Low-power Distributed NoSQL Database for IoT Middleware

Pornpat Paethong, Mikiko Sato, Mitaro Namiki
Tokyo University of Agriculture and Technology
Tokyo, Japan
pornpat.pp@gmail.com, mikiko@namikilab.tuat.ac.jp, namiki@cc.tuat.ac.jp

***Abstract*—The Internet will become to the Internet of Things (IoT). It provides connectivity for everyone and everything that embeds some intelligence in Internet-connected objects to communicate, exchange information, take decisions, invoke actions and provide amazing services. This will bring a new ubiquitous computing and communication era and change people's lives (drastically/entirely). Moreover, the most important component in IoT is database that used for collecting and storing a lot of data from ubiquitous sensing devices. Even though, cloud-based storage solutions are becoming increasingly popular in recent years, it is inconvenient for residential environments because the sensing devices should be connected to the internet for simultaneously sending data to cloud computing. However, this solution has a high cost implication and high power consumption. Therefore, the credit-card sized computer which is fully functional is a good alternative. It is small and inexpensive computer that provides a new opportunity for IoT hardware. As a result, it is possible to use Raspberry Pi as a database server. In this paper, we will explain how to construct a database server for IoT middleware that has data distribution and low-power consumption by using credit-card size computers which have satisfactory performances and affordable price.**

Keywords—Credit-card Sized Computer, Distributed Database, Embedded System, Internet of Things, IoT Middleware, MongoDB, NoSQL Database, Low-power, Raspberry Pi

I. INTRODUCTION

Nowadays, the Internet will become to the Internet of Things (IoT) that is able to have an immediate access to information about the physical world and its objects. It has been introduced to integrate the virtual world of information and the real world of devices. IoT covers the infrastructure, which can be hardware, software and services, to support the networking of physical world objects [1,2]. The IoT aims to provide a simple interaction between the physical world and the virtual world, by integrating a large numbers of real-world physical devices (or things) into the Internet. Therefore, it provides connectivity for everyone and everything that embeds some intelligence in Internet-connected objects to communicate, exchange information, take decisions, invoke actions and provide amazing services [2,3]. This will bring a new ubiquitous computing and communication era and change people's lives (drastically/entirely).

One of the most important components in IoT is the database. It will work in role for collecting and storing a lot of data from ubiquitous sensing devices [1]. The presence of smart devices able to sense physical objects and translate them into a stream of information data, as well as the IoT devices able to trigger actions, maximizes safety, security, comfort, convenience and energy-savings. These devices will reach nearly 26 billion connected devices by 2020. It is important to develop artificial intelligence algorithms which could be centralized or distributed for supporting. One of the key platforms for IoT is the credit-card size computer such as Raspberry Pi. It is a popular platform because it offers a complete Linux server in a tiny device with very low cost and satisfactory performance.

In this paper, we will explain how to construct database server for IoT middleware that have data distribution and low-power consumption by using credit-card size computer like Raspberry Pi with satisfactory performances and affordable price. It is a perfect platform for interfacing with many ubiquitous sensing devices of residential environment such as homes, offices, or farms.

II. ISSUES AND GOALS

The internet is expected to become the Internet of Things that integrate the virtual world of information and the real world of devices. It is able to have an immediate access to information about the physical world and its objects. This means a lot of data from ubiquitous sensing devices will be present in a big database for analyzing and visualizing.

Normally, the database component of IoT will implement on high-end servers as a cloud-based storage that is high reliability, redundant, and security [1,2]. Even though, cloud-based storage solutions are becoming increasingly popular in the years ahead, it is inconvenient for residential environment because the sensing devices should be connected to the internet for simultaneously sending data to cloud computing. However, this concept is expensive and has high power consumption when applied to homes, offices, or farms environments. So, the credit-card sized computer like Raspberry Pi which is fully functional is a good alternative [3,4,5]. It is a small, inexpensive computer that provides a new opportunity for IoT hardware. On the other hand, it is possible to use Raspberry Pi as a database servers for IoT local-base storage.

The main goal of this research is to define and present advantages and disadvantages of using credit-card size computer as a database server for middleware and abilities of its usage in the development of the next generation of IoT. The purposed concept can provide a simple and flexible database by using

978-1-5090-1125-4/16/$31.00 ©2016 IEEE

NoSQL database with low-cost and low-power between sensor node and cloud computing.

III. Challenge and Ideas

Even though, cloud-based storage is easy and popular for IoT in the near future, it is inconvenient for sensing devices in residential environment. It should be possible to connect to Internet all the time for passing data. This is costly and has high power consumption when applied to homes, offices, or farm environments because of limited power source and internet connectivity.

Raspberry Pi is a credit-card size computer that was developed in the Laboratory of University of Cambridge and released by Raspberry Pi Foundation in 2012. It is a low-power with ARM processors, commodity Ethernet interconnects, and low-power flash based local storage. Moreover, it has general-purpose input and general-purpose output (GPIO) connectors for communication with sensors, motors and other embedded systems. It offers good enough performances and affordable price hardware [5,6,7].

MongoDB database is a cross-platform document-oriented database. Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. As of July 2015, MongoDB was the fourth most popular type of database management system, and the most popular for document stores [8,9].

With the power of these two main technologies, Raspberry Pi and MongoDB database, we can create the database server by using credit-card sized computer and NoSQL database engine for IoT middleware. This system can use Raspberry Pi or some common alternatives such as Banana Pi, Orange Pi or BeagleBone Black because it is also small, powerful, cheap, and programmable computer board.

IV. System Structure

In this system, it is contains two main parts: master node and data nodes. They work together as a single system which the master node is able to connect to one or more data nodes for data distribution but data node is able to connect to only one master node.

Conceptually, we will be running NoSQL database by using MongoDB in individual data node for collecting data from their own sensor data which is in wake up and sleep mode. The master node will wake an individual data node up from sleep mode and get data from them. In every data node, normally, they runs MongoDB database by themselves as an individual node and they are also able to receive command from master node. Master node always stays awake to wait for accesses from clients or users. By using wake up and sleep mode as active and inactive concepts, it can reduce power consumption of each node following by Fig.1.

A. Master Node

In master node, it will play in roles of communicator with clients and commander with data nodes for controlling. This node can connect to one or more data nodes via local network such as Ethernet, WiFi or XBee communication. It will always stay awake for receiving and responding to client via internet. Moreover, it can be active and inactive to data node for power saving.

B. Data Node

In data node, it should be connected with sensors, motors and other embedded systems by using GPIO pins. It will store these data into database on itself. Finally, it will provide these data to master node when it requests.

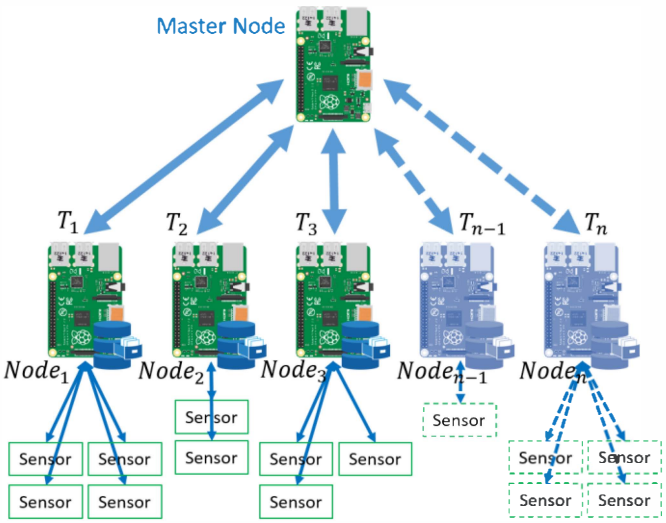

Fig.1: System Structure and Design

V. System Design

Regarding the design, we are proposing for two solutions for low-power distributed NoSQL database on credit-card size computer that have flash-based local storage. There are non-data copy oriented and data copy oriented from data node to master node. The client will connect to master node by using database application program interface (Database API) for query command. The master node will wake data node up by active signal before getting the data (shown in Fig.2).

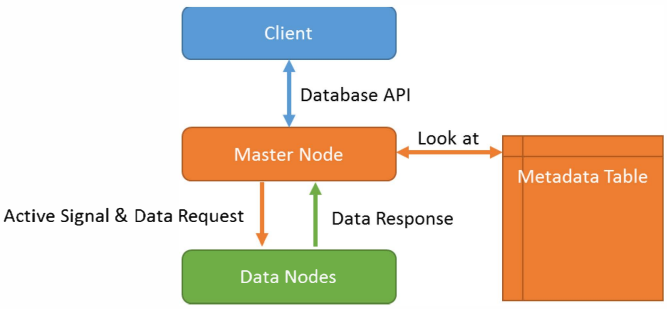

Fig.2: System Design Conceptual

A. Non-data Copy Oriented Design

In the first step, the client will send the query request to master node by using Database API. Second, the master node will look at metadata table of sensor nodes stored within it and find the node with data that matches with query request. Third,

the master node will send an active signal to selected data nodes that matched the metadata table. Fourth, the data node will send the sensor data to master node. Finally, the master node will combine all of sensor data from selected data nodes then respond to the client.

1) Advantages

- The data node will use their own storage for collecting and storing sensor data in individuals.
- It can use its own local storage efficiently.
- It is distributed data.

2) Disadvantages

- It is slow because it will send all the data that matched for every request from master node.

B. Data Copy Oriented Design

Both the data copy oriented and non-data copy oriented have the same designs although there are a few differences. Normally, the data nodes do periodic transmission of updated sensor data to store in master node. In fourth step, when the master node collect data from data node, the data node will sent updated sensor data to master node only.

1) Advantages

- It is faster than non-data copy oriented design.

2) Disadvantages

- The master node should have a very large volume of local storage.

These two solutions: non-data copy oriented and data copy oriented. They offer different advantages and disadvantage s depending on need. If usage storage efficiently is important, the non-data copy oriented solution is better. On the other hand, if speed is the priority, the data copy oriented solution is good enough.

C. Query Divide Algorithm

In master node, when it is receiving a query request from the client, it will forward that query to selected data nodes. Before this, master node will select data node for getting data by looking at sensor node metadata on itself first. In sensor node metadata, it is a table in master node that contains information of data node as metadata: data node id, sensor key and last active. The master node will use these metadata for selecting data nodes. Then, it will be sent an active signal to selected data nodes and get data these node by using same query request from client.

D. Power Control Algorithm

In credit-card size computer board, mostly, it does not have a power management component on itself. Therefore, we will suggest an ideas and algorithms to power control and operations for low power consumption.

1) Wake-on LAN (WOL)

It is an Ethernet or Token ring computer networking standard that allows a computer to be turned on or awakened by a network message. The message is usually sent by a program executed on another computer on the same local area network. It can make wake data node up and sleep by using magic network packet.

2) CPU Clock Speed and Frequency

As the Raspberry Pi does not have a conventional BIOS, the various system configuration parameters that would normally be kept and set using the BIOS are now stored in a text file named "config.txt". Therefore, it can experience overclocking.

Regarding the overclocking, it has some important configuration options for low power.

1) arm_freq: frequency of ARM in MHz. 2) sdram_freq: frequency of SDRAM in MHz. 3) arm_freq_min: minimum value of arm_freq used for dynamic clocking. 4) core_freq_min: minimum value of core_freq used for dynamic clocking. 5) sdram_freq_min: minimum value of sdram_freq used for dynamic clocking. 6) over_voltage_min: minimum value of over_voltage used for dynamic clocking.

With the overclocking configuration, it can control power of Raspberry Pi. When the work load is high, it will use maximum power. On the other hand, when it is standby or free mode, it will use minimum power for power saving.

VI. EVALUATION AND DISCUSSIONS

In this evaluation, we will compare between x86 machine and Raspberry Pi for operation time and power consumption to support our proposed. The evaluation environments for x86, Intel Core i7 870 @ 2.93GHz, 3.9 GB of memory, 128 GB of disk storage, and Ubuntu 14.04 LTS. For Raspberry Pi 2 Model B, ARM Cortex-A7 @ 900 MHz, 1 GB of memory, 16 GB of SD card and Raspbian Jessie 4.1. In addition, MongoDB 2.6.12 and MySQL 5.5.49 for x86 machine and MongoDB 2.4.10 and MySQL 5.5.44 for Raspberry Pi.

A. MySQL and MongoDB Comparison

This comparison shown the MySQL and MongoDB performance for data inserting and data selecting on Raspberry Pi as shown in Table I. In the table, "I" is represents inserting data into database, "SA" is represents searching of all records, "SW" is represents searching with where conditions, "UA" is represents updating of all records, "UW" is represents update with where conditions, "DA" is represents delete all records, and "DW" is represents deleting with where conditions.

TABLE I. MYSQL AND MONGODB COMPARISON

Database	I	SA	SW	UA	UW	DA	DW
MySQL/ x86	76.64	6.72	0.85	8.93	1.98	0.03	0.50
MySQL/ Rasp. Pi	765.54	4.98	2.14	31.62	6.15	3.03	2.00
MongoDB/ x86	1,084.24	8.87	0.28	3.70	0.29	6.40	0.28
MongoDB/ Rasp. Pi	1,086.61	7.42	4.27	45.88	4.48	126.20	4.31

The results shown the execution time of database operation with one million data records for MySQL and MongoDB on x86 and Raspberry Pi. In case of inserting, searching with where condition and updating with where condition, MySQL on Raspberry Pi was faster than MySQL on x86, and MongoDB Raspberry Pi on was faster than MongoDB on x86 also. In the other case, MySQL and MongoDB on x86 was faster than Raspberry Pi.

B. *Operation Time Measurement*

This evaluation shows how long MongoDB is used for database operations such as searching, based on one million, two million and five million data records (Fig.3 and Fig.4).

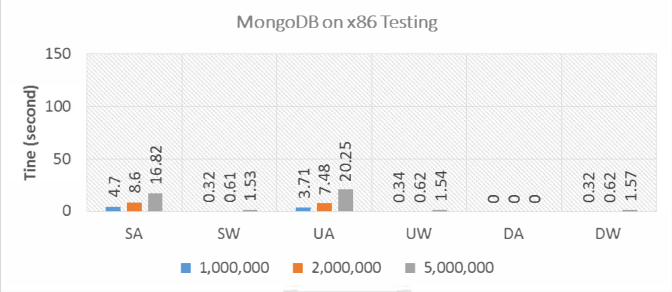

Fig.3: Operation Time of MongoDB on x86 Machine

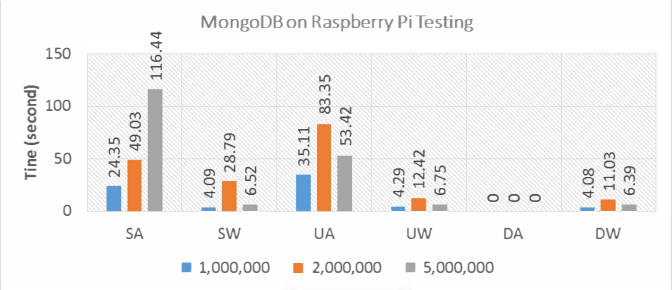

Fig.4: Operation Time of MongoDB on Raspberry Pi

The results show that x86 machine consumes 5 seconds, 9 seconds and 17 seconds for searching on one million, two million and five million records respectively. In contrast, Raspberry Pi consumes 25 seconds, 49 seconds and 116 seconds for searching on one million, two million and five million records respectively. Moreover, x86 machine consumes less time than Raspberry Pi for other database operations. In brief, MongoDB on Raspberry Pi is 80% slower than x86 machine in every database operation.

C. *Power Consumption*

The watt-second is the energy equivalent to the power of one watt sustained for one second. This evaluation shows how much energy that MongoDB used for database operations such as searching, based on one million, two million and five million data records (Fig.5 and Fig.6).

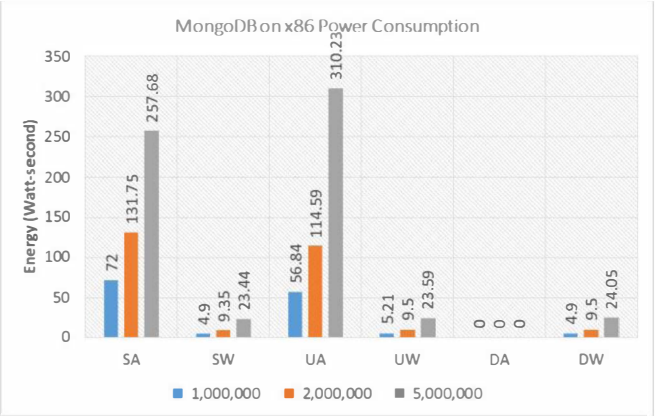

Fig.5: Power Consumption Measurement of MongoDB on x86 Machine

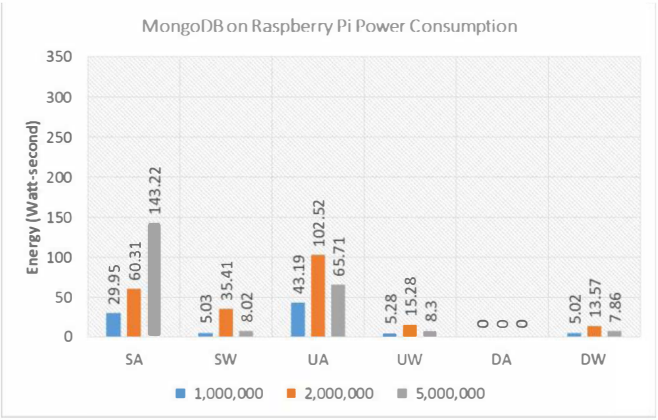

Fig.6: Power Consumption Measurement of MongoDB on Raspberry Pi

As a result, while executing in the same database operation, x86 machine consumes 72 joules, 132 joules and 258 joules for searching on one million, two million and five million records respectively. In contrast, Raspberry Pi consumes 30 joules, 60 joules and 143 joule for searching on one million, two million and five million records respectively. In brief, Raspberry Pi reduces power consumption for 45% compare to x86 machine.

VII. Conclusion and Future Works

In conclusion, we discuss operation time, power consumption, energy efficiency of Raspberry Pi. It offers a distinct advantage over x86 machine in efficiency. In our performance comparison, we have evaluated the operation time and power consumption of Raspberry Pi and x86 machine. The results show that Raspberry Pi consumes 45% less than x86 machine in terms of power consumption.

Regarding the evaluation result, it will support us to develop a simple and flexible database for IoT middleware using credit-card size computer hardware and MongoDB database engine as affordable database server in the future.

References

[1] Gubbi, Jayavardhana, et al. “Internet of Things (IoT): A vision, architectural elements, and future directions.” Future Generation Computer Systems 29.7, pp. 1645-1660, 2014.

[2] Zarghami, Shirin. “Middleware for Internet of things.”, 2013.

[3] Brock, J. Dean, Rebecca F. Bruce, and Marietta E. Cameron. “Changing the world with a Raspberry Pi.” Journal of Computing Sciences in Colleges 29.2, pp. 151-153, 2013.

[4] Cox, S.J., Cox, J.T., Boardman, R.P., Johnston, S.J., Scott, M. and O’brien, N.S., “Iridis-pi: a low-cost, compact demonstration cluster. Cluster Computing”, 17.2, pp.349-358, 2015.

[5] Anwaar, Waqas, and Munam Ali Shah. “Energy Efficient Computing: A Comparison of Raspberry PI with Modern Devices.” Energy 4.02, 2015.

[6] Snyder, Robin M. “Power monitoring using the Raspberry Pi.” Association Supporting Computer Users in Education “Our Second Quarter Century of Resource Sharing” pp. 82, 2014.

[7] Maksimović, Mirjana, et al. "Raspberry Pi as Internet of things hardware: performnces and constraints." Design Issues 3, pp. 8, 2014.

[8] Strauch, Christof, Ultra-Large Scale Sites, and Walter Kriha. “NoSQL databases.” Lecture Notes, Stuttgart Media University, 2011.

[9] Parker, Zachary, Scott Poe, and Susan V. Vrbsky. “Comparing NoSQL MongoDB to an SQL DB.” Proceedings of the 51st ACM Southeast Conference. ACM, 2013.