Consistency Model and Linearizability

Hui Chen [a]

[a]CUNY Brooklyn College

October 8, 2025

Distributed Storage?

Common design:

- Storage service
- Computation service

Examples:

- Web application vs. database
- MapReduce vs. GFS/Hadoop vs. Hadoop vs Hadoop FS

Replication and Consistency

Using replication: replicate data/services across n servers

- Fault tolerance. Under the fail-stop failure model, if up to $n-1$ of n servers crash, at least 1 is alive.
- Availability. $P(\text{Availability}) = 1 - P(\text{All Servers Crash}) = 1 - p^n$
- Load balancing. Assign requests to different servers (simple implemenation: DNS round-robin)

Question: How to maintain consistency across different data replicas?

But, what is "consistency"?

Consistency: Correct Behavior of the Storage Service

The basic operations of a storage service:

- put(k, v) $\rightarrow$ <done>
- get(k) $\rightarrow$ v

Consistency: Correct Behavior of the Storage Service: Sequential Programming

In ordinary sequential programming, we expect a read to yield the last written value, i.e., get(k) $\rightarrow$ v, where v is the value of the last put(k, v) operation.

Consistency: Correct Behavior of the Storage Service: Distributed Storage

In distributed storage, the answer is not as simple:

- Producer. It computes, then executes

```
put("grade", "A")
put("done", true)
```

- Consumer. It executes

```
while get("done") == false:
  pause
v = get("grade")
```

is v guaranteed to be "A"?

Consistency: Correct Behavior of the Storage Service: Distributed Storage

Consistency model: a specification for the relationship of different clients' views of a service, i.e., a defininition of correct behavior for distributed storage.

When might there be any question about what's correct?

- read concurrent with write
- replicas
- caches
- failure, recovery
- lost messages
- retransmission
- ...

Consistency Models

A few consistency models have been proposed over the years due to various design goals (performance / convenience / fault-tolerance tradeoffs):

- improving storage performance
- making programming easier
- making it easier to write specifications for implementors

Examples of consistency models:

- Linearizability (Strong consistency)
- Eventual consistency
- Causal consistency
- Fork consistency
- Sequential consistency

There can also be overlapping definitions for:

- file-systems, databases, CPU memory, cache memory, ...

Linearizability

Linearizability matches programmer intuitions just like single process sequential programming.

- Strongest consistency model
- But rules out many optimizations

Linearizability: Multiple Clients

We expect that a read operation would return the most recent write according to the *single physical-time order*.

- get(k) $\rightarrow$ v, where v is the value of the last put(k, v) operation, regardless of the clients.

In other words, we expect read/write to behave as if there were a single (combined) client making all the requests.

Linearizability: Multiple Storage Replicas

A single client with multiple storage replicas:

- A read operation would return the most recent write, regardless of how many copies there are.
- Read/write would behave as if there were a single copy.

Linearizability

To summarize:

- A read operation should return the most recent write according to physical time,
- regardless of how many clients there are, and
- regardless of how many copies there are.

This is: the single-client, single-copy semantics.

- Read/write should behave as if there were. . . a single client making all the (combined) requests in their original physical-time order

A storage system guarantees linearizability when it provides the above behavior

Linearizability: Formal Definition

A system is linearizable if the result of any execution is the same as if the operations of all the clients were executed in some sequential order, and the operations of each individual client appear in this sequence in the order issued by the client.

In other words, there exists a total order of all operations such that:

- The order respects the real-time ordering of operations (if op1 completes before op2 starts, then op1 precedes op2 in the total order)
- Each read returns the value of the most recent write preceding it in the total order.

Design and Implementation of Linearizable Storage

- How to design and implement a linearizable storage system?
- What are the tradeoffs?