Network music big data platform storage system construction

Tong Ouyang[1,a], Yi-zhen Cao[1,b]

[1]School of computer, Communication University Of China, Beijing, China

[a] oyoung_tony@cuc.edu.cn,[b]caoyizhen@cuc.edu.cn

Keywords: Big data platform, Online music, HBase, Distributed storage, Sqoop

Abstract: To build the network music big data platform, We need to collect a lot of music resources from the Internet, And store it on big data platforms. Therefore, it is a key problem to build a storage system that is high performance, scalable and capable of supporting big data. The network music data is very large and the metadata structure is not consistent, HBase and other distributed databases are widely used to build it. However, single RowKey index query in HBase is inefficient, it is difficult to meet the demand of big data platform for real-time or quasi-real-time query and statistical analysis. To this end, this paper proposes a solution, First, we migrated to HBase through Sqoop, which stored the music resources stored in multiple relational databases; Then use Phoenix to establish a secondary index query mechanism, build the Phoenix + HBase storage structure, and store management of the heterogeneous music resources; Finally, the read-write performance and query performance of the storage architecture are tested in the Spark parallel computing framework. Compared with the experimental results, the efficiency of Phoenix + HBase query performance is significantly higher than that of single computer and direct call to HBase, and it can operate unstructured data and structured data, which is in line with the requirement of the storage system of music big data platform.

1. Introduction

According to an investigation by IDC, an authoritative American firm, with the rapid development of the Internet, today's companies are producing data at an unprecedented rate and a very rich set of types. In 2000, the global storage of EB level data was expected to become ZB level by 2020, and 80 per cent of it was unstructured data. Following Google's 2006 paper on BigTable[1], In order to effectively deal with the storage and query management of massive data, more and more NoSQL distributed data storage system has replaced the traditional relational database management system (RDBMS), The storage capacity of NoSQL databases is also growing.

The same goes for music data. With the development of social informationization, music non-material cultural heritage resources digitization has become an inevitable trend and is changing[2], and increasing in time. The collection of music resources is continuous, the data volume is increasing, the data structure and variety are various, The query efficiency of existing MySQL relational database is gradually decreasing, which can not meet the requirement of big data efficient query[3]. There are various technologies in the field of big data, and it is not realistic for small and medium-sized development projects to carry out comprehensive concrete learning and research and development requirements in a short period of time[4]. Therefore, we have established high reliability, high performance, oriented and scalable distributed database HBase[5] as the main research object of big data platform storage system through extensive investigation. The network music big data platform storage system needs to provide read-write and analysis services such as Spark and Flume, Including the collected scattered collection of music source files and metadata resources together to store classification, and be able to support more efficient query conditions, extract the data, and to scale horizontally in the original data sheet. At the same time, the existing music metadata itself is stored in the existing multiple MySQL databases and needs to be transferred. This paper intends to use Sqoop to perform the music metadata migration of MySQL to HBase, It also constructs the Phoenix + HBase storage structure, and uses the familiar SQL language to make efficient and delete the basic operation of the unstructured database HBase, building a high performance and scalable network music data platform storage system.

2. Works

2.1. Phoenix + HBase storage architecture

Phoenix is a Java middle tier developed by Salesforce.com's James Taylor, which is open source, and provides a client-side embeddable JDBC driver that provides users with an additional search service[6]. Phoenix maintains SYSTEM TABLE in HBase and stores metadata about the Phoenix TABLE, The query engine converts the SQL query to one or more HBase Scan and executes to generate the standard JDBC result set. Unlike Hive on HBase, Phoenix uses the HBase API to implement the Query Plan, avoids the MapReduce framework, reduces Query time delay, and has minimal intrusion on HBase, without affecting the default operation of HBase.

Phoenix's deployment plan is to manually download the corresponding HBase version of the Phoenix package from the official website, upload it to each machine in the cluster for decompression installation, and configure the HBase and ZooKeeper configuration files to complete the deployment.

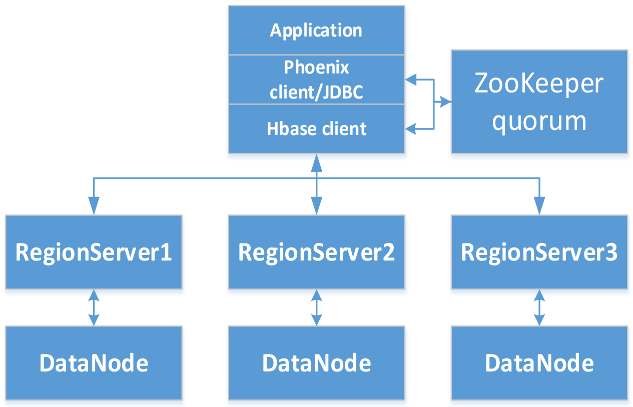


Figure 1 The Phoenix architecture diagram

In figure 1, the scheduling of cluster tasks is coordinated by the ZooKeeper distributed application, ZooKeepoer sends the mission to Phoenix, located at the top of HBase, via 2181 port. Phoenix parses and executes the task requirements and uses the JDBC connection to pass the processed tasks to HBase, which is then distributed to each of the regionservers and stored in the various datanodes of HDFS.

2.2. Phoenix + HBase correspondence

HBase creates a mapping relationship between the table in the original HBase through the RowKey, Column Family, Column structure, and the SQL statement to create the TEST table.

```
CREATE TABLE IF NOT EXISTS TEST(
PK PRIMARY KEY,
Family.Qualifier_1 BIGINT,
Family.Qualifier_2 BIGINT,
.......
);
```

Table 1 Corresponding relationship between Phoenix and HBase

Table / Function	Phoenix Table	HBase Table
Index	Primary Key	Row Key
Defines a collection of many columns of a category	Family	Family
Columns contained outside the primary key	Qualifier	Columns
Each unique primary key corresponds to the value of the column	Value	Value

If you do not name the columns under the cluster when the mapping is established, Phoenix will automatically update the column name by subscript "Family. _1", "Family. _2". Phoenix is mainly used to perform efficient query work services for HBase, Since the table is set up in Phoenix and deleted, the corresponding table in HBase will also be deleted accordingly, which has some inconvenience to the work. So we typically use Phoenix to build the VIEW relationship with the HBase table, and build the full map, so that we can make the most of the services provided by Phoenix. Table and column names and column names need to be enclosed in double quotation marks, Because HBase is case sensitive, If you don't use double quotation marks, Phoenix will automatically convert lowercase to capital letters when you create a table.

2.3. Phoenix secondary index

Phoenix's emergence is to provide more efficient data query based on HBase, In HBase, however, there is only one single RowKey index that is sorted by dictionary, When using RowKey for data queries, it is faster, However, if you do not use RowKey to query, you will use filter to scan the full table, greatly reducing the retrieval performance.In this regard, Phoenix provides a solution for secondary indexing technology.

The Phoenix secondary index allows the user to create an index through the DDL command CREATE INDEX, which creates a HBase table in the background that is different from the original table. At the time of the search, Phoenix carefully selects the most appropriate table based on the number of rowkeys formed.In the initial state, two types of indexes are supported:

1 Local indexing, The immutable data of HBase is loaded by HFile. It is applicable to situations where the operation is frequent, Phoenix will automatically determine whether when queried using the index, and index data and the data in the table is stored in the same server, avoided at the time of write operations to different server index table of the write index extra overhead.

2 Global Indexing, Support for changing data, under this index, when the data table changes, Phoenix will add a WAL record that contains enough information to ensure that the index can be generated in any error condition. This is true for a business scenario where you can read and write less, because all of the updates to the data table (DELETE, UPSERT VALUES and UPSERT SELECT) will cause an update to the index table, and the index table is distributed in different data nodes, and data transmission across nodes leads to greater performance consumption. When reading the data, Phoenix selects the index table based on the index condition to reduce the time of query consumption.

3. Project application and analysis

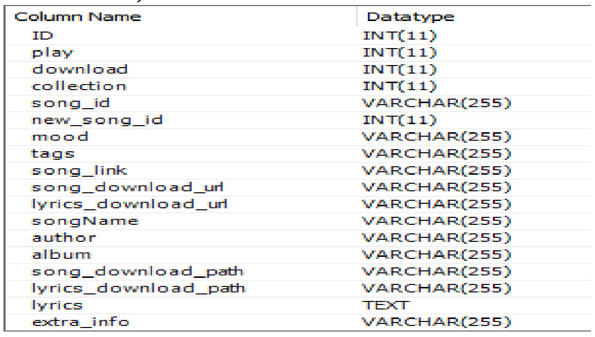
3.1. System environment

The network music big data platform storage system needs to store a lot of unstructured data, low density and large capacity, at the same time, it needs to provide data services for multiple departments and systems, and the data processing and analysis of internal data is large and read and write frequently. Therefore, a distributed storage architecture based on cheap PC server + large capacity SATA hard drive is adopted, this can not only improve system storage performance, but also reduce storage cost. We chose six virtual hosts to build a Hadoop cluster. The hardware condition is 2 identical dell servers. The configuration of each table is: 4 nuclear Inter (R) Xeon (R) E5-4607 V2 CPU, and the main frequency is 2.60 GHz; Memory is 32GB; 20 TB hard disk.

Software environment: Centos7 operating system 64 bit; The large data platform of Hadoop2.7.2; ZooKeeper3.4.9 distributed application coordination services; HBase1.1.9 distributed storage system; Sqoop1.4.6 data transmission tool; MySQL5.6.0 relational database; Phoenix4.8.2 OLTP storage tools; Squirrel3.7.1 - SQL database management tool.

3.2. The experimental data

The data of the experiment is 100W of audio metadata stored in existing MySQL. Each data represents a piece of audio information, and the data format of one table is shown in the figure:



Column Name	Datatype
ID	INT(11)
play	INT(11)
download	INT(11)
collection	INT(11)
song_id	VARCHAR(255)
new_song_id	INT(11)
mood	VARCHAR(255)
tags	VARCHAR(255)
song_link	VARCHAR(255)
song_download_url	VARCHAR(255)
lyrics_download_url	VARCHAR(255)
songName	VARCHAR(255)
author	VARCHAR(255)
album	VARCHAR(255)
song_download_path	VARCHAR(255)
lyrics_download_path	VARCHAR(255)
lyrics	TEXT
extra_info	VARCHAR(255)

Figure 2 The audio metadata format of one of the tables

Each of the audio's related information is stored in separate fields, Since music resources are climbing down from different sources, the format of each table is not uniform, the length varies, and there are no strict rules.

3.3. Sqoop data migration

Sqoop is an open source tool that is developed to exchange data between Hadoop and relational databases [7]. In order to avoid hot issues, the load imbalance caused by the operation of HBase will be designed as the form of the hash value of "reverse timestamp _ song name". After generating RowKey through the Python script, we add a RowKey field to the songs table in MySQL's music library, and create an empty table music in HBase to prepare a cluster song for data migration.

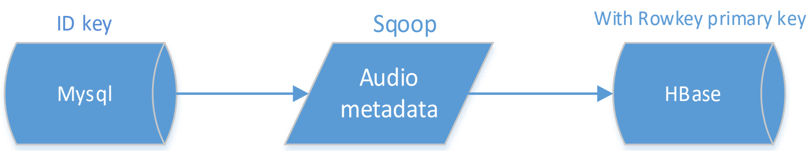
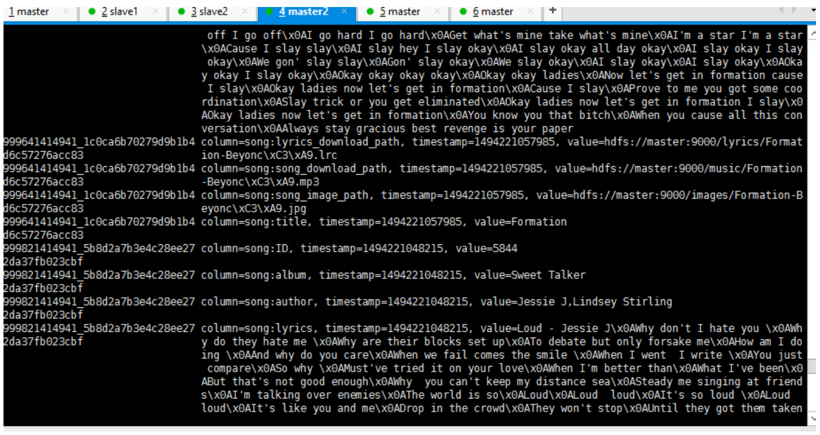


Figure 3 Sqoop import process

The Scan music table is shown as follows:



```
                             off I go off\x0AI go hard I go hard\x0AGet what's mine take what's mine\x0AI'm a star I'm a star
                            \x0ACause I slay slay\x0AI slay hey I slay okay\x0AI slay okay all day okay\x0AI slay okay I slay
                             okay\x0AWe gon' slay slay\x0AGon' slay okay\x0AWe slay okay\x0AI slay okay\x0AI slay okay\x0AOka
                            y okay I slay okay\x0AOkay okay okay okay\x0AOkay okay ladies\x0ANow let's get in formation cause
                             I slay\x0AOkay ladies now let's get in formation\x0ACause I slay\x0AProve to me you got some coo
                            rdination\x0ASlay trick or you get eliminated\x0AOkay ladies now let's get in formation I slay\x0
                            AOkay ladies now let's get in formation\x0AYou know you that bitch\x0AWhen you cause all this con
                            versation\x0AAlways stay gracious best revenge is your paper
999641414941_1c0ca6b70279d9b1b4 column=song:lyrics_download_path, timestamp=1494221057985, value=hdfs://master:9000/lyrics/Format
d6c57276acc83                   ion-Beyonc\xC3\xA9.lrc
999641414941_1c0ca6b70279d9b1b4 column=song:song_download_path, timestamp=1494221057985, value=hdfs://master:9000/music/Formation
d6c57276acc83                   -Beyonc\xC3\xA9.mp3
999641414941_1c0ca6b70279d9b1b4 column=song:song_image_path, timestamp=1494221057985, value=hdfs://master:9000/images/Formation-B
d6c57276acc83                   eyonc\xC3\xA9.jpg
999641414941_1c0ca6b70279d9b1b4 column=song:title, timestamp=1494221057985, value=Formation
d6c57276acc83
999821414941_5b8d2a7b3e4c28ee27 column=song:ID, timestamp=1494221048215, value=5844
2da37fb023cbf
999821414941_5b8d2a7b3e4c28ee27 column=song:album, timestamp=1494221048215, value=Sweet Talker
2da37fb023cbf
999821414941_5b8d2a7b3e4c28ee27 column=song:author, timestamp=1494221048215, value=Jessie J,Lindsey Stirling
2da37fb023cbf
999821414941_5b8d2a7b3e4c28ee27 column=song:lyrics, timestamp=1494221048215, value=Loud - Jessie J\x0AWhy don't I hate you \x0AWh
2da37fb023cbf                   y do they hate me \x0AWhy are their blocks set up\x0ATo debate but only forsake me\x0AHow am I do
                                ing \x0AAnd why do you care\x0AWhen we fail comes the smile \x0AWhen I went  I write \x0AYou just
                                 compare\x0ASo why \x0AMust've tried it on your love\x0AWhen I'm better than\x0AWhat I've been\x0
                                ABut that's not good enough\x0AWhy  you can't keep my distance sea\x0ASteady me singing at friend
                                s\x0AI'm talking over enemies\x0AThe world is so\x0ALoud\x0ALoud  loud\x0AIt's so loud \x0ALoud
                                loud\x0AIt's like you and me\x0ADrop in the crowd\x0AThey won't stop\x0AUntil they got them taken
```

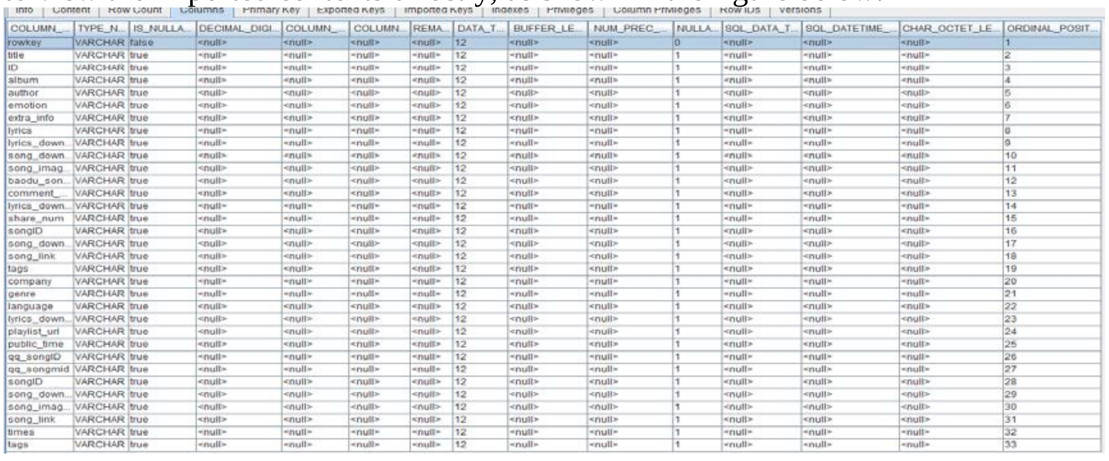
Figure 4 Table structure after HBase scan

Sqoop is transmitted through mapreduce tasks, The RowKey field that has been added to the MySQL table songs in the command is transferred as the RowKey primary key in HBase table music. After the transmission is completed, the original primary key ID is converted to the family in the music table, and the entire column can be deleted directly. When the Sqoop statement that imports data from MySQL to HBase is performed again, the corresponding rows in HBase are updated and replaced based on the same RowKey value. And save different versions of the data, which can limit the quality of the data saved by setting the expiration date and number of versions.

We kept the original crawler, Spark architecture and MySQL architecture unchanged, by using synchronous data to the relevant HBase method, the original system and the existing system can be decoupled without intrusion. In this way, we can selectively use the original relational database based on the business requirements, or even use the HBase database in part or even fully use the HBase database.In the process of project operation, the upper framework can be selected according to the strategy of the business, whether it is the original relational database operation, or the HBase database or both.

3.4. HBase + Phoenix build

After the success of the Phoenix architecture deployment, use the command line to start the HBase cluster on Phoenix and visualize the Phoenix using the Squirrel tool.First, create a view in Phoneix to establish a mapping relationship with the table in HBase. This is where we can query work and create global indexes for fields that need to be queried. After the success, select view in Squirrel to view the imported contents directly, as shown in the figure below:



COLUMN_...	TYPE_N...	IS_NULLA...	DECIMAL_DIGI...	COLUMN_...	COLUMN...	REMA...	DATA_T...	BUFFER_LE...	NUM_PREC_...	NULLA...	SQL_DATA_T...	SQL_DATETIME_...	CHAR_OCTET_LE...	ORDINAL_POSIT...
rowkey	VARCHAR	false	<null>	<null>	<null>	<null>	12	<null>	<null>	0	<null>	<null>	<null>	1
title	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	2
ID	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	3
album	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	4
author	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	5
emotion	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	6
extra_info	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	7
lyrics	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	8
lyrics_down...	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	9
song_down...	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	10
song_imag...	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	11
baodu_son...	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	12
comment_...	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	13
lyrics_down...	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	14
share_num	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	15
songID	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	16
song_down...	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	17
song_link	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	18
tags	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	19
company	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	20
genre	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	21
language	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	22
lyrics_down...	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	23
playlist_url	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	24
public_time	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	25
qq_songID	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	26
qq_songmid	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	27
songID	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	28
song_down...	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	29
song_imag...	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	30
song_link	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	31
times	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	32
tags	VARCHAR	true	<null>	<null>	<null>	<null>	12	<null>	<null>	1	<null>	<null>	<null>	33

Figure 5 Squirrel manages Phoenix

This building to complete the entire Phoenix, originally a different table now stored in HBase different columns in the table cluster, in the same RowKey format stored together, some of the same column names, but the column cluster name is different, can distinguish them very well. Such tables are generally very sparse, because each music has only its own fields, but since HBase is a column -

oriented storage, there is no waste of storage space.

3.5. General query comparison experiment

In order to verify the feasibility of big data storage system, we conducted a comparison experiment of general query. Experiments using the Python programming for MySQL (no index), MySQL, native HBase, Phoenix + HBase (without secondary indexes), Phoenix + HBase (secondary indexes) the five general query, by comparing their respective by the length of time taken for analysis. The Hadoop platform writes [8] completely in native JAVA language and can deploy the Thrift interface to implement Python operation HBase. Under the four conditions, the general query is carried out in the case of 20w-100w data, and the response time is output, which can be used to measure the efficiency of the query. The test results are as shown in the figure:

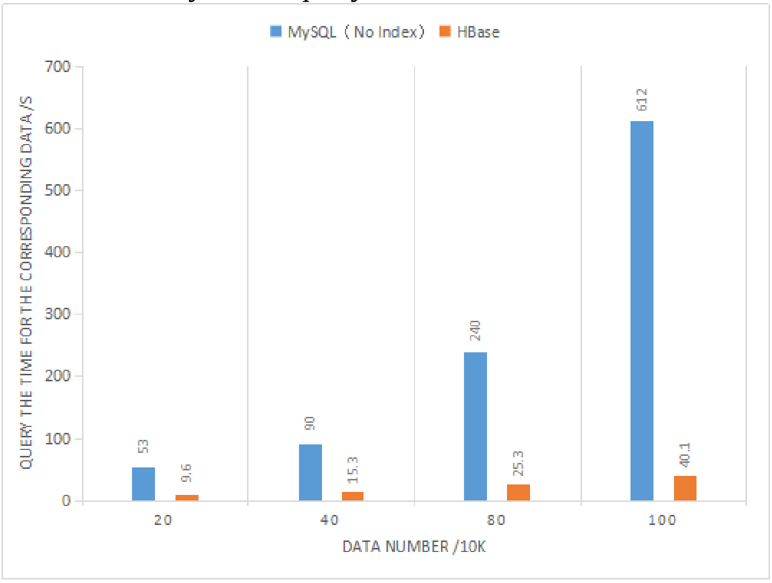


Figure 6 MySQL has no index and HBase comparison

As shown in figure 6, the query efficiency of MySQL is greatly reduced with the increase of data volume without index creation. The reason is that MySQL's query is performed on a single host, and when the HBase of the distributed cluster of multiple hosts is used to query, the processing efficiency is significantly increased due to the increase of hardware conditions.

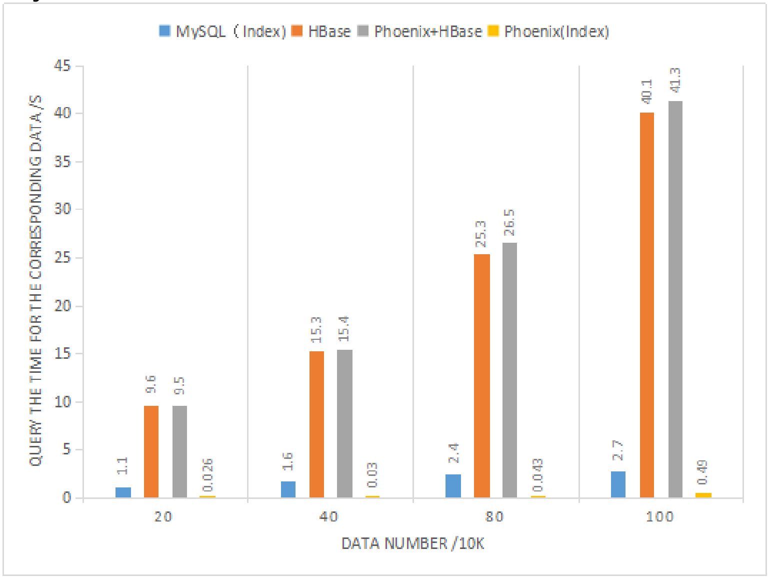


Figure 7 General query comparison

As shown in figure 7, for the MySQL database, after the index is established, when the data volume increases, the query speed is gradually slowing down. What is predictable is that when the amount of data is large enough, MySQL's query efficiency will be greatly reduced; When a direct call to the Thrift operation HBase is performed to query the operation, it is equivalent to the direct query of the Phoenix at the upper level using SQL statements. Can be understood as Phoenix is also use HBase inherent query mechanism and method, but the mechanism of the original packaging and transformed into new widely familiar SQL statements can be used to replace the original command line.Therefore, a full table scan was performed without index, which resulted in the search results, slow and inefficient; After the second index of Phienix, the query efficiency is significantly improved. The index uses a space-changing idea, such as setting the value of a field to the RowKey to achieve extremely efficient lookups.

3.6. Spark batch writes

Batch write experiment based on Apache Spark[9] parallel computing framework, use sc. TextFile reading the data on the HDFS, and transformed into RDD data sets, the RDD transformation and action operation make the Spark to operations, including the map () and filter (), collect (), etc. Spark operates on MySQL and Phoenix + HBase using standard SQL language. Read music metadata table file in HDFS, from the RDD data format conversion for DataFrame data format in MySQL and Phoenix + HBase respectively, after data in the same article number and contents of the case through the task execution duration to measure the efficiency of batch write. The test results are as follows:

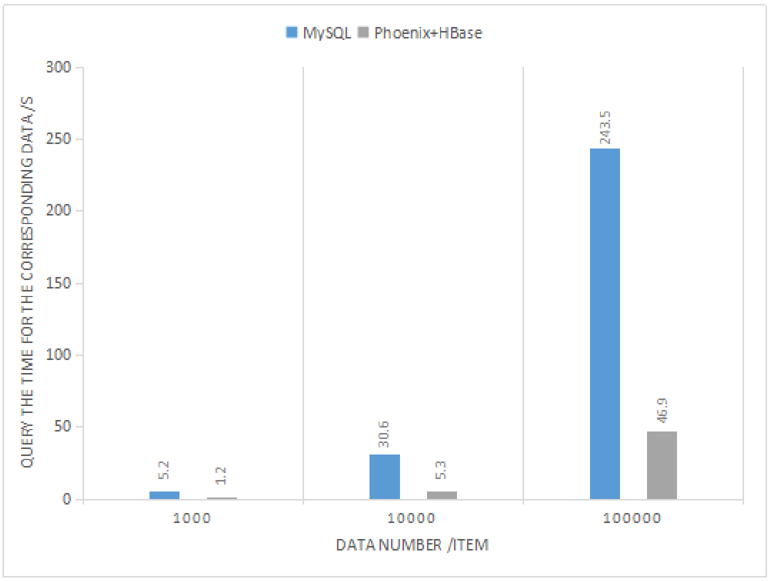


Figure 8 Spark batch write contrast

The figure 8 shows than on a single server MySQL and two servers by the virtual machine in the building of the Phoenix + HBase cluster architecture was associated with significantly lower processing speed, is largely due to the distributed under the conditions of advantage of the hardware and the inner mechanism of optimization, temporarily to satisfy the requirements for big data platform in performance improvement.

To sum up, the HBase distributed database brings together all the node hardware advantage, to make the bigger of the owned resources, embody the advantages of distributed, improve processing efficiency and capacity; By establishing the Phoenix + HBase structure, the data is stored and managed, and the data is visually visible, and the migration is simple and fast.After Phoenix set up secondary indexes, query efficiency greatly, speaking, reading and writing for big data platform services such as storage and literacy laid a foundation, for music of heterogeneous data resources for the demand of storage management, complete the network construction of music big data storage system.

4. Conclusion

This paper has successfully established the Phoenix in network music big data platform + HBase framework, using Sqoop data migration, Squirrel for visual management, and several cases of database query efficiency comparison experiment. The experimental results show that Sqoop migration is simple and fast. Phoenix + HBase is applicable to the management of heterogeneous music data resources, and the query efficiency is significantly improved compared to MySQL and native HBase in the case of establishing secondary indexes. In the experiment of Spark batch writing, Phoenix + HBase is also more efficient, which can meet the demand of storage management for big data platforms. The following will continue to optimize and improve the architecture, improve literacy, and develop the best performance of the big data platform as much as possible using multi-thread concurrent processing.

Acknowledgements

National science and technology support plan project in China, item no.2015BAK22B01.

References

[1] Chang F. et al. Bigtable:Adistributedstoragesystemfor structured data[C]// ACM Transactions on Computer Systems. 2006:205--218.

[2] Teng Teng. "digital music resources" and "music resources of digitization" -- reflections on the construction of music non-material cultural heritage database construction work [J]. Music communication, 2015, (3) : 50-54.

[3] Jeffrey Dean,Sanjay Ghemawat.MapReduce:Simplified Data Processing on Large Clusters.Google,Inc,2004.

[4] Xiaoliang Ma,Feng Tian. Hadoop framework native Hbase, Hive, Lealone, Phoenix each operating components such as comparison [J]. Journal of Guangdong communication technology,2017,37(3):71-74.DOI:10.3969/j.issn.1006-6403.2017.03.018.

[5] Lars George."Hbase:The Definitive Guide".1st edition,O'Reilly Media,2008.

[6] Zhaozhao Shi,Xin Wang,Jin Xu,et al. Phoenix technology tracking and application on HBASE [J]. Business, 2016, (47) : 135-136.

[7] Jinliang Yu,Zhixiang Zhu,Xiaojiang Liang et al. A data exchange system based on Sqoop [J]. Internet of things technology,2016,6(3):35-37.DOI:10.16667/j.issn.2095-1302.2016.03.010.

[8] Jun Lei,Hangjun Ye,Zesheng Wu, Peng Zhang,Long Xie,Yanxiang He.Big-Data Platform Based on Open Source Ecosystem[J].Journal of Computer Research and Development,2017,54(1):80-93.

[9] Zhongyi Sun,Fengke Chen,Mingmin Chi et al.A Spark-Based Big Data Platform for Massive Remote Sensing Data Processing[C].//Data science: Second International Conference, ICDS 2015, Sydney, Australia, August 8 – 9, 2015, Proceedings.2015:120-126.