The many faces of consistency

Marcos K. Aguilera, Douglas B. Terry
Presented by Ji Wang

Outline

- Introduction: What is consistency and why important
- Abstract model and terminology
- Two types of consistency
  - State consistency
  - Operation consistency
  - Comparison
- Consistency in different disciplines
  - Distributed systems
  - Database systems
  - Computer architecture

Introduction (1)

- Share data: From calculating machines to tools of information exchange
  - Distributed system: files, network names, configuration info, etc.
  - DB: related tables
  - Architecture: processor cores share cache lines and physical memory
- Replicate data: For speed or to tolerate disasters
  - Distributed system: each site holds a local replica
  - DB: rows or tables are replicated
  - Architecture: memory hierarchy

Introduction (2)

- Fundamental question:

  What should happen if a client modifies some data items and simultaneously, or within a short time, another client reads or modifies the same items, possibly at a different replica?

- Consistency varies significantly across different disciplines.
- But in general, it places constraints on the allowable outcomes by limiting how data sharing and replication work.

Abstract model

We consider a setting with multiple *clients* that submit *operations* to be executed by the system.

- Clients: human users, computer programs, etc.
- Operations: simple read and write, read-modify-write, transactions, and queries.
- Operation execution is not instantaneous: *starts* when a client submit the operation, and *finishes* when the client obtains its response from the system.
- State: current value of the data items.

State consistency (1)

Properties of system state that users expect to satisfy despite concurrent access and the existence of multiple replicas.

1. Invariants:

DB system	Distributed system
uniqueness constraints: e.g. primary key	mutual consistency
referential integrity: e.g. foreign key	

State consistency (2)

2. Error bounds

   For numerical data, a deviation(error) from the expected is allowed.

3. Limits on proportion of violation

   Not all, but a high percentage of properties are expected to hold.

4. Importance

   Only the critical properties to hold at all times.

5. Eventual invariants

   An invariant may need to hold only after some time has passed.

Operation consistency (1)

Properties that indicate whether operations return acceptable results.

1. Sequential equivalence:
   - Linearizability[*Strong*]:

     op1 < op2 iff op1 finishes before op2 starts. There must exist a legal total order T of all operations with their results, such that:

     (1) T is consistent with <, meaning that if op1 < op2 then op1 appears before op2 in T,
     (2) T defines a correct *sequential execution*.

Operation consistency (2)

1. Sequential equivalence:
   - Sequential consistency[*Strong, weaker than linearizability*]:

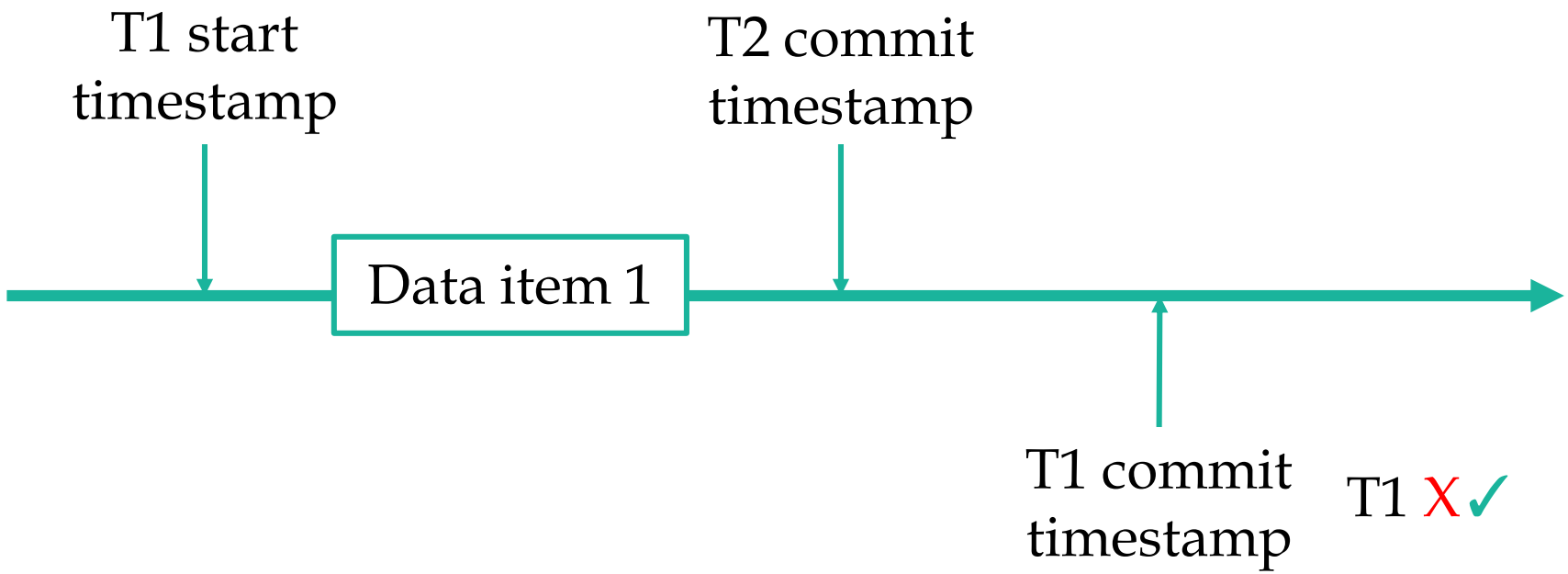
     operations are issued by the same client.
   - Serializability: each transaction appears to execute in series.
     - Strong session serializability:

       each transaction is associated with a session, the serialization must respect the order of transactions within every session.
     - Order-preserving serializability:

       serialization order must respect the real-time ordering of transactions.

Operation consistency (3)

2. Reference equivalence: requires the concurrent execution to be equivalent to a given reference.
   - Snapshot isolation:



Operation consistency (4)

3. Read-write centric:

   Write may cover the *entire* data item, or update just *part* of a data item, the crucial consideration is the set of writes that could have potentially *affected* the read; We say that the read sees those writes.

   - Read-my-writes

     a read by a client sees at least all writes previously executed by the same client.

   - Bounded staleness

     a read must see at least all writes that complete δ time before the read starts.

State consistency VS operation consistency

	State consistency	Operation consistency	Note
Level of abstraction		High	Whether clients can observe directly
Complexity		High	
Application dependence	High		The correct state of a system varies.

Consistency in different disciplines

Area	Type
Distributed system	• Either state or operation consistency. often uses weaker level of consistency due to difficulty in client coordination, high availability, scalability and geo-distribution.
DB system	• State consistency data is more important than operation result.
Computer architecture	• Operation consistency consistency constrains the behavior of reads and writes(loads and stores) across all the memory locations.

Summary

The concern of consistency stems from the rise of concurrency and replication.

Unfortunately, consistency is subtle and has different names and meanings across communities.

We identify two broad types of consistency—state consistency and operation consistency—and exemplify them in different disciplines.

Thank you