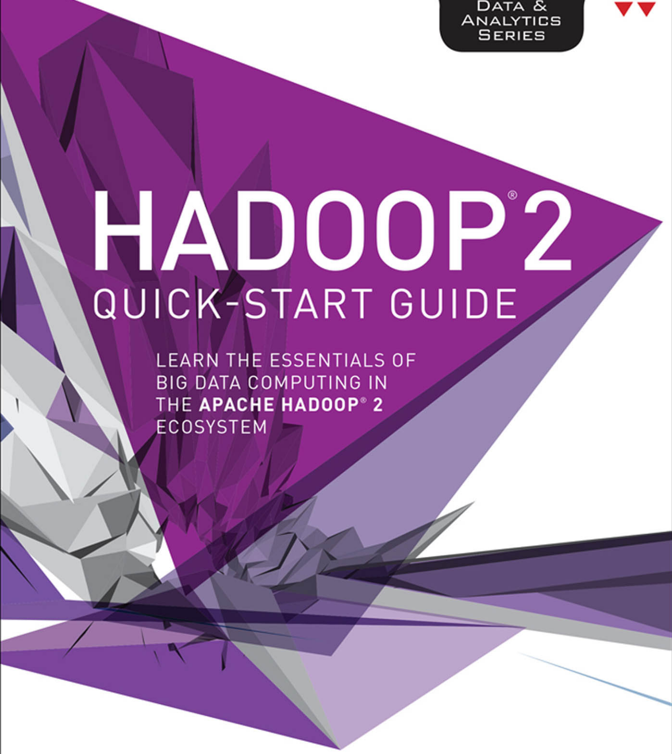
Addison Wesley Data & Analytics Series

Hadoop® 2 Quick-Start Guide

Learn the Essentials of Big Data Computing in the **Apache Hadoop® 2** Ecosystem



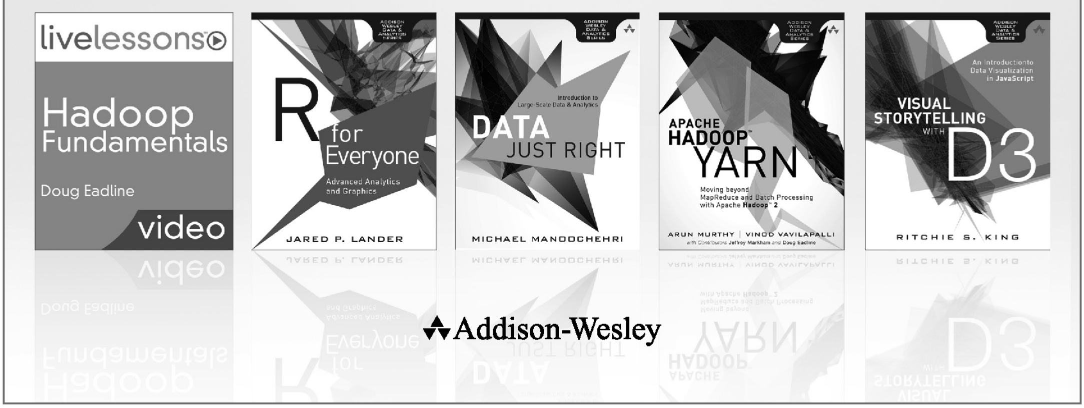
Douglas Eadline

Free Sample Chapter

Share with Others

Hadoop® 2 Quick-Start Guide

The Addison-Wesley Data and Analytics Series



Visit **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

informIT.com
the trusted technology learning source | Addison-Wesley | Safari Books Online

ALWAYS LEARNING PEARSON

Hadoop® 2 Quick-Start Guide

Learn the Essentials of Big Data Computing in the Apache Hadoop® 2 Ecosystem

Douglas Eadline

Addison-Wesley

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

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 or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the United States, please contact international@pearsoned.com.

Visit us on the Web: 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

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

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*.

4

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.
  dbcount: 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 π 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

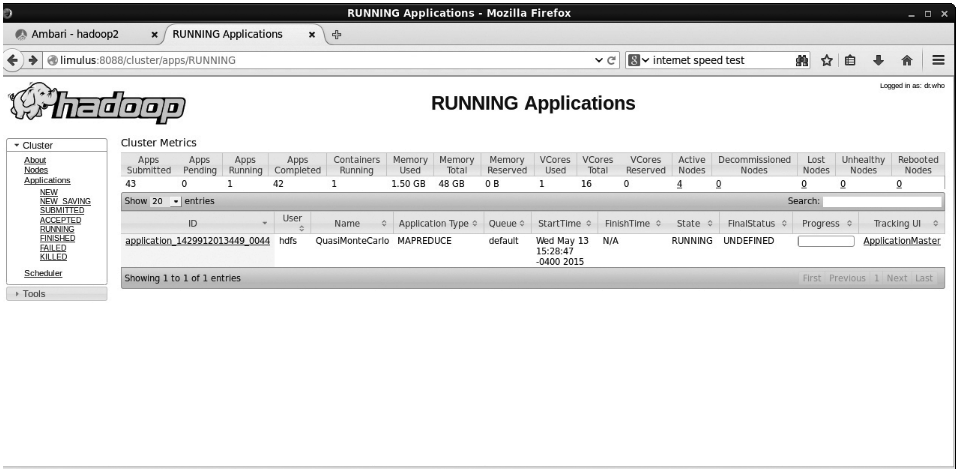


Figure 4.1 Hadoop RUNNING Applications web GUI for the pi example

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

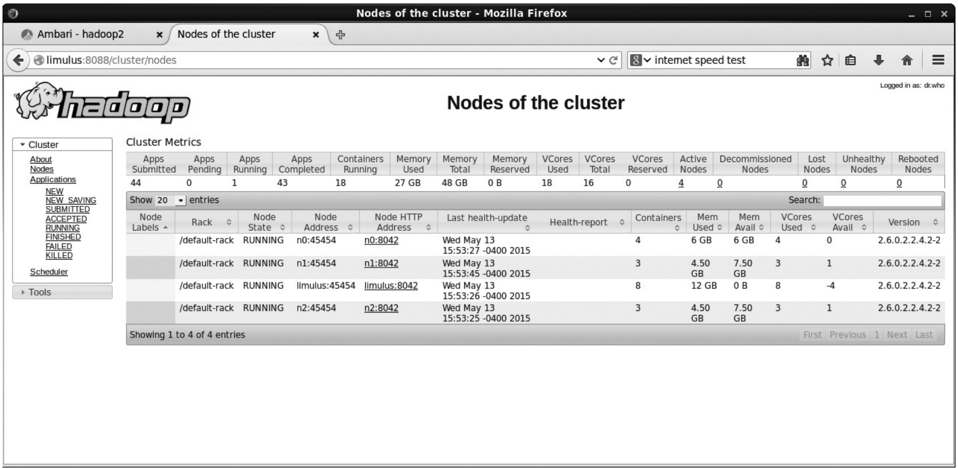


Figure 4.2 Hadoop YARN ResourceManager nodes status window

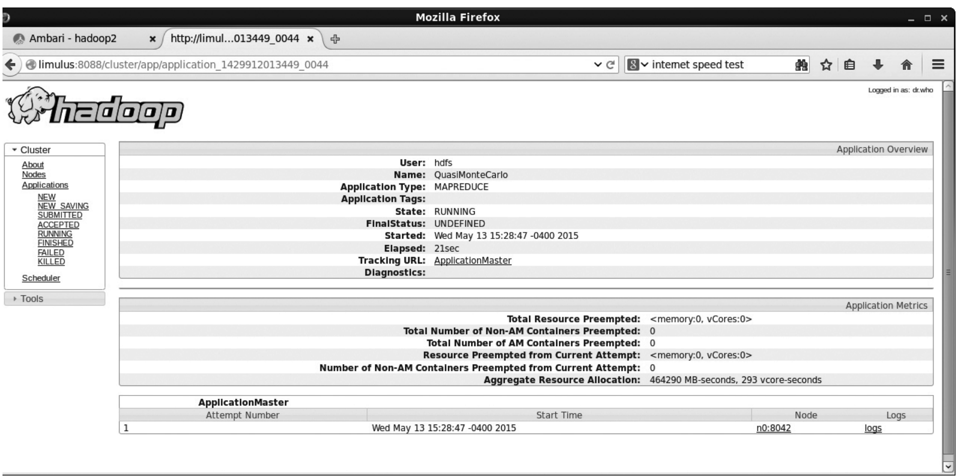


Figure 4.3 Hadoop YARN application status for the `pi` example

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).

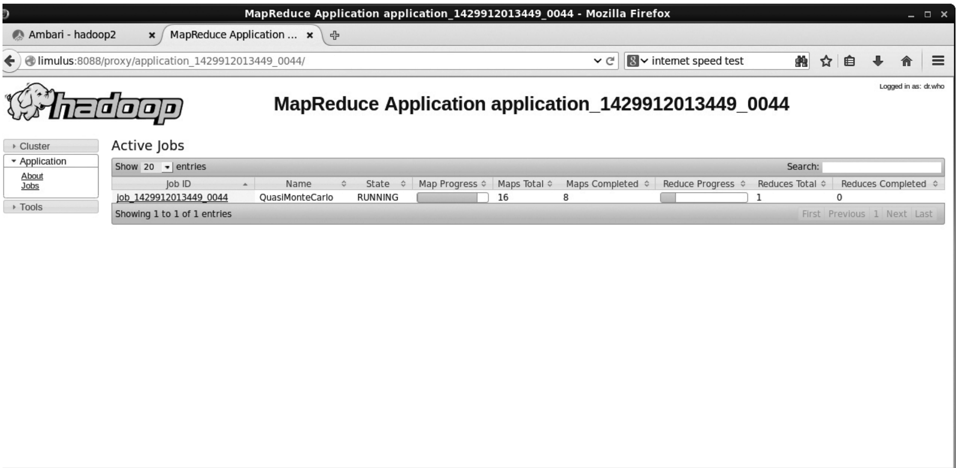


Figure 4.4 Hadoop YARN ApplicationMaster for MapReduce application

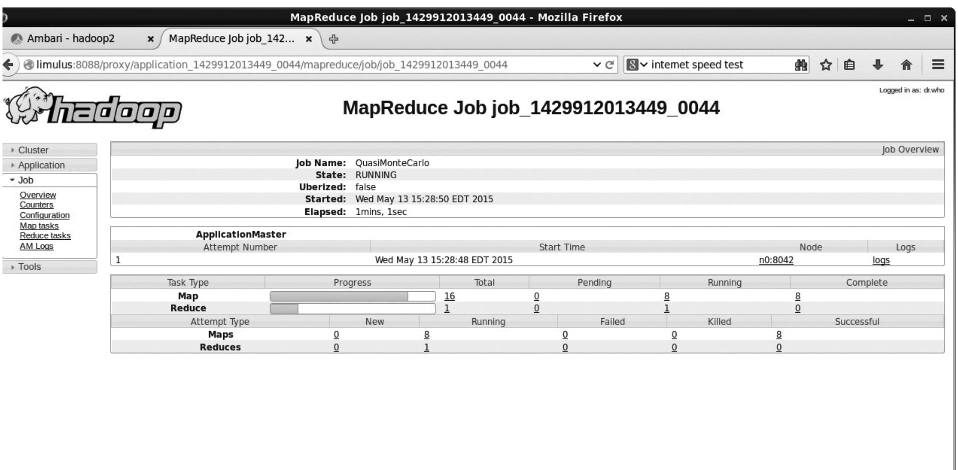


Figure 4.5 Hadoop YARN MapReduce job progress

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

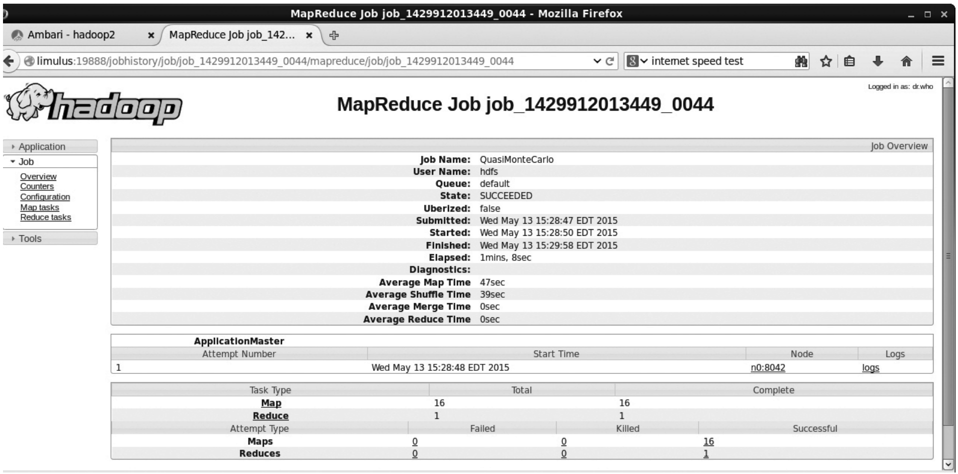


Figure 4.6 Hadoop YARN completed MapReduce job summary

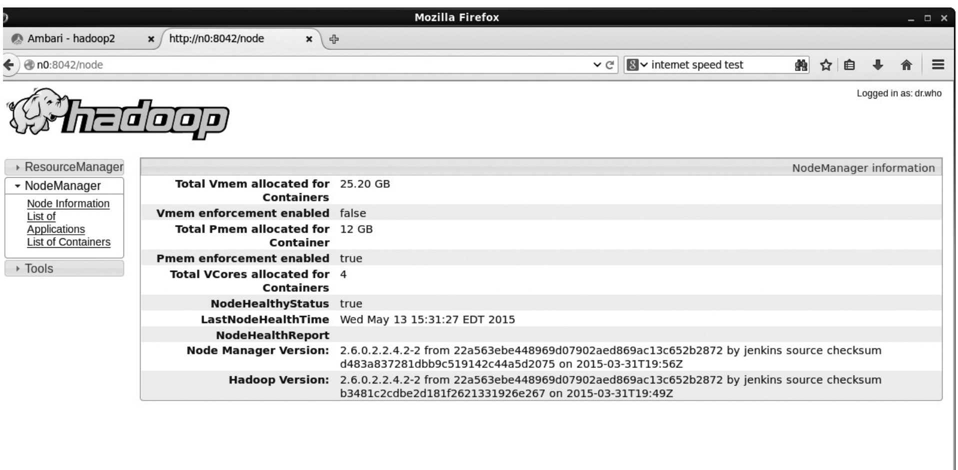


Figure 4.7 Hadoop YARN NodeManager for n0 job summary

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."

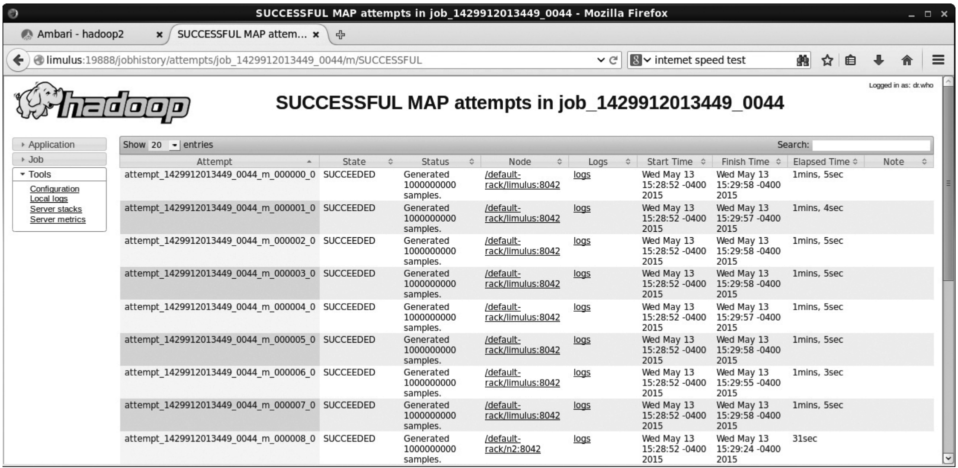


Figure 4.8 Hadoop YARN MapReduce logs available for browsing

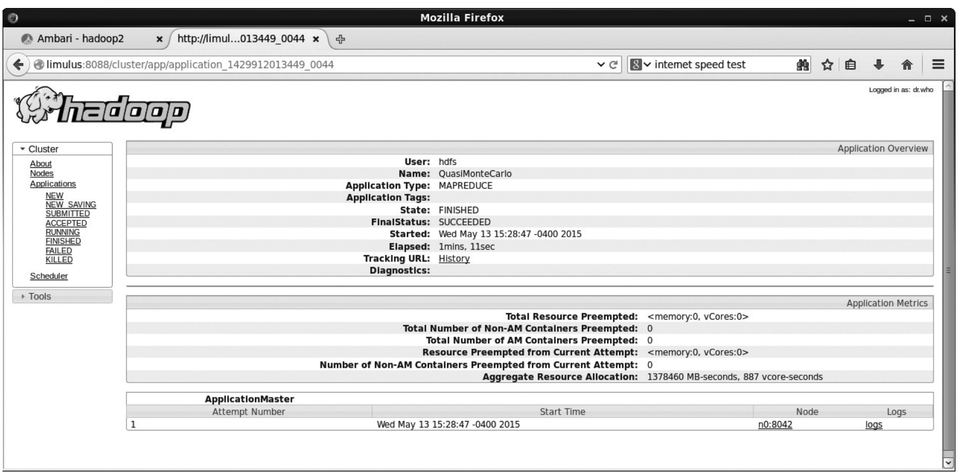


Figure 4.9 Hadoop YARN application summary page

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 *application*s 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 ResourceManager 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 500000000
➥/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

D

E

F

Z