Cloud Computing III

UNIT III - CLOUD STORAGE

▼ Understanding data storage

- **File storage** - NAS, DAS
- **Block storage** - Arbitrarily organised, evenly sized volumes. SAN environments. Limited capability to handle metadata - dealt with in application or db level.
- **Object storage** - work as modular units. Storage system uses metadata and identifiers.

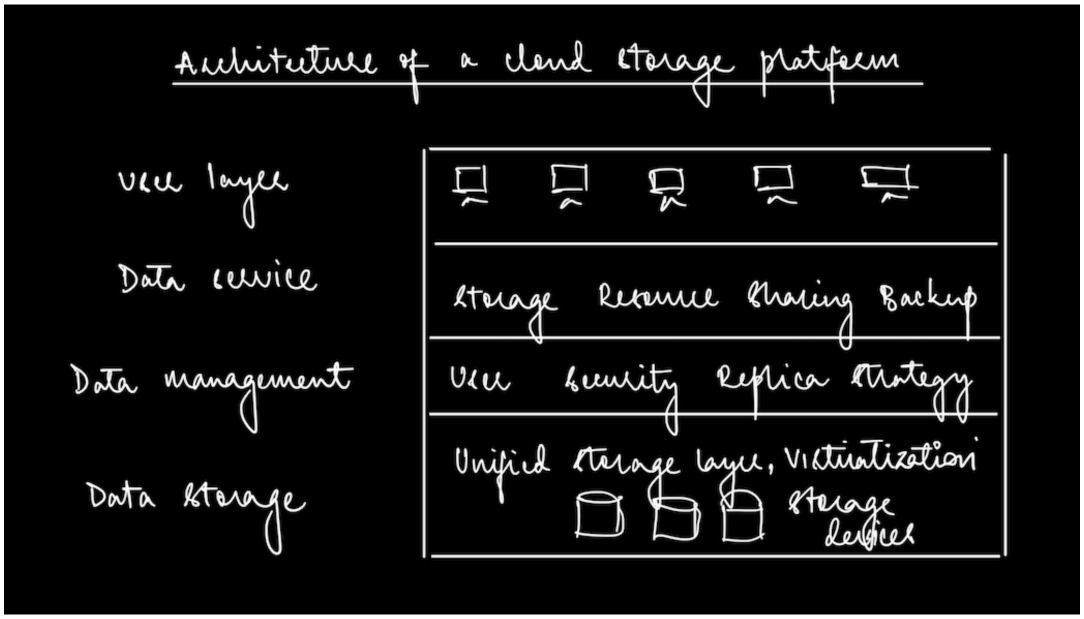
▼ Cloud storage

Data storage model in which data is maintained, managed and backed up remotely and made available to users over a network.

web browser → http server (load balanced) → applications → data organisation → storage devices

▼ Architecture of a cloud storage platform

1. User access layer - authorised user can login to cloud storage via standard public interface.
2. Data service layer - deals directly with users and user demands
3. Data management layer - unified public management for different services.
4. Data storage layer - data stored in system forms a pool to be organised.


Architecture of a cloud storage platform
User layer
Data service
Data management
Data storage
Storage Resource Sharing Backup
User Security Replica Strategy
Unified storage layer, Virtualization
Storage devices

Pros: Dramatic reduction in TCO, Unlimited scalability, Elasticity, on-demand, Universal access, Multi tenancy, Data durability and availability, Usability and Disaster recovery.

▼ Storage virtualisation

Storage virtualisation is a means through which physical storage subsystems are abstracted from the user's application and presented as logical entities, hiding the underlying framework and complexity of the storage systems and nature of access, network or changes to the physical devices.

▼ Enablers for Storage virtualisation

1. **File systems** - structure and logic rules used to manage groups of information
2. **LVMs (Logical Volume Managers)** - independent layer between the file system and the disk drives.

3. **Virtual provisioning** - Thinly provisioned logical units

Categories of Storage Virtualisation

- ▼ File level Virtualisation
  - ▼ Distributed file system with centralised metadata

    Dedicated metadata server. Lock based synchronisation. Metadata server can become a bottle neck if there are too many metadata operations.

    Lustre

    Massively parallel, scalable dfs for Linux which employs a cluster based architecture with centralised metadata. Three main functional components include -

    1. OSS (Object storage servers) that store file data on OSTs (Object storage targets)
    2. Single MDT (meta data target) that stores meta on MDS (meta data servers)
    3. Lustre clients that access data using a POSIX interface

  - ▼ Distributed file system with distributed metadata

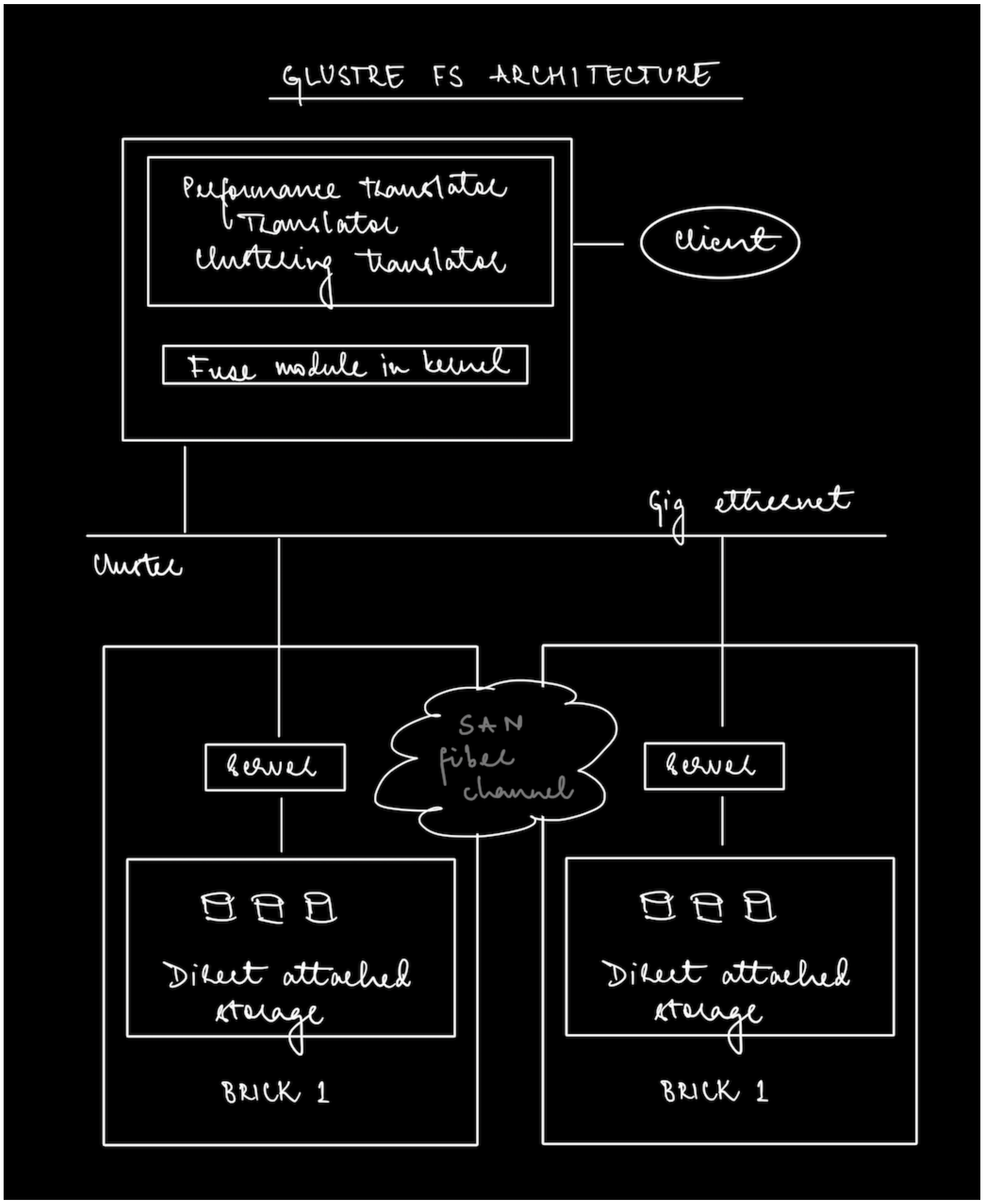
    meta data is distributed across all nodes in the system. greater complexity.

    Gluster

    Open source, distributed cluster file system with a modular architecture and stackable user space design.

    Aggregates multiple storage bricks on a network and delivers as a NFS with a global namespace.

    Consists of Clients and Servers.



Gluster FS uses the concept of a storage brick consisting of a server that is attached to storage directly or through a SAN. Gluster employs a mechanism called translators to implement the filesystem capabilities. Translators are inserted between the actual content of a file and the user accessing the file as a basic file system interface. Each translator implements a particular feature of GlusterFS. Translators can be loaded into both servers and clients to improve functionalities.

Gluster also performs very good load balancing using IO Scheduler translators.

Also supports file replication with AFR automatic file replication translator which keeps identical copies of a file on all its sub volumes.

▼ Block Virtualisation

1. **Host based** - LVM supports creation of a storage pool. Allows transparent allocation to dynamically expand with data volumes.
2. **Storage device** - Host agnostic and low latency, virtualization is a part of storage device itself. Uses RAID techniques.
3. **Network level virtualisation** -
    - Switch based network virtualisation (intelligent switch in conjunction w/ metadata manager)
    - Appliance based network virtualisation (appliance controls virtualisation layer)

▼ Storage examples

▼ Amazon S3

- Data is stored as objects.
- Objects are stored in resources called buckets.
- Objects are referred to with keys - optional directory path name

Data protection:

- Replications - 2 replica failures. Reduced redundancy storage.
- Regions - bucket level locations
- Versioning - stores full history of all objects
-

▼ Open stack Swift

- Swift partitions - core of replication system
- Account - user
- Container - accounts are created and stored. Namespaces used to group objects
- Object - actual data
- Ring - maps partition space to physical locations on the disk

▼ **NoSQL Database - Dynamo DB**

- Key-value database, supports item level consistency
- Stores data in partitions that can be accessed using Partition key. Partition key and sort key (composite key that is optional).

▼ Partitioning

→ Skewed: some partitions have more data than others

→ Hot spot: partition with a disproportionately high load

▼ Approaches of partitioning:

1. **Vertical partitioning**
2. **Workload driven partitioning** (data access patterns analysed to form partitions. increases scalability in terms of throughput and response time)
3. **Partitioning by random assignment** (avoid hotspots)
4. **Horizontal partitioning**
    - Partitioning by key range (assign continuous range of keys to each partition)
    - Partitioning using schema (related rows are kept in the same partition)
    - Partitioning using graph partitioning (analyse data to partition)
    - Partitioning using Hashing (consistent hashing)

- ▼ Partitioning with secondary indexes:
    - Document based partitioning
    - Term Based partitioning

▼ **Rebalancing partitions**

The process of moving load from one partition to another is called repartitioning

Strategies for rebalancing partitions:

- Hash based rebalancing (Hash mod n)
- Fixed number of partitions
- Dynamic partitioning
- Partitioning proportionally to nodes

▼ **Request routing**

Approaches -

1. Round robin load balancer
2. Routing tier that acts as a partition aware load balancer
3. Clients are aware of partition and connect directly to node
4. Client - routing tier - zookeeper - nodes

▼ **Replication**

- ▼ Leader based/Single leader replication

    Leader is a dedicated compute node. Sends changes as replication logs to followers.

- Synchronous replication - leader waits for confirmation before making write visible
- Asynchronous - leader only send the message but doesn't wait for confirmation

Implementation of replication logs:

1. Statement based replication
2. Write ahead log shipping
3. Change data capture based replication
4. Trigger based replication

Follower failure: Catch up recovery (request leader for log to understand data changes)

Leader failure: Failover (follower prompted to become new leader)
Leads to replication lags.

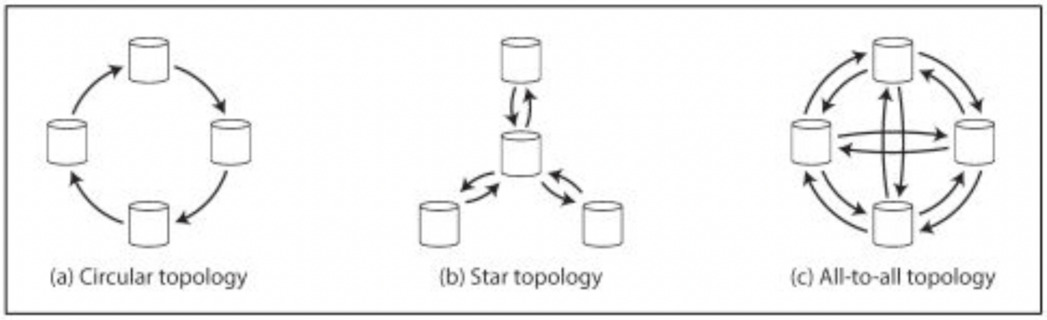
▼ Multileader replication

- This approach is also known as master/master or active/active replication
- Each leader simultaneously acts as a follower to the other leaders.

Problems faced:

1. Conflict avoidance
2. Converging toward a consistent state
3. Custom conflict resolution logic

Multileader topologies:



▼ Leaderless replication

- Upon a write or read, client broadcasts request call to call replicas and waits for a certain number of ACKs (quorum)
- Anti entropy process: some datastores have a background process that constantly looks for differences in data bw replicas and copies any missing data from one replica to another.
- Use timestamps to address concurrent writes (last write wins)

▼ **Consistency Models SCPS**

A consistency model is a contract between a distributed data store and process in which the processes agree to obey certain rules in contrast the store premises to work correctly

▼ Sequential consistency (Lamport)

All process see the same order of all memory access operators. Non deterministic because multiple execution of the distributed program might lead to different order of operations

▼ Casual consistency

Two events are casually related if one can influence the other. Memory reference operations that are not casually related may be seen in a different order.

- ▼ PRAM consistency

  All write operations performed by a single process are seen by order in which they were performed. Different write operations, different processes, different order.

- ▼ Strict consistency

  Value returned by most recent read operation is same as write operation. Order of programs between processes must be the order in which those operations were received.

▼ **Linearizability**

Recency guarantee: a read is guaranteed to see the latest value written.

Single leader and consensus algorithms are linearizable.

▼ **CAP theorem**

→Consistency: all reads receive the most recent write or an error

→Availability: All reads contain data, but it might not be the most recent

→Partition tolerance: The system continues to operate despite network failures

The CAP theorem state that it is not possible to guarantee all three of the desirable properties at the same time in a distributed system with data replication.

Trade-off between requirements-

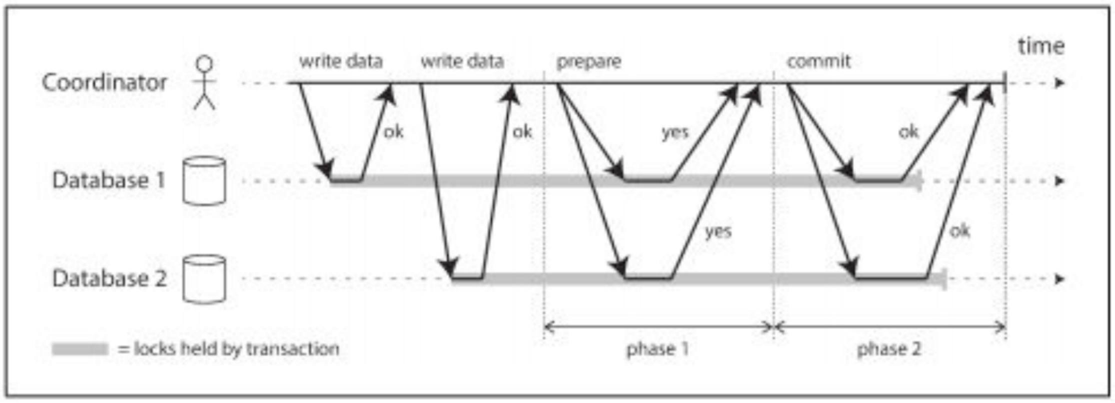
1. Comprised consistency
2. Comprised availability
3. Comprised partition tolerance

▼ **Distributed transactions**

ACID properties of transactions (Atomicity, Consistency, Isolation and Durability)

Local transaction manager, Transaction coordinator.

Concurrency control and commit protocols such as 2PC


Coordinator
Database 1
Database 2
write data
write data
prepare
commit
time
ok
ok
yes
yes
ok
ok
= locks held by transaction
phase 1
phase 2