CS-631

Course-Project

Distributed Caching in PostgreSQL

Team members

Akash Sahoo - 22M0797
Suman Saurav Panda - 22M0783
Yashwant Singh Parihar - 22M0798

Anant Utgikar - (Guide- PhD Scholar)

Dr. S. Sudarshan - (Instructor)

Project Proposal

Goal of the project

In the contemporary period, where almost all of us depend on cloud based services, caching at various levels plays a crucial role to speed up any task. As a part of our course project we are going to explore how caching at database level may improve the overall query processing task where we have a distributed database architecture for data storage.

More specifically we have client-server architecture where the client has a local database instance available with him which stores only a subset of actual data whereas the server database(say **master database**) has all the records. We would be providing a software abstraction of end to end data retrieval process for any external application.

Our approach to solve this problem

- We will install two instances of postgres in our local machine on different port numbers. One will act as local db-instance and the other will act remote db-instance.
- We will create a connection between these 2 instances(udp/tcp).
- Upon receiving a sql query from the external application we will check whether the required record is present in our local db-instance if not then we will query it in the remote db-instance.
- To achieve this task we will explore the postgres source code on how to process queries
- Initially we will target only small tasks like predicate search for equality or range queries.
- We will try to change the logic for data fetching so that if something is not found in the local instance we will use the connection and try to fetch that same query from the remote instance and we will update the local table using LRU policy.
- If something is not found in both local and remote then also we will maintain table where it will mark something is not present in the database so that when next time someone queries we can quickly respond that the record is not present without querying in the remote instance
- For performance analysis we will try to compare it with memcached.

Research and Experiments

1. We have tried to identify the code blocks handling the query part for simple select statements
    a. For this we have followed the Professor's lecture on Postgres internals and then executed simple queries in debug mode from the eclipse ide.
    b. For a single user mode the entry point for Postgres is in src/backend/main/main.c
    c. Inside main.c for simple select query it invokes the function in this order
        i. PostgresMain() → src/backend/tcop/postgres.c
        ii. exec_simple_query() → src/backend/tcop/postgres.c
        iii. **PortalRun()** → src/backend/tcop/pquey.c
        iv. PortalRunSelect() → src/backend/tcop/pquey.c
    d. Here an important thing we noticed that if we want to insert something the control flow comes to the same function **PortalRun()** from where it calls a different function PortalRunMulti()
2. Strategy for introducing new logic to connect to the remote server
    a. Use of **libpq C library** for connecting to remote servers from the C code. https://www.postgresql.org/docs/8.1/libpq.html
    b. Creating a new function **RemoteQuery()** in pquery.c
    c. **Invoking this function when nprocessed is 0 [variable nprocessed is the return value from PortalSelect()]**
    d. Then from exec_simple_query() function invoke the insert statement for the returned query result for the remote server using our function RemoteQuery()
        i. As an optimization we can call the PortalMulti() from the PortalRun() function itself instead of going back to exec_simple_query() and thereby by-passing a whole bunch of processing steps executed in between such as creating a parse tree for the insert statement

3. Changes required in the Server side (identified so far)
    https://www.thegeekstuff.com/2014/02/enable-remote-postgresql-connection/?utm_source=tuicool

    a. Modify pg_hba.conf to add client authentication record

```
# IPv4 local connections:
host    all         all         127.0.0.1/32          trust
# IPv6 local connections:
host    all         all         ::1/128               ident
```

```
# vi  /var/lib/pgsql/data/pg_hba.conf
host    all         all         192.168.101.20/24    trust
```

   b. Update Remote Server PSQL Configuration - Set listen_addresses = '*' in postgresql.conf file, path in general **/postgresql/install/data/postgresql.conf**

4. Use of Logical decoding plugins for easy checking of the changes to our database tables during our code implementation phase
   a. Need to integrate this with the local db server https://github.com/eulerto/wal2json

5. Advanced steps which we might be implementing (if time permits)
   a. Maintaining a LRU table in the database
   b. For queries whose final return value is null, keeping another table to store the key values and return quickly without investigating remote for the subsequent search

Project status on final semester demonstration

1. As an initial attempt to fetch the record we were able to use **libpq-fe** library in order to communicate with the server database and fetch the record from there for simple query statements. Here are some key notes for this experiment
   a. Despite being a simple task it had its own challenges for resolving header and make dependency so the libpq-fe library can be executed from the backend code of postgres
   b. Introduction of ***myremote()*** function in *postgres.c* to establish connection between local and remote db when client tries to connect to postgres server(in his local system)

c. Modification to the ***exec_simple_query()*** function in postgres.c which fetches data from remote and prints them to console when the record is not found in local.
d. Modification of function definition **PortalRun()** to indicate whether we have found any values or not from the local db and if the record is not present in the local db then we fetch it from remote in the *exec_simple_query() method.*
e. *We also figured out a few drawbacks with libpq. Handling complex queries and Inserting a remote record to the local db for the caching required huge efforts of bookkeeping(e.g. - extracting column header name and their type,* table schema informations, various integrity constraints etc)
f. Show demo for this part to professor

2. Our second approach to this problem is using postgres foreign data wrapper interface. This task is currently **ongoing.**
    a. We were able to create foreign tables on our local db and query on them to fetch the remote records.
    b. Now the real challenge using postgres_fdw is to parse the sql query and modify the query in such a way that it can access the remote db and fetch the whole record
        i. fetching the whole record makes sense because we want to cache the record in our local db and there would be query which would require projection of only a subset of columns
        ii. We would first insert the whole record in our local db and then re-execute the original query after insertion and show its result to the user.
        iii. Initially we were trying to write the logic for parsing of simple sql queries but then figured out that proper parsing would require too much coding so we are currently looking for two more alternatives
            1. Explore the pg_parse_query logic in postgres and try to make use of the parse tree created by it

               OR
            2. For the sql query parsing find open-source sql parser and integrate with the source code
3. Also the foreign table creation is happening by providing sql commands in the psql terminal now and we have to take the server information as config parameters and execute the sql commands from the backend itself. We are currently working on this feature.

Key takeaways and future work

During this course project we got ample exposure to postgres source code. The theory we learnt in class, we were able to see their actual implementation in it. This project gave us hands-on experience on how to modify such a huge source code by properly understanding the project structure, the make dependency and library and package dependency.

We were able to fetch the records from a remote database using libpq library. But as libpq library is a low level implementation, it needed more efforts for the caching part. Hence currently we are leveraging the foreign table concept provided by postgres which provides easy access to remote databases.

The task of implementing caching using postgres_fdw is ongoing and we would like to drive this project to completion during this vacation and this work would be available on git. All progress can be tracked here https://github.com/sumanpanda1997/distributed_caching_in_postgres

Thank you