



INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal, Hyderabad - 500 043

Lab Manual:

BIG DATA AND ANALYTICS LABORATORY(ACSC34)

Prepared by

Ms. K ASWINI(IARE10750)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INSTITUTE OF AERONAUTICAL ENGINEERING

August 28, 2024

Contents

Course Information

1. Introduction	i
1.1 Student Responsibilities	ii
1.2 Laboratory Teaching Assistant Responsibilities	iii
1.3 Faculty Coordinator Responsibilities	iv
1.4 Lab Policy and Grading	v
1.5 Course Goals and Objectives	vi
1.6 Use of Laboratory Instruments	vii
1.7 Instrument Protection Rules	viii
2. Data Recording and Reports	ix
2.1 The Laboratory Notebook	x
2.2 TheLabReport	xi
2.3 Order of Lab Report Components	xii

LAB 1 - INSTALL VMWARE	10
LAB 2 - HADOOP MODES	13
LAB 3- USING LINUX OPERATING SYSTEM	16
LAB 4 - FILE MANAGEMENT IN HADOOP	20
LAB 5- MAPREDUCE PROGRAM 1	23
LAB 6- MAPREDUCE PROGRAM 2	26
LAB 7- MAPREDUCE PROGRAM 3	29
LAB 8 - PIG LATIN LANGUAGE – PIG	33
LAB 9- PIG COMMANDS	37
LAB 10- PIG LATIN MODES, PROGRAMS	41
LAB 11- HIVE	45
LAB 12 -HIVE OPERATIONS	48

Course Information

1 Introduction

This course is all about Business Analytics which gives exposure to various types of Business analytics, types of data, data sources, understanding of Big data and Big data analytics. One has to take this course as huge volume of Data brings endless opportunities with it . Data is an asset with enormous Insights hidden within it . To bring value out of Data and to transform it to Information and knowledge we need to understand Business and Data analytics along with tools associated with it. How the student performs in the lab depends on his/her preparation and participation. Each student must participate in all aspects of the lab to insure a thorough understanding of the concepts. The student, lab teaching assistant, and faculty coordinator all have certain responsibilities toward successful completion of the lab's goals and objectives.

1.1 Student Responsibilities

The student is expected to be prepared for each lab. Lab preparation includes reading the lab experiment and related textbook material. If you have questions or problems with the preparation, contact your Laboratory Teaching Assistant (LTA), but in a timely manner. Do not wait until an hour or two before the lab and then expect the LTA to be immediately available.

Active participation by each student in lab activities is expected. The student is expected to ask the LTA any questions they may have. Do not make costly mistakes because you did not ask a question before proceeding.

A large portion of the student's grade is determined in the comprehensive final exam, resulting in a requirement of understanding the concepts and procedure of each lab experiment for the successful completion of the lab class. The student should remain alert and use common sense while performing a lab experiment. They are also responsible for keeping a professional and accurate record of the lab experiments in the lab manual wherever tables are provided. Students should report any errors in the lab manual to the teaching assistant.

1.2 Laboratory Faculty Responsibilities

The LTA shall be completely familiar with each lab prior to class. The LTA shall provide the students with a syllabus and safety review during the first class. The syllabus shall include the LTA's office hours, telephone number, and the name of the faculty coordinator. The LTA is responsible for ensuring that all the necessary equipment and/or preparations for the lab are available and in working condition. Lab experiments should be checked in advance to make sure everything is in working order. The LTA should fully answer any questions posed by the students and supervise the students performing the lab experiments. The LTA is expected to grade the lab notebooks and reports in a fair and timely manner. The reports should be returned to the students in the next lab period following submission. The LTA should report any errors in the lab manual to the faculty coordinator.

1.3 Faculty Co-ordinator Responsibilities

The faculty coordinator should ensure that the laboratory is properly equipped, i.e., that the teaching assistants receive any equipment necessary to perform the experiments. The coordinator is responsible for supervising the teaching assistants and resolving any questions or problems that are identified by the teaching assistants or the students. The coordinator may supervise the format of the final exam for the lab. They are also responsible for making any necessary corrections to this manual and ensuring that it is continually updated and available.

1.4 Lab Policy and Grading

The student should understand the following policy:

Attendance

Attendance is mandatory and any absence must be for a valid excuse and must be documented. If the instructor is more than 15 minutes late, students may consider lab for the day cancelled.

Lab Records

The student must:

1. Perform the Pre Lab assignment before the beginning of each lab,
2. Keep all work in preparation of and obtained during lab
3. Prepare a lab report on experiments selected by the laboratory faculty.

Grading Policy

The final grade of this course is determined using the criterion detailed in the syllabus.

Instructions to Students

- Before entering the lab the student should carry the following things (MANDATORY)
 - Identity card issued by the college.
 - Work Sheets
- Student must sign in and sign out in the register provided when attending the lab session without fail.
- Come to the laboratory in time. Students, who are late more than 15 min., will not be allowed to attend the lab.
- Students need to maintain 100% attendance in lab if not a strict action will be taken.
- All students must follow a Dress Code while in the laboratory
- Foods, drinks are NOT allowed.
- All bags must be left at the indicated place.
- Refer to the lab staff if you need any help in using the lab.
- Respect the laboratory and its other users.
- Workspace must be kept clean and tidy after experiment is completed.
- Read the Manual carefully before coming to the laboratory and be sure about what you are supposed to do.
- Do the experiments as per the instructions given in the manual.
- Copy all the programs to observation which are taught in class before attending the lab session.
- Students are not supposed to use floppy disks, pen drives without permission of lab- incharge.
- Lab records need to be submitted on or before the date of submission.
- Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
- Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
- Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
- Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

1.5 Course Goals and Objectives

Goal:

The big data and business analytics laboratory is designed to provide the student with the knowledge how to install VM Ware software and different modes of hadoop ,and also learn how to execute basic commands in linux operating systems like creation deletion operations and students will learn how to manage file system and also learn pig latin language and hive system in hadoop In addition, the student should learn how to record execution results effectively and present these results in a written report. More explicitly, the class

Objectives:

Students will try to learn:

- Optimize business decisions and create competitive advantage with Big data analytics.
- Practice java concepts required for developing map reduce programs.
- Impart the architectural concepts of Hadoop and introducing map reduce paradigm.
- Practice programming tools PIG and HIVE in Hadoop eco system.
- Implement best practices for Hadoop development.

2 Data Recording and Reports

2.1 The Laboratory Note book

Students must write their experimental outputs in the provided tables in this laboratory manual and reproduce them in the lab reports. Reports are integral to recording the methodology and results of an experiment. In engineering practice, the laboratory notebook serves as an invaluable reference to the technique used in the lab and is essential when trying to duplicate a result or write a report. Therefore, it is important to learn to keep accurate data. Make plots of data and sketches when these are appropriate in the recording and analysis of observations. Note that the data collected will be an accurate and permanent record of the data obtained during the experiment and the analysis of the results. You will need this record when you are ready to prepare a lab report.

2.2 The Lab Report

Reports are the primary means of communicating your experience and conclusions to other professionals. In this course you will use the lab report to inform your laboratory faculty about what you did and what you have learned from the experience. Engineering results are meaningless unless they can be communicated to others. You will be directed by your laboratory faculty to prepare a lab report on a few selected lab experiments during the semester. Your assignment might be different from your lab partner's assignment.

Your laboratory report should be clear and concise. The lab report shall be typed on a word processor. As a guide, use the format on the next page. Use tables, diagrams, sketches, as necessary to show what you did, what was observed, and what conclusions you can draw from this. Even though you will work with one or more lab partners, your report will be the result of your individual effort in order to provide you with practice in technical communication.

Formatting and Style

- The lab report shall be typed in a word processor.
- All page margins must be 1.25 inches. All content (including text, figures, tables, etc.) must fit within the margins.
- Body text should be double-spaced.
- Basic text should be in 12-point size in a commonly used text font.
- Set your main text justified (with even left/right margins).
- The first line of each paragraph should have a left indent.
- All the tables should have titles and should be numbered. Tables should be labeled numerically as Table 1, Table 2, etc. Table captions appear above the table. The column headings should be labeled with the units specified.
- Use MS-Word equation (under Insert Equation menu), MathType, or a similar tool to type formulas.
- If you need to copy a schematic or figure from the lab manual to your report, use Copy and Paste function or take a screenshot by using Snipping Tool in MS-Windows.
- Do not place screenshots of your lab notebook in the report! Diagrams, tables, calculations, etc. must be generated using the existing tools in the word processor.

2.3 Order of Lab Report Components

Coverpage

Cover page must include lab name and number, your name and the date the lab was performed.

Objective

Clearly state the experiment objective in your own words.

Software Needed

Indicate which software was used in performing the experiment.

For Each Part of the Lab

- Write the lab's part number and title in bold font. Firstly, describe the problem that you studied in this part, give an introduction of the theory, and explain why you did this experiment. Do not lift the text from the lab manual; use your own words.
- Secondly, describe the experimental setup and procedures. Do not follow the lab manual in listing out individual pieces of equipment and assembly instructions. That is not relevant information in a lab report! Your description should take the form of a narrative, and include information not present in the manual, such as descriptions of what happened during intermediate steps of the experiment.

- Thirdly, explain your findings. This is the most important part of your report, because here, you show that you understand the experiment beyond the simple level of completing it. Explain (compare expected results with those obtained). Analyse (analyze experimental error). Interpret (explain your results in terms of theoretical issues and relate to your experimental objectives). All the results should be presented even if there is any inconsistency with the theory. It should be possible to understand what is going on by just reading through the text paragraphs, without looking at the figures.

- Finally, provide a summary of what was learned from this part of the laboratory experiment. If the results seem unexpected or unreliable, discuss them and give possible explanations.

Conclusion

The conclusion section should provide a take-home message summing up what has been learned from the experiment:

- Briefly restate the purpose of the experiment (the question it was seeking to answer)
- Identify the main findings (answer to the research question)
- Note the main limitations that are relevant to the interpretation of the results
- Summarize what the experiment has contributed to your understanding of the problem.

Probing Further Experiments

Questions pertaining to this lab must be answered at the end of laboratory report.

Lab-Orientation

Introduction

In the first lab period, the students should become familiar with the location of computer and components in the lab, the course requirements such as software installation procedure, and the teaching Instructor.

Objective

1. To familiarize the students with the lab facilities, equipment, standard operating procedures, lab safety, and the course requirements.
2. Discover the procedure to write, test, and debug lab assignments in the lab.
3. Learn how to submit lab assignments for evaluation.

PreLab

Download and install the “VM Ware” software on personal computer, available in the lab.

Software Needed:

HADOOP software.

Apache Hadoop software is an open source framework that allows for the distributed storage and processing of large datasets across clusters of computers using simple programming models.

In this way, Hadoop can efficiently store and process large datasets ranging in size from gigabytes to petabytes of data.

Hadoop is an open source software framework that supports distributed storage and processing of huge amount of data set. It is most powerful big data tool in the market because of its features.

Features like Fault tolerance, Reliability, High Availability etc. Hadoop provides- HDFS – World most reliable storage layer.

Continuous Evaluation

Lab-1 INSTALL VMWARE(Getting Started)

Installation of VMWare to setup the Hadoop environment and its ecosystems.

1.Introduction

VMware, founded in 1998, is a pioneering company that has played a pivotal role in the development and popularization of virtualization technology. Virtualization is the process of creating a virtual representation of computer hardware, storage, and networking resources to enable the efficient utilization of physical hardware and enhance the flexibility and manageability of IT infrastructures. VMware Workstation Pro is the industry standard for running multiple operating systems as virtual machines (VMs) on a single Linux or Windows PC to build, test, or demo software.

At its core, VMware offers a suite of virtualization solutions that allow businesses to run multiple virtual machines (VMs) on a single physical server. Each virtual machine operates as an independent, isolated system with its own operating system and applications, making it possible to consolidate multiple workloads on a single piece of hardware. This approach has transformed the way data centers are designed and managed, leading to significant cost savings, improved resource utilization, and enhanced disaster recovery capabilities.

One of VMware's most notable products is VMware vSphere, which encompasses a range of tools and services for building and managing virtualized environments. Some key components of vSphere include:

- 1.ESXi: A hypervisor that runs directly on the physical server hardware, enabling the creation and management of virtual machines.
- 2.vCenter Server: A centralized management platform that provides tools for configuring, monitoring, and automating virtualized environments.
- 3.vMotion: A technology that enables live migration of virtual machines from one physical host to another without disruption, ensuring high availability and load balancing.
- 4.DRS (Distributed Resource Scheduler): A feature that automatically balances workloads across a cluster of hosts to optimize resource utilization.
- 5.HA (High Availability): Ensures that virtual machines remain available in the event of host failures by restarting them on other hosts.
- 6.NSX: A software-defined networking solution that virtualizes network services and security, allowing for more agile and flexible network management.
- 7.vSAN (Virtual SAN): A software-defined storage solution that pools together direct-attached storage across multiple hosts to create a shared storage resource for virtual machines.

2.Objective

By the end of this lab, the student should learn how to Install VMWare.

3.PreLab Preparation :

Resource:

Supplementary Resources:
Guest Operating System Installation Guide.
VMware Compatibility Guide.
OVF (Open Virtualization Format) Resources.
Using vmrun to Control Virtual Machines.
VMCI Sockets Programming Guide.
VIX API Resources

Equipment need:

VMware Workstation 16 Pro

Back ground:

Processor - 2.0GHz or higher Intel or AMD x86 processor. Processor requirements can be larger if your database is run on the same hardware. Memory - 2GB RAM minimum. RAM requirements can be larger if your database is run on the same hardware.

4.Pre lab Questions:

1. Define is VMWare stack?
2. List out various data formats?
3. List out the characteristics of big data?
4. Define the importance of virtualization?
5. Outline the different types of virtualization available?

5.Post Lab Preparation

Preparing for a VMware installation involves several important steps to ensure a smooth and successful deployment of virtualization technology. Here's a comprehensive post-lab preparation guide:

1. Hardware Compatibility Check:

Before you start, verify that your hardware meets VMware's compatibility requirements. Use the VMware Compatibility Guide to ensure that your server, storage, and network components are supported by the version of VMware you intend to install.

2. Software Licensing:

Make sure you have the necessary licenses for your VMware products. This might include licenses for vSphere, vCenter Server, and any other VMware tools you plan to use.

3. Network Planning:

Plan your network configuration, including IP addresses, VLANs, and network topology. Ensure that the physical network infrastructure is properly configured to support virtualization requirements.

4. Storage Planning:

Determine how you will configure storage for your virtual machines. Consider using SAN (Storage Area Network), NAS (Network Attached Storage), or local storage options. Plan for storage capacity, performance, and redundancy.

5. Server Hardware Configuration:

Configure the hardware settings on your servers, such as BIOS/UEFI settings, RAID configurations, and hardware monitoring. Ensure that the hardware is set up to support virtualization features like VT-x (Intel) or AMD-V (AMD) and that the appropriate settings are enabled.

6. Installation Media:

Obtain the installation media for the VMware product you're installing. This might be an ISO file that you can burn to a DVD or mount directly to your virtualization server.

7. Data Backup:

Before making any changes to your infrastructure, back up critical data and configurations. This ensures that you can revert to a known state if anything goes wrong during the installation process.

8. Documentation:

Gather any documentation and installation guides provided by VMware. Having these resources on hand can be very helpful during the installation process.

9. Installation Steps:

Follow these general steps for installing VMware:

- a. Boot your server from the installation media.
- b. Follow the prompts to install the VMware hypervisor (e.g., ESXi).
- c. Configure network settings, IP addresses, and hostnames.
- d. Set up storage configurations.
- e. Create a strong password for the root account.
- f. Review and confirm the installation settings.
- g. Complete the installation and reboot the server.

10. Post-Installation Tasks:

After the installation is complete, there are a few more tasks to perform:

sql

Copy code

- a. Configure networking settings for management access.
- b. Apply any updates or patches to the VMware software.
- c. Install VMware Tools on guest operating systems for improved performance and functionality.
- d. Set up security features like firewalls and intrusion detection.
- e. Create virtual networks, storage, and start creating virtual machines.
- f. Configure backup and monitoring solutions.

11. Testing and Validation:

Before deploying critical workloads, conduct testing to ensure that virtualization is working as expected. Test VM creation, migration, networking, and performance.

12. Training and Documentation:

Provide training to your team on how to manage and operate the VMware environment. Also, document the configurations, settings, and procedures for future reference.

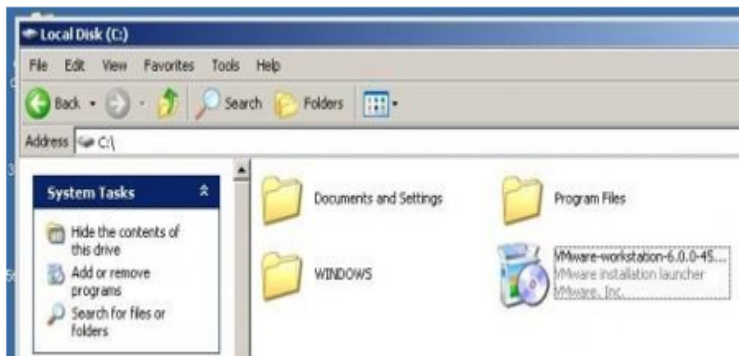
By following these steps, you'll be well-prepared to successfully install and set up your VMware virtualization environment. Remember to tailor the process to your specific infrastructure and business requirements.

The Following of the Steps are considered for implementation of VMWare Installation. **Procedure**

6.Procedure

Step 1: STEP 1. First of all, enter to the official site of VMware and download VMware Workstation <https://www.vmware.com/tryvmware/?p=workstation-w>

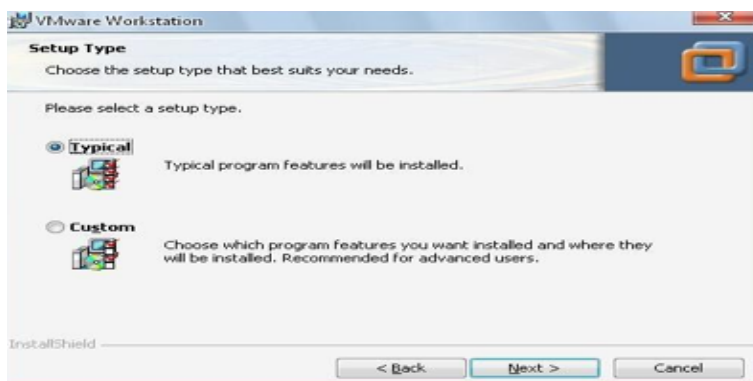
Step2: STEP 2. After downloading VMware workstation, install it on your PCSTEP 2. After downloading VMware workstation, install it on your PC



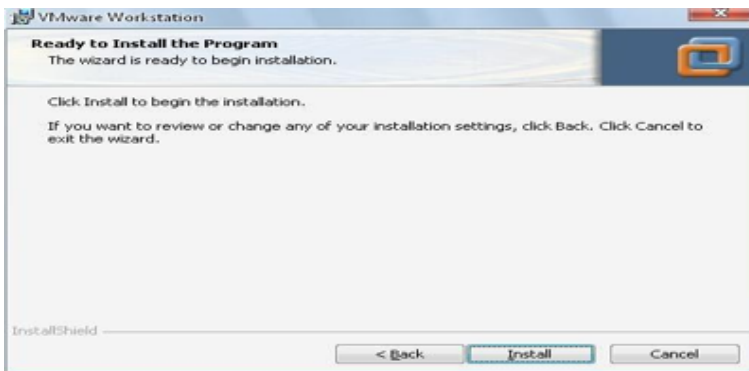
Step3:Setup will open Welcome Screen.



Click on Next button and choose Typical option



Step 4: By clicking “Next” buttons, to begin the installation, click on Install button at the end.

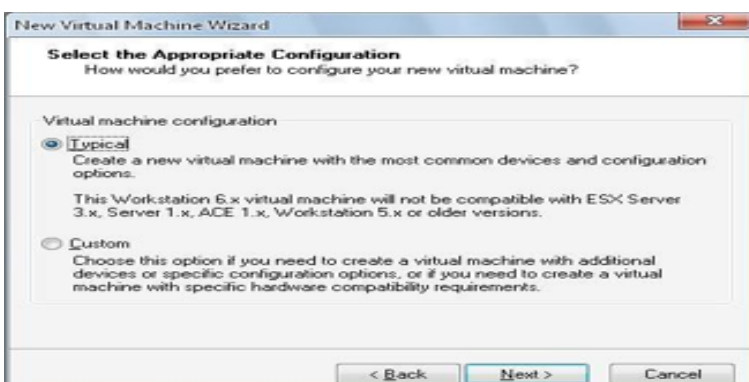


Step 5: This will install VMware Workstation software on your PC, After installation complete, click on Finish button. Then restart your PC. Then open this software.



Step 6: In this step we try to create new “virtual machine”. Enter to File menu, then New-> Virtual Machine

Click on Next button, then check Typical option as below

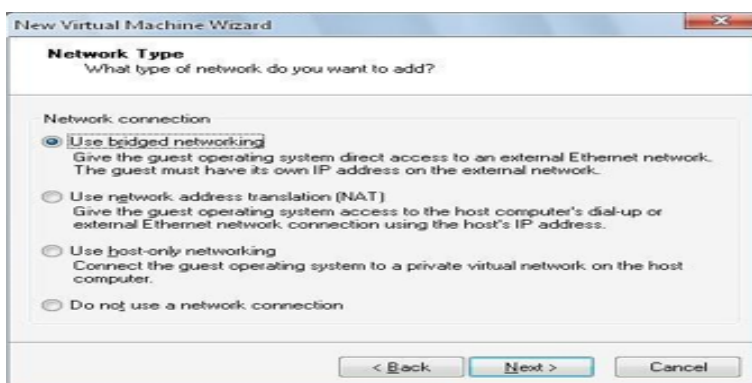
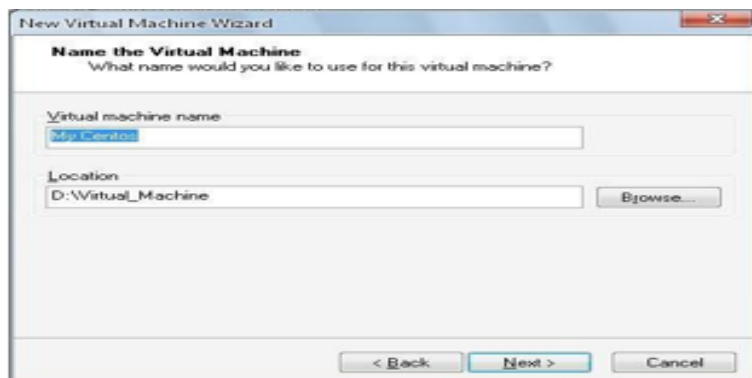
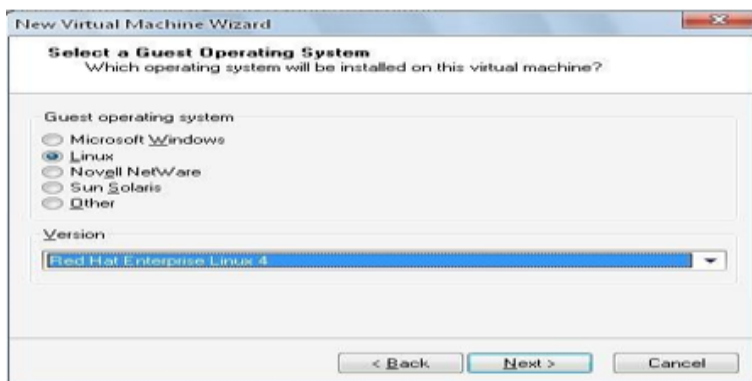


Then click Next button, and check your OS version. In this example, as we’re going to setup Oracle server on CentOS, we’ll check Linux option and from “version” option we’ll check Red Hat

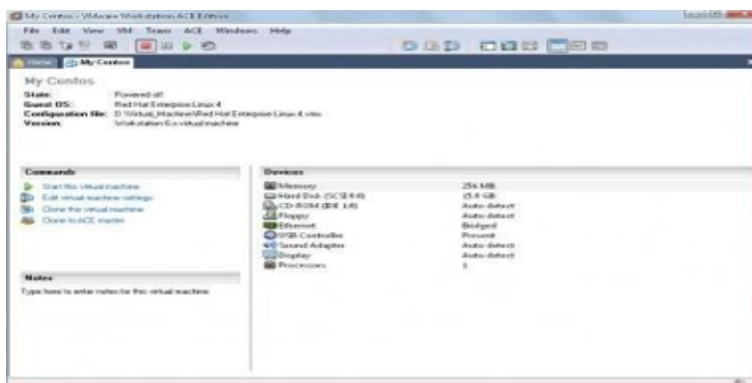
Enterprise Linux 4

By clicking Next button, we’ll give a name to our virtual machine, and give directory to create this new virtual machine

Then select Use bridged networking option and click Next

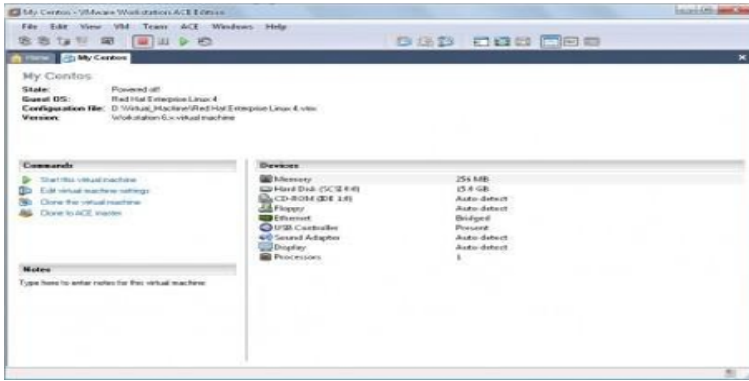


Then you've to define size of hard disk by entering its size. I'll give 15 GB hard disk space and please check Allocate all disk space now option



Here, you can delete Sound Adapter, Floppy and USB Controller by entering “Edit virtual machine settings”. If you're going to setup Oracle Server, please make sure you've increased your Memory (RAM) to 1GB.

7.Results and Analysis



8.POST LAB VIVAVOCE QUESTIONS:

1. List out various terminologies in Big Data environments?
2. Define big data analytics?
3. Define the importance of virtualization?
4. Summarize VMKernel and its importance
5. Name some of the VMware products.

9.Further Probing Experiments

Certainly, if you're looking to conduct further probing experiments after the initial VMware installation, here are some advanced experiments and tasks you can undertake to explore and optimize your virtualization environment:

1. Hypervisor Performance Testing:

Use benchmarking tools to measure the performance of the hypervisor (ESXi) under different workloads. Monitor metrics like CPU usage, memory usage, disk I/O, and network throughput. Compare these results to baseline performance to ensure optimal operation.

2. Virtual Machine Performance Tuning:

Create various virtual machines with different configurations (CPU, RAM, disk, etc.). Run performance tests on these VMs to analyze how different resource allocations impact their performance. This can help you fine-tune resource settings for optimal performance and resource utilization.

3. High Availability Testing:

Experiment with VMware's High Availability (HA) feature by intentionally causing host failures. Observe how VMs are automatically restarted on other hosts and the impact on application availability.

4. Distributed Resource Scheduler (DRS) Testing:

Implement DRS to dynamically balance workloads across hosts in a cluster. Generate varying workloads and observe how DRS responds to distribute resources efficiently.

5. Storage Performance Testing:

Test storage performance using tools like Iometer or Diskspd. Measure read/write speeds and latency to identify potential bottlenecks in your storage infrastructure.

6. Network Performance and Security Testing:

Assess network performance by transferring large files between VMs. Implement and test network security policies using features like distributed firewalls and network segmentation.

Lab-2 HADOOP MODES

1.Introduction to Hadoop:

Hadoop is an open-source framework which is mainly used for storage purpose and maintaining and analyzing a large amount of data or datasets on the clusters of commodity hardware, which means it is actually a data management tool. Hadoop also possesses a scale-out storage property, which means that we can scale up or scale down the number of nodes as per a requirement in the future which is really a cool feature.

Hadoop is a powerful and widely used open-source framework designed for storing, processing, and analyzing large volumes of data in a distributed and scalable manner. It was created by Doug Cutting and Mike Cafarella and is named after a toy elephant. Hadoop fundamentally changed the way organizations handle Big Data, making it feasible to process and gain insights from massive datasets that were previously too complex or time-consuming to manage.

In the modern digital landscape, data is being generated at an unprecedented rate. This data holds immense potential for organizations seeking valuable insights to make informed decisions, drive innovations, and improve various aspects of their operations. However, traditional relational databases and data processing tools often struggle to handle the sheer volume, variety, and velocity of this data. This is where Hadoop comes into play.

Key Concepts:

Hadoop is built on a few key concepts that enable it to efficiently manage and process large datasets:

Distributed Storage: Hadoop's HDFS (Hadoop Distributed File System) breaks down large files into smaller blocks and stores these blocks across multiple machines in a cluster. This enables parallel processing and fault tolerance, as data is replicated across different nodes.

MapReduce: MapReduce is a programming model and processing engine used to perform distributed data processing tasks. It divides a task into two main phases: the Map phase, where data is processed and transformed, and the Reduce phase, where the processed data is aggregated and summarized.

Scalability: Hadoop's architecture allows organizations to add more nodes to the cluster as data volume grows, ensuring seamless scalability. This "scale-out" approach contrasts with traditional databases that often require "scale-up" by upgrading hardware.

Fault Tolerance: Hadoop is designed to handle node failures gracefully. Data is replicated across multiple nodes, so if one node fails, data can still be accessed and processed from other nodes.

Ecosystem: Hadoop is not just a single tool but an ecosystem that includes various components like Hive (data warehousing), Pig (data scripting), HBase (NoSQL database), Spark (in-memory processing), and more. This ecosystem addresses different data processing needs.

Use Cases:

Hadoop has found applications in various industries and domains:

Business Analytics: Analyzing customer behavior, market trends, and business performance.

Scientific Research: Analyzing large scientific datasets, such as genetics or climate data.

Internet of Things (IoT): Processing and analyzing data from connected devices.

Fraud Detection: Identifying fraudulent transactions through pattern analysis.

Recommendation Systems: Providing personalized recommendations to users.

Log Analysis: Analyzing system logs for troubleshooting and optimization.

Challenges:

While Hadoop is incredibly powerful, it's not without challenges:

Complexity: Hadoop's ecosystem is vast and can be complex to set up and maintain. **Skill Gap:** Organizations need skilled professionals who understand Hadoop's architecture and tools. **Real-Time Processing:** Traditional Hadoop MapReduce is optimized for batch processing, which might not suit all real-time requirements.

Conclusion:

Hadoop has reshaped the way organizations handle and analyze data, allowing them to turn massive datasets into valuable insights. As the Big Data landscape continues to evolve, Hadoop remains a crucial tool in managing the challenges and opportunities presented by large-scale data processing and analysis.

2.Objective

1. Perform setting up and Installing Hadoop in its three operating modes.

i. Standalone.

ii.Pseudo distributed

iii.Fully distributed.

2. Use web based tools to monitor your Hadoop setup.

3.Pre lab Preparation

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Hadoop is a popular open-source framework used for distributed storage and processing of large datasets. It supports three main operating modes: Local (Standalone) Mode, Pseudo-Distributed Mode, and Fully-Distributed Mode. Here are resources for installing Hadoop in each of these modes:

1.Local (Standalone) Mode:

In this mode, Hadoop runs on a single machine without utilizing the distributed capabilities of the framework. It's useful for development and testing.

i.Hadoop Official Documentation: Standalone Operation

ii.Hadoop Setup in Standalone Mode Tutorial: Hadoop Standalone Mode Tutorial

2.Pseudo-Distributed Mode:

This mode simulates a fully-distributed environment on a single machine. Each Hadoop daemon runs in a separate Java process.

i.Hadoop Official Documentation: Pseudo-Distributed Operation

ii.Hadoop Pseudo-Distributed Setup Tutorial: Hadoop Pseudo-Distributed Mode Setup

3.Fully-Distributed Mode:

This is the typical production mode where Hadoop runs on a cluster of machines. It's used to process and store large datasets across multiple nodes.

i.Hadoop Official Documentation: Cluster Setup

ii.Cloudera's Guide for Setting Up a Hadoop Cluster: Cloudera Installation Guide

Remember that Hadoop's ecosystem and installation process might change over time, so it's a good practice to consult the official documentation and resources from reliable sources that are up-to-date with the latest version of Hadoop.

Additionally, when installing Hadoop, you'll likely need to configure various XML files, environment variables, and network settings. These configurations can be complex, so following step-by-step tutorials and guides will be very helpful.

Equipment Needed:

The requirements for installing Hadoop can vary depending on the mode (local, pseudo-distributed, fully-distributed) you intend to use and the scale of your deployment. Here are the general requirements for installing Hadoop:

1.Hardware Requirements:

i.Processor:

Hadoop benefits from multiple cores, so a multicore processor is recommended.

ii.Memory (RAM):

Hadoop requires a substantial amount of memory, especially for map and reduce tasks. A minimum of 4GB is recommended, but 8GB or more is better.

iii.Disk Space:

Hadoop needs disk space for storing data and intermediate results. The more space available, the better, as large datasets and processing can require significant storage.

2.Operating System:

Hadoop can run on various Unix-based systems, including Linux and macOS. It's recommended to use a Linux distribution, such as Ubuntu, CentOS, or Debian, for production deployments.

3.Java:

Hadoop is a Java-based framework, so you need Java installed on your system. Make sure you have Java Development Kit (JDK) 8 or later. It's recommended to use Oracle JDK or OpenJDK.

4.Network Configuration:

Ensure that network settings, including hostname resolution and SSH connectivity, are properly configured. Hadoop relies on SSH for communication between nodes.

5.Dependencies and Libraries:

Hadoop relies on various libraries like HDFS (Hadoop Distributed File System), YARN (Yet Another Resource Negotiator), and MapReduce. These components have their own dependencies that need to be installed.

6.SSH Key Pair:

Hadoop requires SSH connectivity between nodes for secure communication. Set up SSH keys to enable password-less SSH access between nodes.

7.Configuration:

Hadoop has multiple configuration files for different components. Properly configuring these files according to your installation mode and hardware specifications is crucial.

8.Network and Firewall Configuration:

Ensure that ports required for Hadoop services are open and not blocked by firewalls.

9.User and Group Setup:

Create dedicated users and groups for Hadoop services to ensure proper access control and security.

10.Environmental Variables:

Configure environment variables like JAVA_HOME and HADOOP_HOME to point to the correct directories.

11.Security Considerations:

In production environments, security becomes a critical concern. Hadoop supports integration with Kerberos for authentication and authorization.

12.Cluster Management Tools (For Fully-Distributed Mode):

In a fully-distributed setup, you might consider using cluster management tools like Apache Ambari or Cloudera Manager to simplify installation, configuration, and monitoring.

It's important to refer to the official documentation of the specific Hadoop distribution or version you're working with for detailed and up-to-date requirements. Additionally, online tutorials and guides can help walk you through the installation process based on your specific environment and use case.

Big data and business analytics laboratory (AITB16) lab manual and work sheets,VMWARE Workstation,Hadoop,jdk

Background

Certainly, let's delve into the background required for Hadoop installation.

Distributed Computing and Big Data:

Before discussing Hadoop, it's important to understand the context of distributed computing and big data:

1.Distributed Computing: Traditional computing models often rely on a single powerful machine. However, as data volumes grew, the need arose for distributed computing, where tasks are divided among multiple machines for parallel processing. This approach provides scalability, fault tolerance, and faster data processing.

2.Big Data: With the rise of the internet, social media, IoT, and other sources, data production exploded. Big data refers to datasets that are so large and complex that traditional data processing applications struggle to handle them efficiently.

Hadoop's Origins and Purpose:

Hadoop emerged as a solution to address the challenges of distributed computing and big data processing:

1.Inspiration from Google: Hadoop's design was heavily influenced by Google's MapReduce and Google File System (GFS) papers. These papers introduced the concepts of parallel processing and distributed storage that Hadoop adopted.

2.Distributed File System: Hadoop Distributed File System (HDFS) enables data to be stored across clusters of machines, replicating it for fault tolerance. It's optimized for handling large files, making it suitable for big data storage.

3.MapReduce: Hadoop introduced a programming model called MapReduce. It breaks down data processing tasks into map and reduce stages, which can be distributed across nodes. This model simplifies parallel processing of data-intensive tasks.

Key Components of Hadoop:

1.HDFS (Hadoop Distributed File System):

HDFS is designed to store vast amounts of data across multiple machines. It's built to be fault-tolerant and highly available, using data replication.

2.MapReduce:

MapReduce is a parallel programming model for processing and generating large datasets that can be parallelized across a Hadoop cluster. It abstracts the complexity of distributed computing.

3.YARN (Yet Another Resource Negotiator):

YARN was introduced to separate resource management and job scheduling from the MapReduce programming model. It allows multiple applications to share cluster resources effectively.

Installation Modes and Requirements:

Hadoop offers various installation modes to cater to different scenarios:

Local (Standalone) Mode: Hadoop runs on a single machine without exploiting distributed capabilities. Useful for local development and testing.

Pseudo-Distributed Mode: Simulates a multi-node cluster on a single machine. Each Hadoop daemon runs as a separate process.

Fully-Distributed Mode: Utilizes a cluster of machines to harness the full power of distributed computing. Used for processing large datasets in production environments.

Installation Requirements:

Hadoop installation demands considerations such as hardware resources (CPU, RAM, disk space), compatible Java version, proper configuration of XML files, network setup for SSH communication, and environment variables.

In summary, Hadoop installation is driven by the need to manage and process big data efficiently through distributed computing. It's essential to understand the motivations behind Hadoop's development, its key components, and the various installation modes and requirements to ensure a successful setup for handling large-scale data processing tasks.

4.Pre lab questions

Purpose and Goals:

- 1.What is the primary goal of using Hadoop in your project? (e.g., data processing, analysis, storage)
- 2.What specific problems or challenges are you trying to address with Hadoop? **Data Considerations:**
- 3.What type of data will you be working with? (e.g., structured, semi-structured, unstructured)
- 4.How large is the dataset you intend to process or analyze with Hadoop? Do you have a plan for ingesting and storing the data in Hadoop's HDFS? **Hardware and Infrastructure:**
- 5.Do you have a dedicated cluster of machines for running Hadoop, or will you use cloud services?
- 6.Are the cluster nodes properly configured with necessary hardware resources (RAM, CPU, disk space)? **Software and Tools:**
- 7.Have you selected a distribution of Hadoop (e.g., Apache Hadoop, Cloudera, Hortonworks)?
- 8.Are all the required software components (HDFS, MapReduce, etc.) properly installed?

5.Post Lab Preparation

Preparing for post-lab activities after setting up and installing Hadoop in its three operating modes (Standalone, Pseudo-Distributed, and Fully Distributed) involves verifying your installation, understanding the differences between these modes, and documenting your setup. Here are the steps you can follow:

Verification of Hadoop Installation:

Verify that Hadoop is installed correctly by running basic Hadoop commands and utilities. For example: `hadoop version` to check the Hadoop version. `hdfs dfs -ls /` to list the root directory in HDFS. `yarn node -list` to check the status of YARN nodes.

Understanding the Three Operating Modes:

Write down the key differences between Standalone, Pseudo-Distributed, and Fully Distributed modes. This includes how they handle Hadoop's components like NameNode, DataNode, ResourceManager, and NodeManager. Understand the use cases for each mode. For instance, Standalone is primarily for development and testing, Pseudo-Distributed is for single-node setups for development, and Fully Distributed is for production clusters.

Documentation:

Create a document that describes your Hadoop setup in each of the three operating modes. Include the following details: Configuration settings: Document the key configuration parameters you set or modified in each mode. Ports: Note the ports used by different Hadoop components in each mode. Filesystem layout: Describe how data is stored on HDFS. Execution modes: Explain how MapReduce or Spark jobs are executed in each mode. Include any issues you encountered during installation and how you resolved them.

Testing and Experimentation:

Run sample Hadoop jobs or Spark applications in each operating mode to ensure that they function as expected. Experiment with configurations to observe how changes impact performance and behavior in different modes.

Performance Evaluation:

If your lab or project involves performance evaluation, conduct benchmarking tests in each operating mode. Measure factors like execution time, resource utilization, and scalability.

Backup and Cleanup:

If you are working with real data, ensure that you back up your data if needed. Clean up any temporary files, logs, or unnecessary data generated during the lab to maintain a clean environment.

Reporting and Presentation:

If required, prepare a report or presentation summarizing your Hadoop setup and installation experiences. Include your documentation, key findings, and any insights into the differences between operating modes.

Reflection:

Reflect on what you learned from setting up Hadoop in these three operating modes. Consider the advantages and limitations of each mode and how they might be applicable in different scenarios.

Future Work:

Identify any additional steps or tasks that need to be performed based on the objectives of your lab or project. This could include further experimentation, optimization, or integration with other technologies.

The Following of the steps are considered for implementation of Hadoop Modes

6.PROCEDURE:

a) STANDALONE MODE:

Installation of jdk 7

Command: `sudo apt-get install openjdk-7-jdk`

Download and extract Hadoop

Command: `wget http://archive.apache.org/dist/hadoop/core/hadoop-1.2.0/hadoop-1.2.0.tar.gz`

Command: `tar-xvf hadoop-1.2.0.tar.gz`

Command: `sudo mv hadoop-1.2.0 /usr/lib/hadoop`

Set the path for java and hadoop

Command: sudo gedit \$HOME/.bashrc

```
export JAVA-HOME=/usr/lib/jvm/java-7 openjdk-i386 export PATH=$PATH:$JAVA-HOME/bin
export HADOOP-COMMON-HOME=/usr/lib/hadoop export HADOOP-MAPRED-HOME=/usr/lib/hadoop
export PATH=$PATH: $HADOOP-COMMON-HOME/bin
export PATH=$PATH:$HADOOP-COMMON-HOME/Sbin
```

Checking of java and hadoop

Command: java version

Command: hadoop version

B)PSEUDO MODE:

Hadoop single node cluster runs on single machine. The namenodes and datanodes are performing on the one machine. The installation and configuration steps as given below:

Installation of secured shell

Command: sudo apt-get install openssh-server

Create a ssh key for passwordless ssh configuration

Command: ssh-keygen -t rsa -P ""

Moving the key to authorized key

Command: cat *HOME/.ssh/id_rsa.pub* >>*HOME/.ssh/authorized-keys*
/*****RESTART THE COMPUTER*****/

Checking of secured shell login

Command: ssh localhost

Add JAVA-HOME directory in hadoop-env.sh file

Command: sudo gedit /usr/lib/hadoop/conf/hadoop-env.sh export JAVA-HOME=/usr/lib/jvm/java-7-openjdk-i386 Creating namenode and datanode directories for hadoop

Command: sudo mkdir -p /usr/lib/hadoop/dfs/namenode

Command: sudo mkdir -p /usr/lib/hadoop/dfs/datanode

Configure core-site.xml

Command: sudo gedit /usr/lib/hadoop/conf/core-site.xml

```
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:8020</value>
</property>
```

Configure hdfs-site.xml

Command: sudo gedit /usr/lib/hadoop/conf/hdfs-site.xml

```
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permissions</name>
<value>>false</value>
</property>
<property> <name>dfs.name.dir</name>
<value>/usr/lib/hadoop/dfs/namenode</value>
</property>
<property>
<name>dfs.data.dir</name>
```

```
<value>/usr/lib/hadoop/dfs/datanode</value>
</property>
```

Configure mapred-site.xml

Command: sudo gedit /usr/lib/hadoop/conf/mapred-site.xml

```
<property>
<name>mapred.job.tracker</name>
```

```
<value>localhost:8021</value>
</property>
```

Format the name node

Command:hadoop namenode -format

Start the namenode, datanode

Command:start-dfs.sh

Start the task tracker and job tracker

Command:start-mapred.sh

To check if Hadoop started correctly

Command: jps namenode secondarynamenode datanode

jobtracker tasktracker

C)FULLY DISTRIBUTED MODE:

All the demons like namenodes and datanodes are runs on different machines. The data will replicate according to the replication factor in client machines. The secondary namenode will store the mirror images of namenode periodically. The namenode having the metadata where the blocks are stored and number of replicas in the client machines. The slaves and master communicate each other periodically. The configurations of multinode cluster are given below:

Configure the hosts in all nodes/machines

Command:sudo gedit /etc/hosts/ 192.168.1.58 pcetcse1

pcetcse2

pcetcse3

pcetcse4

pcetcse5

Passwordless Ssh Configuration Create ssh key on namenode/master.

Command: ssh-keygen -t rsa -p ""

Copy the generated public key all datanodes/slaves

Command: ssh-copy-id -i /.ssh/id-rsa.pub huser@pcetcse2

Command: ssh-copy-id -i /.ssh/id-rsa.pub huser@pcetcse3

Command: ssh-copy-id -i /.ssh/id-rsa.pub huser@pcetcse4

Command: ssh-copy-id -i /.ssh/id-rsa.pub huser@pcetcse5

/*****RESTART ALL NODES/COMPUTERS/MACHINES *****/

NOTE: Verify the passwordless ssh environment from namenode to all datanodes as "huser" user.

Login to master node

Command: ssh pcetcse1

Command: ssh pcetcse2

Command:ssh pcetcse3

Command:ssh pcetcse4

Command:ssh pcetcse5

Add JAVA-HOME directory in hadoop-env.sh file in all nodes/machines

Command: sudo gedit /usr/lib/hadoop/conf/hadoop-env.sh export JAVA-HOME=/usr/lib/jvm/java-7-openjdk-i386

Creating namenode directory in namenode/master

Command: sudo mkdir -p /usr/lib/hadoop/dfs/namenode

Creating namenode directory in datanodes/slaves

Command: sudo mkdir -p /usr/lib/hadoop/dfs/datanode Close HTML tag.

Use web based tools to monitor your Hadoop setup.

HDFS Namenode on UI <http://localhost:50070/>

6.RESULTS AND ANALYSIS

ubuntu @localhost> jps

Data node, name node

Secondary name node, NodeManager, Resource Manager



NameNode 'localhost:8020'

Started: Fri May 08 12:09:25 IST 2015
Version: 1.2.0, r1479473
Compiled: Mon May 6 06:59:37 UTC 2013 by hortonfo
Upgrades: There are no upgrades in progress.

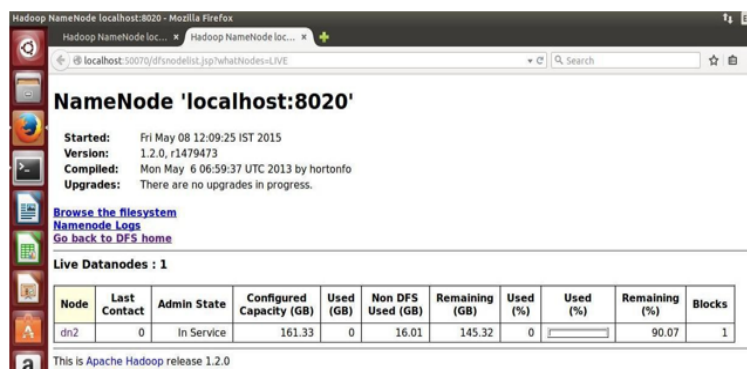
[Browse the filesystem](#)
[NameNode Logs](#)

Cluster Summary
6 files and directories, 1 blocks = 7 total. Heap Size is 60 MB / 889 MB (6%)

Configured Capacity	161.33 GB
DFS Used	28.01 KB
Non DFS Used	16.01 GB
DFS Remaining	145.32 GB
DFS Used%	0 %
DFS Remaining%	90.07 %
Live Nodes	1
Dead Nodes	0
Decommissioning Nodes	0
Number of Under-Replicated Blocks	0

NameNode Storage:

Storage Directory	Type	State
-------------------	------	-------



NameNode 'localhost:8020'

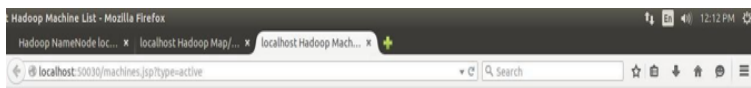
Started: Fri May 08 12:09:25 IST 2015
Version: 1.2.0, r1479473
Compiled: Mon May 6 06:59:37 UTC 2013 by hortonfo
Upgrades: There are no upgrades in progress.

[Browse the filesystem](#)
[NameNode Logs](#)
[Go back to DFS home](#)

Live Datanodes : 1

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
dn2	0	In Service	161.33	0	16.01	145.32	0		90.07	1

This is Apache Hadoop release 1.2.0



localhost Hadoop Machine List

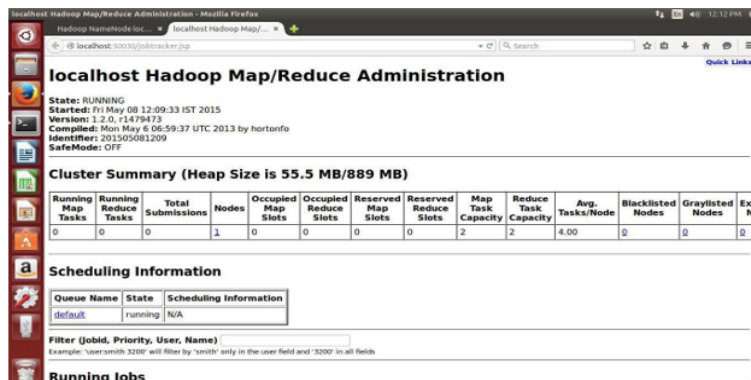
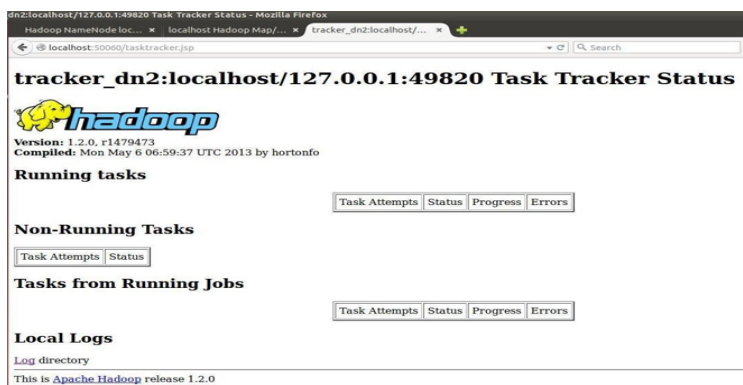
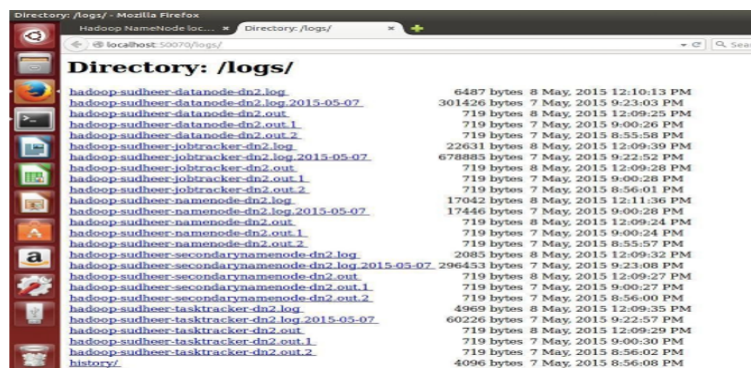
Active Task Trackers

Task Trackers												
Name	Host	# running tasks	Max Map Tasks	Max Reduce Tasks	Task Failures	Directory Failures	Node Health Status	Seconds Since Node Last Healthy	Total Tasks Since Start	Succeeded Tasks Since Start	Total Tasks Last Day	Succeeded Tasks Last Day
tracker_dn2:localhost/127.0.0.1:49820	dn2	0	2	2	0	0	N/A	0	0	0	0	0

This is Apache Hadoop release 1.2.0

HDFS Jobtracker <http://localhost:50030/>

HDFS Logs <http://localhost:50070/logs/>



7.VIVA VOCE QUESTIONS:

1. List the main components of a Hadoop Application?
2. Explain the difference between NameNode,
3. Define jps command
4. How to restart Namenode?
5. Differentiate between Structured and Unstructured data?

8.Further probing experiments

1.Operating System and Version:

Which operating system and version are you planning to install Hadoop on (e.g., Ubuntu 20.04, CentOS 8, etc.)?

2.Cluster Setup:

Is this a single-node or a multi-node cluster installation? If multi-node, how many nodes are you planning to have in your cluster?

3.Hardware Requirements:

What are the hardware specifications of the machines you'll be using (CPU cores, RAM, disk space)? Do you have a plan for managing network connectivity between nodes?

4.Java Installation:

Have you installed the required version of Java Development Kit (JDK)? Which version are you using?

5.Hadoop Distribution:

Are you planning to install a specific distribution of Hadoop, such as Apache Hadoop or a vendor-specific distribution like Cloudera or Hortonworks?

Lab-3 USING LINUX OPERATING SYSTEM

1.INTRODUCTION

Linux distribution is an operating system that is made up of a collection of software based on Linux kernel or you can say distribution contains the Linux kernel and supporting libraries and software.

Linux is a widely used open-source operating system kernel that serves as the foundation for various operating systems, often referred to as "Linux distributions" or simply "Linux distros." It was created by Linus Torvalds in 1991 and has since become a cornerstone of modern computing. Linux distributions are known for their stability, security, flexibility, and extensive community support.

Key Characteristics of Linux:

- 1.Open Source: Linux is open-source software, which means its source code is freely available to the public. This fosters collaboration, innovation, and customization.
- 2.Kernel: The Linux kernel is the core component of the operating system. It manages hardware resources, provides basic services to other software, and acts as an intermediary between applications and the hardware.
- 3.Variety of Distributions: Linux comes in various distributions, each tailored to specific use cases and preferences. Some well-known distributions include Ubuntu, Fedora, Debian, CentOS, and Arch Linux.
- 4.Package Management: Linux distributions use package managers to easily install, update, and remove software packages. These tools streamline software management and dependency handling.
- 5.Shell: Linux offers a powerful command-line interface called the shell. Users interact with the system by typing commands, enabling efficient system administration and automation.
- 6.Multi-User and Multi-Tasking: Linux is designed for multi-user environments, allowing multiple users to log in and use the system simultaneously. It also supports multitasking, enabling users to run several applications concurrently.
- 7.Security: Linux is known for its strong security features. Users and processes have distinct permissions, reducing the risk of unauthorized access and malicious activity.
- 8.Stability and Reliability: Linux systems are renowned for their stability and reliability. They can run for long periods without needing a reboot, which is beneficial for servers and critical systems.
- 9.Customization: Linux allows users to customize nearly every aspect of the system, from the desktop environment to system behaviors and configurations.
- 10.Server and Embedded Systems: Linux is widely used in server environments due to its stability, scalability, and performance. It's also a popular choice for embedded systems, powering devices like routers, smart TVs, and IoT devices.
- 11.Community Support: The Linux community is vast and vibrant. Online forums, documentation, and collaborative development contribute to robust support and rapid problem-solving.

2.Objective

Implementing the basic commands of LINUX Operating System – File/Directory creation, deletion, update operations.

- a. Create a directory in HDFS at given path(s).
- b. List the contents of a directory.
- c. Upload and download a file in HDFS.
- d. See contents of a file
- e. Copy a file from source to destination
- f. Copy a file from / To Local file system to HDFS
- g. Move file from source to destination.
- h. Remove a file or directory in HDFS.

3.Pre Lab Preparation

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Equipment Needed,Software Required

Big data and business analytics laboratory (AITB16) lab manual and work sheets,ubuntu

Background

In Linux, process running independently of the shell. One can leave the terminal window and, but process executes in the background without any interaction from users. For example, Apache or Nginx web server always runs in the background to serve you images and dynamic content.

4.Pre lab Questions

1. What is ls command?
2. What are the attributes of ls command?
3. What is an operating system?
- 4.What are basic elements or components of Linux?

5.Post Lab Preparation

The Following of the steps are considered for Implementing the basic commands of LINUX Operating System – File/Directory creation, deletion,update operations

Procedure:

To create a directory in HDFS (Hadoop Distributed File System), you can use the `hdfs dfs -mkdir` command. Here's the syntax:

A screenshot of a terminal window with a dark background. The top bar is dark gray with the text 'bash' on the left and a 'Copy code' button on the right. The main area is black with the command 'hdfs dfs -mkdir <path>' written in a light blue monospace font. The command is on a single line, and the prompt is not visible.


```
bash  
  
hdfs dfs -mkdir <path>
```

Replace `<path>` with the desired directory path within HDFS. Make sure you have the necessary permissions to create directories in the specified location.

For example, if you want to create a directory named "mydir" in the root directory of HDFS, you would use:


If you want to create a directory inside an existing directory, you can provide the full path:

bash

 Copy code

```
hdfs dfs -mkdir /mydir
```

bash

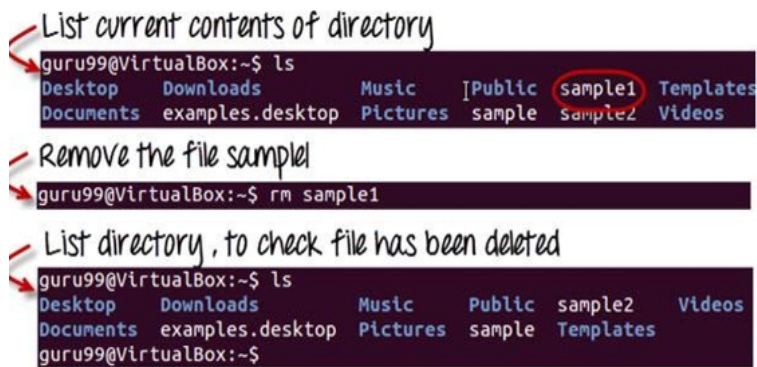
 Copy code

```
hdfs dfs -mkdir /parentdir/childir
```

Remember that HDFS follows a Unix-like file system structure, so you use forward slashes (/) to separate directories. Make sure you have the required permissions to create directories in the desired location within HDFS.

1. cat > filename
2. Add content
3. Press 'ctrl + d' to return to command prompt. To remove a file use syntax - rm filename

RESULTS AND ANALYSIS



The image contains three terminal screenshots with handwritten annotations. The first screenshot is annotated with 'List current contents of directory' and shows the output of 'ls' command, with 'sample1' circled in red. The second screenshot is annotated with 'Remove the file sample1' and shows the execution of 'rm sample1'. The third screenshot is annotated with 'List directory, to check file has been deleted' and shows the output of 'ls' command after deletion, where 'sample1' is no longer present.

```
guru99@VirtualBox:~$ ls
Desktop  Downloads  Music  [Public sample1 Templates
Documents examples.desktop Pictures sample sample2 Videos

guru99@VirtualBox:~$ rm sample1

guru99@VirtualBox:~$ ls
Desktop  Downloads  Music  Public sample2 Videos
Documents examples.desktop Pictures sample Templates
guru99@VirtualBox:~$
```

POST LAB VIVAVOCE QUESTIONS:

1. What is the purpose of rm command?
2. What is the difference between Linux and windows commands?
3. Enumerate File Register System?
4. Demonstrate Directory?

FUTHER PROBING EXPERIMENTS

1.File and Directory Manipulation:

How do you create a new directory in Linux using the mkdir command? Can you explain how to use the cp command to copy a file from one directory to another? What's the difference between the rm and rmdir commands when it comes to deleting directories?

2.File Viewing and Editing:

How do you display the first 10 lines of a text file using the head command? Can you use the grep command to search for a specific word in a file? What are some options you can use with it? Explain the process of editing a file using the nano or vi text editor.

3.Permissions and Ownership:

How would you change the permissions of a file to make it executable using the chmod command? Can you describe how to change the ownership of a directory and its contents using the chown command?

4.Process Management:

Using the ps command, how can you list all processes running in the system? If you have a process running in the foreground, how can you move it to the background using the appropriate command?

Lab-4 FILE MANAGEMENT IN HADOOP

Introduction

HDFS is the primary or major component of the Hadoop ecosystem which is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files. To use the HDFS commands, first you need to start the Hadoop services using the following command:

```
sbin/start-all.sh
```

Hadoop is a widely used framework for processing and storing large datasets in a distributed and scalable manner. It includes a distributed file storage system called Hadoop Distributed File System (HDFS) as its primary file management system. HDFS is designed to handle massive amounts of data across a cluster of computers and is a crucial component of the Hadoop ecosystem. Here's an overview of the Hadoop Distributed File System:

Hadoop Distributed File System (HDFS):

HDFS is designed to store and manage large datasets across a distributed cluster of commodity hardware. It is built to provide high throughput, fault tolerance, and reliability for big data applications. Some key features of HDFS are:

- 1.Distributed Storage: HDFS splits large files into smaller blocks (default size is 128 MB or 256 MB) and stores these blocks across multiple nodes in the cluster. This distribution of data improves performance and enables parallel processing.
- 2.Replication: HDFS ensures data durability and fault tolerance by replicating each block multiple times across different nodes in the cluster. The default replication factor is usually 3, meaning that each block is stored on three different nodes. If a node fails, the data is still available from other replicas.
- 3.Master-Slave Architecture: HDFS follows a master-slave architecture where there are two main components: NameNode and DataNode. The NameNode manages the file system metadata (directory structure, file permissions, etc.), while DataNodes store the actual data blocks.
- 4.Data Integrity: HDFS uses checksums to verify the integrity of data blocks. Data is checked for corruption both during storage and when transferred between nodes.
- 5.Streaming Data Access: HDFS is optimized for batch processing and large-scale data access patterns. It provides efficient sequential read and write operations rather than random access.

To check the Hadoop services are up and running use the following command:

```
jps
```

Objective:

Implement the following file management tasks in Hadoop:

- a. Copy a file from/ To Local file system to HDFS
- b. Move file from source to destination.
- c. Remove a file or directory in HDFS.
- d. Display the aggregate length of a file.

Pre Lab Preparation

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Equipment Needed,Software required

Big data and business analytics laboratory (AITB16) lab manual and work sheets,Apache Hadoop

Background

Hadoop file systems are used to manage the files by using different operations like adding the file,retrieving the file,deleting the file.

Pre lab Questions

- 1) Define Hadoop?
- 2) List out the various use cases of Hadoop?
- 3) What are the differences between regular FileSystem and HDFS?
- 4) What is Hadoop? Name the Main Components of a Hadoop Application.?

Post Lab Preparation

The Following of the steps are considered for implementation of Adding files and directories, Retrieving files, Deleting files

Procedure:

Adding Files and Directories to HDFS:

Before you can run Hadoop programs on data stored in HDFS, you'll need to put the data into HDFS first. Let's create a directory and put a file in it. HDFS has a default working directory of /user/\$USER, where\$USER is your login user name. This directory isn't automatically created for you, though, so let's create it with the mkdir command. For the purpose of illustration, we use chuck. You should substitute your user name in the example commands.

```
hadoop fs -mkdir /user/chuck
```

```
hadoop fs -put
```

```
hadoop fs -put example.txt /user/chuck
```

Retrieving Files from HDFS

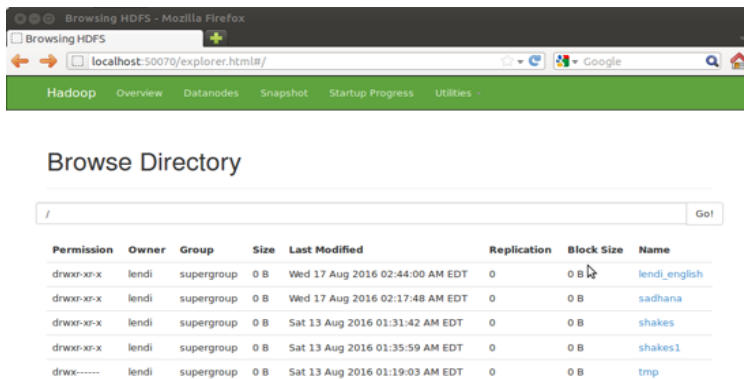
The Hadoop command get copies files from HDFS back to the local filesystem. To retrieve example.txt, we can run the following command:

```
hadoop fs -cat example.txt
```

```
hadoop fs -rm example.txt
```

- Command for creating a directory in hdfs is "hdfs dfs -mkdir /lendicse".
- Adding directory is done through the command "hdfs dfs -put lendi-english /".

INPUT/OUTPUT:



Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	lendl	supergroup	0 B	Wed 17 Aug 2016 02:44:00 AM EDT	0	0 B	lendl_english
drwxr-xr-x	lendl	supergroup	0 B	Wed 17 Aug 2016 02:17:48 AM EDT	0	0 B	sadhana
drwxr-xr-x	lendl	supergroup	0 B	Sat 13 Aug 2016 01:31:42 AM EDT	0	0 B	shakes
drwxr-xr-x	lendl	supergroup	0 B	Sat 13 Aug 2016 01:35:59 AM EDT	0	0 B	shakes1
drwx-----	lendl	supergroup	0 B	Sat 13 Aug 2016 01:19:03 AM EDT	0	0 B	tmp

VIVA VOCE QUESTIONS:

1. Distinguish the Hadoop Ecosystem?
2. Demonstrate divide and conquer philosophy in Hadoop Cluster?

Further Probing Questions

Certainly, here are some further probing questions related to file management in a Hadoop environment:

1.HDFS Basics:

What is HDFS, and how does it differ from a traditional file system? Describe the architecture of HDFS and its components (NameNode, DataNode, Secondary NameNode).

2.Uploading and Downloading Files:

How can you upload a file from your local machine to HDFS? What command or tool would you use? Explain the process of downloading a file from HDFS to your local machine.

3.Creating and Deleting Directories:

Using HDFS commands, how do you create a new directory within HDFS? Can you provide the command to delete an empty directory in HDFS?

4.Listing and Viewing Files:

How would you list the contents of a directory in HDFS using the appropriate command? Describe how to view the contents of a text file stored in HDFS without downloading it.

Lab-5 MAPREDUCE PROGRAM 1

Introduction

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.

The MapReduce paradigm is a programming model and processing framework designed for distributed and parallel processing of large volumes of data across clusters of computers. It was developed by Google to tackle the challenge of efficiently processing massive datasets using a simple and scalable approach. The MapReduce paradigm has had a profound impact on the field of big data processing and forms the foundation for many distributed data processing frameworks, including Hadoop.

Key characteristics and concepts of the MapReduce paradigm include:

- 1.Parallel Processing: MapReduce allows data processing to be distributed across multiple machines, enabling the efficient processing of vast amounts of data in parallel. This parallelism leads to significant performance gains.
- 2.Fault Tolerance: MapReduce is designed to handle hardware failures gracefully. If a machine fails during processing, the framework automatically redistributes the work to other available nodes, ensuring data integrity and job completion.
- 3.Divide and Conquer: The paradigm follows a "divide and conquer" approach, dividing a complex problem into smaller, independent tasks that can be processed in parallel. These tasks are divided into two phases: the Map phase and the Reduce phase.
- 4.Map Phase: During the Map phase, input data is divided into smaller chunks, and each chunk is processed independently by a mapper function. Mappers extract key-value pairs from the input data and emit intermediate key-value pairs.
- 5.Shuffling and Sorting: After the Map phase, intermediate key-value pairs are grouped by key and sorted. This process, known as shuffling and sorting, ensures that all values for a particular key are processed together during the Reduce phase.
- 6.Reduce Phase: In the Reduce phase, a reducer function processes the grouped and sorted key-value pairs. Reducers typically perform aggregation, summarization, or further computation on the data and emit the final output.
- 7.Data Locality: MapReduce takes advantage of data locality by processing data on the same nodes where it resides, minimizing data transfer over the network and improving performance.
- 8.Scalability: MapReduce systems can scale horizontally by adding more nodes to a cluster, allowing them to handle increasingly larger datasets and processing tasks.
- 9.Programming Model: MapReduce provides a straightforward programming model where developers write custom mapper and reducer functions to define the processing logic for their specific use cases.
- 10.Ecosystem Integration: The MapReduce paradigm is often integrated with other big data technologies and ecosystems, such as Hadoop, Apache Spark, and Apache Flink, to provide a comprehensive platform for data storage, processing, and analysis.

MapReduce is particularly well-suited for batch processing tasks, such as log analysis, data transformation, and ETL (Extract, Transform, Load) processes. While it remains a valuable approach for certain workloads, newer technologies like Apache Spark have expanded on the MapReduce paradigm, offering enhanced performance and support for additional data processing paradigms, including real-time streaming and machine learning.

In summary, the MapReduce paradigm is a fundamental concept in the world of big data processing. It simplifies the parallel processing of large datasets by breaking down complex tasks into smaller, manageable steps, making it possible to harness the power of distributed computing for data-intensive applications.

Objective

Run a basic word count Map Reduce program to understand Map Reduce Paradigm.

Pre Lab Preparation

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Equipment Needed, Software required

Big data and business analytics laboratory (AITB16) lab manual and work sheets ubuntu

Background

WordCount is a simple program which counts the number of occurrences of each word in a given text input data set.

WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. Our implementation consists of three main parts:

1. Mapper
2. Reducer
3. Driver

Pre Lab Questions

1. What is MapReduce?
2. What are the main components of MapReduce?
3. Justify how hadoop technology satisfies the business insights now -a -days?
4. Define Filesystem?

Post Lab Preparation

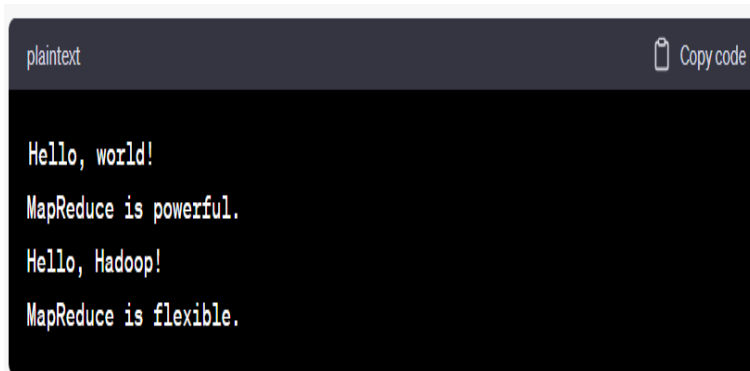
The Following of the steps are considered for implementation of Run a basic word count Map Reduce program to understand Map Reduce Paradigm

Procedure:

To run a basic Word Count MapReduce program, you will need access to a Hadoop cluster or a Hadoop installation on your local machine. Here's a step-by-step guide to running a Word Count program to understand the MapReduce paradigm:

Note: This example assumes you have Hadoop installed and configured. Adjust the paths and configurations according to your Hadoop setup.

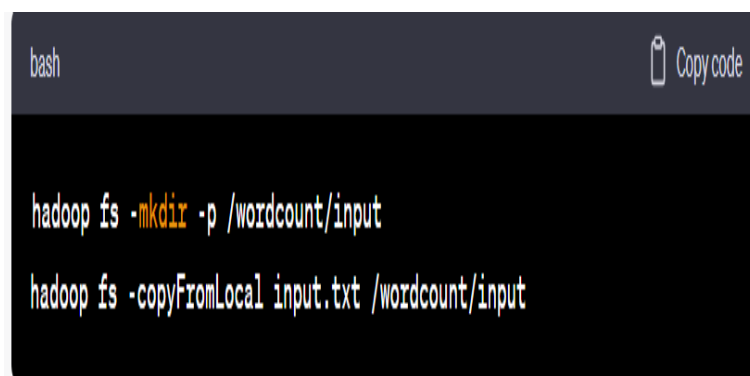
1. Create a text file with sample input data. You can use any text editor to create a file, e.g., input.txt, and add some text.

A screenshot of a text editor window with a dark background. The title bar at the top left says 'plaintext' and at the top right is a 'Copy code' button. The text inside the editor consists of four lines: 'Hello, world!', 'MapReduce is powerful.', 'Hello, Hadoop!', and 'MapReduce is flexible.'

```
plaintext Copy code

Hello, world!
MapReduce is powerful.
Hello, Hadoop!
MapReduce is flexible.
```

2. Upload the input file to HDFS.

A screenshot of a terminal window with a dark background. The title bar at the top left says 'bash' and at the top right is a 'Copy code' button. Two commands are entered: 'hadoop fs -mkdir -p /wordcount/input' and 'hadoop fs -copyFromLocal input.txt /wordcount/input'.

```
bash Copy code

hadoop fs -mkdir -p /wordcount/input
hadoop fs -copyFromLocal input.txt /wordcount/input
```

This creates a directory /wordcount/input in HDFS and copies the input.txt file to it.

3. Write the Word Count MapReduce code. Create a Java class, for example, WordCount.java, with the following code:

```
import java.io.IOException; import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
public class WordCount
public static class TokenizerMapper
extends Mapper<Object, Text, Text, IntWritable>{

private final static IntWritable one = new IntWritable(1);
private Text word = new Text();
```

```

public void map(Object key, Text value, Context context
) throws IOException, InterruptedException
StringTokenizer itr = new StringTokenizer(value.toString());
while (itr.hasMoreTokens())
word.set(itr.nextToken());
context.write(word, one);

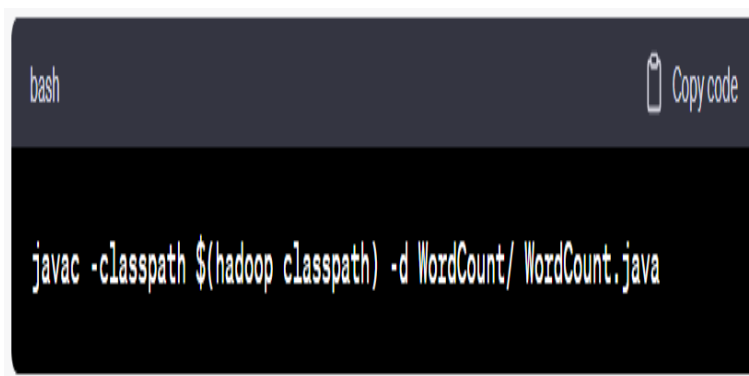
public static class IntSumReducer
extends Reducer<Text,IntWritable,Text,IntWritable>
private IntWritable result = new IntWritable();

public void reduce(Text key, Iterable<IntWritable> values,
Context context
) throws IOException, InterruptedException
int sum = 0;
for (IntWritable val : values)
sum += val.get();
}
result.set(sum);
context.write(key, result);
}
}

public static void main(String[] args) throws Exception
Configuration conf = new Configuration();
Job job = Job.getInstance(conf, "word count");
job.setJarByClass(WordCount.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);
job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path("/wordcount/input"));
FileOutputFormat.setOutputPath(job, new Path("/wordcount/output"));
System.exit(job.waitForCompletion(true) ? 0 : 1);
} }

```

Compile the Java code:



A terminal window with a dark background. The top bar is dark blue with the text 'bash' on the left and a 'Copy code' button on the right. The main area is black with green and white text showing the command: `javac -classpath $(hadoop classpath) -d WordCount/ WordCount.java`

Create a JAR file:

```
bash Copy code

jar -cvf wordcount.jar -C WordCount/ .
```

Run the Word Count MapReduce program:

```
bash Copy code

hadoop jar wordcount.jar WordCount /wordcount/input /wordcount/output
```

Replace /wordcount/input and /wordcount/output with appropriate HDFS paths.

Check the output:

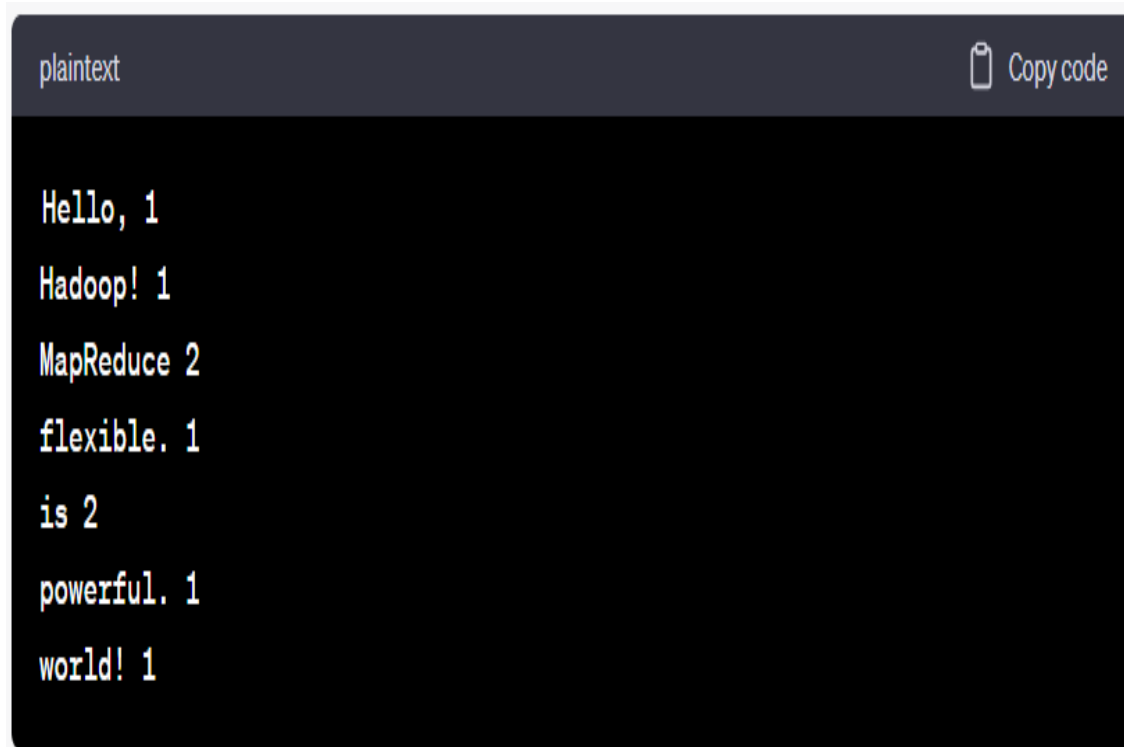
Once the job completes, you can view the word count results in the HDFS output directory:

```
bash Copy code

hadoop fs -cat /wordcount/output/part-r-00000
```

You should see the word count results like this:

Result and Analysis



```
plaintext Copy code

Hello, 1
Hadoop! 1
MapReduce 2
flexible. 1
is 2
powerful. 1
world! 1
```

This basic Word Count MapReduce program demonstrates the fundamental concepts of the MapReduce paradigm. It involves a mapper that processes input data and emits key-value pairs, a combiner (optional) that performs local aggregation, and a reducer that aggregates the intermediate results to produce the final output. This example can serve as a starting point for understanding and working with Hadoop MapReduce

VIVA VOCE QUESTIONS:

1. Define what is block in HDFS?
2. Why is a block in HDFS so large?

Further Probing Questions

1. MapReduce Conceptual Understanding:

What is the core idea behind the MapReduce programming model? How does MapReduce enable the processing of large-scale data in a distributed manner? What is the role of mappers and reducers in a MapReduce job?

2. MapReduce Components:

Can you explain the purpose of the Mapper and Reducer classes in the Word Count program? What is the function of the Context object in the Mapper and Reducer classes? Why is the IntSumReducer class used as both the combiner and the reducer in this example?

3. Input and Output Data:

What is the format of the input data that the Word Count program processes? How does the program

tokenize and process the input text data? What is the format of the output data generated by the Word Count program?

4.Hadoop Configuration:

What role does the Hadoop configuration play in a MapReduce program? How is the configuration set up in the Word Count program? Are there any specific configurations or properties that can be customized for MapReduce jobs?

5.Jar File Creation:

Why is it necessary to create a JAR file for the MapReduce program? What is the significance of specifying the main class when creating the JAR file? How can you include external dependencies or libraries in the JAR file?

6.Running a MapReduce Job:

What are the key steps involved in running a MapReduce job using the `hadoop jar` command? Can you explain the purpose of specifying input and output paths when running the job? How can you monitor the progress of a running MapReduce job?

7.Data Output and Interpretation:

How does the program handle the aggregation of word counts across different mappers and reducers? What does the final output of the Word Count program represent? Are there any additional steps or tools you can use to analyze or visualize the results further?

8.Scaling and Performance:

How would you modify the Word Count program to handle even larger datasets? What considerations should be taken into account when scaling a MapReduce job for big data processing? Are there any performance optimizations or best practices for MapReduce programming?

9.MapReduce Ecosystem:

Beyond the Word Count example, what are some real-world applications or use cases for MapReduce? How do other Hadoop ecosystem tools and libraries, such as Hive and Pig, relate to MapReduce?

10.Challenges and Limitations:

What are some challenges or limitations of the MapReduce paradigm, especially when dealing with complex data processing tasks?

These questions should help you gain a deeper understanding of MapReduce and its practical implementation in the context of a Word Count program. Feel free to explore each aspect further to enhance your knowledge of distributed data processing with Hadoop.

Lab-6 MAPREDUCE PROGRAM 2

Introduction

In Hadoop, MapReduce is a computation that decomposes large manipulation jobs into individual tasks that can be executed in parallel across a cluster of servers. The results of tasks can be joined together to compute final results.

Weather data plays a crucial role in various fields, from agriculture and disaster prediction to climate research and urban planning. Analyzing large volumes of weather data can provide valuable insights that help us make informed decisions. However, processing and extracting meaningful information from these vast datasets can be a daunting task. This is where MapReduce comes into play. MapReduce is a programming model and processing technique that can efficiently handle massive datasets by distributing the workload across multiple nodes in a cluster.

In this context, we will introduce a MapReduce program designed to mine weather data. This program leverages the power of parallel processing to perform data transformation, aggregation, and analysis on weather records, making it easier for meteorologists, researchers, and decision-makers to derive insights from historical and real-time weather data.

Why Use MapReduce for Weather Data Mining:

1.Scability: Weather data can be massive, with numerous data points collected over time and across various locations. MapReduce allows us to distribute the data and processing tasks across multiple machines, enabling the analysis of large-scale datasets in a reasonable amount of time.

2.Parallel Processing: MapReduce divides the data into smaller chunks, processes them in parallel, and then aggregates the results. This parallelism significantly accelerates data processing, making it well-suited for the high-throughput demands of weather data analysis.

3,Flexibility: MapReduce is a flexible framework that can handle a wide range of data transformations and analysis tasks. Whether you need to compute averages, detect anomalies, or generate forecasts, MapReduce can be adapted to suit your specific weather data mining needs.

4.Fault Tolerance: The MapReduce framework is designed to handle hardware failures gracefully. If a node in the cluster fails during processing, the framework automatically redistributes the task to other available nodes, ensuring the overall job's success.

Key Steps in a MapReduce Weather Data Mining Program:

1.Data Ingestion: The program starts by ingesting weather data from various sources, such as weather stations, satellites, or remote sensors. This data may include temperature, humidity, wind speed, precipitation, and more.

2.Data Preprocessing: Before analysis, the raw weather data often requires preprocessing to clean and format it correctly. This step may involve data validation, filtering out irrelevant information, and transforming data into a suitable format.

3.Map Phase: In this phase, the program distributes the preprocessed data across multiple nodes, and each node applies a "map" function to extract key information. For weather data, this might involve categorizing data points by location, time, or weather conditions.

4.Shuffle and Sort: After mapping, the framework sorts and shuffles the data, ensuring that related information is grouped together for the subsequent processing phase.

5.Reduce Phase: In the final phase, the program applies a "reduce" function to aggregate and analyze the data. This phase can involve various operations, such as calculating averages, identifying weather patterns, or generating reports.

By harnessing the power of MapReduce, weather data mining becomes a more efficient and scalable

process. This program enables us to gain valuable insights into past weather patterns, make predictions, and support decision-making in agriculture, disaster management, and many other fields reliant on accurate weather information. In the following sections, we will delve deeper into the implementation and specific use cases of MapReduce for weather data mining.

Objective

Write a Map Reduce program that mines weather data. Hint: Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with Map Reduce, since it is semi structured and record- oriented.

Pre lab Preparation

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Equipement Needed,Software required

Big data and business analytics laboratory (AITB16) lab manual and work sheets,Hadoop,VMWare

Background

WordCount is a simple program which counts the number of occurrences of each word in a given text input data set.

WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. Our implementation consists of three main parts:

1. Mapper
2. Reducer
3. Driver

Pre Lab Questions

- 1) Explain the function of MapReducer partitioner?
- 2) What is the difference between an Input Split and HDFS Block?
- 3) What is Sequencefileinputformat?

Post Lab Preparation

The Following of the steps are considered for implementation of Map Reduce program that mines weather data

Procedure:

Step-1. Write a Mapper

A Mapper overrides the `-map` function from the Class `"org.apache.hadoop.mapreduce.Mapper"` which provides `<key, value>` pairs as the input. A Mapper implementation may output `<key,value>` pairs using the provided Context . Input value of the WordCount Map task will be a line of text from the input data file and the key would be the line number `<line-number, line-of-text>` . Map task outputs `<word, one>` for each word in the line of text.

Pseudo-code

```
void Map (key, value){
for each max-temp x in value: output.collect(x, 1);
}
void Map (key, value){
```

for each min-temp x in value:

```
output.collect(x, 1);  
}
```

Step-2 Write a Reducer

A Reducer collects the intermediate <key,value> output from multiple map tasks and assemble a single result. Here, the WordCount program will sum up the occurrence of each word to pairs as <word, occurrence>

Pseudo-code

```
void Reduce (max-temp, <list of value>)  
{ for each x in <list of value>:  
  sum+=x; final-output.collect(max-temp, sum);  
}  
void Reduce (min-temp, <list of value>){ for each x in <list of value>:  
  sum+=x; final-output.collect(min-temp, sum);  
}
```

3. Write Driver

The Driver program configures and run the MapReduce job. We use the main program to perform basic configurations such as:

Job Name : name of this Job Executable (Jar)

Class: the main executable class. For here, WordCount.

Mapper Class: class which overrides the "map" function. For here, Map. Reducer: class which override the "reduce" function.

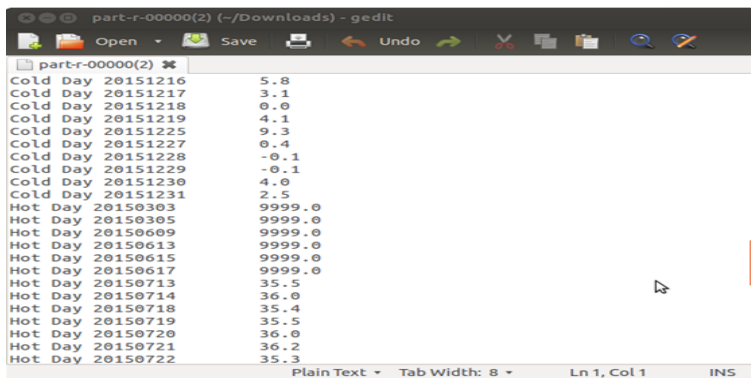
For here , Reduce. Output Key: type of output key. For here, Text.Output

Value: type of output value. For here, IntWritable. File Input Path

File Output Path

RESULTS AND ANALYSIS

Set of Weather Data over the years



Cold Day 20151216	5.8
Cold Day 20151217	3.1
Cold Day 20151218	0.0
Cold Day 20151219	4.1
Cold Day 20151225	9.3
Cold Day 20151227	0.4
Cold Day 20151228	-0.1
Cold Day 20151229	-0.1
Cold Day 20151230	4.0
Cold Day 20151231	2.5
Hot Day 20150303	9999.0
Hot Day 20150305	9999.0
Hot Day 20150609	9999.0
Hot Day 20150613	9999.0
Hot Day 20150615	9999.0
Hot Day 20150617	9999.0
Hot Day 20150713	35.5
Hot Day 20150714	36.0
Hot Day 20150718	35.4
Hot Day 20150719	35.5
Hot Day 20150720	36.0
Hot Day 20150721	36.2
Hot Day 20150722	35.3

LAB ASSIGNMENT:

1. Using Map Reduce job to Identify language by merging multi language dictionary files into a single dictionary file.
2. Join multiple datasets using a MapReduce Job.

VIVA VOCE QUESTIONS:

- 1) In Hadoop what is InputSplit?
- 2) Explain what is a sequence file in Hadoop?

FURTHER PROBING QUESTIONS:

1.Data Sources and Ingestion:

What are the primary sources of weather data that your MapReduce program accesses? How do you ensure data quality and consistency during the data ingestion process? Are there any challenges or considerations specific to handling real-time weather data streams?

2.Data Preprocessing:

Can you describe the data preprocessing steps in more detail? What kind of data cleaning and formatting are involved? Are there any data transformation techniques or algorithms applied before the data is fed into the MapReduce framework? How do you handle missing or incomplete data points in the weather dataset?

3.Map Phase:

What specific tasks or operations are performed in the "map" phase of your MapReduce program? How do you distribute data across nodes for parallel processing, and what criteria do you use for data partitioning? Can you provide examples of key-value pairs generated during the mapping phase for weather data analysis?

4.Shuffle and Sort:

How does the framework handle data shuffling and sorting to prepare data for the "reduce" phase? Are there any optimizations or techniques employed to minimize data movement between nodes? What considerations are made to ensure efficient grouping of related data during this phase?

5.Reduce Phase:

In the "reduce" phase, what types of analysis or computations are typically performed on weather data? Can you elaborate on the specific weather-related insights or patterns that your program aims to extract? Are there any challenges related to scalability or computational complexity when dealing with large datasets in this phase?

6.Data Security and Privacy:

How do you handle sensitive or private weather data to ensure compliance with data protection regulations? What security measures are in place to protect weather data during the MapReduce process? Are there any anonymization or encryption techniques used when dealing with user-specific weather data?

7.Fault Tolerance and Data Recovery:

Could you explain the mechanisms in place to handle node failures or errors during data processing? How is data recovery managed in the event of hardware or software failures within the MapReduce cluster? Are there any backup strategies for ensuring the integrity of the weather data being processed?

8.Scalability and Performance:

What strategies or technologies are employed to scale the MapReduce program for handling increasing volumes of weather data? How do you optimize the program's performance to deliver timely results, especially for real-time weather analysis? Have you encountered any performance bottlenecks or challenges, and how were they addressed?

9. Use Cases and Applications:

Can you provide examples of practical use cases or applications where your MapReduce program for weather data mining has been particularly valuable? Are there any specific industries or research fields that have benefited significantly from the insights derived from weather data analysis using this program?

10. Future Developments:

Are there any plans for future enhancements or extensions to your MapReduce program for weather data mining? Are there emerging technologies or trends in data analysis and processing that you anticipate integrating into your program?

These probing questions should help you gain a deeper understanding of the technical and practical aspects of a MapReduce program designed for mining weather data.

Lab-7 MAPREDUCE PROGRAM 3

Introduction

MapReduce is a technique in which a huge program is subdivided into small tasks and run parallelly to make computation faster, save time, and mostly used in distributed systems. It has 2 important parts:

- **Mapper:** It takes raw data input and organizes into key, value pairs. For example, In a dictionary, you search for the word “Data” and its associated meaning is “facts and statistics collected together for reference or analysis”. Here the Key is Data and the Value associated with is facts and statistics collected together for reference or analysis.
- **Reducer:** It is responsible for processing data in parallel and produce final output.

Objective

Write a Map Reduce program that mines weather data. Hint: Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with Map Reduce, since it is semi structured and record- oriented.

Pre Lab:

1. Explain what is “map” and what is “reducer” in Hadoop?
2. Mention what daemons run on a master node and slave nodes?
3. Mention what is the use of Context Object?

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Equipment Needed,Software required

Big data and business analytics laboratory (AITB16) lab manual and work sheets,VMWare,Hadoop

Background:

- The input information of the reduce() step (function) of the MapReduce algorithm are: One row vector from matrix A. One column vector from matrix B.
- The reduce() function will compute: The inner product of the. One row vector from matrix A. One column vector from matrix B.

1. Mapper
2. Reducer
3. Main program

Post Lab

The Following of the steps are considered for developments of Map Reduce program that mines weather data.

Procedure:

Implementing matrix multiplication using Hadoop MapReduce involves breaking down the matrix multiplication algorithm into a series of Map and Reduce operations. This process can be complex,

but I'll provide a simplified example to help you understand the key steps involved. In practice, matrix multiplication on large matrices is computationally intensive and benefits from Hadoop's parallel processing capabilities.

Assumptions: For simplicity, let's assume we have two square matrices, Matrix A and Matrix B, both of size $n \times n$.

MapReduce Steps:

Map Phase:

In this phase, we will emit key-value pairs where the key represents the matrix cell's position (i, j), and the value contains the matrix name (A or B) and the cell value.

```
python Copy code

Mapper Input (Matrix A):
(i, j), A_ij

Mapper Input (Matrix B):
(j, k), B_jk
```

The Mapper processes each cell of Matrix A and Matrix B and emits intermediate key-value pairs:

```
Mapper Output:

(i, k), (A_ij, B_jk)
```

Shuffle and Sort Phase: Hadoop automatically groups intermediate key-value pairs by their keys, so all values corresponding to the same (i, k) pair will be grouped together.

```
Shuffled and Sorted Data:

(i, k), [(A_ij, B_jk1), (A_ij, B_jk2), ...]
```


Reduce Phase:

In the Reduce phase, we calculate the dot product of the values associated with the same (i, k) key. This is where the matrix multiplication takes place. For each (i, k) key, we iterate through the list of (A_{ij}, B_{jk}) pairs and calculate the sum of products.

Reducer Input:

(i, k), [(A_{ij1}, B_{jk1}), (A_{ij2}, B_{jk2}), ...]

Reducer Output:

(i, k), C_{ik}

RESULT AND ANALYSIS:

The final output will be a key-value pair where the key represents the resulting matrix cell's position (i, k), and the value contains the computed value C_{ik}.

Pseudo-Code for MapReduce:

```
Mapper:
map(key, value):
    # key: matrix cell position (i, j) or (j, k)
    # value: matrix cell value
    if key represents a cell in Matrix A:
        for k in range(matrix_dimension):
            emit((i, k), ('A', value))
    else: # key represents a cell in Matrix B
        for i in range(matrix_dimension):
            emit((i, k), ('B', value))

Reducer:
reduce(key, values):
    # key: (i, k)
    # values: list of ('A' or 'B', value)
    total = 0
    for v in values:
        if v[0] == 'A':
            A_ij = v[1]
        else: # 'B'
            B_jk = v[1]
    total += A_ij * B_jk
    emit(key, total)
```

LAB ASSIGNMENT:

1. Implement matrix addition with Hadoop Map Reduce.

VIVA Voce QUESTIONS:

1. What is partitioner in Hadoop?
2. Explain of RecordReader in Hadoop?

FURTHER PROBING QUESTIONS

1. Matrix Size and Scalability:

What are the practical limitations or considerations for the size of matrices that can be efficiently multiplied using Hadoop MapReduce? How does the size of the input matrices impact the distribution of computation across the Hadoop cluster?

2. Data Partitioning:

Can you describe the strategies used for partitioning and distributing matrix data across mapper

tasks? How does this affect load balancing? Are there any challenges related to data skew when partitioning matrices for MapReduce?

3.Matrix Input Formats:

What input formats are commonly used for providing matrix data to the MapReduce job? (e.g., text-based formats like CSV or custom binary formats) How do you handle sparse matrices where many values are zero?

4.Combiners and Optimization:

Have you implemented any combiners or optimization techniques to reduce data transfer during the shuffle and sort phase? What considerations are made to minimize network traffic and intermediate data size in matrix multiplication?

5.Parallelism and Performance:

How is parallelism achieved in the matrix multiplication process, and how does it relate to the number of available mapper and reducer tasks? Can you discuss any performance benchmarks or optimizations that have been applied to speed up matrix multiplication?

6.Matrix Output Format:

What output format is typically used to represent the resulting matrix C after multiplication?

Lab-8 PIG LATIN LANGUAGE - PIG

Introduction

Pig is an open-source high level data flow system. It provides a simple language called Pig Latin, for queries and data manipulation, which are then compiled in to MapReduce jobs that run on Hadoop.

Apache Pig is a high-level platform and scripting language designed for processing and analyzing large datasets in a Hadoop ecosystem. It was developed by Yahoo! and is now an open-source project managed by the Apache Software Foundation. Pig simplifies the process of working with big data by providing a user-friendly abstraction over complex Hadoop MapReduce jobs. Here's an introduction to Apache Pig:

1. Why Pig?

Simplicity: Pig offers a simple and easy-to-understand scripting language called Pig Latin, which abstracts the complexities of writing low-level MapReduce programs. **Flexibility:** Pig is flexible and can handle a wide range of data sources and formats, making it suitable for various data processing tasks. **Scalability:** Pig is designed to work with large-scale datasets, making it a valuable tool for big data processing.

2. Pig Latin:

Pig Latin is the scripting language used with Apache Pig. It is similar to SQL in some ways but is specifically designed for processing data on Hadoop. Pig Latin scripts are easy to write and read, making them accessible to both data analysts and engineers.

3. Data Flow Language:

Pig Latin scripts describe a series of data transformations and operations as a directed acyclic graph (DAG). These transformations are represented as a sequence of data flows. Operations in Pig Latin can be categorized into two main types: transformations (such as filtering, grouping, and joining) and actions (such as storing data or displaying results).

4. Common Operations:

LOAD: Reads data from various sources like HDFS, local file systems, and HBase. **FILTER:** Filters rows based on a specified condition. **GROUP:** Groups data based on one or more fields. **JOIN:** Combines two or more datasets based on a common field. **FOREACH:** Applies transformations to each tuple in a relation. **STORE:** Writes the results of a Pig Latin script to a storage system.

5. Execution:

Pig Latin scripts are executed using the Pig runtime environment. Pig converts these scripts into a series of MapReduce jobs that run on the Hadoop cluster.

6. Ecosystem Integration:

Pig integrates seamlessly with other Hadoop ecosystem components like HBase, Hive, and HDFS, allowing users to leverage the capabilities of these tools in their data processing workflows.

7. User-Defined Functions (UDFs):

Pig supports custom User-Defined Functions (UDFs) written in Java, Python, and other languages, which allows users to extend Pig's functionality to suit their specific needs.

8. Use Cases:

Apache Pig is commonly used for ETL (Extract, Transform, Load) tasks, data cleaning, data transformation, log processing, and more. It is widely adopted in industries where big data processing is essential, including finance, e-commerce, social media, and healthcare.

In summary, Apache Pig is a powerful tool for simplifying and streamlining the process of working with large-scale datasets on Hadoop clusters. Its user-friendly scripting language, Pig Latin, and integration with the Hadoop ecosystem make it a valuable choice for data analysts and engineers dealing with big data processing tasks.

Objective

Installation of PIG.

Pre lab Preparation

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Equipment Needed

Big data and business analytics laboratory (AITB16) lab manual and work sheets.

Background:

By using Pig, we can perform all the data manipulation operations in Hadoop. In addition, Pig offers a high-level language to write data analysis programs which we call as Pig Latin. One of the major advantages of this language is, it offers several operators.

Pre Lab Questions

1. What do you mean by a bag in Pig?
- 2) Differentiate between PigLatin and HiveQL
- 3) How will you merge the contents of two or more relations and divide a single relation into two or more relations?

Post Lab

The Following of the steps are considered for Installation of PIG

Procedure:

- 1) Extract the pig-0.15.0.tar.gz and move to home directory
- 2) Set the environment of PIG in bashrc file.
- 3) Pig can run in two modes Local Mode and Hadoop Mode Pig -x local and pig
- 4) Grunt Shell Grunt >
- 5) LOADING Data into Grunt Shell
DATA = LOAD <CLASSPATH> USING PigStorage(DELIMITER) as (ATTRIBUTE :
DataType1, ATTRIBUTE : DataType2.....)
- 6) Describe Data Describe DATA;
- 7) DUMP Data Dump DATA;

RESULTS AND ANALYSIS

Input as Website Click Count Data

```

lendi@ubuntu: ~
grunt> ad1 = load '/home/lendi/Desktop/static_data/ad_data/ad_data1.txt' using PigStorage(',') as (item:chararray,campaignId:chararray,date:chararray,time:chararray,display_site:chararray,was_clicked:int,cpc:int,country:chararray,placement:chararray);
2016-10-14 02:35:32,441 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-10-14 02:35:32,441 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> describe ad1;
ad1: {item: chararray,campaignId: chararray,date: chararray,time: chararray,display_site: chararray,was_clicked: int,cpc: int,country: chararray,placement: chararray}
grunt> ad2 = load '/home/lendi/Desktop/static_data/ad_data/ad_data2.txt' using PigStorage(',') as (campaignId:chararray,date:chararray,time:chararray,display_site:chararray,placement:chararray,was_clicked:int,cpc:int,item:chararray);
2016-10-14 02:36:08,732 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-10-14 02:36:08,732 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> describe ad2;
ad2: {campaignId: chararray,date: chararray,time: chararray,display_site: chararray,placement: chararray,was_clicked: int,cpc: int,item: chararray}
grunt>

```

LAB ASSIGNMENT:

1. Process baseball data using Apache Pig.

VIVA Voce QUESTIONS:

1. What is the usage of foreach operation in Pig scripts?
2. What does Flatten do in Pig

FURTHER PROBING QUESTIONS

1. Operating System: Which operating system are you installing Pig on? (e.g., Linux, Windows, macOS)
2. Hadoop Version: Are you planning to run Pig on a Hadoop cluster, and if so, what version of Hadoop are you using?
3. Installation Location: Where did you install Apache Pig on your system? Can you provide the full path to the Pig installation directory?
4. Environment Variables: Have you set up any environment variables for Pig, such as PIG_HOME and PATH? How did you configure these variables, and are they working correctly?
5. Configuration Files: Did you customize any Pig configuration files (e.g., pig.properties)? What changes did you make, if any?
6. Java Setup: Have you verified that Java is correctly installed and configured on your system? What version of Java are you using?
7. Hadoop Integration: If you are running Pig on a Hadoop cluster, have you configured Pig to work with your Hadoop cluster properly? Are the Hadoop configuration files set up correctly?
8. Testing Pig: Have you tested your Pig installation by running a simple Pig Latin script? If so, what was the script, and did it execute successfully?

Lab-9 PIG COMMANDS

Introduction

Pig is a high level scripting language that is used with Apache Hadoop. ... Pig works with data from many sources, including structured and unstructured data, and store the results into the Hadoop Data File System. Pig scripts are translated into a series of MapReduce jobs that are run on the Apache Hadoop cluster.

Apache Pig commands are used to interact with the Apache Pig platform and execute data processing tasks on large datasets. Pig provides both interactive and batch processing modes, allowing users to work with data using a high-level scripting language called Pig Latin. Here's an introduction to some of the essential Pig commands and their functions:

1.Grunt Shell:

Command: `pig` (without any arguments) Description: Launches the Pig Grunt shell, which allows you to interactively write and execute Pig Latin scripts. It provides a REPL (Read-Eval-Print Loop) interface for running Pig commands line by line.

2.Running a Pig Script:

Command: `pig -x local script.pig` or `pig -x mapreduce script.pig` Description: Executes a Pig Latin script stored in a file. The `-x` flag specifies the execution mode, which can be either local (for testing on a local machine) or mapreduce (for running on a Hadoop cluster).

3.Loading Data:

Command: `LOAD` Description: Loads data from various sources, such as HDFS, local file systems, HBase, and more, into Pig relations. You specify the data source and the schema (if necessary).

4.Storing Data:

Command: `STORE` Description: Writes the results of Pig Latin operations to a storage location. This can be HDFS or another supported storage system. You specify the relation to store and the destination.

5.Dumping Data:

Command: `DUMP` Description: Displays the contents of a Pig relation on the console. It is often used for debugging purposes to inspect intermediate data.

6.Filtering Data:

Command: `FILTER` Description: Filters rows in a relation based on a specified condition. It is used to select a subset of the data that meets specific criteria.

7.Grouping Data:

Command: `GROUP` Description: Groups data in a relation based on one or more fields. Typically used before aggregation operations like counting or summing.

8.Sorting Data:

Command: `ORDER` Description: Sorts data in a relation based on one or more fields in ascending or descending order. Useful for arranging data in a specific order.

9.Joining Data:

Command: `JOIN` Description: Combines two or more relations based on a common field. Joins are used to combine data from multiple sources into a single dataset.

10.Foreach and Generate:

Command: `FOREACH` and `GENERATE` Description: The `FOREACH` command allows you to apply a transformation or operation to each tuple in a relation, while `GENERATE` specifies what fields or expressions to generate in the output.

11.Defining User-Defined Functions (UDFs):

Command: `REGISTER` Description: Registers external User-Defined Functions (UDFs) written in Java, Python, or other languages. UDFs can extend Pig's functionality.

12.Illustrate:

Command: `ILLUSTRATE` Description: Helps users understand the execution plan of a Pig Latin script by showing how data flows through the operations step by step. Useful for debugging and

optimization.

Objective

Write Pig Latin scripts sort, group, join, project, and filter your data

PRE LAB PREPARATION

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Equipement Needed

Big data and business analytics laboratory (AITB16) lab manual and work sheets.

Background:

- Apache Pig a tool/platform which is used to analyze large datasets and perform long series of data operations. Pig is used with Hadoop. All pig scripts internally get converted into map-reduce tasks and then get executed. It can handle structured, semi-structured and unstructured data. Pig stores, its result into HDFS

Pre lab Questions

1. How will you merge the contents of two or more relations and divide a single relation into two or more relations?
2. What is the usage of foreach operation in Pig scripts?
3. What does Flatten do in Pig?

Post Lab Preparation

The following of the steps are considered for Write Pig Latin scripts sort, group, join, project, and filter your data

Procedure:

FILTER Data

FDATA = FILTER DATA by ATTRIBUTE = VALUE;

GROUP Data

GDATA = GROUP DATA by ATTRIBUTE;

Iterating Data

FOR-DATA = FOREACH DATA GENERATE GROUP AS GROUP-FUN,
ATTRIBUTE = <VALUE>

Sorting Data

SORT-DATA = ORDER DATA BY ATTRIBUTE WITH CONDITION;

LIMIT Data

LIMIT-DATA = LIMIT DATA COUNT;

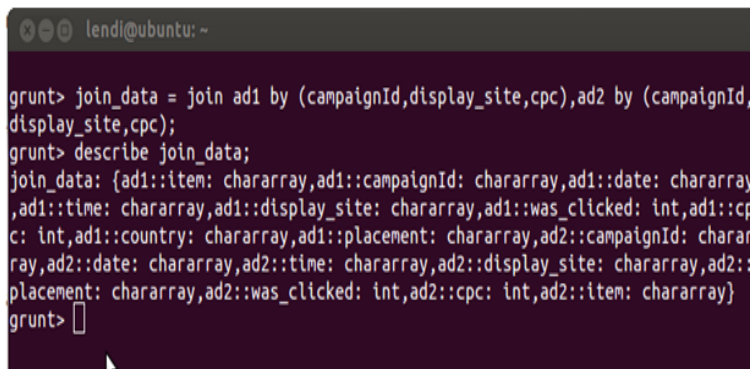
JOIN Data

JOIN DATA1 BY (ATTRIBUTE1,ATTRIBUTE2....) , DATA2 BY (ATTRIBUTE3,ATTRIBUTE....N)

RESULT AND ANALYSIS

Input as Website Click Count Data

LAB ASSIGNMENT:

A terminal window with a dark background and light text. The prompt is 'lendi@ubuntu: ~'. The user enters 'grunt> join_data = join ad1 by (campaignId,display_site,cpc),ad2 by (campaignId,display_site,cpc);'. The prompt changes to 'grunt> describe join_data;'. The output is a multi-line string describing the schema of 'join_data', listing fields like 'ad1::item', 'ad1::campaignId', 'ad1::date', 'ad1::time', 'ad1::display_site', 'ad1::was_clicked', 'ad1::cpc', 'ad1::country', 'ad1::placement', 'ad2::campaignId', 'ad2::date', 'ad2::time', 'ad2::display_site', 'ad2::placement', 'ad2::was_clicked', 'ad2::cpc', and 'ad2::item' with their respective data types. The prompt returns to 'grunt>'.

1. Using Apache Pig to develop User Defined Functions for student data.

VIVA VOCE QUESTIONS:

1. What do you mean by a bag in Pig?
2. Differentiate between PigLatin and HiveQL

FURTHER PROBING QUESTIONS

Sorting Data:

- 1.Sorting Order: In your sorting script, do you typically sort data in ascending or descending order? How do you specify the sorting order?
- 2.Sorting Keys: What criteria or columns do you use as keys for sorting data? Are there multiple columns involved in sorting?
- 3.Performance Considerations: Have you encountered any performance issues with sorting large datasets? How do you optimize the sorting process?

Grouping Data:

- 1.Grouping Criteria: When grouping data, what criteria or columns do you use for grouping? Are there cases where you group data based on multiple columns?
- 2.Aggregation: After grouping, do you typically perform aggregation operations like counting or summing within each group?
- 3.Handling Null Values: How do you handle cases where the grouping key contains null values or missing data?

Joining Data:

- 1.Join Types: What types of joins do you frequently perform (e.g., inner join, outer join, left join, right join)? Can you provide examples of scenarios where you use each type?
- 2.Join Conditions: How do you specify the join conditions in your scripts? Are there complex join conditions that involve multiple columns or criteria?
- 3.Performance Optimization: Have you encountered performance challenges with joins, especially when dealing with large datasets? How do you optimize join operations?

Projecting Data (Selecting Columns):

- 1.Column Selection: In your projection scripts, do you often select a specific subset of columns from the input data? How do you specify the columns to include in the output?
- 2.Data Transformation: Are there cases where you perform data transformations in your projection step, such as data type conversions or calculations?

Filtering Data:

- 1.Filter Conditions: How do you specify filter conditions in your scripts? Are they based on simple comparisons (e.g., greater than, equal to) or more complex criteria?

2. Multiple Filters: Do you apply multiple filters in a sequence in your scripts? How do you combine or chain filter conditions?

3. Handling Missing Data: How do you handle cases where data may have missing values or nulls in filter conditions?

4. Performance Impact: Have you observed any performance impacts when applying filters to large datasets? How do you optimize filter operations?

5. Error Handling: How do you handle cases where filter conditions are not met, and data may be excluded from the output?

6. Boolean Operations: Do you use Boolean operations (AND, OR, NOT) in your filter conditions? Can you provide examples of such conditions?

Understanding these aspects of sorting, grouping, joining, projecting, and filtering data in your Pig Latin scripts can help in optimizing your data processing workflows and addressing specific challenges you may encounter with your datasets and business requirements.

Lab-10 PIG LATIN MODES, PROGRAMS

Introduction

To start with the word count in pig Latin, you need a file in which you will have to do the word count. It is a PDF file and so you need to first convert it into a text file which you can easily do using any PDF to text converter. Here are both PDF and Text file for your reference. It is recommended for you to download both the file to start with word count example in pig Latin.

In Apache Pig, "Pig Latin" refers to the high-level scripting language used for expressing data processing and analysis tasks. Pig Latin is designed to work with large datasets and abstracts away many of the complexities of writing low-level MapReduce code. Pig Latin scripts can be executed in different modes, and Pig programs consist of these scripts along with the necessary data processing logic. Here's an introduction to Pig Latin modes and programs:

Pig Latin Modes:

Local Mode:

Description: In Local Mode, Pig runs on a single machine without the need for a Hadoop cluster. It is primarily used for development, testing, and debugging purposes. Use Cases: Local Mode is suitable for small-scale data processing when you want to quickly test and debug Pig scripts on a local dataset.

MapReduce Mode:

Description: In MapReduce Mode, Pig runs on a Hadoop cluster, leveraging the Hadoop MapReduce framework for distributed data processing. Use Cases: MapReduce Mode is used for processing large-scale datasets in a distributed and parallelized manner on a Hadoop cluster. It's ideal for production data processing.

Pig Programs:

Pig Script:

Description: A Pig script is a sequence of Pig Latin commands and statements written in a file with a .pig extension. It defines the data processing steps and transformations to be applied to the data. Use Cases: Pig scripts are used for expressing data processing tasks in a high-level and readable format. They are the core of Pig programs.

User-Defined Functions (UDFs):

Description: Pig allows you to write custom User-Defined Functions (UDFs) in languages like Java, Python, and more. UDFs can be called from Pig scripts to perform specialized processing. Use Cases: UDFs are used when you need to apply custom logic or operations to your data that cannot be expressed using built-in Pig Latin functions.

Pig Latin Operators:

Description: Pig provides a rich set of built-in operators and functions (e.g., LOAD, FILTER, GROUP, JOIN, FOREACH, STORE, etc.) that you can use in Pig scripts to perform data transformations and operations. Use Cases: Pig operators are used for common data processing tasks like loading data, filtering, grouping, joining, and aggregating data.

Execution Plan:

Description: Pig automatically generates an execution plan for Pig scripts in both Local Mode and MapReduce Mode. This plan outlines the sequence of steps Pig will take to process the data. Use Cases: The execution plan helps users understand how Pig will execute their scripts, making it useful for optimization and troubleshooting.

Pig Latin Program Logic:

Description: Pig programs combine Pig scripts with data loading, execution, and storage logic. These programs may also include error handling, logging, and orchestration of Pig script executions. Use

Cases: Pig programs provide a complete framework for data processing tasks, making it easier to manage, schedule, and automate data processing workflows.

In summary, Pig Latin modes (Local and MapReduce) determine where and how Pig scripts are executed, while Pig programs encompass the scripts themselves, along with any custom UDFs, operators, and program logic needed to perform data processing tasks. Pig's flexibility and high-level abstractions make it a valuable tool for big data processing in various use cases.

Objective

1. Run the Pig Latin Scripts to find Word Count. 2. Run the Pig Latin Scripts to find a max temp for each and every year.

Pre lab Preparation

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Equipement Needed

Big data and business analytics laboratory (AITB16) lab manual and work sheets.

Background:

- Apache Pig a tool/platform which is used to analyze large datasets and perform long series of data operations. Pig is used with Hadoop. All pig scripts internally get converted into map-reduce tasks and then get executed. It can handle structured, semi-structured and unstructured data. Pig stores, its result into HDFS

Pre Lab Questions

1. List out the benefits of Pig?
2. Classify Pig Latin commands in Pig?

Post Lab

The Following of the steps are considered for 1. Run the Pig Latin Scripts to find Word Count. 2. Run the Pig Latin Scripts to find a max temp for each and every year.

Procedure:

Run the Pig Latin Scripts to find Word Count.

```
lines = LOAD '/user/hadoop/HDFS-File.txt' AS (line:chararray);
words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word; grouped = GROUP
words BY word;
wordcount = FOREACH grouped GENERATE group, COUNT(words); DUMP wordcount;
Run the Pig Latin Scripts to find a max temp for each and every year
```

– max-temp.pig: Finds the maximum temperature by year records = LOAD 'input/ncdc/micro-tab/sample.txt'
AS (year:chararray, temperature:int, quality:int);

```

filtered-records = FILTER records BY temperature != 9999 AND
(quality == 0 OR quality == 1 OR quality == 4 OR quality == 5 OR quality == 9); grouped-
records = GROUP filtered-records BY year; max-temp = FOREACH grouped-records GENERATE
group, MAX(filtered-records.temperature);
DUMP max-temp;

```

RESULT AND ANALYSIS

```

(1950,0,1)
(1950,22,1)
(1950,-11,1)
(1949,111,1)
(1949,78,1)

```

LAB ASSIGNMENT:

1. Analyzing average stock price from the stock data using Apache Pig.

VIVA VOCE QUESTIONS:

1. Discuss the modes of Pig scripts?
2. Explain the Pig Latin application flow?

Further Probing Questions

Pig Latin Modes:

1. Local Mode:

Use Cases: Can you provide specific use cases where Local Mode is beneficial over MapReduce Mode?

Performance Considerations: How do you ensure that performance in Local Mode is representative of MapReduce Mode in a cluster environment? Data Size: What is the typical size of datasets you work with in Local Mode? When do you decide to switch to MapReduce Mode for larger datasets?

2. MapReduce Mode:

Cluster Configuration: How is your Hadoop cluster configured for running Pig in MapReduce Mode?

Are there any cluster-specific considerations? Scaling: How do you scale your cluster to handle larger workloads and more data? Monitoring: What tools or practices do you use to monitor Pig jobs running in MapReduce Mode for performance and resource utilization?

Pig Programs:

1. Pig Scripts:

Script Complexity: Can you describe the complexity of your Pig scripts? Are they simple data transformations, or do they involve complex data processing logic? Script Maintenance: How do you manage and maintain Pig scripts, especially as they grow in size and complexity? Version Control: Do you use version control systems (e.g., Git) to track changes and collaborate on Pig scripts?

2. User-Defined Functions (UDFs):

UDF Types: What types of UDFs do you commonly use (e.g., Java, Python)? Can you provide examples of scenarios where custom UDFs are essential? Testing UDFs: How do you test and validate

the correctness and performance of your custom UDFs?

3.Pig Latin Operators:

Operator Choices: Can you explain your decision-making process when choosing Pig operators for specific data processing tasks? Do you prefer built-in operators over custom UDFs in certain situations? Operator Optimization: Are there any optimization techniques or best practices you follow when using Pig operators?

4.Execution Plan:

Understanding Plans: How do you interpret and analyze the execution plans generated by Pig? What information do you look for when optimizing your scripts? Optimization: Have you performed any optimizations based on insights from the execution plan? Can you share examples of optimizations?

5.Pig Latin Program Logic:

Program Complexity: How do you manage the complexity of Pig programs as they grow? Are there specific design patterns or practices you follow?

Error Handling: How do you handle errors and exceptions within Pig programs? Are there strategies for graceful error recovery?

Orchestration: How do you schedule and orchestrate the execution of Pig programs within larger data workflows? Do you use any workflow management tools?

By asking these probing questions, you can gain a deeper understanding of how Pig modes and programs are used in your specific context, helping you optimize and manage your data processing tasks more effectively

Lab-11 HIVE

Introduction

The Apache Hive data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL. Structure can be projected onto data already in storage. A command line tool and JDBC driver are provided to connect users to Hive.

Apache Hive is a data warehousing and SQL-like query language system built on top of Hadoop. It provides a high-level interface to query and analyze data stored in Hadoop Distributed File System (HDFS) or other compatible storage systems. Hive is part of the Hadoop ecosystem and is widely used for its simplicity in querying and summarizing large datasets. Here's an introduction to Hive:

Key Features of Apache Hive:

1. **SQL-Like Query Language:** Hive Query Language (HiveQL) is similar to SQL, making it accessible to analysts and data scientists who are already familiar with SQL. It allows users to express complex data transformations and aggregations.
2. **Schema-on-Read:** Unlike traditional relational databases with a fixed schema, Hive follows a schema-on-read approach. Data can be ingested into Hive without specifying a schema upfront, making it suitable for handling semi-structured and unstructured data.
3. **Data Integration:** Hive can seamlessly integrate with various data sources, including HDFS, HBase, and external databases. This versatility allows users to query and analyze data from diverse data storage systems.
4. **Built-In UDFs:** Hive provides a set of built-in User-Defined Functions (UDFs) for data processing and analysis. Users can also create custom UDFs in programming languages like Java, Python, and more to extend Hive's functionality.
5. **Metastore:** Hive uses a metastore to store metadata about tables, columns, partitions, and data locations. This centralized metadata store simplifies schema management and table discovery.
6. **Optimized Execution:** Hive can generate optimized execution plans, which may include using the Hadoop MapReduce framework or Apache Tez for query execution. Users can also leverage cost-based optimization for query performance.
7. **Partitioning and Bucketing:** Hive supports data partitioning and bucketing, allowing users to efficiently organize and access data based on specific criteria. This enhances query performance when working with large datasets.
8. **Extensibility:** Hive can be extended with custom SerDes (Serialization/Deserialization) to handle various data formats, enabling compatibility with a wide range of file formats and data types.
9. **Integration with Ecosystem:** Hive integrates seamlessly with other Hadoop ecosystem components like Pig, HBase, and Spark, allowing users to build comprehensive data processing pipelines.

Use Cases for Apache Hive:

1. **Data Warehousing:** Hive is commonly used for building data warehouses on top of Hadoop. It allows organizations to store, manage, and analyze large volumes of structured and semi-structured data.
2. **Batch Processing:** Users can write HiveQL queries to perform batch processing tasks like data transformation, ETL (Extract, Transform, Load), and data cleaning.
3. **Log Analysis:** Organizations use Hive to analyze log files and gain insights from web server logs, application logs, and more.
4. **Data Exploration:** Data analysts and data scientists use Hive to explore large datasets and generate reports, summaries, and visualizations for decision-making.
5. **Ad-Hoc Queries:** Hive is valuable for running ad-hoc queries on big data, as it provides a familiar SQL interface for querying Hadoop data.
6. **Structured Data Analysis:** Although Hive can handle semi-structured data, it is also suitable for structured data analysis when schema-on-read flexibility is required.

Objective

Installation of HIVE.

Pre Lab Preparation

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Equipement Needed,Software needed

Big data and business analytics laboratory (AITB16) lab manual and work sheets, VMWare,Hadoop

Background:

Apache Hive is a data warehouse software project built on top of Apache Hadoop for providing data query and analysis.

Pre Lab Questions

1. In Hive, explain the term ‘aggregation’ and its uses?
2. List out the Data types in Hive?

Post Lab

The following of the steps are considered for Installation of HIVE.

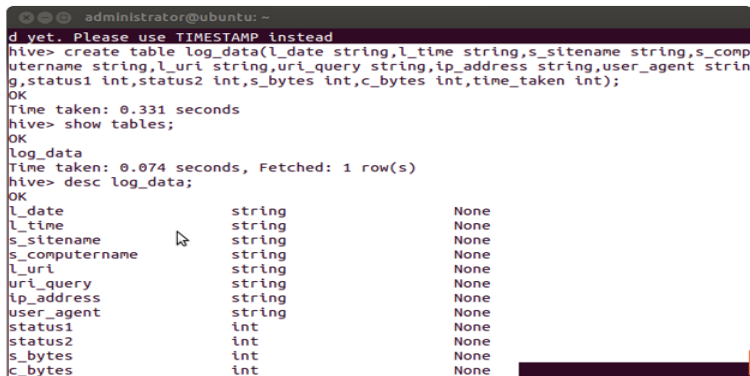
Procedure:

Install MySQL-Server

- 1) Sudo apt-get install mysql-server
- 2) Configuring MySQL UserName and Password
- 3) Creating User and granting all Privileges Mysql –uroot –proot
Create user <USER-NAME> identified by <PASSWORD>
- 4) Extract and Configure Apache Hive tar xvfz apache-hive-1.0.1.bin.tar.gz
- 5) Move Apache Hive from Local directory to Home directory
- 6) Set CLASSPATH in bashrc
Export HIVE-HOME = /home/apache-hive Export PATH = \$PATH:\$HIVE-HOME/bin
- 7) Configuring hive-default.xml by adding My SQL Server Credentials
<property>
<name>javax.jdo.option.ConnectionURL</name>
<value> jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true
</value>
</property>
<property>
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.jdbc.Driver</value>
</property>
<property>
<name>javax.jdo.option.ConnectionUserName</name>
<value>hadoop</value>
</property>
</property>

```
<name>javax.jdo.option.ConnectionPassword</name>
<value>hadoop</value>
</property>
Copying mysql-java-connector.jar to hive/lib directory.
```

RESULT AND ANALYSIS



```

administrator@ubuntu: ~
d yet. Please use TIMESTAMP instead
hive> create table log_data(l_date string,l_time string,s_sitename string,s_computername string,l_uri string,uri_query string,ip_address string,user_agent string,status1 int,status2 int,s_bytes int,c_bytes int,time_taken int);
OK
Time taken: 0.331 seconds
hive> show tables;
OK
log_data
Time taken: 0.074 seconds, Fetched: 1 row(s)
hive> desc log_data;
OK
l_date          string          None
l_time          string          None
s_sitename      string          None
s_computername  string          None
l_uri           string          None
uri_query       string          None
ip_address      string          None
user_agent      string          None
status1         int             None
status2         int             None
s_bytes         int             None
c_bytes         int             None

```

LAB ASSIGNMENT:

1. Analyze twitter data using Apache Hive.

VIVA VOCE QUESTIONS:

1. Explain the Built-in Functions in Hive?
2. Describe the various Hive Data types?

FURTHER PROBING QUESTIONS

1.Operating System:

What operating system are you using for the installation (e.g., Linux, Windows, macOS)? Have you considered any specific distribution or version requirements for your OS?

2.Hadoop Version Compatibility:

Are you installing Hive on a Hadoop cluster? If so, what version of Hadoop is your cluster running? Have you verified that the Hive version you plan to install is compatible with your Hadoop version?

3.Installation Mode:

Are you planning to install Hive in local (standalone) mode or in a distributed mode as part of a Hadoop cluster? If it's a distributed installation, do you have Hadoop HDFS and YARN configured and running?

4.Java Installation:

Have you installed Java on the target system? What is the Java version you're using? Have you set the JAVA_HOME environment variable appropriately?

Lab-12 HIVE OPERATIONS

Introduction

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.

Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive. It is used by different companies. For example, Amazon uses it in Amazon Elastic MapReduce.

Hive is a data warehousing and SQL-like query language for Hadoop, designed to facilitate the processing and analysis of large-scale datasets. It provides a high-level abstraction over the Hadoop Distributed File System (HDFS) and allows users to query data using SQL-like syntax, making it accessible to those familiar with relational databases. Here's an overview of some key Hive operations and concepts:

Database and Table Creation:

Hive allows you to create databases and tables to organize your data. Databases act as namespaces, and tables define the structure of your data. You can create tables with specified columns, data types, and storage properties.

Data Ingestion:

You can load data into Hive tables from various sources, including HDFS, local files, and external databases. Hive provides the LOAD DATA command to facilitate data ingestion.

Querying Data:

Hive supports SQL-like queries to retrieve and analyze data stored in tables. You can use the SELECT statement to perform queries, apply filters, and aggregate data.

Data Transformation:

You can perform data transformations within Hive by applying functions and operations to the columns in your tables. This includes calculations, string manipulations, and date/time functions.

Joins and Aggregations:

Hive supports various join operations (e.g., INNER JOIN, LEFT JOIN, RIGHT JOIN) to combine data from multiple tables. Additionally, you can perform aggregations using functions like SUM, AVG, and GROUP BY.

Views:

Hive allows you to create views, which are virtual tables generated from SQL queries. Views simplify complex queries and can provide restricted access to data.

Partitions and Buckets:

For improved query performance, you can partition tables based on one or more columns. Partitions allow for faster data retrieval when filtering by the partition key. Additionally, you can use bucketing to further optimize queries.

User-Defined Functions (UDFs):

Hive supports custom functions called User-Defined Functions (UDFs). You can write UDFs in various programming languages (e.g., Java, Python) to perform specialized operations on your data.

Indexing:

To speed up data retrieval, Hive supports indexing. You can create indexes on tables, which allow for faster lookups on indexed columns.

Security and Access Control:

Hive provides security features for data access control. You can set up role-based access control (RBAC) and integrate with external authentication systems for user management.

Objective

Use Hive to create, alter, and drop databases, tables, views, functions, and indexes.

Pre Lab Preparation

Resources:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

Equipement Needed,Software required

Big data and business analytics laboratory (AITB16) lab manual and work sheets,VMWARE,HADOOP

Background:

The Hive Query Language (HiveQL) is a query language for Hive to process and analyze structured data in a Metastore. This chapter explains how to use the SELECT statement with WHERE clause. SELECT statement is used to retrieve the data from a table. WHERE clause works similar to a condition. It filters the data using the condition and gives you a finite result. The built-in operators and functions generate an expression, which fulfils the condition.

Pre lab Questions

1. How many types of joins are there in Pig Latin with an examples?
2. Write the Hive command to create a table with four columns: First name, last name, age, and income?

Post Lab Preparation

The following of the steps are considered for Use Hive to create, alter, and drop databases, tables, views, functions, and indexes

Procedure:

SYNTAX for HIVE Database Operations DATABASE Creation

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

Drop Database Statement

DROP DATABASE StatementDROP (DATABASE/SCHEMA) [IF EXISTS]

database-name [RESTRICT|CASCADE];

Creating and Dropping Table in HIVE

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db-name.]

table-name

(col – namedata – type[COMMENTcol – comment],...)

COMMENTtable – comment

ROWFORMATrow – format

STOREDASfile – format

Loading Data into table log-data Syntax:

```
LOAD DATA LOCAL INPATH '<path>/u.data' OVERWRITE INTO TABLE  
u-data;
```

Alter Table in HIVE

Syntax

```
ALTER TABLE name RENAME TO new-name
```

```
ALTER TABLE name ADD COLUMNS (col-spec[, col-spec ...]) ALTER TABLE name DROP [COL-  
UMN] column-name
```

```
ALTER TABLE name CHANGE column-name new-name new-type ALTER TABLE name REPLACE  
COLUMNS (col-spec[col-spec ...])
```

 Creating and Dropping View

Dropping View Syntax:

```
DROP VIEW view-name
```

Functions in HIVE

String Functions:- round(), ceil(), substr(), upper(), reg-exp() etc Date and Time Functions:- year(), month(), day(), to-date() etc Aggregate Functions :- sum(), min(), max(), count(), avg() etc

```
CREATE VIEW [IF NOT EXISTS] view-name [(column-name [COMMENT column-comment], ...) ]  
[COMMENT table-comment] AS SELECT
```

...

INDEXES

```
CREATE INDEX index-name ON TABLE base-table-name (col-name, ...) AS 'index.handler.class.name'
```

WITHDEFERREDREBUILD

IDXPROPERTIES(property – name = property – value, ...)

INTABLEindex – table – name

PARTITIONEDBY(col – name, ...)

[

ROWFORMAT...

STORED AS ...

| STORED BY ...

]

*LOCATION*hdfs – path

TBLPROPERTIES(...)

Creating Index

CREATE INDEX index-ip ON TABLE log-data(ip-address) AS

'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED REBUILD;

Altering and Inserting Index

ALTER INDEX index-ip-address ON log-data REBUILD;

Storing Index Data in Metastore

SET

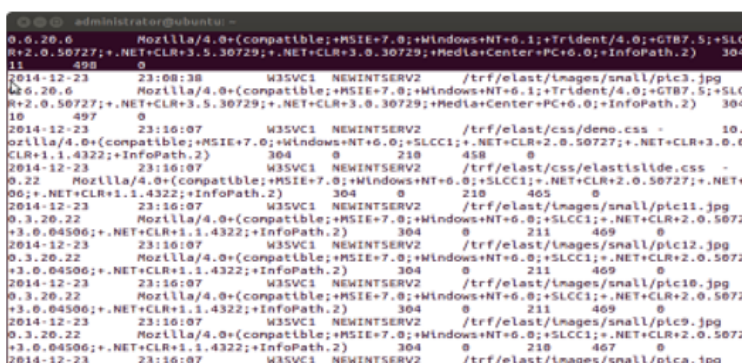
hive.index.compact.file=/home/administrator/Desktop/big/metastore-db/tmp/index-ipadd ress-result;

SET

hive.input.format=org.apache.hadoop.hive.ql.index.compact.HiveCompactIndexInputFor mat;

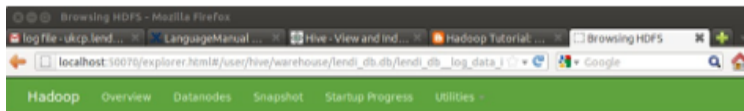
Dropping Index DROP INDEX INDEX-NAME on TABLE-NAME;

RESULT AND ANALYSIS



0.0.20.0	Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.1;+Trident/4.0;+GTB7.5;+SLC	R+2.0.50727;+.NET+CLR+3.5.30729;+.NET+CLR+3.0.30729;+Media+Center+PC+6.0;+InfoPath.2)	304	11	498	0
2014-12-23	23:08:38	W3SVC1 NEWINTSERV2	/trf/elastic/images/small/pic3.jpg			
0.0.20.0	Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.1;+Trident/4.0;+GTB7.5;+SLC	R+2.0.50727;+.NET+CLR+3.5.30729;+.NET+CLR+3.0.30729;+Media+Center+PC+6.0;+InfoPath.2)	304	10	497	0
2014-12-23	23:10:07	W3SVC1 NEWINTSERV2	/trf/elastic/css/demo.css	-	10.	
0.0.20.0	Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.50727;+.NET+CLR+3.0.0	CLR+1.1.4322;+InfoPath.2)	304	0	210	458
2014-12-23	23:10:07	W3SVC1 NEWINTSERV2	/trf/elastic/css/elasticslide.css	-	0.22	
0.0.20.0	Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.50727;+.NET+	06;+.NET+CLR+1.1.4322;+InfoPath.2)	304	0	210	465
2014-12-23	23:10:07	W3SVC1 NEWINTSERV2	/trf/elastic/images/small/pic11.jpg			
0.0.20.0	Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072	+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2)	304	0	211	469
2014-12-23	23:10:07	W3SVC1 NEWINTSERV2	/trf/elastic/images/small/pic12.jpg			
0.0.20.0	Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072	+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2)	304	0	211	469
2014-12-23	23:10:07	W3SVC1 NEWINTSERV2	/trf/elastic/images/small/pic9.jpg			
0.0.20.0	Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072	+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2)	304	0	210	467
2014-12-23	23:10:07	W3SVC1 NEWINTSERV2	/trf/elastic/images/small/pica.jpg			

```
administrator@ubuntu: ~$ hiveshell
hive> select * from index_ip;
FAILED: SemanticException [Error 10001]: Line 1:14 Table not found 'index_ip'
hive> INSERT OVERWRITE DIRECTORY '/home/administrator/Desktop/hive_data/index_test_result' SELECT '
bucketname' , '_offsets' FROM lendi_db.lendi_db_log_data_index_ip__ where ip_address='141.0.11.19
9';
Total MapReduce jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1476764326039_0014, Tracking URL = http://ubuntu.ubuntu-domain:8088/proxy/applica
tion_1476764326039_0014/
Kill Command = /home/administrator/hadoop-2.7.1/bin/hadoop job -kill job_1476764326039_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2016-10-18 02:16:23,240 Stage-1 map = 0%, reduce = 0%
2016-10-18 02:16:27,406 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
2016-10-18 02:16:28,442 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
2016-10-18 02:16:29,472 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
MapReduce Total cumulative CPU time: 1 seconds 320 nsec
Ended Job = job_1476764326039_0014
Stage-3 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Stage-4 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/tmp/hive-administrator/hive_2016-10-18_02-16-17_425_5894975364
0454830/-ext-10000
Moving data to: /home/administrator/Desktop/hive_data/index_test_result
```



Browse Directory

/user/hive/warehouse/lendi_db/lendi_db_log_data_index_ip_...

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	administrator	supergroup	106.78 KB	Tue 18 Oct 2016 02:09:07 AM EDT	1	128 MB	000000_0

LAB ASSIGNMENT:

1. Analyze stock data using Apache Hive.

VIVA VOCE QUESTIONS:

1. Write a shell command in Hive to list all the files in the current directory?
2. List the collection types provided by Hive for the purpose a start-up company want to use Hive for storing its data.

FURTHER PROBING QUESTIONS

- 1.Data Sources:

What are the primary data sources that you are ingesting into Hive? (e.g., HDFS, external databases, streaming data) How frequently is new data being added or updated in these sources?

- 2.Data Formats and Schemas:

What data formats are you working with (e.g., Parquet, ORC, Avro, CSV)? Have you defined and enforced data schemas for your Hive tables? Are there any data serialization or deserialization challenges?

- 3.Data Ingestion Process:

What tools or methods are you using for data ingestion into Hive? Are there any data preprocessing or cleansing steps involved before loading data into Hive tables?

- 4.Query Performance:

How critical is query performance in your use case? Are there specific queries or reporting requirements that demand optimization? What techniques do you employ to monitor and enhance query performance?