Wireless Communications and Mobile Internet Project Report

廖一鸣 5140219375
Mail：lglayama@sjtu.edu.cn

1. Abstract

This report mainly introduce the work I have done in 2016 spring semester for the course project—AceMap: Academic Information System. Specifically, my project is main focus on the optimization of the speed of queries to our database, which is significant for the user experience of our system. Since there some delay during the display of our system, our group try to use distributed tools based on different database to speed up for our system. Window function based on PostgreSQL is a very powerful method to accelerate the process of queries, which I use to optimize AceMap and finish my course project. The results show that our system will be more efficient based on the window function in PostgreSQL database.

2. Introduction

AceMap is a novel academic information system put forward by professor Xinbing Wang. The primary feature of our system is the academic maps, showing the relationship between the academic information in the database. Meanwhile, The website has already online with its excellent search function. Users can type in any academic keywords of their interest and get related search results, including papers, authors, affiliations, fields of study and other academic information.

In this semester, my teammates and I focus on the performance of our system for the query to our database. Our database, which is Microsoft ”mag” database, contains more than 100 million papers. More than 22000 topics have their unique paper maps in our system and More than 2200 affiliations have their unique author maps in our system. Sometimes, the response speed of our system is relatively slow so that it has a bad influence on the user experience. So we need to deal with this problem to make our system faster and more robust.

After the optimization of my project, the response speed of our system improves in a surprising way. Different queries have different improvements. In other word, each query increases in various degrees.

The following part of this report is organized by the process how I finish the optimization of the query. I will introduce the window function and PostgreSQL first. Then I will introduce the process of the conversion between the MYSQL and PostgreSQL. Moreover, I will elaborate how to rewrite the query with window function.

3. PostgreSQL and window function

PostgreSQL is an object-relational database with additional "object" features – with an emphasis on extensibility and standards compliance. As a database server, its primary functions are to store data securely and return that data in response to requests from other software applications. It can handle workloads ranging from small single-machine applications to large Internet-facing applications (or for data warehousing) with many concurrent user. Comparing to the MYSQL, PostgreSQL is more advanced in some ways, such as the support for the array and json, the scripts in server side like Python and R. And PostgreSQL have some advanced characteristic to improve the database's performance like Window function. In addition, PostgreSQL is a open-source database which is free for us.

At present, our system use MySQL to store the data. In most time, the performance is good while querying for tremendous data at the same time may come across the situation that the website have apparent delay. So I try to convert the database from MySQL to PostgreSQL to see whether the performance will improve.

A window function performs a calculation across a set of table rows that are somehow related to the current row. This is comparable to the type of calculation that can be done with an aggregate function. But unlike regular aggregate functions, use of a window function does not cause rows to become grouped into a single output row — the rows retain their separate identities.

The query of the users of our system always need search for massive data, so we try to use window function to rewrite the SQL statement now so that improve the speed of response.

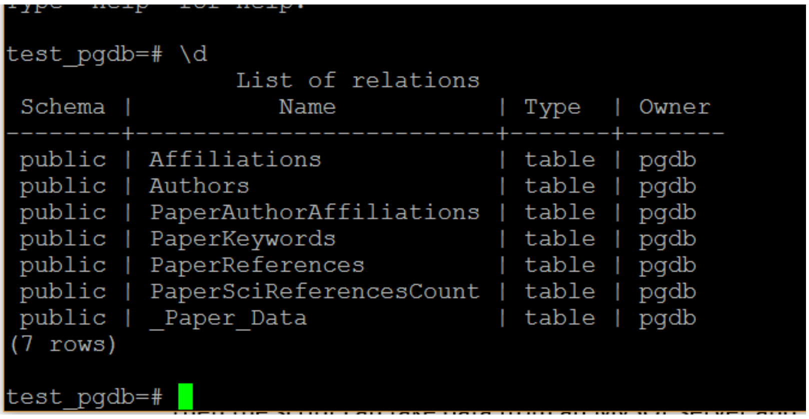
4. Conversion from MySQL to PostgreSQL

The first step of my project is convert the database from MySQL to PostgreSQL. Then we can test the performance on the two different database. First, we should consider about the difference of the data type between the two databases. Since there is not a one-to-one mapping between MySQL and PostgreSQL data types, listed below are the conversions that are applied.

MySQL	PostgreSQL
char	character
varchar	character varying
tinytext	text
mediumtext	text
text	text
longtext	text
tinyblob	bytea
mediumblob	bytea
blob	bytea
longblob	bytea
binary	bytea
varbinary	bytea

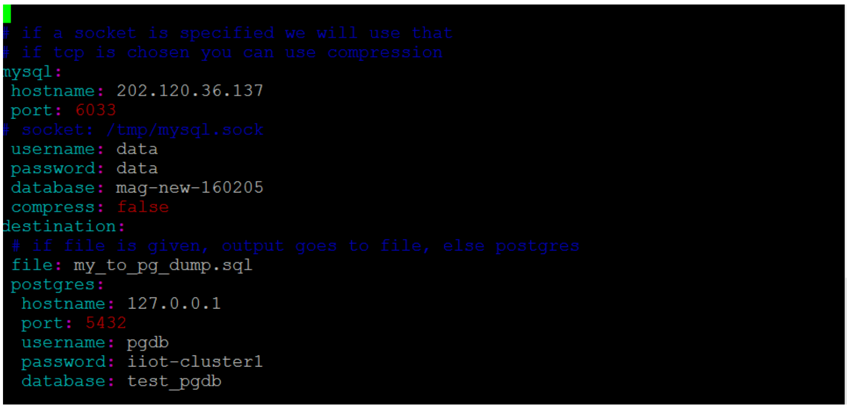
MySQL	PostgreSQL
bit	bit varying
tinyint	smallint
tinyint unsigned	smallint
smallint	smallint
smallint unsigned	integer
mediumint	integer
mediumint unsigned	integer
int	integer

Then I use a python script to accomplish the process that read data from MySQL and store data to the PostgreSQL. I establish a new PostgreSQL on our server first:



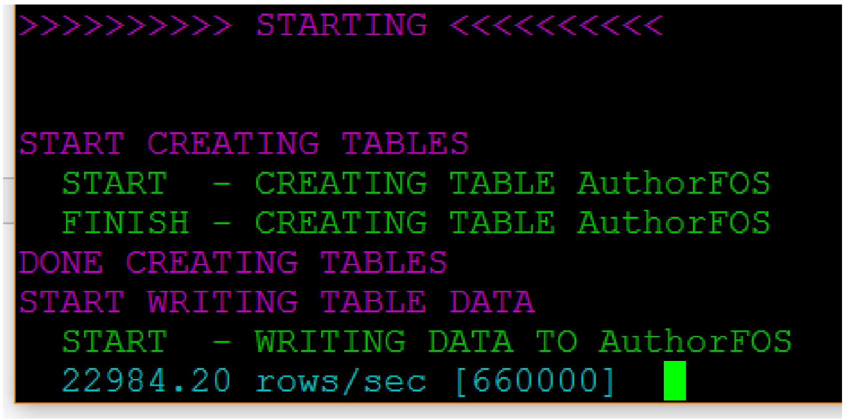
```
Type "help" for help.

test_pgdb=# \d
                 List of relations
 Schema |           Name            | Type  | Owner
--------+---------------------------+-------+-------
 public | Affiliations              | table | pgdb
 public | Authors                   | table | pgdb
 public | PaperAuthorAffiliations   | table | pgdb
 public | PaperKeywords             | table | pgdb
 public | PaperReferences           | table | pgdb
 public | PaperSciReferencesCount   | table | pgdb
 public | _Paper_Data               | table | pgdb
(7 rows)

test_pgdb=#
```

Then I modify the configuration file to connect the MySQL and PostgreSQL like this:



```
# if a socket is specified we will use that
# if tcp is chosen you can use compression
mysql:
 hostname: 202.120.36.137
 port: 6033
# socket: /tmp/mysql.sock
 username: data
 password: data
 database: mag-new-160205
 compress: false
destination:
 # if file is given, output goes to file, else postgres
 file: my_to_pg_dump.sql
 postgres:
  hostname: 127.0.0.1
  port: 5432
  username: pgdb
  password: iiot-cluster1
  database: test_pgdb
```

Then the script can take data from an MySQL server and write a PostgreSQL compatable dump file. Then I can use the dump file to store data to the PostgreSQL database:



```
>>>>>>>>>> STARTING <<<<<<<<<<


START CREATING TABLES
  START  - CREATING TABLE AuthorFOS
  FINISH - CREATING TABLE AuthorFOS
DONE CREATING TABLES
START WRITING TABLE DATA
  START  - WRITING DATA TO AuthorFOS
  22984.20 rows/sec [660000]
```

After that, I finish the conversion from MySQL to PostgreSQL.

5. Rewrite SQL query with window function

After the conversion of our database, we need use window function to improve the response speed of our system further. Window functions belong to a type of function known as a 'set function', which means a function that applies to a set of rows. The word 'window' is used to refer to the set of rows that the function works on.

The SQL statement we used now is find the specific author first, then try to do more specific query just like calculate· the citation of one author. Then I use the window function to rewrite the query. With the window function, the new query focus on the specific tasks with a large number of author. For some special query, window function can obviously improve the speed of response. I will give the examples to you show how I modify the SQL statement with window function.

Example: Find the number of SCI references to an author

Old query:

```
SELECT count(*),SUM(SCICitation) as sum from
PaperSciReferencesCount INNER JOIN
(select PaperID from PaperAuthorAffiliations
where AuthorID = ? ) AS TB1
on PaperSciReferencesCount.PaperReferenceID = TB1.PaperID
```

New query:

```
SELECT DISTINCT count(*),SUM(SCICitation) over (PARTITION BY
"AuthorID") as sum from "PaperSciReferencesCount" INNER JOIN
"PaperAuthorAffiliations" AS "TB1"
on "PaperSciReferencesCount"."PaperReferenceID" = "TB1"."PaperID"
where "AuthorID"= '?';
```

This is a very simple but common query for our system. The application of window function is the over() clause. With this clause, the statement can perform an aggregate operation against a user-defined range of rows (the window) and return a detail-level value for each row. So I can rewrite the SQL statement in a more efficient way.

6. Experiment

After do these two step to optimize our query. I test the different query respectively on MySQL, PostgreSQL and window function to see the improvement of my project. The following statement is used by me for the test.

(1) Find the number of SCI references to an author

```
SELECT    count(*),SUM(SCICitation) as sum from
          PaperSciReferencesCount INNER JOIN
          (select PaperID from PaperAuthorAffiliations
          where AuthorID = ? ) AS TB1
          on PaperSciReferencesCount.PaperReferenceID = TB1.PaperID
```

(2) Search for top five collaborators working with the two authors

```
SELECT AuthorID,AuthorName from `Authors` NATURAL JOIN
       (select count(PaperID) as coCount, AuthorID from PaperAuthorAffiliations
       natural join
       ( select * from (SELECT PaperID from PaperAuthorAffiliations where
       AuthorID = ? ) as TBMain NATURAL JOIN
       ( select PaperID from PaperAuthorAffiliations where AuthorID = ?) as TBAsso)
       as TB4 where AuthorID != ? and AuthorID != ? group by AuthorID
       order by coCount desc limit 5) as TBA
```

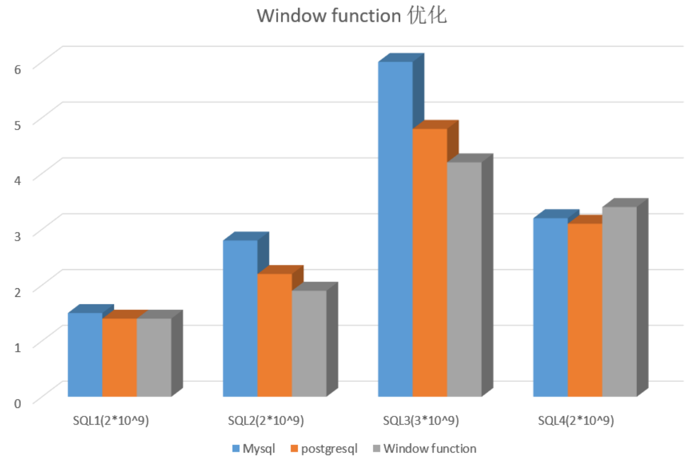
(3) Search for the maximum of 25 key words for the papers which cite a paper

```
SELECT FieldsOfStudyID,FieldsOfStudyName,FieldCitation from FieldsOfStudy
INNER JOIN (select FieldOfStudyIDMappedToKeyword,COUNT(*) as FieldCitation
from PaperKeywords INNER JOIN
(select PaperID from PaperReferences where PaperReferenceID = ? ) as TB1 on
TB1.PaperID = PaperKeywords.PaperID where `MagProvide`= 1 GROUP by
FieldOfStudyIDMappedToKeyword)
as TB2 on TB2.FieldOfStudyIDMappedToKeyword = FieldsOfStudy.FieldsOfStudyID
order by FieldCitation desc limit 25
```

(4) Find a joint recommendation to the author of the paper

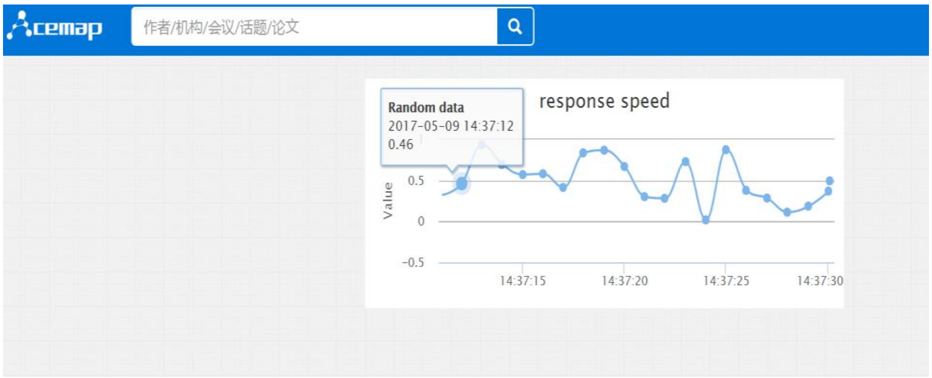
```
SELECT RecomID FROM
    (select `PaperID` from `PaperAuthorAffiliations` where `AuthorID` = ? ) as tb1
    Natural JOIN `PaperRecommenderList` group by RecomID
    having RecomID not in
    (select `PaperID` from `PaperAuthorAffiliations` where `AuthorID` = ? )
    order by `FutureRank` desc limit 15;
```

I get a result of the different query and I draw a picture for the results:



From the result we can find that there different improvement for the different SQL statement. Thinking about the data involved in different query, we can find that the more data involved, the better improvement generated. There are also many other factors have effect on the response speed. So the response speed is not strictly positive correlation with the amount of data.

In order to show the improvement for the Window function and PostgreSQL, our group make a demo in our website for the developer of Acemap:



Then we can see the performance of our system for further study and improvement.

7. Conclusion

Through this course project, I am able to master tools including SQL, python language. I also learned many details about the database so that I could have deeper understanding about the database. I also truly experience the big data. This project helps AceMap website to have faster response to the query from users. Currently our group have successfully solve the problem that the delay in some pages of our website.

I would like to thank all the fellows in our database group for helping me during this semester. Also my group leader Huo Xiaoyang and Luo Xiyi has given me plenty of useful suggestion about my project. Their helps made it a lot easier for me to accomplish this project.