Erasure Coding for In-Memory Key-Value Store

Rishika Reddy Akavaram
Graduate Student, Dept. of Computer Science
Texas Tech University
Lubbock, TX, United States
shashank-koundinya.cheruku@ttu.edu

***Abstract*— In-memory data storage systems have been proposed to keep primary data in memory to enable scalable, low-latency data access. Due to the high failure rate, most existing in-memory storage systems replicate data copies across storage servers. However, replication has a high storage overhead that could multiply the cost due to the higher cost compared to disk-based storage. There have been proposals to use erasure coding instead to have a better storage efficiency.**

Keywords- Redis, RAMCloud, MemEC, YCSB Benchmark, In-memory key value systems

I. Introduction

In-memory key-value (KV) stores are hugely used and deployed for their scalability, low-latency access. Some of the examples include MemEc, Redis, RAMCloud, Memcached. In distributed storage systems failures are prevalent, so we start using methods for effective retrieval and storage like replication in DRAM. As this has high storage overheads like HDD's, helps in replicas which has high latency for random I/O. In replication, each object in memory store multiple replicas, and one has to use many KV stores.

The main purpose of using Erasure Coding over Replication is because of its minimum data redundancy and when storing this redundant information in the memory there will be low latency access, helps in fast retrieval and recovery of data under failures.

There are few challenges when we use erasure coding - It's very expensive in updating the data and for failure's recovery. Small size objects are always dominant in in-memory storage workloads. In real-time Erasure coding is used for large objects. If the centralized meta data look up is not available, then In-memory KV stores issue decentralized requests. Erasure coding must maintain the data consistency in case when failures happen.

In this project, I installed and deployed some existing in-memory data storage systems Like Replication-can be performed by Redis and RAMCloud using benchmark tools YCSB. Whereas Erasure Coding- can be performed by memEC using benchmark YCSB. As well as we calculate the storage efficiency and performance of using these in memory data storage systems and predict which is better in performance.

II. REDIS

Redis (REmote Dictionary Server) is an in-memory, key value database, commonly referred to as an information structure server. One of the key contrasts between Redis and other key-value databases is Redis capacity to store and control high-level information sorts. These information sorts are principal information structures (records, maps, sets, and sorted sets) that most designers are recognizable with. Redis uncommon execution, straightforwardness, and nuclear control of information structures loans itself to understanding issues that are troublesome or perform ineffectively when actualized with conventional social databases

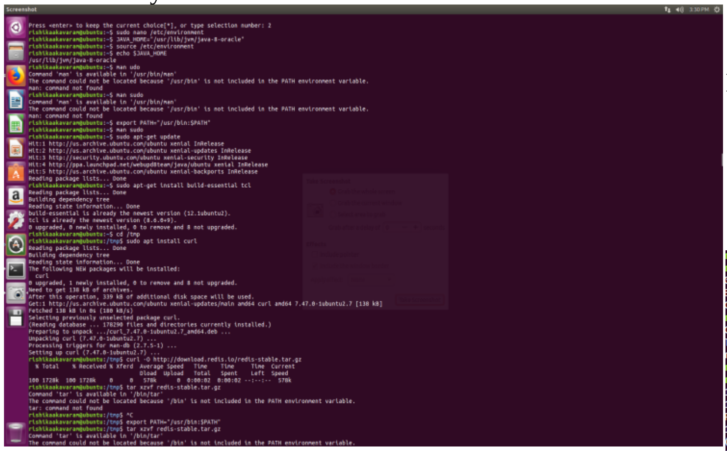
A. Install and Configure Redis:

Redis is an in-memory key-value store known for its adaptability, execution, and wide language support. In this direct, we are going illustrate how to introduce and arrange Redis.

1) Install the construct and test dependencies We can update our nearby apt package cache and introduce the conditions by
```
$sudo able -get update
$sudo able -get install build-essential tcl
```
2) Download and extract the source code
```
$cd /tmp
$sudo apt install curl
$curl -O http://download.redis.io/redis-stable.tar.gz
$tar xzvf redis-stable.tar.gz
$cd redis-stale
```
3) Build and introduce the redis $make
```
$sudo make install
$make test
```
Once the binaries are compiled, run the test suite will make beyond any doubt everything is build correctly. After the run is wrapped up, ready to introduce the doubles by
```
$sudo make install.
```
4) Configure Redis Creating a modern directory
```
$ sudo mkdir /etc/redis
```
Now, duplicate over the test Redis arrangement record included within the Redis source archive:
```
$sudo cp /tmp/redis-stable/redis.co
```
Next, open redis.conf to adjust the configuration:

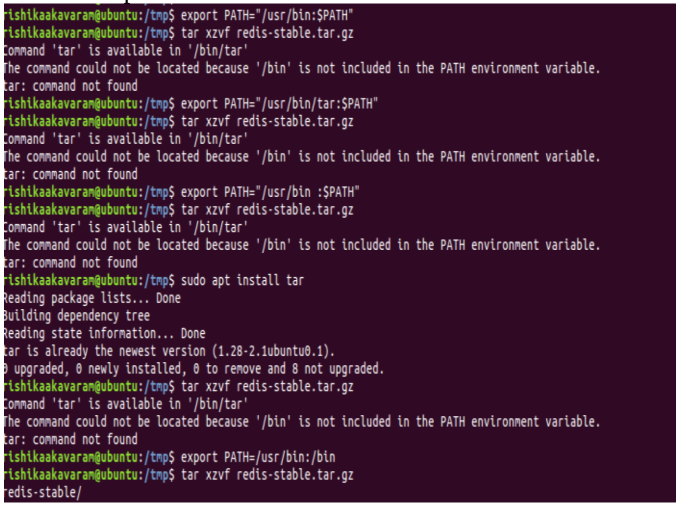
$sudo nano /etc/redis/redis.conf

one must set the number in the supervised directive of the system



```
Press <enter> to keep the current choice[*], or type selection number: 2
rishikaakavaram@ubuntu:~$ sudo nano /etc/environment
rishikaakavaram@ubuntu:~$ JAVA_HOME="/usr/lib/jvm/java-8-oracle"
rishikaakavaram@ubuntu:~$ source /etc/environment
rishikaakavaram@ubuntu:~$ echo $JAVA_HOME
/usr/lib/jvm/java-8-oracle
rishikaakavaram@ubuntu:~$ man udo
Command 'man' is available in '/usr/bin/man'
The command could not be located because '/usr/bin' is not included in the PATH environment variable.
man: command not found
rishikaakavaram@ubuntu:~$ man sudo
Command 'man' is available in '/usr/bin/man'
The command could not be located because '/usr/bin' is not included in the PATH environment variable.
man: command not found
rishikaakavaram@ubuntu:~$ export PATH="/usr/bin:$PATH"
rishikaakavaram@ubuntu:~$ man sudo
rishikaakavaram@ubuntu:~$ sudo apt-get update
Hit:1 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:3 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:4 http://ppa.launchpad.net/webupd8team/java/ubuntu xenial InRelease
Hit:5 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease
Reading package lists... Done
rishikaakavaram@ubuntu:~$ sudo apt-get install build-essential tcl
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.1ubuntu2).
tcl is already the newest version (8.6.0+9).
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
rishikaakavaram@ubuntu:~$ cd /tmp
rishikaakavaram@ubuntu:/tmp$ sudo apt install curl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 8 not upgraded.
Need to get 138 kB of archives.
After this operation, 339 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 curl amd64 7.47.0-1ubuntu2.7 [138 kB]
Fetched 138 kB in 0s (180 kB/s)
Selecting previously unselected package curl.
(Reading database ... 178290 files and directories currently installed.)
Preparing to unpack .../curl_7.47.0-1ubuntu2.7_amd64.deb ...
Unpacking curl (7.47.0-1ubuntu2.7) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up curl (7.47.0-1ubuntu2.7) ...
rishikaakavaram@ubuntu:/tmp$ curl -O http://download.redis.io/redis-stable.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 1728k  100 1728k    0     0   578k      0  0:00:02  0:00:02 --:--:--  578k
rishikaakavaram@ubuntu:/tmp$ tar xzvf redis-stable.tar.gz
Command 'tar' is available in '/bin/tar'
The command could not be located because '/bin' is not included in the PATH environment variable.
tar: command not found
rishikaakavaram@ubuntu:/tmp$ ^C
rishikaakavaram@ubuntu:/tmp$ export PATH="/usr/bin:$PATH"
rishikaakavaram@ubuntu:/tmp$ tar xzvf redis-stable.tar.gz
Command 'tar' is available in '/bin/tar'
The command could not be located because '/bin' is not included in the PATH environment variable.
```

In the directory dir., for the write permissions that aren't viewable by normal users, we can pick the location where we have those permissions and use dir/var/lib/redis



```
rishikaakavaram@ubuntu:/tmp$ export PATH="/usr/bin:$PATH"
rishikaakavaram@ubuntu:/tmp$ tar xzvf redis-stable.tar.gz
Command 'tar' is available in '/bin/tar'
The command could not be located because '/bin' is not included in the PATH environment variable.
tar: command not found
rishikaakavaram@ubuntu:/tmp$ export PATH="/usr/bin/tar:$PATH"
rishikaakavaram@ubuntu:/tmp$ tar xzvf redis-stable.tar.gz
Command 'tar' is available in '/bin/tar'
The command could not be located because '/bin' is not included in the PATH environment variable.
tar: command not found
rishikaakavaram@ubuntu:/tmp$ export PATH="/usr/bin :$PATH"
rishikaakavaram@ubuntu:/tmp$ tar xzvf redis-stable.tar.gz
Command 'tar' is available in '/bin/tar'
The command could not be located because '/bin' is not included in the PATH environment variable.
tar: command not found
rishikaakavaram@ubuntu:/tmp$ sudo apt install tar
Reading package lists... Done
Building dependency tree
Reading state information... Done
tar is already the newest version (1.28-2.1ubuntu0.1).
0 upgraded, 0 newly installed, 0 to remove and 8 not upgraded.
rishikaakavaram@ubuntu:/tmp$ tar xzvf redis-stable.tar.gz
Command 'tar' is available in '/bin/tar'
The command could not be located because '/bin' is not included in the PATH environment variable.
tar: command not found
rishikaakavaram@ubuntu:/tmp$ export PATH=/usr/bin:/bin
rishikaakavaram@ubuntu:/tmp$ tar xzvf redis-stable.tar.gz
redis-stable/
```

5) Creating a Redis system unit file

$sudo nano /etc/system/system/redis.service

$sudo nano /etc/redis/redis.conf

Redis-server binary has to be called to initiate the server, to the pointed configuration . Then we can execute the redis-cli and restart the sever one more time to check in case if the system has any failures.

[Unit] Description=Redis In-Memory Data Store
After=network.target
[Service]
User=redis
Group=redis
ExecStart=/usr/local/bin/redis-server /etc/redis/redis.conf
ExecStop=/usr/local/bin/redis-cli shutdown
Restart=always
[Install]
WantedBy=multi-user.target

6) Creating the Redis User, Group and Directories This command creates the redis user and group

$sudo adduser –system –group –no-create-home redis

$sudo mkdir /var/lib/redis

To give the ownership over the directory mention above, we have to create the redis user, group permissions.

$sudo chown redis:redis /var/lib/redis

$sudo chmod 770 /var/lib/redis

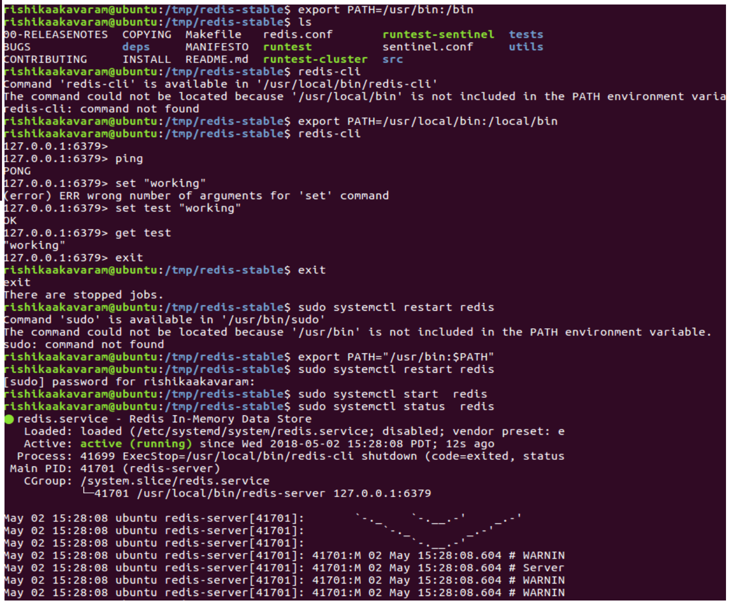
Enable Redis to start at the boot

$sudo systemctl enable redis

7) Start and Test Redis

$sudo systemctl start redis

$sudo systemctl status redis

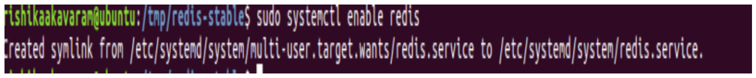


```
rishikaakavaram@ubuntu:/tmp/redis-stable$ export PATH=/usr/bin:/bin
rishikaakavaram@ubuntu:/tmp/redis-stable$ ls
00-RELEASENOTES  COPYING  Makefile   redis.conf       runtest-sentinel  tests
BUGS             deps     MANIFESTO  runtest          sentinel.conf     utils
CONTRIBUTING     INSTALL  README.md  runtest-cluster  src
rishikaakavaram@ubuntu:/tmp/redis-stable$ redis-cli
Command 'redis-cli' is available in '/usr/local/bin/redis-cli'
The command could not be located because '/usr/local/bin' is not included in the PATH environment varia
redis-cli: command not found
rishikaakavaram@ubuntu:/tmp/redis-stable$ export PATH=/usr/local/bin:/local/bin
rishikaakavaram@ubuntu:/tmp/redis-stable$ redis-cli
127.0.0.1:6379>
127.0.0.1:6379> ping
PONG
127.0.0.1:6379> set "working"
(error) ERR wrong number of arguments for 'set' command
127.0.0.1:6379> set test "working"
OK
127.0.0.1:6379> get test
"working"
127.0.0.1:6379> exit
rishikaakavaram@ubuntu:/tmp/redis-stable$ exit
exit
There are stopped jobs.
rishikaakavaram@ubuntu:/tmp/redis-stable$ sudo systemctl restart redis
Command 'sudo' is available in '/usr/bin/sudo'
The command could not be located because '/usr/bin' is not included in the PATH environment variable.
sudo: command not found
rishikaakavaram@ubuntu:/tmp/redis-stable$ export PATH="/usr/bin:$PATH"
rishikaakavaram@ubuntu:/tmp/redis-stable$ sudo systemctl restart redis
[sudo] password for rishikaakavaram:
rishikaakavaram@ubuntu:/tmp/redis-stable$ sudo systemctl start  redis
rishikaakavaram@ubuntu:/tmp/redis-stable$ sudo systemctl status  redis
● redis.service - Redis In-Memory Data Store
   Loaded: loaded (/etc/systemd/system/redis.service; disabled; vendor preset: e
   Active: active (running) since Wed 2018-05-02 15:28:08 PDT; 12s ago
  Process: 41699 ExecStop=/usr/local/bin/redis-cli shutdown (code=exited, status
 Main PID: 41701 (redis-server)
   CGroup: /system.slice/redis.service
           └─41701 /usr/local/bin/redis-server 127.0.0.1:6379

May 02 15:28:08 ubuntu redis-server[41701]:       `-._    `-.__.-'    _.-'
May 02 15:28:08 ubuntu redis-server[41701]:           `-._        _.-'
May 02 15:28:08 ubuntu redis-server[41701]:               `-.__.-'
May 02 15:28:08 ubuntu redis-server[41701]: 41701:M 02 May 15:28:08.604 # WARNIN
May 02 15:28:08 ubuntu redis-server[41701]: 41701:M 02 May 15:28:08.604 # Server
May 02 15:28:08 ubuntu redis-server[41701]: 41701:M 02 May 15:28:08.604 # WARNIN
May 02 15:28:08 ubuntu redis-server[41701]: 41701:M 02 May 15:28:08.604 # WARNIN
```

8) At the boot, enable Redis to start

$sudo systemctl enable redis



```
rishikaakavaram@ubuntu:/tmp/redis-stable$ sudo systemctl enable redis
Created symlink from /etc/systemd/system/multi-user.target.wants/redis.service to /etc/systemd/system/redis.service.
```

III. RAMCLOUD

RAMCloud could be a unused lesson of super-high-speed capacity for large-scale datacenter applications. It is planned for applications in which a huge number of servers in a datacenter require low-latency get to a huge tough datastore. RAMCloud is more powerful as this data model is used in a large scale and known for its low latency.

1. Installing and Configuring RAMCloud:

For installing, clone the RAMCloud repository and start compiling it. If any errors are present by using git commit fix all the bugs. To configure the cluster,one need to set up three kinds of servers:

Storage servers- for implementing the ramcloud functionality. theses servers contain a master that store part of RAMCloud key-value store and manages redundant data on disk.

Coordinator- At any given time there is one machine filling in as group facilitator. This machine oversees general group arrangement data, and it organizes recuperation when capacity

servers crash. One will regularly begin 2-3 occurrences of the organizer.
External storage: It is used for storing higher level information of the cluster and its configuration.

```
$git clone git@github.com:PlatformLab/RAMCloud.git
$cd RAMCloud
$git submodule update –init –recursive
$ln -s ../../hooks/pre-commit .git/hooks/pre-commit
```

2. Compiling

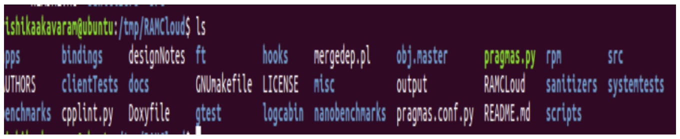
```
$cd RAMCloud $make -j12
```

3. Testing Make Build- Build the Server and the software for the client Make tests- especially unit tests are done by building and running the RAMCloud
4. Cppunit

```
$make
$make test
$sudo apt-get install Cppunit
$make tests cppunit
```

5. Output

```
$Obj.master/ client/client
$obj.master/server/server
```



```
ishikaakavaram@ubuntu:/tmp/RAMCloud$ ls
pps       bindings    designNotes ft         hooks    mergedep.pl    obj.master      pragmas.py rpm        src
UTHORS    clientTests docs        GNUmakefile LICENSE  misc           output          RAMCloud   sanitizers systemtests
enchmarks cpplint.py  Doxyfile    gtest      logcabin nanobenchmarks pragmas.conf.py README.md  scripts
```

IV. MemEc

A. *Erasure coding based in memory systems aims for fast*

recovery, low latency, and it has to be storage efficient. To implement this in memory system we have to build MemEc. This can be done by installing the in-memory system and evaluating the performance on YCSB workloads.

- MemEC is specifically designed for workloads dominated by small objects. By encoding objects in entirety, MemEC is shown to incur 60% small objects than existing replication and erasure-coding based approaches. It also supports graceful transitions between decentralized requests in normal mode (i.e., no failures) and coordinated requests in degraded mode (i.e.,with failures).
- We evaluate our MemEC prototype via testbed experiments under read-heavy and update-heavy YCSB workloads.
- We show that MemEC achieves high throughput and low latency in both normal and degraded modes, and support fast transitions between the two modes.
  keep primary data in memory to enable scalable, low latency data access. They are often realized as in-memory key value (KV) stores, which organize data in memory as KV pairs (called objects) to form structured storage.
- Enterprises have deployed in-memory KV stores for low-latency operations in social networking, web search, and analytics Failures are prevalent in distributed storage.

B. Installation and configuration of MemEc:

- Build-essential:

```
$ sudo apt-get install build-essential
$ ./configure
$ make
$ sudo make install
```

Install required packages:

```
$sudo apt-get install automake yasm libtool
```

- Check the version of yasm

```
$wget
$tar zxvf yasm-1.2.0.tar.gz
$cd yasm-1.2.0
$make
$sudo make install
$tar zxf autoconf-2.69.tar.gz
$cd autoconf-2.69
$make
```

- Configure and compile ISA-L under lib/isa-l-2.14.0/: ./configure --prefix=$(pwd) and

```
$make install
```

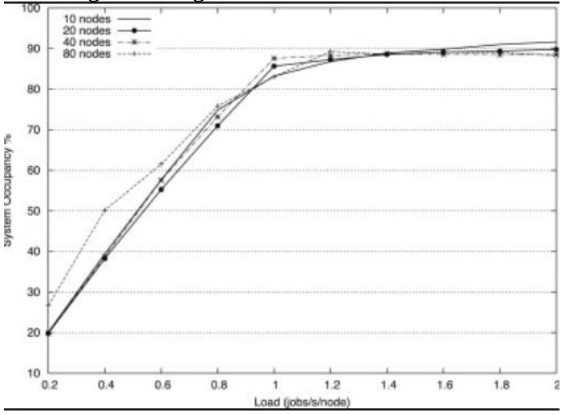
Add libtool support:

```
$ libtoolize
```

Continue to install:

```
$ ./configure --prefix=$(pwd); make install
```

Performance characters when runs on a local machine, over YCSB benchmark, the graph generated between latency and throughput results in higher throughput and lower latency. That's one advantage of using MemEC



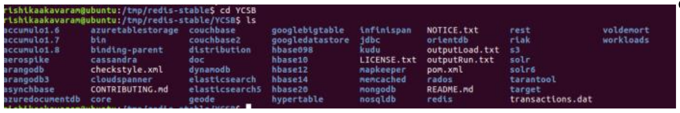
V. Benchmarking using YCSB

1. YCSB is the Yahoo Cloud Serving Benchmark server which is an open source specification and program suite for retrieval and maintenance capabilities of computer programs. Its often used to compare

relative performance of other database management systems.

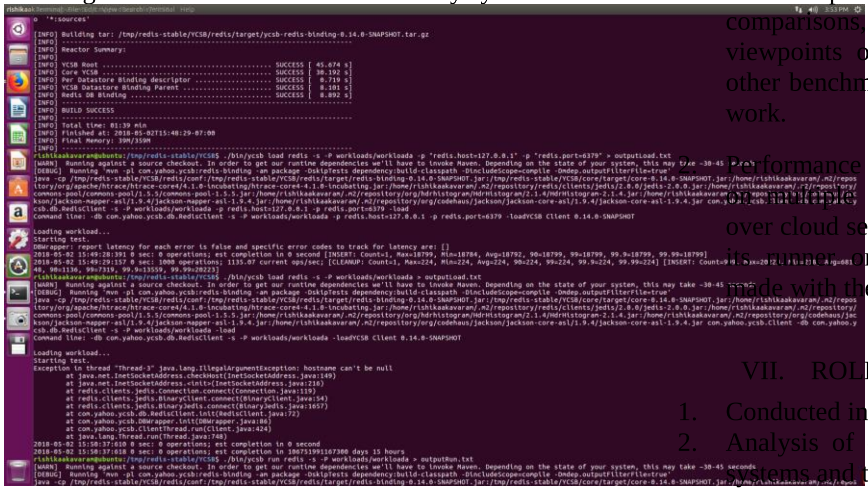
2. To use the Benchmark tool on in-memory systems, we need to follow few steps

- When calculating the performance, we have to start the Redis or RAMCloud or MemEC in memory storage systems
- Check whether Java and Maven are installed or not
- Then setting up YCSB by
  $sudo apt-get install jdk
  $sudo apt-get install jre
- $java –version
- $sudo apt install git
- Git clone YCSB
- $git clone http://githhub.com/brianfrankcooper/YCSB.git
  $cd YCSB
  $mvn -pl com.yahoo.ycsb:redis-binding -am clean package
- Provide Redis Connection Parameters
- Set the host,port and password in the workload, set configure
- $./bin/ycsb load redis -s -P workloads/workloada -p “redis.host=127.0.0.1” -p “redis.port=6379” >outputLoad.txt
- Load and run the tests
  $./bin/ycsb load redis -s -P workloads/workloada > outputLoad.txt
  $./bin/ycsb load redis -s -P workloads/workloada>outputRun.txt
  List of files in YCSB are mentioned below



```
rishikaakavaram@ubuntu:/tmp/redis-stable$ cd YCSB
rishikaakavaram@ubuntu:/tmp/redis-stable/YCSB$ ls
accumulo1.6     azuretablestorage  couchbase        googlebigtable   infinispan   NOTICE.txt       rest              voldemort
accumulo1.7     bin                couchbase2       googledatastore  jdbc         orientdb         riak              workloads
accumulo1.8     binding-parent     distribution     hbase098         kudu         outputLoad.txt   s3
aerospike       cassandra          doc              hbase10          LICENSE.txt  outputRun.txt    solr
arangodb        checkstyle.xml     dynamodb         hbase12          mapkeeper    pom.xml          solr6
arangodb3       cloudspanner       elasticsearch    hbase14          memcached    rados            tarantool
asynchbase      CONTRIBUTING.md    elasticsearch5   hbase20          mongodb      README.md        target
azuredocumentdb core               geode            hypertable       nosqldb      redis            transactions.dat
```

Running of YCSB over various in memory systems



```
o  '*:sources'

[INFO] Building tar: /tmp/redis-stable/YCSB/redis/target/ycsb-redis-binding-0.14.0-SNAPSHOT.tar.gz
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] YCSB Root .......................................... SUCCESS [ 45.674 s]
[INFO] Core YCSB .......................................... SUCCESS [ 30.192 s]
[INFO] Per Datastore Binding descriptor ................... SUCCESS [  0.719 s]
[INFO] YCSB Datastore Binding Parent ...................... SUCCESS [  8.101 s]
[INFO] Redis DB Binding ................................... SUCCESS [  8.892 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 01:39 min
[INFO] Finished at: 2018-05-02T15:48:29-07:00
[INFO] Final Memory: 39M/359M
[INFO] ------------------------------------------------------------------------
rishikaakavaram@ubuntu:/tmp/redis-stable/YCSB$ ./bin/ycsb load redis -s -P workloads/workloada -p "redis.host=127.0.0.1" -p "redis.port=6379" > outputLoad.txt
[WARN]  Running against a source checkout. In order to get our runtime dependencies we'll have to invoke Maven. Depending on the state of your system, this may take ~30-45 seconds
[DEBUG]  Running 'mvn -pl com.yahoo.ycsb:redis-binding -am package -DskipTests dependency:build-classpath -DincludeScope=compile -Dmdep.outputFilterFile=true'
java -cp /tmp/redis-stable/YCSB/redis/conf:/tmp/redis-stable/YCSB/redis/target/redis-binding-0.14.0-SNAPSHOT.jar:/tmp/redis-stable/YCSB/core/target/core-0.14.0-SNAPSHOT.jar:/home/rishikaakavaram/.m2/repos
itory/org/apache/htrace/htrace-core4/4.1.0-incubating/htrace-core4-4.1.0-incubating.jar:/home/rishikaakavaram/.m2/repository/redis/clients/jedis/2.0.0/jedis-2.0.0.jar:/home/rishikaakavaram/.m2/repository/
commons-pool/commons-pool/1.5.5/commons-pool-1.5.5.jar:/home/rishikaakavaram/.m2/repository/org/hdrhistogram/HdrHistogram/2.1.4/HdrHistogram-2.1.4.jar:/home/rishikaakavaram/.m2/repository/org/codehaus/jac
kson/jackson-mapper-asl/1.9.4/jackson-mapper-asl-1.9.4.jar:/home/rishikaakavaram/.m2/repository/org/codehaus/jackson/jackson-core-asl/1.9.4/jackson-core-asl-1.9.4.jar com.yahoo.ycsb.Client -db com.yahoo.y
csb.db.RedisClient -s -P workloads/workloada -p redis.host=127.0.0.1 -p redis.port=6379 -load
Command line: -db com.yahoo.ycsb.db.RedisClient -s -P workloads/workloada -p redis.host=127.0.0.1 -p redis.port=6379 -loadYCSB Client 0.14.0-SNAPSHOT

Loading workload...
Starting test.
DBWrapper: report latency for each error is false and specific error codes to track for latency are: []
2018-05-02 15:49:28:391 0 sec: 0 operations; est completion in 0 second [INSERT: Count=1, Max=18799, Min=18784, Avg=18792, 90=18799, 99=18799, 99.9=18799, 99.99=18799]
2018-05-02 15:49:29:157 0 sec: 1000 operations; 1135.07 current ops/sec; [CLEANUP: Count=1, Max=224, Min=224, Avg=224, 90=224, 99=224, 99.9=224, 99.99=224] [INSERT: Count=999, Max=20223, Min=216, Avg=681.
48, 90=1136, 99=7319, 99.9=13559, 99.99=20223]
rishikaakavaram@ubuntu:/tmp/redis-stable/YCSB$ ./bin/ycsb load redis -s -P workloads/workloada > outputLoad.txt
[WARN]  Running against a source checkout. In order to get our runtime dependencies we'll have to invoke Maven. Depending on the state of your system, this may take ~30-45 seconds
[DEBUG]  Running 'mvn -pl com.yahoo.ycsb:redis-binding -am package -DskipTests dependency:build-classpath -DincludeScope=compile -Dmdep.outputFilterFile=true'
java -cp /tmp/redis-stable/YCSB/redis/conf:/tmp/redis-stable/YCSB/redis/target/redis-binding-0.14.0-SNAPSHOT.jar:/tmp/redis-stable/YCSB/core/target/core-0.14.0-SNAPSHOT.jar:/home/rishikaakavaram/.m2/repos
itory/org/apache/htrace/htrace-core4/4.1.0-incubating/htrace-core4-4.1.0-incubating.jar:/home/rishikaakavaram/.m2/repository/redis/clients/jedis/2.0.0/jedis-2.0.0.jar:/home/rishikaakavaram/.m2/repository/
commons-pool/commons-pool/1.5.5/commons-pool-1.5.5.jar:/home/rishikaakavaram/.m2/repository/org/hdrhistogram/HdrHistogram/2.1.4/HdrHistogram-2.1.4.jar:/home/rishikaakavaram/.m2/repository/org/codehaus/jac
kson/jackson-mapper-asl/1.9.4/jackson-mapper-asl-1.9.4.jar:/home/rishikaakavaram/.m2/repository/org/codehaus/jackson/jackson-core-asl/1.9.4/jackson-core-asl-1.9.4.jar com.yahoo.ycsb.Client -db com.yahoo.y
csb.db.RedisClient -s -P workloads/workloada -load
Command line: -db com.yahoo.ycsb.db.RedisClient -s -P workloads/workloada -loadYCSB Client 0.14.0-SNAPSHOT

Loading workload...
Starting test.
Exception in thread "Thread-3" java.lang.IllegalArgumentException: hostname can't be null
        at java.net.InetSocketAddress.checkHost(InetSocketAddress.java:149)
        at java.net.InetSocketAddress.<init>(InetSocketAddress.java:216)
        at redis.clients.jedis.Connection.connect(Connection.java:119)
        at redis.clients.jedis.BinaryClient.connect(BinaryClient.java:54)
        at redis.clients.jedis.BinaryJedis.connect(BinaryJedis.java:1657)
        at com.yahoo.ycsb.db.RedisClient.init(RedisClient.java:72)
        at com.yahoo.ycsb.DBWrapper.init(DBWrapper.java:86)
        at com.yahoo.ycsb.ClientThread.run(Client.java:424)
        at java.lang.Thread.run(Thread.java:748)
2018-05-02 15:50:37:610 0 sec: 0 operations; est completion in 0 second
2018-05-02 15:50:37:618 0 sec: 0 operations; est completion in 106751991167300 days 15 hours
rishikaakavaram@ubuntu:/tmp/redis-stable/YCSB$ ./bin/ycsb run redis -s -P workloads/workloada > outputRun.txt
[WARN]  Running against a source checkout. In order to get our runtime dependencies we'll have to invoke Maven. Depending on the state of your system, this may take ~30-45 seconds
[DEBUG]  Running 'mvn -pl com.yahoo.ycsb:redis-binding -am package -DskipTests dependency:build-classpath -DincludeScope=compile -Dmdep.outputFilterFile=true'
java -cp /tmp/redis-stable/YCSB/redis/conf:/tmp/redis-stable/YCSB/redis/target/redis-binding-0.14.0-SNAPSHOT.jar:/tmp/redis-stable/YCSB/core/target/core-0.14.0-SNAPSHOT.jar:/home/rishikaakavaram/.m2/repos
```

V CONCLUSIONS

1. This projects explain the need of erasure coding to small objects presenting MemEC in-memory KV storage. It’s accurate from the paper that this data mode reduces the fault tolerance by up to 60% in comparison with the existing data models like replication or hybrid data model.
2. The performance analysis of MemEC is stable in both normal and degraded modes.
3. We evaluate the prototype for Redis and RAMCloud system running on local machine with multiple nodes and tested the performances using YCSB benchmark and calculated the performance in form of graphs.
4. Following is the source code for installing MemEC https://github.com/mtyiu/memec
5. Observations include, On comparing Redis and RAMCloud, Redis is faster in Read/Write speed, as well as memory usage of redis is better as it doesn’t use beyond its limit and what ever its not using is returned back. RAMCloud is equally fast but RAMcloud takes the full chunk of data but it has lesser latency. Whereas Redis can act cache as well.
6. So, the performance analysis observed between Redis and MemEC. MemEC uses less disk seeks than any replicated systems as MemEC achieves high throughput and low latency, supporting the very fast transitions between two modes.
7. Most of the enterprises are moving to erasure coding because of its cost saving and the property of reducing overheads.
8. Replication mode is better for read(throughput), while erasure coding is better for write throughput and its smaller storage overhead.

VI. FUTURE WORK

1. In the aspect of YCSB, expansion to performance comparisons, it is imperative to look at other viewpoints of cloud serving frameworks. we have other benchmark levels which can be created in later work.

2. Performance calculation of MemEc can be evaluated on multiple nodes as mentioned in research paper over cloud server under YCSB benchmarking. (So far its runner on local machine and comparisons are made with the replication data models)

VII. ROLES AND RESPONSIBILITIES

1. Conducted initial research about the project topic.
2. Analysis of different in-memory key value storage systems and their findings.
3. Installing the different versions of data storage servers to evaluate the performances over benchmarking tool YCSB.
4. Implementation of MemEc on YCSB and comparing it with Redis and RAMCloud storage servers

5. Presentation slides and report writing.

ACKNOWLEDGMENT

This research paper was published as a part of Course Project for the Advanced Operating Systems Design class in Spring 2018. We thank Dr. Yong Chen, Assistant Professor, Department of Computer Science, Texas Tech University, for offering this project and guiding us throughout our work.

We would also like to thank our supervisor, Mr.Wei Zie, PhD Student, Texas Tech University, for his patient guidance, encouragement and advice he has provided throughout the commencement of the project, for his undeniable support in maintaining and resolving issues with the HPCC server for our project execution.

REFERENCES

[1] Matt M. T. Yiu, Helen H. W. Chan, and Patrick P. C. Lee, SYSTOR'17, May 22-24, 2017, Haifa, Israel c 2017 ACM. ISBN 978-1-4503-5035-8/17/05, DOI: http://dx.doi.org/10.1145/3078468.3078470 Chris, https://github.com/serendependy/parallel-j/blob/master/OpenMP/mergeSort-omp.c

[2] Mtyiu, memec, https://github.com/mtyiu/memec

[3] https://redis.io/topics/quickstart

[4] https://ramcloud.atlassian.net/wiki/spaces/RAM/pages/6848532/Setting+Up+a+RAMCloud+Cluster

[5] https://github.com/brianfrankcooper/YCSB

[6] file:///C:/Users/rakavara/Downloads/ReadingList/ReadingList/ramcloud.pdf

[7] Key-EC-Cache: Load-Balanced, Low-Latency Cluster Caching with Online Erasure Coding, K. V. Rashmi, Mosharaf Chowdhury, Jack Kosaian, Kannan Ramchandran, Ion Stoica, November 2–4, 2016 ,Savannah, GA, USA ISBN 978-1-931971-33-1

[8] Benchmarking Cloud Serving Systems with YCSB

[9] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, Russell Sears, Benchmarking Cloud Serving Systems with YCSB SoCC'10, June10–11, 2010, Indianapolis, Indiana, USA. Copyright 2010ACM 978-1-4503-0036-0/10/06

[10] Memcached. http://memcached.org.

[11] Redis. http://redis.io.

[12] Redis Latency Problems Troubleshotting. http://redis. io/topics/latency.