**UNIT-5, CLOUD COMPUTING**

**PROGRAMMING MODELS AND ADVANCES**

**Programming Models:**

1) MapReduce

2) Apache Spark

3) Tensor Flow

Intercloud Architecture

**Advances:**

1) Mobile Cloud Computing

2) Edge Computing

3) Fog Computing

--------------------------------------------------------------------------

**MAP REDUCE**

\* MapReduce is a programming model developed by Google.

\* Objective is to implement large scale search, and text processing on massively scalable web data stored using big table and GFS distributed file systems.

\* Designed for processing and generating large volumes of data via massively parallel computations, utilizing tens of thousands of processors at a time.

\* Fault Tolerant: Ensure progress of computation even if processors and networks fail.

\* Example: Hadoop: An open-source implementation of MapReduce (developed at Yahoo!), available on pre-packaged AMIs on Amazon EC2 cloud platform

*Parallel Computing*

*Different Models of Parallel Computing*

\* Nature and Evolution of multi-processing computer architecture

\* Shared-Memory Model: Assumes that any processor can access any memory location, unequal latency

\* Distributed Memory Model: Each processor can access only its own memory and communicates with other processors using message passing

*Parallel Computing*

\* Developed for computing intensive scientific tasks

\* Later found application in the database arena – shared memory, shared disk, shared nothing

*1) Shared Memory*

\* Suitable for servers with multiple CPUs

\* Memory address space is shared and managed by a symmetric multi-processing (SMP) operating system

\* SMP: schedules processes in parallel exploiting all the process

*2) Shared Nothing*

\* Cluster of independent servers each with its own disk space

\* Connected by a network

*3) Shared Disk*

\* Hybrid Architecture

\* Independent server cluster share storage through high-speed network storage viz NAS or SAN

\* Clusters are connected to storage via standard ethernet or faster Fibre channel or infiband connections

**Parallel Efficiency**

\* If a task takes time ‘T’ in uniprocessor system, it should take ‘T/p’ time if executed on ‘p’ processors.

\* Inefficiencies introduced in distributed computation due to:

a) Need for synchronization among processes

b) Overheads of message communication between processors

c) Imbalance in the distribution of work to the processors

\* Parallel Efficiency of an algorithm is defined as e = ?

*Scalable parallel implementation*

\* Parallel efficiency remains constant as the size of the data is increased along with a corresponding increase in processors

\* Parallel efficiency increase with the size of the data for a fixed number of processors

**MapReduce Model**

\* It is a parallel programming abstraction.

\* Used by many different parallel applications which carry out large scale computation involving thousands of processors.

\* Leverages a common underlying fault-tolerant implementation.

\* MapReduce has two phases: Map Operation, Reduce Operation.

\* A configurable number of M ‘mapper’ processors and ‘R’ reducer processors are assigned to work on the problem.

\* Computation is coordinated by a single master process.

*Map Phase:*

\* Each mapper reads approximately 1/M of the input from the global file system, using locations given by the master.

\* Map operation consists of transforming one set of key-value pairs to another: Map: (K1, V1) -> [K2, V2]

\* Each mapper writes computation results in one file per reducer.

\* Files are sorted by a key and stored to the local file system.

\* The master keeps track of the location of these files.

*Reduce Phase*:

\* The master informs the reducers where the partial computation have been stored on the local files of the respective mappers.

\* Reducers make remote procedure call to the mappers to fetch the files.

\* Each reducer groups the results of the map step using the same key and performs a function f on the list of values that correspond to these key values: reduce: (K2, [V2]) -> (K2, f[V2]).

\* Final results are written back to the GPS file system.

**MapReduce: Fault Tolerance**

*Heartbeat communication:*

\* Updates are exchanged regarding the status of the task assigned to workers.

\* Communication exists, but no progess: master duplicates those tasks and assigns to processors who have already completed.

\* If a mapper fails, the master reassigns key range designated to it to another working node for re-execution.

\* Re-execution is required as the partial computations are written into local files, rather than GFS file system.

\* If a reducer fails, only the remaining tasks are reassigned to another node, since the completed tasks are already written back into GFS.

--------------------------------------------------------------------------

**HADOOP**

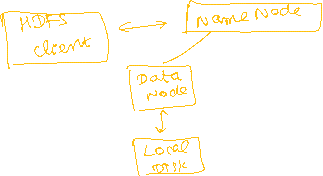
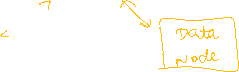
\* Hadoop is an open source framework that allows us to store and process large datasets in parallel and distributed manner.

\* Two main components: HDFS and MapReduce.

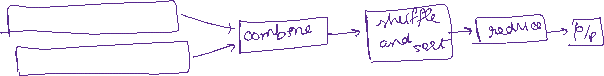
\* Hadoop Distributed File System [HDFS] is the primary data storage system used by Hadoop Applications.

\* MapReduce is the processing unit of Hadoop.

*Hadoop Distributed File System (HDFS)*



*MapReduce*



*Hadoop Eco System*

**APACHE SPARK**

\* Framework of Spark

\* Resilient Distributed Datasets (RDDs)

\* Applications: PageRank, GrapghX

\* Apache Stack is a big data analytics framework that was designed by University of California in 2012. Since then, it has gained a lot of attraction both in academia and industry.

\* It is another system for big data analytics.

**Need of Spark**

\* Is MapReduce not good enough? Simplifies batch processing on large commodity clusters.

\* MapReduce can be expensive for some applications such as iterative and interactive applications.

\* Lacks efficient data sharing.

\* Specialized frameworks did evolve for different programming models: Bulk synchronous processing (Pregel) and Iterative MapReduce (Hadoop)

**Solution: Resilient Distributed Datasets**

\* Immutable, partitioned collection of records

\* Built through coarse grained transformations (map, join, ...)

\* Can be cached for efficient reuse.

*Fault Recovery*

\* Lineage!

\* Log the coarse grained operation applied to a partitioned dataset

\* Simply recompute the lost partition in failure occurs

\* No cost if no failure

\* The RDDs track the graph of the transformations that built them (their lineage) to rebuild lost data

*What can we do with Spark?*

*RDD Operations:*

\* Transformations (eg filter, join, map, group by, ...)

\* Actions (eg count, print, …)

*Control:*

\* Partition: Spark gives control over how you can partition your RDDs

\* Persistence: Allows you to choose whether you want to persist RDD onto disk or not

--------------------------------------------------------------------------

**INTERCLOUD**

\* The concept of connected cloud networks including public, private and hybrid clouds.

\* It improves the inter-operability and portability among the cloud networks.

\* It is used to connect different cloud computing platforms and allows the data and application to be ported between data centres as cloud services.

\* The main focus is a direct interoperability between public cloud service providers.