**Tune Hadoop Cluster to get Maximum Performance (Part 1)**

**Author:**Kuldeep Kulkarni /

**Posted on:**June 28, 2015 /

**Categories:**[Hadoop](http://crazyadmins.com/category/hadoop/)

I have been working on production Hadoop clusters for a while and have learned many performance tuning tips and tricks. In this blog I will explain how to tune Hadoop Cluster to get maximum performance. Just installing Hadoop for production clusters or to do some development POC does not give expected results, because default Hadoop configuration settings are done keeping in mind the minimum hardware configuration. Its responsibility of Hadoop Administrator to understand the hardware specs like amount of RAM, Total number of CPU Cores, Physical Cores, Virtual Cores, Understand if hyper threading is supported by Processor, NIC Cards, Number of Disks that are mounted on Datanodes etc.

[](http://crazyadmins.com/wp-content/uploads/2015/06/tune.jpg)

**For Better Understanding I have divided this blog into two main parts.**

1. Tune your Hadoop Cluster to get Maximum Performance (Part 1) – In this part I will explain how to tune your operating system in order to get maximum performance for your Hadoop jobs.

2. Tune your Hadoop Cluster to get Maximum Performance (Part 2) – In this part I will explain how to modify your Hadoop configurations parameters so that it should use your hardware very efficiently.

**How OS tuning will improve performance of Hadoop?**

Tuning your Centos6.X/Redhat 6.X can increase performance of Hadoop by 20-25%. Yes! 20-25% 

Let’s get started and see what parameters we need to change on OS level.

**1. Turn off the Power savings option in BIOS:**

This will increase the overall system performance and Hadoop Performance. You can go to your BIOS Settings and change it to PerfOptimized from power saving mode (this option may be different for your server based on vendor). If you have remote console command line available then you can use racadm commands to check the status and update it. You need to restart the system in order to get your changes in effect.

**2. Open file handles and files:**

By default number of open file count is 1024 for each user and if you keep it to default then you may face java.io.FileNotFoundException: (Too many open files) and your job will get failed. In order to avoid this scenario set this number of open file limit to unlimited or some higher number like 32832.

Commands:

ulimit –S 4096

ulimit –H 32832

Also, Please set the system wide file descriptors by using below command:

sysctl –w fs.file-max=6544018

As above kernel variable is temporary and we need to make it permanent by adding it to /etc/sysctl.conf. Just edit /etc/sysctl.conf and add below value at the end of it

fs.file-max=6544018

**3. FileSystem Type & Reserved Space:**

In order to get maximum performance for your Hadoop job, I personally suggest by using **ext4** filesystem as it has some advantage over ext3 like multi-block and delayed allocation etc. How you mount your file-system will make difference because if you mount it using default option then there will excessive writes for file or directory access times which we do not need in case of Hadoop. Mount your local disks using option **noatime** will surely improve your performance by disabling those excessive and unnecessary writes to disks.

Below is the sample of how it should look like:

UUID=gfd3f77-6b11-4ba0-8df9-75feb03476efs /disk1                 ext4   noatime       0 0

UUID=524cb4ff-a2a5-4e8b-bd24-0bbfd829a099 /disk2                 ext4   noatime       0 0

UUID=s04877f0-40e0-4f5e-a15b-b9e4b0d94bb6 /disk3                 ext4   noatime       0 0

UUID=3420618c-dd41-4b58-ab23-476b719aaes /disk4                 ext4   noatime       0 0

**Note – noatime option will also cover noadirtime so no need to mention that.**

Many of you must be aware that after formatting your disk partition with ext4 partition, there is 5% space reserved for special operations like 100% disk full so root should be able to delete the files by using this reserved space. In case of Hadoop we don’t need to reserve that 5% space so please get it removed using tune2fs command.

Command:

tune2fs -m 0 /dev/sdXY

**Note – 0 indicates that 0% space is reserved.**

**4. Network Parameters Tuning:**

Network parameters tuning also helps to get performance boost! This is kinda risky stuff because if you are working on remote server and you did a mistake while updating Network parameters then there can be a connectivity loss and you may not be able to connect to that server again unless you correct the configuration mistake by taking IPMI/iDrac/iLo console etc. Modify the parameters only when you know what you are doing 

Modifying the **net.core.somaxconn** to **1024** from default value of **128** will help Hadoop by as this changes will have increased listening queue between the master and slave services so ultimately number of connections between master and slaves will be higher than before.

Command to modify net.core.somaxconnection:

sysctl –w net.core.somaxconn=1024

To make above change permanent, simple add below variable value at the end of /etc/sysctl.conf

net.core.somaxconn=1024

**MTU Settings:**

Maximum transmission unit. This value indicates the size which can be sent in a packet/frame over TCP. By default MTU is set to 1500 and you can tune it have its value=9000, when value of MTU is greater than its default value then it’s called as Jumbo Frames.

Command to change value of MTU:

You need to add MTU=9000 in /etc/sysconfig/network-scripts/ifcfg-eth0 or whatever your eth device name. Restart the network service in order to have this change in effect.

**Note – Before modifying this value please make sure that all the nodes in your cluster including switches are supported for jumbo frames, if not then \*PLEASE DO NOT ATTEMPT THIS\***

**5. Transparent Huge Page Compaction:**

This feature of linux is really helpful to get the better performance for application including Hadoop workloads however one of the subpart of Transparent Huge Pages called **Compaction** causes issues with Hadoop job(it causes high processor usage while defragmentation of the memory). When I was benchmarking our client’s cluster I have observed some fluctuations ~15% with the output and when I disabled it then that fluctuation was gone. So I recommend to disable it for Hadoop.

Command:

echo never > /sys/kernel/mm/redhat\_transparent\_hugepages/defrag

In order to make above change permanent, please add below script in your /etc/rc.local file.

if test -f /sys/kernel/mm/redhat\_transparent\_hugepage/defrag; then echo never > /sys/kernel/mm/redhat\_transparent\_hugepage/defrag ;fi

**6. Memory Swapping:**

For Hadoop Swapping reduces the job performance, you should have maximum data in-memory and tune your OS so that it will do memory swap only when there is situation like OOM (OutOfMemory). To do so we need to set vm.swappiness kernel parameter to 0

Command:

sysctl -w vm.swappiness=0

Please add below variable in /etc/sysctl.conf to make it persistent.

vm.swappiness=0

<http://crazyadmins.com/tune-hadoop-cluster-to-get-maximum-performance-part-1/>

# Tune Hadoop Cluster to get Maximum Performance (Part 2)

**Author:**Kuldeep Kulkarni /

**Posted on:**July 19, 2015 /

**Categories:**[Hadoop](http://crazyadmins.com/category/hadoop/)

# In [previous part](http://crazyadmins.com/tune-hadoop-cluster-to-get-maximum-performance-part-1/) we have seen that how can we tune our operating system to get maximum performance for Hadoop, in this article I will be focusing on how to tune hadoop cluster to get performance boost on hadoop level :-)

[](http://crazyadmins.com/wp-content/uploads/2015/06/tune.jpg)

Tune Hadoop Cluster to get Maximum Performance (Part 2) – http://crazyadmins.com

# Before I actually start explaining tuning parameters let me cover some basic terms that are required to understand Hadoop level tuning.

# What is YARN?

### YARN – Yet another resource negotiator, this is Map-reduce version 2 with many new features such as dynamic memory assignment for mappers and reducers rather than having fixed slots etc.

# What is Container?

### Container represents allocated Resources like CPU, RAM etc. It’s a JVM process, in YARN AppMaster, Mapper and Reducer runs inside the Container.

# 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

### Default value is 1024m

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

yarn.scheduler.maximum-allocation-mb

### Default value is 8192m

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

yarn.nodemanager.resource.memory-mb

### Default value is 8GB Total amount of physical memory (RAM) for Containers on worker 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 splilling 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