Distributed Systems

Spring Semester 2020

Lecture 19: Pub/Sub (Wormhole)

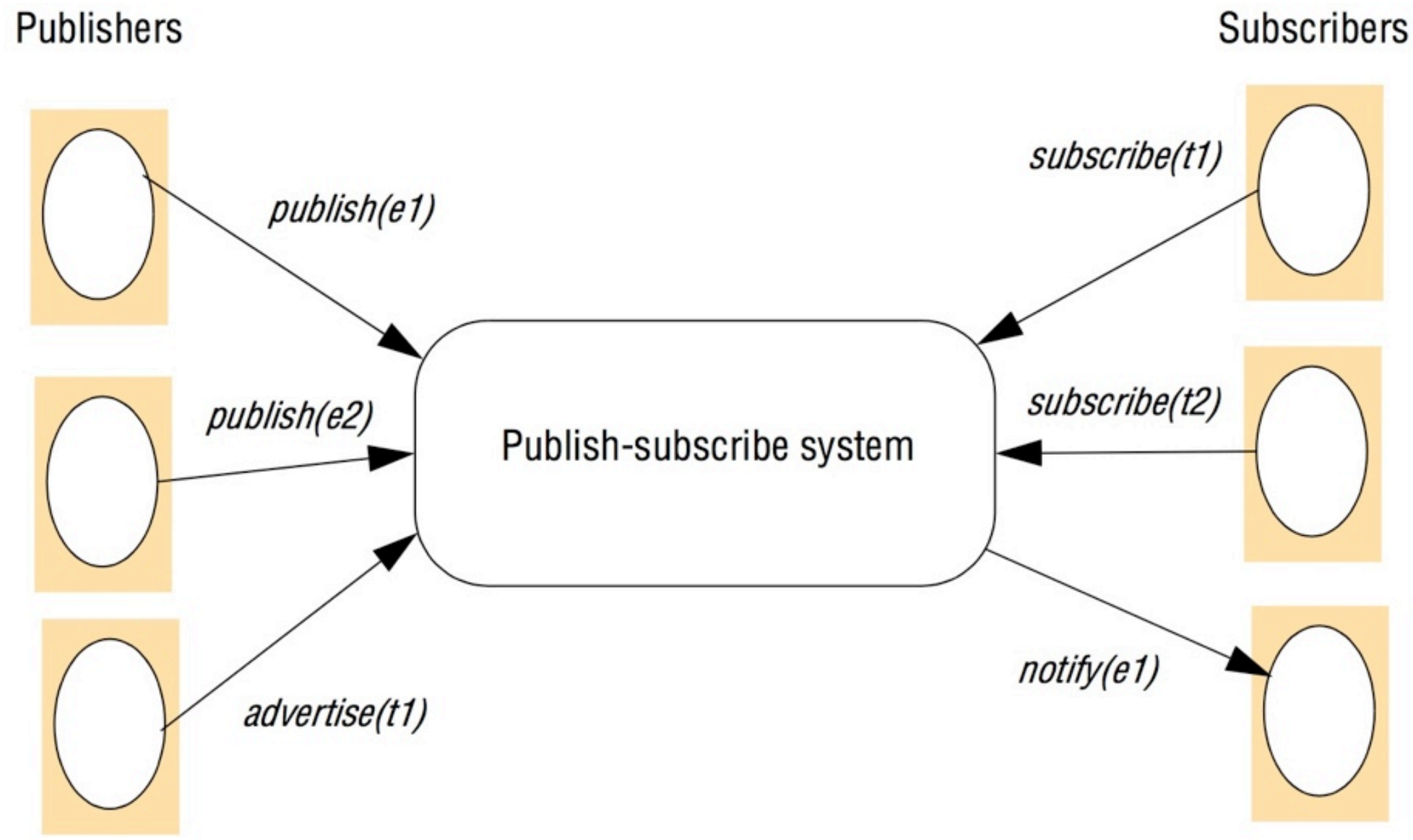
John Liagouris
liagos@bu.edu

Why this paper

- Wide-area Pub-Sub Infrastructure in more detail
  - A common building block you will see in practical distributed systems in the wild
  - Today: FaceBook's Wormhole case study
  - But many other implementations exist, e.g., Apache Kafka, Google Cloud Pub/Sub, Yahoo Message Broker (used in PNUTS), and many more.

Publish-subscribe

Distributed Event Based Systems

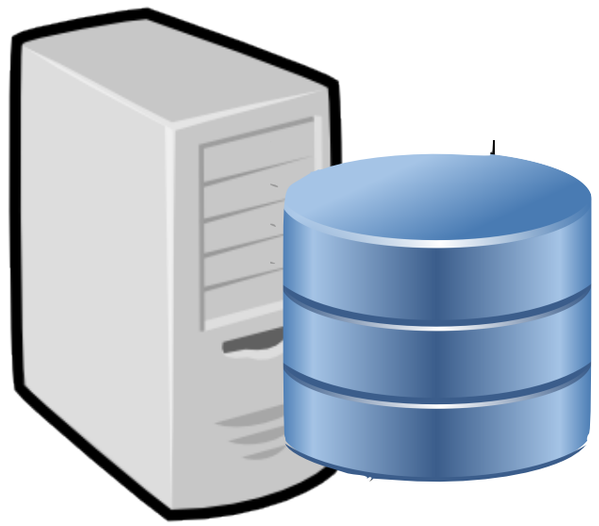


Let's think about what lead FB to Wormhole

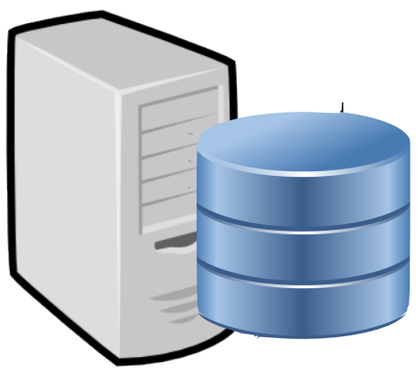
- What is the life cycle of a web site as it has to scale up with growing load?

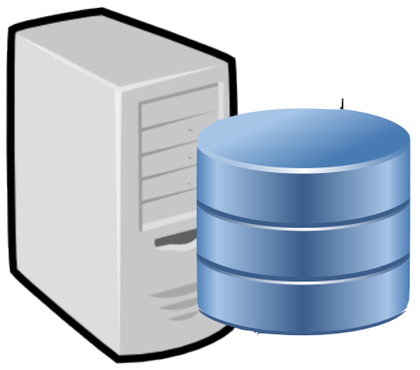
Single WS + DB

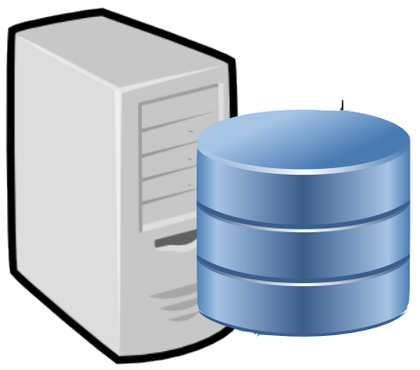
Multiple WS + Single DB



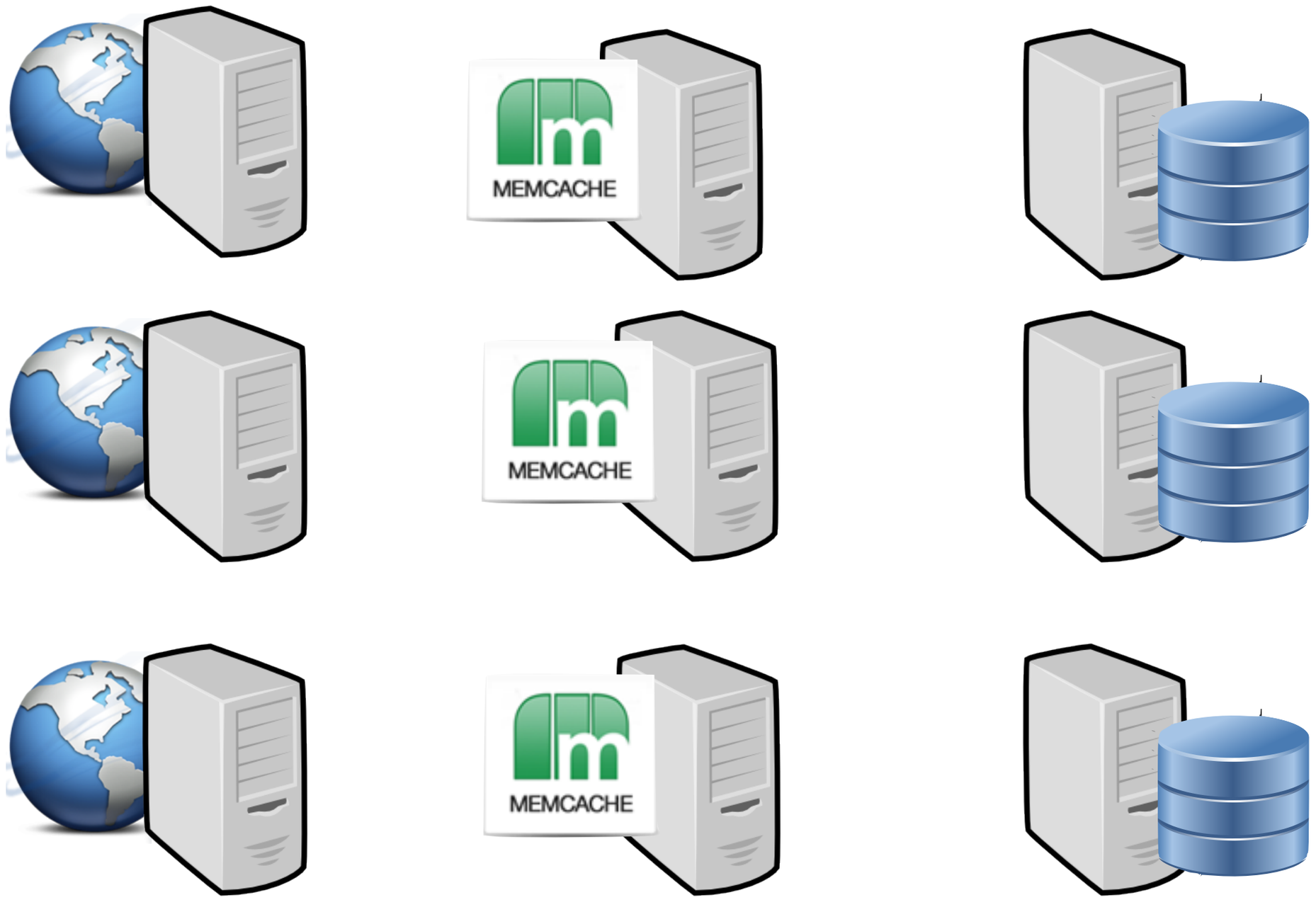
Multiple WS + MultiShard DB



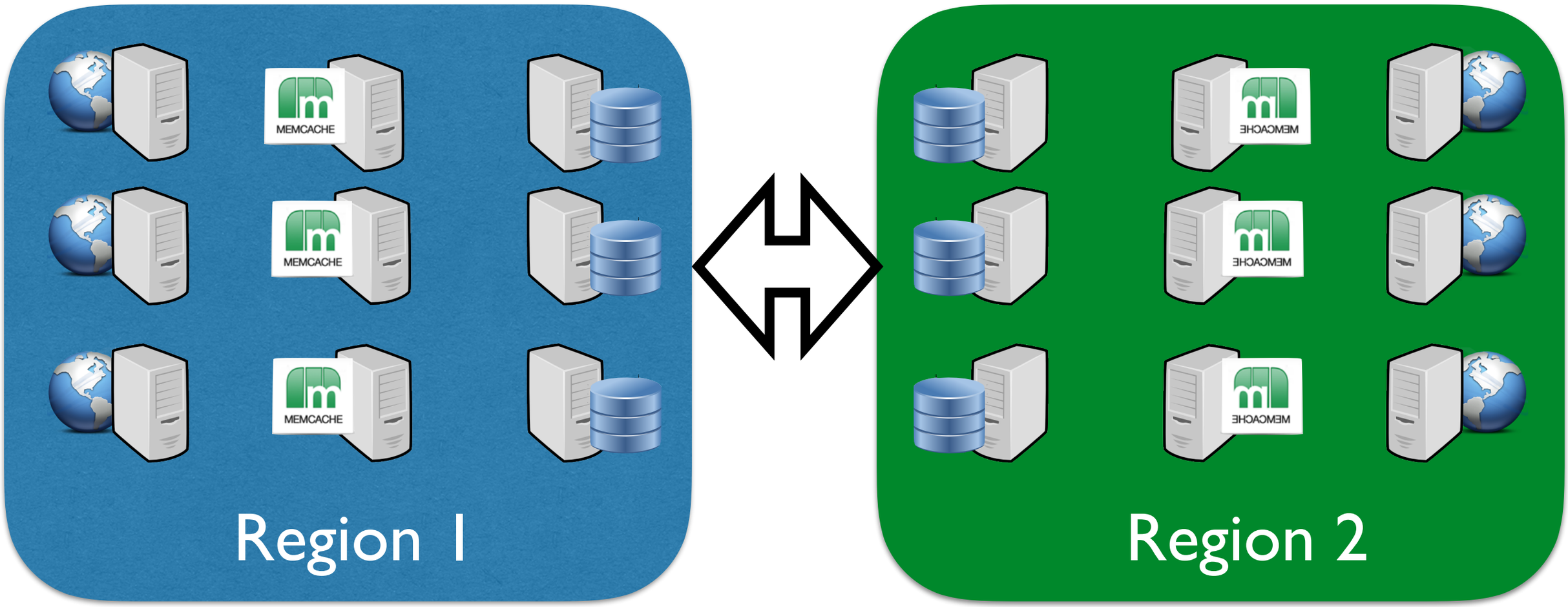




Multiple WS + Multiple MemCache + MultiShard DB



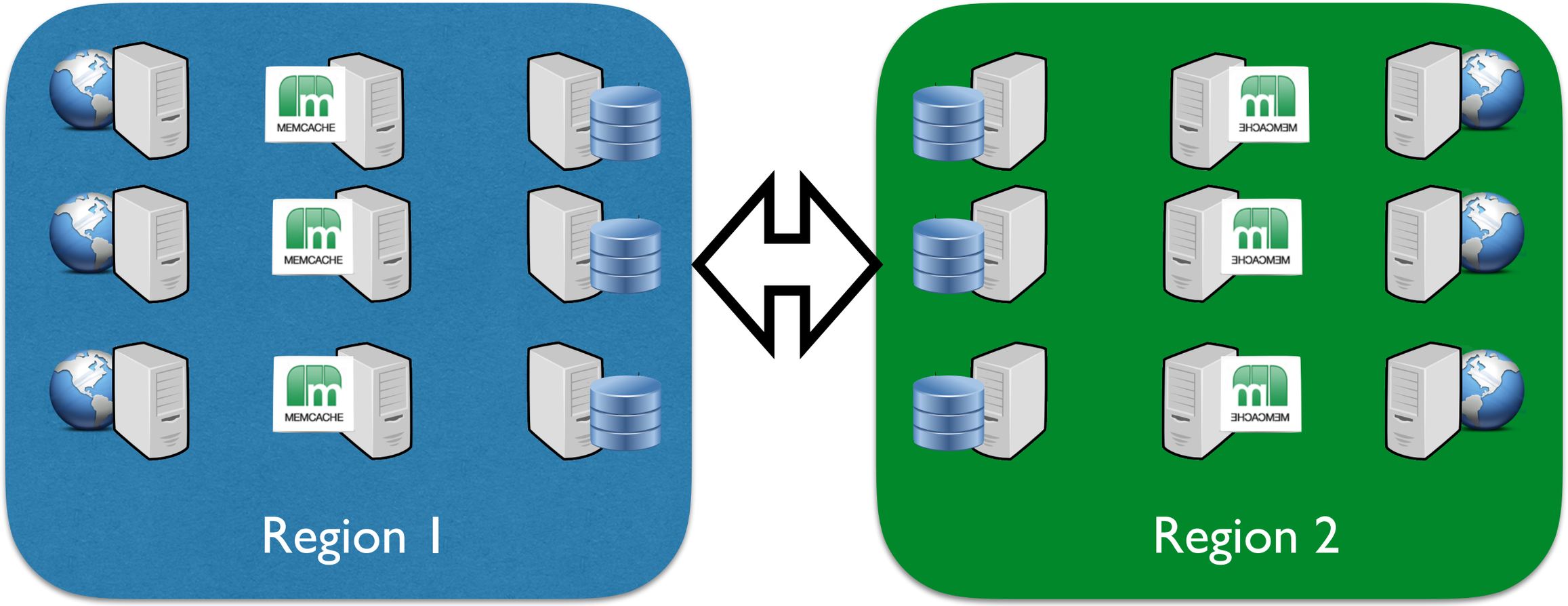
FB Infrastructure



DB Replication

How to arrange that DB in different regions get updated?

- **Master DB receives all writes (similar to PNUTS)**
- **Adds entry to transaction log**
- **Replicates transaction log to slaves**

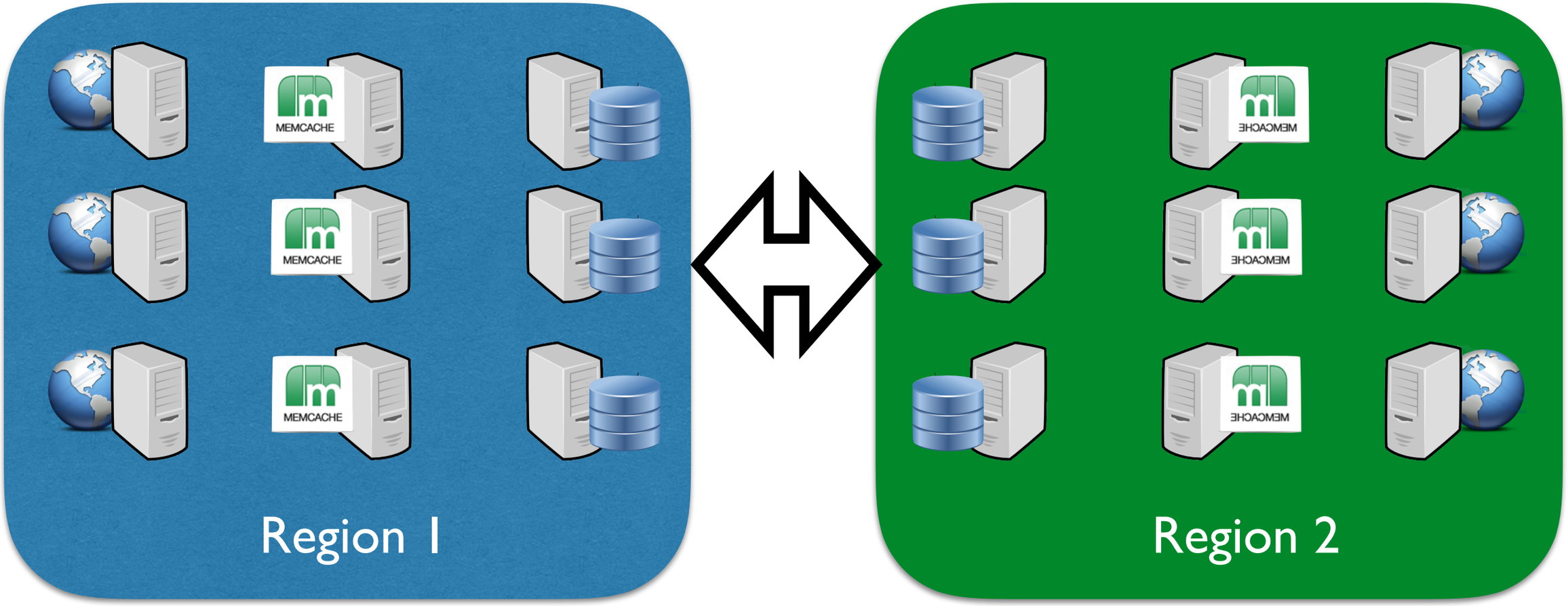


What about MC

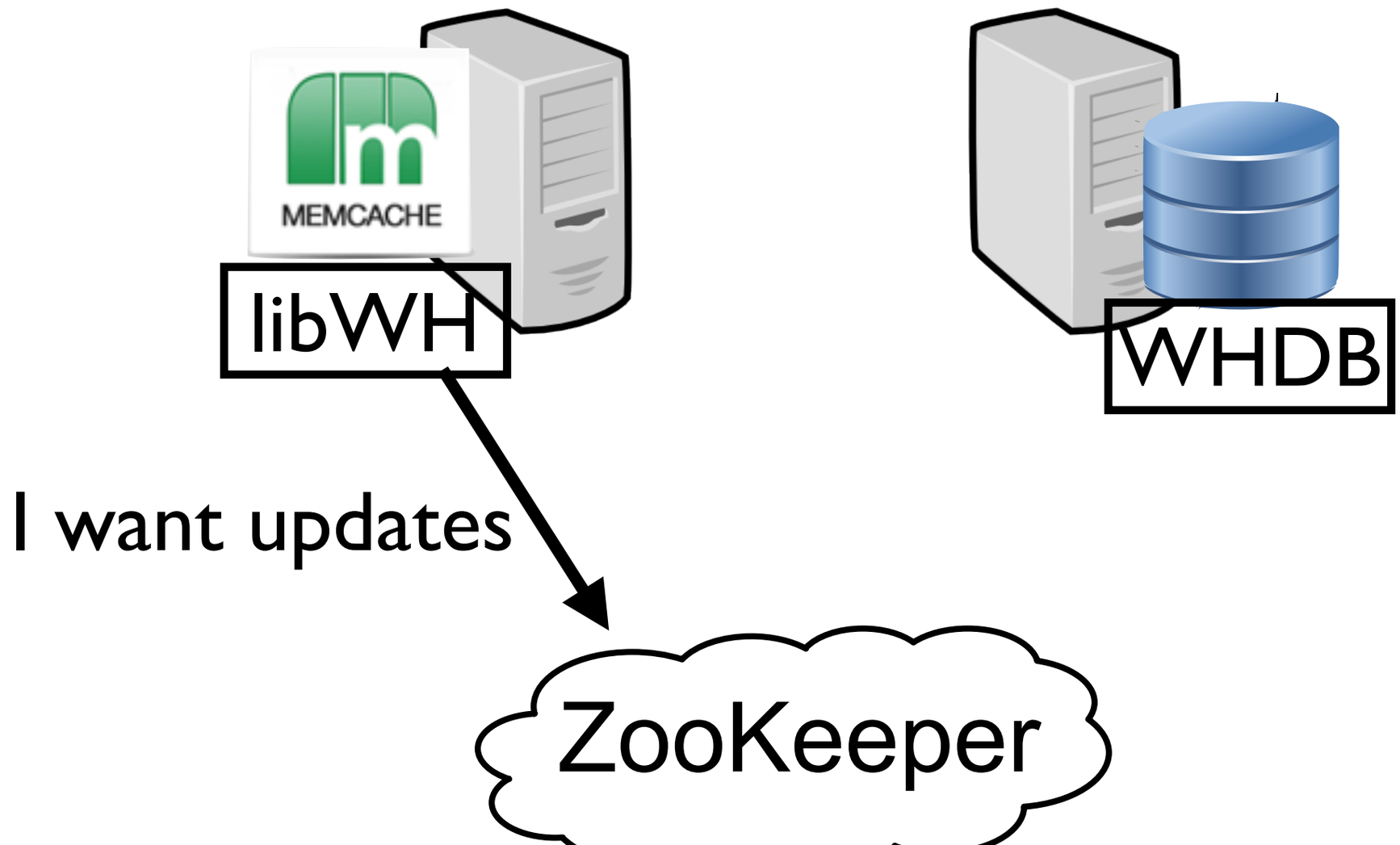
Need to invalidate/update mc entry after write:

Option 1: MC in remote region polls its local DB increases read load on DB.

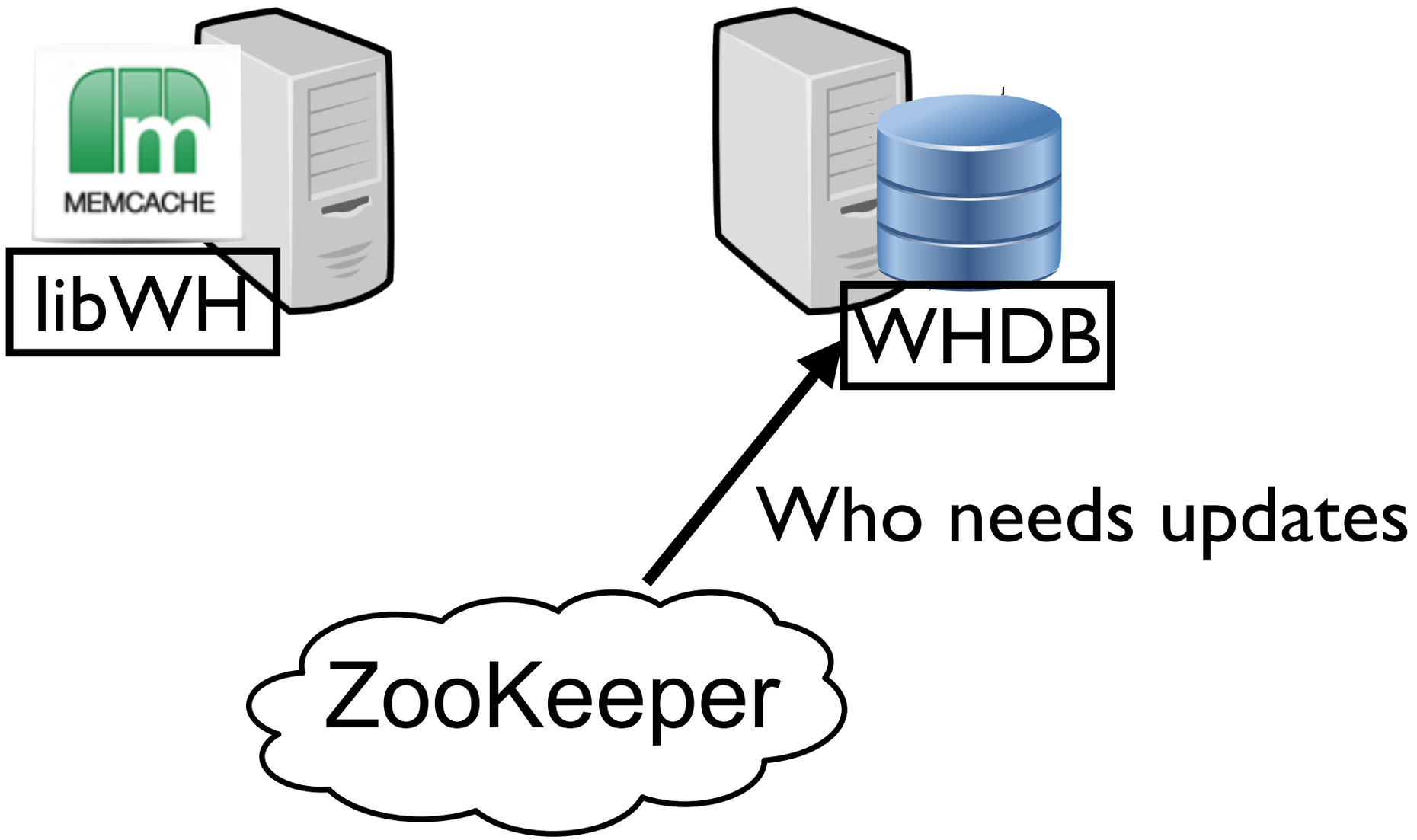
Option 2: Wormhole Pub/Sub



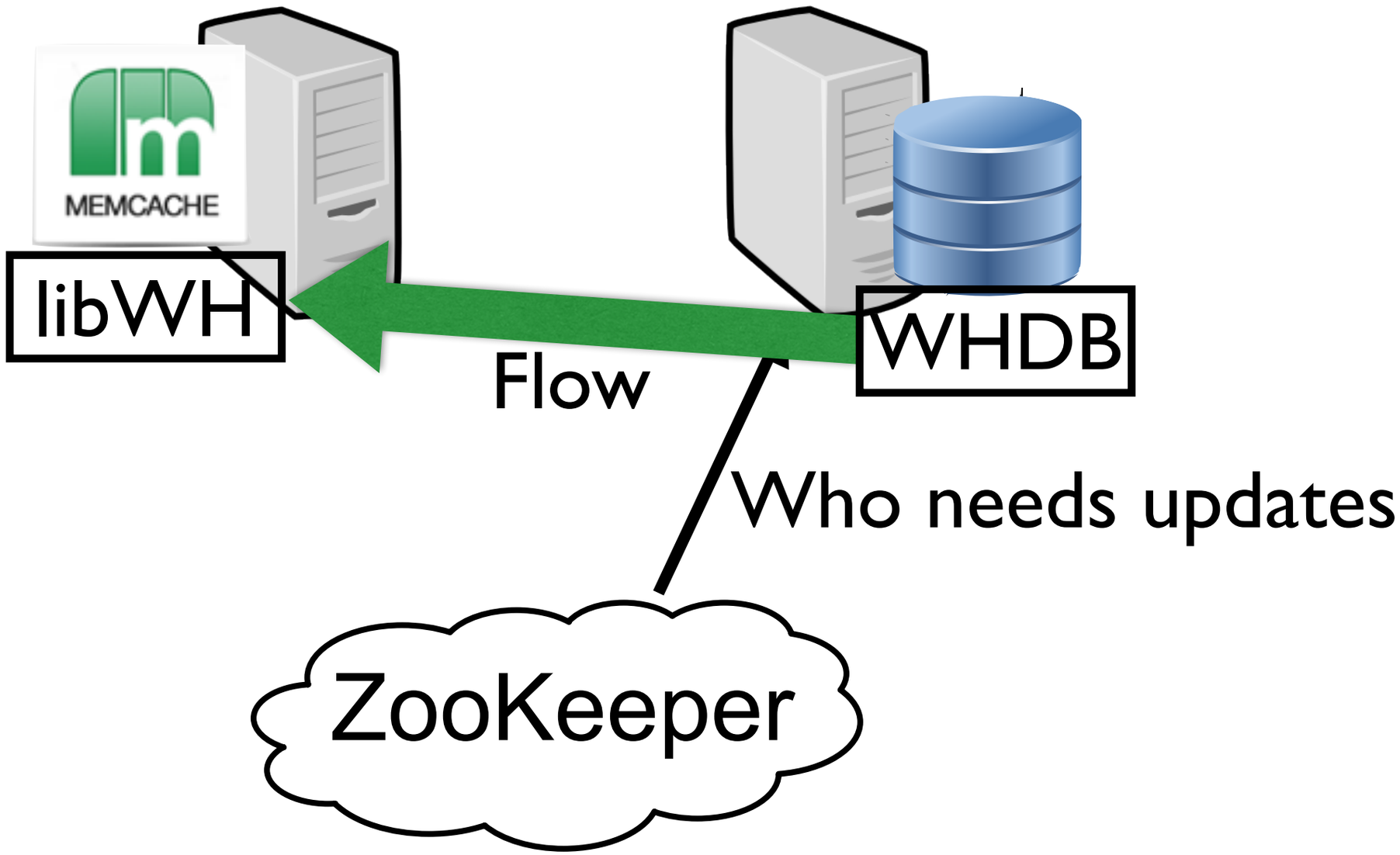
FB Memcache : Pub/Sub



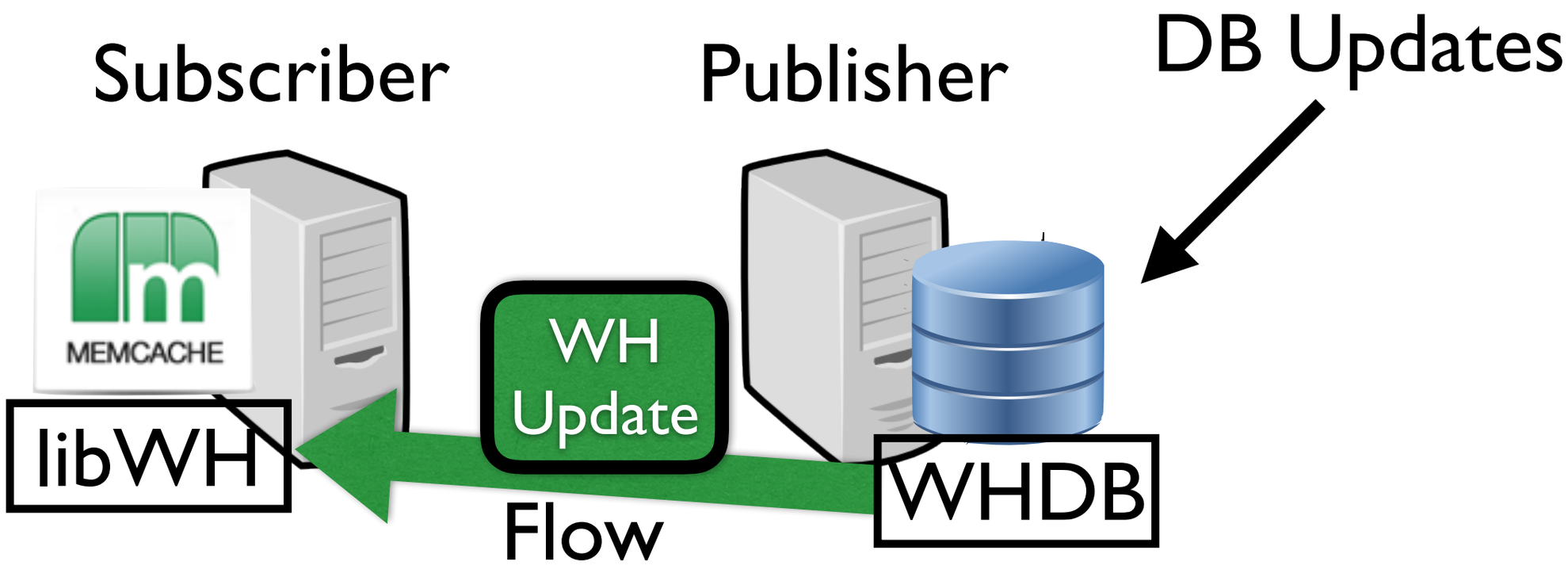
FB Memcache : Pub/Sub



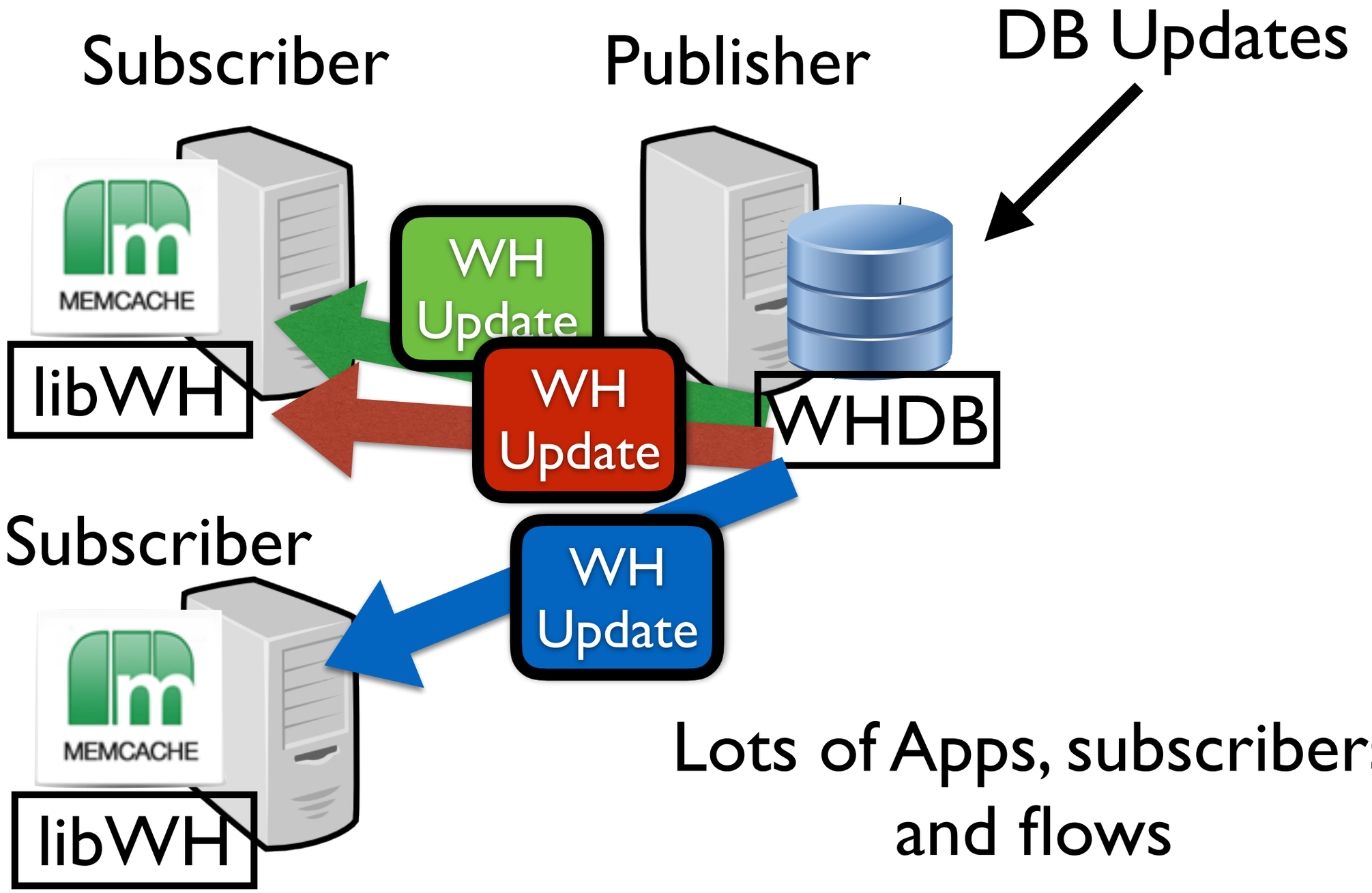
FB Memcache : Pub/Sub



FB Memcache : Pub/Sub



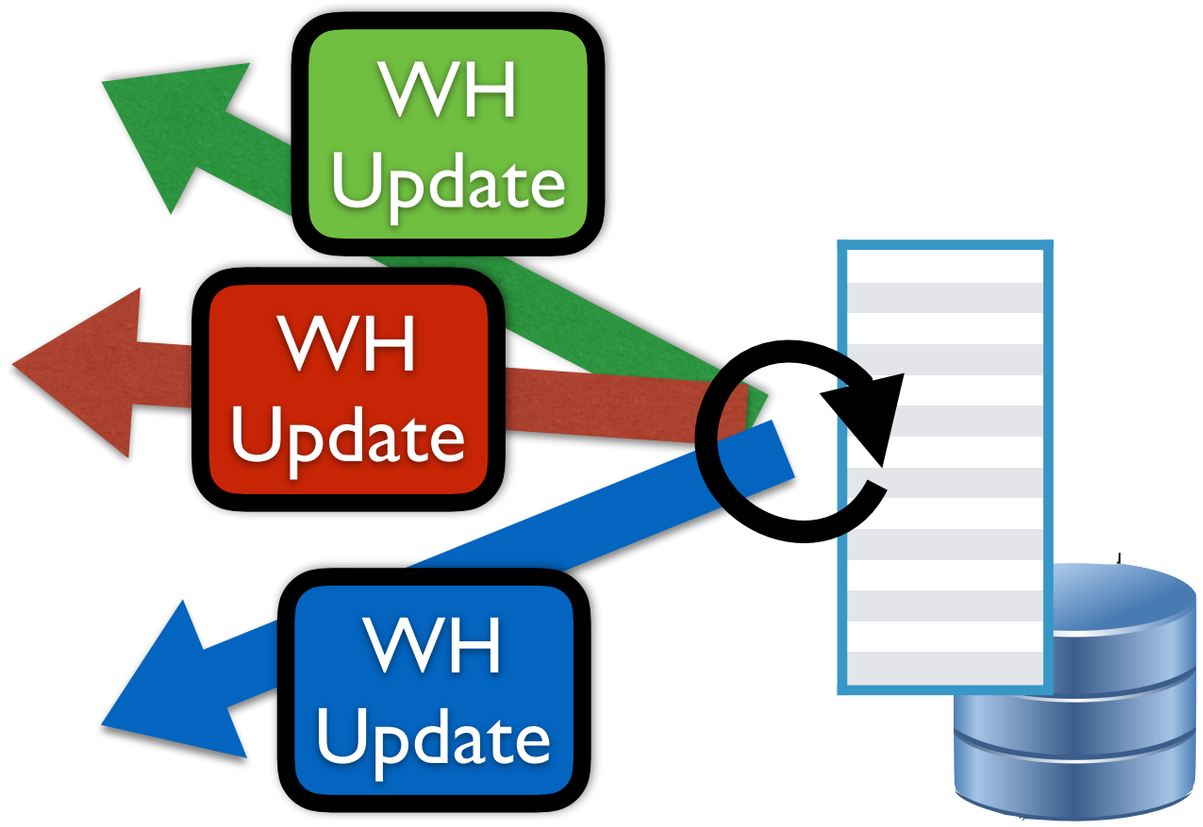
FB Memcache : Pub/Sub



Lots of Apps, subscribers and flows

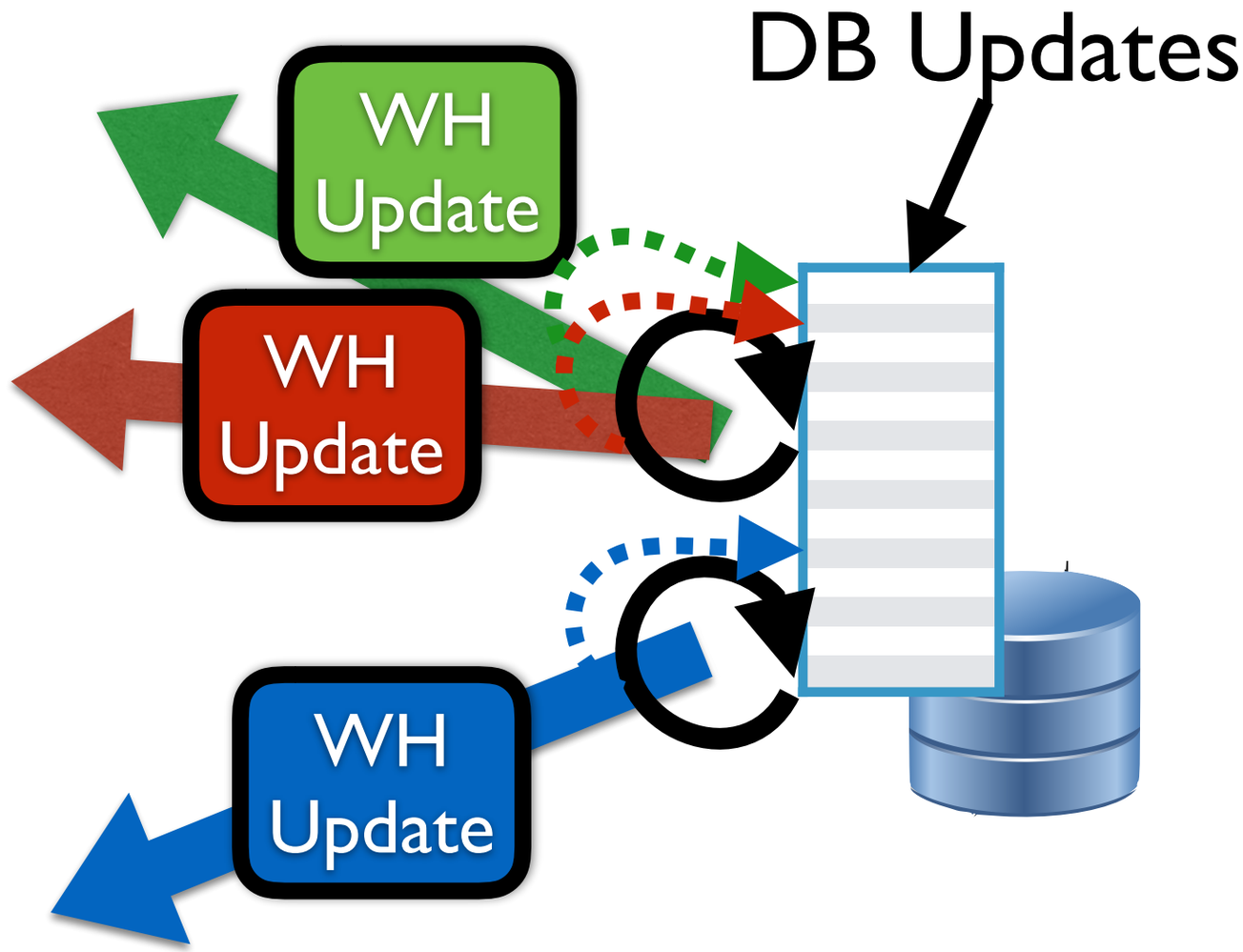
Optimization: Caravan

- Problem: Each flow requires reading of DB
  - Per-flow reader — lots of overhead
  - Single Shared reader for all flows
    - What is the problem with this?

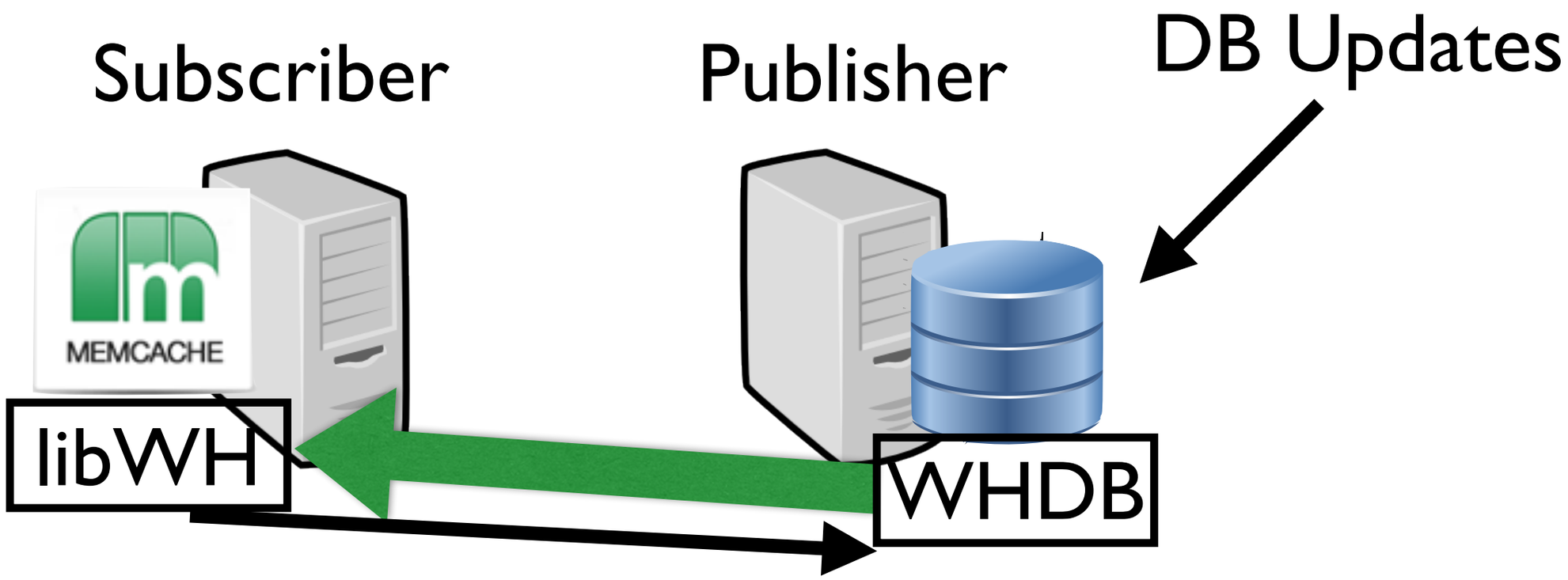


Optimization: Caravan

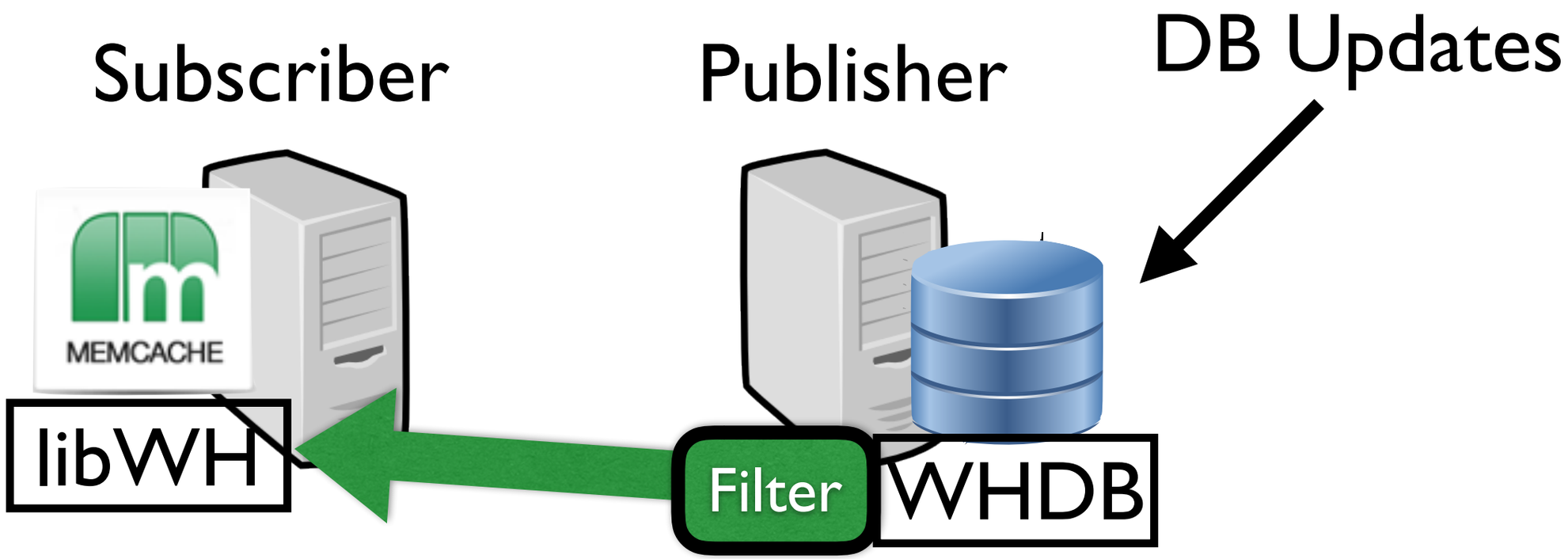
- Problem: Each flow requires reading of DB
  - Group flow's based on their progress into small number of readers — Caravans
- Allows tradeoff in overhead and latency



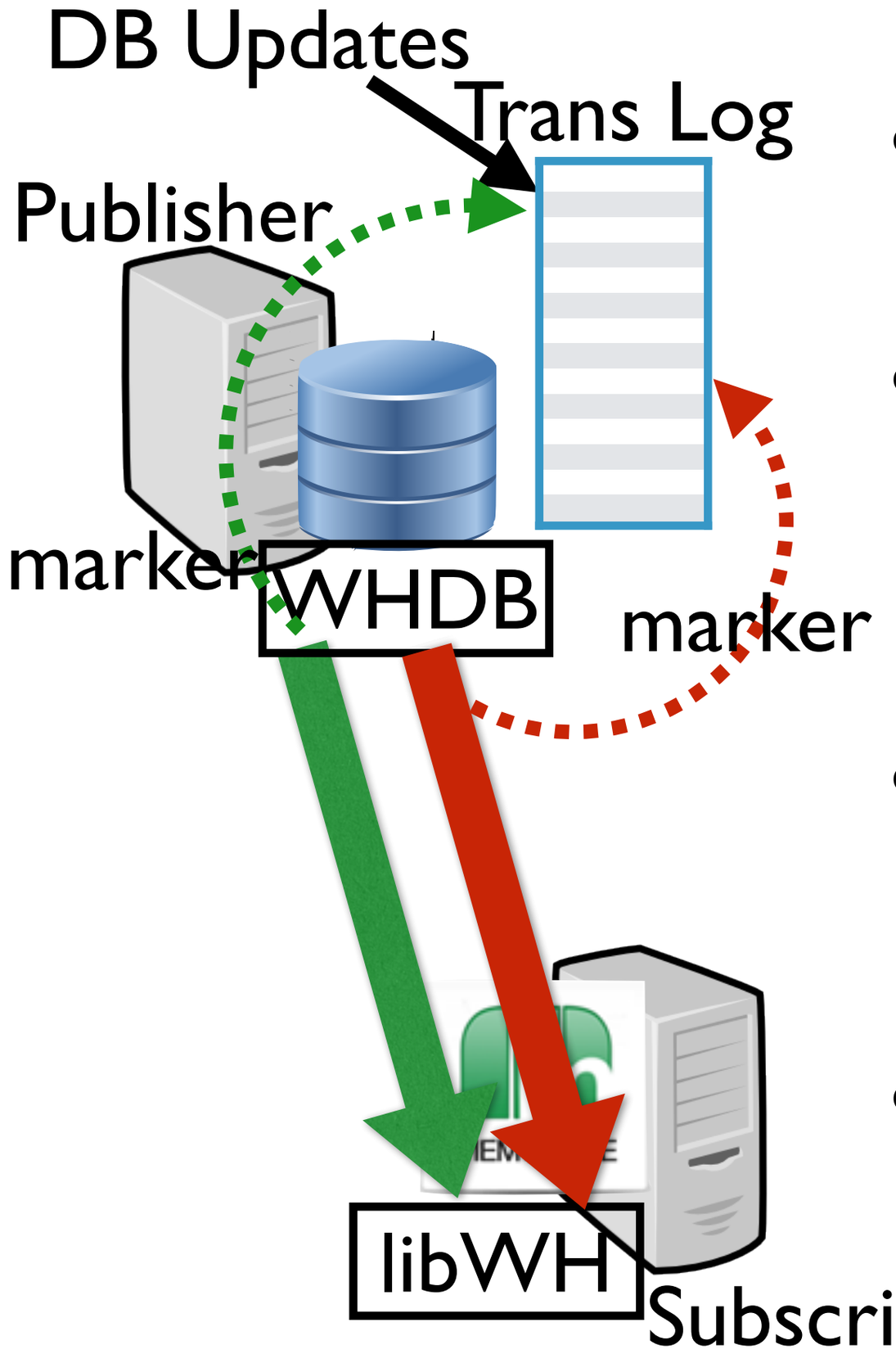
FB Memcache : Pub/Sub



FB Memcache : Pub/Sub

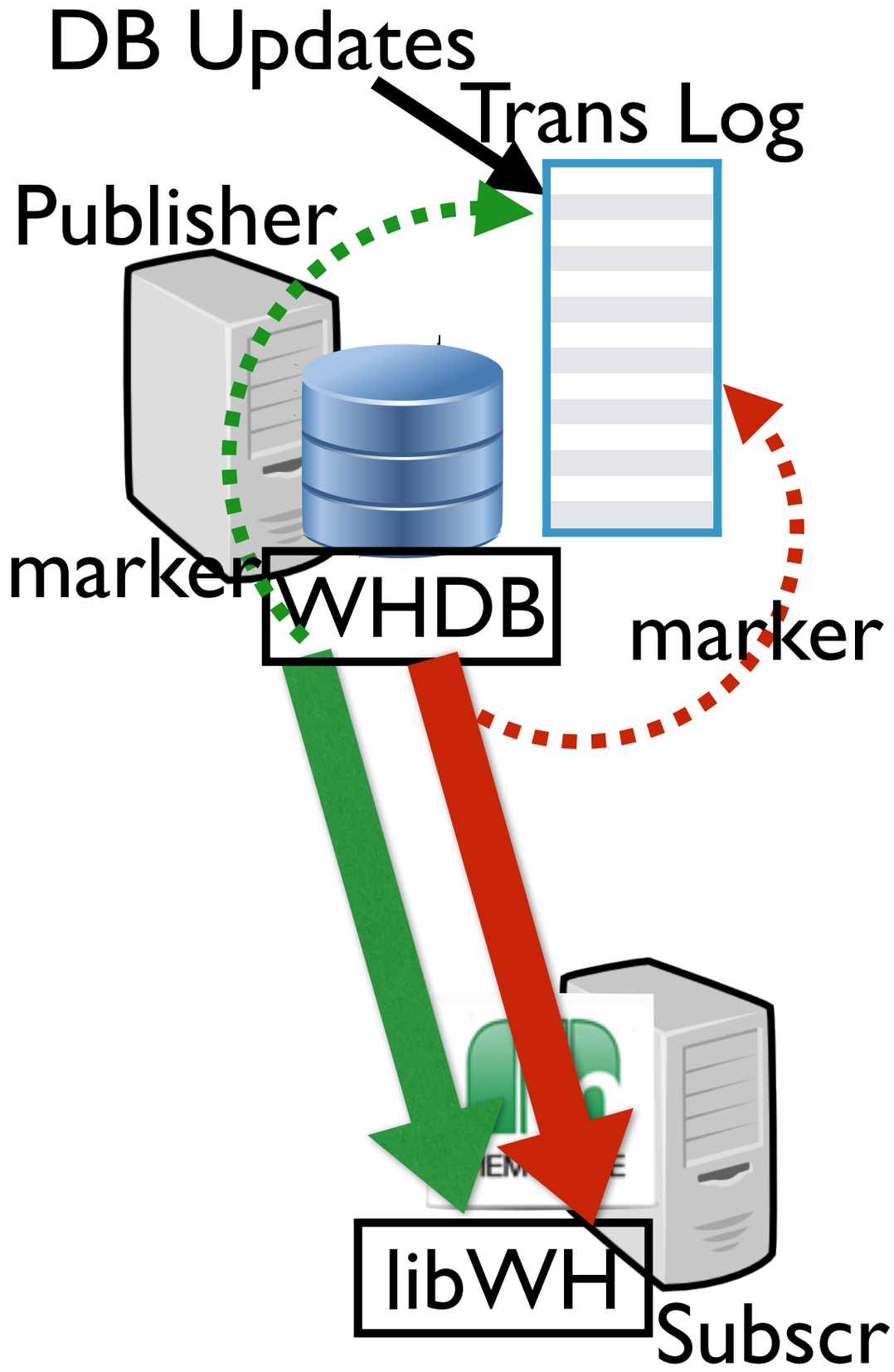


Delivery Semantics



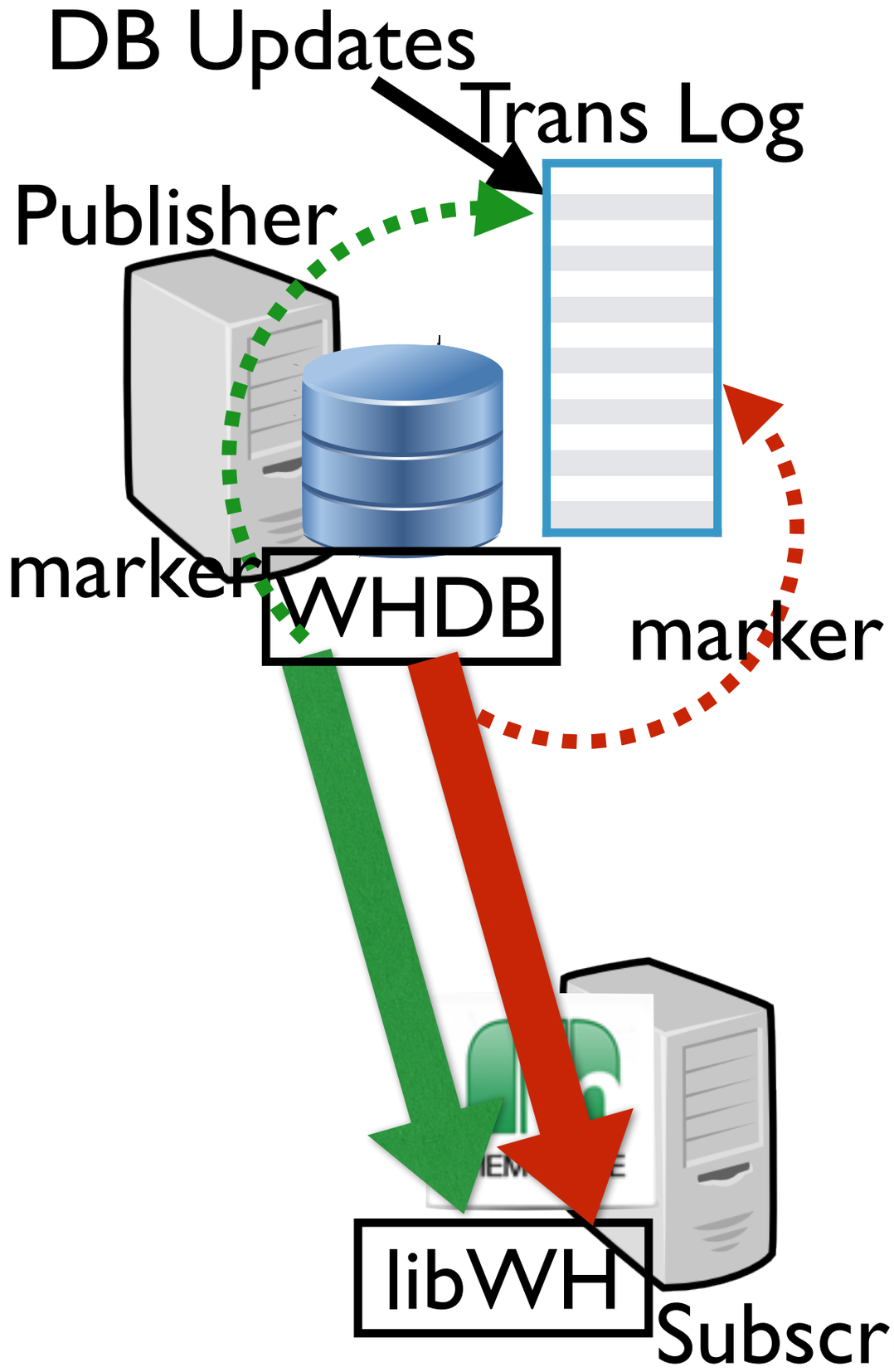
- Updates on a flow are delivered in order
- Publisher maintains per subscriber a "Data Marker" — Sequence number of an update in transaction log — tracks what subscriber has received
- Publisher ask sub periodically for what it has received — marker is lower bound
- Updates are delivered "at least once" — publisher persists marker

Delivery Semantics



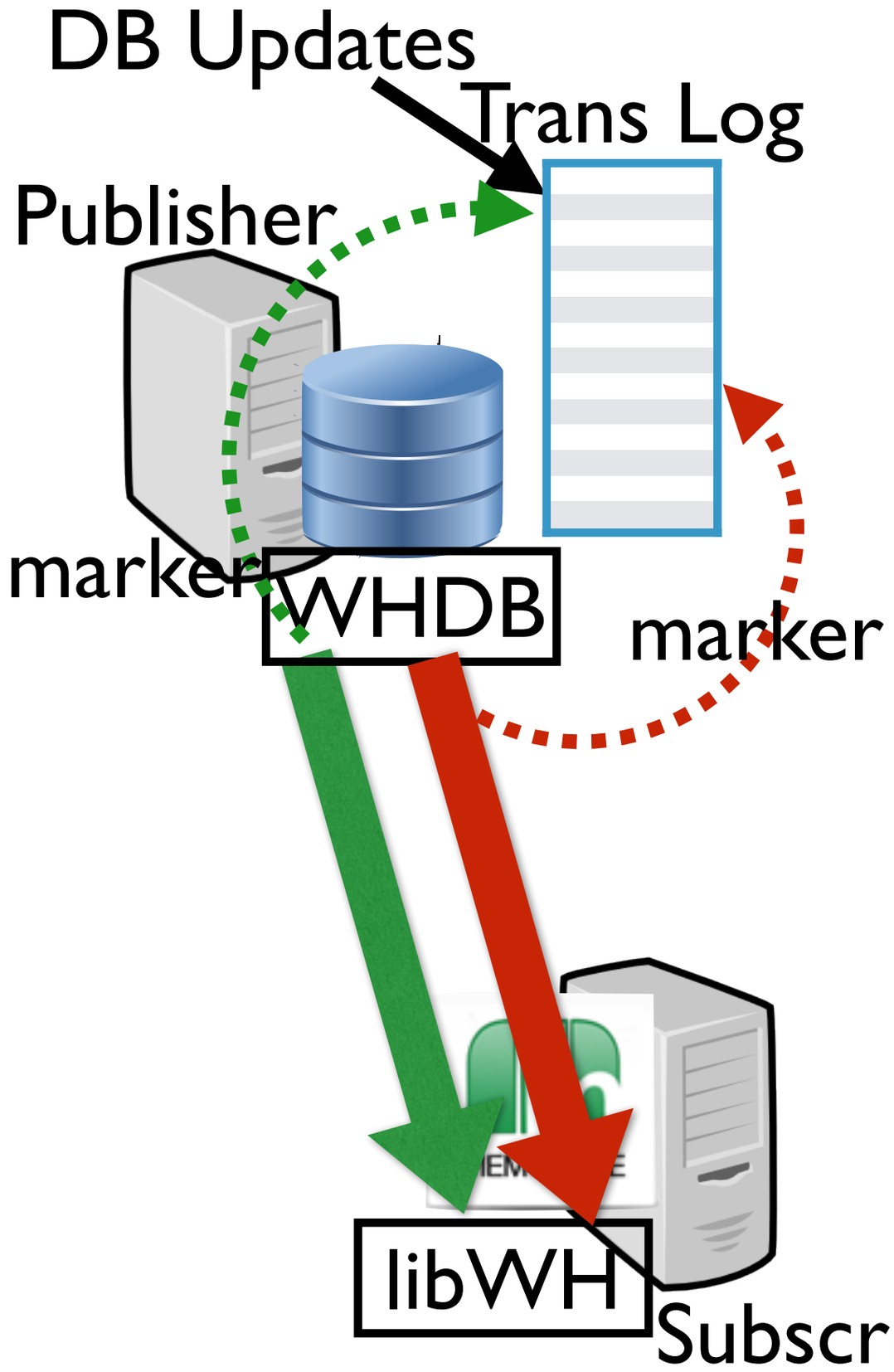
- If publisher fails start sending from last marker Publisher
- Q: How do subscribers deal with and update delivered several times?

Delivery Semantics



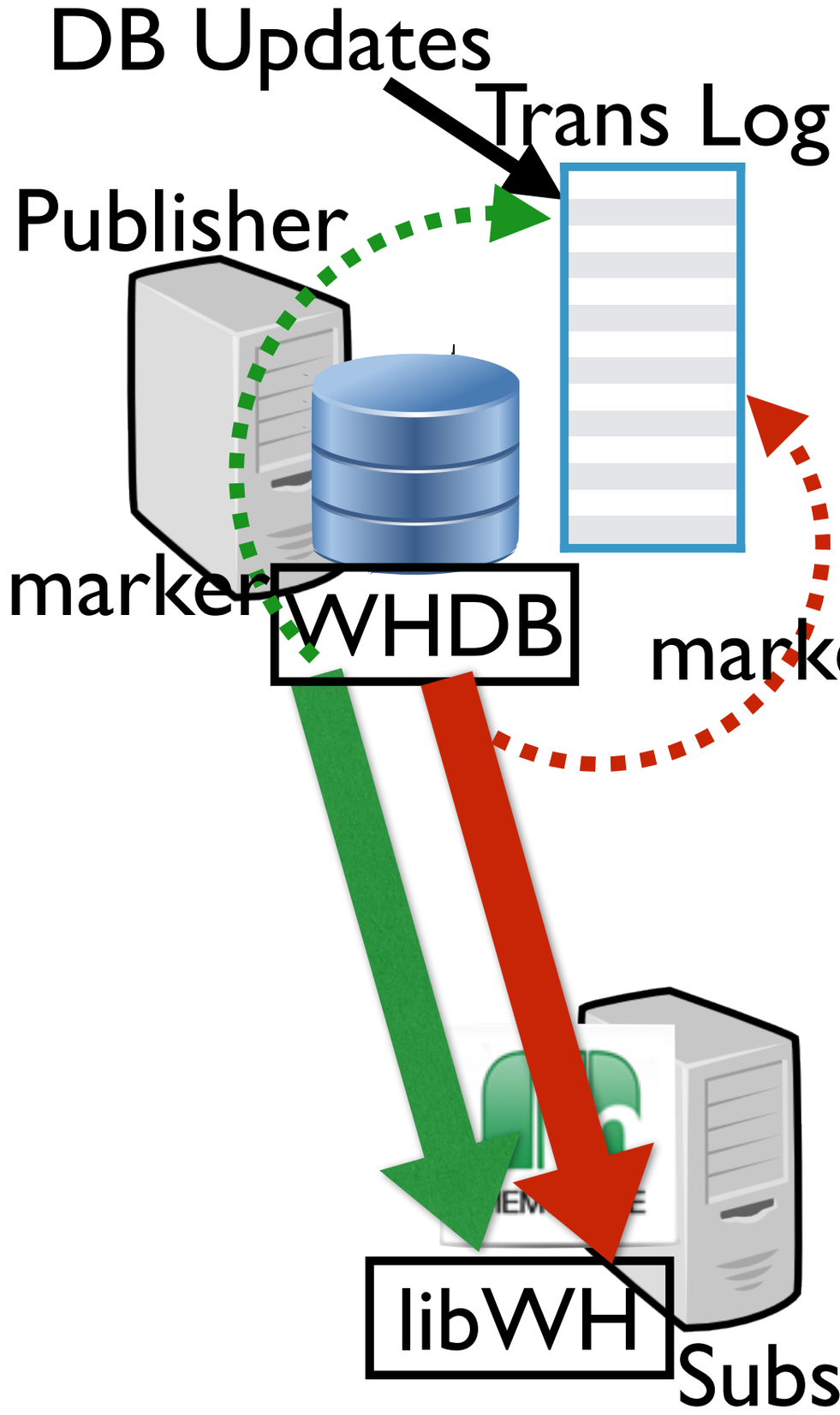
- If publisher fails start sending from last marker Publisher
  - Q: How do subscribers deal with and update delivered several times?
    - A: No problem for cache
    - A: App must do duplicate filtering
- Q: Can an update be never delivered?

Delivery Semantics



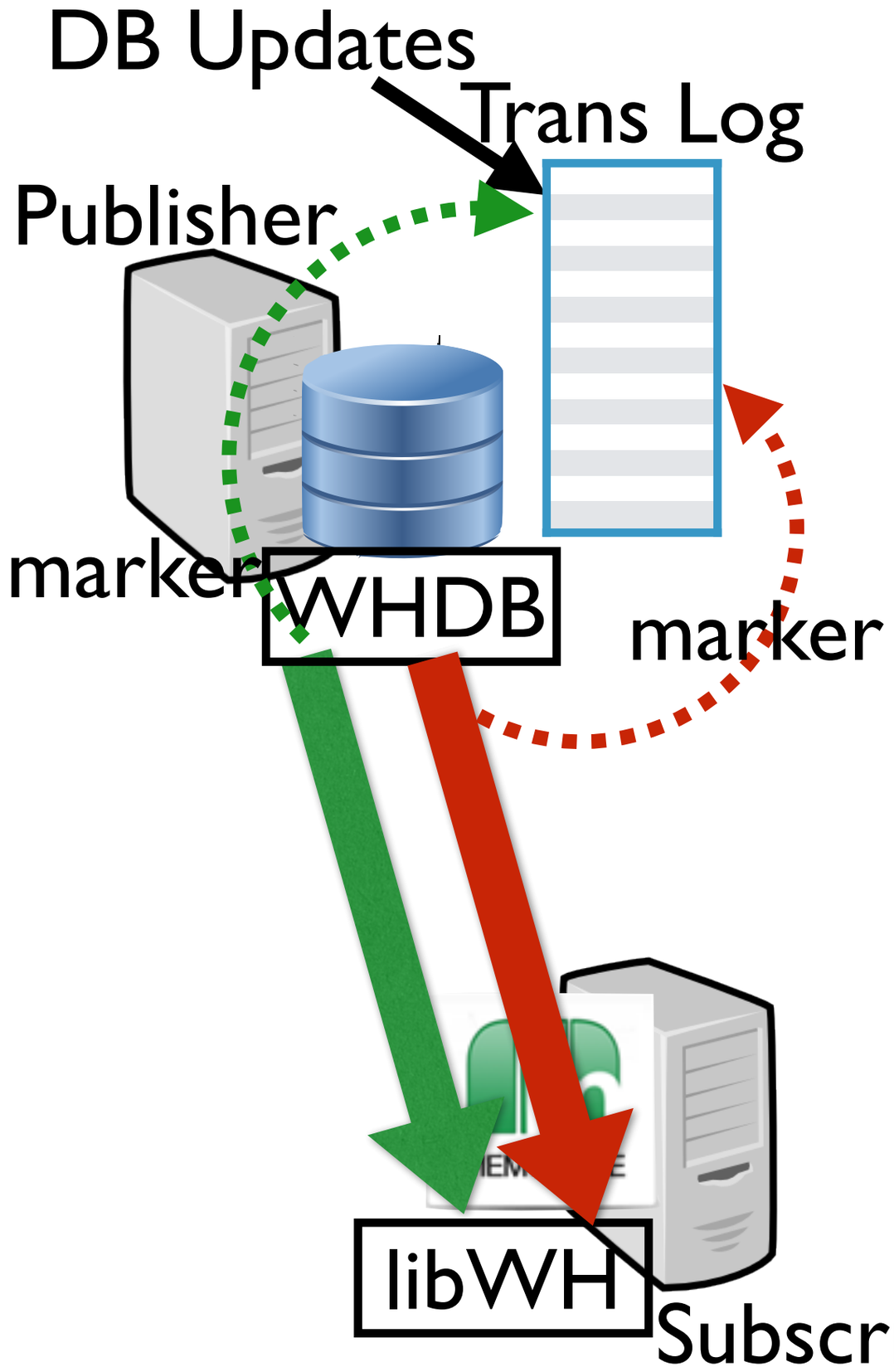
- If publisher fails start sending from last marker Publisher
  - Q: How do subscribers deal with an update delivered several times?
    - A: No problem for cache
    - A: App must do duplicate filtering
  - Q: Can an update be never delivered?
    - A: Yes - Trans log may have been truncated (data in log 1-2 days)

Delivery Semantics



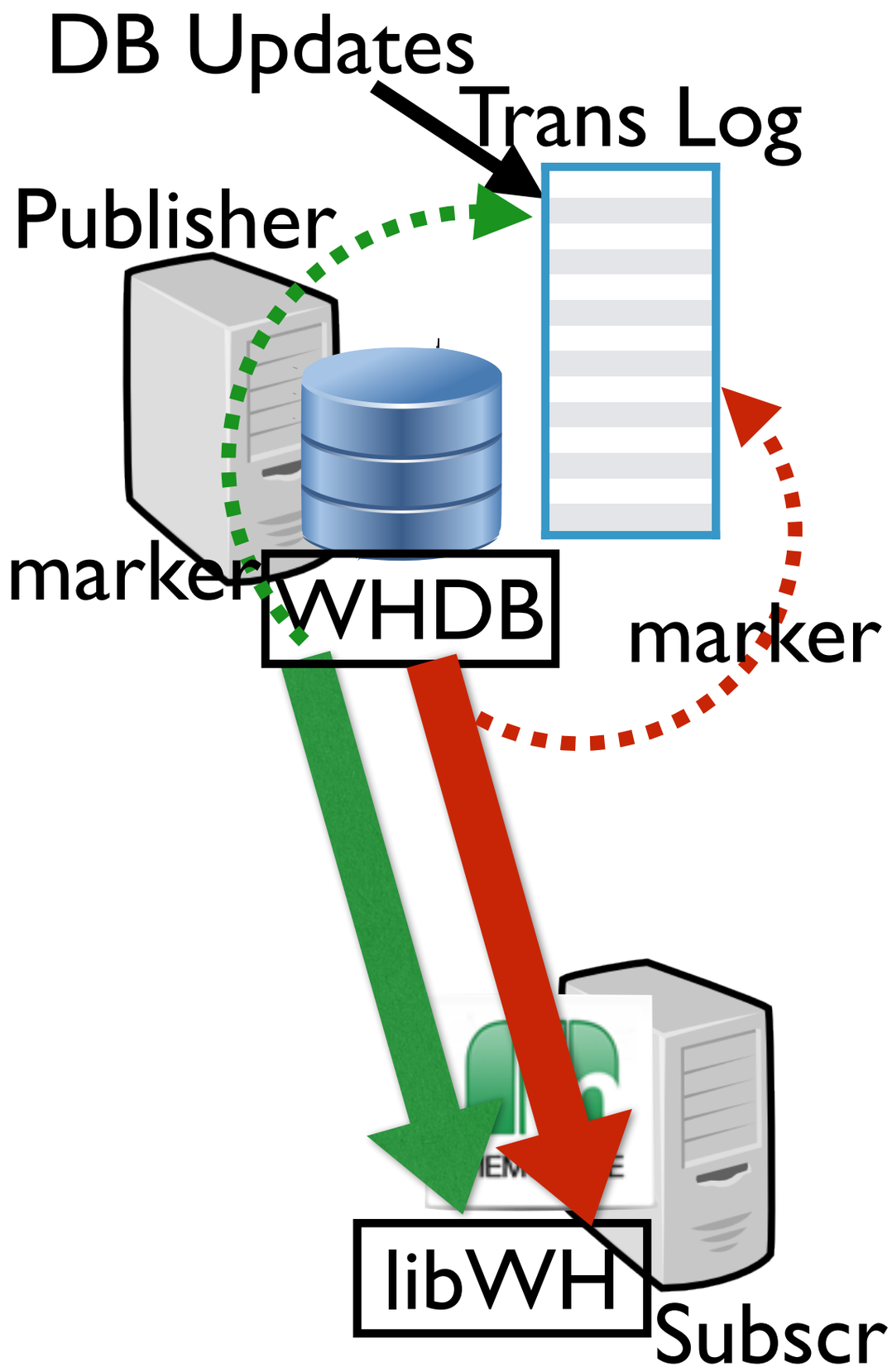
- Q: why does subscriber not keep track of marker?

Delivery Semantics



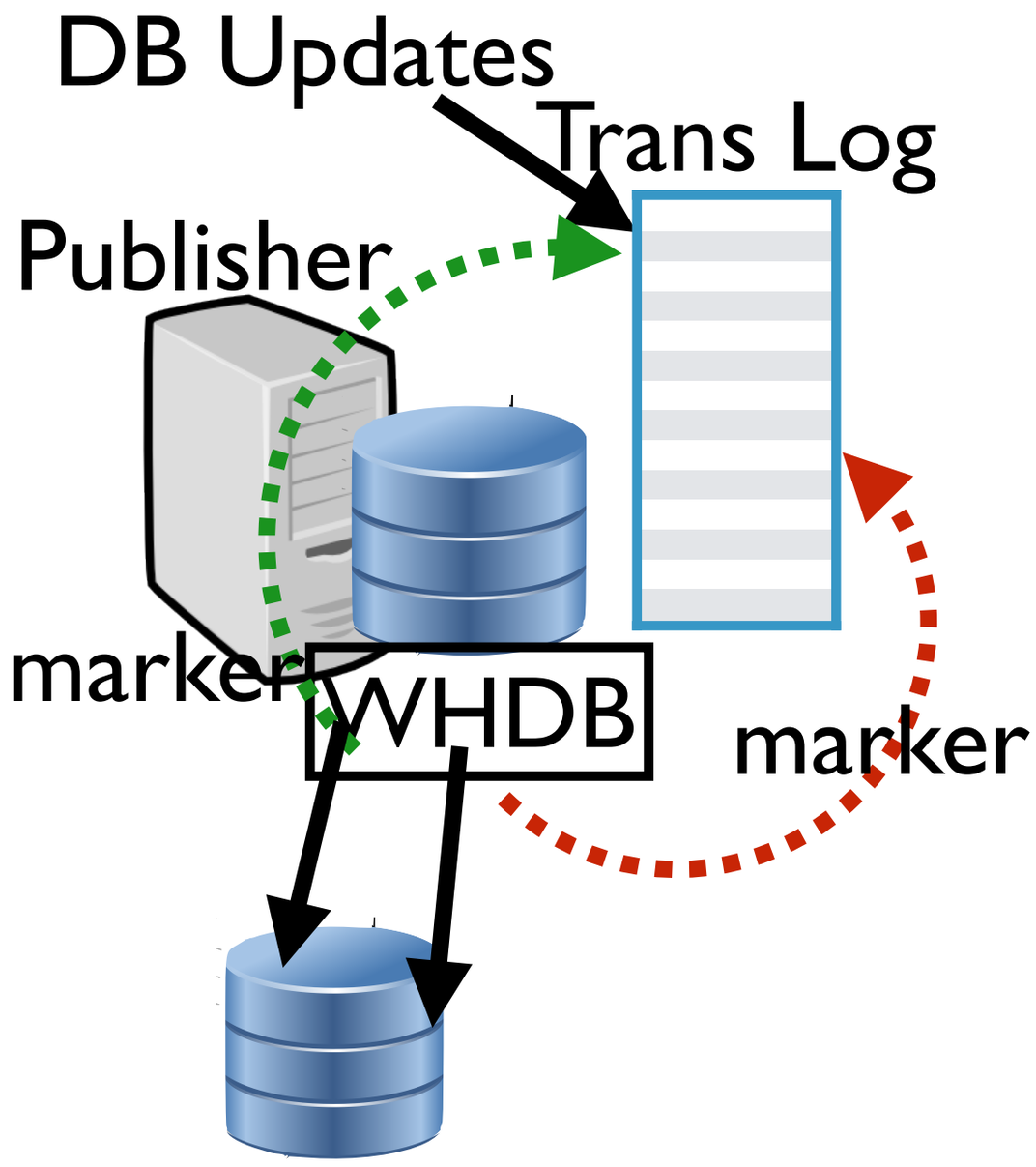
- Q: why does subscriber not keep track of marker?
- A: FB wants subscribers to be stateless
- Q: why are markers periodically acked by subscribers

Delivery Semantics



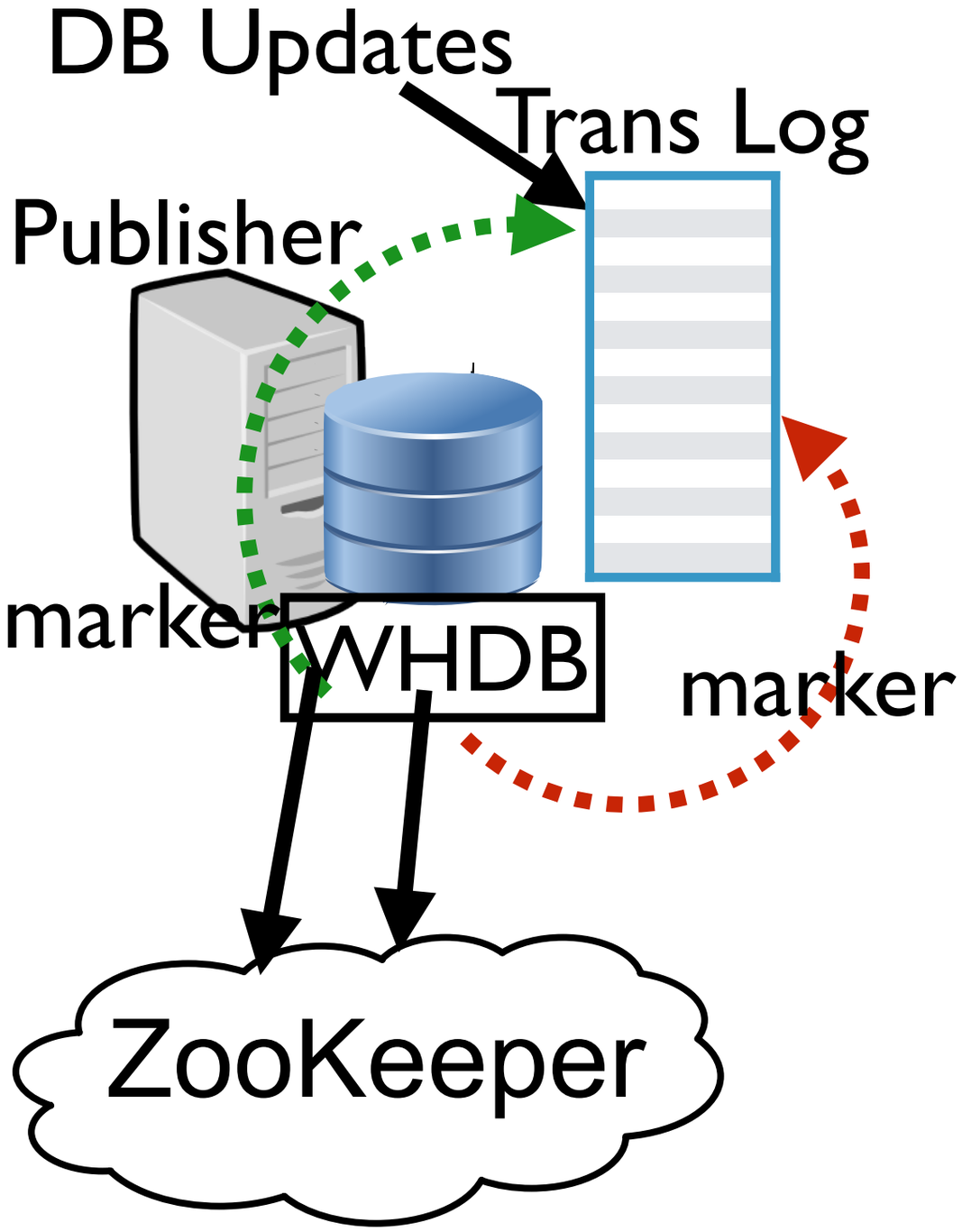
- Q: why does subscriber not keep track of marker?
  - A: FB wants subscribers to be stateless
- Q: why are markers periodically acked by subscribers
  - A: Expensive to ack each update
  - A: Uses TCP for delivery — don't have to worry about packet loss

Single Copy Reliable Delivery



- App subscribes to a single copy Data Set
- SCRD: publisher stores marker in local persistent storage
- if storage is unavailable, cannot fail over to new publisher
  - caches will be stale
  - if storage fails, lose marker

Multiple Copy Reliable Delivery



- Geo-replicated dataset
- MCRD: publishers stores marker in Zookeeper — if publisher fails, another publisher can take over
- read last marker from zookeeper
- implementation challenge: format of marker — replicas of same log have different binary format
- solution: "logical" positions