



An Empirical Evaluation of Distributed Key/Value Storage Systems

Patel Harsh - A20392023

Patel Rahul - A20370701

Introduction

distributed key/value storage systems are the computer network where information has been stored on more than one node. These are normally non relational database hence make a quick access of the data on large number of nodes. A key-value store is the database that uses an associative array where each key is connected with one and only one value so it's called as key-value pair. In key/value storage. These system aims to address the issue of very high concurrent data volume by the use of distribution and simplified data models.

Facebook, Amazon and LiveJournal are the domain in development of the new set of databases. In 2008, Facebook developed a decentralized storage system named Cassandra[1] to deal with constantly increasing data throughput. In short for high volumes of concurrent queries, distributed database is required.

Background Information

Distributed storage system should be transparent and shown as a single coherent system to its users. The design of such a system can be difficult and there are few reason like network going down or nodes working erroneously. The pros of a distributed storage system is its wide availability, reliability and strong performance. Centralized and decentralized are the type of distributed system. In the former case, one node is being allocated by special responsibilities so it acts as a coordinator for all other nodes. This one is simple to implement but it has lower reliability due to having one single point of failure. key/value store is a database with a simple data model similar to hash table. This type of database is less complicated to distribute than one with fully relational model .

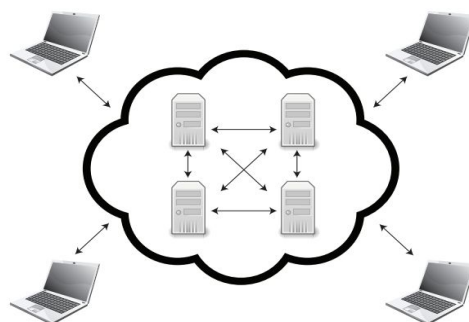


Figure 1. Distributed system

Problem statement

Distributed key/value storage is nested network so when we send request to server, it will take time to get response from server means latency will be higher. When we move data from one place to another in the given time period, all data is not move successfully so these storage system has some throughput. We don't have comparison of latency and throughput for different operation .

MongoDB and Features

MongoDB is a standout amongst the most mainstream document stores.

- MongoDB, Inc is its engineer.
- Initial adaptation was discharged in 2009 and it is open source.
- It is actualized in c++.
- Linux, OS X, Solaris, Windows are the server working frameworks for it.
- MongoDB upheld many programming dialects.
- JavaScript is the server-side content for MongoDB.
- It bolsters MapReduce.
- It utilizes Master-slave replication.
- It has Eventual and Immediate consistency.
- Specific attributes:-

MongoDB is the cutting edge database that enables organizations to change their industries by tackling the energy of information. The world's most modern organizations, from forefront new businesses to the biggest organizations, utilize MongoDB to create applications at no other time conceivable, at a small amount of the cost of inheritance databases. MongoDB was likewise named a pioneer in the Forrester Wave™: Big Data NoSQL, Q3 2016.

MongoDB is the quickest developing database biological system, with more than 20 million downloads, thousands of clients, and more than 1,000 innovation and administration accomplices.

- Competitive points of interest

By offering the best of customary databases and also the adaptability, scale, and performance required by the present applications, MongoDB gives trend-setters a chance to convey applications as big as they can dream. From new businesses to endeavors, for the cutting edge and the mission-basic, MongoDB is the database for mammoth thoughts.

MongoDB keeps up the most important highlights of social databases: solid consistency, expressive question dialect and auxiliary records. Subsequently, engineers can assemble exceptionally useful applications quicker than NoSQL databases.

MongoDB gives the information demonstrate adaptability, versatile adaptability, and superior and accessibility of NoSQL databases. Subsequently, architects can consistently improve applications, and convey them at practically boundless scale on item equipment.

- Typical application situations

Web of Things (Bosch, Silver Spring Networks)

Portable (The Weather Channel, ADP, O2)

Single View (MetLife)

Continuous Analytics (Buzzfeed, City of Chicago, Crittercism)

Personalization (Expedia, eHarmony, Gilt)

Inventories (Under Armor, Otto)

Content Management (eBay, Forbes)

- Market measurements

20 million downloads (developing at thousands downloads every day).

2,000+ clients including more than 33% of the Fortune 100.

Named a pioneer in the Forrester Wave™: Big Data NoSQL, Q3 2016.

Most astounding put non-social database in DB Engines rankings

- Document approval

- Journaling

- Powerful accumulation structure

- On 32bit frameworks, restricted to ~2.5Gb

- Text seek coordinated

- GridFS to store enormous information + metadata (not really a FS)

- Has geospatial ordering

- Data focus mindful

—>Best used: If you need dynamic queries. If you prefer to define indexes, not map/reduce functions. If you need good performance on a big DB. If you wanted CouchDB, but your data changes too much, filling up disks.

MongoDB's Pros

- Schema-less:- If you have a flexible schema, this is ideal for a document store like MongoDB. This is difficult to implement in a performant manner in RDBMS
- Ease of scale-out:- Scale reads by using replica sets. Scale writes by using sharding (auto balancing). Just fire up another machine and away you go. Adding more machines = adding more RAM over which to distribute your working set.
- Cost:- Depends on which RDBMS of course, but MongoDB is free and can run on Linux, ideal for running on cheaper commodity kit.
- you can choose what level of consistency you want depending on the value of the data (e.g. faster performance = fire and forget inserts to MongoDB, slower performance = wait til insert has been replicated to multiple nodes before returning)

MongoDB's Cons

- Data size in MongoDB is typically higher due to e.g. each document has field names stored it
- Less flexibility with querying (e.g. no JOINS)
- No support for transactions - certain atomic operations are supported, at a single document level
- At the moment Map/Reduce (e.g. to do aggregations/data analysis) is OK, but not blisteringly fast. So if that's required, something like Hadoop may need to be added into the mix
- Less up to date information available/fast evolving product.

Setup Manual

-->Type following command in terminal to install MongoDB server

```
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv EA312927
echo "deb http://repo.mongodb.org/apt/ubuntu "$(lsb_release -sc)"/mongodb-org/3.2
multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.2.list
sudo apt-get update
sudo apt-get install -y mongodb-org
cd /etc
sudo vi mongod.conf
```

command the bindIP & save it

-->For start the server just give this command

`sudo service mongod start`

-->For check the status of server,use it

`sudo service mongod status`

-->Use below command to run MongoDB client on multiple machines.

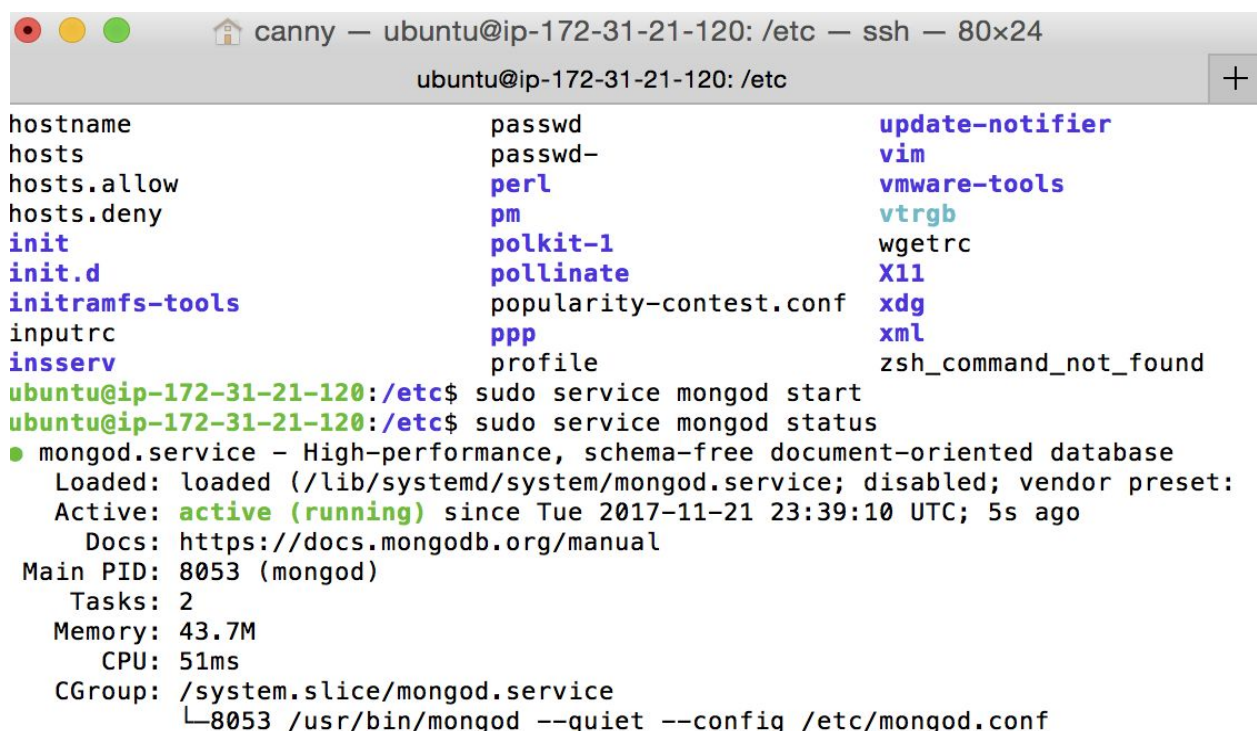
`Javac Mongo.java`

`Java Mongo`

Design of MongoDB

-We are using aws instances (t2.micro) for the benchmark of ALL-to-ALL communication.

-When we start server



```

canny — ubuntu@ip-172-31-21-120: /etc — ssh — 80x24
ubuntu@ip-172-31-21-120: /etc
hostname          passwd            update-notifier
hosts             passwd-          vim
hosts.allow       perl             vmware-tools
hosts.deny        pm              vtrgb
init              polkit-1         wgetrc
init.d            pollinate        X11
initramfs-tools   popularity-contest.conf xdg
inputrc           ppp             xml
insserv           profile         zsh_command_not_found
ubuntu@ip-172-31-21-120:/etc$ sudo service mongod start
ubuntu@ip-172-31-21-120:/etc$ sudo service mongod status
● mongod.service - High-performance, schema-free document-oriented database
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset:
   Active: active (running) since Tue 2017-11-21 23:39:10 UTC; 5s ago
     Docs: https://docs.mongodb.org/manual
   Main PID: 8053 (mongod)
    Tasks: 2
   Memory: 43.7M
      CPU: 51ms
   CGroup: /system.slice/mongod.service
           └─8053 /usr/bin/mongod --quiet --config /etc/mongod.conf
  
```

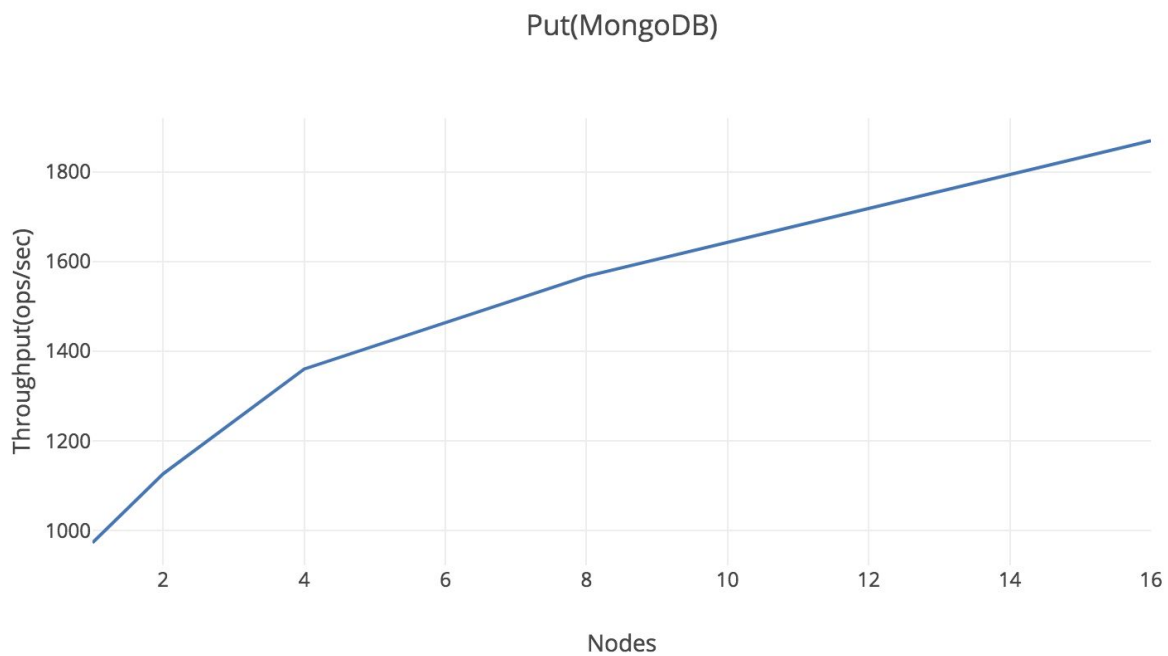
-When we run client code,it gives following output.

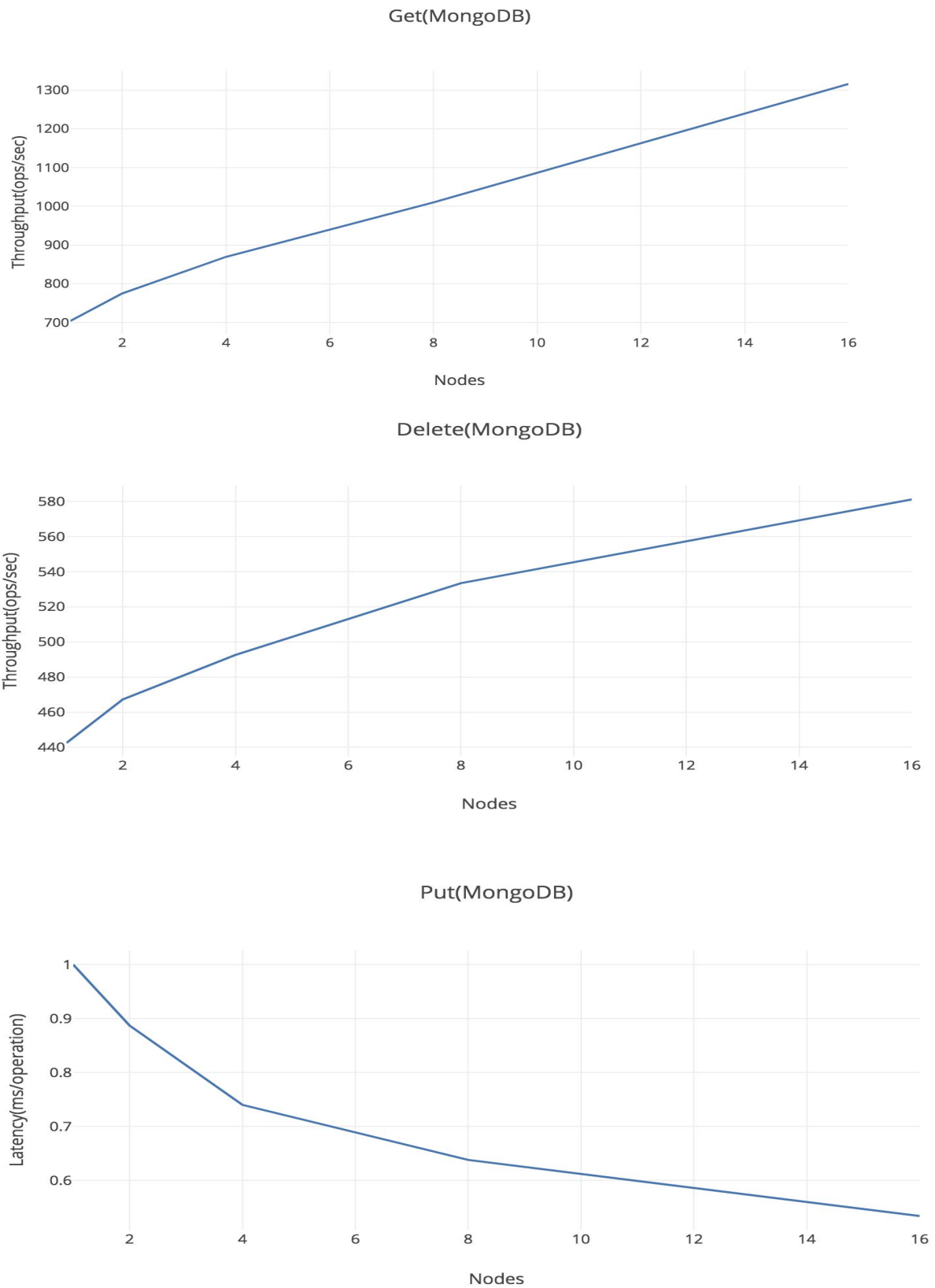
```
ubuntu@ip-172-31-21-120: /etc/redis  ubuntu@ip-172-31-21-120: ~ +
ubuntu@ip-172-31-21-120:~$ java MongoDB
Connected with MongoDB server successfully...
100K PUT operations.
100K PUT operations completed in 100.726 seconds.
100K GET operations.
100K GET operations completed in 142.45 seconds.
100K DEL operations.
100K DEL operations completed in 226.7 seconds.
```

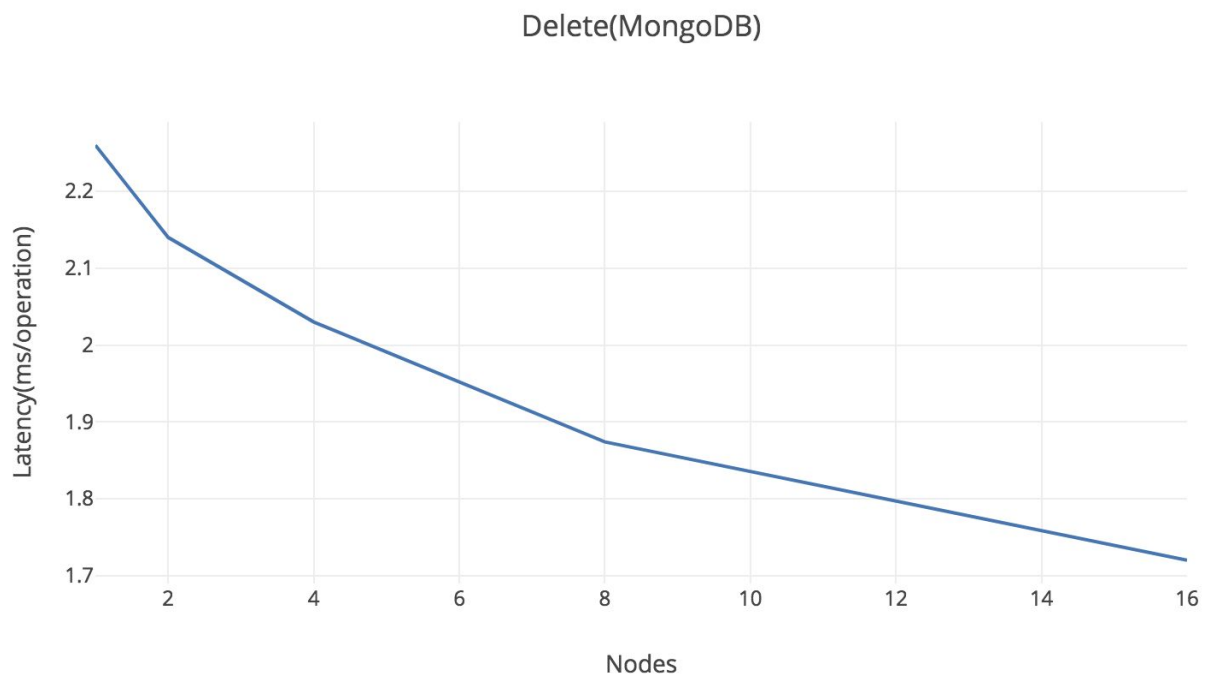
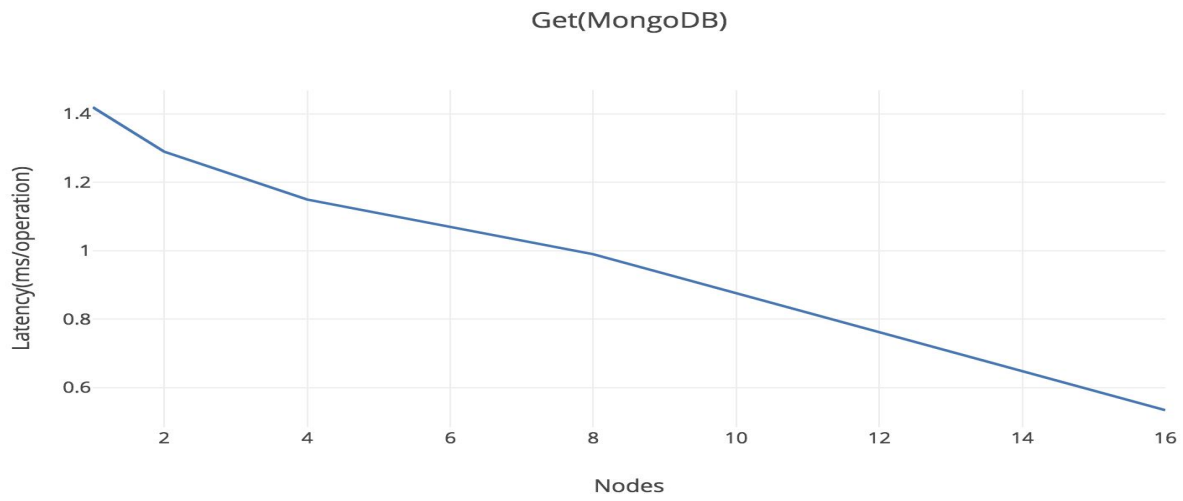
-For all to all communication,we do the same process on different instances.

-I uploaded the client side source code for 100k iteration.

Throughput and Latency of MongoDB







Redis and Features

Redis is in-memory information structure store, utilized as database, reserve and message agent.

- It is key-value store.
- Salvatore Sanfilippo is its developer.
- Initial rendition was discharged in 2009 and it is open source.
- It is executed in c.
- Linux, OS X, BSD, Windows are the server working frameworks for it.
- Redis bolstered many programming dialects.
- Lua is the server-side content for Redis.
- It doesn't bolster MapReduce.
- It utilize Master-slave replication.
- It has Eventual consistency.
- Specific attributes:-

Redis, benchmarked as the world's speediest database, decreases application many-sided quality, simplifies advancement, quickens time to showcase and gives extraordinary flexibility to designers with its visionary information structures and modules.

Redis Labs is the home of open source Redis and business supplier of Redis Enterprise, the in-memory database stage for value-based, investigation and mixture deployments. Redis Enterprise is accessible, both as an administration in significant open, private and hybrid mists, and also downloadable programming.

- Competitive favorable circumstances

Redis is an in-memory database stage offering local help of extensive variety of information structures, for example, strings, hashes, sets, records, arranged sets, bitmaps, hyperlog logs, and geospatial lists. Redis Modules broaden the capacities of Redis changing it into a multi-show database. With Redis Modules, one can utilize Redis as a web index, time arrangement database, chart database, JSON store, rate limiter, and that's just the beginning.

Redis Enterprise expands the energy of Redis with an improved arrangement architecture that incorporates a zero dormancy disseminated intermediary, a bunch supervisor, and a shared-nothing group design. It offers:

Effortless scaling in a completely mechanized way by administering every one of the operations of sharding, re-sharding, movement

A high accessibility suite including perseverance, moment programmed disappointment discovery, reinforcement and recuperation, and in-memory replication crosswise over racks, zones, datacenters, areas, and cloud stages.

Substantial cost funds with worked in multi-tenure, more noteworthy productivity in equipment utilization, bolster for streak as a RAM augmentation

ACID controls for exchanges in a bunched situation

- Typical application situations

Redis powers personalization, continuous extortion recognition, IoT, internet business, constant metering, and social applications among numerous arrangements. It's utilization cases go from caching, rapid exchanges, time-arrangement logs, message dealer, continuous examination, data ingestion, to employment and line administration, and that's just the beginning.

- Market measurements

Redis Labs serves more than 60,000 clients all around and is reliably positioned as a leader in top investigator writes about NoSQL, in-memory and operational databases. Redis has been evaluated the #1 cloud database, #1 database in Docker, #1 NoSQL datastore, most well known NoSQL database in holders, and InfoWorld's 2017 Technology of the Year.

- Has sets (likewise union/diff/entomb)
- Has records (likewise a line; blocking pop)
- Has hashes (objects of different fields)
- Sorted sets (high score table, useful for go questions)
- Lua scripting abilities
- Values can be set to terminate (as in a reserve)
- Pub/Sub gives you a chance to actualize informing
- GEO API to question by range (!)

—>Best used: For rapidly changing data with a foreseeable database size (should fit mostly in memory).

Redis's Pros

- It's blazing fast
- It supports a wide variety of data types
- It's open source and has an active community.
- It's simple to install and has no dependencies
- Stores generic data types for any purpose
- Easy to get started on a single cheap/free server

Redis's Cons

- Your dataset has to fit comfortably in memory
- No joins or query language
- You have to learn lua if you want something like stored procedures

Setup Manual

-->Type following command in terminal to install Redis server

```
sudo add-apt-repository ppa:rwky/redis
```

```
sudo apt-get update
```

```
sudo apt-get -y install redis-server
```

```
command the bindIP & save it
```

-->For start the server just give this command

```
sudo redis-server
```

-->Use below command to run Redis client on multiple machines.

```
Javac Redis.java
```

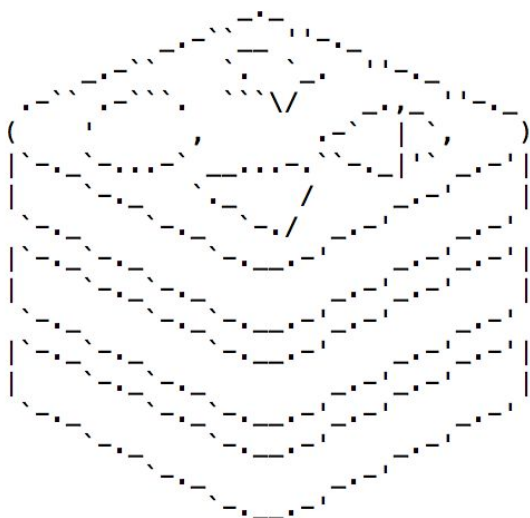
```
Java Redis
```

Design of Redis

-We are using aws instances (t2.micro) for the benchmark of ALL-to-ALL communication.

-When we start server

```
ubuntu@ip-172-31-21-120:/etc/redis$ redis-server
8128:C 21 Nov 23:40:12.794 # Warning: no config file specified, using the default
t config. In order to specify a config file use redis-server /path/to/redis.conf
8128:M 21 Nov 23:40:12.794 * Increased maximum number of open files to 10032 (it
was originally set to 1024).
```



Redis 3.0.6 (00000000/0) 64 bit

Running in standalone mode

Port: 6379

PID: 8128

<http://redis.io>

```
8128:M 21 Nov 23:40:12.795 # WARNING: The TCP backlog setting of 511 cannot be e
nforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
8128:M 21 Nov 23:40:12.795 # Server started, Redis version 3.0.6
8128:M 21 Nov 23:40:12.795 # WARNING overcommit_memory is set to 0! Background s
ave may fail under low memory condition. To fix this issue add 'vm.overcommit_me
memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.over
commit_memory=1' for this to take effect.
8128:M 21 Nov 23:40:12.795 # WARNING you have Transparent Huge Pages (THP) suppo
rt enabled in your kernel. This will create latency and memory usage issues with
Redis. To fix this issue run the command 'echo never > /sys/kernel/mm/transpare
nt_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retai
n the setting after a reboot. Redis must be restarted after THP is disabled.
8128:M 21 Nov 23:40:13.751 * DB loaded from disk: 0.955 seconds
8128:M 21 Nov 23:40:13.751 * The server is now ready to accept connections on po
rt 6379
```

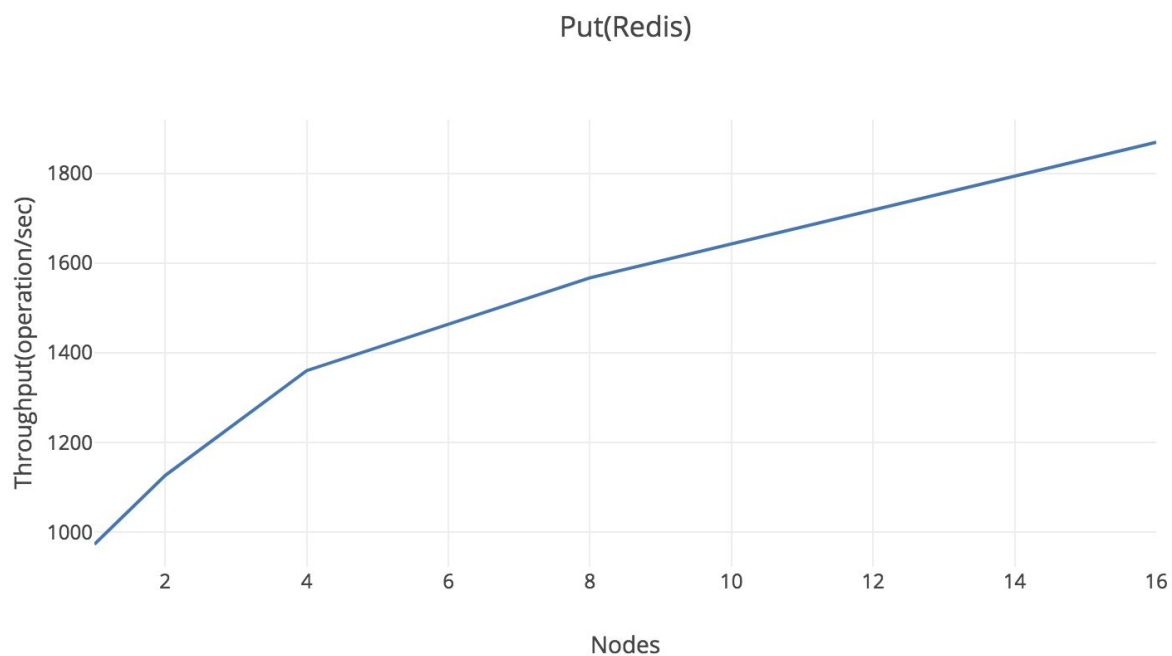
-When we run client code,it gives following output.

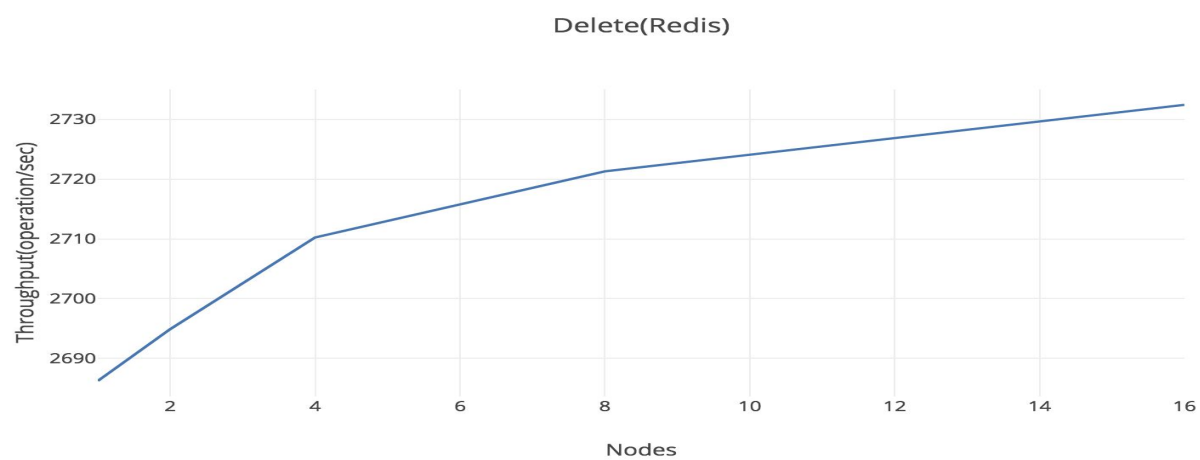
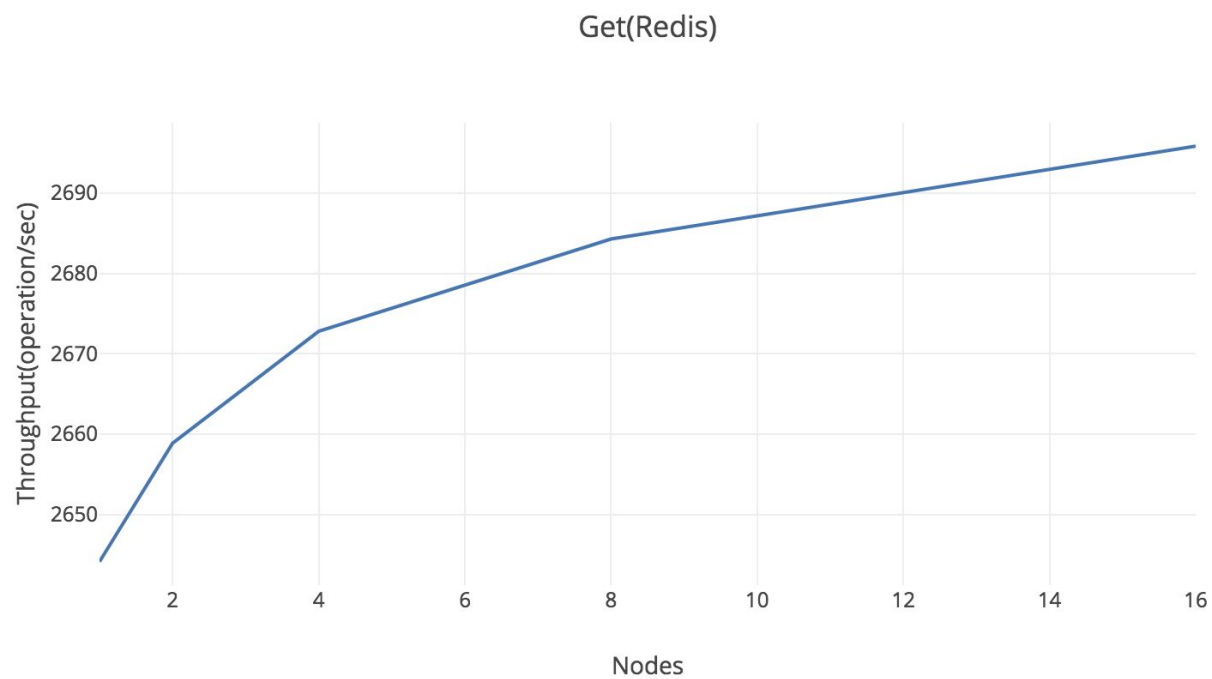
```
canny — ubuntu@ip-172-31-21-120: ~ — ssh — 80x29
ubuntu@ip-172-31-21-120: /etc/redis
ubuntu@ip-172-31-21-120: ~
ubuntu@ip-172-31-21-120:~$ java Redis
Connected with Redis server successfully...
100K PUT operations.
100K PUT operations completed in 38.201 seconds.
100K GET operations.
100K GET operations completed in 37.82 seconds.
100K DEL operations.
100K DEL operations completed in 37.277 seconds.
```

-For all to all communication,we do the same process on different instances.

-I uploaded the client side source code for 100k iteration.

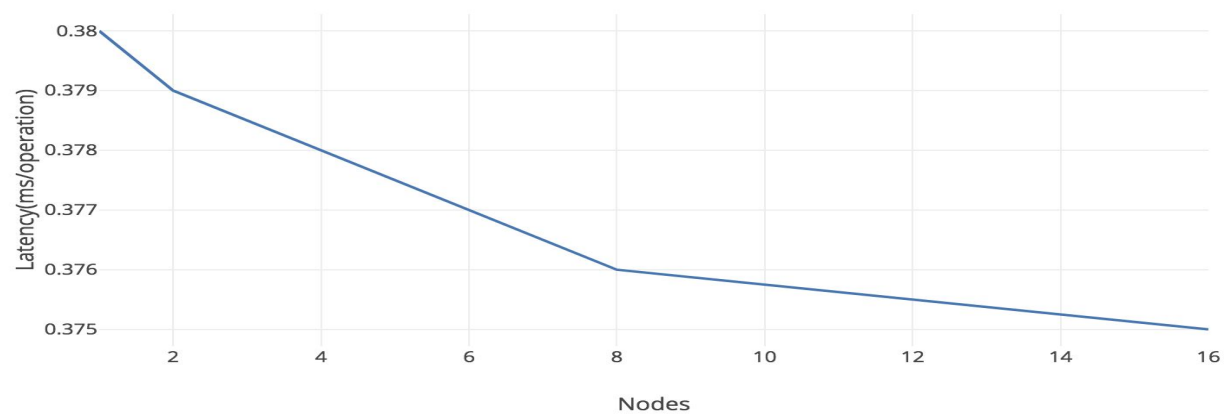
Throughput and Latency of MongoDB



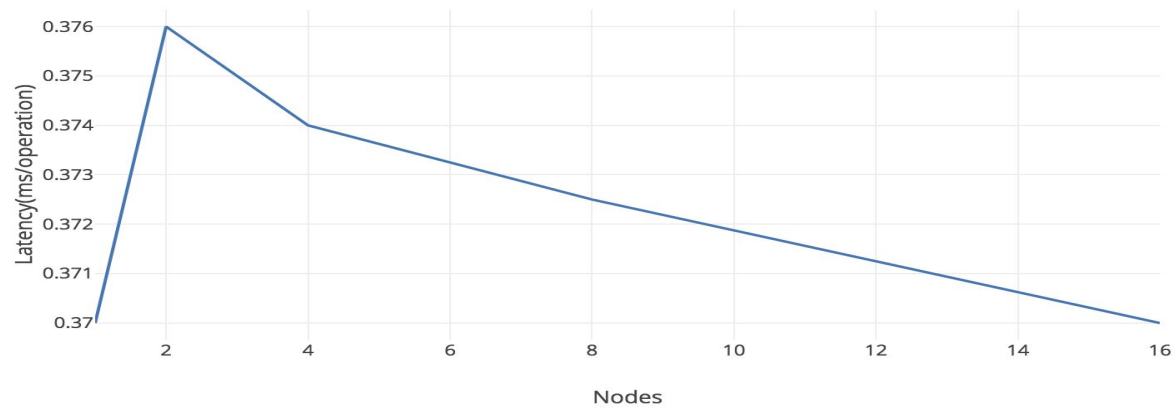




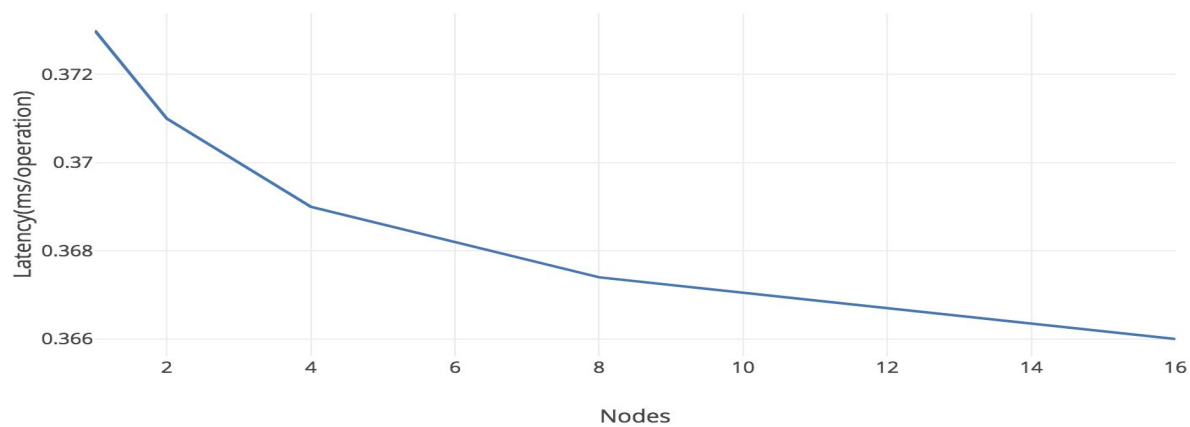
Put(Redis)



Get(Redis)



Delete(Redis)



Riak and Features

Riak is Distributed, fault tolerant key-value store

- Basho Technologies is the developer of it.
- Initial version was released in 2009 and it is open source.
- It is implemented in Erlang & C, some JavaScript.
- Linux, OS X are the server operating systems for it.
- Riak supported many programming languages but less than MongoDB and Redis.
- JavaScript and Erlang are the server-side scripts for Riak.
- It support MapReduce.
- It use selectable replication factor.
- It has Eventual consistency.
- Links & link walking: use it as a graph database
- Secondary indices: but only one at once
- Large object support (Luwak)
- Full-text search, indexing, querying with Riak Search
- In the process of migrating the storing backend from "Bitcask" to Google's "LevelDB"
- Masterless multi-site replication and SNMP monitoring are commercially licensed

—>Best used: If you want something Dynamo-like data storage, but no way you're gonna deal with the bloat and complexity. If you need very good single-site scalability, availability and fault-tolerance, but you're ready to pay for multi-site replication.

Riak's Pros

- Two Modes (Eventual Consistency(All Nodes available during partition) or Strongly Consistent(some nodes might be unavailable for writes as it provides most updated data))
- Masterless
- It's best suited if we have distributed database architecture
- Available Data Types give us high availability advantage
- Uses Vector Clocks to resolve Version Conflicts
- Key/Value DB

- Accepts Read/Write requests while in partition., but returns the most updated data

Riak's Cons

- Best if we use no fewer than 5 Database servers
- Query Efficiency drops if we don't use key/value lookup(If we do "Select * from Table)

Setup Manual

-->Type following command in terminal to install Riak server

```
curl https://packagecloud.io/gpg.key | sudo apt-key add -  
echo "deb https://packagecloud.io/basho/riak/ubuntu/ trusty main" | sudo tee -a  
/etc/apt/sources.list.d/riak.list  
echo "deb-src https://packagecloud.io/basho/riak/ubuntu/ trusty main" | sudo tee -a  
/etc/apt/sources.list.d/riak.list  
sudo apt-get update  
sudo apt-get install riak
```

-->For start the server just give this command

```
sudo service riak start
```

-->For check the status of server,use it

```
sudo service riak status
```

-->Use below command to run Riak client on multiple machines.

```
Javac Riak.java
```

```
Java Riak
```

Design of Riak

-We are using aws instances (t2.micro) for the benchmark of ALL-to-ALL communication.

-When we start server

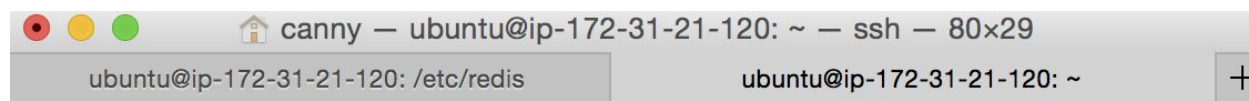


Get cloud support with Ubuntu Advantage Cloud Guest:
<http://www.ubuntu.com/business/services/cloud>

79 packages can be updated.
 0 updates are security updates.

```
*** System restart required ***
Last login: Tue Nov 21 23:37:55 2017 from 108.227.188.7
ubuntu@ip-172-31-21-120:~$ sudo service riak start
ubuntu@ip-172-31-21-120:~$ sudo service riak status
● riak.service - Riak is a distributed data store
   Loaded: loaded (/lib/systemd/system/riak.service; disabled; vendor preset: en
   Active: active (running) since Tue 2017-11-21 23:42:06 UTC; 4s ago
   Process: 8220 ExecStart=/usr/sbin/riak start (code=exited, status=0/SUCCESS)
  Main PID: 8287 (beam.smp)
    Tasks: 162
   Memory: 100.9M
      CPU: 3.279s
   CGroup: /system.slice/riak.service
           └─8276 /usr/lib/riak/erts-5.10.3/bin/epmd -daemon
             └─8286 /usr/lib/riak/erts-5.10.3/bin/run_erl -daemon /tmp/riak// /var
               └─8287 /usr/lib/riak/erts-5.10.3/bin/beam.smp -scl false -sfwi 500 -P
                 └─8486 sh -s disksup
                   └─8488 /usr/lib/riak/lib/os_mon-2.2.13/priv/bin/memsup
                     └─8489 /usr/lib/riak/lib/os_mon-2.2.13/priv/bin/cpu_sup
```

-When we run client code, it gives following output.

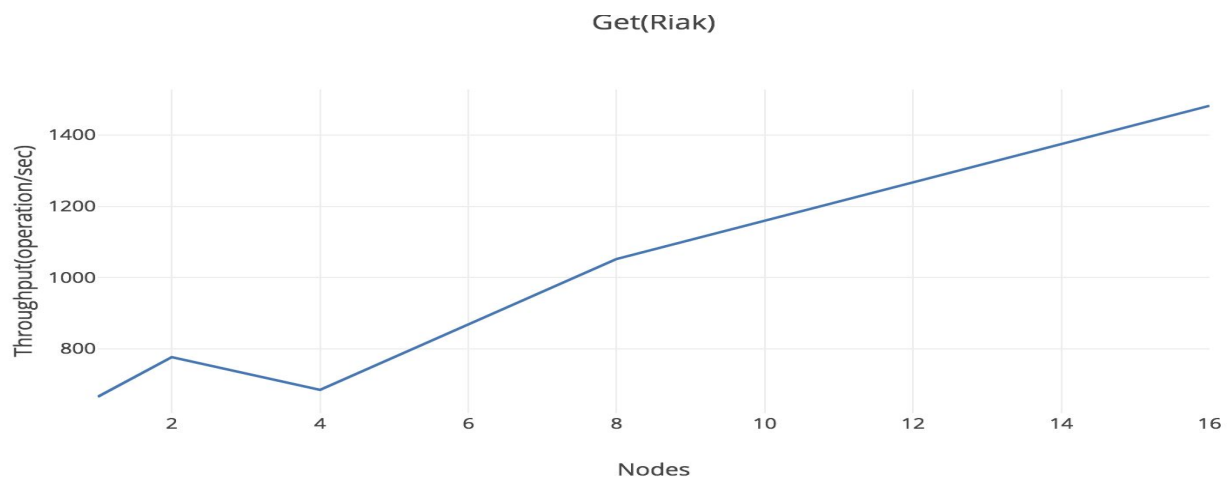
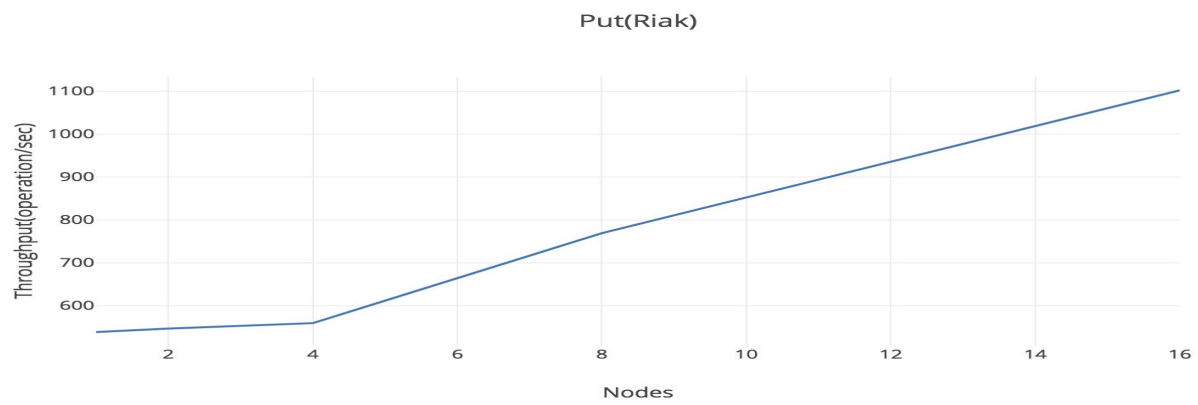


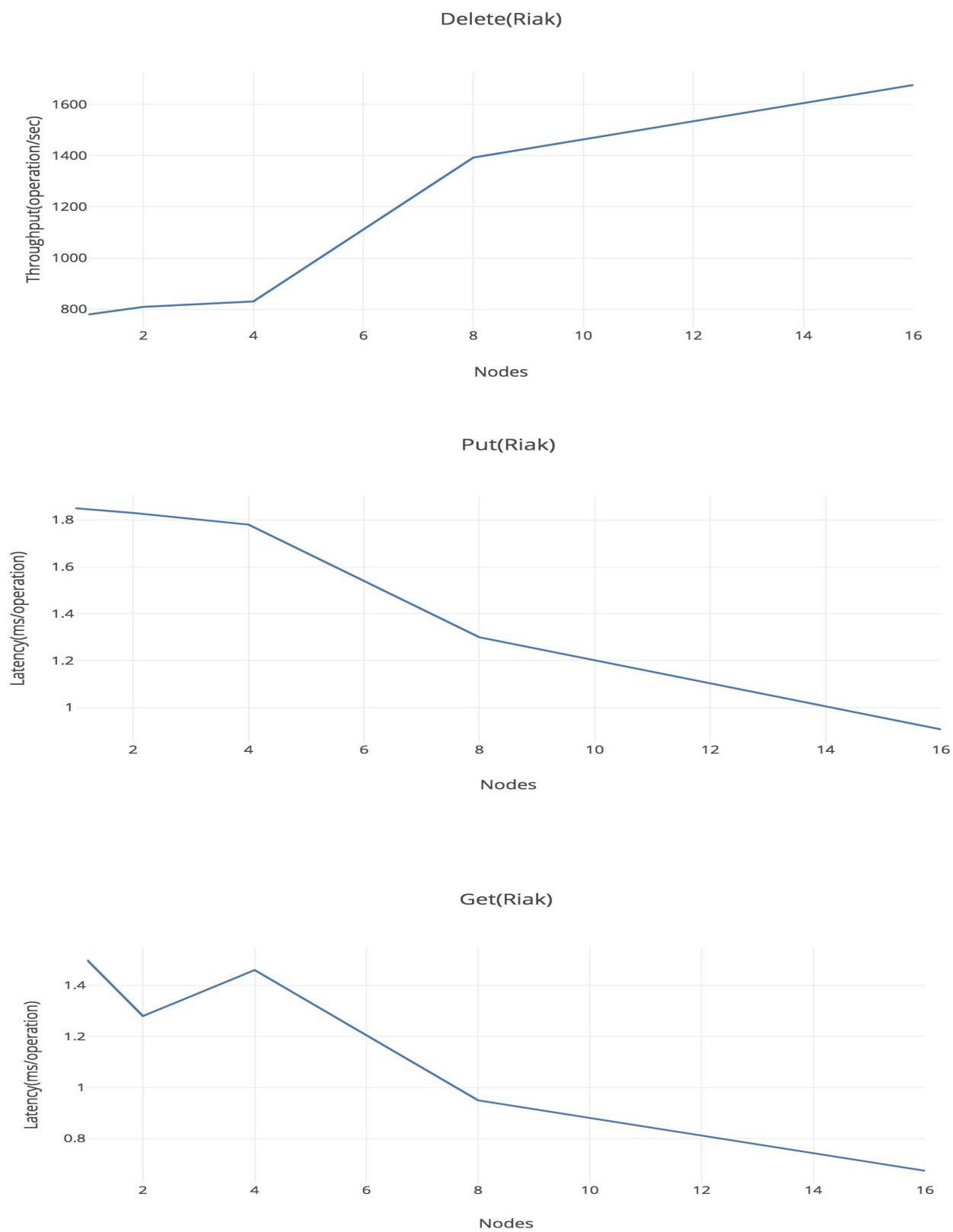
```
ubuntu@ip-172-31-21-120:~$ java Riak
Connected with Riak server successfully...
100K PUT operations.
100K PUT operations completed in 185.772 seconds.
100K GET operations.
100K GET operations completed in 150.283 seconds.
100K DEL operations.
100K DEL operations completed in 125.687 seconds.
```

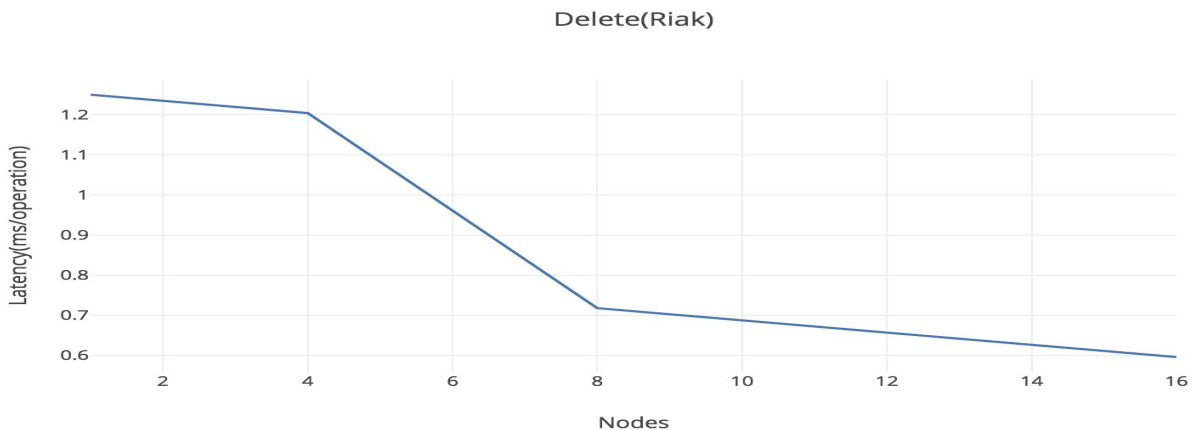
-For all to all communication, we do the same process on different instances.

-I uploaded client side source code for 100k iteration.

Throughput and Latency of Riak







HBase and Features

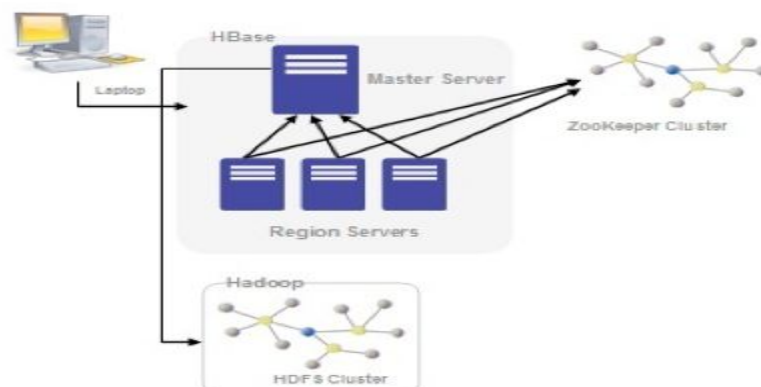
Few years back RDBMS was the only solution for data storage. Then after people realize the use of big data and its usage so people started using Hadoop. It is a distributed file system which is mainly useful for big data it is use MapReduce for processing that data

As we know there are many databases there like HBase, couch DB all this store huge amount of data and use that data in random manner.

What is HBase: HBase uses Hadoop file system it is apaches open source and distributed data base. Cassandra is linearly distributed and HBase is horizontally distributed.

Google big table was renamed as HBase. HBase is built on top of Hadoop so it will give fast random access to big data or huge data. As we know that because of Hadoop we can have facility to provide real time read write access of data.

This Hadoop file system is called HDFS in HBase data will be read and write through HBase instead of direct insertion of data.



We can setup HBase cluster as shown in figure

HBase Pros

- HBase is well known or works good for applications which requires more write.
- Because of Hadoop it gives very fast access of data.
- HBase has one service called Hmaster that will take charge as auto failover.

HBase Cons

- Hmaster is one benefit for HBase but if we have only one Hmaster then we will have single point failure
- Because of the problem of join we must have to use Hadoop MapReduce.
- To improve performance HBase have lower security because security will add so much overhead

Setup Manual

- For HBase we need SSH up and running as well we must have key value. We can generate that pair by

```
$ ssh-keygen -t rsa
```

- Java is prerequisite of HBase

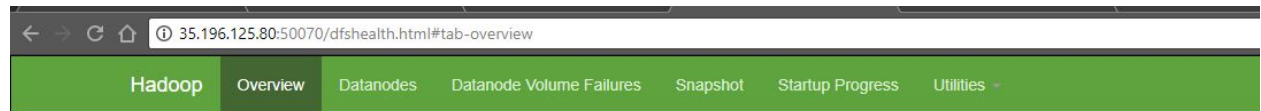
```
$ java -version
```

- Now we need to install Hadoop and verify Hadoop by

```
$ /hadoop/hadoop version
```

- After we need to configure hadoop by step by step manner.
- We can check hadoop is working or not by

<input type="checkbox"/>	✓	hadoop	us-east1-b	10.142.0.3	35.196.125.80 ↗	SSH ▾	⋮
<input type="checkbox"/>	✓	lamp-1-vm	us-central1-f	10.128.0.5	35.192.83.174	SSH ▾	⋮
<input type="checkbox"/>	✓	patelrahul	us-east1-b	10.142.0.2	35.185.29.105 ↗	SSH ▾	⋮



Overview 'localhost:9000' (active)

Started:	Sat Nov 11 05:47:36 UTC 2017
Version:	2.7.1, r15ecc87ccf4a0228f35af08fc56de536e6ce657a
Compiled:	2015-06-29T06:04Z by jenkins from (detached from 15ecc87)
Cluster ID:	CID-555785f9-7c37-4539-a05a-7b7552d145b7
Block Pool ID:	BP-721843312-10.142.0.3-1510379251551

- Now last step installation of HBase by

```
$wget http://apache.claz.org/hbase/stable/hbase-1.2.6-bin.tar.gz
```

```
$tar -zxvf hbase-0.98.8-hadoop2-bin.tar.gz
```
- After installing HBase we can check HBase is working or not by



Master `hadoop.c.hbase-test-184619.internal`

Region Servers

Design of HBase

- We are using aws instances (t2.micro) for the benchmark of ALL-to-ALL communication.
- We can see cli of HBase


```
hbase(main):001:0>
```

```
-m Set memory debugging flag (trace|record|usage)
```

```
root@hadoop:/home#
```

1681 history

```
root@hadoop:/usr/local#
```

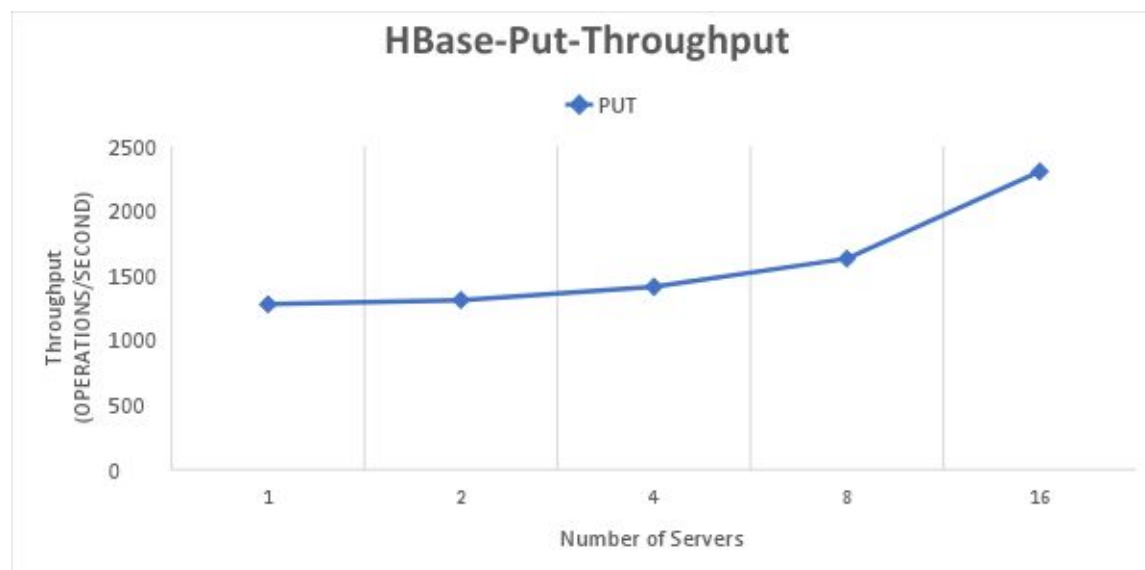
- Get output of 100k operations.

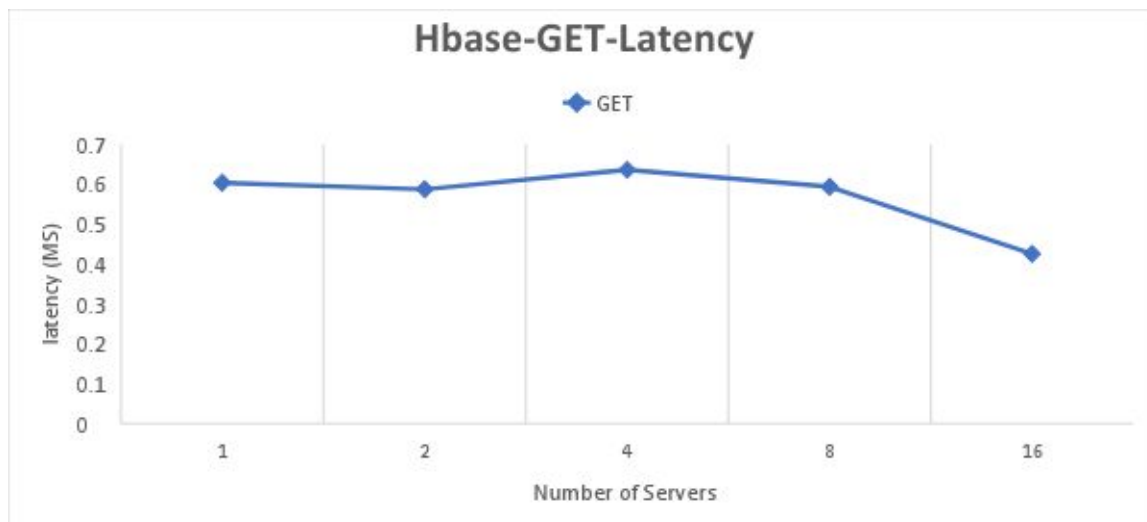
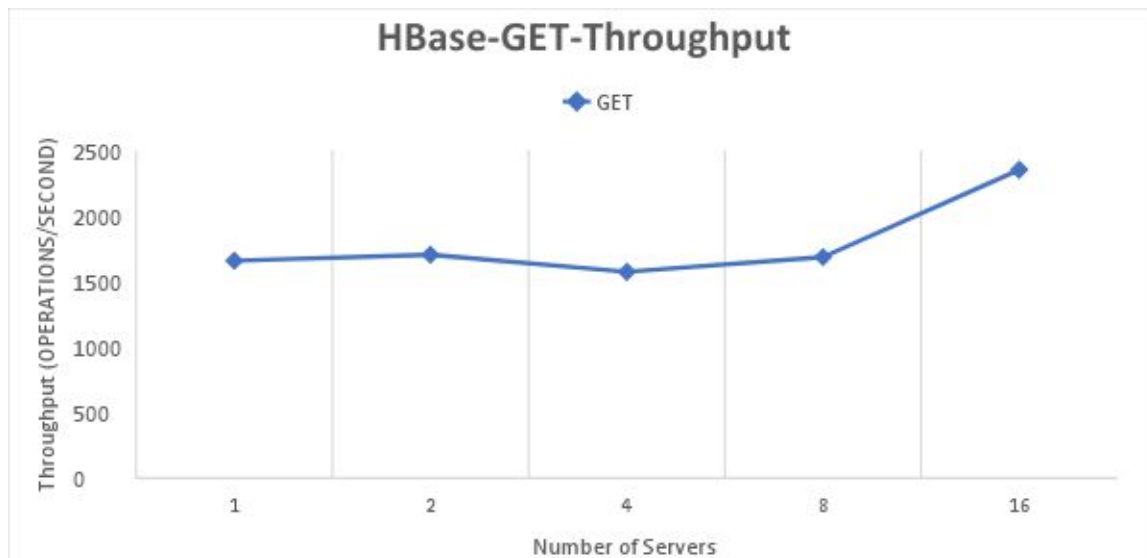
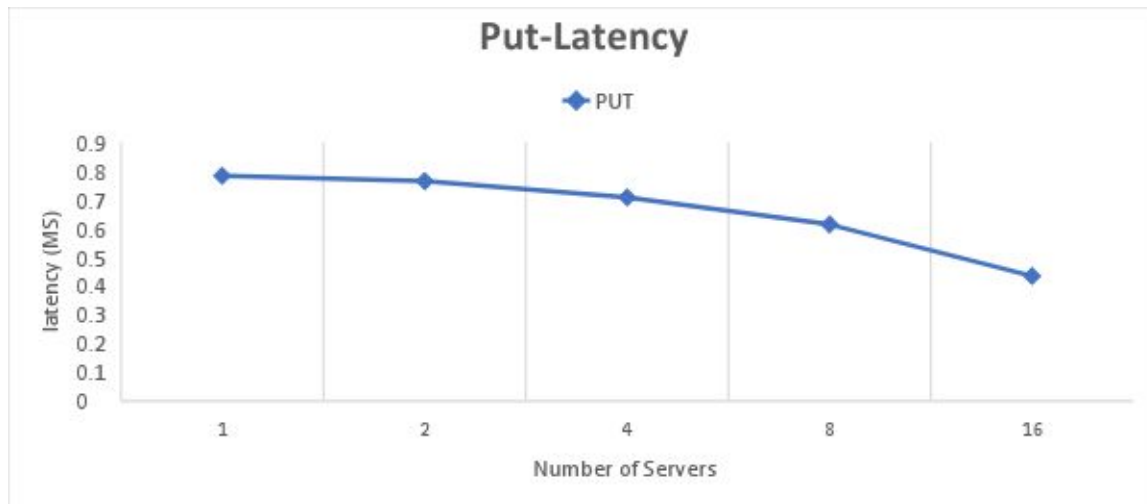
```

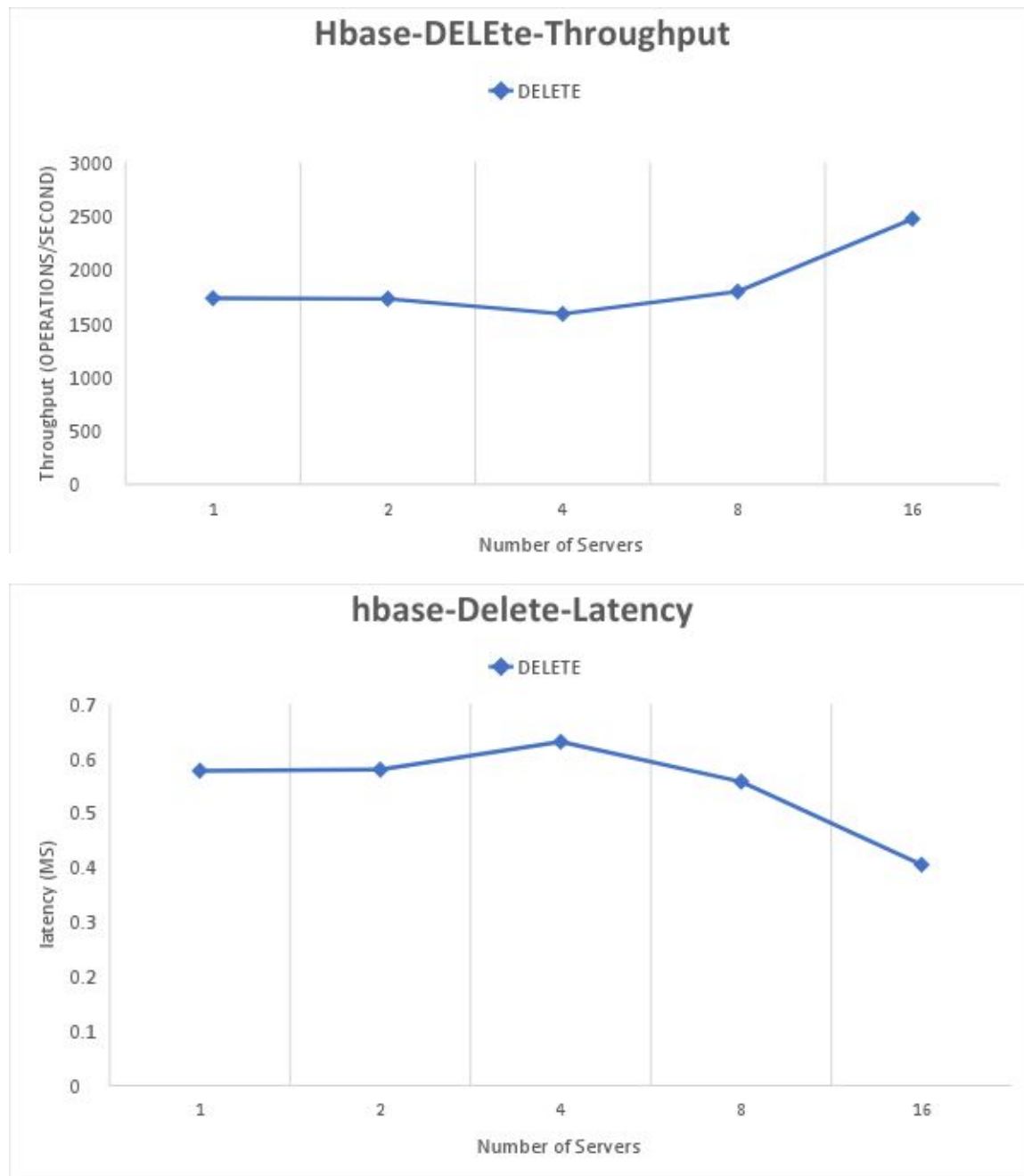
2017-11-29 17:43:16,601 INFO [main] client.HBaseAdmin: Created bbbb
Table created
value: gEAS7cNwqLYoAxspG1j6hsera9NLQNinChst4M2QRaNgLmaTzRID6H1pQehG5F5Lac5koWJpaLiEBFHy35WkHqQ7j2 key: 1HgOSpKrBRNO:99999
Put Started in milliseconds:1511977397813
Put ended in milliseconds:1511977450477
total time in milliseconds:52664
total time in seconds:52
Read Started in milliseconds:1511977450478
Read ended in milliseconds:1511977464945
total time in milliseconds:14467
total time in seconds:14
value: oKIzXq6HeiNuU2W7ttAGKCi5jYgRAZsVEOvTsEnWUtU1pYt7hat5ASHTpx3pxxliKkT3epzhNMBw67sbP1be62RZPg key: Yi24PU8GT1 NO: 0
value: gEAS7cNwqLYoAxspG1j6hsera9NLQNinChst4M2QRaNgLmaTzRID6H1pQehG5F5Lac5koWJpaLiEBFHy35WkHqQ7j2 key: 1HgOSpKrBR NO: 99999
Delete Started in milliseconds:1511977464945
Delete ended in milliseconds:1511977499894
total time in milliseconds:34949
total time in seconds:34
root@hadoop:/var/www/html/BigDataAndHadoop.Session18.Assignment2#

```

Throughput and Latency of HBase







Cassandra and Features

Cassandra is very popular database. It is also one distributed and high-performance database. We know that in HBase there is problem of single point of failure, but that problem is solved in Cassandra.

Cassandra is also one of the NoSQL. SQL use only for tabular data but NoSQL can be use for non-tabular database.

Cassandra was developed by one well known social website named FACEBOOK for its searching mechanism of inbox. After some time Facebook make it open source in 2008.

Gossip protocol was developed for Cassandra which will take care of node to node communication and check for any faulty one in cluster.

We Cassandra is combination of dynamo and bigdata. Prerequisite for Cassandra are JAVA and Datastax.

Cassandra Pros

- Cassandra can support all data types like structured, unstructured and semi as well.
- Cassandra is developed for low performance hardware so its can give high performance with good hardware.
- HBase can support horizontally scalable and Cassandra is linearly so for linearity that will give superior performance with more servers.
- Most beneficial thing is that Cassandra have similar cli as SQL so its really easy to use Cassandra.

Cassandra Cons

- As we know same as HBase Cassandra also doesn't support join.
- We know that Cassandra uses gossip protocol, so it will add some background overhead and that's why performance of Cassandra can be unpredictable sometimes.

Setup Manual

- First of all we will check that prerequisite of Cassandra is there or not so we check java was installed or not if not then we will first install java.

```
$ java -version
```

- Now we move on the installation of Cassandra for that we add Cassandra repository by

```
$ echo "deb http://www.apache.org/dist/cassandra/debian 311x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list
```

- Add key for that

```
$ curl https://www.apache.org/dist/cassandra/KEYS | sudo apt-key add -
```

```
$ sudo apt-key adv --keyserver pool.sks-keyservers.net --recv-key A278B781FE4B2BDA
```

- Now we are good to go and install Cassandra by

```
$ sudo apt-get install cassandra
```

- We can start Cassandra by

```
$ sudo systemctl start cassandra.service
```

<input type="checkbox"/> Name ^	Zone	Recommendation	Internal IP	External IP	Connect
<input type="checkbox"/>  cassandra-3-01	us-central1-a		10.128.0.4	35.184.136.65	SSH ▾ ⋮
<input type="checkbox"/>  cassandra-3-02	us-central1-a		10.128.0.3	104.154.255.78	SSH ▾ ⋮

```
Last login: Sun Nov 26 19:36:30 2017 from 74.125.72.34
thatsmerahul1992@cassandra-3-01:~$ sudo su
bash: export: `=': not a valid identifier
bash: export: `:/home/thatsmerahul1992/': not a valid identifier
root@cassandra-3-01:/home/thatsmerahul1992# cd /
root@cassandra-3-01:/# cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.9 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
cqlsh>
```

- Now we only left with driver that is java api driver. We can use datastax driver for Cassandra <https://github.com/datastax/java-driver>. We can install that driver by using installation manual of datastax.

Design of Cassandra:

-Get output of 100k operations.

```
Put Started in milliseconds:1511993672735
Put ended in milliseconds:1511993715737

total time in milliseconds:43002
total time in seconds:43

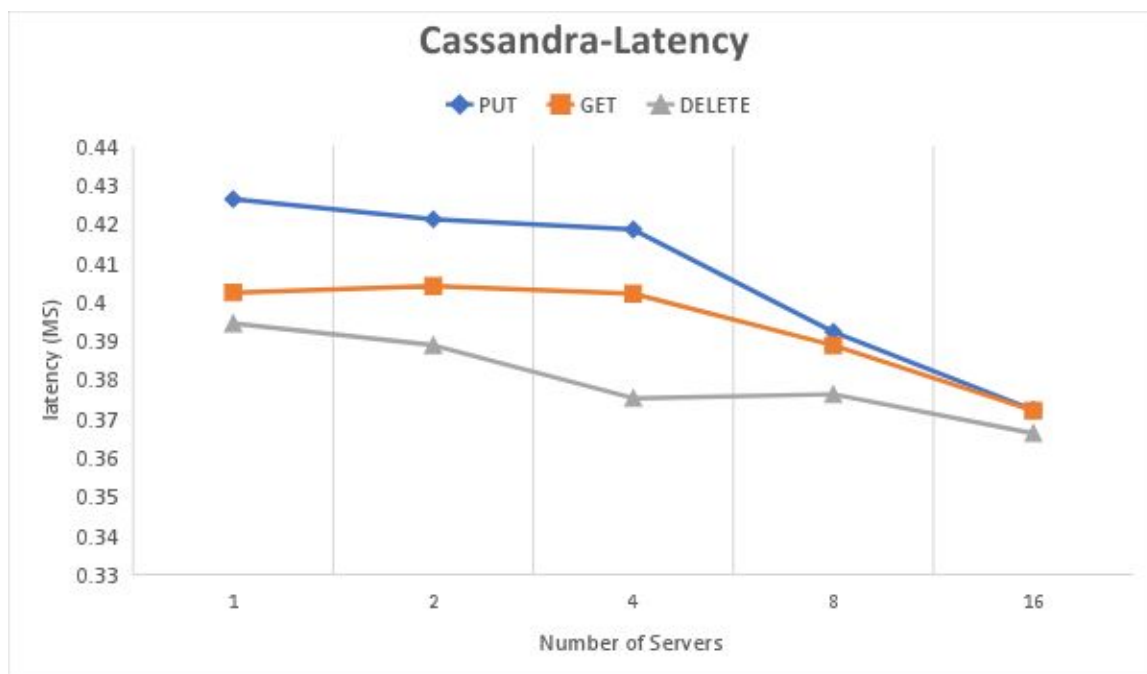
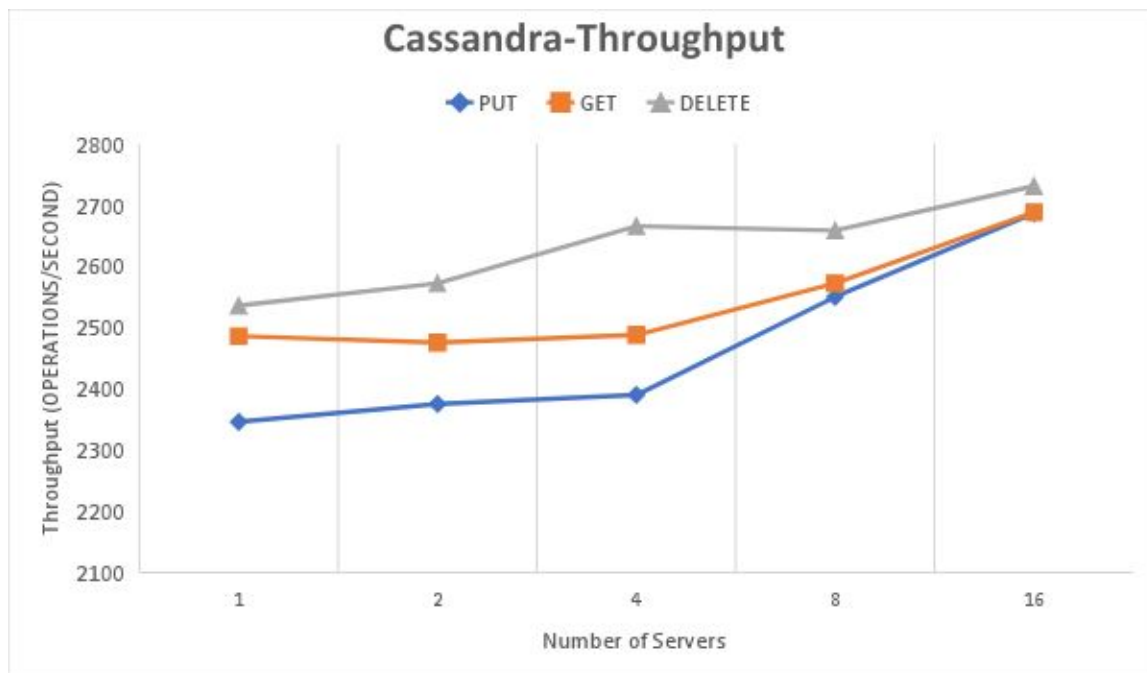
Read Started in milliseconds:1511993715737
Read ended in milliseconds:1511993727520

total time in milliseconds:11783
total time in seconds:11

Delete Started in milliseconds:1511993727520
Delete ended in milliseconds:1511993755868

total time in milliseconds:28348
total time in seconds:28
```

Throughput and Latency of Cassandra:



Couch DB and its Features:

There are 3 type of NoSQL database these are Key-value, Column Store and Document Store. From there three database MongoDB and CouchDB uses Document storage.

CouchDB its name tells everything about that database. Couch means relax this database is developed by apache for easy to use purpose.

It is not that much famous as HBase and Cassandra because of its performance. That degradation in performance is because of its database type.

CouchDB Pros:

- CouchDB is known for easy to use rest api. It provides HTTP Rest api.
- CouchDB is very simple for replication.
- CouchDB is good for security purpose because it will check authentication at data insertion.

CouchDB Cons

- It is slow because of querying is expansive.
- CouchDB have more overhead as other database.

Setup Manual

- Installation of CouchDB is very simple as compared to other databases.

```
$ sudo apt-get install couchdb
```

- CouchDB have HTTP rest api so we need to install curl




```
$ sudo apt-get install curl
```

- Now we are good to go we use by admin panel like phpMyAdmin on port 5984.

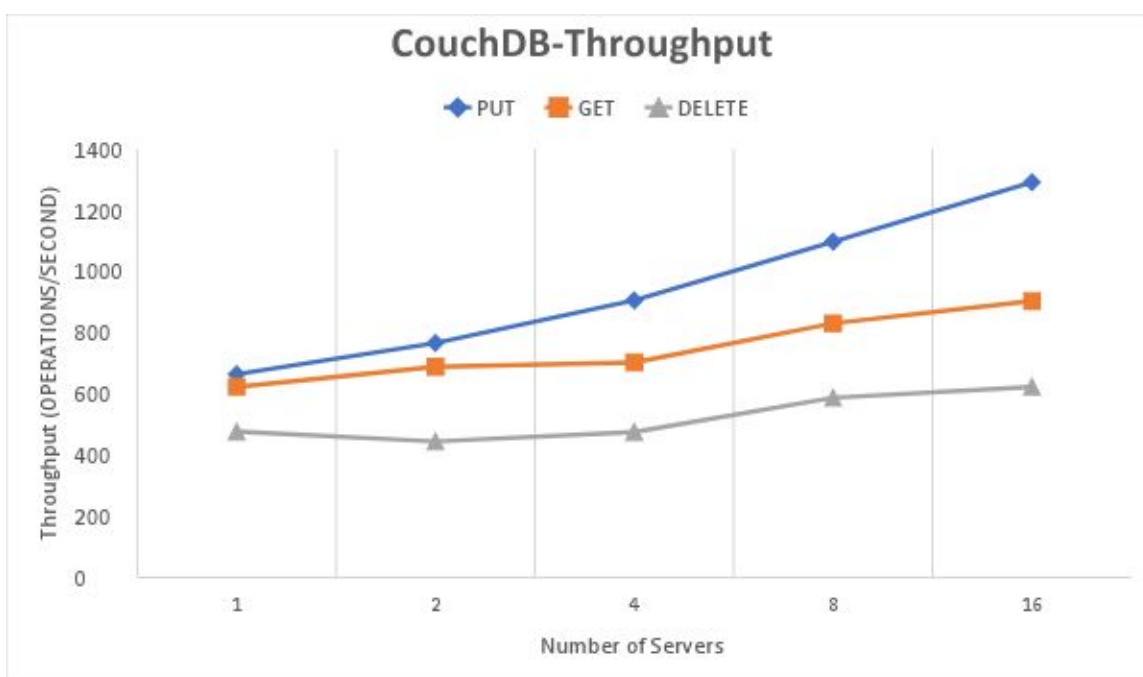
<input type="checkbox"/>	<input checked="" type="checkbox"/>	couchdb-2-vm	us-central1-f	10.128.0.2	35.192.53.183	SSH	▼	⋮
<input type="checkbox"/>	<input checked="" type="checkbox"/>	hadoop	us-east1-b	10.142.0.3	35.196.125.80 ↗	SSH	▼	⋮
<input type="checkbox"/>	<input checked="" type="checkbox"/>	lamp-1-vm	us-central1-f	10.128.0.5	35.192.83.174	SSH	▼	⋮
<input type="checkbox"/>	<input checked="" type="checkbox"/>	patelrahul	us-east1-b	10.142.0.2	35.185.29.105 ↗	SSH	▼	⋮

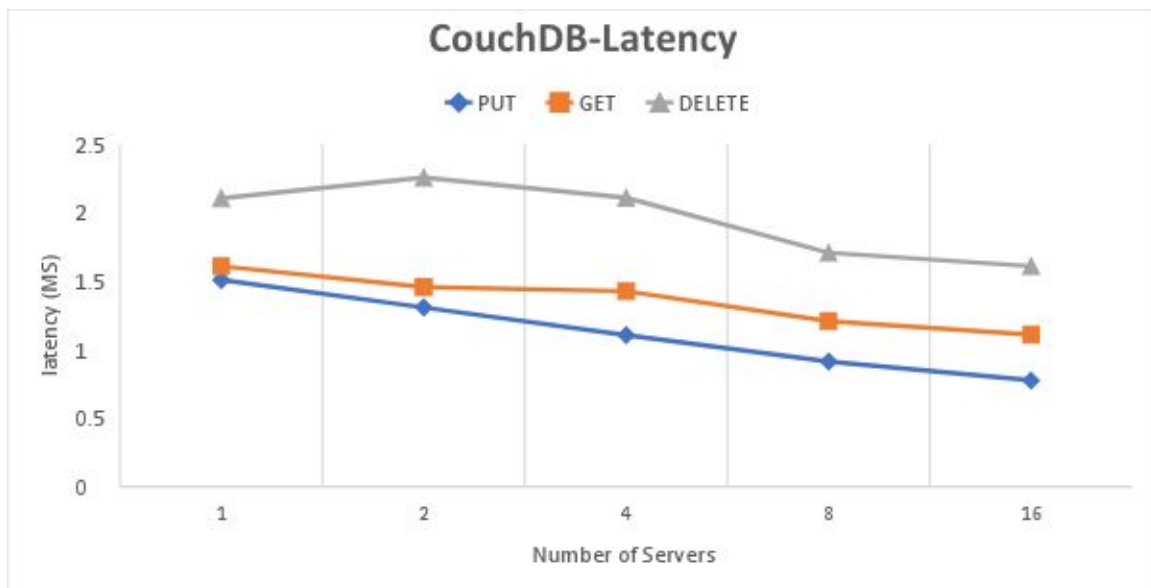
← → ↻ 🏠 ⓘ Not secure | 35.192.53.183:5984/_utils/#/_all_dbs ☆

Databases Database name ▼ Create Databases

Name	Size	# of Docs	Actions
_replicator	This database failed to load.		
_users	This database failed to load.		
verifytestdb	33.2 KB	0	  

Throughput and Latency of CouchDB:

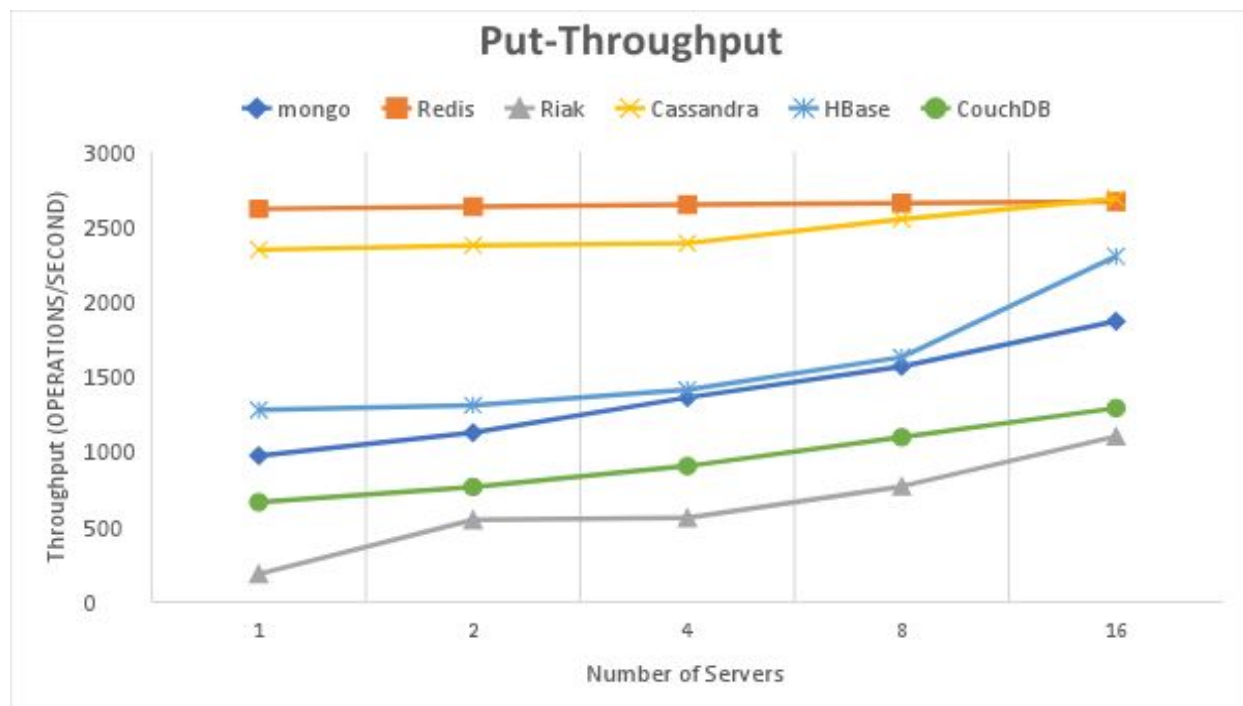
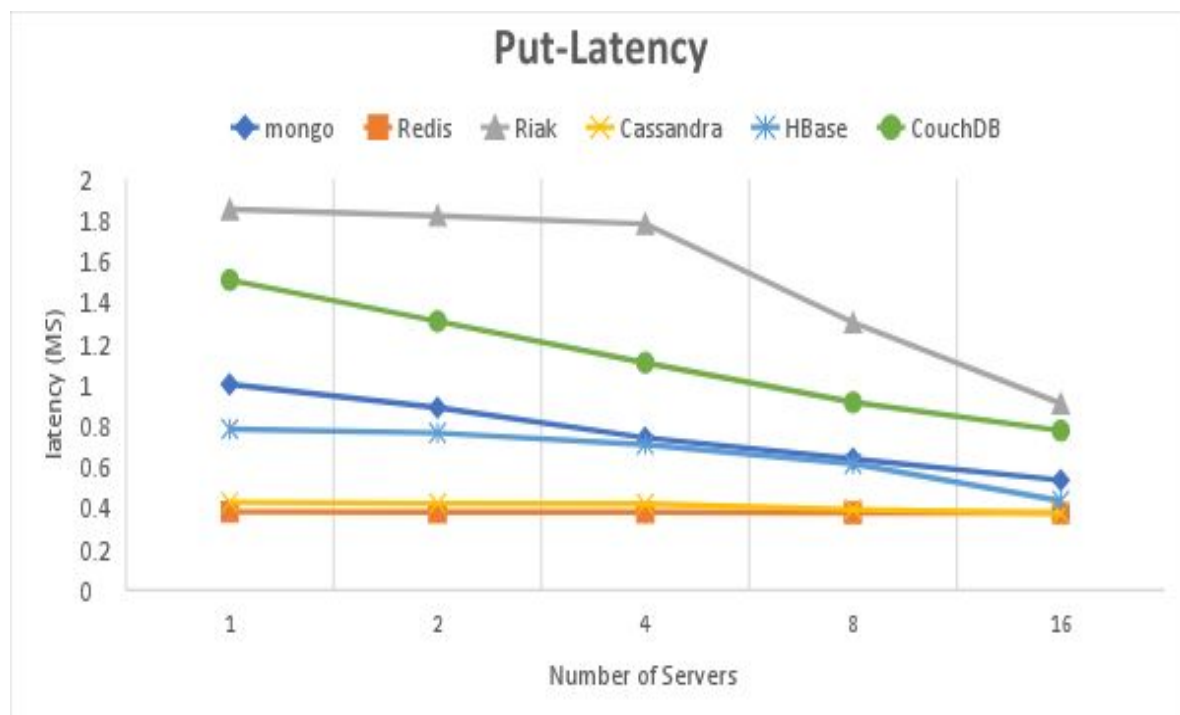




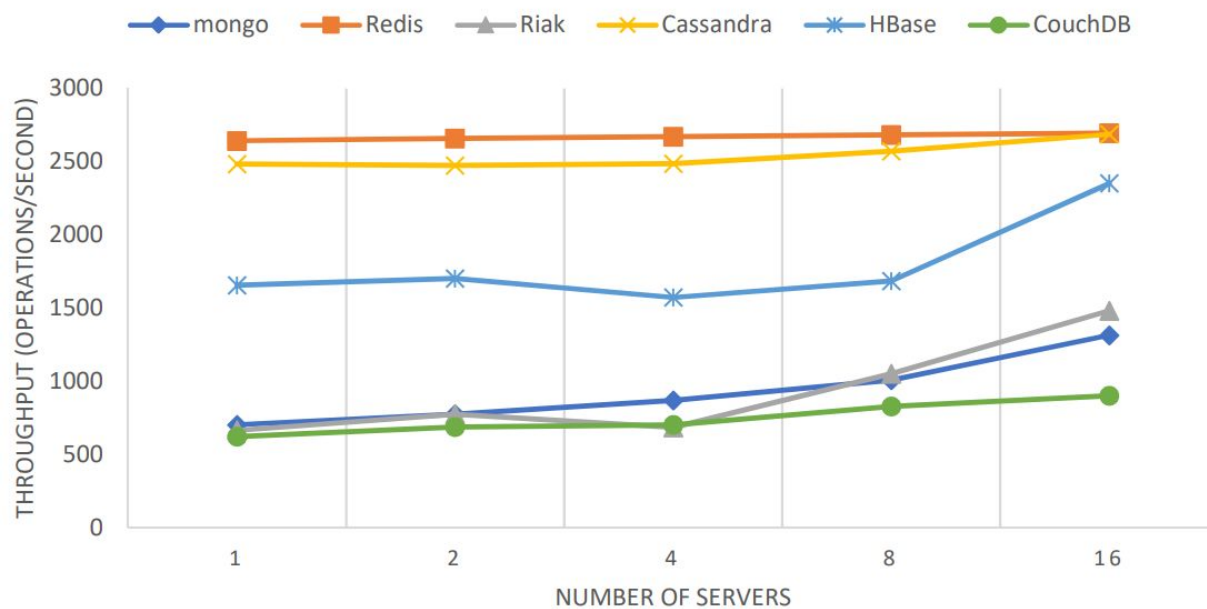
Comparison Graphs of Throughput and Latency:

In this comparison we plot all 6 databases for throughput and latency. We compare all database performance for 1,2,4,8,16 no of servers.

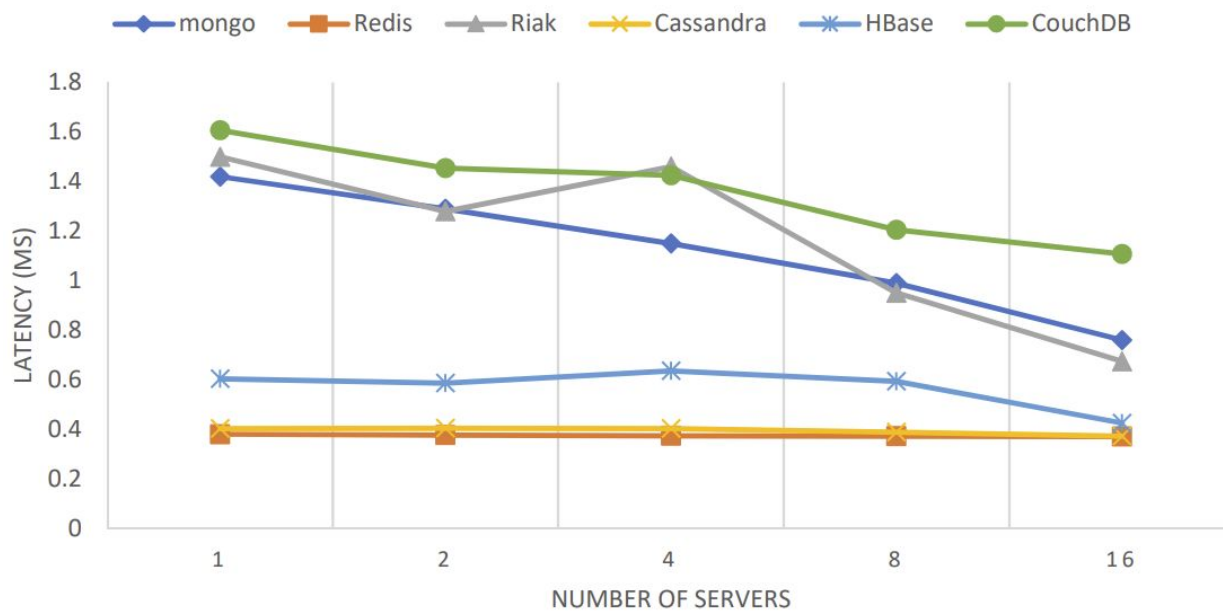
We took individual readings for PUT, GET and DELETE over 100k operations. After getting all results in time unit of milliseconds. We calculated throughput by (100k operations / total time taken) and we calculated latency by (1/ Throughput).

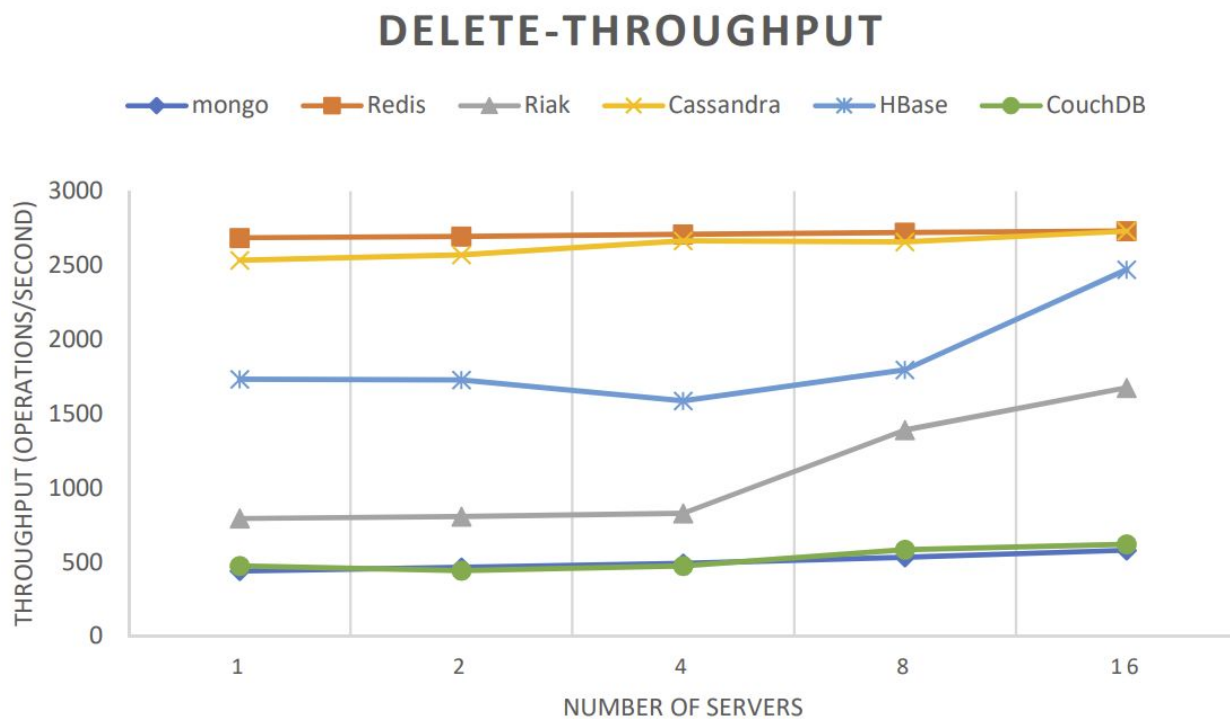
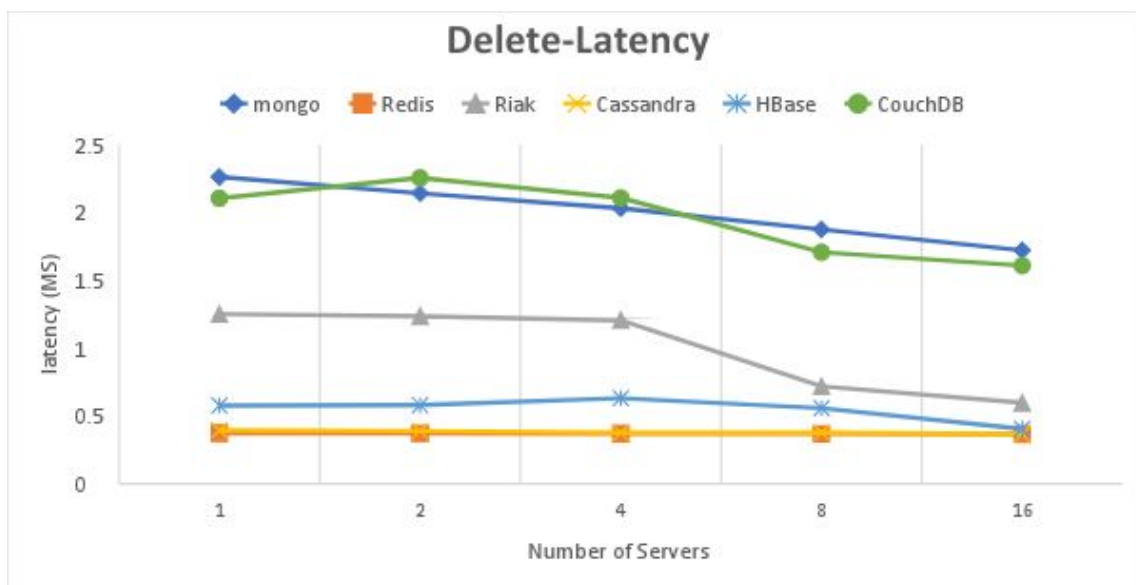


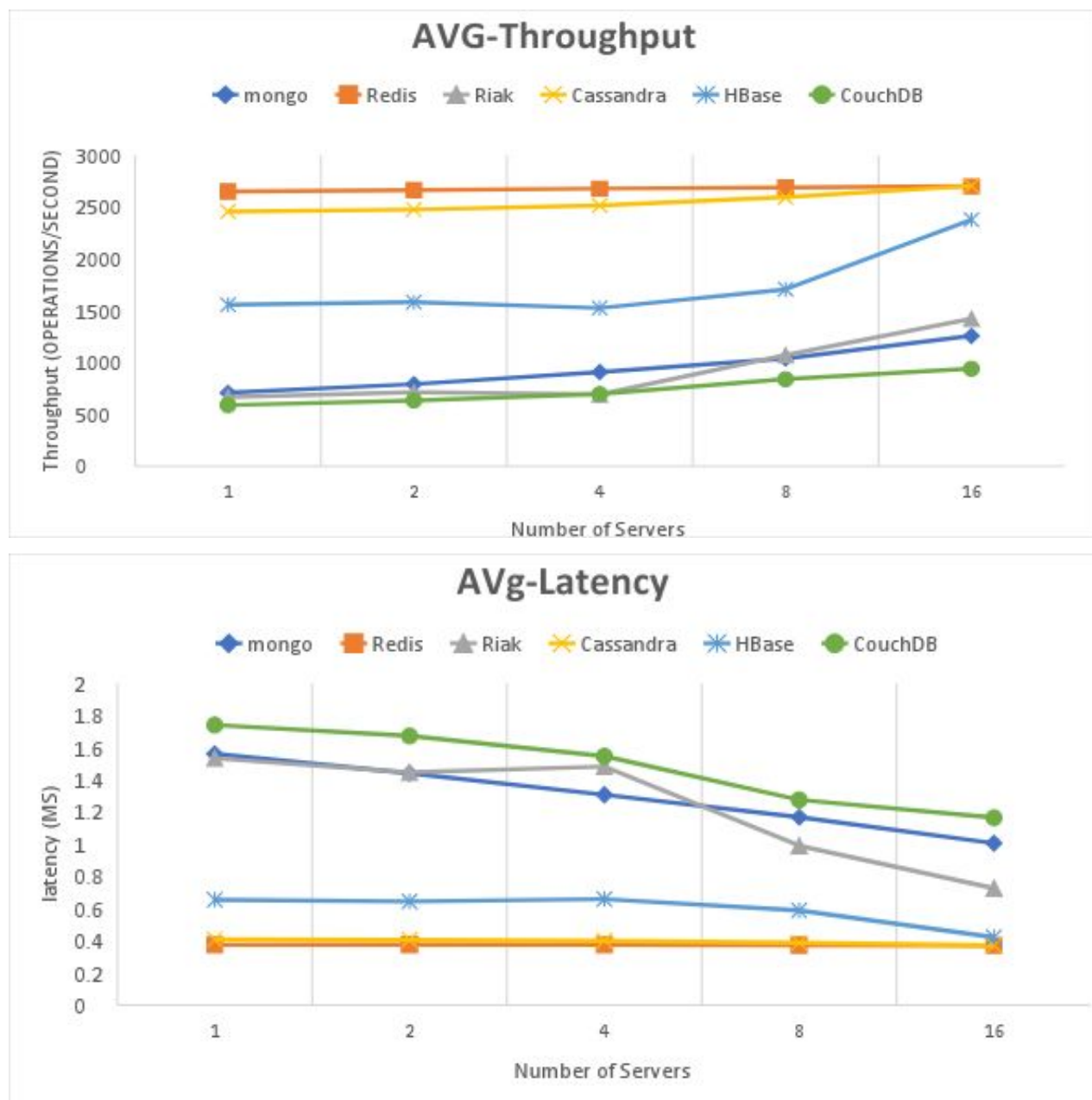
GET-THROUGHPUT



GET-LATENCY







Conclusion

In this project, we evaluated 6 distributed key-value storage systems for different numbers of servers. From our noted results, Redis and Cassandra have the maximum range of throughput, so we can use these key-value storage systems for quick access of values compared to others. For document storage, we can use MongoDB over CouchDB because MongoDB has higher throughput and lower latency than CouchDB. Overall, Redis gives better performance for all the operations.

Future Work

There should involve more data stores in the benchmark stage. For example, we may involve MySQL, a relational database, to compare the difference between a NoSQL data store and a relational database. In addition, although we have picked up Redis, Cassandra, Riak, MongoDB, CouchDB and HBase as six target key-value stores, there are still a large number of key-value stores that may perform variously. These six key-value stores were excluded by this dissertation because of time limitation. As a result, by comparing various implementations of key-value and relational databases we might have a more precise conclusion about the suitability of data stores for scientific applications.

Finally, the evaluation script could be improved in several approaches. First, the loading script could be parallelized to reduce the time consumption of inserting data rows. Second, the script of client could be rewritten to involve the slave nodes for evaluation of partition tolerance. Thirdly, looking into the source code of client to investigate the root cause of the issue of partition tolerance.

References

1. Avinash Lakshman and Prashant Malik. Cassandra: a decentralized structured storage system. SIGOPS Oper. Syst. Rev., 44(2):35–40, April 2010.
2. Dilipbhai Jogi, Vishal & Sinha, Ashay. (2016). Performance evaluation of MySQL, Cassandra and HBase for heavy write operation. 586-590. 10.1109/RAIT.2016.7507964.
3. Yishan Li, Sathiamoorthy Manoharan, "A performance comparison of SQL and NoSQL databases", proc. of Pacific Rim Conference on Communications Computers and Signal Processing (PACRIM) IEEE, pp. 15-19, 2013.
4. Chen, Hang & Lu, Kun & Sun, Mingming & Li, Changlong & Zhuang, Hang & Zhou, Xuehai. (2015). Enumeration System on HBase for Low-Latency. 1185-1188. 10.1109/CCGrid.2015.78.
5. Ampaporn, Pornpan & Gertphol, Sethavidh. (2015). Performance measurement of SimpleDB APIs for different data consistency models. 1-6. 10.1109/ICSEC.2015.7401438.
6. Tan, Zhipeng & Dang, Yongxing & Sun, Jianliang & Zhou, Wei & Feng, Dan. (2014). PaxStore : A Distributed Key Value Storage System. 8707. 471-484. 10.1007/978-3-662-44917-2_39.
7. Zhong, Yunqin & Sun, Shangchun & Liao, Haojun & Zhao, Yanwei & Fang, Jinyun. (2011). A novel method to manage very large raster data on distributed key-value storage



system. Proceedings - 2011 19th International Conference on Geoinformatics, Geoinformatics 2011. 1-6. 10.1109/GeoInformatics.2011.5980711.

8. Wu, Hui-jun & Lu, Kai & Li, Gen. (2016). Design and Implementation of Distributed Stage DB: A High Performance Distributed Key-Value Database. 189-198. 10.2991/978-94-6239-145-1_19.