Massive Power Information Processing Scheme Based on MongoDB

Yao Xu*, Jiayang Wang*
The School of Computer Science, Central South University, Changsha 410083, China.

*Corresponding author: 1273234935@qq.com, *csuwjy@mail.csu.edu.cn

Abstract. In order to solve the problem that the power consumption data presents a trend of sea quantification, and the huge power consumption data brings tremendous pressure to the system, especially slows down the speed of data query and gives users poor experience in using it, this paper proposes to use one of the non-relational databases (NoSQL) to store power consumption data instead of the traditional relational databases. To solve the problem that the collected electricity data can not meet the demand of page display, Hadoop MapReduce is proposed to preprocess the electricity data. Distributed architecture is adopted for storage and processing. MongoDB cluster and Hadoop cluster are overlapped and deployed. Combined with MongoDB's powerful storage capacity and Hadoop MapReduce's analytical and computational capability, a set of high-availability and high-performance data storage and pre-processing scheme for power-consuming universities is constructed.

1. Introduction
With the deep development of the power market, the pilot scope of incremental distribution reform has been expanding [7]. All localities have promoted the overall liberalization of the power generation plan for operational power users. The main power sellers represented by different market participants will show a diversified trend. If we can make full use of the vast amount of electricity information and analyze the potential data value, such as electricity consumption behavior, abnormal electricity consumption, line loss, etc., it will have important guiding significance for the maintenance of power grid lines, cracking down on leakage and stealing users, and pushing forward the ladder price. This year, the NPC and CPPCC put forward the strategic plan of "three types of two nets and world class". The construction of ubiquitous electric power Internet of things (SG-eIoT) is put on the agenda. We plan to build the ubiquitous power Internet of things in 2021, and build the ubiquitous electric power Internet of things by 2024. SG-eIoT provides us with more power information and makes full use of big data technology to achieve the goal of high perception and comprehensive interaction. According to the national data, there are more than 500 million terminals (450 million meters, all kinds of protection, acquisition and control devices) in the State Grid system. The number of data collected is more than 60TB, covering 450 million users. According to the planning of the State Grid, the number of devices connected to SG-eIoT system is expected to exceed 1 billion by 2025. By 2030, the number of devices connected to SG-eIoT system will reach 2 billion. The SG-eIoT will be the largest Internet of Things ecosystem.
The power monitoring and management system is an important means of managing power consumption data. Most of these systems are built on relational databases MySQL and Oracle. Their working principle is to report the power consumption data to the acquisition process and write the data to the database through the electric energy metering device, and display the desired functional results on the web page according to the functional requirements. This process includes a large number of data storage and complex query database operations [1]. The performance of database is very important to the performance of the whole system [2, 3]. The performance of traditional relational database has reached the bottleneck, which exposes many insurmountable shortcomings. NoSQL database is a new revolutionary movement of database. It uses more loose data format, more flexible data organization, strong ability of reading and writing, and supports distributed expansion. MongoDB, one of the representatives of NoSQL, which stores data in the form of documents, focuses on the performance of writing and searching large amounts of data, guarantees the efficiency of reading and writing data, and supports horizontal expansion. This is also in line with the characteristics of dealing with large amounts of power consumption information, which needs fast processing and does not have strong transactional nature.

"Data is assets". Big data brings new development opportunities and challenges to the power industry. In this paper, according to the characteristics of massive power consumption information, a scheme of power data storage and analysis based on the combination of MongoDB and Hadoop MapReduce is presented. Through experiments, a combined cluster of MongoDB and Hadoop is built as a data processing platform, which proves that the scheme is available, efficient and has better performance than the traditional mode.

2. MongoDB database
MongoDB is an open source, efficient and high performance document-based non-relational database. A data unit consists of a set of key-value pairs, which are similar to a table record of a relational database. But the key-value pairs of MongoDB can be documents, arrays and document arrays. This feature also makes the data structure of MongoDB much more complex than the tables of a relational database and makes good use of this feature. In order to better play the performance of MongoDB [13].

Electric sensor equipment has the characteristics of large-scale data acquisition and high-frequency command issuance, which requires high performance of data reading and writing. The performance of relational database is limited by single computer. The cost of improving concurrency performance by adding hardware specifications is too high, and the performance reached is limited. Using multiple MongoDB servers to store data together can improve database concurrency, query efficiency, solve the problem of large-scale data storage and reduce the cost of storing massive data. Through the distributed cluster architecture, the "smooth expansion" is realized, and the business can be maintained uninterrupted in the process of expansion. It is an excellent alternative to relational databases. The advantages of MongoDB are summarized as follows:

1) High concurrent writing meets the requirement of large capacity data storage and real-time mass writing.
2) Quick query and reasonable fragmentation rules can double the query speed.
3) Low cost, uniform distribution of data to multiple low-cost servers, reduce the cost of data storage.

3. Hadoop Distributed Architecture
Hadoop is an open source software architecture for distributed systems developed by Apache Foundation. It can efficiently store and compute PB-level data. It uses a simple programming model to distribute large data sets across computer clusters. The aim is to expand from a single server to thousands of computers, each providing local computing and storage. The two core components of Hadoop are distributed file system HDFS and programming model MapReduce. HDFS is a non-relational database with strong fault tolerance and can build scalable distributed file systems on multiple low-cost servers. MapReduce is a model developed by Google Corporation in 2004 for parallel processing and generating big data. It is a linear and scalable programming model [10]. It encapsulates complex distributed storage
management, data distribution, data communication and synchronization, and results summarization, and simplifies this series of processes into "Map (mapping)" and "Reduce (simplification)" exercises. The main task of MapReduce is to map a group of data into a new group of data according to some rules, and then summarize and output them through Reduce simplification [4]. The system details are separated from functions. Users only need to describe the calculation content. MapReduce will automatically distribute the tasks written by users to Hadoop cluster nodes for calculation, which greatly simplifies the development process and improves the development efficiency.

4. Data Processing Scheme Design

4.1. Overall Scheme Design
MongoDB provides powerful fragmented storage function, expands data storage space and improves data query speed. It is very suitable for handling a large number of historical electricity data and log records. It also provides MapReduce functionality, written in javaScript, but with poor performance, it is not suitable for data containing some complex calculations or requiring real-time synchronization. MapReduce is good at analyzing and calculating massive data. It can distribute huge and complex data computation to many low-end and inexpensive servers. MongoDB can also build clusters to realize distributed storage. So why not combine these two technologies? Deploying clusters does not require high performance of server configuration. For most enterprises, it saves money. It is a feasible scheme to increase the cost and performance. Hadoop's MapReduce is used to replace MongoDB's MapReduce as the program and framework of data processing, and MongoDB fragmentation is used to replace Hadoop's HDFS as the input source of data processing. The overall scheme of the system is shown in Figure 1.

![Figure 1. Overall Scheme for Massive Power Consumption Data Processing](image)

4.2. MongoDB cluster deployment
There are three ways to build a MongoDB database cluster: master-slave mode, replica set and fragmentation. To ensure the efficiency and security of the cluster, the cluster is deployed in the form of fragmentation + replica set, which consists of Config Server configuration server, Mongos routing server, Shard fragmentation and replica set. Config Server only stores meta-information of database; data requests must pass through Mongos, which is responsible for forwarding these requests to the
corresponding Shard server; Shard is responsible for fragmenting storage of data, different server's fragmentation nodes save different data; replica set is to ensure the reliability of fragmentation, if a machine fails, replica set replaces the fault fragmentation to realize the function. In the production environment, the combination of fragmentation and replica set can make MongoDB perform better.

Now we are going to build a cluster. Firstly, we will determine the number of components, three mongos, three config servers, three shard data fragments, one replica set and one arbitration node for each fragment, three respectively. A total of 15 instances need to be deployed. Because of the limited resources, in order to make full use of the server resources, we do not place a single server for each component. We deploy the master node, replica set and arbitration node on the same server. The physical structure of the MongoDB database cluster is shown in Figure 2.

![Figure 2. Physical Structure of MongoDB Database Cluster](image)

4.3. Joint deployment of MongoDB and Hadoop

So far, MongoDB cluster has been built. How to deploy Hadoop cluster reasonably on the basis of MongoDB cluster has a great impact on the performance of the whole data processing framework. When deploying the joint cluster, it is required to minimize the data transmission overhead in the calculation
process. By placing MongoDB fragments on Hadoop's local file system, the computing nodes can read less to be calculated. The time cost and error rate of data can improve the efficiency of cluster computing. So directly on the three servers of the original MongoDB cluster, we continue to build the Hadoop distributed framework, and deploy Hadoop YARN and Data Node overlapping with the MongoDB fragmentation server. MapReduce can read data from the local MongoDB nodes, thus improving the performance of the cluster. The structure of the federated cluster is shown in Figure 3.

Figure 3. Hadoop Cluster deployment

5. Electricity Information Analysis

5.1. Data preparation and loading
Prepare all the historical electricity consumption data of a school and a campus. This data records the daily electricity consumption data of each meter in the campus since its use. The number of users in a school is not small. You can imagine that the cumulative records of electricity consumption for a meter device every day are tens of points. The data records include the primary key id, the meter id, the amount, the time and the meter start code. There are nearly 3G data and millions of records. These records are exported from Oracle database. The format is Json format similar to that of MongoDB storage documents. MongoDB supports direct import of Json format files.

1. Creating databases and collections
   db.createCollection("T_METER_HIS")
2. Import the Json file into the built collection
   Mongo import --db dbName --collection Name --file file Path/xx.json

5.2. MapReduce execution and analysis
The original data collected on the spot is also called irregular data. The format of the data cannot meet the basic requirements of the data effect that we want to present. If we need to calculate and screen the data every time, it will be too time-consuming and laborious, so we need to pretreat them and convert them into the more regular data we need, which is based on the historical data of students' electricity
consumption[4, 5, 6]. On the basis of this, if we want to filter all the outliers in a certain period of time, how to define the outliers, and whether the filtering algorithm can process zero or empty data, all of these are what MapReduce needs to do. Follow-up researchers can query directly in the database without further calculation and judgment. Detailed steps of data processing are as follows:

1. Establish the Maven project and import the jar package needed to run the program.
2. For the errors or missing field information in the acquisition, the interpolation method is based on the mean value of nearly two frozen data.

5.3. Comparing with the results of relational database
Storage performance comparison: Set the number of MongoDB replica sets in the MongoDB-Hadoop cluster to 1. Store three historical records of the same file size in MySQL, HDFS and MongoDB databases, and observe the storage space occupied by each database, as shown in Figure 4. As can be seen from the results in the figure, MongoDB saves more disk space than MySQL and HDFS.

Search performance comparison: execute query statements with the same effect in MySQL, HDFS and MongoDB databases, set up three query statements with increasing complexity, namely, SQL1, SQL 2 and SQL3, and record the execution time in three databases. As you can see, with the increasing complexity of SQL statements, MongoDB is far more efficient than ordinary SQL statements in relational databases, but slightly less efficient than HDFS local file systems. The comparison results of the three execution speeds are shown in Table 1.

Table 1. Comparison of MySQL, HDFS and MongoDB execution speeds

|        | SQL1  | SQL2  | SQL3  |
|--------|-------|-------|-------|
| MySQL  | 106.67| 153.24| 367.29|
| HDFS   | 78.27 | 91.32 | 132.17|
| MongoDB| 80.16 | 108.13| 152.40|
5.4. Performance Test of System Fault Handling

Main Node Downtime Test: Open MongoDB Cluster and Hadoop Cluster System, after connecting successfully and reading data normally, start MapReduce task, shut down one of the machines quickly without shutting down the service process. We observed the following phenomena: for a short time, data cannot be obtained from MongoDB, but can be retrieved immediately, and the weight of the machine just now will be increased. New start, check the status of the MongoDB node, and find that the node just started has become a slave node. This is because we have set up the MongoDB replica set, which provides the function of automatic failover, that is, when the primary node loses connection, the most active slave node (if there are more than one slave node) will be elected by the arbitration node to serve the system. When the fault node recovers, it no longer acts as the primary node but becomes a replica node.

Storage space expansion test: Because our experimental environment is virtual machine, each virtual machine only allocates 20G hard disk storage space, MongoDB database usable space is less than 10G, three MongoDB master nodes are 60GB storage space, can accommodate tens of millions of power records, because our real power consumption data is not so much, in order to test the storage expansion performance of MongoDB[11]. To create a new collection, write a large number of documents into the collection using programs, check how much data you have using db.stats(), stop the cluster and add two new virtual servers to the cluster, set up the replica set and configuration nodes of the new node according to the 4.1.2 deployment mode, restart the cluster, use db.Stats() command again to view the cluster information and find the total capacity of the cluster. The increased capacity is just the capacity of the new server database. After the test of storage space expansion, the MongoDB cluster has the ability of lateral expansion and high availability.

6. Conclusion

This paper presents a scheme of using MongoDB to store massive power consumption data and Hadoop MapReduce to preprocess power consumption data. By using the technology of Hadoop and MongoDB, which are currently hot in concurrent data processing, the problem that the performance of single computer is difficult to expand is solved by building clusters. Experiments show that the scheme uses non-relational database instead of conventional relational database, and the storage performance and read-write performance of the server are improved, which can better reflect its advantages when the amount of data is large enough. The purpose of this scheme is to store and reorganize massive and disorderly electricity data[12,13], to provide a more rapid and reliable operation for displaying information of power supervision system, not only to meet the needs of users, but also to reduce the cost of the system. Of course, MongoDB and Hadoop can do more than that. In the field of statistics, the larger the sample size, the more the statistical data can reflect the real individual situation. The era of big data has arrived. We hope that we can use the power of big data to dig out the potential value of massive data in the power system and find more useful information so as to enable future power regulation. The system is more humanized and intelligent.

Acknowledgments

This work was financially supported by the National Natural Science Foundation of China (No.61772031) and the Hunan Natural Science Foundation of China in 2020.

References

[1] Divya Mahajan, Cody Blakeney, Ziliang Zong, Improving the energy efficiency of relational and NoSQL databases via query optimizations, J. Sustainable Computing: Informatics and Systems.

[2] Roman Čerešňák,Michal Kvet, Comparison of query performance in relational a non-relation databases, J. Transportation Research Procedia.

[3] J. Derek Colley, Clare Stanier, Identifying New Directions in Database Performance Tuning, J. Procedia Computer Science.
[4] Kuo Alex, Chrimes Dillon, Qin Pinle, Zamani Hamid, A Hadoop/MapReduce Based Platform for Supporting Health Big Data Analytics, J. Studies in health technology and informatics.

[5] Jin-woo Lee, Gwangseon Jang, Hohyun Jung, Jae-Gil Lee, Uichin Lee, Maximizing MapReduce job speed and reliability in the mobile cloud by optimizing task allocation, J. Pervasive and Mobile Computing.

[6] Mazin Alkathiri, Abdul Jhummarwala, M. B. Potdar, Multi-dimensional geospatial data mining in a distributed environment using MapReduce, J. Journal of Big Data, 2019, Vol.6 (1), pp.1-34.

[7] Yang Zhou, Shuaishuai Zhang, Libo Wu, Yingjie Tian, Predicting sectoral electricity consumption based on complex network analysis, J. Applied Energy.

[8] Alexander Thür, Toni Calabrese, Wolfgang Streicher, Smart grid and PV driven ground heat pump as thermal battery in small buildings for optimized electricity consumption, J. Solar Energy.

[9] Cho-Liang Tsai, Wei Tong Chen, Chin-Shiang Chang, Polynomial-Fourier series model for analyzing and predicting electricity consumption in buildings, J. Energy & Buildings.

[10] Alexander Thür, Toni Calabrese, Wolfgang Streicher, Smart grid and PV driven ground heat pump as thermal battery in small buildings for optimized electricity consumption, J. Solar Energy.

[11] J. McClay Wilbert, A Magnetoencephalographic/Encephalographic (MEG/EEG) Brain-Computer Interface Driver for Interactive iOS Mobile Videogame Applications Utilizing the Hadoop Ecosystem, MongoDB, and Cassandra NoSQL Databases, Diseases (Basel, Switzerland). 2018 (4).

[12] Mohamed Ben Brahim, Wassim Drira, Fethi Filali, Noureddine Hamdi, Spatial data extension for Cassandra NoSQL database, J. Journal of Big Data. 2016 (1).

[13] Hanen Abbes, Faiez Gargouri, Big Data Integration: A MongoDB Database and Modular Ontologies based Approach, J. Procedia Computer Science.