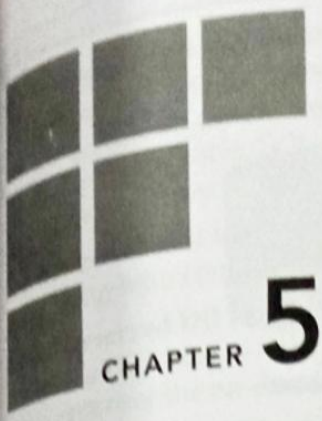

CHAPTER 5

Cloud Databases and File Systems in Cloud

Learning Objectives

After reading this chapter, you will be able to:

- Understand the concepts of cloud database.
- Understand different types of NoSQL databases.
- Understand file system concept.
- Understand technicalities of the Google file system (GFS) and the Hadoop distributed file system (HDFS).
- Understand MapReduce programming model.
- Understand different operations that use MapReduce programming.

5.1 Cloud Database

Cloud database is a database that runs on a cloud computing platform like Amazon EC2, Rackspace and GoGrid.

There are two ways to deploy a database – users can either run the database inside a secured virtual machine (VM) or subscribe for particular database services managed by a cloud service provider. Currently, there are some SQL-based and some NoSQL-based database offerings.

5.1.1 Operation Model for Cloud Database

Figure 5.1 describes two primary methods of running a database on the cloud.

5.1.1.1 VM Image

Cloud platforms allow users to purchase VM instances for a limited time. A cloud provider facilitates more security for running databases inside a VM. If users have their own VM image, then they upload it and run the database inside that or do so through preinstalled databases. Oracle, for example, provides preinstalled image with the Oracle database 11g for Amazon EC2 instances.

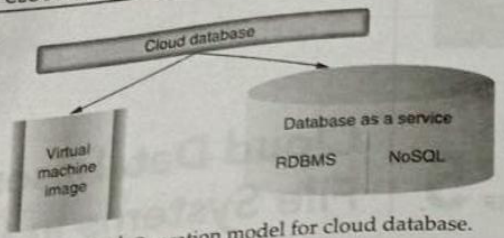

Figure 5.1 | Operation model for cloud database.

5.1.1.2 Database as a Service

Some cloud platform and infrastructure service providers offer database services just as other services offerings, in which case we do not need to launch any instance or individual VM for database installation. All database licensing, updating and configuration are managed by the cloud provider. Application owners have to, each month, pay-per-use of database volume. AWS provides many database services offering to their customers including relational database services (RDS) and NoSQL services such as Amazon RDS, Amazon DynamoDB, Amazon SimpleDB and Amazon Redshift.

The traditional application owner prefers RDS, and in RDS, users have many choices such as MySQL, Oracle, SQL Server or PostgreSQL database engines. All enterprise licensing issue and updates are taken care of by the provider.

5.1.1.2.1 Architectural and Common Characteristics

Following are some architectural and common characteristics of the database as a service offering:

1. Database services provide easy-access Web interface for end users. Database configuration is done by a Web console interface, but some cloud providers offer command line interfaces for managing database operations. AWS, for example, provides Web console interface for launching database instances, taking snapshots, backup and monitoring of database instance.
2. Database services providers offer more flexibility and transparency to users. The service provider is entirely responsible for database installation, patch update and regular maintenances of the respective database. These services offer user transparency during software stack selection, including operating system, database and other third-party software installation.
3. Database services automatically handle high availability and scalability issues. Scalability features differ between cloud providers because some providers offer auto-scaling and some offer API for scale or some offer both, but all giant cloud providers assure 99.9% availability of services.

5.1.2 Types of Cloud Database

It is also important to differentiate between cloud databases that are relational as opposed to non-relational or NoSQL (not only SQL). In this regard, the details of each type of cloud database are discussed in the following subsections.

5.1.2.1 Cloud Relation Databases

Many cloud providers offer RDS nowadays. Some popular and most adopted RDS across the globe are as follows:

1. **Amazon relational database service:** Amazon RDS is very popular and widely adopted Web service. It looks like other AWS services and provides easy management consoles for operating RDS on cloud. Amazon RDS is a highly cost-efficient and secured service. Currently it supports Oracle, SQL Server, MySQL and PostgreSQL database. Amazon RDS specifically offers two types of RDS instances.
   - **On-demand instances:** An on-demand instance offering is a pay-per-use instance with no long-term commitment.
   - **Reserved DB instances:** Reserved DB instances give the flexibility of one-time payment for the DB instance if the database usage is predictable. There is also an offer of 30%–50% price cut over the on-demand price.
2. **Google cloud SQL:** Google cloud SQL is a MySQL database service that is managed by Google, and the entire management, data replication, encryption, security and backups are handled by Google's cloud infrastructure. Google claims maximum availability of its data because its data centers are located across every region of the world. Google cloud is also integrated with the other Google cloud services discussed in Chapter 3. Google cloud SQL is a very flexible, easy-to-use service, which enables connecting and managing cloud SQL with an existing application, just as is done with MySQL.
3. **Heroku Postgres:** Heroku Postgres is a relational SQL database offered by Heroku. It is accessible through all programming languages supported by Heroku. It basically provisioned as an add-on service. Heroku Postgres offers fully reliability of services, which means around 99.99% uptime and 99.999999999% durability of data. One of the advance features of Heroku Postgres is Dataclips, which enables users to send the results of the SQL query via the URL.
4. **HP cloud relational database for MySQL:** HP cloud RDS automate application deployment, configuration management and patch-up task database. It currently supports command line interface (CLI), but an easy-to-use Web-based console interface through APIs is expected soon. It also provides database snapshot facility in multiple availability zones for providing more reliability. It is also built atop an OpenStack-based MySQL distribution, which provides database interoperability from one cloud provider to other.
5. **Microsoft Azure SQL database:** Earlier it was known as SQL Azure. It is the most important component of the Microsoft Azure cloud service; however, it can be operated as a standalone cloud database also. The database can be synched easily with other SQL server databases within the cloud infrastructure of the company or organization. With Microsoft Azure SQL database, the performance of database can be predicted irrespective of whether the service chosen is basic, standard or premium.
6. **Oracle database cloud service:** Oracle database cloud offers two options for users: one is a single schema-based service and another is fully configured Oracle database installed virtual machine. Oracle database can be quickly provisioned, and the user can spin up a database instance with just a few clicks. It also provides flexibility in the management option: self-managed service or fully managed by Oracle.
7. **Rackspace cloud databases:** Rackspace cloud databases are based on open standards. These currently support MySQL, Percona and MariaDB databases. Rackspace cloud provides high database performance using container-based virtualization. It provides automated configuration, which reduces operational costs and team effort. Rackspace cloud is built on top of an open source technology like the OpenStack cloud platform. Rackspace cloud is also connected to SAN storage, which provides built-in data replication for high data replication.

5.1.2.2 Cloud NoSQL Databases

NoSQL database is "not only SQL" database. The evolution of NoSQL database started in early 2009 and has been growing rapidly since because of some limitations with relational databases. NoSQL database is categorized as a non-relational database.

5.1.2.2.1 Limitation with Existing Database

There are certain limitations with our traditional database system and they cannot fit into the current scenario of big data-related application because data is growing exponentially in every industry. Traditional databases are not capable of handling such data growth, which is now in terabytes (TB) and petabytes (PB). Following are some of the key limitations that became the reason behind the birth of NoSQL databases. Traditional databases are unable to:

1. Store data in TB/PB; even a good processor cannot process millions of rows.
2. Process TB of data on a single machine.
3. Be scalable after a certain limit.
4. Provide fault tolerance capability because they have a single point of failure.

5.1.2.2.2 Types of NoSQL Database

There are basically four types of NoSQL databases:

1. **Key-value store:** Based on table keys and values (e.g. AWS DynamoDB).
2. **Document-based store:** Document-based database stores records that are made of tagged elements (e.g. MongoDB, CouchDB).
3. **Column-based store:** Data divided into multiple columns and every storage block contains data of each column (e.g., Apache HBase, Cassandra).
4. **Graph-based store:** A network graph storage that uses edges and nodes for storing data (e.g. Neo 4j).

Following are the most popular and widely adopted NoSQL databases:

1. **Amazon DynamoDB:** Dynamo DB is one of the most popular NoSQL databases of Amazon, which is based on the key-value data store. It was first developed for an internal purpose like handling customer-related big data applications. DynomoDB is built on the top of a single solid device (SSD) architecture; therefore, it scales according to the data in the system. It provides very high-quality reliable throughput and single-digit millisecond latency, which is preferred in building fast gaming and analytics-based applications.
2. **MongoDB:** MongoDB is one of the most popular and fast growing NoSQL databases. It can be managed easily on a cloud infrastructure; therefore, various hosted MongoDB services are available. MongoDB is written in C++ and comes under the category of open source document databases, which belongs to the NoSQL database family. Using MongoDB search queries can be done by fields, range queries and regular expression.
3. **Apache Cassandra:** DataStax is the leading commercial company behind Cassandra. Founded by the Apache chairman of the Cassandra project, DataStax employs most of the project's committers and provides an enterprise-ready and tested version of Cassandra in its Edition 4.5. Cassandra delivers fast performance, high availability and scalability with dynamic schema. It is designed specifically for handling real time data analytics-related high workload without a single point of failure. It offers high data availability and quick scalability to many enterprises.
4. **CouchDB:** CouchDB is one of the most appreciated projects of the Apache Foundation. The term "couch" represents "cluster of unreliable commodity hardware." CouchDB NoSQL database is completely Web oriented. It stores the data in the JavaScript Object Notation (JSON) document format, which can be accessed using the JSON API, and query of index can be done with the help of Web browsers. CouchDB works for all types of Web and mobile apps. Also, the data of the Web applications can be replicated and distributed using incremental policies of CouchDB. It offers great reliability, availability and scalability as well.
5. **Apache HBase:** Apache HBase is mostly used in Hadoop-related projects (Hadoop is explained in Section 5.2.3). HBase is another sub-project of the Apache Foundation, which is strongly associated with the Hadoop ecosystem. HBase is used to write and update data in real time and can be placed as the traditional database. It can contain as many columns for a row key as required; however, there can be exactly one value for per row for every column. HBase also provides various APIs which supports many operations using application programming languages.
6. **Neo 4j:** Neo 4j is the most popular graph database used in various mission critical applications. It is the preferred choice of many start-ups and mid-size organizations across the globe. Its highly robust and scalable database comprises ACID (Atomicity, Consistency, Isolation, Durability) properties. Because Neo 4j provides high-availability clustering with a graph query language, the customer can easily import the data into the CSV format. It also supports advance monitoring.

5.2 Cloud File System

As we have already learned in Chapter 1, cloud is an advanced concept of distributed computing; therefore, to understand cloud file system, we must know that what distributed file system is.

5.2.1 Distributed File System Basics

Distributed file system (DFS) is basically used for storing huge amount of data and provides accessibility of stored data to all distributed clients across the network. The objective of the DFS is to provide a system for all the geographically distributed users as a common file system for data sharing and storage.

DFS comprises various software components that run as a single system entity on multiple systems. An Internet search engine is the most common example of DFS, which is used for indexing millions of Web pages. There are a number of DFS that solve this problem in different ways. Some popular file systems are:

1. Andrew file system (AFS)
2. Network file system (NFS)
3. Coda

4. Microsoft distributed file system (DFS)
5. Apple filing protocol (AFP)
6. Google file system (GFS)
7. Hadoop distributed file system (HDFS)

Of all the file systems listed earlier, NFS is the most commonly adopted DFS. It grants remote access to a logical volume that resides on a single machine and makes some segment on its local file system which provides accessibility to different distributed clients. Mounting of a remote file system can be done easily with the client's file system and can be viewed as a local drive.

NFS is one of the oldest file systems because of which it has some limitation as well. Here all the data resides on one machine, so it does not provide much reliability of data because it has only a single point of failure. NFS generally experiences the power of overloading if large numbers of clients access data simultaneously. To handle these challenges GFS and HDFS follow different approach.

5.2.2 Concept of GFS

Google invented and implemented a scalable DFS to handle their huge internal distributed data exhaustive applications and named it the Google File System. In 2002–03, Google launched its file system based on DFS architecture but added some advance features that are driven by Google's unique workload and environment.

5.2.2.1 Google File System Architecture

A cluster of a Google file system contains a single master and multiple chunk servers that are associated with many clients. The master holds the metadata of chunk servers. All the data processing happens through these chunk servers. The client first contacts the master and retrieves the metadata of the chunk server, which is then stored in the chunk servers. So the next time, client directly connects to the chunk servers. Figure 5.2 describes the GFS architecture.

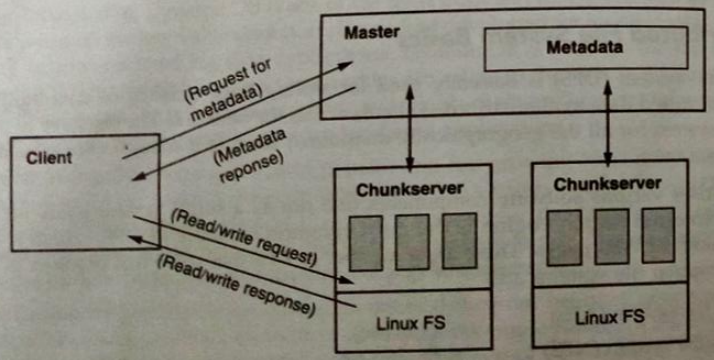

Figure 5.2 | GFS architecture.

following are the details of each component of GFS:

1. **Chunk:** A chunk is very similar to concept of block in a file system, but chunk size is larger than the traditional file system block. The block of chunk is 64 MB. This is specifically designed for the Google environment.
2. **Master:** Master is a single process that runs on entirely separate machine for security purposes. It only stores metadata-related information, chunk location, file mapping information and access control information. The client first contacts the master for information about metadata and then connects to that particular chunk server.
3. **Metadata:** Metadata is stored in the memory of a master; therefore, master operations are much faster. Metadata contains three types of information:
   - Namespaces of file and chunk
   - Location of each chunk
   - Mapping from file to chunk

5.2.3 Concept of HDFS

HDFS is a DFS based on GFS that provides high throughput access to application data. It uses the commodity hardware with the expectation that failures will occur and provides portability across heterogeneous hardware and software platforms. HDFS is acquired by Hadoop Apache open source project, which is very popular these days for its ability to handle big data.

The Hadoop core consists of two modules:

1. **Hadoop distributed file system:** Used for storing huge amount of data.
2. **MapReduce programming mode:** Used for processing of large set of data.

Hadoop provides a DFS, called HDFS, and a framework for the analysis and transformation of very large data sets using the MapReduce paradigm. HDFS is designed to store very large data sets (terabytes or even petabytes) reliably and to stream those data sets to high bandwidths for user application. Hadoop is an open source implementation of the MapReduce programming and HDFS written in Java. Files are stored in a redundant fashion across multiple machines to ensure their durability to failure and high availability to very parallel applications. That is why Hadoop is very useful in a cloud environment; it can easily process vast amounts of data. HDFS supports the cloud environment because of the following reasons:

1. HDFS is designed to store vast amounts of information in terabytes or petabytes, which means spreading the data across a large number of machines. It also supports much greater file sizes than does the NFS.
2. HDFS provides data reliably. If individual machines in the cluster malfunction, data will be available due to its replication policy.
3. HDFS provides fast accessibility to retrieve stored information. Scalability can be achieved any time by just adding more machines to the existing cluster and machines can be removed easily anytime.
4. HDFS integrates with MapReduce programming model easily, which allows data to be read and computed on system locally.

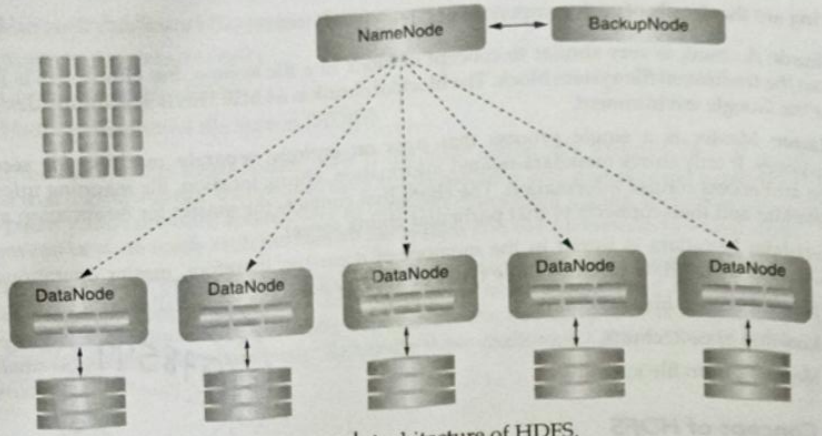

Figure 5.3 | Architecture of HDFS.

5.2.3.1 Architecture of HDFS

The working architecture of HDFS is almost similar to GFS. Figure 5.3 shows details about the architecture of HDFS.

An HDFS cluster consists of multiple commodity machines that can be classified into the following three types:

1. Name node (runs on master machine)
2. Secondary name node or backup node (runs on separate machine)
3. Data node (runs on slave machine)

The working of an HDFS is the same as master slave architecture. Here the master is the name node that contains the metadata of the cluster, but the processing occurs through data nodes. The client first connects to the metadata and receives information about the data node and the next time directly connects to the data node. GFS works the same way as well. The different colors in Figure 5.2 represent data replication across the cluster. By default the replication factor is three times, but it can be adjusted to higher or lower levels also; therefore, the fault tolerance feature is automatically enabled with HDFS.

5.2.3.2 Features of HDFS

Following are some technical features of HDFS:

1. **Durability:** Replication of each block of file thrice on different data nodes makes the cluster robust against data loss due to node failure. It is unlikely that Yahoo! has ever lost any data like that. The probability of losing a block is less than 0.005 in a year.
2. **Placement policy:** For a huge HDFS architecture deployment, nodes are replicated across the multiple racks and one common switch is shared among all nodes that are connected

by two or more switches. The HDFS placement policy also ensures that there are no more than two replica of the same block in the same rack.

3. **Replication management:** In HDFS, whenever a replica becomes under-replicated, it is managed by the replication priority queue. Replication management handles priority issues of replica, if there is only one replica of one block then this replica would have the highest priority over other blocks that may have two or three replications.
4. **Load balancing:** The balancer is a default application program deployed on HDFS. Disk space utilization is balanced across HDFS with the help of this program.
5. **Data integrity:** The data integrity is provided by the use of checksum. When a client writes a block to the HDFS, checksum is calculated for the block and saved along with the data. When the same block is read in the future, the respective checksum is checked and if there is a mismatch, an error is thrown and NameNode is notified. The corrupted data block is then replaced with the correct one immediately.

5.2.4 Comparison of Features

Some key differences between HDFS and GFS are given in Table 5.1. The table includes both similarities and differences.

Table 5.1 Comparison between HDFS and GFS

Features	GFS	HDFS
Architecture	Clustered based, asymmetric parallel, object based	Clustered based, asymmetric, parallel, object based
Process	Stateful	Stateful
Communication	RPC/TCP	RPC/TCP and UDP
Naming	Central metadata server	Central metadata server
Synchronization	Follows write-once-read-many policy, various producers but only one consumer	Follows write-once-read many policy,
	Provides locks on objects to clients	Provides locks on objects to clients
Consistency and replication	Replication on server side, checksum policy and replication of data objects	Replication on server side, checksum policies and replication of data objects
Fault tolerance	Failure an exception	Failure may be expected
Security	No dedicated security mechanism	No dedicated security mechanism
Read rate	6 MB/s per client	1.02 MB/s per node
Write rate	2.2 MB/s per client	1.02 MB/s per node
Garbage collection	Yes	No
Inter-cluster data copying	No	Yes
Data integrity	Checksum	Checksum