Hadoop Cluster Setup

Purpose

This document describes how to install and configure Hadoop clusters ranging from a few nodes to extremely large clusters with thousands of nodes. To play with Hadoop, you may first want to install it on a single machine (see Single Node Setup).

This document does not cover advanced topics such as High Availability.

Important: all production Hadoop clusters use Kerberos to authenticate callers and secure access to HDFS data as well as restriction access to computation services (YARN etc.).

These instructions do not cover integration with any Kerberos services, -everyone bringing up a production cluster should include connecting to their organisation's Kerberos infrastructure as a key part of the deployment.

See Security for details on how to secure a cluster.

Prerequisites

- Install Java. See the Hadoop Wiki for known good versions.
- Download a stable version of Hadoop from Apache mirrors.

Installation

Installing a Hadoop cluster typically involves unpacking the software on all the machines in the cluster or installing it via a packaging system as appropriate for your operating system. It is important to divide up the hardware into functions.

Typically one machine in the cluster is designated as the NameNode and another machine as the ResourceManager, exclusively. These are the masters. Other services (such as Web App Proxy Server and MapReduce Job History server) are usually run either on dedicated hardware or on shared infrastructure, depending upon the load.

The rest of the machines in the cluster act as both DataNode and NodeManager. These are the workers.

Configuring Hadoop in Non-Secure Mode

Hadoop's Java configuration is driven by two types of important configuration files:

- Read-only default configuration - `core-default.xml`, `hdfs-default.xml`, `yarn-default.xml` and `mapred-default.xml`.
- Site-specific configuration - `etc/hadoop/core-site.xml`, `etc/hadoop/hdfs-site.xml`, `etc/hadoop/yarn-site.xml` and `etc/hadoop/mapred-site.xml`.

Additionally, you can control the Hadoop scripts found in the bin/ directory of the distribution, by setting site-specific values via the `etc/hadoop/hadoop-env.sh` and `etc/hadoop/yarn-env.sh`.

To configure the Hadoop cluster you will need to configure the `environment` in which the Hadoop daemons execute as well as the `configuration parameters` for the Hadoop daemons.

HDFS daemons are NameNode, SecondaryNameNode, and DataNode. YARN daemons are ResourceManager, NodeManager, and WebAppProxy. If MapReduce is to be used, then the MapReduce Job History Server will also be running. For large installations, these are generally running on separate hosts.

Configuring Environment of Hadoop Daemons

Administrators should use the `etc/hadoop/hadoop-env.sh` and optionally the `etc/hadoop/mapred-env.sh` and `etc/hadoop/yarn-env.sh` scripts to do site-specific customization of the Hadoop daemons' process environment.

At the very least, you must specify the `JAVA_HOME` so that it is correctly defined on each remote node.

Administrators can configure individual daemons using the configuration options shown below in the table:

Daemon	Environment Variable
NameNode	HDFS_NAMENODE_OPTS
DataNode	HDFS_DATANODE_OPTS
Secondary NameNode	HDFS_SECONDARYNAMENODE_OPTS
ResourceManager	YARN_RESOURCEMANAGER_OPTS
NodeManager	YARN_NODEMANAGER_OPTS
WebAppProxy	YARN_PROXYSERVER_OPTS
Map Reduce Job History Server	MAPRED_HISTORYSERVER_OPTS

For example, To configure Namenode to use parallelGC and a 4GB Java Heap, the following statement should be added in hadoop-env.sh :

```
export HDFS_NAMENODE_OPTS="-XX:+UseParallelGC -Xmx4g"
```

See `etc/hadoop/hadoop-env.sh` for other examples.

Other useful configuration parameters that you can customize include:

- `HADOOP_PID_DIR` - The directory where the daemons' process id files are stored.
- `HADOOP_LOG_DIR` - The directory where the daemons' log files are stored. Log files are automatically created if they don't exist.
- `HADOOP_HEAPSIZE_MAX` - The maximum amount of memory to use for the Java heapsize. Units supported by the JVM are also supported here. If no unit is present, it will be assumed the number is in megabytes. By default, Hadoop will let the JVM determine how much to use. This value can be overriden on a per-daemon basis using the appropriate `_OPTS` variable listed above. For example, setting `HADOOP_HEAPSIZE_MAX=1g` and `HADOOP_NAMENODE_OPTS="-Xmx5g"` will configure the NameNode with 5GB heap.

In most cases, you should specify the `HADOOP_PID_DIR` and `HADOOP_LOG_DIR` directories such that they can only be written to by the users that are going to run the hadoop daemons. Otherwise there is the potential for a symlink attack.

It is also traditional to configure `HADOOP_HOME` in the system-wide shell environment configuration. For example, a simple script inside `/etc/profile.d`:

```
HADOOP_HOME=/path/to/hadoop
export HADOOP_HOME
```

Configuring the Hadoop Daemons

This section deals with important parameters to be specified in the given configuration files:

- `etc/hadoop/core-site.xml`

Parameter	Value	Notes
`fs.defaultFS`	NameNode URI	hdfs://host:port/
`io.file.buffer.size`	131072	Size of read/write buffer used in SequenceFiles.

- `etc/hadoop/hdfs-site.xml`
- Configurations for NameNode:

Parameter	Value	Notes
`dfs.namenode.name.dir`	Path on the local filesystem where the NameNode stores the namespace and transactions logs persistently.	If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy.
`dfs.hosts` / `dfs.hosts.exclude`	List of permitted/excluded DataNodes.	If necessary, use these files to control the list of allowable datanodes.
`dfs.blocksize`	268435456	HDFS blocksize of 256MB for large file-systems.
`dfs.namenode.handler.count`	100	More NameNode server threads to handle RPCs from large number of DataNodes.

- Configurations for DataNode:

Parameter	Value	Notes
`dfs.datanode.data.dir`	Comma separated list of paths on the local filesystem of a `DataNode` where it should store its blocks.	If this is a comma-delimited list of directories, then data will be stored in all named directories, typically on different devices.

- `etc/hadoop/yarn-site.xml`
- Configurations for ResourceManager and NodeManager:

Parameter	Value	Notes
`yarn.acl.enable`	`true` / `false`	Enable ACLs? Defaults to *false*.
`yarn.admin.acl`	Admin ACL	ACL to set admins on the cluster. ACLs are of for *comma-separated-usersspacecomma-separated-groups*. Defaults to special value of * which means *anyone*. Special value of just *space* means no one has access.
`yarn.log-aggregation-enable`	*false*	Configuration to enable or disable log aggregation

- Configurations for ResourceManager:

Parameter	Value	Notes
`yarn.resourcemanager.address`	`ResourceManager` host:port for clients to submit jobs.	*host:port* If set, overrides the hostname set in `yarn.resourcemanager.hostname`.
`yarn.resourcemanager.scheduler.address`	`ResourceManager` host:port for ApplicationMasters to talk to Scheduler to obtain resources.	*host:port* If set, overrides the hostname set in `yarn.resourcemanager.hostname`.
`yarn.resourcemanager.resource-tracker.address`	`ResourceManager` host:port for NodeManagers.	*host:port* If set, overrides the hostname set in `yarn.resourcemanager.hostname`.
`yarn.resourcemanager.admin.address`	`ResourceManager` host:port for administrative commands.	*host:port* If set, overrides the hostname set in `yarn.resourcemanager.hostname`.
`yarn.resourcemanager.webapp.address`	`ResourceManager` web-ui host:port.	*host:port* If set, overrides the hostname set in `yarn.resourcemanager.hostname`.
`yarn.resourcemanager.hostname`	`ResourceManager` host.	*host* Single hostname that can be set in place of setting all `yarn.resourcemanager*address` resources. Results in default ports for ResourceManager components.
`yarn.resourcemanager.scheduler.class`	`ResourceManager` Scheduler class.	`CapacityScheduler` (recommended), `FairScheduler` (also recommended), or `FifoScheduler`. Use a fully qualified class name, e.g., `org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler`.
`yarn.scheduler.minimum-allocation-mb`	Minimum limit of memory to allocate to each container request at the `Resource Manager`.	In MBs
`yarn.scheduler.maximum-allocation-mb`	Maximum limit of memory to allocate to each container request at the `Resource Manager`.	In MBs
`yarn.resourcemanager.nodes.include-path` / `yarn.resourcemanager.nodes.exclude-path`	List of permitted/excluded NodeManagers.	If necessary, use these files to control the list of allowable NodeManagers.

- Configurations for NodeManager:

Parameter	Value	Notes
`yarn.nodemanager.resource.memory-mb`	Resource i.e. available physical memory, in MB, for given `NodeManager`	Defines total available resources on the `NodeManager` to be made available to running containers
`yarn.nodemanager.vmem-pmem-ratio`	Maximum ratio by which virtual memory usage of tasks may exceed physical memory	The virtual memory usage of each task may exceed its physical memory limit by this ratio. The total amount of virtual memory used by tasks on the NodeManager may exceed its physical memory usage by this ratio.
`yarn.nodemanager.local-dirs`	Comma-separated list of paths on the local filesystem where intermediate data is written.	Multiple paths help spread disk i/o.
`yarn.nodemanager.log-dirs`	Comma-separated list of paths on the local filesystem where logs are written.	Multiple paths help spread disk i/o.
`yarn.nodemanager.log.retain-seconds`	*10800*	Default time (in seconds) to retain log files on the NodeManager Only applicable if log-aggregation is disabled.
`yarn.nodemanager.remote-app-log-dir`	*/logs*	HDFS directory where the application logs are moved on application completion. Need to set appropriate permissions. Only applicable if log-aggregation is enabled.
`yarn.nodemanager.remote-app-log-dir-suffix`	*logs*	Suffix appended to the remote log dir. Logs will be aggregated to ${yarn.nodemanager.remote-app-log-dir}/${user}/${thisParam} Only applicable if log-aggregation is enabled.
`yarn.nodemanager.aux-services`	mapreduce_shuffle	Shuffle service that needs to be set for Map Reduce applications.
`yarn.nodemanager.env-whitelist`	Environment properties to be inherited by containers from NodeManagers	For mapreduce application in addition to the default values HADOOP_MAPRED_HOME should to be added. Property value should JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_HOME,PATH,LANG,TZ,HADOOP_MAPRED_HOME

- Configurations for History Server (Needs to be moved elsewhere):

Parameter	Value	Notes
`yarn.log-aggregation.retain-seconds`	*-1*	How long to keep aggregation logs before deleting them. -1 disables. Be careful, set this too small and you will spam the name node.
`yarn.log-aggregation.retain-check-interval-seconds`	*-1*	Time between checks for aggregated log retention. If set to 0 or a negative value then the value is computed as one-tenth of the aggregated log retention time. Be careful, set this too small and you will spam the name node.

- `etc/hadoop/mapred-site.xml`
- Configurations for MapReduce Applications:

Parameter	Value	Notes
`mapreduce.framework.name`	yarn	Execution framework set to Hadoop YARN.
`mapreduce.map.memory.mb`	1536	Larger resource limit for maps.
`mapreduce.map.java.opts`	-Xmx1024M	Larger heap-size for child jvms of maps.
`mapreduce.reduce.memory.mb`	3072	Larger resource limit for reduces.
`mapreduce.reduce.java.opts`	-Xmx2560M	Larger heap-size for child jvms of reduces.
`mapreduce.task.io.sort.mb`	512	Higher memory-limit while sorting data for efficiency.
`mapreduce.task.io.sort.factor`	100	More streams merged at once while sorting files.
`mapreduce.reduce.shuffle.parallelcopies`	50	Higher number of parallel copies run by reduces to fetch outputs from very large number of maps.

- Configurations for MapReduce JobHistory Server:

Parameter	Value	Notes
`mapreduce.jobhistory.address`	MapReduce JobHistory Server *host:port*	Default port is 10020.
`mapreduce.jobhistory.webapp.address`	MapReduce JobHistory Server Web UI *host:port*	Default port is 19888.
`mapreduce.jobhistory.intermediate-done-dir`	/mr-history/tmp	Directory where history files are written by MapReduce jobs.
`mapreduce.jobhistory.done-dir`	/mr-history/done	Directory where history files are managed by the MR JobHistory Server.

Monitoring Health of NodeManagers

Hadoop provides a mechanism by which administrators can configure the NodeManager to run an administrator supplied script periodically to determine if a node is healthy or not.

Administrators can determine if the node is in a healthy state by performing any checks of their choice in the script. If the script detects the node to be in an unhealthy state, it must print a line to standard output beginning with the string ERROR. The NodeManager spawns the script periodically and checks its output. If the script's output contains the string ERROR, as described above, the node's status is reported as `unhealthy` and the node is black-listed by the ResourceManager. No further tasks will be assigned to this node. However, the NodeManager continues to run the script, so that if the node becomes healthy again, it will be removed from the blacklisted nodes on the ResourceManager automatically. The node's health along with the output of the script, if it is unhealthy, is available to the administrator in the ResourceManager web interface. The time since the node was healthy is also displayed on the web interface.

The following parameters can be used to control the node health monitoring script in `etc/hadoop/yarn-site.xml`.

Parameter	Value	Notes
`yarn.nodemanager.health-checker.script.path`	Node health script	Script to check for node's health status.
`yarn.nodemanager.health-checker.script.opts`	Node health script options	Options for script to check for node's health status.
`yarn.nodemanager.health-checker.interval-ms`	Node health script interval	Time interval for running health script.
`yarn.nodemanager.health-checker.script.timeout-ms`	Node health script timeout interval	Timeout for health script execution.

The health checker script is not supposed to give ERROR if only some of the local disks become bad. NodeManager has the ability to periodically check the health of the local disks (specifically checks nodemanager-local-dirs and nodemanager-log-dirs) and after reaching the threshold of number of bad directories based on the value set for the config property yarn.nodemanager.disk-health-checker.min-healthy-disks, the whole node is marked unhealthy and this info is sent to resource manager also. The boot disk is either raided or a failure in the boot disk is identified by the health checker script.

Slaves File

List all worker hostnames or IP addresses in your `etc/hadoop/workers` file, one per line. Helper scripts (described below) will use the `etc/hadoop/workers` file to run commands on many hosts at once. It is not used for any of the Java-based Hadoop configuration. In order to use this functionality, ssh trusts (via either passphraseless ssh or some other means, such as Kerberos) must be established for the accounts used to run Hadoop.

Hadoop Rack Awareness

Many Hadoop components are rack-aware and take advantage of the network topology for performance and safety. Hadoop daemons obtain the rack information of the workers in the cluster by invoking an administrator configured module. See the Rack Awareness documentation for more specific information.

It is highly recommended configuring rack awareness prior to starting HDFS.

Logging

Hadoop uses the Apache log4j via the Apache Commons Logging framework for logging. Edit the `etc/hadoop/log4j.properties` file to customize the Hadoop daemons' logging configuration (log-formats and so on).

Operating the Hadoop Cluster

Once all the necessary configuration is complete, distribute the files to the `HADOOP_CONF_DIR` directory on all the machines. This should be the same directory on all machines.

In general, it is recommended that HDFS and YARN run as separate users. In the majority of installations, HDFS processes execute as 'hdfs'. YARN is typically using the 'yarn' account.

Hadoop Startup

To start a Hadoop cluster you will need to start both the HDFS and YARN cluster.

The first time you bring up HDFS, it must be formatted. Format a new distributed filesystem as *hdfs*:

```
[hdfs]$ $HADOOP_HOME/bin/hdfs namenode -format
```

Start the HDFS NameNode with the following command on the designated node as *hdfs*:

```
[hdfs]$ $HADOOP_HOME/bin/hdfs --daemon start namenode
```

Start a HDFS DataNode with the following command on each designated node as *hdfs*:

```
[hdfs]$ $HADOOP_HOME/bin/hdfs --daemon start datanode
```

If `etc/hadoop/workers` and ssh trusted access is configured (see Single Node Setup), all of the HDFS processes can be started with a utility script. As *hdfs*:

```
[hdfs]$ $HADOOP_HOME/sbin/start-dfs.sh
```

Start the YARN with the following command, run on the designated ResourceManager as *yarn*:

```
[yarn]$ $HADOOP_HOME/bin/yarn --daemon start resourcemanager
```

Run a script to start a NodeManager on each designated host as *yarn*:

```
[yarn]$ $HADOOP_HOME/bin/yarn --daemon start nodemanager
```

Start a standalone WebAppProxy server. Run on the WebAppProxy server as *yarn*. If multiple servers are used with load balancing it should be run on each of them:

```
[yarn]$ $HADOOP_HOME/bin/yarn --daemon start proxyserver
```

If `etc/hadoop/workers` and ssh trusted access is configured (see Single Node Setup), all of the YARN processes can be started with a utility script. As *yarn*:

```
[yarn]$ $HADOOP_HOME/sbin/start-yarn.sh
```

Start the MapReduce JobHistory Server with the following command, run on the designated server as *mapred*:

```
[mapred]$ $HADOOP_HOME/bin/mapred --daemon start historyserver
```

Hadoop Shutdown

Stop the NameNode with the following command, run on the designated NameNode as *hdfs*:

```
[hdfs]$ $HADOOP_HOME/bin/hdfs --daemon stop namenode
```

Run a script to stop a DataNode as *hdfs*:

```
[hdfs]$ $HADOOP_HOME/bin/hdfs --daemon stop datanode
```

If `etc/hadoop/workers` and ssh trusted access is configured (see Single Node Setup), all of the HDFS processes may be stopped with a utility script. As *hdfs*:

```
[hdfs]$ $HADOOP_HOME/sbin/stop-dfs.sh
```

Stop the ResourceManager with the following command, run on the designated ResourceManager as *yarn*:

```
[yarn]$ $HADOOP_HOME/bin/yarn --daemon stop resourcemanager
```

Run a script to stop a NodeManager on a worker as *yarn*:

```
[yarn]$ $HADOOP_HOME/bin/yarn --daemon stop nodemanager
```

If `etc/hadoop/workers` and ssh trusted access is configured (see Single Node Setup), all of the YARN processes can be stopped with a utility script. As *yarn*:

```
[yarn]$ $HADOOP_HOME/sbin/stop-yarn.sh
```

Stop the WebAppProxy server. Run on the WebAppProxy server as *yarn*. If multiple servers are used with load balancing it should be run on each of them:

```
[yarn]$ $HADOOP_HOME/bin/yarn stop proxyserver
```

Stop the MapReduce JobHistory Server with the following command, run on the designated server as *mapred*:

```
[mapred]$ $HADOOP_HOME/bin/mapred --daemon stop historyserver
```

Web Interfaces

Once the Hadoop cluster is up and running check the web-ui of the components as described below:

Daemon	Web Interface	Notes
NameNode	http://nn_host:port/	Default HTTP port is 9870.
ResourceManager	http://rm_host:port/	Default HTTP port is 8088.
MapReduce JobHistory Server	http://jhs_host:port/	Default HTTP port is 19888.