Distributed Database Project Final Report

Team: Zhao Dongjie P18206023, Shi Lingqi P18206024

Author: Zhao Dongjie Phone: 13121595665 Email:

zdj8023first@163.com

Abstract

The project is aimed to design and implement a distributed application to support the reading experience for all kinds of users. There are two layers in the application. The first layer called client layer is implemented as a command line interface to support the operations and the data manipulation. The other layer is data center which is established on Ubuntu18.04 LTS. The data layer consists of two sets of MongoDB and Redis representing two locations in the Docker containers.Meeting the basic user operations and project requirements is the main consideration of the whole project.

Problem background and motivation

With the rapid development of the information society, students are facing increasing pressure. Therefore, mental health is getting more and more attention.Reading can bring spiritual pleasure and spiritual relaxation, so rich reading resources can help students alleviate a lot of stress.

The project is supposed to design and implement the electronic system to provide reading experience and stress reduction. The application is designed to support a smooth user experience and various user operations. As we are in the early stages of the project, we should try to be flexible.

Existing solutions

Innovation in today’s organizations comes from software, where all companies are becoming software companies and need to empower their developers to deliver new customer experiences quickly. Innovation can come in many different application formats - from traditional, monolithic applications to cloud-native and 12-factor applications.The Docker Platform is a set of integrated technologies and solutions for building, sharing and running container-based applications, from the developer’s desktop to the cloud. It is based on Docker’s core building blocks including Docker Desktop, Docker Hub, and Docker Engine.

With Docker we can easily implement the basic configuration and it’s far more a light-weight way compared to virtual machine tech.

There are many mature products in the choice of database. For relational

databases, common existing solutions for server-side database include MySQL, Oracle, SQL Server, etc. Among them, MySQL has a unique advantage because it is completely open source and free. On the other hand, MongoDB is a representative of a non-relational database, which has collections and

JSON-like documents instead of tables and rows. Non-relational databases provide more flexibility and scalability for data storage and operations. Since we are only at the beginning of the project, for quick deployment, we chose MongoDB as a solution for data storage.

At the same time, for a large number of identical read operations, we use the cache to speed up the system's response speed. Two popular options are Redis and Memcached, which are both in-memory key-value data stores. Among them, Redis offers a richer data type and is more efficient. Because the project includes different data types, we chose Redis as the caching technology in the project.

Problem definition

So we need to generate data based on project requirements, store the data in MongoDB, and put frequently accessed data into the Redis cache. The data is fragmented as required while the data is stored in the database.

At the same time, in order to quickly realize the main functions of the project, the project design and implementation of the command line client for user interaction

Proposed solutions

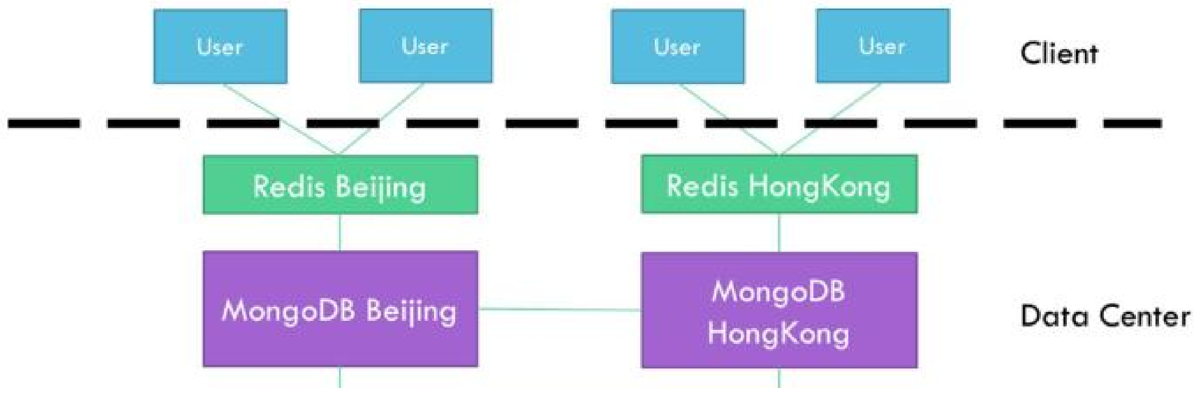


Figure1 Architecture of the system

In order to realize the function of the system, we divide the system into two layers: user layer and data layer. Depending on where the user is located, the query will first enter the corresponding Redis cache query. If the data is not in the cache, it will enter MongoDB for query.

Docker

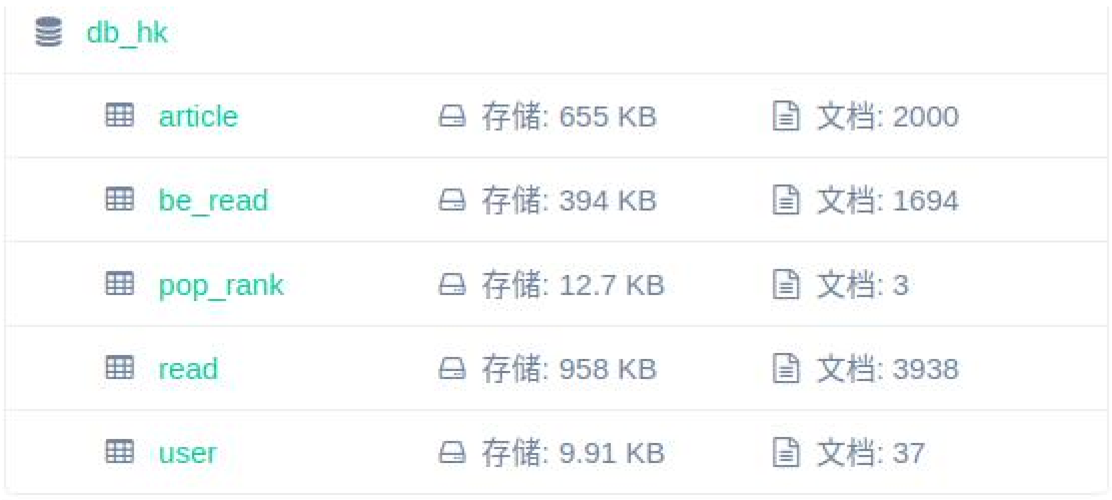
MongoDB and Redis have official images in Docker that can be pulled quickly and easily. We created two different containers for MongoDB and Redis through the Docker container by installing Docker on Ubuntu to simulate different database servers.

MongoDB

There are two Docker containers for MongoDB representing for Beijing and HongKong。

The different table are stored as collections in MongoDB and each data point is a document. User table is fragmented by region, hence the separate Beijing and Hong Kong database. The Read table is fragmented based on the User table by “uid”. Article category is used to fragment the Article table, where MongoDB

Beijing holds only articles with category “science” while MongoDB Hong Kong holds category “science” and “technology”. Be-Read table is fragmented based on the Article table using “aid” and the Popular Rank table is replicated in both databases, and they both are calculated based the basic three collections.



db_hk

article	存储: 655 KB	文档: 2000
be_read	存储: 394 KB	文档: 1694
pop_rank	存储: 12.7 KB	文档: 3
read	存储: 958 KB	文档: 3938
user	存储: 9.91 KB	文档: 37

Figure 2 the MongoDB of Hong Kong Data

Redis

There are two Docker containers for Redis representing for Beijing and Hong Kong. Data derived from database are stored as key-value in a Redis cache. Each time the user performs a read operation, it first goes to the Redis cache. If there is no result, it enters the MongoDB query and updates the data to the cache. Each time the user writes, the data is first written to MongoDB. After the write is successful, the data is updated to Redis to maintain user consistency.

User Command Line Interface

The project uses pycharm as a development platform, so it directly interacts with the system directly through pycharm. The specific work is developed by another teammate of mine, and she is also introduced in detail.

Solutions evaluation

The underlying foundation for quickly configuring a project with Docker is very fast and efficient. Also we use MongoAdmin and redis board to monitor the status of the both database running status.

The combination of MongoDB and Redis speed up the application effectively.

We tested the main features of the project to ensure that users are properly interacting with the system.

Conclusion

Future work

At present, the project only meets the basic requirements of the course design, and only the command line interface, so you can consider implementing a web user interface to provide a better experience for users.

Reference

[1]Anne N. Gade,Tine S. Larsen,Søren B. Nissen,Rasmus L. Jensen. REDIS: A value-based decision support tool for renovation of building portfolios[J]. Building and Environment,2018,142.

[2]Mohamed Mohamed,Robert Engel,Amit Warke,Shay Berman,Heiko Ludwig. Extensible persistence as a service for containers[J]. Future Generation Computer Systems,2019,97.

[3]. Mongodb Inc.; Mongodb, Inc. Files SEC Form 8-K, Current Report: (Dec. 21, 2018)[J]. Computers, Networks & Communications,2019.

Work allocation

Zhao Dongjie: System configuration, data generation, sharding, importation

Shi Lingqi: User interface and function implementation

System manual

System requirement:

Ubuntu 18.04.2 LTS

Docker 18.09.6

Mongo docker image(adminMongo)

Redis docker image(redisboard)

Python virtualenv Python 3.6.7

Pycharm

MongoDB Setup

Create volumes:

```
1. docker volume create db_hk
2. docker volume create db_bj
```

Start the MongoDB containers:

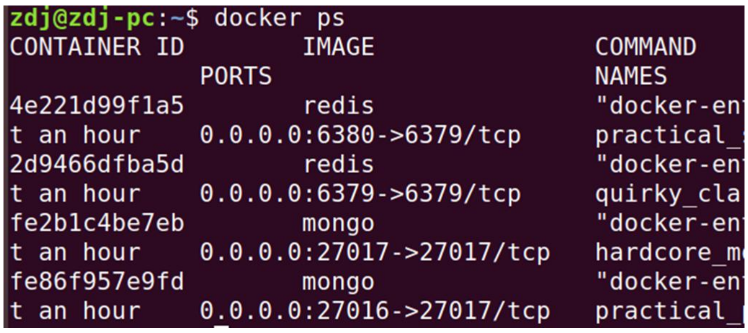
```
1. docker run -d -p 27016:27017 -v db_hk:/data/db
2. docker run -d -p 27017:27017 -v db_bj:/data/db
```

Redis Setup

Start the Redis containers:

```
1. docker run -d -p 6379:6379 redis
2. docker run -d -p 6380:6379 redis
```

After the container is running, the docker status should looks like the figure3

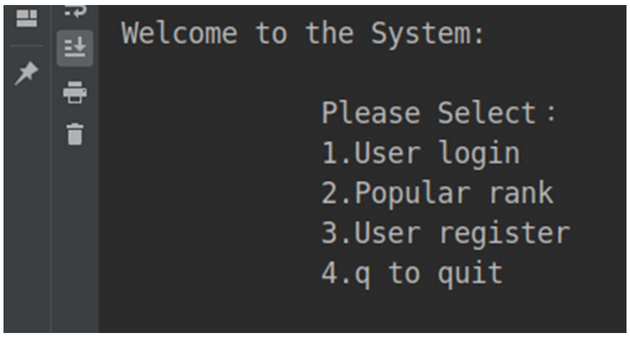


```
zdj@zdj-pc:~$ docker ps
CONTAINER ID        IMAGE               COMMAND
                PORTS                    NAMES
4e221d99f1a5        redis               "docker-en
t an hour    0.0.0.0:6380->6379/tcp      practical_
2d9466dfba5d        redis               "docker-en
t an hour    0.0.0.0:6379->6379/tcp      quirky_cla
fe2b1c4be7eb        mongo               "docker-en
t an hour    0.0.0.0:27017->27017/tcp    hardcore_m
fe86f957e9fd        mongo               "docker-en
t an hour    0.0.0.0:27016->27017/tcp    practical_
```

Figure 3 the docker container status

Command UI

After the system is running on Pycharm or terminal. The system entry should look like figure 4



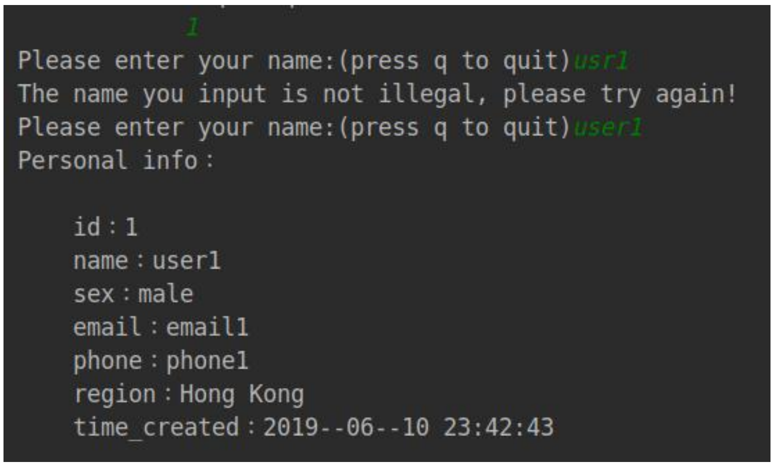
```
Welcome to the System:

              Please Select:
              1.User login
              2.Popular rank
              3.User register
              4.q to quit
```

Figure4 the system entry

Then you can choose the corresponding operation.

1. User login

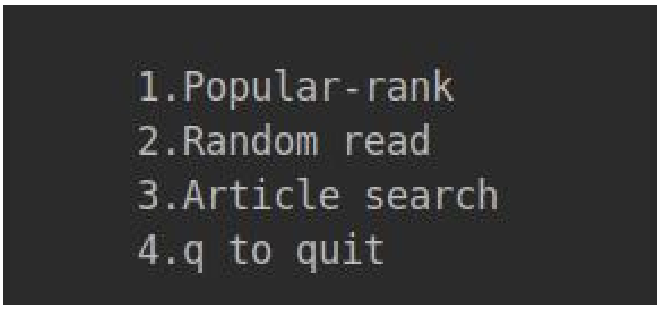
You can sign in the system by username like figure5



```
            1
Please enter your name:(press q to quit)usr1
The name you input is not illegal, please try again!
Please enter your name:(press q to quit)user1
Personal info:

    id:1
    name:user1
    sex:male
    email:email1
    phone:phone1
    region:Hong Kong
    time_created:2019--06--10 23:42:43
```

Figure5 user login

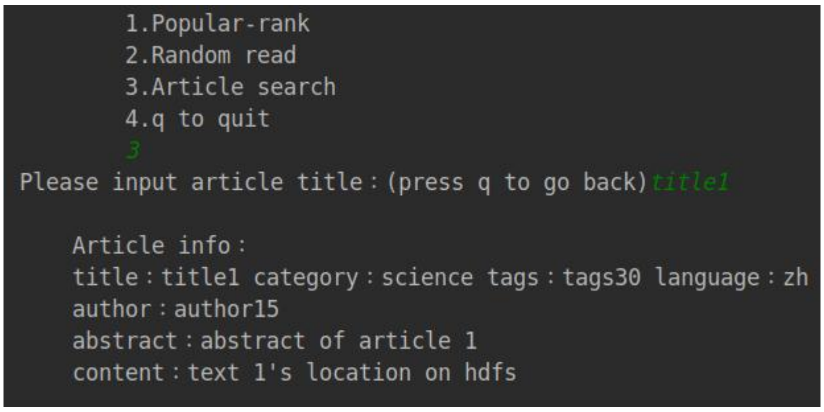
2. After login, you can have diff operation selection, like figure6



```
1.Popular-rank
2.Random read
3.Article search
4.q to quit
```

Figure6

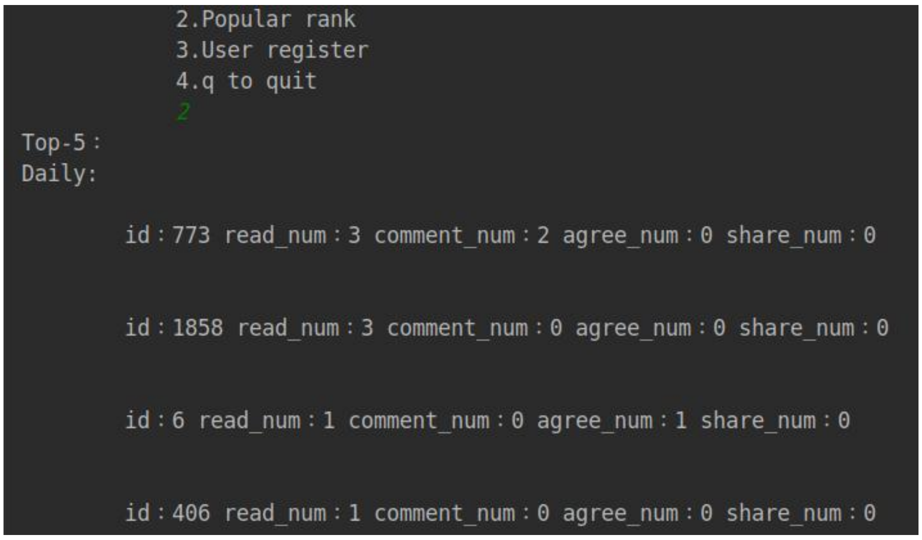
3. If you select 3 , then you can search article by title, like figure7



```
        1.Popular-rank
        2.Random read
        3.Article search
        4.q to quit
        3
Please input article title：(press q to go back)title1

    Article info：
    title：title1 category：science tags：tags30 language：zh
    author：author15
    abstract：abstract of article 1
    content：text 1's location on hdfs
```

Figure7

4. Also you can choose to see the popular rank daily, monthly and weekly, like figure8



```
            2.Popular rank
            3.User register
            4.q to quit
            2
Top-5：
Daily:

        id：773 read_num：3 comment_num：2 agree_num：0 share_num：0


        id：1858 read_num：3 comment_num：0 agree_num：0 share_num：0


        id：6 read_num：1 comment_num：0 agree_num：1 share_num：0


        id：406 read_num：1 comment_num：0 agree_num：0 share_num：0
```

Figure8