A Technique for Smooth Switching from Conventional Database to Big Data System

Peng ZeWu
Guangdong Gower Grid Company of China Southern Power Grid, DongFeng East Road No. 757
Yuexiu District of Guangzhou Gity, Guangdong Province
China

Huang JianWen*
Guangdong Gower Grid Company of China Southern Power Grid, DongFeng East Road No. 757
Yuexiu District of Guangzhou Gity, Guangdong Province
China
*952898435@qq.com

***Abstract*—In order to comprehensively analyze enterprise data using big data techniques, data need to be transformed from conventional database to unstructured database, where smooth switching of database has a great significance for enterprises. In this study, a technique for smooth switching from conventional database to big data system is proposed based on a practical case of conventional enterprise. We will also briefly introduce the process of database switching technique and describe solutions to the problems associated with traditional database switch techniques in detail. Finally, the proposed technique is verified using a practical case of enterprise application.**

Keywords—Conventional database, Big data system, Switch, Scheme

I. Introduction

With the development of cloud computing and internet technologies, vast amount of data are generated every day. Apart from structured data, there also exist numerous unstructured data, such as video, audio, and text[1]. For massive data, traditional database has the features of high storage efficiency, but low query efficiency and lack of extendibility[2]. Unstructured database can efficiently solve the problems existing in the traditional structured database. For massive data, unstructured database features excellent read/write performance, easy extendibility, and instantaneous storage of data in custom format[3]. In order to more comprehensively analyze enterprise data using the big data technologies, data storage requires transformation from conventional database to unstructured database[4].

Up to now, few associated studies have paid attention to the switching techniques from traditional relational database to new unstructured database[5]. The following three challenges are the major challenges: 1) Inability of retreat and data loss caused by single-system stop switch. In case of single-system stop, data loss during the process of write interface switch would generate irrecoverable losses for enterprises. Existence of problems in the write interface of new data can make it impossible to go back to the original database state[6]. 2) Unable to process dirty data produced in switching tests. Some problems may occur when data are stored in a new database through the new write interface, generating a great number of dirty data with unsatisfying quality that are hard to repair. 3) Big data system would cause excessive learning costs for traditional developers and a great difficulty to switch the data access layer. For data access of the new database, techniques of big data system are necessary for developers, requiring a long-term and high-cost learning. In addition, transplantation of upper-level application would exhibit a high difficulty.

During the process of database switch, data loss would cause huge economic losses for enterprises. Therefore, smooth switching technique has a great significance for enterprises. Based on a practical case of traditional enterprise, this research proposed a technique for smooth switching from conventional database to big data system. Section 2 briefly introduced the detailed process of database switch. Section 3 described solutions to the problems associated with the traditional database switch technique in detail. Section 4 demonstrated a practical case of database switch technique applied in enterprise and Section 5 exhibited the conclusions of this paper.

II. Switch Process

Database switching technique contains four stages with fifteen steps in total.

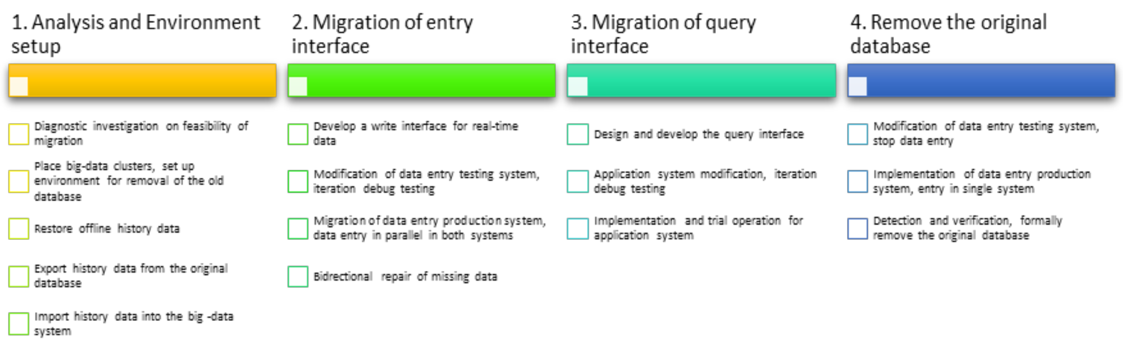
1. Analysis and Environment setup
2. Migration of entry interface
3. Migration of query interface
4. Remove the original database
Diagnostic investigation on feasibility of migration
Place big-data clusters, set up environment for removal of the old database
Restore offline history data
Export history data from the original database
Import history data into the big -data system
Develop a write interface for real-time data
Modification of data entry testing system, iteration debug testing
Migration of data entry production system, data entry in parallel in both systems
Bidrectional repair of missing data
Design and develop the query interface
Application system modification, iteration debug testing
Implementation and trial operation for application system
Modification of data entry testing system, stop data entry
Implementation of data entry production system, entry in single system
Detection and verification, formally remove the original database

Fig. 1. Working stage division and specific steps of database switch

A. Analysis and Context Preparation

In the first stage, we need to analyze the switch-out requirement, build a complete hardware and software environment for data switch-out, and then import the historical data to the big data system.

1) Big data cluster arrangement and switch-out environment establishment

Design the architecture of big data cluster environment according to the switch-out requirement, such as underlying storage architecture, offline & online analysis architecture, and query architecture. Determine the open-source tools to conduct data management. In this way the hardware system planning of big data will be established with consideration factors including network throughput, data capacity, network bandwidth between nodes, and environmental management requirements. Apart from the testing environment, an equivalent testing environment for manufacturing system needs to be built as the basis of subsequent joint-test verification.

2) Offline historical data restoration

If it exists the situation of historical data detaching from the production database due to the limitation of conventional database, a backup environment of historical data (hardware & software) should be built on the side of production system to restore the historical data using assistive tools of the legacy system.

3) Exporting historical data from original database

Export historical data from the offline historical database into an intermediate storage in batches according to the prescribed format. At the same time, export data of the manufacturing system into this area.

4) Importing historical data into big data system

In order to conduct batch import of historical data, we need to pay attention to the following issues: 1) Recording the progress of import program to estimate the processing time; 2) Developing the import program and designing the corresponding batch-cleaning procedure for the sake of backspacing; 3) Planning for a correctness verification after the import. Import should be conducted using a scale-increase incremental import method. Implement correctness verification every time before the scale-increase incremental import. Pay attention to all the error information shown in the import process for the convenience of backspacing in time.

B. Write Interface Switch

Subsequent to the historical data import, we need to modify the manufacturing system for the write interface switch and real-time data access.

1) Write *interface* development of real-time data

Develop a write interface of real-time data according to the architecture of production system, namely, an adapter interface designed for the upper-level system based on practical applications. Allocation methods need to be provided during the developing process to meet the requirements of relocation and update of big data cluster. Abnormal log is also required in order to conduct fault diagnosis in case of failure. For databases with distributed architecture, we need to consider the issues of temporary downtime control and interface switch strategy on the aspect of interface. To prevent data loss under the action of impact load or long-term heavy load, a certain buffer mechanism of temporary data has to be developed, which could thereby improve the writing reliability.

2) Improvement of data entry testing system, iterative joint-test verification

Joint-test verification must be carried out subsequent to the write interface development. Regular machine for joint-test verification can control the load for testing. Similar to the import and export process of data, the testing process also needs to be conducted following the idea of systematization. That is, to implement the procedure in the order of correctness test to performance test then to pressure test and fatigue test. System improvement is completed once all of the work above is fulfilled. On this basis, a detailed test report can be written for the exhibition of system performance in the future conclusions.

3) Data entry production system switch and double-system parallel writing

The product can be released once the joint-test verification is conducted. During this period, writer interface of the legacy database system cannot be deleted due to the unfinished modification of query interface for the production system at this point of time. Moreover, in case of errors in the write interface, data can be repaired through the backup system and the original system can be switched back even under the worst conditions. Therefore, this process of double-system parallel writing needs to be remained till the complete switch-out.

4) Bidirectional repairing of missing data

Various problems may occur during the entire process of write interface switch in the production system, leading to diverse data quality issues throughout the whole process of switch. Therefore, bidirectional repairing is necessary, which simultaneously export data from the backup database, delete the existing data of this stage in the new database, and import the backup data into the new database.

C. Query Interface Switch

The historical data will be totally transferred into the big data system subsequent to the writer interface switch. Online data will be imported into the system in real time. The condition of business system switch is not satisfied until data are in the system. The next step is to conduct query interface switch.

1) Design and development of query interface

Determine the query interface form and design a matched query interface according to the scheme. The adoption of early development framework in upper-level applications, such as JDBC, Hibernate, and JPA, makes a difficult compatibility with the big data system during access. Hence, the big data system can be reformed from the following aspects: providing visual table interface in the database, shielding the learning curves of traditional application developers, reducing the modification of the system, and calling the query interface in a conventional way using a certain type of matched interface.

2) Application system improvement and iterative joint-test verification

Similar to the alteration of writer interface, query interface must experience the joint-test verification as well as the incremental test in a scale-increase manner, i.e., following the order of correctness test to performance test and pressure test then to fatigue test. In this way, the overall performance of query interface can be obtained. For example, we can acquire the QPS (queries per second) quantile out of 90. Meanwhile, abnormal log of the query interface is also in a great significance.

3) Online commissioning of application manufacturing system

After all the query interfaces are modified, appropriate practical case can be adopted to conduct online switch and business verification of the production system.

D. Original Database Switch-out

The original historical database system can be switched out only when the entire process of historical data import, writer interface modification, query interface modification, and database switch-out is completed and all the links are checked without any problems. Generally, the switch-out operation should not be carried out immediately. It usually requires three to six months to conduct test run. The further operation can be conducted only subsequent to double-check.

1) Improvement of data entry testing system, data entry cut-off

Modify the manufacturing system of data entry, delete the real-time written code, cease the data entry, and implementing test to insure the total stop. This process demands a complete modification strictly in this process. This can be explained by the fact that unidentified legacy code in the system would continue to write data into the original database, causing a system halt or error if the problem is neglected.

2) Data entry production system release and single-system data writing

Once there is no problem, the system can be released and the data entry can be ceased.

3) Close all the interfaces associated with the old database for a period of production system double-check (3 – 6 months). Switch-out ends.

III. CORE ISSUES AND SOLUTIONS

A. Double-system Parallel Switch

Traditional database switching techniques could lead to no backspacing and data loss due to the single-system halt switch, causing irrecoverable losses for enterprises. Problems may occur during the operation process in the newly-released database because of the impossibility of returning to the original database after the upper-level application modification.

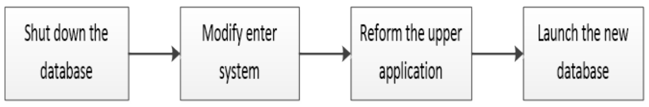

Fig. 2. Conventional database switch technique

Solution: Data can be written into the new database through the adapter interface developed according to the architecture of the production system. After the iterative joint-test verification on the interface, data can be written into the new and old databases in parallel. Problems generated during the writing process can be repaired till the data can be stored in the new database in real time to fulfil the write interface switch. Upper-level application can be modified till the new application is released. Then the old database can be switched out. In the database switching test, the old database is always in the state of running and storing data. The entire system can return to the old database storage state at least without data loss as shown in Fig. 3.

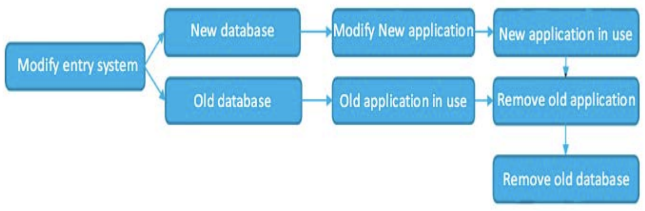

Fig. 3. Technique process of double-system parallel database switch

B. *Repairing Technology of Lossless Data*

Conventional database switching technique is unable to process data loss generated in the test of database switch as shown in Fig.4. There are mainly three types of data loss: 1) Error loss caused by the ambiguity of database switch time. Subsequent to a successful database switch, data need to be imported into the new database from the original database of switch. Ambiguity of the database switch time may lead to a partial data loss during the importing process. 2) Stop loss happened in the database switch process. Switch-out of the original database may have a time delay during which transmitted data may be lost. 3) Dirty data loss generated in the database switching test. The new write interface may experience some problems that produce a huge number of dirty data in the new database. In this situation, data repair would be hard to carry out.

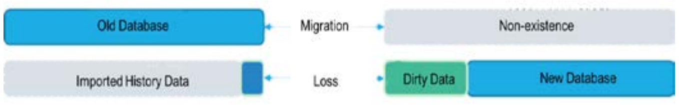

Fig. 4. Data loss generated during the switch test

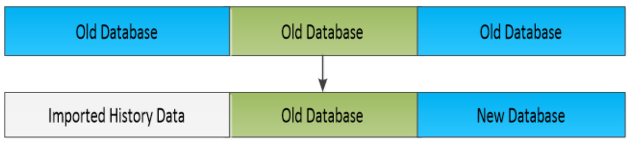

Fig. 5. Lossless data repairing

Solution: Choose two time points, starting point and ending point that are before and after the release of the new data entry interface, respectively. Delete all the data in the new database from the starting point to the ending point. Import data corresponding to this period from the original database to realize the lossless data repair as shown in Fig. 5.

C. *Application Transplantation of Friendly Data Access*

Conventional database switching technique could cause structural variation of the data access layer. Learning the big data system would cost a great deal of money for traditional developers and transformation has a high difficulty. Due to upper-level applications usually adopt the early program-development frameworks, such as JDBC and Hibernate; variation of data access layer requires developer to learn a huge amount of techniques associated with the big data, which increases the learning costs.

Solution: By encapsulating the big data system using the table function technique, we can realize the implantation of from sentence during the storing process, combine the advantages of structured data and unstructured data, and lower the learning threshold for developers. By transparently connecting the big data system with relational database using special agent service, advantage integration of relational database and big data system can be realized at the lowest cost, helping legacy applications smoothly transit to the big data platform.

IV. Application Case

A certain traditional enterprise receives real-time data from 6000 machines every day, including working condition data generated during the operating process of main engines, working condition data generated by the electronic devices installed on the main engines, and real-time location data sent by the main engines in each location. The total number of daily data is 50 million. The switching process of traditional enterprise data from Oracle database to LAUD big data system is realized using this technique. The structure of data collection and processing platforms of this corporate is shown in Fig.6, including a M2M platform for data collection and processing as well as the upper-layer application system based on the M2M platform.

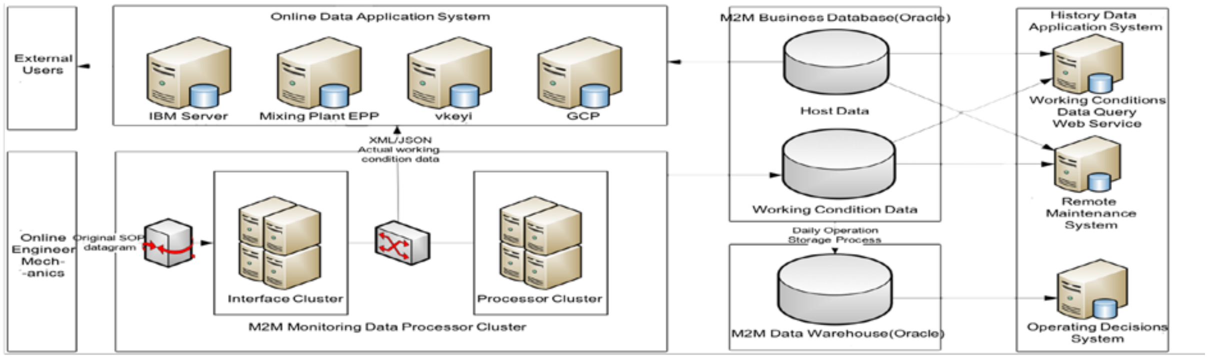
External Users
Online Data Application System
IBM Server
Mixing Plant EPP
vkeyi
GCP
XML/JSON
Actual working condition data
Online Engineer Mech- -anics
Original SOP datagram
Interface Cluster
Processor Cluster
M2M Monitoring Data Processor Cluster
M2M Business Database(Oracle)
Host Data
Working Condition Data
Daily Operation Storage Process
M2M Data Warehouse(Oracle)
History Data Application System
Working Conditions Data Query Web Service
Remote Maintenance System
Operating Decisions System

Fig. 6. Structural diagram of the data collection and processing platforms in the corporate example

As shown in Fig. 6, the M2M platform is the distributed processing cluster responsible for receiving and parsing the working condition data from the operating devices in the corporate. The M2M platform is composed of interface PCs clusters, processor clusters, business databases and data warehouses.

V. Conclusions

Traditional database cannot satisfy the increasing requirement of data storage and the traditional database switching technique could bring huge losses to enterprises due to the data loss. This paper designs and implements a switching technique for smooth switching from conventional database to big data system. Advantages of this technique in the execution process are listed below. 1) There is no data loss because data are written into the new database in parallel through the double-system where the original database can fulfil data backup. 2) Delete dirty data in the write interface test and importing the data in the corresponding period from the original database. 3) Learning costs are reduced for developers by encapsulating the SQL-like language for data access of big data system and decreasing the modification of upper-level applications. The future work would focus on studying and designing more high-efficiency database switch techniques.

REFERENCES

[1] Abouzied, A., Bajda-Pawlikowski, K., Huang, J., Abadi, D.J., Silberschatz, A.:Hadoop DB in action: building real world applications. In: Proceedings of the ACMSIGMOD International Conference on Management of Data, pp. 1111–1114 (2010)

[2] Baru, C., et al.: Discussion of bigbench: a proposed industry standard performance benchmark for big data. In: Nambiar, R., Poess, M. (eds.) TPCTC 2014. LNCS, pp. 44–63. Springer, Hiedelbeg (2015)

[3] Baru, C., Bhandarkar, M., Nambiar, R., Poess, M., Rabl, T.: Big data benchmarking and the BigData top100 list. Big Data J. 1(1), 60–64 (2013)

[4] Chowdhury, B., Rabl, T., Saadatpanah, P., Du, J., Jacobsen, H.A.: A BigBench implementation in the Hadoop ecosystem. In: Rabl, T., Raghunath, N., Poess, M., Bhandarkar, M., Jacobsen, H.A., Baru, C. (eds.) WBDB 2013. LNCS, vol. 8585, pp. 3–18. Springer, Heidelberg (2014)

[5] Transaction Processing Performance Council, TPCx-HS, February 2015.www.tpc.org/tpcx-hs/

[6] Floratou, A., Minhas, U.F., ¨ Ozcan, F.: SQL-on-hadoop: full circle back to sharednothing database architectures. Proc. VLDB Endowment (PVLDB) 7(12), 295–1306 (2014).