Consistency

Suppose, you deposit Rs.5000 in your Bank Account where before depositing your account balance was Rs.0. In the evening, you urgently need money for paying your house rent. You rush back to the ATM machine nearest to your home and try to withdraw money but the account balance is still 0. You connect with the technical staff and know that the database has not been updated yet and money can only be withdrawn once the server is updated. This is an example of poor consistency. Poor consistency causes irregular data flow through the system and is a cause of major system issues along with poor user experience.

Let’s deep dive into consistency.

What is consistency?

Consistency is uniformity in data. When a user requests data and the system reverts back with the same data irrespective of the geographical location, time etc.

For a consistent system, the server at which data is updated or changed should effectively replicate the new data to all the nodes before the user reads the data from any node.

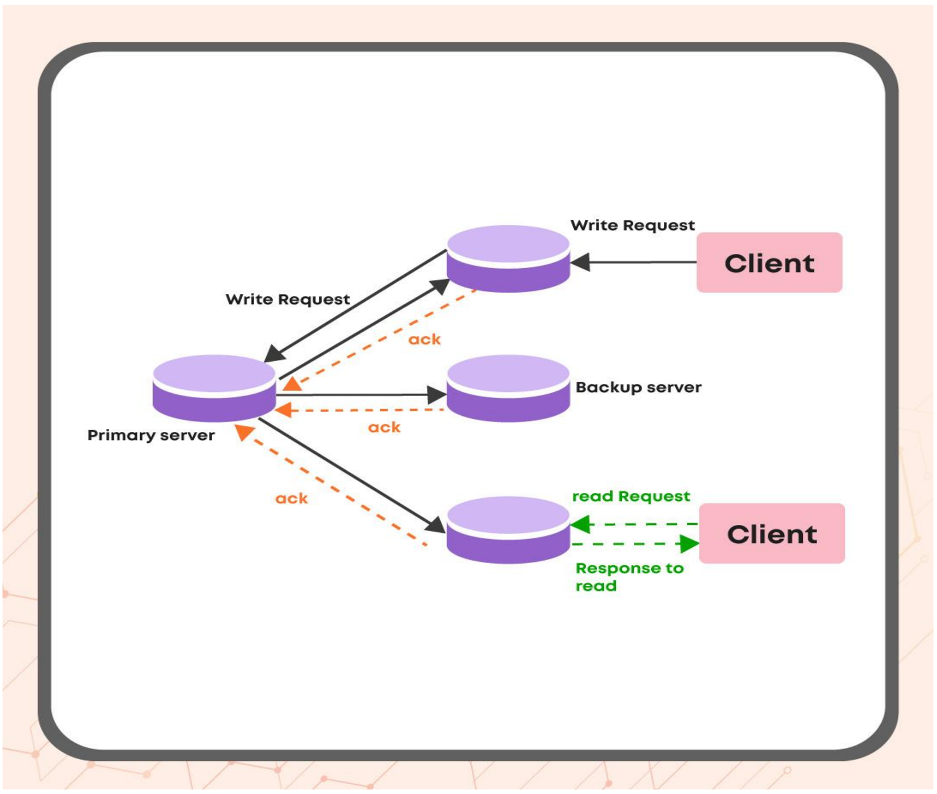

Fig: Consistency Model

Example: A system has three nodes A,B and C. The user has changed data at node A and the replication of new data from A to B takes t1 time and A to C takes t2 time. The system should not accept any read operations till t3 time which is the maximum of t1 and t2. So that no user gets inconsistent data or the old data.

Dirty read: Dirty Read is a phenomenon that occurs when a transaction reads data that has not yet been updated and returns the old data to the user.
For example: Suppose transaction A updates a row in node x. Before the same row is updated in node y, the user reads the information. This is when the user receives old data. This is a situation of dirty reading.

Consistency and Architecture

Monolithic Architecture:

- Centralised unit with all the code and layers implemented in a single codebase.
- Single node/server with no concept of replication.
- The data read is the data written last.
- Because the data is not replicated, the monolithic systems are naturally consistent.

Monolithic Architecture => No Replication => Natively Consistent

Distributed System:

- Use Data Replication for improving data accessibility and availability.
- Therefore, data consistency is not naturally present.
- It is the process of different nodes having the same value.
- In a distributed system, different nodes reach a single consistent state.
- The consistency depends on how fast the replicas of the data are updated.
- The faster the update, the better the consistency.
- Nodes must be efficiently synchronized and the read process must be stopped during the update process.

Distributed System => Replication => Needs efficient update

Factors Improving Consistency

1. <u>Halt the read:</u>
   For ensuring a consistent system, read must be stopped until data at all the nodes are updated.
2. <u>Improving Network Bandwidth:</u>
   Consistency depends on the speed of data update of replicas. By improving the network bandwidth, the speed can be increased. This will help synchronize the replicas.
3. <u>Replication based on Distance aware strategies:</u>
   If most of the replicas are stored in the same data center, the speed of update should increase and the process will be more optimised.

Types of Consistency

There are different types of consistency models.

1. <u>Strong Consistency:</u>
   It is the strongest model of memory coherence. Strict consistency is when the system doesn't allow read operation until all the nodes with replicate data are updated. For n number of nodes in the distributed system, the time of updating the replicas be t1,t2...tn. The maximum of t1,t2...tn is the time after which all the replicas are updated and overwritten with the latest data. The read process can be continued only after this.
   When a user requests data and the system reverts back with the latest data irrespective of the geographical location, time etc.

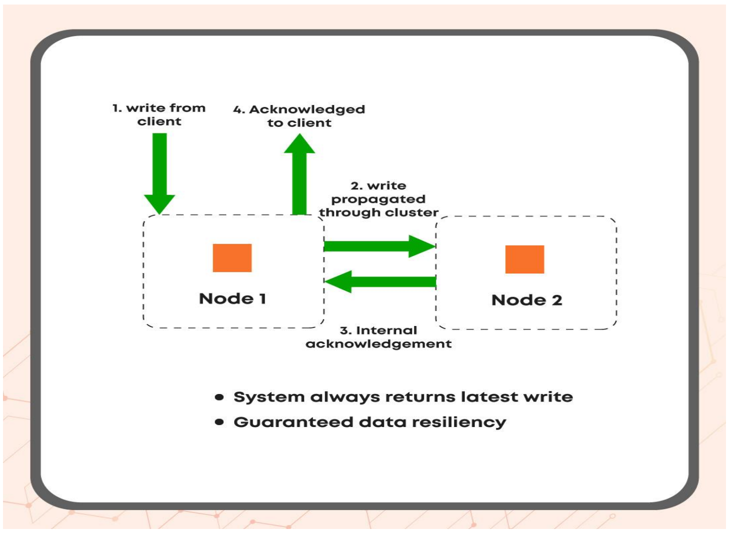
1. write from client
4. Acknowledged to client
2. write propagated through cluster
Node 1
Node 2
3. Internal acknowledgement
System always returns latest write
Guaranteed data resiliency

Fig: Strong Consistency

2. <u>Eventual Consistency:</u>

   User Read requests are not halted till all the replicas are updated rather the update process is eventual. Some users might receive old data but eventually all the data is updated to the latest data. It weekends the aspect of consistency because it is an eventual process.

   If the node where data was written breaks down due to any reason, the data would be lost. It also doesn't guarantee that the data responses back would be the latest replica.

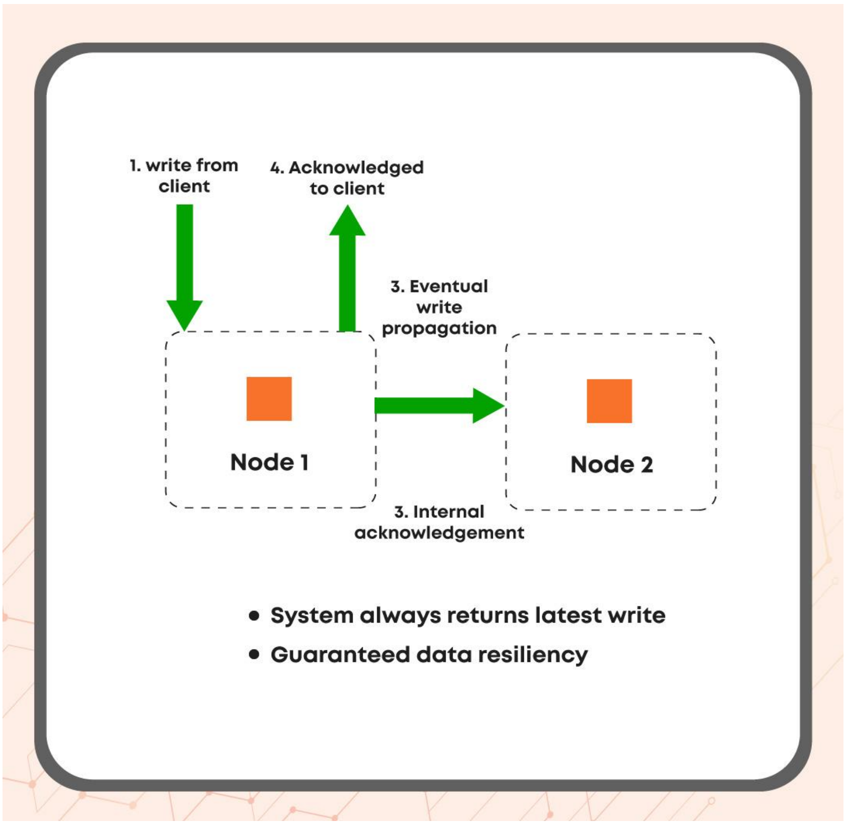
1. write from client
4. Acknowledged to client
3. Eventual write propagation
Node 1
Node 2
3. Internal acknowledgement
● System always returns latest write
● Guaranteed data resiliency

Fig: Eventual consistency

<u>3. Weak Consistency:</u>

It is the worst model of consistency which essentially doesn't ensure consistency. During a variable or data change by the user, it is not mandatory to inform or update the changed information in all the replicas. Each node would carry different states of data and therefore should be avoided.

There is no guarantee that all the nodes in a distributed system would carry the same information at a point in time.