JetChars Documentation

Release 0.1

JetChars

May 18, 2015

CHAPTER

ONE

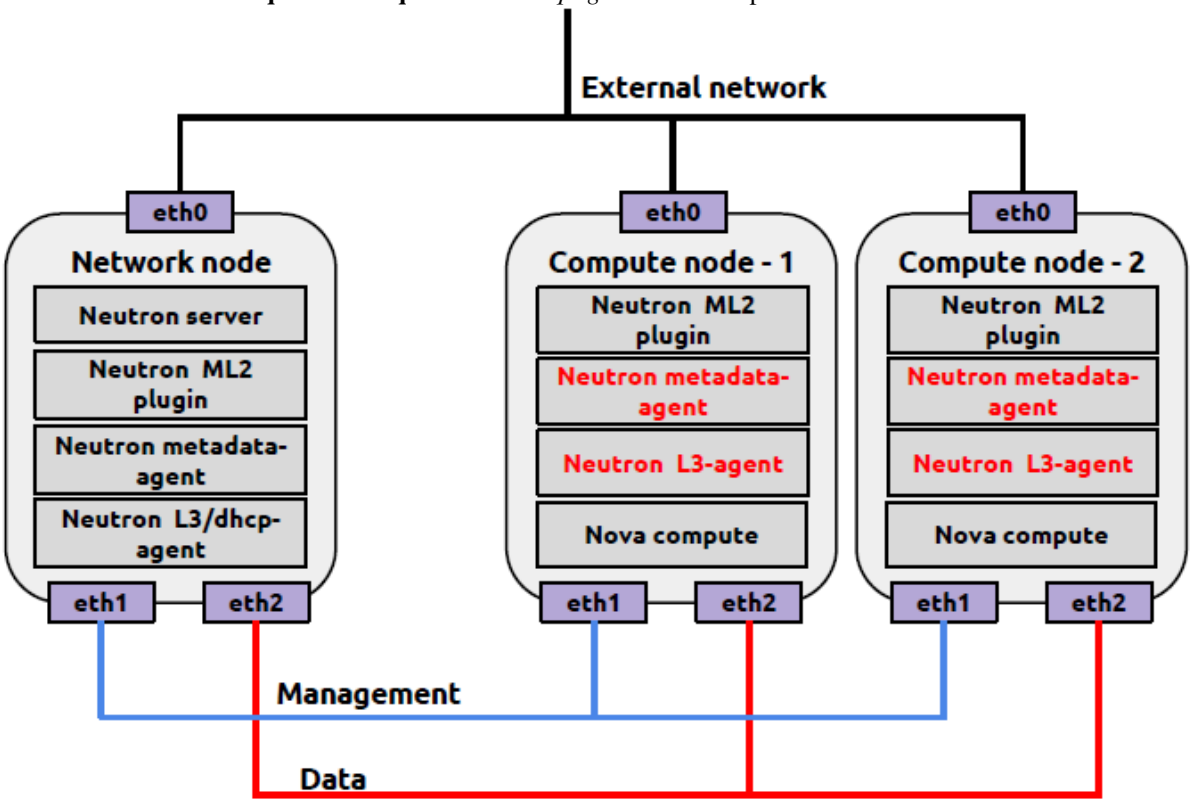
DEVSTACK HACKING

1.1 Enable DVR with DevStack

DVR is short for distributed virtual router, with this feature enabled packets flow with floating IP will no longer send to network node. It helps to alleviate network node's pressure greatly when large amount of north-south data flow occurs. [1]

1.1.1 Brief Intro

In order to enable distributed router on each compute-node, Neutron-metadata-agent and Neutron-L3-agent are both needed. So we need to add **q-meta** and **q-l3** as well as *q-agt* on each computer node's `local.conf` file.



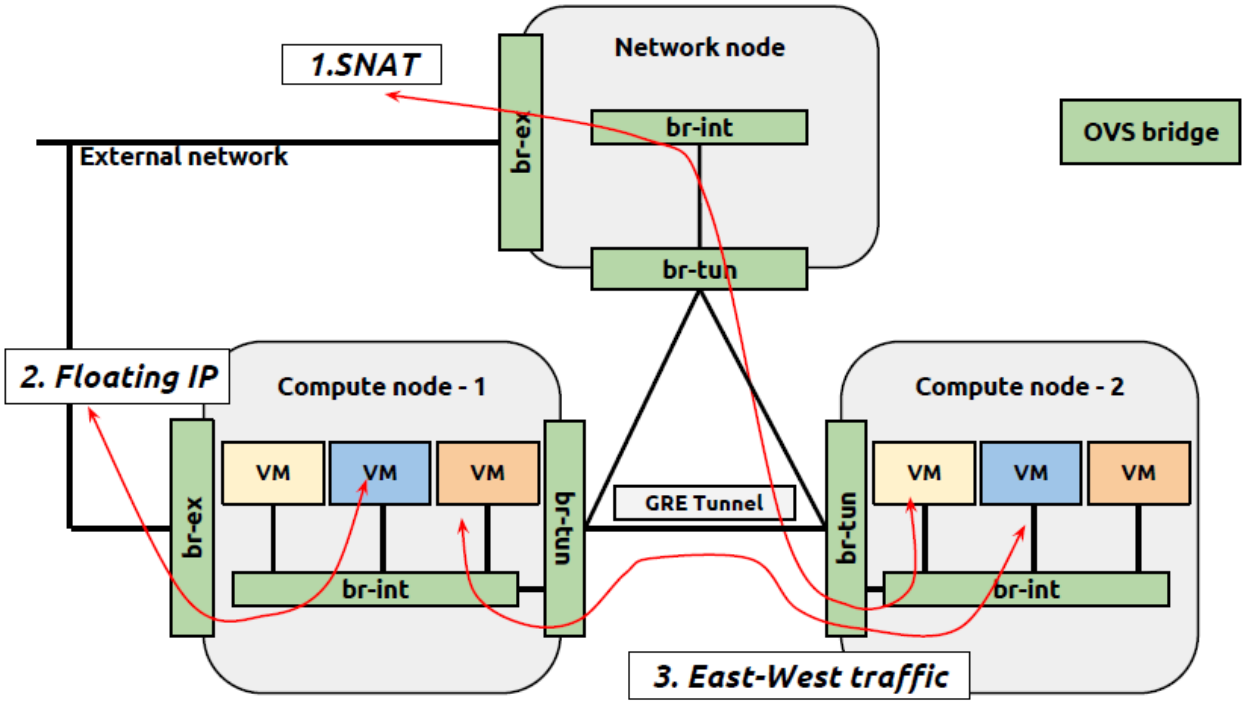
[1] https://wiki.openstack.org/wiki/Neutron/DVR/HowTo

Warning: Currently devstack doesn't support deploying DVR on GRE tunnel [a] , and tunnel type has been hard coded to vxlan mode, the following is a part of devstack's code `lib/neutron_plugins/ml2`:

[a] https://blueprints.launchpad.net/neutron/+spec/neutron-ovs-dvr

```
if [[ "$Q_DVR_MODE" != "legacy" ]]; then
    populate_ml2_config /$Q_PLUGIN_CONF_FILE agent l2_population=True
    populate_ml2_config /$Q_PLUGIN_CONF_FILE agent tunnel_types=vxlan
    populate_ml2_config /$Q_PLUGIN_CONF_FILE agent enable_distributed_routing=True
fi
```

With DVR, floating IPs can be accessed directly from each compute node, but SNAT still need to be centralized to network node.



1.1.2 Configure Network Node

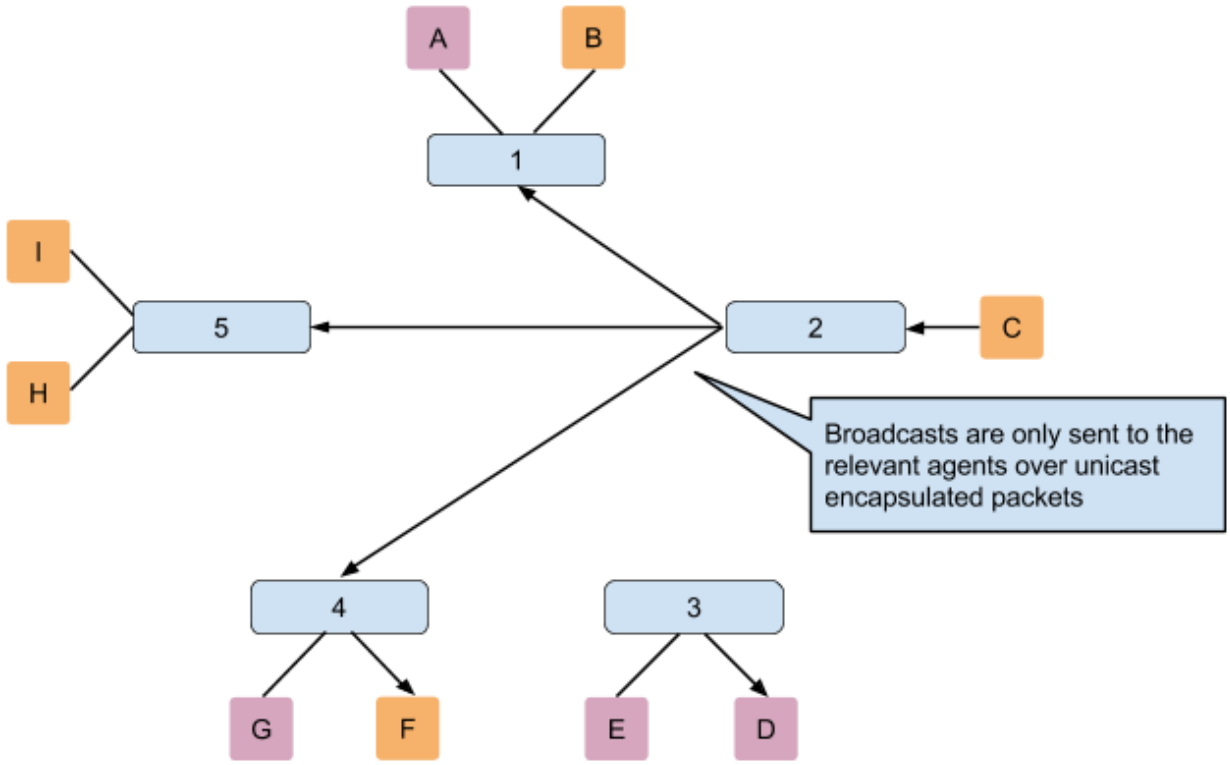
Here's the neutron configuration part of `local.conf` on network node.

```
# Neutron-vxlan-tunnel-DVR
##########################
ENABLED_SERVICES+=,q-svc,q-agt,q-dhcp,q-l3,q-meta,neutron
Q_FLOATING_ALLOCATION_POOL=start=192.168.137.166,end=192.168.137.253
Q_ROUTER_NAME=default_router
PUBLIC_NETWORK_GATEWAY=192.168.137.254
FLOATING_RANGE=192.168.0.0/16

FIXED_RANGE=10.1.1.0/24
FIXED_NETWORK_SIZE=256
NETWORK_GATEWAY=10.1.1.1

```

```
Q_PLUGIN=ml2
Q_ML2_TENANT_NETWORK_TYPE=vxlan
TUNNEL_ENDPOINT_IP=192.168.1.37
Q_DVR_MODE=dvr_snat
Q_SERVICE_PLUGIN_CLASSES=neutron.services.l3_router.l3_router_plugin.L3RouterPlugin
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch,linuxbridge,l2population
```

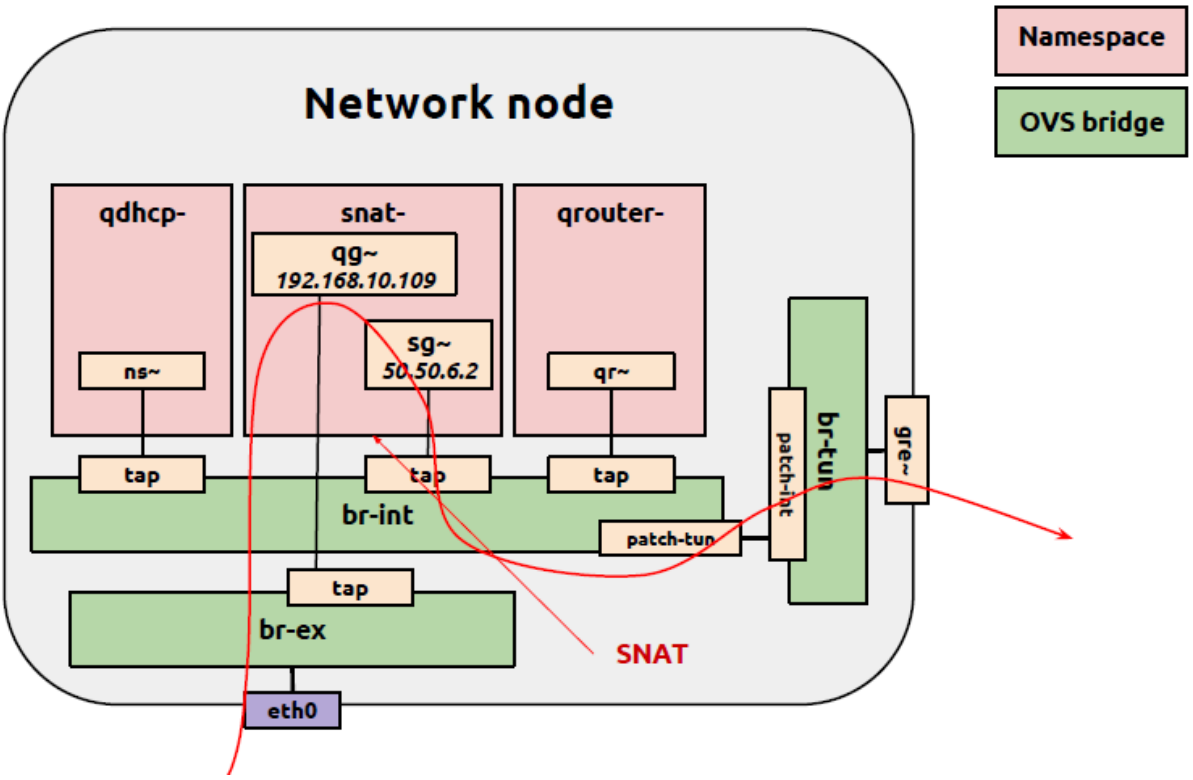
Note: DVR mode can be **dvr_snat** , **dvr** or **legacy**. *Legacy* is Q_DVR_MODE 's default value, *dvr_snat* is for network node which enables snat router, and *dvr* mode is for compute node.

L2population is needed by DVR. The L2 Population driver enables broadcast, multicast, and unicast traffic to scale out on large overlay networks. This traffic is sent to the relevant agent via encapsulation as a targeted unicast. [2]



After Installation you might see 3 bridges and 4 namespaces on network node.

[2] https://wiki.openstack.org/wiki/Neutron/DVR_L2_Agent




```
root@m0137:~# ovs-vsctl show | grep Bridge
    Bridge br-ex
    Bridge br-tun
    Bridge br-int
root@m0137:~# ip netns
fip-a8d72e44-d20d-4343-ae93-c24bcf89868c
qdhcp-cdc7119b-67a0-4639-b848-2d98cd9b23d6
snat-bc2a0cca-8b33-42fc-9efa-ee14a239052a
qrouter-bc2a0cca-8b33-42fc-9efa-ee14a239052a
```

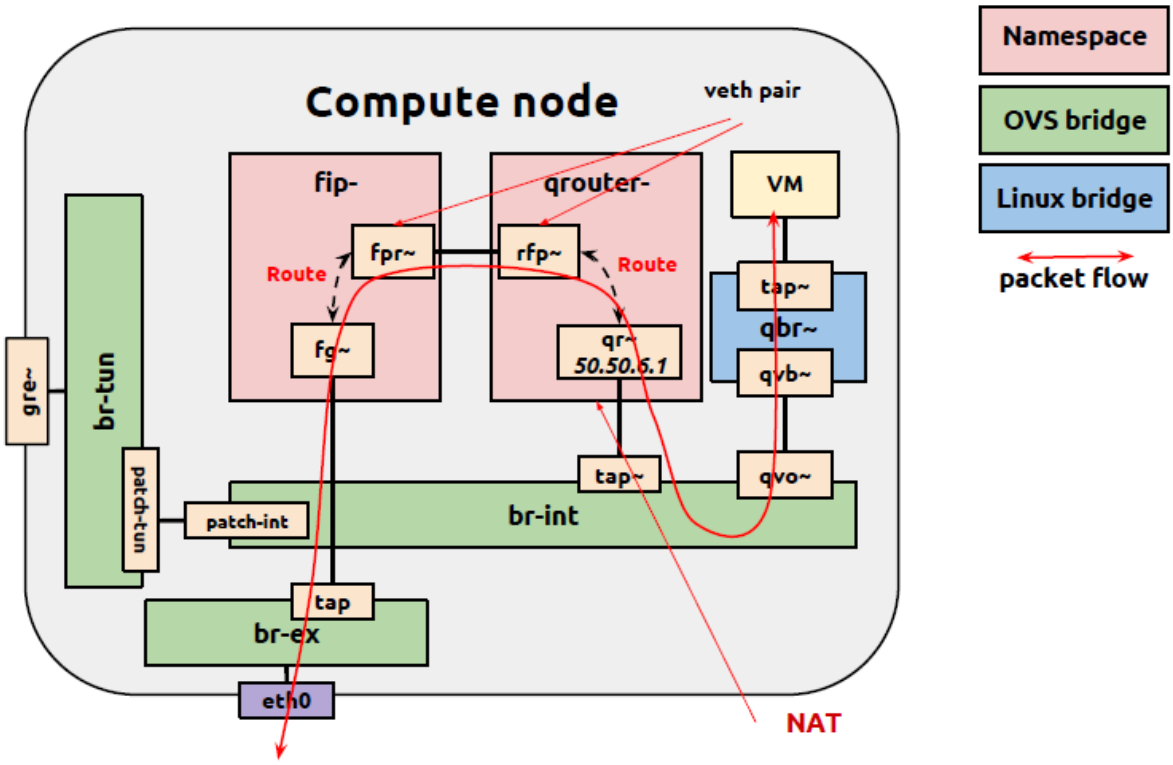
Namespace fip* is for floating IP accessing. qdhcp* is for allocating IP addresses. snat* is for SNAT function. qrouter* only serves VM in current host.

1.1.3 Configure Compute Node

The following is the neutron configuration part of `local.conf` on compute node

```
# Neutron-vxlan-tunnel-DVR
##########################
ENABLED_SERVICES+= q-agt,q-l3,q-meta, neutron
Q_PLUGIN=ml2
Q_ML2_TENANT_NETWORK_TYPE=vxlan
TUNNEL_ENDPOINT_IP=192.168.1.34
Q_DVR_MODE=dvr
Q_SERVICE_PLUGIN_CLASSES=neutron.services.l3_router.l3_router_plugin.L3RouterPlugin
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch,linuxbridge,l2population
```

After installation you might see 3 bridges and 2 namespaces.




```
root@m0134:~# ovs-vsctl show | grep Bridge
    Bridge br-tun
    Bridge br-ex
    Bridge br-int
root@m0134:~# ip netns
fip-a8d72e44-d20d-4343-ae93-c24bcf89868c
qrouter-bc2a0cca-8b33-42fc-9efa-ee14a239052a
```

fip* and qrouter* did the same job as two virtual devices on network node. We still need to do some configurations manually.

1. Add an free physical device(NIC) to br-ex

```
$ sudo ovs-vsctl add-port br-ex eth1
```

2. Allocate an IP for br-ex as a gateway

```
$ sudo ifconfig br-ex 192.168.137.253
```

3. Add a route to floating network via fip*

Before we adding this route, we need to know fip's IP address.

```
root@m0134:~# ip netns
fip-a8d72e44-d20d-4343-ae93-c24bcf89868c
qrouter-bc2a0cca-8b33-42fc-9efa-ee14a239052a
root@m0134:~# ip netns exec fip-a8d72e44-d20d-4343-ae93-c24bcf89868c ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
3: fpr-bc2a0cca-8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether ee:7c:26:79:0d:25 brd ff:ff:ff:ff:ff:ff
    inet 169.254.31.29/31 scope global fpr-bc2a0cca-8
       valid_lft forever preferred_lft forever
    inet6 fe80::ec7c:26ff:fe79:d25/64 scope link
       valid_lft forever preferred_lft forever
42: fg-6397841f-d3: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default
    link/ether fa:16:3e:f9:f4:0f brd ff:ff:ff:ff:ff:ff
    inet 192.168.137.171/16 brd 192.168.255.255 scope global fg-6397841f-d3
       valid_lft forever preferred_lft forever
    inet6 fe80::f816:3eff:fef9:f40f/64 scope link
       valid_lft forever preferred_lft forever
```

We use the IP on fg* .

```
$ sudo ip route add 192.168.0.0/16 via 192.168.137.171
```

1.1.4 References

CHAPTER

TWO

OPENSTACK CUSTOMIZATION

2.1 KVM Optimization

2.1.1 Disk Optimization

Asynchronous IO

Disk Cache Mode

2.1.2 Memory Optimization

Transparent HugePage

2.1.3 Network Optimization

MTU Size

CHAPTER

THREE

HADOOP TUNING

3.1 HiBench - The Hadoop BenchMark Suit

Most of my hadoop tuning data derived with this benchmark tool.

3.1.1 What is HiBench? [1]

This benchmark suite contains 10 typical Hadoop workloads (including micro benchmarks, HDFS benchmarks, web search benchmarks, machine learning benchmarks, and data analytics benchmarks). [2]

Advantages:

- Realistic and comprehensive
- Quantitive characteriztion of different workload
- Evaluation of different deployment

3.1.2 BenchMark Types

HiBench Contains 5 different types of benchmark.

1. micro benchmarks: `Sort` `WordCount` `TeraSort`
2. HDFS benchmarks: `enhanced DFSIO` `Sleep`
3. web search benchmarks: `Nutching indexing` `PageRank`
4. machine learning benchmarks: `Bayes Classification` `K-means Clustering`
5. data analytics benchmarks: `Hive Query Benchmark`

3.1.3 Configuration & Running scripts

Global enviroment variable in `bin/hibench-config.sh`

Each workload can run separately:

- `conf/configure.sh` Configuration file contains all parameters such as data size and test options.
- `bin/prepare*.sh` Generate or copy each workload's prepare data into HDFS.
- `bin/run*.sh` Execute the workload

[1] https://github.com/intel-hadoop/Hibench
[2] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=5452747

3.1.4 Default Configurations

Default configurations' data size normally very small.(Release version: 3.0)

DFSIOE

```
## paths
INPUT_HDFS=${DATA_HDFS}/benchmarks/TestDFSIO-Enh

export HADOOP_OPTS="$HADOOP_OPTS -Dtest.build.data=${INPUT_HDFS}"
MAP_JAVA_OPTS=`cat $HADOOP_CONF_DIR/mapred-site.xml | grep "mapreduce.map.java.opts" | awk -F\< '{pri
RED_JAVA_OPTS=`cat $HADOOP_CONF_DIR/mapred-site.xml | grep "mapreduce.reduce.java.opts" | awk -F\< '

# dfsioe-read
RD_NUM_OF_FILES=256
RD_FILE_SIZE=200 #2000

# dfsioe-write
WT_NUM_OF_FILES=256
WT_FILE_SIZE=100 #1000
```

Sort

```
# compress
COMPRESS=$COMPRESS_GLOBAL
COMPRESS_CODEC=$COMPRESS_CODEC_GLOBAL

# paths
INPUT_HDFS=${DATA_HDFS}/Sort/Input
OUTPUT_HDFS=${DATA_HDFS}/Sort/Output

if [ $COMPRESS -eq 1 ]; then
    INPUT_HDFS=${INPUT_HDFS}-comp
    OUTPUT_HDFS=${OUTPUT_HDFS}-comp
fi

# for prepare (per node) - 24G/node
#DATASIZE=24000000000
DATASIZE=2400000000
NUM_MAPS=16

# for running (in total)
NUM_REDS=48
```

WordCount

```
# compress
# for best performance set COMPRESS=1 for MR1 and COMPRESS=0 for MR2 (for WordCount)
COMPRESS=$COMPRESS_GLOBAL
COMPRESS_CODEC=$COMPRESS_CODEC_GLOBAL

# paths
INPUT_HDFS=${DATA_HDFS}/Wordcount/Input
OUTPUT_HDFS=${DATA_HDFS}/Wordcount/Output
```

```
9
10 if [ $COMPRESS -eq 1 ]; then
11     INPUT_HDFS=${INPUT_HDFS}-comp
12     OUTPUT_HDFS=${OUTPUT_HDFS}-comp
13 fi
14
15 # for preparation (per node) - 32G
16 #DATASIZE=32000000000
17 DATASIZE=3200000000
18 NUM_MAPS=16
19
20 # for running (in total)
21 NUM_REDS=48
```

k-means

```
1  # compress
2  COMPRESS=$COMPRESS_GLOBAL
3  COMPRESS_CODEC=$COMPRESS_CODEC_GLOBAL
4
5  # paths
6  INPUT_HDFS=${DATA_HDFS}/KMeans/Input
7  OUTPUT_HDFS=${DATA_HDFS}/KMeans/Output
8  if [ $COMPRESS -eq 1 ]; then
9      INPUT_HDFS=${INPUT_HDFS}-comp
10     OUTPUT_HDFS=${OUTPUT_HDFS}-comp
11 fi
12 INPUT_SAMPLE=${INPUT_HDFS}/samples
13 INPUT_CLUSTER=${INPUT_HDFS}/cluster
14
15 # for prepare
16 NUM_OF_CLUSTERS=5
17 #NUM_OF_SAMPLES=20000000
18 NUM_OF_SAMPLES=3000000
19 #SAMPLES_PER_INPUTFILE=4000000
20 SAMPLES_PER_INPUTFILE=600000
21 DIMENSIONS=20
22
23 # for running
24 MAX_ITERATION=5
```

3.1.5 References

CHAPTER

FOUR

LINUX TOOLS

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search