

HADOOP CLUSTER MAINTAINANCE

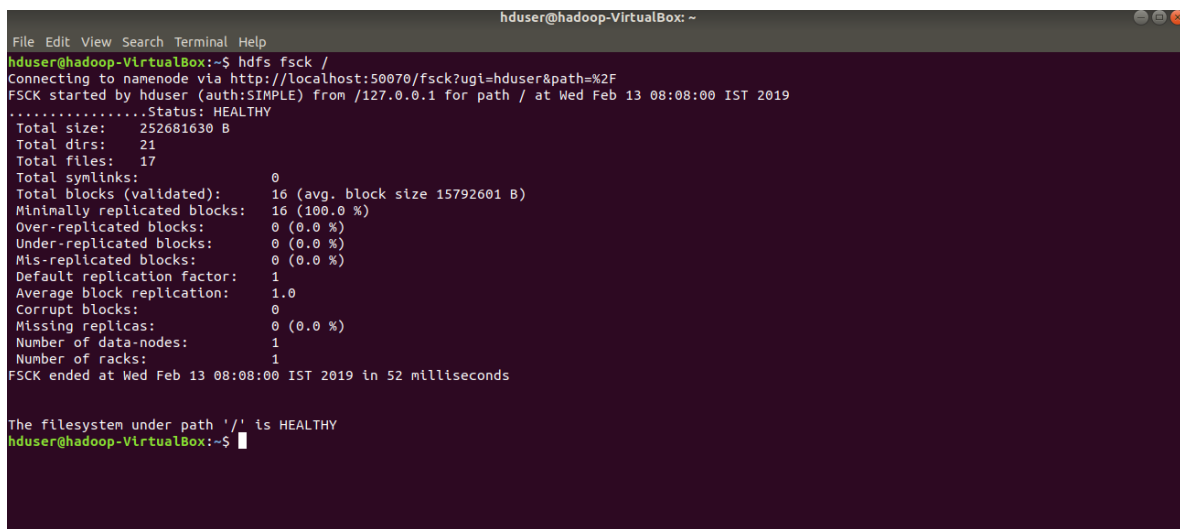
Hadoop Admin responsibility to perform Hadoop Cluster Maintenance frequently. Let's see what we can do to keep our **Big Elephant Happy!**

1. FILE SYSTEM CHECKS

Develop a habit of Check health of HDFS periodically by running `-fsck` command

hdfs fsck /

This command contacts the Namenode and checks each file recursively which comes under the provided path. Below is the sample output of `-fsck` command.



```
hduser@hadoop-VirtualBox:~$ hdfs fsck /
Connecting to namenode via http://localhost:50070/fsck?ugi=hduser&path=%2F
FSCK started by hduser (auth:SIMPLE) from /127.0.0.1 for path / at Wed Feb 13 08:08:00 IST 2019
.....Status: HEALTHY
Total size:      252681630 B
Total dirs:      21
Total files:     17
Total symlinks:   0
Total blocks (validated): 16 (avg. block size 15792601 B)
Minimally replicated blocks: 16 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Wed Feb 13 08:08:00 IST 2019 in 52 milliseconds

The filesystem under path '/' is HEALTHY
hduser@hadoop-VirtualBox:~$
```

Other Usage:

Displays all the files in HDFS while checking

hadoop fsck / -files

Displays all the blocks of the files while checking

hadoop fsck / -files -blocks

Displays all the files block locations while checking

hadoop fsck / -files -blocks -locations

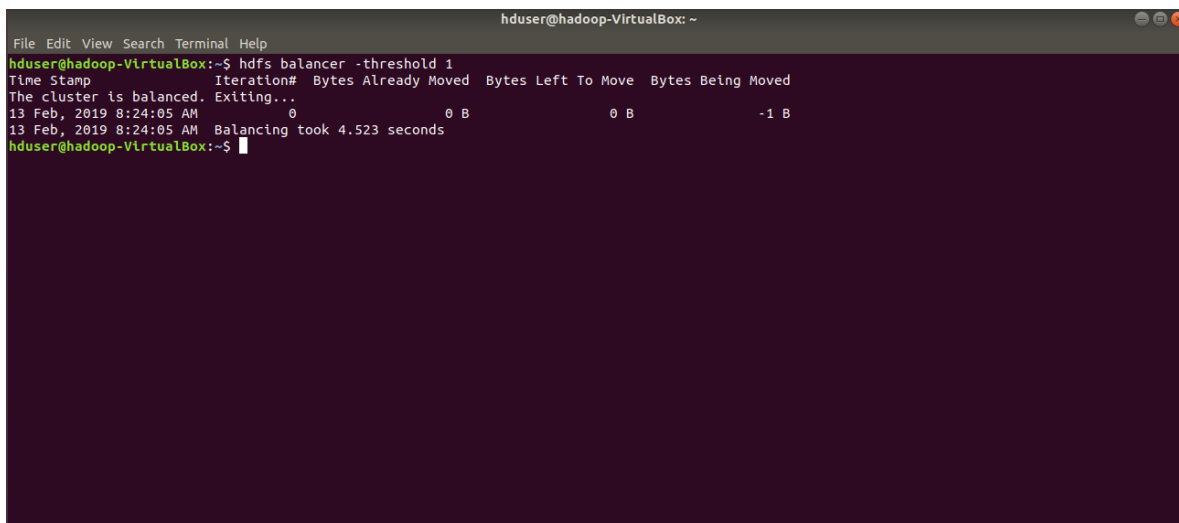
Command is used to display the networking topology for data-node locations

```
hadoop fsck / -files -blocks -locations -racks
```

HDFS Balancer Utility

Over the period of time data becomes un-balanced across all Data Nodes in the Hadoop Cluster, this could be because of maintenance activity on specific Data Nodes, power failure, hardware failures, kernel panic, unexpected reboots etc. In this case because of data locality, Datanodes which are having more data will get churned and ultimately ***un-balanced cluster can directly affect your MapReduce job performance.***

```
hdfs balancer -threshold 1
```

A screenshot of a terminal window titled 'hduser@hadoop-VirtualBox: ~'. The terminal shows the command 'hdfs balancer -threshold 1' being executed. The output includes a table with columns: 'Time Stamp', 'Iteration#', 'Bytes Already Moved', 'Bytes Left To Move', and 'Bytes Being Moved'. The first row shows '13 Feb, 2019 8:24:05 AM', '0', '0 B', '0 B', and '-1 B'. Below the table, it says 'The cluster is balanced. Exiting..' and 'Balancing took 4.523 seconds'. The prompt 'hduser@hadoop-VirtualBox:~\$' is visible at the bottom.

```
File Edit View Search Terminal Help
hduser@hadoop-VirtualBox:~$ hdfs balancer -threshold 1
Time Stamp      Iteration#  Bytes Already Moved  Bytes Left To Move  Bytes Being Moved
The cluster is balanced. Exiting..
13 Feb, 2019 8:24:05 AM      0              0 B                0 B                -1 B
13 Feb, 2019 8:24:05 AM Balancing took 4.523 seconds
hduser@hadoop-VirtualBox:~$
```

By default threshold value is 10, we can reduce it upto 1 (It's better to run balancer with lowest threshold)

Adjust the network bandwidth used by the balancer, by running the dfsadmin - setBalancerBandwidth command before you run the balancer

For example:

```
dfsadmin -setBalancerBandwidth newbandwidth
```

PERFORMANCE TUNING

Let's get into the game now:

1. Resource Manager (RM) is responsible for allocating resources to mapreduce jobs.
2. For brand new Hadoop cluster (without any tuning) resource manager will get 8192MB ("yarn.nodemanager.resource.memory-mb") memory per node only.
3. RM can allocate up to 8192 MB ("yarn.scheduler.maximum-allocation-mb") to the Application Master container.
4. Default minimum allocation is 1024 MB ("yarn.scheduler.minimum-allocation-mb").
5. The AM can only negotiate resources from Resource Manager that are in increments of ("yarn.scheduler.minimum-allocation-mb") & it cannot exceed ("yarn.scheduler.maximum-allocation-mb").
6. Application Master Rounds off ("mapreduce.map.memory.mb") & ("mapreduce.reduce.memory.mb") to a value devisable by ("yarn.scheduler.minimum-allocation-mb").

What are these properties ? What can we tune ?

yarn.scheduler.minimum-allocation-mb

Sets the minimum size of container that YARN will allow for running mapreduce jobs.

Default value is 1024m

yarn.scheduler.maximum-allocation-mb

The largest size of container that YARN will allow us to run the Mapreduce jobs.

Default value is 8192m

yarn.nodemanager.resource.memory-mb

Default value is 8GB

Total amount of physical memory (RAM) for Containers on Data Node.

Set this property= Total RAM - (RAM for OS + Hadoop Daemons + Other services)

yarn.nodemanager.vmem-pmem-ratio

Default value is 2.1

The amount of virtual memory that each Container is allowed

This can be calculated with: `containerMemoryRequest*vmem-pmem-ratio`

mapreduce.map.memory.mb

mapreduce.reduce.memory.mb

These are the hard limits enforced by Hadoop on each mapper or reducer task. (Maximum memory that can be assigned to mapper or reducer's container)

Default value - 1GB

mapreduce.map.java.opts

mapreduce.reduce.java.opts

The heapsize of the jvm -Xmx for the mapper or reducer task.

This value should always be lower than mapreduce.[map|reduce].memory.mb.

Recommended value is 80% of mapreduce.map.memory.mb/ mapreduce.reduce.memory.mb

yarn.app.mapreduce.am.resource.mb

The amount of memory for ApplicationMaster

yarn.app.mapreduce.am.command-opts

heapsize for application Master

yarn.nodemanager.resource.cpu-vcores

The number of cores that a node manager can allocate to containers is controlled by the `yarn.nodemanager.resource.cpu-vcores` property. It should be set to the total number of cores on the machine, minus a core for each daemon process running on the machine (datanode, node manager, and any other long-running processes).

mapreduce.task.io.sort.mb

Default value - 100MB

This is very important property to tune, when map task is in progress it writes output into a circular in-memory buffer. The size of this buffer is fixed and determined by `io.sort.mb` property

When this circular in-memory buffer gets filled (`mapreduce.map.sort.spill.percent`: 80% by default), the SPILLING to disk will start (in parallel using a separate thread). Notice that if the spilling thread is too slow and the buffer is 100% full, then the map cannot be executed and thus it has to wait.

io.file.buffer.size

Hadoop uses buffer size of 4KB by default for its I/O operations, we can increase it to 128K in order to get good performance and this value can be increased by setting `io.file.buffer.size= 131072` (value in bytes) in `core-site.xml`

dfs.client.read.shortcircuit

Short-circuit reads - When reading a file from HDFS, the client contacts the datanode and the data is sent to the client via a TCP connection. If the block being read is on the same node as the client, then it is more efficient for the client to bypass the network and read the block data directly from the disk.

We can enable short-circuit reads by setting this property to "true"

mapreduce.task.io.sort.factor

Default value is 10.

Now imagine the situation where map task is running, each time the memory buffer reaches the spill threshold, a new spill file is created, after the map task has written its last output record, there could be several spill files. Before the task is finished, the spill files are merged into a single partitioned and sorted output file.

The configuration property `mapreduce.task.io.sort.factor` controls the maximum number of streams to merge at once.

mapreduce.reduce.shuffle.parallelcopies

Default value is 5

The map output file is sitting on the local disk of the machine that ran the map task

The map tasks may finish at different times, so the reduce task starts copying their outputs as soon as each completes

The reduce task has a small number of copier threads so that it can fetch map outputs in parallel.

The default is five threads, but this number can be changed by setting the `mapreduce.reduce.shuffle.parallelcopies` property