

CSCI E63 - Final Project

Redis - No SQL Database

Table of Contents

Redis - NoSQL Database	2
Objectives	2
Redis (Remote Dictionary Server)	2
Super Quick Installation	3
Redis Cluster Installation (3 EC2 Instances)	5
Master Slave Replication 101	16
MovieLens Dataset - Load Users in Cluster	18
MovieLensJedisQuery	22
Mass Insertion Protocol (Pipeline)	25
YouTube Video Links	29
References	29



redis

Redis - NoSQL Database

Objectives

- Install **Redis Cluster** 3.0.0 (Released April 2015) on AWS EC2 (3 Node cluster) and configure a **Master Slave replication**.
- Read MovieLens dataset and store it in REDIS (Sets) - Run queries - Using Java Client.
- REDIS Mass Insertion Protocol (with Redis Pipeline) - 21 Million Rows Bulk Insert

Redis (Remote Dictionary Server)

- Advanced Key Value Cache and Store (Persistence)
- Fast (In Memory)
- Open Source (BSD Licensed)
- Simple and Flexible
- Currently most popular key-value database

51 systems in ranking, May 2015

Rank	May 2015	Apr 2015	May 2014	DBMS	Database Model	Score		
						May 2015	Apr 2015	May 2014
1.	1.	1.	1.	Redis	Key-value store	94.73	+0.17	+32.69
2.	2.	2.	2.	Memcached	Key-value store	33.11	-0.93	+1.37
3.	3.	↑ 4.	4.	Amazon DynamoDB	Multi-model	14.87	+0.29	+5.60
4.	4.	↓ 3.	3.	Riak	Key-value store	12.89	-0.20	+2.36
5.	5.	5.	5.	Ehcache	Key-value store	7.92	+0.21	+1.47
6.	6.	6.	6.	Hazelcast	Key-value store	5.82	+0.14	+1.83
7.	↑ 8.	↑ 10.	10.	OrientDB	Multi-model	3.99	+0.62	+2.31
8.	↓ 7.	↓ 7.	7.	Berkeley DB	Key-value store	3.60	-0.32	+0.53
9.	9.	9.	9.	Oracle Coherence	Key-value store	3.30	-0.03	+0.72
10.	10.	↓ 8.	8.	Amazon SimpleDB	Key-value store	2.96	-0.24	+0.00

<http://db-engines.com/en/ranking/key-value+store>

Super Quick Installation

(Follow this if you just want to get started, Redis Cluster on EC2 - see next page)

Installation (On Mac / *nix)

-Couldn't be any simpler (<http://redis.io/download>) :

```
$ wget http://download.redis.io/releases/redis-3.0.0.tar.gz  
$ tar xzf redis-3.0.0.tar.gz  
$ cd redis-3.0.0  
$ make
```

```
$ src/redis-server  
$ src/redis-cli  
redis> set foo bar  
OK  
redis> get foo  
"bar"
```

Windows - see <http://redis.io/download> and <https://github.com/MSOpenTech/redis>

Redis Cluster (Redis Version 3.0.0 - Released April 2015)

Redis Cluster 101

- Redis Cluster provides a way to run a Redis installation where data is automatically sharded across multiple Redis nodes.
- Redis Cluster has the ability to automatically split your dataset among multiple nodes.
- Redis Cluster has the ability to continue operations when a subset of the nodes are experiencing failures or are unable to communicate with the rest of the cluster.

Installation Overview

- 3 EC2 Instances + 1 EC2 Instances for Master Slave
- Do the following on all 3 Instances (We'll setup master slave later)
- Download and Extract Redis
- Install build-essential, make
- Make Redis libraries
- Install tk8.5 tcl8.5 - (C functions/Libraries)
- Update configuration (redis.conf)
- Start Redis Server
- Install Ruby and Ruby Redis gems
- Create the cluster and test it!!

Redis Cluster Installation (3 EC2 Instances)

Redis Cluster - Redis Cluster is the mechanism to distribute shards to multiple Redis instances

(Before moving on make sure you have a Key Pair set up)

Watch Video here -> https://www.youtube.com/watch?v=s4YpCA2Y_-Q

Step 1: AWS EC2 -> Instances -> Launch an Instance -> Select Ubuntu Server 14.04



Step 2: Choose an Instance Type

Select Memory Optimized r3.large
(t2.micro works well too)

Step 3: Configure Instance Details

Select Number of Instances: 4 (*We will use 1 for Master Slave replication*)

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

This screenshot shows the "Configure Instance Details" step of the AWS EC2 instance creation wizard. It includes fields for the number of instances (set to 4), purchasing options (Request Spot Instances checked), network settings (VPC, Subnet, Auto-assign Public IP, Placement group), IAM role (None), shutdown behavior (Stop), termination protection (unchecked), monitoring (unchecked), and tenancy (Shared tenancy). At the bottom, there are "Cancel", "Previous", "Review and Launch" (highlighted in blue), and "Next: Add Storage" buttons.

Step:4 Configure Security Group

Type <i>i</i>	Protocol <i>i</i>	Port Range <i>i</i>	Source <i>i</i>
SSH	TCP	22	0.0.0.0/0
All TCP	TCP	0 - 65535	0.0.0.0/0
Custom TCP Rule	TCP	6379	0.0.0.0/0
HTTP	TCP	80	0.0.0.0/0

Review and Launch (Use the existing key pair)

- Rename the instances as RedisClusterNode1, RedisClusterNode2 and RedisClusterNode3

The screenshot shows the AWS EC2 Instances page. At the top, there are buttons for 'Launch Instance', 'Connect', and 'Actions'. A search bar is present above the table. The table lists four instances:

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
RedisClusterNode1	i-adf4d250	r3.xlarge	us-east-1b	running	Initializing	None	ec2-52-6-250-55.compute-1.amazonaws.com	52.6.250.5
RedisClusterNode2	i-acf4d251	r3.xlarge	us-east-1b	running	Initializing	None	ec2-52-7-60-17.compute-1.amazonaws.com	52.7.60.17
RedisClusterNode3	i-aef4d253	r3.xlarge	us-east-1b	running	Initializing	None	ec2-52-6-251-225.compute-1.amazonaws.com	52.6.251.2
RedisClusterSLAVE	i-a3f4d25e	r3.xlarge	us-east-1b	running	Initializing	None	ec2-52-7-67-100.compute-1.amazonaws.com	52.7.67.10

Below the table, the instance details for RedisClusterNode3 are shown:

Description	Status Checks	Monitoring	Tags
Instance ID	i-aef4d253	Public DNS	ec2-52-6-251-225.compute-1.amazonaws.com
Instance state	running	Public IP	52.6.251.225
Instance type	r3.xlarge	Elastic IP	-
Private DNS	ip-172-31-41-100.ec2.internal	Availability zone	us-east-1b
Private IPs	172.31.41.100	Security groups	RedisSG. view rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-53b1c836	AMI ID	ubuntu-trusty-14.04-amd64-server-20150325 (ami-d05e75b8)
Subnet ID	subnet-63920f14	Platform	-
Network interfaces	eth0	IAM role	-
Source/dest. check	True	Key pair name	HadoopEC2Cluster
		Owner	744940782804

Lets fill up the following table:

	Public IP	Private IP
RedisClusterNode1	52.6.250.55	172.31.41.98
RedisClusterNode2	52.7.60.17	172.31.41.99
RedisClusterNode3	52.6.251.225	172.31.41.100

SSH to all three instances

```
ssh -i HadoopEC2Cluster.pem ubuntu@ec2-52-6-250-55.compute-1.amazonaws.com
ssh -i HadoopEC2Cluster.pem ubuntu@ec2-52-7-60-17.compute-1.amazonaws.com
ssh -i HadoopEC2Cluster.pem ubuntu@ec2-52-6-251-225.compute-1.amazonaws.com
```

The image shows three terminal windows side-by-side. The leftmost window is on an Ubuntu 14.04.2 LTS machine (IP 172.31.41.98) and displays the system status command output. The middle window is on another Ubuntu 14.04.2 LTS machine (IP 172.31.41.99) and also shows system status. The rightmost window is on a Mac OS X system (Rashmi's MacBook Pro) and shows the Redis build command being run.

```
* Documentation: https://help.ubuntu.com/
System information as of Thu May 7 11:41:19 UTC 2015
System load: 0.08      Memory usage: 0%   Processes: 101
Usage of /: 9.8% of 7.74GB Swap usage: 0%   Users logged in: 0
Graph this data and manage this system at:
https://landscape.canonical.com/
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@ip-172-31-41-98:~$
```

```
* Documentation: https://help.ubuntu.com/
System information as of Thu May 7 11:41:18 UTC 2015
System load: 0.32      Memory usage: 0%   Processes: 101
Usage of /: 9.8% of 7.74GB Swap usage: 0%   Users logged in: 0
Graph this data and manage this system at:
https://landscape.canonical.com/
Get cloud support with Ubuntu Advantage Cloud Guest:
http://www.ubuntu.com/business/services/cloud
0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@ip-172-31-41-100:~$
```

```
Rashmis-MacBook-Pro:week9 rashmi$
```

Download and Extract Redis on all 3 instances

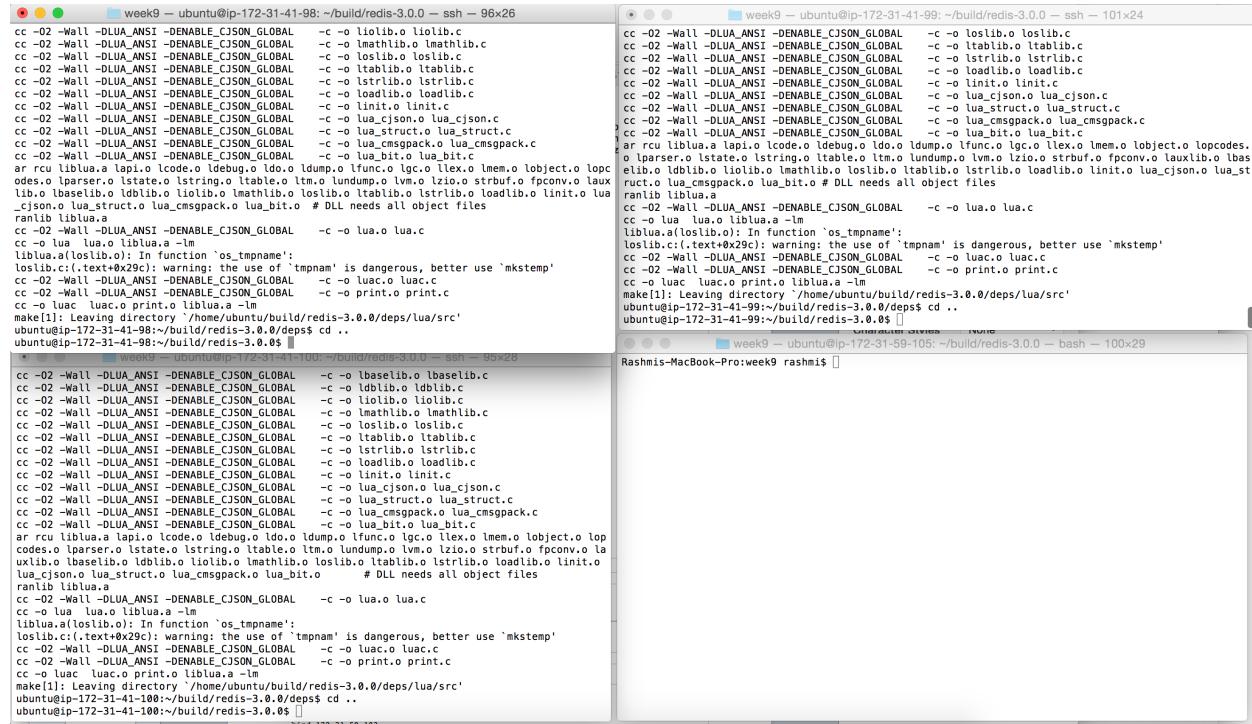
```
mkdir build && cd build
wget http://download.redis.io/releases/redis-3.0.0.tar.gz
tar -xvzf redis-3.0.0.tar.gz
cd redis-3.0.0/
```

Now we can build it

```
sudo apt-get install -y make gcc build-essential
sudo apt-get update
sudo apt-get install make
sudo apt-get install gcc
cd deps
make hiredis jemalloc linenoise lua
cd ..
```

(Watch for any error messages in any of the above steps)

Now my shell looks like this:



```
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lio/lib.o lio/lib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lmthlib.o lmthlib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o loslib.o loslib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o ltlib.o ltlib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lstrlib.o lstrlib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o loadlib.o loadlib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o linit.o linit.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lua_cjson.o lua_cjson.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lua_struct.o lua_struct.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lua_bit.o lua_bit.c
ar rrcu liblua.a lapi.o lcode.o ldebug.o ldo.o ldump.o lfunc.o lgc.o llex.o lmem.o lobject.o lop
odes.o lparser.o lstate.o lstring.o ltable.o ltm.o lundump.o lvm.o lzio.o strbuf.o fpconv.o lau
lib.o lbaselib.o ldblib.o liolib.o lmthlib.o loslib.o ltlib.o lstrlib.o loadlib.o linit.o lu
c_json.o lua_struct.o lua_cmsgpack.o lua_bit.o # DLL needs all object files
ranlib liblua.a
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lua.o lua.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lua_bit.o lua_bit.c
liblua.aloslib.o: In function `os_tmpname':
liblua.c:(.text+0x29c): warning: the use of `tmpnam' is dangerous, better use `mkstemp'
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o luac.o luac.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o print.o print.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o luaL.o luaL.c
make[1]: Leaving directory '/home/ubuntu/build/redis-3.0.0deps/luasrc'
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0deps$ cd ..
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$ make[1]: Leaving directory '/home/ubuntu/build/redis-3.0.0deps/luasrc'
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$ cd ..
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$
```



```
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lbaselib.o lbaselib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o ldblib.o ldblib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o liolib.o liolib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lmthlib.o lmthlib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o loslib.o loslib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o ltlib.o ltlib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lstrlib.o lstrlib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o loadlib.o loadlib.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o linit.o linit.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lua_cjson.o lua_cjson.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lua_struct.o lua_struct.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lua_bit.o lua_bit.c
ar rrcu liblua.a lapi.o lcode.o ldebug.o ldo.o ldump.o lfunc.o lgc.o llex.o lmem.o lobject.o lop
odes.o lparser.o lstate.o lstring.o ltable.o ltm.o lundump.o lvm.o lzio.o strbuf.o fpconv.o lau
lib.o lbaselib.o ldblib.o liolib.o lmthlib.o loslib.o ltlib.o lstrlib.o loadlib.o linit.o lu
c_json.o lua_struct.o lua_cmsgpack.o lua_bit.o # DLL needs all object files
ranlib liblua.a
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lua.o lua.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o lua_bit.o lua_bit.c
liblua.aloslib.o: In function `os_tmpname':
liblua.c:(.text+0x29c): warning: the use of `tmpnam' is dangerous, better use `mkstemp'
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o luac.o luac.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o print.o print.c
cc -O2 -Wall -DLUA_ANSI -DENABLE_CJSON_GLOBAL -c -o luaL.o luaL.c
make[1]: Leaving directory '/home/ubuntu/build/redis-3.0.0deps/luasrc'
ubuntu@ip-172-31-41-99:~/build/redis-3.0.0$ cd ..
ubuntu@ip-172-31-41-99:~/build/redis-3.0.0$ make[1]: Leaving directory '/home/ubuntu/build/redis-3.0.0deps/luasrc'
ubuntu@ip-172-31-41-99:~/build/redis-3.0.0$ bash
```

Now Run the following command:

```
make MALLOC=libc
```

The image displays four terminal windows side-by-side, each showing the output of a Redis build process. The top-left window shows the build on an Ubuntu system (ip-172-31-41-98). The top-right window shows the build on another Ubuntu system (ip-172-31-41-99). The bottom-left window shows the build on a third Ubuntu system (ip-172-31-41-100). The bottom-right window shows the build on a Mac OS X system (Rashmi's MacBook Pro).

Terminal 1 (Ubuntu ip-172-31-41-98):

```
CC memtest.o  
CC crc64.o  
CC bitops.o  
CC sentinel.o  
CC notify.o  
CC setproctitle.o  
CC blocked.o  
CC hyperloglog.o  
CC latency.o  
CC sparkline.o  
LINK redis-server  
INSTALL redis-sentinel  
CC redis-cli.o  
LINK redis-cli  
CC redis-benchmark.o  
LINK redis-benchmark  
CC redis-check-dump.o  
LINK redis-check-dump  
CC redis-check-aof.o  
LINK redis-check-aof  
LINK redis-check-aof  
  
Hint: It's a good idea to run 'make test' ;)  
make[1]: Leaving directory '/home/ubuntu/build/redis-3.0.0/src'  
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$
```

Terminal 2 (Ubuntu ip-172-31-41-99):

```
CC crc64.o  
CC bitops.o  
CC sentinel.o  
CC notify.o  
CC setproctitle.o  
CC blocked.o  
CC hyperloglog.o  
CC latency.o  
CC sparkline.o  
LINK redis-server  
INSTALL redis-sentinel  
CC redis-cli.o  
LINK redis-cli  
CC redis-benchmark.o  
LINK redis-benchmark  
CC redis-check-dump.o  
LINK redis-check-dump  
CC redis-check-aof.o  
LINK redis-check-aof  
  
Hint: It's a good idea to run 'make test' ;)  
make[1]: Leaving directory '/home/ubuntu/build/redis-3.0.0/src'  
ubuntu@ip-172-31-41-99:~/build/redis-3.0.0$
```

Terminal 3 (Ubuntu ip-172-31-41-100):

```
CC bio.o  
CC rio.o  
CC rand.o  
CC memtest.o  
CC crc64.o  
CC bitops.o  
CC sentinel.o  
CC notify.o  
CC setproctitle.o  
CC blocked.o  
CC hyperloglog.o  
CC latency.o  
CC sparkline.o  
LINK redis-server  
INSTALL redis-sentinel  
CC redis-cli.o  
LINK redis-cli  
CC redis-benchmark.o  
LINK redis-benchmark  
CC redis-check-dump.o  
LINK redis-check-dump  
CC redis-check-aof.o  
LINK redis-check-aof  
  
Hint: It's a good idea to run 'make test' ;)  
make[1]: Leaving directory '/home/ubuntu/build/redis-3.0.0/src'  
ubuntu@ip-172-31-41-100:~/build/redis-3.0.0$
```

Terminal 4 (Mac OS X):

```
Rashmi-MacBook-Pro:week9 rashmi$
```

Now let's run tests

```
sudo apt-get install -y tk8.5 tcl8.5
make test
(You should see lot of green [ok] messages)
```

The image shows four terminal windows side-by-side. The first three windows are from a Linux machine (Ubuntu) and the fourth is from a Mac (Rashmi's MacBook Pro). Each window displays the output of a Redis test suite. The test results are mostly 'ok' messages, indicating successful execution. The fourth window on the Mac shows a single command: 'rashmis-MacBook-Pro:week9 rashmi\$'. The terminal windows are titled 'week9' and show the command 'make test' being run.

```
week9 — ubuntu@ip-172-31-41-98: ~/build/redis-3.0.0 — ssh — 96x25
1 seconds - unit/limits
22 seconds - unit/dump
5 seconds - unit/scripting
24 seconds - unit/type/set
26 seconds - unit/type/zset
30 seconds - unit/sort
14 seconds - unit/maxmemory
32 seconds - integration/replication-2
14 seconds - unit/bits
15 seconds - unit/memefficiency
38 seconds - unit/basic
39 seconds - unit/type/list-3
45 seconds - unit/aofrw
26 seconds - unit/hyperloglog
52 seconds - integration/replication-3
43 seconds - integration/replication-psync
54 seconds - integration/replication-4
66 seconds - integration/replication
84 seconds - unit/obuf-limits
\o/ All tests passed without errors!
Cleanup: may take some time... OK
make[1]: Leaving directory '/home/ubuntu/build/redis-3.0.0/src'
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$ ①

week9 — ubuntu@ip-172-31-41-100: ~/build/redis-3.0.0 — ssh — 95x28
1 seconds - unit/slowlog
1 seconds - unit/introspection
1 seconds - unit/limits
19 seconds - unit/other
22 seconds - unit/type/set
7 seconds - unit/scripting
23 seconds - unit/dump
28 seconds - unit/sort
13 seconds - unit/maxmemory
30 seconds - unit/type/zset
14 seconds - unit/bits
33 seconds - unit/basic
32 seconds - integration/replication-2
38 seconds - unit/type/list-3
17 seconds - unit/memefficiency
26 seconds - unit/hyperloglog
48 seconds - unit/aofrw
52 seconds - integration/replication-3
45 seconds - integration/replication-psync
55 seconds - integration/replication-4
67 seconds - integration/replication
86 seconds - unit/obuf-limits
\o/ All tests passed without errors!
Cleanup: may take some time... OK
make[1]: Leaving directory '/home/ubuntu/build/redis-3.0.0/src'
ubuntu@ip-172-31-41-100:~/build/redis-3.0.0$ ②

week9 — ubuntu@ip-172-31-41-99: ~/build/redis-3.0.0 — ssh — 101x24
22 seconds - unit/type/set
22 seconds - unit/dump
24 seconds - unit/type/zset
5 seconds - unit/scripting
31 seconds - integration/replication-2
13 seconds - unit/maxmemory
33 seconds - unit/sort
13 seconds - unit/bits
38 seconds - unit/basic
16 seconds - unit/memefficiency
41 seconds - unit/type/list-3
23 seconds - unit/hyperloglog
48 seconds - unit/aofrw
54 seconds - integration/replication-3
45 seconds - integration/replication-psync
56 seconds - integration/replication-4
75 seconds - integration/replication
89 seconds - unit/obuf-limits
\o/ All tests passed without errors!
Cleanup: may take some time... OK
make[1]: Leaving directory '/home/ubuntu/build/redis-3.0.0/src'
ubuntu@ip-172-31-41-99:~/build/redis-3.0.0$ ③

Character Set Encoding: None
Rashmis-MacBook-Pro:week9 rashmi$ ④
```

Now lets update the configuration for cluster:

```
sudo vi redis.conf
```

Modify the following:

```
#On RedisClusterNode1 - redis.conf
bind 172.31.41.98
cluster-enabled yes
cluster-config-file nodes-6379.conf
cluster-node-timeout 5000
```

```
#On RedisClusterNode2 - redis.conf
```

```
bind 172.31.41.99
cluster-enabled yes
cluster-config-file nodes-6379.conf
cluster-node-timeout 5000
```

```
#On RedisClusterNode3 - redis.conf
bind 172.31.41.100
cluster-enabled yes
cluster-config-file nodes-6379.conf
cluster-node-timeout 5000
```

Next start Redis Server On all 3 nodes
src/redis-server ./redis.conf

Note: At any point if you want to **shutdown** redis-server:
connect to redis client -

```
ubuntu@ip-172-31-12-238:~/build/redis-3.0.0$ src/redis-cli -h 172.31.12.238
172.31.12.238:6379>shutdown
```

Now we have 3 nodes and we need to connect them: (Open another tab on each of the terminal)

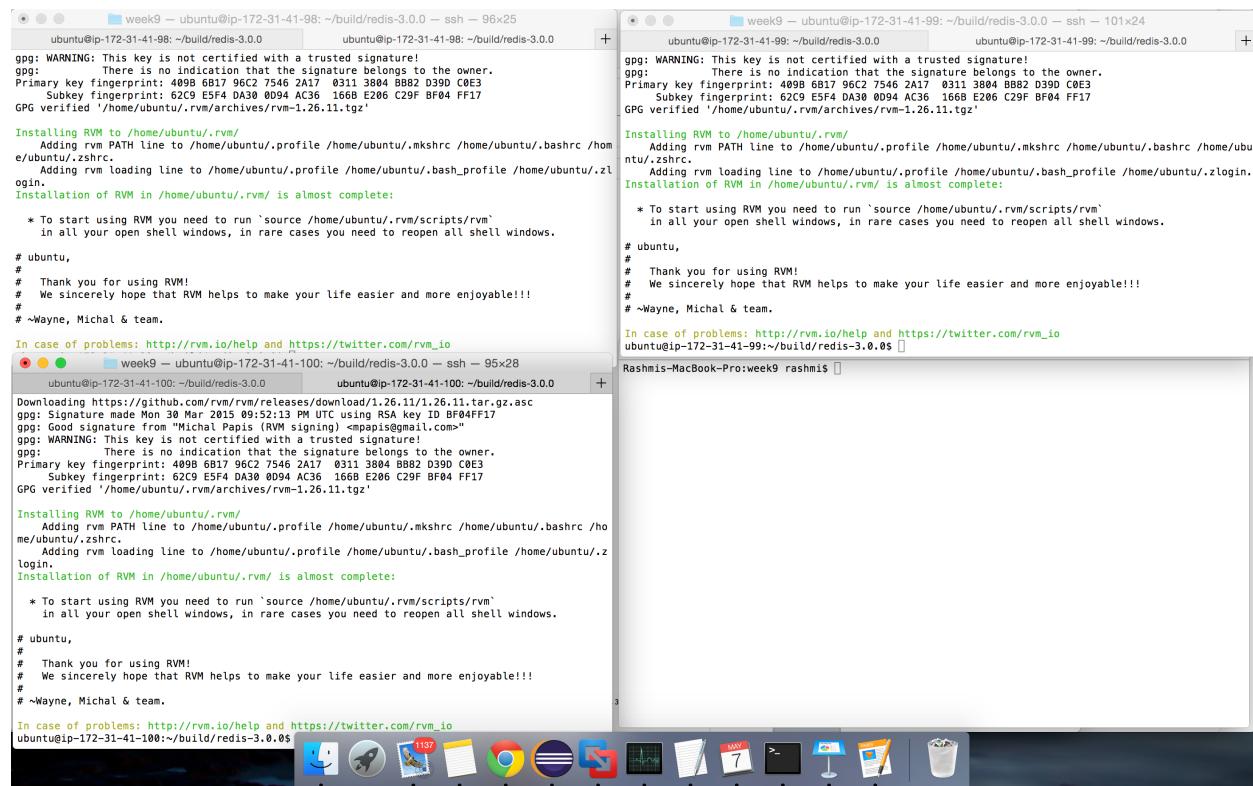
Install Ruby Redis gems

```
cd build/redis-3.0.0/  
\curl -L https://get.rvm.io | bash -s stable  
-This command gives error GPG signature verification failed  
Run the following command (Copy it from your terminal)
```

```
gpg --keyserver hkp://keys.gnupg.net --recv-keys  
409B6B1796C275462A1703113804BB82D39DC0E3
```

Run this again

```
\curl -L https://get.rvm.io | bash -s stable
```



```
week9 ~ ubuntu@ip-172-31-41-98:~/build/redis-3.0.0 - ssh - 96x25  
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0  
gpg: WARNING: This key is not certified with a trusted signature!  
gpg:   There is no indication that the signature belongs to the owner.  
Primary key fingerprint: 409B 6B17 96C2 7546 2A17  0311 3804 B8B2 D39D C0E3  
Subkey fingerprint: 62C9 E5F4 DA30 0D94 AC36  1668 E206 C29F BF04 FF17  
GPG verified '/home/ubuntu/.rvm/archives/rvm-1.26.11.tgz'  
  
Installing RVM to /home/ubuntu/.rvm/  
  Adding rvm PATH line to /home/ubuntu/.profile /home/ubuntu/.mkshrc /home/ubuntu/.bashrc /home/ubuntu/.zshrc.  
    Adding rvm loading line to /home/ubuntu/.profile /home/ubuntu/.bash_profile /home/ubuntu/.zlogin.  
  Installation of RVM in /home/ubuntu/.rvm/ is almost complete:  
    * To start using RVM you need to run `source /home/ubuntu/.rvm/scripts/rvm`  
      in all your open shell windows, in rare cases you need to reopen all shell windows.  
# ubuntu,  
#  
#   Thank you for using RVM!  
#   We sincerely hope that RVM helps to make your life easier and more enjoyable!!!  
#  
# ~Wayne, Michal & team.  
  
In case of problems: http://rvm.io/help and https://twitter.com/rvm_io  
week9 ~ ubuntu@ip-172-31-41-100:~/build/redis-3.0.0 - ssh - 95x28  
ubuntu@ip-172-31-41-100:~/build/redis-3.0.0  
Downloading https://github.com/rvm/rvm/releases/download/1.26.11/1.26.11.tar.gz.asc  
gpg: Signature made Mon 30 May 2016 09:52:13 PM UTC using RSA key ID BF04FF17  
gpg: Good signature from "Michal Čihař (signature) <cihar@gmail.com>"  
gpg: WARNING: This key is not certified with a trusted signature!  
gpg:   There is no indication that the signature belongs to the owner.  
Primary key fingerprint: 409B 6B17 96C2 7546 2A17  0311 3804 B8B2 D39D C0E3  
Subkey fingerprint: 62C9 E5F4 DA30 0D94 AC36  1668 E206 C29F BF04 FF17  
GPG verified '/home/ubuntu/.rvm/archives/rvm-1.26.11.tgz'  
  
Installing RVM to /home/ubuntu/.rvm/  
  Adding rvm PATH line to /home/ubuntu/.profile /home/ubuntu/.mkshrc /home/ubuntu/.bashrc /home/ubuntu/.zshrc.  
    Adding rvm loading line to /home/ubuntu/.profile /home/ubuntu/.bash_profile /home/ubuntu/.zlogin.  
  Installation of RVM in /home/ubuntu/.rvm/ is almost complete:  
    * To start using RVM you need to run `source /home/ubuntu/.rvm/scripts/rvm`  
      in all your open shell windows, in rare cases you need to reopen all shell windows.  
# ubuntu,  
#  
#   Thank you for using RVM!  
#   We sincerely hope that RVM helps to make your life easier and more enjoyable!!!  
#  
# ~Wayne, Michal & team.  
  
In case of problems: http://rvm.io/help and https://twitter.com/rvm_io  
ubuntu@ip-172-31-41-100:~/build/redis-3.0.0
```

```
source ~/.rvm/scripts/rvm  
rvm requirements  
rvm install ruby  
gem install redis
```

Finally:

```
src/redis-trib.rb create 172.31.41.98:6379 172.31.41.99:6379 172.31.41.100:6379
```

```
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$ src/redis-trib.rb create 172.31.41.98:6379 172.31.41.99:6379 172.31.41.100:6379
>>> Creating cluster
Connecting to node 172.31.41.98:6379: OK
Connecting to node 172.31.41.99:6379: OK
Connecting to node 172.31.41.100:6379: OK
>>> Performing hash slots allocation on 3 nodes...
Using 3 masters:
172.31.41.98:6379
172.31.41.99:6379
172.31.41.100:6379
M: d5deb40afbbf607c0c6f4165db64ed923f0af8e0 172.31.41.98:6379
  slots:0-5460 (5461 slots) master
M: f67869697e331d265590181a7db0c54a4207f746 172.31.41.99:6379
  slots:5461-10922 (5462 slots) master
M: c1a4998fbff1330fb9de95d1d338c9cd37ee4eae9 172.31.41.100:6379
  slots:10923-16383 (5461 slots) master
Can I set the above configuration? (type 'yes' to accept): yes
>>> Nodes configuration updated
>>> Assign a different config epoch to each node
>>> Sending CLUSTER MEET messages to join the cluster
Waiting for the cluster to join.
>>> Performing Cluster Check (using node 172.31.41.98:6379)
M: d5deb40afbbf607c0c6f4165db64ed923f0af8e0 172.31.41.98:6379
  slots:0-5460 (5461 slots) master
M: f67869697e331d265590181a7db0c54a4207f746 172.31.41.99:6379
  slots:5461-10922 (5462 slots) master
M: c1a4998fbff1330fb9de95d1d338c9cd37ee4eae9 172.31.41.100:6379
  slots:10923-16383 (5461 slots) master
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$
```

```
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$ ssh -t 172.31.41.98
M: d5deb40afbbf607c0c6f4165db64ed923f0af8e0 172.31.41.98:6379
  slots:0-5460 (5461 slots) master
M: f67869697e331d265590181a7db0c54a4207f746 172.31.41.99:6379
  slots:5461-10922 (5462 slots) master
M: c1a4998fbff1330fb9de95d1d338c9cd37ee4eae9 172.31.41.100:6379
  slots:10923-16383 (5461 slots) master
Can I set the above configuration? (type 'yes' to accept): yes
>>> Nodes configuration updated
>>> Assign a different config epoch to each node
Waiting for the cluster to join.
>>> Sending CLUSTER MEET messages to join the cluster
>>> Performing Cluster Check (using node 172.31.41.98:6379)
M: d5deb40afbbf607c0c6f4165db64ed923f0af8e0 172.31.41.98:6379
  slots:0-5460 (5461 slots) master
M: f67869697e331d265590181a7db0c54a4207f746 172.31.41.99:6379
  slots:5461-10922 (5462 slots) master
M: c1a4998fbff1330fb9de95d1d338c9cd37ee4eae9 172.31.41.100:6379
  slots:10923-16383 (5461 slots) master
[OK] All nodes agree about slots configuration.
>>> Check for open slots...
>>> Check slots coverage...
[OK] All 16384 slots covered.
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$
```



```
ubuntu@ip-172-31-41-99:~/build/redis-3.0.0$ ssh -t 172.31.41.99
13433:M 07 May 11:56:23.567 # Server started, Redis version 3.0.0
13433:M 07 May 11:56:23.567 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
13433:M 07 May 11:56:23.567 # WARNING you have Transparent Huge Pages (THP) support 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/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
13433:M 07 May 11:56:23.567 # WARNING The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to a lower value: 28.
13433:M 07 May 11:56:23.568 * The server is now ready to accept connections on port 6379
13433:M 07 May 12:02:43.083 # configEpoch set to 2 via CLUSTER SET-CONFIG-EPOCH
13433:M 07 May 12:02:43.082 # IP address for this node updated to 172.31.41.99
13433:M 07 May 12:02:48.002 # Cluster state changed: ok
```



```
Rashmis-MacBook-Pro:week9 rashmi$
```



```
13321:M 07 May 11:56:25.491 # Server started, Redis version 3.0.0
13321:M 07 May 11:56:25.491 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
13321:M 07 May 11:56:25.491 # WARNING you have Transparent Huge Pages (THP) support 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/transparent_hugepage/enabled' as root, and add it to your /etc/rc.local in order to retain the setting after a reboot. Redis must be restarted after THP is disabled.
13321:M 07 May 11:56:25.491 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to a lower value of 128.
13321:M 07 May 11:56:25.491 * The server is now ready to accept connections on port 6379
13321:M 07 May 12:02:43.122 # configEpoch set to 3 via CLUSTER SET-CONFIG-EPOCH
13321:M 07 May 12:02:43.122 # IP address for this node updated to 172.31.41.100
13321:M 07 May 12:02:48.122 # Cluster state changed: ok
```

Test the cluster:

On RedisClusterNode2

```
src/redis-cli -h 172.31.41.98 cluster nodes
```

```
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$ src/redis-cli -h 172.31.41.98 cluster
nodes
c1a4998fbf1330fb9de95d1d338c9cd37ee4eae9 172.31.41.100:6379 master - 0 1431000234118 3
connected 10923-16383
f67869697e331d265590181a7db0c54a4207f746 172.31.41.99:6379 master - 0 1431000235121 2
connected 5461-10922
d5deb40afbbf607c0c6f4165db64ed923f0af8e0 172.31.41.98:6379 myself,master - 0 0 1
connected 0-5460
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0$
```

Benchmarks -

```
wget https://github.com/antirez/redis-rb-cluster/archive/master.zip
sudo apt-get install unzip
unzip master.zip
cd redis-rb-cluster-master

ubuntu@ip-172-31-41-98:~/build/redis-3.0.0/redis-rb-cluster-master$ ruby example.rb
172.31.41.98 6379
```

Interrupt the program. Use **info** to check keys on all the clients. To delete all keys run **flushdb** on **redis-cli**.

The screenshot shows three terminal windows side-by-side:

- Terminal 1 (Left):** Shows the output of running `ruby example.rb`. It lists approximately 150 keys starting from 517 and ending at 172, all of which have the value "6379".
- Terminal 2 (Middle):** Shows the output of running `info keyspace` on a Redis client. It lists the same set of keys (517 to 172) and their values.
- Terminal 3 (Right):** Shows a Redis CLI session. The user runs `get foo87`, receives the response "6379", then runs `get foo087`, receives the response "6379", and finally runs `get foo504`, receiving the response "504".

Now, let's setup the Master Slave Replication

Master Slave Replication 101

- This replication allows for a database service to replicate data written to the master instance to a slave instance. This slave instance could be located within the same facility for scaling out requests or in another facility in order to mitigate a disaster recovery scenario.
- Master-slave replication allows slave Redis servers to be exact copies of master servers.
- Replication is asynchronous because of performance concerns. It means that Redis HA (High Availability) does not guarantee strong data consistency, i.e. It is not possible to ensure the slave actually received a given write, so there is always a window for data loss.

Add a Slave

(We are only replicating RedisClusterNode2 - procedure remains the same for other Nodes)

```
ec2-52-7-67-100.compute-1.amazonaws.com  
52.7.67.100  
ssh -i HadoopEC2Cluster.pem ubuntu@ec2-52-7-67-100.compute-1.amazonaws.com  
sudo apt-get install redis-server  
sudo apt-get install redis-tools
```

Configure the Slave

```
sudo vi /etc/redis/redis.conf  
  
bind 172.31.41.97 #Private IP  
slaveof 172.31.41.99 6379 #RedisClusterNode2
```

Start the server

```
sudo redis-server /etc/redis/redis.conf
```

Test

```
ubuntu@ip-172-31-41-97:~$ redis-cli -h 172.31.41.97
```

The screenshot shows two terminal windows side-by-side. Both windows have a title bar labeled "week9 — ubuntu@ip-172-31-41-99: ~/build/redis-3.0.0 — ssh — 101x24" and "ubuntu@ip-172-31-41-99: ~/build/redis-3.0.0". The left window has a red dot icon in its title bar. The right window has a plus sign icon in its title bar.

The left terminal window displays the following Redis command and response:

```
151) "foo126"
152) "foo113"
153) "foo216"
154) "foo401"
155) "foo211"
156) "foo32"
157) "foo395"
158) "foo54"
159) "foo359"
160) "foo69"
161) "foo337"
162) "foo309"
163) "foo396"
164) "foo39"
165) "foo455"
166) "foo147"
167) "foo464"
168) "foo364"
169) "foo506"
170) "foo150"
171) "foo220"
172.31.41.99:6379> get foo220
"220"
172.31.41.99:6379>
```

The right terminal window displays the following Redis command and response:

```
147) "foo330"
148) "foo126"
149) "foo499"
150) "foo212"
151) "foo161"
152) "foo495"
153) "foo491"
154) "foo216"
155) "foo506"
156) "foo93"
157) "foo511"
158) "foo43"
159) "foo455"
160) "foo459"
161) "foo285"
162) "foo249"
163) "foo355"
164) "foo98"
165) "foo301"
166) "foo8"
167) "foo139"
168) "foo312"
169) "foo198"
170) "foo412"
171) "foo289"
172.31.41.97:6379> get foo220
"220"
172.31.41.97:6379>
```

MovieLens Dataset - Load Users in Cluster

(Use Java Client - Jedis to load 230,000 users and query)

On RedisClusterNode1

Install JAVA

```
sudo apt-get install openjdk-7-jdk
sudo apt-get install unzip
```

Create directories on RedisClusterNode1:

```
cd ;mkdir -p RedisProject/src/org/hu/e63/
cd RedisProject; mkdir classes; mkdir input; mkdir jars
```

Download and unzip latest MovieLens Dataset (21million ratings and 132 MB zip file)

```
wget http://files.grouplens.org/datasets/movielens/ml-latest.zip
unzip ml-latest.zip
rm ml-latest.zip
mv ml-latest/ratings.csv input/ratings.csv
rm -R ml-latest/
```

MovieLens - ratings.csv

```
ubuntu@ip-172-31-41-98:~/RedisProject/input$ cat ratings.csv | head
userId,movieId,rating,timestamp
1,253,3.0,900660748
1,1387,3.0,900660748
1,1407,5.0,900660871
1,1717,5.0,900660871
1,1876,3.0,900660543
1,1882,3.0,900660584
1,1895,4.0,900660518
1,1909,3.0,900660468
1,1911,3.0,900660543
ubuntu@ip-172-31-41-98:~/RedisProject/input$ 

ubuntu@ip-172-31-41-98:~/RedisProject/input$ ls -l
total 554976
-rw-r--r-- 1 ubuntu ubuntu 568287348 May  7 12:35 ratings.csv
ubuntu@ip-172-31-41-98:~/RedisProject/input$ cat ratings.csv | wc -l
21063129
ubuntu@ip-172-31-41-98:~/RedisProject/input$
```

Now we will load ratings.csv to Redis, Our goal is to be able get query by user. For eg: **User X has rated which movies ?** We will look at other queries later.

Redis is very flexible (as there is no precise schema required), I am storing ratings.csv in Redis sets, Eg:

SADD u:1:m 253

SADD u:1:m 1387

#Above statements create a set with key u:1:m (userid 1) and values [253, 1387] (movie nos.)

Following is MovieLensJedis.java program - which creates SADD commands:
MovieLensJedis.java

```
package org.hu.e63;

import java.io.File;
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;

import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.Jedis;
import redis.clients.jedis.JedisCluster;

public class MovieLensJedis {
    public static void main(String[] args) {
        Jedis jedis = new Jedis("172.31.41.98", 6379);
        System.out.println("Connection to server sucessfully hello!!!");
        //check whether server is running or not
        System.out.println("Server is running: "+jedis.ping());

        Set<HostAndPort> jedisClusterNodes = new HashSet<HostAndPort>();
        //Jedis Cluster will attempt to discover cluster nodes automatically
        jedisClusterNodes.add(new HostAndPort("172.31.41.98", 6379));
        jedisClusterNodes.add(new HostAndPort("172.31.41.99", 6379));
        jedisClusterNodes.add(new HostAndPort("172.31.41.100", 6379));
        JedisCluster jc = new JedisCluster(jedisClusterNodes);
        try {
            Scanner s = new Scanner(new File("input/ratings.csv"));
            String s1 = s.nextLine(); //Skip first line
            String key = "";
            String member = "";
            while (s.hasNextLine()){
                s1 = s.nextLine();
                String[] spl = s1.split(",");
                key = "u:"+spl[0]+":m";
                member = spl[1];
                jc.sadd(key, member);
            }
        }
        catch(Exception e)
        {
            System.out.println("bye" + e.getMessage());
        }
        jedis.close();
    }
}
```

(I wrote the code on Eclipse on my macbook - then copied over the JAVA, JAR and input files to EC2 instance)

Copy java and jar files

Remember to change the IP address to connect in .JAVA file

Copy JAVA source file

```
scp -i HadoopEC2Cluster.pem ../../workspace/RedisProject/src/org/hu/e63/  
MovieLensJedis.java ubuntu@ec2-52-6-250-55.compute-1.amazonaws.com:RedisProject/src/  
org/hu/e63/MovieLensJedis.java
```

Copy required JARS

Required Jar - jedis-2.4.2.jar, commons-pool2-2.0.jar

```
scp -i HadoopEC2Cluster.pem ../FinalProject/Redis/jedis-2.4.2.jar ../FinalProject/  
Redis/commons-pool2-2.0.jar  
ubuntu@ec2-52-6-250-55.compute-1.amazonaws.com:RedisProject/jars
```

Compile

```
javac -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes -d classes  
src/org/hu/e63/MovieLensJedis.java
```

Run

```
java -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes  
org.hu.e63.MovieLensJedis
```

After copying over and compiling my directory :

```
ubuntu@ip-172-31-59-103:~$ tree RedisProject/  
RedisProject/  
├── classes  
│   └── org  
│       └── hu  
│           └── e63  
│               └── MovieLensJedis.class  
├── input  
│   └── ratings.csv  
└── jars  
    ├── commons-pool2-2.0.jar  
    └── jedis-2.4.2.jar  
└── src  
    └── org  
        └── hu  
            └── e63  
                └── MovieLensJedis.java
```

```
10 directories, 5 files  
ubuntu@ip-172-31-59-103:~$
```

Following screenshot shows about 230,000 keys on Redis Cluster (about 76000 in each node)

The screenshot displays three terminal windows from a Mac OS X desktop environment. Each window shows Redis statistics for a different node in a cluster. The nodes are identified by their IP addresses: 172.31.41.98, 172.31.41.100, and 172.31.41.97. The statistics include replication details, CPU usage, and keyspace information. The keyspace section shows approximately 76,000 keys per node.

```
ubuntu@ip-172-31-41-98:~/build/redis-3.0.0 - ssh - 96x25
ubuntu@ip-17...uild/redis-3.0.0 ... ubuntu@ip-17...uild/redis-3.0.0 ubuntu@ip-17...uild/redis-3.0.0 +
pubsub_patterns:0
latest_fork_usec:1822
migrate_cached_sockets:0
# Replication
role:master
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0
# CPU
used_cpu_sys:186.72
used_cpu_user:35.23
used_cpu_sys_children:0.91
used_cpu_user_children:8.79
# Cluster
cluster_enabled:1
# Keyspace
db0:keys=76379,expires=0,avg_ttl=0
172.31.41.98:6379> [1]
ubuntu@ip-172-31-41-100:~/build/redis-3.0.0 b... ubuntu@ip-172-31-41-100:~/build/redis-3.0.0 +
keyspace_hits:178
keyspace_misses:1
pubsub_channels:0
pubsub_patterns:0
latest_fork_usec:2228
migrate_cached_sockets:0
# Replication
role:master
connected_slaves:0
master_repl_offset:0
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:0
repl_backlog_histlen:0
# CPU
used_cpu_sys:69.56
used_cpu_user:47.51
used_cpu_sys_children:9.69
used_cpu_user_children:9.65
# Cluster
cluster_enabled:1
# Keyspace
db0:keys=76341,expires=0,avg_ttl=0
172.31.41.100:6379> [1]

week9 — ubuntu@ip-172-31-41-99: ~/build/redis-3.0.0 - ssh - 101x24
ubuntu@ip-172-31-41-99:~/build/redis-3.0.0 ubuntu@ip-172-31-41-99:~/build/redis-3.0.0 +
latest_fork_usec:1537
migrate_cached_sockets:0
# Replication
role:master
connected_slaves:0
master_repl_offset:98763005
repl_backlog_active:0
repl_backlog_size:1048576
repl_backlog_first_byte_offset:97714430
repl_backlog_histlen:1048576
# CPU
used_cpu_sys:66.58
used_cpu_user:46.11
used_cpu_sys_children:0.85
used_cpu_user_children:9.29
# Cluster
cluster_enabled:1
# Keyspace
db0:keys=76340,expires=0,avg_ttl=0
172.31.41.99:6379> [1]
ubuntu@ip-172-31-41-97:~$ exit
logout
Connection to ec2-52-7-67-100.compute-1.amazonaws.com closed.
Rashmis-MacBook-Pro:week9 rashmi$
```

MovieLensJedisQuery

Now we write a program to query above stored data:

```
package org.hu.e63;

import java.util.HashSet;
import java.util.Set;
import redis.clients.jedis.HostAndPort;
import redis.clients.jedis.JedisCluster;

public class MovieLensJedisQuery {
    public static void main(String[] args) {

        Set<HostAndPort> jedisClusterNodes = new HashSet<HostAndPort>();
        //Jedis Cluster will attempt to discover cluster nodes automatically
        jedisClusterNodes.add(new HostAndPort("172.31.2.131", 6379));
        JedisCluster jc = new JedisCluster(jedisClusterNodes);
        try {
            System.out.println(jc.smembers("u:1:m"));
        }
        catch(Exception e)
        {
            System.out.println("ERROR:" + e.getMessage());
        }
    }
}
```

Copy java and jar files

Remember to change the IP address to connect in .JAVA file

Copy JAVA source file

```
scp -i HadoopEC2Cluster.pem ../../workspace/RedisProject/src/org/hu/e63/
MovieLensJedisQuery.java ubuntu@ec2-52-0-211-157.compute-1.amazonaws.com:RedisProject/
src/org/hu/e63/MovieLensJedisQuery.java
```

Copy required JARS

Required Jar - jedis-2.4.2.jar, commons-pool2-2.0.jar

```
scp -i HadoopEC2Cluster.pem ../FinalProject/Redis/jedis-2.4.2.jar ../FinalProject/
Redis/commons-pool2-2.0.jar
ubuntu@ec2-52-0-211-157.compute-1.amazonaws.com:RedisProject/jars
```

Compile

```
javac -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes -d classes
src/org/hu/e63/MovieLensJedisQuery.java
```

Run

```
java -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes  
org.hu.e63.MovieLensJedisQuery  
[1895, 2002, 1407, 2003, 1917, 1918, 1876, 1882, 1911, 1982, 1909, 1717, 1983, 1387,  
253, 2001, 2000]
```

After copying over and compiling my directory :

```
ubuntu@ip-172-31-2-131:~/RedisProject$ tree  
. .  
|   └── classes  
|       └── org  
|           └── hu  
|               └── e63  
|                   ├── MovieLensJedis.class  
|                   └── MovieLensJedisQuery.class  
└── input  
    └── ratings.csv  
└── jars  
    ├── commons-pool2-2.0.jar  
    └── jedis-2.4.2.jar  
└── src  
    └── org  
        └── hu  
            └── e63  
                ├── MovieLensJedis.java  
                └── MovieLensJedisQuery.java  
  
10 directories, 7 files
```

To retrieve **User 706 has rated which movies?** run the following on Redis Client

```
172.31.41.100:6379> smembers u:706:m  
1) "170"  
2) "355"  
3) "1129"  
4) "1339"  
5) "1381"  
6) "1892"  
7) "2454"  
8) "2605"  
9) "2724"  
10) "2763"  
11) "2881"
```

```
12) "3476"
13) "3686"
14) "3798"
15) "3827"
172.31.41.100:6379>
```

Mass Insertion Protocol (Pipeline)

(Load 21M records on Redis using Mass Insertion Protocol)

Do the following on a node that is not part of Cluster or your Virtual Machine (Redis Cluster doesn't support pipeline)

Install Redis and Java

Install JAVA

```
sudo apt-get install openjdk-7-jdk  
sudo apt-get install unzip
```

Create directories on RedisClusterNode1:

```
mkdir -p RedisProject/src/org/hu/e63/  
cd RedisProject; mkdir classes; mkdir input; mkdir jars
```

Download and unzip latest MovieLens Dataset (21million ratings and 132 MB zip file)

```
wget http://files.grouplens.org/datasets/movielens/ml-latest.zip  
unzip ml-latest.zip  
rm ml-latest.zip  
mv ml-latest/ratings.csv input/ratings.csv  
rm -R ml-latest/
```

MovieLens - ratings.csv

```
ubuntu@ip-172-31-41-98:~/RedisProject/input$ cat ratings.csv | head  
userId,movieId,rating,timestamp  
1,253,3.0,900660748  
1,1387,3.0,900660748  
1,1407,5.0,900660871  
1,1717,5.0,900660871  
1,1876,3.0,900660543  
1,1882,3.0,900660584  
1,1895,4.0,900660518  
1,1909,3.0,900660468  
1,1911,3.0,900660543  
ubuntu@ip-172-31-41-98:~/RedisProject/input$  
  
ubuntu@ip-172-31-41-98:~/RedisProject/input$ ls -l  
total 554976  
-rw-r--r-- 1 ubuntu ubuntu 568287348 May  7 12:35 ratings.csv  
ubuntu@ip-172-31-41-98:~/RedisProject/input$ cat ratings.csv | wc -l  
21063129  
ubuntu@ip-172-31-41-98:~/RedisProject/input$
```

Now We will try to add 21M movie ratings on to Node1 (not in cluster mode)

For eg:

```
HMSET u:1:m:253 RATING 3.0 TIMESTAMP 900660748  
HMSET u:1:m:1387 RATING 3.0 TIMESTAMP 900660748
```

(I wrote the code on Eclipse on my macbook - then copied over the JAVA, JAR and input files to EC2 instance)

Source Code (MovieLens21M.java)

```
package org.hu.e63;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.util.Scanner;

public class MovieLens21M {
    public static void main(String[] args) {
        try{
            Scanner s = new Scanner(new File("input/ratings.csv"));
            File fileDir = new File("outputfile1.txt");
            Writer out = new BufferedWriter(new OutputStreamWriter(
                new FileOutputStream(fileDir), "UTF8"));
            String resp = "";
            String s1 = s.nextLine(); //Skip first line
            while (s.hasNextLine()){
                s1 = s.nextLine();
                String[] spl = s1.split(",");
                int ln = 0;

                ln = 5 + spl[0].length()+spl[1].length();
                resp = "*" + 6 + "\r\n" +
                    "$5\r\n" +
                    "HMSET\r\n" +
                    "$" + ln + "\r\n" +
                    "u:" + spl[0] + ":m:" + spl[1] + "\r\n" +
                    "$6\r\n" +
                    "RATING\r\n" +
                    "$"+spl[2].length()+"\r\n" +
                    spl[2]+"\r\n" +
                    "$9\r\n" +
                    "TIMESTAMP\r\n" +
                    "$" + spl[3].length()+"\r\n" +
                    spl[3] + "\r\n"
                ;
                out.append(resp);
            }

            s.close();
            out.flush();
            out.close();
        }
        catch(Exception e)
        {
            System.out.println("bye" + e.getMessage());
        }
    }
}
```

Copy JAVA source file

From the terminal run the following command

```
scp -i HadoopEC2Cluster.pem ../../workspace/RedisProject/src/org/hu/e63/  
MovieLens21M.java ubuntu@ec2-52-7-171-102.compute-1.amazonaws.com:RedisProject/src/  
org/hu/e63/MovieLens21M.java
```

Copy required JARS

(Required Jar - jedis-2.4.2.jar, commons-pool2-2.0.jar)

From the terminal run the following command

```
scp -i HadoopEC2Cluster.pem ../../FinalProject/Redis/jedis-2.4.2.jar ../../FinalProject/  
Redis/commons-pool2-2.0.jar  
ubuntu@ec2-52-7-171-102.compute-1.amazonaws.com:RedisProject/jars
```

Compile (on RedisClusterNode1)

```
javac -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes -d classes  
src/org/hu/e63/MovieLens21M.java
```

Run (on RedisClusterNode1)

```
java -classpath jars/jedis-2.4.2.jar:jars/commons-pool2-2.0.jar:classes  
org.hu.e63.MovieLens21M
```

Directory Structure (After Run my directory looks like this:)

```
ubuntu@ip-172-31-12-238:~/RedisProject$ tree
```

```
.-  
|   +-- classes  
|   |   +-- org  
|   |   |   +-- hu  
|   |   |   |   +-- e63  
|   |   |   |       +-- MovieLens21M.class  
|   +-- input  
|       +-- ratings.csv  
+-- jars  
    +-- commons-pool2-2.0.jar  
    +-- jedis-2.4.2.jar  
+-- outputfile1.txt  
+-- src  
    +-- org  
        +-- hu  
            +-- e63  
                +-- MovieLens21M.java
```

```
10 directories, 6 files  
ubuntu@ip-172-31-12-238:~/RedisProject$
```

Load 21M records on Redis:

```

cd; cd build/redis-3.0.0/
cat ../../RedisProject/outputfile1.txt | src/redis-cli -h 172.31.12.238 --pipe

ubuntu@ip-172-31-12-238:~/build/redis-3.0.0$ date;cat ../../RedisProject/
outputfile1.txt | src/redis-cli -h 172.31.12.238 --pipe;date
Mon May 11 15:02:59 UTC 2015
All data transferred. Waiting for the last reply...
Last reply received from server.
errors: 0, replies: 21063128
Mon May 11 15:03:59 UTC 2015
ubuntu@ip-172-31-12-238:~/build/redis-3.0.0$

```

Note that It took 1 min. to load 21M keys.

Verification

```

ubuntu@ip-172-31-12-238:~/build/redis-3.0.0$ src/redis-cli -h 172.31.12.238
172.31.12.238:6379> HMGET u:1:m:253 RATING
1) "3.0"
172.31.12.238:6379> HMGET u:1:m:253 TIMESTAMP
1) "900660748"
172.31.12.238:6379> HGETALL u:1:m:253
1) "RATING"
2) "3.0"
3) "TIMESTAMP"
4) "900660748"
172.31.12.238:6379>

```

Partial output of info command on redis-cli

```

# Keyspace
db0:keys=21063128,expires=0,avg_ttl=0

```

YouTube Video Links

Short Video

<https://www.youtube.com/watch?v=eYFnniP-ELk>

Full Video

https://www.youtube.com/watch?v=fk3_5U3TI9c

Redis Cluster Installation (Extra Video)

https://www.youtube.com/watch?v=s4YpCA2Y_Q

References

<http://ilyabylich.svbtle.com/redis-cluster-quick-overview>

<http://redis.io/topics/cluster-tutorial>

<http://redis.io/topics/mass-insert>

<http://bencane.com/2013/11/12/installing-redis-and-setting-up-master-slave-replication/>

<http://redis.io/topics/replication>

<https://www.digitalocean.com/community/tutorials/how-to-install-ruby-on-rails-on-ubuntu-12-04-lts-precise-pangolin-with-rvm>