

Oracle® Big Data Appliance

Software User's Guide

Release 4 (4.7)

E81303-03

February 2017

Describes the Oracle Big Data Appliance software available to administrators and software developers.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface	ix
Audience	ix
Documentation Accessibility	ix
Related Documents.....	ix
Conventions.....	x
Backus-Naur Form Syntax.....	x

Part I Administration

1 Introducing Oracle Big Data Appliance

1.1 What Is Big Data?.....	1-1
1.1.1 High Variety.....	1-1
1.1.2 High Complexity	1-2
1.1.3 High Volume.....	1-2
1.1.4 High Velocity	1-2
1.2 The Oracle Big Data Solution.....	1-2
1.3 Software for Big Data Appliance.....	1-3
1.3.1 Software Component Overview.....	1-4
1.4 Acquiring Data for Analysis	1-5
1.4.1 Hadoop Distributed File System.....	1-5
1.4.2 Apache Hive.....	1-5
1.4.3 Oracle NoSQL Database	1-6
1.5 Organizing Big Data	1-6
1.5.1 MapReduce.....	1-7
1.5.2 Oracle Big Data SQL	1-7
1.5.3 Oracle Big Data Connectors	1-8
1.5.4 Oracle R Support for Big Data	1-9
1.6 Analyzing and Visualizing Big Data.....	1-10

2 Administering Oracle Big Data Appliance

2.1 Monitoring Multiple Clusters Using Oracle Enterprise Manager.....	2-1
2.1.1 Using the Enterprise Manager Web Interface	2-2

2.1.2	Using the Enterprise Manager Command-Line Interface	2-3
2.2	Managing Operations Using Cloudera Manager.....	2-3
2.2.1	Monitoring the Status of Oracle Big Data Appliance.....	2-4
2.2.2	Performing Administrative Tasks.....	2-5
2.2.3	Managing CDH Services With Cloudera Manager	2-5
2.3	Using Hadoop Monitoring Utilities	2-6
2.3.1	Monitoring MapReduce Jobs	2-6
2.3.2	Monitoring the Health of HDFS.....	2-6
2.4	Using Cloudera Hue to Interact With Hadoop	2-7
2.5	About the Oracle Big Data Appliance Software.....	2-9
2.5.1	Software Components	2-9
2.5.2	Unconfigured Software	2-11
2.5.3	Allocating Resources Among Services	2-12
2.6	About the CDH Software Services	2-12
2.6.1	Where Do the Services Run on a Three-Node, Development Cluster?	2-12
2.6.2	Where Do the Services Run on a Single-Rack CDH Cluster?	2-13
2.6.3	Where Do the Services Run on a Multirack CDH Cluster?.....	2-15
2.6.4	About MapReduce	2-19
2.6.5	Automatic Failover of the NameNode.....	2-19
2.6.6	Automatic Failover of the ResourceManager.....	2-20
2.6.7	Map and Reduce Resource Allocation	2-21
2.7	Effects of Hardware on Software Availability.....	2-21
2.7.1	Logical Disk Layout	2-21
2.7.2	Critical and Noncritical CDH Nodes.....	2-22
2.7.3	First NameNode Node	2-23
2.7.4	Second NameNode Node.....	2-23
2.7.5	First ResourceManager Node	2-24
2.7.6	Second ResourceManager Node	2-24
2.7.7	Noncritical CDH Nodes	2-25
2.8	Managing a Hardware Failure.....	2-25
2.8.1	About Oracle NoSQL Database Clusters	2-25
2.8.2	Prerequisites for Managing a Failing Node.....	2-25
2.8.3	Managing a Failing CDH Critical Node	2-26
2.8.4	Managing a Failing Noncritical Node	2-26
2.9	Stopping and Starting Oracle Big Data Appliance	2-27
2.9.1	Prerequisites.....	2-27
2.9.2	Stopping Oracle Big Data Appliance.....	2-28
2.9.3	Starting Oracle Big Data Appliance.....	2-30
2.10	Managing Oracle Big Data SQL.....	2-31
2.10.1	Adding and Removing the Oracle Big Data SQL Service	2-32
2.10.2	Allocating Resources to Oracle Big Data SQL.....	2-32
2.11	Security on Oracle Big Data Appliance	2-33
2.11.1	About Predefined Users and Groups	2-34

2.11.2	About User Authentication.....	2-35
2.11.3	About Fine-Grained Authorization	2-35
2.11.4	About HDFS Transparent Encryption.....	2-35
2.11.5	About HTTPS/Network Encryption.....	2-36
2.11.6	Port Numbers Used on Oracle Big Data Appliance.....	2-38
2.11.7	About Puppet Security	2-39
2.12	Auditing Oracle Big Data Appliance	2-40
2.12.1	About Oracle Audit Vault and Database Firewall	2-40
2.12.2	Setting Up the Oracle Big Data Appliance Plug-in	2-41
2.12.3	Monitoring Oracle Big Data Appliance.....	2-41
2.13	Collecting Diagnostic Information for Oracle Customer Support	2-42
3	Supporting User Access to Oracle Big Data Appliance	
3.1	About Accessing a Kerberos-Secured Cluster	3-1
3.2	Providing Remote Client Access to CDH.....	3-2
3.2.1	Prerequisites.....	3-2
3.2.2	Installing a CDH Client on Any Supported Operating System	3-3
3.2.3	Configuring a CDH Client for an Unsecured Cluster.....	3-3
3.2.4	Configuring a CDH Client for a Kerberos-Secured Cluster.....	3-4
3.2.5	Verifying Access to a Cluster from the CDH Client.....	3-6
3.3	Providing Remote Client Access to Hive	3-7
3.4	Managing User Accounts	3-8
3.4.1	Creating Hadoop Cluster Users	3-8
3.4.2	Providing User Login Privileges (Optional).....	3-10
3.5	Recovering Deleted Files	3-11
3.5.1	Restoring Files from the Trash	3-11
3.5.2	Changing the Trash Interval.....	3-11
3.5.3	Disabling the Trash Facility	3-12
4	Configuring Oracle Exadata Database Machine for Use with Oracle Big Data Appliance	
4.1	About Optimizing Communications	4-1
4.1.1	About Applications that Pull Data Into Oracle Exadata Database Machine	4-1
4.1.2	About Applications that Push Data Into Oracle Exadata Database Machine	4-2
4.2	Prerequisites for Optimizing Communications	4-2
4.3	Specifying the InfiniBand Connections to Oracle Big Data Appliance.....	4-2
4.4	Specifying the InfiniBand Connections to Oracle Exadata Database Machine	4-3
4.5	Enabling SDP on Exadata Database Nodes	4-4
4.6	Creating an SDP Listener on the InfiniBand Network.....	4-5

Part II Oracle Big Data Appliance Software

5 Optimizing MapReduce Jobs Using Perfect Balance

5.1	What is Perfect Balance?	5-1
5.1.1	About Balancing Jobs Across Map and Reduce Tasks.....	5-2
5.1.2	Ways to Use Perfect Balance Features.....	5-2
5.1.3	Perfect Balance Components	5-2
5.2	Application Requirements.....	5-2
5.3	Getting Started with Perfect Balance	5-3
5.4	Analyzing a Job's Reducer Load.....	5-4
5.4.1	About Job Analyzer.....	5-4
5.4.2	Running Job Analyzer as a Standalone Utility	5-4
5.4.3	Running Job Analyzer Using Perfect Balance	5-6
5.4.4	Reading the Job Analyzer Report	5-8
5.5	About Configuring Perfect Balance	5-9
5.6	Running a Balanced MapReduce Job Using Perfect Balance	5-11
5.7	About Perfect Balance Reports	5-13
5.8	About Chopping	5-13
5.8.1	Selecting a Chopping Method	5-13
5.8.2	How Chopping Impacts Applications	5-14
5.9	Troubleshooting Jobs Running with Perfect Balance	5-15
5.10	Using the Perfect Balance API	5-15
5.10.1	Modifying Your Java Code to Use Perfect Balance	5-15
5.10.2	Running Your Modified Java Code with Perfect Balance	5-16
5.11	About the Perfect Balance Examples	5-17
5.11.1	About the Examples in This Chapter	5-18
5.11.2	Extracting the Example Data Set.....	5-18
5.12	Perfect Balance Configuration Property Reference	5-18

Part III Oracle Table Access for Hadoop and Spark

6 Oracle DataSource for Apache Hadoop (OD4H)

6.1	Operational Data, Big Data and Requirements	6-1
6.2	Overview of Oracle DataSource for Apache Hadoop (OD4H).....	6-1
6.2.1	Opportunity with Hadoop 2.x.....	6-2
6.2.2	Oracle Tables as Hadoop Data Source	6-2
6.2.3	External Tables.....	6-3
6.2.4	List of jars in the OD4H package	6-5
6.3	How does OD4H work?.....	6-5
6.3.1	Create a new Oracle Database Table or Reuse an Existing Table	6-6
6.3.2	Hive DDL.....	6-6
6.3.3	Creating External Tables in Hive	6-8

6.4	Features of OD4H	6-8
6.4.1	Performance And Scalability Features	6-9
6.4.2	Smart Connection Management.....	6-13
6.4.3	Security Features	6-14
6.5	Using HiveQL with OD4H.....	6-17
6.6	Using Spark SQL with OD4H	6-17
6.7	Writing Back to Oracle Database.....	6-18

Glossary

Index

Preface

This guide describes how to manage and use the installed Oracle Big Data Appliance software.

Note: Oracle Big Data SQL is no longer documented within this guide. See the [Oracle Big Data Appliance User's Guide](#) for instructions on how to install and use Oracle Big Data SQL.

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 that have purchased support 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.
<i>italic</i>	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.
# prompt	The pound (#) prompt indicates a command that is run as the Linux root user.

Backus-Naur Form Syntax

The syntax in this reference is presented in a simple variation of Backus-Naur Form (BNF) that uses the following symbols and conventions:

Symbol or Convention	Description
[]	Brackets enclose optional items.
{ }	Braces enclose a choice of items, only one of which is required.
	A vertical bar separates alternatives within brackets or braces.
...	Ellipses indicate that the preceding syntactic element can be repeated.
delimiters	Delimiters other than brackets, braces, and vertical bars must be entered as shown.
boldface	Words appearing in boldface are keywords. They must be typed as shown. (Keywords are case-sensitive in some, but not all, operating systems.) Words that are not in boldface are placeholders for which you must substitute a name or value.

Part I

Administration

This part describes Oracle Big Data Appliance and provides instructions for routine administrative tasks. It contains the following chapters:

- [Introducing Oracle Big Data Appliance](#)
- [Administering Oracle Big Data Appliance](#)
- [Supporting User Access to Oracle Big Data Appliance](#)
- [Configuring Oracle Exadata Database Machine for Use with Oracle Big Data Appliance](#)

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 Appliance](#)
- [Acquiring Data for Analysis](#)
- [Organizing Big Data](#)
- [Analyzing and Visualizing Big Data](#)

1.1 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:

- A variety of data sources: [High Variety](#)
- A complexity of data types: [High Complexity](#)
- A high volume of data flow: [High Volume](#)
- A high velocity of data transactions: [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.

1.1.1 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

1.1.2 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.

1.1.3 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

1.1.4 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

1.2 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.

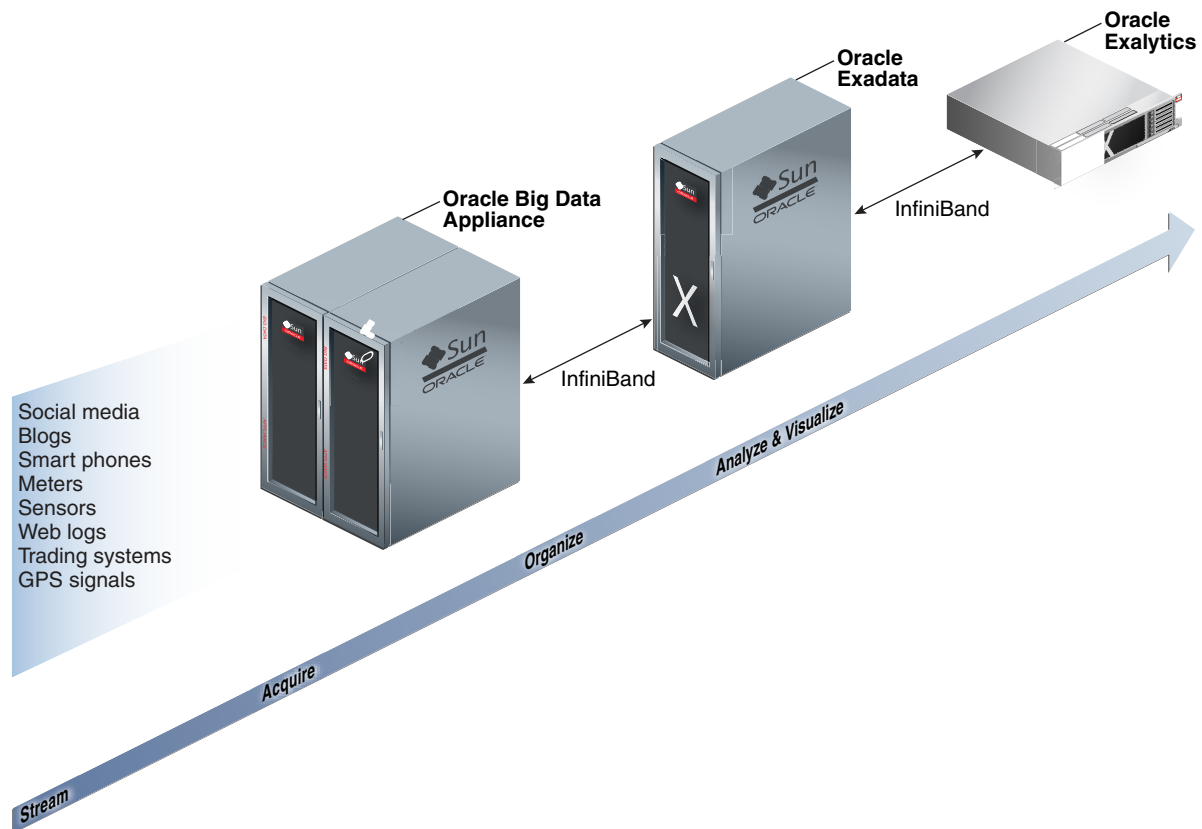
Oracle Database enables all data to be accessed and analyzed by a large user community using identical methods. By adding Oracle Big Data Appliance in front of Oracle Database, you can bring new sources of information to an existing data warehouse. Oracle Big Data Appliance is the platform for acquiring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



1.3 Software for Big Data Appliance

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. Oracle Big Data Appliance 3.0 runs **YARN** by default.
- **Administrative framework:** **Cloudera Manager** is a comprehensive administrative tool for CDH. In addition, you can use Oracle Enterprise Manager to monitor both the hardware and software on Oracle Big Data Appliance.
- **Apache projects:** CDH includes Apache projects for MapReduce and HDFS, such as **Hive**, **Pig**, **Oozie**, **ZooKeeper**, **HBase**, **Sqoop**, and **Spark**.
- **Cloudera applications:** Oracle Big Data Appliance installs all products included in Cloudera Enterprise Data Hub Edition, including **Impala**, **Search**, and **Navigator**.

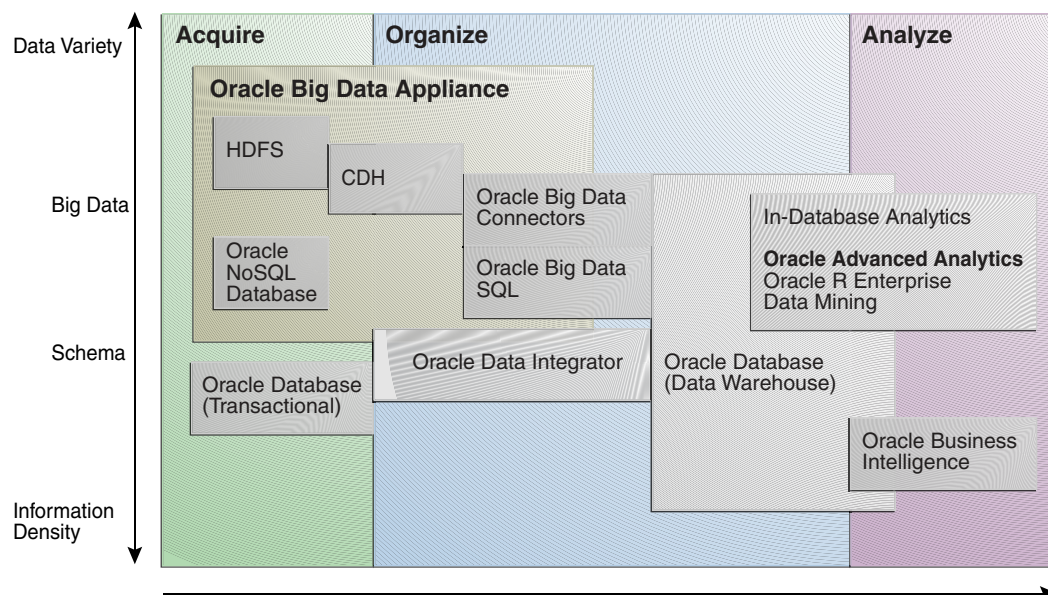
1.3.1 Software Component Overview

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



1.4 Acquiring Data for Analysis

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.

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

- [Hadoop Distributed File System](#)
- [Apache Hive](#)
- [Oracle NoSQL Database](#)

1.4.1 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/>
-
-

1.4.2 Apache 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.

1.4.3 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 documentation](#)
 - *Oracle Big Data Appliance Licensing Information*
-

1.5 Organizing Big Data

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

- [MapReduce](#)
- [Oracle Big Data SQL](#)
- [Oracle Big Data Connectors](#)
- [Oracle R Support for Big Data](#)

1.5.1 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 uses the Yet Another Resource Negotiator (YARN) implementation of MapReduce.

1.5.2 Oracle Big Data SQL

Oracle Big Data SQL supports queries against vast amounts of big data stored in multiple data sources, including Apache Hive, HDFS, Oracle NoSQL Database, and Apache HBase. You can view and analyze data from various data stores together, as if it were all stored in an Oracle database.

Using Oracle Big Data SQL, you can query data stored in a Hadoop cluster using the complete SQL syntax. You can execute the most complex SQL `SELECT` statements against data in Hadoop, either manually or using your existing applications, to tease out the most significant insights.

Oracle Big Data SQL is licensed separately from Oracle Big Data Appliance.

Oracle Big Data SQL includes the Copy to Hadoop and Oracle Shell for Hadoop Loaders features. The Oracle Shell for Hadoop Loaders helper shell uses the Copy to Hadoop feature of Big Data SQL to identify and copy Oracle Database data to the Hadoop Distributed File System. An Apache Hive table is created over the data that is copied, allowing Hive to process the data locally.

See Also:

Oracle Big Data SQL User's Guide

1.5.3 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 Enterprise Edition](#)
- [Oracle Shell for Hadoop Loaders](#)

See Also:

Oracle Big Data Connectors User's Guide

1.5.3.1 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 Data Pump files generated by Oracle Loader for Hadoop
- Load data extracted and transformed by Oracle Data Integrator

1.5.3.2 Oracle Loader for Hadoop

Oracle Loader for Hadoop is an efficient and high-performance loader for fast movement of data from a Hadoop cluster into a table in an Oracle database. It can read and load data from a wide variety of formats. Oracle Loader for Hadoop partitions the data and transforms it into a database-ready format in Hadoop. It optionally sorts records by primary key before loading the data or creating output files. The load runs as a MapReduce job on the Hadoop cluster.

1.5.3.3 Oracle Data Integrator Enterprise Edition

Oracle Data Integrator (ODI) Enterprise Edition 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 Enterprise Edition contains the KMs specifically for use with big data.

1.5.3.4 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.

1.5.3.5 Oracle R Advanced Analytics for Hadoop

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

- 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 write mappers and reducers in R, schedule these R programs to execute as Hadoop MapReduce jobs, and return the results to any of those locations.

1.5.3.6 Oracle Shell for Hadoop Loaders

Oracle Shell for Hadoop Loaders is a helper shell that provides a simple to use command line interface to Oracle Loader for Hadoop, Oracle SQL Connector for HDFS, and the Copy to Hadoop feature of Big Data SQL.

1.5.4 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.

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](#)".

Oracle R Enterprise is a component of the Oracle Advanced Analytics option to Oracle Database. It provides:

- Transparent access to database data for data preparation and statistical analysis from R
- Execution of R scripts at the database server, accessible from both R and SQL
- A wide range of predictive and data mining in-database algorithms

Oracle R Enterprise enables you to store the results of your analysis of big data in an Oracle database, or accessed for display in dashboards and applications.

Both Oracle R Advanced Analytics for Hadoop and Oracle R Enterprise 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/E67822_01/index.htm
-
-

1.6 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.

Oracle Big Data Discovery

Oracle Big Data Discovery is an end-to-end product for visual analysis of Big Data, built natively on Hadoop. It transforms raw data into business insight in minutes, without the need to learn complex products or rely only on highly specialized resources. This product can be installed on Oracle Big Data Appliance as a post-Mammoth installation add-on. The installation process is largely automated, which make it easy to add Oracle Big Data Discovery to the Oracle Big Data Appliance tool set.

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://docs.oracle.com/database/121/nav/portal_6.htm
-

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 Multiple Clusters Using Oracle Enterprise Manager](#)
- [Managing 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](#)
- [Managing a Hardware Failure](#)
- [Stopping and Starting Oracle Big Data Appliance](#)
- [Managing Oracle Big Data SQL](#)
- [Security on Oracle Big Data Appliance](#)
- [Auditing Oracle Big Data Appliance](#)
- [Collecting Diagnostic Information for Oracle Customer Support](#)

2.1 Monitoring Multiple Clusters 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.

Oracle Enterprise Manager enables you to monitor all Oracle Big Data Appliance racks on the same InfiniBand fabric. It provides summary views of both the rack hardware and the software layout of the logical clusters.

Note: Before you start, contact Oracle Support for up-to-date information about Enterprise Manager plug-in functionality.

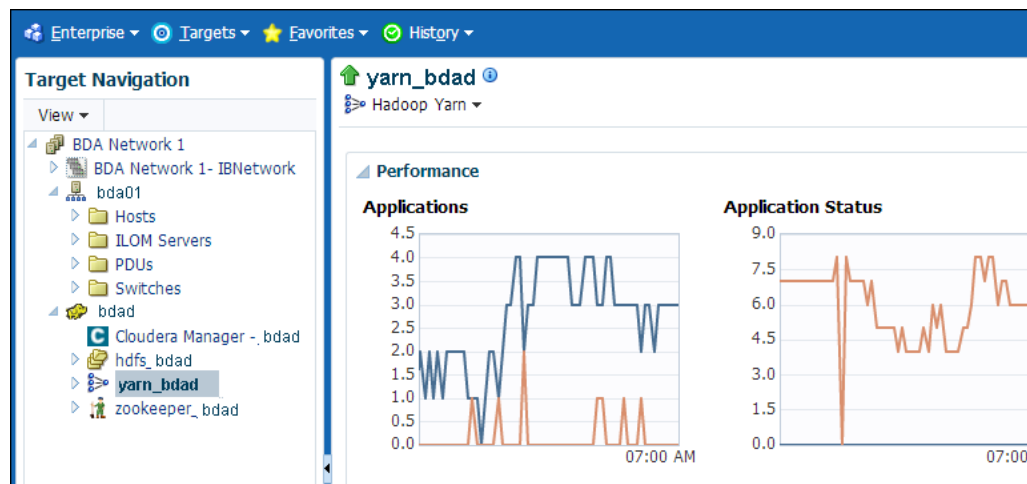
2.1.1 Using the Enterprise Manager Web Interface

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 YARN Page 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.

2.1.2 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](#)

2.2 Managing 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 ResourceManager node (`node03`) and is available on port 7180.

To use Cloudera Manager:

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

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-docs/CM5/latest/Cloudera-Manager-Diagnostics-Guide/Cloudera-Manager-Diagnostics-Guide.html>

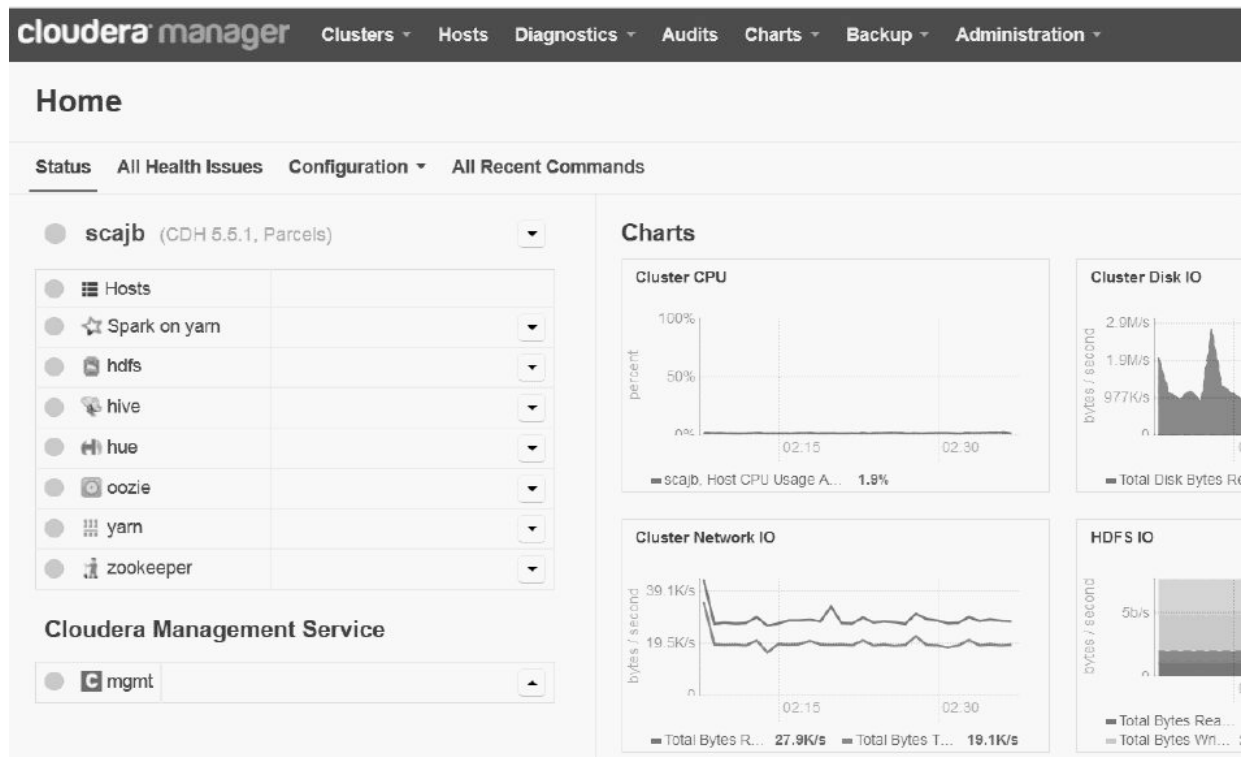
or click **Help** on the Cloudera Manager Support menu

2.2.1 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 and links to all services controlled by Cloudera Manager. See [Figure 2-2](#).
- **Clusters:** Accesses the services on multiple clusters.
- **Hosts:** Monitors the health, disk usage, load, physical memory, swap space, and other statistics for all servers in the cluster.
- **Diagnostics:** Accesses events and logs. 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:** Enables you to view metrics from the Cloudera Manager time-series data store in a variety of chart types, such as line and bar.
- **Backup:** Accesses snapshot policies and scheduled replications.
- **Administration:** Provides a variety of administrative options, including Settings, Alerts, Users, and Kerberos.

[Figure 2-2](#) shows the Cloudera Manager home page.

Figure 2-2 Cloudera Manager Home Page

2.2.2 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. Click **Administration**, and select a task from the menu.

2.2.3 Managing CDH Services With Cloudera Manager

Cloudera Manager provides the interface for managing these services:

- HDFS
- Hive
- Hue
- Oozie
- YARN
- 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](#)."

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.

2.3 Using Hadoop Monitoring Utilities

You also have the option of using the native Hadoop utilities. These utilities are read-only and do not require authentication.

Cloudera Manager provides an easy way to obtain the correct URLs for these utilities. On the YARN service page, expand the Web UI submenu.

2.3.1 Monitoring MapReduce Jobs

You can monitor MapReduce jobs using the resource manager interface.

To monitor MapReduce jobs:

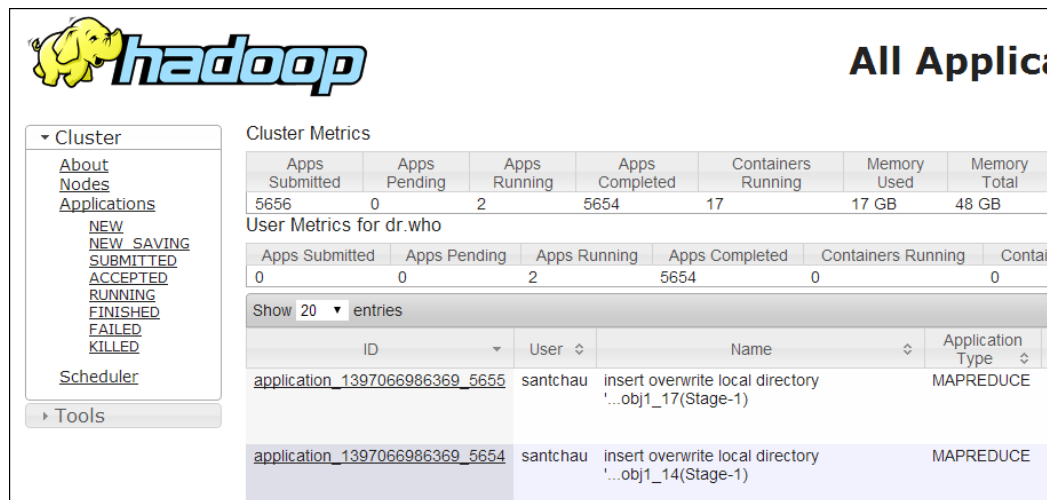
- Open a browser and enter a URL like the following:

`http://bda1node03.example.com:8088`

In this example, `bda1` is the name of the rack, `node03` is the name of the server where the YARN resource manager runs, and `8088` is the default port number for the user interface.

Figure 2-3 shows the resource manager interface.

Figure 2-3 YARN Resource Manager Interface



2.3.2 Monitoring the Health of HDFS

You can monitor the health of the Hadoop file system by using the DFS health utility on the first two nodes of a cluster.

To monitor HDFS:

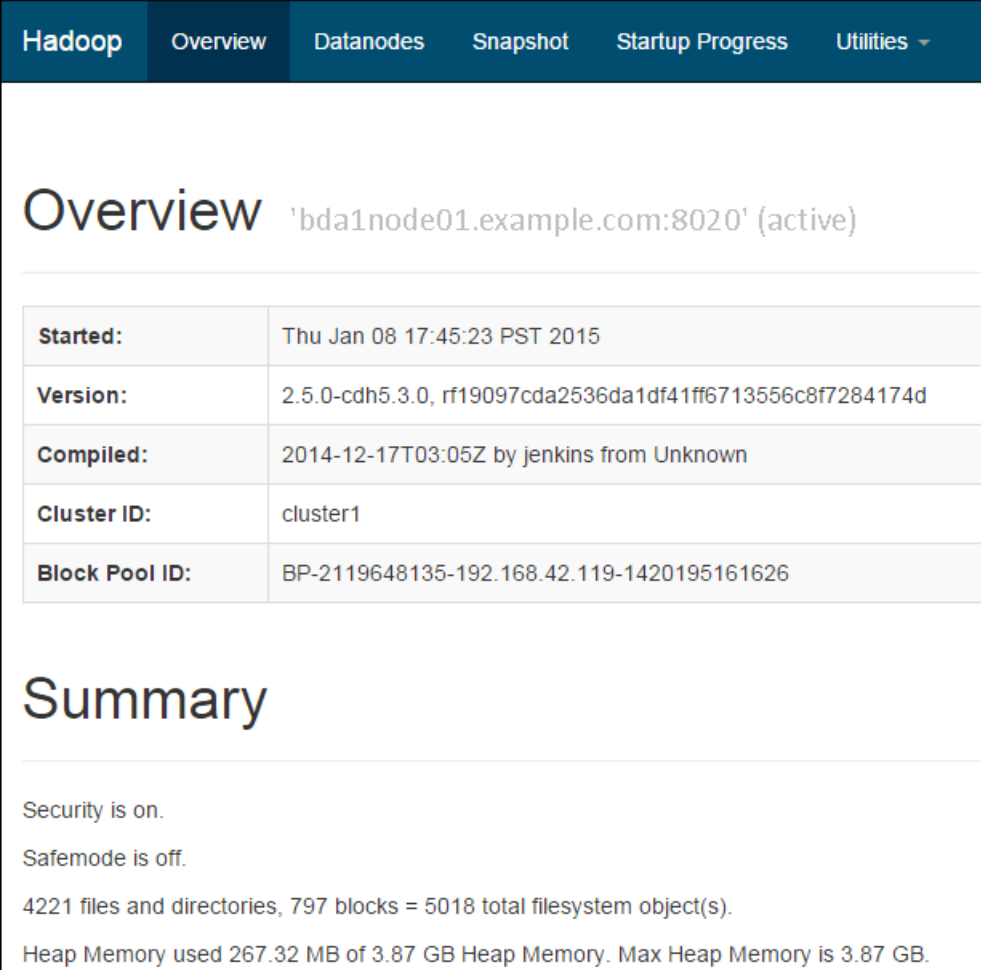
- Open a browser and enter a URL like the following:

`http://bda1node01.example.com:50070`

In this example, `bda1` is the name of the rack, `node01` is the name of the server where the `dfshealth` utility runs, and `50070` is the default port number for the user interface.

Figure 2-3 shows the DFS health utility interface.

Figure 2-4 DFS Health Utility



Hadoop Overview Datanodes Snapshot Startup Progress Utilities ▾

Overview

'bda1node01.example.com:8020' (active)

Started:	Thu Jan 08 17:45:23 PST 2015
Version:	2.5.0-cdh5.3.0, rf19097cda2536da1df41ff6713556c8f7284174d
Compiled:	2014-12-17T03:05Z by jenkins from Unknown
Cluster ID:	cluster1
Block Pool ID:	BP-2119648135-192.168.42.119-1420195161626

Summary

Security is on.

Safemode is off.

4221 files and directories, 797 blocks = 5018 total filesystem object(s).

Heap Memory used 267.32 MB of 3.87 GB Heap Memory. Max Heap Memory is 3.87 GB.

2.4 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 is automatically installed and configured on Oracle Big Data Appliance. It runs on port 8888 of the ResourceManager node. See the tables in [About the CDH Software Services](#) for Hue's location within different cluster configurations.

To use Hue:

1. Log in to Cloudera Manager and click the **hue** service on the Home page.
2. On the hue page under Quick Links, click Hue Web UI.
3. Bookmark the Hue URL, so that you can open Hue directly in your browser. The following URL is an example:

`http://bdalnode03.example.com:8888`

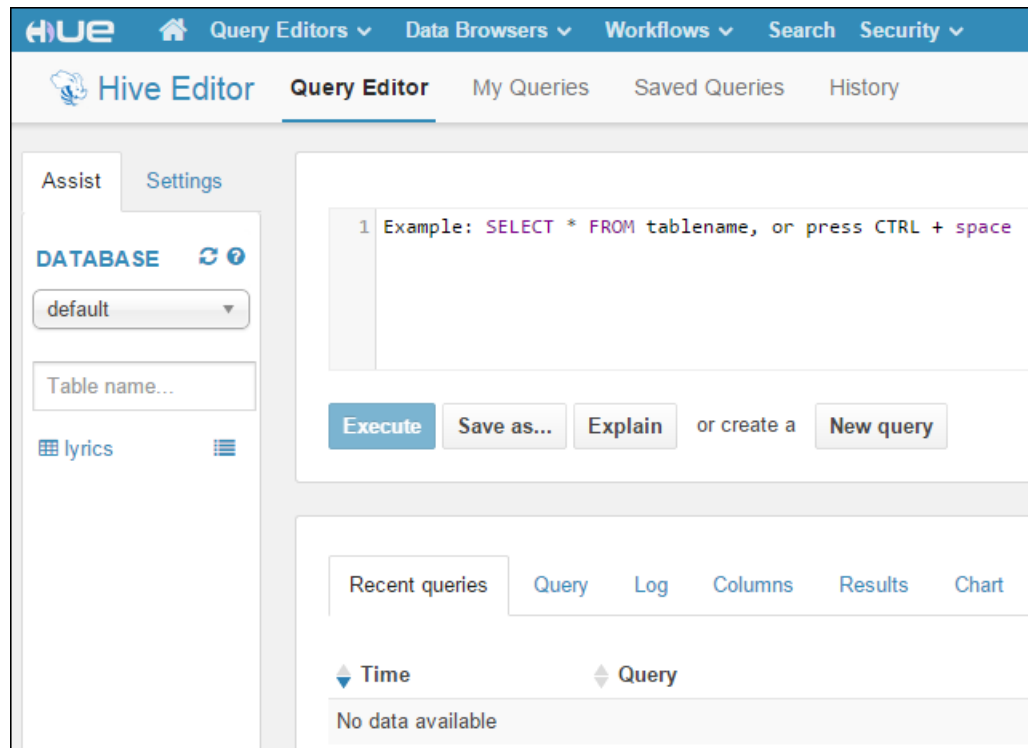
4. Log in with your Hue credentials.

If Hue accounts have not been created yet, log into the default Hue administrator account by using the following credentials:

- Username: `admin`
- Password: `cm-admin-password`

where `cm-admin-password` is the password specified when the cluster for the Cloudera Manager admin user was activated. You can then create other user and administrator accounts.

[Figure 2-5](#) shows the Hive Query Editor.

Figure 2-5 *Hive Query Editor***See Also:**

Hue User Guide at

<http://archive-primary.cloudera.com/cdh5/cdh/5/hue/user-guide/>

2.5 About the Oracle Big Data Appliance Software

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

This section contains the following topics:

- [Software Components](#)
- [Unconfigured Software](#)
- [Allocating Resources Among Services](#)

2.5.1 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 6.8 with Oracle Unbreakable Enterprise Kernel version 4 (UEK4). Oracle Linux 5 Big Data Appliance upgrades stay at 5.11 (UEK2).
- Java JDK 8u111
- [Oracle R Distribution 3.2.0](#)
- [MySQL Enterprise Server 5.6.32](#)
- Puppet, firmware, Oracle Big Data Appliance utilities
- Oracle InfiniBand software

Mammoth installation:

- Cloudera's Distribution including Apache Hadoop Release 5.9 including:
 - Apache Hive
 - Apache HBase
 - [Apache Sentry](#)
 - Apache Spark
 - [Cloudera Impala](#)
 - [Cloudera Search](#)
- Cloudera Manager Release 5.9.
- [Oracle Database Instant Client 12.1](#)
- Oracle Big Data SQL 3.0.1 (optional). Also compatible with Oracle Big Data SQL 3.1.
- Oracle NoSQL Database Community Edition or Enterprise Edition 4.2.9 (optional)
- Oracle Big Data Connectors 4.7 (optional):
 - Oracle SQL Connector for Hadoop Distributed File System (HDFS)
 - Oracle Loader for Hadoop
 - Oracle Data Integrator Agent
 - Oracle XQuery for Hadoop
 - Oracle R Advanced Analytics for Hadoop

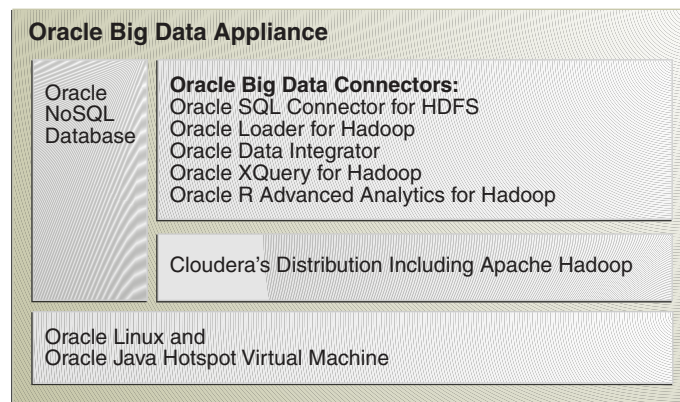
- Oracle Big Data Discovery 1.3.2, or, 1.4 (1.4.0.37.1388 or greater). Oracle Big Data Discovery can be downloaded from the [Oracle Software Delivery Cloud](#).

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



2.5.2 Unconfigured Software

Your Oracle Big Data Appliance license includes all components in Cloudera Enterprise Data Hub Edition. All CDH components are installed automatically by the Mammoth utility. Do not download them from the Cloudera website.

However, you must use Cloudera Manager to add some services before you can use them, such as the following:

- [Apache Flume](#)
- [Apache HBase](#)
- [Apache Spark](#)
- [Apache Sqoop](#)
- [Cloudera Impala](#)
- [Cloudera Search](#)

To add a service:

1. Log in to Cloudera Manager as the admin user.
2. On the Home page, expand the cluster menu in the left panel and choose **Add a Service** to open the Add Service wizard. The first page lists the services you can add.
3. Follow the steps of the wizard.

See Also:

- For a list of key CDH components:
<http://www.cloudera.com/content/www/en-us/products/apache-hadoop/key-cdh-components.html>
 - *CDH5 Installation and Configuration Guide* for configuration procedures at
<http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH5/latest/CDH5-Installation-Guide/CDH5-Installation-Guide.html>
-

2.5.3 Allocating Resources Among Services

You can allocate resources to each service—HDFS, YARN, Oracle Big Data SQL, Hive, and so forth—as a percentage of the total resource pool. Cloudera Manager automatically calculates the recommended resource management settings based on these percentages. The static service pools isolate services on the cluster, so that a high load on one service has a limited impact on the other services.

To allocate resources among services:

1. Log in as admin to Cloudera Manager.
2. Open the Clusters menu at the top of the page, then select **Static Service Pools** under Resource Management.
3. Select **Configuration**.
4. Follow the steps of the wizard, or click **Change Settings Directly** to edit the current settings.

2.6 About the CDH Software Services

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.

This section describes the services in a default YARN configuration.

This section contains the following topics:

- [Where Do the Services Run on a Single-Rack CDH Cluster?](#)
- [Where Do the Services Run on a Multirack CDH Cluster?](#)
- [About MapReduce](#)
- [Automatic Failover of the NameNode](#)
- [Automatic Failover of the ResourceManager](#)

2.6.1 Where Do the Services Run on a Three-Node, Development Cluster?

Oracle Big Data Appliance now enables the use of three-node clusters for development purposes.

Caution: Three-node clusters are generally not suitable for production environments because all of the nodes are master nodes. This puts constraints on high availability. The minimum recommended cluster size for a production environment is five nodes

Table 2-1 Service Locations for a Three-Node Development Cluster

Node1	Node2	Node3
NameNode	NameNode/Failover	-
Failover Controller	Failover Controller	-
DataNode	DataNode	DataNode
NodeManager	NodeManager	NodeManager
JournalNode	JournalNode	JournalNode
Sentry	HttpFS	Cloudera Manager and CM roles
-	MySQL Backup	MySQL Primary
ResourceManager	Hive	ResourceManager
-	Hive Metastore	JobHistory
-	ODI	Spark History
-	Oozie	-
-	Hue	-
-	WebHCat	-
ZooKeeper	ZooKeeper	ZooKeeper
Active Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled)	Passive Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled)	-
Kerberos Master KDC (Only if MIT Kerberos is enabled and on-BDA KDCs are being used.)	Kerberos Slave KDC (Only if MIT Kerberos is enabled and on-BDA KDCs are being used.)	-

2.6.2 Where Do the Services Run on a Single-Rack CDH Cluster?

Table 2-2 identifies the services in CDH clusters configured within a single rack, including starter racks and clusters with five or more nodes. Node01 is the first server in the cluster (server 1, 7, or 10), and nodenn is the last server in the cluster (server 6, 9, 12, or 18). Multirack clusters have different services layouts, as do three-node clusters for development purposes. Both are described separately in this chapter.

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

Node01	Node02	Node03	Node04	Node05 to nn
Balancer	-	Cloudera Manager Server	-	-
Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent
DataNode	DataNode	DataNode	DataNode	DataNode
Failover Controller	Failover Controller	-	Hive, Hue, Oozie	-
JournalNode	JournalNode	JournalNode	-	-
-	MySQL Backup	MySQL Primary	-	-
NameNode	NameNode	Navigator Audit Server and Navigator Metadata Server	-	-
NodeManager (in clusters of eight nodes or less)	NodeManager (in clusters of eight nodes or less)	NodeManager	NodeManager	NodeManager
-	-	SparkHistoryServer	Oracle Data Integrator Agent	-
-	-	ResourceManager	ResourceManager	-
ZooKeeper	ZooKeeper	ZooKeeper	-	-
Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)
Kerberos KDC (if MIT Kerberos is enabled and on-BDA KDCs are being used)	Kerberos KDC (if MIT Kerberos is enabled and on-BDA KDCs are being used)	JobHistory	-	-
Sentry Server (if enabled)	-	-	-	-
Active Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled)	Passive Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled)	-	-	-
-	HttpFS	-	-	-

Note: If Oracle Big Data Discovery is installed, the NodeManager and DataNode on Node05 of the primary cluster are decommissioned.

If Oozie high availability is enabled, then Oozie servers are hosted on Node04 and another node (preferably a ResourceNode) selected by the customer.

2.6.3 Where Do the Services Run on a Multirack CDH Cluster?

When multiple racks are configured as a single CDH cluster, some critical services are installed on the second rack. There can be variations in the distribution of services between the first and second racks in different multirack clusters. The two scenarios that account for these differences are:

- The cluster spanned multiple racks in its original configuration.

The resulting service locations across the first and second rack are described in [Table 2-3](#) and [Table 2-4](#). In this case, the JournalNode, Mysql Primary, ResourceManager, and Zookeeper are installed on node 2 of the first rack.

- The cluster was single-rack in the original configuration and was extended later.

The resulting service locations across the first and second rack are described in [Table 2-5](#) and [Table 2-6](#). In this case, the JournalNode, Mysql Primary, ResourceManager, and Zookeeper are installed on node 3 of the first rack.

Note: If Oracle Big Data Discovery is installed, the NodeManager and DataNode on Node05 in the first rack of the primary cluster are decommissioned.

There is one variant that is determined specifically by cluster size – for clusters of eight nodes less, nodes that run NameNode also run NodeManager. This is not true for clusters larger than eight nodes.

Table 2-3 First Rack Service Locations (When the Cluster Started as Multirack Cluster)

Node01	Node02	Node03	Node04	Node05 to <i>nn</i> 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	-	-	-	-
JournalNode	JournalNode	Navigator Audit Server and Navigator Metadata Server	-	-

Table 2-3 (Cont.) First Rack Service Locations (When the Cluster Started as Multirack Cluster)

Node01	Node02	Node03	Node04	Node05 to <i>nn</i> ¹
NameNode	MySQL Primary	SparkHistoryServer	-	-
NodeManager (in clusters of eight nodes or less)	NodeManager	NodeManager	NodeManager	NodeManager
-	ResourceManager	-	-	-
ZooKeeper	ZooKeeper	-	-	-
Kerberos KDC (Only if MIT Kerberos is enabled and on-BDA KDCs are being used.)	Kerberos KDC (Only if MIT Kerberos is enabled and on-BDA KDCs are being used.)			
Sentry Server (Only if Sentry is enabled.)				
Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)
Active Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled)				

¹ *nn* includes the servers in additional racks.

Table 2-4 shows the service locations in the second rack of a cluster that was originally configured to span multiple racks.

Table 2-4 Second Rack Service Locations (When the Cluster Started as Multirack Cluster)

Node01	Node02	Node03	Node04	Node05 to <i>nn</i>
Balancer	-	-	-	-
Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent
DataNode	DataNode	DataNode	DataNode	DataNode
Failover Controller	-	-	-	-

Table 2-4 (Cont.) Second Rack Service Locations (When the Cluster Started as Multirack Cluster)

Node01	Node02	Node03	Node04	Node05 to nn
JournalNode	Hive, Hue, Oozie	-	-	-
MySQL Backup	-	-	-	-
NameNode	-	-	-	-
NodeManager (in clusters of eight nodes or less)	NodeManager	NodeManager	NodeManager	NodeManager
HttpFS	Oracle Data Integrator Agent	-	-	-
-	ResourceManager	-	-	-
ZooKeeper	-	-	-	-
Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)
Passive Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled)				

[Table 2-5](#) shows the service locations in the first rack of a cluster that was originally configured as a single-rack cluster and subsequently extended.

Table 2-5 First Rack Service Locations (When a Single-Rack Cluster is Extended)

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	-	Navigator Audit Server and Navigator Metadata Server	-	-
JournalNode	-	JournalNode	-	-
NameNode	-	MySQL Primary	-	-
NodeManager (in clusters of eight nodes or less)	NodeManager	NodeManager	NodeManager	NodeManager

Table 2-5 (Cont.) First Rack Service Locations (When a Single-Rack Cluster is Extended)

Node01	Node02	Node03	Node04	Node05 to <i>nn</i> ¹
-	-	ResourceManager	-	-
ZooKeeper	-	ZooKeeper	-	-
Kerberos KDC (Only if MIT Kerberos is enabled and on- BDA KDCs are being used.)	Kerberos KDC (Only if MIT Kerberos is enabled and on- BDA KDCs are being used.)	SparkHistoryServer	-	-
Sentry Server (Only if Sentry is enabled.)	-	-	-	-
Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)
Active Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled)	-	-	-	-

¹ *nn* includes the servers in additional racks.

Table 2-6 shows the service locations in the second rack of a cluster originally configured as a single-rack cluster and subsequently extended.

Table 2-6 Second Rack Service Locations (When a Single-Rack Cluster is Extended)

Node01	Node02	Node03	Node04	Node05 to <i>nn</i>
Balancer				
Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent	Cloudera Manager Agent
DataNode	DataNode	DataNode	DataNode	DataNode
Failover Controller	-	-	-	-
JournalNode	Hive, Hue, Oozie, Solr	-	-	-
MySQL Backup	-	-	-	-
NameNode	-	-	-	-
NodeManager (in clusters of eight nodes or less)	NodeManager	NodeManager	NodeManager	NodeManager

Table 2-6 (Cont.) Second Rack Service Locations (When a Single-Rack Cluster is Extended)

Node01	Node02	Node03	Node04	Node05 to nn
HttpFS	Oracle Data Integrator Agent	-	-	-
-	ResourceManager	-	-	-
ZooKeeper	-	-	-	-
Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)	Big Data SQL (if enabled)
Passive Navigator Key Trustee Server (if HDFS Transparent Encryption is enabled)				

Note:

When expanding a cluster from one to two racks, Mammoth moves all critical services from nodes 2 and 4 of the first rack to nodes 1 and 2 of the second rack. Nodes 2 and 4 of the first rack become noncritical nodes.

2.6.4 About MapReduce

Yet Another Resource Negotiator (YARN) is the version of MapReduce that runs on Oracle Big Data Appliance, beginning with version 3.0. MapReduce applications developed using MapReduce 1 (MRv1) may require recompilation to run under YARN.

The ResourceManager performs all resource management tasks. An MRAppMaster performs the job management tasks. Each job has its own MRAppMaster. The NodeManager has containers that can run a map task, a reduce task, or an MRAppMaster. The NodeManager can dynamically allocate containers using the available memory. This architecture results in improved scalability and better use of the cluster than MRv1.

YARN also manages resources for Spark and Impala.

See Also:

"Running Existing Applications on Hadoop 2 YARN" at

<http://hortonworks.com/blog/running-existing-applications-on-hadoop-2-yarn/>

2.6.5 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 (CDH5) 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 of a cluster. 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**.

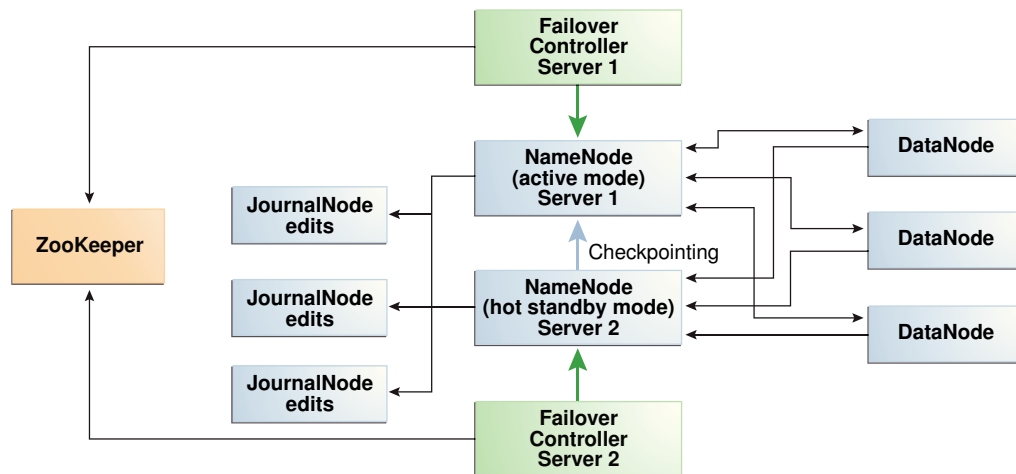
On Oracle Big Data Appliance, the default log level of the NameNode is `DEBUG`, to support the Oracle Audit Vault and Database Firewall plugin. If this option is not configured, then you can reset the log level to `INFO`.

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



2.6.6 Automatic Failover of the ResourceManager

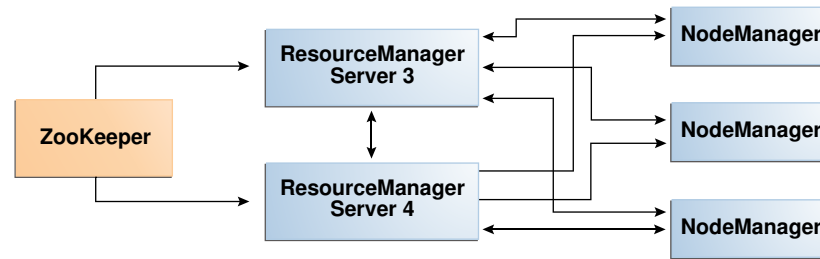
The ResourceManager allocates resources for application tasks and application masters across the cluster. Like the NameNode, the ResourceManager is a critical point of failure for the cluster. If all ResourceManagers fail, then all jobs stop running.

Oracle Big Data Appliance supports ResourceManager High Availability in Cloudera 5 to reduce this vulnerability.

CDH maintains redundant ResourceManager services on node03 and node04. One of the services is in active mode, and the other service is in hot standby mode. If the active service fails, then the role of active ResourceManager automatically fails over to the standby service. No failover controllers are required.

[Figure 2-8](#) shows the relationships among the processes that support automatic failover of the ResourceManager.

Figure 2-8 Automatic Failover of the ResourceManager on Oracle Big Data Appliance



2.6.7 Map and Reduce Resource Allocation

Oracle Big Data Appliance dynamically allocates memory to YARN. The allocation depends upon the total memory on the node and whether the node is one of the four critical nodes.

If you add memory, update the NodeManager container memory by increasing it by 80% of the memory added. Leave the remaining 20% for overhead.

2.7 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 Services Run on a Single-Rack CDH Cluster?"](#)

Note:

In a multirack cluster, some critical services run on the first server of the second rack. See ["Where Do the Services Run on a Multirack CDH Cluster?"](#)

2.7.1 Logical Disk Layout

Each server has 12 disks. . Disk partitioning is described in the table below.

[#unique_101/unique_101_Connect_42_CHDCJBAD](#) describes how the disks are partitioned.

The operating system is installed on disks 1 and 2. These two disks are mirrored. They include the Linux operating system, all installed software, NameNode data, and MySQL Database data. The NameNode and MySQL Database data are replicated on the two servers for a total of four copies. As shown in the table, in cases where Disk 1

and 2 are 4 TB drives these include a 3440 GB HDFS data partition. If Disk 1 and 2 are 8 TB, each includes a 7314 GB HDFS data partition.

Drive 3 through 12 each contain a single HDFS or Oracle NoSQL Database data partition

Table 2-7 Oracle Big Data Appliance Server Disk Partitioning

Disks 1 and 2 (OS)					Disks 3 – 12 (Data)				
8 TB Drives:					8 TB Drives:				
Number	Start	End	Size	File	Number	Start	End	Size	File
system	Name	Flags			system	Name	Flags		
1	1049kB	500MB	499MB		1	1049kB	7864GB	7864GB	
ext4		primary	boot		ext4		primary		
2	500MB	501GB							
500GB				primary raid					
3	501GB	550GB	50.0GB	linux-					
swap(v1)	primary								
4	550GB	7864GB	7314GB						
ext4		primary							
4 TB Drives:					4 TB Drives:				
Number	Start	End	Size	File	Number	Start	End	Size	File
system	Name	Flags			system	Name	Flags		
1	1049kB	500MB	499MB		1	1049kB	4000GB	4000GB	
ext4		primary	boot		ext4		primary		
2	500MB	501GB							
500GB				primary raid					
3	501GB	560GB	59.5GB	linux-					
swap(v1)	primary								
4	560GB	4000GB	3440GB						
ext4		primary							

2.7.2 Critical and Noncritical CDH 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 Services Run on a Single-Rack CDH Cluster?"](#)

2.7.2.1 High Availability or Single Points of Failure?

Some services have high availability and automatic failover. Other services have a single point of failure. The following list summarizes the critical services:

- **NameNodes:** High availability with automatic failover

- **ResourceManagers:** High availability with automatic failover
- **MySQL Database:** Primary and backup databases are configured with replication of the primary database to the backup database. There is no automatic failover. If the primary database fails, the functionality of the cluster is diminished, but no data is lost.
- **Cloudera Manager:** The Cloudera Manager server runs on one node. If it fails, then Cloudera Manager functionality is unavailable.
- **Oozie server, Hive server, Hue server, and Oracle Data Integrator agent:** These services have no redundancy. If the node fails, then the services are unavailable.

2.7.2.2 Where Do the Critical Services Run?

[Table 2-8](#) identifies where the critical services run in a CDH cluster. These four nodes are described in more detail in the topics that follow.

Table 2-8 Critical Service Locations on a Single Rack

Node Name	Critical Functions
First NameNode	Balancer, Failover Controller, JournalNode, NameNode, Puppet Master, ZooKeeper
Second NameNode	Failover Controller, JournalNode, MySQL Backup Database, NameNode, ZooKeeper
First ResourceManager Node	Cloudera Manager Server, JobHistory, JournalNode, MySQL Primary Database, ResourceManager, ZooKeeper.
Second ResourceManager Node	Hive, Hue, Oozie, Solr, Oracle Data Integrator Agent, ResourceManager

In a single-rack cluster, the four critical nodes are created initially on the first four nodes. See "[Where Do the Services Run on a Single-Rack CDH Cluster?](#)"

In a multirack cluster, the Second NameNode and the Second ResourceManager nodes are moved to the first two nodes of the second rack. See "[Where Do the Services Run on a Multirack CDH Cluster?](#)".

2.7.3 First NameNode Node

If the first NameNode fails or goes offline (such as a restart), then the second NameNode 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 utility uses Puppet, and so you cannot install or reinstall the software if, for example, you must replace a disk drive elsewhere in the rack.

2.7.4 Second NameNode Node

If the second NameNode 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.

2.7.5 First ResourceManager Node

If the first ResourceManager node fails or goes offline (such as in a restart of the server where the node is running), then the second ResourceManager automatically takes over the distribution of MapReduce tasks to specific nodes across the cluster.

If the second ResourceManager is already active when the first ResourceManager becomes inaccessible, then it continues as ResourceManager, but without a backup. With only one ResourceManager, the cluster is vulnerable because it has lost the redundancy needed for automatic failover.

If the first ResourceManager node fails or goes offline (such as a restart), then the second ResourceManager automatically takes over to distribute MapReduce tasks to specific nodes across the cluster.

Alternatively, if the second ResourceManager is already active, it continues without a backup. With only one ResourceManager, 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](#)".
- **MySQL 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.

2.7.6 Second ResourceManager Node

If the second ResourceManager node fails, then the function of the ResourceManager either fails over to the first ResourceManager or continues there without a backup. However, the cluster has lost the redundancy needed for automatic failover if the first ResourceManager also fails.

These services are also disrupted:

- **Oracle Data Integrator Agent** This service supports Oracle Data Integrator, which is one of the Oracle Big Data Connectors. You cannot use Oracle Data Integrator when the ResourceManager node is down.
- **Hive:** Hive provides a SQL-like interface to data that is stored in HDFS. Oracle Big Data SQL and 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 ResourceManager node is down.
- **Oozie:** This workflow and coordination service runs on the ResourceManager node, and is unavailable when the node is down.

2.7.7 Noncritical CDH Nodes

The noncritical nodes 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 always maintain three copies. 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.

2.8 Managing a Hardware Failure

If a server starts failing, you must take steps to maintain the services of the cluster with as little interruption as possible. You can manage a failing server easily using the `bdaccli` utility, as described in the following procedures. One of the management steps is called decommissioning. *Decommissioning* stops all roles for all services, thereby preventing data loss. Cloudera Manager requires that you decommission a CDH node before retiring it.

When a noncritical node fails, there is no loss of service. However, when a critical node fails in a CDH cluster, services with a single point of failure are unavailable, as described in "[Effects of Hardware on Software Availability](#)". You must decide between these alternatives:

- Wait for repairs to be made, and endure the loss of service until they are complete.
- Move the critical services to another node. This choice may require that some clients are reconfigured with the address of the new node. For example, if the second ResourceManager node (typically node03) fails, then users must redirect their browsers to the new node to access Cloudera Manager.

You must weigh the loss of services against the inconvenience of reconfiguring the clients.

2.8.1 About Oracle NoSQL Database Clusters

Oracle NoSQL Database clusters do not have critical nodes. The storage nodes are replicated, and users can choose from three administrative processes on different nodes. There is no loss of services.

If the node hosting Mammoth fails (the first node of the cluster), then follow the procedure for reinstalling it in "[Prerequisites for Managing a Failing Node](#)".

To repair or replace any failing NoSQL node, follow the procedure in "[Managing a Failing Noncritical Node](#)".

2.8.2 Prerequisites for Managing a Failing Node

Ensure that you do the following before managing a failing or failed server, whether it is configured as a CDH node or an Oracle NoSQL Database node:

- Try restarting the services or rebooting the server.
- Determine whether the failing node is critical or noncritical.
- If the failing node is where Mammoth is installed:
 1. For a CDH node, select a noncritical node in the same cluster as the failing node.

For a NoSQL node, repair or replace the failed server first, and use it for these steps.

2. Upload the Mammoth bundle to that node and unzip it.
3. Extract all files from `BDAmammoth-version.run`, using a command like the following:

```
# ./BDAmammoth-ol6-4.0.0.run
```

Afterward, you must run all Mammoth operations from this node.

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

4. Follow the appropriate procedure in this section for managing a failing node.

Mammoth is installed on the first node of the cluster, unless its services were migrated previously.

2.8.3 Managing a Failing CDH Critical Node

Only CDH clusters have critical nodes.

To manage a failing critical node:

1. Log in as `root` to the node where Mammoth is installed.
2. Migrate the services to a noncritical node. Replace *node_name* with the name of the failing node, such as `bda1node02`.

```
bdacli admin_cluster migrate node_name
```

When the command finishes, *node_name* is decommissioned and its services are running on a previously noncritical node.

3. Announce the change to the user community, so that they can redirect their clients to the new critical node as required.
4. Repair or replace the failed server.
5. From the Mammoth node as `root`, reprovision the repaired or replaced server as a noncritical node. Use the same name as the migrated node for *node_name*, such as `bda1node02`:

```
bdacli admin_cluster reprovision node_name
```

6. If the failed node supported services like HBase or Impala, which Mammoth installs but does not configure, then use Cloudera Manager to reconfigure them on the new node.

2.8.4 Managing a Failing Noncritical Node

Use the following procedure to replace a failing node in either a CDH or a NoSQL cluster.

To manage a failing noncritical node:

1. Log in as `root` to the node where Mammoth is installed (typically `node01`).

2. Decommission the failing node. Replace *node_name* with the name of the failing node, such as `bda1node07`.

```
bdaccli admin_cluster decommission node_name
```

3. Repair or replace the failed server.

4. From the Mammoth node as `root`, recommission the repaired or replaced server. Use the same name as the decommissioned node for *node_name*, such as `bda1node07`:

```
bdaccli admin_cluster recommission node_name
```

See Also:

Oracle Big Data Appliance Owner's Guide for the complete `bdaccli` syntax

2.9 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](#)

2.9.1 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.

2.9.2 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
-

2.9.2.1 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.

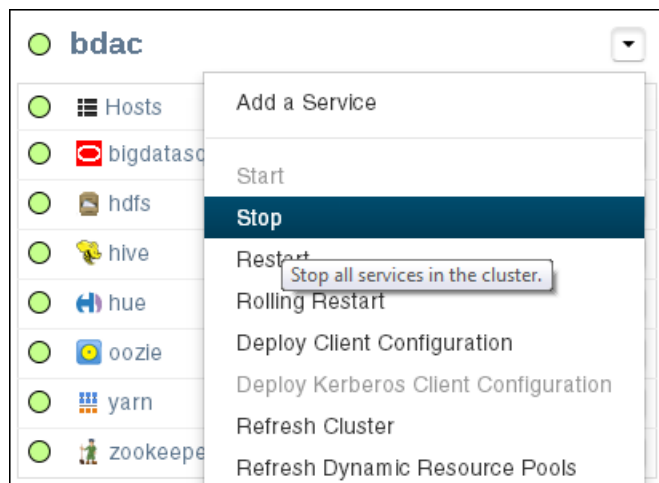
See "[Managing Operations Using Cloudera Manager](#)".

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



2.9.2.2 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.

2.9.2.3 Stopping Oracle Data Integrator Agent

If Oracle Data Integrator is used on the cluster:

1. Check the status of the Oracle Data Integrator agent:

```
# dcli -C service odi-agent status
```

2. Stop the Oracle Data Integrator agent, if it is running:

```
# dcli -C service odi-agent stop
```

3. Ensure that the Oracle Data Integrator agent stopped running:

```
# dcli -C service odi-agent status
```

2.9.2.4 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 sharedir
192.0.2.1: bdalnode03.example.com:/opt/exportdir on /opt/sharedir type nfs
(rw,tcp,soft,intr,timeo=10,retrans=10,addr=192.0.2.3)
192.0.2.2: bdalnode03.example.com:/opt/exportdir on /opt/sharedir type nfs
(rw,tcp,soft,intr,timeo=10,retrans=10,addr=192.0.2.3)
192.0.2.3: /opt/exportdir on /opt/sharedir 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/sharedir
```

3. Dismount any custom NFS directories.

2.9.2.5 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
```

2.9.2.6 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.

2.9.3 Starting Oracle Big Data Appliance

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

2.9.3.1 Powering Up Oracle Big Data Appliance

1. Switch on all 12 breakers on both PDUs.
2. Allow 4 to 5 minutes for Oracle ILOM and the Linux operating system to start on the servers.

If the servers do not start automatically, then you can start them 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. Use whichever interface you prefer.

For example, you can log in to 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
```

2.9.3.2 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.

See "[Managing Operations Using Cloudera Manager](#)".

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).

2.9.3.3 Starting Oracle Data Integrator Agent

If Oracle Data Integrator is used on this cluster:

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]
```

2.10 Managing Oracle Big Data SQL

Oracle Big Data SQL is registered with Cloudera Manager as an add-on service. You can use Cloudera Manager to start, stop, and restart the Oracle Big Data SQL service or individual role instances, the same way as a CDH service.

Cloudera Manager also monitors the health of the Oracle Big Data SQL service, reports service outages, and sends alerts if the service is not healthy.

2.10.1 Adding and Removing the Oracle Big Data SQL Service

Oracle Big Data SQL is an optional package that can be installed on Oracle Big Data Appliance. You can use Cloudera Manager to determine if Oracle Big Data SQL is installed, but cannot use it to add or remove the service from a CDH cluster on Oracle Big Data Appliance.

A version of this software is included in the Mammoth bundle for each Oracle Big Data Appliance release. The bundled version can be installed with the other client software during the initial software installation or an upgrade. A more recent version of Oracle Big Data SQL than the one included with Oracle Big Data Appliance may be available. The Preface of the *Oracle Big Data Appliance Owner's Guide* identifies the changes in each release, including which version of Oracle Big Data SQL is bundled with the release. See the *Oracle Big Data SQL Master Compatibility Matrix* (Doc ID 2119369.1) in [My Oracle Support](#) for up-to-date information on other versions of Oracle Big Data SQL that are compatible with your current release of Oracle Big Data Appliance.

To install the version of Oracle Big Data SQL that is included with Oracle Big Data Appliance, log in to the server where Mammoth is installed (usually the first node of the cluster). use the following commands in the `bdaccli` utility:

- To enable Oracle Big Data SQL

```
bdaccli enable big_data_sql
```

- To disable Oracle Big Data SQL:

```
bdaccli disable big_data_sql
```

To add or remove other versions of Oracle Big Data SQL, see the installation guide for that release.

A separate license is required for Oracle Big Data SQL. The license is not included with the Oracle Big Data Appliance license agreement.

2.10.2 Allocating Resources to Oracle Big Data SQL

You can modify the property values in a Linux kernel Control Group (Cgroup) to reserve resources for Oracle Big Data SQL.

To modify the resource management configuration settings:

1. Log in as admin to Cloudera Manager.
2. On the Home page, click **bigdatasql** from the list of services.
3. On the bigdatasql page, click **Configuration**.
4. Under Category, expand BDS Server Default Group and click **Resource Management**.
5. Modify the values of the following properties as required:
 - Cgroup CPU Shares
 - Cgroup I/O Weight
 - Cgroup Memory Soft Limit

- Cgroup Memory Hard Limit

See the Description column on the page for guidelines.

6. Click **Save Changes**.
7. From the Actions menu, click **Restart**.

Figure 2-10 shows the bigdatasql service configuration page.

Figure 2-10 Modifying the Cgroup Settings for Oracle Big Data SQL

The screenshot shows the 'bigdatasql' Configuration page. The 'Configuration' tab is selected. On the left, a sidebar lists categories: Service-Wide, BDS Server Default Group, and Resource Management (which is highlighted). Under 'Resource Management', there are sub-items: Advanced, Monitoring, Performance, and Resource Management. The main area displays a table of properties with their current values and reset links.

Category	Property	Value
Service-Wide	Cgroup CPU Shares cpu.shares	500 Reset to the default value: 1024 ↗
	Cgroup I/O Weight blkio.weight	250 Reset to the default value: 500 ↗
BDS Server Default Group	Cgroup Memory Soft Limit memory.soft_limit_in_bytes	-1 MiB Reset to the default value: -1 MiB ↗
	Cgroup Memory Hard Limit memory.limit_in_bytes	-1 MiB Reset to the default value: -1 MiB ↗

See Also:

["Allocating Resources Among Services"](#).

2.11 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](#)
- [About Fine-Grained Authorization](#)

- [Port Numbers Used on Oracle Big Data Appliance](#)
- [About Puppet Security](#)

2.11.1 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 2-9](#) 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 2-9 Operating System Users and Groups

User Name	Group	Used By	Login Rights
flume	flume	Apache Flume parent and nodes	No
hbase	hbase	Apache HBase processes	No
hdfs	hadoop	NameNode , DataNode	No
hive	hive	Hive metastore and server processes	No
hue	hue	Hue processes	No
mapred	hadoop	ResourceManager, NodeManager, 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	Apache Sqoop metastore	No
svctag		Auto Service Request	No

Table 2-9 (Cont.) Operating System Users and Groups

User Name	Group	Used By	Login Rights
zookeeper	zookeeper	ZooKeeper processes	No

2.11.2 About User Authentication

Oracle Big Data Appliance supports Kerberos security as a software installation option. See [Supporting User Access to Oracle Big Data Appliance](#) for details about setting up clients and users to access a Kerberos-protected cluster.

2.11.3 About Fine-Grained Authorization

The typical authorization model on Hadoop is at the HDFS file level, such that users either have access to all of the data in the file or none. In contrast, Apache Sentry integrates with the Hive and Impala SQL-query engines to provide fine-grained authorization to data and metadata stored in Hadoop.

Oracle Big Data Appliance automatically configures Sentry during software installation, beginning with Mammoth utility version 2.5.

See Also:

- Cloudera Manager Help
 - *Managing Clusters with Cloudera Manager* at <https://www.cloudera.com/content/cloudera-content/cloudera-docs/CM4Ent/latest/Cloudera-Manager-Managing-Clusters/Managing-Clusters-with-Cloudera-Manager.html>
-

2.11.4 About HDFS Transparent Encryption

HDFS Transparent Encryption protects Hadoop data that is at rest on disk. After HDFS Transparent Encryption is enabled for a cluster on Oracle Big Data Appliance, data writes and reads to encrypted *zones* (HDFS directories) on the disk are automatically encrypted and decrypted. This process is “transparent” because it is invisible to the application working with the data.

HDFS Transparent Encryption does not affect user access to Hadoop data, although it can have a minor impact on performance.

HDFS Transparent Encryption is an option that you can select during the initial installation of the software by the Mammoth utility. You can also enable or disable HDFS Transparent Encryption at any time by using the `bdaccli` utility. Note that HDFS Transparent Encryption can be installed only on a Kerberos-secured cluster.

Oracle recommends that you set up the Navigator Key Trustee (the service that manages keys and certificates) on a separate server, external to the Oracle Big Data Appliance.

See the following MOS documents at [My Oracle Support](#) for instructions on installing and enabling HDFS Transparent Encryption.

Title	MOS Doc ID
<i>How to Setup Highly Available Active and Passive Key Trustee Servers on BDA V4.4 Using 5.5 Parcels</i>	2112644.1 Installing using parcels as described in this MOS document is recommended over package-based installation. See Cloudera's comments on Parcels .
<i>How to Enable/Disable HDFS Transparent Encryption on Oracle Big Data Appliance V4.4 with bdaccli</i>	2111343.1
<i>How to Create Encryption Zones on HDFS on Oracle Big Data Appliance V4.4</i>	2111829.1

Note:

If either HDFS Transparent Encryption or Kerberos is disabled, data stored in the HDFS Transparent Encryption zones in the cluster will remain encrypted and therefore inaccessible. To restore access to the data, re-enable HDFS Transparent Encryption using the same key provider.

See Also:

Cloudera documentation about HDFS at-rest encryption at <http://www.cloudera.com> for more information about managing files in encrypted zones.

2.11.5 About HTTPS/Network Encryption

HTTPS Network/Encryption on the Big Data Appliance has two components :

- **Web Interface Encryption**

Configures HTTPS for the following web interfaces: Cloudera Manager, Oozie, and HUE. This encryption is now enabled automatically in new Mammoth installations. For current installations it can be enabled via the bdaccli utility. This feature does not require that Kerberos is enabled.

- **Encryption for Data in Transit and Services**

There are two subcomponents to this feature. Both are options that can be enabled in the Configuration Utility at installation time or enabled/disabled using the bdaccli utility at any time. Both require that Kerberos is enabled.

- **Encrypt Hadoop Services**

This includes SSL encryption for HDFS, MapReduce, and YARN web interfaces, as well as encrypted shuffle for MapReduce and YARN. It also enable authentication for access to the web consoles for the MapReduce, and YARN roles.

- **Encrypt HDFS Data Transport**

This option will enable encryption of data transferred between DataNodes and clients, and among DataNodes.

HTTPS/Network Encryption is enabled and disabled on a per cluster basis. The Configuration Utility described in the *Oracle Big Data Appliance Owner's Guide*, includes settings for enabling encryption for Hadoop Services and HDFS Data Transport when a cluster is created. The `bdacli` utility reference pages (also in the *Oracle Big Data Appliance Owner's Guide*) provide HTTPS/Network Encryption command line options.

See Also:

[Supporting User Access to Oracle Big Data Appliance](#) for an overview of how Kerberos is used to secure CDH clusters.

[About HDFS Transparent Encryption](#) for information about Oracle Big Data Appliance security for Hadoop data at-rest.

Cloudera documentation at <http://www.cloudera.com> for more information about HTTPS communication in Cloudera Manager and network-level encryption in CDH.

2.11.5.1 Configuring Web Browsers to use Kerberos Authentication

If web interface encryption is enabled, each web browser accessing an HDFS, MapReduce, or YARN-encrypted web interface must be configured to authenticate with Kerberos. Note that this is not necessary for the Cloudera Manager, Oozie, and Hue web interfaces, which do not require Kerberos.

The following are the steps to configure Mozilla Firefox¹, Microsoft Internet Explorer², and Google Chrome³ for Kerberos authentication.

To configure Mozilla Firefox:

1. Enter `about:config` in the Location Bar.
2. In the **Search** box on the `about:config` page, enter: `network.negotiate-auth.trusted-uris`
3. Under Preference Name, double-click the `network.negotiate-auth.trusted-uris`.
4. In the **Enter string value** dialog, enter the hostname or the domain name of the web server that is protected by Kerberos. Separate multiple domains and hostnames with a comma.

To configure Microsoft Internet Explorer:

1. Configure the Local Intranet Domain:
 - a. Open Microsoft Internet Explorer and click the Settings "gear" icon in the top-right corner. Select **Internet options**.
 - b. Select the **Security** tab.
 - c. Select the **Local intranet** zone and click **Sites**.

¹ Mozilla Firefox is a registered trademark of the Mozilla Foundation.

² Microsoft Internet Explorer is a registered trademark of Microsoft Corporation.

³ Google Chrome is a registered trademark of Google Inc

- d. Make sure that the first two options, `Include all local (intranet) sites not listed in other zones` and `Include all sites that bypass the proxy server` are checked.
 - e. Click **Advanced** on the `Local intranet` dialog box and, one at a time, add the names of the Kerberos-protected domains to the list of websites.
 - f. Click **Close**.
 - g. Click **OK** to save your configuration changes, then click **OK** again to exit the Internet Options panel.
2. Configure Intranet Authentication for Microsoft Internet Explorer:
 - a. Click the **Settings** "gear" icon in the top-right corner. Select `Internet Options`.
 - b. Select the **Security** tab.
 - c. Select the `Local Intranet` zone and click the `Custom level...` button to open the `Security Settings - Local Intranet Zone` dialog box.
 - d. Scroll down to the **User Authentication** options and select `Automatic logon only in Intranet zone`.
 - e. Click **OK** to save your changes.

To configure Google Chrome:

If you are using Microsoft Windows, use the Control Panel to navigate to the Internet Options dialogue box. Configuration changes required are the same as those described above for Microsoft Internet Explorer.

On⁴⁴ or on Linux, add the `--auth-server-whitelist` parameter to the `google-chrome` command. For example, to run Chrome from a Linux prompt, run the `google-chrome` command as follows

```
google-chrome --auth-server-whitelist = "hostname/domain"
```

Note: On Microsoft Windows, the Windows user must be an user in the Kerberos realm and must possess a valid ticket. If these requirements are not met, an HTTP 403 is returned to the browser upon attempt to access a Kerberos-secured web interface.

2.11.6 Port Numbers Used on Oracle Big Data Appliance

[Table 2-10](#) 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.

⁴ Mac OS is a registered trademark of Apple, Inc.

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-docs/CM5/latest/Cloudera-Manager-Installation-Guide/cm5ig_config_ports.html

Table 2-10 Oracle Big Data Appliance Port Numbers

Service	Port
Automated Service Monitor (ASM)	30920
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

2.11.7 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.

2.12 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](#)

2.12.1 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.
- **Hive DDL:** Who makes Hive database changes.
- **MapReduce:** Who runs MapReduce jobs that correspond to file access.
- **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-technologies/audit-vault-and-database-firewall/overview/index.html>

2.12.2 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.

You can also add the plug-in at a later time using `bdaccli`. See *Oracle Big Data Appliance Owner's Guide*.

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

2.12.3 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 2-11 shows the beginning of an activity report, which records access to a Hadoop sequence file.

Figure 2-11 Activity Report in Audit Vault Server

All Activity Report	
Report View	< Row 0 of <input type="checkbox"/> Exclude Null Values <input type="checkbox"/> Displayed Columns
<div>Secured Target</div> <div> Secured Target Name bda1node02 Secured Target Type Oracle BDA Hadoop Cluster Service Name Policy Name Class Other </div>	
<div>Event</div> <div> Server Time 8/20/2013 3:09:51 PM Event Time 8/14/2013 9:28:06 AM User Name mapred(AUTH:SIMPLE) Event Status SUCCESS Error Code Error Message Event Name ACCESS Command Class SELECT Action Taken Threat Severity Log Cause Location Audit File </div>	
<div>Target</div> <div> Target Type Target Object /tmp/mapred/system/seq-0000000000003 Target Owner </div>	

See Also:

Oracle Audit Vault and Database Firewall Auditor's Guide

2.13 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.
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.

See Also:

My Oracle Support Note 549180.1 at
<http://support.oracle.com>

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 use 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](#)

3.1 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. Oracle Big Data Appliance support two forms of Kerberos Hadoop security: MIT Kerberos and Microsoft Active Directory Kerberos.

CDH provides these securities when configured to use Kerberos:

- The CDH master nodes, NameNodes, and JournalNodes 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.

Oracle Big Data Appliance provides the ability to configure Kerberos security directly using a Microsoft Active Directory (AD) server for Kerberos support (as supported by Cloudera Manager).

You have the option of enabling either form of Kerberos as part of the Mammoth configuration. You can also enable or disable Kerberos later through the `bdaccli` utility.

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
<code>kinit <i>userid@realm</i></code>	Obtains a Kerberos ticket.
<code>klist</code>	Lists a Kerberos ticket if you have one already.
<code>kdestroy</code>	Invalidates a ticket before it expires.
<code>kpasswd <i>userid@realm</i></code>	Changes your password.

See Also:

- MIT Kerberos Documentation at <http://web.mit.edu/kerberos/krb5-latest/doc/>
 - *CDH5 Security Guide* at <http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH5/latest/CDH5-Security-Guide/CDH5-Security-Guide.html>
 - If you choose to enable Active Directory Kerberos, either with Mammoth or with the `bdacli` utility, first read MOS (My Oracle Support) documents 2029378.1 and 2013585.1. These documents explain required preliminary steps and provide important information on known issues.
-

3.2 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

3.2.1 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 CDH5. See the *Cloudera CDH5 Installation Guide* at

<http://www.cloudera.com/content/www/en-us/documentation/cdh/5-0-x/CDH5-Installation-Guide/CDH5-Installation-Guide.html>

- The client system must run Oracle JDK 1.7.0_25 or later.

To verify the version, use this command:

```
$ java -version
java version "1.7.0_65"
Java(TM) SE Runtime Environment (build 1.7.0_65-b17)
Java HotSpot(TM) 64-Bit Server VM (build 24.65-b04, mixed mode)
```

3.2.2 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 an earlier CDH version at
http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH5/latest/CDH5-Installation-Guide/cdh5ig_to_cdh5_upgrade.html?scroll=topic_6_3_4_unique_1
3. Copy the CDH software from any node in a CDH cluster on Oracle Big Data Appliance. For example, the following file contains the CDH 5.8.0 software:
`/opt/oss/src/CDH/5.8.0-ol5/hadoop-2.6.0-cdh5.8.0.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-2.6.0-cdh5.8.0` in the current directory:
`tar -xvzf hadoop-2.6.0-cdh5.8.0.tar.gz`
5. Configure the CDH client. See "[Configuring a CDH Client for an Unsecured Cluster](#)."

3.2.3 Configuring a CDH Client for an Unsecured Cluster

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

The commands in this procedure that reference `HADOOP_HOME` are used to support older Hadoop clients that require this environment variable. The cluster uses YARN

(MRv2) and does not use `HADOOP_HOME`. If no older clients access the cluster, then you can omit these commands.

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 `yarn-conf.zip`.

```
$ wget -O yarn-conf.zip http://bda01node03.example.com:7180/cmfs/services/3/
client-config
```

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

```
$ unzip yarn-config.zip
Archive:  yarn-config.zip
  inflating: yarn-conf/hadoop-env.sh
  inflating: yarn-conf/hdfs-site.xml
  inflating: yarn-conf/core-site.xml
  inflating: yarn-conf/mapred-site.xml
  inflating: yarn-conf/log4j.properties
  inflating: yarn-conf/yarn-site.xml
```

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

3. Set the symbolic links:

```
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
```

4. 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=$HADOOP_HOME
export HADOOP_CONF_DIR=yarn-conf-dir
export JAVA_HOME=/usr/java/version
alias hadoop=$HADOOP_HOME/bin/hadoop
alias hdfs=$HADOOP_HOME/../../../../bin/hdfs
```

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

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

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

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

3.2.4 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>

Java 8: <http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html>

b. Decompress the downloaded file. This example unzips JCE-8:

```
$ unzip UnlimitedJCEPolicyJDK8.zip
Archive:  UnlimitedJCEPolicyJDK8.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](#)."

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](#)."

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](#)."

3.2.5 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 6 items
drwxr-xr-x - jdoe      hadoop      0 2014-04-03 00:08 /user/jdoe
drwxrwxrwx - mapred    hadoop      0 2014-04-02 23:25 /user/history
drwxr-xr-x - hive      supergroup  0 2014-04-02 23:27 /user/hive
drwxrwxr-x - impala    impala      0 2014-04-03 10:45 /user/impala
drwxr-xr-x - oozie     hadoop      0 2014-04-02 23:27 /user/oozie
drwxr-xr-x - oracle    hadoop      0 2014-04-03 11:49 /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.

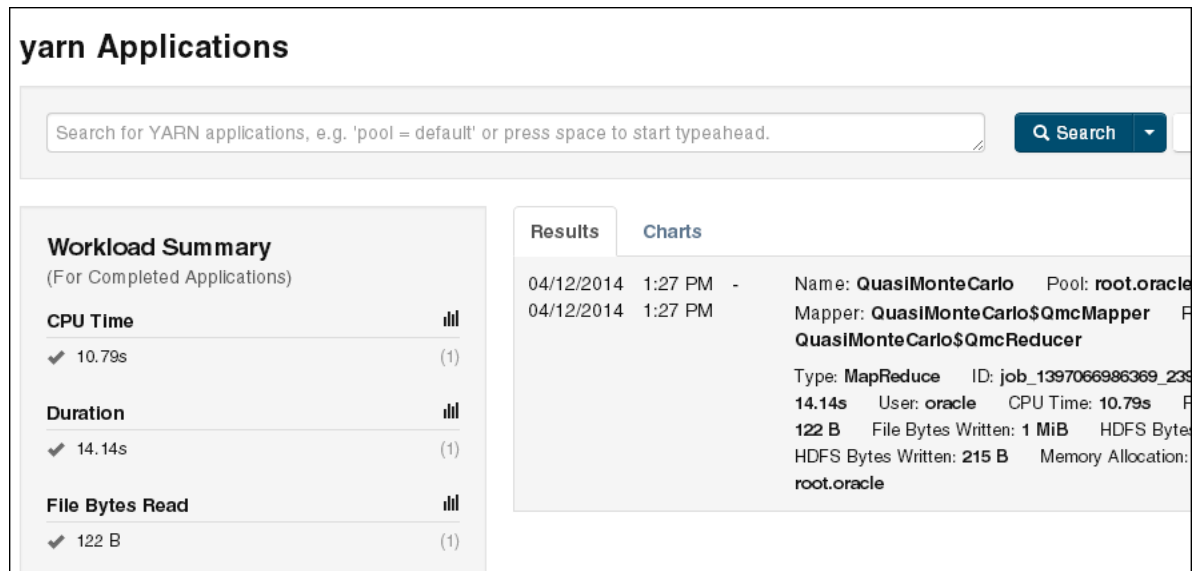
The following example calculates the value of *pi*:

```
$ hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-mapreduce-examples-*.jar pi 10 1000000
Number of Maps = 10
Samples per Map = 1000000
Wrote input for Map #0
Wrote input for Map #1
.
.
.
Job Finished in 12.403 seconds
Estimated value of Pi is 3.141584400000000000000
```

4. Use Cloudera Manager to verify that the job ran on Oracle Big Data Appliance instead of the local system. Select **mapreduce Jobs** 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 YARN Job in Cloudera Manager



3.3 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.](#)"
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.example.com:7180/cmfs/services/5/
client-config
Length: 1283 (1.3K) [application/zip]
Saving to: 'hive-conf.zip'
100%[=====>] 1,283      --.-K/s   in 0.001s
2016-05-15 08:19:06 (2.17 MB/s) - 'hive-conf.zip' saved [1283/1283]
```

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/cdh5/cdh/5/hive-<version>-
cdh5.<version>.tar.gz
Length: 49637596 (47M) [application/x-gzip]
Saving to: 'hive-<version>-cdh5.<version>.tar.gz'
100%[=====>] 49,637,596   839K/s   in 47s
```

```
2016-05-15 08:22:18 (1.02 MB/s) - `hive-<version>-cdh5.<version>.tar.gz' saved
[49637596/49637596]
```

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-0.12.0-cdh5.0.0`:

```
$ tar -xvzf hive-<version>-cdh5.<version>.tar.gz
hive-<version>-cdh5.<version>/
hive-<version>-cdh5.<version>/examples/
.
.
.
```

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-<version>-cdh5.<version>.jar!/hive-log4j.properties
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
```

3.4 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 \(Optional\)](#)

3.4.1 Creating Hadoop Cluster Users

When creating user accounts, define them as follows:

- To run MapReduce jobs, users must either be in the `hadoop` group or be granted the equivalent permissions.

- To create and modify tables in Hive, users must either be in the `hive` group or be granted the equivalent permissions.
- To create Hue users, open Hue in a browser and click the User Admin icon. See ["Using Cloudera Hue to Interact With Hadoop."](#)

3.4.1.1 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 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
bdalnode01: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),127(hive),123(hadoop)
bdalnode02: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),123(hadoop),127(hive)
bdalnode03: uid=1001(jdoe) gid=1003(jdoe) groups=1003(jdoe),123(hadoop),127(hive)
.
.
.
# dcli ls /home | grep jdoe
bdalnode01: jdoe
```

```
bdalnode02: jdoe
bdalnode03: jdoe
```

[Example 3-1](#) creates a user named `jdoe` with a primary group of `hadoop` and an addition group of `hive`.

3.4.1.2 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
```

3.4.2 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](#)".
2. Confirm that the user does not have a password:

```
# dcli passwd -s user_name
bdalnode01.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
bdalnode02.example.com: jdoe NP 2013-01-22 0 99999 7 -1 (Empty password.)
bdalnode03.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
bdalnode01.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
bdalnode02.example.com: jdoe PS 2013-01-24 0 99999 7 -1 (Password set, MD5
crypt.)
bdalnode03.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.
-

3.5 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.

3.5.1 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
```

3.5.2 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 Operations Using Cloudera Manager](#)".
2. On the Home page under Status, click **hdfs**.
3. On the `hdfs` page, click the Configuration subtab, and then select **View and Edit**.
4. Search for or scroll down to the Filesystem Trash Interval property under NameNode Default Group. 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

The screenshot shows the Cloudera Manager interface for the HDFS configuration. The top navigation bar includes 'Status', 'Instances', 'Commands', 'Configuration' (selected), 'Audits', 'Charts Library', and 'File Browser'. Below the navigation bar, there are tabs for 'Cache Statistics', 'Replication', and 'Web UI'. The main content area is titled 'Configuration' and features a search bar with 'trash' entered. A green banner indicates '3 validation checks'. Below this, a table lists configuration properties:

Category	Property	Value	Description
Gateway Default Group	Use Trash	<input checked="" type="checkbox"/> Reset to the default value: false ↩ HDFS Trash is enabled.	Move deleted files to the trash so that they can be recovered if necessary. This client side configuration takes effect only if the HDFS service-wide trash is disabled (NameNode Filesystem Trash Interval set to 0) and is ignored otherwise. The trash is not automatically emptied when enabled with this configuration.
NameNode Default Group	Filesystem Trash Interval fs.trash.interval	1 day(s) default value Trash checkpointing is on	Number of minutes between trash checkpoints. Also controls the number of minutes after which a trash checkpoint directory is deleted. To disable the trash feature, enter 0.

3.5.3 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.

3.5.3.1 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 Operations Using Cloudera Manager](#)".
2. On the Home page under Status, click **hdfs**.
3. On the hdfs page, click the Configuration subtab, and then select **View and Edit**.
4. Search for or scroll down to the Filesystem Trash Interval property under NameNode Default Group. 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**.

3.5.3.2 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."](#)

To disable the trash facility for local HDFS clients:

1. Open Cloudera Manager. See ["Managing Operations Using Cloudera Manager"](#).
2. On the Home page under Status, click **hdfs**.
3. On the hdfs page, click the **Configuration** subtab, and then select **View and Edit**.
4. Search for or scroll down to the Filesystem Trash Interval property under Gateway Default Group. See [Figure 3-2](#).
5. Search for or scroll down to the Use Trash property under Client Settings. See [Figure 3-2](#).
6. Deselect the Use Trash check box.
7. Click **Save Changes**. This setting is used to configure all new HDFS clients downloaded to Oracle Big Data Appliance.
8. Open a connection as `root` to a node in the cluster.
9. Deploy the new configuration:

```
dcli -C bdagetcclientconfig
```

3.5.3.3 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."](#) 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. Set the trash interval to zero:

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

```
        <value>0</value>  
    </property>
```

4. Save the file.

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 for Optimizing Communications](#)
- [Specifying the InfiniBand Connections to Oracle Big Data Appliance](#)
- [Specifying the InfiniBand Connections to Oracle Exadata Database Machine](#)
- [Enabling SDP on Exadata Database Nodes](#)
- [Creating an SDP Listener on the InfiniBand Network](#)

4.1 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.

4.1.1 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](#)."

See Also :

Oracle Big Data Connectors User's Guide for information about Oracle SQL Connector for HDFS

4.1.2 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

4.2 Prerequisites for Optimizing Communications

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
-
-

4.3 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](#)."
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      bdalnode01.example.com  bdalnode01
192.168.8.2      bdalnode02.example.com  bdalnode02
192.168.8.3      bdalnode03.example.com  bdalnode03
192.168.8.4      bdalnode04.example.com  bdalnode04
192.168.8.5      bdalnode05.example.com  bdalnode05
192.168.8.6      bdalnode06.example.com  bdalnode06
```

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 bdalnode01.example.com
PING bdalnode01.example.com (192.168.8.1) 56(84) bytes of data.
64 bytes from bdalnode01.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://bdalnode05.example.com/testdir/
```

4.4 Specifying the InfiniBand Connections to Oracle Exadata Database Machine

You can 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 `dm01cel08` storage server:

```
$ scp mydata.json oracle@dm01cel08-priv.example.com:mybigdata.json
oracle@dm01cel08-priv.example.com's password: password
```

4.5 Enabling SDP on Exadata Database Nodes

SDP improves the performance of client applications that run on Oracle Big Data Appliance and push large data loads to Oracle Database on Oracle Exadata Database Machine.

The following procedure describes how to enable SDP on the database nodes in an Oracle Exadata Database Machine running Oracle Linux. You must also configure your application on a job-by-job basis to use SDP.

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.

To specify SDP protocol for a load job:

1. Add JVM options to the `HADOOP_OPTS` environment variable to enable JDBC SDP export:

```
HADOOP_OPTS="-Doracle.net.SDP=true -Djava.net.preferIPv4Stack=true"
```

2. In either the Hadoop command or the configuration file for the job, set the `mapred.child.java.opts` configuration property to enable the child task JVMs for SDP.

For example, use these options in the command line for a MapReduce job:

```
-D mapred.child.java.opts="-Doracle.net.SDP=true -Djava.net.preferIPv4Stack=true"
```

3. Configure standard Ethernet communications for the job.

For example, Oracle Loader for Hadoop reads the value of the `oracle.hadoop.loader.connection.url` property from a job configuration file. The value has this syntax:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=
  (ADDRESS=(PROTOCOL=TCP)(HOST=hostName)(PORT=portNumber)))
(CONNECT_DATA=(SERVICE_NAME=serviceName)))
```

Replace *hostName*, *portNumber*, and *serviceName* with the appropriate values to identify the SDP listener on your Oracle Exadata Database Machine.

4. Configure the Oracle listener on Exadata to support the SDP protocol and bind it to a specific port address (such as 1522).

For example, Oracle Loader for Hadoop reads the value of the `oracle.hadoop.loader.connection.oci_url` property from a job configuration file. The value has this syntax:

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=SDP)
  (HOST=hostName)(PORT=portNumber))
(CONNECT_DATA=(SERVICE_NAME=serviceName)))
```

4.6 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:

These instructions apply to Exadata V2, X2-2, and X3-2 nodes running Oracle Linux 5. Document 1580584.1 in [My Oracle Support](#) provides instructions for these same systems as well as for X4-2, X5-2, and X6-2 nodes running Oracle Linux 6.

This example below 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/12.1.0.1/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/12.1.0.1/bin/crsctl stat res -t | grep net
ora.net1.network
ora.net2.network -- Output indicating new Network resource
```

or

```
# /u01/app/grid/product/12.1.0.1/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
```

Part II

Oracle Big Data Appliance Software

This part describes the software that is available only on Oracle Big Data Appliance. It contains the following chapters:

- [Optimizing MapReduce Jobs Using Perfect Balance](#)

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's 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](#)

5.1 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.

5.1.1 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.

5.1.2 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](#)", 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](#)".

5.1.3 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.

5.2 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](#)".

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.

5.3 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](#)."
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](#)."
4. Set the following variables using the Bash `export` command:
 - `BALANCER_HOME`: Set to the Perfect Balance installation directory, such as `/opt/oracle/orabalancer-<version>-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`: PREPEND `${BALANCER_HOME}/jlib/orabalancer-<version>.jar` and `${BALANCER_HOME}/jlib/commons-math-2.2.jar` to the existing value. Also prepend the JAR files for your application.

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

- `HADOOP_USER_CLASSPATH_FIRST`: Set to `true` to enable Perfect Balance; API mode does not need this variable.

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's Reducer Load](#)."

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](#)."

7. Run the job using Perfect Balance.

See "[Running a Balanced MapReduce Job Using Perfect Balance](#)."

8. Use the Job Analyzer report to evaluate the effectiveness of using Perfect Balance. See "[Reading the Job Analyzer Report](#)."
9. Modify the job configuration properties as desired before rerunning the job with Perfect Balance. See "[About Configuring Perfect Balance](#)."

5.4 Analyzing a Job's 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](#)

5.4.1 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.

5.4.1.1 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 when you want to analyze a job that previously ran.
- **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 want to analyze the current job.

5.4.2 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 `oracle.hadoop.balancer.application_id` to the job ID of the job you want to analyze.

You can obtain the job ID from the YARN Resource Manager web interface. Click the application ID of a job, and then click **Tracking URL**. The job ID typically begins with "job_".

Alternately, if you already ran Perfect Balance or Job Analyzer on this job, you can read the job ID from the `application_id` file generated by Perfect Balance in its report directory (`outdir/_balancer` by default).

3. Run the Job Analyzer utility as described in "[Job Analyzer Utility Syntax](#)."
4. View the Job Analyzer report in a browser.

5.4.2.1 Job Analyzer Utility Example

[Example 5-1](#) runs a script that sets the required variables, uses the MapReduce job logs for a job with an application ID of `job_1396563311211_0947`, 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.

To run this example on a YARN cluster, replace the application ID with the application ID of the job. The application ID of the job looks like this example: `job_1396563311211_0947`.

Example 5-1 Running the Job Analyzer Utility

```
$ cat runja.sh

BALANCER_HOME=/opt/oracle/orabalancer-<version>-h2
export HADOOP_CLASSPATH=${BALANCER_HOME}/jlib/orabalancer-<version>.jar:${
{BALANCER_HOME}/jlib/commons-math-2.2.jar:$HADOOP_CLASSPATH
export HADOOP_USER_CLASSPATH_FIRST=true

# Command on YARN cluster
hadoop jar orabalancer-<version>.jar oracle.hadoop.balancer.tools.JobAnalyzer \
-D oracle.hadoop.balancer.application_id=job_1396563311211_0947

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

5.4.2.2 Job Analyzer Utility Syntax

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

```
hadoop jar ${BALANCER_HOME}/jlib/orabalancer-<version>.jar
oracle.hadoop.balancer.tools.JobAnalyzer \
-D oracle.hadoop.balancer.application_id=job_number \
[ja_report_path]
```

job_number

The application ID previously assigned to the job.

ja_report_path

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

5.4.3 Running Job Analyzer Using Perfect Balance

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

- [Running Job Analyzer Using Perfect Balance](#)
- [Collecting Additional Metrics](#)

5.4.3.1 Running Job Analyzer Using 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](#)."
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](#)."
4. Decide which additional configuration properties to set, if any.

See "[Perfect Balance Configuration Property Reference](#)."

5. Run the job.

Example 5-2 Running Job Analyzer with Perfect Balance

```
$ cat ja_nobalance.sh

# set up perfect balance
BALANCER_HOME=/opt/oracle/orabalancer-<version>-h2
export HADOOP_CLASSPATH=${BALANCER_HOME}/jlib/orabalancerclient-<version>.jar:${BALANCER_HOME}/jlib/orabalancer-<version>.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 mapreduce.input.fileinputformat.inputdir=jdoe_application/input \
-D mapreduce.output.fileoutputformat.outputdir=jdoe_nobal_outdir \
-D mapreduce.job.name=nobal \
-D mapreduce.job.reduces=10 \
-D oracle.hadoop.balancer.autoBalance=false \
-D oracle.hadoop.balancer.autoAnalyze=REDUCER_REPORT \
```

```

-conf application_config_file.xml

$ sh ja_nobalance.sh
14/04/14 14:52:42 INFO input.FileInputFormat: Total input paths to process : 5
14/04/14 14:52:42 INFO mapreduce.JobSubmitter: number of splits:5
14/04/14 14:52:42 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1397066986369_3478
14/04/14 14:52:43 INFO impl.YarnClientImpl: Submitted application
application_1397066986369_3478
.
.
.
File Input Format Counters
Bytes Read=112652976
File Output Format Counters
Bytes Written=384974202

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

```

[Example 5-2](#) 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.

5.4.3.2 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.

Job Analyzer might recommend key load coefficients for the Perfect Balance key load model, based on its analysis of the job load. To use these recommended coefficients when running a job with Perfect Balance, set the [oracle.hadoop.balancer.linearKeyLoad.feedbackDir](#) property to the directory containing the Job Analyzer report of a previously analyzed run of the job.

If the report contains recommended coefficients, then Perfect Balance automatically uses them. If Job Analyzer encounters an error while collecting the additional metrics, then the report does not contain the additional metrics.

Use the `feedbackDir` property when you do not know the values of the load model coefficients for a job, but you have the Job Analyzer output from a previous run of the job. Then you can set the value of `feedbackDir` to the directory where that output is stored. The values recommended from those files typically perform better than the Perfect Balance default values, because the recommended values are based on an analysis of your job's load.

Alternately, if you already know good values of the load model coefficients for your job, you can set the load model properties:

- [oracle.hadoop.balancer.linearKeyLoad.byteWeight](#)
- [oracle.hadoop.balancer.linearKeyLoad.keyWeight](#)

- [oracle.hadoop.balancer.linearKeyLoad.rowWeight](#)

Running the job with these coefficients results in a more balanced job.

5.4.4 Reading the Job Analyzer Report

Job Analyzer writes its report in two formats: HTML for you, and XML for Perfect Balance. You can open the report in a browser, either directly in HDFS or after copying it to the local file system

To open a Job Analyzer report in HDFS in a browser:

1. Open the HDFS web interface on port 50070 of a NameNode node (node01 or node02), using a URL like the following:

```
http://bdalnode01.example.com:50070
```

2. From the Utilities menu, choose **Browse the File System**.
3. Navigate to the `job_output_dir/_balancer` directory.

To open a Job Analyzer report in the local file system in a browser:

1. Copy the report from HDFS to the local file system:

```
$ hadoop fs -get job_output_dir/_balancer/jobanalyzer-report.html /home/jdoe
```

2. Switch to the local directory:

```
$ cd /home/jdoe
```

3. Open the file in a browser:

```
$ firefox jobanalyzer-report.html
```

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 5-1](#) 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](#)."

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 29%, and their corresponding elapsed times range from 5 to 15 seconds. This variation indicates skew.

Figure 5-1 Job Analyzer Report for Unbalanced Inverted Index Job

Job Information		Time Information	
Job Name	invindx	Map Phase	00:00:29
Job Id	job_1405207264024_0122	Reduce Phase	00:00:19
Start Time	2014-07-22 15:03:39	Shuffle	00:00:04
Finish Time	2014-07-22 15:04:35	Merge	00:00:08
		Reduce	00:00:15
		Job	00:00:56

Reduce Tasks Metrics Summary

Task ID	Time			%Load
	Start	Finish	Elapsed	Observed
0	15:04:15	15:04:27	00:00:11	12
1	15:04:15	15:04:25	00:00:09	7
2	15:04:15	15:04:25	00:00:09	8
3	15:04:15	15:04:25	00:00:09	7
4	15:04:15	15:04:27	00:00:11	10
5	15:04:15	15:04:28	00:00:12	13
6	15:04:16	15:04:24	00:00:07	4
7	15:04:16	15:04:23	00:00:06	3
8	15:04:16	15:04:35	00:00:18	29
9	15:04:16	15:04:26	00:00:09	8

Reduce Tasks Metrics

Task ID	ElapsedTime			Input								Output			
	Shuffle	Merge	Reduce	Shuffle Bytes		Keys		Records		ValueBytes		Records		Bytes	
				count	%	count	%	count	%	count	%	count	%	count	%
0	00:00:03	00:00:02	00:00:05	20,987,702	12	13	13	2,300,118	12	43,702,242	12	1,698,171	12	48,022,226	12
1	00:00:03	00:00:01	00:00:04	15,630,571	9	12	12	1,438,876	7	27,338,644	7	1,309,838	9	36,424,492	9
2	00:00:03	00:00:02	00:00:04	14,682,583	8	7	7	1,503,328	8	28,563,232	8	1,220,301	9	33,206,894	9
3	00:00:03	00:00:01	00:00:04	14,977,081	9	10	10	1,450,603	7	27,561,457	7	1,253,630	9	33,638,218	9
4	00:00:03	00:00:02	00:00:05	21,175,220	12	11	11	2,073,885	10	39,403,815	10	1,768,698	13	51,191,915	13
5	00:00:03	00:00:03	00:00:05	19,664,004	11	10	10	2,580,271	13	49,025,149	13	1,497,856	11	40,709,092	11
6	00:00:03	00:00:01	00:00:03	8,867,408	5	9	9	775,690	4	14,738,110	4	740,944	5	20,503,323	5
7	00:00:03	00:00:00	00:00:02	5,912,478	3	6	6	531,247	3	10,093,693	3	494,798	4	13,336,988	3
8	00:00:03	00:00:06	00:00:08	35,890,632	21	11	11	5,824,950	29	110,674,050	29	2,527,641	18	69,087,902	18
9	00:00:03	00:00:02	00:00:04	16,252,928	9	11	11	1,521,032	8	28,899,608	8	1,359,917	10	38,889,633	10
Total	00:00:04	00:00:08	00:00:15	174,040,607	-	100	-	20,000,000	-	380,000,000	-	13,871,794	-	385,010,683	-

5.5 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](#)" 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.application_id`
- `oracle.hadoop.balancer.tools.writeKeyBytes`

Key Chopping Properties

- `oracle.hadoop.balancer.choppingStrategy`
- `oracle.hadoop.balancer.keyLoad.minChopBytes`
- `oracle.hadoop.balancer.enableSorting` (deprecated)

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.feedbackDir`
- `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`

5.6 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."](#)
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."](#)
 - Decide which additional configuration properties to set, if any.
See ["About Configuring Perfect Balance."](#)
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."](#)

[Example 5-3](#) 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 5-1](#).

Example 5-3 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-<version>-h2
export HADOOP_CLASSPATH=${BALANCER_HOME}/jlib/orabalancerclient-<version>.jar:${BALANCER_HOME}/jlib/orabalancer-<version>.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="-Xmx1024M ${HADOOP_CLIENT_OPTS}"

# run the job with balancing and job analyzer enabled
hadoop jar application_jarfile.jarApplicationClass
```

```
-D application_config_property \  
-D mapreduce.input.fileinputformat.inputdir=jdoe_application/input \  
-D mapreduce.output.fileoutputformat.outputdir=jdoe_outdir \  
-D mapreduce.job.name="autoinvoke" \  
-D mapreduce.job.reduces=10 \  
-D oracle.hadoop.balancer.autoBalance=true \  
-D oracle.hadoop.balancer.autoAnalyze=REDUCER_REPORT \  
-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
```

```
14/04/14 14:59:42 INFO balancer.Balancer: Creating balancer  
14/04/14 14:59:42 INFO balancer.Balancer: Starting Balancer  
14/04/14 14:59:43 INFO input.FileInputFormat: Total input paths to process : 5  
14/04/14 14:59:46 INFO balancer.Balancer: Balancer completed  
14/04/14 14:59:47 INFO input.FileInputFormat: Total input paths to process : 5  
14/04/14 14:59:47 INFO mapreduce.JobSubmitter: number of splits:5  
14/04/14 14:59:47 INFO mapreduce.JobSubmitter: Submitting tokens for job:  
job_1397066986369_3500  
14/04/14 14:59:47 INFO impl.YarnClientImpl: Submitted application  
application_1397066986369_3500  
14/04/14 14:59:47 INFO mapreduce.Job: The url to track the job:
```

```
.  
.  
.  
Map-Reduce Framework  
Map input records=1000000  
Map output records=20000000  
Map output bytes=872652976  
Map output materialized bytes=175650573  
Input split bytes=580  
Combine input records=0  
Combine output records=0  
Reduce input groups=106  
Reduce shuffle bytes=175650573  
Reduce input records=20000000  
Reduce output records=13871794  
Spilled Records=60000000  
Shuffled Maps =50  
Failed Shuffles=0  
Merged Map outputs=50  
GC time elapsed (ms)=1573  
CPU time spent (ms)=242850  
Physical memory (bytes) snapshot=6789033984  
Virtual memory (bytes) snapshot=24548044800  
Total committed heap usage (bytes)=11921457152  
Shuffle Errors  
BAD_ID=0  
CONNECTION=0  
IO_ERROR=0  
WRONG_LENGTH=0  
WRONG_MAP=0  
WRONG_REDUCE=0  
File Input Format Counters  
Bytes Read=112652976  
File Output Format Counters  
Bytes Written=384974202
```


5.7 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](#)."
- **Partition report:** Identifies the keys that are assigned to the various reducers. This report is saved in JSON for Perfect Balance to use; it does not contain information of use to you. The report is named `${job_output_dir}/_balancer/orabalancer_report.json`. It is only generated for balanced jobs.
- **Reduce key metric reports:** Perfect Balance generates a report for each file partition, when the appropriate configuration properties are set. The reports are saved in XML for Perfect Balance to use; they do not contain information of use to you. They are named `${job_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](#)."

The reports are stored by default in the job output directory (`${mapreduce.output.fileoutputformat.outputdir}`) in YARN. 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.json
/part-r-00000
/part-r-00001
.
.
.
```

5.8 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**.

5.8.1 Selecting a Chopping Method

You can configure how Perfect Balance chops the values by setting the [oracle.hadoop.balancer.choppingStrategy](#) configuration property:

- **Chopping by hash partitioning:** Set `choppingStrategy=hash` when sorting is not required. This is the default chopping strategy.
- **Chopping by round robin:** Set `choppingStrategy=roundRobin` as an alternative strategy when total-order chopping is not required. If the load for a hash chopped key is unbalanced among reducers, try to use this chopping strategy.
- **Chopping by total-order partitioning:** Set `choppingStrategy=range` 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 `choppingStrategy=range`, 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**.

5.8.2 How Chopping Impacts Applications

If a MapReduce job aggregates the data by reduce key, then each reduce task aggregates the values for each key within that task. However, when chopping is enabled in Perfect Balance, the rows associated with a reduce key might be in different reduce tasks, leading to partial aggregation. Thus, values for a reduce key are aggregated within a reduce task, but not across reduce tasks. (The values for a reduce key across reduce tasks can be sorted, as discussed in "[Selecting a Chopping Method](#)".)

When complete aggregation is required, you can disable chopping. Alternatively, you can examine the application that consumes the output of your MapReduce job. The application might work well with partial aggregation.

For example, a search engine might read in parallel the output from a MapReduce job that creates an inverted index. The output of a reduce task is a list of words, and for each word, a list of documents in which the word occurs. The word is the key, and the list of documents is the value. With partial aggregation, some words have multiple document lists instead of one aggregated list. Multiple lists are convenient for the search engine to consume in parallel. A parallel search engine might even require document lists to be split instead of aggregated into one list. See "[About the Perfect Balance Examples](#)" for a Hadoop job that creates an inverted index from a document collection.

As another example, Oracle Loader for Hadoop loads data from multiple files to the correct partition of a target table. The load step is faster when there are multiple files for a reduce key, because they enable a higher degree of parallelism than loading from one file for a reduce key.

5.9 Troubleshooting Jobs Running with Perfect Balance

If you get Java "out of heap space" or "GC overhead limit exceeded" errors on the client node while running the Perfect Balance sampler, 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="-Xmx1024M $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.

Setting `HADOOP_CLIENT_OPTS` is sufficient to increase the heap size for the sampler, regardless of whether [oracle.hadoop.balancer.runMode](#) is set to `local` or `distributed`. When `runMode=local`, the sampler runs on the client node, and `HADOOP_CLIENT_OPTS` sets the heap size on the client node. When `runMode=distributed`, Perfect Balance automatically sets the heap size for the sampler Hadoop job based on the `-Xmx` setting you provide in `HADOOP_CLIENT_OPTS`. Perfect Balance never changes the heap size for the map and reduce tasks of your job, only for its sampler job.

5.10 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.

5.10.1 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-<version>-h2/examples/invindx/README.txt`.

To explore the modified Java code, see `orabalancer-<version>-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 5-4](#) shows fragments from the inverted index Java code.

See Also:

Oracle Big Data Appliance Perfect Balance Java API Reference

Example 5-4 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) //////////
.
.
.
}
```

5.10.2 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 application_jar_path.jar...
```

Example 5-5 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 5-1](#).

To run the `InvertedIndexMapreduce` class example, see "[About the Perfect Balance Examples](#)."

Example 5-5 Running the InvertedIndexMapreduce Class

```
$ cat pb_balanceapi.sh
BALANCER_HOME=/opt/oracle/orabalancer-<version>-h2
APP_JAR_FILE=/opt/oracle/orabalancer-<version>-h2/jlib/orabalancer-<version>.jar
export HADOOP_CLASSPATH=${BALANCER_HOME}/jlib/orabalancer-<version>.jar:${
{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.invidx.InvertedIndexMapreduce \
-D mapreduce.input.fileinputformat.inputdir=invidx/input \
-D mapreduce.output.fileoutputformat.outputdir=jdoe_outdir_api \
-D mapreduce.job.name=jdoe_invidx_api \
-D mapreduce.job.reduces=10 \
-D oracle.hadoop.balancer.linearKeyLoad.keyWeight=93.981394 \
-D oracle.hadoop.balancer.linearKeyLoad.rowWeight=0.001126 \
-D oracle.hadoop.balancer.linearKeyLoad.byteWeight=0.0

$ sh ./balanceapi.sh
14/04/14 15:03:51 INFO balancer.Balancer: Creating balancer
14/04/14 15:03:51 INFO balancer.Balancer: Starting Balancer
14/04/14 15:03:51 INFO input.FileInputFormat: Total input paths to process : 5
14/04/14 15:03:54 INFO balancer.Balancer: Balancer completed
14/04/14 15:03:55 INFO input.FileInputFormat: Total input paths to process : 5
14/04/14 15:03:55 INFO mapreduce.JobSubmitter: number of splits:5
14/04/14 15:03:55 INFO mapreduce.JobSubmitter: Submitting tokens for job:
job_1397066986369_3510
14/04/14 15:03:55 INFO impl.YarnClientImpl: Submitted application
application_1397066986369_3510
.
.
.
File Input Format Counters
Bytes Read=112652976
File Output Format Counters
Bytes Written=384974202
```

5.11 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.

5.11.1 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 `mapreduce.input.fileinputformat.inputdir` 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."](#)

- Replace `jdoe` with your Hadoop user name.
- Review the configuration setting and the shell script to ensure that they are appropriate for the 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.

5.11.2 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-<version>-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-<version>-h2/examples/invindx/README.txt`.

5.12 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](#)
- [Job Analyzer Configuration Properties](#)
- [Perfect Balance Configuration Properties](#)

See "[About Configuring Perfect Balance](#)" for a list of the properties organized into functional categories.

Note:

CDH5 deprecates many MapReduce properties and replaces them with new properties. Perfect Balance continues to work with the old property names, but Oracle recommends that you use the new names. For the new MapReduce property names, see the Cloudera website at:

<http://archive.cloudera.com/cdh5/cdh/5/hadoop/hadoop-project-dist/hadoop-common/DeprecatedProperties.html>

MapReduce Configuration Properties

Property	Type, Default Value, Description
<code>mapreduce.input.fileinputformat.inputdir</code>	Type: String Default Value: Not defined Description: A comma-separated list of input directories.
<code>mapreduce.inputformat.class</code>	Type: String Default Value: <code>org.apache.hadoop.mapreduce.lib.input.TextInputFormat</code> Description: The full name of the <code>InputFormat</code> class.
<code>mapreduce.map.class</code>	Type: String Default Value: <code>org.apache.hadoop.mapreduce.Mapper</code> Description: The full name of the mapper class.
<code>mapreduce.output.fileoutputformat.outputdir</code>	Type: String Default Value: Not defined Description: The job output directory.
<code>mapreduce.partition.class</code>	Type: String Default Value: <code>org.apache.hadoop.mapreduce.lib.partition.HashPartitioner</code> Description: The full name of the partitioner class.

Property	Type, Default Value, Description
<code>mapreduce.reduce.class</code>	<p>Type: String</p> <p>Default Value: <code>org.apache.hadoop.mapreduce.Reducer</code></p> <p>Description: The full name of the reducer class.</p>

Job Analyzer Configuration Properties

Property	Type, Default Value, Description
<code>oracle.hadoop.balancer.application_id</code>	<p>Type: String</p> <p>Default Value: Not defined</p> <p>Description: The job identifier of the job you want to analyze with Job Analyzer. This property is a parameter to the Job Analyzer utility in standalone mode on YARN clusters; it does not apply to MRv1 clusters. See "Running Job Analyzer as a Standalone Utility".</p>
<code>oracle.hadoop.balancer.tools.writeKeyBytes</code>	<p>Type: Boolean</p> <p>Default Value: <code>false</code></p> <p>Description: Controls whether the counting reducer collects the byte representations of the reduce keys for the Job Analyzer. Set this property to <code>true</code> to represent the unique key values in Base64 encoding in the report. A string representation of the key, created using <code>key.toString</code>, is also provided in the report. This string value may not be unique for each key.</p>

Perfect Balance Configuration Properties

Property	Type, Default Value, Description
<code>oracle.hadoop.balancer.autoAnalyze</code>	<p>Type: Enumeration</p> <p>Default Value: <code>BASIC_REPORT</code> if <code>oracle.hadoop.balancer.autoBalance</code> is true; otherwise <code>NONE</code></p> <p>Description: Controls the behavior of the Job Analyzer when it is called using Perfect Balance. The following values are valid:</p> <ul style="list-style-type: none"> <code>NONE</code>: Disables Job Analyzer. <code>BASIC_REPORT</code>: Enables Job Analyzer <code>REDUCER_REPORT</code>: Enables Job Analyzer such that it collects additional load statistics for each reduce task in a job. See "Collecting Additional Metrics." <p>Perfect Balance uses this property; the Perfect BalanceAPI does not.</p>
<code>oracle.hadoop.balancer.autoBalance</code>	<p>Type: Boolean</p> <p>Default Value: <code>false</code></p> <p>Description: Controls whether load balancing is enabled. Set to <code>false</code> to turn off balancing. Perfect Balance uses this property; the Perfect Balance API does not.</p>
<p><code>oracle.hadoop.balancer.choppingStrategy</code></p> <p>Note that the <code>choppingStrategy</code> property takes precedence over the deprecated property <code>oracle.hadoop.balancer.enableSorting</code>. If the <code>choppingStrategy</code> property is not set, <code>oracle.hadoop.balancer.enableSorting=true</code> is equivalent to setting the <code>choppingStrategy</code> property to <code>range</code>. Likewise, setting <code>oracle.hadoop.balancer.enableSorting=false</code> is equivalent to setting the <code>choppingStrategy</code> property to <code>hash</code>.</p>	<p>Type: String</p> <p>Default Value: <code>hash</code></p> <p>Description: This property controls the behavior of sampler when it needs to chop a key. The following values are valid:</p> <ul style="list-style-type: none"> <code>range</code>: Records of chopped keys are assigned to different reducers according to the total-order partitioning function specified by the map output key sorting comparator, so balancer will preserve a total order over the values of a chopped key. <code>hash</code>: Records of chopped keys are assigned to different reducers according to the hashCode on the map output values. In most cases, this approach gives a balanced work load among reducers. <code>roundRobin</code>: Records of chopped keys are assigned to different reducers in round-robin order. This is an alternative strategy when it is not required to preserve a total order over the value of a chopped key. If the load for a hash chopped key is unbalanced among reducers, try to use this chopping strategy. <p>See also the deprecated property: oracle.hadoop.balancer.enableSorting</p>

Property	Type, Default Value, Description
<code>oracle.hadoop.balancer.confidence</code>	<p>Type: Float</p> <p>Default Value: 0.95</p> <p>Description: The statistical confidence indicator for the load factor specified by the oracle.hadoop.balancer.maxLoadFactor property.</p> <p>This property accepts values greater than or equal to 0.5 and less than 1.0 ($0.5 \leq \text{value} < 1.0$). 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.</p>
<code>oracle.hadoop.balancer.enableSorting</code>	<p>Type: Boolean</p> <p>Default Value: false</p> <p>Description: This property is deprecated. To use the map output key sorting comparator as a total-order partitioning function, set <code>oracle.hadoop.balancer.choppingStrategy</code> to <code>range</code>.</p> <p>When this property is false, map output keys will be chopped using a hash function. When this property is true, map output keys will be chopped using the map output key sorting comparator as a total-order partitioning function. When this property is true, balancer will preserve a total order over the values of a chopped key.</p> <p>See also: oracle.hadoop.balancer.choppingStrategy</p>

Property	Type, Default Value, Description
<code>oracle.hadoop.balancer.inputFormat.mapred.map.tasks</code>	<p>Type: Integer</p> <p>Default Value: 100</p> <p>Description: Sets the Hadoop <code>mapred.map.tasks</code> property for the duration of sampling, just before calling the input format <code>getSplits</code> method. It does not change <code>mapred.map.tasks</code> 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).</p> <p>Set this property to a value greater than or equal to one (1). A value less than 1 disables the property.</p> <p>Some input formats, such as <code>DBInputFormat</code>, use this property as a hint to determine the number of splits returned by <code>getSplits</code>. Higher values indicate that more chunks of data are sampled at random, which improves the sample.</p> <p>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 <code>getSplits</code> method to run out of memory by returning too many splits.</p>
<code>oracle.hadoop.balancer.inputFormat.mapred.max.split.size</code>	<p>Type: Long</p> <p>Default Value: 1048576 (1 MB)</p> <p>Description: Sets the Hadoop <code>mapred.max.split.size</code> property for the duration of sampling, just before calling the input format's <code>getSplits</code> method. It does not change <code>mapred.max.split.size</code> for the actual job.</p> <p>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).</p> <p>Some input formats, such as <code>FileInputFormat</code>, use the maximum split size as a hint to determine the number of splits returned by <code>getSplits</code>. 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.</p> <p>You can increase the value for larger data sets (tens of terabytes) or if the input format's <code>getSplits</code> method throws an out of memory error. Large splits are better for I/O performance, but not for sampling.</p>

Property	Type, Default Value, Description
<code>oracle.hadoop.balancer.keyLoad.minChopBytes</code>	<p>Type: Long</p> <p>Default Value: 0</p> <p>Description: Controls whether Perfect Balance chops large map output keys into medium keys:</p> <ul style="list-style-type: none">• <code>-1</code>: Perfect Balance does not chop large map output keys.• <code>0</code>: Perfect Balance chops large map output keys and determines the optimal size of each medium key.• <i>Positive integer</i>: Perfect Balance chops large map output keys into medium keys with a size greater than or equal to the specified integer.
<code>oracle.hadoop.balancer.linearKeyLoad.byteWeight</code>	<p>Type: Float</p> <p>Default Value: 0.05</p> <p>Description: Weights the number of bytes per key in the linear key load model specified by the <code>oracle.hadoop.balancer.KeyLoadLinear</code> class.</p>

Property	Type, Default Value, Description
<code>oracle.hadoop.balancer.linearKeyLoad.feedbackDir</code>	<p>Type: String</p> <p>Default Value: Not defined</p> <p>Description: The path to a directory that contains the Job Analyzer report for a job that it previously analyzed. The sampler reads this report for feedback to use to optimize the current balancing plan. You can set this property to the Job Analyzer report directory of a job that is the same or similar to the current job, so that the feedback is directly applicable.</p> <p>If the feedback directory contains a Job Analyzer report with recommended values for the Perfect Balance linear key load model coefficients, then Perfect Balance automatically reads and uses them. The recommended values take precedence over user-specified values in these configuration parameters:</p> <ul style="list-style-type: none"> • oracle.hadoop.balancer.linearKeyLoad.byteWeight • oracle.hadoop.balancer.linearKeyLoad.keyWeight • oracle.hadoop.balancer.linearKeyLoad.rowWeight <p>Job Analyzer attempts to recommend good values for these coefficients. However, Perfect Balance reads the load model coefficients from this list of configuration properties under the following circumstances:</p> <ul style="list-style-type: none"> • The <code>feedbackDir</code> property is not set. • The <code>feedbackDir</code> property is set, but the Job Analyzer report in the specified directory does not contain a good recommendation for the load model coefficients.
<code>oracle.hadoop.balancer.linearKeyLoad.keyWeight</code>	<p>Type: Float</p> <p>Default Value: 50.0</p> <p>Description: Weights the number of medium keys per large key in the linear key load model specified by the <code>oracle.hadoop.balancer.KeyLoadLinear</code> class.</p>
<code>oracle.hadoop.balancer.linearKeyLoad.rowWeight</code>	<p>Type: Float</p> <p>Default Value: 0.05</p> <p>Description: Weights the number of rows per key in the linear key load model specified by the <code>oracle.hadoop.balancer.KeyLoadLinear</code> class.</p>

Property	Type, Default Value, Description
<code>oracle.hadoop.balancer.maxLoadFactor</code>	<p>Type: Float</p> <p>Default Value: 0.05</p> <p>Description: The target reducer load factor that you want the balancer's partition plan to achieve.</p> <p>The load factor is the relative deviation from an estimated value. For example, if <code>maxLoadFactor=0.05</code> and <code>confidence=0.95</code>, 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.</p> <p>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.</p>
<code>oracle.hadoop.balancer.maxSamplesPct</code>	<p>Type: Float</p> <p>Default Value: 0.01 (1%)</p> <p>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).</p> <p>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.</p> <p>See oracle.hadoop.balancer.useClusterStats.</p>
<code>oracle.hadoop.balancer.minSplits</code>	<p>Type: Integer</p> <p>Default Value: 5</p> <p>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.</p>

Property	Type, Default Value, Description
<code>oracle.hadoop.balancer.numThreads</code>	<p>Type: Integer</p> <p>Default Value: 5</p> <p>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.</p>
<code>oracle.hadoop.balancer.report.override</code>	<p>Type: Boolean</p> <p>Default Value: false</p> <p>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.</p>
<code>oracle.hadoop.balancer.reportPath</code>	<p>Type: String</p> <p>Default Value: <code>directory/orabalancer_report-random_unique_string.json</code>, 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.</p> <p>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 <code>job_output_dir/_balancer/orabalancer_report.json</code>. In the API, the <code>save</code> method does this task.</p>
<code>oracle.hadoop.balancer.runMode</code>	<p>Type: String</p> <p>Default Value: <code>local</code></p> <p>Description: Specifies how to run the Perfect Balance sampler. The following values are valid:</p> <ul style="list-style-type: none"> <code>local</code>: The sampler runs on the client node where the job is submitted. <code>distributed</code>: The sampler runs as a Hadoop job. If the job uses the distributed cache, then Perfect Balance automatically sets this property to <code>distributed</code>. <p>If this property is set to an invalid string, Perfect Balance resets it to <code>local</code>.</p>

Property	Type, Default Value, Description
<code>oracle.hadoop.balancer.tmpDir</code>	<p>Type: String</p> <p>Default Value: <code>/tmp/orabalancer-user_name</code></p> <p>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.</p>
<code>oracle.hadoop.balancer.useClusterStats</code>	<p>Type: Boolean</p> <p>Default Value: <code>true</code></p> <p>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.</p> <p>Set this property to <code>false</code> 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 <code>true</code>, because the distribution of map-output keys is usually unknown.</p>
<code>oracle.hadoop.balancer.useMapreduceApi</code>	<p>Type: Boolean</p> <p>Default Value: <code>true</code></p> <p>Description: Identifies the MapReduce API used in the Hadoop job:</p> <ul style="list-style-type: none">• <code>true</code>: The job uses the mapreduce API.• <code>false</code>: The job uses the mapred API.

Part III

Oracle Table Access for Hadoop and Spark

This part describes Oracle Table Access for Hadoop and Spark storage handler for Oracle Database. It contains the following chapters:

- [Oracle DataSource for Apache Hadoop \(OD4H\)](#)

Oracle DataSource for Apache Hadoop (OD4H)

Oracle DataSource for Apache Hadoop (OD4H) allows direct, fast, parallel, secure and consistent access to master data in Oracle Database using Spark SQL via Hive metastore. This chapter discusses Oracle DataSource for Apache Hadoop (OD4H) in the following sections:

- [Operational Data, Big Data and Requirements](#)
- [Overview of Oracle DataSource for Apache Hadoop \(OD4H\)](#)
- [How Does OD4H Work?](#)
- [Features of OD4H](#)
- [Using Hive SQL with OD4H](#)
- [Using Spark SQL with OD4H](#)
- [Writing Back To Oracle Database](#)

6.1 Operational Data, Big Data and Requirements

The common data architecture in most companies nowadays generally comprises of the following components:

- Oracle Database(s) for operational, transactional, and master data, that is shared business object such as customers, products, employees and so on
- Big Data

Hadoop applications such as Master Data Management (MDM), Events processing, and others, need access to data in both Hadoop storages (such as HDFS and NoSQL Database as a landing point for weblogs, and so on) and Oracle Database (as the reliable and auditable source of truth). There are two approaches to process such data that reside in both Hadoop storage and Oracle Database:

- ETL Copy using tools such as Oracle's Copy to BDA
- Direct Access using Oracle Big Data SQL and Oracle DataSource for Apache Hadoop (OD4H).

In this chapter, we will discuss Oracle DataSource for Apache Hadoop (OD4H).

6.2 Overview of Oracle DataSource for Apache Hadoop (OD4H)

Oracle DataSource for Apache Hadoop (OD4H) is the storage handler for Oracle Database that uses HCatalog and InputFormat.

This section discusses the following concepts:

- [Opportunity with Hadoop 2.x](#)
- [Oracle Tables as Hadoop Data Source](#)
- [External Tables](#)

6.2.1 Opportunity with Hadoop 2.x

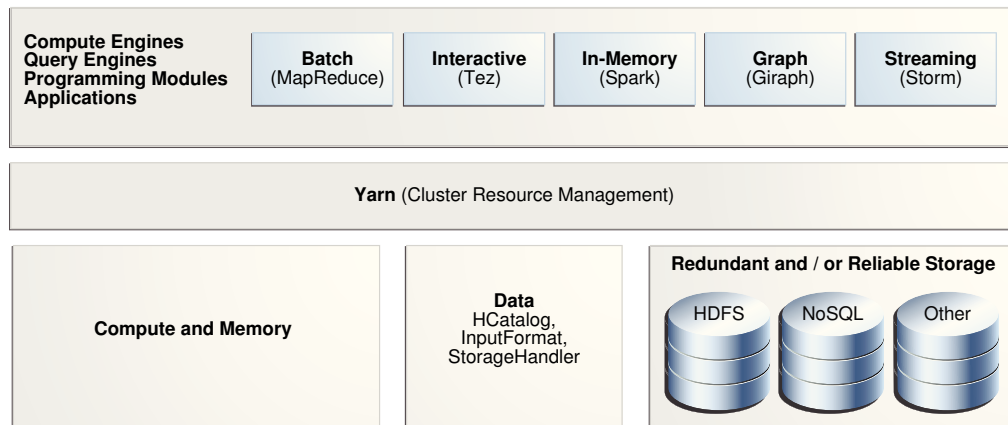
Hadoop 2.x architecture decouples compute engines from cluster resources management and storages. It enables:

- A variety of SQL query engines. For instance, Hive SQL, Spark SQL, Big Data SQL, and so on.
- A variety of programmatic compute engines. For instance, MapReduce, Pig, Storm, Solr, Cascading, and so on.
- Elastic allocation of compute resources (CPU, memory) through YARN.
- A variety of data stores such as HDFS, NoSQL, as well as remote storages through HCatalog, InputFormat, OutputFormat and StorageHandler interfaces.

Oracle DataSource for Apache Hadoop (OD4H) is the storage handler for Oracle Database that uses HCatalog and InputFormat.

Following is an illustration of Hadoop 2.0 Architecture:

Figure 6-1 Hadoop 2.0 Architecture



6.2.2 Oracle Tables as Hadoop Data Source

OD4H enables current and ad-hoc querying. This makes querying data faster and more secure. You can query data directly and retrieve only the data that you need, when you need it.

OD4H also provides Oracle's end-to-end security. This includes Identity Management, Column Masking, and Label and Row Security.

OD4H also furnishes direct access for Hadoop and Spark APIs such as Pig, MapReduce and others.

6.2.3 External Tables

External Tables turn Oracle tables into Hadoop and/or Spark datasources. The DDL for declaring External Tables is as follows:

```
CREATE[TEMPORARY] EXTERNAL TABLE [IF NOT EXISTS] [db_name.]table_name
[(col_name data_type [COMMENTcol_comment],...)]
[COMMENT table_comment]
STORED BY 'oracle.hcat.osh.OracleStorageHandler' [WITHSERDEPROPERTIES(...)]
[TBLPROPERTIES (property_name=property_value,...)]
```

```
data_type
|SMALLINT
|INT
|BIGINT
|BOOLEAN
|FLOAT
|DOUBLE
|STRING
|BINARY
|TIMESTAMP
|DECIMAL
|DECIMAL(precision,scale)
|VARCHAR
|CHAR
```

See Also: Refer the following link for Hive External Table syntax <https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-CreateTable>

Note:

Oracle supports only primitive types.

The properties of external tables can be described as follows:

6.2.3.1 TBLPROPERTIES

Property	Use
oracle.hcat.osh.columns.mapping	Comma separated list to specify mapping between Hive columns and Oracle table columns. All external tables using OracleStorageHandler must define this.
mapreduce.jdbc.url	Connection URL to connect to the database
mapreduce.jdbc.username	Connection user name to connect to the database
mapreduce.jdbc.password	Connection password to connect to the database
mapreduce.jdbc.input.table.name	Oracle table name
mapreduce.jdbc.input.conditions	To be used for querying the database. Must be used for query pushdown.

Property	Use
mapreduce.jdbc.input.query	To be used for querying the database. Query should be used only when a subset of the columns is selected.
mapreduce.jdbc.input.orderby	ORDER BY clause to be specified for pushing ordering to the database.
oracle.hcat.osh.splitterKind	To be used to specify how OracleStorageHandler must create splits, so that they are a good match for the physical structure of the target table in Oracle Database. The splitter kind applicable could be SINGLE_SPLITTER, PARTITION_SPLITTER, ROW_SPLITTER, BLOCK_SPLITTER.
oracle.hcat.osh.rowsPerSplit	Used only when ROW_SPLITTER splitterKind is applied on the table. Represents Number of rows per split for LIMIT_RANGE splitter. Default is 1000
oracle.hcat.osh.authentication	Authentication method used to connect to Oracle Database. Can be SIMPLE (default), ORACLE_WALLET, KERBEROS
sun.security.krb5.principal	Kerberos principal. Used only when KERBEROS authentication is applied.
oracle.hcat.osh.kerb.callback	Callback for Kerberos authentication. Used only when Kerberos authentication is applied.
oracle.hcat.osh.maxSplits	Maximum number of splits for any splitter kind
oracle.hcat.osh.useChunkSplitter	Use chunk based ROW_SPLITTER and BLOCK_SPLITTER that use DBMS_PARALLEL_EXECUTE package to divide table into chunks that will map to hadoop splits. The default value is set to 'true'.
oracle.hcat.osh.chunkSQL	Used by CUSTOM_SPLITTER to create splits. The SQL string should be a SELECT statement that returns range of each chunk must have two columns: start_id and end_id The columns must be of ROWID type.
oracle.hcat.osh.useOracleParallelism	When configured, parallel queries will be executed while fetching rows from Oracle. Default value: 'false'
oracle.hcat.osh.fetchSize	JDBC fetchsize for generated select queries used to fetch rows. Default value: 10 (set by Oracle JDBC Driver)

Note:

In addition to the above, any JDBC connection properties (oracle.jdbc.* and oracle.net.*) can be specified as TBLPROPERTIES. They will be used while establishing connection to Oracle Database using JDBC driver.

Note:

Oracle DataSource for Apache Hadoop (OD4H) works with Oracle View and Oracle Tables.

6.2.3.2 SERDE PROPERTIES

Property	Use
oracle.hcat.osh.columns.mapping	All external tables using OracleStorageHandler must define this. Its a comma separated list to specify mapping between hive columns (specified in create table) and oracle table columns. WITHSERDEPROPERTIES also enables the external table definition to refer only to select columns in the actual Oracle table. In other words, not all columns from the Oracle table need to be part of the Hive external table. The ordering of oracle columns in the mapping is the same as ordering of hive columns specified in create table.

6.2.4 List of jars in the OD4H package

Oracle DataSource for Apache Hadoop (OD4H) contains the following list of jars.

OD4H consists of the following list of jars.

Table 6-1 List of jars in OD4H

Name of JAR	Use
osh.jar	Contains OracleStorageHandler Implementation
ojdbc7.jar	An OD4H specific JDBC driver (which is optimized with internal calls), used by Spark or Hadoop tasks to connect to the database.
ucp.jar	For creating connection pools in OracleStorageHandler
oraclepki103.jar, osdt_core.jar, osdt_cert.jar, osdt_jce.jar	For Oracle Wallet authentication
orai18n.jar	Oracle Globalization Support
xdb.jar	Oracle XDB jar

6.3 How does OD4H work?

Oracle DataSource for Apache Hadoop (OD4H) does not require creating a new table. You can start working with OD4H using the following steps:

1. Create a new Oracle table, or, reuse an existing table.

2. Create the Hive DDL for creating the external table referencing the Oracle Table.
3. Issue HiveSQL, SparkSQL, or other Spark/Hadoop queries and API calls.

The following sections show how to create a new Oracle Database Table, and a Hive DDL:

- [Create a New Oracle Database Table](#)
- [Hive DDL](#)
- [Creating External Table in Hive](#)

6.3.1 Create a new Oracle Database Table or Reuse an Existing Table

Here is an illustration of a partitioned Oracle table that we will use to demo how partition pruning works:

1.

```
CREATE TABLE EmployeeData ( Emp_ID NUMBER,
                             First_Name VARCHAR2(20),
                             Last_Name  VARCHAR2(20),
                             Job_Title  VARCHAR2(40),
                             Salary     NUMBER)
PARTITION BY RANGE (Salary)
( PARTITION salary_1 VALUES LESS THAN (60000)
  TABLESPACE tsa
, PARTITION salary_2 VALUES LESS THAN (70000)
  TABLESPACE tsb
, PARTITION salary_3 VALUES LESS THAN (80000)
  TABLESPACE tsc
, PARTITION salary_4 VALUES LESS THAN (90000)
  TABLESPACE tsd
, PARTITION salary_5 VALUES LESS THAN (100000)
  TABLESPACE tse
);
```

Note:

You can use this syntax for table creation, in the following examples listed in this Book.

2. Issue queries from Hive, Spark, or any other Hadoop models (including joins with local Hive Tables.)

6.3.2 Hive DDL

In this example, we will associate two Hive external tables to the same Oracle table, using two different split patterns:

- SIMPLE_SPLITTER
- PARTITION_SPLITTER

Note: It is possible that the external table has fewer columns than the base Oracle table.

Since columns can have different names, use TBLPROPERTY for mapping with the base table.

In the following examples, we are using the following variables:

```
connection_string = jdbc:oracle:thin:@localhost:1521/
<servicename>

oracle_user=od4h

oracle_pwd=od4h
```

The following command creates a Hive external table with the default split pattern, that is SIMPLE_SPLITTER.

```
CREATE EXTERNAL TABLE EmployeeDataSimple (
  Emp_ID int,
  First_Name string,
  Last_Name string,
  Job_Title string,
  Salary int
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
WITH SERDEPROPERTIES (
  'oracle.hcat.osh.columns.mapping' =
  'Emp_ID,First_Name,Last_Name,Job_Title,Salary')
TBLPROPERTIES (
  'mapreduce.jdbc.url' = '${hiveconf:jdbc:oracle:thin:@localhost:1521/<servicename>}',
  'mapreduce.jdbc.username' = '${hiveconf:od4h}',
  'mapreduce.jdbc.password' = '${hiveconf:od4h}',
  'mapreduce.jdbc.input.table.name' = 'EmployeeData'
);
```

The following example creates a Hive external table using PARTITION_SPLITTER.

```
DROP TABLE EmployeeDataPartitioned;
CREATE EXTERNAL TABLE EmployeeDataPartitioned (
  Emp_ID int,
  First_Name string,
  Last_Name string,
  Job_Title string,
  Salary int
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
WITH SERDEPROPERTIES (
  'oracle.hcat.osh.columns.mapping' =
  'Emp_ID,First_Name,Last_Name,Job_Title,Salary')
TBLPROPERTIES (
  'mapreduce.jdbc.url' = '${hiveconf:jdbc:oracle:thin:@localhost:1521/<servicename>}',
  'mapreduce.jdbc.username' = '${hiveconf:od4h}',
  'mapreduce.jdbc.password' = '${hiveconf:od4h}',
  'mapreduce.jdbc.input.table.name' = 'EmployeeData',
  'oracle.hcat.osh.splitterKind' = 'PARTITIONED_TABLE'
);
```

See Also: <http://www.oracle.com/technetwork/database/bigdata-appliance/overview/index.html> for demo code samples

6.3.3 Creating External Tables in Hive

You can create an external table in Hive in the following way:

```
DROP TABLE employees;

CREATE EXTERNAL TABLE employees (
  EMPLOYEE_ID INT,
  FIRST_NAME  STRING,
  LAST_NAME   STRING,
  SALARY      DOUBLE,
  HIRE_DATE   TIMESTAMP,
  JOB_ID      STRING
)

STORED BY 'oracle.hcat.osh.OracleStorageHandler'

WITH SERDEPROPERTIES (
  'oracle.hcat.osh.columns.mapping' =
  'employee_id,first_name,last_name,salary,hire_date,job_id')

TBLPROPERTIES (
  'mapreduce.jdbc.url' = 'jdbc:oracle:thin:@localhost:1521:orcl',
  'mapreduce.jdbc.username' = 'hr',
  'mapreduce.jdbc.password' = 'hr',
  'mapreduce.jdbc.input.table.name' = 'EMPLOYEES'
);
```

Note: Include `ucp.jar`, `ojdbc7.jar` and `osh.jar` in the Hive auxpath controlled by `HIVE_AUX_JARS_PATH` environment variable that is present in `hive-env.sh`, `hive.aux.jars.path` configuration property or `--auxpath` option when you invoke Hive. On BDA, you can configure these using Cloudera Manager interface. You should also add these jars to classpath of hadoop tasks using `add jar` on Hive command line.

For various Hive Command Line options and configuration properties, refer the following sources:

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+Cli>
<https://cwiki.apache.org/confluence/display/Hive/Configuration+Properties>

6.4 Features of OD4H

The following topics discuss features of OD4H.

- [Performance and Scalability Features](#)
- [Security Features](#)
- [Using Hive SQL with OD4H](#)
- [Using Spark SQL with OD4H](#)

6.4.1 Performance And Scalability Features

Following sections discuss the performance and scalability features of OD4H:

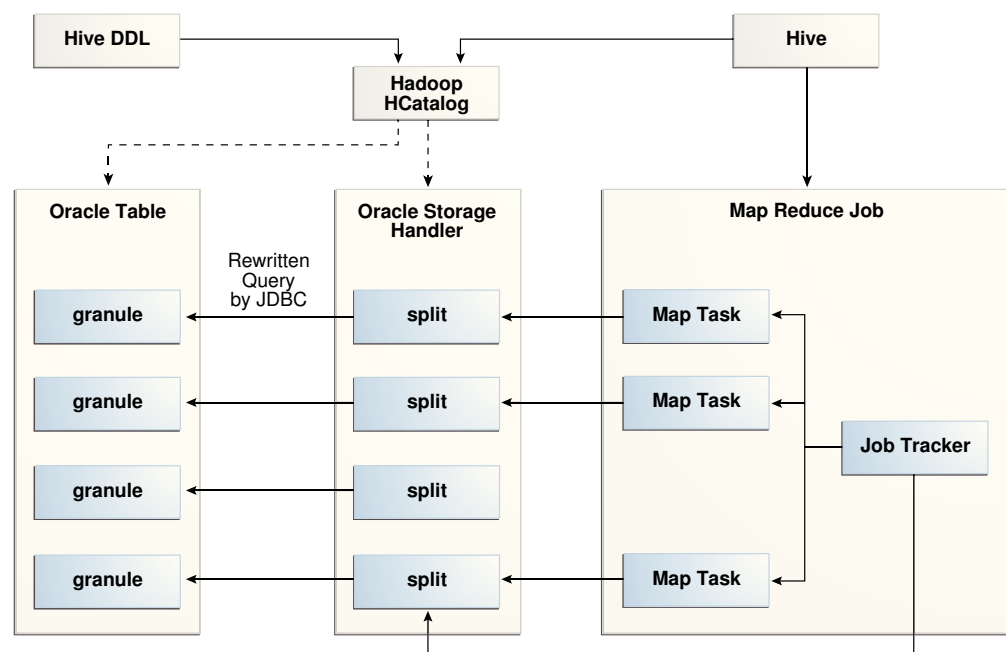
- [Splitters](#)
- [Predicate Pushdown](#)
- [Projection Pushdown](#)
- [Partition Pruning](#)
- [Smart Connection Management](#)

HCatalog stores table metadata from Hive DDL, HiveSQL, Spark SQL and others, then use this metadata while submitting queries.

The Oracle table is divided into granules determined by the `splitterKind` property. These granules are then read into a split by `OracleStorageHandler`, by submitting generated queries.

`OracleStorageHandler` will not have to test all possible query types if the query plan determines which splits need to be scanned.

Figure 6-2 OD4H in a Nutshell



6.4.1.1 Splitters

While executing a query on a Hive external table through OD4H, the underlying Oracle table is dynamically divided into granules, which correspond to splits on the Hadoop side. Each split is processed by a single map task. With the help of the `ORACLE_SPLITTER_KIND` property, you can specify how the splits are created. This ensures that the splits are a good match for the physical structure of the target table in Oracle Database.

The different kinds of splitters available are:

SINGLE_SPLITTER

Creates one split for the table. Use `SINGLE_SPLITTER` where a single task is sufficient to process the query against the entire table.

ROW_SPLITTER

Limits the number of rows per Split. The default number of rows is 1000. You can specify number of rows by setting the `oracle.hcat.osh.rowsPerSplit` property. The default value of `oracle.hcat.osh.maxSplits` is 1 when `ROW_SPLITTER` is used. You can increase this value to enable parallel reads.

Based on the values provided in the `rowsPerSplit` property, OD4H will divide tables into splits. If the number of splits obtained is higher than the `maxSplits`, then `maxSplits` property will be used. The rows per split will be divided accordingly.

Note: `oracle.hcat.osh.rowsPerSplit` is used only by `ROW_SPLITTER` and not any other splitter kind.

BLOCK_SPLITTER

Creates splits based on underlying storage of data blocks. With Block Splitter, you can specify the maximum number of splits to be generated. The default value of `oracle.hcat.osh.maxSplits` is 1, when `BLOCK_SPLITTER` is used. You can increase this value to enable parallel reads. `BLOCK_SPLITTER` requires `SELECT` privilege on the `SYS.DBA.EXTENTS` table, granted to the schema containing the Oracle target table. In the event that this permission does not exist, OD4H will use `SINGLE_SPLITTER`.

Note: The actual number of splits under `BLOCK_SPLITTER` may be lesser than the value specified in the `oracle.hcat.osh.maxSplits` property.

Do not use `BLOCK_SPLITTER` on partitioned tables or Index Organized tables.

Note: For `ROW_SPLITTER` and `BLOCK_SPLITTER` types, use `oracle.hcat.osh.useChunkSplitter` to specify splitting mechanism. The default property value is `true`. This enables creating chunks corresponding to splits using the `DBMS_PARALLEL_EXECUTE` package. When the property value is `false`, custom SQL is generated for splitting.

PARTITION_SPLITTER

Creates one split per partition. `PARTITION_SPLITTER` is used by default when the table is partitioned. You can override this setting by specifying `ROW_SPLITTER` in table properties. With `PARTITION_SPLITTER`, the default value of `oracle.hcat.osh.maxSplits` table property is 64.

Following is an illustration of `ROW_SPLITTER`:

```
DROP TABLE employees;

CREATE EXTERNAL TABLE employees (
  EMPLOYEE_ID INT,
  FIRST_NAME  STRING,
```

```

    LAST_NAME    STRING,
    SALARY        DOUBLE,
    HIRE_DATE     TIMESTAMP,
    JOB_ID        STRING
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'

WITH SERDEPROPERTIES (
  'oracle.hcat.osh.columns.mapping' =
  'employee_id,first_name,last_name,salary,hire_date,job_id')

TBLPROPERTIES (
  'mapreduce.jdbc.url' = 'jdbc:oracle:thin:@localhost:1521:orcl',
  'mapreduce.jdbc.username' = 'hr',
  'mapreduce.jdbc.password' = 'hr',
  'mapreduce.jdbc.input.table.name' = 'EMPLOYEES',
  'oracle.hcat.osh.splitterKind' = 'ROW_SPLITTER',
  'oracle.hcat.osh.rowsPerSplit' = '1500'
);

```

CUSTOM_SPLITTER

Use `CUSTOM_SPLITTER` If you want to provide a custom split generation mechanism. You can do this using `CUSTOM_SPLITTER` through `oracle.hcat.osh.splitterKind` property and a `SELECT` statement that emits ROWIDs corresponding to start and end of each split in `oracle.hcat.osh.chunkSQL`.

6.4.1.2 Choosing a Splitter

`SINGLE_SPLITTER` is used by default if no splitter is specified in the table properties for Hive external table, and the target Oracle table is not partitioned.

For an unpartitioned table, the default value of `oracle.hcat.osh.maxSplits` will be 1. For partitioned table, the default value of the same will be 64, and the default splitter will be `PARTITION_SPLITTER`. The default for `maxSplits` is set to limit the number of connections to the Oracle server. To increase this limit, you must increase the value of `oracle.hcat.osh.maxSplits` explicitly in hive table properties.

Use the following guidelines while choosing a splitter kind for a hive external table:

Splitter Kind	Use
<code>SINGLE_SPLITTER</code>	When no parallelism is required.
<code>PARTITION_SPLITTER</code>	Used by default when target table is partitioned
<code>BLOCK_SPLITTER</code>	When Oracle user has <code>SELECT</code> privilege on <code>SYS.DBA_EXTENTS</code> , and target table is not partitioned.
<code>ROW_SPLITTER</code>	When Oracle user does not have <code>SELECT</code> privilege on <code>SYS.DBA_EXTENTS</code> .
<code>CUSTOM_SPLITTER</code>	For fine grain control over generated splits.

6.4.1.3 Predicate Pushdown

Predicate Pushdown is an optimization technique, in which you push predicates (*WHERE* condition) down to be evaluated by Oracle Database at the time of querying. This minimizes the amount of data fetched from Oracle Database to Hive, while performing a query.

Set the configuration property `hive.optimize.ppd` to either `true` or `false` for enabling Predicate Pushdown. The default value on `hive-1.1.0` is set to `true`. Hence, Predicate Pushdown is always performed, unless you want to disable it.

Note:

OD4H does not push down all possible predicates. It considers only the part of the execution plan pertaining to Oracle table declared as external table. OD4H also rewrites sub-queries for the Oracle SQL engine and each split task. At present conditions involving operators `>`, `=`, `<` in a single condition over a column (e.g. `key > 10`) or a combination of multiple conditions separated by `AND` (e.g. `key > 10 AND key < 20 AND key != 17`) are pushed down.

Another option to reduce the amount of data fetched from the Oracle Database is to specify a condition at the time of table creation, using `TBLPROPERTY` `mapreduce.jdbc.input.conditions`. For instance:

```
mapreduce.jdbc.input.conditions = 'key > 10 OR key = 0'.
```

This will restrict the rows fetched from Oracle Database whenever any query is performed based on the condition specified. The external table that gets created, is analogous to a view on Oracle Database. This approach is only useful when you want to push down complex predicates that cannot be analyzed and automatically pushed down by OD4H.

Table Level Predicate Pushdown

For Table Level Predicate Pushdown to be enabled, you must specify a condition at the time of table creation, using `TBLPROPERTY` `mapreduce.jdbc.input.conditions`.

Following is an illustration:

```
mapreduce.jdbc.input.conditions = 'key > 10 OR key = 0'.
```

This will restrict the rows fetched from Oracle Database when any query is performed based on the condition specified. The table created will be analogous to a view on Oracle database.

However, Table Level Predicate Pushdown is ignored when a predicate (*awhere* clause) is specified in the query.

6.4.1.4 Projection Pushdown

Projection Pushdown is an optimization technique that fetches only the required columns from Oracle Database when a query is performed. If you want to fetch all columns during a query (not recommended), you can disable it by setting the `hive.io.file.read.all.columns` connection property to `true`. On `Hive-1.1.0`, this property is `false` by default.

6.4.1.5 Partition Pruning

If you refer to Employee Data Partition table, the partitions irrelevant to the query are removed from the partition list. This is done by executing an explain plan on the query to obtain the list of partitions and sub-partitions that are relevant to the query.

Table level partition pruning uses table level predicate pushdown, on the other hand partition pruning at the query level uses query level predicate pushdown.

Partition pruning is active when a `SELECT` query is run, in which the `WHERE` clause uses the partitioning key. Following is an example of partition pruning:

To query the partition, where salary is in the above range and prune other partitions, perform the following:

Hive External Table:

```
CREATE EXTERNAL TABLE EmployeeDataPartitioned (
  Emp_ID int,
  First_Name string,
  Last_Name string,
  Job_Title string,
  Salary int
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
WITH SERDEPROPERTIES (
  'oracle.hcat.osh.columns.mapping' =
  'Emp_ID,First_Name,Last_Name,Job_Title,Salary'
)
TBLPROPERTIES (
  'mapreduce.jdbc.url' = '${hiveconf:connection_string}',
  'mapreduce.jdbc.username' = '${hiveconf:oracle_user}',
  'mapreduce.jdbc.password' = '${hiveconf:oracle_pwd}',
  'mapreduce.jdbc.input.table.name' = 'EmployeeData',
  'oracle.hcat.osh.oosKind' = 'PARTITIONED_TABLE'
);
```

The following `SELECT` statement shows how to query the partition, where salary is between 72000 to 78000, and prunes other partitions:

```
select * from EmployeeDataPartitioned where salary > 72000 and salary < 78000;
```

6.4.2 Smart Connection Management

Connection Caching

Each map task runs in its own JVM. Each JVM in turn caches a single connection to the Oracle database that you can reuse within the same query. The Mapper checks the cache before establishing a new connection and caching is not done once the query has completed executing.

Oracle RAC Awareness

JDBC and UCP are aware of various Oracle RAC instances. This can be used to split queries submitted to JDBC. The StorageHandler will depend on listener for load balancing.

Handling Logon Storms

Hadoop allows you to limit the number of mappers attempting to connect to the Database. Hadoop allows you to limit the number of mappers attempting to connect to the Database using `oracle.hcat.osh.maxSplits`. This parameter controls the degree of concurrency. However, subsequent tasks of the same query are guaranteed to query their table granule as per the System Commit Number (SCN) of the query. This ensures consistency of the result sets.

Database Resident Connection Pooling (DRCP)

It is recommended to configure DRCP for OD4H, and limit the maximum number of concurrent connections to the Oracle Database from OD4H.

Configuring Database Resident Connection Pooling

To configure DRCP, use the following steps:

1. Login as SYSDBA.
2. Start the default pool, `SYS_DEFAULT_CONNECTION_POOL` using `DBMS_CONNECTION_POOL.START_POOL` with the default settings.

You can use `DBMS_CONNECTION_POOL.MINSIZE` and `DBMS_CONNECTION_POOL.MAXSIZE` with the default settings.

Note: Oracle Database Administrator's Guide for more information on Configuring DRCP.

6.4.3 Security Features

Following are the security features of OD4H:

6.4.3.1 Improved Authentication

OD4H uses Oracle JDBC driver for connecting to Oracle Database. It provides all authentication methods supported by Oracle JDBC. OD4H supports authentication through use of basic authentication (username and password), Oracle Wallet, and Kerberos. You can specify the authentication to be used for a table created in Hive, through the `oracle.hcat.osh.authentication` table property. This is useful only for strong authentication.

- Kerberos
- Oracle Wallet
- Basic Authentication

Note: Oracle recommends using strong authentication such as Kerberos.

The various authentication processes are described with examples as follows:

1. Kerberos

Uses Kerberos credentials of the Hadoop engine process. This `principal` should have access to the table.

See Also:

[Oracle Database JDBC Developer's Guide](#) for information on configuring database for Kerberos and details of client parameters

You can enable Kerberos configuration on Hive, by adding to `hive-env.sh` the following:

```
export HADOOP_OPTS="$HADOOP_OPTS -Djava.security.krb5.conf=<path to kerberos configuration>
```

To enable child JVMs to use Kerberos configuration, edit the `mapred-site.xml` to include the following property on all nodes of the cluster:

```
<property><name>mapred.child.java.opts</name> <value>-Djava.security.krb5.conf=<path to kerberos configuration></value></property>
```

Enable these configurations on BDA using Cloudera manager..

Following is an illustration of Kerberos authentication:

```
CREATE EXTERNAL TABLE kerb_example (
  id DECIMAL,
  name STRING,
  salary DECIMAL
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
WITH SERDEPROPERTIES (
  'oracle.hcat.osh.columns.mapping' = 'id,name,salary')
TBLPROPERTIES (
  'mapreduce.jdbc.url' = 'jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)
(HOST=adc*****.xxxxxx.com)(PORT=5521))(CONNECT_DATA=
(SERVICE_NAME=project_name.xxx.rdbms.xxxx.com)))',
  'mapreduce.jdbc.input.table.name' = 'kerb_example',
  'mapreduce.jdbc.username' = 'CLIENT@xxxxxx.COM',
  'oracle.hcat.osh.authentication' = 'KERBEROS',
  'oracle.net.kerberos5_cc_name' = '/tmp/krb5cc_xxxxx',
  'java.security.krb5.conf' = '/home/user/kerberos/krb5.conf',
  'oracle.hcat.osh.kerb.callback' = 'KrbCallbackHandler',
  'sun.security.krb5.principal' = 'CLIENT@xxxxxx.COM'
);
```

The path specified in `oracle.security.krb5.conf` should be accessible to all nodes of the cluster. These paths should also match with the path of the corresponding properties in Oracle Database `sqlnet.ora`. The keytab path provided in `sqlnet.ora` should also be accessible from all nodes of the cluster.

If `sun.security.krb5.principal` is not specified, OD4H will attempt to authenticate using default principal in Credential Cache specified by the `oracle.net.kerberos5_cc_name` property.

Note:

The callback will be called only if the principal cannot be authenticated using a ticket obtained from the credential cache specified in `oracle.net.kerberos5_cc_name` property.

A simple callback handler class is described as follows (The callback class must be available to the hive classpath):

```
class KrbCallbackHandler
    implements CallbackHandler{

    @Override
    public void handle(Callback[] callbacks) throws IOException,
        UnsupportedOperationException{
        for (int i = 0; i < callbacks.length; i++){
            if (callbacks[i] instanceof PasswordCallback){
                PasswordCallback pc = (PasswordCallback)callbacks[i];
                System.out.println("set password to 'welcome'");
                pc.setPassword((new String("welcome")).toCharArray());
            } else if (callbacks[i] instanceof NameCallback) {
                ((NameCallback)callbacks[i]).setName("client@xxxxx.COM");
            }else{
                throw new UnsupportedOperationException(callbacks[i],
                    "Unrecognized Callback");
            }
        }
    }
}
```

2. Oracle Wallet

The wallet should be available in the OS environment of each engine process. Following is an illustration of how to add Wallet authentication:

```
CREATE EXTERNAL TABLE wallet_example (
    id DECIMAL,
    name STRING,
    salary DECIMAL
)
STORED BY 'oracle.hcat.osh.OracleStorageHandler'
WITH SERDEPROPERTIES (
    'oracle.hcat.osh.columns.mapping' = 'id,name,salary')
TBLPROPERTIES (
    'mapreduce.jdbc.url' = 'jdbc:oracle:thin:@inst1',
    'mapreduce.jdbc.input.table.name' = 'wallet_example',
    'oracle.hcat.osh.authentication' = 'ORACLE_WALLET',
    'oracle.net.tns_admin' = '/scratch/user/view_storage/user_project6/work',
    'oracle.net.wallet_location' = '/scratch/user/view_storage/user_project6/work'
);
```

Note: The paths specified in `oracle.net.tns_admin` and `oracle.net.wallet_location` should be accessible from all nodes of the cluster.

See Also:

Managing the Secure External Password Store for Password Credentials section in the *Oracle Database Security Guide*.

3. Basic Authentication (for demo purposes only)

This is stored in HCatalog TBLPROPERTIES or supplied on HiveQL SELECT statement.

When Basic Authentication is used, the username and password for Oracle Schema is specified in Hive external Table properties.

Note:

Oracle does not recommend this in the production environment, since the password is stored in clear in HCatalog.

6.5 Using HiveQL with OD4H

HiveQL is a SQL like language provided by Hive. It can be used to query hive external tables created using OD4H.

You can run the Resource Manager web interface in your browser (<http://bigdatalite.localdomain:8088/cluster>), to track the status of a running query on BDA.

You can also see the logs of a query in Cloudera Manager, which also indicates the actual query sent to Oracle Database corresponding to your query on HiveQL. Hive and OD4H use slf4j framework for logging. You can control logging level for OD4H related classes using logging configuration techniques of Hive.

6.6 Using Spark SQL with OD4H

Spark SQL enables relational queries expressed in SQL and HiveSQL to be executed using Spark. Spark SQL allows you to mix SQL queries with programmatic data manipulations supported by RDDs (Resilient Distributed Datasets) in Java, Python and Scala, with a single application.

Spark SQL enables you to submit relational queries using SQL or HiveQL. You can also use it to query external tables created using OD4H.

Perform the following steps to configure Spark-SQL on BigDataLite-4.2 VM, before running queries:

1. Add `ojdbc7.jar` and `osh.jar` to CLASSPATH in `/usr/lib/spark/bin/compute-classpath.sh`

```
CLASSPATH="$CLASSPATH:/opt/oracle/od4h/lib/osh.jar"
CLASSPATH="$CLASSPATH:/opt/oracle/od4h/lib/ojdbc7.jar"
```

2. Edit `SPARK_HOME` in `/usr/lib/spark/conf/spark-env.sh`

```
export SPARK_HOME=/usr/lib/spark:/etc/hive/conf
```

3. You will need to specify additional environment variables in `/usr/lib/spark/conf/spark-env.sh`.

The Hive related variables that need to be added are marked in bold. The file already contains Hadoop related environment variables.

```
export DEFAULT_HADOOP=/usr/lib/hadoop
export DEFAULT_HIVE=/usr/lib/hive
export DEFAULT_HADOOP_CONF=/etc/hadoop/conf
export DEFAULT_HIVE_CONF=/etc/hive/conf
export HADOOP_HOME=${HADOOP_HOME:-$DEFAULT_HADOOP}
export HADOOP_HDFS_HOME=${HADOOP_HDFS_HOME:-${HADOOP_HOME}/../hadoop-hdfs}
export HADOOP_MAPRED_HOME=${HADOOP_MAPRED_HOME:-${HADOOP_HOME}/../hadoop-
```

```
mapreduce}
export HADOOP_YARN_HOME=${HADOOP_YARN_HOME:-${HADOOP_HOME}/../hadoop-yarn}
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-$DEFAULT_HADOOP_CONF}
export HIVE_CONF_DIR=${HIVE_CONF_DIR:-$DEFAULT_HIVE_CONF}

CLASSPATH="$CLASSPATH:$HIVE_CONF_DIR"
CLASSPATH="$CLASSPATH:$HADOOP_CONF_DIR"

if [ "x" != "x$YARN_CONF_DIR" ]; then
    CLASSPATH="$CLASSPATH:$YARN_CONF_DIR"
fi

# Let's make sure that all needed hadoop libs are added properly
CLASSPATH="$CLASSPATH:$HADOOP_HOME/client/*"
CLASSPATH="$CLASSPATH:$HIVE_HOME/lib/*"
CLASSPATH="$CLASSPATH:${HADOOP_HOME}/bin/hadoop classpath"
```

Once configured, you can run some sample queries on spark SQL using scripts included in `demo:/shell/*QuerySpark.sh`. By default, Spark prints queries on the console. To modify this behavior you can edit the spark logging configuration file `/usr/lib/spark/conf/log4j.properties`.

The log printed by `OracleRecordReader` shows the actual query sent to Oracle Database, as follows:

```
15/03/18 10:36:08 INFO OracleRecordReader: Reading records from Oracle Table
using Query: SELECT FIRST_NAME, LAST_NAME, EMP_ID FROM EmployeeData
```

6.7 Writing Back to Oracle Database

In the typical use case for OD4H, you store the result sets of Hive or Spark SQL queries back to Oracle Database. OD4H implements `OutputFormat` to enable you to write back to an Oracle Database table from Hadoop.

After the data is inserted into an Oracle Database table, you can then use your favorite business intelligence tools for further data mining

The following query is from the OD4H demo code samples. It demonstrates writing back to an external table called `EmployeeBonusReport`.

Example 6-1 Writing Hive or Spark Result Sets Back to Oracle Database

```
INSERT INTO EmployeeBonusReport
    SELECT EmployeeDataSimple.First_Name, EmployeeDataSimple.Last_Name,
           EmployeeBonus.bonus
FROM EmployeeDataSimple JOIN EmployeeBonus ON
    (EmployeeDataSimple.Emp_ID=EmployeeBonus.Emp_ID)
WHERE salary > 70000 and bonus > 7000"
```

Glossary

Apache Flume

A distributed service for collecting and aggregating data from almost any source into a data store such as HDFS or HBase.

See also [Apache HBase](#); [HDFS](#).

Apache 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.

Apache 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](#); [MapReduce](#).

Apache Sentry

Integrates with the Hive and Impala SQL-query engines to provide fine-grained authorization to data and metadata stored in Hadoop.

Apache Solr

Provides an enterprise search platform that includes full-text search, faceted search, geospatial search, and hit highlighting.

Apache Spark

A fast engine for processing large-scale data. It supports Java, Scala, and Python applications. Because it provides primitives for in-memory cluster computing, it is particularly suited to machine-learning algorithms. It promises performance up to 100 times faster than MapReduce.

Apache 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.

Apache YARN

An updated version of MapReduce, also called MapReduce 2. The acronym stands for Yet Another Resource Negotiator.

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](#).

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 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](#); [Apache 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 Manager

Cloudera Manager enables you to monitor, diagnose, and manage CDH services in a cluster.

The Cloudera Manager agents on Oracle Big Data Appliance also provide information to Oracle Enterprise Manager, which you can use to monitor both software and hardware.

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. Based on Apache Solr.

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

See [Apache Flume](#).

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

See [Apache HBase](#).

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

See [Apache Hive](#).

Hive Thrift

A remote procedure call (RPC) interface for remote access to CDH for Hive queries.

See also [CDH](#); [Apache Hive](#).

HiveQL

A SQL-like query language used by Hive.

See also [Apache 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 tasks to specific nodes in the CDH cluster, preferably those nodes storing the data. MRv1 only.

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. Two versions of MapReduce are available, MapReduce 1 and YARN (MapReduce 2). The default version on Oracle Big Data Appliance 3.0 and later is YARN.

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 and not supported by all applications).

See also [Apache YARN](#).

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](#).

NodeManager

A service that runs on each node and executes the tasks assigned to it by the ResourceManager. YARN only.

See also [ResourceManager](#); [YARN](#).

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

Oracle Linux is Oracle's commercial version of the Linux operating system. Oracle Linux is free to download, use, and redistribute without a support contract.

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](#).

ResourceManager

A service that assigns tasks to specific nodes in the CDH cluster, preferably those nodes storing the data. YARN only.

See also [Hadoop](#); [YARN](#).

Search

See [Cloudera Search](#).

Sentry

See [Apache Sentry](#).

Solr

See [Apache Solr](#).

Spark

See [Apache Spark](#).

Sqoop

See [Apache Sqoop](#).

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. MRv1 only.

See also [JobTracker](#).

Whirr

Apache Whirr is a set of libraries for running cloud services.

YARN

See [Apache YARN](#).

ZooKeeper

A MapReduce 1 centralized coordination service for CDH distributed processes that maintains configuration information and naming, and provides distributed synchronization and group services.

Index

A

activity reports, [2-41](#)
Apache Sentry, [2-35](#)
application adapters, [1-9](#)
applications
 data pull, [4-1](#)
 data push, [4-2](#)
Audit Vault plug-in, [2-41](#)
auditing data collected from services, [2-40](#)
authentication, [3-1](#)
authorization, [2-35](#)
autoAnalyze configuration property, [5-18](#)
autoAnalyze property, [5-7](#)
autoBalance configuration property, [5-18](#)
Automated Service Manager, [2-39](#)

B

bddiag utility, [2-42](#)
Berkeley DB, [1-6](#)
best practices, [5-1](#)
big data description, [1-1](#)
business intelligence, [1-3](#), [1-5](#), [1-10](#)
byteWeight configuration property, [5-18](#)

C

CDH
 about, [1-3](#)
 diagnostics, [2-42](#)
 file system, [1-5](#)
 remote client access, [3-2](#)
 security, [3-1](#)
 version, [2-10](#)
chopped keys, [5-18](#)
chunking files, [1-5](#)
client access
 HDFS cluster, [3-3](#)
 HDFS secured cluster, [3-4](#)
 Hive, [3-7](#)
client configuration, [3-2](#)
Cloudera Manager

Cloudera Manager (*continued*)

 about, [2-3](#)
 accessing administrative tools, [2-5](#)
 connecting to, [2-3](#)
 effect of hardware failure on, [2-24](#)
 software dependencies, [2-24](#)
 starting, [2-3](#)
 UI overview, [2-4](#)
 version, [2-10](#)
Cloudera's Distribution including Apache Hadoop,
 [1-5](#)
clusters, definition, [1-3](#)
confidence configuration property, [5-18](#)
Counting Reducer, [5-2](#)

D

data replication, [1-5](#)
data skew, [5-1](#)
DataNode, [2-22](#)
diagnostics, collecting, [2-42](#)
disks, [2-21](#)
dnsmasq service, [4-4](#)
duplicating data, [1-5](#)

E

emcli utility, [2-3](#)
enableSorting configuration property, [5-18](#)
encryption, [2-35](#), [2-36](#)
engineered systems, [1-3](#)
Exadata Database Machine, [1-3](#)
Exadata InfiniBand connections, [4-2](#)
Exalytics In-Memory Machine, [1-3](#)
external tables, [1-8](#)

F

failover
 JobTracker, [2-20](#)
 NameNode, [2-19](#)
feedbackDir configuration property, [5-18](#)

files, recovering HDFS, [3-11](#)
first NameNode, [2-23](#)
Flume, [2-11](#)
ftp.oracle.com, [2-42](#)

G

groups, [2-34](#), [3-8](#)

H

Hadoop Distributed File System, [1-4](#)
hadoop group, [3-8](#)
Hadoop version, [1-3](#)
HADOOP_CLASSPATH environment variable, [5-18](#)
HBase, [2-11](#)
HDFS
 about, [1-4](#), [1-5](#)
 auditing, [2-40](#)
HDFS Transparent Encryption, [2-35](#)
help from Oracle Support, [2-42](#)
Hive
 about, [1-5](#)
 auditing, [2-40](#)
 client access, [3-7](#)
 node location, [2-24](#)
 software dependencies, [2-24](#)
 tables, [3-9](#)
 user identity, [2-34](#)
hive group, [3-8](#)
HiveQL, [1-5](#)
HotSpot, [2-10](#)
HTTPS/Network Encryption, [2-36](#)
Hue
 users, [3-9](#)

I

Impala, [2-11](#)
InfiniBand connections to Exadata, [4-2](#)
InfiniBand network configuration, [4-1](#)
inputFormat.mapred.* configuration properties, [5-18](#)
installing CDH client, [3-2](#)

J

Java HotSpot Virtual Machine, [2-10](#)
Job Analyzer, [5-2](#), [5-4](#)
job duration, [5-1](#)
jobconfPath property, [5-18](#)
jobHistoryPath configuration property, [5-18](#)
JobTracker
 failover, [2-20](#)
 security, [3-1](#)
JobTracker node, [2-24](#)

K

Kerberos authentication, [3-1](#)
Kerberos commands, [3-2](#)
Kerberos user setup, [3-10](#)
key chopping, [5-2](#)
key-value database, [1-6](#)
keyLoad.minChopBytes configuration property, [5-18](#)
keys, assigning to reducers, [5-2](#)
keyWeight configuration property, [5-18](#)
knowledge modules, [1-9](#)

L

linearKeyLoad.* configuration properties, [5-18](#)
Linux
 disk location, [2-21](#)
 installation, [2-9](#)
load, [5-1](#)
Load Balancer, [5-2](#)
loading data, [1-8](#)
login privileges, [3-10](#)

M

map.tasks property, [5-18](#)
mapper workload, [5-2](#)
mapred configuration properties, [5-18](#)
mapred.map.tasks configuration property, [5-23](#)
MapReduce, [1-4](#), [1-7](#), [2-40](#), [3-1](#), [3-8](#)
mapreduce configuration properties, [5-18](#)
max.split.size configuration property, [5-18](#)
maxLoadFactor configuration property, [5-18](#)
maxSamplesPct configuration property, [5-18](#)
minChopBytes configuration property, [5-18](#)
minSplits configuration property, [5-18](#)
monitoring activity, [2-41](#)
multitrack clusters
 service locations, [2-15](#)
MySQL Database
 about, [2-24](#)
 port number, [2-39](#)
 user identity, [2-34](#)
 version, [2-10](#)

N

NameNode
 first, [2-23](#)
NameNode failover, [2-19](#)
NoSQL databases, [1-6](#)
numThreads configuration property, [5-18](#)

O

OASM, port number, [2-39](#)

- ODI, [1-8](#)
- oinstall group, [3-8](#)
- Oozie
 - auditing, [2-40](#)
 - software dependencies, [2-24](#)
- operating system users, [2-34](#)
- Oracle Audit Vault and Database Firewall, [2-40](#)
- Oracle Automated Service Manager, [2-39](#)
- Oracle Big Data SQL
 - general description, [1-7](#)
- Oracle Data Integrator
 - about, [1-8](#), [1-9](#)
 - node location, [2-24](#)
 - software dependencies, [2-24](#)
- Oracle Data Integrator agent, [2-39](#)
- Oracle Database Instant Client, [2-10](#)
- Oracle Exadata Database Machine, [1-3](#), [4-1](#)
- Oracle Exalytics In-Memory Machine, [1-3](#)
- Oracle Linux
 - about, [1-3](#)
 - relationship to HDFS, [1-4](#)
 - version, [2-10](#)
- Oracle Loader for Hadoop, [1-8](#), [2-10](#)
- Oracle NoSQL Database
 - about, [1-6](#), [1-9](#)
 - port numbers, [2-39](#)
 - version, [2-10](#)
- Oracle R Advanced Analytics for Hadoop, [1-9](#), [2-10](#)
- Oracle R Enterprise, [1-9](#)
- Oracle SQL Connector for HDFS, [1-8](#)
- Oracle Support, creating a service request, [2-42](#)
- oracle user, [3-8](#)
- Oracle XQuery for Hadoop, [1-9](#), [2-10](#)
- oracle.hadoop.balancer.* configuration properties, [5-18](#)
- oracle.hadoop.balancer.autoAnalyze property, [5-7](#)
- oracle.hadoop.balancer.Balancer class, [5-15](#)
- oracle.hadoop.balancer.KeyLoadLinear class, [5-25](#)
- out of heap space errors, [5-15](#)

P

- partitioning, [2-21](#), [5-2](#)
- Perfect Balance
 - application requirements, [5-2](#)
 - basic steps, [5-3](#)
 - description, [5-1](#)
- planning applications, [1-3](#)
- port map, [2-39](#)
- port numbers, [2-38](#), [2-39](#)
- pulling data into Exadata, [4-1](#)
- puppet
 - port numbers, [2-39](#)
 - security, [2-39](#)
- puppet master
 - node location, [2-23](#)

- pushing data into Exadata, [4-2](#)

R

- R Connector, [1-9](#)
- R distribution, [2-10](#)
- R language support, [1-9](#)
- range partitioning, [5-2](#)
- recovering HDFS files, [3-11](#)
- reducer load, [5-1](#)
- remote client access, [3-2](#), [3-7](#)
- replicating data, [1-5](#)
- report.overwrite configuration property, [5-18](#)
- reportPath configuration property, [5-18](#)
- resource management, [2-12](#), [2-32](#)
- rowWeight configuration property, [5-18](#)
- rpc.statd service, [2-39](#)

S

- SDP listener configuration, [4-5](#)
- SDP over InfiniBand, [4-1](#)
- SDP, enabling on Exadata, [4-4](#)
- Search, [2-11](#)
- security, [2-33](#)
- Sentry, [2-35](#)
- service requests, creating for CDH, [2-42](#)
- service tags, [2-39](#)
- services
 - auditing, [2-40](#)
 - node locations, [2-12](#), [2-13](#)
- skew, [5-1](#)
- Sockets Direct Protocol, [4-1](#)
- software components, [2-9](#)
- software framework, [1-3](#)
- software services
 - node locations, [2-12](#), [2-13](#)
 - port numbers, [2-38](#)
- Spark, [2-11](#)
- Sqoop, [2-11](#)
- ssh service, [2-39](#)

T

- tables, [1-8](#), [3-9](#)
- tmpDir configuration property, [5-18](#)
- tools.* configuration properties, [5-18](#)
- trash facility, [3-11](#)
- trash facility, disabling, [3-12](#)
- trash interval, [3-11](#)
- troubleshooting CDH, [2-42](#)

U

- uploading diagnostics, [2-42](#)
- useClusterStats configuration property, [5-18](#)

useMapreduceApi configuration property, [5-18](#)
user accounts, [3-8](#)
user groups, [3-8](#)
users
 Cloudera Manager, [2-5](#)
 operating system, [2-34](#)

W

writeKeyBytes configuration property, [5-18](#)

X

xinetd service, [2-39](#)

XQuery connector, [1-9](#)

Y

YARN support, [1-7](#)

Z

zones, [2-35](#)