Data Collection System Using Smart Device Databases and a Hash Chain Protocol

Lei Tong
Graduate School of Information and Telecommunication Engineering, Tokai University, Tokyo, Japan
Email: 3bji1111@mail.u-tokai.ac.jp

Hiromi Kobayashi
Department of Information Media Technology, Tokai University, Tokyo, Japan
Email: koba@tokai.ac.jp

***Abstract*—This paper presents a database service system in which smart device databases are used. Every smart device, such as a smartphone or tablet device, has a relational database management system (RDBMS) as the default. First, an image data collection system is proposed. This system is composed of databases in smart devices as client side databases and a database in a server. The data service should ideally be an online real-time service; however, real-time is not always required. Next, a hash chain protocol called a laced hash chain protocol is presented to preserve the consistency of data transferred from each client to a server. Developing a prototype system consisting of several smart devices with SQLite and a Windows database server with MySQL is an ongoing work.**

***Index Terms*—database system, smart device, image data, hash chain, SQLite**

I. Introduction

A database service system is normally constructed from a center database server system these days. Smart devices such as smartphones or tablet devices are used as client retrieving devices to obtain database services. A smart device contains a RDBMS such as SQLite as a library in Android. However, these smart device databases are not used effectively in database service systems. When many clients access a Web database service system to exchange large amounts of data such as image data, the system performance is degraded because of the data transmission line overload or insufficient access capacity of the database system. To overcome this problem, we propose the following system architecture that uses databases in smart devices in a data collection system. First, a large volume of image data such as pictures or videos is collected using client's databases in smart devices. Next, these data are transferred to a database server for accumulation of these data during a convenient time, considering the entire load of the system, perhaps once a day at night using batch processing. Holding consistency between relational databases is easy. The data service should ideally be an online real-time service. However, real-time is not required according to the type of service for technical or economical reasons. Next, we propose a laced hash chain protocol to reserve consistency of data transmission. A hash chain [1]-[3] is a cascade function originally proposed by Lamport [1] adapted in systems such as a practical Byzantine fault (PBFT) protocol called Zyzzyva [4]-[7] for reliable distributed database systems. Additionally, a hash chain is used as a block chain in a bitcoin system [8]-[12]. A hash sequence is used between a database server and each client's database in our protocol as a bootlace.

Manuscript received January 2, 2017; revised June 28, 2017.

II. Related Work

Large object data such as a set of pictures or image data can be stored in a table as attribute values in ANSI/ISO SQL-92, or SQL2 standard using Binary Large Object (BLOB) type. However, sometimes only a link of each large object is stored in a table and the large object is stored outside of the RDBMS in practical use because of disk capacity limitations. We attempt to use these two methods in our system for comparison.

A hash chain is used to improve data transmission reliability. A hash chain is a cascade function denoted as a recurrence relation $h_n=H(h_{n-1}, d)$ where h_n and h_{n-1} respectively denote successive values at step n and step n-1, and d denotes the digest of data. Here, h_0 is a nonce, or an initial value. H is a hash function. This sequence forms the history of data. A hash chain is applied to the protocol in a highly reliable distributed database system such as a practical Byzantine fault (PBFT) protocol called Zyzzyva [4]-[7]. The Byzantine fault is the severest fault: a faulty node sends different values to different nodes by some unforeseen timing in a distributed system [13], [14]. A hash chain is used in the block chain mechanism preventing asset data from rewriting dishonestly in a bitcoin system [8]-[12]. It is used in online game protocols to prevent cheating of players [15], [16]. We apply a hash chain called a laced hash chain protocol to the transfer mechanism in our system.

III. System Architecture Outline

Our data collection system consists of a client-server system. The client comprises a plural number of smart

doi: 10.18178/lnit.5.1.13-16

devices such as smart phones and tablet devices. Each device contains an embedded RDBMS such as SQLite [17] in Android. On the server side, a server contains a RDBMS such as MySQL [18], as presented in Fig. 1. The procedure of this system is the following.

(1) First, each smart device collects image data in its database. When each image datum is inserted in a table, the time stamp and id of each device are also inserted in each row of the table.

(2) Next, these data are transmitted to the database server. This data transfer is performed where the capacity of transmission line has room, as in a high traffic Wi-Fi spot and when the database server load is low.

Step (1) presents two methods for storing image data in the table of a relational database. By one, each image is inserted into a table directly using binary large object (BLOB) type, as presented in Fig. 2. By another, only the link of each image is inserted into a table. The image itself is stored at the outside space of the relational database, as depicted in Fig. 3. An application program is necessary to manage links and corresponding images in the latter case, as shown in Fig. 4.

In fact, the timestamp type or datetime type is useful in MySQL, but these take on different forms depending on database products. In addition, these types cannot be used in SQLite. Therefore, varchar type is used to maintain compatibility between databases.

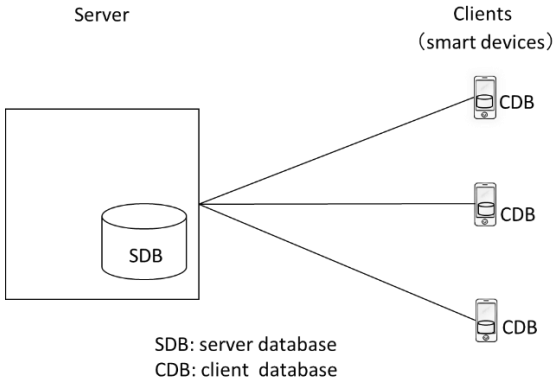

Figure 1. System architecture outline.

```
create table imagedata(
  pid    int not null auto_increment,
  image  blob,
  caption varchar(255),
  timestamp varchar(255),
  nid    varchar(255),
  primary key(pid)
 )
```

Figure 2. Definition of a table of image data.

```
create table imagelink(
  pid  int not null auto_increment,
  file_path varchar(255),
  caption varchar(255),
  timestamp varchar(255),
  nid    varchar(255),
  primary key(pid)
 )
```

Figure 3. Definition of a table of image file path. (Image data are stored outside of the database.)

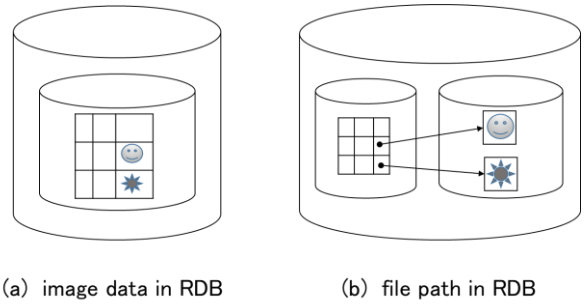

Figure 4. Two methods for storing image data.

IV. LACED HASH CHAIN

We propose a hash chain protocol called the laced hash chain protocol for data safety of this data collection system. The principle of the hash chain uses a function such that $h_n=H(h_{n-1}, d)$ where h_n and h_{n-1} respectively denote successive values at step n and step n-1, and d denotes the digest of data. This sequence forms the history of data. Fig. 5 presents the main part of our protocol between a server (node 0: n_0) and each client (node k (k=1,2,…m):n_k). Here $m_{0,k,i}$ is a part of data composed of appending all rows of <pid, cimg, cp, ts, nid> and $h_{k,\ i-1}$ where pid, cp, ts, and nid respectively denote the image datum id, caption, timestamp, and node id. These are attributes of data in a table, as presented in Fig. 2-3. Also, cimg represents the upper 128 bits of an image datum when using BLOB type, as presented in Fig. 2, because the quantity of image data is too large. Otherwise, cimg represents the file_path when storing only the link of each image datum in a table, as depicted in Fig. 3. $h_{k,\ i-1}$ denotes a hash value made in client node k received at the previous step. $d_{0,k,i}$ is the digest of data such that $d_{0,k,i} = H(m_{0,k,i})$. Actually, $h_{0,k,i}$ is the summary of the history such that $h_{0,k,i} = H(h_{0,k,i-1}, d_{0,k,i})$. Subsequently, message < $h_{0,k,i}$, nid> is sent to client n_k.

@server(n_0: node 0) step i

while (1){
 $m_{0,k,i}$ = <append all rows of<pid, cimg, cp, ts, nid>, $h_{k,\ i-1}$ >
 $d_{0,k,i} = H(m_{0,k,i})$ //digest of data
 $h_{0,k,i} = H(h_{0,k,i-1}, d_{0,k,i})$ //summary of the history
 send < $h_{0,k,i}$, nid> to n_k
 wait
 receive <$h_{k,i}$, $\Delta D_{k,i}$, nid > from n_k
 i++
}

@client k (n_k: node k) step j (k = 1, …, m)

while (1){
 wait
 receive < $h_{0,k,i}$, nid> from n_0
 $m_{k,i}$ = <append all rows of<pid, cimg, cp, ts, nid>, $h_{0,k,\ i}$ >
 $d_{k,i} = H(m_{k,i})$ //digest of data
 $h_{k,i} = H(h_{k,i-1}, d_{k,i})$ //summary of the history
 send < $h_{k,i}$, $\Delta D_{k,i}$, nid> to n_0
 i++
}

Figure 5. Main part of our laced hash chain protocol.

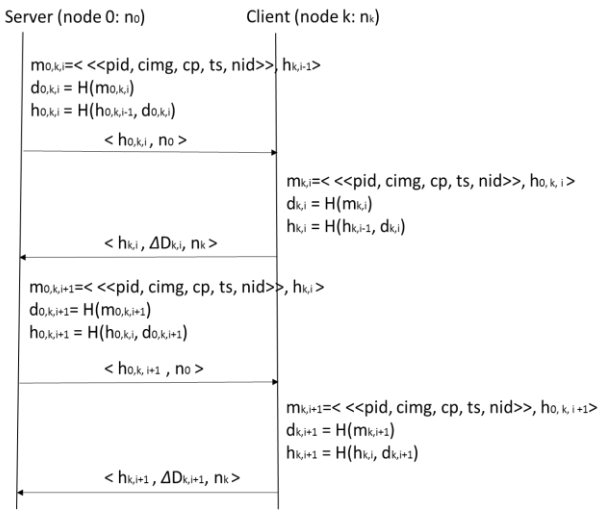

Figure 6. Sequence of laced hash chain protocol.

The procedure at client k is similar to that in the server. $h_{0,k,\ i}$ denotes a hash value made in server (node 0) received at the previous step. $d_{k,i}$ is the digest of data such that $d_{k,i} = H(m_{k,i})$. $h_{k,i}$ is the summary of the history such that $h_{k,i} = H(h_{k,i-1,}\ d_{k,i})$. Subsequently, message < $h_{k,i}$, $\Delta D_{k,i}$, nid> sends to server n_0. $\Delta D_{k,i}$ denotes the incremental data at step i in client k. The sequence diagram is shown between the server and a client at step i and step i+1 in Fig. 6.

V. Implementation

We are implementing our system as described below.

A. System Architecture

The system consists of a Windows server with a relational database and several smart phones and tablet devices with Android OS as clients. Jetty [19] is used as servlet engine for developing Web applications. Java is used as the programming language. Not HTTP but Web socket is used as this transmission protocol in preparation for a large volume of data between each client and the server.

B. RDBMS

MySQL is used as a server side RDBMS on Windows OS. It is an open source RDBMS. Several smart phones and tablet devices based on Android OS are used as clients. SQLite is embedded RDBMS in Android OS as default. It is useful as a library of programming language such as Java. SQLite can be called from a single application in Android. A special mechanism called a content provider is used when called from multiple applications.

C. Data Type

MySQL has various data types. Regarding BLOB type, TINYBLOB, BLOB, MEDIUMBLOB, and LONGBLOB are useful depending upon the object size. DATETIME and TIMESTAMP types are useful, too. On the other hand, SQLite has only five data types: INTEGER, REAL, TEXT, BLOB, and NULL. However, field types are useful for compatibility of other RDBMSs. For example, CHAR, VARCHAR, CLOB: character large object, and TEXT are regarded as TEXT type in table definition. Undefined data are regarded as NUMERIC.

D. Incremental Data

Data transmission is performed using incremental data at each step. The incremental data are discriminated using the time stamp value stored as VARCHAR type at each row in each table. In Fig. 5, $\Delta D_{0,i}$ denotes the incremental data at step i in the server such that

$\Delta D_{0,i} = D_{0,i} - D_{0,i-1}$.

$D_{0,i} = D_{0,i-1} \cup D_{1,i} \cup D_{2,i} \cup \cdots \cup D_{k,i}$.

Therefore, a server database reflects the change of incremental data in each client at step i.

$\Delta D_{k,i}$ denotes the incremental data at step i in client k such that $\Delta D_{k,i} = D_{k,i} - D_{k,i-1}$.

Attributes of $D_{0,i}$ and $D_{k,i}$ are (pid, image [or file_path], caption, timestamp, nid), as presented in Fig. 2 and Fig. 3.

E. Hash Value Calculation

A detailed description of hash value calculation is provided as follows in our prototype system.

In server n_0: At step i, $m_{0,k,i}$ = <append all rows of <pid, cimg, cp, ts, nid>, $h_{k,\ i-1}$ > where pid, cp, ts, and nid respectively denote the image datum id, caption, timestamp and node id. These values are obtained from all rows in imagedata table or imagelink table. cimg represents the upper 128 bits of the image datum in each row when using BLOB type as in imagedata table. Otherwise, cimg represents the file_path when storing only the link of each image datum in the table. $h_{k,\ i-1}$ denotes a hash value generated in n_k and sent at the previous step from n_k to n_0. $d_{0,k,i}$ is the digest of data such that $d_{0,k,i} = H(m_{0,k,i})$. $h_{0,k,i}$ is the summary of the history such that $h_{0,k,i} = H(h_{0,k,i-1,}\ d_{0,k,i})$. Subsequently, message < $h_{0,k,i}$, nid> sends to client n_k.

The procedure at client k is similar to that in the server as follows. In client n_k: At step i, $m_{k,i}$ = <append all rows of <pid, cimg, cp, ts, nid>, $h_{0,k\ ,i-1}$ > where pid, cp, ts, and nid respectively denote image datum id, caption, timestamp and node id. These values are obtained from all rows in imagedata table or imagelink table. cimg represents the upper 128 bits of image datum in each row when using BLOB type as in imagedata table. Otherwise, cimg represents the file_path when storing only the link of each image datum in the table. $h_{0,k,\ i}$ denotes a hash value generated in n_0 and sent at the previous step from n_0 to n_k. $d_{k,i}$ is the digest of data such that $d_{k,i} = H(m_{k,i})$. $h_{k,i}$ is the summary of the history such that $h_{k,i} = H(h_{k,i-1,}\ d_{k,i})$. Subsequently, message < $h_{k,i}$, $\Delta D_{k,i}$, nid> sends to server n_0. $\Delta D_{k,i}$ denotes the incremental data at step i in client n_k.

VI. Conclusion and Future Work

It is desirable that database services be online in a real-time manner. However, a heavy load degrades the system performance when using a large volume of data. The situation is regarded as solvable when the hardware technology and the infrastructure of communication are improved. However, the data volume is increased rapidly

in database services. Therefore, not all database services will be achieved online in real-time in the near future. We propose a data collection system that gathers a large volume of data using smart devices effectively. A relational database is implemented in every smart device such as SQLite in Android. It is easy to hold consistency of data between a client and a server when both sides use relational databases. Next, we propose the laced hash chain protocol to reserve the consistency of data transmission. The novelty our protocol is that our protocol uses a hash chain as a bootlace. Next, we shall attempt to develop a simulation program to estimate our protocol [20]. Additionally, we are developing a prototype system consisting of several smartphones and tablet devices containing SQLite and a Windows database server with MySQL. Then we plan to conduct estimations for this system.

REFERENCES

[1] L. Lamport, "Password authentication with insecure communication," *Communications of the Acm*, vol. 24, pp. 770-772, 1981.

[2] Y. W. Peng and W. M. Chen, "A recursive algorithm for general hash chain traversal," in *Proc. IEEE 17th CSE*, 2014, pp. 1171-1174.

[3] S. Bittl, "Efficient construction of infinite length hash chains with perfect forward secrecy using two independent hash functions," in *Proc. 11th SECPYPT*, 2014, pp. 1-8.

[4] R. Kotla, A. Clement, E. Wong, L. Alvisi, and M. Dahlin, "Zyzzyva: Speculative byzantine fault tolerance," *CACM*, vol. 51, pp. 86-95, 2008.

[5] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative byzantine fault tolerance," in *Proc. SOSP'07*, 2007, pp. 45-58.

[6] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative byzantine fault tolerance," *ACM Trans. on Computer Systems*, vol. 27, no. 4, pp. 1-39, 2009.

[7] W. Thao, *Building Dependable Distributed Systems*, Wiley, 2014.

[8] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[9] S. Y. Yang and J. Kim, "Bitcoin market return and volatility forecasting using transaction network flow properties," in *Proc. IEEE Symposium Series on Computational Intelligence*, 2015, pp. 1778-1785.

[10] E. Anceaume, T. Lajoie-Mazenc, R. Ludinard, and B. Sericola, "Safety analysis of Bitcoin improvement proposals," in *Proc. IEEE 15th NCA*, 2016, pp. 318-325.

[11] G. Hurlburt, "Might the blockchain outlive bitcoin?" *IT Professional*, vol. 18, no. 2, pp. 12-16, 2016.

[12] S. Yunling and M. Xianghua, "An overview of incremental hash function based on pair block chaining," in *Proc. International Forum on Information Technology and Applications*, 2010, pp. 332-335.

[13] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *JACM*, vol. 27, no. 2, pp. 228-234, 1980.

[14] L. Lamport, R. Shostak, and M. Pease, "The Byzantine general problem," *ACM Transactions on Programming Language and Systems*, vol. 4, no. 3, pp. 382-401, 1982.

[15] Y. Matsumoto and H. Kobayashi, "A speculative Byzantine algorithm for P2P system," in *Proc. IEEE PRDC*, 2010, pp. 231-232.

[16] D. Wada, J. Kitagawa, and H. Kobayashi, "Online game protocol for P2P using Byzantine agreement," in *Proc. Second International Conference on Applications and Web Technologies*, 2009, pp. 780-785.

[17] SQLite Home Page. [Online]. Available: https://sqlite.org/

[18] MySQL-Official Site. [Online]. Available: http://www.mysql.com/

[19] Jetty Home Page. [Online]. Available: http://www.eclipse.org/jetty/

[20] D. Buruneo and S. Distefano, *Quantitative Assessments of Distributed Systems: Methodologies and Techniques*, Wiley, 2015.

Lei Tong received the B.Eng. degrees from Tokai University, Japan, in 2017. Currently, he is working toward the M.Eng. degree in the Graduate School of Information and Telecommunication Engineering at Tokai University. His research interests include software applications, database applications, and dependable computer systems.

Hiromi Kobayashi received the Dr. Eng. degree in systems engineering from Shinshu University, Japan, in1993. He previously works for several software companies. He joined the Department of Management Engineering at Tokai University in 1993 and joined the Department of Information Media Technology in 2001. He is currently a professor in the Department of Information Media Technology. He served as an associate editor in chief for the Journal of the Institute of Electronics, Information and Communication Engineers in Japan. His research interests include software specification methods, database applications, mobile phone systems, dependable computer systems, and distributed computer systems. He is a member of the IEEE Computer Society.