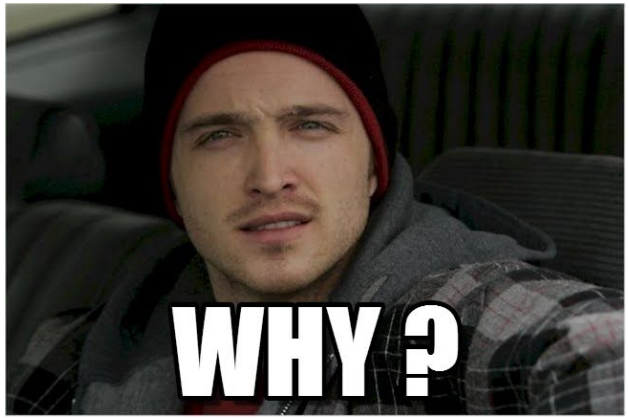
Consistency Protocols

Why ?

- Replication
  - Reliability
  - Improving Performance
- Introduces a consistency problem.
- Solving this leads to degradation of the system performance
- Solution: **Not** to solve this ! **#ConsistencyProtocols**



Overview

- 7.1 Continuous Consistency
- 7.2 Primary Based Protocols
- 7.3 Replicated Write Protocols
- 7.4 Cache Coherence Protocols
- 7.5 Implementing Client Centric Cohrency

Bounding Numerical Deviation

- Writes to a single data item x : **W(x)**.
- **Weight(W)**: numerical value by which X is updated.
- **origin(W)**: First write sent to a replica server(Out of N).
- **TW[i,j]** → writes executed by server Si that originated from Sj

$$TW[i,j] = \sum \{weight(W) \mid origin(W) = S_j \ \& \ W \in L_i\}$$

- The goal is for any time t, to let the current value Vi at server Si deviate within bounds from the actual value v(t) of x.

$$v(t) = v(0) + \sum_{k=1}^{N} TW[k,k]$$

$$v_i = v(0) + \sum_{k=1}^{N} TW[i,k]$$

Bounding Numerical Deviation

- I.e we impose an upper bound δi such that we need to enforce:

$$v(t) - v_i \leq \delta_i$$

Bounding Staleness Deviation

- Many ways,
- Simple Approach:
  - Each S_k has a Real time Vector Clock(RVC_k).
  - **RVCk[i] = T(i)**: S_k has seen all writes that have been submitted to S, up to time T(i).
  - **T(i):** time local to Si.
  - When Clocks are **loosely synchronised**:
    - **$T(k) - RVC_k[i] > n$**
    - Pull writes after **$RVC_k[i]$** from **S_i**.
- Unlike BND: **Pull Approach** - **Better Approach**.

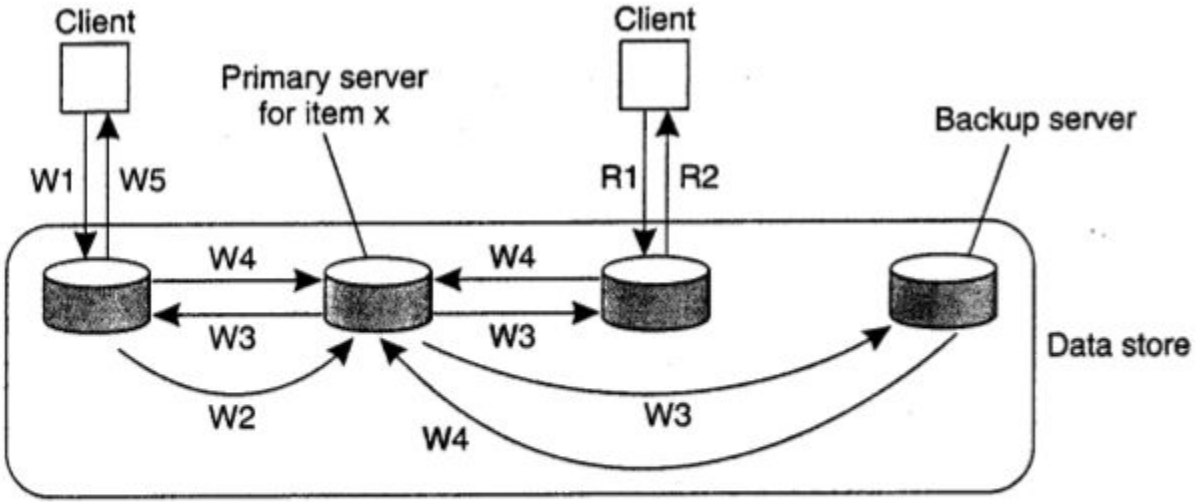
Bounding Ordering Deviation

- Each server **local queue** of writes submitted to it.
- Needs an input called **maximal length**.
- **Queue length > maximal Length**.
- Stop taking in writes and start ordering them by communicating with local servers.

“Primary” - Based Protocols

- In the case of sequential consistency, PBP standout.
- **Primary**: Process responsible for coordinating read and write operations on a data item **x.**
- 2 types based on the position of primary.
  - Remote Write Protocol.
  - Local Write Protocol.

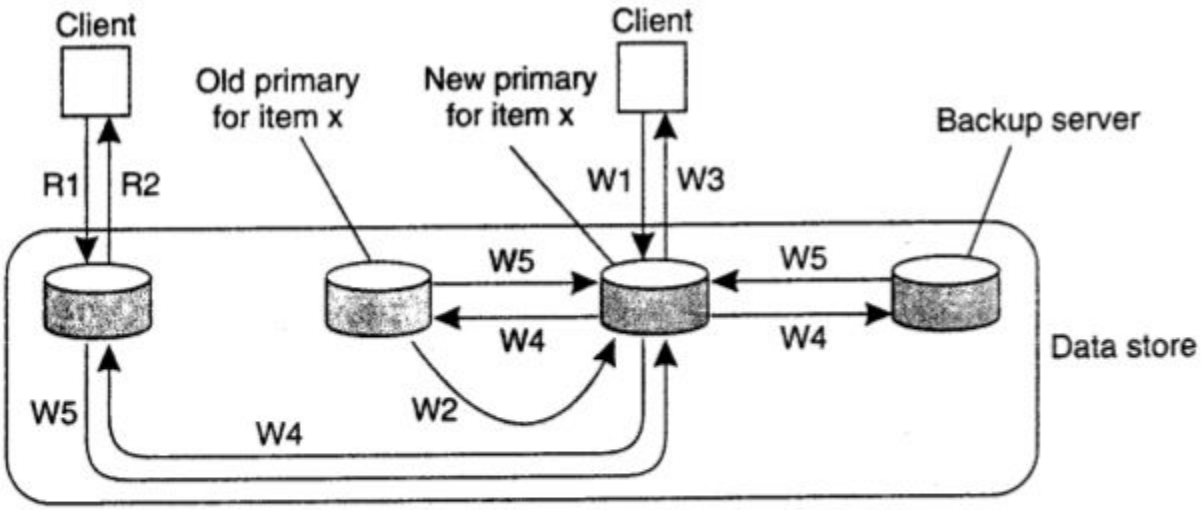
PBP: Remote Write Protocol



W1. Write request
W2. Forward request to primary
W3. Tell backups to update
W4. Acknowledge update
W5. Acknowledge write completed

R1. Read request
R2. Response to read

PBP: Local Write Protocol



W1. Write request
W2. Move item x to new primary
W3. Acknowledge write completed
W4. Tell backups to update
W5. Acknowledge update

R1. Read request
R2. Response to read

RBP: Active Replication

- Write operations can be carried out at multiple replicas instead of only one.
- An operation/update is forwarded to all replicas
- And is propagated by means of the write operation that causes the update.
- Again there is an ordering problem.

References

- Andrew S. Tanenbaum and Van Steen "Distributed Systems", PHI, Second Edition,2014
- http://www.netlib.org/utk/lsi/pcwLSI/text/node444.html.
- Google.

Resources

- https://github.com/it-h1/7sem/DistributedSystems