2. Hadoop Installation

April 20, 2021

1 Hadoop Installation and Testing

WARNING: Be careful, the execution of this notebook can compromise your virtual machines. Do not execute any cell twice: please start from the very first cell if you have problems.

This notebook requires password-less SSH from the machine IT WILL BE EXECUTED ON to the machines in your cluster.

This notebook automatically works on Linux machines only. Windows users must configure the machines by hand.

Load commands from `commands.py`.

[ ]:
```
%reload_ext autoreload
%autoreload 1
%aimport commands
```

Functions we will use:

[ ]:
```
def run_ssh(host, *command):
    """
    :param host: ip address
    :type f: string
    :param command: command to execute
    :type command: string

    Execute with SSH the commands on host as the 'hadoop' user, displaying␣
 ↪information
    """
    print('===== \x1b[31m' + 'Started on ' + host + '\x1b[0m =====')
    for cmd in command:
        print(cmd)
        !ssh hadoop@{host} {cmd}
    print('===== \x1b[31m' + 'Completed on ' + host + '\x1b[0m =====')
```

Generally Hadoop can be run in three modes.

1. **Standalone (or local) mode**: There are no daemons used in this mode. Hadoop uses the local file system as a substitute for HDFS file system. The jobs will run as if there is 1 mapper and 1 reducer.

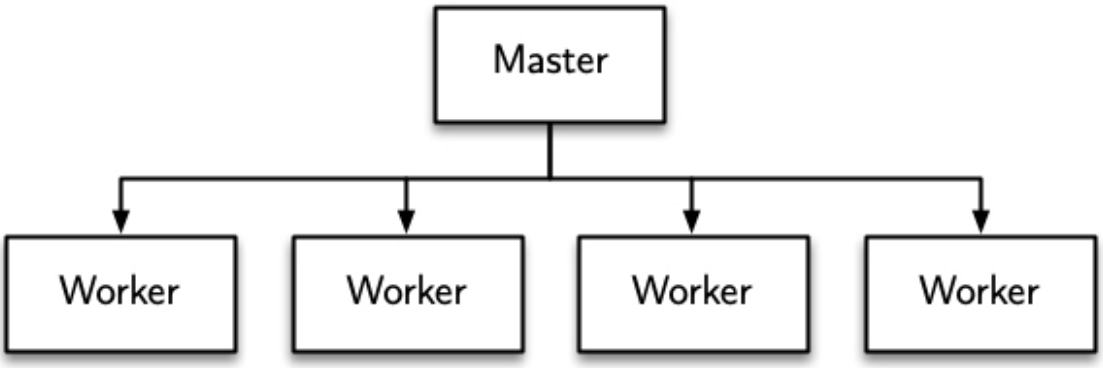
2. **Pseudo-distributed mode**: All the daemons run on a single machine and this setting mimics the behavior of a cluster. All the daemons run on your machine locally using the HDFS protocol. There can be multiple mappers and reducers.
3. **Fully-distributed mode**: This is how Hadoop runs on a real cluster.

In these notes we will describe how to set up an Hadoop 3 installation to work with. We will set up a *fully-distributed cluster* on your assigned virtual machines.

The core of Hadoop is composed by two main subsystem:

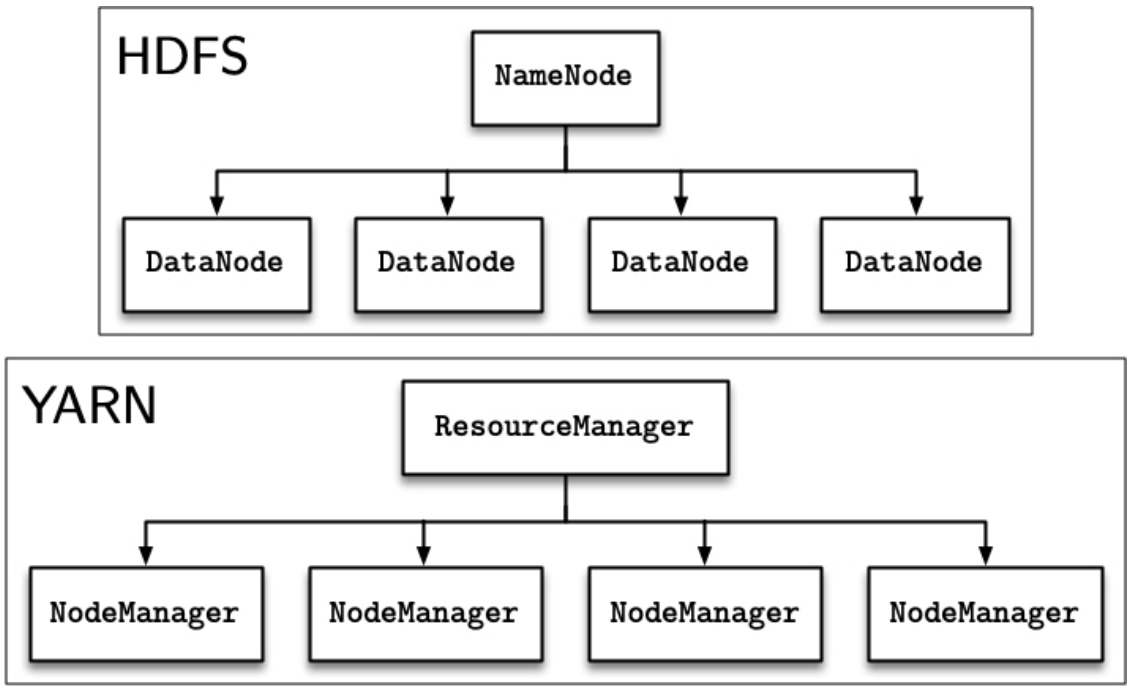
- the **Hadoop Distributed File System** (HDFS), responsible for the distributed data management
- the **Yet Another Resource Negotiator** (YARN), responsible for the distributed code execution

Both subsystems are implemented according to the **master-workers** architecture.



Both HDFS and YARN have their own terminology for master and worker nodes.

	Master	Worker
HDFS	NameNode	DataNode
YARN	ResouceManager	NodeManager



While the masters of HDFS and YARN can, in principle, be located on different machines, we will install the HDFS and YARN masters on a single machine, and install the HDFS and YARN workers on all machines (*including the machine hosting the master*).

This notebook contains the steps necessary to set up and configure correctly Hadoop on our virtual machines. In particular:

1. We will download and install Hadoop on all our virtual machines.
2. We will configure a virtual machine to host the HDFS and YARN masters and workers.
3. We will configure the remaining virtual machines to host the HDFS and YARN workers.
4. We will test your newly install Hadoop cluster.

1.1 0. Prerequisites

a. Populate the following dictionary with the IP addresses (as keys) and the hostnames (as values) of all your virtual machines.

[ ]:
```
VMS = {'172.16.3.79': 'datanode1',
       '172.16.3.80': 'datanode2',
       '172.16.3.81': 'namenode'}
```

b. Populate the following variable with the IP address of the virtual machine with the namenode role.

[ ]:
```
NAMENODE_IP = '172.16.3.81'
```

c. Populate the following variable with the IP address of the remaining virtual machines.

[ ]:
```
REMAINING_IPS = [
    '172.16.3.79',
    '172.16.3.80'
]
```

1.2 ## 1. Download and install Hadoop

a. Download hadoop-3.1.3.tar.gz in your home folder on your virtual machines.

[ ]:
```
for host in VMS:
    run_ssh(host, commands.WGET_CMD)
```

b. Decompress the Hadoop package you can use the following command:

[ ]:
```
for host in VMS:
    run_ssh(host, commands.TAR_CMD, commands.RM_CMD)
```

c. There are environment settings that will be used by Hadoop. The following lines myst be appended to your `/home/hadoop/.bashrc` file **on each machine**:

```
export HADOOP_HOME=/opt/hadoop
export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:$HADOOP_HOME/bin:$HADO
export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native:$LD_LIBRARY_PATH
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HDFS_NAMENODE_USER=hadoop
export HDFS_DATANODE_USER=hadoop
export HDFS_SECONDARYNAMENODE_USER=hadoop
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_LOG_DIR=$HADOOP_HOME/logs
```

The following commands append the correct environment variables to your `/home/hadoop/.bashrc` files:

[ ]:
```
for host in VMS:
    !ssh hadoop@{host} "sed -i '1,10 s/^/#/' ~/.bashrc"
    !ssh hadoop@{host} 'printf "%s\n" {commands.get_bashrc()} >> ~/.bashrc'
```

d. The following commands check Hadoop installation (you should see no errors):

[ ]:
```
for host in VMS:
    run_ssh(host, 'hadoop version')
```

1.3 2. Configure the `hadoop-namenode` machine

a. Update the `core-site.xml` file located at `/opt/hadoop/etc/hadoop/` to define the name node URI on this machine. The file must contain the following lines.

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://hadoop-namenode:9820/</value>
  </property>
</configuration>
```

The following command updates the file automatically.

```
[ ]: !ssh hadoop@{NAMENODE_IP} 'printf "%s\n" {commands.
     ↪get_namenode_core_site(VMS[NAMENODE_IP])} > /opt/hadoop/etc/hadoop/core-site.
     ↪xml'
```

b. Update the `hdfs-site.xml` file located at `/opt/hadoop/etc/hadoop/` to define the path on the local filesystem where the name node stores the namespace and transactions logs persistently and to configure the HDFS subsystem. The file must contain the following lines.

```
<configuration>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///opt/hdfs/namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///opt/hdfs/datanode</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
  <property>
    <name>dfs.datanode.use.datanode.hostname</name>
    <value>false</value>
  </property>
</configuration>
```

The following command updates the file automatically.

```
[ ]: !ssh hadoop@{NAMENODE_IP} 'printf "%s\n" {commands.get_namenode_hdfs_site()} > /
     ↪opt/hadoop/etc/hadoop/hdfs-site.xml'
```

c. Update the `yarn-site.xml` file located at `/opt/hadoop/etc/hadoop` to configure the YARN subsystem. The file must contain the following lines.

```
<configuration>
  <property>
```

```
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>{hadoop_namenode}</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.nodemanager.local-dirs</name>
    <value>file:///opt/yarn/local</value>
  </property>
  <property>
    <name>yarn.nodemanager.log-dirs</name>
    <value>file:///opt/yarn/logs</value>
  </property>
  <property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <value>1536</value>
  </property>
  <property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>1536</value>
  </property>
  <property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>128</value>
  </property>
  <property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
  </property>
</configuration>
```

The following command updates the file automatically.

[ ]:
```
!ssh hadoop@{NAMENODE_IP} 'printf "%s\n" {commands.
↪get_namenode_yarn_site(VMS[NAMENODE_IP])} > /opt/hadoop/etc/hadoop/yarn-site.
↪xml'
```

d. Update the `mapred-site.xml` file located at `/opt/hadoop/etc/hadoop` to configure the MAPREDUCE subsystem. The file must contain the following lines.

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
```

```
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.address</name>
    <value>{namenode_hostname}:10020</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>{namenode_hostname}:19888</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.intermediate-done-dir</name>
    <value>/mr-history/tmp</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.done-dir</name>
    <value>/mr-history/done</value>
  </property>
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
  <property>
    <name>yarn.app.mapreduce.am.resource.mb</name>
    <value>512</value>
  </property>
  <property>
    <name>mapreduce.map.memory.mb</name>
    <value>256</value>
  </property>
  <property>
    <name>mapreduce.reduce.memory.mb</name>
    <value>256</value>
  </property>
</configuration>
```

The following command updates the file automatically.

[ ]:

```
!ssh hadoop@{NAMENODE_IP} 'printf "%s\n" {commands.
↪get_namenode_mapred_site(VMS[NAMENODE_IP])} > /opt/hadoop/etc/hadoop/
↪mapred-site.xml'
```

If your machines have more than 2 GB of RAM or if you are interested in the numbers we specified in the YARN and MAPRED configuration files, check the Appendix A on the Hadoop Memory Allocaltion.

e. Update the `workers` file located in `/opt/hadoop/etc/hadoop` to define the MAPREDUCE workers. With our virtual machines listed here, the file must contain the following lines.

```
172.16.0.225
172.16.0.221
172.16.0.224
```

The following command updates the file automatically.

[ ]:
```
!ssh hadoop@{NAMENODE_IP} 'printf "%s\n" {commands.get_workers(VMS)} > /opt/
↪hadoop/etc/hadoop/workers'
```

1.4 3. Configure the `hadoop-datanode` machines

a. Update the `core-site.xml` file located at `/opt/hadoop/etc/hadoop/` to define the name node URI on thie other datanodes. The file must contain the following lines.

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://hadoop-namenode:9820/</value>
  </property>
</configuration>
```

The following command updates the file automatically.

[ ]:
```
for host in REMAINING_IPS:
    !ssh hadoop@{host} 'printf "%s\n" {commands.
↪get_datanode_core_site(VMS[NAMENODE_IP])} > /opt/hadoop/etc/hadoop/core-site.
↪xml'
```

b. Update the `hdfs-site.xml` file located at `/opt/hadoop/etc/hadoop/` to configure the HDFS subsystem. The file must contain the following lines.

```
<configuration>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///opt/hdfs/datanode</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
```

```
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
  <property>
    <name>dfs.datanode.use.datanode.hostname</name>
    <value>false</value>
  </property>
</configuration>
```

The following command updates the file automatically.

[ ]:
```
for host in REMAINING_IPS:
    !ssh hadoop@{host} 'printf "%s\n" {commands.get_datanode_hdfs_site()} > /
↪opt/hadoop/etc/hadoop/hdfs-site.xml'
```

c. Update the `yarn-site.xml` file located at `/opt/hadoop/etc/hadoop` to configure the YARN subsystem. The file must contain the following lines.

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>{namenode_hostname}</value>
  </property>
  <property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <value>1536</value>
  </property>
  <property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>1536</value>
  </property>
  <property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>128</value>
  </property>
  <property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
  </property>
</configuration>
```

The following command updates the file automatically.

```
[ ]: for host in REMAINING_IPS:
         !ssh hadoop@{host} 'printf "%s\n" {commands.
     ↪get_datanode_yarn_site(VMS[NAMENODE_IP])} > /opt/hadoop/etc/hadoop/yarn-site.
     ↪xml'
```

d. Update the `mapred-site.xml` file located at `/opt/hadoop/etc/hadoop` to configure the MAPREDUCE subsystem. The file must contain the following lines.

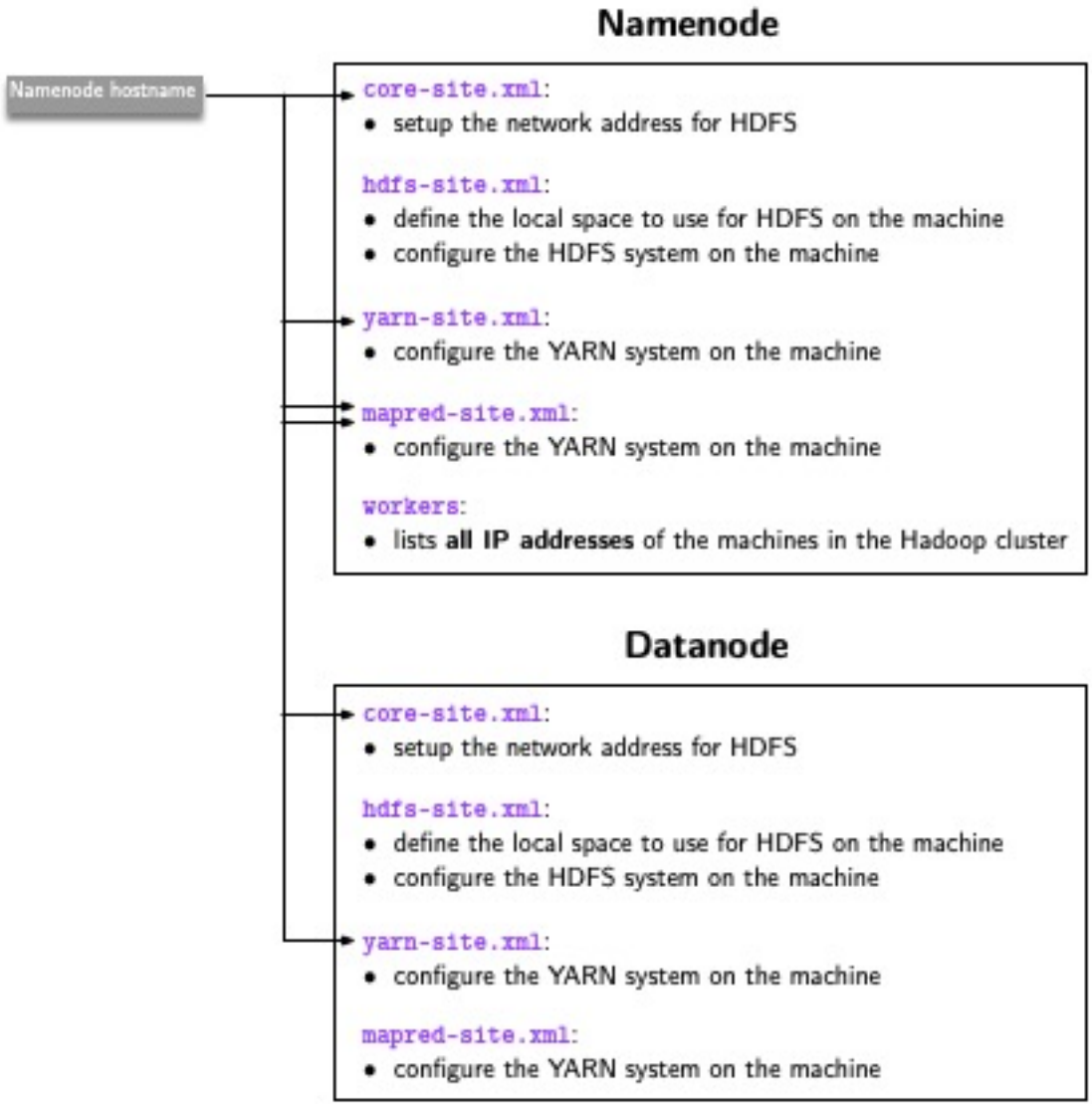
```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
  <property>
    <name>yarn.app.mapreduce.am.resource.mb</name>
    <value>512</value>
  </property>
  <property>
    <name>mapreduce.map.memory.mb</name>
    <value>256</value>
  </property>
  <property>
    <name>mapreduce.reduce.memory.mb</name>
    <value>256</value>
  </property>
</configuration>
```

The following command updates the file automatically.

```
[ ]: for host in REMAINING_IPS:
         !ssh hadoop@{host} 'printf "%s\n" {commands.get_datanode_mapred_site()} > /
     ↪opt/hadoop/etc/hadoop/mapred-site.xml'
```

If your machines have more than 2 GB of RAM or if you are interested in the numbers we specified in the YARN and MAPRED configuration files, check the Append A on the Hadoop Memory Allocaltion.

As a summary, please double check the content of the files list in the following picture, on the cluster machines:



1.5 4. Start, test and stop Hadoop

From now, all commands will be issued from the `hadoop-namenode` virtual machine.

a. Delete the contents of the local HDFS file system. Note: **This causes the loss of all information stored in HDFS**.

[ ]:
```
for host in VMS:
    run_ssh(host, 'rm -rf /opt/hdfs/namenode/*')
    run_ssh(host, 'rm -rf /opt/hdfs/datanode/*')

host = NAMENODE_IP
!ssh hadoop@{host} 'stop-dfs.sh'
```

b. Format the HDFS filesystem at the namenode.

```
[ ]: host = NAMENODE_IP
     !ssh hadoop@{host} 'hdfs namenode -format -force'
```

c. Creating the HDFS home folder for the `hadoop` user.

```
[ ]: !ssh hadoop@{host} 'start-dfs.sh'
     !ssh hadoop@{host} 'hadoop fs -mkdir -p /user/hadoop'
     !ssh hadoop@{host} 'stop-dfs.sh'
```

d. Starting HDFS and YARN.

```
[ ]: host = NAMENODE_IP
     !ssh hadoop@{host} 'start-dfs.sh'
     !ssh hadoop@{host} 'start-yarn.sh'
```

e. Checking all daemons up and running. You should receive 5 lines from the namenode machine and 3 lines from the datanote machines.

```
[ ]: for host in VMS:
         run_ssh(host, 'jps')
```

You may check logs at `/opt/hadoop/logs` on the 3 machines and check if everything is alright, or running the `hdfs dfsadmin -report` command (it must return `Live datanodes (3)`).

You can access Hadoop on a browser on your local machine (use IP addresses, not hostnames): - HDFS subsystem: `http://172.16.3.81:9870/` - YARN subsystem: `http://172.16.3.81:8088/`

f. Run an example provided by Hadoop. * Wait a minute before running, the daemons should perform some initialization steps * Ignore initial errors `No such file or directory` * Ignore logger message by logger `sasl.SaslDataTransferClient`

```
[ ]: host = NAMENODE_IP
     !ssh hadoop@{host} 'hadoop fs -rm -r input output'
     !ssh hadoop@{host} 'hadoop fs -put /opt/hadoop/etc/hadoop/ input'
     !ssh hadoop@{host} "hadoop jar /opt/hadoop/share/hadoop/mapreduce/
      ↪hadoop-mapreduce-examples-3.1.3.jar grep /user/hadoop/input/*.xml /user/
      ↪hadoop/output 'dfs[a-z.]+'"
     !ssh hadoop@{host} 'hadoop fs -cat output/part-r-00000'
```

g. Stop HDFS and YARN.

```
[ ]: host = NAMENODE_IP
     !ssh hadoop@{host} 'stop-dfs.sh'
     !ssh hadoop@{host} 'stop-yarn.sh'
```

1.6 Troubleshooting

If you get an error like the following:

`[2020-01-14 08:48:28.567]Container [pid=155967,containerID=container_1578991625193_0002_01_000`

you are using more virtual memory than your current limit of 2.1 Gb. This can be resolved in two ways:

1. **Disable Virtual Memory Limit Checking**YARN will simply ignore the limit; in order to do this, add this to your `yarn-site.xml` *on each machine*: `bash <property> <name>yarn.nodemanager.vmem-check-enabled</name> <value>false</value> </property>` The default for this setting is `true`.
2. **Increase Virtual Memory to Physical Memory Ratio**In your `yarn-site.xml` change this to a higher value than is currently set, *on each machine*: `bash <property> <name>yarn.nodemanager.vmem-pmem-ratio</name> <value>5</value> </property>` The default is 2.1. You could also increase the amount of physical memory you allocate to a container. *Make sure you don't forget to restart yarn after you change the configuration.*

2 Appendix A. Understanding the Hadoop Memory Allocation

Memory allocation can be complex on low RAM nodes because default values are not suitable for nodes with less than 8 GB of RAM. In this section we highlight how memory allocation works for MapReduce jobs, and provide a sample configuration for 2 GB RAM nodes.

A YARN job is executed with two kind of resources:

- An **application master** (AM), which is responsible for monitoring the application and coordinating distributed executors in the cluster.
- Some **executors**, that are created by the AM to actually run the job. For a MapReduce job, they will perform map or reduce operation, in parallel.

Both are run in **containers** on **worker nodes**. Each worker node runs a **NodeManager** daemon that is responsible for container creation on the node. The whole cluster is managed by a **ResourceManager** that schedules container allocation on all the worker nodes, depending on capacity requirements and current charge.

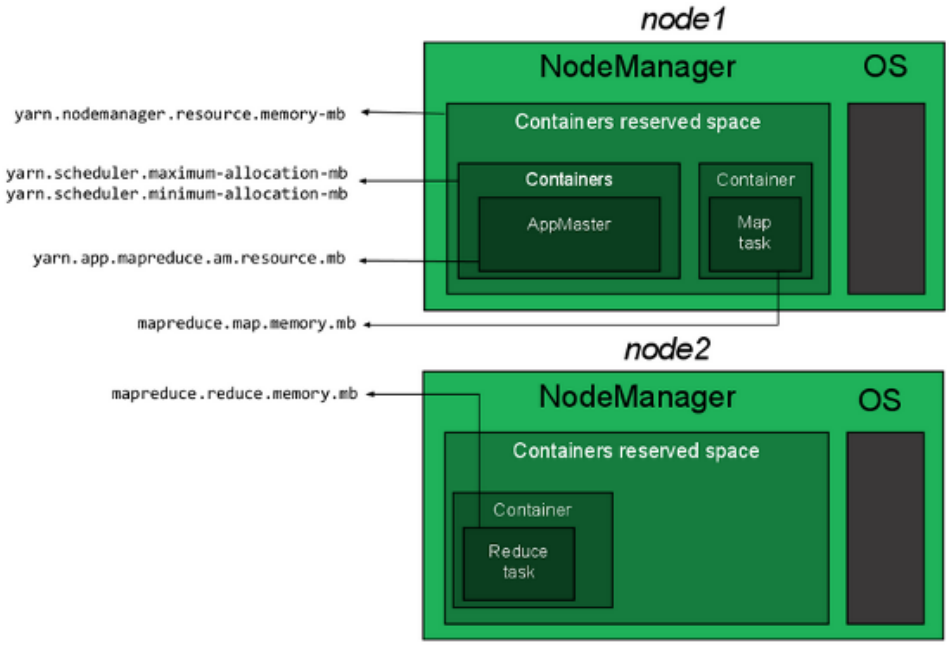
Four types of resource allocations need to be configured properly for the cluster to work. These are:

1. *How much memory can be allocated for YARN containers on a single node.* This limit should be higher than all the others; otherwise, container allocation will be rejected and applications will fail. However, it should not be the entire amount of RAM on the node. This value is configured in the `yarn-site.xml` file with the `yarn.nodemanager.resource.memory-mb` property.
2. *How much memory a single container can consume and the minimum memory allocation allowed.* A container will never be bigger than the maximum, or else allocation will fail and will always be allocated as a multiple of the minimum amount of RAM. Those values are configured in the `yarn-site.xml` file with the `yarn.scheduler.maximum-allocation-mb` and `yarn.scheduler.minimum-allocation-mb` properties.
3. *How much memory will be allocated to the ApplicationMaster.* This is a constant value that should fit in the container maximum size. This value is configured in the `mapred-site.xml`

with the `yarn.app.mapreduce.am.resource.mb` property.

4. *How much memory will be allocated to each map or reduce operation.* This should be less than the maximum size. This value is configured in the `mapred-site.xml` file with the `mapreduce.map.memory.mb` and `mapreduce.reduce.memory.mb` properties.

The relationship between all those properties can be seen in the following figure:



For 2 GB nodes, a working configuration may be:

Property	Value
`yarn.nodemanager.resource.memory-mb`	1536
`yarn.scheduler.maximum-allocation-mb`	1536
`yarn.scheduler.minimum-allocation-mb`	128
`yarn.app.mapreduce.am.resource.mb`	512
`mapreduce.map.memory.mb`	256
`mapreduce.reduce.memory.mb`	256

[ ]: