Mastering Hadoop

Go beyond the basics and master the next generation of Hadoop data processing platforms

Sandeep Karanth
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 "Hadoop 2.X"
- A synopsis of the book’s content
- More information on Mastering Hadoop

About the Author

**Sandeep Karanth** is a technical architect who specializes in building and operationalizing software systems. He has more than 14 years of experience in the software industry, working on a gamut of products ranging from enterprise data applications to newer-generation mobile applications. He has primarily worked at Microsoft Corporation in Redmond, Microsoft Research in India, and is currently a cofounder at Scibler, architecting data intelligence products.

Sandeep has special interest in data modeling and architecting data applications. In his area of interest, he has successfully built and deployed applications, catering to a variety of business use cases such as vulnerability detection from machine logs, churn analysis from subscription data, and sentiment analyses from chat logs. These applications were built using next generation big data technologies such as Hadoop, Spark, and Microsoft StreamInsight and deployed on cloud platforms such as Amazon AWS and Microsoft Azure.

Sandeep is also experienced and interested in areas such as green computing and the emerging Internet of Things. He frequently trains professionals and gives talks on topics such as big data and cloud computing. Sandeep believes in inculcating skill-oriented and industry-related topics in the undergraduate engineering curriculum, and his talks are geared with this in mind. Sandeep has a Master's degree in Computer and Information Sciences from the University of Minnesota, Twin Cities.

Sandeep's twitter handle is @karanth. His GitHub profile is https://github.com/Karanth, and he writes technical snippets at https://gist.github.com/Karanth.
Acknowledgments

I would like to dedicate this book to my loving daughter, Avani, who has taught me many a lesson in effective time management. I would like to thank my wife and parents for their constant support that has helped me complete this book on time. Packt Publishing have been gracious enough to give me this opportunity, and I would like to thank all individuals who were involved in editing, reviewing, and publishing this book. Questions and feedback from curious audiences at my lectures have driven much of the content of this book. Some of the subtopics are from experiences I gained working on a wide variety of projects throughout my career. I would like to thank my audience and also my employers for indirectly helping me write this book.
Mastering Hadoop

We are in an age where data is the primary driver in decision-making. With storage costs declining, network speeds increasing, and everything around us becoming digital, we do not hesitate a bit to download, store, or share data with others around us. About 20 years back, a camera was a device used to capture pictures on film. Every photograph had to be captured almost perfectly. The storage of film negatives was done carefully lest they get damaged. There was a higher cost associated with taking prints of these photographs. The time taken between a picture click and to view it was almost a day. This meant that less data was being captured as these factors presented a cliff for people from recording each and every moment of their life, unless it was very significant.

However, with cameras becoming digital, this has changed. We do not hesitate to click a photograph of almost anything anytime. We do not worry about storage as our externals disks of a terabyte capacity always provide a reliable backup. We seldom take our cameras anywhere as we have mobile devices that we can use to take photographs. We have applications such as Instagram that can be used to add effects to our pictures and share them. We gather opinions and information about the pictures, and we click and base some of our decisions on them. We capture almost every moment, of great significance or not, and push it into our memory books. The era of big data has arrived!

This era of Big Data has similar changes in businesses as well. Almost everything in a business is logged. Every action taken by a user on the page of an e-commerce page is recorded to improve quality of service and every item bought by the user are recorded to cross-sell or up-sell other items. Businesses want to understand the DNA of their customers and try to infer it by pinching out every possible data they can get about these customers. Businesses are not worried about the format of the data. They are ready to accept speech, images, natural language text, or structured data. These data points are used to drive business decisions and personalize experiences for the user. The more data, the higher the degree of personalization and better the experience for the user.

We saw that we are ready, in some aspects, to take on this Big Data challenge. However, what about the tools used to analyze this data? Can they handle the volume, velocity, and variety of the incoming data? Theoretically, all this data can reside on a single machine, but what is the cost of such a machine? Will it be able to cater to the variations in loads? We know that supercomputers are available, but there are only a handful of them in the world. Supercomputers don't scale. The alternative is to build a team of machines, a cluster, or individual computing units that work in tandem to achieve a task. A team of machines are interconnected via a very fast network and provide better scaling and elasticity, but that is not enough. These clusters have to be programmed. A greater number of machines, just like a team of human beings, require more coordination and synchronization. The higher the number of machines, the greater the possibility of
failures in the cluster. How do we handle synchronization and fault tolerance in a simple way easing the burden on the programmer? The answer is systems such as Hadoop.

Hadoop is synonymous with Big Data processing. Its simple programming model, "code once and deploy at any scale" paradigm, and an ever-growing ecosystem make Hadoop an inclusive platform for programmers with different levels of expertise and breadth of knowledge. Today, it is the number-one sought after job skill in the data sciences space. To handle and analyze Big Data, Hadoop has become the go-to tool. Hadoop 2.0 is spreading its wings to cover a variety of application paradigms and solve a wider range of data problems. It is rapidly becoming a general-purpose cluster platform for all data processing needs, and will soon become a mandatory skill for every engineer across verticals.

This book covers optimizations and advanced features of MapReduce, Pig, and Hive. It also covers Hadoop 2.0 and illustrates how it can be used to extend the capabilities of Hadoop.

Hadoop, in its 2.0 release, has evolved to become a general-purpose cluster-computing platform. The book will explain the platform-level changes that enable this. Industry guidelines to optimize MapReduce jobs and higher-level abstractions such as Pig and Hive in Hadoop 2.0 are covered. Some advanced job patterns and their applications are also discussed. These topics will empower the Hadoop user to optimize existing jobs and migrate them to Hadoop 2.0. Subsequently, it will dive deeper into Hadoop 2.0-specific features such as YARN (Yet Another Resource Negotiator) and HDFS Federation, along with examples. Replacing HDFS with other filesystems is another topic that will be covered in the latter half of the book. Understanding these topics will enable Hadoop users to extend Hadoop to other application paradigms and data stores, making efficient use of the available cluster resources.

This book is a guide focusing on advanced concepts and features in Hadoop. Foundations of every concept are explained with code fragments or schematic illustrations. The data processing flow dictates the order of the concepts in each chapter.

**What This Book Covers**

*Chapter 1, Hadoop 2.X*, discusses the improvements in Hadoop 2.X in comparison to its predecessor generation.

*Chapter 2, Advanced MapReduce*, helps you understand the best practices and patterns for Hadoop MapReduce, with examples.

*Chapter 3, Advanced Pig*, discusses the advanced features of Pig, a framework to script MapReduce jobs on Hadoop.

*Chapter 4, Advanced Hive*, discusses the advanced features of a higher-level SQL abstraction on Hadoop MapReduce called Hive.
Chapter 5, *Serialization and Hadoop I/O*, discusses the IO capabilities in Hadoop. Specifically, this chapter covers the concepts of serialization and deserialization support and their necessity within Hadoop; Avro, an external serialization framework; data compression codecs available within Hadoop; their tradeoffs; and finally, the special file formats in Hadoop.

Chapter 6, *YARN – Bringing Other Paradigms to Hadoop*, discusses YARN (Yet Another Resource Negotiator), a new resource manager that has been included in Hadoop 2.X, and how it is generalizing the Hadoop platform to include other computing paradigms.

Chapter 7, *Storm on YARN – Low Latency Processing in Hadoop*, discusses the opposite paradigm, that is, moving data to the compute, and compares and contrasts it with batch processing systems such as MapReduce. It also discusses the Apache Storm framework and how to develop applications in Storm. Finally, you will learn how to install Storm on Hadoop 2.X with YARN.

Chapter 8, *Hadoop on the Cloud*, discusses the characteristics of cloud computing and Hadoop's Platform as a Service offering across cloud computing service providers. Further, it delves into Amazon's managed Hadoop services, also known as Elastic MapReduce (EMR) and looks into how to provision and run jobs on a Hadoop EMR cluster.

Chapter 9, *HDFS Replacements*, discusses the strengths and drawbacks of HDFS when compared to other file systems. The chapter also draws attention to Hadoop's support for Amazon's S3 cloud storage service. At the end, the chapter illustrates Hadoop HDFS extensibility features by implementing Hadoop's support for S3's native file system to extend Hadoop.

Chapter 10, *HDFS Federation*, discusses the advantages of HDFS Federation and its architecture. Block placement strategies, which are central to the success of HDFS in the MapReduce environment, are also discussed in the chapter.

Chapter 11, *Hadoop Security*, focuses on the security aspects of a Hadoop cluster. The main pillars of security are authentication, authorization, auditing, and data protection. We will look at Hadoop's features in each of these pillars.

Chapter 12, *Analytics Using Hadoop*, discusses higher-level analytic workflows, techniques such as machine learning, and their support in Hadoop. We take document analysis as an example to illustrate analytics using Pig on Hadoop.

Appendix, *Hadoop for Microsoft Windows*, explores Microsoft Window Operating System's native support for Hadoop that has been introduced in Hadoop 2.0. In this chapter, we look at how to build and deploy Hadoop on Microsoft Windows natively.
Hadoop is the de facto open source framework used in the industry for large scale, massively parallel, and distributed data processing. It provides a computation layer for parallel and distributed computation processing. Closely associated with the computation layer is a highly fault-tolerant data storage layer, the Hadoop Distributed File System (HDFS). Both the computation and data layers run on commodity hardware, which is inexpensive, easily available, and compatible with other similar hardware.

In this chapter, we will look at the journey of Hadoop, with a focus on the features that make it enterprise-ready. Hadoop, with 6 years of development and deployment under its belt, has moved from a framework that supports the MapReduce paradigm exclusively to a more generic cluster-computing framework. This chapter covers the following topics:

- An outline of Hadoop's code evolution, with major milestones highlighted
- An introduction to the changes that Hadoop has undergone as it has moved from 1.X releases to 2.X releases, and how it is evolving into a generic cluster-computing framework
- An introduction to the options available for enterprise-grade Hadoop, and the parameters for their evaluation
- An overview of a few popular enterprise-ready Hadoop distributions
Hadoop 2.X

The inception of Hadoop

The birth and evolution of the Internet led to World Wide Web (WWW), a huge set of documents written in the markup language, HTML, and linked with one another via hyperlinks. Clients, known as browsers, became the user's window to WWW. Ease of creation, editing, and publishing of these web documents meant an explosion of document volume on the Web.

In the latter half of the 90s, the huge volume of web documents led to discoverability problems. Users found it hard to discover and locate the right document for their information needs, leading to a gold rush among web companies in the space of web discovery and search. Search engines and directory services for the Web, such as Lycos, Altavista, Yahoo!, and Ask Jeeves, became commonplace.

These search engines started ingesting and summarizing the Web. The process of traversing the Web and ingesting the documents is known as crawling. Smart crawlers, those that can download documents quickly, avoid link cycles, and detect document updates, have been developed.

In the early part of this century, Google emerged as the torchbearer of the search technology. Its success was attributed not only to the introduction of robust, spam-defiant relevance technology, but also its minimalistic approach, speed, and quick data processing. It achieved the former goals by developing novel concepts such as PageRank, and the latter goals by innovative tweaking and applying existing techniques, such as MapReduce, for large-scale parallel and distributed data processing.

PageRank is an algorithm named after Google's founder Larry Page. It is one of the algorithms used to rank web search results for a user. Search engines use keyword matching on websites to determine relevance corresponding to a search query. This prompts spammers to include many keywords, relevant or irrelevant, on websites to trick these search engines and appear in almost all queries. For example, a car dealer can include keywords related to shopping or movies and appear in a wider range of search queries. The user experience suffers because of irrelevant results.

PageRank thwarted this kind of fraud by analyzing the quality and quantity of links to a particular web page. The intention was that important pages have more inbound links.
In Circa 2004, Google published and disclosed its MapReduce technique and implementation to the world. It introduced **Google File System (GFS)** that accompanies the MapReduce engine. Since then, the MapReduce paradigm has become the most popular technique to process massive datasets in parallel and distributed settings across many other companies. Hadoop is an open source implementation of the MapReduce framework, and Hadoop and its associated filesystem, HDFS, are inspired by Google's MapReduce and GFS, respectively.

Since its inception, Hadoop and other MapReduce-based systems run a diverse set of workloads from different verticals, web search being one of them. As an example, Hadoop is extensively used in [http://www.last.fm/](http://www.last.fm/) to generate charts and track usage statistics. It is used for log processing in the cloud provider, Rackspace. Yahoo!, one of the biggest proponents of Hadoop, uses Hadoop clusters not only to build web indexes for search, but also to run sophisticated advertisement placement and content optimization algorithms.

## The evolution of Hadoop

Around the year 2003, Doug Cutting and Mike Cafarella started work on a project called **Nutch**, a highly extensible, feature-rich, and open source crawler and indexer project. The goal was to provide an off-the-shelf crawler to meet the demands of document discovery. Nutch can work in a distributed fashion on a handful of machines and be polite by respecting the `robots.txt` file on websites. It is highly extensible by providing the plugin architecture for developers to add custom components, for example, third-party plugins, to read different media types from the Web.

**Robot Exclusion Standard** or the `robots.txt` protocol is an advisory protocol that suggests crawling behavior. It is a file placed on website roots that suggest the public pages and directories that can or cannot be crawled. One characteristic of a polite crawler is its respect for the advisory instructions placed within the `robots.txt` file.

Nutch, together with indexing technologies such as Lucene and Solr, provided the necessary components to build search engines, but this project was not at web scale. The initial demonstration of Nutch involved crawling 100 million web pages using four machines. Moreover, debugging and maintaining it was tedious. In 2004, concepts from the seminal MapReduce and GFS publications from Google addressed some of Nutch's scaling issues. The Nutch contributors started integrating distributed filesystem features and the MapReduce programming model into the project. The scalability of Nutch improved by 2006, but it was not yet web scale. A few 100 million web documents could be crawled and indexed using 20 machines. Programming, debugging, and maintaining these search engines became easier.
In 2006, Yahoo hired Doug Cutting, and Hadoop was born. The Hadoop project was part of Apache Software Foundation (ASF), but was factored out of the existing Nutch project and allowed to evolve independently. A number of minor releases were done between 2006 and 2008, at the end of which Hadoop became a stable and web-scale data-processing MapReduce framework. In 2008, Hadoop won the terabyte sort benchmark competition, announcing its suitability for large-scale, reliable cluster-computing using MapReduce.

**Hadoop's genealogy**

The Hadoop project has a long genealogy, starting from the early releases in 2007 and 2008. This project that is part of Apache Software Foundation (ASF) will be termed Apache Hadoop throughout this book. The Apache Hadoop project is the parent project for subsequent releases of Hadoop and its distribution. It is analogous to the main stem of a river, while branches or distributions can be compared to the distributaries of a river.

The following figure shows the Hadoop lineage with respect to Apache Hadoop. In the figure, the black squares represent the major Apache Hadoop releases, and the ovals represent the distributions of Hadoop. Other releases of Hadoop are represented by dotted black squares.
Apache Hadoop has three important branches that are very relevant. They are:

- The 0.20.1 branch
- The 0.20.2 branch
- The 0.21 branch

The Apache Hadoop releases followed a straight line till 0.20. It always had a single major release, and there was no forking of the code into other branches. At release 0.20, there was a fan out of the project into three major branches. The 0.20.2 branch is often termed MapReduce v1.0, MRv1, or simply Hadoop 1.0.0. The 0.21 branch is termed MapReduce v2.0, MRv2, or Hadoop 2.0. A few older distributions are derived from 0.20.1. The year 2011 marked a record number of releases across different branches.

There are two other releases of significance, though they are not considered major releases. They are the Hadoop-0.20-append and Hadoop-0.20-Security releases. These releases introduced the HDFS append and security-related features into Hadoop, respectively. With these enhancements, Apache Hadoop came closer to becoming enterprise-ready.

**Hadoop-0.20-append**

Append is the primary feature of the Hadoop-0.20-append release. It allows users to run HBase without the risk of data loss. HBase is a popular column-family store that runs on HDFS, providing an online data store in a batch-oriented Hadoop environment. Specifically, the append feature helps write durability of HBase logs, ensuring data safety. Traditionally, HDFS supported input-output for MapReduce batch jobs. The requirement for these jobs was to open a file once, write a lot of data into it, and close the file. The closed file was immutable and read many times. The semantics supported were write-once-read-many-times. No one could read the file when a write was in progress.

Any process that failed or crashed during a write had to rewrite the file. In MapReduce, a user always reran tasks to generate the file. However, this is not true for transaction logs for online systems such as HBase. If the log-writing process fails, it can lead to data loss as the transaction cannot be reproduced. Reproducibility of a transaction, and in turn data safety, comes from log writing. The append feature in HDFS mitigates this risk by enabling HBase and other transactional operations on HDFS.
Hadoop-0.20-security

The Hadoop team at Yahoo took the initiative to add security-related features in the Hadoop-0.20-Security release. Enterprises have different teams, with each team working on different kinds of data. For compliance, client privacy, and security, isolation, authentication, and authorization of Hadoop jobs and data is important. The security release is feature-rich to provide these three pillars of enterprise security.

The full Kerberos authentication system is integrated with Hadoop in this release. Access Control Lists (ACLs) were introduced on MapReduce jobs to ensure proper authority in exercising jobs and using resources. Authentication and authorization put together provided the isolation necessary between both jobs and data of the different users of the system.

Hadoop's timeline

The following figure gives a timeline view of the major releases and milestones of Apache Hadoop. The project has been there for 8 years, but the last 4 years has seen Hadoop make giant strides in big data processing. In January 2010, Google was awarded a patent for the MapReduce technology. This technology was licensed to the Apache Software Foundation 4 months later, a shot in the arm for Hadoop. With legal complications out of the way, enterprises—small, medium, and large—were ready to embrace Hadoop. Since then, Hadoop has come up with a number of major enhancements and releases. It has given rise to businesses selling Hadoop distributions, support, training, and other services.
Hadoop 1.0 releases, referred to as 1.X in this book, saw the inception and evolution of Hadoop as a pure MapReduce job-processing framework. It has exceeded its expectations with a wide adoption of massive data processing. The stable 1.X release at this point of time is 1.2.1, which includes features such as append and security. Hadoop 1.X tried to stay flexible by making changes, such as HDFS append, to support online systems such as HBase. Meanwhile, big data applications evolved in range beyond MapReduce computation models. The flexibility of Hadoop 1.X releases had been stretched; it was no longer possible to widen its net to cater to the variety of applications without architectural changes.

Hadoop 2.0 releases, referred to as 2.X in this book, came into existence in 2013. This release family has major changes to widen the range of applications Hadoop can solve. These releases can even increase efficiencies and mileage derived from existing Hadoop clusters in enterprises. Clearly, Hadoop is moving fast beyond MapReduce to stay as the leader in massive scale data processing with the challenge of being backward compatible. It is becoming a generic cluster-computing and storage platform from being only a MapReduce-specific framework.

**Hadoop 2.X**

The extensive success of Hadoop 1.X in organizations also led to the understanding of its limitations, which are as follows:

- Hadoop gives unprecedented access to cluster computational resources to every individual in an organization. The MapReduce programming model is simple and supports a *develop once deploy at any scale* paradigm. This leads to users exploiting Hadoop for data processing jobs where MapReduce is not a good fit, for example, web servers being deployed in long-running map jobs. MapReduce is not known to be affable for iterative algorithms. Hacks were developed to make Hadoop run iterative algorithms. These hacks posed severe challenges to cluster resource utilization and capacity planning.

- Hadoop 1.X has a centralized job flow control. Centralized systems are hard to scale as they are the single point of load lifting. JobTracker failure means that all the jobs in the system have to be restarted, exerting extreme pressure on a centralized component. Integration of Hadoop with other kinds of clusters is difficult with this model.
Hadoop 2.X

- The early releases in Hadoop 1.X had a single NameNode that stored all the metadata about the HDFS directories and files. The data on the entire cluster hinged on this single point of failure. Subsequent releases had a cold standby in the form of a secondary NameNode. The secondary NameNode merged the edit logs and NameNode image files, periodically bringing in two benefits. One, the primary NameNode startup time was reduced as the NameNode did not have to do the entire merge on startup. Two, the secondary NameNode acted as a replica that could minimize data loss on NameNode disasters. However, the secondary NameNode (secondary NameNode is not a backup node for NameNode) was still not a hot standby, leading to high failover and recovery times and affecting cluster availability.

- Hadoop 1.X is mainly a Unix-based massive data processing framework. Native support on machines running Microsoft Windows Server is not possible. With Microsoft entering cloud computing and big data analytics in a big way, coupled with existing heavy Windows Server investments in the industry, it's very important for Hadoop to enter the Microsoft Windows landscape as well.

- Hadoop’s success comes mainly from enterprise play. Adoption of Hadoop mainly comes from the availability of enterprise features. Though Hadoop 1.X tries to support some of them, such as security, there is a list of other features that are badly needed by the enterprise.

**Yet Another Resource Negotiator (YARN)**

In Hadoop 1.X, resource allocation and job execution were the responsibilities of JobTracker. Since the computing model was closely tied to the resources in the cluster, MapReduce was the only supported model. This tight coupling led to developers force-fitting other paradigms, leading to unintended use of MapReduce.

The primary goal of YARN is to separate concerns relating to resource management and application execution. By separating these functions, other application paradigms can be added onboard a Hadoop computing cluster. Improvements in interoperability and support for diverse applications lead to efficient and effective utilization of resources. It integrates well with the existing infrastructure in an enterprise.

Achieving loose coupling between resource management and job management should not be at the cost of loss in backward compatibility. For almost 6 years, Hadoop has been the leading software to crunch massive datasets in a parallel and distributed fashion. This means huge investments in development, testing and deployment were already in place.
YARN maintains backward compatibility with Hadoop 1.X (hadoop-0.20.205+) APIs. An older MapReduce program can continue execution in YARN with no code changes. However, recompiling the older code is mandatory.

**Architecture overview**

The following figure lays out the architecture of YARN. YARN abstracts out resource management functions to a platform layer called **ResourceManager (RM)**. There is a per-cluster RM that primarily keeps track of cluster resource usage and activity. It is also responsible for allocation of resources and resolving contentions among resource seekers in the cluster. RM uses a generalized resource model and is agnostic to application-specific resource needs. For example, RM need not know the resources corresponding to a single Map or Reduce slot.

Planning and executing a single job is the responsibility of **Application Master (AM)**. There is an AM instance per running application. For example, there is an AM for each MapReduce job. It has to request for resources from the RM, use them to execute the job, and work around failures, if any.

The general cluster layout has RM running as a daemon on a dedicated machine with a global view of the cluster and its resources. Being a global entity, RM can ensure fairness depending on the resource utilization of the cluster resources. When requested for resources, RM allocates them dynamically as a node-specific bundle called a **container**. For example, 2 CPUs and 4 GB of RAM on a particular node can be specified as a container.
Every node in the cluster runs a daemon called NodeManager (NM). RM uses NM as its node local assistant. NMs are used for container management functions, such as starting and releasing containers, tracking local resource usage, and fault reporting. NMs send heartbeats to RM. The RM view of the system is the aggregate of the views reported by each NM.

Jobs are submitted directly to RMs. Based on resource availability, jobs are scheduled to run by RMs. The metadata of the jobs are stored in persistent storage to recover from RM crashes. When a job is scheduled, RM allocates a container for the AM of the job on a node in the cluster.

AM then takes over orchestrating the specifics of the job. These specifics include requesting resources, managing task execution, optimizations, and handling tasks or job failures. AM can be written in any language, and different versions of AM can execute independently on a cluster.

An AM resource request contains specifications about the locality and the kind of resource expected by it. RM puts in its best effort to satisfy AM's needs based on policies and availability of resources. When a container is available for use by AM, it can launch application-specific code in this container. The container is free to communicate with its AM. RM is agnostic to this communication.

**Storage layer enhancements**

A number of storage layer enhancements were undertaken in the Hadoop 2.X releases. The number one goal of the enhancements was to make Hadoop enterprise ready.

**High availability**

NameNode is a directory service for Hadoop and contains metadata pertaining to the files within cluster storage. Hadoop 1.X had a secondary Namenode, a cold standby that needed minutes to come up. Hadoop 2.X provides features to have a hot standby of NameNode. On the failure of an active NameNode, the standby can become the active Namenode in a matter of minutes. There is no data loss or loss of NameNode service availability. With hot standbys, automated failover becomes easier too.
The key to keep the standby in a hot state is to keep its data as current as possible with respect to the active Namenode. This is achieved by reading the edit logs of the active NameNode and applying it onto itself with very low latency. The sharing of edit logs can be done using the following two methods:

- A shared NFS storage directory between the active and standby NameNodes: the active writes the logs to the shared location. The standby monitors the shared directory and pulls in the changes.
- A quorum of Journal Nodes: the active NameNode presents its edits to a subset of journal daemons that record this information. The standby node constantly monitors these journal daemons for updates and syncs the state with itself.

The following figure shows the high availability architecture using a quorum of Journal Nodes. The data nodes themselves send block reports directly to both the active and standby NameNodes:

Zookeeper or any other High Availability monitoring service can be used to track NameNode failures. With the assistance of Zookeeper, failover procedures to promote the hot standby as the active NameNode can be triggered.
HDFS Federation
Similar to what YARN did to Hadoop's computation layer, a more generalized storage model has been implemented in Hadoop 2.X. The block storage layer has been generalized and separated out from the filesystem layer. This separation has given an opening for other storage services to be integrated into a Hadoop cluster. Previously, HDFS and the block storage layer were tightly coupled.

One use case that has come forth from this generalized storage model is HDFS Federation. Federation allows multiple HDFS namespaces to use the same underlying storage. Federated NameNodes provide isolation at the filesystem level. In Chapter 10, HDFS Federation, we will delve into the details of this feature.

HDFS snapshots
Snapshots are point-in-time, read-only images of the entire or a particular subset of a filesystem. Snapshots are taken for three general reasons:

- Protection against user errors
- Backup
- Disaster recovery

Snapshooting is implemented only on NameNode. It does not involve copying data from the data nodes. It is a persistent copy of the block list and file size. The process of taking a snapshot is almost instantaneous and does not affect the performance of NameNode.

Other enhancements
There are a number of other enhancements in Hadoop 2.X, which are as follows:

- The wire protocol for RPCs within Hadoop is now based on Protocol Buffers. Previously, Java serialization via Writables was used. This improvement not only eases maintaining backward compatibility, but also aids in rolling the upgrades of different cluster components. RPCs allow for client-side retries as well.
- HDFS in Hadoop 1.X was agnostic about the type of storage being used. Mechanical or SSD drives were treated uniformly. The user did not have any control on data placement. Hadoop 2.X releases in 2014 are aware of the type of storage and expose this information to applications as well. Applications can use this to optimize their data fetch and placement strategies.
- HDFS append support has been brought into Hadoop 2.X.
• HDFS access in Hadoop 1.X releases has been through HDFS clients. In Hadoop 2.X, support for NFSv3 has been brought into the NFS gateway component. Clients can now mount HDFS onto their compatible local filesystem, allowing them to download and upload files directly to and from HDFS. Appends to files are allowed, but random writes are not.

• A number of I/O improvements have been brought into Hadoop. For example, in Hadoop 1.X, clients collocated with data nodes had to read data via TCP sockets. However, with short-circuit local reads, clients can directly read off the data nodes. This particular interface also supports zero-copy reads. The CRC checksum that is calculated for reads and writes of data has been optimized using the Intel SSE4.2 CRC32 instruction.

Support enhancements

Hadoop is also widening its application net by supporting other platforms and frameworks. One dimension we saw was onboarding of other computational models with YARN or other storage systems with the Block Storage layer. The other enhancements are as follows:

• Hadoop 2.X supports Microsoft Windows natively. This translates to a huge opportunity to penetrate the Microsoft Windows server land for massive data processing. This was partially possible because of the use of the highly portable Java programming language for Hadoop development. The other critical enhancement was the generalization of compute and storage management to include Microsoft Windows.

• As part of Platform-as-a-Service offerings, cloud vendors give out on-demand Hadoop as a service. OpenStack support in Hadoop 2.X makes it conducive for deployment in elastic and virtualized cloud environments.

Hadoop distributions

In the present day, Hadoop and its individual ecosystem components are complex projects. As we saw earlier in this chapter, Hadoop has a number of different forks or code branches over a large number of releases. There are also a lot of different distributions of Hadoop. The distribution with the most activity and community involvement is the one that resides as part of Apache Software Foundation. This distribution is free and has a very large community behind it. The community contributions to the Apache Hadoop distribution shape the general direction taken by Hadoop. Support in the Apache Hadoop distribution is via online forums, where questions are addressed to the community and answered by its members.
Deployment and management of the Apache Hadoop distribution within an enterprise is tedious and nontrivial. Apache Hadoop is written in Java and optimized to run on Linux filesystems. This can lead to impedance mismatch between Hadoop and existing enterprise applications and infrastructures. Integration between the Hadoop ecosystem components is buggy and not straightforward.

To bridge these issues, a few companies came up with distribution models for Hadoop. There are three primary kinds of Hadoop distribution flavors. One flavor is to provide commercial or paid support and training for the Apache Hadoop distribution. Secondly, there are companies that provide a set of supporting tools for deployment and management of Apache Hadoop as an alternative flavor. These companies also provide robust integration layers between the different Hadoop ecosystem components. The third model is for companies to supplement Apache Hadoop with proprietary features and code. These features are paid enhancements, many of which solve certain use cases.

The parent of all these distributions is Apache Software Foundation’s Hadoop sources. Users of these other distributions, particularly from companies following the third distribution model, might integrate proprietary code into Apache Hadoop. However, these distributions will always stay in touching distance with Apache Hadoop and follow its trends. Distributions are generally well tested and supported in a deep and timely manner, saving administration and management costs for an organization. The downside of using a distribution other than Apache Hadoop is vendor lock-in. The tools and proprietary features provided by one vendor might not be available in another distribution or be noncompatible with other third-party tools, bringing in a cost of migration. The cost of migration is not limited to technology shifts alone. It also involves training, capacity planning, and rearchitecting costs for the organization.

**Which Hadoop distribution?**

There are a number of Hadoop distributions offered by companies since 2008. Distributions excel in some or the other attribute. Decisions on the right distribution for an enterprise or organization should be made on a case-by-case basis. There are different criteria to evaluate distributions. We will inspect a few important ones.
Performance

The ability of the Hadoop distribution running on a cluster to process data quickly is obviously a desired feature. Traditionally, this has been the cornerstone for all performance benchmarks. This particular performance measure is termed as "throughput". A wide range of analysis workloads that are being processed on Hadoop, coupled with the diversity of use cases supported by analytics, brings in "latency" as an important performance criterion as well. The ability of the cluster to ingest input data and emit output data at a quick rate becomes very important for low-latency analytics. This input-output cost forms an integral part of the data processing workflow.

Latency is the time required to perform an action. It is measured in time units such as milliseconds, seconds, minutes, or hours.

Throughput is the number of actions that can be performed in unit time. It gives a sense of the amount of work done for every time unit.

Scaling up hardware is one way to achieve low latency independent of the Hadoop distribution. However, this approach will be expensive and saturate out quickly. Architecturally, low I/O latency can be achieved in different ways; one will be able to reduce the number of intermediate data-staging layers between the data source or the data sink and Hadoop cluster. Some distributions provide streaming writes into the Hadoop cluster in an attempt to reduce intermediate staging layers. Operators used for filtering, compressing, and lightweight data processing can be plugged into the streaming layer to preprocess the data before it flows into storage.

The Apache Hadoop distribution is written in Java, a language that runs in its own virtual machine. Though this increases application portability, it comes with overheads such as an extra layer of indirection during execution by means of byte-code interpretation and background garbage collection. It is not as fast as an application directly compiled for target hardware. Some vendors optimize their distributions for particular hardware, increasing job performance per node. Features such as compression and decompression can also be optimized for certain hardware types.
**Scalability**

Over time, data outgrows the physical capacity of the compute and storage resources provisioned by an organization. This will require expansion of resources in both the compute and storage dimensions. Scaling can be done vertically or horizontally. Vertical scaling or scaling up is expensive and tightly bound to hardware advancements. Lack of elasticity is another downside with vertical scaling. Horizontal scaling or scaling out is a preferred mode of scaling compute and storage.

Ideally, scaling out should be limited to addition of more nodes and disks to the cluster network, with minimal configuration changes. However, distributions might impose different degrees of difficulty, both in terms of effort and cost on scaling a Hadoop cluster. Scaling out might mean heavy administrative and deployment costs, rewriting a lot of the application's code, or a combination of both. Scaling costs will depend on the existing architecture and how it complements and complies with the Hadoop distribution that is being evaluated.

**Vertical scaling** or **scaling up** is the process of adding more resources to a single node in a system. For example, adding additional CPUs, memory, or storage to a single computer comes under this bucket of scaling. Vertical scaling increases capacity, but does not decrease system load.

**Horizontal scaling** or **scaling out** is the process of adding additional nodes to a system. For example, adding another computer to a distributed system by connecting it to the network comes under this category of scaling. Horizontal scaling decreases the load on a system as the new node takes a part of the load. The capacity of individual nodes does not increase.

**Reliability**

Any distributed system is subject to partial failures. Failures can stem from hardware, software, or network issues, and have a smaller mean time when running on commodity hardware. Dealing with these failures without disrupting services or compromising data integrity is the primary goal of any highly available and consistent system.

A distribution that treats reliability seriously provides high availability of its components out of the box. Eliminating **Single Point of Failures (SPOF)** ensures availability. The means of eliminating SPOFs is to increase the redundancy of components. For a long time, Apache Hadoop had a single NameNode. Any failure to the NameNode's hardware meant the entire cluster becoming unusable. Now, there is the concept of a secondary NameNode and hot standbys that can be used to restore the name node in the event of NameNode failure.
Distributions that reduce manual tasks for cluster administrators are more reliable. Human intervention is directly correlated to higher error rates. An example of this is handling failovers. Failovers are critical periods for systems as they operate with lower degrees of redundancy. Any error during these periods can be disastrous for the application. Also, automated failover handling means the system can recover and run in a short amount of time. Lower the recovery time from failure better is the availability of the system.

The integrity of data needs to be maintained during normal operations and when failures are encountered. Data checksums for error detection and possible recovery, data replication, data mirroring, and snapshots are some ways to ensure data safety. Replication follows the redundancy theme to ensure data availability. Rack-aware smart placement of data and handling under or over replication are parameters to watch out for. Mirroring helps recovery from site failures by asynchronous replication across the Internet. Snapshotting is a desirable feature in any distribution; not only do they aid disaster recovery but also facilitate offline access to data. Data analytics involves experimentation and evaluation of rich data. Snapshots can be a way to facilitate this to a data scientist without disrupting production.

**Manageability**

Deploying and managing the Apache Hadoop open source distribution requires internal understanding of the source code and configuration. This is not a widely available skill in IT administration. Also, administrators in enterprises are caretakers of a wide range of systems, Hadoop being one of them.

Versions of Hadoop and its ecosystem components that are supported by a distribution might need to be evaluated for suitability. Newer versions of Hadoop support paradigms other than MapReduce within clusters. Depending on the plans of the enterprise, newer versions can increase the efficiency of enterprise-provisioned hardware.

Capabilities of Hadoop management tools are key differentiators when choosing an appropriate distribution for an enterprise. Management tools need to provide centralized cluster administration, resource management, configuration management, and user management. Job scheduling, automatic software upgrades, user quotas, and centralized troubleshooting are other desirable features.
Monitoring cluster health is also a key feature in the manageability function. Dashboards for visualization of cluster health and integration points for other tools are good features to have in distribution. Ease of data access is another parameter that needs to be evaluated; for example, support for POSIX filesystems on Hadoop will make browsing and accessing data convenient for engineers and scientists within any enterprise. On the flip side, this makes mutability of data possible, which can prove to be risky in certain situations.

Evaluation of options for data security of a distribution is extremely important as well. Data security entails authentication of a Hadoop user and authorization to datasets and data confidentiality. Every organization or enterprise might have its authentication systems such as Kerberos or LDAP already in place. Hadoop distribution, with capabilities to integrate with existing authentication systems, is a big plus in terms of lower costs and higher compliance. Fine-grained authorization might help control access to datasets and jobs at different levels. When data is moving in and out of an organization, encryption of the bits travelling over the wire becomes important to protect against data snooping.

Distributions offer integration with development and debugging tools. Developers and scientists in an enterprise will already be using a set of tools. The more overlap between the toolset used by the organization and distribution, the better it is. The advantage of overlap not only comes in the form of licensing costs, but also in a lesser need for training and orientation. It might also increase productivity within the organization as people are already accustomed to certain tools.

Available distributions
There are a number of distributions of Hadoop. A comprehensive list can be found at http://wiki.apache.org/hadoop/Distributions%20and%20Commercial%20Support. We will be examining four of them:

- Cloudera Distribution of Hadoop (CDH)
- Hortonworks Data Platform (HDP)
- MapR
- Pivotal HD
Cloudera Distribution of Hadoop (CDH)
Cloudera was formed in March 2009 with a primary objective of providing Apache Hadoop software, support, services, and training for enterprise-class deployment of Hadoop and its ecosystem components. The software suite is branded as Cloudera Distribution of Hadoop (CDH). The company being one of the Apache Software Foundation sponsors, pushes most enhancements it makes during support and servicing of Hadoop deployments upstream back into Apache Hadoop.

CDH is in its fifth major version right now and is considered a mature Hadoop distribution. The paid version of CDH comes with a proprietary management software, Cloudera Manager.

Hortonworks Data Platform (HDP)
The Yahoo Hadoop team spurned off to form Hortonworks in June, 2011, a company with objectives similar to Cloudera. Their distribution is branded as Hortonworks Data Platform (HDP). The HDP suite's Hadoop and other software are completely free, with paid support and training. Hortonworks also pushes enhancements upstream, back to Apache Hadoop.

HDP is in its second major version currently and is considered the rising star in Hadoop distributions. It comes with a free and open source management software called Ambari.

MapR
MapR was founded in 2009 with a mission to bring enterprise-grade Hadoop. The Hadoop distribution they provide has significant proprietary code when compared to Apache Hadoop. There are a handful of components where they guarantee compatibility with existing Apache Hadoop projects. Key proprietary code for the MapR distribution is the replacement of HDFS with a POSIX-compatible NFS. Another key feature is the capability of taking snapshots.

MapR comes with its own management console. The different grades of the product are named as M3, M5, and M7. M5 is a standard commercial distribution from the company, M3 is a free version without high availability, and M7 is a paid version with a rewritten HBase API.
Pivotal HD
Greenplum is a marquee parallel data store from EMC. EMC integrated Greenplum within Hadoop, giving way to an advanced Hadoop distribution called Pivotal HD. This move alleviated the need to import and export data between stores such as Greenplum and HDFS, bringing down both costs and latency.

The HAWQ technology provided by Pivotal HD allows efficient and low-latency query execution on data stored in HDFS. The HAWQ technology has been found to give 100 times more improvement on certain MapReduce workloads when compared to Apache Hadoop. HAWQ also provides SQL processing in Hadoop, increasing its popularity among users who are familiar with SQL.

Summary
In this chapter, we saw the evolution of Hadoop and some of its milestones and releases. We went into depth on Hadoop 2.X and the changes it brings into Hadoop. The key takeaways from this chapter are:

- MapReduce was born out of the necessity to gather, process, and index data at web scale. Apache Hadoop is an open source distribution of the MapReduce computational model.
- In over 6 years of its existence, Hadoop has become the number one choice as a framework for massively parallel and distributed computing. The community has been shaping Hadoop to gear up for enterprise use. In 1.X releases, HDFS append and security, were the key features that made Hadoop enterprise-friendly.
- MapReduce supports a limited set of use cases. Onboarding other paradigms into Hadoop enables support for a wider range of analytics and can also increase cluster resource utilization. In Hadoop 2.X, the JobTracker functions are separated and YARN handles cluster resource management and scheduling. MapReduce is one of the applications that can run on YARN.
- Hadoop’s storage layer was enhanced in 2.X to separate the filesystem from the block storage service. This enables features such as supporting multiple namespaces and integration with other filesystems. 2.X shows improvements in Hadoop storage availability and snapshotting.
- Distributions of Hadoop provide enterprise-grade management software, tools, support, training, and services. Most distributions shadow Apache Hadoop in their capabilities.

MapReduce is still an integral part of Hadoop’s DNA. In the next chapter, we will explore MapReduce optimizations and best practices.
Where to buy this book

You can buy Mastering Hadoop from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.