ISSN: 2664-9640

Big Data Query Optimization Based On MongoDB

Jian Hu[1], Jin Hou[2]

[1] School of Automation & Information Engineering, Sichuan University of Science & Engineering, Yibin 644000, China;

[2] Artificial Intelligence Key laboratory of Sichuan Province, Yibin 644000, China.

Abstract

Nowadays, with the rapid increase of data volume, MongoDB database is inefficient when using skip limit paging query, which also leads to the reduction of database access performance. Now, the key factors affecting database paging query are analyzed. It is found that the paging display efficiency of skip data is too low when processing a large amount of data, this article mainly improves the built-in paging query technology of MongoDB database, and adds the "conditional query + sorting + time limit return record" method to the original skip-limit technology to increase the rate of paging query, Through theoretical analysis and experimental data comparison, the improved skip limit method and adding index have greatly improved the query rate of a large amount of data.

Keywords

Big data, MongoDB, NoSQL, query optimization, index.

1. Introduction

In the context of the era of big data, data is growing explosively. Semi-structured data and unstructured data are growing rapidly. In the face of massive amounts of stored data, traditional databases can no longer meet the current needs [1]. NoSQL database has existed for many years, but with the rise of big data and cloud computing, it has been applied on a large scale only. Compared with traditional relational databases, it has unique advantages in performance and ease of use, It can easily handle large amounts of data and high user load [2]. As one of the representatives of NoSQL, MongoDB shows its own unique advantages under many advantageous conditions. The skip-limit method of paging data query shows a normal query rate when the amount of data and concurrent users are small, Larger will cause slow operation. and the paging query of MongoDB needs to be improved and optimized. Creating an index can also make us query the database faster and more efficient. MongoDB's index has many things in common with traditional relational databases. The creation of multiple indexes reduces response time and allows faster query processing.

2. Introdution to Mongodb

MongoDB is a database system based on distributed file storage, written in C++ language, with the purpose of providing a scalable and high-performance data storage solution for WEB applications [3].

MongoDB is a high-performance, open source, schemaless document database, and it is also one of the more popular NoSQL databases. MongoDB is a product between relational and non-relational databases. It is the most versatile and closest to relational database among non-relational databases. It supports a very loose data structure, which is similar to json's bson format, so that it can store more complex data types [4].

ISSN: 2664-9640

2.1. Features of MongoDB Database

The MongoDB database has many characteristics such as high performance, easy deployment, and convenient storage. The main functional characteristics [5]:

（1）Mode freedom. MongoDB is different from traditional relational databases that require pre-defined data tables. Each piece of data in it has its own attributes and structure.

(2)Collection oriented storage, easy to store object type data. MongoDB uses the bson format for exchange and storage. This storage format similar to json is particularly suitable for storing object data types.

(3)Full indexing is supported, including internal objects. MongoDB supports multiple index modes of relational databases (ascending, descending, unique, composite, etc.), and also supports secondary indexes (implemented through B-tree), and each collection supports 64 indexes.

(4)Support dynamic query. Storage is dynamic. Compared with traditional databases, MongoDB's document-oriented form allows its attribute values to be added and deleted arbitrarily.

(5)Automatically process fragments .This is to support scalability at the cloud computing level. MongoDB can automatically segment data in the cluster, allowing the cluster to store more data under the same conditions and achieve greater load capacity.

(6)Support replication and fault recovery. The log function can help the system recover complete data when the system is down. By default, MongoDB enables this function, so that the data server can quickly recover unfinished operations through log files.

(7)MongoDB supports Golang, RUBY, Python, Java, C++, PHP, C# and other languages. Developers can use any mainstream language to start programming.

3. Principles of Mongodb's Two Query Methods

3.1. MongoDB Paging Query Technology

With the rapid development of the Internet, it has become more and more inseparable from our lives, and the Web serves as a window to users. When all the information is displayed, it will appear lengthy and affect the user's sense of experience. Paging query is to page the content to be browsed reasonably, which is convenient for users to query and view, and divide large pieces of content to achieve the dual effect of beautiful page and good user experience [6].

Generally speaking, paging query technology can be used to increase the query rate when querying and extracting very large-capacity data. First, the sentence needs to be filtered; secondly, After selecting n pieces of data to query, query other data in the same way [7]. Displaying large amounts of data directly will bring difficulties to the browser and affect the system. At the same time, pushing large amounts of data to the client will also increase the burden on network bandwidth. When faced with this situation, paging query on the data will be the best choice.

Skip-limit paging query technology will use the find function to return the set of results according to the query conditions; Skip the specified data with skip function to find the location where the user requires to display the first record; finally limit is used to limit the number of pages displaying records . For example: you need to query the information of all students who are older than 14 in the students collection of the MongoDB database. Assuming that 20 pieces of information are displayed on each page, the query formula should have a specific format.

Page 1

```
db.students.find({age:{$gt:14}}).skip(0).limit(20);
```

Page 2

ISSN: 2664-9640

db.students.find({age:{$gt:14}}).skip(20).limit(20);

……

Page n

db.students.find({age:{$gt:14}}).skip(（n-1）*20).limit(20).

The general format is: database name.collection name.find (query condition). ((Page number - 1)*page number record number).limit (page number record number), through skip-limit this method to achieve paging technology, improve the query rate [8].

3.2. MongoDB Index

MongoDB database and traditional relational database can also add indexes to increase the query rate of the database. The same is that MongoDB also supports multiple types of indexes, including single-field indexes, composite indexes, multi-key indexes, text indexes, etc. Each index has its own use cases [9].

When you insert multiple documents into the collection, a location information will appear after each document passes through the underlying storage engine. We can use this information to read the document from the engine. For example, in the mmapv1 engine, the location information is "file + offset in the file"; in the wiredtiger storage engine, the location information is a key generated by the stored document, and the document is accessed through the key [10]. Suppose there is a query "db.students.find({age:14})" to query all students whose age is 14. At this time, you need to traverse all documents. When the amount of data in the table is not large, the cost of document scanning is not large; but when the amount of data reaches millions or tens of millions, such scanning may take several seconds or even minutes. At this time, we can create an index, such as changing to "db.students.createIndex ({age:1})". After establishing the ascending index of the age field, MongoDB will store a copy of the ascending data of the age field and persist it in a structure similar to B-tree, so as to ensure that we can quickly find the location information corresponding to age = 14 to obtain the corresponding document.

4. Mongodb Two Query Optimization Methods

4.1. Improved Skip-limit Method

The mongodb database has built-in skip limit paging method, which can cope with small data , and the corresponding query rate has also increased; but when the amount of data reaches tens of millions, this method appears to be inadequate, the loading speed of the corresponding page will become particularly slow , and the performance of the system will also be affected [11]. At this time, we can improve this method by using the method of "conditional query + sorting + time-returned records" to increase the query rate. We can sort while querying. After sorting, we can extract the last record of the first page as the condition of the second page , and so on... The flow chart of the improved query is shown as Figure 1.

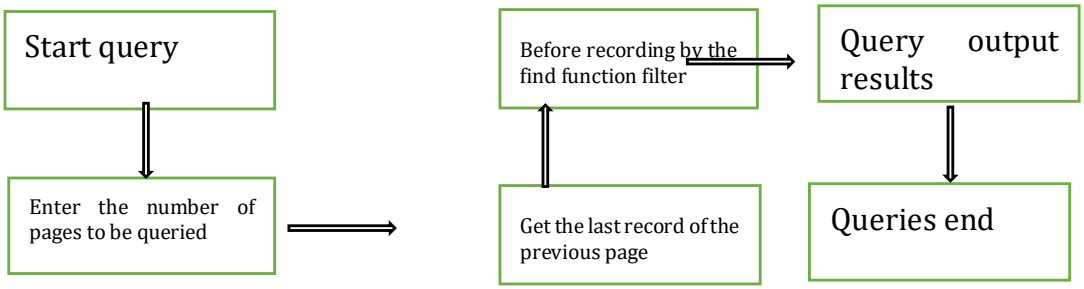

Figure 1: Improved skip-limit flow chart

ISSN: 2664-9640

Facing the huge amount of data, if we want to improve the query rate, we can verify the feasibility of this method from experiments. First, you need to prepare the data. Import 1,000,000 data in MongoDB into the database text import collective user. The structure in the set includes num, name, and age[12].

>for(i=0;i<1000000;i++){db.user.insert("num":i,{"name":"xiaoming"+i,"age":(i+10)%100});}

Paging query 10 data with the 800000th data in ascending order of num. explain("executionStats") is a built-in analysis tool to help us intuitively see the running time, whether to add an index, and so on.

First use the skip-limit method (the experimental results are shown as Figure. 2):

>db.user.find().sort({"num":1}).skip((80000-1)*10).limit(10).explain("executionStats");

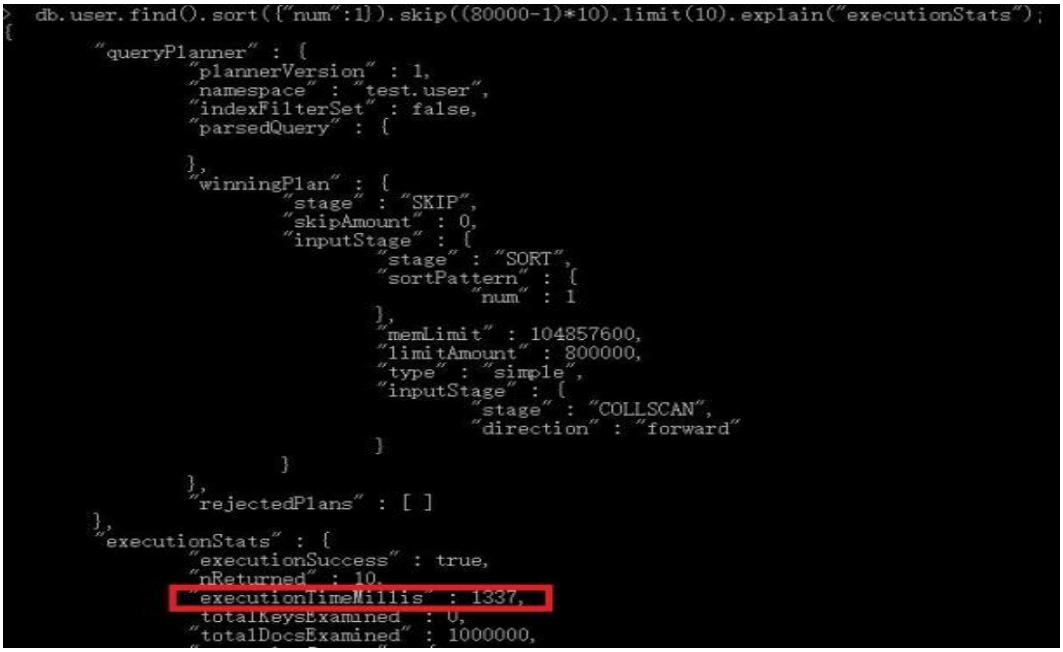

Figure 2: skip-limit paging query for 800000 data

Then use the improved query method (experimental results are shown as Figure. 3):

>db.user.find({num:{"$gt":800000}}).sort({"num":1}).limit(10).explain("executionStats");

Figure 3: Improved skip-limit paging query for 800000 data

ISSN: 2664-9640

4.2. Create An Index to Optimize The Query

The improved paging query obviously shows that the speed is increasing, but it will consume a lot of memory. Considering those factors, we can also use the MongoDB database to create indexes, which is also an efficient query method. MongoDB has two commands to create an index, one is db. collection.ensureInder ({attribute: 1}) 1 means ascending order, -1 means descending order, and the other is db. collection.creatInder ({attribute: 1}). These two command methods are equivalent.

Here we use the same method as the improved skip-limit verification to verify the feasibility of the experimental scheme . First insert 100,000 data into the database, and the time taken by explain() to display the 10,000th data [14].

>for(i=0;i<100000;i++){db.user.insert({name“zhangsan”+i,age:i})}

Before creating the index (the experimental results are shown as Figure. 4):

>db.user.find({name:“zhangsan10000”})

>db.user.find({name:“zhangsan10000”}).explain(“executionStats”)

Figure 4: Before adding index

After the index is created (the experimental results are shown as Figure. 5):

>db.user.ensureIndex({“name”:1})

>db.user.find({name:“zhangsan10000”}).explain(“executionStats”)

ISSN: 2664-9640

Figure 5: After adding index

4.3. Comparative Results are Summarized

It can be seen from the data in the figure that when querying the same 800,000 data, the skip-limit method takes 1337ms, while the improved method takes 841ms, which significantly improves the query rate before a large amount of data; when querying the same 10,000 data amount, it takes 62ms to create the index without index, but 16ms to create the index. It is obvious that the query rate is accelerated after creation. The experimental results show that the two methods are effective.

5. Experimental Verification Analysis

In order to further eliminate the error caused by accidental factors, make the data more convincing, and more accurately show the feasibility of the experimental method, I have tested each method for many times, adopted the average method, and obtained the following experimental results.

5.1. Improved Paging Query

For the experiment, 10,000, 100,000 and 500,000 test tests are carried out. Using the skip-limit method to get the query time is 540ms, 1027ms, 1337ms, and the improved skip-limit method to query the data took 531ms, 686ms, and 841ms respectively. The result shows that there is little difference when the amount of data is known. When the amount is large, the speed increases significantly. Take the average of the results of multiple experiments to get a line chart as shown in Figure. 6.

ISSN: 2664-9640

Figure 6: Comparison chart of paging query rate

5.2. Create Index Optimization

For the experiment, the data volume of 5000, 10000 and 50000 is tested. It took 56ms, 62ms, and 98ms when there was no index, and it took 0ms, 16ms, and 54ms after adding an index. The comparison found that after adding, the query rate has increased slightly, which proves that the query rate can indeed be improved. A small improvement. Take the average of the results of multiple experiments to obtain a line graph as shown in Figure. 7.

Figure 7: Adding index query rate comparison chart

6. Conclusion

Through experiments, we can see that MongoDB can cope with skip-limit when the amount of data is small, but when the amount of data is too large, using the improved skip-limit index can better improve the paging query rate. After adding indexes, we can also increase our query rate appropriately, but the disadvantage is for tables that are frequently modified, table fields with

ISSN: 2664-9640

repeated data and evenly distributed, and table fields that are frequently queried with the main field but with more index values for the main field, the use of an index structure is likely to reduce the rate due to the addition of an index. Improved paging query and indexing are both good processing methods for big data.

From the experimental results, although the condition and limit are used for accurate positioning, there is no record traversal in the results, and the efficiency is greatly improved, the key value array has high requirements for memory capacity when the record is large, so the split array can be used for optimization. The limitation is that various methods must establish indexes for paging under big data, so it will have a certain performance impact on applications with many records added and modified.

References

[1] X.F. Cheng. *The authoritative guide to MongoDB*. (People's Posts and Telecommunications Press, Beijing,2011).

[2] Z.J. Chen. Research and Application of NoSQL Database . Computer Programming Technology and Maintenance, vol. 9 (2020), p. 81-83.

[3] Hui-min Ren, Xu-hui Yang, Xian-hong Liu, Fei Liu. Research on the application of MongoDB database in the concrete industry . Technology and Innovation, vol.20 (2018), p. 38-39.

[4] P. Wu, H.W. Liu, H.J. Ding. Research on knowledge representation and storage of mechanical product scheme design based on ontology and NoSQL. Journal of Information, vol. 3 (2017), p.285-296.

[5] J.J. Sun, L.Y. Zeng, Y. Zhang. Research on Web Spatial Data Storage and Management Based on MongoDB. Surveying and Mapping, vol.2.(2017), p.72-74.

[6] M.R. Tian. Paging query based on PHP + MySQL in the Web. Scientific Consulting, vol.10 (2017), p.40-41.

[7] X.P. Su. Research on query optimization based on massive data in Oracle database. Modern Industrial Economy and Modernization, vol.3 (2017), p.84-85.

[8] Z.H. Wang, Z.D. Wang. Data paging optimization technology in MongoDB. Computer System Applications, vol.6 (2015), p.243-246.

[9] Z.F. Yan, W.H. Zhang. Research progress of index structure for big data. Big Data, vol.4 (2019), p.3-15.

[10] P. Zong, X.J. Wu. Research on composite index technology based on NoSQL system . Computer Technology and Development, vol.12 (2014), p. 53-56.

[11] Malgorzata Bach, Aleksandra Werner. *Standardization of NoSQL database languages*.Springer International Publishing, (2014).

[12] C.F. Dai, M.D. Ma. Research on the optimization of MongoDB paging technology [J]. Computer Technology and Development, vol.6 (2018), p.97-101.

[13] Z.H.Wang, Z.D. Wang, J.T. Wang. MongoDB paging technology improvement and optimization [J]. Computer Technology and Automation, vol.3 (2015), p.127-130.

[14] C. Feng, C.D. Li, R. Li. *Indexing Techniques of Distributed Ordered Tables: A Survey and Analysis*. Journal of Computer Science and Technology: English Edition, vol. 1 (2018), p.169-189.

[15] G.L. Wang. Application research and program optimization of MongoDB database [J]. China Science and Technology Information, vol.20 (2011), p.93-95.