Session Guarantees for Weakly Consistent Replicated Data

Some Context

Bayou: A replicated database platform design for mobile devices.

This was done in the early 90’s.

Lazy propagation of updates between servers.

“Provide a client with a view of the replicated data that is consistent with its own actions.”

Session Guarantees

Session: A sequence of read or write applications.

Four guarantees:

- Read your writes: reads reflect previous writes by a user.
- Monotonic Writes: successive reads reflect a set of writes
- Writes Follow Reads: Writes propagated after reads they are dependent on.
- Monotonic Writes: Writes are propagated after logical predecessors.

Assumptions

Replicated Database across multiple servers.

Client accesses different servers based on a few factors.

Write ID (WID)

DB(S,t): sequence of writes recieved at time t

DB(S1, t) != DB(S2,t)

Order of non commutative writes is preserved. WriteOrder(1, W2)

Read Your Writes

If Read R follows a write W in a given session and R is performed on S at t, then W is in DB(S,t).

Monotonic Reads

RelevantWrites(S,t,R) smallest set of writes that is complete for R and DB(S,t)

If a read R1 occurs before R2 and R1 accesses S1 at time t1 and R2 accesses S2 at time t2 then Relevant Writes(S1,t1,R1) is a subset of DB(S2, t2)

Writes Follow Reads

If a read R1 precedes write W2 and R1 is performed at server S1 at time t1 then for any server S2, if W2 is in DB(S2) then any W1 in RelevantWrites(S1, t1,R1) is also in DB(S2) and WriteOrder(W1, W2).

WFRO: If a read precedes W2 on S1 then WriteOrder(W1, W2) for any W1 in Relevant Writes(S1, t1, R1)

WFRP: For any server S2 if W2 is in DB(S2) then any W1 in RelevantWrites(S1, t1, R1) is also in DB(S2)

Monotonic Writes

If write W1 precedes W2 then for any server S2, if W2 in DB(S2) then W1 is also in DB(S2) and WriteOrder(W1, W2)

Example 1

Appointment calendar modifiable by the user and a scheduler.

The system does not allow access to a replica that is behind the last read value.

Example 2

Bibliographic database, a user reads a value and makes a minor update

The update can only be applied if all relevant writes for the read are present.

Example 3

Text editor

Version N is written, then N+1.

What can be used to ensure that N can’t be written after N+1.

Example 4

After changing a password not all nodes may be aware.

A per user session for the lifetime of each account.

How can we always read the most recent user password?

Providing Guarantees

Session Manager

Maintains read-set and write-set

Server must return info about a WID

WFR and MW need two more guarantees:

Guarantee	session state updated on	session state checked on
Read Your Writes	Write	Read
Monotonic Reads	Read	Read
Writes Follow Reads	Read	Write
Monotonic Writes	Write	Write

Additional Constraints for WFR and MW

When a new write W2 is accepted S must ensure WriteOrder(W1, W2) is true for any W1 in DB(S,t)

Anti-entropy is performed so that if W2 propagated from S1 to S2 at t then any W1 in DB(S1,t) such that WriteOrder(W1, W2) is also propagated.

Implementation Problems

Session state can get large

Set of returned or checked WID’s can be large

Servers recording writes can get large.

Finding a server with all the information needed

Bookkeeping to determine what reads and writes are relevant.

Version Vectors

<server, clock>

Sequence of pairs for each server.

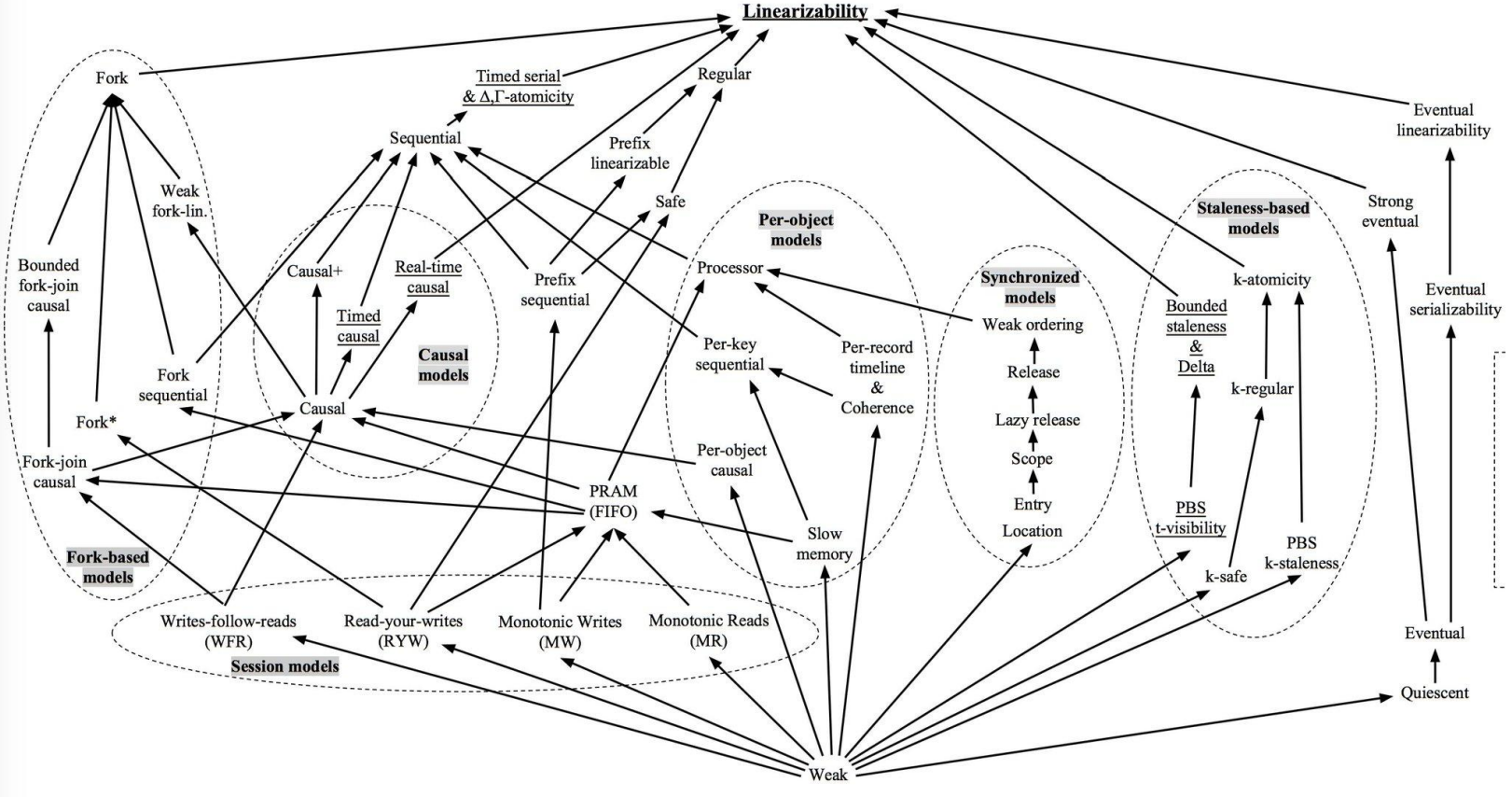
Implementation option for a write ID.

Two maintained by a session, writes and reads.

```
Read(R,S) = {
    if MR then
        check S.vector dominates read-vector
    if RYW then
        check S.vector dominates write-vector
    [result, relevant-write-vector] := read R from S
    read-vector := MAX(read-vector,
        relevant-write-vector)
    return result
    }

Write(W,S) = {
    if WFR then
        check S.vector dominates read-vector
    if MW then
        check S.vector dominates write-vector
    wid := write W to S
    write-vector[S] := wid.clock
    }
```

Where does this all fit in?



Actually using this stuff

Consistent from one user’s perspective.

No attempt at atomicity or serializability.

Choose what guarantees you want.

Reduced availability (latency).

Example 5

Weakly consistent bulletin board.

Ensuring that users see replies to a posted article only after they have seen the original.

Example 6

Bibliographic database

Permit users to see the newest version of a bibliography even if the older ones haven’t made it to the server yet.