

Bright Cluster Manager 7.2

Hadoop Deployment Manual

Revision: f6f97b6

Date: Fri Sep 13 2024



©2015 Bright Computing, Inc. All Rights Reserved. This manual or parts thereof may not be reproduced in any form unless permitted by contract or by written permission of Bright Computing, Inc.

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 NVIDIA Corporation. FLEXlm is a registered trademark of Flexera Software, 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

| | |
|--|-----------|
| Table of Contents | i |
| 0.1 About This Manual | v |
| 0.2 About The Manuals In General | v |
| 0.3 Getting Administrator-Level Support | vi |
| 1 Introduction | 1 |
| 1.1 What Is Hadoop About? | 1 |
| 1.2 Available Hadoop Implementations | 2 |
| 1.3 Further Documentation | 2 |
| 1.4 Version Support Matrix | 2 |
| 1.4.1 Apache Hadoop 1.2.1 | 3 |
| 1.4.2 Hortonworks HDP 1.3.11 | 4 |
| 1.4.3 Apache Hadoop 2.7.2 | 4 |
| 1.4.4 Cloudera CDH 4.7.1 | 5 |
| 1.4.5 Cloudera CDH 5.4.10 | 5 |
| 1.4.6 Cloudera CDH 5.5.4 | 6 |
| 1.4.7 Cloudera CDH 5.6.1 | 6 |
| 1.4.8 Cloudera CDH 5.7.1 | 7 |
| 1.4.9 Hortonworks HDP 2.2.9 | 7 |
| 1.4.10 Hortonworks HDP 2.3.4.7 | 8 |
| 1.4.11 Hortonworks HDP 2.4.2 | 8 |
| 1.4.12 Pivotal HD 2.1.0 | 9 |
| 1.4.13 Pivotal HD 3.0.1 | 9 |
| 2 Installing Hadoop | 11 |
| 2.1 Command-line Installation Of Hadoop Using <code>cm-hadoop-setup -c <filename></code> . . | 11 |
| 2.1.1 Usage | 11 |
| 2.1.2 An Install Run | 12 |
| 2.2 Ncurses Installation Of Hadoop Using <code>cm-hadoop-setup</code> | 14 |
| 2.3 Installation And Removal Of Hadoop In <code>cmgui</code> | 15 |
| 2.4 Installing Hadoop With Lustre | 21 |
| 2.4.1 Lustre Internal Server Installation | 21 |
| 2.4.2 Lustre External Server Installation | 21 |
| 2.4.3 Lustre Client Installation | 22 |
| 2.4.4 Lustre Hadoop Configuration With HAL | 22 |
| 2.4.5 Lustre Hadoop Configuration With HAL And HAM | 24 |
| 2.5 Installation of other Hadoop components | 25 |
| 2.6 Hadoop Installation In A Cloud | 25 |

| | | |
|----------|---|-----------|
| 3 | Hadoop Cluster Management | 27 |
| 3.1 | Managing A Hadoop Instance With <code>cmgui</code> | 27 |
| 3.1.1 | The HDFS Instance Overview Tab | 28 |
| 3.1.2 | The HDFS Instance Settings Tab | 28 |
| 3.1.3 | The HDFS Instance HDFS Tab | 31 |
| 3.1.4 | The HDFS Instance MapReduce Or YARN Tab | 31 |
| 3.1.5 | The HDFS Instance HBase Tab | 32 |
| 3.1.6 | The HDFS Instance Zookeeper Tab | 33 |
| 3.1.7 | The HDFS Instance Spark Tab | 33 |
| 3.1.8 | The HDFS Instance More Tab | 33 |
| 3.1.9 | The HDFS Instance Hadoop Configuration Groups Tab | 33 |
| 3.1.10 | The HDFS Instance Monitoring Tab | 38 |
| 3.1.11 | The HDFS Instance Notes Tab | 38 |
| 3.2 | Managing A Hadoop Instance With <code>cmsh</code> | 38 |
| 3.2.1 | <code>cmsh</code> And <code>hadoop</code> Mode | 38 |
| 3.2.2 | <code>cmsh</code> And <code>configurationoverlay</code> Mode | 43 |
| 3.2.3 | <code>cmsh</code> And The <code>roleoverview</code> Command In <code>device</code> Mode | 45 |
| 3.3 | Hadoop Maintenance Operations With <code>cm-hadoop-maint</code> | 45 |
| 4 | Running Hadoop Jobs | 49 |
| 4.1 | Shakedown Runs | 49 |
| 4.2 | Example End User Job Run | 51 |
| 5 | Spark Support In Bright Cluster Manager | 53 |
| 5.1 | Spark Installation In Bright Cluster Manager–Overview | 53 |
| 5.1.1 | Prerequisites For Spark Installation, And What Spark Installation Does | 53 |
| 5.1.2 | Spark Installation Utility: <code>cm-spark-setup</code> | 54 |
| 5.2 | Spark Installation In YARN Mode | 54 |
| 5.2.1 | Using Spark In YARN Mode | 55 |
| 5.2.2 | Spark Removal With <code>cm-spark-setup</code> | 57 |
| 5.3 | Spark Installation In Standalone Mode | 57 |
| 5.3.1 | Using Spark In Standalone Mode | 59 |
| 5.4 | Zeppelin Installation | 62 |
| 5.4.1 | Zeppelin Installation With <code>cmhadoop-zeppelin-setup</code> | 62 |
| 5.4.2 | Zeppelin Removal With <code>cmhadoop-zeppelin-setup</code> | 63 |
| 5.5 | Tachyon Installation | 63 |
| 5.5.1 | Tachyon Installation With <code>cmhadoop-tachyon-setup</code> | 63 |
| 5.5.2 | Tachyon Removal With <code>cmhadoop-tachyon-setup</code> | 64 |
| 5.5.3 | Using Tachyon | 65 |
| 6 | Hadoop-related Projects | 67 |
| 6.1 | Accumulo | 67 |
| 6.1.1 | Accumulo Installation With <code>cmhadoop-accumulo-setup</code> | 68 |
| 6.1.2 | Accumulo Removal With <code>cmhadoop-accumulo-setup</code> | 69 |
| 6.1.3 | Accumulo MapReduce Example | 70 |
| 6.2 | Drill | 70 |
| 6.2.1 | Drill Installation With <code>cmhadoop-drill-setup</code> | 70 |

| | | |
|----------|--|-----------|
| 6.2.2 | Drill Removal With <code>cmhadoop-drill-setup</code> | 71 |
| 6.3 | Flink | 71 |
| 6.3.1 | Flink Installation With <code>cmhadoop-flink-setup</code> | 71 |
| 6.3.2 | Flink Removal With <code>cmhadoop-flink-setup</code> | 72 |
| 6.4 | Giraph | 73 |
| 6.4.1 | Giraph Installation With <code>cmhadoop-giraph-setup</code> | 73 |
| 6.4.2 | Giraph Removal With <code>cmhadoop-giraph-setup</code> | 74 |
| 6.5 | Hive | 74 |
| 6.5.1 | Hive Installation With <code>cmhadoop-hive-setup</code> | 74 |
| 6.5.2 | Hive Removal With <code>cmhadoop-hive-setup</code> | 76 |
| 6.5.3 | Beeline | 76 |
| 6.6 | Ignite | 77 |
| 6.6.1 | Ignite Installation With <code>cmhadoop-ignite-setup</code> | 77 |
| 6.6.2 | Ignite Removal With <code>cmhadoop-ignite-setup</code> | 78 |
| 6.6.3 | Using Ignite | 78 |
| 6.7 | Kafka | 79 |
| 6.7.1 | Kafka Installation With <code>cmhadoop-kafka-setup</code> | 79 |
| 6.7.2 | Kafka Removal With <code>cmhadoop-kafka-setup</code> | 80 |
| 6.8 | Pig | 80 |
| 6.8.1 | Pig Installation With <code>cmhadoop-pig-setup</code> | 81 |
| 6.8.2 | Pig Removal With <code>cmhadoop-pig-setup</code> | 81 |
| 6.8.3 | Using Pig | 82 |
| 6.9 | Sqoop | 82 |
| 6.9.1 | Sqoop Installation With <code>cmhadoop-sqoop-setup</code> | 82 |
| 6.9.2 | Sqoop Removal With <code>cmhadoop-sqoop-setup</code> | 83 |
| 6.9.3 | Using Sqoop To Import A Table From MySQL | 84 |
| 6.10 | Sqoop2 | 85 |
| 6.10.1 | Sqoop2 Removal With <code>cmhadoop-sqoop-setup</code> | 85 |
| 6.11 | Storm | 86 |
| 6.11.1 | Storm Installation With <code>cmhadoop-storm-setup</code> | 86 |
| 6.11.2 | Storm Removal With <code>cmhadoop-storm-setup</code> | 87 |
| 6.11.3 | Using Storm | 87 |
| 6.12 | Tachyon | 88 |
| 6.12.1 | Tachyon Installation With <code>cmhadoop-tachyon-setup</code> | 88 |
| 6.12.2 | Tachyon Removal With <code>cmhadoop-tachyon-setup</code> | 89 |
| 6.12.3 | Using Tachyon | 90 |
| A | Details And Examples Of Hadoop Configuration | 91 |
| A.1 | Hadoop Components Activation And Deactivation Using Roles | 91 |
| A.2 | Only The Enabled Hadoop Components And Roles Are Available For Activation From <code>cmgui</code> And <code>cmsh</code> | 91 |
| A.3 | Example Of Role Priority Overrides In Configuration Groups With <code>cmsh</code> | 92 |
| A.4 | Cloning Hadoop Configuration Groups In <code>cmgui</code> And <code>cmsh</code> | 94 |
| A.4.1 | Cloning Hadoop Configuration Groups In <code>cmgui</code> | 94 |
| A.4.2 | Cloning Hadoop Configuration Groups In <code>cmsh</code> | 98 |
| A.5 | Considerations And Best Practices When Creating Or Cloning Hadoop Configurations . . | 99 |
| A.6 | Customizations For Configuration Overlays | 100 |

| | | |
|-------|---|-----|
| A.6.1 | Customization Example Overview | 100 |
| A.6.2 | Adding Customizations To A Configuration Overlay In <code>cmsh</code> | 101 |
| A.6.3 | Managing Customizations From <code>cmgui</code> | 102 |

Preface

Welcome to the *Hadoop Deployment Manual* for Bright Cluster Manager 7.2.

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 7.2 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

If the reseller from whom Bright Cluster Manager was bought offers direct support, then the reseller should be contacted.

Otherwise the primary means of support is via the website <https://support.brightcomputing.com>. This allows the administrator to submit a support request via a web form, and opens up a trouble ticket. It is a good idea to try to use a clear subject header, since that is used as part of a reference tag as the ticket progresses. Also helpful is a good description of the issue. The followup communication for this ticket goes via standard e-mail. Section 11.2 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 distributed data processing technology used for the analysis of very large and often unstructured datasets. The dataset size typically ranges from several terabytes to petabytes. The size and lack of structure of the dataset means that it cannot be stored or handled efficiently in regular relational databases, which typically manage regularly structured data of the order of terabytes.

For very large unstructured data-sets, the term *big data* is often used. The analysis, or *data-mining* of big data is typically carried out more efficiently by Hadoop than by relational databases, for certain types of parallelizable problems. This is because of the following characteristics of Hadoop, 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 7.2 integrates with a number of Hadoop distributions 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 (<http://pivotal.io/big-data/pivotal-hd>): Pivotal Hadoop Distribution is a completely Apache-compliant distribution with extensive analytic toolsets. Pivotal HD, versions 2.1.0 and 3.0.1, are based on Apache Hadoop 2.2.0 and 2.6.0 respectively. As of April 9, 2016, Pivotal HD has reached its “End of Availability”, which means that it is no longer for sale.

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 distributions supported by Bright Computing are listed in Section 1.4.

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. The default location for the tarballs is under `/cm/local/apps/hadoop`. The documentation is unpacked into a relative directory path, with a starting point indicated in the table below:

| Hadoop version | Relative path |
|--------------------|---|
| Apache 1.2.1 | <code>hadoop-1.2.1/docs/index.html</code> |
| Apache 2.7.1 | <code>hadoop-2.7.1/share/doc/hadoop/index.html</code> |
| Cloudera CDH 5.5.4 | <code>hadoop-2.6.0-cdh5.5.4/share/doc/index.html</code> |
| Cloudera CDH 5.7.1 | <code>hadoop-2.6.0-cdh5.7.1/share/doc/index.html</code> |
| Hortonworks HDP | <i>Online documentation is available at http://docs.hortonworks.com/</i> |

1.4 Version Support Matrix

The Hadoop and Hadoop-related software versions that Bright Cluster Manager supports are listed in this section for the various Hadoop implementations in sections 1.4.1-1.4.13.

Each software is provided as a package, either from a Bright repository, or from the project site, or from the implementation provider. How it is obtained, and where it is obtained from, are indicated by superscripts as follows:

| Superscript | Obtained as | Location |
|-------------|--------------|---|
| a | package in | cm-apache-hadoop |
| b | package in | cm-apache-hadoop-extras |
| c | package in | cm-cloudera-hadoop |
| d | package in | cm-hortonworks-hadoop |
| x | pick up from | Drill, Flink, Ignite, Sqoop, Spark, Storm |
| <i>none</i> | pick up from | Hortonworks, Cloudera, Pivotal |

Thus, x as a superscript means the software must be picked up from the corresponding Apache project website. The website URL associated with the project is given in the following table:

| Project | URL |
|--------------|---|
| Drill | http://drill.apache.org |
| Flink | http://flink.apache.org |
| Ignite | http://ignite.apache.org |
| Sqoop | http://sqoop.apache.org |
| Spark | http://spark.apache.org |
| Apache Storm | http://storm.apache.org |

Similarly, no superscript means that the software is available from the corresponding implementation provider website, as follows:

| Project | URL |
|-------------|--|
| Hortonworks | http://hortonworks.com or directly from this URL at the time of writing (March 2016): http://s3.amazonaws.com/public-repo-1.hortonworks.com/index.html |
| Cloudera | http://www.cloudera.com or directly from these URLs, depending on version: http://archive.cloudera.com/cdh4/cdh/4/ http://archive.cloudera.com/cdh5/cdh/5/ http://archive.cloudera.com/cdh6/cdh/6/ |
| Pivotal | http://pivotal.io/big-data/pivotal-hd |

Some other tools, like Apache Giraph, should be built from source, depending on the chosen Hadoop version and distribution. More details are given in the sections for these tools in Chapter 6.

1.4.1 Apache Hadoop 1.2.1

- `hadoop-1.2.1.tar.gz`^a
- `zookeeper-3.4.8.tar.gz`^a
- `hbase-0.98.20-hadoop1-bin.tar.gz`^a
- `apache-hive-1.2.1-bin.tar.gz`^x
- `pig-0.16.0.tar.gz`^b
- `spark-1.6.2-bin-hadoop1-scala2.11.tgz`^x

- accumulo-1.5.4-bin.tar.gz^x
- apache-storm-1.0.1.tar.gz^b
- sqoop-1.4.6.bin__hadoop-1.0.0.tar.gz^x
- kafka_2.11-0.10.0.0.tgz^b
- apache-drill-1.7.0.tar.gz^x
- flink-0.9.1-bin-hadoop1.tgz^x
- Ignite not supported

1.4.2 Hortonworks HDP 1.3.11

This software is available from the Hortonworks website except where specified.

- hadoop-1.2.0.1.3.11.0-26.tar.gz^d
- zookeeper-3.4.5.1.3.11.0-26.tar.gz^d
- hbase-0.94.6.1.3.11.0-26-security.tar.gz^d
- hive-0.11.0.1.3.11.0-26.tar.gz
- pig-0.11.1.1.3.11.0-26.tar.gz
- spark-1.5.1-bin-hadoop1.tgz^x
- accumulo-1.5.4-bin.tar.gz^x
- apache-storm-1.0.1.tar.gz^b
- sqoop-1.4.3.1.3.11.0-26.bin__hadoop-1.2.0.1.3.11.0-26.tar.gz
- kafka_2.11-0.10.0.0.tgz^b
- apache-drill-1.7.0.tar.gz^x
- Flink not supported
- Ignite not supported

1.4.3 Apache Hadoop 2.7.2

- hadoop-2.7.2.tar.gz^a
- zookeeper-3.4.8.tar.gz^a
- hbase-1.2.1-bin.tar.gz^a
- apache-hive-2.1.0-bin.tar.gz^b
- pig-0.16.0.tar.gz^b
- spark-1.6.2-bin-hadoop2.6.tgz^b
- accumulo-1.7.2-bin.tar.gz^b
- apache-storm-1.0.1.tar.gz^b
- sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz^b
- sqoop-1.99.6-bin-hadoop200.tar.gz^x

- kafka_2.11-0.10.0.0.tgz^b
- apache-drill-1.7.0.tar.gz^x
- flink-1.0.3-bin-hadoop27-scala_2.11.tgz^x
- apache-ignite-hadoop-1.6.0-bin.zip^x

1.4.4 Cloudera CDH 4.7.1

This software is available from the Cloudera website except where specified.

- hadoop-2.0.0-cdh4.7.1.tar.gz^c
- zookeeper-3.4.5-cdh4.7.1.tar.gz^c
- hbase-0.94.15-cdh4.7.1.tar.gz^c
- hive-0.10.0-cdh4.7.1.tar.gz
- pig-0.11.0-cdh4.7.1.tar.gz
- spark-1.6.2-bin-cdh4.tgz^b
- accumulo-1.6.2-bin.tar.gz^x
- apache-storm-1.0.1.tar.gz^b
- sqoop-1.4.3-cdh4.7.1.tar.gz
- sqoop2-1.99.2-cdh4.7.1.tar.gz
- kafka_2.11-0.10.0.0.tgz^b
- apache-drill-1.7.0.tar.gz^x
- flink-0.10.1-bin-hadoop2-scala_2.10.tgz^x
- Ignite not supported

1.4.5 Cloudera CDH 5.4.10

This software is available from the Cloudera website except where specified.

- hadoop-2.6.0-cdh5.4.10.tar.gz
- zookeeper-3.4.5-cdh5.4.10.tar.gz
- hbase-1.0.0-cdh5.4.10.tar.gz
- hive-1.1.0-cdh5.4.10.tar.gz
- pig-0.12.0-cdh5.4.10.tar.gz
- spark-1.6.2-bin-hadoop2.6.tgz^b
- accumulo-1.7.2-bin.tar.gz^b
- apache-storm-1.0.1.tar.gz^b
- sqoop-1.4.5-cdh5.4.10.tar.gz
- sqoop2-1.99.5-cdh5.4.10.tar.gz
- kafka_2.11-0.10.0.0.tgz^b

- `apache-drill-1.7.0.tar.gzx`
- `flink-1.0.3-bin-hadoop26-scala_2.11.tgzx`
- `apache-ignite-hadoop-1.6.0-bin.zipx`

1.4.6 Cloudera CDH 5.5.4

This software is available from the Cloudera website except where specified.

- `hadoop-2.6.0-cdh5.5.4.tar.gz`
- `zookeeper-3.4.5-cdh5.5.4.tar.gz`
- `hbase-1.0.0-cdh5.5.4.tar.gz`
- `hive-1.1.0-cdh5.5.4.tar.gz`
- `pig-0.12.0-cdh5.5.4.tar.gz`
- `spark-1.6.2-bin-hadoop2.6.tgzb`
- `accumulo-1.7.2-bin.tar.gzb`
- `apache-storm-1.0.0.tar.gzb`
- `sqoop-1.4.5-cdh5.5.4.tar.gz`
- `sqoop2-1.99.5-cdh5.5.4.tar.gz`
- `kafka_2.11-0.10.0.0.tgzb`
- `apache-drill-1.7.0.tar.gzx`
- `flink-1.0.3-bin-hadoop26-scala_2.11.tgzx`
- `apache-ignite-hadoop-1.6.0-bin.zipx`

1.4.7 Cloudera CDH 5.6.1

This software is available from the Cloudera website except where specified.

- `hadoop-2.6.0-cdh5.6.1.tar.gzx`
- `zookeeper-3.4.5-cdh5.6.1.tar.gzx`
- `hbase-1.0.0-cdh5.6.1.tar.gzx`
- `hive-1.1.0-cdh5.6.1.tar.gz`
- `pig-0.12.0-cdh5.6.1.tar.gz`
- `spark-1.6.2-bin-hadoop2.6.tgzb`
- `accumulo-1.7.2-bin.tar.gzb`
- `apache-storm-1.0.1.tar.gzb`
- `sqoop-1.4.5-cdh5.6.1.tar.gz`
- `sqoop2-1.99.5-cdh5.6.1.tar.gz`
- `kafka_2.11-0.10.0.0.tgzb`
- `apache-drill-1.7.0.tar.gzx`
- `flink-1.0.3-bin-hadoop26-scala_2.11.tgzx`
- `apache-ignite-hadoop-1.6.0-bin.zipx`

1.4.8 Cloudera CDH 5.7.1

This software is available from the Cloudera website except where specified.

- `hadoop-2.6.0-cdh5.7.1.tar.gzc`
- `zookeeper-3.4.5-cdh5.7.1.tar.gzc`
- `hbase-1.2.0-cdh5.7.1.tar.gzc`
- `hive-1.1.0-cdh5.7.1.tar.gz`
- `pig-0.12.0-cdh5.7.1.tar.gz`
- `spark-1.6.2-bin-hadoop2.6.tgzb`
- `accumulo-1.7.2-bin.tar.gzb`
- `apache-storm-1.0.1.tar.gzb`
- `sqoop-1.4.5-cdh5.7.1.tar.gz`
- `sqoop2-1.99.5-cdh5.7.1.tar.gz`
- `kafka_2.11-0.10.0.0.tgzb`
- `apache-drill-1.7.0.tar.gzx`
- `flink-1.0.3-bin-hadoop26-scala_2.11.tgzx`
- `apache-ignite-hadoop-1.6.0-bin.zipx`

1.4.9 Hortonworks HDP 2.2.9

This software is available from the Hortonworks website except where specified.

- `hadoop-2.6.0.2.2.9.0-3393.tar.gz`
- `zookeeper-3.4.6.2.2.9.0-3393.tar.gz`
- `hbase-0.98.4.2.2.9.0-3393-hadoop2.tar.gz`
- `apache-hive-0.14.0.2.2.9.0-3393-bin.tar.gz`
- `pig-0.14.0.2.2.9.0-3393.tar.gz`
- `spark-1.6.2-bin-hadoop2.6.tgzb`
- `accumulo-1.6.1.2.2.9.0-3393-bin.tar.gz`
- `apache-storm-0.9.3.2.2.9.0-3393.tar.gz`
- `sqoop-1.4.5.2.2.6.0-2800.bin__hadoop-2.6.0.2.2.9.0-3393.tar.gz`
- `sqoop-1.99.6-bin-hadoop200.tar.gzx`
- `kafka_2.11-0.10.0.0.tgzb`
- `apache-drill-1.7.0.tar.gzx`
- `flink-1.0.3-bin-hadoop26-scala_2.11.tgzx`
- `apache-ignite-hadoop-1.6.0-bin.zipx`

1.4.10 Hortonworks HDP 2.3.4.7

This software is available from the Hortonworks website except where specified.

- `hadoop-2.7.1.2.3.4.7-4.tar.gz`
- `zookeeper-3.4.6.2.3.4.7-4.tar.gz`
- `hbase-1.1.2.2.3.4.7-4.tar.gz`
- `apache-hive-1.2.1.2.3.4.7-4-bin.tar.gz`
- `pig-0.15.0.2.3.4.7-4.tar.gz`
- `spark-1.6.2-bin-hadoop2.6.tgzb`
- `accumulo-1.7.0.2.3.4.7-4-bin.tar.gz`
- `apache-storm-0.10.0.2.3.4.7-4.tar.gz`
- `sqoop-1.4.6.2.3.4.7-4.bin__hadoop-2.7.1.2.3.4.7-4.tar.gz`
- `sqoop-1.99.6-bin-hadoop200.tar.gzx`
- `kafka_2.11-0.10.0.0.tgzb`
- `apache-drill-1.7.0.tar.gzx`
- `flink-1.0.3-bin-hadoop27-scala_2.11.tgzx`
- `apache-ignite-hadoop-1.6.0-bin.zipx`

1.4.11 Hortonworks HDP 2.4.2

This software is available from the Hortonworks website except where specified.

- `hadoop-2.7.1.2.4.2.0-258.tar.gzd`
- `zookeeper-3.4.6.2.4.2.0-258.tar.gzd`
- `hbase-1.1.2.2.4.2.0-258.tar.gzd`
- `apache-hive-1.2.1000.2.4.2.0-258-bin.tar.gz`
- `pig-0.15.0.2.4.2.0-258.tar.gz`
- `spark-1.6.2-bin-hadoop2.6.tgzb`
- `accumulo-1.7.0.2.4.2.0-258-bin.tar.gz`
- `apache-storm-0.10.0.2.4.2.0-258.tar.gz`
- `sqoop-1.4.6.2.4.0.0-169.bin__hadoop-2.7.1.2.4.2.0-258.tar.gz`
- `sqoop-1.99.6-bin-hadoop200.tar.gzx`
- `kafka_2.11-0.10.0.0.tgzb`
- `apache-drill-1.7.0.tar.gzx`
- `flink-1.0.3-bin-hadoop27-scala_2.11.tgzx`
- `apache-ignite-hadoop-1.6.0-bin.zipx`

1.4.12 Pivotal HD 2.1.0

The software is available from the Pivotal website except where specified. As of April 9, 2016, Pivotal HD has reached its “End of Availability”, which means that it is no longer for sale.

- PHD-2.1.0.0-175.tar.gz
- apache-hive-2.0.0-bin.tar.gz^x
- pig-0.15.0.tar.gz^x
- spark-1.2.1-bin-hadoop2.4.tgz^x
- accumulo-1.7.1-bin.tar.gz^x
- apache-storm-0.10.0.tar.gz^x
- sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz^b
- kafka_2.11-0.9.0.1.tgz^x
- apache-drill-1.6.0.tar.gz^x
- flink-0.10.1-bin-hadoop2-scala_2.10.tgz^x
- apache-ignite-hadoop-1.5.0.final-bin.zip^x

1.4.13 Pivotal HD 3.0.1

The software is available from the Pivotal website except where specified. As of April 9, 2016, Pivotal HD has reached its “End of Availability”, which means that it is no longer for sale.

- PHD-3.0.1.0-1-centos6.tar.gz
or
PHD-3.0.1.0-1-suse11sp3.tar.gz
- apache-hive-2.0.0-bin.tar.gz^x
- pig-0.15.0.tar.gz^x
- spark-1.6.1-bin-hadoop2.6.tgz^x
- accumulo-1.7.1-bin.tar.gz^x
- apache-storm-0.10.0.tar.gz^x
- sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz^b
- sqoop-1.99.5-bin-hadoop200.tar.gz^x
- kafka_2.11-0.9.0.1.tgz^x
- apache-drill-1.6.0.tar.gz^x
- flink-1.0.0-bin-hadoop26-scala_2.10.tgz^x
- apache-ignite-hadoop-1.5.0.final-bin.zip^x

2

Installing Hadoop

If a big data distribution has been selected, then the user can install Bright Cluster Manager with it. There are currently 12 supported Hadoop/Spark cluster versions, along with a chosen mode of operation in Bright Cluster Manager 7.2. Besides this, a number of Hadoop components can be installed/removed depending on the user's needs.

A Hadoop instance can be installed via

- the command-line (section 2.1)
- an ncurses GUI (section 2.2)
- or via a Hadoop installation wizard in `cmgui` (section 2.3)

The first two options are carried out with the `cm-hadoop-setup` script, which is run from a head node. The script is part of the `cluster-tools` package, and uses tarballs from the Apache Hadoop project. The third option runs as its own application.

Finally, if the Bright Cluster Manager installation ISO provided by Bright Computing is the Bright Cluster Manager With Hadoop installation ISO, then this 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@bright72 ~]# 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@bright72 ~]# cd /cm/local/apps/cluster-tools/hadoop/conf
[root@bright72 conf]# grep 'archive>' hadoop1conf.xml | grep -o /.*.gz
/tmp/hadoop-1.2.1.tar.gz
/tmp/zookeeper-3.4.6.tar.gz
/tmp/hbase-0.98.12.1-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@bright72 ~]# 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\
figure 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 Big Data resource tree folder (figure 2.1).

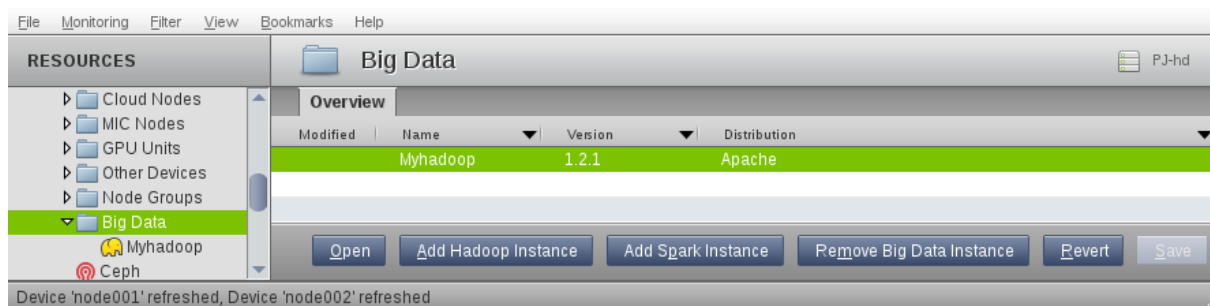


Figure 2.1: A Hadoop Instance In `cmgui`

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

Example

```
[root@bright72 ~] cmlsh
[bright72]% hadoop
[bright72->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@bright72 ~]# 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).

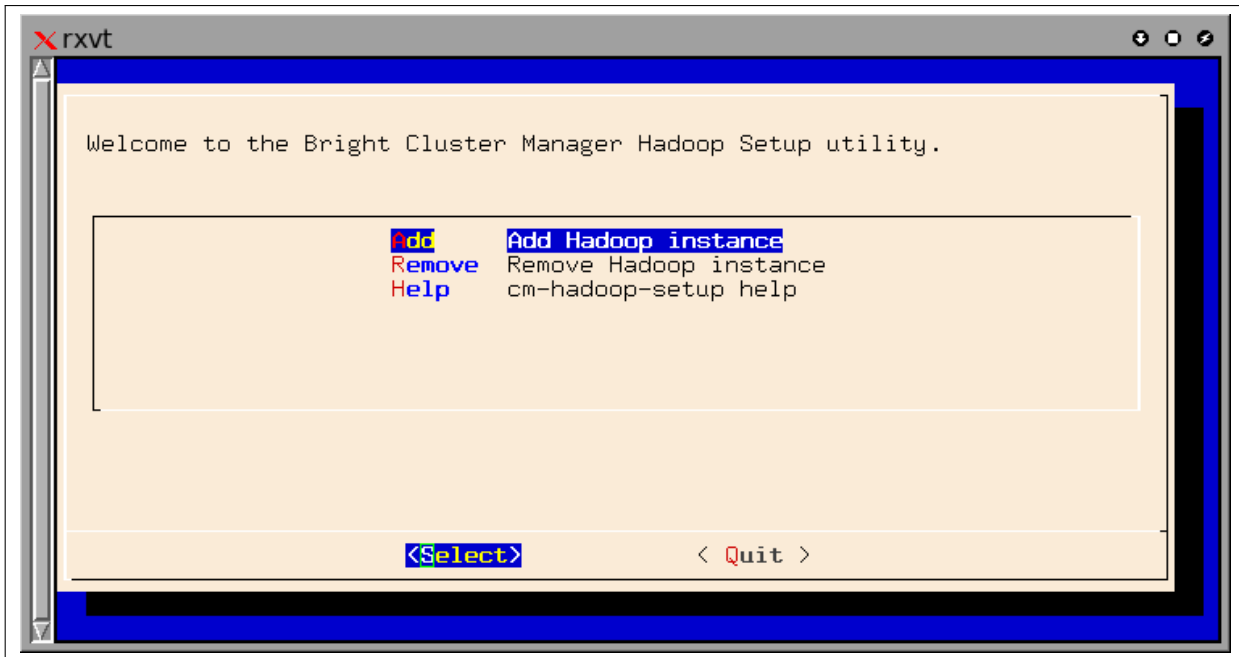


Figure 2.2: The cm-hadoop-setup Welcome Screen

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.

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 Installation And Removal Of Hadoop In cmgui

A Hadoop instance can be installed or removed using a `cmgui` wizard.

Packages that provide the Hadoop or Spark instance should be installed separately, before the wizard is used. How to pick up the correct packages is covered in section 1.4.

Typically, the administrator uses `yum` to install a package from the Bright Computing repository, with the default paths already preconfigured:

- `cm-apache-hadoop.noarch`
- `cm-apache-hadoop-extras.noarch`
- `cm-hortonworks-hadoop.noarch`
- `cm-cloudera-hadoop.noarch`

An alternative is that the administrator can download a big data distribution, and when the wizard is run, the paths can be set.

The wizard is launched from within the `Overview` tab of the `Big Data` resource, by clicking on a button to add a Hadoop or Spark instance (figure 2.3).

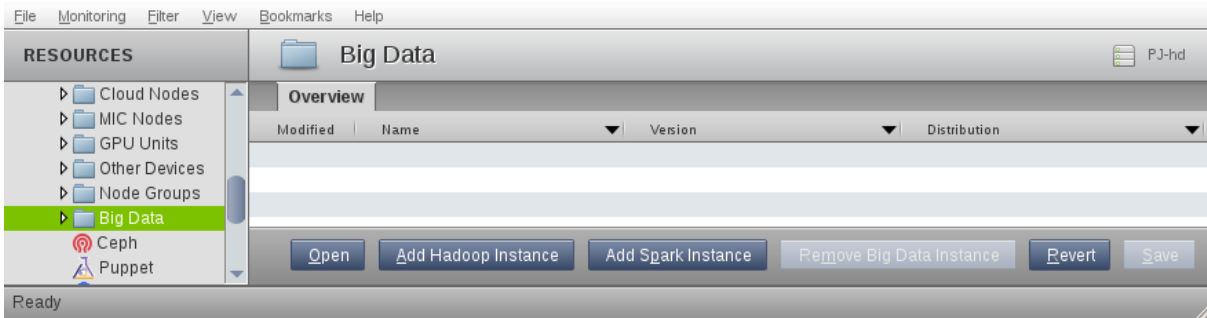


Figure 2.3: Installing Hadoop In cmgui

The wizard (figure 2.4) starts off with a page that allows the instance name and root points to be set. Notes are included to help make choices.

Hadoop main details

Please set the name of the instance and the Java home directory. Also set the root directory for where Hadoop stores its data. By default it is set to:
`/var/lib/hadoop/{instanceName}/`

Multiple root directories can be set using a comma to separate them.
 Example: `'/data1,/data2'`

Instance name:

Java home: ☒ Use default

Generic data root:

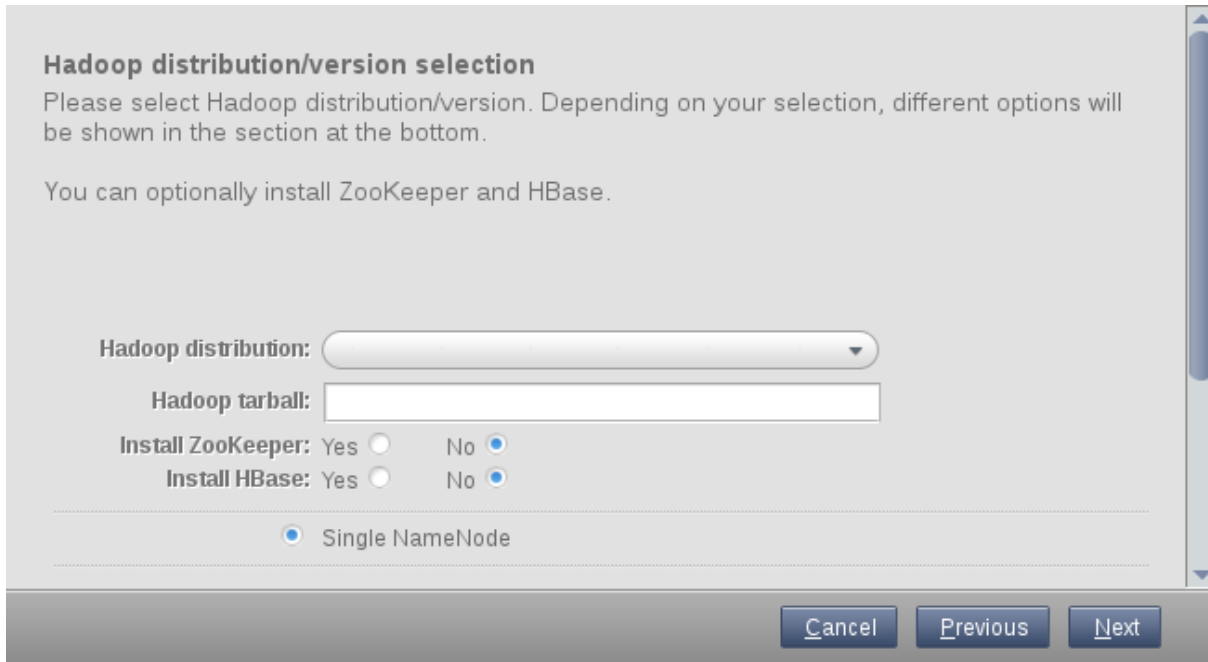
NameNode and DataNodes will use by default the 'Generic data root'. However, it's possible to specify dedicated directories for them.
 Example: `'/datann1,/datann2'`

NameNode data root:

DataNode data root:

Figure 2.4: Installing Hadoop In cmgui: Main Details Page

The next screen (figure 2.5) allows a Hadoop version to be chosen.



Hadoop distribution/version selection

Please select Hadoop distribution/version. Depending on your selection, different options will be shown in the section at the bottom.

You can optionally install ZooKeeper and HBase.

Hadoop distribution:

Hadoop tarball:

Install ZooKeeper: Yes ☐ No ☒

Install HBase: Yes ☐ No ☒

☒ Single NameNode

Figure 2.5: Installing Hadoop In cmgui: Version, Tarball, Related Extras Page

The administrator should decide whether to install Zookeeper or HBase. If HBase is used, then Zookeeper must also be running. The corresponding tabs for these Zookeeper and HBase become available from cmgui (section 3.1.5 and 3.1.6) when it is run later.

Depending on the selected Hadoop distribution, and whether it is based on Hadoop 1.x (2 possibilities) or Hadoop 2.x (10 possibilities), the administrator sees either:

- only one option, `Single NameNode` (figure 2.5)
- several installation options for Hadoop 2.x-based distros (figure 2.6)

Hadoop distribution/version selection

Please select Hadoop distribution/version. Depending on your selection, different options will be shown in the section at the bottom.

You can optionally install ZooKeeper and HBase.

Hadoop distribution: Cloudera CDH 5.3.x (based on Hadoop 2.5.0)

Hadoop tarball: /cm/local/apps/hadoop/hadoop-2.5.0-cdh5.3.9.tar.gz

Install ZooKeeper: Yes ☐ No ☒

Install HBase: Yes ☐ No ☒

☒ Single NameNode
☐ NameNode HA (manual failover)
☐ NameNode HA (automatic failover)
☐ NameNode federation
☐ YARN HA (automatic failover)

Cancel Previous Next

Figure 2.6: Installing Hadoop In cmgui: Distribution/Version Selection Page

The options are as follows:

- **Single NameNode:** This corresponds to regular HDFS, with no High Availability. This is the option for Hadoop 1.x-based distributions as well an option for Hadoop 2.x-based distributions that are configured to run like Hadoop 1.x distributions.

NameNode can optionally have a SecondaryNameNode, which is configurable in the next screen of the wizard.

A 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

The items listed next are only for Hadoop 2.x versions:

- **NameNode High Availability (manual failover):** This provides HDFS High Availability 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 High Availability (automatic failover):** This provides HDFS High Availability with automatic failover¹ It is as for the manual case, except that in this case ZooKeeper

¹

<http://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-hdfs/>

manages failover too. Which NameNode is active and which is on standby is therefore decided automatically.

- **NameNode Federation:** In NameNode Federation, 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.

Federation is described further at <http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/Federation.html>.

- **YARN HA (automatic failover:** This option is available only for more recent Hadoop 2.x distributions, for example
 - Apache Hadoop 2.7.x
 - Cloudera CDH 5.3.x and higher
 - Hortonworks HDP 2.1 and higher
 - Pivotal HD 3.0.x

This option also requires Zookeeper to be installed.

Having chosen the Hadoop version and required associated software, the next screens are service allocation screens. These screens allow the administrator to define which nodes (head nodes or regular nodes) are allocated to Hadoop and associated services. Suggestions are given and some restrictions are enforced.

The allocations can be made to the following host types:

- JournalNode
- JobTracker NameNode
- YARN server
- Key Management server
- DataNode
- Zookeeper
- HBase

[HDFSHighAvailabilityWithQJM.html#Automatic_Failover](#) explains the difference between manual and automatic failovers. From a Bright Cluster Manager perspective, manual failover requires journalnode services. Hence, if selected, journalnode roles will be assigned to some nodes. Automatic failover also relies on journalnodes and Zookeeper. Hence if automatic failover is chosen then Zookeeper is marked for installation.

The roles associated with these services are then assigned to these nodes. Some notes about the allocation choices that can be made:

- If YARN is chosen, then the administrator has to define two nodes for YARN ResourceManagers.
- DataNodes are configured explicitly by the administrator too. DataNodes are automatically coupled with MapReduce TaskTrackers for Hadoop 1.x-based distributions, and with their successor, YARN NodeManagers for Hadoop 2.x-based distributions.
- The wizard groups TaskTrackers/NodeManagers with the DataNode service, so that there is no dedicated selection window for TaskTrackers/NodeManagers. After installation, TaskTrackers/NodeManagers can be de-coupled from DataNodes if needed.
- If the chosen distribution is Cloudera CDH 5.3 or higher, or HDP 2.2 or higher, then an administrator can choose to install Key Management Server.

After nodes have been allocated to the various Hadoop components, the screen that follows is the summary page (figure 2.7).

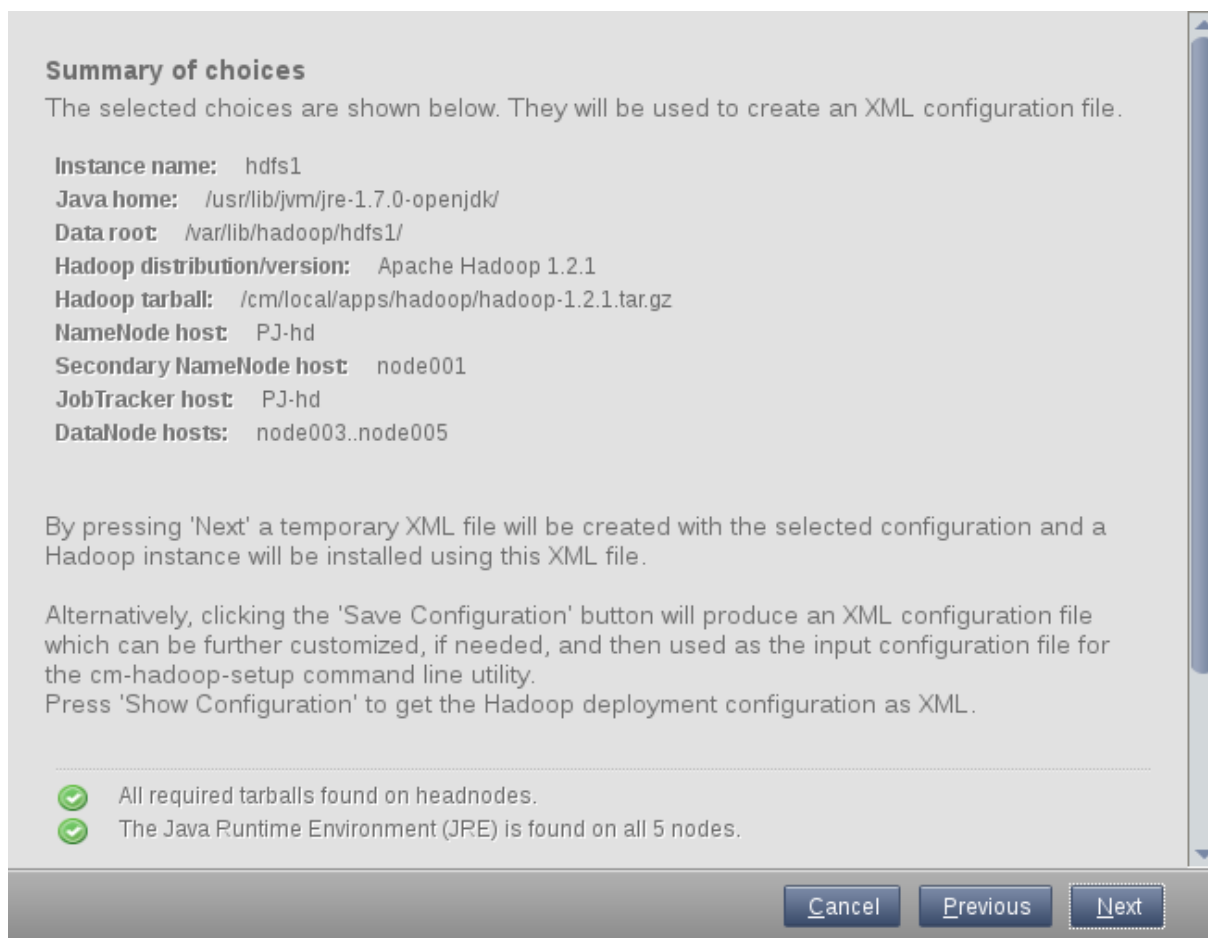


Figure 2.7: Installing Hadoop In cmgui: Wizard Summary Page

The summary shows the configuration settings, and is the final page in the wizard before installation. It does some validation to check all the required tarballs are in place, and that the JRE is available on all necessary nodes.

The summary page also allows the configuration to be saved as an XML file. The saved XML file can be further customized if needed, and a saved XML configuration can be reloaded in the first page of the wizard using the Open button (figure 2.4).

After the configuration has been checked by the administrator, the installation of the instance can be carried out. Installation progress is displayed, and the `Finish` button becomes active when installation is complete (figure 2.8):

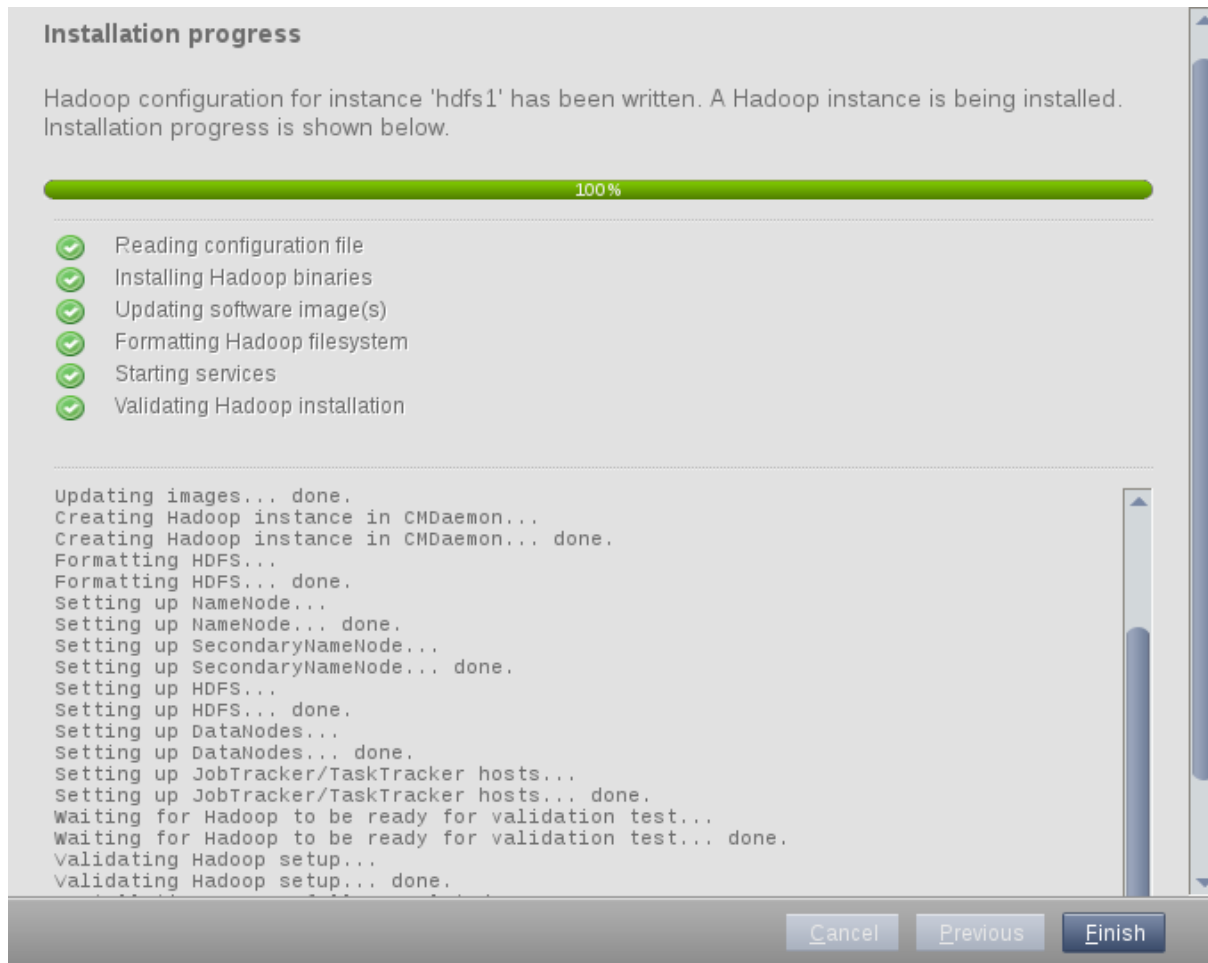


Figure 2.8: Installing Hadoop In cmgui: Wizard Installation Progress Page

Clicking on the `Finish` button ends the wizard.

2.4 Installing Hadoop With Lustre

The Lustre filesystem has a client-server configuration. Its installation on Bright Cluster Manager is covered in section 7.7 of the *Installation Manual*.

2.4.1 Lustre Internal Server Installation

The procedure for installing a Lustre server within Bright Cluster Manager varies. It is covered in section 7.7.3 of the *Installation Manual*.

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, hive`
- **groups:** `hadoop, zookeeper, hbase, hive`

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 With HAL

The HAL (Hadoop Adapter for Lustre) plugin allows Hadoop to use Lustre as a replacement for HDFS.

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 vanilla Apache and Cloudera CDH can both run with Lustre under Bright Cluster Manager. The configuration for these can be done as follows:

- 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/Cloudera/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 2.9).

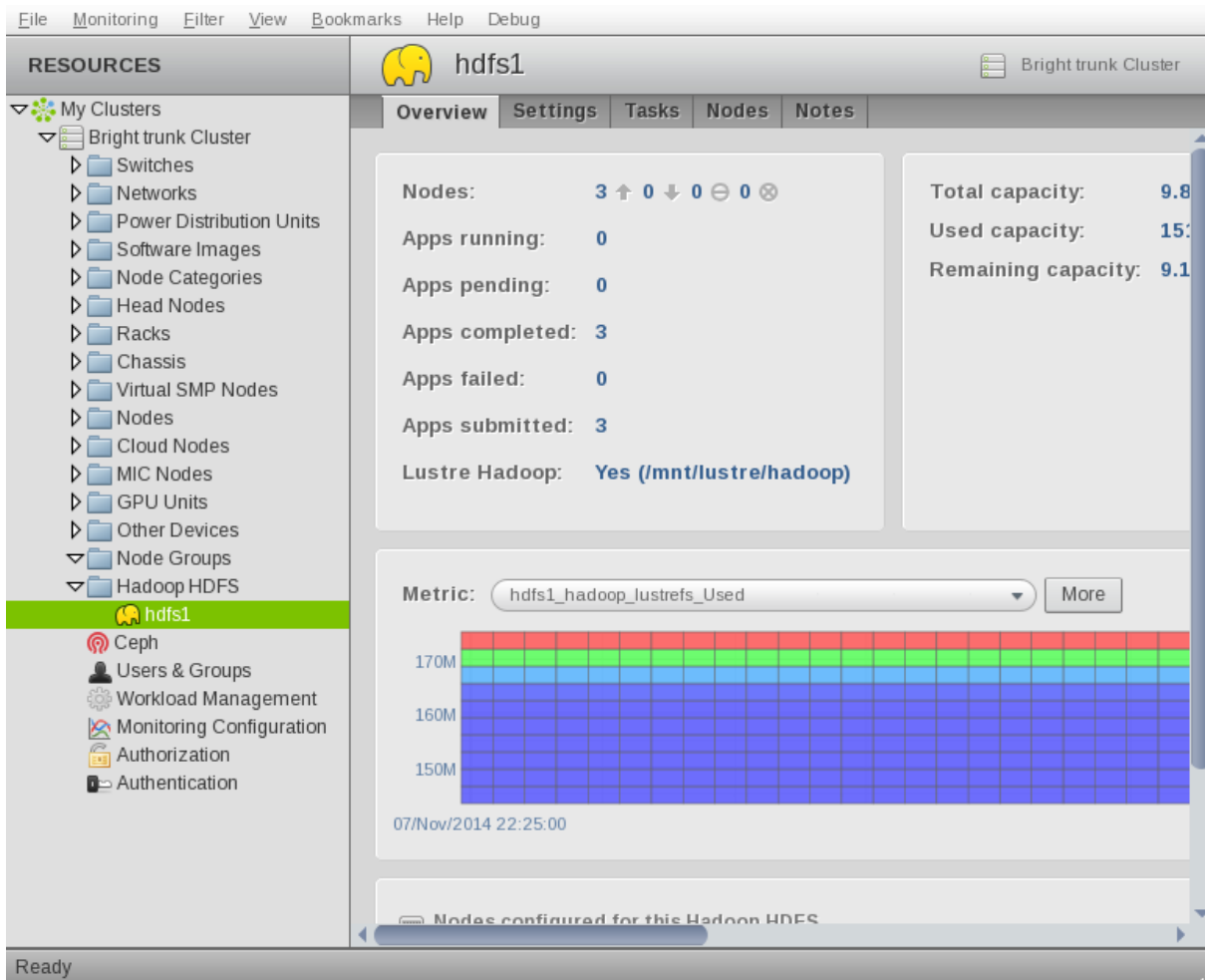


Figure 2.9: A Hadoop Instance With Lustre In cmgui

Installation Of Additional Tools

Sections 2.1 and 2.2 cover the the installation of Hadoop with a minimal configuration. Support for ZooKeeper, HBase, and additional tools such as Hive and Spark depends upon the Hadoop distribution and version. The version support matrix (section 1.4), and the appropriate sections in chapter 6 describe installation of the additional components.

2.4.5 Lustre Hadoop Configuration With HAL And HAM

In addition to the HAL plugin, also the HAM (Hadoop Apapter for MapReduce) can be deployed. It will allow to use Slurm as a replacement for YARN. Hadoop jobs will be run on all the nodes with Slurm Client role. In addition to the instructions in section 2.4.4, the following configurations must be taken care of.

Lustre Hadoop XML Configuration File Setup

Hadoop integration can be configured by using the file `/cm/local/apps/cluster-tools/hadoop/conf/hadoop2lustreslurmconf.xml` as a starting point for the configuration. It can be copied over to, for example, `/root/hadooplustreslurmconfig.xml`.

The configuration for these can be done as follows:

- The binary path for the Slurm installation must be specified in the `hadoop2lustreslurmconf.xml` file within the `<wlm></wlm>` tag pair:

- In addition, an `<adapterjar></adapterjar>` tag pair must be specified manually for the jar that the Intel IEEL 2.x distribution provides:

Example

```
<wlm>
  <name>slurm</name>
  <binpath>/cm/shared/apps/slurm/14.11.6/bin</binpath>
  <adapterjar>/root/lustre/hadoop-hpc-scheduler-3.1.0-ieel-2.2.jar</adapterjar>
</wlm>
```

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

- The `<datanodes></datanodes>` tag pair should include (in `<hosts></hosts>`) the list of Slurm Clients
- The `<yarnserver></yarnserver>` tag pair should include (in `<host></host>`) one of the Slurm Clients, which will be used as “edge node” for the Hadoop installation.

2.5 Installation of other Hadoop components

Bright Cluster Manager supports a number of popular systems relying on the main Hadoop framework. Section 6 overviews some of these systems and describes their installation/de-installation procedures.

2.6 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

Hadoop Cluster Management

The basic management of a Hadoop cluster using `cmgui`, `cmsh`, and the command line, is described in this chapter. Further details are given in Appendix A.

3.1 Managing A Hadoop Instance With `cmgui`

In `cmgui`, the `Hadoop instances` folder in the resource tree opens up to display the Hadoop instances running on the cluster (figure 2.1). Clicking on a Hadoop instance makes the tabs associated with Hadoop data management accessible (figure 3.1).

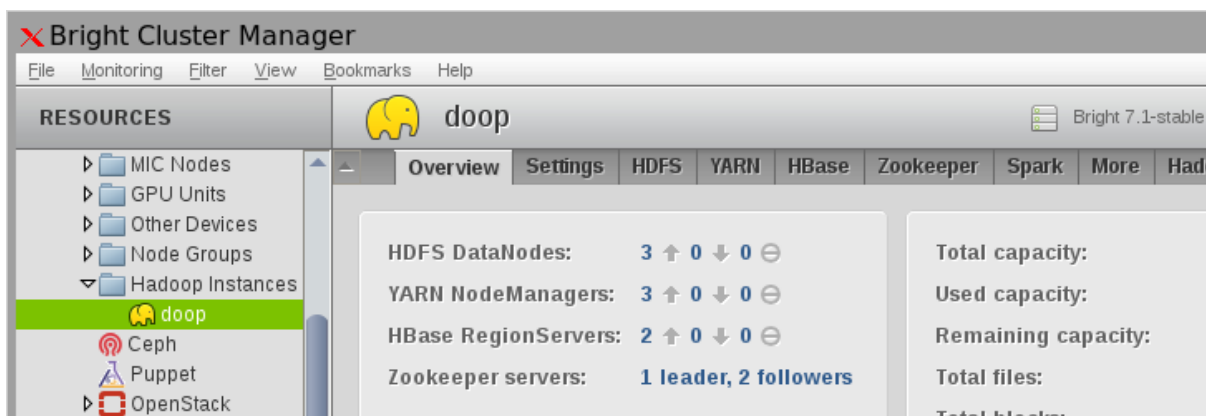


Figure 3.1: Tabs For A Hadoop Instance In `cmgui`

The following Hadoop tabs are described within this section:

1. Overview Tab (section 3.1.1)
2. Settings Tab (section 3.1.2)
3. HDFS Tab (section 3.1.3)
4. MapReduce or YARN Tab (section 3.1.4)
5. HBase Tab (section 3.1.5)
6. Zookeeper Tab (section 3.1.6)
7. Spark Tab (section 3.1.7)
8. More Tab (section 3.1.8)
9. Hadoop Configuration Groups Tab (section 3.1.9)

10. Monitoring Tab (section 3.1.10)

11. Notes Tab (section 3.1.11)

Not all of these tabs are necessarily displayed, depending on the software installed.

For example, if a user chooses to not install the HBase and Zookeeper components during the Hadoop installation procedure then the HBase and Zookeeper tabs are not displayed for this instance.

3.1.1 The HDFS Instance Overview Tab

The Overview tab pane (figure 3.2) aggregates the information about all Hadoop components and conveniently displays it in blocks.

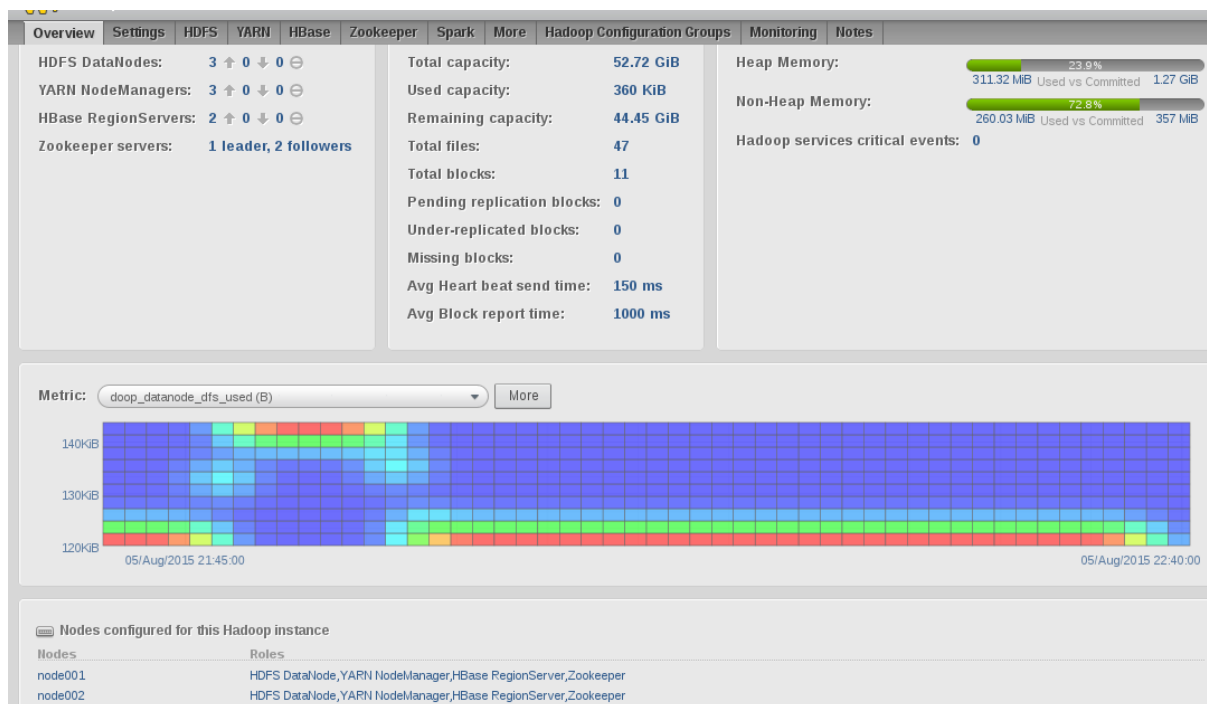


Figure 3.2: Overview Tab View For A Hadoop Instance In cmgui

The following items can be viewed:

- **Statistics:** In the top block there are statistics associated with the Hadoop instance. These include numbers of live/dead/decommissioned Hadoop services running on a cluster, as well as memory and capacity usage.
- **Metrics:** In the middle block, a metric can be selected for display as a heat map.
 - The `More` button allows other Hadoop-related metrics to be monitored in a somewhat similar way to the monitoring visualization system (section 10.3 of the *Administrator Manual*). The extra Hadoop-related metrics that can then be viewed are organized in subtabs, and further views of selected nodes can be added with the `Node details` button.
- **Roles:** The third block displays the Hadoop/Spark roles associated with each node used by this Hadoop instance.

3.1.2 The HDFS Instance Settings Tab

The Settings tab pane (Figure 3.3) presents the general details about Hadoop instance installation and configuration and allows a user to configure a number of HDFS-related parameters.

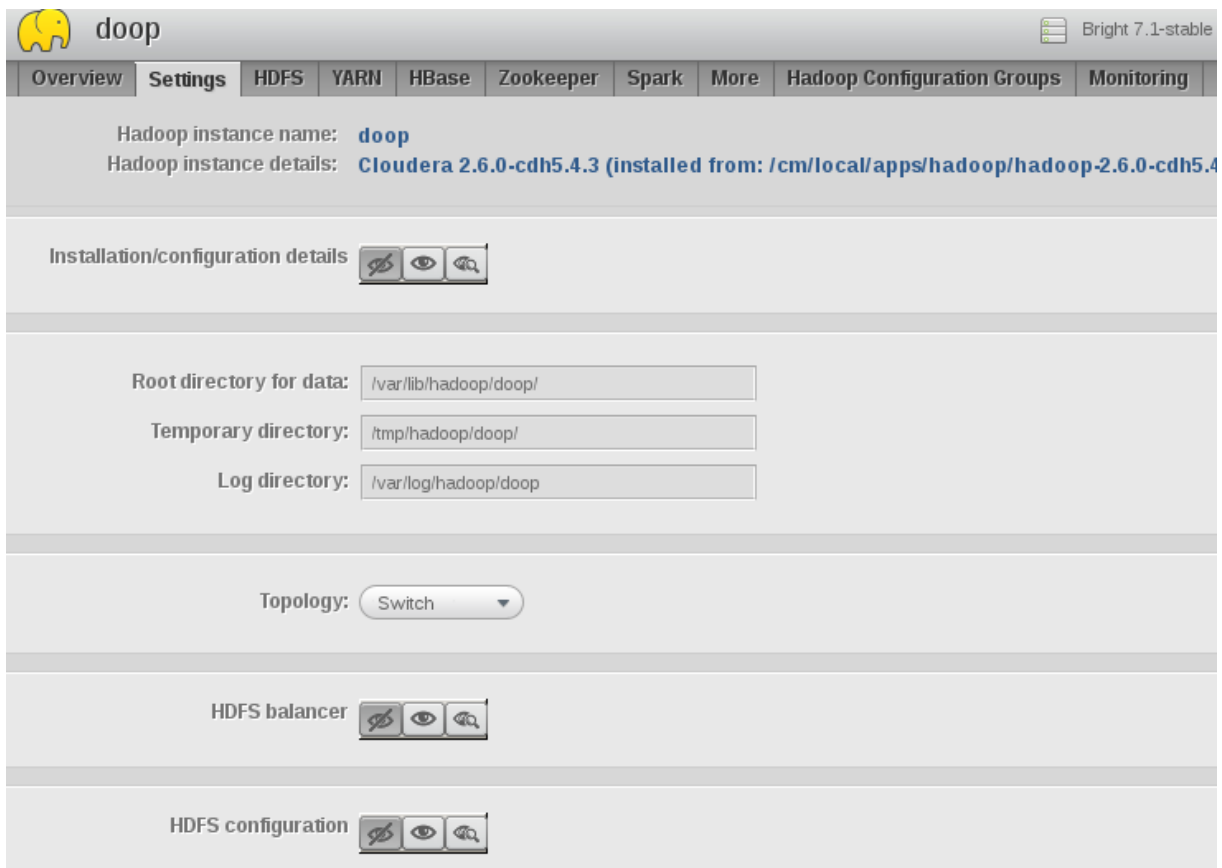


Figure 3.3: Settings Tab View For A Hadoop Instance In cmgui

Major details about Hadoop installation, such as the locations of Hadoop components or temporary files placement, can be viewed, but cannot be changed.

The remaining parameters (figure 3.4) can be viewed and changed. These are:

Topology: Switch

None
Switch
Rack

HDFS balancer

Balancer period: 48 hours Balancer threshold: 10 Balancer policy: datanode

HDFS configuration

HDFS default block size: 134217728 bytes I/O buffer size: 65536 bytes ☒ HDFS permissions enabled

HDFS default replication factor: 3 First block report delay: 10 s ☐ HTTPS for web UIs enabled

HDFS maximum replication factor: 50 HDFS Umask: 022 ☐ WebHDFS enabled

Compression codec: +

Serialization classes:

- org.apache.hadoop.io.serializer
- org.apache.hadoop.io.serializer
- org.apache.hadoop.io.serializer

Figure 3.4: Settings Tab View Details For A Hadoop Instance In cmgui

- **Topology:** Hadoop can be made aware of a cluster topology so that HDFS data replication is done more efficiently. Topology options are:
 - none: No topology-based optimization is set.
 - Switch: HDFS datanodes become switch-aware, which allows HDFS to minimize data access between switches.
 - Rack: HDFS datanodes become rack-aware to minimize data access between racks.
- **HDFS balancer:** Configuration values used for HDFS balancing (i.e., moving data blocks from over-utilized to under-utilized nodes). The parameters are:
 - Balancer period: Sets the period in hours between balancing operations.
 - Balancer threshold: Defines the maximum difference (in %) between the percentage of disk usage on any given DataNode and the average percentage of disk usage across all DataNodes.
 - Balancer policy: Sets a balancing policy.
 - * blockpool: Balancing is done at the block pool level.
 - * datanode: (default) Balances the storage at the DataNode level.
- **HDFS configuration:** Global settings for HDFS filesystem including the following parameters:
 - HDFS default block size
 - HDFS default replication factor
 - HDFS maximum replication factor
 - I/O buffer size
 - First block report delay
 - HDFS Umask
 - HDFS permissions enabled
 - HTTPS for web UIs enabled
 - WebHDFS enabled
 - ...

3.1.3 The HDFS Instance HDFS Tab

The HDFS tab as well as tabs displayed in sections 3.1.4-3.1.7 all follow a similar layout pattern. The pattern is: a block at the top overviews the resources of a corresponding Hadoop component and subtabs below, *Operations* and *Configuration*, allow a user to manage the resources of this component.

Specifically, the HDFS tab pane (figure 3.5), which focuses on the HDFS component, displays HDFS NameNode and DataNode activities at the top, and available HDFS-specific operations and configuration in the associated subtabs beneath. This is now elaborated upon next.



Figure 3.5: HDFS Tab For A Hadoop Instance In cmgui

For the HDFS tabbed pane, the top of the pane displays NameNode and DataNode JVM use, DataNodes' status information, DFS disk usage and some file/block-related total counts.

Below the main pane are the following two subtabs:

- An *Operations* subtab. This allows the following operations to be carried out with buttons:
 - HDFS: HDFS start, stop, and restart
 - (De-)commission: add and remove DataNodes from the overall DataNodes pool
 - HDFS Balancer: start or stop the HDFS balancer
 - Safemode: enter or leave safemode (i.e., a read-only mode for the HDFS).
 - Format: Format the HDFS filesystem
- A *Configuration* subtab. This provides a list of Hadoop configuration groups (section 3.1.9) that use the HDFS service and the roles associated with these configuration groups.

Hadoop configuration groups are discussed in the dedicated section 3.1.9.

Double-clicking on a configuration group, or clicking on the open button for a selected configuration group, opens up a configuration group editor window for that configuration group.

3.1.4 The HDFS Instance MapReduce Or YARN Tab

The next tab in the row of Hadoop tabs in figure 3.1, after the HDFS tab, is:

- either the MapReduce tab (figure 3.6), as used in older Hadoop distributions such as Apache Hadoop 1.2.1
- or the YARN tab (figure 3.7) for more recent distributions



Figure 3.6: MapReduce Tab For A Hadoop Instance In cmgui

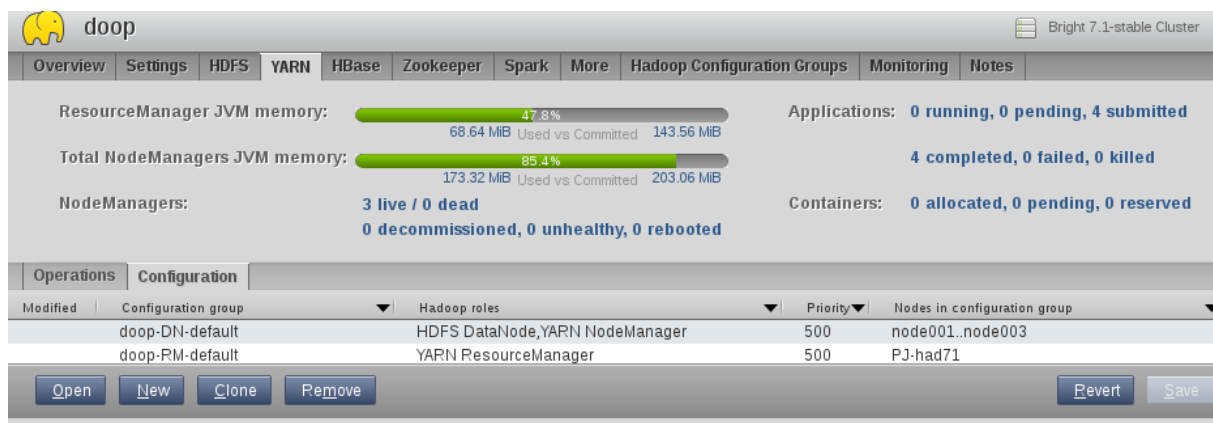


Figure 3.7: YARN Tab For A Hadoop Instance In cmgui

The MapReduce or YARN tab pane follow the pattern of sections 3.1.3-3.1.7. That is: the top block of the pane tracks job/application execution resources, while the two subtabs below the top block, Operations and Configuration, allow a user to perform operations on MapReduce (or YARN) and to configure MapReduce (or YARN) components via corresponding configuration groups.

The following operations can be performed in the Operations subtab:

- MapReduce or YARN: MapReduce (or YARN) start, stop, and restart
- (De-) commission: add and remove TaskTrackers (or NodeManagers) from the overall TaskTrackers (NodeManagers) pool

3.1.5 The HDFS Instance HBase Tab

In the HBase tab pane (figure 3.8), the pattern of sections 3.1.3-3.1.7 is followed. Thus, the top block tracks HBase resources, while the subtabs below it allow a user to perform HBase operations, or allow configuration of nodes via the HBase-related configuration groups.

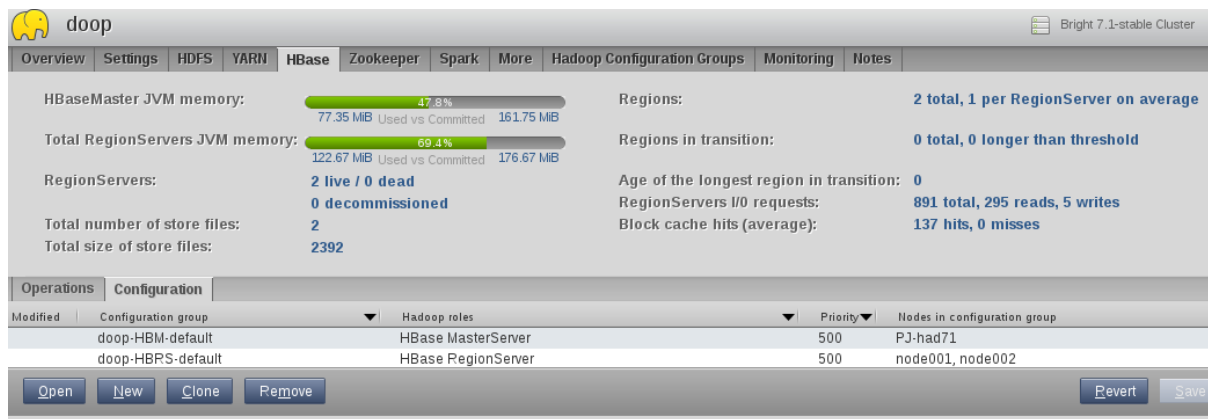


Figure 3.8: HBase Tab For A Hadoop Instance In cmgui

3.1.6 The HDFS Instance Zookeeper Tab

In the Zookeeper tab pane (figure 3.9), following the pattern of sections 3.1.3-3.1.7, the top block tracks Zookeeper resources, while the subtabs below it allow a user to perform Zookeeper operations, or allow configuration of nodes via the Zookeeper-related configuration groups.

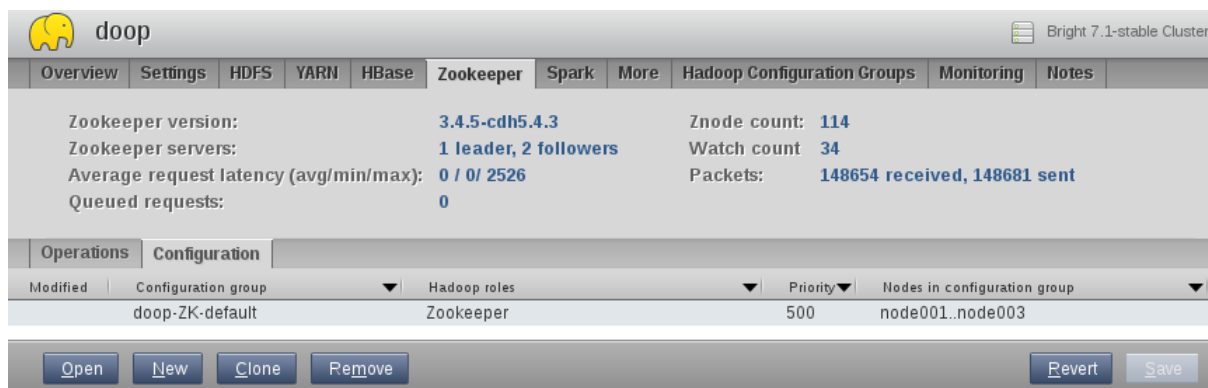


Figure 3.9: Zookeeper Tab For A Hadoop Instance In cmgui

3.1.7 The HDFS Instance Spark Tab

The Spark tab pane appears if Spark (Chapter 5) has been installed. The tab follows the common pattern of Sections 3.1.3-3.1.7.

3.1.8 The HDFS Instance More Tab

The More tab pane is reserved for future use for the Hive and Sqoop projects.

3.1.9 The HDFS Instance Hadoop Configuration Groups Tab

While the main Hadoop Configuration Groups tab shows all of the Hadoop configuration groups for the Hadoop instance, the Configuration sub-tabs described earlier in sections 3.1.3-3.1.7 show only those Hadoop configuration groups that have at least one role related to a corresponding component. Modifications done in main tab, or modifications done in one of the sub-tabs in sections 3.1.3-3.1.7, are automatically synchronized with each other.

Configuration Overlays And Hadoop Configuration Groups

In Bright Cluster Manager, a Hadoop configuration group is a special Hadoop case of the general Bright Cluster Manager concept of a configuration overlay.

- A *configuration overlay* assigns roles (section 2.1.5 of the *Administrator Manual*) for groups of nodes. The number of roles can be quite large, and priorities can be set for these.

Multiple configuration overlays can be set for a node. A priority can be set for each configuration overlay, so that a configuration overlay with a higher priority is applied to its associated node instead of a configuration overlay with a lower priority. The configuration overlay with the highest priority then determines the actual assigned role.

- A *Hadoop configuration group* is a configuration overlay that assigns a group of roles to a Hadoop instance. Thus, when the Hadoop configuration group overlays the Hadoop instance, then roles are assigned to nodes according to the configuration, along with a priority. Whether the Hadoop configuration group assignment is used, or whether the original role assignment is used, depends upon the configured priorities.

Configuration overlays can take on priorities in the range 0-1000, except for 250 and 750, which are forbidden. Setting a priority of -1 means that the configuration group is ignored.

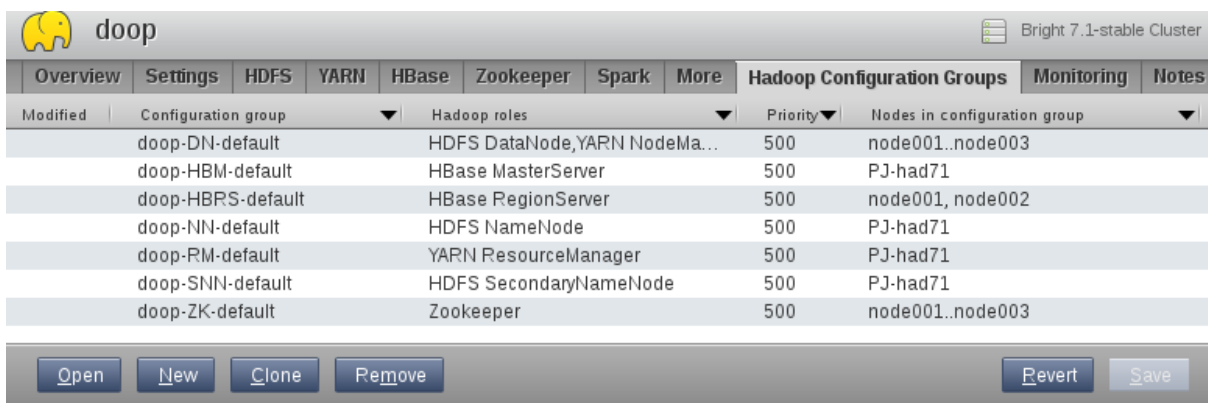
The priorities of 250, 500, and 750 are also special, as indicated by the following table:

| priority | assigned to node from |
|----------|---|
| -1 | configuration group not assigned |
| 250 | category |
| 500 | configuration overlay with default priority |
| 750 | node |

Roles assigned at category level have a fixed priority of 250, while roles assigned at node level have a fixed priority of 750. The configuration overlay priority is variable, but is set to 500 by default. Thus, for example, roles assigned at the node level override roles assigned at the category level. Roles assigned at the node level also override roles assigned by the default configuration overlay.

Display And Management Of Hadoop Configuration Groups Within Hadoop Tab Of cmgui

The Hadoop Configuration Groups tab pane (figure 3.10) displays a list of all the configuration groups used by the Hadoop instance.



| Modified | Configuration group | Hadoop roles | Priority | Nodes in configuration group |
|----------|---------------------|------------------------------|----------|------------------------------|
| | doop-DN-default | HDFS DataNode,YARN NodeMa... | 500 | node001..node003 |
| | doop-HBM-default | HBase MasterServer | 500 | PJ-had71 |
| | doop-HBRS-default | HBase RegionServer | 500 | node001, node002 |
| | doop-NN-default | HDFS NameNode | 500 | PJ-had71 |
| | doop-RM-default | YARN ResourceManager | 500 | PJ-had71 |
| | doop-SNN-default | HDFS SecondaryNameNode | 500 | PJ-had71 |
| | doop-ZK-default | Zookeeper | 500 | node001..node003 |

Figure 3.10: Hadoop Configuration Groups Tab For A Hadoop Instance In cmgui

The names of the configuration groups take the following form by default:

<hadoop instance name>-<role abbreviation>-default

Example

doop-DN-default

Hadoop/Spark Roles: The role abbreviations used are indicated by the following table of roles available under Hadoop:

| role | abbreviation | cmsh role |
|-----------------------------------|-----------------|---------------------------|
| DataNode | DN | Hadoop::DataNode |
| HBase MasterServer* | HBM | Hadoop::HBaseServer |
| HBase RegionServer | HBRS | Hadoop::HBaseClient |
| Hive | HV | Hadoop::Hive |
| JournalNode | JN | Hadoop::Journal |
| JobTracker ¹ | JT | Hadoop::JobTracker |
| Key Management Server | KM | Hadoop::KMServer |
| HDFS NFS Gateway | NFS | Hadoop::NFSGateway |
| NameNode* | NN [†] | Hadoop::NameNode |
| YARN ResourceManager ² | RM | Hadoop::YARNServer |
| SecondaryNameNode | SNN | Hadoop::SecondaryNameNode |
| Spark YARN | SY | Hadoop::SparkYARN |
| Spark Master* | SM | Hadoop::SparkMaster |
| Spark Worker | SW | Hadoop::SparkWorker |
| Sqoop | SQ | Hadoop::Sqoop |
| ZooKeeper | ZK | Hadoop::ZooKeeper |

* If these are in use, then modifying them should be done with great care due to the dependency of other roles on them

¹ for Hadoop v1

² for Hadoop v2

[†] Becomes NN1 and NN2 with high-availability

Each configuration group in figure 3.10 can be double-clicked in order to configure the group and their underlying role or roles. Double-clicking or using the `Open` button on a group, for example `doop-DN-default` in the figure, opens up an editor window within which the priority of the configuration group can be adjusted and other parameters of the underlying roles can also be adjusted, from within role subtabs (figures 3.11 and 3.12).

The screenshot shows a window titled "Edit Hadoop Configuration Group". At the top, there are fields for "Configuration Group" (set to "doop-DN-default"), "Priority" (set to "500"), and "Nodes in Configuration Group" (set to "node001..node003"). An "Add/remove nodes" button is next to the nodes field. Below these fields are two tabs: "Configure HDFS DataNode" (selected) and "Configure YARN NodeManager". The "Configure HDFS DataNode" tab shows settings applied to "node001..node003". On the left, there are fields for "Data directories:" (set to "/var/lib/hadoop/doop/hadoop/hd"), "Number of failed volumes tolerated:" (set to "0"), "Reserved space for Non DFS use:" (set to "1073741824" byte), and "Bandwidth for balancer:" (set to "1048576" byte/sec). On the right, there are fields for "DataNode Java heap size:" (set to "512" MB), "DataNode ports" (with icons for edit, view, and refresh), "IPC port:" (set to "50020"), "HTTP port:" (set to "50075"), "HTTPS port:" (set to "50475"), "Transceiver port:" (set to "50010"), "More DataNode parameters" (with icons for edit, view, and refresh), "Handler count:" (set to "10"), "Max number of transfer threads:" (set to "4096"), and "Heartbeat interval:" (set to "3" sec). At the bottom, there are buttons for "Add role", "Remove role", "Cancel", and "Ok".

Figure 3.11: Hadoop Configuration Groups Tab, After Opening DataNode Configuration Group: DataNode Role Configuration

Edit Hadoop Configuration Group

Configuration Group:

Priority:

Nodes in Configuration Group: **Add/remove nodes**

Configure HDFS DataNode | **Configure YARN NodeManager**

Settings applied to:

Localization directories: **+**

Log directories: **+**

Log aggregation enabled: ☐

Application log directory:

Application log directory suffix:

Log retain time: sec

More NodeManager parameters:

Docker container execution:

Shuffle service name:

Shuffle class:

ShuffleHandler port:

NodeManager heap size: MB

Container manager:

Container manager port:

Containers memory: MB

Physical memory limits enforced: ☒

Virtual memory limits enforced: ☒

Virtual/physical memory ratio:

Vcores capacity:

Monitoring interval: msec

Handler count:

NodeManager ports:

ApplicationMaster:

MapReduce jobs:

Add role **Remove role** **Cancel** **Ok**

Figure 3.12: Hadoop Configuration Groups Tab, After Opening DataNode Configuration Group: YARN NodeManager Role Configuration

There is a great deal of flexibility in dealing with configuration groups and roles. Configuration groups can be created, cloned, and removed using the buttons in figure 3.10, while roles that have been opened for editing can not only be modified, but also added or removed.

However, it should be noted that the asterisked roles in the preceding table are roles that other roles can depend upon. Modifying them should therefore only be done with extreme care. It is not difficult to misconfigure the Hadoop NameNode role so that it leads to the HDFS filesystem becoming unavailable, and hence to potential data loss.

Subsets of the configuration groups of figure 3.10 are displayed in the individual service resource tabs, such in the HDFS or Zookeeper resource tabs, under their individual (HDFS or Zookeeper) Configuration subtab. The subsets displayed are the ones associated with the resource.

For example: In the Hadoop Configuration Groups tab (figure 3.10) all the configuration groups are shown. On the other hand, in the HBase tab (figure 3.8) only the subset of HBase-related configuration groups are shown.

Double-clicking or using the Open button on a configuration group within a subset also opens up an editor window for the configuration group just as in figure 3.10.

Further roles can be assigned within the editor window by clicking on the Add Role button.

3.1.10 The HDFS Instance Monitoring Tab

The Monitoring tab pane (figure 3.13), displays metrics related to Hadoop monitoring.

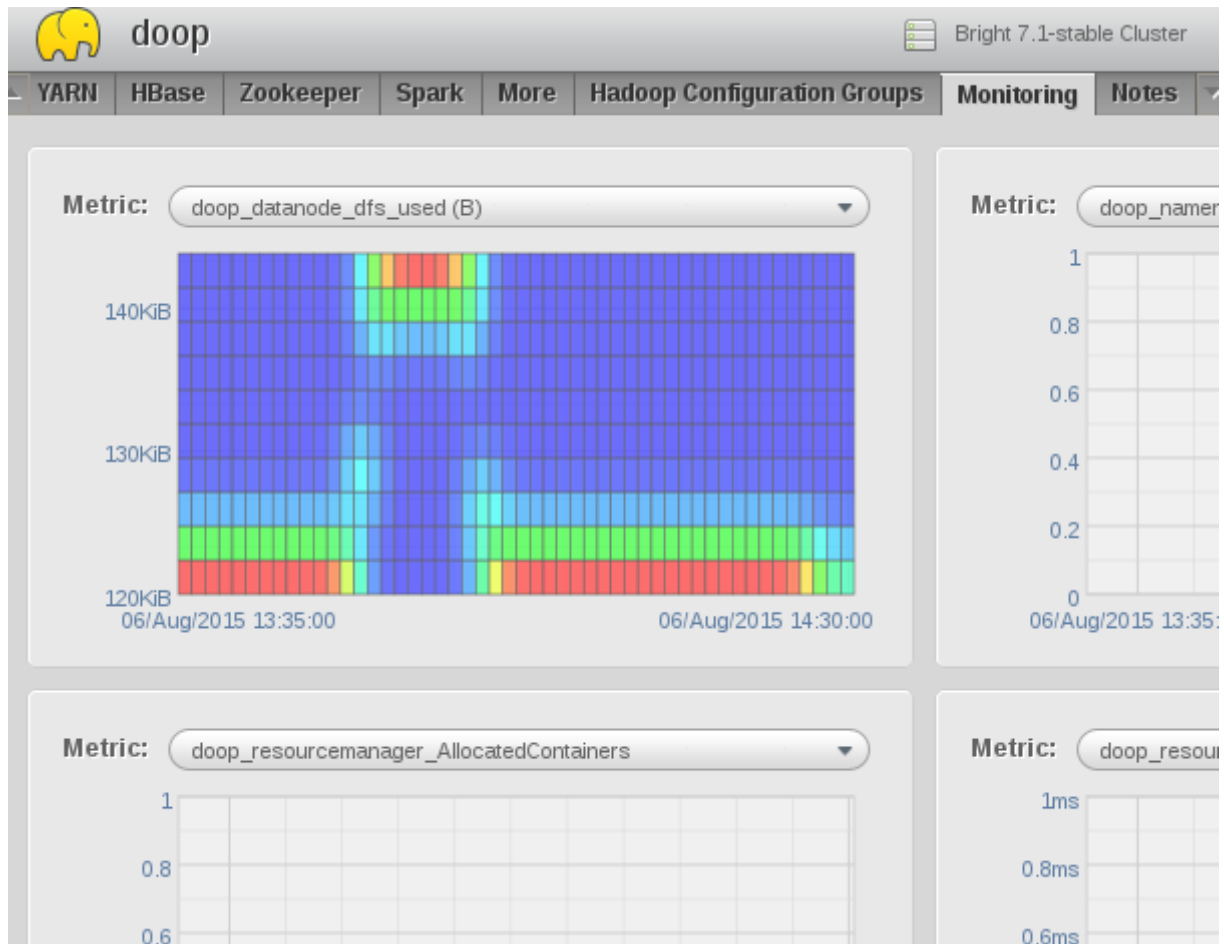


Figure 3.13: Monitoring Tab For A Hadoop Instance In cmgui

3.1.11 The HDFS Instance Notes Tab

This tab provides a simple notepad for the administrator for each Hadoop instance.

3.2 Managing A Hadoop Instance With cmsh

3.2.1 cmsh And hadoop Mode

The cmsh front end uses the hadoop mode to display information on Hadoop-related values and to carry out Hadoop-related tasks.

Example

```
[root@bright72 conf]# cmsh
[bright72]% hadoop
[bright72->hadoop]%
```

The show And overview Commands

The overview Command: Within hadoop mode, the overview command displays two sections of interest, that correspond somewhat to cmgui's Overview tab in the Hadoop resource (section 3.1.1), providing Hadoop-related information on the system resources that are used. The first section gives an overview of the cluster state with regard to Hadoop usage. The second section shows the Hadoop role node assignments along with the configuration groups that the roles are associated with.

Example

```
[bright72->hadoop]% overview doop
```

| Parameter | Value |
|--------------------------------|---------|
| ----- | |
| Name | doop |
| Capacity total | 52.72GB |
| Capacity used | 282.3MB |
| Capacity remaining | 43.97GB |
| Heap memory total | 1.279GB |
| Heap memory used | 305.5MB |
| Heap memory remaining | 1004MB |
| Non-heap memory total | 348MB |
| Non-heap memory used | 252.3MB |
| Non-heap memory remaining | 95.72MB |
| Nodes available | 3 |
| Nodes dead | 0 |
| Nodes decommissioned | 0 |
| Nodes decommission in progress | 0 |
| Total files | 52 |
| Total blocks | 15 |
| Missing blocks | 0 |
| Under-replicated blocks | 0 |
| Scheduled replication blocks | 0 |
| Pending replication blocks | 0 |
| Block report average Time | 1000 |
| Applications running | 0 |
| Applications pending | 0 |
| Applications submitted | 0 |
| Applications completed | 0 |
| Applications failed | 0 |
| Federation setup | no |

| Hadoop role | Nodes | Configuration group | Nodes up |
|---------------------------|------------------|---------------------|----------|
| ----- | | | |
| Hadoop::DataNode | node001..node003 | doop-DN-default | 3 of 3 |
| Hadoop::HBaseClient | node001,node002 | doop-HBRS-default | 2 of 2 |
| Hadoop::HBaseServer | bright72 | doop-HBM-default | 1 of 1 |
| Hadoop::NameNode | bright72 | doop-NN-default | 1 of 1 |
| Hadoop::SecondaryNameNode | bright72 | doop-SNN-default | 1 of 1 |
| Hadoop::SparkYARN | bright72 | doop-SY-default | 1 of 1 |
| Hadoop::YARNClient | node001..node003 | doop-DN-default | 3 of 3 |
| Hadoop::YARNServer | bright72 | doop-RM-default | 1 of 1 |
| Hadoop::ZooKeeper | node001..node003 | doop-ZK-default | 3 of 3 |

The show Command: The `show` command displays parameters that correspond mostly to the Settings tab of `cmgui` in the Hadoop resource (section 3.1.2):

Example

```
[bright72->hadoop[doop]]% show
```

| Parameter | Value |
|--------------------|----------|
| ----- | |
| Automatic failover | Disabled |
| Buffer size | 65536 |
| Cluster ID | |

| | |
|---|-------------------------|
| Compression codecs | |
| Configuration directory | /etc/hadoop/doop |
| Configuration directory for HBase | /etc/hadoop/doop/hbase |
| Configuration directory for Hive | |
| Configuration directory for Spark | /etc/hadoop/doop/spark |
| Configuration directory for Sqoop | |
| Configuration directory for ZooKeeper | /etc/hadoop/doop/zooke+ |
| Connection maximum idle time | 30000 |
| Creation time | Mon, 03 Aug 2015 16:56+ |
| Delay for first block report | 10 |
| Enable HA for YARN | no |
| Enable NFS gateway | no |
| Enable WebHDFS | no |
| HA enabled | no |
| HA name service | ha |
| HBase version | 1.0.0-cdh5.4.3 |
| HDFS Permission | yes |
| HDFS Umask | 022 |
| HDFS audit enabled | no |
| HDFS balancer period | 48 |
| HDFS balancer policy | dataNode |
| HDFS balancer threshold | 10 |
| HDFS default block size | 134217728 |
| HDFS default replication factor | 3 |
| HDFS maximum replication factor | 50 |
| Hadoop distribution | Cloudera |
| Hadoop root for Lustre | |
| Hadoop version | 2.6.0-cdh5.4.3 |
| Hive version | |
| Idle threshold number of connections | 4000 |
| Installation directory for HBase | /cm/shared/apps/hadoop+ |
| Installation directory for Hadoop instance | /cm/shared/apps/hadoop+ |
| Installation directory for Hive | |
| Installation directory for Spark | /cm/shared/apps/hadoop+ |
| Installation directory for Sqoop | |
| Installation directory for ZooKeeper | /cm/shared/apps/hadoop+ |
| Log directory for Hadoop instance | /var/log/hadoop/doop |
| Maximum number of retries for IPC connections | 30 |
| Name | doop |
| Network | |
| Readonly | no |
| Revision | |
| Root directory for data | /var/lib/hadoop/doop/ |
| Serialization classes | org.apache.hadoop.io.s+ |
| Spark version | |
| Sqoop version | |
| Temporary directory for Hadoop instance | /tmp/hadoop/doop/ |
| Topology | Switch |
| Use HTTPS | no |
| Use Lustre | no |
| Use federation | no |
| Use only HTTPS | no |
| Whether is a Spark instance | no |
| YARN automatic failover | Disabled |
| ZooKeeper version | 3.4.5-cdh5.4.3 |


```
description                               installed from: /cm/lo+
notes                                    notes here!
```

If setting or getting a value, then using the `set` or `get` command on its own within `hadoop` mode shows a help text that can help decide on what parameter should be set or gotten.

Example

```
[bright72->hadoop[doop]]% get
...
Parameters:
    Readonly ..... Mark data as readonly
    Revision ..... Entity revision
    automaticfailover ... Automatic failover can be controlled by\
                        either Hadoop itself of CMDaemon
    buffersize ..... Buffer size in I/O operations (in bytes\
                        ). Defined in io.file.buffer.size
    clusterid ..... Cluster ID for federation
    compressioncodecs ... Comma-separated list of compression cod\
                        ec classes. Defined in io.compression.codecs
    configurationdirectory Configuration directory
    configurationdirectoryforhbase Configuration directory for HBase
    configurationdirectoryforhive Configuration directory for Hive
    ...
```

The `*services` Commands For Hadoop Services

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

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

Example

```
[bright72->hadoop]% restartallservices apache121
Will now stop all Hadoop services for instance 'apache121'... done.
Will now start all Hadoop services for instance 'apache121'... done.
[bright72->hadoop]%
```

The `*balancer` Commands For Hadoop, And Related Parameters

For applications to have efficient access to HDFS, the file block level usage across nodes need to be reasonably balanced. The following balancer commands can be run from within `hadoop` mode:

- `startbalancer`: starts balancing
- `stopbalancer`: stops balancing
- `statusbalancer`: displays status of balancer

Example

```
[bright72->hadoop]% statusbalancer doop
Code: 1
Redirecting to /bin/systemctl status  hadoop-doop-balancer.service
hadoop-doop-balancer.service - Hadoop Balancer daemon for instance doop
    Loaded: loaded (/usr/lib/systemd/system/hadoop-doop-balancer.service
    static)
    Active: inactive (dead)
```

The following parameters:

- `hdfsbalancerperiod`
- `hdfsbalancerthreshold`
- `hdfsbalancerpolicy`

can be used to set or retrieve the period, threshold, and policy for the balancer running on the instance.

Example

```
[bright72->hadoop]% get doop hdfsbalancerperiod
2
[bright72->hadoop]% set doop balancerperiod 3
[bright72->hadoop*]% commit
[bright72->hadoop]% startbalancer doop
Code: 0
Starting Hadoop balancer daemon (hadoop-doop-balancer):starting ba\
lancer, logging to /var/log/hadoop/doop/hdfs/hadoop-hdfs-balancer-\
bright72.out
Time Stamp  Iteration#  Bytes Moved  Bytes To Move  Bytes Being Moved
The cluster is balanced. Exiting...
```

```
[bright72->hadoop]%
Thu Mar 20 15:27:02 2014 [notice] bright72: Started balancer for doop
For details type: events details 152727
```

The preceding Hadoop services and balancer commands run tasks and use parameters that correspond mostly to the HDFS tab (section 3.1.2) of `cmgui`.

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

```
[bright72->hadoop]% formathdfs doop
Will now format and set up HDFS for instance 'doop'.
Stopping datanodes... done.
Stopping namenodes... done.
Formatting HDFS... done.
Starting namenode (host 'bright72')... done.
Starting datanodes... done.
Waiting for datanodes to come up... done.
Setting up HDFS... done.
[bright72->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 within Hadoop distributions that support NameNode failover.

3.2.2 cmsh And configurationoverlay Mode

Hadoop configuration groups are introduced in section 3.1.9 as a special case of configuration overlays. Within cmgui Hadoop configuration groups can be accessed from within the tabs associated with a Hadoop instance (section 3.1.9).

Configuration Overlay Listing

In cmsh, the Hadoop configuration groups are listed and accessed via configurationoverlay mode:

Example

```
[root@bright72 ~]# cmsh
[bright72->configurationoverlay]% list -f name:17,nodes:16,roles:36
name (key)          nodes          roles
-----
doop-DN-default     node001..node003 Hadoop::DataNode, Hadoop::YARNClient
doop-HBM-default    bright72        Hadoop::HBaseServer
doop-HBRS-default   node001,node002 Hadoop::HBaseClient
doop-NN-default     bright72        Hadoop::NameNode
doop-RM-default     bright72        Hadoop::YARNServer
doop-SNN-default    bright72        Hadoop::SecondaryNameNode
doop-SY-default     bright72        Hadoop::SparkYARN
doop-ZK-default     node001..node003 Hadoop::ZooKeeper
```

Configuration Overlay Mode And Configuration Overlay Properties

A configuration overlay object can be used. That is, the shell can drop within a particular Hadoop configuration group with the use command. The properties of the object, that is the Hadoop configuration group, can then be shown:

Example

```
[bright72->configurationoverlay]% use doop-dn-default
[bright72->configurationoverlay[doop-DN-default]]% show
Parameter          Value
-----
Categories
Name                doop-DN-default
Nodes               node001..node003
Priority             500
Readonly            no
Revision
Roles               Hadoop::DataNode, Hadoop::YARNClient
```

Configuration Overlay Roles Submode, And Role Properties – All Instances

A roles submode can be entered within the configuration overlay object. That is, the Hadoop configuration group roles submode can be entered. The roles that the configuration overlay is associated with can be listed:

Example

```
[bright72->configurationoverlay[doop-DN-default]]% roles
[bright72->configurationoverlay[doop-DN-default]->roles]% list
Name (key)
-----
Hadoop::DataNode
Hadoop::YARNClient
```

A particular role can be used and its CMDaemon properties, relevant to all instances, viewed and modified:

```
...figurationoverlay[doop-DN-default]->roles]% use hadoop::datanode
...figurationoverlay[doop-DN-default]->roles[Hadoop::DataNode]]% show
Parameter                                     Value
-----
Configurations                               <1 in submode>
Name                                           Hadoop::DataNode
Provisioning associations                     <0 internally used>
Readonly                                       no
Revision
Type                                           HadoopDataNodeRole
```

Configuration Overlay Roles Submode, Role Properties – For A Selected Instance

Within a role, the configurations submode can be used to modify the properties of the role itself. The configuration list shows which instances are available.

Example

```
[...-DN-default]->roles[Hadoop::DataNode]]% configurations
[...-DN-default]->roles[Hadoop::DataNode]->configurations]% list
HDFS
-----
doop
doop2
```

Choosing an instance means that configuration settings will apply only to that instance. In the following example, the doop instance is chosen:

```
[...-DN-default]->roles[Hadoop::DataNode]->configurations]% use doop
[...-DN-default]->roles[Hadoop::DataNode]->configurations[doop]]% show
Parameter                                     Value
-----
Bandwidth for balancer                       1048576
Data directories                             /var/lib/hadoop/doop/hadoop/hdfs/
                                              datanode
DataNode Java heap size                      512
HDFS                                          doop
HTTP port                                    50075
HTTPS port                                   50475
Handler count                                10
Heap size                                    0
Heartbeat interval                           3
Maximum number of transfer threads           4096
Network
Number of failed volumes tolerated            0
Protocol port                                50020
Readonly                                     no
Reserved spaced for Non DFS use              1073741824
Revision
Transceiver port                             50010
Type                                           HadoopDataNodeHDFSConfiguration
```

The properties available here for the Hadoop::DataNode role correspond to the properties shown in the Configure HDFS DataNode subtab for figure 3.11.

3.2.3 `cmsh` And The `roleoverview` Command In `device` Mode

The `roleoverview` command can be run from `device` mode. It gives an overview of the roles associated with nodes, categories, and configuration overlays.

Example

```
[bright72->device]% roleoverview
Role                      Nodes                      Categories Configuration Overlays
-----
Hadoop::DataNode          node001..node003          doop-DN-default
Hadoop::HBaseClient       node001,node002          doop-HBRS-default
Hadoop::HBaseServer       bright72                  doop-HBM-default
Hadoop::NameNode          bright72                  doop-NN-default
Hadoop::SparkYARN         bright72                  doop-SY-default
Hadoop::YARNClient        node001..node003          doop-DN-default
Hadoop::YARNServer        bright72                  doop-RM-default
Hadoop::ZooKeeper         node001..node003          doop-ZK-default
boot                      bright72
login                     bright72
master                    bright72
monitoring                bright72
provisioning              bright72
slurmclient               node001..node003 default
slurmserver               bright72
storage                   bright72
```

3.3 Hadoop Maintenance Operations With `cm-hadoop-maint`

The Hadoop maintenance script, `cm-hadoop-maint`, is a Python script. It is called using the full path. If it is run with no arguments, then it displays a help page:

Example

```
[root@bright72 ~]# /cm/local/apps/cluster-tools/hadoop/cm-hadoop-maint
```

Hadoop instance name must be specified. Exiting.

```
USAGE: /cm/local/apps/cluster-tools/hadoop/cm-hadoop-maint -i <name>
[ -b | -f | --start | --stop | --restart | --startonly <set> |
--stoponly <set> | --restartonly <set> | --enterSafeMode |
--leaveSafeMode | --failover [ <from> <to> ] | --failoverstatus |
--yarnfailover [ <from> <to> ] | --yarnfailoverstatus |
--setJavaHome <path> <tool> | --copyconfig <nodes> |
--prepare <nodes> | --optimizeYarn | -h ]
```

OPTIONS:

```
-i <name>          -- instance name
-b                -- cluster balancer utility
-f                -- format & init HDFS
--start           -- start all services
--stop            -- stop all services
--restart         -- restart all services
--startonly <set> -- start all services for <set>
--stoponly <set>  -- stop all services for <set>
--restartonly <set> -- restart all services for <set>
--enterSafeMode   -- enter safemode
```

```

--leaveSafeMode          -- leave safemode
--failover               -- execute a manual failover for HDFS
--failoverstatus         -- return failover status for HDFS
--yarnfailover           -- execute a manual failover for YARN
--yarnfailoverstatus     -- return failover status for YARN
--setJavaHome <path> <tool> -- update JAVA_HOME for selected tool
--optimizeYarn           -- optimize YARN configuration parameters
--copyconfig <nodes>     -- copy Hadoop configuration files to nodes (e.g. login nodes)
--prepare <nodes>        -- prepare nodes to be used for Hadoop deployment (e.g. new nodes)
-h                       -- show usage

```

<set> can be one of the following values: hdfs, mapred, yarn, zookeeper, hbase, spark, sqoop, hive, accumulo, drill, flink, kafka, storm
 <tool> can be one of the following values: hadoop, zookeeper, hbase, spark, sqoop, hive, accumulo, drill, flink, kafka, storm

EXAMPLES:

```

cm-hadoop-maint -i hdfs1 -f
cm-hadoop-maint -i hdfs2 --stop
cm-hadoop-maint -i hdfs2 --stoponly hdfs
cm-hadoop-maint -i hdfs1 --failover nn1 nn2
    executes failover from nn1 to nn2
cm-hadoop-maint -i hdfs1 --failover
    executes failover from active to standby namenode
    if both namenodes are standby, automatically chooses one
cm-hadoop-maint -i hdfs1 --copyconfig node005..node007

```

If Hadoop is used with options, then the name of the Hadoop instance, specified with `-i`, is mandatory.

The other options are now explained in some more detail:

- `-b`: start the balancer daemon
- `-f`: format the Hadoop filesystem and reinitialize it with a standard set of directories, for example: `/user, /tmp`.
- `--start`, `--stop`, `--restart`: allow administrators to start, stop, or restart all services relevant to the Hadoop instance.

To operate on only one of the services, the suffix `only` is appended to the options, and the service is specified as the parameter to the option. The specific service is chosen from `hdfs`, `mapred`, `yarn`, `zk`, `hbase`, `spark`, `sqoop`, or `hive`, so that the format for these options is:

- `--startonly <service>`
- `--stoponly <service>`
- `--restartonly <service>`

- `--enterSafeMode` and `--leaveSafeMode`: enter or leave NameNode safe mode
- `--failover`, `--yarnfailover`: trigger a failover for HDFS, or for YARN
- `--failoverstatus`, `--yarnfailoverstatus`: get the status of High Availability for HDFS or for YARN
- `--copyconfig <nodes>`: Copy Hadoop configuration files to one or more nodes. For example, a Hadoop administrator may wish to add a login node to the Hadoop instance. The login node needs to have relevant Hadoop configuration files, under `/etc/hadoop`. The administrator assigns the login role to the node, and then copies configuration files with the `--copyconfig` option.

- `--prepare <nodes>`: Prepare a node that has a different image for use with the Hadoop instance. For example, a Hadoop administrator may wish to add a new node, such as a `DataNode`, to the Hadoop instance. If the new node has to use a software image that the other Hadoop nodes are already using, then the new node is automatically provisioned with the needed Hadoop configuration files and directories. However, if the new node is to use a different software image, then the new node is not automatically provisioned. It should instead be “prepared” with the `--prepare` option. Running the script with this option provisions the node. After the node has rebooted and is up and running again, the node should be added by the administrator to the Hadoop instance by using Hadoop configuration groups.
- `--setJavaHome <path> <tool>`: allows administrators to change the value of `JAVA_HOME` after the instance has been deployed. The value should be set on a per-tool basis. The script will update the corresponding environment files and restart the relevant services.

4

Running Hadoop Jobs

4.1 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@bright72 ~]# cd /cm/local/apps/cluster-tools/hadoop/
[root@bright72 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@bright72 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 4.1):

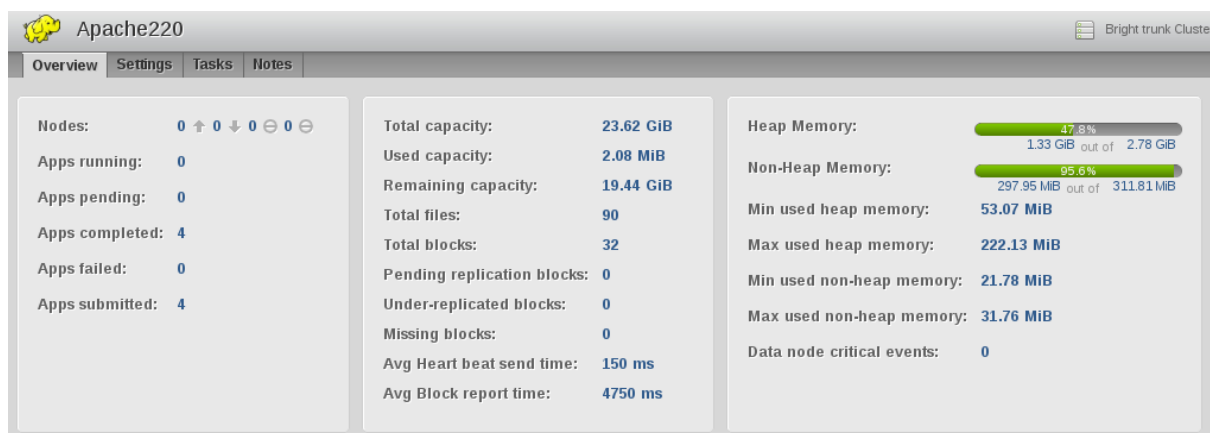


Figure 4.1: Overview Of Activity Seen For A Hadoop Instance In cmgui

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 |

4.2 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

```
[bright72->user[fred]]% set hadoopdfsaccess 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@bright72 ~]$ module add hadoop/Apache220/Apache/2.2.0
[fred@bright72 ~]$ 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.

5

Spark Support In Bright Cluster Manager

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 can be downloaded from <http://spark.apache.org/>. Different pre-built tarballs are available there, for Hadoop 1.x, for CDH 4, and for Hadoop 2.x.

Apache Spark can be installed in YARN mode, that is on top of an existing Hadoop instance (section 5.2) or in Standalone Mode, that is without Hadoop (section 5.3).

Bright Cluster Manager also provides scripts to install Apache Zeppelin (section 5.4) and Tachyon (section 5.5).

5.1 Spark Installation In Bright Cluster Manager—Overview

5.1.1 Prerequisites For Spark Installation, And What Spark Installation Does

The following applies to installing Spark (section 5.1.2) on Bright Cluster Manager:

- Spark can be installed in two different deployment modes: Standalone or YARN.
- When installing in YARN mode, the script installs Spark only on the active head node.
- YARN mode is not supported for Apache Hadoop 1.x, Cloudera CDH 4.x, and Hortonworks HDP 1.3.x.
- Depending on the installation mode, the script creates one or more dedicated Hadoop Configuration Groups for Spark:
 - Standalone mode: Two Hadoop Configuration Groups are created, one for Spark Master and one for Spark Worker roles.
 - YARN mode: Only one Hadoop Configuration Group are created, for Spark YARN role.
- 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/`

- If installing Spark on a Bright Cluster Manager which has Lustre running on it, and which has a Hadoop instance installed on top of it, as described in section 2.4, then both installation modes are available:
 - Standalone mode: Only nodes that can access LustreFS should be selected as worker nodes. It is recommended to set `SPARK_WORKER_DIR` to use a subdirectory of LustreFS that uses the hostname as part of its path, in order to avoid having different workers using the same directory. The additional option `--workerdir` can be used. Care may be needed to escape characters:

Example

```
--workerdir "/mnt/hadoop/tmp/spark-\'hostname\'/"
```

- YARN mode: Configurations are written to the NodeManager. Subsequent operations with Spark should then be carried out on that node.

5.1.2 Spark Installation Utility: `cm-spark-setup`

Bright Cluster Manager provides `cm-spark-setup` to carry out Spark installation. The `cm-spark-setup` utility has the following usage:

```
[root@bright72 ~]# cm-spark-setup -h
```

```
USAGE: /cm/local/apps/cluster-tools/bin/cm-spark-setup [-i <name> -t <file>      [-j <path>] | [-n <node>]
```

OPTIONS:

```
-i <name>      -- instance name
-t <file>      -- Spark tarball
-j <path>      -- Java home path
-c <filename>  -- add Spark instance using config file <filename>
-u <name>      -- uninstall Spark for instance <name>
-n <node>      -- node for Spark YARN role
-h            -- show usage
```

EXAMPLES:

Spark installation in YARN mode

```
cm-spark-setup -i hdfs1 -t /tmp/spark-1.5.2-bin-hadoop26.tgz
```

```
cm-spark-setup -i hdfs1 -t /tmp/spark-1.5.2-bin-hadoop26.tgz -j /usr/lib/jvm/jre-1.7.0-openjdk.
```

Spark installation in Standalone mode

```
cm-spark-setup -c /tmp/spark-conf.xml
```

```
cm-spark-setup -u hdfs1
```

5.2 Spark Installation In YARN Mode

Hadoop 2.x is required for Spark in YARN mode. Cloudera CDH 4.x is not supported for this deployment mode.

Both the options `-i <name>` and `-t <file>` are required, in order to specify the name of the Hadoop instance and the Spark tarball.

The option `-j <path>` is not mandatory. It is used to set the Java Home in Spark environment files. If it is not specified, then the script will use the value retrieved from the Hadoop instance.

`cm-spark-setup` installs Spark by default on the active head node. A different node can be specified with the option `--node`.

Example

```
[root@bright72 ~]# cm-spark-setup -i hdfs1 \
-t /cm/local/apps/hadoop/spark-1.5.2-bin-hadoop2.6.tgz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Spark release '1.5.2-bin-hadoop2.6'
Found Hadoop instance 'hdfs1', release: 2.7.1
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.
Updating images... done.
Waiting for NameNode to be ready... done.
Copying Spark assembly jar to HDFS... done.
Initializing Spark YARN role... done.
Updating configuration in CMDaemon... done.
Waiting for NameNode to be ready... done.
Validating Spark setup...
-- testing 'yarn-client' mode...
-- testing 'yarn-cluster' mode...
Validating Spark setup... done.
Installation successfully completed.
Finished.
```

5.2.1 Using Spark In YARN Mode

Spark supports two deploy modes for launching Spark applications on YARN:

- yarn-client
- yarn-cluster

An example Spark application that comes with the Spark installation is SparkPi. SparkPi can be launched in the two deploy modes as follows:

1. In `yarn-client` mode, the Spark driver runs as a client process. The SparkPi application then runs as a child thread of Application Master.

```
[root@bright72 ~]# module load spark/hdfs1
[root@bright72 ~]# spark-submit --master yarn-client \
--class org.apache.spark.examples.SparkPi \
$SPARK_PREFIX/lib/spark-examples-*.jar
```

2. In `yarn-cluster` mode, the Spark driver runs inside an Application Master process. This is then managed by YARN on the cluster.

```
[root@bright72 ~]# module load spark/hdfs1
[root@bright72 ~]# spark-submit --master yarn-cluster \
--class org.apache.spark.examples.SparkPi \
$SPARK_PREFIX/lib/spark-examples-*.jar
```

Spark has an interactive Scala shell and an interactive Python shell. They can be run as indicated in following two examples:

Example

Scala shell example

```
[root@bright72 ~]# module load spark/hdfs1
[root@bright72 ~]# spark-shell
15/12/23 18:11:14 INFO SecurityManager: Changing view acls to: root
15/12/23 18:11:14 INFO SecurityManager: Changing modify acls to: root
15/12/23 18:11:14 INFO SecurityManager: SecurityManager: authentication disabled;\
  ui acls disabled; users with view permissions: Set(root); users with modify perm\
issions: Set(root)
15/12/23 18:11:14 INFO HttpServer: Starting HTTP Server
15/12/23 18:11:14 INFO Server: jetty-8.y.z-SNAPSHOT
15/12/23 18:11:14 INFO AbstractConnector: Started SocketConnector@0.0.0.0:56526
15/12/23 18:11:14 INFO Utils: Successfully started service 'HTTP class server' on\
port 56526.
Welcome to
```

```
  ____
 /  __/  _/  _/  _/  _/
_  _  _  \  _/  ' _/
/_/_/  .__/_/_/_/_/  /_/_/  version 1.5.1
  _/
```

```
Using Scala version 2.10.4 (OpenJDK 64-Bit Server VM, Java 1.7.0_91)
Type in expressions to have them evaluated.
Type :help for more information.
15/12/23 18:11:20 INFO SparkContext: Running Spark version 1.5.2
```

...

scala>

Example

Python shell example

```
[root@bright72 ~]# module load spark/hdfs1
[root@bright72 ~]# pyspark --master yarn-client
Python 2.6.6 (r266:84292, Jul 23 2015, 15:22:56)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-11)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
15/12/23 18:21:14 INFO SparkContext: Running Spark version 1.5.2

...

15/12/23 18:21:26 INFO EventLoggingListener: Logging events to hdfs:///tmp/spark-ev\
ents/application_1450867470638_0015
15/12/23 18:21:30 INFO YarnClientSchedulerBackend: Registered executor: AkkaRpcEndp\
ointRef(Actor[akka.tcp://sparkExecutor@node001.cm.cluster:48203/user/Executor#14184\
13758]) with ID 1
15/12/23 18:21:30 INFO BlockManagerMasterEndpoint: Registering block manager node00\
1.cm.cluster:56317 with 530.3 MB RAM, BlockManagerId(1, node001.cm.cluster, 56317)
15/12/23 18:21:31 INFO YarnClientSchedulerBackend: Registered executor: AkkaRpcEndp\
ointRef(Actor[akka.tcp://sparkExecutor@node005.cm.cluster:53627/user/Executor#-5008\
22833]) with ID 2
15/12/23 18:21:31 INFO YarnClientSchedulerBackend: SchedulerBackend is ready for sc\
heduling beginning after reached minRegisteredResourcesRatio: 0.8
15/12/23 18:21:31 INFO BlockManagerMasterEndpoint: Registering block manager node00\
5.cm.cluster:48392 with 530.3 MB RAM, BlockManagerId(2, node005.cm.cluster, 48392)
```


Welcome to

```

  ____
 /  __/  _  ____  ____/  /  __
 _  _  _  \  __/  '  __/
/_  /  .__/_/_/_/_/_/_/_  version 1.5.1
  /  /

```

Using Python version 2.6.6 (r266:84292, Jul 23 2015 15:22:56)
 SparkContext available as sc, HiveContext available as sqlContext.
 >>>

5.2.2 Spark Removal With `cm-spark-setup`

`cm-spark-setup` uses the `-u` option to uninstall the Spark instance:

Example

```

[root@bright72 ~]# cm-spark-setup -u hdfs1
Requested removal of Spark for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Cleaning ZooKeeper... done.
Removing additional Spark directories... done.
Removing Spark-related metrics... done.
Removal successfully completed.
Finished.

```

5.3 Spark Installation In Standalone Mode

Spark installation in Standalone mode is also carried out by `cm-spark-setup`, which takes an XML configuration file with the `-c` option. The configuration file has three main sections:

- `<masters>`: for Spark Master configuration
- `<workers>`: for Spark Workers configuration
- `<zookeeper>`: for ZooKeeper configuration, if Zookeeper is used for Spark Master HA

More information on Spark installation configuration can be found at <http://spark.apache.org/docs/latest/>.

An example XML file follows:

Example

```

<sparkConfig>
  <archive>/cm/local/apps/hadoop/spark-1.5.2-bin-hadoop2.6.tgz</archive>
  <javahome>/usr/lib/jvm/jre-1.7.0-openjdk.x86_64</javahome>
  <instance>
    <name>spark1</name>
    <masters recovery="ZOOKEEPER">
      <hosts>node001</hosts>
      <port>7070</port>
      <webport>9080</webport>
      <historywebport>19080</historywebport>
      <localdir>/tmp/spark/spark1</localdir>
    </masters>
  </workers>
</sparkConfig>

```

```

    <hosts>node00[1..5]</hosts>
    <port>7071</port>
    <webport>9081</webport>
    <numcores>0</numcores> <!-- automatic -->
    <numinstances>1</numinstances>
    <workerdir>/tmp/spark/spark1</workerdir>
    <cleanupenabled>true</cleanupenabled>
    <cleanupinterval>1800</cleanupinterval>
    <cleanupappdatattl>604800</cleanupappdatattl>
  </workers>
  <zookeeper>
    <archive>/cm/local/apps/hadoop/zookeeper-3.4.8.tar.gz</archive>
    <hosts>node00[1..3]</hosts>
    <clientport>12181</clientport>
    <serverport>12888</serverport>
    <electionport>13888</electionport>
  </zookeeper>
</instance>
</sparkConfig>

```

Regarding the tags in the configuration file:

- The attribute `recovery` can assume three different values: `NONE`, `FILESYSTEM` or `ZOOKEEPER`. The value is set for the Spark property `spark.deploy.recoveryMode`. If `ZOOKEEPER` is chosen, then the section with tag `<zookeeper>` is mandatory.
- The value for the tag `<localdir>` sets the environment variable `SPARK_LOCAL_DIRS`
- The value for the tag `<numcores>` sets the environment variable `SPARK_WORKER_CORES`. The default value 0 means automatic
- The value for the tag `<numinstances>` sets the environment variable `SPARK_WORKER_INSTANCES`
- The value for tag `<workerdir>` sets the environment variable `SPARK_WORKER_DIR`
- The `cleanup*` tags have values that work as follows:
 - `<cleanupenabled>` enables the periodic cleanup of worker/application directories. This corresponds to the property `spark.worker.cleanup.enabled`.
 - `<cleanupinterval>` sets the interval, in seconds, for cleaning up the worker directory. This corresponds to the property `spark.worker.cleanup.interval`
 - `<cleanupappdatattl>` sets the number of seconds to retain application worker directories. This corresponds to `spark.worker.cleanup.appDataTtl`.

An installation with an XML configuration file should result in a session similar to the following:

Example

```

[root@bright72 ~]# cm-spark-setup -c /root/sparkconf.xml
Reading config from file '/root/sparkconf.xml'... done.
Spark release '1.5.2-bin-hadoop2.6'
Creating Spark instance 'spark1'... done.
Spark will be installed in Standalone mode with ZooKeeper.
Spark Master service will be run on: node001
Spark Worker service will be run on: node001,node002,node003,node004,node005
ZooKeeper will be run on: node001,node002,node003
Spark being installed... done.

```

```
ZooKeeper being installed... done.
Creating directories for Spark... done.
Creating module file for Spark... done.
Creating module file for ZooKeeper... done.
Creating configuration files for Spark... done.
Creating configuration files for ZooKeeper... done.
Updating images... done.
Initializing ZooKeeper service... done.
Initializing Spark Master service... done.
Initializing Spark Worker services... done.
Updating configuration in CMDaemon... done.
Validating ZooKeeper setup... done.
Validating Spark setup...
-- testing Python application...
-- testing R application...
-- testing Java application...
-- testing Scala application...
Validating Spark setup... done.
Installation successfully completed.
Finished.
```

5.3.1 Using Spark In Standalone Mode

Spark can run applications written in different languages, such as Java, Python, R, and Scala:

Example

Calculating Pi in Java

```
[root@bright72 ~]# module load spark/spark1
[root@bright72 ~]# run-example JavaSparkPi
15/12/28 16:36:45 INFO SparkContext: Running Spark version 1.5.2

...

15/12/28 16:36:55 INFO DAGScheduler: Job 0 finished: reduce at JavaSparkPi.java:4\
8, took 5.635850 s
Pi is roughly 3.14114
15/12/28 16:36:55 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler/met\
rics/json,null

...

15/12/28 16:36:56 INFO RemoteActorRefProvider$RemotingTerminator: Shutting down r\
emote daemon.
15/12/28 16:36:56 INFO ShutdownHookManager: Deleting directory /tmp/spark-ce4f294\
c-a008-4eee-8c3c-a295db4166f8
15/12/28 16:36:56 INFO RemoteActorRefProvider$RemotingTerminator: Remote daemon s\
hut down; proceeding with flushing remote transports.
```

Example

Calculating Pi in Python

```
[root@bright72 ~]# module load spark/spark1
[root@bright72 ~]# spark-submit $SPARK_PREFIX/examples/src/main/python/pi.py
15/12/28 17:03:19 INFO SparkContext: Running Spark version 1.5.2
```

```
...

15/12/28 17:03:27 INFO DAGScheduler: Job 0 finished: reduce at /cm/shared/apps/hadoop/Apache/spark-1.5.2-bin-hadoop2.6/examples/src/main/python/pi.py:39, took 3.9\
84307 s
Pi is roughly 3.145680

...

15/12/28 17:03:27 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler/met\
rics/json,null
15/12/28 17:03:27 INFO RemoteActorRefProvider$RemotingTerminator: Remoting shut d\
own.
15/12/28 17:03:28 INFO ShutdownHookManager: Shutdown hook called
15/12/28 17:03:28 INFO ShutdownHookManager: Deleting directory /tmp/spark-e09c71a\
0-469a-442c-a243-a01d6524928b
```

Example

Doing some operations on DataFrames in R

```
[root@bright72 ~]# module load spark/spark1
[root@bright72 ~]# spark-submit $SPARK_PREFIX/examples/src/main/r/dataframe.R
Loading required package: methods
```

```
Attaching package: 'SparkR'
```

```
The following objects are masked from 'package:stats':
```

```
filter, na.omit
```

```
The following objects are masked from 'package:base':
```

```
intersect, rbind, sample, subset, summary, table, transform
```

```
15/12/28 17:08:46 INFO SparkContext: Running Spark version 1.5.2
```

```
...
```

```
15/12/28 17:08:54 INFO BlockManagerMasterEndpoint: Registering block manager 10.1\
41.0.1:57452 with 530.3 MB RAM, BlockManagerId(0, 10.141.0.1, 57452)
root
```

```
|-- name: string (nullable = true)
```

```
|-- age: double (nullable = true)
```

```
15/12/28 17:08:54 INFO JSONRelation: Listing file:/cm/shared/apps/hadoop/Apache/s\
park-1.5.1-bin-hadoop2.6/examples/src/main/resources/people.json on driver
```

```
...
```

```
15/12/28 17:08:57 INFO DAGScheduler: Job 1 finished: jsonFile at NativeMethodAcce\
ssorImpl.java:-2, took 2.258387 s
```

```
root
```

```
|-- age: long (nullable = true)
```

```
|-- name: string (nullable = true)
```

```
15/12/28 17:08:58 INFO BlockManagerInfo: Removed broadcast_2_piece0 on 10.141.255\
.254:56423 in memory (size: 2.2 KB, free: 530.3 MB)
```

```
...

15/12/28 17:09:01 INFO DAGScheduler: ResultStage 2 (dfToCols at NativeMethodAcces\
sorImpl.java:-2) finished in 2.743 s
15/12/28 17:09:01 INFO DAGScheduler: Job 2 finished: dfToCols at NativeMethodAcce\
ssorImpl.java:-2, took 2.758451 s
    name
1 Justin
15/12/28 17:09:01 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler/sta\
tic/sql,null

...

15/12/28 17:09:01 INFO RemoteActorRefProvider$RemotingTerminator: Remoting shut d\
own.
15/12/28 17:09:02 INFO ShutdownHookManager: Shutdown hook called
15/12/28 17:09:02 INFO ShutdownHookManager: Deleting directory /tmp/spark-b517cc6\
8-5c91-4b1f-9e6d-287ab68fd2de
```

Example

Calculating Pi in Scala

```
[root@bright72 ~]# module load spark/spark1
[root@bright72 ~]# run-example SparkPi
15/12/28 17:07:32 INFO SparkContext: Running Spark version 1.5.2

...

15/12/28 17:07:42 INFO DAGScheduler: Job 0 finished: reduce at SparkPi.scala:36, \
took 5.624478 s
Pi is roughly 3.14848
15/12/28 17:07:42 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler/metrics/json,null

...

15/12/28 17:07:42 INFO ShutdownHookManager: Shutdown hook called
15/12/28 17:07:42 INFO RemoteActorRefProvider$RemotingTerminator: Shutting down remote daemon.
15/12/28 17:07:42 INFO ShutdownHookManager: Deleting directory /tmp/spark-ed276e5\
2-6de7-4459-aeda-2169b0300217
```

Example

Doing a wordcount on a file stored in HDFS

```
[foobar@bright72 ~]$ module load hadoop/hdfs1
[foobar@bright72 ~]$ hdfs dfs -copyFromLocal /tmp/hamlet.txt /user/foobar/hamlet.txt
[foobar@bright72 ~]$ module load spark/hdfs1_spark
[foobar@bright72 ~]$ spark-submit $SPARK_PREFIX/examples/src/main/python/wordcoun\
t.py hdfs://node001:8020/user/foobar/hamlet.txt
15/12/24 19:23:58 INFO SparkContext: Running Spark version 1.5.2

...

15/12/24 19:24:07 INFO TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in \
112 ms on 10.141.0.4 (1/1)
```

```

15/12/24 19:24:07 INFO TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have a\
ll completed, from pool
15/12/24 19:24:07 INFO DAGScheduler: Job 0 finished: collect at /cm/shared/apps/h\
adoop/Spark/spark-1.5.2-bin-hadoop2.6/examples/src/main/python/wordcount.py:35, \
took 4.837430 s
: 1285
pardon: 1
cheefe: 1
better.: 1
ever!: 1
Student): 1

...

smot: 1
goodly: 1
Sits: 1
indirectly: 1
15/12/24 19:24:08 INFO ContextHandler: stopped o.s.j.s.ServletContextHandler/met\
rics/json,null

...

15/12/24 19:24:08 INFO SparkContext: Successfully stopped SparkContext
15/12/24 19:24:08 INFO RemoteActorRefProvider$RemotingTerminator: Shutting down r\
emote daemon.
15/12/24 19:24:08 INFO RemoteActorRefProvider$RemotingTerminator: Remote daemon s\
hut down; proceeding with flushing remote transports.
15/12/24 19:24:08 INFO RemoteActorRefProvider$RemotingTerminator: Remoting shut d\
own.
15/12/24 19:24:09 INFO ShutdownHookManager: Shutdown hook called
15/12/24 19:24:09 INFO ShutdownHookManager: Deleting directory /tmp/spark/hdfs1_s\
park/spark-0ca6228e-f6fb-4f41-bd95-810da70c625f/pyspark-45726fbf-1b89-4278-a508-e\
e57882ecb54
15/12/24 19:24:09 INFO ShutdownHookManager: Deleting directory /tmp/spark/hdfs1_s\
park/spark-0ca6228e-f6fb-4f41-bd95-810da70c625f

```

5.4 Zeppelin Installation

Apache Zeppelin is a web-based notebook that enables interactive data analytics. Zeppelin can be installed when Spark has already been deployed in YARN mode (section 5.2) or in Standalone mode (section 5.3).

The Apache Zeppelin tarball should be downloaded from <http://zeppelin.apache.org/>.

5.4.1 Zeppelin Installation With `cmhadoop-zeppelin-setup`

Bright Cluster Manager provides `cmhadoop-zeppelin-setup` to install Zeppelin.

Prerequisites For Zeppelin Installation, And What Zeppelin Installation Does

The following applies to using `cmhadoop-zeppelin-setup`:

- Spark should already be installed
- The `cmhadoop-zeppelin-setup` script installs a Zeppelin service by default on the active headnode. A different host can be specified by using the option `--host`
- The script creates no roles for Zeppelin.

- Zeppelin is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Zeppelin configuration files are copied by the script to the directory `/etc/hadoop/`
- The Zeppelin web application runs by default on port 18090 of its host.

An Example Run With `cmhadoop-zeppelin-setup`

The option

`-j <path>`

is mandatory. It is used to set the Java Home in Zeppelin environment files. The path should point to a Java Development Kit.

Example

```
[root@bright72 ~]# cmhadoop-zeppelin-setup -i spark1 -j /usr/lib/jvm/java-1.7.0-
openjdk.x86_64/ -t /tmp/zeppelin-0.5.5-incubating-bin-all.tgz
Zeppelin release '0.5.5-incubating-bin-all'
Zeppelin service will be run on the head node.
Found Hadoop instance 'spark1', release: 1.6.0-bin-hadoop2.6
Spark Master found.
Zeppelin being installed... done.
Creating directories for Zeppelin... done.
Creating module file for Zeppelin... done.
Creating configuration files for Zeppelin... done.
Updating images... done.
Initializing Zeppelin service... done.
Installation successfully completed.
Finished.
```

5.4.2 Zeppelin Removal With `cmhadoop-zeppelin-setup`

`cmhadoop-zeppelin-setup` uses the `-u` option to uninstall the Zeppelin instance.

Example

```
[root@bright72 ~]# cmhadoop-zeppelin-setup -u spark1
Requested removal of Zeppelin for Hadoop instance 'spark1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Zeppelin directories... done.
Removal successfully completed.
Finished.
```

5.5 Tachyon Installation

Tachyon is a memory-centric distributed storage system. It enables reliable data sharing across cluster jobs, since it lies between computation frameworks and various kinds of storage systems.

The Tachyon tarball for Spark should be downloaded from <http://tachyon-project.org/>. The Tachyon version to download is dependent on the Spark version; a compatibility table is available at <http://tachyon-project.org/documentation/master/Running-Spark-on-Tachyon.html>.

5.5.1 Tachyon Installation With `cmhadoop-tachyon-setup`

Bright Cluster Manager provides `cmhadoop-tachyon-setup` to carry out Tachyon installation:

Prerequisites For Tachyon Installation, And What Tachyon Installation Does

The following applies to using `cmhadoop-tachyon-setup`:

- A Spark instance must already be installed.
- `cmhadoop-tachyon-setup` installs Tachyon only on the active head node and on the Spark Worker nodes of the chosen Spark instance.
- The Tachyon installation directory is under `/cm/shared/hadoop/`.
- Tachyon is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- The script sets the storage layer address to a subdirectory of the Tachyon installation directory.
- The script creates no roles for Tachyon
- Tachyon configuration files are copied by the script to under `/etc/hadoop/`

An Example Run With `cmhadoop-tachyon-setup`

The option

`-j <path>`

is not mandatory. It is used to set the Java Home in Tachyon environment files. If the option is not specified, then the script will use the value retrieved from the Spark instance.

Example

```
[root@bright72 ~]# cmhadoop-tachyon-setup -i spark1 -t /tmp/tachyon-0.8.2-bin.tar.gz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Tachyon release '0.8.2-bin'
Found Spark instance 'spark1', release: 1.6.0-bin-hadoop2.6
Tachyon Master will be run on the head node.
Tachyon being installed... done.
Creating directories for Tachyon... done.
Creating module file for Tachyon... done.
Creating configuration files for Tachyon... done.
Updating images... done.
Formatting Tachyon FS... done.
Initializing services for Tachyon... done.
Validating Tachyon setup... done.
Installation successfully completed.
Finished.
```

5.5.2 Tachyon Removal With `cmhadoop-tachyon-setup`

`cmhadoop-tachyon-setup` uses the `-u` option to uninstall the Tachyon instance.

Example

```
[root@bright72 ~]# cmhadoop-tachyon-setup -u spark1
Requested removal of Tachyon for Hadoop instance 'spark1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Tachyon directories... done.
Removal successfully completed.
Finished.
```


5.5.3 Using Tachyon

Tachyon consists of several components: one Master, multiple Workers, and an executable client, `tachyon`, that can be run after the user loads the corresponding module. The following example shows how to execute a Spark word count example on data copied to the Tachyon storage layer.

Example

```
[root@bright72 ~]# module load tachyon/spark1
[root@bright72 ~]# tachyon tfs copyFromLocal /cm/local/examples/hamlet.txt\
/wordcount/input.txt
Copied /cm/local/examples/hamlet.txt to /wordcount/input.txt
[root@bright72 ~]# module load spark/spark1
[root@bright72 ~]# spark-submit /cm/local/apps/cluster-tools/hadoop/conf/tachyon/\
test.py tachyon://ml-hadoopdev.cm.cluster:19998/wordcount/input.txt tachyon://ml-\
hadoopdev.cm.cluster:19998/wordcount/output

...

16/01/14 10:59:42 INFO : Tachyon client (version 0.8.2) is trying to connect with\
FileSystemMaster master @ ml-hadoopdev.cm.cluster/10.141.255.254:19998
16/01/14 10:59:42 INFO : Client registered with FileSystemMaster master @ ml-hado\
opdev.cm.cluster/10.141.255.254:19998
16/01/14 10:59:42 INFO : tachyon://ml-hadoopdev.cm.cluster:19998 tachyon://ml-had\
oopdev.cm.cluster:19998 /cm/shared/apps/hadoop/Apache/tachyon-0.8.2-bin//underFSS\
torage
16/01/14 10:59:42 INFO : getFileStatus(tachyon://ml-hadoopdev.cm.cluster:19998/wo\
rdcount/input.txt): HDFS Path: /cm/shared/apps/hadoop/Apache/tachyon-0.8.2-bin/un\
derFSSStorage/wordcount/input.txt TPath: tachyon://ml-hadoopdev.cm.cluster:19998/w\
ordcount/input.txt

...

16/01/14 10:59:46 INFO : create(tachyon://ml-hadoopdev.cm.cluster:19998/wordcount\
/output/_SUCCESS, rw-r--r--, true, 65536, 1, 536870912, null)
16/01/14 10:59:46 INFO : Tachyon client (version 0.8.2) is trying to connect with\
FileSystemMaster master @ ml-hadoopdev.cm.cluster/10.141.255.254:19998
16/01/14 10:59:46 INFO : Client registered with FileSystemMaster master @ ml-hado\
opdev.cm.cluster/10.141.255.254:19998

...

16/01/14 10:59:47 INFO ShutdownHookManager: Shutdown hook called
16/01/14 10:59:47 INFO ShutdownHookManager: Deleting directory /tmp/spark-5841602\
9-4f26-4982-9f85-59fa5bfaf7c4
16/01/14 10:59:47 INFO ShutdownHookManager: Deleting directory /tmp/spark-5841602\
9-4f26-4982-9f85-59fa5bfaf7c4/httpd-02ca9c4f-8ac5-4bf2-bd53-f35111f812cd
16/01/14 10:59:47 INFO ShutdownHookManager: Deleting directory /tmp/spark-5841602\
9-4f26-4982-9f85-59fa5bfaf7c4/pyspark-1230b258-85a0-4ed5-a43e-012348050581
[root@bright72 ~]# tachyon tfs cat /wordcount/output/*
(u'', 167)
(u'fardels', 1)
(u'flesh', 1)

...

(u'dread', 1)
```

```
(u'the', 14)
(u'resolution', 1)
```

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)
- Drill (section 6.2)
- Flink (section 6.3)
- Giraph (section 6.4)
- Hive (section 6.5)
- Ignite (section 6.6)
- Kafka (section 6.7)
- Pig (section 6.8)
- Sqoop (section 6.9)
- Sqoop2 (section 6.10)
- Storm (section 6.11)
- Tachyon (section 6.12)

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 `cmhadoop-accumulo-setup`

Bright Cluster Manager provides `cmhadoop-accumulo-setup` to carry out the installation of Accumulo as part of the `cm-apache-hadoop-extras` package.

Prerequisites For Accumulo Installation, And What Accumulo Installation Does

The following applies to using `cmhadoop-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 `cmhadoop-accumulo-setup` script only installs Accumulo on the active head node and on the DataNodes of the chosen Hadoop instance.
- The script creates two dedicated Hadoop Configuration Groups for Accumulo: one for Accumulo Master and one for Accumulo Tablet Servers.
- 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 `cmhadoop-accumulo-setup`.
- The secret string for the instance is a random string created by `cmhadoop-accumulo-setup`.
- A password for the `root` user must be specified.
- The Tracer service uses 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.
- `cmhadoop-accumulo-setup` tries to build the native map library. If no Java Development Kit is available, then the script displays a warning message.
- Validation tests are carried out by the script.
- When installing Accumulo on a Hadoop instance configured to run on Lustre within Bright Cluster Manager (section 2.4), the services for Garbage Collector, Master, Tracer, and Monitor will be run on the node which is the ResourceManager.

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

An Example Run With `cmhadoop-accumulo-setup`

The option

`-j <path>`

is not mandatory. It is used to set the Java home path in Accumulo environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance.

The option

`-p <rootpass>`

is mandatory. The specified password is also used by the Tracer service to connect to Accumulo. The password is 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@bright72 ~]# cmhadoop-accumulo-setup -i hdfs1 -p 12345 -s 900MB \
-t /tmp/accumulo-1.7.0-bin.tar.gz --master node005
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Accumulo release '1.7.0'
Accumulo GC, Master, Monitor, and Tracer services will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.7.1
Accumulo being installed... done (with native library).
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... done.
Initializing master services for Accumulo... done.
Updating configuration in CMDaemon... done.
Waiting for NameNode to be ready... done.
Executing validation test... done.
Installation successfully completed.
Finished.
```

6.1.2 Accumulo Removal With `cmhadoop-accumulo-setup`

`cmhadoop-accumulo-setup` uses the `-u` option to uninstall the Accumulo instance. Data and meta-data are not removed.

Example

```
[root@bright72 ~]# cmhadoop-accumulo-setup -u hdfs1
Requested removal of Accumulo for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
```

```
Cleaning ZooKeeper... done.
Removing additional Accumulo directories... done.
Removal successfully completed.
Finished.
```

6.1.3 Accumulo MapReduce Example

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

Example

```
[root@bright72 ~]# 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 Drill

Apache Drill is an SQL query engine for Big Data exploration. Drill supports a variety of NoSQL databases and file systems. A single query can join data from multiple datastores. In addition, Drill supports data locality, so putting Drill and the datastore on the same nodes is a good idea.

The Apache Drill tarball can be downloaded from <http://drill.apache.org/>. Hadoop version and distribution compatibilities are listed in section 1.4.

6.2.1 Drill Installation With `cmhadoop-drill-setup`

Bright Cluster Manager provides `cmhadoop-drill-setup` to carry out Drill installation:

Prerequisites For Drill Installation, And What Drill Installation Does

The following applies to using `cmhadoop-drill-setup`:

- A Hadoop instance must already be installed.
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- The `cmhadoop-drill-setup` script installs Drill services by default on the DataNodes of the chosen Hadoop instance.
- The script creates a dedicated Hadoop Configuration Group for Drill.
- Drill executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Drill configuration files are copied by the script to under `/etc/hadoop/`
- The Drillbit services are started up by the script
- Validation tests are carried out by the script using `sqllline`.

The options for `cmhadoop-drill-setup` are listed on running `cmhadoop-drill-setup -h`.

An Example Run With `cmhadoop-drill-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path in Drill environment files. If the option is not specified, then the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright72 ~]# cmhadoop-drill-setup -i hdfs1 -t /tmp/apache-drill-1.4.0.tar.gz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Drill release '1.4.0'
Found Hadoop instance 'hdfs1', release: 2.7.1
Drill being installed... done.
Creating directories for Drill... done.
Creating module file for Drill... done.
Creating configuration files for Drill... done.
Updating images... done.
Initializing services for Drill... done.
Updating configuration in CMDaemon... done.
Validating Drill setup... done.
Installation successfully completed.
Finished.
```

6.2.2 Drill Removal With `cmhadoop-drill-setup`

`cmhadoop-drill-setup` uses the `-u` option to uninstall Drill.

Example

```
[root@bright72 ~]# cmhadoop-drill-setup -u hdfs1
Requested removal of Drill for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Cleaning ZooKeeper... done.
Removing additional Drill directories... done.
Removal successfully completed.
Finished.
```

6.3 Flink

Apache Flink is an open source platform for distributed stream and batch data processing. Flink's pipelined runtime system enables the execution of bulk batch and stream processing programs. Flink does not provide its own data storage system—input data must be stored in a distributed storage system such as HDFS or HBase.

The Apache Flink tarball can be downloaded from <http://flink.apache.org/>. Hadoop versions and distributions compatibilities are listed in section 1.4.

6.3.1 Flink Installation With `cmhadoop-flink-setup`

Bright Cluster Manager provides `cmhadoop-flink-setup` to carry out Flink installation:

Prerequisites For Flink Installation, And What Flink Installation Does

The following applies to using `cmhadoop-flink-setup`:

- A Hadoop instance must already be installed.
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.

- The `cmhadoop-flink-setup` script by default only installs Flink Job Manager on the active head node and Flink Task Managers on the DataNodes of the chosen Hadoop instance. A node other than master can be specified by using the option `--master`, or its alias for this setup script, `--jobmanager`.
- The script creates two dedicated Hadoop Configuration Groups for Flink: one for the Job Manager and one for Task Managers.
- SLES11sp4 ships with the IBM JVM, which has incompatibility issues with Flink. The Oracle JVM should be used, for example with the option `-j /usr/java/jre1.7.0_80/`. Oracle JVM can be downloaded from <http://www.java.com/en/download/>
- Flink executables are copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Flink configuration files are copied by the script to under `/etc/hadoop/`
- The Flink Job Manager and Task Manager services are started up by the script
- Validation tests are carried out by the script using `flink`.

The options for `cmhadoop-flink-setup` are listed on running `cmhadoop-flink-setup -h`.

An Example Run With `cmhadoop-flink-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path in Flink environment files. If the option is not specified, then the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright72 ~]# cmhadoop-flink-setup -i hdfs1 -t /tmp/flink-0.10.1-bin-hadoop27.tgz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Flink release '0.10.1-bin-hadoop27'
Found Hadoop instance 'hdfs1', release: 2.7.1
ZooKeeper found. Configuring Flink JobManager HA.
Flink Job Manager will be run on the head node.
Flink being installed... done.
Creating directories for Flink... done.
Creating module file for Flink... done.
Creating configuration files for Flink... done.
Updating images... done.
Initializing Task Managers for Flink... done.
Initializing Job Manager for Flink... done.
Updating configuration in CMDaemon... done.
Waiting for NameNode to be ready... done.
Validating Flink setup... done.
Installation successfully completed.
Finished.
```

6.3.2 Flink Removal With `cmhadoop-flink-setup`

`cmhadoop-flink-setup` uses the `-u` option to uninstall Flink.

Example

```
[root@bright72 ~]# cmhadoop-flink-setup -u hdfs1
Requested removal of Flink for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
```



```
Removing additional Flink directories... done.
Removal successfully completed.
Finished.
```

6.4 Giraph

Apache Giraph is an iterative graph processing system built for high scalability. Giraph is inspired by the Bulk Synchronous Parallel model of distributed computation introduced by Leslie Valiant. Giraph is built on top of Apache Hadoop and it uses the MapReduce framework to run Giraph jobs. The input to a Giraph computation is a graph composed of vertices and directed edges. As an example, Giraph can compute the length of the shortest paths from a source node to all other nodes.

The Apache Giraph tarball should be built from sources, since it depends on the Hadoop distribution and version.

```
[foobar@bright72 ~]$ git clone http://git-wip-us.apache.org/repos/asf/giraph.git
[foobar@bright72 ~]$ cd giraph/
[foobar@bright72 ~]$ mvn -Dhadoop.version=2.7.1 -Phadoop_2 -fae -DskipTests clean package
```

For installation, the tarball can be found in `./giraph-dist/target/`, and the JAR file can be found in `./giraph-examples/target/`. For example:

```
[foobar@bright72 ~]$ ls giraph-dist/target/giraph-1.2.0-SNAPSHOT-for-hadoop-2.7.1\
-bin.tar.gz
[foobar@bright72 ~]$ ls giraph-examples/target/giraph-examples-1.2.0-SNAPSHOT-for\
-hadoop-2.7.1-jar-with-dependencies.jar
```

6.4.1 Giraph Installation With `cmhadoop-giraph-setup`

Bright Cluster Manager provides `cmhadoop-giraph-setup` to carry out Giraph installation:

Prerequisites For Giraph Installation, And What Giraph Installation Does

The following applies to using `cmhadoop-giraph-setup`:

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

The options for `cmhadoop-giraph-setup` are listed on running `cmhadoop-giraph-setup -h`.

An Example Run With `cmhadoop-giraph-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path in Giraph environment files. If the option is not specified, then the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright72 ~]# cmhadoop-giraph-setup -i hdfs1 -t /root/giraph/giraph-1.2.0\
-SNAPSHOT-for-hadoop-2.7.1-bin.tar.gz --examplejar /root/giraph/giraph-examples-1.2.0\
-SNAPSHOT-for-hadoop-2.7.1-jar-with-dependencies.jar
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Giraph release '1.2.0-SNAPSHOT-for-hadoop-2.7.1-bin'
Found Hadoop instance 'hdfs1', release: 2.7.1
ZooKeeper found for instance hdfs1.
```

```
Giraph being installed... done.
Creating directories for Giraph... done.
Creating module file for Giraph... done.
Creating configuration files for Giraph... done.
Waiting for NameNode to be ready... done.
Validating Giraph setup... done.
Installation successfully completed.
Finished.
```

6.4.2 Giraph Removal With `cmhadoop-giraph-setup`

`cmhadoop-giraph-setup` uses the `-u` option to uninstall the Giraph instance. Data and metadata will not be removed.

Example

```
[root@bright72 ~]# cmhadoop-giraph-setup -u hdfs1
Requested removal of Giraph for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Cleaning ZooKeeper... done.
Removing additional Giraph directories... done.
Removal successfully completed.
Finished.
```

6.5 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.

6.5.1 Hive Installation With `cmhadoop-hive-setup`

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

Prerequisites For Hive Installation, And What Hive Installation Does

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

- A Hadoop instance must already be installed.
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- 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/>

- `cmhadoop-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
```

- For SLES11sp4, the package `java-1_7_1-ibm-jdbc` should be installed on the head node
- 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 `cmhadoop-hive-setup`.
 - `<hivepass>` is the password for hive user. The same password is used later by `cmhadoop-hive-setup`.
 - The DROP line is needed only if a database with that name already exists.
- The `cmhadoop-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 creates a dedicated Hadoop Configuration Group for Hive.
 - 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`.
 - When installing Hive on a Hadoop instance configured to run on Lustre within Bright Cluster Manager (section 2.4), Hive should be deployed on a node that has access to LustreFS (by using the `--master` option if needed). Subsequent operations with Hive should be carried out on that node.

The options for `cmhadoop-hive-setup` are listed on running `cmhadoop-hive-setup -h`.

An Example Run With `cmhadoop-hive-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path in Hive environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright72 ~]# cmhadoop-hive-setup -i hdfs1 -p <hivepass> --metastoredb <metastoredb> \
-t /tmp/apache-hive-1.2.1-bin.tar.gz --master node005
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Hive release '1.2.1-bin'
Using MySQL server on active headnode.
Hive service will be run on node: node005
Hive being installed... done.
Using MySQL Connector/J installed in /usr/share/java/
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.
Updating configuration in CMDaemon... 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.
```

6.5.2 Hive Removal With `cmhadoop-hive-setup`

`cmhadoop-hive-setup` uses the `-u` option to uninstall the Hive instance. Data and metadata will not be removed.

Example

```
[root@bright72 ~]# cmhadoop-hive-setup -u hdfs1
Requested removal of Hive for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Cleaning ZooKeeper... done.
Removing additional Hive directories... done.
Removal successfully completed.
Finished.
```

6.5.3 Beeline

The latest Hive releases include HiveServer2, which supports the 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@bright72 ~]# 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

```

6.6 Ignite

Apache Ignite is a high-performance, integrated and distributed in-memory platform for computing. Specifically Apache Ignite In-Memory MapReduce eliminates the overhead associated with NameNode, since data locality is assured via a hashing function. It also uses push-based resource allocation for better performance.

6.6.1 Ignite Installation With `cmhadoop-ignite-setup`

Bright Cluster Manager provides `cmhadoop-ignite-setup` to carry out Ignite installation.

Prerequisites For Ignite Installation, And What Ignite Installation Does

The following applies to using `cmhadoop-ignite-setup`:

- A Hadoop instance must already be installed. Ignite installation is not supported for Hadoop 1.x and Cloudera CDH 4.x
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- `cmhadoop-ignite-setup` installs Ignite only on the ResourceManager nodes
- The script creates no roles for Ignite
- Ignite is copied by the script to a subdirectory under `/cm/shared/hadoop/`
- Ignite configuration files are copied by the script to under `/etc/hadoop/`
- `cmhadoop-ignite-setup` creates a configuration overlay (section 3.1.9), for example `hdfs1-ignite`, comprising the nodes for the Hadoop instance. The Overlay contains a customization (section A.6), that is used to set the property `mapreduce.jobtracker.address` equals to one of the Ignite servers in `mapred-site.xml`. For example: `mapreduce.jobtracker.address` is set with value `node001.cm.cluster:11211`

The options for `cmhadoop-ignite-setup` are listed on running `cmhadoop-ignite-setup -h`.

An Example Run With `cmhadoop-ignite-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path in Ignite environment files. If the option is not specified, then the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright72 ~]# cmhadoop-ignite-setup -i hdfs1 -t /tmp/apache-ignite-hadoop-1.4.0-bin.zip
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Ignite release '1.4.0-bin'
Found Hadoop instance 'hdfs1', release: 2.7.1
Ignite being installed... done.
Creating directories for Ignite... done.
Creating module file for Ignite... done.
Creating configuration files for Ignite... done.
Updating images... done.
Initializing services for Ignite... done.
Updating configuration in CMDaemon... done.
Waiting for NameNode to be ready... done.
Validating Ignite setup... done.
Installation successfully completed.
Finished.
```

6.6.2 Ignite Removal With `cmhadoop-ignite-setup`

`cmhadoop-ignite-setup` uses the `-u` option to uninstall the Ignite instance.

Example

```
[root@bright72 ~]# cmhadoop-ignite-setup -u hdfs1
Requested removal of Ignite for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Ignite directories... done.
Removal successfully completed.
Finished.
```

6.6.3 Using Ignite

Ignite can be used to run Hadoop jobs, using the Ignite job tracker, by using “In-Memory MapReduce”. Further details on this can be found at <http://ignite.apache.org/use-cases/hadoop/mapreduce>. The examples that follow show how to calculate Pi using the Ignite framework or YARN.

Example

Using Ignite

```
[root@bright72 ~]# module load hadoop/hdfs1
[root@bright72 ~]# hadoop jar $HADOOP_PREFIX/share/hadoop/mapreduce/hadoop-mapred\
uce-examples-2.7.1.jar pi 10 1000
Number of Maps = 10
Samples per Map = 1000

...

Starting Job
Jan 04, 2016 11:30:22 AM org.apache.ignite.internal.client.impl.connection.GridCl\
ientNioTcpConnection <init>

...

16/01/04 11:30:29 INFO mapreduce.Job: map 100% reduce 100%
16/01/04 11:30:29 INFO mapreduce.Job: Job job_1a759599-56fc-43da-baf0-c68548a7c5d\
```

```
b_0002 completed successfully
16/01/04 11:30:29 INFO mapreduce.Job: Counters: 0
Job Finished in 8.015 seconds
Estimated value of Pi is 3.14080000000000000000
```

The MapReduce framework can be switched via cmsh:

```
[ml-hadooptest->hadoop]% use hdfs1
[ml-hadooptest->hadoop[hdfs1]]% get frameworkformapreduce
Ignite
[ml-hadooptest->hadoop[hdfs1]]% set frameworkformapreduce yarn
[ml-hadooptest->hadoop*[hdfs1*]]% commit
```

The same example can be now run using YARN framework. For the same run it results in a more than 7 times slower job execution.

Example

Using YARN

```
[root@bright72 ~]# module load hadoop/hdfs1
[root@bright72 ~]# hadoop jar $HADOOP_PREFIX/share/hadoop/mapreduce/hadoop-mapred\
uce-examples-2.7.1.jar pi 10 1000
Number of Maps = 10
Samples per Map = 1000

...

Starting Job
16/01/04 11:33:37 INFO impl.TimelineClientImpl: Timeline service address: http://\
node003.cm.cluster:8188/ws/v1/timeline/

...

16/01/04 11:34:22 INFO mapreduce.Job: map 100% reduce 100%
16/01/04 11:34:36 INFO mapreduce.Job: Job job_1451903604981_0001 completed succes\
sfully

...

Job Finished in 60.144 seconds
Estimated value of Pi is 3.14080000000000000000
```

6.7 Kafka

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

6.7.1 Kafka Installation With cmhadoop-kafka-setup

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

Prerequisites For Kafka Installation, And What Kafka Installation Does

The following applies to using cmhadoop-kafka-setup:

- A Hadoop instance, with ZooKeeper, must already be installed.

- `cmhadoop-kafka-setup` installs Kafka only on the ZooKeeper nodes.
- The script creates a dedicated Hadoop Configuration Group for Kafka.
- SLES11sp4 ships with the IBM JVM, which has incompatibility issues with Kafka. The Oracle JVM should be used, for example with the option `-j /usr/java/jre1.7.0_80/`. Oracle JVM can be downloaded from <http://www.java.com/en/download/>
- 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/`.

The options for `cmhadoop-kafka-setup` are listed on running `cmhadoop-kafka-setup -h`.

An Example Run With `cmhadoop-kafka-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path in Kafka environment files. If the option is not specified, the script will use the value retrieved from the Hadoop instance.

Example

```
[root@bright72 ~]# cmhadoop-kafka-setup -i hdfs1 -t /tmp/kafka_2.11-0.8.2.2.tgz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Kafka release '0.8.2.2' 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.
Updating configuration in CMDaemon... done.
Executing validation test... done.
Installation successfully completed.
Finished.
```

6.7.2 Kafka Removal With `cmhadoop-kafka-setup`

`cmhadoop-kafka-setup` uses the `-u` option to uninstall the Kafka instance.

Example

```
[root@bright72 ~]# cmhadoop-kafka-setup -u hdfs1
Requested removal of Kafka for Hadoop instance hdfs1.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Cleaning ZooKeeper... done.
Removing additional Kafka directories... done.
Removal successfully completed.
Finished.
```

6.8 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.

6.8.1 Pig Installation With `cmhadoop-pig-setup`

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

Prerequisites For Pig Installation, And What Pig Installation Does

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

- A Hadoop instance must already be installed.
- Hadoop can be configured with a single NameNode or NameNode HA, but not with NameNode federation.
- `cmhadoop-pig-setup` installs Pig by default on the active head node. A different node can be specified by using the option `--node`.
- The script creates a dedicated Hadoop Configuration Group for Pig.
- 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/`.
- When installing Pig on a Hadoop instance configured to run on Lustre within Bright Cluster Manager (section 2.4), Pig configuration files are automatically copied to a node that has access to LustreFS (NodeManager). Subsequent operations with Pig should be carried out on that node.

The options for `cmhadoop-pig-setup` are listed on running `cmhadoop-pig-setup -h`.

An Example Run With `cmhadoop-pig-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path in Pig environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright72 ~]# cmhadoop-pig-setup -i hdfs1 -t /tmp/pig-0.16.0.tar.gz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Pig release '0.16.0'
Found Hadoop instance 'hdfs1', release: 2.7.1
Pig configuration files will be written on the head node.
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.
```

6.8.2 Pig Removal With `cmhadoop-pig-setup`

`cmhadoop-pig-setup` uses the `-u` option to uninstall the Pig instance.

Example

```
[root@bright72 ~]# cmhadoop-pig-setup -u hdfs1
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Removing additional Pig directories... done.
Removal successfully completed.
Finished.
```

6.8.3 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. Further ocumentation for Pig can be found at <http://pig.apache.org/docs/r0.16.0/start.html>.

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

```
[root@bright72 ~]# module load hadoop/hdfs1
[root@bright72 ~]# module load pig/hdfs1
[root@bright72 ~]# 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@bright72 ~]# module load hadoop/hdfs1
[root@bright72 ~]# module load pig/hdfs1
[root@bright72 ~]# pig -v -f /tmp/smoke.pig
```

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

6.9 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.

At present, the latest Sqoop stable release is 1.4.6, while the latest Sqoop2 version is 1.99.6. Sqoop2 is incompatible with Sqoop; it is not feature-complete; and it is not yet intended for production use. Sqoop2 is described in section 6.10.

6.9.1 Sqoop Installation With `cmhadoop-sqoop-setup`

Bright Cluster Manager provides `cmhadoop-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 `cmhadoop-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/>
 - `cmhadoop-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 `cmhadoop-sqoop-setup` script installs Sqoop only on the active head node. A different node can be specified by using the option `--master`.
- The script creates a dedicated Hadoop Configuration Group for Sqoop.
- 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.
- When installing Sqoop on a Hadoop instance configured to run on Lustre within Bright Cluster Manager (section 2.4), Sqoop should be deployed on a node that has access to LustreFS (by using the `--master` option if needed). Subsequent operations with Sqoop should be carried out on that node.

The options for `cmhadoop-sqoop-setup` are listed on running `cmhadoop-sqoop-setup -h`.

An Example Run With `cmhadoop-sqoop-setup`

The option `-j <path>` is mandatory. It is used to set the Java Home in Sqoop environment files. The path should point to a Java Development Kit.

Example

```
[root@bright72 ~]# cmhadoop-sqoop-setup -i hdfs1 -j /usr/lib/jvm/java-1.7.0-op\
enjdk.x86_64/ -t /tmp/sqoop-1.4.6.bin__hadoop-2.0.4-alpha.tar.gz --master node005
Using MySQL Connector/J installed in /usr/share/java/
Sqoop release '1.4.6.bin__hadoop-2.0.4-alpha'
Sqoop service will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.7.1
Sqoop being installed... done.
Creating directories for Sqoop... done.
Creating module file for Sqoop... done.
Creating configuration files for Sqoop... done.
Updating images... done.
Updating configuration in CMDaemon... done.
Installation successfully completed.
Finished.
```

6.9.2 Sqoop Removal With `cmhadoop-sqoop-setup`

`cmhadoop-sqoop-setup` uses the `-u` option to remove the Sqoop instance.

Example

```
[root@bright72 ~]# cmhadoop-sqoop-setup -u hdfs1
Requested removal of Sqoop for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Sqoop directories... done.
Removal successfully completed.
Finished.
```

6.9.3 Using Sqoop To Import A Table From MySQL

The following example shows how to import data from the MySQL instance installed on the head node. First, the following statements must be executed explicitly by the administrator, using a MySQL client:

```
GRANT SELECT ON cmdaemon.* TO 'sqoop'@'%%' IDENTIFIED BY 'sqoop';
FLUSH PRIVILEGES;
```

Now that user sqoop has been granted access to MySQL, it's possible to proceed with the import (some lines elided):

```
[root@bright72 ~]# module load sqoop/hdfs1
[root@bright72 ~]# sqoop import --connect jdbc:mysql://hadoop.cm.cluster/cmdae\
mon --username sqoop --password sqoop --table Devices --direct --verbose --fet\
ch-size 0

...

15/12/16 14:17:33 INFO sqoop.Sqoop: Running Sqoop version: 1.4.6
15/12/16 14:17:33 DEBUG tool.BaseSqoopTool: Enabled debug logging.

...

15/12/16 14:17:34 INFO manager.SqlManager: Executing SQL statement: SELECT t.*\
FROM 'Devices' AS t LIMIT 1

...

15/12/16 14:17:34 DEBUG orm.ClassWriter: Writing source file: /tmp/sqoop-root/\
compile/732ef860a1294c9c074a7d963519fe5c/Devices.java
15/12/16 14:17:34 DEBUG orm.ClassWriter: Table name: Devices
15/12/16 14:17:34 DEBUG orm.ClassWriter: Columns: uniqueKey:-5, revision:12, r\
eadonly:-7, tag:12, hostname:12, mac:12, creationTime:-5, partition:-5, ethern\
etSwitch:-5, rack:-5, rackPosition:-5, rackHeight:-5, indexInsideContainer:-5,\
powerControl:12, customPowerScript:12, customPowerScriptArgument:12, customPi\
ngScript:12, customPingScriptArgument:12, notes:-1, userdefined1:12, userdefin\
ed2:12, derivedMonConfId:-5,
15/12/16 14:17:34 DEBUG orm.ClassWriter: sourceFilename is Devices.java
15/12/16 14:17:34 DEBUG orm.CompilationManager: Found existing /tmp/sqoop-root/\
compile/732ef860a1294c9c074a7d963519fe5c/
15/12/16 14:17:34 INFO orm.CompilationManager: HADOOP_MAPRED_HOME is /cm/share\
d/apps/hadoop/hdfs1

...

15/12/16 14:18:04 INFO mapreduce.ImportJobBase: Transferred 875 bytes in 26.16\
34 seconds (33.4437 bytes/sec)
15/12/16 14:18:04 INFO mapreduce.ImportJobBase: Retrieved 7 records.
15/12/16 14:18:04 DEBUG util.ClassLoaderStack: Restoring classloader: sun.misc\
.Launcher$AppClassLoader@21533b2c
```

Sqoop creates a directory inside HDFS, and it saves the result of the import operation, which can be shown with:

```
[root@bright72 ~]# module load hadoop/hdfs1
[root@bright72 ~]# hdfs dfs -cat /user/root/Devices/*
38654705665,,0,00000000a000,switch01,00:00:00:00:00:00,1444115644,21474836481,\
```

```

281474976710713,0,0,1,0,apc,,,,,,,,,81604382722
38654705666,,0,00000000a000,hadoop,FA:16:3E:23:45:3C,1448905041,21474836481,28\
1474976710714,0,0,1,0,apc,,,,,,,,,81604382723
38654705667,,0,00000000a000,node001,FA:16:3E:D7:3F:95,1448905042,21474836481,2\
81474976710742,0,0,1,0,apc,,,,,,,,,81604382724
38654705668,,0,00000000a000,node002,FA:16:3E:FE:8E:EA,1448905041,21474836481,2\
81474976710745,0,0,1,0,apc,,,,,,,,,81604382725
38654705669,,0,00000000a000,node003,FA:16:3E:60:38:36,1448905041,21474836481,2\
81474976710748,0,0,1,0,apc,,,,,,,,,81604382726
38654705670,,0,00000000a000,node004,FA:16:3E:6A:03:D6,1448905042,21474836481,2\
81474976710751,0,0,1,0,apc,,,,,,,,,81604382727
38654705671,,0,00000000a000,node005,FA:16:3E:07:28:C1,1448905042,21474836481,2\
81474976710754,0,0,1,0,apc,,,,,,,,,81604382728

```

6.10 Sqoop2

Sqoop2 is the forthcoming version of Sqoop, designed to support data transfer across any two data sources. Bright Cluster Manager provides `cmhadoop-sqoop-setup` for Sqoop2 installation, as well as for Sqoop installation. The `cmhadoop-sqoop-setup` behavior follows the same pattern as described in the Sqoop section (section 6.9).

An Example Run With `cmhadoop-sqoop-setup`

The option `-j <path>` is mandatory. It is used to set the Java home path in Sqoop2 environment files. The path should point to a Java Development Kit.

Example

```

[root@bright72 ~]# cmhadoop-sqoop-setup -i hdfs1 -j /usr/lib/jvm/java-1.7.0-op\
  enjdk.x86_64/ -t /tmp/sqoop-1.99.6-bin-hadoop200.tar.gz --master node005
Using MySQL Connector/J installed in /usr/share/java/
Sqoop release '1.99.6-bin-hadoop200'
Sqoop service will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.7.1
Sqoop being installed... done.
Creating directories for Sqoop... done.
Creating module file for Sqoop... done.
Creating configuration files for Sqoop... done.
Updating images... done.
Updating configuration in CMDaemon... done.
Validating Sqoop setup... done.
Installation successfully completed.
Finished.

```

6.10.1 Sqoop2 Removal With `cmhadoop-sqoop-setup`

`cmhadoop-sqoop-setup` uses the `-u` option to remove the Sqoop2 instance.

Example

```

[root@bright72 ~]# cmhadoop-sqoop-setup -u hdfs1
Requested removal of Sqoop for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Sqoop directories... done.
Removal successfully completed.
Finished.

```

6.11 Storm

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

Other comparisons 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

6.11.1 Storm Installation With `cmhadoop-storm-setup`

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

Prerequisites For Storm Installation, And What Storm Installation Does

The following applies to using `cmhadoop-storm-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 `cmhadoop-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 creates two dedicated Hadoop Configuration Groups for Storm: one for the Storm Nimbus and one for Storm Supervisors.
- 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.

The options for `cmhadoop-storm-setup` are listed on running `cmhadoop-storm-setup -h`.

An Example Run With `cmhadoop-storm-setup`

The option `-j <path>` is not mandatory. It is used to set the Java Home in Storm environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright72 ~]# cmhadoop-storm-setup -i hdfs1 -t /tmp/apache-storm-0.10.0.tar.gz \
--nimbus node005
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Storm release '0.10.0'
Storm Nimbus and UI services will be run on node: node005
Found Hadoop instance 'hdfs1', release: 2.7.1
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... done.
Initializing Nimbus services for Storm... done.
Updating configuration in CMDaemon... done.
Executing validation test... done.
Installation successfully completed.
Finished.

```

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

6.11.2 Storm Removal With `cmhadoop-storm-setup`

`cmhadoop-storm-setup` uses the `-u` option to remove the Storm instance.

Example

```

[root@bright72 ~]# cmhadoop-storm-setup -u hdfs1
Requested removal of Storm for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Cleaning ZooKeeper... done.
Removing additional Storm directories... done.
Removal successfully completed.
Finished.

```

6.11.3 Using Storm

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

```

[root@bright72 ~]# module load storm/hdfs1
[root@bright72 ~]# storm jar $STORM_BASE_DIR/examples/storm-starter/\
storm-starter-topologies-*.jar\
storm.starter.WordCountTopology WordCount2

...

638 [main] INFO b.s.u.Utils - Using defaults.yaml from resources
745 [main] INFO b.s.u.Utils - Using storm.yaml from resources
819 [main] INFO b.s.u.Utils - Using defaults.yaml from resources
847 [main] INFO b.s.u.Utils - Using storm.yaml from resources
851 [main] INFO b.s.StormSubmitter - Generated ZooKeeper secret payload for MD5-\
digest: -6720275370401821887:-5556780662562789347
853 [main] INFO b.s.s.a.AuthUtils - Got AutoCreds []
871 [main] INFO b.s.u.StormBoundedExponentialBackoffRetry - The baseSleepTimeMs \
[2000] the maxSleepTimeMs [60000] the maxRetries [5]
884 [main] INFO b.s.u.StormBoundedExponentialBackoffRetry - The baseSleepTimeMs \
[2000] the maxSleepTimeMs [60000] the maxRetries [5]
914 [main] INFO b.s.u.StormBoundedExponentialBackoffRetry - The baseSleepTimeMs \
[2000] the maxSleepTimeMs [60000] the maxRetries [5]
921 [main] INFO b.s.StormSubmitter - Uploading topology jar /cm/shared/apps/hado\
op/Apache/apache-storm-0.10.0/examples/storm-starter/storm-starter-topologies-0.1\

```

```

0.0.jar to assigned location: /tmp/storm-hdfs1-local/nimbus/inbox/stormjar-6b72206\
e-237a-4aca-8d1d-dcf1e93ec2f1.jar
Start uploading file '/cm/shared/apps/hadoop/Apache/apache-storm-0.10.0//examples/\
storm-starter/storm-starter-topologies-0.10.0.jar' to '/tmp/storm-hdfs1-local/nimb\
us/inbox/stormjar-6b72206e-237a-4aca-8d1d-dcf1e93ec2f1.jar' (3305718 bytes)
[=====] 3305718 / 3305718
File '/cm/shared/apps/hadoop/Apache/apache-storm-0.10.0//examples/storm-starter/st\
orm-starter-topologies-0.10.0.jar' uploaded to '/tmp/storm-hdfs1-local/nimbus/inbo\
x/stormjar-6b72206e-237a-4aca-8d1d-dcf1e93ec2f1.jar' (3305718 bytes)
1002 [main] INFO b.s.StormSubmitter - Successfully uploaded topology jar to assign\
ed location: /tmp/storm-hdfs1-local/nimbus/inbox/stormjar-6b72206e-237a-4aca-8d1d\
-dcf1e93ec2f1.jar
1002 [main] INFO b.s.StormSubmitter - Submitting topology WordCount2 in distribut\
ed mode with conf "storm.zookeeper.topology.auth.scheme":"digest","storm.zookeeper\
.topology.auth.payload":"-6720275370401821887:-5556780662562789347","topology.work\
ers":3,"topology.debug":true
1168 [main] INFO b.s.StormSubmitter - Finished submitting topology: WordCount2

[root@bright72 ~]# storm list

...

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

```

6.12 Tachyon

Tachyon is a memory-centric distributed storage system. It lies between computation frameworks and various storage systems, in order to enable reliable data sharing across cluster jobs.

The Tachyon tarball should be built from sources, since it depends on the Hadoop distribution and version.

```

[foobar@bright72 ~]$ git clone git://github.com/amplab/tachyon.git
[foobar@bright72 ~]$ cd tachyon/
[foobar@bright72 ~]$ mvn -Dhadoop.version=2.7.1 -DskipTests package assembly:single

```

The tarball needed for the installation can be found in `assembly/target`.

6.12.1 Tachyon Installation With `cmhadoop-tachyon-setup`

Bright Cluster Manager provides `cmhadoop-tachyon-setup` to carry out Tachyon installation:

Prerequisites For Tachyon Installation, And What Tachyon Installation Does

The following applies to using `cmhadoop-tachyon-setup`:

- A Hadoop instance must already be installed.
- `cmhadoop-tachyon-setup` installs Tachyon only on the active head node and on the DataNodes of the chosen Hadoop instance.
- The script sets the under storage address to the HDFS namenode address
- The script creates no roles for for Tachyon.
- Tachyon is copied by the script to a subdirectory under `/cm/shared/hadoop/`

- Tachyon configuration files are copied by the script to under `/etc/hadoop/`
- `cmhadoop-tachyon-setup` creates a configuration overlay (section 3.1.9), for example: `hdfs1-tachyon`, comprising the nodes for the Hadoop instance. The overlay contains a customization (section A.6) that is used to set `HADOOP_CLASSPATH` to the path of the Tachyon JAR in `hadoop-env.sh`. For example:
`HADOOP_CLASSPATH=/cm/shared/apps/hadoop/Apache/tachyon-0.9.0-SNAPSHOT/\`
`assembly/target/tachyon-assemblies-0.9.0-SNAPSHOT-jar-with-dependencies.jar.`

The options for `cmhadoop-tachyon-setup` are listed on running `cmhadoop-tachyon-setup -h`.

An Example Run With `cmhadoop-tachyon-setup`

The option `-j <path>` is not mandatory. It is used to set the Java home path in Tachyon environment files. If the option is not specified, the script uses the value retrieved from the Hadoop instance.

Example

```
[root@bright72 ~]# cmhadoop-tachyon-setup -i hdfs1 -t /root/tachyon/2.7.1/tachyon\
-0.9.0-SNAPSHOT.tar.gz
Java home not specified, using: /usr/lib/jvm/jre-1.7.0-openjdk.x86_64/
Tachyon release '0.9.0-SNAPSHOT'
Found Hadoop instance 'hdfs1', release: 2.7.1
Tachyon Master will be run on the head node.
Tachyon being installed... done.
Creating directories for Tachyon... done.
Creating module file for Tachyon... done.
Creating configuration files for Tachyon... done.
Updating images... done.
Initializing Tachyon directory in HDFS... done.
Updating Hadoop configuration... done.
Formatting Tachyon FS... done.
Waiting for NameNode to be ready... done.
Initializing services for Tachyon... done.
Validating Tachyon setup... done.
Installation successfully completed.
Finished.
```

6.12.2 Tachyon Removal With `cmhadoop-tachyon-setup`

`cmhadoop-tachyon-setup` uses the `-u` option to remove the Tachyon instance. The corresponding configuration overlay (section 3.1.9), for example, `hdfs1-tachyon`, is removed.

Example

```
[root@bright72 ~]# cmhadoop-tachyon-setup -u hdfs1
Requested removal of Tachyon for Hadoop instance 'hdfs1'.
Stopping/removing services... done.
Removing module file... done.
Removing configuration directory... done.
Updating images... done.
Removing additional Tachyon directories... done.
Removal successfully completed.
Finished.
```

6.12.3 Using Tachyon

Tachyon consists of several components: one Master, multiple Workers, and an executable client, `tachyon`, that can be run after the user loads the corresponding module. The following example shows how to execute a MapReduce job, wordcount, on data copied to the Tachyon storage layer.

Example

```
[root@bright72 ~]# module load tachyon/hdfs1
[root@bright72 ~]# tachyon tfs copyFromLocal /cm/local/examples/hamlet.txt\
/wordcount/hamlet.txt
Copied /cm/local/examples/hamlet.txt to /wordcount/hamlet.txt
[root@bright72 ~]# module load hadoop/hdfs1/Apache/2.7.1
[root@bright72 ~]# hadoop jar /cm/shared/apps/hadoop/hdfs1/share/hadoop/mapreduce/\
hadoop-mapreduce-examples-*.jar wordcount -libjars /cm/shared/apps/hadoop/Apache/t\
achyon-0.9.0-SNAPSHOT//assembly/target/tachyon-assemblies-0.9.0-SNAPSHOT-jar-with-\
dependencies.jar -Dtachyon.user.file.understoragetype.default=SYNC_PERSIST tachyon\
://hadoop.cm.cluster:19998/wordcount/input.txt tachyon://hadoop.cm.cluster:19998/w\
ordcount/outputham

...

15/12/23 16:01:37 INFO : initialize(tachyon://hadoop.cm.cluster:19998/wordcount/in\
put.txt, Configuration: core-default.xml, core-site.xml, mapred-default.xml, mapre\
d-site.xml, yarn-default.xml, yarn-site.xml, hdfs-default.xml, hdfs-site.xml). Con\
necting to Tachyon: tachyon://hadoop.cm.cluster:19998/wordcount/input.txt
15/12/23 16:01:37 INFO : Loading Tachyon properties from Hadoop configuration: ta\
chyon.user.file.understoragetype.default=SYNC_PERSIST
15/12/23 16:01:37 INFO : Tachyon client (version 0.9.0-SNAPSHOT) is trying to conn\
ect with FileSystemMasterClient master @ hadoop.cm.cluster/10.141.255.254:19998
15/12/23 16:01:37 INFO : Client registered with FileSystemMasterClient master @ ha\
dooop.cm.cluster/10.141.255.254:19998
15/12/23 16:01:37 INFO : tachyon://hadoop.cm.cluster:19998 tachyon://hadoop.cm.clu\
ster:19998 hdfs://node001.cm.cluster:8020/tachyon

...

15/12/23 16:01:50 INFO mapreduce.Job: Job job_1450867470638_0008 running in uber m\
ode : false
15/12/23 16:01:50 INFO mapreduce.Job: map 0% reduce 0%
15/12/23 16:01:59 INFO mapreduce.Job: map 100% reduce 0%
15/12/23 16:02:10 INFO mapreduce.Job: map 100% reduce 100%
15/12/23 16:02:11 INFO mapreduce.Job: Job job_1450867470638_0008 completed success\
fully

...

[root@bright72 ~]# tachyon tfs cat /wordcount/outputham/*
'Tis 1
'tis 1
And 5

...

would 2
wrong,1
you 1
```



Details And Examples Of Hadoop Configuration

This appendix supplements section 3.1.9's introduction to Hadoop/Sqoop configuration under Bright Cluster Manager.

A.1 Hadoop Components Activation And Deactivation Using Roles

Hadoop components such as HDFS or YARN are activated and deactivated using roles. Bright Cluster Manager 7.2 includes 18 possible roles representing possible Hadoop- or Spark-related service, at the time of writing (August 2015).

For example, assigning the HadoopNameNode role to a node makes the node store HDFS meta-data, and be in control of HDFS datanodes that store the actual data in HDFS. Similarly, assigning the DataNode role to a node makes it serve as an HDFS datanode.

A.2 Only The Enabled Hadoop Components And Roles Are Available For Activation From `cmgui` And `cmsh`

Bright Cluster Manager version 7.1 introduced *configuration overlays* (section 3.1.9) to deal with the challenges in configuring Hadoop/Spark components, such as large number of configuration parameters, flexible assignment of services to groups of nodes, and so on. Configuration overlays are the main way of configuring Hadoop- or Spark-related components.

For a given Hadoop cluster instance only a subset of the Hadoop/Spark roles shown in table 3.1.9 is available to the cluster administrator. The actual set of enabled and disabled roles depends on a chosen Hadoop distribution, on the configuration mode (for example HDFS HA *versus* HDFS non-HA) (page 18) and on the Hadoop-components that are actually selected during the installation procedure.

Example

Hadoop 1.x installation, with HDFS High Availability with manual failover (section 2.3), and with the HBase datastore component, enables and disables the roles indicated by the following table:

| Enabled | Disabled |
|---------------------|---------------------------|
| Hadoop::NameNode | Hadoop::SecondaryNameNode |
| Hadoop::DataNode | |
| Hadoop::Journal | |
| Hadoop::JobTracker | Hadoop::YARNServer |
| Hadoop::TaskTracker | Hadoop::YARNClient |
| Hadoop::HBaseServer | |
| Hadoop::HBaseClient | |
| Hadoop::Zookeeper | |

Among the disabled roles are two YARN roles. This is because YARN resource manager is a part of Hadoop 2.x distributions.

A.3 Example Of Role Priority Overrides In Configuration Groups With `cmsh`

Configuration groups and role priorities are introduced in section 3.1.9. A summary of some of the important points from there is:

- A role can be directly assigned to a node. The fixed priority for the assignment is then 750.
- A role can be assigned to a node via a category to which the node belongs to. The fixed priority for the assignment is then 250.
- A role can be assigned to a node via a Hadoop configuration group. The default priority for a configuration group is 500, but can be set to any integer from -1 to 1000, except for the values 250 and 750. The values 250 and 750 are reserved for category assignment and for direct role assignment respectively. A priority of -1 disables a Hadoop configuration group.

Thus, due to priority considerations, the configuration of a role assigned via a Hadoop configuration group by default overrides configuration of a role assigned via a category. In turn, a role assigned directly to via node a node assignment overrides the category role and default Hadoop configuration group role.

To illustrate role priorities further, an example Hadoop configuration group, `examplehcg`, is created for an existing Hadoop instance `doop`. For the instance, from within `cmsh`, four Hadoop roles are set for five nodes, and their configuration overlay priority is set to 400 as follows (some text omitted):

Example

```
[bright72]% configurationoverlay
[bright72->configurationoverlay]% add examplehcg
...verlay*[examplehcg*]]% set nodes node001..node005
...verlay*[examplehcg*]]% set priority 400
...verlay*[examplehcg*]]% roles
...verlay*[examplehcg*]->roles]% assign hadoop::datanode
...amplehcg*]->roles*[Hadoop::DataNode*]]% assign hadoop::yarnclient
...amplehcg*]->roles*[Hadoop::YARNClient*]]% assign hadoop::hbaseclient
...amplehcg*]->roles*[Hadoop::HBaseClient*]]% assign hadoop::zookeeper
...amplehcg*]->roles*[Hadoop::ZooKeeper*]]% commit
...
[hadoopdev->configurationoverlay]% list
Name (key) Pri Nodes      Cat Roles
-----
examplehcg 400 node001.. node005      Hadoop::DataNode, Hadoop::YARNClient,
                                     Hadoop::HBaseClient, Hadoop::ZooKeeper
```

Next, the following role assignments:

- Hadoop::HBaseClient to the default category default
- Hadoop::DataNode directly to node002 and node003
- Hadoop::HBaseClient directly to node005

can be carried out in cmsh as follows:

Example

```
[bright72->category]% !#check if nodes in default category first
[bright72->category]% listnodes default
Type                Hostname ...
-----
PhysicalNode        node001 ...
PhysicalNode        node002 ...
PhysicalNode        node003 ...
PhysicalNode        node004 ...
PhysicalNode        node005 ...
PhysicalNode        node006 ...
PhysicalNode        node007 ...
[bright72->category]% use default
[bright72->category[default]]% roles; assign hadoop::hbaseclient; commit
...
[bright72]% device; use node002
[bright72->device[node002]]% roles; assign hadoop::datanode; commit
[bright72]% device; use node003
[bright72->device[node003]]% roles; assign hadoop::datanode; commit
[bright72]% device; use node005
[bright72->device[node005]]% roles; assign hadoop::hbaseclient; commit
```

An overview of the configuration with the `overview` command with the `-verbose` option then shows the sources of the roles, in order of priority (some text omitted and reformatted for clarity):

```
[bright72->hadoop]% overview -v doop
Parameter      Value
-----
Name           doop
...
Hadoop role    Node      Source
-----
Hadoop::DataNode node001  overlay:examplehcg
Hadoop::DataNode node002  node002 [750], overlay:examplehcg [400]
Hadoop::DataNode node003  node003 [750], overlay:examplehcg [400]
Hadoop::DataNode node004  overlay:examplehcg
Hadoop::DataNode node005  overlay:examplehcg

Hadoop::HBaseClient node001  overlay:examplehcg [400], category:default [250]
Hadoop::HBaseClient node002  overlay:examplehcg [400], category:default [250]
Hadoop::HBaseClient node003  overlay:examplehcg [400], category:default [250]
Hadoop::HBaseClient node004  overlay:examplehcg [400], category:default [250]
Hadoop::HBaseClient node005  node005 [750], overlay:examplehcg [400], category:default [250]
Hadoop::HBaseClient node006  category:default
Hadoop::HBaseClient node007  category:default
```

```

Hadoop::YARNClient      node001  overlay:examplehcg
Hadoop::YARNClient      node002  overlay:examplehcg
Hadoop::YARNClient      node003  overlay:examplehcg
Hadoop::YARNClient      node004  overlay:examplehcg
Hadoop::YARNClient      node005  overlay:examplehcg

Hadoop::ZooKeeper       node001  overlay:examplehcg
Hadoop::ZooKeeper       node002  overlay:examplehcg
Hadoop::ZooKeeper       node003  overlay:examplehcg
Hadoop::ZooKeeper       node004  overlay:examplehcg
Hadoop::ZooKeeper       node005  overlay:examplehcg
...

```

The logic behind the results of the preceding setup is as follows:

- The `Hadoop::HBaseClient`, `Hadoop::YARNClient`, and `Hadoop::Zookeeper` roles are first assigned at configuration overlay level to `node001..node005`. These roles initially take the altered preset priority of 400 instead of the default of 500, and are active for these nodes, unless overridden by changes further on.
- The `Hadoop::HBaseClient` role is assigned from category level to `node001..node007`. The role on the nodes takes on a priority of 250, and because of that cannot override the configuration overlay role for `node001..node005`. The role is active at this point for `node006` and `node007`.
- Next, the `Hadoop::DataNode` role is assigned directly from node level to `node002` and `node003`. The role on the nodes take on a priority of 750. The value of 400 from the `examplehcg` configuration group assignment is overridden. However, the `Hadoop::DataNode` configuration of `examplehcg` still remains valid for `node001`, `node004`, `node005` so far.
- Then, the `Hadoop::HBaseClient` role is assigned directly from node level to `node005`. The role on the node takes on a priority of 750. The value of 400 for the role from the `examplehcg` configuration is overridden for this node too.

A.4 Cloning Hadoop Configuration Groups In `cmgui` And `cmsh`

Hadoop contains many components, which results in many corresponding Bright Cluster Manager roles. The huge number of configurable parameters for these components results in an unfeasibly large number of settings—more than 220—for configuring Hadoop/Spark.

For ease of use, it is expected that most Hadoop management and configuration operations are carried out with the `cmgui` front end (section 3.1), rather than with the `cmsh` front end (section 3.2). This is because `cmgui` displays Hadoop-related configurations in a more user-friendly manner than `cmsh`.

The `cmsh` front end, however, provides full access to the management capabilities of Bright Cluster Manager. In terms of the number of roles and types of roles to be assigned, `cmsh` is more flexible than `cmgui` because:

- it allows a Hadoop configuration group (configuration overlay) to be created with zero roles
- it allows any available role in Bright Cluster Manager to be assigned. These roles can be outside of Hadoop- or Spark-related roles.

The cloning operations of Hadoop using `cmgui` are covered first in this section A.4.1. The same operations using `cmsh` are described afterwards, in section A.4.2.

A.4.1 Cloning Hadoop Configuration Groups In `cmgui`

In the following example, the `cmgui` front end is used to manage the Hadoop cluster instance shown in figure A.1.

| Modified | Configuration group | Hadoop roles | Priority | Nodes in configuration group |
|----------|--------------------------|---------------------------------|----------|------------------------------|
| | hadoop-test-DN-default | HDFS DataNode, MRv1 TaskTracker | 500 | node003..node006 |
| | hadoop-test-HBM-default | HBase MasterServer | 500 | node002 |
| | hadoop-test-HBRS-default | HBase RegionServer | 500 | node003..node006 |
| | hadoop-test-JT-default | MRv1 JobTracker | 500 | node002 |
| | hadoop-test-NN-default | HDFS NameNode | 500 | node001 |
| | hadoop-test-SNN-default | HDFS SecondaryNameNode | 500 | node003 |
| | hadoop-test-ZK-default | Zookeeper | 500 | node003..node005 |

Buttons at the bottom: Open, New, Clone, Remove, Revert, Save.

Figure A.1: Hadoop Configuration Group tab in cmgui

For this cluster, a situation is imagined where the nodes `node005` and `node006` suddenly experience an extra, non-Hadoop-related, memory-intensive workload, while the remaining nodes `node003` and `node004` are fully dedicated to Hadoop usage. In that case it makes sense to reduce the memory that Hadoop requires for `node005` and `node006`. The MapReduce TaskTracker services on `node005` and `node006` could have their memory parameters reduced, such as the Java heap size, max map tasks number, and so on. At the same time, the configurations of HDFS DataNodes on these two nodes should be left alone. These requirements can be achieved as follows:

- The `hadoop-test-DN-default` configuration group can be cloned with the `Clone` button in the Hadoop Configurations Groups tab. An editing window 'Clone Hadoop Configuration Group' pops up with a new, cloned-from-`hadoop-test-DN-default` group. It gets a default suffix of `'-cloned'`.
- The nodes in the cloned configuration group are set to `node005` and `node006`.
- The HDFS DataNode role is removed from the configuration group. In this particular example, the DataNode role might also be left as is.
- The priority of the group should be checked to see that it is set to higher than that of `hadoop-test-DN-default`. By default, a cloned group is set to the priority of the parent group, plus 10.
- Lower values are set for relevant TaskTracker configuration parameters. In this case, the Java heap size value within TaskTracker can be reduced. Figures A.2 and A.3 show the original state of the configuration group before clicking on the `Clone` button, and the cloned state after reducing the memory-related parameters.

Edit Hadoop Configuration Group

Configuration Group:

Priority: ⓘ

Nodes in Configuration Group: [Add/remove nodes](#)

Configure HDFS DataNode | **Configure MRv1 TaskTracker**

Settings applied to:

TaskTracker Web UI port: TaskTracker Java heap size: MB

HTTP threads count: JVM settings: ☐ ☐ ☐

Maximum map tasks: Number of tasks per JVM:

Maximum reduce tasks: Map JVM options:

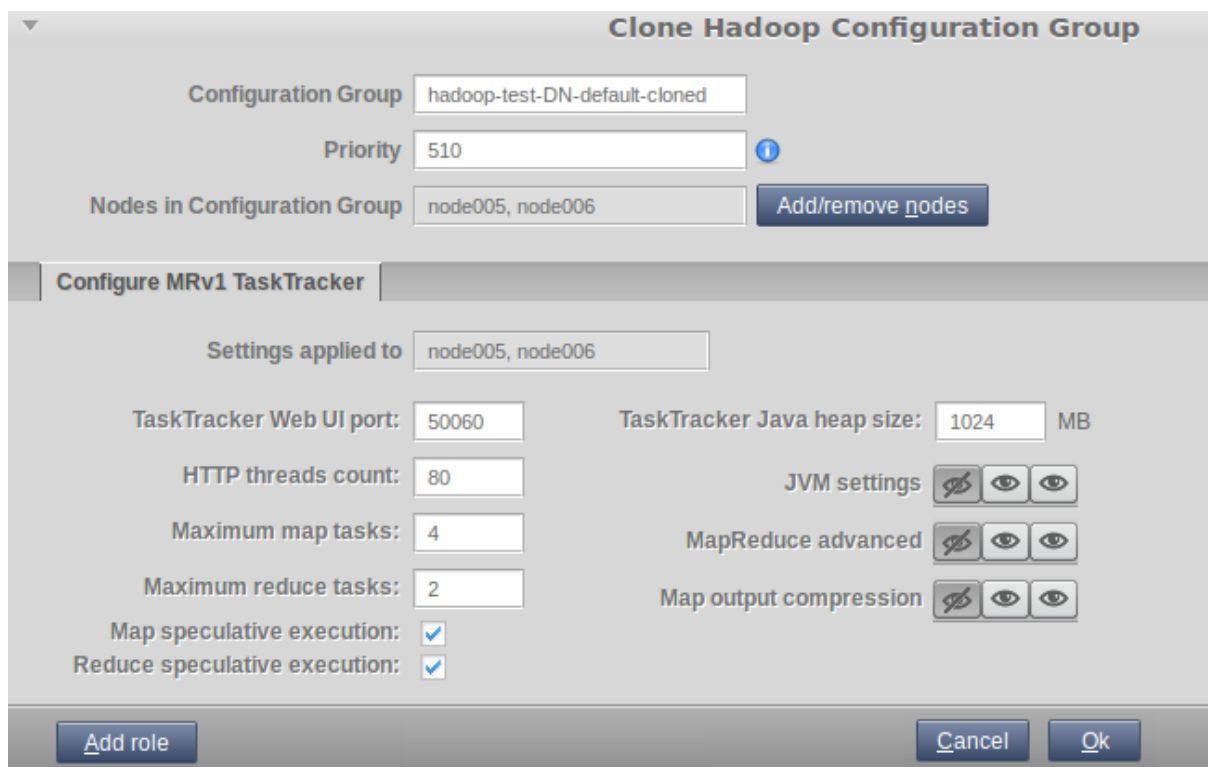
Map speculative execution: ☒ Reduce JVM options:

Reduce speculative execution: ☒ MapReduce advanced: ☐ ☐ ☐

Map output compression: ☐ ☐ ☐

[Add role](#) [Remove role](#) [Cancel](#) [Ok](#)

Figure A.2: Hadoop Configuration Group Prior To Cloning



Clone Hadoop Configuration Group

Configuration Group:

Priority: ⓘ

Nodes in Configuration Group: [Add/remove nodes](#)

Configure MRv1 TaskTracker

Settings applied to:

TaskTracker Web UI port: TaskTracker Java heap size: MB

HTTP threads count: JVM settings: ☐ ☐ ☐

Maximum map tasks: MapReduce advanced: ☐ ☐ ☐

Maximum reduce tasks: Map output compression: ☐ ☐ ☐

Map speculative execution: ☒

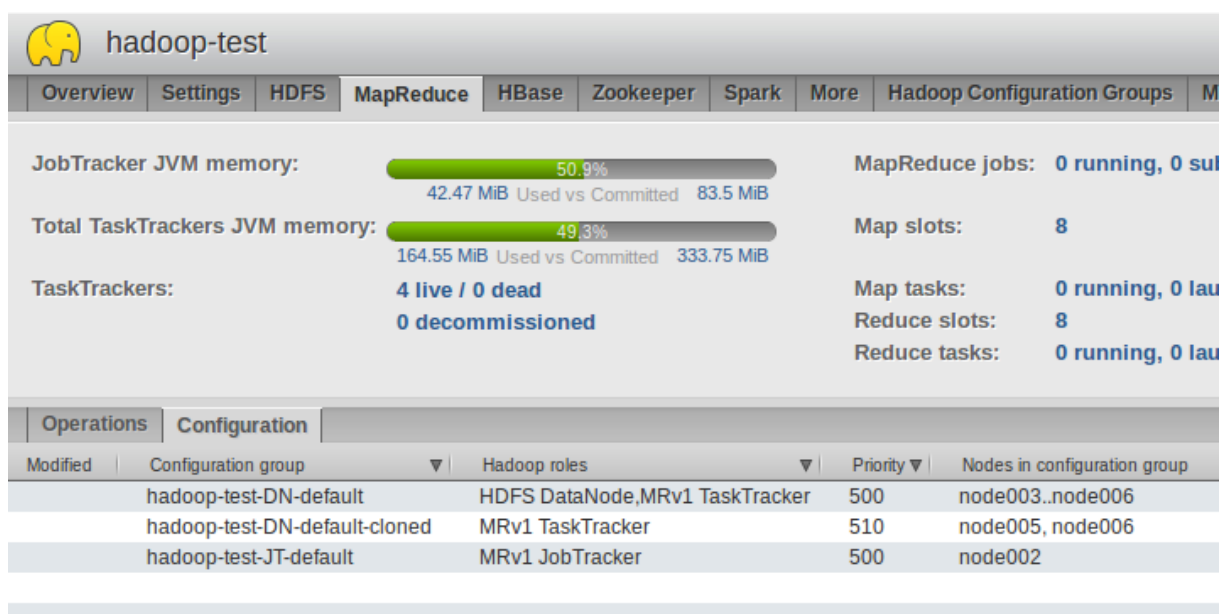
Reduce speculative execution: ☒

[Add role](#) [Cancel](#) [Ok](#)

Figure A.3: Example Of Cloned Hadoop Configuration Group

- The cloned Hadoop configuration group and all the changes to it should be saved, by clicking on the OK button of the edit window, then on the Save button of the parent Hadoop Configuration Groups window.

As a result of these changes, Bright Cluster Manager restarts MapReduce TaskTracker service with the configuration settings that are defined in `hadoop-test-DN-default-cloned`. MapReduce in figure A.4 compared with before now displays one more Hadoop configuration group—the cloned group.



hadoop-test

Overview Settings HDFS **MapReduce** HBase Zookeeper Spark More Hadoop Configuration Groups M

JobTracker JVM memory: 50.9%
42.47 MiB Used vs Committed 83.5 MiB

Total TaskTrackers JVM memory: 49.8%
164.55 MiB Used vs Committed 333.75 MiB

TaskTrackers: **4 live / 0 dead**
0 decommissioned

MapReduce jobs: **0 running, 0 sul**

Map slots: **8**

Map tasks: **0 running, 0 lau**

Reduce slots: **8**

Reduce tasks: **0 running, 0 lau**

| Modified | Configuration group | Hadoop roles | Priority | Nodes in configuration group |
|----------|-------------------------------|---------------------------------|----------|------------------------------|
| | hadoop-test-DN-default | HDFS DataNode, MRv1 TaskTracker | 500 | node003..node006 |
| | hadoop-test-DN-default-cloned | MRv1 TaskTracker | 510 | node005, node006 |
| | hadoop-test-JT-default | MRv1 JobTracker | 500 | node002 |

Figure A.4: Hadoop Configuration Groups for MapReduce after example configuration

There is no imposed limit on the number of Hadoop configuration groups that can be used for a given Hadoop cluster instance. For large numbers, it can be difficult to see which configurations from which groups are actually applied to nodes or sets of nodes.

To help with that, the Hadoop Configuration Groups display window (figure A.1) displays updated information on the roles and configuration groups that are applied to the nodes. For example, the MapReduce TaskTracker defined in `hadoop-test-DN-default-cloned` has the Settings applied to field in figure A.3, where `node005` and `node006` are listed. These nodes are displayed in the Hadoop Configuration Groups display window right away.

Also at the same time, the nodes in `hadoop-test-DN-default` have changed. The role settings for its TaskTracker nodes are now applied only to `node003` and `node004`. These changes are also displayed in the Hadoop Configuration Groups display window right away.

A.4.2 Cloning Hadoop Configuration Groups In `cmsh`

The following session discusses the cloning operation that is described in section A.4.1 once more. Only this time, it is done using `cmsh` rather than `cmgui` (some text omitted for clarity):

Example

```
[hadoopdev]% configurationoverlay
[hadoopdev->configurationoverlay]% list
Name (key)                Pri Nodes                Roles
-----
hadoop-test-DN-default    500 node003..node006 Hadoop::DataNode, Hadoop::
hadoop-test-HBM-default   500 node002              Hadoop::HBaseServer
hadoop-test-HBRS-default  500 node003..node006 Hadoop::HBaseClient
hadoop-test-JT-default    500 node002              Hadoop::JobTracker
hadoop-test-NN-default    500 node001              Hadoop::NameNode
hadoop-test-SNN-default   500 node003              Hadoop::SecondaryNameNode
hadoop-test-ZK-default    500 node003..node005 Hadoop::ZooKeeper
[...overlay]% clone hadoop-test-dn-default hadoop-test-dn-default-cloned
[...overlay*[hadoop-test-dn-default-cloned*]]% set priority 510
[...hadoop-test-dn-default-cloned*]]% roles; unassign hadoop::datanode
[...overlay*[hadoop-test-dn-default-cloned*]]% commit
[...overlay[hadoop-test-dn-default-cloned]->roles]% list
Name (key)
-----
Hadoop::TaskTracker
[...fault-cloned]->roles]% use hadoop::tasktracker; configurations; list
HDFS
-----
hadoop-test
[->roles[Hadoop::TaskTracker]->configurations]% use hadoop-test; show
Parameter                Value
-----
File merging number       32
HDFS                      hadoop-test
HTTP port                 50060
...
Map speculative execution yes
Maximum map tasks         8
...
TaskTracker heap size     2048
Type                      HadoopTaskTrackerHDFSConfiguration
[...ker]->configurations[hadoop-test]]% set tasktrackerheapsize 1024
[...ker*]->configurations*[hadoop-test*]]% set maximummaptasks 4; commit
```

The result of this is the Hadoop configuration group `hadoop-test-DN-default-cloned`, which is seen in the `cmgui` equivalent in figure A.3.

A.5 Considerations And Best Practices When Creating Or Cloning Hadoop Configurations

The `cmgui` front end is the recommended way to carry out Hadoop configuration operations, and for installing, configuring and managing the Hadoop cluster instances. The following are considerations and best practices:

- Naming conventions: It is recommended to start a name for a new or cloned Hadoop configuration group with the name of the Hadoop cluster instance. This is automatically done for the default Hadoop configuration groups created during Hadoop installation.
- A Hadoop configuration group can include zero nodes, but it has to have at least one role assigned. An exception to this is that the `cmsh` front end allows a user to create a Hadoop configuration group with no roles assigned, but such a group cannot be connected to any Hadoop instance, and such groups are therefore not displayed in `cmgui`.
- If a Hadoop configuration group has no roles assigned to it, then it can be seen only via the `configurationoverlay` mode of `cmsh`.
- Hadoop configuration groups that are not in use should be disabled using `-1` as a priority value. If the configuration group is disabled, then the configurations in all roles, for all nodes in this group, will no longer be used. Instead the next highest priority configuration will be used.
- A history of configuration changes can be tracked using the cloning functionality. For example, the parent group can be the configuration group that always has the current configuration. A list of groups with earlier configurations can then be kept, where each is derived from a parent by cloning it, and setting its priority to `-1`, and also including the timestamp (for example, `YYYYMMDD`, for easy sorting) in its name:

Example

```
hadoop-config[500]
hadoop-config-cloned-20150514[-1]
hadoop-config-cloned-20141104[-1]
hadoop-config-cloned-20131008[-1]
...
```

- Hadoop/Spark roles that correspond to key Hadoop services (the asterisked services in table 3.1.9) are deliberately not provided by `cmgui` or `cmsh` as options for addition or removal when editing or creating a Hadoop configuration group. This is done because of the risk of data loss if the key services are misconfigured.

A workaround for this restriction is that a configuration group with a key Hadoop role can be cloned. The cloned group, which includes the service, can then be built upon further.

- A Hadoop configuration group is associated with a Hadoop instance if it has at least one role with a configuration linked to that Hadoop instance. For example, the following commands investigate the `hadoop-test-dn-default` group. The Hadoop cluster instances for which the MapReduce TaskTracker role configurations are defined are shown:

```
[hadoopdev]% configurationoverlay; use hadoop-test-dn-default; roles
[hadoopdev->configurationoverlay[hadoop-test-DN-default]->roles]%
```

```
[...-DN-default]->roles]% use hadoop::tasktracker; configurations; list
HDFS
-----
hadoop-test
```

- Assignment of Hadoop or Spark-related roles directly to nodes or to node categories should be avoided. Hadoop configuration groups (configuration overlays) should be used instead.

If the setup can benefit from the direct assignment of roles to nodes or to categories, then the administrator should be aware of priorities and their outcome for role assignments that overlay each other (example in section A.3).

A.6 Customizations For Configuration Overlays

Configuration overlays (section 3.1.9) have a feature called *customizations* which have been introduced in Bright Cluster Manager version 7.2.

A customization in the configuration overlay sense is a set of modifications applicable to a file. A configuration overlay may contain multiple customizations so that several files are modified by one overlay.

Customizations are useful for Big Data and for OpenStack, where new plugin installations often require this kind of flexibility.

If a given configuration can however be managed using roles and configuration overlays, then using those is recommended instead of using customizations.

A.6.1 Customization Example Overview

Original Configuration File

For Hadoop, a typical configuration file for an instance `hdfs1` might be at `/etc/hadoop/hdfs1/hdfs-site.xml`, containing key-value properties as follows:

```
<property>
  <name>mapreduce.job.map.output.collector.class</name>
  <value>org.apache.hadoop.mapred.MapTask$MapOutputBuffer</value>
</property>

<property>
  <name>mapreduce.job.reduce.shuffle.consumer.plugin.class</name>
  <value>org.apache.hadoop.mapreduce.task.reduce.Shuffle</value>
</property>

<property>
  <!-- automatically managed by cmdaemon -->
  <name>dfs.datanode.data.dir</name>
  <value>/var/lib/hadoop/hdfs1/datanode</value>
  <final>true</final>
</property>
```

The last property is already managed by Bright Cluster Manager. Using customizations to change this property section is therefore not recommended, and the appropriate `DataNode` role (section 3.1.9) should be used instead.

Original Customization State

Bright Cluster Manager reads a customization file specification, then modifies the configuration file, then writes the modified configuration files to disk, and then restarts the dependent services.

Bright Cluster Manager recognizes whether the file is managed by OpenStack or Hadoop. The manager for the customization is indicated by the entry under the `Manager` heading of the `customizationoverview` command in `cmsh`:

```
[cluster1->configurationoverlay[hdfs1-DataNode]]% customizationsoverview
```

| Node | File [Type] | Customization | Conf. Overlay [Prio] | Manager |
|---------|---------------------|---------------------------|----------------------|---------|
| node001 | hdfs-site.xml [XML] | mapreduce.job.map.output+ | hdfs1-DataNode [500] | hadoop |
| node002 | hdfs-site.xml [XML] | mapreduce.job.map.output+ | hdfs1-DataNode [500] | hadoop |
| node003 | hdfs-site.xml [XML] | mapreduce.job.map.output+ | hdfs1-DataNode [500] | hadoop |
| node001 | hdfs-site.xml [XML] | mapreduce.job.reduce.shu+ | hdfs1-DataNode [500] | hadoop |
| node002 | hdfs-site.xml [XML] | mapreduce.job.reduce.shu+ | hdfs1-DataNode [500] | hadoop |
| node003 | hdfs-site.xml [XML] | mapreduce.job.reduce.shu+ | hdfs1-DataNode [500] | hadoop |

The customization strings in the output to `customizationsoverview` displayed in the preceding transcript are abbreviations of the following key=value settings:

```
mapreduce.job.map.output.collector.class = org.apache.hadoop.mapred.MyFirstPlugin$MapOutputBuffer
and
```

```
mapreduce.job.reduce.shuffle.consumer.plugin.class = org.apache.hadoop.mapred.MyFirstPlugin$Shuffle
```

Modification Applied

The modifications can be carried out via `cmsh` (section A.6.2) or `cmgui` (section A.6.3).

The MapReduce properties, which currently contain the two default `mapreduce.job*` values, can be modified by applying a reference to a custom class.

When the planned customizations are applied, the configuration file, `hdfs-site.xml` is modified. The modifications can be applied to nodes `node00[1-3]` according to the following configuration overlay specification:

```
<property>
  <!-- automatically customized by cmdaemon -->
  <name>mapreduce.job.map.output.collector.class</name>
  <value>org.apache.hadoop.mapred.MyFirstPlugin$MapOutputBuffer</value>
</property>

<property>
  <!-- automatically customized by cmdaemon -->
  <name>mapreduce.job.reduce.shuffle.consumer.plugin.class</name>
  <value>org.apache.hadoop.mapred.MyFirstPlugin$Shuffle</value>
</property>
```

The dependent services on these nodes are correctly restarted by Bright Cluster Manager.

A.6.2 Adding Customizations To A Configuration Overlay In `cmsh`

Inside the `configuration overlays` mode there is a submode for customizations. One or more customization files can be added in the `customizations` submode.

A file type, typically XML, is guessed by Bright Cluster Manager using its knowledge of certain paths on the filesystem and the file extension. The file type can be changed manually.

Inside a customization file there is another submode for entries. Customization entries can be added to the file. These entries are typically key-value pairs with an action, where the default action is to add the key-value pair to the configuration. How these entries are handled depends on the customization file type.

```
[cluster1->configurationoverlay[hdfs1-DataNode]]% customizations
[cluster1...-DataNode]->customizations*]% add /etc/hadoop/hdfs1/hdfs-site.xml
ok.
[...customizations*[/etc/hadoop/hdfs1/hdfs-site.xml*]]% list
```

| File (key) | Type | Label | Enabled |
|---------------------------------|----------|-------|---------|
| /etc/hadoop/hdfs1/hdfs-site.xml | XML File | | yes |

```
[.../hdfs-site.xml*]]% entries
[.../hdfs-site.xml*]->entries] add "mapreduce.job.map.output.collector.class"\
    "org.apache.hadoop.mapred.MyFirstPlugin$MapOutputBuffer"
ok.
[...entries[mapreduce.job.map.output.collector.class]]% list
Key                                     Value                                     Action   Enabled
-----
mapreduce.job.map.output.collector.class org.apache.hadoop.mapred.MyFirstPlugin$MapOutputBuffer Smart Add yes

[...entries[mapreduce.job.map.output.collector.class]]% commit
Commit configuration overlay 'hdfs1-DataNode' ... ok.
```

A label can be assigned to related customizations. This is useful for customizations that are spread out over multiple configuration overlays. The `cmsh` commands `customizationsenable` and `customizationsdisable` allow a label to be set.

A.6.3 Managing Customizations From `cmgui`

In `cmgui`, within the Configuration Groups tab for the big data instance, there is a further subtab Configuration Groups. Opening a configuration group opens up the a configuration group dialog (figure A.5) within which there is a Manage customizations button. The Manage customizations button opens up a customizations configuration dialog. This allows one or more customization files to be added or removed.

The screenshot shows a configuration dialog for 'Myhadoop-DataNode'. At the top, there are fields for 'Configuration Group' (Myhadoop-DataNode), 'Priority' (500), and 'Nodes in Configuration Group' (node001..node005) with an 'Add/remove nodes' button. Below this are two tabs: 'Configure HDFS DataNode' (selected) and 'Configure MRv1 TaskTracker'. Under the 'Configure HDFS DataNode' tab, there are several settings: 'Settings applied to' (node001..node005), 'Data directories' (/var/lib/hadoop/Myhadoop/data) with expand/collapse buttons, 'DataNode Java heap size' (512 MB), 'Number of failed volumes tolerated' (0), 'Reserved space for Non DFS use' (1073741824 byte), 'Bandwidth for balancer' (1048576 byte/sec), 'DataNode ports' (with a visibility icon), and 'More DataNode parameters' (with a visibility icon). At the bottom of the dialog are five buttons: 'Add role', 'Remove role', 'Manage customizations', 'Cancel', and 'Ok'.

Figure A.5: Hadoop configuration group showing the customization button

The screenshot displays the 'Customizations' tab for a Hadoop configuration group named 'Myhadoop-DataNode'. The group has a priority of 500 and applies to nodes 'node001..node005'. A table lists the customized files:

| Modified | File path | Manage... | Num Entries |
|----------|---------------------------|-----------|-------------|
| ✓ | /etc/hadoop/Myhadoop/h... | ✓ | 0 |

Below the table, the details for the selected file are shown:

File path: /etc/hadoop/Myhadoop/hdfs-site.xml
File type: XML
Managed by: hadoop
Enabled: ☒

The 'Entries' section contains a table with two entries:

| Key | Value | Enabled |
|--------------------------|-----------------------------|-------------------------------------|
| mapreduce.job.map.output | org.apache.hadoop.mapre | <input checked="" type="checkbox"/> |
| mapreduce.job.reduce.shu | ored.MyFirstPlugin\$Shuffle | <input checked="" type="checkbox"/> |

At the bottom, there are buttons for 'Add', 'Remove', 'Cancel', and 'Ok'.

Figure A.6: Managing customizations within a Hadoop configuration group