HOD Scheduler

Table of contents

1. Introduction

Hadoop On Demand (HOD) is a system for provisioning and managing independent Hadoop MapReduce and Hadoop Distributed File System (HDFS) instances on a shared cluster of nodes. HOD is a tool that makes it easy for administrators and users to quickly setup and use Hadoop. HOD is also a very useful tool for Hadoop developers and testers who need to share a physical cluster for testing their own Hadoop versions.

HOD uses the Torque resource manager to do node allocation. On the allocated nodes, it can start Hadoop MapReduce and HDFS daemons. It automatically generates the appropriate configuration files (hadoop-site.xml) for the Hadoop daemons and client. HOD also has the capability to distribute Hadoop to the nodes in the virtual cluster that it allocates. HOD supports Hadoop from version 0.15 onwards.

2. HOD Users

This section shows users how to get started using HOD, reviews various HOD features and command line options, and provides detailed troubleshooting help.

2.1. Getting Started

In this section, we shall see a step-by-step introduction on how to use HOD for the most basic operations. Before following these steps, it is assumed that HOD and its dependent hardware and software components are setup and configured correctly. This is a step that is generally performed by system administrators of the cluster.

The HOD user interface is a command line utility called `hod`. It is driven by a configuration file, that is typically setup for users by system administrators. Users can override this configuration when using the `hod`, which is described later in this documentation. The configuration file can be specified in two ways when using `hod`, as described below:

- Specify it on command line, using the -c option. Such as `hod <operation> <required-args> -c path-to-the-configuration-file [other-options]`
- Set up an environment variable *HOD_CONF_DIR* where `hod` will be run. This should be pointed to a directory on the local file system, containing a file called *hodrc*. Note that this is analogous to the *HADOOP_CONF_DIR* and *hadoop-site.xml* file for Hadoop. If no configuration file is specified on the command line, `hod` shall look for the *HOD_CONF_DIR* environment variable and a *hodrc* file under that.

In examples listed below, we shall not explicitly point to the configuration option, assuming it is correctly specified.

2.1.1. A Typical HOD Session

A typical session of HOD will involve at least three steps: allocate, run hadoop jobs, deallocate. In order to do this, perform the following steps.

Create a Cluster Directory

The *cluster directory* is a directory on the local file system where `hod` will generate the Hadoop configuration, *hadoop-site.xml*, corresponding to the cluster it allocates. Pass this directory to the `hod` operations as stated below. If the cluster directory passed doesn't already exist, HOD will automatically try to create it and use it. Once a cluster is allocated, a user can utilize it to run Hadoop jobs by specifying the cluster directory as the Hadoop --config option.

Operation allocate

The *allocate* operation is used to allocate a set of nodes and install and provision Hadoop on them. It has the following syntax. Note that it requires a cluster_dir (-d, --hod.clusterdir) and the number of nodes (-n, --hod.nodecount) needed to be allocated:

```
$ hod allocate -d cluster_dir -n number_of_nodes [OPTIONS]
```

If the command completes successfully, then `cluster_dir/hadoop-site.xml` will be generated and will contain information about the allocated cluster. It will also print out the information about the Hadoop web UIs.

An example run of this command produces the following output. Note in this example that `~/hod-clusters/test` is the cluster directory, and we are allocating 5 nodes:

```
$ hod allocate -d ~/hod-clusters/test -n 5
INFO - HDFS UI on http://foo1.bar.com:53422
INFO - Mapred UI on http://foo2.bar.com:55380
```

Running Hadoop jobs using the allocated cluster

Now, one can run Hadoop jobs using the allocated cluster in the usual manner. This assumes variables like *JAVA_HOME* and path to the Hadoop installation are set up correctly.:

```
$ hadoop --config cluster_dir hadoop_command hadoop_command_args
```

or

```
$ export HADOOP_CONF_DIR=cluster_dir
$ hadoop hadoop_command hadoop_command_args
```

Continuing our example, the following command will run a wordcount example on the allocated cluster:

```
$ hadoop --config ~/hod-clusters/test jar
```

```
/path/to/hadoop/hadoop-examples.jar wordcount /path/to/input
/path/to/output
```

or

```
$ export HADOOP_CONF_DIR=~/hod-clusters/test
$ hadoop jar /path/to/hadoop/hadoop-examples.jar wordcount /path/to/input
/path/to/output
```

Operation deallocate

The *deallocate* operation is used to release an allocated cluster. When finished with a cluster, deallocate must be run so that the nodes become free for others to use. The *deallocate* operation has the following syntax. Note that it requires the cluster_dir (-d, --hod.clusterdir) argument:

```
$ hod deallocate -d cluster_dir
```

Continuing our example, the following command will deallocate the cluster:

```
$ hod deallocate -d ~/hod-clusters/test
```

As can be seen, HOD allows the users to allocate a cluster, and use it flexibly for running Hadoop jobs. For example, users can run multiple jobs in parallel on the same cluster, by running hadoop from multiple shells pointing to the same configuration.

2.1.2. Running Hadoop Scripts Using HOD

The HOD *script operation* combines the operations of allocating, using and deallocating a cluster into a single operation. This is very useful for users who want to run a script of hadoop jobs and let HOD handle the cleanup automatically once the script completes. In order to run hadoop scripts using `hod`, do the following:

Create a script file

This will be a regular shell script that will typically contain hadoop commands, such as:

```
$ hadoop jar jar_file options
```

However, the user can add any valid commands as part of the script. HOD will execute this script setting *HADOOP_CONF_DIR* automatically to point to the allocated cluster. So users do not need to worry about this. The users however need to specify a cluster directory just like when using the allocate operation.

Running the script

The syntax for the *script operation* as is as follows. Note that it requires a cluster directory (-d, --hod.clusterdir), number of nodes (-n, --hod.nodecount) and a script file (-s, --hod.script):

```
$ hod script -d cluster_directory -n number_of_nodes -s script_file
```

Note that HOD will deallocate the cluster as soon as the script completes, and this means that

the script must not complete until the hadoop jobs themselves are completed. Users must take care of this while writing the script.

2.2. HOD Features

2.2.1. Provisioning and Managing Hadoop Clusters

The primary feature of HOD is to provision Hadoop MapReduce and HDFS clusters. This is described above in the Getting Started section. Also, as long as nodes are available, and organizational policies allow, a user can use HOD to allocate multiple MapReduce clusters simultaneously. The user would need to specify different paths for the `cluster_dir` parameter mentioned above for each cluster he/she allocates. HOD provides the *list* and the *info* operations to enable managing multiple clusters.

Operation list

The list operation lists all the clusters allocated so far by a user. The cluster directory where the hadoop-site.xml is stored for the cluster, and its status vis-a-vis connectivity with the JobTracker and/or HDFS is shown. The list operation has the following syntax:

```
$ hod list
```

Operation info

The info operation shows information about a given cluster. The information shown includes the Torque job id, and locations of the important daemons like the HOD Ringmaster process, and the Hadoop JobTracker and NameNode daemons. The info operation has the following syntax. Note that it requires a cluster directory (-d, --hod.clusterdir):

```
$ hod info -d cluster_dir
```

The `cluster_dir` should be a valid cluster directory specified in an earlier *allocate* operation.

2.2.2. Using a Tarball to Distribute Hadoop

When provisioning Hadoop, HOD can use either a pre-installed Hadoop on the cluster nodes or distribute and install a Hadoop tarball as part of the provisioning operation. If the tarball option is being used, there is no need to have a pre-installed Hadoop on the cluster nodes, nor a need to use a pre-installed one. This is especially useful in a development / QE environment where individual developers may have different versions of Hadoop to test on a shared cluster.

In order to use a pre-installed Hadoop, you must specify, in the hodrc, the `pkgs` option in the `gridservice-hdfs` and `gridservice-mapred` sections. This must point to the path

where Hadoop is installed on all nodes of the cluster.

The syntax for specifying tarball is as follows:

```
$ hod allocate -d cluster_dir -n number_of_nodes -t hadoop_tarball_location
```

For example, the following command allocates Hadoop provided by the tarball `~/share/hadoop.tar.gz`:

```
$ hod allocate -d ~/hadoop-cluster -n 10 -t ~/share/hadoop.tar.gz
```

Similarly, when using hod script, the syntax is as follows:

```
$ hod script -d cluster_directory -s script_file -n number_of_nodes -t
hadoop_tarball_location
```

The hadoop_tarball specified in the syntax above should point to a path on a shared file system that is accessible from all the compute nodes. Currently, HOD only supports NFS mounted file systems.

Note:

- For better distribution performance it is recommended that the Hadoop tarball contain only the libraries and binaries, and not the source or documentation.
- When you want to run jobs against a cluster allocated using the tarball, you must use a compatible version of hadoop to submit your jobs. The best would be to untar and use the version that is present in the tarball itself.
- You need to make sure that there are no Hadoop configuration files, hadoop-env.sh and hadoop-site.xml, present in the conf directory of the tarred distribution. The presence of these files with incorrect values could make the cluster allocation to fail.

2.2.3. Using an External HDFS

In typical Hadoop clusters provisioned by HOD, HDFS is already set up statically (without using HOD). This allows data to persist in HDFS after the HOD provisioned clusters is deallocated. To use a statically configured HDFS, your hodrc must point to an external HDFS. Specifically, set the following options to the correct values in the section `gridservice-hdfs` of the hodrc:

```
external = true
host = Hostname of the HDFS NameNode
fs_port = Port number of the HDFS NameNode
info_port = Port number of the HDFS NameNode web UI
```

Note: You can also enable this option from command line. That is, to use a static HDFS, you will need to say:

```
$ hod allocate -d cluster_dir -n number_of_nodes
--gridservice-hdfs.external
```

HOD can be used to provision an HDFS cluster as well as a MapReduce cluster, if required.

To do so, set the following option in the section `gridservice-hdfs` of the hodrc:

```
external = false
```

2.2.4. Options for Configuring Hadoop

HOD provides a very convenient mechanism to configure both the Hadoop daemons that it provisions and also the hadoop-site.xml that it generates on the client side. This is done by specifying Hadoop configuration parameters in either the HOD configuration file, or from the command line when allocating clusters.

Configuring Hadoop Daemons

For configuring the Hadoop daemons, you can do the following:

For MapReduce, specify the options as a comma separated list of key-value pairs to the `server-params` option in the `gridservice-mapred` section. Likewise for a dynamically provisioned HDFS cluster, specify the options in the `server-params` option in the `gridservice-hdfs` section. If these parameters should be marked as *final*, then include these in the `final-server-params` option of the appropriate section.

For example:

```
server-params =
mapred.reduce.parallel.copies=20,io.sort.factor=100,io.sort.mb=128,io.file.buffer.size=
final-server-params =
mapred.child.java.opts=-Xmx512m,dfs.block.size=134217728,fs.inmemory.size.mb=128
```

In order to provide the options from command line, you can use the following syntax:

For configuring the MapReduce daemons use:

```
$ hod allocate -d cluster_dir -n number_of_nodes
-Mmapred.reduce.parallel.copies=20 -Mio.sort.factor=100
```

In the example above, the *mapred.reduce.parallel.copies* parameter and the *io.sort.factor* parameter will be appended to the other `server-params` or if they already exist in `server-params`, will override them. In order to specify these are *final* parameters, you can use:

```
$ hod allocate -d cluster_dir -n number_of_nodes
-Fmapred.reduce.parallel.copies=20 -Fio.sort.factor=100
```

However, note that final parameters cannot be overwritten from command line. They can only be appended if not already specified.

Similar options exist for configuring dynamically provisioned HDFS daemons. For doing so, replace -M with -H and -F with -S.

Configuring Hadoop Job Submission (Client) Programs

As mentioned above, if the allocation operation completes successfully then `cluster_dir/hadoop-site.xml` will be generated and will contain information about the allocated cluster's JobTracker and NameNode. This configuration is used when submitting jobs to the cluster. HOD provides an option to include additional Hadoop configuration parameters into this file. The syntax for doing so is as follows:

```
$ hod allocate -d cluster_dir -n number_of_nodes
-Cmapred.userlog.limit.kb=200 -Cmapred.child.java.opts=-Xmx512m
```

In this example, the *mapred.userlog.limit.kb* and *mapred.child.java.opts* options will be included into the hadoop-site.xml that is generated by HOD.

2.2.5. Viewing Hadoop Web-UIs

The HOD allocation operation prints the JobTracker and NameNode web UI URLs. For example:

```
$ hod allocate -d ~/hadoop-cluster -n 10 -c ~/hod-conf-dir/hodrc
INFO - HDFS UI on http://host242.foo.com:55391
INFO - Mapred UI on http://host521.foo.com:54874
```

The same information is also available via the *info* operation described above.

2.2.6. Collecting and Viewing Hadoop Logs

To get the Hadoop logs of the daemons running on one of the allocated nodes:

- Log into the node of interest. If you want to look at the logs of the JobTracker or NameNode, then you can find the node running these by using the *list* and *info* operations mentioned above.
- Get the process information of the daemon of interest (for example, `ps ux | grep TaskTracker`)
- In the process information, search for the value of the variable `-Dhadoop.log.dir`. Typically this will be a decendent directory of the `hodring.temp-dir` value from the hod configuration file.
- Change to the `hadoop.log.dir` directory to view daemon and user logs.

HOD also provides a mechanism to collect logs when a cluster is being deallocated and persist them into a file system, or an externally configured HDFS. By doing so, these logs can be viewed after the jobs are completed and the nodes are released. In order to do so, configure the log-destination-uri to a URI as follows:

```
log-destination-uri = hdfs://host123:45678/user/hod/logs
log-destination-uri = file://path/to/store/log/files
```

Under the root directory specified above in the path, HOD will create a path

user_name/torque_jobid and store gzipped log files for each node that was part of the job.

Note that to store the files to HDFS, you may need to configure the `hodring.pkgs` option with the Hadoop version that matches the HDFS mentioned. If not, HOD will try to use the Hadoop version that it is using to provision the Hadoop cluster itself.

2.2.7. Auto-deallocation of Idle Clusters

HOD automatically deallocates clusters that are not running Hadoop jobs for a given period of time. Each HOD allocation includes a monitoring facility that constantly checks for running Hadoop jobs. If it detects no running Hadoop jobs for a given period, it will automatically deallocate its own cluster and thus free up nodes which are not being used effectively.

Note: While the cluster is deallocated, the *cluster directory* is not cleaned up automatically. The user must deallocate this cluster through the regular *deallocate* operation to clean this up.

2.2.8. Specifying Additional Job Attributes

HOD allows the user to specify a wallclock time and a name (or title) for a Torque job.

The wallclock time is the estimated amount of time for which the Torque job will be valid. After this time has expired, Torque will automatically delete the job and free up the nodes. Specifying the wallclock time can also help the job scheduler to better schedule jobs, and help improve utilization of cluster resources.

To specify the wallclock time, use the following syntax:

```
$ hod allocate -d cluster_dir -n number_of_nodes -l time_in_seconds
```

The name or title of a Torque job helps in user friendly identification of the job. The string specified here will show up in all information where Torque job attributes are displayed, including the `qstat` command.

To specify the name or title, use the following syntax:

```
$ hod allocate -d cluster_dir -n number_of_nodes -N name_of_job
```

Note: Due to restriction in the underlying Torque resource manager, names which do not start with an alphabet character or contain a 'space' will cause the job to fail. The failure message points to the problem being in the specified job name.

2.2.9. Capturing HOD Exit Codes in Torque

HOD exit codes are captured in the Torque exit_status field. This will help users and system

administrators to distinguish successful runs from unsuccessful runs of HOD. The exit codes are 0 if allocation succeeded and all hadoop jobs ran on the allocated cluster correctly. They are non-zero if allocation failed or some of the hadoop jobs failed on the allocated cluster. The exit codes that are possible are mentioned in the table below. *Note: Hadoop job status is captured only if the version of Hadoop used is 16 or above.*

Exit Code	Meaning
6	Ringmaster failure
7	HDFS failure
8	Job tracker failure
10	Cluster dead
12	Cluster already allocated
13	HDFS dead
14	Mapred dead
16	All MapReduce jobs that ran on the cluster failed. Refer to hadoop logs for more details.
17	Some of the MapReduce jobs that ran on the cluster failed. Refer to hadoop logs for more details.

2.2.10. Command Line

HOD command line has the following general syntax:

```
hod <operation> [ARGS] [OPTIONS]
```

Allowed operations are 'allocate', 'deallocate', 'info', 'list', 'script' and 'help'. For help with a particular operation do:

```
hod help <operation>
```

To have a look at possible options do:

```
hod help options
```

- *allocate*
  Usage : hod allocate -d cluster_dir -n number_of_nodes [OPTIONS]
  Allocates a cluster on the given number of cluster nodes, and store the allocation information in cluster_dir for use with subsequent `hadoop` commands. Note that the `cluster_dir` must exist before running the command.
- *list*
  Usage : hod list [OPTIONS]

Lists the clusters allocated by this user. Information provided includes the Torque job id corresponding to the cluster, the cluster directory where the allocation information is stored, and whether the MapReduce daemon is still active or not.

- *info*
  Usage : hod info -d cluster_dir [OPTIONS]
  Lists information about the cluster whose allocation information is stored in the specified cluster directory.
- *deallocate*
  Usage : hod deallocate -d cluster_dir [OPTIONS]
  Deallocates the cluster whose allocation information is stored in the specified cluster directory.
- *script*
  Usage : hod script -s script_file -d cluster_directory -n number_of_nodes [OPTIONS]
  Runs a hadoop script using HOD*script* operation. Provisions Hadoop on a given number of nodes, executes the given script from the submitting node, and deallocates the cluster when the script completes.
- *help*
  Usage : hod help [operation | 'options']
  When no argument is specified, `hod help` gives the usage and basic options, and is equivalent to `hod --help` (See below). When 'options' is given as argument, hod displays only the basic options that hod takes. When an operation is specified, it displays the usage and description corresponding to that particular operation. For e.g, to know about allocate operation, one can do a `hod help allocate`

Besides the operations, HOD can take the following command line options.

- *--help*
  Prints out the help message to see the usage and basic options.
- *--verbose-help*
  All configuration options provided in the hodrc file can be passed on the command line, using the syntax `--section_name.option_name[=value]`. When provided this way, the value provided on command line overrides the option provided in hodrc. The verbose-help command lists all the available options in the hodrc file. This is also a nice way to see the meaning of the configuration options.
  "

See Options Configuring HOD for a description of most important hod configuration options. For basic options do `hod help options` and for all options possible in hod configuration do `hod --verbose-help`. See HOD Configuration for a description of all options.

2.2.11. Options Configuring HOD

As described above, HOD is configured using a configuration file that is usually set up by system administrators. This is a INI style configuration file that is divided into sections, and options inside each section. Each section relates to one of the HOD processes: client, ringmaster, hodring, mapreduce or hdfs. The options inside a section comprise of an option name and value.

Users can override the configuration defined in the default configuration in two ways:

- Users can supply their own configuration file to HOD in each of the commands, using the `-c` option
- Users can supply specific configuration options to HOD/ Options provided on command line *override* the values provided in the configuration file being used.

This section describes some of the most commonly used configuration options. These commonly used options are provided with a *short* option for convenience of specification. All other options can be specified using a *long* option that is also described below.

- *-c config_file*
  Provides the configuration file to use. Can be used with all other options of HOD. Alternatively, the `HOD_CONF_DIR` environment variable can be defined to specify a directory that contains a file named `hodrc`, alleviating the need to specify the configuration file in each HOD command.
- *-d cluster_dir*
  This is required for most of the hod operations. As described under Create a Cluster Directory, the *cluster directory* is a directory on the local file system where `hod` will generate the Hadoop configuration, *hadoop-site.xml*, corresponding to the cluster it allocates. Pass it to the `hod` operations as an argument to -d or --hod.clusterdir. If it doesn't already exist, HOD will automatically try to create it and use it. Once a cluster is allocated, a user can utilize it to run Hadoop jobs by specifying the clusterdirectory as the Hadoop --config option.
- *-n number_of_nodes*
  This is required for the hod 'allocation' operation and for script operation. This denotes the number of nodes to be allocated.
- *-s script-file*
  Required when using script operation, specifies the script file to execute.
- *-b 1|2|3|4*
  Enables the given debug level. Can be used with all other options of HOD. 4 is most verbose.
- *-t hadoop_tarball*
  Provisions Hadoop from the given tar.gz file. This option is only applicable to the *allocate* operation. For better distribution performance it is strongly recommended that the Hadoop tarball is created *after* removing the source or documentation.

- *-N job-name*
  The Name to give to the resource manager job that HOD uses underneath. For e.g. in the case of Torque, this translates to the `qsub -N` option, and can be seen as the job name using the `qstat` command.
- *-l wall-clock-time*
  The amount of time for which the user expects to have work on the allocated cluster. This is passed to the resource manager underneath HOD, and can be used in more efficient scheduling and utilization of the cluster. Note that in the case of Torque, the cluster is automatically deallocated after this time expires.
- *-j java-home*
  Path to be set to the JAVA_HOME environment variable. This is used in the *script* operation. HOD sets the JAVA_HOME environment variable tot his value and launches the user script in that.
- *-A account-string*
  Accounting information to pass to underlying resource manager.
- *-Q queue-name*
  Name of the queue in the underlying resource manager to which the job must be submitted.
- *-Mkey1=value1 -Mkey2=value2*
  Provides configuration parameters for the provisioned MapReduce daemons (JobTracker and TaskTrackers). A hadoop-site.xml is generated with these values on the cluster nodes.
  Note: Values which have the following characters: space, comma, equal-to, semi-colon need to be escaped with a '\' character, and need to be enclosed within quotes. You can escape a '\' with a '\' too.
- *-Hkey1=value1 -Hkey2=value2*
  Provides configuration parameters for the provisioned HDFS daemons (NameNode and DataNodes). A hadoop-site.xml is generated with these values on the cluster nodes
  Note: Values which have the following characters: space, comma, equal-to, semi-colon need to be escaped with a '\' character, and need to be enclosed within quotes. You can escape a '\' with a '\' too.
- *-Ckey1=value1 -Ckey2=value2*
  Provides configuration parameters for the client from where jobs can be submitted. A hadoop-site.xml is generated with these values on the submit node.
  Note: Values which have the following characters: space, comma, equal-to, semi-colon need to be escaped with a '\' character, and need to be enclosed within quotes. You can escape a '\' with a '\' too.
- *--section-name.option-name=value*
  This is the method to provide options using the *long* format. For e.g. you could say *--hod.script-wait-time=20*

2.3. Troubleshooting

The following section identifies some of the most likely error conditions users can run into when using HOD and ways to trouble-shoot them

2.3.1. HOD Hangs During Allocation

Possible Cause: One of the HOD or Hadoop components have failed to come up. In such a case, the `hod` command will return after a few minutes (typically 2-3 minutes) with an error code of either 7 or 8 as defined in the Error Codes section. Refer to that section for further details.

Possible Cause: A large allocation is fired with a tarball. Sometimes due to load in the network, or on the allocated nodes, the tarball distribution might be significantly slow and take a couple of minutes to come back. Wait for completion. Also check that the tarball does not have the Hadoop sources or documentation.

Possible Cause: A Torque related problem. If the cause is Torque related, the `hod` command will not return for more than 5 minutes. Running `hod` in debug mode may show the `qstat` command being executed repeatedly. Executing the `qstat` command from a separate shell may show that the job is in the `Q` (Queued) state. This usually indicates a problem with Torque. Possible causes could include some nodes being down, or new nodes added that Torque is not aware of. Generally, system administator help is needed to resolve this problem.

2.3.2. HOD Hangs During Deallocation

Possible Cause: A Torque related problem, usually load on the Torque server, or the allocation is very large. Generally, waiting for the command to complete is the only option.

2.3.3. HOD Fails With an Error Code and Error Message

If the exit code of the `hod` command is not `0`, then refer to the following table of error exit codes to determine why the code may have occurred and how to debug the situation.

Error Codes

Error Code	Meaning	Possible Causes and Remedial Actions
1	Configuration error	Incorrect configuration values specified in hodrc, or other errors related to HOD

		configuration. The error messages in this case must be sufficient to debug and fix the problem.
2	Invalid operation	Do `hod help` for the list of valid operations.
3	Invalid operation arguments	Do `hod help operation` for listing the usage of a particular operation.
4	Scheduler failure	1. Requested more resources than available. Run `checknodes cluster_name` to see if enough nodes are available. 2. Requested resources exceed resource manager limits. 3. Torque is misconfigured, the path to Torque binaries is misconfigured, or other Torque problems. Contact system administrator.
5	Job execution failure	1. Torque Job was deleted from outside. Execute the Torque `qstat` command to see if you have any jobs in the R (Running) state. If none exist, try re-executing HOD. 2. Torque problems such as the server momentarily going down, or becoming unresponsive. Contact system administrator. 3. The system administrator might have configured account verification, and an invalid account is specified. Contact system administrator.
6	Ringmaster failure	HOD prints the message "Cluster could not be allocated because of the following errors on the ringmaster host <hostname>". The actual error

		message may indicate one of the following: 1. Invalid configuration on the node running the ringmaster, specified by the hostname in the error message. 2. Invalid configuration in the `ringmaster` section, 3. Invalid `pkgs` option in `gridservice-mapred` or `gridservice-hdfs` section, 4. An invalid hadoop tarball, or a tarball which has bundled an invalid configuration file in the conf directory, 5. Mismatched version in Hadoop between the MapReduce and an external HDFS. The Torque `qstat` command will most likely show a job in the `C` (Completed) state. One can login to the ringmaster host as given by HOD failure message and debug the problem with the help of the error message. If the error message doesn't give complete information, ringmaster logs should help finding out the root cause of the problem. Refer to the section *Locating Ringmaster Logs* below for more information.
7	HDFS failure	When HOD fails to allocate due to HDFS failures (or Job tracker failures, error code 8, see below), it prints a failure message "Hodring at <hostname> failed with following errors:" and then gives the actual error message, which may indicate one of the following: 1. Problem in starting Hadoop clusters. Usually the actual cause in the error message will

		indicate the problem on the hostname mentioned. Also, review the Hadoop related configuration in the HOD configuration files. Look at the Hadoop logs using information specified in *Collecting and Viewing Hadoop Logs* section above. 2. Invalid configuration on the node running the hodring, specified by the hostname in the error message 3. Invalid configuration in the `hodring` section of hodrc. `ssh` to the hostname specified in the error message and grep for `ERROR` or `CRITICAL` in hodring logs. Refer to the section *Locating Hodring Logs* below for more information. 4. Invalid tarball specified which is not packaged correctly. 5. Cannot communicate with an externally configured HDFS. When such HDFS or Job tracker failure occurs, one can login into the host with hostname mentioned in HOD failure message and debug the problem. While fixing the problem, one should also review other log messages in the ringmaster log to see which other machines also might have had problems bringing up the jobtracker/namenode, apart from the hostname that is reported in the failure message. This possibility of other machines also having problems occurs because HOD continues to try and launch hadoop daemons on multiple machines one after another depending upon the value of the configuration variable

		ringmaster.max-master-failures. See Locating Ringmaster Logs for more information.
8	Job tracker failure	Similar to the causes in *DFS failure* case.
10	Cluster dead	1. Cluster was auto-deallocated because it was idle for a long time. 2. Cluster was auto-deallocated because the wallclock time specified by the system administrator or user was exceeded. 3. Cannot communicate with the JobTracker and HDFS NameNode which were successfully allocated. Deallocate the cluster, and allocate again.
12	Cluster already allocated	The cluster directory specified has been used in a previous allocate operation and is not yet deallocated. Specify a different directory, or deallocate the previous allocation first.
13	HDFS dead	Cannot communicate with the HDFS NameNode. HDFS NameNode went down.
14	Mapred dead	1. Cluster was auto-deallocated because it was idle for a long time. 2. Cluster was auto-deallocated because the wallclock time specified by the system administrator or user was exceeded. 3. Cannot communicate with the MapReduce JobTracker. JobTracker node went down.
15	Cluster not allocated	An operation which requires an allocated cluster is given a cluster directory with no state information.

Any non-zero exit code	HOD script error	If the hod script option was used, it is likely that the exit code is from the script. Unfortunately, this could clash with the exit codes of the hod command itself. In order to help users differentiate these two, hod writes the script's exit code to a file called script.exitcode in the cluster directory, if the script returned an exit code. You can cat this file to determine the script's exit code. If it does not exist, then it is a hod command exit code.

2.3.4. Hadoop DFSClient Warns with a NotReplicatedYetException

Sometimes, when you try to upload a file to the HDFS immediately after allocating a HOD cluster, DFSClient warns with a NotReplicatedYetException. It usually shows a message something like -

```
WARN hdfs.DFSClient: NotReplicatedYetException sleeping  <filename> retries
left 3
08/01/25 16:31:40 INFO hdfs.DFSClient:
org.apache.hadoop.ipc.RemoteException: java.io.IOException:
File <filename> could only be replicated to 0 nodes, instead of 1
```

This scenario arises when you try to upload a file to the HDFS while the DataNodes are still in the process of contacting the NameNode. This can be resolved by waiting for some time before uploading a new file to the HDFS, so that enough DataNodes start and contact the NameNode.

2.3.5. Hadoop Jobs Not Running on a Successfully Allocated Cluster

This scenario generally occurs when a cluster is allocated, and is left inactive for sometime, and then hadoop jobs are attempted to be run on them. Then Hadoop jobs fail with the following exception:

```
08/01/25 16:31:40 INFO ipc.Client: Retrying connect to server:
foo.bar.com/1.1.1.1:53567. Already tried 1 time(s).
```

Possible Cause: No Hadoop jobs were run for a significant portion of time. Thus the cluster would have got deallocated as described in the section *Auto-deallocation of Idle Clusters*. Deallocate the cluster and allocate it again.

Possible Cause: The wallclock limit specified by the Torque administrator or the `-l` option

defined in the section *Specifying Additional Job Attributes* was exceeded since allocation time. Thus the cluster would have got released. Deallocate the cluster and allocate it again.

Possible Cause: There is a version mismatch between the version of the hadoop being used in provisioning (typically via the tarball option) and the external HDFS. Ensure compatible versions are being used.

Possible Cause: There is a version mismatch between the version of the hadoop client being used to submit jobs and the hadoop used in provisioning (typically via the tarball option). Ensure compatible versions are being used.

Possible Cause: You used one of the options for specifying Hadoop configuration `-M or -H`, which had special characters like space or comma that were not escaped correctly. Refer to the section *Options Configuring HOD* for checking how to specify such options correctly.

2.3.6. My Hadoop Job Got Killed

Possible Cause: The wallclock limit specified by the Torque administrator or the `-l` option defined in the section *Specifying Additional Job Attributes* was exceeded since allocation time. Thus the cluster would have got released. Deallocate the cluster and allocate it again, this time with a larger wallclock time.

Possible Cause: Problems with the JobTracker node. Refer to the section in *Collecting and Viewing Hadoop Logs* to get more information.

2.3.7. Hadoop Job Fails with Message: 'Job tracker still initializing'

Possible Cause: The hadoop job was being run as part of the HOD script command, and it started before the JobTracker could come up fully. Allocate the cluster using a large value for the configuration option `--hod.script-wait-time`. Typically a value of 120 should work, though it is typically unnecessary to be that large.

2.3.8. The Exit Codes For HOD Are Not Getting Into Torque

Possible Cause: Version 0.16 of hadoop is required for this functionality to work. The version of Hadoop used does not match. Use the required version of Hadoop.

Possible Cause: The deallocation was done without using the `hod` command; for e.g. directly using `qdel`. When the cluster is deallocated in this manner, the HOD processes are terminated using signals. This results in the exit code to be based on the signal number, rather than the exit code of the program.

2.3.9. The Hadoop Logs are Not Uploaded to HDFS

Possible Cause: There is a version mismatch between the version of the hadoop being used for uploading the logs and the external HDFS. Ensure that the correct version is specified in the `hodring.pkgs` option.

2.3.10. Locating Ringmaster Logs

To locate the ringmaster logs, follow these steps:

- Execute hod in the debug mode using the -b option. This will print the Torque job id for the current run.
- Execute `qstat -f torque_job_id` and look up the value of the `exec_host` parameter in the output. The first host in this list is the ringmaster node.
- Login to this node.
- The ringmaster log location is specified by the `ringmaster.log-dir` option in the hodrc. The name of the log file will be `username.torque_job_id/ringmaster-main.log`.
- If you don't get enough information, you may want to set the ringmaster debug level to 4. This can be done by passing `--ringmaster.debug 4` to the hod command line.

2.3.11. Locating Hodring Logs

To locate hodring logs, follow the steps below:

- Execute hod in the debug mode using the -b option. This will print the Torque job id for the current run.
- Execute `qstat -f torque_job_id` and look up the value of the `exec_host` parameter in the output. All nodes in this list should have a hodring on them.
- Login to any of these nodes.
- The hodring log location is specified by the `hodring.log-dir` option in the hodrc. The name of the log file will be `username.torque_job_id/hodring-main.log`.
- If you don't get enough information, you may want to set the hodring debug level to 4. This can be done by passing `--hodring.debug 4` to the hod command line.

3. HOD Administrators

This section show administrators how to install, configure and run HOD.

3.1. Getting Started

The basic system architecture of HOD includes these components:

- A Resource manager, possibly together with a scheduler (see Prerequisites)
- Various HOD components
- Hadoop MapReduce and HDFS daemons

HOD provisions and maintains Hadoop MapReduce and, optionally, HDFS instances through interaction with the above components on a given cluster of nodes. A cluster of nodes can be thought of as comprising two sets of nodes:

- Submit nodes: Users use the HOD client on these nodes to allocate clusters, and then use the Hadoop client to submit Hadoop jobs.
- Compute nodes: Using the resource manager, HOD components are run on these nodes to provision the Hadoop daemons. After that Hadoop jobs run on them.

Here is a brief description of the sequence of operations in allocating a cluster and running jobs on them.

- The user uses the HOD client on the Submit node to allocate a desired number of cluster nodes and to provision Hadoop on them.
- The HOD client uses a resource manager interface (qsub, in Torque) to submit a HOD process, called the RingMaster, as a Resource Manager job, to request the user's desired number of nodes. This job is submitted to the central server of the resource manager (pbs_server, in Torque).
- On the compute nodes, the resource manager slave daemons (pbs_moms in Torque) accept and run jobs that they are assigned by the central server (pbs_server in Torque). The RingMaster process is started on one of the compute nodes (mother superior, in Torque).
- The RingMaster then uses another resource manager interface (pbsdsh, in Torque) to run the second HOD component, HodRing, as distributed tasks on each of the compute nodes allocated.
- The HodRings, after initializing, communicate with the RingMaster to get Hadoop commands, and run them accordingly. Once the Hadoop commands are started, they register with the RingMaster, giving information about the daemons.
- All the configuration files needed for Hadoop instances are generated by HOD itself, some obtained from options given by user in its own configuration file.
- The HOD client keeps communicating with the RingMaster to find out the location of the JobTracker and HDFS daemons.

3.2. Prerequisites

To use HOD, your system should include the following components.

- Operating System: HOD is currently tested on RHEL4.
- Nodes: HOD requires a minimum of three nodes configured through a resource manager.

- Software: The following components must be installed on ALL nodes before using HOD:
  - Torque: Resource manager
  - Python : HOD requires version 2.5.1 of Python.
- Software (optional): The following components are optional and can be installed to obtain better functionality from HOD:
  - Twisted Python: This can be used for improving the scalability of HOD. If this module is detected to be installed, HOD uses it, else it falls back to default modules.
  - Hadoop: HOD can automatically distribute Hadoop to all nodes in the cluster. However, it can also use a pre-installed version of Hadoop, if it is available on all nodes in the cluster. HOD currently supports Hadoop 0.15 and above.

Note: HOD configuration requires the location of installs of these components to be the same on all nodes in the cluster. It will also make the configuration simpler to have the same location on the submit nodes.

3.3. Resource Manager

Currently HOD works with the Torque resource manager, which it uses for its node allocation and job submission. Torque is an open source resource manager from Cluster Resources, a community effort based on the PBS project. It provides control over batch jobs and distributed compute nodes. Torque is freely available for download from here.

All documentation related to torque can be seen under the section TORQUE Resource Manager here. You can get wiki documentation from here. Users may wish to subscribe to TORQUE's mailing list or view the archive for questions, comments here.

To use HOD with Torque:

- Install Torque components: pbs_server on one node (head node), pbs_mom on all compute nodes, and PBS client tools on all compute nodes and submit nodes. Perform at least a basic configuration so that the Torque system is up and running, that is, pbs_server knows which machines to talk to. Look here for basic configuration. For advanced configuration, see here
- Create a queue for submitting jobs on the pbs_server. The name of the queue is the same as the HOD configuration parameter, resource-manager.queue. The HOD client uses this queue to submit the RingMaster process as a Torque job.
- Specify a cluster name as a property for all nodes in the cluster. This can be done by using the qmgr command. For example: `qmgr -c "set node node properties=cluster-name"`. The name of the cluster is the same as the HOD configuration parameter, hod.cluster.
- Make sure that jobs can be submitted to the nodes. This can be done by using the qsub

command. For example: `echo "sleep 30" | qsub -l nodes=3`

3.4. Installing HOD

Once the resource manager is set up, you can obtain and install HOD.

- If you are getting HOD from the Hadoop tarball, it is available under the 'contrib' section of Hadoop, under the root directory 'hod'.
- If you are building from source, you can run ant tar from the Hadoop root directory to generate the Hadoop tarball, and then get HOD from there, as described above.
- Distribute the files under this directory to all the nodes in the cluster. Note that the location where the files are copied should be the same on all the nodes.
- Note that compiling hadoop would build HOD with appropriate permissions set on all the required script files in HOD.

3.5. Configuring HOD

You can configure HOD once it is installed. The minimal configuration needed to run HOD is described below. More advanced configuration options are discussed in the HOD Configuration.

3.5.1. Minimal Configuration

To get started using HOD, the following minimal configuration is required:

- On the node from where you want to run HOD, edit the file hodrc located in the <install dir>/conf directory. This file contains the minimal set of values required to run hod.
- Specify values suitable to your environment for the following variables defined in the configuration file. Note that some of these variables are defined at more than one place in the file.
  - ${JAVA_HOME}: Location of Java for Hadoop. Hadoop supports Sun JDK 1.6.x and above.
  - ${CLUSTER_NAME}: Name of the cluster which is specified in the 'node property' as mentioned in resource manager configuration.
  - ${HADOOP_HOME}: Location of Hadoop installation on the compute and submit nodes.
  - ${RM_QUEUE}: Queue configured for submitting jobs in the resource manager configuration.
  - ${RM_HOME}: Location of the resource manager installation on the compute and submit nodes.
- The following environment variables may need to be set depending on your environment.

These variables must be defined where you run the HOD client and must also be specified in the HOD configuration file as the value of the key resource_manager.env-vars. Multiple variables can be specified as a comma separated list of key=value pairs.

- HOD_PYTHON_HOME: If you install python to a non-default location of the compute nodes, or submit nodes, then this variable must be defined to point to the python executable in the non-standard location.

3.5.2. Advanced Configuration

You can review and modify other configuration options to suit your specific needs. See HOD Configuration for more information.

3.6. Running HOD

You can run HOD once it is configured. Refer to HOD Users for more information.

3.7. Supporting Tools and Utilities

This section describes supporting tools and utilities that can be used to manage HOD deployments.

3.7.1. logcondense.py - Manage Log Files

As mentioned under Collecting and Viewing Hadoop Logs, HOD can be configured to upload Hadoop logs to a statically configured HDFS. Over time, the number of logs uploaded to HDFS could increase. logcondense.py is a tool that helps administrators to remove log files uploaded to HDFS.

3.7.1.1. Running logcondense.py

logcondense.py is available under hod_install_location/support folder. You can either run it using python, for example, *python logcondense.py*, or give execute permissions to the file, and directly run it as *logcondense.py*. logcondense.py needs to be run by a user who has sufficient permissions to remove files from locations where log files are uploaded in the HDFS, if permissions are enabled. For example as mentioned under hodring options, the logs could be configured to come under the user's home directory in HDFS. In that case, the user running logcondense.py should have super user privileges to remove the files from under all user home directories.

3.7.1.2. Command Line Options for logcondense.py

The following command line options are supported for logcondense.py.

Short Option	Long option	Meaning	Example
-p	--package	Complete path to the hadoop script. The version of hadoop must be the same as the one running HDFS.	/usr/bin/hadoop
-d	--days	Delete log files older than the specified number of days	7
-c	--config	Path to the Hadoop configuration directory, under which hadoop-site.xml resides. The hadoop-site.xml must point to the HDFS NameNode from where logs are to be removed.	/home/foo/hadoop/conf
-l	--logs	A HDFS path, this must be the same HDFS path as specified for the log-destination-uri, as mentioned under hodring options, without the hdfs:// URI string	/user
-n	--dynamicdfs	If true, this will indicate that the logcondense.py script should delete HDFS logs in addition to MapReduce logs. Otherwise, it only deletes MapReduce logs, which is also the default if this option is not specified. This	false

		option is useful if dynamic HDFS installations are being provisioned by HOD, and the static HDFS installation is being used only to collect logs - a scenario that may be common in test clusters.	
-r	--retain-master-logs	If true, this will keep the JobTracker logs of job in hod-logs inside HDFS and it will delete only the TaskTracker logs. Also, this will keep the Namenode logs along with JobTracker logs and will only delete the Datanode logs if 'dynamicdfs' options is set to true. Otherwise, it will delete the complete job directory from hod-logs inside HDFS. By default it is set to false.	false

So, for example, to delete all log files older than 7 days using a hadoop-site.xml stored in ~/hadoop-conf, using the hadoop installation under ~/hadoop-0.17.0, you could say:

python logcondense.py -p ~/hadoop-0.17.0/bin/hadoop -d 7 -c ~/hadoop-conf -l /user

3.7.2. checklimits.sh - Monitor Resource Limits

checklimits.sh is a HOD tool specific to the Torque/Maui environment (Maui Cluster Scheduler is an open source job scheduler for clusters and supercomputers, from clusterresources). The checklimits.sh script updates the torque comment field when newly submitted job(s) violate or exceed over user limits set up in Maui scheduler. It uses qstat, does one pass over the torque job-list to determine queued or unfinished jobs, runs Maui tool checkjob on each job to see if user limits are violated and then runs torque's qalter utility to update job attribute 'comment'. Currently it updates the comment as *User-limits exceeded. Requested:([0-9]*) Used:([0-9]*) MaxLimit:([0-9]*)* for those jobs that violate limits. This

comment field is then used by HOD to behave accordingly depending on the type of violation.

3.7.2.1. Running checklimits.sh

checklimits.sh is available under the hod_install_location/support folder. This shell script can be run directly as *sh checklimits.sh* or as *./checklimits.sh* after enabling execute permissions. Torque and Maui binaries should be available on the machine where the tool is run and should be in the path of the shell script process. To update the comment field of jobs from different users, this tool must be run with torque administrative privileges. This tool must be run repeatedly after specific intervals of time to frequently update jobs violating constraints, for example via cron. Please note that the resource manager and scheduler commands used in this script can be expensive and so it is better not to run this inside a tight loop without sleeping.

3.7.3. verify-account Script

Production systems use accounting packages to charge users for using shared compute resources. HOD supports a parameter *resource_manager.pbs-account* to allow users to identify the account under which they would like to submit jobs. It may be necessary to verify that this account is a valid one configured in an accounting system. The *hod-install-dir/bin/verify-account* script provides a mechanism to plug-in a custom script that can do this verification.

3.7.3.1. Integrating the verify-account script with HOD

HOD runs the *verify-account* script passing in the *resource_manager.pbs-account* value as argument to the script, before allocating a cluster. Sites can write a script that verify this account against their accounting systems. Returning a non-zero exit code from this script will cause HOD to fail allocation. Also, in case of an error, HOD will print the output of script to the user. Any descriptive error message can be passed to the user from the script in this manner.

The default script that comes with the HOD installation does not do any validation, and returns a zero exit code.

If the verify-account script is not found, then HOD will treat that verification is disabled, and continue allocation as is.

4. HOD Configuration

This section discusses how to work with the HOD configuration options.

4.1. Getting Started

Configuration options can be specified in two ways: as a configuration file in the INI format and as command line options to the HOD shell, specified in the format --section.option[=value]. If the same option is specified in both places, the value specified on the command line overrides the value in the configuration file.

To get a simple description of all configuration options use:

```
$ hod --verbose-help
```

4.2. Configuation Options

HOD organizes configuration options into these sections:

- common: Options that appear in more than one section. Options defined in a section are used by the process for which that section applies. Common options have the same meaning, but can have different values in each section.
- hod: Options for the HOD client
- resource_manager: Options for specifying which resource manager to use, and other parameters for using that resource manager
- ringmaster: Options for the RingMaster process,
- hodring: Options for the HodRing processes
- gridservice-mapred: Options for the MapReduce daemons
- gridservice-hdfs: Options for the HDFS daemons.

4.2.1. common options

- temp-dir: Temporary directory for usage by the HOD processes. Make sure that the users who will run hod have rights to create directories under the directory specified here. If you wish to make this directory vary across allocations, you can make use of the environmental variables which will be made available by the resource manager to the HOD processes. For example, in a Torque setup, having --ringmaster.temp-dir=/tmp/hod-temp-dir.$PBS_JOBID would let ringmaster use different temp-dir for each allocation; Torque expands this variable before starting the ringmaster.
- debug: Numeric value from 1-4. 4 produces the most log information, and 1 the least.
- log-dir: Directory where log files are stored. By default, this is <install-location>/logs/. The restrictions and notes for the temp-dir variable apply here too.
- xrs-port-range: Range of ports, among which an available port shall be picked for use to run an XML-RPC server.
- http-port-range: Range of ports, among which an available port shall be picked for use to run an HTTP server.

- java-home: Location of Java to be used by Hadoop.
- syslog-address: Address to which a syslog daemon is bound to. The format of the value is host:port. If configured, HOD log messages will be logged to syslog using this value.

4.2.2. hod options

- cluster: Descriptive name given to the cluster. For Torque, this is specified as a 'Node property' for every node in the cluster. HOD uses this value to compute the number of available nodes.
- client-params: Comma-separated list of hadoop config parameters specified as key-value pairs. These will be used to generate a hadoop-site.xml on the submit node that should be used for running MapReduce jobs.
- job-feasibility-attr: Regular expression string that specifies whether and how to check job feasibility - resource manager or scheduler limits. The current implementation corresponds to the torque job attribute 'comment' and by default is disabled. When set, HOD uses it to decide what type of limit violation is triggered and either deallocates the cluster or stays in queued state according as the request is beyond maximum limits or the cumulative usage has crossed maximum limits. The torque comment attribute may be updated periodically by an external mechanism. For example, comment attribute can be updated by running checklimits.sh script in hod/support directory, and then setting job-feasibility-attr equal to the value TORQUE_USER_LIMITS_COMMENT_FIELD, "User-limits exceeded. Requested:([0-9]*) Used:([0-9]*) MaxLimit:([0-9]*)", will make HOD behave accordingly.

4.2.3. resource_manager options

- queue: Name of the queue configured in the resource manager to which jobs are to be submitted.
- batch-home: Install directory to which 'bin' is appended and under which the executables of the resource manager can be found.
- env-vars: Comma-separated list of key-value pairs, expressed as key=value, which would be passed to the jobs launched on the compute nodes. For example, if the python installation is in a non-standard location, one can set the environment variable 'HOD_PYTHON_HOME' to the path to the python executable. The HOD processes launched on the compute nodes can then use this variable.
- options: Comma-separated list of key-value pairs, expressed as <option>:<sub-option>=<value>. When passing to the job submission program, these are expanded as -<option> <sub-option>=<value>. These are generally used for specifying additional resource contraints for scheduling. For instance, with a Torque setup, one can specify --resource_manager.options='l:arch=x86_64' for constraining the nodes being allocated to a particular architecture; this option will be passed to Torque's qsub

command as "-l arch=x86_64".

4.2.4. ringmaster options

- work-dirs: Comma-separated list of paths that will serve as the root for directories that HOD generates and passes to Hadoop for use to store DFS and MapReduce data. For example, this is where DFS data blocks will be stored. Typically, as many paths are specified as there are disks available to ensure all disks are being utilized. The restrictions and notes for the temp-dir variable apply here too.
- max-master-failures: Number of times a hadoop master daemon can fail to launch, beyond which HOD will fail the cluster allocation altogether. In HOD clusters, sometimes there might be a single or few "bad" nodes due to issues like missing java, missing or incorrect version of Hadoop etc. When this configuration variable is set to a positive integer, the RingMaster returns an error to the client only when the number of times a hadoop master (JobTracker or NameNode) fails to start on these bad nodes because of above issues, exceeds the specified value. If the number is not exceeded, the next HodRing which requests for a command to launch is given the same hadoop master again. This way, HOD tries its best for a successful allocation even in the presence of a few bad nodes in the cluster.
- workers_per_ring: Number of workers per service per HodRing. By default this is set to 1. If this configuration variable is set to a value 'n', the HodRing will run 'n' instances of the workers (TaskTrackers or DataNodes) on each node acting as a slave. This can be used to run multiple workers per HodRing, so that the total number of workers in a HOD cluster is not limited by the total number of nodes requested during allocation. However, note that this will mean each worker should be configured to use only a proportional fraction of the capacity of the resources on the node. In general, this feature is only useful for testing and simulation purposes, and not for production use.

4.2.5. gridservice-hdfs options

- external: If false, indicates that a HDFS cluster must be bought up by the HOD system, on the nodes which it allocates via the allocate command. Note that in that case, when the cluster is de-allocated, it will bring down the HDFS cluster, and all the data will be lost. If true, it will try and connect to an externally configured HDFS system. Typically, because input for jobs are placed into HDFS before jobs are run, and also the output from jobs in HDFS is required to be persistent, an internal HDFS cluster is of little value in a production system. However, it allows for quick testing.
- host: Hostname of the externally configured NameNode, if any
- fs_port: Port to which NameNode RPC server is bound.
- info_port: Port to which the NameNode web UI server is bound.
- pkgs: Installation directory, under which bin/hadoop executable is located. This can be

used to use a pre-installed version of Hadoop on the cluster.
- server-params: Comma-separated list of hadoop config parameters specified key-value pairs. These will be used to generate a hadoop-site.xml that will be used by the NameNode and DataNodes.
- final-server-params: Same as above, except they will be marked final.

4.2.6. gridservice-mapred options

- external: If false, indicates that a MapReduce cluster must be bought up by the HOD system on the nodes which it allocates via the allocate command. If true, if will try and connect to an externally configured MapReduce system.
- host: Hostname of the externally configured JobTracker, if any
- tracker_port: Port to which the JobTracker RPC server is bound
- info_port: Port to which the JobTracker web UI server is bound.
- pkgs: Installation directory, under which bin/hadoop executable is located
- server-params: Comma-separated list of hadoop config parameters specified key-value pairs. These will be used to generate a hadoop-site.xml that will be used by the JobTracker and TaskTrackers
- final-server-params: Same as above, except they will be marked final.

4.2.7. hodring options

- mapred-system-dir-root: Directory in the DFS under which HOD will generate sub-directory names and pass the full path as the value of the 'mapred.system.dir' configuration parameter to Hadoop daemons. The format of the full path will be value-of-this-option/userid/mapredsystem/cluster-id. Note that the directory specified here should be such that all users can create directories under this, if permissions are enabled in HDFS. Setting the value of this option to /user will make HOD use the user's home directory to generate the mapred.system.dir value.
- log-destination-uri: URL describing a path in an external, static DFS or the cluster node's local file system where HOD will upload Hadoop logs when a cluster is deallocated. To specify a DFS path, use the format 'hdfs://path'. To specify a cluster node's local file path, use the format 'file://path'. When clusters are deallocated by HOD, the hadoop logs will be deleted as part of HOD's cleanup process. To ensure these logs persist, you can use this configuration option. The format of the path is value-of-this-option/userid/hod-logs/cluster-id Note that the directory you specify here must be such that all users can create sub-directories under this. Setting this value to hdfs://user will make the logs come in the user's home directory in DFS.
- pkgs: Installation directory, under which bin/hadoop executable is located. This will be used by HOD to upload logs if a HDFS URL is specified in log-destination-uri option. Note that this is useful if the users are using a tarball whose version may differ from the

external, static HDFS version.

- hadoop-port-range: Range of ports, among which an available port shall be picked for use to run a Hadoop Service, like JobTracker or TaskTracker.