Module - 03
Map Reduce

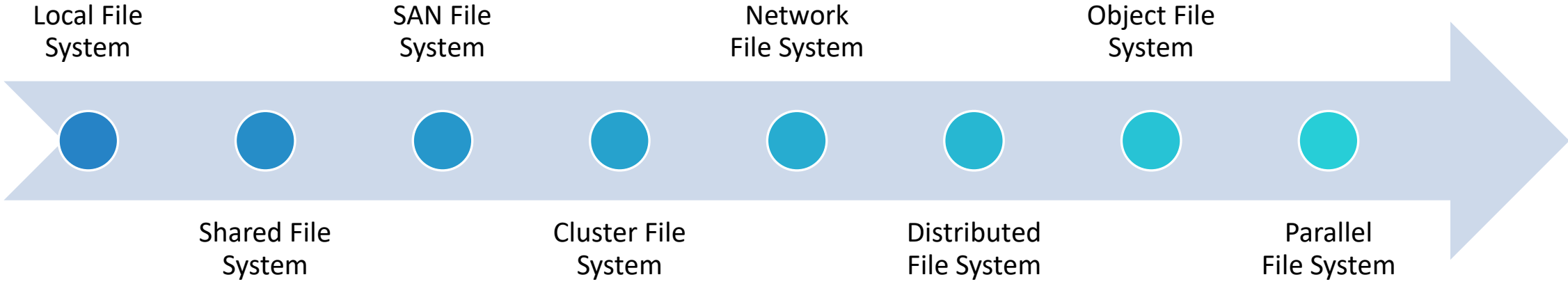
DISTRIBUTED FILE SYSTEM(DFS)

MAP REDUCE, ALGORITHMS USING MAP REDUCE

COMMUNICATION COST MODEL

GRAPH MODEL FOR MAP REDUCE PROBLEM.

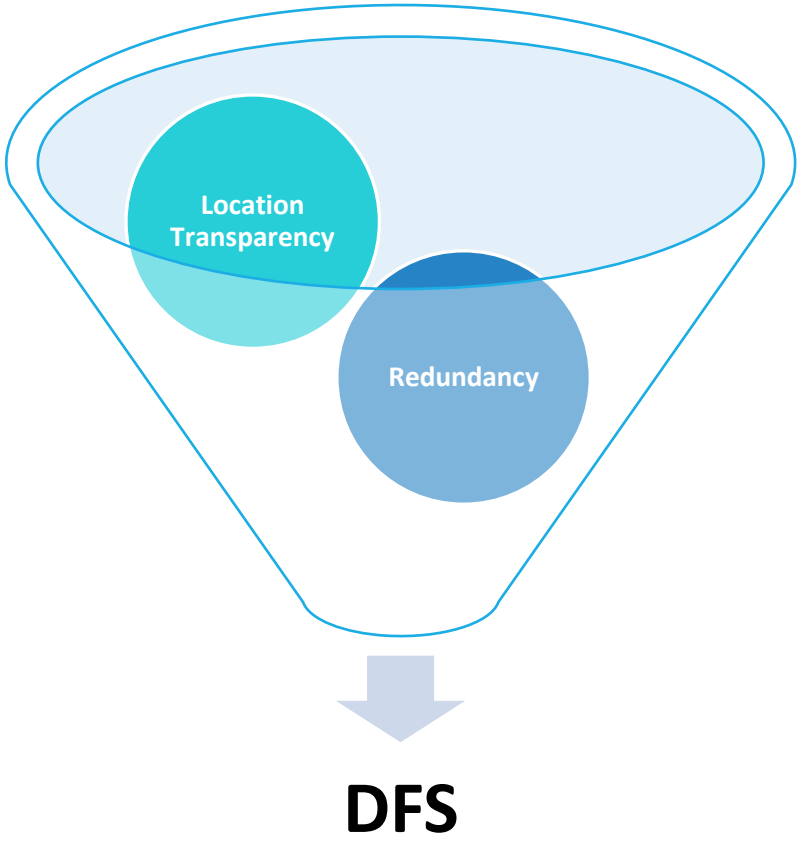
Distributed File System(DFS)



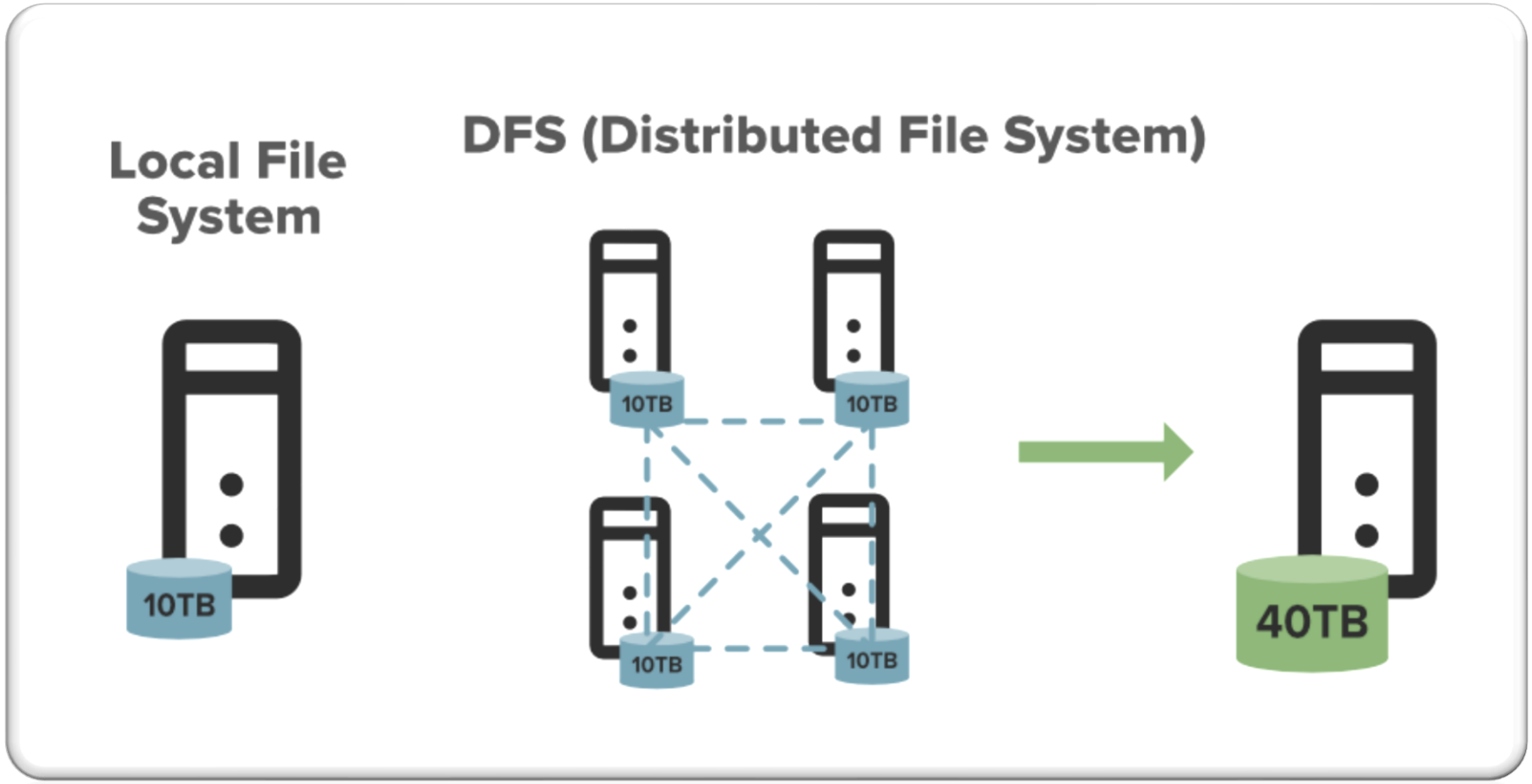
Distributed File System(DFS)

- It is a file system that is distributed on multiple file servers or multiple locations.
- A DFS is also called a client-server architecture based application.
- The main purpose of the DFS is to allows users of physically distributed systems to share their data and resources by using a Common File System.
- Location Transparency achieves through the namespace component.
- Redundancy is done through a file replication component.

DFS root



Difference between Local File System (LFS) and Distributed File System (DFS)



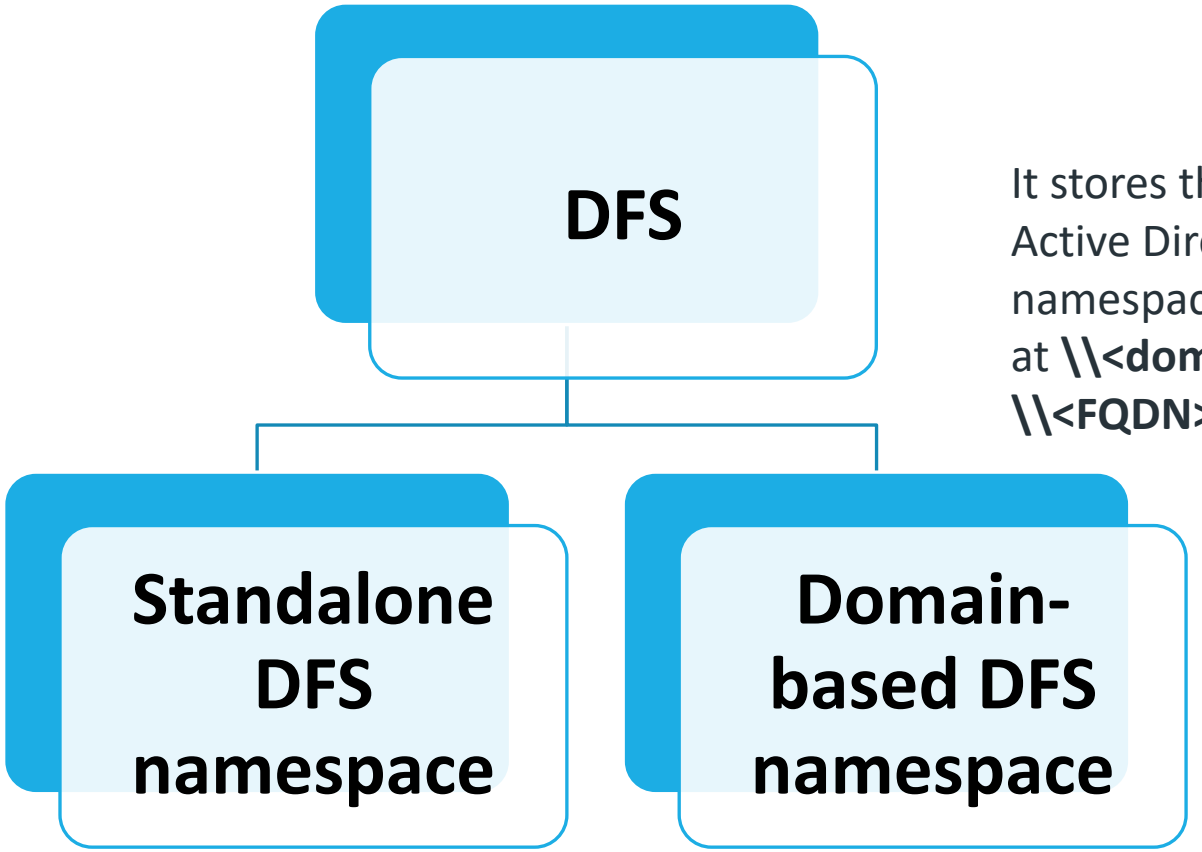
Difference between Local File System (LFS) and Distributed File System (DFS)

Local File System	Distributed File System
LFS **stores** data as a single block.	DFS divides data as multiple blocks and stores it into different DataNodes.
LFS uses Tree **format** to store Data.	DFS provides Master-Slave architecture for Data storage.
Data retrieval in LFS is slow.	Data retrieval in DFS is fast.
It is not **reliable** because LFS data does not **replicate** the Data files.	It is reliable because in DFS data blocks are replicated into different DataNodes.
LFS is **cheaper** because it does not needs extra memory for storing any data file.	DFS is expensive because it needs extra memory to replicate the same data blocks.

Difference between Local File System (LFS) and Distributed File System (DFS)

Local File System	Distributed File System
Files can be accessed directly in LFS.	Files can not be accessed directly in DFS because the actual location of data blocks are only known by NameNode.
LFS is not appropriate for analysis of very big file of data because it needs large time to process.	DFS is appropriate for analysis of big file of data because it needs less amount of time to process as compare to Local file system.
LFS is less complex than DFS.	DFS is more complex than LFS.

Working of Distributed File System



It stores the configuration of DFS in Active Directory, creating the DFS namespace root accessible at **\\<domainname>\<dfsroot>** or **\\<FQDN>\<dfsroot>**

It allows only for those DFS roots that exist on the local computer and are not using Active Directory.

Applications of Distributed File System(DFS)

- **Hadoop**
- NFS (Network File System)
- SMB (Server Message Block)
- NetWare
- CIFS (Common Internet File System)

Advantages of Distributed File System(DFS)

- It allows the users to access and store the data.
- It helps to improve the access time, network efficiency, and availability of files.
- It provides the transparency of data even if the server of disk files.
- It permits the data to be shared remotely.
- It helps to enhance the ability to change the amount of data and exchange data.

Disadvantages of Distributed File System(DFS)

- In a DFS, the database connection is complicated.
- In a DFS, database handling is also more complex than in a single-user system.
- If all nodes try to transfer data simultaneously, there is a chance that overloading will happen.
- There is a possibility that messages and data would be missed in the network while moving from one node to another.

How to setup Windows DFS Share in your environment?

Steps to setup DFS Share in the network are given below

- Enabling Server
- Configuring DFS Namespaces
- Configuring DFS Replication Group
- Configuring Software Repository

https://www.manageengine.com/products/desktop-central/how-to-setup-windows-dfs-share.html

Map Reduce

- MapReduce is a programming model or pattern within the Hadoop framework that is used to access big data stored in the Hadoop File System (HDFS).
- MapReduce facilitates concurrent processing by splitting petabytes of data into smaller chunks, and processing them in parallel on Hadoop commodity servers.
- MapReduce was once the only method through which the data stored in the HDFS could be retrieved, but that is no longer the case.
- Today, there are other query-based systems such as Hive and Pig that are used to retrieve data from the HDFS using SQL-like statements. However, these usually run along with jobs that are written using the MapReduce model.

Map Reduce - Example

- Receives a million requests every day to process payments
- Exceptions

invalid address

out of inventory

payment declined by a payment gateway

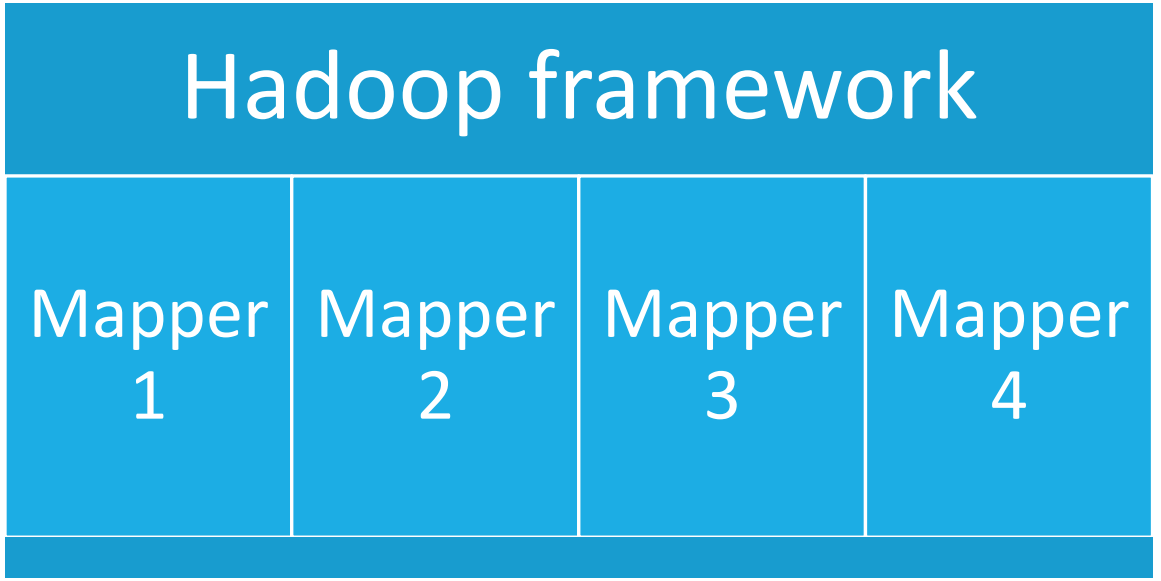
Map Reduce - Example

- A developer wants to analyze last four days' logs to understand which exception is thrown how many times.
- The objective is to isolate use cases that are most prone to errors.

Map Reduce - Example

Map

- The value input to the mapper is one record of the log file.
- The key could be a text string such as "file name + line number."
- The mapper, then, processes each record of the log file to produce key value pairs.



Map Reduce - Example

Output from the mappers

- **Mapper 1** -> <Exception A, 1>, <Exception B, 1>, <Exception A, 1>, <Exception C, 1>, <Exception A, 1>
- **Mapper 2** -> <Exception B, 1>, <Exception B, 1>, <Exception A, 1>, <Exception A, 1>
- **Mapper 3** -> <Exception A, 1>, <Exception C, 1>, <Exception A, 1>, <Exception B, 1>, <Exception A, 1>
- **Mapper 4** -> <Exception B, 1>, <Exception C, 1>, <Exception C, 1>, <Exception A, 1>

Map Reduce - Example

Combine

- **Combiner 1:** <Exception A, 3>, <Exception B, 1>, <Exception C, 1>
- **Combiner 2:** <Exception A, 2>, <Exception B, 2>
- **Combiner 3:** <Exception A, 3> ,<Exception B, 1> ,<Exception C, 1>
- **Combiner 4:** <Exception A, 1>, <Exception B, 1> ,<Exception C, 2>

Map Reduce - Example

Partition

Partitioner allocates the data from the combiners to the reducers.

- **input to the reducers**
- Reducer 1: <Exception A> {3,2,3,1}
- Reducer 2: <Exception B> {1,2,1,1}
- Reducer 3: <Exception C> {1,1,2}

If there were no combiners involved, the input to the reducers will be as below:
Reducer 1: <Exception A> {1,1,1,1,1,1,1,1,1}
Reducer 2: <Exception B> {1,1,1,1,1}
Reducer 3: <Exception C> {1,1,1,1}

Map Reduce - Example

Reduce

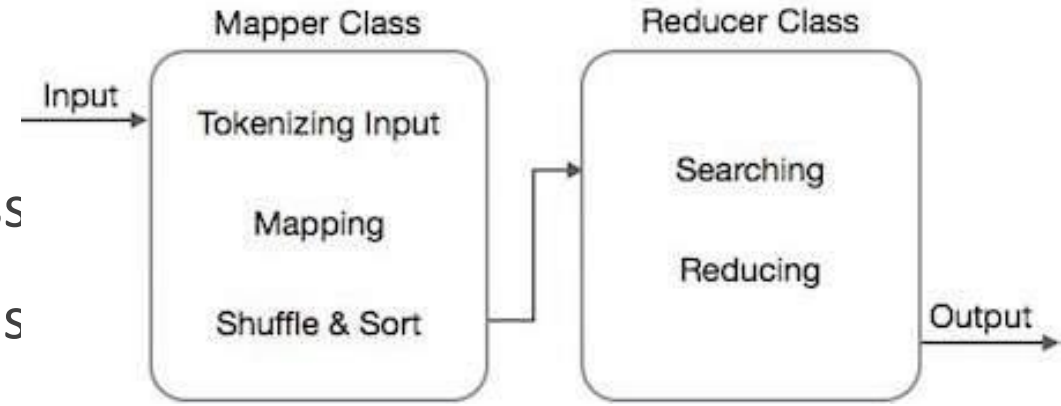
Each reducer just calculates the total count of the exceptions.

- Reducer 1: <Exception A, 9>
- Reducer 2: <Exception B, 5>
- Reducer 3: <Exception C, 4>

Refer: https://www.talend.com/resources/what-is-mapreduce/

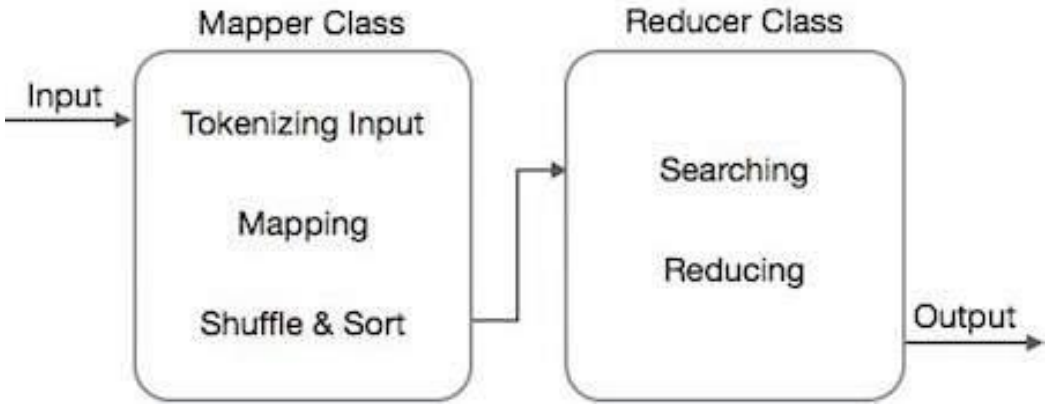
Map Reduce - Algorithm

- MapReduce algorithm contains two important tasks, namely Map and Reduce.
  - Map task is done by means of Mapper Class
  - Reduce task is done by means of Reducer Class
- Mapper class takes the input, tokenizes it, maps and sorts it.
- The output of Mapper class is used as input by Reducer class, which in turn searches matching pairs and reduces them.



Map Reduce - Algorithm

- MapReduce implements various mathematical algorithms to divide a task into small parts and assign them to multiple systems.
- MapReduce algorithm helps in sending the Map & Reduce tasks to appropriate servers in a cluster.
  - Sorting
  - Searching
  - Indexing
  - TF-IDF (Term Frequency – Inverse Document Frequency)



Map Reduce

Why should we use MapReduce Algorithm?

MapReduce is an application that is used for the processing of huge datasets. These datasets can be processed in parallel. It can process any kind of data like structured, unstructured or semi-structured.

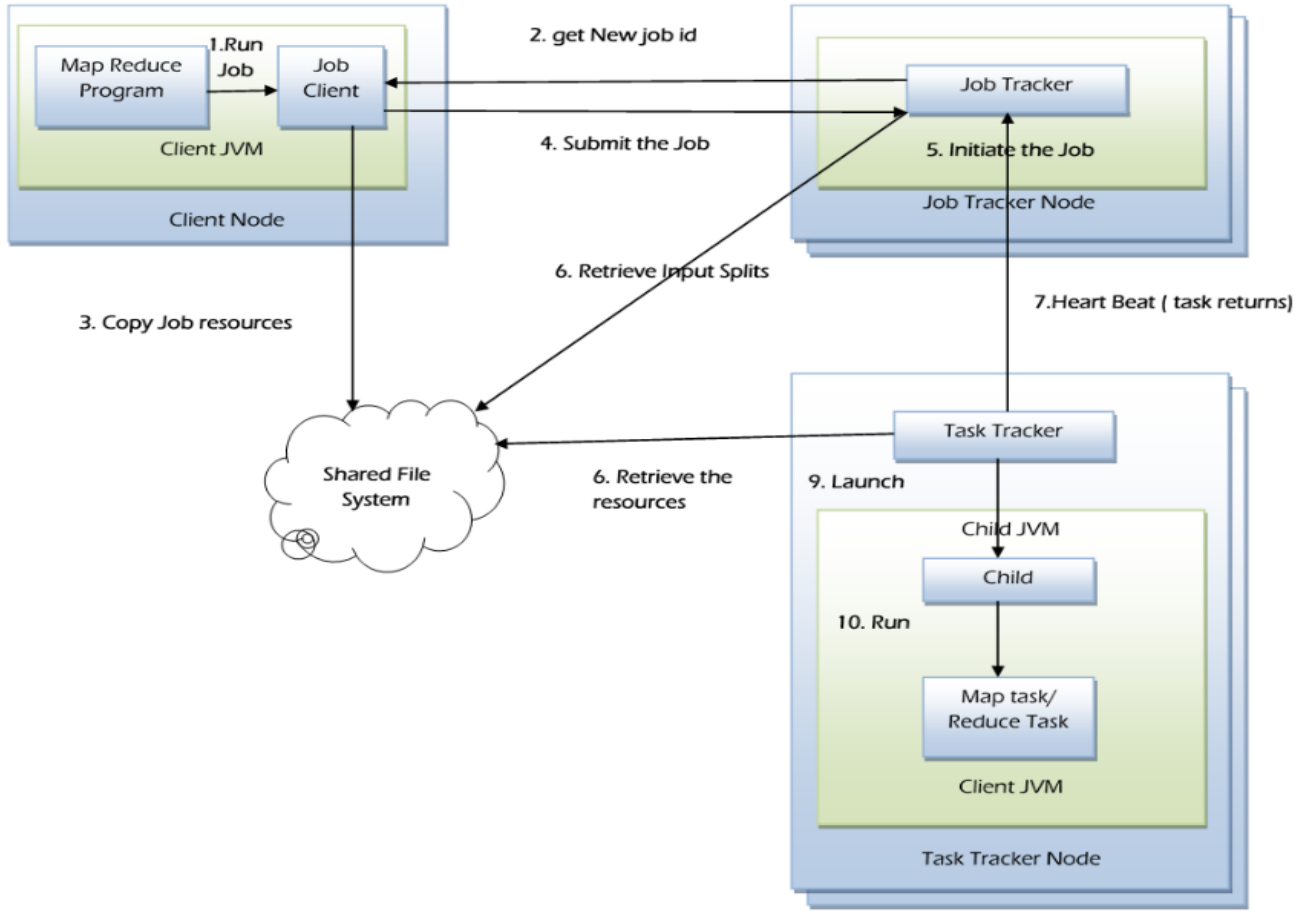
Why do we need the MapReduce Algorithm?

It allows programmers to run models over different data sets and uses advanced statistical techniques and machine learning techniques that help in predicting data. It also helps in predicting and recommending analysis.

Map Reduce

Key roles in Map Reduce Process

- Job Client
- Job Tracker
- Task Tracker
- Map Task
- Reduce Task



Map Reduce

Job Client

- Prepares a job for execution
- Validates the job configuration
- Generates the input splits
- Copies the job resources to a shared location
- Submits the job to the Job Tracker

Map Reduce

Job Tracker

- Scheduling jobs
- Dividing a job into map and reduce tasks
- Distributing map and reduce tasks among worker nodes
- Task failure recovery
- Tracking the job status

Map Reduce

Task Tracker

- Task Tracker runs on the associated nodes
- Fetches job resources locally
- Issues a child JVM on the node to execute map or reduce task
- Reports status to the Job Tracker

Map Reduce

Map Task

- Create input key-value pairs
- Applies the job-supplied map function to each KV pair
- Performs local sorting & aggregation of the results
- Runs the combiner if it is available
- Stores the results in the local file system

Map Reduce

Reduce Task

- It aggregates the results from the map phase into final results
- Fetches local copies from all the map worker nodes
- Performs sort phase to merge and copies into a single sorted array (K, value-list) pairs

References

https://docs.microsoft.com/en-us/windows/win32/stg/the-evolution-of-file-systems

https://docs.microsoft.com/en-us/windows/win32/dfs/distributed-file-system-dfs-functions

https://www.javatpoint.com/distributed-file-system

https://www.techopedia.com/definition/1825/distributed-file-system-dfs

https://www.geeksforgeeks.org/what-is-dfsdistributed-file-system/

https://www.snia.org/sites/default/education/tutorials/2012/fall/file/ThomasRivera_The_Evolution_of_File_Systems_10-1-2012.pdf

https://www.manageengine.com/products/desktop-central/how-to-setup-windows-dfs-share.html

https://www.tutorialscampus.com/map-reduce/algorithm.htm

https://www.talend.com/resources/what-is-mapreduce/

https://www.tutorialspoint.com/map_reduce/map_reduce_algorithm.htm#:~:text=MapReduce%20implements%20various%20mathematical%20algorithms,appropriate%20servers%20in%20a%20cluster

https://www.journaldev.com/8848/mapreduce-algorithm-example