Oracle® Big Data Appliance

Software User's Guide

Release 2 (2.4)

E51159-02

January 2014

Provides an introduction to the Oracle Big Data Appliance software, tools, and administrative procedures.

ORACLE®

Oracle Big Data Appliance Software User's Guide, Release 2 (2.4)

E51159-02

Contents

4 Optimizing MapReduce Jobs Using Perfect Balance

5 Configuring Oracle Exadata Database Machine for Use with Oracle Big Data Appliance

Glossary

Index

Preface

The *Oracle Big Data Appliance Software User's Guide* describes how to manage and use the installed software.

Audience

This guide is intended for users of Oracle Big Data Appliance including:

- Application developers
- Data analysts
- Data scientists
- Database administrators
- System administrators

The *Oracle Big Data Appliance Software User's Guide* introduces Oracle Big Data Appliance installed software, features, concepts, and terminology. However, you must acquire the necessary information about administering Hadoop clusters and writing MapReduce programs from other sources.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information, see the following documents:

- *Oracle Big Data Appliance Perfect Balance Java API Reference*
- *Oracle Enterprise Manager System Monitoring Plug-in Installation Guide for Oracle Big Data Appliance*
- *Oracle Big Data Appliance Owner's Guide*
- *Oracle Big Data Connectors User's Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introducing Oracle Big Data Appliance

This chapter presents an overview of Oracle Big Data Appliance and describes the software installed on the system. This chapter contains the following sections:

- What Is Big Data?
- The Oracle Big Data Solution
- Software for Big Data
- Acquiring Data for Analysis
- Organizing Big Data
- Analyzing and Visualizing Big Data

What Is Big Data?

Using transactional data as the source of business intelligence has been commonplace for many years. As digital technology and the World Wide Web spread into every aspect of modern life, other sources of data can make important contributions to business decision making. Many businesses are looking to these new data sources. They are finding opportunities in analyzing vast amounts of data that until recently was discarded.

Big data is characterized by:

- High Variety
- High Complexity
- High Volume
- High Velocity

These characteristics pinpoint the challenges in deriving value from big data, and the differences between big data and traditional data sources that primarily provide highly structured, transactional data.

High Variety

Big data is derived from a variety of sources, such as:

- Equipment sensors: Medical, manufacturing, transportation, and other machine sensor transmissions
- Machines: Call detail records, web logs, smart meter readings, Global Positioning System (GPS) transmissions, and trading systems records

- Social media: Data streams from social media sites such as Facebook and blogging sites such as Twitter

Analysts can mine this data repeatedly as they devise new ways of extracting meaningful insights. What seems irrelevant today might prove to be highly pertinent to your business tomorrow.

Challenge: Delivering flexible systems to handle this high variety

High Complexity

As the variety of data types increases, the complexity of the system increases. The complexity of data types also increases in big data because of its low structure.

Challenge: Finding solutions that apply across a broad range of data types.

High Volume

Social media can generate terabytes of daily data. Equipment sensors and other machines can generate that much data in less than an hour.

Even traditional data sources for data warehouses, such as customer profiles from customer relationship management (CRM) systems, transactional enterprise resource planning (ERP) data, store transactions, and general ledger data, have increased tenfold in volume over the past decade.

Challenge: Providing scalability and ease in growing the system

High Velocity

Huge numbers of sensors, web logs, and other machine sources generate data continuously and at a much higher speed than traditional sources, such as individuals entering orders into a transactional database.

Challenge: Handling the data at high speed without stressing the structured systems

The Oracle Big Data Solution

Oracle Big Data Appliance is an engineered system comprising both hardware and software components. The hardware is optimized to run the enhanced big data software components.

Oracle Big Data Appliance delivers:

- A complete and optimized solution for big data
- Single-vendor support for both hardware and software
- An easy-to-deploy solution
- Tight integration with Oracle Database and Oracle Exadata Database Machine

Oracle provides a big data platform that captures, organizes, and supports deep analytics on extremely large, complex data streams flowing into your enterprise from many data sources. You can choose the best storage and processing location for your data depending on its structure, workload characteristics, and end-user requirements.

Oracle Database enables all data to be accessed and analyzed by a large user community using identical methods. By adding Oracle Big Data Appliance in front of Oracle Database, you can bring new sources of information to an existing data warehouse. Oracle Big Data Appliance is the platform for acquiring and organizing

big data so that the relevant portions with true business value can be analyzed in Oracle Database.

For maximum speed and efficiency, Oracle Big Data Appliance can be connected to Oracle Exadata Database Machine running Oracle Database. Oracle Exadata Database Machine provides outstanding performance in hosting data warehouses and transaction processing databases. Moreover, Oracle Exadata Database Machine can be connected to Oracle Exalytics In-Memory Machine for the best performance of business intelligence and planning applications. The InfiniBand connections between these engineered systems provide high parallelism, which enables high-speed data transfer for batch or query workloads.

Figure 1–1 shows the relationships among these engineered systems.

Figure 1–1   Oracle Engineered Systems for Big Data
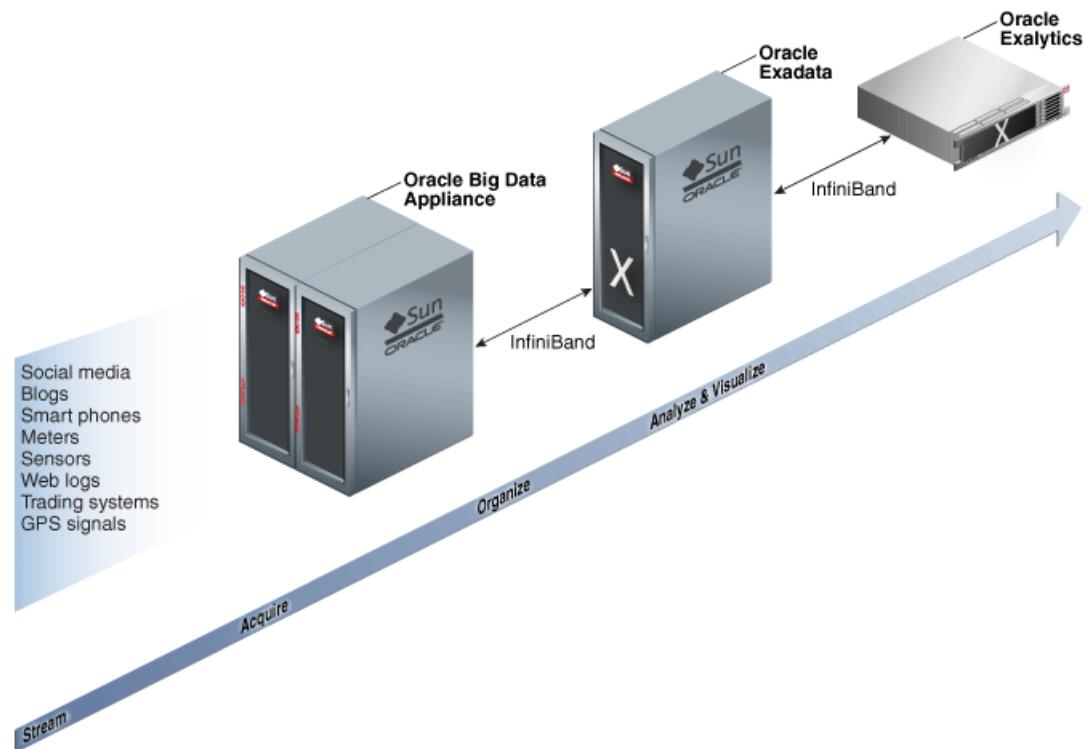
Software for Big Data

The **Oracle Linux** operating system and Cloudera's Distribution including Apache Hadoop (CDH) underlie all other software components installed on Oracle Big Data Appliance. **CDH** is an integrated stack of components that have been tested and packaged to work together.

CDH has a batch processing infrastructure that can store files and distribute work across a set of computers. Data is processed on the same computer where it is stored. In a single Oracle Big Data Appliance rack, CDH distributes the files and workload across 18 servers, which compose a **cluster**. Each server is a node in the cluster.

The software framework consists of these primary components:

- File system: The **Hadoop Distributed File System (HDFS)** is a highly scalable file system that stores large files across multiple servers. It achieves reliability by

replicating data across multiple servers without RAID technology. It runs on top of the Linux file system on Oracle Big Data Appliance.

- MapReduce engine: The MapReduce engine provides a platform for the massively parallel execution of algorithms written in Java.

- Administrative framework: Cloudera Manager is a comprehensive administrative tool for CDH.

CDH is written in Java, and Java is the language for applications development. However, several CDH utilities and other software available on Oracle Big Data Appliance provide graphical, web-based, and other language interfaces for ease of use.

Software Component Overview

The major software components perform three basic tasks:

- Acquire

- Organize

- Analyze and visualize

The best tool for each task depends on the density of the information and the degree of structure. Figure 1–2 shows the relationships among the tools and identifies the tasks that they perform.

Figure 1–2   Oracle Big Data Appliance Software Overview
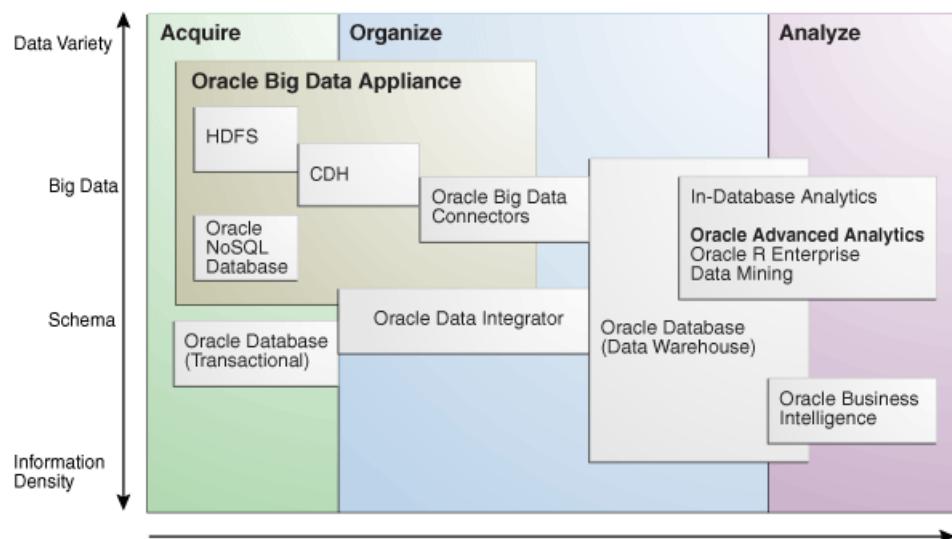
Acquiring Data for Analysis

Oracle Big Data Appliance provides these facilities for capturing and storing big data:

- Hadoop Distributed File System (HDFS)

- Oracle NoSQL Database

- **Hive**

Databases used for online transaction processing (OLTP) are the traditional data sources for data warehouses. The Oracle solution enables you to analyze traditional data stores with big data in the same Oracle data warehouse. Relational data continues

to be an important source of business intelligence, although it runs on separate hardware from Oracle Big Data Appliance.

Hadoop Distributed File System

Cloudera's Distribution including Apache Hadoop (CDH) on Oracle Big Data Appliance uses the Hadoop Distributed File System (HDFS). HDFS stores extremely large files containing record-oriented data. On Oracle Big Data Appliance, HDFS splits large data files into chunks of 256 megabytes (MB), and replicates each chunk across three different nodes in the cluster. The size of the chunks and the number of replications are configurable.

Chunking enables HDFS to store files that are larger than the physical storage of one server. It also allows the data to be processed in parallel across multiple computers with multiple processors, all working on data that is stored locally. Replication ensures the high availability of the data: if a server fails, the other servers automatically take over its work load.

HDFS is typically used to store all types of big data.

> **See Also:**
>
> - For conceptual information about Hadoop technologies, refer to this third-party publication:
>
>   *Hadoop: The Definitive Guide, Third Edition* by Tom White (O'Reilly Media Inc., 2012., ISBN: 978-1449311520).
>
> - For documentation about Cloudera's Distribution including Apache Hadoop, see the Cloudera library at
>
>   http://oracle.cloudera.com/

Hive

Hive is an open-source data warehouse that supports data summarization, ad hoc querying, and data analysis of data stored in HDFS. It uses a SQL-like language called **HiveQL**. An interpreter generates MapReduce code from the HiveQL queries. By storing data in Hive, you can avoid writing MapReduce programs in Java.

Hive is a component of CDH and is always installed on Oracle Big Data Appliance. Oracle Big Data Connectors can access Hive tables.

Oracle NoSQL Database

Oracle NoSQL Database is a distributed key-value database built on the proven storage technology of Berkeley DB Java Edition. Whereas HDFS stores unstructured data in very large files, Oracle NoSQL Database indexes the data and supports transactions. But unlike Oracle Database, which stores highly structured data, Oracle NoSQL Database has relaxed consistency rules, no schema structure, and only modest support for joins, particularly across storage nodes.

NoSQL databases, or "Not Only SQL" databases, have developed over the past decade specifically for storing big data. However, they vary widely in implementation. Oracle NoSQL Database has these characteristics:

- Uses a system-defined, consistent hash index for data distribution
- Supports high availability through replication

- Provides single-record, single-operation transactions with relaxed consistency guarantees
- Provides a Java API

Oracle NoSQL Database is designed to provide highly reliable, scalable, predictable, and available data storage. The key-value pairs are stored in shards or partitions (that is, subsets of data) based on a primary key. Data on each shard is replicated across multiple storage nodes to ensure high availability. Oracle NoSQL Database supports fast querying of the data, typically by key lookup.

An intelligent driver links the NoSQL database with client applications and provides access to the requested key-value on the storage node with the lowest latency.

Oracle NoSQL Database includes hashing and balancing algorithms to ensure proper data distribution and optimal load balancing, replication management components to handle storage node failure and recovery, and an easy-to-use administrative interface to monitor the state of the database.

Oracle NoSQL Database is typically used to store customer profiles and similar data for identifying and analyzing big data. For example, you might log in to a website and see advertisements based on your stored customer profile (a record in Oracle NoSQL Database) and your recent activity on the site (web logs currently streaming into HDFS).

Oracle NoSQL Database is an optional component of Oracle Big Data Appliance and runs on a separate cluster from CDH.

See Also:

- *Oracle NoSQL Database Getting Started Guide* at

  http://docs.oracle.com/cd/NOSQL/html/index.html

- *Oracle Big Data Appliance Licensing Information*

Organizing Big Data

Oracle Big Data Appliance provides several ways of organizing, transforming, and reducing big data for analysis:

- MapReduce
- Oracle R Support for Big Data
- Oracle Big Data Connectors

MapReduce

The MapReduce engine provides a platform for the massively parallel execution of algorithms written in Java. MapReduce uses a parallel programming model for processing data on a distributed system. It can process vast amounts of data quickly and can scale linearly. It is particularly effective as a mechanism for batch processing of unstructured and semistructured data. MapReduce abstracts lower-level operations into computations over a set of keys and values.

Although big data is often described as unstructured, incoming data always has some structure. However, it does not have a fixed, predefined structure when written to HDFS. Instead, MapReduce creates the desired structure as it reads the data for a particular job. The same data can have many different structures imposed by different MapReduce jobs.

A simplified description of a MapReduce job is the successive alternation of two phases: the Map phase and the Reduce phase. Each Map phase applies a transform function over each record in the input data to produce a set of records expressed as key-value pairs. The output from the Map phase is input to the Reduce phase. In the Reduce phase, the Map output records are sorted into key-value sets, so that all records in a set have the same key value. A reducer function is applied to all the records in a set, and a set of output records is produced as key-value pairs. The Map phase is logically run in parallel over each record, whereas the Reduce phase is run in parallel over all key values.

> **Note:** Oracle Big Data Appliance supports the Yet Another Resource Negotiator (YARN) implementation of MapReduce. However, the Mammoth utility installs and configures only classic MapReduce (MR v1). You cannot use both implementations in the same cluster; you can activate either the MapReduce or the YARN service. If you want to use YARN, Oracle recommends that you use Cloudera Manager to stop MapReduce and start YARN on a test cluster.

Oracle R Support for Big Data

R is an open-source language and environment for statistical analysis and graphing It provides linear and nonlinear modeling, standard statistical methods, time-series analysis, classification, clustering, and graphical data displays. Thousands of open-source packages are available in the Comprehensive R Archive Network (CRAN) for a spectrum of applications, such as bioinformatics, spatial statistics, and financial and marketing analysis. The popularity of R has increased as its functionality matured to rival that of costly proprietary statistical packages.

Analysts typically use R on a PC, which limits the amount of data and the processing power available for analysis. Oracle eliminates this restriction by extending the R platform to directly leverage Oracle Big Data Appliance. Oracle R Distribution is installed on all nodes of Oracle Big Data Appliance. Analysts continue to work on their PCs using the familiar R user interface while manipulating huge amounts of data stored in HDFS using massively parallel processing.

Oracle R Advanced Analytics for Hadoop provides R users with high-performance, native access to HDFS and the **MapReduce** programming framework, which enables R programs to run as MapReduce jobs on vast amounts of data. Oracle R Advanced Analytics for Hadoop is included in the Oracle Big Data Connectors. See "Oracle R Advanced Analytics for Hadoop" on page 1-8.

Oracle R Enterprise is a separate package that provides real-time access to Oracle Database. It enables you to store the results of your analysis of big data in an Oracle database, where it can be analyzed further.

These two Oracle R packages make Oracle Database and the Hadoop computational infrastructure available to statistical users without requiring them to learn the native programming languages of either one.

> **See Also:**
>
> ■ For information about R, go to
>
>   http://www.r-project.org/
>
> ■ For information about Oracle R Enterprise, go to
>
>   http://docs.oracle.com/cd/E36939_01/welcome.html

Oracle Big Data Connectors

Oracle Big Data Connectors facilitate data access between data stored in CDH and Oracle Database. The connectors are licensed separately from Oracle Big Data Appliance and include:

- Oracle SQL Connector for Hadoop Distributed File System
- Oracle Loader for Hadoop
- Oracle XQuery for Hadoop
- Oracle R Advanced Analytics for Hadoop
- Oracle Data Integrator Application Adapter for Hadoop

> **See Also:** *Oracle Big Data Connectors User's Guide*

Oracle SQL Connector for Hadoop Distributed File System

Oracle SQL Connector for Hadoop Distributed File System (Oracle SQL Connector for HDFS) provides read access to HDFS from an Oracle database using **external tables**.

An external table is an Oracle Database object that identifies the location of data outside of the database. Oracle Database accesses the data by using the metadata provided when the external table was created. By querying the external tables, users can access data stored in HDFS as if that data were stored in tables in the database. External tables are often used to stage data to be transformed during a database load.

You can use Oracle SQL Connector for HDFS to:

- Access data stored in HDFS files
- Access Hive tables.
- Access comma-separated value (CSV) files generated by Oracle Loader for Hadoop
- Load data extracted and transformed by Oracle Data Integrator

Oracle Loader for Hadoop

Oracle Loader for Hadoop is an efficient and high-performance loader for fast movement of data from CDH into a table in an Oracle database. Oracle Loader for Hadoop partitions the data and transforms it into a database-ready format on CDH. It optionally sorts records by primary key before loading the data or creating output files.

You can use Oracle Loader for Hadoop as either a Java program or a command-line utility. The load runs as a MapReduce job on the CDH cluster.

Oracle Loader for Hadoop also reads from and writes to Oracle Data Pump files.

Oracle XQuery for Hadoop

Oracle XQuery for Hadoop runs transformations expressed in the XQuery language by translating them into a series of MapReduce jobs, which are executed in parallel on the Hadoop cluster. The input data can be located in HDFS or Oracle NoSQL Database. Oracle XQuery for Hadoop can write the transformation results to HDFS, Oracle NoSQL Database, or Oracle Database.

Oracle R Advanced Analytics for Hadoop

Oracle R Advanced Analytics for Hadoop is a collection of R packages that provide:

- Interfaces to work with Hive tables, Apache Hadoop compute infrastructure, local R environment and database tables

- Predictive analytic techniques written in R or Java as Hadoop MapReduce jobs that can be applied to data in HDFS files

Using simple R functions, you can copy data between R memory, the local file system, HDFS, and Hive. You can schedule R programs to execute as Hadoop MapReduce jobs and return the results to any of those locations. You can transfer existing R scripts and packages from your PC to Oracle Big Data Appliance, and use Oracle R Advanced Analytics for Hadoop functions to convert them to the MapReduce paradigm.

Oracle Data Integrator Application Adapter for Hadoop

Oracle Data Integrator (ODI) extracts, transforms, and loads data into Oracle Database from a wide range of sources.

In ODI, a knowledge module (KM) is a code template dedicated to a specific task in the data integration process. You use Oracle Data Integrator Studio to load, select, and configure the KMs for your particular application. More than 150 KMs are available to help you acquire data from a wide range of third-party databases and other data repositories. You only need to load a few KMs for any particular job.

Oracle Data Integrator Application Adapter for Hadoop contains the KMs specifically for use with big data.

Analyzing and Visualizing Big Data

After big data is transformed and loaded in Oracle Database, you can use the full spectrum of Oracle business intelligence solutions and decision support products to further analyze and visualize all your data.

See Also:

- Oracle Business Intelligence website at

  http://www.oracle.com/us/solutions/ent-performance-bi/business-intelligence/index.html

- Data Warehousing and Business Intelligence in the Oracle Database Documentation Library at

  http://www.oracle.com/pls/db112/portal.portal_db?selected=6&frame=

2

Administering Oracle Big Data Appliance

This chapter provides information about the software and services installed on Oracle Big Data Appliance. It contains these sections:

- Monitoring a Cluster Using Oracle Enterprise Manager
- Managing CDH Operations Using Cloudera Manager
- Using Hadoop Monitoring Utilities
- Using Cloudera Hue to Interact With Hadoop
- About the Oracle Big Data Appliance Software
- About the CDH Software Services
- Effects of Hardware on Software Availability
- Stopping and Starting Oracle Big Data Appliance
- Security on Oracle Big Data Appliance
- Auditing Oracle Big Data Appliance
- Collecting Diagnostic Information for Oracle Customer Support

Monitoring a Cluster Using Oracle Enterprise Manager

An Oracle Enterprise Manager plug-in enables you to use the same system monitoring tool for Oracle Big Data Appliance as you use for Oracle Exadata Database Machine or any other Oracle Database installation. With the plug-in, you can view the status of the installed software components in tabular or graphic presentations, and start and stop these software services. You can also monitor the health of the network and the rack components.

Using the Enterprise Manager Web Interface

After opening Oracle Enterprise Manager web interface, logging in, and selecting a target cluster, you can drill down into these primary areas:

- **InfiniBand network**: Network topology and status for InfiniBand switches and ports. See Figure 2–1.
- **Hadoop cluster**: Software services for HDFS, MapReduce, and ZooKeeper.
- **Oracle Big Data Appliance rack**: Hardware status including server hosts, Oracle Integrated Lights Out Manager (Oracle ILOM) servers, power distribution units (PDUs), and the Ethernet switch.

Figure 2–1 shows a small section of the cluster home page.

Figure 2–1   Cluster Home in Oracle Enterprise Manager
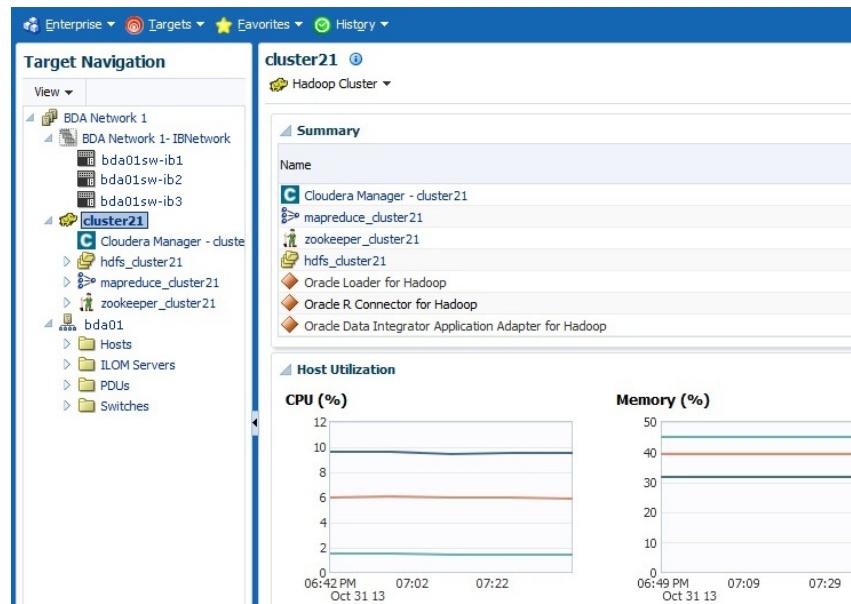
To monitor Oracle Big Data Appliance using Oracle Enterprise Manager:

1. Download and install the plug-in. See *Oracle Enterprise Manager System Monitoring Plug-in Installation Guide for Oracle Big Data Appliance*.

2. Log in to Oracle Enterprise Manager as a privileged user.

3. From the Targets menu, choose **Big Data Appliance** to view the Big Data page. You can see the overall status of the targets already discovered by Oracle Enterprise Manager.

4. Select a target cluster to view its detail pages.

5. Expand the target navigation tree to display the components. Information is available at all levels.

6. Select a component in the tree to display its home page.

7. To change the display, choose an item from the drop-down menu at the top left of the main display area.

> **See Also:**   *Oracle Enterprise Manager System Monitoring Plug-in Installation Guide for Oracle Big Data Appliance* for installation instructions and use cases.

Using the Enterprise Manager Command-Line Interface

The Enterprise Manager command-line interface (`emcli`) is installed on Oracle Big Data Appliance along with all the other software. It provides the same functionality as the web interface. You must provide credentials to connect to Oracle Management Server.

To get help, enter `emcli help`.

> **See Also:**   *Oracle Enterprise Manager Command Line Interface Guide*

Managing CDH Operations Using Cloudera Manager

Cloudera Manager is installed on Oracle Big Data Appliance to help you with Cloudera's Distribution including Apache Hadoop (CDH) operations. Cloudera Manager provides a single administrative interface to all Oracle Big Data Appliance servers configured as part of the Hadoop cluster.

Cloudera Manager simplifies the performance of these administrative tasks:

- Monitor jobs and services

- Start and stop services

- Manage security and Kerberos credentials

- Monitor user activity

- Monitor the health of the system

- Monitor performance metrics

- Track hardware use (disk, CPU, and RAM)

Cloudera Manager runs on the JobTracker node (node03) and is available on port 7180.

To use Cloudera Manager:

1. Open a browser and enter a URL like the following:

   ```
   http://bda1node03.example.com:7180
   ```

   In this example, `bda1` is the name of the appliance, `node03` is the name of the server, `example.com` is the domain, and `7180` is the default port number for Cloudera Manager.

2. Log in with a user name and password for Cloudera Manager. Only a user with administrative privileges can change the settings. Other Cloudera Manager users can view the status of Oracle Big Data Appliance.

   > **See Also:** *Cloudera Manager Monitoring and Diagnostics Guide* at
   >
   > http://www.cloudera.com/content/cloudera-content/cloudera-do
   > cs/CM4Ent/latest/Cloudera-Manager-Diagnostics-Guide/Cloudera
   > -Manager-Diagnostics-Guide.html
   >
   > or click **Help** on the Cloudera Manager Help menu

Monitoring the Status of Oracle Big Data Appliance

In Cloudera Manager, you can choose any of the following pages from the menu bar across the top of the display:

- **Home**: Provides a graphic overview of activities on the cluster, and links to all services controlled by Cloudera Manager. See Figure 2–2.

- **Services**: You can choose **All Services** from the Services menu for a top-level view of the status and health of services running on Oracle Big Data Appliance. Or you can choose a particular service for detailed information.

- **Hosts**: Monitors the health, disk usage, load, physical memory, swap space, and other statistics for all servers in the cluster.

- **Activities**: Monitors all MapReduce jobs running in the selected time period.

- **Diagnose**: You can view events or logs from the Diagnose menu. Cloudera Manager collects historical information about the systems and services. You can

search for a particular phrase for a selected server, service, and time period. You can also select the minimum severity level of the logged messages included in the search: TRACE, DEBUG, INFO, WARN, ERROR, or FATAL.

- **Audits**: Displays the audit history log for a selected time range. You can filter the results by user name, service, or other criteria, and download the log as a CSV file.

- **Charts**: Displays metrics from the Cloudera Manager time-series data store. You can choose from a variety of chart types, such as line and bar.

- **Reports**: Generates reports on demand for disk and MapReduce use.

- **Administration**: Select from a variety of options on the Administration menu, including Settings, Alerts, Users, and Kerberos.

Figure 2–2 shows the Cloudera Manager home page.

Figure 2–2   Cloudera Manager Home Page
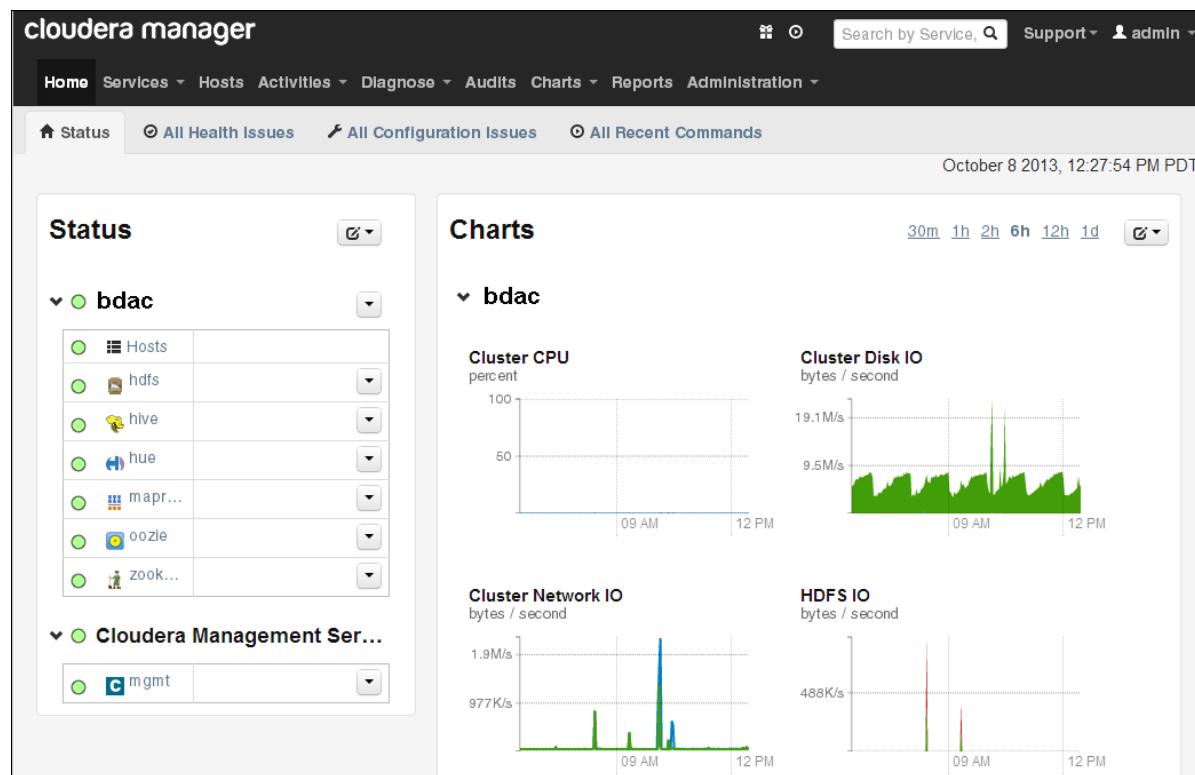
Performing Administrative Tasks

As a Cloudera Manager administrator, you can change various properties for monitoring the health and use of Oracle Big Data Appliance, add users, and set up Kerberos security.

To access Cloudera Manager Administration:

1. Log in to Cloudera Manager with administrative privileges.

2. Select the Administration tab.

Managing Services With Cloudera Manager

Cloudera Manager provides the interface for managing these services:

- HDFS

- Hive

- Hue

- MapReduce

- Oozie

- ZooKeeper

You can use Cloudera Manager to change the configuration of these services, stop, and restart them. Additional services are also available, which require configuration before you can use them. See "Unconfigured Software" on page 2-13.

> **Note:** Manual edits to Linux service scripts or Hadoop configuration files do not affect these services. You must manage and configure them using Cloudera Manager.

Using Hadoop Monitoring Utilities

Users can monitor MapReduce jobs without providing a Cloudera Manager user name and password.

Monitoring the JobTracker

Hadoop Map/Reduce Administration monitors the **JobTracker**, which runs on port 50030 of the JobTracker node (node03) on Oracle Big Data Appliance.

To monitor the JobTracker:

- Open a browser and enter a URL like the following:

  `http://bda1node03.example.com:50030`

  In this example, `bda1` is the name of the appliance, `node03` is the name of the server, and `50030` is the default port number for Hadoop Map/Reduce Administration.

Figure 2–3 shows part of a Hadoop Map/Reduce Administration display.

Figure 2–3   Hadoop Map/Reduce Administration

Monitoring the TaskTracker

The Task Tracker Status interface monitors the **TaskTracker** on a single node. It is available on port 50060 of all noncritical nodes (node04 to node18) in Oracle Big Data Appliance. On six-node clusters, the TaskTracker also runs on node01 and node02.

To monitor a TaskTracker:

■ Open a browser and enter the URL for a particular node like the following:

```
http://bda1node13.example.com:50060
```

In this example, `bda1` is the name of the rack, `node13` is the name of the server, and `50060` is the default port number for the Task Tracker Status interface.

Figure 2–4 shows the Task Tracker Status interface.

Figure 2–4   Task Tracker Status Interface

Using Cloudera Hue to Interact With Hadoop

Hue runs in a browser and provides an easy-to-use interface to several applications to support interaction with Hadoop and HDFS. You can use Hue to perform any of the following tasks:

- Query Hive data stores

- Create, load, and delete Hive tables

- Work with HDFS files and directories

- Create, submit, and monitor MapReduce jobs

- Monitor MapReduce jobs

- Create, edit, and submit workflows using the Oozie dashboard

- Manage users and groups

Hue runs on port 8888 of the JobTracker node (node03).

To use Hue:

1. Open Hue in a browser using an address like the one in this example:

   `http://bda1node03.example.com:8888`

   In this example, `bda1` is the cluster name, `node03` is the server name, and `example.com` is the domain.

2. Log in with your Hue credentials.

   Oracle Big Data Appliance is not configured initially with any Hue user accounts. The first user who connects to Hue can log in with any user name and password, and automatically becomes an administrator. This user can create other user and administrator accounts.

3. Use the icons across the top to open a utility.

Figure 2–5 shows the Beeswax Query Editor for entering Hive queries.

Figure 2–5   Beeswax Query Editor
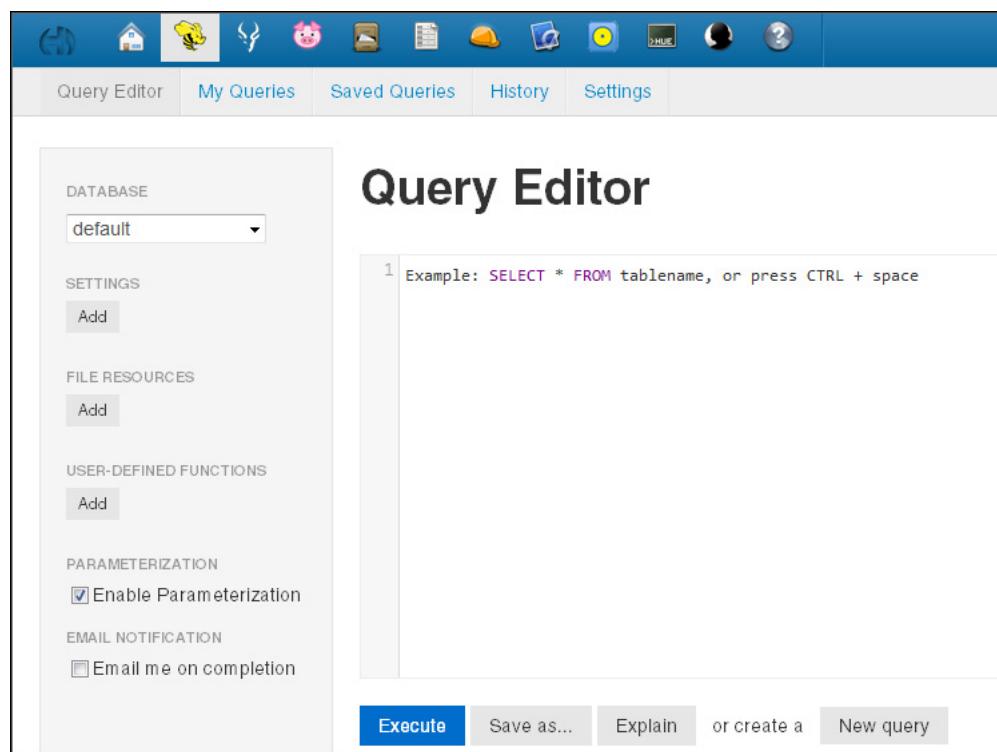
> **See Also:**   *Hue Installation Guide* for information about using Hue, which is already installed and configured on Oracle Big Data Appliance, at
>
> http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/latest/Hue-2-User-Guide/Hue-2-User-Guide.html

About the Oracle Big Data Appliance Software

The following sections identify the software installed on Oracle Big Data Appliance and where it runs in the rack. Some components operate with Oracle Database 11.2.0.2 and later releases.

Software Components

These software components are installed on all servers in the cluster. Oracle Linux, required drivers, firmware, and hardware verification utilities are factory installed. All other software is installed on site. The optional software components may not be configured in your installation.

> **Note:**   You do not need to install additional software on Oracle Big Data Appliance. Doing so may result in a loss of warranty and support. See the *Oracle Big Data Appliance Owner's Guide*.

Base image software:

- Oracle Linux 6.4 (upgrades stay at 5.8) with Oracle Unbreakable Enterprise Kernel version 2 (UEK2)

- Java **HotSpot** Virtual Machine 7 version 25 (JDK 7u25)
- **Oracle R Distribution** 3.0.1-2
- **MySQL Database** 5.5.17 Advanced Edition
- Puppet, firmware, utilities

Mammoth installation:

- Cloudera's Distribution including Apache Hadoop Release 4 Update 4 (4.5)
- Cloudera Manager 4.8
- **Oracle Database Instant Client** 12.1
- Oracle NoSQL Database Community Edition or Enterprise Edition 12*c* Release 1 (optional)
- Oracle Big Data Connectors 2.4 (optional):
  - Oracle SQL Connector for Hadoop Distributed File System (HDFS)
  - Oracle Loader for Hadoop
  - Oracle Data Integrator Agent 11.1.1.7.0
  - Oracle XQuery for Hadoop
  - Oracle R Advanced Analytics for Hadoop

> **See Also:** *Oracle Big Data Appliance Owner's Guide* for information about the Mammoth utility

Figure 2–6 shows the relationships among the major components.

Figure 2–6   Major Software Components of Oracle Big Data Appliance
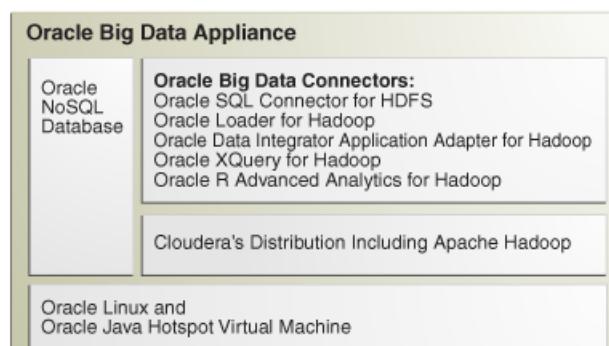
Logical Disk Layout

Each server has 12 disks. The critical operating system is stored on disks 1 and 2.

Table 2–1 describes how the disks are partitioned.

Table 2–1   Logical Disk Layout

Disk	Description
1 to 2	150 gigabytes (GB) physical and logical partition, mirrored to create two copies, with the Linux operating system, all installed software, NameNode data, and MySQL Database data. The NameNode and MySQL Database data are replicated on two servers for a total of four copies.
	2.8 terabytes (TB) HDFS data partition
3 to 12	Single HDFS or Oracle NoSQL Database data partition

About the CDH Software Services

This section contains the following topics:

- Monitoring the CDH Services
- Where Do the CDH Services Run?
- Automatic Failover of the NameNode
- Unconfigured Software

Monitoring the CDH Services

You can use Cloudera Manager to monitor the CDH services on Oracle Big Data Appliance.

To monitor the services:

1. In Cloudera Manager, click the **Services** tab, and then the name of a service to see its Status page.

2. Click a subtab for the information you want to view, such as Status, Instances, Commands, Configuration, or Audits.

Where Do the CDH Services Run?

All services are installed on all nodes in a CDH cluster, but individual services run only on designated nodes. There are slight variations in the location of the services depending on the configuration of the cluster.

Service Locations on a Single Rack

Table 2–2 identifies the services in CDH clusters configured within a single rack, including starter racks and clusters with six nodes. Node01 is the first server in the cluster (server 1, 7, or 10), and node*xx* is the last server in the cluster (server 6, 9, 12, or 18).

Table 2–2   Service Locations for One or More CDH Clusters in a Single Rack

Node01	Node02	Node03	Node04	Node05 to *nn*
Balancer				
			Hive, Hue, Oozie	
Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent
		Cloudera Manager Server		

Table 2–2 (Cont.) Service Locations for One or More CDH Clusters in a Single Rack

Node01	Node02	Node03	Node04	Node05 to *nn*
DataNode	DataNode	DataNode	DataNode	DataNode
Failover Controller	Failover Controller	Failover Controller	Failover Controller	
JournalNode	JournalNode	JournalNode		
	MySQL Backup	MySQL Primary		
NameNode 1	NameNode 2			
			Oracle Data Integrator Agent	
Puppet	Puppet	Puppet	Puppet	Puppet
Puppet Master				
		JobTracker 1	JobTracker 2	
Task Tracker[1]	Task Tracker[1]	TaskTracker	TaskTracker	TaskTracker
ZooKeeper Server	ZooKeeper Server	ZooKeeper Server		

[1] Starter racks and six-node clusters only

Service Locations in Multirack Clusters

When multiple racks are configured as a single CDH cluster, some critical services are moved to the first server of the second rack.

Critical services on the first server of the second rack:

- Balancer
- Failover Controller
- Journal Node
- NameNode 1
- ZooKeeper Server

The DataNode, Cloudera Manager agent, and Puppet services also run on this server.

Table 2–3 identify the location of services on the first rack of a multirack cluster.

Table 2–3   Service Locations in the First Rack of a Multirack CDH Cluster

Node01	Node02	Node03	Node04	Node05 to *nn*[1]
Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent
		Cloudera Manager Server		
DataNode	DataNode	DataNode	DataNode	DataNode
	Failover Controller	Failover Controller	Failover Controller	
			Hive, Hue, Oozie	
	JournalNode	JournalNode		
	MySQL Backup	MySQL Primary		
	NameNode 2			

Table 2–3 (Cont.) Service Locations in the First Rack of a Multirack CDH Cluster

Node01	Node02	Node03	Node04	Node05 to *nn*[1]
			Oracle Data Integrator agent	
Puppet	Puppet	Puppet	Puppet	Puppet
Puppet Master				
		JobTracker 1	JobTracker 2	
TaskTracker		TaskTracker	TaskTracker	TaskTracker
	ZooKeeper Server	ZooKeeper Server		

[1] *nn* includes the servers in additional racks.

Automatic Failover of the NameNode

The NameNode is the most critical process because it keeps track of the location of all data. Without a healthy NameNode, the entire cluster fails. Apache Hadoop v0.20.2 and earlier are vulnerable to failure because they have a single name node.

Cloudera's Distribution including Apache Hadoop Version 4 (CDH4) reduces this vulnerability by maintaining redundant NameNodes. The data is replicated during normal operation as follows:

- CDH maintains redundant NameNodes on the first two nodes. One of the NameNodes is in active mode, and the other NameNode is in hot standby mode. If the active NameNode fails, then the role of active NameNode automatically fails over to the standby NameNode.

- The NameNode data is written to a mirrored partition so that the loss of a single disk can be tolerated. This mirroring is done at the factory as part of the operating system installation.

- The active NameNode records all changes to the file system metadata in at least two JournalNode processes, which the standby NameNode reads. There are three JournalNodes, which run on the first three nodes of each cluster.

- The changes recorded in the journals are periodically consolidated into a single fsimage file in a process called checkpointing.

> **Note:**   Oracle Big Data Appliance 2.0 and later releases do not support the use of an external NFS filer for backups and do not use NameNode federation.

Figure 2–7 shows the relationships among the processes that support automatic failover of the NameNode.

Figure 2–7   Automatic Failover of the NameNode on Oracle Big Data Appliance
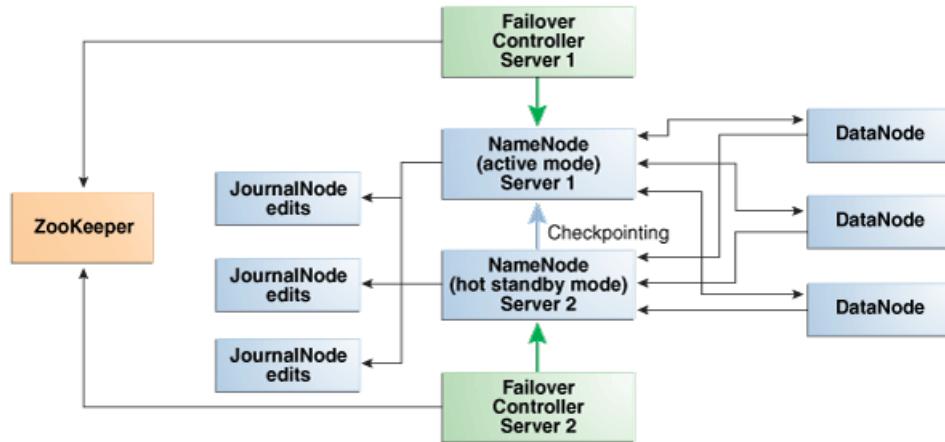
Automatic Failover of the JobTracker

The JobTracker distributes MapReduce tasks across the cluster. Like the NameNode, the JobTracker is a critical point of failure for the node. If the JobTracker fails, then all jobs stop running.

Oracle Big Data Appliance 2.2 and later releases support JobTracker High Available in CDH4 to reduce this vulnerability.

CDH maintains redundant JobTracker services on node03 and -04. One of the services is in active mode, and the other service is in hot standby mode. Failover controllers on the two nodes monitor the health of the services. If the active service fails, then the role of active JobTracker automatically fails over to the standby service.

Figure 2–7 shows the relationships among the processes that support automatic failover of the NameNode.

Figure 2–8   Automatic Failover of the JobTracker on Oracle Big Data Appliance
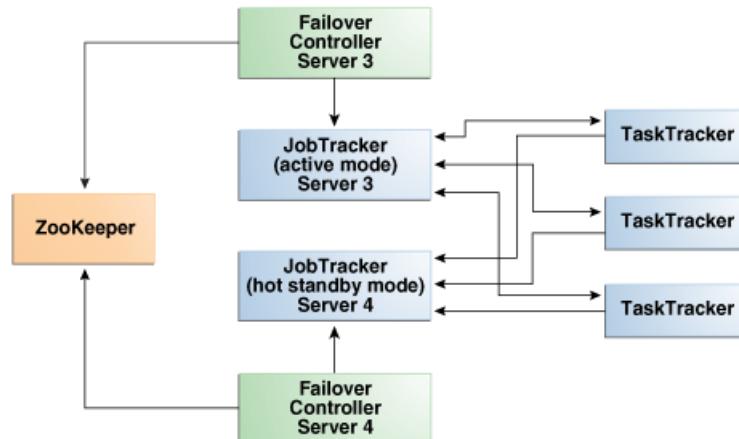
Unconfigured Software

The following tools are included in your Oracle Big Data Appliance license, and they are installed with the other CDH components by the Mammoth utility. Do not download them from the Cloudera website. However, you must configure them before they are available for use.

- Cloudera BDR (built into Cloudera Manager)

- Cloudera Impala

- Cloudera Navigator (built into Cloudera Manager)

- Cloudera Search

- Flume NG

- HBase

- Mahout

- Sqoop

- Whirr

You can find the RPM files on the first server of each cluster in /opt/oracle/BDAMammoth/bdarepo/RPMS/noarch.

> **See Also:** *CDH4 Installation and Configuration Guide* for configuration procedures at
>
> http://www.cloudera.com/content/cloudera-content/cloudera-do
> cs/CDH4/latest/CDH4-Installation-Guide/CDH4-Installation-Gui
> de.html

Map and Reduce Resource Configuration

Each node in a cluster has a maximum number of map and reduce tasks that are allowed to run simultaneously. Table 2–4 shows the default configuration of resources for the MapReduce service on Oracle Big Data Appliance X4-2 and X3-2.

Table 2–4   Maximum Map and Reduce Tasks on Oracle Big Data Appliance X3-2

Node	6-Node Cluster	Larger Clusters[1]
Node01 and Node02	14 map 8 reduce	None
Node03 and Node04	10 map 6 reduce	10 map 6 reduce
Node05 to Node*nn*	20 map 13 reduce	20 map 13 reduce

[1] 9 or more nodes

Table 2–5 shows the default configuration of resources for the MapReduce service on Oracle Big Data Appliance Sun Fire X4270 M2-based racks.

Table 2–5   Maximum Map and Reduce Tasks on Sun Fire X4270 M2-Based Racks

Node	6-Node Cluster	Larger Clusters[1]
Node01 and Node02	10 map 6 reduce	None
Node03 and Node04	7 map 4 reduce	7 map 4 reduce
Node05 to Node*nn*	15 map 10 reduce	15 map 10 reduce

[1] 9 or more nodes

Effects of Hardware on Software Availability

The effects of a server failure vary depending on the server's function within the CDH cluster. Oracle Big Data Appliance servers are more robust than commodity hardware, so you should experience fewer hardware failures. This section highlights the most important services that run on the various servers of the primary rack. For a full list, see "Where Do the CDH Services Run?" on page 2-10.

> **Note:** In a multirack cluster, some critical services run on the first server of the second rack. See "Service Locations in Multirack Clusters" on page 2-11.

Critical and Noncritical Nodes

Critical nodes are required for the cluster to operate normally and provide all services to users. In contrast, the cluster continues to operate with no loss of service when a noncritical node fails.

On single-rack clusters, the critical services are installed initially on the first four nodes of the cluster. The remaining nodes (node05 up to node18) only run noncritical services. If a hardware failure occurs on one of the critical nodes, then the services can be moved to another, noncritical server. For example, if node02 fails, then you might move its critical services node05. Table 2–2 identifies the initial location of services for clusters that are configured on a single rack.

In a multirack cluster, some critical services run on the first server of the second rack. See "Where Do the CDH Services Run?" on page 2-10.

Moving a critical node requires that all clients be reconfigured with the address of the new node. The other alternative is to wait for the failed server to be repaired. You must weigh the loss of services against the inconvenience of reconfiguring the clients.

Table 2–6   Critical Service Locations

Node Name	Initial Node Position	Critical Functions
First NameNode	Node01	ZooKeeper[1], first NameNode[1], failover controller[1], balancer[1], puppet master
Second NameNode	Node02	ZooKeeper, second NameNode, failover controller, MySQL backup server
First JobTracker Node	Node03	ZooKeeper, first JobTracker, failover controller, Cloudera Manager server, MySQL primary server
Second JobTracker Node	Node04	Second JobTracker, failover controller, Oracle Data Integrator agent, Hue, Hive, Oozie

[1] In multirack clusters, this service is initially installed on the first server of the second rack.

First Namenode

One instance of the NameNode initially runs on node01. If this node fails or goes offline (such as a restart), then the second NameNode (node02) automatically takes over to maintain the normal activities of the cluster.

Alternatively, if the second NameNode is already active, it continues without a backup. With only one NameNode, the cluster is vulnerable to failure. The cluster has lost the redundancy needed for automatic failover.

The puppet master also runs on this node. The Mammoth utilities use Puppet, and so you cannot install or reinstall the software if, for example, you must replace a disk drive elsewhere in the rack.

> **Note:** In multirack clusters, the NameNode service is installed on the first server of the *second* rack.

Second NameNode

One instance of the NameNode initially runs on node02. If this node fails, then the function of the NameNode either fails over to the first NameNode (node01) or continues there without a backup. However, the cluster has lost the redundancy needed for automatic failover if the first NameNode also fails.

The MySQL backup database also runs on this node. MySQL Database continues to run, although there is no backup of the master database.

First JobTracker

One instance of the JobTracker initially runs on node03. If this node fails or goes offline (such as a restart, then the second JobTracker (node04) automatically takes over to distribute MapReduce tasks to specific nodes across the cluster.

Alternatively, if the second JobTracker is already active, it continues without a backup. With only one JobTracker, the cluster is vulnerable to failure. The cluster has lost the redundancy needed for automatic failover.

These services are also disrupted:

- **Cloudera Manager**: This tool provides central management for the entire CDH cluster. Without this tool, you can still monitor activities using the utilities described in "Using Hadoop Monitoring Utilities" on page 2-5.

- **MySQL Master Database**: Cloudera Manager, Oracle Data Integrator, Hive, and Oozie use MySQL Database. The data is replicated automatically, but you cannot access it when the master database server is down.

Second JobTracker

One instance of the JobTracker initially runs on node04. If this node fails, then the function of the JobTracker either fails over to the first JobTracker (node03) or continues there without a backup. However, the cluster has lost the redundancy needed for automatic failover if the first JobTracker also fails.

These services are also disrupted:

- **Oracle Data Integrator**: This service supports Oracle Data Integrator Application Adapter for Hadoop. You cannot use this connector when the JobTracker node is down.

- **Hive**: Hive provides a SQL-like interface to data that is stored in HDFS. Most of the Oracle Big Data Connectors can access Hive tables, which are not available if this node fails.

- **Hue**: This administrative tool is not available when the JobTracker node is down.

■ **Oozie**: This workflow and coordination service runs on the JobTracker node, and is unavailable when the node is down.

Noncritical Nodes

The noncritical nodes (node05 to node18) are optional in that Oracle Big Data Appliance continues to operate with no loss of service if a failure occurs. The NameNode automatically replicates the lost data to maintain three copies at all times. MapReduce jobs execute on copies of the data stored elsewhere in the cluster. The only loss is in computational power, because there are fewer servers on which to distribute the work.

Stopping and Starting Oracle Big Data Appliance

This section describes how to shut down Oracle Big Data Appliance gracefully and restart it.

■ Prerequisites

■ Stopping Oracle Big Data Appliance

■ Starting Oracle Big Data Appliance

Prerequisites

You must have `root` access. Passwordless SSH must be set up on the cluster, so that you can use the `dcli` utility.

To ensure that passwordless-ssh is set up:

1. Log in to the first node of the cluster as `root`.

2. Use a `dcli` command to verify it is working. This command should return the IP address and host name of every node in the cluster:

```
# dcli -C hostname
192.0.2.1: bda1node01.example.com
192.0.2.2: bda1node02.example.com
     .
     .
     .
```

3. If you do not get these results, then set up `dcli` on the cluster:

```
# setup-root-ssh -C
```

> **See Also:** *Oracle Big Data Appliance Owner's Guide* for details about these commands.

Stopping Oracle Big Data Appliance

Follow these procedures to shut down all Oracle Big Data Appliance software and hardware components.

> **Note:** The following services stop automatically when the system shuts down. You do not need to take any action:
>
> ■ Oracle Enterprise Manager agent
>
> ■ Auto Service Request agents

Task 1 Stopping All Managed Services

Use Cloudera Manager to stop the services it manages, including `flume`, `hbase`, `hdfs`, `hive`, `hue`, `mapreduce`, `oozie`, and `zookeeper`.

1. Log in to Cloudera Manager as the `admin` user.

    See "Managing CDH Operations Using Cloudera Manager" on page 2-3.

2. In the Status pane of the opening page, expand the menu for the cluster and click **Stop**, and then click **Stop** again when prompted to confirm. See Figure 2–9.

    To navigate to this page, click the **Home** tab, and then the **Status** subtab.

3. On the Command Details page, click **Close** when all processes are stopped.

4. In the same pane under Cloudera Management Services, expand the menu for the `mgmt` service and click **Stop**.

5. Log out of Cloudera Manager.

Figure 2–9 Stopping HDFS Services
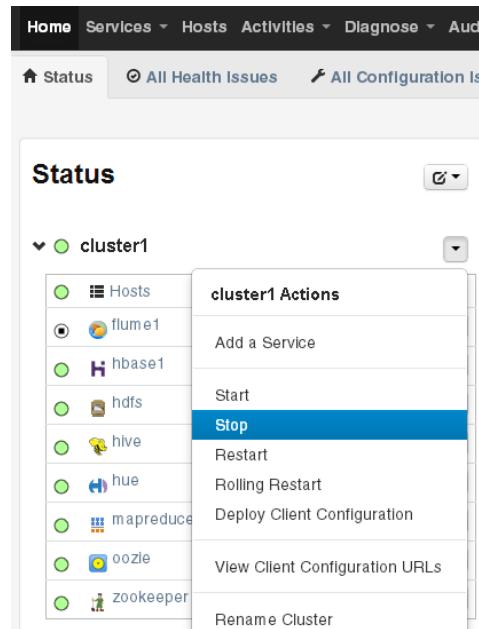
Task 2 Stopping Cloudera Manager Server

Follow this procedure to stop Cloudera Manager Server.

1. Log in as root to the node where Cloudera Manager runs (initially node03).

    > **Note:** The remaining tasks presume that you are logged in to a server as `root`. You can enter the commands from any server by using the `dcli` command. This example runs the `pwd` command on node03 from any node in the cluster:
    >
    > ```
    > # dcli -c node03 pwd
    > ```

2. Stop the Cloudera Manager server:

    ```
    # service cloudera-scm-server stop
    Stopping cloudera-scm-server:                              [  OK  ]
    ```

3. Verify that the server is stopped:

```
# service cloudera-scm-server status
cloudera-scm-server is stopped
```

After stopping Cloudera Manager, you cannot access it using the web console.

Task 3 Stopping Oracle Data Integrator Agent

If Oracle Data Integrator Application Adapter for Hadoop is installed on the cluster, then stop the agent. You must know the agent name and port number used previously to start the agent. The default agent name is OracleDIAgent, and the default port number is 20910.

1. Check the status of the agent:

```
/opt/oracle/odiagent/agent_standalone/oracledi/agent/bin/startcmd.sh
OdiPingAgent [-AGENT_NAME=name]
```

2. Stop the agent, if it is running:

```
/opt/oracle/odiagent/agent_standalone/oracledi/agent/bin/agentstop.sh
[-NAME=agent_name] [-PORT=port_number]
```

Task 4 Dismounting NFS Directories

All nodes share an NFS directory on node03, and additional directories may also exist. If a server with the NFS directory (/opt/exportdir) is unavailable, then the other servers hang when attempting to shut down. Thus, you must dismount the NFS directories first.

1. Locate any mounted NFS directories:

```
# dcli -C mount | grep shareddir
192.0.2.1: bda1node03.example.com:/opt/exportdir on /opt/shareddir type nfs
(rw,tcp,soft,intr,timeo=10,retrans=10,addr=192.0.2.3)
192.0.2.2: bda1node03.example.com:/opt/exportdir on /opt/shareddir type nfs
(rw,tcp,soft,intr,timeo=10,retrans=10,addr=192.0.2.3)
192.0.2.3: /opt/exportdir on /opt/shareddir type none (rw,bind)
    .
    .
    .
```

The sample output shows a shared directory on node03 (192.0.2.3).

2. Dismount the shared directory:

```
# dcli -C umount /opt/shareddir
```

3. Dismount any custom NFS directories.

Task 5 Stopping the Servers

The Linux shutdown -h command powers down individual servers. You can use the dcli -g command to stop multiple servers.

1. Create a file that lists the names or IP addresses of the other servers in the cluster, that is, not including the one you are logged in to.

2. Stop the other servers:

```
# dcli -g filename shutdown -h now
```

For *filename*, enter the name of the file that you created in step 1.

3. Stop the server you are logged in to:

```
# shutdown -h now
```

Task 6 Stopping the InfiniBand and Cisco Switches

To stop the network switches, turn off a PDU or a breaker in the data center. The switches only turn off when power is removed.

The network switches do not have power buttons. They shut down only when power is removed

To stop the switches, turn off all breakers in the two PDUs.

Starting Oracle Big Data Appliance

Follow these procedures to power up the hardware and start all services on Oracle Big Data Appliance.

Task 1 Powering Up Oracle Big Data Appliance

1. Switch on all 12 breakers on both PDUs.

2. Allow 4 to 5 minutes for Oracle ILOM to start.

Task 2 Powering Up the Servers

You can power up Oracle Big Data Appliance servers locally by pressing the power button on the front of the servers, or remotely by using Oracle ILOM.

Oracle ILOM has several interfaces, including a command-line interface (CLI) and a web console. You can use whichever interface you prefer.

For example, you can log into the web interface as `root` and start the server from the Remote Power Control page. The URL for Oracle ILOM is the same as for the host, except that it typically has a `-c` or `-ilom` extension. This URL connects to Oracle ILOM for bda1node4:

```
http://bda1node04-ilom.example.com
```

Task 3 Starting the HDFS Software Services

Use Cloudera Manager to start all the HDFS services that it controls.

1. Log in as `root` to the node where Cloudera Manager runs (initially node03).

> **Note:** The remaining tasks presume that you are logged in to a server as `root`. You can enter the commands from any server by using the `dcli` command. This example runs the `pwd` command on node03 from any node in the cluster:
>
> ```
> # dcli -c node03 pwd
> ```

---

2. Verify that the Cloudera Manager started automatically on node03:

```
# service cloudera-scm-server status
cloudera-scm-server (pid  11399) is running...
```

3. If it is not running, then start it:

```
# service cloudera-scm-server start
```

4. Log in to Cloudera Manager as the `admin` user.

5. In the Status pane of the opening page, expand the menu for the cluster and click **Start**, and then click **Start** again when prompted to confirm. See Figure 2–9.

    To navigate to this page, click the **Home** tab, and then the **Status** subtab.

6. On the Command Details page, click **Close** when all processes are started.

7. In the same pane under Cloudera Management Services, expand the menu for the `mgmt` service and click **Start**.

8. Log out of Cloudera Manager (optional).

Task 4 Starting Oracle Data Integrator Agent

If Oracle Data Integrator Application Adapter for Hadoop is used on this cluster, then start the agent.

1. Check the status of the agent:

```
# /opt/oracle/odiagent/agent_standalone/oracledi/agent/bin/startcmd.sh
OdiPingAgent [-AGENT_NAME=agent_name]
```

2. Start the agent:

```
# /opt/oracle/odiagent/agent_standalone/oracledi/agent/bin/agent.sh
[-NAME=agent_name] [-PORT=port_number]
```

Security on Oracle Big Data Appliance

You can take precautions to prevent unauthorized use of the software and data on Oracle Big Data Appliance.

This section contains these topics:

- About Predefined Users and Groups
- About User Authentication
- Port Numbers Used on Oracle Big Data Appliance
- About Puppet Security

About Predefined Users and Groups

Every open-source package installed on Oracle Big Data Appliance creates one or more users and groups. Most of these users do not have login privileges, shells, or home directories. They are used by daemons and are not intended as an interface for individual users. For example, Hadoop operates as the `hdfs` user, MapReduce operates as `mapred`, and Hive operates as `hive`.

You can use the `oracle` identity to run Hadoop and Hive jobs immediately after the Oracle Big Data Appliance software is installed. This user account has login privileges, a shell, and a home directory.

Oracle NoSQL Database and Oracle Data Integrator run as the `oracle` user. Its primary group is `oinstall`.

> **Note:** Do not delete, re-create, or modify the users that are created during installation, because they are required for the software to operate.

Table 2–7 identifies the operating system users and groups that are created automatically during installation of Oracle Big Data Appliance software for use by CDH components and other software packages.

Table 2–7   Operating System Users and Groups

User Name	Group	Used By	Login Rights
flume	flume	**Flume NG** parent and nodes	No
hbase	hbase	**HBase** processes	No
hdfs	hadoop	**NameNode**, **DataNode**	No
hive	hive	Hive metastore and server processes	No
hue	hue	**Hue** processes	No
mapred	hadoop	JobTracker, TaskTracker, **Hive Thrift** daemon	Yes
mysql	mysql	MySQL server	Yes
oozie	oozie	**Oozie** server	No
oracle	dba, oinstall	Oracle NoSQL Database, Oracle Loader for Hadoop, Oracle Data Integrator, and the Oracle DBA	Yes
puppet	puppet	Puppet parent (puppet nodes run as root)	No
sqoop	sqoop	**Sqoop** metastore	No
svctag		Auto Service Request	No
zookeeper	zookeeper	**ZooKeeper** processes	No

About User Authentication

Oracle Big Data Appliance supports Kerberos security as a software installation option. See Chapter 3 for details about setting up clients and users to access a Kerberos-protected cluster.

Port Numbers Used on Oracle Big Data Appliance

Table 2–8 identifies the port numbers that might be used in addition to those used by CDH.

To view the ports used on a particular server:

1. In Cloudera Manager, click the **Hosts** tab at the top of the page to display the Hosts page.

2. In the Name column, click a server link to see its detail page.

3. Scroll down to the Ports section.

> **See Also:**   For the full list of CDH port numbers, go to the Cloudera website at
>
> http://www.cloudera.com/content/cloudera-content/cloudera-do cs/CDH4/latest/CDH4-Installation-Guide/cdh4ig_topic_9.html

Table 2–8   Oracle Big Data Appliance Port Numbers

Service	Port
Automated Service Monitor (ASM)	30920

Table 2–8   (Cont.) Oracle Big Data Appliance Port Numbers

Service	Port
HBase master service (node01)	60010
MySQL Database	3306
Oracle Data Integrator Agent	20910
Oracle NoSQL Database administration	5001
Oracle NoSQL Database processes	5010 to 5020
Oracle NoSQL Database registration	5000
Port map	111
Puppet master service	8140
Puppet node service	8139
rpc.statd	668
ssh	22
xinetd (service tag)	6481

About Puppet Security

The puppet node service (`puppetd`) runs continuously as `root` on all servers. It listens on port 8139 for "kick" requests, which trigger it to request updates from the puppet master. It does not receive updates on this port.

The puppet master service (`puppetmasterd`) runs continuously as the puppet user on the first server of the primary Oracle Big Data Appliance rack. It listens on port 8140 for requests to push updates to puppet nodes.

The puppet nodes generate and send certificates to the puppet master to register initially during installation of the software. For updates to the software, the puppet master signals ("kicks") the puppet nodes, which then request all configuration changes from the puppet master node that they are registered with.

The puppet master sends updates only to puppet nodes that have known, valid certificates. Puppet nodes only accept updates from the puppet master host name they initially registered with. Because Oracle Big Data Appliance uses an internal network for communication within the rack, the puppet master host name resolves using `/etc/hosts` to an internal, private IP address.

Auditing Oracle Big Data Appliance

You can use Oracle Audit Vault and Database Firewall to create and monitor the audit trails for HDFS and MapReduce on Oracle Big Data Appliance.

This section describes the Oracle Big Data Appliance plug-in:

- About Oracle Audit Vault and Database Firewall
- Setting Up the Oracle Big Data Appliance Plug-in
- Monitoring Oracle Big Data Appliance

About Oracle Audit Vault and Database Firewall

Oracle Audit Vault and Database Firewall secures databases and other critical components of IT infrastructure in these key ways:

- Provides an integrated auditing platform for your enterprise.

- Captures activity on Oracle Database, Oracle Big Data Appliance, operating systems, directories, file systems, and so forth.

- Makes the auditing information available in a single reporting framework so that you can understand the activities across the enterprise. You do not need to monitor each system individually; you can view your computer infrastructure as a whole.

Audit Vault Server provides a web-based, graphic user interface for both administrators and auditors.

You can configure CDH/Hadoop clusters on Oracle Big Data Appliance as secured targets. The Audit Vault plug-in on Oracle Big Data Appliance collects audit and logging data from these services:

- **HDFS**: Who makes changes to the file system.

- **MapReduce**: Who runs MapReduce jobs that correspond to file access.

- **Hive DDL**: Who makes Hive database changes.

- **Oozie workflows**: Who runs workflow activities.

The Audit Vault plug-in is an installation option. The Mammoth utility automatically configures monitoring on Oracle Big Data Appliance as part of the software installation process.

> **See Also:** For more information about Oracle Audit Vault and Database Firewall:
>
> http://www.oracle.com/technetwork/database/database-technolo gies/audit-vault-and-database-firewall/overview/index.html

Setting Up the Oracle Big Data Appliance Plug-in

The Mammoth utility on Oracle Big Data Appliance performs all the steps needed to setup the plug-in, using information that you provide.

To set up the Audit Vault plug-in for Oracle Big Data Appliance:

1. Ensure that Oracle Audit Vault and Database Firewall Server Release 12.1.1 is up and running on the same network as Oracle Big Data Appliance.

   > **See Also:** *Oracle Audit Vault and Database Firewall Installation Guide*

2. Complete the Audit Vault Plug-in section of Oracle Big Data Appliance Configuration Generation Utility.

3. Install the Oracle Big Data Appliance software using the Mammoth utility. An Oracle representative typically performs this step.

When the software installation is complete, the Audit Vault plug-in is installed on Oracle Big Data Appliance, and Oracle Audit Vault and Database Firewall is collecting its audit information. You do not need to perform any other installation steps.

> **See Also:** *Oracle Big Data Appliance Owner's Guide* for using Oracle Big Data Appliance Configuration Generation Utility

Monitoring Oracle Big Data Appliance

After installing the plug-in, you can monitor Oracle Big Data Appliance the same as any other secured target. Audit Vault Server collects activity reports automatically.

The following procedure describes one type of monitoring activity.

To view an Oracle Big Data Appliance activity report:

1. Log in to Audit Vault Server as an auditor.

2. Click the **Reports** tab.

3. Under Built-in Reports, click **Audit Reports**.

4. To browse all activities, in the Activity Reports list, click the **Browse report data** icon for All Activity.

5. Add or remove the filters to list the events.

   Event names include ACCESS, CREATE, DELETE, and OPEN.

6. Click the **Single row view** icon in the first column to see a detailed report.

Figure 2–10 shows the beginning of an activity report, which records access to a Hadoop sequence file.

Figure 2–10   Activity Report in Audit Vault Server

See Also: *Oracle Audit Vault and Database Firewall Auditor's Guide*

Collecting Diagnostic Information for Oracle Customer Support

If you need help from Oracle Support to troubleshoot CDH issues, then you should first collect diagnostic information using the bdadiag utility with the cm option.

To collect diagnostic information:

1. Log in to an Oracle Big Data Appliance server as root.

2. Run bdadiag with at least the cm option. You can include additional options on the command line as appropriate. See the *Oracle Big Data Appliance Owner's Guide* for a complete description of the bdadiag syntax.

   ```
   # bdadiag cm
   ```

   The command output identifies the name and the location of the diagnostic file.

3. Go to My Oracle Support at http://support.oracle.com.

4. Open a Service Request (SR) if you have not already done so.

5. Upload the bz2 file into the SR. If the file is too large, then upload it to sftp.oracle.com, as described in the next procedure.

To upload the diagnostics to ftp.oracle.com:

1. Open an SFTP client and connect to sftp.oracle.com. Specify port 2021 and remote directory /support/incoming/*target*, where *target* is the folder name given to you by Oracle Support.

   See Example 2–1 if you are using a command-line SFTP client.

2. Log in with your Oracle Single Sign-on account and password.

3. Upload the diagnostic file to the new directory.

4. Update the SR with the full path and the file name.

Example 2–1 shows the commands to upload the diagnostics using the SFTP command interface.

Example 2–1   Uploading Diagnostics Using FTP

```
$ sftp -o port=2021 my.user.name@oracle.com@sftp.oracle.com
Connecting to sftp.oracle.com...
    .
    .
    .
Enter password for my.user.name@oracle.com
Password: password
sftp> cd support/incoming/SR123456
sftp> put /tmp/bdadiag_bda1node01_1216FM5497_2013_07_18_07_33.tar.bz2
Uploading bdadiag_bda1node01_1216FM5497_2013_07_18_07_33.tar.bz2 to
//support/incoming/create_table.sql
bdadiag_bda1node01_1216FM5497_2013_07_18_07_33.tar.bz2 to support/incoming/create_
table.sql                   100%  311    0.3KB/s   00:00
sftp> exit
$
```

> **See Also:**   My Oracle Support Note 549180.1 at
>
> http://support.oracle.com

3

Supporting User Access to Oracle Big Data Appliance

This chapter describes how you can support users who run MapReduce jobs on Oracle Big Data Appliance or using Oracle Big Data Connectors. It contains these sections:

- About Accessing a Kerberos-Secured Cluster
- Providing Remote Client Access to CDH
- Providing Remote Client Access to Hive
- Managing User Accounts
- Recovering Deleted Files

About Accessing a Kerberos-Secured Cluster

Apache Hadoop is not an inherently secure system. It is protected only by network security. After a connection is established, a client has full access to the system.

To counterbalance this open environment, Oracle Big Data Appliance supports Kerberos security as a software installation option. Kerberos is a network authentication protocol that helps prevent malicious impersonation.

CDH provides these securities when configured to use Kerberos:

- The CDH master nodes, NameNode, and JobTracker resolve the group name so that users cannot manipulate their group memberships.
- Map tasks run under the identity of the user who submitted the job.
- Authorization mechanisms in HDFS and MapReduce help control user access to data.

If the Oracle Big Data Appliance cluster is secured with Kerberos, then you must take additional steps to authenticate a CDH client and individual users, as described in this chapter. Users must know their Kerberos user name, password, and realm.

Table 3–1 describes some frequently used Kerberos commands. For more information, see the MIT Kerberos documentation.

Table 3–1   Kerberos User Commands

Command	Description
kinit *userid@realm*	Obtains a Kerberos ticket.
klist	Lists a Kerberos ticket if you have one already.
kdestroy	Invalidates a ticket before it expires.

Table 3–1   (Cont.) Kerberos User Commands

Command	Description
kpasswd *userid@realm*	Changes your password.

> **See Also:**
> - MIT Kerberos Documentation at
>
>   http://web.mit.edu/kerberos/krb5-latest/doc/
> - *CDH4 Security Guide* at
>
>   http://oracle.cloudera.com

Providing Remote Client Access to CDH

Oracle Big Data Appliance supports full local access to all commands and utilities in Cloudera's Distribution including Apache Hadoop (CDH).

You can use a browser on any computer that has access to the client network of Oracle Big Data Appliance to access Cloudera Manager, Hadoop Map/Reduce Administration, the Hadoop Task Tracker interface, and other browser-based Hadoop tools.

To issue Hadoop commands remotely, however, you must connect from a system configured as a CDH client with access to the Oracle Big Data Appliance client network. This section explains how to set up a computer so that you can access HDFS and submit MapReduce jobs on Oracle Big Data Appliance.

> **See Also:**   My Oracle Support ID 1506203.1

Prerequisites

Ensure that you have met the following prerequisites:

- You must have these access privileges:
  - Sudo access to the client system
  - Login access to Cloudera Manager

  If you do not have these privileges, then contact your system administrator for help.

- The client system must run an operating system that Cloudera supports for CDH4. See the *Cloudera CDH4 Installation Guide* at

  http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/latest/CDH4-Requirements-and-Supported-Versions/cdhrsv_topic_1.html

- The client system must run Oracle JDK 1.6_28 or later. CDH4 is certified with 1.6.0_31 and 1.7.0_15.

  To verify the version, use this command:

  ```
  $ java -version
  java version "1.7.0_25"
  Java(TM) SE Runtime Environment (build 1.7.0_25-b15)
  Java HotSpot(TM) 64-Bit Server VM (build 23.25-b01, mixed mode)
  ```

Installing CDH on Oracle Exadata Database Machine

When you use Oracle Exadata Database Machine as the client, you can use the RPM files on Oracle Big Data Appliance when both engineered systems use the same operating system (Oracle Linux 5.*x* or 6.*x*). Copying the files across the local network is faster than downloading them from the Cloudera website.

> **Note:**   In the following steps, replace *version_number* with the missing portion of the file name, such as `2.2.0+189-1.cdh4.2.0.p0.8.el5`.

To install a CDH client on Oracle Exadata Database Machine:

1. Log into an Exadata database server.

2. Verify that Hadoop is not installed on your Exadata system:

    ```
    rpm -qa | grep hadoop
    ```

3. If the `rpm` command returns a value, then remove the existing Hadoop software:

    ```
    rpm -e hadoop_rpm
    ```

4. Copy the following Linux RPMs to the database server from the first server of Oracle Big Data Appliance. The RPMs are located in the `/opt/oracle/BDAMammoth/bdarepo/RPMS/x86_64` directory.

    - `ed-version_number.x86_64.rpm`

    - `m4-version_number.x86_64.rpm`

    - `nc-version_number.x86_64.rpm`

    - `redhat-lsb-version_number.x86_64.rpm`

5. Install the Oracle Linux RPMs from Step 4 on all database nodes. For example:

    ```
    sudo yum --nogpgcheck localinstall ed-0.2-39.el5_2.x86_64.rpm
    sudo yum --nogpgcheck localinstall m4-1.4.5-3.el5.1.x86_64.rpm
    sudo yum --nogpgcheck localinstall nc-1.84-10.fc6.x86_64.rpm
    sudo yum --nogpgcheck localinstall redhat-lsb-4.0-2.1.4.0.2.el5.x86_64.rpm
    ```

    Be sure to install the Oracle Linux RPMs before installing the CDH RPMs.

6. Copy the following CDH RPMs from the `/opt/oracle/BDAMammoth/bdarepo/RPMS/noarch` directory:

    - `bigtop-utils-version_number.noarch.rpm`

    - `zookeeper-version_number.noarch.rpm`

7. Copy the following CDH RPMs from the `/opt/oracle/BDAMammoth/bdarepo/RPMS/x86_64` directory:

    - `hadoop-version_number.x86_64.rpm`

    - `bigtop-jsvc-version_number.x86_64.rpm`

    - `hadoop-hdfs-version_number.x86_64.rpm`

    - `hadoop-0.20-mapreduce-version_number.x86_64.rpm`

    - `hadoop-yarn-version_number.x86_64.rpm`

    - `hadoop-mapreduce-version_number.x86_64.rpm`

    - `hadoop-client-version_number.x86_64.rpm`

8. Install the CDH RPMs in the exact order shown in Steps 6 and 7 on all database servers. For example:

```
rpm -ihv /bigtop-utils-0.4+502-1.cdh4.2.0.p0.12.el5.noarch.rpm
rpm -ihv zookeeper-3.4.5+14-1.cdh4.2.0.p0.12.el5.noarch.rpm
rpm -ihv hadoop-2.0.0+922-1.cdh4.2.0.p0.12.el5.x86_64.rpm
rpm -ihv bigtop-jsvc-1.0.10-1.cdh4.2.0.p0.13.el5.x86_64.rpm
rpm -ihv hadoop-hdfs-2.0.0+922-1.cdh4.2.0.p0.12.el5.x86_64.rpm
rpm -ihv hadoop-0.20-mapreduce-0.20.2+1341-1.cdh4.2.0.p0.21.el5.x86_64.rpm
rpm -ihv hadoop-yarn-2.0.0+922-1.cdh4.2.0.p0.12.el5.x86_64.rpm
rpm -ihv hadoop-mapreduce-2.0.0+922-1.cdh4.2.0.p0.12.el5.x86_64.rpm
rpm -ihv hadoop-client-2.0.0+922-1.cdh4.2.0.p0.12.el5.x86_64.rpm
```

9. Configure the CDH client. See "Configuring a CDH Client for an Unsecured Cluster" on page 3-4.

Installing a CDH Client on Any Supported Operating System

To install a CDH client on any operating system identified as supported by Cloudera, follow these instructions.

To install the CDH client software:

1. Log in to the client system.

2. If an earlier version of Hadoop is already installed, then remove it.

   See the Cloudera documentation for removing CDH3 at

   http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/latest/CDH4-Installation-Guide/cdh4ig_topic_6_3.html

3. Download the CDH 4.5 software from the Cloudera website:

   ```
   $ wget http://archive.cloudera.com/cdh4/cdh/4/hadoop-2.0.0-cdh4.5.0.tar.gz
   ```

4. Decompress the file into a permanent location, which will be the Hadoop home directory. The following command unzips the files into hadoop-2.0.0-cdh4.4.0 in the current directory:

   ```
   tar -xvzf hadoop-2.0.0-cdh4.5.0.tar.gz
   ```

5. Configure the CDH client. See "Configuring a CDH Client for an Unsecured Cluster" on page 3-4.

Configuring a CDH Client for an Unsecured Cluster

After installing CDH, you must configure it for use with Oracle Big Data Appliance.

To configure the Hadoop client:

1. Log in to the client system and download the MapReduce client configuration from Cloudera Manager. In this example, Cloudera Manager listens on port 7180 (the default) of bda01node03.example.com, and the configuration is stored in a file named mapreduce-conf.zip.

   ```
   $ wget -O mapreduce-conf.zip
   http://bda01node03.examplecom:7180/cmf/services/2/client-config
   ```

2. Unzip mapreduce-config.zip into a permanent location on the client system.

   ```
   $ unzip mapreduce-config.zip
   Archive:  mapreduce-clientconfig.zip
   ```

```
inflating: hadoop-conf/hadoop-env.sh
inflating: hadoop-conf/core-site.xml
inflating: hadoop-conf/hdfs-site.xml
inflating: hadoop-conf/log4j.properties
inflating: hadoop-conf/mapred-site.xml
```

All files are stored in a subdirectory named `hadoop-config`.

3. Open `hadoop-env.sh` in a text editor and set the environment variables to the actual paths on your system:

```
export HADOOP_HOME=hadoop-home-dir/share/hadoop/mapreduce1
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=hadoop-conf-dir
export JAVA_HOME=/usr/java/version
ln -s $HADOOP_HOME/../../../bin-mapreduce1 $HADOOP_HOME/bin
ln -s $HADOOP_HOME/../../../etc/hadoop-mapreduce1 $HADOOP_HOME/conf
rm -f $HADOOP_HOME/lib/slf4j-log4j*jar
alias hadoop=$HADOOP_HOME/bin/hadoop
alias hdfs=$HADOOP_HOME/../../../bin/hdfs
```

4. Make a backup copy of the Hadoop configuration files:

```
# cp /full_path/hadoop-conf /full_path/hadoop-conf-bak
```

5. Overwrite the existing configuration files with the downloaded configuration files in Step 2.

```
# cd /full_path/hadoop-conf
# cp * /usr/lib/hadoop/conf
```

Configuring a CDH Client for a Kerberos-Secured Cluster

Follow these steps to enable the CDH client to work with a secure CDH cluster.

To configure a CDH client for Kerberos:

1. Log in to the system where you created the CDH client.

2. Install the Java Cryptography Extension Unlimited Strength Jurisdiction Policy Files:

    a. Download the files for your Java version:

        Java 6:
        http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html

        Java 7:
        http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html

    b. Decompress the downloaded file. This example unzips JCE-7:

```
$ unzip UnlimitedJCEPolicyJDK7.zip
Archive:  UnlimitedJCEPolicyJDK7.zip
  creating: UnlimitedJCEPolicy/
 inflating: UnlimitedJCEPolicy/US_export_policy.jar
 inflating: UnlimitedJCEPolicy/local_policy.jar
 inflating: UnlimitedJCEPolicy/README.txt
```

> **Note:** The JCE-6 files unzip into a directory named jce instead of UnlimitedJCEPolicy.

    c. Copy the unzipped files into the Java security directory. For example:

```
$ cp UnlimitedJCEPolicy/* /usr/java/latest/jre/lib/security/
```

3. Follow the steps for configuring an unsecured client.

See "Configuring a CDH Client for an Unsecured Cluster" on page 3-4.

4. Ensure that you have a user ID on the CDH cluster that had been added to the Kerberos realm.

See "Creating Hadoop Cluster Users" on page 3-9.

5. On the CDH client system, create a file named krb5.conf in the $HADOOP_CONF_DIR directory. Enter configuration settings like the following, using values appropriate for your installation for the server names, domain, and realm:

```
[libdefaults]
     default_realm = EXAMPLE.COM
     dns_lookup_realm = false
     dns_lookup_kdc = false
     clockskew = 3600
     ticket_lifetime = 24h
     renew_lifetime = 7d
     forwardable = true

[realms]
     EXAMPLE.COM = {
          kdc = bda01node01.example:88
          admin_server = bda01node07:749
          default_domain = example.com
 }

[domain_realm]
        .com = EXAMPLE.COM
```

6. Activate the new configuration file:

```
export KRB5_CONFIG=$HADOOP_CONF_DIR/krb5.conf
export HADOOP_OPTS="-Djava.security.krb5.conf=$HADOOP_CONF_DIR/krb5.conf"
export KRB5CCNAME=$HADOOP_CONF_DIR/krb5cc_$USER
```

7. Verify that you have access to the Oracle Big Data Appliance cluster.

See "Verifying Access to a Cluster from the CDH Client" on page 3-6.

Verifying Access to a Cluster from the CDH Client

Follow this procedure to ensure that you have access to the Oracle Big Data Appliance cluster.

To verify cluster access:

1. To access a Kerberos-protected CDH cluster, first obtain a ticket granting ticket (TGT):

```
$ kinit userid@realm
```

2. Verify that you can access HDFS on Oracle Big Data Appliance from the client, by entering a simple Hadoop file system command like the following:

```
$ hadoop fs -ls /user
Found 4 items
drwx------   - hdfs     supergroup          0 2013-01-16 13:50 /user/hdfs
drwxr-xr-x   - hive     supergroup          0 2013-01-16 12:58 /user/hive
drwxr-xr-x   - oozie    hadoop              0 2013-01-16 13:01 /user/oozie
drwxr-xr-x   - oracle   hadoop              0 2013-01-29 12:50 /user/oracle
```

Check the output for HDFS users defined on Oracle Big Data Appliance, and not on the client system. You should see the same results as you would after entering the command directly on Oracle Big Data Appliance.

3. Submit a MapReduce job. You must be logged in to the client system under the same user name as your HDFS user name on Oracle Big Data Appliance.

The following example calculates the value of *pi*:

```
$ hadoop jar
/usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples-2.0.0-cdh4.2.0.jar pi 10
1000000
Number of Maps  = 10
Samples per Map = 1000000
Wrote input for Map #0
Wrote input for Map #1
     .
     .
     .
13/04/30 08:15:50 INFO mapred.JobClient:     BYTES_READ=240
Job Finished in 12.403 seconds
Estimated value of Pi is 3.14158440000000000000
```

4. Use Cloudera Manager to verify that the job ran on Oracle Big Data Appliance instead of the local system. Select **mapreduce** from the Activities menu for a list of jobs.

Figure 3–1 shows the job created by the previous example.

Figure 3–1   Monitoring a MapReduce Job in Cloudera Manager

Providing Remote Client Access to Hive

Follow this procedure to provide remote client access to Hive.

To set up a Hive client:

1. Set up a CDH client. See "Providing Remote Client Access to CDH" on page 3-2.

2. Log in to the client system and download the Hive client configuration from Cloudera Manager. In this example, Cloudera Manager listens on port 7180 (the default) of bda01node03.example.com, and the configuration is stored in a file named hive-conf.zip.

   ```
   $ wget -O hive-conf.zip
   http://bda01node03.examplecom:7180/cmf/services/5/client-config
   ```

3. Unzip the file into a permanent installation directory, which will be the Hive configuration directory:

   ```
   $ unzip hive-conf.zip
   Archive:  hive-conf.zip
     inflating: hive-conf/hive-env.sh
     inflating: hive-conf/hive-site.xml
   ```

4. Download the Hive software from the Cloudera website:

   ```
   $ wget http://archive.cloudera.com/cdh4/cdh/4/hive-0.10.0-cdh4.4.0.tar.gz
   ```

5. Decompress the file into a permanent installation directory, which will be the Hive home directory. The following command unzips the files into the current directory in a subdirectory named hive-0.10.0-cdh4.4.0:

   ```
   $ tar -xvzf hive-0.10.0-cdh4.4.0.tar.gz
   ```

6. Set the following variables, replacing *hive-home-dir* and *hive-conf-dir* with the directories you created in steps 3 and 5.

   ```
   export HIVE_HOME=hive-home-dir
   export HIVE_CONF_DIR=hive-conf-dir
   alias hive=$HIVE_HOME/bin/hive
   ```

The following steps test whether you successfully set up a Hive client.

To verify Hive access:

1. To access a Kerberos-protected CDH cluster, first obtain a ticket granting ticket (TGT):

   ```
   $ kinit userid@realm
   ```

2. Open the Hive console:

   ```
   $ hive
   Logging initialized using configuration in
   jar:file:/usr/lib/hive/lib/hive-common-0.10.0-cdh4.4.0.jar!/hive-log4j.properti
   es
   Hive history file=/tmp/oracle/hive_job_log_
   e10527ee-9637-4c08-9559-a2e5cea6cef1_831268640.txt
   hive>
   ```

3. List all tables:

   ```
   hive> show tables;
   OK
   src
   ```

Managing User Accounts

This section describes how to create users who can access HDFS, MapReduce, and Hive. It contains the following topics:

- Creating Hadoop Cluster Users
- Providing User Login Privileges (Optional)

Creating Hadoop Cluster Users

When creating user accounts, define them as follows:

- To run MapReduce jobs, users must be in the `hadoop` group.
- To create and modify tables in Hive, users must be in the `hive` group.
- To create Hue users, open Hue in a browser and click the User Admin icon. See "Using Cloudera Hue to Interact With Hadoop" on page 2-7.

Creating Users on an Unsecured Cluster

To create a user on an unsecured Hadoop cluster:

1. Open an ssh connection as the `root` user to a noncritical node (node04 to node18).

2. Create the user's home directory:

   ```
   # sudo -u hdfs hadoop fs -mkdir /user/user_name
   ```

   You use `sudo` because the HDFS super user is `hdfs` (not `root`).

3. Change the ownership of the directory:

   ```
   # sudo -u hdfs hadoop fs -chown user_name:hadoop /user/user_name
   ```

4. Verify that the directory is set up correctly:

   ```
   # hadoop fs -ls /user
   ```

5. Create the operating system user across all nodes in the cluster:

   ```
   # dcli useradd -G hadoop,hive[,group_name...] -m user_name
   ```

   In this syntax, replace *group_name* with an existing group and *user_name* with the new name.

6. Verify that the operating system user belongs to the correct groups:

   ```
   # dcli id user_name
   ```

7. Verify that the user's home directory was created on all nodes:

   ```
   # dcli ls /home | grep user_name
   ```

Example 3–1 creates a user named `jdoe` with a primary group of `hadoop` and an addition group of `hive`.

Example 3–1   Creating a Hadoop User

```
# sudo -u hdfs hadoop fs -mkdir /user/jdoe
# sudo -u hdfs hadoop fs -chown jdoe:hadoop /user/jdoe
# hadoop fs -ls /user
Found 5 items
drwx------   - hdfs     supergroup          0 2013-01-16 13:50 /user/hdfs
```

```
drwxr-xr-x   - hive      supergroup          0 2013-01-16 12:58 /user/hive
drwxr-xr-x   - jdoe      jdoe                0 2013-01-18 14:04 /user/jdoe
drwxr-xr-x   - oozie     hadoop              0 2013-01-16 13:01 /user/oozie
drwxr-xr-x   - oracle    hadoop              0 2013-01-16 13:01 /user/oracle
# dcli useradd -G hadoop,hive -m jdoe
# dcli id jdoe
bda1node01: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),127(hive),123(hadoop)
bda1node02: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),123(hadoop),127(hive)
bda1node03: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),123(hadoop),127(hive)
     .
     .
     .
# dcli ls /home | grep jdoe
bda1node01: jdoe
bda1node02: jdoe
bda1node03: jdoe
```

Creating Users on a Secured Cluster

To create a user on a Kerberos-secured cluster:

1. Connect to Kerberos as the HDFS principal and execute the following commands, replacing jdoe with the actual user name:

    ```
    hdfs dfs -mkdir /user/jdoe
    hdfs dfs -chown jdoe /user/jdoe
    dcli -C useradd -G hadoop,hive -m jdoe
    hash=$(echo "hadoop" | openssl passwd -1 -stdin)
    dcli -C "usermod --pass='$hash' jdoe"
    ```

2. Log in to the key distribution center (KDC) and add a principal for the user. In the following example, replace jdoe, bda01node01, and example.com with the correct user name, server name, domain, and realm.

    ```
    ssh -l root bda01node01.example.com kadmin.local
    add_principal user_name@EXAMPLE.COM
    ```

Providing User Login Privileges (Optional)

Users do not need login privileges on Oracle Big Data Appliance to run MapReduce jobs from a remote client. However, for those who want to log in to Oracle Big Data Appliance, you must set a password. You can set or reset a password the same way.

To set a user password across all Oracle Big Data Appliance servers:

1. Create a Hadoop cluster user as described in "Creating Hadoop Cluster Users" on page 3-9..

2. Confirm that the user does not have a password:

    ```
    # dcli passwd -S user_name
    bda1node01.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
    bda1node02.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
    bda1node03.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
    ```

    If the output shows either "Empty password" or "Password locked," then you must set a password.

3. Set the password:

    ```
     hash=$(echo 'password' | openssl passwd -1 -stdin); dcli "usermod
    --pass='$hash' user_name"
    ```

4. Confirm that the password is set across all servers:

```
# dcli passwd -S user_name
bda1node01.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
bda1node02.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
bda1node03.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
```

> **See Also:**
>
> - *Oracle Big Data Appliance Owner's Guide* for information about `dcli`.
> - The Linux `man` page for the full syntax of the `useradd` command.

Recovering Deleted Files

CDH provides an optional trash facility, so that a deleted file or directory is moved to a trash directory for a set period, instead of being deleted immediately from the system. By default, the trash facility is enabled for HDFS and all HDFS clients.

Restoring Files from the Trash

When the trash facility is enabled, you can easily restore files that were previously deleted.

To restore a file from the trash directory:

1. Check that the deleted file is in the trash. The following example checks for files deleted by the `oracle` user:

```
$ hadoop fs -ls .Trash/Current/user/oracle
Found 1 items
-rw-r--r--  3 oracle hadoop  242510990 2012-08-31 11:20
/user/oracle/.Trash/Current/user/oracle/ontime_s.dat
```

2. Move or copy the file to its previous location. The following example moves ontime_s.dat from the trash to the HDFS /user/oracle directory.

```
$ hadoop fs -mv .Trash/Current/user/oracle/ontime_s.dat /user/oracle/ontime_
s.dat
```

Changing the Trash Interval

The **trash interval** is the minimum number of minutes that a file remains in the trash directory before being deleted permanently from the system. The default value is 1 day (24 hours).

To change the trash interval:

1. Open Cloudera Manager. See "Managing CDH Operations Using Cloudera Manager" on page 2-3.

2. On the All Services page under Name, click **hdfs**.

3. On the hdfs page, click **Configuration**, and then select **View and Edit.**

4. Search for or scroll down to the Filesystem Trash Interval property under NameNode Settings. See Figure 3–2.

5. Click the current value, and enter a new value in the pop-up form.

6. Click **Save Changes**.

7. Expand the Actions menu at the top of the page and choose **Restart**.

8. Open a connection as `root` to a node in the cluster.

9. Deploy the new configuration:

```
dcli -C bdagetclientconfig
```

Figure 3–2 shows the Filesystem Trash Interval property in Cloudera Manager.

Figure 3–2   HDFS Property Settings in Cloudera Manager

Disabling the Trash Facility

The trash facility on Oracle Big Data Appliance is enabled by default. You can change this configuration for a cluster. When the trash facility is disabled, deleted files and directories are not moved to the trash. They are not recoverable.

Completely Disabling the Trash Facility

The following procedure disables the trash facility for HDFS. When the trash facility is completely disabled, the client configuration is irrelevant.

To completely disable the trash facility:

1. Open Cloudera Manager. See "Managing CDH Operations Using Cloudera Manager" on page 2-3.

2. On the All Services page under Name, click **hdfs**.

3. On the hdfs page, click the **Configuration** subtab.

4. Search for or scroll down to the Filesystem Trash Interval property under NameNode Settings. See Figure 3–2.

5. Click the current value, and enter a value of 0 (zero) in the pop-up form.

6. Click **Save Changes**.

7. Expand the Actions menu at the top of the page and choose **Restart**.

Disabling the Trash Facility for Local HDFS Clients

All HDFS clients that are installed on Oracle Big Data Appliance are configured to use the trash facility. An **HDFS client** is any software that connects to HDFS to perform operations such as listing HDFS files, copying files to and from HDFS, and creating directories.

You can use Cloudera Manager to change the local client configuration setting, although the trash facility is still enabled.

> **Note:** If you do not want any clients to use the trash, then you can completely disable the trash facility. See "Completely Disabling the Trash Facility" on page 3-12.

To disable the trash facility for local HDFS clients:

1. Open Cloudera Manager. See "Managing CDH Operations Using Cloudera Manager" on page 2-3.

2. On the All Services page under Name, click **hdfs**.

3. On the hdfs page, click the **Configuration** subtab.

4. Search for or scroll down to the Use Trash property under Client Settings. See Figure 3–2.

5. Deselect the Use Trash check box.

6. Click **Save Changes**. This setting is used to configure all new HDFS clients downloaded to Oracle Big Data Appliance.

7. Open a connection as `root` to a node in the cluster.

8. Deploy the new configuration:

```
dcli -C bdagetclientconfig
```

Disabling the Trash Facility for a Remote HDFS Client

Remote HDFS clients are typically configured by downloading and installing a CDH client, as described in "Providing Remote Client Access to CDH" on page 3-2. Oracle SQL Connector for HDFS and Oracle R Advanced Analytics for Hadoop are examples of remote clients.

To disable the trash facility for a remote HDFS client:

1. Open a connection to the system where the CDH client is installed.

2. Open `/etc/hadoop/conf/hdfs-site.xml` in a text editor.

3. Change the trash interval to zero:

```
<property>
     <name>fs.trash.interval</name>
     <value>0</value>
</property>
```

4. Save the file.

4

Optimizing MapReduce Jobs Using Perfect Balance

This chapter describes how you can shorten the run time of some MapReduce jobs by using Perfect Balance. It contains the following sections:

- What is Perfect Balance?
- Application Requirements
- Getting Started with Perfect Balance
- Analyzing a Job Reducer Load
- About Configuring Perfect Balance
- Running a Balanced MapReduce Job Using Perfect Balance
- About Perfect Balance Reports
- About Chopping
- Troubleshooting Jobs Running with Perfect Balance
- Using the Perfect Balance API
- About the Perfect Balance Examples
- Perfect Balance Configuration Property Reference

What is Perfect Balance?

The Perfect Balance feature of Oracle Big Data Appliance distributes the reducer load in a MapReduce application so that each reduce task does approximately the same amount of work. While the default Hadoop method of distributing the reduce load is appropriate for many jobs, it does not distribute the load evenly for jobs with significant data skew.

Data skew is an imbalance in the load assigned to different reduce tasks. The **load** is a function of:

- The number of keys assigned to a reducer.
- The number of records and the number of bytes in the values per key.

The total run time for a job is extended, to varying degrees, by the time that the reducer with the greatest load takes to finish. In jobs with a skewed load, some reducers complete the job quickly, while others take much longer. Perfect Balance can significantly shorten the total run time by distributing the load evenly, enabling all reducers to finish at about the same time.

Your MapReduce job can be written using either the `mapred` or `mapreduce` APIs; Perfect Balance supports both of them.

Perfect Balance was tested on MapReduce 1 (MRv1) CDH clusters, which is the default installation on Oracle Big Data Appliance.

About Balancing Jobs Across Map and Reduce Tasks

A typical Hadoop job has map and reduce tasks. Hadoop distributes the *mapper* workload uniformly across Hadoop Distributed File System (HDFS) and across map tasks, while preserving the data locality. In this way, it reduces skew in the mappers.

Hadoop also hashes the map-output keys uniformly across all *reducers*. This strategy works well when there are many more keys than reducers, and each key represents a very small portion of the workload. However, it is not effective when the mapper output is concentrated into a small number of keys. Hashing these keys results in skew and does not work in applications like sorting, which require range partitioning.

Perfect Balance distributes the load evenly across reducers by first sampling the data, optionally chopping large keys into two or more smaller keys, and using a load-aware partitioning strategy to assign keys to reduce tasks.

Ways to Use Perfect Balance Features

You can choose from two methods of running Perfect Balance features:

- **Perfect Balance**: You can run a job without changing your application code by properly configuring Perfect Balance. This is the preferred method and appropriate for most jobs.

  This method is described in "Running a Balanced MapReduce Job Using Perfect Balance" on page 4-10, and is the primary focus of this chapter.

- **Perfect Balance API**: You can add the code to run Perfect Balance to your application code. Use this method when your application setup code must run before using Perfect Balance. Otherwise, you should use the previous method, which requires no change to your code.

  This method is described in "Using the Perfect Balance API" on page 4-14.

Perfect Balance Components

Perfect Balance has these components:

- **Job Analyzer**: Gathers and reports statistics about the MapReduce job so that you can determine whether to use Perfect Balance.

- **Counting Reducer**: Provides additional statistics to the Job Analyzer to help gauge the effectiveness of Perfect Balance.

- **Load Balancer**: Runs before the MapReduce job to generate a static partition plan, and reconfigures the job to use the plan. The balancer includes a user-configurable, progressive sampler that stops sampling the data as soon as it can generate a good partitioning plan.

- **Driver**: Enables you to run Perfect Balance in a `hadoop` command.

Application Requirements

To use Perfect Balance successfully, your application must meet the following requirements:

■   The job is distributive, so that splitting a group of records associated with a reduce key does not produce incorrect results for the application.

To balance a load, Perfect Balance subpartitions the values of large reduce keys and sends each subpartition to a different reducer. This distribution contrasts with the standard Hadoop practice of sending all values for a single reduce key to the same reducer. Your application must be able to handle output from the reducers that is not fully aggregated, so that it does not produce incorrect results.

This partitioning of values is called **chopping**. Applications that support chopping have **distributive reduce functions**. See "About Chopping" on page 4-12.

If your application is not distributive, then you can still run Perfect Balance after disabling the key-splitting feature. The job still benefits from using Perfect Balance, but the load is not as evenly balanced as it is when key splitting is in effect. See the `oracle.hadoop.balancer.keyLoad.minChopBytes` configuration property to disable key splitting.

■   This release does not support combiners. Perfect Balance detects the presence of combiners and does not balance when they are present.

Getting Started with Perfect Balance

Take the following steps to use Perfect Balance:

1. Ensure that your application meets the requirements listed in "Application Requirements" on page 4-3.

2. Log in to the server where you will submit the job.

3. Run the examples provided with Perfect Balance to become familiar with the product. All examples shown in this chapter are based on the shipped examples and use the same data set. See "About the Perfect Balance Examples" on page 4-16.

4. Set the following variables using the Bash `export` command:

    ■   `BALANCER_HOME`: Set to the Perfect Balance installation directory, such as `/opt/oracle/orabalancer-1.1.1-h2` on Oracle Big Data Appliance (optional). The examples in this chapter use this variable, and you can also define it for your convenience. Perfect Balance does not require `BALANCER_HOME`.

    ■   `HADOOP_CLASSPATH`: Add `${BALANCER_HOME}/jlib/orabalancer-1.1.1.jar` and `${BALANCER_HOME}/jlib/commons-math-2.2.jar` to the existing value. Also add the JAR files for your application.

    To enable Perfect Balance, add `${BALANCER_HOME}/jlib/orabalancerclient-1.1.1.jar`. (API mode does not use this JAR file.)

    ■   `HADOOP_USER_CLASSPATH_FIRST`: Set to `true` to use the Perfect Balance version of `commons-math.jar`.

    You do not need to set these variables unless you are using Perfect Balance.

5. Run Job Analyzer without the balancer and use the generated report to decide whether the job is a good candidate for using Perfect Balance.

    See "Analyzing a Job Reducer Load" on page 4-4.

6. Decide which configuration properties to set. Create a configuration file or enter the settings individually in the `hadoop` command.

   See "About Configuring Perfect Balance" on page 4-9.

7. Run the job using Perfect Balance.

   See "Running a Balanced MapReduce Job Using Perfect Balance" on page 4-10.

8. Use the Job Analyzer report to evaluate the effectiveness of using Perfect Balance. See "Reading the Job Analyzer Report" on page 4-7.

9. Modify the job configuration properties as desired before rerunning the job with Perfect Balance. See "About Configuring Perfect Balance" on page 4-9.

Analyzing a Job Reducer Load

Job Analyzer is a component of Perfect Balance that identifies imbalances in a load, and how effective Perfect Balance is in correcting the imbalance when actually running the job. This section contains the following topics:

- About Job Analyzer
- Running Job Analyzer as a Standalone Utility
- Running Job Analyzer using Perfect Balance
- Reading the Job Analyzer Report

About Job Analyzer

You can use Job Analyzer to decide whether a job is a candidate for load balancing with Perfect Balance. Job Analyzer uses the output logs of a MapReduce job to generate a simple report with statistics like the elapsed time and the load for each reduce task. By default, it uses the standard Hadoop counters displayed by the JobTracker user interface, but organizes the data to emphasize the relative performance and load of the reduce tasks, so that you can more easily interpret the results.

If the report shows that the data is skewed (that is, the reducers processed very different loads and the run times varied widely), then the application is a good candidate for Perfect Balance.

Methods of Running Job Analyzer

You can choose between two methods of running Job Analyzer:

- **As a standalone utility**: Job Analyzer runs against existing job output logs. This is a good choice if you already ran your application on Oracle Big Data Appliance without using Perfect Balance.

- **While using Perfect Balance**: Job Analyzer runs against the output logs for the current job running with Perfect Balance. This is a good choice when you run an application on Oracle Big Data Appliance for the first time, because you do not have existing job output logs to analyze using the standalone Job Analyzer utility.

Running Job Analyzer as a Standalone Utility

As a standalone utility, Job Analyzer provides a quick way to analyze the reduce load of a previously run job.

To run Job Analyzer as a standalone utility:

1. Log in to the server where you will run Job Analyzer.

2. Locate the output logs from the job to analyze. Set `mapred.output.dir` to this directory.

3. Run the Job Analyzer utility as described in "Job Analyzer Utility Syntax" on page 4-5.

4. View the Job Analyzer report in a browser.

Job Analyzer Utility Example

Example 4–1 runs a script that sets the required variables, uses the MapReduce job logs stored in `jdoe_nobal_outdir`, and creates the report in the default location. It then copies the HTML version of the report from HDFS to the `/home/jdoe` local directory and opens the report in a browser.

If you want to run this example, then replace `jdoe_nobal_outdir` with the HDFS path to the output logs from a previous job run. You can use the output logs from running the InvertedIndex example. The default HDFS output directory is in `invindx/output`. See "About the Perfect Balance Examples" on page 4-16.

Example 4–1   Running the Job Analyzer Utility

```
$ cat runja.sh

BALANCER_HOME=/opt/oracle/orabalancer-1.1.1-h2
export HADOOP_CLASSPATH=${BALANCER_HOME}/jlib/orabalancer-1.1.1.jar:${BALANCER_
HOME}/jlib/commons-math-2.2.jar:$HADOOP_CLASSPATH
export HADOOP_USER_CLASSPATH_FIRST=true

hadoop jar ${BALANCER_HOME}/jlib/orabalancer-1.1.1.jar
oracle.hadoop.balancer.tools.JobAnalyzer \
-D mapred.output.dir=jdoe_nobal_outdir

$ sh ./runja.sh
$
$ hadoop fs -get jdoe_nobal_outdir/_balancer/jobanalyzer-report.html /home/jdoe
$ cd /home/jdoe
$ firefox jobanalyzer-report.html
```

Job Analyzer Utility Syntax

The following is the syntax to run the Job Analyzer utility:

```
bin/hadoop jar ${BALANCER_HOME}/jlib/orabalancer-1.1.1
oracle.hadoop.balancer.tools.JobAnalyzer \
-D mapred.output.dir=job_output_dir \
[ja_report_path]
```

job_output_dir
An HDFS directory where the job files are stored from previously executing the application.

ja_report_path
An HDFS directory where Job Analyzer creates its report (optional). The default directory is *job_output_dir*/_balancer.

Running Job Analyzer using Perfect Balance

When you run a job with Perfect Balance, you can configure it to run Job Analyzer automatically. This section contains the following topics:

- Running Job Analyzer with Perfect Balance

- Collecting Additional Metrics

Running Job Analyzer with Perfect Balance

Follow these steps to run Job Analyzer using Perfect Balance:

1. Log in to the server where you will submit the job that uses Perfect Balance.

2. Set up Perfect Balance by taking the steps in "Getting Started with Perfect Balance" on page 4-3.

3. To enable Job Analyzer, set the `oracle.hadoop.balancer.autoAnalyze` configuration property to one of these values:

   - `BASIC_REPORT`: Enables Job Analyzer. If you set `oracle.hadoop.balancer.autoBalance` to `true`, then Perfect Balance automatically sets `oracle.hadoop.balancer.autoAnalyze` to `BASIC_REPORT`.

   - `REDUCER_REPORT`: Configures Job Analyzer to collect additional load statistics. See "Collecting Additional Metrics" on page 4-7.

4. To obtain recommendations for configuring future runs, set `oracle.hadoop.balancer.tools.printRecommendation` to `true`.

5. Decide which additional configuration properties to set, if any.

   See "Perfect Balance Configuration Property Reference" on page 4-17.

6. Run the job.

Example 4–2 runs a script that sets the required variables, uses Perfect Balance to run a job with Job Analyzer and without load balancing, and creates the report in the default location. It then copies the HTML version of the report from HDFS to the `/home/jdoe` local directory and opens the report in a browser. The output includes warnings, which you can ignore.

Example 4–2   Running Job Analyzer with Perfect Balance

```
$ cat ja_nobalance.sh

# set up perfect balance
BALANCER_HOME=/opt/oracle/orabalancer-1.1.1-h2
export HADOOP_CLASSPATH=${BALANCER_
HOME}/jlib/orabalancerclient-1.1.1.jar:${BALANCER_
HOME}/jlib/orabalancer-1.1.jar:${BALANCER_
HOME}/jlib/commons-math-2.2.jar:${HADOOP_CLASSPATH}
export HADOOP_USER_CLASSPATH_FIRST=true

# run the job
hadoop jar application_jarfile.jar ApplicationClass \
 -D application_config_property \
 -D mapred.input.dir=jdoe_application/input \
 -D mapred.output.dir=jdoe_nobal_outdir \
 -D mapred.job.name=nobal \
 -D mapred.reduce.tasks=10 \
 -D oracle.hadoop.balancer.autoBalance=false \
 -D oracle.hadoop.balancer.autoAnalyze=REDUCER_REPORT \
```

```
  -D oracle.hadoop.balancer.tools.printRecommendation=true \
  -conf application_config_file.xml


$ sh ja_nobalance.sh
13/10/23 17:35:09 WARN mapred.JobClient: Use GenericOptionsParser for parsing the
arguments. Applications should implement Tool for the same.
13/10/23 17:35:10 INFO input.FileInputFormat: Total input paths to process : 5
13/10/23 17:35:10 INFO mapred.JobClient: Running job: job_201310212138_0169
13/10/23 17:35:11 INFO mapred.JobClient:  map 0% reduce 0%
13/10/23 17:35:24 INFO mapred.JobClient:  map 31% reduce 0%
    .
    .
    .
13/10/23 17:36:04 INFO mapred.JobClient:     Reduce input records=20000000
13/10/23 17:36:04 INFO mapred.JobClient:     Reduce output records=13871794
13/10/23 17:36:04 INFO mapred.JobClient:     Spilled Records=60000000
13/10/23 17:36:04 INFO mapred.JobClient:     CPU time spent (ms)=268450
13/10/23 17:36:04 INFO mapred.JobClient:     Physical memory (bytes)
snapshot=10195607552
13/10/23 17:36:04 INFO mapred.JobClient:     Virtual memory (bytes)
snapshot=34434174976
13/10/23 17:36:04 INFO mapred.JobClient:     Total committed heap usage
(bytes)=16998989824

$ hadoop fs -get jdoe_nobal_outdir/_balancer/jobanalyzer-report.html /home/jdoe
$ cd /home/jdoe
$ firefox jobanalyzer-report.html
```

Collecting Additional Metrics

The Job Analyzer report includes the load metrics for each key, if you set the
oracle.hadoop.balancer.autoAnalyze property to REDUCER_REPORT. This additional
information provides a more detailed picture of the load for each reducer, with metrics
that are not available in the standard Hadoop counters.

The Job Analyzer report also compares its predicted load with the actual load. The
difference between these values measures how effective Perfect Balance was in
balancing the job.

If you also set oracle.hadoop.balancer.tools.printRecommendation to true, then
Job Analyzer may recommend key load coefficients for the key load model. Use the
recommended values to set the following configuration properties:

- oracle.hadoop.balancer.linearKeyLoad.byteWeight

- oracle.hadoop.balancer.linearKeyLoad.keyWeight

- oracle.hadoop.balancer.linearKeyLoad.rowWeight

When you run the job again, these coefficients result in a more balanced job.

Reading the Job Analyzer Report

Job Analyzer writes its report in two formats: HTML for you, and XML for Perfect
Balance. Copy the HTML version from HDFS to the local file system and open it in a
browser, as shown in Example 4–2.

When inspecting the Job Analyzer report, look for indicators of skew such as:

- The execution time of some reducers is longer than others.

- Some reducers process more records or bytes than others.

- Some map output keys have more records than others.

- Some map output records have more bytes than others.

Figure 4–1 shows the beginning of the analyzer report for the inverted index (invindx) example. It displays the key load coefficient recommendations, because this job ran with the appropriate configuration settings. See "Collecting Additional Metrics" on page 4-7.

The task IDs are links to tables that show the analysis of specific tasks, enabling you to drill down for more details from the first, summary table.

This example uses an extremely small data set, but notice the differences between tasks 7 and 8: The input records range from 3% to 18%, and their corresponding elapsed times range from 6 to 20 seconds. This variation indicates skew.

Figure 4–1   Job Analyzer Report for Unbalanced Inverted Index Job

Job Information

Job Name	inverted_index_pb_jobanalyzer_nobal
Job Id	job_201310212138_0169
Start Time	2013-10-23 17:35:10
Finish Time	2013-10-23 17:36:01
Map Time	00:00:28
Reduce Time	00:00:17
Elapsed Time	00:00:51

Key Load Coefficient Recommendations

The following values are recommended for the coefficients of the key load model:

Coefficient	Value
oracle.hadoop.balancer.linearKeyLoad.keyWeight	93.981394
oracle.hadoop.balancer.linearKeyLoad.rowWeight	0.001126
oracle.hadoop.balancer.linearKeyLoad.byteWeight	0.000000

Model Messages

The following messages were created by the key load model analysis:

Model	Type	Message
byteRecordModel	info	Detected a perfect multicollinearity: records vs bytes. Dropping the bytes predictor from the model.

Reduce Tasks

Task ID	Time					Input									Output					%Load
	Start	Finish	Elapsed	CPU time	%	Shuffle Bytes count	%	Keys count	%	Records count	%	ValueBytes count	%	Records count	%	Bytes count	%		Observed
0	17:35:42	17:35:53	00:00:11	00:00:12	11	20,987,702	12	13	13	2,300,118	12	43,702,242	12	1,698,171	12	48,022,186	12		12
1	17:35:42	17:35:51	00:00:09	00:00:10	9	15,630,571	9	12	12	1,438,876	7	27,338,644	7	1,309,838	9	36,424,455	9		7
2	17:35:42	17:35:51	00:00:09	00:00:09	8	14,682,583	8	7	7	1,503,328	8	28,563,232	8	1,220,301	9	33,206,872	9		8
3	17:35:42	17:35:51	00:00:09	00:00:10	9	14,977,081	9	10	10	1,450,603	7	27,561,457	7	1,253,630	9	33,638,187	9		7
4	17:35:42	17:35:52	00:00:10	00:00:12	11	21,175,220	12	11	11	2,073,885	10	39,403,815	10	1,768,698	13	51,191,881	13		10
5	17:35:42	17:35:53	00:00:11	00:00:12	11	19,664,004	11	10	10	2,580,271	13	49,025,149	13	1,497,856	11	40,709,061	11		13
6	17:35:42	17:35:50	00:00:08	00:00:08	7	8,867,408	5	9	9	775,690	4	14,738,110	4	740,944	5	20,503,295	5		4
7	17:35:42	17:35:49	00:00:07	00:00:06	6	5,912,478	3	6	6	531,247	3	10,093,693	3	494,798	4	13,336,969	3		3
8	17:35:42	17:35:59	00:00:17	00:00:20	18	35,890,632	21	11	11	5,824,950	29	110,674,050	29	2,527,641	18	69,087,868	18		29
9	17:35:42	17:35:51	00:00:09	00:00:10	9	16,252,928	9	11	11	1,521,032	8	28,899,608	8	1,359,917	10	38,889,599	10		8

About Configuring Perfect Balance

Perfect Balance uses the standard Hadoop methods of specifying configuration properties in the command line. You can use the -conf option to identify a configuration file, or the -D option to specify individual properties. All Perfect Balance configuration properties have default values, and so setting them is optional.

"Perfect Balance Configuration Property Reference" on page 4-17 lists the configuration properties in alphabetical order with a full description. The following are functional groups of properties.

Perfect Balance Basic Properties

- oracle.hadoop.balancer.autoAnalyze
- oracle.hadoop.balancer.autoBalance

Job Analyzer Properties

- oracle.hadoop.balancer.tools.jobConfPath
- oracle.hadoop.balancer.tools.jobHistoryPath
- oracle.hadoop.balancer.tools.printRecommendation
- oracle.hadoop.balancer.tools.writeKeyBytes

Key Chopping Properties

- oracle.hadoop.balancer.enableSorting
- oracle.hadoop.balancer.keyLoad.minChopBytes

Load Balancing Properties

- oracle.hadoop.balancer.confidence
- oracle.hadoop.balancer.maxLoadFactor
- oracle.hadoop.balancer.maxSamplesPct
- oracle.hadoop.balancer.minSplits

Load Model Properties

- oracle.hadoop.balancer.linearKeyLoad.byteWeight
- oracle.hadoop.balancer.linearKeyLoad.keyWeight
- oracle.hadoop.balancer.linearKeyLoad.rowWeight

MapReduce-Related Properties

- oracle.hadoop.balancer.useMapreduceApi
- oracle.hadoop.balancer.inputFormat.mapred.map.tasks
- oracle.hadoop.balancer.inputFormat.mapred.max.split.size

Partition Report Properties

- oracle.hadoop.balancer.report.overwrite
- oracle.hadoop.balancer.reportPath
- oracle.hadoop.balancer.tmpDir

Sampler Properties

- oracle.hadoop.balancer.minSplits

- oracle.hadoop.balancer.numThreads
- oracle.hadoop.balancer.runMode
- oracle.hadoop.balancer.useClusterStats

Running a Balanced MapReduce Job Using Perfect Balance

Perfect Balance does not require you to make any changes to your application code. It works by automatically running Perfect Balance for your job when you submit it to Hadoop for execution.

To run a job with Perfect Balance:

1. Log in to the server where you will submit the job.

2. Set up Perfect Balance by following the steps in "Getting Started with Perfect Balance" on page 4-3.

3. Configure the job with these Perfect Balance properties:

   - To enable balancing, set oracle.hadoop.balancer.autoBalance to true. This setting also runs Job Analyzer. Load balancing is not enabled by default.

   - To allow Job Analyzer to collect additional metrics, set oracle.hadoop.balancer.autoAnalyze to REDUCER_REPORT.

     See "Collecting Additional Metrics" on page 4-7.

   - Decide which additional configuration properties to set, if any.

     See "About Configuring Perfect Balance" on page 4-9.

4. Run your job as usual, using the following syntax:

```
bin/hadoop jar application_jarfile.jar ApplicationClass\
-D application_config_property \
-D oracle.hadoop.balancer.autoBalance=true \
-D other_perfect_balance_config_property \
-conf application_config_file.xml \
-conf perfect_balance_config_file.xml
```

You do not need to make any code changes to your application. You can provide Perfect Balance configuration properties either on the command line or in a configuration file. You can also combine Perfect Balance properties and MapReduce properties in the same configuration file. "About Configuring Perfect Balance" on page 4-9.

Example 4–3 runs a script named pb_balance.sh, which sets up Perfect Balance for a job, and then runs the job. The key load metric properties are set to the values recommended in the Job Analyzer report shown in Figure 4–1.

Example 4–3   Running a Job Using Perfect Balance

```
$ cat pb_balance.sh

#setup perfect balance as described in Getting Started with Perfect Balance
BALANCER_HOME=/opt/oracle/orabalancer-1.10.0-h2
export HADOOP_CLASSPATH=${BALANCER_HOME}/jlib/orabalancerclient-1.1.1.jar:
${BALANCER_HOME}/jlib/orabalancer.jar:${BALANCER_
HOME}/jlib/commons-math-2.2.jar:$HADOOP_CLASSPATH
export HADOOP_USER_CLASSPATH_FIRST=true

# setup  optional properties like java heap size and garbage collector
```

```
export HADOOP_CLIENT_OPTS="-XX:+UseConcMarkSweepGC -Xmx512M ${HADOOP_CLIENT_OPTS}"

# run the job with balancing and job analyzer enabled
hadoop jar  application_jarfile.jarApplicationClass
 -D application_config_property \
 -D mapred.input.dir=jdoe_application/input  \
 -D mapred.output.dir=jdoe_outdir \
 -D mapred.job.name="autoinvoke" \
 -D mapred.reduce.tasks=10 \
 -D oracle.hadoop.balancer.autoBalance=true \
 -D oracle.hadoop.balancer.autoAnalyze=REDUCER_REPORT \
 -D oracle.hadoop.balancer.tools.printRecommendation=true \
 -D oracle.hadoop.balancer.linearKeyLoad.keyWeight=93.98 \
 -D oracle.hadoop.balancer.linearKeyLoad.rowWeight=0.001126 \
 -D oracle.hadoop.balancer.linearKeyLoad.byteWeight=0.0 \
 -conf application_config_file.xml
```

```
$ sh ./pb_balance.sh
13/10/23 17:47:48 INFO balancer.Balancer: Starting Balancer
13/10/23 17:47:49 INFO input.FileInputFormat: Total input paths to process : 5
13/10/23 17:47:54 INFO balancer.Balancer: Balancer completed
13/10/23 17:47:54 WARN mapred.JobClient: Use GenericOptionsParser for parsing the
arguments. Applications should implement Tool for the same.
13/10/23 17:47:54 INFO input.FileInputFormat: Total input paths to process : 5
13/10/23 17:47:54 INFO mapred.JobClient: Running job: job_201310212138_0170
13/10/23 17:47:55 INFO mapred.JobClient:  map 0% reduce 0%
13/10/23 17:48:09 INFO mapred.JobClient:  map 18% reduce 0%
13/10/23 17:48:10 INFO mapred.JobClient:  map 31% reduce 0%
    .
    .
    .
13/10/23 17:48:45 INFO mapred.JobClient:     Reduce input records=20000000
13/10/23 17:48:45 INFO mapred.JobClient:     Reduce output records=13871794
13/10/23 17:48:45 INFO mapred.JobClient:     Spilled Records=60000000
13/10/23 17:48:45 INFO mapred.JobClient:     CPU time spent (ms)=275560
13/10/23 17:48:45 INFO mapred.JobClient:     Physical memory (bytes)
snapshot=10480836608
13/10/23 17:48:45 INFO mapred.JobClient:     Virtual memory (bytes)
snapshot=34511245312
13/10/23 17:48:45 INFO mapred.JobClient:     Total committed heap usage
(bytes)=17533239296
```

About Perfect Balance Reports

Perfect Balance generates these reports when it runs a job:

- **Job Analyzer report**: Contains various indicators about the distribution of the load in a job. The report is saved in HTML for you, and XML for Perfect Balance to use. The report is always named jobanalyzer-report.html and -.xml. See "Reading the Job Analyzer Report" on page 4-7.

- **Partition report**: Identifies the keys that are assigned to the various mappers. This report is saved in XML for Perfect Balance to use; it does not contain information of use to you. The report is named ${mapred_output_dir}/_balancer/orabalancer_report.xml. It is only generated for balanced jobs.

- **Reduce key metric reports**: Perfect Balance generates a report for each file partition, when the appropriate configuration properties are set. The reports are saved in XML for Perfect Balance to use; they do not contain information of use to

you. They are named `${mapred_output_dir}/_balancer/ReduceKeyMetricList-attempt_`*`jobid_taskid_task_attemptid`*`.xml`. They are generated only when the counting reducer is used (that is, `oracle.hadoop.balancer.autoAnalyze=REDUCER_REPORT` when using Perfect Balance, or a call to the `Balancer.configureCountingReducer` method when using the API.

See "Collecting Additional Metrics" on page 4-7.

The reports are stored by default in the job output directory (`${mapred.output.dir}`). Following is the structure of that directory:

```
job_output_directory
   /_SUCCESS
   /_balancer
      ReduceKeyMetricList-attempt_201305031125_0016_r_000000_0.xml
      ReduceKeyMetricList-attempt_201305031125_0016_r_000001_0.xml
         .
         .
         .
   jobanalyzer-report.html
   jobanalyzer-report.xml
   orabalancer_report.xml
   /_logs
      /history
         job_201305031125_0016_1372277234720_jdoe_invindx_nobal
         localhost.localdomain_1367594758596_job_201305031125_0016_conf.xml
/part-r-00000
/part-r-00001
   .
   .
   .
```

About Chopping

To balance a load, Perfect Balance might subpartition the values of a single reduce key and send each subpartition to a different reducer. This partitioning of values is called **chopping**.

Selecting a Chopping Method

You can configure how Perfect Balance chops the values by setting the `oracle.hadoop.balancer.enableSorting` configuration parameter:

- **Chopping by hash partitioning**: Set `enableSorting=false` when sorting is not required. This is the default chopping strategy.

- **Chopping by sorting**: Set `enableSorting=true` to sort the values in each subpartition and order them across all subpartitions. In any parallel sort job, each task sort the rows within the task. The job must ensure that the values in reduce task 2 are greater than values in reduce task 1, the values in reduce task 3 are greater than the values in reduce task 2, and so on. The job generates multiple files containing data in sorted order, instead of one large file with sorted data.

  For example, if a key is chopped into three subpartitions, and the subpartitions are sent to reducers 5, 8 and 9, then the values for that key in reducer 9 are greater than all values for that key in reducer 8, and the values for that key in reducer 8 are greater than all values for that key in reducer 5. When `enableSorting=true`, Perfect Balance ensures this ordering across reduce tasks.

If an application requires that the data is aggregated across files, then you can disable chopping by setting `oracle.hadoop.balancer.keyLoad.minChopBytes`=-1. Perfect Balance still offers performance gains by combining smaller reduce keys, called **bin packing**.

How Chopping Impacts Applications

In general, when a MapReduce job aggregates the data by reduce key, each reduce task aggregates the rows within that task. This level of aggregation might be sufficient for the application, and might be preferred if further operations on the data are done in parallel. If further aggregation across reduce tasks is needed, then the application must perform the final aggregation after the MapReduce job.

For example, a typical search engine operating in parallel probably works well with output from the inverted index application, in which the data is aggregated within each file, but not fully aggregated across all files. See "About the Perfect Balance Examples" on page 4-16.

As another example, Oracle Loader for Hadoop loads data from multiple reduce tasks into the same partition of a target table. Oracle Loader for Hadoop is an example of an application that can handle chopping, because it does not require that all rows for a target database partition come from a single reducer.

Troubleshooting Jobs Running with Perfect Balance

Check these tips for help troubleshoot a job running with Perfect Balance.

Java GC Overhead Limit Exceeded Error

If you get a Java "GC overhead limit exceeded" error on the client node while running a job with Perfect Balance, then change the client JVM garbage collector for your job.

Set the Java JVM `-XX:+UseConcMarkSweepGC` option. You can specify client JVM options before running the Hadoop job, by setting the `HADOOP_CLIENT_OPTS` variable:

```
export HADOOP_CLIENT_OPTS="-XX:+UseConcMarkSweepGC $HADOOP_CLIENT_OPTS"
```

Setting `HADOOP_CLIENT_OPTS` changes the JVM options only on the client node. It does not change JVM options in the map and reduce tasks. See the invindx script for an example of setting this variable.

You can also increase the client JVM heap size. See "Java Out of Heap Space Errors" on page 4-13.

Java Out of Heap Space Errors

If you get Java "out of heap space" errors on the client node while running a job with Perfect Balance, then increase the client JVM heap size for the job.

Use the Java JVM `-Xmx` option. You can specify client JVM options before running the Hadoop job, by setting the `HADOOP_CLIENT_OPTS` variable:

```
$ export HADOOP_CLIENT_OPTS="-Xmx512M $HADOOP_CLIENT_OPTS"
```

Setting `HADOOP_CLIENT_OPTS` changes the JVM options only on the client node. It does not change JVM options in the map and reduce tasks. See the `invindx` script for an example of setting this variable.

Using the Perfect Balance API

The `oracle.hadoop.balancer.Balancer` class contains methods for creating a partitioning plan, saving the plan to a file, and running the MapReduce job using the plan. You only need to add the code to the application's job driver Java class, not redesign the application. When you run a shell script to run the application, you can include Perfect Balance configuration settings.

Modifying Your Java Code to Use Perfect Balance

The Perfect Balance installation directory contains a complete example, including input data, of a Java MapReduce program that uses the Perfect Balance API.

For a description of the inverted index example and execution instructions, see `orabalancer-1.1.1-h2/examples/invindx/README.txt`.

To explore the modified Java code, see `orabalancer-1.1.1-h2/examples/jsrc/oracle/hadoop/balancer/examples/invindx/InvertedIndexMapred.java` or `InvertedIndexMapreduce.java`.

The modifications to run Perfect Balance include the following:

- The `createBalancer` method validates the configuration properties and returns a `Balancer` instance.

- The `waitForCompletion` method samples the data and creates a partitioning plan.

- The `addBalancingPlan` method adds the partitioning plan to the job configuration settings.

- The `configureCountingReducer` method collects additional load statistics.

- The `save` method saves the partition report and generates the Job Analyzer report.

Example 4–4 shows fragments from the inverted index Java code.

Example 4–4   Running Perfect Balance in a MapReduce Job

```
   .
   .
   .
import oracle.hadoop.balancer.Balancer;
   .
   .
   .
///// BEGIN: CODE TO INVOKE BALANCER (PART-1, before job submission) //////
    Configuration conf = job.getConfiguration();

    Balancer balancer = null;

    boolean useBalancer =
        conf.getBoolean("oracle.hadoop.balancer.driver.balance", true);
    if(useBalancer)
    {
      balancer = Balancer.createBalancer(conf);
      balancer.waitForCompletion();
      balancer.addBalancingPlan(conf);
    }

    if(conf.getBoolean("oracle.hadoop.balancer.tools.useCountingReducer", true))
    {
      Balancer.configureCountingReducer(conf);
```

```
      }
      /////////////// END: CODE TO INVOKE BALANCER (PART-1) ////////////////////////

      boolean isSuccess = job.waitForCompletion(true);

      ///////////////////////////////////////////////////////////////////////////
      // BEGIN: CODE TO INVOKE BALANCER (PART-2, after job completion, optional)
      // If balancer ran, this saves the partition file report into the _balancer
      // sub-directory of the job output directory. It also writes a JobAnalyzer
      // report.
      Balancer.save(job);
      /////////////// END: CODE TO INVOKE BALANCER (PART-2) ////////////////////////
       .
       .
       .
}
```

> **See Also:**   *Oracle Big Data Appliance Perfect Balance Java API Reference*

Running Your Modified Java Code with Perfect Balance

When you run your modified Java code, you can set the Perfect Balance properties by using the standard `hadoop` command syntax:

```
bin/hadoop jar application_jarfile.jar ApplicationClass \
-conf application_config.xml \
-conf perfect_balance_config.xml \
-D application_config_property \
-D perfect_balance_config_property \
-libjars
/opt/oracle/orabalancer-1.1.1-h2/jlib/commons-math-2.2.jar,/opt/oracle/orabalancer
-1.1.1-h2/jlib/orabalancer-1.1.1.jar,application_jar_path.jar...
```

Example 4–5 runs a script named `pb_balanceapi.sh`, which runs the `InvertedIndexMapreduce` class example packaged in the Perfect Balance JAR file. The key load metric properties are set to the values recommended in the Job Analyzer report shown in Figure 4–1.

To run the InvertedIndexMapreduce class example, see "About the Perfect Balance Examples" on page 4-16.

Example 4–5   Running the InvertedIndexMapreduce Class

```
$ cat pb_balanceapi.sh
BALANCER_HOME=/opt/oracle/orabalancer-1.1.1-h2
APP_JAR_FILE=/opt/oracle/orabalancer-1.1.1-h2/jlib/orabalancer-1.1.1.jar
export HADOOP_CLASSPATH=${BALANCER_HOME}/jlib/orabalancer-1.1.1.jar:${BALANCER_
HOME}/jlib/commons-math-2.2.jar:$HADOOP_CLASSPATH
export HADOOP_USER_CLASSPATH_FIRST=true

hadoop jar ${APP_JAR_FILE}
oracle.hadoop.balancer.examples.invindx.InvertedIndexMapreduce \
 -D mapred.input.dir=invindx/input \
 -D mapred.output.dir=jdoe_outdir_api \
 -D mapred.job.name=jdoe_invindx_api \
 -D mapred.reduce.tasks=10 \
 -D oracle.hadoop.balancer.autoAnalyze=true \
 -D oracle.hadoop.balancer.tools.printRecommendation=true \
 -D oracle.hadoop.balancer.linearKeyLoad.keyWeight=93.981394 \
 -D oracle.hadoop.balancer.linearKeyLoad.rowWeight=0.001126 \
 -D oracle.hadoop.balancer.linearKeyLoad.byteWeight=0.0 \
```

```
                -libjars ${BALANCER_HOME}/jlib/commons-math-2.2.jar,${BALANCER_
                HOME}/jlib/orabalancer-1.1.1.jar

                $ sh ./balanceapi.sh
                13/06/24 17:21:47 INFO balancer.Balancer: Creating balancer
                13/06/24 17:21:48 INFO input.FileInputFormat: Total input paths to process : 5
                13/06/24 17:21:48 INFO balancer.Balancer: Starting Balancer
                13/06/24 17:21:54 INFO balancer.Balancer: Balancer completed
                     .
                     .
                     .
```

About the Perfect Balance Examples

The Perfect Balance installation files include a full set of examples that you can run immediately. The InvertedIndex example is a MapReduce application that creates an inverted index on an input set of text files. The inverted index maps words to the location of the words in the text files. The input data is included.

About the Examples in This Chapter

The InvertedIndex example provides the basis for all examples in this chapter. They use the same data set and run the same MapReduce application. The modifications to the InvertedIndex example simply highlight the steps you must perform in running your own applications with Perfect Balance.

If you want to run the examples in this chapter, or use them as the basis for running your own jobs, then make the following changes:

- If you are modifying the examples to run your own application, then add your application JAR files to HADOOP_CLASSPATH and -libjars.

- Ensure that the value of mapred.input.dir identifies the location of your data.

  The invindx/input directory contains the sample data for the InvertedIndex example. To use this data, you must first set it up. See "Extracting the Example Data Set" on page 4-16.

- Replace jdoe with your Hadoop user name.

- Set the -conf option to an existing configuration file.

  The jdoe_conf_invindx.xml file is a modification of the configuration file for the InvertedIndex examples. The modified file does not have performance optimizing settings. You can use the example configuration file as is or modify it. See /opt/oracle/orabalancer-1.1.1-h2/examples/invindx/conf_mapreduce.xml (or conf_mapred.xml).

- Review the configuration settings in the file and in the shell script to ensure they are appropriate for your job.

- You can run the browser from your laptop or connect to Oracle Big Data Appliance using a client that supports graphical interfaces, such as VNC.

Extracting the Example Data Set

To run the InvertedIndex examples or any of the examples in this chapter, you must first set up the data files.

To extract the InvertedIndex data files:

1. Log in to a server where Perfect Balance is installed.

2. Change to the `examples/invindx` subdirectory:

   ```
   cd /opt/oracle/orabalancer-1.1.1-h2/examples/invindx
   ```

3. Unzip the data and copy it to the HDFS `invindx/input` directory:

   ```
   ./invindx -setup
   ```

For complete instructions for running the InvertedIndex example, see
`/opt/oracle/orabalancer-1.1.1-h2/examples/invindx/README.txt`.

Perfect Balance Configuration Property Reference

This section describes the Perfect Balance configuration properties and a few generic
Hadoop MapReduce properties that Perfect Balance reads from the job configuration:

- MapReduce Configuration Properties
- Perfect Balance Configuration Properties

See "About Configuring Perfect Balance" on page 4-9 for a list of the properties
organized into functional categories.

MapReduce Configuration Properties

mapred.input.dir
Type: String

Default Value: Not defined

Description: A comma-separated list of input directories.

mapred.input.format.class
Type: String

Default Value: `org.apache.hadoop.mapred.TextInputFormat`

Description: The full name of the `InputFormat` class.

mapred.mapper.class
Type: String

Default Value: `org.apache.hadoop.mapred.lib.IdentityMapper`

Description: The full name of the mapper class.

mapred.output.dir
Type: String

Default Value: Not defined

Description: The job output directory.

mapred.partitioner.class
Type: String

Default Value: `org.apache.hadoop.mapred.lib.HashPartitioner`

Description: The full name of the partitioner class.

mapred.reducer.class
Type: String

Default Value: `org.apache.hadoop.mapred.lib.IdentityReducer`

Description: The full name of the reducer class.

mapreduce.inputformat.class
Type: String

Default Value: `org.apache.hadoop.mapreduce.lib.input.TextInputFormat`

Description: The full name of the `InputFormat` class.

mapreduce.map.class
Type: String

Default Value: `org.apache.hadoop.mapreduce.Mapper`

Description: The full name of the mapper class.

mapreduce.partitioner.class
Type: String

Default Value: `org.apache.hadoop.mapreduce.lib.partition.HashPartitioner`

Description: The full name of the partitioner class.

mapreduce.reduce.class
Type: String

Default Value: `org.apache.hadoop.mapreduce.Reducer`

Description: The full name of the reducer class.

Perfect Balance Configuration Properties

oracle.hadoop.balancer.autoAnalyze
Type: Enumeration

Default Value: `BASIC_REPORT` if `oracle.hadoop.balancer.autoBalance` is true; otherwise `NONE`

Description: Controls the behavior of the Job Analyzer when it is called using Perfect Balance. The following values are valid:

- `NONE`: Disables Job Analyzer.
- `BASIC_REPORT`: Enables Job Analyzer
- `REDUCER_REPORT`: Enables Job Analyzer such that it collects additional load statistics for each reduce task in a job. See "Collecting Additional Metrics" on page 4-7..

Perfect Balance uses this property; the Perfect BalanceAPI does not.

oracle.hadoop.balancer.autoBalance
Type: Boolean

Default Value: `false`

Description: Controls whether load balancing is enabled. Set to `false` to turn off balancing. Perfect Balance uses this property; the Perfect Balance API does not.

oracle.hadoop.balancer.confidence
Type: Float

Default Value: `0.95`

Description: The statistical confidence indicator for the load factor specified by the `oracle.hadoop.balancer.maxLoadFactor` property.

This property accepts values greater than or equal to 0.5 and less than 1.0 (0.5 <= *value* < 1.0). A value less than 0.5 resets the property to its default value. Oracle recommends a value greater than or equal to 0.9. Typical values are 0.95 and 0.99.

oracle.hadoop.balancer.enableSorting
Type: Boolean

Default Value: `false`

Description: Controls how the map output keys are **chopped**, that is, split into smaller keys:

- `false`: Uses a hash function.

- `true`: Uses the map output key sorting comparator as a total-order partitioning function. The balancer preserves the total order over the values of the chopped keys.

Perfect Balance sets this property to `false` if the job is not configured for secondary sorting.

oracle.hadoop.balancer.inputFormat.mapred.map.tasks
Type: Integer

Default Value: `100`

Description: Sets the Hadoop `mapred.map.tasks` property for the duration of sampling, just before calling the input format `getSplits` method. It does not change `mapred.map.tasks` for the actual job. The optimal number of map tasks is a trade-off between obtaining a good sample (larger number) and having finite memory resources (smaller number).

Set this property to a value greater than or equal to one (1). A value less than 1 disables the property.

Some input formats, such as `DBInputFormat`, use this property as a hint to determine the number of splits returned by `getSplits`. Higher values indicate that more chunks of data are sampled at random, which improves the sample.

You can increase the value for larger data sets, that is, more than a million rows of about 100 bytes per row. However, extremely large values can cause the input format's `getSplits` method to run out of memory by returning too many splits.

oracle.hadoop.balancer.inputFormat.mapred.max.split.size
Type: Long

Default Value: `1048576` (1 MB)

Description: Sets the Hadoop `mapred.max.split.size` property for the duration of sampling, just before calling the input format's `getSplits` method. It does not change `mapred.max.split.size` for the actual job.

Set this property to a value greater than or equal to one (1). A value less than 1 disables the property. The optimal split size is a trade-off between obtaining a good sample (smaller splits) and efficient I/O performance (larger splits).

Some input formats, such as `FileInputFormat`, use the maximum split size as a hint to determine the number of splits returned by `getSplits`. Smaller split sizes indicate that more chunks of data are sampled at random, which improves the sample. Set the value

small enough for good sampling performance, but no smaller. Extremely small values can cause inefficient I/O performance, while not improving the sample.

You can increase the value for larger data sets (tens of terabytes) or if the input format's `getSplits` method throws an out of memory error. Large splits are better for I/O performance, but not for sampling.

oracle.hadoop.balancer.keyLoad.minChopBytes
Type: Long

Default Value: 0

Description: Controls whether Perfect Balance chops large map output keys into medium keys:

- `-1`: Perfect Balance does not chop large map output keys.

- `0`: Perfect Balance chops large map output keys and determines the optimal size of each medium key.

- *Positive integer*: Perfect Balance chops large map output keys into medium keys with a size greater than or equal to the specified integer.

oracle.hadoop.balancer.linearKeyLoad.byteWeight
Type: Float

Default Value: 0.05

Description: Weights the number of bytes per key in the linear key load model specified by the `oracle.hadoop.balancer.KeyLoadLinear` class.

oracle.hadoop.balancer.linearKeyLoad.keyWeight
Type: Float

Default Value: 50.0

Description: Weights the number of medium keys per large key in the linear key load model specified by the `oracle.hadoop.balancer.KeyLoadLinear` class.

oracle.hadoop.balancer.linearKeyLoad.rowWeight
Type: Float

Default Value: 0.05

Description: Weights the number of rows per key in the linear key load model specified by the `oracle.hadoop.balancer.KeyLoadLinear` class.

oracle.hadoop.balancer.maxLoadFactor
Type: Float

Default Value: 0.05

Description: The target reducer load factor that you want the balancer's partition plan to achieve.

The load factor is the relative deviation from an estimated value. For example, if `maxLoadFactor=0.05` and `confidence=0.95`, then with a confidence greater than 95%, the job's reducer loads should be, at most, 5% greater than the value in the partition plan.

The values of these two properties determine the sampler's stopping condition. The balancer samples until it can generate a plan that guarantees the specified load factor at the specified confidence level. This guarantee may not hold if the sampler stops early because of other stopping conditions, such as the number of samples exceeds

oracle.hadoop.balancer.maxSamplesPct. The partition report logs the stopping condition.

See oracle.hadoop.balancer.confidence.

oracle.hadoop.balancer.maxSamplesPct
Type: Float

Default Value: 0.01 (1%)

Description: Limits the number of samples that Perfect Balance can collect to a fraction of the total input records. A value less than zero disables the property (no limit).

You may need to increase the value for Hadoop applications with very unbalanced reducer partitions or densely clustered map-output keys. The sampler needs to sample more data to achieve a good partitioning plan in these cases.

See oracle.hadoop.balancer.useClusterStats.

oracle.hadoop.balancer.minSplits
Type: Integer

Default Value: 5

Description: Sets the minimum number of splits that the sampler reads. If the total number of splits is less than this value, then the sampler reads all splits. Set this property to a value greater than or equal to one (1). A nonpositive number sets the property to 1.

oracle.hadoop.balancer.numThreads
Type: Integer

Default Value: 5

Description: Number of sampler threads. Set this value based on the processor and memory resources available on the node where the job is initiated. A higher number of sampler threads implies higher concurrency in sampling. Set this property to one (1) to disable multithreading in the sampler.

oracle.hadoop.balancer.report.overwrite
Type: Boolean

Default Value: false

Description: Controls whether Perfect Balance overwrites files in the location specified by the oracle.hadoop.balancer.reportPath property. By default, Perfect Balance does not overwrite files; it throws an exception. Set this property to true to allow partition reports to be overwritten.

oracle.hadoop.balancer.reportPath
Type: String

Default Value: *directory*/orabalancer_report-*random_unique_string*.xml, where directory for HDFS is the home directory of the user who submits the job. For the local file system, it is the directory where the job is submitted.

Description: The path where Perfect Balance writes the partition report before the Hadoop job output directory is available, that is, before the MapReduce job finishes running. At the end of the job, Perfect Balance moves the file to *job_output_dir*/_balancer/orabalancer_report.xml. In the API, the save method does this task.

oracle.hadoop.balancer.runMode
Type: String

Default Value: `local`

Description: Specifies how to run the Perfect Balance sampler. The following values are valid:

- `local`: The sampler runs on the client node where the job is submitted.

- `distributed`: The sampler runs as a Hadoop job. If the job uses DistributedCache symlinks, then Perfect Balance automatically sets this property to `distributed`.

If this property is set to an invalid string, Perfect Balance resets it to `local`.

oracle.hadoop.balancer.tmpDir
Type: String

Default Value: /tmp/orabalancer-*user_name*

Description: The path to a staging directory in the file system of the job output directory (HDFS or local). Perfect Balance creates the directory if it does not exist, and copies the partition report to it for loading into the Hadoop distributed cache.

oracle.hadoop.balancer.tools.jobConfPath
Type: String

Default Value: `${mapred.output.dir}`/_logs/history

Description: The path to a Hadoop job configuration file. Job Analyzer uses this setting to locate the file.

oracle.hadoop.balancer.tools.jobHistoryPath
Type: String

Default Value: `${mapred.output.dir}`/_logs/history

Description: The path to a Hadoop job history file. Job Analyzer uses this setting to locate the file.

oracle.hadoop.balancer.tools.printRecommendation
Type: Boolean

Default Value: `false`

Description: Controls whether Job Analyzer recommends values for the key load model properties, based on the elapsed time, input record, and input value byte statistics it gathers for each key. Job Analyzer does not print a recommendation in its report if it cannot make a confident recommendation. You can set the load model properties to the recommended values when running a balanced job. See "Load Model Properties" on page 4-9.

oracle.hadoop.balancer.tools.writeKeyBytes
Type: Boolean

Default Value: `false`

Description: Controls whether the counting reducer collects the byte representations of the reduce keys for the Job Analyzer. Set this property to `true` to represent the unique key values in Base64 encoding in the report. A string representation of the key, created using `key.toString`, is also provided in the report. This string value may not be unique for each key.

oracle.hadoop.balancer.useClusterStats
Type: Boolean

Default Value: `true`

Description: Enables the sampler to use cluster sampling statistics. These statistics improve the accuracy of sampled estimates, such as the number of records in a map-output key, when the map-output keys are distributed in clusters across input splits, instead of being distributed independently across all input splits.

Set this property to `false` only if you are absolutely certain that the map-output keys are not clustered. This setting improves the sampler's estimates only when there is, in fact, no clustering. Oracle recommends leaving this property set to `true`, because the distribution of map-output keys is usually unknown.

oracle.hadoop.balancer.useMapreduceApi
Type: Boolean

Default Value: `true`

Description: Identifies the MapReduce API used in the Hadoop job:

- `true`: The job uses the `mapreduce` API.
- `false`: The job uses the `mapred` API.

5

Configuring Oracle Exadata Database Machine for Use with Oracle Big Data Appliance

This chapter provides information about optimizing communications between Oracle Exadata Database Machine and Oracle Big Data Appliance. It describes how you can configure Oracle Exadata Database Machine to use InfiniBand alone, or SDP over InfiniBand, to communicate with Oracle Big Data Appliance.

This chapter contains the following sections:

- About Optimizing Communications
- Prerequisites
- Specifying the InfiniBand Connections to Oracle Big Data Appliance
- Specifying the InfiniBand Connections to Oracle Exadata Database Machine
- Enabling SDP on Exadata Database Nodes
- Configuring a JDBC Client for SDP
- Creating an SDP Listener on the InfiniBand Network

About Optimizing Communications

Oracle Exadata Database Machine and Oracle Big Data Appliance use Ethernet by default, although typically they are also connected by an InfiniBand network. Ethernet communications are much slower than InfiniBand. After you configure Oracle Exadata Database Machine to communicate using InfiniBand, it can obtain data from Oracle Big Data Appliance many times faster than before.

Moreover, client applications that run on Oracle Big Data Appliance and push the data to Oracle Database can use Sockets Direct Protocol (SDP) for an additional performance boost. SDP is a standard communication protocol for clustered server environments, providing an interface between the network interface card and the application. By using SDP, applications place most of the messaging burden upon the network interface card, which frees the CPU for other tasks. As a result, SDP decreases network latency and CPU utilization, and thereby improves performance.

About Applications that Pull Data Into Oracle Exadata Database Machine

Oracle SQL Connector for Hadoop Distributed File System (HDFS) is an example of an application that pulls data into Oracle Exadata Database Machine. The connector enables an Oracle external table to access data stored in either HDFS files or a Hive table.

The external table provide access to the HDFS data. You can use the external table for querying HDFS data or for loading it into an Oracle database table.

Oracle SQL Connector for HDFS functions as a Hadoop client running on the database servers in Oracle Exadata Database Machine.

If you use Oracle SQL Connector for HDFS or another tool that pulls the data into Oracle Exadata Database Machine, then for the best performance, you should configure the system to use InfiniBand. See "Specifying the InfiniBand Connections to Oracle Big Data Appliance" on page 5-2.

> **See Also :** *Oracle Big Data Connectors User's Guide* for information about Oracle SQL Connector for HDFS

About Applications that Push Data Into Oracle Exadata Database Machine

Oracle Loader for Hadoop is an example of an application that pushes data into Oracle Exadata Database Machine. The connector is an efficient and high-performance loader for fast movement of data from a Hadoop cluster into a table in an Oracle database. You can use it to load data from Oracle Big Data Appliance to Oracle Exadata Database Machine.

Oracle Loader for Hadoop functions as a database client running on the Oracle Big Data Appliance. It must make database connections from Oracle Big Data Appliance to Oracle Exadata Database Machine over the InfiniBand network. Use of Sockets Direct Protocol (SDP) for these database connections further improves performance.

If you use Oracle Loader for Hadoop or another tool that pushes the data into Oracle Exadata Database Machine, then for the best performance, you should configure the system to use SDP over InfiniBand as described in this chapter.

> **See Also :** *Oracle Big Data Connectors User's Guide* for information about Oracle Loader for Hadoop

Prerequisites

Oracle Big Data Appliance and Oracle Exadata Database Machine racks must be cabled together using InfiniBand cables. The IP addresses must be unique across all racks and use the same subnet for the InfiniBand network.

> **See Also:**
>
> - *Oracle Big Data Appliance Owner's Guide* about multirack cabling
> - *Oracle Big Data Appliance Owner's Guide* about IP addresses and subnets

Specifying the InfiniBand Connections to Oracle Big Data Appliance

You can configure Oracle Exadata Database Machine to use the InfiniBand IP addresses of the Oracle Big Data Appliance servers. Otherwise, the default network is Ethernet. Use of the InfiniBand network improves the performance of all data transfers between Oracle Big Data Appliance and Oracle Exadata Database Machine.

To identify the Oracle Big Data Appliance InfiniBand IP addresses:

1. If you have not done so already, install a CDH client on Oracle Exadata Database Machine. See "Providing Remote Client Access to CDH" on page 3-2.

2. Obtain a list of private host names and InfiniBand IP addresses for all Oracle Big Data Appliance servers.

An Oracle Big Data Appliance rack can have 6, 12, or 18 servers.

3. Log in to Oracle Exadata Database Machine with `root` privileges.

4. Edit `/etc/hosts` on Oracle Exadata Database Machine and add the Oracle Big Data Appliance host names and InfiniBand IP addresses. The following example shows the sequential IP numbering:

```
192.168.8.1        bda1node01.example.com      bda1node01
192.168.8.2        bda1node02.example.com      bda1node02
192.168.8.3        bda1node03.example.com      bda1node03
192.168.8.4        bda1node04.example.com      bda1node04
192.168.8.5        bda1node05.example.com      bda1node05
192.168.8.6        bda1node06.example.com      bda1node06
```

5. Check `/etc/nsswitch.conf` for a line like the following:

```
hosts:       files dns
```

Ensure that the line does not reverse the order (`dns files`); if it does, your additions to `/etc/hosts` will not be used. Edit the file if necessary.

6. Ping all Oracle Big Data Appliance servers. Ensure that `ping` completes and shows the InfiniBand IP addresses.

```
# ping bda1node01.example.com
PING bda1node01.example.com (192.168.8.1) 56(84) bytes of data.
64 bytes from bda1node01.example.com (192.168.8.1): icmp_seq=1 ttl=50 time=20.2
ms
     .
     .
     .
```

7. Run CDH locally on Oracle Exadata Database Machine and test HDFS functionality by uploading a large file to an Oracle Big Data Appliance server. Check that your network monitoring tools (such as `sar`) show I/O activity on the InfiniBand devices.

To upload a file, use syntax like the following, which copies localfile.dat to the HDFS testdir directory on node05 of Oracle Big Data Appliance:

```
hadoop fs -put localfile.dat hdfs://bda1node05.example.com/testdir/
```

Specifying the InfiniBand Connections to Oracle Exadata Database Machine

You can also configure Oracle Big Data Appliance to use the InfiniBand IP addresses of the Oracle Exadata Database Machine servers. This configuration supports applications on Oracle Big Data Appliance that must connect to Oracle Exadata Database Machine.

To identify the Oracle Exadata Database Machine InfiniBand IP addresses:

1. Obtain a list of private host names and InfiniBand IP addresses for all Oracle Exadata Database Machine servers.

2. Log in to Oracle Big Data Appliance with `root` privileges.

3. Edit `/etc/hosts` on Oracle Big Data Appliance and add the Oracle Exadata Database Machine host names and InfiniBand IP addresses.

4. Check `/etc/nsswitch.conf` for a line like the following:

   ```
   hosts:      files dns
   ```

   Ensure that the line does not reverse the order (`dns files`); if it does, your additions to /etc/hosts will not be used. Edit the file if necessary.

5. Restart the `dnsmasq` service:

   ```
   # service dnsmasq restart
   ```

6. Ping all Oracle Exadata Database Machine servers. Ensure that `ping` completes and shows the InfiniBand IP addresses.

7. Test the connection by downloading a large file to an Oracle Exadata Database Machine server. Check that your network monitoring tools (such as `sar`) show I/O activity on the InfiniBand devices.

   To download a file, use syntax like the following, which copies a file named `mydata.json` to the `dm01ce108` storage server:

   ```
   $ scp mydata.json oracle@dm01cel08-priv.example.com:mybigdata.json
   oracle@dm01cel08-priv.example.com's password: password
   ```

Enabling SDP on Exadata Database Nodes

The following procedure describes how to enable SDP on database nodes in an Oracle Exadata Database Machine running Oracle Linux. SDP improves the performance of client applications that run on Oracle Big Data Appliance and push data to Oracle Exadata Database Machine.

To enable SDP on Oracle Exadata Database Machine:

1. Open `/etc/infiniband/openib.conf` file in a text editor, and add the following line:

   ```
   set: SDP_LOAD=yes
   ```

2. Save these changes and close the file.

3. To enable both SDP and TCP, open `/etc/ofed/libsdp.conf` in a text editor, and add the `use both` rule:

   ```
   use both server * :
   use both client * :
   ```

4. Save these changes and close the file.

5. Open `/etc/modprobe.conf` file in a text editor, and add this setting:

   ```
   options ib_sdp sdp_zcopy_thresh=0 recv_poll=0
   ```

6. Save these changes and close the file.

7. Replicate these changes across all database nodes in the Oracle Exadata Database Machine rack.

8. Restart all database nodes for the changes to take effect.

9. If you have multiple Oracle Exadata Database Machine racks, then repeat these steps on all of them.

Configuring a JDBC Client for SDP

The following procedure explains how to configure a JDBC client to use SDP.

To enable SDP support for JDBC:

1. Configure the database to support InfiniBand, as described in the *Oracle Database Net Services Administrator's Guide*. Ensure that you set the protocol to SDP.

2. Set the LD_PRELOAD environment variable to libsdp.so before starting the Java virtual machine. This example uses the Bash shell:

   ```
   export LD_PRELOAD="libsdp.so"
   ```

Creating an SDP Listener on the InfiniBand Network

To add a listener for the Oracle Big Data Appliance connections coming in on the InfiniBand network, first add a network resource for the InfiniBand network with virtual IP addresses.

> **Note:** This example lists two nodes for an Oracle Exadata Database Machine quarter rack. If you have an Oracle Exadata Database Machine half or full rack, you must repeat node-specific lines for each node in the cluster.

1. Edit /etc/hosts on each node in the Exadata rack to add the virtual IP addresses for the InfiniBand network. Make sure that these IP addresses are not in use. For example:

   ```
   # Added for Listener over IB
   192.168.10.21 dm01db01-ibvip.example.com dm01db01-ibvip
   192.168.10.22 dm01db02-ibvip.example.com dm01db02-ibvip
   ```

2. As the root user, create a network resource on one database node for the InfiniBand network. For example:

   ```
   # /u01/app/grid/product/11.2.0.2/bin/srvctl add network -k 2 -S
   192.168.10.0/255.255.255.0/bondib0
   ```

3. Verify that the network was added correctly with a command like the following examples:

   ```
   # /u01/app/grid/product/11.2.0.2/bin/crsctl stat res -t | grep net
   ora.net1.network
   ora.net2.network -- Output indicating new Network resource
   ```

   or

   ```
   # /u01/app/grid/product/11.2.0.2/bin/srvctl config network -k 2
   Network exists: 2/192.168.10.0/255.255.255.0/bondib0, type static -- Output
   indicating Network resource on the 192.168.10.0 subnet
   ```

4. Add the virtual IP addresses on the network created in Step 2, for each node in the cluster. For example:

   ```
   # srvctl add vip -n dm01db01 -A dm01db01-ibvip/255.255.255.0/bondib0 -k 2
   #
   # srvctl add vip -n dm01db02 -A dm01db02-ibvip/255.255.255.0/bondib0 -k 2
   ```

5. As the `oracle` user who owns Grid Infrastructure Home, add a listener for the virtual IP addresses created in Step 4.

```
# srvctl add listener -l LISTENER_IB -k 2 -p TCP:1522,/SDP:1522
```

6. For each database that will accept connections from the middle tier, modify the `listener_networks init` parameter to allow load balancing and failover across multiple networks (Ethernet and InfiniBand). You can either enter the full `TNSNAMES` syntax in the initialization parameter or create entries in `tnsnames.ora` in the `$ORACLE_HOME/network/admin` directory. The `TNSNAMES.ORA` entries must exist in `GRID_HOME`. The following example first updates `tnsnames.ora`.

Complete this step on each node in the cluster with the correct IP addresses for that node. `LISTENER_IBREMOTE` should list all other nodes that are in the cluster. `DBM_IB` should list all nodes in the cluster.

> **Note:** The database instance reads the `TNSNAMES` only on startup. Thus, if you modify an entry that is referred to by any `init.ora` parameter (`LISTENER_NETWORKS`), then you must either restart the instance or issue an `ALTER SYSTEM SET LISTENER_NETWORKS` command for the modifications to take affect by the instance.

```
DBM =
(DESCRIPTION =
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01-scan)(PORT = 1521))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = dbm)
))

DBM_IB =
(DESCRIPTION =
(LOAD_BALANCE=on)
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01db01-ibvip)(PORT = 1522))
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01db02-ibvip)(PORT = 1522))
(CONNECT_DATA =
(SERVER = DEDICATED)
(SERVICE_NAME = dbm)
))

LISTENER_IBREMOTE =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01db02-ibvip.mycompany.com)(PORT = 1522))
))

LISTENER_IBLOCAL =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01db01-ibvip.mycompany.com)(PORT = 1522))
(ADDRESS = (PROTOCOL = SDP)(HOST = dm01db01-ibvip.mycompany.com)(PORT = 1523))
))

LISTENER_IPLOCAL =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = dm0101-vip.mycompany.com)(PORT = 1521))
))
```

```
LISTENER_IPREMOTE =
(DESCRIPTION =
(ADDRESS_LIST =
(ADDRESS = (PROTOCOL = TCP)(HOST = dm01-scan.mycompany.com)(PORT = 1521))
))
```

7. Connect to the database instance as `sysdba`.

8. Modify the `listener_networks init` parameter by using the SQL `ALTER SYSTEM` command:

```
SQL> alter system set listener_networks=
     '((NAME=network2) (LOCAL_LISTENER=LISTENER_IBLOCAL)
       (REMOTE_LISTENER=LISTENER_IBREMOTE))',
     '((NAME=network1)(LOCAL_LISTENER=LISTENER_IPLOCAL)
       (REMOTE_LISTENER=LISTENER_IPREMOTE))' scope=both;
```

9. On the Linux command line, use the `srvctl` command to restart `LISTENER_IB` to implement the modification in Step 7:

```
# srvctl stop listener -l LISTENER_IB
# srvctl start listener -l LISTENER_IB
```

Glossary

ASR

Oracle Auto Service Request, a software tool that monitors the health of the hardware and automatically generates a service request if it detects a problem.

See also **OASM**.

BDR

See **Cloudera Backup and Disaster Recovery**.

Balancer

A service that ensures that all nodes in the cluster store about the same amount of data, within a set range. Data is balanced over the nodes in the cluster, not over the disks in a node.

CDH

Cloudera's Distribution including Apache Hadoop, the version of Apache Hadoop and related components installed on Oracle Big Data Appliance.

Cloudera Backup and Disaster Recovery

Replicates data stored in HDFS and accessed through Hive to ensure that the data is always available, even if the data center completely shuts down.

Cloudera Hue

Hadoop User Experience, a web user interface in CDH that includes several applications, including a file browser for HDFS, a job browser, an account management tool, a MapReduce job designer, and Hive wizards. Cloudera Manager runs on Hue.

See also **HDFS**; **Hive**.

Cloudera Impala

A massively parallel processing query engine that delivers better performance for SQL queries against data in HDFS and HBase, without moving or transforming the data.

Cloudera Navigator

Verifies access privileges and audits access to data stored in Hadoop, including Hive metadata and HDFS data accessed through HDFS, Hive, or HBase.

Cloudera Search

Provides search and navigation tools for data stored in Hadoop.

Cloudera's Distribution including Apache Hadoop (CDH)

See **CDH**.

cluster

A group of servers on a network that are configured to work together. A server is either a master node or a worker node.

All servers in an Oracle Big Data Appliance rack form a cluster. Servers 1, 2, and 3 are master nodes. Servers 4 to 18 are worker nodes.

See **Hadoop**.

DataNode

A server in a CDH cluster that stores data in HDFS. A DataNode performs file system operations assigned by the NameNode.

See also **HDFS**; **NameNode**.

Flume NG

A distributed service in CDH for collecting and aggregating data from almost any source into a data store such as HDFS or HBase. NG refers to "next generation," and identifies a major revision in the architecture from the original implementation.

See also **HBase**; **HDFS**.

Hadoop

A batch processing infrastructure that stores files and distributes work across a group of servers. Oracle Big Data Appliance uses Cloudera's Distribution including Apache Hadoop (CDH).

Hadoop Distributed File System (HDFS)

See **HDFS**.

Hadoop User Experience (Hue)

See **Cloudera Hue**.

HBase

An open-source, column-oriented database that provides random, read/write access to large amounts of sparse data stored in a CDH cluster. It provides fast lookup of values by key and can perform thousands of insert, update, and delete operations per second.

HDFS

Hadoop Distributed File System, an open-source file system designed to store extremely large data files (megabytes to petabytes) with streaming data access patterns. HDFS splits these files into data blocks and distributes the blocks across a CDH cluster.

When a data set is larger than the storage capacity of a single computer, then it must be partitioned across several computers. A distributed file system can manage the storage of a data set across a network of computers.

See also **cluster**.

Hive

An open-source data warehouse in CDH that supports data summarization, ad hoc querying, and data analysis of data stored in HDFS. It uses a SQL-like language called HiveQL. An interpreter generates MapReduce code from the HiveQL queries.

By using Hive, you can avoid writing MapReduce programs in Java.

See also **Hive Thrift**; **HiveQL**; **MapReduce**.

Hive Thrift

A remote procedure call (RPC) interface for remote access to CDH for Hive queries.

See also **CDH**; **Hive**.

HiveQL

A SQL-like query language used by Hive.

See also **Hive**.

HotSpot

A Java Virtual Machine (JVM) that is maintained and distributed by Oracle. It automatically optimizes code that executes frequently, leading to high performance. HotSpot is the standard JVM for the other components of the Oracle Big Data Appliance stack.

Hue

See **Cloudera Hue**.

Impala

See **Cloudera Impala**.

Java HotSpot Virtual Machine

See **HotSpot**.

JobTracker

A service that assigns MapReduce tasks to specific nodes in the CDH cluster, preferably those nodes storing the data.

See also **Hadoop**; **MapReduce**.

Kerberos

A network authentication protocol that helps prevent malicious impersonation. It was developed at the Massachusetts Institute of Technology (MIT).

Mahout

Apache Mahout is a machine learning library that includes core algorithms for clustering, classification, and batch-based collaborative filtering.

MapReduce

A parallel programming model for processing data on a distributed system.

A MapReduce program contains these functions:

- Mappers: Process the records of the data set.
- Reducers: Merge the output from several mappers.

■ Combiners: Optimizes the result sets from the mappers before sending them to the reducers (optional).

MySQL Database

A SQL-based relational database management system. Cloudera Manager, Oracle Data Integrator, Hive, and Oozie use MySQL Database as a metadata repository on Oracle Big Data Appliance.

NameNode

A service that maintains a directory of all files in HDFS and tracks where data is stored in the CDH cluster.

See also **HDFS**.

Navigator

See **Cloudera Navigator**.

node

A server in a CDH cluster.

See also **cluster**.

NoSQL Database

See **Oracle NoSQL Database**.

OASM

Oracle Automated Service Manager, a service for monitoring the health of Oracle Sun hardware systems. Formerly named Sun Automatic Service Manager (SASM).

Oozie

An open-source workflow and coordination service for managing data processing jobs in CDH.

Oracle Database Instant Client

A small-footprint client that enables Oracle applications to run without a standard Oracle Database client.

Oracle Linux

An open-source operating system. Oracle Linux 5.6 is the same version used by Exalogic 1.1. It features the Oracle Unbreakable Enterprise Kernel.

Oracle NoSQL Database

A distributed key-value database that supports fast querying of the data, typically by key lookup.

Oracle R Distribution

An Oracle-supported distribution of the R open-source language and environment for statistical analysis and graphing.

Oracle R Enterprise

A component of the Oracle Advanced Analytics Option. It enables R users to run R commands and scripts for statistical and graphical analyses on data stored in an Oracle database.

Pig

An open-source platform for analyzing large data sets that consists of the following:

- Pig Latin scripting language

- Pig interpreter that converts Pig Latin scripts into MapReduce jobs

Pig runs as a client application.

See also **MapReduce**.

Puppet

A configuration management tool for deploying and configuring software components across a cluster. The Oracle Big Data Appliance initial software installation uses Puppet.

The Puppet tool consists of these components: puppet agents, typically just called puppets; the puppet master server; a console; and a cloud provisioner.

See also **puppet agent**; **puppet master**.

puppet agent

A service that primarily pulls configurations from the puppet master and applies them. Puppet agents run on every server in Oracle Big Data Appliance.

See also **Puppet**; **puppet master**

puppet master

A service that primarily serves configurations to the puppet agents.

See also **Puppet**; **puppet agent**.

Search

See **Cloudera Search**.

Sentry

Integrates with the Hive and Impala SQL-query engines to provide fine-grained authorization to data stored in Hadoop.

Sqoop

A command-line tool that imports and exports data between HDFS or Hive and structured databases. The name Sqoop comes from "SQL to Hadoop." Oracle R Advanced Analytics for Hadoop uses the Sqoop executable to move data between HDFS and Oracle Database.

table

In Hive, all files in a directory stored in HDFS.

See also **HDFS**.

TaskTracker

A service that runs on each node and executes the tasks assigned to it by the JobTracker service.

See also **JobTracker**.

Whirr

Apache Whirr is a set of libraries for running cloud services.

ZooKeeper

A centralized coordination service for CDH distributed processes that maintains
configuration information and naming, and provides distributed synchronization and
group services.

Index

ssh service, 2-23
svctag user, 2-22

T

tables, 1-8, 3-9
Task Tracker Status interface, 2-6
TaskTracker
    monitoring, 2-6
    user identity, 2-22
tmpDir configuration property, 4-22
tools.* configuration properties, 4-22
trash facility, 3-11
trash facility, disabling, 3-12
trash interval, 3-11
troubleshooting CDH, 2-26

U

uploading diagnostics, 2-26
useClusterStats configuration property, 4-22
useMapreduceApi configuration property, 4-23
user accounts, 3-9
user groups, 3-9
users
    Cloudera Manager, 2-4
    operating system, 2-22

W

Whirr, 2-14
writeKeyBytes configuration property, 4-22

X

xinetd service, 2-23
XQuery connector
    See Oracle XQuery for Hadoop

Y

YARN support, 1-7

Z

ZooKeeper, 2-22