THE UNIVERSITY *of* EDINBURGH

Distributed Systems Fall 2024

Yuvraj Patel

Today's Agenda

Replication

- Client-Centric Consistency Models

Slack

Next Class Monday(11/11)

Consistency Models

A consistency model is a contract between the programmer and a system

- The system guarantees that if the programmer follows the rules for operations on data, data will be consistent
- Result of the reading, writing, updating data will be predictable

Two consistency models

- Data-centric consistency models – Defines consistency as experienced by all the clients; provides a system wide consistent view on the data store
- Client-centric consistency models – Defines consistency of the data store only from one client's perspective; Different clients might see different sequences of operations at their replicas

Data-Centric Consistency Models

Strong Consistency Models – Operations on shared data are synchronized; do not require synchronization operations

- Strict Consistency – Absolute time ordering of all shared accesses matters
- Sequential Consistency – All processes see all shared accesses in the same order
- Linearizability – Sequential Consistency + Operations are ordered according to a global time
- Causal Consistency – All processes see causally-related shared accesses in the same order
- FIFO Consistency – All processes see writes from each other in the order they were used

Weak Consistency Models – Synchronization occurs only when shared data is locked and unlocked; rely on synchronization operations

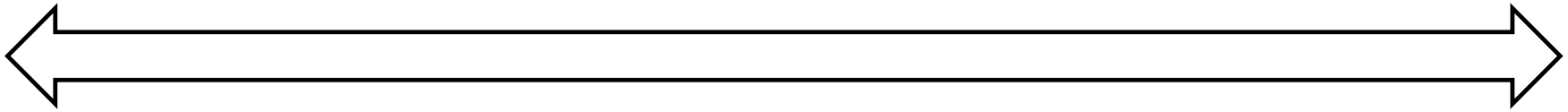
- Weak Consistency – Shared data can be counted on to be consistent only after a synchronization is done
- Release Consistency – Shared data are made consistent when a critical region is exited
- Entry Consistency – Shared data pertaining to a critical region are made consistent when a critical region is entered

Weaker the consistency models, the more scalable it is

Data Replication in Cloud

Client-Centric Consistency Models

Spectrum of Consistency



Lots of consistencies proposed in research community

Today, we will consider the six Guarantees discussed by Doug Terry

Baseball Game

One game comprises 9 innings

Game starts with 0-0 score

Visitors bat first and remain at bat until they make three outs

Then home team bats until they make three outs

Continue for 9 innings

Key-Value Store for Score

Score recorded in a Key-Value (KV) store in two objects

One object for visitor's score, another for home team

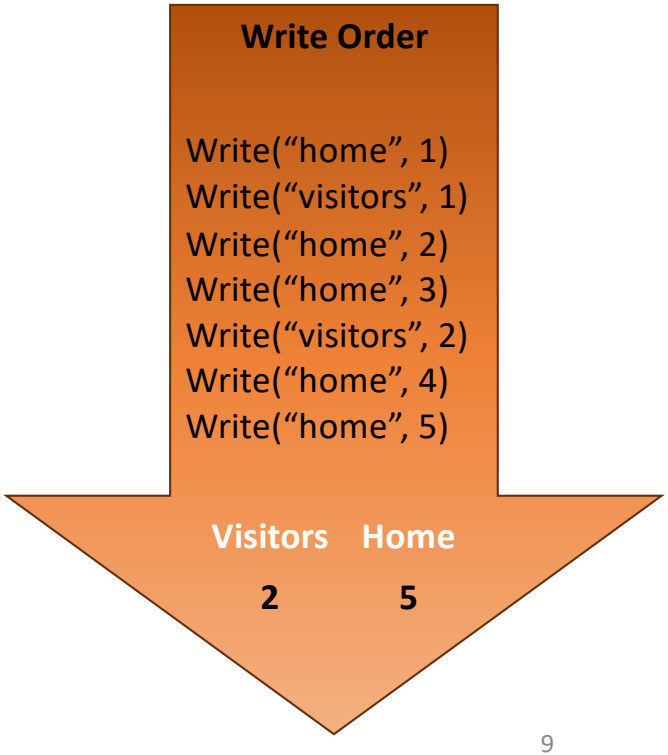
When a team scores a run,

- Read operation is performed on its current score
- The returned value is incremented by 1
- The new value is written back to the KV store

Sample Game Score

Existing Score

	1	2	3	4	5	6	7	8	9	RUNS
Visitors	0	0	1	0	1	0	0			2
Home	1	0	1	1	0	2				5



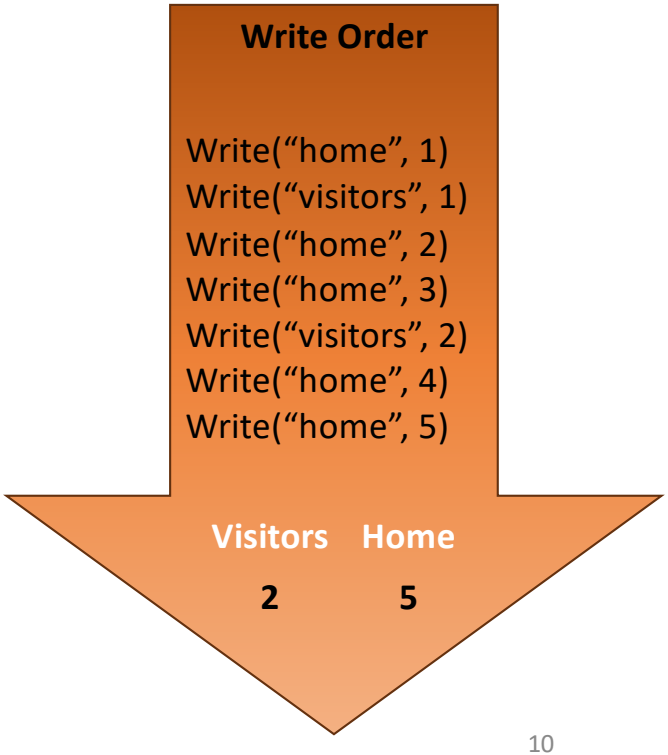
Strong Consistency

Guarantee – See all previous writes

All readers read the same data

Possible values

2, 5



Eventual Consistency

Guarantee – See subset of previous writes

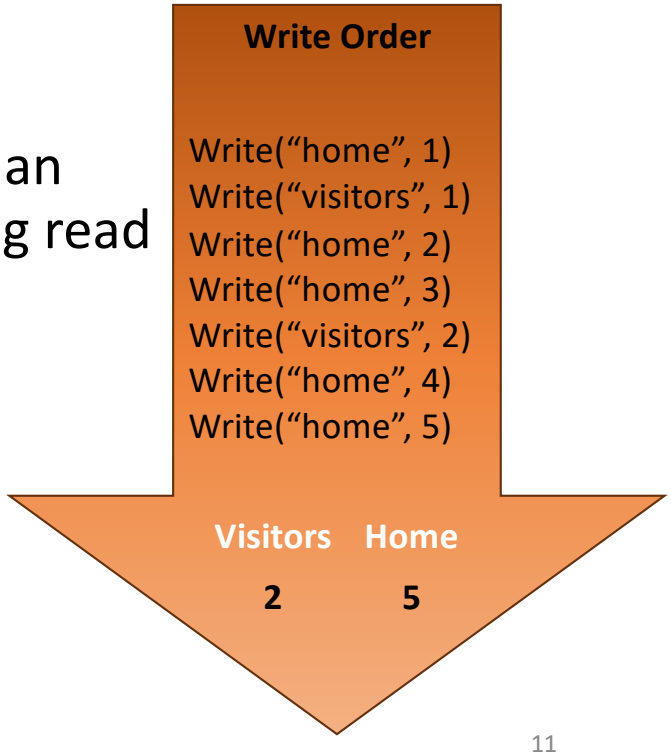
Readers can read data that is written in the past

Read can return from a replica that has received an arbitrary subset of writes to the data object being read

Eventually see all writes

Possible values

0-0, 0-1, 0-2, 0-3, 0-4, 0-5, 1-0, 1-1, 1-2, 1-3, 1-4, 1-5, 2-0, 2-1, 2-2, 2-3, 2-4, 2-5



Consistent Prefix

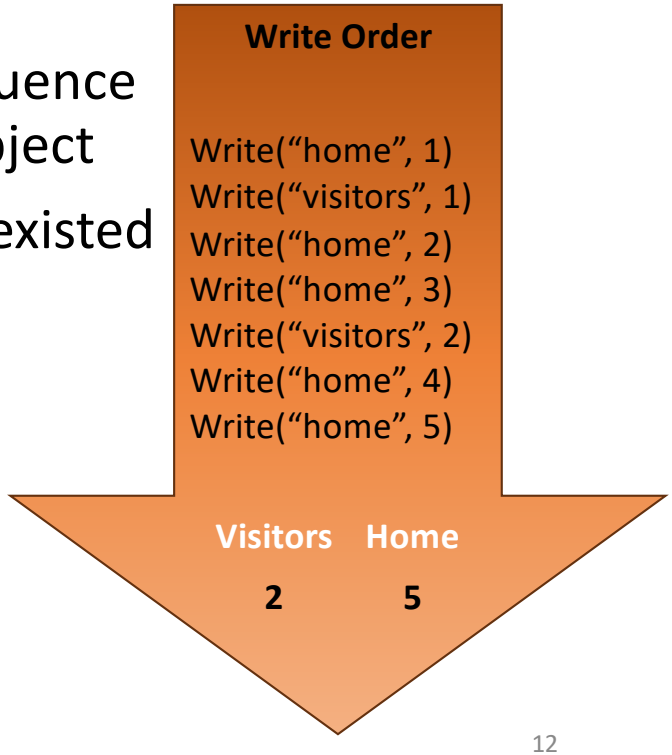
Guarantee – See initial sequence of writes

Reader is guaranteed to observe an ordered sequence of writes starting with the first write to a data object

The reader sees a version of the data store that existed at the master at some time in the past

Possible values

0-0, 0-1, 1-1, 1-2, 1-3, 2-3, 2-4, 2-5



Bounded Staleness

Guarantee – See all ”old” writes

Reader ensured that read results are not too out-of-date

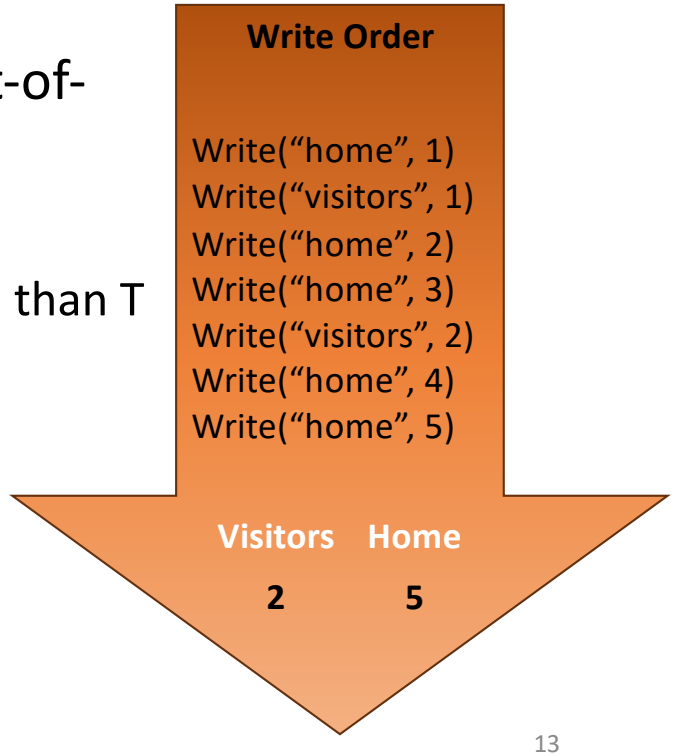
Staleness factor

- Read operation will return any values written more than T minutes ago or more recently written value

Possible values

At most one inning out-of-date score like

2-3, 2-4, 2-5



Monotonic Reads

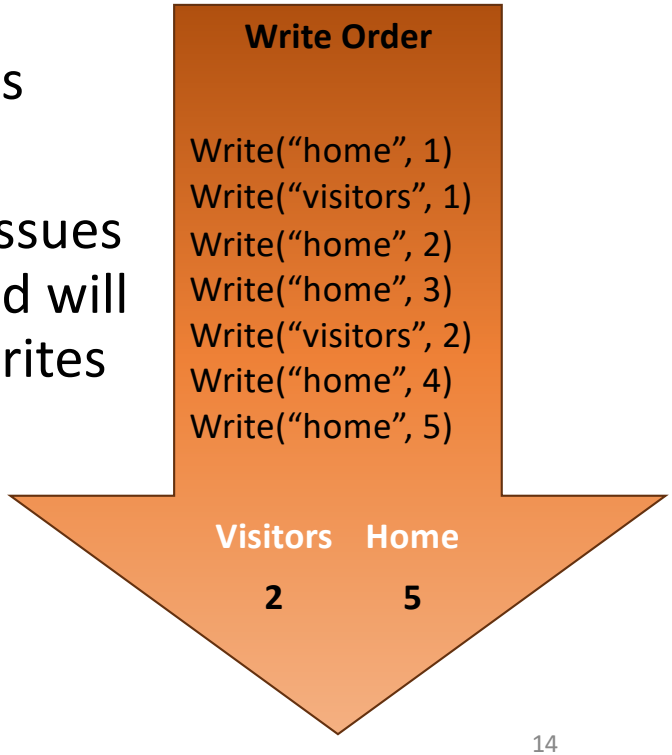
Guarantee – See increasing subset of writes

Reader guaranteed to observe a data store that is increasingly up-to-date over time

If reader issues a read operation and then later issues another read to the same object; the second read will return the same value(s) or the results of later writes

Possible value

After reading 1-3, 1-3, 1-4, 1-5, 2-3, 2-4, 2-5



Read My Writes

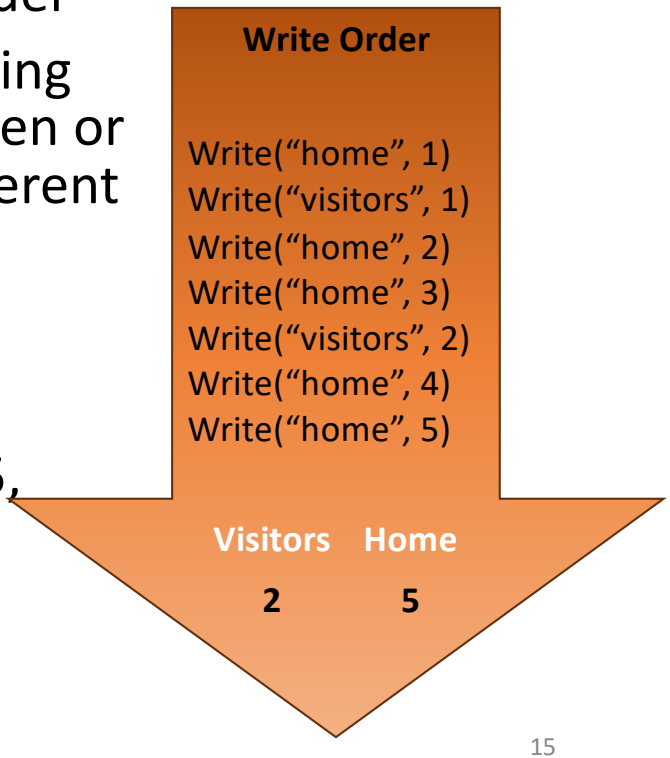
Guarantee – See all writes performed by the reader

After writing a new value and subsequently reading the value will return the value that was last written or some other value that was later written by a different client

Possible value

For the writer: 2-5

For anyone else: 0-0, 0-1, 0-2, 0-3, 0-4, 0-5, 1-0, 1-1, 1-2, 1-3, 1-4, 1-5, 2-0, 2-1, 2-2, 2-3, 2-4, 2-5



Summary

Different read guarantees

- Strong Consistency – See all previous writes
- Eventual Consistency – See subset of previous writes
- Consistent Prefix – See initial sequence of writes
- Bounded Staleness – See all “old” writes
- Monotonic Reads – See increasing subset of writes
- Read My Writes – See all writes performed by the reader

Consistency Trade-off

Trade-off between three properties – Consistency, Availability, Performance

Some entries may vary based on implementation, deployment, operating details, etc.

	Consistency	**Performance**	**Availability**
Strong Consistency	Excellent	Poor	Poor
Eventual Consistency	Poor	Excellent	Excellent
Consistent Prefix	Okay	Good	Excellent
Bounded Staleness	Good	Okay	Poor
Monotonic Reads	Okay	Good	Good
Read My Writes	Okay	Okay	Okay

Qualitatively Accurate General Comparison