Big Data
- Hadoop Admin

Dr. Qing “Matt” Zhang

ITU

Checking HDFS Status

- *hdfs fsck* checks for missing or corrupt data blocks
  - Unlike Linux system fsck, it does not attempt to repair errors
- Can list all files, all blocks for each file, all block locations, or all racks
- Examples:
  - *hdfs fsck /*
  - *hdfs fsck / -files*
  - *hdfs fsck / -files -blocks*
  - *hdfs fsck / -files -blocks -locations*
  - *hdfs fsck / -files -blocks -locations -racks*

Checking HDFS Status

- Good idea to run *hdfs fsck* as a regular cron job that emails the results to administrators
- Choose a low-usage time to run the check
- *-move* option moves corrupted files to /lost+found
  - A corrupted file is one where all replicas of a block are missing
- *-delete* option deletes corrupted files

dfsadmin

- The *hdfs dfsadmin* command provides a number of administrative features including:
- List information about HDFS on a per-datanode basis:
  - *hdfs dfsadmin –report*
- Re-read the dfs.hosts and dfs.hosts.exclude files
  - These are defined in hdfs-site.xml, contains file of list of hosts which are (not) allowed to connect to namenode
  - *hdfs dfsadmin -refreshNodes*

Cluster Rebalancing

- An HDFS cluster can become ‘unbalanced’
  - Some nodes have much more data on them than others
  - Example: add a new node to the cluster
    - Even after adding some files to HDFS, this node will have far less data than the others
    - During MapReduce processing, this node will use much more network bandwidth as it retrieves data from other nodes
- Clusters can be rebalanced using the *hdfs balancer* utility

Using hdfs balancer

- *hdfs balancer* reviews data block placement on nodes and adjusts blocks to ensure all nodes are within x% utilization of each other
  - Utilization is defined as amount of data storage used
  - x is known as the threshold
- A node is under-utilized if its utilization is less than (average utilization - threshold)
- A node is over-utilized if its utilization is more than (average utilization + threshold)
- Note: hdfs balancer does not consider block placement on individual disks on a node
  - Only the utilization of the node as a whole

Using hdfs balancer

- Syntax:

  hdfs balancer -threshold x

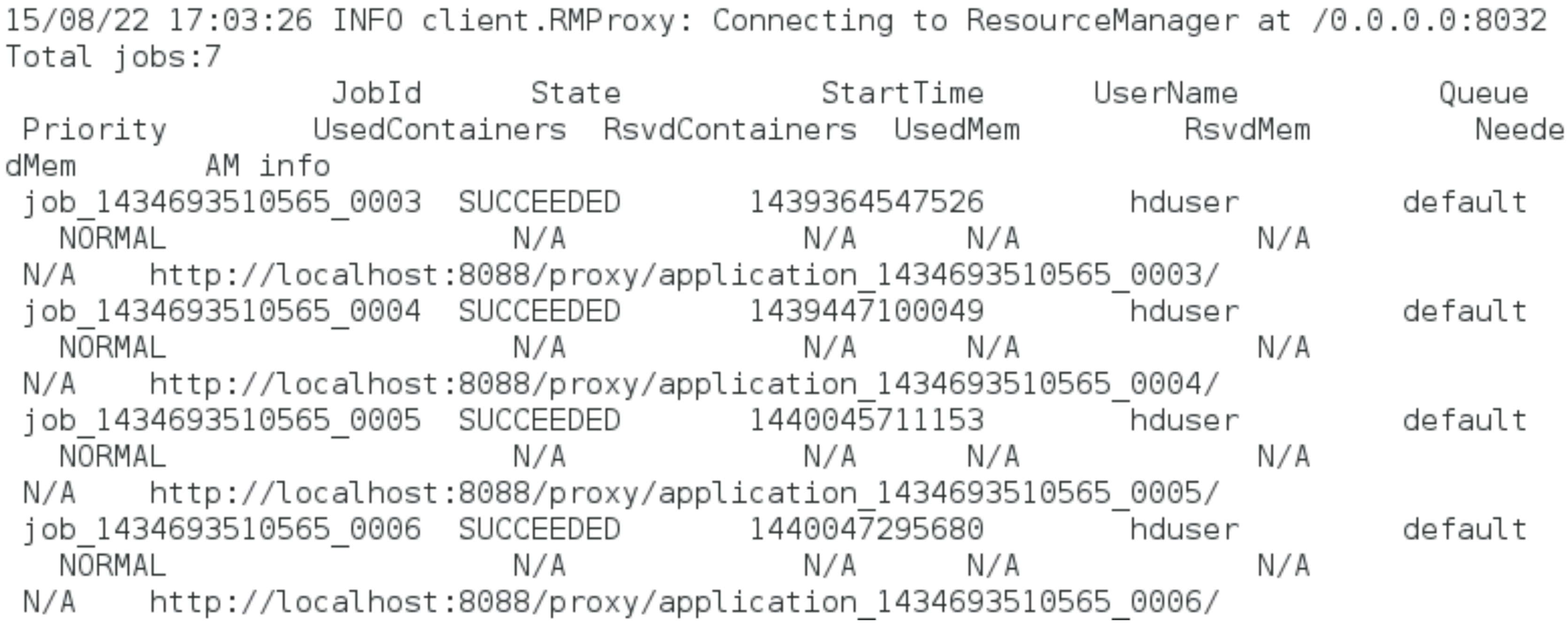
- Threshold is optional
  - Defaults to 10 (i.e., 10% difference in utilization between nodes)
- Rebalancing can be canceled at any time
  - Interrupt the command with Ctrl+C

When to Rebalance

- Rebalance immediately after adding new nodes to the cluster
- Rebalance during non-peak usage times
  - Rebalancing does not interfere with any existing MapReduce jobs
  - However, it does use bandwidth

Job Management

- To view all jobs running on the cluster
  - *mapred job -list*
- To view all jobs including completed ones
  - *mapred job -list all*



```
15/08/22 17:03:26 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
Total jobs:7
                  JobId      State             StartTime      UserName           Queue
 Priority        UsedContainers  RsvdContainers  UsedMem         RsvdMem          Neede
dMem         AM info
 job_1434693510565_0003   SUCCEEDED       1439364547526        hduser          default
   NORMAL                   N/A              N/A      N/A              N/A
 N/A     http://localhost:8088/proxy/application_1434693510565_0003/
 job_1434693510565_0004   SUCCEEDED       1439447100049        hduser          default
   NORMAL                   N/A              N/A      N/A              N/A
 N/A     http://localhost:8088/proxy/application_1434693510565_0004/
 job_1434693510565_0005   SUCCEEDED       1440045711153        hduser          default
   NORMAL                   N/A              N/A      N/A              N/A
 N/A     http://localhost:8088/proxy/application_1434693510565_0005/
 job_1434693510565_0006   SUCCEEDED       1440047295680        hduser          default
   NORMAL                   N/A              N/A      N/A              N/A
 N/A     http://localhost:8088/proxy/application_1434693510565_0006/
```

Display individual job status

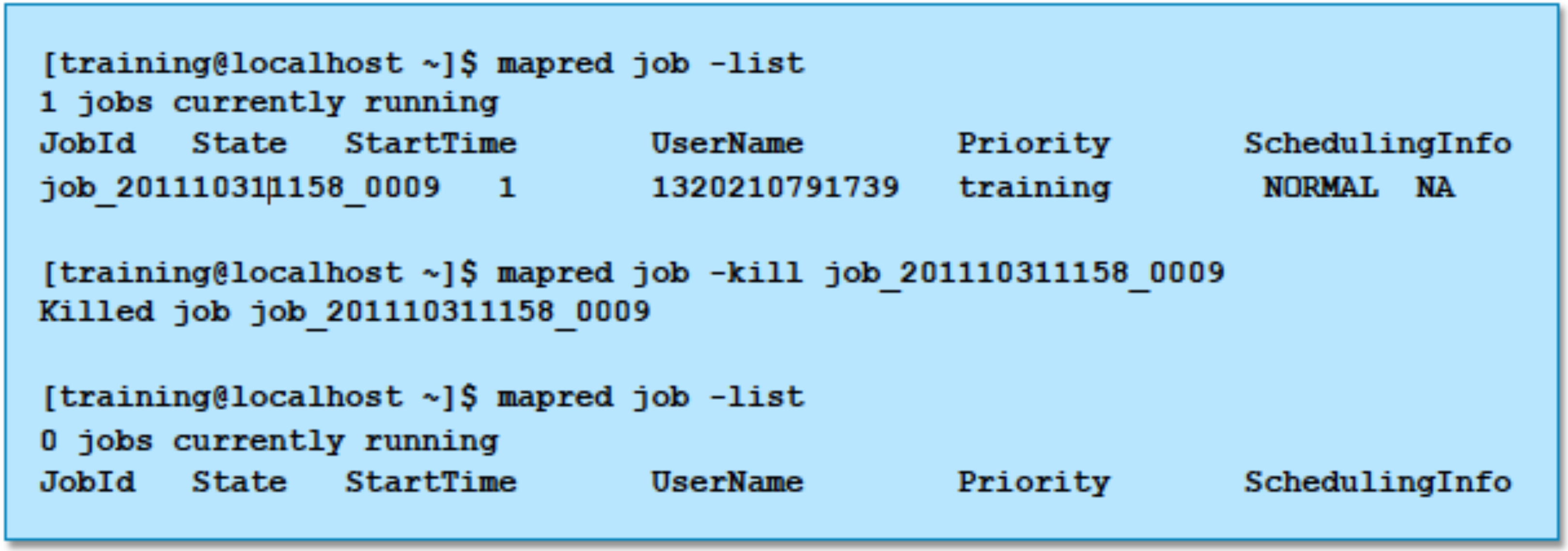
- To display individual job status:
    - *mapred job -status <job_id>*
    - It provides completion percentage, values of counters, etc
    - Job name is not displayed
    - The Web user interface is the most convenient way to view more details about an individual job

Kill a job

- It is important to note that once a user has submitted a job, they can not stop it just by hitting CTRL+C on their terminal
    - This stops job output appearing on the user’s console
    - The job is still running on the cluster!

Kill a job

- To kill a job use *mapred job -kill <job_id>*



```
[training@localhost ~]$ mapred job -list
1 jobs currently running
JobId    State    StartTime        UserName        Priority        SchedulingInfo
job_201110311158_0009    1        1320210791739   training         NORMAL   NA

[training@localhost ~]$ mapred job -kill job_201110311158_0009
Killed job job_201110311158_0009

[training@localhost ~]$ mapred job -list
0 jobs currently running
JobId    State    StartTime        UserName        Priority        SchedulingInfo
```

Web monitoring

- Namenode Web UI:

  <NameNode ADDR>:50070
- JobTracker Web UI (Hadoop 1.x):

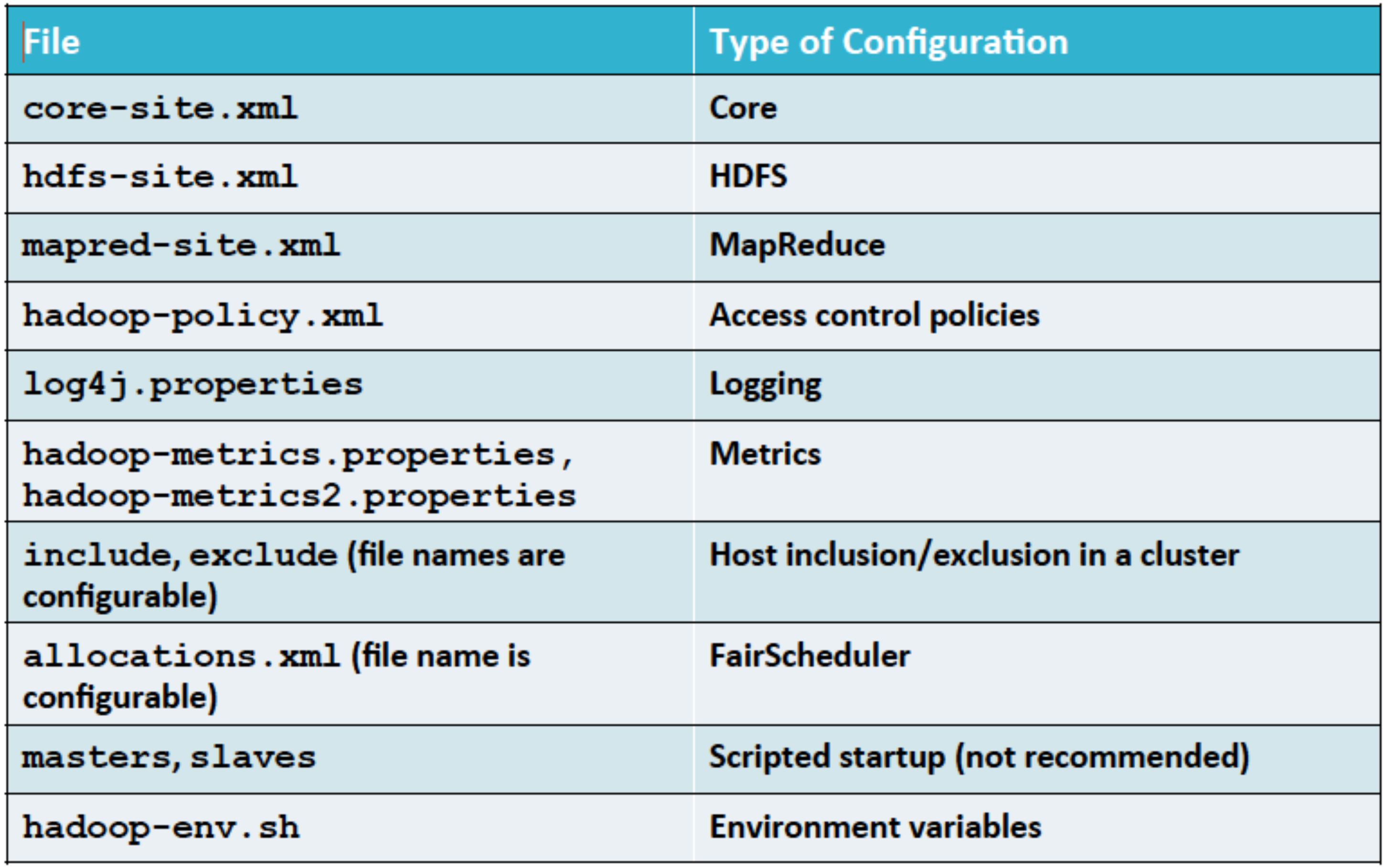
  <JobTracker ADDR>:50030
- ResourceManager Web UI (YARN):

  <ResourceManager ADDR>:8088

Hadoop Configuration Files

- Each machine in the Hadoop cluster has its own set of configuration files
- Configuration files all reside in Hadoop’s conf directory
  - Typically /etc/hadoop/conf
- Most of the configuration files are written in XML
- Upon startup, the Hadoop daemons access the configuration files
  - After modifying configuration parameters, you must restart Hadoop daemons for your changes to take effect

Hadoop Configuration Files Overview



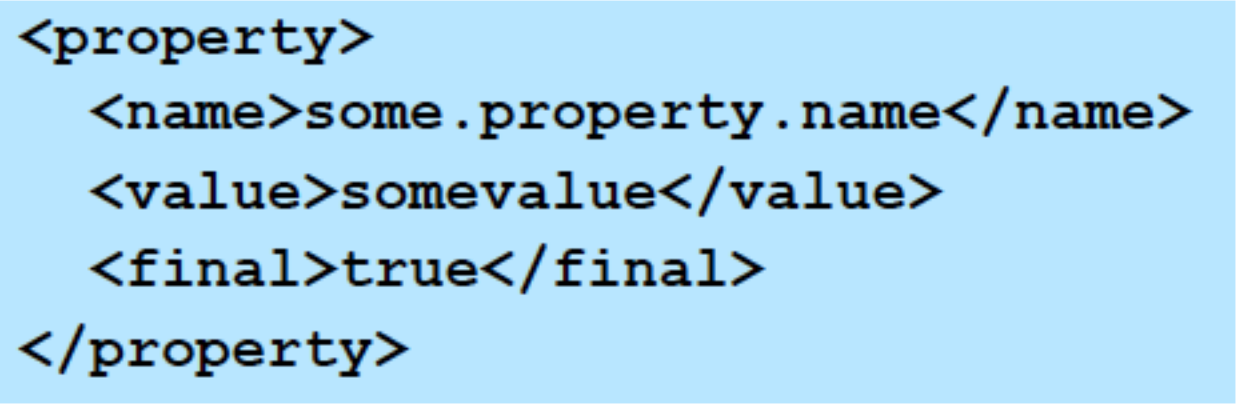
File	Type of Configuration
core-site.xml	Core
hdfs-site.xml	HDFS
mapred-site.xml	MapReduce
hadoop-policy.xml	Access control policies
log4j.properties	Logging
hadoop-metrics.properties, hadoop-metrics2.properties	Metrics
include, exclude (file names are configurable)	Host inclusion/exclusion in a cluster
allocations.xml (file name is configurable)	FairScheduler
masters, slaves	Scripted startup (not recommended)
hadoop-env.sh	Environment variables

Configuration Value Precedence

- Configuration parameters can be specified more than once
- Highest precedence value takes priority
- Precedence order (lowest to highest):
  - *-site.xml on the slave node
  - *-site.xml on the client machine
  - Values set explicitly in the Job object for a MapReduce job

Configuration Value Precedence

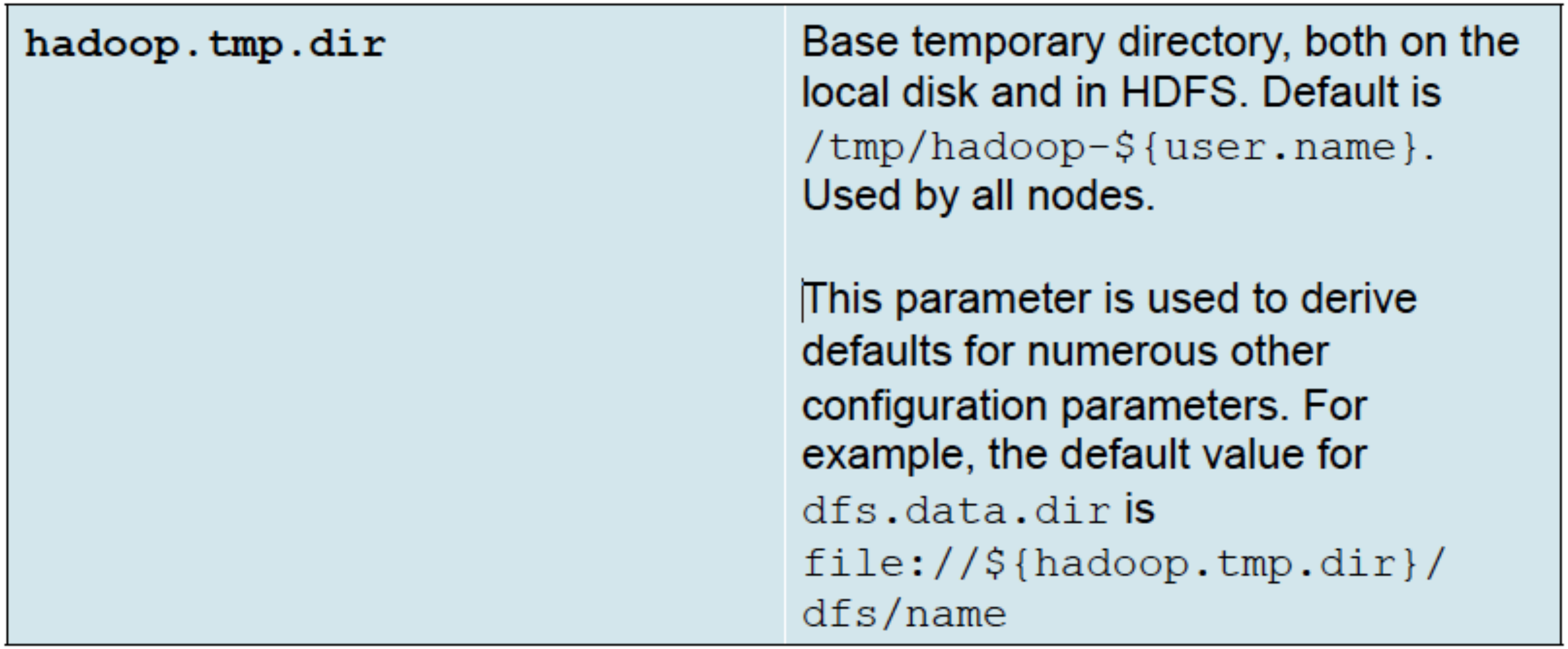
- If a value in a configuration file is marked as final it overrides all others



```
<property>
  <name>some.property.name</name>
  <value>somevalue</value>
  <final>true</final>
</property>
```

Important Configurations

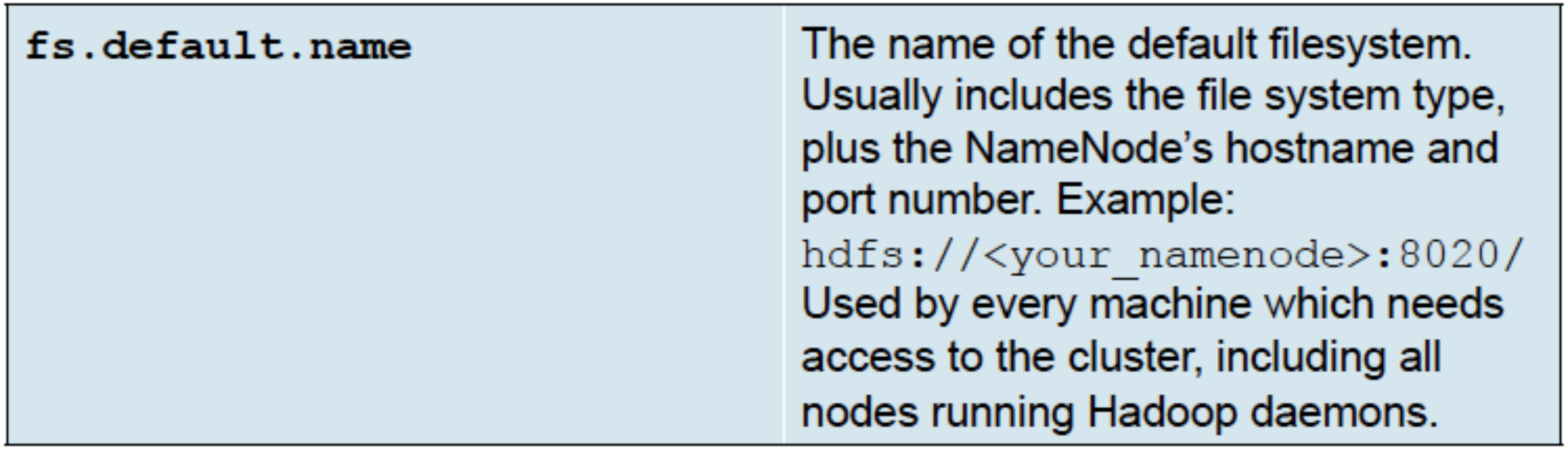
- core-site.xml:



`hadoop.tmp.dir`	Base temporary directory, both on the local disk and in HDFS. Default is `/tmp/hadoop-${user.name}`. Used by all nodes. This parameter is used to derive defaults for numerous other configuration parameters. For example, the default value for `dfs.data.dir` is `file://${hadoop.tmp.dir}/dfs/name`

- In our system, it's /tmp/hadoop-hduser/

core-site.xml



`fs.default.name`	The name of the default filesystem. Usually includes the file system type, plus the NameNode's hostname and port number. Example: `hdfs://<your_namenode>:8020/` Used by every machine which needs access to the cluster, including all nodes running Hadoop daemons.

- In YARN, it's replaced by fs.defaultFS

hdfs-site.xml

- The single most important configuration value on your entire cluster, used by the NameNode
- dfs.name.dir: A comma separated list of directories, describing where namenode stores the HDFS metadata (fsimage + Edit log)
  - default value = ${hadoop.tmp.dir}/dfs/name
- Loss of a NameNode’s metadata will result in the loss of all the data in its namespace
  - Although the blocks will remain, there is no way of reconstructing the original files without the metadata
- There must be at least two disks (or a RAID volume) on the NameNode, plus an NFS mount elsewhere on the network
  - Failure to set this correctly will result in eventual loss of your cluster’s data

hdfs-site.xml

- A NameNode will write to the edit log in all directories in *dfs.name.dir* synchronously
- If a directory in the list disappears, the NameNode will continue to function
  - It will ignore that directory until it is restarted
- Note: no space between the comma and next directory name in the list!
  - Example: */disk1/dfs/nn,/disk2/dfs/nn*

hdfs-site.xml

- dfs.block.size: The block size for new files, in bytes.
  - Default is 67108864 (64MB)
- dfs.data.dir: A comma separated list of directories, describing where a datanode stores its blocks
  - default value = ${hadoop.tmp.dir}/dfs/data
  - No space between the comma and the path
  - Round-robin writes to the directories in the list
  - Used by DataNodes
  - Can be different on each DataNode

hdfs-site.xml

- fs.checkpoint.dir: property for secondary namenode to store its checkpoints for the filesystem
  - default value = ${hadoop.tmp.dir}/dfs/namesecondary
- You can see fsimage, edits files and the md5 CRC code in *dfs.data.dir* and *fs.checkpoint.dir*

hdfs-site.xml

- dfs.http.address: The address and port used for the NameNode Web UI, used by NameNode
  - Default is <*your_namenode>:50070*
- dfs.replication: The number of times each block should be replicated when a file is written.
  - Default is *3*

Environment Setup: hadoop-env.sh

- hadoop-env.sh sets environment variables necessary for Hadoop to run
  - HADOOP_CLASSPATH
  - HADOOP_HEAPSIZE
  - HADOOP_LOG_DIR
  - HADOOP_PID_DIR
  - JAVA_HOME
- Values are sourced into all Hadoop control scripts and therefore the Hadoop daemons
- If you need to set environment variables, do it here to ensure that they are passed through to the control scripts