Distributed Data Processing

In this lesson, we will discuss distributed data processing and the technologies used for it.

WE'LL COVER THE FOLLOWING

Alright!! Fellas, this lesson is all about distributed data processing. I'll talk about what it is? How different is it in comparison to a centralized data processing system? What are the architectures involved in it? And other similar topics.

So, let's get on with it.

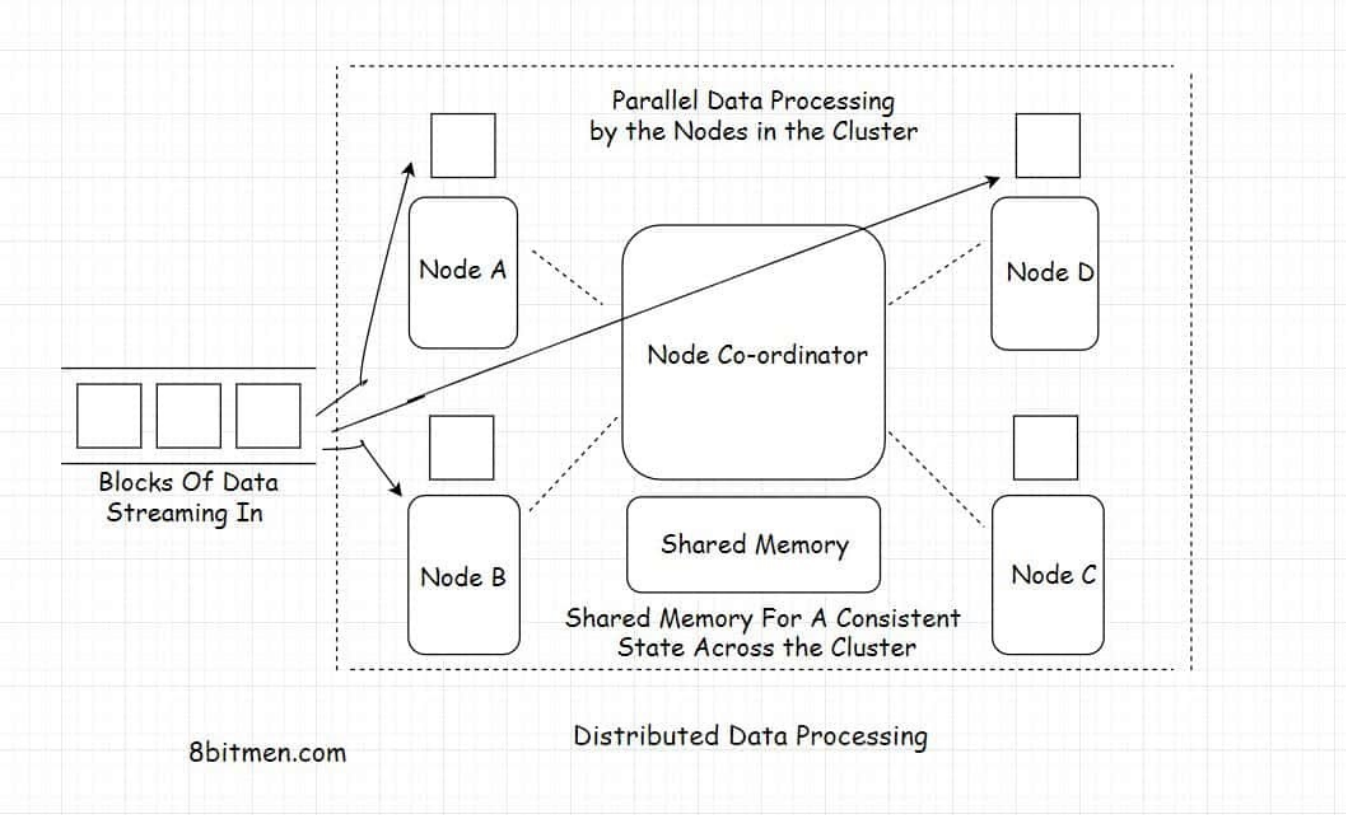
What Is Distributed Data Processing?

> *Distributed data processing* means diverging large amounts of data to several different nodes, running in a cluster, for parallel processing.

All the nodes execute the task allotted parallelly, working in conjunction with each other co-ordinated by a node-co-ordinator. *Apache Zookeeper* is a pretty popular, de-facto, node co-ordinator used in the industry.

Since the nodes are distributed and the tasks are executed parallelly, this makes the entire set-up pretty *scalable* & *highly available*. The workload can be scaled both horizontally & vertically. Data is made *redundant* & *replicated*

across the cluster to avoid any sort of data loss.



Processing data in a distributed environment helps accomplish the task in a significantly less amount of time as opposed to when running it on a centralized data processing system.

In a distributed system the tasks are shared by several nodes on the contrary in a centralized system the tasks are queued in a queue to be processed one by one.

Distributed Data Processing Technologies

Here are some of the popular technologies, I've listed, that are used in the industry for large scale data processing.

MapReduce – Apache Hadoop

MapReduce is a programming model written for managing distributed data processing across several different machines in a cluster, distributing tasks to several machines, running work in parallel, managing all the communication and data transfer within different parts of the system.

The *Map* part of the programming model involves sorting the data based on a parameter and the *Reduce* part involves summarizing the sorted data.

The most popular open-source implementation of the *MapReduce programming model* is *Apache Hadoop*. The framework is used by all big guns in the industry to manage massive amounts of data in their system. It is used by Twitter for running analytics. It is used by Facebook for storing big data.

Apache Spark

Apache Spark is an open-source cluster computing framework. It provides high performance for both batch & real-time in-stream processing. It can work with diverse data sources & facilitates parallel execution of work in a cluster.

Spark has a cluster manager and distributed data storage. The cluster manager facilitates communication between different nodes running together in a cluster whereas the distributed storage facilitates storage of big data. Spark seamlessly integrates with distributed data stores like *Cassandra, HDFS, MapReduce File System, Amazon S3* etc.

Apache Storm

Apache Storm is a distributed stream processing framework. In the industry, it is primarily used for processing massive amounts of streaming data. It has several different use cases such as real-time analytics, machine learning, distributed remote procedure calls etc.

Apache Kafka

Apache Kafka is an open-source distributed stream processing & messaging platform. It's written using *Java* & *Scala* & was developed by *LinkedIn*.

The storage layer of Kafka involves a distributed scalable pub/sub message queue. It helps read & write streams of data like a messaging system.

Kafka is used in the industry to develop real-time features such as notification platforms, managing streams of massive amounts of data, monitoring website activity & metrics, messaging, log aggregation.

Hadoop is preferred for *batch processing* of data whereas *Spark, Kafka* & *Storm* are preferred for processing real-time streaming data.

So, by now, I am sure you have a good idea of what data processing is. It's use-

cases in modern application development. The technologies involved etc.

Let’s have a look at a couple of architectures involved in the process. *Lambda* & *Kappa*.