

Hadoop操作手册 (影印版)



Hadoop Operations

O'REILLY®

東南大學出版社

Eric Sammer 著

Hadoop操作手册 (影印版)

Hadoop Operations

Eric Sammer 著

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc. 授权东南大学出版社出版

南京 东南大学出版社

图书在版编目 (CIP) 数据

Hadoop 操作手册: 英文/(美)萨默尔 (Sammer, E.)
著. —影印本. —南京: 东南大学出版社, 2013.6
书名原文: Hadoop Operations
ISBN 978-7-5641-4258-2

I. ① H… II. ① 萨… III. ① 数据处理软件—技术手册—英文 IV. ① TP274.62

中国版本图书馆 CIP 数据核字 (2013) 第 104947 号

江苏省版权局著作权合同登记

图字: 10-2013-118 号

©2012 by O'Reilly Media, Inc.

Reprint of the English Edition, jointly published by O'Reilly Media, Inc. and Southeast University Press, 2013. Authorized reprint of the original English edition, 2013 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

英文原版由 O'Reilly Media, Inc. 出版 2012。

英文影印版由东南大学出版社出版 2013。此影印版的出版和销售得到出版权和销售权的所有者——O'Reilly Media, Inc. 的许可。

版权所有, 未得书面许可, 本书的任何部分和全部不得以任何形式重制。

Hadoop 操作手册 (影印版)

出版发行: 东南大学出版社
地 址: 南京四牌楼 2 号 邮编: 210096
出 版 人: 江建中
网 址: <http://www.seupress.com>
电子邮件: press@seupress.com
印 刷: 扬中市印刷有限公司
开 本: 787 毫米 × 980 毫米 16 开本
印 张: 18.75
字 数: 367 千字
版 次: 2013 年 6 月第 1 版
印 次: 2013 年 6 月第 1 次印刷
书 号: ISBN 978-7-5641-4258-2
定 价: 59.00 元 (册)

本社图书若有印装质量问题, 请直接与营销部联系。电话 (传真): 025-83791830

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

Constant width bold

Shows commands or other text that should be typed literally by the user.

Constant width italic

Shows text that should be replaced with user-supplied values or by values determined by context.



This icon signifies a tip, suggestion, or general note.



This icon indicates a warning or caution.

Using Code Examples


This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does

require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: “*Hadoop Operations* by Eric Sammer (O'Reilly). Copyright 2012 Eric Sammer, 978-1-449-32705-7.”

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.

Safari® Books Online

Safari  Safari Books Online (www.safaribooksonline.com) is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at http://oreil.ly/hadoop_operations.

To comment or ask technical questions about this book, send email to bookquestions@oreilly.com.

For more information about our books, courses, conferences, and news, see our website at <http://www.oreilly.com>.

Find us on Facebook: <http://facebook.com/oreilly>

Follow us on Twitter: <http://twitter.com/oreillymedia>

Watch us on YouTube: <http://www.youtube.com/oreillymedia>

Acknowledgments

I want to thank Aida Escrive-Sammer, my wife, best friend, and favorite sysadmin, for putting up with me while I wrote this.

None of this was possible without the support and hard work of the larger Apache Hadoop community and ecosystem projects. I want to encourage all readers to get involved in the community and open source in general.

Matt Massie gave me the opportunity to do this, along with O'Reilly, and then cheered me on the whole way. Both Matt and Tom White coached me through the proposal process. Mike Olson, Omer Trajman, Amr Awadallah, Peter Cooper-Ellis, Angus Klein, and the rest of the Cloudera management team made sure I had the time, resources, and encouragement to get this done. Aparna Ramani, Rob Weltman, Jolly Chen, and Helen Friedland were instrumental throughout this process and forgiving of my constant interruptions of their teams. Special thanks to Christophe Bisciglia for giving me an opportunity at Cloudera and for the advice along the way.

Many people provided valuable feedback and input throughout the entire process, but especially Aida Escrive-Sammer, Tom White, Alejandro Abdelnur, Amina Abdulla, Patrick Angeles, Paul Battaglia, Will Chase, Yanpei Chen, Eli Collins, Joe Crobak, Doug Cutting, Joey Echeverria, Sameer Farooqui, Andrew Ferguson, Brad Hedlund, Linden Hillenbrand, Patrick Hunt, Matt Jacobs, Amandeep Khurana, Aaron Kimball, Hal Lee, Justin Lintz, Todd Lipcon, Cameron Martin, Chad Metcalf, Meg McRoberts, Aaron T. Myers, Kay Ousterhout, Greg Rahn, Henry Robinson, Mark Roddy, Jonathan Seidman, Ed Sexton, Loren Siebert, Sunil Sitaula, Ben Spivey, Dan Spiewak, Omer Trajman, Kathleen Ting, Erik-Jan van Baaren, Vinithra Varadharajan, Patrick Wendell, Tom Wheeler, Ian Wrigley, Nezih Yigitbasi, and Philip Zeyliger. To those whom I may have omitted from this list, please forgive me.

The folks at O'Reilly have been amazing, especially Courtney Nash, Mike Loukides, Maria Stallone, Arlette Labat, and Meghan Blanchette.

Jaime Caban, Victor Nee, Travis Melo, Andrew Bayer, Liz Pennell, and Michael Demetria provided additional administrative, technical, and contract support.

Finally, a special thank you to Kathy Sammer for her unwavering support, and for teaching me to do *exactly* what others say you cannot.

Portions of this book have been reproduced or derived from software and documentation available under the Apache Software License, version 2 (<http://www.apache.org/licenses/LICENSE-2.0>).

Table of Contents

Preface	ix
1. Introduction	1
2. HDFS	7
Goals and Motivation	7
Design	8
Daemons	9
Reading and Writing Data	11
The Read Path	12
The Write Path	13
Managing Filesystem Metadata	14
Namenode High Availability	16
Namenode Federation	18
Access and Integration	20
Command-Line Tools	20
FUSE	23
REST Support	23
3. MapReduce	25
The Stages of MapReduce	26
Introducing Hadoop MapReduce	33
Daemons	34
When It All Goes Wrong	36
YARN	37
4. Planning a Hadoop Cluster	41
Picking a Distribution and Version of Hadoop	41
Apache Hadoop	41
Cloudera's Distribution Including Apache Hadoop	42
Versions and Features	42

What Should I Use?	44
Hardware Selection	45
Master Hardware Selection	46
Worker Hardware Selection	48
Cluster Sizing	50
Blades, SANs, and Virtualization	52
Operating System Selection and Preparation	54
Deployment Layout	54
Software	56
Hostnames, DNS, and Identification	57
Users, Groups, and Privileges	60
Kernel Tuning	62
vm.swappiness	62
vm.overcommit_memory	62
Disk Configuration	63
Choosing a Filesystem	64
Mount Options	66
Network Design	66
Network Usage in Hadoop: A Review	67
1 Gb versus 10 Gb Networks	69
Typical Network Topologies	69
5. Installation and Configuration	75
Installing Hadoop	75
Apache Hadoop	76
CDH	80
Configuration: An Overview	84
The Hadoop XML Configuration Files	87
Environment Variables and Shell Scripts	88
Logging Configuration	90
HDFS	93
Identification and Location	93
Optimization and Tuning	95
Formatting the Namenode	99
Creating a /tmp Directory	100
Namenode High Availability	100
Fencing Options	102
Basic Configuration	104
Automatic Failover Configuration	105
Format and Bootstrap the Namenodes	108
Namenode Federation	113
MapReduce	120
Identification and Location	120

Optimization and Tuning	122
Rack Topology	130
Security	133
6. Identity, Authentication, and Authorization	135
Identity	137
Kerberos and Hadoop	137
Kerberos: A Refresher	138
Kerberos Support in Hadoop	140
Authorization	153
HDFS	153
MapReduce	155
Other Tools and Systems	159
Tying It Together	164
7. Resource Management	167
What Is Resource Management?	167
HDFS Quotas	168
MapReduce Schedulers	170
The FIFO Scheduler	171
The Fair Scheduler	173
The Capacity Scheduler	185
The Future	193
8. Cluster Maintenance	195
Managing Hadoop Processes	195
Starting and Stopping Processes with Init Scripts	195
Starting and Stopping Processes Manually	196
HDFS Maintenance Tasks	196
Adding a Datanode	196
Decommissioning a Datanode	197
Checking Filesystem Integrity with fsck	198
Balancing HDFS Block Data	202
Dealing with a Failed Disk	204
MapReduce Maintenance Tasks	205
Adding a Tasktracker	205
Decommissioning a Tasktracker	206
Killing a MapReduce Job	206
Killing a MapReduce Task	207
Dealing with a Blacklisted Tasktracker	207
9. Troubleshooting	209
Differential Diagnosis Applied to Systems	209

Common Failures and Problems	211
Humans (You)	211
Misconfiguration	212
Hardware Failure	213
Resource Exhaustion	213
Host Identification and Naming	214
Network Partitions	214
“Is the Computer Plugged In?”	215
E-SPORE	215
Treatment and Care	217
War Stories	220
A Mystery Bottleneck	221
There’s No Place Like 127.0.0.1	224
10. Monitoring	229
An Overview	229
Hadoop Metrics	230
Apache Hadoop 0.20.0 and CDH3 (metrics1)	231
Apache Hadoop 0.20.203 and Later, and CDH4 (metrics2)	237
What about SNMP?	239
Health Monitoring	239
Host-Level Checks	240
All Hadoop Processes	242
HDFS Checks	244
MapReduce Checks	246
11. Backup and Recovery	249
Data Backup	249
Distributed Copy (distcp)	250
Parallel Data Ingestion	252
Namenode Metadata	254
Appendix: Deprecated Configuration Properties	257
Index	267

Introduction

Over the past few years, there has been a fundamental shift in data storage, management, and processing. Companies are storing more data from more sources in more formats than ever before. This isn't just about being a "data packrat" but rather building products, features, and intelligence predicated on knowing more about the world (where the world can be users, searches, machine logs, or whatever is relevant to an organization). Organizations are finding new ways to use data that was previously believed to be of little value, or far too expensive to retain, to better serve their constituents. Sourcing and storing data is one half of the equation. Processing that data to produce *information* is fundamental to the daily operations of every modern business.

Data storage and processing isn't a new problem, though. Fraud detection in commerce and finance, anomaly detection in operational systems, demographic analysis in advertising, and many other applications have had to deal with these issues for decades. What has happened is that the volume, velocity, and variety of this data has changed, and in some cases, rather dramatically. This makes sense, as many algorithms benefit from access to more data. Take, for instance, the problem of recommending products to a visitor of an ecommerce website. You could simply show each visitor a rotating list of products they could buy, hoping that one would appeal to them. It's not exactly an informed decision, but it's a start. The question is what do you need to improve the chance of showing the right person the right product? Maybe it makes sense to show them what you think they like, based on what they've previously looked at. For some products, it's useful to know what they already own. Customers who already bought a specific brand of laptop computer from you may be interested in compatible accessories and upgrades.¹ One of the most common techniques is to cluster users by similar behavior (such as purchase patterns) and recommend products purchased by "similar" users. No matter the solution, all of the algorithms behind these options require data

1. I once worked on a data-driven marketing project for a company that sold beauty products. Using purchase transactions of all customers over a long period of time, the company was able to predict when a customer would run out of a given product after purchasing it. As it turned out, simply offering them the same thing about a week before they ran out resulted in a (very) noticeable lift in sales.

and generally improve in quality with more of it. Knowing more about a problem space generally leads to better decisions (or algorithm efficacy), which in turn leads to happier users, more money, reduced fraud, healthier people, safer conditions, or whatever the desired result might be.

Apache Hadoop is a platform that provides pragmatic, cost-effective, scalable infrastructure for building many of the types of applications described earlier. Made up of a distributed filesystem called the Hadoop Distributed Filesystem (HDFS) and a computation layer that implements a processing paradigm called MapReduce, Hadoop is an open source, batch data processing system for enormous amounts of data. We live in a flawed world, and Hadoop is designed to survive in it by not only tolerating hardware and software failures, but also treating them as first-class conditions that happen regularly. Hadoop uses a cluster of plain old commodity servers with no specialized hardware or network infrastructure to form a single, logical, storage and compute platform, or *cluster*, that can be shared by multiple individuals or groups. Computation in Hadoop MapReduce is performed in parallel, automatically, with a simple abstraction for developers that obviates complex synchronization and network programming. Unlike many other distributed data processing systems, Hadoop runs the user-provided processing logic on the machine where the data lives rather than dragging the data across the network; a huge win for performance.

For those interested in the history, Hadoop was modeled after two papers produced by Google, one of the many companies to have these kinds of data-intensive processing problems. The first, presented in 2003, describes a pragmatic, scalable, distributed filesystem optimized for storing enormous datasets, called the Google Filesystem (<http://research.google.com/archive/gfs.html>), or *GFS*. In addition to simple storage, GFS was built to support large-scale, data-intensive, distributed processing applications. The following year, another paper, titled "MapReduce: Simplified Data Processing on Large Clusters" (<http://research.google.com/archive/mapreduce.html>), was presented, defining a programming model and accompanying framework that provided automatic parallelization, fault tolerance, and the scale to process hundreds of terabytes of data in a single job over thousands of machines. When paired, these two systems could be used to build large data processing clusters on relatively inexpensive, commodity machines. These papers directly inspired the development of HDFS and Hadoop MapReduce, respectively.

Interest and investment in Hadoop has led to an entire ecosystem of related software both open source and commercial. Within the Apache Software Foundation alone, projects that explicitly make use of, or integrate with, Hadoop are springing up regularly. Some of these projects make authoring MapReduce jobs easier and more accessible, while others focus on getting data in and out of HDFS, simplify operations, enable deployment in cloud environments, and so on. Here is a sampling of the more popular projects with which you should familiarize yourself:

Apache Hive (<http://hive.apache.org>)

Hive creates a relational database-style abstraction that allows developers to write a dialect of SQL, which in turn is executed as one or more MapReduce jobs on the cluster. Developers, analysts, and existing third-party packages already know and speak SQL (Hive's dialect of SQL is called HiveQL and implements only a subset of any of the common standards). Hive takes advantage of this and provides a quick way to reduce the learning curve to adopting Hadoop and writing MapReduce jobs. For this reason, Hive is by far one of the most popular Hadoop ecosystem projects.

Hive works by defining a table-like schema over an *existing* set of files in HDFS and handling the gory details of extracting records from those files when a query is run. The data on disk is never actually changed, just parsed at query time. HiveQL statements are interpreted and an execution plan of prebuilt map and reduce classes is assembled to perform the MapReduce equivalent of the SQL statement.

Apache Pig (<http://pig.apache.org>)

Like Hive, Apache Pig was created to simplify the authoring of MapReduce jobs, obviating the need to write Java code. Instead, users write data processing jobs in a high-level scripting language from which Pig builds an execution plan and executes a series of MapReduce jobs to do the heavy lifting. In cases where Pig doesn't support a necessary function, developers can extend its set of built-in operations by writing user-defined functions in Java (Hive supports similar functionality as well). If you know Perl, Python, Ruby, JavaScript, or even shell script, you can learn Pig's syntax in the morning and be running MapReduce jobs by lunchtime.

Apache Sqoop (<http://sqoop.apache.org>)

Not only does Hadoop not want to replace your database, it wants to be friends with it. Exchanging data with relational databases is one of the most popular integration points with Apache Hadoop. Sqoop, short for "SQL to Hadoop," performs bidirectional data transfer between Hadoop and almost any database with a JDBC driver. Using MapReduce, Sqoop performs these operations in parallel with no need to write code.

For even greater performance, Sqoop supports database-specific plug-ins that use native features of the RDBMS rather than incurring the overhead of JDBC. Many of these connectors are open source, while others are free or available from commercial vendors at a cost. Today, Sqoop includes native connectors (called *direct* support) for MySQL and PostgreSQL. Free connectors exist for Teradata, Netezza, SQL Server, and Oracle (from Quest Software), and are available for download from their respective company websites.

Apache Flume (<http://flume.apache.org>)

Apache Flume is a streaming data collection and aggregation system designed to transport massive volumes of data into systems such as Hadoop. It supports native connectivity and support for writing directly to HDFS, and simplifies reliable, streaming data delivery from a variety of sources including RPC services, log4j appenders, syslog, and even the output from OS commands. Data can be routed,

load-balanced, replicated to multiple destinations, and aggregated from thousands of hosts by a tier of agents.

Apache Oozie (<http://incubator.apache.org/oozie/>)

It's not uncommon for large production clusters to run many coordinated Map-Reduce jobs in a workflow. Apache Oozie is a workflow engine and scheduler built specifically for large-scale job orchestration on a Hadoop cluster. Workflows can be triggered by time or events such as data arriving in a directory, and job failure handling logic can be implemented so that policies are adhered to. Oozie presents a REST service for programmatic management of workflows and status retrieval.

Apache Whirr (<http://whirr.apache.org>)

Apache Whirr was developed to simplify the creation and deployment of ephemeral clusters in cloud environments such as Amazon's AWS. Run as a command-line tool either locally or within the cloud, Whirr can spin up instances, deploy Hadoop, configure the software, and tear it down on demand. Under the hood, Whirr uses the powerful jclouds (<http://www.jclouds.org/>) library so that it is cloud provider-neutral. The developers have put in the work to make Whirr support both Amazon EC2 and Rackspace Cloud. In addition to Hadoop, Whirr understands how to provision Apache Cassandra, Apache ZooKeeper, Apache HBase, ElasticSearch, Voldemort, and Apache Hama.

Apache HBase (<http://hbase.apache.org>)

Apache HBase is a low-latency, distributed (nonrelational) database built on top of HDFS. Modeled after Google's Bigtable (<http://research.google.com/archive/bigtable.html>), HBase presents a flexible data model with scale-out properties and a very simple API. Data in HBase is stored in a semi-columnar format partitioned by rows into *regions*. It's not uncommon for a single table in HBase to be well into the hundreds of terabytes or in some cases petabytes. Over the past few years, HBase has gained a massive following based on some very public deployments such as Facebook's Messages platform (http://www.facebook.com/note.php?note_id=454991608919). Today, HBase is used to serve huge amounts of data to real-time systems in major production deployments.

Apache ZooKeeper (<http://zookeeper.apache.org>)

A true workhorse, Apache ZooKeeper is a distributed, consensus-based coordination system used to support distributed applications. Distributed applications that require leader election, locking, group membership, service location, and configuration services can use ZooKeeper rather than reimplement the complex coordination and error handling that comes with these functions. In fact, many projects within the Hadoop ecosystem use ZooKeeper for exactly this purpose (most notably, HBase).

Apache HCatalog (<http://incubator.apache.org/hcatalog/>)

A relatively new entry, Apache HCatalog is a service that provides shared schema and data access abstraction services to applications with the ecosystem. The

long-term goal of HCatalog is to enable interoperability between tools such as Apache Hive and Pig so that they can share dataset metadata information.

The Hadoop ecosystem is exploding into the commercial world as well. Vendors such as Oracle, SAS, MicroStrategy, Tableau, Informatica, Microsoft, Pentaho, Talend, HP, Dell, and dozens of others have all developed integration or support for Hadoop within one or more of their products. Hadoop is fast becoming (or, as an increasingly growing group would believe, already has become) the de facto standard for truly large-scale data processing in the data center.

If you're reading this book, you may be a developer with some exposure to Hadoop looking to learn more about managing the system in a production environment. Alternatively, it could be that you're an application or system administrator tasked with owning the current or planned production cluster. Those in the latter camp may be rolling their eyes at the prospect of dealing with yet another system. That's fair, and we won't spend a ton of time talking about writing applications, APIs, and other pesky code problems. There are other fantastic books on those topics, especially *Hadoop: The Definitive Guide* (<http://shop.oreilly.com/product/0636920021773.do>) by Tom White (O'Reilly). Administrators do, however, play an absolutely critical role in planning, installing, configuring, maintaining, and monitoring Hadoop clusters. Hadoop is a comparatively low-level system, leaning heavily on the host operating system for many features, and it works best when developers and administrators collaborate regularly. What you do impacts how things work.

It's an extremely exciting time to get into Apache Hadoop. The so-called *big data* space is all the rage, sure, but more importantly, Hadoop is growing and changing at a staggering rate. Each new version—and there have been a few big ones in the past year or two—brings another truckload of features for both developers and administrators alike. You could say that Hadoop is experiencing software puberty; thanks to its rapid growth and adoption, it's also a little awkward at times. You'll find, throughout this book, that there are significant changes between even minor versions. It's a lot to keep up with, admittedly, but don't let it overwhelm you. Where necessary, the differences are called out, and a section in Chapter 4 is devoted to walking you through the most commonly encountered versions.

This book is intended to be a pragmatic guide to running Hadoop in production. Those who have some familiarity with Hadoop may already know alternative methods for installation or have differing thoughts on how to properly tune the number of map slots based on CPU utilization.² That's expected and more than fine. The goal is not to enumerate all possible scenarios, but rather to call out what works, as demonstrated in critical deployments.

Chapters 2 and 3 provide the necessary background, describing what HDFS and Map-Reduce are, why they exist, and at a high level, how they work. Chapter 4 walks you

2. We also briefly cover the flux capacitor and discuss the burn rate of energon cubes during combat.

through the process of planning for an Hadoop deployment including hardware selection, basic resource planning, operating system selection and configuration, Hadoop distribution and version selection, and network concerns for Hadoop clusters. If you are looking for the meat and potatoes, Chapter 5 is where it's at, with configuration and setup information, including a listing of the most critical properties, organized by topic. Those that have strong security requirements or want to understand identity, access, and authorization within Hadoop will want to pay particular attention to Chapter 6. Chapter 7 explains the nuts and bolts of sharing a single large cluster across multiple groups and why this is beneficial while still adhering to service-level agreements by managing and allocating resources accordingly. Once everything is up and running, Chapter 8 acts as a run book for the most common operations and tasks. Chapter 9 is the rainy day chapter, covering the theory and practice of troubleshooting complex distributed systems such as Hadoop, including some real-world war stories. In an attempt to minimize those rainy days, Chapter 10 is all about how to effectively monitor your Hadoop cluster. Finally, Chapter 11 provides some basic tools and techniques for backing up Hadoop and dealing with catastrophic failure.