Cluster Setup

Table of contents

1. Purpose

This document describes how to install, configure and manage non-trivial 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 Hadoop on a single machine (see Single Node Setup).

2. Prerequisites

1. Make sure all required software is installed on all nodes in your cluster.
2. Download the Hadoop software.

3. Installation

Installing a Hadoop cluster typically involves unpacking the software on all the machines in the cluster.

Typically one machine in the cluster is designated as the `NameNode` and another machine the as `JobTracker`, exclusively. These are the *masters*. The rest of the machines in the cluster act as both `DataNode` *and* `TaskTracker`. These are the *slaves*.

The root of the distribution is referred to as `HADOOP_HOME`. All machines in the cluster usually have the same `HADOOP_HOME` path.

4. Configuration

The following sections describe how to configure a Hadoop cluster.

4.1. Configuration Files

Hadoop configuration is driven by two types of important configuration files:

1. Read-only default configuration - src/core/core-default.xml, src/hdfs/hdfs-default.xml and src/mapred/mapred-default.xml.
2. Site-specific configuration - *conf/core-site.xml*, *conf/hdfs-site.xml* and *conf/mapred-site.xml*.

To learn more about how the Hadoop framework is controlled by these configuration files, look here.

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

4.2. Site Configuration

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.

The Hadoop daemons are `NameNode`/`DataNode` and `JobTracker`/`TaskTracker`.

4.2.1. Configuring the Environment of the Hadoop Daemons

Administrators should use the `conf/hadoop-env.sh` script to do site-specific customization of the Hadoop daemons' process environment.

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

Administrators can configure individual daemons using the configuration options `HADOOP_*_OPTS`. Various options available are shown below in the table.

Daemon	Configure Options
NameNode	HADOOP_NAMENODE_OPTS
DataNode	HADOOP_DATANODE_OPTS
SecondaryNamenode	HADOOP_SECONDARYNAMENODE_OPTS
JobTracker	HADOOP_JOBTRACKER_OPTS
TaskTracker	HADOOP_TASKTRACKER_OPTS

For example, To configure Namenode to use parallelGC, the following statement should be added in `hadoop-env.sh`:
```
export HADOOP_NAMENODE_OPTS="-XX:+UseParallelGC
${HADOOP_NAMENODE_OPTS}"
```

Other useful configuration parameters that you can customize include:
* `HADOOP_LOG_DIR` - The directory where the daemons' log files are stored. They are automatically created if they don't exist.
* `HADOOP_HEAPSIZE` - The maximum amount of heapsize to use, in MB e.g. `1000MB`. This is used to configure the heap size for the hadoop daemon. By default, the value is `1000MB`.

4.2.2. Configuring the Hadoop Daemons

This section deals with important parameters to be specified in the following:
`conf/core-site.xml`:

Parameter	Value	Notes
fs.default.name	URI of `NameNode`.	*hdfs://hostname/*

`conf/hdfs-site.xml`:

Parameter	Value	Notes
dfs.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.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.

`conf/mapred-site.xml`:

Parameter	Value	Notes
mapred.job.tracker	Host or IP and port of `JobTracker`.	*host:port* pair.
mapred.system.dir	Path on the HDFS where where the MapReduce framework stores system files e.g. `/hadoop/mapred/system/`.	This is in the default filesystem (HDFS) and must be accessible from both the server and client machines.
mapred.local.dir	Comma-separated list of paths on the local filesystem where temporary MapReduce data is written.	Multiple paths help spread disk i/o.
mapred.tasktracker.{map|reduce}	The maximum number of MapReduce tasks, which are run simultaneously on a given `TaskTracker`, individually.	Defaults to 2 (2 maps and 2 reduces), but vary it depending on your hardware.
dfs.hosts/dfs.hosts.exclude	List of permitted/excluded DataNodes.	If necessary, use these files to control the list of allowable datanodes.
mapred.hosts/mapred.hosts.excl	List of permitted/excluded TaskTrackers.	If necessary, use these files to control the list of allowable

		TaskTrackers.
mapred.queue.names	Comma separated list of queues to which jobs can be submitted.	The MapReduce system always supports atleast one queue with the name as *default*. Hence, this parameter's value should always contain the string *default*. Some job schedulers supported in Hadoop, like the Capacity Scheduler, support multiple queues. If such a scheduler is being used, the list of configured queue names must be specified here. Once queues are defined, users can submit jobs to a queue using the property name *mapred.job.queue.name* in the job configuration. There could be a separate configuration file for configuring properties of these queues that is managed by the scheduler. Refer to the documentation of the scheduler for information on the same.
mapred.acls.enabled	Boolean, specifying whether checks for queue ACLs and job ACLs are to be done for authorizing users for doing queue operations and job operations.	If *true*, queue ACLs are checked while submitting and administering jobs and job ACLs are checked for authorizing view and modification of jobs. Queue ACLs are specified using the configuration parameters of the form *mapred.queue.queue-name.acl-name*, defined below under mapred-queue-acls.xml. Job ACLs are described at Job Authorization

`conf/mapred-queue-acls.xml`

Parameter	Value	Notes
mapred.queue.*queue-name*.acl-s	List of users and groups that can submit jobs to the specified	The list of users and groups are both comma separated list of

	queue-name.	names. The two lists are separated by a blank. Example: *user1,user2 group1,group2*. If you wish to define only a list of groups, provide a blank at the beginning of the value.
mapred.queue.*queue-name*.acl-a	List of users and groups that can view job details, change the priority or kill jobs that have been submitted to the specified *queue-name*.	The list of users and groups are both comma separated list of names. The two lists are separated by a blank. Example: *user1,user2 group1,group2*. If you wish to define only a list of groups, provide a blank at the beginning of the value. Note that the owner of a job can always change the priority or kill his/her own job, irrespective of the ACLs.

Typically all the above parameters are marked as <u>final</u> to ensure that they cannot be overriden by user-applications.

4.2.2.1. Real-World Cluster Configurations

This section lists some non-default configuration parameters which have been used to run the *sort* benchmark on very large clusters.

- Some non-default configuration values used to run sort900, that is 9TB of data sorted on a cluster with 900 nodes:

Configuration File	Parameter	Value	Notes
conf/hdfs-site.xml	dfs.block.size	134217728	HDFS blocksize of 128MB for large file-systems.
conf/hdfs-site.xml	dfs.namenode.handl	40	More NameNode server threads to handle RPCs from large number of DataNodes.
conf/mapred-site.xm	mapred.reduce.para	20	Higher number of parallel copies run by reduces to fetch outputs from very large number of

			maps.
conf/mapred-site.xm	mapred.map.child.ja	-Xmx512M	Larger heap-size for child jvms of maps.
conf/mapred-site.xm	mapred.reduce.child	-Xmx512M	Larger heap-size for child jvms of reduces.
conf/core-site.xml	fs.inmemory.size.mb	200	Larger amount of memory allocated for the in-memory file-system used to merge map-outputs at the reduces.
conf/core-site.xml	io.sort.factor	100	More streams merged at once while sorting files.
conf/core-site.xml	io.sort.mb	200	Higher memory-limit while sorting data.
conf/core-site.xml	io.file.buffer.size	131072	Size of read/write buffer used in SequenceFiles.

• Updates to some configuration values to run sort1400 and sort2000, that is 14TB of data sorted on 1400 nodes and 20TB of data sorted on 2000 nodes:

Configuration File	Parameter	Value	Notes
conf/mapred-site.xm	mapred.job.tracker.h	60	More JobTracker server threads to handle RPCs from large number of TaskTrackers.
conf/mapred-site.xm	mapred.reduce.para	50	
conf/mapred-site.xm	tasktracker.http.thre	50	More worker threads for the TaskTracker's http server. The http server is used by reduces to fetch intermediate

			map-outputs.
conf/mapred-site.xm	mapred.map.child.ja	-Xmx512M	Larger heap-size for child jvms of maps.
conf/mapred-site.xm	mapred.reduce.child	-Xmx1024M	Larger heap-size for child jvms of reduces.

4.2.2.2. Task Controllers

Task controllers are classes in the Hadoop MapReduce framework that define how user's map and reduce tasks are launched and controlled. They can be used in clusters that require some customization in the process of launching or controlling the user tasks. For example, in some clusters, there may be a requirement to run tasks as the user who submitted the job, instead of as the task tracker user, which is how tasks are launched by default. This section describes how to configure and use task controllers.

The following task controllers are the available in Hadoop.

Name	Class Name	Description
DefaultTaskController	org.apache.hadoop.mapred.Defa	The default task controller which Hadoop uses to manage task execution. The tasks run as the task tracker user.
LinuxTaskController	org.apache.hadoop.mapred.Linu	This task controller, which is supported only on Linux, runs the tasks as the user who submitted the job. It requires these user accounts to be created on the cluster nodes where the tasks are launched. It uses a setuid executable that is included in the Hadoop distribution. The task tracker uses this executable to launch and kill tasks. The setuid executable switches to the user who has submitted the job and launches or kills the tasks. For maximum security, this task controller sets up restricted permissions and user/group ownership of local files and

		directories used by the tasks such as the job jar files, intermediate files, task log files and distributed cache files. Particularly note that, because of this, except the job owner and tasktracker, no other user can access any of the local files/directories including those localized as part of the distributed cache.

Configuring Task Controllers

The task controller to be used can be configured by setting the value of the following key in mapred-site.xml

Property	Value	Notes
mapred.task.tracker.task-controll	Fully qualified class name of the task controller class	Currently there are two implementations of task controller in the Hadoop system, DefaultTaskController and LinuxTaskController. Refer to the class names mentioned above to determine the value to set for the class of choice.

Using the LinuxTaskController

This section of the document describes the steps required to use the LinuxTaskController.

In order to use the LinuxTaskController, a setuid executable should be built and deployed on the compute nodes. The executable is named task-controller. To build the executable, execute *ant task-controller -Dhadoop.conf.dir=/path/to/conf/dir.* The path passed in *-Dhadoop.conf.dir* should be the path on the cluster nodes where a configuration file for the setuid executable would be located. The executable would be built to *build.dir/dist.dir/bin* and should be installed to *$HADOOP_HOME/bin.*

The executable must have specific permissions as follows. The executable should have *6050 or --Sr-s---* permissions user-owned by root(super-user) and group-owned by a special group of which the TaskTracker's user is the group member and no job submitter is. If any job submitter belongs to this special group, security will be compromised. This special group name should be specified for the configuration property *"mapreduce.tasktracker.group"* in both mapred-site.xml and task-controller.cfg. For example, let's say that the TaskTracker is

run as user *mapred* who is part of the groups *users* and *specialGroup* any of them being the primary group. Let also be that *users* has both *mapred* and another user (job submitter) *X* as its members, and X does not belong to *specialGroup*. Going by the above description, the setuid/setgid executable should be set *6050 or --Sr-s---* with user-owner as *mapred* and group-owner as *specialGroup* which has *mapred* as its member(and not *users* which has *X* also as its member besides *mapred*).

The LinuxTaskController requires that paths including and leading up to the directories specified in *mapred.local.dir* and *hadoop.log.dir* to be set 755 permissions.

task-controller.cfg

The executable requires a configuration file called *taskcontroller.cfg* to be present in the configuration directory passed to the ant target mentioned above. If the binary was not built with a specific conf directory, the path defaults to */path-to-binary/../conf*. The configuration file must be owned by the user running TaskTracker (user *mapred* in the above example), group-owned by anyone and should have the permissions *0400 or r--------*.

The executable requires following configuration items to be present in the *taskcontroller.cfg* file. The items should be mentioned as simple *key=value* pairs.

Name	Description
mapred.local.dir	Path to mapred local directories. Should be same as the value which was provided to key in mapred-site.xml. This is required to validate paths passed to the setuid executable in order to prevent arbitrary paths being passed to it.
hadoop.log.dir	Path to hadoop log directory. Should be same as the value which the TaskTracker is started with. This is required to set proper permissions on the log files so that they can be written to by the user's tasks and read by the TaskTracker for serving on the web UI.
mapreduce.tasktracker.group	Group to which the TaskTracker belongs. The group owner of the taskcontroller binary should be this group. Should be same as the value with which the TaskTracker is configured. This configuration is required for validating the secure access of the task-controller binary.

4.2.2.3. Monitoring Health of TaskTracker Nodes

Hadoop MapReduce provides a mechanism by which administrators can configure the

TaskTracker 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 TaskTracker 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 on the JobTracker. No further tasks will be assigned to this node. However, the TaskTracker continues to run the script, so that if the node becomes healthy again, it will be removed from the blacklisted nodes on the JobTracker automatically. The node's health along with the output of the script, if it is unhealthy, is available to the administrator in the JobTracker's web interface. The time since the node was healthy is also displayed on the web interface.

Configuring the Node Health Check Script

The following parameters can be used to control the node health monitoring script in *mapred-site.xml*.

Name	Description
`mapred.healthChecker.script.path`	Absolute path to the script which is periodically run by the TaskTracker to determine if the node is healthy or not. The file should be executable by the TaskTracker. If the value of this key is empty or the file does not exist or is not executable, node health monitoring is not started.
`mapred.healthChecker.interval`	Frequency at which the node health script is run, in milliseconds
`mapred.healthChecker.script.timeout`	Time after which the node health script will be killed by the TaskTracker if unresponsive. The node is marked unhealthy. if node health script times out.
`mapred.healthChecker.script.args`	Extra arguments that can be passed to the node health script when launched. These should be comma separated list of arguments.

4.2.3. Memory monitoring

A `TaskTracker`(TT) can be configured to monitor memory usage of tasks it spawns, so that badly-behaved jobs do not bring down a machine due to excess memory consumption. With monitoring enabled, every task is assigned a task-limit for virtual memory (VMEM). In

addition, every node is assigned a node-limit for VMEM usage. A TT ensures that a task is killed if it, and its descendants, use VMEM over the task's per-task limit. It also ensures that one or more tasks are killed if the sum total of VMEM usage by all tasks, and their descendents, cross the node-limit.

Users can, optionally, specify the VMEM task-limit per job. If no such limit is provided, a default limit is used. A node-limit can be set per node.

Currently the memory monitoring and management is only supported in Linux platform.

To enable monitoring for a TT, the following parameters all need to be set:

Name	Type	Description
mapred.tasktracker.vmem.reserv	long	A number, in bytes, that represents an offset. The total VMEM on the machine, minus this offset, is the VMEM node-limit for all tasks, and their descendants, spawned by the TT.
mapred.task.default.maxvmem	long	A number, in bytes, that represents the default VMEM task-limit associated with a task. Unless overridden by a job's setting, this number defines the VMEM task-limit.
mapred.task.limit.maxvmem	long	A number, in bytes, that represents the upper VMEM task-limit associated with a task. Users, when specifying a VMEM task-limit for their tasks, should not specify a limit which exceeds this amount.

In addition, the following parameters can also be configured.

Name	Type	Description
mapred.tasktracker.taskmemoryr	long	The time interval, in milliseconds, between which the TT checks for any memory violation. The default value is 5000 msec (5 seconds).

Here's how the memory monitoring works for a TT.

1. If one or more of the configuration parameters described above are missing or -1 is specified , memory monitoring is disabled for the TT.
2. In addition, monitoring is disabled if `mapred.task.default.maxvmem` is greater than `mapred.task.limit.maxvmem`.
3. If a TT receives a task whose task-limit is set by the user to a value larger than `mapred.task.limit.maxvmem`, it logs a warning but executes the task.
4. Periodically, the TT checks the following:
   - If any task's current VMEM usage is greater than that task's VMEM task-limit, the task is killed and reason for killing the task is logged in task diagonistics . Such a task is considered failed, i.e., the killing counts towards the task's failure count.
   - If the sum total of VMEM used by all tasks and descendants is greater than the node-limit, the TT kills enough tasks, in the order of least progress made, till the overall VMEM usage falls below the node-limt. Such killed tasks are not considered failed and their killing does not count towards the tasks' failure counts.

Schedulers can choose to ease the monitoring pressure on the TT by preventing too many tasks from running on a node and by scheduling tasks only if the TT has enough VMEM free. In addition, Schedulers may choose to consider the physical memory (RAM) available on the node as well. To enable Scheduler support, TTs report their memory settings to the JobTracker in every heartbeat. Before getting into details, consider the following additional memory-related parameters than can be configured to enable better scheduling:

Name	Type	Description
mapred.tasktracker.pmem.reserv	int	A number, in bytes, that represents an offset. The total physical memory (RAM) on the machine, minus this offset, is the recommended RAM node-limit. The RAM node-limit is a hint to a Scheduler to scheduler only so many tasks such that the sum total of their RAM requirements does not exceed this limit. RAM usage is not monitored by a TT.

A TT reports the following memory-related numbers in every heartbeat:

- The total VMEM available on the node.
- The value of `mapred.tasktracker.vmem.reserved`, if set.
- The total RAM available on the node.
- The value of `mapred.tasktracker.pmem.reserved`, if set.

4.2.4. Slaves

Typically you choose one machine in the cluster to act as the `NameNode` and one machine as to act as the `JobTracker`, exclusively. The rest of the machines act as both a `DataNode` and `TaskTracker` and are referred to as *slaves*.

List all slave hostnames or IP addresses in your `conf/slaves` file, one per line.

4.2.5. Logging

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

4.2.5.1. History Logging

The job history files are stored in central location `hadoop.job.history.location` which can be on DFS also, whose default value is `${HADOOP_LOG_DIR}/history`. The history web UI is accessible from job tracker web UI.

The history files are also logged to user specified directory `hadoop.job.history.user.location` which defaults to job output directory. The files are stored in "_logs/history/" in the specified directory. Hence, by default they will be in "mapred.output.dir/_logs/history/". User can stop logging by giving the value `none` for `hadoop.job.history.user.location`

User can view the history logs summary in specified directory using the following command
`$ bin/hadoop job -history output-dir`
This command will print job details, failed and killed tip details.
More details about the job such as successful tasks and task attempts made for each task can be viewed using the following command
`$ bin/hadoop job -history all output-dir`

Once all the necessary configuration is complete, distribute the files to the `HADOOP_CONF_DIR` directory on all the machines, typically `${HADOOP_HOME}/conf`.

5. Cluster Restartability

5.1. MapReduce

The job tracker restart can recover running jobs if `mapred.jobtracker.restart.recover` is set true and [JobHistory logging](#) is

enabled. Also `mapred.jobtracker.job.history.block.size` value should be set to an optimal value to dump job history to disk as soon as possible, the typical value is 3145728(3MB).

6. Hadoop Rack Awareness

The HDFS and the Map/Reduce components are rack-aware.

The `NameNode` and the `JobTracker` obtains the `rack id` of the slaves in the cluster by invoking an API resolve in an administrator configured module. The API resolves the slave's DNS name (also IP address) to a rack id. What module to use can be configured using the configuration item `topology.node.switch.mapping.impl`. The default implementation of the same runs a script/command configured using `topology.script.file.name`. If topology.script.file.name is not set, the rack id `/default-rack` is returned for any passed IP address. The additional configuration in the Map/Reduce part is `mapred.cache.task.levels` which determines the number of levels (in the network topology) of caches. So, for example, if it is the default value of 2, two levels of caches will be constructed - one for hosts (host -> task mapping) and another for racks (rack -> task mapping).

7. Hadoop Startup

To start a Hadoop cluster you will need to start both the HDFS and Map/Reduce cluster.

Format a new distributed filesystem:
`$ bin/hadoop namenode -format`

Start the HDFS with the following command, run on the designated `NameNode`:
`$ bin/start-dfs.sh`

The `bin/start-dfs.sh` script also consults the `${HADOOP_CONF_DIR}/slaves` file on the `NameNode` and starts the `DataNode` daemon on all the listed slaves.

Start Map-Reduce with the following command, run on the designated `JobTracker`:
`$ bin/start-mapred.sh`

The `bin/start-mapred.sh` script also consults the `${HADOOP_CONF_DIR}/slaves` file on the `JobTracker` and starts the `TaskTracker` daemon on all the listed slaves.

8. Hadoop Shutdown

Stop HDFS with the following command, run on the designated `NameNode`:

```
$ bin/stop-dfs.sh
```

The `bin/stop-dfs.sh` script also consults the `${HADOOP_CONF_DIR}/slaves` file on the `NameNode` and stops the `DataNode` daemon on all the listed slaves.

Stop Map/Reduce with the following command, run on the designated the designated `JobTracker`:
```
$ bin/stop-mapred.sh
```

The `bin/stop-mapred.sh` script also consults the `${HADOOP_CONF_DIR}/slaves` file on the `JobTracker` and stops the `TaskTracker` daemon on all the listed slaves.