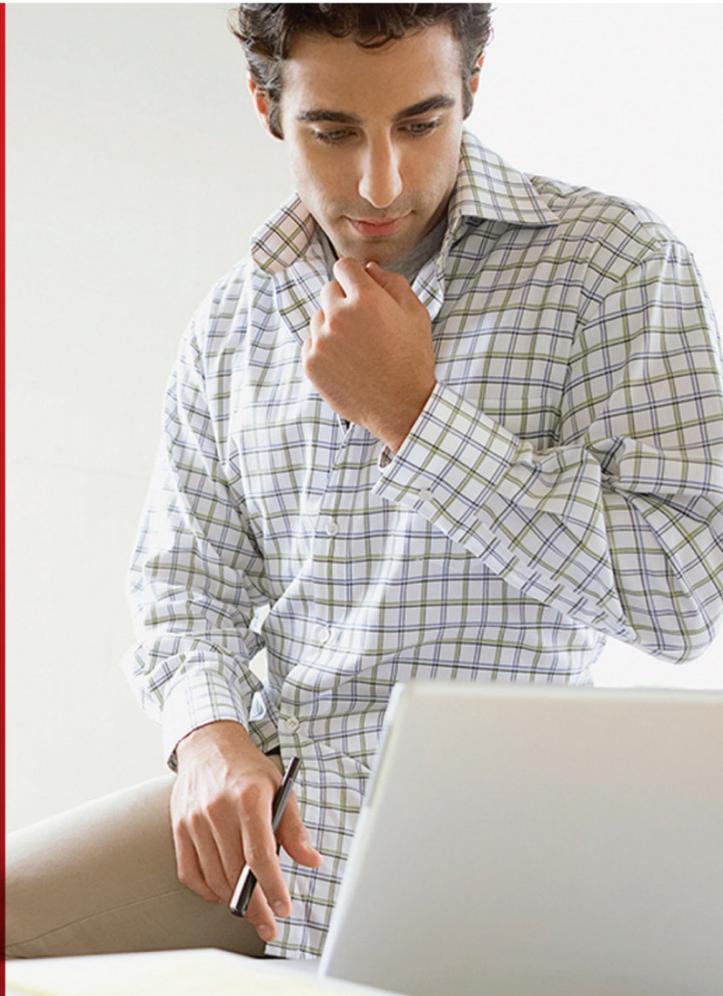




Hardware and Software
Engineered to Work Together



Oracle University and you. You are not a Valid Partner use only

Oracle Big Data Fundamentals

Student Guide – Volume 1

D86898GC10

Edition 1.0 | May 2015 | D91413

Learn more from Oracle University at oracle.com/education/

Authors

Lauran K. Serhal
Brian Pottle
Suresh Mohan

Technical Contributors and Reviewers

Marty Gubar
Melliyal Annamalai
Sharon Stephen
Jean-Pierre Dijcks
Bruce Nelson
Daniel W McClary
Josh Spiegel
Anuj Sahni
Dave Segleau
Ashwin Agarwal
Salome Clement
Donna Carver
Alex Kotopoulos
Marcos Arancibia
Mark Hornick
Charlie Berger
Ryan Stark
Swarnapriya Shridhar
Branislav Valny
Dmitry Lychagin
Mirella Tumolo
S. Matt Taylor
Lakshmi Narapareddi
Drishya Tm

Graphic Editors

Rajiv Chandrabhanu
Maheshwari Krishnamurthy

Editors

Malavika Jinka
Smita Kommini
Arijit Ghosh

Publishers

Veena Narasimhan
Michael Sebastian Almeida
Syed Ali

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

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

Contents

1 Introduction

Objectives	1-2
Questions About You	1-3
Course Objectives	1-4
Course Road Map: Module 1 Big Data Management System	1-5
Course Road Map: Module 2 Data Acquisition and Storage	1-6
Course Road Map: Module 3 Data Access and Processing	1-7
Course Road Map: Module 4 Data Unification and Analysis	1-8
Course Road Map: Module 5 Using and Managing Oracle Big Data Appliance	1-9
The Oracle Big Data Lite Virtual Machine (Used in this Course) Home Page	1-10
Connecting to the Practice Environment	1-11
Starting the Oracle Big Data Lite Virtual Machine (VM) Used in this Course	1-12
Starting the Oracle Big Data Lite (BDLite) Virtual Machine (VM) Used in this Course	1-13
Accessing the Getting Started Page from the Oracle BDLite VM	1-14
Accessing the Practice Files	1-15
Accessing the /home/oracle/exercises Directory	1-16
Accessing /home/oracle/movie Directory	1-17
Appendices	1-18
Oracle Big Data Appliance Documentation	1-19
Additional Resources: Oracle Big Data Tutorials on Oracle Learning Library (OLL)	1-21
Practice 1-1: Overview	1-23
Summary	1-24

2 Big Data and the Oracle Information Management System

Course Road Map	2-2
Lesson Objectives	2-3
Big Data: A Strategic IM Perspective	2-4
Big Data	2-5
Characteristics of Big Data	2-6
Importance of Big Data	2-8
Big Data Opportunities: Some Examples	2-9
Big Data Challenges	2-10
Information Management Landscape	2-12

Extending the Boundaries of Information Management	2-13
A Simple Functional Model for Big Data	2-14
Oracle Information Management Conceptual Architecture	2-16
IM Architecture Design Pattern: Discovery Lab	2-18
IM Architecture Design Pattern: Information Platform	2-19
IM Architecture Design Pattern: Data Application	2-20
IM Architecture Design Pattern: Information Solution	2-21
IM Architecture Design Pattern: Real-Time Events	2-22
Design Patterns to Component Usage Map	2-23
Big Data Adoption and Implementation Patterns	2-24
IM Architecture Data Approaches: Schema-on-Write vs Schema-on-Read	2-26
Course Approach: Big Data Project Phases	2-28
Goal of Oracle's IM System for Big Data	2-29
Additional Resources	2-30
Summary	2-31

3 Using Oracle Big Data Lite Virtual Machine

Course Road Map	3-2
Objectives	3-3
Lesson Agenda	3-4
Oracle Big Data Lite Virtual Machine: Introduction	3-5
Oracle Big Data Lite 4.0.1 VM Components	3-6
Initializing the Environment for the Oracle Big Data Lite VM	3-7
Initializing the Environment	3-8
Lesson Agenda	3-9
Oracle MoviePlex Case Study: Introduction	3-10
Introduction	3-11
Big Data Challenge	3-12
Derive Value from Big Data	3-13
Oracle MoviePlex: Goal	3-14
Oracle MoviePlex: Big Data Challenges	3-15
Oracle MoviePlex: Architecture	3-16
Oracle MoviePlex: Data Generation	3-17
Oracle MoviePlex: Data Generation Format	3-18
Oracle MoviePlex Application	3-19
Summary	3-20
Practice 3: Overview	3-21

- 4 Introduction to the Big Data Ecosystem**
 - Course Road Map 4-2
 - Objectives 4-3
 - Computer Clusters 4-4
 - Distributed Computing 4-5
 - Apache Hadoop 4-6
 - Types of Analysis That Use Hadoop 4-7
 - Apache Hadoop Ecosystem 4-8
 - Apache Hadoop Core Components 4-9
 - HDFS Key Definitions 4-11
 - NameNode (NN) 4-12
 - DataNodes (DN) 4-13
 - MapReduce Framework 4-14
 - Benefits of MapReduce 4-15
 - MapReduce Job 4-16
 - Simple Word Count MapReduce: Example 4-17
 - MapReduce Versions 4-19
 - Choosing a Hadoop Distribution and Version 4-20
 - Additional Resources: Cloudera Distribution 4-21
 - Additional Resources: Apache Hadoop 4-22
 - Cloudera's Distribution Including Apache Hadoop (CDH) 4-23
 - CDH Architecture 4-24
 - CDH Components 4-25
 - CDH Architecture 4-26
 - CDH Components 4-28
 - Where to Go for More Information? 4-29
 - Summary 4-30
- 5 Introduction to the Hadoop Distributed File System (HDFS)**
 - Course Road Map 5-2
 - Objectives 5-3
 - Agenda 5-4
 - HDFS: Characteristics 5-5
 - HDFS Deployments: High Availability (HA) and Non-HA 5-7
 - HDFS Key Definitions 5-8
 - NameNode (NN) 5-9
 - Functions of the NameNode 5-10
 - Secondary NameNode (Non-HA) 5-11
 - DataNodes (DN) 5-12
 - Functions of DataNodes 5-13
 - NameNode and Secondary NameNodes 5-14

Storing and Accessing Data Files in HDFS	5-15
Secondary NameNode, Checkpoint Node, and Backup Node	5-16
HDFS Architecture: HA	5-17
HDFS High Availability (HA) Using the Quorum Journal Manager (QJM)	5-18
HDFS High Availability (HA) Using the Quorum Journal Manager (QJM) Feature	5-19
Configuring an HA Cluster Hardware Resources	5-22
Enabling HDFS HA	5-23
Data Replication Rack-Awareness in HDFS	5-25
Data Replication Process	5-26
Accessing HDFS	5-27
Agenda	5-28
HDFS Commands	5-29
The File System Namespace: The HDFS FS (File System) Shell Interface	5-30
Accessing HDFS	5-32
FS Shell Commands	5-33
Basic File System Operations: Examples	5-34
Sample FS Shell Commands	5-35
Basic File System Operations: Examples	5-36
HDFS Administration Commands	5-38
Using the hdfs fsck Command: Example	5-39
HDFS Features and Benefits	5-40
Summary	5-41
Practice 5: Overview	5-42

6 Acquire Data Using CLI, Fuse DFS, and Flume

Course Road Map	6-2
Objectives	6-3
Reviewing the Command Line Interface (CLI)	6-4
Viewing File System Contents Using the CLI	6-5
Loading Data Using the CLI	6-6
What is Fuse DFS?	6-7
Enabling Fuse DFS on Big Data Lite	6-8
Using Fuse DFS	6-9
What is Flume?	6-10
Flume: Architecture	6-11
Flume Sources (Consume Events)	6-12
Flume Channels (Hold Events)	6-13
Flume Sinks (Deliver Events)	6-14
Flume: Data Flows	6-15
Configuring Flume	6-16

Exploring a flume*.conf File 6-17

Additional Resources 6-18

Summary 6-19

Practice 6: Overview 6-20

7 Acquire and Access Data Using Oracle NoSQL Database

Course Road Map 7-2

Objectives 7-3

What is a NoSQL Database? 7-4

RDBMS Compared to NoSQL 7-5

HDFS Compared to NoSQL 7-6

Oracle NoSQL Database 7-7

Points to Consider Before Choosing NoSQL 7-8

NoSQL Key-Value Data Model 7-9

Acquiring and Accessing Data in a NoSQL DB 7-11

Primary (Parent) Table Data Model 7-12

Table Data Model: Child Tables 7-13

Creating Tables 7-14

Creating Tables: Two Options 7-15

Data Definition Language (DDL) Commands 7-16

CREATE TABLE 7-17

Accessing the CLI 7-19

Executing a DDL Command 7-20

Viewing Table Descriptions 7-21

Recommendation: Using Scripts 7-22

Loading Data Into Tables 7-23

Accessing the KVStore 7-24

Introducing the TableAPI 7-25

Write Operations: put() Methods 7-26

Writing Rows to Tables: Steps 7-27

Constructing a Handle 7-28

Creating Row Object, Adding Fields, and Writing Record 7-29

Reading Data from Tables 7-30

Read Operations: get() Methods 7-31

Retrieving Table Data: Steps 7-32

Retrieving Single a Row 7-33

Retrieving Multiple Rows 7-34

Retrieving Child Tables 7-35

Removing Data From Tables 7-36

Delete Operations: 3 TableAPIs 7-37

Deleting Row(s) From a Table: Steps 7-38

Additional Resources 7-39
Summary 7-40
Practice 7 Overview 7-41

8 Primary Administrative Tasks for Oracle NoSQL Database

Course Road Map 8-2
Objectives 8-3
Installation Planning: KVStore Analysis 8-4
InitialCapacityPlanning Spreadsheet 8-5
Planning Spreadsheet Sections 8-6
Next Topic 8-7
Configuration Requirements 8-8
Determine the Number of Shards 8-9
Determine # of Partitions and Replication Factor 8-10
Determine # of Storage Nodes 8-11
Installation and Configuration Steps 8-12
Step 1: Creating Directories 8-13
Step 2: Extracting Software 8-14
Step 3: Verifying the Installation 8-15
Step 4: Configuring Nodes (Using the makebootconfig Utility) 8-16
Using the makebootconfig Utility 8-18
Starting the Storage Node Agents 8-19
Pinging the Replication Nodes 8-20
Next Topic 8-21
Configuration and Monitoring Tools 8-22
Steps to Deploy a KVStore 8-23
Introducing Plans 8-24
States of a Plan 8-25
Starting the Configuration Tool 8-26
Configuring KVStore 8-27
Creating a Zone 8-28
Deploying Storage and Admin Nodes 8-29
Creating a Storage Pool 8-30
Joining Nodes to the Storage Pool 8-31
Creating a Topology 8-32
Deploying the KVStore 8-33
Testing the KVStore 8-34
Additional Resources 8-35
Summary 8-36

9 Introduction to MapReduce

Course Road Map 9-2
Objectives 9-3
MapReduce 9-4
MapReduce Architecture 9-5
MapReduce Version 1 (MRv1) Architecture 9-6
MapReduce Phases 9-7
MapReduce Framework 9-8
Parallel Processing with MapReduce 9-9
MapReduce Jobs 9-10
Interacting with MapReduce 9-11
MapReduce Processing 9-12
MapReduce (MRv1) Daemons 9-13
Hadoop Basic Cluster (MRv1): Example 9-14
MapReduce Application Workflow 9-15
Data Locality Optimization in Hadoop 9-17
MapReduce Mechanics: Deck of Cards Example 9-18
MapReduce Mechanics Example: Assumptions 9-19
MapReduce Mechanics: The Map Phase 9-20
MapReduce Mechanics: The Shuffle and Sort Phase 9-21
MapReduce Mechanics: The Reduce Phase 9-22
Word Count Process: Example 9-23
Submitting a MapReduce 9-24
Summary 9-25
Practice 9: Overview 9-26

10 Resource Management Using YARN

Course Road Map 10-2
Objectives 10-3
Agenda 10-4
Apache Hadoop YARN: Overview 10-5
MapReduce 2.0 (MRv2) or YARN (Yet Another Resource Negotiator)
Architecture 10-7
MapReduce 2.0 (MRv2) or YARN (Yet Another Resource Negotiator)
Daemons 10-8
Hadoop Basic Cluster YARN (MRv2): Example 10-9
YARN Versus MRv1 Architecture 10-10
YARN (MRv2) Architecture 10-11
MapReduce 2.0 (MRv2) or YARN Daemons 10-13
YARN (MRv2) Daemons 10-14
YARN: Features 10-15

Launching an Application on a YARN Cluster	10-16
MRv1 Versus MRv2	10-18
Agenda	10-19
Job Scheduling in YARN	10-20
YARN Fair Scheduler	10-21
Cloudera Manager Resource Management Features	10-23
Static Service Pools	10-25
Working with the Fair Scheduler	10-26
Cloudera Manager Dynamic Resource Management: Example	10-27
Submitting a Job to hrpool By User lucy from the hr Group	10-33
Monitoring the Status of the Submitted MapReduce Job	10-34
Examining the marketingpool	10-35
Submitting a Job to marketingpool By User lucy from the hr Group	10-36
Monitoring the Status of the Submitted MapReduce Job	10-37
Submitting a Job to marketingpool By User bob from the marketing Group	10-38
Monitoring the Status of the Submitted MapReduce Job	10-39
Delay Scheduling	10-40
Agenda	10-41
YARN application Command	10-42
YARN application Command: Example	10-43
Monitoring an Application Using the UI	10-45
The Scheduler: BDA Example	10-46
Summary	10-47
Practice 10	10-48

11 Overview of Hive and Pig

Course Road Map	11-2
Objectives	11-3
Hive	11-4
Use Case: Storing Clickstream Data	11-5
Defining Tables over HDFS	11-6
Hive: Data Units	11-8
The Hive Metastore Database	11-9
Hive Framework	11-10
Creating a Hive Database	11-11
Data Manipulation in Hive	11-12
Data Manipulation in Hive: Nested Queries	11-13
Steps in a Hive Query	11-14
Hive-Based Applications	11-15
Hive: Limitations	11-16
Pig: Overview	11-17

Pig Latin 11-18
Pig Applications 11-19
Running Pig Latin Statements 11-20
Pig Latin: Features 11-21
Working with Pig 11-22
Summary 11-23
Practice 11: Overview 11-24

12 Overview of Cloudera Impala

Course Road Map 12-2
Objectives 12-3
Hadoop: Some Data Access/Processing Options 12-4
Cloudera Impala 12-5
Cloudera Impala: Key Features 12-6
Cloudera Impala: Supported Data Formats 12-7
Cloudera Impala: Programming Interfaces 12-8
How Impala Fits Into the Hadoop Ecosystem 12-9
How Impala Works with Hive 12-10
How Impala Works with HDFS and HBase 12-11
Summary of Cloudera Impala Benefits 12-12
Impala and Hadoop: Limitations 12-13
Summary 12-14

13 Using Oracle XQuery for Hadoop

Course Road Map 13-2
Objectives 13-3
XML 13-4
Simple XML Document: Example 13-5
XML Elements 13-6
Markup Rules for Elements 13-7
XML Attributes 13-8
XML Path Language 13-9
XPath Terminology: Node Types 13-10
XPath Terminology: Family Relationships 13-11
XPath Expressions 13-12
Location Path Expression: Example 13-13
XQuery: Review 13-14
XQuery Terminology 13-15
XQuery Review: books.xml Document Example 13-16
FLWOR Expressions: Review 13-17
Oracle XQuery for Hadoop (OXH) 13-18

OXH Features	13-19
Oracle XQuery for Hadoop Data Flow	13-20
Using OXH	13-21
OXH Installation	13-22
OXH Functions	13-23
OXH Adapters	13-24
Running a Query: Syntax	13-25
OXH: Configuration Properties	13-26
XQuery Transformation and Basic Filtering: Example	13-27
Viewing the Completed Application in YARN	13-30
Calling Custom Java Functions from XQuery	13-31
Additional Resources	13-32
Summary	13-33
Practice 13: Overview	13-34

14 Overview of Solr

Course Road Map	14-2
Objectives	14-3
Apache Solr (Cloudera Search)	14-4
Cloudera Search: Key Capabilities	14-5
Cloudera Search: Features	14-6
Cloudera Search Tasks	14-8
Indexing in Cloudera Search	14-9
Types of Indexing	14-10
The solrctl Command	14-12
SchemaXML File	14-13
Creating a Solr Collection	14-14
Using OXH with Solr	14-15
Using Solr with Hue	14-16
Summary	14-18
Practice 14: Overview	14-19

15 Apache Spark

Course Road Map	15-2
Objectives	15-3
Apache Spark	15-4
Introduction to Spark	15-5
Spark: Components for Distributed Execution	15-6
Resilient Distributed Dataset (RDD)	15-7
RDD Operations	15-8
Characteristics of RDD	15-9

Directed Acyclic Graph Execution Engine 15-10
Scala Language: Overview 15-11
Scala Program: Word Count Example 15-12
Spark Shells 15-13
Summary 15-14
Practice 15: Overview 15-15

16 Options for Integrating Your Big Data

Course Road Map 16-2
Objectives 16-3
Unifying Data: A Typical Requirement 16-4
Introducing Data Unification Options 16-6
Data Unification: Batch Loading 16-7
Sqoop 16-8
Oracle Loader for Hadoop (OLH) 16-9
Copy to BDA 16-10
Data Unification: Batch and Dynamic Loading 16-11
Oracle SQL Connector for Hadoop 16-12
Data Unification: ETL and Synchronization 16-13
Oracle Data Integrator with Big Data Heterogeneous Integration with
Hadoop Environments 16-14
Data Unification: Dynamic Access 16-16
Oracle Big Data SQL: A New Architecture 16-17
When To Use Different Oracle Technologies? 16-18
Summary 16-19

17 Overview of Apache Sqoop

Course Road Map 17-2
Objectives 17-3
Apache Sqoop 17-4
Sqoop Components 17-5
Sqoop Features 17-6
Sqoop: Connectors 17-7
Importing Data into Hive 17-8
Sqoop: Advantages 17-9
Summary 17-10

18 Using Oracle Loader for Hadoop (OLH)

Course Road Map 18-2
Objectives 18-3
Oracle Loader for Hadoop 18-4

Software Prerequisites	18-5
Modes of Operation	18-6
OLH: Online Database Mode	18-7
Running an OLH Job	18-8
OLH Use Cases	18-9
Load Balancing in OLH	18-10
Input Formats	18-11
OLH: Offline Database Mode	18-12
Offline Load Advantages in OLH	18-13
OLH Versus Sqoop	18-14
OLH: Performance	18-15
Summary	18-16
Practice 18: Overview	18-17

19 Using Copy to BDA

Course Road Map	19-2
Objectives	19-3
Copy to BDA	19-4
Requirements for Using Copy to BDA	19-5
How Does Copy to BDA Work?	19-6
Copy to BDA: Functional Steps	19-7
Step 1: Identify the Target Directory	19-8
Step 2: Create an External Table	19-9
Step 3: Copy Files to HDFS	19-10
Step 4: Create a Hive External Table	19-11
Oracle to Hive Data Type Conversions	19-12
Querying the Data in Hive	19-13
Summary	19-14
Practice 19: Overview	19-15

20 Using Oracle SQL Connector for HDFS

Course Road Map	20-2
Objectives	20-3
Oracle SQL Connector for HDFS	20-4
OSCH Architecture	20-5
Using OSCH: Two Simple Steps	20-6
Using OSCH: Creating External Directory	20-7
Using OSCH: Database Objects and Grants	20-8
Using OSCH: Supported Data Formats	20-9
Using OSCH: HDFS Text File Support	20-10
Using OSCH: Hive Table Support	20-12

Using OSCH: Partitioned Hive Table Support 20-14
OSCH: Features 20-15
Parallelism and Performance 20-16
OSCH: Performance Tuning 20-17
OSCH: Key Benefits 20-18
Loading: Choosing a Connector 20-19
Summary 20-20
Practice 20: Overview 20-21

21 Using Oracle Data Integrator and Oracle GoldenGate with Hadoop

Course Road Map 21-2
Objectives 21-3
Oracle Data Integrator 21-4
ODI's Declarative Design 21-5
ODI Knowledge Modules (KMs) Simpler Physical Design / Shorter Implementation Time 21-6
Using ODI with Big Data Heterogeneous Integration with Hadoop Environments 21-7
Using ODI Studio 21-8
ODI Studio Components: Overview 21-9
ODI Studio: Big Data Knowledge Modules 21-10
Using OGG with Big Data 21-12
Resources 21-13
Summary 21-14
Practice 21: Overview 21-15

22 Using Oracle Big Data SQL

Course Road Map 22-2
Objectives 22-3
Barriers to Effective Big Data Adoption 22-4
Overcoming Big Data Barriers 22-5
Oracle Big Data SQL 22-6
Goal and Benefits 22-7
Using Oracle Big Data SQL 22-8
Configuring Oracle Big Data SQL 22-9
Task 1: Create System Directories on Exadata 22-10
Task 2: Deploy Configuration Files 22-11
Task 3: Create Oracle Directory Objects 22-12
Task 4: Install Required Software 22-13
Create External Tables Over HDFS Data and Query the Data 22-14
Using Access Parameters with oracle_hdfs 22-15

Create External Tables to Leverage the Hive Metastore and Query the Data	22-16
Using Access Parameters with oracle_hive	22-17
Automating External Table Creation	22-19
Applying Oracle Database Security Policies	22-20
Viewing the Results	22-21
Applying Redaction Policies to Data in Hadoop	22-22
Viewing Results from the Hive (Avro) Source	22-23
Viewing the Results from Joined RDBMS and HDFS Data	22-24
Summary	22-25
Practice 22: Overview	22-26

23 Using Oracle Advanced Analytics: Oracle Data Mining and Oracle R Enterprise

Course Road Map	23-2
Objectives	23-3
Oracle Advanced Analytics (OAA)	23-4
OAA: Oracle Data Mining	23-5
What Is Data Mining?	23-6
Common Uses of Data Mining	23-7
Defining Key Data Mining Properties	23-8
Data Mining Categories	23-10
Supervised Data Mining Techniques	23-11
Supervised Data Mining Algorithms	23-12
Unsupervised Data Mining Techniques	23-13
Unsupervised Data Mining Algorithms	23-14
Oracle Data Mining: Overview	23-15
Oracle Data Miner GUI	23-16
ODM SQL Interface	23-17
Oracle Data Miner 4.1 Big Data Enhancement	23-18
Example Workflow Using JSON Query Node	23-19
ODM Resources	23-20
Practice: Overview (ODM)	23-21
OAA: Oracle R Enterprise	23-22
What Is R?	23-23
Who Uses R?	23-24
Why Do Statisticians, Data Analysts, Data Scientists Use R?	23-25
Limitations of R	23-26
Oracle's Strategy for the R Community	23-27
Oracle R Enterprise	23-28
ORE: Software Features	23-29
ORE Packages	23-30
Functions for Interacting with Oracle Database	23-31

ORE: Target Environment	23-32
ORE: Data Sources	23-33
ORE and Hadoop	23-34
ORAAH: Architecture	23-35
ORAAH Package	23-36
HDFS Connectivity and Interaction	23-37
ORAAH Functions for HDFS Interaction	23-38
ORAAH Functions for Predictive Algorithms	23-39
Hadoop Connectivity and Interaction	23-40
Word Count: Example Without ORAAH	23-41
Word Count: Example with ORAAH	23-42
ORE Resources	23-43
Practice: Overview (ORE)	23-44
Summary	23-45

24 Introducing Oracle Big Data Discovery

Course Road Map	24-2
Objectives	24-3
Oracle Big Data Discovery	24-4
Find Data	24-5
Explore Data	24-6
Transform and Enrich Data	24-7
Discover Information	24-8
Share Insights	24-9
BDD: Technical Innovation on Hadoop	24-10
Additional Resources	24-11
Summary	24-12

25 Introduction to the Oracle Big Data Appliance (BDA)

Course Road Map	25-2
Objectives	25-3
Oracle Big Data Appliance	25-4
Oracle Big Data Appliance: Key Component of the Big Data Management System	25-5
Oracle-Engineered Systems for Big Data	25-6
The Available Oracle BDA Configurations	25-7
Using the Mammoth Utility	25-8
Using Oracle BDA Configuration Generation Utility	25-10
Configuring Oracle Big Data Appliance	25-11
The Generated Configuration Files	25-13
The Oracle BDA Configuration Generation Utility Pages	25-15

Using Oracle BDA Configuration Generation Utility: The Customer Details Page	25-16
Using Oracle BDA Configuration Generation Utility: The Hardware Selections Page	25-17
The Oracle BDA Configuration Generation Utility: The Define Clusters Page	25-18
The Oracle BDA Configuration Generation Utility: The Cluster n Page	25-19
BDA Configurations: Full Rack	25-20
BDA Configurations: Starter Rack	25-21
BDA Configurations: In-Rack Expansion	25-22
BDA Starter Rack: Hadoop Cluster Only	25-23
BDA Starter Rack: NoSQL Cluster Only	25-24
Big Data Appliance: Horizontal Scale-Out Model	25-25
Big Data Appliance: Software Components	25-26
Oracle Big Data Appliance and YARN	25-27
Stopping the YARN Service	25-28
Critical and Noncritical Nodes in an Oracle BDA CDH Cluster	25-29
First NameNode and Second NameNode	25-30
First ResourceManager and Second ResourceManager	25-31
Hardware Failure in Oracle NoSQL	25-32
Oracle Integrated Lights Out Manager (ILOM): Overview	25-33
Oracle ILOM Users	25-34
Connecting to Oracle ILOM Using the Network	25-35
Oracle ILOM: Integrated View	25-36
Monitoring the Health of Oracle BDA: Management Utilities	25-37
Big Data Appliance: Security Implementation	25-38
Big Data Appliance: Usage Guidelines	25-39
Summary	25-40
Practice 25: Overview	25-41

26 Managing Oracle BDA

Course Road Map	26-2
Objectives	26-3
Lesson Agenda	26-4
Mammoth Utility	26-5
Installation types	26-6
Mammoth Code: Examples	26-7
Mammoth Installation Steps	26-8
Lesson Agenda	26-10
Monitoring Oracle BDA	26-11
Oracle BDA Command-Line Interface	26-12
bdacli	26-13

setup-root-ssh 26-14
Lesson Agenda 26-15
Monitor BDA with Oracle Enterprise Manager 26-16
OEM: Web and Command-Line Interfaces 26-17
OEM: Hardware Monitoring 26-18
Hadoop Cluster Monitoring 26-19
Lesson Agenda 26-20
Managing CDH Operations 26-21
Using Cloudera Manager 26-22
Monitoring Oracle BDA Status 26-23
Performing Administrative Tasks 26-24
Managing Services 26-25
Lesson Agenda 26-26
Monitoring MapReduce Jobs 26-27
Monitoring the Health of HDFS 26-28
Lesson Agenda 26-29
Cloudera Hue 26-30
Hive Query Editor (Hue) Interface 26-31
Logging in to Hue 26-32
Lesson Agenda 26-33
Starting Oracle BDA 26-34
Stopping Oracle BDA 26-35
BDA Port Assignments 26-36
Summary 26-37
Practice 26: Overview 26-38

27 Balancing MapReduce Jobs

Course Road Map 27-2
Objectives 27-3
Ideal World: Neatly Balanced MapReduce Jobs 27-4
Real World: Skewed Data and Unbalanced Jobs 27-5
Data Skew 27-6
Data Skew Can Slow Down the Entire Hadoop Job 27-7
Perfect Balance 27-8
How Does the Perfect Balance Work? 27-9
Using Perfect Balance 27-10
Application Requirements for Using Perfect Balance 27-11
Perfect Balance: Benefits 27-12
Using Job Analyzer 27-13
Getting Started with Perfect Balance 27-14
Using Job Analyzer 27-16

Environmental Setup for Perfect Balance and Job Analyzer	27-17
Running Job Analyzer as a Stand-alone Utility to Measure Data Skew in Unbalanced Jobs	27-18
Using Job Analyzer as a Stand-Alone Utility: Example with a YARN Cluster	27-19
Configuring Perfect Balance	27-20
Using Perfect Balance to Run a Balanced MapReduce Job	27-21
Running a Job Using Perfect Balance: Examples	27-23
Perfect Balance–Generated Reports	27-25
The Job Analyzer Reports: Structure of the Job Output Directory	27-26
Reading the Job Analyzer Reports	27-27
Reading the Job Analyzer Report in HDFS Using a Web Browser	27-28
Reading the Job Analyzer Report in the Local File System in a Web Browser	27-29
Looking for Skew Indicators in the Job Analyzer Reports	27-30
Job Analyzer Sample Reports	27-31
Collecting Additional Metrics with Job Analyzer	27-32
Using Data from Additional Metrics	27-33
Using Perfect Balance API	27-34
Chopping	27-35
Disabling Chopping	27-36
Troubleshooting Jobs Running with Perfect Balance	27-37
Perfect Balance Examples Available with Installation	27-38
Summary	27-40
Practice 27: Overview	27-41

28 Securing Your Data

Course Road Map	28-2
Objectives	28-3
Security Trends	28-4
Security Levels	28-5
Outline	28-6
Relaxed Security	28-7
Authentication with Relaxed Security	28-8
Authorization	28-9
HDFS ACLs	28-10
Changing Access Privileges	28-11
Relaxed Security Summary	28-12
Challenges with Relaxed Security	28-13
BDA Secure Installation	28-14
Kerberos Key Definitions	28-15
Strong Authentication with Kerberos	28-16
Snapshot of Principals in KDC	28-17

Authentication with Kerberos	28-18
User Authentication: Examples	28-19
Service Authentication and Keytabs	28-20
Review TGT Cache	28-21
Ticket Renewal	28-22
Adding a New User	28-23
Example: Adding a New User	28-24
Example: Adding a User to Hue	28-25
Authorization	28-26
Sentry Authorization Features	28-27
Sentry Configuration	28-28
Users, Groups, and Roles	28-29
Sentry Example: Overview	28-30
Example: Users, Roles, and Groups	28-31
1. Creating Roles	28-32
2. Assigning Roles to Groups	28-33
Show Roles for a Group	28-34
Create Databases (in Hive)	28-35
Privileges on Source Data for Tables	28-36
Granting Privileges on Source Data for Tables	28-38
Creating the Table and Loading the Data	28-39
Attempting to Query the Table Without Privileges	28-40
Grant and Revoke Access to Table	28-41
Sentry Key Configuration Tasks	28-42
Oracle Database Access to HDFS	28-43
Oracle Connection to Hadoop	28-44
Virtual Private Database Policies Restrict Data Access	28-45
Oracle Data Redaction Protects Sensitive Data	28-46
Auditing: Overview	28-47
Auditing	28-48
Oracle Audit Vault and Database Firewall	28-49
Oracle Audit Vault Reporting	28-51
User-Defined Alerts	28-52
Example: Authorized Data Access	28-53
Example: Unauthorized Data Access	28-54
Audit Vault Dashboard	28-55
Interactive Reporting	28-56
Cloudera Navigator	28-57
Cloudera Navigator Reporting	28-58
Cloudera Navigator Lineage Analysis	28-59
Encryption	28-60

Network Encryption 28-61
Data at Rest Encryption 28-62
Summary 28-63

A Glossary

B Resources

1

Introduction

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Discuss the course objectives and road map
- List the appendixes
- Identify the practice environment
- Identify the relevant documentation and other resources



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

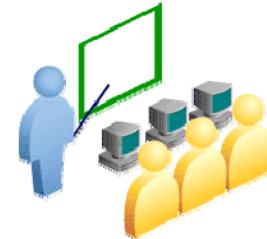
This introductory lesson discusses the course goals and the daily agenda of topics.

The information on how to connect to the practice environment is described followed by information on how to access documentation and Oracle Learning Library (OLL).

Questions About You

To ensure that the class can be customized to meet your specific needs and to encourage interaction, answer the following questions:

- Which organization do you work for?
- What is your role in your organization?
- If you are a DBA, what products have you worked on?
- Have you used Oracle Big Data Appliance (BDA)?
- Have you used any Hadoop components?
- Do you meet the course prerequisites?
- What do you hope to learn from this course?



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can benefit most from this class if you know the background of your classmates and the issues that they face in the development and management of a data warehouse. Each student has unique perspectives, experience, and knowledge from which we all can learn.

Course Objectives

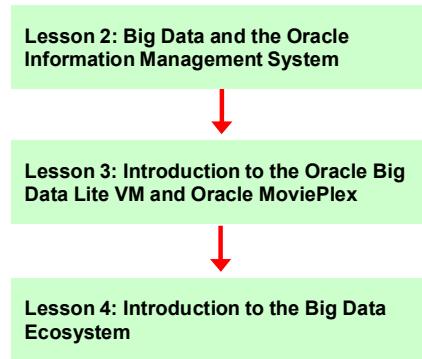
After completing this course, you should be able to:

- Define the term *big data* and Discuss Oracle's Big Data solution
- Review the Hadoop and NoSQL ecosystems
- Acquire data into HDFS and Oracle NoSQL Database using Flume and Sqoop
- Examine and run MapReduce jobs
- Use Hive and Pig
- Use Oracle Big Data Connectors
- Analyze data by using Hadoop, Oracle SQL Analytics, Oracle Advanced Analytics, and Oracle Big Data Discovery
- Introduce the Oracle Big Data Appliance (BDA)
- Provide data security and enable resource management



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map: Module 1 Big Data Management System



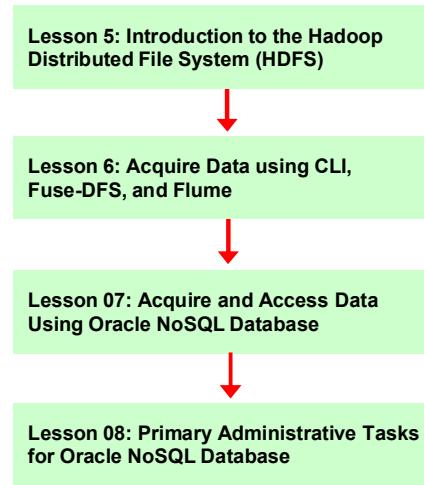
ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The course comprises five independent modules divided into 28 lessons.

- **Module 1: Big Data Management System**
 - Lessons 2–4
- **Module 2: Data Acquisition and Storage**
 - Lessons 5–8
- **Module 3: Data Access and Processing**
 - Lessons 9–15
- **Module 4: Data Unification and Analysis**
 - Lessons 16–24
- **Module 5: Using and Managing Oracle Big Data Appliance**
 - Lessons 25–28

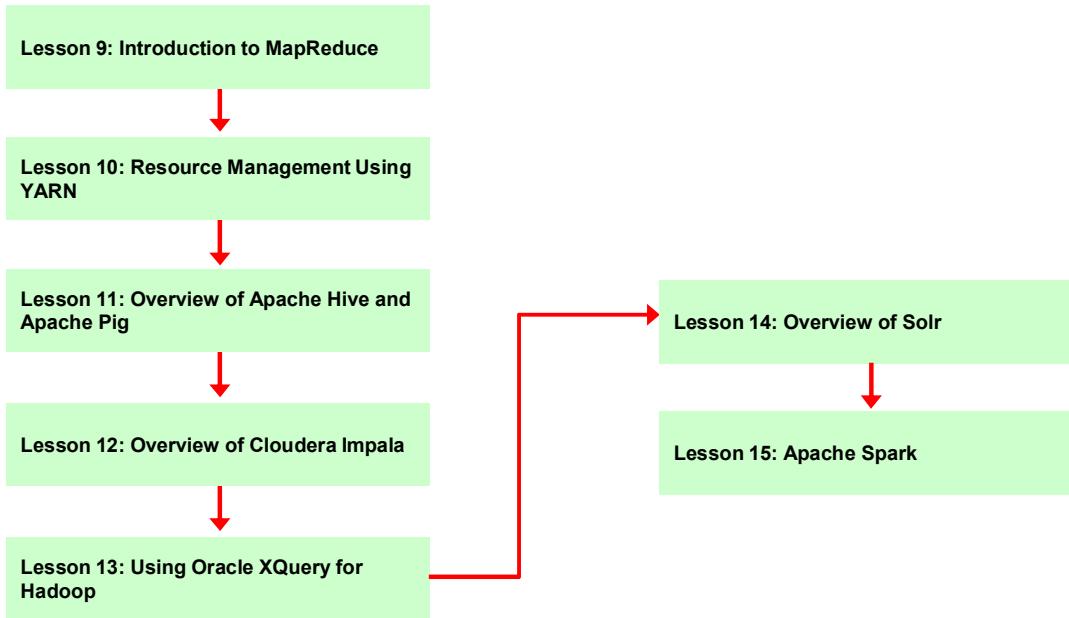
Course Road Map: Module 2 Data Acquisition and Storage



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

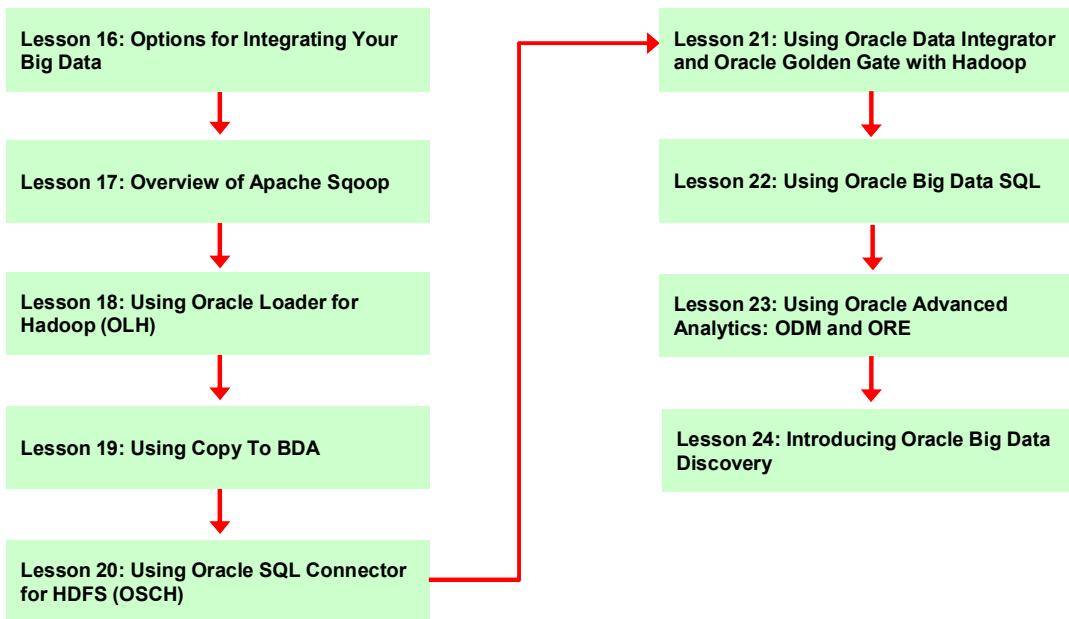
Course Road Map: Module 3 Data Access and Processing



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map: Module 4 Data Unification and Analysis

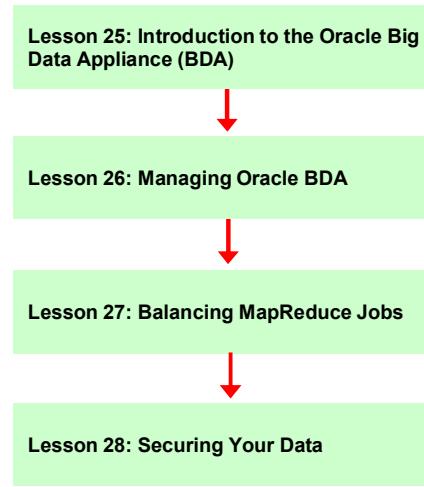


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map: Module 5

Using and Managing Oracle Big Data Appliance



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Oracle Big Data Lite Virtual Machine (Used in this Course) Home Page

Oracle Technology Network > Database > **Big Data Appliance**

Database 12c
Database In-Memory
Multitenant
Options
Application Development
Big Data Appliance
Data Warehousing & Big Data
Database Appliance
Database Cloud
Exadata Database Machine
High Availability
Manageability
Migrations
Security
Unstructured Data
Upgrades
Windows
Database Technology Index

Version 4.0.1

Please note: This appliance is for testing and educational purposes only; it is unsupported and not to be used in production. It includes software products that are optional on the Oracle Big Data Appliance (BDA), including Oracle NoSQL Database Enterprise Edition and Oracle Big Data Connectors.

- [Introduction](#)
- [Download Oracle Big Data Lite Virtual Machine](#)
- [Getting Started](#)
 - [Oracle MoviePlex](#)
 - [Hands-on Labs](#)
 - [Web Sites /White Papers / EBook / Blogs](#)

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle Big Data Lite Virtual Machine provides an integrated environment to help you get started with the Oracle Big Data platform. Many Oracle Big Data platform components have been installed and configured, allowing you to begin using the system right away.

You will be using the Oracle Big Data Lite Virtual Machine in many of the practices.

Connecting to the Practice Environment

To connect to the practice environment, you need:

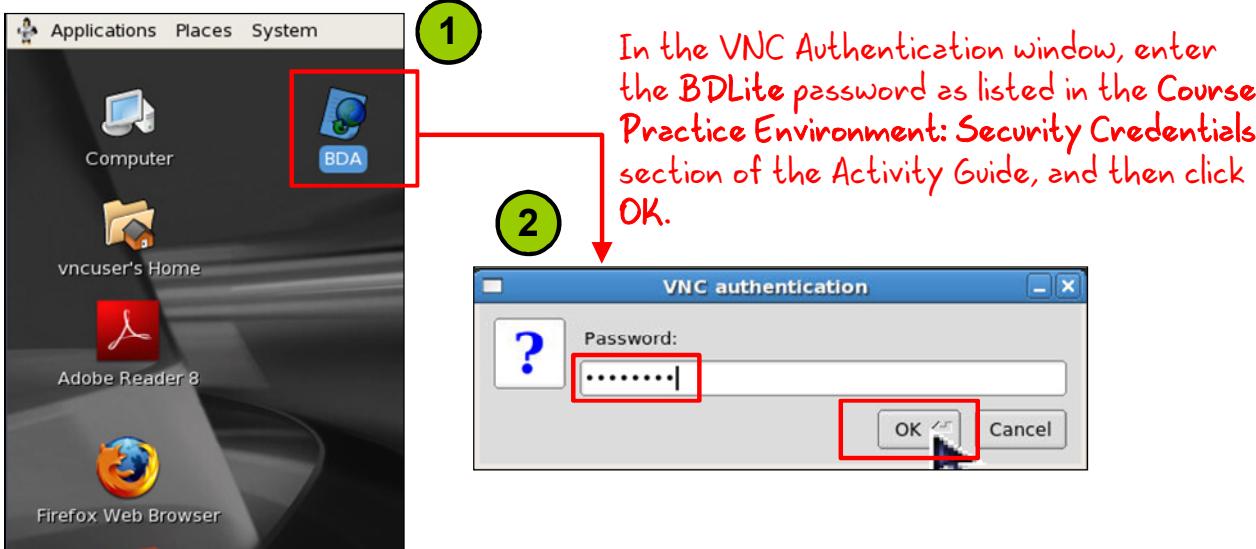
- Your assigned machine (host) name (or IP address) and your machine username and password:
 - This information is provided to you by your instructor
- The username and password to the BDLite VM and the tools that are used in the BDLite VM:
 - You can find this information in the **Course Practice Environment: Security Credentials** section at the start of the Activity Guide



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Starting the Oracle Big Data Lite Virtual Machine (VM) Used in this Course

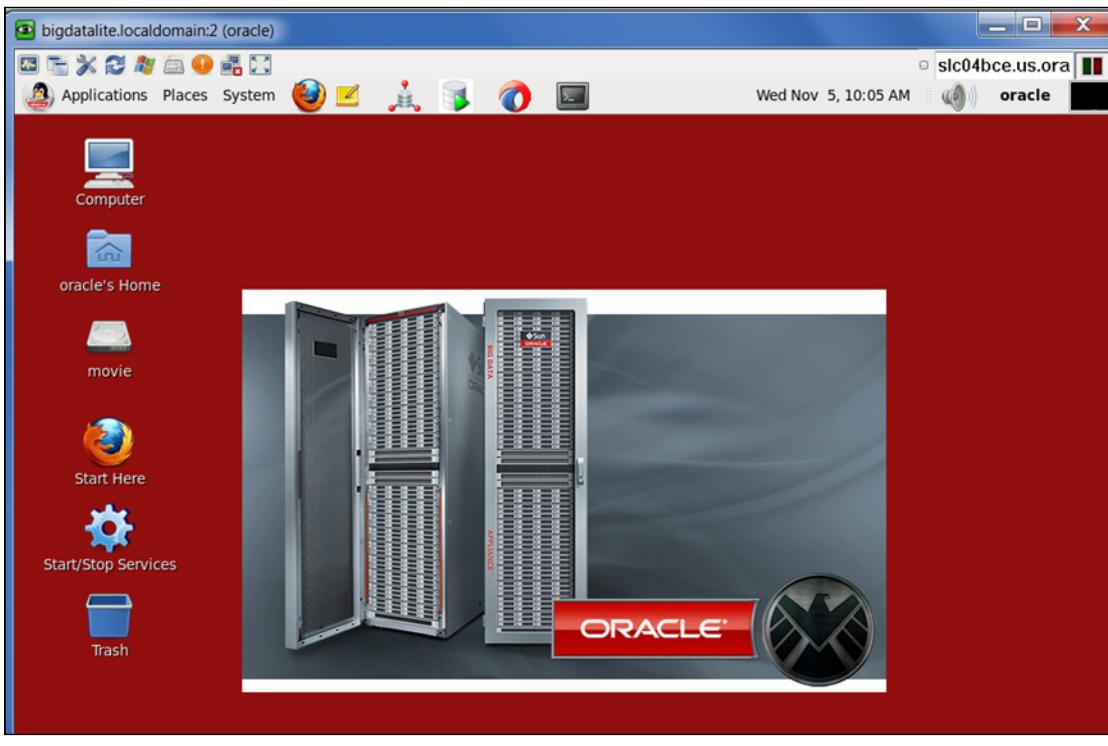
On your assigned host machine desktop, click the BDA icon.



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Starting the Oracle Big Data Lite (BDLite) Virtual Machine (VM) Used in this Course



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Accessing the Getting Started Page from the Oracle BD Lite VM

The screenshot shows a Firefox browser window with the title "Oracle Big Data Lite - Getting Started - Mozilla Firefox". The address bar shows "file:///home/oracle/GettingStarted/StartHere.html". The main content area displays the "Oracle Big Data Lite 4.0.1 Virtual Machine Getting Started" page. Below the heading, there is a note: "Please note: This appliance is for testing and educational purposes only." Under the heading "Introduction", it says: "Oracle Big Data Lite Virtual Machine provides an integrated environment for Oracle Big Data platform components have been installed and configured. The following components are included on Oracle Big Data Lite:" followed by a bulleted list. To the right of the browser window is a table titled "Access Information" which lists various Oracle components and their access details.

Component	Details
Linux	root/welcome1 oracle/welcome1
Oracle Database 12c	SID: orcl Port: 1521 All passwords: welcome1 moviedemo is the owner of the movie schema
Oracle Data Integrator	ODI is available using the tool bar menu at the top of the screen. ODI user name: SUPERVISOR ODI password: welcome1
Oracle NoSQL Database	Administration Page: http://localhost:5001
Hive Metastore (MySQL)	User name: hive Password: welcome1
Hue	http://localhost:8888 User name: oracle Password: welcome1
...	
Oracle MoviePlex Demo	Start the demo: /home/oracle/movie/moviedemo/scripts/1_start_movieapp.sh http://localhost:7001/movieplex/index.jsp User name: guest1 Password: welcome1

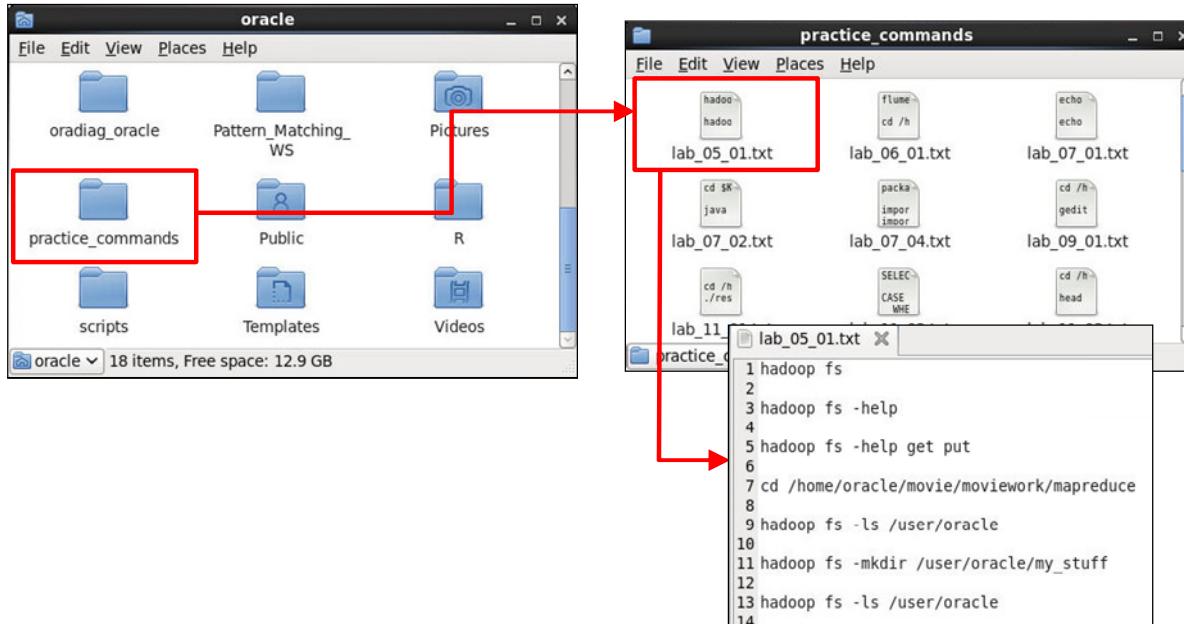
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Note: The **Access Information** screen capture in the slide, shows some and not all of the Oracle Big Data Lite components' details due to page limitation.

Accessing the Practice Files

You can find the list of commands for the practices in the /home/oracle/practice_commands directory.

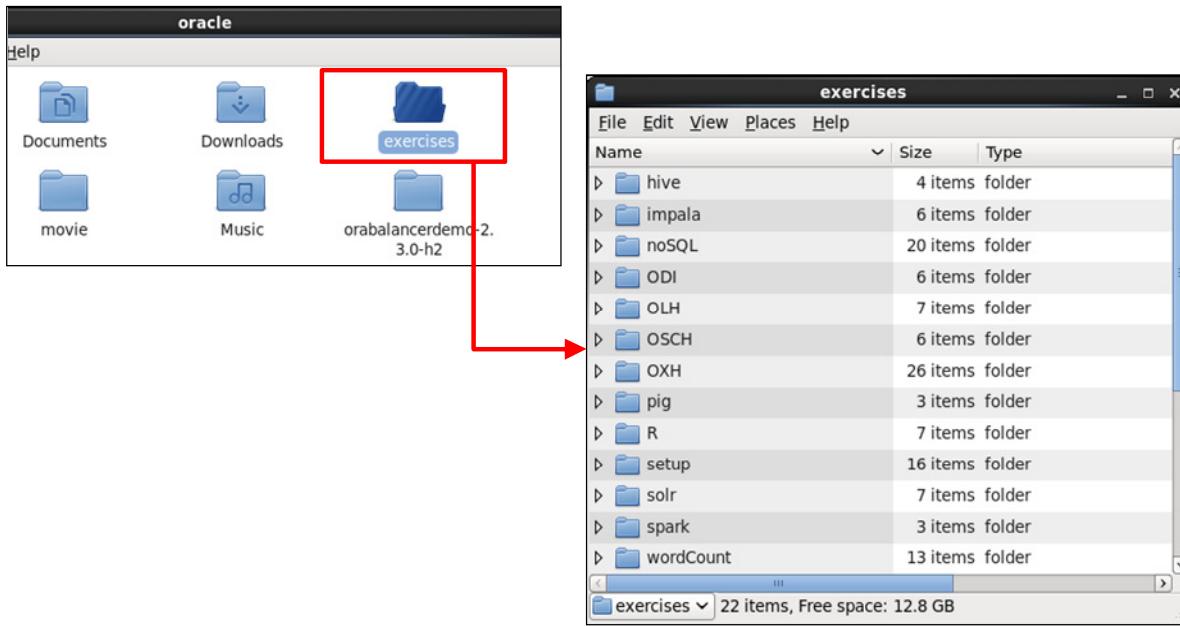


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Accessing the /home/oracle/exercises Directory

The /home/oracle/exercises directory contains the sub-directories needed for some of the practices.

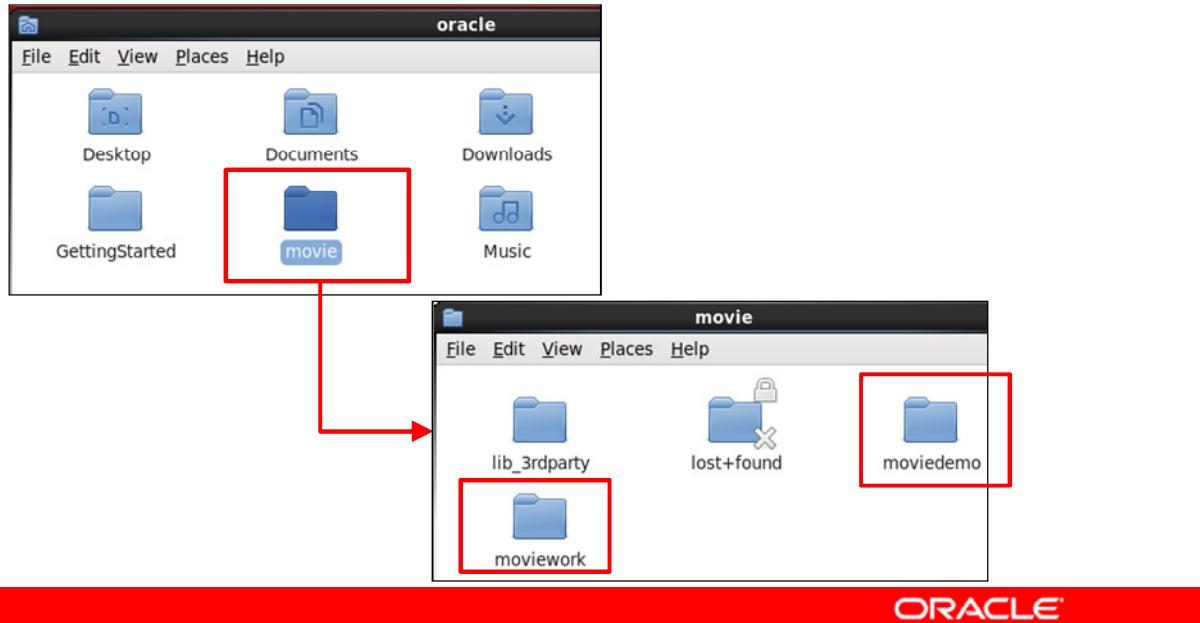


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Accessing /home/oracle/movie Directory

The /home/oracle/movie directory contains the moviedemo and movework sub-directories needed for some of the practices.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Appendices

Appendix A: Glossary

Term	Description
Apache Flume	A distributed service for collecting and aggregating data from almost any source.
Apache Hadoop	A batch processing infrastructure that stores files and distributes work across a cluster of commodity servers. It uses Cloudera's Distribution including Apache Hadoop (CDH).
Apache HBase	An open-source, column-oriented database that provides random, read/write access to a distributed, decentralized store. It provides fast lookup of values by key and can perform thousand of operations per second.

Appendix B: Resources

Web Site	URL
Apache Flume	http://flume.apache.org
Apache Hadoop	http://hadoop.apache.org
Apache Sentry	http://hadoop.apache.org/docs/r2.3.0/hadoop-yarn/hadoop-sentry/
Apache Hbase	http://hbase.apache.org
Apache Hive	https://hive.apache.org
Apache Mahout	http://mahout.apache.org
Apache Oozie	http://oozie.apache.org
Apache Pig	http://pig.apache.org
Apache Solr	http://lucene.apache.org/solr/
Apache Spark	https://spark.apache.org
Apache Spark	https://spark.apache.org
Apache Sqoop	http://sqoop.apache.org

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle Big Data Appliance Documentation

<http://www.oracle.com/technetwork/server-storage/engineered-systems/bigdata-appliance/documentation/index.html>

Oracle Technology Network > Database > Big Data Appliance > Documentation

Database 12c
Database In-Memory
Multitenant
Options
Application Development
Big Data Appliance
Data Warehousing & Big Data
Database Appliance
Database Cloud
Exadata Database Machine
High Availability
Manageability
Migrations
Security
Unstructured Data
Upgrades

Overview Downloads Documentation Community Learn More

Oracle Big Data Documentation

Oracle Big Data offers an integrated portfolio of products to help you organize and analyze your diverse data sources alongside your existing data to find new insights and capitalize on hidden relationships.

Oracle Big Data, Release 4.1.0
E57371_01.zip

Oracle Big Data, Release 4.0.0
E55905_01.zip

Oracle Big Data, Release 3.1.0
E55659_01.zip

Oracle Big Data, Release 3.0.0
E53356_01.zip

Oracle Big Data, Release 2.6.0
E54130_01.zip

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle Big Data Appliance Documentation

Oracle Big Data Documentation Release 4.1

Overview

Welcome

Companies have been making business decisions for decades based on transactional data stored in relational databases. Beyond that critical data is a potential treasure trove of less structured data: weblogs, social media, email, sensors, and photographs that can be mined for useful information.

Oracle offers a broad and integrated portfolio of products to help you acquire and organize these diverse data sources and analyze them alongside your existing data to find new insights and capitalize on hidden relationships.

[Learn how Oracle helps you acquire, organize, and analyze](#)

Related Software

[Oracle R Enterprise](#)
[Oracle NoSQL Database](#)
[Oracle Enterprise Manager Control](#)
[Oracle Audit Vault and Firewall](#)

Related Hardware

[Oracle Server X5-2](#)

Big Data Appliance Installation and Setup

[Safety and Compliance Guide](#) ⓘ ↗
[Site Checklists](#) ⓘ ↗
[Owner's Guide](#) ⓘ ↗

Integrated Software and Big Data Connectors

[Licensing Information](#) ⓘ ↗
[Software User's Guide](#) ⓘ ↗
[Enterprise Manager System Monitoring](#) ⓘ ↗
[Plug-in Installation Guide for Oracle Big Data Appliance](#)
[Connectors User's Guide](#) ⓘ ↗
[Oracle Loader for Hadoop Java Example](#) ⓘ ↗
[Perfect Balance Java API Reference](#) ⓘ ↗
[Cloudera's Distribution Including Apache Hadoop Library](#) ⓘ ↗

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Additional Resources: Oracle Big Data Tutorials on Oracle Learning Library (OLL)

- You can access tutorials, OBEs, and other material by using the OLL at <http://www.oracle.com/goto/oll>.
- Provide step-by-step instructions for performing a variety of tasks in Big Data and related products.
- Reduce the time that you spend investigating the required steps to perform a task.
- Use practical real-world situations so that you can gain knowledge through valuable hands-on experience. You can then use the solutions as the foundation for production implementation, dramatically reducing time to deployment.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Additional Resources: Oracle Big Data Tutorials on Oracle Learning Library (OLL)

https://apex.oracle.com/pls/apex/f?p=44785:141:0:::P141_PAGE_ID,P141_SECTION_ID:27,617

The screenshot shows the Oracle Learning Library interface. At the top, there's a navigation bar with links for Home, Products, Search, and My Library. Below the navigation is a banner for the Oracle Big Data Learning Library, which includes a search bar and tabs for Welcome, Get Started, Learn by Role (selected), Learn by Product, Learning Paths, Latest Additions, and Additional Resources. A sidebar on the left lists courses for 'Developer / Data Scientist' such as 'Oracle Big Data Overview - 1 day training on demand', 'Oracle Big Data Essentials - 3 day class', and 'Using Oracle NoSQL Database - 4 day class'. To the right, there's a 'Big Data and Innovation' section featuring a video thumbnail of two men speaking.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 1-1: Overview

This practice covers using your machine Web browser (Not the BDLite Web browser) to access and review some of the Big Data resources and documentation.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Discuss the course objectives and road map
- List the appendixes
- Identify the practice environment
- Identify the relevant documentation and other resources



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

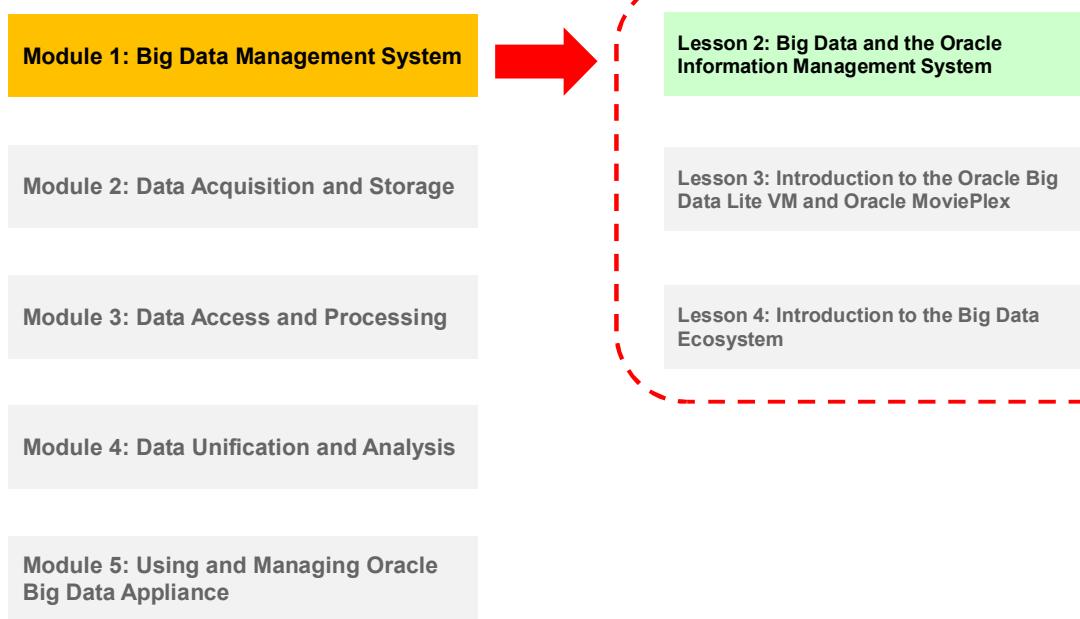
Big Data and the Oracle Information Management System



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This is the Course Road Map slide. You will see an updated version of this slide at the beginning of each lesson throughout the course.

Given the wide variety of Big Data technologies and products that will be covered in this course, the purpose of this slide is to provide context for where you are, where you have been, and where you are going. In addition, as each technology or product is introduced, you are reminded of the particular phase to which the technology belongs.

In the first module, you learn about Oracle's approach and strategy to leverage and manage Big Data.

In this lesson, you learn about the architectural principles associated with Oracle's Information Management System for Big Data.

Lesson Objectives

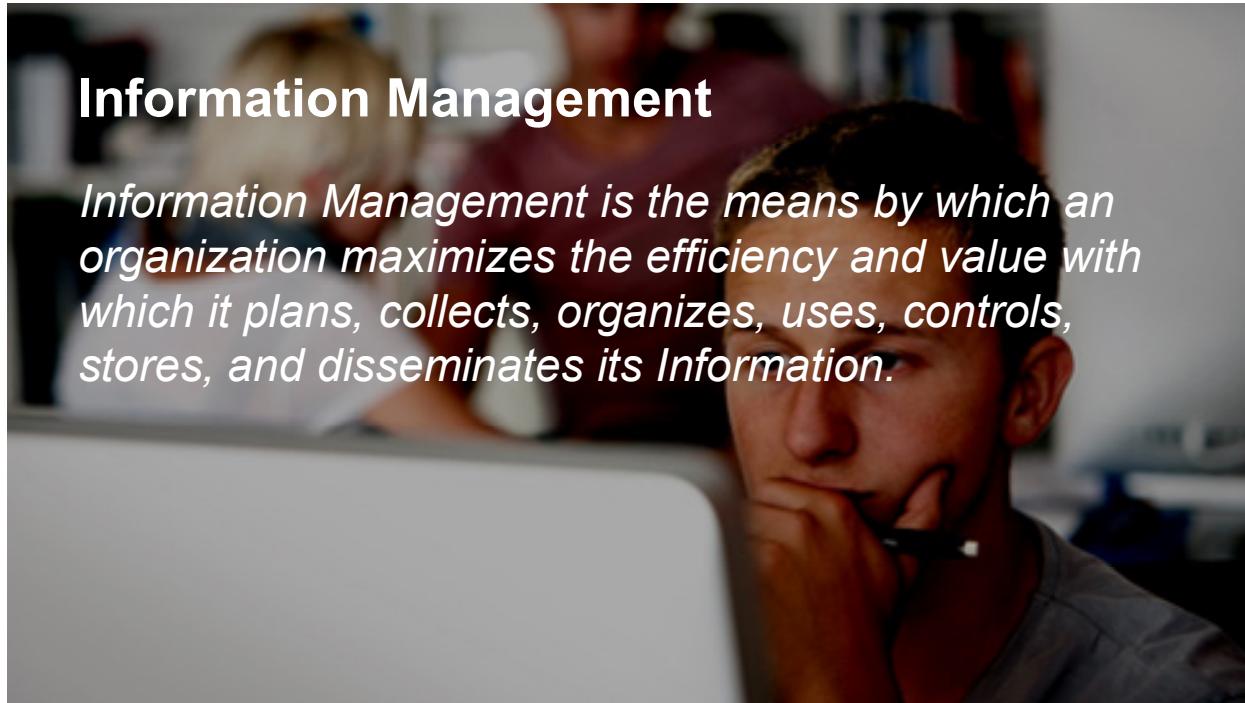
After completing this lesson, you should be able to:

- Define the term *Big Data*
- Identify the challenges and opportunities in implementing Big Data
- Describe the Oracle Information Management Architecture for Big Data
- Describe Oracle's technology approach for Big Data adoption



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Big Data: A Strategic IM Perspective



Information Management

Information Management is the means by which an organization maximizes the efficiency and value with which it plans, collects, organizes, uses, controls, stores, and disseminates its Information.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The last few years have shown a significant rise in the number of organizations adopting Big Data technologies as a means by which they can economically manage more of their data and analyze it to support more *fact based* decision making.

Given the strategic business imperative and increasing technology capability, it is important to understand how these technologies relate to an organization's existing information management system, and how best to combine them (if at all) into a coherent platform that will not only support a management by fact approach, but also facilitate the discovery of new facts and ideas and set these into the business process.

In this course, we explore Big Data within the context of Oracle's Information Management (IM) Reference Architecture. In this lesson, we discuss some of the Big Data background and review how the Reference Architecture can help to integrate structured, semi-structured, and unstructured information into a single logical information resource that can be exploited for commercial gain.

Big Data

Big Data is a term often used to describe data sets whose size is beyond the capability of commonly used software tools to capture, manage, and process.

Big Data can be generated from many different sources, including:

- Social networks
- Banking and financial services
- E-commerce services
- Web-centric services
- Internet search indexes
- Scientific and document searches
- Medical records
- Web logs



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

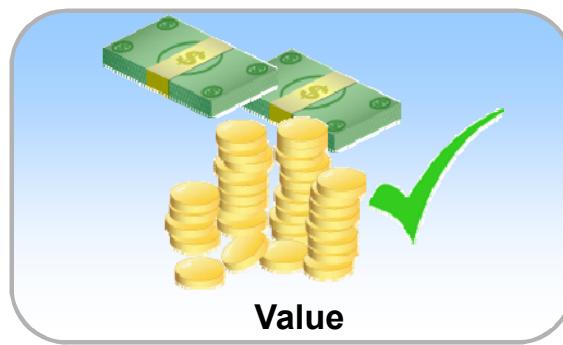
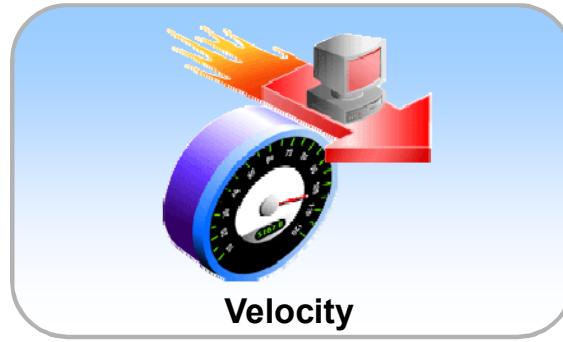
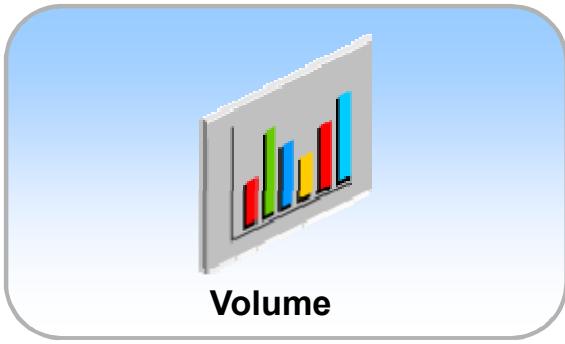
Big Data is a term often applied by people to describe data sets whose size is beyond the capability of commonly used software tools to capture, manage, and process. The sheer size of the data, combined with complexity of analysis and commercial imperative to create value from it, has led to a new class of technologies and tools to exploit it.

The term Big Data tends to be used in multiple ways, often referring to both:

- The type of data being managed
- The technology used to store and process it

Mostly, these technologies originated from companies such as Google, Amazon, Facebook and Linked-In, where they were developed for each company's own use to analyze the massive amounts of social media data being produced. Due to the nature of these companies, the emphasis was on low cost scale-out commodity hardware and open source software.

Characteristics of Big Data



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

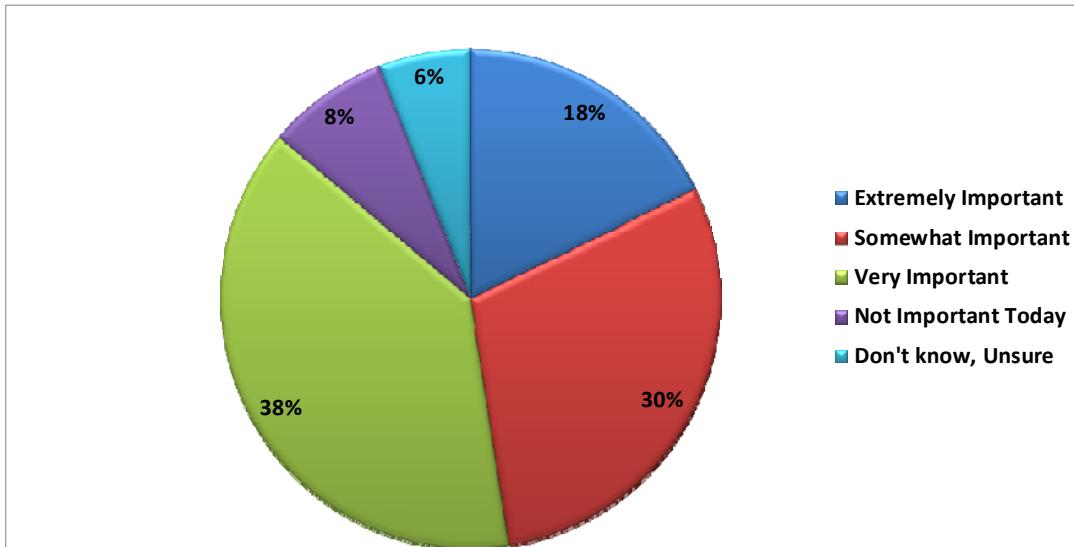
The world of Big Data is increasingly being defined by the 4 Vs. These “Vs” become a reasonable test as to whether a Big Data approach is the right one to adopt for a new area of analysis.

The 4 Vs include: *volume*, *velocity*, *variety*, and *value*.

- Volume. The size of the data. With technology it is often very limiting to talk about data volume in any absolute sense. As technology marches forward, numbers get quickly outdated so it is better to think about volume in a relative sense. If the volume of data you are looking at is an order of magnitude or larger than anything previously encountered in your industry, you are probably dealing with Big Data. For some companies this might be 10s of terabytes, for others it may be 100s of petabytes.
- Velocity. The rate at which data is being received and has to be acted upon is becoming much more real time. While it is unlikely that any real analysis will need to be completed in the same time period, delays in execution will inevitably limit the effectiveness of campaigns, limit interventions or lead to suboptimal processes. For example, some kind of discount offer to a customer based on their location is less likely to be successful if they have already walked some distance past the store.

- Variety. There are two aspects of variety to consider: syntax and semantics. In the past these have determined the extent to which data could be reliably structured into a relational database and content exposed for analysis. Although modern ETL tools are very capable of dealing with data arriving in virtually any syntax, in the past they were less able to deal with semantically rich data such as free text. As a result, many organizations restricted the data coverage of IM systems to a narrow range of data. Deferring the point at which this kind of rich data, which is often not fully understood by the business, also has significant appeal and avoids costly and frustrating modeling mistakes. It follows then that by being more inclusive and allowing greater model flexibility additional value may be created—this is perhaps one of the major appeals of the Big Data approach.
- Value. The commercial value of any new data sources must also be considered. Or, perhaps more appropriately, we must consider the extent to which the commercial value of the data can be predicted ahead of time so that ROI can be calculated and project budget acquired. “Value” offers a particular challenge to IT in the current harsh economic climate. It is difficult to attract funds without certainty of the ROI and payback period. The tractability of the problem is closely related to this issue as problems that are inherently more difficult to solve will carry greater risk, making project funding more uncertain. Big Data technologies can have a significant impact on the overall picture of “information ROI” as well as more specific project viability by minimizing up-front investment before developing a more complete understanding of value through the discovery process. See Extending the Boundaries of Information Management for more details of this discovery process.

Importance of Big Data



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

An Oracle-commissioned study produced the following result: 86% of interviewed executives considered Big Data to be an important part of their business. This study is backed up by a significant rise in the number of organizations adopting Big Data technologies in the last few years.

Given the strategic business imperative of Big Data, and increasing technology capability supporting Big Data, it is important for an organization to understand:

- How Big Data technologies relate to their existing Information Management environment
- How best to combine them (if at all) into a coherent platform that will support a management by fact approach
- How to facilitate the discovery of new facts and ideas and set these into the business process

As mentioned previously, we'll explore these questions within the context of Oracle's Information Management Reference Architecture.

Big Data Opportunities: Some Examples

Today's Challenge	New Data	What's Possible?
Healthcare: Expensive office visits	Remote patient monitoring	Preventive care, reduced hospitalization, epidemiological studies
Manufacturing: In-person support	Product sensors	Automated and predictive diagnosis and support
Location-based services: Based on home ZIP code	Real-time location data	Geo-advertising, personalized notifications and search
Utilities: Complex distribution grid	Detailed consumption statistics	Increased availability, reduced cost, tiered metering plans



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Everyone has a business use case like the following:

"If only I could capture and analyze the appropriate data, I could solve problem X and increase the value of our services and/or offerings."

The slide provides a few typical examples of Big Data challenges and potential opportunities.

Big Data Challenges



Schematize? Analyze? Processing? Governance?

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

There are a variety of challenges organizations face when embarking on a Big Data project. From an engineering standpoint, four questions are common.

When to Schematize?

To make data understandable a schema must be applied to it before analysis. One aspect that most clearly distinguishes Big Data from the relational approach is the point at which data is organized into a schema.

- In the relational approach, data is placed into a schema when initially written to the database (*schema-on-write*).
- In a Big Data approach, data is only organized into a schema immediately before analysis as it is read (*schema-on-read*).

How to Analyze?

One area where there is almost no difference between the approaches is in the analysis technologies applied. Data Scientists typically use a broad range of technologies such as SQL, Data Mining, and statistical and graphical analysis depending on the problem being tackled.

How to Process?

Another challenging area is the processing of unstructured and semi-structured data. This is mostly because the terms fuse two key differences of the *physical representation* of the data (syntax) and its *inherent meaning* (semantics). For example, a file containing JSON or XML data is easily processed by relational and Big Data technologies, but if the meaning of the data is not fully understood or could change over time, having some schema flexibility will be important.

What About Governance?

Perhaps the most critical difference between relational and Big Data technologies is really more to do with a philosophical positioning than anything technical. The technologies come from different places and the people involved have a different world view.

- Because of the schema freedom offered by Big Data technologies we are more able to accept and process really messy data and evolve the schema as we go. This makes it ideal for data discovery and leads to a greater level of agility.
- But this agility comes at a cost. You can imagine what might happen in your business if every time you analyzed total sales for the year you received a different result! It would not take long before trust in the data had broken down and the IM solution abandoned.

Information Management Landscape

Information Management (IM) is the means by which an organization seeks to:

- Maximize the efficiency with which it plans, collects, organizes, uses, controls, stores, disseminates, and disposes of its information
- Ensure that the value of that information is identified and exploited to the maximum extent possible



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Before examining Oracle's strategy for Big Data, let us review some basic Information Management (IM) terms and discuss the new demands that are increasingly felt by organizations across all industry sectors as they seek to exploit new categories and types of data for commercial advantage.

There are many definitions of IM. For the purposes of this lesson we use a broad definition that highlights the full life cycle of the data, as shown in the slide. This definition has a focus on the creation of value from the data, and somewhat inevitably includes aspects of people, process, and technology within it.

Although existing IM solutions have focused efforts on the data that is readily structured and thereby easily analyzed by using standard tools, our definition is deliberately more inclusive. Historically, the scope of data was typically defined by technical and cost limitations, as the cost and complexities of dealing with other forms of data often outweighed additional benefits.

With the advent of new technologies such as Hadoop and NoSQL as well as advances in technologies such as Oracle Exadata, Oracle Big Data Appliance, and the In-Memory Database Option, many of these limitations have been removed, expanding the barriers to include a wider range of data types, volumes, and possibilities.

Modern hardware and software technologies are changing what it's possible to deliver from an IM perspective. Therefore, the overall architecture and organizing principles are becoming more critical to organizing data effectively.

Extending the Boundaries of Information Management



- How can Big Data technologies be applied to create additional business value or reduce the costs of delivering Information Management?
- Bridge Big Data and traditional relational database worlds by integrating structured, semi-structured, and unstructured information.
- Augment Big Data analysis techniques with a portfolio of Oracle Advance Analytics, Business Intelligence, and Data Warehousing technologies.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

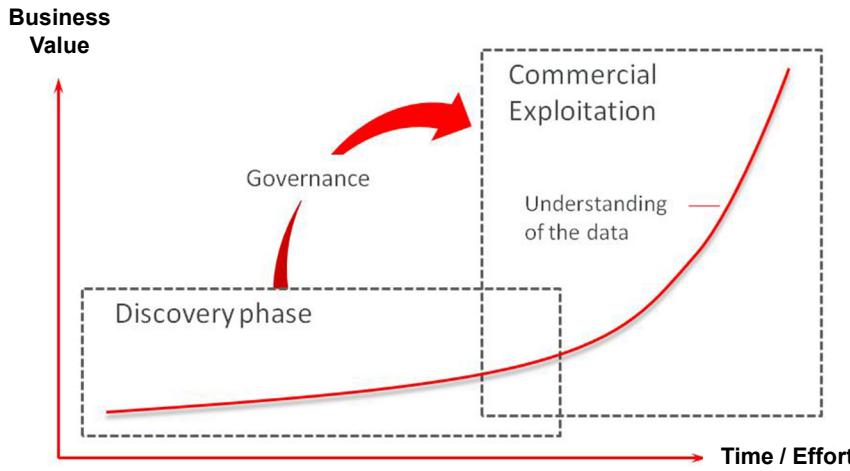
Few organizations would suggest their IM systems are perfect or could not be improved. Many consist of multiple fragmented silos of information, each of which contains a narrow scope of data and serves a limited user community. As a result of the fragmentation and lack of architectural vision, making simple changes (provision new data, new reports, changing hierarchies, and so on) becomes difficult resulting in both business and IT frustrations.

1. Given the complicated IM environment that already exists, the key question you may be asking yourself is “just how can Big Data technologies be applied to create additional business value or reduce the costs of delivering Information Management?” After all, simply adding Big Data technology to your existing estate will do nothing in of itself, other than add additional costs and complexity to an already costly and complex environment.
2. To address this issue, Oracle has developed a complete business solution that integrates big data into the larger, enterprisewide information management system. The solution integrates structured, semi-structured, and unstructured data by using Oracle Database, bridging the traditional database and big data worlds.
3. Oracle provides a powerful analytics solution by combining big data analytic techniques with a variety of in-Database advanced analytic and BI/DW technologies to provide measurable, fact-based value for all elements in the IM system.

In the remainder of this lesson, you will learn about Oracle’s IM system that integrates and exploits big data as part of an enterprisewide information management system.

A Simple Functional Model for Big Data

Discovery and commercial exploitation for new data



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In order for a company to lead the market it must continually innovate to find new opportunities to add value to existing customers, further optimize business processes, cut costs, or find new markets.

With data at the heart of your fact-based management approach it follows that considerable focus should be placed on:

- Enabling new discoveries
- Operationalizing insights quickly and efficiently

By using Big Data technologies, a company may be able to store data at a lower granularity, keep the data for longer, and apply a range of analytical tools that would until now have been too expensive in traditional relational technologies. There are reasonable grounds to say that more data will outperform a cleverer algorithm almost every time.

The slide illustrates a simplified functional model for the kind of *Analyze > Test > Learn > Optimize* process that is key to leveraging value from data. When driven by a new business challenge or the availability of new data, a Data Scientist investigates the data by bringing it together with other enterprise data by using a variety techniques.

As the figure shows, having discovered something of value in the data, the next challenge is to operationalize the insight in some fashion, often by including it within a business process. Then, one monitors the effect on the business process to ensure the behavior first observed on the workbench is exhibited in real life and the results are positive.

Governance

It is important to note that the people, process, and tools used during the discovery phase will be different from those required for commercial exploitation. There is a step required to go from one to the other: Governance. This is shown in the diagram, due to the important role it plays to the success and management of information.

The goal in design is to minimize the time taken for the discovery part of the process and reduce the size of the step between discovery and commercial exploitation.

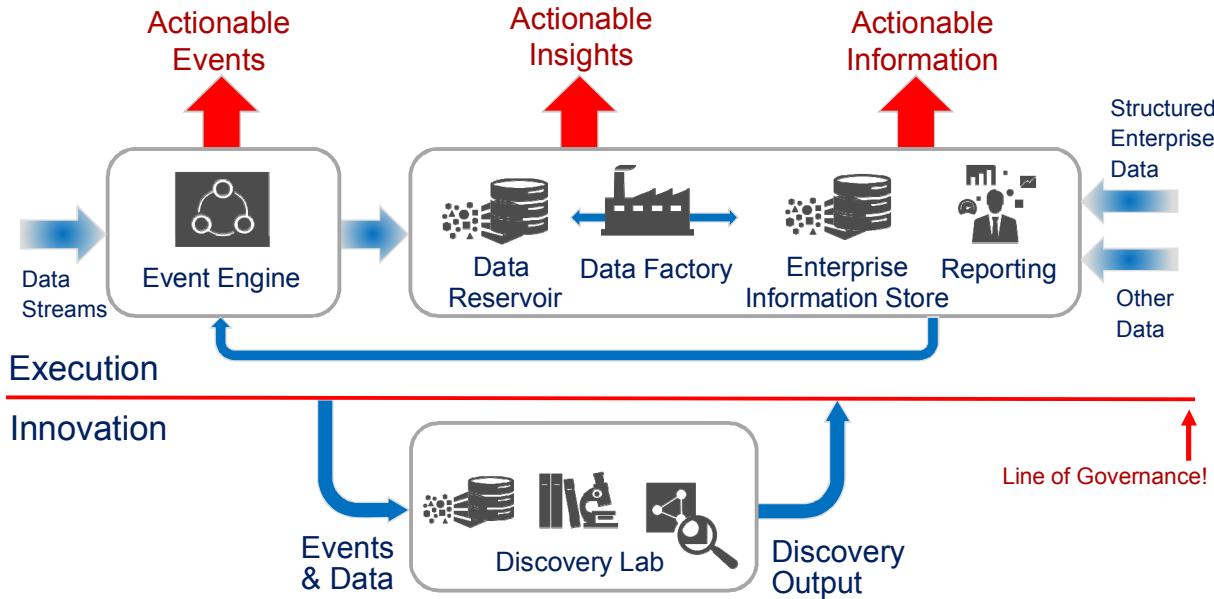
Consider the following:

- The most successful Data Science teams use a wide array of tools and techniques in discovery phase.
- In contrast, most successful organizations have a very limited set of tools for exploitation and enforce strict coding standards.
- Therefore, the governance step is required to reproduce the insights that are produced in the discovery phase – by using a rationalized production toolset that applies coding standards and security constraints.

As part of the “route to production” for every new insight, careful consideration must be applied to ensure:

- The behavior is reproducible in the broader commercial context, and
- How or when the model needs refreshing due to a decline in performance over time.

Oracle Information Management Conceptual Architecture



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle's Information Management Conceptual Architecture, shown in the slide, illustrates key components and flows in a compact form. It highlights in particular how the innovation derived from analytical discovery is somewhat separated from the day-to-day execution, with the governance function previously discussed linking the two.

When defining the role of Big Data components within information architectures, it is helpful to align components and their purposes along the flow of data into the larger data environment. In this compact, flow-based conceptual model, we define a set of components, many of which may be present in your existing information architectures:

- **Event Engine:** Includes components that process data in-flight to identify actionable events. It then determines the *next-best-action* based on decision context and event profile data. Finally, it persists the data in a durable storage system.
- **Data Reservoir:** Economical, scale-out storage and parallel processing for data that does not have stringent requirements for formalization or modeling. Typically manifested as a Hadoop cluster or staging area in a relational database.
- **Data Factory:** Management of data into and between the Data Reservoir and Enterprise Information Store as well as the rapid provisioning of data into the Discovery Lab for agile discovery

Enterprise Information Store: Store for large scale formalized and modeled business critical data, typically manifested by an (Enterprise) Data Warehouse. When combined with a Data Reservoir, these form a Big Data Management System.

Reporting: BI tools and infrastructure components for timely and accurate reporting

Discovery Lab: A set of data stores, processing engines, and analysis tools that are separate from the everyday processing of data. This component facilitates the discovery of new knowledge of value to the business, and includes the ability to provision new data into the Discovery Lab from outside the architecture. This component is also represented as the “Discovery phase” function in the figure on the previous slide.

Line of Governance. The interplay of these components and their assembly into solutions can be further simplified by dividing the flow of data into:

- *Execution:* Tasks that support and inform daily operations
- *Innovation:* Tasks that drive new insights back to the business

Arranging solutions on either side of this division (as shown by the red line in the figure) helps inform system requirements for security, governance, and timeliness.

Note:

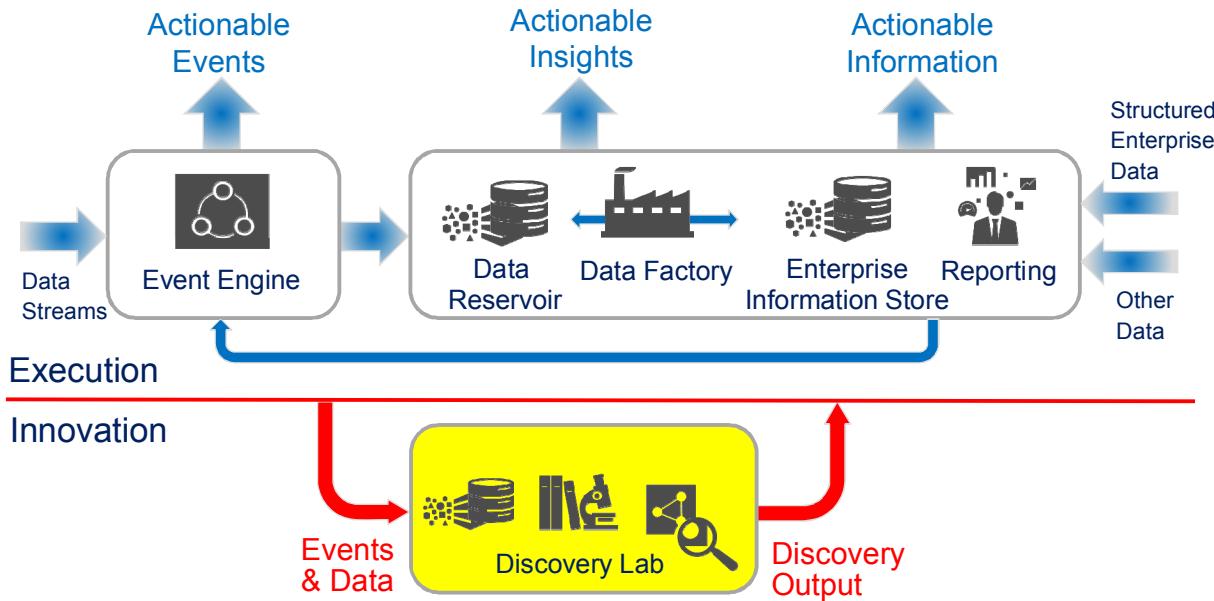
An organization’s priorities define the scope of the deployed solution, especially in the initial project phases.

- For many organizations, the most important first step is often to deliver a Discovery Lab to prove the potential value of their data and analytics overall.
- For others where investment decisions and the project drive have come from senior business executives, a broader implementation scope involving more components has been adopted.

Next:

Using the conceptual architecture diagram, we will now examine how separate elements of the design are used together to deliver different benefits. These combinations are called “Design Patterns.”

IM Architecture Design Pattern: Discovery Lab



ORACLE

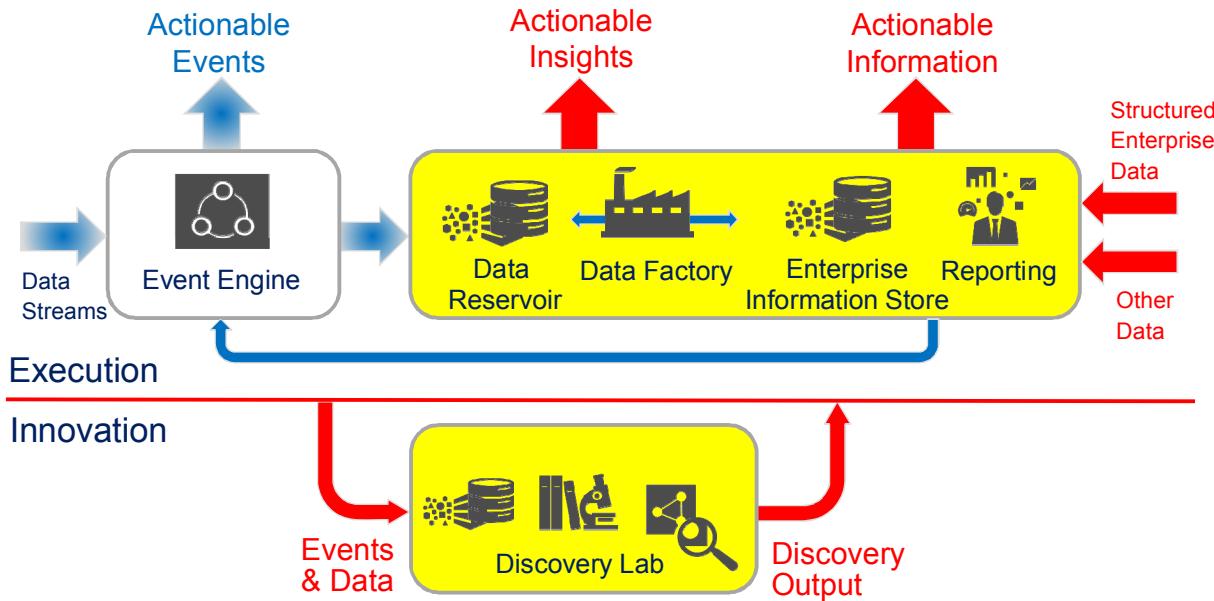
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Discovery Lab is itself a distinct design pattern within the conceptual architecture.

It is characterized by the following:

- Specific focus on identifying commercial value for exploitation
- Small group of highly skilled individuals (aka Data Scientists, Data Mining practitioners, and so on)
- Iterative development approach—data oriented and not development oriented
- Wide range of tools and techniques applied
- Data is provisioned through the Data Factory or own ETL
- Typically has a separate infrastructure, but could also be a unified Data Reservoir if the resource is managed effectively
- Discovery Lab outputs that may include new knowledge, data mining models or parameters, scored data, and others

IM Architecture Design Pattern: Information Platform



ORACLE

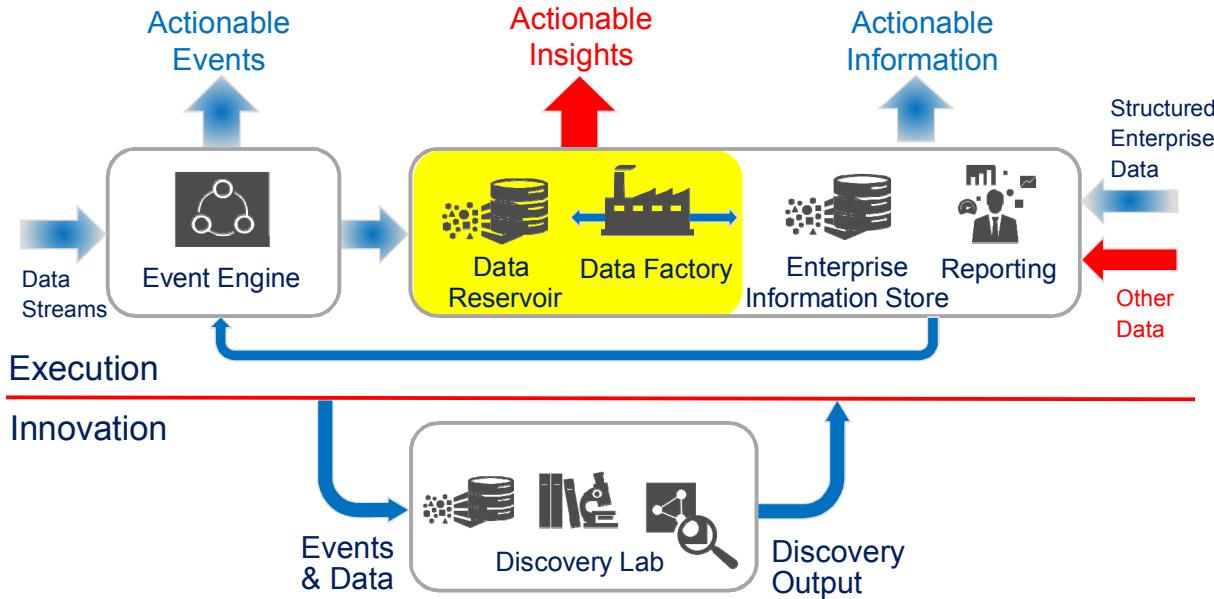
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Information Platform is another design pattern derived from the IM conceptual architecture.

It may be characterized by the following:

- A framework to build the next generation Information Management platform
- Driven by either Business Strategy or IT cost or capability initiatives
- Initial project that may be specifically linked to lower data grain or retention, but it is the platform as a whole that forms the solution required
- Define key issues related to differences in procurement, development process, governance, and skills differences
- Is a platform into which other IM assets may be consolidated

IM Architecture Design Pattern: Data Application



ORACLE

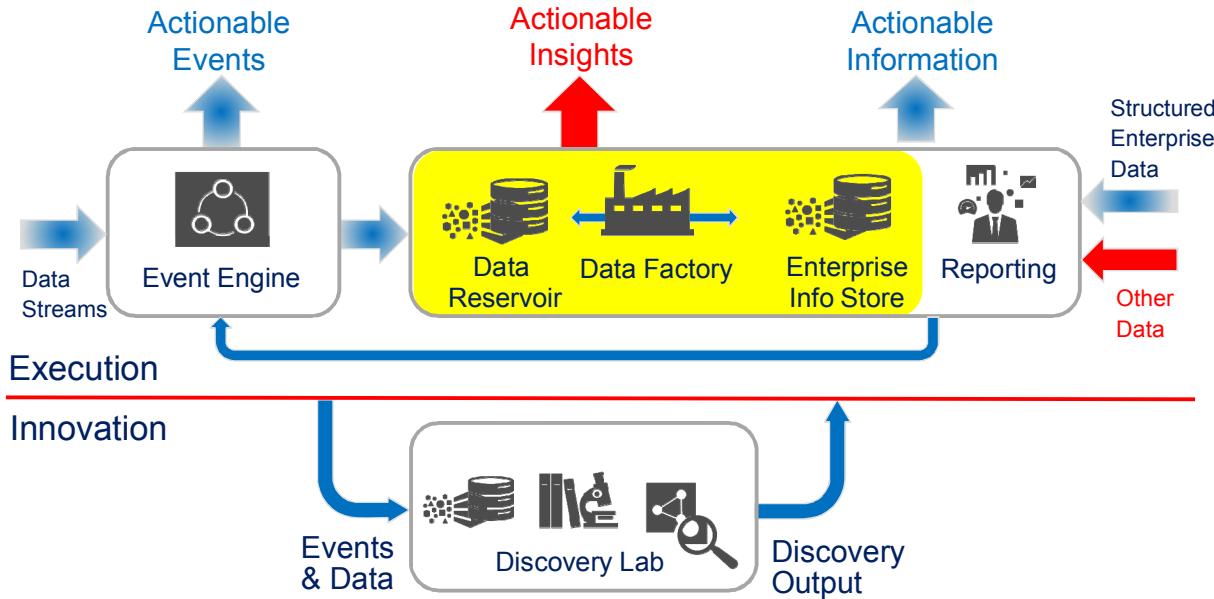
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Another design pattern derived from the IM conceptual architecture is called the Data Application pattern.

It is characterized by the following:

- Big Data technologies being applied to a specific business problem. For example,
 - Genome sequence analysis using BLAST
 - Log data from pharmaceutical production plant and machinery required for traceability
- Limited or no integration to the broader Information Management system
- Specific solution so that nonfunctional requirements have less impact on solution quality or long term costs
- Sensitive to considerations such as platform costs and scalability

IM Architecture Design Pattern: Information Solution



ORACLE

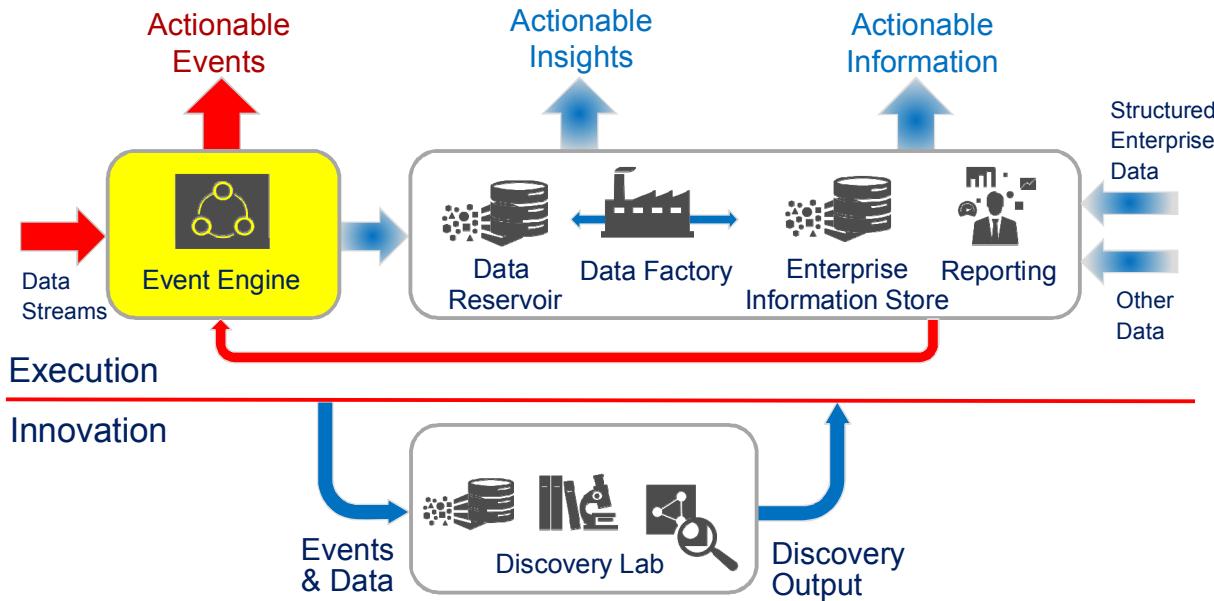
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Expand the Data Application pattern to include the Enterprise Information Store, and you have Information Solution design pattern.

It is characterized by the following:

- A specific solution based on Big Data technologies requiring broader integration to the wider Information Management system, such as:
 - An ETL pre-processor for the Data Warehouse
 - To affordably store a lower level of grain in the data
- Nonfunctional requirements are more critical in a solution of this nature.
- Scalable integration to IM system is an important factor for success.
- Analysis may take place in the Data Reservoir, or the Reservoir may be used only as an aggregator.

IM Architecture Design Pattern: Real-Time Events



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Finally, the Event Engine is a primary element in a design pattern called Real-Time Events. Real-Time Events are characterized by the following:

- May take place at multiple locations between place of data origination and the data center—requiring careful design and implementation
- May include declarative rules and Data Mining technologies to optimize decisions
- May include considerations for personal preferences and privacy (for example, opt-out) for customer-related events
- Are common components seen across many industries and markets

Design Patterns to Component Usage Map

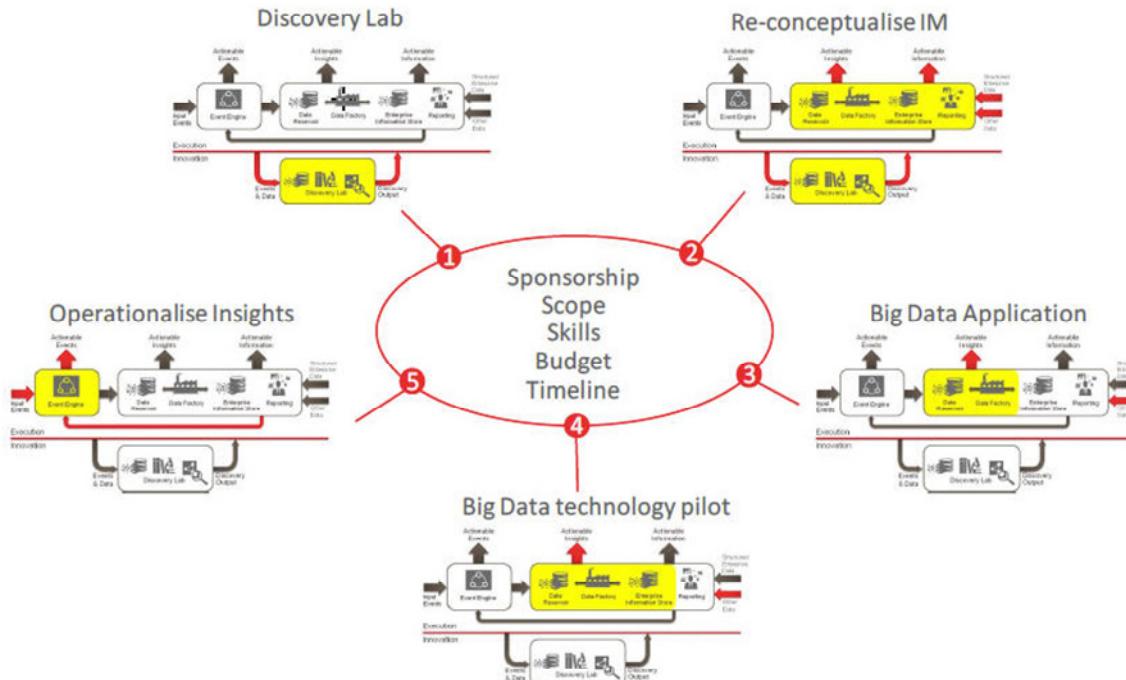
Design Pattern	Discovery Lab	Information Platform	Data Application	Information Solution	R/T Events
Outline	Data science lab Assess the value of the data	Next Generation information platform to align IM capability with business strategy	Addresses a specific data problem in Hadoop with no broader integration required.	Addresses a specific data problem but requires broader enterprise-wide integrations	Execution platform to respond to R/T events
Examples	Gov. Healthcare Mobile operator	US Bank (Bus. led) UK Gov. Dept. (Tech. led)	Pharma Genome project	Bank – trade risk Mobile Operator – ETL processing	Mobile operator – location based offers
Data Engine				Possible	Yes
Data Reservoir		Yes	Yes	Yes	
Data Factory		Yes	Yes	Yes	
Enterprise Data		Yes			
Reporting		Yes			
Discovery Lab	Yes	Implied	Alt to Reservoir + Factory		



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

By using the Oracle Information Management Conceptual Architecture, this table maps the components to the related design patterns from a usage perspective.

Big Data Adoption and Implementation Patterns



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Adoption in the context of Big Data can be a challenge.

Evidence suggests two distinctly different adoption paths. Broadly speaking, one is IT lead and the other is more business lead. Not only is there a difference in sponsorship, but also in the scope of the initial projects:

- IT lead adoption commonly chooses a much narrower scope as a way of learning about the technologies and forcing through required changes to the organizational processes previously highlighted.
- Business led adoption is normally strategic in nature, more far reaching in scope and impact, and driven by business strategy.

The scope and phasing of the initial projects as the points of integration and additional capabilities will have an impact on other systems.

A range of implementation patterns are shown in the slide, each of which has a different scope and focus.

These implementation patterns are shown in the slide.

Implementation Patterns

Common implementation patterns include:

1. **Discovery Lab:** Focus on rolling out discovery capabilities only. There is no broader integration with operationalizing insight.
2. **Re-conceptualize IM:** Broader implementation that adds Big Data capabilities, especially in enabling the Discovery Lab capability, and revisits the overall design of the IM solution.
3. **Big Data Application:** Adoption of Big Data technologies to address a specific application. For example, a pharmaceuticals company who now stores and processes genome data by using Hadoop. The data is only required for research so there is no broader integration with the rest of the company's enterprise data.
4. **Big Data Technology Pilot:** Typically focused on adding Big Data-specific tooling to an existing IM estate (for example, adding "messy" data that was previously unmanaged to the existing enterprise data store).
5. **Operationalize Insights:** Here the focus is more on operationalizing insights, often using NoSQL and next-best-action tooling. This kind of project often follows the successful roll-out of additional Discovery Lab capabilities.

Adoption Strategy

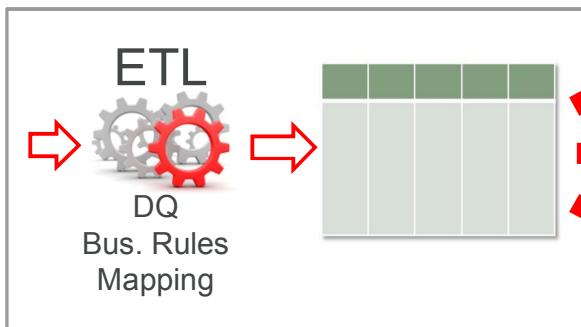
Adoption in the wider context is also important to consider. Oracle has long advocated the need for Information Management to be a first class system in IT. As discussed in this lesson:

- The way information is managed and exploited is becoming of increasing importance in today's ultra competitive world, so it has become critical to think through the IM implications when dealing with any operational application.
- In addition, as applications themselves become more operationally linked to IM systems, the characteristics of those systems must now also be matched.

It follows that if an Information Management system is to be considered as a first class system, IT must do a better job of keeping up with the rate of change of the business, enabling it, not constraining it.

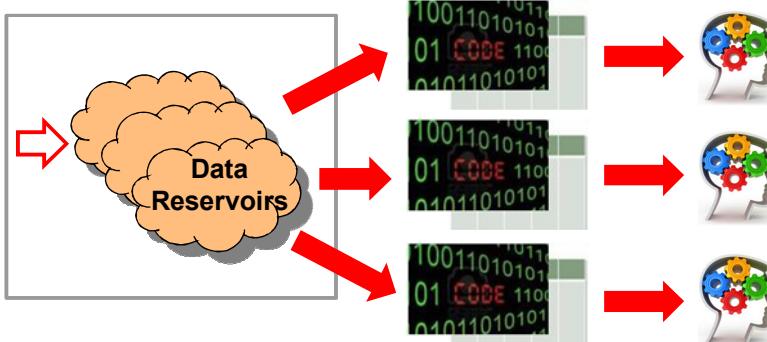
Oracle's Reference Architecture does this through design. As well as clearly defined abstraction layers that limit the impact of change we call out in particular the role of the Virtualization & Query Federation in addition to the Discovery Sandboxes to support iterative development and discovery.

IM Architecture Data Approaches: Schema-on-Write vs Schema-on-Read



Traditional “Schema on Write”

- Data quality managed by formalized ETL process
- Data persisted in tabular, agreed, and consistent form
- Data integration happens in ETL
- Structure must be decided before writing



Big Data “Schema on Read”

- Interpretation of data captured in code for each program accessing the data
- Data quality dependent on code quality
- Data integration happens in code

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Finally, this slide provides a short discussion on the differing approaches to incorporating unstructured and structured data into an information management system.

One key advantage often cited for the Big Data approach is the flexibility afforded by the “schema-on-read” approach when compared to the more structured, formal, and rigid approach imposed by a “schema-on-write” data model, which is typically employed in more traditional Data Warehouses.

It seems clear that “schema-on-read” and “schema-on-write” approaches both have a place in a contemporary Information Management platform.

The differences between the approaches and the typical boundary of IT’s responsibility is illustrated in the slide.

Schema-on-Write

Schema-on-write describes the standard ETL approach for landing data into a relational schema. In this approach, data is sourced and processed in several distinct steps. In most Data Warehouses data quality is validated as part of this process and data is integrated together before writing into the Foundation Data Layer schema. The ETL process used to achieve this would have gone through the usual Systems Development Life Cycle including a full suite of testing.

Schema-on-Read

Hadoop and other Big Data technologies most typically use a schema-on-read approach where data is loaded onto the storage platform typically unmodified by the mechanism used to land the data. A number of mechanisms can *then* be used to interpret the data and present it in a tabular fashion.

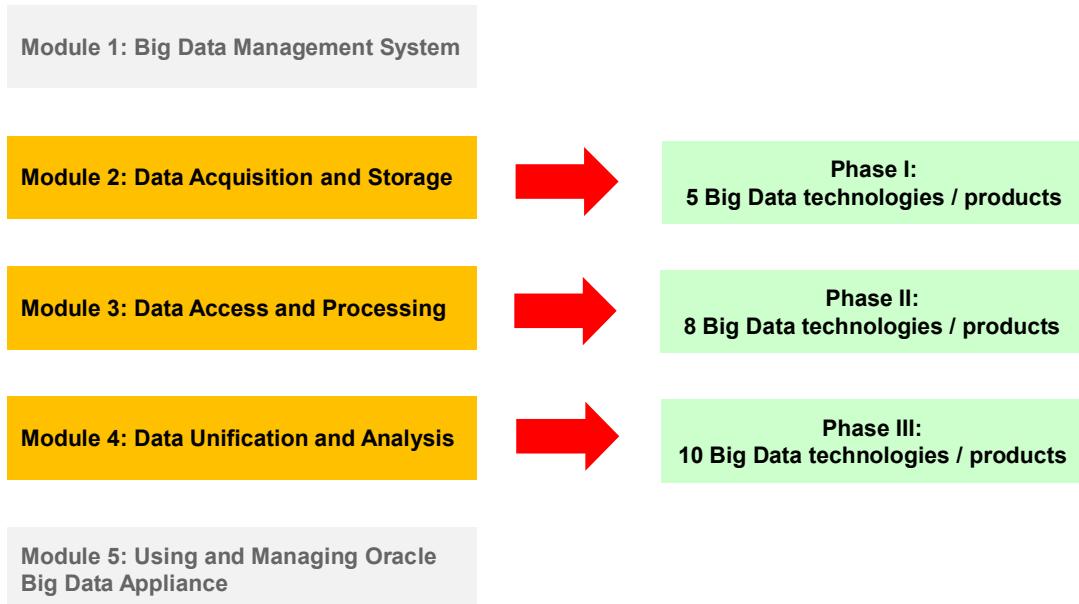
The best example of this is perhaps MapReduce, where the Mapper parses the data and makes it available for further processing. In this model, the quality of the data output is clearly a function of the code that is written (or generated). Crucially, as the interpretation happens at the point in time the data is physically read, the Reader must manage any data differences over time. For example, where source data has changed over time, a small change to the Map:Reduce code could result in significant differences in results if left unchecked.

Contrasting Costs

The two approaches to schema also have a bearing on processing costs and supportability.

- In schema-on-read, the data quality is dependent on the serialization or de-serialization code. And - the cost of code execution may be repeated each time the data is accessed.
- Also, as the data being accessed may be many years ago, it may well be difficult to find people who still understand the data sufficiently to advise on any changes required to the accessing programs.

Course Approach: Big Data Project Phases



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You have just been given a significant amount of conceptual information about Oracle's Information Management Architecture for Big Data. Stated another way: this is the *rationale* behind Oracle's approach and strategy for leveraging and integrating Big Data within the larger Information Management system.

Going forward in this course, the goal is to provide you with concrete, hands-on strategies for leveraging the value of Big Data – as illustrated in the slide.

The purpose of the first module is provide a context for this strategy.

Modules 2 – 4 provide the strategy: the methods, techniques, and technologies for working with Big Data. These modules represent three common phases of any Big Data project:

- Phase I (Module 2): Data Acquisition and Storage
- Phase II (Module 3): Data Access and Processing
- Phase III (Module 4): Data Unification and Analysis

These three phases comprise Oracle's strategy for working with, and leveraging the value of, Big Data.

Finally, in the last module you learn the fundamental techniques for using and managing Oracle Big Data Appliance.

Goal of Oracle's IM System for Big Data

Leverage and Integrate Unstructured and Structured Data
to maximize value to the Enterprise



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In this lesson, we have discussed a range of Big Data adoption and implementation patterns that may be used within the larger enterprise wide Information System architecture.

Within this architecture, there are two significantly differing approaches to capturing, storing, and using data, as described in the slide on Schema-on-Write vs Schema-on- Read.

However, Big Data technology is just that—a technology. Just like any technology it is not the technology itself that is as important as the solutions they can enable, and the business value that can be derived.

Big Data technologies, when combined with relational technologies and others such as Data Warehousing, Spatial, BI, OLAP, and many more, offer the promise of unlocking the untapped potential that exists within the broader set of data that comprises an Information Management system.

Therefore, the goal of Oracle's IM system for Big Data is to provide an architecture and platform that enables organizations to integrate unstructured (and semi-structured) data, commonly found in Big Data environments, with structured data, most commonly found in RDBMS environments.

And, the goal of this class is to show you how to use and integrate the best technologies of both environments, enabling an integration of Hadoop, NoSQL, and relational data to maximize value to the enterprise.

Additional Resources

- The Oracle Big Data Learning Library
https://apexapps.oracle.com/pls/apex/f?p=44785:141:0:::141:P141_PAGE_ID,P141_SECTION_ID:27,615
- Information Management and Big Data: A Reference Architecture (Oracle White Paper)
<http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/2297765.pdf>



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This slide provides a link to the Oracle Big Data Learning Library landing page, and also to the Oracle White Paper that provides greater detail on the principles and concepts covered in this lesson.

Summary

In this lesson, you should have learned how to:

- Define the term Big Data
- Identify the challenges and opportunities in implementing big data
- Describe the Oracle Information Management Architecture for Big Data
- Describe Oracle's technology approach for Big Data adoption



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

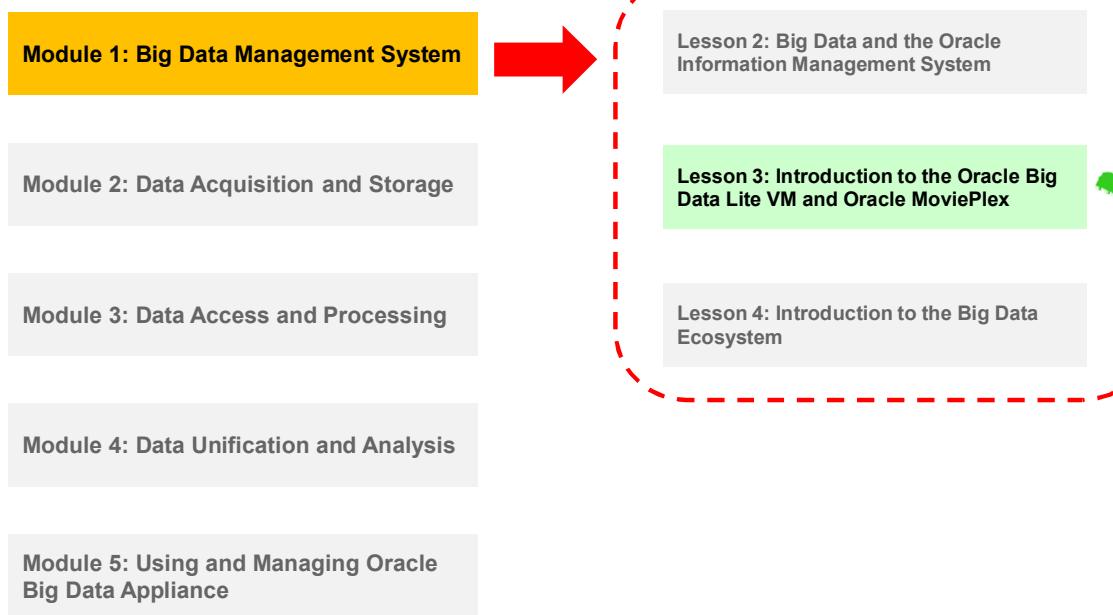
Using Oracle Big Data Lite Virtual Machine



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This lesson covers two main topics: The Oracle Big Data Lite Virtual Machine, which you will use in this course, and also an introduction to the Oracle MoviePlex application. Some of the practices in this course are based on the Oracle MoviePlex application.

Objectives

After completing this lesson, you should be able to use:

- Oracle Big Data Lite Virtual Machine
- The Oracle MoviePlex application



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- Overview of Oracle Big Data Lite Virtual Machine
 - Environment details
 - Software installed
- Deep dive into the Oracle MoviePlex case study



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle Big Data Lite Virtual Machine: Introduction

Provides an integrated environment to help you get started with the Oracle Big Data platform



ORACLE®

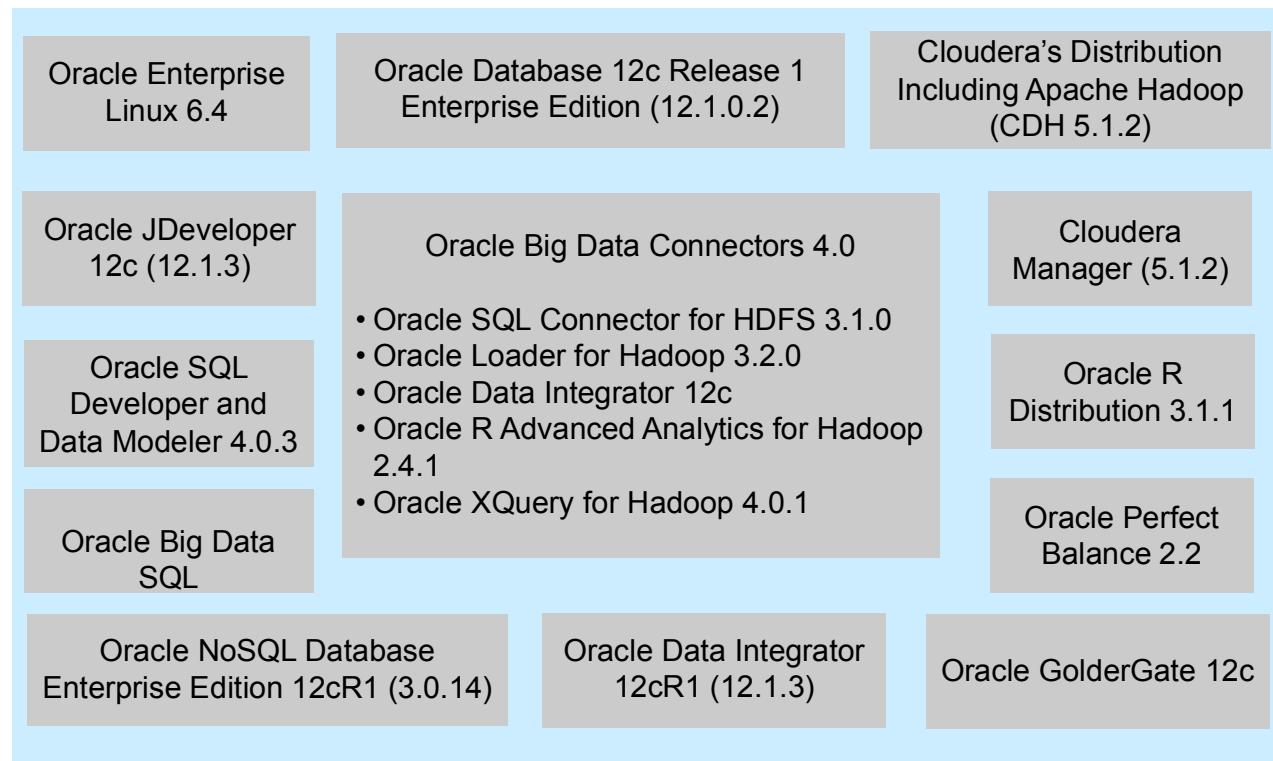
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle Big Data Lite Virtual Machine provides an integrated environment to help you get started with the Oracle Big Data platform. Many Oracle Big Data platform components have been installed and configured, enabling you to begin using the system without difficulty.

In this course, we use Oracle Big Data Lite Virtual Machine 3.0 to perform the practices and demonstrations. For more information, see the following page:

http://www.oracle.com/technetwork/database/bigdata-appliance/oracle-biqdatalite-2104726.html#getting_started

Oracle Big Data Lite 4.0.1 VM Components



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Initializing the Environment for the Oracle Big Data Lite VM

Steps	Commands
1. Start VM and log in.	<code>id/password: welcome1</code>
2. Download and install RStudio (perform this task only once).	<code>cd /home/oracle/scripts ./install_rstudio.sh</code> (uncomment proxy setting in shell script if behind Oracle firewall)
3. Open a Terminal window.	Right-click on the desktop and select Open Terminal .
4. Go to the moviedemo home.	<code>cd movie/moviedemo/scripts</code>
5. Start Oracle NoSQL DB and the Movie Application .	First time (or every time you want to initialize Oracle NoSQL DB). This will take several minutes: <code>./1_start_movieapp.sh</code>



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Initializing the Environment

Steps	Commands
6. Start Hue .	<code>./2_start_hue.sh</code>
7. Start Flume . Minimize the Flume Window. Show the App Log .	<code>./3_flume_tail_movielog_master.sh</code>
8. Start the Oracle NoSQL DB Console App .	<code>./4_nosqldb_console.sh</code>
9. Start Firefox by clicking the web browser icon.	
10. Start the movie application by going to the Oracle MoviePlex bookmark (guest1/welcome1). <ul style="list-style-type: none">In a second tab, connect to Hue by using the bookmark (<code>oracle/oracle</code>).In a third tab, go to the Flume master by using the bookmark under Hadoop Links.In a fourth tab, go to RStudio by using the bookmark.	



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

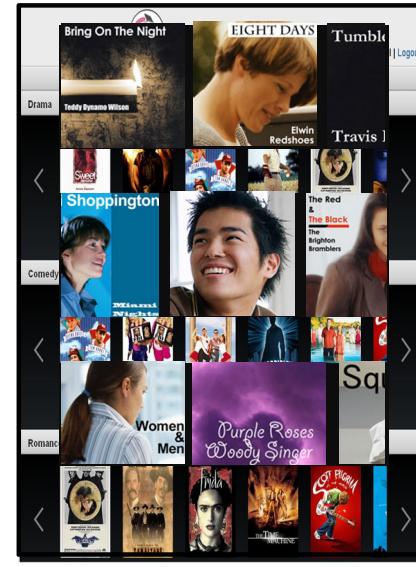
- Overview of Oracle Big Data Lite Virtual Machine
 - Environment details
 - Software installed
- Deep dive into the Oracle MoviePlex case study



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle MoviePlex Case Study: Introduction

- Oracle MoviePlex is an application that is based on a fictitious online movie streaming rental company.
- With this web-based application, you can do the following:
 - Browse a catalog of movies.
 - Watch movie trailers.
 - Rent movies.
 - Review and rank movies.

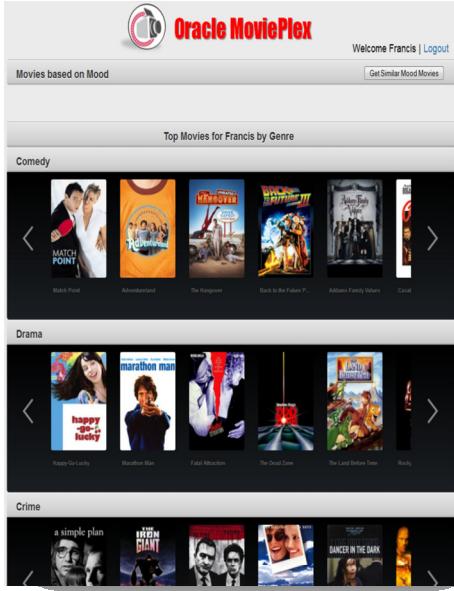


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

When users log in to Oracle MoviePlex, they see a targeted list of movies based on their previous viewing behavior. Because of this personalized experience and the application's reliable and fast performance, users spend a lot of money with the company. Oracle MoviePlex has become extremely profitable.

In the next few slides, you learn about the requirements for this application, and you also learn why Oracle NoSQL Database is an appropriate choice for its development.

Introduction

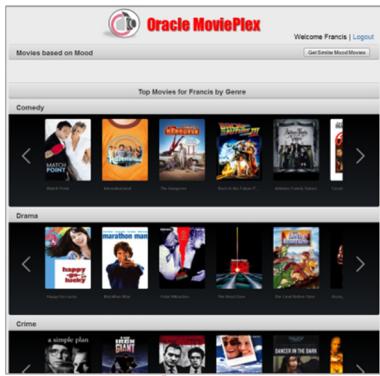


- Like many other online stores, they needed a cost-effective approach to tackle their “big data” challenges.
- They recently implemented Oracle Big Data Platform to better manage their business, identify key opportunities, and enhance customer satisfaction.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Big Data Challenge



- Applications are generating massive volumes of unstructured data that describes user behavior and application performance.
- How can companies fully capitalize on this valuable information due to cost and complexity?
- How do you use this raw data to gain better insights into your customers, enhance their user experience, and ultimately improve profitability?

```
{
  "custId":1185972,"movieId":null,"genrelId":null,"time":"2012-07-01:00:00:07","recommended":null,"activity":8}
  {"custId":1354924,"movieId":1948,"genrelId":9,"time":"2012-07-01:00:00:22","recommended":N,"activity":7}
  {"custId":1083711,"movieId":null,"genrelId":null,"time":"2012-07-01:00:00:26","recommended":null,"activity":9}
  {"custId":1234182,"movieId":11547,"genrelId":44,"time":"2012-07-01:00:00:32","recommended":Y,"activity":7}
  {"custId":1010220,"movieId":11547,"genrelId":44,"time":"2012-07-01:00:00:42","recommended":Y,"activity":6}
  {"custId":1143971,"movieId":null,"genrelId":null,"time":"2012-07-01:00:00:43","recommended":null,"activity":8}
  {"custId":1253676,"movieId":null,"genrelId":null,"time":"2012-07-01:00:00:50","recommended":null,"activity":9}
  {"custId":1351777,"movieId":608,"genrelId":6,"time":"2012-07-01:00:01:03","recommended":N,"activity":7}
  {"custId":1143971,"movieId":null,"genrelId":null,"time":"2012-07-01:00:01:07","recommended":null,"activity":9}
  {"custId":1363545,"movieId":27205,"genrelId":9,"time":"2012-07-01:00:01:18","recommended":Y,"activity":7}
  {"custId":1067283,"movieId":1124,"genrelId":9,"time":"2012-07-01:00:01:26","recommended":Y,"activity":7}
  {"custId":1126174,"movieId":16309,"genrelId":9,"time":"2012-07-01:00:01:35","recommended":N,"activity":7}
  {"custId":1234182,"movieId":11547,"genrelId":44,"time":"2012-07-01:00:01:39","recommended":Y,"activity":7}
  {"custId":1067283,"movieId":null,"genrelId":null,"time":"2012-07-01:00:01:55","recommended":null,"activity":9}
  {"custId":1377537,"movieId":null,"genrelId":null,"time":"2012-07-01:00:01:58","recommended":null,"activity":9}
  {"custId":1347836,"movieId":null,"genrelId":null,"time":"2012-07-01:00:02:03","recommended":null,"activity":8}
  {"custId":1137285,"movieId":null,"genrelId":null,"time":"2012-07-01:00:03:39","recommended":null,"activity":8}
  {"custId":1354924,"movieId":null,"genrelId":null,"time":"2012-07-01:00:03:51","recommended":null,"activity":9}
  {"custId":1036191,"movieId":null,"genrelId":null,"time":"2012-07-01:00:03:55","recommended":null,"activity":8}
  {"custId":1143971,"movieId":1017161,"genrelId":44,"time":"2012-07-01:00:04:00","recommended":Y,"activity":7}
  {"custId":1363545,"movieId":27205,"genrelId":9,"time":"2012-07-01:00:04:03","recommended":Y,"activity":5}
  {"custId":1273464,"movieId":null,"genrelId":null,"time":"2012-07-01:00:04:39","recommended":null,"activity":9}
  {"custId":1346299,"movieId":424,"genrelId":1,"time":"2012-07-01:00:05:02","recommended":Y,"activity":4}
```

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Derive Value from Big Data

How can you:

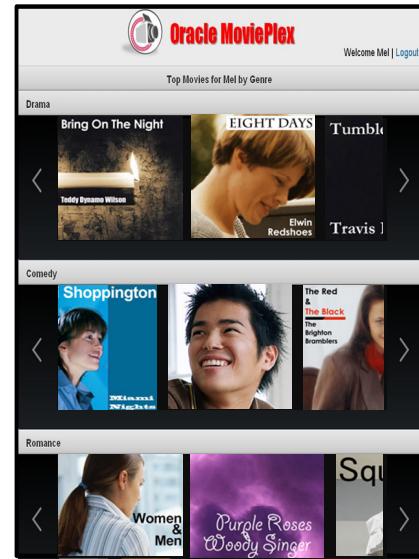
- Make the right movie offers at the right time?
- Better understand the viewing trends of various customer segments?
- Optimize marketing spend by targeting customers with optimal promotional offers?
- Minimize infrastructure spend by understanding bandwidth usage over time?
- Prepare to answer questions that you haven't thought of yet!



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle MoviePlex: Goal

- To deliver a personalized movie-watching experience by collecting and storing:
 - User profiles
 - Movie listings
 - Ratings
 - User's viewing location within paused movie
- All customer information and session details are completely fictitious.



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The goal of the Oracle MoviePlex application is to deliver a personalized experience to each customer. When it is used more and more frequently, the application is able to understand the customer's browsing pattern and displays content that the customer prefers.

To achieve this goal, the application must collect and store all user details, movie listings, movie ratings, user interactions (such as the position where the movie trailer was paused), and so on.

Although, this particular case study pertains to movies, the requirements can be related to any other enterprise where there is a need to better manage the business by identifying key opportunities and enhancing customer satisfaction.

Oracle MoviePlex: Big Data Challenges

- The application generates a huge volume of unstructured data.
- Requests require fast response times (measured in milliseconds).
- Data point: Latency matters.



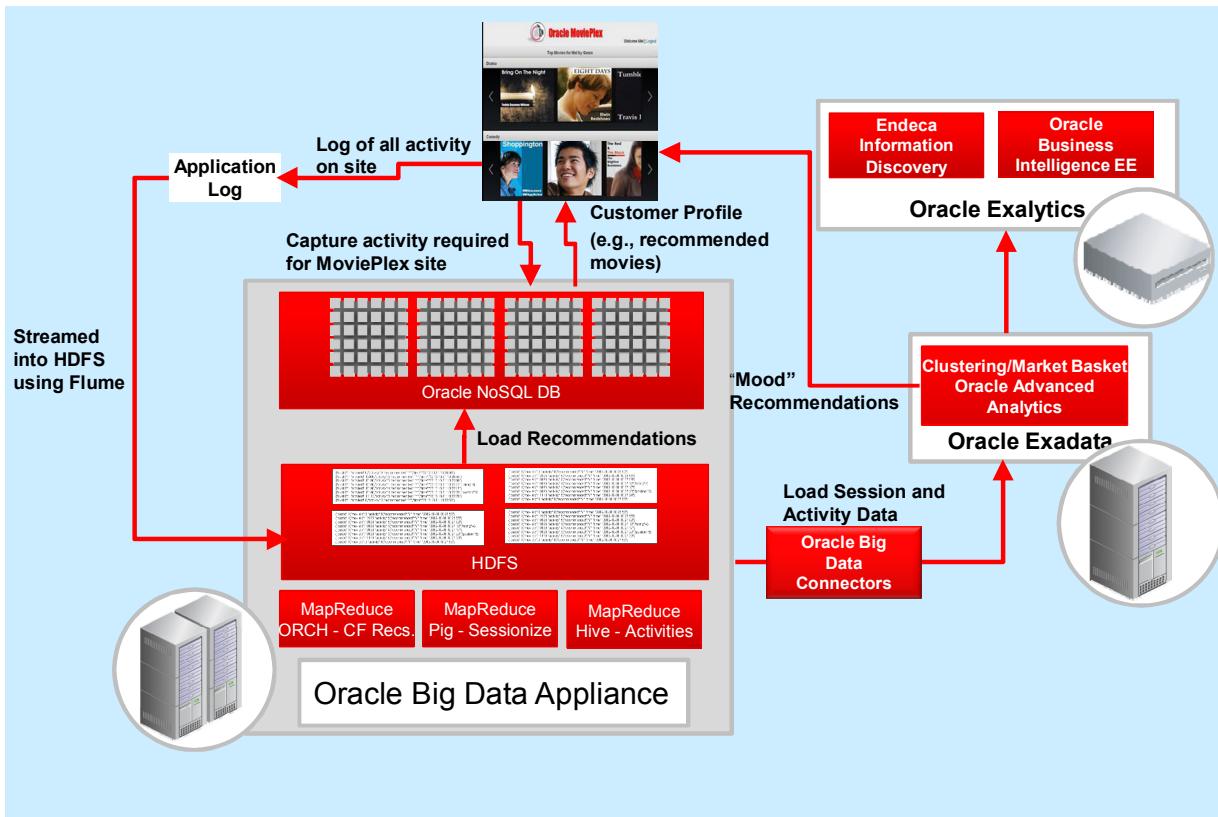
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Applications today are generating massive volumes of unstructured data that describe user behavior and application performance. Most companies are unable to fully capitalize on this potentially valuable information due to cost and complexity.

As a developer for MoviePlex, how would you capitalize on this raw data to gain better insights into your customers, enhance their user experience, and ultimately improve profitability for the application?

Oracle MoviePlex: Architecture



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This slide shows the overall architecture of the Oracle MoviePlex application. The processes that occur within the Oracle Big Data Appliance and other engineered systems are also shown.

Handling Big Data Challenges

The MoviePlex application requires a cost-effective approach to handle its big data challenges. Before understanding how Oracle NoSQL Database can be used to better manage this online business, consider some of the challenges in developing this application.

The application captures every user click. The resulting log file that is generated is estimated to contain huge volumes of semi-structured and simple data with details of the movies that users viewed, the position until which trailers were watched, reviews and ratings that users provided, and so on. In addition, the application must provide these services on a large web scale, supporting thousands of customers accessing and renting thousands of movies.

To give users a satisfying experience, the application must deliver content with minimal latency. Each user profile must be retrieved and updated with minimal latency (that is, with a lightning-fast response time measured in milliseconds).

Oracle MoviePlex: Data Generation

```
{"custId":1185972,"movield":null,"genrelid":null,"time":"2012-07-01:00:00:07","recommended":null,"activity":8}
 {"custId":1354924,"movield":1948,"genrelid":9,"time":"2012-07-01:00:00:22","recommended":"N","activity":7}
 {"custId":1083711,"movield":null,"genrelid":null,"time":"2012-07-01:00:00:26","recommended":null,"activity":9}
 {"custId":1234182,"movield":11547,"genrelid":44,"time":"2012-07-01:00:00:32","recommended":"Y","activity":7}
 {"custId":1010220,"movield":11547,"genrelid":44,"time":"2012-07-01:00:00:42","recommended":"Y","activity":6}
 {"custId":1143971,"movield":null,"genrelid":null,"time":"2012-07-01:00:00:43","recommended":null,"activity":8}
 {"custId":1253676,"movield":null,"genrelid":null,"time":"2012-07-01:00:00:50","recommended":null,"activity":9}
 {"custId":1351777,"movield":608,"genrelid":6,"time":"2012-07-01:00:01:03","recommended":"N","activity":7}
 {"custId":1143971,"movield":null,"genrelid":null,"time":"2012-07-01:00:01:07","recommended":null,"activity":9}
 {"custId":1363545,"movield":27205,"genrelid":9,"time":"2012-07-01:00:01:18","recommended":"Y","activity":7}
 {"custId":1067283,"movield":1124,"genrelid":9,"time":"2012-07-01:00:01:26","recommended":"Y","activity":7}
 {"custId":1126174,"movield":16309,"genrelid":9,"time":"2012-07-01:00:01:35","recommended":"N","activity":7}
 {"custId":1234182,"movield":11547,"genrelid":44,"time":"2012-07-01:00:01:39","recommended":"Y","activity":7}
 {"custId":1067283,"movield":null,"genrelid":null,"time":"2012-07-01:00:01:55","recommended":null,"activity":9}
 {"custId":1377537,"movield":null,"genrelid":null,"time":"2012-07-01:00:01:58","recommended":null,"activity":9}
 {"custId":1347836,"movield":null,"genrelid":null,"time":"2012-07-01:00:02:03","recommended":null,"activity":8}
 {"custId":1137285,"movield":null,"genrelid":null,"time":"2012-07-01:00:03:39","recommended":null,"activity":8}
 {"custId":1354924,"movield":null,"genrelid":null,"time":"2012-07-01:00:03:51","recommended":null,"activity":9}
 {"custId":1036191,"movield":null,"genrelid":null,"time":"2012-07-01:00:03:55","recommended":null,"activity":8}
 {"custId":1143971,"movield":1017161,"genrelid":44,"time":"2012-07-01:00:04:00","recommended":"Y","activity":7}
 {"custId":1363545,"movield":27205,"genrelid":9,"time":"2012-07-01:00:04:03","recommended":"Y","activity":5}
 {"custId":1273464,"movield":null,"genrelid":null,"time":"2012-07-01:00:04:39","recommended":null,"activity":9}
 {"custId":1346299,"movield":424,"genrelid":1,"time":"2012-07-01:00:05:02","recommended":"Y","activity":4}
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This slide shows a sample of the data that is generated in the MoviePlex application. As mentioned earlier, the application captures every user click to generate unstructured data in the log file.

Oracle MoviePlex: Data Generation Format

Fields	Description
<code>custId</code>	The customer's ID
<code>movieId</code>	The ID of the selected movie
<code>genreId</code>	The genre of the selected movie
<code>Time</code>	The timestamp when the customer watched the movie
<code>recommended?</code>	Whether or not the selected movie recommended, Y or N
<code>Activity</code>	<ul style="list-style-type: none">• 1: Rate movie• 2: Completed movie• 3: Not completed• 4: Started movie• 5: Browsed movie• 6: List movies• 7: Search

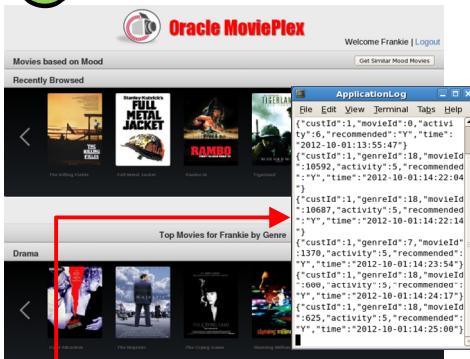
```
{"custId":1354924,"movieId":1948,"genreId":9,"time":"201  
2-07-01:00:00:22","recommended":"N","activity":7}
```



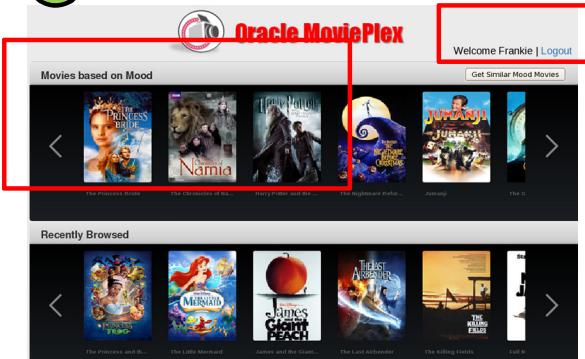
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle MoviePlex Application

1 Simple profile updates



2 Advanced analytics: movies based on mood



4 Advanced profile attributes

```

=====
What do you want to do?
1. Show customer profile
2. Show information about a movie
3. Clear browsed and watched lists
Enter action [1] : 1
Enter customer [guest1]: guest1

Customer Information
-----
key : /CUST/guest1/-/info
value : {"id":1,"name":"Frankie","email":"Frankie.Nash@oraclemail.com","username":"guest1","password":"0n\u0000bN0o-\u0000oR\-\I\u0000f"}

elapsed: 6ms

snippet:
key = Key.fromString("/CUST/guest1/-/info");
value = kvstore.get(key);
  
```

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Application Tasks

- Simple profile updates:** The user profile contains everything required to drive the personalized experience, like the movies recommended to the customers. Deliver a personal experience to every user. Each user profile must be retrieved and updated with minimal latency.
- Advanced analytics (movies based on mood):** Provide a compelling user experience and deliver targeted recommendations based on a user's "current mood."
- Key-value store:** This is a little console application that lets you see what a key-value store is. When you pass a key, you get back a value. This helps customers visualize a key-value store; each user profile must be retrieved and updated with minimal latency. The challenge is to service this request in real time and at scale.
- Advanced profile attributes:** Deliver a personal experience to every user and deliver targeted recommendations based on past movie-viewing habits.

Each of these tasks has different challenges and use different products, such as Oracle NoSQL Database, Oracle Advanced Analytics, and Oracle R Advanced Analytics for Hadoop.

Summary

In this lesson, you should have learned how to use:

- Oracle Big Data Lite Virtual Machine
- The Oracle MoviePlex application



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 3: Overview

This practice is a hands-on exercise that checks your understanding of the lesson topics, guiding you to take a quick tour of the Big Data Lite 4.0.1 VM.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

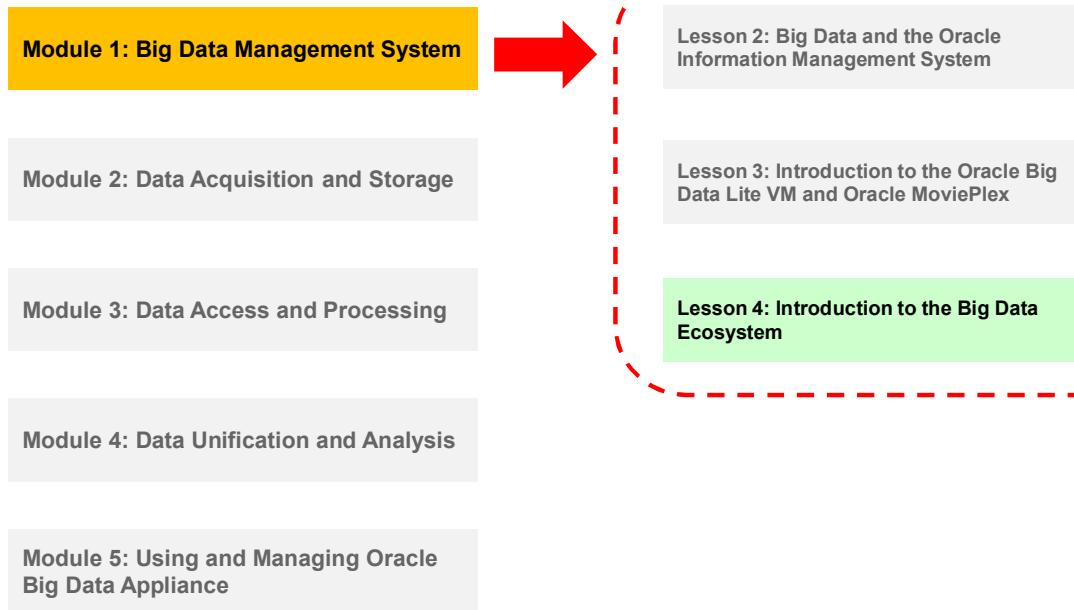
Introduction to the Big Data Ecosystem



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In this lesson, you discuss about the overall Hadoop Ecosystem, describe at a very high level the Hadoop core components, Cloudera's Distribution Including Apache Hadoop (CDH), and mention some of the other Hadoop-related projects.

Objectives

After completing this lesson, you should be able to:

- Define the *Hadoop Ecosystem*
- Describe the Hadoop core components
- Choose a Hadoop Distribution and Version by using:
 - Apache Software Foundation, or
 - Cloudera's Distribution Including Apache Hadoop (CDH)
- List some of the other related projects in the Hadoop Ecosystem



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Computer Clusters

- A computer rack (commonly called a rack) is a metal frame used to hold various hardware devices such as servers, hard disk drives, and other electronic equipment.
- A computer cluster is a single logical unit consisting of multiple computers (or racks) that are linked through a fast local area network (LAN).
- The components of a cluster, nodes (computers used as a servers), run their own instance of an operating system.
- A node typically includes CPU, memory, and disk(s) storage.



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A computer cluster is a single logical unit consisting of multiple computers that are linked through a LAN. The networked computers essentially act as a single, much more powerful machine. A computer cluster provides much faster processing speed, larger storage capacity, better data integrity, superior reliability, and wider availability of resources.

Distributed Computing

- Distributed computing is a technique that allows individual computers to be networked together.
- A distributed file system is a client/server application that allows clients to access and process data stored on the server as if it were stored on their own computer.
- File systems that manage the storage across a network of machines are called distributed file systems.



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Apache Hadoop

- Apache Hadoop is an open-source software framework for distributed storage and distributed processing of Big Data on clusters of commodity hardware.
- Open-source available:
 - From Apache Hadoop Foundation
 - As Distributions such as Cloudera's Distribution Including Apache Hadoop (CDH)
- Distributed storage with HDFS
 - Massive amounts of data
 - HDFS sits on top of a native file system such as Linux
- Distributed and parallel processing with MapReduce



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hadoop was originally built by a Yahoo! engineer named Doug Cutting and was named after his son's yellow stuffed elephant. Hadoop is now an open source project managed by the Apache Software Foundation as you will see in the next few slides. It is made available under the Apache License v2.0 along with other projects that are covered in this lesson at a high level. Later lessons in this course cover some of the related Hadoop projects. Hadoop is a fundamental building block in capturing and processing big data. At a high level, Hadoop is designed to make parallel the data processing across computing nodes (servers) to speed computations and hide latency.

Hadoop is a batch data processing system for enormous amounts of data. It is designed to process huge amounts of structured and unstructured data (terabytes to petabytes) and is implemented on racks of commodity servers as a **Hadoop cluster**. Servers can be added or removed from the cluster dynamically because Hadoop is designed to be “self-healing.” In other words, Hadoop is able to detect changes, including failures, and adjust to those changes and continue to operate without interruption.

Parallel file systems are a type of clustered file systems that spread data across multiple storage nodes, usually for redundancy or performance.

Types of Analysis That Use Hadoop

- Text mining
- Index building
- Graph creation and analysis
- Pattern recognition
- Collaborative filtering
- Prediction models
- Sentiment analysis
- Risk assessment



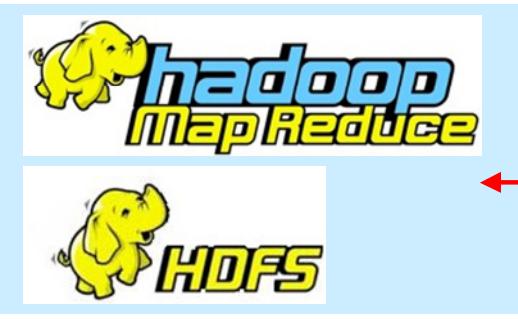
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The slide lists some of the technical areas in which the Hadoop Ecosystem can be used to derive better analysis.

Apache Hadoop Ecosystem



Hadoop partial list of Associated Projects



- Hadoop Core Components:
- HDFS (Storage)
 - MapReduce (Processing)

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Hadoop Ecosystem is a framework that enables distributed storage and processing of huge amounts of data in a cost-effective way.

Oracle Big Data Appliance uses Hadoop Ecosystem to:

- Store and process data
- Provide scaling without limits
- Solve the various problems that are encountered in big data

The Hadoop refers to an ecosystem of related products.

Apache Hadoop Core Components

- Apache Hadoop Distributed File System (HDFS)
 - Master-Slave architecture
 - Based on Google's File System (GFS)
 - Stores and distributes data across the nodes in the cluster
 - Redundant (reliability)
 - Fault tolerant (high availability)
 - Supports MapReduce (distributed/parallel processing)
 - Scalable (out instead of up)
- MapReduce
 - Responsible for parallel processing of the data on the cluster
 - Distributed



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Apache Hadoop contains two main core components:

- Hadoop Distributed File System (HDFS) is a distributed file system for storing information and sits on top of the OS that you are using.
- MapReduce is a parallel processing framework that operates on local data whenever possible. It abstracts the complexity of parallel processing. This enables developers to focus more on the business logic rather than on the processing framework.

Hadoop enables parallel processing of large data sets because the distributed file system has been designed to work in conjunction with the distributed processing framework. This allows for clusters with thousands of nodes.

HDFS stores large files. It breaks those files into blocks of 64, 128, or 256 megabytes (MB). These blocks are then distributed across the cluster and replicated numerous times. The default replication factor is 3. This replication has two main purposes:

1. The redundancy yields fault tolerance. If a server fails, the other servers (two) automatically take over its workload.
2. It allows for colocating processing (MapReduce) and data (HDFS). The goal is to move the processing to the data to achieve better performance.

You can specify the block or "chunk" size and the number of the replication factor.

Oracle Big Data Appliance (BDA) uses Cloudera's Distribution including Apache Hadoop (CDH). On the BDA, 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 workload.

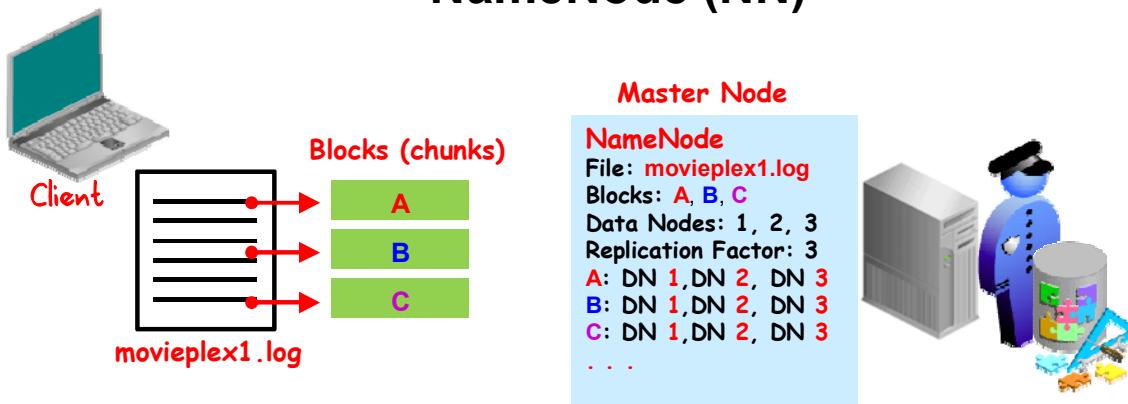
HDFS Key Definitions

Term	Description
Cluster	A group of servers (nodes) on a network that are configured to work together. A server is either a master node or a slave (worker) node.
Hadoop	A batch processing infrastructure that stores and distributes files and distributes work across a group of servers (nodes).
Hadoop Cluster	A collection of Racks containing master and slave nodes.
Blocks	HDFS breaks down a data file into blocks or "chunks" and stores the data blocks on different slave DataNodes in the Hadoop cluster.
Replication Factor	HDFS makes three copies of data blocks and stores on different DataNodes/Racks in the Hadoop cluster.
NameNode (NN)	A service (Daemon) that maintains a directory of all files in HDFS and tracks where data is stored in the HDFS cluster.
Secondary NameNode	Performs internal NameNode transaction log checkpointing
DataNode (DN)	Stores the blocks "chunks" of data for a set of files



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

NameNode (NN)



NameNode Stores File System Metadata

- File information (name, updates, replication factor, etc.)
- File and blocks information and locations
- File to blocks mappings
- Access rights to the file
- Number of files in the cluster
- Number (and health) of DataNodes in the cluster
- etc.

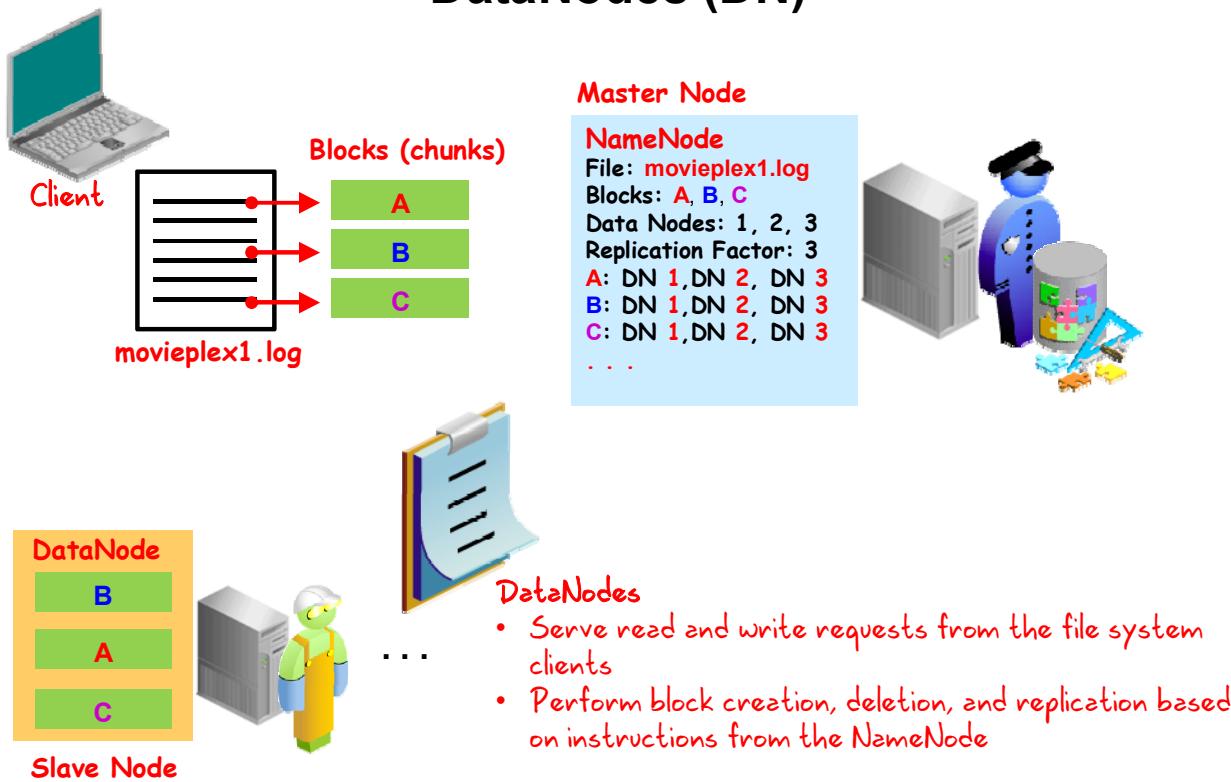
ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace (metadata) and controls access to files by client applications. In addition, there are several DataNodes, usually one per node in the cluster, which manage disks storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into blocks or "chunks" and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines and maintains the mapping of blocks to DataNodes in the cluster. The DataNodes are responsible for serving read and write requests from the file system's client applications. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

The NameNode is the service that stores the file system metadata and maintains a complete picture of the file system. Clients connect to the NameNode to perform file system operations. Block data is streamed to and from DataNodes directly, so bandwidth is not limited by a single node. DataNodes regularly report their status to the NameNode in a heartbeat. This means that, at any given time, the NameNode has a complete view of all of the DataNodes in the cluster, their current health, and what blocks they have available.

DataNodes (DN)



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

DataNodes use local disks in the commodity server for persistence. All the data blocks are stored locally, which improves performance. As mentioned earlier, the data blocks are replicated across several DataNodes. The default replication factor is 3 but that number is configurable. If one of the servers that contain one of the file's block fails, the file in question might not be corrupted. You can choose the degree of replication and the number of the DataNodes in the cluster when you implement the cluster. HDFS is dynamic in nature; therefore, all of the parameters can be adjusted during the operation of the cluster.

The replicated block is stored on different computers in the cluster so that if one of the computers failed, you can still access the block from another computer.

Note: The diagram in the slide will be explained in more detail in a later lesson.

MapReduce Framework

- A software framework that enables you to write applications that will process large amounts of data, in-parallel, on large clusters of commodity hardware, in a reliable and fault-tolerant manner
- Integrates with HDFS and provides the same benefits for parallel data processing
- Sends computations to where the data is stored
- The framework:
 - Schedules and monitors tasks, and re-executes failed tasks
 - Hides complex distributed computing complexity from the developer



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Benefits of MapReduce

MapReduce provides:

- Automatic parallelization and distribution of large data sets that are stored and distributed on a Hadoop cluster slave nodes
- Fault-tolerance
- I/O scheduling
- Status and monitoring



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The MapReduce framework enables fault-tolerant parallel interaction with distributed data. From an industry point of view, MapReduce frameworks enable programmers to write efficient, distributed, data-parallel programs that can operate on various platforms. These frameworks provide flexibility to support a wide variety of programs and data. Programmers must decide and code on the filtering logic.

The MapReduce framework is not suitable for real-time processing and filtering because of its batch-oriented nature.

MapReduce Job

- A MapReduce job is a unit of work requested by a client.
- The job consists of:
 - Input data (HDFS)
 - A MapReduce program (user-defined)
- Hadoop runs the job by dividing it into tasks:
 - Map tasks (user-defined)
 - Shuffle and sort tasks (MapReduce)
 - Reduce tasks (user-defined)



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A MapReduce job is a unit of work that the client application submits. The unit of work consists of the input data (which is already stored, distributed, and replicated in HDFS as blocks) and the MapReduce program created by the developer.

Hadoop runs the job by dividing it into tasks such as the Map and Reduce tasks, which are created by the developer. There is also a Shuffle and Sort phase, which sends the intermediate data from the Mappers to the Reducers.

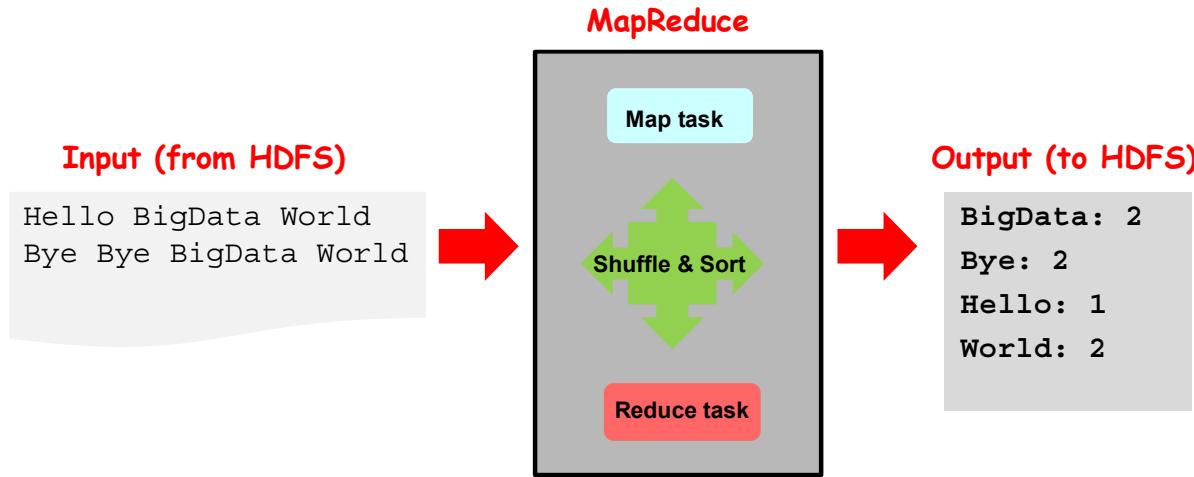
There are two types of nodes that control the job execution process:

- A JobTracker daemon that runs on the master node
- A number of TaskTrackers that run on the slave nodes, one per slave node

The JobTracker coordinates all the jobs run on the system by scheduling tasks to run on TaskTrackers. TaskTrackers run tasks, and send progress reports, block reports, and heartbeats on a regular basis to the JobTracker. The JobTracker keeps a record of the overall progress of each job. If a task fails, the JobTracker can simply reschedule it on a different TaskTracker on a different slave node.

MRv2 uses the ResourceManager master daemon, several NodeManagers slave daemons, Application Masters, and Containers to run the job.

Simple Word Count MapReduce: Example

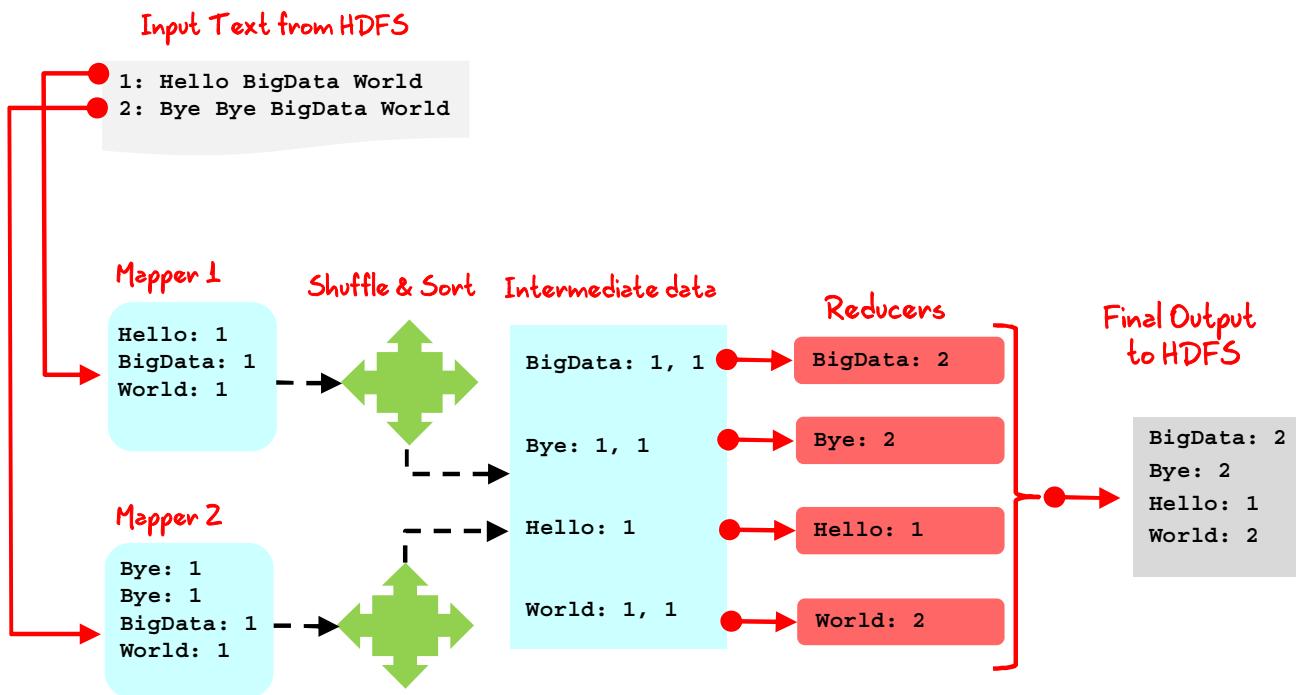


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the simple example in the slide, we want to use MapReduce to count the number of word occurrences in a text input file. The output shows the sorted list of the number of occurrences of each word.

Simple Word Count MapReduce: Example



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows a simple Word Count MapReduce example. The input text is divided into key-value pairs. The Mapper will get as input key-value pairs. The key in this case is the offset into the file, which is not useful in our example, and the value will be the line of text. The goal of the mapper function is to parse the sentence and emit each word as a key, with a value "1". The reducer will receive each key (that is, the word) and a value "1," and simply add up the 1's to compute the word count. The mapper task will input key-value pairs to a set of intermediate key-value pairs. After the MapReduce process, you can see that the text is reduced to another key-value pair to eliminate redundancies.

Maps are the individual tasks that transform input data sets into intermediate data sets. The transformed intermediate data sets need not be of the same type as the input data sets. The given input pair can map to zero or many output pairs.

MapReduce Scales Linearly: Example

Consider a system in which data is distributed across five nodes. In a MapReduce framework, Map operations are run in parallel on subsets of the data on each node. This enables the processing of 50 pieces of data simultaneously (for example, a SELECT with the WHERE clause in SQL). Similarly, you can perform aggregation on each of the nodes in parallel (for example, a JOIN ON clause).

When you add a new node to the cluster, data is automatically and evenly distributed to it. This means that the same process now has access to 60 parallel Map tasks and six parallel aggregations.

MapReduce Versions

- **MapReduce 1** (MRv1 or classic MapReduce):
 - Uses the **JobTracker** and **TaskTracker** daemons
 - Runs only MapReduce applications
 - MapReduce 1 is covered in detail in the lesson titled "Introduction to MapReduce."
- **YARN** (Yet Another Resource Negotiator or MRv2):
 - Uses the **ResourceManager/NodeManager** daemons
 - Runs MapReduce applications, Impala, Spark, and so on
 - Adds the advantage of YARN where ResourceManager is now a scheduler and a resource manager
 - YARN is covered in detail in the lesson titled "Resource Management Using YARN."



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Choosing a Hadoop Distribution and Version

1



The Apache Software Foundation
<http://www.apache.org/>





CDH

- Cloudera Express
- Cloudera Enterprise
- Cloudera Manager
- Cloudera Navigator
- Cloudera Director
- Cloudera Support
- Professional Services

2

Cloudera Enterprise

Hadoop for the Enterprise

Cloudera Enterprise helps you build enterprise capabilities you need in critical environments, Cloudera Enterprise, as well as [advanced system management](#) from our world-class team of Had

[Download Now >](#)

Cloudera Enterprise, with Apache

- **Unified** – one integrated system for all workloads to one pool of data

CDH

BATCH PROCESSING (MapReduce, Hive, Pig)	ANALYTIC SQL (Impala)	SEARCH ENGINE (Cloudera Search)	MACHINE LEARNING (Spark, MapReduce, Mahout)	STREAM PROCESSING (Spark)	3RD PARTY APPS (Partners)
WORKLOAD MANAGEMENT (YARN)					
STORAGE FOR ANY TYPE OF DATA UNIFIED, ELASTIC, RESILIENT, SECURE (Sentry)					
Filesystem (HDFS)		Online NoSQL (HBase)			
DATA INTEGRATION (Sqoop, Flume, NFS)					

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hadoop is an open source software developed by the Apache Software Foundation (ASF). You can download Hadoop directly from the project website at <http://hadoop.apache.org>.

Cloudera, a company that provides support, consulting, and management tools for Hadoop, also has a distribution of software called Cloudera's Distribution Including Apache Hadoop (CDH). CDH is also an open source software, which is available under the Apache Software License and is free for both personal and commercial use. Just as many open source software companies do for other systems, Cloudera starts with a stable Apache Hadoop release, backports critical fixes, provides packages for a number of different operating systems, and QAs and tests all components for compatibility. Cloudera includes other Apache Hadoop components and guarantees compatibility between components. CDH currently includes Apache Hadoop, Apache HBase, Apache Hive, Apache Pig, Apache Sqoop, Apache Flume, Apache ZooKeeper, Apache Oozie, Apache Mahout, and Hue.

A complete list of components included in CDH is available at <http://www.cloudera.com/hadoop-details/>.

Additional Resources: Cloudera Distribution

<http://www.cloudera.com/content/support/en/documentation.html>

Additional Resources: Apache Hadoop

<http://hadoop.apache.org/>

The screenshot shows the Apache Hadoop documentation page. At the top left is the Apache logo (feather) and the Hadoop logo (yellow dog). The URL in the browser is hadoop.apache.org/docs/stable/. The page title is "Apache > Hadoop". The main content area features the Hadoop logo and the text "The Apache Software Foundation". Below this is a section for "Apache Hadoop 2.5.2". The "Documentation" link in the sidebar is highlighted with a red box. The footer contains the Oracle logo.

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

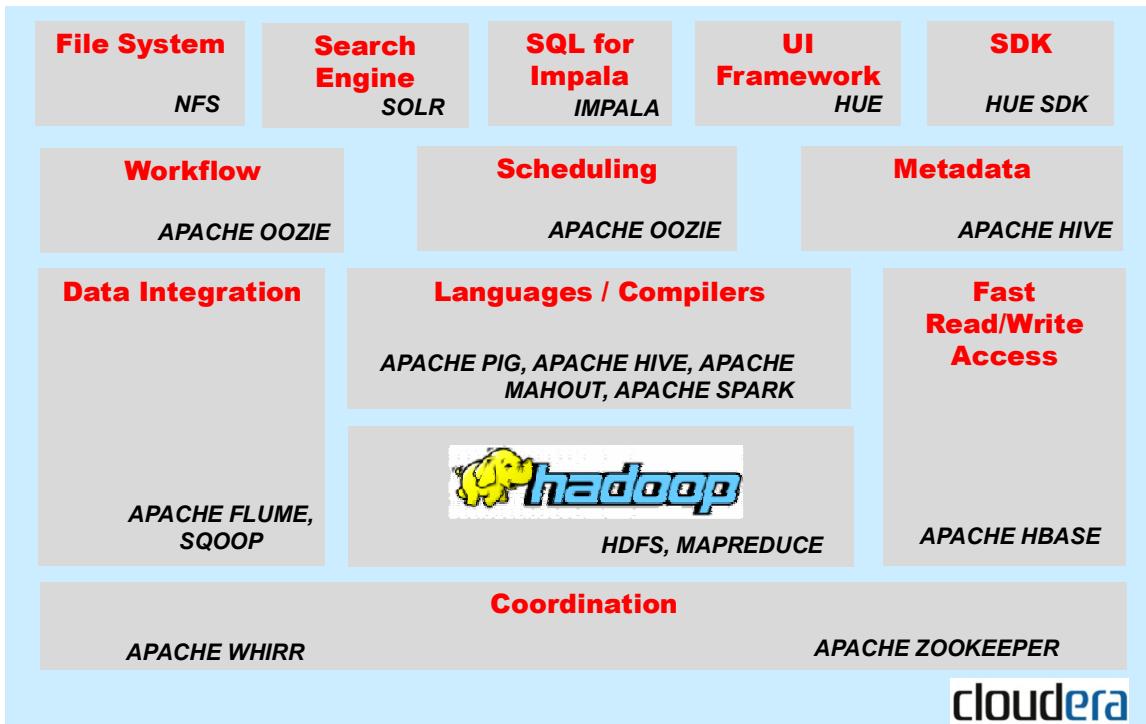
Cloudera's Distribution Including Apache Hadoop (CDH)

- Is a single install package available from the Apache Hadoop core repository
- Includes a stable version of Hadoop, critical bug fixes, and solid new features from the development version
- Includes the following components:
 - Apache Hadoop
 - Hive, Pig, HBase, Solr, Mahout, Spark, YARN
 - Flume, Hue, Oozie, and Sqoop
 - ZooKeeper
- Oracle Big Data Appliance (BDA) optimizes the deployment of CDH on an engineered system to manage and process data.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

CDH Architecture



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

CDH Components

Component	Description
Apache Hadoop	<ul style="list-style-type: none"> A framework for executing applications on a large cluster of servers. It is built for massively parallel processing across a large number of nodes (servers). Consists of the following core components: Hadoop Distributed File System (HDFS) and MapReduce
Hue (Hadoop User Experience)	<ul style="list-style-type: none"> Is an open-source tool Easy to use web front end for viewing files, running queries, performing searches, scheduling jobs, and more Contains several applications to access a Hadoop cluster through a web front end
Apache Oozie	<ul style="list-style-type: none"> Enables developers to create, edit, and submit workflows by using the Oozie dashboard After considering the dependencies between jobs, the Oozie server submits those jobs to the server in the proper sequence.
Apache Spark	<ul style="list-style-type: none"> It is an open source parallel data processing framework. It complements Apache Hadoop. It makes it easy to develop fast, unified Big Data applications combining batch, streaming, and interactive analytics on all your data.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

- Hadoop Distributed File System (HDFS) is a file system that was developed in Java and based on Google's GFS. It is used to store data in the cluster. MapReduce is the system that is used to process data in the Hadoop cluster. You can find more information about Apache Hadoop at <http://hadoop.apache.org/>.
- Hadoop User Experience (Hue) is 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. You can find more information about Hue at <http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/4.2.0/Hue-2-User-Guide/hue2.html>.
- Oozie workflow is a collection of Hadoop Map/Reduce jobs and Pig jobs arranged in a control dependency DAG (Direct Acyclic Graph). "Control dependency" from one action to another means that the second action cannot run until the first action has completed. Oozie workflows contain control flow nodes and action nodes. The Oozie workflow and coordination service runs on the ResourceManager node. You can find more information about Apache Oozie at <http://oozie.apache.org/>.
- Apache Spark (incubating) is an open source, parallel data processing framework that complements Apache Hadoop to make it easy to develop fast, unified Big Data applications combining batch, streaming, and interactive analytics on all your data. It was originally developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010.

CDH Architecture

Component	Description
Apache Solr (Cloudera Search)	<ul style="list-style-type: none"> Cloudera Search is one of Cloudera's near-real-time access products and is powered by Solr. It enables nontechnical users to search and explore data stored in or ingested into Hadoop, Oracle NoSQL Database, and HBase. Users do not need SQL or programming skills to use Cloudera Search because it provides a simple, full-text interface for searching.
Apache Hive	<ul style="list-style-type: none"> Hive Metastore provides a metadata layer that describes the data stored in HDFS. It provides a SQL layer to data on HDFS. It can run SQL queries on HDFS data. It uses Map/Reduce for execution and HDFS for storage.
Apache Pig	<ul style="list-style-type: none"> It is an analysis platform that provides a data flow language called Pig Latin. It is an alternative abstraction on top of MapReduce.
Cloudera Impala	<ul style="list-style-type: none"> The Impala server is a distributed, massively parallel processing (MPP) database engine. It consists of different daemon processes that run on specific hosts within your CDH cluster. The core Impala component is a daemon process that runs on each node of the cluster.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

- Cloudera search is powered by Apache Solr. Using Search with the CDH infrastructure provides:
 - Standard Solr schemas and APIs to define and populate Solr indexes
 - Populate indexes through Apache Flume
 - Index data stored in HDFS, HBase, and Oracle NoSQL Database
 - Allowing end users to easily find data by using faceted search. No knowledge of SQL or APIs is required.
 - Integration with Hue for simplified development
- Hive does not turn the Hadoop cluster into a database. Instead, it provides a query interface to access the flat files on HDFS. You can find more information about Apache Hive at <https://hive.apache.org/>.
- The Impala server is a distributed, massively parallel processing (MPP) database engine. It consists of different daemon processes that run on specific hosts within your CDH cluster. The core Impala component is a daemon process that runs on each node of the cluster, physically represented by the Impala process. It reads and writes to data files; accepts queries transmitted from the Impala-shell command, Hue, JDBC, or ODBC; parallelizes the queries and distributes work to other nodes in the Impala cluster; and transmits intermediate query results back to the central coordinator node.

- Apache Pig is an alternative abstraction on top of the MapReduce framework. Its key features include:
 - Ease of programming
 - Optimization opportunities
 - Extensibility
 - You can find more information about Pig at <http://pig.apache.org>.
- The Pig interpreter runs on the client machine and has the following functions:
 - Taking the PigLatin script and turning it into a series of MapReduce jobs
 - Submitting the jobs to the cluster for execution

CDH Components

Component	Description
Apache Flume	<ul style="list-style-type: none"> A distributed, reliable, available service for efficiently moving large amounts of data as it is generated Ideal for collecting logs from diverse systems and inserting them in HDFS
Apache Sqoop	<ul style="list-style-type: none"> Imports tables from an RDBMS into HDFS Imports data from RDBMS into HDFS as delimited text files or sequence files Generates a class file that can encapsulate a row of the imported data
Apache Hbase	<ul style="list-style-type: none"> Is a NoSQL data store Provides scalable inserts, efficient handling of sparse data, and a constrained data access model
Apache ZooKeeper	<ul style="list-style-type: none"> ZooKeeper is a centralized service for maintaining configuration information, naming, distributed synchronization, and group services. HBase cannot be active without ZooKeeper.
Apache Mahout	<ul style="list-style-type: none"> Scalable machine-learning and data-mining algorithms
Apache Whirr	<ul style="list-style-type: none"> Apache Whirr is a set of libraries for running cloud services. It provides: <ul style="list-style-type: none"> A cloud-neutral way to run services A common service API Smart defaults for services



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

- You can find more information about Flume at <http://flume.apache.org>.
- Sqoop uses MapReduce to import data. You can import either one table or all the tables from an RDBMS. Sqoop also uses a JDBC interface, enabling compatibility with JDBC databases. Sqoop can be used for incremental data imports and for efficiently serializing and deserializing data in subsequent MapReduce jobs. The default import format is “comma-delimited text file.” You can find more information about Apache Sqoop at <http://sqoop.apache.org/>.
- You can find more information about Apache HBase at <http://hbase.apache.org/>.
- You can find more information about Apache ZooKeeper at <http://zookeeper.apache.org>.
- You can find more information about Apache Mahout at <http://mahout.apache.org>.

Where to Go for More Information?

Component	Website
Cloudera Manager	http://www.cloudera.com/content/cloudera/en/products-and-services/cloudera-enterprise/cloudera-manager.html
Apache Hadoop	http://hadoop.apache.org/
fuse-dfs	http://fuse.sourceforge.net/
Cloudera Hue	http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/4.2.0/Hue-2-User-Guide/hue2.html
Apache Oozie	http://oozie.apache.org/
Apache Hive	https://hive.apache.org/
Apache Pig	http://pig.apache.org
Apache Flume	http://flume.apache.org/
Apache Sqoop	http://sqoop.apache.org/
Apache HBase	http://hbase.apache.org/
Apache ZooKeeper	http://zookeeper.apache.org
Apache Mahout	http://mahout.apache.org
Apache Whirr	https://whirr.apache.org/



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Define the *Hadoop Ecosystem*
- Describe the Hadoop core components
- Choose a Hadoop Distribution and Version using:
 - Apache Software Foundation, or
 - Cloudera's Distribution Including Apache Hadoop (CDH)
- List some of the other related projects in the Hadoop Ecosystem



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

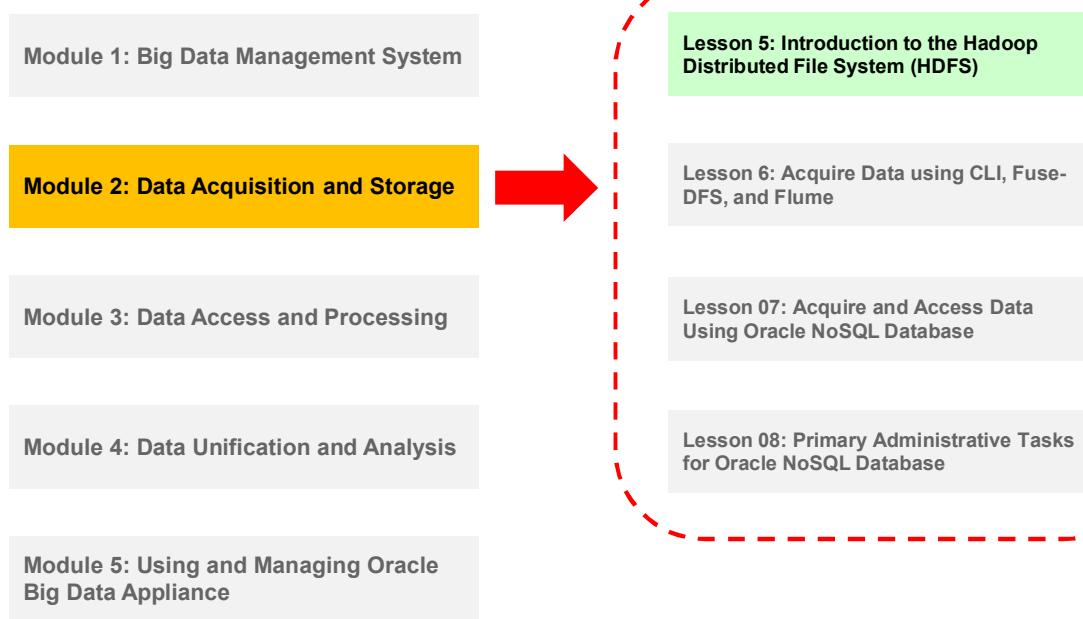
Introduction to the Hadoop Distributed File System (HDFS)



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In this lesson, you learn about the Hadoop Distributed File System (HDFS), describe the architectural components of HDFS, and then list the benefits of using HDFS.

Objectives

After completing this lesson, you should be able to:

- Describe the architectural components of HDFS
- Use the **FS shell** command-line interface (CLI) to interact with data stored in HDFS



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Agenda

- Understand the architectural components of HDFS
- Use the FS shell command-line interface (CLI) to interact with data stored in HDFS



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HDFS: Characteristics

Highly fault-tolerant

High throughput

Suitable for applications with large data sets

Streaming access to file system data

Can be built out of commodity hardware

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hardware failure is the norm rather than the exception in HDFS. An HDFS instance may consist of hundreds or thousands of server machines, each storing part of the file system's data. The fact that there are a huge number of components and each component has probability of failure means that some component of HDFS is always nonfunctional. Therefore, detection of faults, and quick and automatic recovery from failures is a core architectural goal of HDFS.

HDFS is optimized for large, streaming reads and writes rather than low-latency access to many small files. Batch performance is more important than real-time response times.

Applications that run on HDFS have large data sets. A typical file in HDFS is gigabytes to terabytes in size. Thus, HDFS is tuned to support large files. It should provide high aggregate data bandwidth and scale to hundreds of nodes in a single cluster. It should support tens of millions of files in a single instance. HDFS applications need a write-once-read-many access model for files. Once a file is created, written, and closed, it cannot be changed. This assumption enables high throughput data access.

Applications that run on HDFS need streaming access to their data sets. They are not general purpose applications that typically run on general purpose file systems. HDFS is designed more for batch processing rather than interactive use by users. The emphasis is on high throughput of data access rather than low latency of data access.

Use a scale-out model based on inexpensive commodity servers with internal disks rather than RAID to achieve large-scale storage. Accomplish availability and high throughput through application-level replication of data.

HDFS Deployments: High Availability (HA) and Non-HA

- Non-HA Deployment:
 - Uses the NameNode/Secondary NameNode architecture
 - The Secondary NameNode is not a failover for the NameNode.
 - The NameNode was the Single Point of Failure (SPOF) of the cluster before Hadoop 2.0 and CDH 4.0.
- HA Deployment:
 - Active NameNode
 - Standby NameNode



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

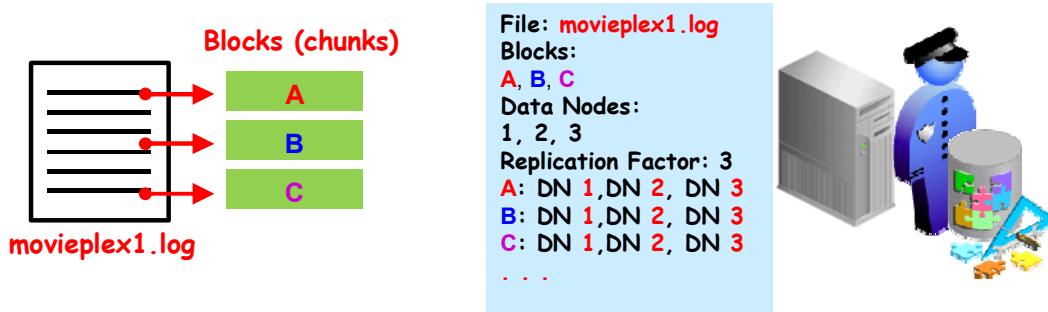
HDFS Key Definitions

Term	Description
Cluster	A group of servers (nodes) on a network that are configured to work together. A server is either a master node or a slave (worker) node.
Hadoop	A batch processing infrastructure that stores and distributes files and distributes work across a group of servers (nodes).
Hadoop Cluster	A collection of Racks containing master and slave nodes
Blocks	HDFS breaks down a data file into blocks or "chunks" and stores the data blocks on different slave DataNodes in the Hadoop cluster.
Replication Factor	HDFS makes three copies of data blocks and stores on different DataNodes/Racks in the Hadoop cluster.
NameNode (NN)	A service (Daemon) that maintains a directory of all files in HDFS and tracks where data is stored in the HDFS cluster.
Secondary NameNode	Performs internal NameNode transaction log checkpointing
DataNode (DN)	Stores the blocks "chunks" of data for a set of files



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

NameNode (NN)



NameNode Stores File System Metadata

- File information (name, updates, replication factor, etc.)
- File blocks information and locations
- Access rights to the file
- Number of files in the cluster
- Number of DataNodes in the cluster
- etc.

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace (metadata) and controls access to files by client applications. In addition, there are several DataNodes, usually one per node in the cluster, which manage disks storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. Internally, a file is split into blocks or "chunks" and these blocks are stored in a set of DataNodes. The NameNode executes file system namespace operations like opening, closing, and renaming files and directories. It also determines and maintains the mapping of blocks to DataNodes in the cluster. The DataNodes are responsible for serving read and write requests from the file system's client applications. The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode.

The NameNode is the service that stores the file system metadata and maintains a complete picture of the file system. Clients connect to the NameNode to perform file system operations. Block data is streamed to and from DataNodes directly, so bandwidth is not limited by a single node. DataNodes regularly report their status to the NameNode in a heartbeat. This means that, at any given time, the NameNode has a complete view of all of the DataNodes in the cluster, their current health, and what blocks they have available.

Functions of the NameNode

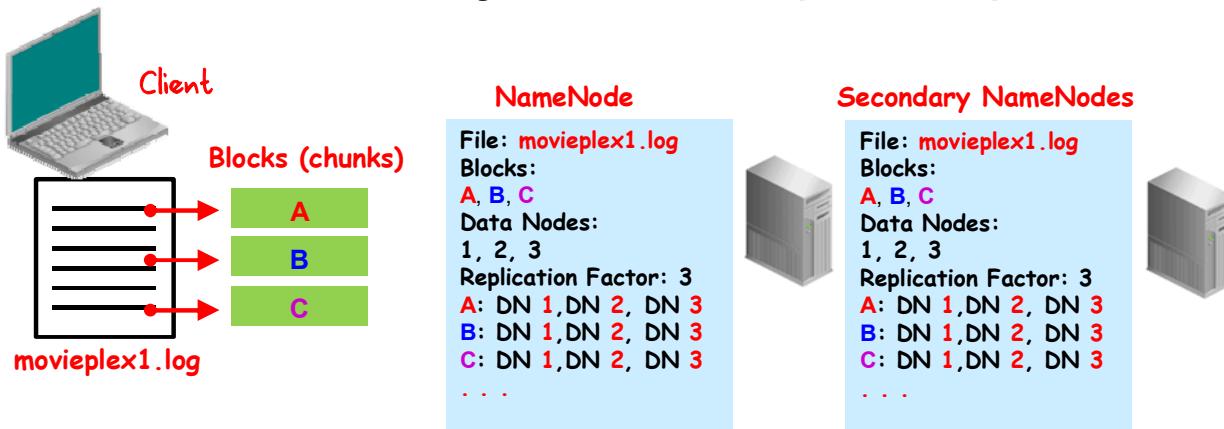
- Acts as the repository for all HDFS metadata
- Maintains the file system namespace
- Executes the directives for opening, closing, and renaming files and directories
- Stores the HDFS state in an image file (`fsimage`)
- Stores file system modifications in an edit log file (`edits`)
- On startup, merges the `fsimage` and `edits` files, and then empties `edits`
- Places replicas of blocks on multiple racks for fault tolerance
- Records the number of replicas (replication factor) of a file specified by an application



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The NameNode stores modifications to the file system as a log appended to a native file system file named `edits`. When a NameNode starts up, it reads HDFS state from an image file named `fsimage`, and then applies the edits from the `edits` log file. Next, it writes the new HDFS state to the `fsimage` file, and then starts normal operation with an empty `edits` file. Because NameNode merges the `fsimage` and `edits` files only during startup, the `edits` log file could get very large over time on a busy cluster. Another side effect of a larger `edits` file is that the next restart of the NameNode will take longer time.

Secondary NameNode (Non-HA)



Secondary NameNode:

- The Secondary NameNode acts as a backup of NameNode metadata that can be imported (if necessary) to the primary NameNode.
- Performs checkpointing to the NameNode
- Runs on a different machine
- Has a directory structure identical to the NameNode directory
- Is not an immediate standby for the NameNode because it connects to the NameNode every hour

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

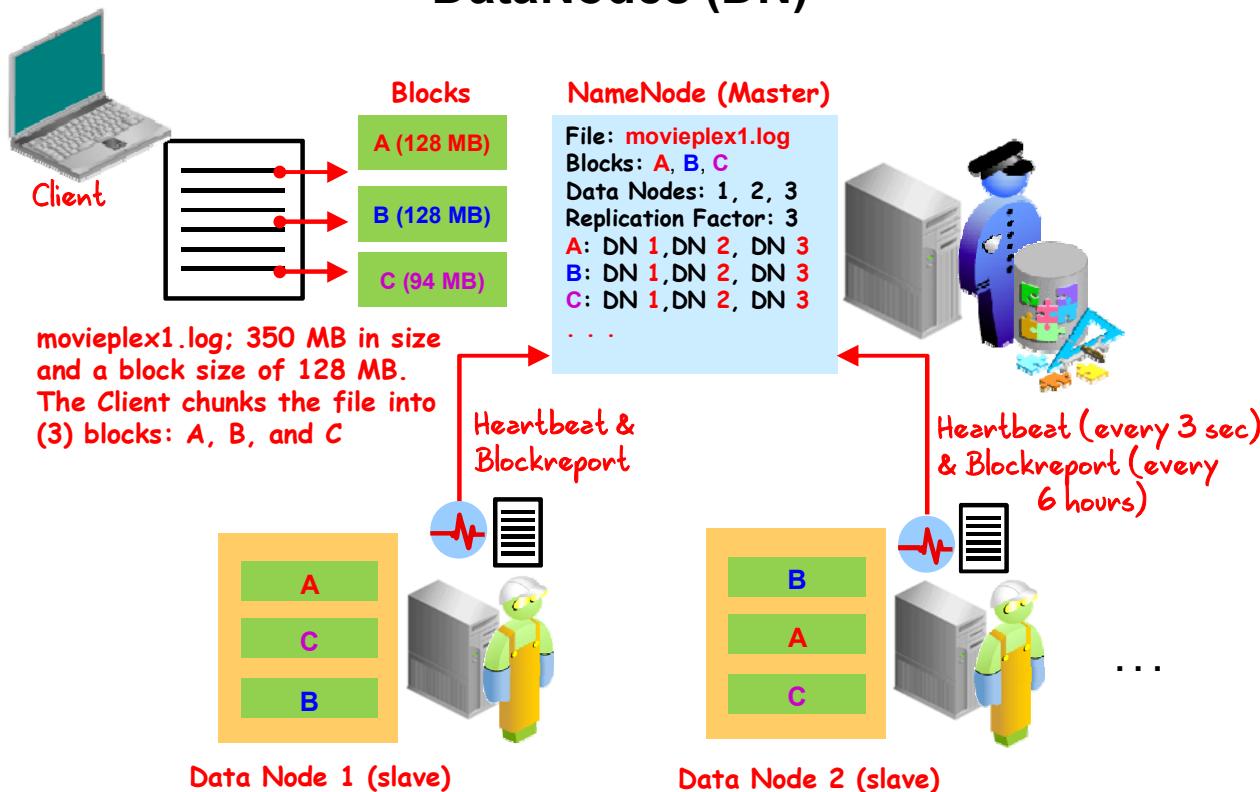
The Secondary NameNode does not provide failover capability. The HDFS High Availability (HA) feature in Hadoop provides the failover capability as discussed in the later slides in this lesson.

The secondary NameNode merges the `fsimage` and the `edits` log files periodically and keeps the `edits` log size within a limit. It is usually run on a different machine than the primary NameNode because its memory requirements are on the same order as the primary NameNode. The secondary NameNode stores the latest checkpoint in a directory, which is structured the same way as the primary NameNode's directory so that the check `fsimage` file is always ready to be read by the primary NameNode if necessary.

The Checkpoint Node Daemon performs periodic checkpoints of the namespace and helps minimize the size of the log stored at the NameNode containing changes to the HDFS. It replaces the role previously filled by the Secondary NameNode.

The Backup Node Daemon performs checkpointing and also receives the edits from the NameNode and maintains its own in-memory copy of the namespace, which is always in sync with the Active NameNode namespace state.

DataNodes (DN)



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

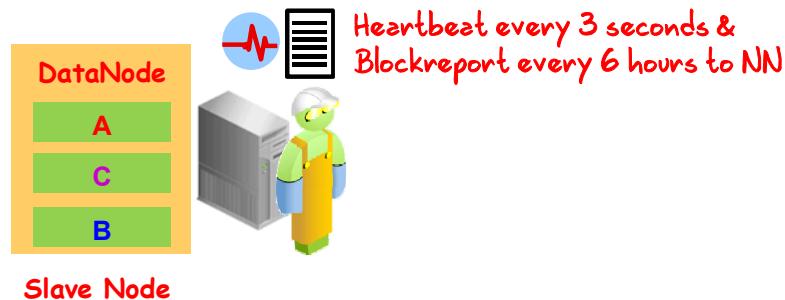
Data nodes report their status to the NameNode by using **heartbeat** messages and **Blockreport** to detect and ensure connectivity between the NameNode and the DataNodes. Receipt of a Heartbeat implies that the DataNode is functioning properly. A Blockreport contains a list of all blocks that are available on a DataNode. When a heartbeat is no longer present, the NameNode assumes that the DataNode is no longer functional and it unmaps the DataNode from the cluster and keeps on operating normally. When the heartbeat for the DataNode returns (or a new heartbeat appears), it is added to the cluster transparently without the user's intervention.

DataNodes use local disks in the commodity server for persistence. All the data blocks are stored locally, which improves performance. As mentioned earlier, the data blocks are replicated across several DataNodes. The default replication factor is 3 but that number is configurable. If one of the servers that contain one of the file's block fails, the file in question might not be corrupted. You can choose the degree of replication and the number of the DataNodes in the cluster when you implement the cluster. HDFS is dynamic in nature; therefore, all of the parameters can be adjusted during the operation of the cluster.

Functions of DataNodes

DataNodes perform the following functions:

- Serving read and write requests from the file system clients
- Performing block creation, deletion, and replication based on instructions from the NameNode
- Providing simultaneous send/receive operations to DataNodes during replication (“replication pipelining”)



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

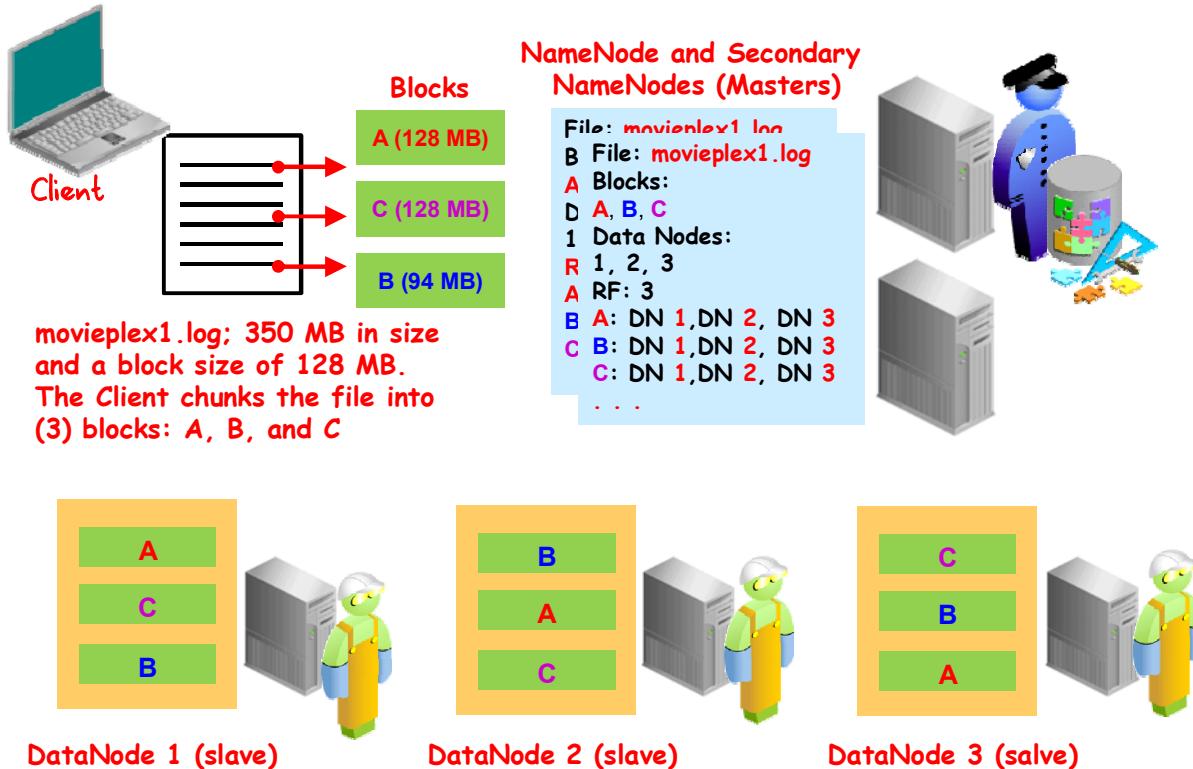
HDFS uses a master/slave architecture, with a single NameNode (master) managing multiple DataNodes (slaves). A typical HDFS deployment consists of a dedicated machine that runs only the NameNode software.

HDFS:

- Manages the file system namespace and regulates access to files from clients
- Executes file system namespace operations (open/close/rename files and directories)
- Splits data into blocks (also known as “chunks”) for storage among DataNodes. The block size is configurable (it is typically 64 MB).
- Replicates data to multiple machines/racks

Note: The Oracle BDA does not use a dedicated machine for the NameNode software. In addition, the HDFS block size on the BDA is 256 MB.

NameNode and Secondary NameNodes

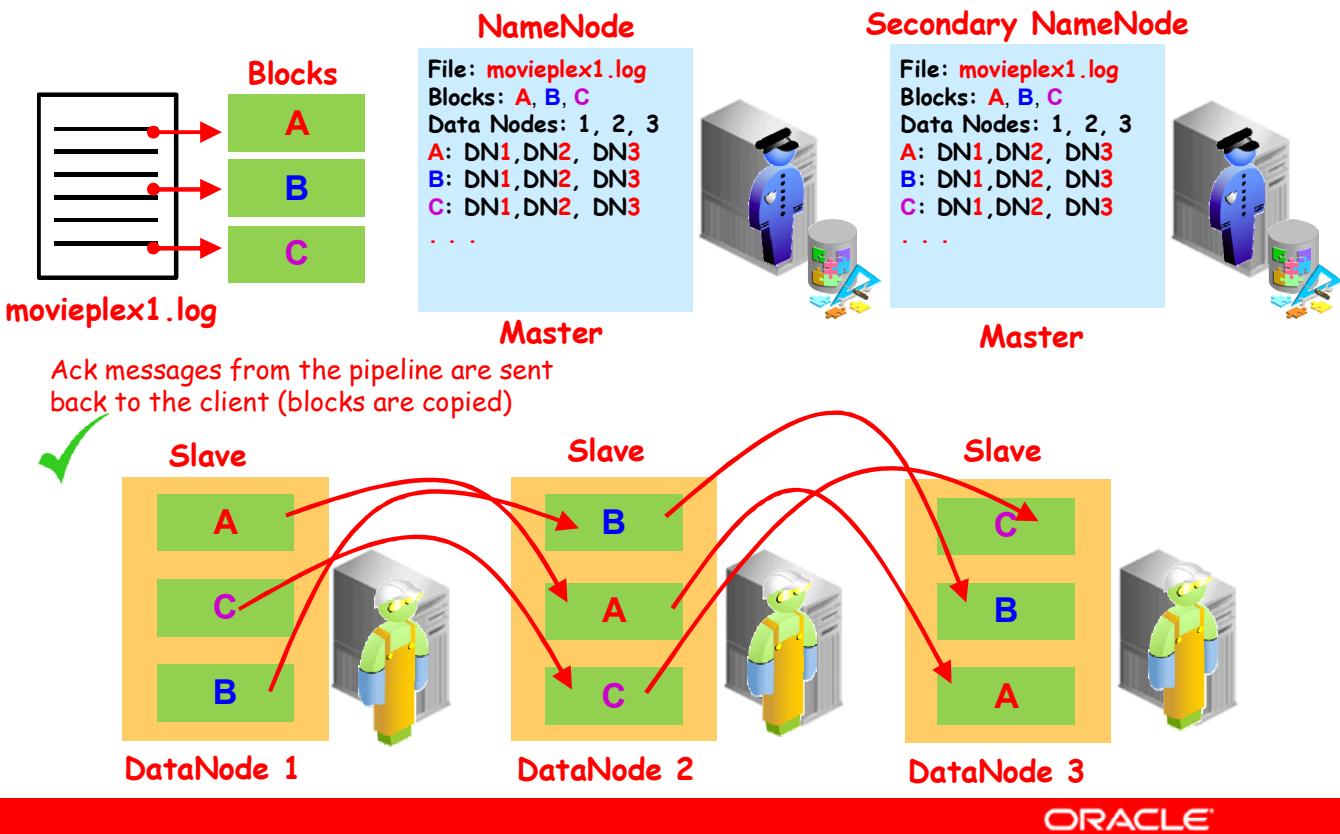


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

HDFS contains several DataNodes. The data that is stored in HDFS is divided into "chunks" or blocks and distributed across the DataNodes. The data blocks are also replicated to increase availability.

Storing and Accessing Data Files in HDFS



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HDFS supports the capability to create data pipelines. A data pipeline is a connection between multiple DataNodes that exists to support the movement of data across the servers in the Hadoop cluster. A client application writes a block to the first DataNode in the pipeline. That DataNode takes over and forwards the data to the next node in the pipeline. This process is continued until all replicas are written to disk. At this point, the client repeats the process by writing the next block in the file to the first DataNode.

When a client is writing data to an HDFS file, its data is first written to a local file. Suppose the HDFS file has a replication factor of three. When the local file accumulates a full block of user data, the client retrieves a list of DataNodes from the NameNode. This list contains the DataNodes that will host a replica of that block. The client then flushes the data block to the first DataNode. The first DataNode starts receiving the data in small portions, writes each portion to its local repository, and transfers that portion to the second DataNode in the list. The second DataNode in turn starts receiving each portion of the data block, writes that portion to its repository, and then flushes that portion to the third DataNode. Finally, the third DataNode writes the data to its local repository. Thus, a DataNode can be receiving data from the previous one in the pipeline and at the same time forwarding data to the next one in the pipeline. Thus, the data is pipelined from one DataNode to the next.

Secondary NameNode, Checkpoint Node, and Backup Node

Node Type	Description
Secondary NameNode	Performs periodic checkpoints (housekeeping tasks) of the namespace and helps keep the size of file containing log file, <code>edits</code> , of HDFS modifications within certain limits at the NameNode. This is not a backup NameNode.
Checkpoint Node	Performs periodic checkpoints of the namespace and helps minimize the size of the log stored at the NameNode containing changes to the HDFS. Replaces the role previously filled by the Secondary NameNode. This is optional.
Backup Node	An extension to the Checkpoint node. In addition to checkpointing it also receives a stream of edits from the NameNode and maintains its own in-memory copy of the namespace, which is always in sync with the active NameNode namespace state. This is optional.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

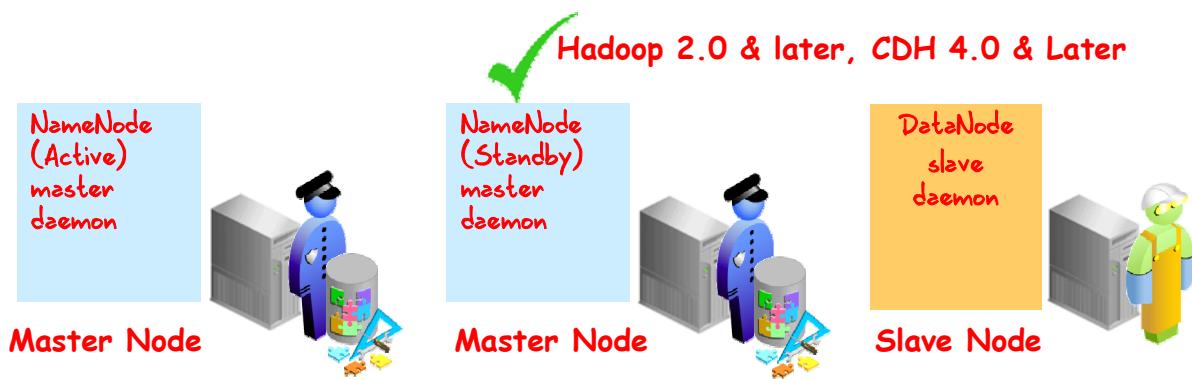
The secondary NameNode merges the `fsimage` and the `edits` log files periodically and keeps `edits` log size within a limit. It is usually run on a different machine than the primary NameNode because its memory requirements are on the same order as the primary NameNode. The secondary NameNode stores the latest checkpoint in a directory, which is structured the same way as the primary NameNode's directory so that the check `fsimage` file is always ready to be read by the primary NameNode if necessary.

The Checkpoint Node Daemon performs periodic checkpoints of the namespace and helps minimize the size of the log stored at the NameNode containing changes to the HDFS. It replaces the role previously filled by the Secondary NameNode.

The Backup Node Daemon performs checkpointing and also receives the edits from the NameNode and maintains its own in-memory copy of the namespace, which is always in sync with the Active NameNode namespace state.

HDFS Architecture: HA

Component	Description
NameNode (Active) Daemon	Responsible for all client operations in the cluster
NameNode (Standby) Daemon	Acts as a slave or "hot" backup to the Active NameNode, maintaining enough information to provide a fast failover if necessary
DataNode Daemon	This is where the data is stored (HDFS) and processed (MapReduce). This is a slave node.



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HDFS High Availability (HA) Using the Quorum Journal Manager (QJM)

- Prior to Hadoop 2.0.0, the NameNode was a single point of failure (SPOF) in an HDFS cluster.
- Each cluster had a single NameNode.
- The cluster is unavailable when the NameNode machine crashes or during software and hardware maintenance.
- HDFS HA addresses this problem by:
 - Running two redundant NameNodes in the same cluster: an **Active** NameNode and a **Hot Standby** NameNode
- HA provides fast failover to a new NameNode when the NameNode machine crashes or during regular software and hardware maintenance.
- **Oracle Big Data Appliance (BDA) uses the HA implementation.**



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

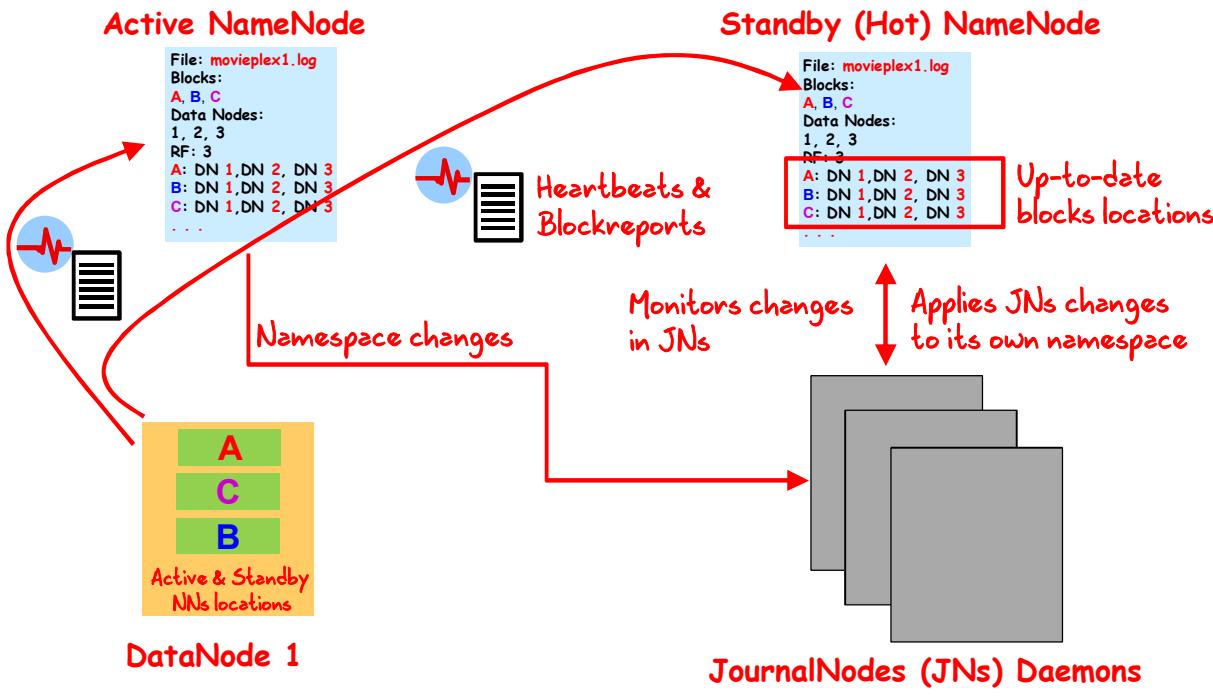
Prior to Hadoop 2.0.0, the NameNode was a single point of failure (SPOF) in an HDFS cluster. Each cluster had a single NameNode, and if that machine or process became unavailable, the cluster as a whole would be unavailable until the NameNode was either restarted or brought up on a separate machine.

This impacted the total availability of the HDFS cluster in two major ways:

- If a machine crashed, the cluster would be unavailable until an operator restarted the NameNode.
- Maintenance operations such as software or hardware upgrades on the NameNode machine would result in the cluster being down.

The HDFS High Availability feature addresses the preceding problems by providing the option of running two redundant NameNodes in the same cluster in an **Active** and **Passive** configuration with a **hot standby**. This allows a fast failover to a new NameNode in the case that a machine crashes, or a graceful administrator-initiated failover for the purpose of planned maintenance.

HDFS High Availability (HA) Using the Quorum Journal Manager (QJM) Feature



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

In a typical HA cluster, two separate machines are configured as NameNodes. At any point in time, exactly one of the NameNodes is in an **Active** state, and the other is in a **Standby** state. The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough information to provide a fast failover if necessary.

The Standby NameNode maintains its state synchronized with the Active NameNode by communicating with a group of separate daemons named **JournalNodes** (JNs). When the Active NameNode performs any namespace modification, it logs a record of the modification to most of the JNs. The Standby NameNode can read the edits from the JNs and is constantly monitoring changes in the edits file. If the Standby NameNode sees the edits, it applies them to its own namespace. In the event of failover in the Active NameNode, the Standby ensures that it has read all of the edits from the JNs before promoting itself to the Active NameNode state. This ensures that the namespace state is fully synchronized before a failover occurs.

The Standby node has up-to-date information about the blocks location in the Hadoop cluster. This provides a fast failover. The DataNodes are configured with the location of both NameNodes, and send **block location** (block reports) information and **heartbeats** to both.

In an HA cluster, only one of the NameNodes can be Active at a time; therefore, the JNs allow only a single NameNode to be a writer at a time. During a failover, the NameNode that is to become active will simply take over the role of writing to the JNs. This prevents the other NameNode from continuing in the Active state, allowing the new Active to safely proceed with failover. Cloudera Manager 5 and CDH 5 supports Quorum-based storage as the only HA implementation.

HDFS High Availability (HA) Using the Quorum Journal Manager (QJM) Feature

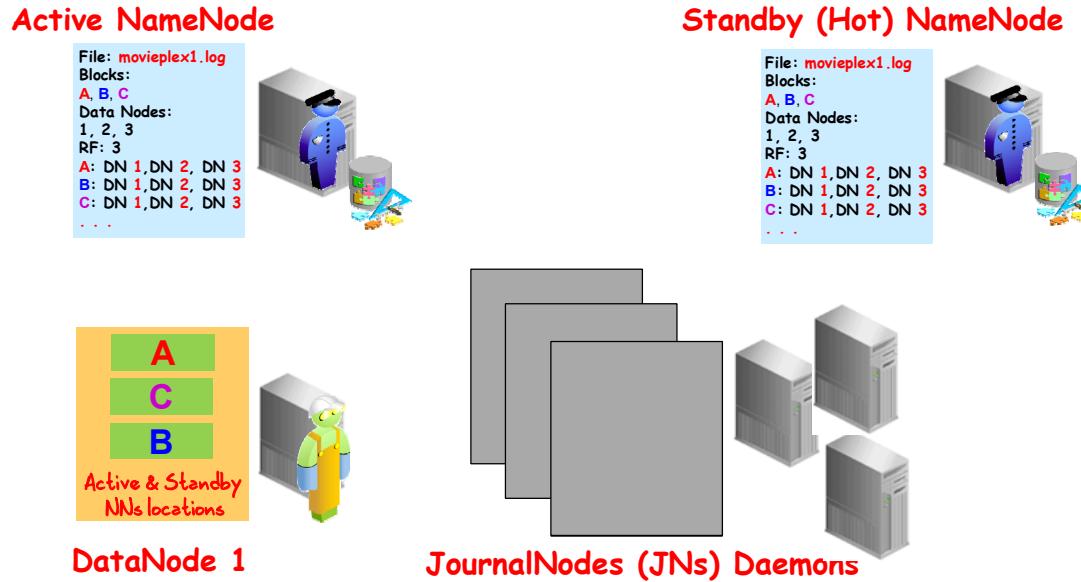
- The Standby NameNode in an HA cluster also performs checkpoints of the namespace state. Therefore, it is not necessary to run the following in the cluster:
 - A Secondary NameNode
 - A CheckpointNode
 - A BackupNode
- If you are reconfiguring a non-HA-enabled HDFS cluster to be HA-enabled, you can reuse the Secondary NameNode.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Configuring an HA Cluster Hardware Resources

To deploy an HA cluster using Quorum-based Storage, you should prepare the following Machines:



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To deploy an HA cluster, you should prepare the following:

1. The NameNode machines on which you will run the **Active** and **Standby NameNodes**.
2. The JournalNode machines on which you will run the **JournalNodes**. The JournalNode daemon is relatively lightweight, so you can run them on the Master hosts machines with other Hadoop daemons such as the NameNodes, the JobTracker, or the YARN ResourceManager. You must have at least three JournalNodes (but can run more than three JNs) daemons because the `edit log` file modifications must be written to a majority of JNs. This will allow the system to tolerate the failure of a single machine. You may also run more than three JournalNodes, but to actually increase the number of failures the system can tolerate, you should run an odd number of JNs (that is, 3, 5, 7, and so on). Note that when running with N JournalNodes, the system can tolerate at most $(N - 1) / 2$ failures and continue to function normally.

Enabling HDFS HA

- Using Cloudera Manager
 - Enable HA and Automatic Failover
- Using the Command Line Interface to configure automatic failover. Automatic failover adds the following components to an HDFS deployment:
 - A ZooKeeper quorum, which provides:
 - Failure Detection
 - Active NameNode Election
 - ZKFailoverController process (ZKFC), which provides:
 - Health Monitoring
 - ZooKeeper Session Management
 - ZooKeeper-based election



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Zookeeper is used to monitor the Active NameNode and to handle the failover logistics if the Active NameNode becomes unavailable.

Both the Active and Standby NameNodes have dedicated Zookeeper Failover Controllers (ZFC) that perform the monitoring and failover tasks. In the event of a failure, the ZFC informs the Zookeeper instances on the cluster, which then elect a new Active NameNode.

This course does not cover the details on how to enable HA automatic failover by using Cloudera Manager or details on how to use the command-line interface to configure automatic failover.

Note

- For details on how to enable HA and automatic failover by using Cloudera Manager, see the following Cloudera website:
http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_ha_g_hdfs_ha_enabling.html#cmug_topic_5_12_7_unique_2
- For detailed information on how to use the command-line interface to configure automatic failover, see the following Apache website:
<http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html>

Automatic failover adds two new components to an HDFS deployment: a ZooKeeper quorum and the ZKFailoverController (ZKFC) process.

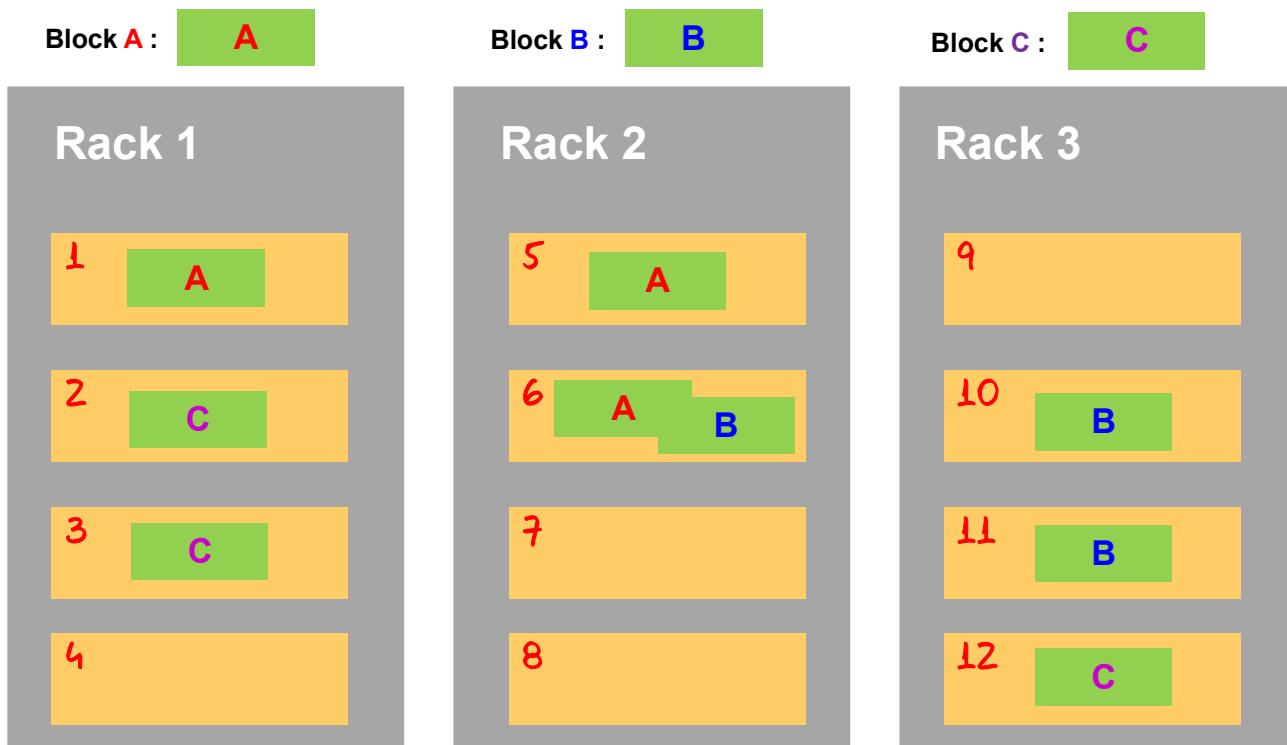
ZooKeeper provides the following tasks when implementing automatic HDFS failover:

- **Failure detection:** The Active and Standby NameNodes machines in the cluster maintain a persistent session in ZooKeeper. If the machine crashes, the ZooKeeper session will expire, notifying the other NameNode that a failover should be triggered.
- **Active NameNode election:** ZooKeeper provides a simple mechanism to exclusively elect a node as the Active NameNode. If the current Active NameNode crashes, another node may take a special exclusive lock in ZooKeeper indicating that it should become the next Active NameNode.

The ZKFailoverController (ZKFC) is a new component, which is a ZooKeeper client that also monitors and manages the state of the NameNode. Each of the machines that runs a NameNode also runs a ZKFC, and that ZKFC is responsible for:

- **Health monitoring:** ZKFC pings its local NameNode on a periodic basis with a health-check command. If NameNode responds in a timely fashion with a healthy status, the ZKFC considers the node healthy. If the node has crashed, frozen, or otherwise entered an unhealthy state, the health monitor will mark it as unhealthy.
- **ZooKeeper session management:** When the local NameNode is healthy, the ZKFC holds a session open in ZooKeeper. If the local NameNode is Active, it also holds a special "lock" znode. This lock uses ZooKeeper's support for "ephemeral" nodes; if the session expires, the lock node will be automatically deleted.
- **ZooKeeper-based election:** If the local NameNode is healthy, and the ZKFC sees that no other node currently holds the lock znode, it will itself try to acquire the lock. If it succeeds, then it has "won the election," and is responsible for running a failover to make its local NameNode active.

Data Replication Rack-Awareness in HDFS



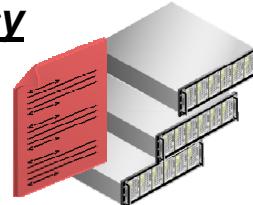
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Data Replication Process

The number of file replicas that will be maintained by HDFS (the “replication factor”) is stored in the NameNode.

- If the factor is (3), the **HDFS Placement Policy** directs replication as follows:
 - One copy on one node in a local rack
 - One copy on a different remote rack
 - One copy on a different node in the same remote rack
- This policy improves the write performance and ensures data reliability and availability.
- *If the reader process requires data, HDFS makes sure that it pulls the nearest replica for the task, thereby reducing the read latency (data locality).*



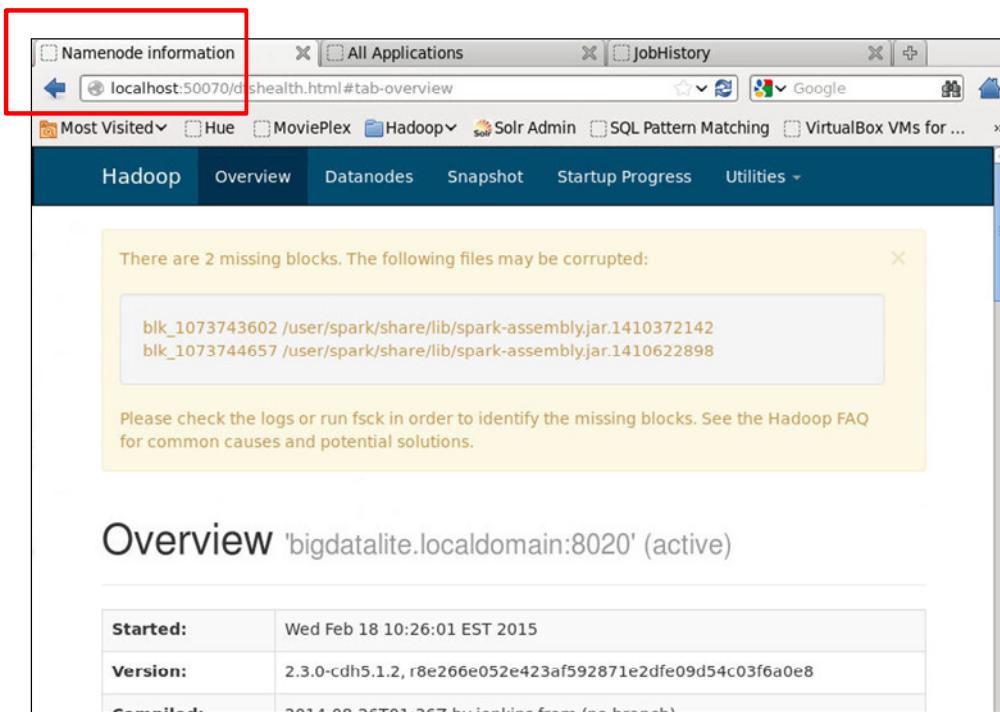
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The NameNode uses the Hadoop Rack Awareness process to determine the rack ID to which each DataNode belongs. A simple but nonoptimal policy is to place replicas on unique racks. This prevents losing data when an entire rack fails and allows use of bandwidth from multiple racks when reading data. This policy evenly distributes replicas in the cluster, which makes it easy to balance load on component failure. However, this policy increases the cost of writes because a write needs to transfer blocks to multiple racks.

For the common case, when the replication factor is three, HDFS’s placement policy is to put one replica on one node in the local rack, another on a node in a different (remote) rack, and the last on a different node in the same remote rack. This policy reduces the inter-rack write traffic, which generally improves write performance. The chance of rack failure is far less than that of node failure; this policy does not impact data reliability and availability guarantees. However, it does reduce the aggregate network bandwidth used when reading data because a block is placed in only two unique racks rather than three. With this policy, the replicas of a file do not evenly distribute across the racks. One-third of replicas are on one node, two-thirds of replicas are on one rack, and the other third are evenly distributed across the remaining racks. This policy improves write performance without compromising data reliability or read performance.

Accessing HDFS



The screenshot shows a web browser window with the following details:

- Address Bar:** localhost:50070/dfshealth.html#tab-overview
- Tab Bar:** Namenode information, All Applications, JobHistory
- Header:** Most Visited ▾, Hue, MoviePlex, Hadoop ▾, Solr Admin, SQL Pattern Matching, VirtualBox VMs for ...
- Navigation Bar:** Hadoop, Overview, Datanodes, Snapshot, Startup Progress, Utilities ▾
- Content Area:**
 - A yellow box displays a warning: "There are 2 missing blocks. The following files may be corrupted:"
 - File names listed: blk_1073743602 /user/spark/share/lib/spark-assembly.jar.1410372142, blk_1073744657 /user/spark/share/lib/spark-assembly.jar.1410622898
 - A note below says: "Please check the logs or run fsck in order to identify the missing blocks. See the Hadoop FAQ for common causes and potential solutions."
- Overview Section:** 'bigdatalite.localdomain:8020' (active)
- Table:**

Started:	Wed Feb 18 10:26:01 EST 2015
Version:	2.3.0-cdh5.1.2, r8e266e052e423af592871e2dfe09d54c03f6a0e8
Committed:	2014-09-26T01:26:7 by binonline from (no branch)



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Agenda

- Understand the architectural components of HDFS
- Use the **FS shell** command-line interface (CLI) to interact with data stored in HDFS



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HDFS Commands

The screenshot shows the 'User Commands' and 'Administration Commands' sections highlighted with red boxes. The 'User Commands' section includes: archive, distcp, fs, fsck, fetchdt, jar, job, pipes, queue, version, CLASSNAME, and classpath. The 'Administration Commands' section includes: balancer, daemonlog, datanode, dfsadmin, mradmin, jobtracker, namenode, secondarynamenode, and tasktracker.

- Overview
- Generic Options
- **User Commands**
 - archive
 - distcp
 - fs
 - fsck
 - fetchdt
 - jar
 - job
 - pipes
 - queue
 - version
 - CLASSNAME
 - classpath
- **Administration Commands**
 - balancer
 - daemonlog
 - datanode
 - dfsadmin
 - mradmin
 - jobtracker
 - namenode
 - secondarynamenode
 - tasktracker

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can access the Hadoop Command Reference by using the following url:

http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/CommandsManual.html#Administration_Commands

The File System Namespace: The HDFS FS (File System) Shell Interface

- HDFS supports a traditional hierarchical file organization.
- You can use the **FS shell** command-line interface to interact with the data in HDFS. The syntax of this command set is similar to other shells (e.g., bash, csh)
 - You can create, remove, rename, and move directories/files.
- You can invoke the FS shell as follows:

```
hadoop fs <args>
```

- The general command-line syntax is as follows:

```
hadoop command [genericOptions] [commandOptions]
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories. The file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from one directory to another, or rename a file. HDFS does not yet implement user quotas. HDFS does not support hard links or soft links. However, the HDFS architecture does not preclude implementing these features.

HDFS organizes data in the form of files and directories. It provides a command-line interface named FS shell that enables you to interact with the data stored in HDFS. The syntax of this command set is similar to other shells such as bash and csh.

The File System (FS) shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS). You can invoke the FS shell as follows:

```
hadoop fs <args>
```

For information about the FS shell commands, see the FileSystem shell site at:

<http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html>

The File System Namespace: The HDFS FS (File System) Shell Interface

```
hadoop fs -help
```

```
[oracle@bigdatalite ~]$ hadoop fs -help
Usage: hadoop fs [generic options]
  [-appendToFile <localsrc> ... <dst>]
  [-cat [-ignoreCrc] <src> ...]
  [-checksum <src> ...]
  [-chgrp [-R] GROUP PATH...]
  [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
  [-chown [-R] [OWNER][:[GROUP]] PATH...]
  [-copyFromLocal [-f] [-p] <localsrc> ... <dst>]
  [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
  [-count [-q] <path> ...]
  [-cp [-f] [-p] <src> ... <dst>]
  [-createSnapshot <snapshotDir> [<snapshotName>]]
  [-deleteSnapshot <snapshotDir> <snapshotName>]
  [-df [-h] [<path> ...]]
  [-du [-s] [-h] <path> ...]
  [-expunge]
  [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
  [-getfacl [-R] <path>]
  [-getmerge [-nL] <src> <localdst>]
  [-help [cmd ...]]
  [-ls [-d] [-h] [-R] [<path> ...]]
  [-mkdir [-p] <path> ...]
  [-moveFromLocal <localsrc> ... <dst>]
  [-moveToLocal <src> <localdst>]
  [-mv <src> ... <dst>]
  [-put [-f] [-p] <localsrc> ... <dst>]
  [-renameSnapshot <snapshotDir> <oldName> <newName>]
  [-rm [-f] [-r|-R] [-skipTrash] <src> ...]
  [-rmdir [-ignore-fail-on-non-empty] <dir> ...]
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories. The file system namespace hierarchy is similar to most other existing file systems; one can create and remove files, move a file from one directory to another, or rename a file. HDFS does not yet implement user quotas. HDFS does not support hard links or soft links. However, the HDFS architecture does not preclude implementing these features.

HDFS organizes data in the form of files and directories. It provides a command-line interface named FS shell that enables you to interact with the data stored in HDFS. The syntax of this command set is similar to other shells such as bash and csh.

For information about the FS shell commands, see the FileSystem shell site at:

<http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html>

Accessing HDFS

The screenshot shows a terminal window with the following content:

```
[oracle@bigdatalite ~]$ hadoop fs
Usage: hadoop fs [generic options]
[-appendToFile <localsrc> ... <dst>]
[-cat [-ignoreCrc] <src> ...]
[-checksum <src> ...]
[-chgrp [-R] GROUP PATH...]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-copyFromLocal [-f] [-p] <localsrc> ... <dst>]
[-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
[-count [-q] <path> ...]
[-cp [-f] [-p] <src> ... <dst>]
[-createSnapshot <snapshotDir> [<snapshotName>]]
[-deleteSnapshot <snapshotDir> <snapshotName>]
[-df [-h] [<path> ...]]
[-du [-s] [-h] <path> ...]
[-expunge]
[-get [-p] [-ig]
[-getfacl [-R]
[-getmerge [-nl]
[-help [cmd ...
[-ls [-d] [-h]
```

A tooltip box is overlaid on the terminal window, containing the following text:

Generic options supported are

- conf <configuration file> specify an application configuration file
- D <property=value> use value for given property
- fs <local|namenode:port> specify a namenode
- jt <local|jobtracker:port> specify a job tracker
- files <comma separated list of files> specify comma separated files to be copied to the map reduce cluster
- libjars <comma separated list of jars> specify comma separated jar files to include in the classpath.
- archives <comma separated list of archives> specify comma separated archives to be unarchived on the compute machines.

The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Using the Oracle Linux command-line terminal, you can access HDFS and load data from the log files.

HDFS commands are similar to Linux commands.

Note: The prefix `hadoop` must be added to all commands as shown in the first screenshot in the slide.

FS Shell Commands

Hadoop Shell Commands

- **FS Shell**
 - [cat](#)
 - [chgrp](#)
 - [chmod](#)
 - [chown](#)
 - [copyFromLocal](#)
 - [copyToLocal](#)
 - [cp](#)
 - [du](#)
 - [dus](#)
 - [expunge](#)
 - [get](#)
 - [getmerge](#)
 - [ls](#)
 - [lsr](#)
 - [mkdir](#)
 - [movefromLocal](#)
 - [mv](#)
 - [put](#)
 - [rm](#)
 - [rmr](#)
 - [setrep](#)

```
[oracle@bigdatalite ~]$ ls -l
total 56
drwxr-xr-x. 2 oracle oracle 4096 Sep 23 15:03 Desktop
drwxr-xr-x. 2 oracle oracle 4096 Sep 7 2013 Documents
drwxr-xr-x. 2 oracle oracle 4096 Sep 23 14:53 Downloads
drwxr-xr-x. 3 oracle oinstall 4096 Sep 28 23:13 GettingStarted
drwx----- 6 oracle oinstall 4096 Aug 18 08:38 movie
drwxr-xr-x. 2 oracle oracle 4096 Sep 7 2013 Music
drwxr-x--- 3 oracle oinstall 4096 Sep 3 13:20 oradiag_oracle
drwxr-x--- 6 oracle oinstall 4096 Jan 25 2014 Pattern_Matching_WS
drwxr-xr-x. 2 oracle oracle 4096 Jan 12 2014 Pictures
drwxr-xr-x. 2 oracle oracle 4096 Sep 7 2013 Public
drwxr-xr-x. 3 oracle oracle 4096 Jan 7 2014 R
drwxr-xr-x. 3 oracle oinstall 4096 Sep 24 09:41 scripts
drwxr-xr-x. 2 oracle oracle 4096 Sep 7 2013 Templates
drwxr-xr-x. 2 oracle oracle 4096 Sep 7 2013 Videos
[oracle@bigdatalite ~]$ hadoop fs -ls
Found 6 items
drwx----- - oracle oracle 0 2014-08-25 05:55 .Trash
drwx----- - oracle oracle 0 2014-09-23 13:25 .staging
drwxr-xr-x - oracle oracle 0 2014-01-12 18:15 moviedemo
drwxr-xr-x - oracle oracle 0 2014-09-24 09:38 moviework
drwxr-xr-x - oracle oracle 0 2014-09-08 15:50 oggdemo
drwxr-xr-x - oracle oracle 0 2014-09-20 13:59 oozie-oozi
```

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The first screenshot in the slide titled Hadoop Shell Commands shows the FS shell commands, which are available from the FileSystem shell site at:

<http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html>

The second screenshot in the slide shows using the `ls` command on both the local file system and on the hadoop namespace, which returns different results.

Basic File System Operations: Examples

```
hadoop fs -ls
```

```
oracle@bigdatalite:~$ hadoop fs -ls wordcount
Found 2 items
drwxr-xr-x  - oracle oracle      0 2015-03-10 02:45 wordcount/input
drwxr-xr-x  - oracle oracle      0 2015-03-10 04:09 wordcount/output
[oracle@bigdatalite ~]$ hadoop fs -ls wordcount/input
Found 2 items
-rw-r--r--  1 oracle oracle    518 2015-03-10 02:45 wordcount/input/file01
-rw-r--r--  1 oracle oracle    518 2015-03-10 02:45 wordcount/input/file02
[oracle@bigdatalite ~]$
```

- For a file returns stat on the file with the following format:
 - permissions number_of_replicas userid groupid filesize modification_date modification_time filename
- For a directory it returns list of its direct children as in UNIX. A directory is listed as:
 - permissions userid groupid modification_date modification_time dirname

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the two screen captures examples shown in the slide, the `ls` command is used to get an HDFS file listing for the HDFS `wordcount` and `wordcount/input` directories.

Column 1 shows the file mode: "d" for a directory and a "-" for a normal file, followed by the file or directory permission. HDFS has a permissions model similar to that used in Linux. The three permission types are: read (r), write (w), and execute (x). The execute permission for a file is ignored because you cannot execute a file on HDFS. The permissions are grouped by owner, group, and public (everyone else). The second column shows the replication factor for files. The concept of replication factor does not apply to directories. Columns 3 and 4 show the file owner and group. Column 5 shows the size of the file, in bytes, or 0 if it is a directory. Columns 6 and 7 show the date and time of the last modification, respectively. Column 8 is the name of the file or directory.

Sample FS Shell Commands

Command	Description
<code>cat</code>	Copies source paths to stdout.
<code>copyFromLocal</code>	Copies a file from the local file system to HDFS. <i>This command is similar to get command, except that the destination is restricted to a local file reference.</i>
<code>copyToLocal</code>	Copies a file from HDFS to the local file system. <i>This command is similar to get command, except that the destination is restricted to a local file reference.</i>
<code>cp</code>	Copy files from source to destination in HDFS
<code>get</code>	Copies files from HDFS to the local file system
<code>jar</code>	Runs a jar file. Users can bundle their Map Reduce code in a jar file and execute it using this command.
<code>mkdir</code>	Creates one or more HDFS directories
<code>mv</code>	Moves files from source to destination. Moving files across file systems is not permitted.
<code>put</code>	Copies files from the local file system to HDFS
<code>Rm</code>	Deletes files specified. The <code>-r</code> option deletes the directory and its contents.
<code>version</code>	Prints the Hadoop version



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Basic File System Operations: Examples

Create an HDFS directory named `curriculum` by using the `mkdir` command:

```
[oracle@bigdatalite ~]$ hadoop fs -mkdir curriculum
[oracle@bigdatalite ~]$ hadoop fs -ls
Found 9 items
drwx-----  - oracle oracle          0 2014-08-25 05:55 .Trash
drwx-----  - oracle oracle          0 2015-03-10 04:09 staging
drwxr-xr-x  - oracle oracle          0 2015-03-24 09:38 curriculum
drwxr-xr-x  - oracle oracle          0 2014-01-12 18:15 moviedemo
drwxr-xr-x  - oracle oracle          0 2014-09-24 09:38 moviework
drwxr-xr-x  - oracle oracle          0 2014-09-08 15:50 oggdemo
drwxr-xr-x  - oracle oracle          0 2014-09-20 13:59 oozie-oozi
drwxr-xr-x  - oracle oracle          0 2015-03-24 00:57 test
drwxr-xr-x  - oracle oracle          0 2015-03-10 04:09 wordcount
[oracle@bigdatalite ~]$
```

Copy `lab_05_01.txt` from the local file system to the `curriculum` HDFS directory by using the `copyFromLocal` command:

```
[oracle@bigdatalite ~]$ cd Practice_Commands
[oracle@bigdatalite Practice_Commands]$ ls
lab_05_01.txt lab_09_01.txt lab_13_01.txt lab_15_01.txt lab_19_02.txt lab_21_02.txt
lab_07_01.txt lab_11_01.txt lab_13_02.txt lab_18_01.txt lab_19_03.txt lab_23_01.txt
lab_07_02.txt lab_11_02.txt lab_13_03.txt lab_18_02.txt lab_20_01.txt lab_27_01.txt
lab_07_04.txt lab_11_03.txt lab_14_01.txt lab_19_01.txt lab_21_01.txt
[oracle@bigdatalite Practice_Commands]$ hadoop fs -copyFromLocal lab_05_01.txt curriculum/lab_05_01.txt
[oracle@bigdatalite Practice_Commands]$ hadoop fs -ls curriculum
Found 1 items
-rw-r--r--  1 oracle oracle      524 2015-03-24 10:14 curriculum/lab_05_01.txt
[oracle@bigdatalite Practice_Commands]$
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Basic File System Operations: Examples

Delete the `curriculum` HDFS directory by using the `rm` command. Use the `-r` option to delete the directory and any content under it recursively:

```
[oracle@bigdatalite Practice_Commands]$ hadoop fs -rm -r curriculum -- --  
15/03/24 10:31:01 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier interval  
= 0 minutes.  
Deleted curriculum
```

Display the contents of the `part-r-00000` HDFS file by using the `cat` command:

```
[oracle@bigdatalite ~]$ hadoop fs -cat /user/oracle/wordcount/output/part-r-00000  
and 12  
awful 2  
bank 2  
company 4  
cover 2  
customer 6  
disappointed 6  
expensive 12  
insurance 18  
is 2  
professional 2  
protocols 2  
service 12  
staff 2  
terrible 4  
the 2  
unreliable 6  
very 6  
with 4  
worst 16  
worthless 4  
[oracle@bigdatalite ~]$
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HDFS Administration Commands

Command	Description
<code>namenode -format</code>	Formats the DFS file system
<code>secondarynamenode</code>	Runs the DFS secondary NameNode
<code>namenode</code>	Runs the DFS NameNode
<code>datanode</code>	Runs a DFS DataNode
<code>balancer</code>	Runs a cluster-balancing utility
<code>fsck - /</code>	Runs a DFS file system checking utility
<code>jobtracker</code>	Runs the mapreduce job tracker node
<code>tasktracker</code>	Runs a mapreduce task tracker node



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The table in the slide shows some of the available administration commands that are useful for administrators of a Hadoop cluster.

For detailed information on using the commands shown in the slide, see the following URL:

http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/CommandsManual.html#Administration_Commands

Using the hdfs fsck Command: Example

```
[oracle@bigdatalite ~]$ hdfs fsck /user/oracle/wordcount/output/part-r-00000 -files -blocks
15/03/26 01:49:08 WARN ssl.FileBasedKeyStoresFactory: The property 'ssl.client.truststore.location' has not been set, no TrustStore will be loaded
Connecting to namenode via http://bigdatalite.localdomain:50070
FSCK started by oracle (auth:SIMPLE) from /127.0.0.1 for path /user/oracle/wordcount/output/part-r-00000 at Thu Mar 26 01:49:09 EDT 2015
/user/oracle/wordcount/output/part-r-00000 208 bytes, 1 block(s): OK
0. BP-703742109-127.0.0.1-1398459391664:blk_1073754500_13678 len=208 repl=1

Status: HEALTHY
Total size: 208 B
Total dirs: 0
Total files: 1
Total symlinks: 0
Total blocks (validated): 1 (avg. block size 208 B)
Minimally replicated blocks: 1 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks: 0 (0.0 %)
Mis-replicated blocks: 0 (0.0 %)
Default replication factor: 1
Average block replication: 1.0
Corrupt blocks: 0
Missing replicas: 0 (0.0 %)
Number of data-nodes: 1
Number of racks: 1
FSCK ended at Thu Mar 26 01:49:09 EDT 2015 in 0 milliseconds

The filesystem under path '/user/oracle/wordcount/output/part-r-00000' is HEALTHY
[oracle@bigdatalite ~]$
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

`fsck` can be run on the whole file system or on a subset of files.

The slide example shows how to use the `hdfs fsck` command on a single file to find out the blocks for that file in addition to some other useful information. In this course, you are using the BDLite VM, which is only one node; therefore, no files are being replicated.

HDFS Features and Benefits

HDFS provides the following features and benefits:

- A Rebalancer to evenly distribute data across the DataNodes
- A file system checking utility (`fck`) to perform health checks on the file system
- Procedures for upgrade and rollback
- A secondary NameNode to enable recovery and keep the edits log file size within a limit
- A Backup Node to keep an in-memory copy of the NameNode contents



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Describe the architectural components of HDFS
- Use the **FS shell** command-line interface (CLI) to interact with data stored in HDFS



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 5: Overview

In this practice, you load a JSON log file into HDFS. This log file was used to track activity in an online movie application.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

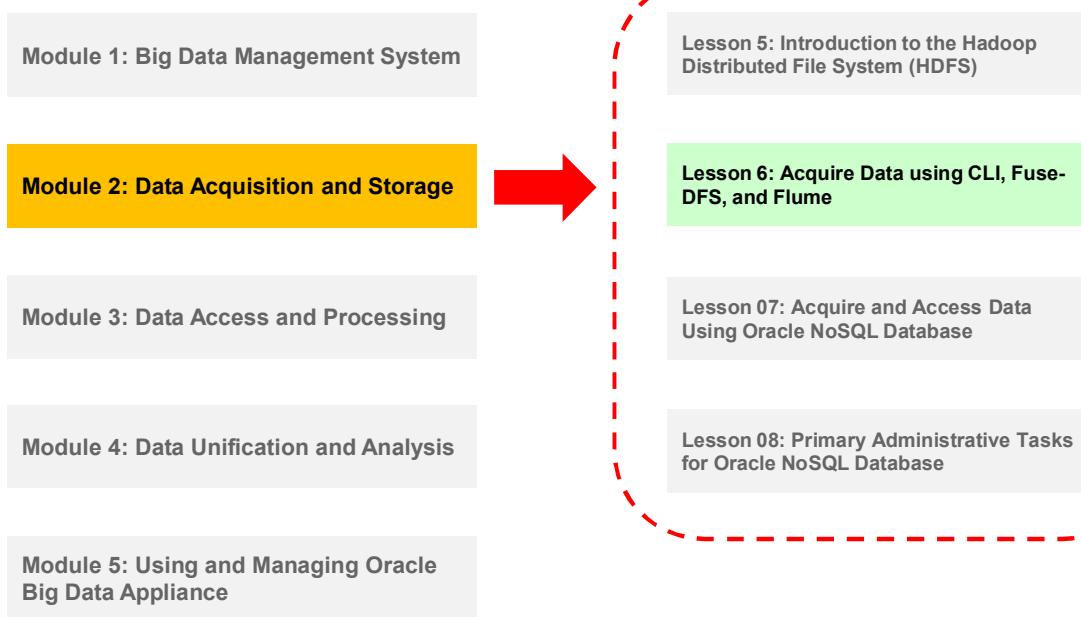
Acquire Data Using CLI, Fuse DFS, and Flume



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

As shown in the Course Road Map, the current module covers several technologies used for acquiring and storing big data.

In the last lesson, you were provided a high level introduction to the Hadoop Distributed File System. In this lesson, you learn more about big data acquisition techniques, including the Command Line Interface (CLI), Fuse-DFS, and Flume.

Objectives

After completing this lesson, you should be able to:

- Describe Uses of the Command Line Interface (CLI)
- Describe the benefits of Fuse DFS
- Define Flume
- Describe the data-flow mechanism of Flume
- Identify the options for configuring Flume



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Reviewing the Command Line Interface (CLI)

View usage and generic options for `hadoop fs`:

```
$ hadoop fs
```

The screenshot shows a terminal window titled "oracle@bigdatalite:~". The command `hadoop fs` was entered, and the output displayed is the usage information for the `hadoop fs` command. The usage text includes various options such as `-appendToFile`, `-cat`, `-checksum`, `-chgrp`, `-chmod`, `-chown`, `-copyFromLocal`, `-copyToLocal`, `-count`, `-cp`, `-createSnapshot`, `-deleteSnapshot`, `-df`, `-du`, `-expunge`, `-get`, `-getfacl`, `-getmerge`, `-help`, `-ls`, `-mkdir`, and `-moveFromLocal`. Each option is followed by its description and usage examples.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

As you learned in the previous lesson, the Command Line Interface (CLI) is used for interacting with big data sources, and is very similar to the Linux CLI. It enables you to interact with a variety of big data formats, including HDFS and NoSQL.

In this course, you will use `hadoop fs` commands in a variety of lessons. And, as you learned in the previous practice, these commands are primarily used for moving data in/out/around the environment, and for viewing data and file system objects.

Recall that typing `hadoop fs` at the command prompt provides usage and options for FS shell commands.

Viewing File System Contents Using the CLI

```
[oracle@bigdatalite ~]$ hadoop fs -ls /user/oracle/moviework/applog.json  
Found 1 items  
-rw-r--r-- 1 oracle oracle 32557883 2014-09-13 13:39 /user/oracle/moviework/applog.json/movieapp_log_json.log  
[oracle@bigdatalite ~]$ hadoop fs -tail /user/oracle/moviework/applog.json/movieapp_log_json.log  
,"recommended": "Y", "activity": 2}  
{"custid": 1135508, "movieid": 240, "genreid": 8, "time": "2012-10-01:02:04:15", "recommended": "Y", "activity": 5}  
{"custid": 1135508, "movieid": 1092, "genreid": 20, "time": "2012-10-01:02:10:23", "recommended": "N", "activity": 5}  
{"custid": 1135508, "movieid": 4638, "genreid": 8, "time": "2012-10-01:02:10:54", "recommended": "N", "activity": 7}  
{"custid": 1135508, "movieid": 4638, "genreid": 8, "time": "2012-10-01:02:16:49", "recommended": "N", "activity": 7}  
{"custid": 1135508, "movieid": null, "genreid": null, "time": "2012-10-01:02:24:00", "recommended": null, "activity": 9}  
{"custid": 1135508, "movieid": 240, "genreid": 8, "time": "2012-10-01:02:31:12", "recommended": "Y", "activity": 11, "price": 2.99}  
{"custid": 1191532, "movieid": 59440, "genreid": 7, "time": "2012-10-01:03:11:35", "recommended": "Y", "activity": 2}  
{"custid": 1191532, "movieid": null, "genreid": null, "time": "2012-10-01:03:15:29", "recommended": null, "activity": 9}  
{"custid": 1191532, "movieid": 59440, "genreid": 7, "time": "2012-10-01:03:19:24", "recommended": "Y", "activity": 11, "price": 3.99}
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the slide screenshot, two hadoop fs commands are executed:

- -ls is used to display the contents of the current directory.
- -tail is executed to display the contents of a specified file.

Loading Data Using the CLI

Put files into HDFS:

```
$ hadoop fs -put *site.xml /u01/bigdatasql_config/bigdatalite
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The `hadoop fs -put` command may be used to load data into HDFS.

In the slide example, all files with the naming pattern of `*site.xml` in the current location are put into an HDFS directory named `/u01/bigdatasql_config/bigdatalite`.

What is Fuse DFS?

Enables access to HDFS as if it were just a file system folder

1. Install hadoop-hdfs-fuse on your file system.
2. Set up and test your mount point (non-HA installation).

```
$ mkdir -p <mount_point>
$ hadoop-fuse-dfs dfs://<name_node_hostname>:<namenode_port> <mount_point>
```

3. Clean up the test.

```
$ umount <mount_point>
```

4. To add the system mount: open /etc/fstab and add lines to the bottom similar to these:

```
hadoop-fuse-dfs#dfs://<name_node_hostname>:<namenode_port> <mount_point>
fuse allow_other,usetrash,rw 2 0
```

Result: Your system is now configured to allow you to use the ls command and use that mount point as if it were a normal system disk.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

CDH 5 includes a FUSE (Filesystem in Userspace) interface into HDFS. The hadoop-hdfs-fuse package enables you to use your HDFS cluster as if it were a traditional file system on Linux.

To use Fuse DFS (also called NFS Mount), perform the following steps:

1. Install hadoop-hdfs-fuse on your file system.
2. Set up and test your mount point.
3. Clean up the test.
4. Add the system mount.

Note

- Fuse DFS may be enabled on the Big Data Lite VM. Use the following script to enable Fuse DFS on Big Data Lite: /opt/bin/fusedfs
- Before configuring Fuse DFS you must have a working HDFS cluster and know the host name and port that your NameNode exposes.
- To find its configuration directory, hadoop-fuse-dfs uses HADOOP_CONF_DIR configured at the time the mount command is invoked.

Enabling Fuse DFS on Big Data Lite

The screenshot shows a terminal window titled "oracle@bigdatalite:/opt/bin". The terminal displays the following command-line session:

```
[oracle@bigdatalite ~]$ cd /opt/bin/  
[oracle@bigdatalite bin]$ ls -l  
total 64  
-rwxrwxrwx. 1 oracle oinstall 105 Sep 20 2014 bee  
-rwxrwxrwx. 1 oracle oinstall 253 May 2 2014 fusedfs  
-rwxrwxrwx. 1 oracle oinstall 954 May 2 2014 nosqlDB  
-rwxrwxrwx. 1 oracle oinstall 74 May 2 2014 nosqlDB_cli  
-rwxrwxrwx. 1 oracle oinstall 337 Sep 17 2014 odi  
-rwxrwxrwx. 1 oracle oinstall 708 May 1 2014 oracledb  
-rwxrwxrwx. 1 oracle oinstall 517 May 1 2014 oraclelsnr  
-rwxrwxrwx. 1 oracle oracle 122 Sep 7 2013 reset_vboxclient.sh  
-rwxrwxrwx. 1 oracle oinstall 9012 Sep 12 2014 services  
-rwxr-xr-x. 1 root root 5885 Aug 12 2014 services_bck  
-rwxrwxrwx. 1 oracle oinstall 1040 Sep 10 2014 services.prop  
-rwxr-xr-x. 1 root root 938 Aug 12 2014 services.prop_bck  
-rwxrwxrwx. 1 oracle oinstall 273 Sep 23 2014 shrink.sh  
[oracle@bigdatalite bin]$ cat fusedfs  
echo Attempting to mount HDFS to /mnt/hdfs  
sudo hadoop-fuse-dfs dfs://localhost:8020 /mnt/hdfs  
echo Wait for mount to complete....  
sleep 5  
echo "Listing contents of hdfs folder /user/oracle (Linux folder /mnt/hdfs/user/oracle)"  
ls /mnt/hdfs/user/oracle  
[oracle@bigdatalite bin]$
```

A red arrow points to the "fusedfs" file in the ls -l output. A red dashed box highlights the entire command-line session from the "cat fusedfs" command to the final "ls /mnt/hdfs/user/oracle" command.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Oracle Big Data Lite VM includes a script that enables Fuse-DFS on the VM. The script is `/opt/bin/fusedfs`.

The screenshot shows:

- The location of the `fusedfs` script
- The contents of the `fusedfs` script, by using the linux `cat` command.

Notice that the `hadoop-fuse-dfs` command in the script sets the mount point for the local user's home directory `/home/oracle` to `/mnt/hdfs/user/oracle`.

Therefore, now you can view linux file system files and HDFS file system files using the same linux command syntax.

To run the script:

1. Change to the `/opt/bin/` directory.
2. At the linux prompt, execute the command `fusedfs`

Using Fuse DFS

```
[oracle@bigdatalite ~]$ ls -l
total 72
drwxr-xr-x. 2 oracle oinstall 4096 Feb 27 04:46 bigdatasql-hol
drwxr-xr-x. 2 oracle oracle 4096 Sep 23 2014 Desktop
drwxr-xr-x. 2 oracle oracle 4096 Sep 7 2013 Documents
drwxr-xr-x. 2 oracle oracle 4096 Sep 23 2014 Downloads
drwxr-xr-x. 16 oracle oinstall 4096 Feb 27 04:43 exercises
drwxr-xr-x. 3 oracle oinstall 4096 Sep 28 2014 GettingStarted
drwx----- 7 oracle oinstall 4096 Feb 27 04:56 movie
drwxr-xr-x. 2 oracle oracle 4096 Sep 7 2013 Music
drwxr-xr-x. 4 oracle oinstall 4096 Jan 26 14:54 orabalancerdemo-2.3.0-h2
drwxr-x---. 3 oracle oinstall 4096 Sep 3 2014 oradiag_oracle
drwxr-x---. 6 oracle oinstall 4096 Jan 25 2014 Pattern_Matching_WS
drwxr-xr-x. 2 oracle oracle 4096 Jan 12 2014 Pictures
drwxr-xr-x. 2 oracle oinstall 4096 Apr 3 08:31 practice_commands
drwxr-xr-x. 2 oracle oracle 4096 Sep 7 2013 Public
drwxr-xr-x. 3 oracle oracle 4096 Jan 7 2014 R
drwxr-xr-x. 3 oracle oinstall 4096 Sep 24 2014 scripts
drwxr-xr-x. 2 oracle oracle 4096 Sep 7 2013 Templates
drwxr-xr-x. 2 oracle oracle 4096 Sep 7 2013 Videos
[oracle@bigdatalite ~]$ ls -l /mnt/hdfs/user/oracle
total 10
drwxr-xr-x. 3 oracle oracle 4096 Jan 12 2014 moviedemo
drwxr-xr-x. 11 oracle oracle 4096 Sep 24 2014 moviework
drwxr-xr-x. 3 oracle oracle 4096 Sep 8 2014 oggdemo
drwxr-xr-x. 2 oracle oracle 4096 Sep 20 2014 oozie-oozi
[oracle@bigdatalite ~]$
```

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Once Fuse-DFS is enabled, it makes the command line look like linux. For example, `ls -l` will give you a directory listing, you can copy files using linux commands, and so on.

Example

As shown in the slide, you can view both linux file system files and HDFS file system files using the same linux command syntax.

In this example, the current directory is set to the user's home directory (`/home/oracle`). The linux `ls -l` command is used to show the contents of both:

- The local file system (linux), at the top
- The hadoop namespace (using the Fuse DFS mount point), at the bottom

Note

- Fuse-DFS is not totally identical to linux . For example, there are linux-like commands are not supported. However, the most common file movement and viewing commands are the same.
- To un-mount the Fuse DFS mount point, use the `umount` command (described on the previous slide) as the same user who enabled the mount. In this case, execute the following command: `sudo umount /mnt/hdfs`

What is Flume?

- Is a distributed service for collecting, aggregating, and moving large data to a centralized data store
- Was developed by Apache
- Has the following features:
 - Simple
 - Reliable
 - Fault tolerant
 - Used for online analytic applications



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

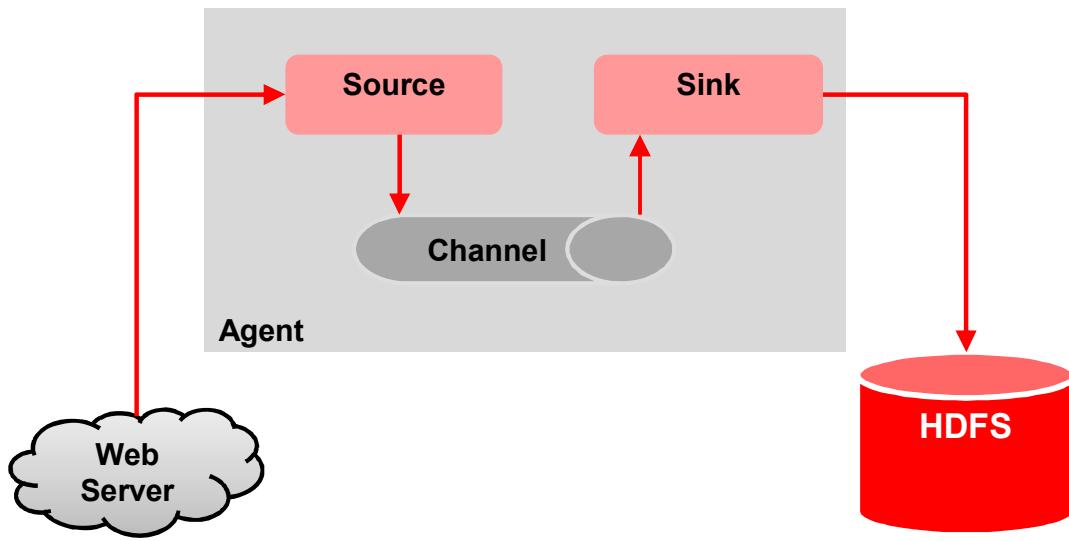
Flume can be simply defined as a distributed system that takes your logs from their source, aggregates them, and then moves them to where you want to process them.

It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant, and it has tunable reliability mechanisms and many failover and recovery mechanisms.

Flume service features include the following:

- Reliability in Flume mainly targets fault tolerance and high availability.
- Scalability focuses on supporting horizontal scalability for all nodes and masters.
- Extensibility is attained by providing a plug-in architecture to support all types of data, sources, and sinks.
- Centralized management is available to support dynamic reconfiguration.

Flume: Architecture



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

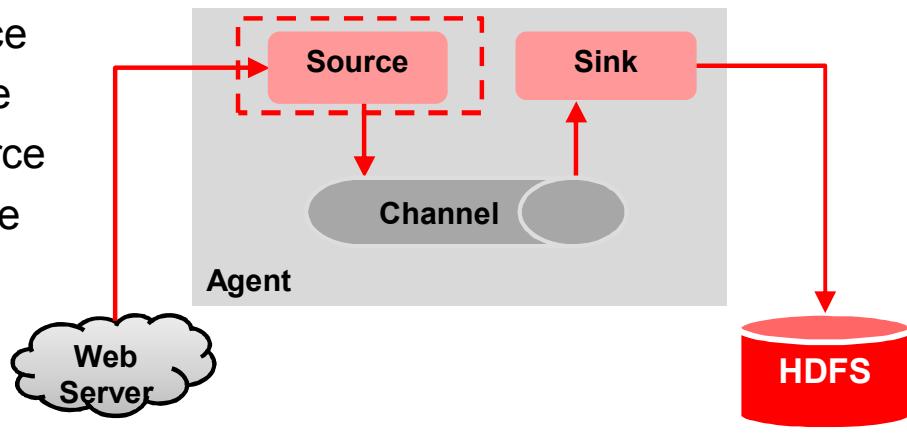
The Flume architecture, illustrated in the slide, comprises the following elements:

- An *event* is a unit of data that flows through a Flume agent. Ultimately, the event flows from source, to channel, to sink, and it is represented by an implementation of the Event interface. An event carries a payload (byte array) that is accompanied by an optional set of headers (string attributes).
- A Flume *agent* is a process that hosts the components that allow events to flow from an external source to an external destination.
- A *source* consumes events having a specific format, and those events are delivered to the source by an external source (such as a web server). When a source receives an event, it stores it in one or more channels.
- The *channel* is a passive store that holds the event until that event is consumed by a sink. One type of channel in Flume is the FileChannel, which uses the local file system as its backing store.
- A *sink* is responsible for delivering the event to the next agent or terminal repository (like HDFS) in the flow. The sink also removes the event from the channel.

Note: The source and sink within the given agent run asynchronously with the events staged in the channel.

Flume Sources (Consume Events)

- Avro source
- Exec source
- Spooling Directory source
- Netcat source
- Sequence Generator source
- Syslog source
- HTTP source
- Custom source
- Scribe source



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

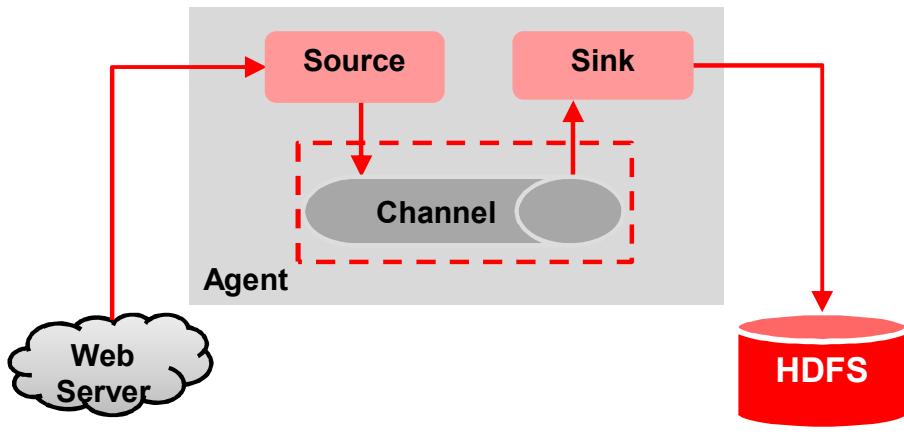
ORACLE

The following is a list of supported Flume sources:

- **Avro source:** Listens on the Avro port and receives events from external Avro client streams
- **Exec source:** Runs a given UNIX command on startup and expects that process to continuously produce data on standard out
- **Spoiling Directory source:** Enables you to ingest data by dropping files in a spooling directory on disk
- **Netcat source:** A netcat-like source that opens a specified port and listens for data. The expectation is that the supplied data is newline-separated text. Each line of text is turned into a Flume event and sent via the connected channel.
- **Sequence Generator source:** A simple sequence generator that continuously generates events with a counter that starts from 0 and increments by 1
- **Syslog source:** Reads syslog data and generates Flume events. The UDP source treats an entire message as a single event. The TCP sources create a new event for each string of characters separated by a new line (`\n`).
- **HTTP source:** A source that accepts Flume events by HTTP POST and GET
- **Custom source:** Your own implementation of the Source interface. The source's class and dependencies must be in the agent's classpath when you start the Flume agent.
- **Scribe source:** Is another type of ingest system

Flume Channels (Hold Events)

- Memory channel
- JDBC channel
- File channel
- Custom channel



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

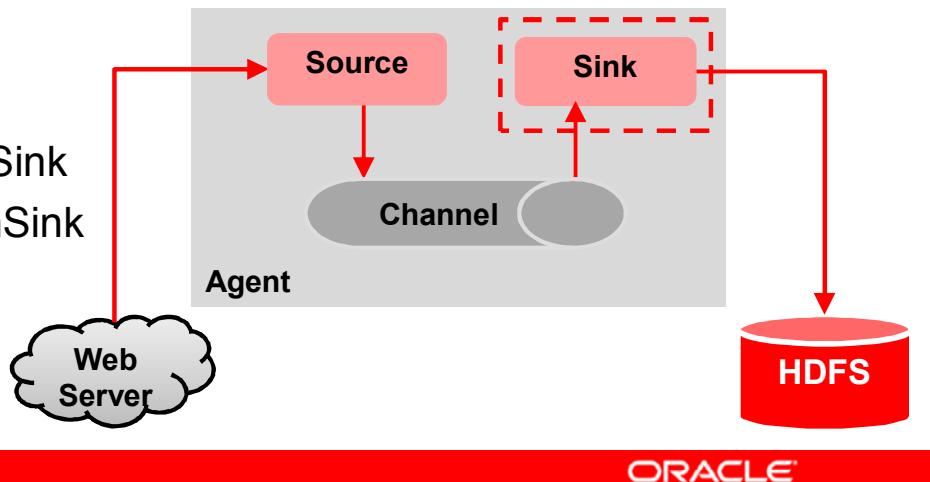
ORACLE

The following is a list of supported Flume channels:

- **Memory channel:** Provides in-memory, fast, nondurable event transport
- **JDBC channel:** Is a JDBC-based, durable event transport that is based on Derby
- **File channel:** Is a durable channel that uses a Write Ahead Log to write events to disk to ensure that data is not lost when an agent or machine shuts down. Any events that have not been taken from the channel at the time of shutdown are replayed when the agent comes back online and are then sent to the next step in the flow. You should use this as your durable channel.
- **Custom channel:** You must specify the fully qualified name of the custom channel and provide that class (and its dependent code) in Flume's classpath. You can do this by creating a JAR file to hold the custom code and then placing the JAR in Flume's `lib` directory.

Flume Sinks (Deliver Events)

- HDFS sink
- Logger sink
- Avro sink
- IRC sink
- File Roll sink
- Null sink
- HBase sink
- AsyncHBaseSink
- ElasticSearchSink
- Custom sink

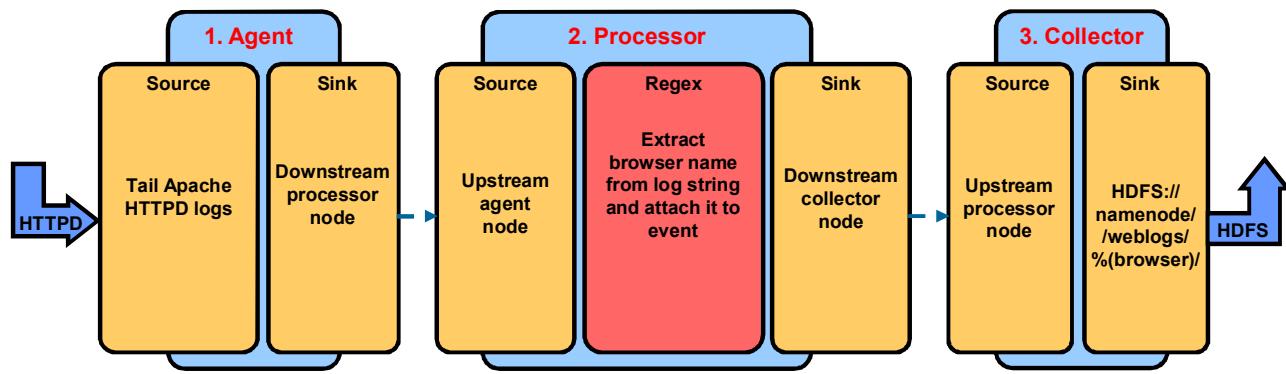


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The following is a list of supported Flume sinks:

- **HDFS sink:** Writes events to HDFS. It currently supports the creation of text and sequence files. It supports compression in both file types.
Note: A version of Hadoop that supports the `sync()` call is required.
- **Logger sink:** Logs events at the INFO level. It is useful for testing and debugging.
- **Avro sink:** Forms one-half of Flume's tiered collection support. Flume events that are sent to this sink are converted to Avro events and then sent to the configured host name.
- **IRC sink:** Takes messages from the attached channel and relays them to configured IRC destinations
- **File Roll sink:** Stores events on the local file system
- **Null sink:** Discards all events that it receives from the channel
- **HBase sink:** Writes data to HBase
- **AsyncHBaseSink:** Writes data to HBase by using an asynchronous model
- **ElasticSearchSink:** Writes data to ElasticSearch
- **Custom sink:** Is your own implementation of the Sink interface. A custom sink's class and its dependencies must be included in the agent's classpath when you start the Flume agent.

Flume: Data Flows



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide shows the layout of a simple flow for capturing web server logs. Three logical nodes are involved, each with different source and sink configurations.

1. The *Agent* watches the tail of an Apache HTTP server log file. Flume's tail source correctly deals with log rotation, so the logging policy does not need to be changed to integrate with Flume. Each line of the log is captured as one event and then sent.
2. The *Processor* receives individual events from the agent, and uses regular expressions to identify and extract the name of the user's browser (Firefox or Internet Explorer) from the log event. This metadata is then attached to the event so that it is available for later processing.
3. The *Collector* receives the annotated events from the processor and writes them to a path in HDFS that is determined by the browser string that was extracted earlier. Flume's HDFS sink uses a simple template language to determine where to write events. In this case, all logs from users using Firefox are saved to /weblogs/Firefox/ in HDFS.

This simple example shows how Flume is able to capture your data to enable you to perform some basic processing on it. This entire flow can be constructed and configured from the Flume Master without logging on to the individual machines involved.

Configuring Flume

1. Create a configuration file (`flume.conf`).
2. Store the file in the `flume-ng/conf` directory.
3. Configure individual components.
4. (Optional) Edit `flume-env.sh`.
5. Verify the installation by running the following command:

```
$ flume-ng help
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Use the following steps to configure Flume:

1. Create a configuration file and edit the file to define the components. You can define your sources, sinks, and channels as well as the flow in an agent. By default, the properties file is configured to work out of the box, with a sequence generator as the source, a logger as the sink, and a memory channel.
2. Store the file in the `flume-ng/conf` directory.
3. Configure individual components.
4. (Optional) Edit `flume-env.sh`. The `flume-ng` executable looks for a file named `flume-env.sh` in the `conf` directory and sources the file if it finds it. This file is effectively empty by default, so it has no effect on Flume unless you add content to it.
5. Verify the installation by running `$ flume-ng help`. You should see the Flume usage options.

Exploring a flume*.conf File

```
[oracle@bigdatalite ~]$ cd /home/oracle/movie/moviedemo/scripts
[oracle@bigdatalite scripts]$ ls
1_start_movieapp.sh      flume_avro.conf      reset_nosqldb.sh
2_flume_tail_movielog.sh flume_json.conf    tail_log.sh
3_nosqldb_console.sh    flume_movieagent.sh
[oracle@bigdatalite scripts]$ more flume_avro.conf
# Chain: log -> json-to-avro interceptor -> memory channel -> hdfs sink
movieagent.sources=logFile
movieagent.channels=memoryChannel
movieagent.sinks=hdfs-sink

# Application log source
movieagent.sources.logFile.type=exec
movieagent.sources.logFile.command=tail -F /u01/Middleware/logs/activity.out
movieagent.sources.logFile.channels=memoryChannel
movieagent.sources.logFile.interceptors=jsonToAvro
movieagent.sources.logFile.interceptors.jsonToAvro.type=oracle.avro.ActivityJson
ToAvroInterceptors$Builder
movieagent.sources.logFile.interceptors.jsonToAvro.key=flume.avro.schema.url
movieagent.sources.logFile.interceptors.jsonToAvro.value=hdfs://bigdatalite.loca
ldomain/user/oracle/moviework/schemas/activity.avsc

# Memory Channel
movieagent.channels.memoryChannel.type=memory
movieagent.channels.memoryChannel.capacity=100

# HDFS Sink
movieagent.sinks.hdfs-sink.type=hdfs
movieagent.sinks.hdfs-sink.hdfs.path=hdfs://bigdatalite.localdomain/user/oracle/
moviework/applog avro
movieagent.sinks.hdfs-sink.hdfs.filePrefix=streamed-movieapp
movieagent.sinks.hdfs-sink.hdfs fileType=DataStream
movieagent.sinks.hdfs-sink.hdfs.rollInterval=60
movieagent.sinks.hdfs-sink.hdfs.fileSuffix=.avro
movieagent.sinks.hdfs-sink.channel=memoryChannel
movieagent.sinks.hdfs-sink.serializer=org.apache.flume.sink.hdfs.AvroEventSerial
izer$Builder
#movieagent.sinks.hdfs-sink.serializer.compressionCodec=snappy
[oracle@bigdatalite scripts]$
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The output in the slide shows the Avro configuration file for the Movieplex sample application in which Flume is used to extract the log files and load them into HDFS.

The Avro configuration file includes:

- A summary of the chain, which is highlighted in the slide
- Values for the Application log sources, Memory Channel, and HDFS Sinks. The application sinks path value is highlighted in the slide.

Note: In addition to the Flume Avro configuration file, notice that the sample application also has a Flume JSON configuration file.

Additional Resources

- <http://flume.apache.org/index.html>



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned to:

- Describe Uses of the Command Line Interface (CLI)
- Describe the benefits of Fuse DFS
- Define Flume
- Describe the data-flow mechanism of Flume
- Identify the options for configuring Flume



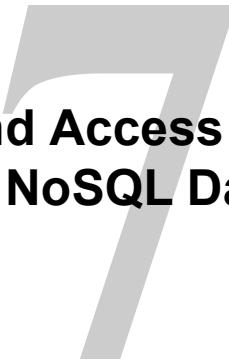
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 6: Overview

In this practice, you view the configuration options used for Flume on the Big Data Lite VM.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.



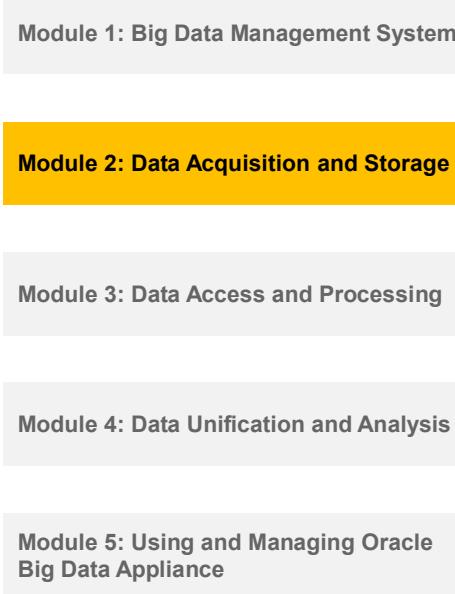
Acquire and Access Data Using Oracle NoSQL Database



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



Lesson 5: Introduction to the Hadoop Distributed File System (HDFS)

Lesson 6: Acquire Data using CLI, Fuse-DFS, and Flume

Lesson 7: Acquire and Access Data Using Oracle NoSQL Database

Lesson 8: Primary Administrative Tasks for Oracle NoSQL Database

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The final big data acquisition and storage technology that we examine in this module is NoSQL.

In this lesson, you learn about NoSQL Database, and how to acquire and access big data using Oracle NoSQL. In the next lesson, you learn some of the administrative tasks associated with Oracle NoSQL Database.

Objectives

After completing this lesson, you should be able to:

- Describe NoSQL Database characteristics
- Differentiate NoSQL from RDBMS and HDFS
- Describe Oracle NoSQL Database benefits
- Load and remove data in an Oracle NoSQL DB
- Retrieve data from an Oracle NoSQL DB



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

What is a NoSQL Database?

- Is a key-value database
- Is accessible by using Java APIs
- Stores unstructured or semi-structured data as byte arrays



Benefits

- Easy to install and configure
- Highly reliable
- General-purpose database system
- Scalable throughput and predictable latency
- Configurable consistency and durability

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

NoSQL Database is a nonrelational database. It is written in Java, and uses a key-value data model.

To write data to, or read data from, a NoSQL Database, you must use Java-based APIs in an application. We will examine some of these API's at a high level later in this lesson. NoSQL stores data as byte arrays. You can store any type of data in NoSQL Database as long as you can convert it into the byte format.

RDBMS Compared to NoSQL

RDBMS	NoSQL
High-value, high-density, complex data	Low-value, low-density, simple data
Complex data relationships	Very simple relationships
Joins	Avoids joins
Schema-centric, structured data	Unstructured or semi-structured data
Designed to scale up (not out)	Distributed storage and processing
Well-defined standards	Standards not yet evolved
Database-centric	Application- and developer-centric
High security	Minimal or no security



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

NoSQL databases differ from the traditional RDBMS in many ways. The slide lists some of the key differences between these two databases.

HDFS Compared to NoSQL

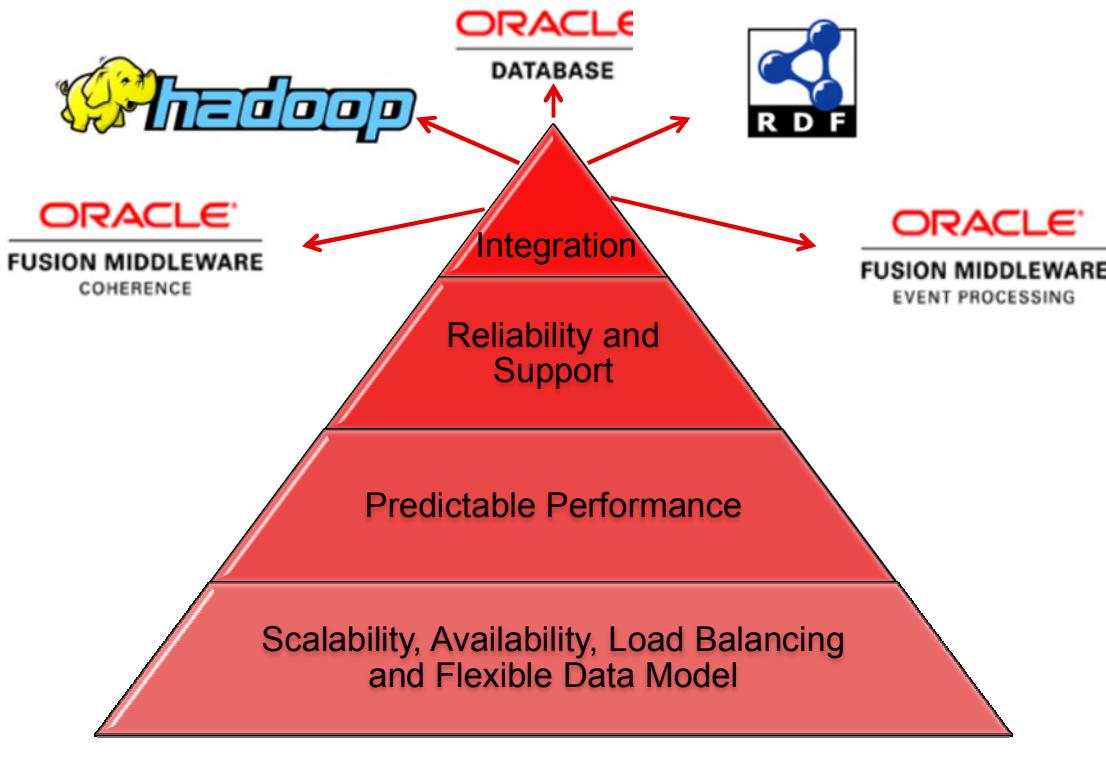
HDFS	NoSQL
File system	Database
No inherent structure	Simple data structure
Batch-oriented	Real-time
Processes data to use	Delivers a service
Bulk storage	Fast access to specific records
Write once, read many	Read, write, delete, update



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

NoSQL databases differ from HDFS in certain ways. The slide lists some of the key differences between these two storage systems.

Oracle NoSQL Database



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

Oracle has its own NoSQL solution, based on the Oracle Berkley DB Java Edition, which provides fast performance for key-value data. Oracle NoSQL Database is designed to provide the following benefits:

- **Integration:** ONDB's integrates with Oracle's enterprise solutions, distinguishing it from other NoSQL solutions. You can query ONDB from Oracle Database using external tables. You can also access ONDB from Hadoop. You can store and query RDF data, and you can cache eventstreams as well as share data for extensible in-memory cache grid.
- **Reliability and support:** ONDB provides reliable commercial grade software and offers excellent support for both the enterprise and community editions.
- **Predictable performance:** ONDB's performance is predictable as you scale out. The throughput increases linearly and the latencies are predictable.
- **Scalability:** When the volume of data increases, you can add additional storage nodes to store the data conveniently and quickly.
- **High availability:** You can configure ONDB to have one or more replicas for each storage node. This guarantees high availability and no single point of failure.
- **Transparent load balancing:** ONDB has an intelligent driver that is attached to your application. Depending on the load, this driver automatically distributes the requests among the master and replica nodes.
- **Flexible data model:** The data is stored in ONDB by using the key-value data model. This is the simplest and most efficient data model for storing unstructured data.

Points to Consider Before Choosing NoSQL

When deciding on an applications database technology, you should analyze:

- The data to be stored
 - High volume with low value?
If answer is “yes,” NoSQL is a good choice.
- The application schema
 - Dynamic?
If answer is “yes,” NoSQL is a good choice.

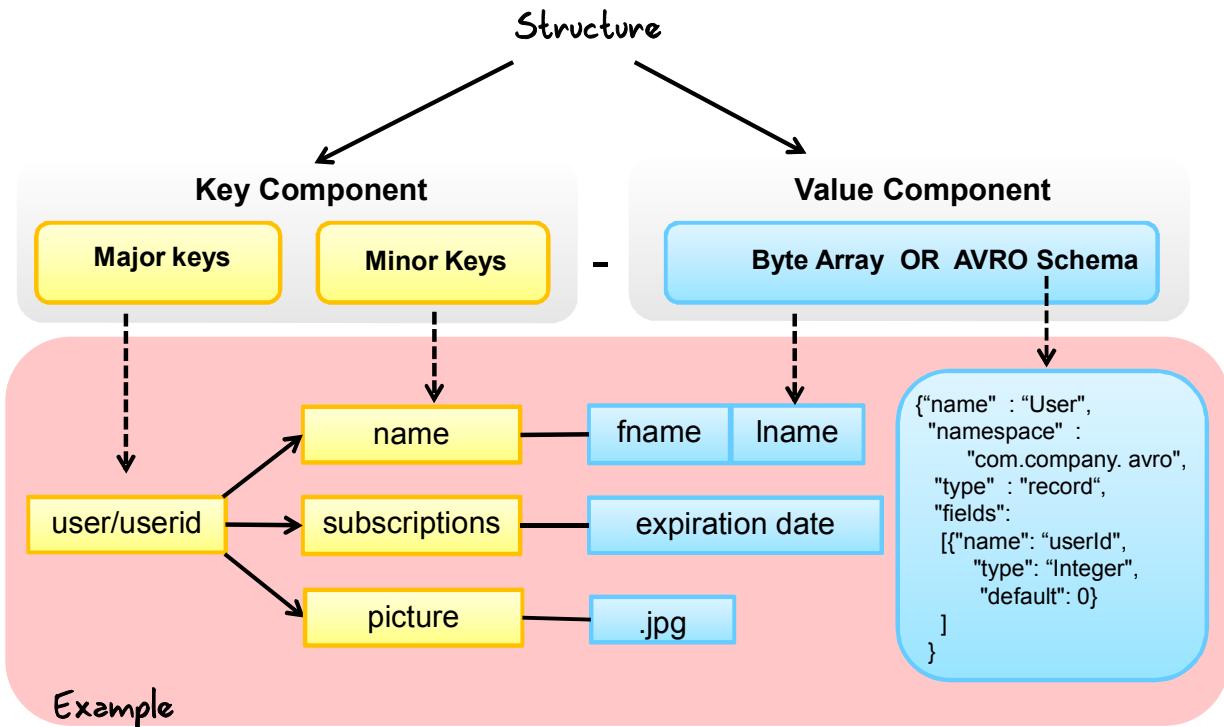


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

While deciding on the storage technology for an application, you must consider two main aspects: the data to be stored and the application schema.

- If you have huge amounts of low-density, sparse, low-value data to be stored, a NoSQL database may be the better choice. If you have high-value data, an RDBMS is better.
- If the application’s schema is dynamic (that is, if it constantly changes), a NoSQL database might be a better choice. If the application’s schema is fixed, an RDBMS can be better.

NoSQL Key-Value Data Model



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The schema structure of the key-value data model is very simple. Data in this model is not stored in fixed table-like structures. Each record consists of a key-value pair.

Key Component

The key (key component) is used to uniquely identify a value in the KVStore. A key is a list of values of the String data type. You define the structure of a key within the application; all the logic to process the key and records is contained within the application itself.

A key may consist of major-key and minor-key components.

- A key can have more than one major or minor component. However, all keys must have at least one major component.
- If a key has minor components, the combination of both the major-key and minor-key components uniquely identifies a record.
- In some situations, defining a key with only one major-key component is sufficient.

In the example, we have an application that tracks details of all employees in an organization. It is using the combination of the word “user” and a user ID as major keys. In addition, there are three different minor key components. The major and minor key components enable the system to identify records uniquely.

As the number of records grows in the store, performance problems can result. As you go, you can define minor-key components to further organize your data. However, when designing a key, remember that there is overhead associated with each key that the store maintains. Defining too many key components might not be the best solution in terms of performance.

Value Component

The *value* part of a record is the data that you want to store and manage by using ONDB. The value component can be any type of data, and is stored in ONDB as a byte array.

Mapping of data structures (that is, serialization and de-serialization) must be taken care of in the application itself. The value component can be as large or as small as you want it to be. However, larger records take a longer time to read and write than shorter records.

In the key-value data model, you can also create the value component by using AVRO to describe a schema for the value component. An AVRO schema describes the fields allowed in the value and their data types. You create and define an AVRO schema and then apply the schema to the value portion of a record using AVRO bindings.

Key-Value Pair

In the slide example:

- The key consists of:
 - Two major keys (static text “user” and userid)
 - Three minor keys (name, subscriptions, and picture)
- The value component can be raw byte values or can be stored by using AVRO schema.

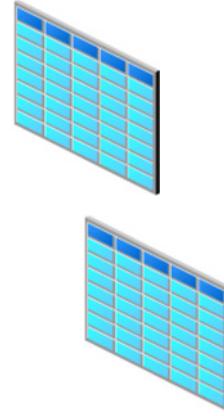
Non Self-Describing

The key-value data model is not self-describing. That is, there is no way to know what the various components of a key are, and no way to know what data is stored as the value. Therefore, the application developer using ONDB must know the structure of the key and value fields.

Acquiring and Accessing Data in a NoSQL DB

Primary Tasks

- Creating Tables
 - Create a table (parent or child)
 - Add the table to the KVStore
- Adding or Removing Data
 - Use the Table API
 - Insert Rows
 - Delete Rows
- Reading Data
 - Retrieve a single record
 - Retrieve multiple records



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To load data into the NoSQL database, you first create tables and add the tables to the KVStore. Second, you load (and remove) data from the tables to provide user access to the data. We will examine these tasks a bit later in this lesson.

First, let us look at the NoSQL table models.

Primary (Parent) Table Data Model

Table Name

Primary Key1	Field 1	Field 2	Field 3
--------------	---------	---------	---------

Table Name : User



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This slide provides a visual illustration of the Parent Table data model.

In this example, there is a single primary key (`userId`), and two fields (`firstName` and `lastName`).

Table Data Model: Child Tables

Table Name : **User**

Primary Key	“Value”
userId	firstName lastName

Child Table Name: **Folder**

Primary Key	“Value”
UserID Folder Name Arrival Date	From To Sender CC Subject Msg Body



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This slide provides a visual illustration of the Child Table data model, where the primary key is made up of the parent table key and one or more child table keys.

The example shows the primary key is made up of three elements (`userID`, `Folder Name`, and `Arrival Date`). And, there are six fields (`From`, `To`, `Sender`, `CC`, `Subject` and `Msg Body`).

Creating Tables

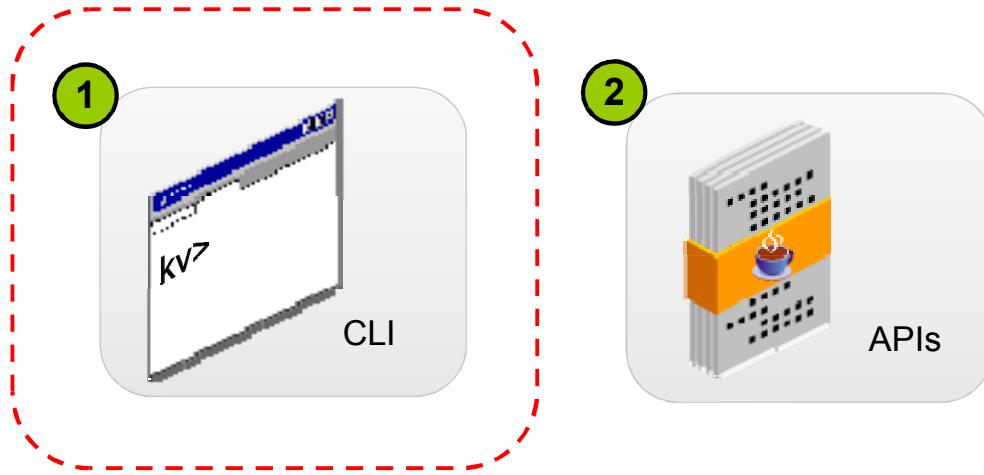


ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In this next section, you learn how to create tables in the KVStore.

Creating Tables: Two Options



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can create, modify, and delete tables in a KVStore using two options:

1. **Data CLI:** Directly execute DDL commands at the `kv` prompt.
2. **Java APIs:** Create DDL commands and invoke APIs to execute the commands from a Java application.

In this lesson, you learn how to create tables using the CLI. First, you will learn to create the DDL commands.

Data Definition Language (DDL) Commands

CREATE TABLE

DESCRIBE [AS JSON] TABLE
SHOW [AS JSON] TABLES

- Keys words are reserved.
- Table and index names limited to 32 chars
- Field names can be 64 chars.
- Only alphanumeric + “_” allowed
- All names to start with a letter

ALTER TABLE ADD FIELD



ALTER TABLE DROP FIELD

DROP TABLE



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The DDL commands available for Oracle NoSQL Database are listed in the slide. You will learn to use the CREATE, ALTER, and DROP commands in the coming slides. To view the description of the created tables, you use the DESCRIBE command. You can specify AS JSON to view the description in JSON format. To view the tables created in the KVStore, you use the SHOW command. Here also, you can specify AS JSON to view the output in JSON format.

While using these commands, remember the following:

- The DDL command keywords are reserved words and you cannot use them to identify table, index, or field names.
- The table and index names are limited to 32 characters.
- Field names can be 64 characters.
- All table, index, and field names are restricted to alphanumeric characters, plus underscore (“_”).
- All names must start with a letter.

Each of these commands is executed by using a Java API or an execute command, which you will learn later in this lesson.

CREATE TABLE

```
CREATE TABLE [IF NOT EXISTS] table-name (
    field-definition,
    field-definition-2 ...,
    PRIMARY KEY
    ([SHARD] field-name, field-name-2...),
    [COMMENT "comment string"] )
```

```
field-name type
[CHECK]
[NOT NULL]
[DEFAULT "default-value"]
[COMMENT "comment-string"]
```

Field constraints

Supported Datatypes

- Array
- Boolean
- Binary
- Double
- Enum
- Fixed Binary
- Float
- Integer
- Long
- Map
- Record
- String

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The slide shows the syntax for creating a table. The `CREATE TABLE` command has the following parts:

- `[IF NOT EXISTS]`: This is an optional phase. If specified, it creates a table only if a similar table is not already present in the KVStore. If not specified and if a table with the same name exists, an error is thrown.
- `table name`: The name you specify for the table. It should conform to the standards listed in the previous slide. To create a child table, prefix the child table name with the parent table name using a dot notation.
- `field definition`: A comma-separated list of field names mentioned as per the syntax for fields shown in the slide. The various types you can specify for a field are also listed in the slide.
 - All the elements of the `ARRAY` must be of the same data type that must be declared when you define the array.
 - When using an `ENUM`, all the acceptable values should be specified when you define the field.
 - Keys in a `MAP` are of String data type. You must specify the data type of the data portion of a `MAP`.
 - Use the `ELEMENTOF` keyword to refer to elements of an `ARRAY` or a `MAP`.

You can create the following constraints on the field values during the field definition itself.

- CHECK: Used to restrict the field values to a range of values. This parameter is discussed in detail in the next slide.
- NOT NULL: To enforce a value for this field. You can specify a default value for a field by using the DEFAULT keyword.

You can also include a description or comment for a field by using the COMMENT keyword.

Comments are written using any of the following styles:

```
/* This is a comment*/  
//This is a comment  
#This is a comment
```

- PRIMARY KEY: While creating a table, you need to specify a primary key field for the table. The values of this field are used to uniquely identify a row in the table. You can specify a composite primary key. That is, two or more fields can be used to create a primary key for the record.
- SHARD: This is an optional keyword used to identify the SHARD key for a table. As learned in Unit I, shard keys are used to ensure a set of records are always stored together in a single shard.
- COMMENT: This is used to document a comment or description for the table.

Accessing the CLI

Access the CLI by invoking the `runadmin` utility or the `kvcli.jar` file.

1

```
java -jar $KVHOME/lib/kvstore.jar runadmin  
      -port <port number>  
      -host <host name>
```

Admin CLI

2

```
java -jar $KVHOME/lib/kvcli.jar  
      -port <port number>  
      -host <host name>  
      -store kvstore
```

Data CLI

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle NoSQL Database provides two command-line interfaces (CLIs): an admin CLI and a data CLI. Currently, both the CLIs provide the same functionality. That is, administration tasks as well as developer tasks can be performed using either of these CLIs.

1. You can access the admin CLI interface by invoking the `runadmin` utility of the `kvstore.jar` file. You need to pass the `port` and `host` parameter details.
2. You can access the data CLI by invoking the `kvcli.jar` file. In addition to the `host` and `port` parameter values you need to pass the name of the store, which you mentioned while configuring the store.

The complete syntax for invoking these CLIs is shown in the slide.

You use the CLI to create and define the application tables, fields, keys, as well as indexes. You can also insert and retrieve table data as well as key-value data using CLI.

You can run single line commands directly when invoking the CLI. You can also use a load command to run a script while invoking the CLI. If no other parameters are passed, then the CLI interface is shown and you can use it interactively to run commands.

Executing a DDL Command

```
kv-> execute "CREATE TABLE user (userId STRING,\n>name RECORD (first STRING, middle STRING, last STRING),\n>age INTEGER CHECK(age >=0 AND age<=200),\n>gender ENUM(M,F),\n>email STRING, \n>password STRING, \n>active BOOLEAN DEFAULT TRUE, \n>createdOn LONG,\n>modifiedOn LONG,\n>PRIMARY KEY (userId))"■
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To run the DDL command at the `kv` prompt, use the `execute` command. The DDL command is passed inside single or double quotation marks to the `execute` command. The DDL command gets executed and the `kv` prompt is returned.

Viewing Table Descriptions

```
All Tables in KVStore
execute "show tables"
execute "show AS JSON
tables"
```

```
kv-> show tables
Tables:
Message
Sequence
User
User.Folder
User.Folder.Message
```

```
Specific Table
```

```
execute "describe AS
JSON table <tablename>
[fields]"
```

```
kv-> show table -name User.Folder.Message
{
  "type" : "table",
  "name" : "Message",
  "comment" : null,
  "parent" : "Folder",
  "shardKey" : [ "userId" ],
  "primaryKey" : [ "userId", "folderId", "messageId" ],
  "fields" : [ {
    "name" : "userId",
    "type" : "string",
    "comment" : null,
    "shardKey" : false,
    "primaryKey" : true,
    "length" : null,
    "precision" : null,
    "scale" : null,
    "scaleType" : null
  },
  {
    "name" : "folderId",
    "type" : "string",
    "comment" : null,
    "shardKey" : false,
    "primaryKey" : false,
    "length" : null,
    "precision" : null,
    "scale" : null,
    "scaleType" : null
  },
  {
    "name" : "messageId",
    "type" : "string",
    "comment" : null,
    "shardKey" : false,
    "primaryKey" : false,
    "length" : null,
    "precision" : null,
    "scale" : null,
    "scaleType" : null
  },
  {
    "name" : "text",
    "type" : "string",
    "comment" : null,
    "shardKey" : false,
    "primaryKey" : false,
    "length" : null,
    "precision" : null,
    "scale" : null,
    "scaleType" : null
  }
]
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

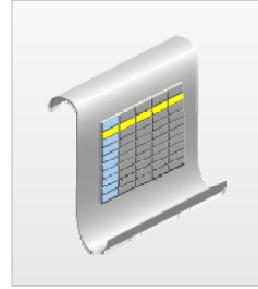
To view the tables existing in a KVStore, use the `show tables` command in the CLI. The parent and child tables are clearly listed in the output of this command.

To view the schema of a specific table, use the `show tables <tablename>` command. The table schema is listed in JSON format. The slide output shows a part of the description for the `User.Folder.Message` table.

Recommendation: Using Scripts

Use scripts for all database operations:

- This is the recommended approach.
- Prevents accidental errors/typos.
- Consistent store environment through all cycles of development, testing, and deployment.



1

```
java -jar $KVHOME/lib/kvstore.jar runadmin  
      -port <port number>  
      -host <host name>  
      load -file <path to script>
```

2

```
kv> load -file <path to script>
```

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can use the CLI interactively for creating tables and performing other operations. However, while working in an enterprise environment it is recommended that you create and run scripts to perform these operations. Database table operations include table creation, evolution, and deletion. Using scripts avoids typos and other errors as code moves from the development environment to a test environment and finally to the production environment.

A CLI script is a simple text file that contains a series of commands. To run the script:

1. You start the CLI and use a `load` command to run the file that has the commands.
2. You can also use the `load` command after connecting to the KVStore.

Loading Data Into Tables



ORACLE®

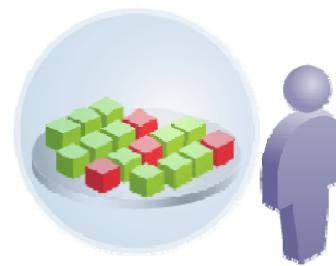
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Next, you learn how to insert data into tables in the KVStore.

Accessing the KVStore

You access the KVStore for two different reasons.

- For access to stored data:
 - Use Java APIs
- For administrative actions:
 - Use the command-line interface
 - Use the graphical web console



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To perform an operation on ONDB data, you must access the KVStore. You can do this by using Java APIs for ONDB in your applications. You learn to use some of the APIs later in this lesson.

You may also access the KVStore while performing administrative operations. You can use the CLI to perform administrative tasks. Alternatively, you can configure and monitor the KVStore by using the graphical web administration console. Installation and configuration of ONDB is covered in the next lesson.

Introducing the TableAPI

- TableAPI is an interface to the Tables in a KVStore.
- To perform any operation on Tables, you should obtain an instance of TableAPI.

TableAPI methods summary			
3 for Deletes 4 for Inserts	5 for fetching records 3 for fetching keys only	2 for creating and executing transactions	Map<String, Table> getTables() Table getTable (String tableName)

Discussed next

Discussed later



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To perform any operation on the tables created in a KVStore, you need to use a TableAPI Java interface written for ONDB. The TableAPI interface consists of a number of useful methods. The slide shows a summary of these methods.

In this lesson, you learn how to use a few of these methods to write data to, and read data from, tables.

Write Operations: `put()` Methods

<code>put</code>	<code>putIfAbsent</code>	<code>putIfPresent</code>	<code>putIfVersion</code>
Writes to the table irrespective of whether the record is present or not	Writes to the table only if the record is not already present	Writes to the table only if the record is already present	Writes to the table only if the record present is same as a specified version
Definite write	Creation	Updates	Conditional Update



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The slide shows the four APIs available to write records to a table. Though all the methods are used to write records to a table, each of the methods has a specific functionality. The `put` method is used to ensure that the record you want to write is always written to the table irrespective of the fact that the same record is already present in the table or not. In other words, if the data is not present in the table, a new row is created. In case the data is already present in the table, the row is updated with the field values you specify in the put operation.

The `putIfAbsent` method is used when you want to make sure that the write operation results in a record creation only. If the same record exists in the table, the write operation is not performed. Similarly, the `putIfPresent` method is used when you want to ensure that the write operation results in a record updation only. If the same record is not present in the table, the write operation is not performed.

Last, the `putIfVersion` method is used when you want to perform a write operation only if the version of the record in the table is same as the version specified in the write operation. If the versions do not match, the write operation is not performed.

Writing Rows to Tables: Steps

To write a row into a table, perform the following steps:

1. Obtain TableAPI handle.
2. Construct handle to the table.
3. Create a Row object.
4. Add field values to the row.
5. Write the row to the table.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To use the put methods discussed in the previous slide and to write records to a table, you have to code the steps listed in the slide. In the next couple of slides, you will look at these steps in detail.

Constructing a Handle

1. Obtain TableAPI handle.
2. Construct handle to the table.

```
TableAPI tableH = kvstore.getTableAPI();  
Table myTable = tableH.getTable("myTable");
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The first steps are to obtain the handles to the TableAPI interface as well as the table where you want to perform the write operation. The syntax for these commands is shown in the slide.

Using the TableAPI handle, you obtain the handle to the required table by calling the getTable method.

Creating Row Object, Adding Fields, and Writing Record

3. Create a Row object.
4. Add field values to the row.
5. Write the row to the table.

```
Row row = myTable.createRow();

row.put("item", "Bolts");
row.put("description", "Hex head, stainless");
row.put("count", 5);
row.put("percentage", 0.2173913);

tableH.put(row, null, null);
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

After obtaining the required handles, you need to:

- Create a Row object for the table.
- Then, using one or more of the put () methods, you specify the field names and field values for all the fields in the table.
- Finally, you write the data to the table by using the appropriate put () method.

In the example in the slide, the basic put () method is used.

Reading Data from Tables



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Now, you learn how to read data from tables in the KVStore.

Read Operations: get () Methods

get	multiGet	multiGetIterator	tableIterator
Retrieves a single record from the table	Retrieves a set of records from the table	Retrieves a set of records from the table in batches	Retrieves a set of records or indexes from the table
-	Atomic	non-atomic	non-atomic
-	-	single thread	multiple threads



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The slides shows the four methods available to read records from tables in a KVStore.

- Use the `get` method to retrieve a single record from the table. If the table has a composite primary key then you must specify the complete primary key to uniquely identify a single record in the table.
- Use the `multiGet` method to retrieve more than one record from the table. This method fetches all the records in a single atomic operation. You should use this method only when you are sure that the result set will completely fit in the memory.
- If you have a very large set of records that need to be fetched, then you can use the `multiGetIterator` method. This method fetches the records in batches. You can specify the number of records to be retrieved in a batch. However, atomicity of the operation is lost.
- If you want to retrieve all the records from the table or you need to iterate over the entire table or read indexes, then use the `tableIterator` method. This method can issue multiple threads to perform parallel scans of the table to improve the performance of the operation.

In the next couple of slides, you see examples of these methods.

Retrieving Table Data: Steps

To fetch data from a table, perform the following steps:

1. Obtain TableAPI handle,
2. Construct handle to the table
3. Create primary key
4. Use appropriate method to fetch rows
5. Retrieve field values from rows



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Similar to the write operations, to perform a read operation you need to follow the steps listed on the slide.

- Create a TableAPI and a Table handle (1. and 2.)
- Next, create a complete or partial primary key and pass to the appropriate get method to fetch a single or multiple records. (3.)
- You can then retrieve the fields values from the fetched rows. (4. and 5.)

Retrieving Single a Row

```
1 TableAPI tableH = kvstore.getTableAPI();  
2 Table myTable = tableH.getTable("myTable");  
3 PrimaryKey primaryKey = myTable.createPrimaryKey();  
primaryKey.put("item", "Bolts");  
4 Row row = tableH.get(primaryKey, null);  
5 String item = row.get("item").asString.get();  
String description = row.get("description").asString.get();  
Integer count = row.get("count").asInteger.get();  
Double percentage = row.get("percentage").asDouble.get();
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The slide shows an example code to fetch a single record from a table:

1. Obtains the TableAPI handle
2. Constructs a handle to the table
3. Creates the primary key
4. Uses the get() method to fetch rows
5. Retrieves field values from the rows using the get() method

Retrieving Multiple Rows

```
A    ArrayList<Row> myRows = null;
try
{
myRows = tableH.multiGet(key, null, null);
} catch ()
{}

B    for (Iterator<Row> iter = myRows.iterator(); iter.hasNext());
{
Row theRow = iter.next();
String itemType = theRow.get("itemType").asString.get();
String itemCategory = theRow.get("itemCategory").asString.get();
String itemClass = theRow.get("itemClass").asString.get();
String itemColor = theRow.get("itemColor").asString.get();
String itemSize = theRow.get("itemSize").asString.get();
Float price = theRow.get("price").asFloat.get();
Integer price = theRow.get("itemCount").asInteger.get();
}
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can retrieve multiple rows from a table by using the `multiGet` method, as shown in the example above. You should use this method only when you know the resulting rows will fit completely in the memory. If you want to retrieve a large set of rows then you should use the `multigetIterator` method.

To retrieve multiple rows, the initial steps are same as retrieving a single row, as shown in the previous example. Namely, you obtain a `TableAPI` handle and a handle to the table you want to retrieve. Then:

- To fetch the rows, create the partial primary key using the `multiget` method (as shown in this example) or the `multigetIterator` method. Note: These methods must be used within a `try catch` loop.
- Then, you can read the individual rows by using an `Iterator` in a `for` loop to retrieve each row and field value from the rows.

Retrieving Child Tables

```
TableAPI tableH = kvstore.getTableAPI();

Table myChildTable = tableH.getTable("myTable.myChildTable");

PrimaryKey primaryKey = myChildTable.createPrimaryKey();
primaryKey.put("itemCategory", "Bolts");
primaryKey.put("itemSKU", "1392610");

Row row = tableH.get(primaryKey, null);

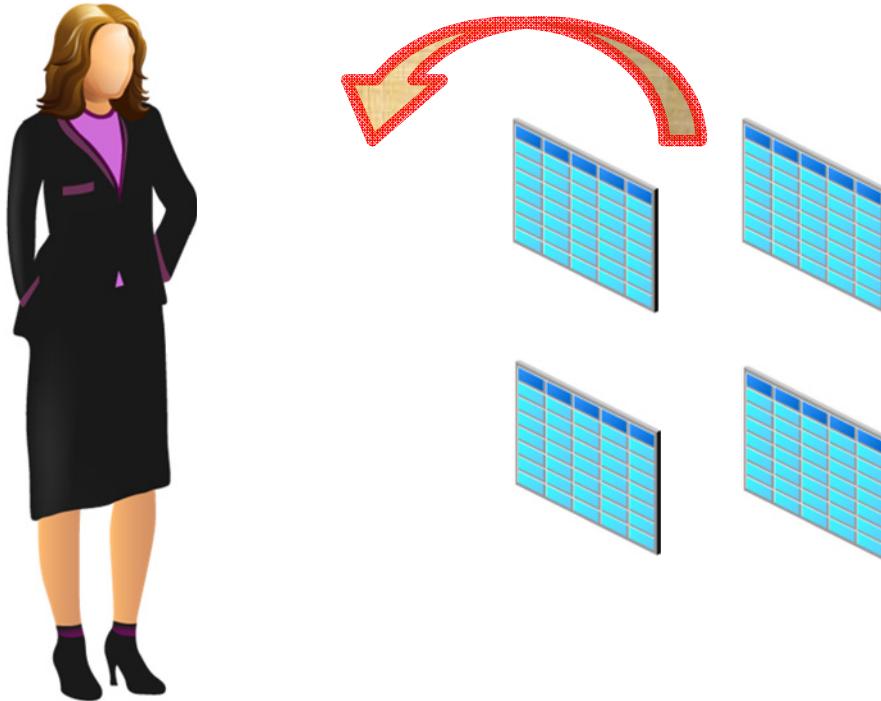
String description = row.get("itemDescription").asString().get();
Float price = row.get("price").asFloat().get();
Integer invCount = row.get("inventoryCount").asInteger().get();
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To retrieve a child table record, you perform the same steps as discussed for retrieving a single row. However, when you create the `primaryKey` object, specify the primary key of the parent table as well as the parent key of the child table.

Removing Data From Tables



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Next, you learn how to delete data from tables in the KVStore.

Delete Operations: 3 Table APIs

delete	multiDelete	deleteIfVersion
Deletes a single record from a table	Deletes multiple records from tables	Deletes a single record from a table if the version of the record is same as a specified version



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

There are three APIs available to delete records from a table.

- You use the `delete` method to delete a single record from the table.
- You can delete multiple rows from a table by using the `multidelete` command. The `multidelete` command uses a partial primary key to identify a set of records for deletion.
- If you want to match the version of a row before it is deleted, you use the `deleteIfVersion` method.

Deleting Row(s) From a Table: Steps

To write a row into a table, perform the following steps:

1. Obtain TableAPI handle.
2. Construct handle to the table.
3. Create the primarykey object.
4. Specify primary key values to be deleted.
5. Delete the row from the table.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To delete a record from a table, you need to perform the steps listed in the slide.

- As in all other table operations, you need to obtain an instance of TableAPI and a handle to the table from which you want to delete the row or rows.
- Then, you then create a primary key object and specify the value of primary key for the rows you want to delete.
- After creating the primary key, you use the appropriate method to delete the rows.

Additional Resources

- Oracle University two-day course:
Oracle NoSQL Database for Developers



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Summary

After completing this lesson, you learned how to:

- Describe NoSQL Database characteristics
- Differentiate NoSQL from RDBMS and HDFS
- Describe Oracle NoSQL Database benefits
- Load and remove data in an Oracle NoSQL DB
- Retrieve data from an Oracle NoSQL DB



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In this lesson, you learned how to acquire and access data by using Oracle NoSQL Database.

Practice 7: Overview

In this practice, you:

- Run the MoviePlex Application
- Start Oracle NoSQL Database instance
- Load User Profile Data
- Load Movie Data
- Query the Movie Data



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This is a hands-on practice.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only

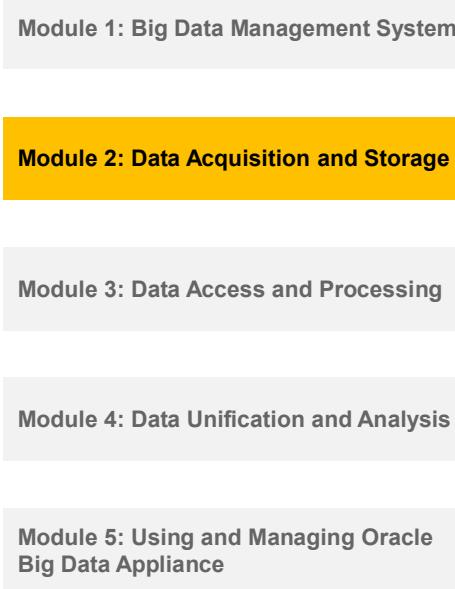
Primary Administrative Tasks for Oracle NoSQL Database

8

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



Lesson 5: Introduction to the Hadoop Distributed File System (HDFS)

Lesson 6: Acquire Data using CLI, Fuse-DFS, and Flume

Lesson 7: Acquire and Access Data Using Oracle NoSQL Database

Lesson 8: Primary Administrative Tasks for Oracle NoSQL Database

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In this last lesson on big data acquisition and storage technologies, you learn about the primary administrative tasks for Oracle NoSQL Database.

Objectives

After completing this lesson, you should be able to:

- Plan for an installation
- Install and configure Nodes
- Deploy a KVStore



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Installation Planning: KVStore Analysis

1. Determine application characteristics:
 - Replication factor
 - Average key and value (record) size
 - Percentage of read operations
2. Determine hardware characteristics:
 - Number of disks per machine
 - Size of disk
 - Input/Output operations per second
3. Determine shard storage and throughput capacities
4. Determine machine memory and network throughput



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To deploy a KVStore, you have to specify:

- A replication factor
- The desired number of partitions
- The storage nodes on which to deploy the store

Before the actual deployment you must calculate these values. Depending on an application's requirements and the characteristics of the hardware available to host the KVStore, you must identify the capacity of a single shard and then identify the number of shards required to fulfill the application requirements.

The slide provides a list of tasks you need to perform while analyzing the workload of a KVStore.

InitialCapacityPlanning Spreadsheet

<KVHOME>/doc/misc/InitialCapacityPlanning.xls

Application Characteristics	
<i>RF</i>	3 The replication factor for the entire store
<i>AvgKeySize</i>	16 the average key size
<i>AvgValueSize</i>	1,000 the average value size
<i>ReadOpsPercent</i>	50% Percentage of total ops that are application reads. The remaining ops are assumed to be writes.
<i>ReadCacheHitPercent</i>	The number of reads that will be satisfied from the FS cache. This number is a function of both the application read patterns and the amount of file system cache. 10%

Hardware Characteristics	
<i>DisksPerMachine</i>	1 The number of disks per machine used to store KV pairs. One RN/disk is assumed in the calculations below.
<i>DiskCapacityGB</i>	900 the usable disk capacity in GB
<i>DiskIopsPerSec</i>	200 IOPs/sec from disk specs ~200 for a hard disk ~2000 for an SSD

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle NoSQL Database software distribution contains a spreadsheet that should be used in the planning process. This spreadsheet is available at the following location in the Oracle NoSQL Database distribution: <KVHOME>/doc/misc/InitialCapacityPlanning.xls.

- Column A of the spreadsheet lists cell names that are associated with the values in column B.
- Column C describes the value or computation associated with the value in column B.
- Cell names in red represent values that are required. Green cell names denote optional inputs; and the default values supplied in the spreadsheet
- All other cells are computed in the spreadsheet by using the inbuilt formulas.

After filling in the required inputs:

- The cell *StoreMachines* value will indicate how many Storage Nodes should be available in the storage node pool.
- The *StorePartitions* value indicates you how many partitions must be specified when creating the store.

Note: These computations yield estimates alone.

Planning Spreadsheet Sections

The InitialCapacityPlanning Spreadsheet has two main sections:

1. Shard capacity
 - Application characteristics
 - Hardware characteristics
 - Shard storage capacity
 - Shard I/O throughput capacity
 - Machine physical memory
 - Machine network throughput
2. Store sizing



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The InitialCapacityPlanning spreadsheet helps you to plan your installation by looking at the configuration details in two steps.

- First, you identify the configuration of a single shard based on the application requirements and the hardware available to deploy the KVStore.
- Second, you identify the number of shards required to fulfill the storewide configuration requirements.

Next Topic

- Plan for an Installation
- Install and Configure Nodes
- Deploy a KVStore



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Configuration Requirements

To configure a KVStore, you must have answers to the following questions:

- How many shards are required?
- How many partitions are required?
- What is the replication factor?
- How many storage nodes are required?



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To configure a KVStore, you need to know the number of shards and partitions required, the replication factor, and the total number of storage nodes.

As discussed in the previous section, much of this information may be derived from the planning spreadsheet.

Note: In the current release of Oracle NoSQL Database, you cannot add storage nodes to a KVStore after it has been configured. Therefore, it is important to carefully estimate the configuration requirements. It is recommended that you configure the KVStore so that there is room for data to grow.

Determine the Number of Shards

```
# of Shards =  
((((avg key size * 2) +  
avg value size) * max kv pairs) * 2) +  
(avg key size * max kv pairs) / 100 ) /  
(node storage capacity)
```

```
((10*2+1000) * (1*10^9)) * 2)  
+ (10 * (1*10^9))/100) / (1*10^12)  
= 2 Shards
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A *shard* is a storage element that contains:

- One master node that handles all the write requests
- One or more replica nodes that handle all read requests

In general, a KVStore with more shards will be better at servicing write requests because the write operations will be distributed across many replication nodes. Therefore, if an Oracle NoSQL Database application requires high throughput on data writes (creations, updates, and deletions), you should configure your store with more shards. The required number of shards also depends on the disk space available on a per-node basis.

Formula for Number of Shards

The slide shows a formula that you can use to obtain an estimate of the required number of shards. A description of the formula is in the top box, and an example of the formula is in the lower code box.

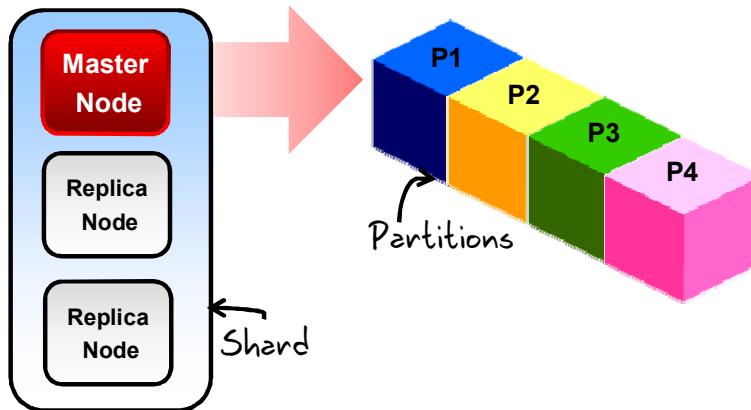
In the example, the KVStore is sized to hold:

- A maximum of 1 billion key-value pairs
- An average key size of 10 bytes and an average value size of 1000 bytes
- 1 TB of storage is available at each node

By applying the formula, you estimate that the KVStore requires two shards.

Determine # of Partitions and Replication Factor

- Number of partitions > maximum number of shards in KVStore
- Number of partitions cannot be changed once specified.
- Replication factor determines number of nodes in a shard.
- Replication factor = 1 + (number of replica nodes in shard)



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A shard also contains one or more partitions. The key-value pairs are evenly distributed across these partitions.

A shard should have at least one partition. For better performance, you may configure a KVStore with many partitions. You specify the number of partitions during configuration, and you cannot change the value at a later time.

It is recommended that you select the number of partitions to be greater than the total number of shards the KVStore will ever contain. There is minimal overhead involved in configuring a large number of partitions. Thus, you can set the number of partitions to be 100 times the maximum number of shards in a KVStore.

Determine # of Storage Nodes

```
Total nodes required = (number of shards) * (RF) /  
(number of disks per node)
```

The identified storage nodes should have:

- Supported operating system
- Sufficient disk space
- Similar configuration



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You estimate the total number of storage nodes needed for a KVStore by multiplying the number of shards you require by the replication factor (RF) and then dividing it by the number of disks per node (assuming the number of disks would be the same across the nodes).

The hard disks in each of these storage nodes should deliver enough input/output operations per second (IOPS) to meet your throughput requirements. Otherwise, you might need to increase the RF or the number of shards.

Installation and Configuration Steps

To set up the nodes for Oracle NoSQL Database:

1. Create the required directories.
2. Extract the Oracle NoSQL Database software.
3. Verify the installation.
4. Configure the replication nodes by using the makebootconfig utility.

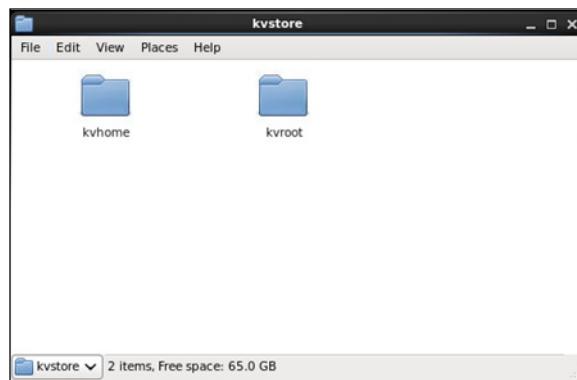


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Next, you learn to perform the steps listed in the slide.

Step 1: Creating Directories

- KVHOME: Storage location for Oracle NoSQL Database package files
- KVROOT: Storage location for Oracle NoSQL Database data files



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

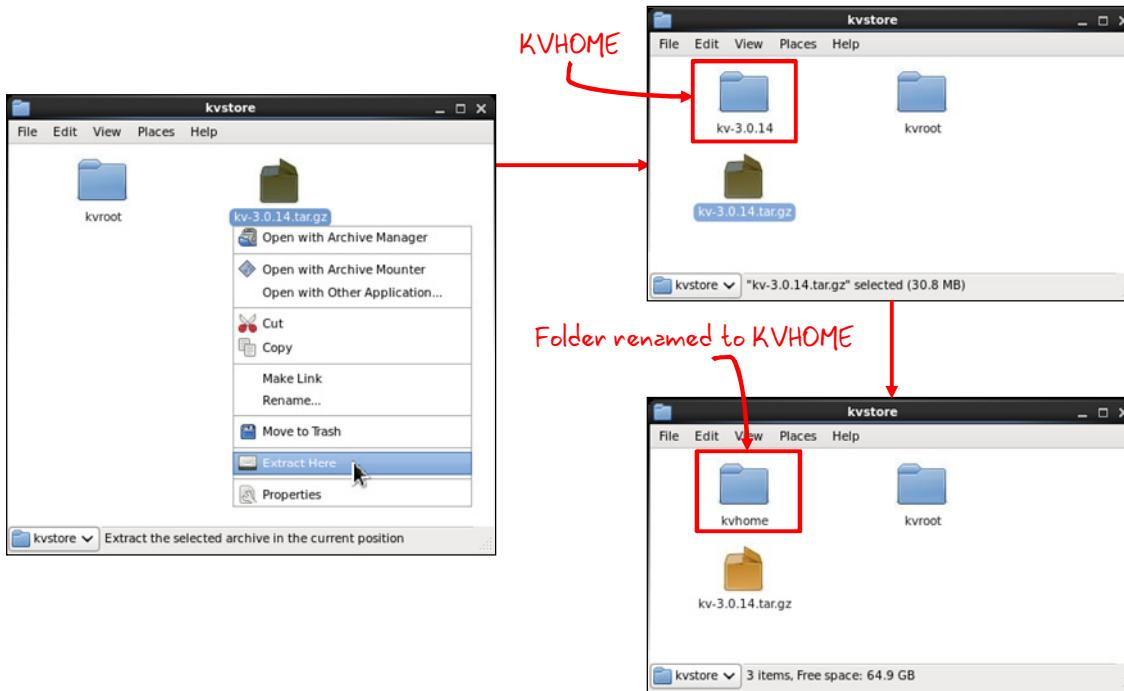
The storage location for the Oracle NoSQL Database package files is KVHOME. The package files contain the libraries, scripts, Javadoc, and other components of Oracle NoSQL Database. This is the directory that is created when you extract the Oracle NoSQL Database software zip file.

The storage location for the Oracle NoSQL Database data is KVROOT.

The KVROOT directory should be different from the KVHOME directory. Also, it is recommended that you keep the paths the same on all storage nodes. The KVHOME and KVROOT directories should be local to the storage nodes and not on a shared network system.

Note: Use the Linux `mkdir` command to create the KVROOT directory. Do not create the KVHOME directory because it is created automatically when the Oracle NoSQL Database software zip file is extracted.

Step 2: Extracting Software



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

After identifying the location for the KVHOME and KVROOT directories on the storage nodes, you can extract the Oracle NoSQL Database software zip file. You should extract the software to each storage node that has been identified for configuring a KVStore.

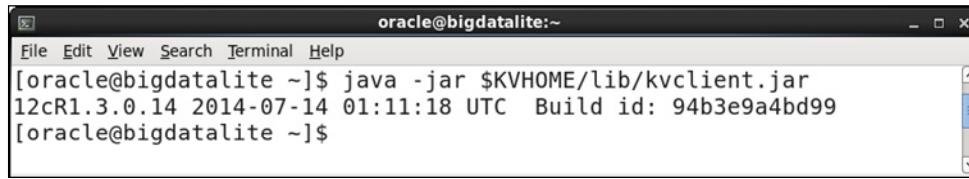
In this course, the software is extracted in the /home/oracle/kvstore directory. The kv-3.1.7 folder is created when the software is extracted. For easy reference to the directory, you rename kv-3.1.7 to KVHOME.

Extracting the software installs Oracle NoSQL Database in the storage node. You can now configure replication nodes in the storage node.

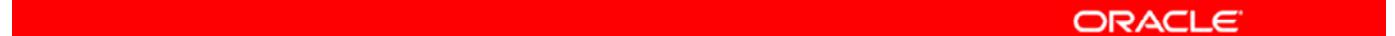
Step 3: Verifying the Installation

The following command is used to verify the installation of Oracle NoSQL Database in a storage node:

```
java -jar $KVHOME/lib/kvclient.jar
```

A screenshot of a terminal window titled "oracle@bigdatalite:~". The window shows the command "java -jar \$KVHOME/lib/kvclient.jar" being run, along with its output: "12cR1.3.0.14 2014-07-14 01:11:18 UTC Build id: 94b3e9a4bd99".

```
[oracle@bigdatalite ~]$ java -jar $KVHOME/lib/kvclient.jar
12cR1.3.0.14 2014-07-14 01:11:18 UTC Build id: 94b3e9a4bd99
[oracle@bigdatalite ~]$
```

The Oracle logo, consisting of the word "ORACLE" in white capital letters on a red background.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You run the Java command shown in the slide to verify that the Oracle NoSQL Database is successfully installed in the storage node. The output shows the version of the Oracle NoSQL Database installed.

Step 4: Configuring Nodes (Using the makebootconfig Utility)

```
java -jar KVHOME/lib/kvstore.jar makebootconfig
  -root <rootDirectory>
  -host <hostname>
  -harange <startPort,endPort>
  -store-security configure|enable|none
  -port <port>
  [-admin <adminPort>] [-runadmin] [-config <configFile>]
  [-storagedir <directory path>] [-capacity <n_rep_nodes>]
  [-num_cpus <ncpus>] [-memory_mb <memory_mb>]
  [-servicerange <startPort,endPort>]
  [-hahost <haHostname>]
  [-semdir <security dir>]
  [-pwdmgr {pwdfile | wallet | <class-name>}]
  [-kspwd <password>] [-param <param=value>]*
  [-mgmt {snmp|jmx|none}] [-pollport <snmp poll port>]
  [-traphost <snmp trap/notification hostname>]
  [-trapport <snmp trap/notification port>]
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

After you install Oracle NoSQL Database in the storage nodes, you can configure one or more replication nodes on each storage node. (It is recommended that you configure only one replication node in a single storage node.) To configure a replication node, you use the makebootconfig utility. The makebootconfig command is shown in the slide. You run this command on each storage node, for every replication node you want to create, with the appropriate parameter values. The parameters highlighted in red are mandatory.

- -root: You should have a KVROOT directory created on each storage node.
- -host: You should know the network name of each storage node that you use to set up the KVStore.
- -harange: You should have a sequential range of available ports on each storage node. These ports are used by the replication nodes to communicate with each other. You should have at least the same number of ports as the number of replication nodes on each storage node. Again, it is recommended that you use the same ports for each storage node.
- -store-security: This parameter states whether security will be used or not.

- -port: You should have an available TCP/IP port on each storage node. This port is used to contact the Oracle NoSQL Database services. It is recommended that you use the same port number on each storage node.
- -admin: You should have an available TCP/IP port on storage nodes where you want to configure the administration services. This port is used to contact the web-based Admin Console. It is recommended that you replicate the administration services in a KVStore. You need to have at least one administration service set up during this initial configuration. You can then create administration service replicas at a later time. You specify the -admin option only for the replication node where you want to enable administration services.

When you run this command, an initial “boot config” configuration file is created. You can explicitly specify the name for this configuration file by using the -config option. This is recommended when you have multiple replication nodes in a single storage node.

Using the makebootconfig Utility

```
java -jar KVHOME/lib/kvstore.jar makebootconfig  
  -root <KVROOT>  
  -port <registry port>  
  -admin <admin port>  
  -host <hostname>  
  -harange <high availability port range>  
  -memory_mb <memory_mb>  
  -store-security configure|enable|none  
  -config <configuration file> (Optional)
```

Syntax

```
java -jar $KVHOME/lib/kvstore.jar makebootconfig  
  -root $KVROOT  
  -port 5000  
  -admin 5001  
  -host localhost  
  -harange 5002,5009  
  -memory_mb 256  
  -store-security none  
  -config node1.xml
```

Example

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

After you identify all the configuration parameters, you use the `makebootconfig` utility to configure the replication nodes. The `makebootconfig` command is shown in the slide.

You run this command on each storage node, for every replication node you want to create, with the appropriate parameter values. You specify the `-admin` option only for the replication node where you want to enable administration services. The slide shows an example command for one of the replication nodes created in this course.

Starting the Storage Node Agents

The command to start an SNA is as follows:

```
nohup
java -jar KVHOME/lib/kvstore.jar start
-root <KVROOT>
-config <configuration file> optional
&
```

A screenshot of a terminal window titled "oracle@bigdatalite:~". The window shows the command "nohup java -jar \$KVHOME/lib/kvstore.jar start -root \$KVROOT &" being run. The output indicates that the process ID is 20554 and that nohup is ignoring input and appending output to 'nohup.out'. The terminal prompt "[oracle@bigdatalite ~]\$ |" is visible at the bottom.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A storage node agent (SNA) manages the Oracle NoSQL Database processes on each replication node. You use the start utility to start an SNA and specify the KVROOT for the replication node that you want to start. Optionally, you can specify the configuration file by using the –config option. This is recommended when you have more than one replication node on a single storage node.

The SNA has to be started for each replication node before configuring the KVStore. Also, the SNAs should be up and running at all times for a KVStore to function properly. The commands highlighted in red are Linux commands. It makes the command run in the background, which ensures that the process is running even after the terminal is closed. It is recommended that you configure your storage nodes so that all the SNAs start automatically when the system boots up.

Pinging the Replication Nodes

```
java -jar KVHOME/lib/kvstore.jar ping  
-port <registry port>  
-host <hostname>
```

1

SNA at hostname: localhost, registry port: 5000 is not registered.
No further information is available

2

Could not connect to registry at localhost:5000: Connection refused to host: localhost; nested exception is:
java.net.ConnectException: Connection refused



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can verify that the SNAs are up and running by issuing the `ping` command (as shown in the slide).

- If the SNA is running, the output shown in example 1 is displayed. If you ping the node after the KVStore is configured, more information will be displayed.
- If the SNA cannot be contacted, the output shown in example 2 is displayed. In this case, the storage node might not be up and running, or it is not configured properly.

Next Topic

- Plan for an installation
- Configure a KVStore
- Deploy a KVStore

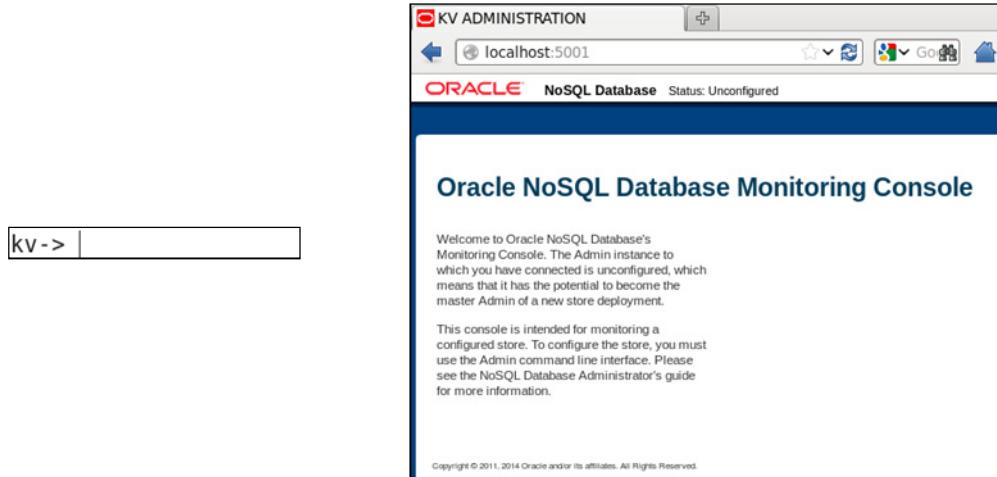


ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Configuration and Monitoring Tools

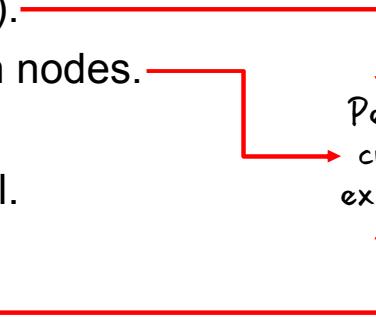
- The `runadmin` utility (command-line interface) is used to configure the KVStore.
- The Admin Console (web interface) is used to monitor the KVStore.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle NoSQL Database provides two tools: a command-line interface (CLI) and a web interface. You use the CLI tool to configure the KVStore and the web interface to monitor the KVStore. The KVStore can also be monitored by using the CLI, but to have a graphical view of the KVStore, the web interface is preferred.

Steps to Deploy a KVStore

1. Start or open the configuration tool.
 2. Configure the store name.
 3. Create a zone (or data center).
 4. Deploy the storage and admin nodes.
 5. Create a storage pool.
 6. Join nodes to the storage pool.
 7. Create a Topology.
 8. Deploy the KVStore.
- 
- Performed by
creating and
executing plans



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The steps to configure and deploy a KVStore are listed in the slide. Regardless of the configuration tool you use, the steps are the same.

1. Open the configuration tool.
2. Specify a name for the store.
3. Deploy a zone (or a data center) that will hold your Replication Nodes.
4. After deploying the zone, you deploy the storage node and the admin nodes to the zone.
5. Create a storage pool. A storage pool is a logical unit grouping storage nodes together. As you can imagine, within one zone there can be lots of different storage nodes belonging to lots of KV stores; a storage pool helps to differentiate them.
6. Add the nodes to the storage pool.
7. Create a topology corresponding to the pool and specify the number of partitions you require.
8. Deploy the KVStore.

Most of the steps are performed by creating and executing plans. Plans are described in the next slide.

Introducing Plans

```
show plans [-id <id> | -last]
```

```
plan deploy-zone -name <zone name> -rf <replication  
factor> [-type [primary | secondary]] [-plan-name  
<name>] [-wait] [-noexecute] [-force]
```

```
plan deploy-sn -zn <id> | -znname <name> -host <host> -  
port <port> [-plan-name <name>] [-wait] [-noexecute]  
[-force]
```

```
plan deploy-admin -sn <id> -port <http port> [-plan-name  
<name>] [-wait] [-noexecute] [-force]
```

```
plan deploy-topology -name <topology name> [-plan-name  
<name>] [-wait] [-noexecute] [-force]
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A plan is a sequence of internal configuration steps that are performed when invoked. Plans are predefined; you cannot create new plans. For each administrative task that you want to perform in a KVStore, there is a corresponding plan.

When you create a plan, by default, the command-line prompt returns immediately, and the plan executes in the background. If you use the optional `-wait` flag for the `plan` command, the command-line prompt will return only when the plan execution has completed.

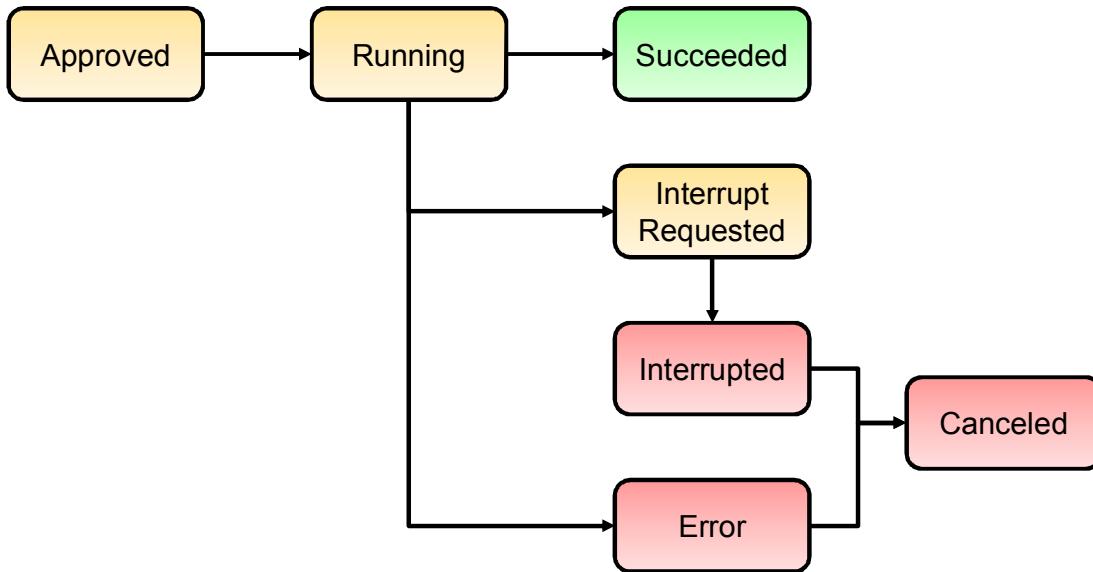
You can check the progress of the plans by using the `show plans` command or the `show plan -id <id>` command. Alternatively, you can see the state of your plans in the Plan History section in the Admin Console.

The following commands can be used to configure a KVStore:

- `deploy-zone`
- `deploy-sn`
- `deploy-admin`
- `deploy-topology`

A plan has various states, which are described in the next slide.

States of a Plan



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A plan can take the following states:

- APPROVED: When it has been created
- RUNNING: When it is executing
- SUCCEEDED: When it successfully completes the specified set of tasks
- INTERRUPT REQUESTED: When an interrupt command is passed using the CLI and the plan is waiting to be INTERRUPTED
- INTERRUPTED: When a RUNNING plan has been completely INTERRUPTED
- ERROR: When a RUNNING plan has encountered a problem and has ended without successfully completing
- CANCELED: When an INTERRUPTED or ERROR plan has been terminated using the cancel command in the CLI

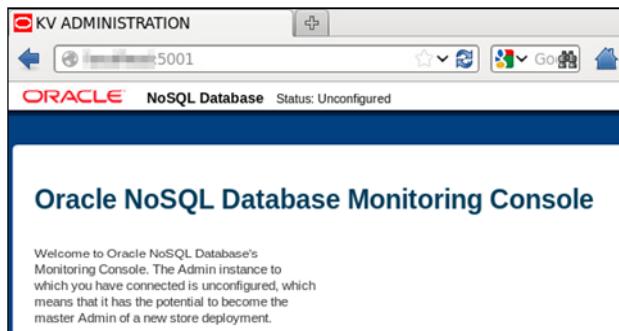
Plans in INTERRUPTED, INTERRUPT REQUESTED, or ERROR state can be retried by using the plan execute command.

Starting the Configuration Tool

Start the Configuration Tool.
Configure the store name.
Create a zone.
Deploy storage and admin nodes.
Create a storage pool.
Join nodes to storage pool.
Create a Topology.
Deploy KVStore.

```
java -jar KVHOME/lib/kvstore.jar runadmin \
-port <registry port> \
-host <hostname>

[oracle@bigdatalite ~]$ java -jar $KVHOME/lib/kvstore.jar runadmin \
> -port 5000 \
> -host [REDACTED]
kv-> |
```



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The CLI can be started by entering the command shown in the slide in a terminal. The port you need to specify is the registry port (not admin port) number of the replication node where an admin instance has been configured.

To start the Admin Console, open a web browser and enter the following in the address bar:

`http://<hostname>:<port>`

Here, the hostname is the name of the machine where the admin instance has been configured. The port number to specify is the port where the admin instance has been configured (not the registry port).

Configuring KVStore

Start the Configuration Tool.
Configure the store name.
 Create a zone.
 Deploy storage and admin nodes.
 Create a storage pool.
 Join nodes to storage pool.
 Create a Topology.
 Deploy KVStore.

The screenshot shows the Oracle NoSQL Database Admin Console interface. At the top, there is a command-line window with the following text:

```
configure -name <storename>
kv-> configure -name demoStore
Store configured: demoStore
```

An arrow points from this command-line output down to the Admin Console interface below. The Admin Console has a header bar with tabs: Topology, Plan History, and Logs. The Topology tab is selected. The main area is titled "Topology Browser" and displays the following information:

- 0 RepNodes (0 Running, 0 Down, 0 Needing Attention)
- 0 StorageNodes (0 Running, 0 Down, 0 Needing Attention)

Below this, there is a table header with columns: Performance/Status, Zone, StorageNode, RepNode/Admin, and RepNode Shard. Under the "Performance/Status" column, there are three metrics: Avg latency, 95th %ile, and Throughput.

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

To configure a store name in the CLI, perform the following steps:

- Ensure that you are at the `kv` prompt.
- Enter `configure -name` followed by the name you want to give the KVStore. The example in the slide configures the KVStore name as `demoStore`.
- Press Enter. The `kv` prompt is returned.

The Admin Console enables you to monitor the KVStore at the same time. After you have configured the name for the KVStore using the CLI, you will notice that the name of the KVStore is updated on the Admin Console.

Creating a Zone

Start the Configuration Tool.
Configure the store name.
Create a zone.
Deploy storage and admin nodes.
Create a storage pool.
Join nodes to storage pool.
Create a Topology.
Deploy KVStore.

```
plan deploy-zone -name <zone name> -rf <replication
    factor> [-type [primary | secondary]] [-plan-name
    <name>] [-wait] [-noexecute] [-force]

kv-> plan deploy-zone -name "New York" -rf 3 -wait
Executed plan 1, waiting for completion...
Plan 1 ended successfully
```

The screenshot shows the Oracle NoSQL Database Admin Console interface. At the top, it displays 'ORACLE NoSQL Database KVStore Name: demoStore'. Below this is a navigation bar with tabs: 'Topology' (selected), 'Plan History', and 'Logs'. Under the 'Topology' tab, there's a section titled 'Topology Browser' with the sub-section 'Performance/Status'. It shows '0 RepNodes (0 Running, 0 Down, 0 Needing Attention)' and '0 StorageNodes (0 Running, 0 Down, 0 Needing Attention)'. To the right of this section is a 'Verify Configuration' button. Below these sections is a table with columns: 'Zone', 'StorageNode', 'RepNode/Admin', and 'RepNode Shard'. The first row in the table is circled in red and contains the text 'New York' under the 'Zone' column. Below the table are three performance metrics: 'Avg latency', '95th %ile', and 'Throughput'.

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

To create a zone, in the CLI, create and execute a plan by entering the command shown in the slide. A plan number is displayed and the kv prompt returned. The name of the zone and the rf are mandatory parameters while the others are optional parameters.

Zones are of two types. Primary zones contain nodes that can serve as masters or replicas. Zones are created as primary zones by default. Secondary zones contain nodes that can serve only as replicas.

It is recommended to configure a KVStore across multiple zones because it provides fault isolation and availability for your data if a single zone fails. Each zone has a copy of your complete store, including a copy of all the shards. With this configuration, reads are always possible, so long as your data's consistency guarantees can be met, because at least one replica is located in every zone. Adding a zone increases the read throughput of the store.

You can verify that the zone is created by clicking the Topology tab in the Admin Console. The example in the slide configures the zone name as "New York."

Deploying Storage and Admin Nodes

Start the Configuration Tool.
 Configure the store name.
 Create a zone.
Deploy storage and admin nodes
 Create a storage pool.
 Join nodes to storage pool.
 Create a Topology.
 Deploy KVStore.

```

plan deploy-sn -zn <id> | -znname <name> -host <host> -  

  port <port> [-plan-name <name>] [-wait] [-noexecute]  

  [-force]  

plan deploy-admin -sn <id> -port <http port> [-plan-name  

  <name>] [-wait] [-noexecute] [-force]  

kv-> plan deploy-sn -znname "New York" -host [REDACTED] -port 5000  

Started plan 2. Use show plan -id 2 to check status.  

  To wait for completion, use plan wait -id 2  

kv-> plan deploy-admin -sn sn1 -port 5001  

Started plan 3. Use show plan -id 3 to check status.  

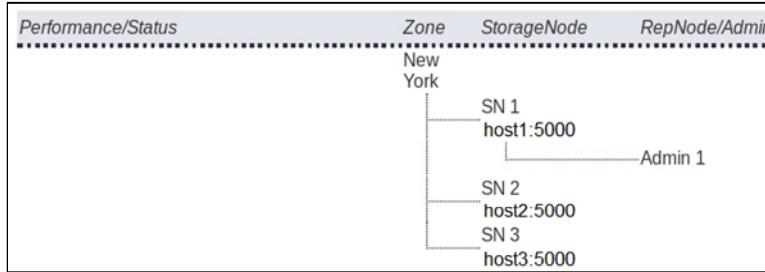
  To wait for completion, use plan wait -id 3  

kv-> plan deploy-sn -znname "New York" -host [REDACTED] -port 5000  

Started plan 4. Use show plan -id 4 to check status.  

  To wait for completion, use plan wait -id 4  

kv-> plan deploy-sn -znname "New York" -host [REDACTED] -port 5000
  
```



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To deploy all the storage nodes and admin nodes by using the CLI, create plans for each node by using the commands shown in the slide.

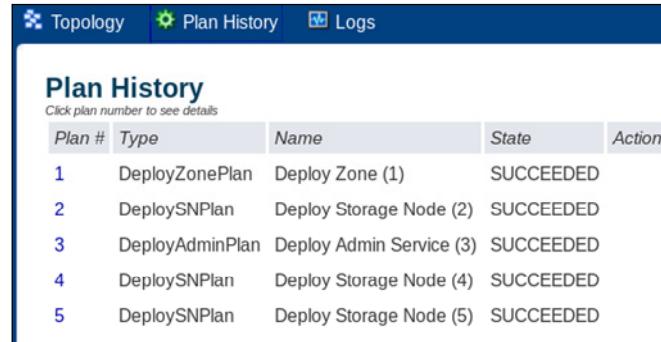
To deploy a storage node, specify the zone name or the zone ID, host name, and registry port of the node. If you do not know the zone ID, you can enter `show topology` in the `kv` prompt. The zone ID is the number after the letters `zn`.

To deploy an admin node, you need to specify the storage node ID and port where the admin process should run. You can run the `show topology` command to get the storage node ID. The number following the letters `sn` is the storage node ID.

Creating a Storage Pool

Start the Configuration Tool.
Configure the store name.
Create a zone.
Deploy storage and admin nodes.
Create a storage pool.
Join nodes to storage pool.
Create a Topology.
Deploy KVStore.

```
pool create -name <name>
kv-> pool create -name demoPool
```



The screenshot shows the Oracle Admin Console interface. At the top, there are three tabs: 'Topology' (selected), 'Plan History' (highlighted with a blue border), and 'Logs'. Below the tabs, the title 'Plan History' is displayed, followed by a sub-instruction 'Click plan number to see details'. A table lists five deployment plans:

Plan #	Type	Name	State	Action
1	DeployZonePlan	Deploy Zone (1)	SUCCEEDED	
2	DeploySNPlan	Deploy Storage Node (2)	SUCCEEDED	
3	DeployAdminPlan	Deploy Admin Service (3)	SUCCEEDED	
4	DeploySNPlan	Deploy Storage Node (4)	SUCCEEDED	
5	DeploySNPlan	Deploy Storage Node (5)	SUCCEEDED	



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To create a storage pool using the CLI, enter the `pool create -name` command followed by the name of the pool. The example in the slide creates a storage pool called `demoPool`.

You can view all the plans that you have executed by clicking the Plan History tab on the Admin Console.

Joining Nodes to the Storage Pool

Start the Configuration Tool.
Configure the store name.
Create a zone.
Deploy storage and admin nodes.
Create a storage pool.
Join nodes to storage pool.
Create a Topology.
Deploy KVStore.

```
pool join -name <name>
          -sn <storage node ID>

kv-> pool join -name demoPool -sn sn1
Added Storage Node(s) [sn1] to pool demoPool
kv-> pool join -name demoPool -sn sn2
Added Storage Node(s) [sn2] to pool demoPool
kv-> pool join -name demoPool -sn sn3
Added Storage Node(s) [sn3] to pool demoPool
```



A screenshot of a terminal window titled "oracle@bigdatalite:~". The window shows the following command and its output:

```
File Edit View Search Terminal Help
kv-> show pools
AllStorageNodes: sn1 sn2 sn3
demoPool: sn1 sn2 sn3
kv-> |
```

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To add storage nodes to the storage pool by using the CLI, enter the `pool join -name` command followed by the name of the pool and the ID of the storage node that you want to add. You run this command for each storage node you want to add to the storage pool.

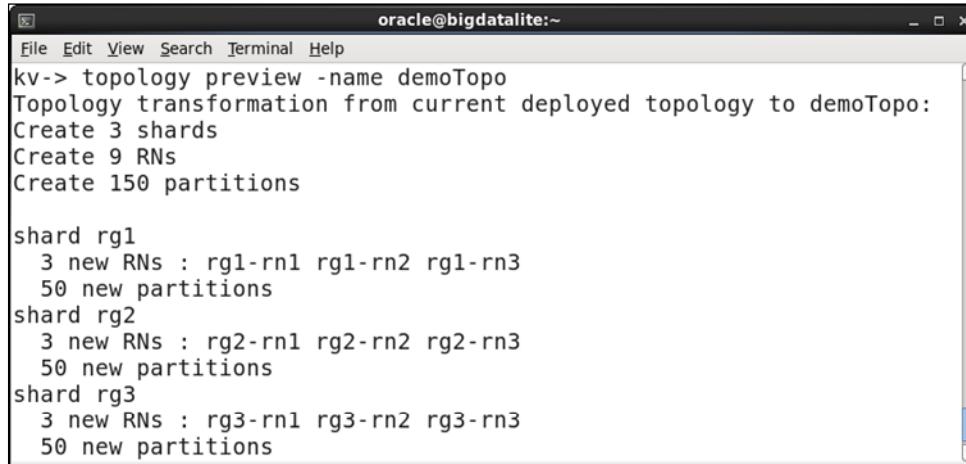
There is a pre-existing storage pool called the `AllStorageNodes`. This pool consists of all the storage nodes in the KVStore. The “Creating a storage pool” and “Joining nodes to storage pool” steps can be skipped if you want to include all the nodes in the KVStore when deploying the store.

Use the `show pools` command to see a list of storage pools present in the KVStore along with the storage nodes added to those pools.

Creating a Topology

```
Start the Configuration Tool.  
Configure the store name.  
Create a zone.  
Deploy storage and admin nodes.  
Create a storage pool.  
Join nodes to storage pool.  
Create a Topology.  
Deploy KVStore.
```

```
topology create -name <candidate name>  
    -pool <pool name> -partitions <num>  
  
kv-> topology create -name demoTopo -pool demoPool -partitions 150  
Created: demoTopo
```



A screenshot of a terminal window titled "oracle@bigdatalite:~". The window displays the command "kv-> topology create -name demoTopo" followed by its output. The output shows the creation of a topology named "demoTopo" from the current deployed topology. It details the creation of 3 shards, 9 RNs, and 150 partitions. It also lists the shard assignments for each shard, indicating 3 new RNs per shard and 50 new partitions per shard.

```
File Edit View Search Terminal Help  
kv-> topology create -name demoTopo  
Topology transformation from current deployed topology to demoTopo:  
Create 3 shards  
Create 9 RNs  
Create 150 partitions  
  
shard rg1  
3 new RNs : rg1-rn1 rg1-rn2 rg1-rn3  
50 new partitions  
shard rg2  
3 new RNs : rg2-rn1 rg2-rn2 rg2-rn3  
50 new partitions  
shard rg3  
3 new RNs : rg3-rn1 rg3-rn2 rg3-rn3  
50 new partitions
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

After creating a pool, you create a topology with the specified number of partitions using the specified storage pool. Remember to choose an appropriate value for partitions as it is a static value and cannot be changed later.

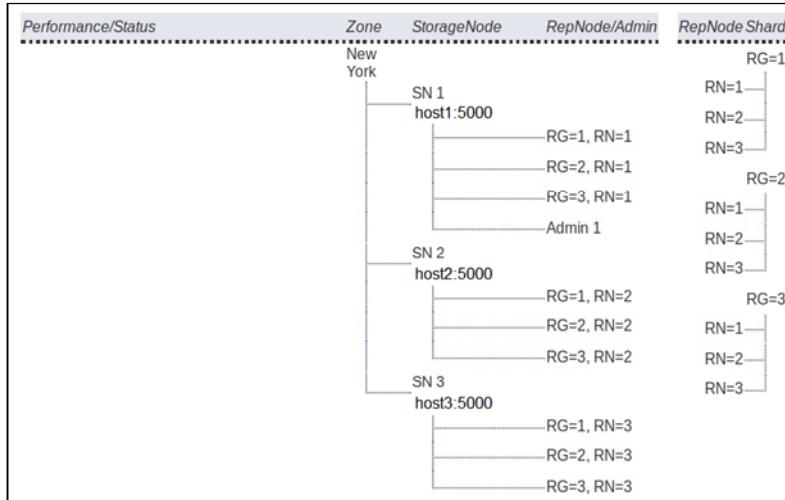
Before deploying the KVStore, you can preview the topology by using the `topology preview -name` command. The output shows you the arrangement of replication nodes into shards and the number of partitions allotted to each shard.

The example in the slide creates a topology called `demoTopo`.

Deploying the KVStore

Start the Configuration Tool.
 Configure the store name.
 Create a zone.
 Deploy storage and admin nodes.
 Create a storage pool.
 Join nodes to storage pool.
 Create a Topology.
Deploy KVStore.

```
plan deploy-topology -name <topology name> [-plan-name <name>] [-wait] [-noexecute] [-force]
kv-> plan deploy-topology -name demoTopo -wait
Executed plan 6, waiting for completion...
Plan 6 ended successfully
```



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To deploy a KVStore by using the CLI, you create a plan (as shown in the slide). To deploy the topology, you need to specify the `plan deploy-topology -name` command followed by the name of the topology that you created.

This example deploys the `demoTopo` topology.

Testing the KVStore

```
java -jar KVHOME/lib/kvstore.jar ping  
-port <registry port>  
-host <hostname>
```

```
oracle@bigdatalite:~  
File Edit View Search Terminal Help  
[oracle@bigdatalite ~]$ java -jar $KVHOME/lib/kvstore.jar  
ping -port 5000 -host [REDACTED]  
Pinging components of store demoStore based upon topology  
sequence #166  
Time: 2014-10-31 09:26:55 UTC  
demoStore comprises 150 partitions and 3 Storage Nodes  
Storage Node [sn1] on [REDACTED]:5000 Zone: [name=New Y  
ork id=zn1 type=PRIMARY] Status: RUNNING Ver: 12cR1.  
3.0.14 2014-07-14 01:11:18 UTC Build id: 94b3e9a4bd99  
Rep Node [rg1-rn1] Status: RUNNING,REPLICA a  
t sequence number: 131 haPort: 5006  
Rep Node [rg2-rn1] Status: RUNNING,REPLICA a  
t sequence number: 131 haPort: 5007  
Rep Node [rg3-rn1] Status: RUNNING,MASTER at  
sequence number: 127 haPort: 5008  
Storage Node [sn2] on [REDACTED] Zone: [name=New Y  
ork id=zn1 type=PRIMARY] Status: RUNNING Ver: 12cR1.  
3.0.14 2014-07-14 01:11:18 UTC Build id: 94b3e9a4bd99
```

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You can confirm whether a KVStore is configured successfully by running the `ping` command, as shown in the slide.

You can also compile and run the simple “Hello World” example present in the KVHOME directory. If the result is displayed, the store is configured successfully.

If you encounter installation problems, stop the roots, remove the content of the roots, and reinstall and configure the nodes by using the same steps.

Additional Resources

- Oracle University 2-day course:
Oracle NoSQL Database for Administrators
(GCC: D77998GC20)



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Plan for an installation
- Install and configure nodes
- Deploy a KVStore



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

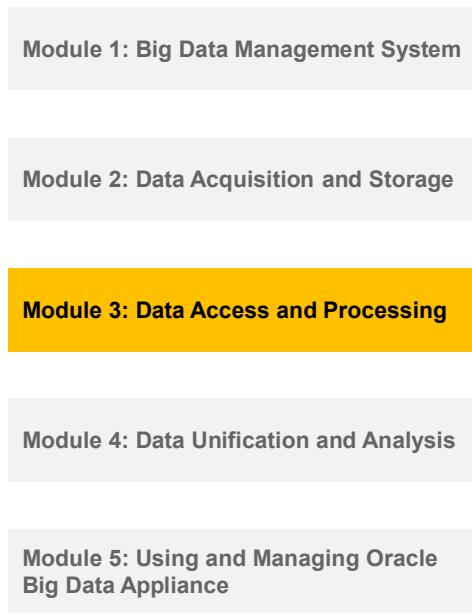
9

Introduction to MapReduce

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



Lesson 9: Introduction to MapReduce

Lesson 10: Resource Management Using YARN

Lesson 11: Overview of Apache Hive and Apache Pig

Lesson 12: Overview of Cloudera Impala

Lesson 13: Using Oracle XQuery for Hadoop

Lesson 14: Overview of Solr

Lesson 15: Apache Spark

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This lesson introduces the MapReduce framework that enables you to write applications that process vast amounts of data, in-parallel, on large clusters of commodity hardware, in a reliable and fault-tolerant manner. In addition, this lesson describes the architectural components of MapReduce and lists the benefits of using MapReduce.

Objectives

After completing this lesson, you should be able to describe the MapReduce process.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

MapReduce

It is a software framework that enables you to write applications that process vast amounts of data, in-parallel on large clusters of commodity hardware in a reliable and fault-tolerant manner.

- Prior to Hadoop 2.0, MapReduce was the only way to process data in Hadoop.
- A MapReduce job usually splits the input data set into independent chunks, which are processed by the map tasks in a completely parallel manner.
- The framework sorts the outputs of the maps, which are then input to the reduce tasks.
- Typically, both the input and the output of the job are stored in a file system.
- *The framework takes care of scheduling tasks, monitors them, and re-executes the failed tasks.*



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

MapReduce Architecture

- Master-slave architecture
- Storing data in HDFS is low cost, fault-tolerant, and easily scalable, to just name a few.
- *MapReduce integrates with HDFS to provide the exact same benefits for parallel data processing.*
- Sends computations where the data is stored on local disks
- Programming model or framework for distributed computing. It hides complex "house keeping" tasks from you as developer.



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

MapReduce Version 1 (MRv1) Architecture

- Typically, MapReduce (compute) framework and the HDFS (storage) are running on the same set of nodes:
 - Allows the framework to effectively schedule tasks on the nodes where data is stored, data locality, which results in better performance
- The MapReduce 1 framework consists of:
 - One master `JobTracker` daemon per cluster
 - One slave `TaskTracker` daemon per cluster-node
- The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them, and re-executing the failed tasks.
- The slaves execute the tasks requested by the master.
- **MRv1 runs only MapReduce jobs.**



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

MapReduce Phases

- Map
 - Each Map task usually works on a single HDFS block (*input split*).
 - Map tasks run on the slave nodes in the cluster where the HDFS data block is stored (data locality).
 - The input presented to the Map task is a key-value pair.
- Shuffle and Sort
 - Sorts and consolidates intermediate output data from all of the completed mappers from the Map phase
- Reduce
 - The intermediate data from the Shuffle and Sort phase is the input to the Reduce phase.
 - The Reduce function (developer) generates the final output.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In Hadoop, files are made up of records that are processed later by the Mapper tasks in MapReduce. In HDFS, the default block size is 64 MB, which means that the data stored in a file are broken down into chunks of exactly 64 MB. A problem arises when the records in the file span block boundaries; that is, one record is contained in two or more HDFS blocks. HDFS has no idea of what is inside the file blocks and it cannot determine when a record might spill over into another block. To solve this problem, Hadoop uses a logical representation of the data stored in file blocks, known as input splits. When a MapReduce job client calculates the input splits, it determines where the first whole record in a block begins and where the last record in the block ends.

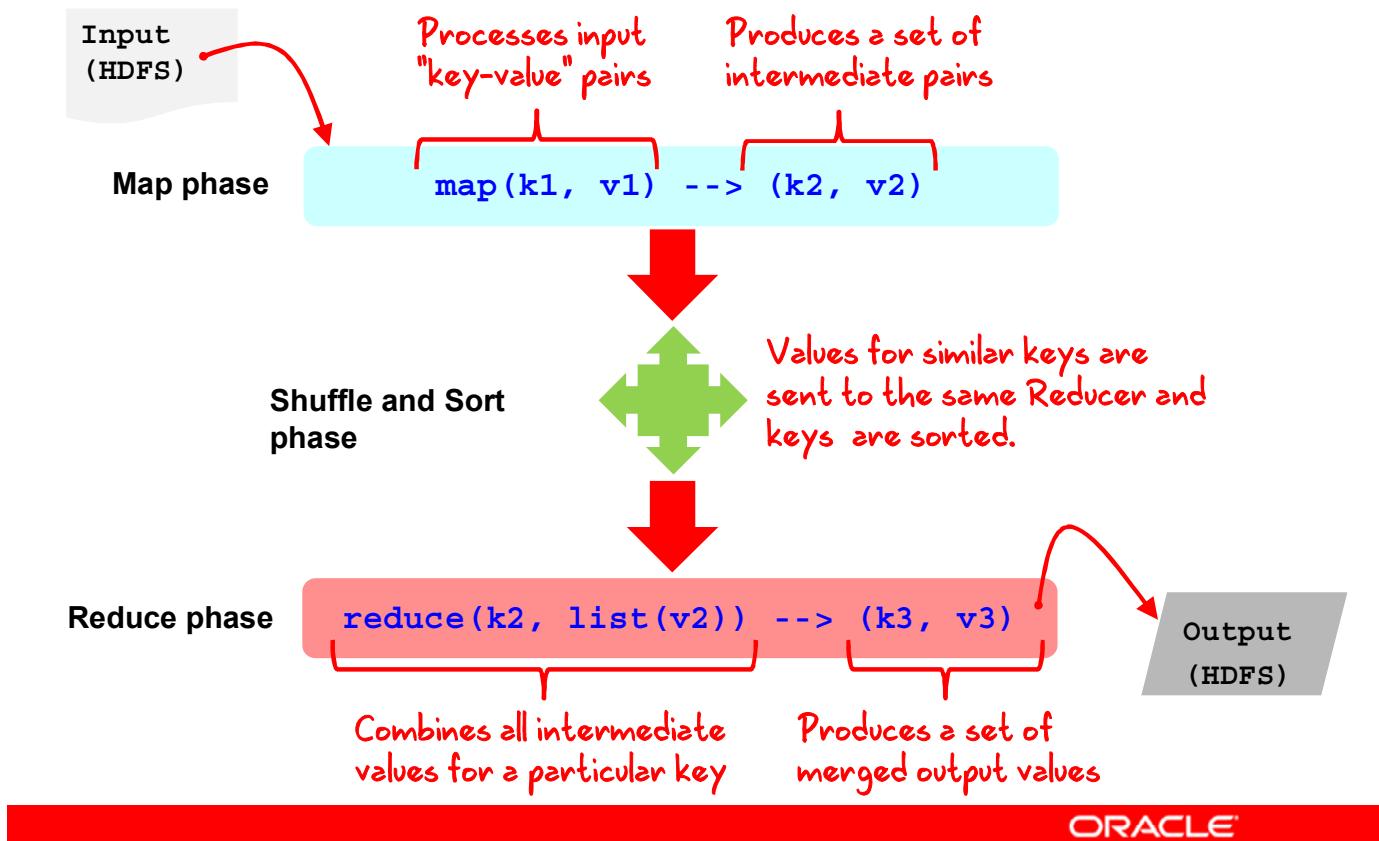
MapReduce Framework

- A software framework that enables you to write applications that will process large amounts of data, in-parallel, on large clusters of commodity hardware, in a reliable and fault-tolerant manner
- Integrates with HDFS and provides the same benefits for parallel data processing
- Sends computations to where the data is stored
- The framework:
 - Schedules and monitors tasks, and re-executes failed tasks
 - Hides complex "house keeping" and distributed computing complexity tasks from the developer



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Parallel Processing with MapReduce



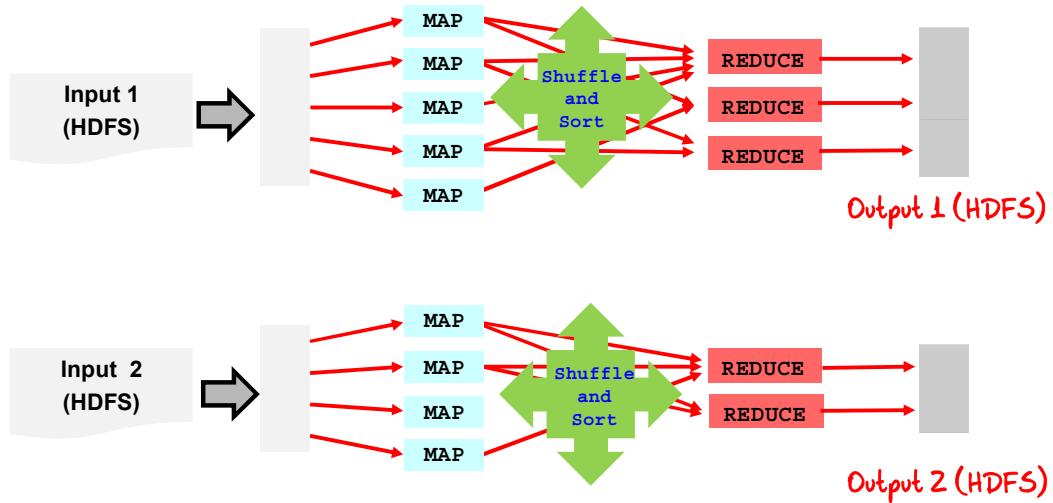
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A map function generates a series of key-value pairs from the input data. This data is then reduced by a function to combine all values that are associated with equivalent keys. Programs are automatically parallelized and executed on a runtime system that manages partitioning the input data, scheduling execution, and managing communication, including recovery from machine failures.

- The records are divided into smaller chunks for efficiency, and each chunk is executed serially on a particular compute engine.
- The output of the Map phase is a set of records that are grouped by the mapper output key. Each group of records is processed by a reducer (again, these are logically in parallel).
- The output of the Reduce phase is the union of all records that are produced by the reducers.

MapReduce Jobs



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hadoop divides the input to a MapReduce job into fixed-size pieces or "chunks" named *input splits*. Hadoop creates one map task (Mapper) for each split.

The Input split (usually an HDFS block) runs the user-defined map function for each *record* in the split.

Hadoop attempts to run the tasks where the data is located.

A mapper task works on one individual record (with a key-value) at a time and stores the intermediate data locally.

The framework shuffles and sorts the outputs of the maps before they become the input to the reducer tasks.

Typically both the input and the output of the job are stored in HDFS.

Interacting with MapReduce

- MapReduce code can be written in Java, C, and scripting languages.
- Higher-level abstractions (Hive, Pig) enable easy interaction.
 - Optimizers construct MapReduce jobs.
- Code is:
 - Submitted to the JobTracker daemons on the Master node
 - Executed by the TaskTrackers on the Slave nodes



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hadoop tries to run the TaskTrackers and DataNodes on the same servers. Hadoop does its best to run the map task on a node where the input data resides in HDFS. This is called the *data locality optimization* because it does not use valuable cluster bandwidth. Sometimes, however, all three nodes hosting the HDFS block replicas (as discussed in the earlier HDFS lesson) for a map task's input split are running other map tasks; therefore, the job scheduler will locate a free map slot on a node in the same rack as one of the HDFS blocks. Sometimes, this is not possible; therefore, an off-rack node is used, which results in an inter-rack network transfer.

MapReduce Processing

- Big Data implementations use either MRv1 or YARN.
 - MRv1 was the key processing framework prior to YARN.
 - YARN introduced both flexibility and scalability to the processing framework.
- This lesson uses MRv1 when describing MR processing.
 - Understanding MRv1 is useful for support purposes.
- YARN is covered in detail in the lesson titled “Resource Management Using YARN.”



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

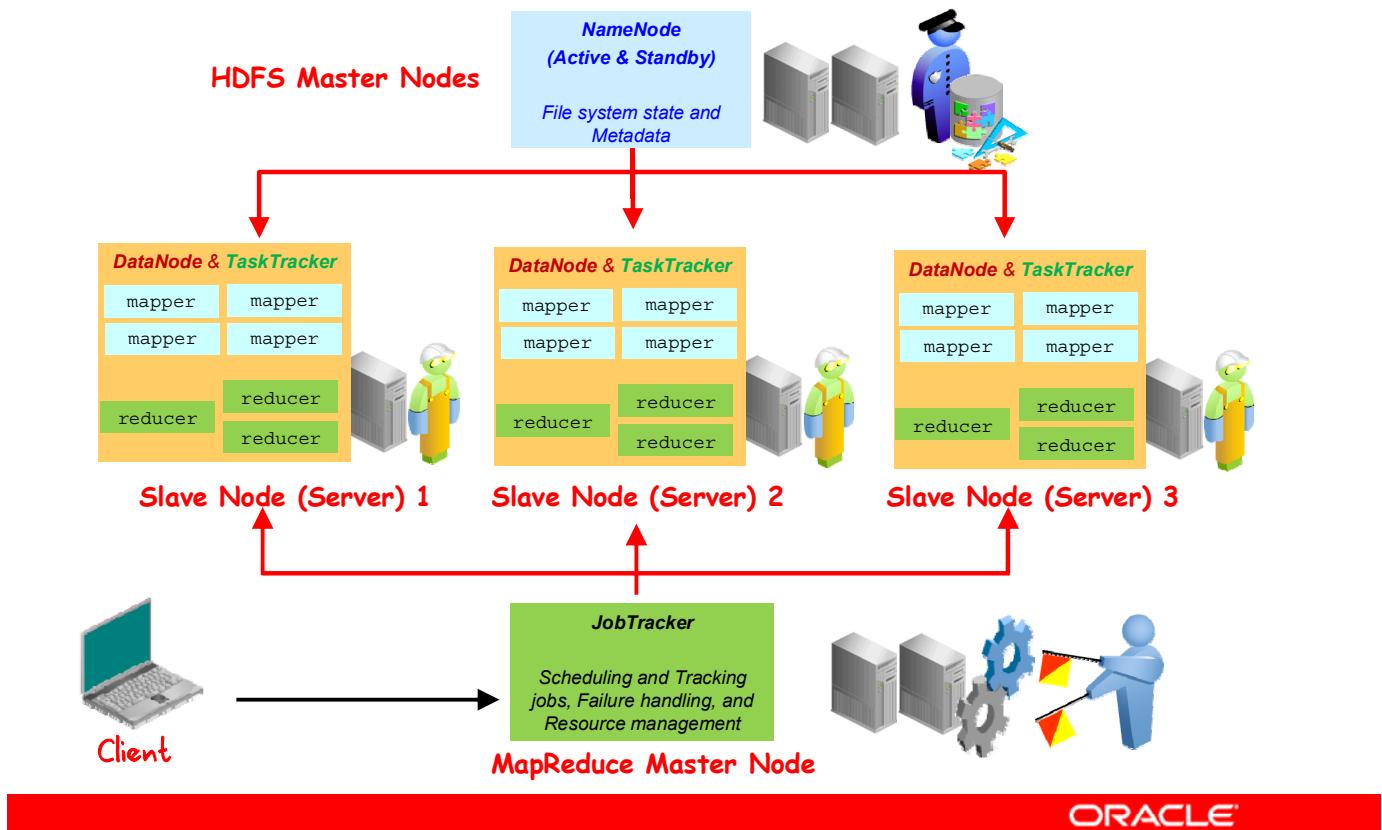
MapReduce (MRv1) Daemons

Component	Description
JobTracker (Master daemon)	<ul style="list-style-type: none">Accepts job submissions from clientsSchedules tasks to run on the slave nodesChecks the health and task progress of slave nodesThere is one JobTracker per MapReduce cluster.
TaskTracker (Slave daemons)	<ul style="list-style-type: none">Accepts task assignments from the JobTrackerInstantiates the user codeExecutes those tasks locally, and reports progress back to the JobTracker periodicallyThere is always a single TaskTracker on each slave node.Both TaskTrackers and DataNodes run on the same machines, which makes each node both a <i>compute node</i> and a <i>storage node</i>.
Map and Reduce Tasks	<ul style="list-style-type: none">Run MapReduce code on the slave nodes.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hadoop Basic Cluster (MRv1): Example



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

A Hadoop cluster is made up of Master and Slave nodes (physical servers) used for both distributed storage of massive data and distributed processing of data stored in the cluster. The nodes are servers, which are part of the rack in the cluster.

- On the HDFS side, the master nodes (servers) are usually higher-end machines with master daemons or services running on the master nodes; namely, the Active NameNode and the Standby NameNode. The NameNode holds the metadata about the cluster and manages the data storages in the cluster.
- On the MapReduce side, MapReduce is the Master Node that contains one Master daemon named the JobTracker. The JobTracker daemon manages the MapReduce jobs and distributes tasks (TaskTrackers) to the slave nodes in the cluster where the data is located.

The important thing to remember for now is that the storage and processing occurs on the slave nodes in the Hadoop cluster whereas managing both the storage and processing occurs on the master nodes in the Hadoop cluster.

MapReduce Application Workflow

1. The client application submits a job to the JobTracker (JT).
2. The JT determines the processing resources that are required to complete the entire job.
3. The JT checks the status of the slave nodes and queues all the map and reduce tasks for execution.
4. Map tasks are started on slave nodes when the map slots become available. The JT monitors task progress.
5. After the map tasks are finished, the Shuffle and Sort phase locally performs an intermediate reduction of the output of each mapper.
6. The reducers aggregate the Shuffle and Sort phase intermediate results to create the final single result.
7. The result set is returned to the client application and the resources are released.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The MapReduce applications have the following phases:

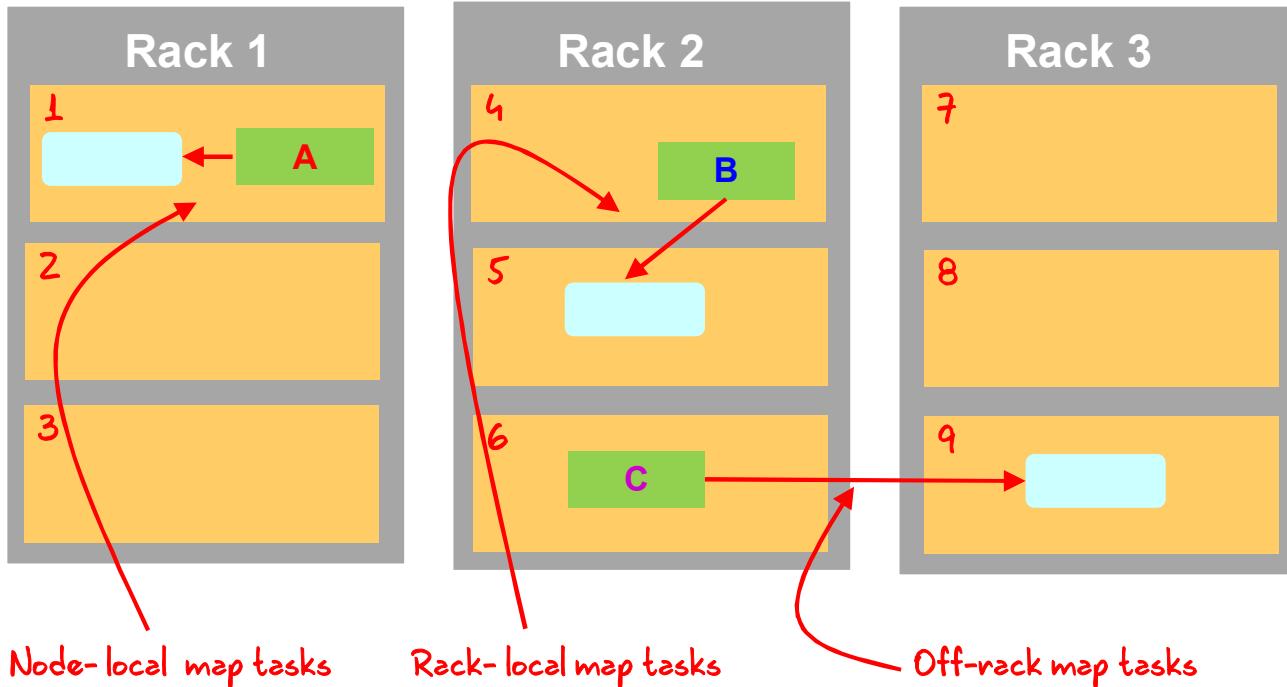
1. The client application submits a job to the JobTracker master node.
2. The JobTracker determines the processing resources that are required to complete the entire job. The JobTracker determines the exact data sets (file names, blocks location, nodes, and so on) required to process from the HDFS data blocks. This involves calculating where the records to be processed are located within the data blocks. This information is all obtained from the NameNode master node, which maintains the information. The JobTracker calculates the number of map and reduce tasks that will be needed to process all this data.
3. The JobTracker checks the status of the slave nodes and queues all the map and reduce tasks for execution.
4. Map tasks are started on slave nodes when the map slots become available. Map tasks assigned to specific blocks of data are assigned to nodes where the same data is stored. The mapper task is run on each record in the data set until all the records are processed. The output of all of the mappers are stored locally on the local file system (not HDFS). The JobTracker monitors task progress, and in the event of a task failure or a node failure, the task is restarted on the next available slot. If the same task fails after four attempts (default), the entire job fails.

5. After the map tasks are finished, the Shuffle and Sort phase locally performs an intermediate reduction of the output of each mapper.
6. The reducers aggregate the Shuffle and Sort phase intermediate results to create the final single result. The final output is stored in HDFS.
7. The result set is returned to the client application and the resources are released.

Data Locality Optimization in Hadoop

HDFS blocks:

Map tasks:



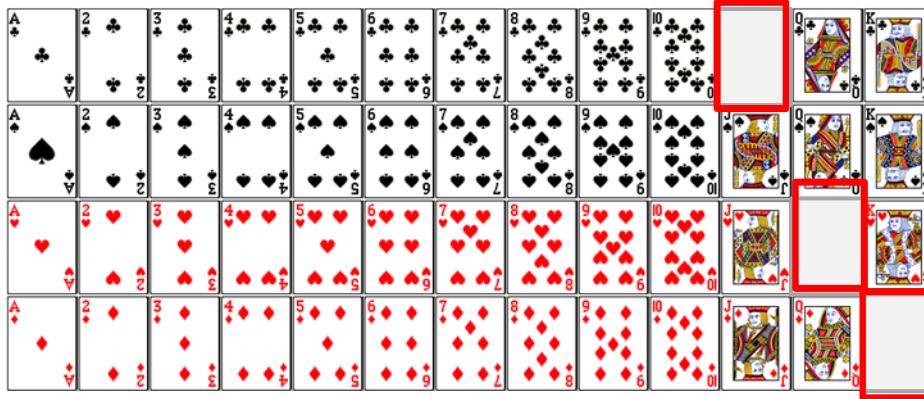
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

ORACLE

The diagram shown in the slide was presented in the earlier HDFS lesson, which illustrated the data replication rack-awareness in HDFS. Hadoop will try to collocate the data and tasks on the same nodes.

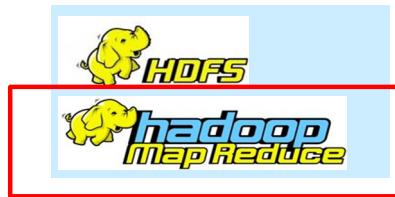
Hadoop tries to run the TaskTrackers and DataNodes on the same servers in the Hadoop cluster. Hadoop does its best to run the map task on a node that contains the HDFS block that is required by the Map task as shown in the first slide example. This is called the *data locality optimization* because it does not use valuable cluster bandwidth because the Map task runs on the same slave node where the data it needs is located. Sometimes, all three nodes hosting the HDFS block replicas (as discussed in the earlier HDFS lesson) for a map task's input split are running other map tasks; therefore, the job scheduler will locate a free map slot on a node in the same rack as one of the HDFS blocks as shown in the second slide example. If both situations are not possible, Hadoop will use an off-rack node in the cluster, which results in an inter-rack network transfer as shown in the third slide example.

MapReduce Mechanics: Deck of Cards Example



Suppose 3 face cards are removed from the deck of playing cards.

How do we find the suits whose face cards are short by using MapReduce?

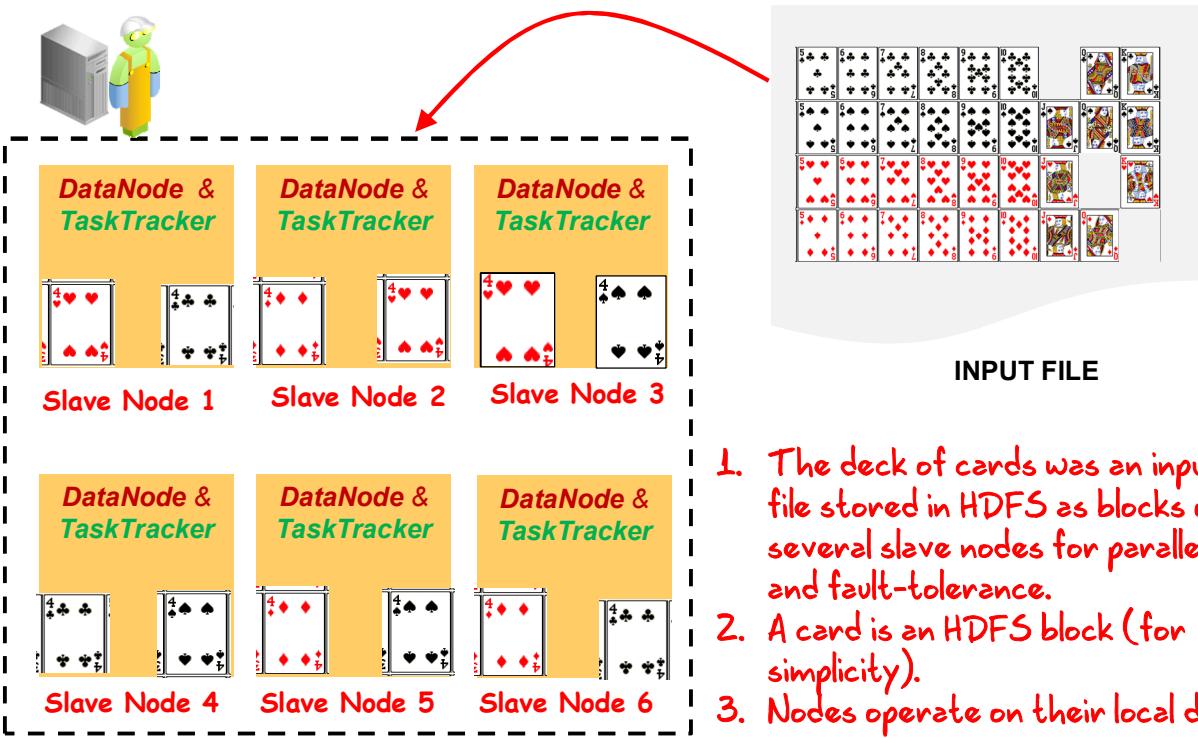


ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Suppose that three face cards are removed. How would you use MapReduce to determine which suits have missing cards? Assume that the deck of cards shown in the slide is the input data, which is already stored on slave nodes in HDFS.

MapReduce Mechanics Example: Assumptions



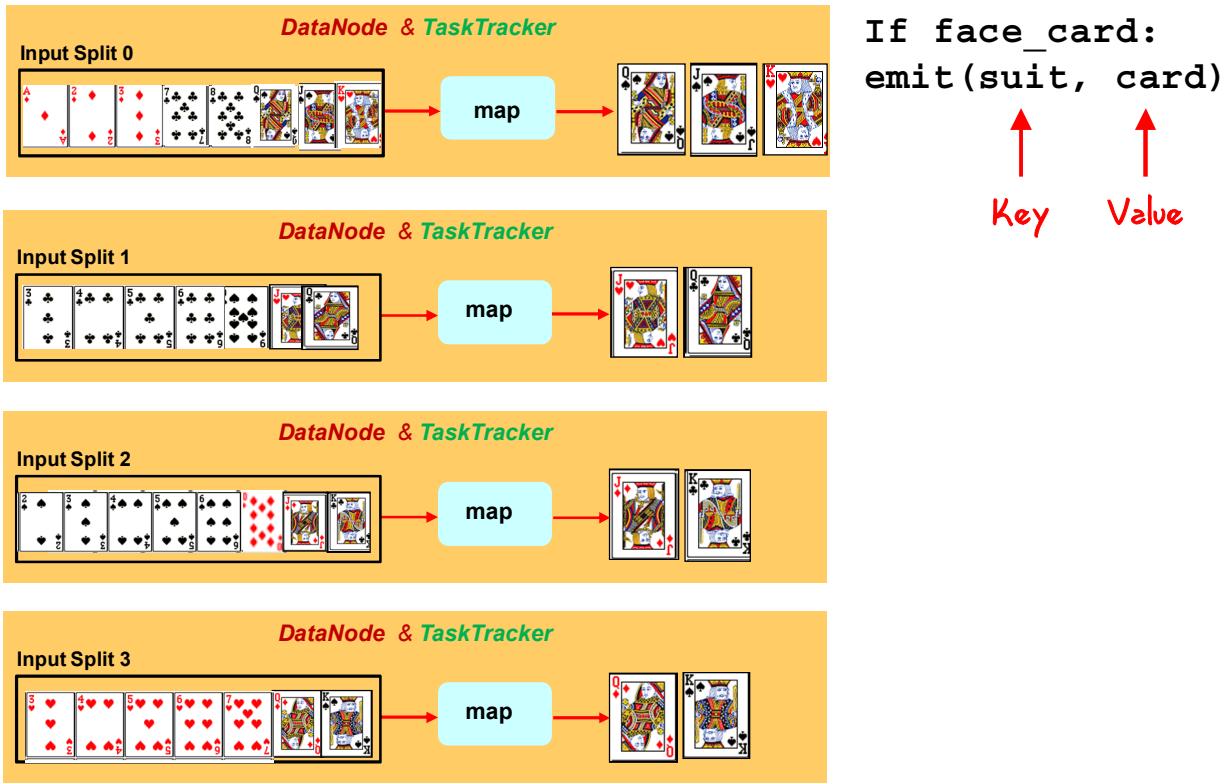
1. The deck of cards was an input file stored in HDFS as blocks on several slave nodes for parallelism and fault-tolerance.
2. A card is an HDFS block (for simplicity).
3. Nodes operate on their local data.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Assume that you have a big data file that contains the cards in a deck of playing cards. Assume that you have a set of DataNodes in the Hadoop cluster. The file will be broken up into blocks. Blocks are stored in multiple locations (DataNodes) in the cluster, which allows for parallelism and fault-tolerance. Nodes operate on their local data.

MapReduce Mechanics: The Map Phase



ORACLE

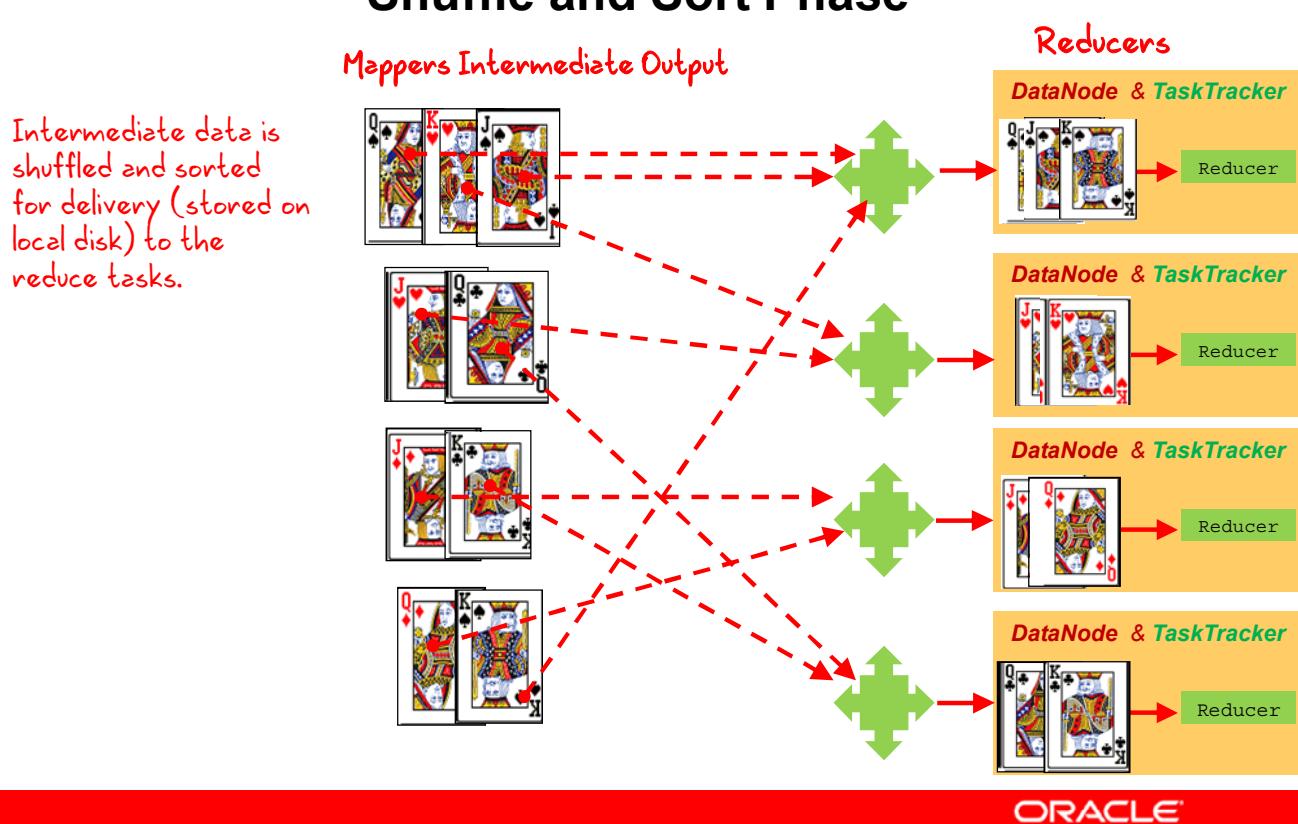
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Let us go back to the example of the (3) missing face cards from the deck of playing cards. Hadoop passes each TaskTracker that has some data local to it. Map tasks operate on this local data.

```
if face_card: emit(suit, card)
```

If the card that was just processed is a facecard, then emit the suite of the card, and the card itself.

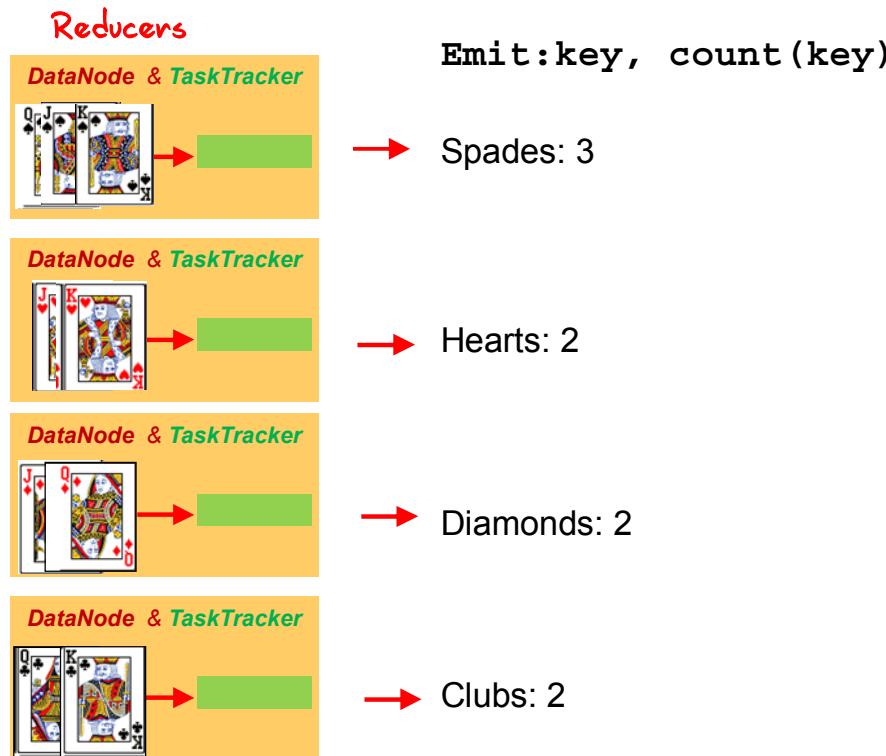
MapReduce Mechanics: The Shuffle and Sort Phase



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Intermediate data is shuffled and sorted for delivery to the Reduce tasks.

MapReduce Mechanics: The Reduce Phase

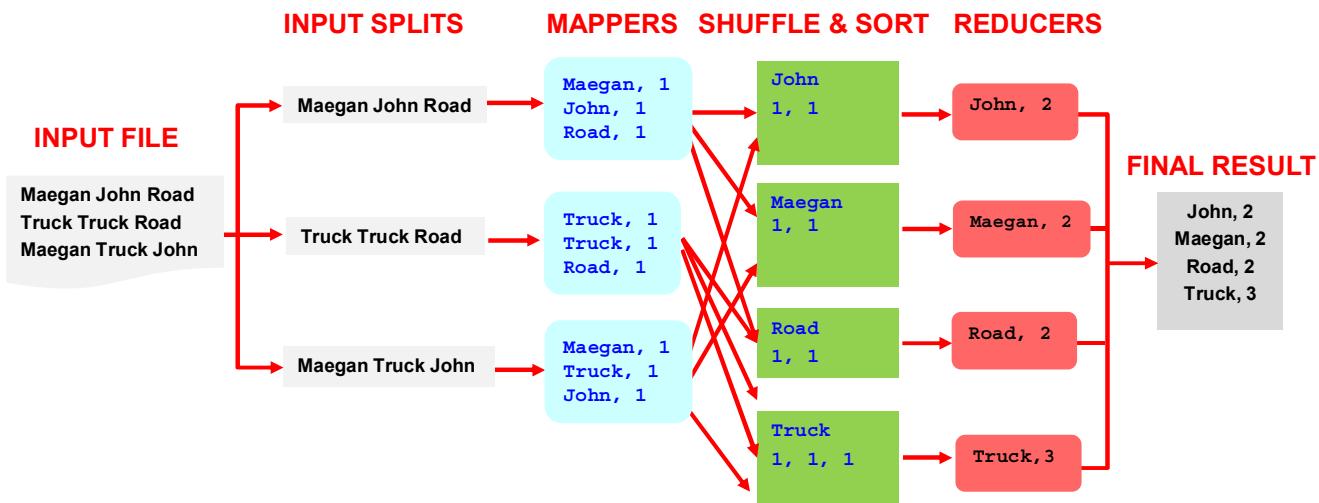
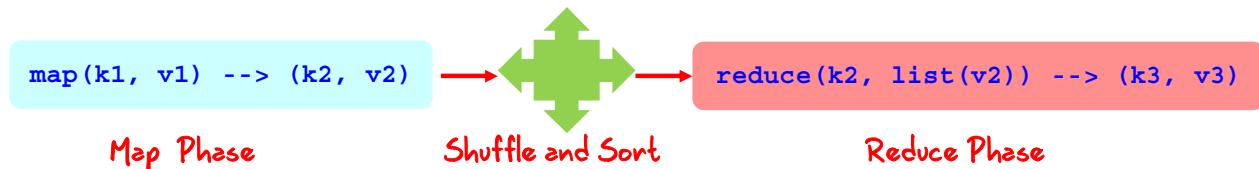


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Reducers operate on local data to produce the final result.

Word Count Process: Example



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The example in the slide shows the raw data in the form of a text file. The data is captured by using Flume or other systems. The data is stored, distributed, and replicated on different nodes in HDFS system as you learned earlier in this course.

The MapReduce process is started. Initially, a mapper task is started on the node that contains the data that is needed in the processing.

The MapReduce process is started. Initially, a mapper chooses a chunk (a record from the input split) to be processed. Each mapper emits a word with the number of times it was found. Each mapper finishes the initial processing and sends the output to the Shuffle and Sort phase.

The shuffler shuffles the data based on the similarity found, sorts the data, and then copies the sorted data as input to the reducers.

The reducer combines the mapper's output into a total.

Submitting a MapReduce

```
hadoop jar WordCount.jar WordCount  
  /user/oracle/wordcount/input  
  /user/oracle/wordcount/output
```

Code	Description
<code>hadoop jar</code>	Tells the client to submit job to the JobTracker
<code>WordCount.jar</code>	The jar file that contains the Map and Reduce code
<code>WordCount</code>	The name of the class that contains the main method where processing starts
<code>/user/oracle/wordcount/ input</code>	The input directory
<code>/user/oracle/wordcount/ output</code>	A single HDFS output path. All final output will be written to this directory.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned about the MapReduce process.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 9: Overview

In these practices, you will:

- Compile a WordCount . java program, which runs on a Hadoop Cluster
- Upload the files on which you run the WordCount into the Hadoop Distributed File System (HDFS)
- Run the WordCount . java program and view the results



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

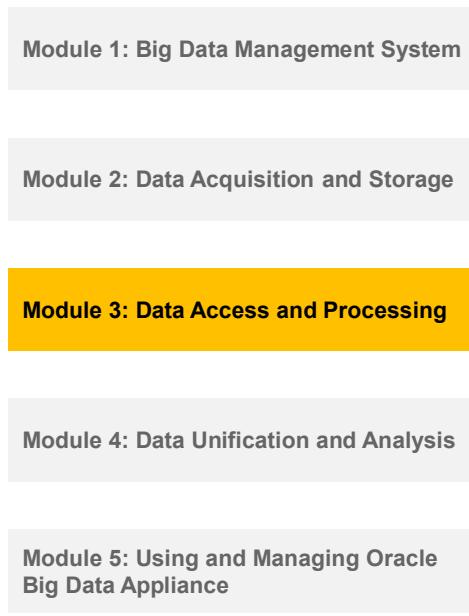
10

Resource Management Using YARN

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



Lesson 9: Introduction to MapReduce

Lesson 10: Resource Management Using YARN

Lesson 11: Overview of Apache Hive and Apache Pig

Lesson 12: Overview of Cloudera Impala

Lesson 13: Using Oracle XQuery for Hadoop

Lesson 14: Overview of Solr

Lesson 15: Apache Spark

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This lesson introduces YARN, yet another resource negotiator, which is the next generation MapReduce, or MR2. This lesson also covers the schedulers including FIFO, Capacity Scheduler, and Fair Share Scheduler (FSS). The lesson also shows an example of using the FSS in Cloudera Manager. Finally, introduces the `yarn application` command that you can use to list and kill applications.

Objectives

After completing this lesson, you should be able to use YARN to manage resources efficiently.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Agenda

- Apache Hadoop YARN architecture
- Job Scheduling in YARN
- YARN application command



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Apache Hadoop YARN: Overview

YARN (Yet Another Resource Negotiator) is:

- A subproject of Hadoop that separates resource-management and processing components
- A resource-management framework for Hadoop that is independent of execution engines



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

YARN is a new framework (introduced in Hadoop 2.0) that facilitates the writing of arbitrary distributed processing frameworks and applications. YARN does the following:

- Provides the daemons and APIs that are necessary to develop generic distributed applications of any kind
- Handles and schedules resource requests (such as memory and CPU) from such applications
- Supervises the execution of those requests

YARN's execution model is more generic than the earlier MapReduce implementation. Unlike the original version of Apache Hadoop MapReduce (also called MR1), YARN can run applications that do not follow the MapReduce model.

MapReduce 2 (MR2)

With the introduction of YARN, there is no longer a single JobTracker to run jobs and no TaskTracker to run tasks of the jobs. The MR1 framework was rewritten to run within a submitted application on top of YARN. This application was called MR2 (MapReduce version 2). It has the familiar MapReduce execution underneath, except that each job now controls its own destiny via its own ApplicationMaster that takes care of execution flow (such as scheduling tasks, handling speculative execution and failures, and so on). It is a more isolated and scalable model than the MR1 system, which had a singular JobTracker to perform all resource management, scheduling, and task monitoring.

MR2 and a new proof-of-concept application called the DistributedShell are the first two applications that use the YARN API in CDH4.

The YARN-based architecture of Hadoop 2.0 provides a more general processing platform that is not constrained to MapReduce.

With YARN, you can now run multiple applications in Hadoop, all sharing common resource management. Many organizations are already building applications on YARN to bring them into Hadoop.

MapReduce 2.0 (MRv2) or YARN (Yet Another Resource Negotiator) Architecture

- MRv2 splits up the two major functionalities of the JobTracker, resource management and job scheduling/monitoring, into separate new daemons.
 - A global ResourceManager (RM)
 - A per-application ApplicationMaster (AM)
- The RM and per-node slave, the NodeManager (NM), form the data-computation framework.
- The RM is the ultimate authority that arbitrates resources among all the applications in the system.
- The per-application AM negotiates resources from the RM and works with the NodeManager(s) to execute and monitor the tasks.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The RM high availability (HA) feature adds redundancy in the form of an Active/Standby RM pair to remove this single point of failure.

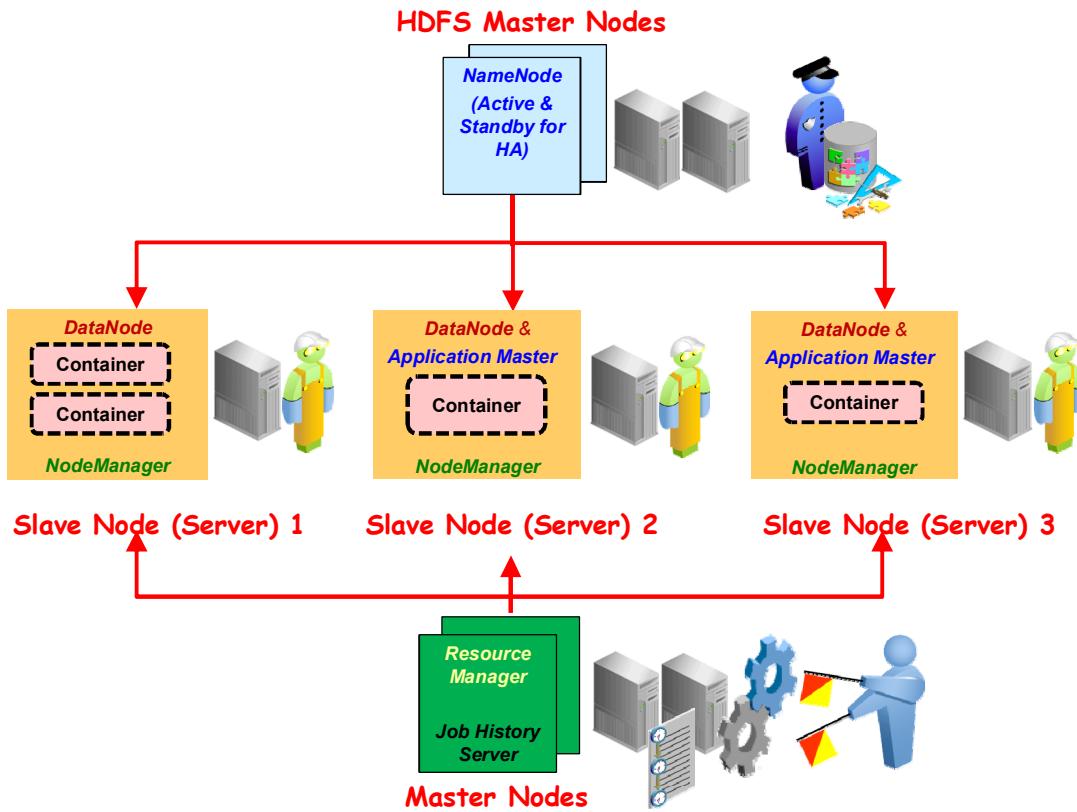
MapReduce 2.0 (MRv2) or YARN (Yet Another Resource Negotiator) Daemons

Component	Description
ResourceManager (RM)	<ul style="list-style-type: none"> A dedicated scheduler that allocates resources to the requesting applications The RM has two main components: Scheduler and ApplicationsManager It is a critical component of the cluster and runs on a dedicated master node.
NodeManager (NM) [Maps to TaskTracker in MRv1]	<ul style="list-style-type: none"> Each slave node in the cluster has an NM daemon, which acts as a slave for the RM. Each NM tracks the available data processing resources and usage (cpu, memory, disk, network) on its slave node and sends regular reports to the RM.
ApplicationMaster (AM)	<ul style="list-style-type: none"> The per-application AM is responsible for negotiating resources from the RM and working with the NM(s) to execute and monitor the tasks. Runs on a slave node
Container	<ul style="list-style-type: none"> A <i>container</i> is a collection of all the resources necessary to run an application: CPU, memory, network bandwidth, and disk space. Runs on a slave node in a cluster.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hadoop Basic Cluster YARN (MRv2): Example



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

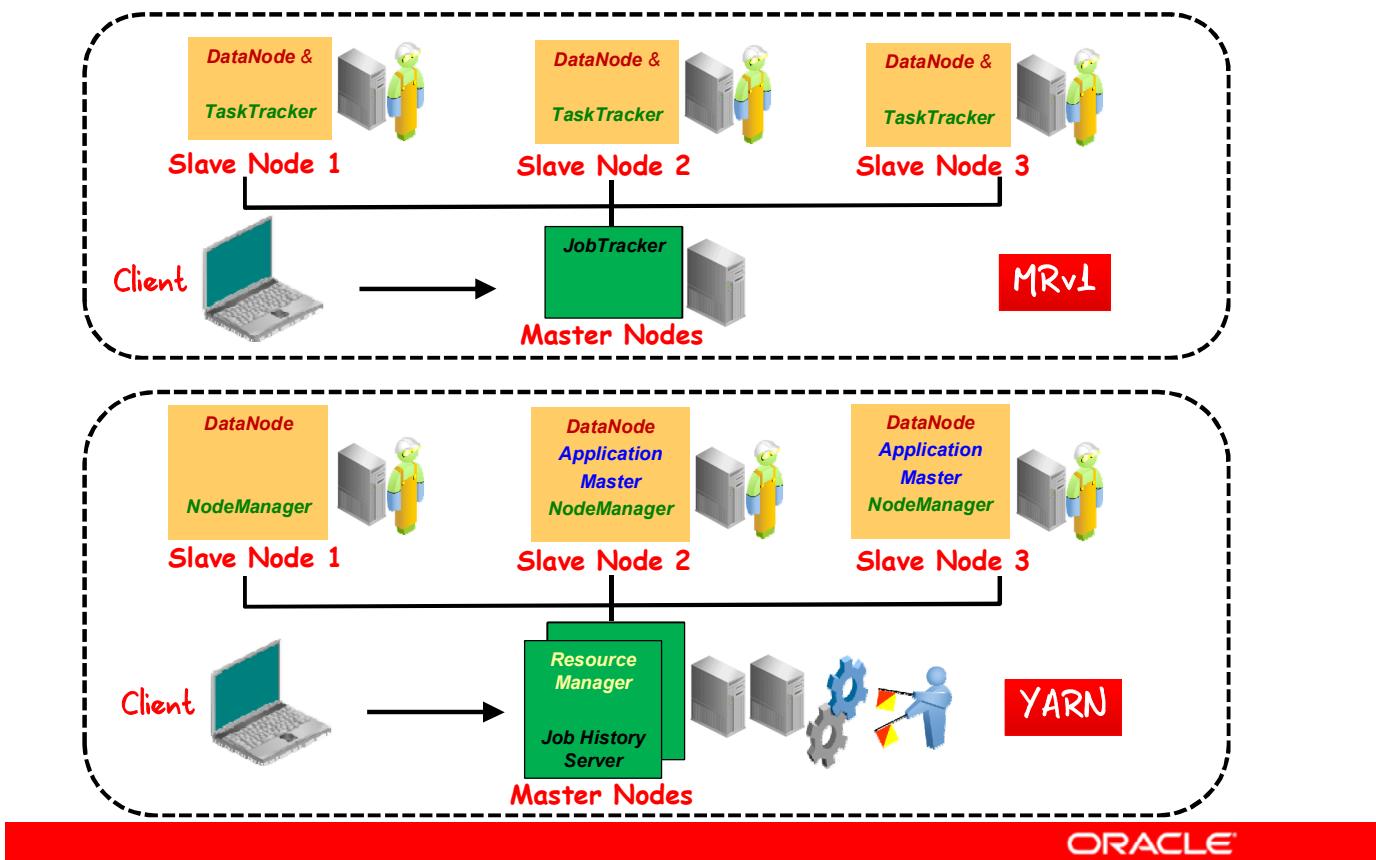
YARN is meant to provide a more efficient and flexible workload scheduling as well as a resource management facility, both of which will ultimately enable Hadoop to run more than just MapReduce jobs.

The second theme behind YARN is to move from the slots approach of Resource management to the physical world approach (Memory, CPU, Disk). This includes determining the amount of resources that can be used by each process. The Scheduler is responsible for allocating resources to the various applications that are running in the cluster. Scheduling is covered in lesson 10.

The ApplicationsManager is the YARN component responsible for handling the applications that are submitted by clients. In addition, it also negotiates the container on behalf of the application for the Application Master. The ApplicationsManager also provides the services of restarting the Application Master in case of failures.

Resource allocation is dynamic. AMs send resource requests with parameters such as the number of containers that will be required, the specifications of resources in each container such as 3 CPUs, 2 cores, and 12 GB RAM. In addition, the requests include any locality preferences for the container at the node or rack level where the required data exists and the priorities of requests within an application.

YARN Versus MRv1 Architecture



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

JobTracker

MapReduce processing in Hadoop 1 is handled by the JobTracker and TaskTracker daemons. As discussed in an earlier lesson, the JobTracker maintains a view of all available processing resources in the Hadoop cluster. As client application requests come in, it schedules and sends them to the TaskTracker nodes in the cluster for execution. While the applications are running, the JobTracker receives status updates from the TaskTracker nodes. In case of any failures, it handles re-running the applications.

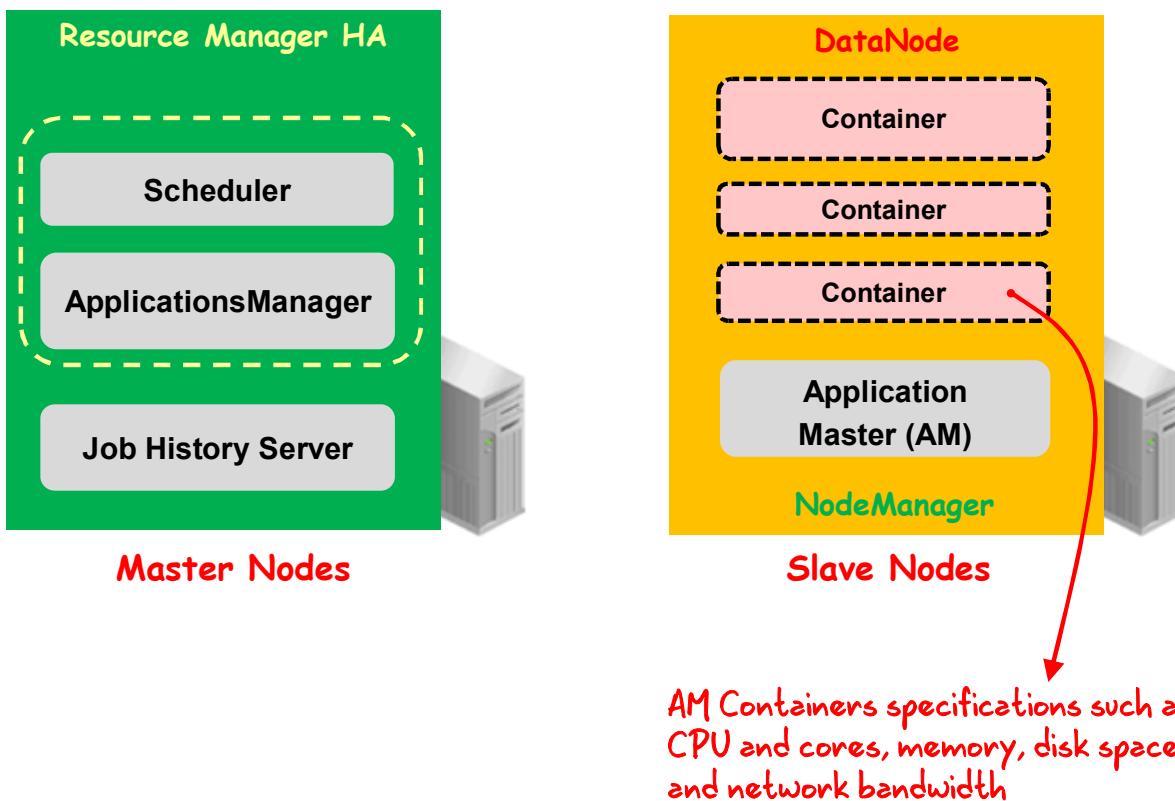
TaskTrackers

An instance of the TaskTracker daemon runs on every slave node in the Hadoop cluster, which means that each slave node has a service that ties it to the processing (TaskTracker) and the storage (DataNode), which enables Hadoop to be a distributed system as shown in the slide example for MRv1. The TaskTracker receives processing requests from the JobTracker. Its main responsibility is to track the execution of MapReduce applications running locally on its slave node and to send status updates to the JobTracker.

TaskTrackers manage the processing resources on each slave node in the form of processing slots. Slave nodes have a fixed number of map slots and reduce slots for the execution of map and reduce tasks, respectively.

YARN is meant to provide a more efficient and flexible workload scheduling as well as a resource management facility, both of which will ultimately enable Hadoop to run more than just MapReduce jobs. The components of YARN are covered on the next few pages.

YARN (MRv2) Architecture



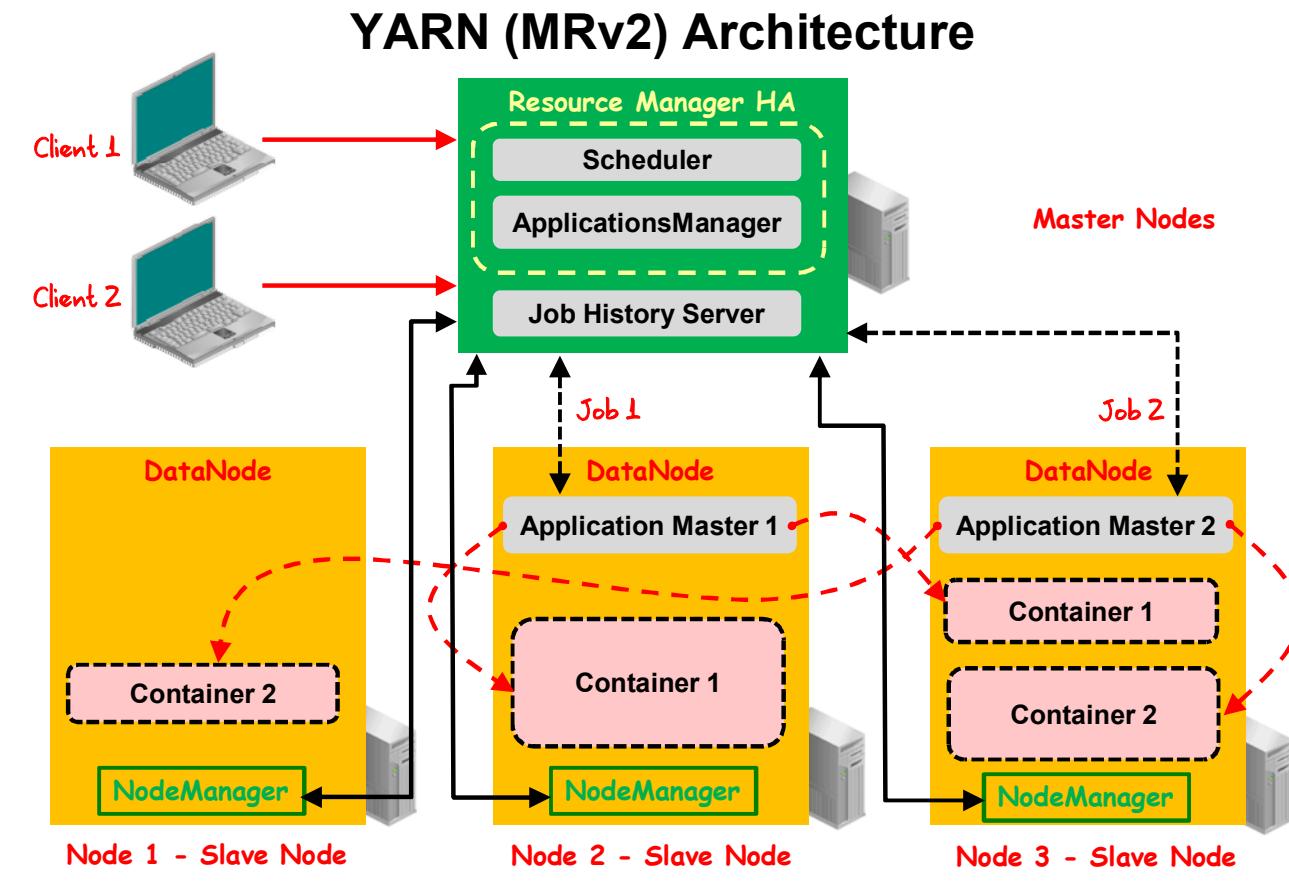
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The second theme behind YARN is to move from the slots approach of Resource management to the physical world approach (Memory, CPU, Disk). This includes determining the amount of resources that can be used by each process for each node (for example, Impala can use 4 core, 16 GB RAM; MapReduce can use 12 core, 32 GB RAM). The Scheduler is responsible for allocating resources to the various applications that are running in the cluster. Scheduling is done based on the global model of the cluster the RM is aware of. It uses queues and capacity parameters during the allocation process. The scheduler will be covered later in this lesson. The scheduling policy can be plugged into the Scheduler. The two popular scheduling policies in Hadoop 1.X and 2.x are Capacity Scheduler and Fair Scheduler, respectively.

The ApplicationsManager is the YARN component responsible for handling the applications that are submitted by clients. In addition, it also negotiates the container on behalf of the application for the Application Master. The ApplicationsManager also provides the services of restarting the Application Master in case of failures.

Resource allocation is dynamic. AMs send resource requests with parameters such as the number of containers that will be required, the specifications of resources in each container such as 3 CPUs, 2 cores, and 12 GB RAM. In addition, the requests include any locality preferences for the container at the node or rack level where the required data exists and the priorities of requests within an application.



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The second theme behind YARN is to:

- Move from the slots approach of Resource management to the physical world approach (Memory, CPU, Disk)
- Determinate the amount of resource that can be used by each process for each node (for example, Impala can use 4 core, 16 GB RAM; MapReduce can use 12 core, 32 GB RAM)
- Some amount of RAM, cores, weight for IO operations is dedicated for each map or reduce.

MapReduce 2.0 (MRv2) or YARN Daemons

- ResourceManager/NodeManager architecture
- YARN splits the two major functionalities of the JobTracker, resource management and job scheduling/monitoring, into separate new daemons.
 - A global ResourceManager (RM)
 - A per-application ApplicationMaster (AM)
- The RM and per-node slave, the NodeManager (NM), form the data-computation framework.
- The RM is the ultimate authority that allocates resources to all of the applications in the cluster.
- The per-application AM negotiates resources from the RM and works with the NodeManager(s) to execute and monitor the tasks.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

YARN (MRv2) Daemons

Component	Description
ResourceManager (RM)	<ul style="list-style-type: none">• A dedicated scheduler that allocates resources to the requesting applications. It has two main components: Scheduler and ApplicationsManager.• It is a critical component of the cluster and runs on a dedicated master node.• HA RM with Active and Standby RMs automatic failover
NodeManager (NM)	<ul style="list-style-type: none">• Each slave node in the cluster has a NM daemon, which acts as a slave for the RM.• Each NM tracks the available data processing resources and usage (CPU, memory, disk, and network) on its slave node and sends regular reports to the RM.
ApplicationMaster (AM)	<ul style="list-style-type: none">• The per-application AM is responsible for negotiating resources from the RM and working with the NM(s) to execute and monitor the tasks. It runs on a slave node.
Container	<ul style="list-style-type: none">• A <i>container</i> is a collection of all the resources necessary to run an application: CPU, memory, network bandwidth, and disk space. It runs on a slave node in a cluster.
Job History Server	<ul style="list-style-type: none">• Archives jobs and metadata



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

YARN is meant to provide a more efficient and flexible workload scheduling as well as a resource management facility, both of which will ultimately enable Hadoop to run more than just MapReduce jobs.

YARN: Features

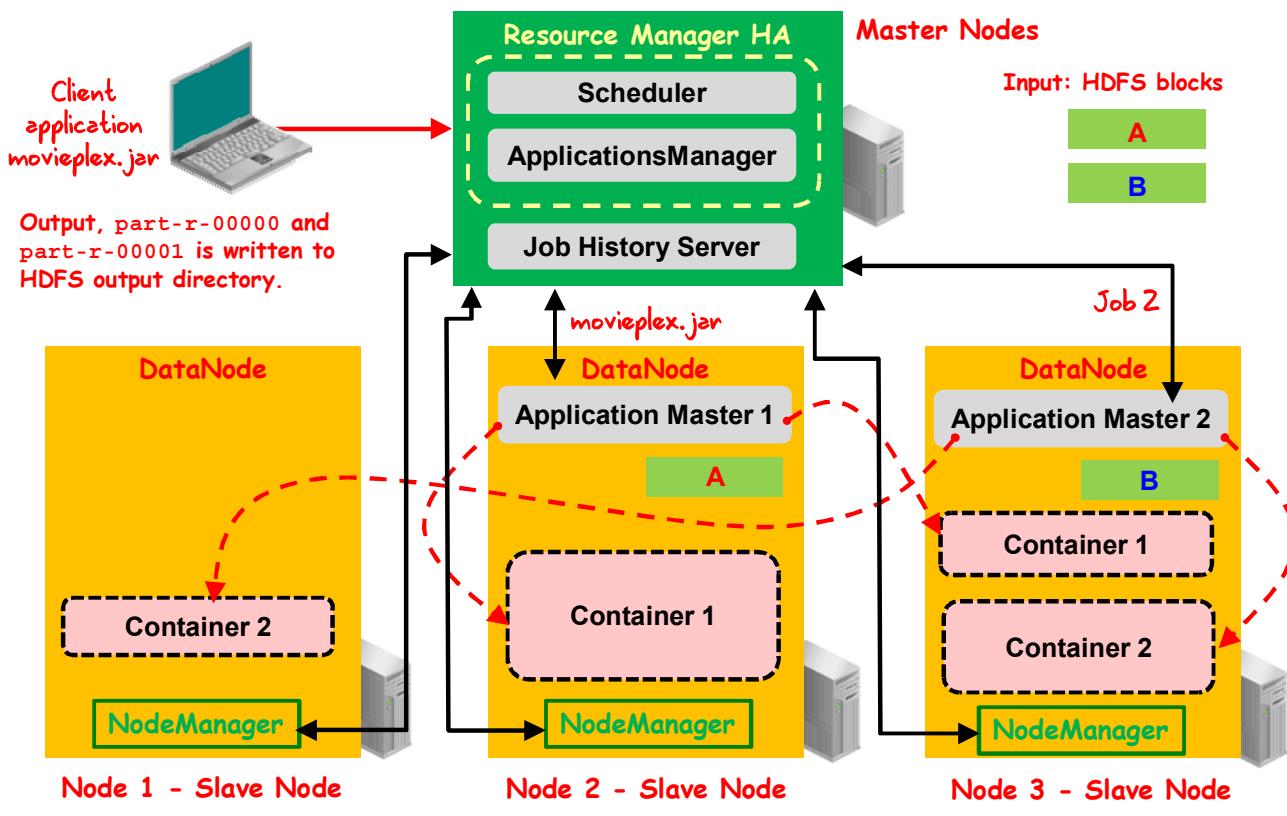
- Scalability
- Compatibility with MapReduce
- Improved cluster utilization
- Support for workloads other than MapReduce



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

- **Scalability:** YARN ResourceManager concentrates exclusively on scheduling, resulting in better cluster management.
- **Compatibility with MapReduce:** Existing MapReduce applications and users can run on top of YARN without disruption to their existing processes.
- **Improved cluster utilization:** There are no named map and reduce slots, which helps use cluster resources more efficiently.
- **Support for workloads other than MapReduce:** Additional programming models such as graph processing and iterative modeling are possible for data processing. These models enable enterprises to realize near-real-time processing and increased ROI on their Hadoop investments.

Launching an Application on a YARN Cluster



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The slide example shows how the YARN components work together when an application such as a MapReduce is executed.

1. The client application, `movieplex.jar` in this example, submits an application request to the Resource Manager. In this example, we are assuming that the HDFS data blocks needed by this application are blocks A and B.
2. The Resource Manager requests from an available Node Manager to create an Application Master instance for the `movieplex.jar` application. The Node Manager gets a free container for the Application Master, and then starts it up. This new Application Master, Application Master 1 in this example, initializes itself by registering itself with the Resource Manager.
3. The Application Master determines the required processing resources to execute the application by communicating with the NameNode in the cluster. The Application Master requests the names and locations of the data files and data blocks that are needed by the application. The Application Master then calculates the number of map and reduce tasks are needed to process all this data.
4. The Application Master then requests the needed resources from the Resource Manager. Since this is a MapReduce application, it requests a map task from the Resource Manager and when the map tasks are completed, it will request the required reduce tasks. The Application Master sends heartbeat messages to the Resource Manager throughout its lifetime, with a list of the requested resources.

5. The Resource Manager accepts the resource request and queues up the specific resource requests alongside all the other resource requests that are already scheduled. As the requested resources become available on the slave nodes, the Resource Manager grants the Application Master leases for containers on specific slave nodes.
6. The Application Master requests the assigned container from the Node Manager and sends it the information that is required to run the application. The Node Manager then creates the requested container process and starts it. The Resource Manager will try to grant the map resources where the data is located, data locality. In this example, blocks A and B are located in slave nodes 2 and 3; therefore, the Resource Manager will assign the map tasks on nodes 2 and 3. The intermediate key-value data generated by the map tasks are stored on the local disks on nodes 2 and 3. This data will be used by the reduce tasks. Remember from the MapReduce lesson that the intermediate data is copied to the reduce tasks during the Shuffle and Sort phase.
7. The application executes while the container processes are running. The Application Master monitors their progress, and in the event of a container failure or a node failure, the task is restarted on the next available container or node. If the same task fails after four attempts (default), the entire job fails. During this phase, the Application Master also communicates directly with the client to respond to status requests.
8. While containers are running, the Resource Manager can send a kill order to the Node Manager to terminate a specific container. This can be as a result of a scheduling priority change (we will discuss the schedulers later on in this lesson) or when the application is completed.
9. When all tasks are completed, the Application Master sends the result set to the client application, informs the Resource Manager that the application has successfully completed, deregisters itself from the Resource Manager, and shuts itself down. The output is written to HDFS and the completed job information is written to the JobHistory server. The resources used by this application are released.

MRv1 Versus MRv2

	MRv1	MRv2
Architecture	JobTracker/TaskTracker	ResourceManager/NodeManager
Scalability	One JobTracker per cluster. It can support 3000-4000 nodes.	The ApplicationMaster provides much of the job-oriented functionality of the JobTracker so that the entire system can scale more dramatically. It can support 10,000 nodes.
Job Type	<i>MapReduce only</i>	<i>Supports multiple frameworks such as MapReduce and Impala</i>
Resource Allocation	Fixed numbers (configured) of map and reduce tasks slots	An application (using the ApplicationMaster) can request resources with highly specific requirements, such as the resource name, memory, number of CPUs, and cores (containers).



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Agenda

- Apache Hadoop YARN architecture
- Job Scheduling in YARN
- YARN application command



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Job Scheduling in YARN

YARN provides a pluggable model to schedule policies. The scheduler is responsible for deciding where and when to run tasks. YARN supports the following pluggable schedulers:

- FIFO (First in, First out)
 - Allocates resources based on arrival time
- Capacity Scheduler
 - Allocates resources to pools, with FIFO scheduling within each pool
 - Default in Hadoop
- Fair Scheduler
 - Allows YARN applications to share resources in large clusters fairly
 - We will focus on the fair scheduler in this course
 - Default in CDH5 (used in Oracle BDA)



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Resource management helps ensure predictable behavior by defining the impact of different services on cluster resources. The goal of resource management features is to guarantee completion in a reasonable time frame for critical workloads.

FIFO Scheduler

The scheduling algorithm that was available in Hadoop version 1 and it stands for "first in, first out." The FIFO scheduler is a simple "first come, first served" scheduler in which the JobTracker pulls jobs from a work queue, oldest job first. The FIFO schedule is fine for small jobs, but inefficient when large shared clusters are used.

Capacity Scheduler

Capacity Scheduler guarantees a tenant-promised capacity on a shared cluster. If other tenants use less of their requested capacity, the scheduler allows the tenant to tap into these unused resources. The Capacity Scheduler uses *job queues* to facilitate the organized sharing of Hadoop clusters. It guarantees minimum capacity levels for all queues and makes unused capacity available to overloaded queues, which leads to optimized cluster utilization.

Fair Scheduler

The Fair Scheduler enables the sharing of large clusters. Fair scheduling is a method of assigning resources to applications such that all applications get, on average, an equal share of resources over time.

YARN Fair Scheduler

- A pluggable scheduler that allows YARN applications to share resources in large clusters fairly.
- In the YARN Fair Scheduler, the terms "queue" and "pool" are used interchangeably.
- It assigns resources to applications so that all applications get an equal share of allocated resources over time.
- It bases scheduling fairness decisions only on memory but you can configure it to be based on both memory and CPU.
- It organizes applications into "queues" (pools), and shares resources fairly between these queues. Every application belongs to a queue ("default" queue is the default).
- It grants YARN containers to the queue with the least amount of allocated resources.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A queue in YARN is a logical view of the available resources on the physical nodes.

When there is a single application running, it can use the entire cluster resources. When other applications are submitted, resources that become available are assigned to the new applications. This ensures that each application eventually gets approximately the same amount of resources. This also enables small applications to finish in reasonable time while not starving long-lived applications.

The Fair Scheduler allows assigning guaranteed minimum shares to queues, which is useful for ensuring that certain users, groups, or production applications always get sufficient resources. When a queue contains applications, it gets at least its minimum share, but when the queue does not need its full guaranteed share, the excess is split between other running applications. This lets the scheduler guarantee capacity for queues while using resources efficiently when these queues do not contain applications.

YARN Fair Scheduler

- Works also with applications priorities. The priorities are used as weights to determine the fraction of total resources that each application must get.
- If an application specifically lists a queue in a container resource request, the request is submitted to that queue.
- The Fair Scheduler supports moving a running application to a different queue. For example:
 - Moving an important application to a higher priority queue
 - Moving an unimportant application to a lower priority queue
- The Fair Scheduler also allows you to assign guaranteed minimum shares to queue.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

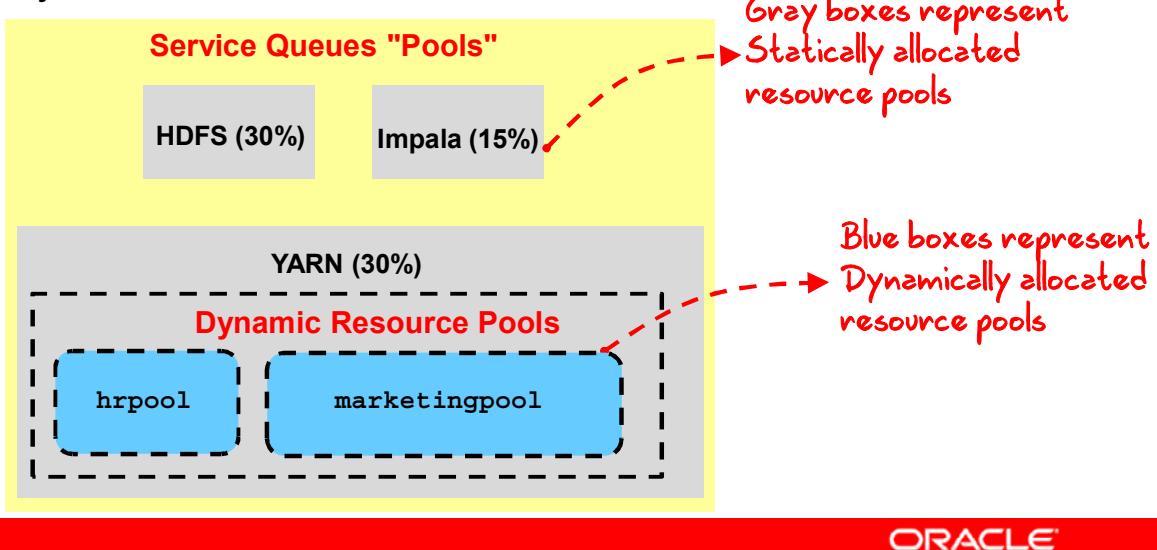
The Fair scheduler can also work with applications priorities. The priorities are used as weights to determine the fraction of total resources that each application should get. Heavier queues get more containers.

Cloudera Manager

Resource Management Features

Cloudera Manager provides the following features to assist you with allocating cluster resources to services:

- Static allocation (% of cluster resources)
- Dynamic allocation



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

If you are running CDH5, you can specify how YARN applications share resources by manually configuring the YARN scheduler . Starting with Cloudera Manager 4, you can partition resources across HBase, HDFS, Impala, MapReduce, and YARN services by allowing by setting configuration properties that were enforced by Linux control groups (Linux cgroups). With Cloudera Manager 5, you can allocate static resources using cgroups through a single static service pool wizard. You allocate services a percentage of total resources and the wizard configures the cgroups as you will see in an example in the next few pages.

You can use Cloudera Manager dynamic allocation to manage the scheduler configuration. YARN manages the virtual cores, memory, running applications, and scheduling policy for each pool. In the slide example, static pools for HDFS (30), Impala (15%), and YARN services are each assigned a percentage of the cluster resources. In addition, there are two dynamic resource pools: hrpool and marketingpool.

A dynamic resource pool is a named configuration of resources and a policy for scheduling the resources among YARN applications or Impala queries running in the pool. Dynamic resource pools allow you to schedule and allocate resources to YARN applications and Impala queries based on the user's access to specific pools and the resources available to those pools. If a pool's allocation is not in use it can be given to other pools. Otherwise, a pool receives a share of resources in accordance with the pool's weight. Dynamic resource pools have ACLs that restrict who can submit work to and administer them.

For more detailed information about Resource Management using Cloudera Manager, see the Cloudera documentation at:

http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/admin_rm.html

Static Service Pools

1 Clusters

2 Static Service Pools

3 Configuration

4 Step navigation (1-4) and Continue button

5 Status

Service	CPU %	Memory
HBase	0.51	158.7 GiB
Impala	0.00	6.1 GiB
hdfs	0.02	6.8 GiB
yarn	0.01	1.9 GiB

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Static service pools isolate the services in your cluster from one another, so that load on one service has a bounded impact on other services. Services are allocated a static percentage of total resources—CPU, memory, and I/O weight—which are not shared with other services.

Working with the Fair Scheduler

- Creating Dynamic Resource Pools
- Assigning priorities to Dynamic Resource Pools
- Monitoring Jobs



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Cloudera Manager Dynamic Resource Management: Example

```
# Linux commands for creating the groups and
# users in the new groups which are used in the
# on the next few slides.
groupadd marketing
groupadd hr
groupadd development
useradd -r -g marketing bob
useradd -r -g hr lucy
. . .
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Resource management uses users and groups that are configured for the cluster. These users may be sourced in a directory (e.g. Active Directory) or the operating system. In our example, we will use users and groups from the operating system. This slide shows how these users/groups were defined.

In the example in the slide, the syntax to create linux groups and users is shown. In the following slides, the user `lucy` from the `hr` group will attempt to submit jobs to the `hrpool` and `marketingpool` dynamic resource pools. User `bob` from the `marketing` group attempts to submit a job to the `marketingpool`.

Cloudera Manager Dynamic Resource Management: Example

The screenshot shows the Cloudera Manager web interface. At the top, there is a navigation bar with links for Home, Clusters (which is highlighted with a red box), Hosts, Diagnostics, Audits, Charts, Backup, and Administration. Below the navigation bar, the main content area shows a cluster named 'scaj51cdh (CDH 5.3.0)'. On the left, there is a sidebar with categories like Status, Configuration, Services, and Categories. Under Services, several services are listed: HBase, Impala, KMS (File), bigdatasql, hdfs, hive, hue, oozie, sentry, yarn, and zookeeper. In the center, under the 'Resource Management' section, there are two options: 'Dynamic Resource Pools' (which is highlighted with a red box and has a cursor icon over it) and 'Static Service Pools'. To the right of the main content area, there is a sidebar with a green status indicator, an 'Actions' dropdown, a link to 'Switch to the new layout', and a 'Save Changes' button.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To access the **Dynamic Resource Pools** page, select **Clusters > Dynamic Resource Pools** in Cloudera Manager.

Cloudera Manager Dynamic Resource Management: Example

The screenshot shows the Cloudera Manager interface for managing dynamic resource pools. The 'Configuration' tab is selected. Below it, a table lists four resource pools: 'root', 'default', 'hrpool', and 'marketingpool'. The 'hrpool' and 'marketingpool' rows are highlighted with a red box.

Name	Weight %	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Scheduling Policy	Action
root	1 100.0%	- / -	- / -	-	DRF	<input type="button" value="Edit"/>
default	1 0.9%	- / -	- / -	-	DRF	<input type="button" value="Edit"/>
hrpool	10 9.0%	5 / 25	100MB / 10000MB	5	DRF	<input type="button" value="Edit"/>
marketingpool	100 90.1%	25 / 50	500MB / 5000MB	10	DRF	<input type="button" value="Edit"/>

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, two dynamic resource pools were defined: `hrpool` and `marketingpool`. To view or edit the configuration of a pool, click the **Edit** button associated with a pool.

Cloudera Manager Dynamic Resource Management: Example

The screenshot shows the Cloudera Manager Dynamic Resource Management interface. At the top, there is a table titled "YARN" listing four resource pools: "root", "default", "hrpool", and "marketingpool". The "hrpool" row is highlighted with a red box. Below the table, a red arrow points from the "Edit" button next to "hrpool" in the table to a detailed "Edit Resource Pool: hrpool" dialog box.

Name	Weight %	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Scheduling Policy	Action
root	1 100.0%	- / -	- / -	-	DRF	<input type="button" value="Edit"/>
default	1 0.9%	- / -	- / -	-	DRF	<input type="button" value="Edit"/>
hrpool	10 9.0%	5 / 25	100MB / 10000MB	5	DRF	<input type="button" value="Edit"/>
marketingpool	100 90.1%	25 / 50	500MB / 5000MB	10	DRF	<input type="button" value="Edit"/>

Edit Resource Pool: hrpool

General **YARN** **Submission Access Control** **Administration Access Control**

Resource Pool Name: Alphanumeric characters only.

Scheduling Policy: DRF: Dominant Resource Fairness. Schedules resources fairly based on both CPU and memory. (Recommended)
 FAIR: Schedules resources fairly based only on memory.
 FIFO: First in, first out.

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, click the **Edit** button associated with `hrpool` to display the **Edit Resource Pool: hrpool**. This page has the following tabs: **General**, **YARN**, **Submission Access Control**, and **Administration Access Control**. In the **General** tab, you can specify the name of the pool, and the appropriate scheduling policy.

Cloudera Manager Dynamic Resource Management: Example

Edit Resource Pool: hrpool

General	YARN	Submission Access Control	Administration Access Control
Multiple configuration sets allow you to specify different settings based on your schedule.			
default			
Weight	10	Share of resources relative to other pools.	
Virtual Cores (Min / Max)	5	/	25
The minimum and the maximum number of virtual cores available to the pool. These override weight settings. (optional)			
Memory (Min / Max)	100	/	10000 MB
The minimum and the maximum amount of aggregate memory available to the pool. These override weight settings. (optional)			
Max Running Apps	5	A limit on the number of applications simultaneously running in a pool.	



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the **YARN** tab, you specify the pool's weight, the minimum and maximum cores, the minimum and maximum memory, and the maximum number of applications that can run simultaneously in this pool.

Cloudera Manager Dynamic Resource Management: Example

Edit Resource Pool: hrpool

General YARN Submission Access Control Administration Access Control

This feature is relevant only if **Enable ResourceManager ACLs** is set to true and **Admin ACL** is NOT set to * (See Other). Fair Scheduler Access Control Lists control who can submit applications to pools. For subpools, users who have permission automatically inherit the same ability for the child.

Allow anyone to submit to this pool
 Allow these users and groups to submit to this pool

Users: Comma separated list of users. Space characters are not allowed.

Inherited from parent pools: oracle,root

Groups hr

Edit Resource Pool: hrpool

General YARN Submission Access Control Administration Access Control

This feature is relevant only if **Enable ResourceManager ACLs** is set to true and **Admin ACL** is NOT set to * (See Other). Fair Scheduler Access Control Lists control who can administer pools. For subpools, users who have permission automatically inherit the same ability for the child.

Allow anyone to administer this pool
 Allow these users and groups to administer this pool

Users: Comma separated list of users. Space characters are not allowed.

Inherited from parent pools: oracle,root

Groups hr

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the **Submission Access Control** tab, you can specify who can submit applications to the `hrpool`. In the example in the slide, the **Allow these users and groups to submit to this pool** is selected. The `hr` group is specified in the **Groups** section. This means that only users in the `hr` group can submit applications to the `hrpool` group.

The `oracle` and `root` users can also submit to this pool. The properties for the pool are derived from the `root` pool.

Submitting a Job to hrpool By User lucy from the hr Group

```
[oracle@scaj51bda12 jars]$ id lucy
uid=478(lucy) gid=1005(hr) groups=1005(hr)
[oracle@scaj51bda12 jars]$ kinit lucy
Password for lucy@DEV.ORACLE.COM:
```

lucy belongs to the hr group; therefore,
she can submit the job to hrpool

```
$ hadoop jar hadoop-mapreduce-examples-2.5.0-cdh5.3.0.jar teragen -
Dmapred.map.tasks=5 -D mapredqueueuname=hrpool 1000000
/user/hr/d2
```



```
15/02/26 10:08:59 INFO mapreduce.Job: map 0% reduce 0%
15/02/26 10:09:09 INFO mapreduce.Job: map 40% reduce 0%
15/02/26 10:10:06 INFO mapreduce.Job: map 60% reduce 0%
15/02/26 10:10:34 INFO mapreduce.Job: map 100% reduce 0%
15/02/26 10:10:34 INFO mapreduce.Job: Job job_1424929332007_0001 completed successfully
15/02/26 10:10:34 INFO mapreduce.Job: Counters: 31
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=576800
    FILE: Number of read operations=0
```

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the user lucy who belongs to the hr group issues the command **kinit**. This is a secure cluster. This means that when you want to run a hadoop command as a specific user, you must **kinit** - which gets you a kerberos ticket. There are other methods to authenticate users. This is covered in the lesson 28, “Securing Your Data.” Next, lucy submits the job which completes successfully because lucy belongs to the hr group. Any user who belongs to the hr group can submit jobs to the hrpool queue.

Monitoring the Status of the Submitted MapReduce Job

```
[oracle@scaj51bda12 jars]$ hadoop fs -ls /user/hr/d2
Found 6 items
-rw-r--r-- 3 lucy hive 0 2015-02-26 10:10 /user/hr/d2/_SUCCESS
-rw-r--r-- 3 lucy hive 20000000 2015-02-26 10:10 /user/hr/d2/part-m-00000
-rw-r--r-- 3 lucy hive 20000000 2015-02-26 10:10 /user/hr/d2/part-m-00001
-rw-r--r-- 3 lucy hive 20000000 2015-02-26 10:09 /user/hr/d2/part-m-00002
-rw-r--r-- 3 lucy hive 20000000 2015-02-26 10:09 /user/hr/d2/part-m-00003
-rw-r--r-- 3 lucy hive 20000000 2015-02-26 10:10 /user/hr/d2/part-m-00004
[oracle@scaj51bda12 jars]$
```

output directory

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	1	0	0 B	159 GB	0 B	0	248	0	4	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	Vcores Used	Vcores Pending	Vcores Reserved
0	0	0	1	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_142492932007_0001	lucy	TeraGen	MAPREDUCE	root.hrpool	Thu Feb 26 13:07:21 -0500 2015	Thu Feb 26 13:10:32 -0500 2015	FINISHED	SUCCEEDED		History

The job was submitted to the root.hrpool.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The first screen capture in the slide shows the output directory.

The second screen capture in the slide shows the successfully completed job using the ResourceManager Web UI.

Examining the marketingpool

Name	Weight %	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Scheduling Policy	
root	1 100.0%	- / -	- / -	-	DRF	Edit ▾
default	1 0.9%	- / -	- / -	-	DRF	Edit ▾
hrpool	10 9.0%	5 / 25	100MB / 10000MB	5	DRF	Edit ▾
marketingpool	100 90.1%	25 / 50	500MB / 5000MB	10	DRF	Edit ▾

Edit Resource Pool: marketingpool

General YARN Submission Access Control Administration Access Control

This feature is relevant only if **Enable ResourceManager ACLs** is set to true and **Admin ACL** is NOT set to * (See Other Settings). Fair Scheduler Access Control Lists control who can submit applications to pools. For subpools, users who have permission to submit a parent pool automatically inherit the same ability for the child.

Allow anyone to submit to this pool
 Allow these users and groups to submit to this pool

Users Comma separated list of users. Space characters are not allowed.
Inherited from parent pools: oracle,root

Groups marketing

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the next example, the user lucy will attempt to submit a job to the marketingpool. As seen in the screen capture in the slide, the marketingpool allows only users who belong to the marketing group to submit jobs to this pool.

Notice that the marketingpool is allocated many more resources than the hrpool. lucy would like to take advantage of these extra resources when running her job; however, she has not been given privileges to use the marketingpool queue.

Submitting a Job to marketingpool By User lucy from the hr Group

```
[oracle@scaj51bda12 jars]$ id lucy
uid=478(lucy) gid=1005(hr) groups=1005(hr)
[oracle@scaj51bda12 jars]$ kinit lucy
Password for lucy@DEV.ORACLE.COM:
```

lucy belongs to the hr group; therefore, she CANNOT submit the job to marketingpool

```
$ hadoop fs -rm -r /user/hr/d2
$ hadoop jar hadoop-mapreduce-examples-2.5.0-cdh5.3.0.jar teragen
-Dmapred.map.tasks=5 -D mapreduce.job.queuename=marketingpool
1000000 /user/hr/d2
```

```
15/02/26 10:43:21 INFO Configuration.deprecation: mapred.map.tasks is deprecated. Instead, use mapred
15/02/26 10:43:21 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1424929332007_0003
15/02/26 10:43:21 WARN token.Token: Cannot find class for token kind kms-dt
15/02/26 10:43:21 WARN token.Token: Cannot find class for token kind kms-dt
Kind: kms-dt, Service: 192.168.42.121:16000, Ident: 00 04 6c 75 63 79 04 79 61 72 6e 00 8a 01 4b c7 33 46 ab 8a
eb 3f ca ab 3c 1d
15/02/26 10:43:21 INFO mapreduce.JobSubmitter: Kind: HDFS_DELEGATION_TOKEN, Service: ha-hdfs:scaj51cdh-ns, Ident: (HD
FS_DELEGATION_TOKEN token 202 for lucy)
15/02/26 10:43:22 INFO impl.YarnClientImpl: Submitted application application_1424929332007_0003
15/02/26 10:43:22 INFO mapreduce.JobSubmitter: Cleaning up the staging area /user/lucy/.staging/job_1424929332007_00
2
15/02/26 10:43:22 WARN security.UserGroupInformation: PrivilegedActionException as:lucy@DEV.ORACLE.COM (auth:KERBERO
S) cause:java.io.IOException: Failed to run job : User lucy cannot submit applications to queue root.marketingpool
java.io.IOException: Failed to run job : User lucy cannot submit applications to queue root.marketingpool
```



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, lucy will attempt to submit a job to the marketingpool. The job will fail since lucy cannot submit applications to the marketingpool queue.

Monitoring the Status of the Submitted MapReduce Job

loop All Applications

Logged in as: dr.who

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
1	0	0	1	0	0 B	159 GB	0 B	0	248	0	4	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	Vcores Used	Vcores Pending	Vcores Total
0	0	0	1	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries Search:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1424929332007_0003	lucy	TeraGen	MAPREDUCE	root.marketingpool	Thu Feb 26 13:43:22 -0500 2015	Thu Feb 26 13:43:22 -0500 2015	FAILED	FAILED		UNASSIGNED
application_1424929332007_0002	lucy	TeraGen	MAPREDUCE	root.marketingpool	Thu Feb 26 13:41:48 -0500 2015	Thu Feb 26 13:41:48 -0500 2015	FAILED	FAILED		UNASSIGNED
application_1424929332007_0001	lucy	TeraGen	MAPREDUCE	root.hrpool	Thu Feb 26 13:07:21 -0500 2015	Thu Feb 26 13:10:32 -0500 2015	FINISHED	SUCCEEDED		History



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The screen capture in the slide shows the failed job using the ResourceManager Web UI.

Submitting a Job to marketingpool By User bob from the marketing Group

```
[oracle@scaj51bda12 jars]$ id bob
uid=477(bob) gid=1007(marketing) groups=1007(marketing)
[oracle@scaj51bda12 jars]$ kinit bob
Password for bob@DEV.ORACLE.COM:
[oracle@scaj51bda12 jars]$ [REDACTED]
```

bob belongs to the marketing group; therefore, he can submit the job to marketingpool.

```
$ hadoop fs -rm -r /user/marketing/d1
$ hadoop jar hadoop-mapreduce-examples-2.5.0-cdh5.3.0.jar teragen
-Dmapred.map.tasks=5 -D mapreduce.job.queuename=marketingpool
1000000 /user/marketing/d1
```

```
15/02/26 16:07:57 INFO mapreduce.Job: The url to track the job: http://scaj51bda12.us.oracle.com:8088/proxy/application_1424929332007_0004/
15/02/26 16:07:57 INFO mapreduce.Job: Running job: job_1424929332007_0004
15/02/26 16:08:09 INFO mapreduce.Job: Job job_1424929332007_0004 running in uber mode : false
15/02/26 16:08:09 INFO mapreduce.Job: map 0% reduce 0%
15/02/26 16:08:19 INFO mapreduce.Job: map 80% reduce 0%
15/02/26 16:08:23 INFO mapreduce.Job: map 100% reduce 0%
15/02/26 16:08:23 INFO mapreduce.Job: Job job_1424929332007_0004 completed successfully
```



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In the next example, user bob who belongs to the marketing group, successfully submits a job to the marketingpool.

Monitoring the Status of the Submitted MapReduce Job

```
[oracle@scraj51bda12 jars]$ hadoop fs -ls /user/marketing/d1
Found 6 items
-rw-r--r-- 3 bob marketing 0 2015-02-26 16:08 /user/marketing/d1/_SUCCESS
-rw-r--r-- 3 bob marketing 2000000 2015-02-26 16:08 /user/marketing/d1/part-m-00000
-rw-r--r-- 3 bob marketing 2000000 2015-02-26 16:08 /user/marketing/d1/part-m-00001
-rw-r--r-- 3 bob marketing 2000000 2015-02-26 16:08 /user/marketing/d1/part-m-00002
-rw-r--r-- 3 bob marketing 2000000 2015-02-26 16:08 /user/marketing/d1/part-m-00003
-rw-r--r-- 3 bob marketing 2000000 2015-02-26 16:08 /user/marketing/d1/part-m-00004
[oracle@scraj51bda12 jars]$
```

output directory

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
2	0	0	2	0	0 B	159 GB	0 B	0	248	0	4	0	0	0	0

User Metrics for dr.who

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	Vcores Used	Vcores Pending	Vcores Reserved
0	0	0	2	0	0	0	0 B	0 B	0 B	0	0	0

Show 20 entries Search:

ID	User	Name	Application	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI
application_1424929332007_0004	bob	TeraGen	MAPREDUCE	root.marketingpool	Thu Feb 26 19:07:56 -0500 2015	Thu Feb 26 19:08:22 -0500 2015	FINISHED	SUCCEEDED		History
application_1424929332007_0003	lucy	TeraGen	MAPREDUCE	root.marketingpool	Thu Feb 26	Thu Feb 26 13:43:22	FAILED	FAILED		UNASSIGNED

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The first screen capture in the slide shows the output directory.

The second screen capture in the slide shows the successfully completed job using the ResourceManager Web UI.

Delay Scheduling

- YARN schedulers attempt to honor locality container requests.
- If the requested node is not available, delaying the granting of an alternative resource for a few seconds can increase the chances of getting a container on the requested node.
- Delay scheduling is supported by both Capacity and Fair Schedulers.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Agenda

- Apache Hadoop YARN architecture
- Job Scheduling in YARN
- YARN application command



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

YARN application Command

You can use the `yarn application` command to list and kill applications:

```
$ yarn application <options>
```

<options>	Description
<code>-list</code>	Lists applications from the Resource Manager.
<code>-appStates</code>	Works with <code>-list</code> to filter applications based on input comma-separated list of application states. The options are: ALL, NEW, NEW_SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, KILLED.
<code>-status ApplicationID</code>	Prints the status of the application.
<code>- kill ApplicationID</code>	Kills the application.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

YARN application Command: Example

```
$ yarn application -list
```

```
[oracle@bigdatalite ~]$ yarn application -list
15/02/12 09:00:00 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
Total number of applications (application-types: [] and states: [SUBMITTED, ACCEPTED, RUNNING]):0
      Application-Id      Application-Name      Application-Type
User    Queue       State      Final-State
Progress   Tracking-URL
[oracle@bigdatalite ~]$
```

1

2. Insert into staging table Inserts filtered and transformed log data into staging table

```
1 INSERT OVERWRITE TABLE moviework.movieapp_log_stage
2 SELECT * FROM (
3   SELECT custid,
4         CASE WHEN ml.genrefield > 0 THEN ml.genrefield ELSE 1 END genrefield,
5         ml.time,
6         CAST((CASE ml.recommended WHEN 'Y' THEN 1 ELSE 0 END) AS INT) recommended,
7         ...
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Execute **Save** **Save as...** **Explain** or create a **New query**

2

```
Progress          TRACKING-URL
[oracle@bigdatalite ~]$ yarn application -list
15/02/12 09:57:57 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
Total number of applications (application-types: [] and states: [SUBMITTED, ACCEPTED, RUNNING]):1
      Application-Id      Application-Name      Application-Type
Queue    State       Final-State      Progress
Tracking-URL
application 1423174990155_0022 INSERT OVERWRITE TABLE moview...union_result(Stage-1)
MAPREDUCE      oracle      root.oracle      RUNNING
DEFINED           5% http://bigdatalite.localdomain:54444
```

3

ORACLE

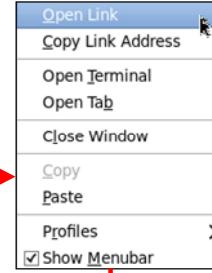
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The code example in the slide uses the `-list` option which lists applications from the Resource Manager. In the first screen capture, there are no running application. In the second screen capture, a saved hiveQL query is executed which will start a MapReduce job. In the third screen capture, the same `yarn application` command is re-executed, and the results show the new hiveQL query, `INSERT`, which starts a MapReduce job. In the third screen capture, it shows only 5% is completed at the time the screen capture is taken.

YARN application Command: Example

```
$ yarn application -status <job_id>
```

```
[oracle@bigdatalite ~]$ yarn application -status application_1423174990155_0022
15/02/12 10:14:39 INFO client.RMProxy: Connecting to ResourceManager at localhost/127
:8032
Application Report :
  Application-Id : application_1423174990155_0022
  Application-Name : INSERT OVERWRITE TABLE moview...union_result(Stage-1)
  Application-Type : MAPREDUCE
  User : oracle
  Queue : root.oracle
  Start-Time : 1423753071390
  Finish-Time : 1423753095903
  Progress : 100%
  State : FINISHED
  Final-State : SUCCEEDED
  Tracking-URL : http://bigdatalite.localdomain:19888/jobhistory/job/job_1423174990155_0022
  RPC Port : 14374
  AM Host : bigdatalite.localdomain
  Diagnostics :
[oracle@bigdatalite ~]$
```



The screenshot shows the Hadoop MapReduce Job interface. At the top, there's a logo and the text "MapReduce Job job_1423174990155_0022". Below that is a table with the following data:

Job	
Job Name:	INSERT OVERWRITE TABLE moview...union_result(Stage-1)
User Name:	oracle
Queue:	root.oracle
State:	SUCCEEDED
Uberized:	false
Submitted:	Thu Feb 12 09:57:51 EST 2015
Started:	Thu Feb 12 09:57:57 EST 2015
Finished:	Thu Feb 12 09:58:15 EST 2015

On the left, there's a sidebar with links: Application, Job (selected), Overview, Counters, Configuration, Map tasks, Reduce tasks, and Tools.

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Monitoring an Application Using the UI

Queue or "pool" name

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus
application_1423174990155_0025	oracle	INSERT OVERWRITE TABLE moview...union_result(Stage-4)	MAPREDUCE	root.oracle	Thu, 12 Feb 2015 14:59:01 GMT	Thu, 12 Feb 2015 14:59:14 GMT	FINISHED	SUCCESS
application_1423174990155_0024	oracle	INSERT OVERWRITE TABLE moview...union_result(Stage-3)	MAPREDUCE	root.oracle	Thu, 12 Feb 2015 14:58:39	Thu, 12 Feb 2015 14:58:59	FINISHED	SUCCESS

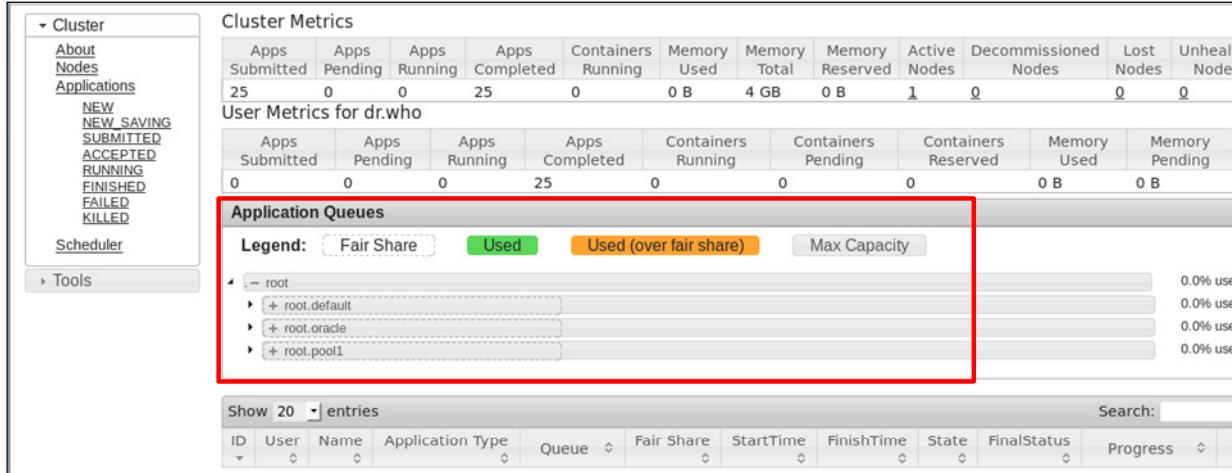
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

You also monitor applications such as the one that was submitted on the previous slide using the UI. You can use the Hadoop bookmark on the browser toolbar to select either the YARN applications, or the JobHistory server to view the status of applications.

In our example, the MapReduce job that was created when the HiveQL query was executed is already completed. You can use the UI to examine job specific information such as the status of applications and if they were completed successfully. You can also filter applications by state, view the name of the application (for Hive, the name of the application is the actual HiveQL query), the resource pool it came from, and much more.

The Scheduler: BDA Example



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to use YARN to manage resources efficiently.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 10

- Monitor a running job by using the CLI.
- Monitor a running job by using the Resource Manager UI.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

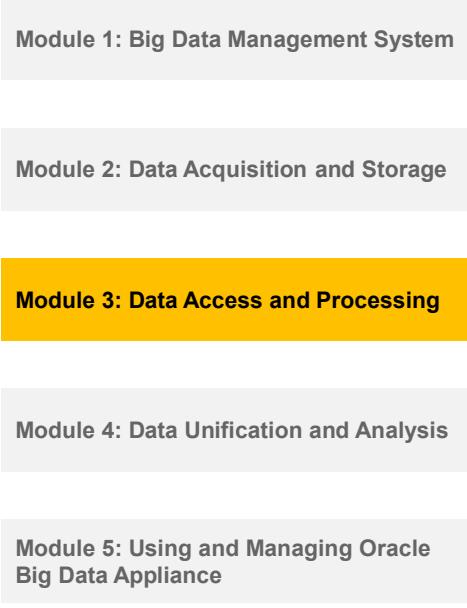
11

Overview of Hive and Pig

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



Lesson 9: Introduction to MapReduce

Lesson 10: Resource Management Using YARN

Lesson 11: Overview of Apache Hive and Apache Pig

Lesson 12: Overview of Cloudera Impala

Lesson 13: Using Oracle XQuery for Hadoop

Lesson 14: Overview of Solr

Lesson 15: Apache Spark

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This lesson introduces Hive and Pig at a high level and lists their benefits.

Objectives

After completing this lesson, you should be able to:

- Define Hive
- Describe the Hive data flow
- Create a Hive database
- Define Pig
- List the features of Pig



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hive

- Hive is an open source Apache project and was originally developed by Facebook.
- Hive enables analysts who are familiar with SQL to query data stored in HDFS by using HiveQL (a SQL-like language).
- It is an infrastructure built on top of Hadoop that supports the analysis of large data sets.
- Hive transforms HiveQL queries into standard MapReduce jobs (high level abstraction on top of MapReduce).
- Hive communicates with the *JobTracker* to initiate the MapReduce job.
- This lesson covers Hive and Pig at a high level.



ORACLE®

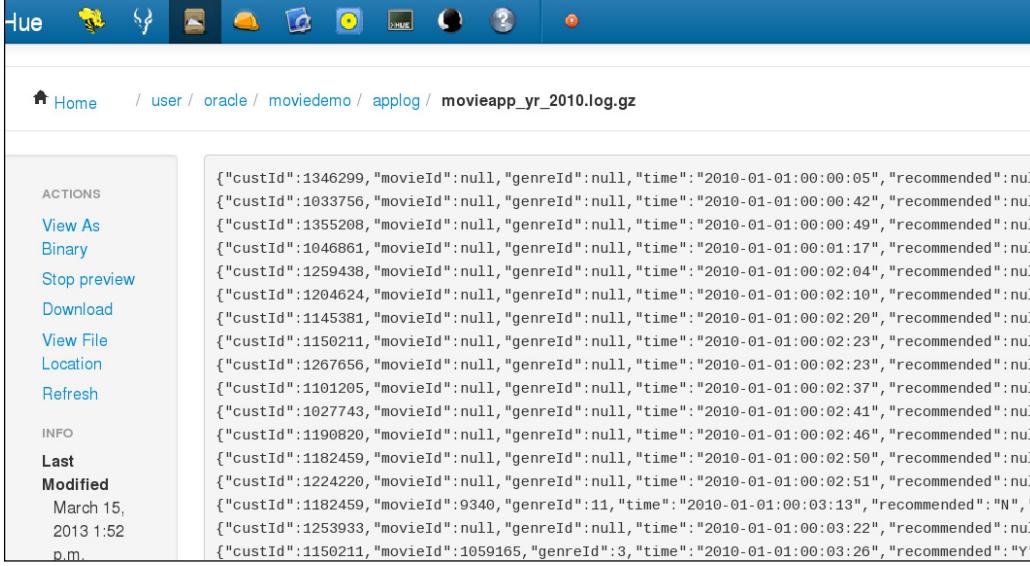
Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hive also supports analyzing large data sets that are stored in Hadoop-compatible file systems such as HDFS, Amazon FS, and so on. It uses a SQL-like query language called HiveQL to define and manipulate data.

It abstracts MapReduce code, and provides and preserves metadata.

Note: This lesson is an introduction to using Hive and Pig, and covers both at a very high level. It assumes that you have working knowledge of SQL and general database concepts.

Use Case: Storing Clickstream Data



The screenshot shows the Hue web interface. The top navigation bar has icons for Home, User, Oracle, MovieDemo, AppLog, and movieapp_yr_2010.log.gz. Below the navigation is a toolbar with actions like View As, Binary, Stop preview, Download, View File, Location, and Refresh. On the left, there's an 'ACTIONS' sidebar with options like View As, Binary, Stop preview, Download, View File, Location, and Refresh. The main content area displays a large JSON object representing a log file. The JSON structure is as follows:

```
{"custId":1346299,"movieId":null,"genreId":null,"time":"2010-01-01:00:00:05","recommended":null}, {"custId":1033756,"movieId":null,"genreId":null,"time":"2010-01-01:00:00:42","recommended":null}, {"custId":1355208,"movieId":null,"genreId":null,"time":"2010-01-01:00:00:49","recommended":null}, {"custId":1046861,"movieId":null,"genreId":null,"time":"2010-01-01:00:01:17","recommended":null}, {"custId":1259438,"movieId":null,"genreId":null,"time":"2010-01-01:00:02:04","recommended":null}, {"custId":1204624,"movieId":null,"genreId":null,"time":"2010-01-01:00:02:10","recommended":null}, {"custId":1145381,"movieId":null,"genreId":null,"time":"2010-01-01:00:02:20","recommended":null}, {"custId":1150211,"movieId":null,"genreId":null,"time":"2010-01-01:00:02:23","recommended":null}, {"custId":1267656,"movieId":null,"genreId":null,"time":"2010-01-01:00:02:23","recommended":null}, {"custId":1101205,"movieId":null,"genreId":null,"time":"2010-01-01:00:02:37","recommended":null}, {"custId":1027743,"movieId":null,"genreId":null,"time":"2010-01-01:00:02:41","recommended":null}, {"custId":1190820,"movieId":null,"genreId":null,"time":"2010-01-01:00:02:46","recommended":null}, {"custId":1182459,"movieId":null,"genreId":null,"time":"2010-01-01:00:02:50","recommended":null}, {"custId":1224220,"movieId":null,"genreId":null,"time":"2010-01-01:00:02:51","recommended":null}, {"custId":1182459,"movieId":9340,"genreId":11,"time":"2010-01-01:00:03:13","recommended":N}, {"custId":1253933,"movieId":null,"genreId":null,"time":"2010-01-01:00:03:22","recommended":null}, {"custId":1150211,"movieId":1059165,"genreId":3,"time":"2010-01-01:00:03:26","recommended":Y}
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

HDFS is commonly used to store large amounts of clickstream data. Individual clicks are not valuable by themselves. It is better to write queries over all clicks.

As you learn in the following slides, Hive is well suited to this usage.

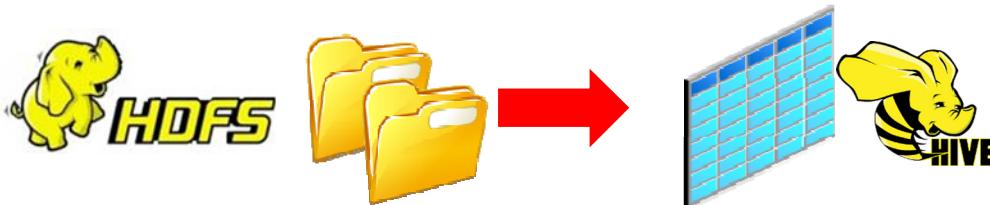
Defining Tables over HDFS

```

22
23 -- Create table over source JSON
24 CREATE EXTERNAL TABLE IF NOT EXISTS movieapp_log_json (
25   custId INT,
26   movieId INT,
27   genreId INT,
28   time STRING,
29   recommended STRING,
30   activity INT, ← Simple SQL syntax
31   rating INT, ← SerDe option
32   price FLOAT
33 )
34 ROW FORMAT SERDE 'org.apache.hadoop.hive.contrib.serde2.JsonSerde'
35 LOCATION '/user/oracle/moviedemo/applog/';

```

A table in Hive is mapped to HDFS directories



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hive enables the definition of tables over HDFS directories. The syntax is simple SQL.

SerDe enables Hive to deserialize data.

SerDe is short for Serializer/Deserializer. Hive uses the SerDe interface for IO. The interface handles both serialization and deserialization and also handles interpreting the results of serialization as individual fields for processing.

A SerDe allows Hive to read in data from a table, and writes it back out to HDFS in any custom format. Anyone can write their own SerDe for their own data formats.

For additional information about SerDe, see the following webpage:

<https://cwiki.apache.org/confluence/display/Hive/DeveloperGuide#DeveloperGuide-HiveSerDe>

For information on the Hive Data Manipulation Language (DML), see the following webpage:

<https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManual-DDL-RowFormat,StorageFormat, and SerDe>

Defining Tables over HDFS

```
{"custid":1185972,"movieid":null,"genreid":null,"time":"2012-07-01:00:00:07","recommended":null,"activity":8}
 {"custid":1354924,"movieid":1948,"genreid":9,"time":"2012-07-01:00:00:22","recommended":"N","activity":7}
 ...
```

```
CREATE EXTERNAL TABLE default.movieapp_log_json(
    custid int ,
    movieid int ,
    genreid int ,
    time string ,
    recommended string ,
    activity int ,
    rating int ,
    price float ,
    position int )
ROW FORMAT SERDE
    'org.apache.hive.hcatalog.data.JsonSerDe'
STORED AS INPUTFORMAT
    'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
    'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
    'hdfs://bigdatalite.localdomain:8020/user/oracle/moviework/applog_json'
```

*HiveQL (simple SQL-Like
SQL Syntax to query the
click stream data)*

SerDe option



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

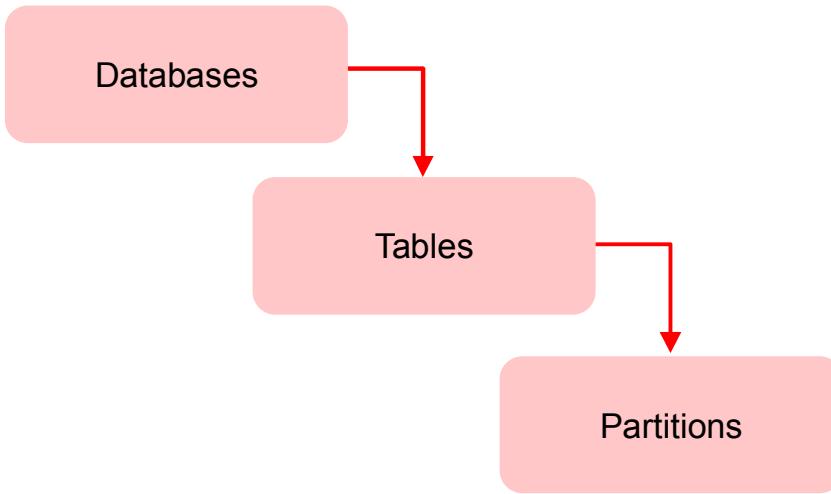
Hadoop processing is prefaced on a schema on read methodology. Oracle DB is schema on write—you are writing data to tables that have a particular structure. Schema on read defines the structure when it is reading the data.

So, how do you read this line when you use Hive? You create a table that defines a schema; it captures the metadata about how to locate the data, how to parse the data (that is, turn key-value attributes into column values) and how to write the data. Consider the table definition in the slide.

The INPUTFORMAT clause specifies the file type as text. The SERDE (serializer/deserializer) parses that row of text by using a Java class that has been written to interpret JSON data: `org.apache.hive.hcatalog.data.JsonSerDe`. This class gets a line of the file as input, converts the attributes into column values, and then returns the result as a row.

This Java class is a JSON parser—but you could use other classes as well. These other classes may interpret Avro Data, CSV, Parquet, and so on. You need to configure Hive so that it can find these SerDe classes and apply them to the data sets.

Hive: Data Units



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

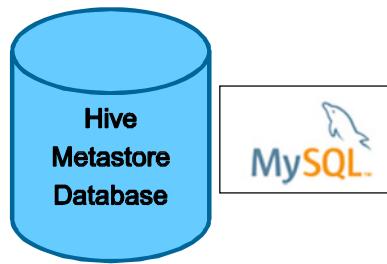
Hive's physical layout can be defined as a warehouse directory in HDFS that is divided into tables, and partitions.

Hive Tables are the same as RDBMS tables, which are made up of rows and columns. Hive sits on top of HDFS and therefore tables are mapped to directories in the HDFS file system.

Partitions: A Hive table can support one or more partitions. These partitions are mapped to subdirectories in the underlying file system.

The Hive Metastore Database

- Contains metadata regarding databases, tables, and partitions
- Contains information about how the rows and columns are delimited in the HDFS files that are used in the queries
- Is an RDBMS database, such as MySQL, where Hive persists table schemas and other system metadata



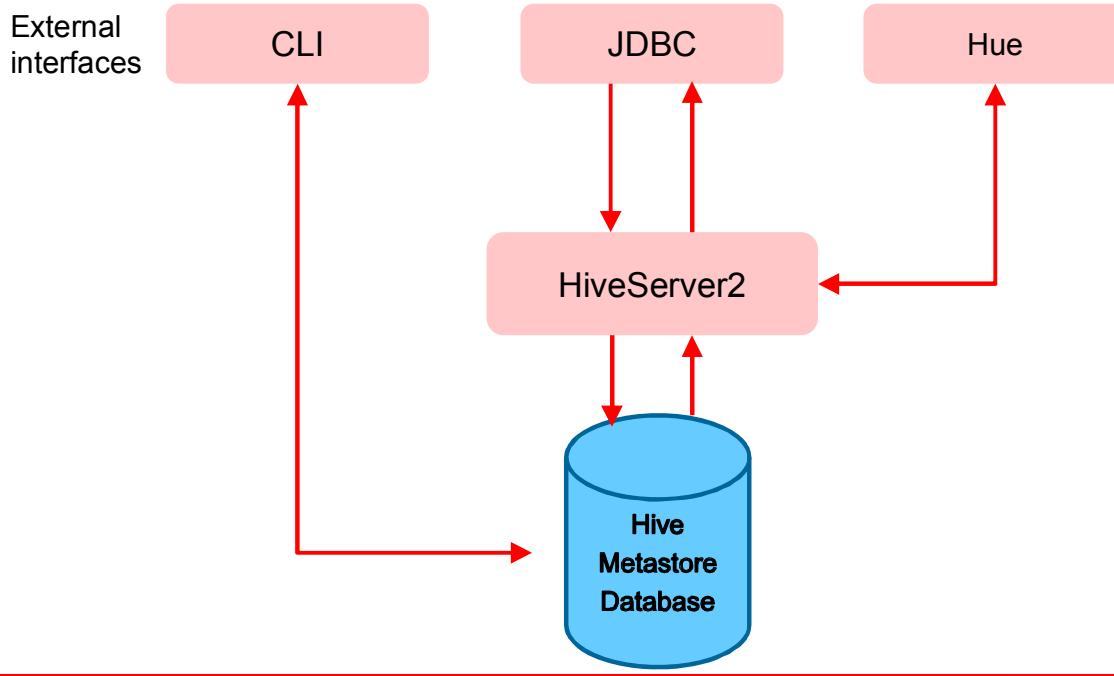
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The metastore contains metadata regarding tables, partitions, and databases. The metastore also contains information about how the rows and columns are delimited in the HDFS files. The metastore is an RDBMS database such as MySQL, the most popular Open Source SQL database management system. It is developed, distributed, and supported by Oracle Corporation.

For information about MySQL, see the MySQL website at <http://www.mysql.com/>.

Hive Framework



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

There are several ways to interact with Hive. Most hive interactions utilize HiveServer2 - which was introduced in CDH 4.1. HiveServer2 communicates to the Hive metastore to identify the databases and tables that are available to the user. It manages query requests - taking the HQL and compiling it into a series of MapReduce jobs. HiveServer2 is also the most secure way to process queries thru integration with Sentry - the authorization service.

The Hive CLI is an extremely common way of executing Hive queries. Note that it does not use HiveServer2. It communicates directly with the Hive Metastore and then uses an embedded compiler to generate and execute MR jobs. Because it doesn't use HiveServer2, it bypasses Sentry's authorization framework. Beeline is the functional replacement for the original CLI.

Creating a Hive Database

1. Start hive.

```
[oracle@localhost mapreduce]$ hive  
Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j.properties  
Hive history file=/tmp/oracle/hive_job_log_oracle_201302071749_169058549.txt  
hive> █
```

```
hive> create database moviework;  
OK  
Time taken: 4.288 seconds  
hive> █
```

2. Create the database.

```
hive> show databases;  
OK  
default  
moviedemo  
moviework  
Time taken: 1.281 seconds  
hive> █
```

3. Verify the database creation.

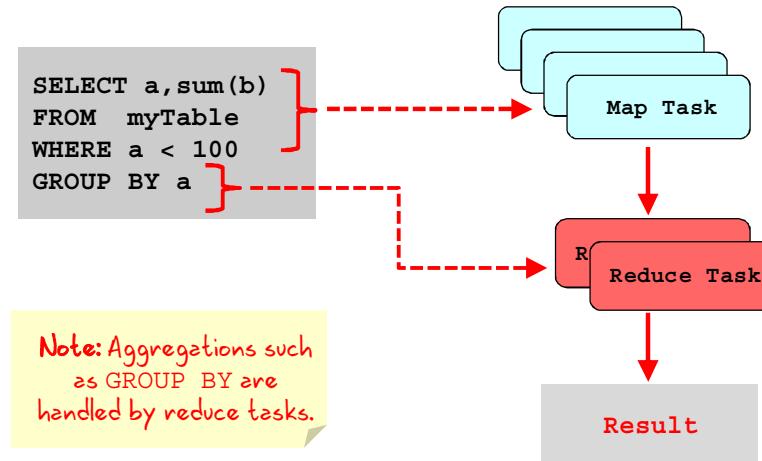
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

1. Open a terminal window and enter `hive`. The `hive` command prompt appears.
2. Create the database by executing the `create database` command.
3. Verify the creation of the database by executing the `show databases` command.

Data Manipulation in Hive

Hive SELECT with a WHERE clause:



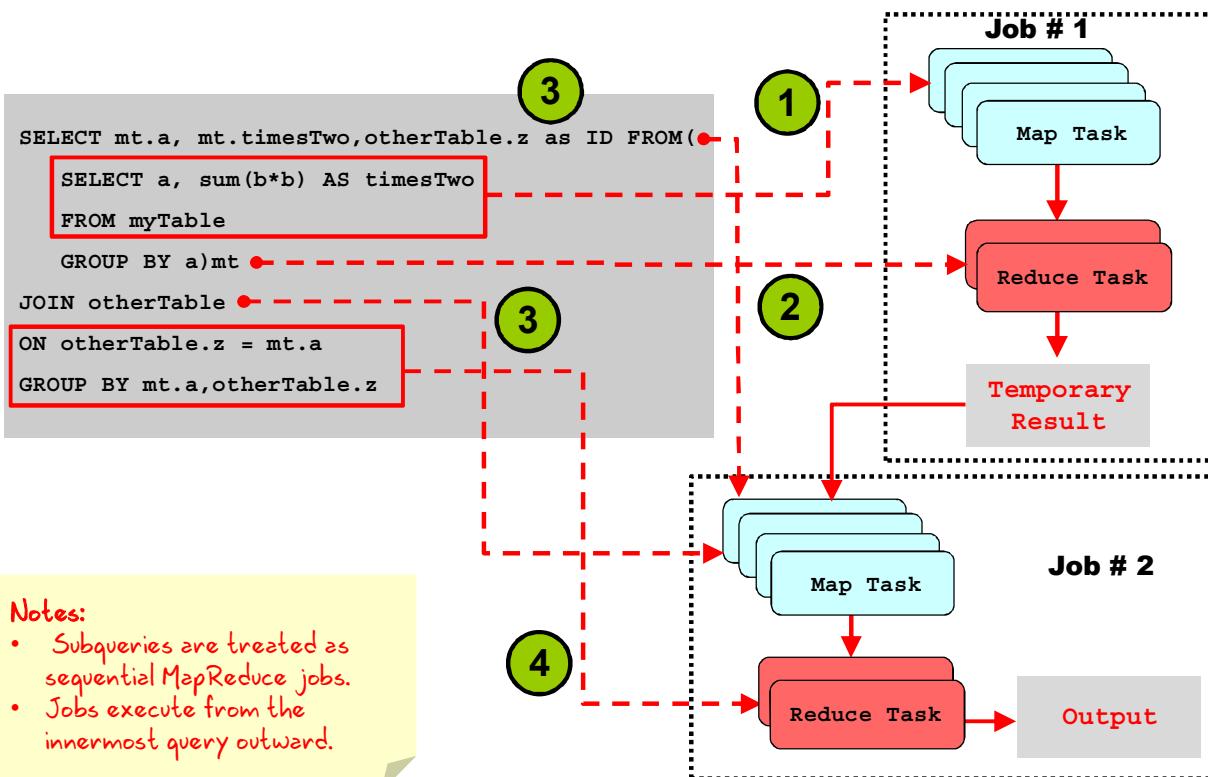
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hive maps queries to MapReduce jobs, simplifying the process of querying large data sets in HDFS. HiveQL statements can be mapped to phases of the MapReduce framework. As illustrated in the diagram in the slide, selections and transformations operations occur in Map tasks, whereas aggregations are handled by reducers.

Join operations are flexible. Depending on the size of the left table, they can be performed in the reducer or the mappers.

Data Manipulation in Hive: Nested Queries

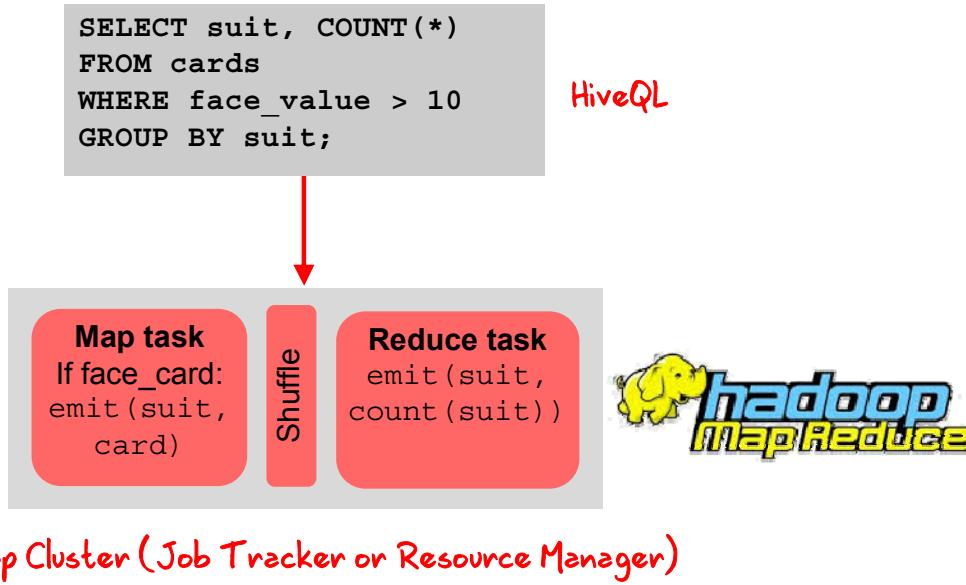


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hive queries that involve nested queries are translated into sequential MapReduce jobs that use temporary tables to store intermediate results. The diagram in the slide illustrates how statements in a nested query are transformed into map and reduce tasks. As in the preceding slide, dashed lines show logical mapping and solid lines define data flow.

Steps in a Hive Query



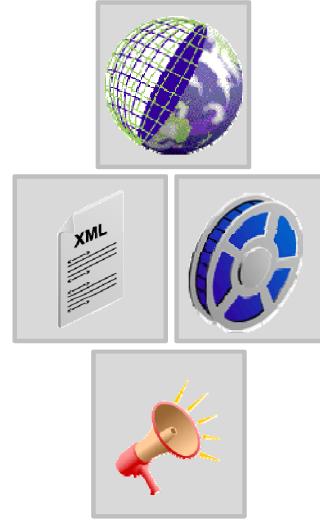
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

1. The Hive optimizer builds a MapReduce job.
2. Projections and predicates become Map code. Aggregations become Reduce code.
3. The job is submitted to MapReduce JobTracker or Resource Manager.

Hive-Based Applications

- Log processing
- Text mining
- Document indexing
- Business analytics
- Predictive modeling



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Hive is a powerful tool that is widely used by Hadoop users to create normal tables and external tables to load delimited and semistructured data. Hive also supports SQL joins and regular expressions. It is mainly used to execute MapReduce jobs for the applications listed in the slide.

Hive: Limitations

- No support for materialized views
- No transaction-level support
- Not ideal for ad hoc work
- Limited subquery support
- Subset of SQL-92
- Immature optimizer



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Pig: Overview

Pig:

- Is an open-source high-level data flow system
- Provides a simple language called Pig Latin for queries and data manipulation, which is compiled into map-reduce jobs that are run on Hadoop



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Pig Latin

- Is a high-level data flow language
- Provides common operations like join, group, sort, and so on
- Works on files in HDFS
- Was developed by Yahoo



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Pig Latin is a high-level language that completely abstracts the Hadoop system from users. It is a data-flow language rather than a procedural language. It was initially developed by Yahoo for internal use; the community then made further contributions and it is now open source.

Pig Latin effectively uses existing user code or libraries for complex, nonregular algorithms. It operates on files in HDFS.

Pig Applications

- Rapid prototyping of algorithms for processing large data sets
- Log analysis
- Ad hoc queries across large data sets
- Analytics and sampling
- PigMix: A set of performance and scalability benchmarks



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Running Pig Latin Statements

You can execute Pig Latin statements:

- Using the grunt shell or command line
- In MapReduce mode or local mode
- Either interactively or in batch

Pig processes Pig Latin statements as follows:

1. It validates the syntax and semantics of all statements.
2. If Pig encounters a DUMP or STORE, it executes the statements.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Pig Latin: Features

Ease of programming

Optimization opportunities

Extensibility

Structure that accommodates substantial parallelization



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

- **Ease of programming:** It is trivial to achieve parallel execution of simple, yet crucially parallel, data analysis tasks. Complex tasks comprising multiple interrelated data transformations are explicitly encoded as data flow sequences, making them easy to write, understand, and maintain.
- **Optimization opportunities:** The way in which tasks are encoded permits the system to optimize their execution automatically, enabling the user to focus on semantics rather than efficiency.
- **Extensibility:** Users can create their own functions to do special-purpose processing.

Working with Pig

1. Open a terminal window, type `pig`, and press Enter.

```
[oracle@localhost pig]$ pig
```

2. To execute scripts, use the grunt shell prompt.

```
grunt>  
grunt> dump monthly_cash_accounts;
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to:

- Define Hive
- Describe the Hive data flow
- Create a Hive database
- Define Pig
- List the features of Pig



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 11: Overview

This practice covers the following topics:

- Practice 11-1: Manipulating Data with Hive
- Practice 11-2: Extracting Facts by Using Hive
- Practice 11-3: Working with Pig Latin semantics



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

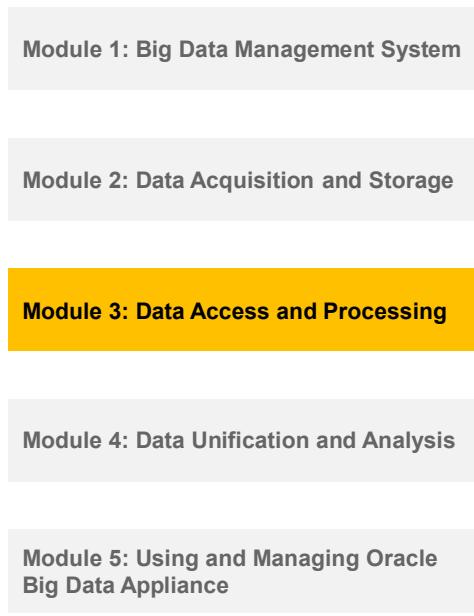
12

Overview of Cloudera Impala

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



Lesson 9: Introduction to MapReduce

Lesson 10: Resource Management Using YARN

Lesson 11: Overview of Apache Hive and Apache Pig

Lesson 12: Overview of Cloudera Impala

Lesson 13: Using Oracle XQuery for Hadoop

Lesson 14: Overview of Solr

Lesson 15: Apache Spark

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Cloudera Impala is yet another big data access method. This lesson provides a high-level overview of this technology.

Objectives

After completing this lesson, you should be able to:

- Describe the features of Cloudera Impala
- Explain how Impala works with Hive, HDFS, and HBase



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This lesson provides an overview of Cloudera Impala features, and its interaction with Hive, HDFS, and HBase.

Hadoop: Some Data Access/Processing Options

Component	Purpose
Hive	Puts a partial SQL interface in front of Hadoop. Includes a metadata “repository” called the Metastore.
Pig	A SQL-like scripting language on top of Java - for MapReduce programming
HBase	Applies a partial columnar scheme on top of Hadoop
HCatalog	A metadata layer to simplify access to data stored in Hadoop
Mahout	A set of data-mining algorithms
Impala	A database-like SQL layer on top of Hadoop



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Some of the various data access and processing options supported by Hadoop are listed in this slide. Impala is one of these options.

In a somewhat similar fashion to Hive and Pig, Impala is also a SQL-like layer on top of Hadoop.

Cloudera Impala

- The Impala server is a distributed, massively parallel processing (MPP) database engine.
- It consists of different daemon processes that run on specific hosts within your CDH cluster.
- The core Impala component is a daemon process that runs on each node of the cluster.
- SQL is the primary development language.



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The Impala server is a distributed, massively parallel processing (MPP) database engine. It consists of different daemon processes that run on specific hosts within your CDH cluster.

The core Impala component is a daemon process that runs on each node of the cluster, physically represented by the impala process. Each daemon process may perform the following:

- Read from and write to data files
- Accept queries transmitted from the impala-shell command, Hue, JDBC, or ODBC
- Parallelize queries and distribute work to other nodes in the Impala cluster
- Transmit intermediate query results back to the central coordinator node

SQL is the primary development language with Impala, but you can also use Java or other languages to interact with Impala by using the standard JDBC and ODBC interfaces.

For specialized kinds of analysis, you can supplement the SQL built-in functions by writing user-defined functions (UDFs) in C++ or Java.

Cloudera Impala: Key Features

- Open source and Apache-licensed
- MPP architecture
- Interactive analysis on data stored in HDFS and HBase
- Incorporates native Hadoop security
- Provides ANSI-92 SQL support
- Shares workload management with Apache
- Supports common Hadoop file formats



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A summary of Cloudera Impala key features is listed in the slide.

- Impala is open source software, and is licensed with Apache.
- Its massively parallel processing architecture helps to improve performance by leveraging Hadoop scalability.
- It allows you to perform a narrow range of interactive analysis on data stored in HDFS and HBase.
- It is integrated with native Hadoop security, including:
 - Kerberos for authentication
 - Apache Sentry for fine-grained, role-based authorization
- It has ANSI-92 SQL support with user-defined functions (UDFs).
- Impala also shares workload management, metadata, ODBC driver, SQL syntax, and user interface with Apache.
- It also supports common Hadoop file formats, as shown in the next slide.

Cloudera Impala: Supported Data Formats

Supported formats include:

- Text and SequenceFiles, which can be compressed as:
 - Snappy
 - GZIP
 - BZIP
- Avro
- RCFile
- LZO text file
- Parquet



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

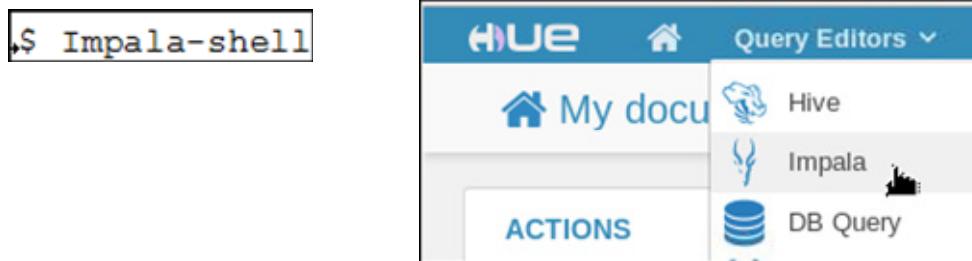
Impala supports common Hadoop file formats like text and SequenceFiles, which can be compressed in several different formats.

In addition, Impala supports Avro, RCFile, LZO, and Parquet formats as well.

Cloudera Impala: Programming Interfaces

You can connect and submit requests to the Impala daemons through:

- The Impala-shell interactive command interpreter
- The Apache Hue web-based user interface
- JDBC and ODBC



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

As shown in the slide, you can use Impala in heterogeneous environments, or with JDBC or ODBC applications running on non-Linux platforms.

Each Impala daemon process, running on separate nodes in a cluster, listens to several ports for incoming requests.

- Requests from Impala-shell and Hue are routed to the Impala daemons through the same port.
- For JDBC and ODBC requests, Impala daemons listen on separate ports.

How Impala Fits Into the Hadoop Ecosystem

Makes use of components within the Hadoop ecosystem:

- Provides a SQL layer on Hadoop
- May interchange data with other Hadoop components
- Can assist in ETL processes



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Impala makes use of familiar components within the Hadoop ecosystem. It:

- Provides a limited SQL layer on top of Hadoop
- Can interchange data with other Hadoop components as either a consumer or a producer
- May play a role within your ETL process or ETL pipelines

How Impala Works with Hive

- Uses existing Hive infrastructure
- Stores its table definitions in the Hive Metastore
- Accesses Hive tables
- Focuses on query performance
- Uses COMPUTE STATS statement instead of Hive ANALYZE TABLE statement



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

A primary Impala goal is to make SQL-on-Hadoop operations efficient enough to appeal to new categories of users and open up Hadoop to new types of use cases. Where practical, it makes use of existing Apache Hive infrastructure that many Hadoop users already have in place to perform long-running, batch-oriented SQL queries.

In particular, Impala keeps its table definitions in a traditional MySQL database known as the *Metastore*, the same database where Hive keeps this type of data. Thus, Impala can access tables defined or loaded by Hive, as long as all columns use Impala-supported data types, file formats, and compression codecs.

One limitation of Impala is that it can read more types of data (SELECT statement) than it can write (INSERT statement). Therefore, to query data using the Avro, RCFile, or SequenceFile file formats, you have to first load the data using Hive.

The Impala query optimizer can make use of table statistics and column statistics in Hive tables, but in Impala 1.2.2 and higher, you must use the Impala COMPUTE STATS statement instead of the Hive ANALYZE TABLE statement.

How Impala Works with HDFS and HBase

- HDFS
 - Impala's primary storage mechanism
 - Data stored as data files
- HBase
 - Alternative to HDFS to store Impala data
 - Impala table definition can be mapped to HBase tables

**ORACLE**

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Impala enables you to work with both HDFS and HBase data.

Impala and HDFS

Impala uses HDFS as its primary data storage medium. Impala relies on the redundancy provided by HDFS to guard against hardware or network outages on individual nodes. Impala table data is physically represented as data files in HDFS, using familiar HDFS file formats and compression codecs. When data files are present in the directory for a new table, Impala reads them all, regardless of file name. New data is added in files with names controlled by Impala.

Impala and HBase

HBase is an alternative to HDFS as a storage medium for Impala data. It is a database storage system built on top of HDFS, without built-in SQL support. By defining tables in Impala and mapping them to equivalent tables in HBase, you can query the contents of the HBase tables through Impala.

Summary of Cloudera Impala Benefits

- MPP performance (uses its own MPP query engine)
- Cost savings
- Analysis of raw and historical data
- Security



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Primary Impala benefits include:

- Performance equivalent to leading MPP databases, and 10+ faster than Apache Hive/Stinger
- Cost savings through reduced data movement, modeling, and storage
- Analysis of raw and historical data
- Security with Kerberos authentication and role-based authorization through the Apache Sentry project

Impala and Hadoop: Limitations

- Primarily used for ETL and archiving needs
- Not designed to support interactive analytics
- Limited SQL support
- Lack of optimizer



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Within the Hadoop ecosystem, Impala also has some limitations. These include:

- A limited feature set that primarily supports only ETL and archiving processes
- A lack of sophisticated analysis features
- Limited support for SQL, especially in terms of analytic techniques
- Lack of a query optimizer
- Dependency on MapReduce algorithms for aggregation

Summary

In this lesson, you should have learned to:

- Describe the features of Cloudera Impala
- Explain how Impala works with Hive, HDFS, and HBase



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

In this lesson, you learned about Cloudera Impala features, and its interaction with Hive, HDFS, and HBase.

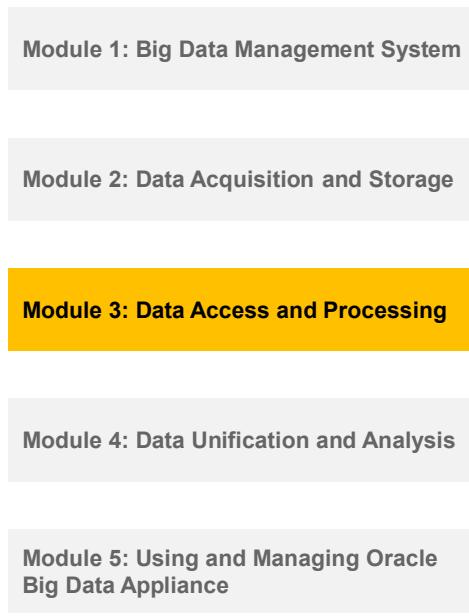
13

Using Oracle XQuery for Hadoop

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map



Lesson 9: Introduction to MapReduce

Lesson 10: Resource Management Using YARN

Lesson 11: Overview of Apache Hive and Apache Pig

Lesson 12: Overview of Cloudera Impala

Lesson 13: Using Oracle XQuery for Hadoop

Lesson 14: Overview of Solr

Lesson 15: Apache Spark

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The first part of this lesson reviews at a high level XML, XPath, and XQuery. In the second part of this lesson, you learn how to use Oracle XQuery for Hadoop to create and execute XQuery transformations.

Objectives

After completing this lesson, you should be able to use Oracle XQuery for Hadoop to create and execute XQuery transformations.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

XML

EXtensible Markup Language

XML is:

- A markup language that is similar to HTML
- Designed to carry data rather than to display data
- Designed to be self-descriptive
- A World Wide Web Consortium (W3C) recommendation
- The most common tool for data transmission among a wide variety of applications



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

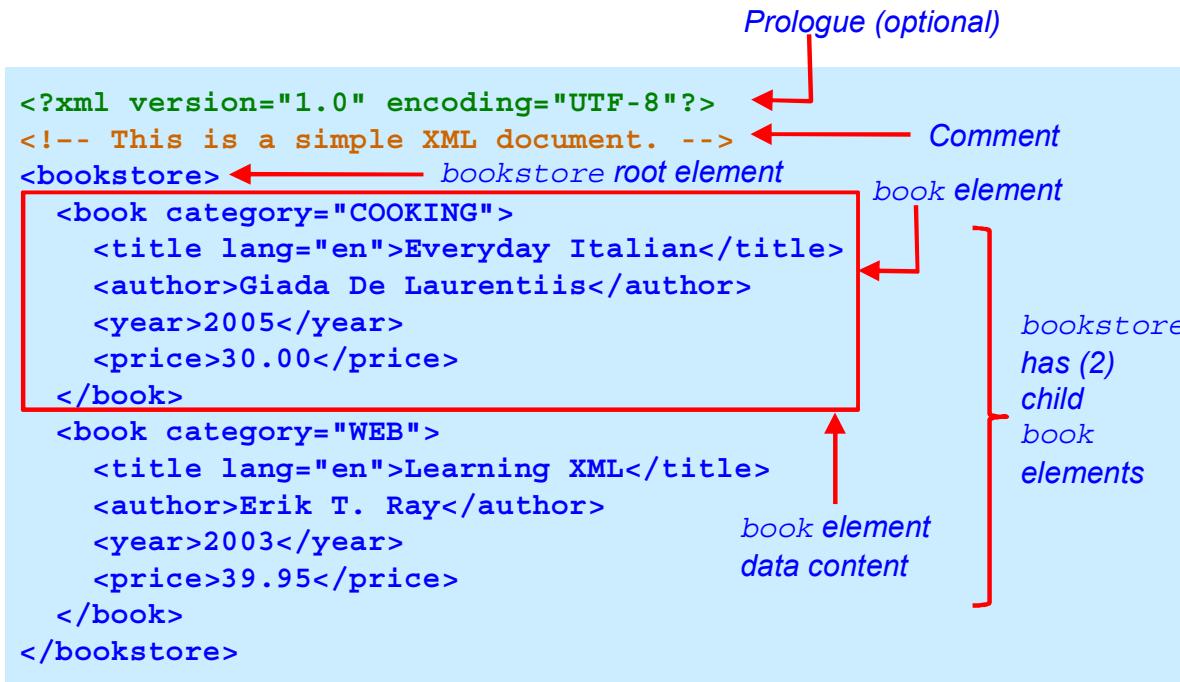
Here is a simple example of XML code:

```
<?xml version="1.0"?>  
  
<note>  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Let us meet this weekend!</body>  
</note>
```

The note is self-descriptive. It has sender and receiver information as well as a heading and a message body.

However, this XML document does not do anything. It is just information wrapped in tags. Someone must write a piece of software to send, receive, or display it.

Simple XML Document: Example



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The slide example of a simple XML document uses nested elements to describe bookstore book data. Elements are identified by tag names such as `book`, `title`, `author`, `year`, and `price`. Tag names are distinguishable as markup, rather than data, because they are surrounded by angle brackets (`<` and `>`). In XML, an element includes a start tag (`<bookstore>`), an end tag (`</bookstore>`), and all the markup and character data contained between those tags. Tag names are case-sensitive (and their case must be identical).

An XML document contains the following parts:

- The prologue, which may contain the following information: XML declaration, document type definition (DTD), which is required only to validate the document structure, and processing instructions and comments, which are optional
- The root element, which is also called the “document element” and contains all other elements
- Any child elements (two `book` elements in the slide example)
- An epilogue, which contains processing instructions and comments. Processing instructions give commands or information to an application that processes the XML data (not shown in the slide example).

XML Elements

- An XML element has a start tag, an end tag, and optional data content such as `employee` and `name`.
- Tag names are case-sensitive.
- Start and end tags must be identical.



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

- A start tag (for example, `<employee>`) includes:
 - The “`<`” character
 - A case-sensitive tag name (`employee`), without leading spaces
 - The “`>`” character
- An end tag (for example, `</employee>`) includes:
 - The “`<`” character
 - A case-sensitive tag name that must be identical to the start tag name, but prefixed with a slash. Leading spaces are not permitted.
 - The “`>`” character
- Data content: It can also contain elements such as `<name>` as seen in the slide example.

In summary, an XML element includes a start tag, an end tag, and everything in between.

Empty elements have no content between the start and end tags. In this case, a shortened form can be used where the start tag name is followed by a slash (for example, `<initials/>`).

Tag names are a descriptive term for an XML element and its content (for example, `employee`). The tag name is known as the *element type name*.

Markup Rules for Elements

- There is one root element, which is sometimes called the *top-level or document element*.
- All elements:
 - Must have matching start and end tags, or must be a self-closing tag (an empty element)
 - Can contain nested elements so that their tags do not overlap
 - Have case-sensitive tag names that are subject to naming conventions (starting with a letter, having no spaces, and not starting with the letters `xml`)
 - May contain white space (spaces, tabs, new lines, and combinations of these) that is considered part of the element data content



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Every XML document must contain one root element (top-level or document element). XML documents are hierarchical in structure with elements nested within others forming a document tree. The start and end tags for elements must not overlap. For example:

```
<employee>
    <first_name>Steven<last_name>king</first_name></last_name>
</employee>
```

Here, the `<last_name>` element overlaps the `<first_name>` element. This is not permissible. The correct form is:

```
<employee>
    <first_name>Steven</first_name><last_name>king</last_name>
</employee>
```

Element start and end tag names must be identical; that is, they are case-sensitive and they must be in the same case. For example, `<Employee>`, `<employee>`, and `<EMPLOYEE>` are distinct and have different tag names.

Element tag names must start with a letter or an underscore (`_`) but not with numeric or punctuation characters. Numeric, dash (`-`), and period (`.`) characters are allowed after the first letter, but not white space. Tag names cannot start with the `xml` letter sequence or any case-sensitive combination thereof, such as `XML` and `XmL`. White space, including new lines, are considered part of the data; that is, no stripping of white space is done. However, XML parsers treat end-of-line characters as a single line-feed.

XML Attributes

An XML attribute is a name-value pair that:

- Is specified in the start tag (after the tag name)

```
<bookstore>
  <book category="COOKING">
    <title lang='en'>Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
```

- Has a case-sensitive name
- Has a case-sensitive value that must be enclosed within matching *single or double quotation marks*
- Provides additional information about the XML document or XML elements



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Attributes are simple name-value pairs that are associated with a particular element. XML attributes must be specified after the start tag of an element or after the tag name of an empty element.

For example: `<book category="COOKING" />`

Attribute names are case-sensitive and follow the naming rules that apply to element names. In general, spaces are not used, but they are allowed on either side of the equal sign. Attribute names should be unique within the start tag.

The attribute values must be within matching quotation marks, either single or double. The example in the slide shows the `category` attribute value enclosed within double quotation marks and the `lang` attribute value within single quotation marks. In the latter case, the `'` entity must be used to include the apostrophe (single quotation mark) character in the name value.

Attributes provide additional information about the XML document's content or other XML elements. Attributes can be used for the following purposes:

- Describing how the XML document data is encoded or represented
- Indicating where the links or external resources are located
- Identifying and calling external processes such as applets and servlets
- Specifying an element instance in the document for facilitating a rapid search

Note: Attributes always have a value. For example, `name= ""` has an empty string value.

XML Path Language

XPath is a language for finding information (elements and attributes) in an XML document. XPath:

- Treats XML documents as trees of nodes
- Uses path expressions to select nodes or node-sets in an XML document
- Is named after the path notation it uses for navigating through the hierarchical structure of an XML document
- Uses a compact, non-XML syntax to form expressions for use in URI and XML attribute values
- Fully supports XML namespaces
- Is designed to be used by XML applications such as XSLT and XPointer
- Contains a library of standard functions



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

XML Path Language (XPath) is primarily used to address parts (nodes) of an XML document. XPath models an XML document as a tree of nodes, and expresses a navigation path through the hierarchical structure of an XML document. XPath:

- Is named after the path notation it uses for navigating through the structure of an XML document
- Uses a compact, non-XML syntax to form expressions that are used within URI and XML attribute values
- Facilitates the manipulation of string, number, and Boolean values
- Operates on the abstract, logical structure of an XML document, which is called the *document data model*, rather than its surface syntax

In addition to its use for addressing parts of an XML document, XPath fully supports XML namespaces and provides a natural expression language subset for pattern matching, which is extensively used in the W3C XSLT recommendation and XPointer. XPointer allows hyperlinks to point to specific parts (fragments) of XML documents.

Note: XSLT is XML Stylesheet Language Transformation, the language used to transform an XML document into another XML document. XPointer is XML Pointer Language, which is used as a fragment identifier for any uniform resource locator (URL) reference that locates a resource with a MIME type of `text/xml` or `application/xml`.

XPath Terminology: Node Types

- Element
- Attribute
- Text
- Namespace
- Processing instruction
- Comment
- Document



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

XPath Terminology: Family Relationships

Family Relationship	Description
Parent	Each element and attribute has one parent.
Children	Element nodes may have zero or more children.
Siblings	These are nodes that share the same parent.
Ancestors	These include a node's parent, parent's parent, and so on.
Descendants	These include a node's children, children's children, and so on.

```
<?xml version="1.0" encoding="UTF-8"?>
<bookshop>
  <book category="CHILDREN">
    <title>Love You Forever</title>
    <author>Robert Munsch</author>
    <year>1995</year>
    <price>4.98</price>
  </book>
</bookshop>
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Examples of family relationships in the XML document in the slide are as follows:

- The *child* of the `<bookshop>` element is the `<book>` element.
- The *children* of the `<book>` element are the `<title>`, `<author>`, and `<year>` elements.
- The `<title>`, `<author>`, `<year>`, and `<price>` elements are *siblings*.
- The *descendants* of the `<bookshop>` element are the `<book>`, `<title>`, `<author>`, `<year>`, and `<price>` elements.
- The *ancestors* of the `<title>`, `<author>`, `<year>`, and `<price>` elements are the `<book>` and `<bookshop>` elements.

XPath Expressions

An XPath expression:

- Is the primary construct in XPath
- Is evaluated to yield an object whose type can be:
 - A node-set, a Boolean, a number, or a string
- Is evaluated within a context (starting point) that includes:
 - A node called the *context node*
 - The context position and the context size
 - A function library
- Can be a location path (the most important and commonly used expression type)
 - An XPath expression that returns a node-set is called a *location path*



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The XPath language provides several kinds of expressions that may be constructed from keywords, symbols, and operands. Generally, an operand forms another kind of expression. An XPath expression is evaluated to yield an object of the following basic types:

- **A node-set:** An unordered collection of nodes, excluding duplicates
- **A Boolean:** A true or false result
- **A number:** A floating-point number
- **A string:** A sequence of Universal Coded Character Set (UCS) characters

XPath expressions are evaluated in a context, which consists of:

- A node, called the *context node*
- A pair of nonzero positive integers known as the context position and context size
- A set of variable bindings
- A function library
- A set of namespace declarations in scope for the expression

The location path is one of the most important XPath expressions that gives an application navigation directions in an XML document to locate a set of nodes. A **location path** can be either an **absolute location path**, which starts with a slash (/) and followed by a relative location path, or a **relative location path**, which is made up of a sequence of one or more **location steps** that are separated by a slash (/). Location steps are composed from left to right by selecting or navigating to a set of nodes, relative to the **context node**.

Location Path Expression: Example

Find the department_id of every department element that is a child of the departments element.

/departments/department/department_id

XPATH expression

```
<?xml version="1.0" encoding="UTF-8"?>
<departments>
  <department num="1">
    <department_id>10</department_id>
    <department_name>Administration</department_name>
  </department>
  <department num="2">
    <department_id>20</department_id>
    <department_name>Marketing</department_name>
  </department>
</departments>
```

XML

Results: 2 Nodes	
Node	Details
↳ department_id	
└ Text	10
↳ department_id	
└ Text	20

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The location path expression shown in the slide is an **absolute path** that uses the document root as the context node. An absolute location path always starts with a slash (/), and is followed by a relative location path that comprises the following location steps:

1. Go to the <departments> root element relative to the document root.
2. Go to the first child <department> element of the <departments> node.
3. Go to the <department_id> child element of the <department> node.

If the sample XML document includes more than one <department> element and child <department_id> element, the XPath expression matches multiple <department_id> nodes or a set of <department_id> nodes. The result is the entire element from the start to the end tags, plus the data between them. As shown in the highlighted portion of the slide, the result includes <department_id>10</department_id> and <department_id>20</department_id>.

XQuery: Review

- XQuery is the W3C standard language that is designed for querying (finding and extracting elements and attributes).
- XQuery is to XML what SQL is to database tables.
- *XQuery is built on XPath expressions.*
- XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators.
- By using XQuery, you can query both structured and unstructured data:
 - Relational databases
 - XML documents
 - Other data sources with XML data view



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

XQuery is the W3C standard language that is designed for querying, transforming, and accessing XML and relational data. XQuery is similar to SQL in many ways. Like SQL is designed for querying structured, relational data, XQuery is designed especially for querying unstructured, XML data from a variety of data sources.

You can use XQuery to query XML data wherever it is found, whether it is stored in the database tables, or available through web services. In addition to querying XML data, you can use XQuery to construct XML data.

Oracle XML DB supports a native XQuery compilation engine that can parse and compile XQuery expressions into SQL-native structures for evaluation. This native execution significantly improves the performance of XQuery expressions in Oracle Database.

XQuery Terminology

- Nodes
 - Elements, attribute, text, namespace
 - Processing-instructions, comments, document nodes
- Family Relationship
 - Parent, children, siblings, ancestors, and descendants



ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

XQuery Review: books.xml Document Example

```
<?xml version="1.0" encoding="UTF-8"?>
<bookshop>
  <book category="CHILDREN">
    <title language="en">Love You Forever</title>
    <author>Robert Munsch</author>
    <year>1995</year>
    <price>4.98</price>
  </book>
  <book category="COMPUTERS">
    <title language="en">XML: Visual QuickStart Guide</title>
    <author>Kevin Howard Goldberg</author>
    <year>2008</year>
    <price>23.17</price>
  </book>
  <book category="COMPUTERS">
    <title language="en">Beginning XML, 5th Edition</title>
    <author>Joe Fawcett</author>
    <author>Liam R.E. Quin</author>
    <author>Danny Ayers</author>
    <year>2012</year>
    <price>25.97</price>
  </book>
</bookshop>
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

FLWOR Expressions: Review

A FLWOR expression is one of the most important and powerful expressions in XQuery syntax. It supports iteration and binding of variables. It stands for:

- for
- let
- where
- order by
- return

Use the doc()
function to
open the
books.xml file

```
for $i in doc("books.xml")/bookshop/book
where $i/price > 20
order by $i/title
return $i/title
```

```
<?xml version="1.0" encoding="UTF-8"?>
<bookshop>
  <book category="CHILDREN">
    <title language="en">Love You Forever</title>
    <author>Robert Munsch</author>
    <year>1995</year>
    <price>4.98</price>
  </book>
  <book category="COMPUTERS">
    <title language="en">XML: Visual QuickStart Guide</title>
    <author>Kevin Howard Goldberg</author>
    <year>2008</year>
    <price>23.17</price>
  </book>
  <book category="COMPUTERS">
    <title language="en">Beginning XML, 5th Edition</title>
    <author>Joe Fawcett</author>
    <author>Liam R.E. Quin</author>
    <author>Denny Ayers</author>
    <year>2012</year>
    <price>25.97</price>
  </book>
</bookshop>
```

books.xml file

```
<title language="en">Beginning XML, 5th Edition</title>
<title language="en">XML: Visual QuickStart Guide</title>
```

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Expressions are the basic building blocks of XQuery. They are case-sensitive.

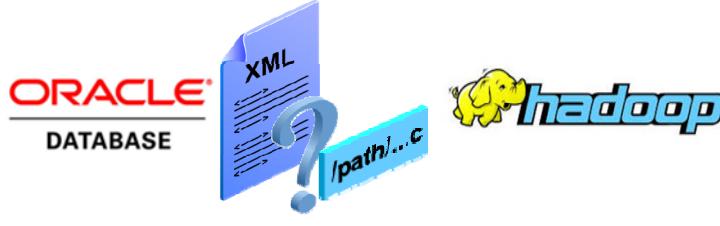
The acronym FLWOR (pronounced “flower”) represents the XQuery clauses for, let, where, order by, and return. A FLWOR expression has at least one for or let clause and a return clause; single where and order by clauses are optional.

- for: Analogous to the FROM clause of a SQL SELECT query. By using the for clause you can iterate across a range of sequence values and bind each one of them to one or more bind variables. At each iteration, the variables are bound in the order in which they appear.
- let: Analogous to the SQL SET statement. You can use the let clause to define variables and assign them, in turn, during iteration through a for clause. You can bind one or more variables. Just as with FOR, a variable can be bound by let to a value that is computed by using another variable that is listed previously in the binding list of let.
- where: Filters the for and let variable bindings according to a condition. This is similar to the SQL WHERE clause.
- order by: Arranges the where clause result in ascending or descending order

The for clause in the slide example assigns all the book elements under bookshop in the books.xml file to a variable named i. The where clause is used to filter out i so that it retains only those book elements with a price > 20. It then sorts the result and returns the title elements.

Oracle XQuery for Hadoop (OXH)

- Is a transformation engine for semistructured data that is stored in Apache Hadoop
- Runs transformations that are expressed in the XQuery language by translating them into a series of MapReduce jobs that are executed in parallel on the Hadoop cluster
- Loads data efficiently into Oracle Database by using Oracle Loader for Hadoop
- Provides extensions to Hive to support massive XML files
- Provides read and write support to Oracle NoSQL Database



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle XQuery for Hadoop is a transformation engine for semistructured big data. 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 an Apache Hadoop cluster. This enables you to focus on the data movement and the transformation logic, instead of the complexities of Java and MapReduce, without sacrificing scalability or performance.

The input data can be located in a file system accessible through the Hadoop File System API, such as the Hadoop Distributed File System (HDFS), or stored in Oracle NoSQL Database. Oracle XQuery for Hadoop can write the transformation results to Hadoop files, Oracle NoSQL Database, or Oracle Database.

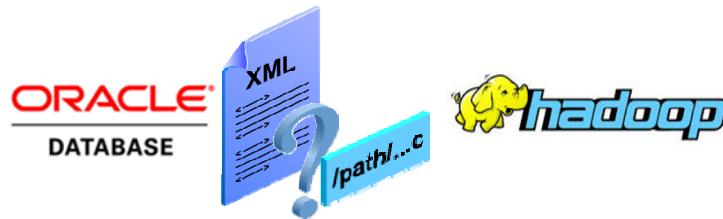
Oracle XQuery for Hadoop also provides extensions to Apache Hive to support massive XML files.

Oracle XQuery for Hadoop is based on mature industry standards including XPath, XQuery, and XQuery Update Facility. It is fully integrated with other Oracle products, which enables Oracle XQuery for Hadoop to:

- Load data efficiently into Oracle Database by using Oracle Loader for Hadoop
- Provide read and write support to Oracle NoSQL Database

OXH Features

- Scalable, native XQuery processing
- Tight integration with Hadoop data stores
- Parallel XML parsing
- Fast loading of XQuery results into the Oracle Database

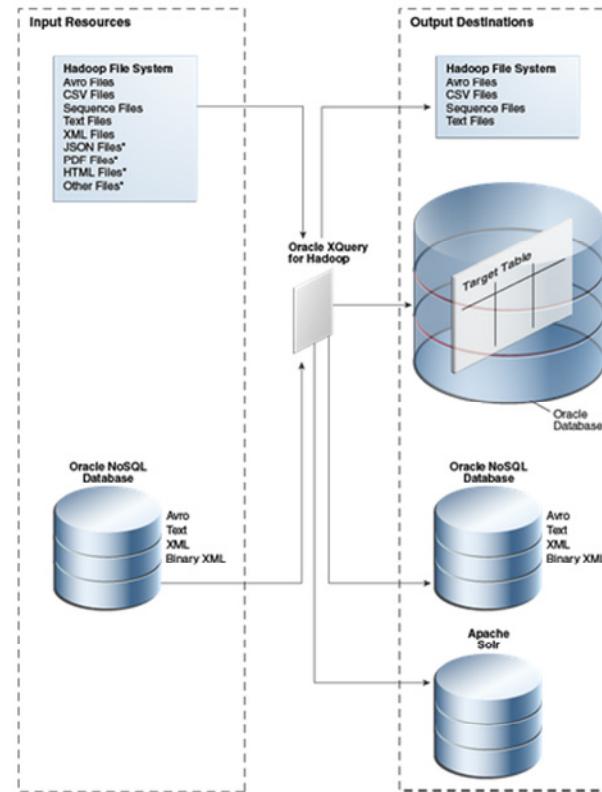


ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

- **Scalable, native XQuery processing:** XQuery engines are automatically distributed across the Hadoop cluster, allowing XQueries to be executed where the data is located.
- **Tight Integration with Hadoop data stores:** OXH can operate on content that is stored in HDFS, Hive, or Oracle NoSQL Database.
- **Parallel XML parsing:** Oracle's new parallel XML parser is integrated into the XQuery processor. This enables very large XML documents to be processed very efficiently by parsing them in parallel on multiple nodes.
- **Fast loading of XQuery results into Oracle Database:** OXH enables the results of XQuery operations to be loaded directly into Oracle Database by using Oracle Loader for Hadoop.

Oracle XQuery for Hadoop Data Flow



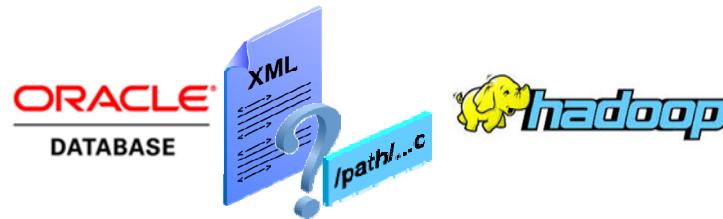
ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The diagram in the slide provides an overview of the data flow using Oracle XQuery for Hadoop.

Using OXH

1. Ensure that the software is installed and configured.
2. Log in to either a node in the Hadoop cluster or a system that is set up as a Hadoop client for the cluster.
3. Create an XQuery transformation that uses OXH functions. The transformation can use various adapters for input and output.
4. Execute the XQuery transformation.



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

OXH Installation

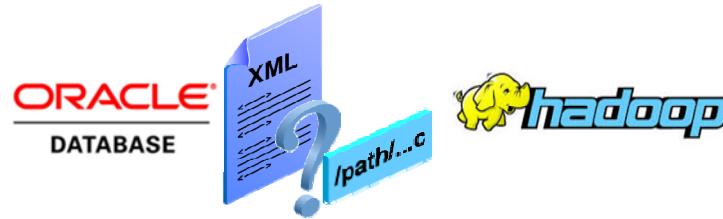
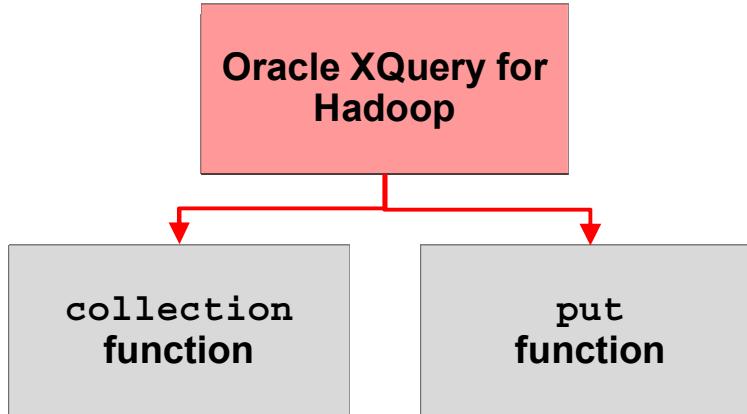
1. Unpack the contents of `oxh-version.zip` into the installation directory:
`$ unzip oxh-4.1.0-cdh-5.0.0.zip`
2. To support data loads into Oracle Database, install Oracle Loader for Hadoop.
3. (Optional) Set the following environment variables:
 - `OLH_HOME`: The Oracle Loader for Hadoop installation directory for access to Oracle Database
 - `KVHOME`: The Oracle NoSQL Database installation directory for access to Oracle NoSQL Database
 - `OXH_SOLR_MR_HOME`: The local directory containing Solr's `search-mr-version.jar` and `search-mr-version-job.jar`



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

- If you are using Oracle Big Data Appliance, OXH is already installed for you.
- If you are installing OXH on a third-party cluster, you must ensure that the following components are installed.
 - Java 6.x or 7.x
 - Cloudera's Distribution including Apache Hadoop Version 3 (CDH 3.3 or above) or Version 4 (CDH 4.1.2 or above)
 - Oracle NoSQL Database 2.x to support reading and writing to Oracle NoSQL Database
 - Oracle Loader for Hadoop 2.3 to support the writing of tables in Oracle databases

OXH Functions



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

OXH reads from and writes to big data sets by using `collection` and `put` functions:

- A `collection` function reads data from Hadoop files or Oracle NoSQL Database as a collection of items. A Hadoop file is one that is accessible through the Hadoop File System API. On Oracle Big Data Appliance and most Hadoop clusters, this file system is Hadoop Distributed File System (HDFS).
- A `put` function adds a single item to a data set that is stored in Oracle Database, Oracle NoSQL Database, or a Hadoop file.

OXH Adapters

Adaptor	Description
Avro File Adapter	The Avro file adapter provides access to Avro container files that are stored in HDFS. It includes <code>collection</code> and <code>put</code> functions for reading from and writing to Avro container files.
JSON File Adapter	The JSON file adapter provides access to JSON files stored in HDFS. It also contains functions for working with JSON data embedded in other file formats.
Oracle Database Adapter	The Oracle Database adapter loads data into Oracle Database.
Oracle NoSQL Database Adapter	The Oracle Database adapter loads data into Oracle Database.
Sequence File Adapter	The sequence file adapter provides access to Hadoop sequence files.
Text File Adapter	The text file adapter provides functions to read and write text files (such as CSV files) stored in HDFS.
XML File Adapter	The XML file adapter provides access to XML files stored in a Hadoop file system.
Solr Adapter	The Solr adapter provides functions to create full-text indexes and load them into Apache Solr servers.
Tika File Adapter	This adapter provides functions to parse files stored in HDFS in various formats using the Apache Tika library.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Oracle XQuery for Hadoop comes with a set of adapters that you can use to define put and collection functions for specific formats and sources. Each adapter has two components:

- A set of built-in put and collection functions that are predefined for your convenience.
- A set of XQuery function annotations that you can use to define custom put and collection functions.

Note: For detailed information about OXH adapters, see the *Oracle Big Data Connectors User's Guide Release 4 (4.1)* documentation.

Running a Query: Syntax

```
hadoop jar $OXH_HOME/lib/oxh.jar -conf job_config.xml query.xq -
    output_directory [-clean] [-ls] [-print]
    [-skiperrors] [-version]
```

Command Line Option	Description
\$OXH_HOME/lib/oxh.jar	Command to run OXH located in the \$OXH_HOME directory.
-conf job_config.xml	Identifies the job configuration file (see next page).
query.xq	Identifies the XQuery file name which contains the XQuery transformation.
-output_directory	Specifies the output directory.
-clean	Deletes all files from the output directory before running the query.
-ls	Lists the contents of the output directory.
-print	Prints the contents of all the files in the output directory.
-skiperrors	The errors are skipped but are counted and logged for the first 20 error messages after the query completes (see next page).
-version	Displays the Oracle XQuery for Hadoop version and exits without running a query.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

To run a query, you call the OXH utility by using the `hadoop jar` command. You can include any generic Hadoop command-line option. OXH implements the `org.apache.hadoop.util.Tool` interface and follows the standard Hadoop methods for building MapReduce applications.

- `query.xq` identifies the XQuery file.
- `-conf job_config.xml` is a generic Hadoop command option that identifies the job configuration file. (See the table on the next page)
- `-clean` deletes all files from the output directory before running the query.
- `-D` is a generic Hadoop command option that identifies a configuration property.
- `-ls` lists the content of the output directory after the query executes.
- `-output` directory specifies the name of the query output directory.
- `-print` prints the content of all files in the output directory to the standard output (your screen). For Avro files, each record prints as JSON text.
- `-skiperrors` enables error recovery so that an error does not halt processing. All errors that occur during query processing are counted, and the total is logged at the end of the query. The error messages of the first 20 errors for each task are also logged.
- `-version` displays the OXH version and exits without running a query.

OXH: Configuration Properties

Property	Description
oracle.hadoop.xquery.output	Sets the output directory for the query. The default value is /tmp/oxh-user_name/output.
oracle.hadoop.xquery.scratch	Sets the HDFS temp directory for Oracle XQuery for Hadoop to store temporary files. The default value is /tmp/oxh-user_name/scratch.
oracle.hadoop.xquery.timezone	The XQuery implicit time zone, which is used in a comparison or arithmetic operation when a date, time, or datetime value does not have a time zone
oracle.hadoop.xquery.skiperrors	Set to true to turn on error recovery, or set to false to stop processing when an error occurs. The default value is false.
oracle.hadoop.xquery.skiperrors.counters	Set to true (default) to group errors by error code, or set to false to report all errors in a single counter.
oracle.hadoop.xquery.skiperrors.max	Sets the maximum number of errors that a single MapReduce task can recover from. The default is Unlimited.
oracle.hadoop.xquery.skiperrors.log.max	Sets the maximum number of errors that a single MapReduce task logs. The default is 20.
log4j.logger.oracle.hadoop.xquery	Configures the log4j logger for each task with the specified threshold level. Set the property to one of these values: OFF, FATAL, ERROR, WARN, INFO, DEBUG, or ALL.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

`oracle.hadoop.xquery.output`: Sets the output directory for the query. This property is equivalent to the `-output` command-line option. The default value is `/tmp/oxh-user_name/output` where `user_name` is the name of the user running Oracle XQuery for Hadoop.

`oracle.hadoop.xquery.scratch`: Sets the HDFS temp directory for Oracle XQuery for Hadoop to store temporary files. The default value is `/tmp/oxh-user_name/scratch`.

`oracle.hadoop.xquery.timezone`: The XQuery implicit time zone, which is used in a comparison or arithmetic operation when a date, time, or datetime value does not have a time zone. The value must be in the format described by the Java `TimeZone` class.

`oracle.hadoop.xquery.skiperrors`: Set to true to turn on error recovery, or set to false (default) to stop processing when an error occurs

`oracle.hadoop.xquery.skiperrors.counters`: Set to true (default) to group errors by error code, or set to false to report all errors in a single counter

`oracle.hadoop.xquery.skiperrors.max`: Sets the maximum number of errors that a single MapReduce task can recover from. The default is Unlimited.

`oracle.hadoop.xquery.skiperrors.log.max`: Sets the maximum number of errors that a single MapReduce task logs. The default is 20.

`log4j.logger.oracle.hadoop.xquery`: Configures the log4j logger for each task with the specified threshold level. Set the property to one of these values: OFF, FATAL, ERROR, WARN, INFO, DEBUG, or ALL. If this property is not set, Oracle XQuery for Hadoop does not configure log4j.

XQuery Transformation and Basic Filtering: Example

mydata/visits1.log

```
2013-10-28T06:00:00, john, index.html, 200
2013-10-28T08:30:02, kelly, index.html, 200
2013-10-28T08:32:50, kelly, about.html, 200
2013-10-30T10:00:10, mike, index.html, 401
```

mydata/visits2.log

```
2013-10-30T10:00:01, john, index.html, 200
2013-10-30T10:05:20, john, about.html, 200
2013-11-01T08:00:08, laura, index.html, 200
2013-11-04T06:12:51, kelly, index.html, 200
2013-11-04T06:12:40, kelly, contact.html, 200
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The two HDFS text files (under the mydata HDFS directory) shown in the slide contain a log of visits to different webpages. Each line represents a visit to a webpage and contains the following information:

- time
- username
- page visited
- status code

We will use the two text files in the query on the next page.

XQuery Transformation and Basic Filtering: Example

The following query filters out pages that are visited by user kelly and writes those files to a text file:

`filtering_demo.xq`

```
import module "oxh:text";

for $line in text:collection("mydata/visits*.log")
let $split := fn:tokenize($line, "\s*,\s*")
where $split[2] eq "kelly"
return text:put($line)
```

To run the query, call the oxh utility using the hadoop jar command.

```
hadoop jar $OXH_HOME/lib/oxh.jar filtering_demo.xq -output
./myout -clean -ls -print
```

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The import module command makes additional built-in functions available to your query. In the example in the slide, it is importing the OXH text file adapter module. The text file adapter provides functions to read and write text files that are stored in HDFS.

The next part of the query uses the FLOWR expression with regular expressions. The code reads any file in the mydata HDFS directory that has a name starting with visits followed by any number of characters and that have a .log file extension.

The let clause uses the tokenize function with regular expressions. The function fn:tokenize breaks up a string into multiple strings based on the pattern argument. In this case the pattern is "\s*,\s*" which splits the string on commas (possibly surrounded by white space). For example, if the string \$line is "2013-10-28T06:00:00, john, index.html, 200", then \$split will be a sequence of four strings: "2013-10-28T06:00:00", "john", "index.html", "200").

The where \$split[2] eq "kelly" clause causes OXH to skip lines that do not satisfy the condition. In this case, it will skip any line where the second string in \$split is not equal to "kelly".

Finally, the return clause uses the text:put function to output the result.

XQuery Transformation and Basic Filtering: Example

```
[oracle@bigdatalite ~]$ hadoop fs -ls /user/oracle/mydata
Found 2 items
-rw-r--r-- 1 oracle oracle      175 2015-03-03 13:29 /user/oracle/mydata/visits1.log
-rw-r--r-- 1 oracle oracle     221 2015-03-03 13:29 /user/oracle/mydata/visits2.log
[oracle@bigdatalite ~]$ hadoop jar $OXH_HOME/lib/oxh.jar filtering_demo.xq -output ./myout -clean -ls -print
15/03/03 13:31:34 INFO hadoop.xquery: OXH: Oracle XQuery for Hadoop 4.0.1 (build 4.0.1-cdh5.0.0-mr1 @mr2). Copyright (c) 2015, Oracle. All rights reserved.
15/03/03 13:31:34 INFO hadoop.xquery: Executing query "filtering_demo.xq". Output path: "hdfs://bigdatalite.localdomain:8020/user/oracle/myout"
15/03/03 13:31:36 INFO hadoop.xquery: Submitting map-reduce job "oxh:filtering_demo.xq#0" id="fe88a6a8-79e7-438c-88a4-7587e0811200.0", inputs=[hdfs://bigdatalite.localdomain:8020/user/oracle/mydata/visits*.log], output=hdfs://bigdatalite.localdomain:8020/user/oracle/myout
15/03/03 13:31:37 INFO client.RMProxy: Connecting to ResourceManager at localhost/127.0.0.1:8032
15/03/03 13:31:37 INFO input.FileInputFormat: Total input paths to process : 2
15/03/03 13:31:38 INFO mapreduce.JobSubmitter: number of splits:2
15/03/03 13:31:38 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1425123851399_0007
15/03/03 13:31:38 INFO impl.YarnClientImpl: Submitted application application_1425123851399_0007
15/03/03 13:31:38 INFO mapreduce.Job: The url to track the job: http://bigdatalite.localdomain:8088/proxy/application_1425123851399_0007/
15/03/03 13:31:38 INFO hadoop.xquery: Waiting for map-reduce job oxh:filtering_demo.xq#0
15/03/03 13:31:38 INFO mapreduce.Job: Running job: job_1425123851399_0007
15/03/03 13:31:45 INFO mapreduce.Job: Job job_1425123851399_0007 running in uber mode : false
15/03/03 13:31:45 INFO mapreduce.Job: map 0% reduce 0%
15/03/03 13:31:55 INFO mapreduce.Job: map 50% reduce 0%
15/03/03 13:31:56 INFO mapreduce.Job: map 100% reduce 0%
15/03/03 13:31:56 INFO mapreduce.Job: Job job_1425123851399_0007 completed successfully
15/03/03 13:31:57 INFO mapreduce.Job: bytes written: 0
15/03/03 13:31:57 INFO hadoop.xquery: Finished executing "filtering_demo.xq". Output path: "hdfs://bigdatalite.localdomain:8020/user/oracle/myout"
Found 3 items
-rw-r--r-- 1 oracle oracle      0 2015-03-03 13:31 hdfs://bigdatalite.localdomain:8020/user/oracle/myout/_SUCCESS
-rw-r--r-- 1 oracle oracle     90 2015-03-03 13:31 hdfs://bigdatalite.localdomain:8020/user/oracle/myout/part-m-00000
-rw-r--r-- 1 oracle oracle     88 2015-03-03 13:31 hdfs://bigdatalite.localdomain:8020/user/oracle/myout/part-m-00001
2013-11-04T06:12:51, kelly, index.html, 200
2013-11-04T06:12:40, kelly, contact.html, 200
2013-10-28T08:30:02, kelly, index.html, 200
2013-10-28T08:32:50, kelly, about.html, 200
[oracle@bigdatalite ~]$
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Viewing the Completed Application in YARN

The screenshot shows the Loop web interface with the title "All Applications". It displays cluster and user metrics. Below is a table of completed applications:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress
application_1425123851399_0007	oracle	oxh:filtering_demo.xq#0	MAPREDUCE	root.oracle	Tue, 03 Mar 2015 18:31:38 GMT	Tue, 03 Mar 2015 18:31:55 GMT	FINISHED	SUCCEEDED	100%

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Calling Custom Java Functions from XQuery

```
%ora-java:binding("java.class.name [#method] ")
```

- `java.class.name` is the fully qualified name of a Java class that contains the implementation method.
- `method` is a Java method name. It is optional and defaults to the XQuery function name.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

OXH is extensible with custom external functions that are implemented in the Java language. A Java implementation must be a static method with the parameter and return types as defined by the “XQuery API for Java” (XQJ) specification.

A custom Java function binding is defined in OXH by annotating an external function definition with the `%ora-java:binding` annotation. This annotation has the syntax shown in the slide.

Additional Resources

The screenshot shows the Oracle Big Data Connectors documentation interface. On the left, there's a sidebar with links like 'Licensing Information', 'Software User's Guide', 'Enterprise Manager System Monitoring', 'Plug-in Installation Guide for Oracle Big Data Appliance', 'Connectors User's Guide' (which is highlighted with a red box and has a cursor icon over it), and 'Oracle Loader for Hadoop Java Example'. A red arrow points from this highlighted section to the 'Part III Oracle XQuery for Hadoop' section in the main 'Contents' area. The 'Contents' area lists several parts and their chapters:

- Title and Copyright Information
- Preface
- Changes in This Release for Oracle Big Data Connectors User's Guide
- Part I Setup
 - 1 Getting Started with Oracle Big Data Connectors
- Part II Oracle Database Connectors
 - 2 Oracle SQL Connector for Hadoop Distributed File System
 - 3 Oracle Loader for Hadoop
- Part III Oracle XQuery for Hadoop
 - 4 Using Oracle XQuery for Hadoop
 - 5 Oracle XQuery for Hadoop Reference
 - 6 Oracle XML Extensions for Hive
- Part IV Oracle R Advanced Analytics for Hadoop
 - 7 Using Oracle R Advanced Analytics for Hadoop
- Index

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to use Oracle XQuery for Hadoop to create and execute XQuery transformations.



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 13: Overview

This practice covers the following topics:

- Practice 13-1: Writing and Executing an XQuery Against Data in the Hadoop Cluster on HDFS
- Practice 13-2: Using OXH to Transform an XML File in Hive



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

14

Overview of Solr

ORACLE®

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Course Road Map

Module 1: Big Data Management System

Module 2: Data Acquisition and Storage

Module 3: Data Access and Processing

Module 4: Data Unification and Analysis

Module 5: Using and Managing Oracle Big Data Appliance



Lesson 9: Introduction to MapReduce

Lesson 10: Resource Management Using YARN

Lesson 11: Overview of Apache Hive and Apache Pig

Lesson 12: Overview of Cloudera Impala

Lesson 13: Using Oracle XQuery for Hadoop

Lesson 14: Overview of Solr

Lesson 15: Apache Spark

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

This lesson provides the overview of Apache Solr by using which you can do search on HDFS.

Objectives

After completing this lesson, you should be able to describe the overview of Cloudera Search (Solr).

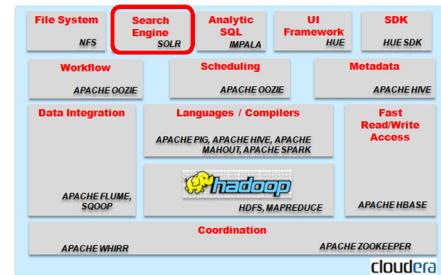


Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Apache Solr (Cloudera Search)

Solr, also called “Cloudera Search,” is a search facility for CDH. Using Search with the CDH infrastructure provides:

- Simplified infrastructure
- Better production visibility
- Quicker insights across various data types
- Quicker problem resolution
- Simplified interaction
- Scalability, flexibility, and reliability of search services



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Cloudera Search (Solr) is one of Cloudera's near-real-time access products. Cloudera Search enables nontechnical users to search and explore data stored in or ingested into Hadoop and HBase. Users do not need SQL or programming skills to use Cloudera Search because it provides a simple, full-text interface for searching.

Using Search with the CDH infrastructure provides:

- Simplified infrastructure
- Better production visibility
- Quicker insights across various data types
- Quicker problem resolution
- Simplified interaction with the ability to open the platform to more users and use cases
- Scalability, flexibility, and reliability of search services on the same platform as where you can execute other types of workloads on the same data

Cloudera Search: Key Capabilities

- Near-real-time indexing
- Batch indexing
- Simple, full-text data exploration
- Navigated drill down



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The main feature of Cloudera Search, compared to stand-alone search solutions, is the fully integrated data processing platform. Search uses the flexible, scalable, and robust storage system included with CDH. This eliminates the need to move larger data sets across infrastructures to address business tasks.

Cloudera Search incorporates Apache Solr, which includes Apache Lucene, SolrCloud, Apache Tika, and Solr Cell. Cloudera Search 1.x is tightly integrated with Cloudera's Distribution, including Apache Hadoop (CDH) and is included with CDH 5. Cloudera Search provides the key capabilities as mentioned in the slide.

Cloudera Search: Features

- Unified management and monitoring with Cloudera Manager
- Index storage in HDFS
- Batch index creation through MapReduce
- Real-time and scalable indexing at data ingest
- Easy interaction and data exploration through Hue
- Simplified data processing for search workloads
- HBase search



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Unified Management and Monitoring with Cloudera Manager

Cloudera Manager provides a unified and centralized management and monitoring experience for both CDH and Cloudera Search. Cloudera Manager simplifies deployment, configuration, and monitoring of your search services. This differs from many existing search solutions that lack management and monitoring capabilities and that fail to provide deep insight into utilization, system health, trending, and various other supportability aspects.

Index Storage in HDFS

Cloudera Search is integrated with HDFS for index storage. Indexes created by Solr/Lucene can be directly written in HDFS with the data, instead of to local disk, thereby providing fault tolerance and redundancy.

Cloudera has optimized Cloudera Search for fast read and write of indexes in HDFS while indexes are served and queried through standard Solr mechanisms. Also, because data and indexes are co-located, once data is found, processing does not require transport or separately managed storage.

Batch Index Creation through MapReduce

To facilitate index creation for large sets of data, Cloudera Search has built-in MapReduce jobs for indexing data stored in HDFS. As a result, the linear scalability of MapReduce is applied to the indexing pipeline.

Real-time and Scalable Indexing at Data Ingest

Cloudera Search provides integration with Flume to support near-real-time indexing. As new events pass through a Flume hierarchy and are written to HDFS, those same events can be written directly to Cloudera Search indexers.

In addition, Flume supports routing events, filtering, and adding annotations on data on its passage to CDH. These features work with Cloudera Search for improved index sharding, index separation, and document-level access control.

Easy Interaction and Data Exploration through Hue

A Cloudera Search GUI is provided as Hue plug-in, enabling users to interactively query data, view result files, and do faceted exploration. Hue can also schedule standing queries and explore index files. This GUI uses the Cloudera Search API, which is based on the standard Solr API.

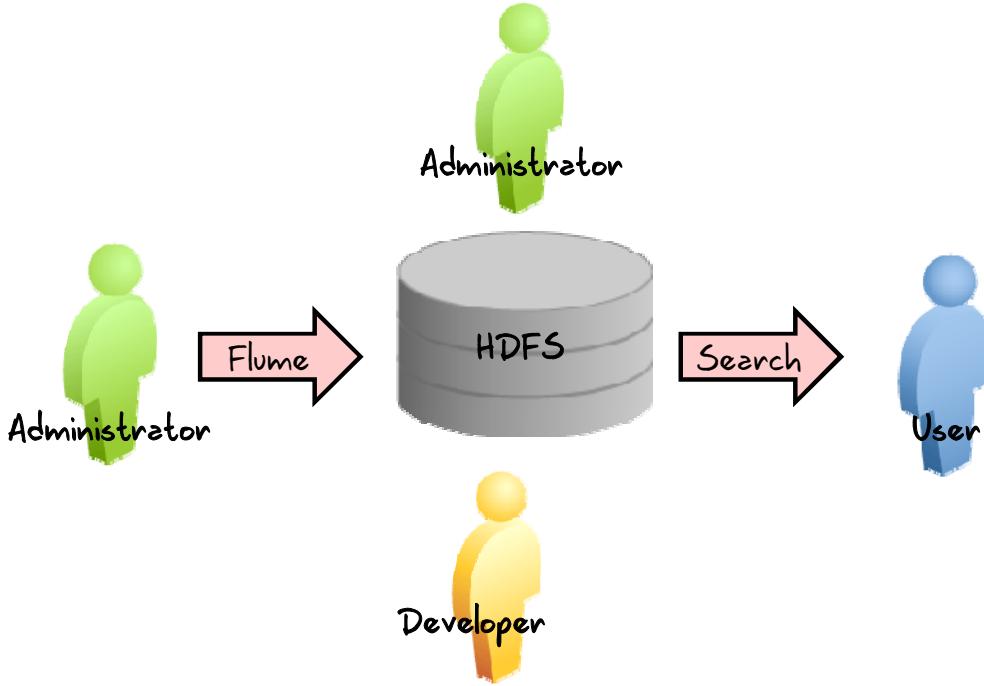
Simplified Data Processing for Search Workloads

Cloudera Search relies on Apache Tika for parsing and preparation of many of the standard file formats for indexing. Additionally, Cloudera Search supports Avro, Hadoop Sequence, and Snappy file format mappings, as well as support for Log file formats, JSON, XML, and HTML. Cloudera Search also provides data preprocessing by using Morphlines. This built-in support simplifies index configuration for these formats, which you can use for other applications such as MapReduce jobs.

HBase Search

Cloudera Search integrates with HBase, enabling full-text search of data stored in HBase. This functionality, which does not affect HBase performance, is based on a listener that monitors the replication event stream. The listener captures each write or update-replicated event, enabling extraction and mapping. The event is then sent directly to Solr indexers, deployed on HDFS, and written to indexes in HDFS, using the same process as for other indexing workloads of Cloudera Search. The indexes can then immediately be served, enabling near real-time search of HBase data.

Cloudera Search Tasks



ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

For content to be searchable, it must exist in CDH and be indexed. Content can either already exist in CDH and be indexed on demand or it can be updated and indexed continuously. The first step toward making content searchable is to ensure it is ingested or stored in CDH.

Ingestion

Content can be moved to CDH through techniques such as using:

- Flume, a flexible, agent-based data ingestion framework
- A copy utility, such as distcp for HDFS
- Sqoop, a structured data ingestion connector
- Fuse-dfs

In a typical environment, administrators establish systems for search. For example, HDFS is established to provide storage; Flume or distcp are established for content ingestion. Once administrators establish these services, users can use ingestion technologies such as file copy utilities or Flume sinks.

Indexing in Cloudera Search

Content must be indexed before it can be searched. Indexing includes a set of steps:

1. ETL steps—Extraction, Transformation, and Loading (ETL) is handled by using existing engines or frameworks such as Apache Tika or Cloudera Morphlines.
2. Content and metadata extraction
3. Schema mapping
4. Index creation, where the Indexes are created by Lucene



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Indexing involves mainly creation and serialization.

Indexes are typically stored on a local file system. Lucene supports additional index writers and readers. One such index interface is HDFS based and has been implemented as part of Apache Blur. This index interface has been integrated with Cloudera Search and modified to perform well with CDH-stored indexes. All index data in Cloudera Search is stored in HDFS and served from HDFS.

Types of Indexing

- Batch indexing using MapReduce
- Near Real Time (NRT) indexing using Flume
- NRT indexing using some other client that uses the NRT API
- Querying



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Batch Indexing Using MapReduce

To use MapReduce to index documents, documents are first written to HDFS. A MapReduce job can then be run on the content in HDFS, producing a Lucene index. The Lucene index is written to HDFS, and this index is subsequently used by search services to provide query results.

Batch indexing is most often used when bootstrapping a search cluster. The Map component of the MapReduce task parses input into indexable documents and the Reduce component contains an embedded Solr server that indexes the documents produced by the Map. A MapReduce-based indexing job can also be configured to use all assigned resources on the cluster, using multiple reducing steps for intermediate indexing and merging operations, with the last step of reduction being to write to the configured set of shard sets for the service. This makes the batch indexing process as scalable as MapReduce workloads.

Near Real Time (NRT) Indexing Using Flume

Cloudera Search includes a Flume sink that includes the option to directly write events to the indexer. This sink provides a flexible, scalable, fault tolerant, near real time (NRT) system for processing continuous streams of records, creating live-searchable, free-text search indexes. Typically, it is a matter of seconds from data ingestion by using the Flume sink to that content potentially appearing in search results, though this duration is tunable.

Search includes parsers for a set of standard data formats including Avro, CSV, Text, HTML, XML, PDF, Word, and Excel. While many formats are supported, you can extend the system by adding additional custom parsers for other file or data formats in the form of Tika plug-ins. Any type of data can be indexed: a record is a byte array of any format and parsers for any data format and any custom ETL logic can be established.

In addition, Cloudera Search comes with a simplifying Extract-Transform-Load framework called Cloudera Morphlines that can help adapt and preprocess data for indexing. This eliminates the need for specific parser deployments, replacing them with simple commands.

Cloudera Search has been designed to efficiently handle a variety of use cases.

Search supports routing to multiple Solr collections as a way of making a single set of servers support multiple user groups (multitenancy).

Search supports routing to multiple shards to improve scalability and reliability.

Index servers can be either co-located with live Solr servers serving end user queries or they can be deployed on separate commodity hardware, for improved scalability and reliability.

Indexing load can be spread across a large number of index servers for improved scalability, and indexing load can be replicated across multiple index servers for high availability.

NRT indexing using some other client that uses the NRT API

Documents written by a third-party directly to HDFS can trigger indexing using the Solr REST API. This API can be used to complete a number of steps:

1. Extract content from the document contained in HDFS where the document is referenced by a URL.
2. Map the content to fields in the search schema.
3. Create or update a Lucene index.

This could be useful if you do indexing as part of a larger workflow. For example, you might choose to trigger indexing from an Oozie workflow.

Querying

Once data has been made available as an index, the query API provided by the search service allows for direct queries to be executed, or facilitated through some third party, such as a command-line tool or graphical interface. Cloudera Search provides a simple UI application that can be deployed with Hue, but it is just as easy to develop a custom application, fitting your needs, based on the standard Solr API. Any application that works with Solr is compatible and runs as a search-serving application for Cloudera Search, as Solr is the core.

The solrctl Command

```
[oracle@bigdatalite /]$ solrctl --help
usage: /usr/bin/solrctl [options] command [command-arg] [command [command-arg]]
...
Options:
  --solr solr_uri
  --zk zk_ensemble
  --help
  --quiet

Commands:
  init      [--force]

  instancedir [--generate path [-schemaless]]
                [--create name path]
                [--update name path]
                [--get name path]
                [--delete name]
                [--list]

  collection  [--create name -s <numShards>
                [-c <collection.configName>]
                [-r <replicationFactor>]
                [-m <maxShardsPerNode>]
                [-n <createNodeSet>]
                [--delete name]
                [--reload name]
                [--stat name]
                [--deletedocs name]
                [--list]
```

Usage Options

Usage Example

```
[oracle@bigdatalite solr]$ solrctl instancedir --generate solr configs
[oracle@bigdatalite solr]$
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

The command-line utility `solrctl` is used to perform administrative operations on configuration bundles and Solr's collection. To understand the usage and the help information about `solrctl`, run `solrctl` with the `-help` option on the command line.

SchemaXML File

The schema.xml file contains the following sections:

- 1.Data Types
- 2.Fields
 - 1.Recommended fields
 - 2.Common field options
 - 3.Dynamic fields
 - 4.Indexing same data in multiple fields
 - 5.Expert field options
- 3.Miscellaneous Settings
 - 1.The Unique Key Field
 - 2.The Default Search Field
 - 3.Default query parser operator
 - 4.Copy Fields
 - 5.Similarity
 - 6.Poly Field Types
 - 7.Schema version attribute in the root node
 - 8.TODO



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

schema.xml is the first file you configure when setting up a new Solr installation.

The schema declares:

- What kinds of fields there are
- Which field should be used as the unique/primary key
- Which fields are required
- How to index and search each field

The schema.xml file contains all of the details about which fields your documents can contain, and how those fields must be dealt with when adding documents to the index, or when querying those fields.

For detailed information on schema.xml, visit the following link:

<http://wiki.apache.org/solr/SchemaXml>

Creating a Solr Collection

```
$ solrctl collection --create collection1 -s  
{ {numOfShards} }
```

Syntax

```
#  
  
export ZKHOSTS=bigdatalite.localdomain:2181/solr  
export ZKQUORUM=bigdatalite.localdomain:2181  
export COLLECTION=collection1  
export HDFS_DATA=/user/oracle/insur_cust_ltv_sample  
export HDFS_TMP=/user/oracle/oxh_temp  
  
.cleanup.sh  
  
solrctl instancedir --generate solr_configs  
solrctl instancedir --create collection1 solr_configs  
solrctl collection -create collection1 -s 1  
  
hadoop jar $OXH_HOME/lib/oxh.jar \  
-Dzkhosts=$ZKHOSTS -Dzkquorum=$ZKQUORUM -Dcollection=$COLLECTION \  
-Dhdfs_data=$HDFS_DATA -Dhdfs_tmp=$HDFS_TMP \  
-conf ./oxh_solr.xml oxh_solr.xq -clean -output $HDFS_TMP/oxh_solr  
  
"oxh_solr.sh" 24L, 685C
```

24,0-1

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

By default, the Solr server comes up with no collections. You must create your first collection by using instancedir that you provided to Solr during the configuration.

Shards: When your data is too large for one node, you can break it up and store it in sections by creating one or more shards. Each is a portion of the logical index, or core, and it's the set of all nodes containing that section of the index.

Using OXH with Solr

You can configure Oracle XQuery for Hadoop to use Solr installation by setting the environment variable to the local directory containing `search-mr-version.jar` and `search-mr-version-job.jar`.

Example:

```
$ export OXH_SOLR_MR_HOME="/usr/lib/solr/contrib/mr"
```



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Solr supports Solr indexing in OXH.

1. Make sure that Solr is installed and configured in your Hadoop cluster. Solr is included in Cloudera Search, which is installed automatically on Oracle Big Data Appliance.
2. Create a collection in your Solr installation into which you will load documents.
3. Configure OXH to use Solr.

Using Solr with Hue

The screenshot shows the Cloudera Hue interface. At the top, there is a navigation bar with tabs: HUE, Home, Query Editors, Data Browsers, Workflows, and Search. The 'Search' tab is highlighted. Below the navigation bar, the main area displays 'My documents'. A sub-menu for 'Search' is open, showing 'moviedemo' and 'Indexes'. A green circle with the number '1' is placed over the 'HUE' logo. A green circle with the number '2' is placed over the 'Indexes' option in the search sub-menu. A green circle with the number '3' is placed over the 'Solr Search' button in the top right corner of the main search area.

Solr Indexer

Collections

Filter collections...

Name

moviedemo

moviedemo_shard1_replica1

ORACLE

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

1. Open Cloudera Hue in the web browser. You can find the search tab.
2. The searchable directories are listed under the directories option in the Search tab.
3. You can choose to search by using the indexes or the data files.

Using Solr with Hue

The screenshot shows the Hue interface for managing Solr collections. The top navigation bar includes links for Query Editors, Data Browsers, Workflows, Search, and various system icons. The main title is "Solr Indexer". On the left, a sidebar titled "ACTIONS" contains "Search", "Index file", and "Delete" options. The main content area displays the "Collections > moviedemo" page. It shows a table of fields with their types and indexing/required status:

Name	Type	Unique key field	Required	Indexed	Stored
links	string		✓	✓	✓
text	text_general		✓		✓
year	int		✓	✓	✓
keywords	text_general		✓	✓	✓

Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Using Hue, you can work on indexing the collections that you have created in Solr.

Summary

In this lesson, you should have learned about the overview of Cloudera Search (Solr).



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

Practice 14: Overview

This practice covers the following topics:

- Indexing the “insur_cust_ltv_sample” data and making it searchable
- Using Solr with Hue



Copyright © 2015, Oracle and/or its affiliates. All rights reserved.

THESE eKIT MATERIALS ARE FOR YOUR USE IN THIS CLASSROOM ONLY. COPYING eKIT MATERIALS FROM THIS COMPUTER IS STRICTLY PROHIBITED

Oracle University and Error : You are not a Valid Partner use only