The MapReduce framework is used by developers to process massive data on distributed platforms. The two most common cluster architectural types are as follows: racks that house 16–64 different servers and connect them all with fast switches, Multiple file parts may exist, and the master node is used to identify where each one is held. Files in a distributed file system are "chunked" before being duplicated across numerous servers.

Each segment of the file becomes the input for a different map operation once it has been separated into portions. The responsibility of applying the map function to each key-value pair in the input chunk for each map operation is assigned to a worker node for each map action.
The procedure creates temporary key-value pairs, which are stored on the local disks of the workers. We temporarily store each key-value pair in a disk container using the hash function h. This expedites the procedure and guarantees that the work is divided equally among the workers. Bucket I is equal to h(k)/(N2 Buckets) mod (key, value). A distinct worker is assigned to each bucket, and this worker uses the reduce function the user selected for that bucket. The first stages of two key processes are started throughout these three phases.

The data is sent between the mappers and reducers through the shuffle function. To expedite the process, this function can start working while the map phase is still in progress. The most expensive and time-consuming process is occasionally the shuffle. A list of items that share a key are arranged in ascending order using the sorting function. When reducers are not provided, this happens. The MapReduce process will end after the map step is finished.

To handle and analyze enormous amounts of data, Amazon Web Services (AWS) created a system called Amazon Elastic MapReduce, which is now simply known as Amazon Elastic MapReduce. Amazon offers EMR as a scalable and minimally-configured service as an alternative to the usual method of building cluster computing internally.

A Java-based programming framework called Amazon Elastic MapReduce (EMR), built on top of Apache Hadoop, makes it easier to process enormous data volumes in a distributed computing environment. Because it enables programmers to create applications that deal with massive amounts of unstructured data, MapReduce is a crucial part of the Hadoop software architecture. Both single computers and shared-processor clusters can run these applications. It is in responsible of indexing websites, and Google developed it in 2004 to replace the company's earlier indexing algorithms and heuristics.

Utilizing Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (ASS), Amazon Elastic MapReduce (EMR) distributes data for processing across a Hadoop cluster (S3). "Elastic" specifically refers to the EMR system's capacity for dynamic resizing. This capability allows system administrators to precisely adjust the amount of resources that are available to satisfy the system's requirements at any given time.

Amazon Elastic MapReduce is used for data analysis in a variety of applications, including log analysis, web indexing, data warehousing, machine learning (ML), financial analysis, scientific modeling, and bioinformatics. Workloads for Presto, Apache HBase, Apache Spark, and Apache Hive are also supported. The latter may communicate with the Hadoop Hive and Pig open-source data warehouse technologies. Workloads for Apache Spark are also supported. While Pig offers a high-level interface for scripting Hadoop-based MapReduce operations, Hive handles data queries and analysis. Pigs have a variety of traits.

Using MapReduce, we may efficiently query massive data sets while prepping the algorithm for horizontal growth. However, MapReduce is not always effective and has drawbacks in certain situations:

Unfortunately, the MapReduce architecture cannot handle all requests. The map's actions are completely independent of one another. This means that the processes cannot communicate with one another. Distributed systems require significantly more monitoring and management than centralized systems. As a result, it is critical to carefully assess if a collection of computers is genuinely necessary.