Determine HDP Memory Configuration Settings

Two methods can be used to determine YARN and MapReduce memory configuration settings:

Running the Yarn Utility Script

Manually Calculating YARN and MapReduce Memory Configuration Settings

The HDP utility script is the recommended method for calculating HDP memory configuration settings, but information about manually calculating YARN and MapReduce memory configuration settings is also provided for reference.

Running the Yarn Utility Script

This section describes how to use the hdp-configuration-utils.py script to calculate YARN, MapReduce, Hive, and Tez memory allocation settings based on the node hardware specifications. The hdp-configuration-utils.py script is included in the HDP companion files. See Download Companion Files.

To run the hdp-configuration-utils.py script, execute the following command from the folder containing the script `hdp-configuration-utils.py options` where options are as follows:

Table 1.5. hdp-configuration-utils.py Options

Option	Description
-c CORES	The number of cores on each host.
-m MEMORY	The amount of memory on each host in GB.
-d DISKS	The number of disks on each host.
-k HBASE	"True" if HBase is installed, "False" if not.

Note
Requires python26 to run.

You can also use the -h or --help option to display a Help message that describes the options.

Example

Running the following command from the hdp_manual_install_rpm_helper_files-2.3.2.0.2950 directory:

```
python hdp-configuration-utils.py -c 16 -m 64 -d 4 -k True
```

Returns:

```
Using cores=16 memory=64GB disks=4 hbase=True
Profile: cores=16 memory=49152MB reserved=16GB usableMem=48GB disks=4
Num Container=8
Container Ram=6144MB
Used Ram=48GB
Unused Ram=16GB
yarn.scheduler.minimum-allocation-mb=6144
yarn.scheduler.maximum-allocation-mb=49152
yarn.nodemanager.resource.memory-mb=49152
mapreduce.map.memory.mb=6144
mapreduce.map.java.opts=-Xmx4096m
mapreduce.reduce.memory.mb=6144
mapreduce.reduce.java.opts=-Xmx4096m
yarn.app.mapreduce.am.resource.mb=6144
yarn.app.mapreduce.am.command-opts=-Xmx4096m
mapreduce.task.io.sort.mb=1792
tez.am.resource.memory.mb=6144
tez.am.launch.cmd-opts =-Xmx4096m
```

```
hive.tez.container.size=6144
hive.tez.java.opts=-Xmx4096m
```

Manually Calculating YARN and MapReduce Memory Configuration Settings

This section describes how to manually configure YARN and MapReduce memory allocation settings based on the node hardware specifications.

YARN takes into account all of the available compute resources on each machine in the cluster. Based on the available resources, YARN negotiates resource requests from applications running in the cluster such as MapReduce. YARN then provides processing capacity to each application by allocating Containers. A Container is the basic unit of processing capacity in YARN, and is an encapsulation of resource elements such as memory and CPU.

In a Hadoop cluster, it is vital to balance the use of memory (RAM), processors (CPU cores), and disks so that processing is not constrained by any one of these cluster resources. As a general recommendation, allowing for two Containers per disk and per core gives the best balance for cluster utilization.

When determining the appropriate YARN and MapReduce memory configurations for a cluster node, start with the available hardware resources. Specifically, note the following values on each node:

RAM (Amount of memory)

CORES (Number of CPU cores)

DISKS (Number of disks)

The total available RAM for YARN and MapReduce should take into account the Reserved Memory. Reserved Memory is the RAM needed by system processes and other Hadoop processes (such as HBase).

Reserved Memory = Reserved for stack memory + Reserved for HBase Memory (If HBase is on the same node).

Use the following table to determine the Reserved Memory per node.

Table 1.6. Reserved Memory Recommendations

Total Memory per Node	Recommended Reserved System Memory	Recommended Reserved HBase Memory
4 GB	1 GB	1 GB
8 GB	2 GB	1 GB
16 GB 24 GB	2 GB 4 GB	2 GB 4 GB
48 GB	6 GB	8 GB
64 GB	8 GB	8 GB
72 GB	8 GB	8 GB
96 GB 128 GB	12 GB 24 GB	16 GB 24 GB
256 GB	32 GB	32 GB
512 GB	64 GB	64 GB

The next calculation is to determine the maximum number of containers allowed per node. The following formula can be used:

# of containers = min (2*CORES, 1.8*DISKS, (Total available RAM) / MIN_CONTAINER_SIZE)

Where DISKS is the value for dfs.data.dirs (number of data disks) per machine.

And MIN_CONTAINER_SIZE is the minimum container size (in RAM). This value is dependent on the amount of RAM available -- in smaller memory nodes, the minimum container size should also be smaller. The following table outlines the recommended values:

Table 1.7. Recommended Values

Total RAM per Node	Recommended Minimum Container Size
Less than 4 GB	256 MB
Between 4 GB and 8 GB	512 MB
Between 8 GB and 24 GB	1024 MB
Above 24 GB	2048 MB

The final calculation is to determine the amount of RAM per container:

RAM-per-container = max(MIN_CONTAINER_SIZE, (Total Available RAM) / containers))

With these calculations, the YARN and MapReduce configurations can be set.

Table 1.8. YARN and MapReduce Configuration Setting Value Calculations

Configuration File	Configuration Setting	Value Calculation
yarn-site.xml	yarn.nodemanager.resource.memory-mb	= containers * RAM-per-container
yarn-site.xml	yarn.scheduler.minimum-allocation-mb	= RAM-per-container
yarn-site.xml	yarn.scheduler.maximum-allocation-mb	= containers * RAM-per-container
mapred-site.xml	mapreduce.map.memory.mb	= RAM-per-container
mapred-site.xml	mapreduce.reduce.memory.mb	= 2 * RAM-per-container
mapred-site.xml	mapreduce.map.java.opts	= 0.8 * RAM-per-container
mapred-site.xml	mapreduce.reduce.java.opts	= 0.8 * 2 * RAM-per-container
mapred-site.xml	yarn.app.mapreduce.am.resource.mb	= 2 * RAM-per-container
mapred-site.xml	yarn.app.mapreduce.am.command-opts	= 0.8 * 2 * RAM-per-container

Note: After installation, both yarn-site.xml and mapred-site.xml are located in the `/etc/hadoop/conf` folder.

Examples

Cluster nodes have 12 CPU cores, 48 GB RAM, and 12 disks.

Reserved Memory = 6 GB reserved for system memory + (if HBase) 8 GB for HBase Min container size = 2 GB

If there is no HBase:

# of containers = min (2*12, 1.8* 12, (48-6)/2) = min (24, 21.6, 21) = 21

RAM-per-container = max (2, (48-6)/21) = max (2, 2) = 2

Table 1.9. Example Value Calculations

Configuration	Value Calculation
yarn.nodemanager.resource.memory-mb	= 21 * 2 = 42*1024 MB
yarn.scheduler.minimum-allocation-mb	= 2*1024 MB
yarn.scheduler.maximum-allocation-mb	= 21 * 2 = 42*1024 MB
mapreduce.map.memory.mb	= 2*1024 MB
mapreduce.reduce.memory.mb	= 2 * 2 = 4*1024 MB
mapreduce.map.java.opts	= 0.8 * 2 = 1.6*1024 MB
mapreduce.reduce.java.opts	= 0.8 * 2 * 2 = 3.2*1024 MB
yarn.app.mapreduce.am.resource.mb	= 2 * 2 = 4*1024 MB
yarn.app.mapreduce.am.command-opts	= 0.8 * 2 * 2 = 3.2*1024 MB

If HBase is included:

# of containers = min (2*12, 1.8* 12, (48-6-8)/2) = min (24, 21.6, 17) = 17

RAM-per-container = max (2, (48-6-8)/17) = max (2, 2) = 2

Table 1.10. Example Value Calculations

Configuration	Value Calculation
yarn.nodemanager.resource.memory-mb	= 17 * 2 = 34*1024 MB
yarn.scheduler.minimum-allocation-mb	= 2*1024 MB
yarn.scheduler.maximum-allocation-mb	= 17 * 2 = 34*1024 MB
mapreduce.map.memory.mb	= 2*1024 MB
mapreduce.reduce.memory.mb	= 2 * 2 = 4*1024 MB
mapreduce.map.java.opts	= 0.8 * 2 = 1.6*1024 MB
mapreduce.reduce.java.opts	= 0.8 * 2 * 2 = 3.2*1024 MB
yarn.app.mapreduce.am.resource.mb	= 2 * 2 = 4*1024 MB
yarn.app.mapreduce.am.command-opts	= 0.8 * 2 * 2 = 3.2*1024 MB

Notes:

Updating values for yarn.scheduler.minimum-allocation-mb without also changing yarn.nodemanager.resource.memory-mb, or changing yarn.nodemanager.resource.memory-mb without also changing yarn.scheduler.minimum-allocation-mb changes the number of containers per node.

If your installation has a large amount of RAM but not many disks/cores, you can free up RAM for other tasks by lowering both yarn.scheduler.minimum-allocation-mb and yarn.nodemanager.resource.memory-mb.

With MapReduce on YARN, there are no longer pre-configured static slots for Map and Reduce tasks. The entire cluster is available for dynamic resource allocation of Map and Reduce tasks as needed by each job. In our example cluster, with the above configurations, YARN will be able to allocate up to 10 Mappers (40/4) or 5 Reducers (40/8) on each node (or some other combination of Mappers and Reducers within the 40 GB per node limit).