UNIT IV

4 Data Storage and Security in Cloud

Syllabus

Cloud file systems : *GFS and HDFS, BigTable, HBase and Dynamo Cloud data stores : Datastore and Simple DB Gautam Shrauf, Cloud Storage - Overview, Cloud Storage Providers.*
Securing the Cloud *- General Security Advantages of Cloud-Based Solutions, Introducing Business Continuity and Disaster Recovery. Disaster Recovery- Understanding the Threats.*

Contents

4.1 Cloud File Systems

- Cloud File system should be scalable enough to adopt large organizations file systems under different workloads with good performance requirements.
- Cloud file storage is a method for storing data in the cloud that provides servers and applications access to data through shared file systems. This compatibility makes cloud file storage ideal for workloads that rely on shared file systems and provides simple integration without code changes.
- A file system in the cloud is a hierarchical storage system that provides shared access to file data. Users can create, delete, modify, read, and write files and can organize them logically in directory trees for intuitive access.

Benefits of Cloud File Storage

- Storing file data in the cloud delivers advantages in three key areas :
  1. **Scalability** : Although not every cloud file storage solution can scale, leveraging all the capabilities of the cloud, the most advanced solutions provide the ability to start with the capacity and performance you need today and grow your capacity as needed. No more over provisioning to try and anticipate future needs.
  2. **Interoperability** : Many existing applications require integration with shared file services that follow existing file system semantics. Cloud file storage solutions offer a distinct advantage as there is no new code to write to have secure, shared file access.
  3. **Budget and Resources** : Operating file services on-premises requires budget for hardware, ongoing maintenance, power, cooling, and physical space. Cloud file storage enables organizations to redeploy technical resources to other projects that bring more value to the business.

4.2 Google File System

- Google file system is "a scalable distributed file system for large distributed data-intensive applications" created by Google. Initially used to store Google's search indexes and the crawling data, GFS is now mostly used to store user generated content.
- GFS was built primarily as the fundamental storage service for Google's search engine.
- GFS typically will hold a large number of huge files, each 100 MB or larger, with files that are multiple GB in size quite common. Thus, Google has chosen its file data block size to be 64 MB instead of the 4 kB in typical traditional file systems. The I/O pattern in the Google application is also special.

- Files are typically written once, and the write operations are often the appending data blocks to the end of files. Multiple appending operations might be concurrent. There will be a lot of large streaming reads and only a little random access.
- There is no data cache in GFS as large streaming reads and writes represent neither time nor space locality

Motivation behind GFS

- Fault tolerance and auto-recovery need to be built into the systems i.e. monitoring, error detection, fault tolerance, automatic recovery. Because problems are very often caused by application bugs, OS bugs, human errors, and the failure of disks, memory, connectors, networking, and power supplies.
- Standard I/O assumptions (e.g. block size) have to be re - examined.
- Record appends are the frequent form of writing and Google applications and GFS should be co - designed.

4.2.1 GFS Architecture

- A GFS cluster consists of a single master and multiple chunk servers and is accessed by multiple clients.

Basic terms :

a. **Master :** Single, coordinates system-wide activities. Can have read-only 'Shadow' servers.

b. **Chunk :** 64 MB storage block representing a file or piece thereof.

c. **Chunkserver :** Many, stores chunks of data.

d. **Replica :** Either primary or secondary. A Chunkserver that replicates a given block.

e. **Client :** Runs tasks on data.

- It is easy to run both a chunkserver and a client on the same machine, as long as machine resources permit.
- Files are divided into fixed-size chunks. Each chunk is identified by an immutable and globally unique 64 bit chunk handle assigned by the master at the time of chunk creation.
- Chunkservers store chunks on local disks as Linux files and read or write chunk data specified by a chunk handle and byte range. For reliability, each chunk is replicated on multiple chunkservers.
- The master maintains all file system metadata. This includes the namespace, access control information, the mapping from files to chunks, and the current locations of chunks.

- Clients interact with the master for metadata operations, but all data-bearing communication goes directly to the chunkservers.
- Neither the client nor the chunkserver caches file data. Fig. 4.2.1 shows GFS architecture.

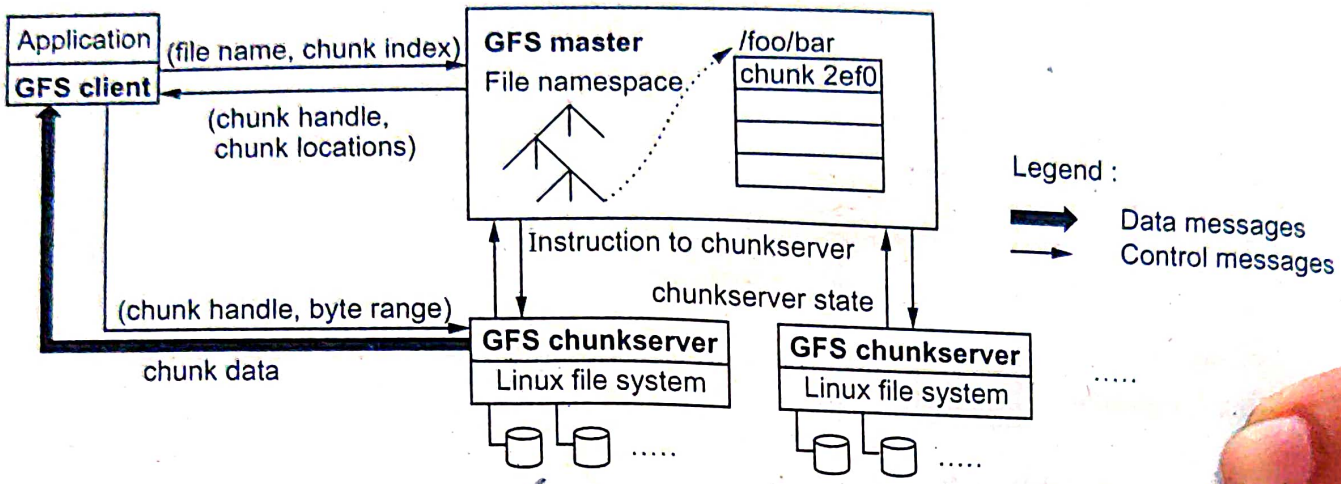

Fig. 4.2.1 GFS architecture

- Clients never read and write file data through the master. Instead, a client asks the master which chunkservers it should contact. It caches this information for a limited time and interacts with the chunkservers directly for many subsequent operations.
- First, using the fixed chunk size, the client translates the file name and byte offset specified by the application into a chunk index within the file. Then, it sends the master a request containing the file name and chunk index.
- The master replies with the corresponding chunk handle and locations of the replicas. The client caches this information using the file name and chunk index as the key. The client then sends a request to one of the replicas, most likely the closest one.
- The request specifies the chunk handle and a byte range within that chunk.
- Further reads of the same chunk require no more client-master interaction until the cached information expires or the file is reopened.
- In fact, the client typically asks for multiple chunks in the same request and the master can also include the information for chunks immediately following those requested. This extra information sidesteps several future client-master interactions at practically no extra cost

4.2.2 Chunk Size and Metadata of GFS

Chunk size :

- Chunk size is 64 MB. Each chunk replica is stored as a plain Linux file on a chunkserver and is extended only as needed.
- Advantages of large chunk :
  1. It reduces the network overheads.
  2. It reduces the size of the metadata stored on the master.
  3. It reduces clients' need to interact with the master because reads and writes on the same chunk require only one initial request to the master for chunk location information.

Metadata :

- The master stores three major types of metadata :
  1. File and chunk namespaces,
  2. Mapping from files to chunks,
  3. Locations of each chunk's replicas.
- All metadata is kept in the master's memory. The first two types are also kept persistent by logging mutations to an operation log stored on the master's local disk and replicated on remote machines.
- Using a log allows us to update the master state simply, reliably, and without risking inconsistencies in the event of a master crash.
- The master does not store chunk location information persistently. Instead, it asks each chunkserver about its chunks at master startup and whenever a chunkserver joins the cluster.
- Chunk replicas are created for three reasons : chunk creation, re-replication, and rebalancing

4.2.3 Data Mutation Sequence in GFS

- Here we discuss Write Control and Data Flow (data mutation) sequence of GFS.
- Fig. 4.2.2 shows process of control flow of a write through.
- The client asks the master which chunk server holds the current lease for the chunk and the locations of the other replicas. If no one has a lease, the master grants one to a replica it chooses.
- The master replies with the identity of the primary and the locations of the other (secondary) replicas. The client caches this data for future mutations. It needs to

contact the master again only when the primary becomes unreachable or replies that it no longer holds a lease.

- The client pushes the data to all the replicas. A client can do so in any order. Each chunk server will store the data in an internal LRU buffer cache until the data is used or aged out.
- Once all the replicas have acknowledged receiving the data, the client sends a write request to the primary.

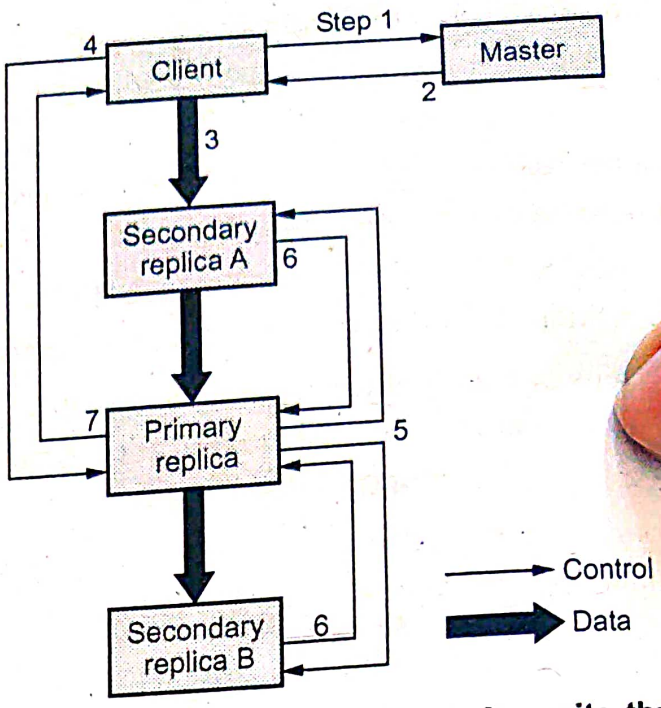

Fig. 4.2.2 Process of control flow of a write through

- The primary forwards the write request to all secondary replicas. Each secondary replica applies mutations in the same serial number order assigned by the primary.
- The secondary's all reply to the primary indicating that they have completed the operation.
- The primary replies to the client. Any errors encountered at any of the replicas are reported to the client.

4.2.4 Big Data Processing Challenge

- Big data organizes and extracts the valued information from the rapidly growing, large volumes, variety forms and frequently changing data sets collected from multiple and autonomous sources in the minimal possible time, using several statistical and machine learning techniques.
- GFS is high performance file system on commodity hardware clusters for large scale data processing
- Google offers big query to operate on Google big tables. New generation of query languages, examples are Google big query.
- Web log mining is the study of the data available in the web. This involves searching for the texts, words, and their occurrences.
- One example for web log mining is searching for the words, and their frequencies by Google big query data analytics use Google big query platform to run on the Google cloud infrastructure.

- Mapreduce framework has made complex large-scale data processing easy and efficient. MapReduce is inherently designed for high throughput batch processing of big data that take several hours and even days, while recent demands are more centered on jobs and queries that should finish in seconds or at most, minutes.

4.3 HDFS

- Hadoop Distributed File System (HDFS) is a distributed file system inspired by GFS that organizes files and stores their data on a distributed computing system.
- The Hadoop core is divided into two fundamental layers : the MapReduce engine and HDFS.
- The MapReduce engine is the computation engine running on top of HDFS as its data storage manager
- Hadoop is an open-source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license. It provide software framework for distributed processing of large datasets in real-time applications.
- Hadoop provides the basic platform for big data processing. The hadoop architecture have mainly two parts: Hadoop distributed File System (HDFS) and the MapReduce engine.
- HDFS is Distributed Files system designed to run on commodity hardware, which is highly fault- tolerant andscalable. Fig. 4.3.1 shows HDFS architecture.

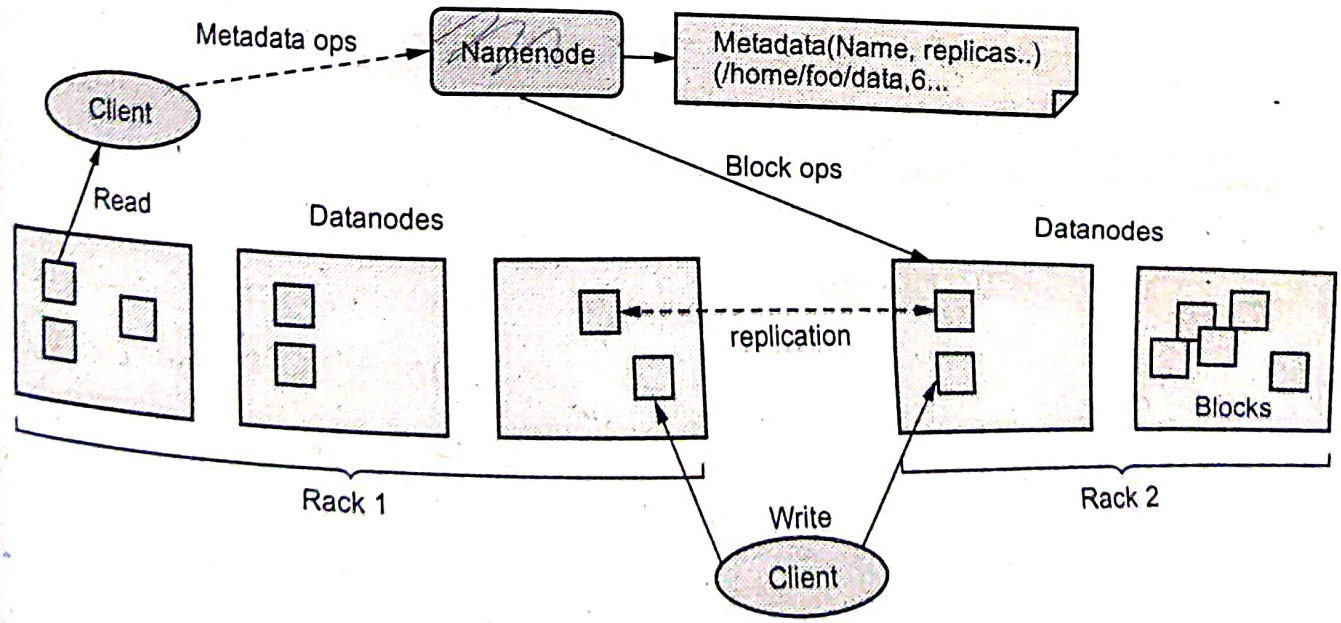

Fig. 4.3.1 Hadoop architecture

- Hadoop Distributed File System is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines.
- Apache Hadoop HDFS Architecture follows a Master/Slave Architecture, where a cluster comprises of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes).
- To store a file in this architecture, HDFS splits the file into fixed-size blocks (e.g., 64 MB) and stores them on workers (DataNodes).
- HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.
- NameNode is the master node in the Hadoop HDFS Architecture that maintains and manages the blocks present on the DataNodes (slave nodes).
- NameNode is a very highly available server that manages the File System Namespace and controls access to files by clients.
- DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is a commodity hardware, that is, a non-expensive system which is not of high quality or high-availability. The DataNode is a block server that stores the data in the local file ext3 or ext4.
- Journal is the modification log of image, which is available in local hosts native file system. Journal is updated for every client transaction.
- Checkpoint is persistent record of the image, which is also stored on local hosts native file system to enable recovery. NameNode is not allowed to update or modify Checkpoint file.
- Administrator or Checkpoint Node can demand to create new checkpoint file on startup, or restart.

4.3.1 Node Types

1. NameNode :

- It is also known as Master node. Namenode stores meta - data i.e. number of blocks, their location, replicas and other details.
- This meta - data is available in memory in the master for faster retrieval of data.
- NameNode maintains and manages the slave nodes, and assigns tasks to them. It should be deployed on reliable hardware as it is the centerpiece of HDFS.

2. Checkpoint node :

- Checkpoint node is a node which periodically creates checkpoints of the namespace.

- Checkpoint Node in Hadoop first downloads fsimage and edits from the active Namenode. Then it merges them locally, and at last it uploads the new image back to the active NameNode.
- The checkpoint Node stores the latest checkpoint in a directory. It is structured in the same as the Namenode's directory. It permits the checkpointed image to available for reading by the namenode.

3. Backup node :

- Backup node provides the same checkpointing functionality as the Checkpoint node. In Hadoop, Backup node keeps an in-memory, up-to-data copy of the file system namespace, which is always synchronized with the active NameNode state.
- The Backup node does not need to download fsimage and edits files from the active NameNode in order to create a checkpoint, as would be required with a Checkpoint node or Secondary Namenode, since it already has an up-to-data state of the namespace state in memory.
- The Backup node checkpoint process is more efficient as it only needs to save the namespace into the local fsimage file and reset edits.
- One Backup node is supported by the NameNode at a time. No checkpoint nodes may be registered if a Backup node is in use.

4.3.2 Block Placement in Hadoop

- For a large cluster, it may not be practical to connect all nodes in a flat topology. A common practice is to spread the nodes across multiple racks.
- Nodes of a rack share a switch, and rack switches are connected by one or more core switches. Communication between two nodes in different racks has to go through multiple switches.
- In most cases, network bandwidth between nodes in the same rack is greater than network bandwidth between nodes in different racks.
- Fig. 4.3.2 shows cluster with two racks, each of which contains three nodes.
- HDFS estimates the network bandwidth between two nodes by their distance. The distance from a node to its parent node is assumed to be one.

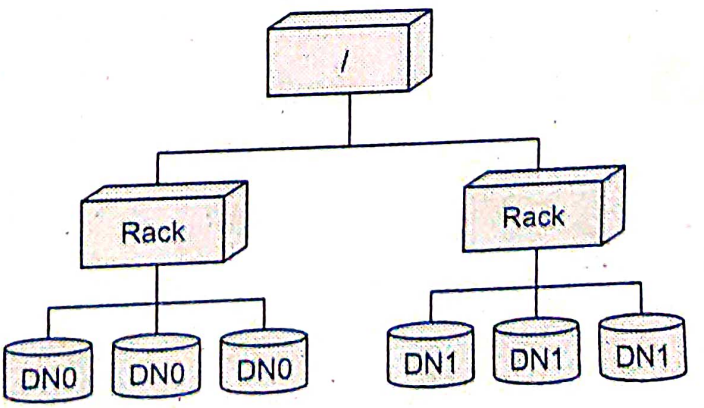

Fig. 4.3.2 Block placement

- A distance between two nodes can be calculated by summing the distances to their closest common ancestor. A shorter distance between two nodes means greater bandwidth they can use to transfer data.
- HDFS allows an administrator to configure a script that returns a node's rack identification given a node's address.
- The NameNode is the central place that resolves the rack location of each DataNode. When a DataNode registers with the NameNode, the NameNode runs the configured script to decide which rack the node belongs to.
- If no such a script is configured, the NameNode assumes that all the nodes belong to a default single rack.
- The default HDFS block placement policy provides a tradeoff between minimizing the write cost, and maximizing data reliability, availability and aggregate read bandwidth.
- When a new block is created, HDFS places the first replica on the node where the writer is located. The second and the third replicas are placed on two different nodes in a different rack.
- The rest are placed on random nodes with restrictions that no more than one replica is placed at any one node and no more than two replicas are placed in the same rack, if possible.
- The choice to place the second and third replicas on a different rack better distributes the block replicas for a single file across the cluster. If the first two replicas were placed on the same rack, for any file, two-thirds of its block replicas would be on the same rack.
- After all target nodes are selected, nodes are organized as a pipeline in the order of their proximity to the first replica. Data are pushed to nodes in this order.
- For reading, the NameNode first checks if the client's host is located in the cluster. If yes, block locations are returned to the client in the order of its closeness to the reader. The block is read from DataNodes in this preference order

4.3.3 File System Namespace

- HDFS supports a traditional hierarchical file organization in which a user or an application can create directories and store files inside them.
- The file system namespace hierarchy is similar to most other existing file systems; you can create, rename, relocate, and remove files.
- HDFS also supports third-party file systems such as CloudStore and Amazon Simple Storage Service.

- **Data replication :** HDFS replicates file blocks for fault tolerance. An application can specify the number of replicas of a file at the time it is created, and this number can be changed any time after that. The name node makes all decisions concerning block replication.
- HDFS uses an intelligent replica placement model for reliability and performance. Optimizing replica placement makes HDFS unique from most other distributed file systems, and is facilitated by a rack-aware replica placement policy that uses network bandwidth efficiently.
- **Data organization :** One of the main goals of HDFS is to support large files. The size of a typical HDFS block is 64MB. Therefore, each HDFS file consists of one or more 64MB blocks. HDFS tries to place each block on separate data nodes.
- HDFS applications need a write-once-read-many access model for files. A file once created, written, and closed need not be changed
- HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients.
- HDFS supports a traditional hierarchical file organization. A user or an application can create directories and store files inside these directories.
- HDFS does not support hard links or soft links. Files in HDFS are write - once and have strictly one writer at any time

4.3.4 MapReduce

- MapReduce is a programming model and software framework first developed by Google. Intended to facilitate and simplify the processing of vast amounts of data in parallel on large clusters of commodity hardware in a reliable, fault-tolerant manner.

Characteristics of MapReduce :

1. Very large scale data : peta, exa bytes
2. Write once and read many data. It allows for parallelism without mutexes
3. Map and Reduce are the main operations : Simple code
4. All the map should be completed before reduce operation starts.
5. Map and reduce operations are typically performed by the same physical processor.
6. Number of map tasks and reduce tasks are configurable

Data flow in the MapReduce programming model

- MapReduce : A programming model to facilitate the development and execution of distributed tasks.
- The programmer defines the program logic as two functions :
  a. Map transforms the input into key - value pairs to process
  b. Reduce aggregates the list of values for each key
- The MapReduce environment takes in charge distribution aspects. A complex program can be decomposed as a succession of Map and Reduce tasks. Higher-level languages (Pig, Hive, etc.) help with writing distributed applications.
- MapReduce is a parallel programming model especially dedicated for complex and distributed computations, which has been derived from the functional paradigm.
- In general, MapReduce processing is composed of two consecutive stages, which for most problems are repeated iteratively: the map and the reduce phase. Fig. 4.3.3 shows data flow in the MapReduce programming model.

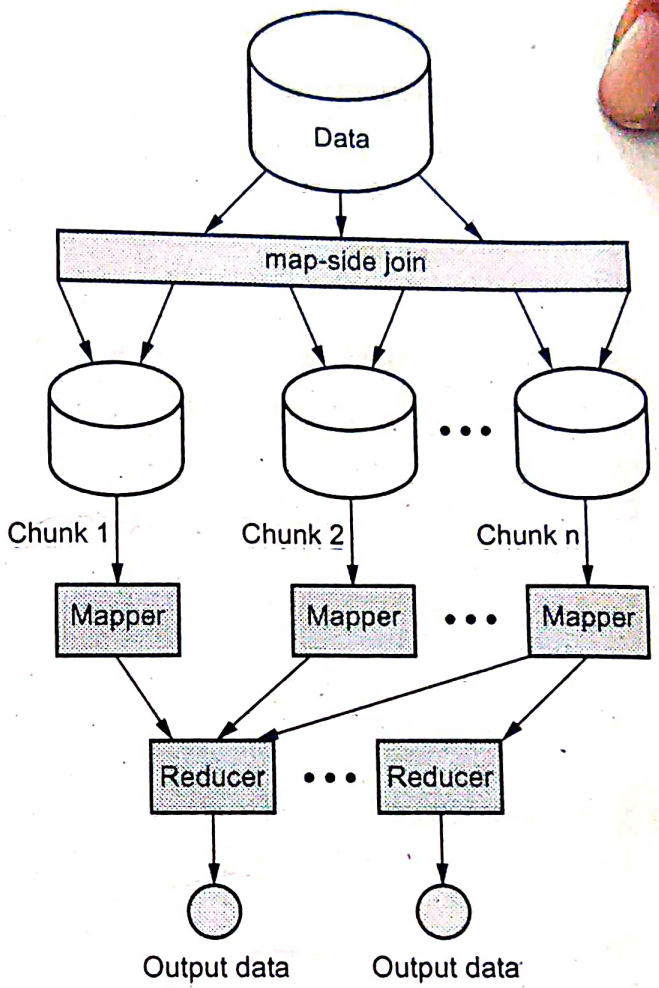

Fig. 4.3.3 Data flow in map reduce

- Map processes the data on hosts in parallel, whereas the reduce aggregates the results. At each iteration independently, the whole data is split into chunks, which, in turn, are used as the input for mappers.
- Each chunk may be processed by only one mapper. Once the data is processed by mappers, they can emit View the MathML source?key,value? pairs to the reduce phase.
- Before the reduce phase the pairs are sorted and collected according to the View the MathML sourcekey values, therefore each reducer gets the list of values related to a given View the MathML sourcekey. The consolidated output of the reduce phase is saved into the distributed file system

MapReduce processes big data :

- With HDFS, we are able to distribute the data so that data is stored on hundreds of nodes instead of a single large machine.
- Mapreduce provides the framework for highly parallel processing of data across clusters of commodity hardware. Fig. 4.3.4 shows MapReduce data processing.

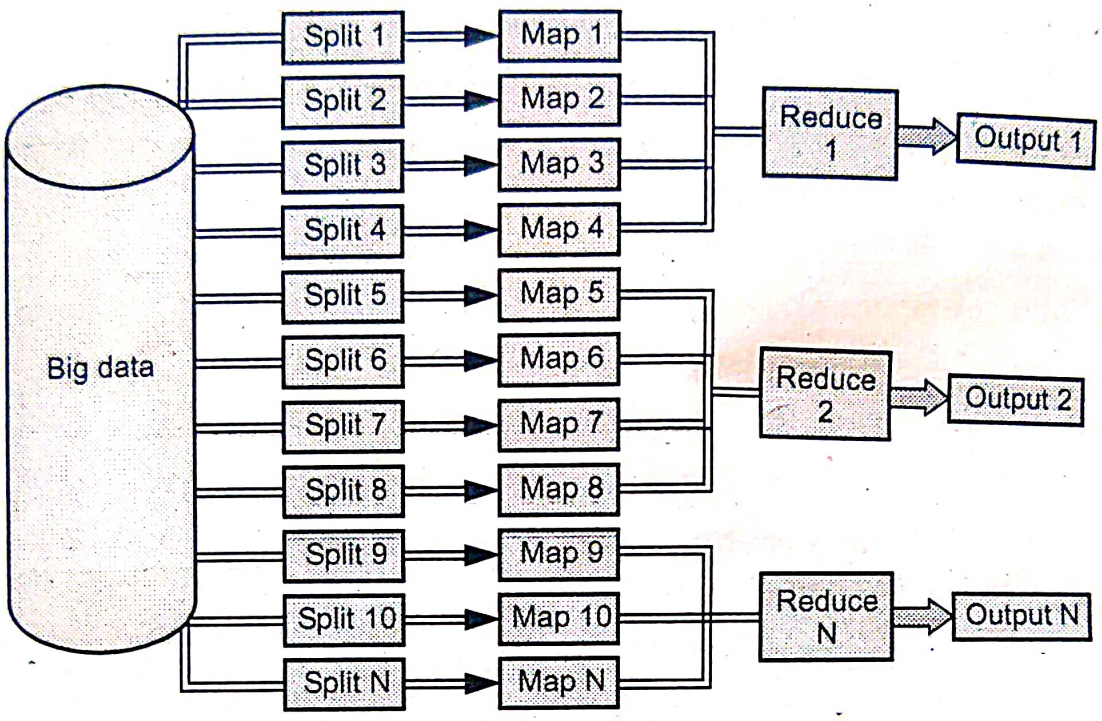

Fig. 4.3.4 MapReduce data processing

- It removes the complicated programming part from the programmers and moves into the framework. Programmers can write simple programs to make use of the parallel processing.
- The framework splits the data into smaller chunks that are processed in parallel on cluster of machines by programs called mappers.
- The output from the mappers is then consolidated by reducers into desired result. The share nothing architecture of mappers and reducers make them highly parallel.
- Data locality is achieved by mapreduce by working closely with HDFS. When you specify the file system as HDFS for mapreduce, it automatically schedules the mappers on the same node as where the block of data exists.
- Mapreduce can get the blocks from HDFS and process them. The final output from Mapreduce also can be stored in HDFS file system. However, the intermediate files between mappers and reducers are not stored in HDFS and are stored on the local file system of the mappers

Hadoop / MapReduce limitations :

1. Cannot control the order in which the maps or reductions are run
2. For maximum parallelism, you need Maps and Reduces to not depend on data generated in the same MapReduce job (i.e. stateless)
3. A database with an index will always be faster than a MapReduce job on unindexed data
4. Reduce operations do not take place until all Maps are complete
5. General assumption that the output of Reduce is smaller than the input to Map large data source used to generate smaller final values

4.4 BigTable

- Bigtable is a distributed storage system that is used for managing and storing structured data at Google.
- Bigtable is designed to reliably scale to petabytes of data and thousands of machine. Bigtable has multiple goals like applicability, high availability, scalability, high performance.
- It is used by approximate sixty Google project or product like Google Analytic, Google Finance, Personalized search, Writely, and Google Earth.
- Bigtable is built on Google file system for storing the data for scheduling large scale data processing. It stored data in form of rows, columns and timestamp that means it maps with arbitrary string value like row key and column key as well as timestamp.
- It uses MapReduce. Bigtable maintains data in alphabetical order by row key. The row keys in a table are arbitrary strings
- Rows are the unit of transactional consistency. Several rows are grouped in tablets which are distributed and stored close to each other. Reads of short row ranges are efficient, typically require communication with only one or a few machines.

Features :

1. Bigtable is a distributed storage system for managing structured data.
2. Bigtable uses the distributed Google File System (GFS) to store log and data files.
3. A Bigtable cluster stores a number of tables. Each table consists of a set of tablets, and each tablet contains all data associated with a row range.

4. Bigtable use a highly available and persistent distributed lock services called Chubby
5. Bigtable supports single - row transactions

- The map is indexed by a row key, column key , and a timestamp; each value in the map is an un-interpreted array of bytes.

1. Row key

- The row keys in a table are arbitrary strings. Every read or write of data under a single row key is atomic
- Bigtable maintains data in lexicographic order by row key. The row range for a table is dynamically partitioned.
- Each row range is called a tablet, which is the unit of distribution and load balancing.

2. Column key

- Column keys are grouped into sets called column families, which form the basic unit of access control. All data stored in a column family is usually of the same type.
- A column family must be created before data can be stored under any column key in that family; after a family has been created, any column key within the family can be used.
- Syntax of column key is **family : qualifier.**
- Column family names must be print-able, but qualifiers may be arbitrary strings. An example column family for the Webtable is language, which stores the language in which a web page was written.
- The qualifier is the name of the referring site; the cell contents is the link text.
- Access control and both disk and memory account-ing are performed at the column-family level.

3. Timestamp

- For each column, bigtables use timestamp (64-bit integer) to track update of data cell. For sequential data, when new data wrote in, it adds an additional layer of data with new timestamp.
- Data structure can be viewed as 3D with third dimension. Timestamp is also convenient for defining garbage-collect which means certain data or only last versions of data will be kept.

4.4.1 BigTable Architecture

- SSTable is a file format to stored persistent, ordered immutable map from key to values. It contains block index and can be applied binary search to locate certain block. Client loaded SSTable loaded into memory.
- Fig. 4.4.1 shows Bigtable architecture.

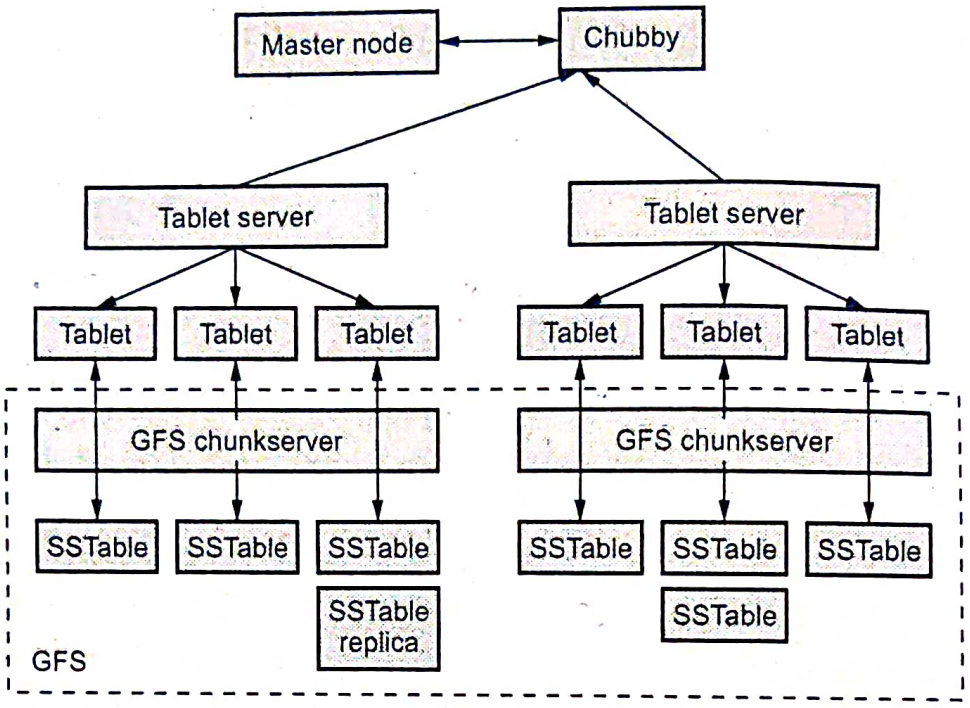

Fig. 4.4.1 Bigtable architecture

- **Master node :** The master is responsible for assigning tablets to tablet servers, detecting the addition and expiration of tablet servers, balancing tablet-server load, and garbage collection in GFS.
- **Tablet server :** Each tablet server manages a set of tablets. The tablet server handles read and write requests to the tablets that it has loaded, and also splits tablets that have grown too large.
- Bigtable uses Chubby to keep track of tablet servers. When a tablet server starts, it creates, and acquires an exclusive lock on, a uniquely named file in a specific Chubby directory.
- The master is responsible for detecting when a tablet server is no longer serving its tablets, and for reassigning those tablets as soon as possible.
- To detect when a tablet server is no longer serving its tablets, the master periodically asks each tablet server for the status of its lock.

- If a tablet server reports that it has lost its lock, or if the master was unable to reach a server during its last several attempts, the master attempts to acquire an exclusive lock on the server's .
- If the master is able to acquire the lock, then Chubby is live and the tablet server is either dead or having trouble reaching Chubby, so the master ensures that the tablet server can never serve again by deleting its server.
- Once a server's has been deleted, the master can move all the tablets that were previously assigned to that server into the set of unassigned tablets.

4.4.2 Tablet Location Hierarchy in Bigtable

- Three level of hierarchy structure is used to store table location information. First level is the chubby with root tablet, it stores locations of all other user tablets and never split.
- In that way it maintains the three level structures. This structure can save up to 2^{61} tablets.
- Client can access table location through cache and the worst situation of client's cache is stale, it takes up to six round trips through the structure.
- Table also contains log information to help debugging and performance analysis.
- The METADATA table stores the location of a tablet under a row key that is an encoding of the tablet' s table identifier and its end row. Each METADATA row stores approximately 1 kB of data in memory.
- With a modest limit of 128 MB METADATA tablets, our three-level location scheme is sufficient to address 2^{34} tablets.
- The client library caches tablet locations. If the client does not know the location of a tablet, or if it discovers that cached location information is incorrect, then it recursively moves up the tablet location hierarchy .

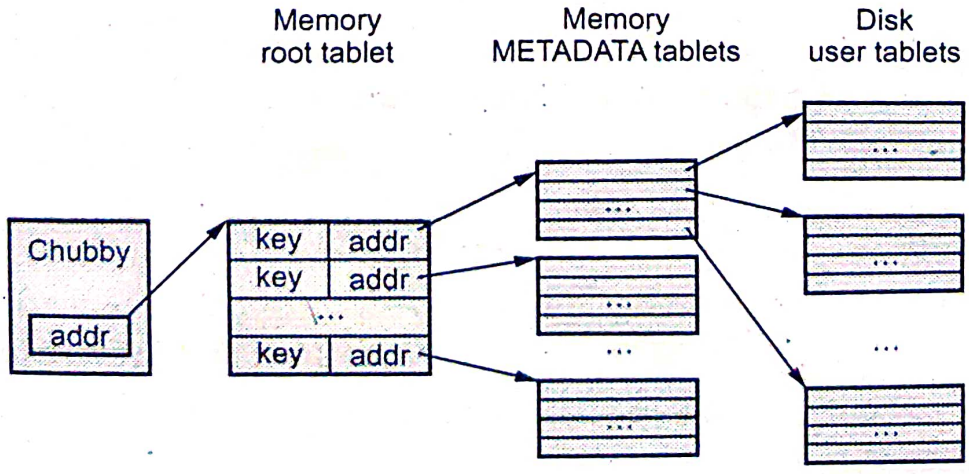

Fig. 4.4.2 Tablet location hierarchy

- If the client' s cache is empty , the location algorithm requires three network round-trips, including one read from Chubby.
- If the client' s cache is stale, the location algorithm could take up to six round-trips, because stale cache entries are only discovered upon misses.
- Although tablet locations are stored in memory, so no GFS accesses are required, we further reduce this cost in the common case by having the client library prefetch tablet locations: it reads the metadata for more than one tablet whenever it reads the METADATA table.

4.4.3 Chubby in Bigtable

- Bigtable relies on a highly-available and persistent distributed lock service called Chubby.
- In Bigtable, Chubby is used to :
  1. ensure there is only one active master
  2. store the bootstrap location of Bigtable data
  3. discover tablet servers
  4. store Bigtable schema information
  5. store access control lists
- A Chubby service consists of five active replicas, one of which is elected to be the master and actively serve requests.
- The service is live when a majority of the replicas are running and can communicate with each other. Chubby uses the Paxos algorithm to keep its replicas consistent in the face of failure.
- Chubby provides a namespace that consists of directories and small files. Each directory or file can be used as a lock, and reads and writes to a file are atomic.
- The Chubby client library provides consistent caching of Chubby files. Each Chubby client maintains a session with a Chubby service.
- A client' s session expires if it is unable to renew its session lease within the lease expiration time. When a client' s session expires, it loses any locks and open handles.
- Chubby clients can also register callbacks on Chubby files and directories for notification of changes or session expiration

4.5 HBase

- HBase is an open source, non-relational, distributed database modeled after Google's BigTable.
- It is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System.
- It runs on top of Hadoop and HDFS, providing BigTable-like capabilities for Hadoop.
- HBase supports massively parallelized processing via MapReduce for using HBase as both source and sink.
- HBase supports an easy to use Java API for programmatic access. It also supports Thrift and REST for non-Java front - ends.
- Hbase is a column oriented distributed database in Hadoop environment. It can store massive amounts of data from terabytes to petabytes. Hbase is scalable, distributed big data storage on top of the Hadoop eco system.
- The HBase Physical Architecture consists of servers in a Master-Slave relationship. Typically, the HBase cluster has one Master node, called HMaster and multiple Region Servers called HRegionServer.
- Fig. 4.5.1 shows Hbase architecture.

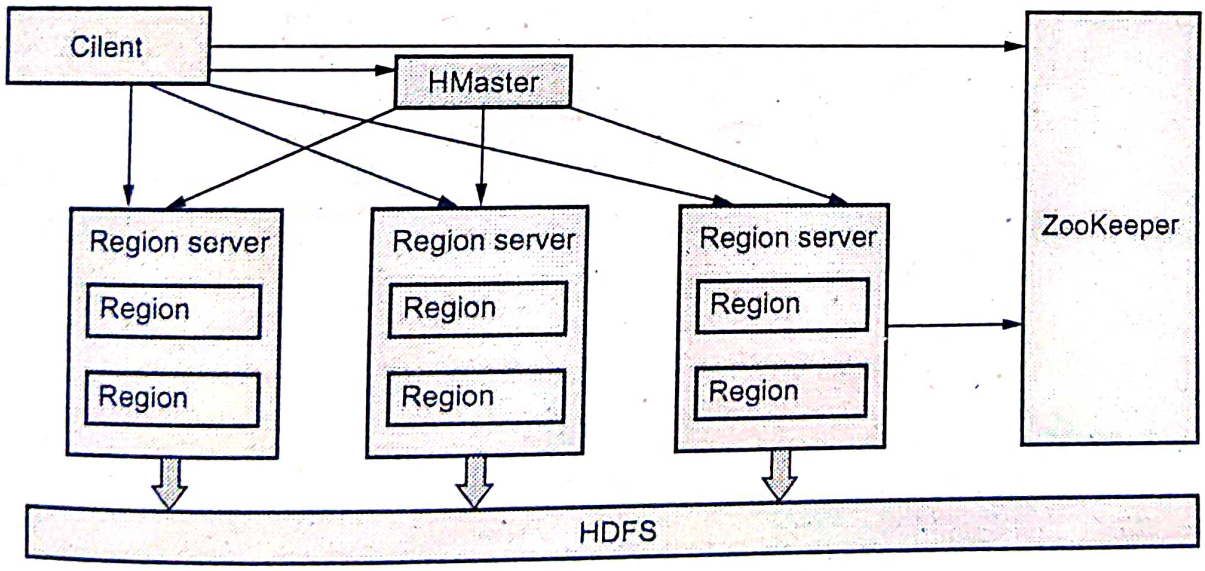

Fig. 4.5.1 Hbase architecture

- Zookeeper is a centralized monitoring server which maintains configuration information and provides distributed synchronization. If the client wants to communicate with regions servers, client has to approach Zookeeper.
- HMaster in the master server of Hbase and it coordinates the HBase cluster. HMaster is responsible for the administrative operations of the cluster.

Big Table vs HBase

- **HRegions Servers :** It will perform the following functions in communication with HMaster and Zookeeper.
  1. Hosting and managing regions.
  2. Splitting regions automatically.
  3. Handling read and writes requests.
  4. Communicating with clients directly.
- **HRegions :** For each column family, HRegions maintain a store. Main components of HRegions are Memstore and Hfile
- Data Model in HBase is designed to accommodate semi-structured data that could vary in field size, data type and columns

4.5.1 Hadoop Ecosystem

- The Hadoop ecosystem refers to the various components of the Apache Hadoop software library, as well as to the accessories and tools provided by the Apache Software Foundation for these types of software projects, and to the ways that they work together. Hadoop is a Java-based framework that is extremely popular for handling and analyzing large sets of data.
- The idea of a Hadoop ecosystem involves the use of different parts of the core Hadoop set such as MapReduce, a framework for handling vast amounts of data, and the Hadoop Distributed File System (HDFS), a sophisticated file-handling system. There is also YARN, a Hadoop resource manager.
- In addition to these core elements of Hadoop, Apache has also delivered other kinds of accessories or complementary tools for developers.
- These include Apache Hive, a data analysis tool; Apache Spark, a general engine for processing big data; Apache Pig, a data flow language; HBase, a database tool; and also Ambarl, which can be considered as a Hadoop ecosystem manager, as it helps to administer the use of these various Apache resources together.
- Fig. 4.5.2 shows Apache Hadoop ecosystem.
- **Hadoop HDFS -** Distributed storage layer for Hadoop.
- **Yarn Hadoop -** Resource management layer introduced in Hadoop 2.x.
- **Hadoop Map-Reduce -** Parallel processing layer for Hadoop.
- **HBase -** It is a column-oriented database that runs on top of HDFS. It is a NoSQL database which does not understand the structured query. For sparse data set, it suits well.
- **Hive -** Apache Hive is a data warehousing infrastructure based on Hadoop and it enables easy data summarization, using SQL queries.

- **Pig** - It is a top-level scripting language. As we use it with Hadoop. Pig enables writing complex data processing without Java programming.

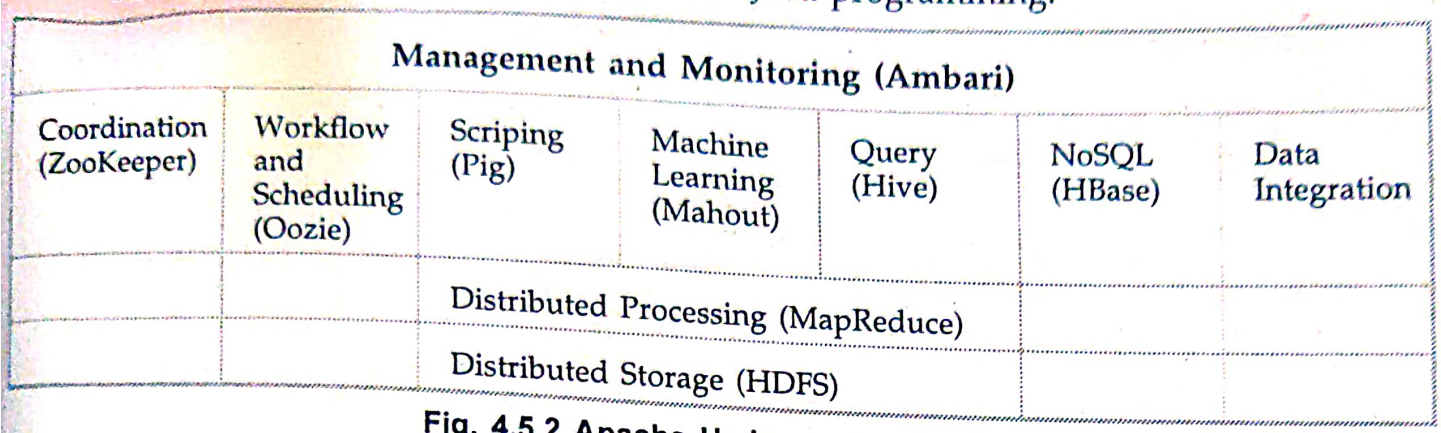

Fig. 4.5.2 Apache Hadoop ecosystem

- **Oozie** - It is a Java Web application uses to schedule Apache Hadoop jobs. It combines multiple jobs sequentially into one logical unit of work.
- **Zookeeper** - A centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.
- **Mahout** - A library of scalable machine-learning algorithms, implemented on top of Apache Hadoop and using the MapReduce paradigm.

4.5.2 Difference HDFS and HBase

HDFS	HBase
HDFS is a distributed file system suitable for storing large files.	HBase is a database built on top of the HDFS
HDFS does not support fast individual record lookups.	HBase provides fast lookups for larger tables.
It provides high latency batch processing; no concept of batch processing.	It provides low latency access to single rows from billions of records (Random access).
It provides only sequential access of data.	HBase internally uses Hash tables and provides random access, and it stores the data in indexed HDFS files for faster lookups.
HDFS are suited for high latency operations	Hbase is suited for low latency operations
In HDFS, data are primarily accessed through Map Reduce jobs.	Hbase provides access to single rows from billions of records
HDFS doesn't have the concept of random read and write operations	Hbase data is accessed through shell commands, client API in Java, REST, Avro or Thrift.

4.6 Dynamo

- Dynamo is propriety key value structured storage system. It can act as database and also distributed hash table.
- Dynamo dynamically partitions a set of keys over a set of storage nodes
- It is most powerful relational database available in WWW. Relational databases have been used a lot in retail sites, to make visitors browse and search for product easily.
- Dynamo does not support replication.
- Dynamo is used to manage the state of services that have very high reliability requirements and need tight control over the tradeoffs between availability, consistency, cost-effectiveness and performance.
- There are many services on Amazon's platform that only need primary-key access to a data store. For many services, such as those that provide best seller lists, shopping carts, customer preferences, session management, sales rank, and product catalog, the common pattern of using a relational database would lead to inefficiencies and limit scale and availability. Dynamo provides a simple primary-key only interface to meet the requirements of these applications.
- Dynamo is a completely decentralized system with minimal need for manual administration. Storage nodes can be added and removed from Dynamo without requiring any manual partitioning or redistribution.
- Compared to Bigtable, Dynamo targets applications that require only key/value access with primary focus on high availability where updates are not rejected even in the wake of network partitions or server failures.
- Dynamo stores objects associated with a key through a simple interface; it exposes two operations : get() and put().
- Dynamo treats both the key and the object supplied by the caller as an opaque array of bytes. It applies a MD5 hash on the key to generate a 128-bit identifier, which is used to determine the storage nodes that are responsible for serving the key.
- Dynamo's partitioning scheme relies on consistent hashing to distribute the load across multiple storage hosts. In consistent hashing , the output range of a hash function is treated as a fixed circular space or "ring".

- Dynamo provides eventual consistency, which allows for updates to be propagated to all replicas asynchronously.
- Dynamo uses vector clocks in order to capture causality between different versions of the same object. A vector clock is effectively a list of (node, counter) pairs. One vector clock is associated with every version of every object.
- In Dynamo, when a client wishes to update an object, it must specify which version it is updating. This is done by passing the context it obtained from an earlier read operation, which contains the vector clock information.
- In Dynamo, each storage node has three main software components: request coordination, membership and failure detection, and a local persistence engine. All these components are implemented in Java.

4.7 Cloud Storage

- Cloud storage is a cloud computing model in which data is stored on remote servers accessed from the internet, or "cloud." It is maintained, operated and managed by a cloud storage service provider on a storage servers that are built on virtualization techniques.
- Cloud storage is part of cloud computing. All cloud storage services provide drag?and?drop accessing and syncing of folders and files between your desktop and mobile devices, and the cloud drive. They also all allow account users to collaborate with each other on documents.
- A cluster of interconnected server farms provided by the cloud storage service connects to the Internet which connects to user computer.
- Cloud storage can reduce costs, simplify IT management, improve user experience, and allow employees to work and collaborate from remote locations
- Cloud storage services may be accessed through a web service API, a cloud storage gateway or through a web-based user interface.
- Cloud storage is used in many different ways. For example : local data (such as on a desktop) can be backed up to cloud storage; a virtual disk can be "synched" to the cloud and distributed to other computers; and the cloud can be used as an archive to retain data for regulatory or other purposes.

- **Benefits**
  1. Files saved to the cloud can be accessed by any device, anywhere you have internet access.
  2. Important files, photos, and videos are saved where they won't be lost if your computer crashes.
  3. Allows easy file sharing with family and/or friends.

Public cloud

- The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

Public cloud benefits

1. Low investment hurdle : Pay for what you use.
2. Good test/development environment for applications that scale to many servers

Public cloud risks

1. Security concerns : Multi - tenancy and transfers over the Internet.
2. IT organization may react negatively to loss of control over data center function

Private cloud

- The cloud infrastructure is operated solely for a single organization. It may be managed by the organization or a third party and may exist on-premises or off-premises.

Private cloud benefits

1. Fewer security concerns as existing data center security stays in place.
2. IT organization retains control over data center.

Private cloud risks

1. High investment hurdle in private cloud implementation, along with purchases of new hardware and software.
2. New operational processes are required; old processes not all suitable for private cloud.

- **Community cloud** : The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g. mission, security requirements, policy, or compliance considerations). It may be managed by the organizations or a third party and may exist on-premises or off - premises.
- **Hybrid cloud** : The cloud infrastructure is a composition of two or more clouds (private, community or public) that remain unique entities but are bound together

by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds).

Hybrid cloud benefits

1. Operational flexibility : Run mission critical on private cloud, dev/test on public cloud
2. Scalability : Run peak and bursty workloads on the public cloud

Hybrid cloud risks

1. Hybrid clouds are still being developed; not many in real use
2. Control of security between private and public clouds, some of same concerns as in public cloud.

4.8 Cloud Storage Providers

- Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.
- Amazon.com has played a vital role in the development of cloud computing. SaaS is a type of cloud computing that delivers applications through a browser to thousands of customers using a multi-user architecture. The focus for SaaS is on the end user as opposed to managed services.
- Salesforce.com is by far the best-known example of SaaS computing among enterprise applications. Salesforce.com was founded in 1999 by former Oracle executive Marc Benioff.
- Another example is Google Apps, which provides online access via a web browser to the most common office and business applications used today, all the while keeping the software and user data stored on Google servers.
- Following are the cloud service providers.

1. Google App Engine

- This is more a web interface for a development environment that offers a one stop facility for design, development and deployment Java and Python-based applications in Java, Go and Python.
- Google offers the same reliability, availability and scalability at par with Google's own applications.

- Interface is software programming based.
- Comprehensive programming platform irrespective of the size (small or large)
- Signature features : Templates and appspot, excellent monitoring and management console.
- The Google App Engine environment includes the following features
  1. Dynamic web serving, with full support for common web technologies.
  2. Persistent storage with queries, sorting, and transactions.
  3. Automatic scaling and load balancing.
  4. APIs for authenticating users and sending email using Google accounts
  5. A fully featured local development environment that simulates Google App Engine on your computer.

2. Amazon EC2

- Amazon EC2 is one large complex web service.
- EC2 provided an API for instantiating computing instances with any of the operating systems supported.
- It can facilitate computations through Amazon Machine Images (AMIs) for various other models.
- Signature features : S3, Cloud Management Console, MapReduce Cloud, Amazon Machine Image (AMI)
- Excellent distribution, load balancing, cloud monitoring tools

3. Windows Azure

- Enterprise-level on-demand capacity builder.
- Fabric of cycles and storage available on-request for a cost.
- You have to use Azure API to work with the infrastructure offered by Microsoft.
- Significant features : Web role, worker role , blob storage, table and drive - storage.

4.9 Securing the Cloud

- The basic security services for information security include assurance of data Confidentiality , Integrity, and Availability
  1. **Confidentiality :** Confidentiality refers to limiting information access. Sensitive information should be kept secret from individuals who are not authorized to see the information.
- In cloud environments, confidentiality primarily pertains to restricting access to data in transit and storage.

2. **Integrity :** This service protects data from malicious modification. When having outsource their data to remote cloud servers, cloud users must have a way to check whether or not their data at rest or in transit are intact. Such a security service would be of the core value to cloud users.

- Integrity can extend to how data is stored, processed, and retrieved by cloud services and cloud-based IT resources.

3. **Availability :** This service assures that data stored in the cloud are available on each user retrieval request. This service is particularly important for data at rest in cloud servers and related to the fulfillment of Service Level Agreement.
4. **Authenticity** is the characteristic of something having been provided by an authorized source. This concept encompasses non-repudiation, which is the inability of a party to deny or challenge the authentication of an interaction.
5. **Threat :** Anything that can exploit vulnerability, intentionally or accidentally, and obtain, damage, or destroy an asset. A threat is what we're trying to protect against. Threat refers to the source and means of a particular type of attack.
6. **Vulnerability :** Vulnerability refers to the security flaws in a system that allows an attack to be successful

4.9.1 Cloud Security Threats

1. Traffic eavesdropping

- Data being passively intercepted by a malicious service agent for illegitimate information gathering purpose while being transferred to or within a cloud

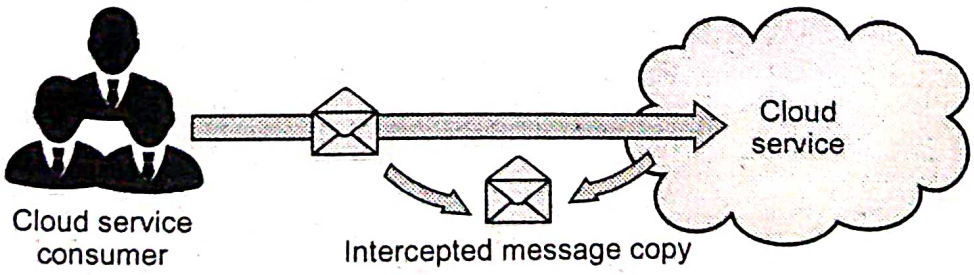

Fig. 4.9.1

- Aim to discredit the confidentiality of data and the relationship between the cloud consumer and cloud provider
- It is hard to detect for a long period of time because of passive nature of the attack.

2. Malicious Intermediary

- Messages intercepted and altered by a malicious service agent discrediting the message's confidentiality and/or integrity
- Possible malicious contents insertion before forwarding it to its destination

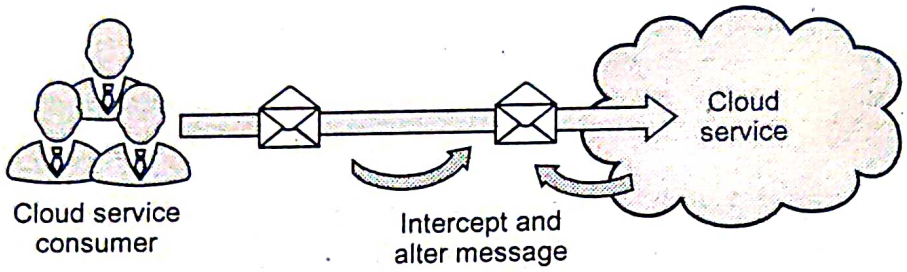

Fig. 4.9.2

3. Denial of Service (DoS)

- Intentional sabotage on shard physical IT resource by overloading it so that the IT resource can hardly be allocated to other consumers sharing the same IT resource
- Typically intentional overloading shared IT resource by generating excessive messages, consuming full network bandwidth, or sending multiple requests that consume excessive CPU time and memory

4. Insufficient authorization

- A case when access is granted to an attacker erroneously or too broadly, resulting in the attacker getting access to IT resources that are normally protected
- Another case **(Weak Authentication)** when weak passwords or shared accounts are used to protect IT resources

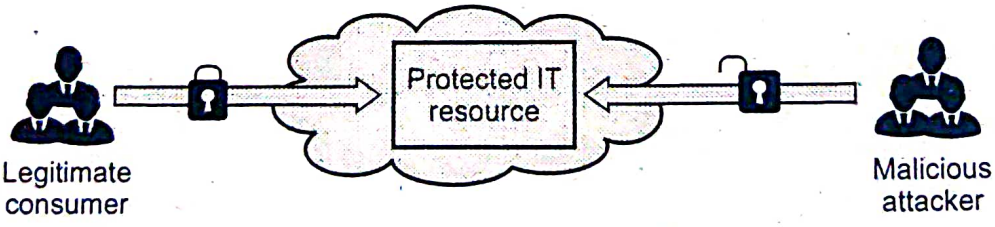

Fig. 4.9.3

5. Virtualization attack (Overlapping Trust Boundaries)

- Physical resources shared by multiple virtual users in virtualized environment by the nature of resource virtualization
- Possible inherent risk that some cloud consumers could abuse their access right to attack the underlying physical IT resources

4.9.2 Single Sign-On (SSO)

- A mechanism enabling one cloud service consumer to be authenticated by a **security broker** which establishes a security context that is persisted while the cloud service consumer accesses other cloud services or cloud-based IT resources in order for the cloud service consumer not to **re-authenticate** itself with every subsequent request.

Implementation mechanisms

- Not a trivial job at all to propagate the authentication and authorization information for a cloud service consumer across multiple cloud services, especially with a numerous cloud services or cloud-based IT resources to be invoked as part of **the same overall runtime activity.**
- SSO (or **security broker**) mechanism to enable mutually independent cloud services and IT resources to generate and circulate **runtime authentication** and **authorization credentials (security token)** in order to allow the credentials provided by the cloud service consumer at its login time to be valid through out the duration of the same session
- Security brokerage mechanism is especially useful when a cloud service consumer needs to access cloud services residing on different clouds.
- Not to counter security threats directly, but **to enhance** the usability of cloud-based environments for access and management of distributed IT resources and solutions without violating security policies

4.9.3 General Security Advantages of Cloud-Based Solutions

1. **Data centralization :** Service provider takes responsibility of storage and small organization need not spend more money for personal storage device.
2. **Incident response :** IaaS providers contribute dedicated legal server which can be used on demand.
3. Forensic image verification time.
4. **Logging :** Storage requirement for benchmark logs is mechanically sloved.

4.10 Introducing Business Continuity and Disaster Recovery

- Business continuity is more proactive and generally refers to the processes and procedures an organization must implement to ensure that mission-critical functions can continue during and after a disaster. BC involves more comprehensive planning geared toward long-term challenges to an organization's success.

- Disaster recovery is more reactive and comprises specific steps an organization must take to resume operations following an incident. Disaster recovery actions take place after the incident, and response times can range from seconds to days.
- Disaster recovery is a piece of business continuity planning and concentrates on accessing data easily following a disaster.
- As cyberthreats increase and the tolerance for downtime decreases, business continuity and disaster recovery gain importance. These practices enable an organization to get back on its feet after problems occur, reduce the risk of data loss and reputational harm, and improve operations while decreasing the cha of emergencies.

Disaster recovery plan

- Disaster recovery plan is a plan designed to recover all the vital business processes during a disaster with in a limited amount of time. This plan has all the procedures required to handle the emergency situations.
- A disaster recovery process should have provable recovery capability, and hence it provides the most efficient method to be adopted immediately after a disaster occurs.
- Mostly the DRP has technology oriented methodologies and concentrates on getting the systems up as soon as possible, within a reasonable amount of time Recovery Time Objective (RTO) and Recovery Point Objective (RPO).
- RTO and RPO are the recovery time objective and recovery point objective, which are the targets of DRP.
- The most successful disaster recovery strategy is the one that will never be implemented; therefore, risk avoidance is a critical element in the disaster recovery process.

Business continuity plan

- Business continuity refers to the activities required to keep the organization running during a period of displacement or interruption of normal operations.
- BCP helps in continuing the business even after a disaster occurs.
- Business has to stay active during the crisis; if it closes its operations even for a day or a week, they are many chances that the organization will experience losses and will have to shut down.
- Moreover, legal issues can arise if the critical services are not provided to clients. This can lead to bad reputation and many more legal problems for an organization in addition to having the pain of being in the state of disaster. Hence an efficient BCP plan can be used to actively run and maintain the business activities.

4.10.1 Differences between Disaster Recovery and Business Continuity Plan

Disaster Recovery Plan	Business Continuity Plan
Main idea : Recover from disasters	Main idea : Continue critical business operations
Disaster recovery is data centric	Business continuity is business centric
DR plan can be built upon a strong business continuity plan.	The business continuity process has a series of DRPs.
Activities are pre-planned to react to disasters	Planning on mitigating risk for the assets, business processes that will adversely impact company, if a disaster happens.
DR plan starts with IT, not because other aspects are not important, but because IT is easiest to recover, and impact is also more.	BC plan is not an IT process; it includes the complete business as a unit.
Disaster recovery is more reactive	Business continuity is more proactive

4.10.2 Understanding the Threats

- The first step in preparing your business for a disaster is identifying the potential threats, such as :
  1. Natural and localized disasters, such as earthquakes, floods, tornadoes, and fires
  2. Failure of IT systems, such as the network, file servers, and software applications
  3. Power outages, such as utility failures
  4. Human threats, such as sabotage, virus attacks, terrorism, and other crimes.
- A business continuity plan takes into account such events and identifies issues that are absolutely critical for maintaining the essential operations to run your business. The plan should also take into account tangible and non-tangible costs your business may need to cover during a failure event.

4.11 Multiple Choice Questions

Q.1 _____ ensures that information is not changed or altered in transit.

a Integrity
b Authentication
c Confidentiality
d Availability