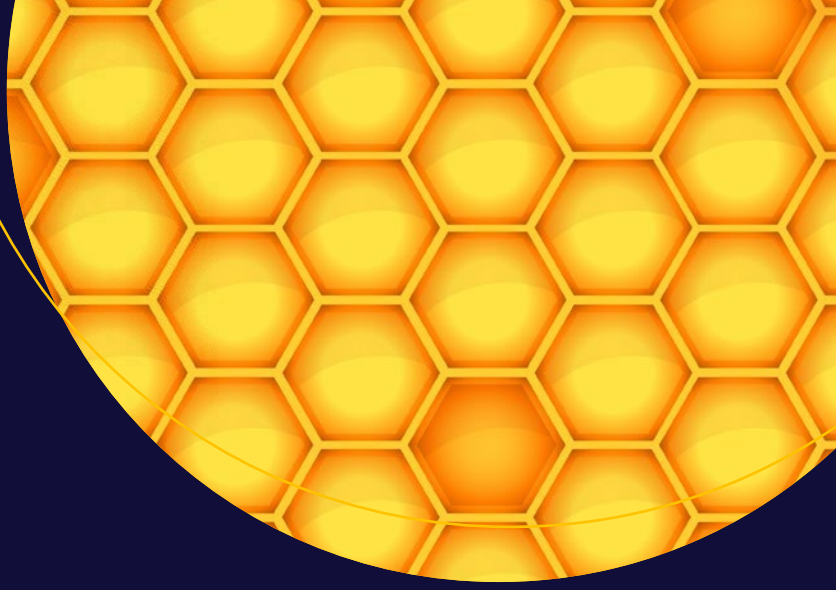

Practical Hive

A Guide to Hadoop's Data Warehouse System

Scott Shaw
Andreas Francois Vermeulen
Ankur Gupta
David Kjerrumgaard

Apress®

Practical Hive

A Guide to Hadoop's Data Warehouse System

Scott Shaw
Andreas François Vermeulen
Ankur Gupta
David Kjerrumgaard

Apress®

Practical Hive: A Guide to Hadoop's Data Warehouse System

Scott Shaw
Saint Louis, Missouri, USA

Andreas François Vermeulen
West Kilbride North Ayrshire, United Kingdom

Ankur Gupta
Uxbridge, United Kingdom

David Kjerrumgaard
Henderson, Nevada, USA

ISBN-13 (pbk): 978-1-4842-0272-2
ISBN-13 (electronic): 978-1-4842-0271-5
DOI 10.1007/978-1-4842-0271-5

Library of Congress Control Number: 2016951940

Managing Director: Welmoed Spahr
Acquisitions Editor: Robert Hutchinson
Developmental Editor: Matt Moodie
Technical Reviewer: Ancil McBarnett, Chris Hillman
Editorial Board: Steve Anglin, Pramila Balen, Laura Berendson, Aaron Black, Louise Corrigan, Jonathan Gennick, Robert Hutchinson, Celestin Suresh John, Nikhil Karkal, James Markham, Susan McDermott, Matthew Moodie, Natalie Pao, Gwenan Spearing
Coordinating Editor: Rita Fernando
Copy Editor: Kezia Endsley
Compositor: SPi Global
Indexer: SPi Global
Cover Image: Designed by FreePik

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springer.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a Delaware corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/.

Printed on acid-free paper

I dedicate this book to my family. They put up with me being on the computer everyday and yet they have no idea what I do for a living. Love you!

—Scott Shaw

I dedicate this book to my family and wise mentors for their support. Special thanks to Denise and Laurence.

—Andreas François Vermeulen

I would like to express my gratitude to the many people who saw me through this book. Above all I want to thank my wife, Jasveen, and the rest of my family, who supported and encouraged me in spite of all the time it took me away from them.

—Ankur Gupta

"By perseverance, study, and eternal desire, any man can become great." —George S. Patton

—David Kjerrumgaard

Contents at a Glance

Contents

About the Authors

Scott Shaw has over 15 years of data management experience. He has worked as an Oracle and SQL Server DBA. He has worked as a consultant on Microsoft business intelligence projects utilizing Tabular and OLAP models and co-authored two T-SQL books by Apress. Scott also enjoys speaking across the country about distributed computing, Big Data concepts, business intelligence, Hive, and the value of Hadoop. Scott works as a Senior Solutions Engineer for Hortonworks and lives in Saint Louis with his wife and two kids.

Andreas François Vermeulen is Consulting Manager of Business Intelligence, Big Data, Data Science, and Computational Analytics at Sopra-Steria, doctoral researcher at University of Dundee and St. Andrews on future concepts in massive distributed computing, mechatronics, Big Data, business intelligence, and deep learning. He owns and incubates the "Rapid Information Factory" data processing framework. He is active in developing next-generation processing frameworks and mechatronics engineering with over 35 years of international experience in data processing, software development, and system architecture. Andre is a data scientist, doctoral trainer, corporate consultant, principal systems architect, and speaker/author/columnist on data science, distributed computing, Big Data, business intelligence, and deep learning. Andre received his Bachelor's degree at the North West University at Potchefstroom, his Master of Business Administration at the University of Manchester, Master of Business Intelligence and Data Science at University of Dundee, and Doctor of Philosophy at the University of Dundee and St. Andrews.

Ankur Gupta is a Senior Solutions Engineer at Hortonworks. He has over 14 years of experience in data management, working as a Data Architect and Oracle DBA. Before joining the world of Big Data, he worked as an Oracle Consultant for Investment Banks in the UK. He is a regular speaker on Big Data concepts, Hive, Hadoop, Oracle in various events, and is an author of the *Oracle Goldengate 11g Complete Cookbook*. Ankur has a Master's degree in Computer Science and International Business. He is a Hadoop Certified Administrator and Oracle Certified Professional and lives in London with his wife.

David Kjerrumgaard is a systems architect at Hortonworks. He has 20 years of experience in software development and is a Certified Developer for Apache Hadoop (CCDH). Kjerrumgaard is the author of *Data Governance with Apache Falcon and Cloudera Developer Training for Apache Hadoop*. He received his bachelor's and master's degrees in Computer Science from Kent State University.

About the Technical Reviewers

Ancil McBarnett has been in the IT industry for over 20 years, where he initially began his "small data" career as an Oracle consultant and DBA in the Caribbean and Latin America. Ancil possesses an MBA with emphasis in Finance and a BSc. in Computer Science/Management.

Prior to working at Hortonworks he was the Architect Manager for a state agency responsible for sharing secure and sensitive data among first responder and justice systems and at Oracle, and was championing several Big Data and next generation Data Integration initiatives in a pre-sales capacity.

Since joining Hortonworks he has worked mainly with health providers who are looking to utilize Hadoop as the ideal platform to store and analyze secure data and to create modern data applications, with Hive as a pivotal tool to accomplish this.

You can find some of his articles on Hive and Tez tuning on the Hortonworks Community Connection.

Chris Hillman is Principal Data Scientist in the International Advanced Analytics team at Teradata. He has 20+ years of experience working in the business intelligence industry, mainly in the Retail and CPGN vertical, working as Solution Architect, Principal Consultant, and Technology Director. Chris works with the Teradata Aster Centre of Expertise and is involved in the pre-sale and start-up activities of Teradata Aster projects, helping customers understand whether MapReduce or SQL is an appropriate technique to use. Chris is currently studying part-time for a PhD in Data Science at the University of Dundee, applying Big Data analytics to the data produced from experimentation into the Human Proteome.

Acknowledgments

Even before I joined Hortonworks I wanted to write a book on Hive. At the time there weren't many and the ones I saw where technically sound but not for the average users and especially not for someone coming from the relational database world. Once I began working at Hortonworks I figured it would be easy to sit down and write the book. I had all the best resources at my fingertips and access to some of the brightest people I've ever met. I had Hive committers like Alan Gates who never hesitated to answer an e-mail or spend a moment to talk to you at a conference. I had the friendship and support of the best Solution Engineering team in the world. Yet almost 2 and a half years later, there was still no book.

What I didn't predict was the incredible pace of this market and the herculean time commitment all of us on the team endure to provide solutions to our customers. It is truly a labor of love, but between work and family, the book had to wait. It waited a long time. I think any other publisher would have kicked me out the door and looked elsewhere, but Apress held steady (although I cannot honestly say they didn't push back a little and deservedly so) and trusted that someday we would have a book.

The struggle with writing a book on Hive is if you wait six months between writing then you're writing a new book. I came to terms that this was not the job of one person and I needed help. Ankur was one of the first to step up. If not for Ankur's perseverance and commitment, this book would not be in your hands right now. It was also Ankur who put us in touch with Andre and, I'm certain Ankur would agree, without Andre's incredible writing output and knowledge, you would also not have a book in your hands or, at the very least, it would be smaller and you would be much less informed. Finally, thank you to David, who has truly provided the technical exclamation point on the book and was vital to rounding out the edges and moving us forward.

There are countless other people who have helped in any way they could with little time they had. Cindy Gross from the Microsoft CAT team was an early participant and helped to keep the project moving forward. Thank you to Ancil for stepping up and helping with much needed technical reviews—especially on my chapters. But most especially thank you to Hortonworks for not only supporting the book but being downright excited about it. The greater Hortonworks team wasn't excited about the book just because it is a Hive book; they were excited for us, the team of authors, for our accomplishment. I never was forced to choose between my work and the book; it was my choice to focus on work.

Finally, thank you to my family. My kids may never have a need for Hive but I know they think it's pretty cool that dad help write a book. It's been a long journey from the days I was an English major to now being a Solutions Engineer for an open source Big Data company writing technical books, but I really do still look around me and count my blessings. I'll say it again I work with some of the brightest people in the industry and although I can't hold a candle to their intelligence, I do know their collective knowledge and insight makes me a better person.

—Scott Shaw

Introduction

When I first learned about Hive I was working as a consultant on two data warehousing projects. One of them was in its sixth month of development. We had a team of 12 consultants and we were showing little progress. The source database was relational but, for some unknown reason, all the constraints such as primary and foreign key references had been turned off. For all intents and purposes, the source was non-relational and the team was struggling with moving the data into our highly structured data warehouse. We struggled with NULL values and building constraints as well as master data management issues and data quality. The goal at the end of the project was to have a data warehouse that would reproduce reports they already had.

The second project was smaller but involved hierarchical relationships. For example, a TV has a brand name, a SKU, a product code, and any number of other descriptive features. Some of these features are dynamic while others apply to one or more different products or brands. The hierarchy of features would be different from one brand to another. Again we were struggling with representing this business requirement in a relational data warehouse.

The first project represented the difficulty in moving from one schema to another. This problem had to be solved before anyone could ask any questions and, even then the questions had to be known ahead of time. The second project showed the difficulty in expressing business rules that did not fit into a rigid data structure. We found ourselves telling the customer to change their business rules to fit the structure.

When I first copied a file into HDFS and created a Hive table on top of the file, I was blown away by the simplicity of the solution yet by the far-reaching impact it would have on data analytics. Since that first simple beginning, I have seen data projects using Hive go from design to real analytic value built in weeks, which would take months with traditional approaches. Hive and the greater Hadoop ecosystem is truly a game-changer for data driven companies and for companies who need answers to critical business questions.

The purpose of this book is the hope that it will provide to you the same “ah-ha” moment I experienced. The purpose is to give you the foundation to explore and experience what Hive and Hadoop have to offer and to help you begin your journey into the technology that will drive innovation for the next decade or more. To survive in the technology field, you must constantly reinvent yourself. Technology is constantly travelling forward. Right now there is a train departing; welcome aboard.

CHAPTER 1

Setting the Stage for Hive: Hadoop

By now, any technical specialist with even a sliver of curiosity has heard the term Hadoop tossed around at the water cooler. The discussion likely ranges from, "Hadoop is a waste-of-time," to "This is big. This will solve all our current problems." You may also have heard your company director, manager, or even CIO ask the team to begin implementing this new Big Data thing and to somehow identify a problem it is meant to solve. One of the first responses I usually get from non-technical folks when mentioning Big Data is, "Oh, you mean like the NSA"? It is true that with Big Data comes big responsibility, but clearly, a lack of knowledge about the uses and benefits of Big Data can breed unnecessary FUD (fear, uncertainty, and doubt).

The fact you have this book in your hands shows you are interested in Hadoop. You may also know already how Hadoop allows you to store and process large quantities of data. We are guessing that you also realize that Hive is a powerful tool that allows familiar access to the data through SQL. As you may glean from its title, this book is about Apache Hive and how Hive is essential in gaining access to large data stores. With that in mind, it helps to understand why we are here. Why do we need Hive when we already have tools like T-SQL, PL/SQL, and any number of other analytical tools capable of retrieving data? Aren't there additional resource costs to adding more tools that demand new skills to an existing environment? The fact of the matter is, the nature of what we consider usable data is changing, and changing rapidly. This fast-paced change is forcing our hand and making us expand our toolsets beyond those we have relied on for the past 30 years. Ultimately, as we'll see in later chapters, we do need to change, but we also need to leverage the effort and skills we have already acquired.

Synonymous with Hadoop is the term *Big Data*. In our opinion, the term Big Data is slowly moving toward the fate of other terms like Decision Support System (DSS) or e-commerce. When people mention "Big Data" as a solution, they are usually viewing the problem from a marketing perspective, not from a tools or capability perspective. I recalled a meeting with a high-level executive who insisted we not use the term Big Data at all in our discussions. I agreed with him because I felt such a term dilutes the conversation by focusing on generic terminology instead of the truly transformative nature of the technology. But then again, the data really is getting big, and we have to start somewhere.

My point is that Hadoop, as we'll see, is a technology originally created to solve specific problems. It is evolving, faster than fruit flies in a jar, into a core technology that is changing the way companies think about their data—how they make use of and gain important insight into all of it—to solve specific business needs and gain a competitive advantage. Existing models and methodologies of handling data are being challenged. As it evolves and grows in acceptance, Hadoop is changing from a niche solution to something from which every enterprise can extract value. Think of it in the way other, now everyday technologies were created from specialized needs, such as those found in the military. Items we take for granted like duct tape and GPS were each developed first for specific military needs. Why did this happen? Innovation requires at least three ingredients: an immediate need, an identifiable problem, and money. The military is a huge,

Electronic supplementary material The online version of this chapter (doi:10.1007/978-1-4842-0271-5_1) contains supplementary material, which is available to authorized users.

S. Shaw et al., *Practical Hive*, DOI 10.1007/978-1-4842-0271-5_1

complex organization that has the talent, the money, the resources, and the need to invent these everyday items. Obviously, products the military invents for its own use are not often the same as those that end up in your retail store. The products get modified, generalized, and refined for everyday use. As we dig deeper into Hadoop, watch for the same process of these unique and tightly focused inventions evolving to meet the broader needs of the enterprise.

If Hadoop and Big Data are anything, they are a journey. Few companies come out of the gate requesting a 1,000-node cluster and decide over happy hour to run critical processes on the platform. Enterprises go through a predictable journey that can take anywhere from months to years. As you read through this book, the expectation is that it will help begin your journey and help elucidate particular steps in the overall journey. This first chapter is an introduction into why this Hadoop world is different and where it all started. This first chapter gives you a foundation for the later discussions. You will understand the platform before the individual technology and you will also learn about why the open source model is so different and disruptive.

An Elephant Is Born

In 2003 Google published an inconspicuous paper titled "The Google Filesystem" (http://static.googleusercontent.com/media/research.google.com/en/us/archive/gfs-sosp2003.pdf). Not many outside of Silicon Valley paid much attention to its publication or the message it was trying to convey. The message it told was directly applicable to a company like Google, whose primary business focused on indexing the Internet, which was not a common use case for most companies. The paper described a storage framework uniquely designed to handling the current future technological demands Google envisioned for its business. In the spirit of TL&DR, here are its most salient points:

- Failures are the norm
- Files are large
- Files are changed by appending, not by updating
- Closely coupled application and filesystem APIs

If you were a planning to become a multi-billion dollar Internet search company, many of these assumptions made sense. You would be primarily concerned with handling large files and executing long sequential reads and writes at the cost of low latency. You would also be interested in distributing your gigantic storage requirements across commodity hardware instead of building a vertical tower of expensive resources. Data ingestion was of primary concern and structuring (schematizing) this data on write would only delay the process. You also had at your disposal a team of world-class developers to architect the scalable, distributed, and highly available solution.

One company who took notice was Yahoo. They were experiencing similar scalability problems along Internet searching and were using an application called Nutch created by Doug Cutting and Mike Caffarella. The whitepaper provided Doug and Mike a framework for solving many problems inherent in the Nutch architecture, most importantly scalability and reliability. What needed to be accomplished next was a re-engineering of the solution based on the whitepaper designs.

■ **Note** Keep in mind the original GFS (Google Filesystem) is not the same as what has become Hadoop. GFS was a framework while Hadoop become the translation of the framework put into action. GFS within Google remained proprietary, i.e., not open source.

When we think of Hadoop, we usually think of the storage portion that Google encapsulated in the GFS whitepaper. In fact, the other half of the equation and, arguably more important, was a paper Google published in 2004 titled "MapReduce: Simplified Data Processing on Large Clusters" (http://static.googleusercontent.com/media/research.google.com/en/us/archive/mapreduce-osdi04.pdf). The MapReduce paper married the storage of data on a large, distributed cluster with the processing of that same data in what is called an "embarrassingly parallel" method.

■ **Note** We'll discuss MapReduce (MR) throughout this book. MR plays both a significant role as well as an increasingly diminishing role in interactive SQL query processing.

Doug Cutting, as well as others at Yahoo, saw the value of GFS and MapReduce for their own use cases at Yahoo and so spun off a separate project from Nutch. Doug named the project after the name of his son's stuffed elephant, Hadoop. Despite the cute name, the project was serious business and Yahoo set to scale it out to handle the demands of its search engine as well as its advertising.

■ **Note** There is an ongoing joke in the Hadoop community that when you leave product naming to engineering and not marketing you get names like Hadoop, Pig, Hive, Storm, Zookeeper, and Kafka. I, for one, love the nuisance and silliness of what is at heart applications solving complex and real-world problems. As far as the fate of Hadoop the elephant, Doug still carries him around to speaking events.

Yahoo's internal Hadoop growth is atypical in size but typical of the pattern of many current implementations. In the case of Yahoo, the initial development was able to scale to only a few nodes but after a few years they were able to scale to hundreds. As clusters grow and scale and begin ingesting more and more corporate data, silos within the organization begin to break down and users begin seeing more value in the data. As these silos break down across functional areas, more data moves into the cluster. What begins with hopeful purpose soon becomes the heart and soul or, more appropriately, the storage and analytical engine of an entire organization. As one author mentions:

> *By the time Yahoo spun out Hortonworks into a separate, Hadoop-focused software company in 2011, Yahoo's Hadoop infrastructure consisted of 42,000 nodes and hundreds of petabytes of storage (http://gigaom.com/2013/03/04/the-history-of-hadoop-from-4-nodes-to-the-future-of-data/).*

Hadoop Mechanics

Hadoop is a general term for two components: storage and processing. The storage component is the Hadoop Distributed File System (HDFS) and the processing is MapReduce.

■ **Note** The environment is changing as this is written. MapReduce has now become only one means of processing Hive on HDFS. MR is a traditional batch-orientated processing framework. New processing engines such as Tez are geared more toward near real-time query access. With the advent of YARN, HDFS is becoming more and more a multitenant environment allowing for many data access patterns such as batch, real-time, and interactive.

When we consider normal filesystems we think of operating systems like Windows or Linux. Those operating systems are installed on a single computer running essential applications. Now what would happen if we took 50 computers and networked them together? We still have 50 different operating systems and this doesn't do us much good if we want to run a single application that uses the compute power and resources of all of them.

For example, I am typing this on Microsoft Word, which can only be installed and run on a single operating system and a single computer. If I want to increase the operational performance of my Word application I have no choice but to add CPU and RAM to my computer. The problem is I am limited to the amount of RAM and CPU I can add. I would quickly hit a physical limitation for a single device.

HDFS, on the other hand, does something unique. You take 50 computers and install an OS on each of them. After networking them together you install HDFS on all them and declare one of the computers a master node and all the other computers worker nodes. This makes up your HDFS cluster. Now when you copy files to a directory, HDFS automatically stores parts of your file on multiple nodes in the cluster. HDFS becomes a virtual filesystem on top of the Linux filesystem. HDFS abstracts away the fact you're storing data on multiple nodes in a cluster. Figure 1-1 shows a high level view of how HDFS abstracts multiple systems away from the client.

Figure 1-1 is simplistic to say the least (we will elaborate on this in the section titled "Hadoop High Availability"). The salient point to take away is the ability to grow is now horizontal instead of vertical. Instead of adding CPU or RAM to a single device, you simply need to add a device, i.e., a node. Linear scalability allows you to quickly expand your capabilities based on your expanding resource needs. The perceptive reader will quickly counter that similar advantages are gained through virtualization. Let's take a look at the same figure through virtual goggles. Figure 1-2 shows this virtual architecture.

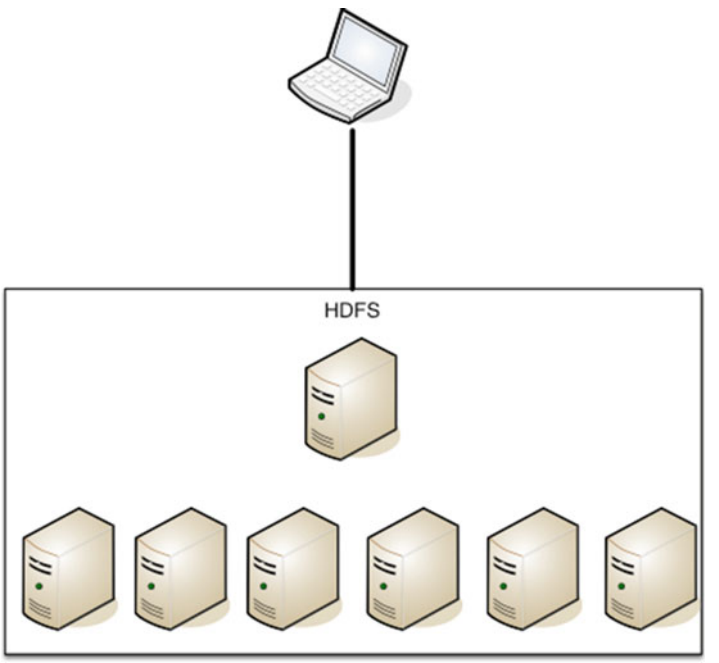

Figure 1-1. *Simplistic view of HDFS*

Administrators install virtual management software on a server or, in most cases, a cluster of servers. The software pools resources such as CPU and memory so that it looks as if there is a single server with a large amount of resources. On top of the virtual OS layer we had guests and divide the available pool of resources to each guest. The benefits include maximization of IO resources, dynamic provisioning of resources, and high availability at the physical cluster layer. Some problems include a dependency on SAN storage, inability to scale horizontally, as well as limitations to vertical scaling and reliance on multiple OS installations. Most current data centers follow this pattern and virtualization has been the primary IT trend for the past decade.

■ **Note** Figure 1-2 uses the term ESX. We certainly don't intend to pick on VMWare. We show the virtualization architecture only to demonstrate how Hadoop fundamentally changes the data center paradigm for unique modern data needs. Private cloud virtualization is a still a viable technology for many use cases and should be considered in conjunction with other architectures like appliances or public cloud.

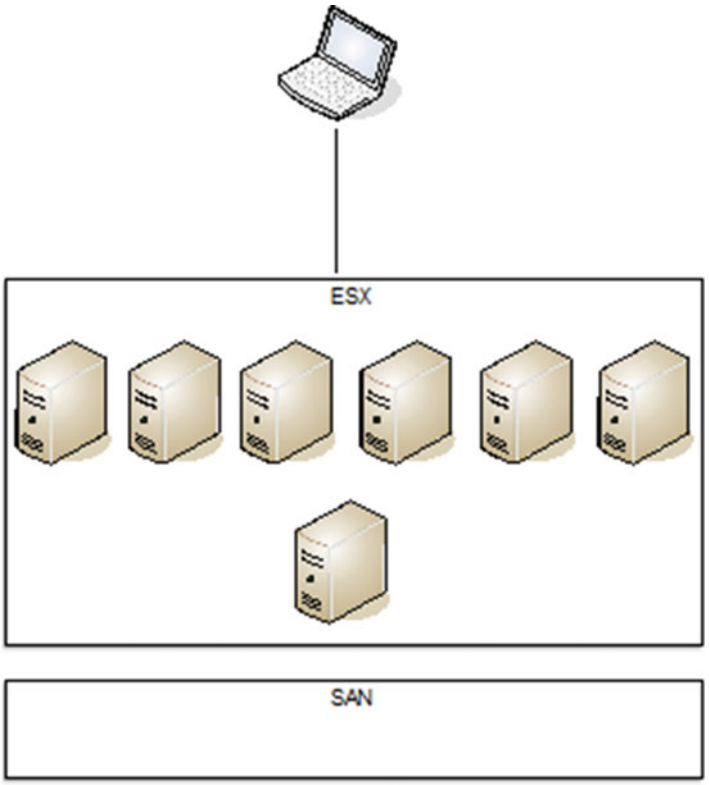

Figure 1-2. *Virtualization architecture*

Other advantages include reduced power consumption and reduced physical server footprint and dynamic provisioning. Hadoop has the unenviable task of going against a decade-long trend in virtual architecture. Enterprises have for years been moving away from physical architecture and making significant headway in diminishing the amount of physical servers they support in their data center. If Hadoop only provided the ability to add another physical node when needed to expand a filesystem, we would not be writing this book and Hadoop would go the way of Pets.com. There's much more to the architecture to make it transformative to businesses and worth the investment in a physical architecture.

Data Redundancy

Data at scale must also be highly available. Hadoop stores data efficiently and cheaply. There are mechanisms built into the Hadoop software architecture that allow us to use inexpensive hardware. As stated in the GFS whitepaper, the original design assumed nodes would fail. As clusters expand horizontally into the 100s, 1,000s, or even 10s of thousands, we are left with no option but to assume at least a few servers in the cluster will fail at any given time.

To have a few server failures jeopardize the health and integrity of the entire cluster would defeat any other benefits provided by HDFS, not to mention the Hadoop administrator turnover rate due to lack of sleep. Google and Yahoo engineers faced the daunting task of reducing cost while increasing uptime. The current HA solutions available were not capable of scaling out to their needs without burying the companies in hardware, software, and maintenance costs. Something had to change in order to meet their demands. Hadoop became the answer but first we need to look at why existing tools were not the solution.

Traditional High Availability

When we normally think of redundancy, we think in terms of high availability (HA). HA is an architecture describing how often you have access to your environment. We normally measure HA in terms of nines. We might say our uptime is 99.999, or five nines. Table 1-1 shows the actual downtime expected based on the HA percentage (http://en.wikipedia.org/wiki/High_availability).

Table 1-1. *HA Percentage Summary*

Availability Percent	Downtime Per Year	Downtime Per Month	Downtime Per Week
90% (“one nine”)	36.5 days	72 hours	16.8 hours
95%	18.25 days	36 hours	8.4 hours
97%	10.96 days	21.6 hours	5.04 hours
98%	7.30 days	14.4 hours	3.36 hours
99% (“two nines”)	3.65 days	7.20 hours	1.68 hours
99.5%	1.83 days	3.60 hours	50.4 minutes
99.8%	17.52 hours	86.23 minutes	20.16 minutes
99.9% (“three nines”)	8.76 hours	43.8 minutes	10.1 minutes
99.95%	4.38 hours	21.56 minutes	5.04 minutes
99.99% (“four nines”)	52.56 minutes	4.32 minutes	1.01 minutes
99.995%	26.28 minutes	2.16 minutes	30.24 seconds
99.999% (“five nines”)	5.26 minutes	25.9 seconds	6.05 seconds
99.9999% (“six nines”)	31.5 seconds	2.59 seconds	0.605 seconds
99.99999% (“seven nines”)	3.15 seconds	0.259 seconds	0.0605 seconds

Cost is traditionally a ratio of uptime. More uptime means higher cost. The majority of HA solutions center on hardware though a few solutions are also software dependent. Most involve the concept of a set of passive systems sitting in wait to be utilized if the primary system fails. Most cluster infrastructures fit this model. You may have a primary node and any number of secondary nodes containing replicated application binaries as well as the cluster specific software. Once the primary node fails, a secondary node takes over.

■ **Note** You can optionally set up an active/active cluster in which both systems are used. Your cost is still high since you need to account for, from a resource perspective, the chance of the applications from both systems running on one server in the event of a failure.

Quick failover minimizes downtime and, if the application running is cluster-aware and can account for the drop in session, the end user may never realize the system has failed. Virtualization uses this model. The physical hosts are generally a cluster of three or more systems in which one system remains passive in order to take over in the event an active system fails. The virtual guests can move across systems without the client even realizing the OS has moved to a different server. This model can also help with maintenance such as applying updates, patches, or swapping out hardware. Administrators perform maintenance on the secondary system and then make the secondary the primary for maintenance on the original system. Private clouds use a similar framework and, in most cases, have an idle server in the cluster primarily used for replacing a failed cluster node. Figure 1-3 shows a typical cluster configuration.

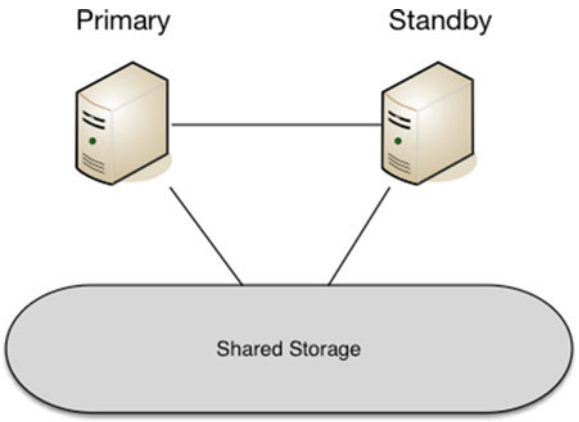

Figure 1-3. *Two-node cluster configuration with shared storage*

The cost for such a model can be high. Clusters require shared storage architecture, usually served by a SAN infrastructure. SANs can store a tremendous amount of data but they are expensive to build and maintain. SANs exist separate from the servers so data transmits across network interfaces. Furthermore, SANs intermix random IO with sequential IO, which means all IO becomes random. Finally, administrators configure most clusters to be active/passive. The passive standby server remains unused until a failure event. In this scenario hardware costs double without doubling your available resources.

Storage vendors use a number of means to maintain storage HA or storage redundancy. The most common is the use of RAID (Redundant Array of Independent Disks) configurations. Table 1-2 shows a quick overview of the most common RAID configurations.

Table 1-2. *The Most Common RAID Levels*

RAID Level	Description	Fault Tolerance
RAID 0	Stripe array	None
RAID 1	Mirror array	One disk
RAID 5	Stripe with parity	One disk
RAID 1+0	Striped mirrors	Multiple disks from one mirror