Session 1: Introduction to Big Data

Introduction to Big Data

1. **What is Big Data?**
   Big Data refers to *bohot saari data* jo itni massive hoti hai ki traditional tools ya software isse handle nahi kar sakte. Yeh data kaafi **large** hota hai aur **complex** bhi, jaise social media posts, online transactions, aur sensors ka data.

2. **Big Deal about Big Data?**
   Big Data itna important kyu hai?
   Kyunki **volume, velocity, aur variety** ke saath, yeh data se **valuable insights** nikalne ka kaam karta hai. Companies isse trends predict karte hain aur better decisions lete hain.

3. **Big Data Sources:**

   - **Social Media:** Facebook, Twitter, Instagram posts.
   - **Sensors and IoT Devices:** Temperature sensors, fitness trackers, etc.
   - **Business Transactions:** Online shopping, bank transactions.
   - **Healthcare:** Patient records aur medical devices.

4. **Industries Using Big Data:**

   - **E-commerce:** Amazon aur Flipkart recommendations system.
   - **Healthcare:** Disease predictions using patient history.
   - **Finance:** Fraud detection in banking.
   - **Transportation:** Ola/Uber route optimization.

5. **Big Data Challenges:**

   - **Storage:** Data itni badi hoti hai ki store karna mushkil hota hai.
   - **Processing:** Fast aur accurate process karna challenging hai.
   - **Security:** Sensitive data ka misuse hone ka risk hota hai.
   - **Integration:** Alag-alag sources se data ko combine karna tough hai.

Big Data Technologies and Hadoop

1. **Solution to Big Data Problems:**
   Big Data problems solve karne ke liye humein chahiye **distributed systems**, jisme data multiple servers pe store aur process hota hai. Hadoop ek powerful tool hai jo yeh kaam

efficiently karta hai.

2. **Various Big Data Technologies:**
    - **Hadoop:** Distributed storage aur processing ke liye.
    - **Spark:** Real-time data analysis.
    - **Kafka:** Data streaming ke liye.
    - **NoSQL Databases:** MongoDB aur Cassandra.
3. **Big Data/Hadoop Platforms:**
   Platforms jaha aap Big Data process karte ho, jaise:
    - **Cloudera, Hortonworks, MapR.**
4. **Hadoop Distributions and Vendors:**
   Hadoop ka original version **Apache** banata hai, but customized versions market mein vendors dete hain like:
    - **Cloudera Hadoop, AWS EMR (Elastic MapReduce), Azure HDInsight.**
5. **Big Data Suites:**
   End-to-end tools jo Big Data ka analysis easy banate hain, jaise:
    - **Microsoft Azure Data Suite, Google BigQuery.**

Introduction to Hadoop

1. **A Brief History of Hadoop:**
   Hadoop ka concept inspired tha Google ke research paper se jo **MapReduce** pe tha.
   Initially, **Doug Cutting aur Mike Cafarella** ne Hadoop develop kiya tha. Fun fact: Hadoop ka naam Doug ke son's toy elephant ke naam pe pada. 🐘

2. **Evolution of Hadoop:**
   Pehle data ko process karna centralized systems pe hota tha jo slow aur costly the.
   Hadoop distributed systems ka use karta hai jaha multiple machines ek saath kaam karti hain.

3. **Comparison with Other Systems:**
    - **Traditional RDBMS:** Structured data ke liye best, but Big Data handle nahi kar sakti.
    - **Hadoop:** Structured, semi-structured, aur unstructured data sab manage karta hai.

4. **Hadoop Releases:**
   Hadoop kaafi versions mein release hua hai, jaise:

   - Hadoop 1.x: Basic distributed system.
   - Hadoop 2.x: YARN introduced for better resource management.
   - Hadoop 3.x: Improved scalability and cloud support.

Session 2: Hadoop Architecture aur installation

Hadoop Architecture

1. **Hadoop Architecture:**
   Hadoop ek **distributed system** hai jo large data ko process karta hai. Is architecture mein data ko **clusters** mein tod diya jata hai, aur multiple machines par parallel process hota hai.
   Hadoop ke architecture ke main parts hain:

   - **HDFS (Hadoop Distributed File System):** Data ko distribute aur store karta hai across multiple servers.
   - **YARN (Yet Another Resource Negotiator):** Resource management aur job scheduling ke liye.
2. **Core Components of Hadoop:**

   - **HDFS:** Data ko chunks mein todkar multiple nodes par store karta hai.
   - **MapReduce:** Data ko process karne ke liye parallel programming model.
   - **YARN:** Computing resources allocate karta hai efficiently.

Getting Started: Hadoop Installation

1. **Setting up a Hadoop Cluster:**
   Hadoop cluster set karne ke liye 2 types of clusters hote hain:

   - **Single-Node Cluster:** Sirf ek machine par Hadoop run karna (mostly testing ke liye).
   - **Multi-Node Cluster:** Multiple machines mein Hadoop install karna (production ke liye).
2. **Logging Configuration:**
   Logs Hadoop ke saare events ko track karte hain, jaise **errors** aur **tasks.**

- **log4j.properties** file ko configure karke logging setup hoti hai.

3. **Cluster Specification:**
   Hadoop cluster ke liye humein kya-kya chahiye hota hai?

   - **Master Node:** Main server jo tasks ko control karta hai.
   - **Slave Nodes:** Jo actual data store aur process karte hain.
   - **Good Hardware:** SSDs, 16+ GB RAM, aur multicore CPUs recommended hain.

4. **Cluster Setup and Installation:**
   Steps:

   - **Install Java** (kyunki Hadoop Java-based hai).
   - Hadoop binaries download karo aur extract karo.
   - Hadoop configuration files edit karo, jaise **core-site.xml, hdfs-site.xml.**
   - Namenode aur Datanode start karo.

5. **Common Hadoop Shell Commands:**

   - **hadoop fs -ls /**: List files in HDFS.
   - **hadoop fs -put <local_file> <hdfs_path>:** Local file ko HDFS mein upload karna.
   - **hadoop fs -get <hdfs_file> <local_path>:** HDFS se local machine pe file lana.
   - **hdfs dfsadmin -report:** Cluster status check karna.

6. **Cluster Monitoring:**
   Hadoop ke performance ko monitor karne ke liye tools:

   - **Web UI:** Default Namenode UI (localhost:9870).
   - **Ganglia:** Resource monitoring ke liye.
   - **Nagios:** Error alerts ke liye.

7. **Single and Multi-Node Cluster Setup on Virtual Machine:**

   - Single Node: Virtual machine pe Hadoop install karke ek **pseudo-distributed mode** use karte hain.
   - Multi-Node: Multiple VMs ya cloud setup par Hadoop install karke saare nodes ko connect karte hain.

8. **Hadoop Configuration:**
   Important config files:

   - **core-site.xml:** Hadoop ke core settings.
   - **hdfs-site.xml:** HDFS configuration (block size, replication).
   - **mapred-site.xml:** MapReduce settings.
   - **yarn-site.xml:** YARN resource settings.

9. **Security in Hadoop:**
   Hadoop me sensitive data ke liye security zaroori hai:

   - **Kerberos:** Authentication ke liye.

- **Encryption:** Data-at-rest aur data-in-transit secure karne ke liye.
- **Access Control:** HDFS files ke liye permissions set karna.

10. **Administering Hadoop:**
    - Cluster health check karna.
    - Backup aur recovery processes.
    - Data nodes add ya remove karna.
11. **HDFS Monitoring & Maintenance:**
    - Regularly check **disk usage, block reports**, aur **replication factor**.
    - Corrupted blocks ko identify aur fix karna.
12. **Hadoop Benchmarks:**
    Performance measure karne ke liye Hadoop benchmarks use karte hain:
    - **Terasort:** Sorting speed check karna.
    - **TestDFSIO:** Read/write performance test.
13. **Hadoop in the Cloud:**
    Hadoop cloud par kaise use hota hai?
    - **AWS EMR (Elastic MapReduce):** Fully managed Hadoop.
    - **Azure HDInsight:** Microsoft ka Hadoop service.
    - **Google Cloud Dataproc:** Google ka Hadoop ecosystem.

Session 3: Hadoop Distributed File System (HDFS) and the **Lab Assignment**

Hadoop Distributed File System (HDFS)

1. **Distributed File System:**
   HDFS ek **distributed file system** hai jo large datasets ko manage karne ke liye design hua hai.
   - **Distributed** ka matlab hai data ko multiple servers (nodes) par store karna, taaki ek system pe load na aaye.
   - It ensures **fault tolerance**, **high throughput**, and **scalability.**
2. **What is HDFS?**
   - HDFS Hadoop ka **storage layer** hai jo huge data ko store karta hai.
   - It splits data into **blocks (default size: 128MB)** and unhe multiple nodes pe store karta hai for reliability.
3. **Major Goals of HDFS Design:**

- **Fault Tolerance:** Agar ek node fail ho jaye, toh bhi data accessible rahe.
- **Scalability:** Easily nodes add karke storage aur processing capacity badha sakte ho.
- **High Throughput:** Batch processing ke liye optimized.
- **Cost-Effective:** Commodity hardware ka use karta hai (no need for high-end machines).

4. **Where does HDFS Fit In?**
    - HDFS **data storage** ke liye hai.
    - **MapReduce** ya **YARN** HDFS pe stored data ko process karte hain.
    - It is the **foundation** for Hadoop ecosystem tools like Hive, Spark, and Pig.
5. **Core Components of HDFS:**
    - **Blocks:** Data chunks (128MB by default).
    - **Replication:** Har block ka multiple copies (default: 3) stored hota hai across different nodes.
    - **Metadata:** Data ke location aur status ka information store karta hai.
6. **Hadoop Server Roles:**
    - **Name Node:**
      Master server jo metadata maintain karta hai (like file names, block locations).
        - Stores **file system namespace.**
        - No actual data, only references.
    - **Secondary Name Node:**
      Backup purpose ke liye **edit logs** aur **fsimage** ko synchronize karta hai.
        - It is NOT a failover for Name Node.
    - **Data Node:**
      Slave nodes jo actual data blocks ko store karte hain.
        - Name Node ke instructions follow karte hain.

Lab Assignment: HDFS Commands

Let's run some common HDFS commands along with a one-line explanation:

1. **`hadoop fs -ls /`**
    - *Description:* HDFS root directory ke files aur folders ko list karta hai.
2. **`hadoop fs -mkdir /user/data`**
    - *Description:* HDFS me `/user/data` directory banata hai.
3. **`hadoop fs -put myfile.txt /user/data/`**

- *Description:* Local system se `myfile.txt` ko HDFS ke `/user/data/` folder me upload karta hai.

4. **`hadoop fs -get /user/data/myfile.txt /localdir/`**
    - *Description:* HDFS file ko local system ke `localdir` folder me download karta hai.
5. **`hadoop fs -rm /user/data/myfile.txt`**
    - *Description:* HDFS ke `/user/data/` directory se file ko delete karta hai.
6. **`hadoop fs -du -h /user/data`**
    - *Description:* `/user/data` directory ke size aur uske andar ke files ka disk usage show karta hai (in human-readable format).
7. **`hdfs dfsadmin -report`**
    - *Description:* Cluster status aur health ka detailed report deta hai.
8. **`hadoop fs -cat /user/data/myfile.txt`**
    - *Description:* HDFS file ka content terminal pe display karta hai.

Lab Exercise: Code Execution in HDFS

Step-by-step process for running code on HDFS:

Setup HDFS Directory:

hadoop fs -mkdir /user/gaurav/input

1.
    - Create an input folder in HDFS to upload your data.

Upload Data File:

hadoop fs -put localfile.txt /user/gaurav/input/

2.
    - Local file ko HDFS me upload karo.

Run Hadoop Job:
Execute your MapReduce or any Hadoop job. Example:

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

3.
    - This runs the **WordCount program** jo input folder ka data process karke output folder me result store karega.

Check Output:

```
hadoop fs -cat /user/gaurav/output/part-00000
```

4.
    - Processed result ko HDFS se read aur terminal me display karta hai.
5. **Monitor HDFS:**
    - Open **Name Node Web UI** at `localhost:9870` to check HDFS file structure and job status.

Session 4 & 5: HDFS Architecture aur HDFS Data Storage Process

---

HDFS Architecture

1. **HDFS Architecture:**
    - HDFS ek **distributed architecture** hai jo **large-scale data** ko manage karne ke liye design hua hai.
    - Main components:
        - **NameNode:** Metadata (file structure, block location) manage karta hai.
        - **DataNode:** Actual data ko store karta hai in the form of **blocks**.
        - **Secondary NameNode:** NameNode ke logs aur checkpoints ko sync karta hai.
2. **Scaling and Rebalancing:**
    - **Scaling:** Cluster me naye nodes add karke storage aur processing capacity badha sakte ho.
    - **Rebalancing:** Agar kuch nodes pe zyada data ho aur kuch nodes idle ho, toh data redistribute hota hai for efficiency.
        - Command: `hdfs balancer`
3. **Big Deal About HDFS:**

    - **Fault Tolerance:** Agar ek DataNode fail ho jaye, toh data replication ke karan accessible rahta hai.
    - **Cost-Effective:** Commodity hardware use karta hai.
    - **High Throughput:** Sequential data processing me efficient hai.
4. **Replication:**
    - Default replication factor: **3**
    - Har data block ke 3 copies banayi jaati hain aur alag-alag nodes par store hoti hain.
    - Ye ensure karta hai ki ek node fail hone par bhi data available rahe.
5. **Rack Awareness:**
    - HDFS me blocks ko **rack-aware algorithm** ke according place kiya jata hai.
    - Data replication ke liye different racks ka use hota hai, jisse **data loss risk** kam ho.
6. **Data Pipelining:**
    - Jab data HDFS me likha jata hai, toh ek **pipeline** banti hai:
        - Data client se ek DataNode me jata hai.
        - Wahan se dusre DataNode me copy hota hai (replication).
    - Ye process **sequential** hoti hai aur efficient hai.
7. **Node Failure Management:**
    - Agar koi DataNode fail ho, toh:
        - **NameNode** us node ka status "dead" mark kar deta hai.
        - Replication ensure karta hai ki data loss na ho.
        - Admin naye nodes add karke cluster ko balance kar sakta hai.
8. **HDFS NameNode High Availability (HA):**
    - HA architecture me **2 NameNodes** hote hain:
        - **Active NameNode:** Real-time metadata serve karta hai.
        - **Standby NameNode:** Backup mode me rehta hai aur ready hota hai failover ke liye.
    - Ye architecture failure se bachata hai aur system ko downtime-free rakhta hai.
9. **HDFS HA-Quorum Cluster Components:**
    - **JournalNodes:** Metadata ka log maintain karte hain jo Active aur Standby ke beech sync hota hai.
    - **Zookeeper:** Failover process ko manage karta hai.
10. **HDFS Federation Use Case:**
    - Federation multiple namespaces ko allow karta hai ek hi cluster me.
    - Alag-alag departments ya projects ke liye namespaces segregate kiye ja sakte hain.

11. **Kerberos (HDFS Security):**
    - HDFS me Kerberos authentication ensure karta hai ki authorized users hi data access kar paayein.
    - Data encryption aur secure access ke liye use hota hai.

HDFS Data Storage Process

1. **HDFS Data Storage Process:**
    - Data ko **blocks** me todkar alag-alag DataNodes par store kiya jata hai.
    - **Replication factor** ensure karta hai ki data redundant aur secure rahe.
2. **Anatomy of Writing a File in HDFS:**
    - Client request karta hai **NameNode** ko file write karne ke liye.
    - NameNode file ko blocks me todta hai aur unke DataNodes assign karta hai.
    - DataNode ek pipeline banata hai aur blocks replicate karte hain.
3. **Anatomy of Reading a File in HDFS:**
    - Client request karta hai **NameNode** se file ka location.
    - NameNode batata hai ki kaunsa block kis DataNode me hai.
    - Client directly DataNode se file blocks ko read karta hai.
4. **HDFS User and Admin Commands:**
    - **User Commands:**
        - `hadoop fs -ls /`: Files list karta hai.
        - `hadoop fs -put <file> <HDFS path>`: File upload karta hai.
        - `hadoop fs -get <HDFS path> <local path>`: File download karta hai.
    - **Admin Commands:**
        - `hdfs dfsadmin -report`: Cluster health report show karta hai.
        - `hdfs dfsadmin -safemode get`: Safe mode ka status check karta hai.
5. **HDFS Web Interface:**
    - URL: `http://<namenode-host>:9870`
    - Features:
        - Cluster summary.
        - File system browser.
        - DataNode health status.

Lab Assignment

1. **Run HDFS Code:**
    - Write aur read process ko hands-on practice karo.
    - Use the provided **WordCount program** ya kisi custom program ko HDFS ke saath execute karo.
2. **Differences: Regular FileSystem vs. HDFS:**
    - **File System:** Data ek hi machine me store hota hai.
    - **HDFS:** Data distributed hota hai across multiple nodes.
    - **File System:** Fault tolerance nahi hota.
    - **HDFS:** Data replication ke karan fault-tolerant hai.
    - **File System:** Scalability limited hoti hai.
    - **HDFS:** Large-scale storage aur processing ke liye scalable hai.
3. **Why is HDFS Fault-Tolerant?**
    - Data ke multiple copies (replication) store hoti hain.
    - Agar ek DataNode fail ho jaye, toh baaki copies se data access hota hai.
    - **Rack Awareness algorithm** ensure karta hai ki data alag-alag racks pe stored ho.

Session 6 & 7: MapReduce Framework and the Lab Assignment

Getting in Touch with the MapReduce Framework

1. **Hadoop MapReduce Paradigm:**
    - MapReduce ek **data processing model** hai jo **parallel** aur **distributed** systems par large-scale data process karta hai.
    - **Two main tasks:**
        - **Map:** Input data ko key-value pairs me todta hai.
        - **Reduce:** Key-value pairs ko aggregate ya summarize karta hai (e.g., total sum, average, etc.).
    - It is inspired by **divide-and-conquer** approach.

---

2. **Stages of MapReduce:**
   MapReduce job ke main stages:

- **Input Split:**
  Input data ko smaller splits me todta hai jo mappers process karenge.
- **Mapping:**
  Input splits ko key-value pairs me convert karta hai.
- **Shuffling and Sorting:**
  - Shuffling: Similar keys ko ek saath laata hai across all mappers.
  - Sorting: Keys ko order me arrange karta hai.
- **Reducing:**
  Reduce phase similar keys ke values ko process karke final output generate karta hai.
- **Output:**
  Final output file ko HDFS me store karta hai.

---

3. **Map and Reduce Tasks:**
   - **Map Task:**
     - Input split ko process karta hai aur key-value pairs generate karta hai.
     - Example:
       - Input: `Hello World`

Map Output:
(Hello, 1)

(World, 1)

     -
   - **Reduce Task:**
     - Mapper ke output ko summarize karta hai.
     - Example:

Input:
(Hello, [1, 1])

(World, [1, 1])

     -

Reduce Output:
(Hello, 2)

(World, 2)

     -

---

4. **MapReduce Execution Framework:**
    - MapReduce framework **YARN (Yet Another Resource Negotiator)** par run karta hai:
        - **ResourceManager:** Resources allocate karta hai.
        - **NodeManager:** Individual nodes ke tasks ko monitor karta hai.
        - **JobTracker:** Job ka status track karta hai.
        - **TaskTracker:** Map aur Reduce tasks execute karta hai.

---

5. **Anatomy of a MapReduce Job Run:**
   Ek MapReduce job execute karne ke steps:
    1. **Client submits a job** to the Hadoop cluster.
    2. **ResourceManager** job ko process karne ke liye resources assign karta hai.
    3. **Input data split hota hai** aur Map tasks par distribute hota hai.
    4. Mapper output ko **shuffling and sorting** ke liye pass karta hai.
    5. Reduce tasks aggregated data ko finalize karte hain aur HDFS me store karte hain.

Lab Assignment

1. **Execute the Train Data Example:**

    - Steps to execute the **train data example:**

Prepare your input data: Example:
Train1, Mumbai, Delhi

Train2, Pune, Chennai

Train3, Bangalore, Kolkata

        -
        - Write a **Mapper program**:
            - Read the train data and output key-value pairs like `(source, destination)`.
        - Write a **Reducer program**:
            - Aggregate the data based on destination counts.

Example Command to Run the Job:

hadoop jar train-example.jar TrainExample /user/gaurav/input /user/gaurav/output

-
    - Input folder me train data file upload karo.
    - Output result HDFS ke output folder me milega.

2. **Execute Using Chained Methods:**
    - Chained methods allow you to run multiple mappers or reducers in a sequence.
    - Example:
        - Mapper 1: Extract source and destination.
        - Reducer 1: Count destinations.
        - Mapper 2: Filter specific routes.
        - Reducer 2: Generate the final output.

Command for Chained Methods:

```
hadoop jar train-example.jar ChainedExample /user/gaurav/input /user/gaurav/output
```

-

Extra Insights

- **Difference between MapReduce and Other Frameworks:**
    - MapReduce is batch-oriented; tools like **Spark** are real-time.
- **Why is MapReduce important?**
    - Parallel processing allows huge datasets to be processed efficiently.
    - Works well with HDFS for scalable data storage and computation.

Session 8, 9 & 10: YARN (Yet Another Resource Negotiator)

YARN Overview

- **What is YARN?**
    - YARN is Hadoop ka **Resource Management Framework**, jo Hadoop cluster ke resources ko manage karta hai aur applications ko efficiently execute karne me help karta hai.
    - Hadoop 2 ke baad se YARN ne Hadoop cluster ko **multi-purpose** banaya, jisme **MapReduce** ke alawa bhi alag-alag data processing engines ko run karna possible hai (e.g., Spark, Hive).

YARN Architecture

1. **YARN ke Core Components:**
    - **ResourceManager (RM):**
        - Cluster ke resources ko manage karta hai.
        - Applications ke execution ke liye **containers** allocate karta hai.
    - **NodeManager (NM):**
        - Individual nodes ke tasks aur resources ko monitor karta hai.
        - Resources ka health status ResourceManager ko report karta hai.
    - **ApplicationMaster (AM):**
        - Ek specific application ke liye execution flow ko handle karta hai.
        - Tasks ko ResourceManager aur NodeManager ke saath coordinate karta hai.
    - **Container:**
        - YARN ka ek **resource unit** hai jo CPU, memory, aur disk ko represent karta hai.

YARN Resource Management

- **Resource Allocation Process:**

    - ResourceManager nodes se **heartbeats** ke through resource availability ka status leta hai.
    - Jab ek job submit hota hai, RM check karta hai ki required resources available hain ya nahi.
    - Containers allocate hote hain tasks ke execution ke liye.

- **Fair Usage:**
    - Multiple applications ke beech resources **fairly distribute** kiye jaate hain.
    - Jahan kisi application ka task complete ho, wahan se free resources dusri applications ko assign kiye jaate hain.

Hadoop Schedulers

Schedulers ka kaam hai **resource allocation ko manage karna.**

1. **FIFO Scheduler:**
    - **First In, First Out** principle follow karta hai.
    - Jo job pehle submit hota hai, usse pehle execute karne ka chance milta hai.
    - Simple hai, lekin resource optimization ke liye ideal nahi.
2. **Fair Scheduler:**
    - Resources ko **equally divide** karta hai active jobs ke beech.
    - Agar ek job khatam ho jaye, toh uske resources dusre jobs ko assign hote hain.
3. **Capacity Scheduler:**
    - Cluster ko **logical queues** me todta hai.
    - Har queue ke paas ek fixed capacity hoti hai, aur applications sirf apni queue ke andar ke resources use karte hain.

Upgrading Cluster from Hadoop 1 to Hadoop 2

1. **Why Upgrade?**
    - Hadoop 1 me MapReduce framework tightly coupled tha (processing aur resource management same layer pe hota tha).
    - Hadoop 2 me YARN introduced hua, jo **resource management aur processing framework** ko separate karta hai.
2. **Steps for Upgrade:**
    - **Backup existing cluster data.**
    - New version of Hadoop (Hadoop 2) install karo.
    - Configuration files update karo (e.g., `core-site.xml`, `hdfs-site.xml`, `yarn-site.xml`).
    - Old logs aur metadata ko migrate karo.

- Testing ke baad production cluster ko launch karo.

MapReduce Job Workflow on YARN

1. **Job Submission:**
    - Client **ResourceManager** ko job submit karta hai.
2. **ApplicationMaster Launch:**
    - ResourceManager ek **ApplicationMaster** start karta hai jo job ka execution manage karega.
3. **Task Scheduling:**
    - ApplicationMaster resources ke liye **ResourceManager** ke saath communicate karta hai.
    - Jaise hi containers allocate hote hain, tasks launch hote hain.
4. **Task Execution:**
    - Tasks **NodeManager** par run hote hain.
    - NodeManager task ka health aur progress report karta hai.
5. **Job Completion:**
    - ApplicationMaster job complete hone ke baad ResourceManager ko notify karta hai.

Migration from MRv1 to MRv2 on YARN

1. **MRv1 vs. MRv2:**
    - **MRv1:**
        - Resource management aur task execution dono ek hi framework (JobTracker) ke through hota tha.
        - Scalability issues the.
    - **MRv2 (YARN):**
        - ResourceManager aur ApplicationMaster separate components hain.
        - Highly scalable aur flexible framework.
2. **Configuration Changes in Files:**
    - Update the following files:

- **core-site.xml:**
  Add `fs.defaultFS` property.
- **yarn-site.xml:**
  Configure `ResourceManager` aur `NodeManager` settings.
- **mapred-site.xml:**
  Set `mapreduce.framework.name` as `yarn`.

Example Configuration Changes:

core-site.xml:

```
 <property>
    <name>fs.defaultFS</name>
    <value>hdfs://<namenode-host>:9000</value>
</property>
```

1.

yarn-site.xml:

```
 <property>
    <name>yarn.resourcemanager.hostname</name>
    <value><resourcemanager-hostname></value>
</property>
```

2.

mapred-site.xml:

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

3.

Session 11 & 12: Security in Hadoop aur Lab Assignment

Security in Hadoop

1. HDFS Security Model

- Hadoop Distributed File System (HDFS) me security ka main focus hai:
    - **Authentication:**
      Verify karna ki user kaun hai.
    - **Authorization:**
      Ensure karna ki user ko specific files ya resources access karne ki permission hai.
    - **Data Privacy and Integrity:**
      Data encrypt hota hai aur unauthorized access ko block karta hai.
- **Core Components of HDFS Security:**
    - **Kerberos Authentication:**
        - Hadoop default me **Kerberos protocol** use karta hai authentication ke liye.
        - Kerberos ek **ticket-based system** hai jo secure authentication provide karta hai.
    - **Access Control Lists (ACLs):**
        - Fine-grained permissions set karne ke liye ACLs ka use hota hai.
        - Example: Specific users ya groups ke liye file permissions define karna.
    - **Encryption:**
        - Data-in-transit (network) aur data-at-rest (HDFS me store) dono encrypt hote hain.

---

2. LDAP and Hadoop

- **LDAP (Lightweight Directory Access Protocol):**
    - Ek protocol jo centralized authentication aur directory services provide karta hai.
    - Example: Ek single LDAP server multiple applications (e.g., Hadoop, Linux, etc.) ke liye user credentials manage karta hai.
- **LDAP Support in Hadoop:**
    - Hadoop **external authentication services** ke liye LDAP ke saath integrate ho sakta hai.

- Use case: Ek centralized LDAP server ke users Hadoop ke resources (HDFS, YARN, etc.) access kar sakte hain bina alag-alag credentials ke.
- **Benefits:**
  - Centralized user management.
  - Secure and scalable authentication.

Lab Assignment

1. Configure LDAP in Linux

Step 1: Install OpenLDAP Server:

sudo apt update

sudo apt install slapd ldap-utils

- During installation, admin password set karna hoga.

- **Step 2: Configure LDAP Server:**

Configure slapd package:
sudo dpkg-reconfigure slapd

  -
    - Provide domain details (e.g., `example.com`).
    - Set up LDAP database backend (use MDB).
- **Step 3: Add LDAP Users and Groups:**

Create an LDIF file for user addition:
nano add_user.ldif

Add the following content:
dn: uid=john,ou=users,dc=example,dc=com

objectClass: inetOrgPerson

sn: Doe

givenName: John

uid: john

mail: john@example.com

userPassword: password123

Apply the LDIF file:
ldapadd -x -D "cn=admin,dc=example,dc=com" -W -f add_user.ldif

-

Step 4: Test LDAP Configuration:
Verify LDAP user with:

ldapsearch -x -LLL -H ldap:/// -b "dc=example,dc=com" uid=john

-

---

2. Integrate LDAP with Hadoop

- **Step 1: Enable LDAP Authentication in Hadoop:**
  Modify the following files:

core-site.xml:

```
<property>
  <name>hadoop.security.authentication</name>
  <value>ldap</value>
</property>
<property>
  <name>hadoop.security.authorization</name>
  <value>true</value>
</property>
```

hdfs-site.xml:

```
<property>
```

```
    <name>dfs.namenode.ldap.url</name>
    <value>ldap://ldap.example.com</value>
</property>
<property>
    <name>dfs.namenode.ldap.base</name>
    <value>dc=example,dc=com</value>
</property>
<property>
    <name>dfs.namenode.ldap.bind.user</name>
    <value>cn=admin,dc=example,dc=com</value>
</property>
<property>
    <name>dfs.namenode.ldap.bind.password</name>
    <value>your-admin-password</value>
</property>
```

Step 2: Restart Hadoop Services:
Restart NameNode and DataNode services for the changes to take effect.

```
start-dfs.sh
start-yarn.sh
```

Step 3: Test LDAP Authentication with HDFS:
Try accessing HDFS with an LDAP user:

```
hdfs dfs -ls /user/john
```

Key Differences and Insights

- **Why Use LDAP with Hadoop?**
    - LDAP centralized authentication aur user management provide karta hai, jo Hadoop ke security model ke saath seamlessly integrate hota hai.
    - It simplifies managing large clusters with multiple users.
- **Testing Checklist:**
    - Ensure LDAP server correctly configured hai.
    - Hadoop ke core-site.xml aur hdfs-site.xml files properly update honi chahiye.
    - Permissions aur ACLs ko validate karo.

Session 13, 14 & 15: Hadoop Cluster Planning aur Maintenance

Hadoop Cluster Planning

1. Choosing Hardware and Operating Systems

- **Hardware Selection:**
  Hadoop cluster ko scale karne ke liye hardware ka selection bohot important hai. Kuch key factors hain:
    - **CPU (Processing Power):**
      Hadoop jobs ke liye powerful CPUs chahiye hote hain, taaki MapReduce tasks efficiently run ho sakein.
    - **RAM (Memory):**
      MapReduce ke liye large datasets ko process karte waqt, sufficient RAM zaruri hoti hai, especially for **in-memory processing**.
    - **Storage (Disks):**
      Hadoop me data **distributed** hota hai, toh storage ki requirement bhi bohot high hoti hai. **High I/O throughput** aur **large capacity** storage devices chahiye hote hain.
    - **Network:**
      High-speed networking (e.g., **1 Gbps or more**) zaruri hai, taaki data nodes ke beech fast communication ho sake.
- **Operating System Comparison:**
  Operating systems me choice generally **Linux-based systems** pe hoti hai, jaise **Ubuntu, CentOS**, or **Red Hat**.
  Features to consider:

- **Kernel Tuning:**
  Hadoop ke performance ko optimize karne ke liye OS kernel ko tune karna padta hai (e.g., virtual memory parameters ko adjust karna).
- **Disk Swapping:**
  Hadoop cluster ke liye disk swapping avoid karna chahiye, kyunki excessive disk swapping performance ko impact karta hai.
- **Filesystem Support:**
  Hadoop HDFS ko run karne ke liye OS ka **ext3** ya **ext4** filesystem ideally hona chahiye.

2. Identifying Hardware and Cluster Size

- **Scenario-based Cluster Planning:**
  Cluster ka size workload aur expected data volume pe depend karta hai. Agar large-scale data processing hai, toh large clusters (more nodes) chahiye hote hain. Agar smaller-scale processing hai, toh chote clusters (fewer nodes) use kar sakte hain.

  - **Small Data Processing:** 10-20 nodes.
  - **Large Data Processing:** 100s of nodes, large storage systems.
- **Hardware Selection for Specific Scenarios:**
  Example:

  - **Real-time Data Processing:** Fast CPUs aur low-latency network connections chahiye.
  - **Batch Processing (large datasets):** Large storage capacity aur high disk throughput chahiye.

3. Identifying Eco-system Components

- **Eco-system Components:**
  Hadoop ke **core components** ke alawa, additional ecosystem components bhi use hote hain:
  - **Hive** (Data warehousing tool)
  - **Pig** (Data processing language)
  - **HBase** (NoSQL database)
  - **ZooKeeper** (Coordination service for distributed apps)
    Identify karna hoga ki kaunsa component use karna hai, based on workload aur data type.

Cluster Maintenance

1. Managing Hadoop Processes

- **Manual Management:**

Hadoop processes ko manually start/stop kiya ja sakta hai using commands:
start-dfs.sh # Start HDFS services

start-yarn.sh # Start YARN services

stop-dfs.sh # Stop HDFS services

stop-yarn.sh # Stop YARN services

    -
    - **Monitoring:**
      Cluster ka status check karna bohot important hai. Hadoop ke logs aur monitoring tools (like **Ganglia** or **Nagios**) se process health monitor karte hain.
- **Scripted Management:**

    - Automated scripts se processes ko manage karna efficient hota hai.
      Example: Create a script to restart all services:

#!/bin/bash

start-dfs.sh

start-yarn.sh

echo "Hadoop Services Started"

-

2. HDFS Maintenance Tasks

- **Adding a DataNode:**

    - New DataNode add karte waqt, **HDFS configuration** files ko update karna padta hai.

    - `hdfs-site.xml` file me `dfs.namenode.replication` aur other parameters adjust karna padta hai.

Example:

hdfs dfsadmin -addDatanode <new-datanode-ip>

- **Decommissioning a DataNode:**
  - DataNode ko **decommission** karte waqt, `dfs.hosts.exclude` file me DataNode ka IP add karna padta hai.
  - Phir, DataNode ko cluster se remove karte hain without affecting data availability.

Example:
hdfs dfsadmin -decommissionDatanode <datanode-ip>

3. MapReduce Maintenance Tasks

- **Adding or Removing TaskTrackers:**
  - Hadoop 2.x me TaskTracker ka concept **deprecated** ho gaya hai, aur **NodeManager** ka role increase ho gaya hai.
  - Aap **NodeManager** ko configure kar sakte hain jisme TaskTracker ka function hota hai.
- **Killing Jobs/Tasks:**
  - Agar koi job ya task stuck ho gaya hai, toh usse manually **kill** karna padta hai:

yarn application -kill <application-id>

4. Backup & Recovery

- **HDFS Backup:**
  - Regular backups banana essential hai. HDFS me data **replicated** hota hai, lekin backup ek extra layer of protection provide karta hai.

Example:
hdfs dfs -cp /user/data /user/data_backup

- **Recovery Process:**

- Agar cluster crash ho jata hai, toh **NameNode logs** aur **DataNode logs** se recovery ki ja sakti hai.
- Hadoop ke **JournalNode** aur **Zookeeper** use kar ke high availability (HA) setup karte hain.

Network Topology and Design

- **Network Usage in Hadoop:**
  - Network topology design Hadoop ki **distributed nature** ko dhyan me rakhe kar hoti hai.
  - **Rack Awareness** bhi zaruri hota hai, taaki data loss ya failure ko minimize kiya ja sake.
- Example:
  - Aapke paas **multiple racks** hone chahiye jisme DataNodes distributed ho, taki agar ek rack fail ho, toh data redundant racks pe available ho.

Key Points for Cluster Planning & Maintenance:

1. **Resource Management:**
   Cluster size aur hardware ka selection workload aur data processing needs pe depend karta hai.

2. **Ongoing Maintenance:**
   Hadoop cluster ko maintain karte waqt, periodic checks, health monitoring aur log analysis zaruri hota hai.

3. **High Availability & Fault Tolerance:**
   Data replication aur proper network design se fault tolerance ensure karte hain. Backup aur recovery processes ka implementation bhi zaruri hai.
