

Distributed Storage System for Electric Power Data Based on HBase

Jiahui Jin, Aibo Song*, Huan Gong, Yingying Xue, Mingyang Du, Fang Dong, and Junzhou Luo

Abstract: Managing massive electric power data is a typical big data application because electric power systems generate millions or billions of status, debugging, and error records every single day. To guarantee the safety and sustainability of electric power systems, massive electric power data need to be processed and analyzed quickly to make real-time decisions. Traditional solutions typically use relational databases to manage electric power data. However, relational databases cannot efficiently process and analyze massive electric power data when the data size increases significantly. In this paper, we show how electric power data can be managed by using HBase, a distributed database maintained by Apache. Our system consists of clients, HBase database, status monitors, data migration modules, and data fragmentation modules. We evaluate the performance of our system through a series of experiments. We also show how HBase's parameters can be tuned to improve the efficiency of our system.

Key words: electric power data; HBase; data storage

1 Introduction

Electric power systems are essential to modern society. As a core subsystem, the Power Dispatching Automation System (PDAS) processes runtime information and makes real-time control decisions, which guarantees the safety and substantiality of electric power systems^[1]. With the increasing scale of PDAS, system-generated data, including status, debugging, and errors, have increased dramatically in recent years. For example, a city-tier PDAS generates millions of records every day, while a province-tier PDAS needs to collect data from tens of city-tier PDASs, and make global decisions in real time.

The electric power data processed by PDAS are typical 4Vs data (data with characteristics of volume,

variety, velocity, and veracity), which are difficult to process, query, and analyze within a tolerable time. Recently, the fast development of electric power systems has significantly increased the size of electric power data. According to a report from IBM (https://www-935.ibm.com/services/multimedia/Managing_big_data_for_smart_grids_and_smart_meters.pdf), “going from one meter reading a month to smart meter readings every 15 minutes works out to 96 million reads per day for every million meters. The result is a 3000-fold increase in data that can be overwhelming if not properly managed.” Traditionally, electric power data are stored in relational databases, where maintenance costs can explode due to the increasing data sizes and requirements of real-time processing. To address this problem, companies like IBM, Oracle, and General Electric, have brought their big-data projects to the power industry. Scientists have also proposed a series of techniques including optimization and data mining to efficiently analyze, process, and visualize electric power data.

Cloud computing is one of the key solutions to process massive electric power data^[2, 3]. With cloud computing, distributed file systems and data management technologies such as Google File

• Jiahui Jin, Aibo Song, Huan Gong, Yingying Xue, Mingyang Du, Fang Dong, and Junzhou Luo are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China. E-mail: {jjin, absong, 220151478, y-xue, 220161558, fdong, jluo}@seu.edu.cn.

*To whom correspondence should be addressed.

Manuscript received: 2017-08-20; accepted: 2018-03-26

System^[4], Hadoop Distributed File System (HDFS)^[5], BigTable^[6], and Apache HBase^[7] can be used to store large amounts of electric power data. Parallel processing technologies, such as MapReduce^[8] and Spark^[9, 10], make real-time processing of electric power data possible.

On the basis of cloud computing, many emerging scalable and distributed architectures and frameworks have been proposed in the power grid area. The Tennessee Valley Authority (<http://www.tva.gov/>) built a power grid information processing architecture using MapReduce and Hadoop to detect power grid anomalies, creating power grid maps and evaluating power consumption history. In Japan, Kyushu Electric Power Company has developed a big data platform to meet the requirements of rapid analysis of vast amounts of power consumption data collected from residences, offices, factories, and other sectors in a power grid^[11]. This big data platform is based on a Hadoop cluster and is deployed in a cloud computing environment that utilizes server visualization technology. OpenPDC (<https://github.com/GridProtectionAlliance/openPDC>) is an open-source project based on Hadoop, which contains a set of applications for processing streaming time-series data from power management units. Overall, most existing information systems are based on Hadoop for batch processing electric power data offline.

However, as a batch processing system, Hadoop would not work for real-time queries. To address this problem, we propose a big data platform that uses Apache HBase distributed database to store and query electric power data. HBase is an open source, non-relational, distributed database running on top of HDFS and providing BigTable-like capabilities for Hadoop. Unlike Hadoop, HBase is well-suited for faster read and write operations on large datasets with high throughput and low input/output latency (https://en.wikipedia.org/wiki/Apache_HBase).

This paper is organized as follows: We first introduce the basic concepts of HBase in Section 2. We then show the architecture of our system in Section 3 and we present the data storage model of electric power data when considering the characteristics of power system data in Section 4. In Section 5, we evaluate the factors that affect HBase performance and propose optimization techniques for tuning HBase for electric power data. We show the related works in Section 6

and conclude the paper in Section 7.

2 Apache HBase

Apache HBase (short for HBase) is a distributed database based on Google’s BigTable, which is the storage layer of the Hadoop ecosystem. HBase takes advantage of HDFS as the underlying file system and the distributed programming framework MapReduce as its implementation framework^[12].

The basic unit of the HBase data table is column family, which consists of one or more columns. Figure 1 shows a column family, namely, *bill*, which contains columns *total* and *balance*. The row key identifies the data for each row and is used as a primary key for retrieving records in HBase. In physical storage, each column family corresponds to a file. The columns that are visited frequently are placed in the same column family to reduce query time when related data are in the column family. In addition, the data columns in a column family can be dynamically updated according to the needs of applications.

The architecture of HBase follows a master-slave structure, which consists of an HMaster and multiple HRegionServers (see Fig. 2). HMaster is a cluster manager which is to operate the data table and balance the loads of the HRegionServers. In a distributed database, the data is distributed over multiple storage

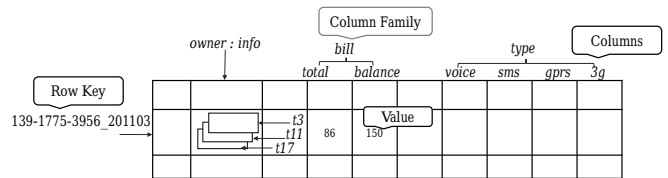


Fig. 1 HBase storage model.

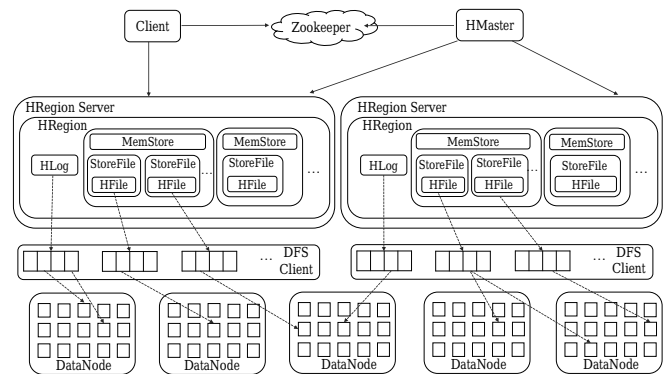


Fig. 2 Architecture of HBase.

servers. HRegion is a HBase database data management unit. A Column Family within HRegion corresponds to a Store instance. In the current HBase design, each Store instance can, in turn, have one or more StoreFile instances, which are lightweight wrappers around the actual storage file called HFile. A Store also has a MemStore which represents the memory cache.

3 Our System Architecture

Our system manages the data that come from PDAS, which consist of many clients and a data processing platform (or platform for short). Figure 3 depicts the design of our system. PDASs can upload data through clients, then the servers use platform to store and analyze data.

Data Format. Different PDASs use different formats to store data. Thus, exchanging data among PDASs and our storage system is challenging, thereby reducing the efficiency of data processing and data collection. To solve the data exchange problem, we use CIM/E, a standard based on the Common Information Model (CIM) (<http://www.dmtf.org/standards/cim>), to describe power grid model data. CIM is a standard developed by the electric power industry, which has been officially adopted by the International Electrotechnical Commission to allow application software to exchange information about an electrical

network. CIM/E is an extension of CIM and is designed for more efficient data exchange. Each PDAS describes model data and runtime data as CIM/E files and then posts these files to the storage system. When querying the data, our system will parse the queried data into CIM/E files and then send them to PDASs. In such a way, we solve the data exchange problem, and allow data sharing among PDASs and our storage system.

Client. Two kinds of clients exist: storage clients and query clients. After collecting data from PDAS, the storage clients parse the data into data records and then call the platform-side storage interface to store the data. Query clients are responsible for answering the queries that are submitted by PDASs and the power system analysts. To answer a query, the query clients search for data through the platform-side interfaces, convert the platform-returned data as a CIM/E file, and then send the CIM/E file to users. Clients are deployed in power plants, city-tier PDASs, or province-tier PDASs.

Platform. The platform is the core part of our system and is responsible for storing and analyzing data. The platform consists of database, status monitor, data migration, data fragmentation, and other modules, which are introduced in the following.

- *Database.* The database is responsible for storing and managing data. We use HBase to store and manage these data. When storing power data, the data should be split and stored across different computers. However, improper data fragmentation may increase the overhead of querying. Our data fragmentation strategy takes the graph partitions of the power grid network into account.
- *Status monitor.* The status monitor collects real-time CPU, memory, network, and disk I/O information of the servers in our platform, and sends the data to the workload balance controller. The traditional systems typically use open-source monitor software to acquire the status information^[13], but the open-source software itself may introduce considerable running overhead. Instead of using the open-source softwares, our status monitor collect the server status by calling the operating system APIs.
- *Workload balance controller.* To determine the time to migrate data, the workload balance controller detects each server's collected status. Our system adopts a straightforward approach based on threshold detection method according to

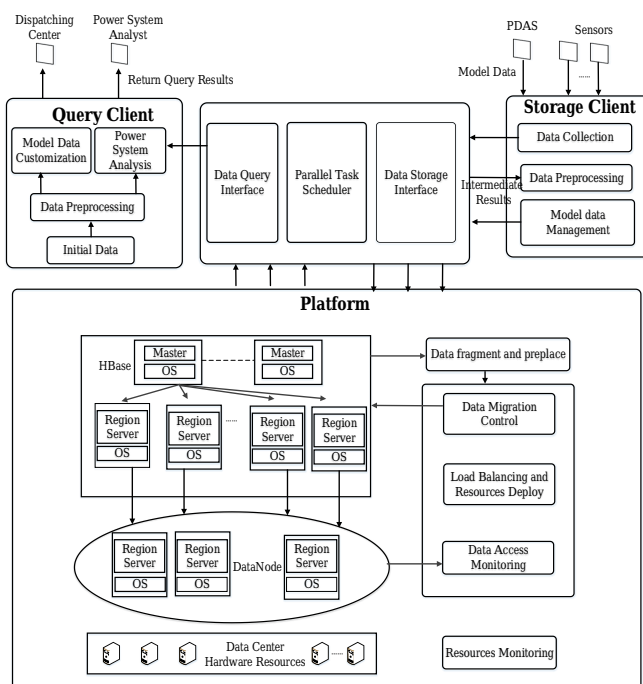


Fig. 3 System design.

the extent of workload unbalance. When the extent of workload unbalance reaches the threshold, data migration will start.

- *Data migration.* When the servers' workloads are unbalanced, the data migration module generates a migration plan based on the server status. It then transfers data among servers according to the migration plan. The traditional data migration strategies are time-costly, because they need to redistribute all the data. To address this problem, our system first calculates the number of data fragments that need to be migrated and then generates a corresponding data migration plan.

4 Managing Electric Power Data on HBase

The electric power data contain the model data and the runtime data. The model data are generated by PDAS and the runtime data are collected by sensors. This subsection first introduces how the data are generated and then show how the data are stored on HBase.

The model data include region, reference voltage, plant station, and voltage level, each of which contains identification, location, class, impedance, reactance, and other attributes. The model data are generated by the real-time system and future system in PDAS. The real-time system stores each power equipment's current running status, and the future system stores each equipment's future operating parameters. Figure 4 shows the process of generating model data. When predicting the future parameters, PDAS will obtain the latest real-time model from a future system, and import the model into a real-time system in an incremental manner. Afterwards, PDAS generates, stores, and publishes a new model. The original model will be stored as the historical model in the database. As soon as the new model has been published, PDAS sends the published incremental model data to the storage system.

The runtime data include running parameters of all kinds of power grid equipment such as breakers, knife gates, and alternators, each of which contains electric

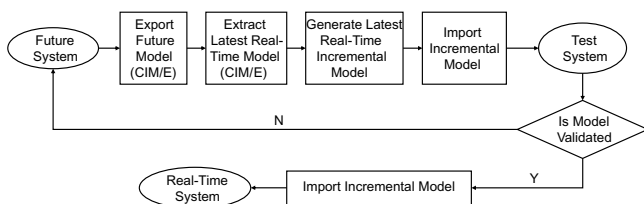


Fig. 4 Process of generating model data.

current, voltage, active power, and inactive power. The runtime data are generated by the sensors in the power grid. The sensors obtain runtime data from equipment, and then send the data to the database of PDAS in near-real time. The transformer substation is an example that illustrates the process of collecting the runtime data, which is shown in Fig. 5.

Our system uses HBase to store the model data and the runtime data. In Fig. 6a, we define the mapping rules that migrate the model data from the relational database (see Fig. 7) to HBase according to the structure features of model data, access patterns and compatibilities. The runtime data are generated in an incremental manner, which has features of low value density and large quantity. Figure 6b shows how the runtime data are stored in HBase. Three column families are present in the runtime data table. The first column family is used to store the values of electric current and voltage. The second family is used to store the values of active power and inactive power. The last one stores the status of the equipment.

5 Evaluation

5.1 System settings

We evaluate our system on a computer cluster. The cluster contains 15 Dawning CB60 servers with 2.60 GHz Intel® Xeon® E5-2670 CPU, 32 GB memory, and 300 GB SAS disk, which are connected by gigabit switches. Figure 8 shows how our system is deployed on the servers. We deploy Red Hat Linux 6.2 operating system, Hadoop 1.0.4, and HBase 0.94.6.1 on the servers. We use two servers to deploy the management

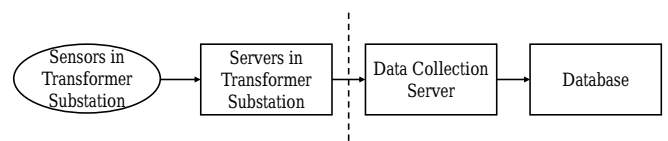
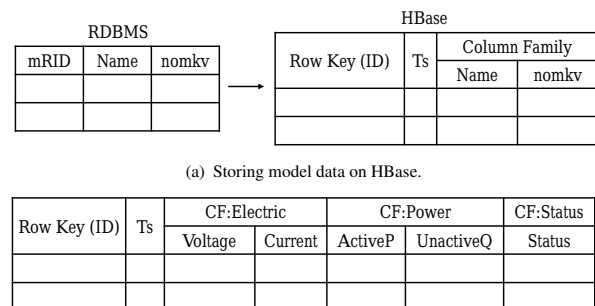


Fig. 5 Process of generating runtime data.



(b) Storing runtime data on HBase.

Fig. 6 Storing electric power data on HBase.

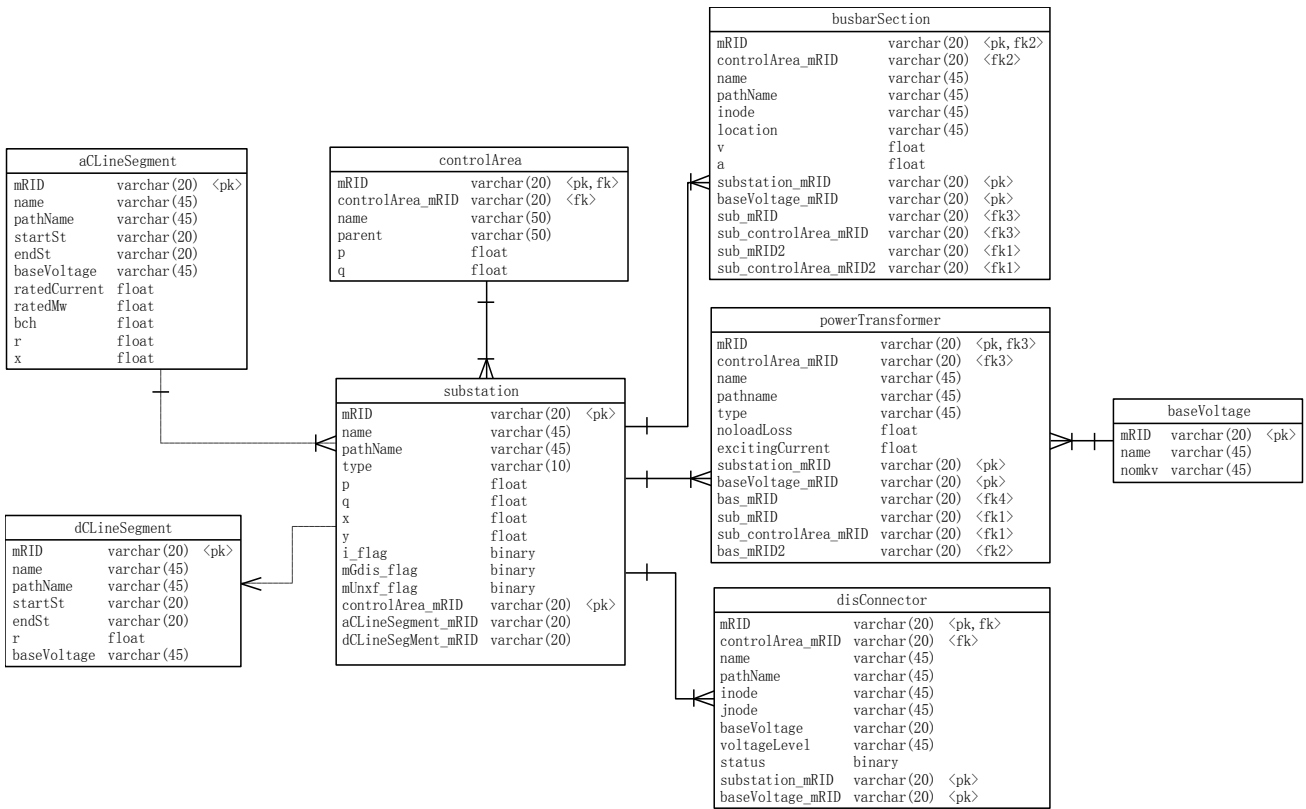


Fig. 7 Schema of model data in relational database.

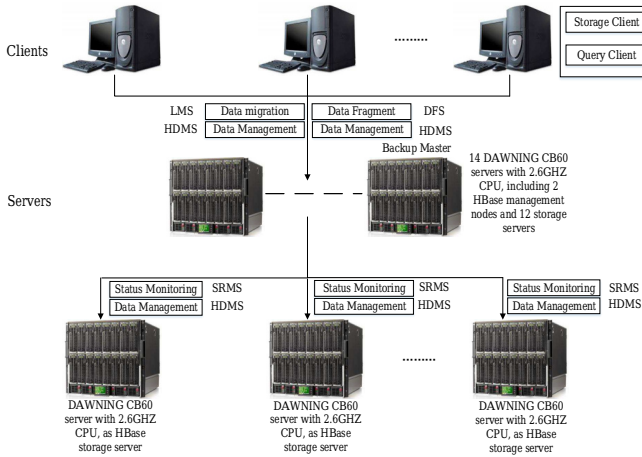


Fig. 8 Deployment of our system.

nodes, Namenode, JobTracker, and HMaster. We also deploy the data migration and data fragment storage module on the two servers. The remaining 13 servers are used as storage nodes, where we deploy DataNode, TaskTracker, and HRegionServer services. The running status of these servers is monitored by status monitor module.

5.2 Comparison with existing system

We compare our system with a MySQL-based PDAS data storage system on the 15-server cluster. In our experiment, we use eight electric power datasets whose sizes vary from 1 million tuples to 128 million tuples. We also perform a query that selects the error information of station 1 issued in January 2016 (select * from net_gk where station_id=1 and time between “2016-01-01 00:00:00” and “2016-01-31 23:59:59”) on the datasets, and measure the query response time of our system and the MySQL-based system. From Fig. 9, we can see that our system

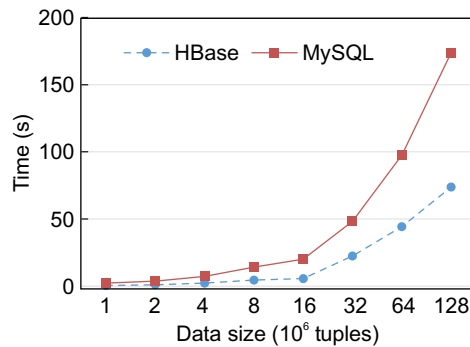


Fig. 9 Comparison with MySQL-based system.

is four times faster than the MySQL-based system (0.53 s vs 2.31 s) on the 1-million-tuple dataset, and 2.3 times faster than the MySQL-based system (73.82 s vs 173.86 s) on the 128-million-tuple dataset. Our system stores the error information in a column family; thus, it can efficiently respond to a query. The performance gap between our system and the MySQL-based system is smaller when the dataset is larger. When the dataset is larger, the data will be distributed on more servers, so the communication overhead cannot be ignored. Nevertheless, our system is faster than the existing MySQL-based system when we perform queries on large datasets.

5.3 Performance of data storage

We evaluate the running time for storing the model data on the storage system by using the collected station model, the PDAS model, and the business model. Figure 10 shows the data format. In our experiment, the size of the data set is about 200 MB. We write the data set to HBase through the interfaces provided by HBase. For each of the station model, the PDAS model, and the business model, we perform 10 tests and measure the writing time of each test. Figure 11 shows the results. In the power system, the time used to write all of the models of one station is less than 30 s. The average time used to write the station models and a full model is about 19 s and 38 s, respectively. The time used to publish business models is less than 35 s.

5.4 Effects of data fragmentation

The whole power grid network model is large; thus it needs to be fragmented. We adopt a data fragmentation strategy based on graph partitioning^[14]. Data are divided into multiple data blocks and placed randomly into multiple storage nodes. Figure 12 shows the results of partitioning the IEEE300 power grid network (<http://www.ee.washington.edu/research/pstca/index.htm>). In HBase, the data from the same sub-network are

```
<! Entity= Fuzhou type=power grid model time= '2012-08-10 11:18:48' >
<ControlArea::Fuzhou>
@Num mRID name Parent p q
#1 113715891 Fuzhou power grid NULL NULL NULL
#2 113715892 virtual area 1137158922 NULL NULL
</ControlArea::Fuzhou>
<BaseVoltage::Fuzhou>
@Num mRID name nom KV
#1 112871873 220kv 230
#2 ...
</BaseVoltage::Fuzhou>
.....
```

Fig. 10 Format of model data.

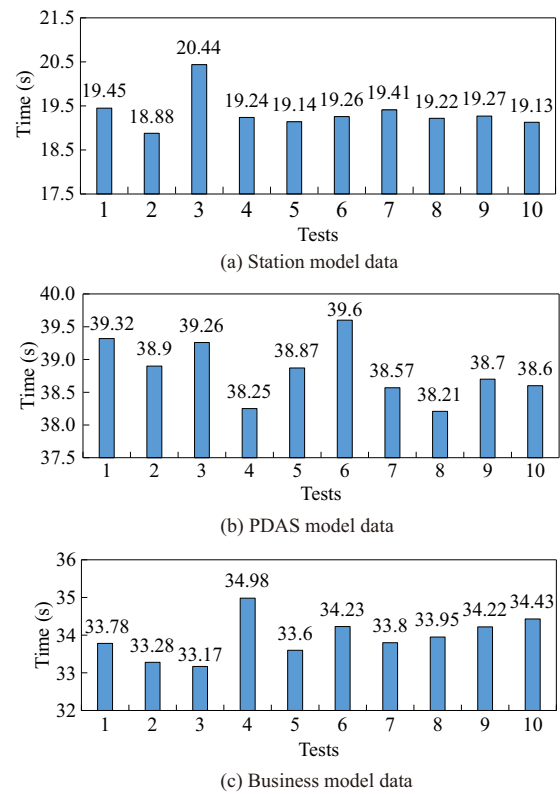


Fig. 11 Time for writing station model data.

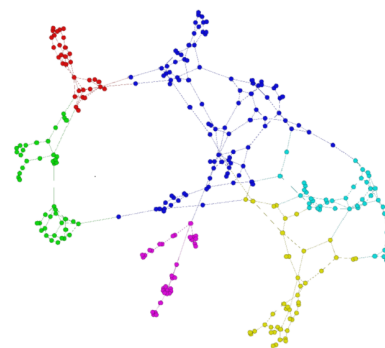


Fig. 12 Partitioning power grid network.

stored in the same data fragment. The row key of a data fragment is designed as regionkey + name + ID + timestamp.

We partition 10 GB data that are generated according to the model data format and write it to the storage system. Then we query the data of the power-generator stations within the same sub-network. The average querying time of the two types of fragmentation strategies is shown in Fig. 13. Experimental results show that the average querying time of random fragmentation is much longer than that of graph-partition based fragmentation. The random fragmentation strategy distributes data randomly to

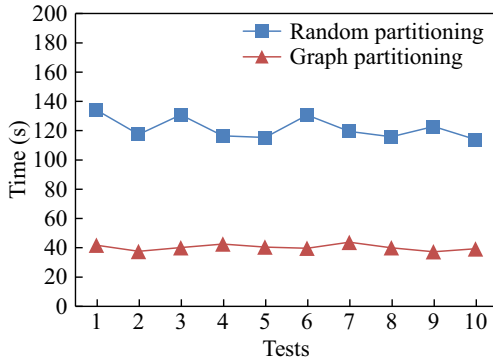


Fig. 13 Effects of two fragmentation strategies on querying time.

multiple nodes, causing multiple nodes that need to be queried, while the graph partition-based fragmentation strategy reduces querying time greatly by storing the data of each sub-network in the same data block.

5.5 Effects of data migration

Data migration affects system performance because the power grid data are divided into multiple data blocks and randomly placed in multiple servers. Figure 14a shows the effect of data migration on the performance of the system. The running time is reduced significantly after data migration, thereby illustrating the effectiveness of data migration. As shown in Fig. 14b, the load of storage server 3 is higher than that of other storage servers before data migration. After data migration, the load of each storage server

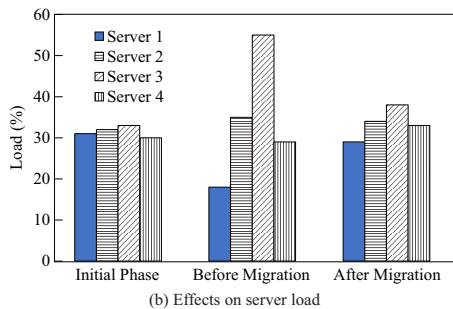
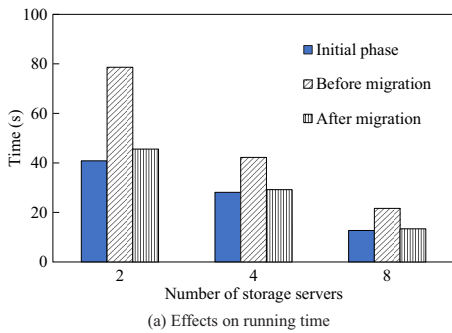


Fig. 14 Effects of data migration.

is basically the same. Experimental results show that data migration strategy can balance the loads of storage servers and improve the efficiency of data management.

5.6 Scalability

We first increase the number of storage servers in the cluster to evaluate the performance of our system when the server number scales. Then, we increase the number of write clients to evaluate the performance of our system when the client number scales.

The experimental results shown in Fig. 15 illustrate that the storage performance is significantly improved when the number of storage servers increases. The running time is reduced by half when there are three storage servers. However, when the number of servers increases further, the performance improves slightly due to the server communication. Furthermore, the number of clients has a great impact on system performance. Given the advantage of the distributed storage architecture, the write speed improves significantly when the number of clients increases.

5.7 Effects of HBase parameters

We evaluate how the HBase parameters affect the performance of HBase. Here we show the effects of Java Virtual Machine (JVM) heap size, HBase region size, HBase cache size, and automatic flushing function.

JVM heap size. HBase is developed based on Java language; thus the performance of HBase is affected by JVM heap size. We evaluate the read-write performance of HBase with different heap sizes. As shown in Fig. 16, when the amount of data increases, (1) the throughput first increases, but then decreases, and (2) the read-write delay first decreases, but then increases.

HBase region size. In Fig. 17, we show the read-write performance of different region sizes when

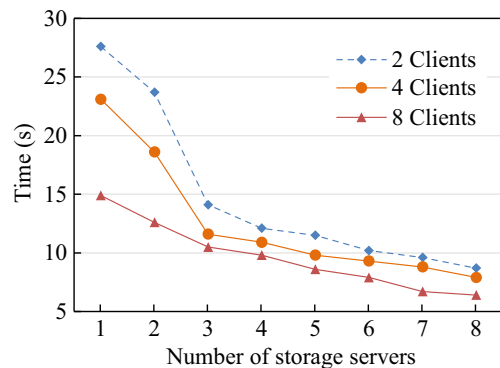
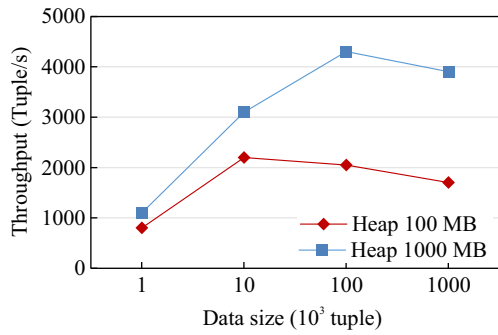
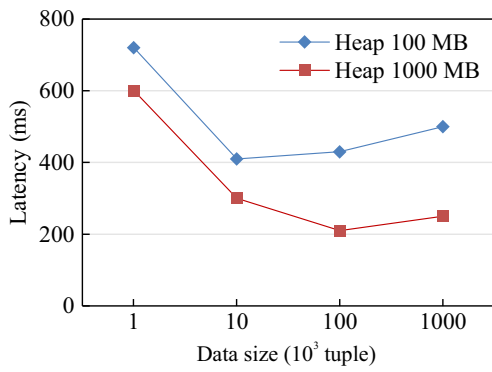


Fig. 15 Scalability.

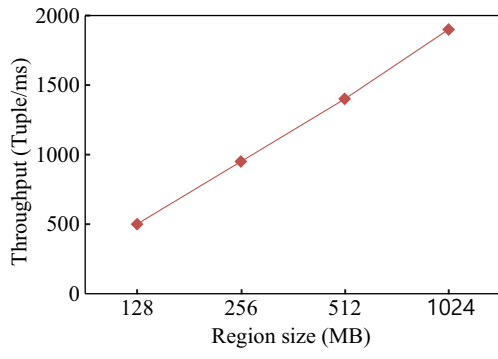


(a) Throughput

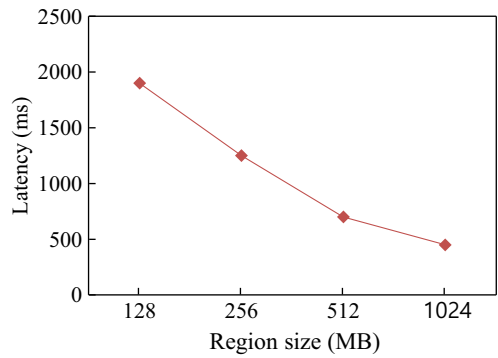


(b) Latency

Fig. 16 Effects of heap size.



(a) Throughput



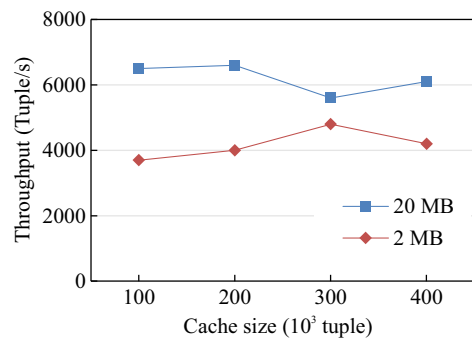
(b) Latency

Fig. 17 Effects of region sizes.

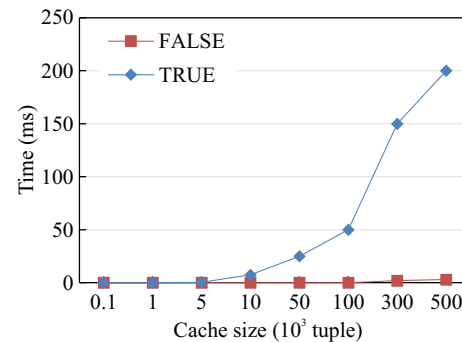
reading and writing 100 000 tuples. When the region size increases, the throughput increases and the read-write delay decreases. The reason for this situation is that our systems will generate more data fragments when the region size is smaller. A large number of data fragments will introduce a large overhead.

Cache size and data flushing: After the client sends the request to write data, RegionServer finds the corresponding region and writes the data to the HLog. HBase caches this part of data in MemStore, and writes the data into HDFS’s HFile when MemStore is full. When the data in MemStore is lost, HBase uses HLog to complete the data recovery. If not enough space is left for new data, JVM will recover heap space and compress the data. Therefore, caching plays an important role in reducing heap space compression.

Figure 18a shows the read-write performance of HBase with different cache sizes. We can see that a large cache can lead to a higher throughput and a lower read-write delay because when the cache is larger, HBase flushes data to the disk less frequently, thereby greatly reducing the cost of reading and writing. Figure 18b shows that the performance of HBase is greatly improved when the automatic flushing function is turned off. Therefore, a large cache size, as well



(a) Throughput when using different cache sizes



(b) Running time when autoflushing is turned on or off

Fig. 18 Effects of cache size and automatic flushing.

as turning off the auto flush, will enhance the HBase performance and improve the query efficiency.

6 Related Work

With the fast development of electric power systems, the size of generated electric power data has increased dramatically^[2]. Conventional technologies store massive electric power data on traditional relational database, but report generation and analytic can become painfully slow due to high volumes of data. To support fast decision making, many organizations and companies, such as the Tennessee Valley Authority (<http://www.tva.gov/>), Kyushu Electric Power Company^[11], IBM, and General Electric, have proposed big-data solutions in the power grid area. Furthermore, scientists have proposed a variety of techniques, including big data processing, data mining, and security protection, to store, analyze, and protect massive data.

Many big data processing frameworks are proposed to support the massive electric power data. Lyu et al.^[15] addressed a load forecasting problem in a smart grid by using a massive data storage platform. On the platform, the electric power data are described by CIM/XML and stored on HBase, which is similar to our solution. However, the goal of our system is different; we focus on PDAS systems rather than load forecasting. We have systematically shown the design and performance of our system. Rusitschka et al.^[16] proposed a distributed data processing framework for geodistributed smart grid meters, leveraging distributed data management for real-time data gathering, parallel processing for real-time information retrieval, and ubiquitous access. Medjroubiet al.^[17] proposed a novel power grid model based on open and publicly available data from an online map, namely OpenStreetMap, using open source software tools. This model can be applied to our system to enhance the schema of electric power data by using accurate geo-information. Meier et al.^[18] proposed a data processing system for synchrophasor data, which processes high-granularity and high-cardinality data gathered from synchrophasor sensors, i.e., Phasor Measurement Units (PMUs), using a correlation method. The deployment of PMUs could improve the measurements of voltages and currents with accurate timestamps^[19]. Thus, the use of PMU data is helpful for more accurate measurement^[2]. However, this system uses the common correlation

method, which is suitable only for data with low diversity and volume^[20, 21]. For the data that could not be correlated in space and time, correlating to a unified and generalized power system model is difficult^[22]. Unlike special-purpose systems, our system is based on a general purpose big-data processing system, which can be extended to support both correlated and anti-correlated data.

Security is another important issue in electric power storage systems. User privacy is a well-known security problem when big data processing is applied in power systems, and researchers have proposed many solutions to solve security problems. Ruj and Nayak^[23] proposed a new decentralized security framework to keep the whole aggregation process safe. Yan et al.^[24] proposed a decentralized security framework for data aggregation and access Control in smart grids. Their approach divides the power network into home area network, building area network, and neighboring area network, so data can be aggregated in each sub-network, thereby protecting customers' privacy. Li et al.^[25] proposed a distributed incremental data aggregation approach by using homomorphic encryption. Kalogridis et al.^[26] suggested that home electrical power routing can be used to moderate a home's load signature to hide appliance usage information. Rastogi and Nath^[27] proposed a private aggregation algorithm for distributed time-series data; this algorithm offers good practical utility without any trusted server. Attacks are another security problems for electric power data storage systems. To address this issue, Tan et al.^[28] studied the impact of integrity attacks on real-time pricing and employed control theory-based approaches to analyze the attack effect on pricing stability. Tan et al.^[29] studied the problem of attackers controlling real-time electrical markets by manipulating meter measurements. We can apply these security techniques to our system to enhance the safety of storing and processing massive electric power data.

7 Conclusion

In this paper, we introduce our system, which uses HBase to store electric power data. The system consists of clients, HBase database, status monitors, data migration modules, and data fragmentation modules. We introduce the designs of the modules and evaluate their performance through experiments. We also present a series of parameter tuning strategies to

improve the efficiency of our system.

Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2017YFB1003000); the National Natural Science Foundation of China (Nos. 61702096, 61572129, 61602112, 61502097, 61320106007, 61632008, and 61702097); the International S&T Cooperation Program of China (No. 2015DFA10490); the Natural Science Foundation of Jiangsu Province (Nos. BK20170689 and BK20160695); the Jiangsu Provincial Key Laboratory of Network and Information Security (No. BM2003201); the Key Laboratory of Computer Network and Information Integration of Ministry of Education of China (No. 93K-9); and the SGCC Science and Technology Program “the Distributed Data Management of Physical Distribution and Logical Integration”; and was partially supported by the Collaborative Innovation Center of Novel Software Technology and Industrialization and Collaborative Innovation Center of Wireless Communications Technology.

References

- [1] S. Y. Pan, T. Morris, and U. Adhikari, Developing a hybrid intrusion detection system using data mining for power systems, *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 3104–3113, 2015.
- [2] H. Jiang, K. Wang, Y. H. Wang, M. Gao, and Y. Zhang, Energy big data: A survey, *IEEE Access*, vol. 4, pp. 3844–3861, 2016.
- [3] M. Yigit, V. C. Gungor, and S. Baktir, Cloud computing for smart grid applications, *Comput. Netw.*, vol. 70, pp. 312–329, 2014.
- [4] S. Ghemawat, H. Gobioff, and S. T. Leung, The Google file system, *ACM SIGOPS Ope. Syst. Rev.*, vol. 37, no. 5, pp. 29–43, 2003.
- [5] Welcome to Apache™ Hadoop®! <http://hadoop.apache.org/>, 2017.
- [6] 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, in *Proc. 7th USENIX Symp. Operating Systems Design and Implementation*, Seattle, WA, USA, 2006, pp. 205–218.
- [7] Apache HBase–Apache HBase™ Home, <http://hbase.apache.org/>, 2017.
- [8] J. Dean and S. Ghemawat, Mapreduce: Simplified data processing on large clusters, *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
- [9] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, Spark: Cluster computing with working sets, in *Proc. 2nd USENIX Conf. Hot Topics in Cloud Computing*, Boston, MA, USA, 2010, p. 10.
- [10] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing, in *Proc. 9th USENIX Conf. Networked Systems Design and Implementation, NSDI’12*, San Jose, CA, USA, 2012, p. 2.
- [11] S. Kawasoe, Y. Igarashi, K. Shibayama, Y. Nagashima, and S. Nagashima, Examples of distributed information platforms constructed by power utilities in Japan, in *Proc. CIGRE Symp. 2012*, Paris, France, 2012, pp. 108–113.
- [12] T. Harter, D. Borthakur, S. Y. Dong, A. Aiyeer, L. Y. Tang, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, Analysis of HDFS under HBase: A facebook messages case study, in *Proc. 12th USENIX Conf. File and Storage Technologies*, Santa Clara, CA, USA, 2014, pp. 199–212.
- [13] Ganglia monitoring system, <http://ganglia.info/>, 2017.
- [14] D. Lasalle and G. Karypis, Multi-threaded graph partitioning, in *Proc. 27th IEEE Int. Symp. Parallel & Distributed Processing*, Boston, MA, USA, 2013, pp. 225–236.
- [15] H. Lyu, P. Li, Y. N. Xiao, H. J. Qian, B. Sheng, and R. M. Shen, Mass data storage platform for smart grid, in *Proc. 2016 Int. Conf. Progress in Informatics and Computing (PIC)*, Shanghai, China, 2016, pp. 530–535.
- [16] S. Rusitschka, K. Eger, and C. Gerdes, Smart grid data cloud: A model for utilizing cloud computing in the smart grid domain, in *Proc. 1st IEEE Int. Conf. Smart Grid Communications*, Gaithersburg, MD, USA, 2010, pp. 483–488.
- [17] W. Medjroubi, U. P. Müller, M. Scharf, C. Matke, and D. Kleinhans, Open data in power grid modelling: New approaches towards transparent grid models, *Energy Rep.*, vol. 3, pp. 14–21, 2017.
- [18] R. Meier, E. Cotilla-Sanchez, B. McCamish, D. Chiu, M. Histan, J. Landford, and R. B. Bass, Power system data management and analysis using synchrophasor data, in *Proc. 2014 IEEE Conf. Technologies for Sustainability (SusTech)*, Portland, OR, USA, 2014, pp. 225–231.
- [19] A. Bose, Smart transmission grid applications and their supporting infrastructure, *IEEE Trans. Smart Grid*, vol. 1, no. 1, pp. 11–19, 2010.
- [20] T. Niimura, M. Dhaliwal, and K. Ozawa, Fuzzy regression models to represent electricity market data in deregulated power industry, in *Proc. Joint 9th IFSA World Congress and 20th NAFIPS Int. Conf.*, Vancouver, Canada, 2001, pp. 2556–2561.
- [21] Z. J. Fu, X. M. Sun, Q. Liu, L. Zhou, and J. G. Shu, Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing, *IEICE Trans. Commun.*, vol. 98, no. 1, pp. 190–200, 2015.
- [22] X. He, Q. Ai, R. C. Qiu, W. T. Huang, L. J. Piao, and H. C. Liu, A big data architecture design for smart grids based on random matrix theory, *IEEE Trans. Smart Grid*, vol. 8, no. 2, pp. 674–686, 2017.

- [23] S. Ruj and A. Nayak, A decentralized security framework for data aggregation and access control in smart grids, *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 196–205, 2013.
- [24] Y. Yan, Y. Qian, and H. Sharif, A secure data aggregation and dispatch scheme for home area networks in smart grid, in *Proc. 2011 IEEE Global Telecommunications Conf.*, Kathmandu, Nepal, 2011, pp. 1–6.
- [25] F. J. Li, B. Luo, and P. Liu, Secure information aggregation for smart grids using homomorphic encryption, in *Proc. 1st IEEE Int. Conf. Smart Grid Communications*, Gaithersburg, MD, USA, 2010, pp. 327–332.
- [26] G. Kalogridis, C. Efthymiou, S. Z. Denic, T. A. Lewis, and R. Cepeda, Privacy for smart meters: Towards undetectable appliance load signatures, in *Proc. 1st IEEE Int. Conf. Smart Grid Communications*, Gaithersburg, MD, USA, 2010, pp. 232–237.
- [27] V. Rastogi and S. Nath, Differentially private aggregation of distributed time-series with transformation and encryption, in *Proc. 2010 ACM SIGMOD Int. Conf. Management of Data*, Indianapolis, IN, USA, 2010, pp. 735–746.
- [28] R. Tan, V. B. Krishna, D. K. Y. Yau, and Z. Kalbarczyk, Impact of integrity attacks on real-time pricing in smart grids, in *Proc. 2013 ACM SIGSAC Conf. Computer & Communications Security*, Berlin, Germany, 2013, pp. 439–450.
- [29] S. Tan, W. Z. Song, M. Stewart, J. J. Yang, and L. Tong, Online data integrity attacks against real-time electrical market in smart grid, *IEEE Trans. Smart Grid*, vol. 9, no. 1, pp. 313–322, 2018.



Jiahui Jin is an assistant professor in the School of Computer Science and Engineering, Southeast University, Nanjing, China. He received the PhD degree in computer science from Southeast University in 2015. He had been a visiting PhD student at University of Massachusetts, Amherst, USA, during

August 2012 to August 2014. His current research interests include large-scale data processing, distributed systems, and parallel task scheduling.



Fang Dong is currently an associate professor in School of Computer Science and Engineering, Southeast University, China. He received the BS and MS degrees in computer science from Nanjing University of Science & Technology, China in 2004 and 2006, respectively, and received the PhD degree in computer

science from Southeast University in 2011. His current research interests include cloud computing, big data processing, and workflow scheduling. He is a member of both IEEE and ACM, and general secretary of ACM SIGCOMM China.



Junzhou Luo is a full professor in the School of Computer Science and Engineering, Southeast University, Nanjing, China. He received the BS degree in applied mathematics from Southeast University in 1982, and then got the MS and PhD degree in computer network both from Southeast University

in 1992 and 2000, respectively. His research interests are next generation network, protocol engineering, network security and management, cloud computing, and wireless LAN. He is a member of both IEEE and ACM, and co-chair of IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design.



Aibo Song received the MS degree from Shandong University of Science and Technology, and the PhD degree from Southeast University, China, in 1996 and 2003 respectively. He is currently an associate professor in the School of Computer Science and Engineering, Southeast University. His current research

interests include cloud computing and big data processing.



Huan Gong received the BE degree in computer science and technology from Northeast University at Qinhuangdao, in 2015. He is currently a master student in the School of Computer Science and Engineering, Southeast University. His research interests include big data and online aggregation.



Yingying Xue received the BS degree in computer science and technology from Nanjing University of Science and Technology in 2015. She is currently a PhD student in the School of Computer Science and Engineering, Southeast University, China. Her research interests mainly focus on spatial retrieval and semantic search on

knowledge graph.



Mingyang Du received the BS degree in computer science and technology from Anhui University of Finance and Economics, in 2016. He is currently a master student in the school of Computer Science and Engineering, Southeast University. His research interests include big data and cloud computing.

