CS 6675 - Programming Assignment 4
Yuyang He
Problem chosen: #1 - Learning BigTable, NoSQL, Key-Value Systems
Option chosen: #1 - Beginner of Big Table/NoQL/Key-Value Store Systems

0 General

I have no experience in using MongoDB even though in my previous intern, I wrote some PHP to insert and query data from MongoDB. This is the first time I try to read, understand, and make benchmark for it. Even though the code may look simple, I take over 10 hours on this homework. Every script is modified at least three times for optimization or decrease the possible side effects (e.g. CSV handling may influence time of database operation).

1 Dataset

The dataset is the same for my assignment 2. Link: https://dumps.wikimedia.org/enwiki/20170120/enwiki-20170120-pages-articles-multistream-index.txt.bz2

I used Wikipedia's dump datasets as the datasets. The original data has the following format:

`number_1:number2:word1 word2 …`

In order to make it simple to run, I wrote a simple python script to make every word separated. Then for each line, there are 15 words (including numbers). The python script is located in the folder with the name "`WordsFilter.py`". All details of files are listed in section 3, part 2.

2 Screenshots

Please check the folder `/screenshots`. Every screenshot has it name with explanation.

3 Runtime Statistics

3.1 Test Environment

Macbook Pro, i5-6360U, 8GB DDR3-1886, 256GB, macOS 10.12.3
MongoDB 3.4.2 with shell 3.4.2, python driver 3.4.0
Python 2.7.10
Python packages are listed in `requirements.txt` by using `pip freeze` command.

3.2 Data Structure

As stated in section 1, each line in CSV file has 15 words mixed with numbers. And one line is treated as one data and will be inserted into MongoDB. I wrote a Python script to insert and record the time for insertion (see section 3.4 for details).

Original file: 188.2 MB when zipped as bz2, 798.9 MB after unzipped with 17,218,994 lines.

Pre-processed dataset: 795.9 MB with 6,119,730 lines.

Section 3.4 ~ 3.6 will generate an excel data in CSV format. But in order to have

graph, the 3 CSV file are processed as an excel file (results.xlsx) with 3tabs.
The database takes 1.873 GB.

3.3 Why Not Use YCSB

I read the github of YCSB (https://github.com/brianfrankcooper/YCSB) and understand it is a very powerful tool. But using the existing tool may not that be helpful for me to understand Mongo (even though configuring YCSB may makes me learn a lot). Instead, I wrote 3 scripts for evaluating 3 operations in MongoDB, insert, find, and delete, which is shown in the following sections 3.4 ~ 3.6.

3.4 Insertion

Script name: `insert_evaluation.py`
Excel data: `insert.csv`
Final processed data: `results.xlsx` (`insertion` tab)

Since insertion is very similar to insert into MySQL, I first use this command to test the speed of insertion. As stated, there are 6,119,730 lines in the CSV file so 6,119,730 in total data will be inserted into the MongoDB. To avoid same data, I simply add an automatically increasing ID in each data, so there are 16 fields in one data, one for ID, and other 15 from the CSV file.

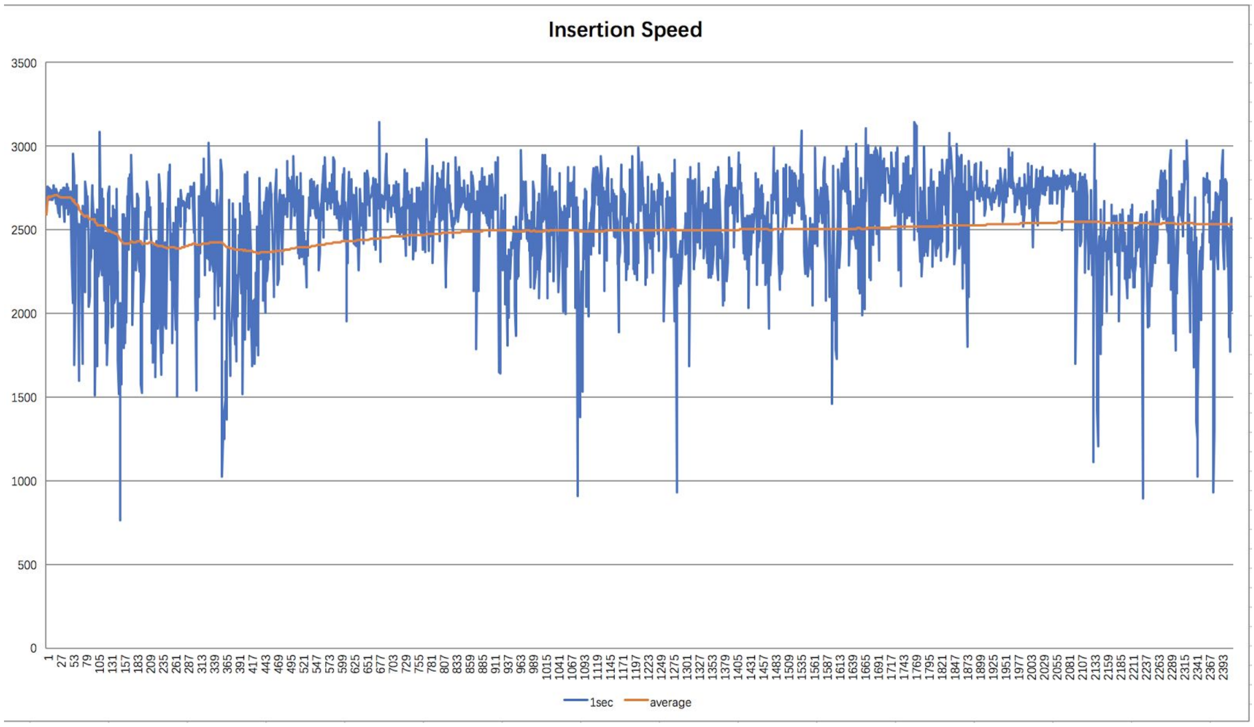
I have also considered the influence of dealing with CSV file that it may influence the speed of insertion. In order to have a fair game, I searched and have the two following conclusion.

1. Compared to insertion, handling one line of CSV file takes too short time which can be ignored. Since I re-write the program for 3 times to make sure the code handling only one line and then insert. So for each insertion, the workload should be similar.
2. I will add same CSV file handling process in both save and find section to make a fair game.

The results are shown as follows:

Total Tuples	Time Spent (Sec)	Avg. Insertion per Second	Maximum Speed	Minimum Speed
6,119,730	2,573	2,533	3,147	765

A more detailed figure of average speed and the differences are shown below. It can be tell that the average speed for 6 million tuples of data are relatively stable, with a higher start and a decrease and then a stable speed. For some insertions, it is obviously that the speed is extremely slow (at time 151, 359, 1082, 1284, and 2233). But I do not have an explanation since I am still very stupid and just learn MongoDB. But the average speed of insertion is close to 2,500 tuples per second, which is much better than some benchmark I found.



3.5 Save

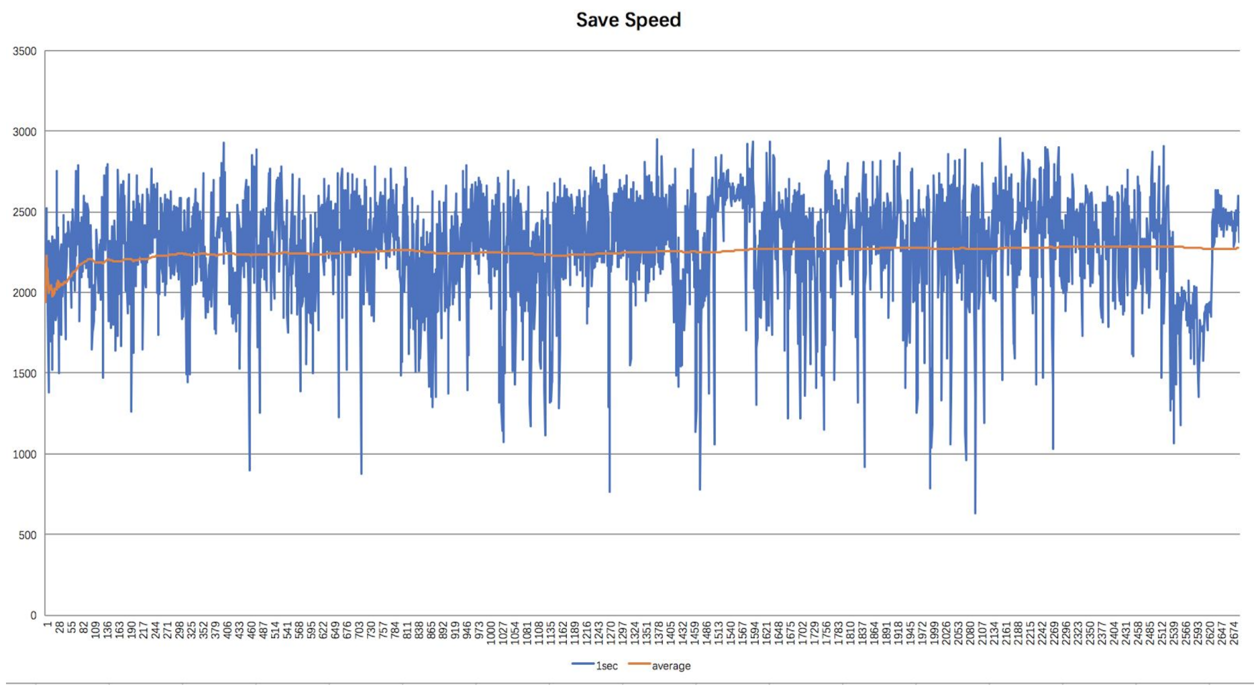
Script name: `save_evaluation.py`
Excel data: `save.csv`
Final processed data: `results.xlsx` (`save` tab)

The script is very similar to insert except insert is changed to save. Save is more like the update operation, where it can be used for evaluation of speed of combination of find and insert. In my script, I switch field 0 ~ 4 with field 10 ~ 14 and see how the speed changes to update the 6 million data. So the relational query could be like:

```
UPDATE test_database.test_collection
     SET field0 = row.field10,field1 = row.field11,
          field2 = row.field12,field3 = row.field13,
          field4 = row.field14,field10 = row.field0,
          field11 = row.field1,field12 = row.field2,
          field13 = row.field3,field14 = row.field4
     WHERE id = row.id
```

The results are shown as follows:

Total Tuples	Time Spent (Sec)	Avg. Save per Second	Maximum Speed	Minimum Speed
6,119,730	2,690	2,274	2,957	631



Save operation is absolutely slower than insertion since save is an update operation which contains query and modification. But surprisingly, its average speed seems more stable than insertion while a bigger differences thus variance and standard error is larger. It has a speed close to 2,300 tuples per second while it has an dramatically decrease after start. But I am not sure why is that. Also from seen the figure, there are some waves in the save operation. But it is relatively stable.

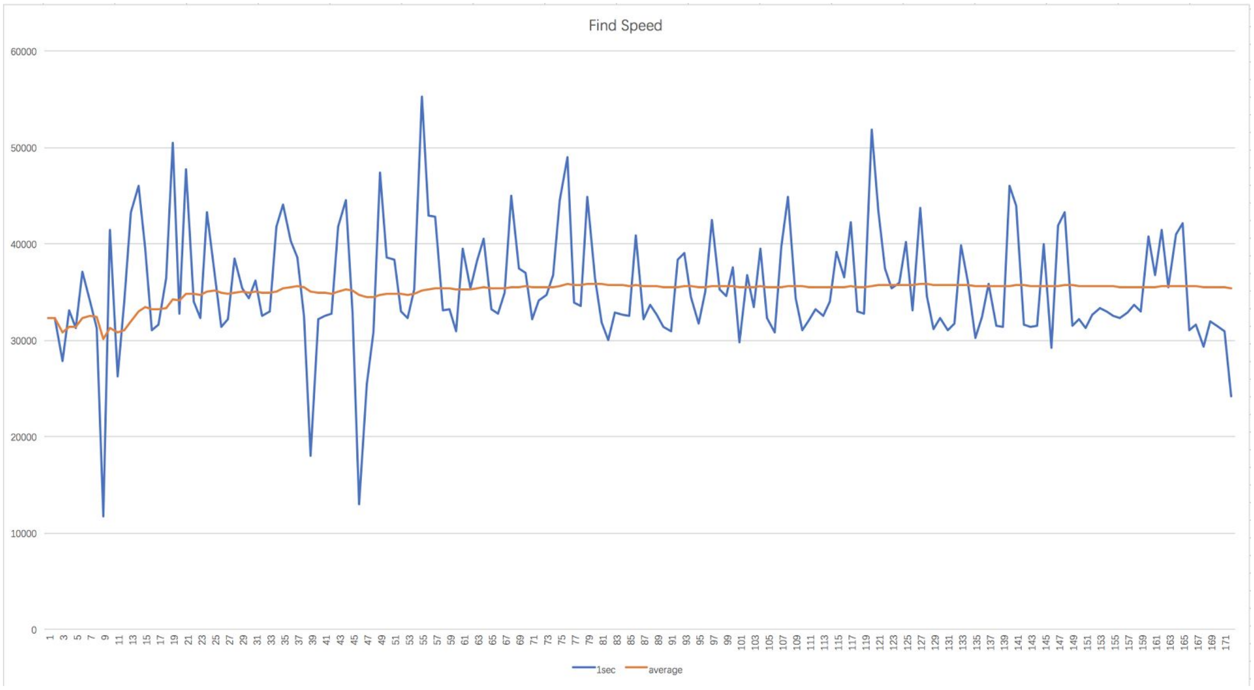
3.6 Find
Script name: `find_evaluation.py`
Excel data: `find.csv`
Final processed data: `results.xlsx` (`find` tab)

Similar to above, the search condition is set as

```
SELECT * FROM test_database.test_collection
     WHERE id = row.id AND field0 = row.field0
```

Where I query all fields.
The results are shown as follows:

Total Tuples	Time Spent (Sec)	Avg. Find per Second	Maximum Speed	Minimum Speed
6,119,730	172	35,441	55,346	11,691



Find is the fastest operation since it does not require write operation. It has an average speed close to 35,00 tuples per second. But I am not understanding why there is three slow query periods that only 10,000 ~ 20,000 queries are done in one second. But as usual, the relatively stable speed is shown.

4 Analysis

I have written some observations in section 3.4 ~ 3.6 with data shown. But I will talk about find function in MongoDB. Unlike relational database management system that requires a specific structure for the data (e.g. name, age, id, school, ...). MongoDB can use any key-value pair, which provides flexibility for run-time generated key-value pair that may have a specific name.

When I had my intern in the gene testing company, different person's gene testing results may have different components. Some people may have some gene pointed tested, but others may not or failed. So my job is to create PHP dictionary for different test results and send them to a pre-written function so results can be saved. But this is the first time I personally use MongoDB, trying to understand every working mechanism. And the result looks good, too.