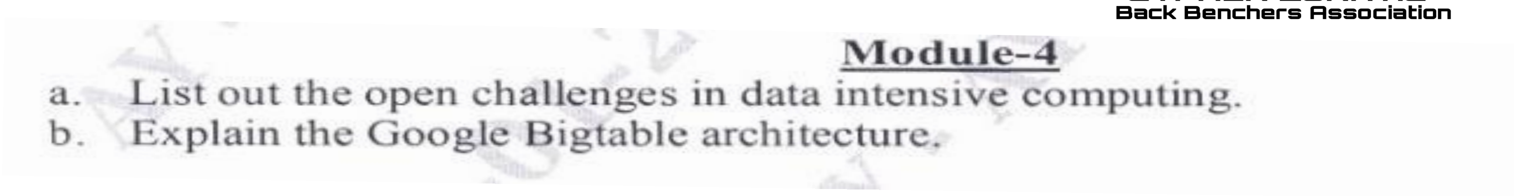
CYPHER LUNATIC
Back Benchers Association



Module-4

a. List out the open challenges in data intensive computing.
b. Explain the Google Bigtable architecture.

The huge amount of data produced, analyzed, or stored imposes requirements on the supporting infrastructures and middleware that are hardly found in the traditional solutions.

Moving terabytes of data becomes an obstacle for high-performing computations.

Data partitioning, content replication and scalable algorithms help in improving the performance.

Open challenges in data-intensive computing given by Ian Gorton et al. are:

1. Scalable algorithms that can search and process massive datasets.

2. New metadata management technologies that can handle complex, heterogeneous, and distributed data sources.

3. Advances in high-performance computing platforms aimed at providing a better support for accessing in-memory multiterabyte data structures.

4. High-performance, highly reliable, petascale distributed file systems.

5. Data signature-generation techniques for data reduction and rapid processing.

6. Software mobility that are able to move the computation to where the data are located.

7. Interconnection architectures that provide better support for filtering multi gigabyte datastreams coming from high-speed networks and scientific instruments.

8. Software integration techniques that facilitate the combination of software modules running on different platforms to quickly form analytical pipelines.

………

Google Bigtable.

Bigtable provides storage support for several Google applications that expose different types of workload: from throughput-oriented batch-processing jobs to latency-sensitive serving of data to end users.

Bigtable’s key design goals are wide applicability, scalability, high performance, and high availability. To achieve these goals, Bigtable organizes the data storage in tables of which the rows are distributed over the distributed file system supporting the middleware, which is the Google File System.

From a logical point of view, a table is a multidimensional sorted map indexed by a key that is represented by a string of arbitrary length. A table is organized into rows and columns; columns can be grouped in column family, which allow for specific optimization for better access control, the storage and the indexing of data.

Bigtable APIs also allow more complex operations such as single row transactions and advanced data manipulation

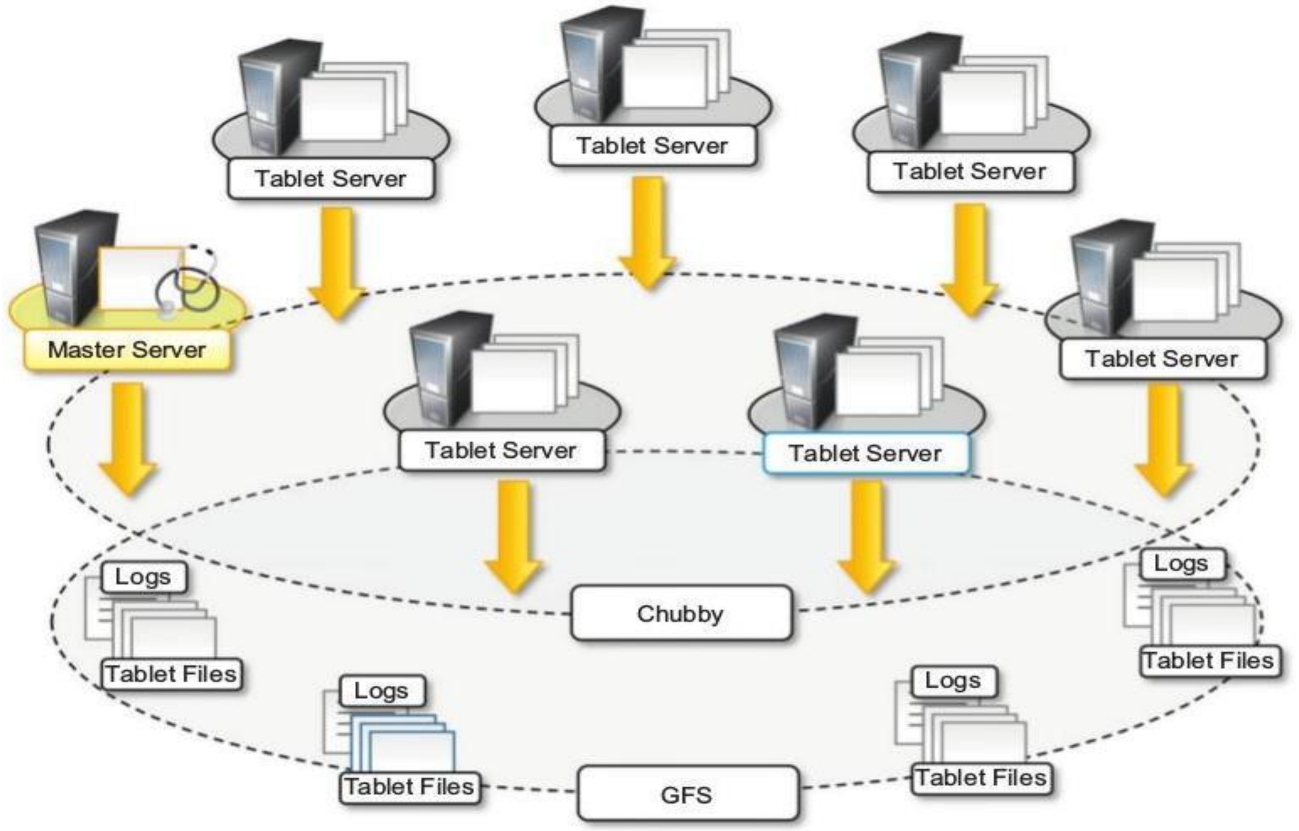


FIGURE 8.4

Bigtable architecture.

HBase is designed by taking inspiration from Google Bigtable; its main goal is to offer real-time read/write operations for tables with billions of rows and millions of columns by leveraging clusters of commodity hardware. The internal architecture and logic model of HBase is very similar to Google Bigtable, and the entire system is backed by the Hadoop Distributed File System (HDFS).

....

Explain the map reduce programming model

MapReduce expresses the computational logic of an application in two simple functions: map and reduce.

Data transfer and management are completely handled by the distributed storage infrastructure (i.e., the Google File System), which is in charge of providing access to data, replicating files, and eventually moving them where needed.

the MapReduce model is expressed in the form of the two functions, which are defined as follows:

$$map\,(k1, v1) \rightarrow list(k2, v2)$$
$$reduce(k2, list(v2)) \rightarrow list(v2)$$

The map function reads a key-value pair and produces a list of key-value pairs of different types. The reduce function reads a pair composed of a key and a list of values and produces a list of values of the same type. The types (k1,v1,k2,kv2) used in the expression of the two functions provide hints as to how these two functions are connected and are executed to carry out the computation of a MapReduce job: The output of map tasks is aggregated together by grouping the values according to their corresponding keys and constitutes the input of reduce tasks that, for each of the keys found, reduces the list of attached values to a single value. Therefore, the input of a MapReduce computation is expressed as a collection of key-value pairs < k1,v1 >, and the final output is represented by a list of values: list(v2).

Figure 8.5 depicts a reference workflow characterizing MapReduce computations. As shown, the user submits a collection of files that are expressed in the form of a list of < k1,v1 > pairs and specifies the map and reduce functions. These files are entered into the distributed file system that supports MapReduce and, if necessary, partitioned in order to be the input of map tasks. Map tasks generate intermediate files that store collections of < k2, list(v2) > pairs, and these files are saved into the distributed file system. These files constitute the input of reduce tasks, which finally produce output files in the form of list(v2).

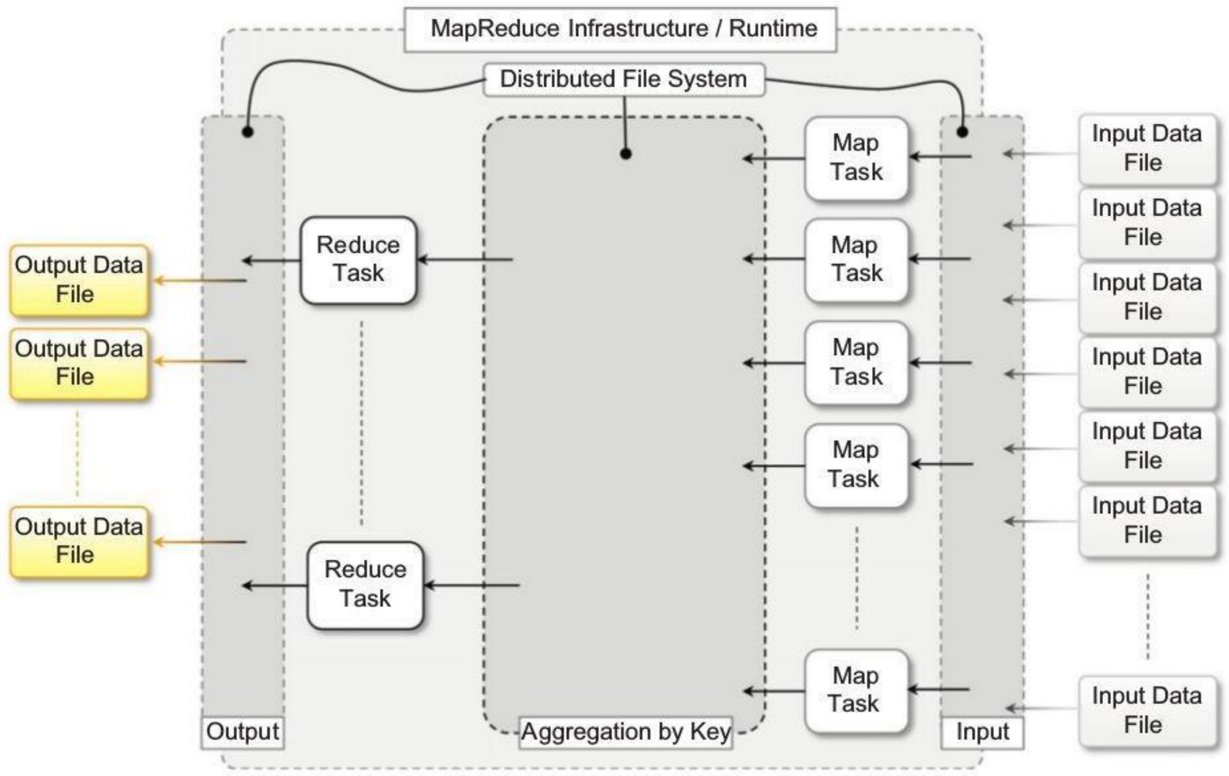


FIGURE 8.5

MapReduce computation workflow.

....

Explain any three distributed file system

1)Hadoop.

Apache Hadoop is a collection of software projects for reliable and scalable distributed computing. The initiative consists of mostly two projects: Hadoop Distributed File System (HDFS) and Hadoop MapReduce. The former is an implementation of the Google File System; the latter provides the same features and abstractions as Google MapReduce

2. Google File System (GFS).

GFS is the storage infrastructure that supports the execution of distributed applications in Google’s computing cloud.

GFS is designed with the following assumptions:

1. The system is built on top of commodity hardware that often fails.

2. The system stores a modest number of large files; multi-GB files are common and should be treated efficiently, and small files must be supported, but there is no need to optimize for that.

3. The workloads primarily consist of two kinds of reads: large streaming reads and small random reads.

4. The workloads also have many large, sequential writes that append data to files.

5. High-sustained bandwidth is more important than low latency.

The architecture of the file system is organized into a single master, which contains the metadata of the entire file system, and a collection of chunk servers, which provide storage space. From a logical point of view the system is composed of a collection of software daemons, which implement either the master server or the chunk server.

3). Amazon Simple Storage Service (S3)

. Amazon S3 is the online storage service provided by Amazon.

The system offers a flat storage space organized into buckets, which are attached to an Amazon Web Services (AWS) account. Each bucket can store multiple objects, each identified by a unique key. Objects are identified by unique URLs and exposed through HTTP, thus allowing very simple get-put semantics.

4) IBM General Parallel File System (GPFS).

GPFS is the high-performance distributed file system developed by IBM that provides support for the RS/6000 supercomputer and Linux computing clusters. GPFS is a multiplatform distributed file system built over several years of academic research and provides advanced recovery mechanisms. GPFS is built on the concept of shared disks, in which a collection of disks is attached to the file system nodes by means of some switching fabric. The file system makes this infrastructure transparent to users and stripes large files over the disk array by replicating portions of the file to ensure high availability

CYPHER LUNATIC
Back Benchers Association