IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 13, 2019, accepted March 22, 2019, date of publication April 3, 2019, date of current version April 15, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2909060

Big Data-Based Improved Data Acquisition and Storage System for Designing Industrial Data Platform

DAOQU GENG[1,2], CHENGYUN ZHANG[1,2], CHENGJING XIA[3], XUE XIA[1,2], QILIN LIU[1,2], AND XINSHUAI FU[1,2]

[1]Key Lab of Industrial Internet of Things and Networked Control, Chongqing University of Posts and Telecommunications, Ministry of Education, Chongqing 400065, China
[2]College of automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[3]College of Electronic Information, Yangtze University, Jingzhou 434023, China

Corresponding author: Daoqu Geng (gengdq@cqupt.edu.cn)

This work was supported in part by the Project of Basic Research and Frontier Exploration of Chongqing Science and Technology Commission under Grant cstc2016jcyja0586, and in part by the Project of Technological Innovation and Application Development of Chongqing Science and Technology Commission under Grant cstc2018jszx-cyzd0078.

ABSTRACT Big data-based acquisition and storage system (ASS) plays an important role in the design of industrial data platform. Many big data frameworks have been integrated compression and serialization method. These methods cannot meet the needs of industrial production information management for requiring time-consuming and mass storage. Based on the existing big data frameworks, we propose an enhanced industrial big data platform in order to reduce the data processing time while requiring fewer data storage space. Specifically, this paper focuses on evaluating the impact of multiple compression and serialization methods on big data platform performance and tries to choose optimal compression and serialization methods for the industrial data platform. Compared to the methods integrated in Hadoop and Spark, the experimental results showed the data compression time of the platform has been reduced by 73.9% with a less than 96% the size of data compressed, furthermore, the data serialization time has been reduced by 80.8%. With the increasing amount of data, it takes less time to compare with benchmark methods.

INDEX TERMS Big data, acquisition and storage system (ASS), industrial data platform, data acquisition, data storage, data compression, serialization.

I. INTRODUCTION

Big data analysis of industry is considered as a necessary aspect for further improvement in order to improve the profit margin of industrial production and operation, and represents the next frontier of innovation, competition and productivity [1]. Nowadays, industrial data platform is the core component of industrial data storage, computation and analysis for the management of intelligent plant. With the increasing number of intelligent equipments used in intelligent plant, however, intelligent plant can acquire a large quantity of data of Radio Frequency Identification (RFID) and intelligent equipments, thus providing rich data sets for manufacturing industry [2]–[7]. The current trend in industrial systems is to use different big data engines as a means of processing a large quantity of data that cannot be processed by ordinary infrastructures. Industrial infrastruct- ure faces a large number of problems, including challenges such as defining different efficient architecture settings for different applications and defining specific models for industrial analysis.

In recent years, Spark and Flink have been well studied and applied on the Internet. The MapReduce is a computing framework used by most industrial enterprises at present. Iterative computing is involved in many aspects of industrial data analysis, which is used to seek optimal control and management solutions. However, the MapReduce framework is ineffective in iterative computing and the performance of the Spark framework has some obvious advantages over the MapReduce [8], [9]. There have been many hot issues and difficult problems in the research of industrial big data platforms: reducing the data processing time and the size of

The associate editor coordinating the review of this manuscript and approving it for publication was Mohamed Elhoseny.

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

space for big data acquisition and storage, optimizing the performance of computing framework, providing a wealth of function, improving system stability.

Our motivation for this study is to look for an efficient plan for acquiring and processing industrial data. Moreover, we need to find compression and serialization method which have good performance in the time of data processing time and the space of data storing. Furthermore, we aim at designing an industrial data platform with higher real-time performance and higher compression ratio. Based on the above considerations, we design and implement an industrial data platform, which integrated both LZ4 and Protobuf method for data processing. The LZ4 is used for data compression, and the Protobuf is used for data serialization. Based Flume and Sqoop, we build up a data acquisition module in the industrial data platform. Meanwhile, the Hadoop is adopted to store data acquired by intelligent equipments in intelligent plant. In order to enhance the data analysis capabilities of the industrial data platform, both the Spark and the Flink are integrated into the computing system. In addition, the industrial data platform adopt Django as front-end framework for the sake of system stability and system maintenance management.

The structure of this paper is as follows. In Section I, we introduced the motivation of this research. In Section II, we introduce the relevant background. We build the data platform for industrial data analysis based on big data in Section III. Then, we show some results of the acquisition and processing system in Section IV. In Section V, we discuss the conclusions of this research.

II. BACKGROUND

As the core of the new generation of information technology and industrial development, industrial big data is deeply affecting all aspects of the whole industry chain, such as R&D, manufacturing, operation management and sales service of China's manufacturing industry [10], [11]. In the future, it will promote the transformation and promotion of traditional manufacturing. It can promote the manufacturing strategy of ''Made in China 2025''.

A. DATA ACQUISITON IN INDUSTRY

There are massive multi-source heterogeneous data inside and outside the plant [12]. The data platform needs to connect multiple types of data in equipments, production lines, products and industrial software. At present, industrial data lack effective interoperability standards or low-cost acquisition schemes. This leads to high costs of data collection and integration of equipments and systems. Thus, it is difficult to support functions of industrial data analysis and application development.

B. DATA ANALYSIS AND INDUSTRIAL BIG DATA PLATFORM

There are a large number of data analysis requirements in the plant. The data platform needs to support multiple types of data analysis and data visualization services, such as Finite Element Analysis [13], Optimization, Deep Learning [14]–[19], Knowledge Mapping [20], Digital Twins [21].

Generally, big data platform include three modules, which are data acquisition module, data storage module and data computation module [22]. The data acquisition module provides a data source for data analysis of the big data platform, and the data storage module provides data source and storage space of the data computation module. It provides data calculation, machine learning, graph analysis, data query, which is the core component of data analysis. The computation module computes massive amounts of data, mines useful data value information, and provides decision-making grounds for industrial decision makers.

Both Sqoop and Flume are the main framework for data acquisition modules [23], [24]. The Sqoop is used for data interaction between Hadoop and relational database, and imports relational database data into Hadoop distributed file storage system (HDFS) [25], [26]. As a data acquisition framework, the Flume supports the development of various data senders in the log file system to acquire their data, which simply processes the data and then stores the processed data in HDFS, Hive, or turns it into a producer of Kafka.

The data storage module mainly uses the HDFS [27]. Hive [28], Shark [29] and Spark SQL [8] support data operations by Structured Query Language (SQL), but the efficiency of the three operations is increasing in steps. Spark SQL is about two orders of magnitude more efficient than Hive.

There are many types of computing frameworks in the big data ecosystem, including MapReduce, Spark, Storm, Flink and so on [30]. The MapReduce is almost eliminated because of requiring constant serialization, which seriously wastes computing system resources. Moreover, many platforms for industrial big data are developed based on MapReduce computing framework. Obviously, the performance of these data platform is inefficient. The Spark, however, is the most active framework for the big data ecosystem. In addition, the Spark is based on memory, and it can automatically adjust the memory usage. When the memory is insufficient, it will automatically transfer the location of data computation from memory to disk. The speed at disk computing is also nearly 10 times faster than MapReduce [31]. Furthermore, Marhout's R&D team is no longer maintaining and perfecting for MapReduce, but the machine-learning module needs to be updated constantly in the field of industrial data analysis. Therefore, in this study, the Spark computing framework is used instead of the Hadoop computing one.

III. PLATFORM ARCHITECTURE

According to industrial data acquisition and processing requirements, this paper designs an industrial big data platform. The overall framework of the industrial data platform is shown in Figure 1. The data platform includes six layers in terms of data flow. These six layers are device layer, acquisition layer, storage layer, computing layer, service layer and display layer, which correspond in turn to data acquisition, data storage, data analysis, service package and front end of

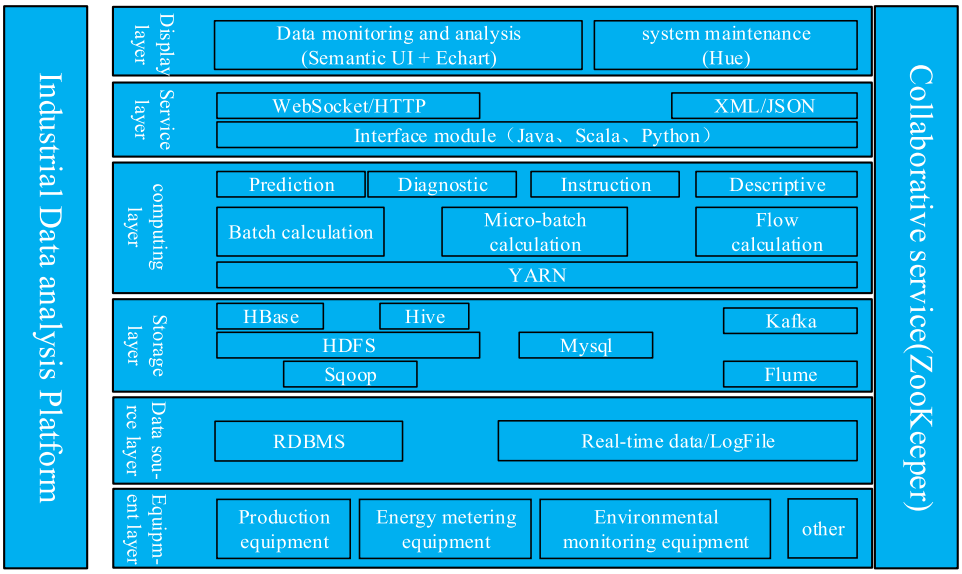


FIGURE 1. Overall architecture of industrial data analysis platform.

industrial data. This study focuses on the acquisition layer, the storage layer and the computing layer.

There are many forms of industrial data distribution, so the data acquisition module of this platform adopts Sqoop and Flume. The Sqoop acquires a lot of data stored in relational databases in the industry, and the real-time requirement of these data is not high. And the Flume acquires real-time data acquired by the intelligent equipments. In the industry, dynamic data, like that of safety monitoring and that of equipment operation status, is acquired by Flume. And then these data is transmitted to the message queue of Kafka. The Kafka becomes the provider of real-time consumption of dynamic data for Spark Streaming and Structured Streaming. Some managers of the plant need to query historical data, so the data acquired from Flume needs to be stored in HBase or HDFS. Yet Another Resource Negotiator (YARN) is adopted by the resource scheduling framework, it supports all computing frameworks and facilitates integration of Spark and Flink. The computing framework uses Spark SQL for batch processing, Spark Streaming for micro-batch processing, Structured Streaming and Flink for stream processing. The results are encapsulated by web services in the service layer, and then the encapsulated data is transmitted to the display layer which integrates Echart, H5 and Hue by Django. Then the user can get the data analysis results by the display interface. In addition, ZooKeeper completes the collaborative management and high availability patterns of some distributed big data frameworks of the whole industrial data platform.

A. DATA ACQUISITION

The system transmits data by three channels. Firstly, data flow is a timeless computing module consisting of Flume, Kafka, Structure Streaming and Flink. Secondly, data flow is a data query module consisting of Flume, HBase, Spark SQL and Table. Thirdly, data flow is a visual analysis module consisting of Flume, HBase, Spark Streaming and Flink.

The data serialized method of the acquisition framework is upgraded. The specific acquisition scheme is as follows. The module for Flume data acquisition is shown in Figure 2. The information configured of Flume requires three parts: Source, Channel, and Sink. The Source receives the event from the server or log file and puts it into the Channel in bulk. Data transmission of Flume is divided into two data transmission channels, so you need to set KafkaChannel, HBase- Channel, KafkaSink and HBaseSink. The information configured of Sources requires to configure type, Channel, Internet Protocol Address (IP), Port and selector type. The specific configuration information is shown in Table 1.

The data acquisition method of Sqoop relational database is shown in Figure 3. After receiving the shell command or Java API command from the client, the Sqoop converts the command to the corresponding MapReduce task by task translator in Sqoop, and then transfers the data from relational database to Hadoop to complete the copy of the data. The Sqoop operates on relational databases by four ways. First of all, the Sqoop gets data from all MySQL databases. Then Sqoop sends these data to HDFS. Next, Sqoop imports data from MySQL to Hive. Last, Sqoop exports data from Hive to MySQL.

B. DATA STORAGE

Hadoop is adopted by the data storage framework. Both file storage and inter-cluster communication on HDFS require data compression. The former can increase the amount of industrial data stored and reduce the cost. And the latter can reduce the bandwidth consumption of data transmission between networks and reduce data transmission time. As we know, industrial big data platform needs high stability.

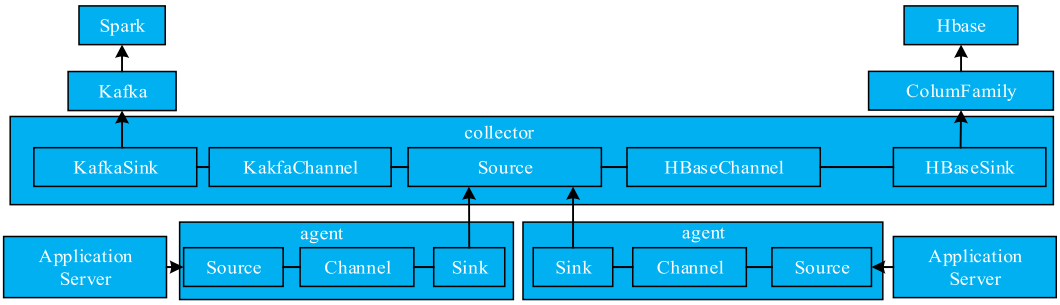


FIGURE 2. Data acquisition scheme of the data platform by Flume.

TABLE 1. Configuration information for both Kafka and HB- ase integrated with Flume.

Partition	Configuration information
agent. source	execSource.type = avro execSource.channels=kafkaChannel hbase-Channel execSource.bind = 0.0.0.0 execSource.port = 1234 execSource.selector.type = replicating
agent. channel	kafkaChannel.type = memory kafkaChannel.capacity = 100000 hbaseChannel.type = memory hbaseChannel.capacity = 100000
agent. sink	kafkaSink.type=org.apache.flume.sink.kafka. Kafka-Sink kafkaSink.kafka.topic = energy kafkaSink.kafka.bootstrap.servers= cheng01:9092,cheng02:9092,cheng03:9092 kafkaSink.channel = kafkaChannel hbaseSink.type = asynchbase hbaseSink.table = energy hbaseSink.columnFamily = info hbaseSink.zookeeperQuorum= cheng01:2181,cheng02:2181,cheng03:2181 hbaseSink.serializer.payloadColumn=acqtime, did,eid,evalue agent.sinks.hbaseSink.znodeParent = /hbase hbaseSink.serializer=org.apache.flume.sink. hbase.DsjAsyncHbaseEventSerializer hbaseSink.channel = hbaseChannel

Therefore, all the distributed frameworks integrated by the data platform are always adopted high availability mode. In the paper, the high availability data stored system of HDFS

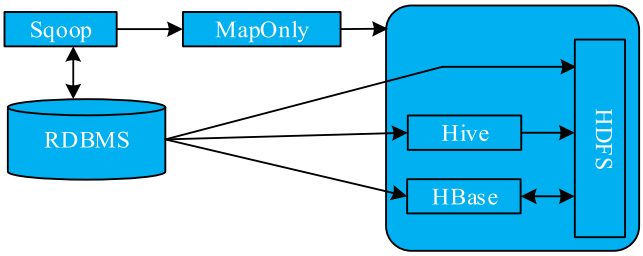


FIGURE 3. Data acquisition scheme of the data platform by Sqoop.

and HBase is built by ZooKeeper. Figure 4 shows the high availability framework of HDFS. The high availability mode of HBase is similar to that of HDFS. Besides, YARN in the later computing framework also uses the same way layout.

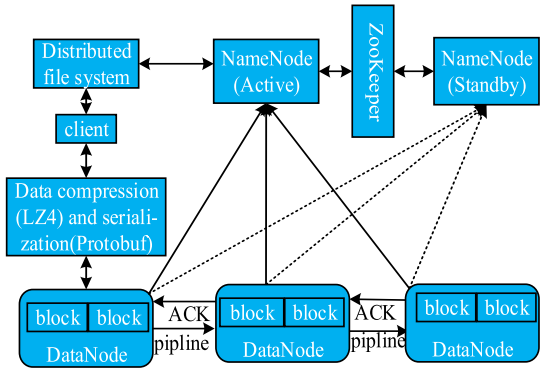


FIGURE 4. High availability data storage system.

In Figure 4, we can see two NameNode (NN) and three DataNode (DN) in the data platform because all versions of Hadoop2 only support two high-availability modes for NN. The client obtains meta information through the NN when the data is written to the DN specified by the metadata. We select LZ4 as the data compression method by evaluating various methods.

As we know, the data compressed time and size of the space compressed should be considered during the selection of compression methods. Large disc capacity means high cost of disk, compared with real-time analysis of industrial data, however, timeliness is more important than the cost of disk.

Furthermore, the timeliness is one of the most important indicators in industrial production, such as intelligent plant safety monitoring and equipment failure diagnosis. Therefore, we give priority to the data compressed time.

There are more than 20 kinds of methods for the evaluation of serialization. Among them, several methods, such as Protobuf, Kryo, Protostuff, Fasterxml, Jackson and Java default method, are selected to evaluate the serialization. In this paper, we mainly evaluate the timeliness of the data serialized method. Except for the Java default method, other methods have no significant difference in the size of the space including the memory space and the storage space. Because of the poor performance of JDK, it is just as a simple reference.

Computing system needs to support multiple types of data analysis and data visualization services, so the data computing module of the industrial big data platform needs to support a variety of data processing methods. Therefore, we evaluate four high-performance data processing frameworks. As we know, the Spark is the third-generation computing framework, whose performance is high and reliable. The Flink is called the fourth-generation computing framework and has high performance. Considering the functionality and the data processing performance of different computing frameworks, we finally chose Spark and Flink. In Table 2, We can see that the Spark and the Flink support more functions of data analysis than the other two frameworks.

TABLE 2. Data analysis methods supported by four computing frameworks.

Framework	Stream processing	Batch processing	Machine learning	SQL
Spark	√	√	√	√
Flink	√	√	√	√
Storm	√	×	×	×
Hadoop	×	√	√	√

Computing model of big data computing framework includes stateful and stateless data computation. Stateless computation is used for real-time environmental monitoring. Intelligent plant often need to accumulate statistics, such as energy consumption, equipment costs, staff working hours, etc, during the process of industrial production. The general computation method can only compute the current data, which is stateless computation. In this case, we need to use big data window mechanism to implement stateful computation.

Both the window mechanism of Spark and that of Flink support common protocol mechanism and incremental protocol mechanism. All data blocks in the window are computed by the general protocol mechanism, even if the data has been computed. Incremental protocol only needs to consider the data entering and leaving windows. The general protocol is shown in Figure 5(a) and the incremental protocol is shown in Figure 5(b). We can see that the size of window is n and the size of sliding step is m. n must be bigger than m in the normal operation of the system. The sampling interval is usually small in industrial data monitoring, such as one second or less. According to different industrial production requirements, the period of data acquisition includes minute, hour, month and year. N is the number of data per partition. So N-1 is the computation times in each partition. The number of sliding times of a conventional protocol is as follows: $(N\text{-}1)*n + n$. The number of sliding times of incremental protocol is as follows: $(N\text{-}1)*2*m + 2*m + 1$. Therefore, When m is less than half of n, the state computation adopts the common protocol. In contrast, the state computation adopts the incremental protocol. When m is equal to half of n, we can choose either of them.

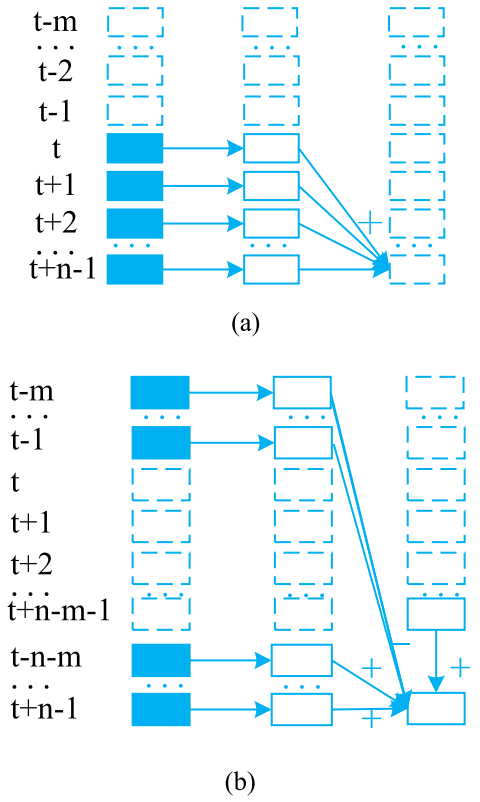


FIGURE 5. Windows mechanism of Spark and Flink computing frame- works.

IV. RESULTS OF EXPERIMENT

Data compression and serialization perform on a Windows 7 64-bits PC equipped with an Intel (R) Core (TM) i5-8400 CPU @2.8GHz 2.81GHz processor, and 4GB-RAM. Three same computers are used in the overall test of the data platform.

A. DATA COMPRESSION METHODS

DefaultCodec is the default compression method for Java. At present, LZO is the method integrated into Hadoop.

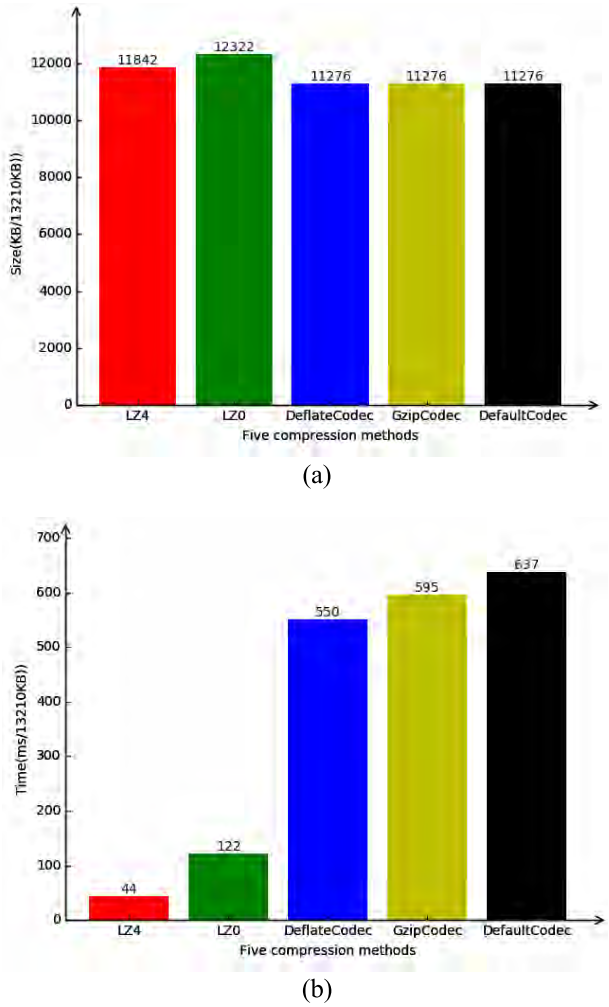


FIGURE 6. The space and time of data compressed of five compression methods.

In Figure 6(a), five data compression methods are tested with data size of 13210 KB. The size of data compressed by LZO is obviously larger than that compressed by the other four methods. Specifically, the size of data compressed by LZ4 is 96% of that compressed by LZO. Moreover, the size of data compressed by the other three methods are 91.5% of that compressed by LZO. The size of data compressed by LZ4 is slightly bigger than that compressed by DeflateCodec, GzipCodec and DefaultCodec.

Five methods of data compression take significantly different time to process the same amount of data as shown in Figure 6(b). LZ4 takes far less time than the other four methods. Specifically, the time spent by LZ4 is only 36.1% of that spent by LZO. Furthermore, LZ4 takes less than 10% of the time that DeflateCodec, GzipCodec and DefaultCodec do.

B. DATA SERIALIZATION METHODS

We evaluate six methods of data serialization, which inc-ludes Protobuf, Kryo, Protostuff, Fasterxml, Jackson and Java default method. During the evaluation, five groups of serialized object of five orders of magnitude between 10,000 and 100 million are tested by each data serialization method. Moreover, each group of serialized object is tested 10 times, and the average results are listed in Table 3.

TABLE 3. The data processing time of different data serializa- tion methods in different order of magnitude.

Method	1×10^4	1×10^5	1×10^6	1×10^7	1×10^8
Kryo(s)	0.3	0.44	2.33	18.42	215
Protostuff(s)	0.11	0.24	1.11	8.37	87
Protobuf(s)	0.45	0.62	1.2	8.73	82
Fasterxml(s)	1.22	1.56	3.69	24.2	245
Jackson(s)	1.7	1.89	3.58	18.67	246
Default method(s)	1.56	7.22	31.56	275.7	---

Table 3 shows that the time spent by Kryo, Protostuff and Protobuf are all significantly less than that done by the other three methods when the number of serialized object tested is 1×10^4. However, with the increasing amount of data, the Kryo takes longer and the Protobuf takes the shortest time.

In order to make the test data more intuitive, we draw a broken line graph and draw a broken line graph every three orders of magnitude. The broken line graph of the data in Table 3 is shown in Figure 7. Kryo can serialize faster with fewer objects in Figure 7(a). Kryo, Protobuf and Protostuff changed significantly when the processing object was about 10000. As the number of objects processed increases, processing time increases sharply. However, Protobuf and Protostuff show a downward trend with the increase of processing object data in Figure 7(a). When 10 million objects are processed, Kryo's performance is very poor and Protostuff's and Protobuf's have very similar performance in Figure 7(b). Protobuf performs better than Protostuff when the amount of data processed is more than 100 million in Figure 7(c). By evaluating above five data serialization methods, we finally choose Protobuf.

C. DATA PLATFORM

The data platform includes three parts: data acquisition module, data computation module and data presentation module. It takes eight steps to set up a data acquisition, storage, computation and visualization of the platform. The display interface of this platform is shown in Figure 8.

–First, Java Tools and SSH are configured.

–Second, distributed coordination service cluster is built. Zookeeper is integrated into server cluster.

–Third, the data structure of intelligent equipment is analyzed. Setting Topic and Docker. Equipment data acquisition location is specified. Flume and Kakfa are integrated.

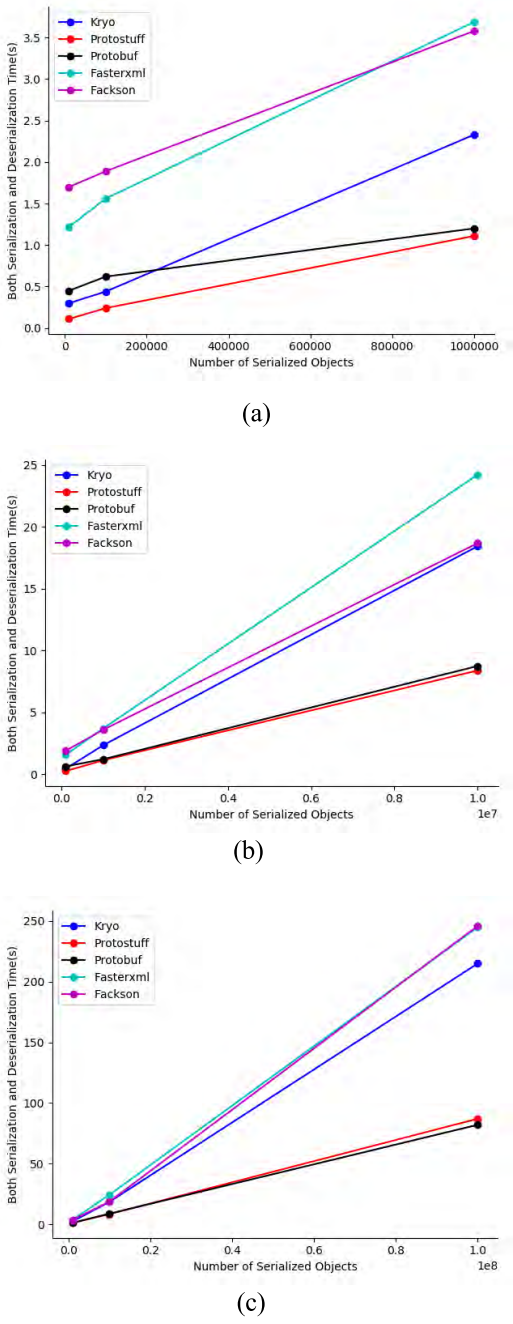


FIGURE 7. The time of data serialized of five methods at data level for different processing objects.

–Fourth, Hadoop is integrated into the server cluster by HDFS and YARN of the configuration files, and adding packages optimized of Java to library files to configure to use.

–Fifth, Hive and HBase are integrated into the built server logical file system. Setting table and columnFamily. Serializer used the optimal Protobuf method. Sink is HBaseSink. By some of the above operations, Flume and HBase are integrated.

–Sixth, Spark and Flink are integrated into the built server logical file system. Adding Java packages optimized to library files to configure to use.

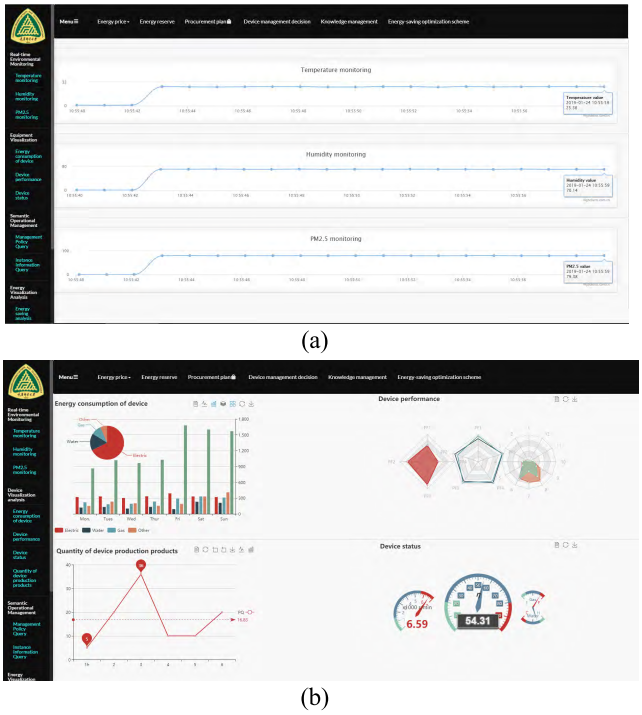


FIGURE 8. Visual display interface of industrial big data platform.

–Seventh, Hue is integrated, at the same time, industrial data visualization front-end is designed by Django frame- work.

–Eighth, the data platform is tested as a whole.

Figure 8(a) is a real-time monitoring interface. Monitoring information includes plant temperature, humidity and PM2.5. Three kinds of information are acquired per second. The monitoring interface displays information for a time span of 20s to prevent monitoring personnel from missing data. Figure 8(b) is visualization of energy consumption, equipments performance, equipment status and production efficiency after data computation and data mining.

V. DISCUSSION

Recently, there are a number of studies on data acquisition and optimization of computing framework methods, and most of these methods provided good performance in the field of Internet. We consider the characteristic of real-time and stability in industrial production in our research. Therefore, Data platform optimization includes data acquisition, data compression, data serialization, and system stability.

With the development of intelligent plant, the data acquisition volume of plants increases rapidly. Flume can meet the needs of data acquisition in plants. In the research of the method of data stored, we compress data by LZ4 and serialize data by Protobuf. Compared with the original method of Hadoop integration, the time of processing data is greatly reduced and the processing effect is better. Both Spark and Flink are used in computing framework. According to the size of sliding window, we adopt appropriate window mechanism

to compute the data for certain time periods. Furthermore, the selection method is given.

VI. CONCLUSION

In this paper, we start with the process of industrial data processing and the requirement of industrial data analysis. High availability file stored system is built to avoid single point failure. In the process of building file stored system, the performance of LZO provided by the framework is inferior to LZ4 in the time and the size, and the time spent by LZ4 is only 36% of that spent by LZO. The size of data compressed by LZ4 is 96% of that compressed by LZO.

We compare the optimal serialization methods provided by big data framework with other high performance methods to optimize the existing framework's method. Compared with the other serialization methods integrated by Hadoop and Spark, Protobuf performs better and better in data processing as the number of serialized objects increases. Compared with Java defaults, Protobuf takes only 7% of the time to process data. At same time, our data platform has many data analysis functions including machine-learning, Finite Element Analysis, Optimization and Knowledge Mapping.

REFERENCES

[1] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "A knowledge-based platform for big data analytics based on publish/subscribe services and stream processing," *Knowl.-Based Syst.*, vol. 79, pp. 3–17, May 2015.

[2] R. Y. Zhong, C. Xu, C. Chen, and G. Q. Huang, "Big data analytics for physical Internet-based intelligent manufacturing shop floors," *Int. J. Prod. Res.*, vol. 55, no. 9, pp. 2610–2621, 2017.

[3] P. Lade, R. Ghosh, and S. Srinivasan, "Manufacturing analytics and industrial Internet of Things," *IEEE Intell. Syst.*, vol. 32, no. 3, pp. 74–79, May/Jun. 2017.

[4] X. Yi, F. Liu, J. Liu, and H. Jin, "Building a network highway for big data: Architecture and challenges," *IEEE Netw.*, vol. 28, no. 4, pp. 5–13, Jul. 2014.

[5] B. Meredig, "Industrial materials informatics: Analyzing large-scale data to solve applied problems in R&D, manufacturing, and supply chain," *Current Opinion Solid State Mater. Sci.*, vol. 21, no. 3, pp. 159–166, 2017.

[6] P. Basanta-Val, "An efficient industrial big-data engine," *IEEE Trans. Ind. Informat.*, vol. 14, no. 4, pp. 1361–1369, Apr. 2018.

[7] X. He, K. Wang, H. Huang, and B. Liu, "QoE-driven big data architecture for smart city," *IEEE Commun. Mag.*, vol. 56, no. 2, pp. 88–93, Feb. 2018.

[8] D. Cheng, X. Zhou, P. Lama, J. Wu, and C. Jiang, "Cross-platform resource scheduling for spark and mapreduce on YARN," *IEEE Trans. Comput.*, vol. 66, no. 8, pp. 1341–1353, Aug. 2017.

[9] X. Yan, J. Zhang, Y. Xun, and Q. Xiao, "A parallel algorithm for mining constrained frequent patterns using MapReduce," *Soft Comput.*, vol. 21, no. 9, pp. 2237–2249, 2017.

[10] H. Huang, S. Guo, and K. Wang, "Envisioned wireless big data storage for low-earth-orbit satellite-based cloud," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 26–31, Feb. 2018.

[11] X. Wang, Y. Zhang, V. C. M. Leung, N. Guizani, and T. Jiang, "D2D big data: Content deliveries over wireless device-to-device sharing in large-scale mobile networks," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 32–38, Feb. 2018.

[12] M. R. Palattella *et al.*, "Internet of Things in the 5G era: Enablers, architecture, and business models," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 510–527, Mar. 2016.

[13] L. Kong, D. Zhang, Z. He, Q. Xiang, J. Wan, and M. Tao, "Embracing big data with compressive sensing: A green approach in industrial wireless networks," *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 53–59, Oct. 2016.

[14] Y. Lei, F. Jia, J. Lin, S. Xing, and S. X. Ding, "An intelligent fault diagnosis method using unsupervised feature learning towards mechanical big data," *IEEE Trans. Ind. Electron.*, vol. 63, no. 5, pp. 3137–3147, May 2016.

[15] G. Gui, H. Huang, Y. Song, and H. Sari, "Deep learning for an effective nonorthogonal multiple access scheme," *IEEE Trans. Veh. Technol.*, vol. 67, no. 9, pp. 8440–8450, Sep. 2018.

[16] T. Zhou, S. Yang, L. Wang, J. Yao, and G. Gui, "Improved cross-label suppression dictionary learning for face recognition," *IEEE Access*, vol. 6, pp. 48716–48725, 2018.

[17] J. Pan, Y. Yin, J. Xiong, L. Wang, G. Gui, and H. Sari, "Deep learning-based unmanned surveillance systems for observing water levels," *IEEE Access*, vol. 6, no. 1, pp. 73561–73571, 2018.

[18] M. Liu, J. Yang, T. Song, J. Hu, and G. Gui, "Deep learning-inspired message passing algorithm for efficient resource allocation in cognitive radio networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 1, pp. 641–653, Jan. 2019.

[19] S.-I. Lee *et al.*, "A machine learning approach to integrate big data for precision medicine in acute myeloid leukemia," *Nature Commun.*, vol. 9, no. 1, 2018, Art. no. 42.

[20] Y. Shi *et al.*, "Machine-learning variables at different scales vs. Knowledge-based variables for mapping multiple soil properties," *Soil Sci. Soc. Amer. J.*, vol. 82, no. 3, pp. 645–656, 2018.

[21] A. E. Saddik, "Digital twins: The convergence of multimedia technologies," *IEEE Multimed.*, vol. 25, no. 2, pp. 87–92, Apr./Jun. 2018.

[22] D. Chen *et al.*, "Real-time or near real-time persisting daily healthcare data into HDFS and elasticsearch index inside a big data platform," *IEEE Trans. Ind. Informat.*, vol. 13, no. 2, pp. 595–606, Apr. 2017.

[23] Z. Ji, I. Ganchev, M. O'Droma, L. Zhao, and X. Zhang, "A cloud-based car parking middleware for IoT-based smart cities: Design and implementation," *Sensors*, vol. 14, no. 12, pp. 22372–22393, 2014.

[24] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big data analytics framework for peer-to-peer botnet detection using random forests," *Inf. Sci.*, vol. 278, pp. 488–497, Sep. 2014.

[25] H. Dan *et al.*, "Achieving load balance for parallel data access on distributed file systems," *IEEE Trans. Comput.*, vol. 67, no. 3, pp. 388–402, Mar. 2018.

[26] M. Shahabinejad, M. Khabbazian, and M. Ardakani, "On the average locality of locally repairable codes," *IEEE Trans. Commun.*, vol. 66, no. 7, pp. 2773–2783, Jul. 2018.

[27] A. P. Mukherjee and S. Tirthapura, "Enumerating maximal bicliques from a large graph using MapReduce," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 771–784, Sep./Oct. 2017.

[28] N. Kari, K. Latif, Z. Anwar, S. Khan, and A. Hayat, "Storage schema and ontology-independent SPARQL to HiveQL translation," *J. Super Comput.*, vol. 71, no. 7, pp. 2694–2719, 2015.

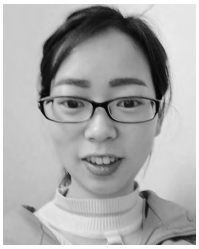
[29] M. Maas, K. Asanović, T. Harris, and J. Kubiatowicz, "Taurus: A Holistic language runtime system for coordinating distributed managed-language applications," *Acm SIGPLAN Notices*, vol. 51, no. 4, pp. 457–471, 2016.

[30] D. García-Gil, S. Ramírez-Gallego, S. García, and F. Herrera, "A comparison on scalability for batch big data processing on Apache Spark and Apache Flink," *Big Data Anal.*, vol. 2, no. 1, pp. 1–11, 2017.

[31] Z. Ma, Z. Sheng, and L. Gu, "DVM: A big virtual machine for cloud computing," *IEEE Trans. Comput.*, vol. 63, no. 9, pp. 2245–2258, Sep. 2014.

DAOQU GENG received the Ph.D. degree in physical electronics from the Graduate University of Chinese Academy of Sciences, Beijing, China, in 2011. He is currently an Associate Professor with the College of Automation, Chongqing University of Posts and Telecommunications, Chongqing, China. His current research interests include industrial big data, the industrial Internet of Things, and semantic sensor networks.

CHENGYUN ZHANG was born in Enshi, Hubei, China, in 1993. He received the B.S. degree in electronic information engineering from Yangtze University, Hubei, in 2017. He is currently pursuing the M.S. degree in control engineering with the Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include machine learning, big data, and data intensive high performance computing, and the industrial Internet of Things.



CHENGJING XIA was born in Xiangyang, Hubei, China, in 1994. She received the B.S. degree in electronic information engineering from Yangtze University, Hubei, in 2017, where she is currently pursuing the M.S. degree in signal and information processing. Her research interests include machine learning, big data, and data intensive high performance computing.

XUE XIA was born in Songyuan, Jilin, China, in 1995. She received the B.S. degree in automation from the Chongqing University of Posts and Telecommunications, in 2017, where she is currently pursuing the M.S. degree in control science and engineering. Her research interests include semantic web and the industrial Internet of things. She currently studies the application of semantic web in video retrieval and big data.

QILIN LIU received the B.S. degree in automation engineering from Nanyang Normal University, Henan, China, in 2016. He is currently pursuing the M.S. degree in control engineering with the Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include semantic web, the Internet of Things technology, and big data.

XINSHUAI FU was born in Linyi, Shandong, China, in 1993. He received the B.S. degree in automation from Weifang Univerity, Shandong, in 2017. He is currently pursuing the master's student in control science and engineering with the Industrial Internet of Things and Networked Control, Ministry of Education, Chongqing University of Posts and Telecommunications. His research interests include semantic web and big data.