Casandra and MongoDB Performance Comparison

Yue Guo
Indiana University
Smith Research Center
Bloomington, IN 47408, USA
yueguo@iu.edu

ABSTRACT

This project is to compare the performance of Cassandra and MongoDB under different coditions and different operations. In this project, Cassandra and MongoDB are deployed in three different platform, Macintosh macOS High Sierra, Google Cloud Ubuntu 14.04 and Raspberry pi Model B Ubuntu 16.04.

KEYWORDS

Cassandra, MongoDB, Raspberry Pi, Google Cloud, hid-sp18-508

1 INTRODUCTION

Cassandra and MongoDB are both not Relational DataBase Management System. There are four kinds of database, including Key-Value Stores, Big Tablle Clones, Document Database and Graph Database. Cassandra is a kind of Big Table Clones, while MongoDB is a kind of Document Database. Cassandra more focuses on Availability while MongoDB focuses more on Consistency. So there should be some difference in their performance when dealing with same data and same operations on the same platform. After deploying both on the same platform and testing same data on them, their performance can be compared. "Understanding the performance behavior of a NoSQL database like Apache Cassandra? under various conditions is critical. Conducting a formal proof of concept (POC) in the environment in which the database will run is the best way to evaluate platforms [2]."

2 TECHNOLOGY USED - SOFTWARE

This section describe the technologies that has been used through out the project. We used two NoSQL database here for Cloud Computing Courses, because for clouding computing NoSQL database works much better than traditional relational databases. Those NoSQL database can "handle load by spreading data among many servers, making them a natural fit for the cloud computing environment" [4].

2.1 Cassandra

Apache Cassandra is a kind of distributed NoSQL database management system, which is free and open source. It can handle a large amount of data on a large number of servers, which can provide high quality and no single point of failure [6].

And Cassandra has several advantages:

"1. Provides simple solutions for complex problems
2. Easy to learn
3. Reduces administration costs and overhead for non-core tasks
4. Fault tolerant to the extreme
5. Fast reads and extremely fast writes" [1].

2.2 MongoDB

MongoDB is also a king of NoSQL database management system, which is free and open source. Besides, it is a document-oriented database program. "MongoDB uses JSON-like documents with schemas" [8].

3 TECHNOLOGY USED - HARDWARE

This section describe the technologies that has been used through out the project.

3.1 Macintosh HD

The System Configuration:

MacBook Pro (Retina, 15-inch, Mid 2015)
Processor: 2.5GHz Intel Core i7
Memory: 16 GB 1600MHz DDR3

3.2 Raspberry Pi

Raspberry Pi can be considered as a small computer that can be used as a game machine or a platform to learn programming or a server [5]. Since it can be considered as a computer just like someone's own personal computer, it can be considered exactly the same as PC. Furthermore, using Raspberry Pi as a server can have several advantages. It can "provide you with a low-cost, silent, non-heating machine that fits easily into a room, and with very low power consumption of electricity" [3].

The hardware used for Raspberry Pi consists of:

one Raspberry Pi 3 Model B computer The pi is shown in following Figure 22.

[Figure 1 about here.]

one AmazonBasics High-speed HDMI cable

one Elecrow RPA050 HDMI 5-inch 800*480 TFT LCD Display with Touch Screen Monitor for Raspberry Pi The monitor is shown in following Figure 22.

[Figure 2 about here.]

one CanaKit 5V 2.5A Raspberry Pi 3 Model B Power Supply
one Happy Hacking keyboard

3.3 Google Cloud

Google Cloud provides a series of modular cloud services alongside a lot of management tools [7]. Just as Raspberry Pi, we can consider it as a PC. The thing that is different from Raspberry Pi is that it is a distributed system of computing. Although it is not open source and free, when open an account, it provides three hundred dollar which can be used in the first year.

The System Configuration of VM instance on Google Cloud:

Machine type: n1-standard-8 (8 vCPUs, 30 GB memory)

Operating System: Ubuntu 14.04

4 DEPLOYMENT

The following subsection will show how to deploy MongoDB and Cassandra on different operating system.

4.1 Deploy Cassandra on Mac OS X

Some steps are retrieved from https://www.datastax.com/2012/01/working-with-apache-cassandra-on-mac-os-x/

Download Cassandra

curl -OL http://downloads.datastax.com/community/dsc.tar.gz

Install Cassandra

tar -xzf dsc.tar.gz

Then switch to the new Cassandra bin directory and start up Cassandra:

cd dsc-cassandra-1.2.2/bin

sudo ./cassandra

Now that you have Cassandra running, the next thing to do is connect to the server and begin creating database objects. This is done with the Cassandra Query Language (CQL) utility. CQL is a very SQL-like language that lets you create objects as you?re likely used to doing in the RDBMS world.

./cqlsh

Connected to Test Cluster at localhost:9160.

For this brief introduction, we will just create a basic keyspace to hold some example data objects we will create:

cqlsh> create keyspace dev

Let us create a base table to hold train data:

cqlsh>use dev;

cqlsh:dev> create table test(uid varchar primary key);

cqlsh:dev> insert into test(uid) values('1');

4.2 Deploy Cassandra on Ubuntu 14.04

Some steps are retrieved from https://www.digitalocean.com/community/tutorials/how-to-install-cassandra-and-run-a-single-node-cluster-on-ubuntu-14-04

Installing Cassandra

echo deb http://www.apache.org/dist/cassandra/debian 22x main — sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list

Add the public key using this pair of commands, which must be run one after the other

gpg –keyserver pgp.mit.edu –recv-keys F758CE318D77295D

gpg –export –armor F758CE318D77295D — sudo apt-key add -

Update the package database once again:

sudo apt-get update

Finally, install Cassandra:

sudo apt-get install cassandra

Starting Cassandra:

sudo service cassandra status

Connecting to the Cluster:

sudo nodetool status

cqlsh

Then the following steps are the same with Mac Os

4.3 Deploy MongoDB on Mac OS X

Steps are retrieved from https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/

Install MongoDB Community Edition Manually

Download the binary files for the desired release of MongoDB.

Extract the files from the downloaded archive.

For example, from a system shell, you can extract through the tar command:

$tar - zxvf mongodb - osx - ssl - x86_64 - 3.6.4.tgz$

Copy the extracted archive to the target directory.

Copy the extracted folder to the location from which MongoDB will run.

mkdir -p mongodb

$cp - R - nmongodb - osx - ssl - x86_64 - 3.6.4/mongodb$

Ensure the location of the binaries is in the PATH variable.

The MongoDB binaries are in the bin/ directory of the archive. To ensure that the binaries are in your PATH, you can modify your PATH.

For example, you can add the following line to your shell?s rc file (e.g. /.bashrc):

$exportPATH =< mongodb - install - directory > /bin : PATH$

Replace $< mongodb - install - directory >$ with the path to the extracted MongoDB archive.

4.4 Deploy MongoDB on Ubuntu 14.04

Some steps are retrieved from https://docs.mongodb.com/manual/tutorial/install-mongodb-on-ubuntu/

Install MongoDB Community on Ubuntu Import the public key used by the package management system

sudo apt-key adv –keyserver hkp://keyserver.ubuntu.com:80 –recv 2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5 Create a list file for MongoDB

echo deb [arch=amd64] https://repo.mongodb.org/apt/ubuntu trusty /mongodb-org/3.6 multiverse — sudo tee /etc/apt/sources.list.d/mongodb-org-3.6.list

Reload local package database

sudo apt-get update

Install the MongoDB packages, Install the latest stable version of MongoDB.

sudo apt-get install -y mongodb-org

Run MongoDB Community Edition

sudo service mongod start

Stop MongoDB

sudo service mongod stop

Restart MongoDB

sudo service mongod restart

4.5 Deploy Raspberry Pi

Some steps are copied from https://www.ubuntu.com /download/iot/raspberry-pi-2-3

Setup an Ubuntu SSO account

1. An Ubuntu SSO account is required to create the first user on an Ubuntu Core installation.

2. Start by creating an Ubuntu SSO account

3. Import an SSH key into your Ubuntu SSO accout

Create installation medias for Ubuntu Core on Mac OS, some steps are copied from https://developer.ubuntu.com/core/get-started/installation-medias

Download Raspberry Pi 3

Insert your SD card or USB flash drive

Open a terminal window (Go to Application -¿ Utilities, you will find the Terminal app there), then run the following command:

diskutil list

Unmount your SD card with the following command:

diskutil unmountDisk ¡drive address¿

You can now copy the image to the SD card, using the following command:

sudo sh -c $'xzcat /Downloads/< imagefile > |sudoddof =< driveaddress > bs = 32m'$

5 BENCHMARK

Choose MongoDB as backend first, then use Cassandra as backend. Choose Macintosh first, then do same job on Raspberry Pi and finally on Google Cloud. The performance metrics include:

1. How fast can it process when using different backends

2. How fast can it process when under different hardware and different system

5.1 Benchmark - Scenarios

All the data used for inserting and searching is a single varchar type user-id.

All the details of MongoDB, Cassandra, Macintosh, Raspberry Pi and Google Cloud used here is introduced in section Technology Used - Software and Technology Used - Hardware.

Scenario 1. Insert 1000 user-id 1000 times on Macintosh with MongoDB

Scenario 2. Insert 5000 user-id 1000 times on Macintosh with MongoDB

Scenario 3. Insert 10000 user-id 1000 times on Macintosh with MongoDB

Scenario 4. Search 1000 user-id 1000 times on Macintosh with MongoDB

Scenario 5. Search 5000 user-id 1000 times on Macintosh with MongoDB

Scenario 6. Search 10000 user-id 1000 times on Macintosh with MongoDB

Scenario 7. Insert 1000 user-id 1000 times on Raspberry Pi with MongoDB

Scenario 8. Insert 5000 user-id 1000 times on Raspberry Pi with MongoDB

Scenario 9. Insert 10000 user-id 1000 times on Raspberry Pi with MongoDB

Scenario 10. Search 1000 user-id 1000 times on Raspberry Pi with MongoDB

Scenario 11. Search 5000 user-id 1000 times on Raspberry Pi with MongoDB

Scenario 12. Search 10000 user-id 1000 times on Raspberry Pi with MongoDB

Scenario 13. Insert 1000 user-id 1000 times on Google Cloud with MongoDB

Scenario 14. Insert 5000 user-id 1000 times on Google Cloud with MongoDB

Scenario 15. Insert 10000 user-id 1000 times on Google Cloud with MongoDB

Scenario 16. Search 1000 user-id 1000 times on Google Cloud with MongoDB

Scenario 17. Search 5000 user-id 1000 times on Google Cloud with MongoDB

Scenario 18. Search 10000 user-id 1000 times on Google Cloud with MongoDB

Scenario 19. Insert 1000 user-id 1000 times on Macintosh with Cassandra

Scenario 20. Insert 5000 user-id 1000 times on Macintosh with Cassandra

Scenario 21. Insert 10000 user-id 1000 times on Macintosh with Cassandra

Scenario 22. Search 1000 user-id 1000 times on Macintosh with Cassandra

Scenario 23. Search 5000 user-id 1000 times on Macintosh with Cassandra

Scenario 24. Search 10000 user-id 1000 times on Macintosh with Cassandra

Scenario 25. Insert 1000 user-id 1000 times on Raspberry Pi with Cassandra

Scenario 26. Insert 5000 user-id 1000 times on Raspberry Pi with Cassandra

Scenario 27. Insert 10000 user-id 1000 times on Raspberry Pi with Cassandra

Scenario 28. Search 1000 user-id 1000 times on Raspberry Pi with Cassandra

Scenario 29. Search 5000 user-id 1000 times on Raspberry Pi with Cassandra

Scenario 30. Search 10000 user-id 1000 times on Raspberry Pi with Cassandra

Scenario 31. Insert 1000 user-id 1000 times on Google Cloud with Cassandra

Scenario 32. Insert 5000 user-id 1000 times on Google Cloud with Cassandra

Scenario 33. Insert 10000 user-id 1000 times on Google Cloud with Cassandra

Scenario 34. Search 1000 user-id 1000 times on Google Cloud with Cassandra

Scenario 35. Search 5000 user-id 1000 times on Google Cloud with Cassandra

Scenario 36. Search 10000 user-id 1000 times on Google Cloud with Cassandra

6 ASSESSMENT

What parameter:

use time package, time_after - time_pre

How to process:

Redirect the output of time result into the specified file. Use Matlab program and compare the output of MongoDB and Cassandra to draw some plots.

7 RESULTS

Compare insert 1000 data 1000 times on Macintosh 22.

[Figure 3 about here.]

Compare insert 5000 data 1000 times on Macintos 22.

[Figure 4 about here.]

Compare insert 10000 data 1000 times on Macintos 22.

[Figure 5 about here.]

Compare search 1000 data 1000 times on Macintosh 22.

[Figure 6 about here.]

Compare search 5000 data 1000 times on Macintos 22.

[Figure 7 about here.]

Compare search 10000 data 1000 times on Macintos 22.

[Figure 8 about here.]

Compare insert 1000 data 1000 times on Raspberry Pi 22.

[Figure 9 about here.]

Compare insert 5000 data 1000 times on Raspberry Pi 22.

[Figure 10 about here.]

Compare insert 10000 data 1000 times on Raspberry Pi 22.

[Figure 11 about here.]

Compare search 1000 data 1000 times on Raspberry Pi 22.

[Figure 12 about here.]

Compare search 5000 data 1000 times on Raspberry Pi 22.

[Figure 13 about here.]

Compare insert 1000 data 1000 times on Google Cloud 22.

[Figure 14 about here.]

Compare insert 5000 data 1000 times on Google Cloud 22.

[Figure 15 about here.]

Compare insert 10000 data 1000 times on Google Cloud 22.

[Figure 16 about here.]

Compare search 1000 data 1000 times on Google Cloud 22.

[Figure 17 about here.]

Compare search 5000 data 1000 times on Google Cloud 22.

[Figure 18 about here.]

Compare search 10000 data 1000 times on Google Cloud 22.

[Figure 19 about here.]

Compare insert 1000 data 1000 times on Mac, Raspberry Pi and Google Cloud 22.

[Figure 20 about here.]

Compare insert 5000 data 1000 times on Mac, Raspberry Pi and Google Cloud 22.

[Figure 21 about here.]

Compare insert 10000 data 1000 times on Mac, Raspberry Pi and Google Cloud 22.

[Figure 22 about here.]

8 CONCLUSION

Compare the performance of MongoDB and Cassandra on Macintosh.

8.1 Scenario 1, 4, 19 and 22

Insert and search 1000 data 1000 times on Macintosh (Scenario 1,4 compare with Scenario 19,22):

The average insert time of mongoDB is 0.2 ms, while the average insert time of Cassandra is 0.4 ms.

The average search time of mongoDB is 0.007 ms, while the average insert time of Cassandra is 0.8 ms.

8.2 Scenario 2, 5, 20 and 23

Insert and search 5000 data 1000 times on Macintosh (Scenario 2, 5 compare with Scenario 20, 23):

The average insert time of mongoDB is 1.2 ms, while the average insert time of Cassandra is 0.4 ms.

The average search time of mongoDB is 0.03 ms, while the average insert time of Cassandra is 5 ms.

However, since this test are run on Macintosh parallely, they impact each other's performance.

8.3 Scenario 3, 6, 21 and 24

Insert and search 10000 data 1000 times on Macintosh (Scenario 3, 6 compare with Scenario 21, 24):

The average insert time of mongoDB is 2.2 ms, while the average insert time of Cassandra is 5 ms.

The average search time of mongoDB is 0.07 ms, while the average insert time of Cassandra is 10 ms.

Overall, Cassandra takes more time to insert and search, especially search.

However, on Raspberry Pi the difference of their performance become closer.

The performance of MongoDB and Cassandra on Google Cloud is more stable.

8.4 Scenario 7,10 ,25 and 28

Insert and search 1000 data 1000 times on Raspberry Pi (Scenario 7, 10 compare with Scenario 25, 28):

The average insert time of mongoDB is 0.4 ms, while the average insert time of Cassandra is 0.5 ms.

The average search time of mongoDB is 0.007 ms, while the average insert time of Cassandra is 1.3 ms.

As we can see from the plot, the result is much more stable than w

8.5 Scenario 8, 11, 26 and 29

Insert and search 5000 data 1000 times on Raspberry Pi (Scenario 8,11 compare with Scenario 26, 29):

The average insert time of mongoDB is 2.0 ms, while the average insert time of Cassandra is 4.5 ms.

The average search time of mongoDB is 0.04 ms, while the average insert time of Cassandra is 6 ms.

However, since this test are run on Macintosh parallely, they impact each other's performance.

8.6 Scenario 9, 12, 27 and 30

Insert and search 10000 data 1000 times on Macintosh (Scenario 9,12 compare with Scenario 27, 30):

The average insert time of mongoDB is 4.5 ms, while the average insert time of Cassandra is 6 ms.

The average search time of mongoDB is 0.08 ms, while the average insert time of Cassandra is 14 ms.

Overall, Cassandra takes more time to insert and search, especially search.

However, on Raspberry Pi performance of both the MongoDB and Cassandra become worse.

8.7 Scenario 13, 16, 31 and 34

Insert and search 1000 data 1000 times on Google Cloud (Scenario 13,16 compare with Scenario 31, 34):

The average insert time of mongoDB is 0.25 ms, while the average insert time of Cassandra is 0.5 ms.

The average search time of mongoDB is 0.006 ms, while the average insert time of Cassandra is 1.3 ms.

As we can see from the plot, the result is much more stable than running on Macintosh, because they are run serially. And no any other applications run at the same time.

8.8 Scenario 14, 17, 32 and 35

Insert and search 5000 data 1000 times on Google Cloud (Scenario 14, 17 compare with Scenario 32, 35):

The average insert time of mongoDB is 1.2 ms, while the average insert time of Cassandra is 2.0 ms.

The average search time of mongoDB is 0.03 ms, while the average insert time of Cassandra is 6 ms.

With the amount of queries we insert and search every time, the result is even more stable.

8.9 Scenario 15,18, 33 and 36

Insert and search 10000 data 1000 times on Google Cloud (Scenario 15,18 compare with Scenario 33, 36):

The average insert time of mongoDB is 2.3 ms, while the average insert time of Cassandra is 5 ms.

The average search time of mongoDB is 0.07 ms, while the average insert time of Cassandra is 13 ms.

Overall, Cassandra takes more time to insert and search, especially search. And the performance of MongoDB and Cassandra on Google Cloud is more stable.

8.10 Scenario 1, 7 and 13

Insert 1000 data 1000 times on Mac, Raspberry Pi and Google Cloud (Scenario 1, 7 compare with Scenario 13):

The average insert time on Mac is 0.22 ms, the average insert time on Google Cloud is 0.25 ms while the average insert time on Raspberry Pi is 0.4ms.

8.11 Scenario 2, 8 and 14

Insert 50000 data 1000 times on Mac, Raspberry Pi and Google Cloud (Scenario 2, 8 compare with Scenario 14):

The average insert time on Mac is 1.22 ms, the average insert time on Google Cloud is 1.25 ms while the average insert time on Raspberry Pi is 2.4ms.

8.12 Scenario 3, 9 and 15

Insert 10000 data 1000 times on Mac, Raspberry Pi and Google Cloud (Scenario 3, 9 compare with Scenario 15):

The average insert time on Mac is 2.22 ms, the average insert time on Google Cloud is 2.4 ms while the average insert time on Raspberry Pi is 5.4ms.

Overall, performance on Raspberry Pi is always the worst.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Gregor von Laszewski for his support and suggestions to write this paper.

REFERENCES

[1] Akbar S. Ahmed. 2015. 5 reasons why you should use Cassandra. Exponential IO website. (2015). http://exponential.io/blog/2015/01/13/5-reasons-why-you-should-use-cassandra/

[2] Datastax. 2018. Benchmarking NoSQL Databases: Cassandra vs. MongoDB vs. HBase vs. Couchbase. Datastax website. (2018). https://www.datastax.com/nosql-databases/benchmarks-cassandra-vs-mongodb-vs-hbase

[3] Mis jour le. 2017. Why you should but a Raspberry Pi? website. (2017). https://howtoraspberrypi.com/why-buy-raspberry-pi//

[4] MongoDB. 2018. Cloud Computing. MongoDB website. (2018). https://www.mongodb.com/big-data-explained

[5] Raspberry Pi. 2018. Raspberry Pi Blog. Raspberry Pi website. (2018). https://www.raspberrypi.org/

[6] Wikipedia. 2018. Cassandra Wiki. Wikipedia website. (2018). https://en.wikipedia.org/wiki/Apache_Cassandra

[7] Wikipedia. 2018. Google Cloud Wiki. Wikipedia website. (2018). https://en.wikipedia.org/wiki/Google_Cloud_Platform

[8] Wikipedia. 2018. MongoDB Wiki. Wikipedia website. (2018). https://en.wikipedia.org/wiki/MongoDB

A CHKTEX

```
cd ../../hid-sample; git pull
Already up-to-date.
cp ../../hid-sample/paper/Makefile .
cp ../../hid-sample/paper/report.tex .
WARNING: line longer than 80 characters
       84: have  several advantages. It can ``provide you with a low-cost, silent, non-heating

WARNING: line longer than 80 characters
       81: tutorials/how-to-install-cassandra-and-run-a-single-node-cluster-on-ubuntu-14-04

WARNING: line longer than 80 characters
       86: Add the public key using this pair of commands, which must be run one after the other

WARNING: line longer than 80 characters
     135: The MongoDB binaries are in the bin/ directory of the archive. To ensure that the binaries are in your PATH, you can

WARNING: line longer than 80 characters
       86: For example, you can add the following line to your shell?s rc file (e.g. ~/.bashrc):

WARNING: line longer than 80 characters
       86: Replace $<mongodb-install-directory>$ with the path to the extracted MongoDB archive.

WARNING: line longer than 80 characters
     107: sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 2930ADAE8CAF5059EE73BB4B58712A2291FA4AD5

WARNING: line longer than 80 characters
       81: Create installation medias for Ubuntu Core on Mac OS, some steps are copied from

WARNING: line longer than 80 characters
       84: Open a terminal window (Go to Application -> Utilities, you will find the Terminal

WARNING: line longer than 80 characters
       82: sudo sh -c $'xzcat ~/Downloads/<image file> | sudo dd of=<drive address> bs=32m'$

WARNING: line longer than 80 characters
       86: Choose Macintosh first, then do same job on Raspberry Pi and finally on Google Cloud.

WARNING: line longer than 80 characters
       87: Compare insert 1000 data 1000 times on Mac, Raspberry Pi and Google Cloud~\ref{f:fly}.

WARNING: line longer than 80 characters
       87: Compare insert 5000 data 1000 times on Mac, Raspberry Pi and Google Cloud~\ref{f:fly}.

WARNING: line longer than 80 characters
       88: Compare insert 10000 data 1000 times on Mac, Raspberry Pi and Google Cloud~\ref{f:fly}.

WARNING: line longer than 80 characters
       82: The average search time of mongoDB is 0.007 ms, while the average insert time of

WARNING: line longer than 80 characters
       81: The average search time of mongoDB is 0.03 ms, while the average insert time of
```

```
WARNING: line longer than 80 characters
       83: However, since this test are run on Macintosh parallely, they impact each other's

WARNING: line longer than 80 characters
       81: The average search time of mongoDB is 0.07 ms, while the average insert time of

WARNING: line longer than 80 characters
       82: The average search time of mongoDB is 0.007 ms, while the average insert time of

WARNING: line longer than 80 characters
       81: The average search time of mongoDB is 0.04 ms, while the average insert time of

WARNING: line longer than 80 characters
       83: However, since this test are run on Macintosh parallely, they impact each other's

WARNING: line longer than 80 characters
       81: The average search time of mongoDB is 0.08 ms, while the average insert time of

WARNING: line longer than 80 characters
       86: However, on Raspberry Pi performance of  both the MongoDB and Cassandra become worse.

WARNING: line longer than 80 characters
       81: The average insert time of mongoDB is 0.25 ms, while the average insert time of

WARNING: line longer than 80 characters
       82: The average search time of mongoDB is 0.006 ms, while the average insert time of

WARNING: line longer than 80 characters
       88: As we can see from the plot, the result is much more stable than running on Macintosh,

WARNING: line longer than 80 characters
       84: because they are run serially. And no any other applications run at the same time.

WARNING: line longer than 80 characters
       81: The average search time of mongoDB is 0.03 ms, while the average insert time of

WARNING: line longer than 80 characters
       81: The average search time of mongoDB is 0.07 ms, while the average insert time of

WARNING: line longer than 80 characters
       84: The average insert time on Mac  is 0.22 ms, the average insert time on Google Cloud

WARNING: line longer than 80 characters
       83: The average insert time on Mac is 1.22 ms, the average insert time on Google Cloud

WARNING: line longer than 80 characters
       83: The average insert time on Mac is 2.22 ms, the average insert time on Google Cloud

Warning 26 in content.tex line 164: You ought to remove spaces in front of punctuation.
sudo ./cassandra
    ^

Warning 36 in content.tex line 184: You should put a space in front of parenthesis.
cqlsh:dev$>$ create table test(uid varchar primary key);
                              ^

Warning 36 in content.tex line 186: You should put a space in front of parenthesis.
cqlsh:dev$>$ insert into test(uid) values('1');
                             ^

Warning 36 in content.tex line 186: You should put a space in front of parenthesis.
cqlsh:dev$>$ insert into test(uid) values('1');
                                          ^

Warning 25 in content.tex line 237: You might wish to put this between a pair of `{}'
$tar -zxvf mongodb-osx-ssl-x86_64-3.6.4.tgz$
                           ^^

Warning 25 in content.tex line 246: You might wish to put this between a pair of `{}'
$cp -R -n mongodb-osx-ssl-x86_64-3.6.4/ mongodb$
                           ^^

Warning 39 in content.tex line 252: Double space found.
For example, you can add the following line to your shell?s rc file (e.g. ~/.bashrc):
                                                                      ^
```

List of Figures

Figure 1: Raspberry Pi 3 Model B

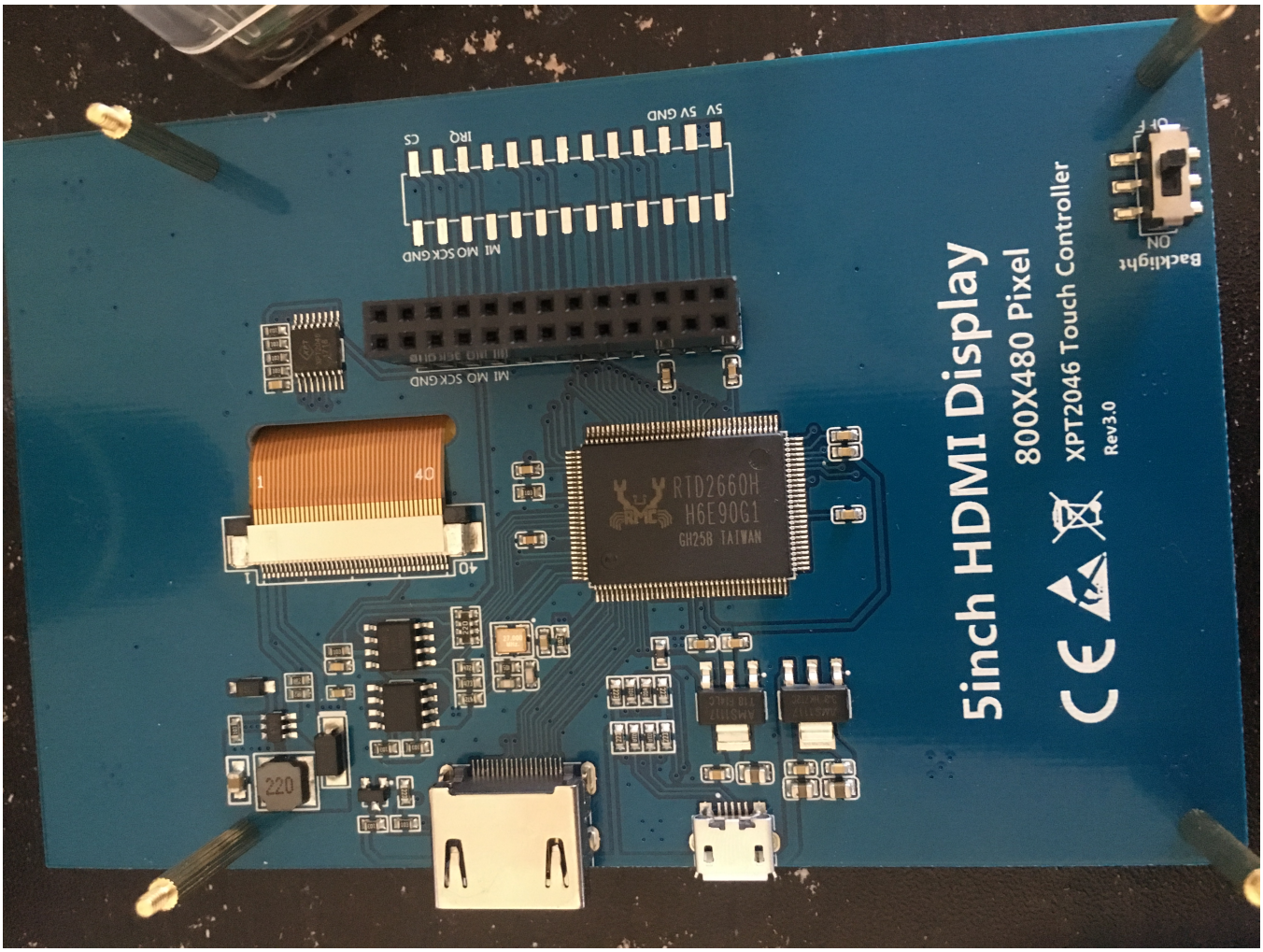

Figure 2: Raspberry Pi 3 Model B

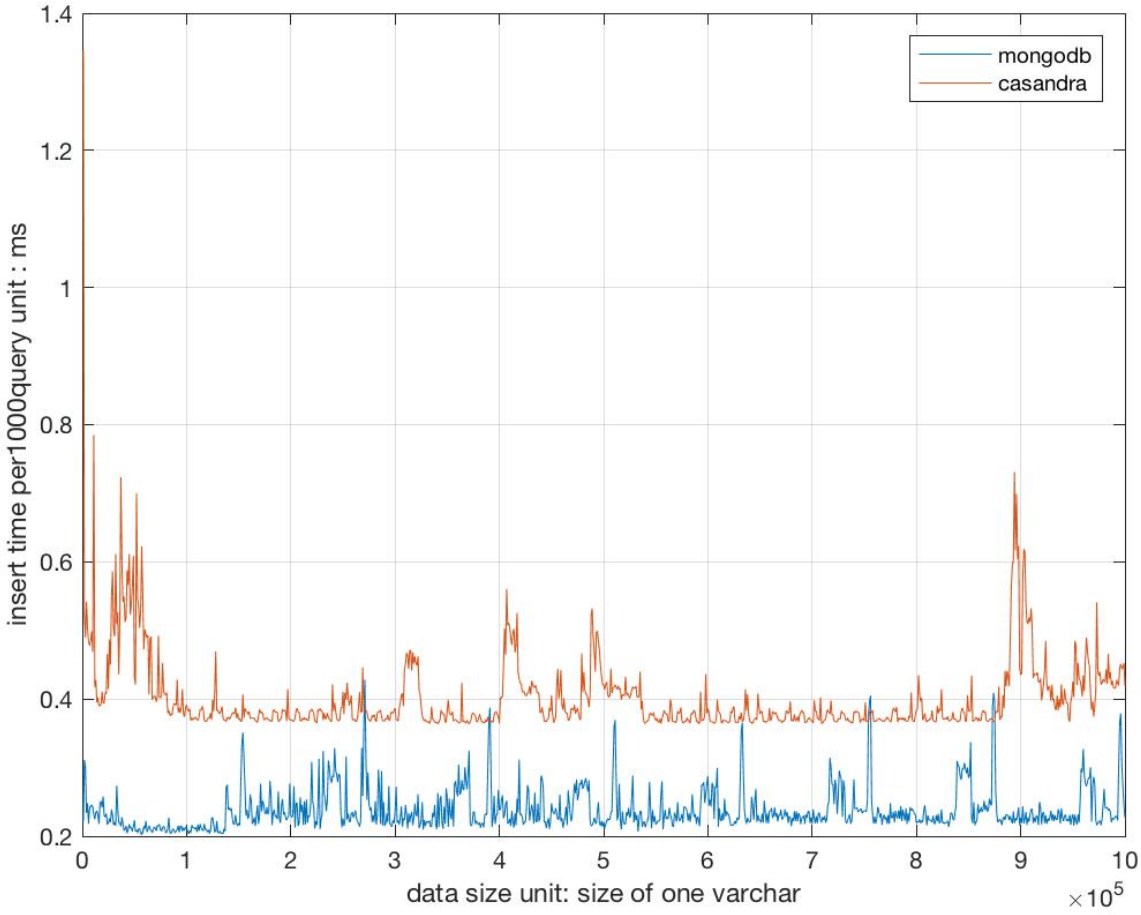

Figure 3: Insert 1000 data 1000 times on Macintosh (Scenario 1 compare with Scenario 19)

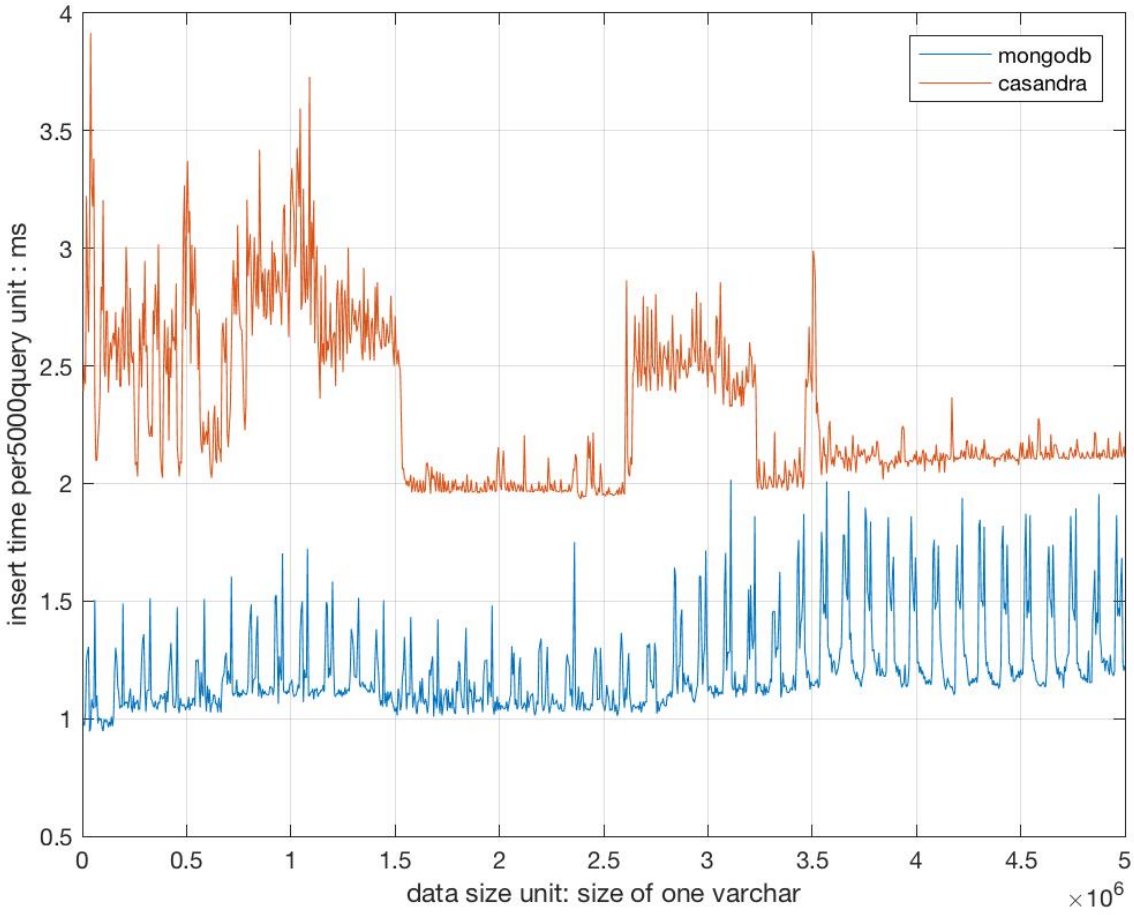

Figure 4: Insert 5000 data 1000 times on Macintosh (Scenario 2 compare with Scenario 20)

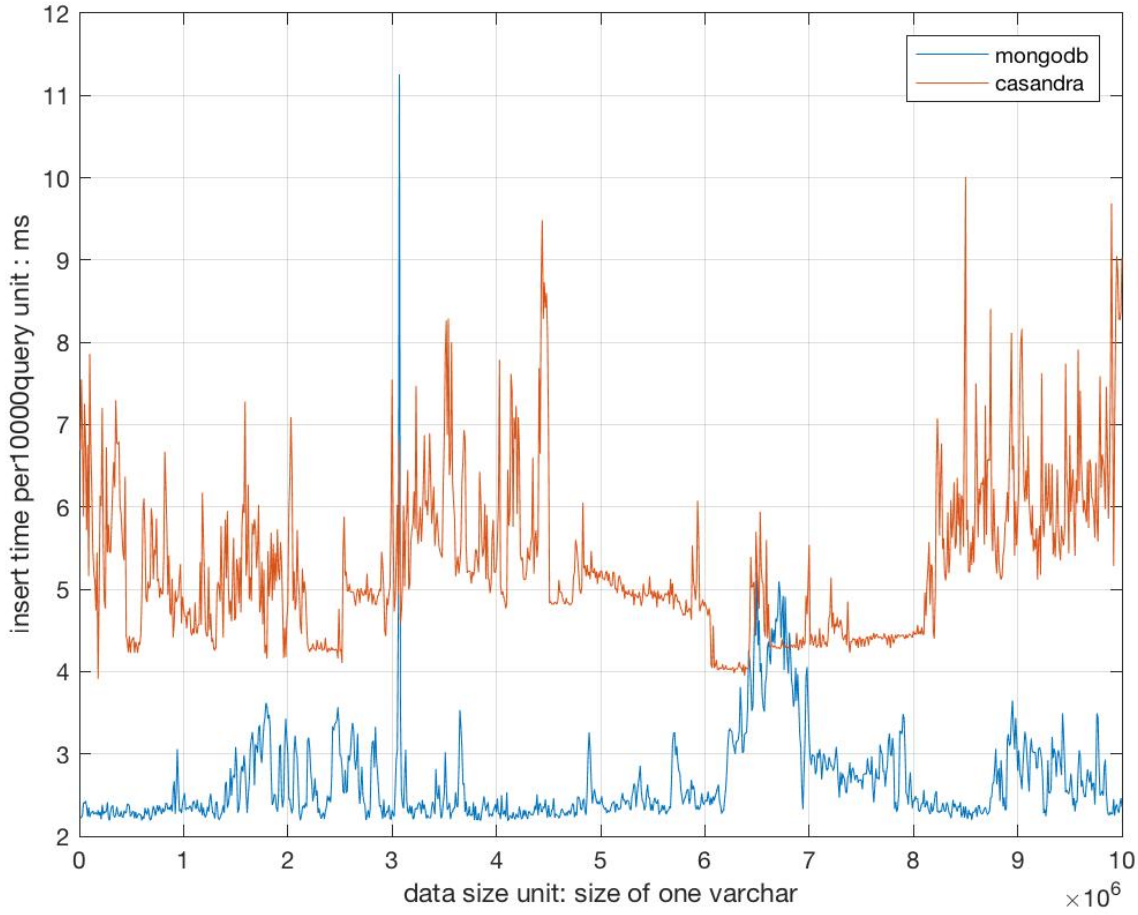

Figure 5: Insert 10000 data 1000 times on Macintosh (Scenario 3 compare with Scenario 21)

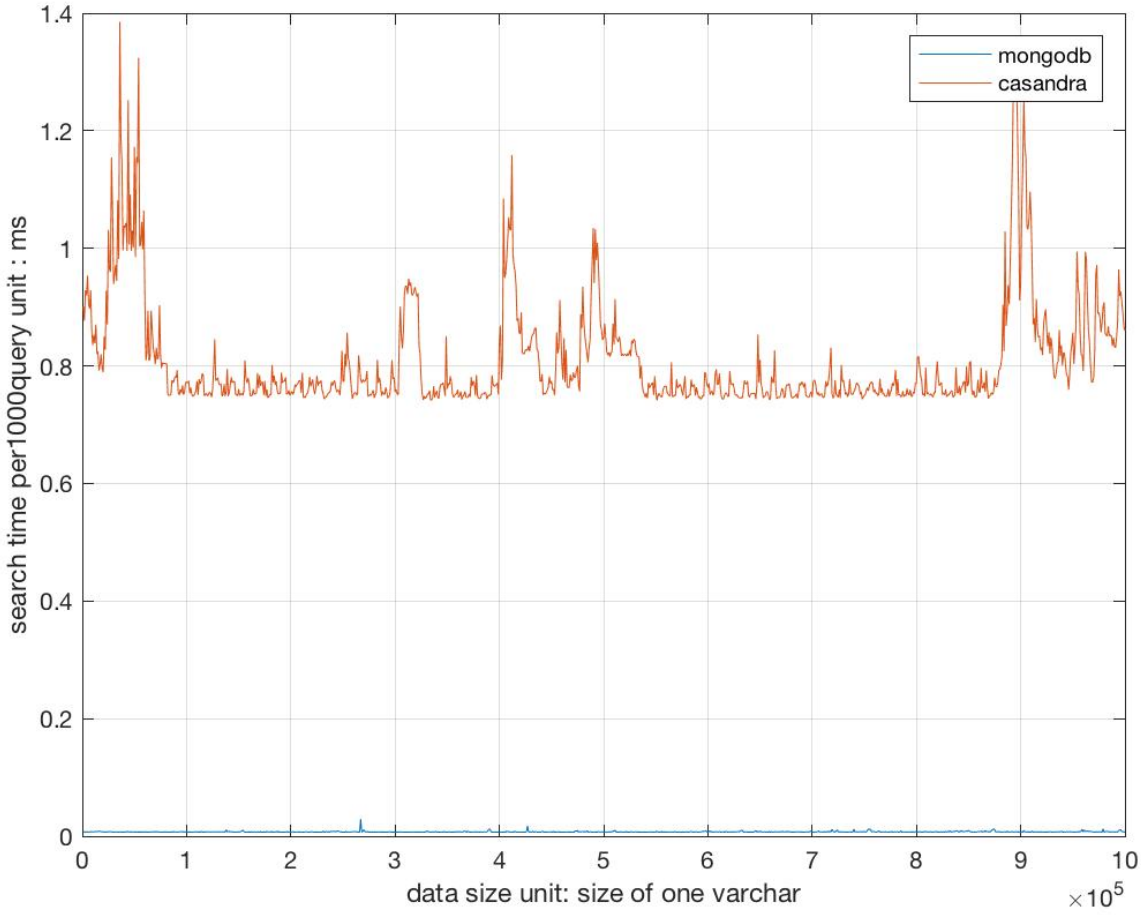

Figure 6: Serach 1000 data 1000 times on Macintosh (Scenario 4 compare with Scenario 22)

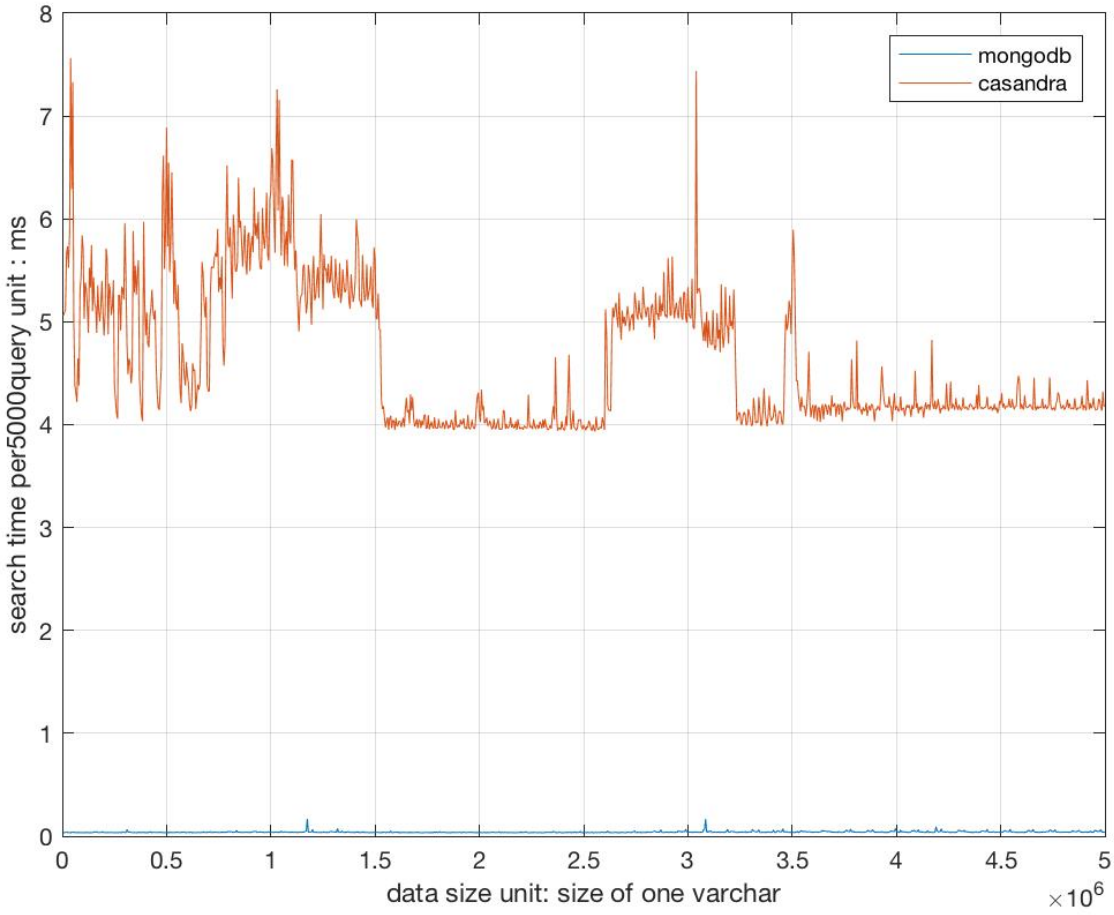

Figure 7: Serach 5000 data 1000 times on Macintosh (Scenario 5 compare with Scenario 23)

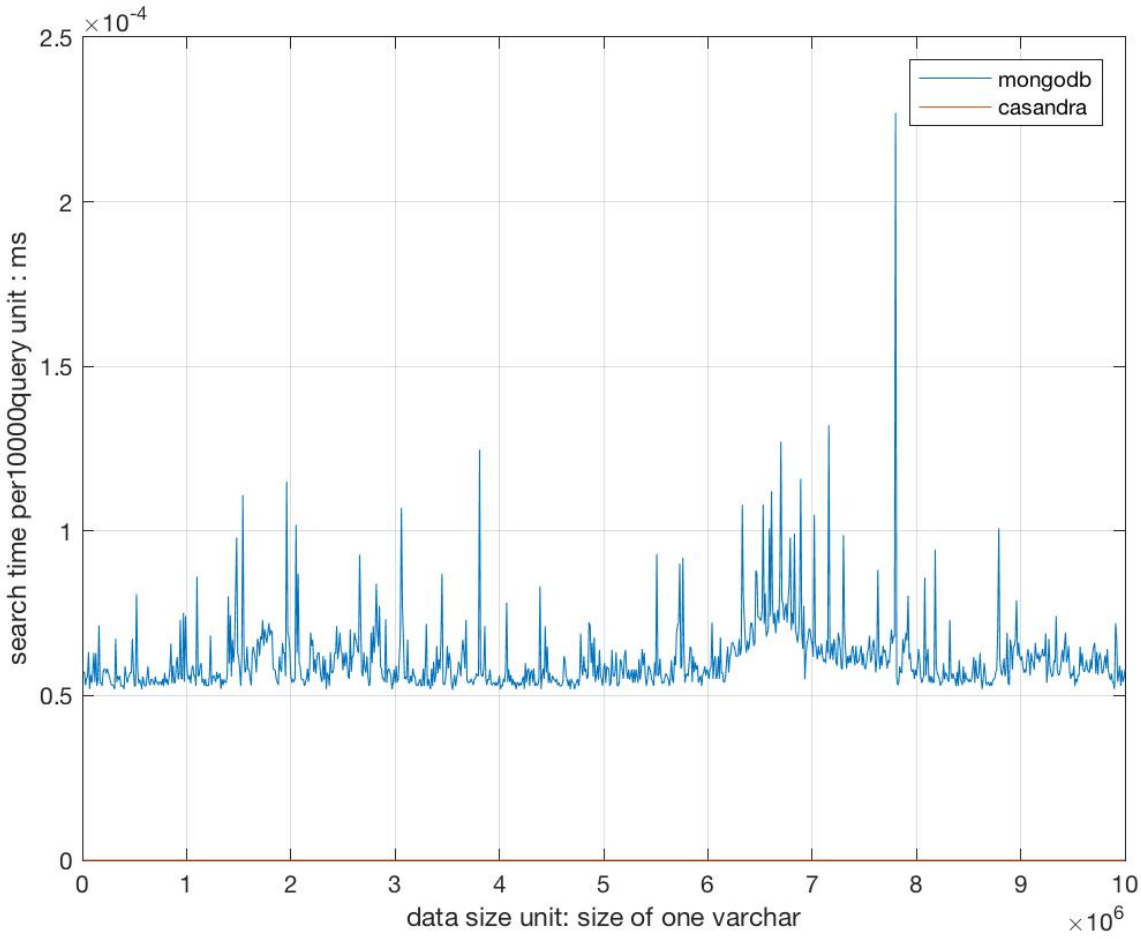

Figure 8: Serach 10000 data 1000 times on Macintosh (Scenario 6 compare with Scenario 24)

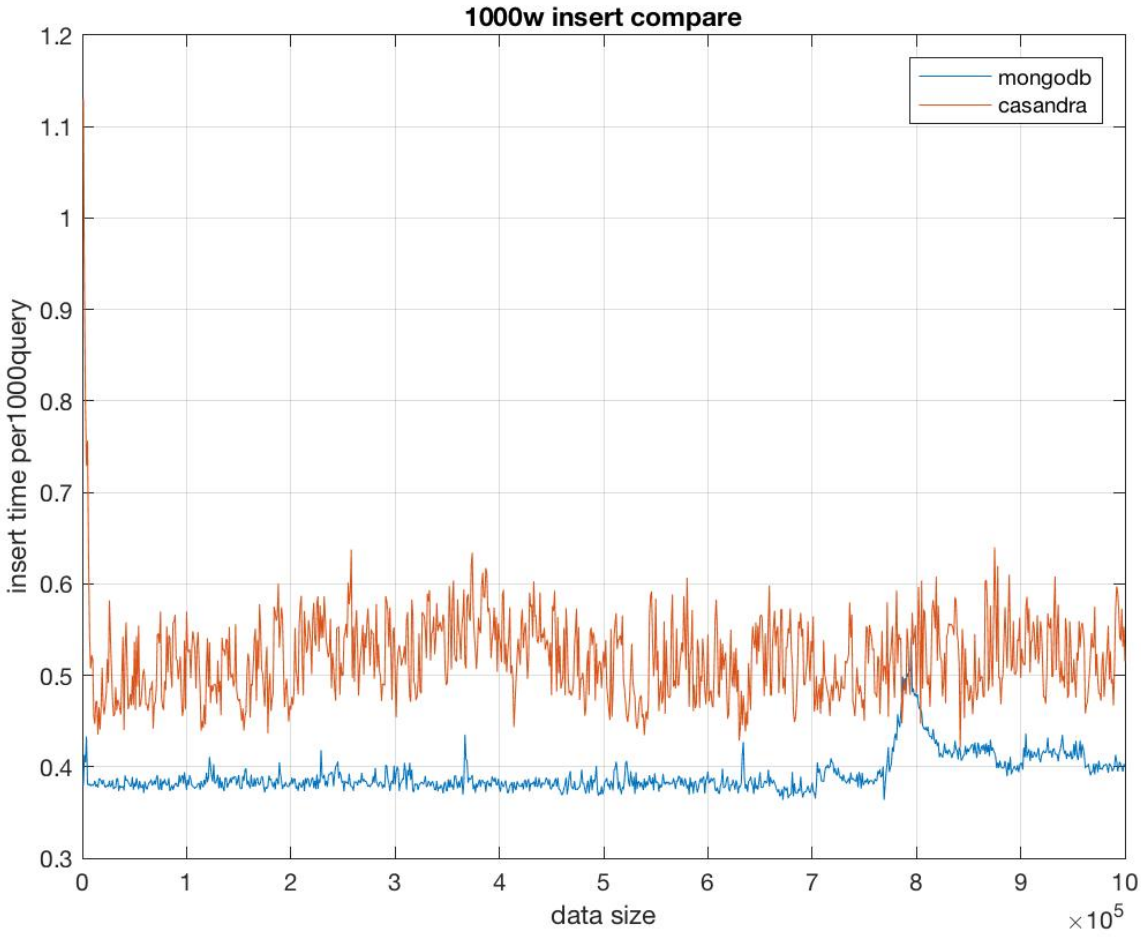

Figure 9: Insert 1000 data 1000 times on Raspberry Pi (Scenario 7 compare with Scenario 25)

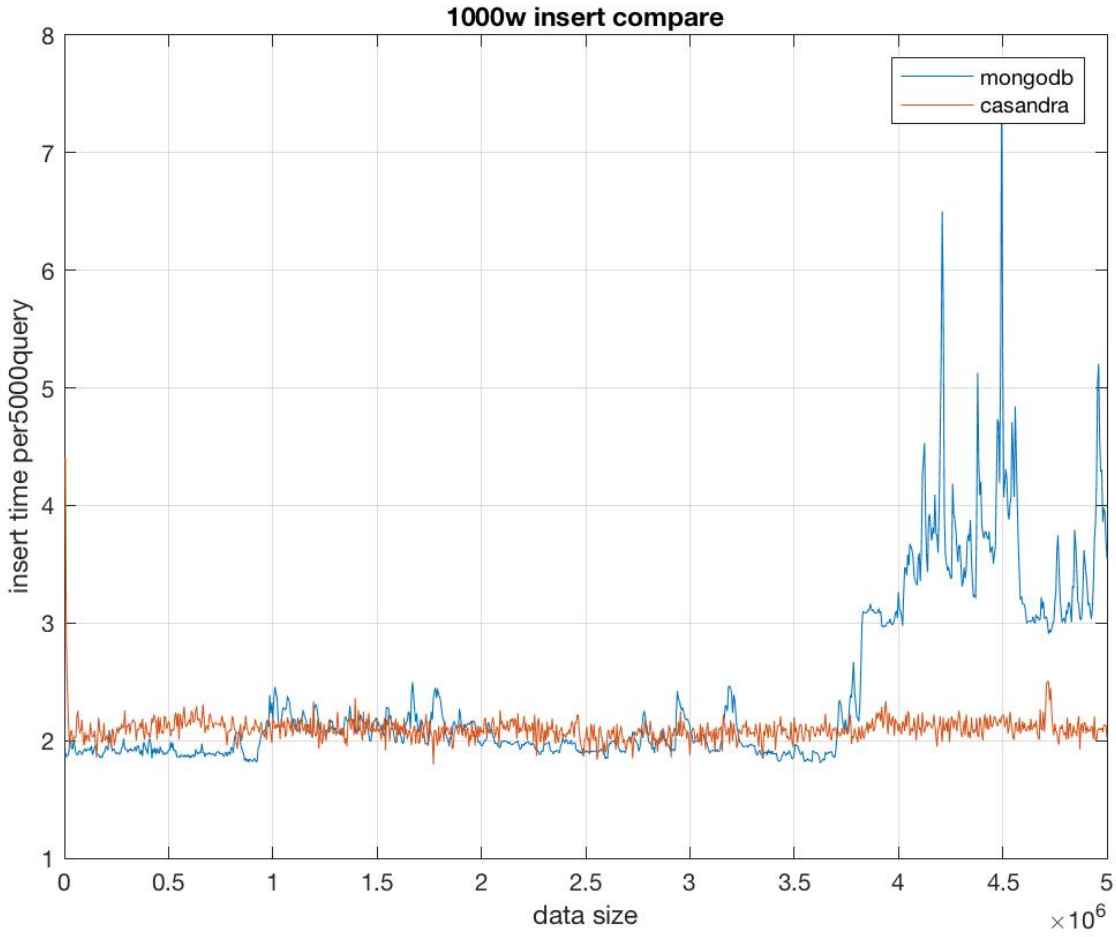

Figure 10: Insert 5000 data 1000 times on Raspberry Pi (Scenario 8 compare with Scenario 26)

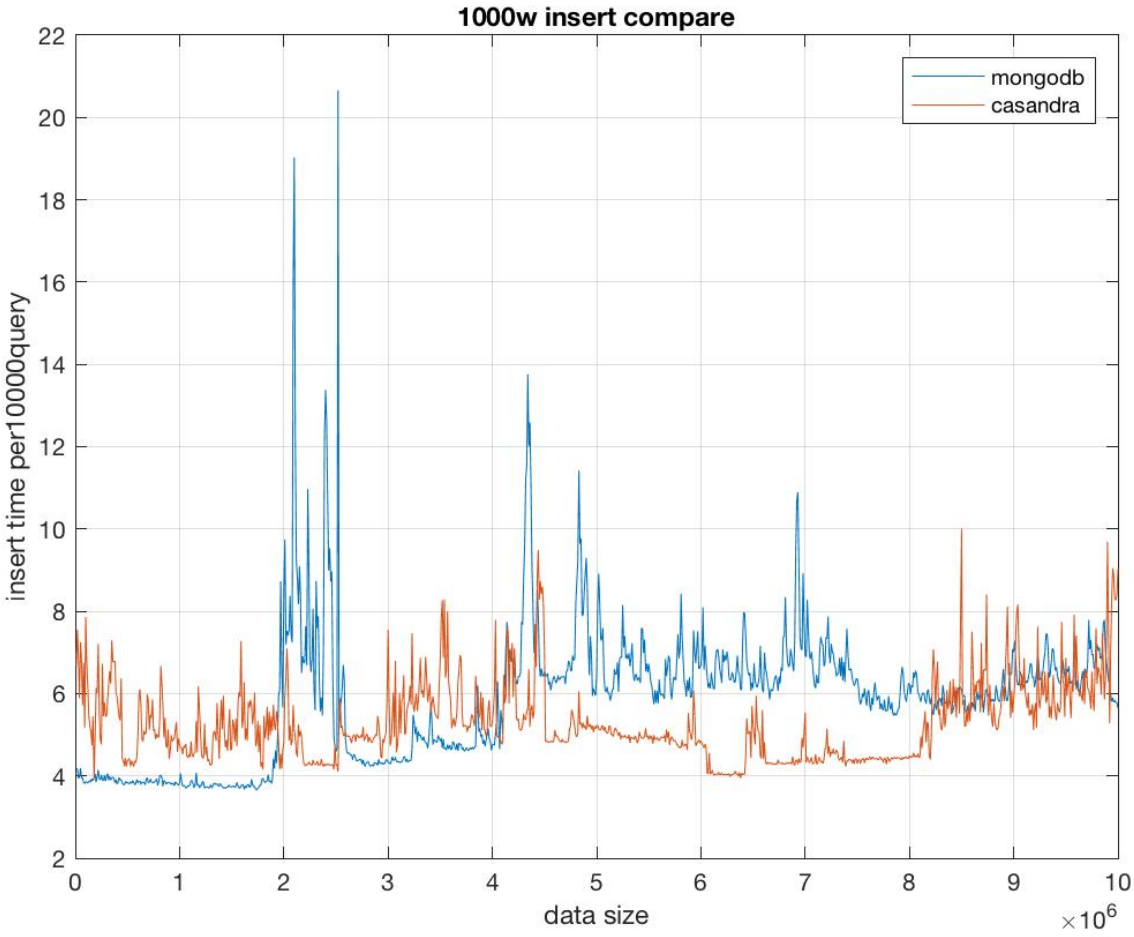

Figure 11: Insert 10000 data 1000 times on Raspberry Pi (Scenario 9 compare with Scenario 27)

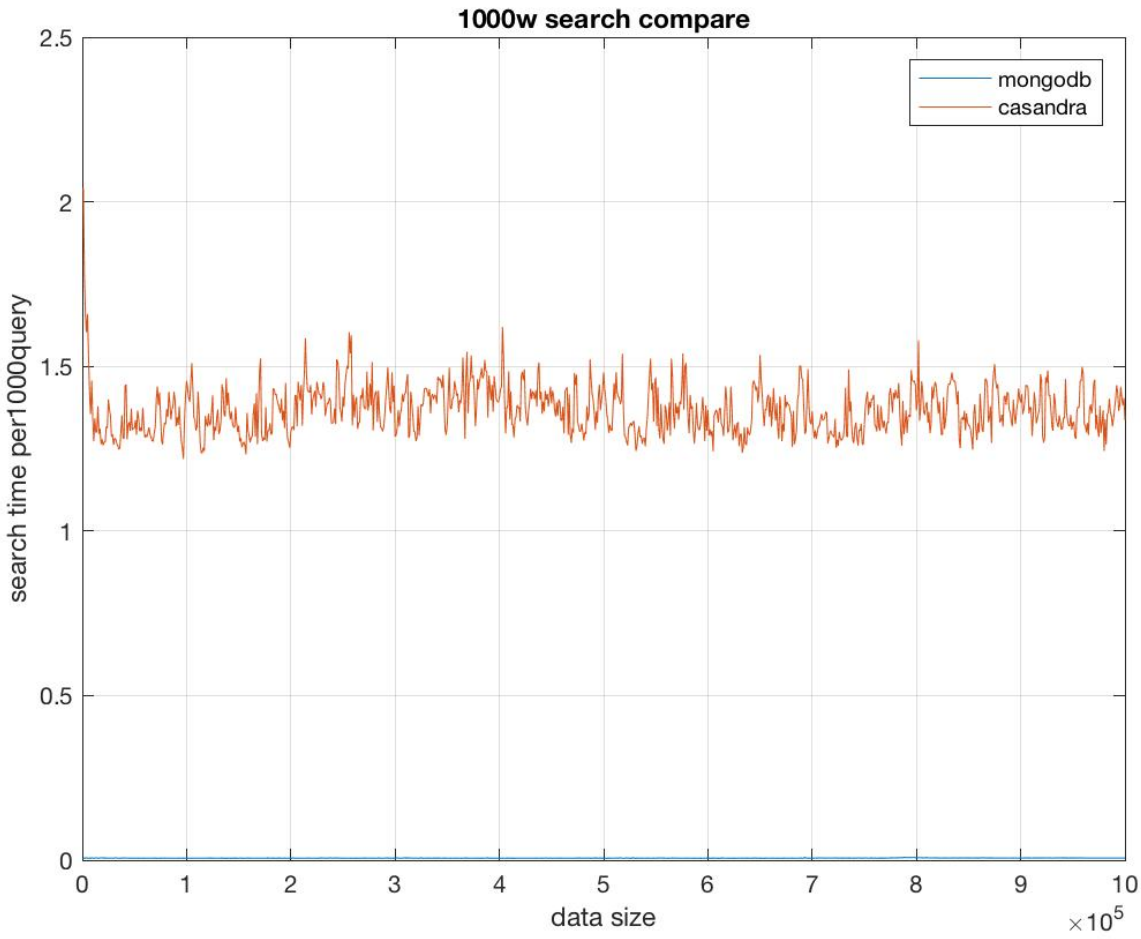

Figure 12: Serach 1000 data 1000 times on Raspberry Pi (Scenario 10 compare with Scenario 28)

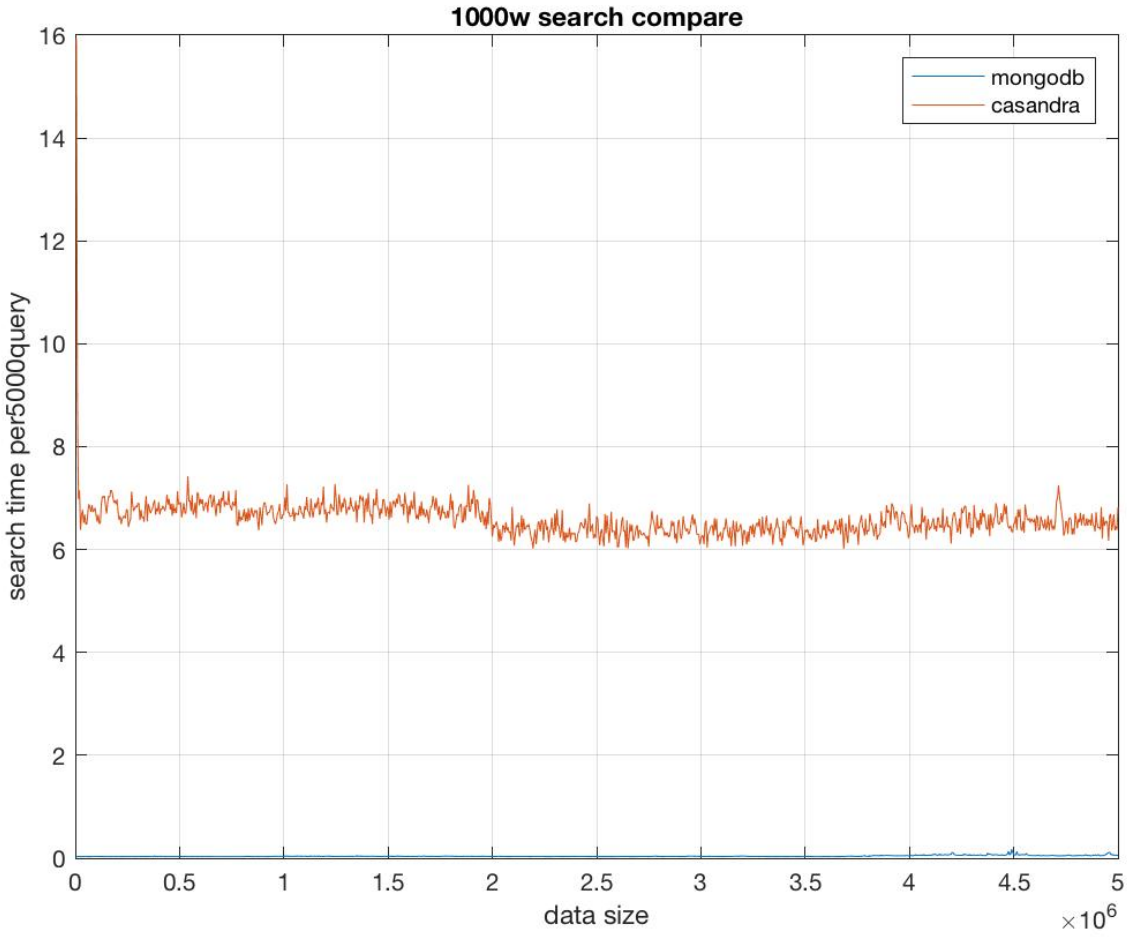

Figure 13: Search 5000 data 1000 times on Raspberry Pi (Scenario 11 compare with Scenario 29)

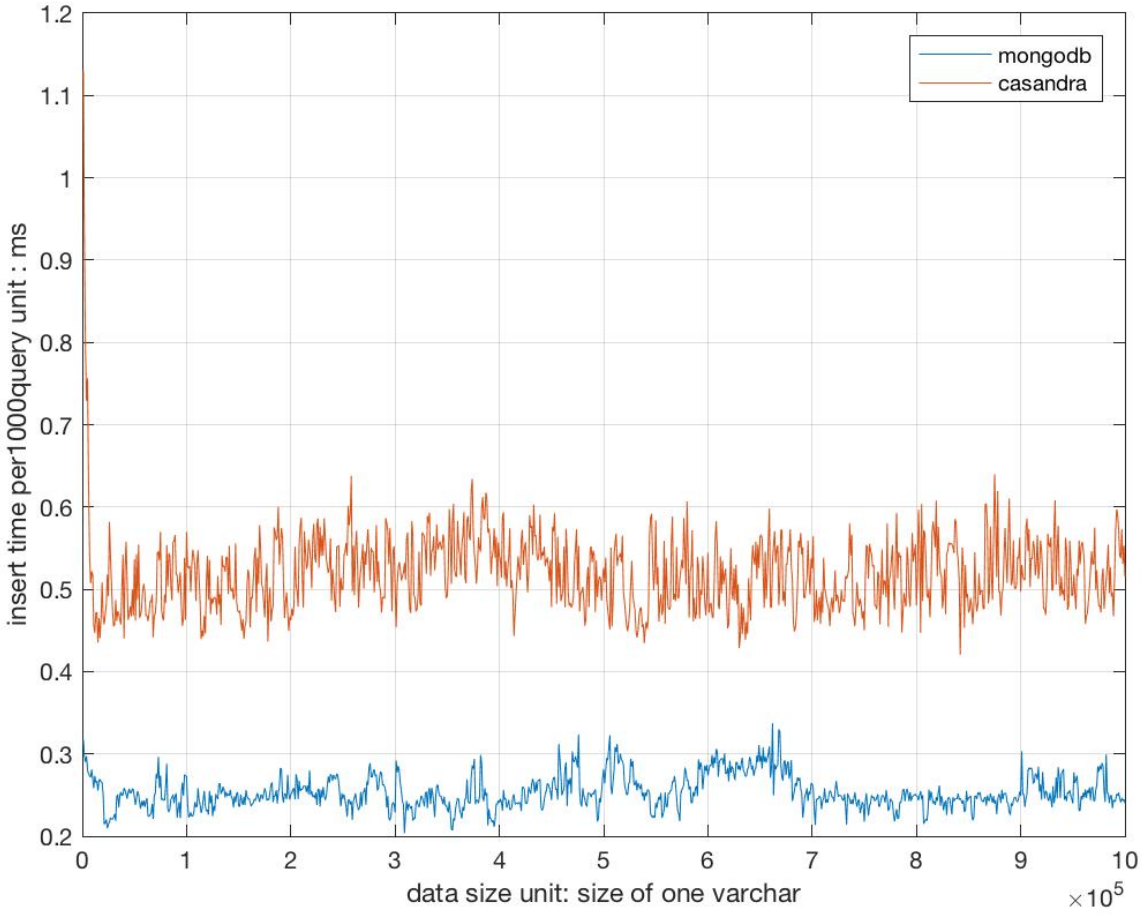

Figure 14: Insert 1000 data 1000 times on Google Cloud (Scenario 13 compare with Scenario 31)

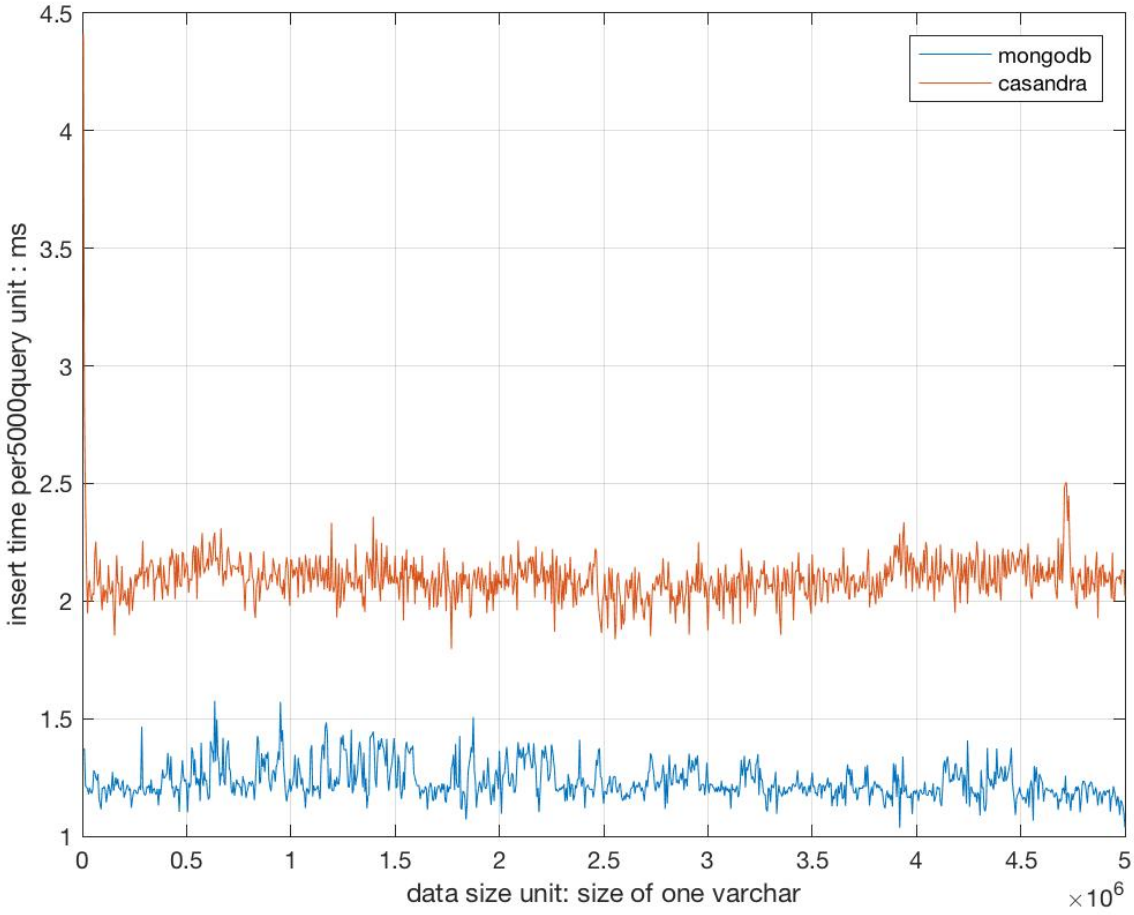

Figure 15: Insert 5000 data 1000 times on Google Cloud (Scenario 14 compare with Scenario 32)

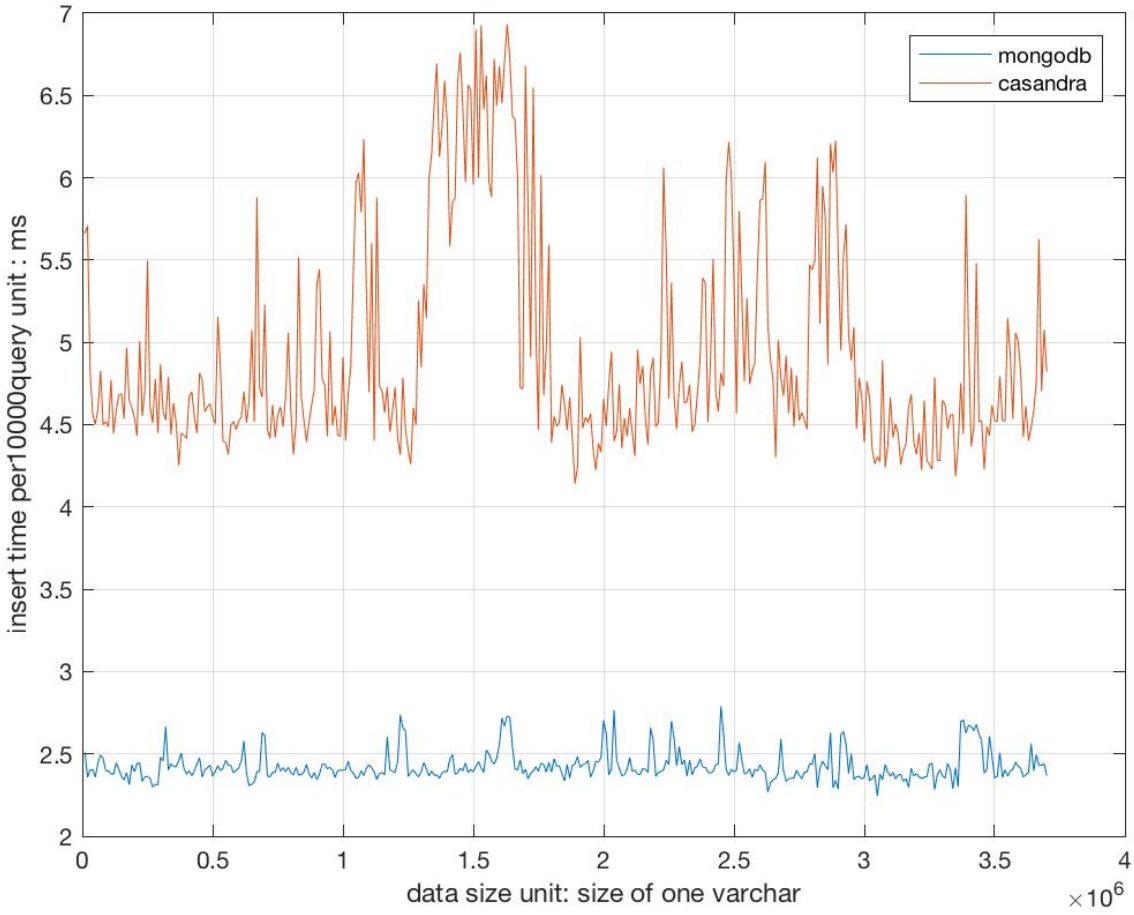

Figure 16: Insert 10000 data 1000 times on Google Cloud (Scenario 15 compare with Scenario 33)

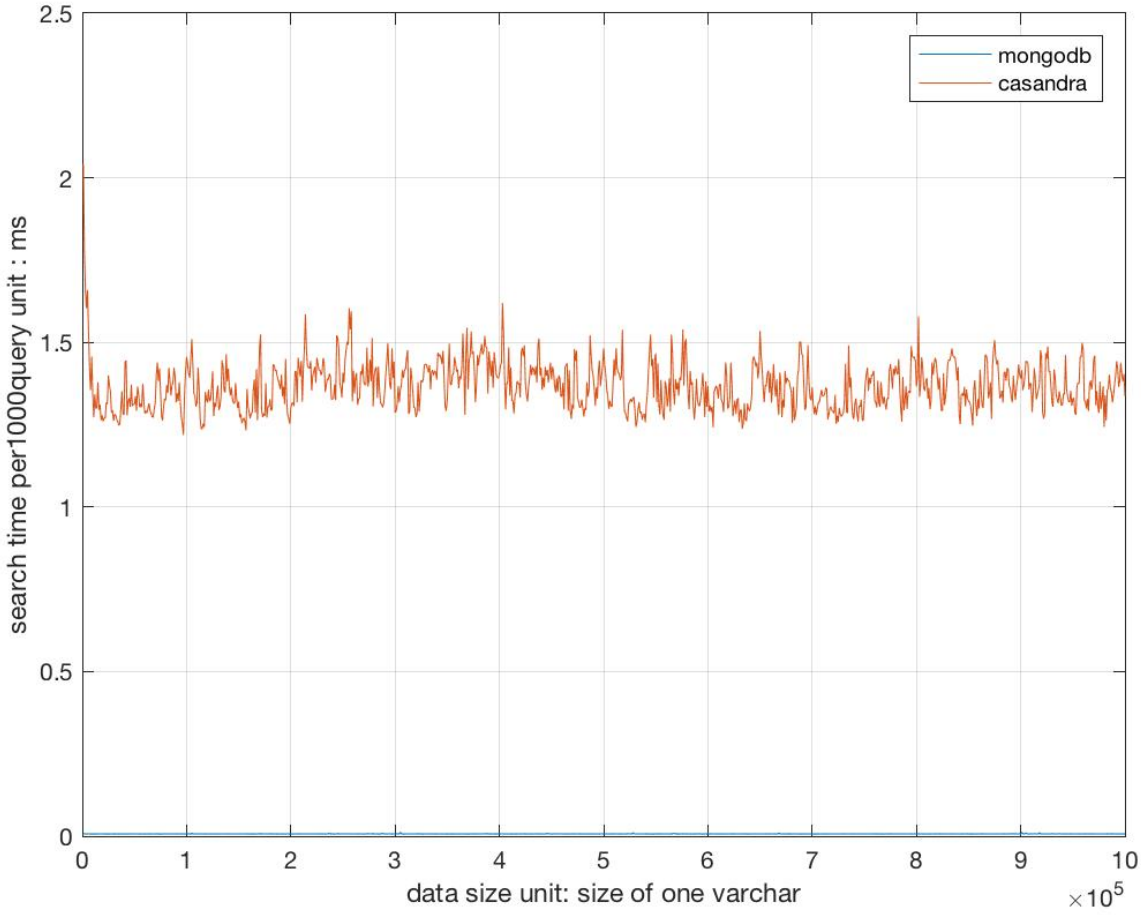

Figure 17: Search 1000 data 1000 times on Google Cloud (Scenario 16 compare with Scenario 34)

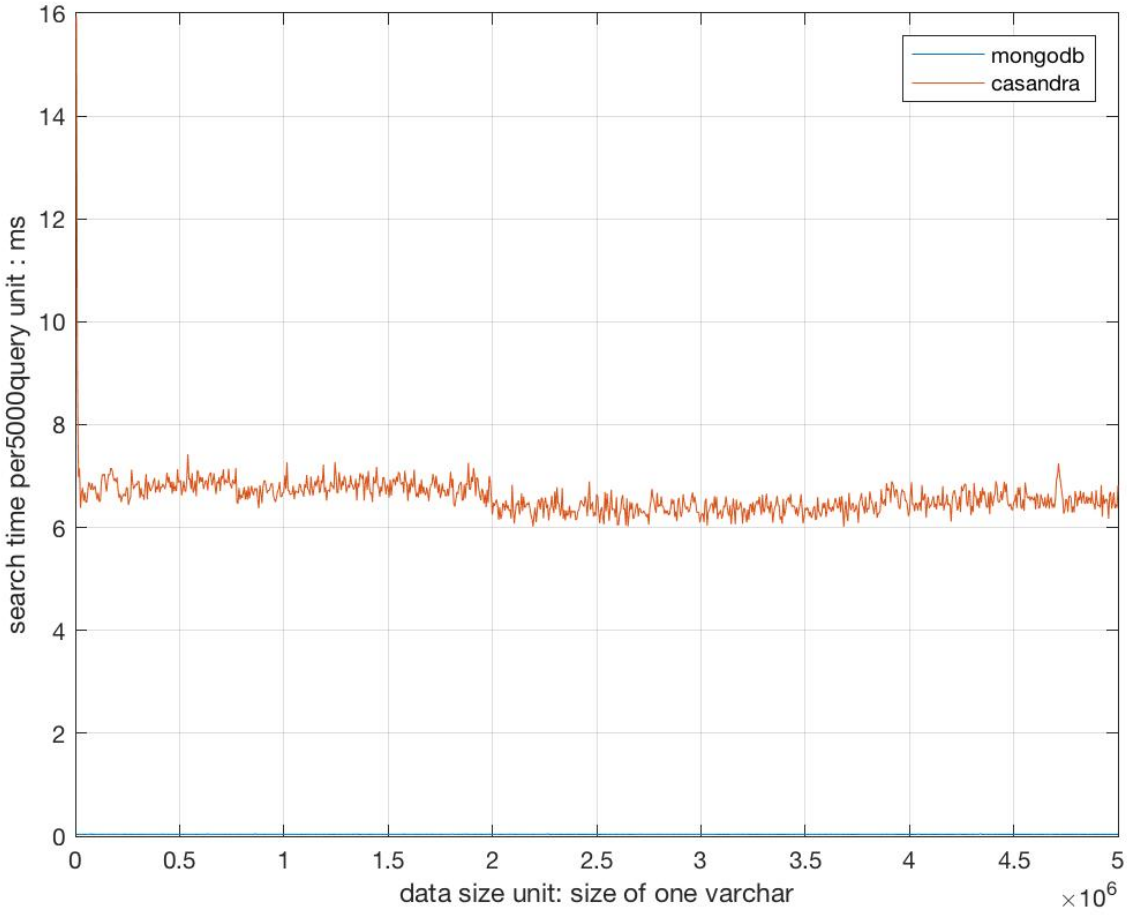

Figure 18: Search 5000 data 1000 times on Google Cloud (Scenario 17 compare with Scenario 34)

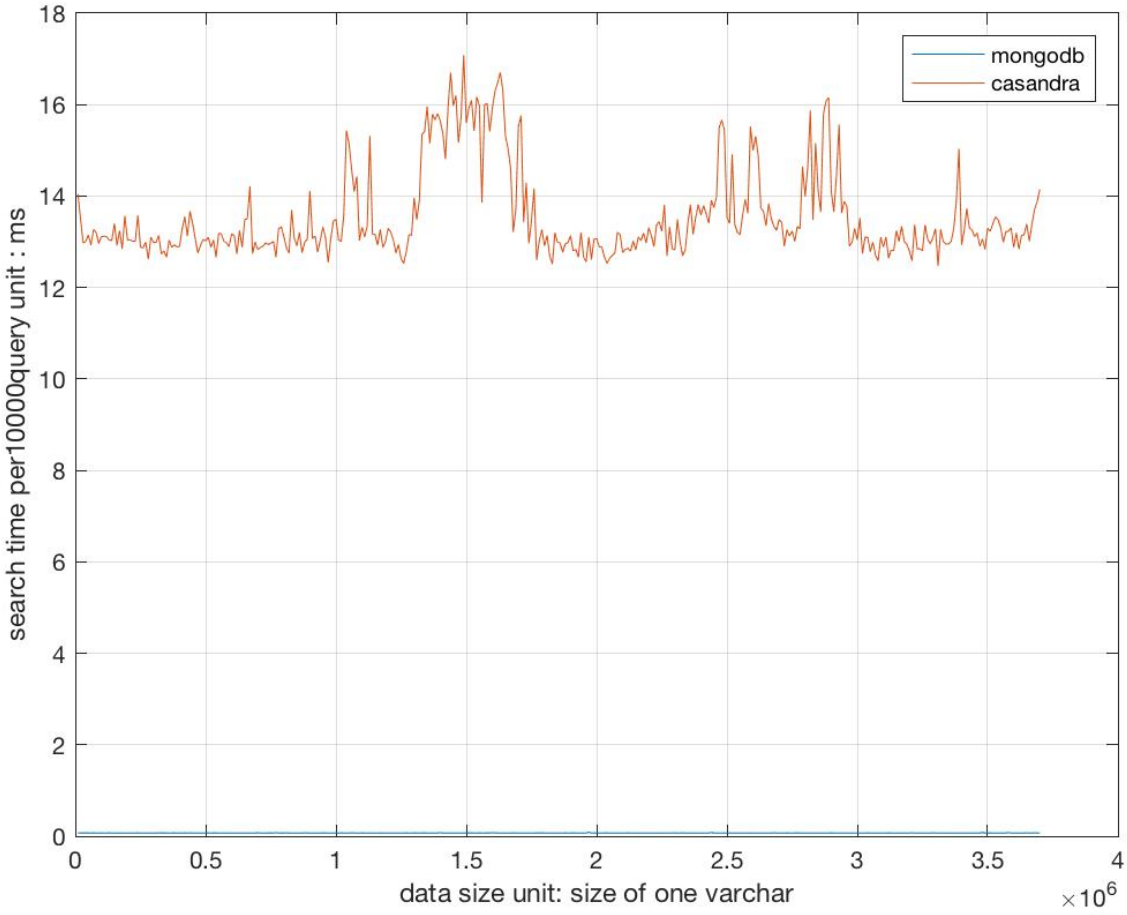

Figure 19: Search 10000 data 1000 times on Google Cloud (Scenario 18 compare with Scenario 35)

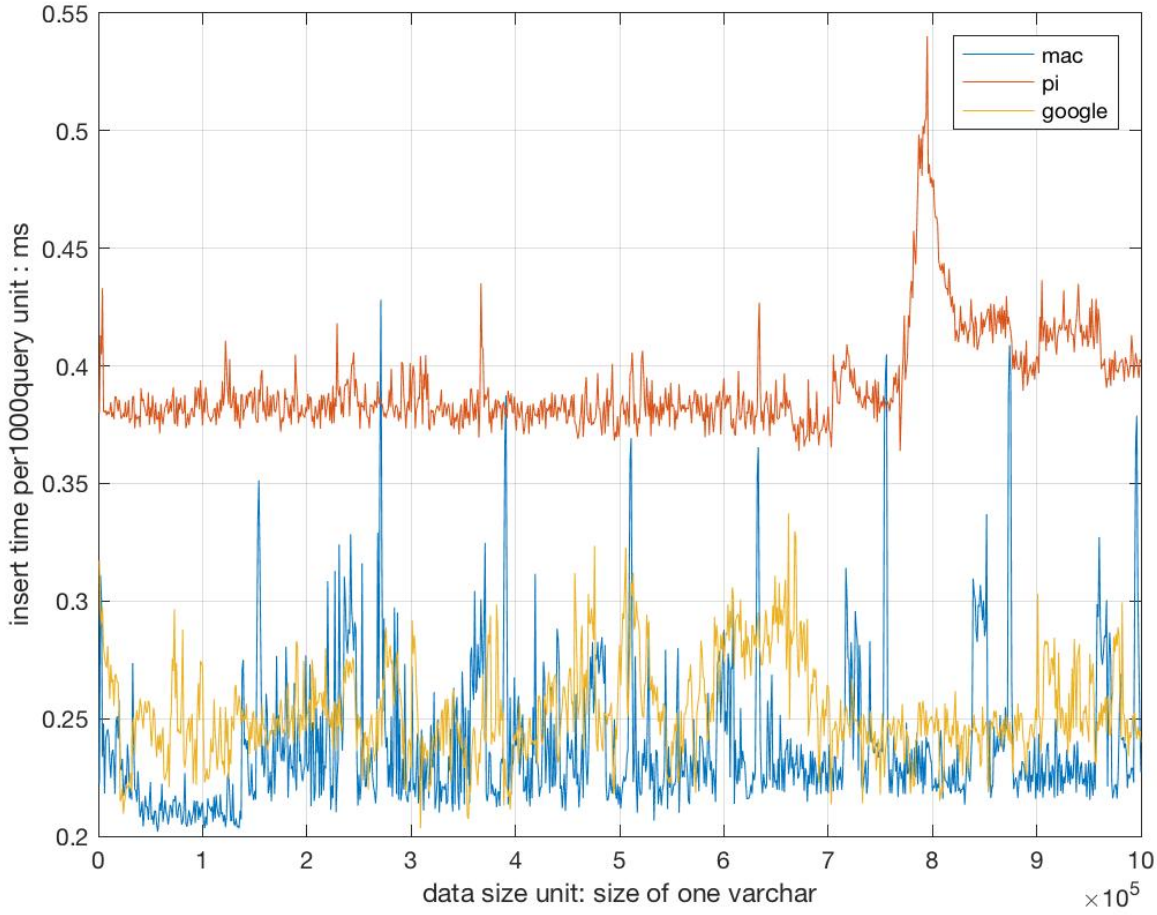

Figure 20: Insert 1000 data 1000 times on Mac, Raspberry Pi and Google Cloud (Scenario 1, 7 compare with Scenario 13)

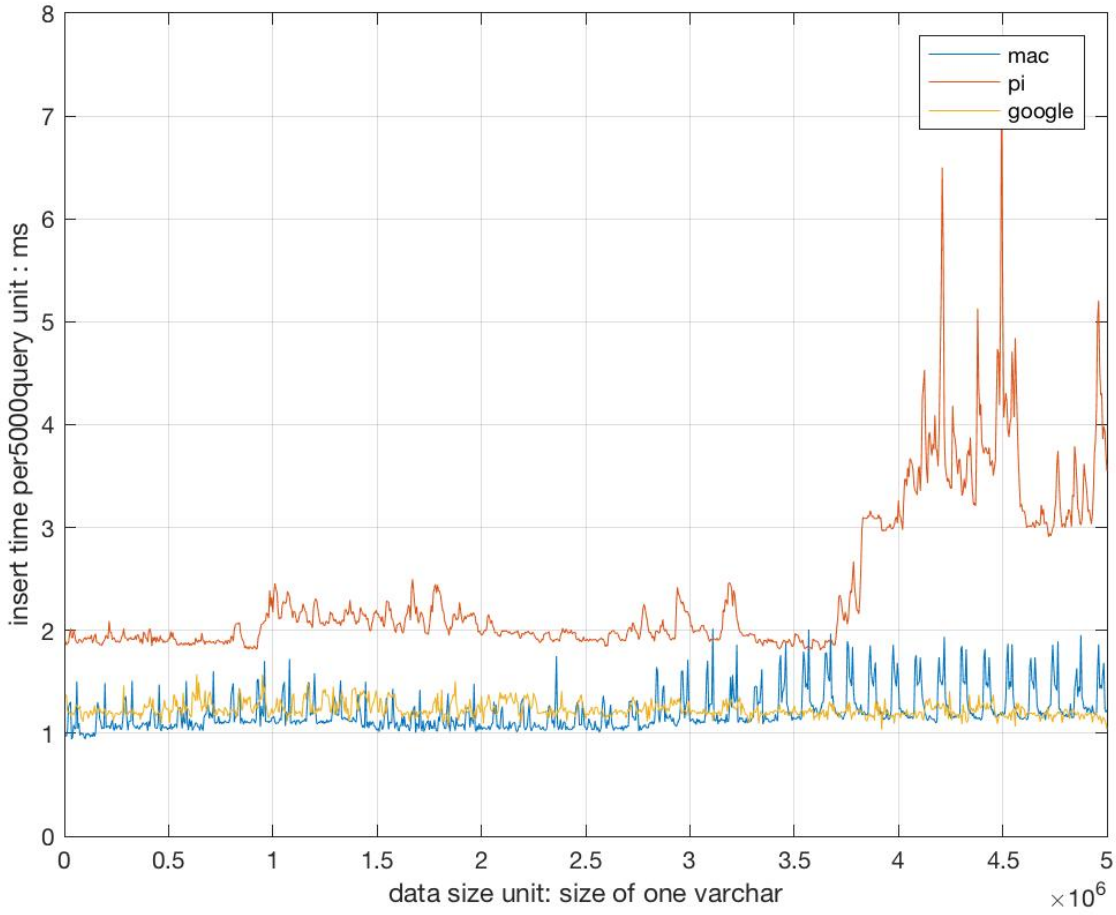

Figure 21: Insert 5000 data 1000 times on Mac, Raspberry Pi and Google Cloud (Scenario 2, 8 compare with Scenario 14)

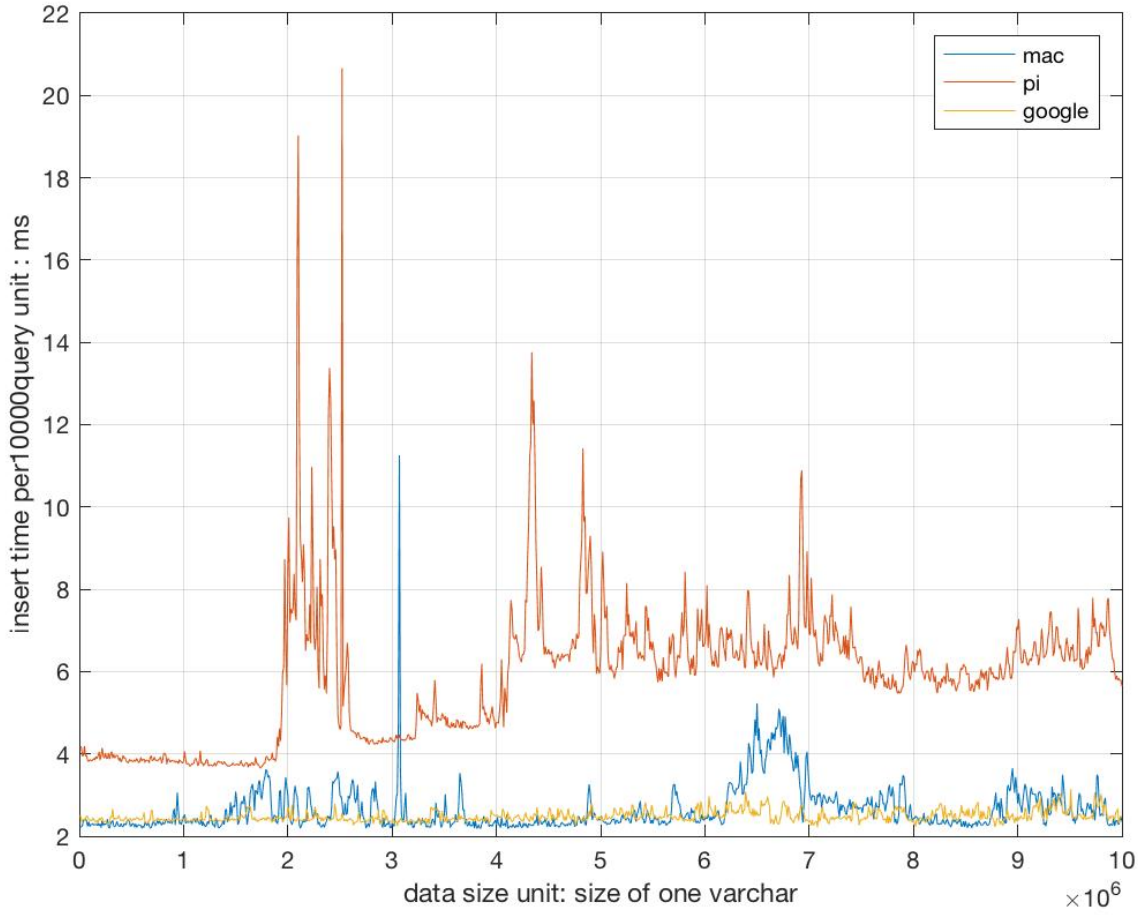

Figure 22: Insert 10000 data 1000 times on Mac, Raspberry Pi and Google Cloud (Scenario 3, 9 compare with Scenario 15)