Cluster Setup

Table of contents

Copyright © 2009 The Apache Software Foundation. All rights reserved.

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 Hadoop Quick Start).

2. Pre-requisites

1. Make sure all requisite software is installed on all nodes in your cluster.
2. Get 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, src/mapred/mapred-default.xml and conf/mapred-queues.xml.template.
2. Site-specific configuration - conf/core-site.xml, conf/hdfs-site.xml, conf/mapred-site.xml and conf/mapred-queues.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`.

Copyright © 2009 The Apache Software Foundation. All rights reserved.

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

Copyright © 2009 The Apache Software Foundation. All rights reserved.

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
mapreduce.jobtracker.address	Host or IP and port of `JobTracker`.	*host:port* pair.
mapreduce.jobtracker.system.dir	Path on the HDFS where where the Map/Reduce 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.
mapreduce.cluster.local.dir	Comma-separated list of paths on the local filesystem where temporary Map/Reduce data is written.	Multiple paths help spread disk i/o.
mapred.tasktracker.{map|reduce}	The maximum number of Map/Reduce 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.
mapreduce.jobtracker.hosts.filen	List of permitted/excluded	If necessary, use these files to

Copyright © 2009 The Apache Software Foundation. All rights reserved.

	TaskTrackers.	control the list of allowable TaskTrackers.
mapreduce.cluster.job-authorizat	Boolean, specifying whether job ACLs are supported for authorizing view and modification of a job	If *true*, job ACLs would be checked while viewing or modifying a job. More details are available at Job Authorization.

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

`conf/mapred-queues.xml :`

This file is used to configure the queues in the Map/Reduce system. Queues are abstract entities in the JobTracker that can be used to manage collections of jobs. They provide a way for administrators to organize jobs in specific ways and to enforce certain policies on such collections, thus providing varying levels of administrative control and management functions on jobs.

One can imagine the following sample scenarios:

- Jobs submitted by a particular group of users can all be submitted to one queue.
- Long running jobs in an organization can be submitted to a queue.
- Short running jobs can be submitted to a queue and the number of jobs that can run concurrently can be restricted.

The usage of queues is closely tied to the scheduler configured at the JobTracker via *mapreduce.jobtracker.taskscheduler*. The degree of support of queues depends on the scheduler used. Some schedulers support a single queue, while others support more complex configurations. Schedulers also implement the policies that apply to jobs in a queue. Some schedulers, such as the Fairshare scheduler, implement their own mechanisms for collections of jobs and do not rely on queues provided by the framework. The administrators are encouraged to refer to the documentation of the scheduler they are interested in for determining the level of support for queues.

The Map/Reduce framework supports some basic operations on queues such as job submission to a specific queue, access control for queues, queue states, viewing configured queues and their properties and refresh of queue properties. In order to fully implement some of these operations, the framework takes the help of the configured scheduler.

The following types of queue configurations are possible:

- Single queue: The default configuration in Map/Reduce comprises of a single queue, as supported by the default scheduler. All jobs are submitted to this default queue which

Copyright © 2009 The Apache Software Foundation. All rights reserved.

maintains jobs in a priority based FIFO order.

- Multiple single level queues: Multiple queues are defined, and jobs can be submitted to any of these queues. Different policies can be applied to these queues by schedulers that support this configuration to provide a better level of support. For example, the capacity scheduler provides ways of configuring different capacity and fairness guarantees on these queues.
- Hierarchical queues: Hierarchical queues are a configuration in which queues can contain other queues within them recursively. The queues that contain other queues are referred to as container queues. Queues that do not contain other queues are referred as leaf or job queues. Jobs can only be submitted to leaf queues. Hierarchical queues can potentially offer a higher level of control to administrators, as schedulers can now build a hierarchy of policies where policies applicable to a container queue can provide context for policies applicable to queues it contains. It also opens up possibilities for delegating queue administration where administration of queues in a container queue can be turned over to a different set of administrators, within the context provided by the container queue. For example, the capacity scheduler uses hierarchical queues to partition capacity of a cluster among container queues, and allowing queues they contain to divide that capacity in more ways.

Most of the configuration of the queues can be refreshed/reloaded without restarting the Map/Reduce sub-system by editing this configuration file as described in the section on reloading queue configuration. Not all configuration properties can be reloaded of course, as will description of each property below explain.

The format of conf/mapred-queues.xml is different from the other configuration files, supporting nested configuration elements to support hierarchical queues. The format is as follows:

```
<queues aclsEnabled="$aclsEnabled">
  <queue>
    <name>$queue-name</name>
    <state>$state</state>
    <queue>
      <name>$child-queue1</name>
      <properties>
         <property key="$key" value="$value"/>
         ...
      </properties>
      <queue>
        <name>$grand-child-queue1</name>
        ...
      </queue>
    </queue>
    <queue>
      <name>$child-queue2</name>
```


Copyright © 2009 The Apache Software Foundation. All rights reserved.

```
            ...
          </queue>
          ...
          ...
          ...
          <queue>
            <name>$leaf-queue</name>
            <acl-submit-job>$acls</acl-submit-job>
            <acl-administer-jobs>$acls</acl-administer-jobs>
            <properties>
               <property key="$key" value="$value"/>
               ...
            </properties>
          </queue>
        </queue>
      </queues>
```

Tag/Attribute	Value	Refresh-able?	Notes
queues	Root element of the configuration file.	Not-applicable	All the queues are nested inside this root element of the file. There can be only one root queues element in the file.
aclsEnabled	Boolean attribute to the *<queues>* tag specifying whether ACLs are supported for controlling job submission and administration for *all* the queues configured.	Yes	If *false*, ACLs are ignored for *all* the configured queues. If *true*, the user and group details of the user are checked against the configured ACLs of the corresponding job-queue while submitting and administering jobs. ACLs can be specified for each queue using the queue-specific tags "acl-$acl_name", defined below. ACLs are checked only against the job-queues, i.e. the leaf-level queues; ACLs configured for the rest of the queues in the hierarchy are

Copyright © 2009 The Apache Software Foundation. All rights reserved.

			ignored.
queue	A child element of the *<queues>* tag or another *<queue>*. Denotes a queue in the system.	Not applicable	Queues can be hierarchical and so this element can contain children of this same type.
name	Child element of a *<queue>* specifying the name of the queue.	No	Name of the queue cannot contain the character *":"* which is reserved as the queue-name delimiter when addressing a queue in a hierarchy.
state	Child element of a *<queue>* specifying the state of the queue.	Yes	Each queue has a corresponding state. A queue in *'running'* state can accept new jobs, while a queue in *'stopped'* state will stop accepting any new jobs. State is defined and respected by the framework only for the leaf-level queues and is ignored for all other queues. The state of the queue can be viewed from the command line using `'bin/mapred queue'` command and also on the the Web UI. Administrators can stop and start queues at runtime using the feature of reloading queue configuration. If a queue is stopped at runtime, it will complete all the existing running jobs and will stop accepting any new jobs.

Copyright © 2009 The Apache Software Foundation. All rights reserved.

acl-submit-job	Child element of a *<queue>* specifying the list of users and groups that can submit jobs to the specified queue.	Yes	Applicable only to leaf-queues. 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.
acl-administer-job	Child element of a *<queue>* specifying the list of users and groups that can change the priority of a job or kill a job that has been submitted to the specified queue.	Yes	Applicable only to leaf-queues. 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 an owner of a job can always change the priority or kill his/her own job, irrespective of the ACLs.
properties	Child element of a *<queue>* specifying the scheduler specific properties.	Not applicable	The scheduler specific properties are the children of this element specified as a group of <property> tags described below. The JobTracker completely ignores these properties. These can be used as per-queue properties needed by the scheduler being configured. Please

Copyright © 2009 The Apache Software Foundation. All rights reserved.

			look at the scheduler specific documentation as to how these properties are used by that particular scheduler.
property	Child element of *<properties>* for a specific queue.	Not applicable	A single scheduler specific queue-property. Ignored by the JobTracker and used by the scheduler that is configured.
key	Attribute of a *<property>* for a specific queue.	Scheduler-specific	The name of a single scheduler specific queue-property.
value	Attribute of a *<property>* for a specific queue.	Scheduler-specific	The value of a single scheduler specific queue-property. The value can be anything that is left for the proper interpretation by the scheduler that is configured.

Once the queues are configured properly and the Map/Reduce system is up and running, from the command line one can get the list of queues and obtain information specific to each queue. This information is also available from the web UI. On the web UI, queue information can be seen by going to queueinfo.jsp, linked to from the queues table-cell in the cluster-summary table. The queueinfo.jsp prints the hierarchy of queues as well as the specific information for each queue.

Users can submit jobs only to a leaf-level queue by specifying the fully-qualified queue-name for the property name *mapreduce.job.queuename* in the job configuration. The character ':' is the queue-name delimiter and so, for e.g., if one wants to submit to a configured job-queue 'Queue-C' which is one of the sub-queues of 'Queue-B' which in-turn is a sub-queue of 'Queue-A', then the job configuration should contain property *mapreduce.job.queuename* set to the *<value>Queue-A:Queue-B:Queue-C</value>*

4.2.3. Real-World Cluster Configurations

This section lists some non-default configuration parameters which have been used to run the

Copyright © 2009 The Apache Software Foundation. All rights reserved.

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	mapreduce.reduce.s	20	Higher number of parallel copies run by reduces to fetch outputs from very large number of maps.
conf/mapred-site.xm	mapreduce.map.java	-Xmx512M	Larger heap-size for child jvms of maps.
conf/mapred-site.xm	mapreduce.reduce.j	-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	mapreduce.task.io.s	100	More streams merged at once while sorting files.
conf/core-site.xml	mapreduce.task.io.s	200	Higher memory-limit while sorting data.
conf/core-site.xml	io.file.buffer.size	131072	Size of read/write buffer used in

Copyright © 2009 The Apache Software Foundation. All rights reserved.

			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	mapreduce.jobtracke	60	More JobTracker server threads to handle RPCs from large number of TaskTrackers.
conf/mapred-site.xm	mapreduce.reduce.s	50	
conf/mapred-site.xm	mapreduce.tasktrack	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	mapreduce.map.java	-Xmx512M	Larger heap-size for child jvms of maps.
conf/mapred-site.xm	mapreduce.reduce.j	-Xmx1024M	Larger heap-size for child jvms of reduces.

4.2.4. Configuring Memory Parameters for MapReduce Jobs

As MapReduce jobs could use varying amounts of memory, Hadoop provides various configuration options to users and administrators for managing memory effectively. Some of these options are job specific and can be used by users. While setting up a cluster, administrators can configure appropriate default values for these options so that users jobs run out of the box. Other options are cluster specific and can be used by administrators to enforce limits and prevent misconfigured or memory intensive jobs from causing undesired side effects on the cluster.

The values configured should take into account the hardware resources of the cluster, such as the amount of physical and virtual memory available for tasks, the number of slots configured on the slaves and the requirements for other processes running on the slaves. If right values are not set, it is likely that jobs start failing with memory related errors or in the

Copyright © 2009 The Apache Software Foundation. All rights reserved.

worst case, even affect other tasks or the slaves themselves.

4.2.4.1. Monitoring Task Memory Usage

Before describing the memory options, it is useful to look at a feature provided by Hadoop to monitor memory usage of MapReduce tasks it runs. The basic objective of this feature is to prevent MapReduce tasks from consuming memory beyond a limit that would result in their affecting other processes running on the slave, including other tasks and daemons like the DataNode or TaskTracker.

Note: For the time being, this feature is available only for the Linux platform.

Hadoop allows monitoring to be done both for virtual and physical memory usage of tasks. This monitoring can be done independently of each other, and therefore the options can be configured independently of each other. It has been found in some environments, particularly related to streaming, that virtual memory recorded for tasks is high because of libraries loaded by the programs used to run the tasks. However, this memory is largely unused and does not affect the slaves's memory itself. In such cases, monitoring based on physical memory can provide a more accurate picture of memory usage.

This feature considers that there is a limit on the amount of virtual or physical memory on the slaves that can be used by the running MapReduce tasks. The rest of the memory is assumed to be required for the system and other processes. Since some jobs may require higher amount of memory for their tasks than others, Hadoop allows jobs to specify how much memory they expect to use at a maximum. Then by using resource aware scheduling and monitoring, Hadoop tries to ensure that at any time, only enough tasks are running on the slaves as can meet the dual constraints of an individual job's memory requirements and the total amount of memory available for all MapReduce tasks.

The TaskTracker monitors tasks in regular intervals. Each time, it operates in two steps:

- In the first step, it checks that a job's task and any child processes it launches are not cumulatively using more virtual or physical memory than specified. If both virtual and physical memory monitoring is enabled, then virtual memory usage is checked first, followed by physical memory usage. Any task that is found to use more memory is killed along with any child processes it might have launched, and the task status is marked *failed*. Repeated failures such as this will terminate the job.
- In the next step, it checks that the cumulative virtual and physical memory used by all running tasks and their child processes does not exceed the total virtual and physical memory limit, respectively. Again, virtual memory limit is checked first, followed by physical memory limit. In this case, it kills enough number of tasks, along with any child processes they might have launched, until the cumulative memory usage is brought under limit. In the case of virtual memory limit being exceeded, the tasks chosen for killing are

Copyright © 2009 The Apache Software Foundation. All rights reserved.

the ones that have made the least progress. In the case of physical memory limit being exceeded, the tasks chosen for killing are the ones that have used the maximum amount of physical memory. Also, the status of these tasks is marked as *killed*, and hence repeated occurrence of this will not result in a job failure.

In either case, the task's diagnostic message will indicate the reason why the task was terminated.

Resource aware scheduling can ensure that tasks are scheduled on a slave only if their memory requirement can be satisfied by the slave. The Capacity Scheduler, for example, takes virtual memory requirements into account while scheduling tasks, as described in the section on memory based scheduling.

Memory monitoring is enabled when certain configuration variables are defined with non-zero values, as described below.

4.2.4.2. Job Specific Options

Memory related options that can be configured individually per job are described in detail in the section on Configuring Memory Requirements For A Job in the MapReduce tutorial. While setting up the cluster, the Hadoop defaults for these options can be reviewed and changed to better suit the job profiles expected to be run on the clusters, as also the hardware configuration.

As with any other configuration option in Hadoop, if the administrators desire to prevent users from overriding these options in jobs they submit, these values can be marked as *final* in the cluster configuration.

4.2.4.3. Cluster Specific Options

This section describes the memory related options that are used by the JobTracker and TaskTrackers, and cannot be changed by jobs. The values set for these options should be the same for all the slave nodes in a cluster.

- `mapreduce.cluster.{map|reduce}memory.mb`: These options define the default amount of virtual memory that should be allocated for MapReduce tasks running in the cluster. They typically match the default values set for the options `mapreduce.{map|reduce}.memory.mb`. They help in the calculation of the total amount of virtual memory available for MapReduce tasks on a slave, using the following equation:
  *Total virtual memory for all MapReduce tasks = (mapreduce.cluster.mapmemory.mb * mapreduce.tasktracker.map.tasks.maximum) + (mapreduce.cluster.reducememory.mb * mapreduce.tasktracker.reduce.tasks.maximum)*

Copyright © 2009 The Apache Software Foundation. All rights reserved.

Typically, reduce tasks require more memory than map tasks. Hence a higher value is recommended for *mapreduce.cluster.reducememory.mb*. The value is specified in MB. To set a value of 2GB for reduce tasks, set *mapreduce.cluster.reducememory.mb* to 2048.

- `mapreduce.jobtracker.max{map|reduce}memory.mb`: These options define the maximum amount of virtual memory that can be requested by jobs using the parameters `mapreduce.{map|reduce}.memory.mb`. The system will reject any job that is submitted requesting for more memory than these limits. Typically, the values for these options should be set to satisfy the following constraint:
  *mapreduce.jobtracker.maxmapmemory.mb = mapreduce.cluster.mapmemory.mb * mapreduce.tasktracker.map.tasks.maximum*
  *mapreduce.jobtracker.maxreducememory.mb = mapreduce.cluster.reducememory.mb * mapreduce.tasktracker.reduce.tasks.maximum*
  The value is specified in MB. If `mapreduce.cluster.reducememory.mb` is set to 2GB and there are 2 reduce slots configured in the slaves, the value for `mapreduce.jobtracker.maxreducememory.mb` should be set to 4096.
- `mapreduce.tasktracker.reserved.physicalmemory.mb`: This option defines the amount of physical memory that is marked for system and daemon processes. Using this, the amount of physical memory available for MapReduce tasks is calculated using the following equation:
  Total physical memory for all MapReduce tasks = Total physical memory available on the system - mapreduce.tasktracker.reserved.physicalmemory.mb
  The value is specified in MB. To set this value to 2GB, specify the value as 2048.
- `mapreduce.tasktracker.taskmemorymanager.monitoringinterval`: This option defines the time the TaskTracker waits between two cycles of memory monitoring. The value is specified in milliseconds.

Note: The virtual memory monitoring function is only enabled if the variables `mapreduce.cluster.{map|reduce}memory.mb` and `mapreduce.jobtracker.max{map|reduce}memory.mb` are set to values greater than zero. Likewise, the physical memory monitoring function is only enabled if the variable `mapreduce.tasktracker.reserved.physicalmemory.mb` is set to a value greater than zero.

4.2.5. Task Controllers

Task controllers are classes in the Hadoop Map/Reduce 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.

Copyright © 2009 The Apache Software Foundation. All rights reserved.

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.

4.2.5.1. 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
mapreduce.tasktracker.taskcontr	Fully qualified class name of	Currently there are two

Copyright © 2009 The Apache Software Foundation. All rights reserved.

	the task controller class	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.

4.2.5.2. 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 *mapreduce.cluster.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--------*.

Copyright © 2009 The Apache Software Foundation. All rights reserved.

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
mapreduce.cluster.local.dir	Path to mapreduce.cluster.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.

4.2.6. Monitoring Health of TaskTracker Nodes

Hadoop Map/Reduce 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.

4.2.6.1. Configuring the Node Health Check Script

The following parameters can be used to control the node health monitoring script in

Copyright © 2009 The Apache Software Foundation. All rights reserved.

mapred-site.xml.

Name	Description
`mapreduce.tasktracker.healthchecker.s`	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.
`mapreduce.tasktracker.healthchecker.i`	Frequency at which the node health script is run, in milliseconds
`mapreduce.tasktracker.healthchecker.s`	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.
`mapreduce.tasktracker.healthchecker.s`	Extra arguments that can be passed to the node health script when launched. These should be comma separated list of arguments.

4.3. 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.

4.4. 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).

4.4.1. History Logging

The job history files are stored in central location `mapreduce.jobtracker.jobhistory.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

Copyright © 2009 The Apache Software Foundation. All rights reserved.

`mapreduce.job.userhistorylocation` which defaults to job output directory. The files are stored in "_logs/history/" in the specified directory. Hence, by default they will be in "mapreduce.output.fileoutputformat.outputdir/_logs/history/". User can stop logging by giving the value `none` for `mapreduce.job.userhistorylocation`

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. Map/Reduce

The job tracker restart can recover running jobs if `mapreduce.jobtracker.restart.recover` is set true and JobHistory logging is enabled. Also `mapreduce.jobtracker.jobhistory.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).

Copyright © 2009 The Apache Software Foundation. All rights reserved.

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.

Copyright © 2009 The Apache Software Foundation. All rights reserved.