

ADDISON  
WESLEY  
DATA &  
ANALYTICS  
SERIES



# HADOOP<sup>®</sup> 2

## QUICK-START GUIDE

LEARN THE ESSENTIALS OF  
BIG DATA COMPUTING IN  
THE **APACHE HADOOP<sup>®</sup> 2**  
ECOSYSTEM

DOUGLAS EADLINE

FREE SAMPLE CHAPTER

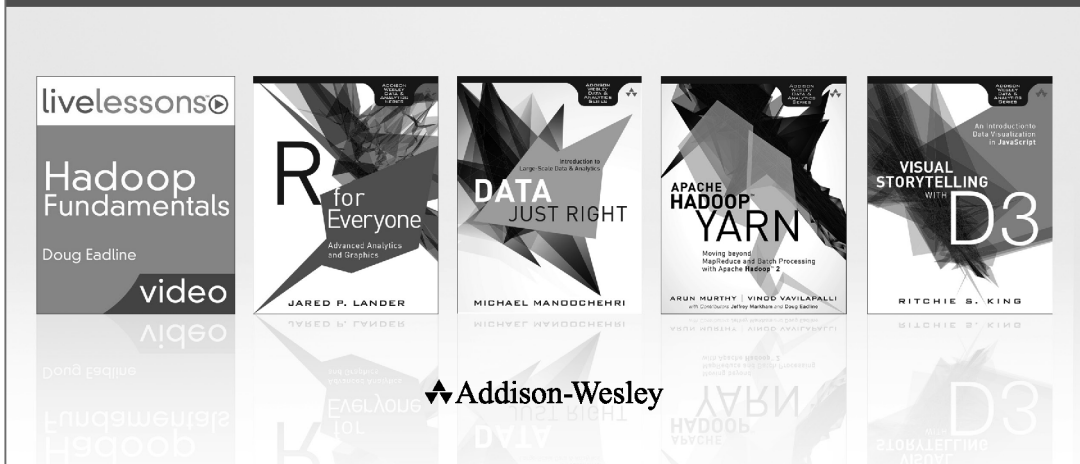


SHARE WITH OTHERS

# Hadoop<sup>®</sup> 2 Quick-Start Guide

---

# The Addison-Wesley Data and Analytics Series



Visit [informit.com/awdataseries](http://informit.com/awdataseries) for a complete list of available publications.

The Addison-Wesley Data and Analytics Series provides readers with practical knowledge for solving problems and answering questions with data. Titles in this series primarily focus on three areas:

1. **Infrastructure:** how to store, move, and manage data
2. **Algorithms:** how to mine intelligence or make predictions based on data
3. **Visualizations:** how to represent data and insights in a meaningful and compelling way

The series aims to tie all three of these areas together to help the reader build end-to-end systems for fighting spam; making recommendations; building personalization; detecting trends, patterns, or problems; and gaining insight from the data exhaust of systems and user interactions.



Make sure to connect with us!  
[informit.com/socialconnect](http://informit.com/socialconnect)

**informit.com**  
the trusted technology learning source

Addison-Wesley

**Safari**  
Books Online

# Hadoop<sup>®</sup> 2 Quick-Start Guide

---

Learn the Essentials of Big  
Data Computing in the Apache  
Hadoop<sup>®</sup> 2 Ecosystem

Douglas Eadline

◆ Addison-Wesley

New York • Boston • Indianapolis • San Francisco  
Toronto • Montreal • London • Munich • Paris • Madrid  
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at [corpsales@pearsoned.com](mailto:corpsales@pearsoned.com) or (800) 382-3419.

For government sales inquiries, please contact [governmentsales@pearsoned.com](mailto:governmentsales@pearsoned.com).

For questions about sales outside the United States, please contact [international@pearsoned.com](mailto:international@pearsoned.com).

Visit us on the Web: [informit.com/aw](http://informit.com/aw)

*Library of Congress Cataloging-in-Publication Data*

Eadline, Doug, 1956-author.

Learn the essential aspects of big data computing in the Apache Hadoop 2 ecosystem /  
Doug Eadline.

pages cm

Includes bibliographical references and index.

ISBN 978-0-13-404994-6 (pbk. : alk. paper)—ISBN 0-13-404994-2 (pbk. : alk. paper)

1. Big data. 2. Data mining. 3. Apache Hadoop. I. Title.

QA76.9.B45E24 2016

006.3'12—dc23

2015030746

Copyright © 2016 Pearson Education, Inc.

Apache®, Apache Hadoop®, and Hadoop® are trademarks of The Apache Software Foundation. Used with permission. No endorsement by The Apache Software Foundation is implied by the use of these marks.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, 200 Old Tappan Road, Old Tappan, New Jersey 07675, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-13-404994-6

ISBN-10: 0-13-404994-2

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.

First printing, November 2015

# Contents

<b>Foreword</b>	<b>xi</b>
<b>Preface</b>	<b>xiii</b>
<b>Acknowledgments</b>	<b>xix</b>
<b>About the Author</b>	<b>xxi</b>
<b>1 Background and Concepts</b>	<b>1</b>
Defining Apache Hadoop	<b>1</b>
A Brief History of Apache Hadoop	<b>3</b>
Defining Big Data	<b>4</b>
Hadoop as a Data Lake	<b>5</b>
Using Hadoop: Administrator, User, or Both	<b>6</b>
First There Was MapReduce	<b>7</b>
Apache Hadoop Design Principles	<b>7</b>
Apache Hadoop MapReduce Example	<b>8</b>
MapReduce Advantages	<b>10</b>
Apache Hadoop V1 MapReduce Operation	<b>11</b>
Moving Beyond MapReduce with Hadoop V2	<b>13</b>
Hadoop V2 YARN Operation Design	<b>13</b>
The Apache Hadoop Project Ecosystem	<b>15</b>
Summary and Additional Resources	<b>18</b>
<b>2 Installation Recipes</b>	<b>19</b>
Core Hadoop Services	<b>19</b>
Hadoop Configuration Files	<b>20</b>
Planning Your Resources	<b>21</b>
Hardware Choices	<b>21</b>
Software Choices	<b>22</b>
Installing on a Desktop or Laptop	<b>23</b>
Installing Hortonworks HDP 2.2 Sandbox	<b>23</b>
Installing Hadoop from Apache Sources	<b>29</b>
Installing Hadoop with Ambari	<b>40</b>
Performing an Ambari Installation	<b>42</b>
Undoing the Ambari Install	<b>55</b>
Installing Hadoop in the Cloud Using Apache Whirr	<b>56</b>
Step 1: Install Whirr	<b>57</b>
Step 2: Configure Whirr	<b>57</b>

Step 3: Launch the Cluster	59
Step 4: Take Down Your Cluster	61
Summary and Additional Resources	62

**3 Hadoop Distributed File System Basics 63**

Hadoop Distributed File System Design Features	63
HDFS Components	64
HDFS Block Replication	67
HDFS Safe Mode	68
Rack Awareness	68
NameNode High Availability	69
HDFS Namespace Federation	70
HDFS Checkpoints and Backups	71
HDFS Snapshots	71
HDFS NFS Gateway	72
HDFS User Commands	72
Brief HDFS Command Reference	72
General HDFS Commands	73
List Files in HDFS	75
Make a Directory in HDFS	76
Copy Files to HDFS	76
Copy Files from HDFS	76
Copy Files within HDFS	76
Delete a File within HDFS	76
Delete a Directory in HDFS	77
Get an HDFS Status Report	77
HDFS Web GUI	77
Using HDFS in Programs	77
HDFS Java Application Example	78
HDFS C Application Example	82
Summary and Additional Resources	83

**4 Running Example Programs and Benchmarks 85**

Running MapReduce Examples	85
Listing Available Examples	86

Running the Pi Example	<b>87</b>
Using the Web GUI to Monitor Examples	<b>89</b>
Running Basic Hadoop Benchmarks	<b>95</b>
Running the Terasort Test	<b>95</b>
Running the TestDFSIO Benchmark	<b>96</b>
Managing Hadoop MapReduce Jobs	<b>97</b>
Summary and Additional Resources	<b>98</b>
<b>5 Hadoop MapReduce Framework</b>	<b>101</b>
The MapReduce Model	<b>101</b>
MapReduce Parallel Data Flow	<b>104</b>
Fault Tolerance and Speculative Execution	<b>107</b>
Speculative Execution	<b>108</b>
Hadoop MapReduce Hardware	<b>108</b>
Summary and Additional Resources	<b>109</b>
<b>6 MapReduce Programming</b>	<b>111</b>
Compiling and Running the Hadoop WordCount Example	<b>111</b>
Using the Streaming Interface	<b>116</b>
Using the Pipes Interface	<b>119</b>
Compiling and Running the Hadoop Grep Chaining Example	<b>121</b>
Debugging MapReduce	<b>124</b>
Listing, Killing, and Job Status	<b>125</b>
Hadoop Log Management	<b>125</b>
Summary and Additional Resources	<b>128</b>
<b>7 Essential Hadoop Tools</b>	<b>131</b>
Using Apache Pig	<b>131</b>
Pig Example Walk-Through	<b>132</b>
Using Apache Hive	<b>134</b>
Hive Example Walk-Through	<b>134</b>
A More Advanced Hive Example	<b>136</b>
Using Apache Sqoop to Acquire Relational Data	<b>139</b>
Apache Sqoop Import and Export Methods	<b>139</b>
Apache Sqoop Version Changes	<b>140</b>
Sqoop Example Walk-Through	<b>142</b>



Using Apache Flume to Acquire Data Streams	<b>148</b>
Flume Example Walk-Through	<b>151</b>
Manage Hadoop Workflows with Apache Oozie	<b>154</b>
Oozie Example Walk-Through	<b>156</b>
Using Apache HBase	<b>163</b>
HBase Data Model Overview	<b>164</b>
HBase Example Walk-Through	<b>164</b>
Summary and Additional Resources	<b>169</b>
<b>8 Hadoop YARN Applications</b>	<b>171</b>
YARN Distributed-Shell	<b>171</b>
Using the YARN Distributed-Shell	<b>172</b>
A Simple Example	<b>174</b>
Using More Containers	<b>175</b>
Distributed-Shell Examples with Shell Arguments	<b>176</b>
Structure of YARN Applications	<b>178</b>
YARN Application Frameworks	<b>179</b>
Distributed-Shell	<b>180</b>
Hadoop MapReduce	<b>181</b>
Apache Tez	<b>181</b>
Apache Giraph	<b>181</b>
Hoya: HBase on YARN	<b>181</b>
Dryad on YARN	<b>182</b>
Apache Spark	<b>182</b>
Apache Storm	<b>182</b>
Apache REEF: Retainable Evaluator Execution Framework	<b>182</b>
Hamster: Hadoop and MPI on the Same Cluster	<b>183</b>
Apache Flink: Scalable Batch and Stream Data Processing	<b>183</b>
Apache Slider: Dynamic Application Management	<b>183</b>
Summary and Additional Resources	<b>184</b>
<b>9 Managing Hadoop with Apache Ambari</b>	<b>185</b>
Quick Tour of Apache Ambari	<b>186</b>
Dashboard View	<b>186</b>

Services View	<b>189</b>
Hosts View	<b>191</b>
Admin View	<b>193</b>
Views View	<b>193</b>
Admin Pull-Down Menu	<b>194</b>
Managing Hadoop Services	<b>194</b>
Changing Hadoop Properties	<b>198</b>
Summary and Additional Resources	<b>204</b>
<b>10 Basic Hadoop Administration Procedures</b>	<b>205</b>
Basic Hadoop YARN Administration	<b>206</b>
Decommissioning YARN Nodes	<b>206</b>
YARN WebProxy	<b>206</b>
Using the JobHistoryServer	<b>207</b>
Managing YARN Jobs	<b>207</b>
Setting Container Memory	<b>207</b>
Setting Container Cores	<b>208</b>
Setting MapReduce Properties	<b>208</b>
Basic HDFS Administration	<b>208</b>
The NameNode User Interface	<b>208</b>
Adding Users to HDFS	<b>211</b>
Perform an FSCK on HDFS	<b>212</b>
Balancing HDFS	<b>213</b>
HDFS Safe Mode	<b>214</b>
Decommissioning HDFS Nodes	<b>214</b>
SecondaryNameNode	<b>214</b>
HDFS Snapshots	<b>215</b>
Configuring an NFSv3 Gateway to HDFS	<b>217</b>
Capacity Scheduler Background	<b>220</b>
Hadoop Version 2 MapReduce Compatibility	<b>222</b>
Enabling ApplicationMaster Restarts	<b>222</b>
Calculating the Capacity of a Node	<b>222</b>
Running Hadoop Version 1 Applications	<b>224</b>
Summary and Additional Resources	<b>225</b>
<b>A Book Webpage and Code Download</b>	<b>227</b>

<b>B</b>	<b>Getting Started Flowchart and Troubleshooting Guide</b>	<b>229</b>
	Getting Started Flowchart	<b>229</b>
	General Hadoop Troubleshooting Guide	<b>229</b>
	Rule 1: Don't Panic	<b>229</b>
	Rule 2: Install and Use Ambari	<b>234</b>
	Rule 3: Check the Logs	<b>234</b>
	Rule 4: Simplify the Situation	<b>235</b>
	Rule 5: Ask the Internet	<b>235</b>
	Other Helpful Tips	<b>235</b>
<b>C</b>	<b>Summary of Apache Hadoop Resources by Topic</b>	<b>243</b>
	General Hadoop Information	<b>243</b>
	Hadoop Installation Recipes	<b>243</b>
	HDFS	<b>244</b>
	Examples	<b>244</b>
	MapReduce	<b>245</b>
	MapReduce Programming	<b>245</b>
	Essential Tools	<b>245</b>
	YARN Application Frameworks	<b>246</b>
	Ambari Administration	<b>246</b>
	Basic Hadoop Administration	<b>247</b>
<b>D</b>	<b>Installing the Hue Hadoop GUI</b>	<b>249</b>
	Hue Installation	<b>249</b>
	Steps Performed with Ambari	<b>250</b>
	Install and Configure Hue	<b>252</b>
	Starting Hue	<b>253</b>
	Hue User Interface	<b>253</b>
<b>E</b>	<b>Installing Apache Spark</b>	<b>257</b>
	Spark Installation on a Cluster	<b>257</b>
	Starting Spark across the Cluster	<b>258</b>
	Installing and Starting Spark on the Pseudo-distributed Single-Node Installation	<b>260</b>
	Run Spark Examples	<b>260</b>
	<b>Index</b>	<b>261</b>

# Foreword

Apache Hadoop 2 introduced new methods of processing and working with data that moved beyond the basic MapReduce paradigm of the original Hadoop implementation. Whether you are a newcomer to Hadoop or a seasoned professional who has worked with the previous version, this book provides a fantastic introduction to the concepts and tools within Hadoop 2.

Over the past few years, many projects have fallen under the umbrella of the original Hadoop project to make storing, processing, and collecting large quantities easier while integrating with the original Hadoop project. This book introduces many of these projects in the larger Hadoop ecosystem, giving readers the high-level basics to get them started using tools that fit their needs.

Doug Eadline adapted much of this material from his very popular video series *Hadoop Fundamentals Live Lessons*. However, his qualifications don't stop there. As a coauthor on the in-depth book *Apache Hadoop™ YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop™ 2*, few are as well qualified to deliver coverage of Hadoop 2 and the new features it brings to users.

I'm excited about the great wealth of knowledge that Doug has brought to the series with his books covering Hadoop and its related projects. This book will be a great resource for both newcomers looking to learn more about the problems that Hadoop can help them solve and for existing users looking to learn about the benefits of upgrading to the new version.

—Paul Dix, Series Editor



# Preface

Apache Hadoop 2 has changed the data analytics landscape. The Hadoop 2 ecosystem has moved beyond a single MapReduce data processing methodology and framework. That is, Hadoop version 2 offers the Hadoop version 1 methodology to almost any type of data processing and provides full backward compatibility with the vulnerable MapReduce paradigm from version 1.

This change has already had a dramatic effect on many areas of data processing and data analytics. The increased volume of online data has invited new and scalable approaches to data analytics. As discussed in Chapter 1, the concept of the Hadoop data lake represents a paradigm shift away from many established approaches to online data usage and storage. A Hadoop version 2 installation is an *extensible platform* that can grow and adapt as both data volumes increase and new processing models become available.

For this reason, the “Hadoop approach” is important and should not be dismissed as a simple “one-trick pony” for Big Data applications. In addition, the open source nature of Hadoop and much of the surrounding ecosystem provides an important incentive for adoption. Thanks to the Apache Software Foundation (ASF), Hadoop has always been an open source project whose inner workings are available to anyone. The open model has allowed vendors and users to share a common goal without lock-in or legal barriers that might otherwise splinter a huge and important project such as Hadoop. All software used in this book is open source and is freely available. Links leading to the software are provided at the end of each chapter and in Appendix C.

## Focus of the Book

As the title implies, this book is a quick-start guide to Hadoop version 2. By design, most topics are summarized, illustrated with an example, and left a bit unfinished. Indeed, many of the tools and subjects covered here are treated elsewhere as completely independent books. Thus, the biggest hurdle in creating a quick-start guide is deciding what *not* to include while simultaneously giving the reader a sense of what is important.

To this end, all topics are designed with what I call the hello-world.c experience. That is, provide some background on what the tool or service does, then provide a beginning-to-end example that allows the reader to get started quickly, and finally, provide resources where additional information and more nitty-gritty details can be

found. This approach allows the reader to make changes and implement variations that move away from the simple working example to something that solves the reader's particular problem. For most of us, our programming experience started from applying incremental changes to working examples—so the approach in this book should be a familiar one.

## Who Should Read This Book

The book is intended for those readers who want to learn about Hadoop version 2, but not get mired in technical details. New users, system administrators, and devops personnel should all be able to get up to speed with many of the important Hadoop topics and tools by working through this text. In particular, readers with no Hadoop experience should find the book highly usable, even if they have no Java programming experience. Experience with Linux command-line tools is helpful, as all of the examples involve command-line interaction with Hadoop.

Users and administrators who are currently using Hadoop version 1 should find value as well. The changes in Hadoop version 2 are rather substantial, and this book's discussion of YARN and some of the changes in the MapReduce framework is important.

## Book Structure

The basic structure of this book was adapted from my video tutorial, *Hadoop Fundamentals LiveLessons, Second Edition* and *Apache Hadoop YARN Fundamentals LiveLessons* from Addison-Wesley. Almost all of the examples are identical to those found in the videos. Some readers may find it beneficial to watch the videos in conjunction with reading the book as I carefully step through all the examples.

A few small pieces have been borrowed from *Apache Hadoop™ YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop™ 2*, a book that I coauthored. If you want to explore YARN application development in more detail, you may want to consider reading this book and viewing its companion video.

Much of this book uses the Hortonworks Data Platform (HDP) for Hadoop. The HDP is a fully open source Hadoop distribution made available by Hortonworks. While it is possible to download and install the core Hadoop system and tools (as is discussed in Chapter 2), using an integrated distribution reduces many of the issues that may arise from the “roll your own” approach. In addition, the Apache Ambari graphical installation and management tool is too good to pass up and supports the Hortonworks HDP packages. HDP version 2.2 and Ambari 1.7 were used for this book. As I write this preface, Hortonworks has just announced the launch of HDP version 2.3 with Apache Ambari 2.0. (So much for staying ahead of the curve in the Hadoop world!) Fortunately, the fundamentals remain the same and the examples are all still relevant.

The chapters in this text have been arranged to provide a flexible introduction for new readers. As delineated in Appendix B, “Getting Started Flowchart and Troubleshooting Guide,” there are two paths you can follow: read Chapters 1, 3, and 5 and then start playing with the examples, or jump right in and run the examples in Chapter 4. If you don’t have a Hadoop environment, Chapter 2 provides a way to install Hadoop on a variety of systems, including a laptop or small desk-side computer, a cluster, or even in the cloud. Presumably after running examples, you will go back and read the background chapters.

Chapter 1 provides essential background on Hadoop technology and history. The Hadoop data lake is introduced, along with an overview of the MapReduce process found in version 1 of Hadoop. The big changes in Hadoop version 2 are described, and the YARN resource manager is introduced as a way forward for almost any computing model. Finally, a brief overview of the many software projects that make up the Hadoop ecosystem is presented. This chapter provides an underpinning for the rest of the book.

If you need access to a Hadoop system, a series of installation recipes is provided in Chapter 2. There is also an explanation of the core Hadoop services and the way in which they are configured. Some general advice for choosing hardware and software environments is provided, but the main focus is on providing a platform to learn about Hadoop. Fortunately, there are two ways to do this without purchasing or renting any hardware. The Hortonworks Hadoop sandbox provides a Linux virtual machine that can be run on almost any platform. The sandbox is a full Hadoop install and provides an environment through which to explore Hadoop. As an alternative to the sandbox, the installation of Hadoop on a single Linux machine provides a learning platform and offers some insights into the Hadoop core components. Chapter 2 also addresses cluster installation using Apache Ambari for a local cluster or Apache Whirr for a cloud deployment.

All Hadoop applications use the Hadoop Distributed File System (HDFS). Chapter 3 covers some essential HDFS features and offers quick tips on how to navigate and use the file system. The chapter concludes with some HDFS programming examples. It provides important background and should be consulted before trying the examples in later chapters.

Chapter 4 provides a show-and-tell walk-through of some Hadoop examples and benchmarks. The Hadoop Resource Manager web GUI is also introduced as a way to observe application progress. The chapter concludes with some tips on controlling Hadoop MapReduce jobs. Use this chapter to get a feel for how Hadoop applications run and operate.

The MapReduce programming model, while simple in nature, can be a bit confusing when run across a cluster. Chapter 5 provides a basic introduction to the MapReduce programming model using simple examples. The chapter concludes with a simplified walk-through of the parallel Hadoop MapReduce process. This chapter will help you understand the basic Hadoop MapReduce terminology.



If you are interested in low-level Hadoop programming, Chapter 6 provides an introduction to Hadoop MapReduce programming. Several basic approaches are covered, including Java, the streaming interface with Python, and the C++ Pipes interface. A short example also explains how to view application logs. This chapter is not essential for using Hadoop. In fact, many Hadoop users begin with the high-level tools discussed in Chapter 7.

While many applications have been written to run on the native Hadoop Java interface, a wide variety of tools are available that provide a high-level approach to programing and data movement. Chapter 7 introduces (with examples) essential Hadoop tools including Apache Pig (scripting language), Apache Hive (SQL-like language), Apache Sqoop (RDMS import/export), and Apache Flume (serial data import). An example demonstrating how to use the Oozie workflow manager is also provided. The chapter concludes with an Apache HBase (big table database) example.

If you are interested in learning more about Hadoop YARN applications, Chapter 8 introduces non-MapReduce applications under Hadoop. As a simple example, the YARN Distributed-Shell is presented, along with a discussion of how YARN applications work under Hadoop version 2. A description of the latest non-MapReduce YARN applications is provided as well.

If you installed Hadoop with Apache Ambari in Chapter 2, Chapter 9 provides a tour of its capabilities and offers some examples that demonstrate how to use Ambari on a real Hadoop cluster. A tour of Ambari features and procedures to restart Hadoop services and change system-wide Hadoop properties is presented as well. The basic steps outlined in this chapter are used in Chapter 10 to make administrative changes to the cluster.

Chapter 10 provides some basic Hadoop administration procedures. Although administrators will find information on basic procedures and advice in this chapter, other users will also benefit by discovering how HDFS, YARN, and the Capacity scheduler can be configured for their workloads.

Consult the appendixes for information on the book webpage, a getting started flowchart, and a general Hadoop troubleshooting guide. The appendixes also include a resources summary page and procedures for installing Apache Hue (a high-level Hadoop GUI) and Apache Spark (a popular non-MapReduce programming model).

Finally, the Hadoop ecosystem continues to grow rapidly. Many of the existing Hadoop applications and tools were intentionally not covered in this text because their inclusion would have turned this book into a longer and slower introduction to Hadoop 2. And, there are many more tools and applications on the way! Given the dynamic nature of the Hadoop ecosystem, this introduction to Apache Hadoop 2 is meant to provide both a compass and some important waypoints to aid in your navigation of the Hadoop 2 data lake.

## **Book Conventions**

Code and file references are displayed in a monospaced font. Code input lines that wrap because they are too long to fit on one line in this book are denoted with this symbol: ➤. Long output lines are wrapped at page boundaries without the symbol.

## **Accompanying Code**

Please see Appendix A, “Book Webpage and Code Download,” for the location of all code used in this book.



# Acknowledgments

Some of the figures and examples were inspired and derived from the Yahoo! Hadoop Tutorial (<https://developer.yahoo.com/hadoop/tutorial/>), the Apache Software Foundation (ASF; <http://www.apache.org>), Hortonworks (<http://hortonworks.com>), and Michael Noll (<http://www.michael-noll.com>). Any copied items either had permission for use granted by the author or were available under an open sharing license.

Many people have worked behind the scenes to make this book possible. Thank you to the reviewers who took the time to carefully read the rough drafts: Jim Lux, Prentice Bisbal, Jeremy Fischer, Fabricio Cannini, Joshua Mora, Matthew Helmke, Charlie Peck, and Robert P. J. Day. Your feedback was very valuable and helped make for a sturdier book.

To Debra Williams Cauley of Addison-Wesley, your kind efforts and office at the GCT Oyster Bar made the book-writing process almost easy. I also cannot forget to thank my support crew: Emily, Marlee, Carla, and Taylor—yes, another book you know nothing about. And, finally, the biggest thank you to my patient and wonderful wife, Maddy, for her constant support.



# About the Author

**Douglas Eadline, Ph.D.**, began his career as a practitioner and a chronicler of the Linux cluster HPC revolution and now documents Big Data analytics. Starting with the first Beowulf how-to document, Doug has written hundreds of articles, white papers, and instructional documents covering virtually all aspects of HPC computing. Prior to starting and editing the popular ClusterMonkey.net website in 2005, he served as editor-in-chief for *ClusterWorld Magazine*, and was senior HPC editor for *Linux Magazine*. He has practical, hands-on experience in many aspects of HPC, including hardware and software design, benchmarking, storage, GPU, cloud computing, and parallel computing. Currently, he is a writer and consultant to the HPC industry and leader of the Limulus Personal Cluster Project (<http://limulus.basement-supercomputing.com>). He is author of *Hadoop Fundamentals LiveLessons* and *Apache Hadoop YARN Fundamentals LiveLessons* videos from Addison-Wesley and book coauthor of *Apache Hadoop™ YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop™ 2*.



# Running Example Programs and Benchmarks

## In This Chapter:

- The steps needed to run the Hadoop MapReduce examples are provided.
- An overview of the YARN ResourceManager web GUI is presented.
- The steps needed to run two important benchmarks are provided.
- The `mapred` command is introduced as a way to list and kill MapReduce jobs.

When using new or updated hardware or software, simple examples and benchmarks help confirm proper operation. Apache Hadoop includes many examples and benchmarks to aid in this task. This chapter provides instructions on how to run, monitor, and manage some basic MapReduce examples and benchmarks.

## Running MapReduce Examples

All Hadoop releases come with MapReduce example applications. Running the existing MapReduce examples is a simple process—once the example files are located, that is. For example, if you installed Hadoop version 2.6.0 from the Apache sources under `/opt`, the examples will be in the following directory:

```
/opt/hadoop-2.6.0/share/hadoop/mapreduce/
```

In other versions, the examples may be in `/usr/lib/hadoop-mapreduce/` or some other location. The exact location of the example jar file can be found using the `find` command:

```
$ find / -name "hadoop-mapreduce-examples*.jar" -print
```



For this chapter the following software environment will be used:

- OS: Linux
- Platform: RHEL 6.6
- Hortonworks HDP 2.2 with Hadoop Version: 2.6

In this environment, the location of the examples is `/usr/hdp/2.2.4.2-2/hadoop-mapreduce`. For the purposes of this example, an environment variable called `HADOOP_EXAMPLES` can be defined as follows:

```
$ export HADOOP_EXAMPLES=/usr/hdp/2.2.4.2-2/hadoop-mapreduce
```

Once you define the examples path, you can run the Hadoop examples using the commands discussed in the following sections.

### Listing Available Examples

A list of the available examples can be found by running the following command. In some cases, the version number may be part of the jar file (e.g., in the version 2.6 Apache sources, the file is named `hadoop-mapreduce-examples-2.6.0.jar`).

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar
```

#### Note

In previous versions of Hadoop, the command `hadoop jar . . .` was used to run MapReduce programs. Newer versions provide the `yarn` command, which offers more capabilities. Both commands will work for these examples.

The possible examples are as follows:

An example program must be given as the first argument.

Valid program names are:

- `aggregatewordcount`: An Aggregate based map/reduce program that counts the words in the input files.
- `aggregatewordhist`: An Aggregate based map/reduce program that computes the histogram of the words in the input files.
- `bbp`: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
- `dbcoun`: An example job that count the pageview counts from a database.
- `distbbp`: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.
- `grep`: A map/reduce program that counts the matches of a regex in the input.
- `join`: A job that effects a join over sorted, equally partitioned datasets
- `multifilewc`: A job that counts words from several files.
- `pentomino`: A map/reduce tile laying program to find solutions to pentomino problems.

pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.

randomtextwriter: A map/reduce program that writes 10GB of random textual data per node.

randomwriter: A map/reduce program that writes 10GB of random data per node.

secondarysort: An example defining a secondary sort to the reduce.

sort: A map/reduce program that sorts the data written by the random writer.

sudoku: A sudoku solver.

teragen: Generate data for the terasort

terasort: Run the terasort

teravalidate: Checking results of terasort

wordcount: A map/reduce program that counts the words in the input files.

wordmean: A map/reduce program that counts the average length of the words in the input files.

wordmedian: A map/reduce program that counts the median length of the words in the input files.

wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.

To illustrate several features of Hadoop and the YARN ResourceManager service GUI, the pi and terasort examples are presented next. To find help for running the other examples, enter the example name without any arguments. Chapter 6, “MapReduce Programming,” covers one of the other popular examples called wordcount.

## Running the Pi Example

The pi example calculates the digits of  $\pi$  using a quasi-Monte Carlo method. If you have not added users to HDFS (see Chapter 10, “Basic Hadoop Administration Procedures”), run these tests as user hdfs. To run the pi example with 16 maps and 1,000,000 samples per map, enter the following command:

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar pi 16 1000000
```

If the program runs correctly, you should see output similar to the following. (Some of the Hadoop INFO messages have been removed for clarity.)

```
Number of Maps = 16
Samples per Map = 1000000
Wrote input for Map #0
Wrote input for Map #1
Wrote input for Map #2
Wrote input for Map #3
Wrote input for Map #4
Wrote input for Map #5
Wrote input for Map #6
```

```

Wrote input for Map #7
Wrote input for Map #8
Wrote input for Map #9
Wrote input for Map #10
Wrote input for Map #11
Wrote input for Map #12
Wrote input for Map #13
Wrote input for Map #14
Wrote input for Map #15
Starting Job
...
15/05/13 20:10:30 INFO mapreduce.Job: map 0% reduce 0%
15/05/13 20:10:37 INFO mapreduce.Job: map 19% reduce 0%
15/05/13 20:10:39 INFO mapreduce.Job: map 50% reduce 0%
15/05/13 20:10:46 INFO mapreduce.Job: map 56% reduce 0%
15/05/13 20:10:47 INFO mapreduce.Job: map 94% reduce 0%
15/05/13 20:10:48 INFO mapreduce.Job: map 100% reduce 100%
15/05/13 20:10:48 INFO mapreduce.Job: Job job_1429912013449_0047 completed
successfully
15/05/13 20:10:48 INFO mapreduce.Job: Counters: 49
  File System Counters
    FILE: Number of bytes read=358
    FILE: Number of bytes written=1949395
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=4198
    HDFS: Number of bytes written=215
    HDFS: Number of read operations=67
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=3
  Job Counters
    Launched map tasks=16
    Launched reduce tasks=1
    Data-local map tasks=16
    Total time spent by all maps in occupied slots (ms)=158378
    Total time spent by all reduces in occupied slots (ms)=8462
    Total time spent by all map tasks (ms)=158378
    Total time spent by all reduce tasks (ms)=8462
    Total vcore-seconds taken by all map tasks=158378
    Total vcore-seconds taken by all reduce tasks=8462
    Total megabyte-seconds taken by all map tasks=243268608
    Total megabyte-seconds taken by all reduce tasks=12997632
  Map-Reduce Framework
    Map input records=16
    Map output records=32
    Map output bytes=288
    Map output materialized bytes=448

```

```
Input split bytes=2310
Combine input records=0
Combine output records=0
Reduce input groups=2
Reduce shuffle bytes=448
Reduce input records=32
Reduce output records=0
Spilled Records=64
Shuffled Maps=16
Failed Shuffles=0
Merged Map outputs=16
GC time elapsed (ms)=1842
CPU time spent (ms)=11420
Physical memory (bytes) snapshot=13405769728
Virtual memory (bytes) snapshot=33911930880
Total committed heap usage (bytes)=17026777088

Shuffle Errors
  BAD_ID=0
  CONNECTION=0
  IO_ERROR=0
  WRONG_LENGTH=0
  WRONG_MAP=0
  WRONG_REDUCE=0

File Input Format Counters
  Bytes Read=1888
File Output Format Counters
  Bytes Written=97

Job Finished in 23.718 seconds
Estimated value of Pi is 3.14159125000000000000
```

Notice that the MapReduce progress is shown in the same way as Hadoop version 1, but the application statistics are different. Most of the statistics are self-explanatory. The one important item to note is that the YARN MapReduce framework is used to run the program. (See Chapter 1, “Background and Concepts,” and Chapter 8, “Hadoop YARN Applications,” for more information about YARN frameworks.)

## Using the Web GUI to Monitor Examples

This section provides an illustration of using the YARN ResourceManager web GUI to monitor and find information about YARN jobs. The Hadoop version 2 YARN ResourceManager web GUI differs significantly from the MapReduce web GUI found in Hadoop version 1. Figure 4.1 shows the main YARN web interface. The cluster metrics are displayed in the top row, while the running applications are displayed in the main table. A menu on the left provides navigation to the nodes table, various job categories (e.g., New, Accepted, Running, Finished, Failed), and the Capacity Scheduler (covered in Chapter 10, “Basic Hadoop Administration Procedures”). This interface can be opened directly from the Ambari YARN service Quick Links menu or by

The screenshot displays the Hadoop RUNNING Applications web GUI. The browser window shows the URL `limulus:8088/cluster/apps/RUNNING`. The page title is "RUNNING Applications". On the left, there is a sidebar with a "Cluster" dropdown menu and a list of application states: NEW, NEW SAVING, SUBMITTED, ACCEPTED, RUNNING, FINISHED, FAILED, and KILLED. Below this is a "Scheduler" link and a "Tools" section. The main content area is titled "Cluster Metrics" and contains a table with the following columns: Apps Submitted (43), Apps Pending (0), Apps Running (1), Apps Completed (42), Containers Running (1), Memory Used (1.50 GB), Memory Total (48 GB), Memory Reserved (0 B), VCores Used (1), VCores Total (16), VCores Reserved (0), Active Nodes (4), Decommissioned Nodes (0), Lost Nodes (0), Unhealthy Nodes (0), and Rebooted Nodes (0). Below this is a table of running applications with columns: ID, User, Name, Application Type, Queue, StartTime, FinishTime, State, FinalStatus, Progress, and Tracking UI. One application is listed: `application_1429912013449_0044` by user `hdfs`, running `MAPREDUCE` in the `default` queue, starting on `Wed May 13 15:28:47 -0400 2015`. The state is `RUNNING` and the final status is `UNDEFINED`. The tracking UI is `ApplicationMaster`. At the bottom, it says "Showing 1 to 1 of 1 entries" and provides navigation links: "First Previous 1 Next Last".

Figure 4.1 Hadoop RUNNING Applications web GUI for the pi example

directly entering `http://hostname:8088` into a local web browser. For this example, the `pi` application is used. Note that the application can run quickly and may finish before you have fully explored the GUI. A longer-running application, such as `terasort`, may be helpful when exploring all the various links in the GUI.

For those readers who have used or read about Hadoop version 1, if you look at the Cluster Metrics table, you will see some new information. First, you will notice that the “Map/Reduce Task Capacity” has been replaced by the number of running containers. If YARN is running a MapReduce job, these containers can be used for both map and reduce tasks. Unlike in Hadoop version 1, the number of mappers and reducers is not fixed. There are also memory metrics and links to node status. If you click on the Nodes link (left menu under About), you can get a summary of the node activity and state. For example, Figure 4.2 is a snapshot of the node activity while the `pi` application is running. Notice the number of containers, which are used by the MapReduce framework as either mappers or reducers.

Going back to the main Applications/Running window (Figure 4.1), if you click on the `application_14299...` link, the Application status window in Figure 4.3 will appear. This window provides an application overview and metrics, including the cluster node on which the ApplicationMaster container is running.

Clicking the ApplicationMaster link next to “Tracking URL:” in Figure 4.3 leads to the window shown in Figure 4.4. Note that the link to the application’s ApplicationMaster is also found in the last column on the main Running Applications screen shown in Figure 4.1.

In the MapReduce Application window, you can see the details of the MapReduce application and the overall progress of mappers and reducers. Instead of containers, the MapReduce application now refers to maps and reducers. Clicking `job_14299...` brings up the window shown in Figure 4.5. This window displays more detail about the

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
44	0	1	43	18	27 GB	48 GB	0 B	18	16	0	1	0	0	0	0
Showing 1 to 4 of 4 entries															

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
/default-rack		RUNNING	n0:45454	n0:8042	Wed May 13 15:53:27 -0400 2015		4	6 GB	6 GB	4	0	2.6.0.2.2.4.2-2
/default-rack		RUNNING	n1:45454	n1:8042	Wed May 13 15:53:45 -0400 2015		3	4.50 GB	7.50 GB	3	1	2.6.0.2.2.4.2-2
/default-rack		RUNNING	limulus:45454	limulus:8042	Wed May 13 15:53:26 -0400 2015		8	12 GB	0 B	8	-4	2.6.0.2.2.4.2-2
/default-rack		RUNNING	n2:45454	n2:8042	Wed May 13 15:53:25 -0400 2015		3	4.50 GB	7.50 GB	3	1	2.6.0.2.2.4.2-2

Figure 4.2 Hadoop YARN ResourceManager nodes status window

Application Overview			
User:	hdfs	Name:	QuasiMonteCarlo
Application Type:	MAPREDUCE	Application Tags:	
State:	RUNNING	FinalStatus:	UNDEFINED
Started:	Wed May 13 15:28:47 -0400 2015	Elapsed:	215sec
Tracking URL:	ApplicationMaster	Diagnostics:	

Application Metrics			
Total Resource Preempted:	<memory:0, vCores:0>	Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0	Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0	Aggregate Resource Allocation:	464290 MB-seconds, 293 vcore-seconds

ApplicationMaster			
Attempt Number	Start Time	Node	Logs
1	Wed May 13 15:28:47 -0400 2015	n0:8042	logs

Figure 4.3 Hadoop YARN application status for the pi example

number of pending, running, completed, and failed mappers and reducers, including the elapsed time since the job started.

The status of the job in Figure 4.5 will be updated as the job progresses (the window needs to be refreshed manually). The ApplicationMaster collects and reports the progress of each mapper and reducer task. When the job is finished, the window is updated to that shown in Figure 4.6. It reports the overall run time and provides a breakdown of the timing of the key phases of the MapReduce job (map, shuffle, merge, reduce).

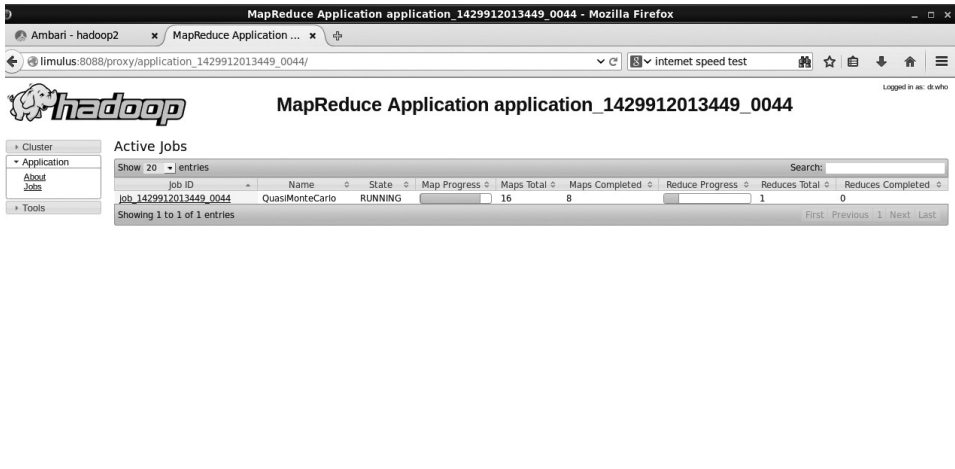


Figure 4.4 Hadoop YARN ApplicationMaster for MapReduce application



Figure 4.5 Hadoop YARN MapReduce job progress

If you click the node used to run the ApplicationMaster (n0:8042 in Figure 4.6), the window in Figure 4.7 opens and provides a summary from the NodeManager on node n0. Again, the NodeManager tracks only containers; the actual tasks running in the containers are determined by the ApplicationMaster.

Going back to the job summary page (Figure 4.6), you can also examine the logs for the ApplicationMaster by clicking the “logs” link. To find information about the mappers and reducers, click the numbers under the Failed, Killed, and Successful columns. In this example, there were 16 successful mappers and

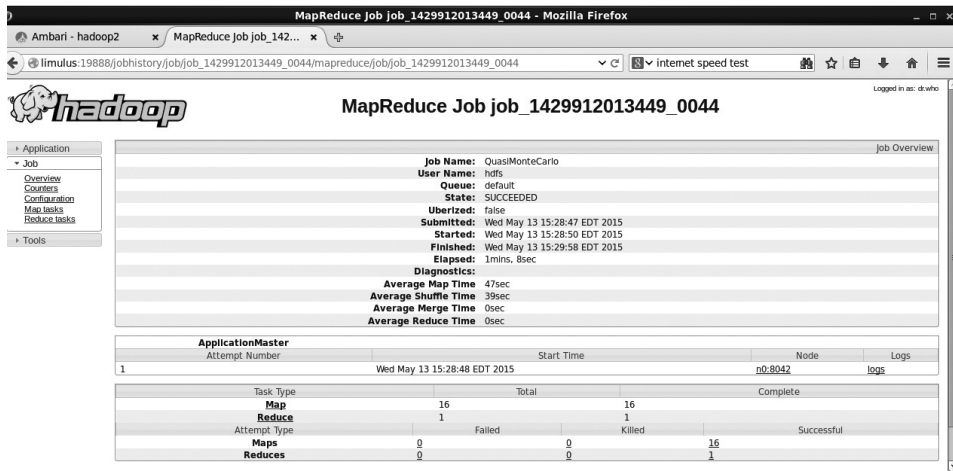


Figure 4.6 Hadoop YARN completed MapReduce job summary

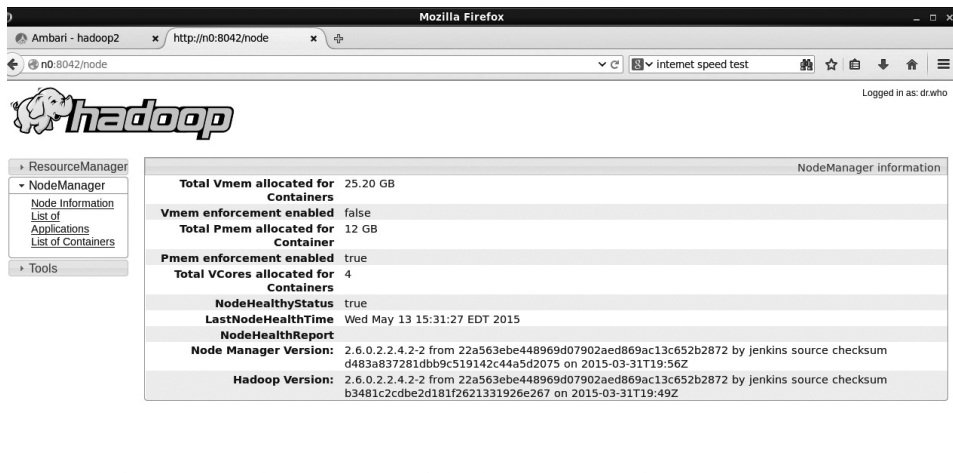


Figure 4.7 Hadoop YARN NodeManager for n0 job summary

one successful reducer. All the numbers in these columns lead to more information about individual map or reduce process. For instance, clicking the “16” under “Successful” in Figure 4.6 displays the table of map tasks in Figure 4.8. The metrics for the Application Master container are displayed in table form. There is also a link to the log file for each process (in this case, a map process). Viewing the logs requires that the `yarn.log.aggregation-enable` variable in the `yarn-site.xml` file be set. For more on changing Hadoop settings, see Chapter 9, “Managing Hadoop with Apache Ambari.”



Attempt	State	Status	Node	Start Time	Finish Time	Elapsed Time
attempt_1429912013449_0044_m_000000_0	SUCCEEDED	Generated 100000000 samples.	rack1limulus:8042	Wed May 13 15:28:52 -0400 2015	Wed May 13 15:29:58 -0400 2015	1mins, 5sec
attempt_1429912013449_0044_m_000001_0	SUCCEEDED	Generated 100000000 samples.	rack1limulus:8042	Wed May 13 15:28:52 -0400 2015	Wed May 13 15:29:57 -0400 2015	1mins, 4sec
attempt_1429912013449_0044_m_000002_0	SUCCEEDED	Generated 100000000 samples.	rack1limulus:8042	Wed May 13 15:28:52 -0400 2015	Wed May 13 15:29:58 -0400 2015	1mins, 5sec
attempt_1429912013449_0044_m_000003_0	SUCCEEDED	Generated 100000000 samples.	rack1limulus:8042	Wed May 13 15:28:52 -0400 2015	Wed May 13 15:29:57 -0400 2015	1mins, 5sec
attempt_1429912013449_0044_m_000004_0	SUCCEEDED	Generated 100000000 samples.	rack1limulus:8042	Wed May 13 15:28:52 -0400 2015	Wed May 13 15:29:58 -0400 2015	1mins, 5sec
attempt_1429912013449_0044_m_000005_0	SUCCEEDED	Generated 100000000 samples.	rack1limulus:8042	Wed May 13 15:28:52 -0400 2015	Wed May 13 15:29:58 -0400 2015	1mins, 5sec
attempt_1429912013449_0044_m_000006_0	SUCCEEDED	Generated 100000000 samples.	rack1limulus:8042	Wed May 13 15:28:52 -0400 2015	Wed May 13 15:29:55 -0400 2015	1mins, 3sec
attempt_1429912013449_0044_m_000007_0	SUCCEEDED	Generated 100000000 samples.	rack1limulus:8042	Wed May 13 15:28:52 -0400 2015	Wed May 13 15:29:58 -0400 2015	1mins, 5sec
attempt_1429912013449_0044_m_000008_0	SUCCEEDED	Generated 100000000 samples.	rack1limulus:8042	Wed May 13 15:28:52 -0400 2015	Wed May 13 15:29:24 -0400 2015	31sec

Figure 4.8 Hadoop YARN MapReduce logs available for browsing

ApplicationMaster	Attempt Number	Start Time	Node	Logs
1	1	Wed May 13 15:28:47 -0400 2015	n0-8042	logs

Figure 4.9 Hadoop YARN application summary page

If you return to the main cluster window (Figure 4.1), choose Applications/ Finished, and then select our application, you will see the summary page shown in Figure 4.9.

There are a few things to notice in the previous windows. First, because YARN manages applications, all information reported by the ResourceManager concerns the resources provided and the application type (in this case, MAPREDUCE). In Figure 4.1 and Figure 4.4, the YARN ResourceManager refers to the pi example by its

application-id (application\_1429912013449\_0044). YARN has no data about the actual application other than the fact that it is a MapReduce job. Data from the actual MapReduce job are provided by the MapReduce framework and referenced by a job-id (job\_1429912013449\_0044) in Figure 4.6. Thus, two clearly different data streams are combined in the web GUI: YARN *applications* and MapReduce framework *jobs*. If the framework does not provide job information, then certain parts of the web GUI will not have anything to display.

Another interesting aspect of the previous windows is the dynamic nature of the mapper and reducer tasks. These tasks are executed as YARN containers, and their number will change as the application runs. Users may request specific numbers of mappers and reducers, but the ApplicationMaster uses them in a dynamic fashion. As mappers complete, the ApplicationMaster will return the containers to the Resource-Manager and request a smaller number of reducer containers. This feature provides for much better cluster utilization because mappers and reducers are dynamic—rather than fixed—resources.

## Running Basic Hadoop Benchmarks

Many Hadoop benchmarks can provide insight into cluster performance. The best benchmarks are always those that reflect real application performance. The two benchmarks discussed in this section, `terasort` and `TestDFSIO`, provide a good sense of how well your Hadoop installation is operating and can be compared with public data published for other Hadoop systems. The results, however, should not be taken as a single indicator for system-wide performance on all applications.

The following benchmarks are designed for full Hadoop cluster installations. These tests assume a multi-disk HDFS environment. Running these benchmarks in the Hortonworks Sandbox or in the pseudo-distributed single-node install from Chapter 2 is not recommended because all input and output (I/O) are done using a single system disk drive.

### Running the Terasort Test

The `terasort` benchmark sorts a specified amount of randomly generated data. This benchmark provides combined testing of the HDFS and MapReduce layers of a Hadoop cluster. A full `terasort` benchmark run consists of the following three steps:

1. Generating the input data via `teragen` program.
2. Running the actual `terasort` benchmark on the input data.
3. Validating the sorted output data via the `teravalidate` program.

In general, each row is 100 bytes long; thus the total amount of data written is 100 times the number of rows specified as part of the benchmark (i.e., to write 100GB of data, use 1 billion rows). The input and output directories need to be specified in HDFS. The following sequence of commands will run the benchmark for 50GB of

data as user `hdfs`. Make sure the `/user/hdfs` directory exists in HDFS before running the benchmarks.

1. Run `teragen` to generate rows of random data to sort.

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar teragen 50000000
➤ /user/hdfs/TeraGen-50GB
```

2. Run `terasort` to sort the database.

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar terasort
➤ /user/hdfs/TeraGen-50GB /user/hdfs/TeraSort-50GB
```

3. Run `teravalidate` to validate the sort.

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar teravalidate
➤ /user/hdfs/TeraSort-50GB /user/hdfs/TeraValid-50GB
```

To report results, the time for the actual sort (`terasort`) is measured and the benchmark rate in megabytes/second (MB/s) is calculated. For best performance, the actual `terasort` benchmark should be run with a replication factor of 1. In addition, the default number of `terasort` reducer tasks is set to 1. Increasing the number of reducers often helps with benchmark performance. For example, the following command will instruct `terasort` to use four reducer tasks:

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-examples.jar terasort
➤ -Dmapred.reduce.tasks=4 /user/hdfs/TeraGen-50GB /user/hdfs/TeraSort-50GB
```

Also, do not forget to clean up the `terasort` data between runs (and after testing is finished). The following command will perform the cleanup for the previous example:

```
$ hdfs dfs -rm -r -skipTrash Tera*
```

## Running the TestDFSIO Benchmark

Hadoop also includes an HDFS benchmark application called `TestDFSIO`. The `TestDFSIO` benchmark is a read and write test for HDFS. That is, it will write or read a number of files to and from HDFS and is designed in such a way that it will use one map task per file. The file size and number of files are specified as command-line arguments. Similar to the `terasort` benchmark, you should run this test as user `hdfs`.

Similar to `terasort`, `TestDFSIO` has several steps. In the following example, 16 files of size 1GB are specified. Note that the `TestDFSIO` benchmark is part of the `hadoop-mapreduce-client-jobclient.jar`. Other benchmarks are also available as part of this jar file. Running it with no arguments will yield a list. In addition to `TestDFSIO`, `NNBench` (load testing the NameNode) and `MRBench` (load testing the MapReduce framework) are commonly used Hadoop benchmarks. Nevertheless, `TestDFSIO` is perhaps the most widely reported of these benchmarks. The steps to run `TestDFSIO` are as follows:

1. Run `TestDFSIO` in write mode and create data.

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-client-jobclient-tests.jar
➤ TestDFSIO -write -nrFiles 16 -fileSize 1000
```

Example results are as follows (date and time prefix removed).

```
fs.TestDFSIO: ----- TestDFSIO ----- : write
fs.TestDFSIO:           Date & time: Thu May 14 10:39:33 EDT 2015
fs.TestDFSIO:           Number of files: 16
fs.TestDFSIO: Total MBytes processed: 16000.0
fs.TestDFSIO:           Throughput mb/sec: 14.890106361891005
fs.TestDFSIO: Average IO rate mb/sec: 15.690713882446289
fs.TestDFSIO: IO rate std deviation: 4.0227035201665595
fs.TestDFSIO:           Test exec time sec: 105.631
```

## 2. Run TestDFSIO in read mode.

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-client-jobclient-tests.jar
➔ TestDFSIO -read -nrFiles 16 -fileSize 1000
```

Example results are as follows (date and time prefix removed). The large standard deviation is due to the placement of tasks in the cluster on a small four-node cluster.

```
fs.TestDFSIO: ----- TestDFSIO ----- : read
fs.TestDFSIO:           Date & time: Thu May 14 10:44:09 EDT 2015
fs.TestDFSIO:           Number of files: 16
fs.TestDFSIO: Total MBytes processed: 16000.0
fs.TestDFSIO:           Throughput mb/sec: 32.38643494172466
fs.TestDFSIO: Average IO rate mb/sec: 58.72880554199219
fs.TestDFSIO: IO rate std deviation: 64.60017624360337
fs.TestDFSIO:           Test exec time sec: 62.798
```

## 3. Clean up the TestDFSIO data.

```
$ yarn jar $HADOOP_EXAMPLES/hadoop-mapreduce-client-jobclient-tests.jar
➔ TestDFSIO -clean
```

Running the TestDFSIO and terasort benchmarks help you gain confidence in a Hadoop installation and detect any potential problems. It is also instructive to view the Ambari dashboard and the YARN web GUI (as described previously) as the tests run.

## Managing Hadoop MapReduce Jobs

Hadoop MapReduce jobs can be managed using the `mapred job` command. The most important options for this command in terms of the examples and benchmarks are `-list`, `-kill`, and `-status`. In particular, if you need to kill one of the examples or benchmarks, you can use the `mapred job -list` command to find the `job-id` and then use `mapred job -kill <job-id>` to kill the job across the cluster. MapReduce jobs can also be controlled at the application level with the `yarn application` command (see Chapter 10, “Basic Hadoop Administration Procedures”). The possible options for `mapred job` are as follows:

```
$ mapred job
Usage: CLI <command> <args>
       [-submit <job-file>]
```

```

[-status <job-id>]
[-counter <job-id> <group-name> <counter-name>]
[-kill <job-id>]
[-set-priority <job-id> <priority>]. Valid values for priorities
are: VERY_HIGH HIGH NORMAL LOW VERY_LOW
[-events <job-id> <from-event-#> <#-of-events>]
[-history <jobHistoryFile>]
[-list [all]]
[-list-active-trackers]
[-list-blacklisted-trackers]
[-list-attempt-ids <job-id> <task-type> <task-state>]. Valid values
for <task-type> are REDUCE MAP. Valid values for <task-state> are
running, completed
[-kill-task <task-attempt-id>]
[-fail-task <task-attempt-id>]
[-logs <job-id> <task-attempt-id>]

```

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|resourcemanager:port> specify a ResourceManager
-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]
```

## Summary and Additional Resources

No matter what the size of the Hadoop cluster, confirming and measuring the MapReduce performance of that cluster is an important first step. Hadoop includes some simple applications and benchmarks that can be used for this purpose. The YARN ResourceManager web GUI is a good way to monitor the progress of any application. Jobs that run under the MapReduce framework report a large number of run-time metrics directly (including logs) back to the GUI; these metrics are then presented to the user in a clear and coherent fashion. Should issues arise when running the examples and benchmarks, the `mapred job` command can be used to kill a MapReduce job.

Additional information and background on each of the examples and benchmarks can be found from the following resources:

- **Pi Benchmark**

- <https://hadoop.apache.org/docs/current/api/org/apache/hadoop/examples/pi/package-summary.html>

- **Terasort Benchmark**

- <https://hadoop.apache.org/docs/current/api/org/apache/hadoop/examples/terasort/package-summary.html>

- **Benchmarking and Stress Testing an Hadoop Cluster**

- <http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasort-testdfsio-nnbench-mrbench> (uses Hadoop V1, will work with V2)



# Index

-- (dashes), Pig comment delimiters, 133  
/\* \*/ (slash asterisk...), Pig comment delimiters, 133  
: (semicolon), Pig command terminator, 133  
' ' (single quotes), HBase argument delimiters, 164

## A

Action flow nodes, 155–156  
Ad-hoc queries. *See* Hive.  
Admin pull-down menu, Ambari, 194  
Admin view, Ambari, 193  
Ambari  
    Admin pull-down menu, 194  
    Admin view, 193  
    changing the password, 186  
    console, 45–55  
    customizing, 193  
    dashboard, 41  
    Dashboard view, 186–189  
    host management, 191–192  
    Hosts view, 191–192  
    installing Hadoop. *See* Installing Hadoop, with Ambari.  
    listing installed software, 193  
    management screen, opening, 194  
    overview, 185–186  
    progress window, disabling, 194  
    properties, setting, 189–191  
    screen shots, 46–55  
    signing in, 45–46  
    signing out, 194  
    status widgets, 186–189  
    as troubleshooting tool, 234  
    Views view, 193  
Ambari, managing Hadoop services  
    configuring services, 189–191  
    listing service accounts, 193  
    reporting service interruptions, 194–195  
    resolving service interruptions, 195–198  
    restarting after configuration change, 200–201  
    reverting to a previous version, 202–204  
    Services view, 189–191  
    starting and stopping services, 190–191  
Ambari Installation Guide, 42  
ambari-agent, 47–49  
ambari-server, 44–45  
*Apache Hadoop 2 Quick-Start Guide*, 226, 230–233  
*Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2*, 17, 178, 221  
Apache projects. *See specific projects.*  
ApplicationHistoryServer, 20  
ApplicationMaster  
    definition, 178–179  
    enabling restarts, 222  
Applications. *See also* YARN applications.  
    C language, examples, 82–83  
    dynamic application management, online resources, 183–184  
    Hadoop V1, compatibility with MapReduce, 224–225  
    Java language, examples, 78–81  
    managing dynamically, 183–184  
    monitoring, 89–95  
    parallel, debugging, 124  
ASF (Apache Software Foundation), 2



- Avro
  - data transfer format, 150
  - in the Hadoop ecosystem, 17
- B**
- BackupNode, 71
- Backups, HDFS, 71
- Balancing
  - DataNodes, online resources, 214
  - HDFS, 213–214
- Bash scripts, HBase, 167
- Benchmarks
  - terasort test, 95–96
  - TestDFSIO, 96–97
- Big Data
  - characteristics of, 4
  - definition, 4
  - examples, 4–5
  - storing. *See* Data lakes.
  - typical size, 4
- “Big Data Surprises,” 4
- Block locations, printing, 213
- Block replication, 67–68
- Block reports, 213
- blocks option, 213
- Books and publications. *See also* Online resources.
  - Apache Hadoop 2 Quick-Start Guide*, 226, 230–233
  - Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2*, 17, 178, 221, 243
  - “Nobody Ever Got Fired for Using Hadoop,” 4
- Bottlenecks, resolving, 8
- Bulk data, adding, 168
- Bundle jobs, 155
- C**
- C++
  - on MapReduce, 119–121
  - WordCount example program, 119–121
- C application, HDFS programming
  - examples, 82–83
- Cafarella, Michael J., 3
- Capacity scheduler
  - administration, online resources, 247
  - description, 220–221
  - online resources, 221, 226
- cat command, 145
- Cells, deleting, 167
- Channel component of Flume, 148
- CheckpointNode, 20, 66. *See also* SecondaryNameNode.
- Checkpoints
  - HDFS, 71
  - NameNode, recovering from, 240–241
- Cloud, installing Hadoop in. *See* Installing Hadoop in the cloud, with Whirr.
- Cloud tools. *See* Whirr.
- Clusters
  - benchmarking, examples, 245
  - coordinating services across, 17
  - executing MapReduce process V1 on, 11–12
  - heatmaps, 187–188
  - installing. *See* Installing Hadoop, with Ambari.
  - installing Hadoop in the cloud, with Whirr, 59–61
  - installing Spark on, 257–258
  - launching the cluster, 59–61
  - MPI (Message Passing Interface) and Hadoop on the same, 183
  - nodes. *See* NodeManagers.
  - preparing notes for Hadoop installation, 43–44
  - security settings, 193
  - starting Spark across, 258–259
  - stress testing, 245
  - taking down, 61
- Cluster-wide settings
  - properties, 189–191, 198–204
  - security, 193
- Code samples used in this book, online resources, 227
- Command line scripts, compatibility with MapReduce and Hadoop V2, 224
- Commands. *See also specific commands.*
  - HDFS, 75–77
  - Hive, 135–136
  - Oozie, 163

- Compatibility, MapReduce and Hadoop V2
    - ApplicationMaster restarts, enabling, 222
    - command line scripts, 224
    - Hive queries under YARN, 225
    - `rmadmin` function. *See* `rmadmin` function.
    - node capacity, calculating, 222
    - Oozie workflows under YARN, 225
    - `org.apache.hadoop.mapred` APIs, 224
    - `org.apache.hadoop.mapreduce` APIs, 224
    - overview, 222
    - Pig scripts under YARN, 225
    - running V1 applications, 223
    - sample MapReduce settings, 223
  - Configuration check screen, Hue GUI, 254, 255
  - Configuration files. *See* XML configuration files.
  - Configuration screen, Hue GUI, 254, 255
  - Configuring
    - Hadoop services, 189–191
    - HDFS services and proxy users, 250–251
    - Hive Webchat service, 251
    - Hue, with Ambari, 250–252
    - Hue GUI, 252–253
  - Configuring an NFSv3 gateway
    - configuration files, 217–218
    - current NFSv3 capabilities, 217
    - mount HDFS, 219–220
    - overview, 217
    - starting the gateway, 218–219
    - user name, specifying, 218
  - Containers, YARN
    - cores, setting, 208
    - definition, 178
    - memory, setting, 207
    - monitoring, 184
    - running commands in, 174–176
  - Control flow nodes, 155–156
  - Coordinator jobs, 155
  - Copying files, 75
  - `core-default.xml` file, 205
  - `core-site.xml`, configuring, 31–32
  - Corrupt file blocks, listing, 213
  - Corrupted files, moving and deleting, 212
  - `create` command, HBase, 165–166
  - `CREATE TABLE` command, 135–136
  - Cutting, Doug, 3
- ## D
- DAGs (directed acrylic graphs), 154
  - Dashboard view, Ambari, 186–189
  - Dashes (--), Pig comment delimiters, 133
  - Data directories, creating, 31
  - Data lakes. *See also* Big Data.
    - vs.* data warehouses, 5–6
    - definition, 5
    - schema on read, 5
    - schema on write, 5
    - vs.* traditional data storage, 5–6
    - traditional data transform. *See* ETL (extract, transform, and load).
  - Data locality. *See also* Rack awareness.
    - design aspect of MapReduce, 64
    - HDFS, 64
    - levels of, 69
  - Data streams, acquiring. *See* Flume.
  - Data summation. *See* Hive.
  - Data transfer format, 150
  - Data warehouses
    - analyzing large sets of data. *See* Hive.
    - bottlenecks, resolving, 8
    - vs.* data lakes, 5–6
  - Databases
    - distributed, online resources, 246
    - in the Hadoop ecosystem, 16
    - listing, 144
    - MySQL, loading, 142–143
    - non-relational. *See* HBase.
    - relational. *See* Sqoop.
    - tables, listing, 144
  - DataNodes, 64–67
  - Datanodes tab, NameNode UI, 209–210
  - Debugging. *See* MapReduce, debugging; Troubleshooting.
  - Decommissioning
    - HDFS nodes, 214
    - YARN nodes, 206
  - `delete` command, Flume, 167
  - `-delete` command, Sqoop, 148
  - `-delete` option, 212
  - `deleteall` command, 167

Deleting

- all HDFS data, 239
- bypassing the Trash directory, 75
- cells, 167
- corrupted files, 212
- data from tables, 148
- directories, 75, 77
- files, 75
- rows, 167
- tables. *See* Dropping tables.

dfs command (deprecated), 72

dfs.webhdfs.enabled property, setting, 250–251

Directed acrylic graphs (DAGs), 154

Directories

- deleting, 75, 77
- making, 76

disable command, 167

Documentation, online resources, 20, 233

Downloading/uploading files

- Flume, 151
- Hadoop, 30
- HDFS, 72
- NFS Gateway, 72
- Oozie examples, 156–157
- Sandbox, 23
- Sqoop examples, 142–143
- VirtualBox, 23

drop command, HBase, 167

DROP TABLE command, 135–136

Dropping tables with

- HBase, 167
- Hive, 135–136
- MySQL, 148

Dryad on YARN, 182

Dynamic application management, online resources, 183–184

**E**

End users

- adding to HDFS, 211–212
- creating, 31
- essential tasks, 7

Errors and messages, Java, 233–234

ETL (extract, transform, and load), 5.

Examples. *See also* Online resources, examples.

Big Data, 4–5

C application, 82–83

of code used in this book, online resources, 227

Flume, 151–154

HBase. *See* HBase, examples.

Hive, 134–139

Java application, 78–81

MapReduce. *See* MapReduce, examples.

NameNode Federation, 71

Pig, 132–134

querying movie reviews, 136–139

YARN Distributed-Shell. *See* YARN Distributed-Shell, examples.

exit command

- HBase, 164
- Hive, 136

Exporting data from HDFS, 147–148. *See also* Sqoop.

## F

Fair scheduler, online resources, 221

Falcon, in the Hadoop ecosystem, 17

Fault tolerance, 107–108

File system checks, 212–213

Files

- configuration. *See* XML configuration files.
- copying, 75
- corrupted, moving, 212
- deleting, 75
- downloading/uploading. *See* Downloading/uploading files.
- listing, 75
- printing, 212–213
- files option, 212

Flink, 183

Flume

- channel component, 148
- components, 148
- configuration files, 153–154
- downloading, 151
- examples, 151–154
- flume.ng command, 151–152
- in the Hadoop ecosystem, 17
- install command, 151
- installing Flume, 151

- installing Telnet, 151
- log, checking, 153
- online resources, 246
- pipelines, 149–150
- sink component, 148
- source component, 148
- tail command, 153
- testing, 151–152
- uses for, 148
- weblog example, 152–153
- website, 153, 170

Flume User Guide, 150

flume.ng command, 151–152

Fork/join nodes, 155–156

Formatting an HDFS, 239–240

fsimage\_\* file

- definition, 66
- location, 66
- updating, 67

**G**

-get command, 75

get command, 166

Getting

- columns, 166
- rows, 166
- table cells, 166

GFS (Google File System), 64

Giraph, 181

Google BigTable, 163

Google Protocol Buffers, online resources, 179

Graph processing, 181

grep chaining, 121–124

Grep examples, online resources, 245

grep.java command, 121–124

GroupLens Research website, 135

Groups, creating, 31

grunt> prompt, 133

GUI (graphical user interface). *See* Ambari; HDFS, web GUI; Hue GUI.

## H

Hadoop

- core components, 1–2, 19–21. *See also* NameNode; *specific components*.
- daemons, starting and stopping, 238–239
- definition, 1
- history of, 3–4
- key features, 1–2
- name origin, 3
- prominent users, 3–4
- uses for, 3

Hadoop, V1

- design principles, 7–8
- MapReduce process, 8–12

Hadoop, V2

- design overview, 13–15
- MapReduce compatibility. *See* Compatibility, MapReduce and Hadoop V2.
- YARN, operation design, 13–15

Hadoop administration, 7. *See also* Ambari GUI.

Hadoop database, in the Hadoop ecosystem, 16

Hadoop Distributed File System (HDFS). *See* HDFS (Hadoop Distributed File System).

Hadoop ecosystem, components

- administration, 17
- Avro, 17
- Falcon, 17
- Flume, 17
- Hadoop database, 16
- HBase, 16
- HCatalog, 16
- HDFS, 15
- Hive, 16–17
- import/export, 17
- machine learning, 17
- Mahout, 17
- MapReduce, 15
- MapReduce query tools, 16–17
- Oozie, 17
- Pig, 16
- Sqoop, 17
- workflow automation, 17
- YARN, 15
- YARN application frameworks, 17
- Zookeeper, 17

- Hadoop services
  - assigning to hosts, 49–52
  - core services, 19–21. *See also* HDFS (Hadoop Distributed File System); YARN (Yet Another Resource Negotiator).
- Hadoop services, managing with Ambari
  - configuring services, 189–191
  - listing service accounts, 193
  - reporting service interruptions, 194–195
  - resolving service interruptions, 195–198
  - restarting after configuration change, 200–201
  - reverting to a previous version, 202–204
  - Services view, 189–191
  - starting and stopping services, 190–191
- Hadoop User Experience (Hue) GUI. *See* Hue (Hadoop User Experience) GUI.
- Hadoop V1 applications, compatibility with MapReduce, 224–225
- HADOOP\_HOME, setting, 30–31
- Hamster, 183
- HBase
  - data model, 164
  - distributed databases, online resources, 246
  - features, 163
  - in the Hadoop ecosystem, 16
  - overview, 163
  - website, 164
  - on YARN, 181–182
- HBase, examples
  - ‘ ‘ (single quotes), argument delimiters, 164
  - adding bulk data, 168
  - bash scripts, 167
  - create command, 165–166
  - creating a database, 165–166
  - delete command, 167
  - deleteall command, 167
  - deleting cells, 167
  - deleting rows, 167
  - disable command, 167
  - drop command, 167
  - dropping tables, 167
  - exit command, 164
  - exiting HBase, 164
  - get command, 166
  - getting a row, 166
  - getting columns, 166
  - getting table cells, 166
  - importing tab-delimited data, 168
  - ImportTsv command, 168
  - inspecting a database, 166
  - required software environment, 164
  - scan command, 166
  - scripting input, 167
  - shell command, 164
  - starting HBase, 164
  - web interface, 168–169
- HCatalog in the Hadoop ecosystem, 16
- HDFS (Hadoop Distributed File System)
  - BackupNode, 71
  - backups, 71. *See also* Checkpoints, HDFS.
  - block replication, 67–68
  - checkpoints, 71
  - components, 64–67. *See also* DataNodes; NameNode.
  - data locality, 64. *See also* Rack awareness.
  - definition, 2
  - deleting data, 239
  - design features, 63–64
  - in the Hadoop ecosystem, 15
  - master/slave architecture, 65
  - NameNode. *See* NameNode.
  - NFS Gateway, 72
  - NFSv3 mounting, 72
  - permissions, online resources, 211
  - roles, 67
  - Safe Mode, 68
  - snapshots, 71–72. *See also* Backups, HDFS.
  - status report, 77
  - streaming data, 72
  - uploading/downloading files, 72
  - version, getting, 73
  - web GUI, 77, 89–95
- HDFS administration
  - adding users, 211–212
  - balancing HDFS, 213–214
  - block reports, 213
  - checking HDFS health, 212–213
  - data snapshots, 213
  - decommissioning HDFS nodes, 214
  - deleting corrupted files, 212
  - Hadoop, online resources, 247

- listing corrupt file blocks, 213
- monitoring HDFS. *See* NameNode UI.
- moving corrupted files, 212
- NameNode UI, 208–211
- online resources, 226
- permissions, online resources, 211
- printing block locations, 213
- printing files, 212–213
- printing network topology, 213
- Safe Mode, 214
- SecondaryNameNode, verifying, 214–215
- snapshots, 215–216
- snappable directories, 215
- HDFS administration, configuring an NFSv3 gateway
  - configuration files, 217–218
  - current NFSv3 capabilities, 217
  - mount HDFS, 219–220
  - overview, 217
  - starting the gateway, 218–219
  - user name, specifying, 218
- hdfs command, 72
- hdfs fsck command, 212
- HDFS in pseudo-distributed mode
  - configuring, 31–32
  - starting, 35–36
  - verifying, 37
- HDFS programming examples
  - C application, 82–83
  - Java application, 78–81
- HDFS user commands
  - copying files, 75
  - deleting directories, 77
  - deleting files, 75
  - dfs (deprecated), 72
  - general, 73–75
  - generic options, 74–75
  - get, 75
  - hdfs, 72
  - HDFS status report, 77
  - HDFS version, getting, 73
  - help for, 73
  - listing, 73
  - listing files, 75
  - ls, 75
  - making a directory, 76
  - mkdir, 75
  - overview, 72–73
  - put, 75
  - report, 77
  - rm, 75
  - rm, 77
  - skipTrash option, 76
  - version option, 73
- hdfs-default.xml file, 205
- hdfs-site.xml, configuring, 32
- Help
  - in books and publications. *See* Books and publications.
  - for HDFS commands, 73
  - on websites. *See* Online resources.
- High availability. *See* NameNode HA (High Availability).
- Hive. *See also* ETL (extract, transform, and load).
  - ; (semicolon), command terminator, 135
  - creating tables, 135–136
  - dropping tables, 135–136
  - examples, 134–139
  - exiting, 136
  - features, 134
  - in the Hadoop ecosystem, 16–17
  - installing, 39
  - queries under YARN, compatibility with MapReduce and Hadoop V2, 225
  - query language, online resources, 170
  - SQL-like queries, online resources, 246
  - starting, 135
  - Webchat service, configuring, 251
- hive command, 135
- Hive commands
  - CREATE TABLE, 135–136
  - DROP TABLE, 135–136
  - exit, 136
- hive> prompt, 135
- HiveQL, 134
- Hortonworks HDP 2.2 Sandbox. *See* Sandbox.
- Hortonworks Sandbox, online resources, 244
- Hosts, assigning services to, 49–52
- Hosts view, Ambari, 191–192
- Hoya, 181–182
- HPC (high-performance computing), 183

Hue (Hadoop User Experience) GUI  
 configuration check screen, 254, 255  
 configuration screen, 254, 255  
 configuring, 252–253  
 configuring HDFS services and proxy users, 250–251  
 configuring Hive Webchat service, 251  
 configuring with Ambari, 250–252  
 description, 249  
 initial login screen, 254  
 installing, 249–253  
 logging in, 253–254  
 main icon bar, 254, 256  
 online resources, 249, 254  
 Oozie workflow, modifying, 252  
 starting, 253  
 user interface, 253–256  
 Users administration screen, 255, 256  
 hue start command, 253

## I

import command, 145–146  
 Import/export in the Hadoop ecosystem, 17  
 Importing data. *See also* Sqoop.  
   cleaning up imported files, 148  
   ImportTsv utility, 168  
   from MySQL, 144–147  
   tab-delimited, 168  
   viewing imported files, 145  
 ImportTsv command, 168  
 -includeSnapshots option, 213  
 -info command, Oozie, 163  
 INFO messages, turning off, 237  
 install command, 151, 252  
 Installation recipes, online resources, 243–244  
 Installing  
   Flume, 151  
   Hive, 39  
   Hue GUI, 249–253  
   Pig, 38–39. *See also* Installing Hadoop, in pseudo-distributed mode.  
   Sqoop, 143  
   Telnet, 151  
   YARN Distributed-Shell, 172–174  
 Installing Hadoop

in the cloud. *See* Installing Hadoop in the cloud, with Whirr.  
 single-machine versions. *See* Installing Hadoop, in pseudo-distributed mode; Installing Sandbox.  
 Installing Hadoop, in pseudo-distributed mode. *See also* Installing, Hive; Installing, Pig.  
   configuring a single-node YARN server, 30  
   core components, 30  
   core-site.xml, configuring, 31–32  
   data directories, creating, 31  
   downloading Hadoop, 30  
   groups, creating, 31  
   HADOOP\_HOME, setting, 30–31  
   HDFS, configuring, 31–32  
   HDFS services, starting, 35–36  
   HDFS services, verifying, 37  
   hdfs-site.xml, configuring, 32  
   Java heap sizes, modifying, 34  
   JAVA\_HOME, setting, 30–31  
   log directories, creating, 31  
   mapred-site.xml, configuring, 33  
   MapReduce, example, 37–38  
   MapReduce, specifying a framework name, 33  
   minimal hardware requirements, 30  
   NameNode, formatting, 34  
   NodeManagers, configuring, 33  
   overview, 29–30  
   users, creating, 31  
   YARN services, starting, 36  
   YARN services, verifying, 37  
   yarn-site.xml, configuring, 33  
 Installing Hadoop, with Ambari  
   Ambari console, 45–55  
   Ambari dashboard, 41  
   Ambari screen shots, 46–55  
   ambari-agent, 47–49  
   ambari-server, 44–45  
   assigning services to hosts, 49–52  
   overview, 40–41, 42  
   requirements, 42–43  
   undoing the install, 55–56  
   warning messages, 53, 55  
 Installing Hadoop in the cloud, with Whirr

- benefits of, 56
- configuring Whirr, 57–59
- installation procedure, 57
- overview, 56–57
- Installing Sandbox, VirtualBox
  - connecting to the Hadoop appliance, 28–29
  - downloading, 23
  - loading, 25–28
  - minimum requirements, 23
  - overview, 23–24
  - saving, 29
  - shutting down, 29
  - starting, 24–25
  - virtual machine, definition, 24
- Internet, as a troubleshooting tool, 235

## J

- Java
  - applications, HDFS programming
    - examples, 78–81
  - errors and messages, 233–234
  - heap sizes, modifying, 34
  - and MapReduce, online examples, 245
  - on MapReduce, 111–116
  - supported versions, online resources, 62, 243
  - WordCount example program, 111–116
- JAVA\_HOME, setting, 30–31
- JIRA issues, tracking, 235
- JIRAs, online resources, 235
- job command, 97–98
- Job information, getting, 89–95
- Job types, Oozie, 154–155
- JobHistoryServer, 20, 207
- job.properties file, 157
- Jobs, MapReduce. *See* MapReduce jobs.

## K

- kill command, Oozie, 163
- kill option, 97–98, 125, 207
- Killing
  - MapReduce jobs, 97–98, 125
  - Oozie jobs, 163
  - YARN applications, 207

## L

- list option, 97–98, 125, 207
- list-corruptfileblocks option, 213
- Listing
  - corrupt file blocks, 213
  - database tables, 144
  - databases, 144
  - files, 75
  - HDFS commands, 73
  - installed software, 193
  - MapReduce jobs, 97–98, 125, 207
  - service accounts, 193
- locations option, 213
- Log aggregation, 125–126
- Log management
  - enabling log aggregation, 125–128
  - online resources, 235
- Logging in to Hue GUI, 253–254
- Login screen, Hue GUI, 254
- Logs
  - checking, Flume, 153
  - creating directories, 31
  - troubleshooting, 234–235
- logs command, Oozie, 163
- ls command, 75

## M

- Machine learning in the Hadoop
  - ecosystem, 17
- Mahout in the Hadoop ecosystem, 17
- Main icon bar, Hue GUI, 254, 256
- Management screen in Ambari, opening, 194
- Managing. *See also* ResourceManager.
  - applications, dynamically, 183–184
  - Hadoop, graphical interface for. *See* Ambari.
  - HDFS. *See* HDFS administration.
  - hosts, with Ambari, 191–192
  - log, 125–128
  - MapReduce jobs, 97–98, 125
  - workflow, MapReduce, V1, 10
  - YARN. *See* YARN administration.
- mapred commands. *See* specific commands.
- mapred.child.java.opts property, 208
- mapred-default.xml file, 205
- mapred-site.xml, configuring, 33, 207
- mapred-site.xml file, 222–223



- MapReduce
  - chaining, 121–124
  - compatibility with Hadoop V2. *See* Compatibility, MapReduce and Hadoop V2.
  - data locality, 64
  - debugging examples, online resources, 245
  - example, pseudo-distributed mode, 37–38
  - fault tolerance, 107–108
  - in the Hadoop ecosystem, 15
  - hardware, 108
  - heap size, setting, 208
  - log management, 125–127
  - node capacity, compatibility with MapReduce and Hadoop V2, 222
  - parallel data flow, 104–107
  - Pipes interface, 119–121
  - programming, online resources, 245
  - programming examples, online resources, 245
  - properties, setting, 208
  - query tools in the Hadoop ecosystem, 16–17
  - replacing with Tez engine, 181
  - resource limits, setting, 208
  - vs.* Spark, 182
  - specifying a framework name, 33
  - speculative execution, 108
  - streaming interface, 116–119
  - in the YARN framework, 181
- MapReduce, debugging
  - checking job status, 125
  - example, online resources, 245
  - killing jobs, 125
  - listing jobs, 125
  - log aggregation, 125–126
  - log management, 125–128
  - parallel applications, 124
  - recommended install types, 124–125
  - viewing logs, 127–128
- MapReduce, examples
  - computational model, 101–104
  - files for, 85–86
  - listing available examples, 86–87
  - monitoring, 89–95
  - pi program, 37–38, 87–89
  - V1, 8–10
  - War and Peace* example, 101–104
  - word count program, 101–104, 104–107. *See also* WordCount example program.
- MapReduce, multi-platform support
  - C++, 119–121
  - Java, 111–116
  - Python, 116–119
- MapReduce computational model
  - example, 101–104
  - important properties, 103
  - mapping process, 101–104
- MapReduce jobs
  - finding, 97–98, 125
  - killing, 97–98, 125
  - listing, 97–98, 125
  - managing, 97–98, 125
  - status check, 97–98, 125, 207
- MapReduce process, V1
  - advantages, 10–11
  - availability, 12
  - basic aspects, 10
  - design principles, 7–8
  - example, 8–10
  - fault tolerance, 11
  - Job Tracker, 12
  - limitations, 12
  - managing workflow, 10
  - mapping step, 8–10
  - master control process, 12
  - reducing step, 8–10
  - resource utilization, 12
  - scalability, 10, 12
  - support for alternative paradigms and services, 12
  - Task Tracker, 12
  - typical job progress, 12
- MapReduce process, V2
  - advantages of, 14–15
  - Job Tracker, 13
  - overview, 13
  - typical job progress, 13–14
- MapReduce V2 administration, online resources, 226, 247
- mapreduce.am.max-attempts property, 222
- mapreduce.map.java.opts property, 223
- mapreduce.map.memory.mb property, 208, 223

mapreduce.reduce.java.opts property, 208, 223

mapreduce.reduce.memory.mb property, 208, 223

Master user job, YARN applications, 178–179

Masters. *See* NameNode.

Master/slave architecture in HDFS, 65

Messages, during an Ambari installation, 53

Messages and errors, Java, 233–234

Metadata, tracking changes to, 66. *See also* CheckPointNode.

Microsoft projects. *See specific projects.*

-mkdir command, 75

Monitoring. *See also* Zookeeper.

- applications, 89–95
- HDFS. *See* NameNode UI.
- MapReduce examples, 89–95
- YARN containers, 184

-move option, 212

Movie reviews, query example, 136–139

MovieLens website, 135

Moving computation *vs.* moving data, 8

Moving corrupted files, 212

rmadmin function. *See* rmadm function.

Murthy, Arun C., 13

MySQL databases, loading, 142–143

## N

NameNode

- description, 64–67
- formatting, 34
- health, monitoring. *See* Zookeeper.
- periodic checkpoints, 66

NameNode, troubleshooting

- checkpoint, recovering from, 240–241
- failure and recovery, 239–241
- reformatting, 239–240

NameNode Federation

- description, 22
- example, 71
- key benefits, 70

NameNode HA (High Availability), 69–70

NameNode UI

- Datanodes tab, 209–210
- directory browser, 211
- overview, 208–209

- Overview tab, 209
- Snapshot tab, 209, 216
- startup progress, 209–210

Namespace Federation, 70–71

Network topology, printing, 213

newbandwidth option, 213

NFSv3 gateway, configuring

- configuration files, 217–218
- current NFSv3 capabilities, 217
- mounting HDFS, 219–220
- overview, 217
- starting the gateway, 218–219
- user name, specifying, 218

NFSv3 Gateway, mounting an HDFS, 72

“Nobody Ever Got Fired for Using Hadoop,” 4

Node capacity, compatibility with

- MapReduce and Hadoop V2, 222

Node types, Oozie, 155–156

NodeManagers

- configuring, 33
- definition, 20

## O

Official Hadoop sources, 62

Online resources. *See also* Books and publications.

- balancing HDFS, 214
- “Big Data Surprises,” 4
- Capacity scheduler administration, 221, 226
- downloading Hadoop, 30
- Dryad on YARN, 182
- dynamic application management, 183–184
- essential tools, 245–246
- Fair scheduler, 221
- Flink, 183
- Flume, streaming data and transport, 246
- Flume configuration files, 153–154
- Flume User Guide, 150
- Flume website, 153, 170
- Google BigTable, 163
- Google Protocol Buffers, 179
- GroupLens Research website, 135
- HBase distributed database, 246
- HBase on YARN, 182
- HBase website, 164

- Online resources (*continued*)
  - HDFS administration, 226
  - HDFS permissions, 211
  - Hive query language, 170
  - Hive SQL-like queries, 246
  - Hoya, 182
  - Hue GUI, 254
  - log management, 235
  - MovieLens website, 135
  - MPI (Message Passing Interface), 183
  - official Hadoop sources, 62
  - Oozie, 170
  - Oozie workflow manager, 246
  - Pig scripting, 134, 170, 245
  - Pig website, 132
  - REEF, 183
  - Sandbox, 62
  - Secure Mode Hadoop, 22
  - Slider, 183–184
  - Spark, 182, 260
  - SQL-like query language, 170
  - Sqoop, data import/export, 170
  - Sqoop RDBMS import/export, 246
  - Storm, 182
  - supported Java versions, 62
  - Tez, 181
  - VirtualBox, 62
  - Whirr, 56, 62
  - XML configuration files, 206
  - XML configuration files, description, 62
  - YARN administration, 226, 247
  - YARN application frameworks, 246
  - YARN development, 184, 246
- Online resources, Ambari
  - administration, 246
  - Ambari Installation Guide, 42
  - Ambari Views, 193
  - installation guide, 62, 244
  - project page, 62, 244
  - project website, 204
  - troubleshooting guide, 62, 244
  - Whirr cloud tools, 244
- Online resources, examples
  - debugging, 245
  - Grep, 245
  - Java MapReduce, 245
  - MapReduce programming, 245
  - Pi benchmark, 244
  - Pipes, 245
  - streaming data, 245
  - Terasort benchmark, 244
- Online resources, Hadoop
  - administration, 247
  - Capacity scheduler administration, 247
  - code and examples used in this book, 227
  - documentation page, 20
  - general information, 227
  - HDFS administration, 247
  - Hortonworks Sandbox, 244
  - installation recipes, 243–244
  - JIRAs, 235
  - MapReduce V2 administration, 247
  - Oracle VirtualBox, 244
  - supported Java versions, 243
  - XML configuration files, 243
  - YARN administration, 247
- Online resources, HDFS
  - background, 244
  - Java programming, 244
  - libhdfs programming in C, 244
  - user commands, 244
- Online resources, MapReduce
  - background, 245
  - debugging, example, 245
  - Grep example, 245
  - Java MapReduce example, 245
  - MapReduce V2 administration, 226
  - Pipes example, 245
  - programming, 245
  - streaming data example, 245
- Oozie
  - action flow nodes, 155–156
  - bundle jobs, 155
  - command summary, 163
  - control flow nodes, 155–156
  - coordinator jobs, 155
  - DAGs (directed acrylic graphs), 154
  - fork/join nodes, 155–156
  - in the Hadoop ecosystem, 17
  - job types, 154–155
  - node types, 155–156
  - online resources, 170
  - overview, 154
  - workflow jobs, 154–155
- Oozie, examples
  - demonstration application, 160–162

- downloading examples, 156–157
- job.properties file, 157
- MapReduce, 156–160
- Oozie is not allowed to
  - impersonate Oozie error, 159
- oozie option, setting, 159
- OOZIE\_URL environmental variable, 159
- required environment, 156
- workflow.xml file, 157
- Oozie is not allowed to
  - impersonate Oozie error, 159
- oozie option, setting, 159
- Oozie workflow
  - modifying in the Hue GUI, 252
  - online resources, 246
  - under YARN, 225
- OOZIE\_URL environmental variable, 159
- openforwrite option, 213
- Oracle VirtualBox
  - installing. *See* Installing Sandbox, VirtualBox.
  - online resources, 62, 244
- org.apache.hadoop.mapred APIs, 224
- org.apache.hadoop.mapreduce APIs, 224
- Overview tab, NameNode UI, 209

## P

- Parallel applications, debugging, 124
- Parallel data flow, 104–107
- Parallelizing SQL queries, 146
- Passwords
  - Ambari, 45, 186
  - changing, 45, 186
  - Hadoop appliance, 28
  - Hue, 253
  - Sqoop, 143, 145
- Performance improvement
  - HPC (high-performance computing), 183
  - YARN framework, 182
- Permissions for HDFS, online resources, 211
- Pi benchmark, online resources, 244
- pi program, MapReduce example, 37–38
- Pig
  - (dashes), comment delimiters, 133
  - /\* \*/ (slash asterisk...), comment delimiters, 133

- ; (semicolon), command terminator, 133
- example, 132–134
- grunt> prompt, 133
- in the Hadoop ecosystem, 16
- installing, 38–39
- online resources, 132
- running from a script, 133–134
- Tez engine, 131–134
- uses for, 131–132
- website, 132

### Pig scripting

- online resources, 134, 170, 245
- under YARN, 225

### Pipelines, Flume, 149–150

### Pipes

- examples, online resources, 245
- interface, 119–121

Planning resources. *See* Resource planning; *specific resources.*

### Prepackaged design elements, 182–183

### Printing

- block locations, 213
- files, 212–213
- network topology, 213

Processing unbounded streams of data, 182

Programming MapReduce, online resources, 245

Progress window in Ambari, disabling, 194

Pseudo-distributed mode. *See* Installing

Hadoop, in pseudo-distributed mode.

-put command, 75

Python, 116–119

## R

Rack awareness, 68–69. *See also* Data locality.

-racks option, 213

RDBMS (relational database management system). *See* Sqoop.

REEF (Retainable Evaluator Execution Framework), 182–183

### Reference material

publications. *See* Books and publications. on websites. *See* Online resources.

Reformatting NameNode, 239–240

Relational database management system (RDBMS). *See* Sqoop.

- Relational databases, transferring data
    - between. *See* Sqoop.
  - Removing. *See* Deleting.
  - report command, 77
  - Reporting service interruptions, 194–195
    - rerun command, 163
  - Resolving service interruptions, 195–198
  - Resource management. *See* Oozie; YARN.
  - Resource planning. *See also specific resources.*
  - Resource planning, hardware, 21–22
  - Resource planning, software
    - building a Windows package, 22
    - installation procedures. *See specific software.*
    - NameNode Federation, 22
    - NameNode HA (High Availability), 22
    - official supported versions, 22
    - Secure Mode Hadoop, 22
  - ResourceManager. *See also* NodeManager; Oozie; YARN.
    - definition, 20
    - web GUI, monitoring examples, 89–95
  - Restarting after configuration change, 200–201
    - resume command, 163
  - Retainable Evaluator Execution Framework (REEF), 182–183
  - Reverting to a previous version of Hadoop, 202–204
    - rm command
      - deleting directories, 77
      - deleting files, 75
    - rmadmin function, 224
  - Rows, deleting, 167
    - run command, 163
- S**
- Safe Mode, 68, 214
    - safemode enter command, 214
    - safemode leave command, 214
  - Sandbox
    - installing. *See* Installing Sandbox.
    - online resources, 62
  - Scalable batch and stream processing, 183
  - scan command, 166
  - scheduler.minimum-allocation-mb property, 207
  - Scheduling. *See* Capacity scheduler; Oozie; ResourceManager; YARN.
  - Schema on read, 5
  - Schema on write, 5
  - Scripting
    - command line scripts, compatibility with MapReduce and Hadoop V2, 224
    - HBase input, 167
    - with Pig, 133–134, 170, 225, 245
  - SecondaryNameNode, 20, 214–215. *See also* CheckPointNode.
  - Secure Mode Hadoop, online resources, 22
  - Semicolon (;), Pig command
    - terminator, 133
  - Service interruptions
    - reporting, 194–195
    - resolving, 195–198
  - Services. *See* Hadoop services.
  - Services view, Ambari, 189–191
  - shell command, 164
    - shell\_args option, 176–178
  - Simplifying solutions, 235
  - Single quotes ( ' '), HBase argument
    - delimiters, 164
  - Sink component of Flume, 148
    - skipTrash option, 76
  - Slash asterisk... (/ \* \*/), Pig comment
    - delimiters, 133
  - Slaves. *See* DataNodes.
  - Slider, online resources, 183–184
    - snapshot command, 71
  - Snapshot tab, NameNode UI, 209, 216
  - Snapshots, HDFS
    - description, 71–72
    - including data from, 213
    - overview, 215
    - restoring deleted files, 215–216
    - sample screen, 216
    - shapshotable directories, 215
  - Snapshottable directories, 215
  - Source component of Flume, 148
  - Spark
    - installing, 260
    - vs. MapReduce, 257
    - online resources, 260
  - Speculative execution, 108
    - split-by option, 146
  - SQL queries, parallelizing, 146

- SQL-like query language, online resources, 170. *See also* Hive.
  - Sqoop
    - data import/export, online resources, 170
    - database connectors, 141
    - in the Hadoop ecosystem, 17
    - importing/exporting data, 139–142
    - overview, 139
    - RDBMS import/export, online resources, 246
    - version changes, 140–142
  - Sqoop, examples
    - cat command, 145
    - cleaning up imported files, 148
    - delete command, 148
    - deleting data from tables, 148
    - downloading Sqoop, 142–143
    - drop command, 148
    - dropping tables, 148
    - exporting data from HDFS, 147–148
    - import command, 145–146
    - importing data from MySQL, 144–147
    - installing Sqoop, 143
    - listing database tables, 144
    - listing databases, 144
    - loading MySQL database, 142–143
    - overview, 142
    - parallelizing SQL queries, 146
    - setting Sqoop permissions, 143–144
    - software environment, 142
    - split-by option, 146
    - viewing imported files, 145
    - wget command, 143
  - start command, 163
  - Starting and stopping. *See also specific programs.*
    - Hadoop daemons, 238–239
    - Hadoop services, 190–191
  - Status check, MapReduce jobs, 97–98, 125, 207
  - status option, 97–98, 125, 207
  - Status widgets, Ambari, 186–189
  - stderr, MapReduce log, 125
  - stdout, MapReduce log, 125
  - Storm, 182
  - Streaming data
    - examples, online resources, 245
    - from HDFS, 72
  - Streaming interface, 116–119
  - submit command, 163
  - suspend command, 163
  - syslog, MapReduce log, 125
  - System administration. *See* HDFS administration; YARN administration.
- ## T
- Tables
    - cells, getting, 166
    - creating, 135–136
    - deleting. *See* Tables, dropping.
    - deleting data from, 148
    - listing, 144
  - Tables, dropping with
    - HBase, 167
    - Hive, 135–136
    - MySQL, 148
  - tail command, 153
  - Task neutrality, YARN, 13
  - Terasort benchmark, 244
  - terasort test, 95–96
  - Terminal initialization failed error message, 40
  - TestDFSIO benchmark, 96–97
  - Testing Flume, 151–152
  - Tez engine.
    - Hive, 134
    - online resources, 181
    - Pig, 131–134
    - replacing MapReduce, 181
  - Tools, online resources, 245–246
  - Traditional data transform. *See* ETL (extract, transform, and load).
  - Transforming data. *See* Sqoop.
  - Trash directory, bypassing, 75
  - Troubleshooting
    - with Ambari, 234
    - Apache Hadoop 2 Quick-Start Guide*, 229–233
    - checking the logs, 234–235
    - debugging a MapReduce example, online resources, 245
    - deleting all HDFS data, 239
    - examining job output, 238
    - formatting HDFS, 239–240
    - Hadoop documentation online, 233
    - INFO messages, turning off, 237

Troubleshooting (*continued*)

- Java errors and messages, 233–234
- with MapReduce information streams, 235–238
- searching the Internet, 235
- simplifying solutions, 235
- starting and stopping Hadoop daemons, 238–239
- this book as a resource, 229–233
- tracking JIRA issues, 235

Troubleshooting, NameNode

- checkpoint, recovering from, 240–241
- failure and recovery, 239–241
- reformatting, 239–240

Troubleshooting guide, online resources, 244

## U

Undoing an Ambari install, 55–56

Uploading files. *See* Downloading/uploading files.

uptime command, 174–175

User interface. *See* HDFS, web GUI; Hue (Hadoop User Experience) GUI.

Users. *See* End users.

Users administration screen, Hue GUI, 255

## V

version option, 73

Views view, Ambari, 193

VirtualBox

- installing. *See* Installing Sandbox, VirtualBox.
- online resources, 62, 244

## W

*War and Peace* example, 101–104

Warning messages. *See* Messages and errors.

Web interface. *See* HDFS, web GUI; ResourceManager, web GUI.

Webchat service, configuring, 251

WebProxy server, 206

wget command, 143

## Whirr

- cloud tools for Ambari, online resources, 244
- installing Hadoop in the cloud. *See* Installing Hadoop in the cloud, with Whirr.
- online resources, 56, 62

Word count program, example, 101–104, 104–107

## WordCount example program

- C++-based, 119–121
- grep chaining, 121–124
- grep.java command, 121–124
- Java-based, 111–116
- Python-based, 116–119

WordCount.java program, 111–116

Workflow, managing. *See* Oozie; Resource-Manager; YARN (Yet Another Resource Negotiator).

Workflow automation in the Hadoop ecosystem, 17

Workflow jobs, 154–155

workflow.xml file, 157

## X

XML configuration files

- description, 62
- editing, 20–21
- functions of, 205
- list of, 205
- location, 20
- online resources, 206, 243
- properties of, 206

## Y

YARN (Yet Another Resource Negotiator).

- Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2*, 17
- application frameworks, 13
- Capacity scheduler, 220–221
- definition, 2
- development, online resources, 246
- in the Hadoop ecosystem, 15

- job information, getting, 89–95
- log aggregation, 125–126
- operation design, 13–15
- scheduling and resource management, 13–15
- services, starting, 36
- task neutrality, 13
- YARN administration
  - container cores, setting, 208
  - container memory, setting, 207
  - decommissioning YARN nodes, 206
  - JobHistoryServer, 207
  - managing YARN applications, 178–179
  - managing YARN jobs, 207
  - MapReduce properties, setting, 208
  - online resources, 226, 247
  - YARN WebProxy server, 206
- YARN application frameworks
  - Dryad on YARN, 182
  - Flink, 183
  - Giraph, 181
  - graph processing, 181
  - Hadoop ecosystem, components, 17
  - in the Hadoop ecosystem, 17
  - Hamster, 183
  - HBase on YARN, 181–182
  - Hoya, 181–182
  - HPC (high-performance computing), 183
  - MapReduce, 181
  - online resources, 246
  - performance improvement, 182
  - prepackaged design elements, 182–183
  - processing unbounded streams of data, 182
  - REEF, 182–183
  - resource management, 180
  - scalable batch and stream processing, 183
  - Spark, 182
  - Storm, 182
  - Tez, 181
  - YARN Distributed-Shell, 180
- YARN applications
  - Apache Hadoop YARN: Moving beyond MapReduce and Batch Processing with Apache Hadoop 2*, 178
  - ApplicationMaster, 178–179
    - developing, online resources for, 184
    - killing, 207
    - managing, 178–179
    - master user job, 178–179
    - resource management, 178–179
    - structure of, 178–179
    - vs.* YARN components, 179–180
  - YARN applications, containers
    - definition, 178
    - monitoring, 184
    - running commands in, 174–176
  - YARN components *vs.* YARN applications, 179–180
  - YARN Distributed-Shell
    - description, 171
    - installing, 172–174
    - in the YARN framework, 180
  - YARN Distributed-Shell, examples
    - running commands in containers, 174–176
    - with shell arguments, 176–178
    - `-shell_args` option, 176–178
    - uptime command, 174–175
  - YARN server, single-node
    - configuration, 30
  - YARN WebProxy server, 206
  - `yarn-default.xml` file, 205
  - `yarn.manager.resource.memory-mb` property, 207
  - `yarn.nodemanager.resource.cpu-vcores` property, 208
  - `yarn.nodemanager.resource.memory-mb` property, 223
  - `yarn.nodemanager.vmem-pmem-ratio` property, 223
  - `yarn.resourcemanager.am.max-retries` property, 222
  - `yarn.scheduler.maximum-allocation-mb` property, 207, 223
  - `yarn.scheduler.maximum-allocation-vcores` property, 208
  - `yarn.scheduler.minimum-allocation-mb` property, 223
  - `yarn.scheduler.minimum-allocation-vcores` property, 208



- yarn-site.xml file
  - configuring, 33
  - enabling ApplicationMaster restarts, 222–223
  - important properties, 207
- Yet Another Resource Negotiator (YARN).
  - See YARN (Yet Another Resource Negotiator).

## **Z**

- Zookeeper, 17, 70