**Assignment-3**

**Q.1) Explain Anatomy of Map Reduce Job Run failures?**

|  |
| --- |
| How Hadoop runs a MapReduce job |

* **Anatomy of a MapReduce Job Run process:**

1. **Client:** Submitting the MapReduce job.
2. **Yarn node manager:** In a cluster, it monitors and launches the compute containers on machines.
3. **Yarn resource manager:** Handles the allocation of computing resources coordination on the cluster.
4. **MapReduce application master**Facilitates the tasks running the MapReduce work.
5. **Distributed Filesystem:**Shares job files with other entities.

* **In the MapReduce framework,** job run failures can occur due to various reasons. The anatomy of MapReduce job run failures typically involves several stages and components. Let's go through them:
* **Client-side failures:** These failures occur on the client-side before the job is submitted to the MapReduce framework. They can be caused by issues such as incorrect input parameters, insufficient permissions, or network connectivity problems.
* **Job submission failures:** After the client submits the job to the MapReduce framework, failures can occur during the submission process. This may be due to resource allocation issues, conflicts with other running jobs, or communication failures between the client and the framework.
* **Input data-related failures:** These failures can happen if there are issues with the input data provided for the MapReduce job. It could include missing or corrupted input files, incorrect file formats, or incompatible data types.
* **Map task failures:** Map tasks are responsible for processing input data and generating intermediate key-value pairs. Failures can occur during the execution of map tasks due to software bugs, hardware failures, or errors in the user-defined map function.
* **Reduce task failures:** Reduce tasks process the intermediate data generated by map tasks and produce the final output. Failures during reduce tasks can be caused by similar reasons as map task failures, such as software bugs, hardware issues, or errors in the user-defined reduce function.
* **Shuffle and sort failures:** The shuffle and sort phase involves transferring the intermediate data from the map tasks to the reduce tasks. Failures in this phase can be caused by network errors, disk failures, or inconsistencies in the intermediate data.
* **Output-related failures:** These failures occur when writing the final output of the MapReduce job. They can be caused by issues with the output directory, disk space limitations, or write permissions.
* **Resource-related failures:** MapReduce jobs require resources like memory, CPU, and disk space. Failures can occur if there are resource shortages or if the allocated resources are insufficient to complete the job.

**Q.2) What is meant by Job scheduling? Explain its types?**

* **Job scheduling in MapReduce** refers to the process of assigning and managing the execution of MapReduce jobs in a distributed computing environment. It involves determining the order and allocation of resources for executing individual MapReduce jobs based on various scheduling policies and constraints. The goal of job scheduling is to optimize the utilization of resources, minimize job completion time, and ensure efficient job execution.

|  |
| --- |
| Hadoop schedulers |

* **Types of Job Scheduling in MapReduce:**

1. **FIFO (First-In, First-Out):** In FIFO scheduling, jobs are executed in the order they arrive. The first job submitted to the system is assigned resources and executed first, followed by subsequent jobs in the order of submission. FIFO scheduling is simple and easy to implement but may not provide optimal performance. It can lead to job delays and resource underutilization if long-running jobs are submitted first or if there is a significant variation in job sizes.
2. **Fair Scheduler:** The Fair Scheduler aims to provide fair sharing of resources among different jobs or users in a multi-tenant MapReduce cluster. It dynamically allocates resources based on the demands of the jobs and their configured fairness settings. The Fair Scheduler divides the cluster resources into pools or queues, and each pool is assigned a certain percentage of resources. Within each pool, jobs are scheduled using a round-robin or fair sharing mechanism, ensuring that each job gets a fair share of the resources. The Fair Scheduler adapts to changing job priorities and resource demands, promoting fairness and preventing any single job from monopolizing the cluster resources.
3. **Capacity Scheduler:** The Capacity Scheduler is designed to support workload isolation and resource sharing in a multi-tenant MapReduce cluster. It allows the cluster resources to be divided into multiple capacities or queues, each with its own defined capacity. The Capacity Scheduler guarantees a minimum capacity for each queue, ensuring that jobs in higher-priority queues always have access to a certain portion of the cluster's resources. Within each queue, jobs can be scheduled using various policies such as FIFO, fair sharing, or priority-based scheduling. The Capacity Scheduler provides fine-grained control over resource allocation, enabling organizations to prioritize and allocate resources based on job importance, user priorities, or service level agreements (SLAs).

**Q.3) Explain in short Shuffle & Sort processing using Map-Reduce process with example?**

|  |
| --- |
| Shuffling & Sorting in Hadoop MapReduce |

* **Shuffling** is the process by which it transfers**mappers** intermediate output to the **reducer.** Reducer gets 1 or more keys and associated values on the basis of reducers. The intermediated key – value generated by mapper is sorted automatically by key. In Sort phase merging and sorting of map output takes place. Shuffling and Sorting in Hadoop occurs simultaneously.

### Shuffling in MapReduce: The process of transferring data from the mappers to reducers is shuffling.  It is also the process by which the system performs the sort. Then it transfers the map output to the reducer as input. This is the reason shuffle phase is necessary for the reducers. Otherwise, they would not have any input (or input from every mapper). Since shuffling can start even before the map phase has finished. So this saves some time and completes the tasks in lesser time.

### Sorting in MapReduce: MapReduce Framework automatically sort the keys generated by the mapper. Thus, before starting of reducer, all intermediate key-value pairs get sorted by key and not by value. It does not sort values passed to each reducer. They can be in any order. Sorting in a MapReduce job helps reducer to easily distinguish when a new reduce task should start. This saves time for the reducer. Reducer in MapReduce starts a new reduce task when the next key in the sorted input data is different than the previous. Each reduce task takes key value pairs as input and generates key-value pair as output. The important thing to note is that shuffling and sorting in Hadoop MapReduce are will not take place at all if you specify zero reducers (setNumReduceTasks(0)). If reducer is zero, then the MapReduce job stops at the map phase. And the map phase does not include any kind of sorting (even the map phase is faster).

**Q.4) Define:**

1. **Task Execution:** Task Execution refers to the process of executing individual tasks within a distributed computing system or framework, such as MapReduce. In the context of MapReduce, a task typically represents a unit of work that needs to be performed on a subset of the input data.
2. **Data Node:** In the context of the Hadoop Distributed File System (HDFS), a Data Node is a component that stores and manages data in a distributed manner. Data Nodes are responsible for storing and retrieving data blocks on the physical storage devices (local disks) of the machines in a Hadoop cluster. They communicate with the Name Node to report the status of the stored data blocks and handle read and write requests from clients or other components of the Hadoop ecosystem.
3. **Name Node:** The Name Node is a critical component in the Hadoop Distributed File System (HDFS). It serves as the central metadata repository that manages the file system namespace and tracks the locations of data blocks in a Hadoop cluster. The Name Node maintains information about the hierarchy of files and directories, as well as the mapping of data blocks to Data Nodes.
4. **Task Tracker:** Task Tracker is a component in the Hadoop MapReduce framework that runs on each worker node in a Hadoop cluster. Task Trackers are responsible for executing tasks assigned by the Job Tracker (central coordinator of MapReduce jobs) and reporting their progress. They manage and monitor the execution of Map and Reduce tasks on the worker nodes, handling task scheduling, task execution, and status updates.
5. **Job Tracker:** The Job Tracker is a central component in the Hadoop MapReduce framework. It acts as the master or coordinator of MapReduce jobs within a Hadoop cluster. The Job Tracker is responsible for receiving job submissions, scheduling tasks to Task Trackers (worker nodes), and monitoring the overall progress and status of the jobs.
6. **Meta Store:** The Meta Store is a component in the Apache Hive data warehouse infrastructure. It serves as a metadata repository for storing and managing metadata about Hive tables, partitions, columns, storage properties, and other related information. The Meta Store enables Hive to provide a high-level, SQL-like interface for querying and analyzing data stored in various data formats, such as HDFS, HBase, or Amazon S3.
7. **Secondary Name Node:** The Secondary Name Node is a component in the Hadoop Distributed File System (HDFS). Its main purpose is to assist the Name Node in managing and maintaining the metadata of the HDFS. The Secondary Name Node periodically merges the edits log, which contains all the changes to the file system namespace, with the fsimage, which is a snapshot of the file system metadata.

**Q.5) List features of Map-reduce & explain in short any 2.**

* **Features of MapReduce**

1. **Scalability:** Apache Hadoop is a highly scalable framework. This is because of its ability to store and distribute huge data across plenty of servers. All these servers were inexpensive and can operate in parallel. We can easily scale the storage and computation power by adding servers to the cluster. Hadoop MapReduce programming enables organizations to run applications from large sets of nodes which could involve the use of thousands of terabytes of data. Hadoop MapReduce programming enables business organizations to run applications from large sets of nodes. This can use thousands of terabytes of data.
2. **Flexibility:** MapReduce programming enables companies to access new sources of data. It enables companies to operate on different types of data. It allows enterprises to access structured as well as unstructured data, and derive significant value by gaining insights from the multiple sources of data. Additionally, the MapReduce framework also provides support for the multiple languages and data from sources ranging from email, social media, to clickstream. The MapReduce processes data in simple key-value pairs thus supports data type including meta-data, images, and large files. Hence, MapReduce is flexible to deal with data rather than traditional DBMS.
3. **Security and Authentication:** The MapReduce programming model uses HBase and HDFS security platform that allows access only to the authenticated users to operate on the data. Thus, it protects unauthorized access to system data and enhances system security.
4. **Cost-effective solution:** Hadoop’s scalable architecture with the MapReduce programming framework allows the storage and processing of large data sets in a very affordable manner.
5. **Fast:** Hadoop uses a distributed storage method called as a Hadoop Distributed File System that basically implements a mapping system for locating data in a cluster. The tools that are used for data processing, such as MapReduce programming, are generally located on the very same servers that allow for the faster processing of data. So, Even if we are dealing with large volumes of unstructured data, Hadoop MapReduce just takes minutes to process terabytes of data. It can process petabytes of data in just an hour.
6. **Simple model of programming:** Amongst the various features of Hadoop MapReduce, one of the most important features is that it is based on a simple programming model. Basically, this allows programmers to develop the MapReduce programs which can handle tasks easily and efficiently. The MapReduce programs can be written in Java, which is not very hard to pick up and is also used widely. So, anyone can easily learn and write MapReduce programs and meet their data processing needs.
7. **Parallel Programming:** One of the major aspects of the working of MapReduce programming is its parallel processing. It divides the tasks in a manner that allows their execution in parallel.  
   The parallel processing allows multiple processors to execute these divided tasks. So the entire program is run in less time.
8. **Availability and resilient nature:** Whenever the data is sent to an individual node, the same set of data is forwarded to some other nodes in a cluster. So, if any particular node suffers from a failure, then there are always other copies present on other nodes that can still be accessed whenever needed. This assures high availability of data. One of the major features offered by Apache Hadoop is its fault tolerance. The Hadoop MapReduce framework has the ability to quickly recognizing faults that occur. It then applies a quick and automatic recovery solution. This feature makes it a game-changer in the world of big data processing.

**Q.6) Explain Map-Reduce types & formats.**

* **Map Input Types and Formats:** InputFormat describes the input-specification for execution of the Map-Reduce job. In MapReduce job execution, InputFormat is the first step. InputFormat describes how to split and read input files. InputFormat is responsible for splitting the input data file into records which is used for map-reduce operation.
* **TextInputFormat:** This is the default input format for MapReduce jobs. It treats each line of input as a separate record and provides the key-value pairs to the mapper. The key represents the byte offset of the line, and the value contains the content of the line.
* **KeyValueTextInputFormat:** This input format treats each line of input as a key-value pair, where the key and value are separated by a delimiter (such as a tab or comma). The key-value pairs are passed to the mapper for processing.
* **SequenceFileInputFormat:** This input format is used for reading binary sequence files in MapReduce. Sequence files store key-value pairs and can be compressed, making them efficient for storing large amounts of data.
* **Custom Input Formats:** MapReduce allows developers to define custom input formats by implementing the InputFormat interface. This enables reading data from various sources or in custom formats, such as XML, JSON, or database tables.
* **Map Output Types and Formats:** The outputFormat decides the way the output key-value pairs are written in the output files by RecordWriter. The OutputFormat and InputFormat functions are similar. OutputFormat instances are used to write to files on the local disk or in HDFS[.](https://techvidvan.com/tutorials/hdfs-introduction-tutorial/)
* TextOutputFormat: This is the default output format for MapReduce jobs. It writes the mapper's key-value pairs as plain text files. The key and value are converted to strings and separated by a delimiter (tab by default).
* **SequenceFileOutputFormat:** This output format writes the mapper's key-value pairs into a binary sequence file. It preserves the data types and allows for efficient storage and subsequent processing.
* **Custom Output Formats:** Similar to input formats, developers can define custom output formats by implementing the OutputFormat interface. This enables writing data to various output destinations or in custom formats, such as databases, NoSQL stores, or specialized file formats.

**Q.7) Give any three Typical use cases of Streaming Data?**

* **Streaming data is a continuous flow of data that is generated in real time and needs to be processed and analyzed immediately. Here are three typical use cases of streaming data:**

1. **Real-Time Analytics:** Streaming data is valuable for real-time analytics, where data is continuously collected, processed, and analyzed as it arrives. This allows businesses to gain immediate insights and make informed decisions based on the most up-to-date information. For example, an e-commerce company can analyze streaming data from website clickstreams to monitor user behavior, detect patterns, and provide personalized recommendations in real time**.**
2. **Fraud Detection:** Streaming data is often used for fraud detection in various industries such as banking, finance, and e-commerce. By analyzing real-time transaction data, streaming analytics platforms can detect fraudulent patterns and anomalies as they occur, allowing organizations to take immediate action to prevent or mitigate potential fraud. This could involve monitoring credit card transactions, online transactions, or any other financial activity that requires real-time fraud detection.
3. **Internet of Things (IoT) Monitoring:** With the proliferation of IoT devices, streaming data plays a crucial role in monitoring and managing the vast amount of data generated by these devices. IoT devices continuously produce data streams from sensors, machines, and other connected devices. Streaming data analytics enables real-time monitoring, predictive maintenance, and automated responses based on the data received. For instance, in a smart city, streaming data from various IoT devices can be analyzed to optimize traffic flow, detect environmental changes, and improve overall city operations.

**(PPT Give you more understanding than PDF)** The material for the PDF has been compiled from various sources such as books, tutorials (offline and online), lecture notes, several resources available on Internet. The information contained in this PDF is for general information and education purpose only. While we endeavor to keep the information up to date and correct, **we make no representation of any kind about the completeness and accuracy of the material.** The information shared through this PDF material should be used for educational purpose only.

**Thank you…**