**A screenshot of a computer

Description automatically generated**

**COW (Copy on write):** Is an alternative to overwriting data in-place. Updating in place works with HDDs but SSDs have limitations as discussed earlier that makes this process inefficient. Therefore, copy on write can be implemented to reduce the wear impact and garbage collection complexity.

There are two types of COWs for B-trees, COW-random and COW-sequential.

**LSM-tree**: C0 (component) is in memory, the rest are on-disk components.

C0, which is in memory, is a sorted table. This is also saved in a write ahead log on disk, as a form of backup or reference in case anything goes wrong. C0 is called a memtable

Once memtable is full, they get flushed to disk as SSTABLES which are immutable.

SSTABLEs have segments which have keys in a sorted order

When we search for keys, we first search in the memtable (C0) then if its not there, we check the SSTABLES in a reversed order (latest result and then upwards)

LSM-trees avoid random writes with its append-only data structure. In append-only log, writes are always sequential, which is better for HDDs and somewhat better for SSDs.

**Append-only data structures:**

In append-only log, writes are always sequential, which is better for HDDs and somewhat better for SSDs.

It’s good for recovering from crashes because you don’t have to worry about inconsistencies. For example, you don’t have to worry about the case where a crash happened while a value was being overwritten (resulting in new and old data spliced together). This paves the way for snapshots, replication, and read isolation.

Merging old data values can help avoid fragmentation as they will be grouped together instead and easier to locate.

**Raft:**

A consensus algorithm for fault tolerance in distributed databases. Often used where systems require fault tolerant data replication. RAFT has a concept where the log is replicated to all participants.

**Dynamo:**

No-SQL movement (one of the first)

Eventual consistency, better to trade consistency for high availability

Reliability and efficiency

*Consistent hashing:* Virtual nodes (powerful nodes have more virtual nodes), uses hashing on keys to place data on nodes. Consistent hashing is a way of hashing that supports incremental scalability, where a newly added node will take half of the storage of an existing node.

Replication: after data is added to a node a replica is sent to neighboring nodes.

Vector clocks: Vector clocks let you have the possibility to reconcile updates that have happened “in parallel”. But it is up to the application how the update is reconciled.

Data versioning: Vector clocks are used. To support high availability, Dynamo prioritizes accepting writes over rejecting them due to conflicts or failures. This can result in multiple versions of a data item existing simultaneously. To track the causal relationships between different versions of a data item, Dynamo uses vector clocks. Dynamo reconciles divergent versions in one of two ways:

* Syntacticreconciliation: the system automatically determines the authoritative version, such as when updates are sequential, and a newer version clearly supersedes an older version.
* Semantic reconciliation: when conflicting concurrent updates result in divergent versions that the system cannot reconcile, the client is responsible for reconciling the versions. An example of semantic reconciliation is merging the contents of two versions of a customer's shopping cart.

Generic load balancer: Sends request to random node, the node then forwards the request to top nodes in the preference list.

Merkle trees: Merkle trees are used to fast decide if replicas are not having the same state. They use hashing in multiple levels, where top level hashes may easily be compared.

1. The leaves of the tree contain data or their hashes.
2. Parent nodes contain the hash of their child nodes.
3. The root node contains the hash that represents the entire data structure.

Instead of comparing all data directly, you compare hashes, which is much more efficient.

Dynamo uses Merkle trees for replica synchronization.

Sloppy quorums and hinted handoff: This means that during periods of node failures, some other node will temporarily take over the role for the failed node but will give back the responsibility when the node has recovered.

* N: Total number of replicas for each piece of data.
* W: Minimum number of nodes that must confirm a write.
* R: Minimum number of nodes that must confirm a read.

In a sloppy quorum, if the preferred W or R nodes are unavailable, Dynamo temporarily allows other nodes (not responsible for the data) to handle the operation. This ensures the system remains available even during node failures.

Hinted handoff will eventually move the data to the correct node once the node is available.

Gossip based membership protocol: This is the protocol used for communication between the nodes. Each node contacts a peer chosen at random every second. Seeds are nodes known to all nodes. Every node is doing its own failure detection (local).

Adding and removing nodes: take data, and remove data (add/add)

Cannot scale into tens of thousands because of gossiping, each node has full list of routing info.