Slogger: Scalable, Near-Zero Loss Disaster Recovery for Distributed Data Stores

Ahmed Alquraan[1], Alex Kogan[2], Virendra Marathe[2], Samer Al-Kiswany[1]

[1]University of Waterloo

[2]Oracle Labs

Disaster Recovery Overview

recode

TRENDING TOPICS WRITERS PODCASTS EVENTS RECODE 100

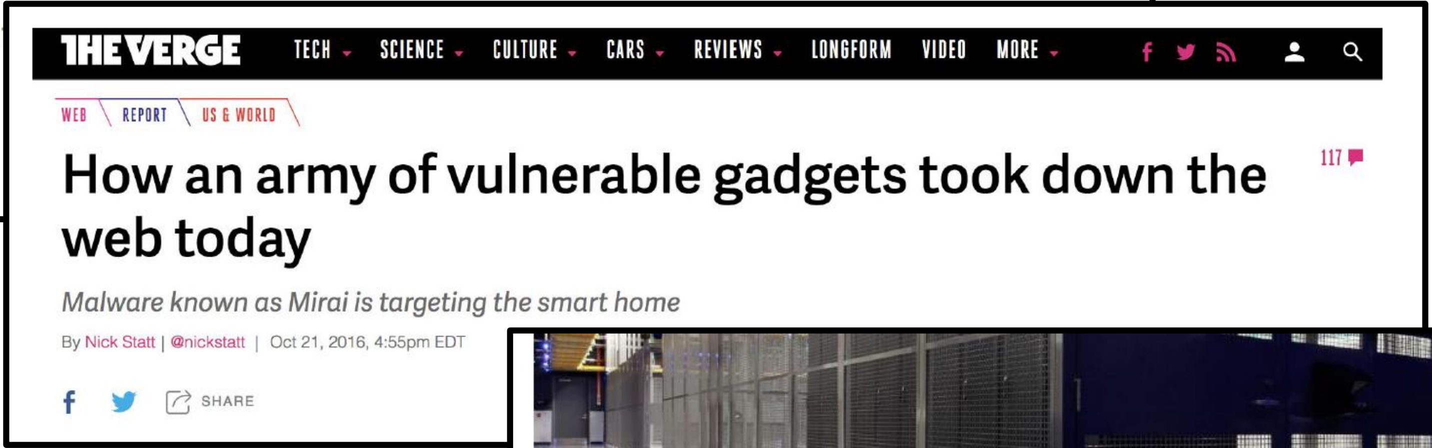
AMAZON CLOUD COMPUTING

Amazon's massive AWS outage was caused by human error

One incorrect command and the whole internet su

By Jason Del Rey | @DelRey | Mar 2, 2017, 2:20pm EST

SHARE



THE VERGE

TECH SCIENCE CULTURE CARS REVIEWS LONGFORM VIDEO MORE

WEB REPORT US & WORLD

How an army of vulnerable gadgets took down the web today

117

Malware known as Mirai is targeting the smart home

By Nick Statt | @nickstatt | Oct 21, 2016, 4:55pm EDT

SHARE

MANAGE > UPTIME

Equinix Power Outage One Reason Behind AWS Cloud Disruption

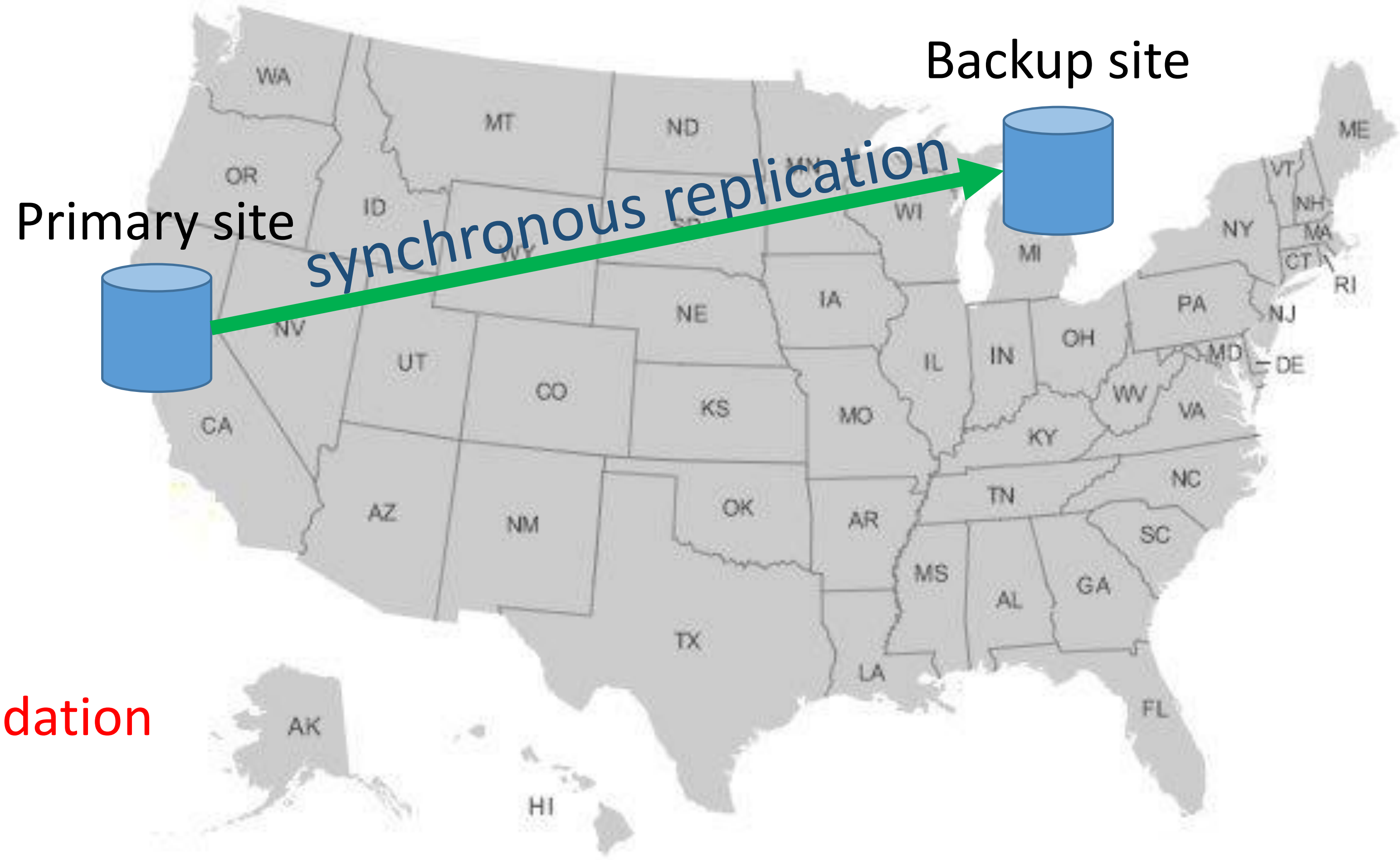
Inside the Equinix DC12 data center in Ashburn, Virginia

Distributed systems are expected to tolerate disasters

Main techniques

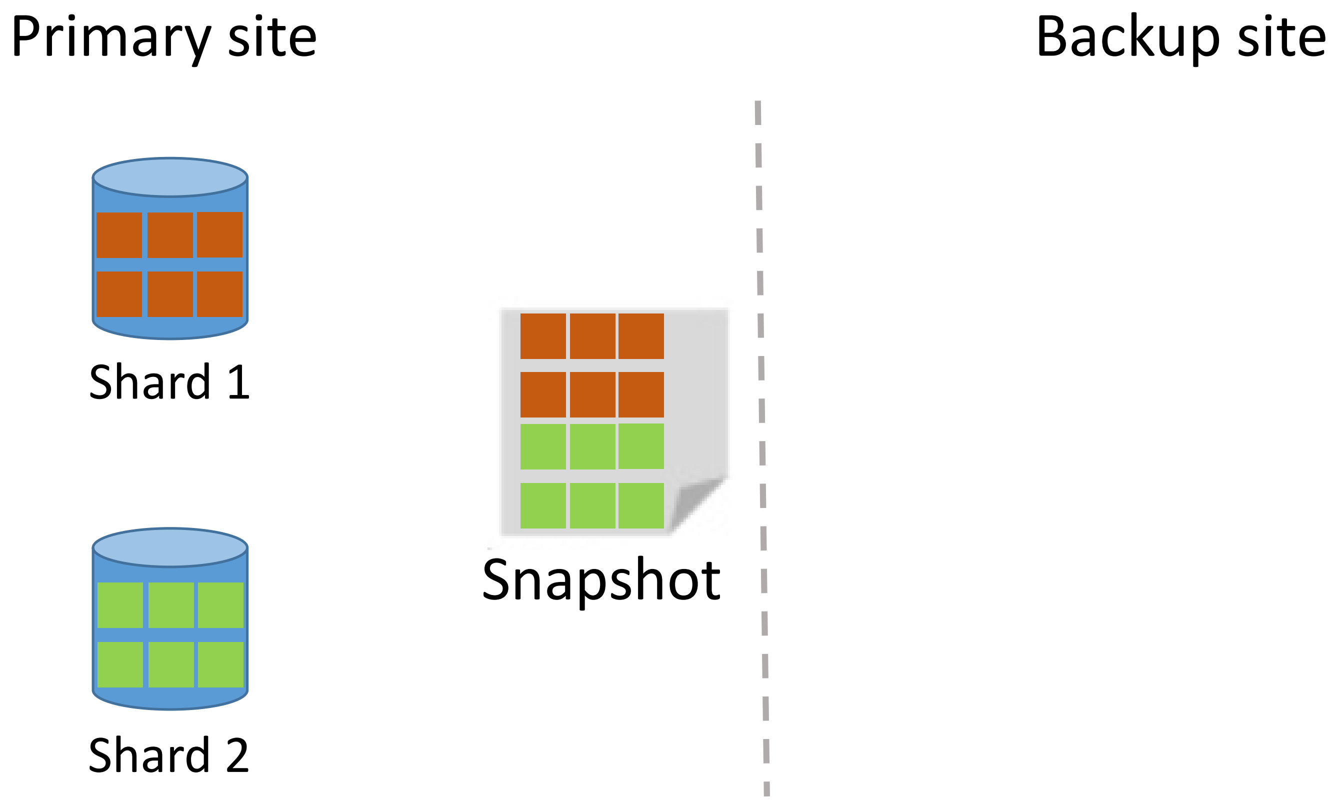
- Synchronous geo-replication
- Snapshotting

Synchronous Geo-Replication

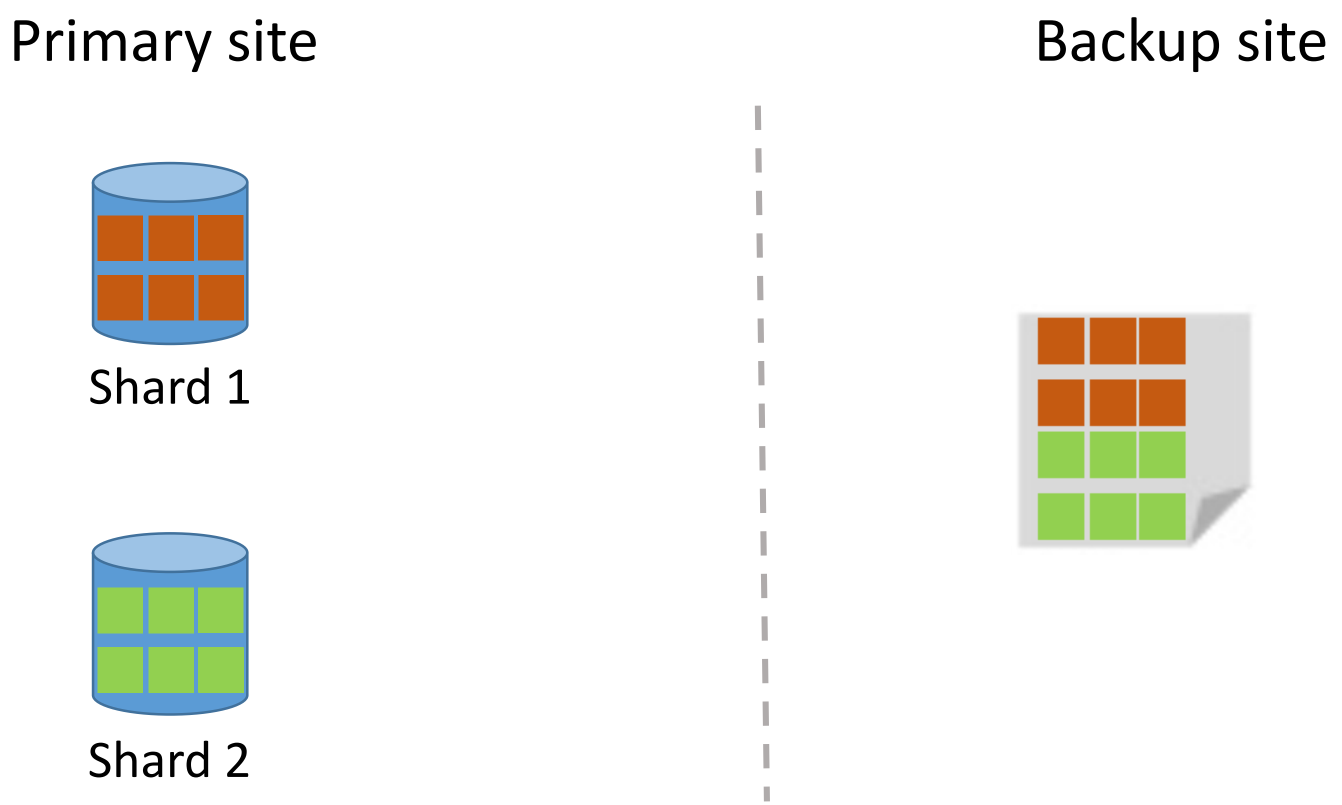


+ No data loss
– Severe performance degradation

Snapshotting



Snapshotting



\+ Higher performance than synchronous geo-replication

– Large data loss window

Can we have a high performance DR system with small data loss window?

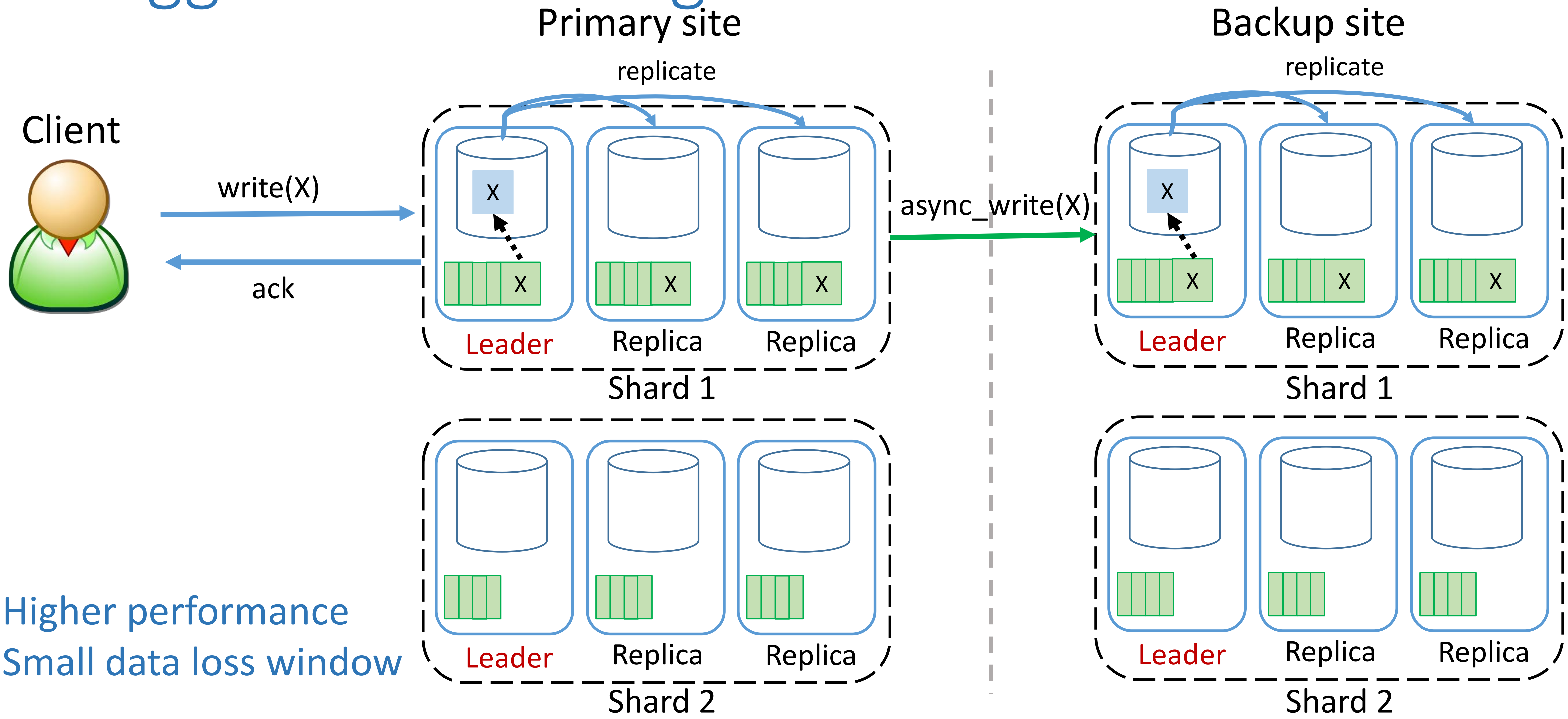
Slogger

A high performance disaster recovery approach that minimizes data loss

Main idea

- Asynchronous geo-replication → improves performance
- Leverages modern data center synchronized clocks → guarantee consistency

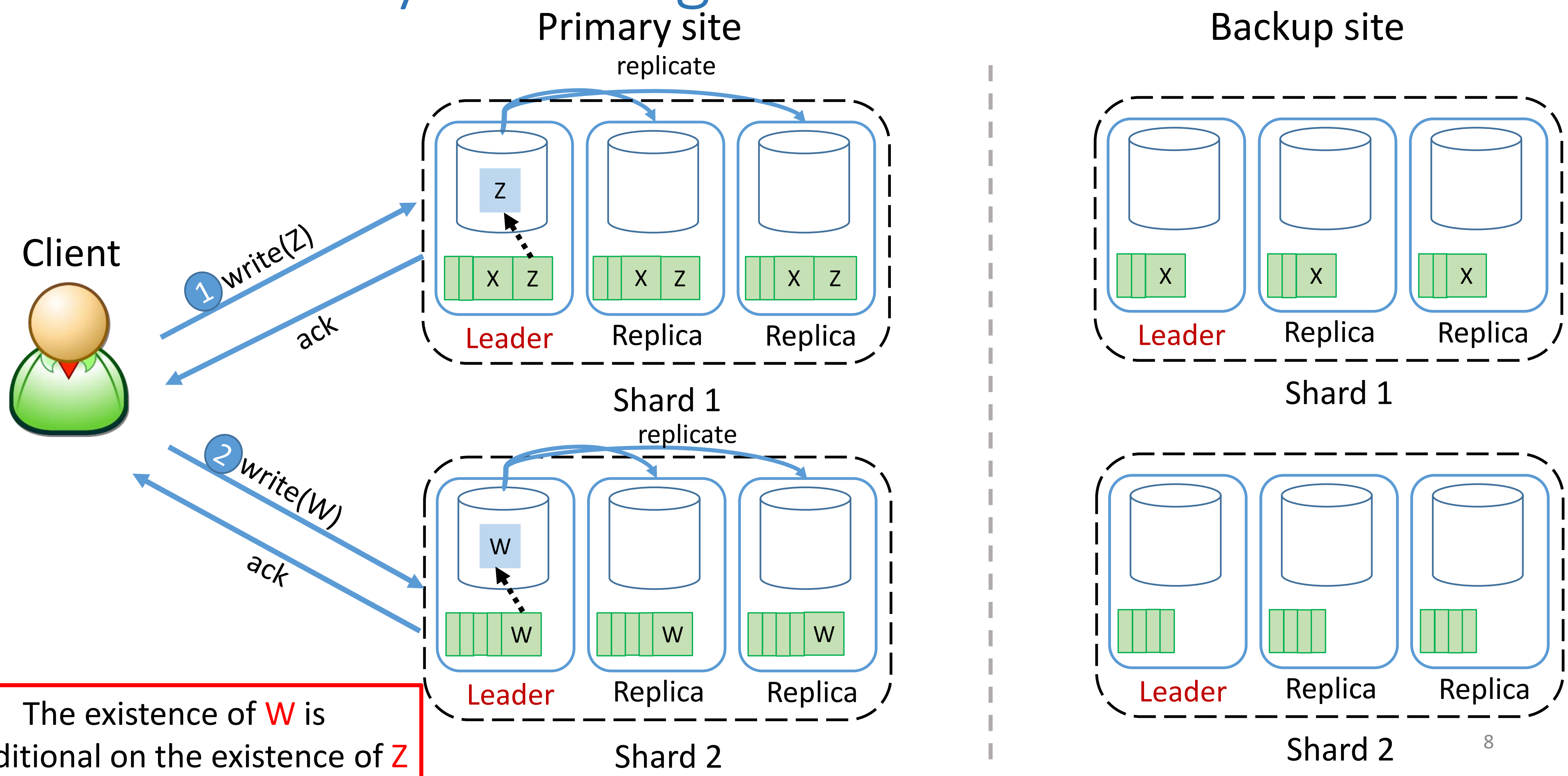
Slogger – Basic Design



+ Higher performance
+ Small data loss window

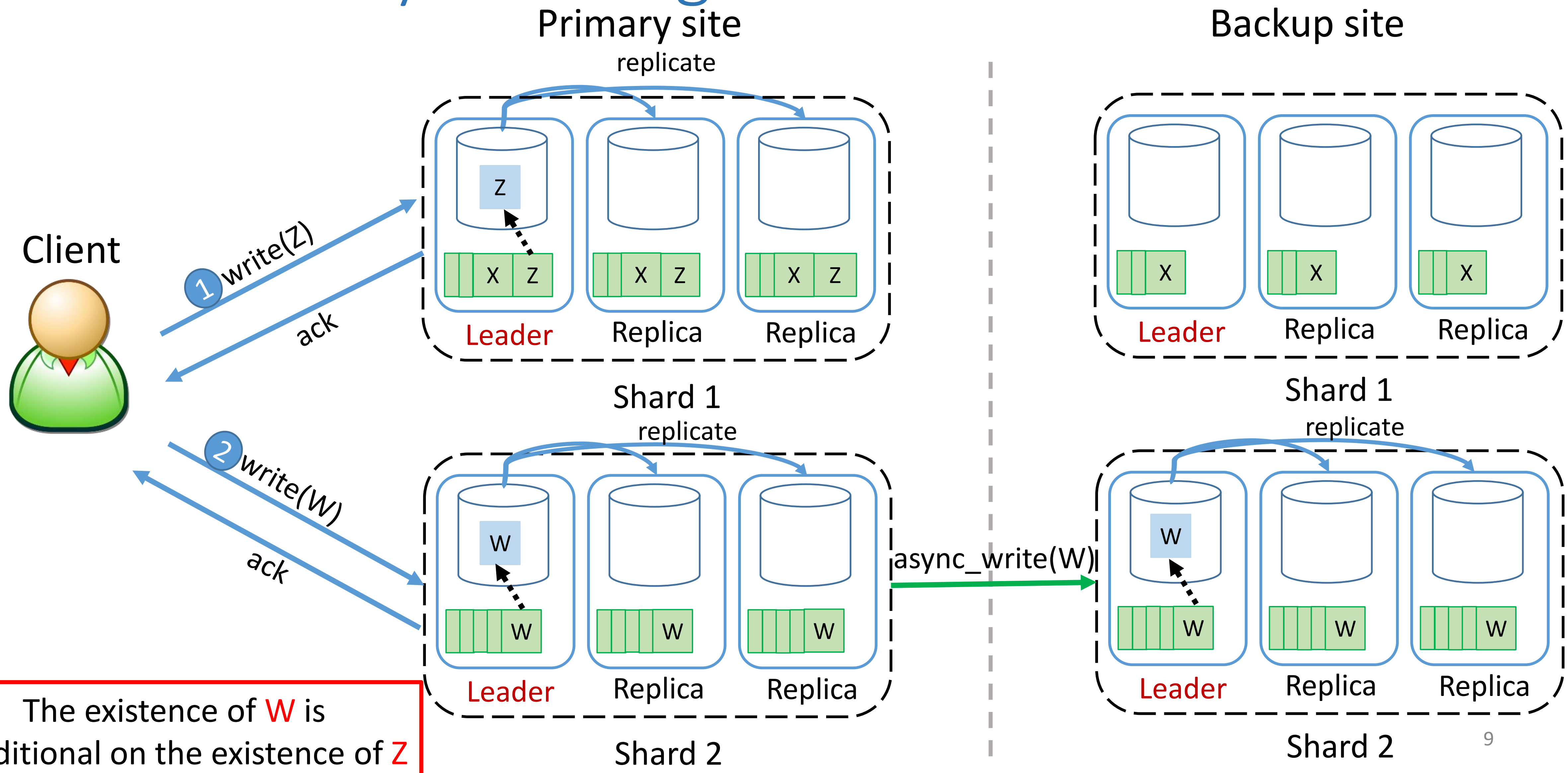
Challenge: consistency across shards?

Consistency Challenge



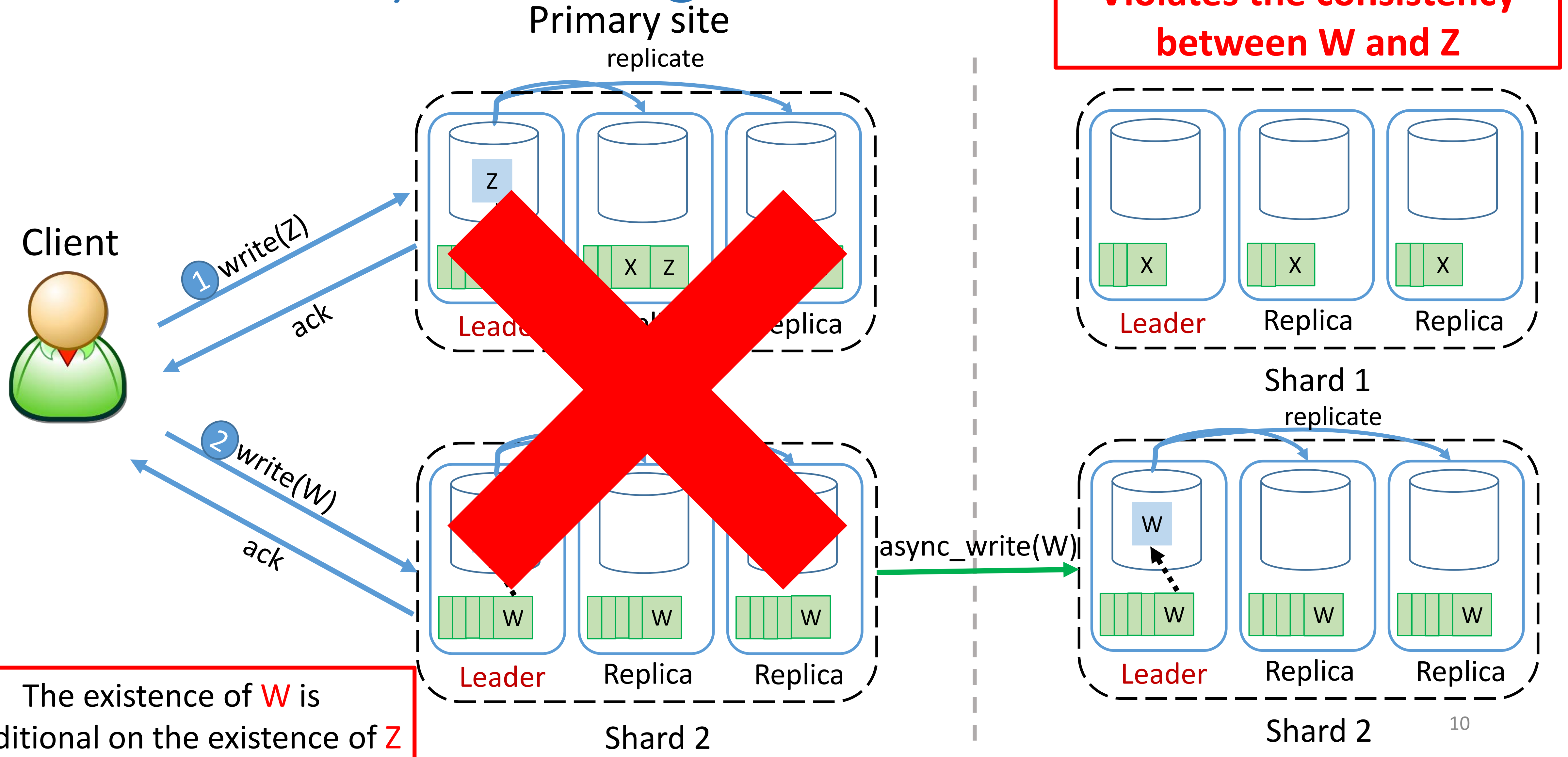
The existence of W is conditional on the existence of Z

Consistency Challenge



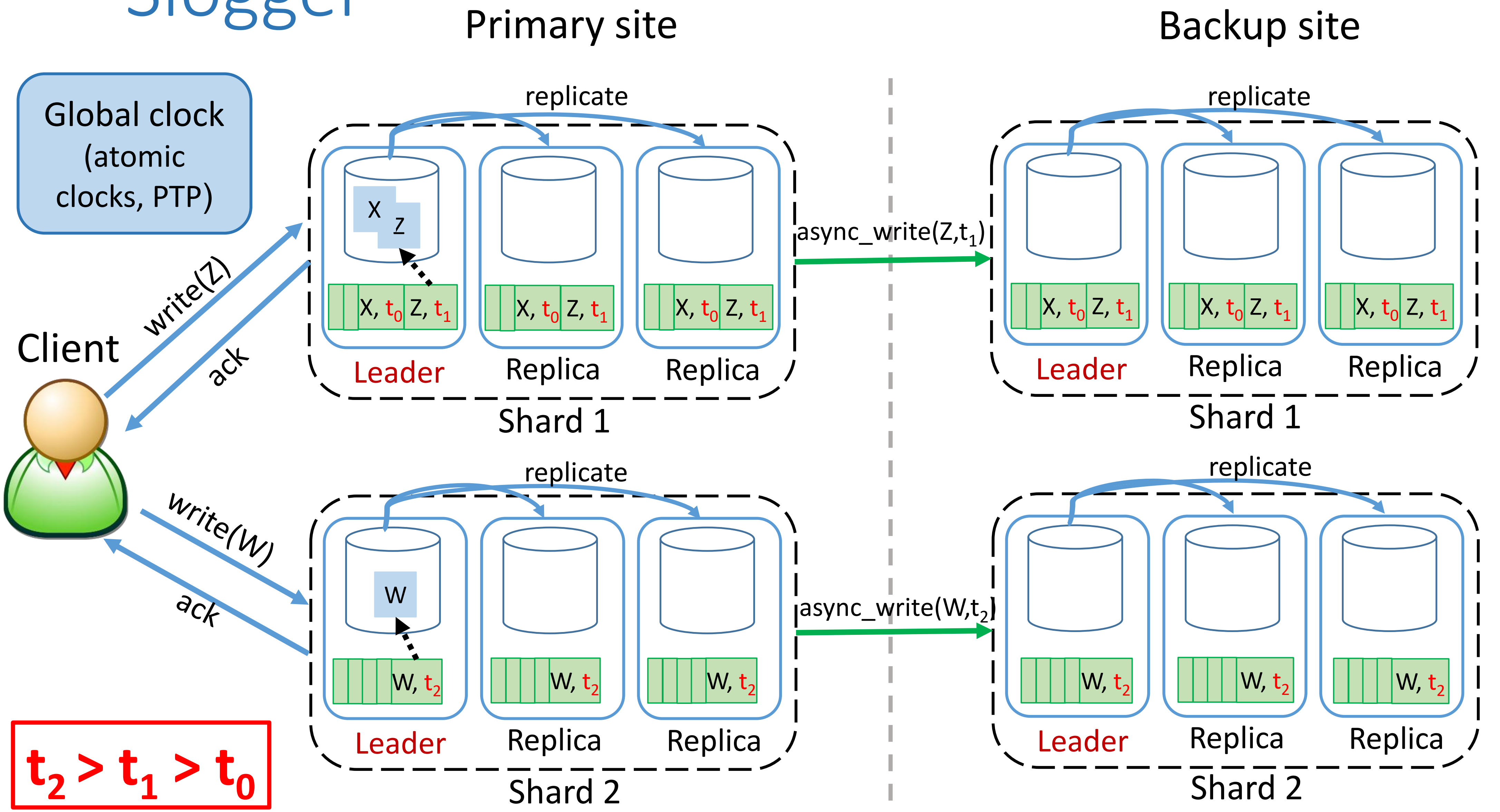
The existence of W is conditional on the existence of Z

Consistency Challenge



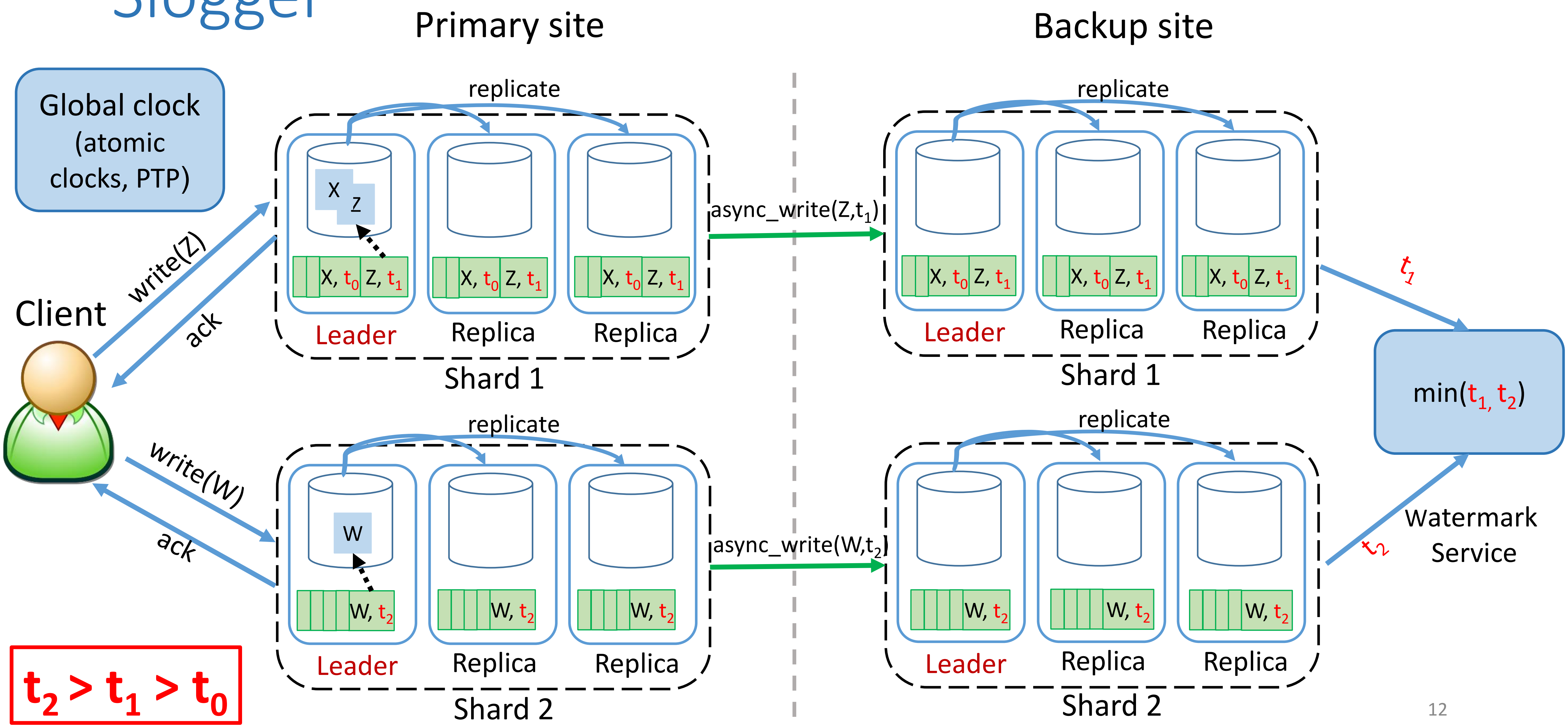
The existence of W is conditional on the existence of Z

Slogger

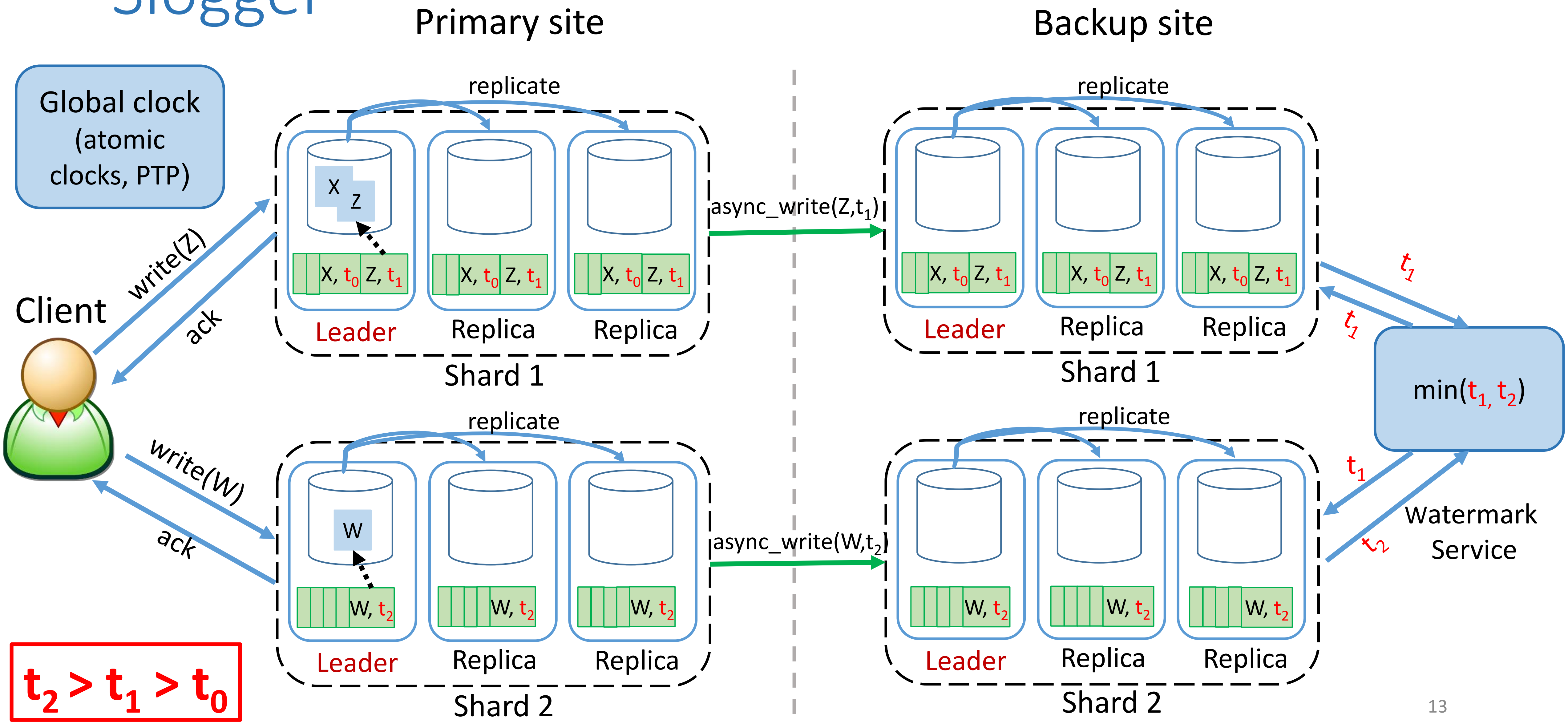


Primary site
Backup site
Global clock (atomic clocks, PTP)
Client
write(Z)
ack
write(W)
ack
replicate
X
Z
X, t0 Z, t1
Leader
Replica
Replica
Shard 1
async_write(Z,t1)
replicate
W
W, t2
Leader
Replica
Replica
Shard 2
async_write(W,t2)
Shard 1
Shard 2
t2 > t1 > t0

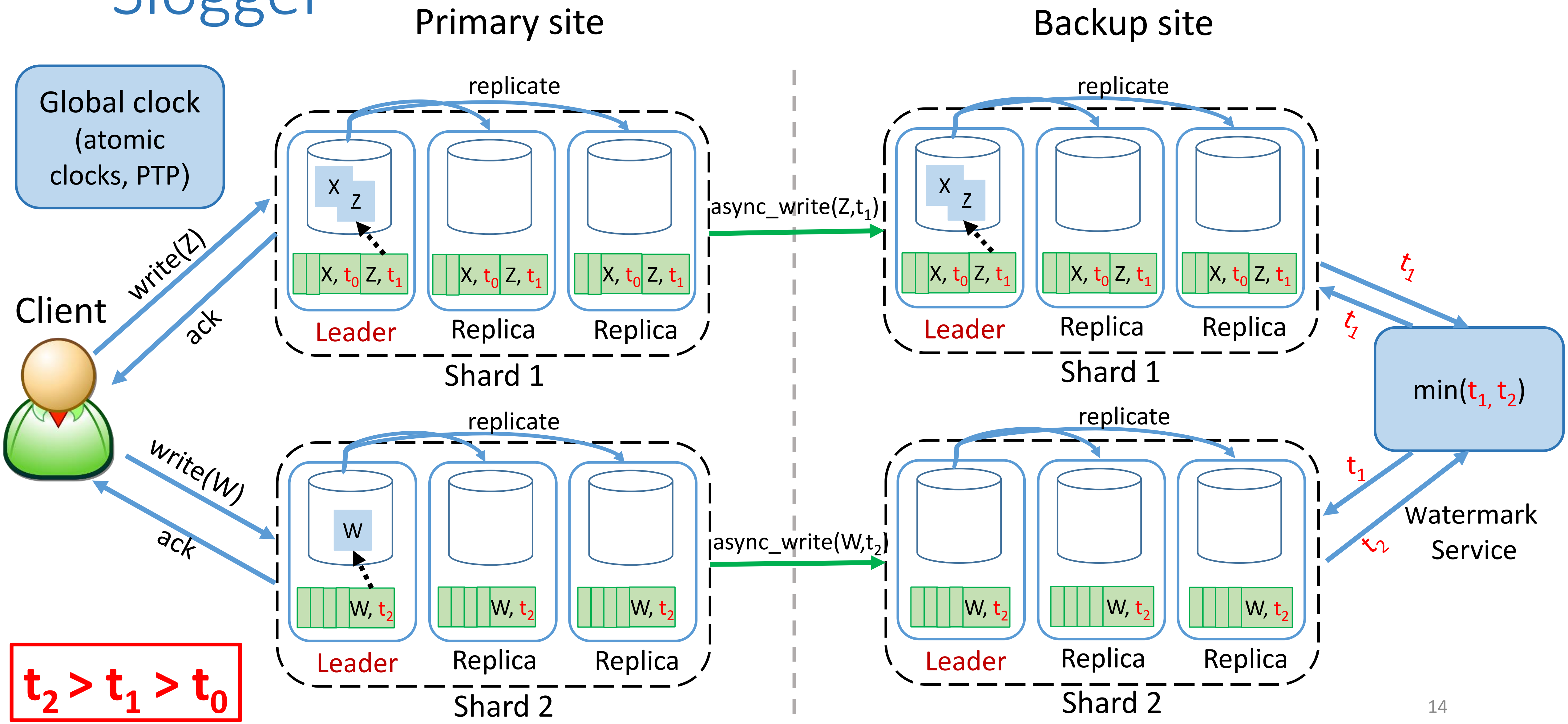
Slogger


Primary site
Backup site
Global clock (atomic clocks, PTP)
Client
write(Z)
ack
write(W)
ack
replicate
X
Z
W
X, t0 Z, t1
W, t2
Leader
Replica
Replica
Shard 1
Shard 2
async_write(Z,t1)
async_write(W,t2)
t1
t2
min(t1, t2)
Watermark Service
t2 > t1 > t0

Slogger

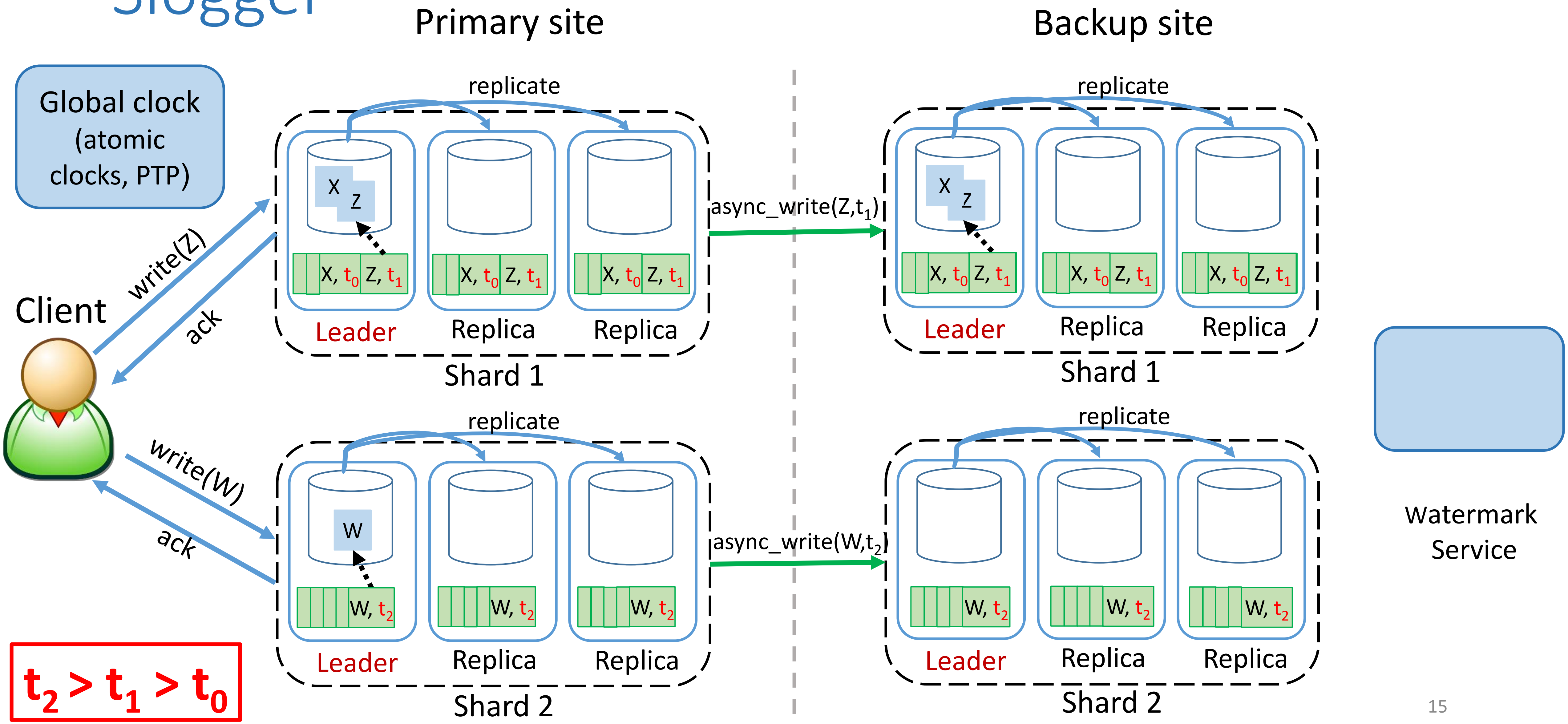

Primary site
Backup site
Global clock (atomic clocks, PTP)
Client
write(Z)
ack
write(W)
ack
replicate
X
Z
W
X, t_0 Z, t_1
W, t_2
Leader
Replica
Replica
Shard 1
Shard 2
async_write(Z,t_1)
async_write(W,t_2)
t_1
t_2
min(t_1, t_2)
Watermark Service
$t_2 > t_1 > t_0$

Slogger



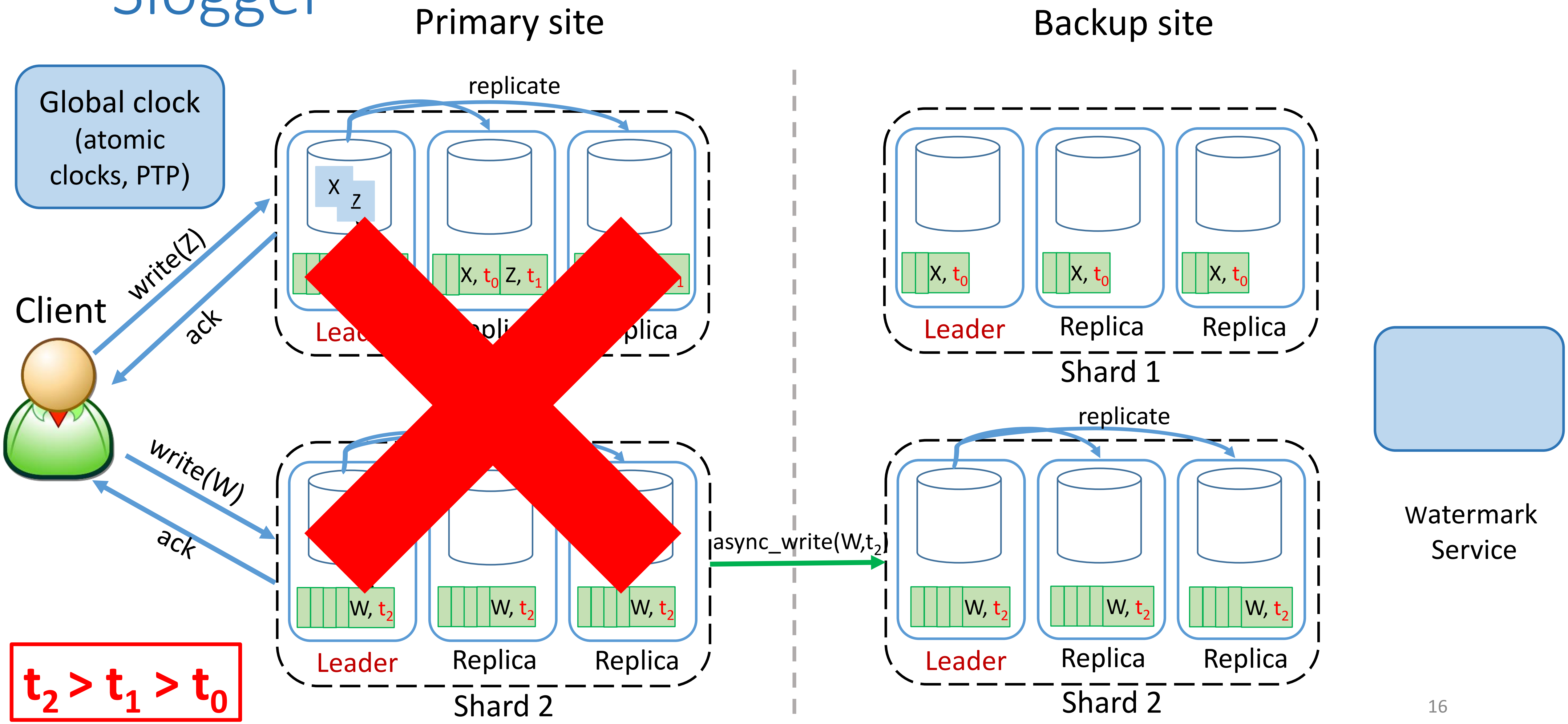
$t_2 > t_1 > t_0$

Slogger

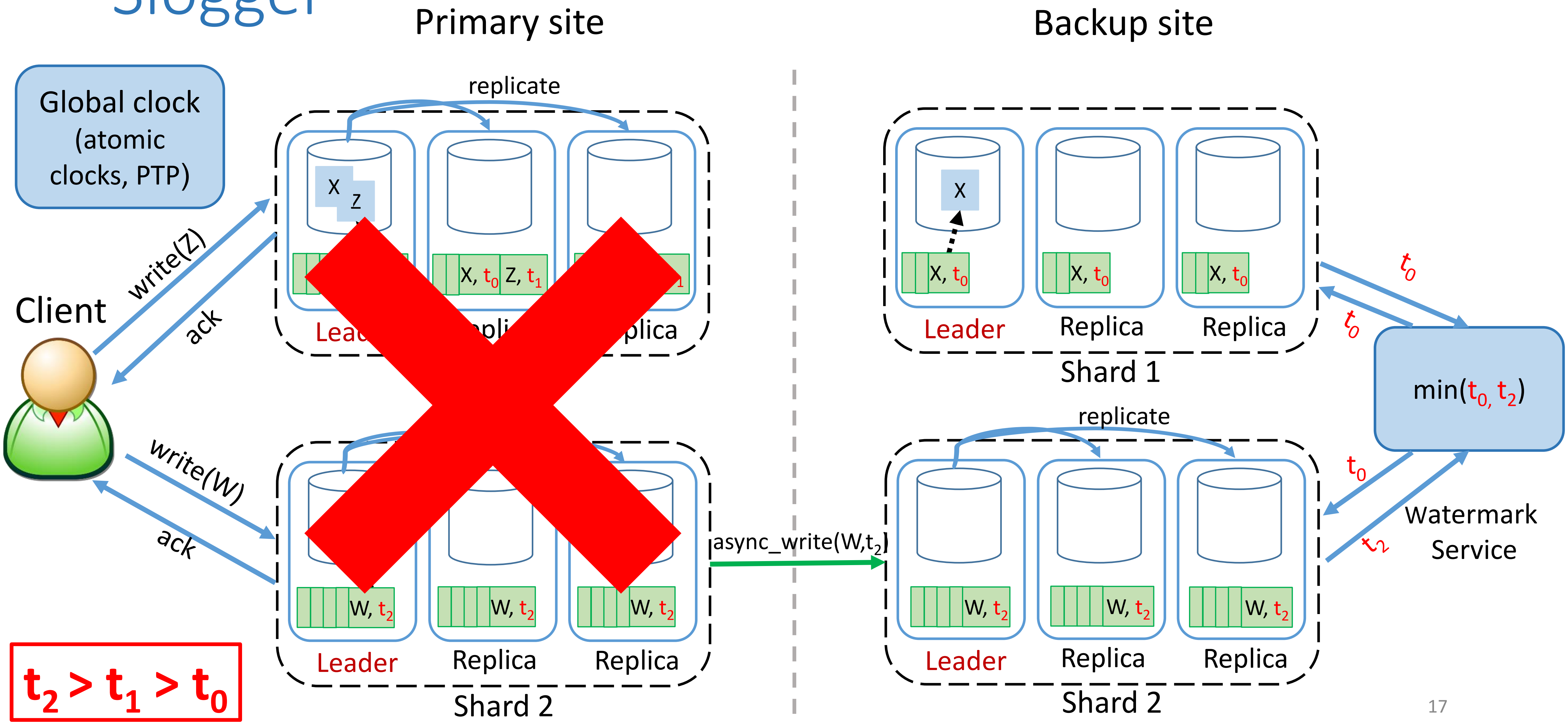


Primary site
Backup site
Global clock (atomic clocks, PTP)
Client
write(Z)
ack
write(W)
ack
replicate
X
Z
X, t0 Z, t1
Leader
Replica
Replica
Shard 1
W
W, t2
Shard 2
async_write(Z,t1)
async_write(W,t2)
Watermark Service
t2 > t1 > t0

Slogger


Primary site
Backup site
Global clock (atomic clocks, PTP)
replicate
X
Z
X, t_0
Z, t_1
Client
write(Z)
ack
write(W)
ack
W, t_2
W, t_2
W, t_2
Leader
Replica
Replica
Shard 2
async_write(W,t_2)
X, t_0
X, t_0
X, t_0
Leader
Replica
Replica
Shard 1
replicate
W, t_2
W, t_2
W, t_2
Leader
Replica
Replica
Shard 2
Watermark Service
$t_2 > t_1 > t_0$

Slogger



Primary site
Backup site
Global clock (atomic clocks, PTP)
Client
write(Z)
ack
write(W)
ack
replicate
X
Z
X, t_0
Z, t_1
W, t_2
Leader
Replica
Shard 1
Shard 2
async_write(W,t_2)
t_0
t_2
min(t_0, t_2)
Watermark Service
$t_2 > t_1 > t_0$

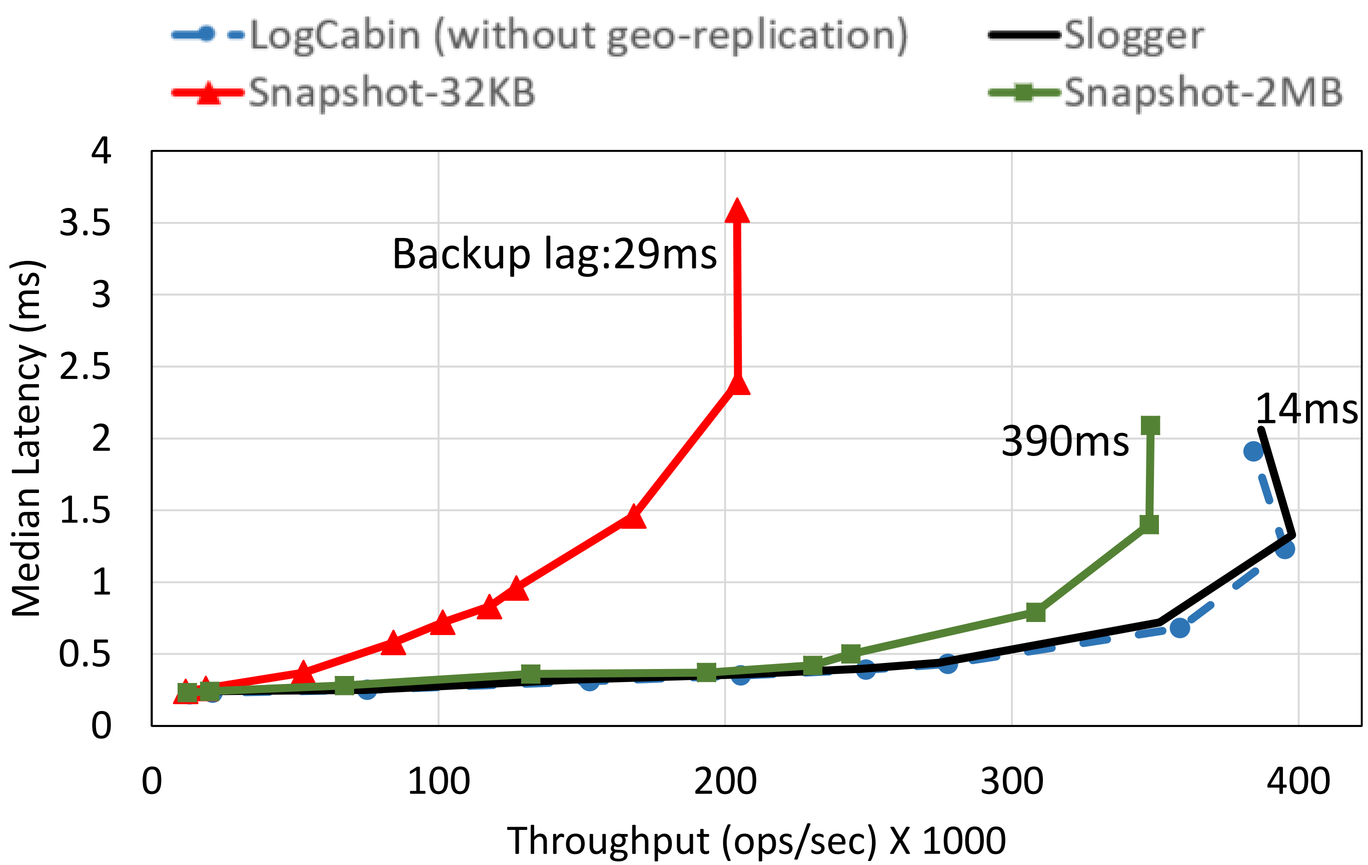
Evaluation

Alternatives
- Slogger on top of LogCabin
- Synchronous geo-replication
- Incremental snapshotting

Metrics
- Performance
- Backup site lag
- Fault tolerance
- Watermark service scalability

Testbed
- Two CloudLab data centers (Clemson & Wisconsin)
- 16 machines for each site
  - Dual Socket CPU – 10 cores/socket
  - Local network: 10Gbps
  - WAN: 1Gbps
  - RTT: 26 milliseconds

Performance Comparison



Slogger achieves optimal performance with a small data loss window

Conclusion

Slogger

- Exploits synchronized clocks within a data center
- Preserves consistency
- Achieves optimal performance with milliseconds data loss window

Thank you!