Effective Data Localization Using Consistent Hashing in Cloud Time-Series Databases

Jiakui Zhao, Pingfei Zhu
Beijing China-Power Information Technology Co., Ltd.
State Grid Electric Power Research Institute
Beijing, China
{zhaojiakui, zhupingfei}@sgepri.sgcc.com.cn

Liang Huai Yang
School of Computer Science and Technology
Zhejiang University of Technology
Hangzhou, China
yanglh@zjut.edu.cn

***Abstract*—The commercial time-series database is suitable for processing the time-series data. However, a single commercial time-series database can only accommodate the time-series data acquired by limited amount of sensors. In this paper, in order to cope with the challenge of massive time-series data processing, we first propose a cloud time-series database framework based on commercial time-series databases, and then propose an effective consistent hashing based algorithm for solving the key problem, i.e., the data localization problem, in cloud time-series databases. A performance study shows the superiority of the framework and the algorithm for processing massive time-series data acquired by large amount of sensors.**

Keywords-time-series data; massive data; data localization; consistent hashing; time-series database; cloud database; sensor.

I. INTRODUCTION

Nowadays, with continue advances of the information technology, especially the development of the Internet of Things (IoT) technology, large amount of sensors are used to acquire massive time-series data. For example, the electric power usage information acquisition system of State Grid Corporation of China (SGCC) utilizes 230 million smart meters to acquire the power usage information of all power users, where each smart meter has more than 10 sensors. The traditional commercial time-series database, e.g., the PI system, is suitable for processing the time-series data [1], [2]. However, a single commercial time-series database can only accommodate the time-series data acquired by limited amount of sensors, e.g., a PI system can accommodate the time-series data acquired by no more than 10 million sensors.

In this paper, in order to cope with the challenge of massive time-series data processing, we first propose a cloud time-series database framework based on commercial time-series databases, and then propose an effective consistent hashing [3], [4] based algorithm for solving the data localization problem which is the key problem in cloud time-series databases. A performance study shows the superiority of the framework and the algorithm for processing massive time-series data acquired by large amount of sensors. To the best of our knowledge, our work is the first that proposes the cloud time-series database framework and gives an effective solution for solving the key problem, i.e., the time-series data localization problem, in scalable cloud time-series databases.

The remainder of this paper is organized as follows: In the Related Work Section, we summarize the related work. In the Cloud Time-Series Database Framework Section, we introduce the cloud time-series database framework based on commercial time-series databases. In the Data Localization Algorithm Section, we introduce the consistent hashing based data localization algorithm. The Performance Study Section reports the result of an extensive performance study, followed by our conclusions in the last Conclusions Section.

II. RELATED WORK

The cloud database [5-20] utilizes the cloud computing technology to process data, and hence achieves very high performance and scalability. For example, the BigTable [8] cloud structured database proposed by Google is a NoSQL database based on the Google File System (GFS) [5], the MapReduce cloud computing framework [6], and the Chubby distributed lock service [7]. BigTable can process structured data with very high performance and scalability. However, the ACID (Atomicity, Consistency, Isolation, and Durability) properties of transactions cannot be guaranteed and the data cannot have the very complex relationships as that in traditional RDBMSs (Relational Data Base Systems).

The commercial time-series database has been proved to be suitable for processing the time-series data [1], [2]. However, to the best of our knowledge, the cloud time-series database which can process massive time-series data with very high performance and scalability has not been proposed.

III. THE CLOUD TIME-SERIES DATABASE FRAMEWORK

In commercial time-series databases, the time-series data is commonly organized as quadruples with the format of "<*id*, *timestamp*, *value*, *quality*>", where *id* is the identifier of a sensor, *timestamp* is the time when the value is acquired by a sensor, *value* is the acquired value, and *quality* is the quality of the acquired value. A non-SQL API is provided for processing the time-series data, including insertion, deletion, update and selection of the time-series quadruples.

The time-series data has a very good property, i.e., there is no inherent relationship between the quadruples generated by different sensors. Therefore, we may randomly distribute the quadruples generated by different sensors into different nodes where each node is a commercial time-series database. In this way, we solve the problem that a single commercial time-series database can only accommodate the time-series data acquired by limited amount of sensors. However, the distribution of the quadruples should be transparent to application developers, i.e., application developers should not be aware of the fragmentation of the time-series data and can only see an integrated view of the time-series quadruples.

According to above analysis, Figure 1 demonstrates the framework of a cloud time-series database which provides PaaS (Platform as a Service) level cloud service via a cloud proxy. The cloud proxy is an access point of the cloud time-series database, and the cloud time-series database may provide multiple access points in case of a single access point becomes the bottleneck. When a computer needs to access the time-series data, it invokes the API functions provided by the cloud proxy to perform the accessing, in which the identifier of the sensor which generates the time-series data, i.e., *id*, is always a parameter of the functions. After the cloud proxy receives the invocation, it utilizes a data localization algorithm to map *id* to *node* where the time-series data locates, and then the invocation is forwarded to the corresponding commercial time-series database. The invocation is processed by the commercial time-series database and the processing result is returned to the cloud proxy. Finally, the result is forwarded to the initial invoker.

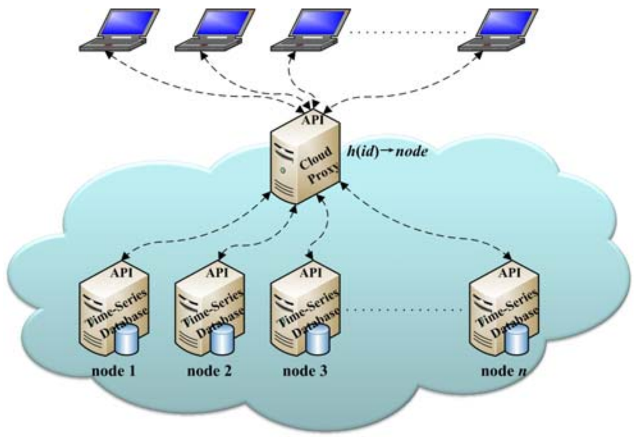

Figure 1. The cloud time-series database framework.

The structure of the time-series quadruples is very simple. However, the performance requirement for processing the time-series data is very high. The commercial time-series database has been proved to have very high performance, and hence the performance of the data localization algorithm determines the performance of the cloud time-series database.

IV. The Data Localization Algorithm

The data localization algorithm can be seen as a function with the form of "*h*(*id*)→*node*", where *id* and *node* are the identifier of a sensor and the position where the time-series data generated by the sensor locates, respectively. The performance of the data localization algorithm determines the performance of the cloud time-series database, and hence the performance of the algorithm should be enhanced as far as possible. We utilize consistent hashing [3], [4] to implement the data localization algorithm which maintains the mapping between *id*s and *node*s using a well designed "circle" instead of a 2-dimensional table, and the space and the time performance of the algorithm is very high. Figure 2 demonstrates how the data localization algorithm works over three nodes where each node is a commercial time-series databases. The circle denotes a 32-bit integer region where 0 and 2^{32}-1 are "connected" to form the circle. Each node is randomly distributed to the circle. The *id* of each sensor is hashed to a 32-bit integer using a base hash function with O(1) time complexity, and the first node clockwise from the 32-bit integer denotes the commercial time-series databases where the timer-series data generated by the sensor locates.

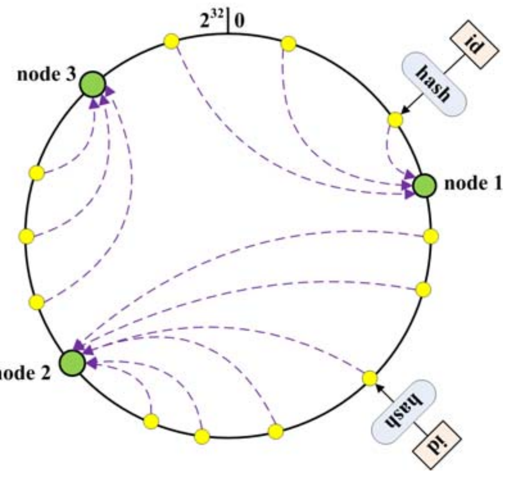

Figure 2. An exampe of data localization using consistent hashing.

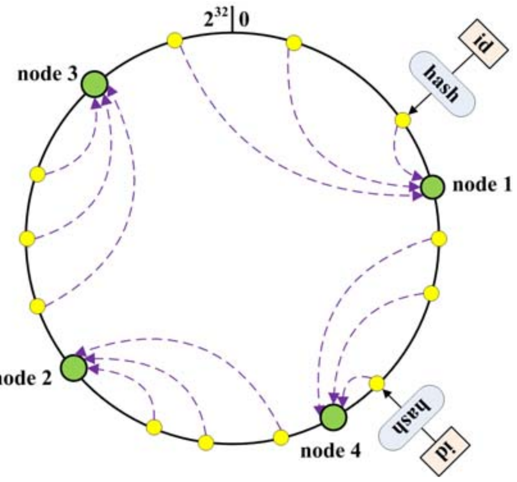

Figure 3. An exampe of adding a new node to the cloud framework.

Figure 3 demonstrates how to add a new node to the "circle" as shown in Figure 2. The new node, i.e., node 4, is randomly distributed to the circle, and the time-series data generated by three sensors which locates at node 2 should be moved to node 4, for node 4 becomes the clockwise nearest node. Compared with the data localization algorithms which maintain the mapping between *id*s and *node*s using a 2-dimensional table, the consistent hashing based algorithm has a drawback, i.e., in the case of the number of the nodes changes, some data needs to move among nodes. However, compared with other hashing based data localization algorithms, the consistent hashing based algorithm only redistributes a small percent of the data. As Equation 1 shows, when the *n*th commercial time-series database is added to the cloud framework, the average percent of the data which should be redistributed, i.e., *MovePercent* (*n*), is only 1/*n*. Compared with the data localization algorithms which maintain the mapping between *id*s and *node*s using a 2-dimensional table, the hashing based algorithms achieve very high space and time performance, which is very important for implementing the cloud time-series database.

$$MovePercent\,(n) = 1/n. \qquad (1)$$

In order that the time-series data can be uniformly distributed over nodes, we may add some virtual nodes to the "circle". Figure 4 demonstrates how the data localization algorithm using consistent hashing with virtual nodes works. Compared with Figure 1, Figure 4 adds two virtual nodes, i.e., the nodes with dotted lines, for each real node, and the data is distributed to the clockwise nearest real/virtual node. In this way, the data can be distributed uniformly over the nodes. For example, in Figure 1, the three nodes contains the time-series data generated by three sensors, six sensors, and three sensors, respectively; however, in Figure 3, each sensor just contains the time-series data generated by four sensors.

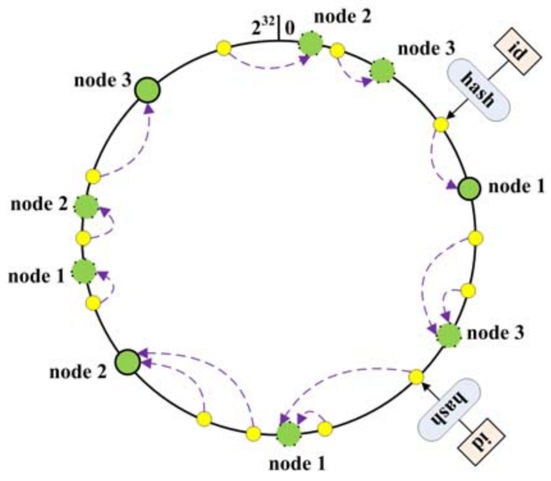

Figure 4. An exampe of data localization using consistent hashing with virtual nodes.

According to above demonstrations in this section, the consistent hashing based data localization algorithm is straightforward, and we will omit the details of the algorithm.

V. A Performance Study

In this section, we evaluate the performance of the cloud time-series database framework and the time-series data location algorithm proposed in this paper by experiments. All the experiments are run on personal computers connected by local area network, each of which has a 2.67GHz Intel Pentium CPU and 2GB of physical memory. One personal computer serves as the cloud proxy and other personal computers serve as commercial time-series databases. The operating system used is Windows 7. The time-series database used is the STR system developed by State Grid Electric Power Research Institute of China, which can accommodate the time-series data acquired by no more than 1 million sensors. The cloud time-series database framework as well as the data localization algorithm is implemented by the C++ programming language under the Microsoft Visual Studio 2008 platform. The dataset used is a real-life dataset exported from the Energy Management System (EMS) of a provincial electric power company of SGCC, which contains the time-series data generated by 10 million sensors.

The base hash function is implemented using a SHA-1 (Secure Hash Standard) algorithm with time complexity of O(1), and the "circle" is implemented using a typical list. In the case of one of the STR systems accommodates the time-series data generated by more than 0.6 million sensors, we will add a new STR system to the cloud time-series database. We add 9 virtual nodes for each real node, and the final real node and virtual node numbers are 18 and 162, respectively.

According to above configuration, the space complexity of the data localization algorithm is O(*n*) where *n* is the number of the nodes, including real nodes and virtual nodes. Compared with the data localization algorithms which maintain the mapping between *id*s and *node*s using a 2-dimensional table, the space complexity may be neglected. Given the *id* of a sensor, the time complexity of the SHA-1 algorithm is O(1), and the time complexity of searching the corresponding node in the node list is O(lg*n*) in the case of using binary search. Therefore, the time complexity of the data localization algorithm is O(lg*n*) where *n* is the number of the nodes, including real nodes and virtual nodes.

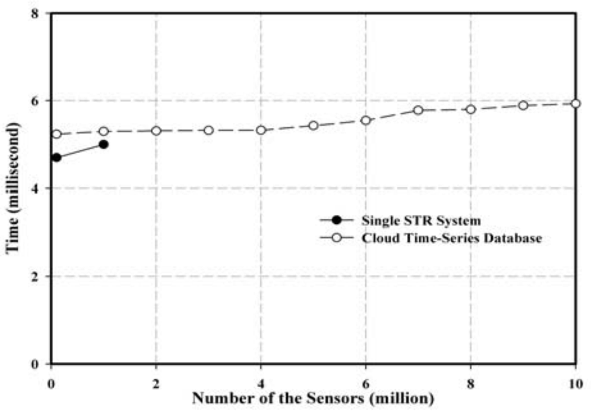

Figure 5. Time performance comparision.

Figure 5 shows the time performance comparison result between a single STR system and a cloud time-series database system based on the STR system, where the time is measured by an API invocation which accesses the historical data generated by a randomly selected sensor at a specified time. As Figure 5 shows, in the case of the number of the sensors is no more than 1 million, the single STR system and the cloud time-series database system all have very high time performance, and the time performance of the cloud time-series database is a little lower than the single STR system due to existence of the cloud proxy. In the case of the number of the sensors is more than 1 million, the single STR system outages due to sensor number limitation, but the cloud time-series database still has a good time performance.

According to above study, the proposed cloud time-series database framework and the consistent hashing based data localization algorithm have good performance and scalability.

VI. CONCLUSIONS

In this paper, in order to cope with the challenge of massive time-series data processing, we propose a cloud time-series databases framework based on commercial time-series databases as well as an effective consistent hashing based algorithm for solving the data localization problem which is the key problem in cloud time-series databases. An extensive performance study shows the superiority of the proposed framework and algorithm for processing massive time-series data acquired by large amount of smart sensors.

ACKNOWLEDGMENT

This work was supported by the Science and Technology Project of SGEPRI (State Grid Electric Power Research Institute) entitled “Research on the Information Processing and Typical Applications for Electric Internet of Things”, the Science and Technology Projects of State Grid Corporation of China entitled “Research on the Integrated Supporting Technologies for Intelligent Marketing Business based on the International IEC-CIM/CIS Standard” and “Research on the Technologies of Analysis and Pre-Warning for Company Operation”, the NSFC under Grant No. 61070042, and the NSF of Zhejiang Province under Grant No. Y1090096.

REFERENCES

[1] G. Chen and L. Li, “An Optimized Algorithm for Lossy Compression of Real-Time Data”, Proceedings of the 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS 10), IEEE Press, Dec. 2010, pp. 187–191.

[2] A. Singhal and D. E. Seborg, “Effect of Data Compression on Pattern Matching in Historical Data”, Industrial & Engineering Chemistry Research, vol. 44, Mar. 2005, pp. 3203–3212.

[3] David R. Karger, Eric Lehman, Frank Thomson Leighton, Rina Panigrahy, Matthew S. Levine, and Daniel Lewin, “Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web”, Proceedings of the 29th ACM Symposium on the Theory of Computing (STOC 97), ACM Press, May 1997, pp. 654-663.

[4] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”, Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM 01), ACM Press, Aug. 2001, pp. 149-160.

[5] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung, “The Google File System”, Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP 03), ACM Press, Oct. 2003, pp. 29-43.

[6] Jeffrey Dean and Sanjay Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, Proceedings of the 6th USENIX Symposium on Operating System Design and Implementation (OSDI 04), USENIX Association, Dec. 2004, pp. 137-150.

[7] Michael Burrows, “The Chubby Lock Service for Loosely-Coupled Distributed Systems”, Proceedings of the 7th USENIX Symposium on Operating System Design and Implementation (OSDI 06), USENIX Association, Nov. 2006, pp. 335-350.

[8] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Michael Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber, “Bigtable: A Distributed Storage System for Structured Data”, Proceedings of the 7th USENIX Symposium on Operating System Design and Implementation (OSDI 06), USENIX Association, Nov. 2006, pp. 205-218.

[9] Tengjiao Wang, Bishan Yang, Allen Huang, Qi Zhang, Jun Gao, Dongqing Yang, Shiwei Tang, and Jinzhong Niu, “Dynamic Data Migration Policies for Query-Intensive Distributed Data Environments”, Proceedings of the Joint Conference of the 11th Asia-Pacific Web Conference and the 10th International Conference on Web-Age Information Management (APWeb/WAIM 09), Apr. 2009, pp. 63-75.

[10] Edward P. Holden, Jai W. Kang, Dianne P. Bills, and Mukhtar Ilyassov, “Databases in the Cloud: A Work in Progress”, Proceedings of the 10th ACM SIGITE International Conference on Information Technology Education (SIGITE 09), ACM Press, Oct. 2009, pp. 138-143.

[11] Ashraf Aboulnaga, Kenneth Salem, Ahmed A. Soror, Umar Farooq Minhas, Peter Kokosielis, and Sunil Kamath, “Deploying Database Appliances in the Cloud”, IEEE Data Engineering Bulletin, vol. 32, Mar. 2009, pp. 13-20.

[12] Daniel Abadi, Michael J. Carey, Surajit Chaudhuri, Hector Garcia-Molina, Jignesh M. Patel, and Raghu Ramakrishnan, “Cloud Databases: What's New?”, Proceedings of Very Large Data Base, vol. 3, Sep. 2010, p. 1657.

[13] Chun Chen, Gang Chen, Dawei Jiang, Beng Chin Ooi, Hoang Tam Vo, Sai Wu, and Quanqing Xu, “Providing Scalable Database Services on the Cloud”, Proceedings of the 11th International Conference on Web Information Systems Engineering (WISE 10), Springer, Dec. 2010. pp. 1-19.

[14] Edward P. Holden, Jai W. Kang, Geoffrey R. Anderson, and Dianne P. Bills, “Databases in the Cloud: A Status Report”, Proceedings of the 12th ACM SIGITE International Conference on Information Technology Education (SIGITE 11), ACM Press, Oct. 2011, pp. 171-176.

[15] Magdalena Balazinska, Bill Howe, and Dan Suciu, “Data Markets in the Cloud: An Opportunity for the Database Community”, Proceedings of Very Large Data Bases, vol. 4, Oct. 2011, pp. 1482-1485.

[16] Maximilian Ahrens and Gustavo Alonso, “Relational Databases, Virtualization, and the Cloud”, Proceedings of the 27th International Conference on Data Engineering (ICDE 11), IEEE Press, Apr. 2011, p. 1254.

[17] PengCheng Xiong, Yun Chi, Shenghuo Zhu, Hyun Jin Moon, Calton Pu, and Hakan Hacigümüs, “Intelligent Management of Virtualized Resources for Database Systems in Cloud Environment”, Proceedings of the 27th International Conference on Data Engineering (ICDE 11), IEEE Press, Apr. 2011, pp. 87-98.

[18] Hoang Tam Vo, Sheng Wang, Divyakant Agrawal, Gang Chen, and Beng Chin Ooi, “LogBase: A Scalable Log-structured Database System in the Cloud”, Proceedings of Very Large Data Bases, vol. 5, Jun. 2012, pp. 1004-1015.

[19] Chao-Rui Chang, Meng-Ju Hsieh, Jan-Jan Wu, Po-Yen Wu, and Pangfeng Liu, “HSQL: A Highly Scalable Cloud Database for Multi-user Query Processing”, Proceedings of the 5th IEEE International Conference on Cloud Computing (CLOUD 12), IEEE Press, Jun. 2012, pp. 943-944.

[20] Carlo Curino, Evan P. C. Jones, Raluca A. Popa, Nirmesh Malviya, Eugene Wu, Samuel Madden, Hari Balakrishnan, and Nickolai Zeldovich, “Relational Cloud: A Database Service for the Cloud”, Proceedings of the 5th Biennial Conference on Innovative Data Systems Research (CIDR 11), Jan. 2011, pp. 235-240.