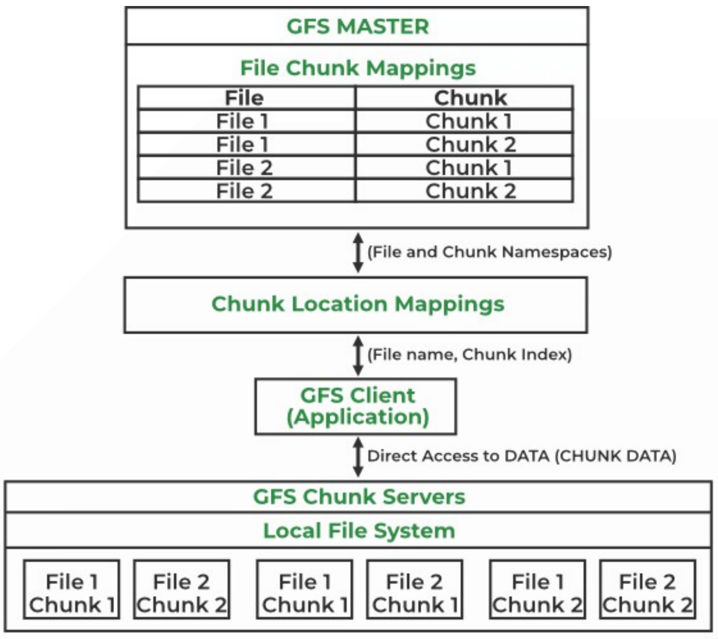
4.Data Storage & Security Cloud

1. Google File System:-

- Google File System (GFS), also called GoogleFS, is a special system created by Google to store and manage very large amounts of data across many computers.
- It is designed to be fast, reliable, and able to keep working even if some parts fail.
- GFS uses cheap hardware and splits large files into big chunks (64 MB each), storing them on several computers (called chunk servers). Each chunk is saved in three copies to prevent data loss.
- One main computer (called the master) keeps track of where all the pieces of data are stored and manages things like file names and permissions.
- The master checks in with the chunk servers regularly to make sure everything is working.
- GFS supports thousands of computers and hundreds of users at the same time, storing huge amounts of data (like 300 TB) and allowing easy and quick access to it.



Components of GFS

A group of computers makes up GFS. A cluster is just a group of connected computers. There could be hundreds or even thousands of computers in each cluster. There are three basic entities included in any GFS cluster as follows:

1. GFS Clients: These are programs or apps that ask for files – they can read, change, or add new files in the system.

2. **GFS Master Server:** This is the main controller. It keeps track of what's happening in the system and stores important info (called metadata) about where each piece of a file is and which file it belongs to.

3. **GFS Chunk Servers:** These are the machines that store the actual pieces of files, each 64 MB in size. They don't send data to the master but send it straight to the client. Each file piece is saved in three copies on different chunk servers to make sure nothing is lost if one server fails.

Features of GFS

- Namespace management and locking.
- Fault tolerance.
- High availability.
- Critical data replication.
- Automatic and efficient data recovery.
- High aggregate throughput.

Advantages of GFS

1. High accessibility Data is still accessible even if a few nodes fail.
2. Excessive throughput. many nodes operating concurrently.
3. Dependable storing.

Disadvantages of GFS

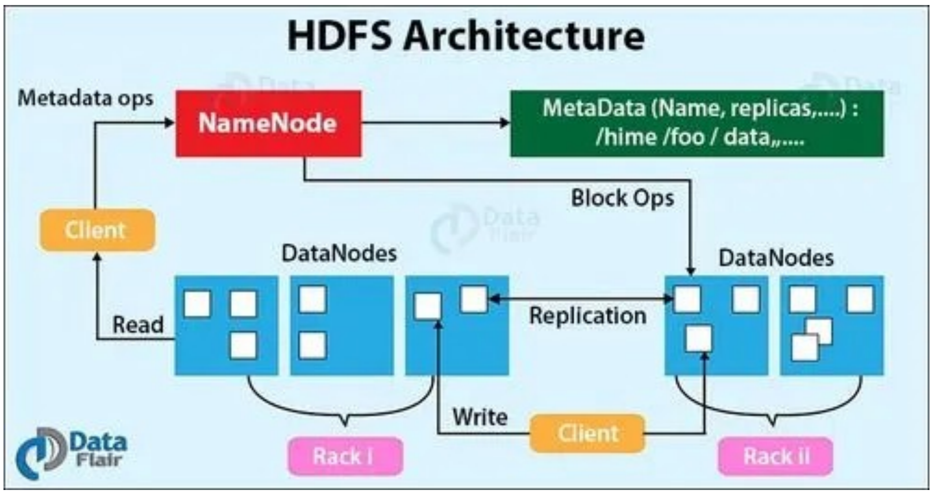
1. Not the best fit for small files.
2. Master may act as a bottleneck.
3. unable to type at random.
4. Suitable for procedures or data that are written once and only read (appended) later.

1.Hadoop Distributed File System(HDFS):-

- With growing data velocity the data size easily outgrows the storage limit of a machine. A solution would be to store the data across a network of machines. Such filesystems are called **distributed filesystems**.

- Hadoop is provides one of the most reliable filesystems. **HDFS (Hadoop Distributed File System)** is a unique design that provides storage for extremely large files with streaming data access pattern, and it runs on commodity hardware.

  - **Extremely large files**: Here, we are talking about the data in a range of petabytes (1000 TB).
  - **Streaming Data Access Pattern**: HDFS is designed on principle of write-once and read- many-times. Once data is written large portions of dataset can be processed any number times.
  - **Commodity hardware:** Hardware that is inexpensive and easily available in the market. This is one of the features that especially distinguishes HDFS from other file systems.



Nodes: Master-slave nodes typically form the HDFS cluster.

1. **NameNode(MasterNode):**
    - Manages all the slave nodes and assigns work to them.
    - It executes filesystem namespace operations like opening, closing, and renaming files and directories.
    - It should be deployed on reliable hardware that has a high configuration. not on commodity hardware.
2. **DataNode(SlaveNode):**
    - Actual worker nodes do the actual work like reading, writing, processing, etc.
    - They also perform creation, deletion, and replication upon instruction from the master.
    - They can be deployed on commodity hardware.

HDFS daemons: Daemons are the processes running in the background.

- **Namenodes:**
    - Run on the master node.
    - Store metadata (data about data) like file path, the number of blocks, block Ids. etc.
    - Requires a high amount of RAM.
    - Store meta-data in RAM for fast retrieval i.e to reduce seek time. Though a persistent copy of it is kept on disk.
- **DataNodes:**
    - Run on slave nodes.
    - Require high memory as data is actually stored here.

Features:

- Distributed data storage.
- Blocks reduce seek time.
- The data is highly available as the same block is present at multiple datanodes.
- High fault tolerance.

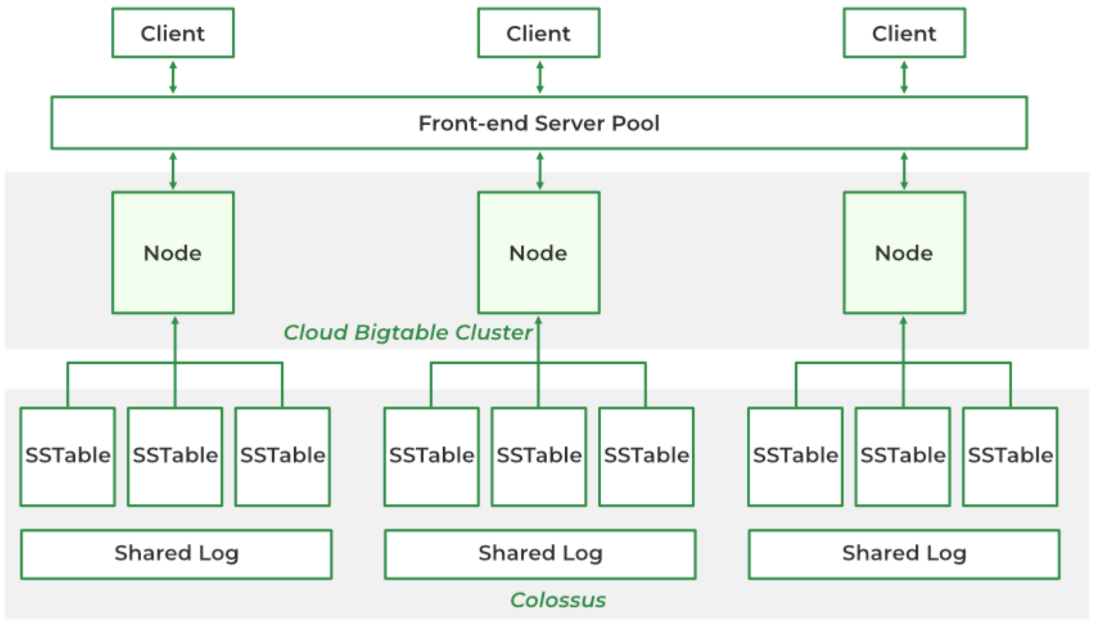
Limitations:

- Low latency data access:
- Small file problem:

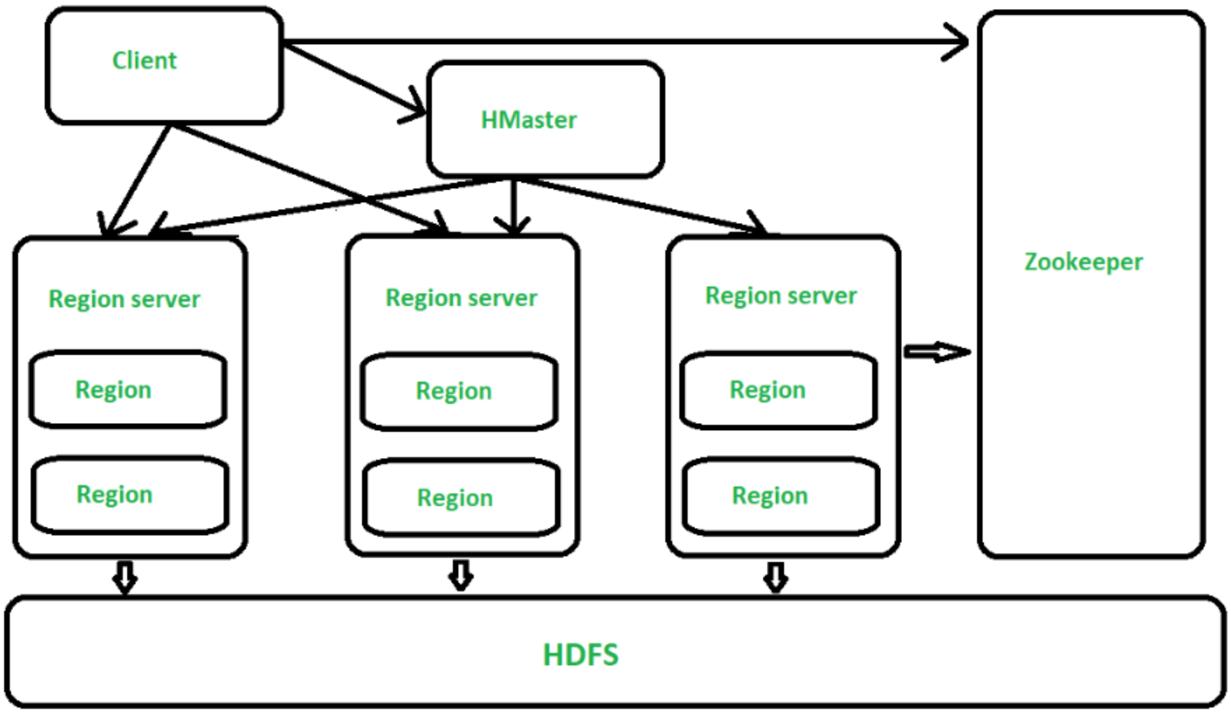
3.Big Table and HBase of cloud computing:-

Feature	Bigtable	HBase
Developer	Bigtable is developed and maintained by Google.	HBase is developed and maintained by the Apache Software Foundation.
Platform	Bigtable runs on the Google Cloud Platform as a fully managed service.	HBase runs on the Hadoop ecosystem and can be deployed on-premises or in the cloud.
Type	Bigtable is a proprietary database provided by Google.	HBase is an open-source database that is freely available.
Storage Engine	Bigtable stores its data using Google Cloud Storage.	HBase stores its data in the Hadoop Distributed File System (HDFS).
Scalability	Bigtable automatically scales according to the workload.	HBase requires manual configuration and tuning for scaling.
Ease of Use	Bigtable is easy to use with built-in cloud features and minimal setup.	HBase requires complex setup and ongoing maintenance.
Integration	Bigtable integrates well with other Google Cloud services like BigQuery.	HBase integrates well with Hadoop tools like Hive, Pig, and MapReduce.
Use Case	Bigtable is ideal for cloud-native applications requiring scalability.	HBase is suitable for custom big data applications running on Hadoop.
Performance	Bigtable offers high and consistent performance with Google infrastructure.	HBase performance depends on cluster setup and tuning.
Working	Bigtable works by storing data in a sparse, distributed, and sorted map, where data is indexed by row key, column key, and timestamp.	HBase works similarly, using tables with rows and columns, and stores data in HFiles on HDFS, using region servers to manage data

Bigtable



HBase



4. DynamoDB:-

Amazon DynamoDB is a fully managed NoSQL database service provided by Amazon Web Services (AWS).

It is designed to handle large-scale applications that need:

- Fast and consistent performance
- Scalability
- Low-latency data access

Instead of using traditional table-based relational models, DynamoDB uses key-value and document-based data models, making it ideal for modern applications like:

- Real-time analytics
- IoT (Internet of Things)
- Mobile and web apps
- Gaming platforms

It automatically handles tasks like data replication, backup, security, and scaling, so developers can focus on building applications without worrying about database infrastructure.

Features of DynamoDB:

1. **Fully Managed Service**
    - AWS handles all database administration like hardware provisioning, setup, configuration, patching, and backups.
2. **NoSQL Database**
    - DynamoDB uses a **key-value and document data model**, not tables with fixed schemas like traditional databases.
3. **High Scalability**
    - It automatically scales up or down based on the size of the data and traffic without any manual effort.
4. **Low Latency Performance**
    - DynamoDB can serve **single-digit millisecond response times**, which is great for real-time applications.
5. **Built-in Security**
    - It integrates with AWS Identity and Access Management (IAM), encryption at rest and in transit, and fine-grained access control.

6. **Automatic Backups and Recovery**
   - It supports **on-demand backup and restore**, as well as **point-in-time recovery** for data protection.
7. **Global Tables**
   - Allows **multi-region replication**, enabling globally distributed applications with low latency.
8. **Event-Driven Programming**
   - Works well with AWS Lambda to trigger automatic functions when data changes.
9. **Time-to-Live (TTL)**
   - You can automatically delete expired data to save storage and improve performance.

Advantages of DynamoDB:

1. **Highly Available and Durable**
   - Data is automatically replicated across multiple availability zones in a region for high availability.
2. **Serverless Architecture**
   - No need to manage servers; you only pay for what you use.
3. **Flexible Schema**
   - You can store different attributes in different items of the same table, unlike traditional databases.
4. **Real-Time Data Access**
   - Ideal for applications that require quick reads and writes.
5. **Integration with AWS Ecosystem**
   - Easily integrates with AWS services like Lambda, API Gateway, CloudWatch, etc.

5. Differentiate Between SQL and NoSQL:-

Aspect	SQL (Relational Database)	NoSQL (Non-Relational Database)
Data Model	SQL databases use a **structured tabular format** with rows and columns.	NoSQL databases use **key-value, document, graph, or wide-column formats**.
Schema	They have a **fixed schema**, meaning the structure must be defined before inserting data.	They have a **flexible schema**, allowing data to be added without a predefined structure.
Scalability	SQL databases are usually **vertically scalable** (add more power to one server).	NoSQL databases are generally **horizontally scalable** (add more servers to handle load).
Query Language	SQL databases use **Structured Query Language (SQL)** for defining and manipulating data.	NoSQL databases use **varied query methods**, often specific to the database type.
Transactions	They support **ACID (Atomicity, Consistency, Isolation, Durability)** transactions.	NoSQL supports **eventual consistency**; ACID compliance is limited or optional.
Use Cases	Ideal for applications with **structured data and complex queries** like banking or ERP.	Suitable for **big data, real-time apps, and unstructured data** like social media or IoT.
Examples	MySQL, PostgreSQL, Oracle, Microsoft SQL Server.	MongoDB, Cassandra, DynamoDB, Redis, Couchbase.
Joins and Relationships	SQL supports **complex joins and relationships** between tables.	NoSQL **avoids joins** and instead uses **denormalized or nested data** formats.

6. Disaster in Cloud Computing:-

- A disaster in cloud computing refers to any unexpected event that causes disruption to cloud-based services, data loss, or infrastructure failure.

- These events can be natural (like earthquakes or floods) or man-made (like cyber attacks or system failures).

- Disasters can affect the availability, integrity, and security of cloud resources, and may lead to service downtime, data corruption, or permanent loss if not properly managed.

- That's why disaster recovery strategies are crucial in cloud environments.

Types of Disasters / Threats in Disaster Recovery:-

1. **Natural Disasters**
   These include earthquakes, floods, or hurricanes that damage data centers and physical infrastructure, affecting cloud service availability.

2. **Hardware Failures**
   Occurs when physical devices like servers, hard drives, or routers fail, disrupting the performance of cloud-hosted applications.

3. **Software Failures**
   Software bugs, corrupted files, or system crashes can stop applications or cause data inconsistencies.

4. **Cyber Attacks**
   Attacks such as hacking, ransomware, or DDoS can steal or lock data, making services unavailable or insecure.

5. **Human Errors**
   Mistakes like deleting files or misconfiguring cloud settings can unintentionally break systems or cause downtime.

6. **Power Outages**
   A sudden power failure in a data center can bring cloud services offline if no backup systems are in place.

7. **Network Failures**
   Internet or internal network outages can cut access to cloud applications and data for users.

Disaster Recovery on Cloud Platform:

Disaster Recovery (DR) is the process and set of procedures used to **restore IT systems, data, and services** after a disaster occurs. The goal is to **minimize downtime** and **prevent data loss** so the business can continue running smoothly.

1. **Data Backup**
   Cloud providers store regular backups in multiple locations, so data can be restored quickly if lost.

2. **Geo-Redundancy**
   Data and applications are replicated in different regions to ensure availability even if one location fails.

3. **Automated Failover**
   If one server or service fails, traffic is automatically redirected to another healthy system with no manual intervention.

4. **Snapshot and Versioning**
   Cloud storage keeps multiple versions of data, allowing easy rollback to previous states in case of errors or corruption.

5. **Disaster Recovery as a Service (DRaaS)**
   Cloud platforms like AWS, Azure, and Google Cloud offer tools to automate and manage disaster recovery processes.

6. **Monitoring and Testing**
   Regular monitoring and disaster recovery drills ensure systems are always ready and downtime is minimized.

7. General Security:-

- General security in cloud-based solutions refers to the overall practices, technologies, and measures used to **protect data, applications, and services** hosted on cloud platforms.

- It ensures that cloud resources are safe from unauthorized access, data breaches, and other cyber threats.

- This includes **controlling who can access the cloud**, **protecting data during storage and transmission**, and **keeping systems updated** to defend against new vulnerabilities — all to maintain privacy, integrity, and availability of cloud services.

General Security Advantages of Cloud-Based Solutions:-

1. **Centralized Security Management**
   Cloud providers manage security centrally, making it easier to update and enforce security policies across all users and systems.

2. **Advanced Security Tools**
   Cloud platforms offer strong security tools like encryption, firewalls, and intrusion detection that might be too expensive for individual organizations.

3. **Regular Security Updates**
   Cloud providers constantly update their systems to fix vulnerabilities and protect against the latest threats.

4. **Data Backup and Recovery**
   Cloud services automatically back up data and offer easy recovery options, reducing the risk of data loss.

5. **Access Control and Identity Management**
   Cloud solutions provide robust methods to control who can access data and applications, using technologies like multi-factor authentication.

6. **Scalability of Security**
   Security measures can scale easily with the growth of users and data without extra hardware or complex setups.

7. **Compliance Support**
   Many cloud providers comply with international security standards and regulations, helping businesses meet legal requirements.

Security Issues in Cloud Computing:-

1. **Data Breaches**
   Unauthorized access to sensitive data stored in the cloud can lead to loss or theft of personal and business information.

2. **Data Loss**
   Accidental deletion, hardware failure, or disasters can cause permanent loss of data if proper backups aren't maintained.

3. **Account Hijacking**
   Attackers may steal user credentials to gain unauthorized access to cloud services and misuse resources or data.

4. **Insider Threats**
   Employees or insiders with access to cloud data might intentionally or accidentally misuse or leak information.

5. **Insecure APIs**
   Cloud services use APIs to communicate; if these are poorly designed or unsecured, attackers can exploit them to gain access.

6. **Denial of Service (DoS) Attacks**
   Attackers can overload cloud services with excessive requests, making the services unavailable to legitimate users.

7. **Lack of Compliance**
   Cloud providers and users must comply with legal and regulatory standards; failure to do so can lead to security risks and penalties.

8. **Data Privacy Issues**
   Storing data in the cloud raises concerns about who can access or control the data, especially across different countries.

9. **Shared Technology Vulnerabilities**
   Since cloud resources are shared among users, vulnerabilities in the shared infrastructure can lead to security risks.

8. Expalin:-

1. Business Continuity:-

- **Business Continuity** means making sure that a company can keep operating smoothly even during unexpected problems like disasters, cyberattacks, or system failures.
- It involves planning and preparing so that critical services and data remain available without major interruption.

How to Approach Business Continuity in Cloud

1. **Risk Assessment**
   Identify potential threats and weaknesses that could impact cloud services and business operations.
2. **Data Backup and Replication**
   Regularly back up data and replicate it across multiple cloud locations to prevent loss.
3. **Disaster Recovery Plan**
   Develop and test a clear plan for quickly restoring cloud systems and data after a disruption.
4. **Use of Multiple Cloud Providers**
   Avoid relying on a single cloud provider by using multiple clouds or hybrid solutions to increase reliability.
5. **Continuous Monitoring**
   Monitor cloud resources and security continuously to detect and respond to issues early.
6. **Regular Testing and Updates**
   Test business continuity plans often and update them to address new risks or changes in technology.

2. Architect for Failure

- **Architect for Failure** means designing cloud systems with the expectation that parts of the system will fail at some point.
- Instead of trying to prevent all failures (which is impossible), you build your applications and infrastructure so they can **handle failures gracefully** without stopping the whole service.

Key ideas include:

- Use **redundancy** (multiple servers or data centers) so if one fails, others take over.
- Design **automatic recovery** mechanisms to detect failures and fix them quickly.
- Use **stateless services** where possible, so no single component holds critical information.
- Test your system with **failure simulations** to find weaknesses and improve resilience.

3. Fault Tolerance:-

- **Fault Tolerance** is the capability of a system, network, or application to continue functioning correctly even when one or more of its components fail.
- Instead of crashing or stopping when errors occur, a fault-tolerant system detects problems and either fixes them automatically or switches to backup components to maintain uninterrupted service.
- Fault tolerance is critical in environments like cloud computing, where continuous availability and reliability are essential.
- It ensures that failures—whether due to hardware malfunctions, software bugs, or network issues—do not cause the entire system to fail.

Characteristics of Fault Tolerance

1. **Reliability**
   The system keeps working without failure for a long time, even if some parts fail.
2. **Redundancy**
   Extra components or backup systems are included to take over when primary ones fail.
3. **Error Detection**
   The system can identify faults quickly to prevent them from causing bigger problems.
4. **Error Recovery**
   It can automatically correct or work around errors to maintain normal operation.
5. **Continuous Operation**
   There is no or minimal downtime, so users don't experience service interruptions.
6. **Graceful Degradation**
   If failure happens, the system reduces functionality but doesn't stop completely.

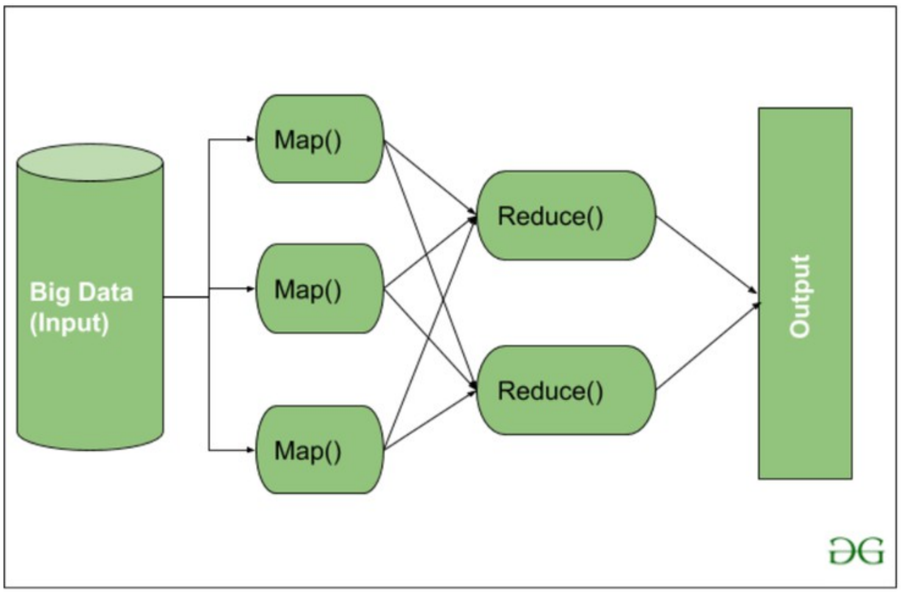
9. Map Reduce in Hadoop:-

• One of the three components of Hadoop is Map Reduce. The first component of Hadoop that is, Hadoop Distributed File System (HDFS) is responsible for storing the file. The second component that is, Map Reduce is responsible for processing the file.

• MapReduce has mainly 2 tasks which are divided phase-wise.In first phase, **Map** is utilised and in next phase **Reduce** is utilised.

• **MapReduce** is required because it provides a simple and efficient way to process this big data in a **distributed** manner.

• It breaks the job into smaller tasks that run in parallel on different nodes, which speeds up processing and makes handling huge data practical. It also handles failures and ensures that the process is reliable and scalable.



Stages of MapReduce

1. **Input Splitting**
   The large input data is split into smaller chunks called input splits. Each split is processed by a separate map task.

2. **Map Stage**
   Each map task processes its input split and converts the data into key-value pairs. The Mapper applies a user-defined function to extract relevant data.

3. **Shuffle and Sort**
   The output from the map tasks (key-value pairs) is shuffled and sorted by keys. This step groups all values related to the same key together to prepare for the reduce stage.

4. **Reduce Stage**
   The Reducer processes each group of key-value pairs, applies a user-defined function to aggregate, summarize, or transform the data, and produces the final output.

5. **Output**
   The result of the reduce stage is written to output files, usually stored in a distributed file system like HDFS.

Example: Word Count

Suppose you want to count the occurrences of each word in a large text document.

- **Input:** Large text file split into chunks.
- **Map:** Each map task reads its chunk and outputs `(word, 1)` for every word found.
- **Shuffle and Sort:** The system groups all `(word, 1)` pairs by word.
- **Reduce:** The reducer sums the counts for each word key to get `(word, total_count)`.
- **Output:** A list of words with their total counts in the document.