Hadoop On Demand

Table of contents

1. Overview

The Hadoop On Demand (HOD) project is a system for provisioning and managing independent Hadoop MapReduce and 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. It 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 relies on a resource manager (RM) for allocation of nodes that it can use for running Hadoop instances. At present it runs with the Torque resource manager.

The basic system architecture of HOD includes components from:

- A Resource manager (possibly together with a scheduler),
- HOD components, and
- Hadoop Map/Reduce and HDFS daemons.

HOD provisions and maintains Hadoop Map/Reduce 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 of 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 required number of cluster nodes, and 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, requesting the user 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 given 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.

The rest of the document deals with the steps needed to setup HOD on a physical cluster of nodes.

2. Pre-requisites

Operating System: HOD is currently tested on RHEL4.
Nodes : HOD requires a minimum of 3 nodes configured through a resource manager.

Software

The following components are to be installed on *ALL* the nodes before using HOD:

- Torque: Resource manager
- Python : HOD requires version 2.5.1 of Python.

The following components can be optionally installed for getting 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. 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.

For using 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 atleast a basic configuration so that the Torque system is up and running i.e 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.
- Ensure 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

4. Installing HOD

Now that the resource manager set up is done, we proceed on to obtaining and installing 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 pick HOD from there, as described in the point 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.

5. Configuring HOD

After HOD installation is done, it has to be configured before we start using it.

5.1. Minimal Configuration to get started

- On the node from where you want to run hod, edit the file hodrc which can be found in the <install dir>/conf directory. This file contains the minimal set of values required for running 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 1.5.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 submiting 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 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.

5.2. Advanced Configuration

You can review other configuration options in the file and modify them to suit your needs. Refer to the Configuration Guide for information about the HOD configuration.

6. Running HOD

You can now proceed to HOD User Guide for information about how to run HOD, what are the various features, options and for help in trouble-shooting.

7. Supporting Tools and Utilities

This section describes certain supporting tools and utilities that can be used in managing HOD deployments.

7.1. logcondense.py - Tool for removing log files uploaded to DFS

As mentioned in this section of the HOD User Guide, HOD can be configured to upload

Hadoop logs to a statically configured HDFS. Over time, the number of logs uploaded to DFS could increase. logcondense.py is a tool that helps administrators to clean-up the log files older than a certain number of days.

7.1.1. Running logcondense.py

logcondense.py is available under hod_install_location/support folder. You can either run it using python, for e.g. *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 DFS, if permissions are enabled. For e.g. as mentioned in the configuration guide, 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.

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	/user

		log-destination-uri, as mentioned in the configuration guide, without the hdfs:// URI string	
-n	--dynamicdfs	If true, this will indicate that the logcondense.py script should delete HDFS logs in addition to Map/Reduce logs. Otherwise, it only deletes Map/Reduce logs, which is also the default if this option is not specified. This option is useful if dynamic DFS installations are being provisioned by HOD, and the static DFS installation is being used only to collect logs - a scenario that may be common in test clusters.	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

7.2. checklimits.sh - Tool to update torque comment field reflecting resource limits

checklimits is a HOD tool specific to Torque/Maui environment (Maui Cluster Scheduler is an open source job scheduler for clusters and supercomputers, from clusterresources). The checklimits.sh script updates torque comment field when newly submitted job(s) violate/cross over user limits set up in Maui scheduler. It uses qstat, does one pass over torque job list to find out 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.

7.2.1. Running checklimits.sh

checklimits.sh is available under hod_install_location/support folder. This is a shell script and 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. In order for this tool to be able to update comment field of jobs from different users, it has to be run with torque administrative privileges. This tool has to be run repeatedly after specific intervals of time to frequently update jobs violating constraints, for e.g. 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.