**COURSE PACK**

**Hadoop**

**Course Code: 605**

**Course: BCA**

**Semester: VI**

**Year: 2021 - 2022**

**Course Leader: Rajni Dubey**

**Forwarded By**
**Dr. Daljeet Singh Bawa**
**HOD**

**Approved By**
**Dr. Yamini Agarwal**
**Director**

**Department of Management**

**Bharati Vidyapeeth University Institute of Management & Research, New Delhi**

**An ISO 9001:2015 & 14001:2015 Certified Institute**

A-4, Paschim Vihar, New Delhi – 110063 (Ph.: 011-25284396, 25285808 Fax: 011-25286442)

**FOR INTERNAL CIRCULATION ONLY**

**BVIMR SNAPSHOT**

Established in 1992, Bharati Vidyapeeth (Deemed to be University) Institute of Management and Research (BVIMR), New Delhi focuses on imbibing the said values across various stakeholders through adequate creation, inclusion and dissemination of knowledge in management education. The institute has over the past few years emerged in the lead with a vision of Leadership in professional education through innovation and excellence. This excellence is sustained by consistent value enhancement and initiation of value-added academic processes in institutue's academic sytems.

Based on the fabulous architecture and layout on the lines of Nalanda Vishwa Vidyalaya, the institute is a scenic marvel of lush green landscape with modern interiors. The Institute which is ISO 9001:2015 certified is under the ambit of Bharati Vidyapeeth University (BVU), Pune as approved by Govt. of India on the recommendation of UGC under Section 3 of UGC Act vide its letter notification No. F. 9 – 16 / 2004 – U3 dated 25th February, 2005.

Strategically located in West Delhi on the main Rohtak Road, BVIMR, New Delhi has splendid layout on sprawling four acres of plot with 'state-of-art' facilities with all class rooms, Library Labs, Auditorium etc., that are fully air-conditioned. The Institute that has an adjacent Metro station "PaschimVihar (East)", connects the entire Delhi and NCR.

We nurture our learners to be job providers rather than job seekers. This is resorted to by fostering the skill and enhancement of knowledge base of our students through various extracurricular, co-curricular and curricular activities by our faculty, who keep themselves abreast by various research and FDPs and attending Seminars/Conferences. The Alumni has a key role here by inception of SAARTHI Mentorship program who update and create professional environment for learners centric academic ambiance and bridging industry-acdemia gap.

Our faculty make distinctive contribution not only to students but to Academia through publications, seminars, conferences apart from quality education. We also believe in enhancing corporate level interaction including industrial projects, undertaken by our students under continuous guidance of our faculty. These form the core of our efforts which has resulted in being one of the premier institutes of management.

At BVIMR, we are imparting quality education in management at Doctorate, Post Graduate and Under Graduate levels.

*********

**PROFILE OF INSTRUCTORS**

**Rajni Dubey**

She is an Assistant Professor (Visiting Faculty) at Bharti Vidyapeeth Institute of Management and Research and at Delhi Institute of Advanced Studies, GGSIP University. She has completed her Bachelor of Engineering in CSE & M.Tech in IT from ITM University Gwalior and RGTU Bhopal respectively. She is GATE qualified and has 6.5 years of teaching experience at different institutes. She has 07 research papers and 01 book chapter in her credit.

**COURSE PACK –BCA VI Sem**

**TABLE OF CONTENTS**

**COURSE OUTLINE BCA –VI Sem**

| | |
|---|---|
| **Course Title** | Hadoop |
| **Course Code** | 605 |
| **No of credits** | 4 as per syllabus-(Expected no. of hours-40) |
| **Department** | Computer Application |
| **Course Leader** | Rajni Dubey |
| **Email** | Rajni.dubey04@gmail.com |
| **Phone no** | 9582903569 |
| **Course Type** | Core course |
| **Offer in Academic Year** | 2021 -22 , BCA – VI Sem |

**Course Overview**

The world of **Hadoop** and **"Big Data"** can be intimidating - hundreds of different technologies with cryptic names form the Hadoop ecosystem. With this Hadoop COURSE, you'll not only understand what those systems are and how they fit together - but you'll go hands-on and learn how to use them to solve real business problems!

- Understanding Hadoop is a highly valuable skill for anyone working at companies with large amounts of data.
- Almost every large company you might want to work at uses Hadoop in some way, including Amazon, Ebay, Facebook, Google, LinkedIn, IBM, Spotify, Twitter, and Yahoo! And it's not just technology companies that need Hadoop; even the New York Times uses Hadoop for processing images.
- You'll walk away from this course with a real, deep understanding of Hadoop and its associated distributed systems, and you can apply Hadoop to real-world problems.

**The students are expected to review the course readings and the indicated portion of the prescribed text for class discussions prior to attending each session.**

**Course Objectives:**

To introduce learner with HADOOP Tool for Business Intelligence, decision making by doing analysis on the data using HADOOP Tool and also managing the Big Data using HADOOP.
Pre-requisites: Preliminary knowledge of computer, Big Data Analysis and Business Intelligence. Also students must know Core Java, C Programming and Data Structure Languages.
Expected Outcome :

- Good knowledge of HADOOP Tool.
- Knowledge of Decision making using HADOOP analysis on the Big Data
- Hands-on Big Data tools- Hadoop, Pig, Hive, HBase

**CONTENT OF THE COURSE**

**Unit-I: BIG DATA Overview : (5 Hours)**

What is Big Data? What Comes Under Big Data?, Benefits of Big Data, Big Data Technologies Operational vs. Analytical Systems, Big Data Challenges.

**Unit-II: Introduction To HADOOP: (5 Hours)**

Hadoop Architecture, MapReduce, Hadoop Distributed File System, How Does Hadoop Work? Advantages of Hadoop.

**Unit III: HDFS Overview: (6 Hours)**

Features of HDFS, HDFS Architecture, Starting HDFS, Listing Files in HDFS, Inserting Data into HDFS, Retrieving Data from HDFS, Shutting Down the HDFS.

**Unit IV: MAPREDUCE: (6 Hours)**

What is MapReduce? The Algorithm for MapReduce, Inputs and Outputs (Java a Perspective), Analyze different use-cases where MapReduce is used, Differentiate between traditional way and MapReduce way.

**Unit V : Introduction To Hadoop Features: (6 Hours)**

New Big Data Architecture, Introducing HADOOP Features – Apache Hive, Apache HBase, Pig.

**Unit VI: Multi Node Cluster: (6 Hours)**

Multi Node Cluster, Install Java, Creating User Account, Mapping the Nodes, Installing Hadoop, Configuring Hadoop, Start Hadoop Services, Adding New Data Node in the Hadoop Cluster, Removing New Data Node from the Hadoop Cluster.

**Unit VII: Environment Setup: (6 Hours)**

Pre-installation Setup, Installing Java Downloading Hadoop Hadoop Operation Modes Installing Hadoop in Standalone Mode Installing Hadoop in Pseudo Distributed Mode Verifying Hadoop Installation, Implement basic Hadoop commands on terminal.

**Learning Outcomes:**

**After undergoing this course, the student will be able to:**

- Good knowledge of HADOOP Tool.
- Knowledge of Decision making using HADOOP analysis on the Big Data
- Hands-on Big Data tools- Hadoop, Pig, Hive, HBase4.

**Recommended/ Reference Text Books and Resources:**

| | |
|---|---|
| **Text Books** | Big Data- Understanding How Big Data Power Big Business –By Bill Schmarzo |
| **Course Reading** | Hadoop in Action (Manning) Paperback – 1 January 2011.<br><br>Hadoop: The Definitive Guide, 4th Edition Paperback – 1 January 2015. |
| **National / International Journals for Hadoop** | 1. International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)<br>Vol. 2, Issue 4, pp: (97-101), Month: October - December 2014, Available at: www.researchpublish.<br>2. INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) |
| **Online Resources** | https://www.researchgate.net/publication/318681674_A_Review_of_Issues_and_Challenges_with_Big_Data]<br><br>https://www.tutorialspoint.com/hadoop/hadoop_introduction.htm |

**ASSESSMENT CRITERIA**

| **Assessment method** | **Description** | **Weightage** | **Learning outcome** |
|---|---|---|---|
| **End Term Exams** | It will be based on conceptual questions, situation specific application oriented questions and short case studies. Course readings are an integral component of learning in this course. | 60% | End term exam will cover both pre mid-term and post mid-term course coverage. |
| **INTERNAL ASSESSMENT :** | | | |
| **Mid Term Exams**<br>*(First Internal Exams and second Internal Exams)* | It will be based on conceptual questions, situation specific application oriented questions and short case studies. Course readings are an integral component of learning in this course. At least one of the questions will be based on these readings which will not be specified to the students. | 20% | First 3 units for First Internal and last 3 units for second internals |
| **First Internal Exams** | First Internal | 10 % | Unit 1 to 3 for First Internal exam |
| **Second Internal exams** | Second Internal | 10 % | Unit 4 to 6 for second Internal exam |
| **CES Activities (3) Best of Two** | Three CES Activities: Class Test, Moodle quiz and Presentation<br>**Best two CES will be considered if student will appear for all three CES activities** | 10% | Must participate in all CES(Marks of two CES are considered)<br>To recall their subject learning. |
| **CES activities**<br>1- Class Test | A case study will be given. Students will be required to brief the given data by using statistical tools. | Marks 10<br>(Best of two will be considered for 5 marks each CES) | |
| 2-Quiz | It will be based on multiple choice, open ended and fill in the blanks type of questions. | | |
| 3- Case Analysis | There would be a class test for which individual assessment will be done to recall their subject learning. | | |
| **Class participation and Attendance** | Students will be awarded marks for active and constructive participation in class. Students need to fulfil the criteria of 75% attendance overall | 10%<br>(Marks 10) | Enhancing the skill sets for dealing in OB |

**Note: Note:**

**1. All CES activities are mandatory. If any students misses any one CES in that case the weightage of each CES would be 3.33 marks and if a student attempts all 3 CES then his/her best 2 CES will be considered in that case weightage would be 5 marks each.**

**2. The passing marks for Internal is 20 (40%). You need to pass separately in internal and External both.**.

**Failure to submit any assignment as per the required date or activity will lead to penalty and no marks will be awarded for activities missed. So make sure you attend and participate in all CES activities on time.**

**SESSION PLAN**

| **UNIT : 1 BIGDATA Overview** | | | |
|---|---|---|---|
| **Session No.** | **Topic to be discussed** | **Requirements:**<br><br>**Readings/ Assignments / Cases** | **Learning outcome** |
| **1.** | What is Big Data?, What Comes under Big Data | https://www.tutorialspoint.com/hadoop/hadoop_big_data_overview.htm | To develop basic understanding of concept of Big data |
| **2.** | Benefits of Big Data, Big Data Technologies Operational vs. Analytical Systems, | https://www.upgrad.com/blog/what-is-big-data-types-characteristics-benefits-and-examples/#:~:text=It%20refers%20to%20a%20massive,data%20sharing%2C%20and%20data%20visualization.<br><br>https://www.arkatechture.com/blog/the-difference-between-operational-and-analytical-data | To understand the importance Big Data |
| **3.** | Characteristics of Big Data | https://www.sas.com/en_in/insights/big-data/what-is-big-data.html#:~:text=Big%20data%20is%20a%20term,day%2Dto%2Dday%20basis. | To know how to characterized data as a Big Data |
| **4.** | Big Data Challenges. | https://www.researchgate.net/publication/318681674_A_Review_of_Issues_and_Challenges_with_Big_Data] | To understand the challenges & Issues |
| **5.** | Applications & Use-cases of Big Data | https://www.upgrad.com/blog/benefits-and-advantages-of-big-data-analytics-in-business/ | To understand Real world applications |
| **UNIT -2 :** Introduction To HADOOP | | | |
| **Session No.** | **Topic to be discussed** | **Requirements:**<br><br>**Readings/ Assignments / Cases** | **Learning outcome** |
| **6.** | Introduction To HADOOP, Hadoop Architecture | https://www.tutorialspoint.com/hadoop/hadoop_introduction.htm<br><br>https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/ | Understand and be aware of Analytical Tool Hadoop |

| | | | |
|---|---|---|---|
| **7.** | MapReduce with Example | https://www.tutorialspoint.com/hadoop/hadoop_introduction.htm | To understand Hadoop Components |
| **8.** | Concept of Hadoop Distributed File System | https://intellipaat.com/blog/tutorial/hadoop-tutorial/hdfs-operations/ | To Understand HDFS concept |
| **9.** | How Does Hadoop Work? Advantages of Hadoop. | https://www.tutorialspoint.com/hadoop/hadoop_introduction.htm | Hadoop working and Hadoop Advantages |
| **10.** | Hadoop Explanation with Real World Example | https://www.tutorialspoint.com/hadoop/hadoop_introduction.htm | To understand with Real world example |
| CES 1 Class Test From Unit I & II | | | |
| **UNIT - 3 : HDFS Overview:** | | | |
| **11.** | Features of HDFS | https://data-flair.training/blogs/features-of-hadoop-hdfs/ | To develop the understanding of features of HDFS. |
| **12.** | HDFS Architecture | https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/ | To study the architecture and basic components of HDFS. |
| **13.** | HDFS Architecture | https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/ | To study the architecture and basic components of HDFS |
| **14.** | Inserting Data into HDFS, Retrieving Data from HDFS, | https://intellipaat.com/blog/tutorial/hadoop-tutorial/hdfs-operations | To study and understanding basic operations of HDFS. |
| **15.** | Inserting Data into HDFS, Retrieving Data from HDFS, | https://intellipaat.com/blog/tutorial/hadoop-tutorial/hdfs-operations | To study and understanding basic operations of HDFS. |

| | | | |
|---|---|---|---|
| **16.** | Hadoop Commands, Shutting Down the HDFS. | https://intellipaat.com/blog/tutorial/hadoop-tutorial/hdfs-operations | To study and understanding basic operations of HDFS. |
| | **CES 2: Class Test From Unit-III .** | | |
| **UNIT - 4 : MAPREDUCE** | | | |
| **17** | **What is MapReduce** | https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm | To develop a basic concept MapReduce processing |
| **18.** | **The Algorithm for MapReduce** | https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm | To understand how MapReduce Algorithm work |
| **19.** | **Inputs and Outputs (Java a Perspective),** | https://www.javatpoint.com/mapreduce-word-count-example | To understand how MapReduce Algorithm work on input data to perform output data |
| **20.** | **Inputs and Outputs (Java a Perspective)** | https://www.javatpoint.com/mapreduce-word-count-example | To understand how MapReduce Algorithm work on input data to perform output data |
| **21.** | **Analyze different use-cases where MapReduce is used** | **https://www.geeksforgeeks.org/mapreduce-understanding-with-real-life-example/** | To Analyze different cases of MapReduce Algorithm |
| **22.** | **Differentiate between traditional way and MapReduce way.** | https://www.edureka.co/blog/mapreduce-tutorial/ | To study Traditional way and MapReduce way of Processing |
| **UNIT -5 : Introduction To Hadoop Features:** | | | |
| **23.** | New Big Data Architecture | https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/big-data#:~:text=A%20big%20data%20architecture%20is,big%20data%20sources%20at%20rest | To develop the understanding of New Big Data. |

| | | | |
|---|---|---|---|
| **24.** | Introducing HADOOP Features – Apache Hive | https://www.tutorialspoint.com/hive/hive_introduction.htm | To study and understanding the concept of Hadoop Features. |
| **25.** | Apache HBase | https://intellipaat.com/blog/what-is-apache-hbase/ | To study HBase concept. |
| **26** | Pig | https://www.geeksforgeeks.org/introduction-to-apache-pig/#:~:text=Pig%20is%20a%20high%2Dlevel,develop%20the%20data%20analysis%20codes. | To study pig concept |
| **27** | Hive | https://www.tutorialspoint.com/hive/hive_introduction.htm | To study Hive concept. |
| **28** | Differences | https://www.projectpro.io/article/difference-between-pig-and-hive-the-two-key-components-of-hadoop-ecosystem/79 | To understanding and Analyze difference between Hadoop features. |
| **5** | **CES-3 Class Test from Unit-4 and 5** | | |
| **Unit – 6 Multi Node Cluster** | | | |
| **29** | Multi Node Cluster, Install Java | https://www.tutorialspoint.com/hadoop/hadoop_multi_node_cluster.htm | To understand and install java to create platform for Hadoop framework |
| **30** | Creating User Account, Mapping the Nodes, | https://www.tutorialspoint.com/hadoop/hadoop_multi_node_cluster.htm | To understand and study to create new user account. |
| **31** | Installing Hadoop. | https://www.tutorialspoint.com/hadoop/hadoop_multi_node_cluster.htm | To study installation of Hadoop Libraries and Packages |
| **32** | Configuring Hadoop, Node from the Hadoop Cluster | https://www.tutorialspoint.com/hadoop/hadoop_multi_node_cluster.htm | To study Configuration Hadoop. |
| **33** | Start Hadoop Services, | https://www.tutorialspoint.com/hadoop/hadoop_multi_node_cluster.htm | After Installation process and configuration to |

| | | | |
|---|---|---|---|
| | | | study steps of start Hadoop services. |
| **34** | Adding New Data Node in the Hadoop Cluster, Removing New Data | https://www.tutorialspoint.com/hadoop/hadoop_multi_node_cluster.htm | To study, how to manipulate node in Hadoop cluster. |
| **Unit – 7 Environment Setup** | | | |
| **35** | Pre-installation Setup, Installing Java Downloading Hadoop | **https://www.tutorialspoint.com/hadoop/hadoop_environment_setup.htm** | To study steps to installing and download Hadoop. |
| **36** | Hadoop Operation Modes Installing Hadoop in Standalone Mode Installing | https://www.tutorialspoint.com/hadoop/hadoop_enviornment_setup.htm | To study Hadoop Operation in Standalone Mode. |
| **37** | Modes Installing Hadoop in Standalone Mode Installing | https://www.tutorialspoint.com/hadoop/hadoop_enviornment_setup.htm | To study Hadoop Operation in Standalone Mode. |
| **38** | Hadoop in Pseudo Distributed Mode Verifying Hadoop Installation, | https://www.tutorialspoint.com/hadoop/hadoop_enviornment_setup.htm | To study Hadoop Operation in Distributed Mode. |
| **39** | Hadoop in Pseudo Distributed Mode Verifying Hadoop Installation, | https://www.tutorialspoint.com/hadoop/hadoop_enviornment_setup.htm | To study and Analyze modes of operation. |
| **40** | Implement basic Hadoop commands on terminal. | https://data-flair.training/blogs/top-hadoop-hdfs-commands-tutorial/ | To study Basic command on Hadoop terminal. |

## Unit I

**BIGDATA Overview: What is Big Data? What Comes Under Big Data?, Benefits of Big Data, Big Data Technologies Operational vs. Analytical Systems, Big Data Challenges.**

### What is Big Data

Big data is a term that describes large, hard-to-manage volumes of data – both structured and unstructured – that inundate businesses on a day-to-day basis. Big data can be analyzed for insights that improve decisions and give confidence for making strategic business moves.

"Big data" is high-volume, velocity, and variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making."

This definition clearly answers the "What is Big Data?" question – Big Data refers to complex and large data sets that have to be processed and analyzed to uncover valuable information that can benefit businesses and organizations.

However, there are certain basic tenets of Big Data that will make it even simpler to answer what is Big Data:

- It refers to a massive amount of data that keeps on growing exponentially with time.
- It is so voluminous that it cannot be processed or analyzed using conventional data processing techniques.
- It includes data mining, data storage, data analysis, data sharing, and data visualization.
- The term is an all-comprehensive one including data, data frameworks, along with the tools and techniques used to process and analyze the data.

### BIG DATA EXAMPLES

- Personalized e-commerce shopping experiences
- Financial market modelling
- Compiling trillions of data points to speed up cancer research
- Media recommendations from streaming services like Spotify, Hulu and Netflix
- Predicting crop yields for farmers
- Analyzing traffic patterns to lessen congestion in cities
- Data tools recognizing retail shopping habits and optimal product placement
- Big data helping sports teams maximize their efficiency and value
- Recognizing trends in education habits from individual students, schools and districts.

### What Comes Under Big Data?

Big data involves the data produced by different devices and applications. Given below are some of the fields that come under the umbrella of Big Data.

- **Black Box Data** − It is a component of helicopter, airplanes, and jets, etc. It captures voices of the flight crew, recordings of microphones and earphones, and the performance information of the aircraft.
- **Social Media Data** − Social media such as Facebook and Twitter hold information and the views posted by millions of people across the globe.
- **Stock Exchange Data** − The stock exchange data holds information about the 'buy' and 'sell' decisions made on a share of different companies made by the customers.
- **Power Grid Data** − the power grid data holds information consumed by a particular node with respect to a base station.
- **Transport Data** − Transport data includes model, capacity, distance and availability of a vehicle.
- **Search Engine Data** − Search engines retrieve lots of data from different databases.


Thus Big Data includes huge volume, high velocity, and extensible variety of data. The data in it will be of three types.

- **Structured data** − Relational data.
- **Semi Structured data** − XML data.
- **Unstructured data** − Word, PDF, Text, Media Logs.

**Types of Big Data**

Now that we are on track with what is big data, let's have a look at the types of big data:

**Structured**

Structured is one of the types of big data and By structured data, we mean data that can be processed, stored, and retrieved in a fixed format. It refers to highly organized information that can be readily and seamlessly stored and accessed from a database by simple search engine algorithms. For instance, the employee table in a company database will be structured as the employee details, their job positions, their salaries, etc., will be present in an organized manner.

**Unstructured**

Unstructured data refers to the data that lacks any specific form or structure whatsoever. This makes it very difficult and time-consuming to process and analyze unstructured data. Email is an example of unstructured data. Structured and unstructured are two important types of big data.

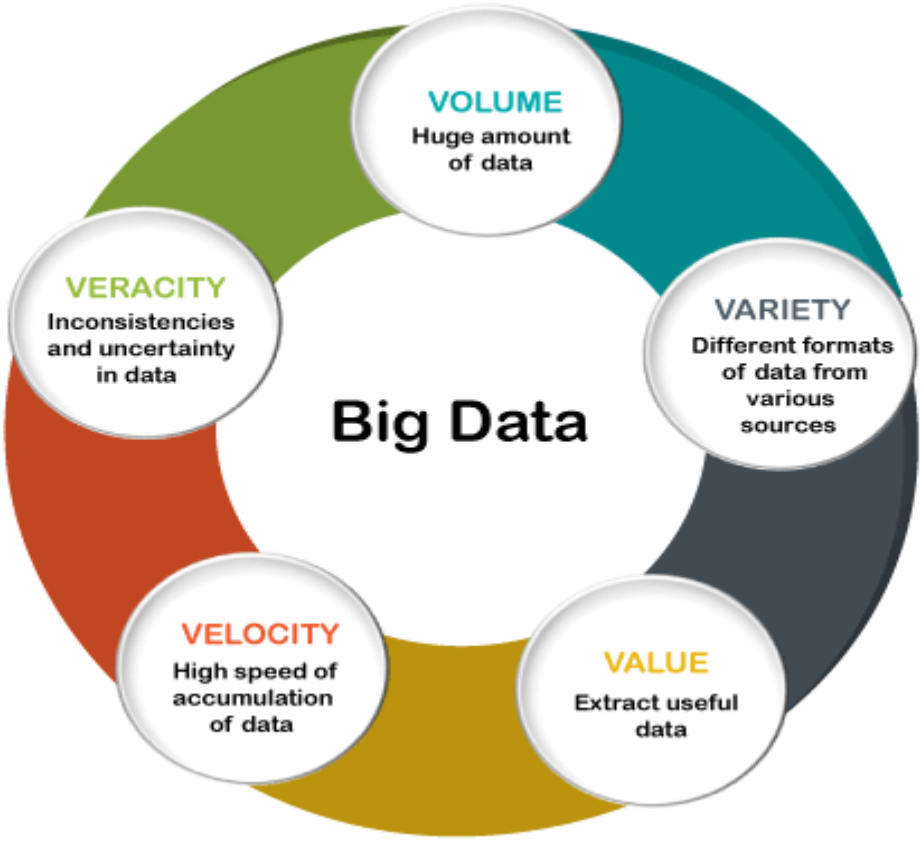
**Semi-structured**

Semi structured is the third type of big data. Semi-structured data pertains to the data containing both the formats mentioned above, that is, structured and unstructured data. To be precise, it refers to the data that although has not been classified under a particular repository (database), yet contains vital information or tags that segregate individual elements within the data.

**History of Big Data**

Big data refers to data that is so large, fast or complex that it's difficult or impossible to process using traditional methods. The act of accessing and storing large amounts of information for analytics has been around for a long time.

But the concept of big data gained momentum in the early 2000s when industry analyst Doug Laney articulated the now-mainstream definition of big data as the three V's:




**Volume**. Organizations collect data from a variety of sources, including transactions, smart (IoT) devices, industrial equipment, videos, images, audio, social media and more. In the past, storing all that data would have been too costly – but cheaper storage using data lakes, Hadoop and the cloud have eased the burden.

**Velocity**. With the growth in the Internet of Things, data streams into businesses at an unprecedented speed and must be handled in a timely manner. RFID tags, sensors and smart meters are driving the need to deal with these torrents of data in near-real time.

**Variety**. Data comes in all types of formats – from structured, numeric data in traditional databases to unstructured text documents, emails, videos, audios, stock ticker data and financial transactions.

At SAS, we consider two additional dimensions when it comes to big data:

**Variability**

In addition to the increasing velocities and varieties of data, data flows are unpredictable – changing often and varying greatly. It's challenging, but businesses need to know when something is trending in social media, and how to manage daily, seasonal and event-triggered peak data loads.

**Veracity**

Veracity refers to the quality of data. Because data comes from so many different sources, it's difficult to link, match, cleanse and transform data across systems. Businesses need to connect and correlate relationships, hierarchies and

multiple data linkages. Otherwise, their data can quickly spiral out of control. [https://www.sas.com/en_in/insights/big-data/what-is-big-data.html]

**Benefits of Big Data**

Big Data can help create pioneering breakthroughs for organizations that know how to use it correctly. Big Data solutions and Big Data Analytics can not only foster data-driven decision making, but they also empower your workforce in ways that add value to your business.

**The benefits of Big Data Analytics and tools are –**

- Data accumulation from multiple sources, including the Internet, social media platforms, online shopping sites, company databases, external third-party sources, etc.
- Real-time forecasting and monitoring of business as well as the market.
- Identify crucial points hidden within large datasets to influence business decisions.
- Promptly mitigate risks by optimizing complex decisions for unforeseen events and potential threats.
- Identify issues in systems and business processes in real-time.
- Unlock the true potential of data-driven marketing.
- Dig in customer data to create tailor-made products, services, offers, discounts, etc.
- Facilitate speedy delivery of products/services that meet and exceed client expectations.
- Diversify revenue streams to boost company profits and ROI.
- Respond to customer requests, grievances, and queries in real-time.
- Foster innovation of new business strategies, products, and services.

Now, we will expand on the most significant advantages of Big Data:

1. **Cost optimization**

   One of the most significant benefits of Big Data tools like Hadoop and Spark is that these offer cost advantages to businesses when it comes to storing, processing, and analyzing large amounts of data. Not just that, Big Data tools can also identify efficient and cost-savvy ways of doing business.

   The logistics industry presents an excellent example to highlight the cost-reduction benefit of Big Data. Usually, the cost of product returns is 1.5 times greater that of actual shipping costs. Big Data Analytics allows companies to minimize product return costs by predicting the likelihood of product returns. They can estimate which products are most likely to be returned, thereby allowing companies to take suitable measures to reduce losses on returns.

2. **Improve efficiency**

   Big Data tools can improve operational efficiency by leaps and bounds. By interacting with customers/clients and gaining their valuable feedback, Big Data tools can amass large amounts of useful customer data. This data can then be analyzed and interpreted to extract meaningful patterns hidden within (customer taste and preferences, pain points, buying behavior, etc.), which allows companies to create personalized products/services.

   Big Data Analytics can identify and analyze the latest market trends, allowing you to keep pace with your competitors in the market. Another benefit of Big Data tools is that they can automate routine processes and tasks. This frees up the valuable time of human employees, which they can devote to tasks that require cognitive skills.

3. **Foster competitive pricing**

   Big Data Analytics facilitates real-time monitoring of the market and your competitors. You can not only keep track of the past actions of your competitors but also see what strategies they are adopting now. Big Data Analytics offers real-time insights that allow you to –

- Calculate and measure the impact of price changes.
- Implement competitive positioning for maximizing company profits.
- Evaluate finances to get a clearer idea of the financial position of your business.
- Implement pricing strategies based on local customer demands, customer purchasing behavior, and competitive market patterns.
- Automate the pricing process of your business to maintain price consistency and eliminate manual errors.

4. **Boost sales and retain customer loyalty**

   Big Data aims to gather and analyze vast volumes of customer data. The digital footprints that customers leave behind reveal a great deal about their preferences, needs, buying behavior, and much more. This customer data offers the scope to design tailor-made products and services to cater to the specific needs of individual customer segments. The higher

the personalization quotient of a business, the more it will attract customers. Naturally, this will boost sales considerably.

Personalization and the quality of product/service also have a positive impact on customer loyalty. If you offer quality products at competitive prices along with personalized features/discounts, customers will keep coming back to you time and again.

5. **Innovate**

   Big Data Analytics and tools can dig into vast datasets to extract valuable insights, which can be transformed into actionable business strategies and decisions. These insights are the key to innovation.

   The insights you gain can be used to tweak business strategies, develop new products/services (that can address specific problems of customers), improve marketing techniques, optimize customer service, improve employee productivity, and find radical ways to expand brand outreach.

6. **Focus on the local environment**

   This is particularly relevant for small businesses that cater to the local market and its customers. Even if your business functions within a constrained setting, it is essential to understand your competitors, what they are offering, and the customers.

   Big Data tools can scan and analyze the local market and offer insights that allow you to see the local trends associated with sellers and customers. Consequently, you can leverage such insights to gain a competitive edge in the local market by delivering highly personalized products/services within your niche, local environment.

**Big Data Technologies Operational vs Analytical Systems**

Operational and Analytical Data Systems are both very similar in how they provide information on your organization, company, or non-profit, but the two are very structurally different, and provide different types of insights.

**Operational Data Systems**

**Operational Data** is exactly what it sounds like - data that is **produced by your organization's day to day operations**. Things like customer, inventory, and purchase data fall into this category. This type of data is pretty straightforward and will generally look the same for most organizations. If you want to know the most up to date information on something - you're using Operational Data! **Operational Data Systems** support high-volume low-latency access, called **Online Transactional Processing** tables, or OLTP, where you want to create, read, update, or delete one piece of data at a time.

**Analytical Data Systems**

**Analytical Data** is a little more complex and will look different for different types of organizations; however, at it's core is an organization's **Operational Data**. Analytical Data is **used to make business decisions**, as opposed to recording the data from actual operational business processes. Examples include grouping customers for market segmentation or changes in purchase volume over time. Every organization will have different questions to answer and different decisions to make, so Analytical Data is definitely not one-size-fits-all by any stretch of the imagination! Analytical Data is best stored in a Data System designed for heavy aggregation, data mining, and ad hoc queries, called an **Online Analytical Processing** system, OLAP, or a Data Warehouse!

**Operational Data Systems**, consisting largely of transactional data, are built for quicker updates. **Analytical Data Systems**, which are intended for decision making, are built for more efficient analysis. Hopefully now you have a better understanding of the difference between Operational and Analytical Data and their corresponding Data Systems! As you can see, both are very important for maintaining and growing an organization, business, or non-profit.

**Big Data Challenges**

In a broad range of application areas, data is being collected at a unique scale. Decisions that previously were based on guesswork, or on the basis of reality, at present now decision to be made using data-driven mathematical models. Such Big Data analysis now drives nearly every aspect of society, including mobile services, retail, manufacturing, financial services, life sciences, and physical sciences.

**Timeliness and heterogeneity**

When there is a lot of information, a great deal of heterogeneity is comfortably tolerated. In fact, the nuance and richness of natural language can provide valuable depth. However, machine analysis algorithms expect homogeneous data, and cannot understand nuance. In consequence, data must be carefully structured prior to data analysis. Even after data cleaning and error correction, some incompleteness and some errors in data are likely to remain. This incompleteness and these errors must be managed during data analysis. Doing this correctly is a challenge.

**Scalability**

Scalability is the major challenge with the big data. You want to be able to scale very rapidly and elastically, whenever and wherever you want. There is a need of effective solution to enable the cost-effective, feasible, scalable storage and processing of large volume of data. Most NoSQL solutions like MongoDB or HBase have their own scaling limitations.

**Performance**

In the world of internet big data must move at extremely high velocities no matter how much you scale or what workloads your database must perform. You need that big data analysis is performed within time constraints as require. The data handling hoops of RDBMS and most NoSQL solutions put a serious drag on performance.

**Continuous Availability**

When you rely on big data to feed your essential, revenue-generating 24/7 business applications, even high availability is not high enough. Your data can never go down. The capabilities of existing system to process streaming information and answer queries in real-time and for thousands of concurrent users are limited. People expect real-time or near real-time responses from the systems they interact with.

**Workload Diversity**

Big data comes in all shapes, colors and sizes. Rigid schemas have no place here; instead you need a more flexible design. You want your technology to fit your data, not the other way around. And you want to be able to do more with all of that data – perform transactions in real-time, run analytics just as fast and find anything you want in an instant from oceans of data, no matter what from that data may take.

**Data Security**

Security is the big concern with the big data. As larger amount of data is processed and transfer among the organizational boundaries and this big data carries some big risks when it contains credit card data, personal ID information and other sensitive assets. Now the challenge is how to protect this sensitive data and how to keep private. Most NoSQL big data platforms have few if any security mechanisms in place to safeguard your big data. Security concerns about data protection are a major obstacle preventing companies from taking full advantage of their data.

**Identifying Right Data**

Identifying the right data from the vast amount of data is the big challenge. Since there are large number of sources such as social networking sites, blogs, different types of content such as articles, comments. Companies have difficulty identifying the right data and determining how to best use it. Therefore, there is the need to find out the rules that will help in identifying the right data Building data-related business cases often means thinking outside of the box and looking for revenue models that are very different from the traditional business.

**Identifying right talent**

Companies are struggling to find the right talent capable of both working with new technologies and of interpreting the data to find meaningful business insights

**Identifying right Platform**

Data access and connectivity can be an obstacle. A majority of data points are not yet connected today, and companies often do not have the right platforms to aggregate and manage the data across the enterprise. In order to address the growing volume of data created as a part of power grid operation

**Identify right architecture**

The technology landscape in the data world is evolving extremely fast. Leveraging data means working with a strong and innovative technology partner that can help create the right IT architecture that can adapt to changes in the landscape in an efficient manner.

**Collaborating across functions and businesses.**
Leveraging big data often means working across functions like IT, engineering, finance and procurement and the ownership of data is fragmented across the organization. To address these organizational challenges means finding new ways of collaborating across functions and businesses.
According to SAS following challenges are outlined in terms of data visualization.

**Meeting the need for speed**
In today's hypercompetitive business environment, companies not only have to find and analyze the relevant data they need, they must find it quickly. Visual-ization helps organizations perform analyses and make decisions much more rapidly, but the challenge is going through the sheer volumes of data and accessing the level of detail needed, all at a high speed. The challenge only grows as the degree of granularity increases. One possible solution is hardware. Some vendors are using increased memory and powerful parallel processing to crunch large volumes of data extremely quickly. Another method is putting data in-memory but using a grid computing approach, where many machines are used to solve a problem. Both approaches allow organizations to explore huge data volumes and gain business insights in near-real time.

**Understanding the data**
It takes a lot of understanding to get data in the right shape so that you can use visualization as part of data analysis. For example, if the data comes from social media content, you need to know who the user is in a general sense – such as a customer using a particular set of products – and understand what it is you're trying to visualize out of the data. Without some sort of context, visualization tools are likely to be of less value to the user.
One solution to this challenge is to have the proper domain expertise in place. Make sure the people analyzing the data have a deep understanding of where the data comes from, what audience will be consuming the data and how that audience will interpret the information.

**Addressing Data Quality**
Even if you can find and analyze data quickly and put it in the proper context for the audience that will be consuming the information, the value of data for decision-making purposes will be jeopardized if the data is not accurate or timely. This is a challenge with any data analysis, but when considering the volumes of information involved in big data projects, it becomes even more pronounced. Again, data visualization will only prove to be a valuable tool if the data quality is assured. To address this issue, companies need to have a data governance or information management process in place to ensure the data is clean. It's always best to have a pro-active method to address data quality issues so problems won't arise later.

**Displaying Meaningful Results**
Plotting points on a graph for analysis becomes difficult when dealing with extremely large amounts of information or a variety of categories of information. For example, imagine you have 10 billion rows of retail SKU data that you're trying to compare. The user trying to view 10 billion plots on the screen will have a hard time seeing so many data points. One way to resolve this is to cluster data into a higher-level view where smaller groups of data become visible. By grouping the data together, or "binning," you can more effectively visualize the data manager.

## UNIT - II

**Introduction to HADOOP: Hadoop Architecture, MapReduce, Hadoop Distributed File System, How Does Hadoop Work? Advantages of Hadoop.**

Apache Hadoop is a collection of open-source software utilities that facilitate using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model.

**Hadoop 1 vs Hadoop 2**

1**. Components:** In Hadoop 1 we have MapReduce but Hadoop 2 has YARN(Yet Another Resource Negotiator) and MapReduce version 2.

| **Hadoop 1** | **Hadoop 2** |
|---|---|
| HDFS | HDFS |
| Map Reduce | YARN / MRv2 |

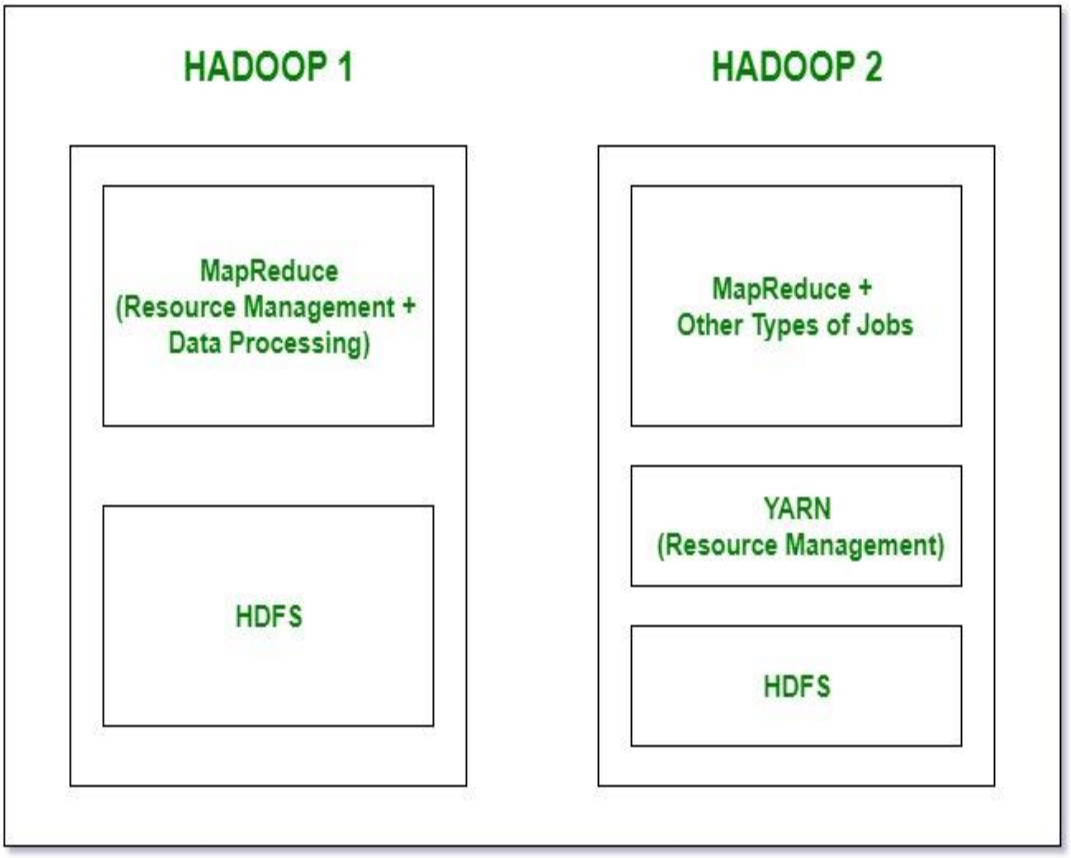
2**. Daemons:**

| **Hadoop 1** | **Hadoop 2** |
|---|---|
| Namenode | Namenode |
| Datanode | Datanode |
| Secondary Namenode | Secondary Namenode |
| Job Tracker | Resource Manager |
| Task Tracker | Node Manager |

**3. Working:**

In Hadoop 1, there is HDFS which is used for storage and top of it, Map Reduce which works as Resource Management as well as Data Processing. Due to this workload on Map Reduce, it will affect the performance.

In Hadoop 2, there is again HDFS which is again used for storage and on the top of HDFS, there is YARN which works as Resource Management. It basically allocates the resources and keeps all the things going on.

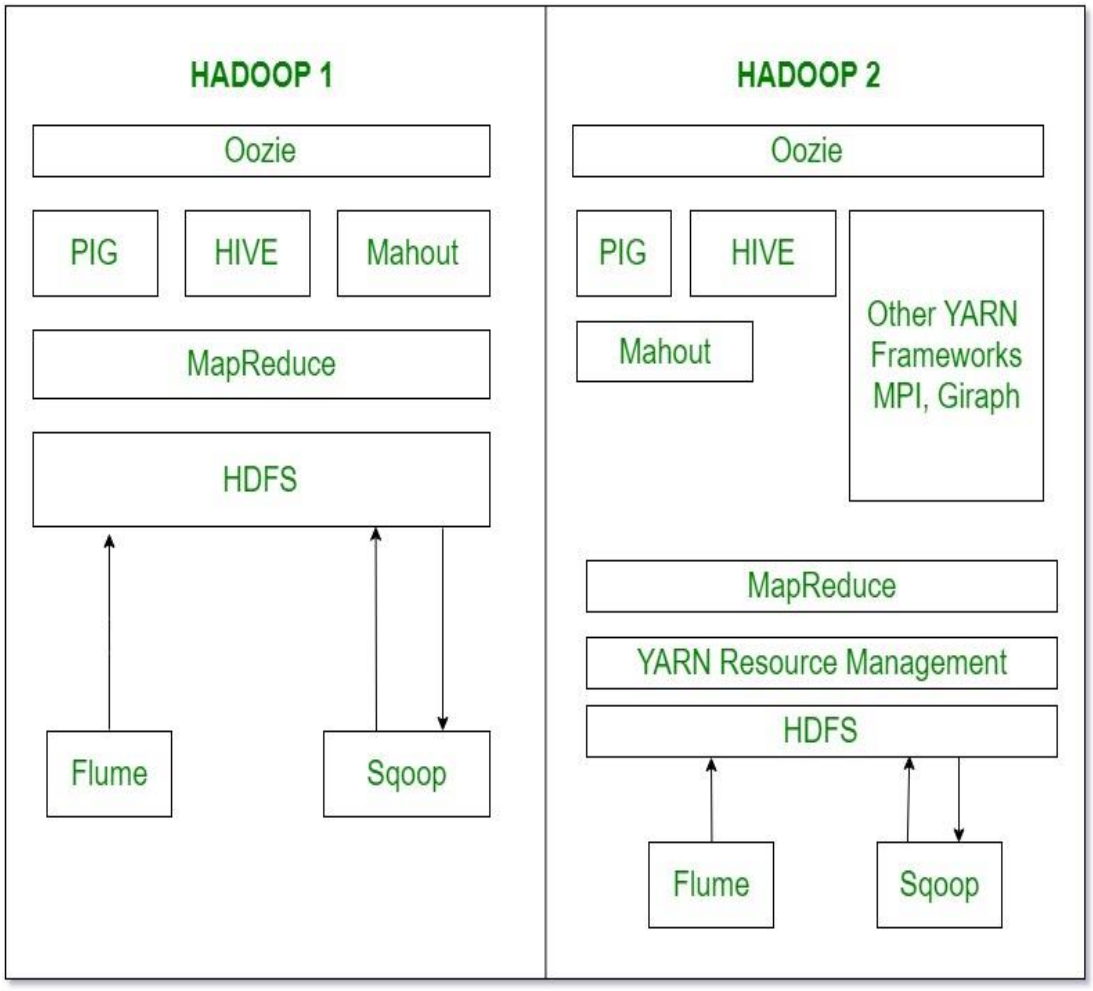



**4. Limitations:**

Hadoop 1 is a Master-Slave architecture. It consists of a single master and multiple slaves. Suppose if master node got crashed then irrespective of your best slave nodes, your cluster will be destroyed. Again for creating that cluster means copying system files, image files, etc. on another system is too much time consuming which will not be tolerated by organizations in today's time.

Hadoop 2 is also a Master-Slave architecture. But this consists of multiple masters (i.e active namenodes and standby namenodes) and multiple slaves. If here master node got crashed then standby master node will take over it. You can make multiple combinations of active-standby nodes. Thus Hadoop 2 will eliminate the problem of a single point of failure.

**5. Ecosystem**




- Oozie is basically Work Flow Scheduler. It decides the particular time of jobs to execute according to their dependency.
- Pig, Hive and Mahout are data processing tools that are working on the top of Hadoop.
- Sqoop is used to import and export structured data. You can directly import and export the data into HDFS using SQL database.
- Flume is used to import and export the unstructured data and streaming data.

**Hadoop Architecture**

**Introduction:**

**The Hadoop Distributed File System (HDFS) is the primary data storage system used by Hadoop applications. HDFS employs a NameNode and DataNode architecture to implement a distributed file system that provides high-performance access to data across highly scalable Hadoop clusters.**

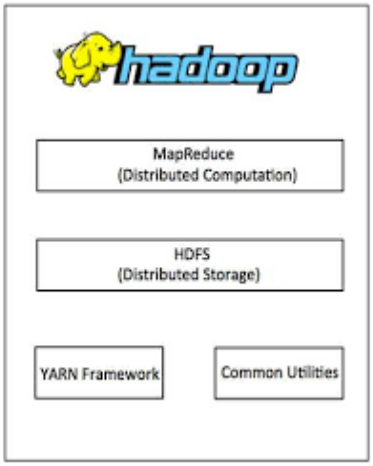
Hadoop itself is an open source distributed processing framework that manages data processing and storage for big data applications. HDFS is a key part of the many Hadoop ecosystem technologies. It provides a reliable means for managing pools of big data and supporting related big data analytics applications.

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment

that provides distributed storage and computation across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage. Hadoop Architecture At its core, Hadoop has two major layers namely –

• Processing/Computation layer (MapReduce), and

• Storage layer (Hadoop Distributed File System)




MapReduce MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework. Hadoop Distributed File System The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets. Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules –

• Hadoop Common − These are Java libraries and utilities required by other Hadoop modules.

• Hadoop YARN − This is a framework for job scheduling and cluster resource management.

What is MapReduce?

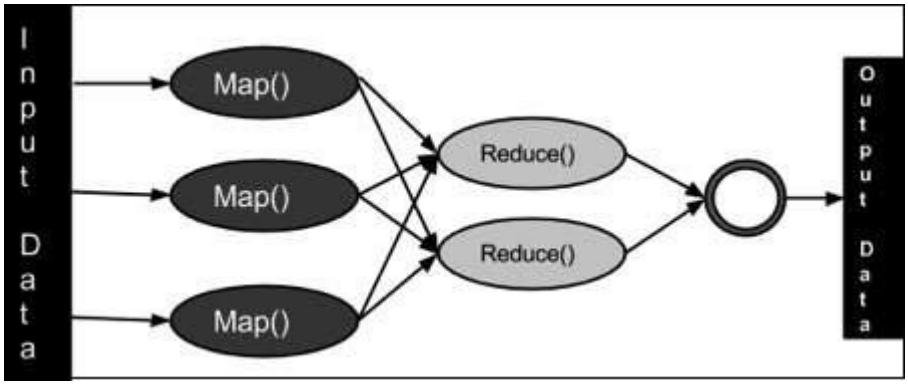
MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into *mappers* and *reducers* is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

The Algorithm

- Generally MapReduce paradigm is based on sending the computer to where the data resides!

- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
    - **Map stage** − The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
    - **Reduce stage** − This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.




**Hadoop Distributed File system**

HDFS is a distributed file system that handles large data sets running on commodity hardware. It is used to scale a single Apache Hadoop cluster to hundreds (and even thousands) of nodes. HDFS is one of the major components of Apache Hadoop, the others being MapReduce and YARN. HDFS should not be confused with or replaced by Apache HBase, which is a column-oriented non-relational database management system that sits on top of HDFS and can better support real-time data needs with its in-memory processing engine.

**The goals of HDFS**

**Fast recovery from hardware failures**

Because one HDFS instance may consist of thousands of servers, failure of at least one server is inevitable. HDFS has been built to detect faults and automatically recover quickly.

**Access to streaming data**

HDFS is intended more for batch processing versus interactive use, so the emphasis in the design is for high data throughput rates, which accommodate streaming access to data sets.

**Accommodation of large data sets**

HDFS accommodates applications that have data sets typically gigabytes to terabytes in size. HDFS provides high aggregate data bandwidth and can scale to hundreds of nodes in a single cluster.

**Portability**

To facilitate adoption, HDFS is designed to be portable across multiple hardware platforms and to be compatible with a variety of underlying operating systems.

**An example of HDFS**

Consider a file that includes the phone numbers for everyone in the United States; the numbers for people with a last name starting with A might be stored on server 1, B on server 2, and so on.

With Hadoop, pieces of this phonebook would be stored across the cluster, and to reconstruct the entire phonebook, your program would need the blocks from every server in the cluster.

To ensure availability if and when a server fails, HDFS replicates these smaller pieces onto two additional servers by default. (The redundancy can be increased or decreased on a per-file basis or for a whole environment; for example, a development Hadoop cluster typically doesn't need any data redundancy.) This redundancy offers multiple benefits, the most obvious being higher availability.

The redundancy also allows the Hadoop cluster to break up work into smaller chunks and run those jobs on all the servers in the cluster for better scalability. Finally, you gain the benefit of data locality, which is critical when working with large data sets.

**How Does Hadoop Work?**

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines. Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs:

- Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M). • These files are then distributed across various cluster nodes for further processing. • HDFS, being on top of the local file system, supervises the processing.
- Blocks are replicated for handling hardware failure.
- Checking that the code was executed successfully.
- Performing the sort that takes place between the map and reduce stages.
  Sending the sorted data to a certain computer.
- Writing the debugging logs for each job.

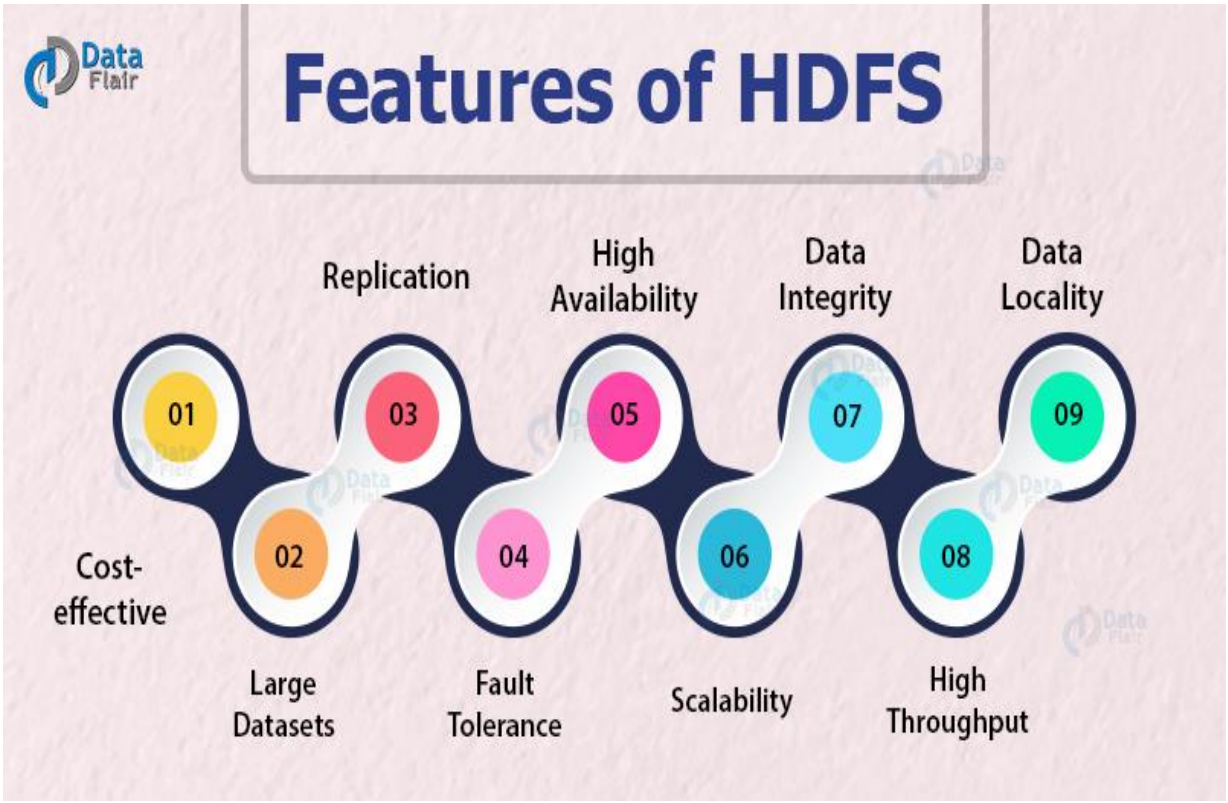
**Advantages of Hadoop**

- Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
- Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
- Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
- Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

**UNIT – III**

**HDFS Overview: Features of HDFS, HDFS Architecture, Starting HDFS, Listing Files in HDFS, Inserting Data into HDFS, Retrieving Data from HDFS, Shutting Down the HDFS.**

**Features of HDFS**




**1. Cost-effective:**

In **HDFS architecture**, the DataNodes, which stores the actual data are **inexpensive commodity hardware**, thus reduces storage costs.

**2. Large Datasets/ Variety and volume of data**

HDFS can store data of any size (ranging from megabytes to petabytes) and of any formats (structured, unstructured).

**3. Replication**

Data Replication is one of the most important and unique features of HDFS. In HDFS replication of data is done to solve the problem of data loss in unfavorable conditions like crashing of a node, hardware failure, and so on.

The data is replicated across a number of machines in the cluster by creating replicas of blocks. The process of replication is maintained at regular intervals of time by HDFS and HDFS keeps creating replicas of user data on different machines present in the cluster.

Hence whenever any machine in the cluster gets crashed, the user can access their data from other machines that contain the blocks of that data. Hence there is no possibility of a loss of user data.

**4. Fault Tolerance and reliability**

HDFS is highly fault-tolerant and reliable. HDFS creates replicas of file blocks depending on the replication factor and stores them on different machines.

If any of the machines containing data blocks fail, other DataNodes containing the replicas of that data blocks are available. Thus ensuring no loss of data and makes the system reliable even in unfavorable conditions.

Hadoop 3 introduced **Erasure Coding** to provide Fault Tolerance. Erasure Coding in HDFS improves storage efficiency while providing the same level of fault tolerance and data durability as traditional replication-based HDFS deployment.

**5. High Availability**

The High availability feature of Hadoop ensures the availability of data even during NameNode or DataNode failure.

Since HDFS creates replicas of data blocks, if any of the DataNodes goes down, the user can access his data from the other DataNodes containing a copy of the same data block.

Also, if the active NameNode goes down, the passive node takes the responsibility of the active NameNode. Thus, data will be available and accessible to the user even during a machine crash.

**6. Scalability**

As HDFS stores data on multiple nodes in the cluster, when requirements increase we can scale the cluster.

There is two scalability mechanism available: Vertical scalability – add more resources (CPU, Memory, Disk) on the existing nodes of the cluster.

Another way is horizontal scalability – Add more machines in the cluster. The horizontal way is preferred since we can scale the cluster from 10s of nodes to 100s of nodes on the fly without any downtime.

**7. Data Integrity**

Data integrity refers to the correctness of data. HDFS ensures data integrity by constantly checking the data against the checksum calculated during the write of the file.

While file reading, if the checksum does not match with the original checksum, the data is said to be corrupted. The client then opts to retrieve the data block from another DataNode that has a replica of that block. The NameNode discards the corrupted block and creates an additional new replica.

**8. High Throughput**

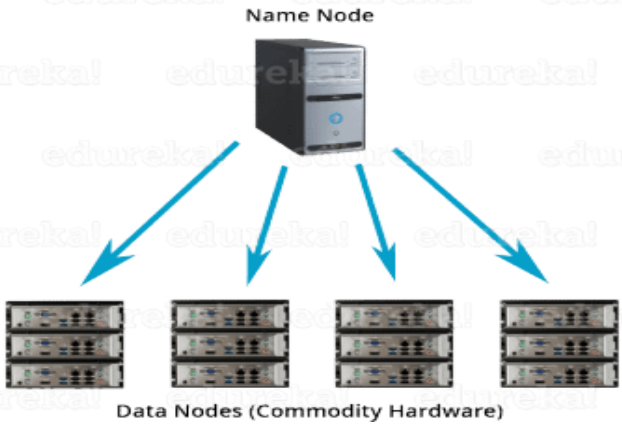
Hadoop HDFS stores data in a distributed fashion, which allows data to be processed parallelly on a cluster of nodes. This decreases the processing time and thus provides high throughput.

**9. Data Locality**

Data locality means moving computation logic to the data rather than moving data to the computational unit.

In the traditional system, the data is brought at the application layer and then gets processed.

But in the present scenario, due to the massive volume of data, bringing data to the application layer degrades the network performance.




In HDFS, we bring the computation part to the Data Nodes where data resides. Hence, with Hadoop HDFS, we are not moving computation logic to the data, rather than moving data to the computation logic. This feature reduces the bandwidth utilization in a system.

**Summary**

In short, after looking at HDFS features we can say that HDFS is a cost-effective, distributed file system. It is highly fault-tolerant. HDFS ensures high availability of the Hadoop cluster.

HDFS provides horizontal scalability. It also checks for data integrity. We can store large volume and variety of data in HDFS.

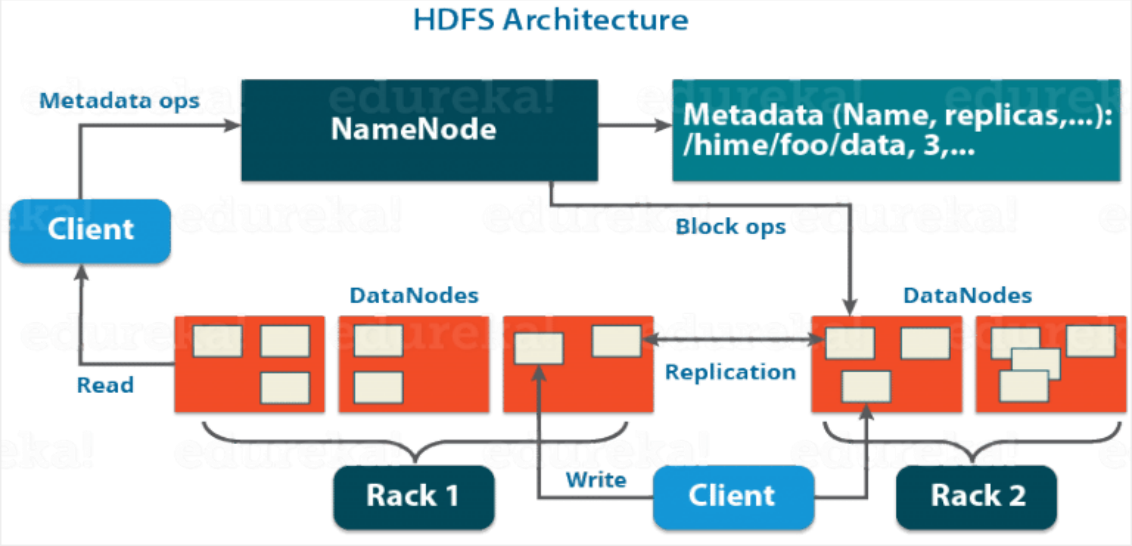
**HDFS Architecture:**

The topics that will be covered in this blog on Apache Hadoop HDFS Architecture are as follows:

- HDFS Master/Slave Topology
- NameNode, DataNode and Secondary NameNode
- What is a block?
- Replication Management
- Rack Awareness
- HDFS Read/Write – Behind the scenes

Apache HDFS or Hadoop Distributed File System is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines. Apache Hadoop HDFS Architecture follows a *Master/Slave Architecture*, where a cluster comprises of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes). HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.

**NameNode:** NameNode is the master node in the Apache Hadoop HDFS Architecture that maintains and manages the blocks present on the DataNodes (slave nodes). NameNode is a very highly available server that manages the File System Namespace and controls access to files by clients. I will be discussing this High Availability feature of Apache Hadoop HDFS in my next blog. The HDFS architecture is built in such a way that the user data never resides on the NameNode. The data resides on DataNodes only.




**Functions of NameNode:**

- It is the master daemon that maintains and manages the DataNodes (slave nodes)
- It records the metadata of all the files stored in the cluster, e.g. The location of blocks stored, the size of the files, permissions, hierarchy, etc. There are two files associated with the metadata:
    - **FsImage:** It contains the complete state of the file system namespace since the start of the NameNode.
    - **EditLogs:** It contains all the recent modifications made to the file system with respect to the most recent FsImage.
- It records each change that takes place to the file system metadata. For example, if a file is deleted in HDFS, the NameNode will immediately record this in the EditLog.
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live.
- It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.
- The NameNode is also responsible to take care of the **replication factor** of all the blocks.

- In **case of the DataNode failure**, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

**DataNode:**

DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is a commodity hardware, that is, a non-expensive system which is not of high quality or high-availability. The DataNode is a block server that stores the data in the local file ext3 or ext4.

**Functions of DataNode:**

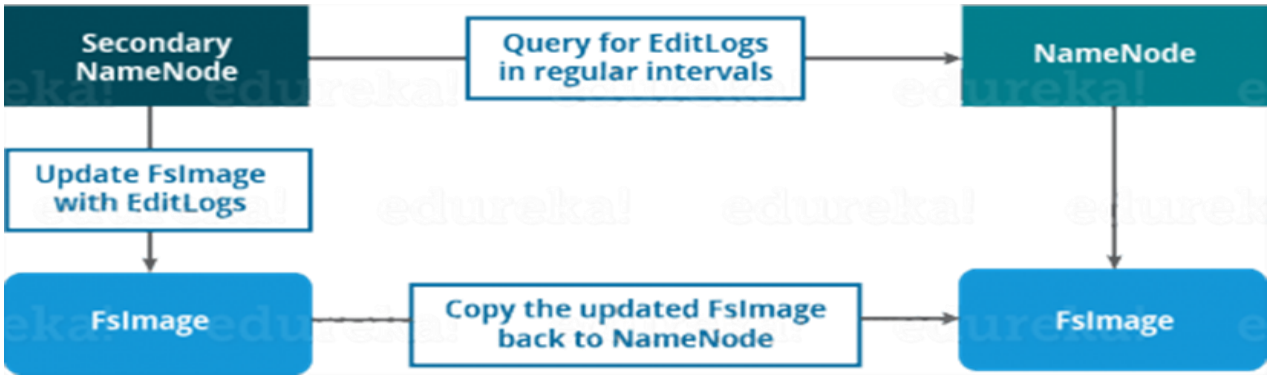
- These are slave daemons or process which runs on each slave machine.
- The actual data is stored on DataNodes.
- The DataNodes perform the low-level read and write requests from the file system's clients.

- They send heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds.

Till now, you must have realized that the NameNode is pretty much important to us. If it fails, we are doomed. But don't worry, we will be talking about how Hadoop solved this single point of failure problem in the next Apache Hadoop HDFS Architecture blog. So, just relax for now and let's take one step at a time.

**Secondary NameNode:**

Apart from these two daemons, there is a third daemon or a process called Secondary NameNode. The Secondary NameNode works concurrently with the primary NameNode as a **helper daemon.** And don't be confused about the Secondary NameNode being a **backup NameNode because it is not**




**Functions of Secondary NameNode:**

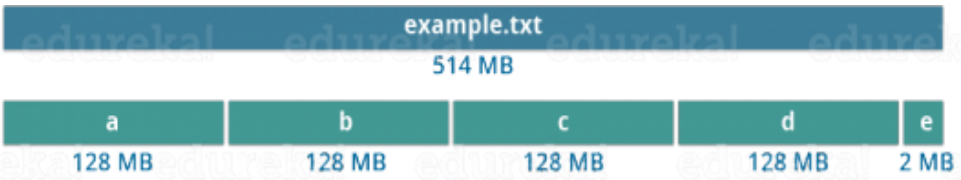
- The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes it into the hard disk or the file system.
- It is responsible for combining the EditLogs with FsImage from the NameNode.
- It downloads the EditLogs from the NameNode at regular intervals and applies to FsImage. The new FsImage is copied back to the NameNode, which is used whenever the NameNode is started the next time.

Hence, Secondary NameNode performs regular checkpoints in HDFS. Therefore, it is also called CheckpointNode.

**Blocks:**

Now, as we know that the data in HDFS is scattered across the DataNodes as blocks.

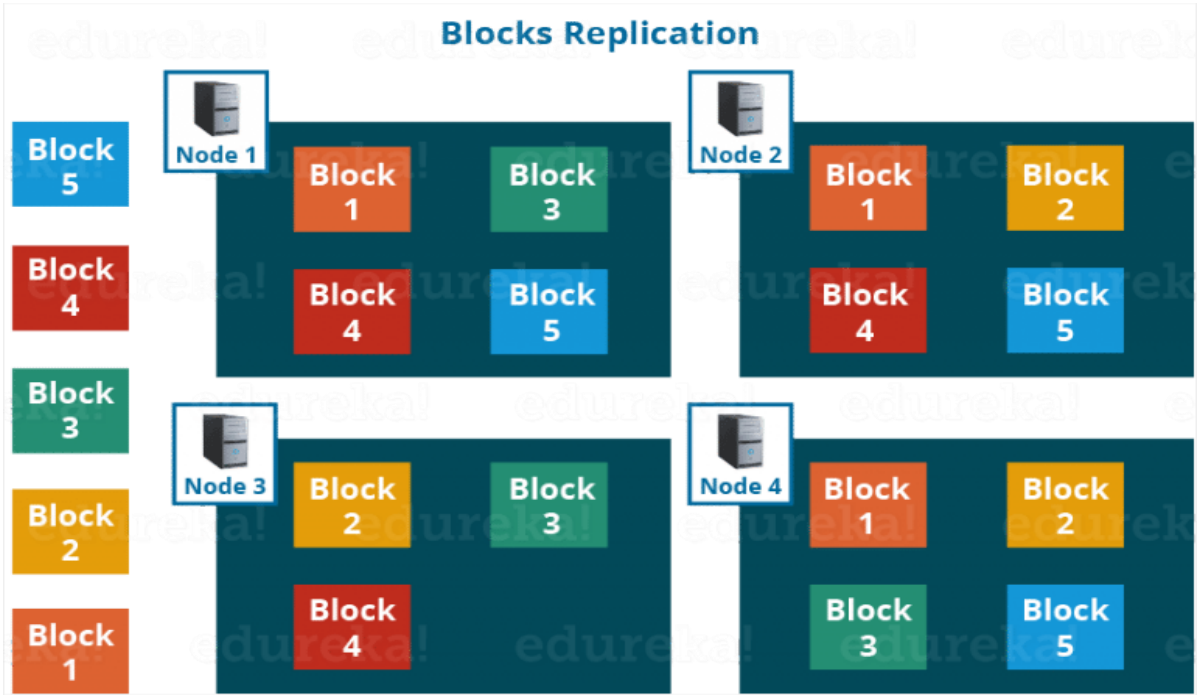
Blocks are the nothing but the smallest continuous location on your hard drive where data is stored. In general, in any of the File System, you store the data as a collection of blocks. Similarly, HDFS stores each file as blocks which are scattered throughout the Apache Hadoop cluster. The default size of each block is 128 MB in Apache Hadoop 2.x (64 MB in Apache Hadoop 1.x) which you can configure as per your requirement.




It is not necessary that in HDFS, each file is stored in exact multiple of the configured block size (128 MB, 256 MB etc.). Let's take an example where I have a file "example.txt" of size 514 MB as shown in above figure. Suppose that we are using the default configuration of blocksize, which is 128 MB. Then, how many blocks will be created? 5, Right. The first four blocks will be of 128 MB. But, the last block will be of 2 MB size only. whenever we talk about HDFS, we talk about huge data sets, i.e. Terabytes and Petabytes of data. So, if we had a block size of let's say of 4 KB, as in Linux file system, we would be having too many blocks and therefore too much of the metadata. So, managing these no. of blocks and metadata will create huge overhead, which is something, we don't want.

## Replication Management:

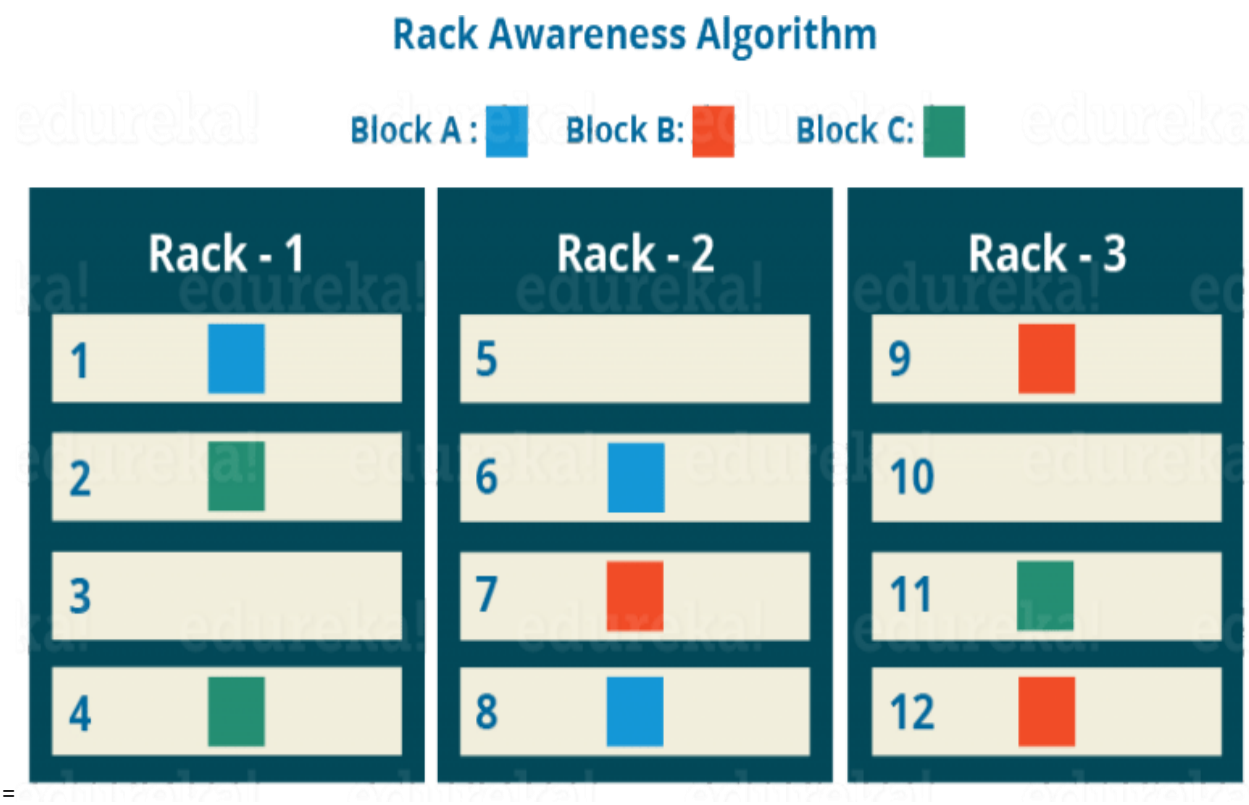
HDFS provides a reliable way to store huge data in a distributed environment as data blocks. The blocks are also replicated to provide fault tolerance. The default replication factor is 3 which is again configurable. So, as you can see in the figure below where each block is replicated three times and stored on different




herefore, if you are storing a file of 128 MB in HDFS using the default configuration, you will end up occupying a space of 384 MB (3*128 MB) as the blocks will be replicated three times and each replica will be residing on a different DataNode.

***Note:*** The NameNode collects block report from DataNode periodically to maintain the replication factor. Therefore, whenever a block is over-replicated or under-replicated the NameNode deletes or add replicas as needed.

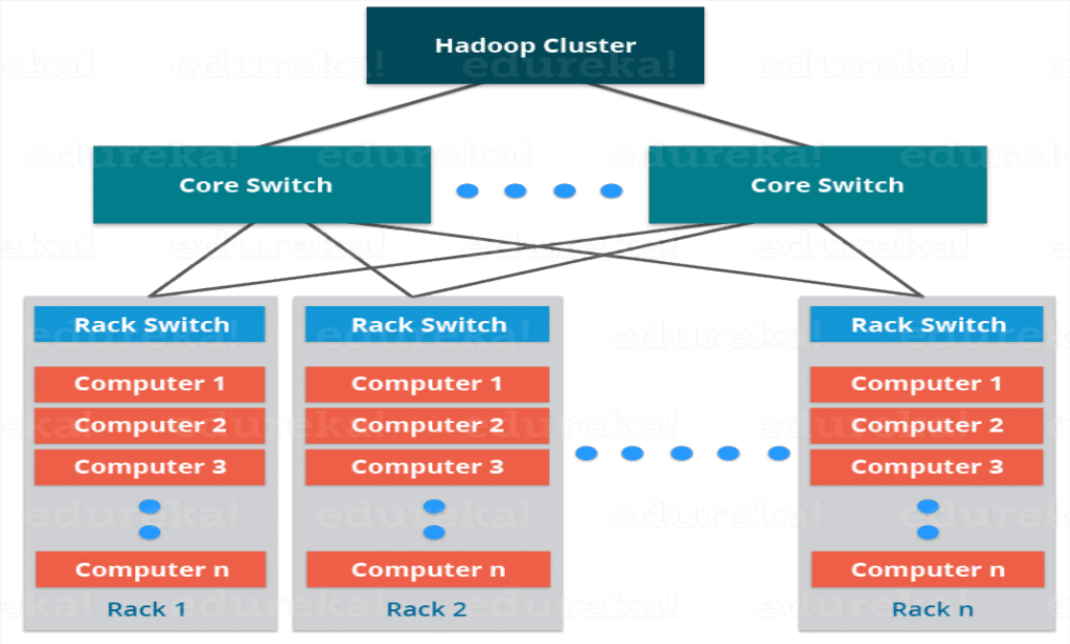
**Rack Awareness:**




=

the NameNode also ensures that all the replicas are not stored on the same rack or a single rack. It follows an in-built Rack Awareness Algorithm to reduce latency as well as provide fault tolerance. Considering the replication factor is 3, the Rack Awareness Algorithm says that the first replica of a block will be stored on a local rack and the next two replicas will be stored on a different (remote) rack but, on a different DataNode within that (remote) rack as shown in the figure above. If you have more replicas, the rest of the replicas will be placed on random DataNodes provided not more than two replicas reside on the same rack, if possible.

This is how an actual Hadoop production cluster looks like. Here, you have multiple racks populated with DataNodes:




**Advantages of Rack Awareness:**

So, now you will be thinking why do we need a Rack Awareness algorithm? The reasons are:

- **To improve the network performance:** The communication between nodes residing on different racks is directed via switch. In general, you will find *greater network bandwidth* between machines in the same rack than the machines residing in different rack. So, the Rack Awareness helps you to have reduce write traffic in between different racks and thus providing a better write performance. Also, you will be gaining increased read performance because you are using the bandwidth of multiple racks.

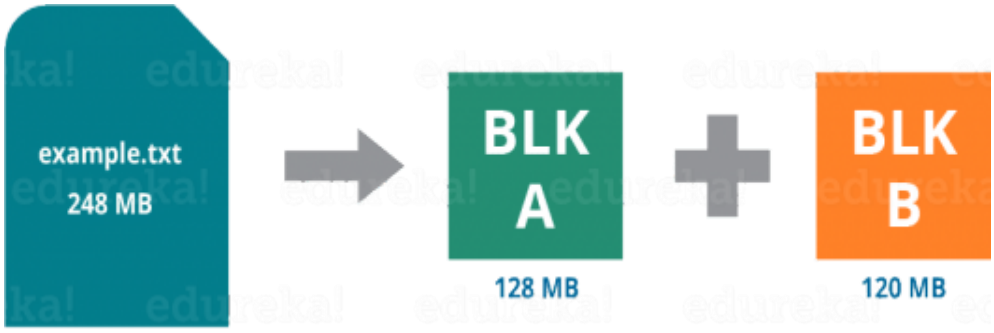
- **To prevent loss of data:** We don't have to worry about the data even if an entire rack fails because of the switch failure or power failure.

**HDFS Read/ Write Architecture:**

HDFS follows Write Once – Read Many Philosophy. So, you can't edit files already stored in HDFS. But, you can append new data by re-opening the file.

**HDFS Write Architecture:**

Suppose a situation where an HDFS client, wants to write a file named “example.txt” of size 248 MB.

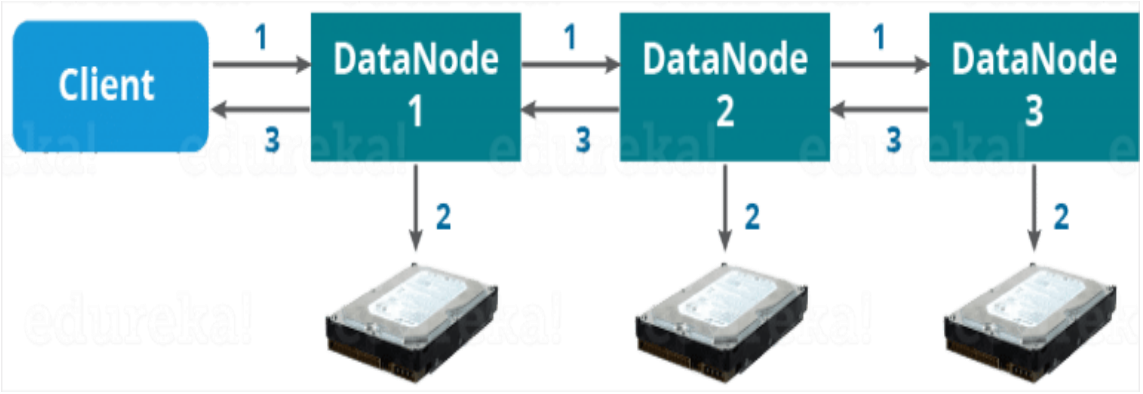



Assume that the system block size is configured for 128 MB (default). So, the client will be dividing the file “example.txt” into 2 blocks – one of 128 MB (Block A) and the other of 120MB (block B).

Now, the following protocol will be followed whenever the data is written into HDFS:

- At first, the HDFS client will reach out to the NameNode for a Write Request against the two blocks, say, Block A & Block B.
- The NameNode will then grant the client the write permission and will provide the IP addresses of the DataNodes where the file blocks will be copied eventually.
- The selection of IP addresses of DataNodes is purely randomized based on availability, replication factor and rack awareness that we have discussed earlier.
- Let’s say the replication factor is set to default i.e. 3. Therefore, for each block the NameNode will be providing the client a list of (3) IP addresses of DataNodes. The list will be unique for each block.
- Suppose, the NameNode provided following lists of IP addresses to the client:
    - For Block A, list A = {IP of DataNode 1, IP of DataNode 4, IP of DataNode 6}
    - For Block B, set B = {IP of DataNode 3, IP of DataNode 7, IP of DataNode 9}

- Each block will be copied in three different DataNodes to maintain the replication factor consistent throughout the cluster.
- Now the whole data copy process will happen in three stages:

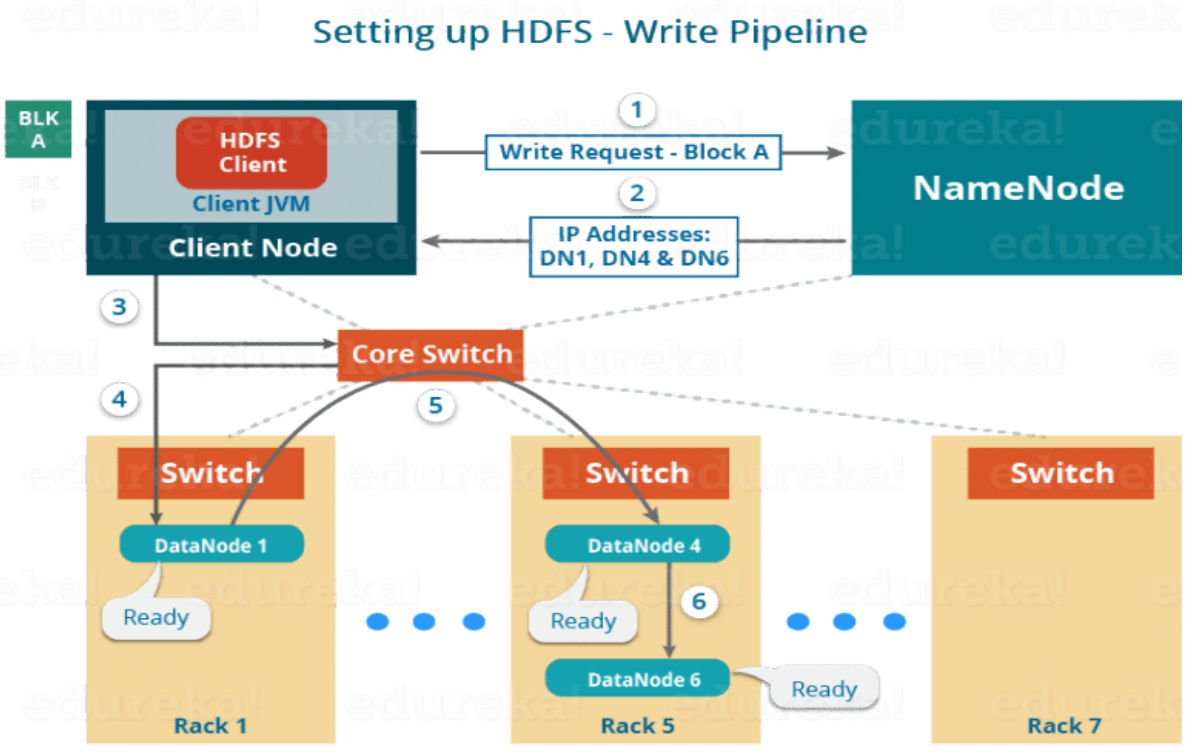



1. Set up of Pipeline
2. Data streaming and replication
3. Shutdown of Pipeline (Acknowledgement stage)

**1. Set up of Pipeline:**

Before writing the blocks, the client confirms whether the DataNodes, present in each of the list of IPs, are ready to receive the data or not. In doing so, the client creates a pipeline for each of the blocks by connecting the individual DataNodes in the respective list for that block. Let us consider Block A. The list of DataNodes provided by the NameNode is:

For Block A, list A = {IP of DataNode 1, IP of DataNode 4, IP of DataNode 6}.

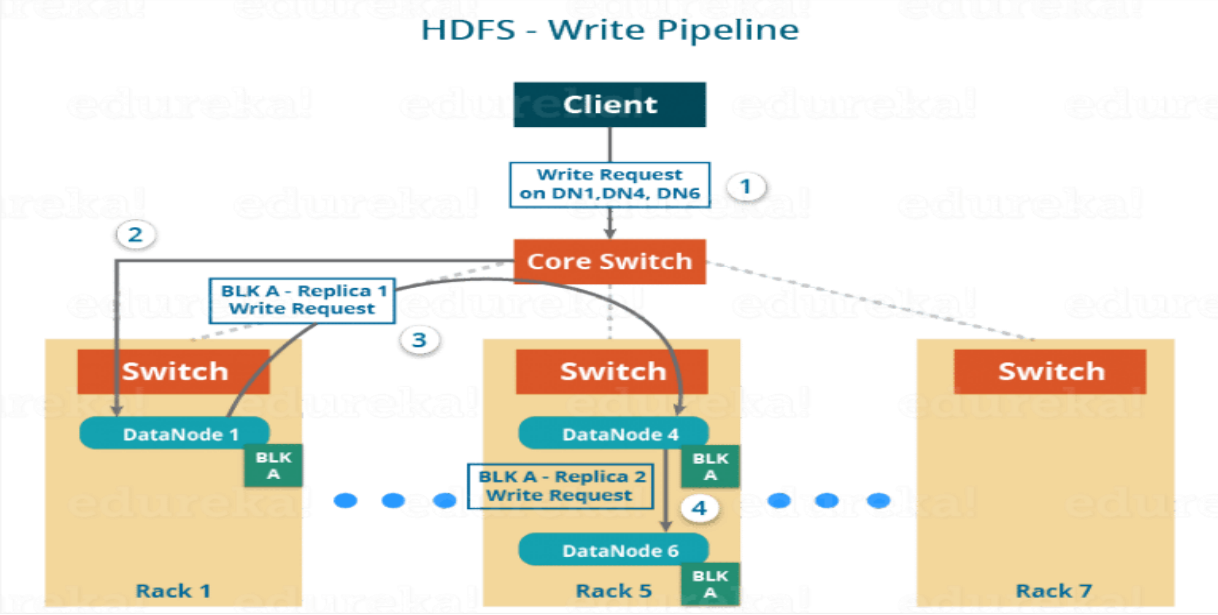


So, for block A, the client will be performing the following steps to create a pipeline:

- The client will choose the first DataNode in the list (DataNode IPs for Block A) which is DataNode 1 and will establish a TCP/IP connection.
- The client will inform DataNode 1 to be ready to receive the block. It will also provide the IPs of next two DataNodes (4 and 6) to the DataNode 1 where the block is supposed to be replicated.
- The DataNode 1 will connect to DataNode 4. The DataNode 1 will inform DataNode 4 to be ready to receive the block and will give it the IP of DataNode 6. Then, DataNode 4 will tell DataNode 6 to be ready for receiving the data.
- Next, the acknowledgement of readiness will follow the reverse sequence, i.e. From the DataNode 6 to 4 and then to 1.
- At last DataNode 1 will inform the client that all the DataNodes are ready and a pipeline will be formed between the client, DataNode 1, 4 and 6.
- Now pipeline set up is complete and the client will finally begin the data copy or streaming process.

**2. Data Streaming:**

As the pipeline has been created, the client will push the data into the pipeline. Now, don't forget that in HDFS, data is replicated based on replication factor. So, here Block A will be stored to three DataNodes as the assumed replication factor is 3. Moving ahead, the client will copy the block (A) to DataNode 1 only. The replication is always done by DataNodes sequentially.




So, the following steps will take place during replication:

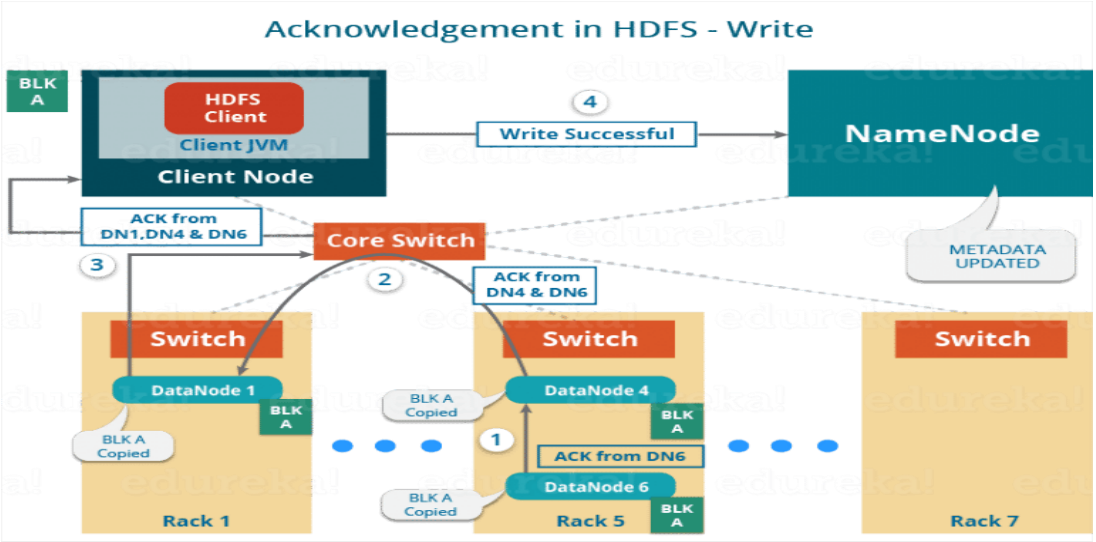
- Once the block has been written to DataNode 1 by the client, DataNode 1 will connect to DataNode 4.
- Then, DataNode1 will push the block in the pipeline and data will be copied to DataNode4.

- Again, DataNode 4 will connect to DataNode 6 and will copy the last replica of the block.

## 3. Shutdown of Pipeline or Acknowledgement stage:

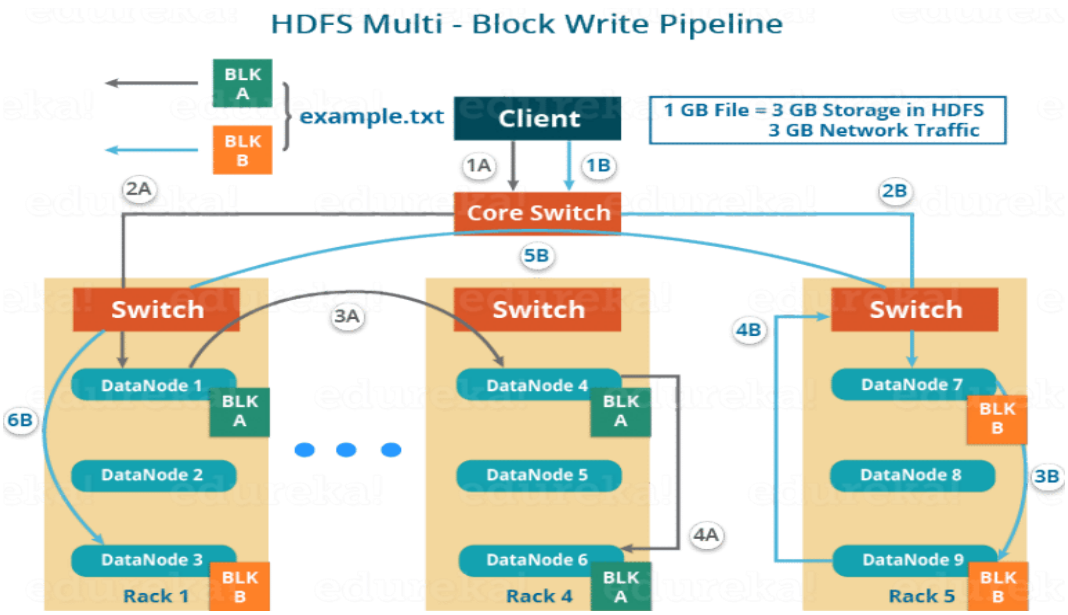
Once the block has been copied into all the three DataNodes, a series of acknowledgements will take place to ensure the client and NameNode that the data has been written successfully. Then, the client will finally close the pipeline to end the TCP session.

in the figure below, the acknowledgement happens in the reverse sequence i.e. from DataNode 6 to 4 and then to 1. Finally, the DataNode 1 will push three acknowledgements (including its own) into the pipeline and send it to the client. The client will inform NameNode that data has been written successfully. The NameNode will update its metadata and the client will shut down the pipeline.




Similarly, Block B will also be copied into the DataNodes in parallel with Block A. So, the following things are to be noticed here:

- The client will copy Block A and Block B to the first DataNode **simultaneously**.
- Therefore, in our case, two pipelines will be formed for each of the block and all the process discussed above will happen in parallel in these two pipelines.
- The client writes the block into the first DataNode and then the DataNodes will be replicating the block sequential




As you can see in the above image, there are two pipelines formed for each block (A and B). Following is the flow of operations that is taking place for each block in their respective pipelines:

- For Block A: 1A -> 2A -> 3A -> 4A
- For Block B: 1B -> 2B -> 3B -> 4B -> 5B -> 6B

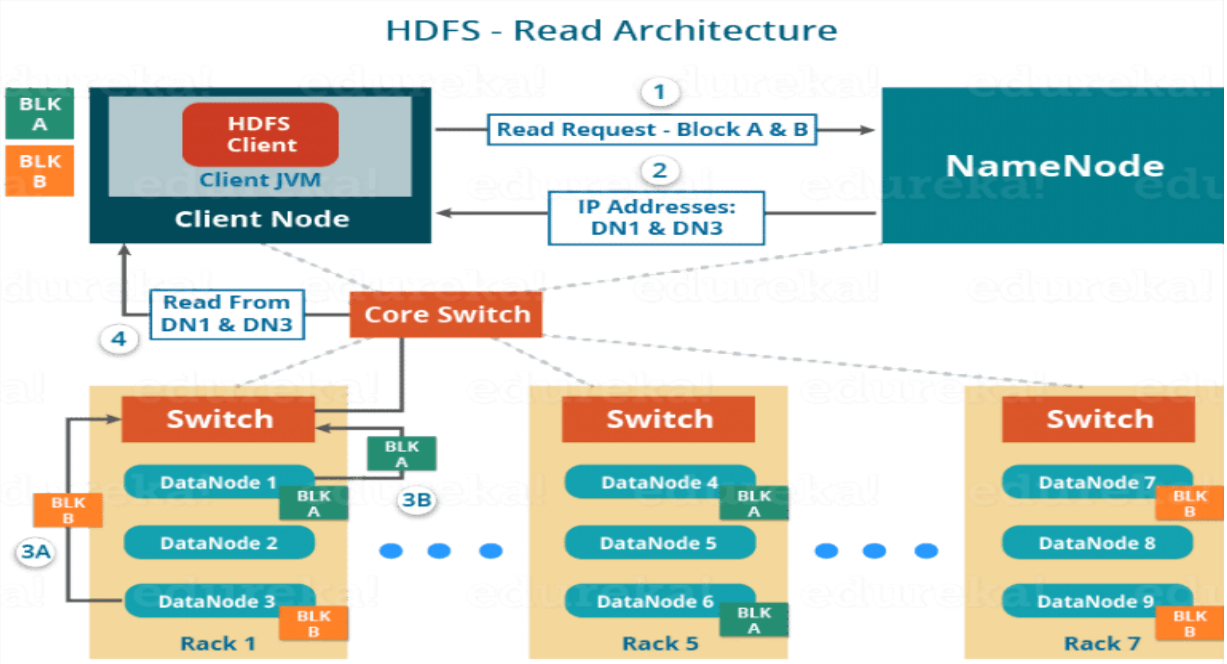
**HDFS Read Architecture:**

HDFS Read architecture is comparatively easy to understand. Let's take the above example again where the HDFS client wants to read the file "example.txt" now.

Now, following steps will be taking place while reading the file:

- The client will reach out to NameNode asking for the block metadata for the file "example.txt".
- The NameNode will return the list of DataNodes where each block (Block A and B) are stored.
- After that client, will connect to the DataNodes where the blocks are stored.
- The client starts reading data parallel from the DataNodes (Block A from DataNode 1 and Block B from DataNode 3).
- Once the client gets all the required file blocks, it will combine these blocks to form a file.

While serving read request of the client, HDFS selects the replica which is closest to the client. This reduces the read latency and the bandwidth consumption. Therefore, that replica is selected which resides on the same rack as the reader node, if possible




**Starting HDFS**

Hadoop file system provides you a privilege as it stores the data in multiple copies. Also, it's a cost-effective solution for any business to store their data efficiently. HDFS Operations acts as the key to open the vaults in which you store the data to be available from remote locations.

Format the configured HDFS file system and then open the namenode (HDFS server) and execute the following command.

```
$ hadoop namenode -format
```

Start the distributed file system and follow the command listed below to start the namenode as well as the data nodes in cluster.

```
$ start-dfs.sh
```

**Read & Write Operations in HDFS**

You can execute almost all operations on Hadoop Distributed File Systems that can be executed on the local file system. You can execute various reading, writing operations such as creating a directory, providing permissions, copying files, updating files, deleting, etc. You can add access rights and browse the file system to get the cluster information like the number of dead nodes, live nodes, spaces used, etc.

**HDFS Operations to Read the file**

To read any file from the HDFS, you have to interact with the NameNode as it stores the metadata about the DataNodes. The user gets a token from the NameNode and that specifies the address where the data is stored.

You can put a read request to NameNode for a particular block location through distributed file systems. The NameNode will then check your privilege to access the DataNode and allows you to read the address block if the access is valid.

```
$ hadoop fs -cat <file
```

**HDFS Operations to write in file**

Similar to the read operation, the HDFS Write operation is used to write the file on a particular address through the NameNode. This NameNode provides the slave address where the client/user can write or add data. After writing on the block location, the slave replicates that block and copies to another slave location using the factor 3 replication. The salve is then reverted back to the client for authentication.

The process for accessing a NameNode is pretty similar to that of a reading operation. Below is the HDFS write commence:

```
bin/hdfs dfs -ls  <path>
```

Listing Files in HDFS

Finding the list of files in a directory and the status of a file using 'ls' command in the terminal. Syntax of ls can be passed to a directory or a filename as an argument which are displayed as follows:

```
$ $HADOOP_HOME/bin/hadoop fs -ls <args>
```

**Inserting Data into HDFS**

Below mentioned steps are followed to insert the required file in the Hadoop file system.

**Step1:** Create an input directory

```
$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/input
```

**Step2:** Use the put command transfer and store the data file from the local systems to the HDFS using the following commands in the terminal.

```
$ $HADOOP_HOME/bin/hadoop fs -put /home/intellipaat.txt /user/input
```

**Step3:** Verify the file using ls command.

```
$ $HADOOP_HOME/bin/hadoop fs -ls /user/input
```

**Retrieving Data from HDFS**

For instance, if you have a file in HDFS called Intellipaat. Then retrieve the required file from the Hadoop file system by carrying out:

**Step1:** View the data from HDFS using the cat command.

```
$ $HADOOP_HOME/bin/hadoop fs -cat /user/output/intellipaat
```

**Step2**: Gets the file from HDFS to the local file system using get command as shown below

```
$ $HADOOP_HOME/bin/hadoop fs -get /user/output/ /home/hadoop_tp/
```

**Shutting Down the HDFS**

Shut down the HDFS files by following the below command

```
$ stop-d
```

## UNIT – IV

**MAPREDUCE : What is MapReduce? The Algorithm for MapReduce, Inputs and Outputs (Java a Perspective), Analyze different use-cases where MapReduce is used, Differentiate between traditional way and MapReduce way.**

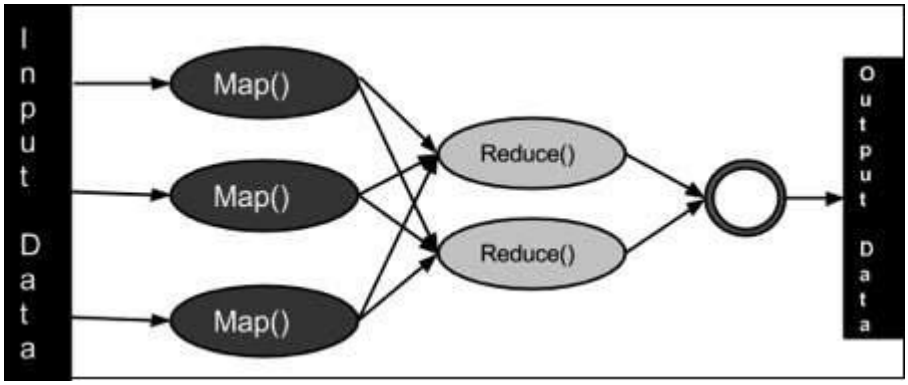
### What is MapReduce?

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The major advantage of MapReduce is that it is easy to scale data processing over multiple computing nodes. Under the MapReduce model, the data processing primitives are called mappers and reducers. Decomposing a data processing application into *mappers* and *reducers* is sometimes nontrivial. But, once we write an application in the MapReduce form, scaling the application to run over hundreds, thousands, or even tens of thousands of machines in a cluster is merely a configuration change. This simple scalability is what has attracted many programmers to use the MapReduce model.

### The Algorithm

- Generally MapReduce paradigm is based on sending the computer to where the data resides!
- MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
    - **Map stage** − The map or mapper's job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
    - **Reduce stage** − This stage is the combination of the **Shuffle** stage and the **Reduce** stage. The Reducer's job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
- During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
- The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
- Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
- After completion of the given tasks, the cluster collects and reduces the data to form an appropriate result, and sends it back to the Hadoop server.



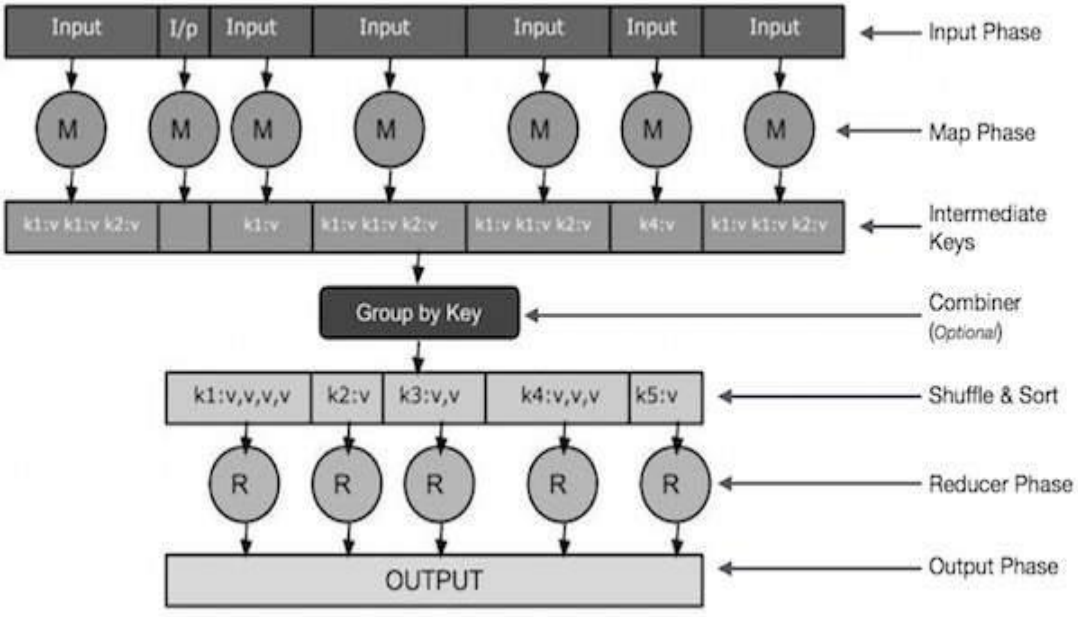

How MapReduce Works?

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key-value pairs).
- The Reduce task takes the output from the Map as an input and combines those data tuples (key-value pairs) into a smaller set of tuples.

The reduce task is always performed after the map job.

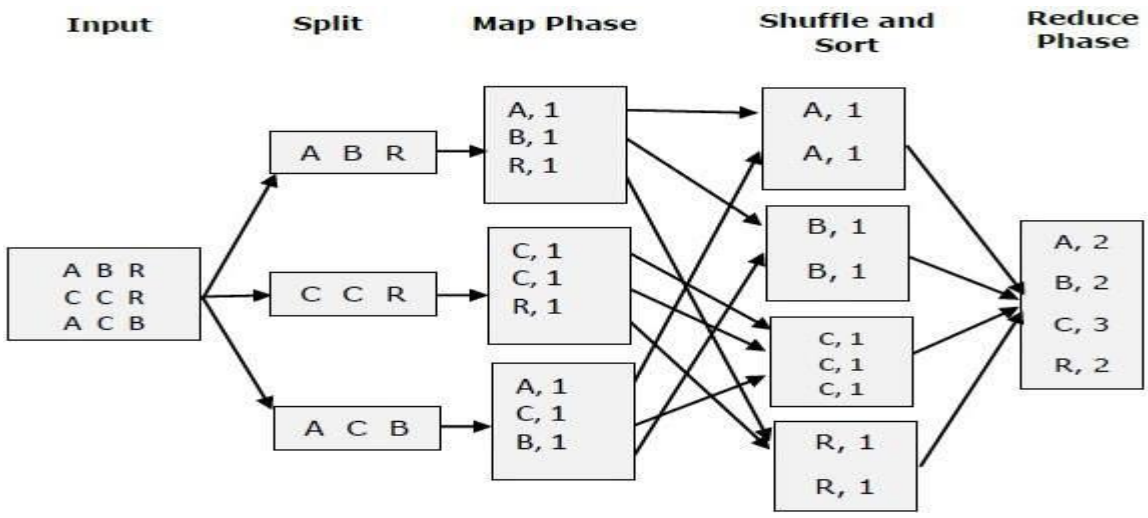
Let us now take a close look at each of the phases and try to understand their significance.




- **Input Phase** − Here we have a Record Reader that translates each record in an input file and sends the parsed data to the mapper in the form of key-value pairs.
- **Map** − Map is a user-defined function, which takes a series of key-value pairs and processes each one of them to generate zero or more key-value pairs.
- **Intermediate Keys** − They key-value pairs generated by the mapper are known as intermediate keys.
- **Combiner** − A combiner is a type of local Reducer that groups similar data from the map phase into identifiable sets. It takes the intermediate keys from the mapper as input and applies a user-defined code to aggregate the values in a small scope of one mapper. It is not a part of the main MapReduce algorithm; it is optional.

- **Shuffle and Sort** − The Reducer task starts with the Shuffle and Sort step. It downloads the grouped key-value pairs onto the local machine, where the Reducer is running. The individual key-value pairs are sorted by key into a larger data list. The data list groups the equivalent keys together so that their values can be iterated easily in the Reducer task.
- **Reducer** − The Reducer takes the grouped key-value paired data as input and runs a Reducer function on each one of them. Here, the data can be aggregated, filtered, and combined in a number of ways, and it requires a wide range of processing. Once the execution is over, it gives zero or more key-value pairs to the final step.
- **Output Phase** − In the output phase, we have an output formatter that translates the final key-value pairs from the Reducer function and writes them onto a file using a record writer.

Let us try to understand the two tasks Map &f Reduce with the help of a small diagram −




**Inputs and Outputs (Java Perspective)**

The MapReduce framework operates on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.

The key and the value classes should be in serialized manner by the framework and hence, need to implement the Writable interface. Additionally, the key classes have to implement the Writable-Comparable interface to facilitate sorting by the framework. Input and Output types of a **MapReduce job** − (Input) <k1, v1> → map → <k2, v2> → reduce → <k3, v3>(Output).

| | **Input** | **Output** |
|---|---|---|
| **Map** | <k1, v1> | list (<k2, v2>) |
| **Reduce** | <k2, list(v2)> | list (<k3, v3>) |

Terminology

- **PayLoad** − Applications implement the Map and the Reduce functions, and form the core of the job.
- **Mapper** − Mapper maps the input key/value pairs to a set of intermediate key/value pair.
- **NamedNode** − Node that manages the Hadoop Distributed File System (HDFS).

- **DataNode** − Node where data is presented in advance before any processing takes place.
- **MasterNode** − Node where JobTracker runs and which accepts job requests from clients.
- **SlaveNode** − Node where Map and Reduce program runs.
- **JobTracker** − Schedules jobs and tracks the assign jobs to Task tracker.
- **Task Tracker** − Tracks the task and reports status to JobTracker.
- **Job** − A program is an execution of a Mapper and Reducer across a dataset.
- **Task** − An execution of a Mapper or a Reducer on a slice of data.
- **Task Attempt** − A particular instance of an attempt to execute a task on a SlaveNode.

**Example Scenario**

**MapReduce Word Count Example**

In MapReduce word count example, we find out the frequency of each word. Here, the role of Mapper is to map the keys to the existing values and the role of Reducer is to aggregate the keys of common values. So, everything is represented in the form of Key-value pair.

**Input** = HDFS is a storage unit of Hadoop
MapReduce is a processing tool of Hadoop

**Output** = HDFS 1
Hadoop 2
MapReduce 1
A 2
Is 2
Processing 1
Storage 1
Tool 1
Unit 1

**Pre-requisite**

**Java Installation -** Check whether the Java is installed or not using the following command.

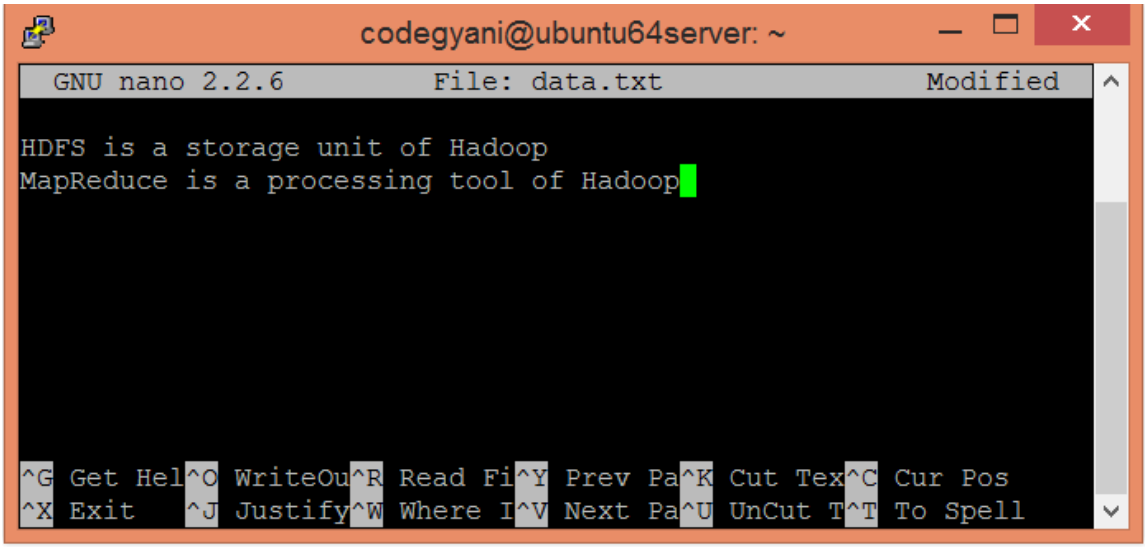
java -version

**Hadoop Installation -** Check whether the Hadoop is installed or not using the following command.

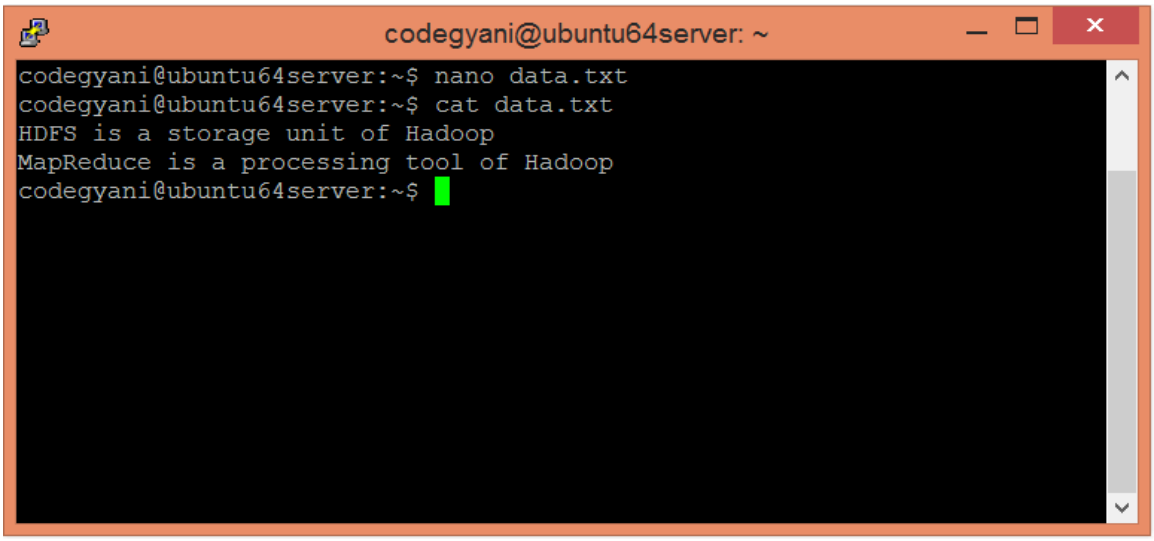
hadoop version

**Steps to execute MapReduce word count example**

- Create a text file in your local machine and write some text into it.
  $ nano data.txt

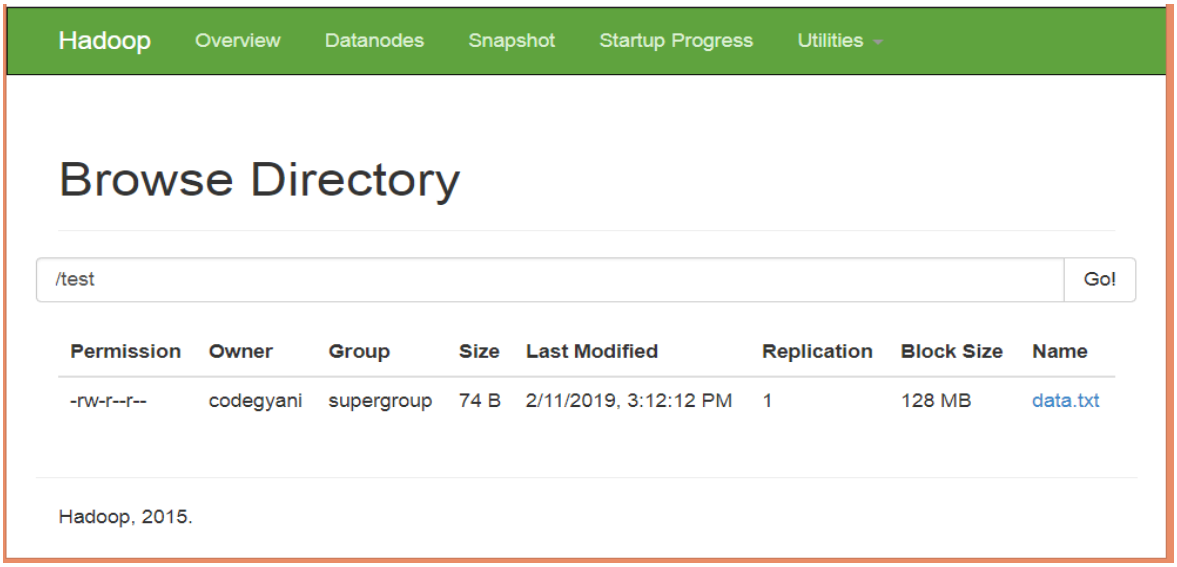



- Check the text written in the data.txt file.
  $ cat data.txt




In this example, we find out the frequency of each word exists in this text file.

- Create a directory in HDFS, where to kept text file.
  $ hdfs dfs -mkdir /test
- Upload the data.txt file on HDFS in the specific directory.
  $ hdfs dfs -put /home/codegyani/data.txt /test




- Write the MapReduce program using eclipse.

**File: WC_Mapper.java**

```
1. package com.javatpoint;
2.
3. import java.io.IOException;
4. import java.util.StringTokenizer;
5. import org.apache.hadoop.io.IntWritable;
6. import org.apache.hadoop.io.LongWritable;
7. import org.apache.hadoop.io.Text;
8. import org.apache.hadoop.mapred.MapReduceBase;
9. import org.apache.hadoop.mapred.Mapper;
10. import org.apache.hadoop.mapred.OutputCollector;
11. import org.apache.hadoop.mapred.Reporter;
12. public class WC_Mapper extends MapReduceBase implements Mapper<LongWritable,Text,Text,I
    ntWritable>{
13.     private final static IntWritable one = new IntWritable(1);
14.     private Text word = new Text();
15.     public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,
16.             Reporter reporter) throws IOException{
17.         String line = value.toString();
18.         StringTokenizer  tokenizer = new StringTokenizer(line);
19.         while (tokenizer.hasMoreTokens()){
20.             word.set(tokenizer.nextToken());
```

```
21.             output.collect(word, one);
22.         }
23.     }
24.
25. }
```

**File: WC_Reducer.java**

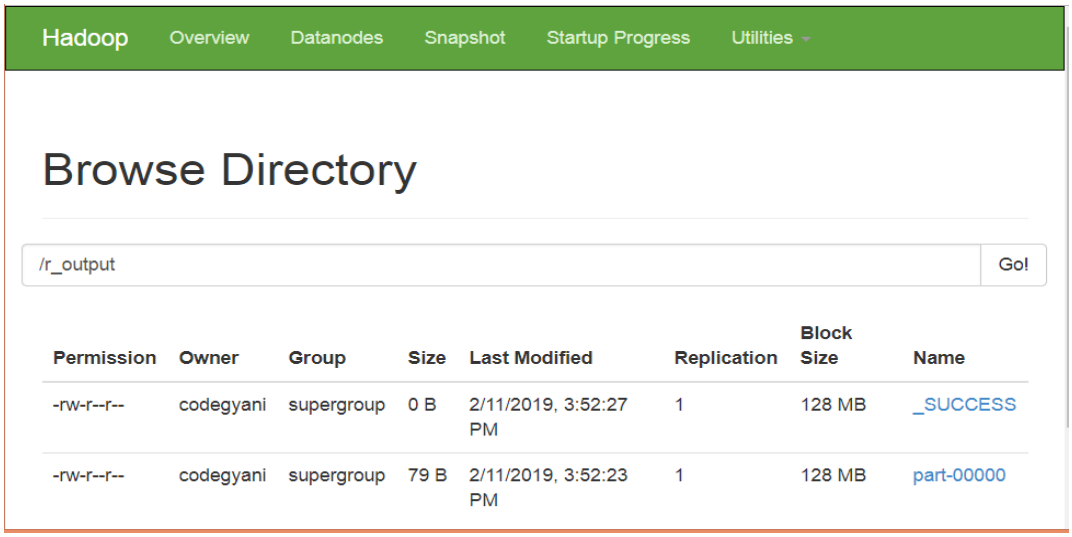
```
1.  package com.javatpoint;
2.      import java.io.IOException;
3.      import java.util.Iterator;
4.      import org.apache.hadoop.io.IntWritable;
5.      import org.apache.hadoop.io.Text;
6.      import org.apache.hadoop.mapred.MapReduceBase;
7.      import org.apache.hadoop.mapred.OutputCollector;
8.      import org.apache.hadoop.mapred.Reducer;
9.      import org.apache.hadoop.mapred.Reporter;
10.
11.     public class WC_Reducer  extends MapReduceBase implements Reducer<Text,IntWritable,Text,I
    ntWritable> {
12.      public void reduce(Text key, Iterator<IntWritable> values,OutputCollector<Text,IntWritable>
    output,
13.      Reporter reporter) throws IOException {
14.     int sum=0;
15.     while (values.hasNext()) {
16.     sum+=values.next().get();
17.     }
18.     output.collect(key,new IntWritable(sum));
19.     }
20.     }
```

**File: WC_Runner.java**

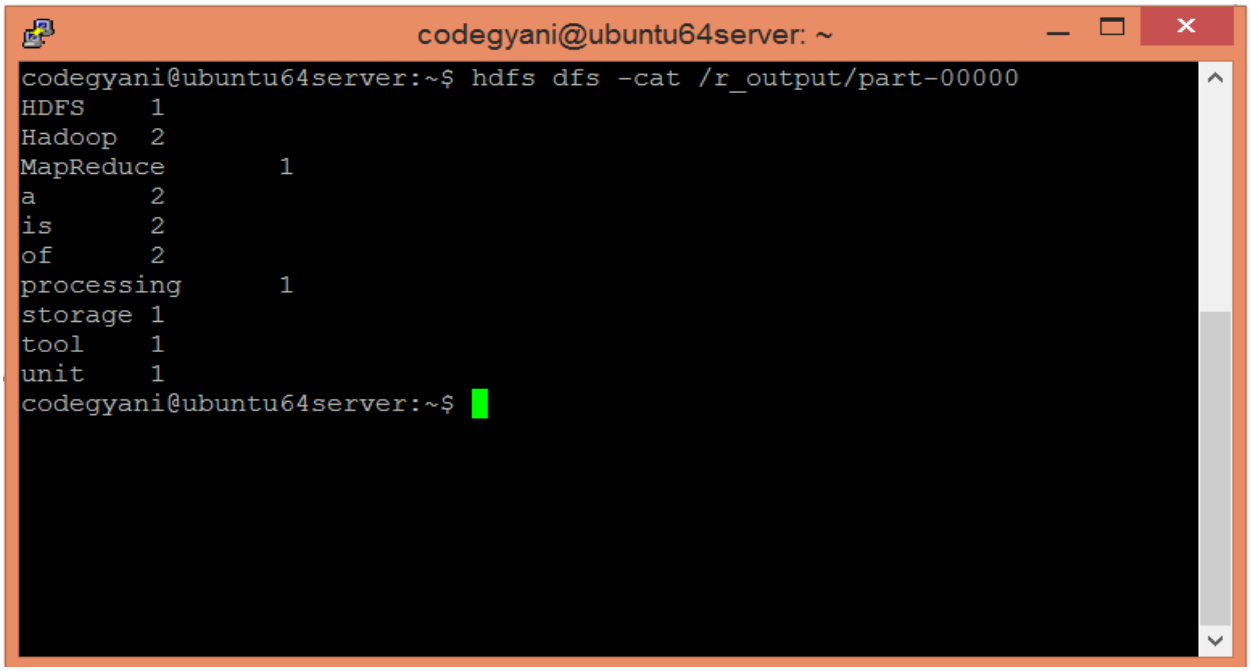
```
1.  package com.javatpoint;
2.
3.      import java.io.IOException;
4.      import org.apache.hadoop.fs.Path;
5.      import org.apache.hadoop.io.IntWritable;
6.      import org.apache.hadoop.io.Text;
```

```
7.  import org.apache.hadoop.mapred.FileInputFormat;
8.  import org.apache.hadoop.mapred.FileOutputFormat;
9.  import org.apache.hadoop.mapred.JobClient;
10. import org.apache.hadoop.mapred.JobConf;
11. import org.apache.hadoop.mapred.TextInputFormat;
12. import org.apache.hadoop.mapred.TextOutputFormat;
13. public class WC_Runner {
14.     public static void main(String[] args) throws IOException{
15.         JobConf conf = new JobConf(WC_Runner.class);
16.         conf.setJobName("WordCount");
17.         conf.setOutputKeyClass(Text.class);
18.         conf.setOutputValueClass(IntWritable.class);
19.         conf.setMapperClass(WC_Mapper.class);
20.         conf.setCombinerClass(WC_Reducer.class);
21.         conf.setReducerClass(WC_Reducer.class);
22.         conf.setInputFormat(TextInputFormat.class);
23.         conf.setOutputFormat(TextOutputFormat.class);
24.         FileInputFormat.setInputPaths(conf,new Path(args[0]));
25.         FileOutputFormat.setOutputPath(conf,new Path(args[1]));
26.         JobClient.runJob(conf);
27.     }
28. }
```

- Create the jar file of this program and name it **countworddemo.jar.**
- Run the jar file hadoop jar /home/codegyani/wordcountdemo.jar com.javatpoint.WC_Runner /test/data.txt /r_output
- The output is stored in /r_output/part-00000




- Now execute the command to see the output. hdfs dfs -cat /r_output/part-00000



**Analyze different use cases where MapReduce is used**

**Use Cases -1**

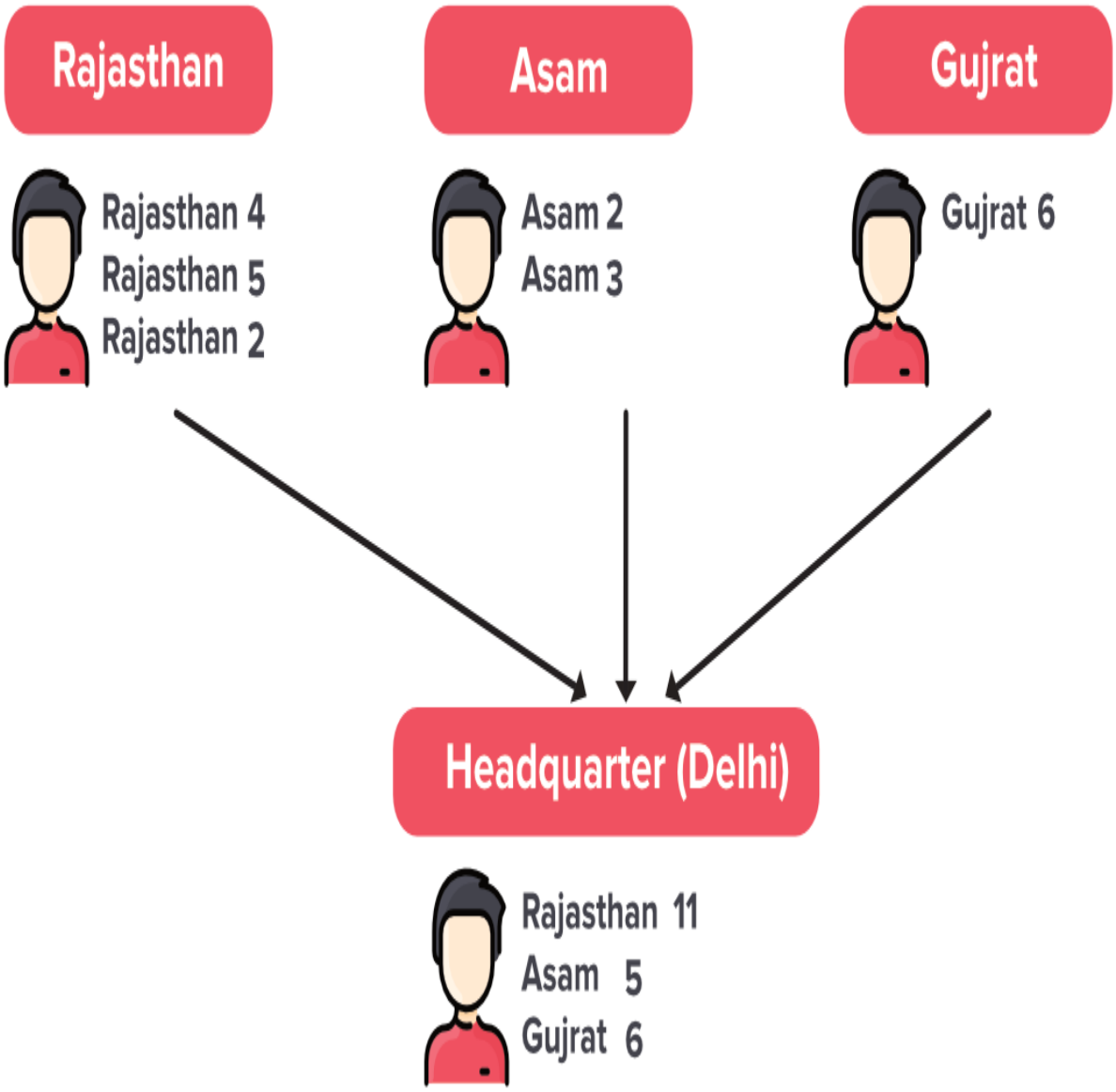
MapReduce is a programming model used to perform distributed processing in parallel in a Hadoop cluster, which Makes Hadoop working so fast. When you are dealing with Big Data, serial processing is no more of any use. MapReduce has mainly two tasks which are divided phase-wise:

- Map Task
- Reduce Task

Let us understand it with a real-time example, and the example helps you understand Mapreduce Programming Model in a story manner:

- Suppose the Indian government has assigned you the task to count the population of India. You can demand all the resources you want, but you have to do this task in 4 months. Calculating the population of such a large country is not an easy task for a single person(you). So what will be your approach?.
- One of the ways to solve this problem is to divide the country by states and assign individual in-charge to each state to count the population of that state.
- Task Of Each Individual: Each Individual has to visit every home present in the state and need to keep a record of each house members as:
- State_Name Member_House1
- State_Name Member_House2
- State_Name Member_House3
- 
- .
- .
- State_Name Member_House n
- .
- .

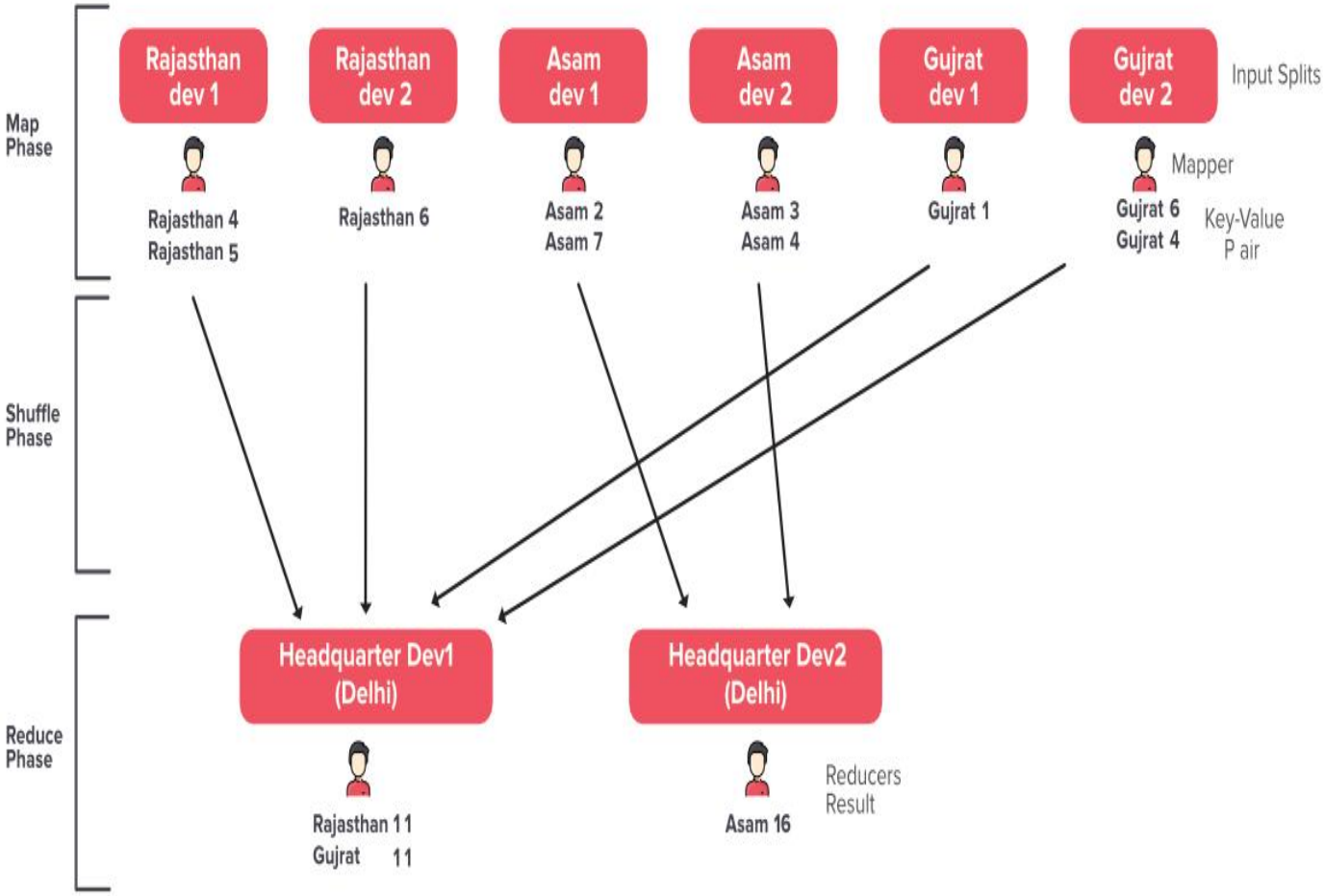
  For Simplicity, we have taken only three states.




This is a simple Divide and Conquer approach and will be followed by each individual to count people in his/her state.

- Once they have counted each house member in their respective state. Now they need to sum up their results and need to send it to the Head-quarter at New Delhi.
- We have a trained officer at the Head-quarter to receive all the results from each state and aggregate them by each state to get the population of that entire state. and Now, with this approach, you are easily able to count the population of India by summing up the results obtained at Head-quarter.
- The Indian Govt. is happy with your work and the next year they asked you to do the same job in 2 months instead of 4 months. Again you will be provided with all the resources you want.
- Since the Govt. has provided you with all the resources, you will simply double the number of assigned individual in-charge for each state from one to two. For that divide each state in 2 division and assigned different in-charge for these two divisions as:
- State_Name_Incharge_division1
- State_Name_Incharge_division2

- Similarly, each individual in charge of its division will gather the information about members from each house and keep its record.
- We can also do the same thing at the Head-quarters, so let's also divide the Head-quarter in two division as:
- Head-qurter_Division1
- Head-qurter_Division2
- Now with this approach, you can find the population of India in two months. But there is a small problem with this, we never want the divisions of the same state to send their result at different Head-quarters then, in that case, we have the partial population of that state in Head-quarter_Division1 and Head-quarter_Division2 which is inconsistent because we want consolidated population by the state, not the partial counting.
- One easy way to solve is that we can instruct all individuals of a state to either send there result to Head-quarter_Division1 or Head-quarter_Division2. Similarly, for all the states.
- Our problem has been solved, and you successfully did it in two months.
- Now, if they ask you to do this process in a month, you know how to approach the solution.
- Great, now we have a good scalable model that works so well. The model we have seen in this example is like the MapReduce Programming model. so now you must be aware that MapReduce is a programming model, not a programming language.




Now let's discuss the phases and important things involved in our model.

**1. Map Phase:** The Phase where the individual in-charges are collecting the population of each house in their division is Map Phase.

- Mapper: Involved individual in-charge for calculating population
- Input Splits: The state or the division of the state
- Key-Value Pair: Output from each individual Mapper like the key is Rajasthan and value is 2

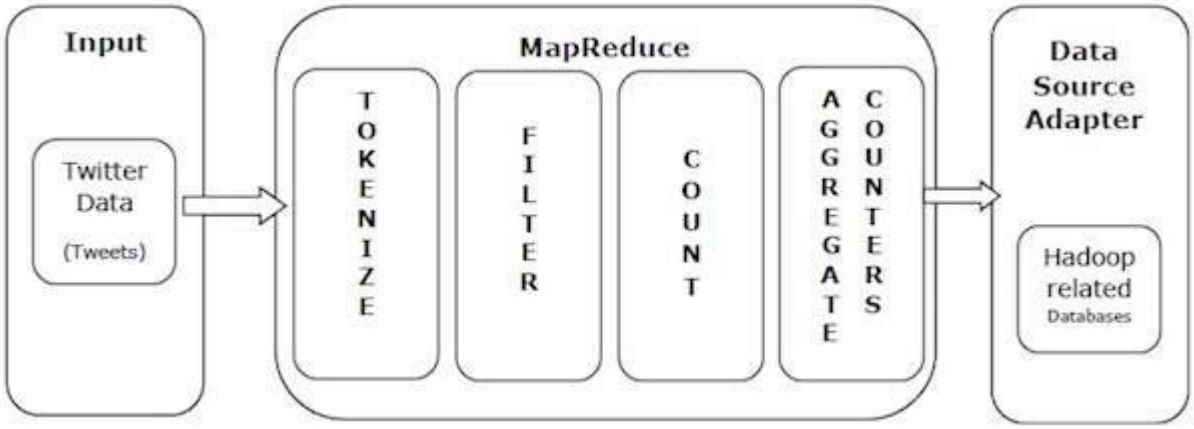
**2. Reduce Phase:** The Phase where you are aggregating your result

- Reducers: Individuals who are aggregating the actual result. Here in our example, the trained-officers. Each Reducer produce the output as a key-value pair

**3. Shuffle Phase:** The Phase where the data is copied from Mappers to Reducers is Shuffler's Phase. It comes in between Map and Reduces phase. Now the Map Phase, Reduce Phase, and Shuffler Phase our three main Phases of our Mapreduce.

Use Case 2

Let us take a real-world example to comprehend the power of MapReduce. Twitter receives around 500 million tweets per day, which is nearly 3000 tweets per second. The following illustration shows how Tweeter manages its tweets with the help of MapReduce.




As shown in the illustration, the MapReduce algorithm performs the following actions −

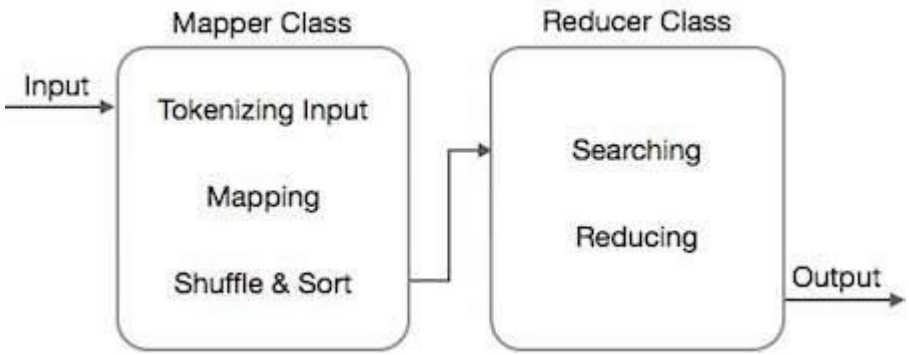
- **Tokenize** − Tokenizes the tweets into maps of tokens and writes them as key-value pairs.
- **Filter** − Filters unwanted words from the maps of tokens and writes the filtered maps as key-value pairs.
- **Count** − Generates a token counter per word.
- **Aggregate Counters** − Prepares an aggregate of similar counter values into small manageable units.

**MapReduce - Algorithm**

The MapReduce algorithm contains two important tasks, namely Map and Reduce.

- The map task is done by means of Mapper Class
- The reduce task is done by means of Reducer Class.

Mapper class takes the input, tokenizes it, maps and sorts it. The output of Mapper class is used as input by Reducer class, which in turn searches matching pairs and reduces them.



MapReduce implements various mathematical algorithms to divide a task into small parts and assign them to multiple systems. In technical terms, MapReduce algorithm helps in sending the Map & Reduce tasks to appropriate servers in a cluster.

These mathematical algorithms may include the following −

- Sorting
- Searching
- Indexing
- TF-IDF

Sorting

Sorting is one of the basic MapReduce algorithms to process and analyze data. MapReduce implements sorting algorithm to automatically sort the output key-value pairs from the mapper by their keys.

- Sorting methods are implemented in the mapper class itself.
- In the Shuffle and Sort phase, after tokenizing the values in the mapper class, the **Context** class (user-defined class) collects the matching valued keys as a collection.
- To collect similar key-value pairs (intermediate keys), the Mapper class takes the help of **RawComparator** class to sort the key-value pairs.
- The set of intermediate key-value pairs for a given Reducer is automatically sorted by Hadoop to form key-values (K2, {V2, V2, …}) before they are presented to the Reducer.

Searching

Searching plays an important role in MapReduce algorithm. It helps in the combiner phase (optional) and in the Reducer phase. Let us try to understand how Searching works with the help of an example.

**Example**

The following example shows how MapReduce employs Searching algorithm to find out the details of the employee who draws the highest salary in a given employee dataset.

- Let us assume we have employee data in four different files − A, B, C, and D. Let us also assume there are duplicate employee records in all four files because of importing the employee data from all database tables repeatedly. See the following illustration.


- **The Map phase** processes each input file and provides the employee data in key-value pairs (<k, v> : <emp name, salary>). See the following illustration.


- **The combiner phase** (searching technique) will accept the input from the Map phase as a key-value pair with employee name and salary. Using searching technique, the combiner will check all the employee salary to find the highest salaried employee in each file. See the following snippet.

```
<k: employee name, v: salary>
Max= the salary of an first employee. Treated as max salary

if(v(second employee).salary > Max){
   Max = v(salary);
}

else{
   Continue checking;
}
```

The expected result is as follows −

| | | | |
|---|---|---|---|
| <satish, 26000> | <gopal, 50000> | <kiran, 45000> | <manisha, 45000> |
| | | | |

- **Reducer phase** − Form each file, you will find the highest salaried employee. To avoid redundancy, check all the <k, v> pairs and eliminate duplicate entries, if any. The same algorithm is used in between the four <k, v> pairs, which are coming from four input files. The final output should be as follows −

<gopal, 50000>

**Indexing**

Normally indexing is used to point to a particular data and its address. It performs batch indexing on the input files for a particular Mapper.

The indexing technique that is normally used in MapReduce is known as **inverted index.** Search engines like Google and Bing use inverted indexing technique. Let us try to understand how Indexing works with the help of a simple example.

**Example**

The following text is the input for inverted indexing. Here T[0], T[1], and t[2] are the file names and their content are in double quotes.

T[0] = "it is what it is"
T[1] = "what is it"
T[2] = "it is a banana"

After applying the Indexing algorithm, we get the following output −

"a": {2}
"banana": {2}
"is": {0, 1, 2}
"it": {0, 1, 2}
"what": {0, 1}

Here "a": {2} implies the term "a" appears in the T[2] file. Similarly, "is": {0, 1, 2} implies the term "is" appears in the files T[0], T[1], and T[2].

**TF-IDF**

TF-IDF is a text processing algorithm which is short for Term Frequency − Inverse Document Frequency. It is one of the common web analysis algorithms. Here, the term 'frequency' refers to the number of times a term appears in a document.

**Term Frequency (TF)**

It measures how frequently a particular term occurs in a document. It is calculated by the number of times a word appears in a document divided by the total number of words in that document.

TF(the) = (Number of times term the ‘the’ appears in a document) / (Total number of terms in the document)

**Inverse Document Frequency (IDF)**

It measures the importance of a term. It is calculated by the number of documents in the text database divided by the number of documents where a specific term appears.

While computing TF, all the terms are considered equally important. That means, TF counts the term frequency for normal words like “is”, “a”, “what”, etc. Thus we need to know the frequent terms while scaling up the rare ones, by computing the following −

IDF(the) = log_e(Total number of documents / Number of documents with term ‘the’ in it).

The algorithm is explained below with the help of a small example.

**Example**

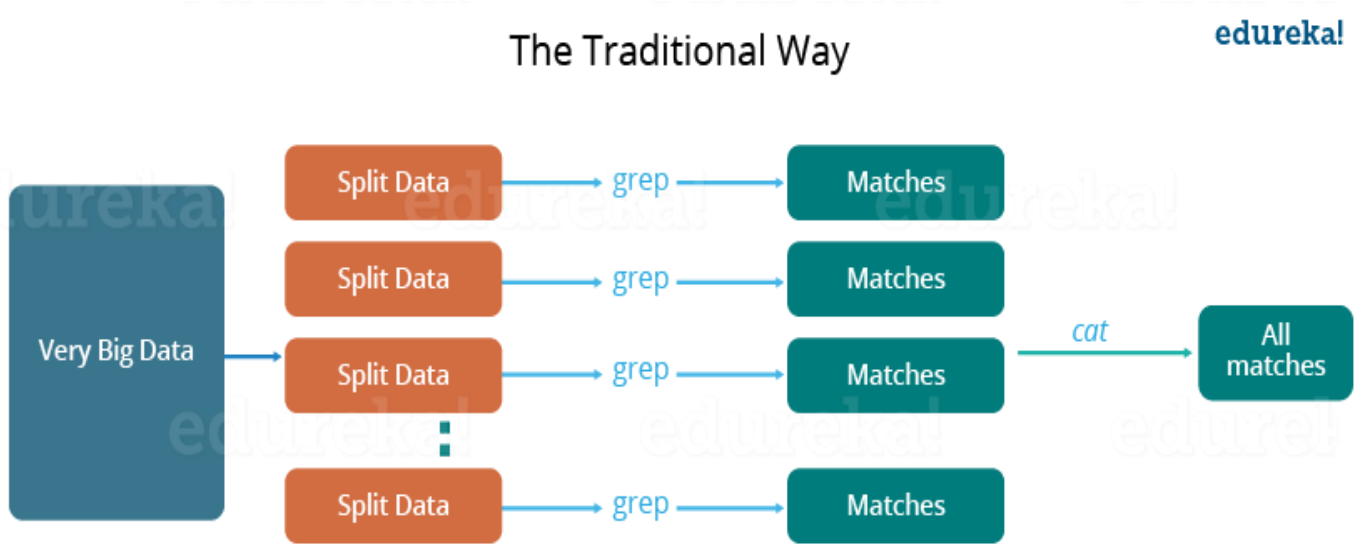
Consider a document containing 1000 words, wherein the word **hive** appears 50 times. The TF for **hive** is then (50 / 1000) = 0.05.

Now, assume we have 10 million documents and the word **hive** appears in 1000 of these. Then, the IDF is calculated as log(10,000,000 / 1,000) = 4.

The TF-IDF weight is the product of these quantities − 0.05 × 4 = 0.20.

**Differentiate between traditional way and MapReduce way**

**Traditional Way**




Let us understand, when the MapReduce framework was not there, how parallel and distributed processing used to happen in a traditional way. So, let us take an example where I have a weather log containing the daily average temperature of the years from 2000 to 2015. Here, I want to calculate the day having the highest temperature in each year.

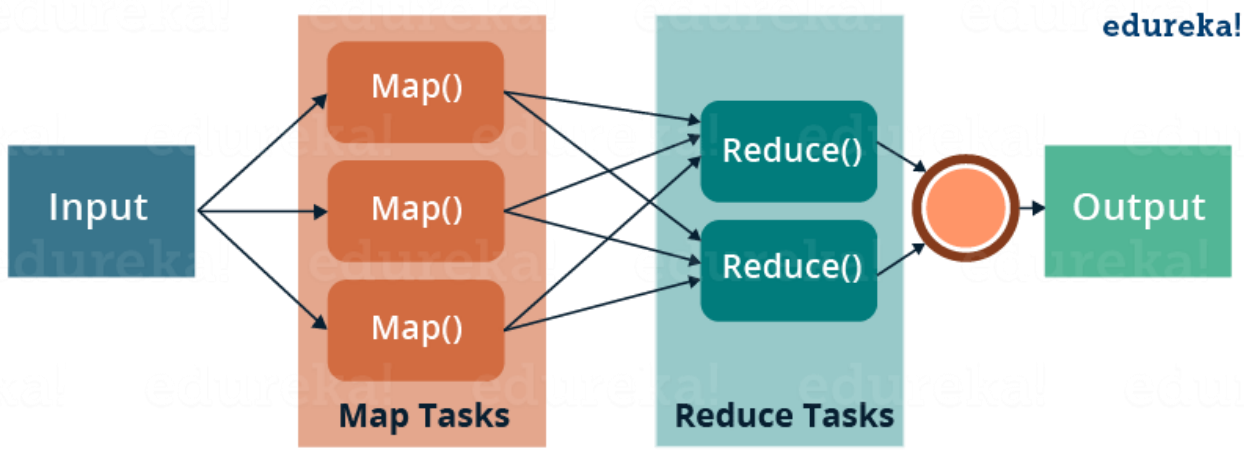
So, just like in the traditional way, I will split the data into smaller parts or blocks and store them in different machines. Then, I will find the highest temperature in each part stored in the corresponding machine. At last, I will combine the results received from each of the machines to have the final output. Let us look at the challenges associated with this traditional approach:

1. **Critical path problem:** It is the amount of time taken to finish the job without delaying the next milestone or actual completion date. So, if, any of the machines delay the job, the whole work gets delayed.
2. **Reliability problem:** What if, any of the machines which are working with a part of data fails? The management of this failover becomes a challenge.
3. **Equal split issue:** How will I divide the data into smaller chunks so that each machine gets even part of data to work with. In other words, how to equally divide the data such that no individual machine is overloaded or underutilized.
4. **The single split may fail:** If any of the machines fail to provide the output, I will not be able to calculate the result. So, there should be a mechanism to ensure this fault tolerance capability of the system.
5. **Aggregation of the result:** There should be a mechanism to aggregate the result generated by each of the machines to produce the final output.

These are the issues which I will have to take care individually while performing parallel processing of huge data sets when using traditional approaches.

To overcome these issues, we have the MapReduce framework which allows us to perform such parallel computations without bothering about the issues like reliability, fault tolerance etc. Therefore, MapReduce gives you the flexibility to write code logic without caring about the design issues of the system.

**What is MapReduce?**




MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment.

- MapReduce consists of two distinct tasks – Map and Reduce.
- As the name MapReduce suggests, the reducer phase takes place after the mapper phase has been completed.
- So, the first is the map job, where a block of data is read and processed to produce key-value pairs as intermediate outputs.
- The output of a Mapper or map job (key-value pairs) is input to the Reducer.
- The reducer receives the key-value pair from multiple map jobs.
- Then, the reducer aggregates those intermediate data tuples (intermediate key-value pair) into a smaller set of tuples or key-value pairs which is the final output.

Let us understand more about MapReduce and its components. MapReduce majorly has the following three Classes. They are,

**Mapper Class**

The first stage in Data Processing using MapReduce is the **Mapper Class.** Here, RecordReader processes each Input record and generates the respective key-value pair. Hadoop's Mapper store saves this intermediate data into the local disk.

- **Input Split**

It is the logical representation of data. It represents a block of work that contains a single map task in the MapReduce Program.

- **RecordReader**

It interacts with the Input split and converts the obtained data in the form of **Key-Value** Pairs.

**Reducer Class**

The Intermediate output generated from the mapper is fed to the reducer which processes it and generates the final output which is then saved in the **HDFS.**

**Driver Class**

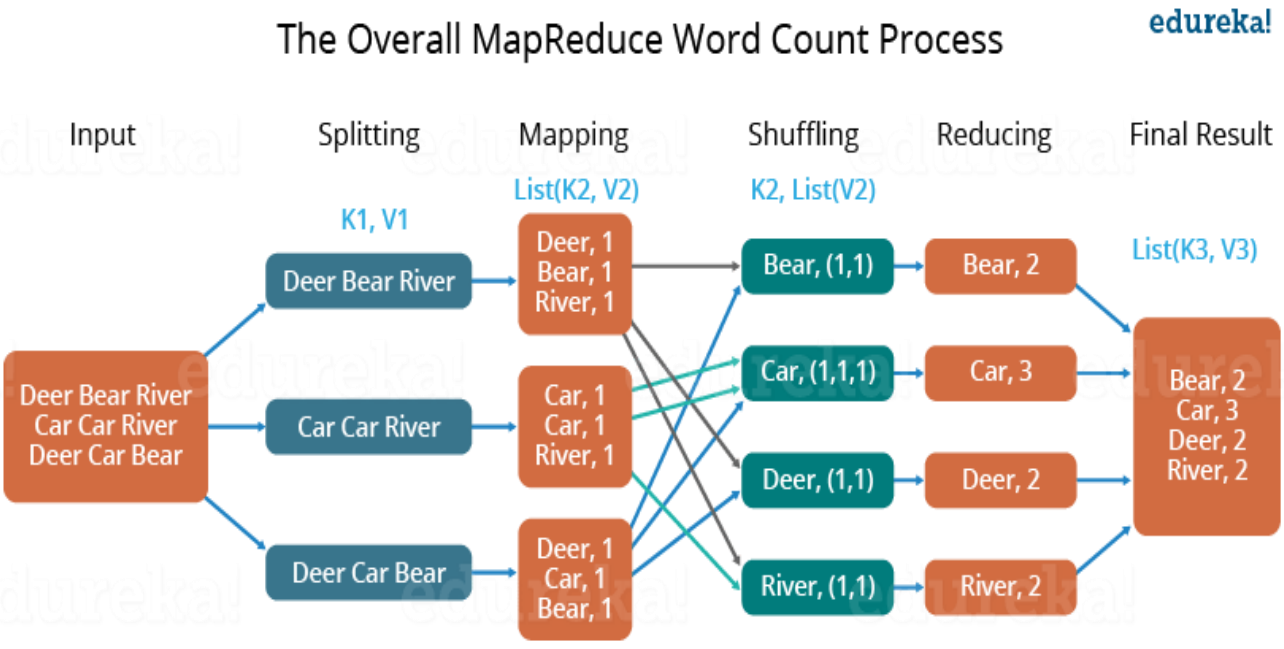
The major component in a MapReduce job is a **Driver Class.** It is responsible for setting up a MapReduce Job to run-in Hadoop. We specify the names of **Mapper** and **Reducer** Classes long with data types and their respective job names.

**A Word Count Example of MapReduce**

Let us understand, how a MapReduce works by taking an example where I have a text file called example.txt whose contents are as follows:

**Dear, Bear, River, Car, Car, River, Deer, Car and Bear**

Now, suppose, we have to perform a word count on the sample.txt using MapReduce. So, we will be finding the unique words and the number of occurrences of those unique words.




- First, we divide the input into three splits as shown in the figure. This will distribute the work among all the map nodes.
- Then, we tokenize the words in each of the mappers and give a hardcoded value (1) to each of the tokens or words. The rationale behind giving a hardcoded value equal to 1 is that every word, in itself, will occur once.
- Now, a list of key-value pair will be created where the key is nothing but the individual words and value is one. So, for the first line (Dear Bear River) we have 3 key-value pairs – Dear, 1; Bear, 1; River, 1. The mapping process remains the same on all the nodes.
- After the mapper phase, a partition process takes place where sorting and shuffling happen so that all the tuples with the same key are sent to the corresponding reducer.

- So, after the sorting and shuffling phase, each reducer will have a unique key and a list of values corresponding to that very key. For example, Bear, [1,1]; Car, [1,1,1].., etc.
- Now, each Reducer counts the values which are present in that list of values. As shown in the figure, reducer gets a list of values which is [1,1] for the key Bear. Then, it counts the number of ones in the very list and gives the final output as – Bear, 2.
- Finally, all the output key/value pairs are then collected and written in the output file.

**Advantages of MapReduce**

The two biggest advantages of MapReduce are:

1. Parallel Processing:

In MapReduce, we are dividing the job among multiple nodes and each node works with a part of the job simultaneously. So, MapReduce is based on Divide and Conquer paradigm which helps us to process the data using different machines. As the data is processed by multiple machines instead of a single machine in parallel, the time taken to process the data gets reduced by a tremendous amount as shown in the figure below

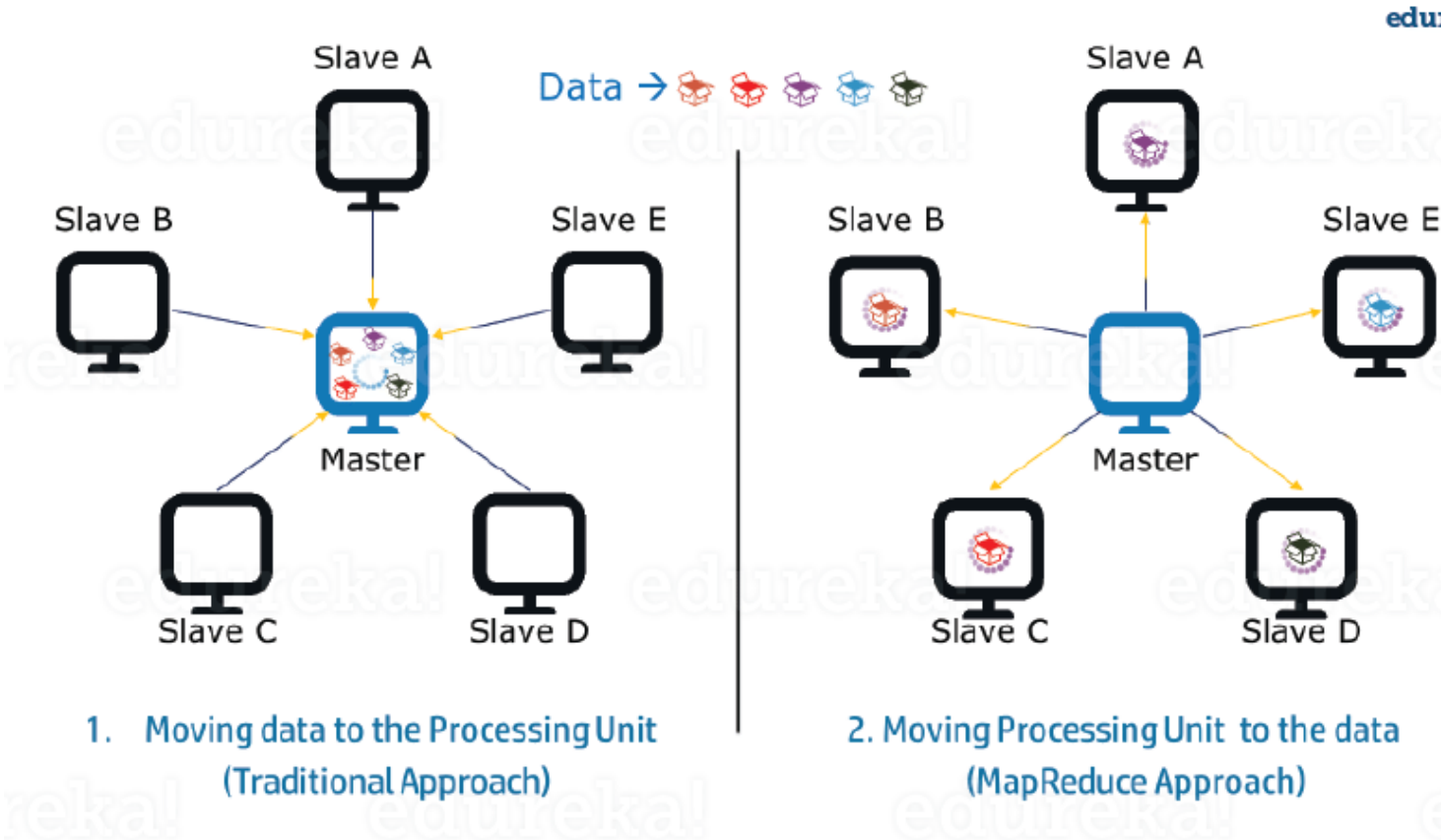



***Fig.:*** Traditional Way Vs. MapReduce Way – MapReduce Tutorial

2. **Data Locality:**

Instead of moving data to the processing unit, we are moving the processing unit to the data in the MapReduce Framework. In the traditional system, we used to bring data to the processing unit and process it. But, as the data grew and became very huge, bringing this huge amount of data to the processing unit posed the following issues:

- Moving huge data to processing is costly and deteriorates the network performance.
- Processing takes time as the data is processed by a single unit which becomes the bottleneck.
- The master node can get over-burdened and may fail.

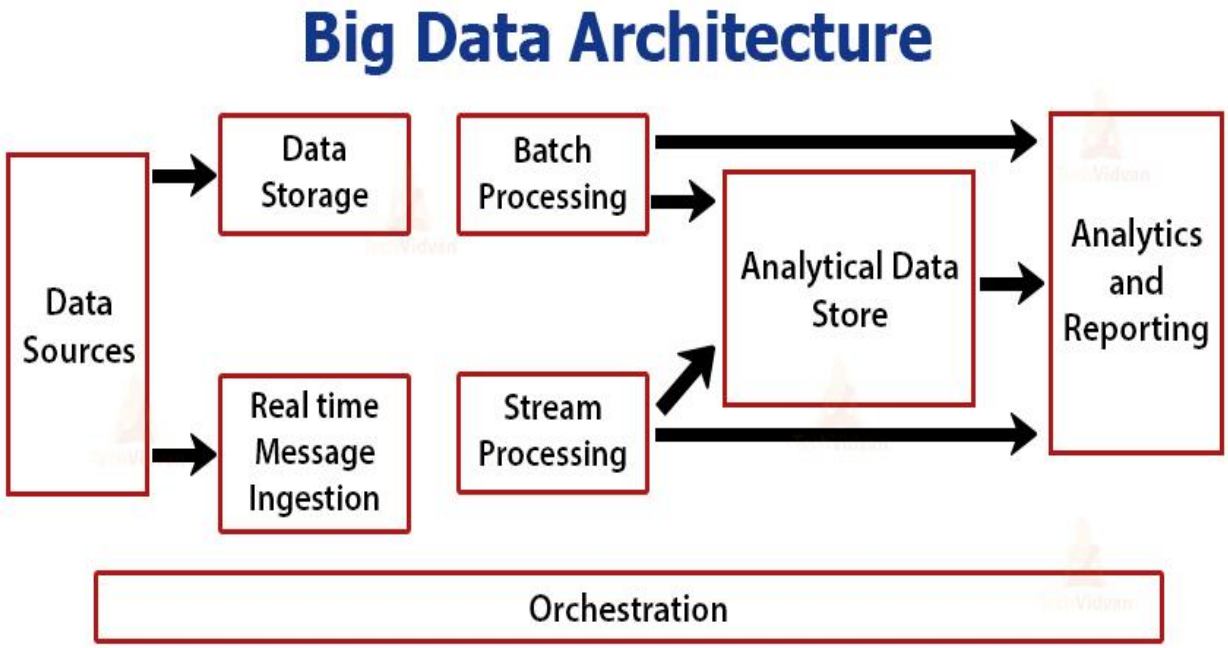
Now, MapReduce allows us to overcome the above issues by bringing the processing unit to the data. So, as you can see in the above image that the data is distributed among multiple nodes where each node processes the part of the data residing on it. This allows us to have the following advantages:

- It is very cost-effective to move processing unit to the data.
- The processing time is reduced as all the nodes are working with their part of the data in parallel.
- Every node gets a part of the data to process and therefore, there is no chance of a node getting overburdened.

**Unit – V**

**Introduction to Hadoop Features: New Big Data Architecture, Introducing HADOOP Features – Apache Hive, Apache HBase, Pig.**

A big data architecture is designed to handle the ingestion, processing, and analysis of data that is too large or complex for traditional database systems.




Big data solutions typically involve one or more of the following types of workload:

- Batch processing of big data sources at rest.
- Real-time processing of big data in motion.
- Interactive exploration of big data.
- Predictive analytics and machine learning.

Most big data architectures include some or all of the following components:

**Data sources**: All big data solutions start with one or more data sources. Examples include:

Application data stores, such as relational databases.

Static files produced by applications, such as web server log files.

Real-time data sources, such as IoT devices.

**Data storage:** Data for batch processing operations is typically stored in a distributed file store that can hold high volumes of large files in various formats. This kind of store is often called a data lake. Options for implementing this storage include Azure Data Lake Store or blob containers in Azure Storage.

**Batch processing:** Because the data sets are so large, often a big data solution must process data files using long-running batch jobs to filter, aggregate, and otherwise prepare the data for analysis. Usually these jobs involve reading source files, processing them, and writing the output to new files. Options include running U-SQL jobs in Azure Data Lake Analytics, using Hive, Pig, or custom Map/Reduce jobs in an HDInsight Hadoop cluster, or using Java, Scala, or Python programs in an HDInsight Spark cluster.

Real-time message ingestion: If the solution includes real-time sources, the architecture must include a way to capture and store real-time messages for stream processing. This might be a simple data store, where incoming messages are dropped into a folder for processing. However, many solutions need a message ingestion store to act as a buffer for messages, and to support scale-out processing, reliable delivery, and other message queuing semantics.

**Stream processing**: After capturing real-time messages, the solution must process them by filtering, aggregating, and otherwise preparing the data for analysis. The processed stream data is then written to an output sink. Azure Stream Analytics provides a managed stream processing service based on perpetually running SQL queries that operate on unbounded streams. You can also use open source Apache streaming technologies like Storm and Spark Streaming in an HDInsight cluster.

**Analytical data store:** Many big data solutions prepare data for analysis and then serve the processed data in a structured format that can be queried using analytical tools. The analytical data store used to serve these queries can be a Kimball-style relational data warehouse, as seen in most traditional business intelligence (BI) solutions. Alternatively, the data could be presented through a low-latency NoSQL technology such as HBase, or an interactive Hive database that provides a metadata abstraction over data files in the distributed data store. Azure Synapse Analytics provides a managed service for large-scale, cloud-based data warehousing. HDInsight supports Interactive Hive, HBase, and Spark SQL, which can also be used to serve data for analysis.

**Analysis and reporting:** The goal of most big data solutions is to provide insights into the data through analysis and reporting. To empower users to analyze the data, the architecture may include a data modeling layer, such as a multidimensional OLAP cube or tabular data model in Azure Analysis Services. It might also support self-service BI, using the modeling and visualization technologies in Microsoft Power BI or Microsoft Excel. Analysis and reporting can also take the form of interactive data exploration by data scientists or data analysts. For these scenarios, many Azure services support analytical notebooks, such as Jupyter, enabling these users to leverage their existing skills with Python or R. For large-scale data exploration, you can use Microsoft R Server, either standalone or with Spark.

**Orchestration:** Most big data solutions consist of repeated data processing operations, encapsulated in work flows, that transform source data, move data between multiple sources and sinks, load the processed data into an analytical data store, or push the results straight to a report or dashboard. To automate these workflows, you can use an orchestration technology such Azure Data Factory or Apache Oozie and Sqoop.

Azure includes many services that can be used in a big data architecture. They fall roughly into two categories:

- Managed services, including Azure Data Lake Store, Azure Data Lake Analytics, Azure Synapse Analytics, Azure Stream Analytics, Azure Event Hub, Azure IoT Hub, and Azure Data Factory.
- Open source technologies based on the Apache Hadoop platform, including HDFS, HBase, Hive, Pig, Spark, Storm, Oozie, Sqoop, and Kafka. These technologies are available on Azure in the Azure HDInsight service.

These options are not mutually exclusive, and many solutions combine open source technologies with Azure services.

When to use this architecture

Consider this architecture style when you need to:

- Store and process data in volumes too large for a traditional database.
- Transform unstructured data for analysis and reporting.
- Capture, process, and analyze unbounded streams of data in real time, or with low latency.
- Use Azure Machine Learning or Microsoft Cognitive Services.

**Benefits**

- Technology choices. You can mix and match Azure managed services and Apache technologies in HDInsight clusters, to capitalize on existing skills or technology investments.
- Performance through parallelism. Big data solutions take advantage of parallelism, enabling high-performance solutions that scale to large volumes of data.
- Elastic scale. All of the components in the big data architecture support scale-out provisioning, so that you can adjust your solution to small or large workloads, and pay only for the resources that you use.
- Interoperability with existing solutions. The components of the big data architecture are also used for IoT processing and enterprise BI solutions, enabling you to create an integrated solution across data workloads.

**Challenges**

**Complexity.** Big data solutions can be extremely complex, with numerous components to handle data ingestion from multiple data sources. It can be challenging to build, test, and troubleshoot big data processes. Moreover, there may be a large number of configuration settings across multiple systems that must be used in order to optimize performance.

**Skillset.** Many big data technologies are highly specialized, and use frameworks and languages that are not typical of more general application architectures. On the other hand, big data technologies are evolving new APIs that build on more established languages. For example, the U-SQL language in Azure Data Lake Analytics is based on a combination of Transact-SQL and C#. Similarly, SQL-based APIs are available for Hive, HBase, and Spark.

**Technology maturity.** Many of the technologies used in big data are evolving. While core Hadoop technologies such as Hive and Pig have stabilized, emerging technologies such as Spark introduce extensive changes and enhancements with each new release. Managed services such as Azure Data Lake Analytics and Azure Data Factory are relatively young, compared with other Azure services, and will likely evolve over time.

**Security.** Big data solutions usually rely on storing all static data in a centralized data lake. Securing access to this data can be challenging, especially when the data must be ingested and consumed by multiple applications and platforms.

**Introducing Hadoop Features –**

**Apache Hive**

"The Apache Hive™ data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. The structure can be projected onto data already in storage."

In other words, Hive is an open-source system that processes structured data in Hadoop, residing on top of the latter for summarizing Big Data, as well as facilitating analysis and queries.

**Hive's Features**

The following are Hive's chief characteristics to keep in mind when using it for data processing:

- Hive is designed for querying and managing only structured data stored in tables
- Hive is scalable, fast, and uses familiar concepts
- Schema gets stored in a database, while processed data goes into a Hadoop Distributed File System (HDFS)
- Tables and databases get created first; then data gets loaded into the proper tables
- Hive supports four file formats: ORC, SEQUENCEFILE, RCFILE (Record Columnar File), and TEXTFILE
- Hive uses an SQL-inspired language, sparing the user from dealing with the complexity of MapReduce programming. It makes learning more accessible by utilizing familiar concepts found in relational databases, such as columns, tables, rows, and schema, etc.
- The most significant difference between the Hive Query Language (HQL) and SQL is that Hive executes queries on Hadoop's infrastructure instead of on a traditional database
- Since Hadoop's programming works on flat files, Hive uses directory structures to "partition" data, improving performance on specific queries
- Hive supports partition and buckets for fast and simple data retrieval
- Hive supports custom user-defined functions (UDF) for tasks like data cleansing and filtering. Hive UDFs can be defined according to programmers' requirements

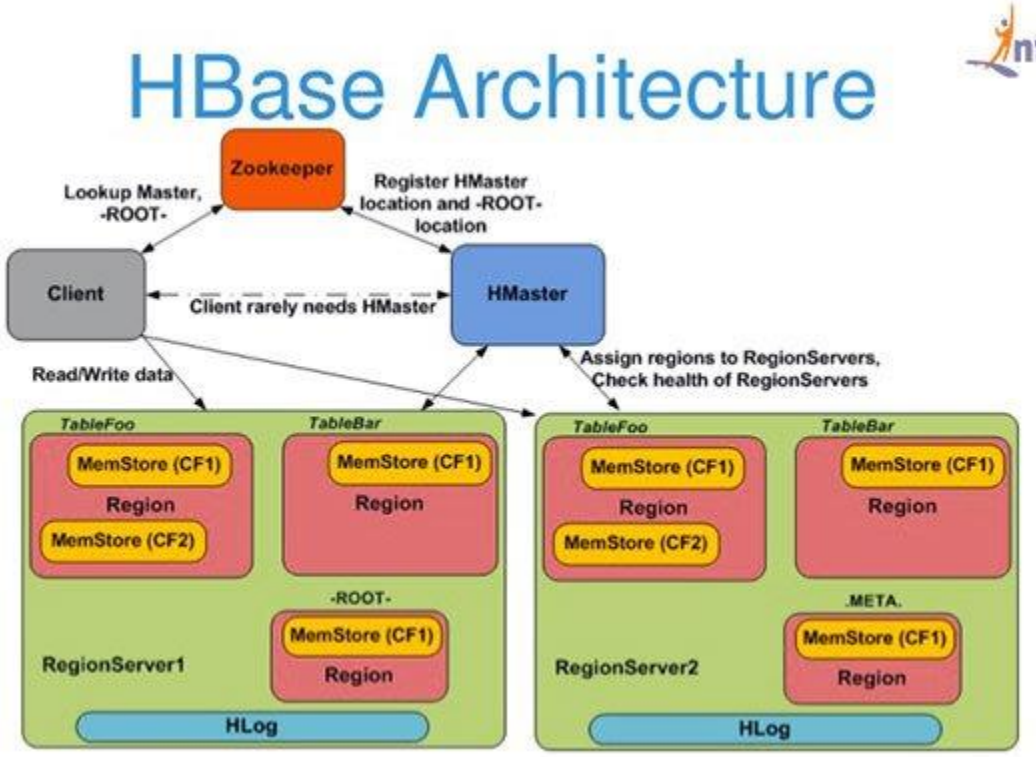
**Apache HBase**

Apache HBase is an Apache Hadoop project and Open Source, non-relational distributed Hadoop database that had its genesis in the Google’s Bigtable. The programming language of HBase is Java. Today it is an integral part of the Apache Software Foundation and the Hadoop ecosystem. It is a high availability database that exclusively runs on top of the HDFS and provides the Capabilities of Google Bigtable for the Hadoop framework for storing huge volumes of unstructured data at breakneck speeds in order to derive valuable insights from it.

It has an extremely fault-tolerant way of storing data and is extremely good for storing sparse data. Sparse data is something like looking for a needle in a haystack. A real-life example of sparse data would be like looking for someone who has spent over $100,000 dollars in a single transaction on Amazon among the tens of millions of transactions that happen on any given week.

**The Architecture of Apache HBase**

The Apache HBase carries all the features of the original Google Bigtable paper like the Bloom filters, in-memory operations and compression. The tables of this database can serve as the input for MapReduce jobs on the Hadoop ecosystem and it can also serve as output after the data is processed by MapReduce. The data can be accessed via the Java API or through the REST API or even the Thrift and AVRO gateways.




What HBase is that it is basically a column-oriented key-value data store, and the since it works extremely fine with the kind of data that Hadoop process it is natural fit for deploying as a top layer on HDFS. It is extremely fast when it comes to both read and write operations and does not lose this extremely important quality even when the datasets are humongous. Therefore it is being widely used by corporations for its high throughput and low input/output latency. It cannot work as a replacement for the SQL database but it is perfectly possible to have an SQL layer on top of HBase to integrate it with the various business intelligence and analytics tools.

As it is obvious that HBase does not support SQL scripting but the same is written in Java like what we do for a MapReduce application.

Some of the salient features of HBase that makes it one of the most sought after message storing system is as follows:

- It has a completely distributed architecture and can work on extremely large scale data
- It works for extremely random read and write operations
- It has high security and easy management of data

- It provides an unprecedented high write throughput
- Scaling to meet additional requirements is seamless and quick
- Can be used for both structured and semi-structured data types
- It is good when you don't need full RDBMS capabilities
- It has a perfectly modular and linear scalability feature
- The data reads and writes are strictly consistent
- The table sharding can be easily configured and automatized
- The various servers are provided automatic failover support
- The MapReduce jobs can be backed with HBase Tables
- Client access is seamless with Java APIs.

**Pig**

Pig is a high-level platform or tool which is used to process the large datasets. It provides a high-level of abstraction for processing over the MapReduce. It provides a high-level scripting language, known as *Pig Latin* which is used to develop the data analysis codes. First, to process the data which is stored in the HDFS, the programmers will write the scripts using the Pig Latin Language. Internally *Pig Engine*(a component of Apache Pig) converted all these scripts into a specific map and reduce task. But these are not visible to the programmers in order to provide a high-level of abstraction. Pig Latin and Pig Engine are the two main components of the Apache Pig tool. The result of Pig always stored in the HDFS.

**Note:** Pig Engine has two type of execution environment i.e. a *local execution environment* in a single JVM (used when dataset is small in size)and *distributed execution environment* in a Hadoop Cluster.

**Need of Pig:** One limitation of MapReduce is that the development cycle is very long. Writing the reducer and mapper, compiling packaging the code, submitting the job and retrieving the output is a time-consuming task. Apache Pig reduces the time of development using the multi-query approach. Also, Pig is beneficial for the programmers who are not from Java background. 200 lines of Java code can be written in only 10 lines using the Pig Latin language. Programmers who have SQL knowledge needed less effort to learn Pig Latin.

**Evolution of Pig:** Earlier in 2006, Apache Pig was developed by Yahoo's researchers. At that time, the main idea to develop Pig was to execute the MapReduce jobs on extremely large datasets. In the year 2007, it moved to Apache Software Foundation(ASF) which makes it an open source project. The first version(*0.1*) of Pig came in the year 2008. The latest version of Apache Pig is *0.18* which came in the year 2017.

**Features of Apache Pig:**

- For performing several operations Apache Pig provides rich sets of operators like the filters, join, sort, etc.
- Easy to learn, read and write. Especially for SQL-programmer, Apache Pig is a boon.
- Apache Pig is extensible so that you can make your own user-defined functions and process.
- Join operation is easy in Apache Pig.
- Fewer lines of code.
- Apache Pig allows splits in the pipeline.
- The data structure is multivalued, nested, and richer.
- Pig can handle the analysis of both structured and unstructured data.

**Difference between Pig and MapReduce**

| Apache Pig | MapReduce |
|---|---|
| It is a scripting language. | It is a compiled programming language. |
| Abstraction is at higher level. | Abstraction is at lower level. |
| It have less line of code as compared to MapReduce. | Lines of code is more. |
| Less effort is needed for Apache Pig. | More development efforts are required for MapReduce. |
| Code efficiency is less as compared to MapReduce. | As compared to Pig efficiency of code is higher. |

**Applications of Apache Pig:**

- For exploring large datasets Pig Scripting is used.
- Provides the supports across large data-sets for Ad-hoc queries.
- In the prototyping of large data-sets processing algorithms.
- Required to process the time sensitive data loads.
- For collecting large amounts of datasets in form of search logs and web crawls.
- Used where the analytical insights are needed using the sampling.

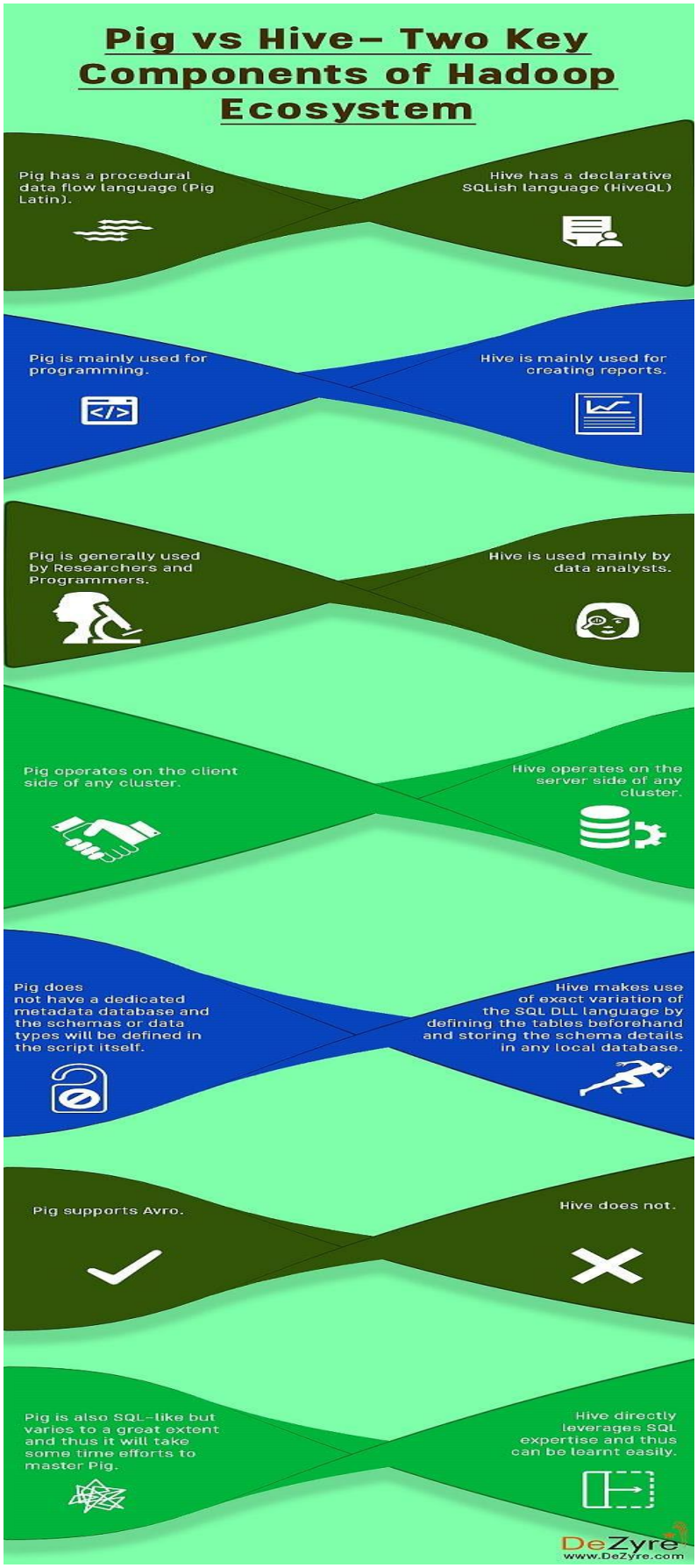
**Types of Data Models in Apache Pig:** It consist of the 4 types of data models as follows:

- **Atom**: It is a atomic data value which is used to store as a string. The main use of this model is that it can be used as a number and as well as a string.
- **Tuple**: It is an ordered set of the fields.
- **Bag**: It is a collection of the tuples.
- **Map**: It is a set of key/value pairs.

t was also quite evident that the employees were not producing up to their fullest capability. This happened in spite of the fact that it was one of the most progressive companies with pension schemes, sickness benefit schemes, and numerous other facilities offered to its employees. The earlier attempts of the efficiency experts produced inconclusive findings. So the company sought help from the group of university professors to find a solution to the problem. The study continued for an extended period of time and had gone' through various phases, which is briefly described here.

**Hive vs Pig**

| **Pig** | **Hive** |
|---|---|
| Procedural Data Flow Language | Declarative SQLish Language |
| For Programming | For creating reports |
| Mainly used by Researchers and Programmers | Mainly used by Data Analysts |
| Operates on the client side of a cluster. | Operates on the server side of a cluster. |
| Does not have a dedicated metadata database. | Makes use of exact variation of dedicated SQL DDL language by defining tables beforehand. |
| Pig is SQL like but varies to a great extent. | Directly leverages SQL and is easy to learn for database experts. |
| Pig supports Avro file format. | Hive does not support it. |


Pig vs Hive– Two Key Components of Hadoop Ecosystem
Pig has a procedural data flow language (Pig Latin).
Hive has a declarative SQLish language (HiveQL)
Pig is mainly used for programming.
Hive is mainly used for creating reports.
Pig is generally used by Researchers and Programmers.
Hive is used mainly by data analysts.
Pig operates on the client side of any cluster.
Hive operates on the server side of any cluster.
Pig does not have a dedicated metadata database and the schemas or data types will be defined in the script itself.
Hive makes use of exact variation of the SQL DLL language by defining the tables beforehand and storing the schema details in any local database.
Pig supports Avro.
Hive does not.
Pig is also SQL-like but varies to a great extent and thus it will take some time efforts to master Pig.
Hive directly leverages SQL expertise and thus can be learnt easily.
DeZyre
www.DeZyre.com

**Difference between Pig and Hive**

| **Characteristic** | **Pig** | **Hive** |
|---|---|---|
| Language Name | Pig Latin | HiveQL |
| Type of Language | Dataflow | Declarative (SQL Dialect) |
| Developed By | Yahoo | Facebook |
| Data Structures Supported | Nested and Complex | |
| Relational Complete | YES | YES |
| Schema Optional | YES | NO |
| Turing Complete | YES,when you extend it with Java User Defined Functions. | YES, when you extend it with Java User Defined Functions. |

## UNIT – VI

**Multi Node Cluster: Multi Node Cluster, Install Java, Creating User Account, Mapping the Nodes, Installing Hadoop, Configuring Hadoop, Start Hadoop Services, Adding New Data Node in the Hadoop Cluster, Removing New Data Node from the Hadoop Cluster.**

A Multi Node Cluster in Hadoop contains two or more DataNodes in a distributed Hadoop environment. This is practically used in organizations to store and analyze their Petabytes and Exabytes of data.

we are explaining the Hadoop cluster environment using three systems (one master and two slaves); given below are their IP addresses.

- Hadoop Master: 192.168.1.15 (hadoop-master)
- Hadoop Slave: 192.168.1.16 (hadoop-slave-1)
- Hadoop Slave: 192.168.1.17 (hadoop-slave-2)

Follow the steps given below to have Hadoop Multi-Node cluster setup.

Installing Java

Java is the main prerequisite for Hadoop. First of all, you should verify the existence of java in your system using "java -version". The syntax of java version command is given below.

```
$ java -version
```

If everything works fine it will give you the following output.

```
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)
```

If java is not installed in your system, then follow the given steps for installing java.

**Step 1**

Download java (JDK <latest version> - X64.tar.gz) by visiting the following link www.oracle.com

Then **jdk-7u71-linux-x64.tar.gz** will be downloaded into your system.

**Step 2**

Generally you will find the downloaded java file in Downloads folder. Verify it and extract the **jdk-7u71-linux-x64.gz** file using the following commands.

```
$ cd Downloads/
$ ls
jdk-7u71-Linux-x64.gz

$ tar zxf jdk-7u71-Linux-x64.gz
$ ls
jdk1.7.0_71 jdk-7u71-Linux-x64.gz
```

**Step 3**

To make java available to all the users, you have to move it to the location "/usr/local/". Open the root, and type the following commands.

```
$ su
password:
# mv jdk1.7.0_71 /usr/local/
# exit
```

**Step 4**

For setting up **PATH** and **JAVA_HOME** variables, add the following commands to **~/.bashrc** file.

```
export JAVA_HOME=/usr/local/jdk1.7.0_71
export PATH=PATH:$JAVA_HOME/bin
```

Now verify the **java -version** command from the terminal as explained above. Follow the above process and install java in all your cluster nodes.

**Creating User Account**

Create a system user account on both master and slave systems to use the Hadoop installation.

```
# useradd hadoop
# passwd hadoop
```

Mapping the nodes

You have to edit **hosts** file in **/etc/** folder on all nodes, specify the IP address of each system followed by their host names.

```
# vi /etc/hosts
enter the following lines in the /etc/hosts file.

192.168.1.109 hadoop-master
192.168.1.145 hadoop-slave-1
192.168.56.1 hadoop-slave-2
```

**Configuring Key Based Login**

Setup ssh in every node such that they can communicate with one another without any prompt for password.

```
# su hadoop
$ ssh-keygen -t rsa
$ ssh-copy-id -i ~/.ssh/id_rsa.pub tutorialspoint@hadoop-master
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop_tp1@hadoop-slave-1
$ ssh-copy-id -i ~/.ssh/id_rsa.pub hadoop_tp2@hadoop-slave-2
$ chmod 0600 ~/.ssh/authorized_keys
$ exit
```

**Installing Hadoop**

In the Master server, download and install Hadoop using the following commands.

```
# mkdir /opt/hadoop
# cd /opt/hadoop/
# wget http://apache.mesi.com.ar/hadoop/common/hadoop-1.2.1/hadoop-1.2.0.tar.gz
# tar -xzf hadoop-1.2.0.tar.gz
# mv hadoop-1.2.0 hadoop
# chown -R hadoop /opt/hadoop
# cd /opt/hadoop/hadoop/
```

**Configuring Hadoop**

You have to configure Hadoop server by making the following changes as given below.

**core-site.xml**

Open the **core-site.xml** file and edit it as shown below.

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://hadoop-master:9000/</value>
  </property>
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
</configuration>
```

**hdfs-site.xml**

Open the **hdfs-site.xml** file and edit it as shown below.

```
<configuration>
  <property>
    <name>dfs.data.dir</name>
    <value>/opt/hadoop/hadoop/dfs/name/data</value>
    <final>true</final>
  </property>

  <property>
    <name>dfs.name.dir</name>
    <value>/opt/hadoop/hadoop/dfs/name</value>
    <final>true</final>
  </property>

  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

**mapred-site.xml**

Open the **mapred-site.xml** file and edit it as shown below.

```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>hadoop-master:9001</value>
  </property>
</configuration>
```

**hadoop-env.sh**

Open the **hadoop-env.sh** file and edit JAVA_HOME, HADOOP_CONF_DIR, and HADOOP_OPTS as shown below.

**Note** − Set the JAVA_HOME as per your system configuration.

```
export JAVA_HOME=/opt/jdk1.7.0_17
export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
export HADOOP_CONF_DIR=/opt/hadoop/hadoop/conf
```

**Installing Hadoop on Slave Servers**

Install Hadoop on all the slave servers by following the given commands.

```
# su hadoop
$ cd /opt/hadoop
$ scp -r hadoop hadoop-slave-1:/opt/hadoop
$ scp -r hadoop hadoop-slave-2:/opt/hadoop
```

**Configuring Hadoop on Master Server**

Open the master server and configure it by following the given commands.

```
# su hadoop
$ cd /opt/hadoop/hadoop
```

**Configuring Master Node**

```
$ vi etc/hadoop/masters

hadoop-master
```

**Configuring Slave Node**

```
$ vi etc/hadoop/slaves

hadoop-slave-1
hadoop-slave-2
```

**Format Name Node on Hadoop Master**

```
# su hadoop
$ cd /opt/hadoop/hadoop
$ bin/hadoop namenode –format
11/10/14 10:58:07 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = hadoop-master/192.168.1.109
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 1.2.0
```

STARTUP_MSG: build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -
r 1479473;
compiled by 'hortonfo' on Mon May 6 06:59:37 UTC 2013
STARTUP_MSG: java = 1.7.0_71

************************************************************/
11/10/14 10:58:08 INFO util.GSet: Computing capacity for map BlocksMap
editlog=/opt/hadoop/hadoop/dfs/name/current/edits
……
……
……
11/10/14 10:58:08 INFO common.Storage: Storage directory
/opt/hadoop/hadoop/dfs/name has been successfully formatted.
11/10/14 10:58:08 INFO namenode.NameNode:
SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at hadoop-master/192.168.1.15
************************************************************/

**Starting Hadoop Services**

The following command is to start all the Hadoop services on the Hadoop-Master.

$ cd $HADOOP_HOME/sbin
$ start-all.sh

Adding a New DataNode in the Hadoop Cluster

Given below are the steps to be followed for adding new nodes to a Hadoop cluster.

**Networking**

Add new nodes to an existing Hadoop cluster with some appropriate network configuration. Assume the following network configuration.

For New node Configuration −

IP address : 192.168.1.103
netmask : 255.255.255.0
hostname : slave3.in

Adding User and SSH Access

**Add a User**

On a new node, add "hadoop" user and set password of Hadoop user to "hadoop123" or anything you want by using the following commands.

useradd hadoop
passwd hadoop

Setup Password less connectivity from master to new slave.

**Execute the following on the master**

```
mkdir -p $HOME/.ssh
chmod 700 $HOME/.ssh
ssh-keygen -t rsa -P " -f $HOME/.ssh/id_rsa
cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
chmod 644 $HOME/.ssh/authorized_keys
```

Copy the public key to new slave node in hadoop user $HOME directory

```
scp $HOME/.ssh/id_rsa.pub hadoop@192.168.1.103:/home/hadoop/
```

**Execute the following on the slaves**

Login to hadoop. If not, login to hadoop user.

```
su hadoop ssh -X hadoop@192.168.1.103
```

Copy the content of public key into file **"$HOME/.ssh/authorized_keys"** and then change the permission for the same by executing the following commands.

```
cd $HOME
mkdir -p $HOME/.ssh
chmod 700 $HOME/.ssh
cat id_rsa.pub >>$HOME/.ssh/authorized_keys
chmod 644 $HOME/.ssh/authorized_keys
```

Check ssh login from the master machine. Now check if you can ssh to the new node without a password from the master.

```
ssh hadoop@192.168.1.103 or hadoop@slave3
```

Set Hostname of New Node

You can set hostname in file **/etc/sysconfig/network**

On new slave3 machine

```
NETWORKING = yes
HOSTNAME = slave3.in
```

To make the changes effective, either restart the machine or run hostname command to a new machine with the respective hostname (restart is a good option).

On slave3 node machine −

```
hostname slave3.in
```

Update **/etc/hosts** on all machines of the cluster with the following lines −

```
192.168.1.102 slave3.in slave3
```

Now try to ping the machine with hostnames to check whether it is resolving to IP or not.

On new node machine −

```
ping master.in
```

Start the DataNode on New Node

Start the datanode daemon manually using **$HADOOP_HOME/bin/hadoop-daemon.sh script**. It will automatically contact the master (NameNode) and join the cluster. We should also add the new node to the conf/slaves file in the master server. The script-based commands will recognize the new node.

**Login to new node**

su hadoop or ssh -X hadoop@192.168.1.103

**Start HDFS on a newly added slave node by using the following command**

./bin/hadoop-daemon.sh start datanode

**Check the output of jps command on a new node. It looks as follows.**

$ jps
7141 DataNode
10312 Jps

Removing a DataNode from the Hadoop Cluster

We can remove a node from a cluster on the fly, while it is running, without any data loss. HDFS provides a decommissioning feature, which ensures that removing a node is performed safely. To use it, follow the steps as given below −

**Step 1 − Login to master**

Login to master machine user where Hadoop is installed.

$ su hadoop

**Step 2 − Change cluster configuration**

An exclude file must be configured before starting the cluster. Add a key named dfs.hosts.exclude to our **$HADOOP_HOME/etc/hadoop/hdfs-site.xml** file. The value associated with this key provides the full path to a file on the NameNode's local file system which contains a list of machines which are not permitted to connect to HDFS.

For example, add these lines to **etc/hadoop/hdfs-site.xml** file.

```
<property>
  <name>dfs.hosts.exclude</name>
  <value>/home/hadoop/hadoop-1.2.1/hdfs_exclude.txt</value>
  <description>DFS exclude</description>
</property>
```

**Step 3 − Determine hosts to decommission**

Each machine to be decommissioned should be added to the file identified by the hdfs_exclude.txt, one domain name per line. This will prevent them from connecting to the NameNode. Content of the **"/home/hadoop/hadoop-1.2.1/hdfs_exclude.txt"** file is shown below, if you want to remove DataNode2.

slave2.in

**Step 4 − Force configuration reload**

Run the command **"$HADOOP_HOME/bin/hadoop dfsadmin -refreshNodes"** without the quotes.

$ $HADOOP_HOME/bin/hadoop dfsadmin -refreshNodes

This will force the NameNode to re-read its configuration, including the newly updated ‘excludes’ file. It will decommission the nodes over a period of time, allowing time for each node's blocks to be replicated onto machines which are scheduled to remain active.

On **slave2.in**, check the jps command output. After some time, you will see the DataNode process is shutdown automatically.

**Step 5 − Shutdown nodes**

After the decommission process has been completed, the decommissioned hardware can be safely shut down for maintenance. Run the report command to dfsadmin to check the status of decommission. The following command will describe the status of the decommission node and the connected nodes to the cluster.

```
$ $HADOOP_HOME/bin/hadoop dfsadmin -report
```

**Step 6 − Edit excludes file again**

Once the machines have been decommissioned, they can be removed from the ‘excludes’ file. Running **"$HADOOP_HOME/bin/hadoop dfsadmin -refreshNodes"** again will read the excludes file back into the NameNode; allowing the DataNodes to rejoin the cluster after the maintenance has been completed, or additional capacity is needed in the cluster again, etc.

**Special Note** − If the above process is followed and the tasktracker process is still running on the node, it needs to be shut down. One way is to disconnect the machine as we did in the above steps. The Master will recognize the process automatically and will declare as dead. There is no need to follow the same process for removing the tasktracker because it is NOT much crucial as compared to the DataNode. DataNode contains the data that you want to remove safely without any loss of data.

The tasktracker can be run/shutdown on the fly by the following command at any point of time.

```
$ $HADOOP_HOME/bin/hadoop-daemon.sh stop tasktracker
$HADOOP_HOME/bin/hadoop-daemon.sh start tasktracker
```

## Unit – VII

**Environment Setup : Pre-installation Setup, Installing Java Downloading Hadoop Hadoop Operation Modes Installing Hadoop in Standalone Mode Installing Hadoop in Pseudo Distributed Mode Verifying Hadoop Installation, Implement basic Hadoop commands on terminal**

### Pre-installation Setup

Hadoop is supported by GNU/Linux platform and its flavors. Therefore, we have to install a Linux operating system for setting up Hadoop environment. In case you have an OS other than Linux, you can install a Virtualbox software in it and have Linux inside the Virtualbox.

Pre-installation Setup

Before installing Hadoop into the Linux environment, we need to set up Linux using **ssh** (Secure Shell). Follow the steps given below for setting up the Linux environment.

### Creating a User

At the beginning, it is recommended to create a separate user for Hadoop to isolate Hadoop file system from Unix file system. Follow the steps given below to create a user −

- Open the root using the command “su”.
- Create a user from the root account using the command “useradd username”.
- Now you can open an existing user account using the command “su username”.

Open the Linux terminal and type the following commands to create a user.

```
$ su
  password:
# useradd hadoop
# passwd hadoop
  New passwd:
  Retype new passwd
```

SSH Setup and Key Generation

SSH setup is required to do different operations on a cluster such as starting, stopping, distributed daemon shell operations. To authenticate different users of Hadoop, it is required to provide public/private key pair for a Hadoop user and share it with different users.

The following commands are used for generating a key value pair using SSH. Copy the public keys form id_rsa.pub to authorized_keys, and provide the owner with read and write permissions to authorized_keys file respectively.

```
$ ssh-keygen -t rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

Installing Java

Java is the main prerequisite for Hadoop. First of all, you should verify the existence of java in your system using the command “java -version”. The syntax of java version command is given below.

```
$ java -version
```

If everything is in order, it will give you the following output.

```
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)
```

If java is not installed in your system, then follow the steps given below for installing java.

**Step 1**

Download java (JDK <latest version> - X64.tar.gz) by visiting the following link www.oracle.com

Then **jdk-7u71-linux-x64.tar.gz** will be downloaded into your system.

**Step 2**

Generally you will find the downloaded java file in Downloads folder. Verify it and extract the **jdk-7u71-linux-x64.gz** file using the following commands.

```
$ cd Downloads/
$ ls
jdk-7u71-linux-x64.gz

$ tar zxf jdk-7u71-linux-x64.gz
$ ls
jdk1.7.0_71   jdk-7u71-linux-x64.gz
```

**Step 3**

To make java available to all the users, you have to move it to the location “/usr/local/”. Open root, and type the following commands.

```
$ su
password:
# mv jdk1.7.0_71 /usr/local/
# exit
```

**Step 4**

For setting up **PATH** and **JAVA_HOME** variables, add the following commands to **~/.bashrc** file.

```
export JAVA_HOME=/usr/local/jdk1.7.0_71
export PATH=$PATH:$JAVA_HOME/bin
```

Now apply all the changes into the current running system.

```
$ source ~/.bashrc
```

**Step 5**

Use the following commands to configure java alternatives −

```
# alternatives --install /usr/bin/java java usr/local/java/bin/java 2
# alternatives --install /usr/bin/javac javac usr/local/java/bin/javac 2
```

```
# alternatives --install /usr/bin/jar jar usr/local/java/bin/jar 2

# alternatives --set java usr/local/java/bin/java
# alternatives --set javac usr/local/java/bin/javac
# alternatives --set jar usr/local/java/bin/jar
```

Now verify the java -version command from the terminal as explained above.

Downloading Hadoop

Download and extract Hadoop 2.4.1 from Apache software foundation using the following commands.

```
$ su
password:
# cd /usr/local
# wget http://apache.claz.org/hadoop/common/hadoop-2.4.1/
hadoop-2.4.1.tar.gz
# tar xzf hadoop-2.4.1.tar.gz
# mv hadoop-2.4.1/* to hadoop/
# exit
```

Hadoop Operation Modes

Once you have downloaded Hadoop, you can operate your Hadoop cluster in one of the three supported modes −

- **Local/Standalone Mode** − After downloading Hadoop in your system, by default, it is configured in a standalone mode and can be run as a single java process.
- **Pseudo Distributed Mode** − It is a distributed simulation on single machine. Each Hadoop daemon such as hdfs, yarn, MapReduce etc., will run as a separate java process. This mode is useful for development.
- **Fully Distributed Mode** − This mode is fully distributed with minimum two or more machines as a cluster. We will come across this mode in detail in the coming chapters.

Installing Hadoop in Standalone Mode

Here we will discuss the installation of **Hadoop 2.4.1** in standalone mode.

There are no daemons running and everything runs in a single JVM. Standalone mode is suitable for running MapReduce programs during development, since it is easy to test and debug them.

**Setting Up Hadoop**

You can set Hadoop environment variables by appending the following commands to **~/.bashrc** file.

```
export HADOOP_HOME=/usr/local/hadoop
```

Before proceeding further, you need to make sure that Hadoop is working fine. Just issue the following command −

```
$ hadoop version
```

If everything is fine with your setup, then you should see the following result −

```
Hadoop 2.4.1
Subversion https://svn.apache.org/repos/asf/hadoop/common -r 1529768
Compiled by hortonmu on 2013-10-07T06:28Z
Compiled with protoc 2.5.0
From source with checksum 79e53ce7994d1628b240f09af91e1af4
```

It means your Hadoop's standalone mode setup is working fine. By default, Hadoop is configured to run in a non-distributed mode on a single machine.

**Example**

Let's check a simple example of Hadoop. Hadoop installation delivers the following example MapReduce jar file, which provides basic functionality of MapReduce and can be used for calculating, like Pi value, word counts in a given list of files, etc.

```
$HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar
```

Let's have an input directory where we will push a few files and our requirement is to count the total number of words in those files. To calculate the total number of words, we do not need to write our MapReduce, provided the .jar file contains the implementation for word count. You can try other examples using the same .jar file; just issue the following commands to check supported MapReduce functional programs by hadoop-mapreduce-examples-2.2.0.jar file.

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduceexamples-2.2.0.jar
```

**Step 1**

Create temporary content files in the input directory. You can create this input directory anywhere you would like to work.

```
$ mkdir input
$ cp $HADOOP_HOME/*.txt input
$ ls -l input
```

It will give the following files in your input directory −

```
total 24
-rw-r--r-- 1 root root 15164 Feb 21 10:14 LICENSE.txt
-rw-r--r-- 1 root root   101 Feb 21 10:14 NOTICE.txt
-rw-r--r-- 1 root root  1366 Feb 21 10:14 README.txt
```

These files have been copied from the Hadoop installation home directory. For your experiment, you can have different and large sets of files.

**Step 2**

Let's start the Hadoop process to count the total number of words in all the files available in the input directory, as follows −

```
$ hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduceexamples-2.2.0.jar
wordcount input output
```

**Step 3**

Step-2 will do the required processing and save the output in output/part-r00000 file, which you can check by using −

```
$cat output/*
```

It will list down all the words along with their total counts available in all the files available in the input directory.

```
"AS      4
"Contribution" 1
"Contributor" 1
"Derivative 1
"Legal 1
"License"      1
"License");     1
"Licensor"      1
"NOTICE”        1
"Not      1
"Object"        1
"Source”        1
"Work”    1
"You"     1
"Your")   1
"[]"      1
"control"       1
"printed        1
"submitted"     1
(50%)     1
(BIS),    1
(C)       1
(Don't)   1
(ECCN)    1
(INCLUDING      2
(INCLUDING,     2
.....
```

Installing Hadoop in Pseudo Distributed Mode

Follow the steps given below to install Hadoop 2.4.1 in pseudo distributed mode.

**Step 1 − Setting Up Hadoop**

You can set Hadoop environment variables by appending the following commands to **~/.bashrc** file.

```
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME

export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_INSTALL=$HADOOP_HOME
```

Now apply all the changes into the current running system.

```
$ source ~/.bashrc
```

**Step 2 − Hadoop Configuration**

You can find all the Hadoop configuration files in the location "$HADOOP_HOME/etc/hadoop". It is required to make changes in those configuration files according to your Hadoop infrastructure.

$ cd $HADOOP_HOME/etc/hadoop

In order to develop Hadoop programs in java, you have to reset the java environment variables in **hadoop-env.sh** file by replacing **JAVA_HOME** value with the location of java in your system.

export JAVA_HOME=/usr/local/jdk1.7.0_71

The following are the list of files that you have to edit to configure Hadoop.

**core-site.xml**

The **core-site.xml** file contains information such as the port number used for Hadoop instance, memory allocated for the file system, memory limit for storing the data, and size of Read/Write buffers.

Open the core-site.xml and add the following properties in between <configuration>, </configuration> tags.

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

**hdfs-site.xml**

The **hdfs-site.xml** file contains information such as the value of replication data, namenode path, and datanode paths of your local file systems. It means the place where you want to store the Hadoop infrastructure.

Let us assume the following data.

dfs.replication (data replication value) = 1

(In the below given path /hadoop/ is the user name.
hadoopinfra/hdfs/namenode is the directory created by hdfs file system.)
namenode path = //home/hadoop/hadoopinfra/hdfs/namenode

(hadoopinfra/hdfs/datanode is the directory created by hdfs file system.)
datanode path = //home/hadoop/hadoopinfra/hdfs/datanode

Open this file and add the following properties in between the <configuration> </configuration> tags in this file.

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

  <property>
    <name>dfs.name.dir</name>
```

```
    <value>file:///home/hadoop/hadoopinfra/hdfs/namenode </value>
  </property>

  <property>
    <name>dfs.data.dir</name>
    <value>file:///home/hadoop/hadoopinfra/hdfs/datanode </value>
  </property>
</configuration>
```

**Note** − In the above file, all the property values are user-defined and you can make changes according to your Hadoop infrastructure.

**yarn-site.xml**

This file is used to configure yarn into Hadoop. Open the yarn-site.xml file and add the following properties in between the <configuration>, </configuration> tags in this file.

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

**mapred-site.xml**

This file is used to specify which MapReduce framework we are using. By default, Hadoop contains a template of yarn-site.xml. First of all, it is required to copy the file from **mapred-site.xml.template** to **mapred-site.xml** file using the following command.

```
$ cp mapred-site.xml.template mapred-site.xml
```

Open **mapred-site.xml** file and add the following properties in between the <configuration>, </configuration>tags in this file.

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Verifying Hadoop Installation

The following steps are used to verify the Hadoop installation.

**Step 1 − Name Node Setup**

Set up the namenode using the command “hdfs namenode -format” as follows.

```
$ cd ~
$ hdfs namenode -format
```

The expected result is as follows.

```
10/24/14 21:30:55 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
```

```
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost/192.168.1.11
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.4.1
...
...
10/24/14 21:30:56 INFO common.Storage: Storage directory
/home/hadoop/hadoopinfra/hdfs/namenode has been successfully formatted.
10/24/14 21:30:56 INFO namenode.NNStorageRetentionManager: Going to
retain 1 images with txid >= 0
10/24/14 21:30:56 INFO util.ExitUtil: Exiting with status 0
10/24/14 21:30:56 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at localhost/192.168.1.11
************************************************************/
```

**Step 2 − Verifying Hadoop dfs**

The following command is used to start dfs. Executing this command will start your Hadoop file system.

```
$ start-dfs.sh
```

The expected output is as follows −

```
10/24/14 21:37:56
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/hadoop/hadoop
2.4.1/logs/hadoop-hadoop-namenode-localhost.out
localhost: starting datanode, logging to /home/hadoop/hadoop
2.4.1/logs/hadoop-hadoop-datanode-localhost.out
Starting secondary namenodes [0.0.0.0]
```

**Step 3 − Verifying Yarn Script**

The following command is used to start the yarn script. Executing this command will start your yarn daemons.

```
$ start-yarn.sh
```

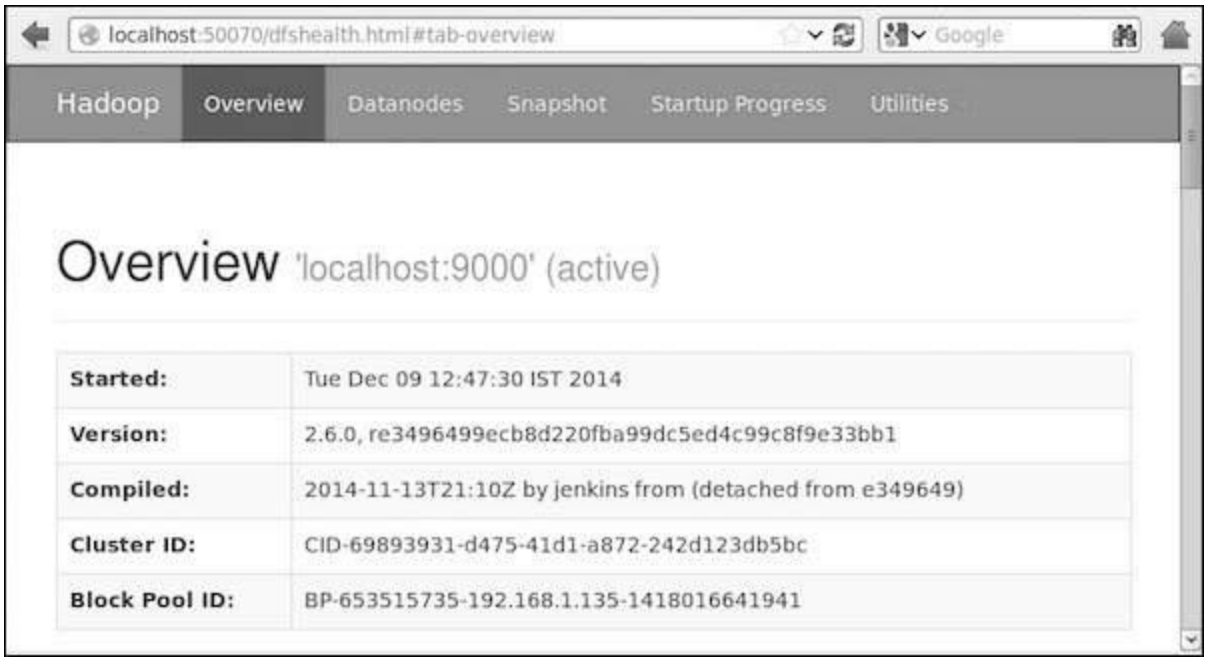
The expected output as follows −

```
starting yarn daemons
starting resourcemanager, logging to /home/hadoop/hadoop
2.4.1/logs/yarn-hadoop-resourcemanager-localhost.out
localhost: starting nodemanager, logging to /home/hadoop/hadoop
2.4.1/logs/yarn-hadoop-nodemanager-localhost.out
```

**Step 4 − Accessing Hadoop on Browser**

The default port number to access Hadoop is 50070. Use the following url to get Hadoop services on browser.

```
http://localhost:50070/
```



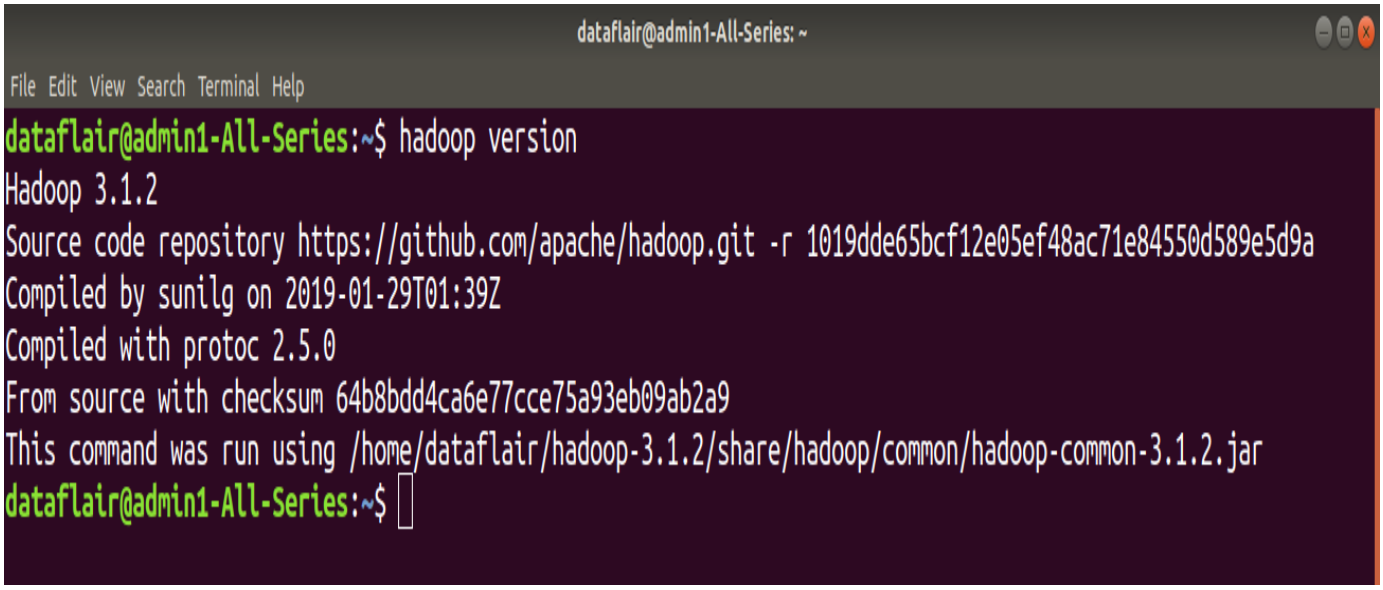

**Step 5 − Verify All Applications for Cluster**

The default port number to access all applications of cluster is 8088. Use the following url to visit this service.

http://localhost:8088/


**Implement basic Hadoop Command on Terminal**

**1. hadoop version**




**Hadoop HDFS version Command Description:**
The Hadoop fs shell command **version** prints the Hadoop version.

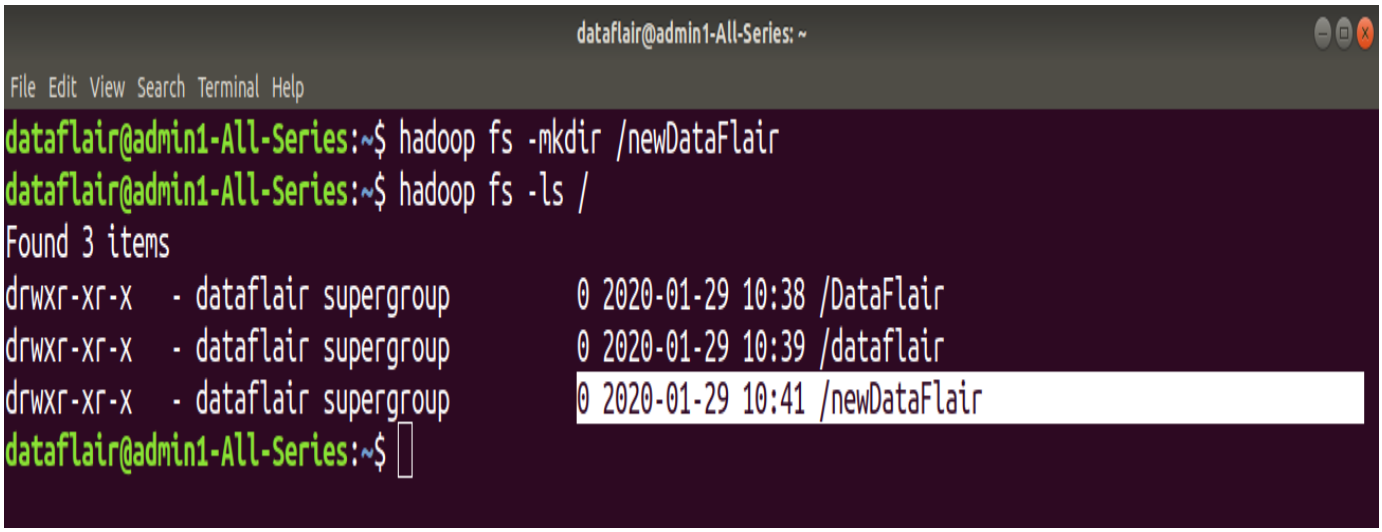
2. **mkdir**

**Hadoop HDFS mkdir Command Usage:**

hadoop fs –mkdir /path/directory_name

In this example, we are trying to create a newDataFlair named directory in HDFS using the **mkdir** command.


Using the **ls** command, we can check for the directories in HDFS.




**Hadoop HDFS mkdir Command Description:**
This command creates the directory in HDFS if it does not already exist.

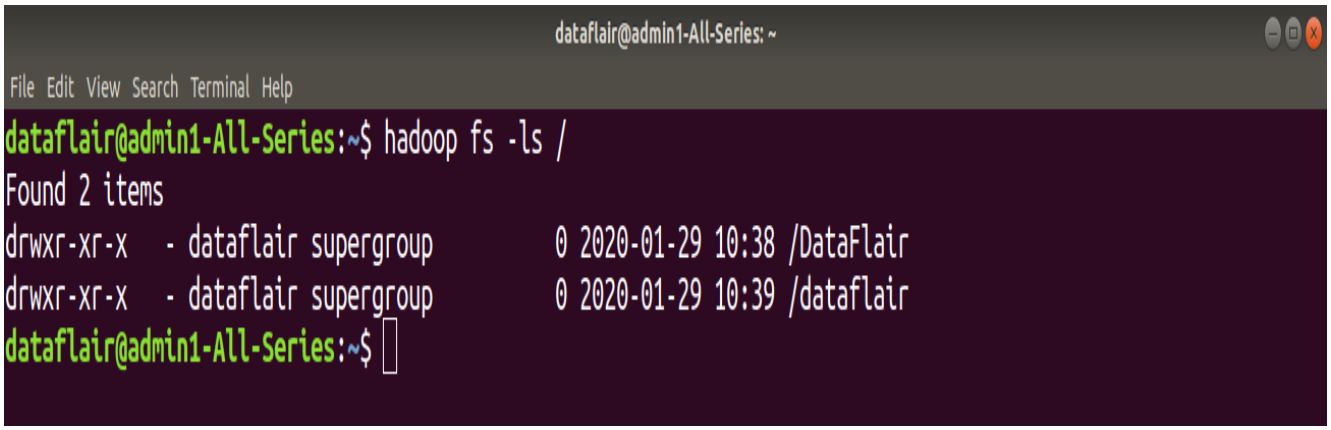
**Note:** If the directory already exists in HDFS, then we will get an error message that file already exists.
Use **hadoop fs mkdir -p /path/directoryname**, so not to fail even if directory exists.

**3. ls**

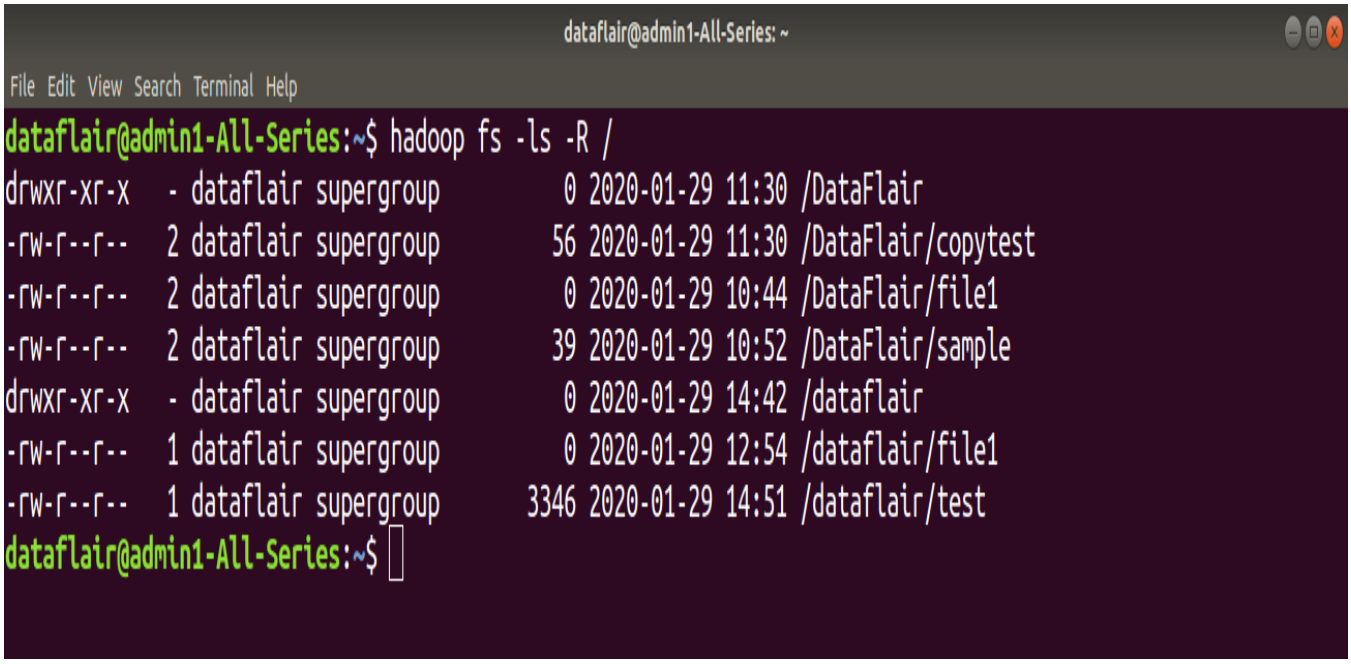
**Hadoop HDFS ls Command Usage:**

hadoop fs -ls /path

**Hadoop HDFS ls Command Example 1:**
Here in the below example, we are using the **ls** command to enlist the files and directories present in HDFS.




**Hadoop HDFS ls Command Description:**
The Hadoop fs shell command **ls** displays a list of the contents of a directory specified in the path provided by the user. It shows the name, permissions, owner, size, and modification date for each file or directories in the specified directory.
**Hadoop HDFS ls Command Example 2:**



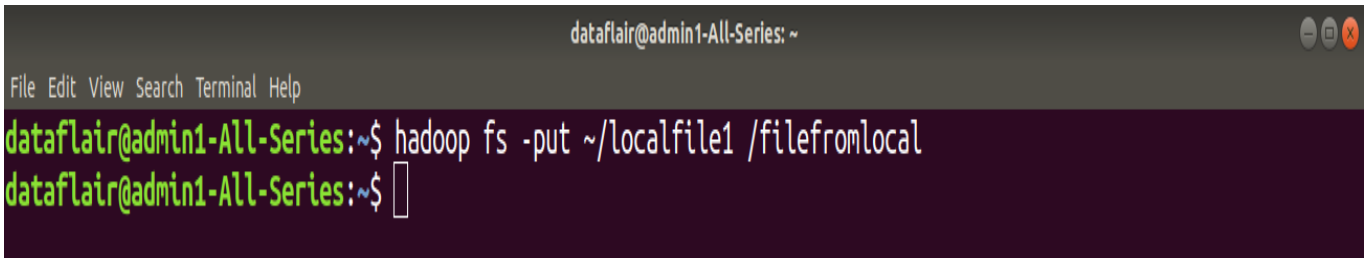

**Hadoop HDFS ls Description:**
This Hadoop fs command behaves like **-ls,** but recursively displays entries in all subdirectories of a path.

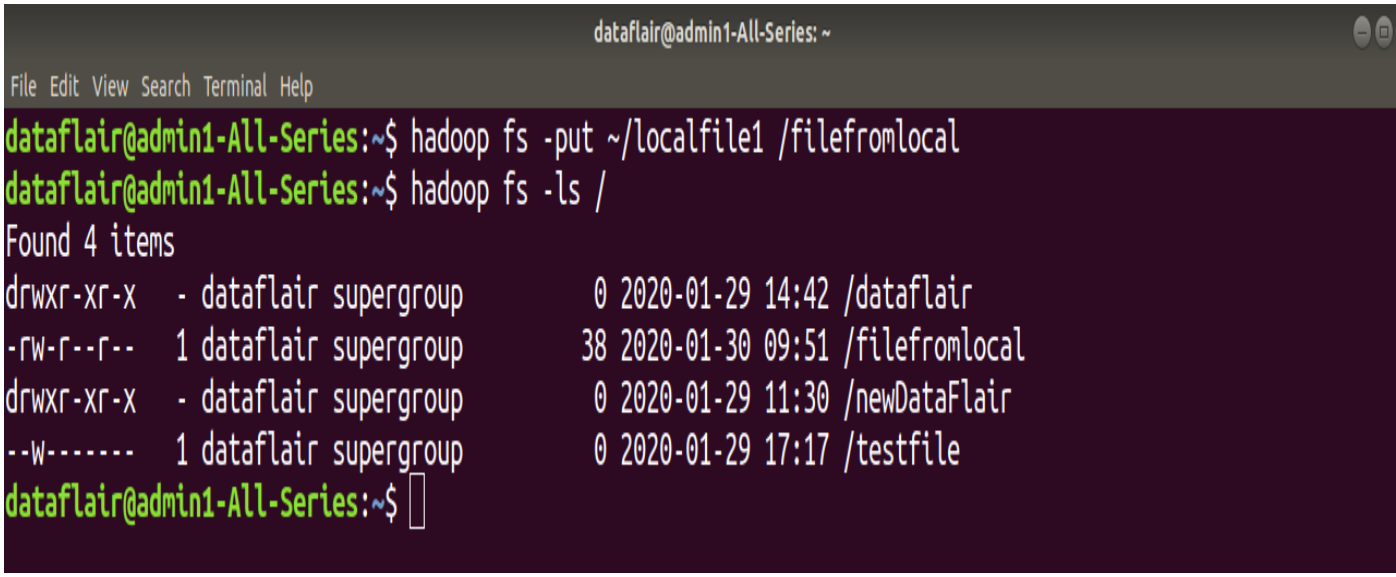
**4. put**

**Hadoop HDFS put Command Usage:**

haoop fs -put <localsrc> <dest>

**Hadoop HDFS put Command Example:**
Here in this example, we are trying to copy localfile1 of the local file system to the Hadoop filesystem.






**Hadoop HDFS put Command Description:**

The Hadoop fs shell command **put** is similar to the **copyFromLocal**, which copies files or directory from the local filesystem to the destination in the Hadoop filesystem.

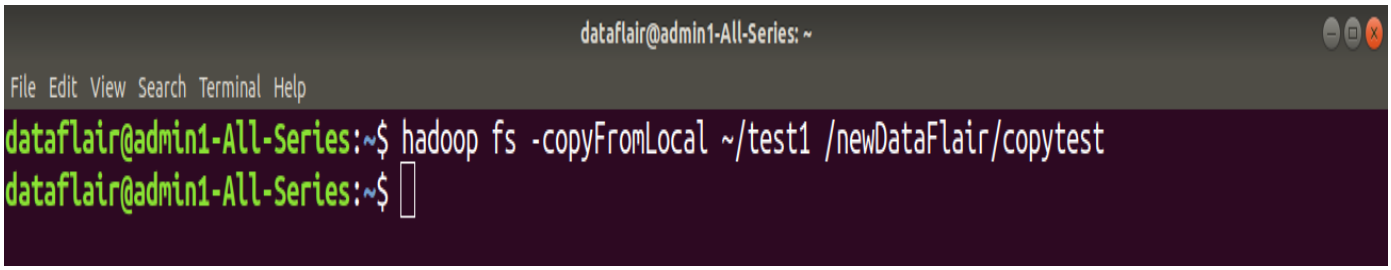
**5. copyFromLocal**

**Hadoop HDFS copyFromLocal Command Usage:**

hadoop fs -copyFromLocal <localsrc> <hdfs destination>

**Hadoop HDFS copyFromLocal Command Example:**

Here in the below example, we are trying to copy the 'test1' file present in the local file system to the newDataFlair directory of Hadoop.




**Hadoop HDFS copyFromLocal Command Description:**
This command copies the file from the local file system to HDFS.

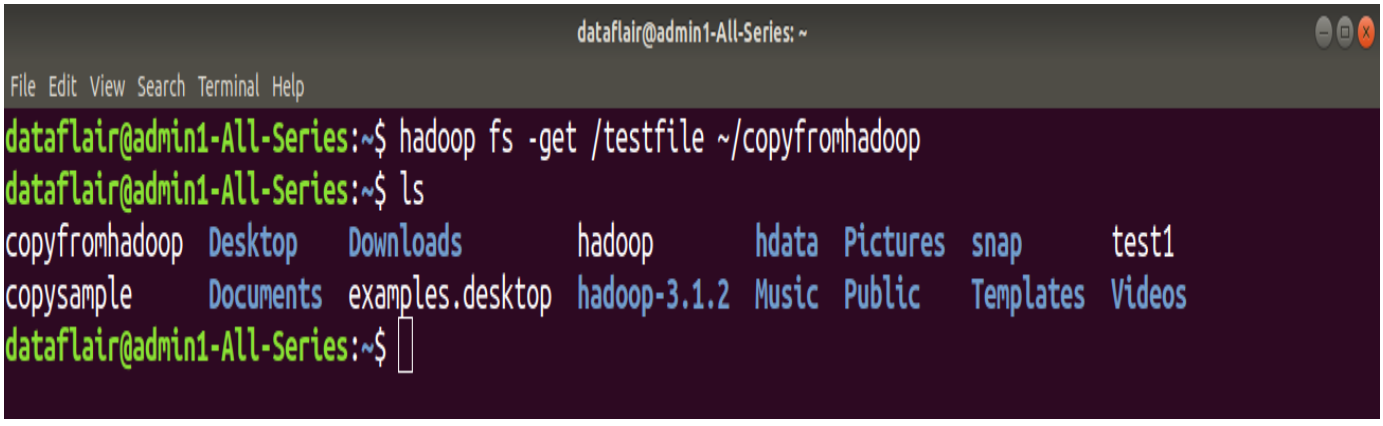
**6. get**

**Hadoop HDFS get Command Usage:**

hadoop fs -get <src> <localdest>

**Hadoop HDFS get Command Example:**
In this example, we are trying to copy the 'testfile' of the hadoop filesystem to the local file system.

**Hadoop HDFS get Command Description:**
The Hadoop fs shell command get copies the file or directory from the Hadoop file system to the local file system.

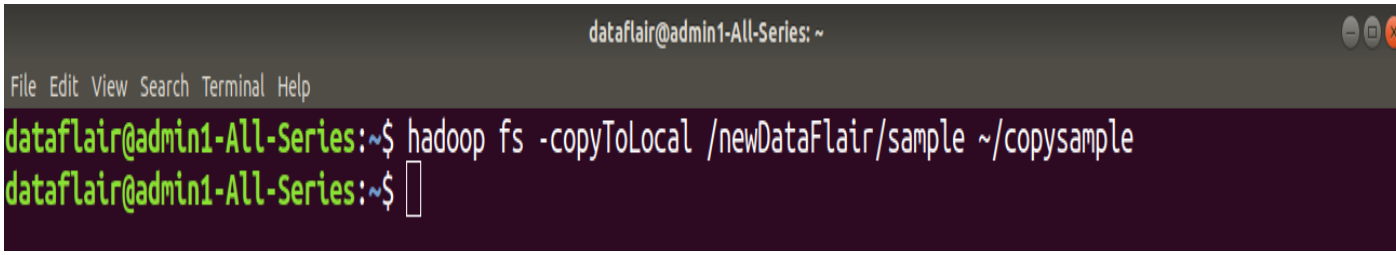



**7. copyToLocal**

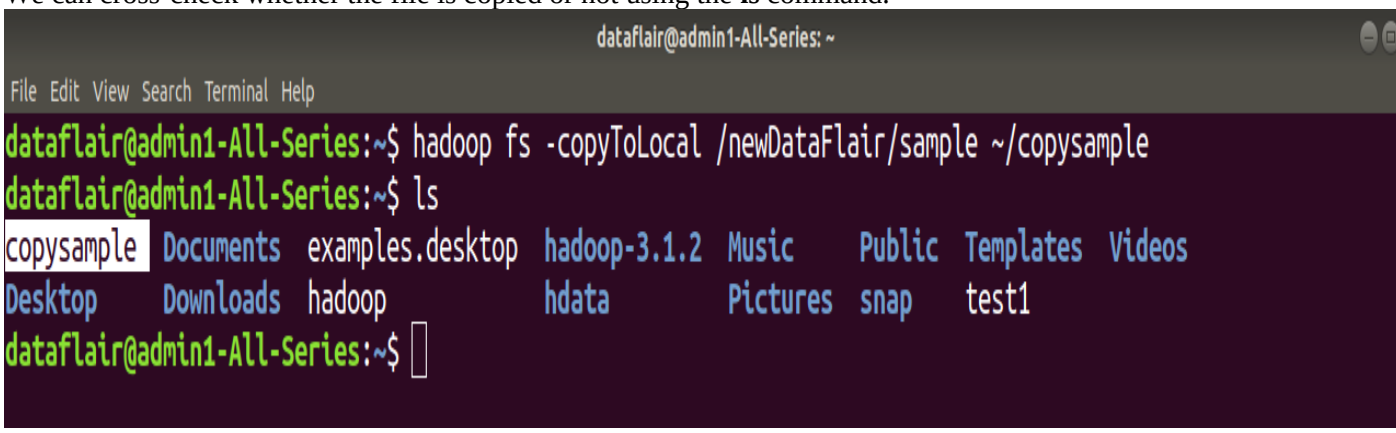
**Hadoop HDFS copyToLocal Command Usage:**

hadoop fs -copyToLocal <hdfs source> <localdst>

**Hadoop HDFS copyToLocal Command Example:**
Here in this example, we are trying to copy the 'sample' file present in the newDataFlair directory of HDFS to the local file system.




We can cross-check whether the file is copied or not using the **ls** command.




**Hadoop HDFS copyToLocal Description:**
**copyToLocal** command copies the file from HDFS to the local file system.

**8. cat**

**Hadoop HDFS cat Command Usage:**

hadoop fs –cat /path_to_file_in_hdfs

**Hadoop HDFS cat Command Example:**
Here in this example, we are using the cat command to display the content of the 'sample' file present in newDataFlair directory of HDFS.


**Hadoop HDFS cat Command Description:**

The **cat** command reads the file in HDFS and displays the content of the file on console or stdout.

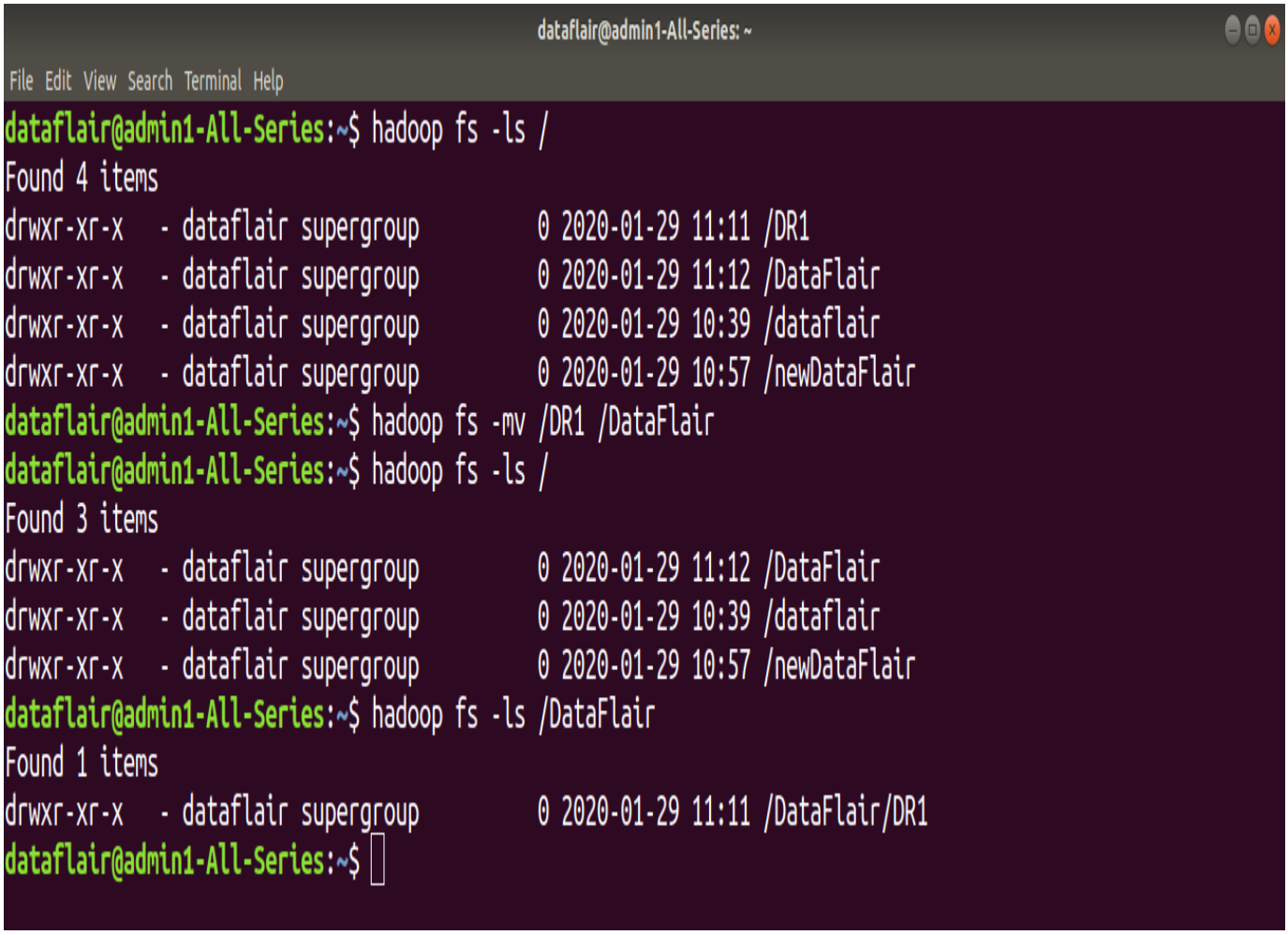
**9. mv**

**Hadoop HDFS mv Command Usage:**

hadoop fs -mv <src> <dest>

**Hadoop HDFS mv Command Example:**

In this example, we have a directory 'DR1' in HDFS. We are using **mv** command to move the DR1 directory to the DataFlair directory in HDFS.



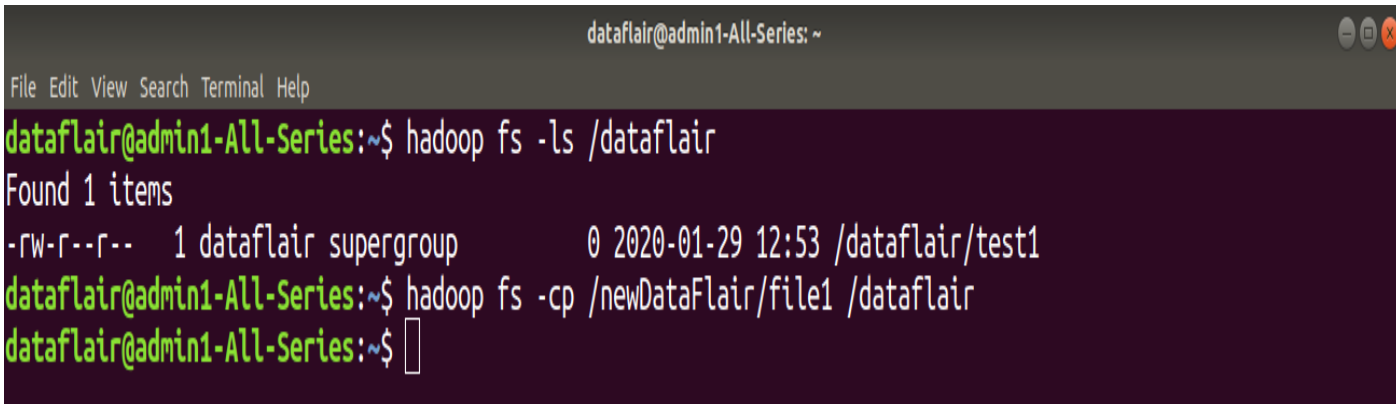

**Hadoop HDFS mv Command Description:**
The HDFS mv command moves the files or directories from the source to a destination within **HDFS.**

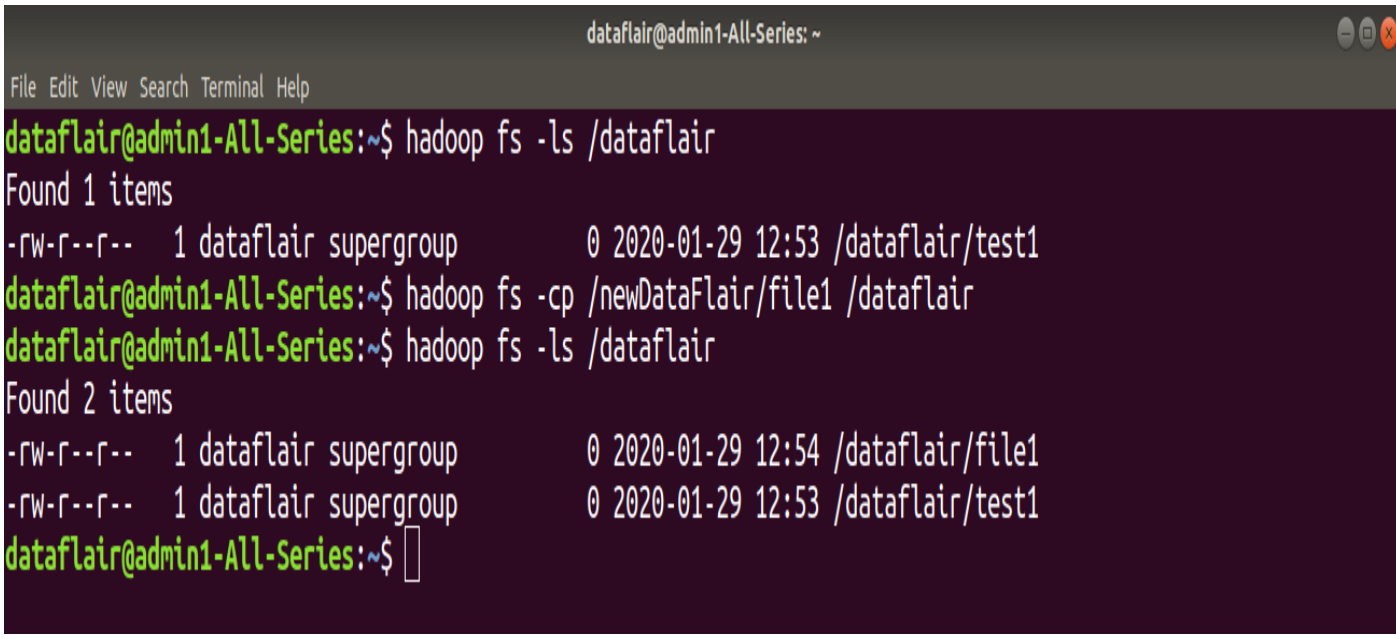
**10. cp**

**Hadoop HDFS cp Command Usage:**

hadoop fs -cp <src> <dest>

**Hadoop HDFS cp Command Example:**
In the below example we are copying the 'file1' present in newDataFlair directory in HDFS to the dataflair directory of HDFS.




**Hadoop HDFS cp Command Description:**
The **cp** command copies a file from one directory to another directory within the HDFS.

## Question Bank

**1. What is Big Data?**
Big data is a field that treats ways to analyze, systematically extract information from, or otherwise deal with data sets that are too large or complex to be dealt with by traditional data-processing application software.

**2. List out the best practices of Big Data Analytics.**
1. Start at the End
2. Build an Analytical Culture.
3. Re-Engineer Data Systems for Analytics
4. Focus on Useful Data Islands.
5. Iterate often.

**3. Write down the characteristics of Big Data Applications.**
**a)** Data Throttling
b) Computation- restricted throttling
c) Large Data Volumes
d) Significant Data Variety
e) Benefits from Data parallelization

**Write down the four computing resources of Big Data Storage.**
a) Processing Capability
b) Memory
c) Storage
d) Network

**5. What is HDFS?**
Apache Hadoop is a collection of open-source software utilities that facilitate using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model.

**6. What is MapReduce?**
MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes

a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).

**7. What is YARN?**

YARN is an Apache Hadoop technology and stands for Yet Another Resource Negotiator. YARN is a large-scale, distributed operating system for big data applications. YARN is a software rewrite that is capable of decoupling MapReduce's resource management and scheduling capabilities from the data processing component.

**8. What is Map Reduce Programming Model?**

MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster. The model is a specialization of the split-apply-combine strategy for data analysis.

**9. What are the characteristics of big data?**

Big data can be described by the following characteristics

**Volume -** The quantity of data generated and stored data. The size of the data determines the value and potential insight- and whether it can actually be considered big data or not.

**Variety -**The type and nature of the data. This helps people who analyze it to effectively use the resulting insight.

**Velocity -**In this context, the speed at which the data is generated and processed to meet the demands and challenges that lie in the path of growth and development.

**Variability-** Inconsistency of the data set can hamper processes to handle and manage it.

**Veracity-**The data quality of captured data can vary greatly, affecting the accurate analysis

**10. What is Big Data Platform?**

• Big Data Platform is integrated IT solution for Big Data management which combines several software systems, software tools and hardware to provide easy to use tools system to enterprises.

• It is a single one-stop solution for all Big Data needs of an enterprise irrespective of size and data volume. Big Data Platform is enterprise class IT solution for developing, deploying and managing Big Data.

**11. What is Bigdata? Describe the main features of a big data in detail.**

**Basics of Bigdata Platform**

• Big Data platform is IT solution which combines several Big Data tools and utilities into one packaged solution for managing and analyzing Big Data.

• **Big data platform** is a type of IT solution that combines the features and capabilities of several **big data** application and utilities within a single solution.

• It is an enterprise class IT **platform** that enables organization in developing, deploying, operating and managing a **big data** infrastructure /environment.

**Big Data Platform**

• Big Data Platform is integrated IT solution for Big Data management which combines several software systems, software tools and hardware to provide easy to use tools system to enterprises.

• It is a single one-stop solution for all Big Data needs of an enterprise irrespective of size and data volume. Big Data Platform is enterprise class IT solution for developing, deploying and managing Big Data.

• There are several Open source and commercial Big Data Platform in the market with varied features which can be used in Big Data environment.

• Big data platform is a type of IT solution that combines the features and capabilities of several big data application and utilities within a single solution.

• It is an enterprise class ITplatformthat enables organization in developing, deploying, operating and managing abig data infrastructure /environment.
• Big data platform generally consists of big data storage, servers, database, big data management, business intelligence and other big data management utilities
• It also supports custom development, querying and integration with other systems.
• The primary benefit behind a big data platform is to reduce the complexity of multiple vendors/ solutions into a one cohesive solution.
• Big data platform are also delivered through cloud where the provider provides an all-inclusive big data solutions and services.

**Features of Big Data Platform**
Here are most important features of any good Big Data Analytics Platform:
a) Big Data platform should be able to accommodate new platforms and tool basedon the business requirement. Because business needs can change due to new technologies or due to change in business process.
b) It should support linear scale-out
c) It should have capability for rapid deployment
d) It should support variety of data format
e) Platform should provide data analysis and reporting tools
f) It should provide real-time data analysis software
g) It should have tools for searching the data through large data sets

Big data is a term for data sets that are so large or complex that traditional data processing applications are inadequate.
Challenges include
• Analysis,
• Capture,
• Data Curation,
• Search,
• Sharing,
• Storage,
• Transfer,
• Visualization,
• Querying,
• Updating
Information Privacy.
• The term often refers simply to the use of predictive analytics or certain other ***advancedmethods*** to extract value from data, and seldom to a particular size of data set.
• **ACCURACY** in big data may lead to more confident decision making, and better decisions can result in greater operational efficiency, cost reduction and reduced risk.
• Big data usually includes data sets with sizes beyond the ability of commonly used
• software tools to capture, curate, manage, and process data within a tolerable elapsed
• time. Big data "size" is a constantly moving target.
• Big data requires a set of techniques and technologies with new forms of integration to
• reveal insights from datasets that are diverse, complex, and of a massive scale

**List of BigData Platforms**
a) Hadoop
b) Cloudera
c) Amazon Web Services
d) Hortonworks

e) MapR
f) IBM Open Platform
g) Microsoft HDInsight
h) Intel Distribution for Apache Hadoop
i) Datastax Enterprise Analytics
j) Teradata Enterprise Access for Hadoop
k) Pivotal HD

a) Hadoop
• Hadoop is open-source, Java based programming framework and server software which is used to save and analyze data with the help of 100s or even 1000s of commodity servers in a clustered environment.
• Hadoop is designed to storage and process large datasets extremely fast and in fault tolerant way.
• Hadoop uses HDFS (Hadoop File System) for storing data on cluster of commodity computers. If any server goes down it know how to replicate the data and there is no loss of data even in hardware failure.
• Hadoop is Apache sponsored project and it consists of many software packages which runs on the top of the Apache Hadoop system.
• Top Hadoop based Commercial Big Data Analytics Platform
• Hadoop provides set of tools and software for making the backbone of the Big Data analytics system.
• Hadoop ecosystem provides necessary tools and software for handling and analyzing Big Data.
• On the top of the Hadoop system many applications can be developed and plugged-in to provide ideal solution for Big Data needs.

b) **Cloudera**
• Cloudra is one of the first commercial Hadoop based Big Data Analytics Platform offering Big Data solution.
• Its product range includes Cloudera Analytic DB, Cloudera Operational DB, Cloudera Data Science & Engineering and Cloudera Essentials.
• All these products are based on the Apache Hadoop and provides real-time processing and analytics of massive data sets.

c) **Amazon Web Services**
• Amazon is offering Hadoop environment in cloud as part of its Amazon Web Services package.
• AWS Hadoop solution is hosted solution which runs on Amazon's Elastic Cloud Compute and Simple Storage Service (S3).
• Enterprises can use the Amazon AWS to run their Big Data processing analytics in the cloud environment.
• Amazon EMR allows companies to setup and easily scale Apache Hadoop, Spark, HBase, Presto, Hive, and other Big Data Frameworks using its cloud hosting environment.

d) **Hortonworks**
• Hortonworks is using 100% open-source software without any propriety software. Hortonworks were the one who first integrated support for Apache HCatalog.
• The Hortonworks is a Big Data company based in California.
• This company is developing and supports application for Apache Hadoop.
Hortonworks Hadoop distribution is 100% open source and its enterprise ready with following features:
• Centralized management and configuration of clusters
• Security and data governance are built in feature of the system
• Centralized security administration across the system

e) **MapR**

• MapR is another Big Data platform which us using the Unix file system for handling data.
• It is not using HDFS and this system is easy to learn anyone familiar with the Unix system.
• This solution integrates Hadoop, Spark, and Apache Drill with a real-time data processing feature.

**12. Explain in detail about Storage Considerations in Big Data.**

In any environment intended to support the analysis of massive amounts of data, there must be the infrastructure supporting the data lifecycle from acquisition, preparation, integration, and execution. The need to acquire and manage massive amounts of data suggests a need for specialty storage systems to accommodate the big data applications. When evaluating specialty storage offerings, some variables to
consider include:

• **Scalability**, which looks at whether expectations for performance improvement are aligned with the additional of storage resources, and the degree to which the storage subsystem can support massive data volumes of increasing size.
• **Extensibility**, which examines how flexible the storage system's architecture is in allowing the system to be grown without the constraint of artificial limits.
• **Accessibility**, which looks at any limitations or constraints in providing simultaneous access to an expanding user community without compromising performance.
• **Fault tolerance**, which imbues the storage environment with the capability to recover from intermittent failures.
• High-speed I/O capacity, which measures whether the input/output channels can satisfy the demanding timing requirements for absorbing, storing, and sharing large data volumes.
• I**ntegrability**, which measures how well the storage environment can be integrated into the production environment. Often, the storage framework involves a software layer for managing a collection of storage resources and providing much of these capabilities. The software configures storage for replication to provide a level of fault tolerance, as well as managing communications using standard protocols (such as UDP or TCP/IP) among the different processing nodes. In addition, some frameworks will replicate stored data, providing redundancy in the event of a fault or failure.

**Sample Questions**

1. Discuss the following in detail

   a. Conventional challenges in big data

   b. Nature of Data

2. Describe any five characteristics of Big Data.

3. a) What is Hadoop? Explain its components

   b) How do you analyze the data in hadoop.

4. Explain the following
   a. Mapper class

   b. Reducer class

5. Define HDFS. Describe namenode, datanode and block. Explain HDFS operations in detail.

6. What is Cluster? Explain the setting up a Hadoop cluster.

7. What are the different types of Hadoop configuration files? Discuss.

8. a) What is PIG ?Explain its installing process.
   b) Explain two execution types or modes in PIG.

9. Explain the process of installing HIVE & features of HIVE.

10. Give a detail note on HBASE.

11. What is the role of Data node and Name node in HDFS?

12. What is a PIG? Specify its Role in Hadoop?

13. How to create and manage data bases in HIVE?

14. What are the modes that a Hadoop can run?

15. Discuss in brief about the building blocks of Hadoop?

16. Discuss in brief about the implementation of map reduce concept with suitable example.

17. Explain with an example, how Hadoop uses Scale out feature to improve the performance?

18. Differentiate between HDFS and GFS.

19. Discuss in brief about the operational modes in Hadoop cluster configuration.

20. What are the real time industry applications of Hadoop?

21. Explain in brief about Name node, Data Node and Secondary Name node in HDFS.

22. Describe in brief about the PIG Architecture.

23. Define structured, semi structured and un-structured data with examples?

24. What are the three key design principles PIG Latin?

25. What are the advantages and disadvantages of Hadoop?

26. Define Data node? How does name node tackle data node failures?

27. Discuss in brief about the Name node, Check point name node and back up node?

28. What are the different modes in which hadoop can be installed and what is the use of each mode from application and developer point of view?

29. Discuss in brief about the Architecture of HIVE.

30. What is Hive meta store? Which classes are used by the Hive to Read and Write HDFS Files?

31. What is a HIVE? Specify its Role in Hadoop.

32. Explain Basic command on Hadoop Terminal

33. Installation Steps of Hadoop, mapping a nodes and configuration of Hadoop.

34. How to Start Hadoop Services, Adding New Data Node in the Hadoop Cluster, Removing New Data Node from the Hadoop Cluster.

**Checklist for Course Packs**

Title page should be standardized bearing title of subject, course, course code, semester, year of batch

- Name of the instructor teaching thecourse
- Name of the courseleader

Forwarding by HOD bearing his/her signature for approval by

Director Sir Logo of BVIMR, name of institution, address

Warning "strictly for internal use" must be printed on the

front title page Table of content bearing

- Serialno.
- Contents
- Pageno.

Copy of latest syllabus of course as specified

by university Lesson plan bearing

- Introduction tocourse
- Courseobjectives
- Learningoutcomes

List of topics/modules

with Content Evaluation

Criteria

- CES evaluationdescription
- Recommended text books and referencebooks
- Internetresource
- Swayam coursesSession

Plan bearing

- Sessionnumber
- Topic
- Reading/caserequired
- Pedagogyfollowed
- Learningoutcome

Contact details of instructor along with profile

Main body of course pack having reading material, exercises.

**Declaration by Faculty** I, **Rajni Dubey**, **Assistant Professor**, Teaching **Hadoop** subject in **BCA VI** have incorporated all the necessary pages/ sections/ question papers mentioned in this check list above