Bright Cluster Manager 7.0

Hadoop Deployment Manual

Revision: c78c0f7

Date: Fri Sep 13 2024

Trademarks

Linux is a registered trademark of Linus Torvalds. PathScale is a registered trademark of Cray, Inc. Red Hat and all Red Hat-based trademarks are trademarks or registered trademarks of Red Hat, Inc. SUSE is a registered trademark of Novell, Inc. PGI is a registered trademark of The Portland Group Compiler Technology, STMicroelectronics, Inc. SGE is a trademark of Sun Microsystems, Inc. FLEXlm is a registered trademark of Globetrotter Software, Inc. Maui Cluster Scheduler is a trademark of Adaptive Computing, Inc. ScaleMP is a registered trademark of ScaleMP, Inc. All other trademarks are the property of their respective owners.

Rights and Restrictions

All statements, specifications, recommendations, and technical information contained herein are current or planned as of the date of publication of this document. They are reliable as of the time of this writing and are presented without warranty of any kind, expressed or implied. Bright Computing, Inc. shall not be liable for technical or editorial errors or omissions which may occur in this document. Bright Computing, Inc. shall not be liable for any damages resulting from the use of this document.

Limitation of Liability and Damages Pertaining to Bright Computing, Inc.

The Bright Cluster Manager product principally consists of free software that is licensed by the Linux authors free of charge. Bright Computing, Inc. shall have no liability nor will Bright Computing, Inc. provide any warranty for the Bright Cluster Manager to the extent that is permitted by law. Unless confirmed in writing, the Linux authors and/or third parties provide the program as is without any warranty, either expressed or implied, including, but not limited to, marketability or suitability for a specific purpose. The user of the Bright Cluster Manager product shall accept the full risk for the quality or performance of the product. Should the product malfunction, the costs for repair, service, or correction will be borne by the user of the Bright Cluster Manager product. No copyright owner or third party who has modified or distributed the program as permitted in this license shall be held liable for damages, including general or specific damages, damages caused by side effects or consequential damages, resulting from the use of the program or the un-usability of the program (including, but not limited to, loss of data, incorrect processing of data, losses that must be borne by you or others, or the inability of the program to work together with any other program), even if a copyright owner or third party had been advised about the possibility of such damages unless such copyright owner or third party has signed a writing to the contrary.

Table of Contents

Preface

Welcome to the *Hadoop Deployment Manual* for Bright Cluster Manager 7.0.

0.1 About This Manual

This manual is aimed at helping cluster administrators install, understand, configure, and manage the Hadoop capabilities of Bright Cluster Manager. The administrator is expected to be reasonably familiar with the Bright Cluster Manager *Administrator Manual*.

0.2 About The Manuals In General

Regularly updated versions of the Bright Cluster Manager 7.0 manuals are available on updated clusters by default at `/cm/shared/docs/cm`. The latest updates are always online at `http://support.brightcomputing.com/manuals`.

- The *Installation Manual* describes installation procedures for a basic cluster.
- The *Administrator Manual* describes the general management of the cluster.
- The *User Manual* describes the user environment and how to submit jobs for the end user.
- The *Cloudbursting Manual* describes how to deploy the cloud capabilities of the cluster.
- The *Developer Manual* has useful information for developers who would like to program with Bright Cluster Manager.
- The *OpenStack Deployment Manual* describes how to deploy OpenStack with Bright Cluster Manager.
- The *Hadoop Deployment Manual* describes how to deploy Hadoop with Bright Cluster Manager.
- The *UCS Deployment Manual* describes how to deploy the Cisco UCS server with Bright Cluster Manager.

If the manuals are downloaded and kept in one local directory, then in most pdf viewers, clicking on a cross-reference in one manual that refers to a section in another manual opens and displays that section in the second manual. Navigating back and forth between documents is usually possible with keystrokes or mouse clicks.

For example: `<Alt>-<Backarrow>` in Acrobat Reader, or clicking on the bottom leftmost navigation button of xpdf, both navigate back to the previous document.

The manuals constantly evolve to keep up with the development of the Bright Cluster Manager environment and the addition of new hardware and/or applications. The manuals also regularly incorporate customer feedback. Administrator and user input is greatly valued at Bright Computing. So any comments, suggestions or corrections will be very gratefully accepted at `manuals@brightcomputing.com`.

0.3 Getting Administrator-Level Support

Unless the Bright Cluster Manager reseller offers support, support is provided by Bright Computing over e-mail via `support@brightcomputing.com`. Section 10.2 of the *Administrator Manual* has more details on working with support.

1

Introduction

1.1 What Is Hadoop About?

Hadoop is the core implementation of a technology mostly used in the analysis of data sets that are large and unstructured in comparison with the data stored in regular relational databases. The analysis of such data, called *data-mining*, can be done better with Hadoop than with relational databases, for certain types of parallelizable problems. Hadoop has the following characteristics in comparison with relational databases:

1. **Less structured input:** Key value pairs are used as records for the data sets instead of a database.
2. **Scale-out rather than scale-up design:** For large data sets, if the size of a parallelizable problem increases linearly, the corresponding cost of scaling up a single machine to solve it tends to grow exponentially, simply because the hardware requirements tend to get exponentially expensive. If, however, the system that solves it is a cluster, then the corresponding cost tends to grow linearly because it can be solved by scaling out the cluster with a linear increase in the number of processing nodes.

   Scaling out can be done, with some effort, for database problems, using a parallel relational database implementation. However scale-out is inherent in Hadoop, and therefore often easier to implement with Hadoop. The Hadoop scale-out approach is based on the following design:

   - **Clustered storage:** Instead of a single node with a special, large, storage device, a distributed filesystem (HDFS) using commodity hardware devices across many nodes stores the data.
   - **Clustered processing:** Instead of using a single node with many processors, the parallel processing needs of the problem are distributed out over many nodes. The procedure is called the *MapReduce* algorithm, and is based on the following approach:
     - The distribution process "maps" the initial state of the problem into processes out to the nodes, ready to be handled in parallel.
     - Processing tasks are carried out on the data at nodes themselves.

- The results are "reduced" back to one result.

3. **Automated failure handling at application level for data:** Replication of the data takes place across the *DataNodes*, which are the nodes holding the data. If a DataNode has failed, then another node which has the replicated data on it is used instead automatically. Hadoop switches over quickly in comparison to replicated database clusters due to not having to check database table consistency.

1.2 Available Hadoop Implementations

Bright Cluster Manager supports the Hadoop implementations provided by the following organizations:

1. Apache (`http://apache.org`): This is the upstream source for the Hadoop core and some related components which all the other implementations use.
2. Cloudera (`http://www.cloudera.com`): Cloudera provides some extra premium functionality and components on top of a Hadoop suite. One of the extra components that Cloudera provides is the Cloudera Management Suite, a major proprietary management layer, with some premium features.
3. Hortonworks (`http://hortonworks.com`): Hortonworks Data Platform (HDP) is a fully open source Hadoop suite.
4. Pivotal HD: Pivotal Hadoop Distribution is a completely Apache-compliant distribution with extensive analytic toolsets.

The ISO image for Bright Cluster Manager, available at `http://www.brightcomputing.com/Download`, can include Hadoop for all 4 implementations. During installation from the ISO, the administrator can choose which implementation to install (section 3.3.14 of the *Installation Manual*).

The contents and versions of the Hadoop implementations provided by Bright Computing at the time of writing (April 2015) are as follows:

- **Apache**, packaged as `cm-apache-hadoop` by Bright Computing. This provides:
  - Hadoop versions 1.2.1, 2.2.0, and 2.6.0
  - HBase version 0.98.11 for Hadoop 1.2.1
  - HBase version 1.0.0 for Hadoop 2.2.0 and Hadoop 2.6.0
  - ZooKeeper version 3.4.6
- **Cloudera**, packaged as `cm-cloudera-hadoop` by Bright Computing. This provides:
  - CDH 4.7.1 (based on Apache Hadoop 2.0, HBase 0.94.15, and ZooKeeper 3.4.5)
  - CDH 5.3.2 (based on Apache Hadoop 2.5.0, HBase 0.98.6, and ZooKeeper 3.4.5)

- **Hortonworks**, packaged as `cm-hortonworks-hadoop` by Bright Computing. This provides:
  - HDP 1.3.9 (based on Apache Hadoop 1.2.0, HBase 0.94.6, and ZooKeeper 3.4.5)
  - HDP 2.2.0 (based on Apache Hadoop 2.6.0, HBase 0.98.4, and ZooKeeper 3.4.6)

Pivotal

Pivotal HD, version 2.1.0, is based on Apache Hadoop 2.2.0. It runs fully integrated on Bright Cluster Manager but is not available as a package in the Bright Cluster Manager repositories. This is because Pivotal prefers to distribute the software itself. The user can download the software from `https://network.pivotal.io/products/pivotal-hd` after registering with Pivotal.

1.3 Further Documentation

Further documentation is provided in the installed tarballs of the Hadoop version, after the Bright Cluster Manager installation (Chapter 2) has been carried out. A starting point is listed in the table below:

Hadoop version	**Path under** `/cm/shared/apps/hadoop`
Apache 1.2.1	`Apache/1.2.1/docs/index.html`
Apache 2.2.0	`Apache/2.2.0/share/doc/hadoop/index.html`
Apache 2.6.0	`Apache/2.6.0/share/doc/hadoop/index.html`
Cloudera CDH 4.7.1	`Cloudera/2.0.0-cdh4.7.1/share/doc/index.html`
Cloudera CDH 5.3.2	`Cloudera/2.5.0-cdh5.3.2/share/doc/index.html`
Hortonworks HDP	*Online documentation is available at* `http://docs.hortonworks.com/`

2

Installing Hadoop

In Bright Cluster Manager, a Hadoop instance can be configured and run either via the command-line (section 2.1) or via an ncurses GUI (section 2.2). Both options can be carried out with the `cm-hadoop-setup` script, which is run from a head node. The script is a part of the `cluster-tools` package, and uses tarballs from the Apache Hadoop project. The Bright Cluster Manager With Hadoop installation ISO includes the `cm-apache-hadoop` package, which contains tarballs from the Apache Hadoop project suitable for `cm-hadoop-setup`.

2.1 Command-line Installation Of Hadoop Using `cm-hadoop-setup -c <filename>`

2.1.1 Usage

```
[root@bright70 ~]# cm-hadoop-setup -h

  USAGE: /cm/local/apps/cluster-tools/bin/cm-hadoop-setup [-c <filename>
         | -u <name> | -h]

  OPTIONS:
  -c <filename>  --  Hadoop config file to use
  -u <name>      --  uninstall Hadoop instance
  -h             --  show usage

  EXAMPLES:
  cm-hadoop-setup -c /tmp/config.xml
  cm-hadoop-setup -u foo
  cm-hadoop-setup    (no options, a gui will be started)

  Some sample configuration files are provided in the directory
  /cm/local/apps/cluster-tools/hadoop/conf/:

    hadoop1conf.xml         (for Hadoop 1.x)
    hadoop2conf.xml         (for Hadoop 2.x)
    hadoop2fedconf.xml      (for Hadoop 2.x with NameNode federation)
    hadoop2haconf.xml       (for Hadoop 2.x with High Availability)
    hadoop2lustreconf.xml   (for Hadoop 2.x with Lustre support)
```

2.1.2 An Install Run

An XML template can be used based on the examples in the directory `/cm/local/apps/cluster-tools/hadoop/conf/`.

In the XML template, the path for a tarball component is enclosed by `<archive> </archive>` tag pairs. The tarball components can be as indicated:

- `<archive>`*hadoop tarball*`</archive>`
- `<archive>`*hbase tarball*`</archive>`
- `<archive>`*zookeeper tarball*`</archive>`

The tarball components can be picked up from URLs as listed in section 1.2. The paths of the tarball component files that are to be used should be set up as needed before running `cm-hadoop-setup`.

The downloaded tarball components should be placed in the `/tmp/` directory if the default definitions in the default XML files are used:

Example

```
[root@bright70 ~]# cd /cm/local/apps/cluster-tools/hadoop/conf
[root@bright70 conf]# grep 'archive>' hadoop1conf.xml | grep -o /.*.gz
/tmp/hadoop-1.2.1.tar.gz
/tmp/zookeeper-3.4.6.tar.gz
/tmp/hbase-0.98.11-hadoop1-bin.tar.gz
```

Files under `/tmp` are not intended to stay around permanently. The administrator may therefore wish to place the tarball components in a more permanent part of the filesystem instead, and change the XML definitions accordingly.

A Hadoop instance name, for example `Myhadoop`, can also be defined in the XML file, within the `<name></name>` tag pair.

Hadoop NameNodes and SecondaryNameNodes handle HDFS metadata, while DataNodes manage HDFS data. The data must be stored in the filesystem of the nodes. The default path for where the data is stored can be specified within the `<dataroot></dataroot>` tag pair. Multiple paths can also be set, using comma-separated paths. NameNodes, SecondaryNameNodes, and DataNodes each use the value, or values, set within the `<dataroot></dataroot>` tag pair for their root directories.

If needed, more specific tags can be used for each node type. This is useful in the case where hardware differs for the various node types. For example:

- a NameNode with 2 disk drives for Hadoop use
- a DataNode with 4 disk drives for Hadoop use

The XML file used by `cm-hadoop-setup` can in this case use the tag pairs:

- `<namenodedatadirs></namenodedatadirs>`
- `<datanodedatadirs></datanodedatadirs>`

If these are not specified, then the value within the `<dataroot></dataroot>` tag pair is used.

Example

- <namenodedatadirs>/data1,/data2</namenodedatadirs>
- <datanodedatadirs>/data1,/data2,/data3,/data4</datanodedatadirs>

Hadoop should then have the following dfs.*.name.dir properties added to it via the hdfs-site.xml configuration file. For the preceding tag pairs, the property values should be set as follows:

Example

- dfs.namenode.name.dir with values:
  /data1/hadoop/hdfs/namenode,/data2/hadoop/hdfs/namenode
- dfs.datanode.name.dir with values:
  /data1/hadoop/hdfs/datanode,/data2/hadoop/hdfs/datanode,/data3/hadoop/hdfs/datanode,/data4/hadoop/hdfs/datanode

An install run then displays output like the following:

Example

```
-rw-r--r-- 1 root root 63851630 Feb  4 15:13 hadoop-1.2.1.tar.gz
[root@bright70 ~]# cm-hadoop-setup -c /tmp/hadoop1conf.xml
Reading config from file '/tmp/hadoop1conf.xml'... done.
Hadoop flavor 'Apache', release '1.2.1'
Will now install Hadoop in /cm/shared/apps/hadoop/Apache/1.2.1 and conf\
igure instance 'Myhadoop'
Hadoop distro being installed... done.
Zookeeper being installed... done.
HBase being installed... done.
Creating module file... done.
Configuring Hadoop instance on local filesystem and images... done.
Updating images:
starting imageupdate for node 'node003'... started.
starting imageupdate for node 'node002'... started.
starting imageupdate for node 'node001'... started.
starting imageupdate for node 'node004'... started.
Waiting for imageupdate to finish... done.
Creating Hadoop instance in cmdaemon... done.
Formatting HDFS... done.
Waiting for datanodes to come up... done.
Setting up HDFS... done.
```

The Hadoop instance should now be running. The name defined for it in the XML file should show up within cmgui, in the Hadoop HDFS resource tree folder (figure 2.1).

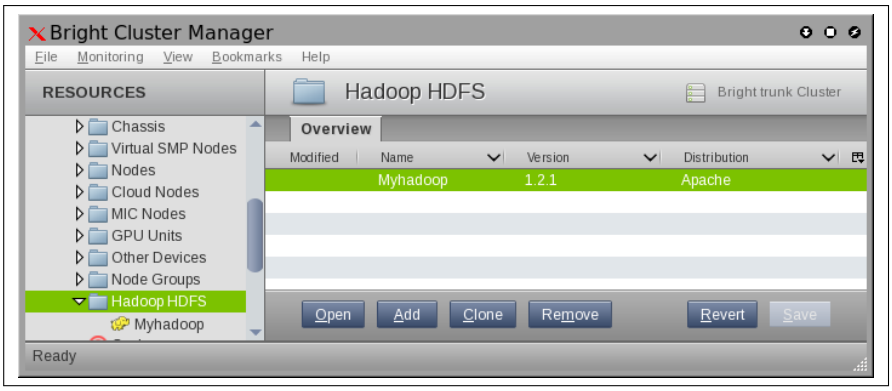

Figure 2.1: A Hadoop Instance In `cmgui`

The instance name is also displayed within `cmsh` when the `list` command is run in `hadoop` mode:

Example

```
[root@bright70 ~] cmsh
[bright70]% hadoop
[bright70->hadoop]% list
Name (key) Hadoop version Hadoop distribution Configuration directory
---------- -------------- ------------------- ----------------------
Myhadoop   1.2.1          Apache              /etc/hadoop/Myhadoop
```

The instance can be removed as follows:

Example

```
[root@bright70 ~]# cm-hadoop-setup -u Myhadoop
Requested uninstall of Hadoop instance 'Myhadoop'.
Uninstalling Hadoop instance 'Myhadoop'

Removing:
                /etc/hadoop/Myhadoop
                /var/lib/hadoop/Myhadoop
                /var/log/hadoop/Myhadoop
                /var/run/hadoop/Myhadoop
                /tmp/hadoop/Myhadoop/
                /etc/hadoop/Myhadoop/zookeeper
                /var/lib/zookeeper/Myhadoop
                /var/log/zookeeper/Myhadoop
                /var/run/zookeeper/Myhadoop
                /etc/hadoop/Myhadoop/hbase
                /var/log/hbase/Myhadoop
                /var/run/hbase/Myhadoop
                /etc/init.d/hadoop-Myhadoop-*
Module file(s) deleted.
Uninstall successfully completed.
```

2.2 Ncurses Installation Of Hadoop Using `cm-hadoop-setup`

Running `cm-hadoop-setup` without any options starts up an ncurses GUI (figure 2.2).

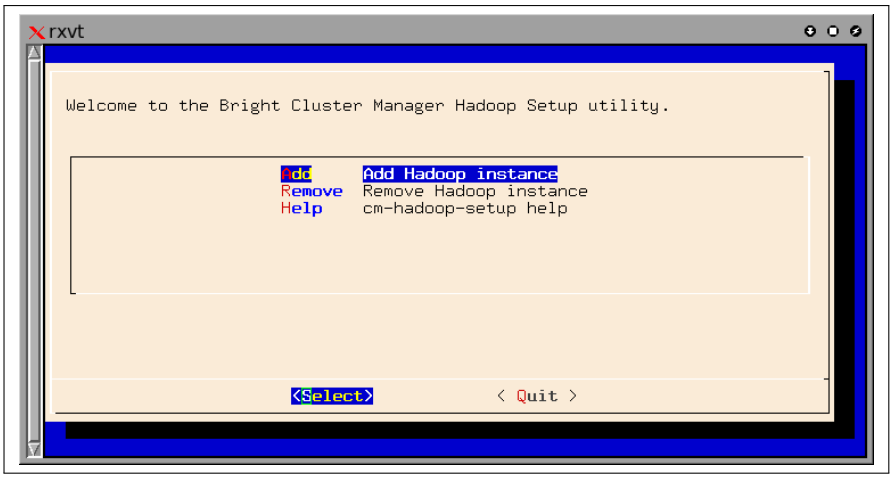

Figure 2.2: The `cm-hadoop-setup` Welcome Screen

This provides an interactive way to add and remove Hadoop instances, along with HBase and Zookeeper components. Some explanations of the items being configured are given along the way. In addition, some minor validation checks are done, and some options are restricted.

The suggested default values will work. Other values can be chosen instead of the defaults, but some care in selection usually a good idea. This is because Hadoop is a complex software, which means that values other than the defaults can sometimes lead to unworkable configurations (section 2.3).

The ncurses installation results in an XML configuration file. This file can be used with the `-c` option of `cm-hadoop-setup` to carry out the installation.

2.3 Avoiding Misconfigurations During Hadoop Installation

A misconfiguration can be defined as a configuration that works badly or not at all.

For Hadoop to work well, the following common misconfigurations should normally be avoided. Some of these result in warnings from Bright Cluster Manager validation checks during configuration, but can be overridden. An override is useful for cases where the administrator would just like to, for example, test some issues not related to scale or performance.

2.3.1 NameNode Configuration Choices

One of the following NameNode configuration options must be chosen when Hadoop is installed. The choice should be made with care, because changing between the options after installation is not possible.

Hadoop 1.x NameNode Configuration Choices

- NameNode can optionally have a SecondaryNameNode.

  SecondaryNameNode offloads metadata operations from NameNode, and also stores the metadata offline to some extent.

  It is not by any means a high availability solution. While recovery

from a failed head node is possible from SecondaryNameNode, it is not easy, and it is not recommended or supported by Bright Cluster Manager.

Hadoop 2.x NameNode Configuration Choices

- NameNode and SecondaryNameNode can run as in Hadoop 1.x.

However, the following configurations are also possible:

- **NameNode HA with manual failover:** In this configuration Hadoop has NameNode1 and NameNode2 up at the same time, with one active and one on standby. Which NameNode is active and which is on standby is set manually by the administrator. If one NameNode fails, then failover must be executed manually. Metadata changes are managed by ZooKeeper, which relies on a quorum of JournalNodes. The number of JournalNodes is therefore set to 3, 5, 7...
- **NameNode HA with automatic failover:** As for the manual case, except that in this case ZooKeeper manages failover too. Which NameNode is active and which is on standby is therefore decided automatically.
- **NameNode Federation:** In NameNode Fedaration, the storage of metadata is split among several NameNodes, each of which has a corresponding SecondaryNameNode. Each pair takes care of a part of HDFS.

  In Bright Cluster Manager there are 4 NameNodes in a default NameNode federation:

  - `/user`
  - `/tmp`
  - `/staging`
  - `/hbase`

  User applications do not have to know this mapping. This is because ViewFS on the client side maps the selected path to the corresponding NameNode. Thus, for example, `hdfs -ls /tmp/example` does not need to know that `/tmp` is managed by another NameNode.

  Cloudera advise against using NameNode Federation for production purposes at present, due to its development status.

2.4 Installing Hadoop With Lustre

The Lustre filesystem has a client-server configuration.

2.4.1 Lustre Internal Server Installation

The procedure for installing a Lustre server varies.

The Bright Cluster Manager Knowledge Base article describes a procedure that uses Lustre sources from Whamcloud. The article is at `http://kb.brightcomputing.com/faq/index.php?action=artikel&cat=9&id=176`, and may be used as guidance.

2.4.2 Lustre External Server Installation

Lustre can also be configured so that the servers run external to Bright Cluster Manager. The Lustre Intel IEEL 2.x version can be configured in this manner.

2.4.3 Lustre Client Installation

It is preferred that the Lustre clients are installed on the head node as well as on all the nodes that are to be Hadoop nodes. The clients should be configured to provide a Lustre mount on the nodes. If the Lustre client cannot be installed on the head node, then Bright Cluster Manager has the following limitations during installation and maintenance:

- the head node cannot be used to run Hadoop services
- end users cannot perform Hadoop operations, such as job submission, on the head node. Operations such as those should instead be carried out while logged in to one of the Hadoop nodes

In the remainder of this section, a Lustre mount point of `/mnt/lustre` is assumed, but it can be set to any convenient directory mount point.

The user IDs and group IDs of the Lustre server and clients should be consistent. It is quite likely that they differ when first set up. The IDs should be checked at least for the following users and groups:

- **users:** `hdfs`, `mapred`, `yarn`, `hbase`, `zookeeper`
- **groups:** `hadoop`, `zookeeper`, `hbase`

If they do not match on the server and clients, then they must be made consistent manually, so that the UID and GID of the Lustre server users are changed to match the UID and GID of the Bright Cluster Manager users.

Once consistency has been checked, and read/write access is working to LustreFS, the Hadoop integration can be configured.

2.4.4 Lustre Hadoop Configuration

Lustre Hadoop XML Configuration File Setup

Hadoop integration can be configured by using the file `/cm/local/apps/cluster-tools/hadoop/conf/hadoop2lustreconf.xml` as a starting point for the configuration. It can be copied over to, for example, `/root/hadooplustreconfig.xml`.

The Intel Distribution for Hadoop (IDH) and Cloudera can both run with Lustre under Bright Cluster Manager. The configuration for these can be done as follows:

- IDH
  - A subdirectory of `/mnt/lustre` must be specified in the `hadoop2lustreconf.xml` file within the `<afs></afs>` tag pair:

    Example

    ```
    <afs>
      <fstype>lustre</fstype>
      <fsroot>/mnt/lustre/hadoop</fsroot>
    ```

```
</afs>
```

- Cloudera
  - A subdirectory of `/mnt/lustre` must be specified in the `hadoop2lustreconf.xml` file within the `<afs></afs>` tag pair:
  - In addition, an `<fsjar></fsjar>` tag pair must be specified manually for the jar that the Intel IEEL 2.x distribution provides:

    Example

    ```
    <afs>
      <fstype>lustre</fstype>
      <fsroot>/mnt/lustre/hadoop</fsroot>
      <fsjar>/root/lustre/hadoop-lustre-plugin-2.3.0.jar</fsjar>
    </afs>
    ```

    The installation of the Lustre plugin is automatic if this jar name is set to the right name, when the `cm-hadoop-setup` script is run.

Lustre Hadoop Installation With `cm-hadoop-setup`

The XML configuration file specifies how Lustre should be integrated in Hadoop. If the configuration file is at `</root/hadooplustreconfig.xml>`, then it can be run as:

Example

```
cm-hadoop-setup -c </root/hadooplustreconfig.xml>
```

As part of configuring Hadoop to use Lustre, the execution will:

- Set the ACLs on the directory specified within the `<fsroot><fsroot>` tag pair. This was set to `/mnt/lustre/hadoop` earlier on as an example.
- Copy the Lustre plugin from its jar path as specified in the XML file, to the correct place on the client nodes.

  Specifically, the subdirectory `./share/hadoop/common/lib` is copied into a directory relative to the Hadoop installation directory. For example, the Cloudera version of Hadoop, version 2.30-cdh5.1.2, has the Hadoop installation directory `/cm/share/apps/hadoop/Clouder/2.3.0-cdh5.1.2`. The copy is therefore carried out in this case from:
  `/root/lustre/hadoop-lustre-plugin-2.3.0.jar`
  to
  `/cm/shared/apps/hadoop/Cloudera/2.3.0-cdh5.1.2/share/hadoop/common/lib`

Lustre Hadoop Integration In `cmsh` and `cmgui`
In `cmsh`, Lustre integration is indicated in `hadoop` mode:

Example

```
[hadoop2->hadoop]%  show hdfs1 | grep -i lustre
Hadoop root for Lustre                  /mnt/lustre/hadoop
Use Lustre                              yes
```

In `cmgui`, the `Overview` tab in the items for the `Hadoop HDFS` resource indicates if Lustre is running, along with its mount point (figure 2.3).

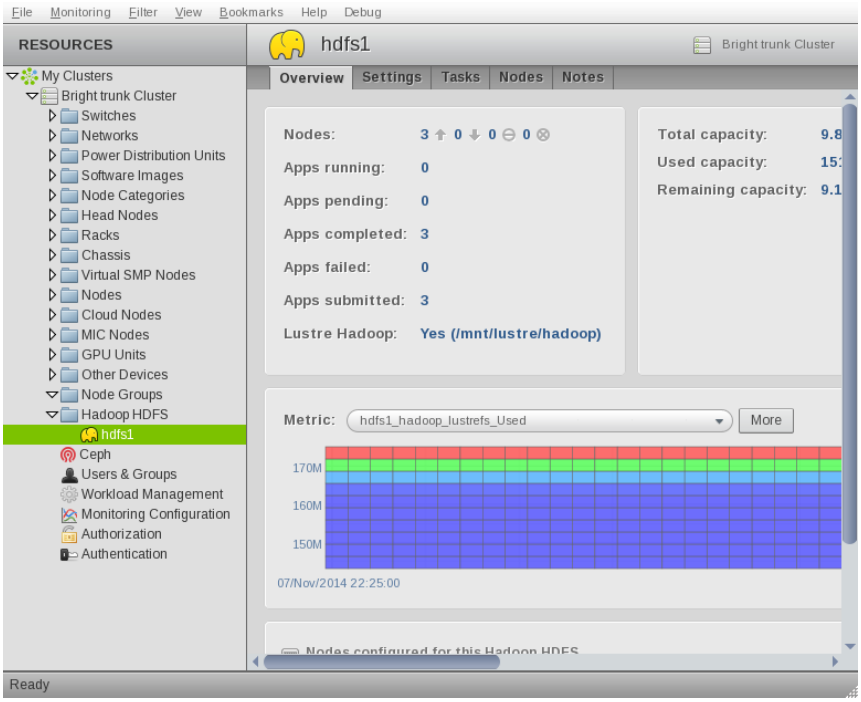

Figure 2.3: A Hadoop Instance With Lustre In `cmgui`

2.5 Hadoop Installation In A Cloud

Hadoop can make use of cloud services so that it runs as a Cluster-On-Demand configuration (Chapter 2 of the *Cloudbursting Manual*), or a Cluster Extension configuration (Chapter 3 of the *Cloudbursting Manual*). In both cases the cloud nodes used should be at least `m1.medium`.

- For Cluster-On-Demand the following considerations apply:
  - There are no specific issues. After a stop/start cycle Hadoop recognizes the new IP addresses, and refreshes the list of nodes accordingly (section 2.4.1 of the *Cloudbursting Manual*).
- For Cluster Extension the following considerations apply:
  - To install Hadoop on cloud nodes, the XML configuration:
    `/cm/local/apps/cluster-tools/hadoop/conf/hadoop2clusterextensionconf.xml`
    can be used as a guide.

– In the `hadoop2clusterextensionconf.xml` file, the cloud director that is to be used with the Hadoop cloud nodes must be specified by the administrator with the `<edge></edge>` tag pair:

Example

```
<edge>
  <hosts>eu-west-1-director</hosts>
</edge>
```

Maintenance operations, such as a format, will automatically and transparently be carried out by `cmdaemon` running on the cloud director, and not on the head node.

There are some shortcomings as a result of relying upon the cloud director:

– Cloud nodes depend on the same cloud director
– While Hadoop installation (`cm-hadoop-setup`) is run on the head node, users must run Hadoop commands—job submissions, and so on—from the director, not from the head node.
– It is not possible to mix cloud and non-cloud nodes for the same Hadoop instance. That is, a local Hadoop instance cannot be extended by adding cloud nodes.

3

Viewing And Managing HDFS From CMDaemon

3.1 `cmgui` And The HDFS Resource

In `cmgui`, clicking on the resource folder for Hadoop in the resource tree opens up an `Overview` tab that displays a list of Hadoop HDFS instances, as in figure 2.1.

Clicking on an instance brings up a tab from a series of tabs associated with that instance, as in figure 3.1.

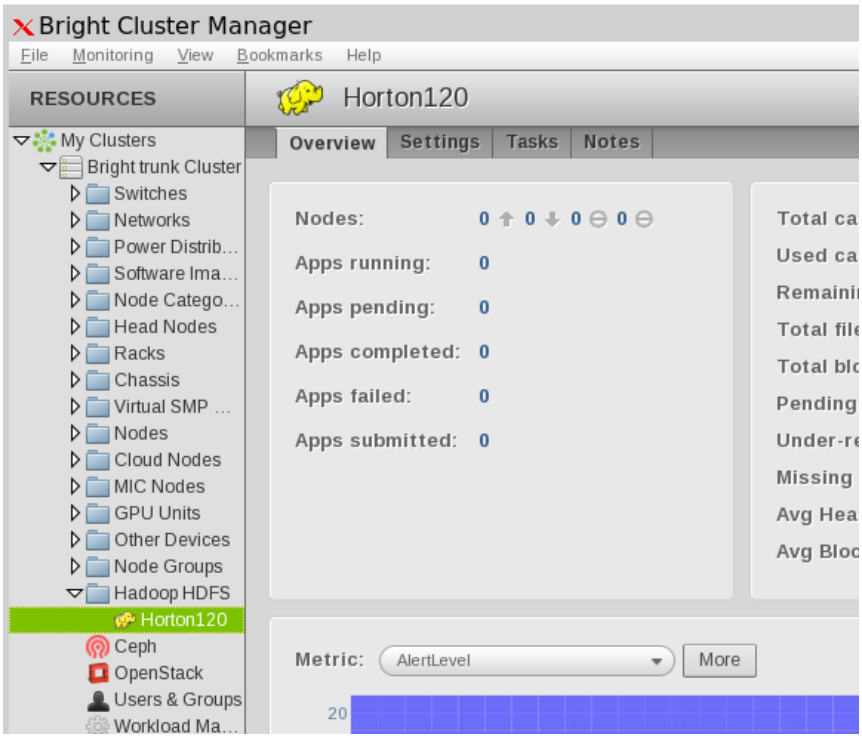

Figure 3.1: Tabs For A Hadoop Instance In `cmgui`

The possible tabs are `Overview` (section 3.1.1), `Settings` (section 3.1.2), `Tasks` (section 3.1.3), and `Notes` (section 3.1.4). Various sub-panes are displayed in the possible tabs.

3.1.1 The HDFS Instance `Overview` Tab

The `Overview` tab pane (figure 3.2) arranges Hadoop-related items in sub-panes for convenient viewing.

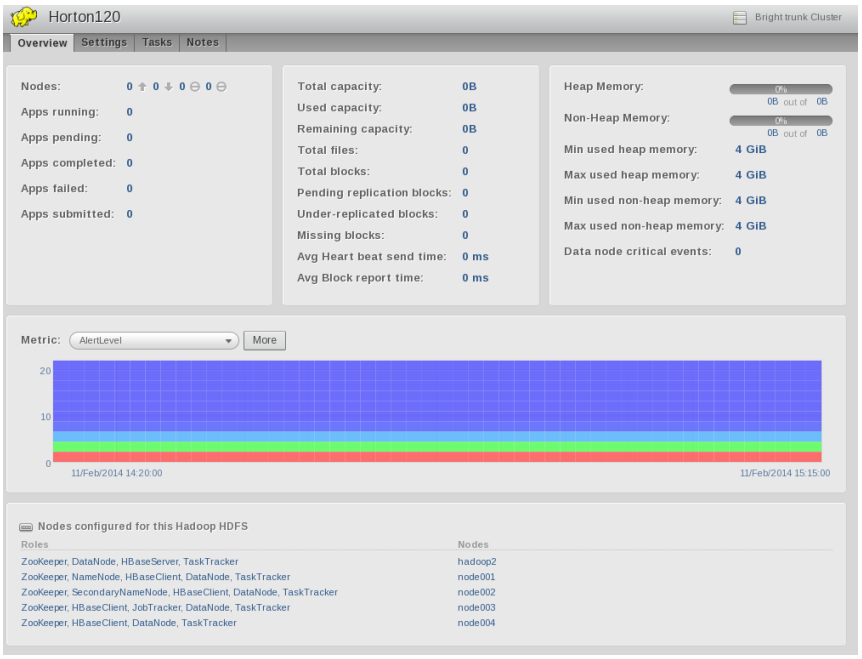

Figure 3.2: Overview Tab View For A Hadoop Instance In `cmgui`

The following items can be viewed:

- **Statistics:** In the first sub-pane row are statistics associated with the Hadoop instance. These include node data on the number of applications that are running, as well as memory and capacity usage.
- **Metrics:** In the second sub-pane row, a metric can be selected for display as a graph.
- **Roles:** In the third sub-pane row, the roles associated with a node are displayed.

3.1.2 The HDFS Instance `Settings` Tab

The `Settings` tab pane (figure 3.3) displays the configuration of the Hadoop instance.

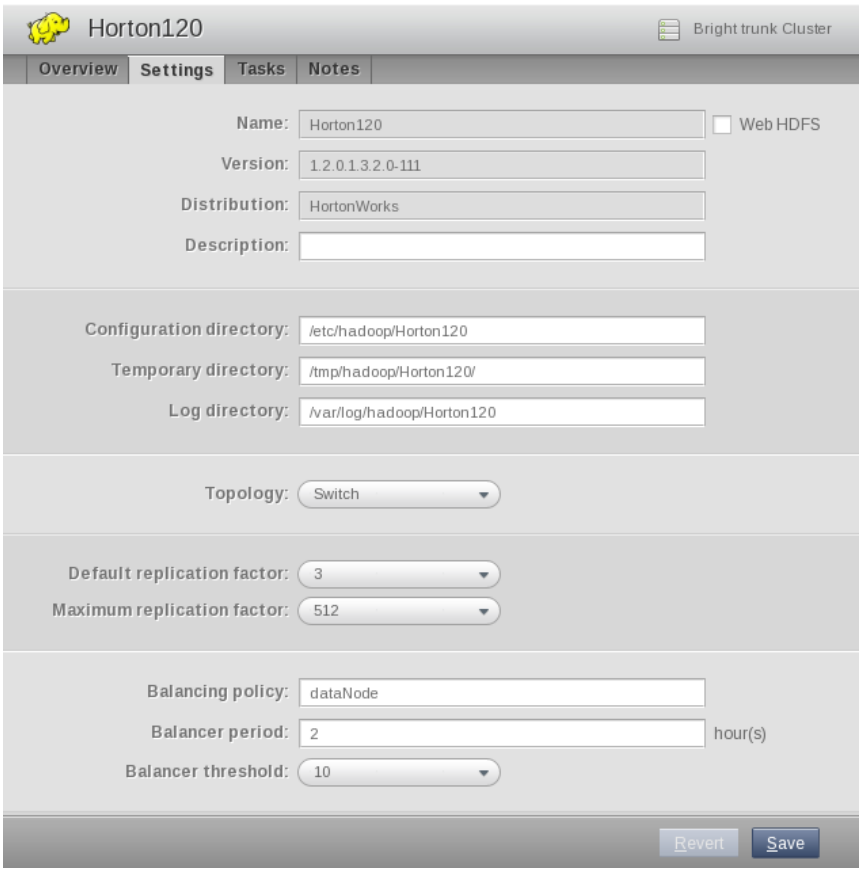

Figure 3.3: Settings Tab View For A Hadoop Instance In cmgui

It allows the following Hadoop-related settings to be changed:

- **WebHDFS:** Allows HDFS to be modified via the web.
- **Description:** An arbitrary, user-defined description.
- **Directory settings for the Hadoop instance:** Directories for configuration, temporary purposes, and logging.
- **Topology awareness setting:** Optimizes Hadoop for racks, switches or nothing, when it comes to network topology.
- **Replication factors:** Set the default and maximum replication allowed.
- **Balancing settings:** Spread loads evenly across the instance.

3.1.3 The HDFS Instance Tasks Tab

The Tasks tab (figure 3.4) displays Hadoop-related tasks that the administrator can execute.

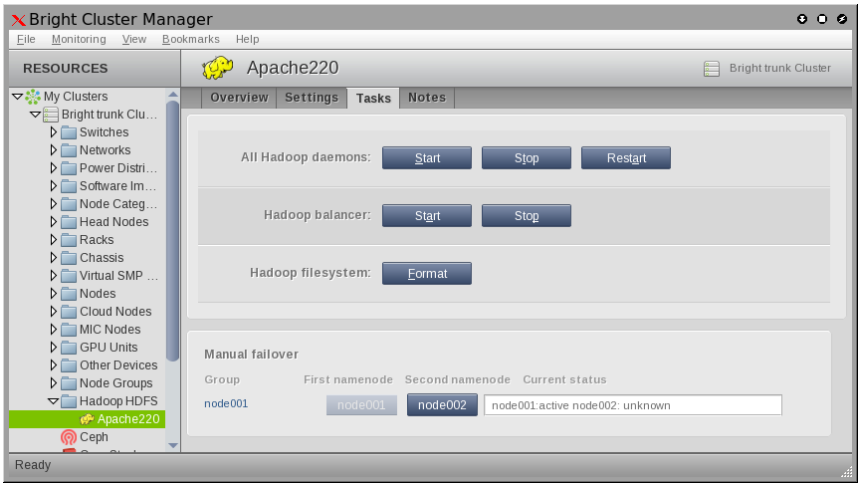

Figure 3.4: Tasks Tab View For A Hadoop Instance In `cmgui`

The `Tasks` tab allows the administrator to:

- **Restart, start, or stop all the Hadoop daemons.**
- **Start/Stop the Hadoop balancer.**
- **Format the Hadoop filesystem.** A warning is given before formatting.
- **Carry out a manual failover for the NameNode.** This feature means making the other NameNode the active one, and is available only for Hadoop instances that support NameNode failover.

3.1.4 The HDFS Instance `Notes` Tab

The `Notes` tab is like a jotting pad, and can be used by the administrator to write notes for the associated Hadoop instance.

3.2 `cmsh` And `hadoop` Mode

The `cmsh` front end can be used instead of `cmgui` to display information on Hadoop-related values, and to carry out Hadoop-related tasks.

3.2.1 The `show` And `overview` Commands

The `show` Command

Within `hadoop` mode, the `show` command displays parameters that correspond mostly to `cmgui`'s `Settings` tab in the Hadoop resource (section 3.1.2):

Example

```
[root@bright70 conf]# cmsh
[bright70]% hadoop
[bright70->hadoop]% list
Name (key) Hadoop version Hadoop distribution Configuration directory
---------- -------------- -------------------- ------------------------
Apache121  1.2.1          Apache               /etc/hadoop/Apache121
[bright70->hadoop]% show apache121
Parameter                        Value
```

```
----------------------------------- -----------------------------------
Automatic failover                  Disabled
Balancer Threshold                  10
Balancer period                     2
Balancing policy                    dataNode
Cluster ID
Configuration directory             /etc/hadoop/Apache121
Configuration directory for HBase   /etc/hadoop/Apache121/hbase
Configuration directory for ZooKeeper /etc/hadoop/Apache121/zookeeper
Creation time                       Fri, 07 Feb 2014 17:03:01 CET
Default replication factor          3
Enable HA for YARN                  no
Enable NFS gateway                  no
Enable Web HDFS                     no
HA enabled                          no
HA name service
HDFS Block size                     67108864
HDFS Permission                     no
HDFS Umask                          077
HDFS log directory                  /var/log/hadoop/Apache121
Hadoop distribution                 Apache
Hadoop root
Hadoop version                      1.2.1
Installation directory for HBase    /cm/shared/apps/hadoop/Apache/hbase-0.+
Installation directory for ZooKeeper  /cm/shared/apps/hadoop/Apache/zookeepe+
Installation directory              /cm/shared/apps/hadoop/Apache/1.2.1
Maximum replication factor          512
Network
Revision
Root directory for data             /var/lib/
Temporary directory                 /tmp/hadoop/Apache121/
Topology                            Switch
Use HTTPS                           no
Use Lustre                          no
Use federation                      no
Use only HTTPS                      no
YARN automatic failover             Hadoop
description                         installed from: /tmp/hadoop-1.2.1.tar.+
name                                Apache121
notes
```

The `overview` Command

Similarly, the `overview` command corresponds somewhat to the `cmgui`'s `Overview` tab in the Hadoop resource (section 3.1.1), and provides hadoop-related information on the system resources that are used:

Example

```
[bright70->hadoop]% overview apache121
Parameter                    Value
---------------------------  --------------------------
Name                         apache121
Capacity total               0B
Capacity used                0B
Capacity remaining           0B
Heap memory total            0B
```

```
Heap memory used                0B
Heap memory remaining           0B
Non-heap memory total           0B
Non-heap memory used            0B
Non-heap memory remaining       0B
Nodes available                 0
Nodes dead                      0
Nodes decommissioned            0
Nodes decommission in progress  0
Total files                     0
Total blocks                    0
Missing blocks                  0
Under-replicated blocks         0
Scheduled replication blocks    0
Pending replication blocks      0
Block report average Time       0
Applications running            0
Applications pending            0
Applications submitted          0
Applications completed          0
Applications failed             0
Federation setup                no

Role                            Node
------------------------------ --------------------------
DataNode, ZooKeeper             bright70,node001,node002
```

3.2.2 The Tasks Commands

Within `hadoop` mode, the following commands run tasks that correspond mostly to the `Tasks` tab (section 3.1.3) in the `cmgui` Hadoop resource:

The `*services` Commands For Hadoop Services

Hadoop services can be started, stopped, and restarted, with:

- `restartallservices`
- `startallservices`
- `stopallservices`

Example

```
[bright70->hadoop]% restartallservices apache121
Will now stop all Hadoop services for instance 'apache121'... done.
Will now start all Hadoop services for instance 'apache121'... done.
[bright70->hadoop]%
```

The `startbalancer` And `stopbalancer` Commands For Hadoop

For efficient access to HDFS, file block level usage across nodes should be reasonably balanced. Hadoop can start and stop balancing across the instance with the following commands:

- `startbalancer`
- `stopbalancer`

The balancer policy, threshold, and period can be retrieved or set for the instance using the parameters:

- balancerperiod
- balancerthreshold
- balancingpolicy

Example

```
[bright70->hadoop]% get apache121 balancerperiod
2
[bright70->hadoop]% set apache121 balancerperiod 3
[bright70->hadoop*]% commit
[bright70->hadoop]% startbalancer apache121
Code: 0
Starting Hadoop balancer daemon (hadoop-apache121-balancer):starting ba\
lancer, logging to /var/log/hadoop/apache121/hdfs/hadoop-hdfs-balancer-\
bright70.out
Time Stamp  Iteration# Bytes Moved  Bytes To Move  Bytes Being Moved
The cluster is balanced. Exiting...

[bright70->hadoop]%
Thu Mar 20 15:27:02 2014 [notice] bright70: Started balancer for apache\
121 For details type: events details 152727
```

The formathdfs Command

Usage:

formathdfs <*HDFS*>

The formathdfs command formats an instance so that it can be reused. Existing Hadoop services for the instance are stopped first before formatting HDFS, and started again after formatting is complete.

Example

```
[bright70->hadoop]% formathdfs apache121
Will now format and set up HDFS for instance 'apache121'.
Stopping datanodes... done.
Stopping namenodes... done.
Formatting HDFS... done.
Starting namenode (host 'bright70')... done.
Starting datanodes... done.
Waiting for datanodes to come up... done.
Setting up HDFS... done.
[bright70->hadoop]%
```

The manualfailover Command

Usage:

manualfailover [-f|--from <*NameNode*>] [-t|--to <*other NameNode*>] <*HDFS*>

The manualfailover command allows the active status of a NameNode to be moved to another NameNode in the Hadoop instance. This is only available for Hadoop instances that support NameNode failover. The active status can be move from, and/or to, a NameNode.

4

Hadoop Services

4.1 Hadoop Services From `cmgui`

4.1.1 Configuring Hadoop Services From `cmgui`

Hadoop services can be configured on head or regular nodes.

Configuration in `cmgui` can be carried out by selecting the node instance on which the service is to run, and selecting the Hadoop tab of that node. This then displays a list of possible Hadoop services within sub-panes (figure 4.1).

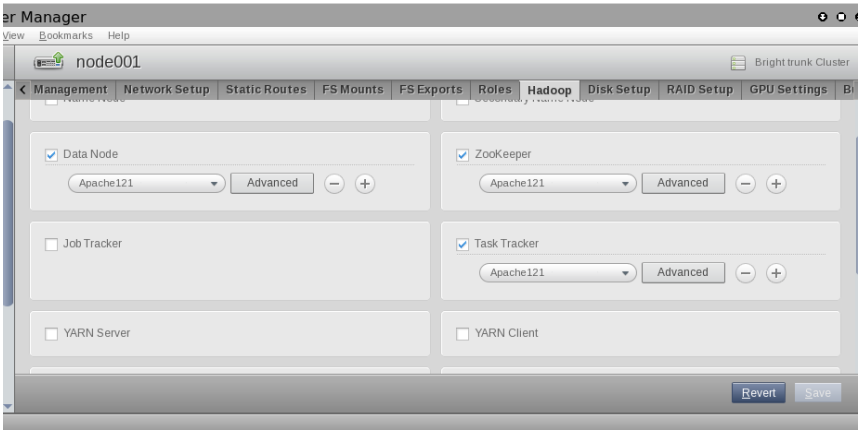

Figure 4.1: Services In The Hadoop Tab Of A Node In `cmgui`

The services displayed are:

- `Name Node`, `Secondary Name Node`
- `Data Node`, `Zookeeper`
- `Job Tracker`, `Task Tracker`
- `YARN Server`, `YARN Client`
- `HBase Server`, `HBase Client`
- `Journal`

For each Hadoop service, instance values can be edited via actions in the associated pane:

- **Selecting A Hadoop instance:** The name of the possible Hadoop instances can be selected via a drop-down menu.
- **Configuring multiple Hadoop instances:** More than one Hadoop instance can run on the cluster. Using the ⊕ button adds an instance.
- **Advanced options for instances:** The `Advanced` button allows many configuration options to be edited for each node instance for each service. This is illustrated by the example in figure 4.2, where the advanced configuration options of the Data Node service of the node001 instance are shown:

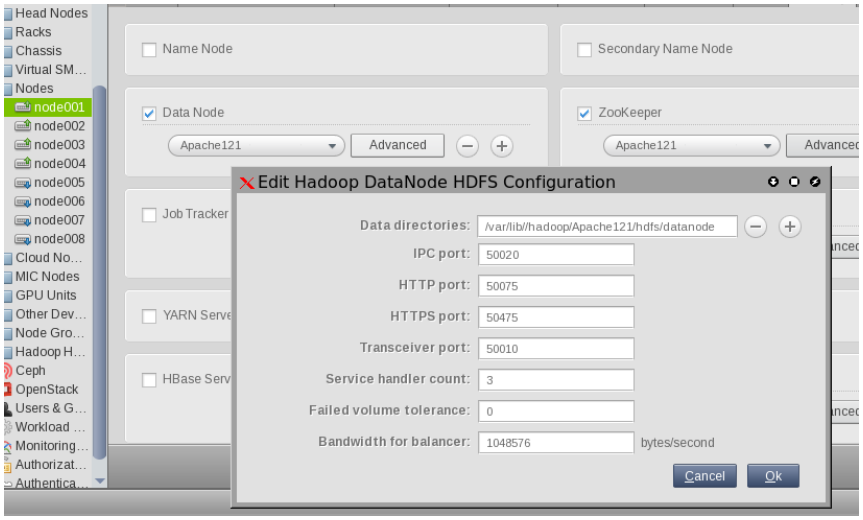

Figure 4.2: Advanced Configuration Options For The Data Node service In `cmgui`

Typically, care must be taken to use non-conflicting ports and directories for each Hadoop instance.

4.1.2 Monitoring And Modifying The Running Of Hadoop Services From `cmgui`

The status of Hadoop services can be viewed and changed in the following locations:

- **The `Services` tab of the node instance**: Within the `Nodes` or `Head Nodes` resource, for a head or regular node instance, the `Services` tab can be selected. The Hadoop services can then be viewed and altered from within the display pane (figure 4.3).

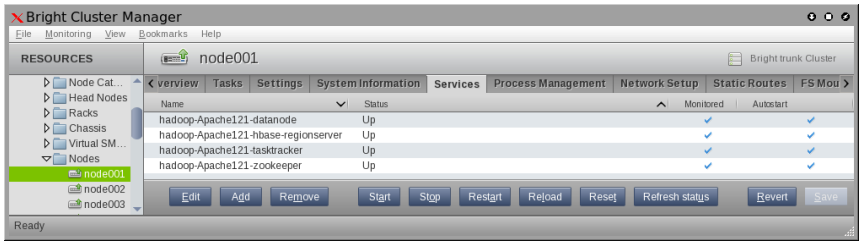

Figure 4.3: Services In The Services Tab Of A Node In `cmgui`

The options buttons act just like for any other service (section 3.11 of the *Administrator Manual*), which means it includes the possibility of acting on multiple service selections.

- **The Tasks tab of the Hadoop instance**: Within the Hadoop HDFS resource, for a specific Hadoop instance, the Tasks tab (section 3.1.3) conveniently allows all Hadoop daemons to be (re)started and stopped directly via buttons.

5

Running Hadoop Jobs

5.1 Shakedown Runs

The `cm-hadoop-tests.sh` script is provided as part of Bright Cluster Manager's `cluster-tools` package. The administrator can use the script to conveniently submit example jar files in the Hadoop installation to a job client of a Hadoop instance:

```
[root@bright70 ~]# cd /cm/local/apps/cluster-tools/hadoop/
[root@bright70 hadoop]#  ./cm-hadoop-tests.sh <instance>
```

The script runs endlessly, and runs several Hadoop test scripts. If most lines in the run output are elided for brevity, then the structure of the truncated output looks something like this in overview:

Example

```
[root@bright70 hadoop]# ./cm-hadoop-tests.sh apache220
...
================================================================
Press [CTRL+C] to stop...
================================================================
...
================================================================
start cleaning directories...
================================================================
...
================================================================
clean directories done
================================================================

================================================================
start doing gen_test...
================================================================
...
14/03/24 15:05:37 INFO terasort.TeraSort: Generating 10000 using 2
14/03/24 15:05:38 INFO mapreduce.JobSubmitter: number of splits:2
...
        Job Counters
                ...
        Map-Reduce Framework
                ...
```

```
        org.apache.hadoop.examples.terasort.TeraGen$Counters
...
14/03/24 15:07:03 INFO terasort.TeraSort: starting
...
14/03/24 15:09:12 INFO terasort.TeraSort: done
...
================================================================================
gen_test done
================================================================================

================================================================================
start doing PI test...
================================================================================
Working Directory = /user/root/bbp
...
```

During the run, the `Overview` tab in `cmgui` (introduced in section 3.1.1) for the Hadoop instance should show activity as it refreshes its overview every three minutes (figure 5.1):

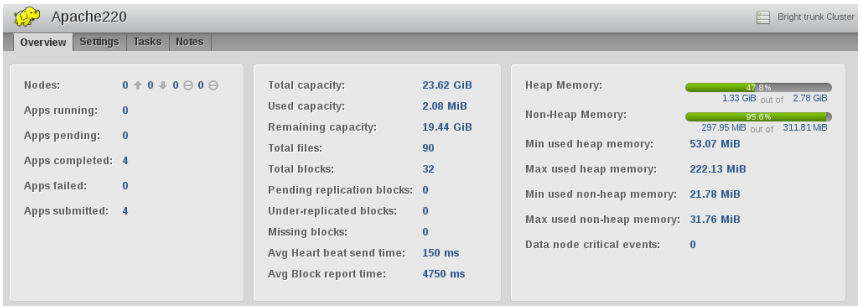

Figure 5.1: Overview Of Activity Seen For A Hadoop Instance In `cmgui`

In `cmsh` the `overview` command shows the most recent values that can be retrieved when the command is run:

```
[mk-hadoop-centos6->hadoop]% overview apache220
Parameter                      Value
------------------------------ --------------------------------
Name                           Apache220
Capacity total                 27.56GB
Capacity used                  7.246MB
Capacity remaining             16.41GB
Heap memory total              280.7MB
Heap memory used               152.1MB
Heap memory remaining          128.7MB
Non-heap memory total          258.1MB
Non-heap memory used           251.9MB
Non-heap memory remaining      6.155MB
Nodes available                3
Nodes dead                     0
Nodes decommissioned           0
Nodes decommission in progress  0
Total files                    72
Total blocks                   31
Missing blocks                 0
```

```
Under-replicated blocks        2
Scheduled replication blocks   0
Pending replication blocks     0
Block report average Time      59666
Applications running           1
Applications pending           0
Applications submitted         7
Applications completed         6
Applications failed            0
High availability              Yes (automatic failover disabled)
Federation setup               no

Role                                                              Node
----------------------------------------------------------------- -------
DataNode, Journal, NameNode, YARNClient, YARNServer, ZooKeeper node001
DataNode, Journal, NameNode, YARNClient, ZooKeeper             node002
```

5.2 Example End User Job Run

Running a job from a jar file individually can be done by an end user.

An end user `fred` can be created and issued a password by the administrator (Chapter 6 of the *Administrator Manual*). The user must then be granted HDFS access for the Hadoop instance by the administrator:

Example

```
[bright70->user[fred]]% set hadoophdfsaccess apache220; commit
```

The possible instance options are shown as tab-completion suggestions. The access can be unset by leaving a blank for the instance option.

The user `fred` can then submit a run from a pi value estimator, from the example jar file, as follows (some output elided):

Example

```
[fred@bright70 ~]$ module add hadoop/Apache220/Apache/2.2.0
[fred@bright70 ~]$ hadoop jar $HADOOP_PREFIX/share/hadoop/mapreduce/hado\
op-mapreduce-examples-2.2.0.jar pi 1 5
...
Job Finished in 19.732 seconds
Estimated value of Pi is 4.00000000000000000000
```

The `module add` line is not needed if the user has the module loaded by default (section 2.2.3 of the *Administrator Manual*).

The input takes the number of maps and number of samples as options—`1` and `5` in the example. The result can be improved with greater values for both.

6

Hadoop-related Projects

Several projects use the Hadoop framework. These projects may be focused on data warehousing, data-flow programming, or other data-processing tasks which Hadoop can handle well. Bright Cluster Manager provides tools to help install the following projects:

- Accumulo (section 6.1)
- Hive (section 6.2)
- Kafka (section 6.3)
- Pig (section 6.4)
- Spark (section 6.5)
- Sqoop (section 6.6)
- Storm (section 6.7)

6.1 Accumulo

Apache Accumulo is a highly-scalable, structured, distributed, key-value store based on Google's BigTable. Accumulo works on top of Hadoop and ZooKeeper. Accumulo stores data in HDFS, and uses a richer model than regular key-value stores. Keys in Accumulo consist of several elements.

An Accumulo instance includes the following main components:

- Tablet Server, which manages subsets of all tables
- Garbage Collector, to delete files no longer needed
- Master, responsible of coordination
- Tracer, collection traces about Accumulo operations
- Monitor, web application showing information about the instance

Also a part of the instance is a client library linked to Accumulo applications.

The Apache Accumulo tarball can be downloaded from `http://accumulo.apache.org/`. For Hortonworks HDP 2.1.x, the Accumulo tarball can be downloaded from the Hortonworks website (section 1.2).

6.1.1 Accumulo Installation With `cm-accumulo-setup`

Bright Cluster Manager provides `cm-accumulo-setup` to carry out the installation of Accumulo.

Prerequisites For Accumulo Installation, And What Accumulo Installation Does

The following applies to using `cm-accumulo-setup`:

- A Hadoop instance, with ZooKeeper, must already be installed.
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- The `cm-accumulo-setup` script only installs Accumulo on the active head node and on the DataNodes of the chosen Hadoop instance.
- The script assigns no roles to nodes.
- Accumulo executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`.
- Accumulo configuration files are copied by the script to under `/etc/hadoop/`. This is done both on the active headnode, and on the necessary image(s).
- By default, Accumulo Tablet Servers are set to use 1GB of memory. A different value can be set via `cm-accumulo-setup`.
- The secret string for the instance is a random string created by `cm-accumulo-setup`.
- A password for the `root` user must be specified.
- The Tracer service will use Accumulo user `root` to connect to Accumulo.
- The services for Garbage Collector, Master, Tracer, and Monitor are, by default, installed and run on the headnode. They can be installed and run on another node instead, as shown in the next example, using the `--master` option.
- A Tablet Server will be started on each DataNode.
- `cm-accumulo-setup` tries to build the native map library.
- Validation tests are carried out by the script.

The options for `cm-accumulo-setup` are listed on running `cm-accumulo-setup -h`.

An Example Run With `cm-accumulo-setup`

The option

`-p` *<rootpass>*

is mandatory. The specified password will also be used by the Tracer service to connect to Accumulo. The password will be stored in `accumulo-site.xml`, with read and write permissions assigned to `root` only.

The option
`-s <heapsize>`
is not mandatory. If not set, a default value of 1GB is used.

The option
`--master <nodename>`
is not mandatory. It is used to set the node on which the Garbage Collector, Master, Tracer, and Monitor services run. If not set, then these services are run on the head node by default.

Example

```
[root@bright70 ~]# cm-accumulo-setup -i hdfs1 -j /usr/lib/jvm/jre-1.7.0-\
openjdk.x86_64/ -p <rootpass> -s <heapsize> -t /tmp/accumulo-1.6.2-bin.tar.gz \
 --master node005
Accumulo release '1.6.2'
Accumulo GC, Master, Monitor, and Tracer services will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.6.0
Accumulo being installed... done.
Creating directories for Accumulo... done.
Creating module file for Accumulo... done.
Creating configuration files for Accumulo... done.
Updating images... done.
Setting up Accumulo directories in HDFS... done.
Executing 'accumulo init'... done.
Initializing services for Accumulo (on DataNodes)... done.
Initializing master services for Accumulo... done.
Waiting for NameNode to be ready... done.
Executing validation test... done.
Installation successfully completed.
Finished.
```

6.1.2 Accumulo Removal With `cm-accumulo-setup`

`cm-accumulo-setup` should also be used to remove the Accumulo instance. Data and metadata will not be removed.

Example

```
[root@bright70 ~]# cm-accumulo-setup -u hdfs1
Requested removal of Accumulo for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Accumulo directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.1.3 Accumulo MapReduce Example

Accumulo jobs must be run using `accumulo` system user.

Example

```
[root@bright70 ~]# su - accumulo
bash-4.1$ module load accumulo/hdfs1
bash-4.1$ cd $ACCUMULO_HOME
bash-4.1$ bin/tool.sh lib/accumulo-examples-simple.jar \
```

```
org.apache.accumulo.examples.simple.mapreduce.TeraSortIngest \
-i hdfs1 -z $ACCUMULO_ZOOKEEPERS -u root -p secret \
--count 10 --minKeySize 10 --maxKeySize 10 \
--minValueSize 78 --maxValueSize 78 --table sort --splits 10
```

6.2 Hive

Apache Hive is a data warehouse software. It stores its data using HDFS, and can query it via the SQL-like HiveQL language. Metadata values for its tables and partitions are kept in the Hive Metastore, which is an SQL database, typically MySQL or Postgres. Data can be exposed to clients using the following client-server mechanisms:

- Metastore, accessed with the `hive` client
- HiveServer2, accessed with the `beeline` client

The Apache Hive tarball should be downloaded from one of the locations specified in Section 1.2, depending on the chosen distribution.

6.2.1 Hive Installation With `cm-hive-setup`

Bright Cluster Manager provides `cm-hive-setup` to carry out Hive installation:

Prerequisites For Hive Installation, And What Hive Installation Does

The following applies to using `cm-hive-setup`:

- A Hadoop instance must already be installed.
- Before running the script, the version of the `mysql-connector-java` package should be checked. Hive works with releases 5.1.18 or earlier of this package. If `mysql-connector-java` provides a newer release, then the following must be done to ensure that Hive setup works:
  - a suitable 5.1.18 or earlier release of Connector/J is downloaded from `http://dev.mysql.com/downloads/connector/j/`
  - `cm-hive-setup` is run with the `--conn` option to specify the connector version to use.

    Example

    `--conn /tmp/mysql-connector-java-5.1.18-bin.jar`

- Before running the script, the following statements must be executed explicitly by the administrator, using a MySQL client:

  ```
  GRANT ALL PRIVILEGES ON <metastoredb>.* TO 'hive'@'%'\
   IDENTIFIED BY '<hivepass>';
  FLUSH PRIVILEGES;
  DROP DATABASE IF EXISTS <metastoredb>;
  ```

  In the preceding statements:

  - *<metastoredb>* is the name of metastore database to be used by Hive. The same name is used later by `cm-hive-setup`.

  - <*hivepass*> is the password for `hive` user. The same password is used later by `cm-hive-setup`.
  - The DROP line is needed only if a database with that name already exists.

- The `cm-hive-setup` script installs Hive by default on the active head node.

  It can be installed on another node instead, as shown in the next example, with the use of the `--master` option. In that case, Connector/J should be installed in the software image of the node.

- The script assigns no roles to nodes
- Hive executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Hive configuration files are copied by the script to under `/etc/hadoop/`
- The instance of MySQL on the head node is initialized as the Metastore database for the Bright Cluster Manager by the script. A different MySQL server can be specified by using the options `--mysqlserver` and `--mysqlport`.
- The data warehouse is created by the script in HDFS, in `/user/hive/warehouse`
- The Metastore and HiveServer2 services are started up by the script
- Validation tests are carried out by the script using `hive` and `beeline`.

An Example Run With `cm-hive-setup`

Example

```
[root@bright70 ~]# cm-hive-setup -i hdfs1 -j /usr/lib/jvm/jre-1.7.0-op\
enjdk.x86_64/ -p <hivepass> --metastoredb <metastoredb> -t /tmp/apache\
-hive-1.1.0-bin.tar.gz --master node005
Hive release '1.1.0-bin'
Using MySQL server on active headnode.
Hive service will be run on node: node005
Using MySQL Connector/J installed in /usr/share/java/
Hive being installed... done.
Creating directories for Hive... done.
Creating module file for Hive... done.
Creating configuration files for Hive... done.
Initializing database 'metastore_hdfs1' in MySQL... done.
Waiting for NameNode to be ready... done.
Creating HDFS directories for Hive... done.
Updating images... done.
Waiting for NameNode to be ready... done.
Hive setup validation...
-- testing 'hive' client...
-- testing 'beeline' client...
Hive setup validation... done.
Installation successfully completed.
Finished.
```

6.2.2 Hive Removal With cm-hive-setup

cm-hive-setup should also be used to remove the Hive instance. Data and metadata will not be removed.

Example

```
[root@bright70 ~]# cm-hive-setup -u hdfs1
Requested removal of Hive for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Hive directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.2.3 Beeline

The latest Hive releases include HiveServer2, which supports Beeline command shell. Beeline is a JDBC client based on the SQLLine CLI (http://sqlline.sourceforge.net/). In the following example, Beeline connects to HiveServer2:

Example

```
[root@bright70 ~]# beeline -u jdbc:hive2://node005.cm.cluster:10000 \
 -d org.apache.hive.jdbc.HiveDriver -e 'SHOW TABLES;'
Connecting to jdbc:hive2://node005.cm.cluster:10000
Connected to: Apache Hive (version 1.1.0)
Driver: Hive JDBC (version 1.1.0)
Transaction isolation: TRANSACTION_REPEATABLE_READ
+-----------+--+
| tab_name  |
+-----------+--+
| test      |
| test2     |
+-----------+--+
2 rows selected (0.243 seconds)
Beeline version 1.1.0 by Apache Hive
Closing: 0: jdbc:hive2://node005.cm.cluster:10000
```

6.3 Kafka

Apache Kafka is a distributed publish-subscribe messaging system. Among other usages, Kafka is used as a replacement for message broker, for website activity tracking, for log aggregation. The Apache Kafka tarball should be downloaded from http://kafka.apache.org/, where different pre-built tarballs are available, depeding on the preferred Scala version.

6.3.1 Kafka Installation With cm-kafka-setup

Bright Cluster Manager provides cm-kafka-setup to carry out Kafka installation.

Prerequisites For Kafka Installation, And What Kafka Installation Does

The following applies to using cm-kafka-setup:

- A Hadoop instance, with ZooKeeper, must already be installed.
- `cm-kafka-setup` installs Kafka only on the ZooKeeper nodes.
- The script assigns no roles to nodes.
- Kafka is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Kafka configuration files are copied by the script to under `/etc/hadoop/`.

An Example Run With `cm-kafka-setup`
Example

```
[root@bright70 ~]# cm-kafka-setup -i hdfs1 -j /usr/lib/jvm/jre-1.7.0-open\
jdk.x86_64/ -t /tmp/kafka_2.11-0.8.2.1.tgz
Kafka release '0.8.2.1' for Scala '2.11'
Found Hadoop instance 'hdfs1', release: 1.2.1
Kafka being installed... done.
Creating directories for Kafka... done.
Creating module file for Kafka... done.
Creating configuration files for Kafka... done.
Updating images... done.
Initializing services for Kafka (on ZooKeeper nodes)... done.
Executing validation test... done.
Installation successfully completed.
Finished.
```

6.3.2 Kafka Removal With `cm-kafka-setup`

`cm-kafka-setup` should also be used to remove the Kafka instance.

Example

```
[root@bright70 ~]# cm-kafka-setup -u hdfs1
Requested removal of Kafka for Hadoop instance hdfs1.
Stopping/removing services... done.
Removing module file... done.
Removing additional Kafka directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.4 Pig

Apache Pig is a platform for analyzing large data sets. Pig consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. Pig programs are intended by language design to fit well with "embarrassingly parallel" problems that deal with large data sets. The Apache Pig tarball should be downloaded from one of the locations specified in Section 1.2, depending on the chosen distribution.

6.4.1 Pig Installation With `cm-pig-setup`

Bright Cluster Manager provides `cm-pig-setup` to carry out Pig installation.

Prerequisites For Pig Installation, And What Pig Installation Does

The following applies to using `cm-pig-setup`:

- A Hadoop instance must already be installed.
- `cm-pig-setup` installs Pig only on the active head node.
- The script assigns no roles to nodes.
- Pig is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Pig configuration files are copied by the script to under `/etc/hadoop/`.

An Example Run With `cm-pig-setup`

Example

```
[root@bright70 ~]# cm-pig-setup -i hdfs1 -j /usr/lib/jvm/jre-1.7.0-open\
jdk.x86_64/ -t /tmp/pig-0.14.0.tar.gz
Pig release '0.14.0'
Pig being installed... done.
Creating directories for Pig... done.
Creating module file for Pig... done.
Creating configuration files for Pig... done.
Waiting for NameNode to be ready...
Waiting for NameNode to be ready... done.
Validating Pig setup...
Validating Pig setup... done.
Installation successfully completed.
Finished.
```

6.4.2 Pig Removal With `cm-pig-setup`

`cm-pig-setup` should also be used to remove the Pig instance.

Example

```
[root@bright70 ~]# cm-pig-setup -u hdfs1
Requested removal of Pig for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Pig directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.4.3 Using Pig

Pig consists of an executable, `pig`, that can be run after the user loads the corresponding module. Pig runs by default in "MapReduce Mode", that is, it uses the corresponding HDFS installation to store and deal with the elaborate processing of data. More thorough documentation for Pig can be found at `http://pig.apache.org/docs/r0.14.0/start.html`.

Pig can be used in interactive mode, using the Grunt shell:

```
[root@bright70 ~]# module load hadoop/hdfs1
[root@bright70 ~]# module load pig/hdfs1
[root@bright70 ~]# pig
```

```
14/08/26 11:57:41 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
14/08/26 11:57:41 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
14/08/26 11:57:41 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the Ex\
ecType
...
...
grunt>
```

or in batch mode, using a Pig Latin script:

```
[root@bright70 ~]# module load hadoop/hdfs1
[root@bright70 ~]# module load pig/hdfs1
[root@bright70 ~]# pig -v -f /tmp/smoke.pig
```

In both cases, Pig runs in "MapReduce mode", thus working on the corresponding HDFS instance.

6.5 Spark

Apache Spark is an engine for processing Hadoop data. It can carry out general data processing, similar to MapReduce, but typically faster.

Spark can also carry out the following, with the associated high-level tools:

- stream feed processing with Spark Streaming
- SQL queries on structured distributed data with Spark SQL
- processing with machine learning algorithms, using MLlib
- graph computation, for arbitrarily-connected networks, with graphX

The Apache Spark tarball should be downloaded from `http://spark.apache.org/`, where different pre-built tarballs are available: for Hadoop 1.x, for CDH 4, and for Hadoop 2.x.

6.5.1 Spark Installation With `cm-spark-setup`

Bright Cluster Manager provides `cm-spark-setup` to carry out Spark installation.

Prerequisites For Spark Installation, And What Spark Installation Does

The following applies to using `cm-spark-setup`:

- A Hadoop instance must already be installed
- Spark can be installed in two different deployment modes: Standalone or YARN.
  - Standalone mode. This is the default for Apache Hadoop 1.x, Cloudera CDH 4.x, and Hortonworks HDP 1.3.x.
    - It is possible to force the Standalone mode deployment by using the additional option:
      `--standalone`
    - When installing in standalone mode, the script installs Spark on the active head node and on the DataNodes of the chosen Hadoop instance.

The Spark Master service runs on the active head node by default, but can be specified to run on another node by using the option `--master`.
Spark Worker services run on all DataNodes.

- YARN mode. This is the default for Apache Hadoop 2.x, Cloudera CDH 5.x, Hortonworks 2.x, and Pivotal 2.x. The default can be overridden by using the `--standalone` option.
  - When installing in YARN mode, the script installs Spark only on the active head node.

- The script assigns no roles to nodes
- Spark is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Spark configuration files are copied by the script to under `/etc/hadoop/`

An Example Run With `cm-spark-setup` in YARN mode
Example

```
[root@bright70 ~]# cm-spark-setup -i hdfs1  \
 -j /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/ \
 -t /tmp/spark-1.3.0-bin-hadoop2.4.tgz
Spark release '1.3.0-bin-hadoop2.4'
Found Hadoop instance 'hdfs1', release: 2.6.0
Spark will be installed in YARN (client/cluster) mode.
Spark being installed... done.
Creating directories for Spark... done.
Creating module file for Spark... done.
Creating configuration files for Spark... done.
Waiting for NameNode to be ready... done.
Copying Spark assembly jar to HDFS... done.
Waiting for NameNode to be ready... done.
Validating Spark setup... done.
Installation successfully completed.
Finished.
```

An Example Run With `cm-spark-setup` in standalone mode
Example

```
[root@bright70 ~]# cm-spark-setup -i hdfs1 \
 -j /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/ \
 -t /tmp/spark-1.3.0-bin-hadoop2.4.tgz  \
 --standalone --master node005
Spark release '1.3.0-bin-hadoop2.4'
Found Hadoop instance 'hdfs1', release: 2.6.0
Spark will be installed to work in Standalone mode.
Spark Master service will be run on node: node005
Spark will use all DataNodes as WorkerNodes.
Spark being installed... done.
Creating directories for Spark... done.
Creating module file for Spark... done.
Creating configuration files for Spark... done.
Updating images... done.
Initializing Spark Master service... done.
Initializing Spark Worker service... done.
Validating Spark setup... done.
```

```
Installation successfully completed.
Finished.
```

6.5.2 Spark Removal With cm-spark-setup

cm-spark-setup should also be used to remove the Spark instance.

Example

```
[root@bright70 ~]# cm-spark-setup -u hdfs1
Requested removal of Spark for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Spark directories... done.
Removal successfully completed.
Finished.
```

6.5.3 Using Spark

Spark supports two deploy modes to launch Spark applications on YARN. Considering the SparkPi example provided with Spark:

- In "yarn-client" mode, the Spark driver runs in the client process, and the SparkPi application is run as a child thread of Application Master.

```
[root@bright70 ~]# module load spark/hdfs1
[root@bright70 ~]# spark-submit --master yarn-client \
 --class org.apache.spark.examples.SparkPi \
 $SPARK_PREFIX/lib/spark-examples-*.jar
```

- In "yarn-cluster" mode, the Spark driver runs inside an Application Master process which is managed by YARN on the cluster.

```
[root@bright70 ~]# module load spark/hdfs1
[root@bright70 ~]# spark-submit --master yarn-cluster \
 --class org.apache.spark.examples.SparkPi \
 $SPARK_PREFIX/lib/spark-examples-*.jar
```

6.6 Sqoop

Apache Sqoop is a tool designed to transfer bulk data between Hadoop and an RDBMS. Sqoop uses MapReduce to import and export data. Bright Cluster Manager supports transfers between Sqoop and MySQL.

RHEL7 and SLES12 use MariaDB, and are not yet supported by the available versions of Sqoop at the time of writing (April 2015). At present, the latest Sqoop stable release is 1.4.5, while the latest Sqoop2 version is 1.99.5. Sqoop2 is incompatible with Sqoop; it is not feature-complete; and it is not yet intended for production use. The Bright Computing utility cm-sqoop-setup does not as yet support Sqoop2.

6.6.1 Sqoop Installation With cm-sqoop-setup

Bright Cluster Manager provides cm-sqoop-setup to carry out Sqoop installation:

Prerequisites For Sqoop Installation, And What Sqoop Installation Does

The following requirements and conditions apply to running the `cm-sqoop-setup` script:

- A Hadoop instance must already be installed.
- Before running the script, the version of the `mysql-connector-java` package should be checked. Sqoop works with releases 5.1.34 or later of this package. If `mysql-connector-java` provides a newer release, then the following must be done to ensure that Sqoop setup works:
  - a suitable 5.1.34 or later release of Connector/J is downloaded from `http://dev.mysql.com/downloads/connector/j/`
  - `cm-sqoop-setup` is run with the `--conn` option in order to specify the connector version to be used.

    Example

    ```
    --conn /tmp/mysql-connector-java-5.1.34-bin.jar
    ```

- The `cm-sqoop-setup` script installs Sqoop only on the active head node. A different node can be specified by using the option `--master`.
- The script assigns no roles to nodes
- Sqoop executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Sqoop configuration files are copied by the script and placed under `/etc/hadoop/`
- The Metastore service is started up by the script.

An Example Run With `cm-sqoop-setup`

Example

```
[root@bright70 ~]# cm-sqoop-setup -i hdfs1 -j /usr/lib/jvm/jre-1.7.0-op\
enjdk.x86_64/ -t /tmp/sqoop-1.4.5.bin__hadoop-2.0.4-alpha.tar.gz\
 --conn /tmp/mysql-connector-java-5.1.34-bin.jar --master node005
Using MySQL Connector/J from /tmp/mysql-connector-java-5.1.34-bin.jar
Sqoop release '1.4.5.bin__hadoop-2.0.4-alpha'
Sqoop service will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.2.0
Sqoop being installed... done.
Creating directories for Sqoop... done.
Creating module file for Sqoop... done.
Creating configuration files for Sqoop... done.
Updating images... done.
Installation successfully completed.
Finished.
```

6.6.2 Sqoop Removal With `cm-sqoop-setup`

`cm-sqoop-setup` should be used to remove the Sqoop instance.

Example

```
[root@bright70 ~]# cm-sqoop-setup -u hdfs1
Requested removal of Sqoop for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Sqoop directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.7 Storm

Apache Storm is a distributed realtime computation system. While Hadoop is focused on batch processing, Storm can process streams of data.

Other parallels between Hadoop and Storm:

- users run "jobs" in Hadoop and "topologies" in Storm
- the master node for Hadoop jobs runs the "JobTracker" or "ResourceManager" daemons to deal with resource management and scheduling, while the master node for Storm runs an analogous daemon called "Nimbus"
- each worker node for Hadoop runs daemons called "TaskTracker" or "NodeManager", while the worker nodes for Storm runs an analogous daemon called "Supervisor"
- both Hadoop, in the case of NameNode HA, and Storm, leverage "ZooKeeper" for coordination

6.7.1 Storm Installation With `cm-storm-setup`

Bright Cluster Manager provides `cm-storm-setup` to carry out Storm installation.

Prerequisites For Storm Installation, And What Storm Installation Does

The following applies to using `cm-storm-setup`:

- A Hadoop instance, with ZooKeeper, must already be installed.
- The `cm-storm-setup` script only installs Storm on the active head node and on the DataNodes of the chosen Hadoop instance by default. A node other than master can be specified by using the option `--master`, or its alias for this setup script, `--nimbus`.
- The script assigns no roles to nodes.
- Storm executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`.
- Storm configuration files are copied by the script to under `/etc/hadoop/`. This is done both on the active headnode, and on the necessary image(s).
- Validation tests are carried out by the script.

An Example Run With `cm-storm-setup`

Example

```
[root@bright70 ~]# cm-storm-setup -i hdfs1 \
 -j /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/ \
 -t apache-storm-0.9.4.tar.gz
 --nimbus node005
Storm release '0.9.4'
Storm Nimbus and UI services will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.2.0
Storm being installed... done.
Creating directories for Storm... done.
Creating module file for Storm... done.
Creating configuration files for Storm... done.
Updating images... done.
Initializing worker services for Storm (on DataNodes)... done.
Initializing Nimbus services for Storm... done.
Executing validation test... done.
Installation successfully completed.
Finished.
```

The `cm-storm-setup` installation script submits a validation topology (topology in the Storm sense) called `WordCount`. After a successful installation, users can connect to the `Storm UI` on the host <nimbus>, the Nimbus server, at `http://<nimbus>:10080/`. There they can check the status of `WordCount`, and can kill it.

6.7.2 Storm Removal With `cm-storm-setup`

The `cm-storm-setup` script should also be used to remove the Storm instance.

Example

```
[root@bright70 ~]# cm-storm-setup -u hdfs1
Requested removal of Storm for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing additional Storm directories... done.
Updating images... done.
Removal successfully completed.
Finished.
```

6.7.3 Using Storm

The following example shows how to submit a topology, and then verify that it has been submitted successfully (some lines elided):

```
[root@bright70 ~]# module load storm/hdfs1
[root@bright70 ~]# storm jar /cm/shared/apps/hadoop/Apache/\
 apache-storm-0.9.3/examples/storm-starter/\
 storm-starter-topologies-*.jar \
 storm.starter.WordCountTopology WordCount2

...

470  [main] INFO  backtype.storm.StormSubmitter - Jar not uploaded to m\
```

```
aster yet. Submitting jar...
476  [main] INFO  backtype.storm.StormSubmitter - Uploading topology ja\
r /cm/shared/apps/hadoop/Apache/apache-storm-0.9.3/examples/storm-start\
er/storm-starter-topologies-0.9.3.jar to assigned location: /tmp/storm-\
hdfs1-local/nimbus/inbox/stormjar-bf1abdd0-f31a-41ff-b808-4daad1dfdaa3.\
jar
Start uploading file '/cm/shared/apps/hadoop/Apache/apache-storm-0.9.3/\
examples/storm-starter/storm-starter-topologies-0.9.3.jar' to '/tmp/sto\
rm-hdfs1-local/nimbus/inbox/stormjar-bf1abdd0-f31a-41ff-b808-4daad1dfda\
a3.jar' (3248859 bytes)
[==================================================] 3248859 / 3248859
File '/cm/shared/apps/hadoop/Apache/apache-storm-0.9.3/examples/storm-s\
tarter/storm-starter-topologies-0.9.3.jar' uploaded to '/tmp/storm-hdfs\
1-local/nimbus/inbox/stormjar-bf1abdd0-f31a-41ff-b808-4daad1dfdaa3.jar'\
 (3248859 bytes)
508  [main] INFO  backtype.storm.StormSubmitter - Successfully uploaded\
 topology jar to assigned location: /tmp/storm-hdfs1-local/nimbus/inbox\
/stormjar-bf1abdd0-f31a-41ff-b808-4daad1dfdaa3.jar
508  [main] INFO  backtype.storm.StormSubmitter - Submitting topology W\
ordCount2 in distributed mode with conf "topology.workers":3,"topology\
.debug":true
687  [main] INFO  backtype.storm.StormSubmitter - Finished submitting t\
opology: WordCount2
[root@hadoopdev ~]# storm list

...

Topology_name        Status     Num_tasks  Num_workers  Uptime_secs
-------------------------------------------------------------------
WordCount2           ACTIVE     28         3            15
```