Big Data

About the Author

Dr. Anil Maheshwari is a Professor of Management Information Systems and Director of Center for Data Analytics at Maharishi University of Management, Fairfield, Iowa, USA.

He received a bachelor's degree in Electrical Engineering from Indian Institute of Technology (IIT) Delhi, and further received his MBA degree from Indian Institute of Management (IIM) Ahmedabad. He earned his PhD from Case Western Reserve University, Cleveland, Ohio. He has been a Professor at the University of Cincinnati, City University of New York, among others. His research has been published in prestigious journals and conferences. He teaches data analytics, big data, leadership, and marketing. He has authored many books in data science and leadership. He has also worked in the global IT industry for over 20 years, including leadership roles at IBM in Austin, Texas. He has completed various leadership and marketing training programs at IBM and also won several awards.

He is a practitioner of Transcendental Meditation (TM) and TM-Sidhi techniques. He blogs on IT and Enlightenment at anilmah.com, and is a popular speaker on those topics. He is also a marathoner. He can be reached at akm2030@gmail.com.

BIG DATA

Dr. Anil Maheshwari

Professor of Management Information Systems and
Director of Center for Data Analytics
Maharishi University of Management
Fairfield, Iowa, USA.

McGraw Hill Education (India) Private Limited
CHENNAI

McGraw Hill Education Offices
Chennai New York St Louis San Francisco Auckland Bogotá Caracas
Kuala Lumpur Lisbon London Madrid Mexico City Milan Montreal
San Juan Santiago Singapore Sydney Tokyo Toronto

McGraw Hill Education (India) Private Limited

Published by McGraw Hill Education (India) Private Limited
444/1, Sri Ekambara Naicker Industrial Estate, Alapakkam, Porur, Chennai - 600 116

Big Data

Print Edition:
ISBN-13: 978-93-5260-502-6
ISBN-10: 93-5260-502-0

E-Book Edition:
ISBN-13: 978-93-5260-454-8
ISBN-10: 93-5260-454-7

Managing Director: *Kaushik Bellani*

Director—Science & Engineering Portfolio: *Vibha Mahajan*
Lead—Science & Engineering Portfolio: *Hemant Jha*

Content Development Lead: *Shalini Jha*
Content Developer: *Shehla Mirza*

Production Head: *Satinder S Baveja*
Sr. Manager—Production: *Piyaray Pandita*

General Manager—Production: *Rajender P Ghansela*
Manager—Production: *Reji Kumar*

Typeset at The Composers, 260, C.A. Apt., Paschim Vihar, New Delhi 110 063 and printed at

Cover Printer:

Visit us at: www.mheducation.co.in

Dedicated to

My family of three lovely ladies
Wife Neerja, and daughters Ankita and Nupur
who taught me love, care and gentleness

Preface

Big Data is a new, and inclusive, natural phenomenon, as big and messy as nature itself. It requires a new kind of consciousness to fathom its scale and scope, and its many opportunities and challenges. Understanding the essentials of Big Data requires suspending many conventional expectations and assumptions about data ... such as completeness, clarity, consistency, and conciseness. Fathoming and taming the multi-layered Big Data is a dream that is slowly becoming a reality. It is a rapidly evolving field that is growing exponentially in value and capabilities.

There are a growing number of books being written on Big Data. They fall mostly in two categories. There are those that focus on business aspects, and discuss the strategic internal shifts required for reaping the business benefits from the many opportunities offered by Big Data. Then there are those that focus on particular technology platforms, such as Hadoop or Spark. This book aims to bring together the business context and the technologies in a seamless way.

Thanks to Maharishi Mahesh Yogi for creating a wonderful university whose consciousness-based environment made writing this evolutionary book possible. Thanks to many current and former students for contributing to this book. Dheeraj Pandey assisted with the Weblog analyzer application and its details. Suraj Thapalia assisted with the Hadoop installation guide. Enkh Tseeleesuren helped write the Spark tutorial. Thanks to my family for supporting me in this process. My daughters Ankita and Nupur reviewed the book and made helpful comments. My father Mr. R L Maheshwari and brother Dr. Sunil Maheshwari also read the book and enthusiastically approved it. My colleague Dr. Edi Shivaji too reviewed the book.

May the Big Data Force be with you!

Dr. Anil Maheshwari
May 2017, Fairfield, IA

Contents

Chapter 1

Wholeness of Big Data

Learning Objectives

- Understand Big Data and its powerful business benefits
- Learn the differences between Big Data and Conventional Data
- Learn the 4 Vs of Big Data – Volume, Velocity, Variety, and Veracity
- Discover three types of business applications of Big Data
- Learn valuable tips on how to manage Big Data
- Conceptualize the Big Data ecosystem and appreciate its key components
- Appreciate major technological challenges in managing Big Data
- Relate to the key technology solutions for addressing those challenges

Introduction

Big Data is an all-inclusive term that refers to extremely large, very fast, highly diverse, and complex data that cannot be managed with traditional data management tools.

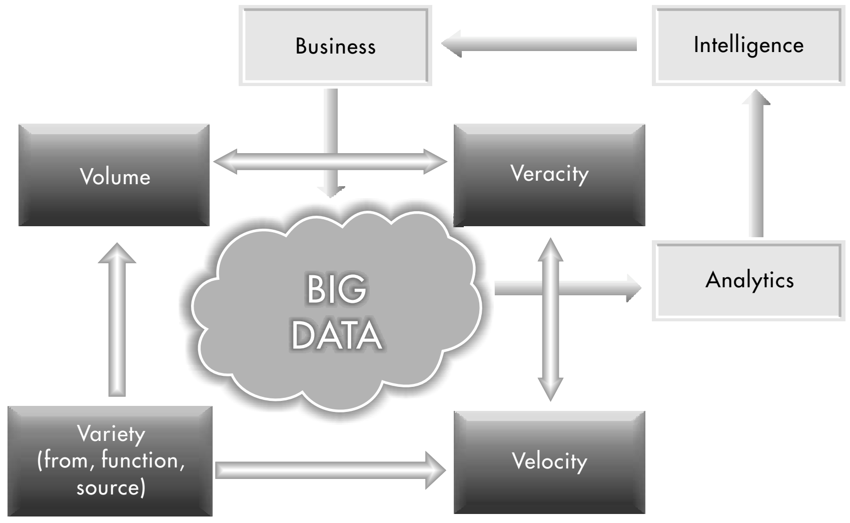


FIGURE 1.1 Big data context

Ideally, Big Data includes all kinds of data, which helps deliver the right information, to the right person, in the right quantity, at the right time, to help make the right decisions. Big Data can be harnessed by developing infinitely scalable, flexible, and evolutionary data architectures, coupled with the use of cost-effective computing machines. The infinite potential knowledge embedded within this Big Data cosmic computer would help connect with and enjoy the support of all the laws of nature.

This book will provide a complete overview of Big Data for executives and data scientists. This chapter will cover the key challenges and benefits of Big Data, and the essential tools and technologies available for organizing and manipulating Big Data.

1.1 UNDERSTANDING BIG DATA

Big Data can be examined at two levels (Figure 1.1). At a fundamental level, it is just another collection of data that can be analyzed and utilized for the benefit of the business. On another level, it is a special kind of data that poses unique challenges and offers unique benefits. This book will focus on the latter level.

At business level, data generated by business operations, can be analyzed to generate insights that can help the business make better decisions. This makes the business grow bigger, and generate even more data, so that the cycle continues. This is represented by the cycle on the top-right of Figure 1.1. This aspect is discussed in Chapter 10, 'A Primer on Data Analytics'.

On the other level, Big Data is different from traditional data in every way, i.e. space, time, and function. Quantitatively, Big Data is 1000 times more than the traditional data and of the data generation and transmission speed is also of the same order. The forms and functions of Big Data are 10 times more diverse: from numbers to text, pictures, audio, videos, web logs, machine data, and more. There are many more sources of data, from individuals to organizations to governments, using a range of devices from mobile phones to computers to industrial machines. Not all Big Data will be of equal quality and value, as represented by the cycle on the bottom left of Figure 1.1. This aspect of Big Data, and its new technologies, is the focus of this book.

Big Data is mostly, over 90 per cent, unstructured data. Every type of Big Data is structured differently, and should be dealt in with an appropriate way. There are huge opportunities for technology providers to innovate and manage the entire life cycle of Big Data; to generate, gather, store, organize, analyze, and visualize this data.

Caselet

IBM Watson: A Big Data System

IBM created the Watson system as a way of pushing the boundaries of Artificial Intelligence and natural language understanding technologies. Watson beat the world champion human players of Jeopardy (quiz style TV show) in Feb 2011. Watson reads up data about everything on the web including the entire Wikipedia. It digests and absorbs the data based on simple generic rules such as: books have authors; stories have heroes; and drugs treat ailments. A Jeopardy clue, received in the form of a cryptic phrase, is broken down into many possible potential sub-clues of the correct answer. Each sub-clue is examined to see the likeliness of its answer being the correct answer for the main problem. Watson calculates the confidence level of each possible answer. If the confidence level reaches more than a desirable threshold level, Watson decides to offer the answer to the clue. It manages to do all this in a mere 3 seconds.

Watson is now being applied to many important applications such as diagnosing diseases, especially cancer. Watson can read all the new research published in the medical journals to update its knowledge base. It can diagnose the probability of various diseases, by applying factors such as patient's current symptoms, health history, genetic history, medication records, and other factors to recommend a particular diagnosis. (*Source:* Smartest machines on Earth: youtube.com/watch?v=TCOhyaw5bwg)

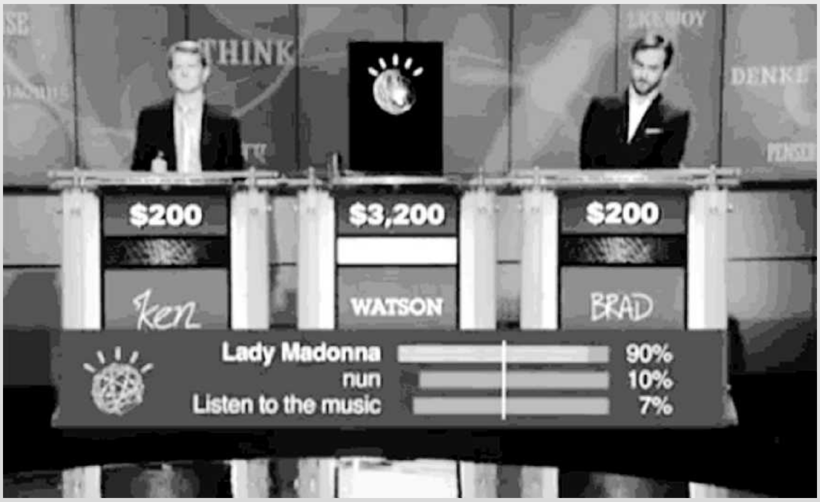


FIGURE 1.2 IBM Watson playing jeopardy

1. What kinds of Big Data knowledge, technologies and skills are required to build a system like Watson? What other resources are needed?
2. Will doctors be able to compete with Watson in diagnosing diseases and prescribing medications? Who else could benefit from a system like Watson?

1.2 CAPTURING BIG DATA

If data were only growing too large, or only moving too fast, or only becoming too diverse, it would have been relatively easy. However, when three Vs (Volume, Velocity, Variety) arrive together in an interactive manner, it creates a perfect storm. While the Volume and Velocity of data drive the major technological concerns and the costs of managing Big Data, these two Vs are themselves being driven by the third V, that is, the Variety of forms and functions and sources of data. The varying veracity and value of data complicate the situation further.

1.2.1 Volume of Data

The quantity of data generated in the world has been relentlessly doubling every 12–18 months. Traditional data is measured in Gigabytes (GB) and Terabytes (TB), but Big Data is measured in Petabytes (PB) and Exabytes (1 Exabyte = 1 Million TB). This data is so huge that it is almost a miracle that one can find any specific thing in it, in a reasonable period of time. Searching the world-wide web was the first true Big Data application. Google perfected the art of this application and developed many of the path-breaking Big Data technologies we see in use today.

The primary reason for the growth of data is the dramatic reduction in the cost of storing data. The costs of storing data have decreased by 30–40 per cent every year. Therefore, there is an incentive to record everything that can be observed. It is called 'datafication' of the world. The costs of computation and communication of data have also been coming down. Another reason for the growth of data is the increase in the number of forms and functions of data. We will discuss more about this in the 'Variety' section.

1.2.2 Velocity of Data

If traditional data is like a lake, Big Data is like a fast-flowing river. Big Data is being generated by billions of devices, and communicated at the speed of the light, through the internet. Ingesting all this data is like drinking from a fire hose. One does not have any control over how fast the data will come. A huge unpredictable data-stream is the new metaphor for thinking about Big Data.

The primary reason for the increased velocity of data is the increase in internet speed. Internet speeds available to homes and offices are now increasing from 10 MB/s to 1 GB/s (100 times faster). More people are getting access to high-speed internet around the world. Another important reason is the increased variety of sources, such as mobile devices, that can generate and communicate data from anywhere, at any time.

1.2.3 Variety of Data

Big Data is inclusive of all forms of data, for all kinds of functions, from all sources and devices. If traditional data forms such as invoices and ledgers were like a small store, Big Data is the biggest imaginable shopping mall that offers unlimited variety. There are three major kinds of variety of data.

1. The *first* aspect of variety is the **Form** of data. Data types range in variety from numbers to text, graph, map, audio, video, and others, with some being simple and others being very complex. There could be composites of data that include many elements in a single file. For example, text documents have graphs and pictures embedded in them. Video movies can have audio songs embedded in them. Audio and video have different and vastly more complex storage formats than numbers and text. Numbers and text can be more easily analyzed than an audio or video file. How should composite entities be stored and analyzed?
2. The *second* aspect of variety is the **Function** of data. There is data from human conversations, songs and movies, business transaction records, machine operations performance data, new product design data, old archived data, etc. Human communication data would need to be processed very differently from operational performance data, with different expectations and objectives. Big Data technologies could be used to recognize people's faces in pictures; compare voices to identify the speaker; and compare handwritings to identify the writer.
3. The *third* aspect of variety is the **Source** of data. Mobile phone and tablet devices enable a wide series of applications (or apps) to access data and, also, generate data from anytime anywhere. Web access and search logs are another new and huge source of data. Business systems generate massive amount of structured business transactional information. Temperature and pressure sensors on machines, and Radio Frequency (RFID) tags on assets, generate incessant and repetitive data. Broadly speaking, there are three broad types of sources of data: Human-human communications; human-machine communications; and machine-to-machine communications. The sources of Big Data, and their business applications, will be discussed in the next chapter.

1.2.4 Veracity of Data

Veracity relates to the truthfulness, believability, and quality of data. Big Data is messy and there is considerable misinformation and disinformation out there. The reasons for poor quality of data can range from technical error, to human error, to malicious intent.

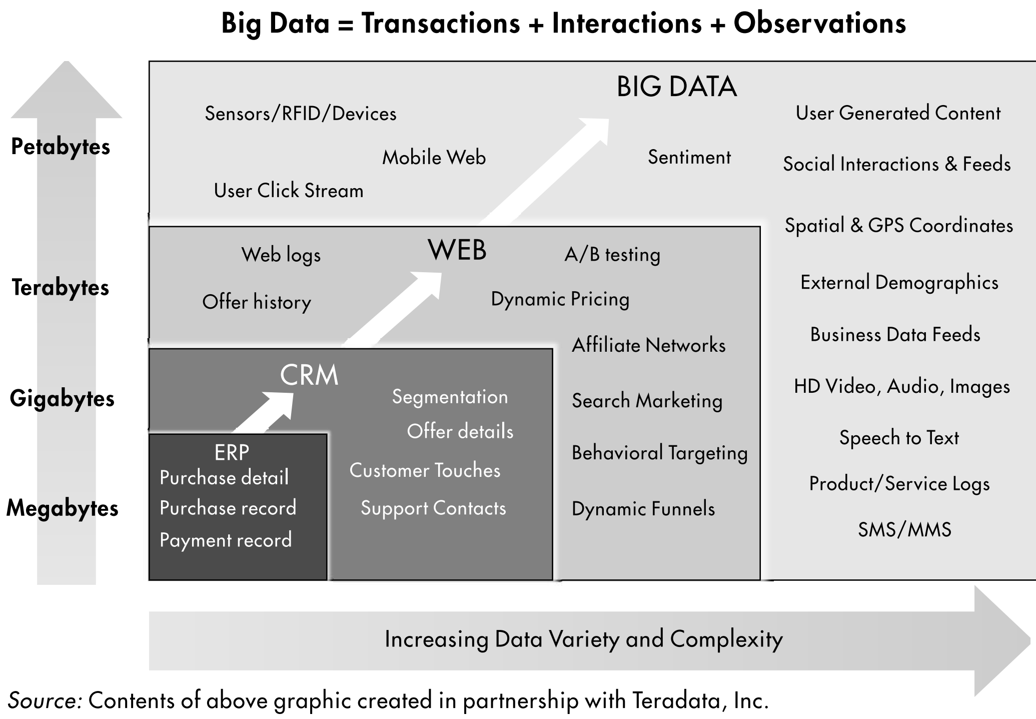


FIGURE 1.3 Sources of big data (*Source:* Hortonworks.com)

1. The source of information may not be authoritative. For example, all websites are not equally trustworthy. Any information from whitehouse.gov or from nytimes.com is more likely to be authentic and complete. Wikipedia is useful, but not all pages are equally reliable. In each case, the communicator may have an agenda or a point of view.
2. The data may not be communicated and received correctly because of human or technical failure. Sensors and machines for gathering and communicating data may malfunction and may record and transmit incorrect data. Urgency may require the transmission of the best data available at a point in time. Such data makes reconciliation with later, accurate, records more problematic.
3. The data provided and received may, however, also be intentionally wrong for competitive or security reasons. There could be disinformation and malicious information spread for strategic reasons.

Big Data need to be sifted and organized by quality, for it to be put to any great use.

1.3 BENEFITTING FROM BIG DATA

Data is the new natural resource. Traditionally, data usually belongs to the organization that generates it. There are other types of data, such as social media data, which are freely accessible under an open general license. Organizations can use this data to learn about their consumers, improve their service delivery, and design new products to delight their customers, and to gain a competitive advantage. Data is also being used to design new digital products, such as on-demand entertainment and learning.

Organizations cannot afford to ignore Big Data. They may choose to gather and store this data for later analysis, or to sell it to other organizations, that might benefit from it. They may also legitimately choose to discard parts of their data for privacy or legal reasons. However, organizations that do not learn to engage with Big Data, could find themselves left far behind their competition, landing in the dustbin of history. Innovative and nimble small organizations can use Big Data to quickly scale up and beat larger and more mature organizations.

Big Data applications exist in all industries and aspects of life. There are three major types of Big Data applications: Monitoring and Tracking, Analysis and Insight, and Digital Product Development.

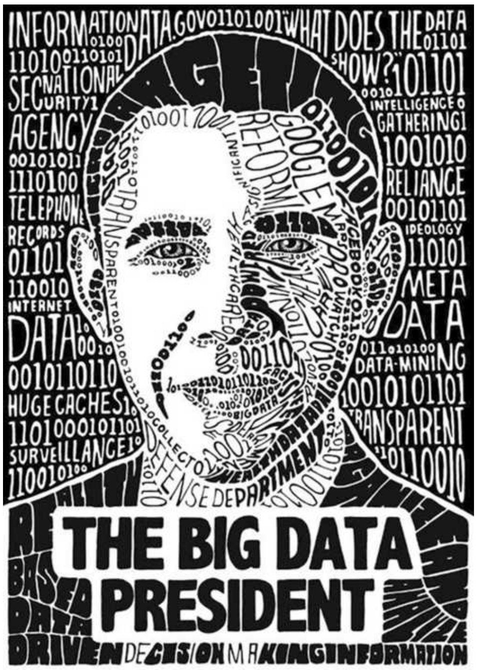


FIGURE 1.4 The first big data president

Monitoring and Tracking Applications Consumer goods producers use monitoring and tracking applications to understand the sentiments and needs of their customers. Industrial organizations use Big Data to track inventory in massive interlinked global supply chains. Factory owners use it to monitor machine performance and do preventive maintenance. Utility companies use it to predict energy consumption, and manage demand and supply. Information Technology companies use it to track website performance and improve its usefulness. Financial organizations use it to project trends better and make more effective and profitable bets, etc.

Analysis and Insight Political organizations use Big Data to micro-target voters and win elections. Police use Big Data to predict and prevent crime. Hospitals use it to better diagnose diseases and make medicine prescriptions. Advertising agencies use it to design more targeted marketing campaigns more quickly. Fashion designers use it to track trends and create more innovative products.

New Product Development Incoming data could be used to design new products such as reality TV entertainment. Stock market feeds could be a digital product. Imagination is the limit on how new such products and services can be developed and delivered at the speed of thought.

1.4 MANAGEMENT OF BIG DATA

Big Data is a new phase representing a digital world. Business and society are not immune from its strong impacts. Many organizations have started initiatives around the use of Big Data. However, most organizations do not necessarily have a good grip on it. Here are some emerging insights into making better use of Big Data.

1. Across all industries, the business case for Big Data is strongly focused on addressing customer-centric objectives. The first focus on deploying Big Data initiatives is to protect and enhance customer relationships and customer experience.
2. Big Data should be used to solve a real pain-point. It should be deployed for specific business objectives of management while avoiding being overwhelmed by the enormity of Big Data.
3. Organizations are beginning their pilot implementations by using existing and newly accessible internal sources of data. It is better to begin with data under one's control and where one has a superior understanding of the data. One should use more diverse data, and not just more data. This would permit tuning into a broader perspective of reality, leading to better insights.

4. Putting humans and data together leads to the most insights. Combining data-based analysis with human intuition and perspectives is better than going just one way. Most organizations lack the advanced analytical capabilities required to get the most value from Big Data. There is a growing awareness of building or hiring those skills and capabilities.
5. The faster one analyzes the data, the more its predictive value. The value of data depreciates rapidly with time. Many kinds of data need to be processed within minutes, or else the immediate competitive advantage is lost.
6. One should not throw away data if no immediate use can be seen for it. Data usually has value beyond what one initially anticipates. Data can add perspective to other data later in a multiplicative manner. One should maintain just one copy of your data, not multiple. This would help avoid confusion and increase efficiency.
7. Big Data is growing exponentially, so one should plan for exponential growth. Storage costs continue to fall, data generation continues to grow, data-based applications continue to grow in capability and functionality.
8. Big Data builds upon a resilient, secure, efficient, flexible, and real-time information processing environment. Big Data is transforming business, just like IT did.

1.5 ORGANIZING BIG DATA

Good organization depends upon the purpose of the organization.

Given huge quantities of data, it would be desirable to organize the data to speed up the search process for finding a specific desired thing in the entire data. The cost of storing and processing the data, too, would be a major driver for the choice of an organizing pattern.

Given fast and variable speed of data, it would be desirable to create a scalable number of ingest points. It will also be desirable to create at least some element of control over the data by maintaining count and averages over time, unique values received, etc.

Given wide variety in form factors, data will need to be stored and analyzed differently. Videos need to be stored separately and used for serving in a streaming mode. Text data may be stored separately, so it can be combined, cleaned, and analyzed for themes and sentiments.

Given different quality of data, various data sources may need to be ranked and prioritized before serving them to the audience. For example, the quality of a webpage and its data may be evaluated using its PageRank value.

1.6 ANALYZING BIG DATA

Big Data can be analyzed in two ways. Big Data can be utilized to visualize a flowing or a static situation. These are called analyzing Big Data in motion AND Big Data at rest. First way is to process the incoming stream of data in real time for quick and effective statistics about the data. The second way is to store and structure batches of data and apply standard analytical techniques on for generating insights. This could then be visualized using real-time dashboards. The nature of processing this huge, diverse, and largely unstructured data, can be limited only by one's imagination.

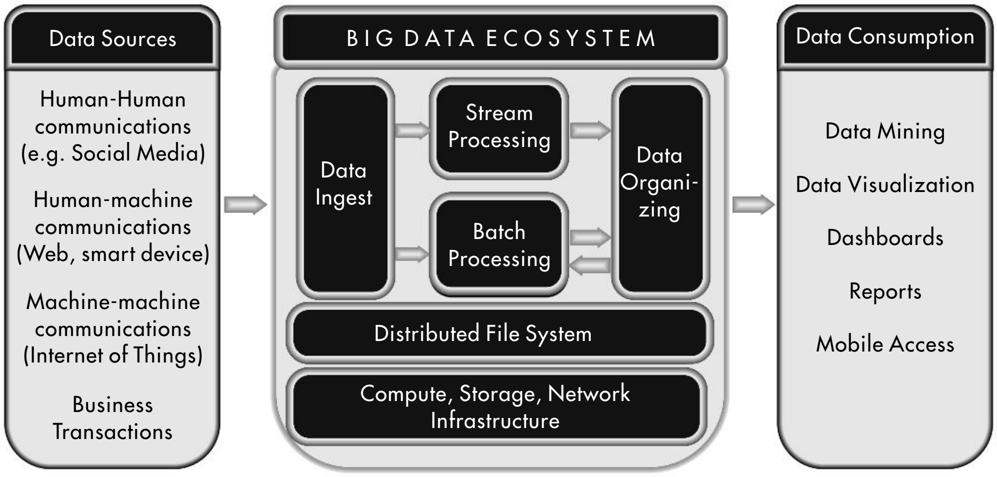
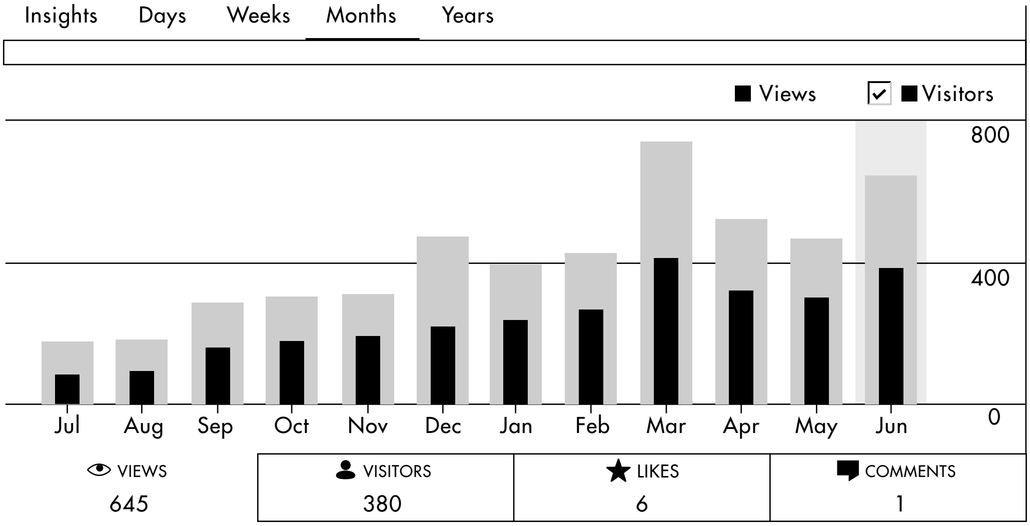


FIGURE 1.5 Big data architecture

A million points of data can be plotted in a graph and offer a view of the density of data. However, plotting a million points on the graph may produce a blurred image which may hide, rather than highlight the distinctions. In such a case, binning the data would help, or selecting the top few frequent categories may deliver greater insights. Streaming data can also be visualized by simple counts and averages over time. For example, below is a dynamically updated chart that shows up-to-date statistics of visitor traffic to the author's blogsite, anilmah.com. The bar shows the number of page views, and the inner darker bar shows the number of unique visitors. The dashboard could show the view by days, weeks, or years also.



VIEWS	VISITORS	LIKES	COMMENTS
645	380	6	1

FIGURE 1.6 Real-time dashboard for website performance for the author's blog

Text Data could be combined, filtered, cleaned, thematically analyzed, and visualized in a wordcloud. Here is wordcloud from a recent stream of tweets (i.e. Twitter messages) from 2016 US Presidential candidates Hillary Clinton and Donald Trump (Figure 1.7). The larger size of words implies greater frequency of occurrence in the tweets. This can help understand the major topics of discussion between the two.

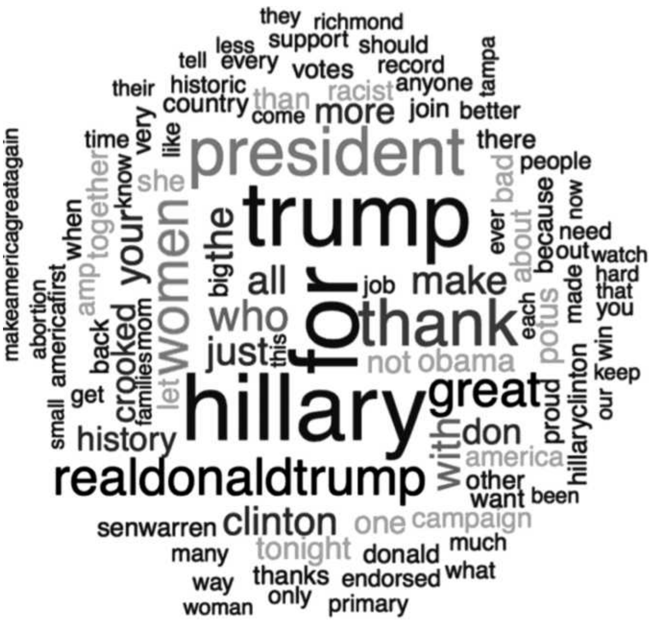


FIGURE 1.7 A wordcloud of Hillary Clinton's and Donald Trump's tweets

1.7 TECHNOLOGY CHALLENGES FOR BIG DATA

There are four major technological challenges in managing Big Data. There are four matching layers of technologies to overcome those challenges.

1.7.1 Storing Huge Volumes

The first challenge relates to storing huge quantities of data. No computing machine is big enough to store the relentlessly growing quantity of data. Therefore, data must be stored in several smaller inexpensive machines. However, with a large number of machines, there is the inevitable challenge of machine failure. Each of these commodity machines will fail at some point or another. Failure of a machine could entail a loss of data stored on it.

The first layer of Big Data technology thus helps store huge volumes of data, at an affordable cost, while avoiding the risk of data loss. It distributes data across a large cluster of inexpensive commodity machines. It ensures that every piece of data is stored on multiple machines to guarantee that at least one copy is always available.

Hadoop is the most well-known clustering technology for Big Data. Its data storage pattern is called Hadoop Distributed File System (HDFS). This system is built on the patterns of Google's Big File systems, designed to store billions of pages, and sort them to answer user search queries.

1.7.2 Ingesting Streams at an Extremely Fast Pace

The second challenge relates to the Velocity of data, i.e. handling torrential streams of data. Some of the data streams may be too large to store, but must still be monitored. The solution lies in creating special scalable ingesting systems that can open an unlimited number of channels for receiving data. These systems can hold data in queues, from which business applications can read and process data at their own pace and convenience.

The second layer of Big Data technology manages this velocity challenge. It uses a special stream-processing engine, where all incoming data is fed into a central queueing system. From there, a fork-shaped system sends data to batch storage as well as to stream processing directions. The stream processing engine can do its work while the batch processing does its work. Apache Spark is the most popular system for streaming applications.

1.7.3 Handling a Variety of Forms and Functions of Data

The third challenge relates to the structuring and access of all varieties of data that comprise Big Data. Storing them in traditional flat or relational structures would be too impractical, wasteful, and slow. Accessing and analyzing them requires different capabilities.

The third layer of Big Data technology solves this problem by storing the data in non-relational systems that relax many of the stringent conditions of the relational model. These are called NoSQL (Not Only SQL) databases. These databases are optimized for certain tasks such as query processing, or graph processing, document processing, etc.

HBase and Cassandra are two of the better known NoSQL databases systems. HBase, for example, stores each data element separately along with its key identifying information. This is called a key-value pair format. Cassandra stores data in a columnar format. There are many other variants of NoSQL databases. NoSQL languages, such as Pig and Hive, are used to access this data.

1.7.4 Processing Data at Huge Speeds

The fourth challenge relates to moving large amounts of data from storage to the processor, as this would consume enormous network capacity and choke the network. The alternative and innovative mode would be to do just the opposite, i.e. to move the processing to where the data is stored.

The fourth layer of Big Data technology avoids the choking of the network. It distributes the task logic throughout the cluster of machines where the data is stored. Those machines work, in parallel, on the data assigned to them, respectively. A follow-up process consolidates the outputs of all the small tasks and delivers the final results.

MapReduce, also invented by Google, is the best-known technology for parallel processing of distributed Big Data. Table 1.1 lists technological challenges and solutions for Big Data.

Table 1.1

Technological challenges and solutions for big data

Challenge	Description	Solution	Technology
Volume	Avoid risk of data loss from machine failure in clusters of commodity machines	Replicate segments of data in multiple machines; master node keeps track of segment location	HDFS

(*Contd.*)

(C*ontd.*)

Challenge	Description	Solution	Technology
Volume & Velocity	Avoid choking of network bandwidth by moving large volumes of data	Move processing logic to where the data is stored; manage using parallel processing algorithms	Map-Reduce
Variety	Efficient storage of large and small data objects	Columnar databases using key-pair values format	HBase, Cassandra
Velocity	Monitoring streams too large to store	Fork-shaped architecture to process data as stream and as batch	Spark

Once these major technological challenges arising from the 4 Vs of data are met, all traditional analytical and presentation tools can be applied to Big Data. There are many additional supportive technologies to make the task of managing Big Data easier. For example, there are technologies to monitor the resource usage and load balancing of the machines in the cluster.

1.8 CONCLUSION

Big Data is a major social and technological phenomenon that impacts everyone. It also provides an opportunity to create new ways of knowing and working. Big Data is extremely large, complex, fast, and not always clean as it is data that comes from many sources such as people, web, and machine communications. Big Data needs to be gathered, organized and processed in a cost-effective way that manages the volume, velocity, variety, and veracity of Big Data. Hadoop, MapReduce, NoSQL, and Spark systems are popular technological platforms for this purpose.

To summarize, here is a list of the many differences between traditional and Big Data (Table 1.2).

Table 1.2

Comparing big data with traditional data

Feature	Traditional Data	Big Data
Representative Structure	Lake/Pool	Flowing Stream/river
Primary Purpose	Manage business activities	Communicate, Monitor
Source of data	Business transactions, documents	Social media, Web access logs, machine generated
Volume of data	Gigabytes, Terabytes	Petabytes, Exabytes
Velocity of data	Ingest level is controlled	Real-time unpredictable ingest
Variety of data	Alphanumeric	Audio, Video, Graphs, Text

(C*ontd.*)

(C*ontd.*)

Feature	Traditional Data	Big Data
Veracity of data	Clean, more trustworthy	Varies depending on source
Structure of data	Well-Structured	Semi- or Un-structured
Physical Storage of Data	In a Storage Area Network	Distributed clusters of commodity computers
Database organization	Relational databases	NoSQL databases
Data Access	SQL	NoSQL such as Pig
Data Manipulation	Conventional data processing	Parallel processing
Data Visualization	Variety of tools	Dynamic dashboards with simple measures
Database Tools	Commercial systems	Open-source - Hadoop, Spark
Total Cost of System	Medium to High	high

1.9 ORGANIZATION OF THE REST OF THE BOOK

This book will cover applications, architectures, and the essential Big Data technologies. The rest of the book is organized as follows.

Section 1 will discuss sources, applications, and architectural topics. Chapter 2 will discuss a few compelling business applications of Big Data, based on understanding the different sources and formats of data. Chapter 3 will cover some examples of Big Data architectures used by leading organizations.

Section 2 will discuss expand on the Big Data Ecosystem (Figure 1.5). It will have six chapters discussing the six major technology elements identified in the ecosystem. Chapter 4 will discuss Hadoop ecosystem and how its Distributed File system (HDFS) works. Chapter 5 will discuss MapReduce parallel processing algorithm and how it helps accomplish big tasks quickly. An introduction to Hive and Pig programming languages will also be included. Chapter 6 will discuss NoSQL databases to learn how to structure Big Data into four major types of databases. In particular, HBase and Cassandra databases will be covered in great detail. Chapter 7 will cover streaming data, and the systems for ingesting and processing this data. This chapter will cover Spark, an integrated, in-memory processing toolset to manage Big Data. Chapter 8 will cover Data ingest system, using Apache Kafka messaging system. Finally, Chapter 9 will be a primer on Cloud Computing technologies used for renting storage and compute capacity at third party networks at a fraction of the cost of in-house IT infrastructure.

Section 3 will include applications, primers, and tutorials. Chapter 10 will present a case study on "web log analyzer", a popular and generic application that ingests a log of a large number of web request entries every day and can create summary and exception reports. Chapter 11 will be a primer on data analytics technologies for analyzing data. A full treatment can be found in my book, *Data Analytics*. Chapter 12 will be a primer on Big Data languages, Apache Hive and Pig. Appendix A will be a tutorial on installing Hadoop cluster on Amazon EC2 cloud. Appendix B will be a tutorial on installing and using Spark.

Review Questions

1. What is Big Data? Why should anyone care?
2. Describe the 4V model of Big Data.
3. What are the major technological challenges in managing Big Data?
4. What are the technologies available to manage Big Data?
5. What kind of analyses can be done on Big Data?
6. Watch Cloudera CEO present the evolution of Hadoop at https://www.youtube.com/watch?v=S9xnYBVqLws . Why did people not pay attention to Hadoop and MapReduce when it was introduced? What implications does it have to emerging technologies?

True/False Questions

1. Big Data is an umbrella term for a collection of data sets so large and complex that they do not fit into a single file system.
2. 'Datafication' means that 'increasing' details about events are being recorded and stored.
3. Data only belongs to the user or the organization that generates it.
4. Big Data is so precious that multiple copies of data should be maintained.
5. Big Data includes approximately equal amounts of structured data and unstructured data.
6. Social media interactions are a form of big data.
7. Big Data is growing steadily at an exponential rate.
8. Big Data at rest means to process the incoming stream of data in real time for quick and effective statistics about the data.
9. The major causes of variety of data are the source, form, and function of data.

10. Transactions + Interactions + Observations = Traditional Data.
11. The primary cause of poor quality of data is because of human error.
12. There are five different categories of Big Data applications.
13. Big Data should be used for solving technology-centric applications.
14. Visualizing Big Data presents no special challenges.
15. Storing huge volumes of data is solved using MapReduce.
16. Processing massive data in parallel is solved using NoSQL databases.
17. Processing massive data in real-time streaming mode is done through Apache Spark.
18. Managing huge variety of data is done through NoSQL databases.
19. Hadoop is a proprietary Big data technology.
20. Hadoop Distributed File System (HDFS) was invented by Google.

Liberty Stores Case Exercise: Step B1

Liberty Stores Inc. is a specialized global retail chain that sells organic food, organic clothing, wellness products, and education products to enlightened LOHAS (Lifestyles of the Healthy and Sustainable) citizens worldwide. The company is 20 years old, and is growing rapidly. It now operates in 5 continents, 50 countries, 150 cities, and has 500 stores. It sells 20000 products and has 10000 employees. The company has revenues of over $5 billion and has a profit of about 5 per cent of its revenue. The company pays special attention to the conditions under which the products are grown and produced. It donates about one-fifth (20 per cent) from its pre-tax profits from global local charitable causes.

1. Suggest a comprehensive Big Data strategy for the CEO of the company.
2. How can a Big Data system such as IBM Watson help this company?

Section 1

This section covers two important high-level topics.

- **Chapter 2** will cover big data sources, and its varied applications in many industries.
- **Chapter 3** will cover examples of Big Data architectures from leading organizations.

Chapter 2

Big Data Sources and Applications

Learning Objectives

- Recognize the three categories of sources of Big Data
- Identify the three categories of applications of Big Data
- Appreciate how Internet of Things (IoT) is driving Big Data
- Understand how analyzing and interpreting Big Data requires caution

Introduction

If a traditional software application is a lovely cat, then a Big Data application is a fierce tiger. An ideal Big Data application will take advantage of all the richness of data and produce relevant information to make the organization responsive and successful. Big Data applications can align the organization with the totality of natural laws, the source of all sustainable success in the world (Figure 2.1).

Companies like the consumer goods giant, Proctor & Gamble, have inserted Big Data into all aspects of its planning and operations. The industrial giant, Volkswagen, requires all its business units to identify some realistic initiative using Big Data to grow their unit's sales. The entertainment giant, Netflix, processes 400 billion user actions every day to understand their customers' needs.

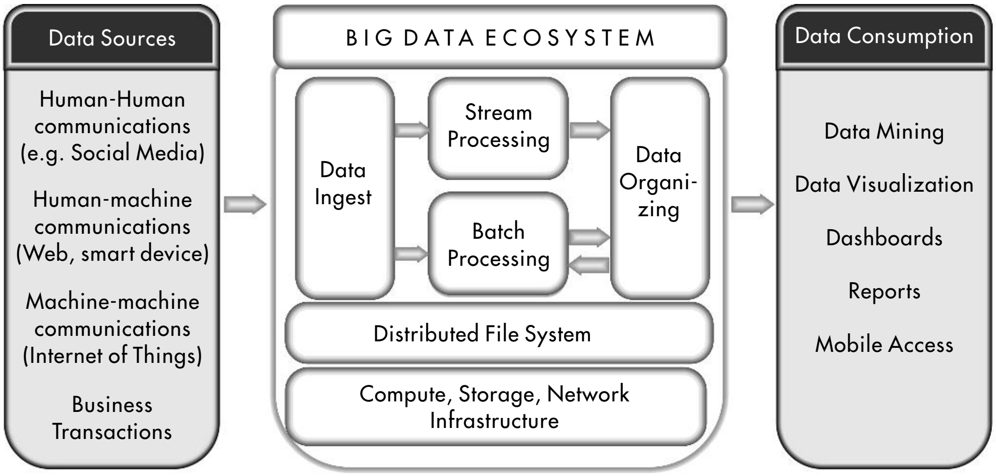


FIGURE 2.1 Big data architecture

Caselet

Big Data Gets the Flu

Google Flu Trends was an enormously successful influenza forecasting service, pioneered by Google. It employed Big Data, such as the stream of search terms used in its ubiquitous Internet search service. The program aimed to better predict flu outbreaks using data and information from the U.S. Centers for Disease Control and Prevention (CDC). What was most amazing was that this application could predict the onset of flu, almost two weeks before CDC saw it coming. From 2004 till about 2012 it was able to successfully predict the timing and geographical location of the arrival of the flu season around the world.

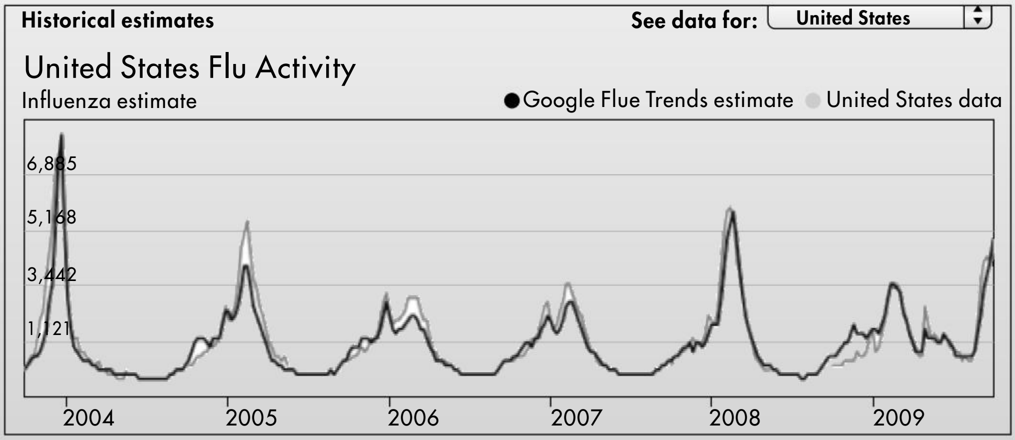


FIGURE 2.2 Google flu trends

However, it failed spectacularly to predict the 2013 flu outbreak. Data used to predict Ebola's spread in 2014–15 yielded wildly inaccurate results, and created a major panic. Newspapers across the globe spread this application's worst-case scenarios for the Ebola outbreak of 2014.

Google Flu Trends failed for two reasons: Big Data hubris, and algorithmic dynamics, (a) The quantity of data does not mean that one can ignore foundational issues of measurement and construct validity and reliability and dependencies among data and (b) Google Flu Trends predictions were based on a commercial search algorithm that frequently changes, based on Google's business goals. This uncertainty skewed the data in ways even Google engineers did not understand, even skewing the accuracy of predictions. Perhaps the biggest lesson is that there is far less information in the data, typically available in the early stages of an outbreak, than is needed to parameterize the test models.

1. What lessons would you learn from the death of a prominent and highly successful Big Data application?
2. What other Big Data applications can be inspired from the success of this application?

2.1 BIG DATA SOURCES

Big Data is inclusive of all data, about all activities, everywhere. It can thus potentially transform our perspective on life and the universe. It can bring new insights in real-time, make life happier, and make the world more productive. Big Data can however also bring perils—in terms of violation of privacy, and social and economic disruption.

There are three major categories of data sources: people-to-people communications, human-machine communications, and machine-machine communications.

2.1.1 People-to-People Communications

People and corporations increasingly communicate instantly over electronic networks. Everyone communicates through phone and email. Influential networks have expanded. Distance and time have been annihilated. News travels instantly. The content of communication has become richer and multimedia. For example, high-resolution cameras in mobile phones enable people to record pictures and videos, and instantly share them with friends and family. All these communications are stored in the facilities of many intermediaries, such as telecom and internet service providers. Social media is a new and particularly transformative type of human-human communications.

2.1.1.1 Social Media

Social media platforms such as Facebook, Twitter, LinkedIn, YouTube, Flickr, Tumblr, Skye, Snapchat, and others have become an increasingly intimate part of modern life.

People instantly share messages and pictures through social media such as Facebook and YouTube. They share photo albums through Flickr. They communicate in short asynchronous messages with each other on Twitter. They make friends on Facebook, and follow others on Twitter. They do video conferencing, using Skype and leaders deliver messages that sometimes go viral through social media.

FIGURE 2.3 Sampling of major social media

Figure 2.3 shows a few among the hundreds of social media that people use and they generate huge streams of text, pictures, videos, logs, and other multimedia data. All these data streams are part of Big Data, and can be monitored and analyzed to understand many phenomena, such as patterns of communication, as well as the gist of the conversations. These media have been used for a wide variety of purposes with stunning effects.

2.1.2 People-to-Machine Communications

Sensors and web are two of the kinds of machines that people communicate with. Personal assistants such as Siri and Cortana are the latest in man–machine communications as they try to understand human requests in natural language, and fulfil them. Wearable devices such as FitBit and smart watch are smart devices that read, store and analyze people's personal data such as blood pressure and weight, food and exercise data, and sleep patterns. The world-wide web is like a knowledge machine that people interact with to get answers for their queries. They also use the web for commerce.

2.1.2.1 Web Access

The world-wide-web has integrated itself into all parts of human and machine activity. The usage of the tens of billions of pages by billions of web users generates huge amount of enormously valuable clickstream data. Every time a web page is requested, a record of this activity is generated at the webpage provider end. The webpage provider tracks the identity of the requesting device and user, and time and spatial location of each request. On the webpage requester side, there are certain small pieces of computer code and data called cookies which track the webpages received, date/time of access, and some identifying information about the user. All these web access logs, and cookie records, can provide web usage records that can be analyzed for discovering opportunities for marketing purposes.

A web log analyzer is an application required to monitor streaming web access logs in real-time to check on website health and to flag errors. A detailed case study of a practical development of this application is shown in Chapter 10.

2.2 MACHINE-TO-MACHINE (M2M) COMMUNICATIONS

M2M communications is also sometimes broadly called the Internet of Things (IoT). A trillion devices are connected to the internet and they communicate with each other, or with some master machines. All this data can be accessed and harnessed by makers and owners of those machines.

Machines and equipment have many kinds of sensors to measure certain environmental parameters, which can be broadcast to communicate their status. RFID tags and sensors embedded in machines help generate the data. Containers on ships are tagged with RFID tags that convey their location to all those who can listen. Similarly, when pallets of goods are moved in warehouses and retail stores, those pallets contain electromagnetic (RFID) tags that convey their location. Cars carry an RFID transponder to identify themselves to automated tollbooths and pay the tolls. Robots in a factory, and internet-connected refrigerators in a house, continually broadcast a ‘heartbeat’ that they are functionally normally. Automobiles contain sensors that record and communicate operational data. A modern car can generate many megabytes of data every day, and there are more than 1 billion motor vehicles on the road. Thus the automotive industry itself generate huge amounts of data. Self-driving cars would only add to the quantity of data generated. Surveillance videos using commodity cameras are another major source of machine-generated data.

2.2.1 RFID Tags

An RFID tag (Figure 2.4) is a radio transmitter with a little antenna that can respond to and communicate essential information to special readers through Radio Frequency (RF) channel. A few years ago, major retailers such as Walmart decided to invest in RFID technology to take the retail industry to a new level. It forced their suppliers to invest in RFID tags on the supplied products. Today, almost all retailers and manufacturers have implemented RFID-tags based solutions.

FIGURE 2.4 A small passive RFID tag

Here is how an RFID tag works. When a passive RFID tag comes in the vicinity of an RF reader and is 'tickled', the tag responds by broadcasting a fixed identifying code. An active RFID tag has its own battery and storage, and can store and communicate a lot more information. Every reading of message from an RFID tag by an RF reader creates a record, or a log entry. Thus there is a steady stream of data from every reader as it records information about all the RFID tags in its area of influence. The records may be logged regularly, to track the location and movement of an item. All the duplicate and redundant records must be removed to produce clean, consolidated data about the location and status of items in real time.

2.2.2 Sensors

A sensor (Figure 2.5) is a small physical device that can observe and record physical or chemical parameters. Sensors are all-pervasive. A photo sensor in the elevator or train door can sense if someone is moving and to thus keep the door from closing. A CCTV camera can record a video for surveillance purposes. A GPS device can record its geographical location every moment. Temperature sensors in a car can measure the temperature of the engine and the tires and more. The thermostat in a building or a refrigerator too have temperature sensors. A pressure sensor can measure the pressure inside an industrial boiler.

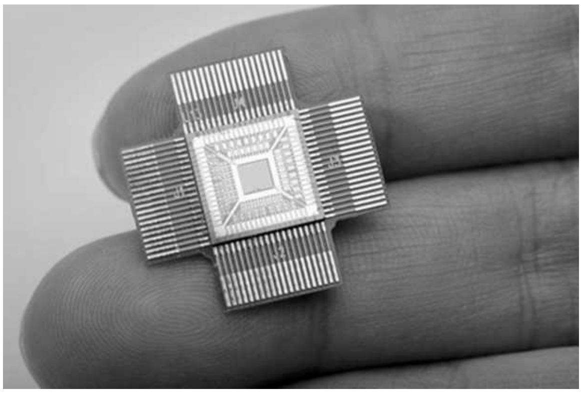


FIGURE 2.5 An embedded sensor

2.3 BIG DATA APPLICATIONS

Recording and processing all data requires much talent, resources, and time. The data can be use imaginatively and meaningfully to derive business benefit. There are three major types of business applications with different levels of transformational potential.

2.3.1 Monitoring and Tracking Applications

These are the first and basic applications of Big Data. They help improve the efficiency of the business, in almost all industries. Here are a few specific applications.

2.3.1.1 Public Health Monitoring

The US government is encouraging all healthcare stakeholders to establish a national platform for interoperability and data sharing standards. This would enable secondary use of health data, which would advance Big Data analytics and personalized holistic precision medicine. This would be a broad-based platform like Google Flu Trends.

2.3.1.2 Consumer Sentiment Monitoring

Social Media has become more powerful than advertising. Many consumer goods companies have moved a bulk of their advertising budgets from traditional media into social media. They have set up Big Data listening platforms (Figure 2.6), where Social Media data streams (including tweets and Facebook posts and blog posts) are filtered and analyzed for certain keywords or sentiments, by certain demographics and regions. Actionable information from this analysis is delivered to marketing professionals for appropriate action, especially when the product is new to the market.

2.3.1.3 Asset Tracking

The US Department of Defense is encouraging the industry to devise a tiny RFID chip that could prevent the counterfeiting of electronic parts that end up in avionics or circuit boards for other devices. Airplanes are one of the heaviest users of sensors which track every aspect of the performance of every part of the plane. The data can be displayed on the dashboard, as well as stored for later detailed analysis. Working with communicating devices, these sensors can produce a torrent of data.

Theft by shoppers and employees is a major source of loss of revenue for retailers. All valuable items in the store can be assigned RFID tags, and the gates of the store can be equipped with RF readers. This can help secure the products, and reduce leakage (theft) from the store.

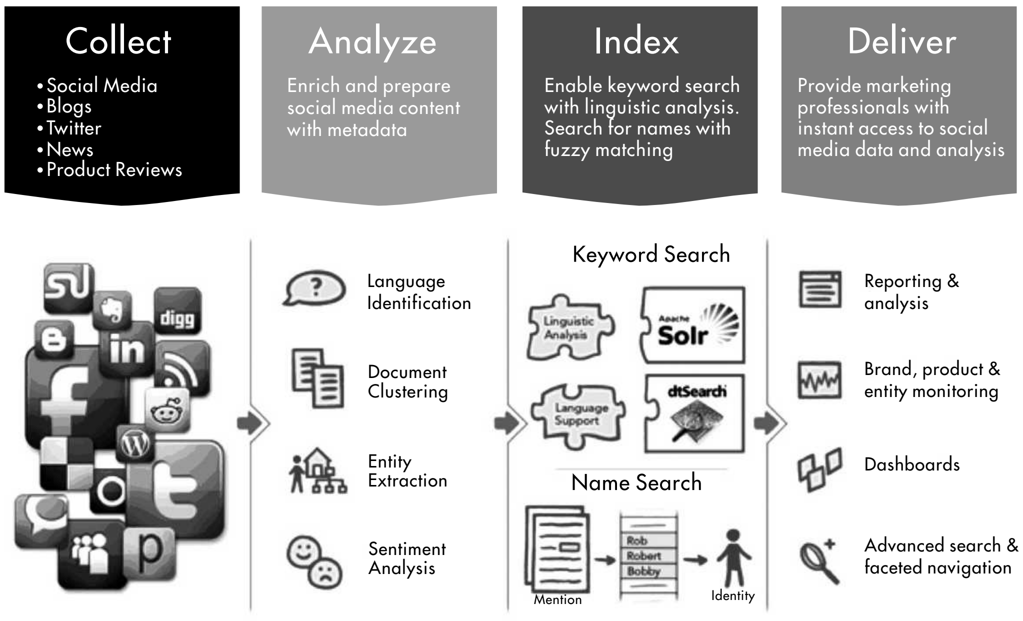


FIGURE 2.6 Architecture for a social media listening platform (*Source:* Intelligenthq.com)

2.3.1.4 Supply Chain Monitoring

All containers on ships communicate their status and location using RFID tags. Thus, retailers and their suppliers can gain real-time visibility to the inventory throughout the global supply chain. Retailers can know exactly where the items are in the warehouse, and so can bring them into the store at the right time. This is particularly relevant for seasonal items that must be sold on time, or else they will be sold at a discount. With item-level RFID tacks, retailers also gain full visibility of each item and can serve their customers better.

2.3.1.5 Electricity Consumption Tracking

Electric utilities can track the status of generating and transmission systems and, also, measure and predict the consumption of electricity. Sophisticated sensors can help monitor voltage, current, frequency, temperature, and other vital operating characteristics of huge and expensive electric distribution infrastructure. Smart meters can measure the consumption of electricity at regular intervals of one hour or less. This data is analyzed to make real-time decisions to maximize power capacity utilization and the total revenue generation.

2.3.1.6 Preventive Machine Maintenance

All machines, including cars and computers, do tend to fail sometimes. This is because one or more or their components may cease to function. As a preventive measure, precious equipment could be equipped with sensors. The continuous stream of data from the sensors could be monitored and analyzed to forecast the status of key components, and thus, monitor the overall machine's health. Preventive maintenance can, thus, reduce the cost of downtime.

2.3.2 Analysis and Insight Applications

These are next level of Big Data applications. They can increase the effectiveness of business and have transformative potential. Big Data can be structured and analyzed to produce insights and patterns that can be used to make business better.

2.3.2.1 Predictive Policing

The Los Angeles Police Department (LAPD) invented the concept of Predictive Policing. The LAPD worked with UC Berkeley researchers to analyze its large database of 13 million crimes recorded over 80 years, and predicted the likeliness of crimes of certain types, at certain times, and in certain locations. They identified hotspots of crime where crimes had occurred, and where crime was likely to happen in the future (Figure 2.7). Crime patterns were mathematically modelled after a simple insight borrowed from a metaphor of earthquakes and its aftershocks. The model said that once a crime occurred in a location, it represented a certain disturbance in harmony, and would thus, lead to a greater likelihood of a similar crime occurring in the local vicinity soon. The model showed for each police beat, the specific neighborhood blocks and specific time slots, where crime was likely to occur.

FIGURE 2.7 LAPD officer on predicting policing (*Source:* nbclosangeles.com)

By aligning the police cars' patrol schedules in accordance with the model's predictions, the LAPD could reduce crime by 12 per cent to 26 per cent for different categories of crime. Recently, the San Francisco Police Department released its own crime data for over 2 years, so data analysts could model that data and prevent future crimes.

2.3.2.2 Winning Political Elections

The US President, Barack Obama, was the first major political candidate to use Big Data in a significant way, in the 2008 elections. He is the first Big Data president. His campaign gathered data about millions of people, including supporters. They invented the mechanism to obtain small campaign contributions from millions of supporters. They created personal profiles of millions of supporters and what they had done and could do for the campaign. Data was used to determine undecided voters who could be converted to their side. They provided phone numbers of these undecided voters to the volunteers. The results of the calls were recorded in real time using interactive web applications. Obama himself used his twitter account to communicate his messages directly with his millions of followers.

After the elections, Obama converted his list of tens of millions of supporters to an advocacy machine that would provide the grassroots support for the President's initiatives. Since then, almost all campaigns use Big Data. Senator Bernie Sanders used the same Big Data playbook to build an effective national political machine powered entirely by small donors.

Elections analyst, Nate Silver, created sophistical predictive models using inputs from many political polls and surveys to win pundits to successfully predict winners of the US elections. Nate was however, unsuccessful in predicting Donald Trump's rise and ultimate victory, and that shows the limits of Big Data.

2.3.2.3 Personal Health

Correct diagnosis is the *sine qua non* of effective treatment. Medical knowledge and technology is growing by leaps and bounds. IBM's Watson system is a Big Data Analytics engine that ingests and digests all the medical information in the world, and then applies it intelligently to an individual situation. Watson can provide a detailed and accurate medical diagnosis using current symptoms, patient history, medication history, and environmental trends, and other parameters. Similar products might be offered as an App to licensed doctors, and even individuals, to improve productivity and accuracy in health care.

2.3.3 New Product Development

These applications are totally new concepts that did not exist earlier. These applications have the transformative potential to disrupt entire industries, and generate new avenues of revenue for businesses.

2.3.3.1 Flexible Auto Insurance

An auto insurance company can use the GPS data from cars to calculate the risk of accidents based on travel patterns (Figure 2.8). The automobile companies can use the car sensor data to track the performance of a car. Safer drivers can be rewarded and the errant drivers can be penalized.

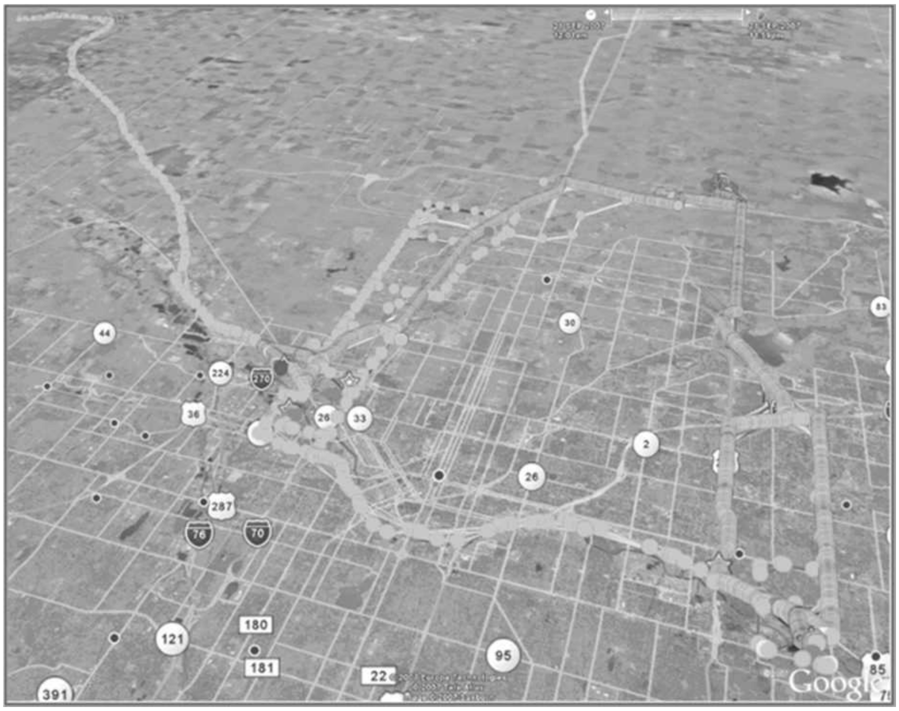


FIGURE 2.8 GPS based tracking of vehicles

2.3.3.2 Location-based Retail Promotion

A retailer, or a third-party advertiser, can target customers with specific promotions and coupons based on location data obtained through Global Positioning System (GPS) the time of day, the presence of stores nearby, and mapping it to the consumer preference data available from social media databases. Advertisements and offers can be delivered through mobile apps, SMS, and email. These are examples of mobile apps.

2.3.3.3 Recommendation Service

E-commerce has been a fast growing industry in the last couple of decades. A variety of products are sold and shared over the internet. Web users' browsing and purchase history on ecommerce sites is utilized to learn about their preferences and needs, and to advertise relevant product and pricing offers in real-time. Amazon uses a personalized recommendation engine system to suggest new additional products to consumers based on affinities of various products. Netflix also uses a recommendation engine to suggest entertainment options to its users.

2.4 CONCLUSION

Big Data is valuable across all industries. There are three major types of data sources of Big Data, viz., people-people communications, people-machine communications, and machine-machine communications. Each type has many sources of data. There are three types of applications. They are the monitoring type, the analysis type, and new product development. They have impact on efficiency, effectiveness, and even disruption of industries. This chapter presents a few business applications of each of those three types.

Review Questions

1. What are the major sources of Big Data? Describe a source of each type.
2. What are the three major types of Big Data applications? Describe two applications of each type.
3. Would it be ethical to arrest someone based on a Big Data Model's prediction of that person likely to commit a crime?
4. An auto insurance company learned about the movements of a person based on the GPS installed in the vehicle. Would it be ethical to use that as a surveillance tool?
5. Research can describe a Big Data application that has a proven return on investment (ROI) for an organization.

True/False Questions

1. One of the major source of Big Data is man-machine communications.
2. The more data is available, the most successful with be the business applications.

3. RFID tags are a form of man-machine communications.
4. Social media are a form of man-machine communications.
5. A sensor is a device that computes whether the social media message is sensible.
6. Tracking the location of assets can lead to great efficiencies.
7. Predictive policing has been shown to reduce crime.
8. Social media data can be used to design flexible auto insurance policies.
9. Big Data applications are relevant mostly for knowledge-intensive industries.
10. Big Data can be used for effective marketing campaigns.

Liberty Stores Case Exercise: Step B2

The Board of Directors asked the company to take concrete and effective steps to become a data-driven company. The company wants to understand its customers better. It wants to improve the happiness levels of its customers and employees. It wants to innovate on new products that its customers would like. It wants to relate its charitable activities to the interests of its customers.

1. What kind of data sources should the company capture for this?
2. What kind of Big Data applications would you suggest for this company?

Chapter 3

Big Data Architecture

Learning Objectives

- Appreciate the variety of architectures for Big Data ecosystem
- Explain the many layers of technologies in the Big Data architecture
- Analyse how leading organizations are architecting Big Data solutions

Introduction

Big Data Application Architecture is the configuration of tools and technologies to accomplish the whole task. An ideal Big Data architecture would be resilient, secure, cost-effective, and adaptive to new needs and environments. This can be achieved by beginning with a proven architecture, and creatively and progressively restructuring it as additional needs and problems arise. Big Data architectures should ultimately align with the architecture of the Universe itself, the source of all invincibility.

Caselet

Google Query Architecture

Google invented the first Big Data architecture. Their goal was to gather all the information on the web, organize it, and search it for specific queries from millions of users. An additional goal was to find a way to monetize this service by serving relevant and prioritized online advertisements on behalf of clients.

Google developed web crawling agents which would follow all the links in the web and make a copy of all the content on all the webpages it visited. Google invented cost-effective, resilient, and fast ways to store and process all that exponentially growing data. It developed a scale-out architecture in which it could linearly increase its storage capacity by inserting additional computers into its computing network. The data files were distributed over the large number of machines in the cluster. This distributed files system was called the Google File system, and was the precursor to Hadoop.

Google would sort or index the data thus gathered so it can be searched efficiently (Figure 3.1). They invented the key-pair NoSQL database architecture to store variety of data objects. They developed the storage system to avoid updates in the same place. Thus the data was written once, and read multiple times.

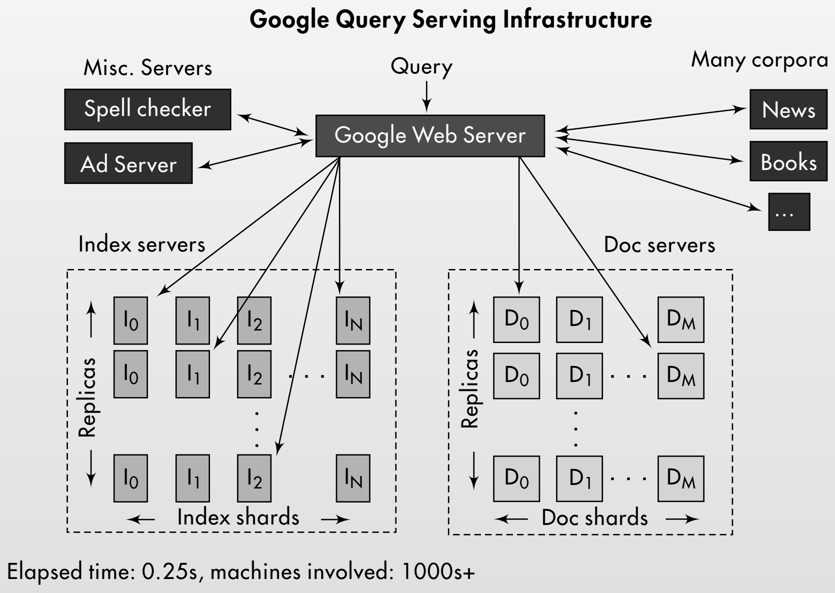


FIGURE 3.1 Google query architecture

Google developed the MapReduce parallel processing architecture whereby large datasets could be processed by thousands of computers in parallel, with each computer processing a chunk of data, to produce quick results for the overall job. The Hadoop ecosystem of data management tools like Hadoop distributed file system (HDFS), columnar database system like HBase, a querying tool such as Hive, and more, emerged from Google's inventions. Apache Spark is a streaming data technology to produce instant results.

1. Why should Google publish its File System and the MapReduce parallel programming system and send it into open-source system?
2. What else good can be done with Google's repository of the entire web's data?

3.1 STANDARD BIG DATA ARCHITECTURE

Here is the generic Big Data Architecture as introduced in Chapter 1 (shown again in Figure 3.2). There are many sources of data. All data is funneled in through an ingest system. The data is forked into two sides: a stream processing system and a batch processing system. The outcome of these processing can be sent into NoSQL databases for later retrieval, or sent directly for consumption by many applications and devices.

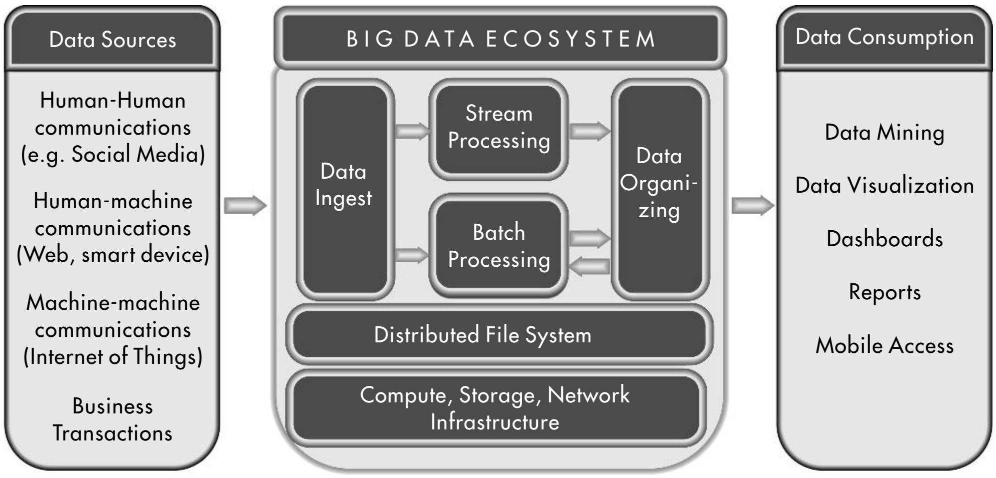


FIGURE 3.2 Generic big data architecture

A Big Data solution typically comprises these as logical layers. Each layer can be represented by one or more available technologies.

Big Data sources layer The choice of sources of data for an application depends upon what data is required to perform the kind of analyses you need. Big Data will vary in origin, size, speed, form, and function, as described by the 4 Vs in Chapter 1. The sources of Big Data, as described in Chapter 2, can be internal or external to the organization. The scope of access to data available could be limited. The level of structure as well as the speed of data and its quantity could be high or low depending upon the data source.

Data Ingest layer This layer is responsible for acquiring data from the various data sources. Incoming data is received through a scalable number of input points, that can acquire data at various speeds and in various quantities. The data is sent to a batch processing system, or a realtime processing system, or to a storage file system such as Hadoop. Compliance regulations and governance policies impact what data can be stored and for how long.

Batch Processing layer The analysis layer receives data from the ingest point or from the file system or from the NoSQL databases. Data is processed using parallel programming techniques (such as MapReduce) to process it and produce the desired results. This batch processing layer thus must understand the data sources and data types, the algorithms that would work on that data, and the format of the desired outcomes. The output of this layer could be sent for instant reporting, or be stored in NoSQL databases for an on-demand report, for the client.

Stream Processing layer This technology layer receives data directly from the ingest point. Data is processed using parallel programming techniques (such as MapReduce) to process it in real time, and produce the desired results. This layer thus should understand the data sources and data types extremely well, and the super-light algorithms that would work on that data to produce the desired results. The outcome of this layer too could be stored in the NoSQL Databases.

Data Organizing Layer This layer receives data from both the batch and stream processing layers. Its objective is to organize the data for easy access. It is represented by NoSQL databases. There are a variety of NoSQL databases to suit different needs. SQL-like languages like Hive and Pig can be used to access data easily and generate reports from these databases.

Infrastructure Layer At bottom there is a layer that manages the raw resources of storage, computation, and communication. This is increasingly provided through a cloud computing paradigm.

Distributed File System Layer This is the heart of a Big Data system. It would store huge quantities of data and make it quickly and securely available and accessible to the other layers. Hadoop Distributed File System (HDFS) is the primary technology in this layer. It would also include supporting applications, such as YARN (Yet Another Resource Manager) that enable the efficient access to data storage and its transfer.

Data Consumption layer This is the final layer, and it consumes the output provided by the analysis layers, directly or through the organizing layer. The outcome could be standard reports, data analytics, dashboards and other visualization applications, recommendation engine, on mobile and other devices.

3.2 BIG DATA ARCHITECTURE EXAMPLES

Every major organization and its applications have a unique optimized infrastructure to suit its specific needs. Some architecture examples from some very prominent users and designers of Big Data applications are as follows.

3.2.1 IBM Watson

IBM Watson uses Spark to manage incoming data streams. It also uses Spark's Machine Learning library (MLLib) to analyze data and predict diseases (Figure 3.3).

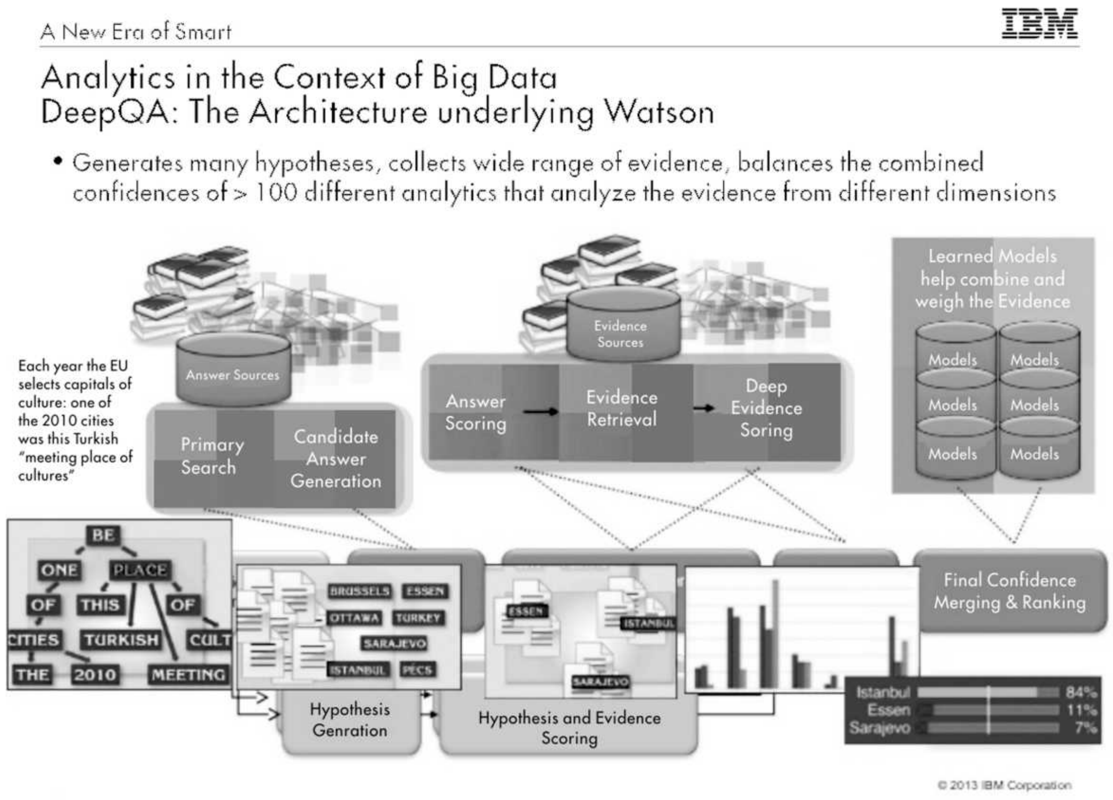

A New Era of Smart
IBM
Analytics in the Context of Big Data
DeepQA: The Architecture underlying Watson
• Generates many hypotheses, collects wide range of evidence, balances the combined confidences of > 100 different analytics that analyze the evidence from different dimensions
Each year the EU selects capitals of culture: one of the 2010 cities was this Turkish "meeting place of cultures"
Answer Sources
Primary Search
Candidate Answer Generation
Evidence Sources
Answer Scoring
Evidence Retrieval
Deep Evidence Soring
Learned Models help combine and weigh the Evidence
Models
Models
Models
Models
Models
Models
BE
ONE
PLACE
OF
THIS
OF
CITIES
TURKISH
CULT
THE
2010
MEETING
BRUSSELS
ESSEN
OTTAWA
TURKEY
SARAJEVO
ISTANBUL
PECS
ESSEN
ISTANBUL
SARAJEVO
Final Confidence Merging & Ranking
Istanbul 84%
Essen 11%
Sarajevo 7%
Hypothesis Genration
Hypothesis and Evidence Scoring
© 2013 IBM Corporation

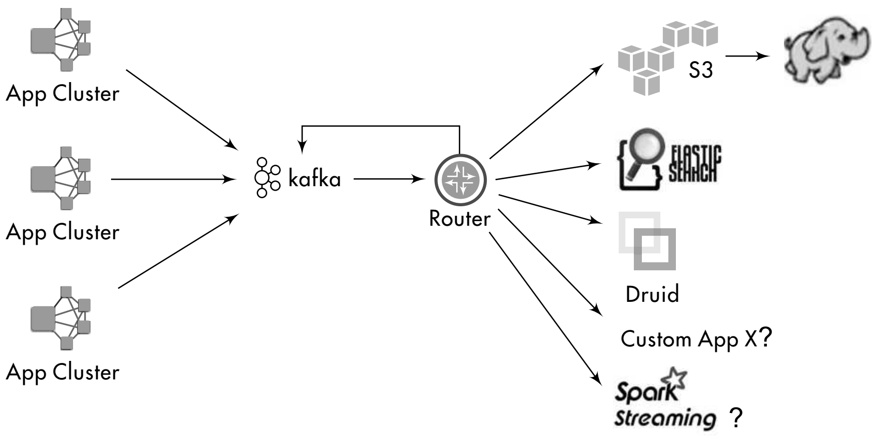
FIGURE 3.3 The architecture underlying IBM Watson

3.2.2 Netflix

This is one of the largest providers of online video entertainment. They handle more than 400 billion online events per day. As a cutting-edge user of Big Data technologies, they are constantly innovating their mix of technologies to deliver the best performance. They use Apache Kafka as the common messaging system for all incoming requests. They use Spark for stream processing and host their entire infrastructure on Amazon Web Services (AWS) cloud service (Figure 3.4). The database system is AWS' S3 as well as Cassandra and Hbase for storing data.

3.2.3 eBay

eBay is the second-largest e-commerce company in the world. It delivers 800 million listings from 25 million sellers to 160 million buyers. To manage this huge stream of activity, eBay uses a stack of Hadoop, Spark, Kafka, and other elements (Figure 3.5). They think that Kafka is the best new thing for processing data streams.



(*Source:* Netflix)

FIGURE 3.4 Netflix database processing and storage system

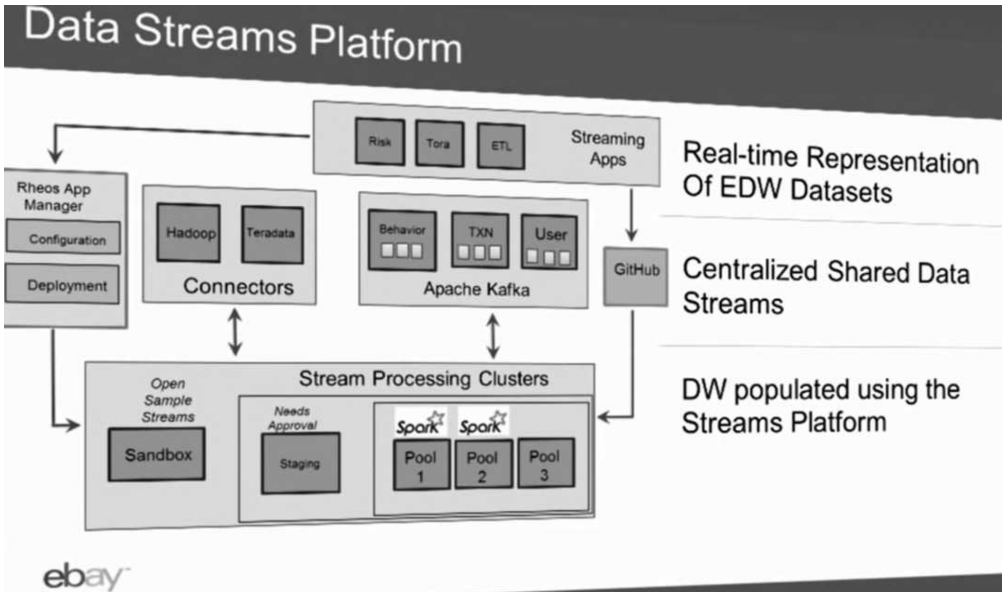


FIGURE 3.5 eBay data stream platforms

3.2.4 VMWare

VMware is a leading provider of virtualization systems used in cloud computing systems. VMware's view of a Big Data architecture is similar to, but more detailed, than our main big architecture diagram (Figure 3.6).

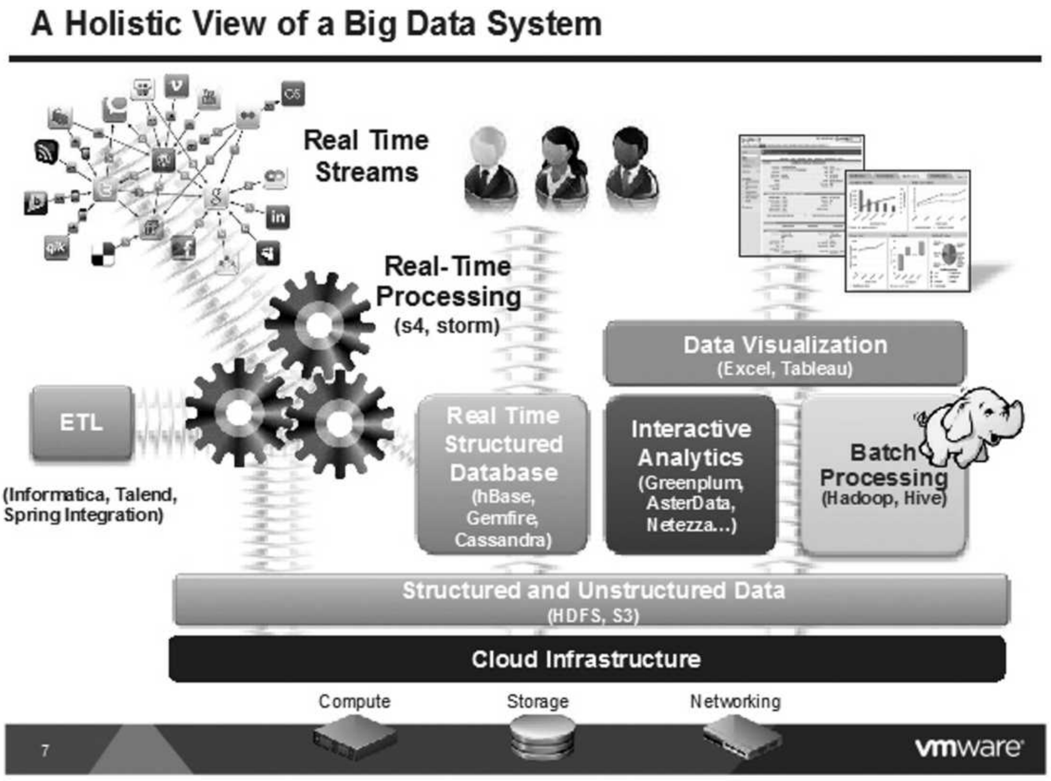

A Holistic View of a Big Data System
Real Time Streams
Real-Time Processing (s4, storm)
ETL
(Informatica, Talend, Spring Integration)
Data Visualization (Excel, Tableau)
Real Time Structured Database (hBase, Gemfire, Cassandra)
Interactive Analytics (Greenplum, AsterData, Netezza...)
Batch Processing (Hadoop, Hive)
Structured and Unstructured Data (HDFS, S3)
Cloud Infrastructure
Compute
Storage
Networking
7
vmware

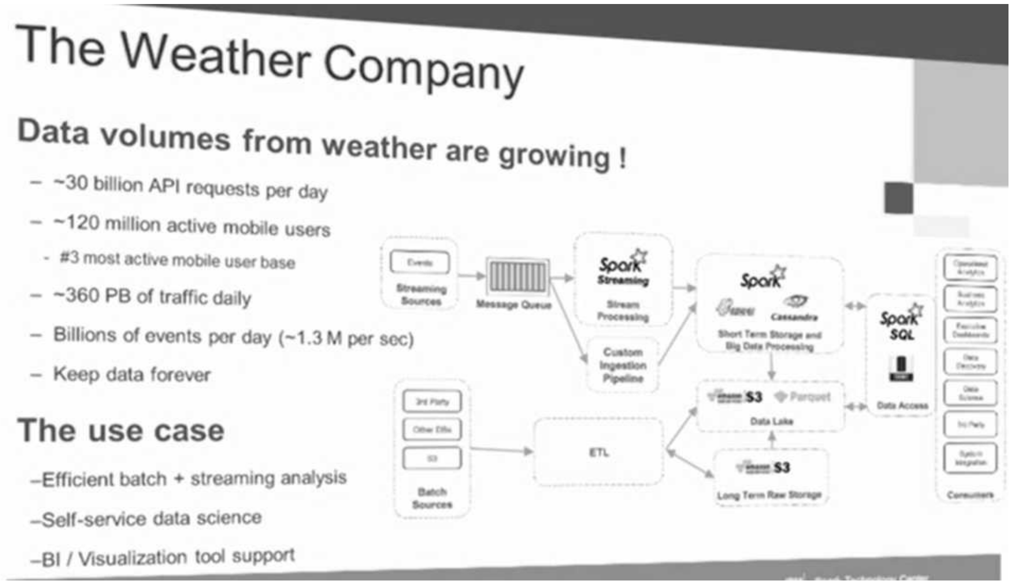
FIGURE 3.6 A holistic view of a big data system in VMWare

3.2.5 The Weather Company

The Weather company serves tens of billions of weather data requests globally through websites and mobile apps, everyday. It uses streaming architecture using Apache Spark to frame its Big Data infrastructure (Figure 3.7).

3.2.6 TicketMaster

This is the world's largest company that sells event tickets. Their goal is to make tickets available to real fans for purchase, and prevent bad actors from manipulating the system to increase the price of the tickets in the secondary markets. They use a mix of Hadoop, Spark, and Kafka systems to manage throughput and reliability (Figure 3.8).



The Weather Company

Data volumes from weather are growing !

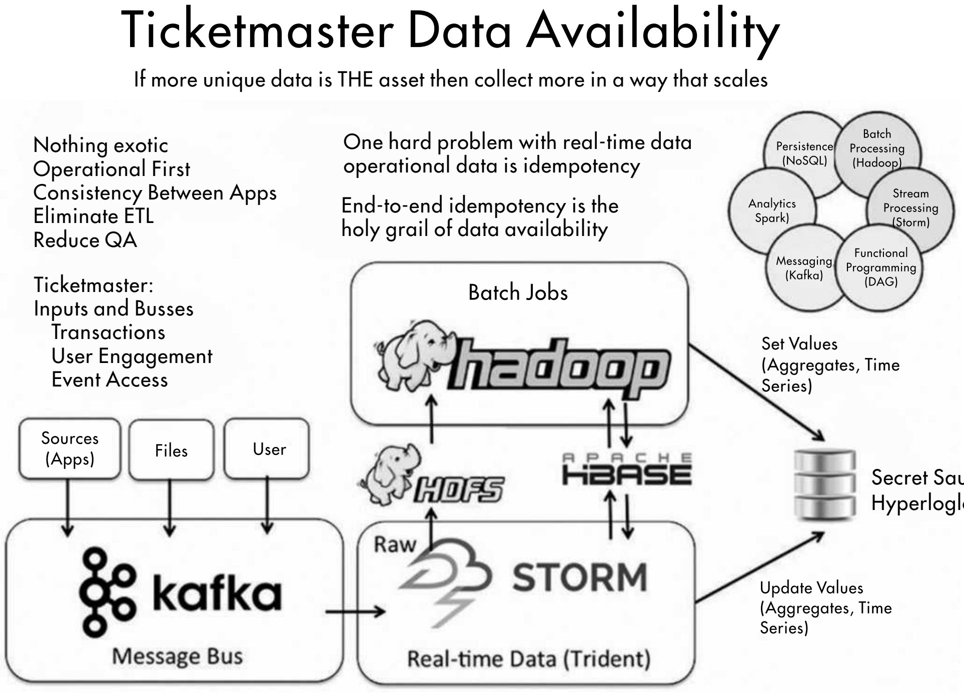
- ~30 billion API requests per day
- ~120 million active mobile users
  - #3 most active mobile user base
- ~360 PB of traffic daily
- Billions of events per day (~1.3 M per sec)
- Keep data forever

The use case

- Efficient batch + streaming analysis
- Self-service data science
- BI / Visualization tool support

Events
Streaming Sources
Message Queue
Spark Streaming
Stream Processing
Custom Ingestion Pipeline
Spark
Cassandra
Short Term Storage and Big Data Processing
Spark SQL
Data Access
S3
Parquet
Data Lake
3rd Party
Other DBs
S3
Batch Sources
ETL
S3
Long Term Raw Storage
Consumers

FIGURE 3.7 Streaming architecture using Apache Spark to frame Big Data infrastructure of the weather company



Ticketmaster Data Availability

If more unique data is THE asset then collect more in a way that scales

Nothing exotic
Operational First
Consistency Between Apps
Eliminate ETL
Reduce QA

Ticketmaster:
Inputs and Busses
Transactions
User Engagement
Event Access

One hard problem with real-time data operational data is idempotency

End-to-end idempotency is the holy grail of data availability

Persistence (NoSQL)
Batch Processing (Hadoop)
Analytics Spark)
Stream Processing (Storm)
Messaging (Kafka)
Functional Programming (DAG)
Batch Jobs
hadoop
Set Values (Aggregates, Time Series)
Sources (Apps)
Files
User
HDFS
APACHE HBASE
Secret Sauce: Hyperloglog
Raw
kafka
Message Bus
STORM
Real-time Data (Trident)
Update Values (Aggregates, Time Series)

FIGURE 3.8 Mix of Hadoop, Spark, and Kafka systems to manage throughput and reliability by TicketMaster

3.2.7 LinkedIn

This professional networking company aims to maintain an efficient system for processing streaming data and make the link options available in real-time. They use Hadoop and Kafka for receiving and storing data, and Samza for stream processing (Figure 3.9).

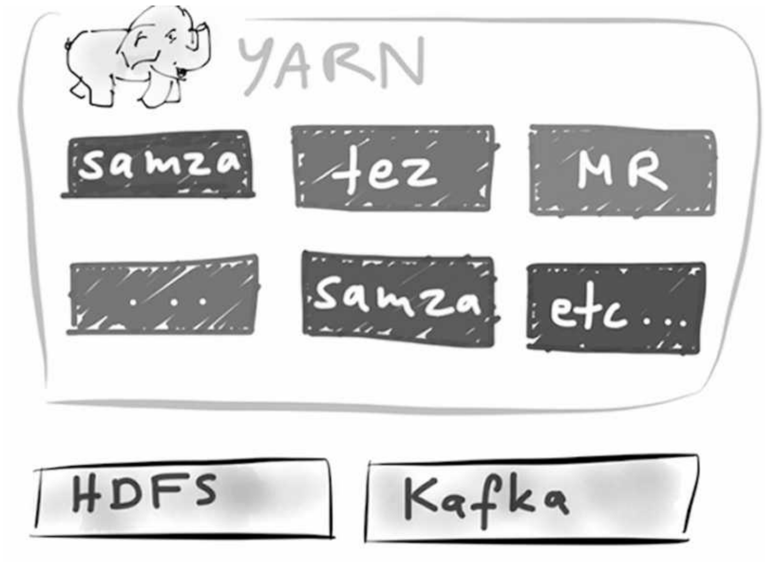


FIGURE 3.9 Mix of Hadoop, Spark, and Kafka systems to manage data storage and Samza for stream processing by LinkedIn

3.2.8 Paypal

This payments-facilitation company needs to understand and acquire customers, and process a large number of payment transactions. They use Apache Storm for stream processing of data. They use Druid for distributed data storage. They also use Apache Titan Graphical (NoSQL) database to manage links and relationship among data elements (Figure 3.10).

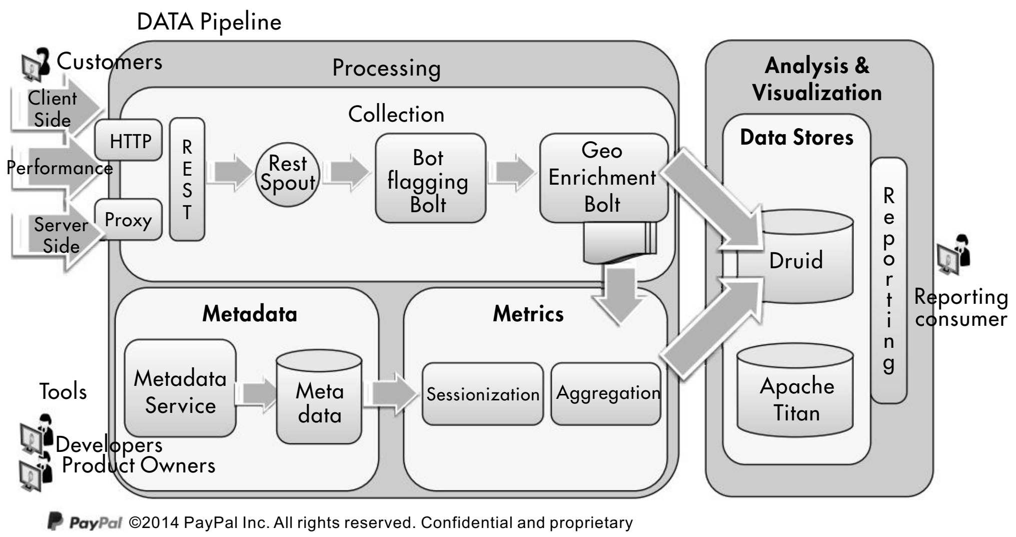


FIGURE 3.10 Apache Storm and Apache Titan Graphical (NoSQL) database to manage links and relationship among data elements in Paypal

3.2.9 CERN

This premier high-energy physics research lab computes petabytes of data using in-memory stream processing to process data from millions of sensors and devices (Figure 3.11).

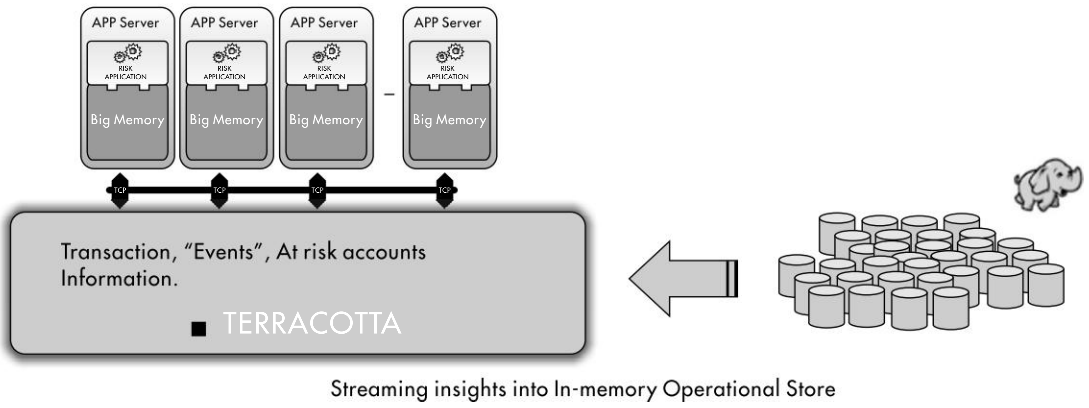

APP Server
RISK APPLICATION
Big Memory
TCP
Transaction, "Events", At risk accounts Information.
TERRACOTTA
Streaming insights into In-memory Operational Store

FIGURE 3.11 Streaming insights into In-memory operational store in CERN lab

3.3 CONCLUSION

Big Data architectures are multi-tier designs that use technologies for ingesting, storing, processing, and delivering data to user applications. Data is ingested and fed into both streaming and batch processing engines. Most tools used for big data processing are open source tools served through the Apache community.

Review Questions

1. Describe the Big Data processing architecture.
2. What are Google's contributions to Big data processing?
3. What are some of the hottest technologies visible in Big Data processing?

True/False Questions

1. The more the sources of Big data included, the more complex will the Big Data architecture be.
2. Big Data architecture includes many kinds of data sources.
3. Big Data architecture includes transaction processing applications.
4. A simple Big Data architecture consists of two layers: a data ingest layer, and a data consumption layer.
5. Once established, a company's Big Data Architecture cannot be changed.
6. Serving web-based apps is a form of a Big Data application.
7. Google invented Big Data architecture with Search Query engine.
8. IBM Watson uses Big Data architecture to do Deep Question-Answer applications.
9. Facebook is a not an active user of Big Data architectures.
10. Government organizations should use a more secure Big Data architecture.

Liberty Stores Case Exercise: Step B3

Liberty wants to build a scalable and futuristic listening platform for understanding its customers and other stakeholders.

1. What kind of Big Data architecture would you suggest for this company?

Section 2

This section covers the six important Big Data technologies as defined in the Big Data architecture (specified in Chapter 3).

- ➨ **Chapter 4** will discuss Hadoop and its Distributed File System (HDFS).
- ➨ **Chapter 5** will delve into MapReduce parallel processing algorithm. It will also cover Pig and Hive languages used for programming as front end to MapReduce.
- ➨ **Chapter 6** will discuss NoSQL databases, including details of popular columnar databases HBase and Cassandra.
- ➨ **Chapter 7** will highlight Streaming systems using Apache Spark.
- ➨ **Chapter 8** will address Data Ingest systems, using Apache Kafka.
- ➨ **Chapter 9** will cover Cloud Computing model and its many benefits and challenges.

Chapter 4

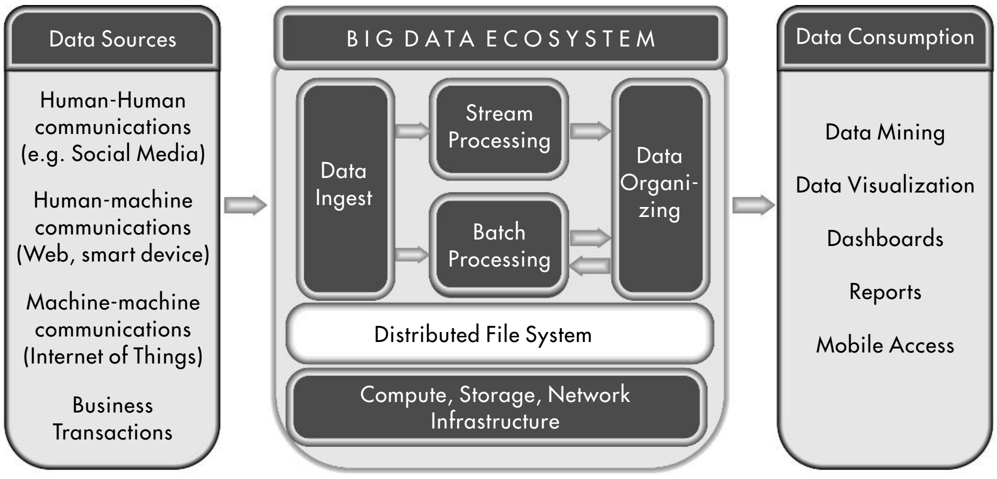
Distributed Computing Using Hadoop

Learning Objectives

- Describe Hadoop as open-source technologies for Big Data processing
- Understand Hadoop Distributed File System (HDFS)
- Appreciate HDFS' hierarchical master-slave architecture
- Recognize the two kinds of Hadoop files, Text, and Sequence files
- View sample code for reading and writing data from HDFS
- Understand the purpose and design of YARN, the Resource Manager

Introduction

A distributed file storage system is a clever way of storing huge quantities of data, securely and cost-effectively, for speed and ease, for retrieval and processing, using a networked collection of commodity machines. The ideal distributed file system would store infinite amounts of data while making the complexity completely hidden from the user, and enabling instant and easy access to the right data. This would be achieved by storing sections of data at different locations, and internally managing the lower-level tasks of integrating and replicating data across the network. The distributed system ultimately leads to the creation of the unbounded cosmic computer that is a manifestation of all the laws of nature.



4.1 Big data architecture

4.1 HADOOP FRAMEWORK

The Apache Hadoop distributed computing framework is composed of the following modules:

1. Hadoop Common – contains libraries and utilities needed by other Hadoop modules
2. Hadoop Distributed File System (HDFS) – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster
3. YARN – a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications, and
4. MapReduce – an implementation of the MapReduce programming model for large scale data processing.

This chapter will cover Hadoop Common, HDFS, and YARN. The next chapter will cover MapReduce.

4.2 HDFS DESIGN GOALS

The Hadoop distributed file system (HDFS) is a distributed and scalable file-system. It is designed for applications that deal with very large data sizes. It is also designed to deal with mostly immutable files, i.e. write data once, but read it many times.

HDFS has the following major design goals:

1. Hardware failure management – it will happen, and one must plan for it
2. Huge volume – create capacity for large number of huge file sizes, with fast read/write throughput
3. High speed – create a mechanism to provide low latency access to streaming applications
4. High variety – Maintain simple data coherence, by writing data once but reading many times
5. Plug-and-play – Maintain easy accessibility of data using any hardware, software, and database platform
6. Network efficiency – Minimize network bandwidth requirement, by minimizing data movement.

4.3 MASTER-SLAVE ARCHITECTURE

Hadoop is an architecture for organizing computers in a master-slave relationship that helps achieve great scalability in processing. A Hadoop cluster has two types of nodes operating in a master–worker pattern: a single master node (called NameNode), and a large number of slave worker nodes (called DataNodes). A small Hadoop cluster includes a single master and multiple worker nodes. A large Hadoop cluster would consist of a master and thousands of small ordinary machines as worker nodes (Figure 4.1).

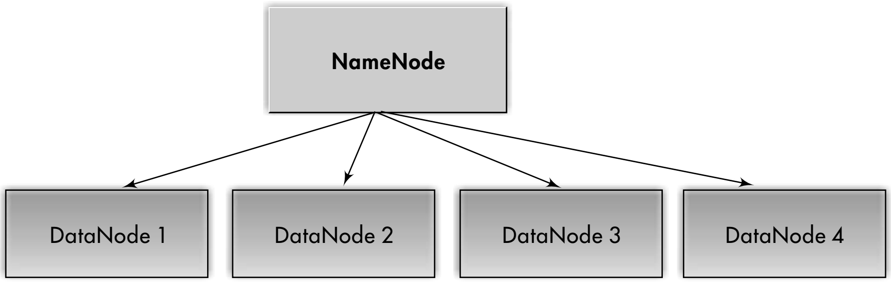


FIGURE 4.1 Master-slave architecture

In Hadoop system, the master node manages the overall file system, its namespace, and controls the access to files by clients. The master node is aware of the data-nodes, i.e. which blocks of which file are stored on which data node. It also controls the processing plan for all applications running on the data on the cluster. There is only one master node. Unfortunately, that makes it a single point of failure. Therefore, whenever possible, the master node has its hot backup always ready to take over, just in case the master node dies unexpectedly. The master node uses a transaction log to persistently record every change that occurs to file system.

The worker nodes, also called DataNodes, store the data blocks in their storage space, as directed by the master node, also called the NameNode. Each worker node typical contains many disks to maximize storage capacity and access speed. Each worker node has its own local file system. A worker node has no awareness of the distributed file structure. It simply stores each block of data as directed, as if each block were a separate file. The DataNodes store and serve up blocks of data over the network using a block protocol, under the direction of the NameNode (Figure 4.2).

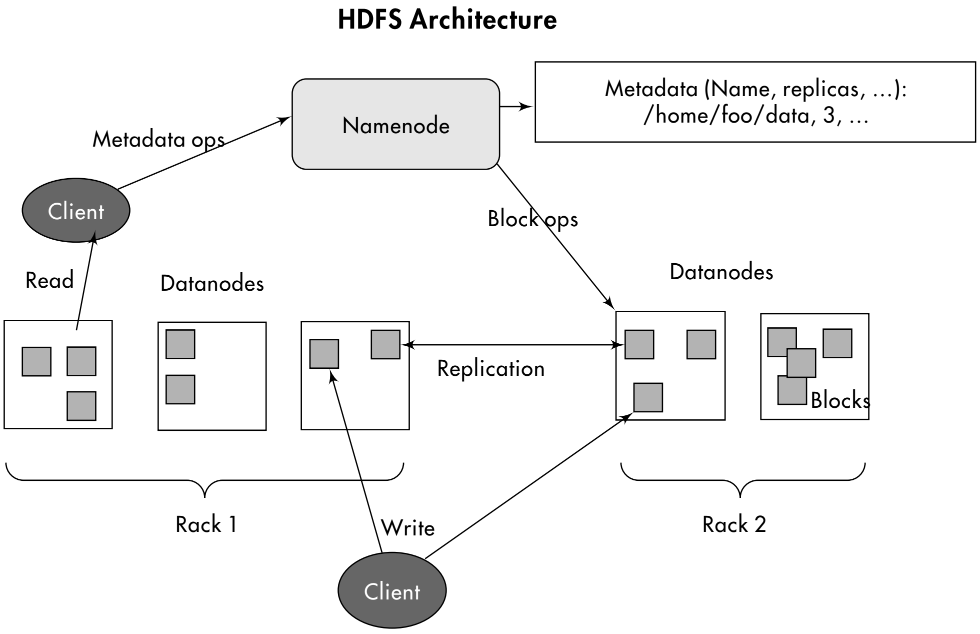


FIGURE 4.2 Hadoop architecture (*Source:* Hadoop.apache.org)

The NameNode stores all relevant information about all the DataNodes, and the files stored in those DataNodes. The NameNode will know the following:

- For every DataNode, its name, rack, capacity, and health
- For every File, its name, replicas, type, size, timeStamp, location, health, etc.

It a DataNode fails, there is no serious problem. The data on the failed DataNode will be accessed from its replicas on other DataNodes. The failed DataNode can be automatically recreated on another machine, by writing all those file blocks of from the other healthy replicas. Each DataNode sends a heartbeat message to the NameNode periodically. Without this message, the DataNode is assumed to be dead. The DataNode replication effort would automatically kick-in to replace the dead DataNode.

The Hadoop file system has a set of features and capabilities to completely hide the splintering and scattering of data, and enable the user to deal with the data at a high, logical level. The NameNode tries to ensure that files are evenly spread across the data-nodes in the cluster. That balances the storage and computing load and, also, limits the extent of loss from the failure of a node. The NameNode also tries to optimize the networking load. When retrieving data or ordering the processing, the NameNode tries to pick Fragments from multiple nodes to balance the processing load and speed up the total processing effort. The NameNode also tries to store fragments of files on the same node for speed of read and writing. Processing is done on the node where the file fragment is stored.

Any piece of data is stored typically on three nodes: two on the same rack, and one on a different rack. DataNodes can talk to each other to rebalance data, to move copies around, and to keep the replication of data high.

4.4 BLOCK SYSTEM

A block of data is the fundamental storage unit in HDFS. HDFS stores large files (typically gigabytes to terabytes) by storing segments (called blocks) of the file across multiple machines. Data files are described, read and written in block-sized granularity. All storage capacity and file sizes are measured in blocks. A block ranges from 16–128MB in size, with a default block size of 64MB. Thus, an HDFS file is chopped up into 64 MB chunks, and if possible, each chunk will reside on a different DataNode.

Every data file takes up a number of blocks depending upon its size. Thus a 100 MB file will occupy two blocks (100MB divided by 64MB), with some room to spare. Every storage disk can accommodate several blocks depending upon the size of the disk. Thus, a 1 Terabyte storage will have 16000 blocks (1TB divided by 64MB).

Every file is organized as a consecutively numbered sequence of blocks. A file's blocks are stored physically close to each other for ease and speed of access, as far as possible. The file's block size and replication factor are configurable by the application that writes the file on HDFS.

4.5 ENSURING DATA INTEGRITY

Hadoop ensures that no data is lost or corrupted during storage or processing. The files are written only once, and never updated in place. They can be read many times. Only one client can write or append to a file, at a time. No concurrent updates are allowed.

If some data on a DataNode is indeed lost or corrupted, or if a part of the disk gets corrupted, a new healthy replica for that lost block will be automatically recreated by copying from the replicas on other data-nodes. At least one of the replicas is stored on a data-node on a different rack. This guards against the failure of the rack of nodes, or the networking router, on it.

A checksum algorithm is applied on all data written to HDFS. A process of serialization is used to turn files into a byte stream for transmission over a network or for writing to persistent storage. Hadoop has additional security built in, using Kerberos verifier.

4.6 INSTALLING HDFS

It is possible to run Hadoop on an in-house cluster of machines, or on the cloud inexpensively. As an example, The New York Times used 100 Amazon Elastic Compute Cloud (EC2) instances (DataNodes) and a Hadoop application to process 4 TB of raw image TIFF data stored in Amazon Simple Storage Service (S3) into 11 million finished PDFs in the space of 24 hours at a computation cost of about $240 (plus the cost of bandwidth). See Chapter 9 for a primer on Cloud Computing and Appendix A for a step-by-step tutorial on installing Hadoop on Amazon EC2.

Hadoop is written in Java. Hadoop also requires a working Java installation. Most access to files is provided through Java abstract class org.apache.hadoop.fs.FileSystem.

Installing Hadoop takes many resources. For example, all information about fragments of files needs to be in NameNode memory. A thumb rule is that Hadoop needs approximately 1GB memory to manage 1 million file fragments. Many easy mechanisms exist to install the entire Hadoop stack. Using a GUI such as Cloudera Resources Manager to install a Cloudera Hadoop stack is easy. This stack includes HDFS and many other related components, such as HBase, Pig, and YARN. Installing it on a cluster on a cloud services provider like AWS is easier than installing Java Virtual Machines (JVMs) on local machines. HDFS can be installed by using Cloudera's GUI Resources Manager. If installing using command line, download Hadoop from one of the Apache mirror sites.

HDFS can be mounted directly with a Filesystem in Userspace (FUSE) virtual file system on Linux and some other Unix systems. File access can be achieved through the native Java APIs. Another API, called Thrift, helps to generate a client in the language of the users' choosing (such as C++, Java, Python). When the Hadoop command is invoked with a *classname* as the first argument, it launches a JVM to run the class, along with the relevant Hadoop libraries (and their dependencies) on the *classpath*.

HDFS has a UNIX-like command like interface (CLI). Use *sh* shell to communicate with Hadoop. HDFS has UNIX-like permissions model for files and directories. There are three progressively increasing levels of permissions: read (r), write (w), and execute (x). Create a *hduser*, and communicate using *ssh* shell on the local machine.

```
% hadoop fs -help     ## get detailed help on every command.
```

4.6.1 Reading and Writing Local Files into HDFS

There are two different ways to transfer data: from the local file system, or form an input/output stream. Copying a file from the local filesystem to HDFS can be done by:

```
% hadoop fs -copyFromLocal path/filename
```

4.6.2 Reading and Writing Data Streams into HDFS

Read a file from HDFS by using a java.net.URL object to open a stream to read the data requires a short script, as below.

```
InputStream in = null;
Start {
instream = new URL("hdfs://host/path").openStream();  // details of process in }
Finish  {  IOUtils.closeStream(instream); }
```

A simple method to create a new file is as follows:

```
public FSDataOutputStream create(Path p) throws IOException
```

Data can be appended to an existing file using the append() method:

```
public FSDataOutputStream append(Path p) throws IOException
```

A directory can be created by a simple method:

```
public boolean mkdirs(Path p) throws IOException
```

List the contents of a directory using:

```
public FileStatus[] listStatus(Path p) throws IOException
public FileStatus[] listStatus(Path p, PathFilter filter) throws IOException
```

4.7 SEQUENCE FILES

The incoming data files can range from very small to extremely large, and with different structures. Big Data files are therefore organized quite differently to handle the diversity of file sizes and type. Large files are stored as Text Files, with File Fragments distributed across the cluster. However, smaller files should be bunched together into single segment for efficient storage.

Sequence Files are a specialized data structure within Hadoop to handle smaller files with smaller record sizes. Sequence File uses a persistent data structure for data available in key-value pair format. These help efficiently store smaller objects. HDFS and MapReduce are designed to work with large files, so packing small files into a Sequence File container, makes storing and processing the smaller files more efficient for HDFS and MapReduce.

Sequence files are row-oriented file formats, which means that the values for each row are stored contiguously in the file. This formats are appropriate when a large number of columns of a single row are needed for processing at the same time. There are easy commands to create, read and write Sequence File structures. Sorting and merging Sequence Files is native to MapReduce system. A MapFile is essentially a sorted Sequence File with an index to permit lookups by key.

4.8 YARN

YARN (Yet Another Resource Negotiator) is the architectural centre of Hadoop, it is often characterized as a large-scale, distributed operating system for big data applications. YARN manages resources and monitors workloads, in a secure multi-tenant environment, while ensuring high availability across multiple Hadoop clusters. YARN also brings great flexibility as a common platform to run multiple tools and applications such as interactive SQL (e.g. Hive), real-time streaming (e.g. Spark), and batch processing (MapReduce), to work on data stored in a single HDFS storage platform (Figure 4.3). It brings clusters more scalability to expand beyond 1000 nodes, it also improves cluster utilization through dynamic allocation of cluster resources to various applications.

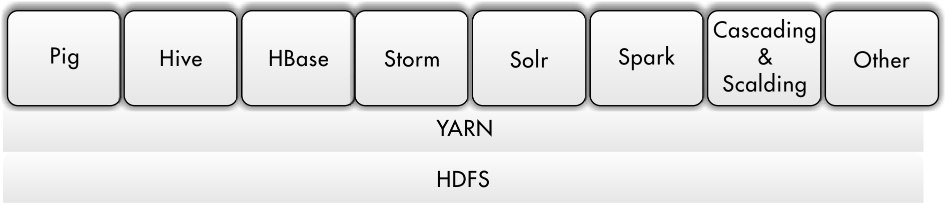


FIGURE 4.3 Hadoop distributed architecture including YARN

The Resource Manager in YARN has two main components: Scheduler and Applications Manager.

YARN Scheduler allocates resources to the various requesting applications. It does so based on an abstract notion of a resource **Container** which incorporates elements such as Memory, CPU, Disk storage, Network, etc. Each machine also has a NodeManager that manages all the Containers on that machine, and reports status on resources and Containers to the YARN Scheduler.

YARN Applications Manager accepts new job submissions from the client. It then requests a first resource Container for the application-specific ApplicationMaster program, and monitors the health and execution of the application. Once running, the ApplicationMaster directly negotiates additional resource containers from the Scheduler as needed.

4.9 CONCLUSION

Hadoop is the dominant technology for managing Big Data. Hadoop Distributed Files system securely stores data on large clusters of commodity machines. A master machine controls the storage and processing activities of the worker machines. A NameNode controls the namespace and storage information for the file system on the DataNodes. A master JobTracker controls the processing of tasks at the DataNodes. YARN is the resources manager that manages all resources dynamically and efficiently across all applications on the cluster. Hadoop File system and other parts of the Hadoop stack are distributed by many providers, and can be easily installed on cloud computing infrastructure. Hadoop installation tutorial is in Appendix A.

Review Questions

1. How does Hadoop differ from a traditional file system?
2. What are the design goals for HDFS?

3. How does HDFS ensure security and integrity of data?
4. How does a master node differ from the worker node?

True/False Questions

1. Hadoop is a big project of the open-source Apache series of Big Data products.
2. Hadoop Distributed File system is a general purpose file manager for Big data projects.
3. Hadoop provides unlimited scalability of data storage.
4. HDFS uses a ring architecture to organize masses of data and machines.
5. A name-node is a machine that stores the names of all the data files and their locations of the various data storage devices.
6. If a data-node fails, data may be lost, unless it is backed up by the user.
7. HDFS permits dozens of file formats to optimize storage of many different of data.
8. One of the design goals of HDFS is to reduce network bandwidth requirements.
9. YARN is the task manager that computes data on a node.
10. Hadoop is written in Scala and Python.

Chapter 5

Parallel Processing with Map Reduce

Learning Objectives

- Recognize the need for parallel processing for Big Data
- Evaluate the MapReduce processing paradigm: Map and Reduce programs
- Understand the key-pair data structure necessary for MapReduce
- Learn the program structure for MapReduce programs
- Analyze the concepts of Job Tracker and Task Tracker programs
- Learn how MapReduce executes programs despite node failures

INTRODUCTION

A parallel processing system is a clever way to process huge amounts of data in a short period by enlisting the services of many computing devices to work on different parts of the job, simultaneously. The ideal parallel processing system will work across any computational problem, using any number of computing devices, across any size of data sets, with ease and high programmer productivity. This is achieved by framing the problem in a way that it can be broken down into many parts, such that each part can be partially processed independently of the other parts; and then the intermediate results from processing the parts can be combined to produce a final solution. Infinite parallel processing is the essence of infinite dynamism of the laws of nature.

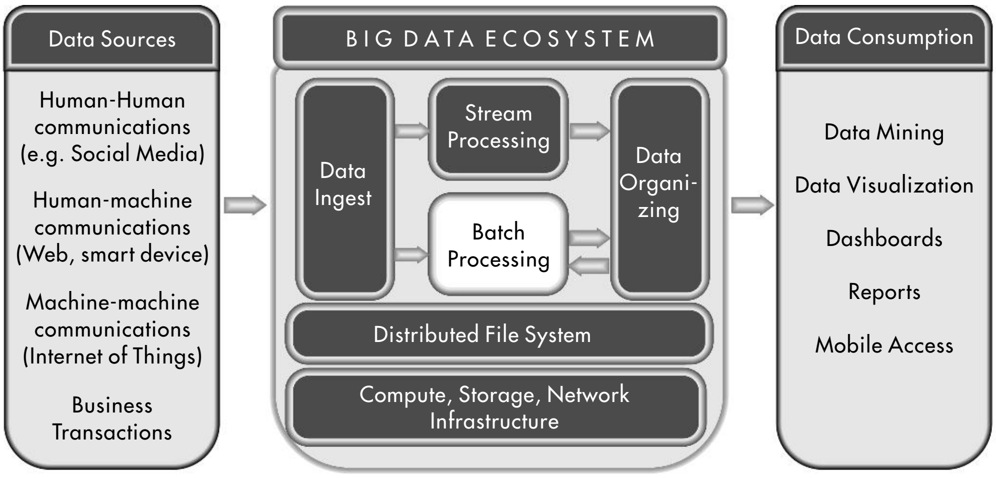


FIGURE 5.1 Big data architecture

Caselet

How Google search works?

When Google search engine receives a search query, it uses MapReduce algorithm to quickly and efficiently search and return the right answers to the query.

First, the search term is broken into smaller query using its key words. For example, if one wants to search for 'US Presidential Election', the search term will be mapped to three sub-queries for each of the three words in this query. A Map program is generated for each of these three sub-queries.

The Map program for each sub-query will then be sent out to tens of thousands of data nodes in data-centers around the world, where the web page data is stored. Each node will process the data and return the matching a list of websites for that subquery to the requesting node.

The intermediate results will be gathered, sorted, and then combined by a Reducer program, which will also sort the results in order of importance according to certain criteria. The sorted results are sent back to the browser. All this back and forth would happen in a fraction of a second.

5.1 MAPREDUCE OVERVIEW

MapReduce is a parallel programming framework for speeding up large scale data processing for certain types of tasks. It achieves so with minimal movement of data on distributed file systems on Hadoop clusters, to achieve near real-time results. There are two major pre-requisites for MapReduce programming. (a) The application must lend itself to parallel programming. (b) The data for the applications can be expressed in key-value pairs.

MapReduce processing is similar to UNIX sequence (also called pipe) structure

e.g. the UNIX command:

```
grep | sort | count  myfile.txt
```

will produce a wordcount in the text document called myfile.txt.

There are three commands in this sequence, and they work as follows: (a) grep is a command to read the text file and create an intermediate file with one word on a line; (b) sort command will sort that intermediate file, and produce an alphabetically sorted list of words in that set; (c) the count command will work on that sorted list, to produce the number of occurrences of each word, and display the results to the user in a "word, frequency" pair format.

For example: Suppose myfile.txt contains the following text:

> Myfile: *We are going to a picnic near our house. Many of our friends are coming. You are welcome to join us. We will have fun.*

The outputs of Grep, Sort and Wordcount will be as follows:

Grep
We
are
going
to
a
picnic
near
our
house
Many
of
our
friends
are
coming
You

Sort
a
Are
Are
Are
Coming
Friends
Fun
Going
Have
House
Join
Many
Near
Of
Our
Our

WordCount	
A	1
Are	3
Coming	1
Friends	1
Fun	1
Going	1
Have	1
House	1
Join	1
Many	1
Near	1
Of	1
Our	2
Picnic	1
To	2
Us	1

are
welcome
to
join
us
we
will
have
fun

Picnic
To
To
Us
We
We
welcome
Will
You

We	2
Welcome	1
will	1
You	1

If the file is very large, then it will be take the computer a long time to process it. Parallel processing can help here.

MapReduce speeds up the computation by reading and processing small chunks of file, by different computers in parallel. Thus if a file can be broken down into 100 small chunks, each chunk can be processed at a separate computer in parallel. The total time taken to process the file could be 1/100 of the time taken otherwise. However, now the results of the computation on small chunks are residing in a 100 different places. These large number of partial results must be combined to produce a composite result. The results of the outputs from various chunks will be combined by another program called the Reduce program.

The Map step will distribute the full job into smaller tasks that can be done on separate computers each using only a part of the data set. The result of the Map step will be considered as intermediate results. The Reduce step will read the intermediate results, and will combine all of them and produce the result. The programmer should specify the functional logic for both the map and reduce steps. The sorting, between the Map and Reduce steps, does not need to be specified and is automatically taken care of the MapReduce system as a standard service provided to every job. The sorting of the data requires a field to sort on. Thus the intermediate results should have a key field, and a set of associated non-key attribute(s) for that key (Figure 5.2).

In practice, to manage the variety of data structures stored in the file system, data is stored as one key and one non-key attribute value. Thus the data is represented as a key-value pair. The intermediate results, and the results all will also be in key-pair format. Thus a key requirement for the use of MapReduce parallel processing system is that the input data and output data must both be represented in key-values formats.

Map step reads data in key-value pair format. The programmer decides what should be the characteristics of the key and value fields. The Map step produces results in

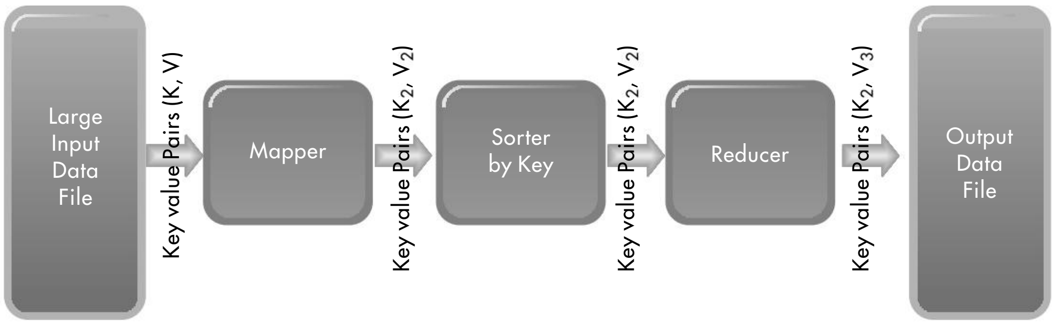

Large Input Data File
Key value Pairs (K, V)
Mapper
Key value Pairs (K_2, V_2)
Sorter by Key
Key value Pairs (K_2, V_2)
Reducer
Key value Pairs (K_2, V_3)
Output Data File

FIGURE 5.2 MapReduce architecture

key-value pair format. However, the characteristics of the keys produced by the Map step, i.e. the intermediate results, need not be same keys at the input data. So, those can be called key2-value2 pairs.

The Reduce step reads the key2-value2 pairs, and produces an output using the same keys that it read. Only the values associated with those keys will change though as a result of processing. Thus the Reducer's output can be labeled as key2-value3 format.

5.2 SAMPLE MAPREDUCE APPLICATION: WORDCOUNT

Suppose we want to identify unique words in a piece of text, and the frequency of the occurrence of each word in the text. Suppose the text in the data file myfile.txt can be split into 4 approximately equal segments. In this case, it could be done with each sentence as a separate piece of text. The four segments will look as following:

Segment1: We are going to a picnic near our house.

Segment2: Many of our friends are coming.

Segment3: You are welcome to join us.

Segment4: We will have fun.

Each of these 4 segments of data could be processed in parallel. The results of those processing could be suitably aggregated to provide results for the text as a whole.

Thus there will be 4 Map tasks, one for each of the segments of data. Each Map process will take in input in a <key-value> pair format. The first column is the key, which is the entire sentence in this case. The second column is the value, which in this application is the frequency of the sentence. Each Map process will also be executed by a different processor.

We are going to a picnic near our house.	1

Many of our friends are coming.	1

You are welcome to join us.	1

We will have fun.	1

This task can be done in parallel by four processors. Each of this segment will be task for a different processor. Thus, each task will produce a file of words, with a count of 1. There will be four intermediate files, in <key2, value2> pair format, as shown here.

Key2	**Value2**
we	1
are	1
going	1
to	1
a	1
picnic	1
near	1
our	1
house	1

Key2	**Value2**
many	1
Of	1
Our	1
friends	1
Are	1
coming	1

Key2	**Value2**
you	1
Are	1
Welcome	1
To	1
Join	1
Us	1

Key2	**Value2**
we	1
will	1
have	1
fun	1

The sort process inherent within MapReduce will sort each of the intermediate files, and produce the following sorted key2-value2 pairs:

Key2	**Value2**
a	1
Are	1
Going	1
House	1
Near	1
Our	1
Picnic	1
to	1
We	1

Key2	**Value2**
Are	1
coming	1
friends	1
many	1
Of	1
our	1

Key2	**Value2**
Are	1
Join	1
To	1
Us	1
welcome	1
you	1

Key2	**Value2**
fun	1
have	1
we	1
will	1

The Reduce function will read the sorted intermediate files, and combine the counts for all the unique words, to produce the following output. The keys remain the same as in the intermediate results. However, the values change as counts from each of the intermediate files are added up for each key. For example, the count for the word 'are' goes up to 3.

Key2	**Value3**
a	1
are	3
coming	1
friends	1
fun	1
going	1
have	1
house	1
join	1
many	1
near	1
of	1
our	2
picnic	1
to	2
us	1
we	2
welcome	1
will	1
you	1

This output will be identical to that produced by the UNIX grep sequence earlier.

5.3 MAPREDUCE PROGRAMMING

A data processing problem needs to be transformed into the MapReduce model. The first step is to visualize the processing plan into a Map and a Reduce step. When the processing gets more complex, this complexity can be generally manifested in having more MapReduce jobs, or more complex Map and Reduce jobs. Having more but simpler MapReduce jobs leads to more easily maintainable Map and Reduce programs.

5.3.1 MapReduce Data Types and Formats

MapReduce has a simple model of data processing: inputs and outputs for the map and reduce functions are key-value pairs. The map and reduce functions in Hadoop MapReduce have the following general form:

> map: (K1, V1) $\rightarrow$ list (K2, V2)
>
> reduce: (K2, list(V2)) $\rightarrow$ list (K2, V3)

In general, the map input key and value types (K1 and V1) are different from the Map output types (K2 and V2). However, the Reduce input must have the same types as the Map output, although the Reduce output types may be different again (K2 and V3).

MapReduce can process many different types of data formats, from flat text files to databases. An input split is a chunk of the input that is processed by a single Map. Each Map processes a single split. Each split is divided into records, and the map processes each record—a key-value pair—in turn. Splits and records are logical: and may map to a full file, a part of a file, or a collection of files. In a database context, a split might correspond to a range of rows from a table and a record to a row in that range.

5.3.2 Writing MapReduce Programming

It should start by writing pseudocode for the Map and Reduce functions. The program code for both the Map and the Reduce function can then be written in Java or other languages. In Java, the Map function is represented by the generic Mapper class. It uses four parameters: (input key, input value, output key, output value). This class uses an abstract map () method. This method receives the input key and input value. It would normally produce and output key and output value. For more complex problems, it is better to use a higher-level language than MapReduce, such as Pig, Hive, and Spark.

A Mapper commonly performs input format parsing, projection (selecting the relevant fields), and filtering (selecting the records of interest). The Reducer typically combines (adds or averages) those values (Figure 5.3).

Following is the step-by-step logic to do a word count of all unique words in a text.

1. The big document is split into many segments. The Map step is run on each segment of data. The output will be a set of (key, value) pairs. In this case, the key will be a word in the document.

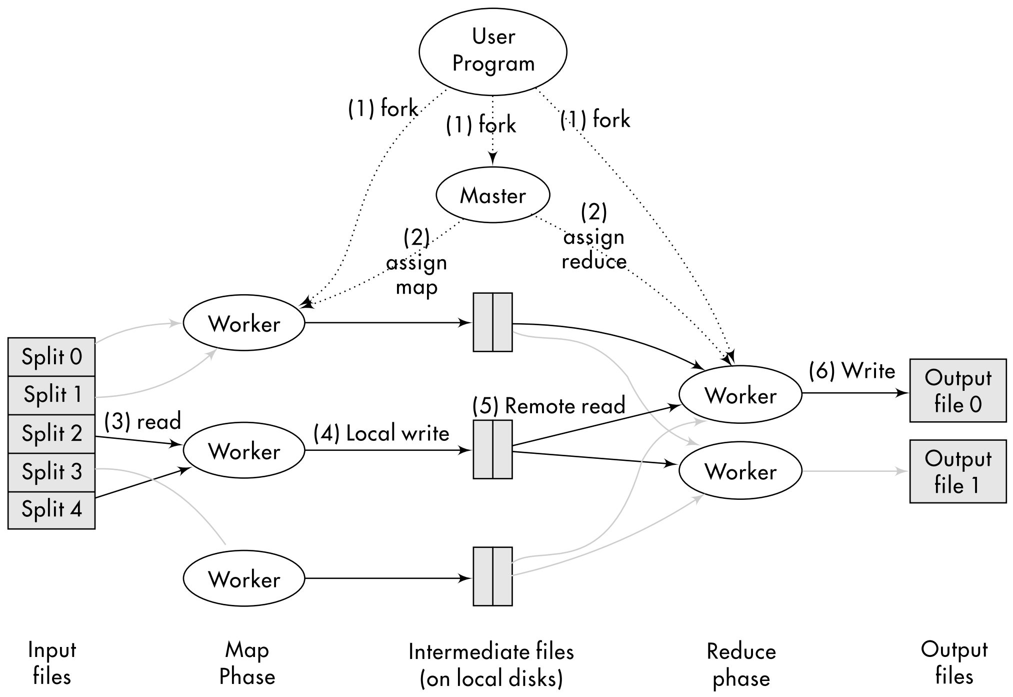


FIGURE 5.3 MapReduce program flow

2. The system will gather the (key, value) pair outputs from all the mappers, and will sort them by key. The sorted list itself may then be split into a few segments.
3. A Reducer task will read the sorted list and produce a combined list of word counts.

Here is the Java code for wordcount:.

```
map(String key, String value):
  for each word w in value:
  EmitIntermediate(w, "1");
reduce(String key, Iterator values):
  int result = 0;
  for each v in values:
    result += ParseInt(v);
Emit(AsString(result));
```

5.3.3 Testing MapReduce Programs

Mapper programs running on a cluster can be complicated to debug. The time-honoured way of debugging programs is via print statements. However, with the programs eventually running on tens or thousands of nodes, it is best to debug the programs in stages. Therefore, (a) run the program using small sample datasets to ensure that the program is working correctly; (b) Expand the unit tests to cover larger dataset and run it on a cluster; (c) Ensure that the Mapper or Reducer can handle the inputs correctly. Running against the full dataset is likely to expose some more issues, which should be fixed, by altering your Mapper or Reducer to handle the new cases. After the program is working, the program may be tuned to make the entire MapReduce job run faster.

It may be desirable to split the logic into many simple Mappers and chaining them into a single Mapper using a facility (the ChainMapper library class) built into Hadoop. It can run a chain of Mappers, followed by a Reducer and another chain of Mappers, in a single MapReduce job.

5.4 MAPREDUCE JOBS EXECUTION

A MapReduce job is specified by the Map program and the Reduce program, along with the data sets associated with that job. There is another master program that resides and runs endlessly on the NameNode. It is called the Job Tracker, and it tracks the progress of the MapReduce jobs from beginning to the completion. Hadoop moves the Map and Reduces computation logic to every DataNode that is hosting a fragment of the data. The communication between the nodes is accomplished using YARN, Hadoop's native resource manager.

The master machine (NameNode) is completely aware of the data stored on each of the worker machines (DataNodes). It schedules the Map or Reduce jobs to Task trackers with full awareness of the data location. For example: if node A contains data (x,y,z) and node B contains data (a,b,c), the Job tracker schedules node B to perform map or Reduce tasks on (a,b,c) and node A would be scheduled to perform Map or Reduce tasks on (x,y,z). This reduces the data traffic and prevents choking of the network.

Each DataNode has a master program called the Task tracker. This program monitors the execution of every task assigned to it by the NameNode. When the task is completed, the Task tracker sends a completion message to the Job tracker program on the Name Node (Figure 5.4). The jobs and tasks work in a 'master-slave' mode.

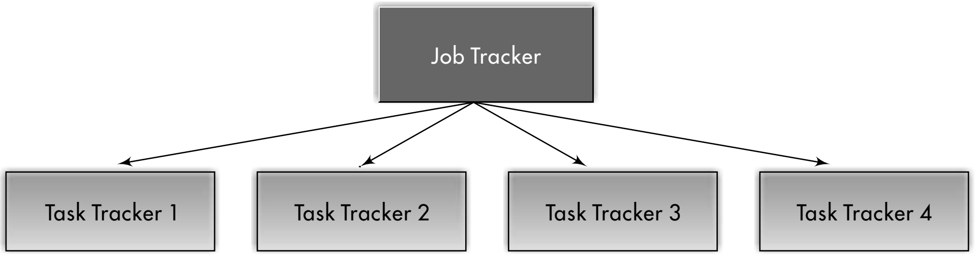


FIGURE 5.4 Hierarchical monitoring architecture

When there is more than one job in a MapReduce workflow, it is necessary that they be executed in the right order. For a linear chain of jobs, it might be easy while for a more complex directed acyclic graph (DAG) of jobs, there are libraries that can help orchestrate the workflow. Or one can use Apache Oozie, a system for running workflows of dependent jobs. Oozie consists of two main parts: a workflow engine that stores and runs workflows composed of different types of Hadoop jobs (MapReduce, Pig, Hive, and so on), and a coordinator engine that runs workflow jobs based on predefined schedules and data availability. Oozie has been designed to scale, and it can manage the timely execution of thousands of workflows in a Hadoop cluster.

The dataset for the MapReduce job is divided into fixed-size pieces called input splits, or just splits. Hadoop creates one map task for each split, which runs the user-defined Map function for each record in the split. The tasks are scheduled using YARN and run on DataNodes in the cluster. YARN ensures that if a task fails or inordinately delayed, it will be automatically scheduled to run on a different node. The outputs of the Map jobs are fed as input to the Reduce job. That logic is also propagated to the node(s) that will do the Reduce jobs. To save on bandwidth, Hadoop allows the use of a combiner function on the Map output. Then the combiner function's output forms the input to the Reduce function.

5.4.1 How MapReduce Works

A MapReduce job can be executed with a single method call: submit () on a Job object. When the resource manager receives a call to its submitApplication() method, it hands off the request to the YARN scheduler. The scheduler allocates a container, and the resource manager then launches the application master's process. The application master for MapReduce jobs is a Java application whose main class is MRAppMaster. It initializes the job by creating several bookkeeping objects to keep track of the job's progress. It retrieves the input splits computed in the client from the shared filesystem. It then creates a Map task object for each split, as well as a number of Reduce tasks. Tasks are assigned IDs at this point.

The application master must decide how to run the tasks that make up the MapReduce job. The application master requests containers for all the Map and Reduce tasks in the job from the resource manager. Once a task has been assigned resources for a container on a particular node by the resource manager's scheduler, the application master starts the container by contacting the node manager. The task is executed by a Java application whose main class is YarnChild.

5.4.2 Managing Failures

There can be failures at the level of the entire job or particular tasks. The entire application master itself could fail.

Task failure usually happens when the user code in the map or reduce task throws a runtime exception. If this happens, the task JVM reports the error to its parent application master, where it is logged into error logs. The application master will then reschedule execution of the task on another data node. The entire job, i.e. MapReduce application master application running on YARN, too can fail. In that case, it is started again, subject to a configurable maximum number.

If a DataNode manager fails by crashing or running very slowly, it will stop sending heartbeats to the resource manager (or send them very infrequently). The resource manager will then remove it from its pool of nodes to schedule containers on. Any task or application master running on the failed node manager will be recovered using error logs, and started on other nodes.

The Resource Manager YARN can also fail. That has more severe consequences, as the entire cluster is affected. Therefore, typically, there will be a hot-standby for YARN. If the active resource manager fails, then the standby can take over without a significant interruption to the client. The new resource manager can read the application information from the state store, and then restart the application that were running on the cluster.

5.4.3 Shuffle and Sort

MapReduce guarantees that the input to every Reducer is sorted by a key. The process by which the system performs the sort—and transfers the Map outputs to the Reducers as inputs—is known as the shuffle.

When the Map function starts producing output, it is not directly written to disk. It takes advantage of buffering — writes in memory, and does some presorting for efficiency reasons. Each Map task has a circular memory buffer that it writes the output to. Before it writes to disk, the thread first divides the data into partitions corresponding to the Reducers that they will ultimately be sent to. Within each partition, the background thread performs an in-memory sort by key. If there is a combiner function, it is run on the output of the sort so that there is less data to transfer to the reducer.

The Reduce task needs the Map output for its particular partition from several Map tasks across the cluster. The Map tasks may finish at different times, so the Reduce task starts reading their outputs as soon as all the Map programs have successfully been executed. When all the Map outputs have been read, the Reduce task merges the Map outputs, maintaining their sort ordering. The Reduce function is invoked for each key in the sorted output. The output of this phase is written directly to the output filesystem.

5.4.4 Progress and Status Updates

MapReduce jobs are long-running batch jobs, and take a long time to run. It is important for the user to get feedback on how the job's progress. A job and each of its tasks have a status value (e.g., running, successfully completed, failed) the progress of maps and reduces, the values of the job's counters. These values are constantly communicated back to the client. When the application master receives a notification that the last task for a job is complete, it changes the status for the job to "successful." Job statistics and counters are communicated to the user.

Hadoop comes with a native web-based GUI for tracking the MapReduce jobs. It displays useful information about a job's progress such as how many tasks have been completed, and which ones are still being executed. Once the job is completed, one can view the job statistics and logs.

5.5 HADOOP STREAMING

Hadoop Streaming uses standard Unix streams as the interface between Hadoop and user program. Streaming is an ideal application for text processing. Map input data is passed over standard input to your map function, which processes it line by line and writes lines to standard output. A Map output key-value pair is written as a single tab-delimited line. Input to the Reduce function is in the same format—a tab-separated key-value pair—passed over standard input. The Reduce function reads lines from standard input, which the framework guarantees and are sorted by key, and writes its results to standard output.

5.6 HIVE LANGUAGE

Hive is a declarative SQL-like language for processing queries. Hive was designed to appeal to a community that is comfortable with SQL. Data analysts mainly use it, on the server side, for generating reports. It has its own metadata section which can be defined ahead of time, i.e. before the data is loaded. Hive supports Map and Reduce transform scripts in the language of the user's choice, which can be embedded within SQL clauses. Hive is best used for producing reports using structured data; it is not designed for online transaction processing.

Hive can be used to query data stored in Hbase, which is a key-value store. Hive's SQL-like structure makes transformation of data to and from an RDBMS is easier. Supporting SQL syntax also makes it easy to integrate with existing Business Intelligence (BI) tools. Hive needs the data to be first imported (or loaded) and after that it can be worked upon. In case of streaming data, one would have to keep filling buckets (or files), and then Hive can be used to process each filled bucket, while using other buckets to keep storing the newly arriving data.

Hive data Columns are mapped to tables in HDFS (Figure 5.5). This mapping is stored in Metadata. All HQL queries are converted to MapReduce jobs. A table can have one or more partition keys. There are usual SQL data types, and Arrays and Maps and Structs to represent more complex types of data.

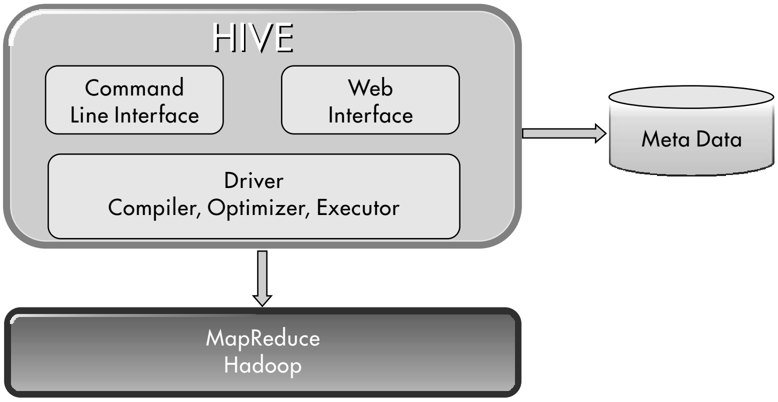


FIGURE 5.5 Hive architecture

5.6.1 HIVE Language Capabilities

Hive's SQL provides almost all basic SQL operations. These operations work on tables and (or) partitions. These operations are: SELECT, FROM, WHERE, JOIN, GROUP BY, ORDER BY. It also allows the results to be stored in another table, or in a HDFS file.

Hive	Relational Database
SQL	SQL
Analytics	OLTP or analytics
Batch only	Real-time or batch
No transactions	Transactions
No INSERT or UPDATE Adding through partitions	Random INSERT or UPDATE
Distributed processing – 100s of nodes	Depends on the system – if available, < 100
Achieve high performance on commodity hardware	Achieve high performance on proprietary hardware
Low cost for huge amounts of storage	Expensive, limited compared to Hadoop based solutions

The statement to create a page view table would be like:

```
CREATE TABLE page_view(viewTime INT, userid BIGINT,
               page_url STRING, referrer_url STRING,
               ip STRING COMMENT 'IP Address of the User')
COMMENT 'This is the page view table'
PARTITIONED BY (dt STRING, country STRING)
STORED AS SEQUENCEFILE;
```

Here is a script for loading data into this file.

```
CREATE EXTERNAL TABLE page_view_stg(viewTime INT, userid BIGINT,
              page_url STRING, referrer_url STRING,
              ip STRING COMMENT 'IP Address of the User',
              country STRING COMMENT 'country of origination')
COMMENT 'This is the staging page view table'
ROW FORMAT DELIMITED FIELDS TERMINATED BY '44' LINES TERMINATED BY '12'
STORED AS TEXTFILE
LOCATION '/user/data/staging/page_view';
```

The table created above can be stored in HDFS as a TextFile or as a SequenceFile.

An INSERT query on this table will look like:

```
hadoop dfs -put /tmp/pv_2008-06-08.txt /user/data/staging/page_view
 FROM page_view_stg pvs
```

```
INSERT OVERWRITE TABLE page_view PARTITION(dt='2008-06-08', country='US')
SELECT pvs.viewTime, pvs.userid, pvs.page_url, pvs.referrer_url, null, null, pvs.ip
WHERE pvs.country = 'US';
```

5.7 PIG LANGUAGE

Pig is also a high-level scripting language/platform for data manipulation, that is used with Hadoop and MapReduce. Compared with Hive, Pig offers greater procedural control over data flows, and thus excels at solving problems such as ETL that require great control over data flows (Table 5.1). Pig is also used for performing tasks involving ad-hoc processing and quick prototyping. For example, to process huge data sources such as web logs, and to perform data processing for search engines. Pig has a scripting part and an execution part. Pig Latin is the scripting language to develop high-level code. It provides a rich set of data types, functions, and operators to perform various operations on the data. This script is then parsed, optimized and run by Pig Engine to create MapReduce code which then runs and delivers the results to the user. Pig thus helps deliver the power to deliver the insights from big data flexibly and efficiently.

Table 5.1

Comparison between Apache Pig and Apache Hive platforms

Apache Pig	Apache Hive
Pig uses a language called Pig Latin. It was originally created at Yahoo.	Hive uses a language called HiveQL. It was originally created at Facebook.
Pig Latin is a data flow language.	HiveQL is a query processing language.
Pig Latin is a procedural language and it fits in a data flow pipeline paradigm.	HiveQL is a declarative language like SQL.
Pig can handle structured, unstructured, and semi-structured data.	Hive is used mostly for structured data.

The architecture of Apache Pig is shown in Figure 5.6.

The Parser checks the syntax of the Pig Latin script, and produces a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators. In the DAG, the logical operators of the script are represented as the nodes and the data flows are represented as edges. The Optimizer carries out the logical optimizations such as projection and pushdown on the DAG. The Compiler compiles the optimized logical plan into a series of MapReduce jobs. The MapReduce jobs are submitted to Hadoop in a sorted order. The desired results are returned to the user.

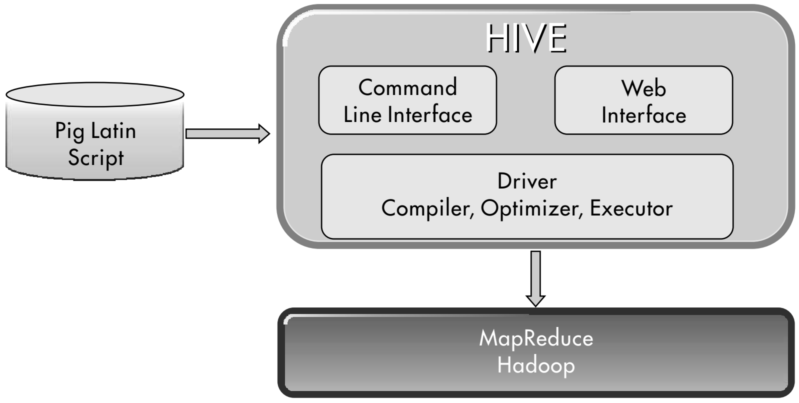


FIGURE 5.6 Architecture of Apache Pig

5.7.1 PIG Language Capabilities

Required data manipulations in Hadoop are possible with Pig Latin. It offers greater control over data flows, and it can perform all the data manipulation operations through MapReduce. Pig is thus very suitable for ETL (Extract, Transform, and Load) jobs. An ETL job extracts data from a source, transforms it according to a rule set, and then loads it into a datastore. Pig Latin allows splits in the data pipeline. It allows developers to store data anywhere in the pipeline. This reduces the need for duplicating the read/write code at multiple places, and significantly reduces the overall length of the code. Pig reduces the program size by a factor of 10 compared to Java, and thus saves development time by a factor of 10, while still delivering comparable data manipulation capabilities.

Pig is also sufficiently SQL-like language that is easy to learn. Anyone with a basic knowledge of SQL can work conveniently with Pig. Pig provides many built-in operators to support data operations like joins, sorts, and filters. Therefore, performing an important relational operation like Join is very simple in Pig. Pig has a rich library of built-in functions. In addition, it allows the use of User-Defined functions (UDFs), that enable sophisticated program logic to be written in Java and other such languages, and then run from within Pig. Conversely Pig scripts can be executed in other languages. Thus one can use Pig to build components of larger and more complex applications that tackle real business problems.

Pig works primarily with data stored in Hadoop Distributed File System. It works through the MapReduce parallel processing mechanism. Pig can ingest data from files, streams, or other sources. Using built-in functions as well as User Defined Functions (UDF), the data can be ingested and transformed in very sophisticated ways. The results are stored back into HDFS.

Pig can handle structured, unstructured, and semi-structured data. Pig enables many kinds of data structures such as Atom, Tuple, bag, Map and Relations. A relational schema is not necessary for a Pig data structure. The data model in Apache Pig is nested-relational. It provides nested data types like tuples, bags, and maps that are not available in MapReduce. Pig does not have a dedicated metadata section; the schema needs to be defined in the program itself. Pig can be easier for someone who had not earlier experience with SQL.

5.7.2 Pig Script Example

Here is Pig Script for loading sample driver data and producing summary reports.

```
# Read file 1 and create a schema
drivers = LOAD ‘drivers.csv’ USING PigStorage(‘,’);
drivers_details = FOREACH drivers GENERATE $0 AS driverId, $1 AS name;

# Read file 2 and create a schema
timesheet = LOAD ‘timesheet.csv’ USING PigStorage(‘,’);
timesheet_logged = FOREACH timesheet GENERATE $0 AS driverId, $2 AS hours_logged,
$3 AS miles_logged;

# generate summary data and add those fields to schema
grp_logged = GROUP timesheet_logged by driverId;

sum_logged = FOREACH grp_logged GENERATE group as driverId,

SUM(timesheet_logged.hours_logged) as sum_hourslogged,

SUM(timesheet_logged.miles_logged) as sum_mileslogged;
# generate a JOIN of two files and generate a report from it
join_sum_logged = JOIN sum_logged by driverId, drivers_details by driverId;
join_data = FOREACH join_sum_logged GENERATE $0 as driverId, $4 as name, $1 as
hours_logged, $2 as miles_logged;

# discard the temporary JOIN relation
dump join_data;
```

5.8 CONCLUSION

MapReduce is the most popular parallel processing framework for Big Data. It works well for applications where the data can be large, divisible into separate sets, and represented in <key, value> pair format. The application logic is divided into two parts: a Map program and a Reduce Program. Each of these programs can be run in parallel using several machines. A Job tracker tracks the processing of the entire MapReduce job, while a Task tracker monitors the performs the processing on a data node. Hive and Pig are high-level languages that make MapReduce programming easier.

Review Questions

1. What is MapReduce? What are its benefits?
2. What is the key-value pair format? How is it different from other data structures? What are its benefits and limitations?
3. What is a Job tracker program? How does it differ from the task tracker program?
4. What are Hive and Pig? How are they different?

True/False Questions

1. MapReduce processing is similar to UNIX sequence (also called pipe) structure.
2. MapReduce speeds up the computation by reading and processing small chunks of file, by different computers in parallel.
3. The MapReduce model can be used for any data processing problem.
4. The Map step reads data in key-value pair format.
5. A mapper commonly performs input format parsing, projection, and filtering of data.
6. The Reduce function reads unsorted intermediate files.
7. One should start MapReduce processing by writing pseudocode for the map and reduce functions.
8. The Task tracker program monitors the progress of the MapReduce jobs from beginning to the completion.
9. One Task Tracker can control many Job Trackers.
10. MapReduce was invented by Google.

Chapter 6

NoSQL Databases

Learning Objectives

- Identify key differences between NoSQL and relational databases
- Appreciate the architecture of NoSQL databases
- Describe the major types of NoSQL databases and their features
- Analyze the architecture and processes of Hadoop HBase
- Analyze the architecture and processes of Cassandra database

Introduction

A NoSQL database is a clever way of cost-effectively organizing large amounts of heterogeneous data for efficient access and updates. An ideal NoSQL database is completely aligned with the nature of the problems being solved, and is superfast in accomplishing that task. This is achieved by relaxing many of the integrity and redundancy constraints of storing data in relational databases. Data is thus stored in many innovative formats closely aligned with business need. The diverse NoSQL databases will ultimately collective evolve into a holistic set of efficient and elegant knowledge stored at the heart of a cosmic computer.

Relational data management systems (RDBMs) are a powerful and universally used database technology by almost all enterprises. Relational databases are structured and optimized to ensure accuracy and consistency of data, while also eliminating any redundancy of data. These databases are stored on the largest and most reliable of computers to ensure that the data is always available at a granular level and at a high speed.

Big data is, however, a much larger and unpredictable stream of data. Relational databases are inadequate for this task, and will also be very expensive for such large data volumes. Managing the costs and speed of managing such large and heteroge-

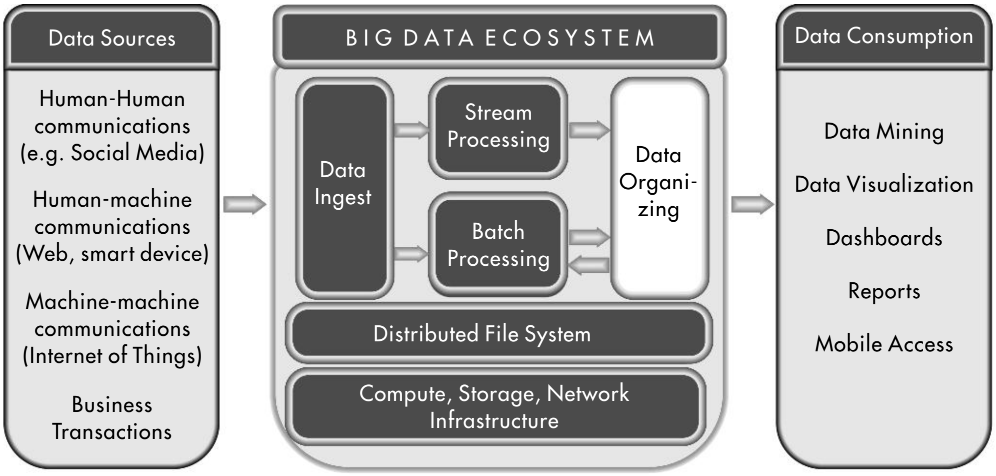


FIGURE 6.1 Big data architecture

neous data streams requires relaxing many of the strict rules and requirements of relational database. Depending upon which constraint(s) are relaxed, a different kind of database structure will emerge. These are called NoSQL databases, to differentiate them from relational databases that use Structured Query Language (SQL) as the primary means to manipulate data.

NoSQL databases are next-generation databases that are non-relational in their design. The name NoSQL is meant to differentiate it from antiquated, 'pre-relational' databases. Today, almost every organization that must gather customer feedback and sentiments to improve their business, uses a NoSQL database. NoSQL is useful when an enterprise needs to access, analyze, and utilize massive amounts of either structured or unstructured data that's stored remotely in virtual servers across the globe.

The constraints of a relational database are relaxed in many ways. For example, relational databases require that any data element could be randomly accessed and its value could be updated in that same physical location. However, the simple physics of storage says that it is simpler and faster to read or write sequential blocks of data on a disk. Therefore, NoSQL database files are written once and almost never updated in place. If a new version of a part of the data become available, it would be appended to the respective files. The system would have the intelligence to link the appended data to the original file.

6.1 RDBMS VS NoSQL

These differ from each other in many ways. First, NoSQL databases do not support relational schema or SQL language. The term NoSQL stands mostly for "Not only

SQL". Second, their transaction processing capabilities are fast but weak, and they do not support the ACID (Atomicity, Consistency, Isolation, Durability) properties associated with transaction processing using relational databases. Instead, they support BASE properties (Basically Available, Soft State, and Eventually Consistent). NoSQL databases are thus approximately accurate at any point in time, and will be eventually consistent. Third, these databases are also distributed and horizontally scalable to manage web-scale databases using Hadoop clusters of storage. Thus, they work well with the write-once and read-many storage mechanism of Hadoop clusters. Table 6.1 lists comparative features of RDBMS and NoSQL.

Table 6.1

Comparative features of RDBMS and NoSQL.

Feature	RDBMS	NoSQL
Applications	Mostly centralized Applications (e.g. ERP)	Mostly designed for the decentralized applications (e.g. Web, mobile, sensors)
Rigor	Support ACID properties for Transaction Processing	Support BASE properties for approximate reporting
Availability	Moderate to high	Continuous availability to receive and serve data
Velocity	Moderate velocity of data	High velocity of data (devices, sensors, social media, etc.). Low latency of access.
Data Volume	Moderate size; archived after for a certain period	Huge volume of data, stored mostly for a long time or forever; Linearly scalable DB.
Data Sources	Data arrives from one or few, mostly predictable sources	Data arrives from multiple locations and are of unpredictable nature
Data type	Data are mostly structured	Structured or unstructured data
Data Access	Primary concern is reading the data	Concern is both read and write
Technology	Standardized relational schemas; SQL language	Many designs with many implementations of data structures and access languages
Cost	Expensive; commercial	Low; open-source software

6.2 TYPES OF NoSQL DATABASES

The variety of big data means that file size and types will vary enormously. Despite the name, a NoSQL database does not necessarily prohibit structured query language (like SQL). While some of the NoSQL systems are entirely non-relational, others just avoid some selected functionality of RDMS such as fixed table schemas and join operations. For NoSQL systems, instead of using tables, the data can be organized the data in key/value pair format, and then SQL can be used. There are specialized NoSQL databases to suit different purposes.

The choice of NoSQL database depends on the system requirements. There are at least 200 implementations of NoSQL databases of these four types. Visit nosql-database.org for more. Here are some recent offerings in these categories (Figure 6.2).

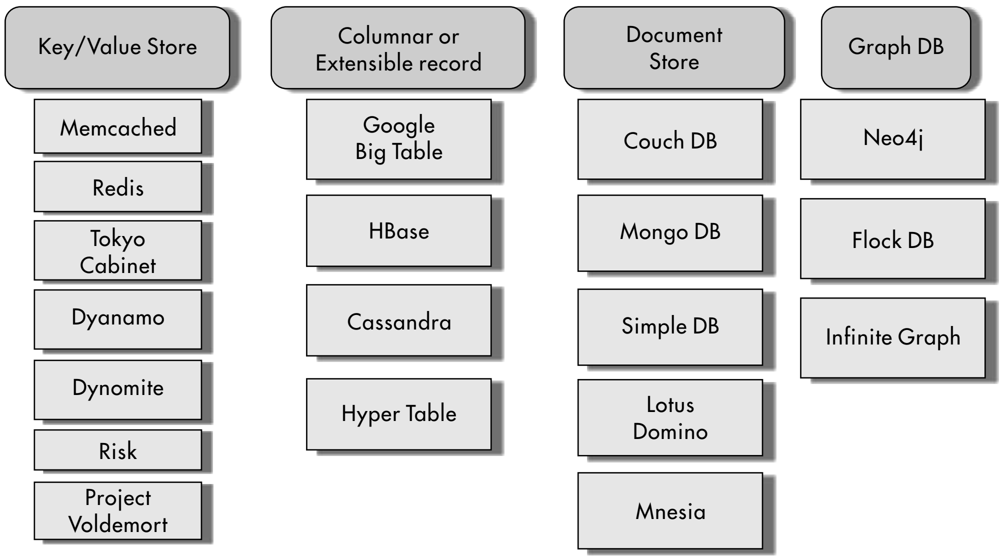


FIGURE 6.2 Offerings in NoSQL databases

1. ***Columnar Databases:*** These are database structures that include only the relevant columns of the dataset, along with the key-identifying information. These are useful in speeding up some oft-sought queries from very large data sets. Suppose there is an extremely large data warehouse of web log access data, which is rolled up by the number of web access by the hour. This needs to be queried, or summarized often, involving only some of the data fields from the database. Thus the query could be speeded up by organizing the database in a columnar format. This is useful for content management systems, blogging platforms, maintaining counters, expiring usage, heavy write volume such as log aggregation. Column family databases for systems well when the query patterns have stabilized.

   HBase and **Cassandra** are the two of the more popular Columnar database offerings. HBase was developed at Yahoo, and comes as part of the Hadoop ecosystem. Cassandra was originally developed at Facebook to serve its exponentially growing user base, which is now close to 2 billion people. It was open sourced in 2008. This chapter will discuss both in detail.

2. ***Key-Value Pair Databases:*** There could be a collection of many data elements such as a collection of text messages, which could also fit into a single physical block of storage. Each text message is a unique object. This data would need to be queried often. That collection of messages could also be stored in a key-value pair format, by combining the identifier of the message and the content of the message. Key-value databases are useful for storing session information, user profiles, preferences, and shopping cart data. Key-value databases do not work so well when we need to query by non-key fields or on multiple key fields at the same time.

   Dynamo is a NoSQL highly available key-value structured storage system that has properties of both databases and distributed hash tables. Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB automatically spreads the data and traffic for your tables over enough servers to handle your throughput and storage requirements, while maintaining consistent and fast performance.

3. ***Document Databases:*** These databases store an entire document of any size, as a single value for a key element. Suppose one is storing a 10GB video movie file as a single object. An index could store the identifying information about the movie, and the address of the starting block. The system could handle the rest of storage details. This storage format would be a called document store format. Document databases are generally useful for content management systems, blogging platforms, web analytics, real-time analytics, ecommerce-applications. Document databases would not be useful for systems that need complex transactions spanning multiple operations or queries against varying aggregate structures.

   MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling. A record in MongoDB is a document, which is a data structure composed of field and value pairs. The values of fields may include other documents, arrays, and arrays of documents.

4. ***Graph Databases:*** Graph databases are very well suited for problem spaces where we have connected data, such as social networks, spatial data, routing information, and recommendation engines. For example, geographic map data used in Google Maps is stored in set of relationships or links between points. For intensive data relationship handling, graph databases improve performance by several orders of magnitude. Tech giants like Google, Facebook, and LinkedIn use graph databases to deliver scalable, insightful, and quick service.

   Neo4j is a highly scalable and most popular ACID-compliant transactional database with native graph storage and processing. It is an open-source graph database, implemented in Java, and accessible from software written in other languages.

The first popular NoSQL database was HBase, which is a part of the Hadoop family. The most popular NoSQL database used today is Apache Cassandra, which was developed and owned by Facebook till it was released as open source in 2008. Other NoSQL database systems are SimpleDB, Google's BigTable, MemcacheDB, Oracle NoSQL, Voldemort, etc.

6.3 ARCHITECTURE OF NoSQL

One of the key concepts underlying the NoSQL databases is that database management has moved to a two-layer architecture; separating the concerns of data modeling and data storage (Figure 6.3). The data storage layer focuses on the task of high-performance scalable data storage for the task at hand. The data management layer a variety of database formats, and allows for low-level access to that data through specialized languages that are more appropriate for the job, rather than being constrained by using the standard SQL format.

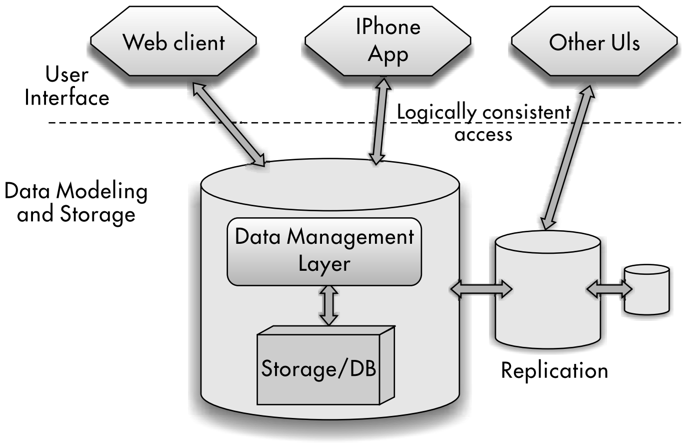


FIGURE 6.3 NoSQL databases architecture

NoSQL databases maps the data in the key/value pairs and saves the data in the storage unit. There is no storage of data in a centralized tabular form, so the database is highly scalable. The data could be of different forms, and coming from different sources, and they can all be stored in similar key/value pair formats. There are a variety of NoSQL architectures. Some popular NoSQL databases like MongoDB are designed in a master/slave model like many RDBMS. But other popular NoSQL databases like Cassandra are designed in a master-less fashion where all the nodes in the clusters are the same. So, it is the architecture of the NoSQL database system that determines the benefits of distributed and scalable system and emerges like continuous availability, distributed access, high speed, and so on.

NoSQL databases provide developers lot of options to choose from and fine tune the system to their specific requirements. Understanding the requirements of how the data is going to be consumed by the system, questions such as is it read heavy vs write heavy, is there a need to query data with random query parameters, will the system be able handle inconsistent data.

6.4 CAP THEOREM

Data is expected to be accurate and available. In a distributed environment, accuracy depends upon the consistency of data. A system is considered Consistent if all replicas of copy contain the same value. The system is considered Available, if the data I is available at all points in time. It is also desirable for the data to be consistent and available even when a network failure renders the database partitioned into two or more islands. A system is considered partition tolerant if processing can continue in both partitions in the case of a network failure. In practice it is hard to achieve all three.

The choice between Consistency and Availability remains the unavoidable reality for distributed data stores. CAP theorem states that in any distributed system one can choose only two out of the three (Consistency, Availability, and Partition Tolerance). The third will be determined by those choices.

NoSQL databases can be tuned to suit one's choice of high consistency or availability. For example, for a NoSQL database, there are essentially three parameters:

- N = replication factor, i.e. the number of replicas created for each piece of data
- R = Minimum number of nodes that should respond to a read request for it to be considered successful
- W = Minimum number of nodes that should respond to a write request before its considered successful.

Setting the values of R and W very high (R=N, and W=N) will make the system more consistent. However, it will be slow to report Consistency, and thus Availability will be low. On the other end, setting R and W to be very low (such as R=1 and W=1), would make the cluster 'highly' available, as even a single successful read (or write) would let the cluster to report success. However, consistency of data on the cluster will be low since many of these may not have yet received the latest copy of the data.

If a network gets partitioned because of a network failure, then one has to trade off availability versus consistency. NoSQL database users often choose availability and partition tolerance over strong consistency. They argue that short periods of application misbehaviour are less problematic than short periods of unavailability.

Consistency is more expensive in terms of throughput or latency, than is Availability. However, HDFS chooses consistency – as three failed DataNodes can potentially render a file's blocks completely unavailable.

6.5 HBASE

Apache HBase is a column-oriented, non-relational, distributed database system that runs on top of HDFS. An HBase system comprises a set of tables; each table contains rows and columns, much like a traditional database. Each table must have an element defined as a Primary Key; all access to HBase tables is done using the Primary Key. An HBase column represents an attribute of an object. For example, if the table is storing diagnostic logs from web servers, each row will be a log record. Each column in that table will represent an attribute such as the date/time of the record, or the server name. HBase permits many attributes to be grouped together into a column family, so that all elements of a column family are all stored as essentially a composite attribute.

Columnar databases are different from a relational database in terms of how the data is stored. In the relational database, all the columns/attributes of a given row are stored together. With HBase you must predefine the table schema and specify the column families. All rows of a column family will be stored sequentially. However, it is very flexible in that new columns can be added to families at any time, making the schema flexible and, therefore, able to adapt to changing application requirements.

6.5.1 Architecture Overview

HBase is built on master-slave concept. In HBase a master node manages the cluster, while the worker nodes (called region servers) store portions of the tables and perform the work on the data (Figure 6.4). HBase is designed after Google Bigtable, and offers similar capabilities on top of HDFS. It does consistent reads and writes. It does automatic and configurable sharding of tables. A shard is a segment of the database.

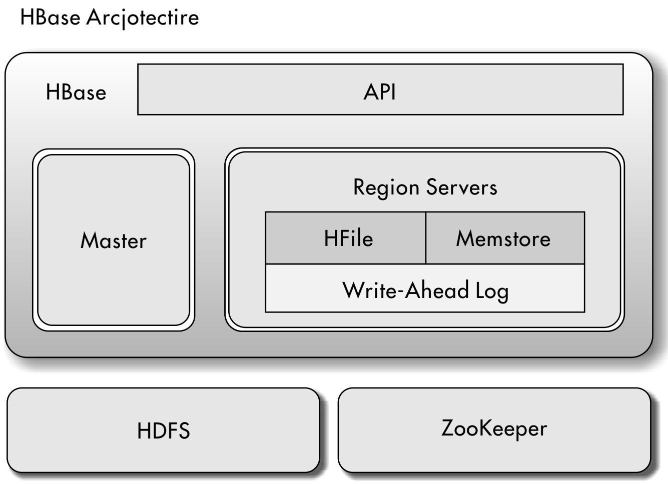


FIGURE 6.4 HBASE Architecture

Physically, HBase is composed of three types of servers in a master-slave type of architecture.

(a) The NameNode maintains metadata information for all the physical data blocks that comprise the files.

(b) Region servers serve data for reads and writes.

(c) The Hadoop DataNode stores the data that the Region Server is managing.

HBase Tables are divided horizontally, by row key range, into "Regions." A region contains all rows in the table between the region's start key and end key. Region assignment, DDL (create, delete tables) operations are handled by the HBase Master process. Zookeeper, which is part of HDFS, maintains a live cluster state. There is an automatic failover support between Region Servers. All HBase data is stored in HDFS files. Region Servers are collocated with the HDFS DataNodes, which enable data locality (putting the data close to where it is needed) for the data served by the Region Servers. HBase data is local when it is written, but when a region is moved, it is not local until compaction. Each Region Server creates an ephemeral node. The HMaster monitors these nodes to discover available region servers, and it also monitors these nodes for server failures. A master is responsible for coordinating the region servers, including assigning regions on startup, load balancing of recovery among regions, and monitoring their health. It is also the interface for creating, deleting, and updating tables.

6.5.2 Reading and Writing Data

There is a special HBase Catalog table called the META table, which holds the location of the regions in the cluster. ZooKeeper stores the location of the META table. When the first time a client reads or writes to HBase, the client gets the Region server that hosts the META table from ZooKeeper. The client will query the META server to get the region server corresponding to the row key it wants to access. The client caches this information along with the META table location. It will get the Row from the corresponding Region Server. For future reads, the client uses the cache to retrieve the META location and previously read row keys. Over time, it does not need to query the META table, unless there is a miss because a region has moved; then it will re-query and update the cache.

6.6 CASSANDRA

Apache Cassandra is a more recent and popular scalable open source non-relational database that offers continuous uptime, simplicity and easy data distribution across multiple data centers and cloud. It is a hybrid between a key-value and a column-oriented database. It provides many benefits over the traditional relational databases for modern online applications like scalable architecture, continuous availability, high data protection, multi-data replications over data centers, data compression, SQL like language and so on.

6.6.1 Architecture Overview

Cassandra architecture provides its ability to scale and provide continuous availability (Figure 6.5). Rather than using master-slave architecture, it has a master-less "ring" design that is easy to set up and maintain. In Cassandra, all nodes play an equal role, all nodes communicate with one another by a distributed and highly scalable protocol called gossip. The read and write processes are summarized in Figure 6.6.

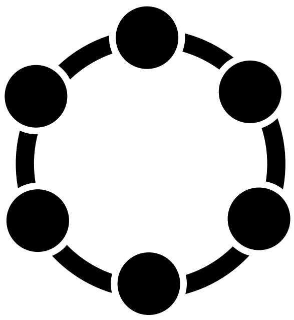


FIGURE 6.5 Cassandra architecture

So, the Cassandra scalable architecture provides the capacity of handling large volume of data, and large number of concurrent users or operations occurring at the same time, across multiple data centers, just as easily as a normal operation for the relational databases. To enhance its capacity, one simply needs to add new nodes to an existing cluster without taking down the system and designing from

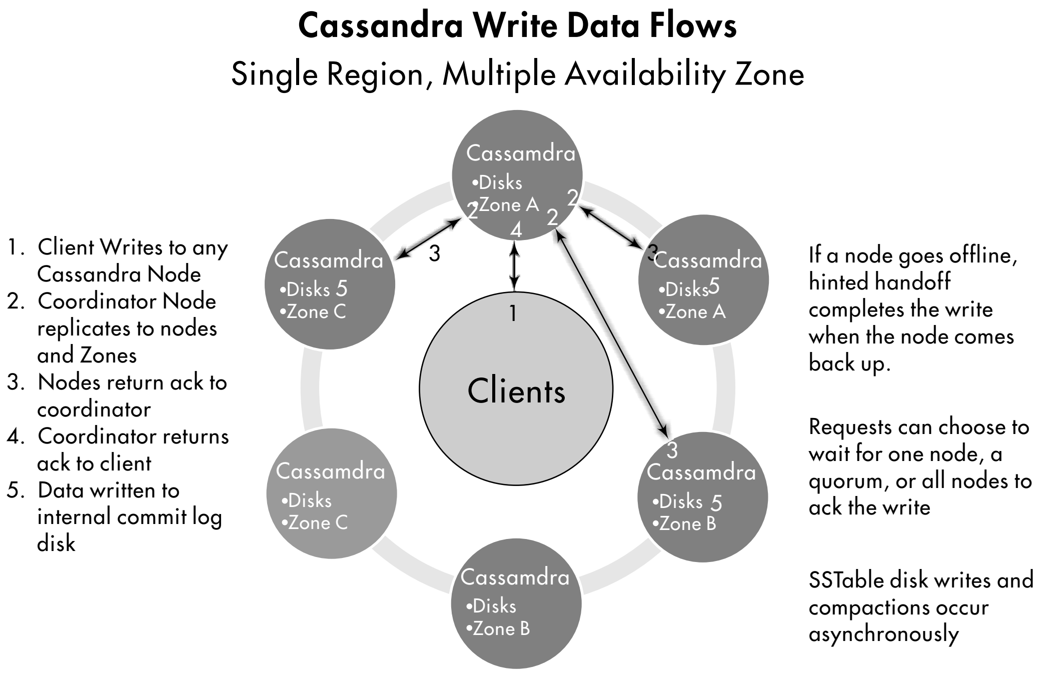

Cassandra Write Data Flows
Single Region, Multiple Availability Zone
1. Client Writes to any Cassandra Node
2. Coordinator Node replicates to nodes and Zones
3. Nodes return ack to coordinator
4. Coordinator returns ack to client
5. Data written to internal commit log disk
Cassamdra •Disks •Zone A
Cassamdra •Disks •Zone C
Cassamdra •Disks •Zone A
Cassamdra •Disks •Zone C
Cassamdra •Disks •Zone B
Cassamdra •Disks •Zone B
Clients
1
2
3
4
5
If a node goes offline, hinted handoff completes the write when the node comes back up.
Requests can choose to wait for one node, a quorum, or all nodes to ack the write
SSTable disk writes and compactions occur asynchronously

FIGURE 6.6 Cassandra read and write processes

the scratch. Also the Cassandra architecture means that unlike other master slave systems, it has no single point of failure and thus is capable of offering continuous availability and uptime.

6.6.2 Protocols

Cassandra was designed to be fault-tolerant. Its peer-to-peer, distributed system results in read/write anywhere design. Data is transparently partitioned among all nodes in the cluster. All the nodes are similar and equal. Each node communicates with other through the GOSSIP protocol, which exchanges information across the cluster every second. A commit log is used on each node to capture every write activity. All data is written to the commit log first for durability. When multiple updates are applied to the same column, Cassandra uses client-provided timestamps to resolve conflicts.

Data of update/delete operations is written in key/value pairs format to an in-memory structure called memTable. The memTable is written to a disk file called a sorted string table (SSTable). An SSTable is an immutable, append-only, data file to which Cassandra writes memtables periodically. After all the commit log's data has been flushed from the memtable to the SStable, the it is archived, deleted, or recycled (Figure 6.7).

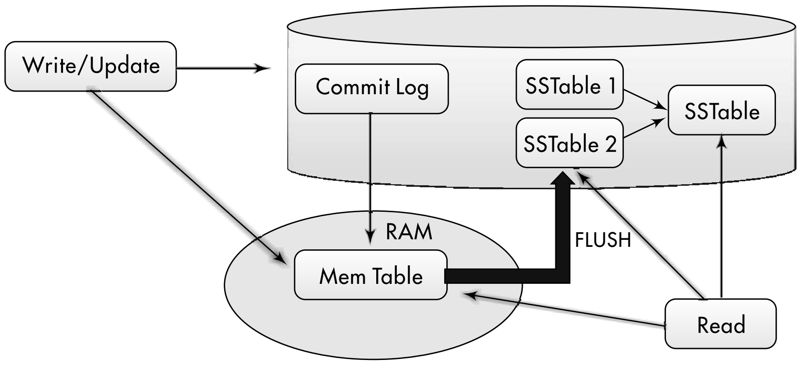


FIGURE 6.7 Cassandra protocols

All data is indexed for faster access. A row in a column family is indexed by its key. The index summary is loaded into the memory. A lookup for actual rows can be performed with a single disk seek and by scanning sequentially for the data. Operations are provided to look up the value associated with a specific key and to iterate over all the column names and value pairs within a specified key range.

6.6.3 Data Model

In Cassandra world, the data model can be seen as a map which is distributed across the cluster. In other words, a table in Cassandra is a distributed multi-dimensional map indexed by a key. The data values can be as columns, column family, or Super columns families.

A column is the smallest container in Cassandra world with following properties, Name and Value and Timestamp. A Super column is a tuple with Name and Value. Thus a value can map to many columns. A Column family is a collection of rows and columns (like an entire table in a traditional RDBMS). Each row in a column family is uniquely identified by a row key (Figure 6.8). Each row can have multiple columns, each of which has a name, value and a timestamp. Unlike traditional RDBMS, different rows in the same column family need not to share the same set of columns. A column may be added to one or multiple rows at any time without affecting the complete dataset. In a column family there can be billions of columns.

A super column contains multiple column families. It is a tuple (pair) that consists of a key-value pair, where the key is mapped to value which are column families. It is something like a “view” on more than one tables. A key space contains the column families just like a database contains tables in relational world. They are used to group column families together. In traditional relational database analogy key spaces can be seen as database schema. It is the outermost grouping of the data in the data store. Generally, in a cluster there is one key space per application.

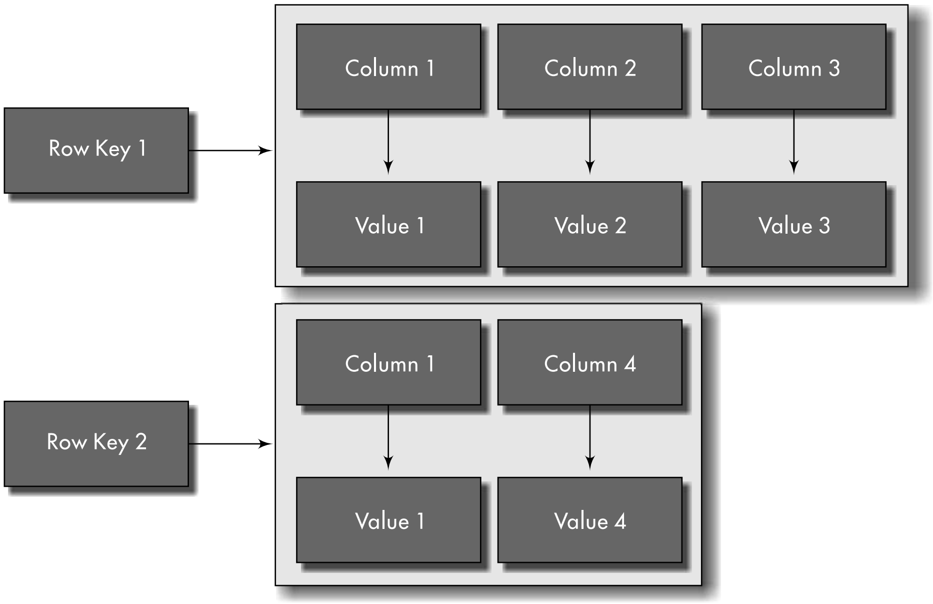


FIGURE 6.8 Cassandra data model

6.6.4 Cassandra Writes

There are two write modes in Cassandra. A Quorum Write is a synchronous operation in which client blocks until the quorum is reached i.e. the data is propagated to all the nodes in quorum. An Asynchronous Write sends the request to any node, further that node will push the data to appropriate nodes but return to client immediately hence client is not blocked in this case.

In a write operation, the Client sends a write request to a single, random Cassandra node. This node writes the data to the cluster. The writes are then replicated to N nodes using a configurable replication strategy to minimize the possibility of data loss. Cassandra will determine the distance of other nodes from the current node. The closest node is on the same rack. The next distant node is in the same data center. The most distant node is in an entirely different data center (Figure 6.9). Consistency can be chosen between strong and eventual (from all to any node responding) depending on the need. As the data files accumulate over time, they are periodically merged and sorted into a new file hence creating new index.

6.6.5 Cassandra Reads

The client can select the strength of the read consistency. A Single Read means the client request returns once it gets the first response, even though the data can be

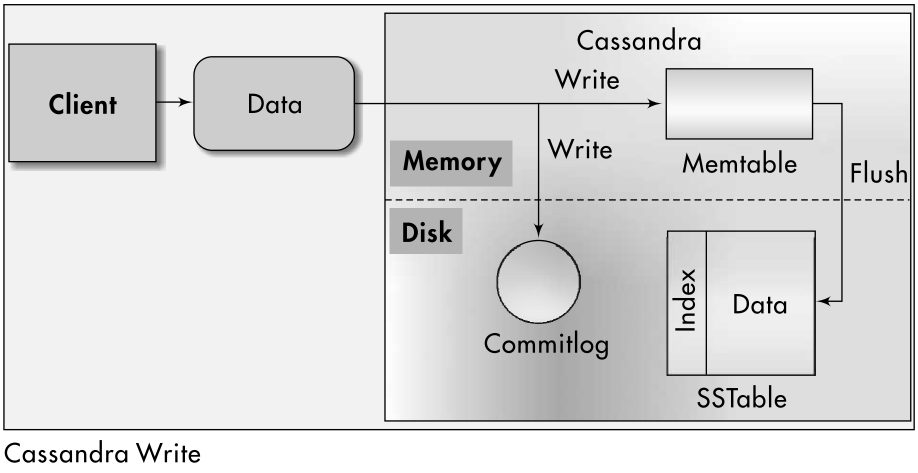


FIGURE 6.9 Cassandra write process

stale. In a Quorum Read the request returns only after most of the nodes responded with the same value, and this reduces the chances of getting stale data.

A random node receives the read request from the client. This node determines the nodes that have copies of data, and then requests the data from each node (Figure 6.10). When a read request comes in to a node, the data to be returned is merged from all the related SSTables and any unflushed memtables.

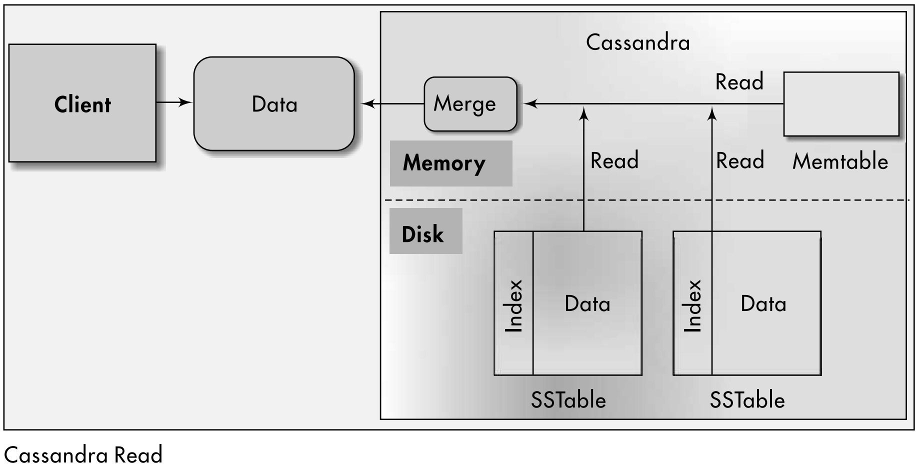


FIGURE 6.10 Cassandra read process

6.6.6 Replication

The term replication means how many copies of each piece of data we need in our cluster. It is the process of storing copies of data on multiple nodes to ensure reliability and fault tolerance. Basically, there are two replication strategies are available. A Simple strategy should be used for a single data centre only. It places the first replica on a node determined by the partitioner. Additional replicas are placed on the next nodes clockwise in the ring without considering topology, i.e. rack or data centre location. The total number of replicas across the cluster are referred as the replication factor. All replicas are equally important for failover as there is no primary or master replica in Cassandra.

The partitioner controls how the data is distributed over your nodes. To find a set of keys, Cassandra must know what nodes have the range of values client is looking for. A partitioner is a hash function for calculating the token or hash of a row key to replicate the data in cluster. Each row of data is uniquely identified by a row key and distributed across the cluster by the value of the token. The token generated by the partitioner is further used by replication strategy to place the replica in the cluster.

Consistency level in Cassandra can be seen as how many replicas must response to declare a successful read or write operation. Consistency refers to how up-to-date and synchronized a row of Cassandra data is on all its replicas. Since Cassandra extends the concept of eventual consistency by offering tunable consistency, hence for any given read or write operation, the client application decides how consistent the requested data must be.

Write Syntax

```
TTransport tr = new TSocket(HOST, PORT);

TFramedTransport tf = new TFramedTransport(tr);
TProtocol protocal = new TBinaryProtocol(tf);
Cassandra.Client client = new Cassandra.Client(protocol);

tf.open();

client.insert(userIDKey, cp,   new Column("Colume-name".getBytes(UTF8),   "Colume-
data".getBytes(), clock), CL);
```

Read Syntax:

```
Column col = client.get(userIDKey, colPathName, CL).getColumn();
LOG.debug("Column name: " + new String(col.Colume-name, UTF8));
LOG.debug("Column value: " + new String(col.Colume-data, UTF8));
```

6.7 CONCLUSION

NoSQL databases emerged in response to the limitations of relational databases in handling the sheer volume, nature, and growth of data. NoSQL databases have the functionality like MapReduce. NoSQL database is proving to be a viable solution to the enterprise data needs and continue to do so. There are four types of NoSQL databases: columnar, Key-pair, document, and graphical databases. Cassandra and HBase are among the most popular NOSQL databases. Hive is an SQL-type language to access data from NoSQL databases. Pig is a procedural high-language that gives greater control over data flows.

Review Questions

1. What is a NoSQL database? What are the different types of it?
2. How does a NoSQL database leverage the power of MapReduce?
3. What are different kinds of NoSQL databases? What are the advantages of each?
4. What are the similarities and differences between Hive and Pig?

True/False Questions

1. Relational databases are better than NoSQL databases in managing large and unpredictable streams of data.
2. NoSQL databases support the ACID (Atomicity, Consistency, Isolation, Durability) properties associated with transaction processing.
3. Document databases are useful for storing session information, user profiles, preferences, and shopping cart data.
4. Key-value NoSQL Databases are a popular form of NoSQL databases, for content management, blogging platforms, etc.
5. Graph databases are very well suited for social network analysis applications.
6. NoSQL databases have a two-layer architecture, separating the concerns of data modeling and data storage.
7. All NoSQL databases work in a master/slave model.
8. A system is considered consistent only if all replicas of copy contain the same value.

9. A system is considered partition tolerant if processing can continue in both partitions in the case of a network failure.
10. Apache HBase is a column-oriented, non-relational, distributed database system.
11. HBase is built on a master-slave concept.
12. HBase serves data for reads and writes data through Region Servers.
13. Apache Cassandra is a scalable NoSQL database that offers continuous uptime, simplicity and easy data distribution across multiple data centers and cloud.
14. Cassandra was developed at Google.
15. Cassandra uses ring-architecture to arrange data servers.
16. Cassandra looks a "Bloom filter" to speed up data access.
17. Pig and Hive are popular languages that work well on NoSQL databases.
18. Hive was designed by FaceBook.
19. Hive allows almost all SQL operations.
20. For analytical needs, Pig is preferable over Hive.

Chapter 7

Stream Processing with Spark

Learning Objectives

- Recognize the need for stream processing
- Discuss Apache Spark and its architecture
- Describe RDD and DAG as major features of Spark
- Understand MLLib and GraphX libraries within Spark
- Discuss many use cases for Spark

Introduction

A stream processing system is a clever way to process large quantities of data from a vast set of extremely fast incoming data streams. The ideal stream processing engine will capture and report in real time the essence of all data streams, no matter the speed or size or number of streams. This can be achieved by using innovative algorithms and filters that relax many computational accuracy requirements, to compute simple approximate metrics in real time. Stream processing engine aligns with the infinite dynamism of the flow of nature.

Apache Spark is an integrated, fast, in-memory, general-purpose engine for large-scale data processing. Spark is ideal for iterative and interactive processing tasks on large data sets and streams. Spark achieves 10–100x performance over Hadoop by operating with an in-memory data construct called Resilient Distributed Datasets (RDDs). It helps avoid the latencies involved in disk reads and writes. Spark also offers built-in libraries for Machine Learning, Graph Processing, Stream processing and SQL to deliver seamless superfast data processing along with high programmer productivity. Spark is compatible with Hadoop file systems and tools. Spark is an alternative to MapReduce rather than a replacement for Hadoop file system. It has

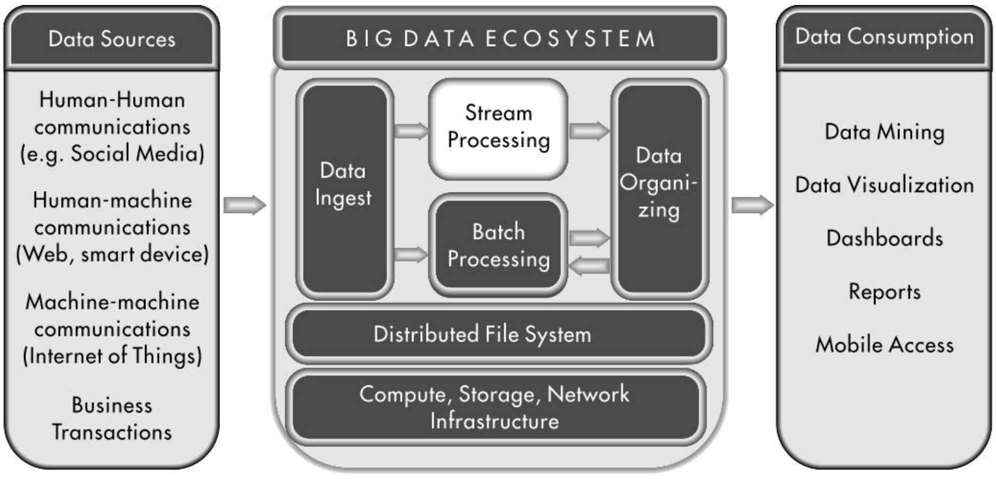


FIGURE 7.1 Big data architecture

also become a more efficient and productive alternative for Hadoop ecosystem, and is increasing being used in industry.

Apache Spark was developed in 2009 in UC Berkeley's AMPLab, and open sourced in 2010 as an Apache project. It can process data from a variety of data repositories, including the Hadoop Distributed File System (HDFS), and NoSQL databases such as HBase and Cassandra. Spark prioritizes in-memory processing to boost the performance of big data analytics applications, however, it can also do conventional disk-based processing when data sets are too large to fit into the available system memory. Spark enables applications in Hadoop clusters to run up to 100 times faster in memory and 10 times faster even when running on disk. Spark gives us a comprehensive, unified solution to manage Big Data processing requirements with a variety of use cases, and data sets that are diverse in nature (text data, graph data etc.) as well as the source of data (batch vs. real-time streaming data).

7.1 SPARK ARCHITECTURE

The core Spark engine functions partly as an application programming interface (API) layer and underpins a set of related tools for managing and analyzing data. These include a SQL query engine, a library of machine learning algorithms, a graph processing system, and streaming data processing software (Figure 7.2). Spark allows programmers to develop complex, multi-step data pipelines using directed acyclic graph (DAG) pattern. It also supports in-memory data sharing across DAGs, so that different jobs can work with the same data. Spark runs on top of existing HDFS infrastructure to provide enhanced and additional functionality.

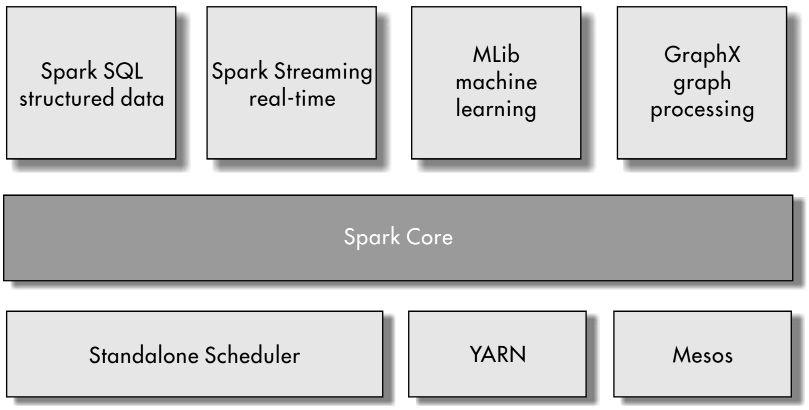


FIGURE 7.2 Spark architecture

Next, we explain the two important innovations in Spark: RDDs and DAG.

7.1.1 Resilient Distributed Datasets (RDDs)

RDDs, Resilient Distributed Datasets, is a distributed memory construct. These are motivated by two types of applications that current computing frameworks handle inefficiently: Iterative algorithms, and Interactive data mining tools. In both cases, keeping data in memory can improve performance by an order of magnitude.

RDDs are immutable and partitioned collection of records. They can only be created by coarse grained operations such as map, filter, group by, etc. Coarse grained operations mean that the operations are applied on all elements in a dataset. RDDs can only be created by:

(a) Reading data from a stable storage such as HDFS or

(b) Transformations on existing RDDs.

Once data is read into an RDD object in Spark, a variety of operations can be performed on the RDD by invoking abstract Spark APIs. The two major types of operation available are transformations and actions.

(a) Transformations return a new, modified RDD based on the original. Several transformations are available through the Spark API, including map (), filter (), sample (), and union ().

(b) Actions return a value based on some computation being performed on an RDD. Some examples of actions supported by the Spark API include reduce (), count (), first (), and for each ().

7.1.2 Directed Acyclic Graph (DAG)

DAG refers to a Directed Acyclic Graph. This approach is an important feature for real-time data processing platforms such as Spark, Storm, and Tez, and helps them offer amazing new capabilities for building highly interactive, real-time computing systems to power real-time BI, predictive analytics, real-time marketing, and other critical systems.

DAG Scheduler is the scheduling layer of Apache Spark that implements stage-oriented scheduling, i.e. after an RDD action has been called it becomes a job that is then transformed into a set of stages that are submitted as task-sets for execution. In general, DAG Scheduler does three things in Spark:

(a) Computes an execution DAG, i.e. DAG of stages, for a job;

(b) Determines the preferred locations to run each task on;

(c) Handles failures due to shuffle output files being lost.

7.2 SPARK ECOSYSTEM

Spark is an integrated stack of tools responsible for scheduling, distributing, and monitoring applications consisting of many computational tasks across many worker machines, or a computing cluster. Spark is written primarily in Scala, but includes code from Python, Java, R, and other languages. Spark comes with a set of integrated tools that reduce learning time and deliver higher user productivity. Spark ecosystem includes Mesos resource manager, and other tools.

Spark has already overtaken Hadoop in general because of benefits it provides in terms of faster execution in iterative processing algorithms.

7.3 SPARK FOR BIG DATA PROCESSING

Spark support big data mining through many tools such as MLlib, GraphX, SparkR, Spark SQL, and Streaming library.

7.3.1 MLlib

MLlib is Spark's machine learning library. It consists of basic machine learning algorithms such as classification, regression, clustering, collaborative filtering, dimensionality reduction, lower-level optimization primitives, and higher-level pipeline APIs. RDDs help Spark excel at iterative computation, thus enabling MLlib to run fast. MLlib contains high-quality algorithms that leverage iteration, and can yield better

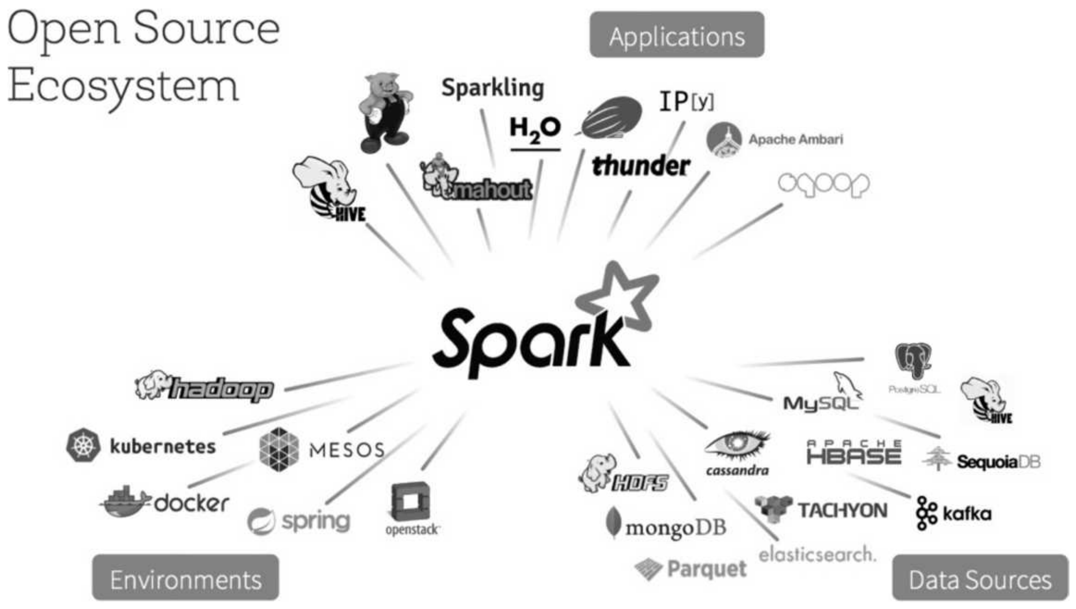


FIGURE 7.3 Spark ecosystem

results than the one-pass approximations sometimes used on MapReduce. In addition, Spark MLlib is easy to use and it can support Scala, Java, Python, and SparkR.

For example, decision trees is a popular data classification technique, Spark MLlib can support decision trees for binary and multiclass classification, using both continuous and categorical features. The implementation partitions data by rows, allowing distributed training with millions of instances.

7.3.2 Spark GraphX

Efficient processing of large graphs is another important and challenging issue. Many practical computing problems concern large graphs. For example, Google runs its PageRank on tens of billions of webpages and maybe a trillion weblinks. GraphX is Spark's component for graphs and graph-parallel computation. At a high level, GraphX extends the Spark RDD by introducing a new Graph abstraction: a directed multi-graph with properties attached to each vertex and edge.

To support graph computation, GraphX exposes a set of fundamental operators such as subgraph, joinVertices, and aggregateMessages on the basis of an optimized variant of the Pregel API (Pregel is the graphs processing system at Google that powers PageRank). In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks.

One can compute the PageRank of nodes in a graph as follows, using Scala on Spark:

```
//load the edges as a graph object
val graph = GraphLoader.edgeListFile (sc, "outlink.txt")
// Run Pagerank
val ranks = graph.pagerank(0.0001).vertices
// join the rank with the webpages
val pages = sc.textFile("pages.txt").map{line => val fields = line.split(",")
(fields(0).toLong, fields(1)) }
val ranksByPagename = pages.join(ranks).map { case (id, (pagename, rank)) =>
(pagename, rank)}
//print the output
println(rankByPagename.collect().mkString("\n"))
```

7.3.3 SparkR

R is a popular statistical programming language with several extensions that support data processing and machine learning tasks. However, interactive data analysis in R is usually limited as the runtime is single-threaded and can only process data sets that fit in a single machine's memory. SparkR is an R package initially developed at the AMPLab to provide an R frontend to Apache Spark. It uses Spark's distributed computation engine to run large scale data analysis from the R shell. SparkR exposes the RDD API of Spark as distributed lists in R.

For example, one can read an input file from HDFS and process every line using *lapply* on a RDD, to compute space-separated words per line.

```
sc<- sparkR.init("local")
lines <- textFile(sc, "hdfs://data.txt")
wordsPerLine <- lapply(lines, function(line)) { length(unlist(strsplit(line," ")))}
```

In addition to lapply, SparkR also allows closures to be applied on every partition using lapplyWithPartition. Other supported RDD functions include operations like reduce, reduceByKey, groupByKey and collect.

7.3.4 SparkSQL

Spark SQL is a language provided to deal with the structured data. Using this one can run queries on the data and get some meaningful result. It supports queries through SQL as well as HiveQL (Apache's Hive version of SQL).

7.3.5 Spark Streaming

Spark Streaming gains data streams from input sources, processes them in a cluster, and pushes out to databases/ dashboards. Spark further chops up data streams into batches of few seconds. Spark treats each batch of data as RDDs and processes them using RDD operations. The processed results are pushed out as batches.

7.4 SPARK APPLICATIONS

Some real uses cases that are solved well by a tool like Apache Spark include:

1. Real-time Log Data monitoring.
2. Massive Natural Language Processing.
3. Large Scale Online Recommendation Systems.

A simple Wordcount application can be run in Spark shell as below.

```
val textFile = sc.textFile("C:\\Users\\MyName\\Documents\\obamaSpeech.txt")
//Comment: saves the text file as textFile
val counts = textFile.flatMap(line => line.split(" ")).map(word => (word, 1))
.reduceByKey(_ + _)
//Comment: Calculate the total words by splitting by space
counts.count();
//Results the output as below
Long = 52
counts.saveAsTextFile("C:\\Users\\MyName\\Desktop\\counts1")
//Comment: saves the file on Desktop
```

7.4.1 Spark vs Hadoop

Spark and Hadoop are both popular top-level Apache projects dedicated to Big Data processing. For many years, Hadoop was the leading open source Big Data platform, and many companies already use a distributed computing framework, e.g. Hadoop based on MapReduce. However, Spark is gradually gaining attention, and has acquired more than 50% market share. Table 7.1 provides a comparison between Hadoop and Spark.

Table 7.1

Comparison between Hadoop and Spark

Feature	Hadoop	Spark
Purpose	Resilient cost-effective storage and processing of large data sets	Fast general-purpose engine for large-scale data processing
Core component	Hadoop Distributed File system (HDFS)	Spark Core, the in-memory processing engine
Storage	HDFS manages massive data collections across multiple nodes within a cluster of commodity servers.	Spark doesn't do distributed storage. It operates on distributed data collections
Fault tolerance	Hadoop uses replication to achieve fault tolerance	Spark uses RDD for fault tolerance that minimizes network I/O
Nature of processing	Accompanied by MapReduce, it includes batch processing of this data in parallel mode	Batch as well as stream processing
Sweet spot	Batch processing	Iterative and interactive processing jobs, that can fit in the memory
Processing speed	Map Reduce is slow	Spark can be up to 10x faster than MapReduce for batch processing and up to 100x faster for stream processing
Security	More secure	Less secure
Failure recovery	Hadoop can recover from system faults or failures since data are written to disk after every operation	With Spark, data objects are stored in RDD. These can be reconstructed after faults or failures
Analytics tools	Separate engine	Built-in MLLib (Machine Learning) and GraphX (Graph Processing) libraries
Compatibility	Primary storage model is HDFS	Compatibility with HDFS and other storage formats
Language support	Java	Scala is native language. APIs for python, java, R, others
Driving organization	Yahoo	AMPLabs from UCBerkeley
Technology owners	Apache, Open-source, free	Open-source, free
Key distributors	Cloudera, Horton, MapR	Databricks, AMPLabs
Cost of system	Medium to High	Medium to High

7.5 CONCLUSION

Spark is a new integrated system for Big Data processing. Its most important core abstraction is Resilient Distributed Datasets (RDDs), an in-memory construct that provides 10–100x processing speed over Hadoop. Directed Acyclic Graphs (DAG) based execution engine delivers superior execution speed. Important libraries like MLlib and GraphX make Spark a powerful open source processing engine that delivers speed, ease of use, and sophisticated analytics.

Review Questions

1. Describe the Apache Spark ecosystem.
2. Compare Spark and Hadoop in terms of their ability to do stream computing?
3. What is an RDD? How does it make Spark faster?
4. Describe three major capabilities in Spark for data analytics.

True/False Questions

1. Apache Spark is an integrated, in-memory, general-purpose engine for batch processing.
2. Spark achieves 10–100x performance over Hadoop by operating with an in-memory construct called 'Resilient Distributed Datasets'.
3. Spark has built-in libraries for Machine Learning and Graph Processing.
4. DAG is a directed acyclic graph for Apache Spark that implements stage-oriented scheduling.
5. Spark is written primarily in R, but includes code from Python, Java, Scala, and other languages.
6. Spark excels at iterative computation, enabling MLlib to run fast.
7. GraphX extends the Spark RDD by introducing a new Graph abstraction: a directed multi-graph.
8. Spark Streaming gains data streams from input sources, processes them in a cluster, and pushes out the data directly to the users.
9. Spark was developed at the MIT's High Performance Computing Lab.
10. Apache Spark is a good system for transaction processing.

Chapter 8

New Ingesting Data

Learning Objectives

- Recognize need for a Big Data ingest system
- Describe messaging system architectures
- Analyze Kafka system, its architecture, and key concepts of producers, consumers, brokers, and topics
- Understand the use cases of Kafka
- Know Zookeeper and its support role for Kafka

Introduction

Big Data arrives at unpredictable speeds and quantities. An ingest system receives data and communicates it to target applications. All data should be smoothly received, and made available for target applications to access securely and reliably at their own convenience. An ideal ingest system would be flexible and scalable to receive data from all types of sources, at any time and speed and in any quantity; and make it available to all target applications without loss of data or speed.

A Data ingest system is a reliable and efficient point of reception for all data coming into a system. Incoming data is treated like messages, and is put into an organized set of locations, from where the target applications can fetch them when they are ready. An effective data ingest mechanism is achieved by creating a fast and flexible buffer for receiving and storing all incoming streams of data. The data in the buffer is stored in a sequential manner, and its contents are made available to all consuming applications in a fast and orderly manner.

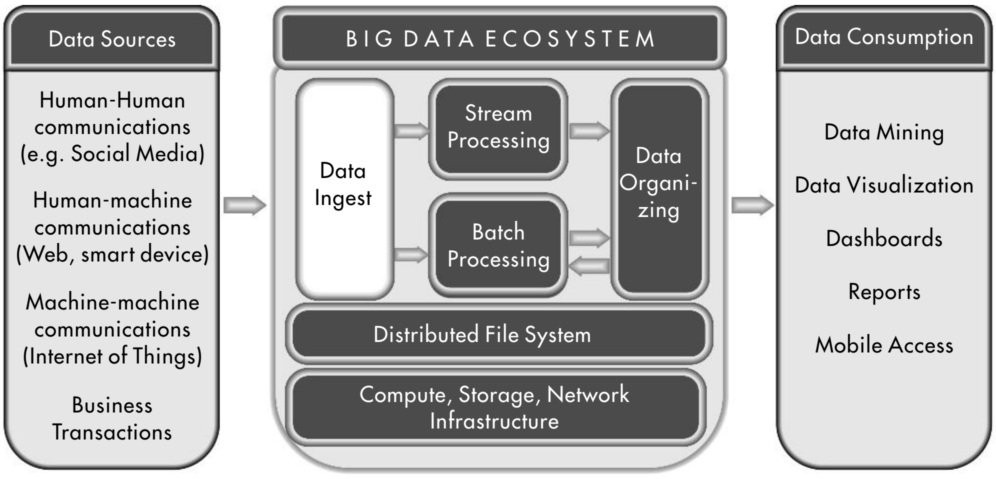


FIGURE 8.1 Big data architecture

8.1 MESSAGING SYSTEMS

A Messaging System is an asynchronous mode of communicating data between application. There are two generic kinds of messaging systems – a point-to-point system, and a publish-subscribe (pub-sub) system. Most of the messaging patterns now follow pub-sub model.

With huge amounts of data coming in from different sources, and many more consuming applications, a point-to-point system of delivering messages from source to target, becomes inadequate and slow. Alternatively, incoming data can be categorized into certain subjects or topics, and it could be stored in the respective location or locations for those topics. Now the data is available for consumption by any application that is interested in data related to a topic. Each consuming application can choose to read data about one or more topics of its interest. This is called the publish-and-subscribe system.

8.1.1 Point to Point Messaging System

In a point-to-point system, every message is directed at a particular receiver. A common queue can receive messages from many producers or messages. Any particular message can be received and consumed by only one receiver. Once that target consumer reads a message in the queue, that message disappears from that queue. The typical example of this system is an Order Processing System, where each order will be processed by one Order Processor.

8.1.2 Publish-Subscribe Messaging System

In a pub-sub messaging system, the applications publish their output to a standard messaging queue. The target recipient will only need to know where to get the message, whenever it is ready to pick up the message. Applications thus can ignore the mechanics of interaction with other applications, and simply care about the message itself. This is especially valuable when there may be many target recipients for a message. In a pub-sub system, messages are entered into the messaging queue asynchronously from client applications. A pub-sub system needs to be fast and secure to serve many applications, both producers and subscribers. Messages are also replicated across multiple locations for reliability of data.

8.2 DATA INGEST SYSTEMS

There are two popular Data ingesting systems used in Big Data. An older system, called Apache Flume, is closely tied to the Hadoop. Flume is a simple, robust, and extensible tool for data ingestion from various data producers and storing into Hadoop distributed file system. Flume agents collect data from data sources, aggregate it, and pushed into a centralized store such as HDFS or HBase.

A new and more popular system is a general-purpose pub-sub system called Apache Kafka. It can connect to all types of sources and target applications. In this chapter we will discuss the new system, Kafka.

8.3 APACHE KAFKA

Apache Kafka is an open source publish-and-subscribe message broker system. Kafka aims to provide an integrated high-throughput, low- latency messaging platform for handling real-time data feeds. In the abstract, it is a single point of contact between all producers and consumers of data. All producers of data send data to Kafka. All consumers of data read data from Kafka (Figure 8.2).

Kafka is a distributed, partitioned, scalable, replicated messaging system, with a simple but unique design. It was initially developed by LinkedIn and was open sourced to Apache in 2011. It is written in Scala programming language. Kafka is a valuable for an enterprises level infrastructure because of its simplicity and scalability.

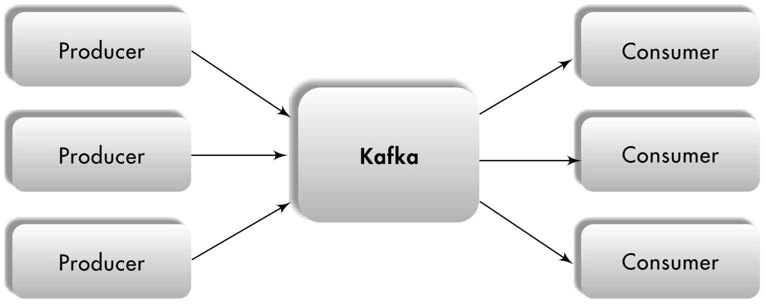


FIGURE 8.2 Kafka core idea

8.4 USE CASES

The very general nature of Kafka's design makes it suitable for many use cases.

1. *Messaging*

Kafka is a very good alternative for a traditional messaging system because Kafka messaging system has better throughput, built in partitioning, replication, and better fault tolerance. Kafka is very good solution for a large scale message processing applications.

2. *Website Activity Tracking*

Website Activity Tracking was one of initial use cases for Kafka for LinkedIn. Users' online activity tracking pipeline was rebuilt as a set of real time data feeds. Web activity tracking includes very large volume of data, and Kafka is very good at handling huge volumes of data. User activity types such as page view, searches, clicks, etc. can be designated as central topics, and the activity data can be published to those topics. Those events are available for real time or offline processing and reporting.

3. *Stream Processing*

Popular frameworks such as Storm and Spark Streaming can read data from a Kafka topic, process it, and send to other users and consumer applications. They may even write it back to Kafka to a new topic. Kafka's strong durability is also very useful for stream processing.

4. *Log Aggregation*

Log aggregation typically gathers physical log files from many servers and puts them all in a central location for processing. Kafka can abstract away the details of the files, and provide a clean abstraction of log data as a stream of messages. Kafka then allows for lower-latency processing and easier support for multiple data sources and distributed consumption of this data. Unlike dedicated log-centric systems, Kafka offers higher performance and stronger durability guarantees due to data replication.

5. *Commit Log*

Kafka can be used as external commit log for a distributed database system. This audit log can help to re-sync data between the failed nodes to restore their data. The log compaction in Kafka helps to achieve this feature more efficiently.

8.5 KAFKA ARCHITECTURE

In the abstract, Kafka is a master broker (or a collection of brokers) that deals with all producers and consumers of data. A producer pushes data into Kafka system at its own speed, scale and convenience. A consumer pulls data out of the system at its own speed, scale and convenience. All the received data in Kafka is organized by categories, called topics. Incoming data is sorted and stored into topic servers. The topic servers are replicated for reliability and ease of access. The consumers of data can subscribe to one or more topics (Figure 8.3).

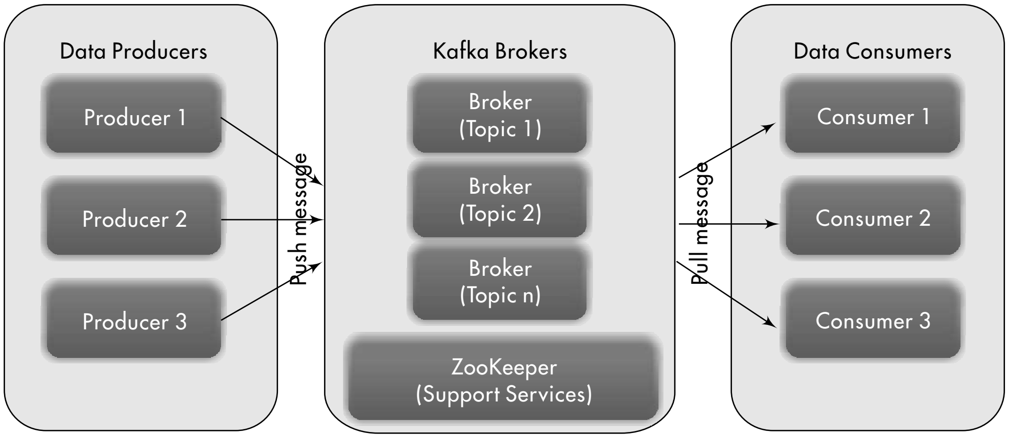


FIGURE 8.3 Kafka ecosystem

In practice, there are many brokers (also called servers, or partitions) for each topic, for reliability of the messaging system. Thus two or more brokers will store data on each topic. One broker is assigned as a leader for a topic. Only one broker can be leader at any given time. In the lead broker fails, then a second one can automatically take over as the leader and prevent the loss of access to data.

Kafka is designed for distributed high throughput systems. In comparison to other messaging systems, Kafka allows almost no processing of data in its servers. That gives it high throughput. It has built-in partitioning to provide infinite scalability. Adding new broker is as simple as adding new partitions on new storage. It has built-in replication of data, and thus delivers fault-tolerance. All these features make it a good fit for large-scale message processing applications. Kafka is very fast and can perform 2 million writes/second. It also guarantees zero downtime and zero data loss.

Kafka also can handle many diverse consumers. It integrates very well with Apache Storm, Spark, and other real-time streaming data applications. There are many contributing organizations helping to improve the Kafka open-source system. It has very well documented online resources. Many big organizations have used it such as LinkedIn, Cisco, Spotify, PayPal, HubSpot, Shopify, Uber and more. HubSpot, for example, uses Kafka to deliver real time notification of when a recipient opens their email.

8.5.1 Producers

A data producer is responsible for selecting the topic, and partition, the message that it wants to convey. The producer can use round-robin algorithm to balance the load among partitions. There can be both synchronous and asynchronous producers of messages. All the messages are published to the partition. It is automatically replicated to other partitions.

8.5.2 Consumers

A data consumer is responsible for reading the data about the topic that it has subscribed. The consumer is responsible for reading the data within a reasonable period of time, before the queues are emptied for efficient management of storage. Different consuming applications can read the data at different times. Kafka has stronger ordering guarantees than a traditional messaging system. A consumer needs to know how far it has read in that queue, so as to avoid duplicates or lose some data.

8.5.3 Broker

A broker is a server in a Kafka cluster. The cluster may have many servers or brokers. Each broker can serve one of more producers, and one or more topics.

8.5.4 Topic

A topic is a category of messages, into which incoming messages are published. For each topic there is a separate partition log for storage of messages. Each partition has an ordered sequence of messages for that topic. Each message in the partition is assigned a unique sequential number, also called the offset. This offset helps to identify each message within the partition (Figure 8.4).

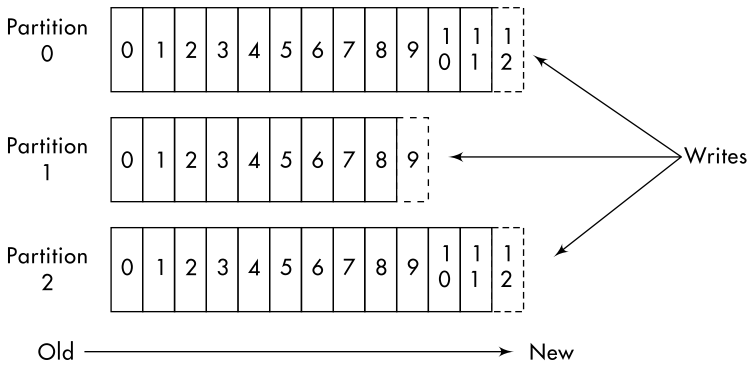


FIGURE 8.4 Anatomy of a topic

The consumer reads the data sequentially according to offset numbers. The consumer is responsible for maintaining the offset to remember how far it has read. Generally, the offset increases linearly as messages are consumed. However, a consumer can reset offset to access the data gain and reprocess it as needed.

The Kafka cluster retains all the published messages whether or not they have been consumed, for a configurable period or not. For example, if the log retention is set to seven days, then for the seven days after publishing, the message is available for consumption. After seven days, Kafka discards the messages to free up space.

Kafka's performance is not affected by the size of data. Each partition must fit on the servers that host it, but a topic may have multiple partitions. This enables Kafka to manage an arbitrary amount of data. Also, it acts as the unit of parallelism.

8.5.5 Summary of Key Attributes

1. *Disk based*: Kafka works on a cluster of disks. It keeps writing to the disk to make the storage permanent.
2. *Fault tolerant*: Data in Kafka is replicated across multiple brokers. When any leader broker fails, a follower broker takes over as leader and everything continues to work normally.
3. *Scalable*: Kafka can scale up easily by adding more partitions or more brokers. More brokers help to spread the load and this provides greater throughput.

4. *Low latency*: Kafka does very little processing on the data. Thus it has very low latency rate. Messages produced by the consumer are published and available to the consumer within a few milliseconds.
5. *Finite Retention*: Kafka by default keeps the message in the cluster for a week. After that the storage is refreshed. Thus the data consumers have up to a week to catch up on data, in case they fall behind for any reason.

8.5.6 Data Replication

The Kafka cluster maintains multiple servers over the distributed network. The partitions of the log are maintained over this network. Each server handles data and requests for a share of the partitions. Each partition is replicated across a configurable number of servers for fault tolerance. But one of the server for each partition acts as the main server also called "leader" while it may or may not have one or more secondary server also known as "followers". The leader server is responsible for handling all the read and write operation for the partition while the followers silently replicates the leader. The follower server becomes very helpful when the leader server fails. The follower server automatically becomes the leader and then handles the failure. One server can be a leader for some of the partitions on it, while it may be follower for other partitions. Thus one server can act as both leader and follower. This helps to balance the work load on the servers within the cluster.

8.5.7 Guarantees

Data sent always maintain the order they were sent. For example, if a message M1 and M2 were sent by the same producer and M1 was sent first then the message M1 will have lower offset than message M2. Therefore, M1 will always appear before the M2 for the consumer.

Each topic has a replication factor N and the system can tolerate up to N-1 server failures without losing any messages committed to the log.

8.5.8 Client Libraries

Kafka supports following client libraries:

1. Python: Pure python implementation with full protocol support

3. C: High performance C library with full protocol support

3. C++, Ruby, JavaScript and more.

8.6 APACHE ZOOKEEPER

Kafka is built on top of ZooKeeper. Apache Zookeeper is a distributed configuration and synchronization service in Hadoop clusters. It serves as the coordination interface between the Kafka brokers and data consumers. The Kafka servers stores basic metadata in Zookeeper and shares information about topics, brokers, and consumer offsets (queue readers) and so on.

Since Zookeeper does its own layer of replication, the failure of a Kafka broker does not affect the state of the Kafka cluster. Even if Zookeeper fails, Kafka will restore the state, once the Zookeeper restarts. This gives zero downtime for Kafka. Zookeeper also manages the alternative leader broker selection, in case of a Kafka leader failure.

8.6.1 Kafka Producer Example in Java

```
//Configure
Properties config =  new Properties();
config.setProperty(ProducerConfig.BOOTSTRAP_SERVER_CONFIG, “localhost:8082”);
KafkaProducer producer = new KafkaProducer(config);
ProducerRecord record= new ProducerRecord(“topic”, “key”.getBytes(), ”value”.
getBytes());
Future<RecordMetaData>  response = producer.send(record);
```

8.7 CONCLUSION

Big data is ingested using a dedicated system. Publish-and-subscribe systems are efficient ways of delivering data from many sources to many targets, in a reliable, secure, and efficient way. Kafka is an open-source, reliable, secure, and scalable data publish-subscribe messaging system. It is a set of brokers that deal with producers as well as consumers of data. Messages are published to a set of central topics. Consumer can subscribe to topics. Kafka uses a leader-follower system of managing replicated partitions for the same set of data, to ensure full reliability and zero downtime.

Review Questions

1. What is a data ingest system? Why is it an important topic?
2. What are the two ways of delivering data from many sources to many targets?

3. What is Kafka? What are its advantages? Describe 3 use cases of Kafka.
4. What is a topic? How does it help with data ingest management?

True/False Questions

1. A dedicated ingest system is needed to accept the data because Hadoop cannot directly receive data.
2. Point-to-point messaging systems are not suitable for large volumes of data.
3. Kafka is a pub-sub messaging system.
4. Kafka is used mostly for batch processing systems.
5. A data producer is a source of incoming data.
6. Kafka can handle up to 64 data sources.
7. Kafka consumers can read the data anytime in the future.
8. Kafka organizes data in terms of categories called Topics.
9. Data for every topic is stored in a separate server.
10. A consumer can subscribe up to 64 Topics.
11. A Kafka broker is dedicated to a particular topic.
12. Kafka guarantees the order of sequence of messages will be in the order in which they were received.
13. ZooKeeper provides support services to Kafka brokers and data producers and consumers.
14. Kafka uses a leader-follower system to manage multiple partitions for a topic.
15. Kafka was developed at Facebook.

Chapter 9

Cloud Computing

Learning Objectives

- Know key concepts of cloud computing
- Examine its technical and business benefits
- Understand the concept of virtualization and hypervisors
- Analyze different cloud computing models
- Be aware of the myths and challenges of cloud computing

Introduction

Cloud computing is a cost-effective and flexible mode of delivering IT infrastructure over internet, as a service to clients on a metered basis. The cloud computing model offers clients enormous flexibility to use as much IT capacity – compute, storage, network – as needed without having to invest in a dedicated IT capacity on one's own. The IT usage can be scaled up or down in minutes. The complex IT infrastructure management skills are all owned by the cloud computing provider, and any problems can be resolved much faster. The client can simply access a smoothly running IT infrastructure over a fast internet connection. IT capacity in the cloud can also be purchased as a custom package depending upon one's needs in terms of average and peak IT requirements. The computing cloud is the ultimate cosmic computer aligned with all laws of nature.

Managing large and fast data streams is a huge challenge. It requires making critical decisions about its storage, structure, and access. This data would be stored in large clusters of hundreds or thousands of inexpensive computers. Such clusters of machines are also called server farms. The location and size of such clusters impacts

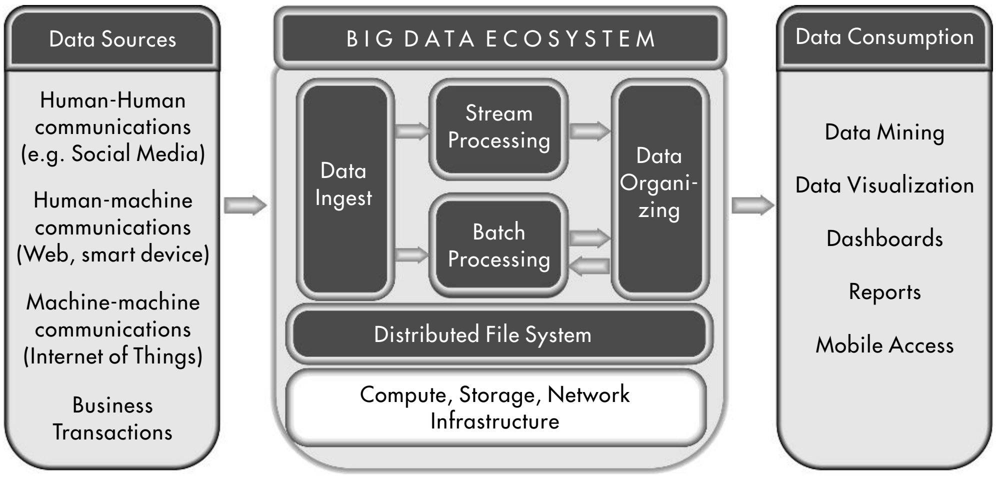


FIGURE 9.1 Big data architecture

costs. The server farms may be located in an organization's own data centers, or they may be rented from specialized third-party organizations called cloud computing service providers.

Cloud computing provides an organization's IT leadership a cost-effective and predictable solution for reliably meeting their large data management needs. There are many cloud computing vendors offering this service. Prices are dropping regularly, because per-unit IT components are getting cheaper by the day. In addition, there is growing volume of business and, also, effective competition. With cloud computing, the IT expense becomes an operating expense rather than a capital expense. The costs of IT become aligned with revenue streams and makes cash flow management easier.

Another major reason for enterprises moving to cloud computing is to experiment with new and risky projects. This flexible model makes it much easier to launch new products and services, without being exposed to the risk of a heavy loss in IT infrastructure. For example, a new Hollywood movie's site will have millions of visitors to its website for a month before and for a month after the movie's release date. After that the visits to the website will drop dramatically. The website owner would benefit enormously from using a cloud computing model where they pay for the peak web usage capacity for those few months, and much less as the usage drops down. More importantly, the flexibility ensures that their website will not crash just in case the movie becomes a super-hit and attracts unusually large number of visitors to the website.

9.1 CLOUD COMPUTING CHARACTERISTICS

Here are the major characteristics of a cloud computing model.

1. ***Flexible Capacity***: The capacity can scale up rapidly. One can expand and reduce resources according to one's specific service requirements, as and when needed. The cloud infrastructure internally does regular workload balancing among the needs of millions of clients, and this helps bring down costs for everyone.
2. ***Attractive payment model***: Cloud computing works on a pay-per-use model. i.e. one pays only for what one uses, and for how long one uses it. IT costs become an expense rather than a capital expense for the client. The resource prices may be negotiated at long-term contract rates, and can also be purchased at spot market rates.
3. ***Resiliency and Security***: The failure of any individual server and storage resources does not impact the user. The servers and storage for all clients are isolated to maximize security of data.

9.1.1 In-house Storage

Most organizations have their dedicated data centres for running their regular IT operations. An organization may decide to expand its own data centre to store large streams of data. The organization can ensure complete security and privacy of its data if it keeps all the data in-house. However, the costs and complexity of managing this data infrastructure are increasing, and it is not cost-effective for every organization to manage huge data centres. Hiring and retaining scarce advanced skills to manage such data centres would also be a challenge.

9.2 CLOUD STORAGE

It is now becoming a trend for organizations to choose to store their data in massive data centers owned by other specialized companies. Their data and processing capacity resides in some sort of a huge cloud out there, which is accessible from anywhere anytime through a simple internet connection (Figure 9.2).

Companies like Amazon, Google, Microsoft, Apple, and IBM are among the major providers of cloud storage and computing services around the world. They own and operate data centers with millions of computers in them.

Commercially, cloud service providers can consolidate the requirements of thousands or millions of customers, and supply flexible amounts of data storage and computing

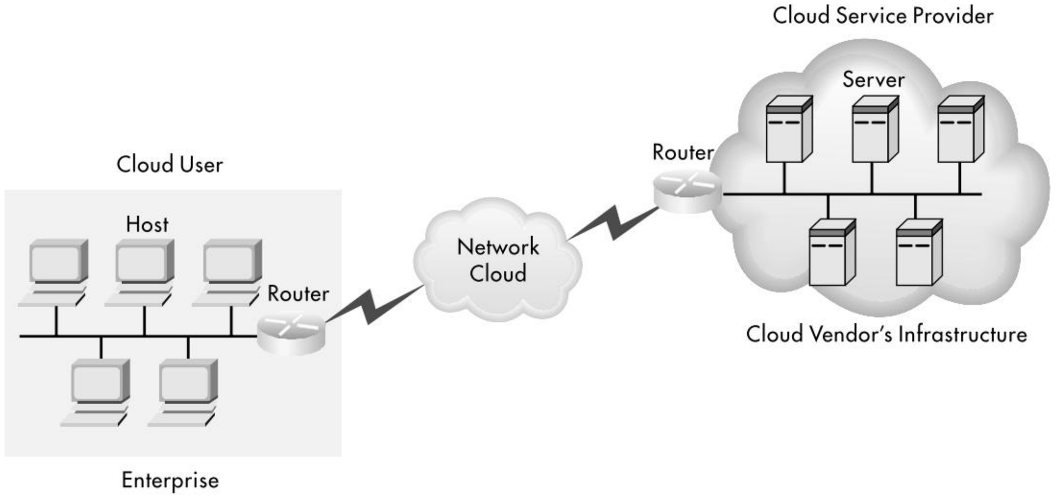


FIGURE 9.2 Cloud storage system

facility available to clients on a per-usage basis. This pay model is similar to how electric utility companies charge consumers for their usage of electricity in homes and offices. Cloud computing offers much lower costs per use, just like using the electric utility costs much less than owning and operating one's own electricity generators (Figure 9.3).

FIGURE 9.3 A cloud computing data center

A major disadvantage of cloud storage is that the data is stored away from one's physical control. Thus security of precious data is left to the hands of the cloud computing provider. While the security protocols are rapidly improving, however, there are no failsafe methods for securing data in the cloud.

There is also a risk of being locked into one provider's infrastructure. The cost-benefit tradeoffs have definitely tilted towards using cloud computing providers. At some future point in time, the cloud services providers might be heavily regulated like the electric utilities.

9.3 CLOUD COMPUTING: EVOLUTION OF VIRTUALIZED ARCHITECTURE

Cloud computing is essentially a commercial model for virtualized server infrastructure. IBM began to offer time-sharing services on its mainframe computers beginning in the 1960s. That same resource-sharing technology has been offered on networks of small machines through the virtualization process.

Virtualization assumes that logical machines can be differentiated from physical machines. A physical server could run multiple *Virtual Machines* (VMs). One virtual machine may span multiple physical servers. The virtualization software is called a hypervisor. It abstracts all machines into Virtual Machines, using easy GUI interface (Figure 9.4). A virtualization software can typically run on a heterogeneous physical

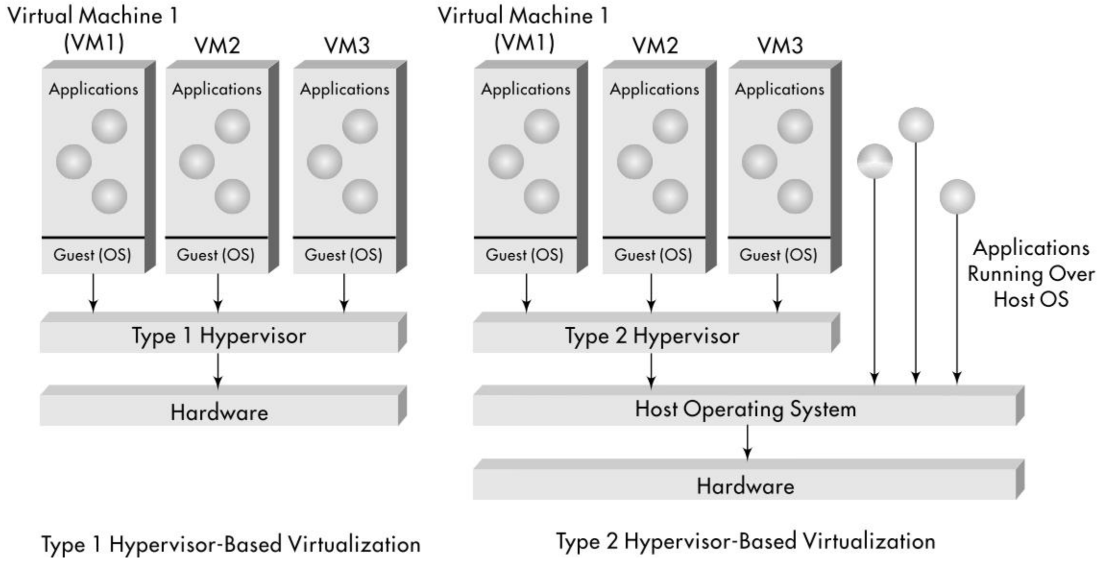


FIGURE 9.4 Virtualized architecture of cloud computing

infrastructure, and convert all IT capacity into a single unified capacity. This capacity can then be provisioned in slices and packages. The user applications are not aware that they are running in a virtualized environment; so they run as if running on a dedicated machine. The applications can also run on top of their own native operating systems.

Cloud Service Models

There are two major dimensions to conceptualize the Cloud computing models: the scope of services received; and the control over and cost of those services (Figure 9.5).

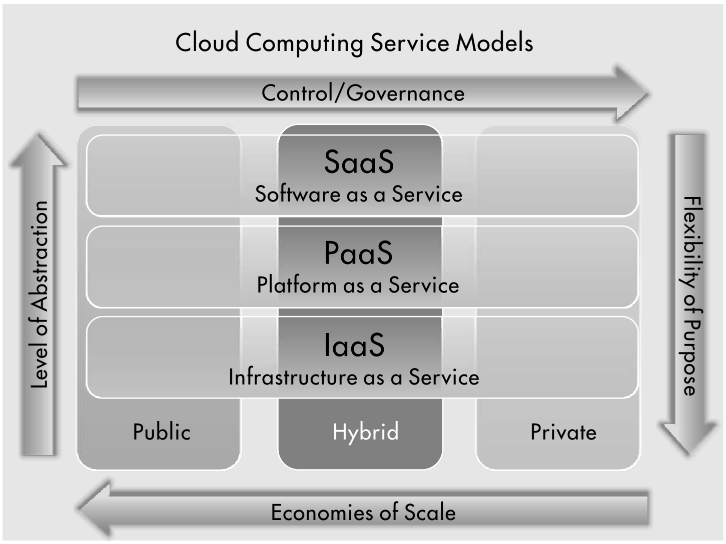


FIGURE 9.5 Cloud computing service models

1. The range of cloud computing services from a cloud computing provider, fall in three broad buckets:
   (a) **Infrastructure as a service**: This is the lowest level of services, and included only raw capacity of compute, storage, and networking. The price for this services is the lowest.
   (b) **Platform as a service**: This includes IaaS, along with other technologies and services. These are still very general tools such open source Hadoop or Spark or Cassandra implementation, along with certain monitoring tools. The costs are a little higher because of the additional management and monitoring services provided by the provider.

   (c) **Software as a service**: This includes the computing platform as well as business applications that get work done. For example, salesforce.com was one of the first CRM application sold only on a SaaS model. Google's email service is sold to organizations on a per-user-per-year basis. The costs of this service model are higher than the platform-as-a-service model.

2. The other way the cloud services differ is in terms of the ownership and control.

   (a) **Public cloud**: This will be a large shared infrastructure made available to all, in a low-cost and multi-tenancy model. The client can access it using any device. The downside is that the data also resides on the cloud, and thus could be vulnerable to theft or hacking. The costs to client are low, and variable depending upon use.

   (b) **Private cloud**: This is a cloud version of an in-house IT infrastructure. The organization will have exclusive control over the entire infrastructure. The costs would be fixed and higher.

   (c) **Hybrid cloud**: This is a mix of flexibility of capacity, and much control over some key aspects of it. One could retain complete control over critical applications, while using shared infrastructure for non-critical applications.

All levels of infrastructure and pay serve different levels of needs for client organizations. However, most of the recent growth in cloud computing is happening in the public cloud model because of the attractiveness of the dramatically lower prices.

9.4 CLOUD COMPUTING MYTHS

There are a couple of misconceptions about the costs and benefits of cloud computing.

1. **Myth:** Public Cloud computing would satisfy all the requirement: scalability, flexibility, pay per use, resilience, multitenancy, and security. Depending upon the type of service selected (SaaS, IaaS, or PaaS), the service can satisfy specific subsets of these requirements.
2. **Myth:** Cloud computing would be useful only if you are outsourcing your IT functions to an external service provider. One could use a private cloud computing model for a section of IT applications to offer on-demand, scalable, and pay-per-use deployments within your enterprise's own data center.

9.5 CLOUD COMPUTING: GETTING STARTED

Learn more about the context for getting benefits from cloud computing. Select the right model and level of cloud capacity. Set up the applications and a monitoring system for those application and the total cloud footprint. Choose a service provider, say Amazon Web Services, the leading provider of cloud computing (Figure 9.6). Use Appendix A to install Hadoop on AWS EC2 public cloud infrastructure.

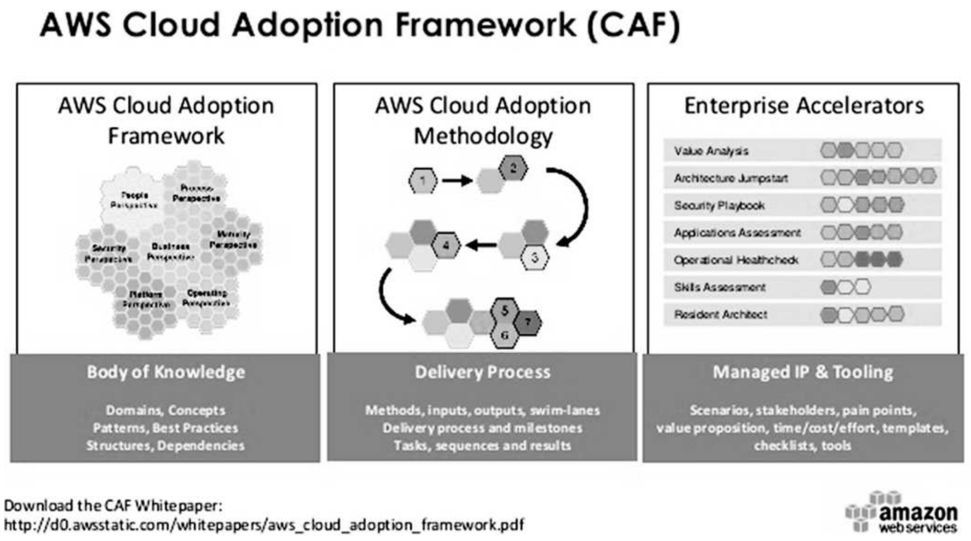


FIGURE 9.6 AWS cloud adoption framework

9.6 CONCLUSION

Cloud computing is a business model to provide shared, flexible, cost-effective IT infrastructure to get started quickly on building an application. For Big Data applications, it can be even more attractive to test the system using rented facilities, before making the determination of investing in dedicated IT infrastructure.

Review Questions

1. Describe Cloud Computing model.
2. What are the advantages of cloud computing over in-house computing.
3. Describe the technical architecture for Cloud computing.
4. Name a few major providers of cloud computing services.

True/False Questions

1. Cloud computing simplifies computing for the end-users.
2. Cloud computing is essentially a business model.
3. Cloud computing offers instant scalability in data storage.
4. Virtualization is a model of creating flexible units of computing and storage.
5. Cloud model offers lower costs of computing.
6. Data security is not an issue in cloud computing.
7. There are two kinds of cloud computing ownership models.
8. Cloud computing models can differ based on the range of services offered.
9. Google is the largest public cloud computing provider.
10. Cloud computing model can be implemented in-house also.

Section 3

This section covers the other relevant concepts and tutorials for effectively managing and utilizing Big Data.

- **Chapter 10** will bring all the tools together in a case study of developing web log analyzer, as an example of a useful Big Data application.
- **Chapter 11** will cover the overall view of Data Mining tools and techniques to extract benefit from Big Data.
- **Chapter 12** will be a primer on Big data programming languages, Hive and Pig.
- **Appendix 1** will present the systematic way to install a Hadoop cluster on a local machine and to run a sample Word count application.
- **Appendix 2** will display systematic way to install a Hadoop cluster on a cloud computing platform and to run a sample Word count application.
- **Appendix 3** will demonstrate a tutorial on installing and running Spark.

Chapter 10

Web Log Analyzer Application Case Study

Learning Objectives

- Understand the architecture of a typical big data application
- Learn the design elements of a Web log analyzer application
- Develop the sample code and outputs for such an application

Introduction

A web log analyzer is an automated software tool that helps to analyze and make decisions on several issues regarding web application server logs. An ideal web log analyzer would analyze unlimited streams of data and help keep the entire universe running smoothly and without fault. This would be done by eliminating the need for manually accessing the logs, automating the flow of information, and alerting the system administrator as needed.

10.1 CLIENT-SERVER ARCHITECTURE

Every web-based application runs on a client-server architecture. Clients are entities that access servers, and servers are entities that respond to the client with a solution. Many clients simultaneously try to access servers. The servers may be database server, network server, the application server, or any server in the n-tier architecture. For each request, a log entry is generated. The speed of access requests determined the stream of log entries. This leads to a potentially huge log over time. The log can be processed as stream of data. This log can also be stored on the servers for later analysis.

Logs can be used for monitoring, audit, and analysis purposes. It can help with error diagnostics in case a website becomes slow or it goes down. Logs can be analyzed to detect hacking activity. They can also be analyzed to summarize the popularity of webpages, and the distribution of the page requesters. It can help with access volumes, and for scaling up or down the infrastructure.

10.2 WEB LOG ANALYZER

The log analyzer received streaming logs from a server location, and analyzes multiple things using many algorithms to generate the desired results. The system is completely automated. The log is produced, and it is consumed to make real-time reports. It is easy to imagine the massive dataflow produced by the log in the server environment while it is also being analyzed simultaneously on the administrator side. `

10.2.1 Requirements

This is a log analyzer to analyze a web application hosted on a server. It is a busy application owned by a big company. It receives more than 15000 web access requests per hour. All the access requests need to be logged, and dumped to Hadoop File system periodically. The analyzer is required to ingest real-time log data, and filter out a part of data for analyzing and dumping to HDFS. It has to do streaming data flow management as well as batch processing. The analyzer needs to process the data before it is dumped into HDFS, and also after it is put into HDFS. The system administrators should be alerted in real time about possible threats, overloads, delays, potentials errors, and any other damages. The results of all the analyses must be stored in a database for later presentation in a graphical format. The results must be made available for any period, without any missing time values. The log data has to be preserved for future without losing any log data.

10.2.2 Solution Architecture

Get streaming data using Apache Flume, and send it to HDFS. Use Apache Spark for data flow management platform and processing engine. Store the results of analysis in MongoDB. This is a safe solution, because the data gets stored into Hadoop cluster and is available for future requirements, even while it is being analyzed in real time. The results of real-time processing also go into MongoDB. Figure 10.1 presents web log analyzer architecture.

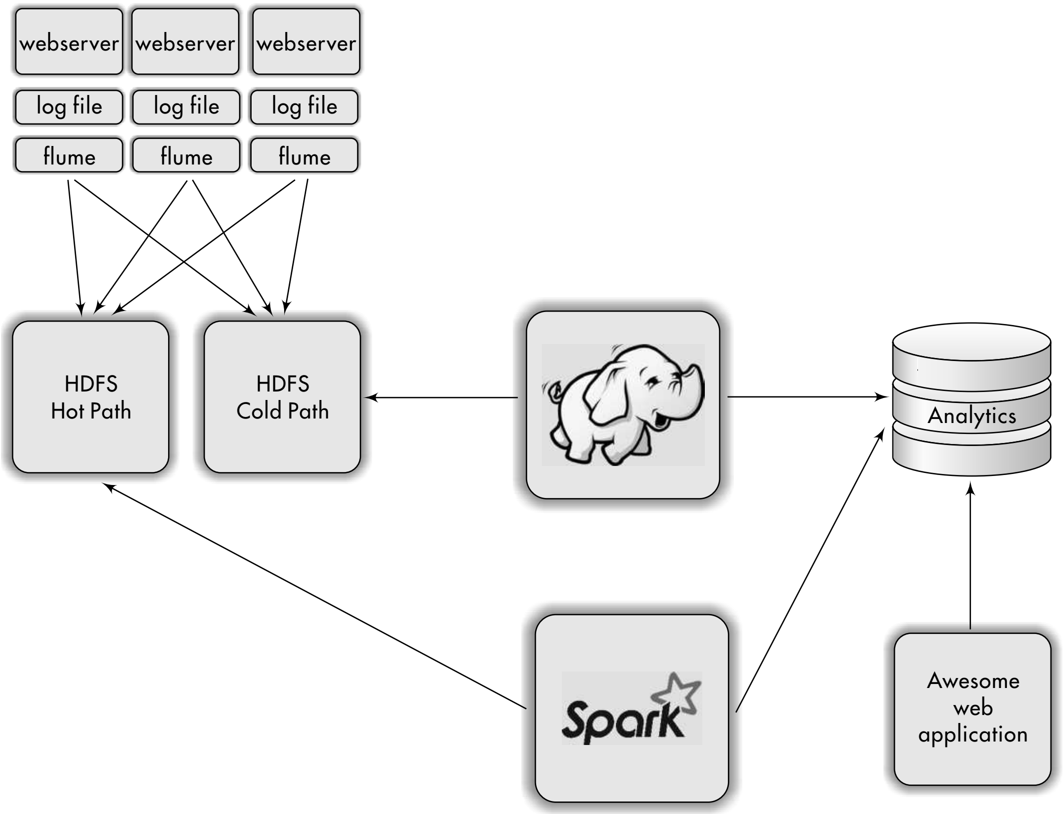


FIGURE 10.1 Web log analyzer architecture

10.2.3 Benefits of Such Solution

The advantages of such solution are:

1. *Real-time logging and analysis data generated on server*. It is streamed directly to HDFS by Flume agent without delay. Every log entry generated over every single point of time is analyzed and used for monitoring and decision making.
2. *Automatic log handling and storage*. Loading data into HDFS normally requires manually running certain Hadoop commands. This log analyzer uses a Flume agent or spark streaming to handle all data on its own, without any externally managed efforts.
3. *Easy and convenient to implement*: It is made possible by using built-in and easy-to-customize machine learning algorithms in Spark.
4. *Easy error handling, server request handling, and overall server performance optimization*. It makes server smarter by keeping track of almost every aspects of server.

10.3 TECHNOLOGY STACK

The technology stack used for this application is shown below. A brief of each component follows.

1. Apache Spark v2
2. Hadoop 2.6.0 cdh5
3. Apache Flume
4. Scala, Java
5. MongoDB
6. RestFul Webservices
7. Front UI tools
8. Linux Shell Scripts

10.3.1 Apache Spark

Spark is fast in-memory-based cluster computing technology, designed for fast and streaming computation. It is built on top of Hadoop and MapReduce system, and it extends MapReduce model to use more types of computation, which includes interactive queries and stream processing. It has lot of libraries and packages like machine learning (MLLib), graph computation (GraphX) etc. It claims to execute 10 to 100 times faster than Hadoop because of its in-memory computation model. It also supports multiple languages such as Scala, Python, Java, and R.

10.3.2 Spark Deployment

1. Standalone
2. Hadoop YARN
3. SIMR: Spark in mapReduce //Mesos

10.3.3 Components of Spark

Spark Sql: Data abstraction called schemaRDD, which provides support for structured and semi-structured data.

Spark Streaming: Ingests data in mini batch and perform RDD transformation on those mini-batches. Streaming data analytics using RDD.

MLLib (Machine learning): It is a distributed machine learning framework, which operates in-memory at high speed, and offers many ML algorithms.

GraphX: This distributed graph processing framework provides API for many graph computation algorithms.

Spark Core: This is a general execution engine for spark platform upon which all other functionality is built. It takes care of task dispatching and scheduling, and basic I/O functionalities.

Spark-shell: It is a powerful tool to analyze data interactively. It is available on scala and python. Spark's primary data abstraction is an in-memory collections of items called RDD. It can be created from Hadoop input formats like HDFS, and by transforming existing RDDs using filters and maps into new RDDs.

Scripting and Programming model using Spark Context: One can use an IDE to develop and test the analytics code. One can then create a jar to run the analytics using Hadoop architecture. The jar can also be submitted using spark-submit utility to the Spark engine. For example:

```
spark-submit --class apache.accesslogs.ServerLogAnalyzer --master
* local ScalaSpark/Scala1/target/scala-2.10/Scala1-assembly-1.0.jar > output.txt
```

10.4 HDFS

HDFS is a distributed file system, that is at the core of Hadoop system.

- Deployed on low cost commodity hardware
- Fault tolerant
- Supports Batch Processing
- Designed for large dataset or large files
- Maintains coherence through write once read many times
- Moving computation to the location of the data.

10.5 MONGODB

It is a free and open-source cross-platform document-oriented database. It came into existence as a NoSQL database and is classified as a NoSQL database program. MongoDB supports field, range queries, regular expression searches.

10.6 APACHE FLUME

Flume is an open source tool for handling streaming logs or data. It is a distributed and reliable system for efficiently collecting, aggregating and moving large amount of data from many different sources to a centralized data store. It is a popular tool to assist with data flow and storage to HDFS. Flume is not restricted to log data. The data sources are customizable so it might be any source like event data, traffic data, social media data, or any other data source. The major Components of Flume are:

- Event
- Agent
- Data Generators
- Centralized Stores

10.7 OVERALL APPLICATION LOGIC

The system reads access logs and presents the results in tabular and graphical form to end users. This system provides the following major functions:

1. Calculate content size
2. Count Response code
3. Analyze requesting IP-address
4. Manage End points

10.8 TECHNICAL PLAN FOR THE APPLICATION

Technically, the project follows the following structure:

1. Flume takes streaming log from running application server and stores in HDFS. Flume uses compression to store huge log files to speed up the data transfer and for storage efficiency.
2. Apache Spark uses HDFS as input source and analyzes data using MLLib. Apache Spark stores analyzed data in MongoDB.
3. RESTful java service presents JSON objects fetching from MongoDB and sending to Front end. Graphical tools are used to present data.

10.9 SCALA SPARK CODE FOR LOG ANALYSIS

Note: This application is written in Scala language. Below is the operative part of the code. Visit github link below for the complete Scala code for this application.

```
   //calculates size of log, and provides min, max and average size
   // caching is done for repeatedly used factors
   def calcContentSize(log: RDD[AccessLogs]) = {
    val size = log.map(log => log.contentSize).cache()
    val average = size.reduce(_ + _) / size.count()
    println("ContentSize:: Average :: " + average + " " +
    " || Maximum :: " + size.max() + " || Minimum ::" + size.min() )
    }

   //Send all the response code with its frequency of occurrence as Output
   def responseCodeCount(log: RDD[AccessLogs]) = {
    val responseCount = log.map(log => (log.responseCode, 1))
    .reduceByKey(_ + _)
    .take(1000)
    println(s"""ResponseCodes Count : ${responseCount.mkString("[", ",", "]")} """)

    }

   //filters ipaddresses that have more then 10 requests in server log
    def ipAddressFilter(log: RDD[AccessLogs]) = {
    val result = log.map(log => (log.ipAddr, 1))
    .reduceByKey(_ + _)
    .filter(count => count._2 > 1)
    // .map(_._1).take(10)
    .collect()
println("IP Addresses Count :: ${result.mkString("[", ",", "]")}" )
} }
```

10.10 SAMPLE LOG DATA

10.10.1 Sample Input Data

Input Fields (Selected Fields)

Certain fields have been omitted to make the code clear. The response code has been colored in red as it is the basis of the major reports.

1. `ipAddress: String,`
2. `dateTime: String,`
3. `method: String,`
4. `endPoint: String,`
5. `protocol: String,`
6. `responseCode: Long,`
7. `contentSize: Long`

Sample Input Rows of Data

```
64.242.88.10 [07/Mar/2014:16:05:49 -0800] “GET
/twiki/bin/edit/Main/Double_bounce_sender?topicparent=Main.ConfigurationVariables
HTTP/1.1” 401 12846

64.242.88.10 [07/Mar/2014:16:06:51 -0800] “GET
/twiki/bin/rdiff/TWiki/NewUserTemplate?rev1=1.3&rev2=1.2 HTTP/1.1” 200 4523

64.242.88.10 [07/Mar/2014:16:10:02 -0800] “GET
/mailman/listinfo/hsdivision HTTP/1.1” 200 6291

64.242.88.10 [07/Mar/2014:16:11:58 -0800] “GET
/twiki/bin/view/TWiki/WikiSyntax HTTP/1.1” 200 7352

64.242.88.10 [07/Mar/2014:16:20:55 -0800] “GET
/twiki/bin/view/Main/DCCAndPostFix HTTP/1.1” 200 5253

64.242.88.10 [07/Mar/2014:16:23:12 -0800] “GET
/twiki/bin/oops/TWiki/AppendixFileSystem?template=oopsmore&param1=1.12&param2=1.12
HTTP/1.1” 200 11382
```

10.11 SAMPLE OUTPUT OF WEB LOG ANALYSIS

```
ContentSize:: Average:: 10101 || Maximum :: 138789 || Minimum ::0
ResponseCodes Count: [(401,113), (200,591), (302,1)]
IP Addresses Count:: [(127.0.0.1, 31), (207.195.59.160, 15), (67.131.107.5, 3),
(203.147.138.233, 13), (64.242.88.10, 452), (10.0.0.153, 188)]
```

```
EndPoints :: [(/wap/Project/login.php,15),(/cgi-bin/mailgraph.cgi/mailgraph_2.png,12),
(/cgi-bin/mailgraph.cgi/mailgraph_0.png,12),(/wap/Project/loginsubmit.php,12),(/cgi-
bin/mailgraph.cgi/mailgraph_2_err.png,12),(/cgi-bin/mailgraph.cgi/mailgraph_1.png,12),
(/cgi-bin/mailgraph.cgi/mailgraph_0_err.png,12),(/cgi-bin/mailgraph.cgi/mailgraph_1_err.
png,12),(/cgi-bin/mailgraph.cgi/mailgraph_3_err.png,12),(/cgi-bin/mailgraph.cgi/mail-
graph_3.png,12)]
```

Intermediate data is stored in Hadoop File System in CSV format

To see detailed code, visit: https://github.com/databricks/reference-apps/blob/master/logs_analyzer/chapter1/scala/src/main/scala/com/databricks/apps/logs/chapter1/LogAnalyzer.scala

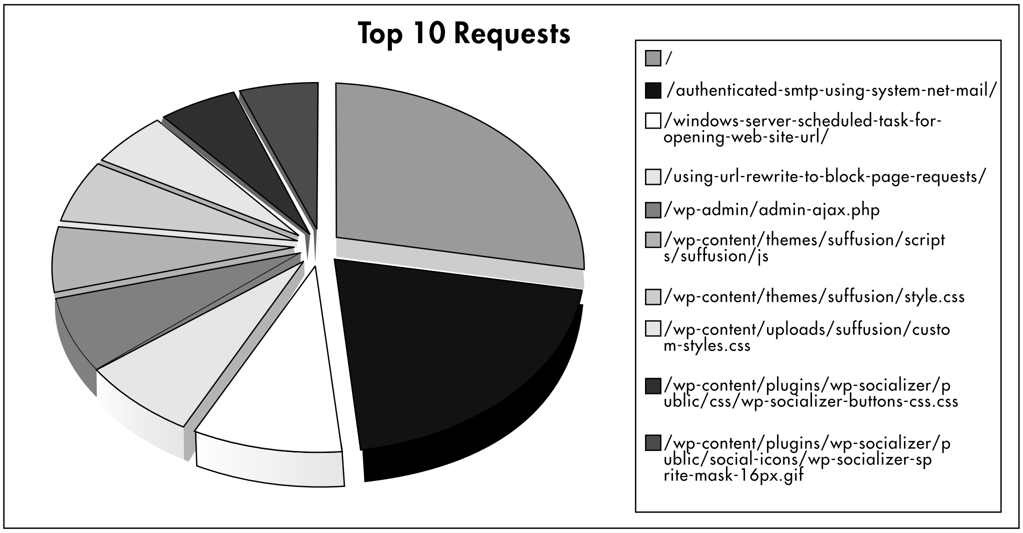


FIGURE 10.2 A pie chart showing top ten requests of web log analyzer

This web log analyzer (Figure 10.2) can be enhanced in many ways. For example, it can analyze history of logs from previous years and discover web access trends. This application can also be made to discard data older than 5 years into permanent and backup storage.

10.12 CONCLUSION

This application shows how to create a simple but useful application using Apache Spark for getting useful value out of Big Data. It also showed how the analyzed data can be visualized for easy decision making. One can experiment with other tools to enhance this application.

Review Questions

1. Describe the advantages of a web log analyzer.
2. Describe the major challenges in developing this application.
3. Identify 3–4 major lessons learned from the code.
4. How can this code be improved?

True/False Questions

1. A web application follows a client-server architecture.
2. A web log analyzer is a very specialized application that differs for every client.
3. The web log application architecture definitely needs Spark for data ingest.
4. HDFS is used in this application for processing data.
5. Many kinds of reports can be produced from web log analysis.

Chapter 11

Data Mining Primer

Learning Objectives

- Understand the concept and process of data mining
- Know the process of gathering and selecting data for mining
- Represent many outputs of data mining
- Evaluate the results of data mining
- Learn many techniques for data mining

INTRODUCTION

Data mining is the art and science of discovering knowledge, insights, and patterns in data. It is the act of extracting useful patterns from an organized collection of data. Patterns must be valid, novel, potentially useful, and understandable. The implicit assumption is that data about the past can reveal patterns of activity that can be projected into the future.

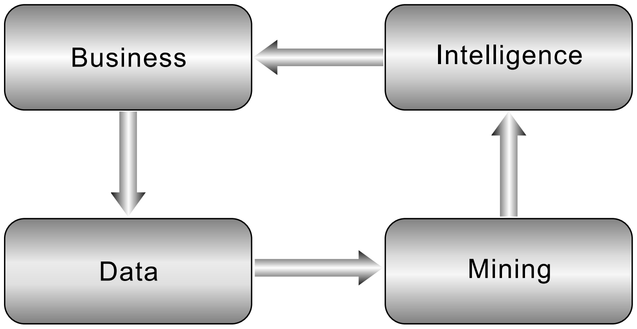


FIGURE 11.1 Data analytics architecture

Data mining is a multidisciplinary field that borrows techniques from a variety of fields. It utilizes the knowledge of data quality and data organizing from the databases area. It draws modeling and analytical techniques from statistics and computer science (artificial intelligence) areas. It also draws the knowledge of decision-making from the field of business management.

The field of data mining emerged in the context of pattern recognition in defense, such as identifying a friend-or-foe on a battlefield. Like many other defense-inspired technologies, it has evolved to help gain a competitive advantage in business.

For example, “customers who buy cheese and milk also buy bread 90 per cent of the time” would be a useful pattern for a grocery store, which can then stock the products appropriately. Similarly, “people with blood pressure greater than 160 and an age greater than 65 were at a high risk of dying from a heart stroke” is of great diagnostic value for doctors, who can then focus on treating such patients with urgent care and great sensitivity.

Past data can be of predictive value in many complex situations, especially where the pattern may not be so easily visible without the modeling technique. Here is a dramatic case of a data-driven decision-making system that beats the best of human experts. Using past data, a decision tree model was developed to predict votes for Justice Sandra Day O’Connor, who had a swing vote in a 5–4 divided US Supreme Court. All her previous decisions were coded on a few variables. What emerged from data mining was a simple four-step decision tree that was able to accurately predict her votes 71 per cent of the time. In contrast, the legal analysts could at best predict correctly 59 per cent of the time.

11.1 GATHERING AND SELECTING DATA

To learn from data, quality data needs to be effectively gathered, cleaned and organized, and then efficiently mined. One requires the skills and technologies for consolidation and integration of data elements from many sources.

Gathering and curating data takes time and effort, particularly when it is unstructured or semi-structured. Unstructured data can come in many forms like databases, blogs, images, videos, audio, and chats. There are streams of unstructured social media data from blogs, chats, and tweets. There are streams of machine-generated data from connected machines, RFID tags, the internet of things, and so on. Eventually the data should be *rectangularized,* that is, put in rectangular data shapes with clear columns and rows, before submitting it to data mining.

Knowledge of the business domain helps select the right streams of data for pursuing new insights. Only the data that suits the nature of the problem being solved should be gathered. The data elements should be relevant, and suitably address the problem being solved. They could directly impact the problem, or they could be a suitable proxy for the effect being measured. Select data could also be gathered from the data warehouse. Every industry and function will have its own requirements and constraints. The health care industry will provide a different type of data with different data names. The HR function would provide different kinds of data. There would be different issues of quality and privacy for these data.

11.2 DATA CLEANSING AND PREPARATION

The quality of data is critical to the success and value of the data mining project. Otherwise, the situation will be of the kind of garbage in and garbage out (GIGO). The quality of incoming data varies by the source and nature of data. Data from internal operations is likely to be of higher quality, as it will be accurate and consistent. Data from social media and other public sources is less under the control of business, and is less likely to be reliable.

Data almost certainly needs to be cleansed and transformed before it can be used for data mining. There are many ways in what data may need to be cleansed – filling missing values, reigning in the effects of outliers, transforming fields, binning continuous variables, and much more – before it can be ready for analysis. Data cleansing and preparation is a labor-intensive or semi-automated activity that can take up to 60–80% of the time needed for a data mining project.

11.3 OUTPUTS OF DATA MINING

Data mining techniques can serve different types of objectives. The outputs of data mining will reflect the objective being served. There are many ways of representing the outputs of data mining.

One popular form of data mining output is a decision tree. It is a hierarchically branched structure that helps visually follow the steps to make a model-based decision. The tree may have certain attributes, such as probabilities assigned to each branch. A related format is a set of business rules, which are if-then statements that show causality. A decision tree can be mapped to business rules. If the objective function is prediction, then a decision tree or business rules are the most appropriate mode of representing the output.

The output can be in the form of a regression equation or mathematical function that represents the best fitting curve to represent the data. This equation may include linear and nonlinear terms. Regression equations are a good way of representing the output of classification exercises. These are also a good representation of forecasting formulae.

Population "centroid" is a statistical measure for describing central tendencies of a collection of data points. These might be defined in a multidimensional space. For example, a centroid could be "middle-aged, highly educated, high-net worth professionals, married with two children, living in the coastal areas". Or a population of "20-something, ivy-league-educated, tech entrepreneurs based in Silicon Valley". Or it could be a collection of "vehicles more than 20 years old, giving low mileage per gallon, which failed environmental inspection". These are typical representations of the output of a cluster analysis exercise.

Business rules are an appropriate representation of the output of a market basket analysis exercise. These rules are if-then statements with some probability parameters associated with each rule. For example, those that buy milk and bread will also buy butter (with 80 per cent probability).

11.4 EVALUATING DATA MINING RESULTS

There are two primary kinds of data mining processes: supervised learning and unsupervised learning. In supervised learning, a decision model can be created using past data, and the model can then be used to predict the correct answer for future data instances. Classification is the main category of supervised learning activity. There are many techniques for classification, decision trees being the most popular one. Each of these techniques can be implemented with many algorithms. A common metric for all of classification techniques is predictive accuracy.

11.4.1 Predictive Accuracy = (Correct Predictions)/Total Predictions

Suppose a data mining project has been initiated to develop a predictive model for cancer patients using a decision tree. Using a relevant set of variables and data instances, a decision tree model has been created. The model is then used to predict other data instances. When a true positive data point is positive, that is a correct prediction, called a true positive (TP). Similarly, when a true negative data point is classified as negative, that is a true negative (TN). On the other hand, when a true-positive data point is classified by the model as negative, that is an incorrect prediction, called a false negative (FN). Similarly, when a true-negative data point is classified as positive, that is classified as a false positive (FP). This is represented using the confusion matrix (Figure 11.2).

Confusion Matrix		True Class	
		Positive	Negative
Predicted Class	Positive	True - Positive (TP)	False - Positive (FP)
	Negative	False - Negative (FN)	True - Negative (TN)

FIGURE 11.2 Confusion matrix

Thus, the predictive accuracy can be specified by the following formula.

Predictive Accuracy = (TP +TN) / (TP + TN + FP + FN).

All classification techniques have a predictive accuracy associated with a predictive model. The highest value can be 100%. In practice, predictive models with more than 70% accuracy can be considered usable in business domains, depending upon the nature of the business.

There are no good objective measures to judge the accuracy of unsupervised learning techniques such as Cluster Analysis. There is no single right answer for the results of these techniques. For example, the value of the segmentation model depends upon the value the decision-maker sees in those results.

11.5 DATA MINING TECHNIQUES

Data may be mined to help make more efficient decisions in the future. Or it may be used to explore the data to find interesting associative patterns. The right technique depends upon the kind of problem being solved (Figure 11.3).

Data Mining Techniques		
Supervised Learning (Predictive ability based on past data)		
	Classification	Decision Trees
	Machine Learning	Neural Networks
	Classification Statistics	Regression
Unsupervised Learning (Exploratory analysis to discover patterns)	Clustering Analysis	
	Association Rules	

FIGURE 11.3 Important data mining techniques

The most important class of problems solved using data mining are classification problems. Classification techniques are called supervised learning as there is a way to supervise whether the model is providing the right or wrong answers. These are problems where data from past decisions is mined to extract the few rules and patterns that would improve the accuracy of the decision making process in the future. The data of past decisions is organized and mined for decision rules or equations, that are then codified to produce more accurate decisions.

Decision trees are the most popular data mining technique, for many reasons.

1. Decision trees are easy to understand and easy to use, by analysts as well as executives. They also show a high predictive accuracy.
2. Decision trees select the most relevant variables automatically out of all the available variables for decision making.
3. Decision trees are tolerant of data quality issues and do not require much data preparation from the users.
4. Even non-linear relationships can be handled well by decision trees.

There are many algorithms to implement decision trees. Some of the popular ones are C5, CART and CHAID.

Regression is a most popular statistical data mining technique. The goal of regression is to derive a smooth well-defined curve to best the data. Regression analysis techniques, for example, can be used to model and predict the energy consumption as a function of daily temperature. Simply plotting the data may show a non-linear curve. Applying a non-linear regression equation will fit the data very well with high accuracy. Once such a regression model has been developed, the energy consumption on any future day can be predicted using this equation. The accuracy of the regression model depends entirely upon the dataset used and not at all on the algorithm or tools used.

***Artificial Neural Networks* (*ANN*)** is a sophisticated data mining technique from the Artificial Intelligence stream in Computer Science. It mimics the behavior of human neural structure: Neurons receive stimuli, process them, and communicate their results to other neurons successively, and eventually a neuron outputs a decision. A decision task may be processed by just one neuron and the result may be communicated soon. Alternatively, there could be many layers of neurons involved in a decision task, depending upon the complexity of the domain. The neural network can be trained by making a decision repeatedly with many data points. It will continue to learn by adjusting its internal computation and communication parameters based on feedback received on its previous decisions. The intermediate values passed within the layers of neurons may not make any intuitive sense to an observer. Thus, the neural networks are considered a black-box system.

Cluster Analysis is an exploratory learning technique that helps in identifying a set of similar groups in the data. It is a technique used for automatic identification of natural groupings of things. Data instances that are similar to (or near) each other are categorized into one cluster, while data instances that are very different (or far away) from each other are categorized into separate clusters. There can be any number of clusters that could be produced by the data. The K-means technique is a popular technique and allows the user guidance in selecting the right number (K) of clusters from the data. Clustering is also known as the segmentation technique. It helps divide and conquer large data sets. The technique shows the clusters of things from past data. The output is the centroids for each cluster and the allocation of data points to their cluster. The centroid definition is used to assign new data instances can be assigned to their cluster homes. Clustering is also a part of the artificial intelligence family of techniques.

Association rules are a popular data mining method in business, especially where selling is involved. Also known as market basket analysis, it helps in answering questions about cross-selling opportunities. This is the heart of the personalization engine used by ecommerce sites like Amazon.com and streaming movie sites like Netflix.com. The technique helps find interesting relationships (affinities) between variables (items or events). These are represented as rules of the form $X \rightarrow Y$, where X and Y are sets of data items. A form of unsupervised learning, it has no dependent variable; and there are no right or wrong answers. There are just stronger and weaker affinities.

Thus, each rule has a confidence level assigned to it. A part of the machine learning family, this technique achieved legendary status when a fascinating relationship was found in the sales of diapers and beers.

11.6 MINING BIG DATA

As data grows larger and larger, there are a few ways in which analyzing Big data is different.

11.6.1 From Causation to Correlation

There is more data available than there are theories and tools available to explain it. Historically, theories of human behaviour, and theories of universe in general, have been intuited and tested using limited and sampled data, with some statistical confidence level. Now that data is available in extremely large quantities about many people and many factors, there may be too much noise in the data to articulate and test clean theories. In that case, it may suffice to value co-occurrences or correlation of events as significant without necessarily establishing strong causation.

11.6.2 From Sampling to the Whole

Pooling all the data together into a single big data system can help discover events, that help bring about a fuller picture of the situation, and highlight threats or opportunities that an organization faces. Working from the full dataset can enable discovering remote but extremely valuable insights. For example, an analysis of the purchasing habits of millions of customers and their billions of transactions at their thousands of stores can give an organization a vast, detailed and dynamic view of sales patterns in their company, which may not be available from the analysis of small samples of data by each store or region.

11.6.3 From Dataset to Data Stream

A flowing stream has a perishable and unlimited connotation to it, while a dataset has a finitude and permanence about it. With any given infrastructure, one can only consume so much data at a time. Data streams are many, large and fast. Thus one has to choose which of the many streams of data does one want to engage with. It is equivalent to deciding which stream to fish in. The metrics used for analysis of streams tend to be relatively simple and relate to time dimension. Most of the metrics are statistical measures such as counts and means. For example, a company might want to monitor customer sentiment about its products. So they could create a social media listening platform that would read all tweets and blogposts about them in real-time. This platform would (a) keep a count of positive and negative sentiment messages every minute, and (b) flag any messages that merit attention such as sending an online advertisement or purchase offer to that customer.

11.7 DATA MINING BEST PRACTICES

Effective and successful use of data mining activity requires both business and technology skills. The business aspects help understand the domain and the key questions. It also helps one imagine possible relationships in the data, and create hypotheses to test it. The IT aspects help fetch the data from many sources, clean up the data, assemble it to meet the needs of the business problem, and then run the data mining techniques on the platform.

An important element is to go after the problem iteratively. It is better to divide and conquer the problem with smaller amounts of data, and get closer to the heart of the solution in an iterative sequence of steps. There are several best practices learned from the use of data mining techniques over a long period of time. The Data Mining industry has proposed a Cross-Industry Standard Process for Data Mining (CRISP-DM). It has six essential steps (Figure 11.4):

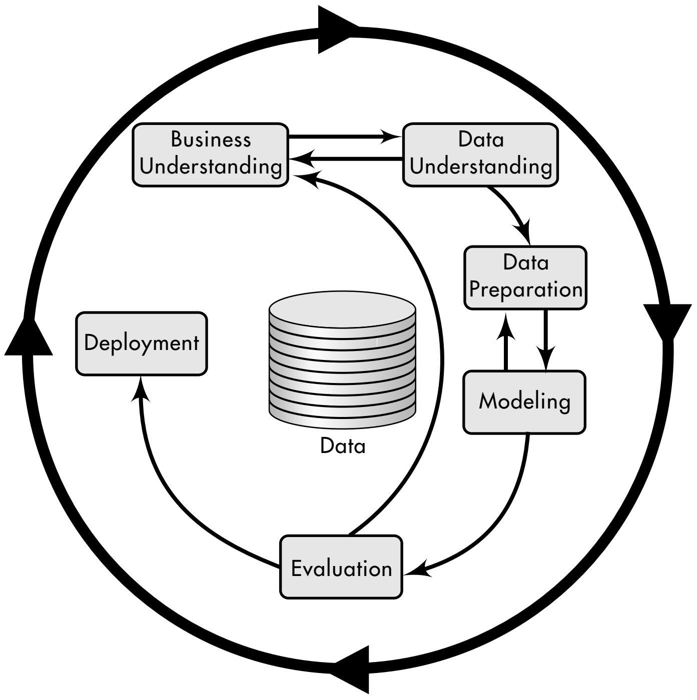


FIGURE 11.4 CRISP-DM data mining cycle

1. ***Business Understanding:*** The first and most important step in data mining is asking the right business questions. A question is a good one if answering it would lead to large payoffs for the organization, financially and otherwise. In other words, selecting a data mining project is like any other project, in that it should show strong payoffs if the project is successful. There should be strong executive support for the data mining project, which means that the project aligns well with the business strategy. A related important step is to be creative and open in proposing imaginative hypotheses for the solution. Thinking outside the box is important, both in terms of a proposed model as well in the data sets available and required.
2. ***Data Understanding:*** A related important step is to understand the data available for mining. One needs to be imaginative in scouring for many elements of data through many sources in helping address the hypotheses to solve a problem. Without relevant data, the hypotheses cannot be tested.
3. ***Data Preparation:*** The data should be relevant, clean and of high quality. It's important to assemble a team that has a mix of technical and business skills, who understand the domain and the data. Data cleaning can take 60–70% of the time in a data mining project. It may be desirable to continue to experiment and add new data elements from external sources of data that could help improve predictive accuracy.
4. ***Modeling:*** This is the actual task of running many algorithms using the available data to discover if the hypotheses are supported. Patience is required in continuously engaging with the data until the data yields some good insights. A host of modeling tools and algorithms should be used. A tool could be tried with different options, such as running different decision tree algorithms.

Model Evaluation: One should not accept what the data says at first. It is better to triangulate the analysis by applying multiple data mining techniques, and conducting many what-if scenarios, to build confidence in the solution. One should evaluate and improve the model's predictive accuracy with more test data. When the accuracy has reached some satisfactory level, then the model should be deployed.

Dissemination and rollout: It is important that the data mining solution is presented to the key stakeholders, and is deployed in the organization. Otherwise the project will be a waste of time and will be a setback for establishing and supporting a data-based decision-process culture in the organization. The model should be eventually embedded in the organization's business processes.

11.8 CONCLUSION

Data Mining is like diving into the rough material to discover a valuable finished nugget. While the technique is important, domain knowledge is also important to provide imaginative solutions that can then be tested with data mining. The business objective should be well understood and should always be kept in mind to ensure that the results are beneficial to the sponsor of the exercise.

Review Questions

1. What is data mining? What are supervised and unsupervised learning techniques?
2. Describe the key steps in the data mining process. Why is it important to follow these processes?
3. What is a confusion matrix?
4. Why is data preparation so important and time consuming?
5. What are some of the most popular data mining techniques?
6. How is mining Big data different from traditional data mining?

True/False Questions

1. Data mining is a multidisciplinary field that borrows techniques from a variety of fields.
2. Past data can be of predictive value in many complex situations.
3. Knowledge of the business domain is irrelevant to selecting the right streams of data.
4. Garbage in and garbage out (GIGO) states that the quality of data is critical to the success and value of the data mining project.
5. Data preparation is a labour-intensive activity that can take up to 60–80 per cent of the time needed for a data mining project.
6. There are two primary kinds of data mining processes: supervised learning and unsupervised learning.
7. A decision tree is a hierarchically branched structure that helps visually follow the steps to make a model-based decision.
8. A population "centroid" is a statistical measure for describing central tendencies of a collection of data points.

9. A common metric for all of classification techniques is predictive accuracy which is defined at (Correct Predictions) / Total Predictions
10. Clustering Analysis is an unsupervised form of learning
11. *Artificial Neural Networks* (ANN) is a data mining technique that mimics the behavior of human neural structure.
12. Association rules are the heart of the personalization engine used by ecommerce sites like Amazon.com.
13. Pooling all the data together can better highlight threats or opportunities that an organization faces.
14. CRISP-DM is called so, because the process is very crisp and clear.
15. Mining data is like searching for diamond in a mine.

Chapter 12

Big Data Programming Primer

Learning Objectives

- Understand the languages used to process Big Data
- Understand the differences between Pig and Hive
- Understand the design and structure of Hive
- Learn Hive commands to access and analyze Hadoop data
- Understand the design and structure of Pig
- Learn Pig commands to access and analyze Hadoop data

INTRODUCTION

The Hadoop ecosystem includes many language tools such as Hive and Pig to easily access and speedily managing large diverse distributed data systems to produce reports and insights. Hive and Pig languages offer easy high-level SQL-like commands, supplemented by complex computation using user-defined functions, and managing partitioned data in Hadoop Distributed File system. These languages ultimately generate code for MapReduce, that does parallel processing to compute the data with great speed. Using these languages greatly helps increase programmer productivity.

This chapter will offer a short primer on both Hive and Pig. It will include detailed listing of functions and features of both languages.

12.1 COMPARING HIVE AND PIG

Hive uses Hive Query Language (HiveQL or HQL), to offer a SQL-like language platform to develop scripts for MapReduce operations for structured, semi-structured, and unstructured data. Hive is convenient since it does not necessarily require knowledge of Java, and can thus be used for ad-hoc analyses. Hive works well with partitioned time series data. Multiple analysts can simultaneously use the database structure to do their work. Hive can use partitions to speed up queries.

Apache Pig includes Pig Latin, a SQL-like high level, yet procedural, language platform that is more suited for developing a script for control of data flows in MapReduce operations to process structured and semi-structured data. Pig is a better choice for doing ETL (Extract-Transform-Load) work, or where one need more control over data flows such as when the data does not cleanly fit into rows and columns. Pig also allows for partitioned data.

Both Apache Pig and Hive create MapReduce jobs at the backend (Table12.1). Hive operates on HDFS in a similar way Apache Pig does. The following table lists a few significant points to compare Hive and Pig.

Table 12.1

A comparative analysis of Hive and Pig

Tool	Key Component	Capabilities	Ease of use	Learning Curve	Data Handling	Execution platform
Apache Hive	HiveQL	SQL-like capabilities for data access & reporting	Transferrable skills from SQL	Easy	Mostly structured data	MapReduce
Apache Pig	Pig Latin	Procedural Java-like control, useful for ETL operations	Java-like detailed programming	Takes more effort	Structured & unstructured data	MapReduce

12.2 APACHE HIVE

Hive is a tool to process structured data in HDFS. It resides on top of HDFS to help summarize and analyze data. Hive was designed and developed at Facebook. It was turned over to the open source Apache Software Foundation. It is highly scalable, and is used widely. It is also available on most cloud infrastructure such as Amazon EC2.

Hive is not a relational database. It is not a tool for transaction processing. Like with almost all Hadoop-based tools, it is not a tool for data updates in place. Hive is useful mostly for Analytical Processing. Hive works through a schema that maps HDFS files into a database structure. This data schema can then be used to access data through SQL-like easy commands.

Visit apache.org to install Hive. Ensure that Linux and Java and Hadoop have been installed.

12.2.1 Architecture of Hive

The component diagram presented in Figure 12.1 depicts the architecture of Hive:

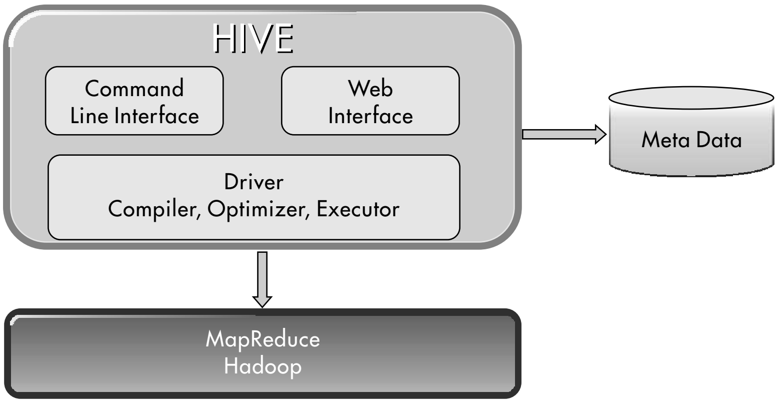


FIGURE 12.1 Apache Hive architecture

Hive supports Web UI, command line, and other user interfaces. Hive stores the schema of tables, the databases, columns in a table, their data types, and HDFS mapping in Metadata server. Writing HiveQL is similar to using SQL for querying on schema info on the Metastore. Hive is a replacement for MapReduce programming. Instead of writing MapReduce program in Java, one can write an HQL query. The Hive Execution engine processes the query and generates results just as MapReduce would. The results are sent to the user, or written back into HDFS as needed.

12.2.2 Working of Hive

Hive interpreter chains pre-built binaries together to create and run a MapReduce job to perform the query. Hive table structures are internal to it, but its data will be stored in HDFS. Hive assumes that no other system is using these internal tables. Thus dropping a table would not affect others. External tables can also be used, and can be defined as such by specifying an HDFS directory path.

Figure 12.2 depicts the workflow between Hive and Hadoop.

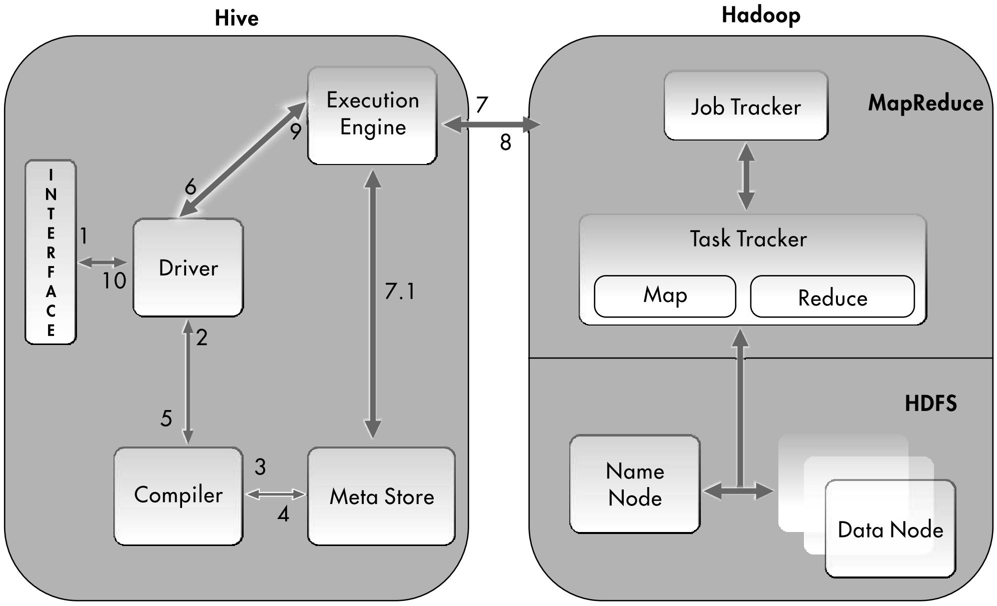


FIGURE 12.2 Workflow between Hive and Hadoop

Here is how Hive interacts with Hadoop framework. The first step is the writing of the Hive query using one of the user interfaces. The Driver parses the query statements to check the syntax. The compiler then requests the Metadata store for relevant metadata for interpreting the query. The compiler then checks requirement and resends the execution plan to the driver. The execution plan is a MapReduce job. The driver sends the execution plan to the execution engine. The execution engine sends the job to JobTracker, which runs on the Namenode. The parts of the job are assigned to TaskTrackers, which run on the Data nodes. Once the MapReduce job is completed, the results are returned to the Hive User Interface.

Not unlike SQL, there are two essential aspects of using Hive.

- The first step is to create a database, and to create tables in the database. This also includes defining the structure of partitions to store the data. This is the Data Definition part.
- The second step is to access and manipulate the data to generate reports and queries. This includes generating views and indexes. This is the Data Manipulation part.

12.2.3 Hive Data Definition

Hive is a database technology that can define databases and tables to analyze structured data. The theme for structured data analysis is to store the data in a tabular manner, and pass queries to analyze it. This section explains how to create a Hive database and tables.

Hive permits data of all types supported by SQL. These include Numeric, Floating, String, Date, DateStamp. NULL values. In addition, Hive permits many complex data types such as a Union. A Union is a collection of heterogeneous data types such as Array types, Maps, Struct, and others.

CREATE DATABASE is a statement used to create a database in Hive. A database in Hive is a namespace, and a collection of tables. The syntax is:

```
CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>
```

Drop Database is the opposite of Create statement, in that it drops all the tables and deletes the database. Its syntax is as follows:

```
DROP DATABASE StatementDROP (DATABASE|SCHEMA) [IF EXISTS] database_name
[RESTRICT|CASCADE];
```

The conventions of creating a table in HIVE are similar to those for creating a table in SQL. CREATE TABLE is a statement used to create a table in Hive. The syntax is as follows:

```
CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db_name.] table_name

[ROW FORMAT row_format]

[STORED AS file_format]
```

The following query creates a table named **customer** with a few fields.

```
hive> CREATE TABLE IF NOT EXISTS customer (eid int, name String,

> salary String, destination String)

> COMMENT ‘customer details’

> ROW FORMAT DELIMITED

> FIELDS TERMINATED BY ‘\t’

> LINES TERMINATED BY ‘\n’

> STORED AS TEXTFILE;
```

Data is inserted in SQL tables using the Insert statement. However, in Hive, data is inserted more often using the LOAD DATA statement, especially when loading bulk records.

There are two ways to load data: one is from local file system and second is from Hadoop file system. The syntax for load data is as follows:

```
LOAD DATA [LOCAL] INPATH ‘filepath’ [OVERWRITE] INTO TABLE tablename
[PARTITION (partcol1=val1, partcol2=val2 ...)]
```

```
LOCAL is identifier to specify the local path. It is optional.
OVERWRITE is optional to overwrite the data in the table.
PARTITION is optional.
The following query loads the given text into the table.
```

```
hive> LOAD DATA LOCAL INPATH ‘/home/user/sample.txt’
> OVERWRITE INTO TABLE customer;
```

The table structures can be altered using the ALTER commands. It is used to alter the attributes of a table such as changing its table name, changing column names, adding columns, and deleting or replacing columns in Hive. Here is the syntax.

```
ALTER TABLE name RENAME TO new_name
ALTER TABLE name ADD COLUMNS (col_spec[, col_spec ...])
ALTER TABLE name DROP [COLUMN] column_name
ALTER TABLE name CHANGE column_name new_name new_type
ALTER TABLE name REPLACE COLUMNS (col_spec[, col_spec ...])
```

For example, the following query renames the table from **customer** to **cust**, and changes the name and types of fields, and adds a new field.

```
hive> ALTER TABLE customer RENAME TO cust;
hive> ALTER TABLE customer CHANGE name cname String;
hive> ALTER TABLE customer CHANGE cred_llimit Double;
hive> ALTER TABLE customer ADD COLUMNS (state STRING);
```

Tables can also be dropped or deleted. When a table is dropped from Hive Metastore, it removes the table/column data and their metadata. It can be a normal table (stored in Metastore) or an external table (stored in local file system); Hive treats both in the same manner, irrespective of their types.

```
DROP TABLE [IF EXISTS] table_name;
```

12.2.4 Hive Partitioning

Consistent with the underlying data storage in HDFS, Hive organizes tables into partitions. It is a way of dividing a table into related parts based on the values of partitioned columns such as date and state. Using partition, it is easy to query a portion of the data.

Tables or partitions are sub-divided into **buckets,** to provide extra structure to the data that may be used for more efficient querying. Bucketing works based on the value of hash function of some column of a table.

For example, a table named **Tab1** contains customer data. Suppose you need to retrieve the details of all customers from a particular year. A query searches the whole table for the required information. However, if you partition the customer data with the year and store it in a separate file, it reduces the query processing time.

The following example shows how to partition a file and its data:

The following file contains customer data table.

```
/table1/ customer /file1

id, name, state, year

The above data is partitioned into two files using year.

/table1/ customer /2015/file2

/table1/ customer /2016/file3
```

Partitions are added by altering the table.

```
ALTER TABLE table_name ADD [IF NOT EXISTS] PARTITION partition_spec
[LOCATION 'location1'] partition_spec [LOCATION 'location2'] ...;
```

The following query is used to add a partition to the customer table. And to rename a partition. And to drop a partition.

```
> ALTER TABLE customer
> ADD PARTITION (year='2015')
> location '/2015/part2015';
```

```
ALTER TABLE customer PARTITION (year='2015')
 > RENAME TO PARTITION (Year = '2015');
ALTER TABLE customer DROP [IF EXISTS]
 > PARTITION (year='2015');
```

12.2.5 Hive Data Manipulation

The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data. SELECT statement is used to retrieve the data from a table or a set of tables. This section shows how to use the SELECT statement and its various clauses.

Here is the syntax of the SELECT query:

```
SELECT [ALL | DISTINCT] select_expr, select_expr, ...
FROM table_reference
[WHERE where_condition]
[GROUP BY col_list]
[HAVING having_condition]
[ORDER BY col_list]]
[CLUSTER BY col_list | [DISTRIBUTE BY col_list] [SORT BY col_list]]
[LIMIT number];
```

The WHERE clause works similar to a condition. It filters the data using the condition and gives you a finite result. The built-in operators and functions generate an expression, which fulfils the condition.

Hive also offers a rich set of SQL-like operators. There are four types of operators in Hive:

Relational Operators such as EQUAL, GREATER, etc.

Arithmetic Operators, such as +, -, /, etc.

Logical Operators, such as AND, OR, NOT, etc.

Complex Operators such as Arrays, Key-value pairs, and Structs.

The following complex operators provide an expression to access the elements of Complex Types. Examples of such functions is shown later.

Operator	Description
A[n]	It returns the nth element in the array A. The first element has index 0.
M[key]	It returns the value corresponding to the key in the M<key,value> map.
S.x	It returns the x field of S.

The following query retrieves the customer details

```
hive> SELECT * FROM customer WHERE state="CA";
```

The following queries retrieve the customer details and orders them by state, and groups the counts by state.

```
hive> SELECT Id, Name, state FROM customer ORDER BY state;

hive> SELECT count(*) FROM customer GROUP BY state;
```

JOIN is a clause that is used for combining specific fields from two tables by using values common to each one. It is used to combine records from two or more tables in the database, just like SQL JOINS. There are different types of joins:

```
JOIN

LEFT OUTER JOIN

RIGHT OUTER JOIN

FULL OUTER JOIN
```

A JOIN is same as OUTER JOIN in SQL. A JOIN condition is to be raised using the primary keys and foreign keys of the tables.

```
Join_table:


 table_reference JOIN table_factor [join_condition]
 | table_reference {LEFT|RIGHT|FULL} [OUTER] JOIN table_reference
 join_condition
 | table_reference LEFT SEMI JOIN table_reference join_condition
 | table_reference CROSS JOIN table_reference [join_condition]
```

As an example, the following query executes JOIN on the CUSTOMER and ORDER tables, and retrieves the records:

```
hive> SELECT c.ID, c.NAME, c.AGE, o.AMOUNT
 > FROM CUSTOMERS c JOIN ORDERS o
 > ON (c.ID = o.CUSTOMER_ID);
```

A LEFT JOIN returns all the values from the left table, plus the matched values from the right table, or NULL in case of no matching JOIN predicate. The following query demonstrates LEFT OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE

 > FROM CUSTOMERS c
 > LEFT OUTER JOIN ORDERS o
 > ON (c.ID = o.CUSTOMER_ID);
```

A RIGHT JOIN returns all the values from the right table, plus the matched values from the left table, or NULL in case of no matching join predicate. The following query demonstrates RIGHT OUTER JOIN between the CUSTOMER and ORDER tables.

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE
 > FROM CUSTOMERS c
 > RIGHT OUTER JOIN ORDERS o
 > ON (c.ID = o.CUSTOMER_ID);
```

The HiveQL FULL OUTER JOIN combines the records of both the left and the right outer tables that fulfil the JOIN condition. The joined table contains either all the records from both the tables, or fills in NULL values for missing matches on either side. The following query demonstrates FULL OUTER JOIN between CUSTOMER and ORDER tables:

```
hive> SELECT c.ID, c.NAME, o.AMOUNT, o.DATE

 > FROM CUSTOMERS c
 > FULL OUTER JOIN ORDERS o
 > ON (c.ID = o.CUSTOMER_ID);
```

Hive Built-in functions

Hive supports the following built-in functions (Table 12.1). The functions look quite similar to SQL functions.

Table 12.1

Hive built-in functions and descriptions

	Command	Description
Numeric functions	round(n)	returns the rounded value of the n.
	floor(n)	returns the maximum value that is equal or less than n.
	ceil(n)	returns the minimum value that is equal or greater than n.
	rand()	returns a string of random numbers that change from row to row.
String functions	concat(A, B)	returns the string resulting from concatenating B after A.
	substr(A, n)	returns the substring of A starting from position n till the end of A.
	upper(A)	returns a string a all characters of A in upper case.
	lower(A)	returns a string a all characters of B in lower case.
	trim(A)	returns a string after trimming spaces from both ends of A.
	ltrim(A)	returns the string after trimming spaces from the left side of A.
	rtrim(A)	returns the string after trimming spaces from the right side of A.
Date	from_unixtime (n)	converts the number of seconds from Unix era (1970-01-01 00:00:00) to a date string in the in the format of “1970-01-01 00:00:00”.
	to_date(string timestamp)	returns the date part of a timestamp string: to_date(“2010-01-01 00:00:00”) = “2010-01-01”.
	year(string date)	returns the year part of a date or a timestamp string.
	month (string date)	returns the month part of a date or a timestamp string.
	day(date)	returns the day part of a date or a timestamp string.
Array functions	Size (Map<K.V>)	returns the number of elements in the map type.
	Size (A)	returns the number of elements in the array type.
	cast(<expr> as <type>)	converts the results of the expression expr to <type>. Returns NULL if the conversion does not succeed.

The following queries demonstrate built-in functions:

```
hive> SELECT round(2.6) from temp;
hive> SELECT upper(name) from customer;
hive> SELECT month(order_date) from orders;
```

12.2.6 Hive View and Indexes

Views are generated based on user requirements. You can save any result set data as a view. The usage of view in Hive is same as that of the view in SQL. It is a

One can create a view at the time of executing a SELECT statement.

```
CREATE VIEW [IF NOT EXISTS] view_name [(column_name [COMMENT column_comment], ...) ]

[COMMENT table_comment]

AS SELECT ...
```

The following query retrieves customer details:

```
hive> CREATE VIEW cust_CA AS

 > SELECT * FROM customer
 > WHERE state="CA";
```

One can drop this view with:

```
hive> DROP VIEW cust_CA;
```

An Index is a pointer on a particular column of a table. Creating an index means creating a pointer on a particular column of a table. Its syntax is as follows:

```
CREATE INDEX index_name
ON TABLE base_table_name (col_name, ...)
AS 'index.handler.class.name'
[WITH DEFERRED REBUILD]
[IDXPROPERTIES (property_name=property_value, ...)]
[IN TABLE index_table_name]
[PARTITIONED BY (col_name, ...)]
[
 [ ROW FORMAT ...] STORED AS ...
 | STORED BY ...
]
[LOCATION hdfs_path]
[TBLPROPERTIES (...)]
```

```
The following query creates an index:

hive> CREATE INDEX index_state ON TABLE customer(state)
 > AS 'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler';
```

It is a pointer to the state column. If the column is modified, the changes are stored using an index value.

The index can be deleted with a DROP INDEX command.

12.3 APACHE PIG

Pig is a high-level procedural language/platform used for programming on Hadoop and MapReduce. Pig helps to create a step-by-step flow of data to do processing. It operates mostly on the client side of the cluster. Pig has two major components.

- Pig Latin is a high-level language that provides various operators using which programmers can develop their own functions for reading, writing, and processing data.
- Pig Engine then converts Pig Latin scripts into MapReduce jobs, and optimizes their execution automatically.

Pig Latin follows a procedure programming model and is more natural to use to build a data pipeline, such as for an ETL job. It gives full control over how the data flows through the pipeline, when to checkpoint the data in pipeline, and it support DAGs in pipeline such as split, and gives more control over optimization. Pig works well with unstructured data. For complex operations such as analyzing matrices, or search for patterns in unstructured data, Pig gives greater control and options. Pig works well for programmers who are not so good at Java, and yet want detailed control over data flows. Compared with Java, Pig reduces the length of codes by a factor of 10, and thus drastically reduces development time.

Pig allows one to load data and user code at any point in the pipeline. This can be important for ingesting streaming data from satellites or instruments. Pig also uses lazy evaluation. It is faster in the data import but slower in actual execution than an RDBMS friendly language like Hive. Pig is well suited to parallelization and so it is better suited for very large datasets throughput (amount of data processed) is more important than latency (speed of response).

Pig Latin is the language used to write code to analyze data in Hadoop using Pig. It is a high-level data processing language which provides a rich set of data types and operators to perform various operations on the data. The Pig Latin scripts are then executed using any of the execution mechanisms (Grunt Shell, UDFs, Embedded). Internally, Apache Pig converts these scripts into a series of MapReduce jobs, and thus, it makes the programmer's job easy.

Architecture

The architecture of Apache Pig is shown in Figure 12.3.

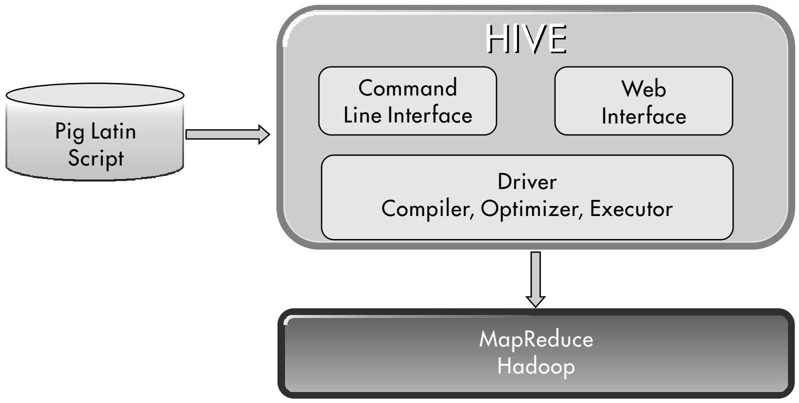


FIGURE 12.3 The architecture of Apache Pig

The programmer uses one of the Pig interfaces to communicate Pig Latin commands or a script. The parser checks the syntax of the Pig Lain script, and data type checking. The output of the parser will be a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators. The logical execution plan (DAG) is passed to the logical optimizer, which carries out the logical optimizations such as projection and pushdown. The compiler compiles the optimized logical plan into a series of MapReduce jobs. The MapReduce jobs are submitted to Hadoop in a sorted order. These MapReduce jobs are executed on Hadoop and the desired results are returned to the user.

12.3.1 Running Pig

It is essential that Hadoop and Java are installed on the system before installing Apache Pig. Download and install the latest version of Apache Pig from https://pig.apache.org/

Pig can be run in two modes, Local and HDFS. In the local mode, all the files are installed and run from your local host and local file system. There is no need of Hadoop or HDFS. This mode is generally used for testing purpose.

```
$ ./pig -x local
```

MapReduce mode is where we load or process the data that exists in the Hadoop File System (HDFS) using Apache Pig. In this mode, whenever we execute the Pig

Latin statements to process the data, a MapReduce job is invoked in the back-end to perform operations on the data that resides on HDFS.

```
$ ./pig -x mapreduce
```

Pig Latin scripts can be executed in three ways, namely, interactive mode, batch mode, and embedded mode. In the Interactive Mode, Pig runs using the Grunt shell. In this shell, you can enter the Pig Latin statements and get the output.

```
grunt> customers = LOAD 'customers.txt' USING PigStorage(',');
```

In the Batch Mode (Script), Pig Latin script is a single file with .pig extension. It can also be run in the Embedded Mode (UDF) – by defining our own functions (User Defined Functions) in programming languages such as Java, and using them in the Pig script.

```
customer = LOAD 'hdfs://localhost:9000/pig_data/customer.txt' USING
 PigStorage(',') as (id:int,name:chararray,city:chararray);
```

The Grunt shell of Apache Pig is mainly used to write Pig Latin scripts. It also provides a set of utility commands. These include utility commands such as clear, help, history, quit, and set; and commands such as exec, kill, and run to control Pig from the Grunt shell.

12.3.2 Pig Latin Data Model

Pig Latin supports many basic data types including int, long, float, chararray, bytearray, Boolean, DateType. It also supports many complex data types such as Tuple, Bag and Relation and also map.

Atom: Any single value in Pig Latin, is known as an Atom. It is stored as a string variable. A simple atomic value is also known as a field.

Tuple: A tuple of record is an ordered set of fields. The fields can be of any type. A tuple is like a row in an RDBMS table.

Bag: A bag is an unordered set of tuples. Each tuple can have any number of fields (flexible schema). A bag is represented by '{}'. Unlike a table in RDBMS, however, it is not necessary that every tuple contain the same number of fields or that the fields in the same position (column) have the same type.

Relation: A relation is a bag of tuples. The relations in Pig Latin are unordered. There is no assurance that tuples will be processed in a particular order.

Map: A data map is a set of key-value pairs. The key is to be of type char-array and should be unique. The value might be of any data type.

12.3.3 Pig Latin Operators

Pig also offer Type construction operators such as () for Tuple, {} for Bag, and [] for Map. Pig also supports all the standard arithmetic operators, comparison operators, and the loop operators (CASE), and WHEN-THEN-ELSE decision-making constructs.

Pig Latin supports many Relational operations. Some of the more important ones are shown in Table 12.2.

Table 12.2

Important relational operations supported by Pig Latin operators

Type	Operator	Description
Data Definition	LOAD	Load the data from the file system (local/HDFS) into a relation.
	STORE	Save a relation to the file system (local/HDFS).
Filtering	FILTER	Remove unwanted rows from a relation.
	DISTINCT	Remove duplicate rows from a relation.
	FOREACH, GENERATE	Generate data transformations based on columns of data.
Grouping, joining, combining	JOIN	Join two or more relations.
	GROUP	Group the data in a single relation.
	COGROUP	Group the data in multiple relations.
	CROSS	Create the cross product of two or more relations.
	UNION	Combine two or more relations into one relation.
	SPLIT	Split a single relation into two or more relations.
Sorting	ORDER	To arrange a relation in a sorted order based on one or more fields (ascending or descending).
	LIMIT	Limit the number of tuples from a relation.
Diagnostic	DUMP	Print the contents of a relation on the console.
	DESCRIBE	Describe the schema of a relation.
	EXPLAIN	View the logical, physical, or MapReduce execution plans to compute a relation.
	ILLUSTRATE	View the step-by-step execution of a series of statements.

12.3.4 Pig Data Definition

In general, Apache Pig works on top of Hadoop Distributed File System. To analyze data using Apache Pig, one needs to load the data into Pig. In MapReduce mode, Pig reads (loads) data from HDFS and stores the results back in HDFS. The input file of Pig contains each tuple/record in individual lines. And, the entities of the record are separated by a delimiter.

In the local file system, create an input file customer_data.txt. Then move the file from the local file system to HDFS using put command as shown below. (Alternatively, one can use copyFromLocal command as well.)

```
$ cd $HADOOP_HOME/bin
$ hdfs dfs -put /home/Hadoop/Pig/Pig_Data/customer_data.txt dfs://localhost:9000/
pig_data/
```

Use the cat command to verify whether the file has been moved into the HDFS, as shown below.

```
$ cd $HADOOP_HOME/bin
$ hdfs dfs -cat hdfs://localhost:9000/pig_data/customer_data.txt
```

One can also load data into Apache Pig from the file system (HDFS/Local) using LOAD operator of Pig Latin. The left-hand side of the load statement is the name of the relation **where** we want to store the data. On the right-hand side, we define **how** to store the data. The syntax is as following:

```
Relation_name = LOAD 'Input file location' USING function as schema;
```

Where:

Input file location is the HDFS directory where the file is stored in MapReduce mode.

function - choose from a set of load functions (BinStorage, JsonLoader, PigStorage, TextLoader).

- Schema - (column1 : data type, column2 : data type, column3 : data type);

One can load the data from the file **customer_data.txt** into Pig by executing the following Pig Latin statement in Grunt shell.

```
    grunt> customer = LOAD 'hdfs://localhost:9000/pig_data/customer_data.txt' USING
    PigStorage(',')as
( id:int, name:chararray, phone:chararray, state:chararray );
```

One can also store the loaded data in the file system using the **store** operator. For example, store the relation in the HDFS directory **"/pig_Output/"** as shown below.

```
grunt> STORE customer INTO ‘ hdfs://localhost:9000/pig_Output/ ‘ USING PigStorage
(‘,’);
```

12.3.5 Pig Diagnostic Operators

One can verify the execution of the Load and other statements, using Diagnostic Operators. Pig Latin provides four different types of diagnostic operators −

- Dump operator
- Describe operator
- Explanation operator
- Illustration operator

The Dump operator is used to display the results on the screen.

```
grunt> Dump Relation_Name
```

for example, the following command will list the contents of the customer table.

```
grunt> Dump customer
```

The Describe operator is used to view the schema of a relation. The following statement will produce the following output.

```
grunt> describe customer;
grunt> customer: { id: int, name: chararray, phone: chararray, state: chararray }
```

The Explain operator is used to display the logical, physical, and MapReduce execution plans of a relation.

```
grunt> explain customer;
```

The Illustrate operator gives you the step-by-step execution of a sequence of statements.

```
grunt> illustrate customer;
```

12.3.6 Pig Data Manipulation

The GROUP operator is used to group the data in one or more relations. It summarizes the data having the same key value.

```
grunt> customer_by_state = GROUP customer by state;
```

We can also group the relation by multiple fields.

The COGROUP operator works with statements involving two or more relations. The co-group operator groups the tuples from each relation according to common field.

The JOIN operator is used to combine records from two or more relation n a shared key fields. When these keys match, the two particular tuples are matched, else the records are dropped. Joins can be of the following types based on the level of participation of the two relations:

Self-join

Inner-join

Outer-join – left join, right join, and full join

Assume again that there are two files namely customers.txt and orders.txt in the / pig_data/ directory of HDFS as shown below. And you have loaded these two files into Pig with the relations customers and orders as shown below.

```
grunt> customers = LOAD 'hdfs://localhost:9000/pig_data/customers.txt' USING
PigStorage(',')
```

as (id:int, name:chararray, phone:int, address:chararray, state:chararray);

```
grunt> orders = LOAD 'hdfs://localhost:9000/pig_data/orders.txt' USING PigStor-
age(',')
```

as (oid:int, date:chararray, customer_id:int, amount:int);

Self-join is used to join a table with itself as if the table were two relations, temporarily renaming at least one relation.

To perform self-join, the same data will be loaded multiple times, under different aliases (names). Therefore, let us load the contents of the file customers.txt as two tables as shown below.

```
grunt> customers1 = LOAD 'hdfs://localhost:9000/pig_data/customers.txt' USING
PigStorage(',')
```

as (id:int, name:chararray, age:int, address:chararray, salary:int);

```
grunt> customers2 = LOAD 'hdfs://localhost:9000/pig_data/customers.txt' USING
PigStorage(',')
```

as (id:int, name:chararray, age:int, address:chararray, salary:int);

Let us perform self-join operation on the relation customers, by joining the two relations customers1 and customers2 as shown below.

```
grunt> customers3 = JOIN customers1 BY id, customers2 BY id;
```

Inner Join is used quite frequently; it is also referred to as equijoin. An inner join returns rows when there is a match in both tables.

Outer Join: Unlike inner join, outer join returns all the rows from at least one of the relations. An outer join operation is carried out in three ways –

1. Left outer join
2. Right outer join
3. Full outer join

The **left outer join** operation returns all rows from the left table, even if there are no matches in the right relation.

```
grunt> outer_left = JOIN customers BY id LEFT OUTER, orders BY customer_id;
```

The **right outer join** operation returns all rows from the right table, even if there are no matches in the left table.

```
grunt> outer_right = JOIN customers BY id RIGHT, orders BY customer_id;
```

The **full outer join** operation returns rows when there is a match in one of the relations.

```
grunt> outer_full = JOIN customers BY id FULL OUTER, orders BY customer_id;
```

The CROSS operator computes the cross-product of two or more relations.

The UNION operator of Pig Latin helps merge the content of two relations with identical structure. i.e. their columns and domains must be identical.

```
grunt> Relation_name3 = UNION Relation_name1, Relation_name2;
```

The SPLIT operator is used to split a relation into two or more relations.

```
grunt> SPLIT Relation1_name INTO Relation2_name IF (condition1), Relation2_name
(condition2),
```

The FILTER operator is used to select the required tuples from a relation based on a condition.

```
grunt> Relation2_name = FILTER Relation1_name BY (condition);
```

The DISTINCT operator is used to remove redundant (duplicate) tuples from a relation.

```
grunt> Relation_name2 = DISTINCT Relatin_name1;
```

The FOREACH operator is used to generate specified data transformations based on the column data.

```
grunt> Relation_name2 = FOREACH Relatin_name1 GENERATE (required data);
```

The ORDER BY operator is used to display the contents of a relation in a sorted order based on one or more fields.

```
grunt> Relation_name2 = ORDER Relatin_name1 BY (ASC|DESC);
```

12.3.7 Pig Built-in Functions

Apache Pig provides various built-in functions namely eval, load, store, math, string, bag and tuple functions.

Table 12.3 provides a representative list of Pig Latin functions.

Table 12.3

A representative list of Pig Latin functions

Type	Function Name	Description
Bag	AVG()	Compute the average of the numerical values within a bag.
	SUM()	Get the total of the numeric values of a column in a single-column bag.
	COUNT()	Get the number of elements in a bag, while counting the number of tuples in a bag.
	COUNT_STAR()	Get the number of elements in a bag.
	IsEmpty()	Check if a bag or map is empty.
	MAX()	Get the highest value for a column (numeric values or chararrays) in a single-column bag.
	MIN()	Get the lowest value (numeric or chararray) for a certain column in a single-column bag.
	PluckTuple()	Filter the columns in a relation beginning with the given prefix.
	SIZE()	Compute the number of elements of any Pig data type.

(*Contd.*)

(Contd.)

Type	Function Name	Description
	SUBTRACT()	Subtract two bags. Returns a bag which contains the tuples of the first bag that are not in the second bag.
	BagToString()	Concatenate the elements of a bag into a string.
	CONCAT()	Concatenate two or more expressions of same type.
	TOKENIZE()	Split a string in a single tuple and return a bag which contains the output of the split operation.
	PigStorage()	Load and store structured files.
	TextLoader()	Load unstructured data into Pig
	BinStorage()	Load and store data using machine readable format.
	TOBAG()	Convert two or more expressions into a bag.
	TOP()	Get the top N tuples of a relation.
	TOTUPLE()	Convert one or more expressions into a tuple.
	TOMAP()	Convert the key-value pairs into a Map.
String	ENDSWITH (string, test)	Verify whether a given string ends with a test substring.
	STARTSWITH (string, test)	Accept two string parameters and verifies whether the first string starts with the test substring.
	SUBSTRING (string, startIndex, stopIndex)	Return a substring from a given string.
	EqualsIgnoreCase (string1, string2)	Compare two stings ignoring the case.
	INDEXOF (string, 'character', startIndex)	Return the first occurrence of a character in a string, searching forward from a start index.
	LCFIRST(expression)	Convert the first character in a string to lower case.
	UCFIRST(expression)	Return a string with the first character converted to upper case.
	UPPER(expression)	Return a string converted to upper case.
	LOWER(expression)	Return a string converted to lower case.
	REPLACE(string, 'oldChar', 'newChar');	Replace existing characters in a string with new characters.

(Contd.)

(C*ontd.*)

Type	Function Name	Description
	STRSPLIT (string, expr, limit)	Split a string around matches of a given regular expression.
	STRSPLITTOBAG (string, regex, limit)	Similar to the STRSPLIT() function, it splits the string by given delimiter and returns the result in a bag.
	TRIM(expression)	Returns a copy of a string with leading and trailing whitespaces removed.
	LTRIM(expression)	Returns a copy of a string with leading whitespaces removed.
	RTRIM(expression)	Returns a copy of a string with trailing whitespaces removed.
Date	ToDate(milliseconds)	This function returns a date-time object according to the given parameters. The other alternative for this function are ToDate(iosstring), ToDate(userstring, format), ToDate(userstring, format, timezone).
	CurrentTime()	returns the date-time object of the current time.
	GetDay(time)	Returns the day of a month from the date-time object.
	GetHour(time)	Returns the hour of a day from the date-time object.
	GetMinute(time)	Returns the minute of an hour from the date-time object.
	GetSecond(time)	Returns the second of a minute from the date-time object.
	GetMilliSecond(time)	Returns the millisecond of a second from the date-time object.
	GetYear(time)	Returns the year from the date-time object.
	GetMonth(time)	Returns the month of a year from the date-time object.
	GetWeek(time)	Returns the week of a year from the date-time object.
	AddDuration (time, duration)	Returns the result of a date-time object along with the duration object.
	SubtractDuration (time, duration)	Subtracts the Duration object from the Date-Time object and returns the result.
Numeric	ABS(expression)	To get the absolute value of an expression.
	CEIL(expression)	This function is used to get the value of an expression rounded up to the nearest integer.
	FLOOR (expression)	To get the value of an expression rounded down to the nearest integer.
	LOG (expression)	To get the natural logarithm (base e) of an expression.

(C*ontd.*)

(C*ontd.*)

Type	Function Name	Description
	RANDOM( )	To get a pseudo random number (type double) greater than or equal to 0.0 and less than 1.0.
	ROUND (expression)	To get the value of an expression rounded to an integer (if the result type is float) or rounded to a long (if the result type is double).
	SQRT (expression)	To get the positive square root of an expression.

12.3.8 Pig User Defined Functions

In addition to the built-in functions, Pig provides extensive support for User Defined Functions (UDF's). The UDF support is provided in six programming languages, namely, Java, Jython, Python, JavaScript, Ruby and Groovy. Complete support is provided for writing UDFs in Java. Limited support is provided in all the remaining languages. Using Java, one can write UDF's involving all parts of the processing like data load/store, column transformation, and aggregation. Since Pig has been written primarily in Java, the UDF's written using Java language work efficiently compared to other languages.

While writing UDF's using Java, we can create and use the following three types of functions −

- *Filter Functions* − The filter functions are used as conditions in filter statements. These functions accept a Pig value as input and return a Boolean value.
- *Evaluation Functions* − The Evaluation functions are used in FOREACH-GENERATE statements. These functions accept a Pig value as input and return a Pig result.
- *Algebraic Functions* − The Algebraic functions act on inner bags in a FOREACH-GENERATE statement. These functions are used to perform full MapReduce operations on an inner bag.

In Apache Pig, there is a ready Java repository for UDF's named **Piggybank**. Using Piggybank, one can access Java UDF's written by others, and also share our own UDF's.

12.3.9 Running Pig Scripts

Here is a sample Pig Script.

```
grunt> customers = LOAD 'hdfs://localhost:9000/pig_data/customers.txt' USING
PigStorage(',')
```

as (id:int, name:chararray, phone:int, address:chararray, state:chararray);

```
customer_order = ORDER customer BY state DESC;
customer_limit = LIMIT customer_order 5;
Dump customer_limit;
```

This file contains statements performing operations and transformations on the **customer** relation, as shown below. This script will print the top 5 tuples of the customer relation.

There are two ways to run Pig scripts: Batch Mode and HDFS mode.

While executing in batch mode, write all the required Pig Latin statements in a single file and save it as **.pig** file. One can execute it from the Grunt shell using the exec command as shown below.

```
grunt> exec /sample_script.pig
```

When executing a Pig script in HDFS, upload the script to HDFS, and then execute it.

```
$./pig -x mapreduce hdfs://localhost:9000/pig_data/sample_script.pig
```

12.4 CONCLUSION

Hive and Pig are major data access languages in the Hadoop ecosystem. Both are easy-to-use high-level SQL-like languages that ultimately generate MapReduce commands to generate reports. Hive is easier to use but works only on structured data; while Pig is more procedural and offers greater control over data flows. Both offer User-Defined functions to apply complex logic on the data. Both can be operated upon from a command-line interface as well as GUI modes. Both offer rich sets of commands and functions for data definition and data manipulation. This chapter listed many of the operators and built-in functions available in each of the language.

Review Questions

1. What is Apache Hive? Describe its architecture.
2. What is Apache Pig? Describe its architecture.
3. Compare SQL with Hive and Pig.
4. What are the major Data definition commands for Hive?
5. How is data definition done in Pig?

6. How is data manipulation done in Hive?
7. How is data manipulation done in Pig?

True/False Questions

1. Hive is primarily a data definition language.
2. Hive is a more powerful language than SQL.
3. Hive is a substitute for MapReduce programming.
4. Hive works only on HDFS.
5. Hive does not need partitions of data.
6. Hive SELECT statements are very different from those of SQL.
7. Hive creates indexes very similar to those of SQL.
8. Hive is more suitable for transaction processing jobs.
9. Hive is primarily a data definition language.
10. Hive offers a FOREACH command for data transformation.
11. Pig is a substitute for Java programming.
12. Pig works only in HDFS.
13. Pig does not need partitions of data.
14. Pig views are very dissimilar to those of SQL.
15. Pig is more suitable for ETL jobs.
16. Pig script can work from a command-line interface.
17. A bag in pig is an unordered collection of tuples.
18. A Map type in Pig is a collection of <key,value> pairs.
19. A user defined function can be very complex, and yet be easily run in Pig.
20. The DUMP operator in Pig is a way to delete a table.

Appendix 1

Installing Hadoop Using Cloudera on Virtual Box

This tutorial is designed to provide a capability to run Hadoop on local machines, using the Oracle VirtualBox application. Cloudera's Hadoop stack is installed. Then a Wordcount application is written in Java and run on this Hadoop installation. The results are ported back to local mode.

A1.1 HIGH LEVEL SUMMARY

Check the Prerequisites:

1. 64 bit OS, 8 GB RAM
2. At least 10–20 GB free space on hard disk
3. Need Virtualization (VTx) enabled

Download and install VirtualBox software:

Link: https://www.virtualbox.org/wiki/Downloads

Download Cloudera Quickstart:

Link: http://www.cloudera.com/downloads/quickstart_vms/5-8.html

Start Cloudera Quickstart in VirtualBox:

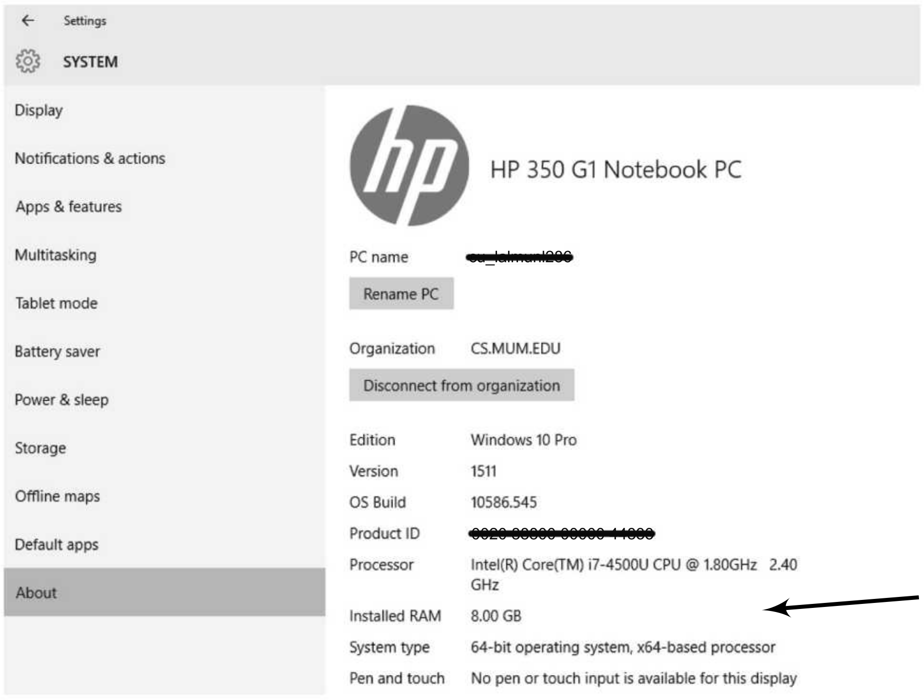
Unzip the "cloudera-quickstart-vm-5.8.0-0-virtualbox" directory

Import from VirtualBox

Create and Run a Wordcount project in Cloudera and in Hadoop

Step 1: Check the Prerequisites

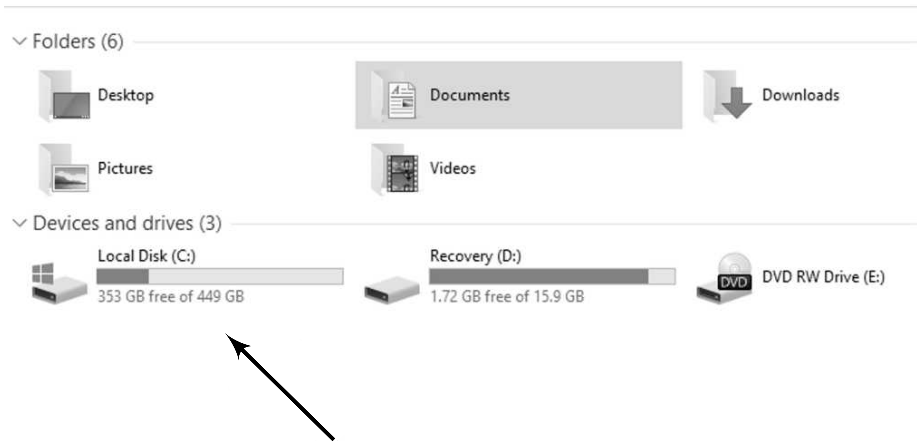
For OS and RAM checking, go to Start Menu -> Settings -> System -> About



HP 350 G1 Notebook PC

PC name	[redacted]
Organization	CS.MUM.EDU
Edition	Windows 10 Pro
Version	1511
OS Build	10586.545
Product ID	[redacted]
Processor	Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz 2.40 GHz
Installed RAM	8.00 GB
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

1. For hard disk checking:

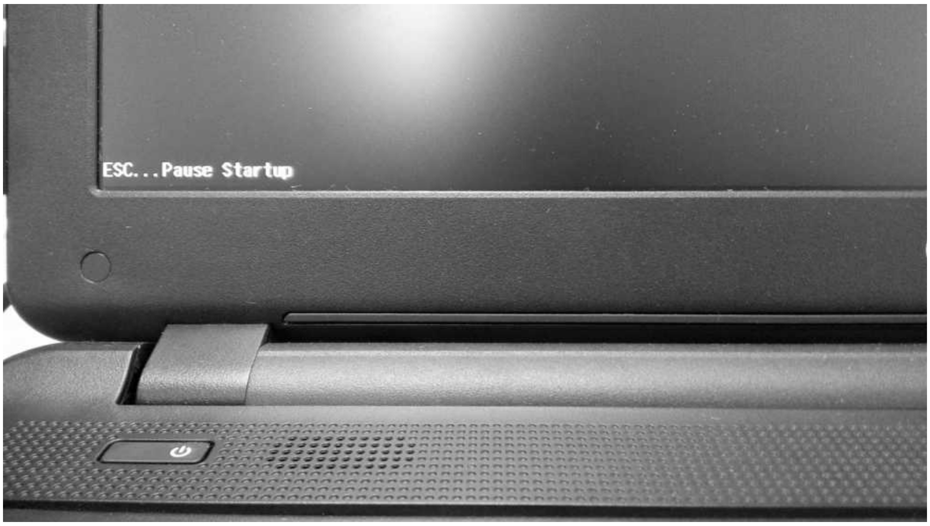


Local Disk (C:)
353 GB free of 449 GB

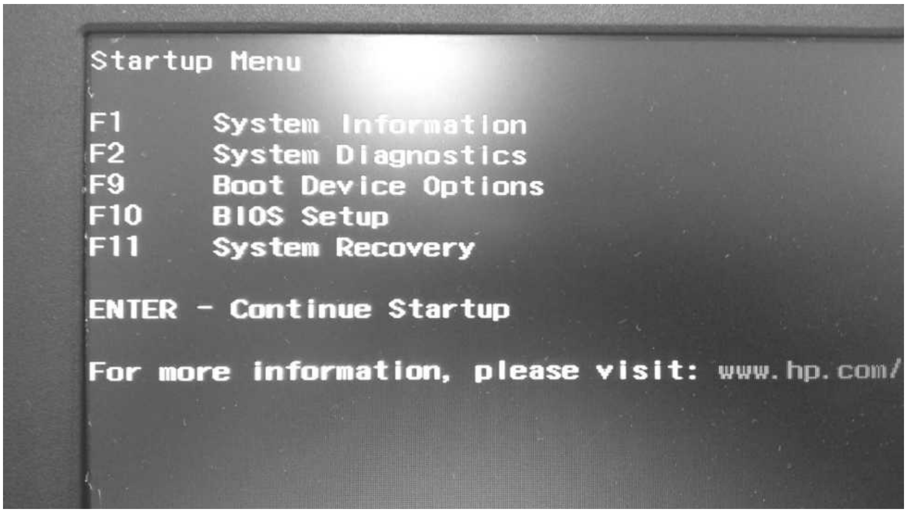
Recovery (D:)
1.72 GB free of 15.9 GB

2. For Virtualization (VTx) making enabled:

Step 1: Press "ESC" to go to BIOS

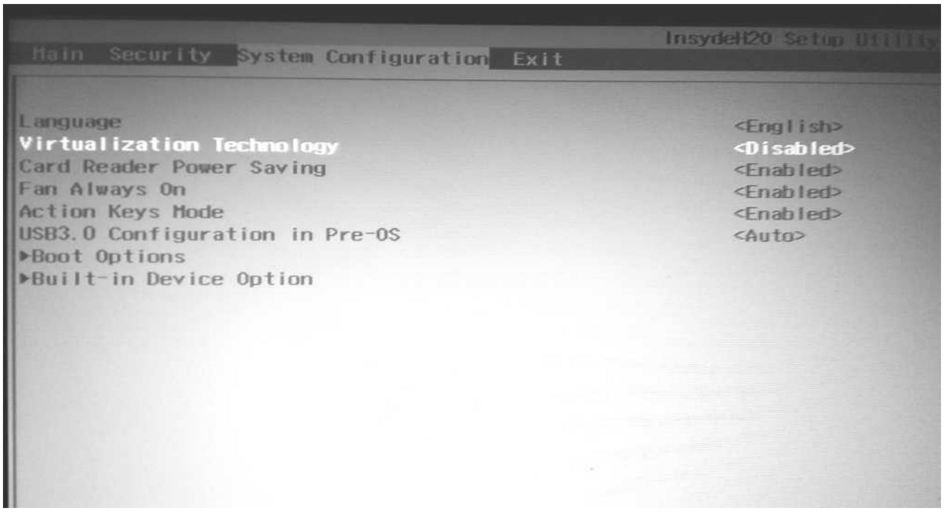


Step 2: Press "F10" for BIOS Setup



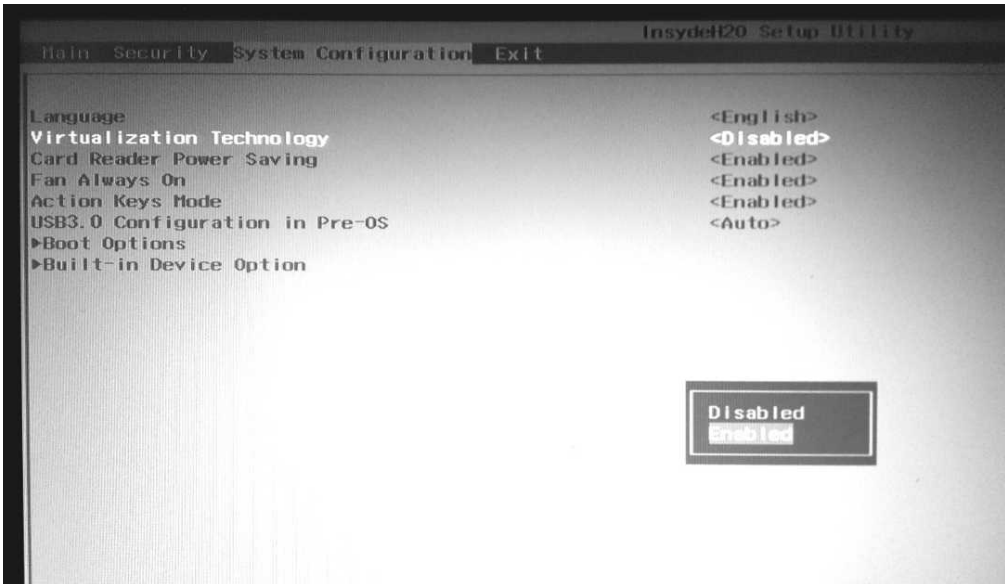
```
Startup Menu

F1      System Information
F2      System Diagnostics
F9      Boot Device Options
F10     BIOS Setup
F11     System Recovery

ENTER - Continue Startup

For more information, please visit: www.hp.com/
```

Step 3: Select “System Configuration”



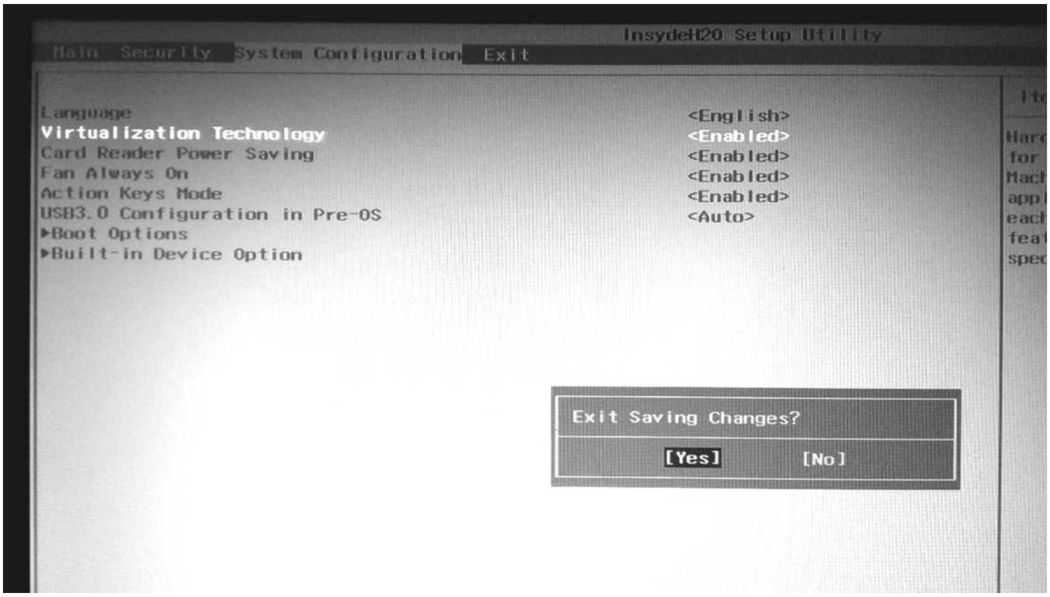
```
                                              InsydeH20 Setup Utility
 Main  Security  System Configuration  Exit

Language                                       <English>
Virtualization Technology                      <Disabled>
Card Reader Power Saving                       <Enabled>
Fan Always On                                  <Enabled>
Action Keys Mode                               <Enabled>
USB3.0 Configuration in Pre-OS                 <Auto>
▶Boot Options
▶Built-in Device Option
```

Step 4: Press “Enter” and change it to “Enabled” then press “Enter” again



```
                                              InsydeH20 Setup Utility
 Main  Security  System Configuration  Exit

Language                                       <English>
Virtualization Technology                      <Disabled>
Card Reader Power Saving                       <Enabled>
Fan Always On                                  <Enabled>
Action Keys Mode                               <Enabled>
USB3.0 Configuration in Pre-OS                 <Auto>
▶Boot Options
▶Built-in Device Option

                                              Disabled
                                              Enabled
```

Step 5: Press "F10" then press "Yes" to save the settings.



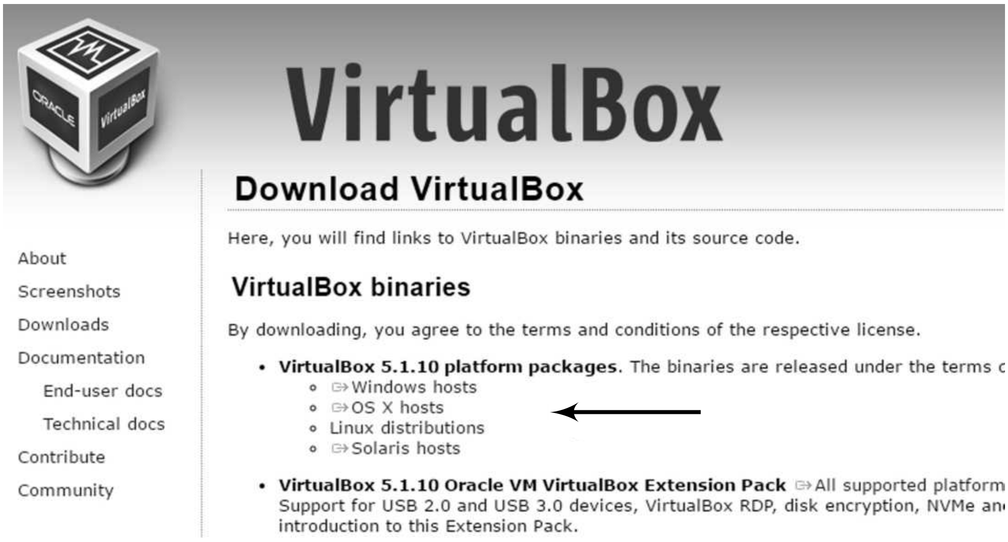
Language <English>
Virtualization Technology <Enabled>
Card Reader Power Saving <Enabled>
Fan Always On <Enabled>
Action Keys Mode <Enabled>
USB3.0 Configuration in Pre-OS <Auto>
▶Boot Options
▶Built-in Device Option

Exit Saving Changes?

[Yes] [No]

Step 2: Download and install VirtualBox software:

1. Download from the link: https://www.virtualbox.org/wiki/Downloads



Download VirtualBox

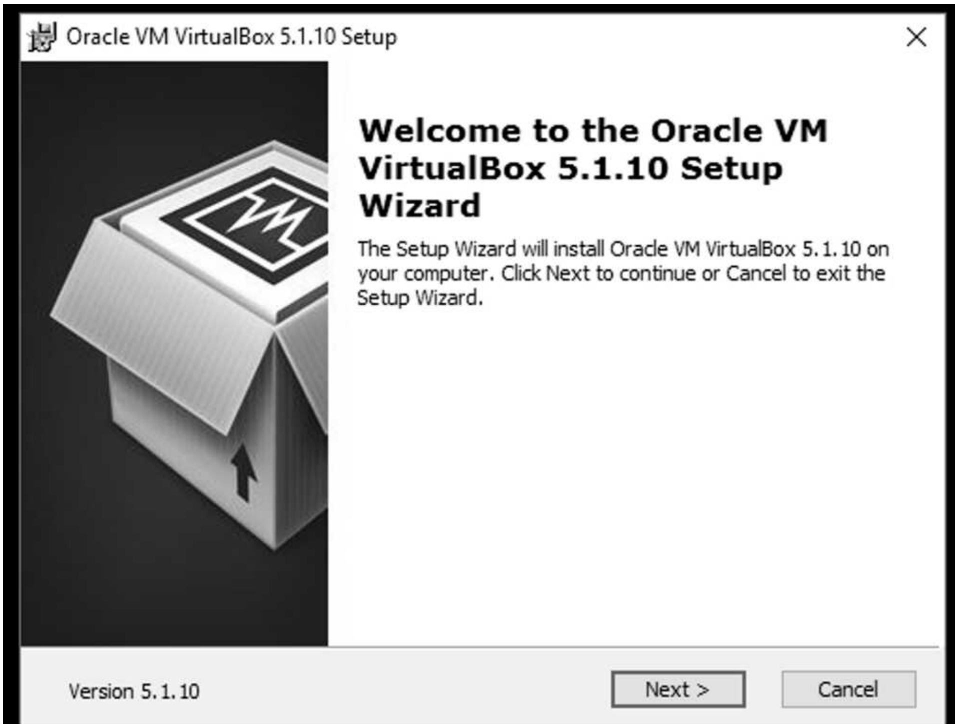
Here, you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

- **VirtualBox 5.1.10 platform packages**. The binaries are released under the terms c
  - Windows hosts
  - OS X hosts
  - Linux distributions
  - Solaris hosts
- **VirtualBox 5.1.10 Oracle VM VirtualBox Extension Pack** All supported platform
  Support for USB 2.0 and USB 3.0 devices, VirtualBox RDP, disk encryption, NVMe an
  introduction to this Extension Pack.

2. Install the VirtualBox by following all the steps

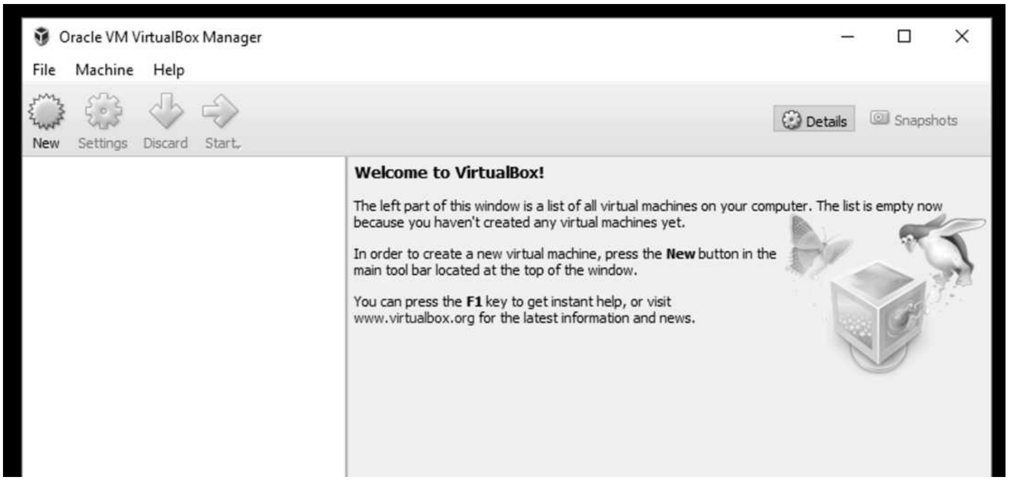


Welcome to the Oracle VM VirtualBox 5.1.10 Setup Wizard

The Setup Wizard will install Oracle VM VirtualBox 5.1.10 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Version 5.1.10

3. Start the VirtualBox



Welcome to VirtualBox!

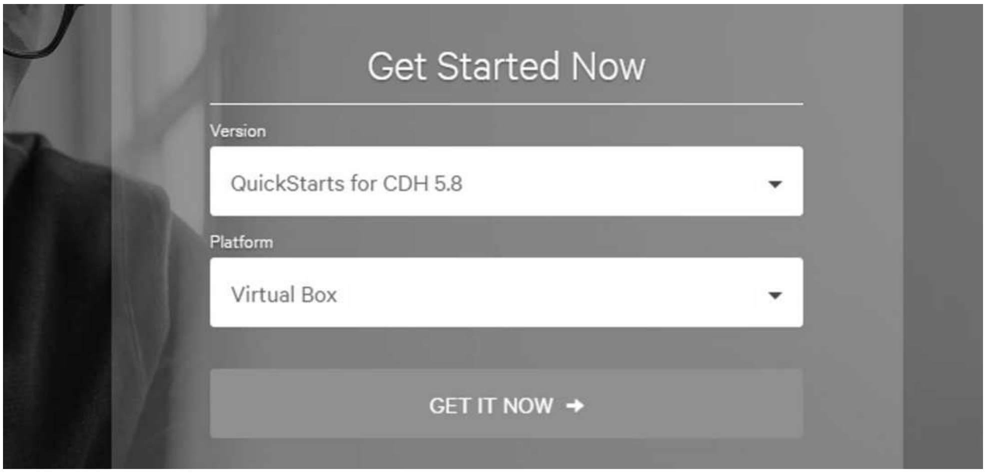
The left part of this window is a list of all virtual machines on your computer. The list is empty now because you haven't created any virtual machines yet.

In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for the latest information and news.

Download Cloudera Quickstart:

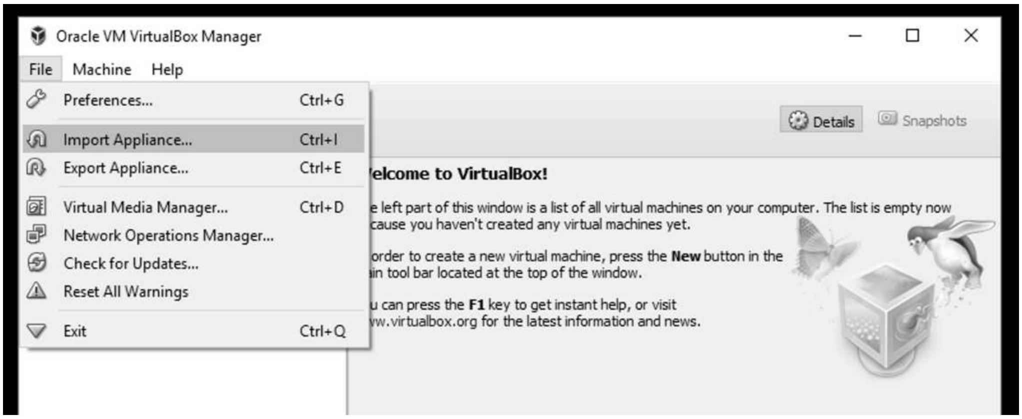
1. Download from the link following the SIGN IN steps: http://www.cloudera.com/downloads/quickstart_vms/5-8.html



Step 3: Start Cloudera Quickstart in VirtualBox

1. Unzip the downloaded “cloudera-quickstart-vm-5.8.0-0-virtualbox” directory
2. Import from VirtualBox:

Step 1: Select “Import Appliance...”


Oracle VM VirtualBox Manager
File
Machine
Help
Preferences... Ctrl+G
Import Appliance... Ctrl+I
Export Appliance... Ctrl+E
Virtual Media Manager... Ctrl+D
Network Operations Manager...
Check for Updates...
Reset All Warnings
Exit Ctrl+Q
Details
Snapshots

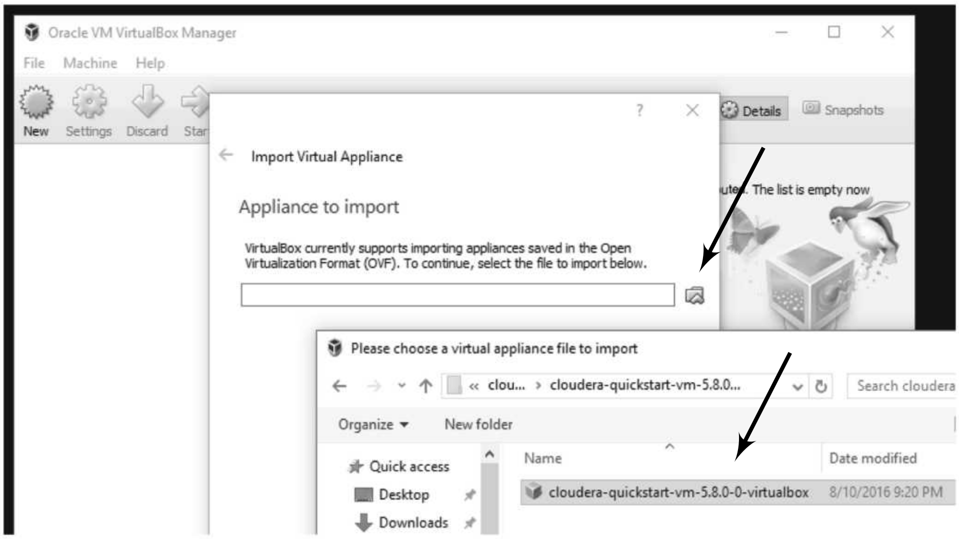
elcome to VirtualBox!

e left part of this window is a list of all virtual machines on your computer. The list is empty now cause you haven't created any virtual machines yet.

order to create a new virtual machine, press the **New** button in the in tool bar located at the top of the window.

u can press the **F1** key to get instant help, or visit w.virtualbox.org for the latest information and news.

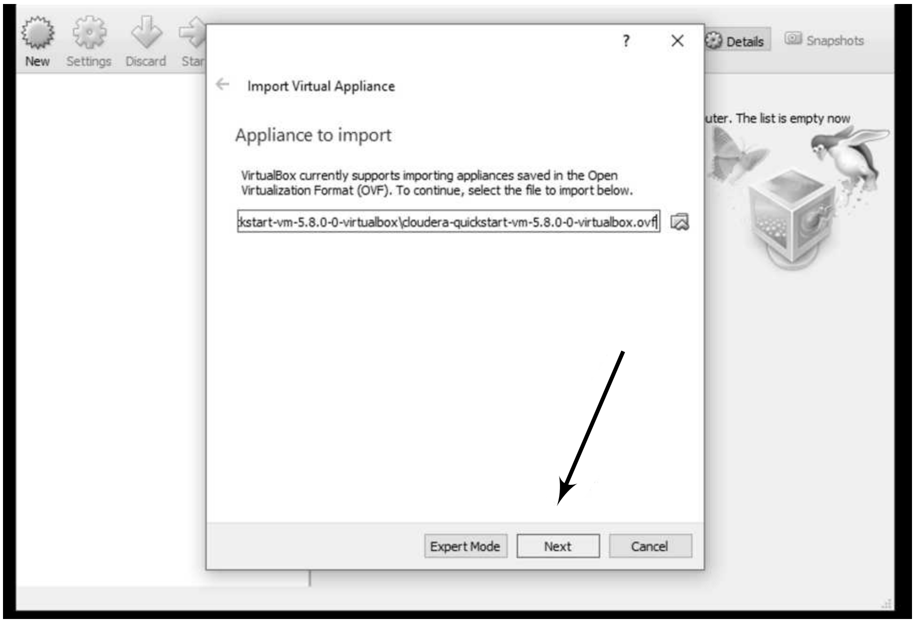
Step 2: Choose the "cloudera-quickstart-vm-5.8.0-0-virtualbox" (.vmx) file



VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

cloudera-quickstart-vm-5.8.0-0-virtualbox 8/10/2016 9:20 PM

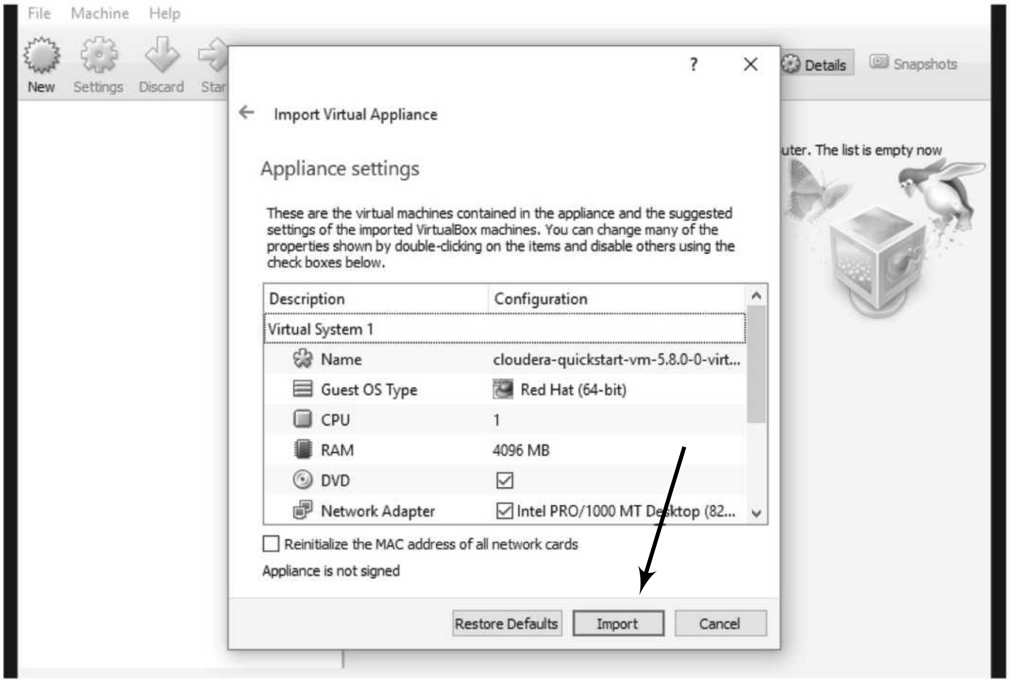
Step 3: Click on Next



VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

kstart-vm-5.8.0-0-virtualbox\cloudera-quickstart-vm-5.8.0-0-virtualbox.ovf

Step 4: Click on Import



Import Virtual Appliance

Appliance settings

These are the virtual machines contained in the appliance and the suggested settings of the imported VirtualBox machines. You can change many of the properties shown by double-clicking on the items and disable others using the check boxes below.

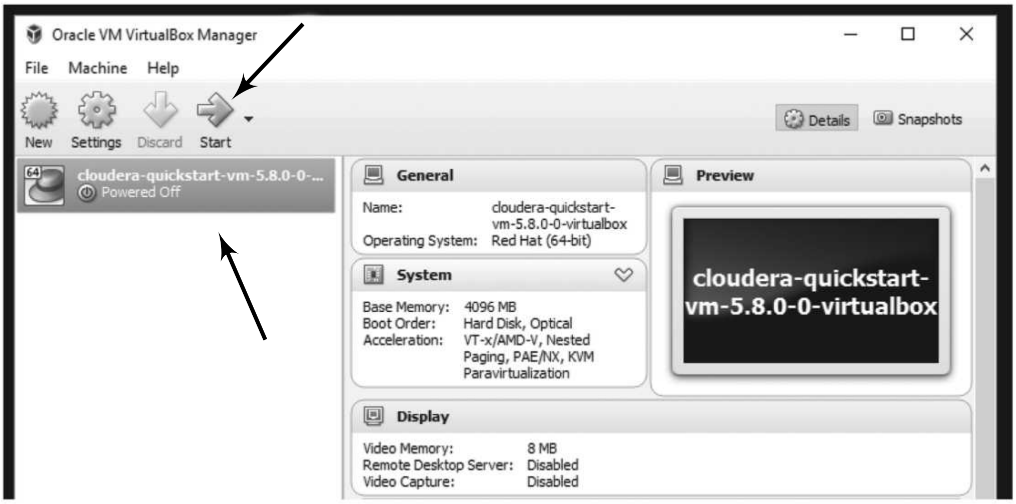
Description	Configuration
Virtual System 1	
Name	cloudera-quickstart-vm-5.8.0-0-virt...
Guest OS Type	Red Hat (64-bit)
CPU	1
RAM	4096 MB
DVD	☑
Network Adapter	☑ Intel PRO/1000 MT Desktop (82...

☐ Reinitialize the MAC address of all network cards

Appliance is not signed

uter. The list is empty now

Step 5: Start Cloudera image after selecting it.

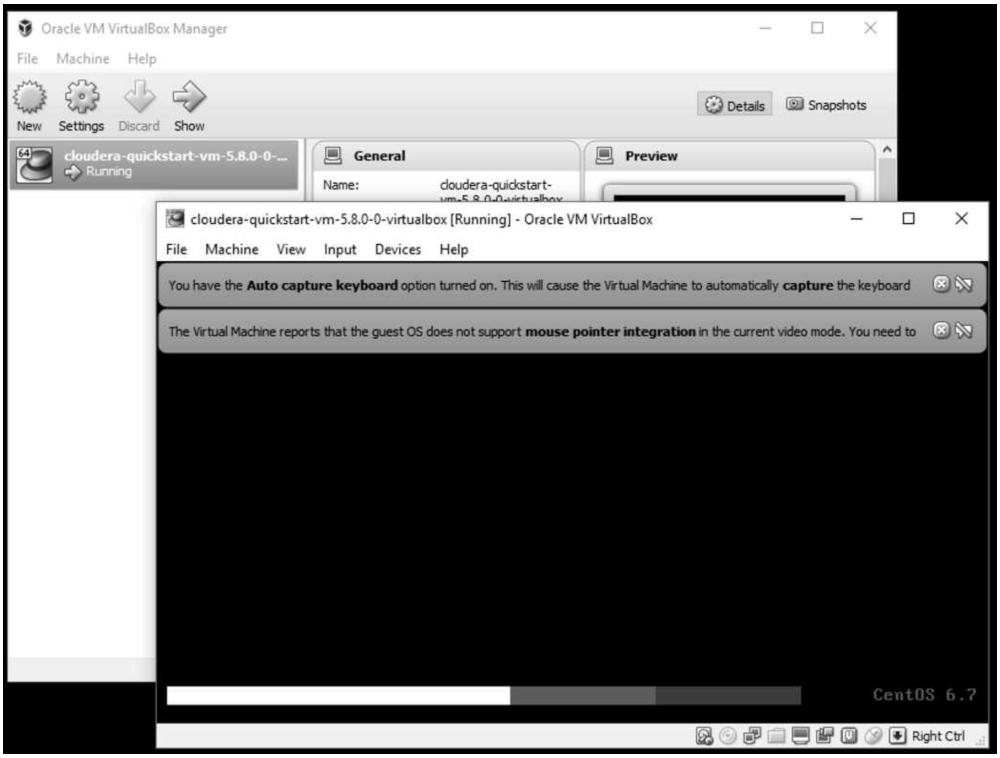

Oracle VM VirtualBox Manager
File
Machine
Help
New
Settings
Discard
Start
Details
Snapshots
cloudera-quickstart-vm-5.8.0-...
Powered Off
General
System
Preview
Display
cloudera-quickstart-vm-5.8.0-0-virtualbox

Name: cloudera-quickstart-vm-5.8.0-0-virtualbox
Operating System: Red Hat (64-bit)

Base Memory: 4096 MB
Boot Order: Hard Disk, Optical
Acceleration: VT-x/AMD-V, Nested Paging, PAE/NX, KVM Paravirtualization

Video Memory: 8 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

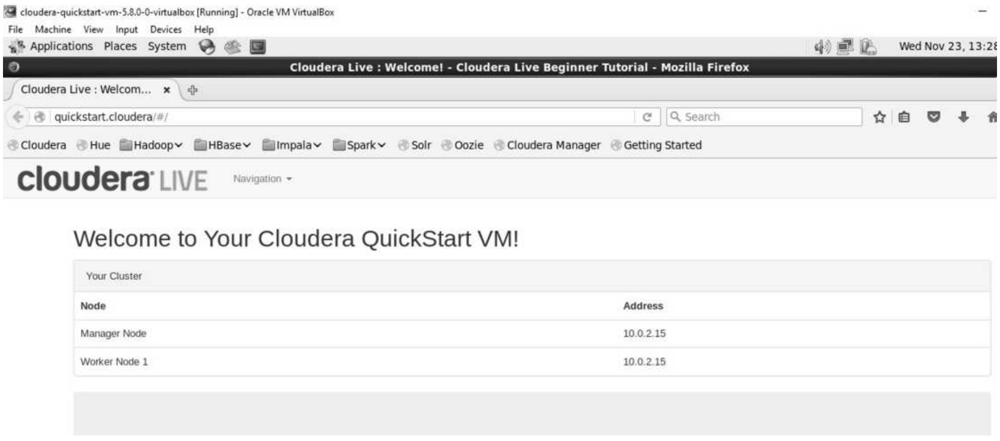
Step 6: Wait and do the needful until starting Cloudera.


Oracle VM VirtualBox Manager
File Machine Help
New Settings Discard Show
Details Snapshots
cloudera-quickstart-vm-5.8.0-0-...
Running
General
Preview
Name: cloudera-quickstart-vm-5.8.0-0-virtualbox
cloudera-quickstart-vm-5.8.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
CentOS 6.7
Right Ctrl

You have the **Auto capture keyboard** option turned on. This will cause the Virtual Machine to automatically **capture** the keyboard

The Virtual Machine reports that the guest OS does not support **mouse pointer integration** in the current video mode. You need to

Then see the Cloudera


cloudera-quickstart-vm-5.8.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Wed Nov 23, 13:28
Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mozilla Firefox
Cloudera Live : Welcom...
quickstart.cloudera/#/
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started
cloudera LIVE
Navigation

Welcome to Your Cloudera QuickStart VM!

Your Cluster

Node	Address
Manager Node	10.0.2.15
Worker Node 1	10.0.2.15

Step 4: Word Count Project

Programming Language:

Java using Eclipse IDE

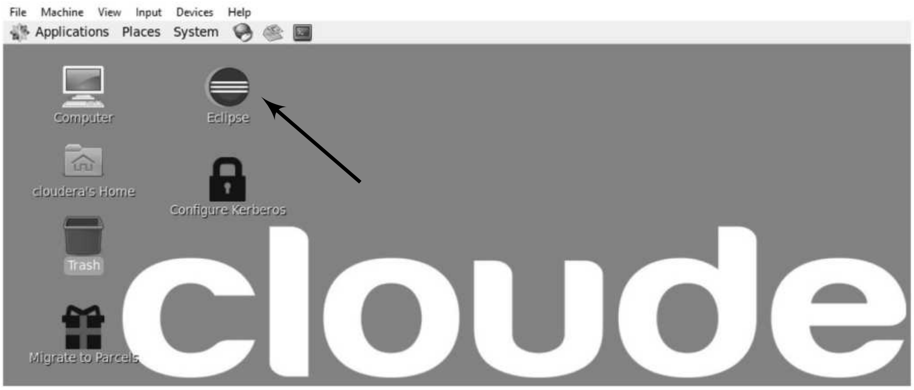
Add external jars to the build path of the project:

File System/usr/lib/hadoop

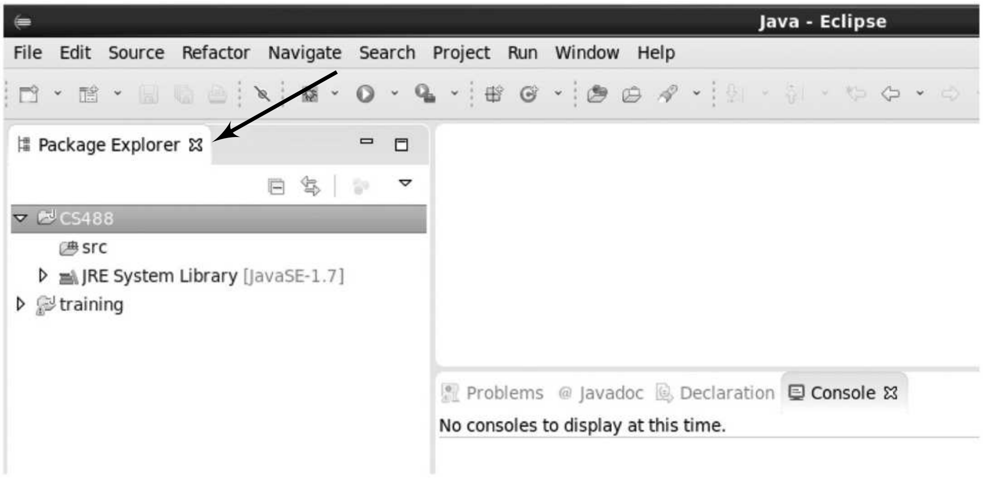
File System/usr/lib/hadoop/client-0.20

File System/usr/lib/hadoop/lib

Step 1: Run Eclipse form Cloudera Desktop

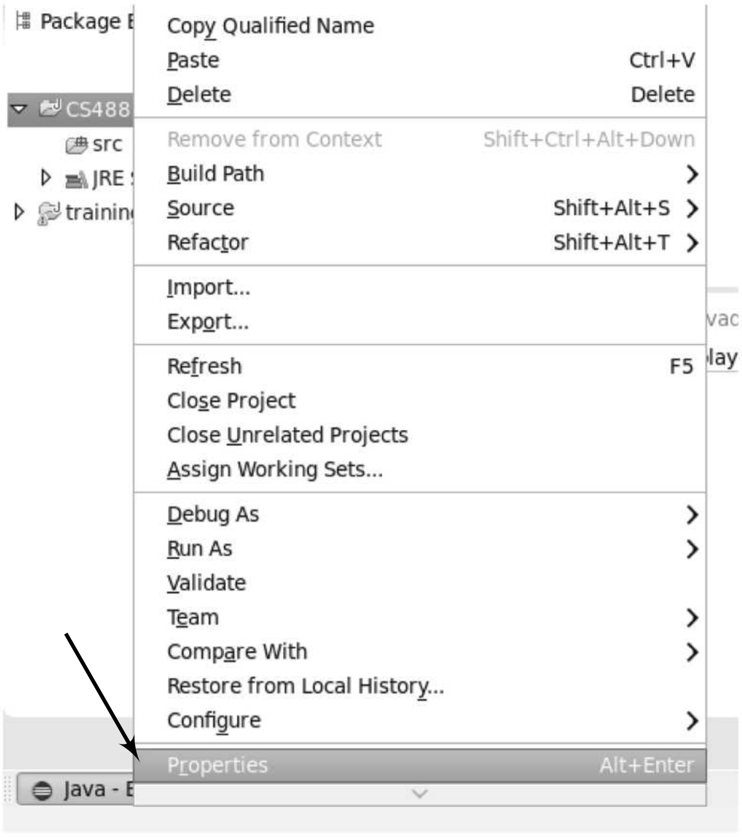


Step 2: Create a project named “CS488” from “File->New->Java Projecct

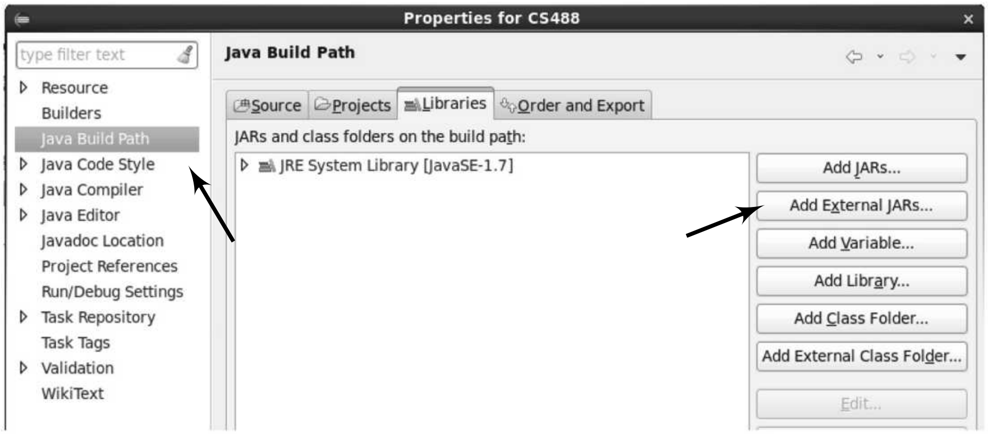


No consoles to display at this time.

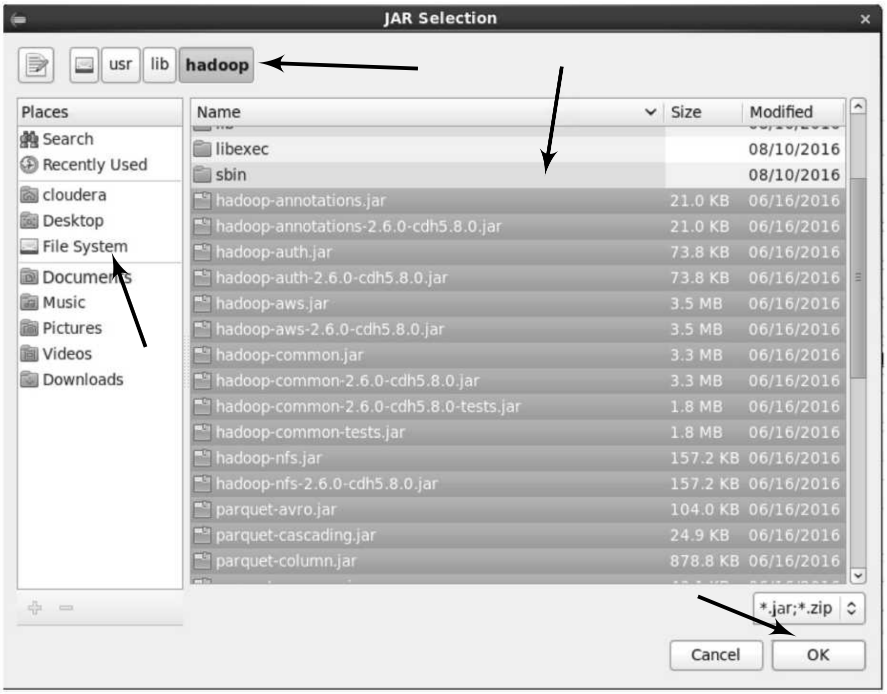
Step 3: Add external jars to the build path of the project



Then

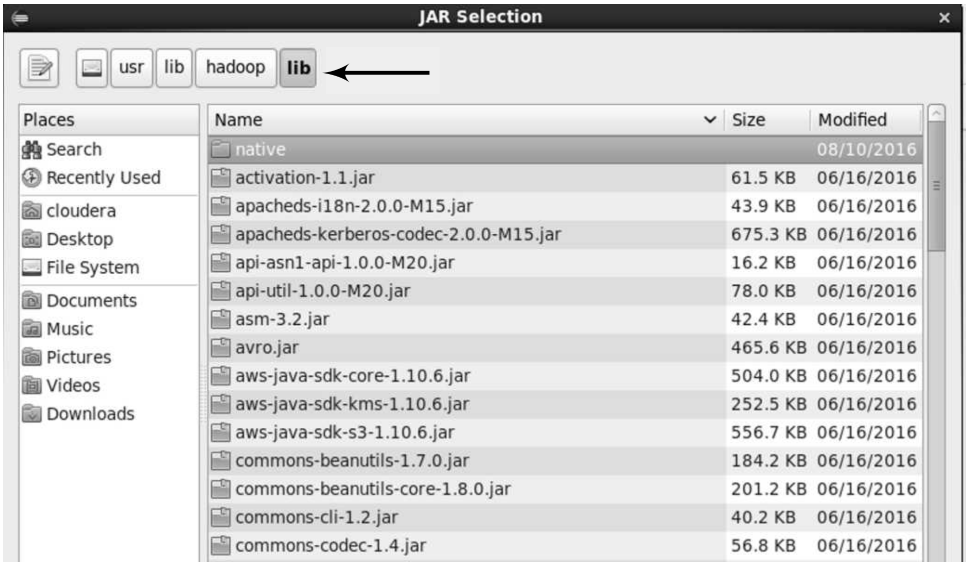


Then click on OK button after selecting all the jars



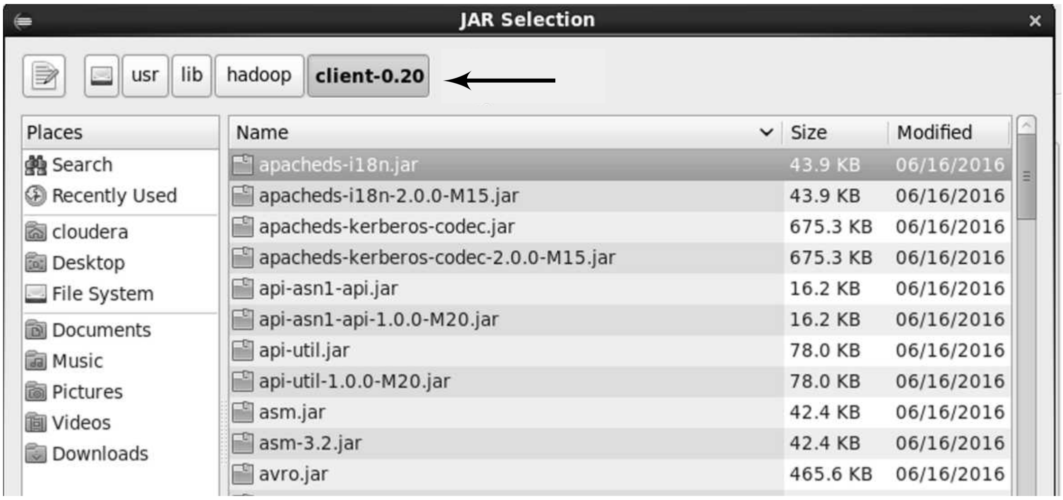
Name	Size	Modified
libexec		08/10/2016
sbin		08/10/2016
hadoop-annotations.jar	21.0 KB	06/16/2016
hadoop-annotations-2.6.0-cdh5.8.0.jar	21.0 KB	06/16/2016
hadoop-auth.jar	73.8 KB	06/16/2016
hadoop-auth-2.6.0-cdh5.8.0.jar	73.8 KB	06/16/2016
hadoop-aws.jar	3.5 MB	06/16/2016
hadoop-aws-2.6.0-cdh5.8.0.jar	3.5 MB	06/16/2016
hadoop-common.jar	3.3 MB	06/16/2016
hadoop-common-2.6.0-cdh5.8.0.jar	3.3 MB	06/16/2016
hadoop-common-2.6.0-cdh5.8.0-tests.jar	1.8 MB	06/16/2016
hadoop-common-tests.jar	1.8 MB	06/16/2016
hadoop-nfs.jar	157.2 KB	06/16/2016
hadoop-nfs-2.6.0-cdh5.8.0.jar	157.2 KB	06/16/2016
parquet-avro.jar	104.0 KB	06/16/2016
parquet-cascading.jar	24.9 KB	06/16/2016
parquet-column.jar	878.8 KB	06/16/2016

Then do same for



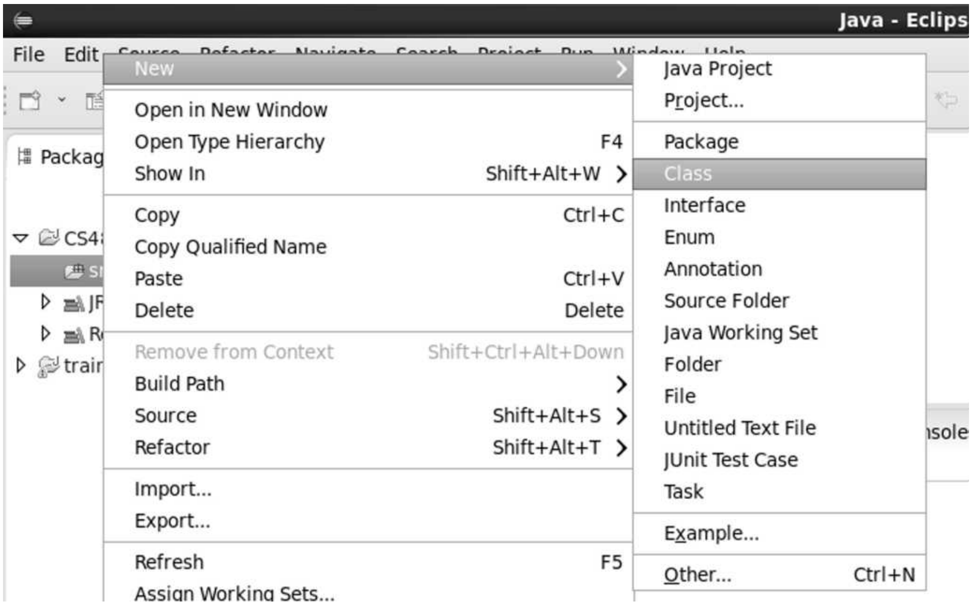
Name	Size	Modified
native		08/10/2016
activation-1.1.jar	61.5 KB	06/16/2016
apacheds-i18n-2.0.0-M15.jar	43.9 KB	06/16/2016
apacheds-kerberos-codec-2.0.0-M15.jar	675.3 KB	06/16/2016
api-asn1-api-1.0.0-M20.jar	16.2 KB	06/16/2016
api-util-1.0.0-M20.jar	78.0 KB	06/16/2016
asm-3.2.jar	42.4 KB	06/16/2016
avro.jar	465.6 KB	06/16/2016
aws-java-sdk-core-1.10.6.jar	504.0 KB	06/16/2016
aws-java-sdk-kms-1.10.6.jar	252.5 KB	06/16/2016
aws-java-sdk-s3-1.10.6.jar	556.7 KB	06/16/2016
commons-beanutils-1.7.0.jar	184.2 KB	06/16/2016
commons-beanutils-core-1.8.0.jar	201.2 KB	06/16/2016
commons-cli-1.2.jar	40.2 KB	06/16/2016
commons-codec-1.4.jar	56.8 KB	06/16/2016

And



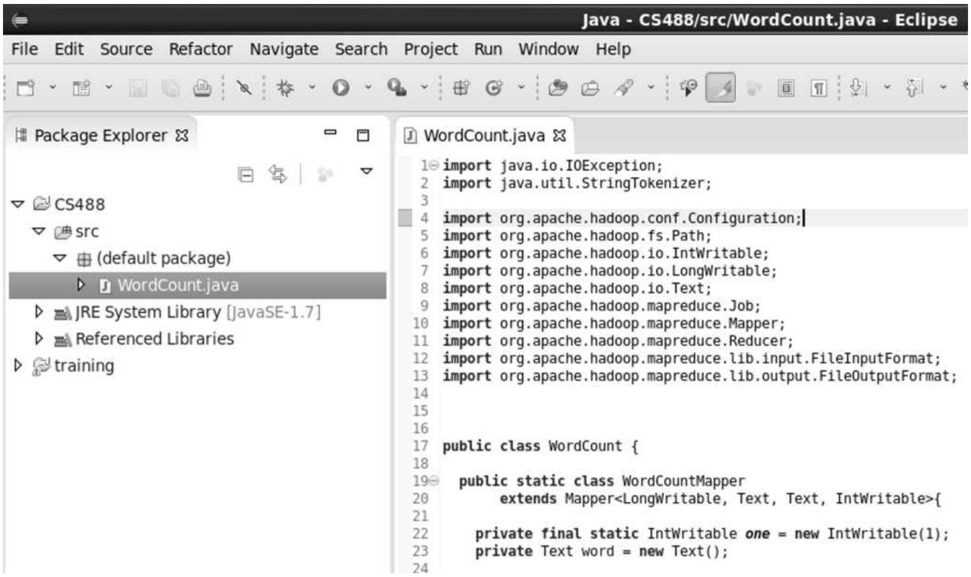
Name	Size	Modified
apacheds-i18n.jar	43.9 KB	06/16/2016
apacheds-i18n-2.0.0-M15.jar	43.9 KB	06/16/2016
apacheds-kerberos-codec.jar	675.3 KB	06/16/2016
apacheds-kerberos-codec-2.0.0-M15.jar	675.3 KB	06/16/2016
api-asn1-api.jar	16.2 KB	06/16/2016
api-asn1-api-1.0.0-M20.jar	16.2 KB	06/16/2016
api-util.jar	78.0 KB	06/16/2016
api-util-1.0.0-M20.jar	78.0 KB	06/16/2016
asm.jar	42.4 KB	06/16/2016
asm-3.2.jar	42.4 KB	06/16/2016
avro.jar	465.6 KB	06/16/2016

Step 4: Create a Java Class named "WordCount"



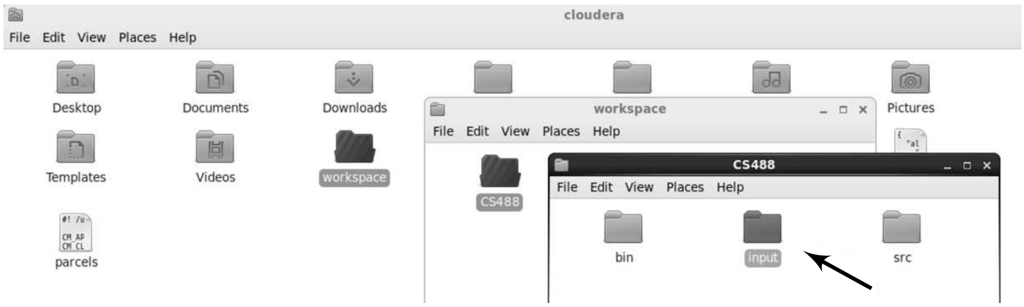
* Then Copy, Paste and save the below code in the "WordCount" class file

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


public class WordCount {

 public static class WordCountMapper
        extends Mapper<LongWritable, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    @Override
    public void map(LongWritable key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }
  public static class WordCountReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context
                    ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, “word count”);

     job.setJarByClass(WordCount.class);

    FileInputFormat.addInputPath(job, new Path(“input”));
    FileOutputFormat.setOutputPath(job, new Path(“output”));

    job.setMapperClass(WordCountMapper.class);
    job.setCombinerClass(WordCountReducer.class);
    job.setReducerClass(WordCountReducer.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

     System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

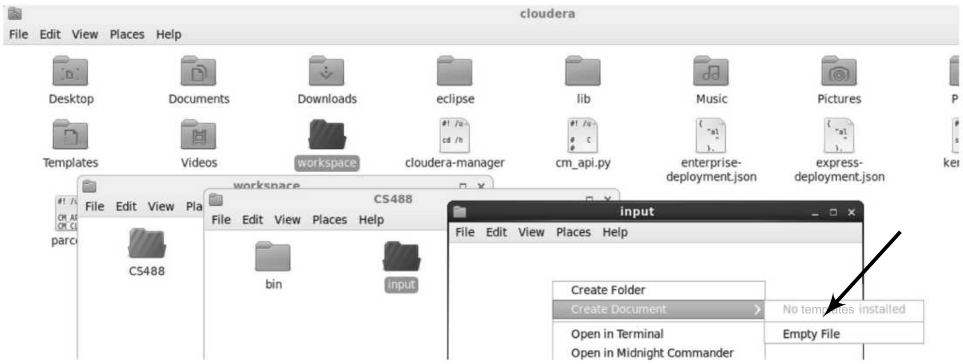

Java - CS488/src/WordCount.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
CS488
src
(default package)
WordCount.java
JRE System Library [JavaSE-1.7]
Referenced Libraries
training
WordCount.java

```
import java.io.IOException;
import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;



public class WordCount {

  public static class WordCountMapper
       extends Mapper<LongWritable, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

```

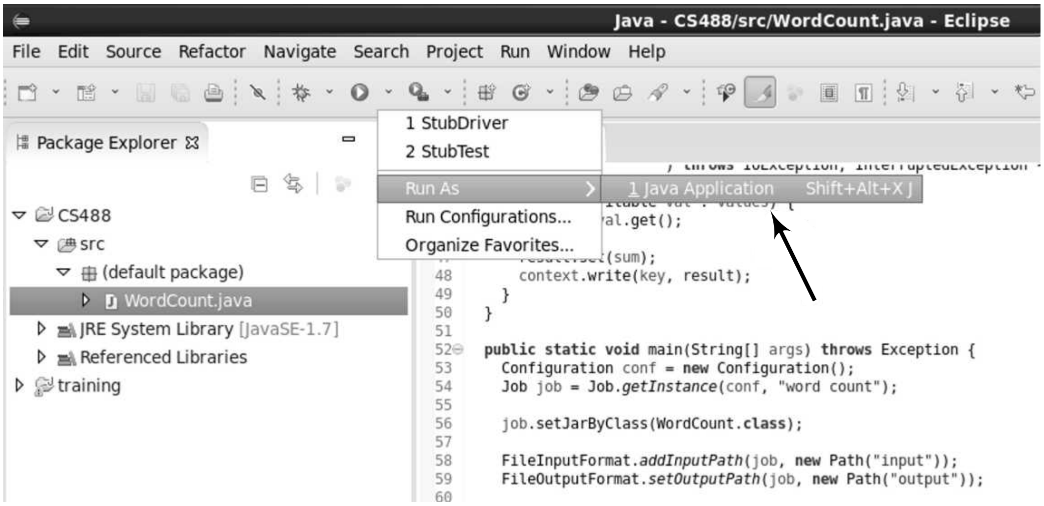
Step 5: Create "input" directory in the workspace->CS488



Step 6: Create a sample file with sample data named "TestWC.txt" in "input" directory

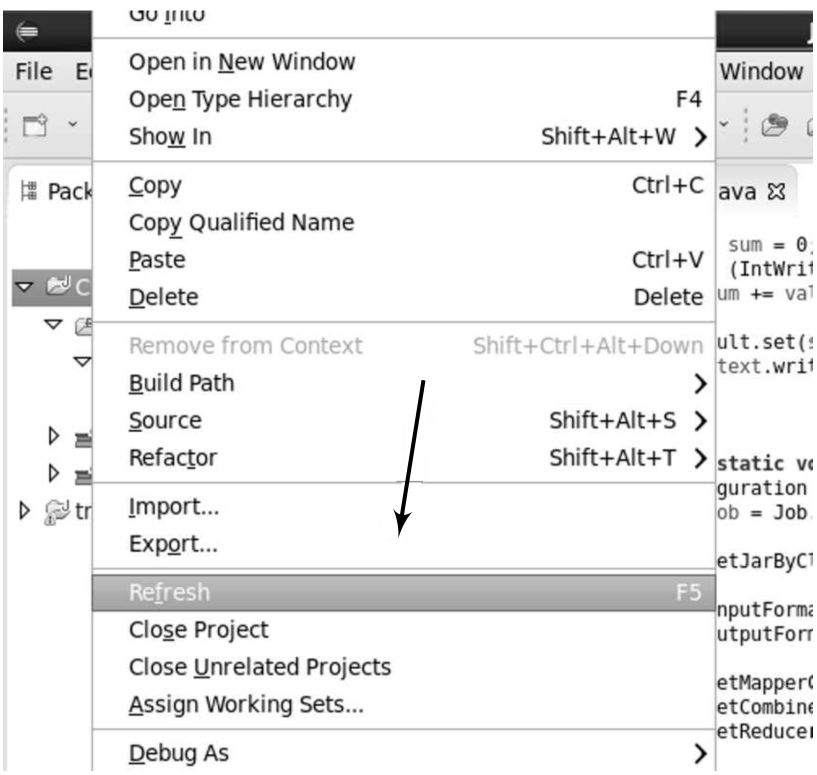


Step 7: Run the Java Application

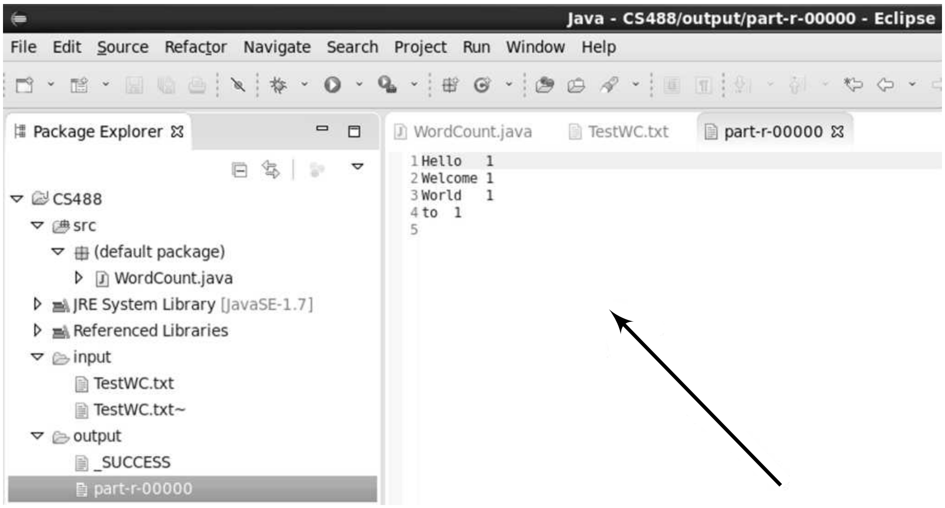

Java - CS488/src/WordCount.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer
CS488
src
(default package)
WordCount.java
JRE System Library [JavaSE-1.7]
Referenced Libraries
training
1 StubDriver
2 StubTest
Run As
1 Java Application Shift+Alt+X J
Run Configurations...
Organize Favorites...

```
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");

    job.setJarByClass(WordCount.class);

    FileInputFormat.addInputPath(job, new Path("input"));
    FileOutputFormat.setOutputPath(job, new Path("output"));
```

Step 8: Refresh the project



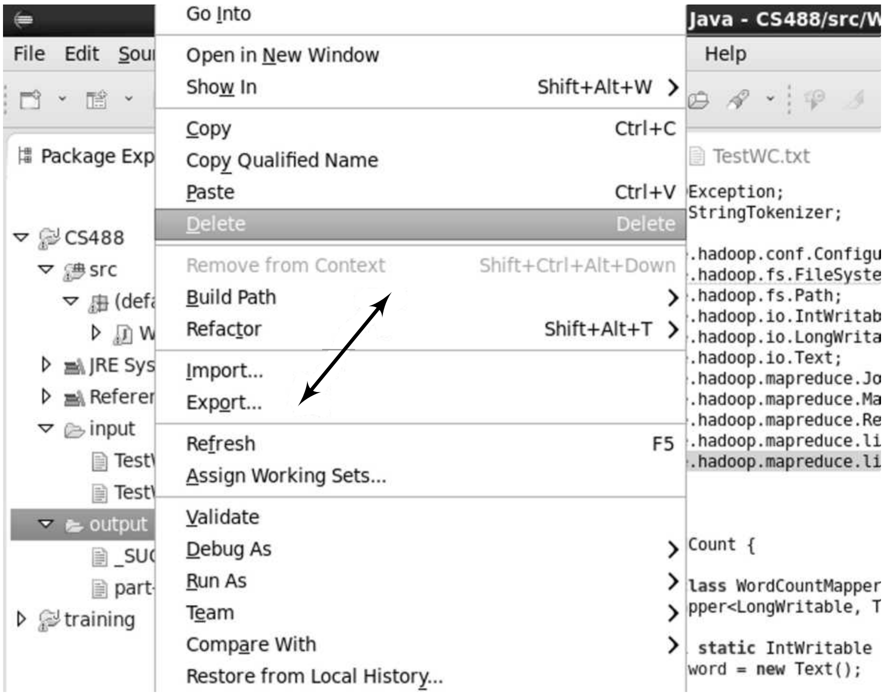
Step 9: Check the input and output



```
Hello   1
Welcome 1
World   1
to  1
```

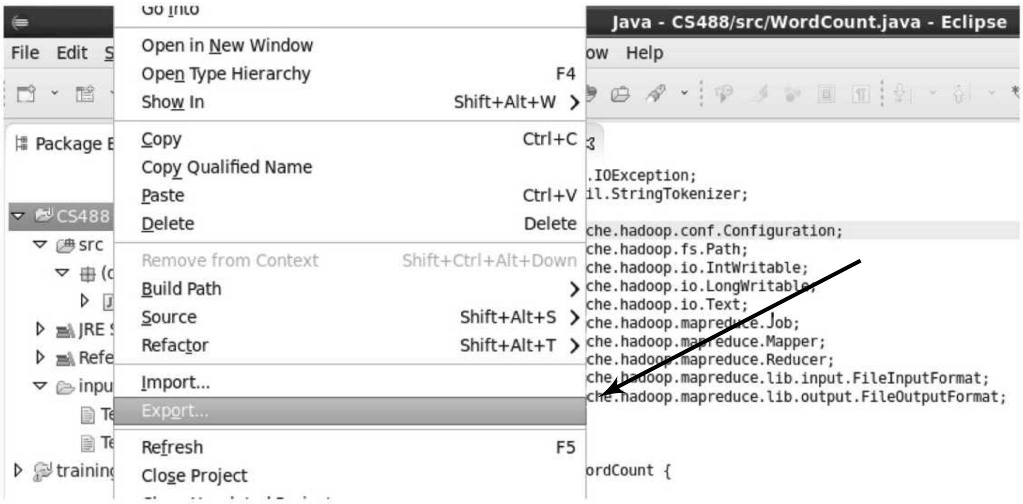
Step 5: Create a Jar File to Run the Application in Hadoop

Step 1: To run the application again Delete the "output" directory



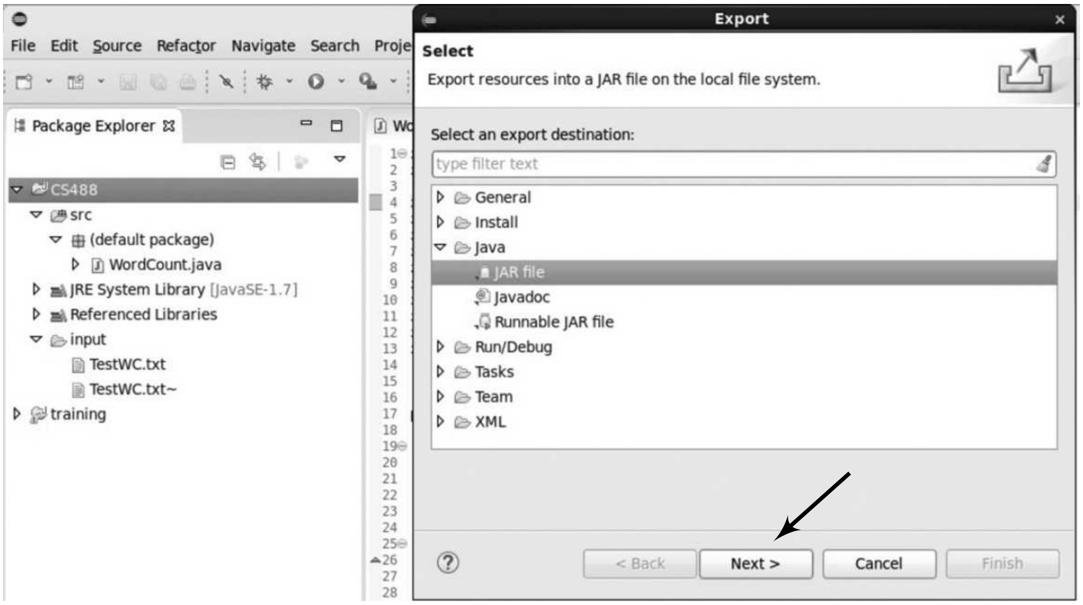
```
Exception;
StringTokenizer;

.hadoop.conf.Configu
.hadoop.fs.FileSyste
.hadoop.fs.Path;
.hadoop.io.IntWritab
.hadoop.io.LongWrita
.hadoop.io.Text;
.hadoop.mapreduce.Jo
.hadoop.mapreduce.Ma
.hadoop.mapreduce.Re
.hadoop.mapreduce.li
.hadoop.mapreduce.li


Count {

lass WordCountMapper
pper<LongWritable, T

 static IntWritable
word = new Text();
```

Step 2: Click on “Export” to create a jar file



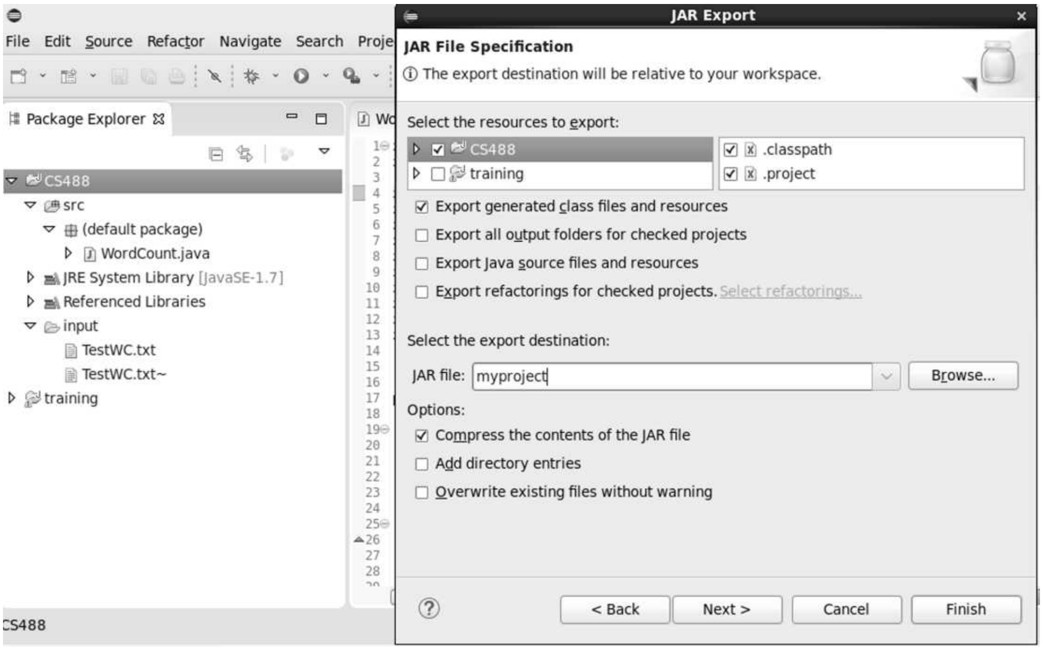
```
.IOException;
il.StringTokenizer;

che.hadoop.conf.Configuration;
che.hadoop.fs.Path;
che.hadoop.io.IntWritable;
che.hadoop.io.LongWritable;
che.hadoop.io.Text;
che.hadoop.mapreduce.Job;
che.hadoop.mapreduce.Mapper;
che.hadoop.mapreduce.Reducer;
che.hadoop.mapreduce.lib.input.FileInputFormat;
che.hadoop.mapreduce.lib.output.FileOutputFormat;


ordCount {
```

Then click Next



Export resources into a JAR file on the local file system.

Then do the next steps to finish the jar file creation



JAR File Specification

The export destination will be relative to your workspace.

Select the resources to export:

- [✓] CS488
- [ ] training
- [✓] .classpath
- [✓] .project

- [✓] Export generated class files and resources
- [ ] Export all output folders for checked projects
- [ ] Export Java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: myproject

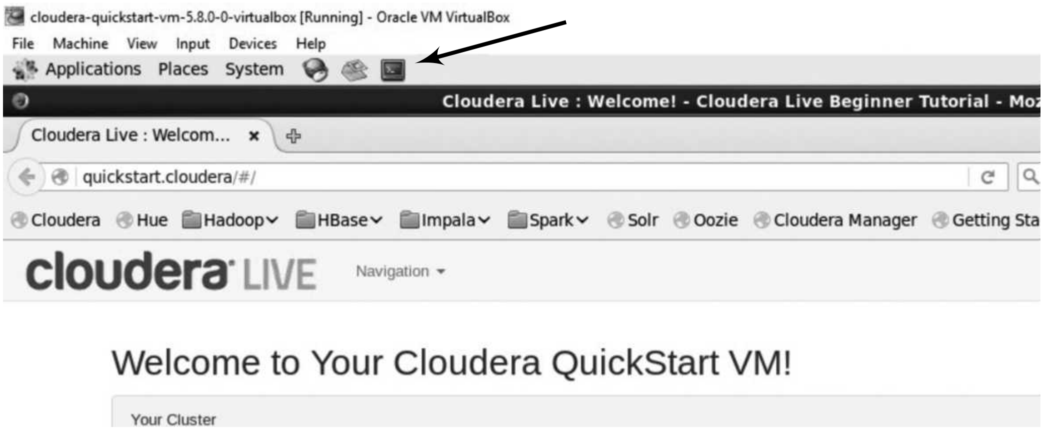
Options:

- [✓] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

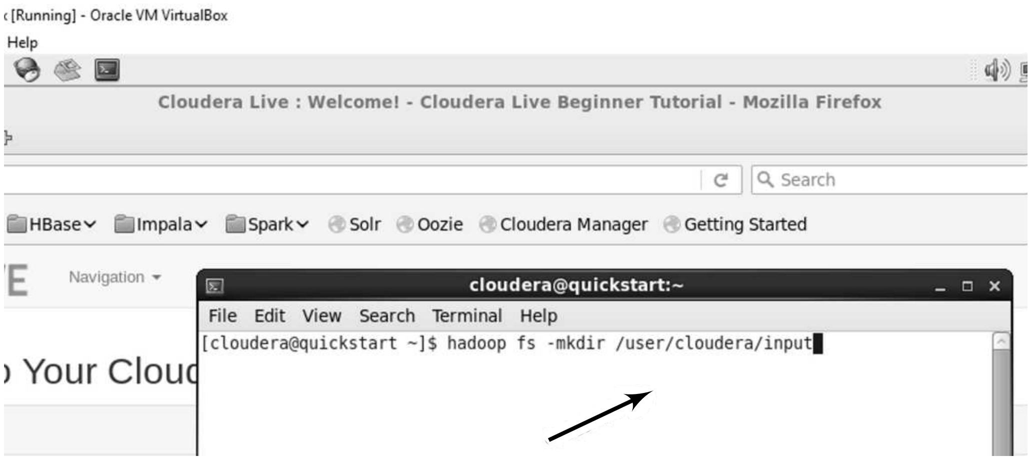
Step 6: Create a Directory and Upload Sample Data File in Hadoop:

Run the commands to create a directory and upload sample data file into Hadoop

Step 1: Click on **Terminal** icon to start

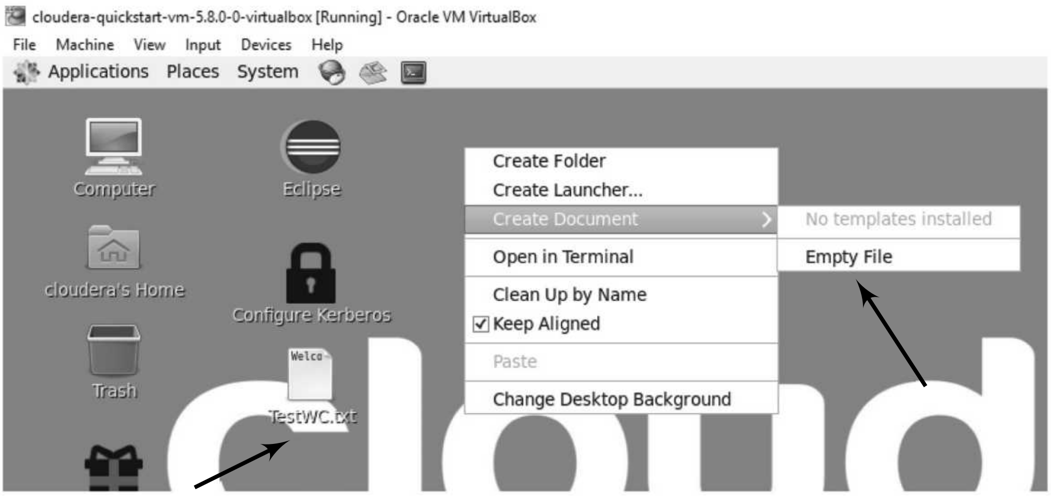

cloudera-quickstart-vm-5.8.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mo
Cloudera Live : Welcom...
quickstart.cloudera/#/
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Sta
cloudera LIVE
Navigation
Welcome to Your Cloudera QuickStart VM!
Your Cluster

Step 2: Run the command "hadoop fs –mkdir /user/cloudera/input" to creatre "input" directory in Hadoop

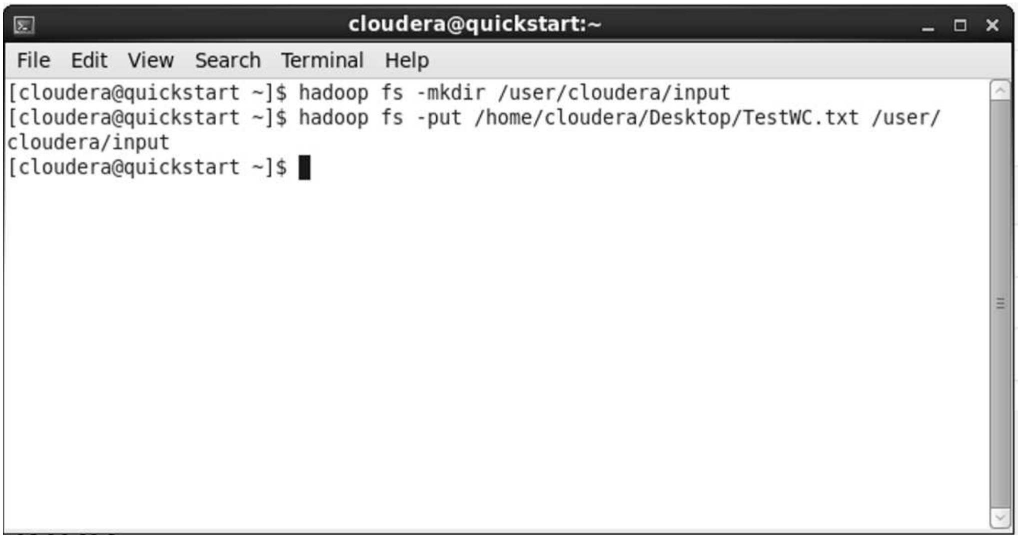

[Running] - Oracle VM VirtualBox
Help
Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mozilla Firefox
Search
HBase
Impala
Spark
Solr
Oozie
Cloudera Manager
Getting Started
Navigation
Your Clou
cloudera@quickstart:~
File Edit View Search Terminal Help

```
[cloudera@quickstart ~]$ hadoop fs -mkdir /user/cloudera/input
```

Step 3: Create the "TestWC.txt" file in Cloudera Desktop



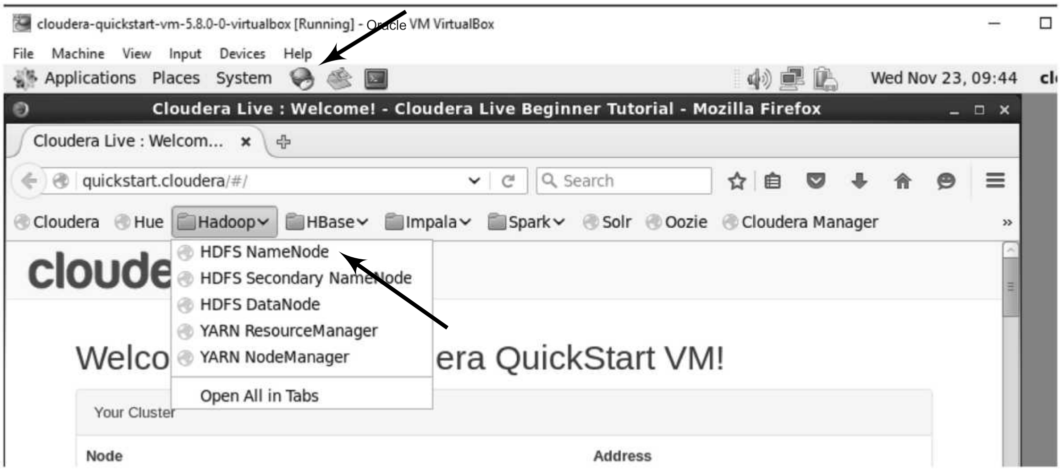
Step 4: Run the command "hadoop fs –put /home/cloudera/Desktop/TestWC.txt /user/cloudera/input" to upload "TestWC.txt" file to Hadoop



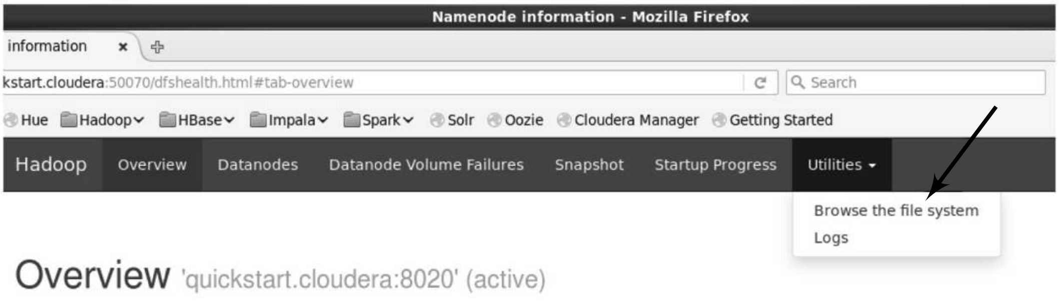
```
[cloudera@quickstart ~]$ hadoop fs -mkdir /user/cloudera/input
[cloudera@quickstart ~]$ hadoop fs -put /home/cloudera/Desktop/TestWC.txt /user/
cloudera/input
[cloudera@quickstart ~]$
```

Check Hadoop file system to see the upload file

Step 1: Click on "HDFS NameNode"

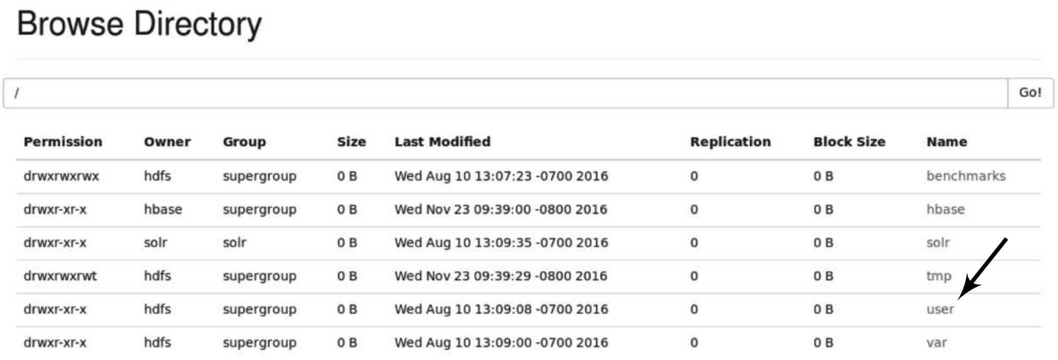

cloudera-quickstart-vm-5.8.0-0-virtualbox [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Wed Nov 23, 09:44
Cloudera Live : Welcome! - Cloudera Live Beginner Tutorial - Mozilla Firefox
Cloudera Live : Welcom...
quickstart.cloudera/#/
Search
Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager
HDFS NameNode
HDFS Secondary NameNode
HDFS DataNode
YARN ResourceManager
YARN NodeManager
Open All in Tabs
Welco era QuickStart VM!
Your Cluster
Node
Address

Step 2: Click on "Browse the file system"



Overview 'quickstart.cloudera:8020' (active)

Step 3: Click on "user"

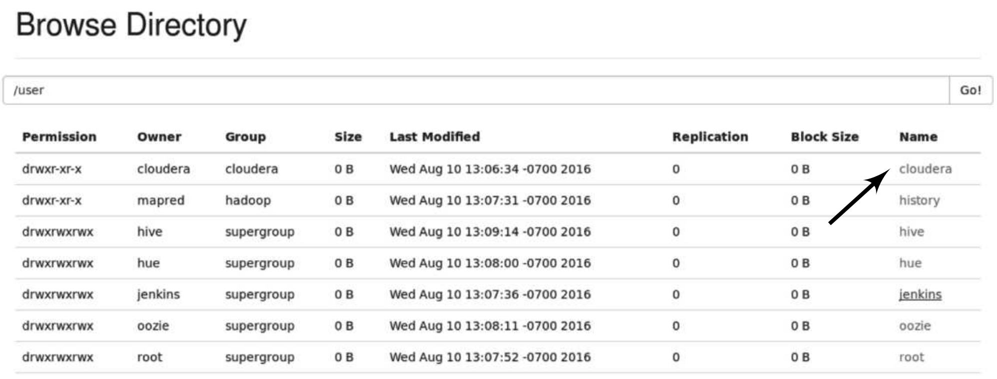


Browse Directory

/ Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxrwxrwx	hdfs	supergroup	0 B	Wed Aug 10 13:07:23 -0700 2016	0	0 B	benchmarks
drwxr-xr-x	hbase	supergroup	0 B	Wed Nov 23 09:39:00 -0800 2016	0	0 B	hbase
drwxr-xr-x	solr	solr	0 B	Wed Aug 10 13:09:35 -0700 2016	0	0 B	solr
drwxrwxrwt	hdfs	supergroup	0 B	Wed Nov 23 09:39:29 -0800 2016	0	0 B	tmp
drwxr-xr-x	hdfs	supergroup	0 B	Wed Aug 10 13:09:08 -0700 2016	0	0 B	user
drwxr-xr-x	hdfs	supergroup	0 B	Wed Aug 10 13:09:00 -0700 2016	0	0 B	var

Step 4: Click on "cloudera"

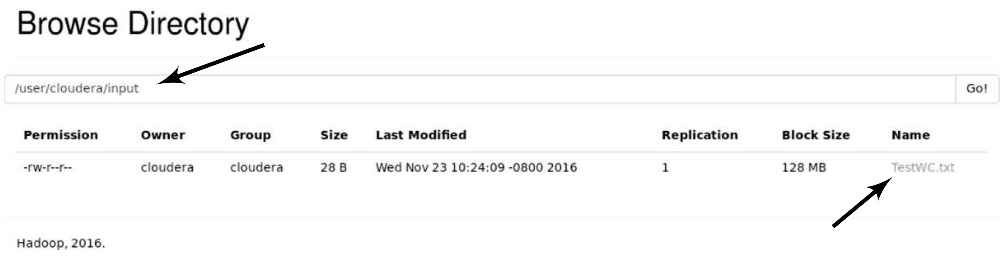


Browse Directory

/user Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	cloudera	cloudera	0 B	Wed Aug 10 13:06:34 -0700 2016	0	0 B	cloudera
drwxr-xr-x	mapred	hadoop	0 B	Wed Aug 10 13:07:31 -0700 2016	0	0 B	history
drwxrwxrwx	hive	supergroup	0 B	Wed Aug 10 13:09:14 -0700 2016	0	0 B	hive
drwxrwxrwx	hue	supergroup	0 B	Wed Aug 10 13:08:00 -0700 2016	0	0 B	hue
drwxrwxrwx	jenkins	supergroup	0 B	Wed Aug 10 13:07:36 -0700 2016	0	0 B	jenkins
drwxrwxrwx	oozie	supergroup	0 B	Wed Aug 10 13:08:11 -0700 2016	0	0 B	oozie
drwxrwxrwx	root	supergroup	0 B	Wed Aug 10 13:07:52 -0700 2016	0	0 B	root

Step 5: We can see the our uploaded "Test.txt" file in "input" directory of Hadoop



Browse Directory

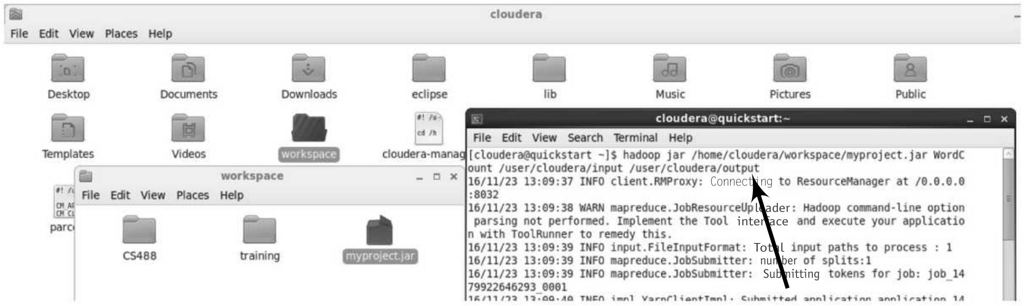
/user/cloudera/input

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	cloudera	cloudera	28 B	Wed Nov 23 10:24:09 -0800 2016	1	128 MB	TestWC.txt

Hadoop, 2016.

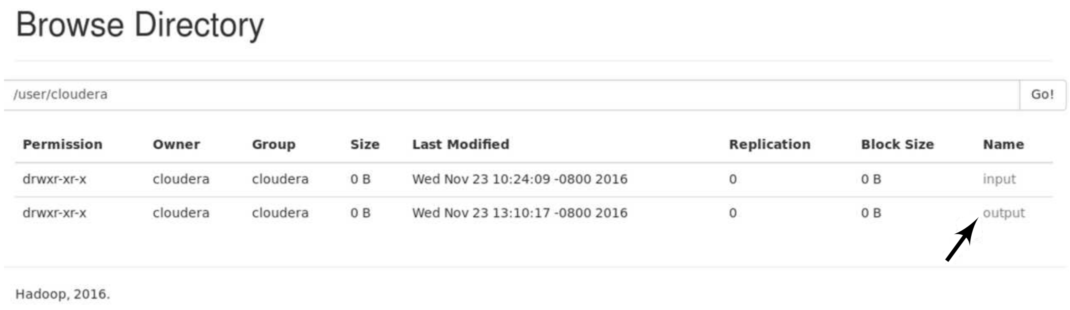
Step 7: Run the Application in Hadoop

Step 1: Run the command "hadoop jar /home/cloudera/workspace/myproject.jar Word-Count /user/cloudera/input /user/cloudera/output" to rut the application in Hadoop



```
[cloudera@quickstart ~]$ hadoop jar /home/cloudera/workspace/myproject.jar WordC
ount /user/cloudera/input /user/cloudera/output
16/11/23 13:09:37 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0
:8032
16/11/23 13:09:38 WARN mapreduce.JobResourceUploader: Hadoop command-line option
 parsing not performed. Implement the Tool interface and execute your applicatio
n with ToolRunner to remedy this.
16/11/23 13:09:39 INFO input.FileInputFormat: Total input paths to process : 1
16/11/23 13:09:39 INFO mapreduce.JobSubmitter: number of splits:1
16/11/23 13:09:39 INFO mapreduce.JobSubmitter:  Submitting tokens for job: job_14
79922646293_0001
16/11/23 13:09:40 INFO impl.YarnClientImpl: Submitted application application_14
```

Step 2: Check the output in the Hadoop in browser



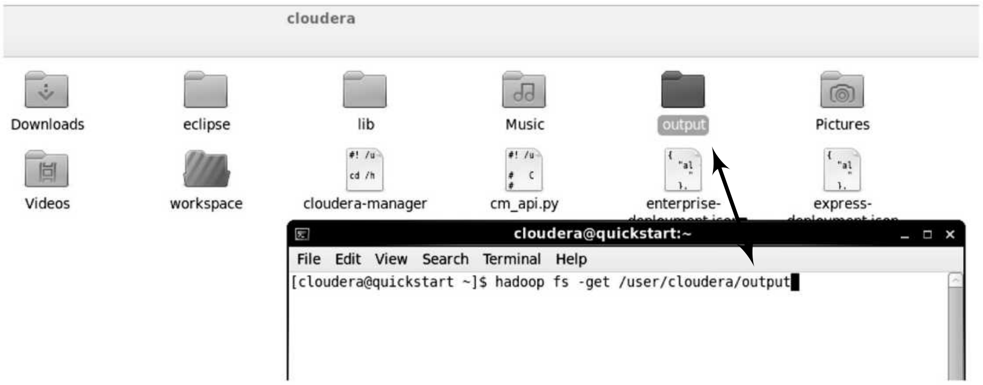
Browse Directory

/user/cloudera

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	cloudera	cloudera	0 B	Wed Nov 23 10:24:09 -0800 2016	0	0 B	input
drwxr-xr-x	cloudera	cloudera	0 B	Wed Nov 23 13:10:17 -0800 2016	0	0 B	output

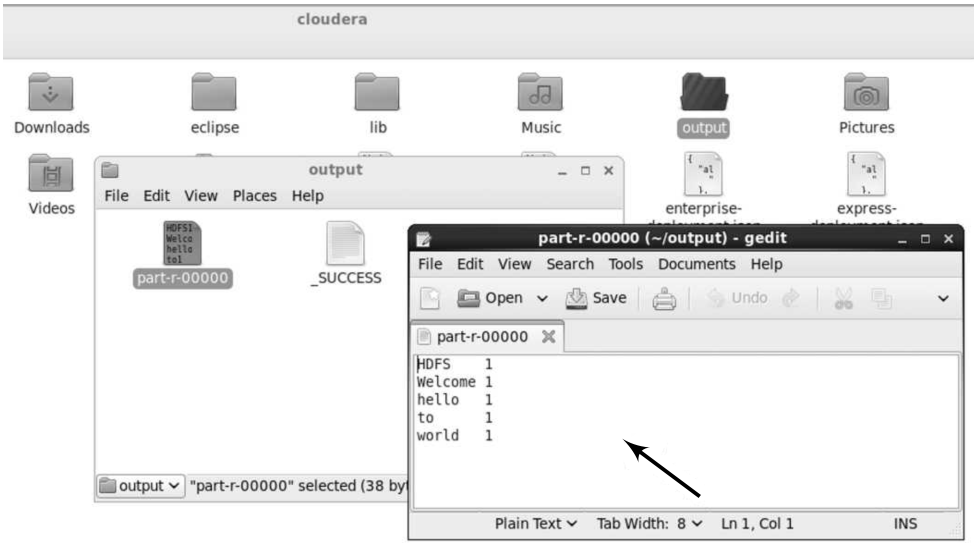
Hadoop, 2016.

Step 3: Run the command "hadoop fs –get /user/cloudera/output" to download the "output" directory with output file from Hadoop



```
[cloudera@quickstart ~]$ hadoop fs -get /user/cloudera/output
```

Step 4: We can see the output



```
HDFS    1
Welcome 1
hello   1
to      1
world   1
```

Note:

1. VirtualBox is a separate virtual machine, so to get a file in VirtualBox from Windows, one can do it through email or FTP or TeamViewer.
2. From Windows, login to email through browser then attach the input files and send it to youself.
3. In VirtualBox, login to email through browser then download the attached file in Cloudera to use.

Appendix 2

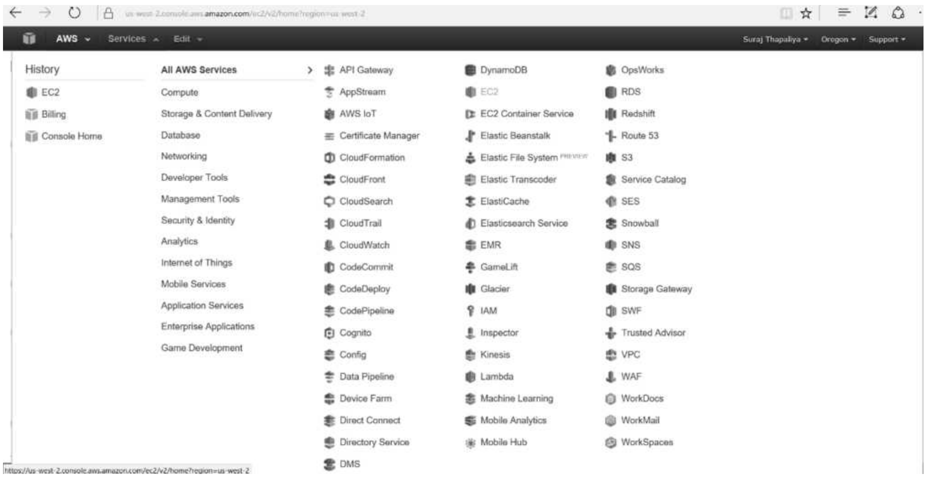
Installing Hadoop on Amazon Web Services (AWS) Elastic Compute Cluster (EC2)

A2.1 CREATING CLUSTER SERVER ON AWS, INSTALL HADOOP FROM CLOUDERA

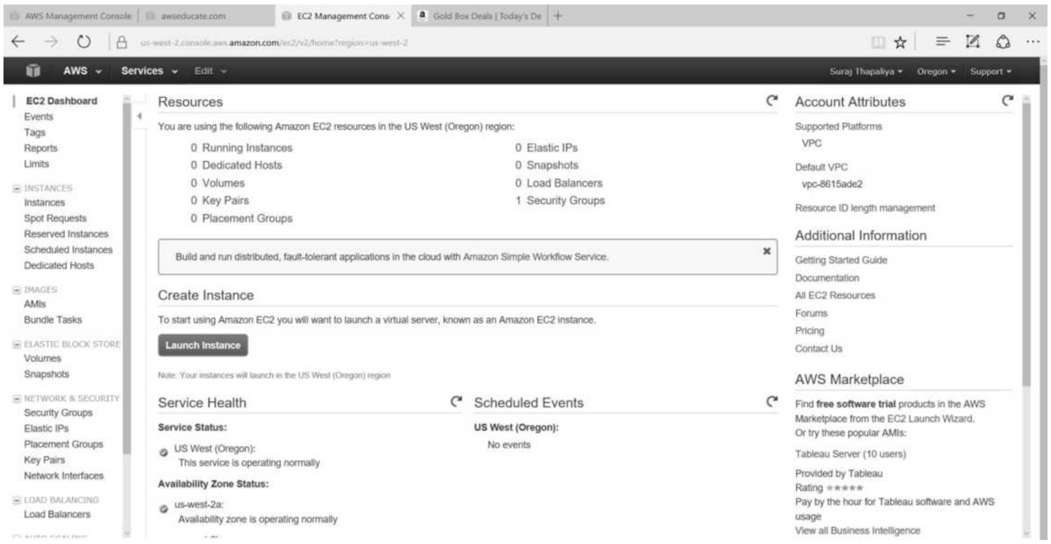
The objective of this tutorial is to set up a big data processing infrastructure using cloud computing, and Hadoop and Spark software.

Step 1: Creating Amazon EC2 Servers.

1. Open https://aws.amazon.com/
2. Click on Services
3. Click on EC2



You can see the below result once you click on EC2. If you already have a server you can see the number of running servers, their volume and other information.



AWS Services Edit — Suraj Thapaliya Oregon Support

EC2 Dashboard, Events, Tags, Reports, Limits
INSTANCES: Instances, Spot Requests, Reserved Instances, Scheduled Instances, Dedicated Hosts
IMAGES: AMIs, Bundle Tasks
ELASTIC BLOCK STORE: Volumes, Snapshots
NETWORK & SECURITY: Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces
LOAD BALANCING: Load Balancers

Resources

You are using the following Amazon EC2 resources in the US West (Oregon) region:

0 Running Instances	0 Elastic IPs
0 Dedicated Hosts	0 Snapshots
0 Volumes	0 Load Balancers
0 Key Pairs	1 Security Groups
0 Placement Groups	

Build and run distributed, fault-tolerant applications in the cloud with Amazon Simple Workflow Service.

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Launch Instance

Note: Your instances will launch in the US West (Oregon) region

Service Health

Service Status:
US West (Oregon):
This service is operating normally

Availability Zone Status:
us-west-2a:
Availability zone is operating normally

Scheduled Events

US West (Oregon):
No events

Account Attributes

Supported Platforms
VPC

Default VPC
vpc-8615ade2

Resource ID length management

Additional Information

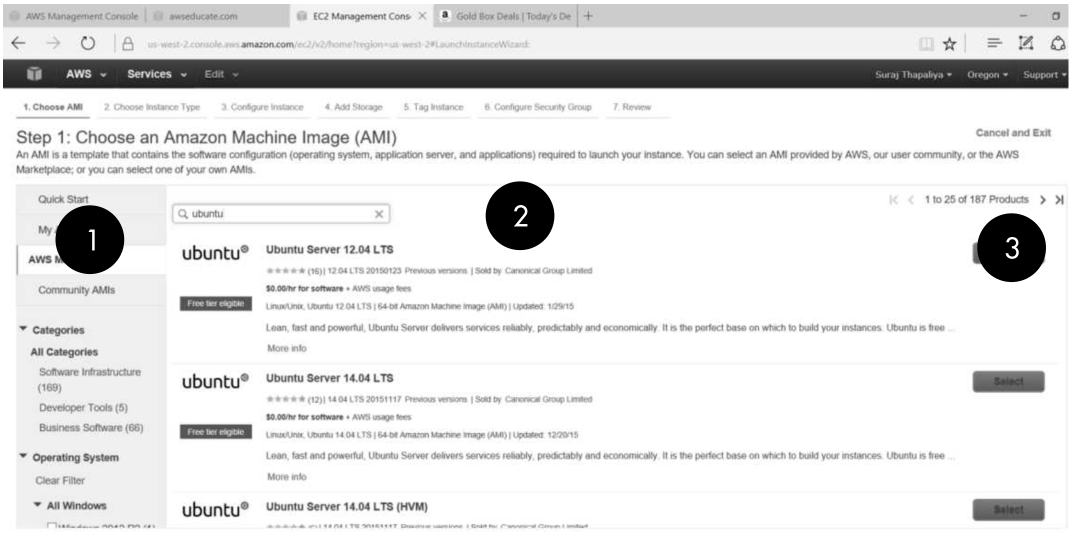
Getting Started Guide
Documentation
All EC2 Resources
Forums
Pricing
Contact Us

AWS Marketplace

Find **free software trial** products in the AWS Marketplace from the EC2 Launch Wizard.
Or try these popular AMIs:

Tableau Server (10 users)
Provided by Tableau
Rating ★★★★★
Pay by the hour for Tableau software and AWS usage
View all Business Intelligence

4. Click on Launch Instance Button. Launch Instance



1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI)

Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Quick Start, My AMIs, AWS Marketplace, Community AMIs

Categories: All Categories, Software Infrastructure (169), Developer Tools (5), Business Software (66)

Operating System: Clear Filter, All Windows

ubuntu — 1 to 25 of 187 Products

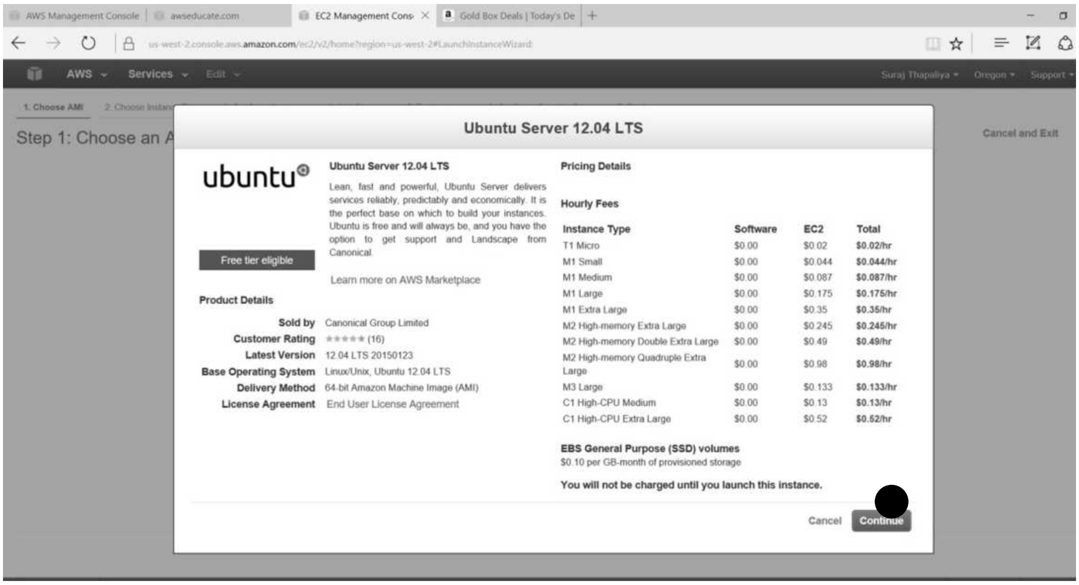
Ubuntu Server 12.04 LTS
★★★★★ (16) | 12.04 LTS 20150123 Previous versions | Sold by Canonical Group Limited
$0.00/hr for software + AWS usage fees
Linux/Unix, Ubuntu 12.04 LTS | 64-bit Amazon Machine Image (AMI) | Updated: 1/29/15
Lean, fast and powerful, Ubuntu Server delivers services reliably, predictably and economically. It is the perfect base on which to build your instances. Ubuntu is free ...
More info
Free tier eligible

Ubuntu Server 14.04 LTS
★★★★★ (12) | 14.04 LTS 20151117 Previous versions | Sold by Canonical Group Limited
$0.00/hr for software + AWS usage fees
Linux/Unix, Ubuntu 14.04 LTS | 64-bit Amazon Machine Image (AMI) | Updated: 12/20/15
Lean, fast and powerful, Ubuntu Server delivers services reliably, predictably and economically. It is the perfect base on which to build your instances. Ubuntu is free ...
More info
Free tier eligible
Select

Ubuntu Server 14.04 LTS (HVM)
Select

5. Click on **AWS MarketePlace**
6. Type Ubuntu in **search** text box.

7. Click on **Select** button



Ubuntu Server 12.04 LTS

ubuntu

Ubuntu Server 12.04 LTS

Lean, fast and powerful, Ubuntu Server delivers services reliably, predictably and economically. It is the perfect base on which to build your instances. Ubuntu is free and will always be, and you have the option to get support and Landscape from Canonical.

Free tier eligible

Learn more on AWS Marketplace

Product Details

Sold by	Canonical Group Limited
Customer Rating	★★★★★ (16)
Latest Version	12.04 LTS 20150123
Base Operating System	Linux/Unix, Ubuntu 12.04 LTS
Delivery Method	64-bit Amazon Machine Image (AMI)
License Agreement	End User License Agreement

Pricing Details

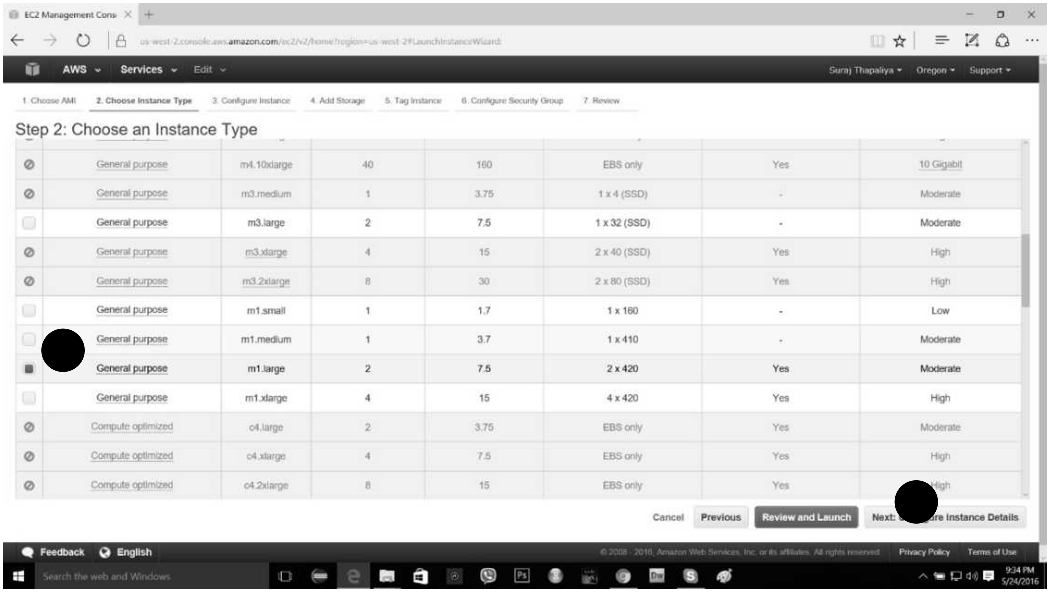
Hourly Fees

Instance Type	Software	EC2	Total
T1 Micro	$0.00	$0.02	**$0.02/hr**
M1 Small	$0.00	$0.044	**$0.044/hr**
M1 Medium	$0.00	$0.087	**$0.087/hr**
M1 Large	$0.00	$0.175	**$0.175/hr**
M1 Extra Large	$0.00	$0.35	**$0.35/hr**
M2 High-memory Extra Large	$0.00	$0.245	**$0.245/hr**
M2 High-memory Double Extra Large	$0.00	$0.49	**$0.49/hr**
M2 High-memory Quadruple Extra Large	$0.00	$0.98	**$0.98/hr**
M3 Large	$0.00	$0.133	**$0.133/hr**
C1 High-CPU Medium	$0.00	$0.13	**$0.13/hr**
C1 High-CPU Extra Large	$0.00	$0.52	**$0.52/hr**

EBS General Purpose (SSD) volumes
$0.10 per GB-month of provisioned storage

You will not be charged until you launch this instance.

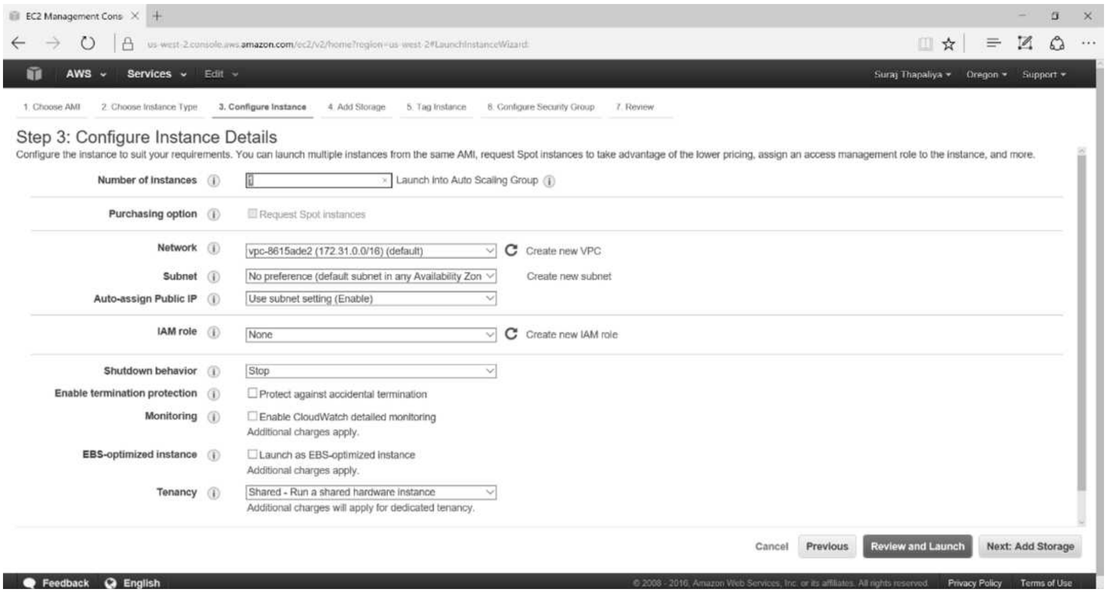
8. Ubuntu is free so you don't have to worry about the service price Click on **Continue** button.



Step 2: Choose an Instance Type

⊘	General purpose	m4.10xlarge	40	160	EBS only	Yes	10 Gigabit
⊘	General purpose	m3.medium	1	3.75	1 x 4 (SSD)	-	Moderate
☐	General purpose	m3.large	2	7.5	1 x 32 (SSD)	-	Moderate
⊘	General purpose	m3.xlarge	4	15	2 x 40 (SSD)	Yes	High
⊘	General purpose	m3.2xlarge	8	30	2 x 80 (SSD)	Yes	High
☐	General purpose	m1.small	1	1.7	1 x 160	-	Low
☐	General purpose	m1.medium	1	3.7	1 x 410	-	Moderate
■	General purpose	m1.large	2	7.5	2 x 420	Yes	Moderate
☐	General purpose	m1.xlarge	4	15	4 x 420	Yes	High
⊘	Compute optimized	c4.large	2	3.75	EBS only	Yes	Moderate
⊘	Compute optimized	c4.xlarge	4	7.5	EBS only	Yes	High
⊘	Compute optimized	c4.2xlarge	8	15	EBS only	Yes	High

9. Choose **General.purpose m1.large** and click on **Next:Configurare Instance Details** (Do not choose the **Micro Instances t1.micro** it is free but it will not able to handle the installation.)

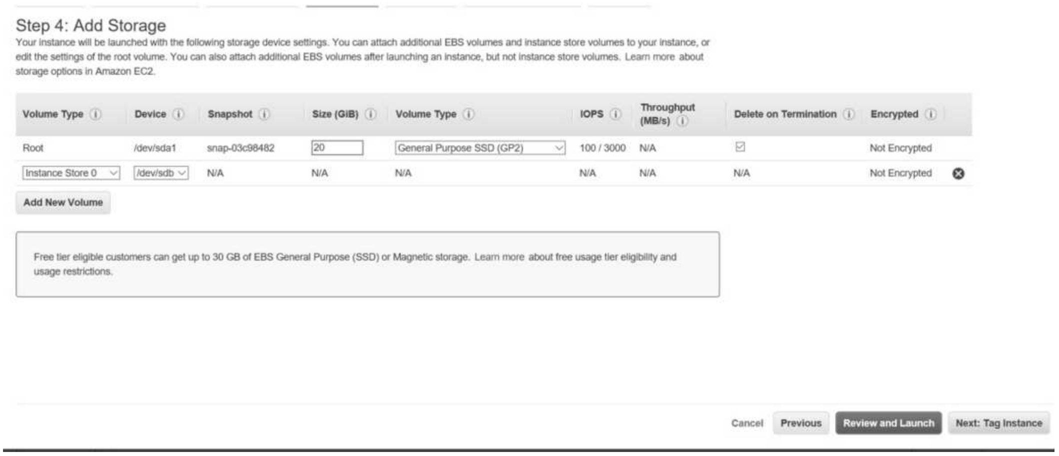


Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1 Launch into Auto Scaling Group
Purchasing option	☐ Request Spot instances
Network	vpc-8615ade2 (172.31.0.0/16) (default) Create new VPC
Subnet	No preference (default subnet in any Availability Zon Create new subnet
Auto-assign Public IP	Use subnet setting (Enable)
IAM role	None Create new IAM role
Shutdown behavior	Stop
Enable termination protection	☐ Protect against accidental termination
Monitoring	☐ Enable CloudWatch detailed monitoring Additional charges apply.
EBS-optimized instance	☐ Launch as EBS-optimized instance Additional charges apply.
Tenancy	Shared - Run a shared hardware instance Additional charges will apply for dedicated tenancy.

10. Click on **Next: Add Storage**



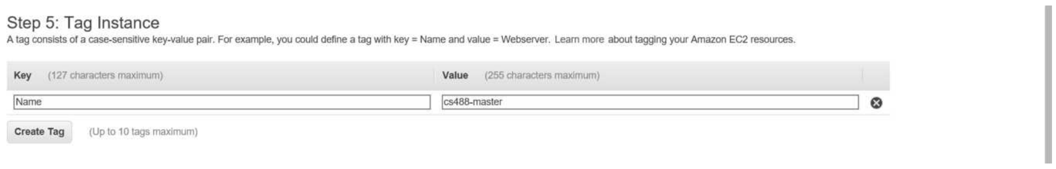
Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. Learn more about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted
Root	/dev/sda1	snap-03c98482	20	General Purpose SSD (GP2)	100 / 3000	N/A	✓	Not Encrypted
Instance Store 0	/dev/sdb	N/A	N/A	N/A	N/A	N/A	N/A	Not Encrypted

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

11. Specify the volume size 20GB (Default will be 8 but it will not sufficient) and Click on **Next: Tag Instance**



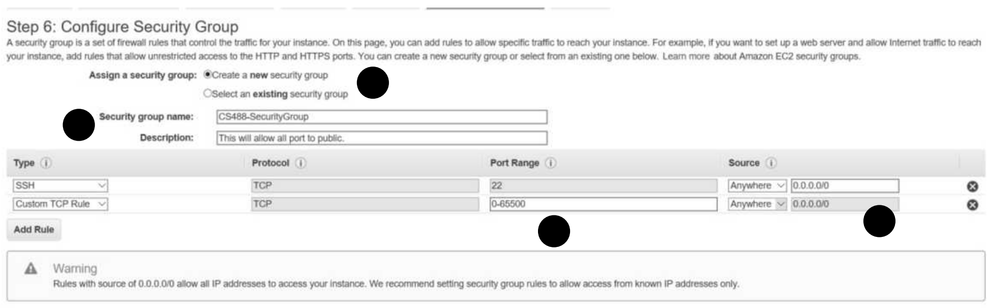
Step 5: Tag Instance

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Learn more about tagging your Amazon EC2 resources.

Key (127 characters maximum)	Value (255 characters maximum)
Name	cs488-master

Create Tag (Up to 10 tags maximum)

12. Type the name cs488-master (This is for label to know which one is master and slave) and click on Next: Security Group



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: [✓ Create a **new** security group] Select an **existing** security group

Security group name: CS488-SecurityGroup

Description: This will allow all port to public.

Type	Protocol	Port Range	Source	
SSH	TCP	22	Anywhere	0.0.0.0/0
Custom TCP Rule	TCP	0-65500	Anywhere	0.0.0.0/0

Add Rule

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

13. We need to open our server to the world including most of the port cause cloudera need to add more port.

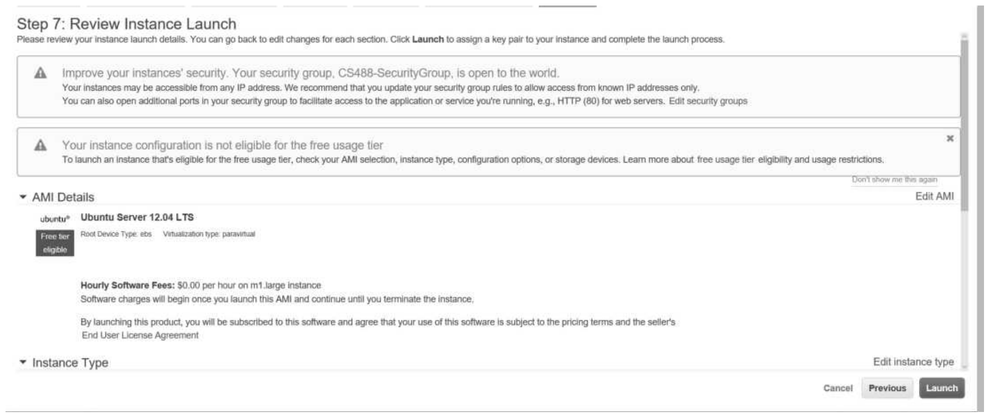
Specify the group name

Type: Choose Custom TCP Rule

Port Range 0-65500

Source : AnyWhere

And Click on Review Instance



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Improve your instances' security. Your security group, CS488-SecurityGroup, is open to the world.
Your instances may be accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only.
You can also open additional ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. Edit security groups

Your instance configuration is not eligible for the free usage tier
To launch an instance that's eligible for the free usage tier, check your AMI selection, instance type, configuration options, or storage devices. Learn more about free usage tier eligibility and usage restrictions.

Don't show me this again

AMI Details — Edit AMI

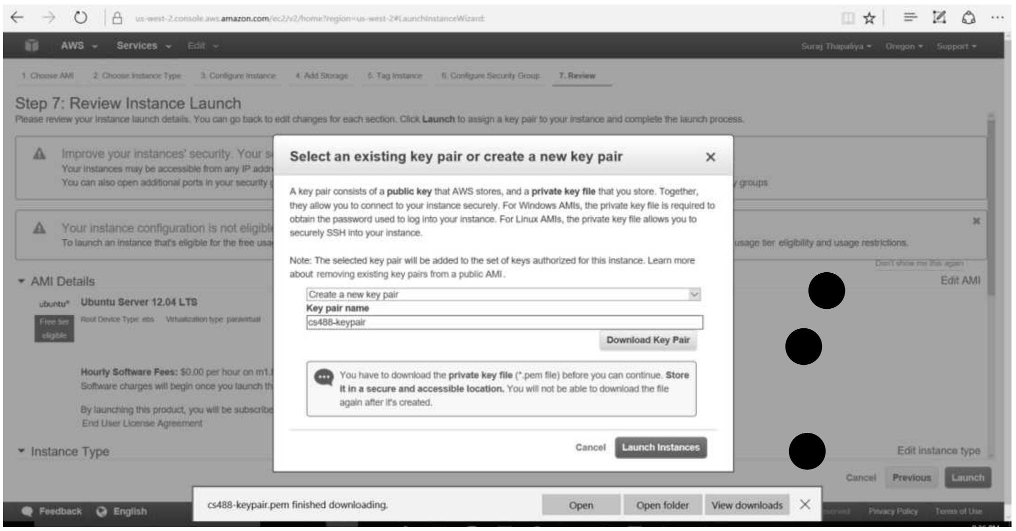
Ubuntu Server 12.04 LTS
Root Device Type: ebs Virtualization type: paravirtual

Hourly Software Fees: $0.00 per hour on m1.large instance
Software charges will begin once you launch this AMI and continue until you terminate the instance.

By launching this product, you will be subscribed to this software and agree that your use of this software is subject to the pricing terms and the seller's End User License Agreement

Instance Type — Edit instance type

14. The message shows the warning this is only that we open our server to world, So ignore it for now. Click on **Launch** button.



Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

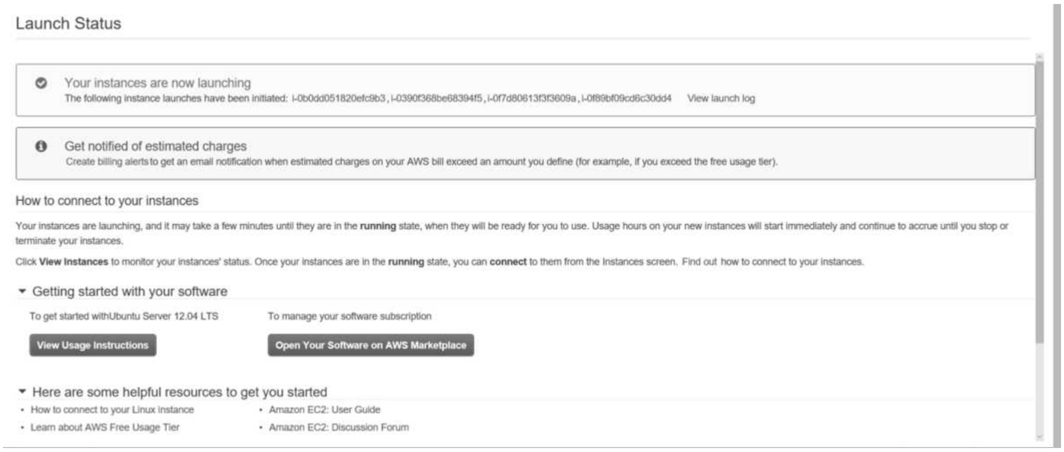
Key pair name

cs488-keypair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

cs488-keypair.pem finished downloading.

15. Type the key pair name and Click on Download Key Pair button (remember the location of downloaded file we need this file to log in to the server.) and Click on Launch Instances.



Launch Status

Your instances are now launching

The following instance launches have been initiated: i-0b0dd051820efc9b3 , i-0390f368be68394f5 , i-0f7d80613f3f3609a , i-0f89bf09cd6c30dd4 View launch log

Get notified of estimated charges

Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instances screen. Find out how to connect to your instances.

Getting started with your software

To get started withUbuntu Server 12.04 LTS

To manage your software subscription

Here are some helpful resources to get you started

- How to connect to your Linux instance
- Learn about AWS Free Usage Tier
- Amazon EC2: User Guide
- Amazon EC2: Discussion Forum

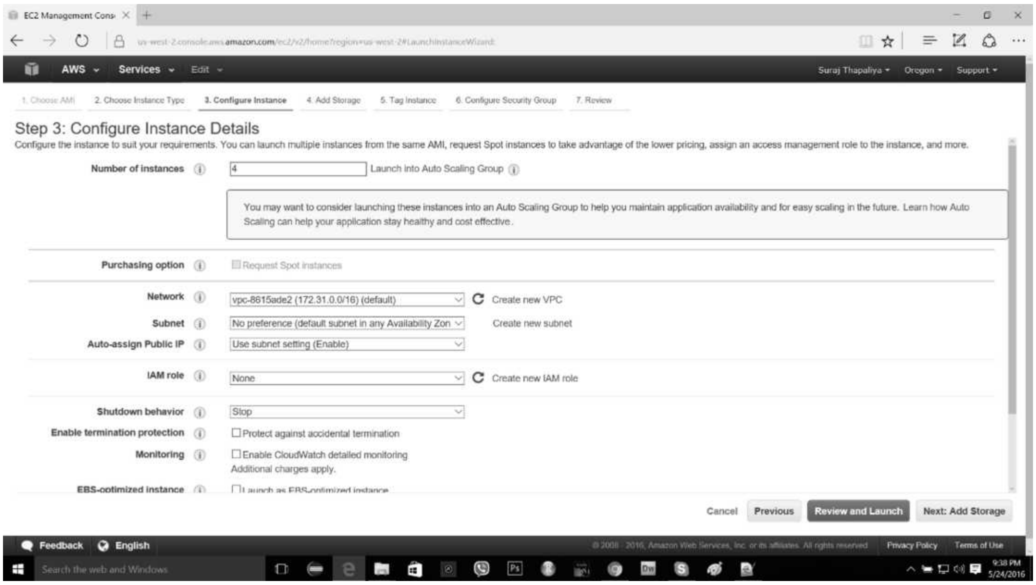
16. Now the master server is created.

    Now, we need four more servers to make the clustering for that we don't need to do these process four times. We just increase the value of no of instance we need and we got the 4 servers.

Now we are going to launch 4 more server which is slaves.

Please repeat step 4-9

Go to amazon market place, choose Ubuntu, select the instance type (General. purpose)



1. Choose AMI | 2. Choose Instance Type | 3. Configure Instance | 4. Add Storage | 5. Tag Instance | 6. Configure Security Group | 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances: 4 — Launch into Auto Scaling Group

You may want to consider launching these instances into an Auto Scaling Group to help you maintain application availability and for easy scaling in the future. Learn how Auto Scaling can help your application stay healthy and cost effective.

Purchasing option: ☐ Request Spot instances

Network: vpc-8615ade2 (172.31.0.0/16) (default) — Create new VPC

Subnet: No preference (default subnet in any Availability Zon — Create new subnet

Auto-assign Public IP: Use subnet setting (Enable)

IAM role: None — Create new IAM role

Shutdown behavior: Stop

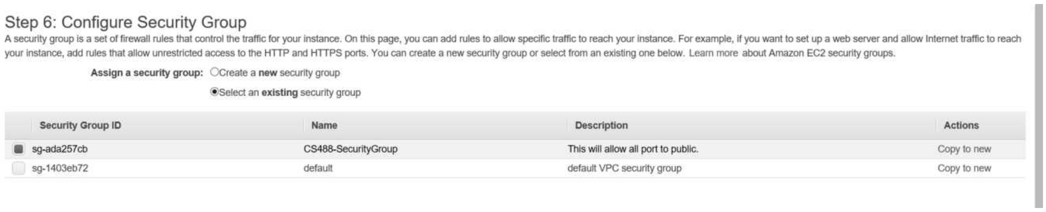
Enable termination protection: ☐ Protect against accidental termination

Monitoring: ☐ Enable CloudWatch detailed monitoring
Additional charges apply.

EBS-optimized instance: ☐ Launch as EBS-optimized instance

Cancel | Previous | Review and Launch | Next: Add Storage

17. Type 4 in Number of Instances. Which will create the 4 more server for us.



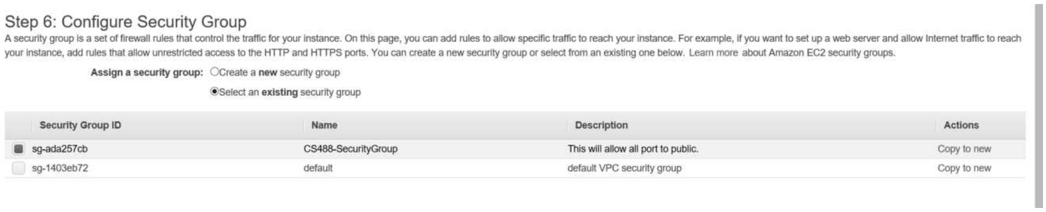
Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: ○ Create a **new** security group
● Select an **existing** security group

	Security Group ID	Name	Description	Actions
■	sg-ada257cb	CS488-SecurityGroup	This will allow all port to public.	Copy to new
☐	sg-1403eb72	default	default VPC security group	Copy to new

18. Name the server cs488-slave



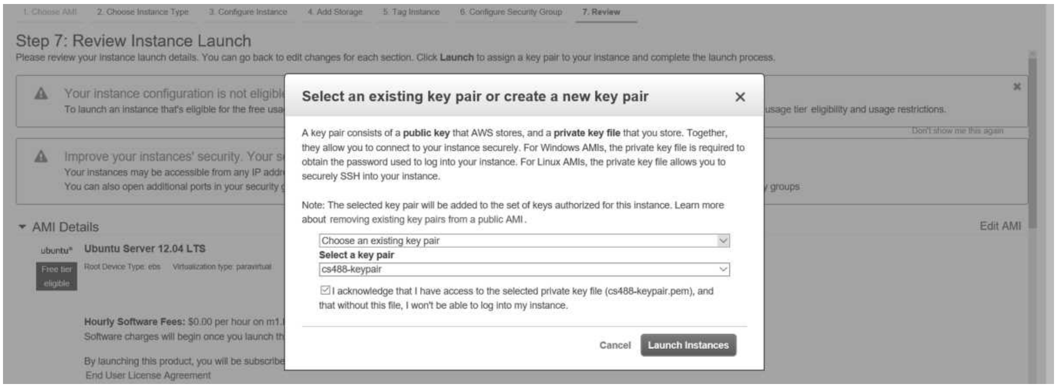
Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: ○ Create a **new** security group
● Select an **existing** security group

	Security Group ID	Name	Description	Actions
■	sg-ada257cb	CS488-SecurityGroup	This will allow all port to public.	Copy to new
☐	sg-1403eb72	default	default VPC security group	Copy to new

19. Select the previous created security group.



1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Tag Instance 6. Configure Security Group 7. Review

Step 7: Review Instance Launch

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Choose an existing key pair

Select a key pair

cs488-keypair

☑ I acknowledge that I have access to the selected private key file (cs488-keypair.pem), and that without this file, I won't be able to log into my instance.

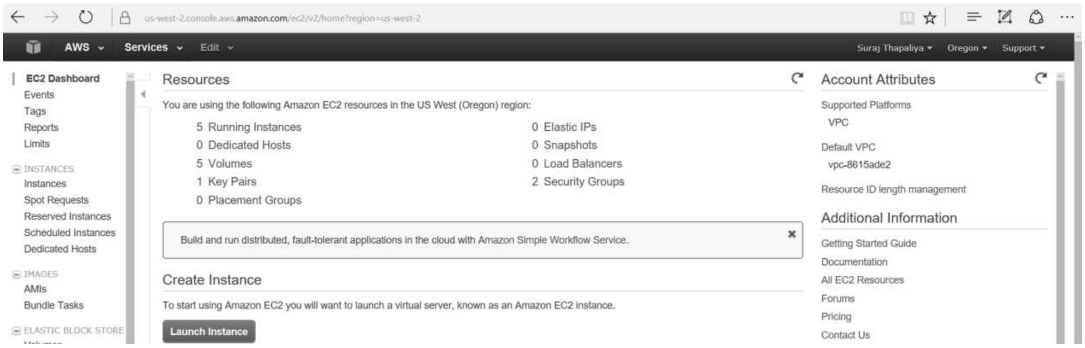
AMI Details

Ubuntu Server 12.04 LTS

Root Device Type: ebs Virtualization type: paravirtual

Hourly Software Fees: $0.00 per hour on m1.

20. It is important that you need to choose the existing key pair for these server too.



us-west-2.console.aws.amazon.com/ec2/v2/home?region=us-west-2

Resources

You are using the following Amazon EC2 resources in the US West (Oregon) region:

```
5 Running Instances        0 Elastic IPs
0 Dedicated Hosts          0 Snapshots
5 Volumes                  0 Load Balancers
1 Key Pairs                2 Security Groups
0 Placement Groups
```

Build and run distributed, fault-tolerant applications in the cloud with Amazon Simple Workflow Service.

Create Instance

To start using Amazon EC2 you will want to launch a virtual server, known as an Amazon EC2 instance.

Account Attributes

Supported Platforms: VPC

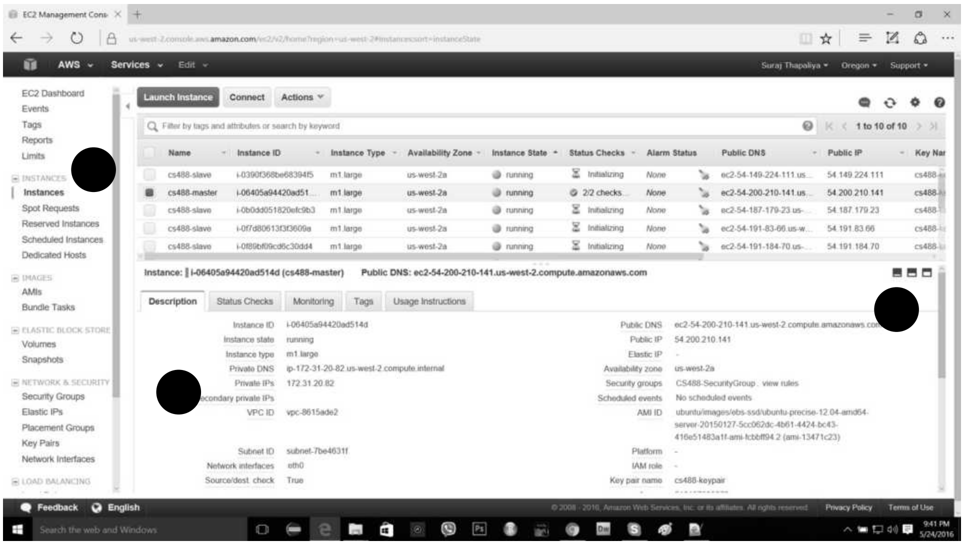
Default VPC: vpc-8615ade2

If everything goes well, you can see have 5 instances, 5 volumes, 1 key pair, 1 or 2 security groups.

We are now successfully created 5 servers.

Step 2: Connecting Server and Installing Required Cloudera Distribution of Hadoop

First, take a note for all your server details, IP Address, DNS address. Master and slaves.



Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Nar
cs488-slave	i-0390f368be68394f5	m1.large	us-west-2a	running	Initializing	None	ec2-54-149-224-111.us...	54.149.224.111	cs488-
cs488-master	i-06405a94420ad51...	m1.large	us-west-2a	running	2/2 checks...	None	ec2-54-200-210-141.us...	54.200.210.141	cs488-
cs488-slave	i-0b0dd051820efc9b3	m1.large	us-west-2a	running	Initializing	None	ec2-54-187-179-23.us-...	54.187.179.23	cs488-
cs488-slave	i-0f7d80613f3f3609a	m1.large	us-west-2a	running	Initializing	None	ec2-54-191-83-66.us-w...	54.191.83.66	cs488-
cs488-slave	i-0f89bf09cd6c30dd4	m1.large	us-west-2a	running	Initializing	None	ec2-54-191-184-70.us-...	54.191.184.70	cs488-

Instance: i-06405a94420ad514d (cs488-master) Public DNS: ec2-54-200-210-141.us-west-2.compute.amazonaws.com

Instance ID	i-06405a94420ad514d	Public DNS	ec2-54-200-210-141.us-west-2.compute.amazonaws.com
Instance state	running	Public IP	54.200.210.141
Instance type	m1.large	Elastic IP	-
Private DNS	ip-172-31-20-82.us-west-2.compute.internal	Availability zone	us-west-2a
Private IPs	172.31.20.82	Security groups	CS488-SecurityGroup . view rules
econdary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-8615ade2	AMI ID	ubuntu/images/ebs-ssd/ubuntu-precise-12.04-amd64-server-20150127-5cc062dc-4b61-4424-bc43-416e51483a1f-ami-fcbbff94.2 (ami-13471c23)
Subnet ID	subnet-7be4631f	Platform	-
Network interfaces	eth0	IAM role	-
Source/dest. check	True	Key pair name	cs488-keypair

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

9:41 PM 5/24/2016

Master Public DNS Address: ec2-54-200-210-141.us-west-2.compute.amazonaws.com

Master Private IP Address: 172.31.20.82

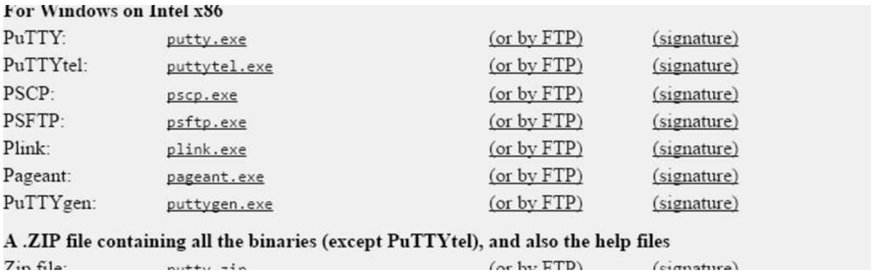
Slave 1 Private IP: 172.31.26.245
Slave 2 Private IP: 172.31.26.242
Slave 3 Private IP: 172.31.26.243
Slave 4 Private IP: 172.31.26.244

Once you have these in recorded, you can connect to the server. If you are using linux as operating system you can use ssh command from terminal to connect it.

Connecting the server (Windows)

1. Download the ssh software (Putty) (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html)

   Also download puttygen to convert our authentication file .pem to .ppk

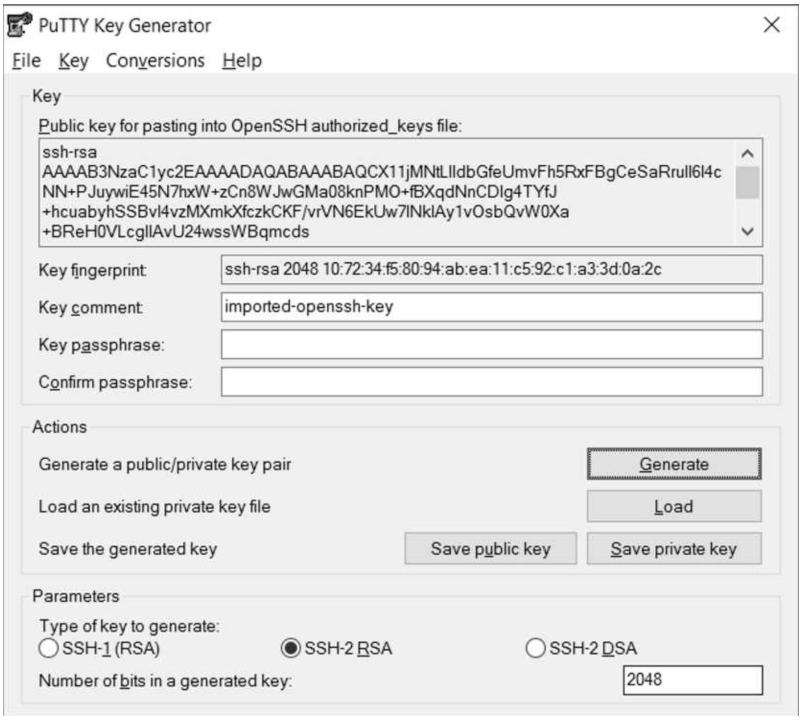


For Windows on Intel x86

PuTTY:	putty.exe	(or by FTP)	(signature)
PuTTYtel:	puttytel.exe	(or by FTP)	(signature)
PSCP:	pscp.exe	(or by FTP)	(signature)
PSFTP:	psftp.exe	(or by FTP)	(signature)
Plink:	plink.exe	(or by FTP)	(signature)
Pageant:	pageant.exe	(or by FTP)	(signature)
PuTTYgen:	puttygen.exe	(or by FTP)	(signature)

A .ZIP file containing all the binaries (except PuTTYtel), and also the help files

Zip file:	putty.zip	(or by FTP)	(signature)



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCX11jMNtLIldbGfeUmvFh5RxFBgCeSaRrull6l4c
NN+PJuywiE45N7hxW+zCn8WJwGMa08knPMO+fBXqdNnCDIg4TYfJ
+hcuabyhSSBvl4vzMXmkXfczkCKF/vrVN6EkUw7INklAy1vOsbQvW0Xa
+BReH0VLcglIAvU24wssWBqmcds
```

Key fingerprint: ssh-rsa 2048 10:72:34:f5:80:94:ab:ea:11:c5:92:c1:a3:3d:0a:2c

Key comment: imported-openssh-key

Key passphrase:

Confirm passphrase:

Generate a public/private key pair

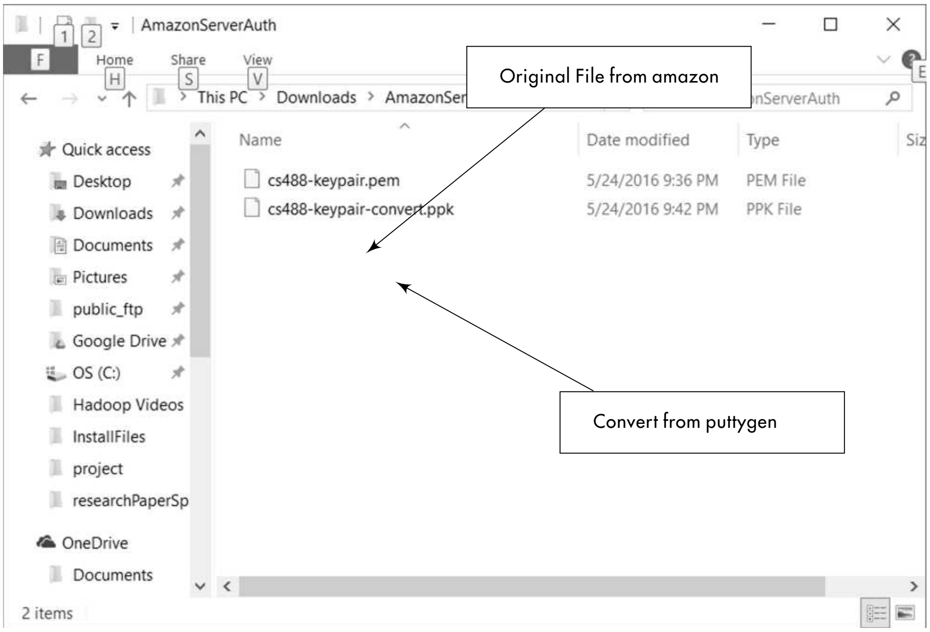
Load an existing private key file

Save the generated key

Type of key to generate:
SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 2048

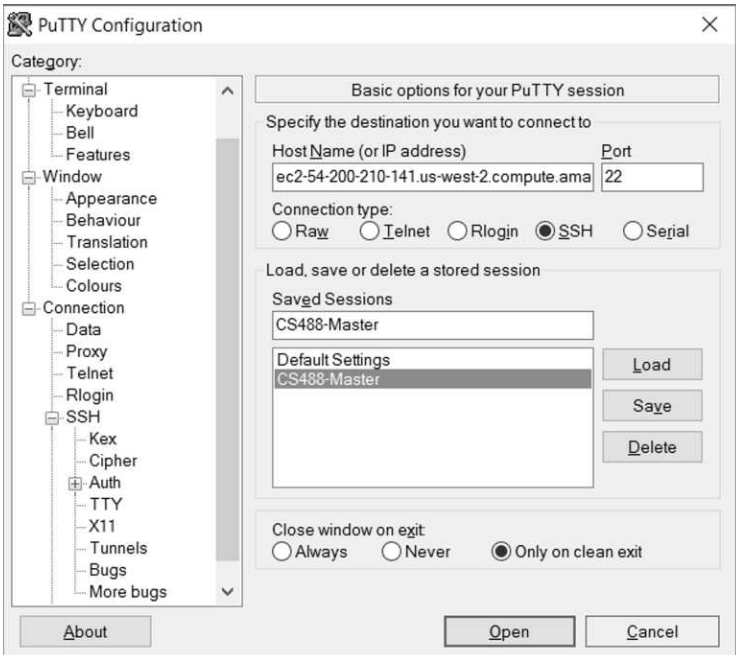
2. Open puttygen load the authentication file
   Click on Save Private Key



Name	Date modified	Type
cs488-keypair.pem	5/24/2016 9:36 PM	PEM File
cs488-keypair-convert.ppk	5/24/2016 9:42 PM	PPK File

2 items

3. Open Putty type the master public dns address in host name and then click on SSH from left panel > Click on Auth >> Select the recent converted authentication file (.ppk) and finally click on Open button.



PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): ec2-54-200-210-141.us-west-2.compute.ama

Port: 22

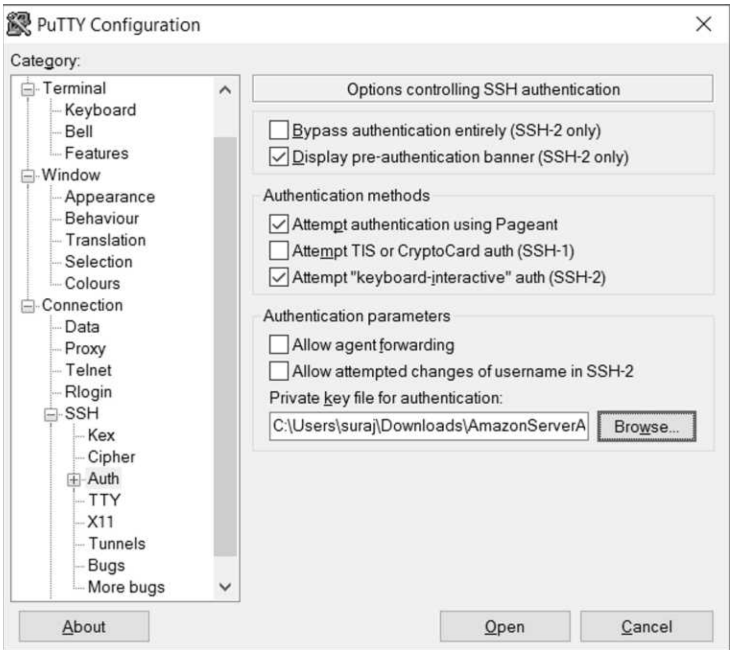
Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: CS488-Master

Default Settings
CS488-Master

Close window on exit: Always Never [✓ Only on clean exit]



PuTTY Configuration

Category:

Options controlling SSH authentication

☐ Bypass authentication entirely (SSH-2 only)
☑ Display pre-authentication banner (SSH-2 only)

Authentication methods

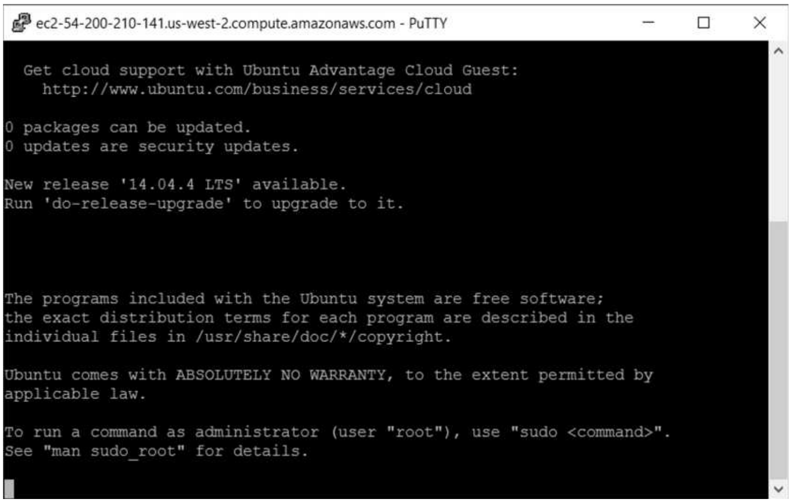
☑ Attempt authentication using Pageant
☐ Attempt TIS or CryptoCard auth (SSH-1)
☑ Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

☐ Allow agent forwarding
☐ Allow attempted changes of username in SSH-2

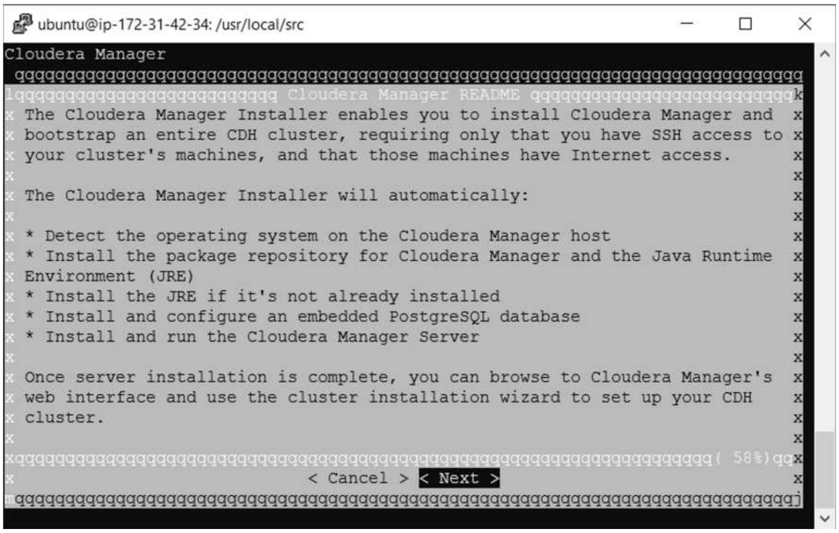
Private key file for authentication: C:\Users\suraj\Downloads\AmazonServerA

4. Now you will able to connect the server please type "ubuntu" the default username to login to the system.



```
  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

0 packages can be updated.
0 updates are security updates.

New release '14.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

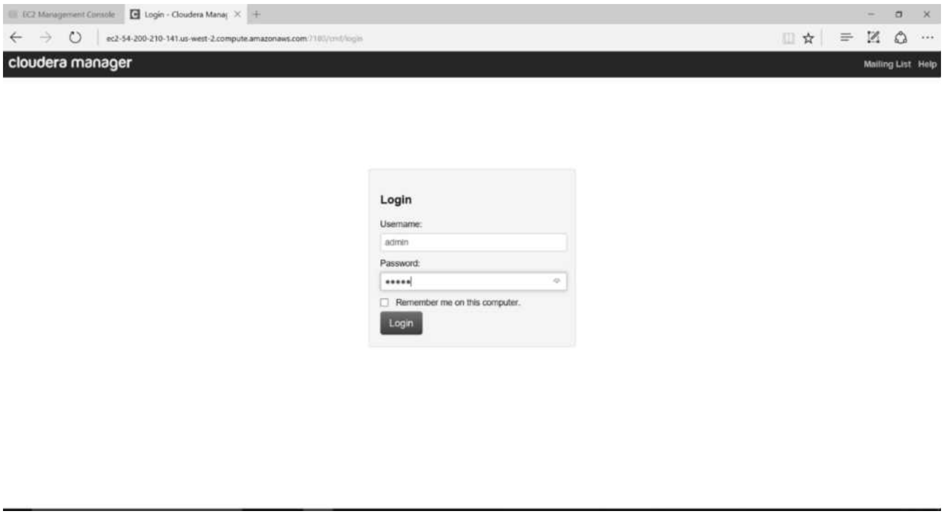
5. Once you connect type the following command into the terminal
6. sudo aptitude update
7. cd /usr/local/src/
8. sudo wget http://archive.cloudera.com/cm4/installer/latest/cloudera-manager-installer.bin
9. sudo chmod u+x cloudera-manager-installer.bin
10. sudo ./cloudera-manager-installer.bin

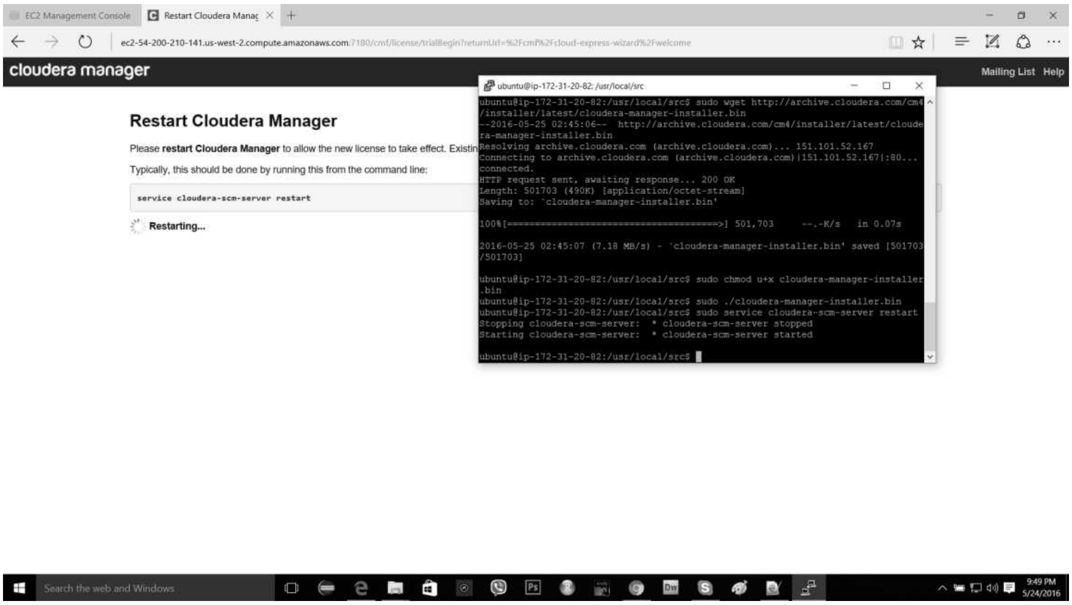


```
Cloudera Manager
           Cloudera Manager README
 The Cloudera Manager Installer enables you to install Cloudera Manager and
 bootstrap an entire CDH cluster, requiring only that you have SSH access to
 your cluster's machines, and that those machines have Internet access.

 The Cloudera Manager Installer will automatically:

 * Detect the operating system on the Cloudera Manager host
 * Install the package repository for Cloudera Manager and the Java Runtime
 Environment (JRE)
 * Install the JRE if it's not already installed
 * Install and configure an embedded PostgreSQL database
 * Install and run the Cloudera Manager Server

 Once server installation is complete, you can browse to Cloudera Manager's
 web interface and use the cluster installation wizard to set up your CDH
 cluster.
                                                                    ( 58%)
                          < Cancel >  < Next >
```

11. There is 4 more step where you click on Next and Yes for license agreement. Once you finish the installation you need to restart the service.
12. sudo service cloudera-scm-server restart

    You are now able to connect the cloudera from your browser. The address will be http://<YOUR PUBLIC DNS SERVER>:7180 e.g. http://ec2-54-200-210-141.us-west-2.compute.amazonaws.com:7180 and default username and password is admin/admin to login to the system.




EC2 Management Console
Restart Cloudera Manag
ec2-54-200-210-141.us-west-2.compute.amazonaws.com:7180/cmf/license/trialBegin?returnUrl=%2Fcmf%2Fcloud-express-wizard%2Fwelcome
cloudera manager
Mailing List
Help
Search the web and Windows
9:49 PM
5/24/2016

Restart Cloudera Manager

Please **restart Cloudera Manager** to allow the new license to take effect. Existin

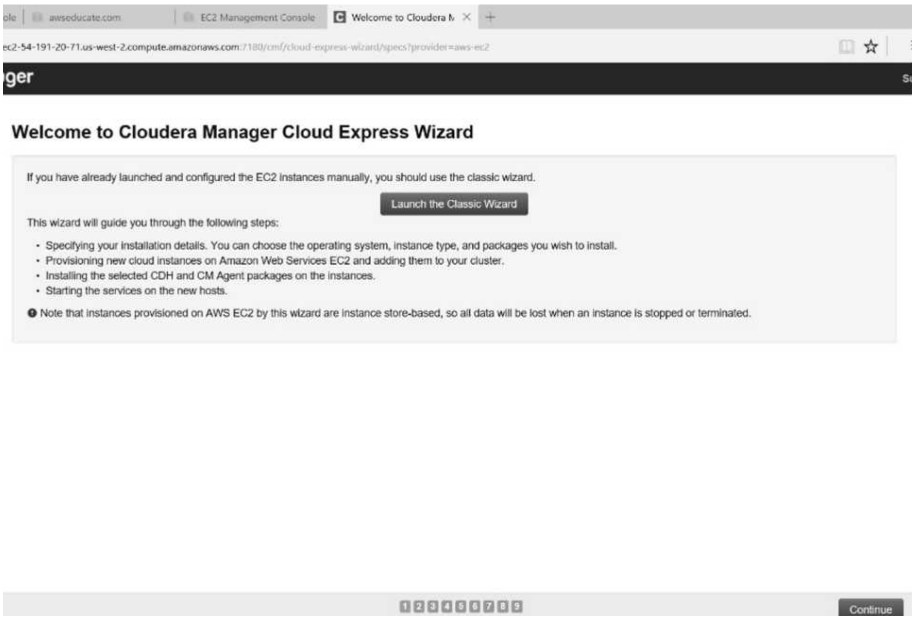
Typically, this should be done by running this from the command line:

```
service cloudera-scm-server restart
```

Restarting...

```
ubuntu@ip-172-31-20-82: /usr/local/src
ubuntu@ip-172-31-20-82:/usr/local/src$ sudo wget http://archive.cloudera.com/cm4
/installer/latest/cloudera-manager-installer.bin
--2016-05-25 02:45:06--  http://archive.cloudera.com/cm4/installer/latest/cloude
ra-manager-installer.bin
Resolving archive.cloudera.com (archive.cloudera.com)... 151.101.52.167
Connecting to archive.cloudera.com (archive.cloudera.com)|151.101.52.167|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 501703 (490K) [application/octet-stream]
Saving to: `cloudera-manager-installer.bin'

100%[======================================>] 501,703     --.-K/s   in 0.07s

2016-05-25 02:45:07 (7.18 MB/s) - `cloudera-manager-installer.bin' saved [501703
/501703]

ubuntu@ip-172-31-20-82:/usr/local/src$ sudo chmod u+x cloudera-manager-installer
.bin
ubuntu@ip-172-31-20-82:/usr/local/src$ sudo ./cloudera-manager-installer.bin
ubuntu@ip-172-31-20-82:/usr/local/src$ sudo service cloudera-scm-server restart
Stopping cloudera-scm-server:  * cloudera-scm-server stopped
Starting cloudera-scm-server:  * cloudera-scm-server started

ubuntu@ip-172-31-20-82:/usr/local/src$
```

Once restart the server it will open the login screen again. The same username and password (admin/admin) is used to login to the system.



ec2-54-191-20-71.us-west-2.compute.amazonaws.com:7180/cmf/cloud-express-wizard/specs?provider=aws-ec2

Welcome to Cloudera Manager Cloud Express Wizard

If you have already launched and configured the EC2 instances manually, you should use the classic wizard.

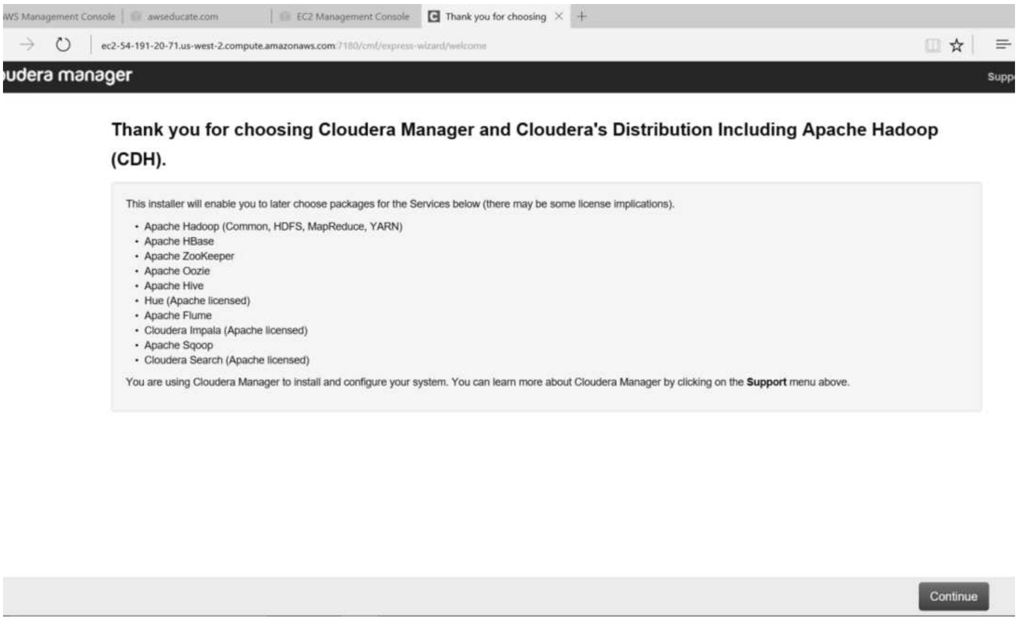
Launch the Classic Wizard

This wizard will guide you through the following steps:

- Specifying your installation details. You can choose the operating system, instance type, and packages you wish to install.
- Provisioning new cloud instances on Amazon Web Services EC2 and adding them to your cluster.
- Installing the selected CDH and CM Agent packages on the instances.
- Starting the services on the new hosts.

Note that instances provisioned on AWS EC2 by this wizard are instance store-based, so all data will be lost when an instance is stopped or terminated.

13. Click on **Launch the Classic wizard**



ec2-54-191-20-71.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/welcome
cloudera manager

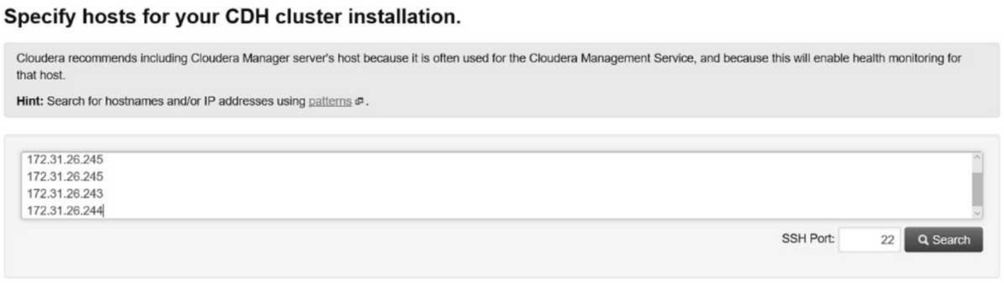
Thank you for choosing Cloudera Manager and Cloudera's Distribution Including Apache Hadoop (CDH).

This installer will enable you to later choose packages for the Services below (there may be some license implications).

- Apache Hadoop (Common, HDFS, MapReduce, YARN)
- Apache HBase
- Apache ZooKeeper
- Apache Oozie
- Apache Hive
- Hue (Apache licensed)
- Apache Flume
- Cloudera Impala (Apache licensed)
- Apache Sqoop
- Cloudera Search (Apache licensed)

You are using Cloudera Manager to install and configure your system. You can learn more about Cloudera Manager by clicking on the **Support** menu above.

14. Click on Continue



Specify hosts for your CDH cluster installation.

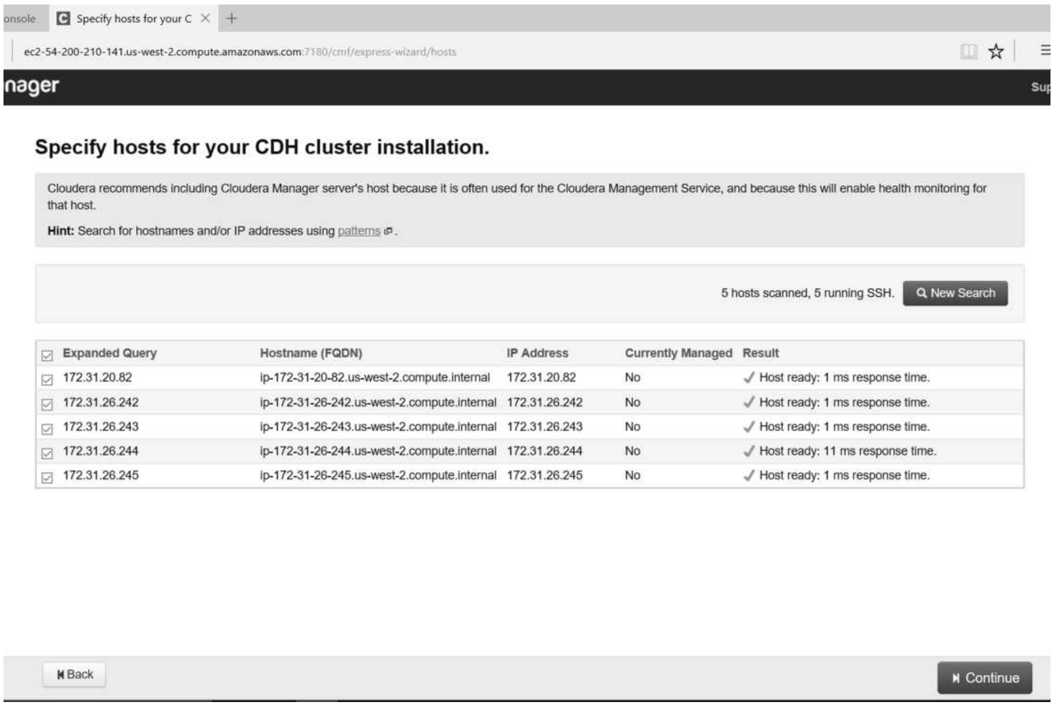
Cloudera recommends including Cloudera Manager server's host because it is often used for the Cloudera Management Service, and because this will enable health monitoring for that host.

Hint: Search for hostnames and/or IP addresses using patterns.

```
172.31.26.245
172.31.26.245
172.31.26.243
172.31.26.244
```

SSH Port: 22

15. Provide all the Private IP address of master and slaves computers and click on Search button.



ec2-54-200-210-141.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/hosts

Specify hosts for your CDH cluster installation.

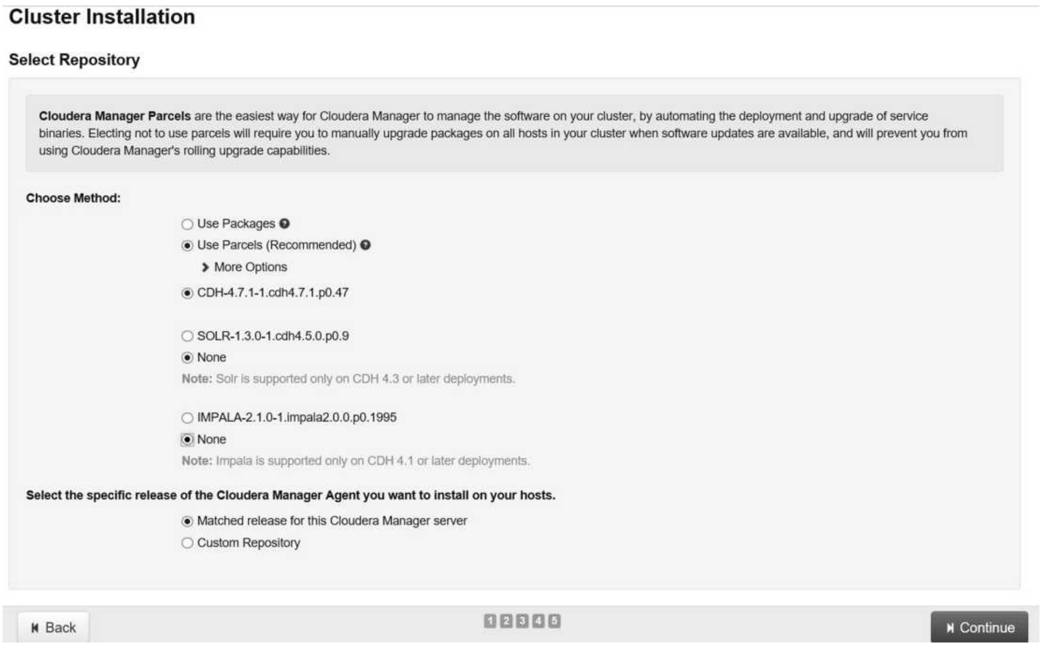
Cloudera recommends including Cloudera Manager server's host because it is often used for the Cloudera Management Service, and because this will enable health monitoring for that host.

Hint: Search for hostnames and/or IP addresses using patterns.

5 hosts scanned, 5 running SSH.

☑	Expanded Query	Hostname (FQDN)	IP Address	Currently Managed	Result
☑	172.31.20.82	ip-172-31-20-82.us-west-2.compute.internal	172.31.20.82	No	✓ Host ready: 1 ms response time.
☑	172.31.26.242	ip-172-31-26-242.us-west-2.compute.internal	172.31.26.242	No	✓ Host ready: 1 ms response time.
☑	172.31.26.243	ip-172-31-26-243.us-west-2.compute.internal	172.31.26.243	No	✓ Host ready: 1 ms response time.
☑	172.31.26.244	ip-172-31-26-244.us-west-2.compute.internal	172.31.26.244	No	✓ Host ready: 11 ms response time.
☑	172.31.26.245	ip-172-31-26-245.us-west-2.compute.internal	172.31.26.245	No	✓ Host ready: 1 ms response time.

16. Click on Continue button.



Cluster Installation

Select Repository

Cloudera Manager Parcels are the easiest way for Cloudera Manager to manage the software on your cluster, by automating the deployment and upgrade of service binaries. Electing not to use parcels will require you to manually upgrade packages on all hosts in your cluster when software updates are available, and will prevent you from using Cloudera Manager's rolling upgrade capabilities.

Choose Method:

- ○ Use Packages
- ● Use Parcels (Recommended)
- › More Options
- ● CDH-4.7.1-1.cdh4.7.1.p0.47
- ○ SOLR-1.3.0-1.cdh4.5.0.p0.9
- ● None

Note: Solr is supported only on CDH 4.3 or later deployments.

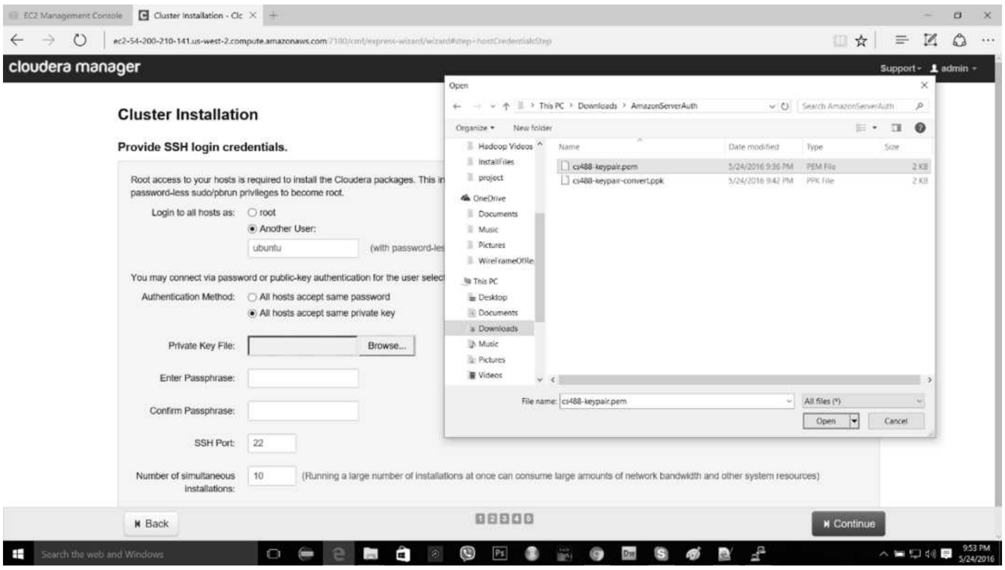
- ○ IMPALA-2.1.0-1.impala2.0.0.p0.1995
- ● None

Note: Impala is supported only on CDH 4.1 or later deployments.

Select the specific release of the Cloudera Manager Agent you want to install on your hosts.

- ● Matched release for this Cloudera Manager server
- ○ Custom Repository

17. Choose **None** for SOLR1.... And **None** for IMPAL.... And Click on Continue button.



EC2 Management Console
Cluster Installation - Clc
ec2-54-200-210-141.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/wizard#step=hostCredentialsStep
cloudera manager
Support
admin

Cluster Installation

Provide SSH login credentials.

Root access to your hosts is required to install the Cloudera packages. This in
password-less sudo/pbrun privileges to become root.

Login to all hosts as: ○ root ● Another User: ubuntu (with password-les

You may connect via password or public-key authentication for the user selec

Authentication Method: ○ All hosts accept same password ● All hosts accept same private key

Private Key File: Browse...

Enter Passphrase:

Confirm Passphrase:

SSH Port: 22

Number of simultaneous installations: 10 (Running a large number of installations at once can consume large amounts of network bandwidth and other system resources)

Open

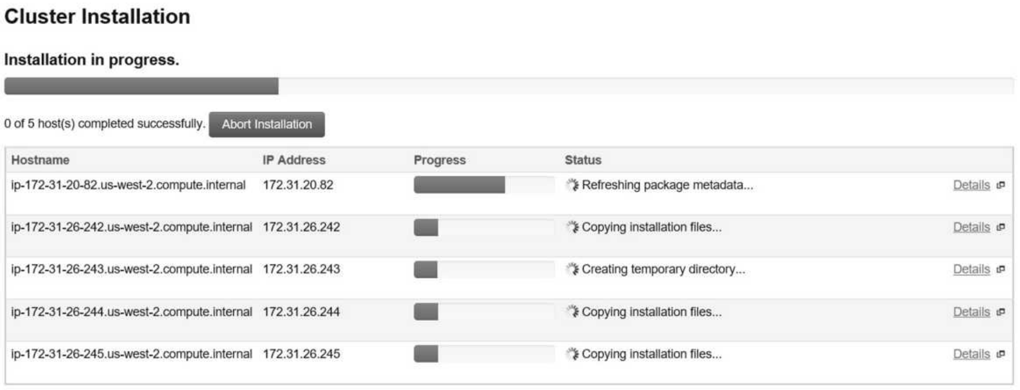
This PC › Downloads › AmazonServerAuth

Name	Date modified	Type	Size
cs488-keypair.pem	5/24/2016 9:36 PM	PEM File	2 KB
cs488-keypair-convert.ppk	5/24/2016 9:42 PM	PPK File	2 KB

File name: cs488-keypair.pem

All files (*)

18. Click on **Another User** >> Type “ubuntu” and select **All hosts accept same private key** >> upload the authentication file .pem and click on Continue button.



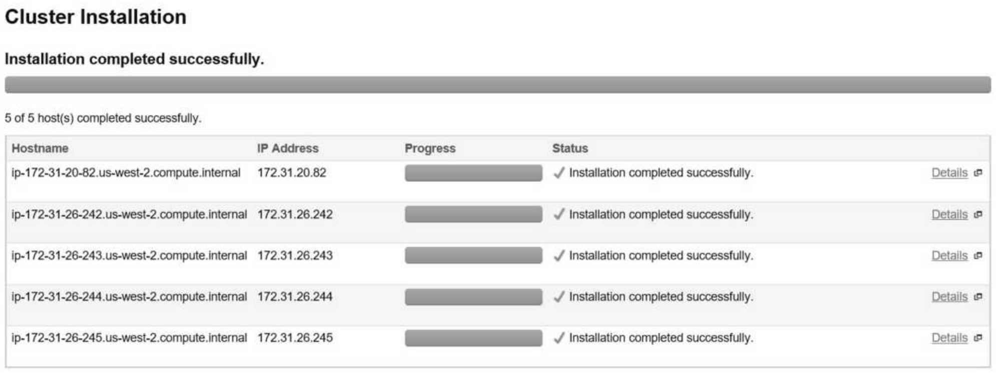
Cluster Installation

Installation in progress.

0 of 5 host(s) completed successfully.

Hostname	IP Address	Progress	Status	
ip-172-31-20-82.us-west-2.compute.internal	172.31.20.82		Refreshing package metadata...	Details
ip-172-31-26-242.us-west-2.compute.internal	172.31.26.242		Copying installation files...	Details
ip-172-31-26-243.us-west-2.compute.internal	172.31.26.243		Creating temporary directory...	Details
ip-172-31-26-244.us-west-2.compute.internal	172.31.26.244		Copying installation files...	Details
ip-172-31-26-245.us-west-2.compute.internal	172.31.26.245		Copying installation files...	Details

19. Now cloudera will install the software for each of our server.



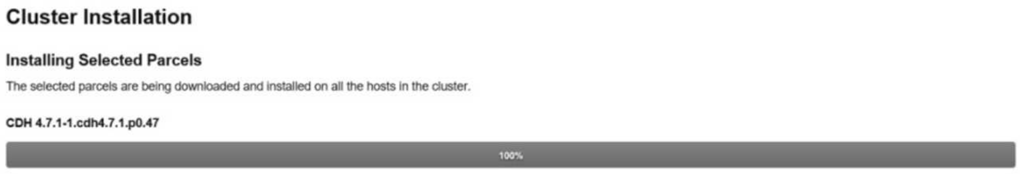
Cluster Installation

Installation completed successfully.

5 of 5 host(s) completed successfully.

Hostname	IP Address	Progress	Status	
ip-172-31-20-82.us-west-2.compute.internal	172.31.20.82		✓ Installation completed successfully.	Details
ip-172-31-26-242.us-west-2.compute.internal	172.31.26.242		✓ Installation completed successfully.	Details
ip-172-31-26-243.us-west-2.compute.internal	172.31.26.243		✓ Installation completed successfully.	Details
ip-172-31-26-244.us-west-2.compute.internal	172.31.26.244		✓ Installation completed successfully.	Details
ip-172-31-26-245.us-west-2.compute.internal	172.31.26.245		✓ Installation completed successfully.	Details

20. Once the installation is complete click on continue button.



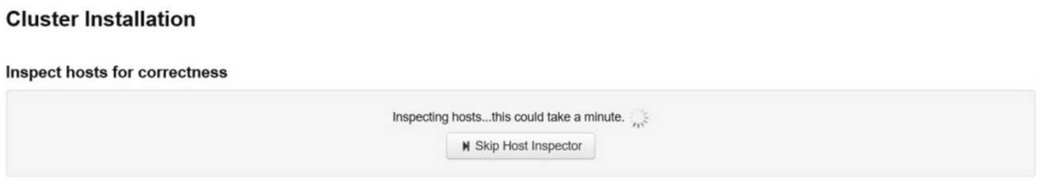
Cluster Installation

Installing Selected Parcels

The selected parcels are being downloaded and installed on all the hosts in the cluster.

CDH 4.7.1-1.cdh4.7.1.p0.47

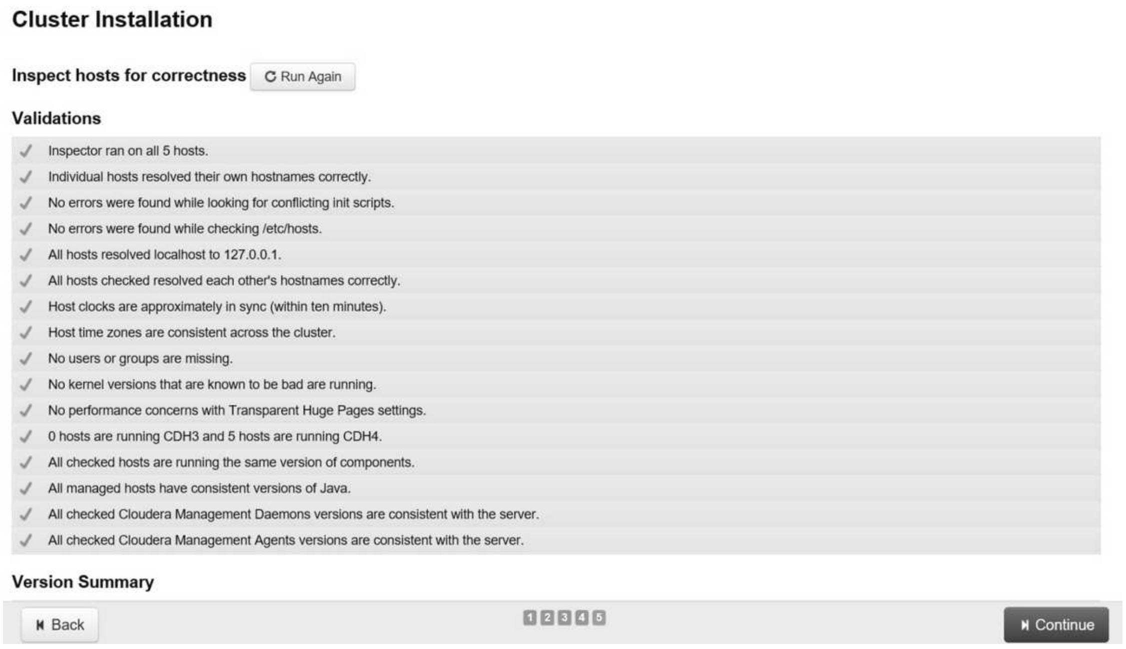
21. Once it reaches to 100% click on continue button. Do not disconnect internet nor shut the machine, If the process will not complete that we need to re-create the whole process. Click on continue button.



Cluster Installation

Inspect hosts for correctness

Inspecting hosts...this could take a minute.



Cluster Installation

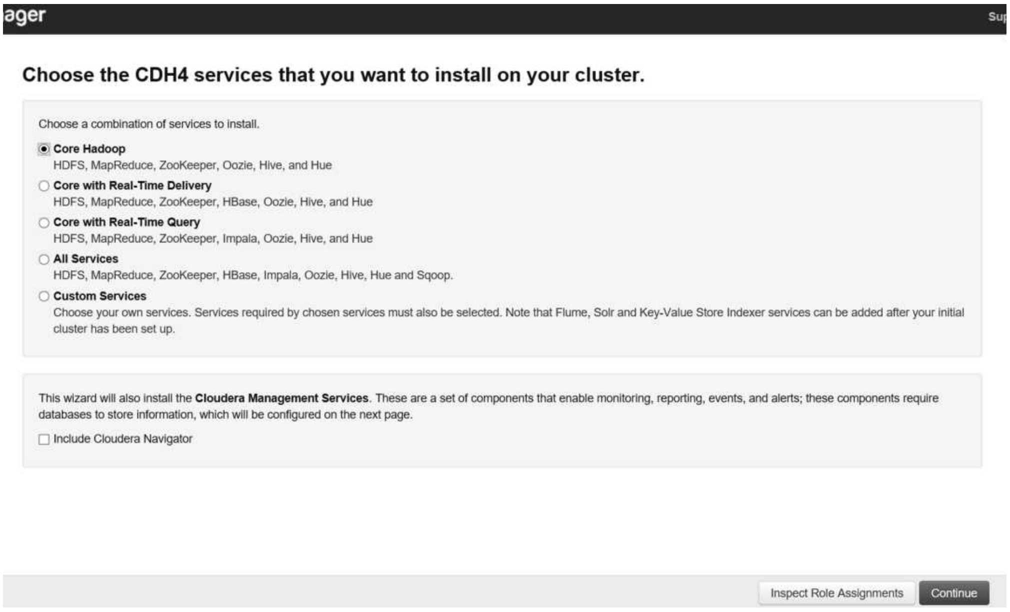
Inspect hosts for correctness

Validations

✓ Inspector ran on all 5 hosts.
✓ Individual hosts resolved their own hostnames correctly.
✓ No errors were found while looking for conflicting init scripts.
✓ No errors were found while checking /etc/hosts.
✓ All hosts resolved localhost to 127.0.0.1.
✓ All hosts checked resolved each other's hostnames correctly.
✓ Host clocks are approximately in sync (within ten minutes).
✓ Host time zones are consistent across the cluster.
✓ No users or groups are missing.
✓ No kernel versions that are known to be bad are running.
✓ No performance concerns with Transparent Huge Pages settings.
✓ 0 hosts are running CDH3 and 5 hosts are running CDH4.
✓ All checked hosts are running the same version of components.
✓ All managed hosts have consistent versions of Java.
✓ All checked Cloudera Management Daemons versions are consistent with the server.
✓ All checked Cloudera Management Agents versions are consistent with the server.

Version Summary

22. Click on Continue.



ager

Choose the CDH4 services that you want to install on your cluster.

Choose a combination of services to install.

[✓ **Core Hadoop**]
HDFS, MapReduce, ZooKeeper, Oozie, Hive, and Hue

Core with Real-Time Delivery
HDFS, MapReduce, ZooKeeper, HBase, Oozie, Hive, and Hue

Core with Real-Time Query
HDFS, MapReduce, ZooKeeper, Impala, Oozie, Hive, and Hue

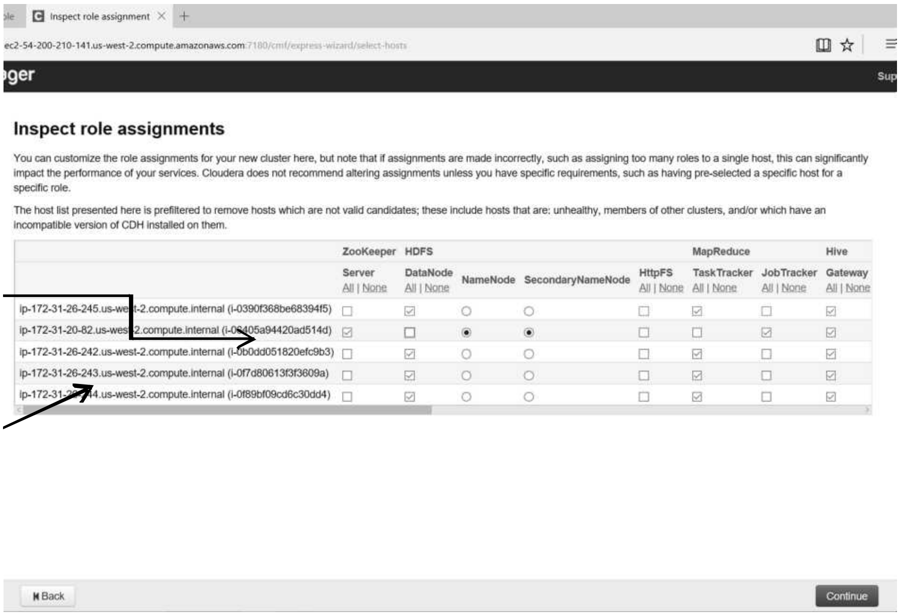
All Services
HDFS, MapReduce, ZooKeeper, HBase, Impala, Oozie, Hive, Hue and Sqoop.

Custom Services
Choose your own services. Services required by chosen services must also be selected. Note that Flume, Solr and Key-Value Store Indexer services can be added after your initial cluster has been set up.

This wizard will also install the **Cloudera Management Services**. These are a set of components that enable monitoring, reporting, events, and alerts; these components require databases to store information, which will be configured on the next page.

☐ Include Cloudera Navigator

23. Choose Core Hadoop and Click on **Inspect Role Assignments** button



Inspect role assignment
ec2-54-200-210-141.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/select-hosts
ager

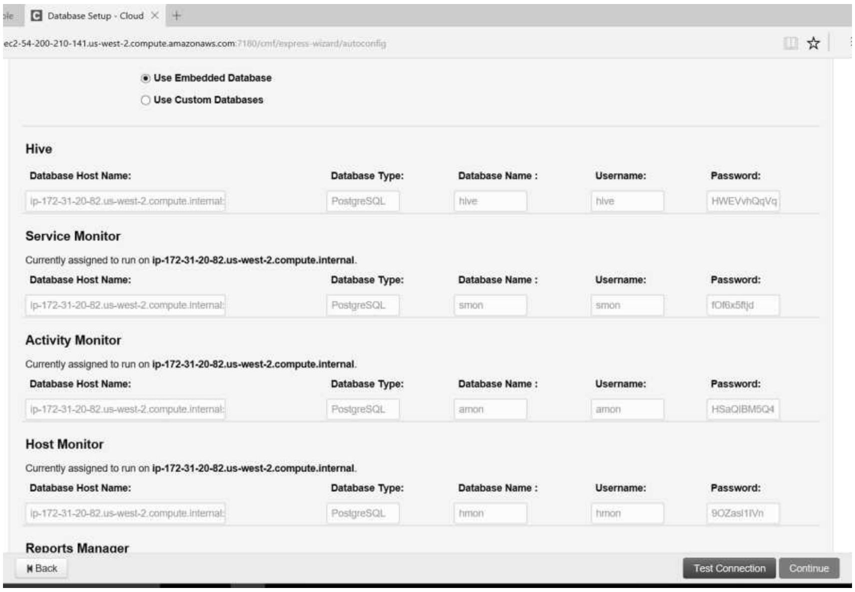
Inspect role assignments

You can customize the role assignments for your new cluster here, but note that if assignments are made incorrectly, such as assigning too many roles to a single host, this can significantly impact the performance of your services. Cloudera does not recommend altering assignments unless you have specific requirements, such as having pre-selected a specific host for a specific role.

The host list presented here is prefiltered to remove hosts which are not valid candidates; these include hosts that are: unhealthy, members of other clusters, and/or which have an incompatible version of CDH installed on them.

	ZooKeeper	HDFS				MapReduce		Hive
	Server All | None	DataNode All | None	NameNode	SecondaryNameNode	HttpFS All | None	TaskTracker All | None	JobTracker All | None	Gateway All | None
ip-172-31-26-245.us-west-2.compute.internal (i-0390f368be68394f5)	☐	☑	○	○	☐	☑	☐	☑
ip-172-31-20-82.us-west-2.compute.internal (i-0[illegible]405a94420ad514d)	☑	☐	●	●	☐	☐	☑	☑
ip-172-31-26-242.us-west-2.compute.internal (i-0b0dd051820efc9b3)	☐	☑	○	○	☐	☑	☐	☑
ip-172-31-26-243.us-west-2.compute.internal (i-0f7d80613f3f3609a)	☐	☑	○	○	☐	☑	☐	☑
ip-172-31-2[illegible]4.us-west-2.compute.internal (i-0f89bf09cd6c30dd4)	☐	☑	○	○	☐	☑	☐	☑

24. Now for you master IP it should have only Name Node selection and unchecked in Data Node. This is important to make the master and slave server.



Database Setup - Cloud

ec2-54-200-210-141.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/autoconfig

(●) Use Embedded Database
( ) Use Custom Databases

Hive

Database Host Name:	Database Type:	Database Name :	Username:	Password:
ip-172-31-20-82.us-west-2.compute.internal:	PostgreSQL	hive	hive	HWEVvhQqVq

Service Monitor

Currently assigned to run on **ip-172-31-20-82.us-west-2.compute.internal**.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
ip-172-31-20-82.us-west-2.compute.internal:	PostgreSQL	smon	smon	fOf6x5ftjd

Activity Monitor

Currently assigned to run on **ip-172-31-20-82.us-west-2.compute.internal**.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
ip-172-31-20-82.us-west-2.compute.internal:	PostgreSQL	amon	amon	HSaQIBM5Q4

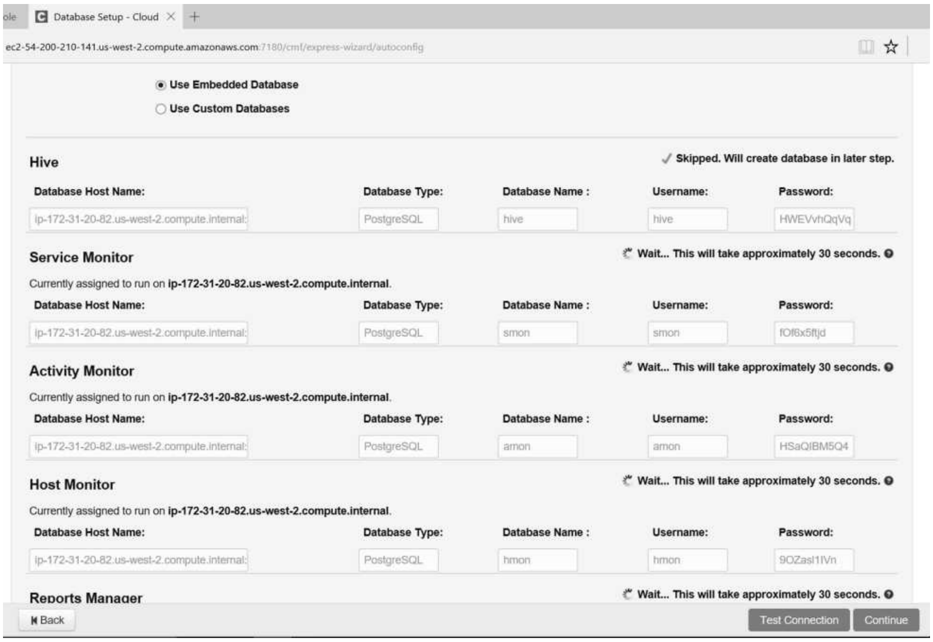
Host Monitor

Currently assigned to run on **ip-172-31-20-82.us-west-2.compute.internal**.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
ip-172-31-20-82.us-west-2.compute.internal:	PostgreSQL	hmon	hmon	9OZasI1IVn

Reports Manager

25. Now the cloudera will install the all the services for your future use; you can record the username and password of each services. Click on Test Connection



Database Setup - Cloud

ec2-54-200-210-141.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/autoconfig

(●) Use Embedded Database
( ) Use Custom Databases

Hive — ✓ Skipped. Will create database in later step.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
ip-172-31-20-82.us-west-2.compute.internal:	PostgreSQL	hive	hive	HWEVvhQqVq

Service Monitor — Wait... This will take approximately 30 seconds.

Currently assigned to run on **ip-172-31-20-82.us-west-2.compute.internal**.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
ip-172-31-20-82.us-west-2.compute.internal:	PostgreSQL	smon	smon	fOf6x5ftjd

Activity Monitor — Wait... This will take approximately 30 seconds.

Currently assigned to run on **ip-172-31-20-82.us-west-2.compute.internal**.

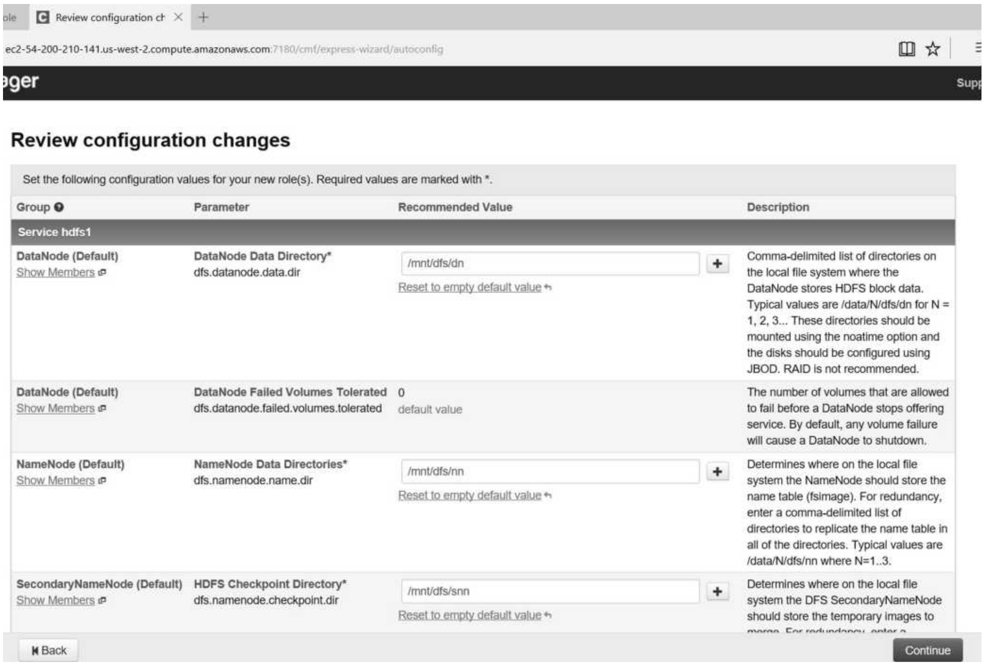
Database Host Name:	Database Type:	Database Name :	Username:	Password:
ip-172-31-20-82.us-west-2.compute.internal:	PostgreSQL	amon	amon	HSaQIBM5Q4

Host Monitor — Wait... This will take approximately 30 seconds.

Currently assigned to run on **ip-172-31-20-82.us-west-2.compute.internal**.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
ip-172-31-20-82.us-west-2.compute.internal:	PostgreSQL	hmon	hmon	9OZasI1IVn

Reports Manager — Wait... This will take approximately 30 seconds.



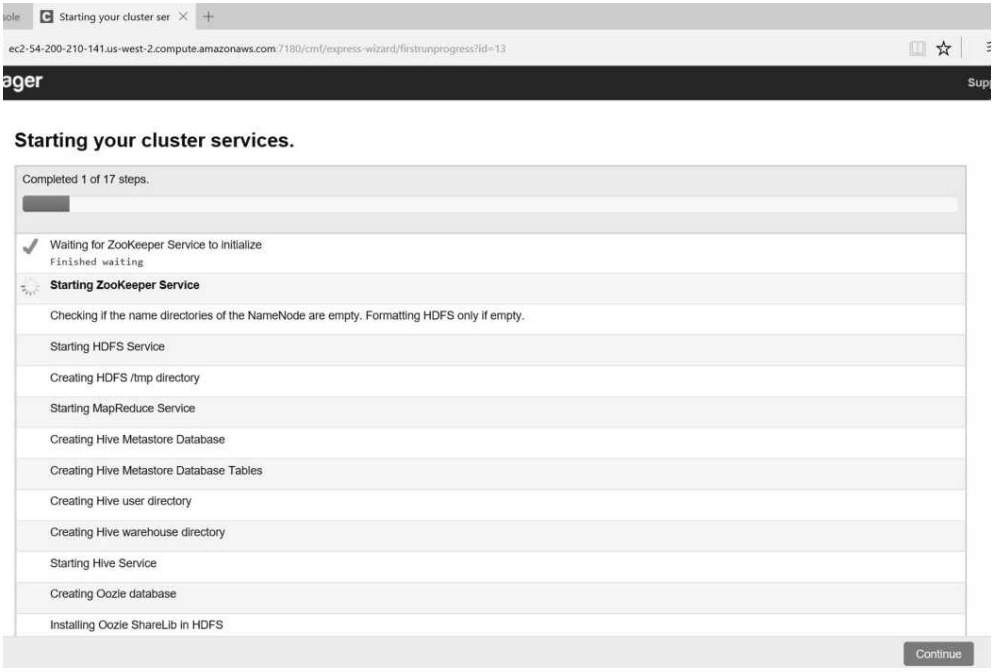
ec2-54-200-210-141.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/autoconfig

Review configuration changes

Set the following configuration values for your new role(s). Required values are marked with *.

Group	Parameter	Recommended Value	Description
Service hdfs1			
DataNode (Default) Show Members	DataNode Data Directory* dfs.datanode.data.dir	/mnt/dfs/dn Reset to empty default value	Comma-delimited list of directories on the local file system where the DataNode stores HDFS block data. Typical values are /data/N/dfs/dn for N = 1, 2, 3... These directories should be mounted using the noatime option and the disks should be configured using JBOD. RAID is not recommended.
DataNode (Default) Show Members	DataNode Failed Volumes Tolerated dfs.datanode.failed.volumes.tolerated	0 default value	The number of volumes that are allowed to fail before a DataNode stops offering service. By default, any volume failure will cause a DataNode to shutdown.
NameNode (Default) Show Members	NameNode Data Directories* dfs.namenode.name.dir	/mnt/dfs/nn Reset to empty default value	Determines where on the local file system the NameNode should store the name table (fsimage). For redundancy, enter a comma-delimited list of directories to replicate the name table in all of the directories. Typical values are /data/N/dfs/nn where N=1..3.
SecondaryNameNode (Default) Show Members	HDFS Checkpoint Directory* dfs.namenode.checkpoint.dir	/mnt/dfs/snn Reset to empty default value	Determines where on the local file system the DFS SecondaryNameNode should store the temporary images to merge. For redundancy, enter a

26. Click on Continue

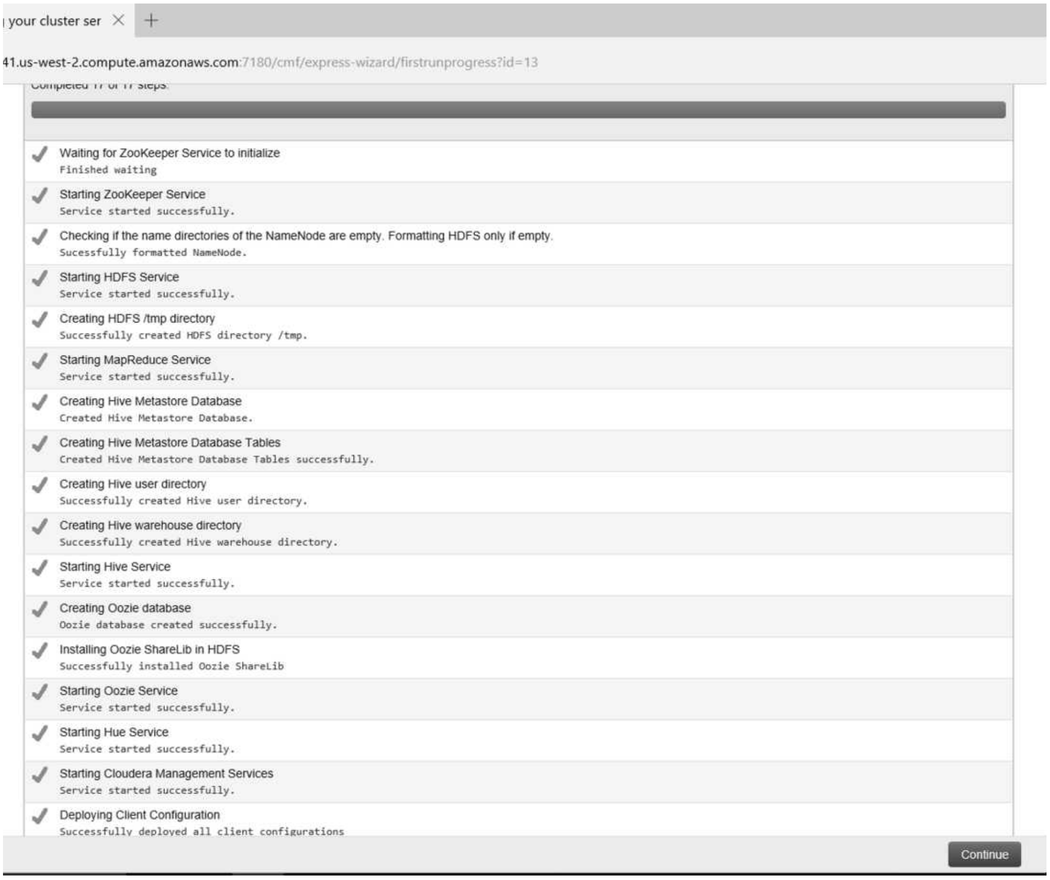


ec2-54-200-210-141.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/firstrunprogress?id=13

Starting your cluster services.

Completed 1 of 17 steps.

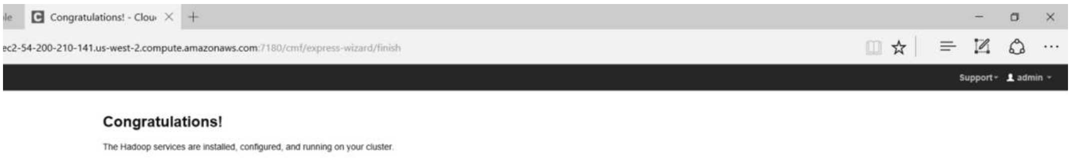
- ✓ Waiting for ZooKeeper Service to initialize
  `Finished waiting`
- **Starting ZooKeeper Service**
- Checking if the name directories of the NameNode are empty. Formatting HDFS only if empty.
- Starting HDFS Service
- Creating HDFS /tmp directory
- Starting MapReduce Service
- Creating Hive Metastore Database
- Creating Hive Metastore Database Tables
- Creating Hive user directory
- Creating Hive warehouse directory
- Starting Hive Service
- Creating Oozie database
- Installing Oozie ShareLib in HDFS



your cluster ser
41.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/firstrunprogress?id=13
Completed 17 of 17 steps.
Continue

```
Waiting for ZooKeeper Service to initialize
Finished waiting
Starting ZooKeeper Service
Service started successfully.
Checking if the name directories of the NameNode are empty. Formatting HDFS only if empty.
Sucessfully formatted NameNode.
Starting HDFS Service
Service started successfully.
Creating HDFS /tmp directory
Successfully created HDFS directory /tmp.
Starting MapReduce Service
Service started successfully.
Creating Hive Metastore Database
Created Hive Metastore Database.
Creating Hive Metastore Database Tables
Created Hive Metastore Database Tables successfully.
Creating Hive user directory
Successfully created Hive user directory.
Creating Hive warehouse directory
Successfully created Hive warehouse directory.
Starting Hive Service
Service started successfully.
Creating Oozie database
Oozie database created successfully.
Installing Oozie ShareLib in HDFS
Successfully installed Oozie ShareLib
Starting Oozie Service
Service started successfully.
Starting Hue Service
Service started successfully.
Starting Cloudera Management Services
Service started successfully.
Deploying Client Configuration
Successfully deployed all client configurations
```

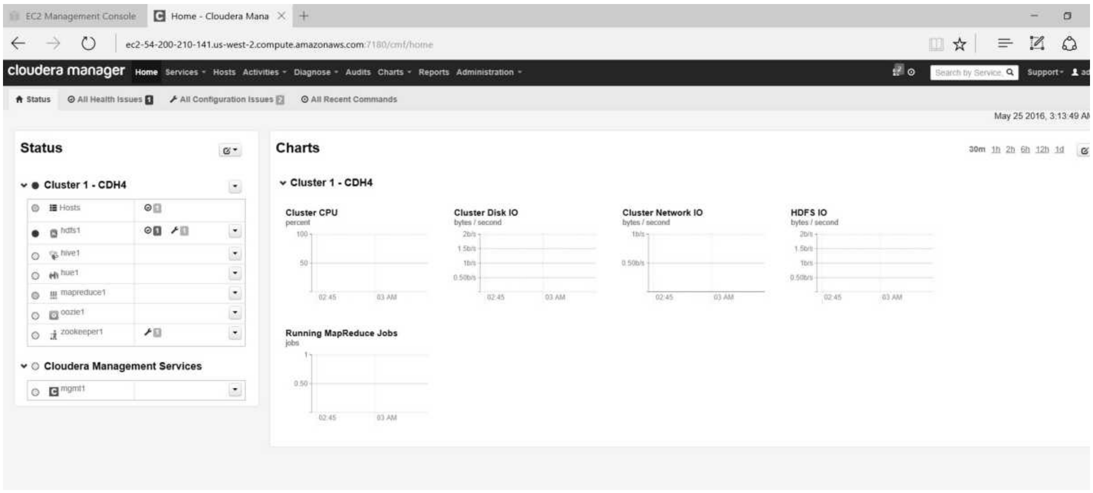
27. Now all the installation is complete you can now have 1 master node 4 data node.



Congratulations! - Clou
ec2-54-200-210-141.us-west-2.compute.amazonaws.com:7180/cmf/express-wizard/finish
Support
admin

Congratulations!

The Hadoop services are installed, configured, and running on your cluster.



28. You should see the dashboard.

Step 3: WordCount using MapReduce

29. Now login to master server from putty.

30. Run the following command

31. cd ~/

32. mkdir code-and-data

33. cd code-and-data

34. sudo wget https://s3.amazonaws.com/learn-hadoop/hadoop-infiniteskills-richmorrow-class.tgz

35. sudo tar -xvzf hadoop-infiniteskills-richmorrow-class.tgz

36. cd data

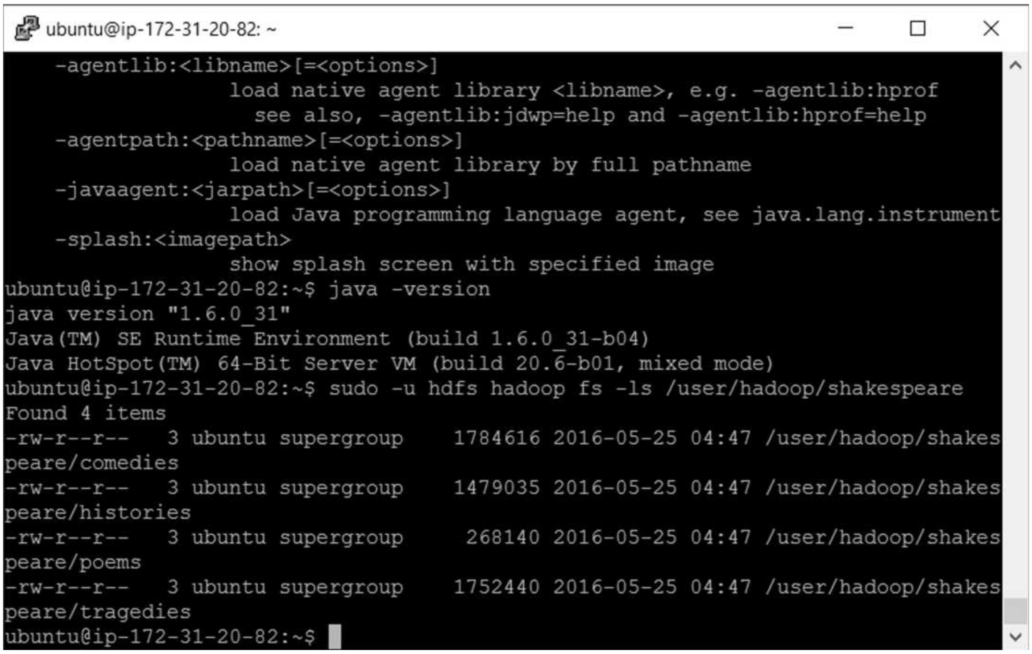
37. sudo -u hdfs hadoop fs -mkdir /user/ubuntu

38. sudo -u hdfs hadoop fs -chown ubuntu /user/ubuntu

39. hadoop fs -put shakespeare shakespeare-hdfs

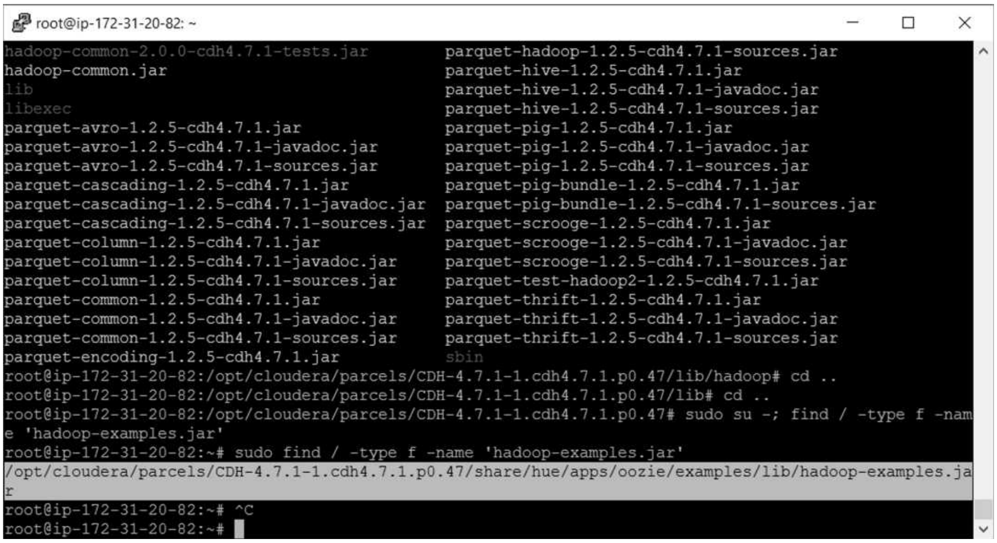
40. hadoop version

41. hadoop fs -ls shakespeare-hdfs

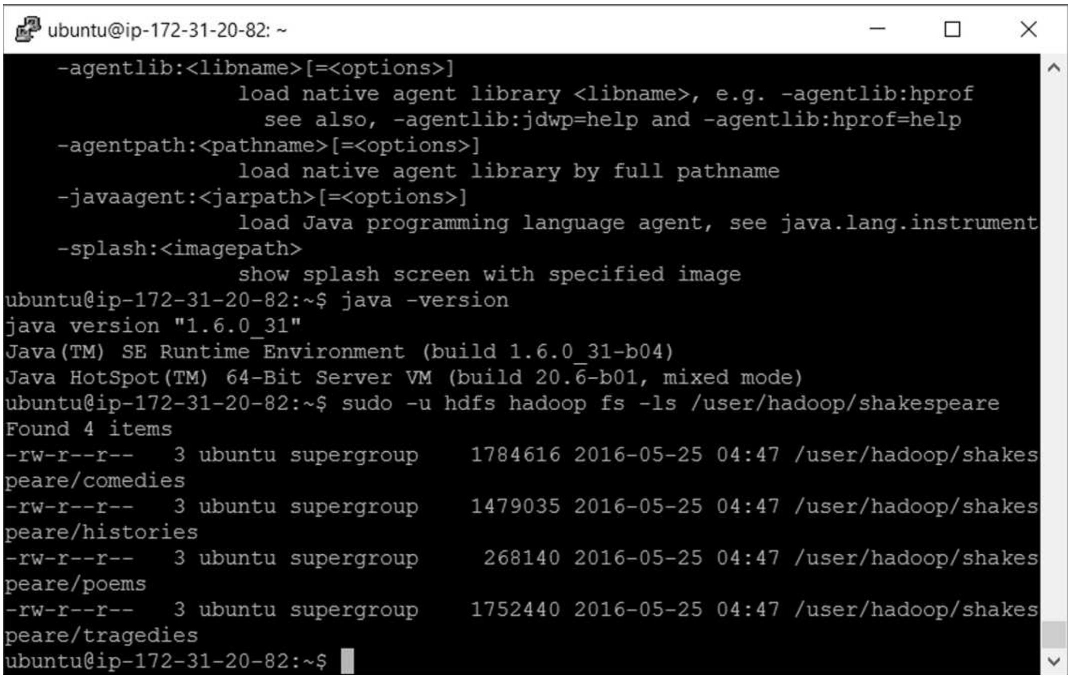


```
    -agentlib:<libname>[=<options>]
                  load native agent library <libname>, e.g. -agentlib:hprof
                    see also, -agentlib:jdwp=help and -agentlib:hprof=help
    -agentpath:<pathname>[=<options>]
                  load native agent library by full pathname
    -javaagent:<jarpath>[=<options>]
                  load Java programming language agent, see java.lang.instrument
    -splash:<imagepath>
                  show splash screen with specified image
ubuntu@ip-172-31-20-82:~$ java -version
java version "1.6.0_31"
Java(TM) SE Runtime Environment (build 1.6.0_31-b04)
Java HotSpot(TM) 64-Bit Server VM (build 20.6-b01, mixed mode)
ubuntu@ip-172-31-20-82:~$ sudo -u hdfs hadoop fs -ls /user/hadoop/shakespeare
Found 4 items
-rw-r--r--   3 ubuntu supergroup    1784616 2016-05-25 04:47 /user/hadoop/shakes
peare/comedies
-rw-r--r--   3 ubuntu supergroup    1479035 2016-05-25 04:47 /user/hadoop/shakes
peare/histories
-rw-r--r--   3 ubuntu supergroup     268140 2016-05-25 04:47 /user/hadoop/shakes
peare/poems
-rw-r--r--   3 ubuntu supergroup    1752440 2016-05-25 04:47 /user/hadoop/shakes
peare/tragedies
ubuntu@ip-172-31-20-82:~$ 
```

42. sudo hadoop jar /opt/cloudera/parcels/CDH-4.7.1-1.cdh4.7.1.p0.47/share/hue/apps/oozie/examples/lib/hadoop-examples.jar wordcount shakespeare-hdfs wordcount-output
43. hadoop jar /opt/cloudera/parcels/CDH-4.7.1-1.cdh4.7.1.p0.47/share/hue/apps/oozie/examples/lib/hadoop-examples.jar sleep -m 10 -r 10 -mt 20000 -rt 20000

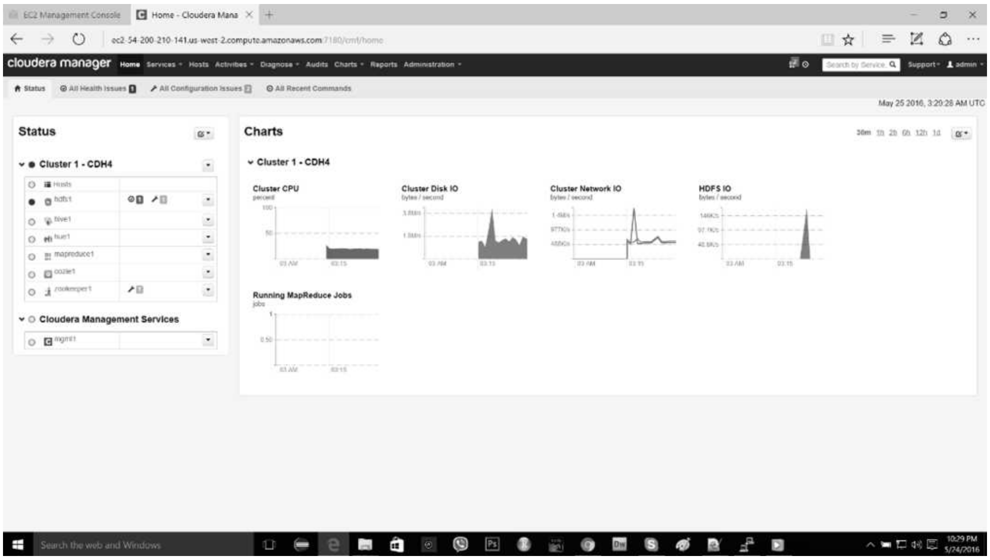


```
hadoop-common-2.0.0-cdh4.7.1-tests.jar       parquet-hadoop-1.2.5-cdh4.7.1-sources.jar
hadoop-common.jar                            parquet-hive-1.2.5-cdh4.7.1.jar
lib                                          parquet-hive-1.2.5-cdh4.7.1-javadoc.jar
libexec                                      parquet-hive-1.2.5-cdh4.7.1-sources.jar
parquet-avro-1.2.5-cdh4.7.1.jar              parquet-pig-1.2.5-cdh4.7.1.jar
parquet-avro-1.2.5-cdh4.7.1-javadoc.jar      parquet-pig-1.2.5-cdh4.7.1-javadoc.jar
parquet-avro-1.2.5-cdh4.7.1-sources.jar      parquet-pig-1.2.5-cdh4.7.1-sources.jar
parquet-cascading-1.2.5-cdh4.7.1.jar         parquet-pig-bundle-1.2.5-cdh4.7.1.jar
parquet-cascading-1.2.5-cdh4.7.1-javadoc.jar parquet-pig-bundle-1.2.5-cdh4.7.1-sources.jar
parquet-cascading-1.2.5-cdh4.7.1-sources.jar parquet-scrooge-1.2.5-cdh4.7.1.jar
parquet-column-1.2.5-cdh4.7.1.jar            parquet-scrooge-1.2.5-cdh4.7.1-javadoc.jar
parquet-column-1.2.5-cdh4.7.1-javadoc.jar    parquet-scrooge-1.2.5-cdh4.7.1-sources.jar
parquet-column-1.2.5-cdh4.7.1-sources.jar    parquet-test-hadoop2-1.2.5-cdh4.7.1.jar
parquet-common-1.2.5-cdh4.7.1.jar            parquet-thrift-1.2.5-cdh4.7.1.jar
parquet-common-1.2.5-cdh4.7.1-javadoc.jar    parquet-thrift-1.2.5-cdh4.7.1-javadoc.jar
parquet-common-1.2.5-cdh4.7.1-sources.jar    parquet-thrift-1.2.5-cdh4.7.1-sources.jar
parquet-encoding-1.2.5-cdh4.7.1.jar          sbin
root@ip-172-31-20-82:/opt/cloudera/parcels/CDH-4.7.1-1.cdh4.7.1.p0.47/lib/hadoop# cd ..
root@ip-172-31-20-82:/opt/cloudera/parcels/CDH-4.7.1-1.cdh4.7.1.p0.47/lib# cd ..
root@ip-172-31-20-82:/opt/cloudera/parcels/CDH-4.7.1-1.cdh4.7.1.p0.47# sudo su -; find / -type f -nam
e 'hadoop-examples.jar'
root@ip-172-31-20-82:~# sudo find / -type f -name 'hadoop-examples.jar'
/opt/cloudera/parcels/CDH-4.7.1-1.cdh4.7.1.p0.47/share/hue/apps/oozie/examples/lib/hadoop-examples.ja
r
root@ip-172-31-20-82:~# ^C
root@ip-172-31-20-82:~# 
```



ubuntu@ip-172-31-20-82: ~

```
    -agentlib:<libname>[=<options>]
                  load native agent library <libname>, e.g. -agentlib:hprof
                    see also, -agentlib:jdwp=help and -agentlib:hprof=help
    -agentpath:<pathname>[=<options>]
                  load native agent library by full pathname
    -javaagent:<jarpath>[=<options>]
                  load Java programming language agent, see java.lang.instrument
    -splash:<imagepath>
                  show splash screen with specified image
ubuntu@ip-172-31-20-82:~$ java -version
java version "1.6.0_31"
Java(TM) SE Runtime Environment (build 1.6.0_31-b04)
Java HotSpot(TM) 64-Bit Server VM (build 20.6-b01, mixed mode)
ubuntu@ip-172-31-20-82:~$ sudo -u hdfs hadoop fs -ls /user/hadoop/shakespeare
Found 4 items
-rw-r--r--   3 ubuntu supergroup    1784616 2016-05-25 04:47 /user/hadoop/shakes
peare/comedies
-rw-r--r--   3 ubuntu supergroup    1479035 2016-05-25 04:47 /user/hadoop/shakes
peare/histories
-rw-r--r--   3 ubuntu supergroup     268140 2016-05-25 04:47 /user/hadoop/shakes
peare/poems
-rw-r--r--   3 ubuntu supergroup    1752440 2016-05-25 04:47 /user/hadoop/shakes
peare/tragedies
ubuntu@ip-172-31-20-82:~$ 
```



Appendix 3

Spark Installation and Tutorial

A3.1 SPARK INSTALLING AND RUNNING

This tutorial will help install Spark and get it running on a standalone machine. It will then help develop a simple analytical application using R language.

Step 1: Verifying Java Installation

Java installation is one of the mandatory things in installing Spark. Try the following command to verify the JAVA version.

```
$java -version
```

If Java is already, installed on your system, you get to see the following response −

```
java version “1.7.0_71”
Java(TM) SE Runtime Environment (build 1.7.0_71-b13)
Java HotSpot(TM) Client VM (build 25.0-b02, mixed mode)
```

In case you do not have Java installed on your system, then Install Java before proceeding to next step.

Step 2: Verifying Scala installation

Verify Scala installation using following command.

```
$scala -version
```

If Scala is already installed on your system, you get to see the following response −

Scala code runner version 2.11.6 -- Copyright 2002-2013, LAMP/EPFL

In case you don’t have Scala installed on your system, then proceed to next step for Scala installation.

Step 3: Downloading Scala

Download the latest version of Scala by visit the following link Download Scala. For this tutorial, we are using scala-2.11.6 version. After downloading, you will find the Scala tar file in the download folder.

Step 4: Installing Scala

Follow the below given steps for installing Scala.

Extract the Scala tar file

Type the following command for extracting the Scala tar file.

```
$ tar xvf scala-2.11.6.tgz
```

Move Scala software files

Use the following commands for moving the Scala software files, to respective directory **(/usr/local/scala)**.

```
$ su -
```

Password:

```
# cd /home/Hadoop/Downloads/
# mv scala-2.11.6 /usr/local/scala
# exit
```

Set PATH for Scala

Use the following command for setting PATH for Scala.

```
$ export PATH = $PATH:/usr/local/scala/bin
```

Verifying Scala Installation

After installation, it is better to verify it. Use the following command for verifying Scala installation.

```
$scala -version
```

If Scala is already installed on your system, you get to see the following response −

Scala code runner version 2.11.6 -- Copyright 2002-2013, LAMP/EPFL

Step 5: Downloading Spark

Download the latest version of Spark. For this tutorial, we are using **spark-1.3.1-bin-hadoop2.6** version. After downloading it, you will find the Spark tar file in the download folder.

Step 6: Installing Spark

Follow the steps given below for installing Spark.

Extracting Spark tar

The following command for extracting the spark tar file.

```
$ tar xvf spark-1.3.1-bin-hadoop2.6.tgz
```

Moving Spark software files

The following commands for moving the Spark software files to respective directory **(/usr/local/spark)**.

```
$ su -
Password:
# cd /home/Hadoop/Downloads/
# mv spark-1.3.1-bin-hadoop2.6 /usr/local/spark
# exit
```

Setting up the environment for Spark

Add the following line to **~/.bashrc** file. It means adding the location, where the spark software file are located to the PATH variable.

export PATH = $PATH:/usr/local/spark/bin

Use the following command for sourcing the ~/.bashrc file.

```
$ source ~/.bashrc
```

Step 7: Verifying the Spark Installation

Write the following command for opening Spark shell.

```
$spark-shell
```

If spark is installed successfully then you will find the following output.

```
Spark assembly has been built with Hive, including Datanucleus jars on classpath

Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties

15/06/04 15:25:22 INFO SecurityManager: Changing view acls to: hadoop

15/06/04 15:25:22 INFO SecurityManager: Changing modify acls to: hadoop

15/06/04 15:25:22 INFO SecurityManager: SecurityManager: authentication disabled;

 ui acls disabled; users with view permissions: Set(hadoop); users with modify per-
missions: Set(hadoop)

15/06/04 15:25:22 INFO HttpServer: Starting HTTP Server

15/06/04 15:25:23 INFO Utils: Successfully started service 'HTTP class server' on port
43292.

Welcome to Spark version 1.4.0

Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_71)

Type in expressions to have them evaluated.

Spark context available as sc

scala>
```

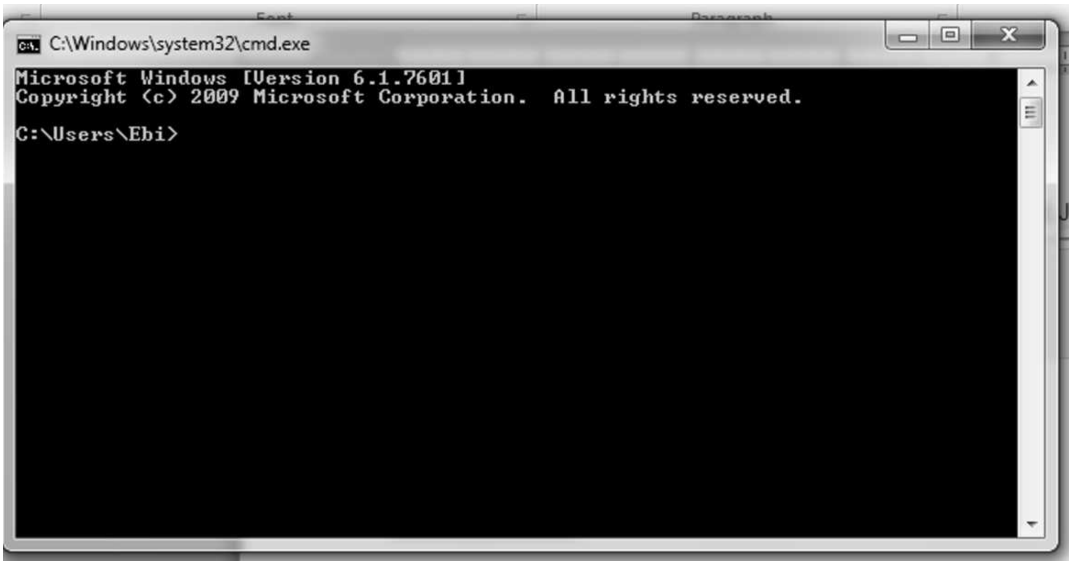
Here you can see the video:

How to install Spark

You might encounter "file specified not found error" when you are first installing SPARK stand alone:

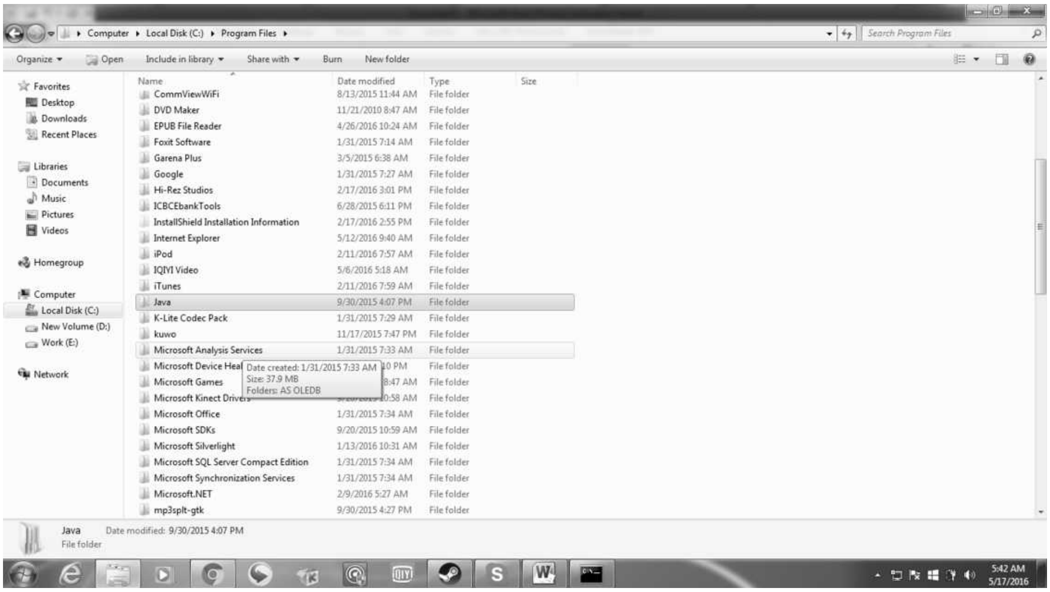
To fix this you have to set up your JAVA_HOME

Step 1: Start->run->command prompt(cmd)



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Ebi>
```

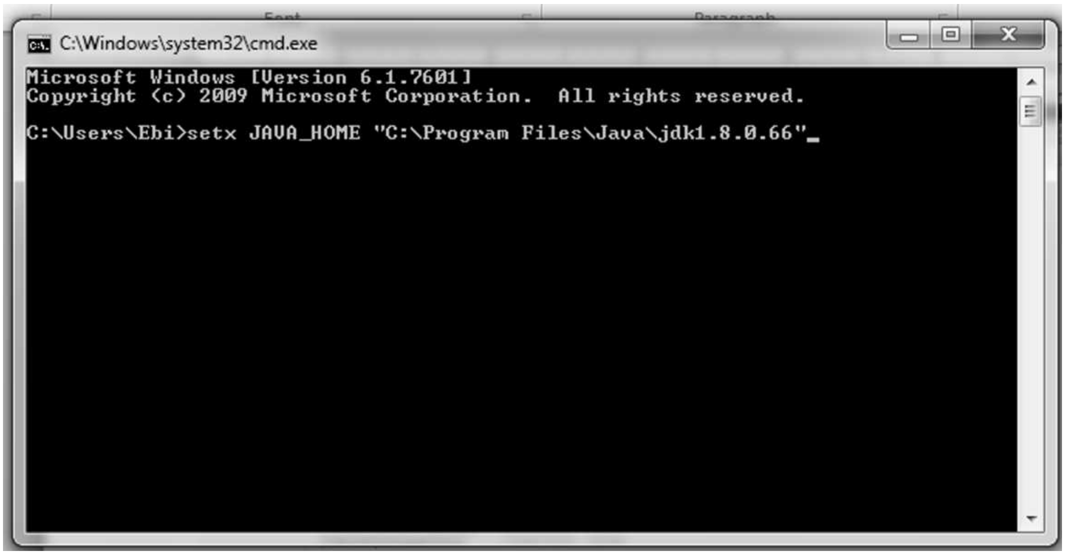
Step 2: Determine where is your JDK is located, by default it is in your C:\program files



```
Name                                    Date modified         Type         Size
CommViewWiFi                            8/13/2015 11:44 AM    File folder
DVD Maker                               11/21/2010 8:47 AM    File folder
EPUB File Reader                        4/26/2016 10:24 AM    File folder
Foxit Software                          1/31/2015 7:14 AM     File folder
Garena Plus                             3/5/2015 6:38 AM      File folder
Google                                  1/31/2015 7:27 AM     File folder
Hi-Rez Studios                          2/17/2016 3:01 PM     File folder
ICBCEbankTools                          6/28/2015 6:11 PM     File folder
InstallShield Installation Information  2/17/2016 2:55 PM     File folder
Internet Explorer                       5/12/2016 9:40 AM     File folder
iPod                                    2/11/2016 7:57 AM     File folder
IQIYI Video                             5/6/2016 5:18 AM      File folder
iTunes                                  2/11/2016 7:59 AM     File folder
Java                                    9/30/2015 4:07 PM     File folder
K-Lite Codec Pack                       1/31/2015 7:29 AM     File folder
kuwo                                    11/17/2015 7:47 PM    File folder
Microsoft Analysis Services             1/31/2015 7:33 AM     File folder
Microsoft Device Heal                        10 PM            File folder
Microsoft Games                              8:47 AM          File folder
Microsoft Kinect Drivers                     0:58 AM          File folder
Microsoft Office                        1/31/2015 7:34 AM     File folder
Microsoft SDKs                          9/20/2015 10:59 AM    File folder
Microsoft Silverlight                   1/13/2016 10:31 AM    File folder
Microsoft SQL Server Compact Edition    1/31/2015 7:34 AM     File folder
Microsoft Synchronization Services      1/31/2015 7:34 AM     File folder
Microsoft.NET                           2/9/2016 5:27 AM      File folder
mp3splt-gtk                             9/30/2015 4:27 PM     File folder

Date created: 1/31/2015 7:33 AM
Size: 37.9 MB
Folders: AS OLEDB

Java    Date modified: 9/30/2015 4:07 PM
File folder
```

Step 3: Select your JDK to use in my case, I will use my JDK_8

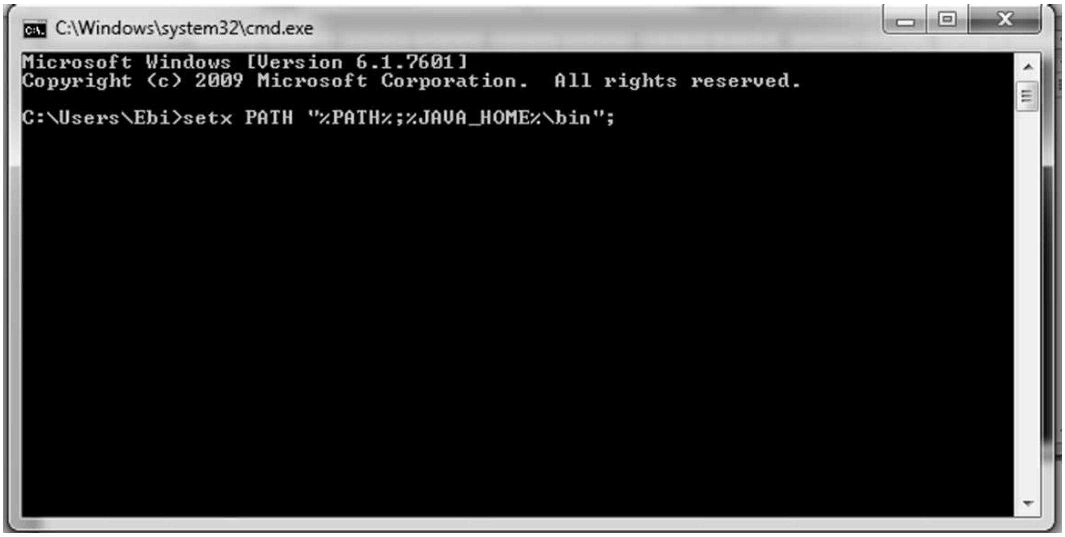
Copy the directory to your clipboard and go to your CMD. And press enter.



C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Ebi>setx JAVA_HOME "C:\Program Files\Java\jdk1.8.0.66"_
```

Step 4: Add it to general PATH



C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Ebi>setx PATH "%PATH%;%JAVA_HOME%\bin";
```

And press enter.

Now go to your spark folder and go to BIN\spark_shell

You have installed spark let's try to use it.

Step 8: Application: WordCount in Scala

Now we will do an example of word count in Scala:

```
text_file = sc.textFile("hdfs://...")
counts = text_file.flatMap(lambda line: line.split(" ")) \
       .map(lambda word: (word, 1)) \
       .reduceByKey(lambda a, b: a + b)
counts.saveAsTextFile("hdfs://...")
```

NOTE: If you are working on a stand-alone Spark:

This `counts.saveAsTextFile("hdfs://...")` command will give you an error of `NullPointerException`.

Solution: `counts.coalesce(1).saveAsTextFile()`

For implementing word cloud we could use R in our spark console:

However, if you click on SparkR straight away you will get an error.

To fix this:

Step 1: Set up the environment variables.

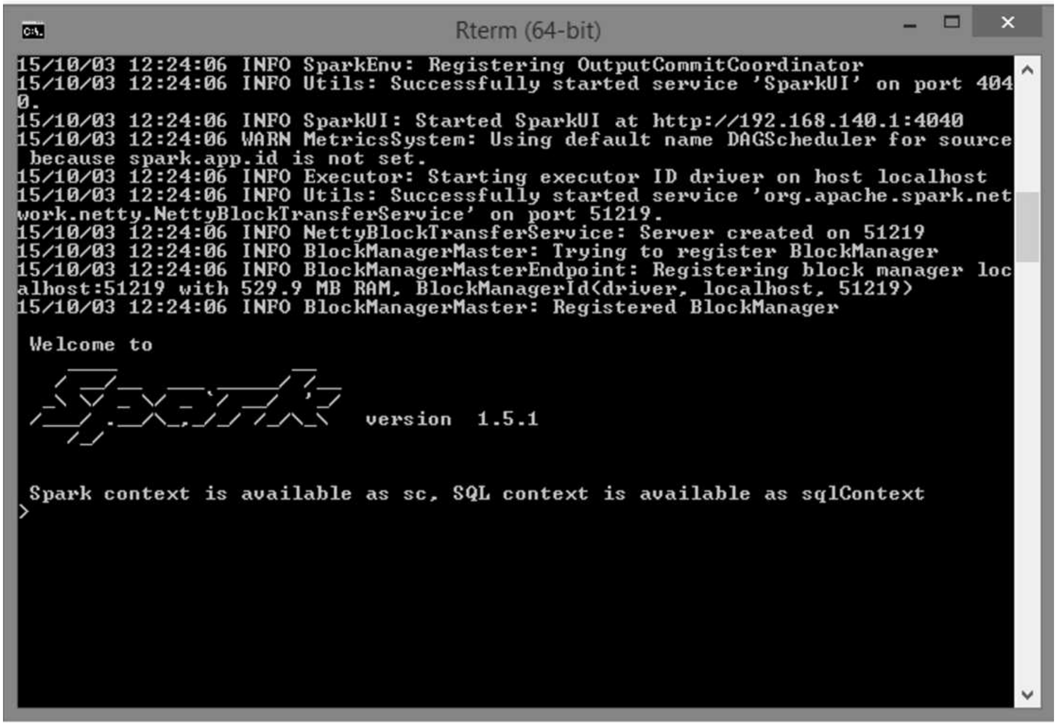
In the PATH Variable add your path : I added -> ;C:\spark-1.5.1-bin-hadoop2.6\spark-1.5.1-bin-hadoop2.6\;C:\spark-1.5.1-bin-hadoop2.6\spark-1.5.1-bin-hadoop2.6\sbin;C:\spark-1.5.1-bin-hadoop2.6\spark-1.5.1-bin-hadoop2.6\bin

Step 2: Install R software and Rstudio. Then add the path of R software path to the PATH variable.

I added this to my existing path -> ;C:\Program Files\R\R-3.2.2\bin\x64\ (Remember each path that you add must be separated by semicolon and no spaces please)

Step 3: Run command prompt as an administrator.

Step 4: Now execute the command > "SparkR" from the command prompt. If successful, you should see message “Spark context is available ... “ as seen below. If you path is not set correctly, you can alternatively navigate to the location where you have downloaded SparkR. In my case (C:\spark-1.5.1-bin-hadoop2.6\spark-1.5.1-bin-hadoop2.6\bin) and execute “SparkR” Command.



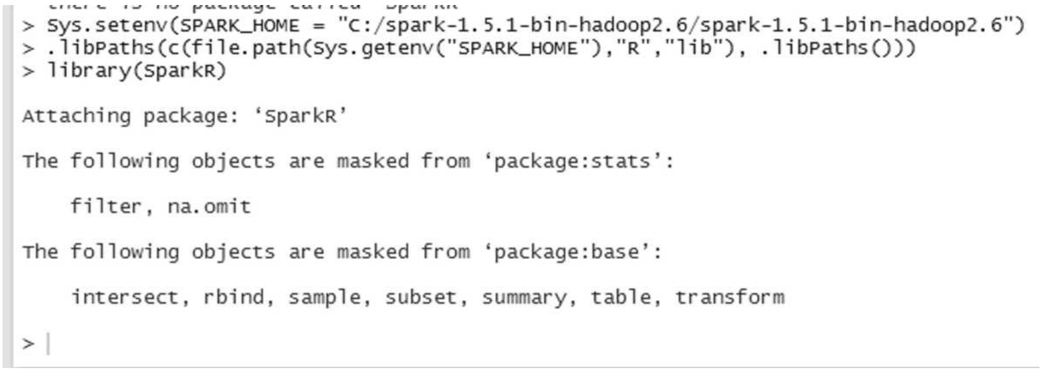
```
Rterm (64-bit)
15/10/03 12:24:06 INFO SparkEnv: Registering OutputCommitCoordinator
15/10/03 12:24:06 INFO Utils: Successfully started service 'SparkUI' on port 404
0.
15/10/03 12:24:06 INFO SparkUI: Started SparkUI at http://192.168.140.1:4040
15/10/03 12:24:06 WARN MetricsSystem: Using default name DAGScheduler for source
 because spark.app.id is not set.
15/10/03 12:24:06 INFO Executor: Starting executor ID driver on host localhost
15/10/03 12:24:06 INFO Utils: Successfully started service 'org.apache.spark.net
work.netty.NettyBlockTransferService' on port 51219.
15/10/03 12:24:06 INFO NettyBlockTransferService: Server created on 51219
15/10/03 12:24:06 INFO BlockManagerMaster: Trying to register BlockManager
15/10/03 12:24:06 INFO BlockManagerMasterEndpoint: Registering block manager loc
alhost:51219 with 529.9 MB RAM, BlockManagerId(driver, localhost, 51219)
15/10/03 12:24:06 INFO BlockManagerMaster: Registered BlockManager

 Welcome to
    ____              __
   / __/__  ___ _____/ /__
  _\ \/ _ \/ _ `/ __/  '_/
 /___/ .__/\_,_/_/ /_/\_\   version  1.5.1
    /_/


 Spark context is available as sc, SQL context is available as sqlContext
>
```

Step 5: Configuration inside the RStudio to connect to Spark!

Execute the below three commands in Rstudio everytime:

```
# Here we are setting up SPARK_HOME environment variable
Sys.setenv(SPARK_HOME = “C:/spark-1.5.1-bin-hadoop2.6/spark-1.5.1-bin-
hadoop2.6”)
# Set the library path
.libPaths(c(file.path(Sys.getenv(“SPARK_HOME”),”R”,”lib”), .libPaths()))
# Loading the SparkR Libary
library(SparkR)
```

If you see the below message then you are all set to start working with SparkR



```
> Sys.setenv(SPARK_HOME = "C:/spark-1.5.1-bin-hadoop2.6/spark-1.5.1-bin-hadoop2.6")
> .libPaths(c(file.path(Sys.getenv("SPARK_HOME"),"R","lib"), .libPaths()))
> library(SparkR)

Attaching package: ‘SparkR’

The following objects are masked from ‘package:stats’:

    filter, na.omit

The following objects are masked from ‘package:base’:

    intersect, rbind, sample, subset, summary, table, transform

> |
```

Now let’s Start Coding in R:

```
lords <- Corpus (DirSource(“temp/”))
```

To see what’s in that corpus, type the command

```
inspect(lords)
```

This should print out contents on the main screen. Next, we need to clean it up. Execute the following in the command line, one line at a time:

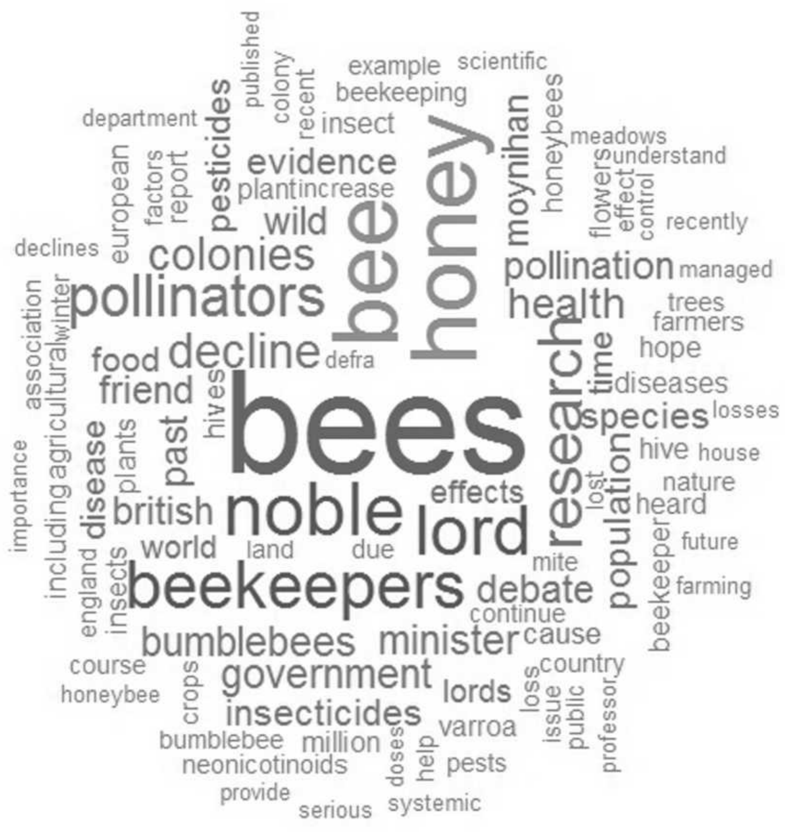
```
lords <- tm_map(lords, stripWhitespace)
lords <- tm_map(lords, tolower)
lords <- tm_map(lords, removeWords, stopwords(“english”))
lords <- tm_map(lords, stemDocument)
```

The **tm_map** function comes with the tm package. The various commands are self-explanatory: strip unnecessary white space, convert everything to lower case (otherwise the wordcloud might highlight capitalised words separately), remove English common words like ‘the’ (so-called ‘stopwords’), and carry out text stemming for the final tidy-up. Depending on what you want to achieve you could also explicitly remove numbers and punctuation with the **removeNumbers** and **removePunctuation** arguments.

It is possible that you may get error messages whilst executing some of the commands, e.g. missing packages. If so install these as outlined above in Step 4, and repeat

If all is well then you should now be ready to create your first wordcloud! Try this:

```
wordcloud(lords, scale=c(5,0.5), max.words=100, random.order=FALSE,rot.
per=0.35, use.r.layout=FALSE, colors=brewer.pal(8, “Dark2”))
```



Additional Resources

Here are some other books and other resources, for learning more about the topics covered in this book.

1. Mayer-Schonberger, Viktor; Cukier, Kenneth (2013). Big Data: A Revolution That Will Transform How We Live, Work, and Think . Houghton Mifflin Harcourt.
2. McKinsey Global Institute Report (2011). Big Data: The Next Frontier For Innovation, Competition, and Productivity. Mckinsey.com
3. Marz, Nathan, and James Warren (2015). Big Data: Principles and Best Practices of Scalable Realtime Data Systems. Manning Publications.
4. Sandy Ryza, Uri Laserson et.al (2014). Advanced-Analytics-with-Spark. OReilley.
5. White, Tom (2014). Mastering Hadoop. OReilley.

Websites:

1. Apache Hadoop resources: https://hadoop.apache.org/docs/r2.7.2/
2. Apache HDFS: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html
3. Hadoop API site: http://hadoop.apache.org/docs/current/api/
4. NoSQL databases: http://nosql-database.org/
5. Apache Spark: http://spark.apache.org/docs/latest/
6. Tutorials on Big Data technologies: https://www.tutorialspoint.com/

Index