IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 16, 2019, accepted February 18, 2020, date of publication February 27, 2020, date of current version March 6, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2976660

Research on a Distributed Processing Model Based on Kafka for Large-Scale Seismic Waveform Data

XU-CHAO CHAI[1], QING-LIANG WANG[1], WEN-SHENG CHEN[1], WEN-QING WANG[1], DAN-NING WANG[1], AND YUE LI[2]

[1]The Second Monitoring and Application Center, CEA, Xi'an 710054, China
[2]School of Astronautics, Northwestern Polytechnical University, Xi'an 710075, China

Corresponding author: Yue Li (yueli@nwpu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFC1500102, in part by the National Natural Science Foundation of China under Grant 61803311, in part by the China Analog Seismic Record Rescue Project under Grant 152038000000180003, in part by the China Integrated Geophysical Field Observation Project under Grant 201508009, and in part by the Science for Earthquake Resilience under Grant XH16056Y and Grant XH18070.

ABSTRACT For storage and recovery requirements on large-scale seismic waveform data of the National Earthquake Data Backup Center (NEDBC), a distributed cluster processing model based on Kafka message queues is designed to optimize the inbound efficiency of seismic waveform data stored in HBase at NEDBC. Firstly, compare the characteristics of big data storage architectures with that of traditional disk array storage architectures. Secondly, realize seismic waveform data analysis and periodic truncation, and write HBase in NoSQL record form through Spark Streaming cluster. Finally, compare and test the read/write performance of the data processing process of the proposed big data platform with that of traditional storage architectures. Results show that the seismic waveform data processing architecture based on Kafka designed and implemented in this paper has a higher read/write speed than the traditional architecture on the basis of the redundancy capability of NEDBC data backup, which verifies the validity and practicability of the proposed approach.

INDEX TERMS HBase, Kafka, key-value, spark streaming, seismic waveform data.

I. INTRODUCTION

Seismic observation waveform data comes from the acquisition and forwarding of seismic sensors at stations of seismic networks, and is gradually aggregated to form real-time stream data by Jopens software [1], [2]. Finally, the real-time stream data is archived and written to the disk array, which is the main storage form of seismic waveform data at present. One of the main tasks of the NEDBC is to back up the two types of seismic data: real-time streaming data of seismic waveform data, and NAS file data provided by traditional disk arrays.

In the era of big data, data mining for different dimensions of data can effectively promote the progress of the industry. With the improvement of seismic observation methods and the encryption of observation networks, the amount of seismic observation data is increasing day by day [3], [4]. Data mining or correlation analysis of seismic data stored in the form of files requires a great amount of system overhead and is difficult to complete. Therefore, it is necessary to store the aforementioned seismic data in the form of databases to meet data application requirements. However, storing traditional seismic data in relational databases, such as the storage of data in traditional two-dimensional tables of precursory data, cannot solve the requirements on reading, writing, and calculations caused by seismic big data, such as the performance bottleneck of table join operations, and the underlying storage of data tables, I/O bottlenecks and so on. Therefore, it is necessary to select a large data storage architecture that is conducive to massive data mining analysis for the storage of seismic data. After Google published a paper describing HDFS and BigTable in [5], the integration of

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

distributed file systems and NoSQL [6] distributed databases are increasingly adopted to take the place of the relational database management system (RDBMS) [7], [8] For example, the Apache community's HBase (Hadoop Database) [9], Facebook's Cassandra, Amazon's Dynamo, etc., all of them use the column database Key-Value data model, of which the Apache community's HBase is the most widely used. Due to its high concurrency, high scalability and high availability, HBase has been extensively applied in Internet industries such as Tencent, Alibaba and Baidu to solve large-scale data storage and high-speed query services. HBase is also an effective way to solve the issues on read/write efficiency, scalability, analytic mining, analytic prediction for seismic big data storage, and has found related applications in the industry [10]–[12].

In the Spark framework [11], the Spark Streaming component is used to process streaming data, and the Kafka [13] message system of the distributed processing architecture is used to write the data to the HBase database with high concurrency. This is a common method for processing data streams under the big data platform in recent years. HBase is a popular distributed, scalable, and column-oriented database storage system built on Hadooop framework, and is a NoSQL database with fast and real-time random data access [14]. So far, the theoretical model based on which seismic waveform data is written into HBase databases in Key-Value form has been applied in the field of seismic industry, which indicates that the use of HBase column-oriented databases to store seismic waveform data can effectively improve the efficiency of such data storage. Reference [15] have conducted comparative tests on structured and unstructured data stored in MySQL and HBase seismic data, which proves that HBase storage seismic data has a very significant improvement in both storage efficiency and query efficiency. However, these methods have not yet proposed solutions for HBase storage of massive seismic data, so that it is impossible to achieve high-speed writing of seismic data by using cluster concurrent processing. To solve this problem, we first introduce the strategy and basis for selecting distributed storage system in NEDBC, and introduce the Kafka message system of the big data distributed processing architecture. Then, we design and propose a seismic waveform data processing architecture based on Kafka message queue and Spark Streaming real-time computing framework. The data processing architecture analyzes seismic waveform data from traditional disk arrays or real-time streaming data into Key-Value format and writes them to HBase distributed databases, solving the speed bottleneck problem on writing seismic waveform data into a distributed database, thus improving analysis, storage, and query efficiency of such data.

In this paper, the NEDBC seismic waveform data storage process model and HBase database structure are designed, a Kafka-based seismic waveform data processing model is constructed, and the seismic data HBase reading/writing method is proposed. The main advantages and contributions of this paper can be summarized as follows:

i. In order to write large-scale seismic waveform data into HBase database at high speed, this paper firstly designs the RowKey rule of HBase database for seismic data, and determines the HBase storage format of seismic data.

ii. We implement Key-Value parse for miniSEED [16] seismic waveform data: (1) Periodic truncation and msgKey-msg [15] parse of seismic data via C language are realized; (2) Parse the msgKey-msg format into the Key-Value format and conduct HBase inbound. (3) in order to further improve the efficiency of writing seismic waveform data into HBase, compared with the current storage solutions in the seismic industry [10], we propose the data inbound process based on Kafka production-consumption model [13] and Spark Streaming real-time computing framework for seismic waveform data, achieve the purpose of the cluster processing seismic waveform data concurrently, and effectively increase the speed of concurrently writing seismic data into HBase.

iii. We perform the inbound performance test on the seismic data analysis storage architecture proposed in this paper with the big data storage computing cluster of the NEDBC, and compare it with the mode under which seismic data is written into HBase with one machine. It is verified that the seismic data storage scheme based on the Kafka production-consumption model can improve the efficiency of writing seismic data into HBase.

To the authors' best knowledge, few works on using cluster advantages to concurrently process seismic data have been achieved, which motivates this study.

The rest of this paper is organized as follows. Section 2 proposes the structure and seismic waveform data storage format of HBase database for the NEDBC; Section 3 design proposes the HBase storage principle and implementation method based on Kafka and Spark Streaming seismic waveform data processing. Section 4 analyzes and evaluates the results of data reading and writing experiments designed in this paper, and summarizes and forecasts.

II. SEISMIC DATA STORAGE COMPARISIONS UNDER DIFFERENT ARCHITECTURE

Due to advances in observation methods and data acquisition capabilities, seismic observation data has been greatly improved in terms of sampling rate and sampling accuracy. On the other hand, due to the dramatic increase of seismic data volume, the data storage methods using traditional disk arrays are unable to meet the demand in terms of storage efficiency, scalability, reliability, cost performance, and etc,. For example, the disk array RAID technology only allows disks in the redundancy file system to be invalid, and there is no rule to avoid severs done. The emergence of Hadoop solves the problem of massive seismic data storage and processing in the era of big data. Hadoop is a framework that supports distributed storage and computing. Its core component, the HDFS distributed file system, can effectively improve the read and write capabilities of massive seismic data, not only supporting node-level system failures, but also allowing the number of disk failures. There is a huge improvement

over disk array storage [17], [18]. HDFS distributed storage system has become the de facto standard for big data storage and can be used for online storage of massive file data.

In a distributed storage file system, data redundancy is the most fundamental factor in guaranteeing system reliability and improving high availability [19]. In the same file system, the effective redundancy of data for a data file can be realized by the method of storing the data file in different nodes, and original data is reconstructed in the case of partial data failure and achieving the purpose of usability [20], [21]. Redundancy strategies of distributed file systems need to determine two points: one is to build redundant data strategies, and the other is to reconstruct data strategies. Currently, the redundancy strategies used by mainstream distributed file systems are the two forms of replication and erasure coding: the former uses a replica mechanism to store data files across racks to achieve high availability redundancy; the latter stripes the data entering file systems, then calculates the number of redundant stripes according to redundancy ratio, and finally is saved in different nodes [5]. Due to the redundancy of data, the system can tolerate the failure of certain nodes, thereby achieving a certain degree of system reliability in the set of unreliable nodes. However, when system scale is expanded and long-term reliable operation is required, theoretical analysis of the reliability of system data redundancy is particularly important. Yet, a large number of literatures [22]–[24] analyzed the data availability of different redundancy strategies, the node failure and repair process of distributed storage systems, and theoretically calculated the probability of storage system failure, the time of reliable operation, the number of required replicas, the life cycle of the system, etc., which proves that the erasure coding technology of HDFS distributed file systems is superior to the copy mechanism in terms of system availability, storage space utilization, system energy consumption, etc..

Relational databases such as MySQL and Oracle have problems in expansion difficulties and complex maintenance, and cannot handle hundreds of millions or even hundreds of billions of records. HBase is a high-reliability, high-performance, column-oriented, scalable distributed database, whose underlying file system uses HDFS to expand the system linearly by adding nodes, and provides large-scale concurrency capability which processing massive seismic data. HBase is the database in the key-value form. The tables in HBase have the following characteristics.

i. Large volume: a table can have hundreds of millions of rows, millions of columns, and seismic observation data with massive time series can be stored in record form;
ii. Column-oriented: column-oriented (family) storage and access control, column (family) independent retrieval.
iii. Sparse table design: since null (NULL) columns do not occupy storage space, the table can be designed to be very sparse.

TABLE 1. Producer of Kafka information format.

msgKey	Network _ Station _ Channel _ Instrument ID _ Start Time _ Sample rate(S)_ Data integrity identifier(C) _ Missing numerical statistics(M)
msg	Data block for the msgKey

TABLE 2. Format of key-value data stored in HBase.

Table Element of Hbase	**Content in the HBase table element**			
RowKey	Network	Station	Channel	StartTime
Column	Base name			
Value	MiniSEED block data (time series data) instrument ID start time sampling rate Data integrity identifier Missing numerical statistics			

iv. Each recorded data can have multiple versions, and the version number is automatically assigned (timestamp) by default, which is operable and traceable for modifying data operations.
v. The data in HBase is all NoSQL records with no special type. Based on the above characteristics, and considering the advantages of HBase own scalability and execution efficiency, the data storage structure of NEDBC selects HBase distributed databases and HDFS file system combination strategy based on disk array erasure coding.

III. HBASE STUCTURE OF SEISMIC WAVEFORM DATA

In order to store massive seismic waveform data in the format of NoSQL, RowKey format of HBase database [9] is designed for seismic data in this section, and the seismic data inbound is conducted in Key-Value format illurstrated in TABLE 1.

As shown in TABLE 2, the same format data formed by parsing the seismic waveform file and the real-time streaming data is stored in the HBase column database. HBase's Key-Value data structure consists of the row key named RowKey, and the column cluster Value. Among them, RowKey is the basis of data design, similar to the primary key in the relational database, which will determine the efficiency of data query and analysis; Value can be regarded as the data body part corresponding to Rowkey, and consists of column name (Column family: column name) and column family.

In order to realize the high-speed writing of seismic waveform data into HBase database, this paper introduces Kafka production-consumption model to process the seismic waveform data concurrently. Kafka is a high-throughput distributed publishing and subscribing message system that handles all action flow data in consumer systems. The goal is to unify online and offline message processing through Hadoop parallel loading mechanism, and also is to provide the ability to process data in real time via clusters. The real-time data stream or archive file of seismic waveform data is

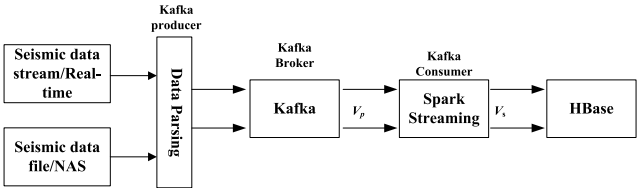


FIGURE 1. Overall structure of data storage.

TABLE 3. Design of HBase table.

Table Name	Interpretation
WaveForm_head	Table for storing header information of seismic data file
WaveForm_YEAR	Seismic data year storage table

parsed into specified format data, and the Kafka message system is forwarded in parallel to the client Spark Streaming as processing object, which will greatly improve the concurrent data processing capability of the Spark cluster [25], [26].

The data parsing is completed before entering the Kafka message system: the input is file data or real-time streaming data, as shown in FIGURE 1. After reading the configuration and data processing, the system forms the data in the specified format msgKey: msg as shown in TABLE 1 and sends it to the Kafka message system. The main purpose of recording the complete identifier and missing number statistics of data in this paper is to collect statistics on the quality of seismic data received by NEDBC in HBase.

The business logic of seismic waveform data conducts query through networks, stations, channels and start time. In this paper, these four items are combined into Rowkey, and the 100 Hz sampled time series specific data is stored in Value of HBase table. According to the read/ write demand characteristics of seismic waveform data and safety considerations for data, NEDBC divides the seismic data on a yearly basis. As shown in TABLE 3, the WaveForm_head data table stores the header file information of seismic waveform data, and the WaveForm_YEAR data table stores the seismic data value chronologically. The table name design rule is the last 4-byte YEAR is the year of the data stored in the table.

According to the above-mentioned HBase design rule, the RowKey consisting of <Network> <Station> <Channel> <Start Time> can conduct combination query according to the requirements in seismic data tables with different years. For example, to query a certain period of time of a certain channel at a certain station, we can specify the starting RowKey as <Station> <Channel> <Start Time>, and end RowKey as <Station><Channel><End Time>, where the end time can be obtained according to the sampling rate of seismic data.

Data storage is conducted after spark streaming obtains the Kafka consumer information: the Kafka consumer information is processed by the spark cluster to form the Key-Value format data, which is written into the HBase. The HBase storing RowKey and Value fields/formats are shown in TABLE 2.

TABLE 4. HBase seismic data record.

Rowkey	Column	Value
AAII_CN_BHE_2018-01-03-23-21	BASE: seismic_data	Miniseed data block of BHE
AAII_CN_BHN_2018-01-03-23-21	BASE: seismic_data	Miniseed data block of BHN
AAII_CN_BHZ_2018-01-03-23-21	BASE: seismic_data	Miniseed data block of BHZ

As described above, the network, station, channel, and start time are four fields of RowKey, and the other fields of msgKey in Kafka producer information are parsed and added to the Value data block:

a) Rowkey: Network_Station_Channel_yyyy-mm-dd-hh-mm],
b) Column: Base name,
c) Value: MinSEED block data of channel.

And the final data record format of HBase is shown in TABLE 4.

IV. KAFKA-BASED SEISMIC WAVEFORM DATA PROCESSING MODEL AND HBASE READ/WRITE IMPLEMENTATION

To write seismic data in the Key-Value format into HBase database at a high speed, in this section, we introduces the Kafka message queue model under big data processing architecture, and designs the seismic data inbound scheme based on Kafka production consumption model. The concurrent high-speed inbound of seismic data is realized via real-time computing framework Spark Streaming. In this paper, HBase's Java client is used for data outbound processing, and meanwhile data consistency check is performed before and after data enters HBase.

A. SEISMIC WAVEFORM DATA ANALYSIS

As mentioned above, real-time streaming data and archived data files (miniSEED) are generated during the production of seismic waveform data. This paper reads and parses the real-time stream and the history file waveform data to form the msgKey-msg as described above, which then is sent the data processing system shown in FIGURE. 1.

The file parsing process is shown in FIGURE 2. The full amount of miniSEED data files archived by NEDBC are processed in the ascending order of each station to form a historical full-scale continuous waveform data. The parsed data becomes the msgKey:msg format, and according to fixed time periods, data information is assembled into Kafka message producer information and posted to the Kafka system.

Input the same network, station and channel sub-block into the data parsing process, and output the kafka producer message msgKey: msg. The main variables of the system are

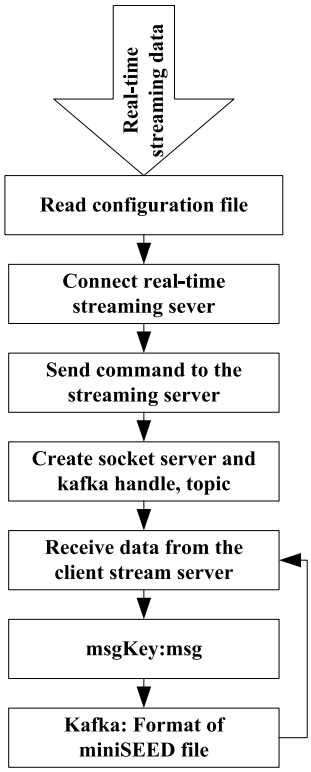


FIGURE 2. Data analysis flow.

TABLE 5. Variable name and meaning.

Name of Variable	Description
data_hdr	Structure storage of Miniseed block
buffer	Pointer of stream data cache
t	Time to truncate the data stream into an HBase record
diffLen	The time difference between the start time of the current data block and the previous sub-block
blktime	Start time of each data sub-block ε
newtime	The last sub-block end time in memory
curblock ->last_time	The start time of a sub-block on the current channel in memory
curblock_start_time	The start time of the current channel in memory
duration	The default time interval (1min). If the time interval is exceeded, the current sub-block is not continuous with the previous sub-block and needs to be placed in the next time interval.
curblock ->last_nsamples	The number of samples appended to the last sub-block in memory
curblock ->last_sample_rate	The sample rate appended to the previous sub-block in memory
m_iTimeDuration	Intercept time interval

shown in TABLE 5 and the implementation process is shown in TABLE 6.

Step1. Take the current sub-block start time blktime.

Step2. Calculate the end time of the previous data sub-block in memory newtime.

TABLE 6. Process the same network station channel sub-block process process_same.

```
Input: Sub-blocks of the same network, station, and channel
Output: message of KAFKA producer msgKey:msg
Process_same(struct input_data_hdr *data_hdr, char *buffer)
{       blktime=data_hdr->time; // Assign the current sub-block
start time to blktime
  newtime =curblock->last_time; // Calculate the end time of this
channel in memory
  Compare(blktime, newtime, &diffLen); // Calculate the difference
between the end time of the block in memory and the current block start
time
  if(abs(diffLen) <=t)
{       Compare(blktime, curblock->start_time, &diffLen);
        if(abs(diffLen) > t)){
          if(diffLen<= 0) // Less than 0, the data has partial
overlap, send directly
                sendDataToKafka(curblock); // Send completed,
reset structure
          else Count the number and number of missing if the data
is missing, then send to KAFKA and reset the structure }
        Update in-memory block information
} else{
        Compare(blktime,newtime)
        sendDataToKafka(curblock); // Send in-memory data blocks
to KAFKA
        Reset data block }
}
```

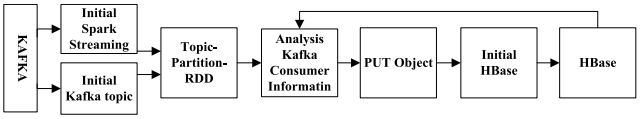


FIGURE 3. Spark streaming data storage architecture diagram.

Step3. Calculate the in-memory block end time and the current block start time difference diffLen.

Step4. If diffLen<= t, compare the in-memory block start time blktime with the current block start time curblock->start_time difference; If the difference is greater than the cycle time, first send the block data in the memory, then save the current block to the memory, and update the memory block information; otherwise, the current block is added to the memory to update the data block information in the memory. If diffLen>0, the data is missing, the calculated data is missing, then sent, the structure is reset, and the in-memory block information is updated.

Step5. If diffLen> t, calculate the data missing between the current data block and the in-memory data block, and update the memory data block information: Compare(blktime, newtime). Then send an in-memory data block (sendDataToKafka) and reset the data block.

B. SPARK STREAMING PROCESSING FLOW

As shown in FIGURE 3, Spark Streaming as the consumer actively gets msgKey:msg information from the Kafka message queue. After generating consumer information, Kafka enters the Spark Streaming distributed cluster processing process. The Spark node parses the Kafka message to form data in the Key-Value format and writes the data into the HBase

database, thereby completing the storage process of seismic waveform data.

The storage process mainly involves Spark Streaming, Kafka Consumer, and HBase. Spark Streaming is a framework set up on Spark for processing Stream data. The Stream data is divided into small time slices to process a small part of data in a batch-like manner. Low-latency of Spark processing engine (100ms+) is used for real-time data processing; Spark essential data model RDD, an elastic distributed data set, is suitable for distributed data processing, while RDD pedigree mechanism (dependencies among RDDs) makes Spark highly tolerate faults and has the RDD functions to recalculate partial failed tasks; Using Kafka consumer groups and specifying the number of topic consumers per Kafka, i.e. the number of threads that consume topic messages at the same time, can quickly consume the messages in the topic; The consumed message is partitioned, each area processes the message, generates a List<Put> object of the HBase storage data, and stores it in the HBase correspondence table.

C. INTERCEPTION OF SEISMIC WAVEFROM DATA

The interception of seismic data is completed before the real-time stream or file data passes through the kafka message system and is written to HBase. The input of this process are the real-time streaming data and data truncation interval t; the kafka producer information in the form of msgKey:msg is output according to the network, station, channel and time. The data truncation process is shown in TABLE 7.

Step1. Initialize the kafka handle and create a kafka topic.

Step2. The real-time stream data is cyclically processed for each sub-block ε (512 Byte), and the truncation time parameter t is initialized, and the parsing link for each sub-block ε is entered.

Step3. If the current sub-block and the parsing record are the same network station channel last time, enter the processing and merge process of the same network station channel (for the process, refer to the previous Process_same).

Step4. If the current sub-block ε is new (curblock equals to NULL), parse each data sub-block, create a new structure data_hdr, create a cache block according to the station and channel, and proceed to step 3).

Step5. Compare the network, station, and channel information. If the current sub-block parsing record is not the same as the last network station channel, look for the same network, station, and channel data block in the memory block list miniblock_head. If it exists, it is processed according to the data of the same station channel (sent to kafka after reaching the specified time length t). If it does not exist, create a cache block and go to the same step 3).

D. OUTBOUND AND VERIFICATION

In order to verify data output efficiency of NEDBC distributed storage platform, we test the HBase distributed database outbound, data consistency, and etc,.

TABLE 7. Data truncation process.

Input : Real-time streaming data (sub-block ε of each data), duration data truncation
Output: Messages in the form of networks, stations, channels and time, sent to the Kafka system

```
While (recv)
                    // Receive real-time streaming data from the
socket
{           Initial: Kafka, Kafka topic
            // Initialize kafka and create a topic for Kafka
  process_stream_data(dealBuf, SOCKET_READ_LEN)
  {  // Handling real-time streaming data
     for ε  // For each data sub-block
          input_data_hdr = (struct input_data_hdr *) input_data_ptr;
          // Parsing data sub-blocks
          process(struct input_data_hdr *data_hdr, char *buffer)
         {
              // Merge and send data
                   if(curblock == NULL)
                   {// Create a new structure
                              create_mini_block(data_hdr, buffer);
                              // Create a cache block according to
the network, station, channel
                   }
                   else if(the network of the current data block, the
station, the channel information is different from the curblock cache
information.)
                   {
                        while(block!= null)
                        {
       Determine if there are any networks, stations, and channels in
       the memory that are the same as the current sub-blocks.
                        }
                        If (doesn’ t exist)
                          create_mini_block(data_hdr, buffer); //
Create a cache block
                        else
                           process_same(data_hdr, buffer) //
Processing data for the same station channel
                   }
                    else
                   {
                      process_same(data_hdr, buffer);// Enter the
same network station channel analysis process
                   }
          }
     }
}
```

1) OUTBOUND PROCESS

The NEDBC seismic waveform data is stored in the HBase database in the form of Key-Value records. The Java client sets the interface related parameters as shown in TABLE 8, via HBase static query interfaces to initialize the HBase connection and to enter the data outbound process, and to write the data into storage system (as shown in FIGURE 4). The return type of the HBase query interface is unified as ResponseBean, and the DataObject types of the returned ResponseBean are all String, which is the name of generated files.

First set up the HBase initialization connection, and set the HBase client parameters, query through the network, station, channel and time conditions. Second, thread pools are created and started by the clusters. Third, manage hTable to

TABLE 8. Interface formats of HBase.

Interface parameter	Parameter order	Parameter Description
HBaseUtil hBaseUtil	1	Connect to the HBase tool class
String filepath	2	Export file path
String net_stations	3	The network corresponds to the station
String channels	4	Channel, can be any combination of 3 channels (BHE, BHN, BHZ)
String startTime	5	Start time of query
String endTime	6	End time of query
WaveFormType	7	Query waveform data type, enumeration class: A// seismic data, B// strong earthquake data, ...
int fileType	8	Export file type: 0-seed file, 1-miniseed file,...

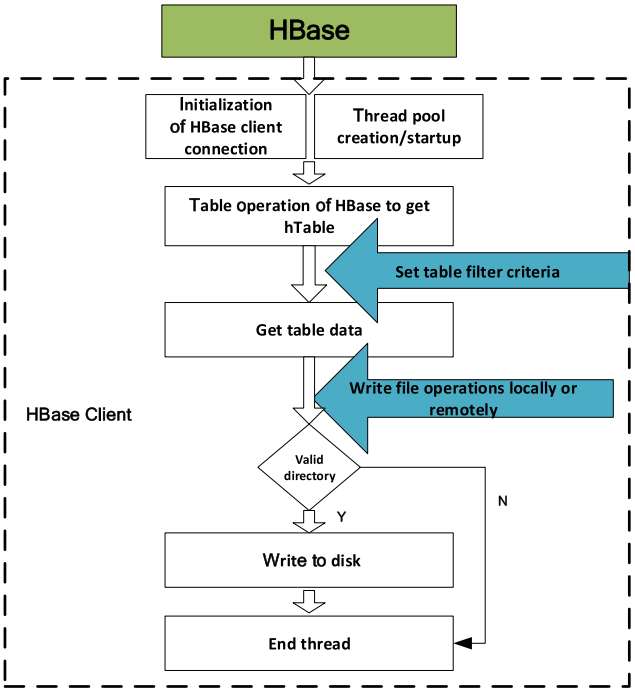


FIGURE 4. HBase data outbound process.

obtain HBase table operation by setting table filter conditions to obtain the table data. Finally, the file write operation is performed: according to the HBase client specific write disk requirement, the outbound data is written to local distributed storage medium or disk array, or the data write disk operation is performed through a remote mount storage path to realize the remote recovery of NEDBC seismic waveform data.

2) HBASE DATA OUTBOUND QUERY CONSISTENCY

The real-time flow of seismic waveform data is from the China Earthquake Network Center (CENC, BEIJING) to the HBase database of NEDBC, and the data is exported from HBase to the user. The data outbound query consistency test is mainly for comparing the miniSEED files, which are formed from the original real-time streaming data in Beijing, with the miniSEED file formed from the Xi'an HBase query and outbound. The contents must be compared to ensure data consistency.

First, prepare the miniSEED file from CENC as the original file. Second, the HBase client of the NEDBC sets query conditions of the same file as that of NEDBC, and writes the queried miniSEED file as a comparison file into disks. Third, use the method in Section III to parse the original file and the comparison file, and further parse them into a text format. Finally, perform text consistency validation statistics. It should be noted that, the duplicated data records in the real-time streaming data will be de-duplicated in NEDBC since the HBase database deletes and marks duplicate records.

TABLE 9. Configuration information of a sever node.

Component	Configuration information
CPU	10 Core Intel Xeon E5-2650v3 2.3GHz×2
Memory	256GB DDR4
Network	Bandwidth 10Gb/s
OS	CentOS 6.5
JVM Version	Java 1.6.0
Hadoop Version	Hadoop 2.6
HBase Version	HBase 1.0
Kafka Version	0.9.00
Spark Version	1.5.0
ZooKeeper Version	ZooKeeper 3.4.6

V. COMPARATIVE TEST AND DISCUSSION

A. TEST PLATFORM ENVIRONMENT AND DATA SET

This paper first compares the average running time of writing data into HBase based on Kafka model in different scale data sets and different cluster sizes, and compares the efficiency of writing data directly into HBase by a single server. Secondly, this paper tests the speed of reading the data from HBase to generate miniSEED files, and compares file consistency between miniSEED data and pre-parsed source files. The test system has been deployed on Sugon's open source HBase and HDFS, and the parameters have not been deeply tuned.

To verify the efficiency and practicability of the seismic waveform data storage and query system based on the model proposed in this paper, we have applied this system model to the NEDBC project constructing of seismic waveform data storage and query platform. The cluster test environment has a total of 12 sever nodes, and seismic waveform data parse program (using 24 processes). Kafka components and Spark Streaming (multi-thread concurrency) are deployed on the 12 compute nodes. The cluster node configuration parameters are shown in TABLE 9.

1) HBASE INBOUND PERFORMANCE TEST

The inbound test of HBase uses part of the miniSEED data file of NEDBC, which contains 1027 continuous observation stations with 100Hz sampling frequency of 33 networks in China mainland area, and each file record

TABLE 10. Major hardware configuration information of client B.

Component	Configuration information
CPU	10 Core Intel Xeon E7-4830v2 2.2GHz×4
Memory	256GB DDR4

length is 24 hours. The total amount of the test data includes 1,899,270 observation data files for seismic stations, about 65.2 TB. As mentioned above, we take t=10 minutes as the truncation time interval the to parse the miniSEED data files into the Key-Value form message records with Spark Streaming framework, and then write these message records to the HBase database concurrently through the Kafka message model. Finally, the HBase read/write performance test is performed. The average running time of processing of the miniSEED files are tested in a single node and different cluster sizes, and the speed-up ratios Speedup of data processing of different cluster sizes to a single node data are tested, as shown in EQ. (1):

$$\text{Speedup} = V_c/V_s \tag{1}$$

where V_c is the cluster concurrent data processing speed, and V_s is the single-node single-process data processing speed.

In order to verify the impact of the HBase client on the outbound performance, the query test without writing the disk is performed after the completion of HBase inbound task. The client with different configurations and performance is used to query the data from three channels of the network SN and the station HEYT with the waveform data that has been written into HBase. Query client Client A is a 2-CPU server, which is the same as compute nodes. Client B is a 4-CPU server. The hardware configuration is shown in TABLE 10.

2) OUTBOUND PERFORMANCE TEST

The HBase outbound performance test samples the inbound seismic data in three channels of all stations from January 1, 2017 to December 23, 2017. The total number of the data is up to about 10TB, and the number of threads is set to 20 to carry out the java class method in jar packages, which have the following format:

java -classpath (jar path) (the java class to be executed in the jar package) (Specify the parameters written by the query).

Specify the outbound parameters such as thread number of networks, stations, channels, clusters, and output file format and path. Use any client host A or B to call HBase data interface to perform the outbound and write disk operations, and write into the NEDBC HDFS with 80% storage utilization, NFS storage disk array with Raid5, and Beijing NFS storage disk array with Raid5 to conduct comparison test.

3) NEDBC SEISMIC WAVEFORM DATA CONSISTENCY TEST

The original data of the consistency test is based on the miniSEED sampled data provided by the CNEC: the miniSEED data of BHE channels of one station in one day (January 5, 2017) is randomly selected from 33 stations across the country as original data, which is a total of 33 files with a total size of 357MB. Then we read the corresponding miniSEED file from HBase of NEDBC for comparison. This process uses the class method carried out by HBase outbound test to match query conditions so as to get 33 files corresponding to the original data. The comparison method is shown in TABLE 11.

TABLE 11. Example of data consistency comparison method.

Network /Station	Source Data size of CENC /kb	Data size of NEDBC/ kb	Reason for inconsistent size of the record	Conclusion of the record consistency
JS/WX	10081	10080	There are two 512 bytes of redundant block from data source	Consistent after removing the CENC data source duplicate block: 10081–(2*512)/1024=10080
JX/SHC	12391	12391	null	Consistent
LN/YKO	9686	9661	There are fifty 512 bytes of redundant block from data source	Consistent after removing the CENC data source duplicate block:9686–(50*512)/1024=9661

TABLE 12. Average running time and query time-consuming of HBase inbound.

Data /TB	Average running time/s			Query time/s	
	Single node sever	12-node sever cluster	Speed-up ratio	Client A	Client B
1	373158	20971	17.79	3.99	3.38
2	741043	41692	17.77	7.56	7.27
3	1135642	64198	17.69	9.29	9.08
4	1497966	87381	17.14	12.44	12.13
5	1859177	104232	17.84	14.41	14.38
6	2263114	124583	18.16	16.50	16.32
7	2612111	143922	18.15	19.18	18.95
8	3017485	167437	18.02	20.73	20.56
9	3358428	189212	17.74	21.94	21.60
10	3718355	208879	17.80	22.82	22.39

B. TEST RESULTS AND DISCUSSION

1) HBASE STORAGE PERFORMANCE AND ANALYSIS OF SEISMIC DATA IN DISASTER RECOVERY CENTER

As shown in TABLE 12, in the cluster size of a single sever with single process and 12-node sever with multiple process, we perform data parsing into the NEDBC about 65TB seismic data. A query comparison of about 10 TB of data was performed, and the arithmetic mean of the 5 results was recorded. Finally, the average running time of each round of data processing was recorded and the speedup ratios were calculated. We then query the data from the three channels of network SN and station HEYT via different performance clients for the HBase distributed database composed of 12 nodes, also test the parameters of each group 5 times and use their arithmetic average to record time.

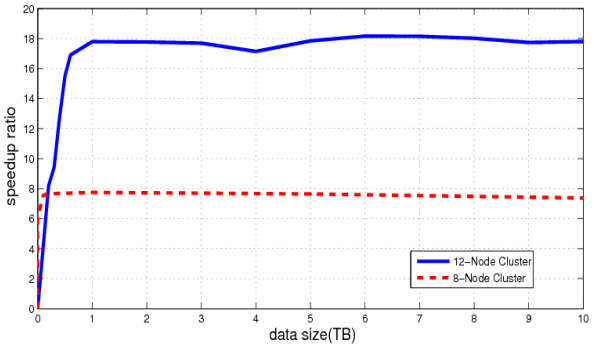


FIGURE 5. Speed-up ratio of different cluster sizes relative to a single server.

It can be seen from TABLE 12 above that the average running time of a single node is 373,158 seconds when processing 1TB of data, and the average running time of 12 compute nodes processing data concurrently is 20,971 seconds, and the speed-up ratio is 17.79. When the data set size increases, the speed-up ratio gradually rise: when processing 10TB of data, the average running time of a single node is 3,718,355 seconds, while the average running time of 12 compute nodes processing data concurrently is 208,879s, and the speed-up ratio is about 17.8. It can be seen from the test results that the average running time of HBase inbound process increases linearly with the increase of the data size. After the consumer information generated by Kafka enters Spark Streaming, the inbound process can be completed before the Kafka submits the next buffer. As the amount of data to be processed increases, the data processing performance in the cluster environment increases and the speed-up ratio gradually increases, eventually becoming stable. The query performance is mainly related to the interface performance provided by HBase itself. Thus, the query efficiency depends on the performance of HBase clusters, and has no direct relationship with the client of query. The small time-consuming fluctuations generated in each round of tests mainly come from the difference between the initializing reading configuration and the environment variables of Spark and HBase.

Comparisons of data inbound speed-up ratios under different cluster sizes are shown in FIGURE 5. It can be observed that since the number of parse program processes is related to the number of cluster nodes, the parallel computing power gradually manifests as the data set size increases, and the speed-up ratio gradually reaches the maximum capacity of concurrent processing. When processing 0.005 TB seismic data, the acceleration ratio of the 12-node cluster is relatively low, only about 5.2; at this time, the acceleration ratio of the 8-node cluster is about 7.1. The main reason for the difference is that the less the number of nodes, the shorter the time required to start the computing task and synchronize the storage system. When the data set size increases to 0.007 TB, the 8-node cluster processing speed-up ratio gradually approaches the value of stable state, which is about 7.6. And when the data set increases to about 0.8 TB, the 12-node cluster processing speed-up ratio reaches 17.7, which is close to the stable state of the system.

TABLE 13. Outbound consuming time of HBase under different storage strategies.

Data(TB)	Running time/s		
	Local distributed disk storage	Local disk arrary	Remote disk arry
0.5	978	1063	48545
1	1959	2135	81920
1.5	2962	3183	140434
2	3855	4341	161319
2.5	4993	5382	238312
3	5935	6432	257846
3.5	6872	7661	308407
4	7913	8665	322638
4.5	8738	9871	362968
5	9986	10765	476625
5.5	10881	11989	480597

2) PERFORMANCE AND ANALYSIS OF NEDBC HBASE OUTBOUND

The test results are shown in TABLE 13. It can be observed that the HBase client program takes about 10881 seconds to write the queried over 5TB seismic waveform data into NEDBC local HDFS with an average speed about 531MB/s, and takes about 11989 seconds to write such data into local disk array storage with an average speed about 488MB/s. It takes a total of 480597 seconds to write the same test data remotely to the storage disk array of CENC, Beijing, and the average speed is about 12MB/s. The difference in the speed at which data is written remotely to CENC storage and written to local storage is primarily due to network bottlenecks: since the bandwidth of the NEDBC to the CENC link is 12.5MB theoretically, the HBase of NEDBC transfers files to CENC at a speed of about 12MB/s, which can almost occupy the bandwidth of the industry network. It is the bottleneck of the HBase outbound tests in this work. Since the internet bandwidth of the NEDBC to the CENC is 12.5MB/s theoretically, the NEDBC HBase database transfers files to CENC at a speed of about 12MB/s, which can almost occupy the bandwidth of the industry network. It is the bottleneck of the HBase outbound tests in this work.

The speed difference between the local distributed storage and the disk array mainly comes from the hardware underlying storage mechanism: the disk array is based on the RAID5 method, and the distributed storage cluster is based on the $8 + 2$ erasure code strategy. It is obvious that the security mechanism of the latter one has a greater advantage for the data security of the disaster recovery center, although the average writing speed of the disk array and distributed storage is close.

3) CONSISTENCY ANALYSIS

The total size of the comparison source data in Beijing is 357M, and the size after export is 356M. The reason for the inconsistent data size is because the source data file from CNEC contains duplicate data blocks. When the

data is parsed into records and written to HBase, the duplicate records are only written once and marked, so there is no duplicate record in the exported file, which causes the file to become smaller. Comparing the data size before HBase inbound and after HBase outbound, since the duplicated records before HBase inbound are deleted, the data is consistent.

By constructing the data processing model based on Kafka cluster production-consumption information, the proposed platform in this paper has carried out the import/export of NEDBC seismic data, and verified that the data parse inbound/outbound model is an effective solution for saving system costs and promoting the speed of computing service.

VI. CONCLUSION

To improve the efficiency of writing seismic waveform data into HBase, this paper realized the Key-Value parse of waveform data, and proposed a data inbound process based on Kafka production-consumption model and Spark Streaming real-time computing framework for seismic waveform data design, achieved the purpose to process seismic waveform data concurrently by clusters. At the same time, the efficiency and consistency of seismic waveform data from HBase outbound were tested, and the effectiveness of the work was verified.

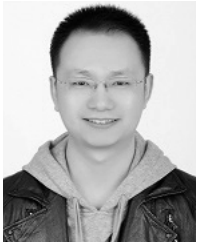
We will focus on research of the interception time interval of the seismic waveform data written into HBase and propose a corresponding optimization scheme in the next stage. In addition, data compression processing based on different algorithms will be attempted when the Kafka message queue receives information and Spark Streaming obtains information, thereby reducing system I/O operations while optimizing storage space utilization.

ACKNOWLEDGMENT

The authors would like to thank Prof. L. Ruifeng from IGCEA for his careful guidance and great help in the research of seismic waveform data; Mr. S. He and T.-Y. Yang from Shaanxi Yunji Huahai Company for providing automated test scripts for the Spark Streaming session; Prof. F.-J. Wang from CENC for providing valuable advices during the establishment of the NEDBC; and K. Guo, Ph.D., from CENC for providing original comparison data.

REFERENCES

[1] J. Su and W. Huang, "Automatic phases recognition technology in MSDP," *Seismolog. Geomagnetic Observ. Res.*, vol. 36, no. 5, pp. 1003–3246, 2015.
[2] Y. Wu and W. Huang, "Design and realization of real-time data stream interface routine between the JOPENS SSS service and the TDE-324 series earthquake data acquisition," *South China J. Seismol.*, vol. 31, no. 3, pp. 50–58, 2011.
[3] P. Zhengjun and Z. Lianfen, "Application and research of massive big data storage system based on HBase," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Big Data Anal. (ICCCBDA)*, Apr. 2018, vol. 18, no. 8, pp. 219–223.
[4] A. Siddiqa, A. Karim, and A. Gani, "Big data storage technologies: A survey," *Frontiers Inf. Technol. Electron. Eng.*, vol. 18, no. 8, pp. 1040–1070, 2017.
[5] 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," *ACM Trans. Comput. Syst.*, vol. 26, no. 2, pp. 1–26, Jun. 2008.
[6] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Record.*, vol. 39, no. 4, pp. 12–27, 2010.
[7] H. Liao, J. Han, and J. Fang, "Multi-dimensional index on Hadoop distributed file system," in *Proc. IEEE 5th Int. Conf. Netw., Archit., Storage*, Macau, China, Jul. 2010, pp. 240–249.
[8] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *ACM SIGOPS Operating Syst. Rev.*, vol. 44, no. 2, pp. 35–40, 2010.
[9] (2010). *HBase: Bigtable-Like Structured Storage for Hadoop HDFS*. [Online]. Available: http://hadoop.apache.org/hbase/
[10] F. Wang, W. Li, and G. Zhao, "Research on the architecture of seismic observation data platform," *Earthq. Res. China*, vol. 25, no. 2, pp. 214–222, 2009.
[11] S. Magana-Zook, J. M. Gaylord, D. R. Knapp, D. A. Dodge, and S. D. Ruppert, "Large-scale seismic waveform quality metric calculation using Hadoop," *Comput. Geosci.*, vol. 94, pp. 18–30, Sep. 2016.
[12] T. G. Addair, D. A. Dodge, W. R. Walter, and S. D. Ruppert, "Large-scale seismic signal analysis with Hadoop," *Comput. Geosci.*, vol. 66, pp. 145–154, May 2014.
[13] *Apache Kafka: A High-Throughput Distributed Messaging System*. Accessed: 2013. [Online]. Available: http://kafka.apache.org/
[14] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," in *Proc. 6th Conf. Symp. Operating Syst. Des. Implement.*, Berkeley, CA, USA, 2004, p. 10.
[15] J. Liu and S. Li, "Research of earthquake big data storage based on Hbase," *J. Geodesy Geodyn.*, vol. 35, no. 5, pp. 890–893, 2015.
[16] (2012). *SEED Reference Manual*. [Online]. Available: http://www.fdsn.org/
[17] J. Dean and S. Ghemawat, "MapReduce: A flexible data processing tool," *Commun. ACM*, vol. 53, no. 1, pp. 72–77, 2010.
[18] J. Chen, D. J. DeWitt, F. Tian, and Y. Wang, "NiagaraCQ: A scalable continuous query system for Internet databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, New York, NY, USA, 2000, pp. 379–390.
[19] (2019). *Apache Hadoop*. [Online]. Available: http://hadoop.apache.org
[20] T. O. Ahmed and M. Miquel, "Multidimensional structures dedicated to continuous spatiotemporal phenomena," in *Proc. Brit. Nat. Conf. Databases, Enterprise, Skills Innov. (BNCOD)*, 2005, pp. 29–40.
[21] A. Abelló, J. Ferrarons, and O. Romero, "Building cubes with MapReduce," in *Proc. ACM 14th Int. Workshop Data Warehousing (OLAP-DOLAP)*, Glasgow, U.K., 2011, pp. 17–24.
[22] G. Utard and A. Vernois, "Data durability in peer to peer storage systems," in *Proc. IEEE Int. Symp. Cluster Comput. Grid, (CCGrid)*, Chicago, IL, USA, Apr. 2004, pp. 90–97.
[23] A. Dandoush, S. Alouf, and P. Nain, "Simulation analysis of download and recovery processes in P2P storage systems," in *Proc. 21st Int. Teletraffic Congr.*, Paris, France, 2009, pp. 1–8.
[24] C. Blake and R. Rodrigues, "High availability, scalable storage, dynamic peer networks: Pick two," in *Proc. 9th Conf. Hot Topics Operating Syst.*, Berkeley, CA, USA, vol. 9, 2003, p. 1.
[25] D. A. Dodge and W. R. Walter, "Initial global seismic cross-correlation results: Implications for empirical signal detectors," *Bull. Seismolog. Soc. Amer.*, vol. 105, no. 1, pp. 240–256, Jan. 2015.
[26] L. Neumeyer, B. Robbins, A. Nair, and A. Kesari, "S4: Distributed stream computing platform," in *Proc. IEEE Int. Conf. Data Mining Workshops*, Sydney, NSW, Australia, Dec. 2010, pp. 170–177.



XU-CHAO CHAI received the B.S. degree in detection guidance and control technology from Northwestern Polytechnical University, Xi'an, China, in 2009, and the M.S. degree in signal and information processing from the Xi'an University of Posts and Telecommunications, Xi'an, in 2012.

From 2012 to 2014, he was a Development Engineer with ZTE Corporation, where he engaged in the development of Linux embedded operating systems. Since 2014, he has been a NEDBC Engineer with the Second Monitoring and Application Center, CEA. His research interests include large-scale seismic data receiving, transmission and storage methods, and has a certain working basis for the application and optimization of Hadoop platform and distributed databases.

QING-LIANG WANG received the B.S. degree in seismic exploration from Jilin University, Changchun, China, in 1984, the M.S. degree in engineering geology from Chang'an University, Xi'an, China, in 1990, and the Ph.D. degree in geodynamics from the Institute of Geophysics, China Earthquake Administration, Beijing, China, in 2003.

From 1984 to 1987, he was with the Department of Geophysical Exploration, Xi'an Institute of Geology, where he mainly engaged in the research of seismic exploration data signal processing and formation wave velocity extraction. From 1987 to 1990, he was with the Department of Hydraulic Engineering, Xi'an Institute of Geology, where he mainly engaged in research work on geological disasters such as ground fissures and landslides. Since 1990, he has been with the Second Monitoring and Application Center, CEA, where he has also been focusing on three-dimensional crustal deformation, geodynamics, earthquake precursor mechanism, volcanic deformation and geological disasters. His research interests include three-dimensional crustal deformation, geodynamics, earthquake precursor mechanism, volcanic deformation, and geological disaster.

WEN-SHENG CHEN received the B.S. degree from the Department of Geology, Peking University, Beijing, China, in 1991.

Since 1991, he has been a Senior Engineer with the Second Monitoring and Application Center, China Earthquake Administration (CEA). He is also the Director of the National Earthquake Data Backup Center, CEA. His research interests include earthquake prediction, earthquake informatization, and information network management planning of CEA.

WEN-QING WANG received the B.S. degree in electronic information science and technology and the M.S. degree in signal and information processing from Northwestern University, Xi'an, China, in 2008.

Since 2008, he has been a NEDBC Engineer with the Second Monitoring and Application Center, CEA. His research interests include earthquake informatization and the application of big data means to seismic data analysis.

DAN-NING WANG received the B.S. degree in computer science from the Xi'an University of Technology, in 2003, and the M.S. degree in solid geophysics from the Institute of Earthquake, China Earthquake Administration (CEA), in 2014.

From January 2017 to 2018, he studied as a Visiting Scholar at the University of California at San Diego, San Diego. Since 2016, he has been a Senior Engineer with the Second Monitoring and Application Center, CEA. His research interests are focused on information system operation and maintenance, cloud computing architecture, and big data. His research areas are on heterogeneous high-performance computing and seismic wave simulation.

YUE LI received the B.Eng. degree in detection homing and control technology and the M.Eng. degree in guidance, navigation and control from Northwestern Polytechnical University, Xi'an, China, in 2009 and 2012, respectively, and the Ph.D. degree from the Department of Mechanical and Biomedical Engineering, City University of Hong Kong, Hong Kong.

She is currently a Lecturer with the School of Astronautics, Northwestern Polytechnical University. Her current research interest includes networked dynamical systems and their applications.