Document for MongoDB Coursework

Part 2

The Architecture of the Data Pipeline and GitHub Link

Shanchuan Wu

Sw9n14@soton.ac.uk

1 The Architecture of Data Pipeline

The architecture of data pipeline consists of several procedures or functions. The procedures or functions are shown as below.

1.1 Importing Data

First of all, data should be imported in the mongodb Database. In the system shell or command prompt, use the script below to insert the documents into the *tweets* collection in the *twitter* database. The *mongoimport* connects to a mongod instance running on localhost on port number 27017.

mongoimport -d twitter -c tweets --type csv --file microblog.csv --headerline

```
ShanchuanWu@Shanchuan-Wu-MacBook-Pro ~/D/microblogging>
mongoimport -d twitter -c tweets --type csv --file microblog.csv --headerline
2015-12-10T15:25:58.770+0000    connected to: localhost
2015-12-10T15:26:01.768+0000    [##......................] twitter.tweets       19.6 MB/198.5 MB (9.9%)
2015-12-10T15:26:04.770+0000    [####....................] twitter.tweets       39.3 MB/198.5 MB (19.8%)
2015-12-10T15:26:07.770+0000    [######..................] twitter.tweets       57.8 MB/198.5 MB (29.1%)
2015-12-10T15:26:10.768+0000    [########................] twitter.tweets       76.2 MB/198.5 MB (38.4%)
2015-12-10T15:26:13.769+0000    [##########..............] twitter.tweets       89.8 MB/198.5 MB (45.2%)
2015-12-10T15:26:16.767+0000    [############............] twitter.tweets       107.0 MB/198.5 MB (53.9%)
2015-12-10T15:26:19.767+0000    [###############.........] twitter.tweets       125.9 MB/198.5 MB (63.4%)
2015-12-10T15:26:22.768+0000    [#################.......] twitter.tweets       145.3 MB/198.5 MB (73.2%)
2015-12-10T15:26:25.768+0000    [###################.....] twitter.tweets       163.8 MB/198.5 MB (82.5%)
2015-12-10T15:26:28.768+0000    [#####################...] twitter.tweets       182.7 MB/198.5 MB (92.0%)
2015-12-10T15:26:31.362+0000    imported 1459861 documents
```

1.2 Preprocessing

When the data are inserted into the database, it is necessary to take some measures to clean the data. Because *id_members* or types of *text* of some messages are not meet the requirements, for instance, their *id_members* are negative. In this way, these messages should be considered as wrong data and need to be removed in this procedure.

The function *isPreProcessDatasuccess* is used to decide whether data are preprocessed or meet the requirements.

```
128     'isPreProcessDatasuccess': function () {
129         var count = db.getCollection(this.collectionName).find({
130             '$where': '(this.id_member < 0) || (typeof(this.text) != \"string\")'
131         }).count();
132         if (count > 0) {
133             return false;
134         } else {
135             return true;
136         }
137     },
```

```
'queryInit': function () { //need to be excuted only when the collction's document change or the first time to query

    if (true == this.isPreProcessDatasuccess()) {
        print("collection is already preprocessed");
    } else {
        print("begin to preprocess data");
        this.preProcessData();

    };
```

If all data meet the requirements, then function *preProcessData* is exeucted to analyses every document in the collection.

```
'preProcessData': function () {
    // var name = this.collectionName
    print("begin data parse , invalid data will be removed");
    db.getCollection(this.collectionName).find().forEach(function (it) {
        if ((it.id_member < 0) || (typeof (it.id_member) != "number") || (typeof (it.text) != "string")) {
            db.getCollection(queryByUser.collectionName).remove({
                "_id": it._id
            });

        }
    })
},
```

1.3 MapReduce

Map-reduce is a data processing paradigm for condensing large volumes of data into useful aggregated results. In this program, two *mapReduce* functions are defined. The first one is defined to create a new collection named *mrout*. By using this collection, it becomes easy to get the results of query1, query2 and query5. Similarly, another collection created by *mapReduce2* can help to group and count the messages according to different unigrams or bigrams. The detailed functions and queries can be seen in another document.

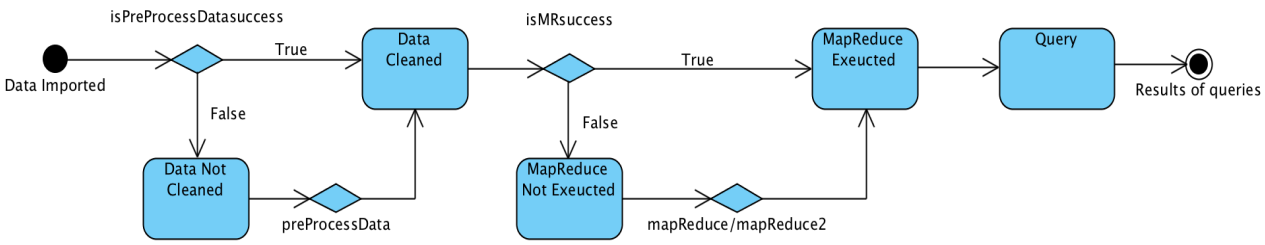
```
    if (true == this.isMRsuccess()) {
        print("collection is already mapReduced");

    } else {
        print("begin mapReduce");
        db.getCollection(this.collectionName).ensureIndex({
            'timestamp': 1
        });
        this.mapReduce();
        this.mapReduce2();
        return this.isMRsuccess();
    }
},
```

1.4 Querying and Output

After two *mapReduce* functions are executed, it is time to use querying function to query the documents in this database. The results of queries and the detailed functions for querying can be seen in another document.

1.5 The State Diagram of Data Pipeline



2 GitHub Link

https://github.com/358203708/MongodbCoursework