Benchmarking Cassandra DB using YCSB

Cassandra is a free and open source distributed database management system that can be used to handle large amounts of data across many servers. It is reliable, clusterable and specially designed to handle very large amounts of structured data. Cassandra also supports replication and multi-data center replication for redundancy, failover, and disaster recovery.

# Installing Cassandra:

We are installing Cassandra on CentOS machine; the following steps must be done on non-root user with sudo privileges.

* First, we have to update the repositories.

*sudo yum update -y*

* Now install java

*sudo yum install java-1.8.0-openjdk -y*

*sudo yum install java-1.8.0-openjdk-devel -y*

*sudo java -version*

*javac*

* By default, Cassandra is not available in CentOS 7 default repository. So, you will need to add the official Apache Software Foundation repositories to your system. You can do this by creating cassandra.repo file inside /etc/yum.repos.d directory.

*sudo vi /etc/yum.repos.d/cassandra.repo*

* Now add following into that file

*[cassandra]*

*name = Apache Cassandra*

*baseurl =* <https://www.apache.org/dist/cassandra/redhat/22x/>

*gpgcheck = 1*

*repo\_gpgcheck=1*

*gpgkey=https*://www.apache.org/dist/cassandra/KEYS

* Just make sure the repository is update

*sudo yum update -y*

* Installing Cassandra

*sudo yum install cassandra -y*

* Start Cassandra service and enable it to start on reboot

*sudo systemctl start cassandra*

*sudo systemctl enable cassandra*

* Check status of Cassandra service

*sudo systemctl status cassandra*

* To enter Cassandra cluster or database

*cqlsh*

* Create key space YCSB to setup its testing environment.

*create keyspace ycsb WITH REPLICATION = {'class' : 'SimpleStrategy', 'replication\_factor': 1 };*

*USE ycsb;*

* Create table named usertable.

*create table usertable (y\_id varchar primary key, field0 varchar, field1 varchar, field2 varchar, field3 varchar, field4 varchar, field5 varchar, field6 varchar, field7 varchar,field8 varchar, field9 varchar);*

* To exit from the Cassandra database

*exit*

# Installing YCSB Client:

YCSB is a popular benchmark tool for NoSQL. It has adapters ready for different NoSQL DB like Cassandra, MongoDB, Redis and others.

YCSB comes with 6 out of the box workloads, each testing a different common use case

* Workload A: Update heavy workload This workload has a mix of 50/50 reads and writes. An application example is a session store recording recent actions.
* Workload B: Read mostly workload This workload has a 95/5 reads/write mix. Application example: photo tagging; add a tag is an update, but most operations are to read tags.
* Workload C: Read only This workload is 100% read. Application example: user profile cache, where profiles are constructed elsewhere (e.g., Hadoop).
* Workload D: Read latest workload in this workload, new records are inserted, and the most recently inserted records are the most popular. Application example: user status updates; people want to read the latest.
* Workload E: Short ranges in this workload, short ranges of records are queried, instead of individual records. Application example: threaded conversations, where each scan is for the posts in a given thread (assumed to be clustered by thread id).
* Workload F: Read-modify-write

The YCSB client can be installed on any machine with minimum requirements and using the ip address of the host machine or server to be tested, we can perform testing operations remotely. But it is suggested to have client and testing server to be in the same network to get accurate results.

## Installation

The YCSB tool requires latest version of JAVA and Maven to be installed in the machine, so we first install those and then extract YCSB tool from Internet.

* Install Java or OpenJDK, am installing OpenJDK here.

*sudo yum install java-1.8.0-openjdk -y*

*sudo yum install java-1.8.0-openjdk-devel -y*

*java -version*

* Installing Maven:

*sudo yum install maven -y*

*mvn -version*

This should work, if not try below steps.

Install Wget, which is a tool used to download using external links.

*sudo yum install wget -y*

Change directory to local.

*cd /usr/local*

Now download Maven files.

*sudo wget* [*http://www-eu.apache.org/dist/maven/maven-3/3.5.2/binaries/apache-maven-3.5.2-bin.tar.gz*](http://www-eu.apache.org/dist/maven/maven-3/3.5.2/binaries/apache-maven-3.5.2-bin.tar.gz)

Extract Maven.

*sudo tar xzf apache-maven-3.5.2-bin.tar.gz*

*sudo ln -s apache-maven-3.5.2 maven*

Set Environment Variable

*sudo vi /etc/profile.d/maven.sh*

It opens an editor, add below details in it

*export M2\_HOME=/usr/local/maven*

*export PATH=${M2\_HOME}/bin:${PATH}*

Check version

*mvn -version*

* Download and extract YCSB

*wget https://github.com/brianfrankcooper/YCSB/releases/download/0.13.0/ycsb-0.13.0.tar.gz*

Extract YCSB

*tar xfvz ycsb-0.13.0.tar.gz*

Change directory to ycsb-xversion

*cd ycsb-0.13.0*

## Performance Testing and Evaluation

After successful setting up Cassandra and installing YCSB, now it’s time to test the machines optimal capability and performance.

### Things to Remember, before running the test:

* Operation type (load/run)
* Workload types (a,b,c,d,e,f)

Example: workloads/workloada, workloads/workloadb

* Datasets (size of the load we are running tests.) we usually take 32gb,64gb,128gb.

Which can be calculate by 1M recordcount = ~1.5GB data. Data size multiplied by field length property, 10 million records with fieldlength=250 equals 25GB of data.

* Recordcount (value for no. of records to load)

Example: recordcount=1000000

* Operationcount (value for no. of records to run)

Example: operationcount=100000

* Threads (no. of ycsb client threads/parallel operations)

Example: -threads 2/4/8/16/32/64/128

* hosts="" (If you are running from another server)

hosts="192.168.50.xxx”

* P file Specify workload file
* P Override workload property

usage: -p key=value

* S Print status to stderr

usage: -s

* Target: Target ops/sec (default: unthrottled)

usage: -target number

### Testing with workloads a & b:

We run workloads depending on our requirement. Let’s Run some default workloads with synchronous and asynchronous process.

Note: Here am running workload on the same machine. To run on remote machine use host parameter which is mentioned in ‘Things to remember’.

Below is workload type ‘a’, with sync and async modes, where test data stores into the file mentioned at the end.

On Local Machine:

*./bin/ycsb load cassandra-cql -s -P workloads/workloada > outputLoad.txt*

*./bin/ycsb run cassandra-cql -s -P workloads/workloada > outputRun.txt*

On Remote Machine:

./bin/ycsb load cassandra-cql -p hosts="192.168.00.xx" -s -P workloads/workloada -s > outputLoadar.txt

*./bin/ycsb run cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloada > outputRunar.txt*

Now Let’s Run some specific workloads to test the machines capabilities.

Here am taking workloads ‘a’ & ‘b’ to test with different properties. We observe ops/sec by increasing no of threads and operations count in the workload, which will let us know the machines capability.

#### Workload A & B with Increasing Threads: -

Workload A:

On Local Machine:

*./bin/ycsb load cassandra-cql -s -P workloads/workloada -p recordcount=1000000 -threads 32 > outputLoadA.txt*

*./bin/ycsb run cassandra-cql -s -P workloads/workloada -p operationcount=1000000 -threads 8 > outputRunA1.txt*

*./bin/ycsb run cassandra-cql -s -P workloads/workloada -p operationcount=1000000 -threads 32 > outputRunA2.txt*

On Remote Machine:

*./bin/ycsb load cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloada -p recordcount=1000000 -threads 16 > outputLoadA.txt*

*./bin/ycsb run cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloada -p operationcount=1000000 -threads 32 > outputRunAr1.txt*

Workload B:

On Local Machine:

*./bin/ycsb load cassandra-cql -s -P workloads/workloadb -p recordcount=1000000 -threads 32 > outputLoadB.txt*

*./bin/ycsb run cassandra-cql -s -P workloads/workloadb -p operationcount=1000000 -threads 8 > outputRunB1.txt*

*./bin/ycsb run cassandra-cql -s -P workloads/workloadb -p operationcount=1000000 -threads 16 > outputRunB2.txt*

On Remote Machine:

*./bin/ycsb load cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloadb -p recordcount=1000000 -threads 32 > outputLoadB.txt*

*./bin/ycsb run cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloadb -p operationcount=1000000 -threads 16 > outputRunBr1.txt*

Keep on increasing the threads, until you see a decrease in the ops/sec, then you have to increase/decrease threads in minimum value possible. From those results decide number of thread counts best suitable.

#### Workload A & B with increasing Operation Count: -

Now from the above test, we decide on best thread count, so keeping that value constant, we change number of operations count to see the maximum possible operations by our machine. Let’s say 16 is the maximum thread count where our machine works properly.

Workload A on Remote Machine:

*./bin/ycsb load cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloada -p recordcount=1000000 -threads 16 > outputLoadA.txt*

*./bin/ycsb run cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloada -p operationcount=100000 -threads 16 > outputRunAr1.txt*

*./bin/ycsb run cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloada -p operationcount=1000000 -threads 16 > outputRunAr2.txt*

*./bin/ycsb run cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloada -p operationcount=10000000 -threads 16 > outputRunAr3.txt*

Workload B on Remote Machine:

*./bin/ycsb load cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloadb -p recordcount=1000000 -threads 16 > outputLoadB.txt*

*./bin/ycsb run cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloadb -p operationcount=100000 -threads 16 > outputRunBr1.txt*

*./bin/ycsb run cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloadb -p operationcount=1000000 -threads 16 > outputRunBr2.txt*

*./bin/ycsb run cassandra-cql -p* hosts="192.168.00.xx" *-s -P workloads/workloadb -p operationcount=10000000 -threads 16 > outputRunBr3.txt*

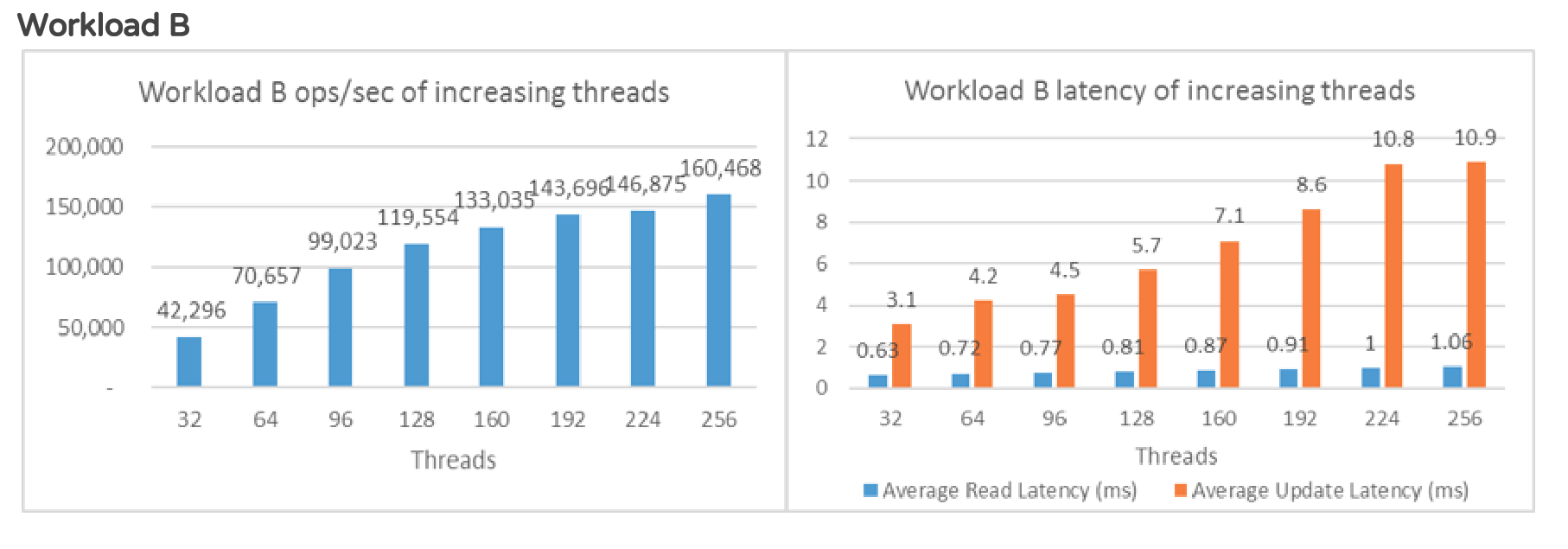
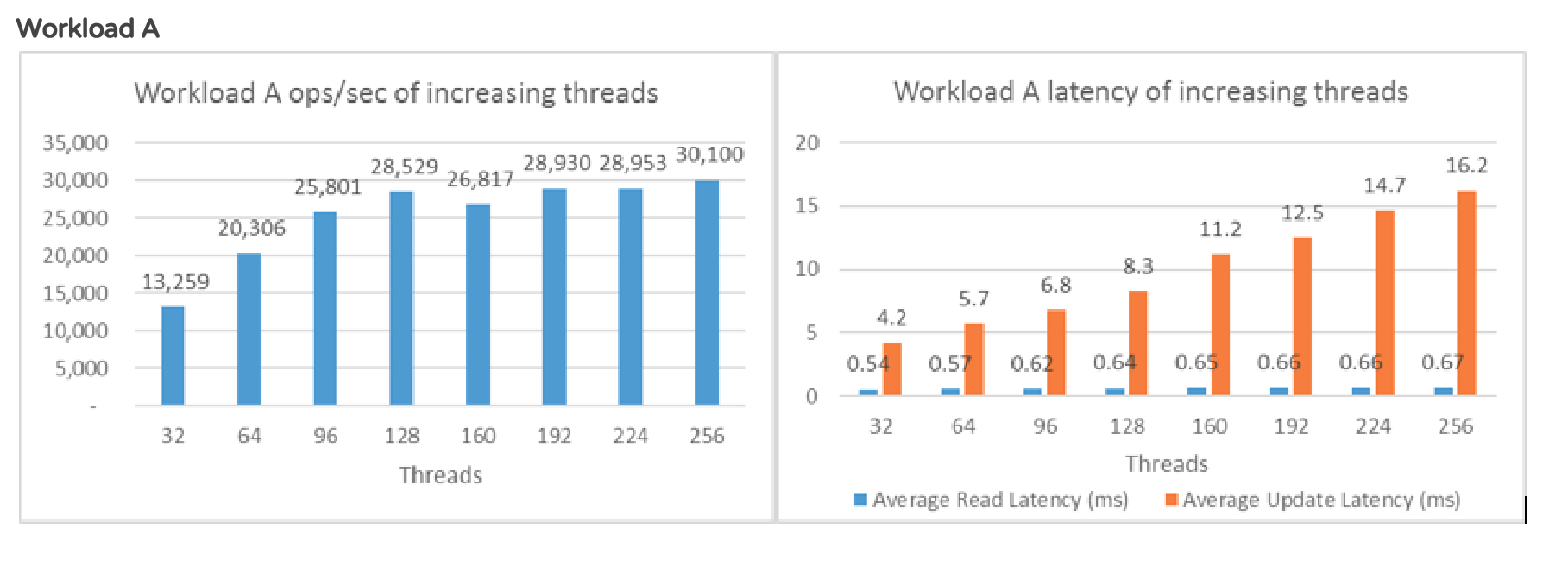
By increasing and decreasing number of operationcount, we can come to a point that at what maximum values our machine is accurate.

Note: We need to look for memory usage and cpu usage of the machine we are running load to know its operational limits.

### Plotting results:

From the above testing results, we gather details such as threads count, operations count, operation/second, read & update latency, test load type to plot graphs to find point at which the machine is working with high performance with great accuracy and least data loss.

The results may look similar as below.

Changing Threads count:

#### Changing Operations count:

