Topic 6: Replication and Consistency

- Why do you need replication?
- Explain the challenges resulting from replication
- What consistency models exist?
- Explain the consistency model and compare them
- Present an execution which is sequentially consistent but not linearizable

Material: slides on the above topics, and DS book: Chapter on Replication.

- [DS 5ed] Sections 18.1-18.3 (replication)
- [DS 5ed] Section 18.4.1 (the gossip architecture)
- [DS 5ed] Section 17.1-17.3 (Distributed Transactions)

The Need for Replication

Replication allows services to provide high availability and fault tolerance *despite* different types of failures, lag, etc.

Replication Challenges

CAP Theorem:

- **Consistency**
- **Availabilty**
- **Partition Tolerance**
- It is impossible for a distributed computer system to simultaneously provide Consistency, Availability and Partition Tolerance.
    - ** A distributed system can satisfy any two at the same time, but not all three.
- ** Means a choice must be made if and when there is a partition
    - Either become less available, or allow the data seen by multiple clients to be inconsistent
    - This depends on the exact application
        - Web stores are likely to prefer Availability, while Safety critical systems prefer Consistency.

Existing Consistency Models

A consistency model specifies the guarantees of DSM system.

- **Strong Consistency**:
    - After an update of Process A completes, any subsequent access (by any process) will return the updated value
- **Weak consistency**:
    - System does not guarantee that subsequent access will return the latest value.

Comparison of Consistency Models

The two main correctness criteria for replicated objects are Linearizability and Sequential Consistency. Linearizability is the stronger of the two correctness criteria, but is also unrealistic wrt performance considerations.

Linearizability must satisfy two conditions:

- The sequence of operations meets the specification of a correct copy of the object, i.e. the operations need to make logical sense.
- The order of operations needs to be the same as the real time order of operations.

Sequential consistency shares the first condition, but ignores the real time property of the second condition, requiring instead that the sequence of operations must happen in the same order as it happens on the clients.

Sequentially Consistent, but not Linearizable

The example given in the slides of a service that is sequentially consistent but not Linearizable:

Client1	**Client 2**
SetBalanced(x,1)	
	GetBalance(y) = 0
	GetBalance(x) = 0
SetBalanced(y,2)	

For this execution, the real time property of Linearlizability does not hold, i.e. this is not correct wrt a banking system. However, will the less strict Sequentially Consistent model, one could shuffle the second event to occur in front of the first, and in this case have a consistent model.

Active vs. Passive Replication

Passive replication uses a master-slave architecture and is linearizable under certain conditions. Active replication uses multicast, and performs better, but is not linearizable, only sequentially consistent.

The Gossip Architecture

Focuses on high availability, where Replication Managers share gossip messages with each other. Each RM is guaranteed to become consistent over time.