Hadoop in Practice

SECOND EDITION

Alex Holmes

INCLUDES 104 TECHNIQUES

MANNING

Praise for the First Edition of *Hadoop in Practice*

A new book from Manning, Hadoop in Practice*, is definitely the most modern book on the topic. Important subjects, like what commercial variants such as MapR offer, and the many different releases and APIs get uniquely good coverage in this book.*

—Ted Dunning, Chief Application Architect, MapR Technologies

Comprehensive coverage of advanced Hadoop usage, including high-quality code samples.

—Chris Nauroth, Senior Staff Software Engineer
The Walt Disney Company

A very pragmatic and broad overview of Hadoop and the Hadoop tools ecosystem, with a wide set of interesting topics that tickle the creative brain.

—Mark Kemna, Chief Technology Officer, Brilig

A practical introduction to the Hadoop ecosystem.

—Philipp K. Janert, Principal Value, LLC

This book is the horizontal roof that each of the pillars of individual Hadoop technology books hold. It expertly ties together all the Hadoop ecosystem technologies.

—Ayon Sinha, Big Data Architect, Britely

I would take this book on my path to the future.

—Alexey Gayduk, Senior Software Engineer, Grid Dynamics

A high-quality and well-written book that is packed with useful examples. The breadth and detail of the material is by far superior to any other Hadoop reference guide. It is perfect for anyone who likes to learn new tools/technologies while following pragmatic, real-world examples.

—Amazon reviewer

Hadoop in Practice
Second Edition

ALEX HOLMES

MANNING
Shelter Island

For online information and ordering of this and other Manning books, please visit www.manning.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact

Special Sales Department
Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964
Email: orders@manning.com

Manning Publications Co.
20 Baldwin Road
Shelter Island, NY 11964

Development editor:	Cynthia Kane
Copyeditor:	Andy Carroll
Proofreader:	Melody Dolab
Typesetter:	Gordan Salinovic
Cover designer:	Marija Tudor

ISBN 9781617292224
Printed in the United States of America
1 2 3 4 5 6 7 8 9 10 – EBM – 19 18 17 16 15 14

brief contents

contents

preface

I first encountered Hadoop in the fall of 2008 when I was working on an internet crawl-and-analysis project at Verisign. We were making discoveries similar to those that Doug Cutting and others at Nutch had made several years earlier about how to efficiently store and manage terabytes of crawl-and-analyzed data. At the time, we were getting by with our homegrown distributed system, but the influx of a new data stream and requirements to join that stream with our crawl data couldn't be supported by our existing system in the required timeline.

After some research, we came across the Hadoop project, which seemed to be a perfect fit for our needs—it supported storing large volumes of data and provided a compute mechanism to combine them. Within a few months, we built and deployed a MapReduce application encompassing a number of MapReduce jobs, woven together with our own MapReduce workflow management system, onto a small cluster of 18 nodes. It was a revelation to observe our MapReduce jobs crunching through our data in minutes. Of course, what we weren't expecting was the amount of time that we would spend debugging and performance-tuning our MapReduce jobs. Not to mention the new roles we took on as production administrators—the biggest surprise in this role was the number of disk failures we encountered during those first few months supporting production.

As our experience and comfort level with Hadoop grew, we continued to build more of our functionality using Hadoop to help with our scaling challenges. We also started to evangelize the use of Hadoop within our organization and helped kick-start other projects that were also facing big data challenges.

The greatest challenge we faced when working with Hadoop, and specifically MapReduce, was relearning how to solve problems with it. MapReduce is its own flavor of parallel programming, and it's quite different from the in-JVM programming that we were accustomed to. The first big hurdle was training our brains to think MapReduce, a topic which the book *Hadoop in Action* by Chuck Lam (Manning Publications, 2010) covers well.

After one is used to thinking in MapReduce, the next challenge is typically related to the logistics of working with Hadoop, such as how to move data in and out of HDFS and effective and efficient ways to work with data in Hadoop. These areas of Hadoop haven't received much coverage, and that's what attracted me to the potential of this book—the chance to go beyond the fundamental word-count Hadoop uses and covering some of the trickier and dirtier aspects of Hadoop.

As I'm sure many authors have experienced, I went into this project confidently believing that writing this book was just a matter of transferring my experiences onto paper. Boy, did I get a reality check, but not altogether an unpleasant one, because writing introduced me to new approaches and tools that ultimately helped better my own Hadoop abilities. I hope that you get as much out of reading this book as I did writing it.

acknowledgments

First and foremost, I want to thank Michael Noll, who pushed me to write this book. He provided invaluable insights into how to structure the content of the book, reviewed my early chapter drafts, and helped mold the book. I can't express how much his support and encouragement has helped me throughout the process.

I'm also indebted to Cynthia Kane, my development editor at Manning, who coached me through writing this book and provided invaluable feedback on my work. Among the many notable "aha!" moments I had when working with Cynthia, the biggest one was when she steered me into using visual aids to help explain some of the complex concepts in this book.

All of the Manning staff were a pleasure to work with, and a special shout out goes to Troy Mott, Nick Chase, Tara Walsh, Bob Herbstman, Michael Stephens, Marjan Bace, Maureen Spencer, and Kevin Sullivan.

I also want to say a big thank you to all the reviewers of this book: Adam Kawa, Andrea Tarocchi, Anna Lahoud, Arthur Zubarev, Edward Ribeiro, Fillipe Massuda, Gerd Koenig, Jeet Marwah, Leon Portman, Mohamed Diouf, Muthuswamy Manigandan, Rodrigo Abreu, and Serega Sheypack. Jonathan Siedman, the primary technical reviewer, did a great job of reviewing the entire book.

Many thanks to Josh Wills, the creator of Crunch, who kindly looked over the chapter that covered that topic. And more thanks go to Josh Patterson, who reviewed my Mahout chapter.

Finally, a special thanks to my wife, Michal, who had to put up with a cranky husband working crazy hours. She was a source of encouragement throughout the entire process.

about this book

Doug Cutting, the creator of Hadoop, likes to call Hadoop the kernel for big data, and I would tend to agree. With its distributed storage and compute capabilities, Hadoop is fundamentally an enabling technology for working with huge datasets. Hadoop provides a bridge between structured (RDBMS) and unstructured (log files, XML, text) data and allows these datasets to be easily joined together. This has evolved from traditional use cases, such as combining OLTP and log files, to more sophisticated uses, such as using Hadoop for data warehousing (exemplified by Facebook) and the field of data science, which studies and makes new discoveries about data.

This book collects a number of intermediary and advanced Hadoop examples and presents them in a problem/solution format. Each technique addresses a specific task you'll face, like using Flume to move log files into Hadoop or using Mahout for predictive analysis. Each problem is explored step by step, and as you work through them, you'll find yourself growing more comfortable with Hadoop and at home in the world of big data.

This hands-on book targets users who have some practical experience with Hadoop and understand the basic concepts of MapReduce and HDFS. Manning's *Hadoop in Action* by Chuck Lam contains the necessary prerequisites to understand and apply the techniques covered in this book.

Many techniques in this book are Java-based, which means readers are expected to possess an intermediate-level knowledge of Java. An excellent text for all levels of Java users is *Effective Java,* Second Edition by Joshua Bloch (Addison-Wesley, 2008).

Roadmap

This book has 10 chapters divided into four parts.

Part 1 contains two chapters that form the introduction to this book. They review Hadoop basics and look at how to get Hadoop up and running on a single host. YARN, which is new in Hadoop version 2, is also examined, and some operational tips are provided for performing basic functions in YARN.

Part 2, "Data logistics," consists of three chapters that cover the techniques and tools required to deal with data fundamentals, how to work with various data formats, how to organize and optimize your data, and getting data into and out of Hadoop. Picking the right format for your data and determining how to organize data in HDFS are the first items you'll need to address when working with Hadoop, and they're covered in chapters 3 and 4 respectively. Getting data into Hadoop is one of the bigger hurdles commonly encountered when working with Hadoop, and chapter 5 is dedicated to looking at a variety of tools that work with common enterprise data sources.

Part 3 is called "Big data patterns," and it looks at techniques to help you work effectively with large volumes of data. Chapter 6 covers how to represent data such as graphs for use with MapReduce, and it looks at several algorithms that operate on graph data. Chapter 7 looks at more advanced data structures and algorithms such as graph processing and using HyperLogLog for working with large datasets. Chapter 8 looks at how to tune, debug, and test MapReduce performance issues, and it also covers a number of techniques to help make your jobs run faster.

Part 4 is titled "Beyond MapReduce," and it examines a number of technologies that make it easier to work with Hadoop. Chapter 9 covers the most prevalent and promising SQL technologies for data processing on Hadoop, and Hive, Impala, and Spark SQL are examined. The final chapter looks at how to write your own YARN application, and it provides some insights into some of the more advanced features you can use in your applications.

The appendix covers instructions for the source code that accompanies this book, as well as installation instructions for Hadoop and all the other related technologies covered in the book.

Finally, there are two bonus chapters available from the publisher's website at www.manning.com/HadoopinPracticeSecondEdition: chapter 11 "Integrating R and Hadoop for statistics and more" and chapter 12 "Predictive analytics with Mahout."

What's new in the second edition?

This second edition covers Hadoop 2, which at the time of writing is the current production-ready version of Hadoop. The first edition of the book covered Hadoop 0.22 (Hadoop 1 wasn't yet out), and Hadoop 2 has turned the world upside-down and opened up the Hadoop platform to processing paradigms beyond MapReduce. YARN, the new scheduler and application manager in Hadoop 2, is complex and new to the community, which prompted me to dedicate a new chapter 2 to covering YARN basics and to discussing how MapReduce now functions as a YARN application.

Parquet has also recently emerged as a new way to store data in HDFS—its columnar format can yield both space and time efficiencies in your data pipelines, and it's quickly becoming the ubiquitous way to store data. Chapter 4 includes extensive coverage of Parquet, which includes how Parquet supports sophisticated object models such as Avro and how various Hadoop tools can use Parquet.

How data is being ingested into Hadoop has also evolved since the first edition, and Kafka has emerged as the new data pipeline, which serves as the transport tier between your data producers and data consumers, where a consumer would be a system such as Camus that can pull data from Kafka into HDFS. Chapter 5, which covers moving data into and out of Hadoop, now includes coverage of Kafka and Camus.

There are many new technologies that YARN now can support side by side in the same cluster, and some of the more exciting and promising technologies are covered in the new part 4, titled "Beyond MapReduce," where I cover some compelling new SQL technologies such as Impala and Spark SQL. The last chapter, also new for this edition, looks at how you can write your own YARN application, and it's packed with information about important features to support your YARN application.

Getting help

You'll no doubt have many questions when working with Hadoop. Luckily, between the wikis and a vibrant user community, your needs should be well covered:

- The main wiki is located at http://wiki.apache.org/hadoop/, and it contains useful presentations, setup instructions, and troubleshooting instructions.
- The Hadoop Common, HDFS, and MapReduce mailing lists can all be found at http://hadoop.apache.org/mailing_lists.html.
- "Search Hadoop" is a useful website that indexes all of Hadoop and its ecosystem projects, and it provides full-text search capabilities: http://search-hadoop.com/.
- You'll find many useful blogs you should subscribe to in order to keep on top of current events in Hadoop. This preface includes a selection of my favorites:
  - Cloudera and Hortonworks are both prolific writers of practical applications on Hadoop—reading their blogs is always educational: http://www.cloudera.com/blog/ and http://hortonworks.com/blog/.
  - Michael Noll is one of the first bloggers to provide detailed setup instructions for Hadoop, and he continues to write about real-life challenges: www.michael-noll.com/.
  - There's a plethora of active Hadoop Twitter users that you may want to follow, including Arun Murthy (@acmurthy), Tom White (@tom_e_white), Eric Sammer (@esammer), Doug Cutting (@cutting), and Todd Lipcon (@tlipcon). The Hadoop project tweets on @hadoop.

Code conventions and downloads

All source code in listings or in text is presented in a `fixed-width font like this` to separate it from ordinary text. Code annotations accompany many of the listings, highlighting important concepts.

All of the text and examples in this book work with Hadoop 2.x, and most of the MapReduce code is written using the newer `org.apache.hadoop.mapreduce` MapReduce APIs. The few examples that use the older `org.apache.hadoop.mapred` package are usually the result of working with a third-party library or a utility that only works with the old API.

All of the code used in this book is available on GitHub at https://github.com/alexholmes/hiped2 and also from the publisher's website at www.manning.com/HadoopinPracticeSecondEdition. The first section in the appendix shows you how to download, install, and get up and running with the code.

Third-party libraries

I use a number of third-party libraries for convenience purposes. They're included in the Maven-built JAR, so there's no extra work required to work with these libraries.

Datasets

Throughout this book, you'll work with three datasets to provide some variety in the examples. All the datasets are small to make them easy to work with. Copies of the exact data used are available in the GitHub repository in the https://github.com/alexholmes/hiped2/tree/master/test-data directory. I also sometimes use data that's specific to a chapter, and it's available within chapter-specific subdirectories under the same GitHub location.

NASDAQ financial stocks

I downloaded the NASDAQ daily exchange data from InfoChimps (www.infochimps.com). I filtered this huge dataset down to just five stocks and their start-of-year values from 2000 through 2009. The data used for this book is available on GitHub at https://github.com/alexholmes/hiped2/blob/master/test-data/stocks.txt.

The data is in CSV form, and the fields are in the following order:

```
Symbol,Date,Open,High,Low,Close,Volume,Adj Close
```

Apache log data

I created a sample log file in Apache Common Log Format[1] with some fake Class E IP addresses and some dummy resources and response codes. The file is available on GitHub at https://github.com/alexholmes/hiped2/blob/master/test-data/apachelog.txt.

[1] See http://httpd.apache.org/docs/1.3/logs.html#common.

Names

Names were retrieved from the U.S. government census at www.census.gov/genealogy/www/data/1990surnames/dist.all.last, and this data is available at https://github.com/alexholmes/hiped2/blob/master/test-data/names.txt.

Author Online

Purchase of *Hadoop in Practice, Second Edition* includes free access to a private web forum run by Manning Publications where you can make comments about the book, ask technical questions, and receive help from the authors and from other users. To access the forum and subscribe to it, point your web browser to www.manning.com/HadoopinPractice, SecondEdition. This page provides information on how to get on the forum once you are registered, what kind of help is available, and the rules of conduct on the forum. It also provides links to the source code for the examples in the book, errata, and other downloads.

Manning's commitment to our readers is to provide a venue where a meaningful dialog between individual readers and between readers and the author can take place. It is not a commitment to any specific amount of participation on the part of the author, whose contribution to the Author Online forum remains voluntary (and unpaid). We suggest you try asking the author challenging questions lest his interest strays!

The Author Online forum and the archives of previous discussions will be accessible from the publisher's website as long as the book is in print.

about the cover illustration

The figure on the cover of *Hadoop in Practice, Second Edition* is captioned "Momak from Kistanja, Dalmatia." The illustration is taken from a reproduction of an album of traditional Croatian costumes from the mid-nineteenth century by Nikola Arsenovic, published by the Ethnographic Museum in Split, Croatia, in 2003. The illustrations were obtained from a helpful librarian at the Ethnographic Museum in Split, itself situated in the Roman core of the medieval center of the town: the ruins of Emperor Diocletian's retirement palace from around AD 304. The book includes finely colored illustrations of figures from different regions of Croatia, accompanied by descriptions of the costumes and of everyday life.

Kistanja is a small town located in Bukovica, a geographical region in Croatia. It is situated in northern Dalmatia, an area rich in Roman and Venetian history. The word "momak" in Croatian means a bachelor, beau, or suitor—a single young man who is of courting age—and the young man on the cover, looking dapper in a crisp, white linen shirt and a colorful, embroidered vest, is clearly dressed in his finest clothes, which would be worn to church and for festive occasions—or to go calling on a young lady.

Dress codes and lifestyles have changed over the last 200 years, and the diversity by region, so rich at the time, has faded away. It is now hard to tell apart the inhabitants of different continents, let alone of different hamlets or towns separated by only a few miles. Perhaps we have traded cultural diversity for a more varied personal life—certainly for a more varied and fast-paced technological life.

Manning celebrates the inventiveness and initiative of the computer business with book covers based on the rich diversity of regional life of two centuries ago, brought back to life by illustrations from old books and collections like this one.

Part 1

Background and fundamentals

Part 1 of this book consists of chapters 1 and 2, which cover the important Hadoop fundamentals.

Chapter 1 covers Hadoop's components and its ecosystem and provides instructions for installing a pseudo-distributed Hadoop setup on a single host, along with a system that will enable you to run all of the examples in the book. Chapter 1 also covers the basics of Hadoop configuration, and walks you through how to write and run a MapReduce job on your new setup.

Chapter 2 introduces YARN, which is a new and exciting development in Hadoop version 2, transitioning Hadoop from being a MapReduce-only system to one that can support many execution engines. Given that YARN is new to the community, the goal of this chapter is to look at some basics such as its components, how configuration works, and also how MapReduce works as a YARN application. Chapter 2 also provides an overview of some applications that YARN has enabled to execute on Hadoop, such as Spark and Storm.

1 Hadoop in a heartbeat

This chapter covers

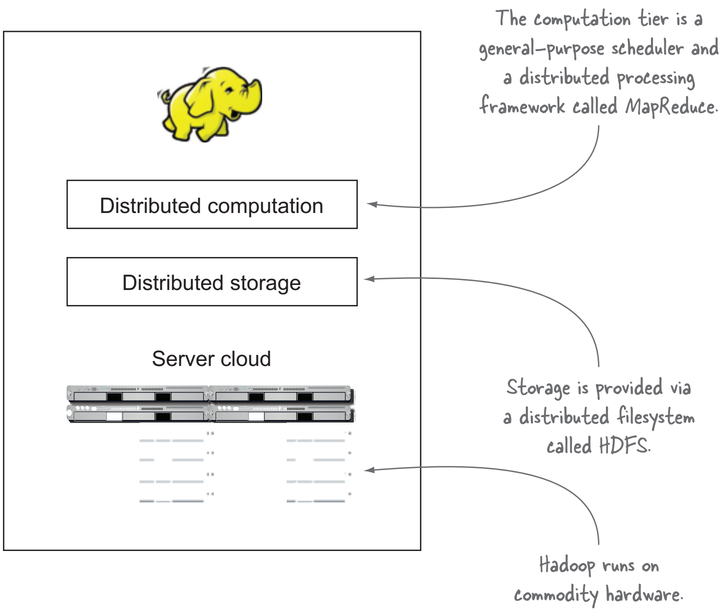
- Examining how the core Hadoop system works
- Understanding the Hadoop ecosystem
- Running a MapReduce job

We live in the age of big data, where the data volumes we need to work with on a day-to-day basis have outgrown the storage and processing capabilities of a single host. Big data brings with it two fundamental challenges: how to store and work with voluminous data sizes, and more important, how to understand data and turn it into a competitive advantage.

Hadoop fills a gap in the market by effectively storing and providing computational capabilities for substantial amounts of data. It's a distributed system made up of a distributed filesystem, and it offers a way to parallelize and execute programs on a cluster of machines (see figure 1.1). You've most likely come across Hadoop because it's been adopted by technology giants like Yahoo!, Facebook, and Twitter to address their big data needs, and it's making inroads across all industrial sectors.

Because you've come to this book to get some practical experience with Hadoop and Java,[1] I'll start with a brief overview and then show you how to install

[1] To benefit from this book, you should have some practical experience with Hadoop and understand the basic concepts of MapReduce and HDFS (covered in Manning's *Hadoop in Action* by Chuck Lam, 2010). Further, you should have an intermediate-level knowledge of Java—*Effective Java,* 2nd Edition by Joshua Bloch (Addison-Wesley, 2008) is an excellent resource on this topic.



The computation tier is a general-purpose scheduler and a distributed processing framework called MapReduce.

Storage is provided via a distributed filesystem called HDFS.

Hadoop runs on commodity hardware.

Figure 1.1 The Hadoop environment is a distributed system that runs on commodity hardware.

Hadoop and run a MapReduce job. By the end of this chapter, you'll have had a basic refresher on the nuts and bolts of Hadoop, which will allow you to move on to the more challenging aspects of working with it.

Let's get started with a detailed overview.

1.1 What is Hadoop?

Hadoop is a platform that provides both distributed storage and computational capabilities. Hadoop was first conceived to fix a scalability issue that existed in Nutch,[2] an open source crawler and search engine. At the time, Google had published papers that described its novel distributed filesystem, the Google File System (GFS), and MapReduce, a computational framework for parallel processing. The successful implementation of these papers' concepts in Nutch resulted in it being split into two separate projects, the second of which became Hadoop, a first-class Apache project.

In this section we'll look at Hadoop from an architectural perspective, examine how industry uses it, and consider some of its weaknesses. Once we've covered this background, we'll look at how to install Hadoop and run a MapReduce job.

Hadoop proper, as shown in figure 1.2, is a distributed master-slave architecture[3] that consists of the following primary components:

[2] The Nutch project, and by extension Hadoop, was led by Doug Cutting and Mike Cafarella.

[3] A model of communication where one process, called the *master*, has control over one or more other processes, called *slaves*.

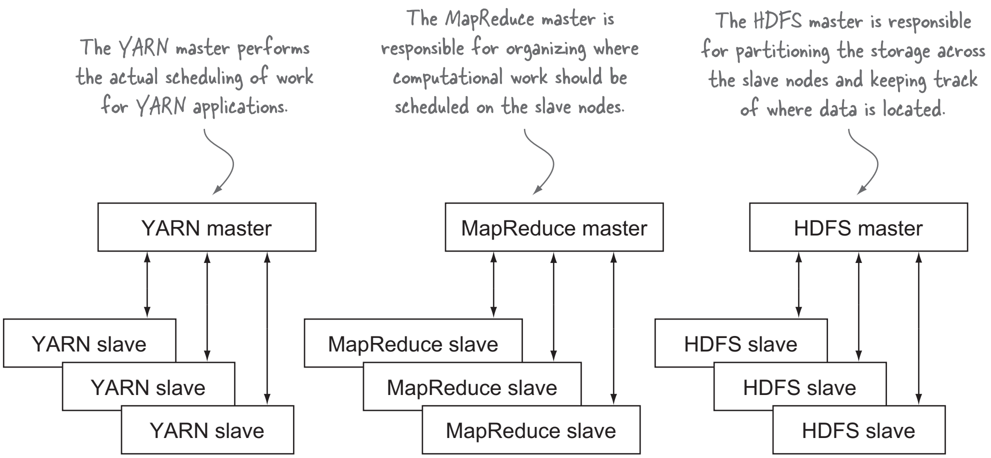

The YARN master performs the actual scheduling of work for YARN applications.
The MapReduce master is responsible for organizing where computational work should be scheduled on the slave nodes.
The HDFS master is responsible for partitioning the storage across the slave nodes and keeping track of where data is located.
YARN master
MapReduce master
HDFS master
YARN slave
YARN slave
YARN slave
MapReduce slave
MapReduce slave
MapReduce slave
HDFS slave
HDFS slave
HDFS slave

Figure 1.2 High-level Hadoop 2 master-slave architecture

- Hadoop Distributed File System (HDFS) for data storage.
- Yet Another Resource Negotiator (YARN), introduced in Hadoop 2, a general-purpose scheduler and resource manager. Any YARN application can run on a Hadoop cluster.
- MapReduce, a batch-based computational engine. In Hadoop 2, MapReduce is implemented as a YARN application.

Traits intrinsic to Hadoop are data partitioning and parallel computation of large datasets. Its storage and computational capabilities scale with the addition of hosts to a Hadoop cluster; clusters with hundreds of hosts can easily reach data volumes in the petabytes.

In the first step in this section, we'll examine the HDFS, YARN, and MapReduce architectures.

1.1.1 Core Hadoop components

To understand Hadoop's architecture we'll start by looking at the basics of HDFS.

HDFS

HDFS is the storage component of Hadoop. It's a distributed filesystem that's modeled after the Google File System (GFS) paper.[4] HDFS is optimized for high throughput and works best when reading and writing large files (gigabytes and larger). To support this throughput, HDFS uses unusually large (for a filesystem) block sizes and data locality optimizations to reduce network input/output (I/O).

Scalability and availability are also key traits of HDFS, achieved in part due to data replication and fault tolerance. HDFS replicates files for a configured number of times, is tolerant of both software and hardware failure, and automatically re-replicates data blocks on nodes that have failed.

[4] See "The Google File System," http://research.google.com/archive/gfs.html.

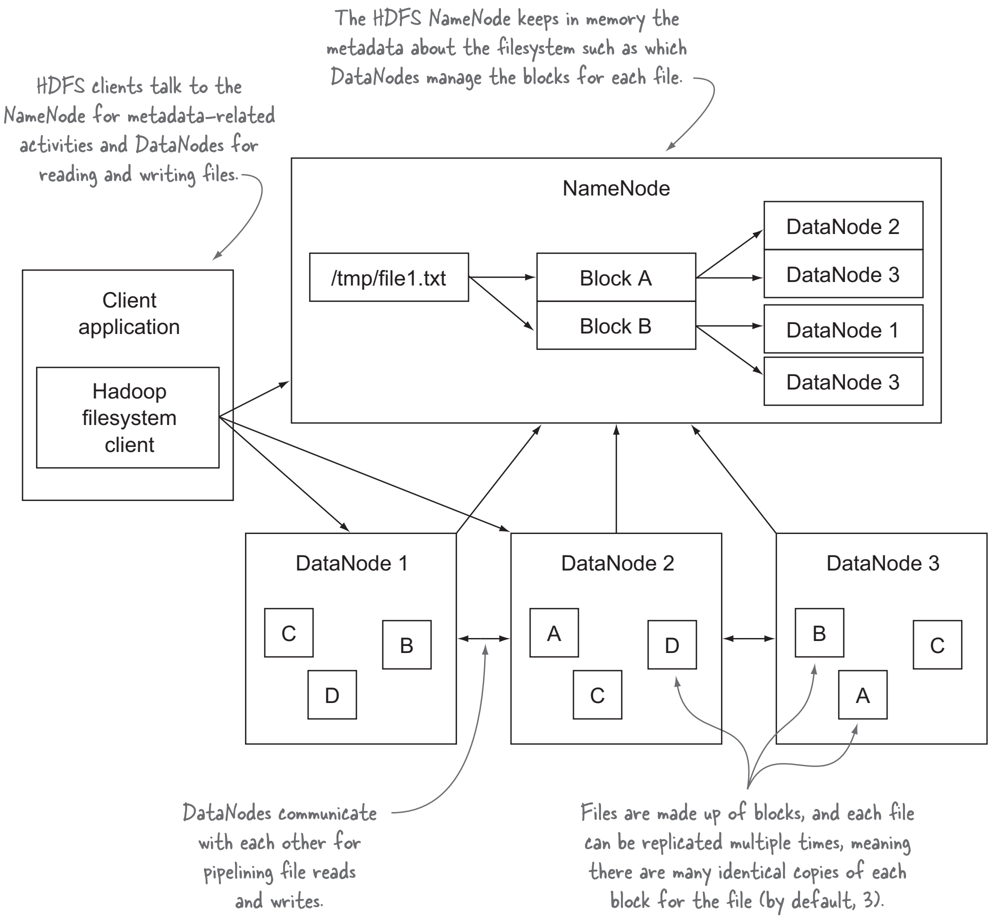

The HDFS NameNode keeps in memory the metadata about the filesystem such as which DataNodes manage the blocks for each file.
HDFS clients talk to the NameNode for metadata-related activities and DataNodes for reading and writing files.
NameNode
/tmp/file1.txt
Block A
Block B
DataNode 2
DataNode 3
DataNode 1
DataNode 3
Client application
Hadoop filesystem client
DataNode 1
C
B
D
DataNode 2
A
D
C
DataNode 3
B
C
A
DataNodes communicate with each other for pipelining file reads and writes.
Files are made up of blocks, and each file can be replicated multiple times, meaning there are many identical copies of each block for the file (by default, 3).

Figure 1.3 An HDFS client communicating with the master NameNode and slave DataNodes

Figure 1.3 shows a logical representation of the components in HDFS: the NameNode and the DataNode. It also shows an application that's using the Hadoop filesystem library to access HDFS.

Hadoop 2 introduced two significant new features for HDFS—Federation and High Availability (HA):

- Federation allows HDFS metadata to be shared across multiple NameNode hosts, which aides with HDFS scalability and also provides data isolation, allowing different applications or teams to run their own NameNodes without fear of impacting other NameNodes on the same cluster.
- High Availability in HDFS removes the single point of failure that existed in Hadoop 1, wherein a NameNode disaster would result in a cluster outage. HDFS HA also offers the ability for failover (the process by which a standby NameNode takes over work from a failed primary NameNode) to be automated.

Now that you have a bit of HDFS knowledge, it's time to look at YARN, Hadoop's scheduler.

YARN

YARN is Hadoop's distributed resource scheduler. YARN is new to Hadoop version 2 and was created to address challenges with the Hadoop 1 architecture:

- Deployments larger than 4,000 nodes encountered scalability issues, and adding additional nodes didn't yield the expected linear scalability improvements.
- Only MapReduce workloads were supported, which meant it wasn't suited to run execution models such as machine learning algorithms that often require iterative computations.

For Hadoop 2 these problems were solved by extracting the scheduling function from MapReduce and reworking it into a generic application scheduler, called YARN. With this change, Hadoop clusters are no longer limited to running MapReduce workloads; YARN enables a new set of workloads to be natively supported on Hadoop, and it allows alternative processing models, such as graph processing and stream processing, to coexist with MapReduce. Chapters 2 and 10 cover YARN and how to write YARN applications.

YARN's architecture is simple because its primary role is to schedule and manage resources in a Hadoop cluster. Figure 1.4 shows a logical representation of the core components in YARN: the ResourceManager and the NodeManager. Also shown are the components specific to YARN applications, namely, the YARN application client, the ApplicationMaster, and the container.

To fully realize the dream of a generalized distributed platform, Hadoop 2 introduced another change—the ability to allocate containers in various configurations.

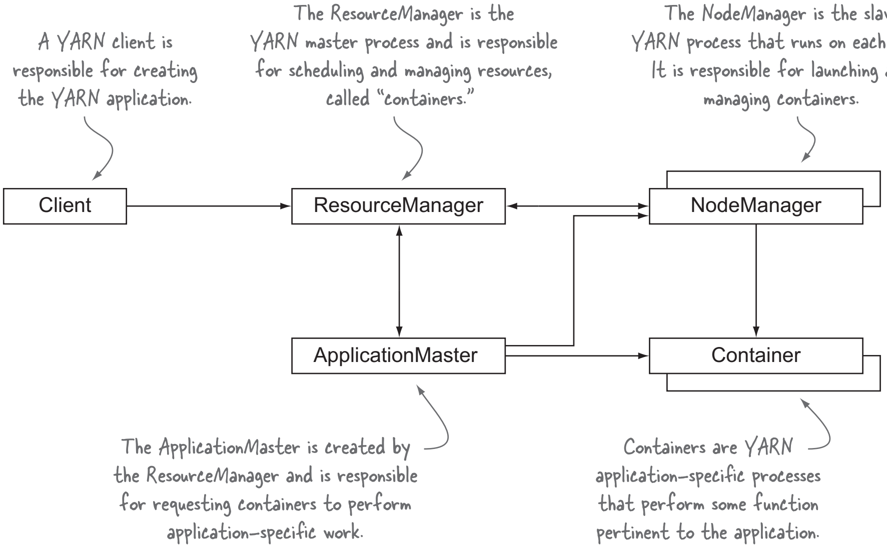

A YARN client is responsible for creating the YARN application.
The ResourceManager is the YARN master process and is responsible for scheduling and managing resources, called "containers."
The NodeManager is the slave YARN process that runs on each node. It is responsible for launching and managing containers.
Client
ResourceManager
NodeManager
ApplicationMaster
Container
The ApplicationMaster is created by the ResourceManager and is responsible for requesting containers to perform application-specific work.
Containers are YARN application-specific processes that perform some function pertinent to the application.

Figure 1.4 The logical YARN architecture showing typical communication between the core YARN components and YARN application components

Hadoop 1 had the notion of "slots," which were a fixed number of map and reduce processes that were allowed to run on a single node. This was wasteful in terms of cluster utilization and resulted in underutilized resources during MapReduce operations, and it also imposed memory limits for map and reduce tasks. With YARN, each container requested by an ApplicationMaster can have disparate memory and CPU traits, and this gives YARN applications full control over the resources they need to fulfill their work.

You'll work with YARN in more detail in chapters 2 and 10, where you'll learn how YARN works and how to write a YARN application. Next up is an examination of MapReduce, Hadoop's computation engine.

MAPREDUCE

MapReduce is a batch-based, distributed computing framework modeled after Google's paper on MapReduce.[5] It allows you to parallelize work over a large amount of raw data, such as combining web logs with relational data from an OLTP database to model how users interact with your website. This type of work, which could take days or longer using conventional serial programming techniques, can be reduced to minutes using MapReduce on a Hadoop cluster.

The MapReduce model simplifies parallel processing by abstracting away the complexities involved in working with distributed systems, such as computational parallelization, work distribution, and dealing with unreliable hardware and software. With this abstraction, MapReduce allows the programmer to focus on addressing business needs rather than getting tangled up in distributed system complications.

MapReduce decomposes work submitted by a client into small parallelized map and reduce tasks, as shown in figure 1.5. The map and reduce constructs used in

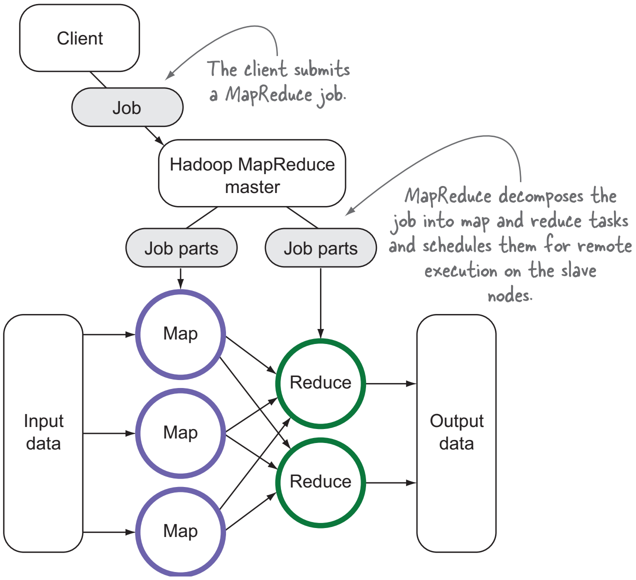
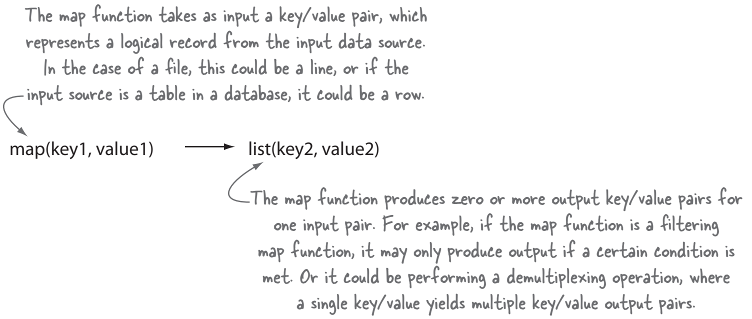

Client
The client submits a MapReduce job.
Job
Hadoop MapReduce master
MapReduce decomposes the job into map and reduce tasks and schedules them for remote execution on the slave nodes.
Job parts
Job parts
Input data
Map
Map
Map
Reduce
Reduce
Output data

Figure 1.5 A client submitting a job to MapReduce, breaking the work into small map and reduce tasks

[5] See "MapReduce: Simplified Data Processing on Large Clusters," http://research.google.com/archive/mapreduce.html.



The map function takes as input a key/value pair, which represents a logical record from the input data source. In the case of a file, this could be a line, or if the input source is a table in a database, it could be a row.

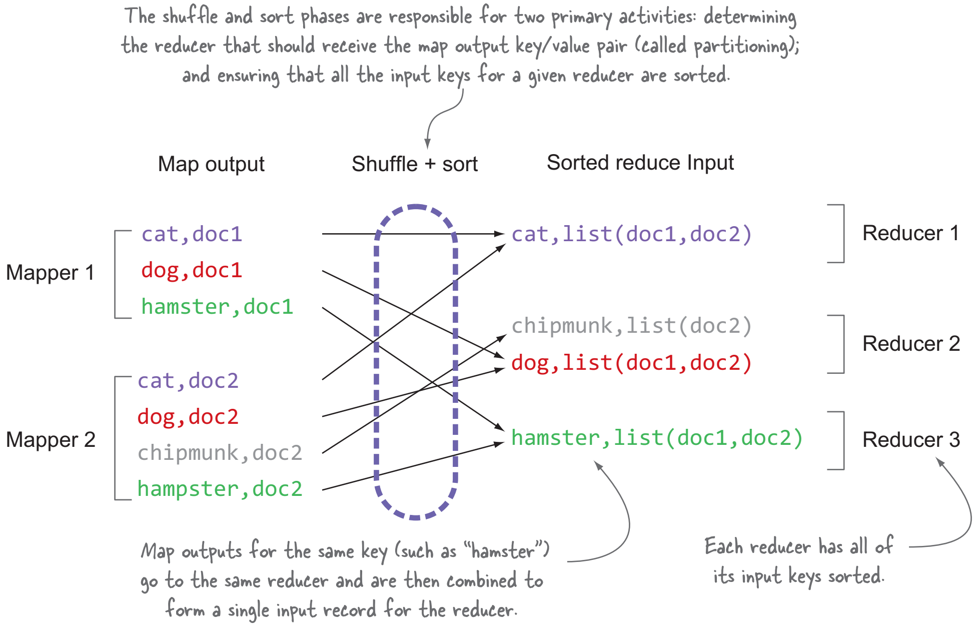
The map function produces zero or more output key/value pairs for one input pair. For example, if the map function is a filtering map function, it may only produce output if a certain condition is met. Or it could be performing a demultiplexing operation, where a single key/value yields multiple key/value output pairs.

Figure 1.6 A logical view of the map function that takes a key/value pair as input

MapReduce are borrowed from those found in the Lisp functional programming language, and they use a shared-nothing model to remove any parallel execution interdependencies that could add unwanted synchronization points or state sharing.[6]

The role of the programmer is to define map and reduce functions where the map function outputs key/value tuples, which are processed by reduce functions to produce the final output. Figure 1.6 shows a pseudocode definition of a map function with regard to its input and output.

The power of MapReduce occurs between the map output and the reduce input in the shuffle and sort phases, as shown in figure 1.7.



The shuffle and sort phases are responsible for two primary activities: determining the reducer that should receive the map output key/value pair (called partitioning); and ensuring that all the input keys for a given reducer are sorted.

Map outputs for the same key (such as "hamster") go to the same reducer and are then combined to form a single input record for the reducer.

Each reducer has all of its input keys sorted.

Figure 1.7 MapReduce's shuffle and sort phases

[6] A shared-nothing architecture is a distributed computing concept that represents the notion that each node is independent and self-sufficient.