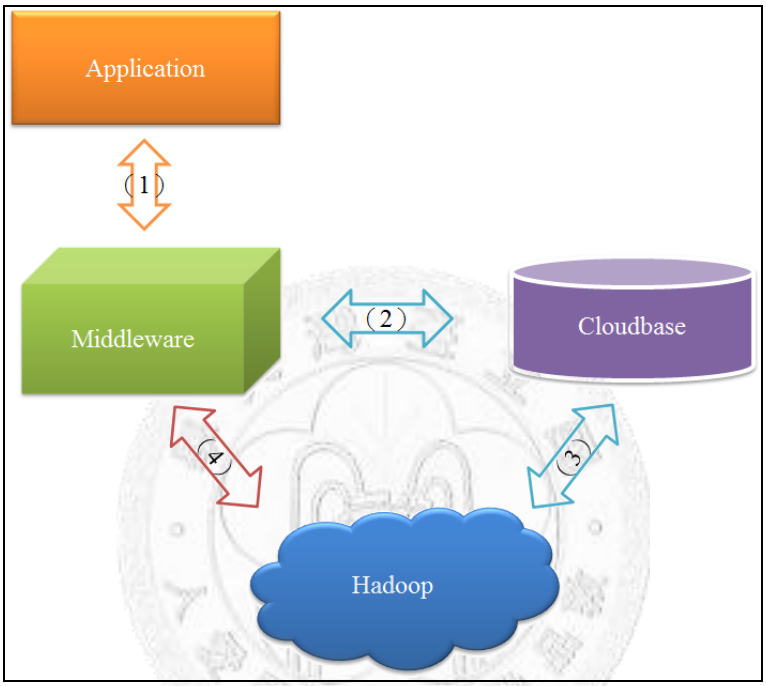

Figure 11 Methods to implement modules

When application executes update query, depending on the types of requests, the responses can be classified into INSERT, UPDATE, and DELETE.

If the request is INSERT, middleware can connect to Hadoop and access the data table from HDFS, then store the data in the memory of middleware system and adjust the data in the memory (Tag 4). Finally, the insertion completes and the data table will overwrite the original files in HDFS. The flow of insertion operation is illustrated in Figure 12.

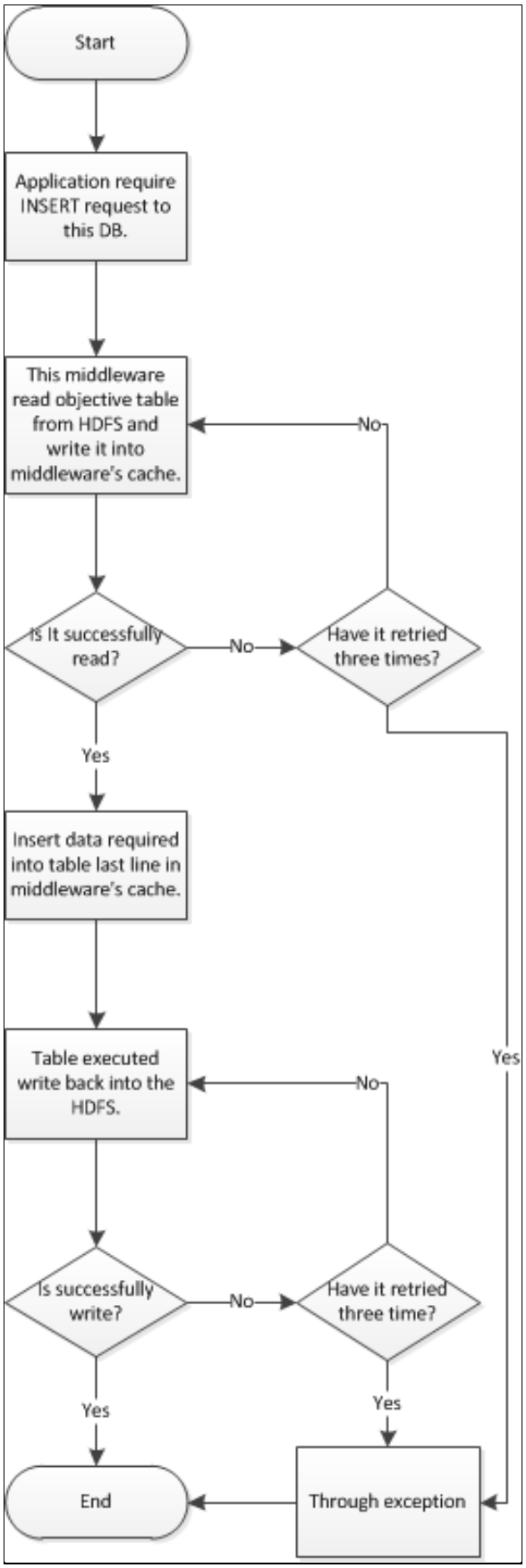

Figure 12 Middleware INSERT process

If the request is UPDATE, middleware has to query data table on CloudBase first (Tag 2). CloudBase can dispatch tasks to Hadoop and send back the results. The results will be stored in the memory of middleware system (Tag 4). Then middleware would access data tables from the HDFS. After wards, middleware can compare them with the data stored in the memory and update corresponding rows on the same table. Finally, the update operation completes and the new table overwrites the old one on HDFS. The flow of UPDATE operation is illustrated in Figure 13.

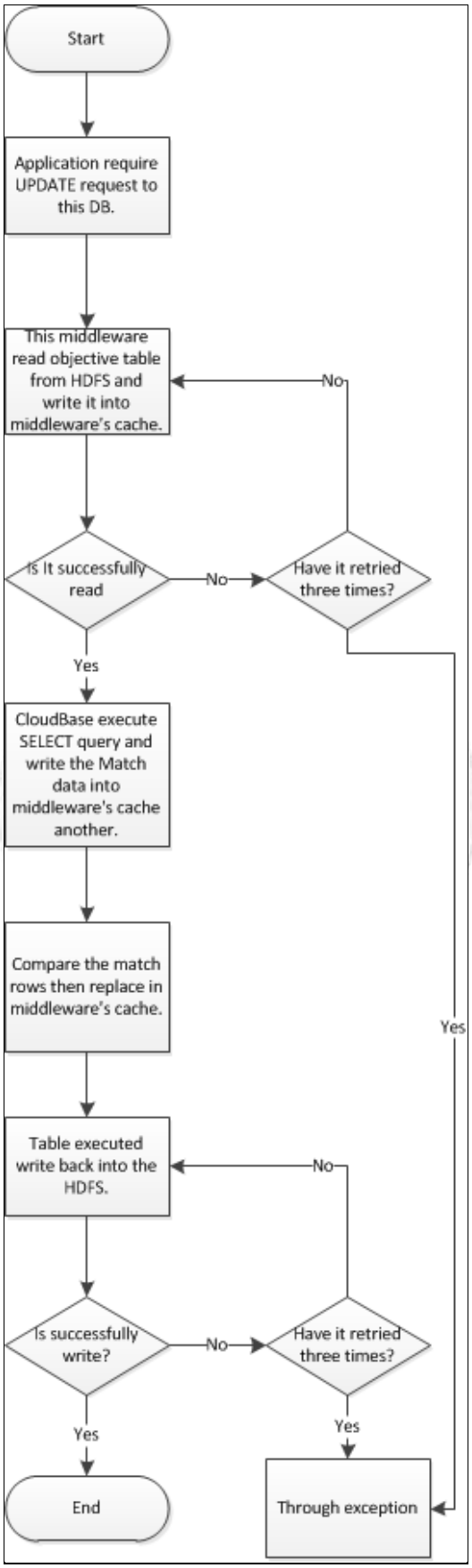

Figure 13 Middleware UPDATE process

If the request is DELETE, middleware has to query data table on CloudBase and store it the memory of middleware system first (Tag 2). Then middleware will access data tables from HDFS (Tag 4). After accessing the data, middleware will compare the data stored in the memory and delete the same rows. Finally, the update completes and the new data table will overwrite the original on HDFS. The flow of deletion operation is illustrated in Figure 14.

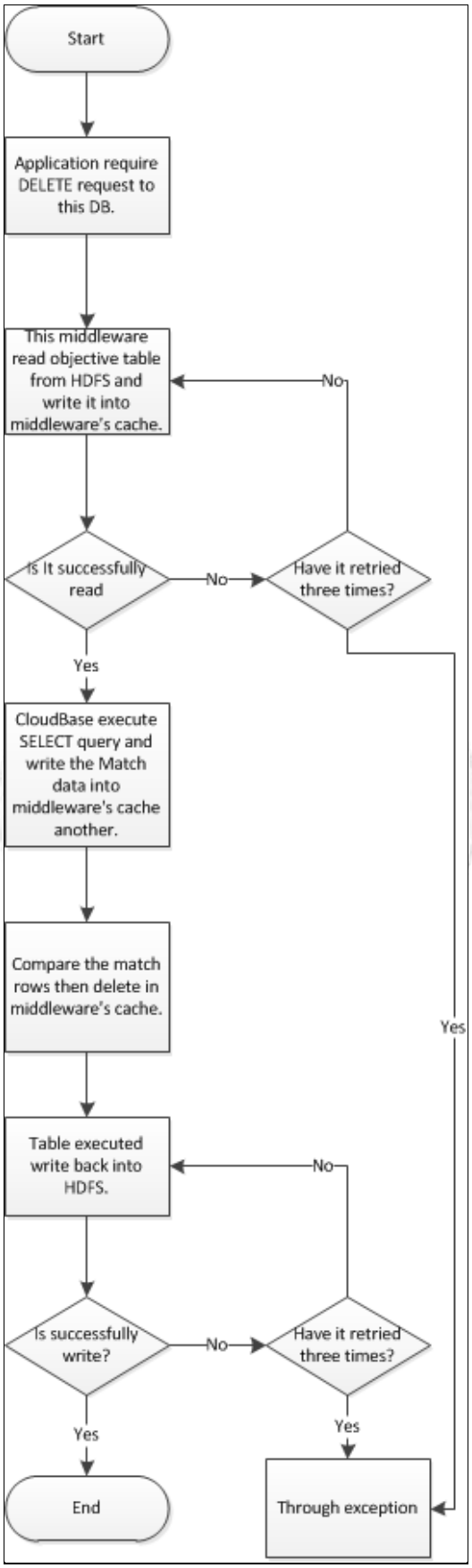

Figure 14 Middleware DELETE process

Chapter 4 Results and Performance Evaluation

4.1 Results

The middleware provided in this thesis is designed to enhance the capacities for systems like “Live E!” when accessing databases, and replace them with Cloud-based ones. The main purpose of this chapter is to verify the expanded functions that middleware offers to the CloudBase, and test it with the INSERT, UPDATE and DELETE database commands. The results listed below are done with test_table1 as shown in Figure 15, where the INSERT, UPDATE and DELETE operations are executed in sequence.

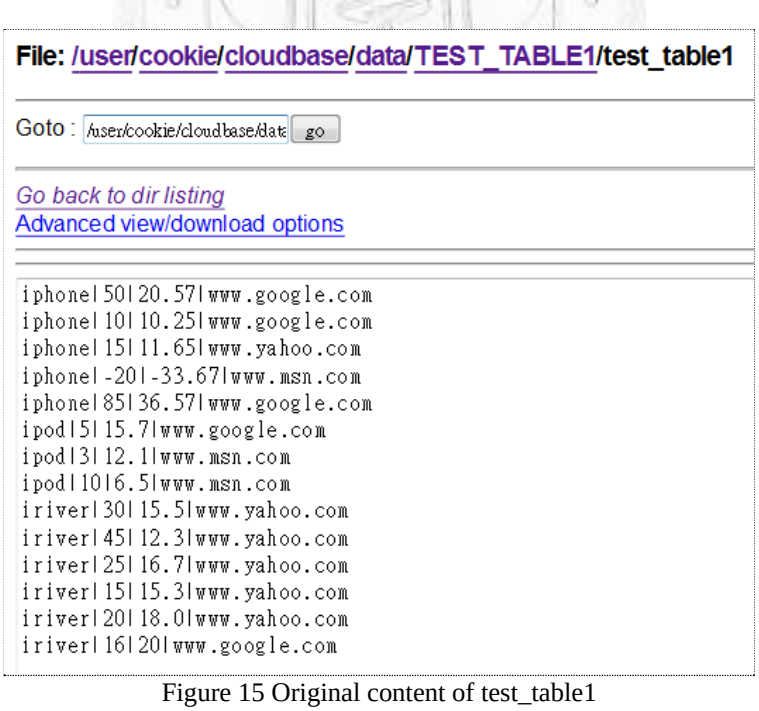

Figure 15 Original content of test_table1

Users can use middleware to execute the INSERT operation, as the query shown below and the execution screen in Figure 16:

```
INSERT INTO test_table1(c1, c2, c3, c4)
VALUES('zune', 100, 55.55, 'www.zune.com');
```

Once the operation is completed, data will be added to the last row of test_table1, which is stored in the HDFS. Execution result is shown below in Figure 17.

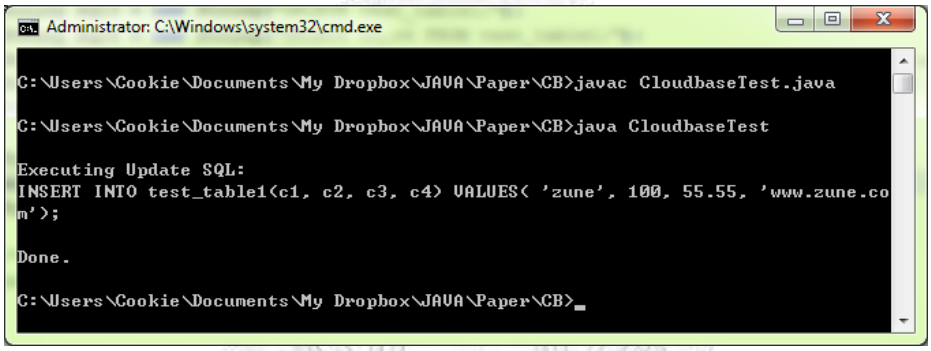

Figure 16 Using Middleware for operation INSERT

Figure 17 Content of test_table1 after the insertion

Users can use middleware to execute the DELETE operation, as the query shown below and the execution screen in Figure 18:

DELETE test_table1 WHERE c1 = 'zune';

Once the operation is completed, the matched row(s) of table test_table1 stored in the HDFS will be deleted. Execution result is shown below in Figure 19.

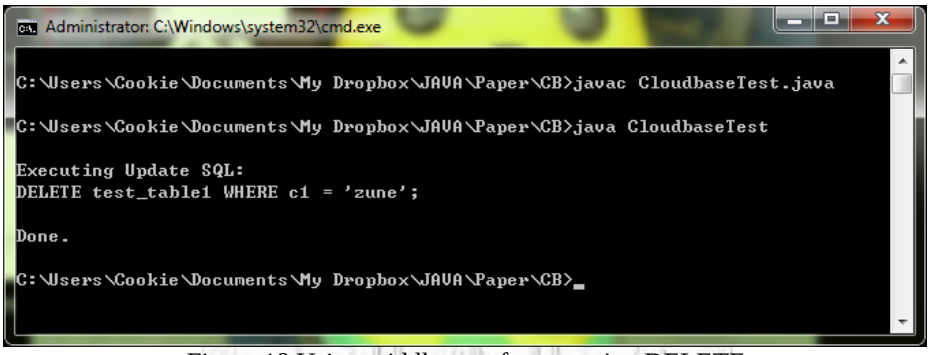

Figure 18 Using middleware for operation DELETE

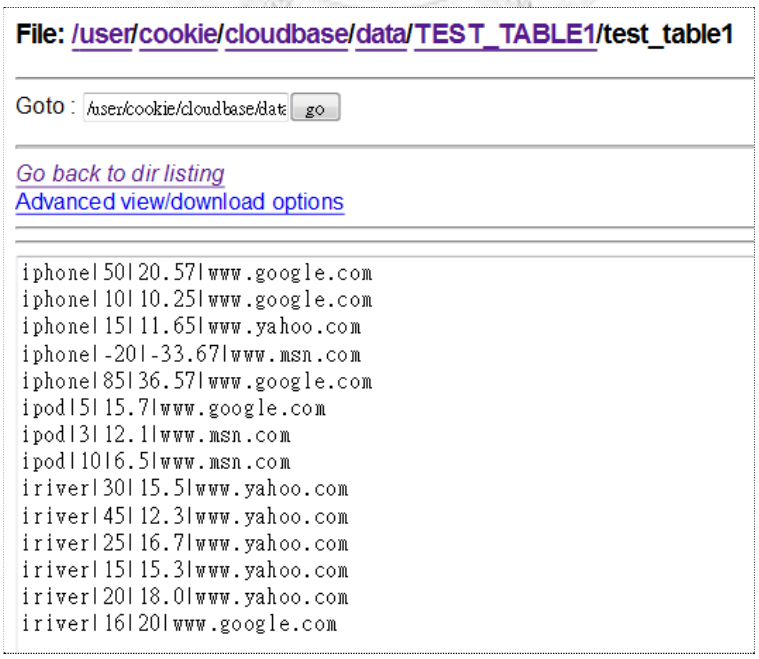

Figure 19 Content of test_table1 after the deletion

Users can use middleware to execute the UPDATE operation, as the query shown below and the execution screen in Figure 20:

```
UPDATE test_table1 SET c1 = 'ipad', c2 = 99 WHERE c1 = 'iphone';
```

Once the operation is completed, the matched row(s) of table test_table1 in the HDFS will be updated. The execution result is shown below in Figure 21.

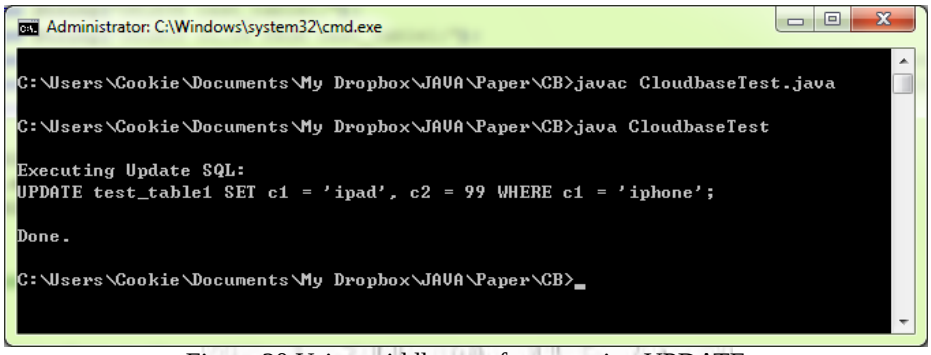

Figure 20 Using middleware for operation UPDATE

Figure 21 Content of test_table1 after the update

4.2 Performance Evaluation

The middleware provided in this thesis can be integrated with CloudBase and form a complete database system. In this chapter, performance analysis will be carried out comparing the PostgreSQL database and the one formed by the middleware, in terms of the time needed to perform a specific command when there are large amounts of data. The commands include SELECT, INSERT, UPDATE and DELETE operations within the database. The table test_table1 is still used here as an example for analyzing and registering the performance of the two database systems when executing SELECT, INSERT, UPDATE and DELETE operations.

Figure 22 and Figure 23 show the time needed to select certain data when both systems have the same number of total registers. The x-axis indicates the number of data already stored in test_table1 and the y-axis shows the time needed to perform a selection under this condition. The execution command is shown below:

```
SELECT * FROM test_table1
WHERE c1 = test0 AND c2 = 0 AND c3 = 0.0 AND c4 = test0
```

As shown in Figure 22, the time needed to complete an execution for the system which integrates CloudBase is between 7.4 to 8.6 seconds, no matter how much data there is in the database. On the other hand, the PostgreSQL performed the SELECT operation quickly when there is not much data, but the extended conversion indicates that its execution time will be longer than the Cloud-based one when total data records reach over 35,000,000 as it is shown in Figure 23.

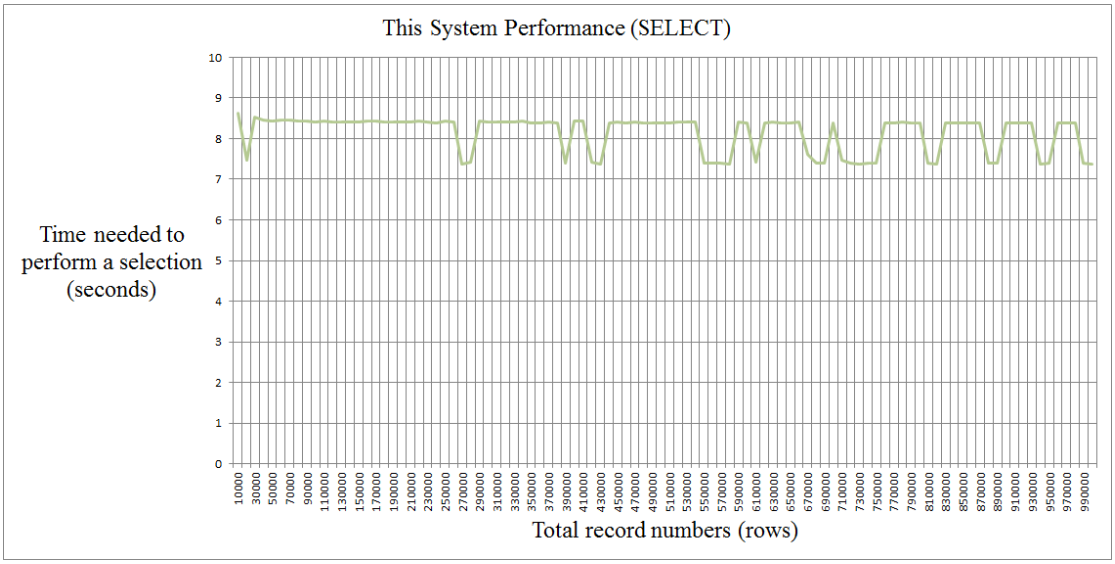

Figure 22 Total row numbers (x-axis) in this system vs. Time (y-axis) needed to perform a selection

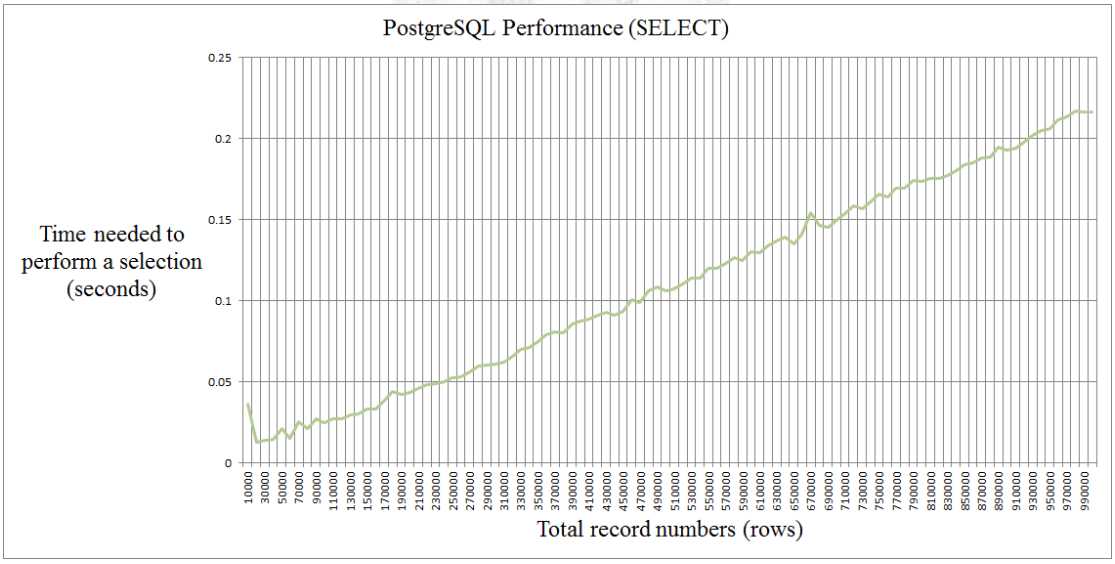

Figure 23 Total row numbers (x-axis) in the PostgreSQL vs. Time (y-axis) needed to perform a selection

Figure 24 and Figure 25 show the time needed to insert a record when both systems have the same number of total registers. The execution command is shown below:

INSERT INTO test_table1(c1, c2, c3, c4) VALUES(test0, 0, 0.0, test0)

Since the INSERT command used in the database requires the middleware to access the last section of the data from the HDFS in Hadoop, and after successfully inserting the data, it will be stored back to the HDFS. It takes at least 0.2 seconds to perform the insertion as it is shown in Figure 24. On the other hand, the PostgreSQL inserts the data directly into the last row of the table and needs only around 0.0007 seconds, as shown in Figure 25. In that sense, the database presented in this thesis will have worse performance than the PostgreSQL one.

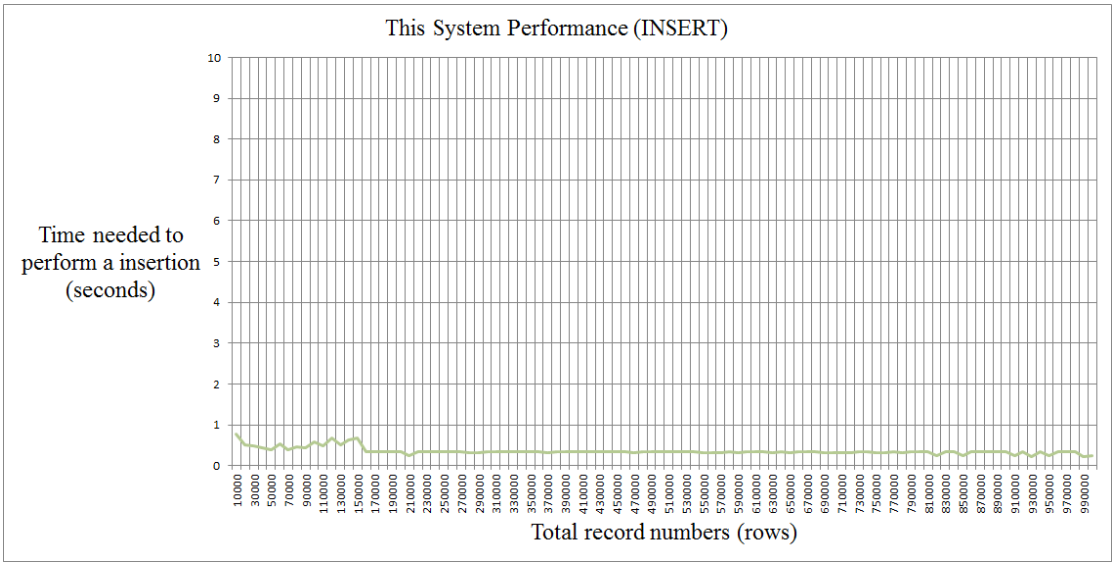

Figure 24 Total row numbers (x-axis) in this system vs. Time (y-axis) needed to perform an insertion

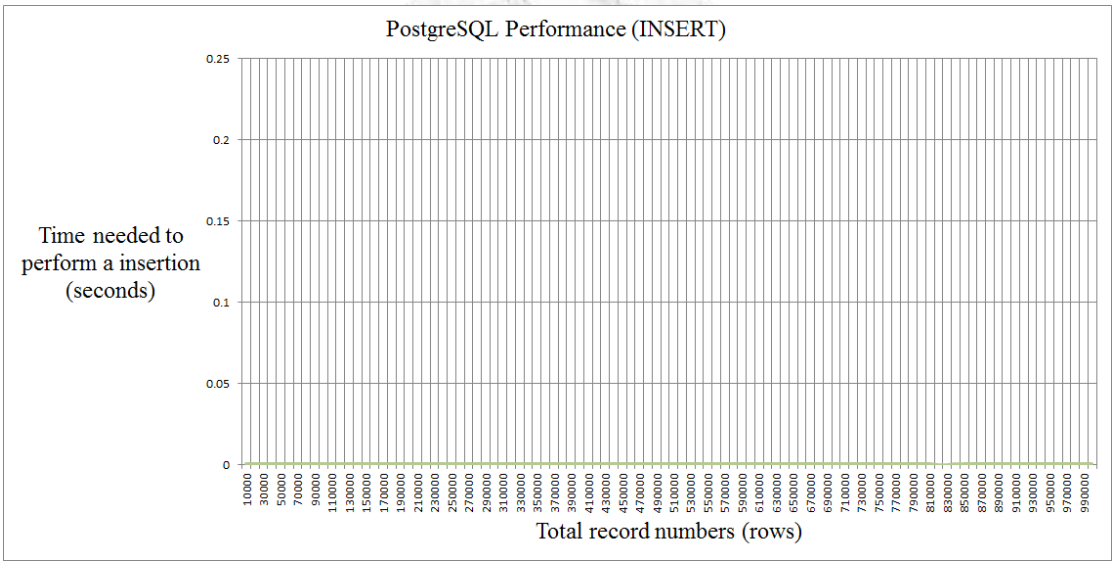

Figure 25 Total row numbers (x-axis) in the PostgreSQL vs. Time (y-axis) needed to perform an insertion

Figure 26 and Figure 27 show the time needed to update a data when both systems have the same number of total registers. The execution command is shown below:

```
UPDATE test_table1 SET c1 = 'test' WHERE c1 =test0 AND c2 = 0 AND c3 = 0.0
AND c4 = test0
```

Just like the time needed for the selection operation, the time used for the middleware-based database is around 8.4 to 9.5 seconds as shown in Figure 26; while the execution time for the PostgreSQL system rises linearly with the total record numbers as shown in Figure 27. After the process of extended conversion, the execution time will be longer than the Cloud-based system when there are more than 35,000,000 records.

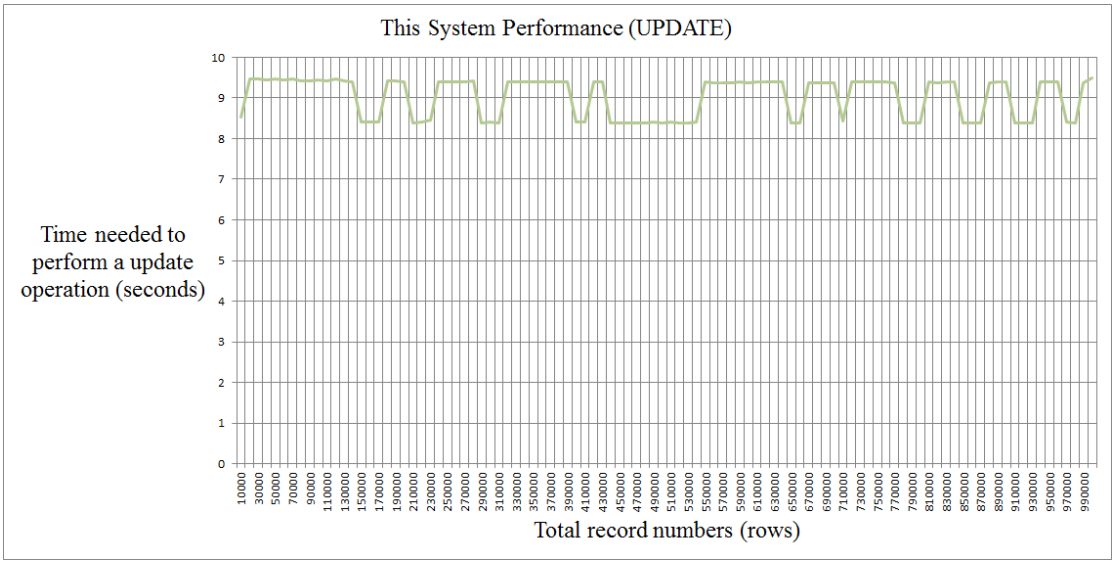

Figure 26 Total row numbers (x-axis) in this system vs. Time (y-axis) needed to perform a update operation

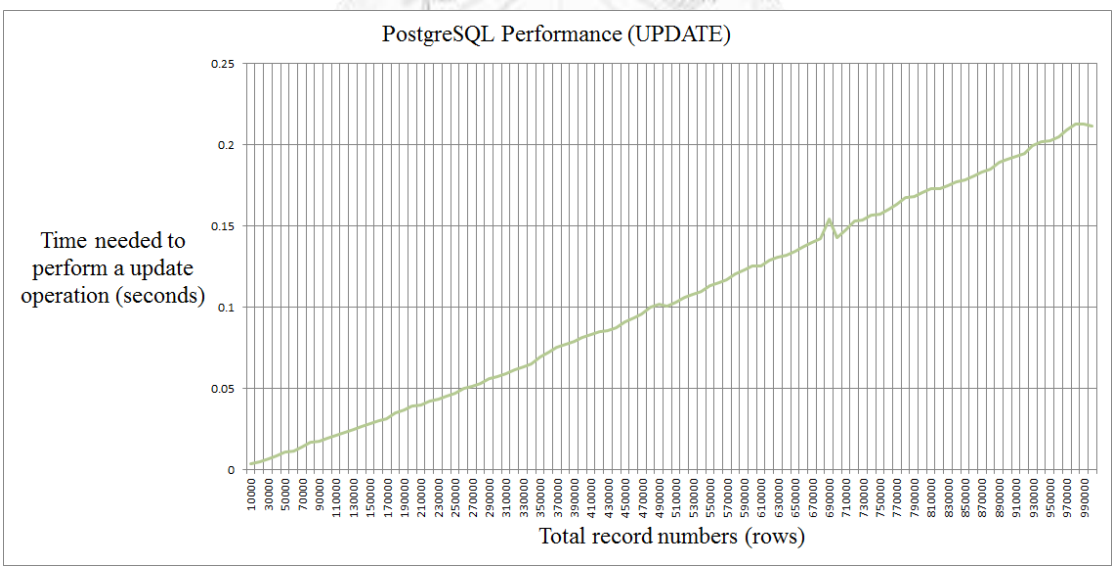

Figure 27 Total row numbers (x-axis) in the PostgreSQL vs. Time (y-axis) needed to perform a update operation

Figure 28 and Figure 29 show the time needed to delete a record when both systems have the same number of total registers. The execution command is shown below:

DELETE FROM test_table1 WHERE c1 = test0 AND c2 = 0 AND c3 = 0.0 AND c4 = test0

Just like the time needed for the deletion operation, the time used for the middleware-based database is around 8.4 to 9.5 seconds as shown in Figure 28; while the execution time of the PostgreSQL system rises linearly with the total record numbers as shown in Figure 29. After the process of extended conversion, the execution time will be longer than the Cloud-based system when there are more than 35,000,000 records.

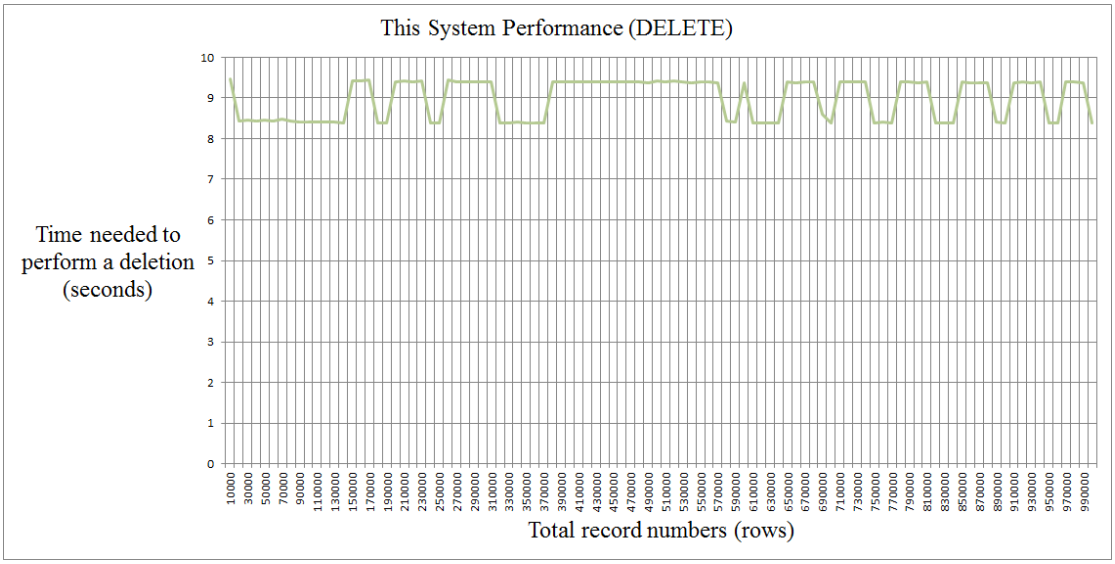

Figure 28 Total row numbers (x-axis) in this system vs. Time (y-axis) needed to perform a deletion

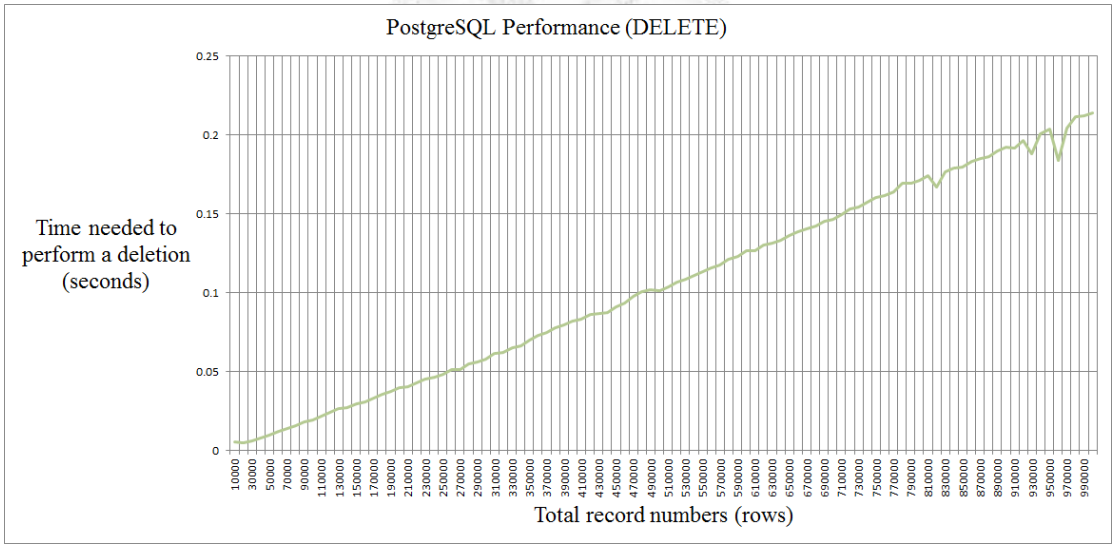

Figure 29 Total row numbers (x-axis) in the PostgreSQL vs. Time (y-axis) needed to perform a deletion

The test is done in an environment based on virtual machines running on a large server (built with Xen 4.0). The specifications of the server are as follows:

A. Number of Clusters：15

B. Type of CPU per Clusters：Intel(R) Xeon(R) CPU X5570 2.93GHz

C. Number of CPU per Clusters：2

D. Capacity of Memory per Clusters：24G Byte

E. Capacity of Disk per Clusters：1.5TB

When testing the PostgreSQL system, a VM is assigned with Xen 4.0, and the configuration is shown below:

A. Operation System of VM：Ubuntu 9.10 64 Bits

B. Speed of CPU：2GHz

C. Number of CPU：2

D. Capacity of Memory：4G Byte

E. Capacity of Disk：100G Byte

When testing the system presented in this thesis, Xen 4.0 is also used to assign VMs for the environment. Since it is based on a Cloud database, there are 5 VMs with the same capacities to carry out the performance test. The configuration for each VM is shown below:

A. Number of VM in Cloud：5

B. Operation System of VM：Ubuntu 9.10 64 Bits

C. Speed of CPU（per VM by Xen）：2GHz

D. Number of CPU（per VM by Xen）：2

E. Capacity of Memory（per VM by Xen）：4G Byte

F. Capacity of Disk（per VM by Xen）：100G Byte

Chapter 5 Conclusion and Future Work

The middleware presented in this thesis can enhance the ability of systems such as "Live E!" to access and store data. With the Cloud computing technology, applications can still access the main database even when some computers or servers in the cluster are damaged or shut down. It also provides better reliability compared with the MySQL or PostgreSQL databases. In addition, Hadoop will make 3 copies of all data in the HDFS and store them evenly in the clusters under default condition, dramatically increasing the security of the information stored in the Cloud database. Finally, Hadoop also offers the function of dynamically increasing or decreasing the number of computers in the cluster, and adjusts the process and storage capacity depending on the current usage of the application. Therefore, it improves the extensibility of the system.

Since the methods of controlling Cloud databases using SQL has been proposed here, it can be applied to redesigning the classic RDBMS in the future, and turn it into a Cloud database. As it is mentioned earlier in this thesis, when "Live E!" collects information with PostgreSQL, it uses only one database node to store immense amounts of external information, and together with other limitations of the database, the reliability and efficiency will decrease when uploading large amounts of data. Therefore, the original data structure as shown in Figure 2 is replaced with the one shown in Figure 30 to improve the ability to access low level databases.

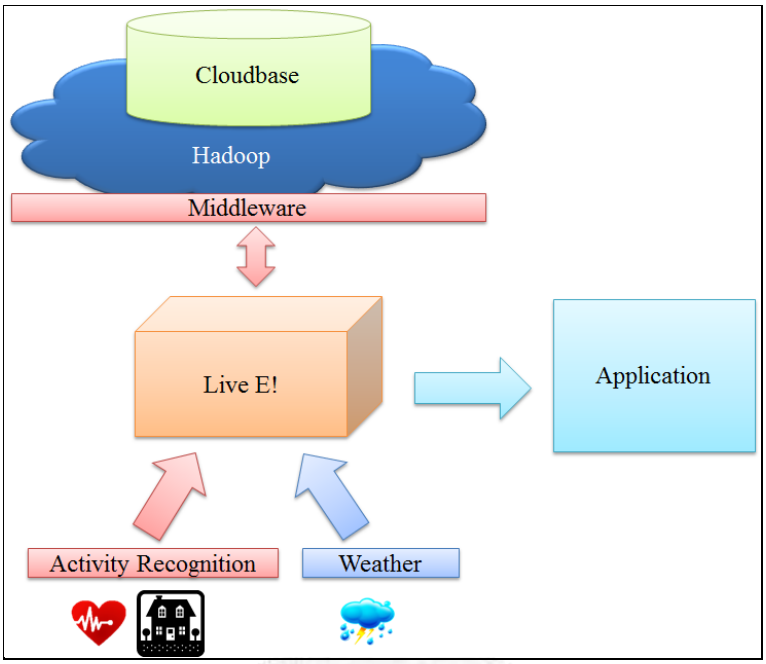

Figure 30 Structure of the integrated “Live E!” system

Other systems using the MySQL and PostgreSQL databases can also convert their databases using the method provided in this thesis without the need to change their system architecture due to limitations from the Cloud. In addition, future work can be focused on integrating the original SQL system and still keeping the benefits of Cloud systems.

References

[1] I. Foster, Y. Zhao, I. Raicu, and S. Lu, “ Cloud Computing and Grid Computing 360-Degree Compared.”, In Proceedings of Grid Computing Environments Workshop, pages 1–10, 2008.

[2] J. Dean and S. Ghemawat, “ MapReduce: Simplifed Data Processing on Large Clusters.”, In Proceedings of the 6th Symposium on Operating System Design and Implementation, pages 137–150, 2004.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, “ The Google File System.”, In Proceedings of the 19th ACM Symposium on Operating Systems Principles, pages 29–43, 2003.

[4] F. Chang, J. Dean, S.Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “ Bigtable: A Distributed Storage System for Structured Data ”, ACM Transactions on Computer Systems, 26(2, Article 4), 2008.

[5] The Hive Project. Hive website, 2009. http://hadoop.apache.org/hive/.

[6] Business.com. Cloudbase website. http://cloudbase.sourceforge.net/.

[7] D. C. Luckham, “ The Power of Event: An Introduction to Complex Event Processing in Distributed Enterprise Systems ”, Addison Wesley, 2002.

[8] S. Rizvi, “ Complex Event Processing Beyond Active Databases: Streams and Uncertainties “, Technical report, Electrical Engineering and Computer Sciences, University of California at Berkeley, 2005.

[9] A. Adi and O. Etzion, “ Amit – The Situation Manager ”, The International Journal on Very Large Data Bases (VLDB）, pages. 177—203, 2004.

[10] F. Wang, S. Liu, and P. Liu, “ Complex RFID Event Processing “, The International Journal on Very Large Data Bases (VLDB), pages. 913—931, 2009.

[11] E. Wu, Y. Diao, and S. Rizvi, “ High-Performance Complex Event Processing over streams ”, In Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pages. 407—418, 2006.

[12] W. Hu, W. Ye, Y. Huang, and S. Zhang, "Complex Event Processing in RFID Middleware: A Three Layer Perspective," Proceeding of third IEEE International Conference on Convergence and Hybrid Information Technology, pages 1121-1125, 2008.

[13] W. Ye, Y. Huang, W. Hu, S. Zhang, and L. Wang, "Towards Passive RFID Event, " Proceeding of 33rd IEEE International Conference on Computer Software and Applications, pages 492-499, 2009.

[14] A. Ranganthan and R. H. Cambell, “An infrastructure for context-awareness based on first order logic “, Personal and Ubiquitous Computing, pages 353–364, 2003.

[15] W. Wang, J. Sung, and D. Kim, “Complex Event Processing in EPC Sensor Network Middleware for Both RFID and WSN “, Proceedings of the 11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing, pages 618—621, 2008.

[16] T. S. Lopez and D. Kim, “Wireless Sensor Networks and RFID Integration for Context Aware Services “, Auto-ID White Paper, number AUTOIDLABS-WP -SWNET-026, 2007.

[17] M. P. Michael and M. Darianian, “ Architectural Solutions for Mobile {RFID} Services for the Internet of Things “, Proceedings of IEEE Congress on Services - Part I, pages 71—74, 2008.

[18] P. Spiess, S. Karnouskos, D. Guinard, D. Savio, O. Baecker, L. M. Souza, and V. Trifa, “ SOA-Based Integration of the Internet of Things in Enterprise Services “, Proceedings of IEEE Conference on Web Service, pages 968—975, 2009.

[19] J. A. Momoh, “ Smart grid design for efficient and flexible power networks operation and control “, Proceedings of the IEEE Conference and Exposition on Power Systems, pages 1—8, 2009.

[20] Appach Hadoop, http://hadoop.apache.org/

[21] J. Sung, T.S. Lopez, D. Kim, “ The EPC Sensor Network for RFID and WSN Integration Infrastructure “, Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, pages 618-621, 2007.

[22] L. Dong, D. Wang, and H. Sheng, “ Design of RFID Middleware Based on Complex Event Processing ”, Proceedings of IEEE Conference on Cybernetics and Intelligent Systems, pages 1-6, 2006.

[23] M. Palmer, “ Seven Principles of Effective RFID Data Management “, Technical Primer, Real Time Division, Progress Software, 2004.

[24] V. A. Crisan, R. Rantzau, "An Alert Notification Facility for RFID Event Repositories", Proceedings of IEEE 24th International Conference on Data Engineering Workshop, pages 88-93, 2008.