
facebook

Wormhole:
Reliable Pub-Sub to support Geo-replicated Internet Services

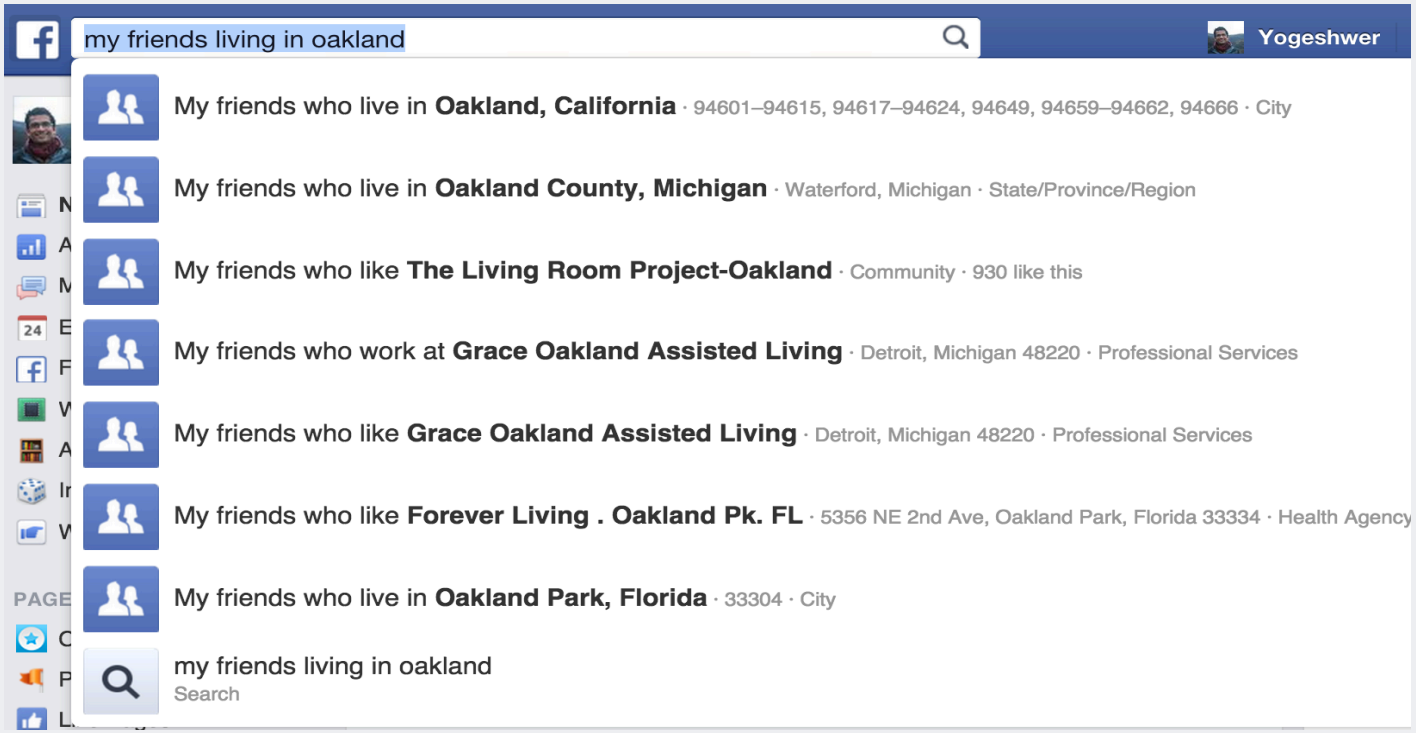
Yogi Sharma

Facebook

Joint work with Philippe Ajoux, Petchean Ang, David Callies, Abhishek Choudhary, Laurent Demailly, Thomas Fersch, Liat Atsmon Guz, Andrzej Kotulski, Sachin Kulkarni, Sanjeev Kumar, Harry Li, Jun Li, Evgeniy Makeev, Kowshik Prakasam, Robbert van Renesse (Cornell), Sabyasachi Roy, Pratyush Seth, Yee Jiun Song, Kaushik Veeraraghavan, Benjamin Wester, Peter Xie.

Challenge: Update Stale Data

Graph Search



Alice moves away from Oakland

Alice: Change
my city to Boston

MySQL

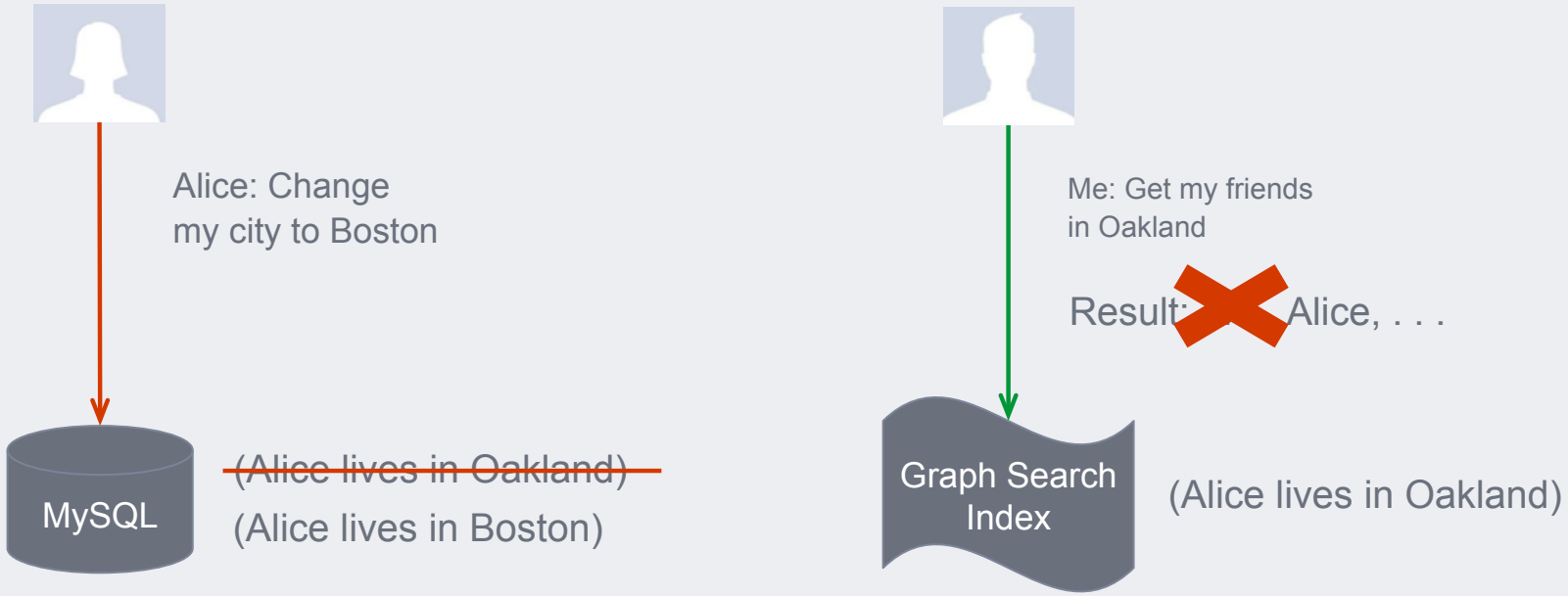
~~(Alice lives in Oakland)~~

(Alice lives in Boston)

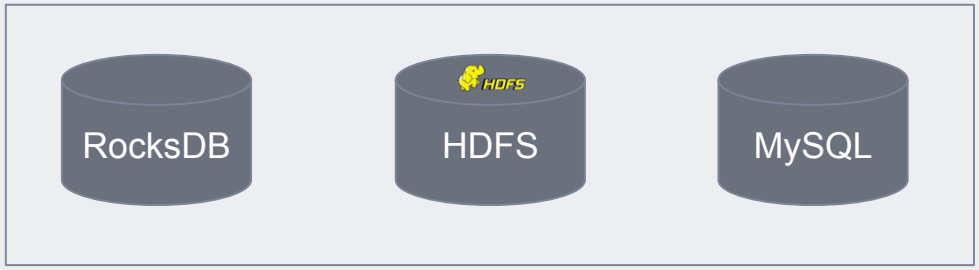
Graph Search Index

(Alice lives in Oakland)

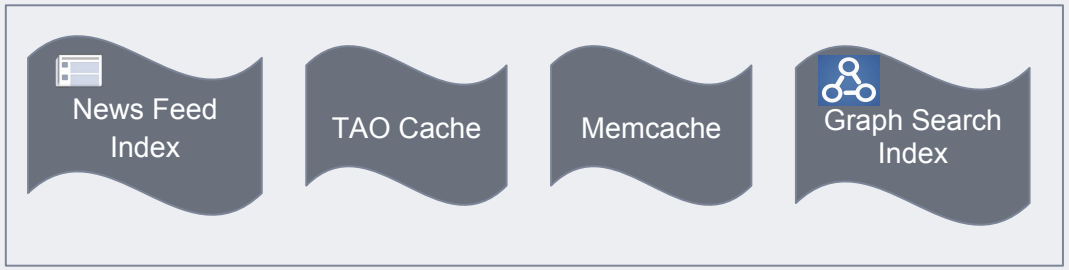
Alice moves away from Oakland



Need for updates and its challenges



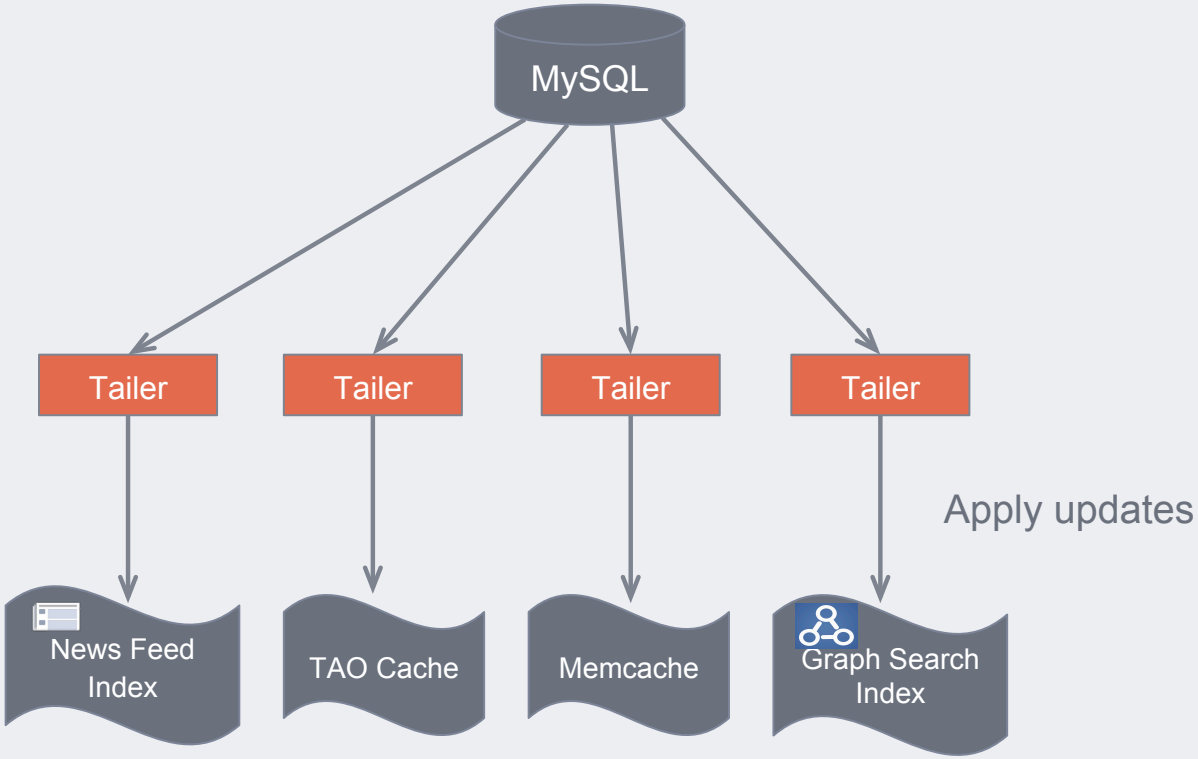
Datastores



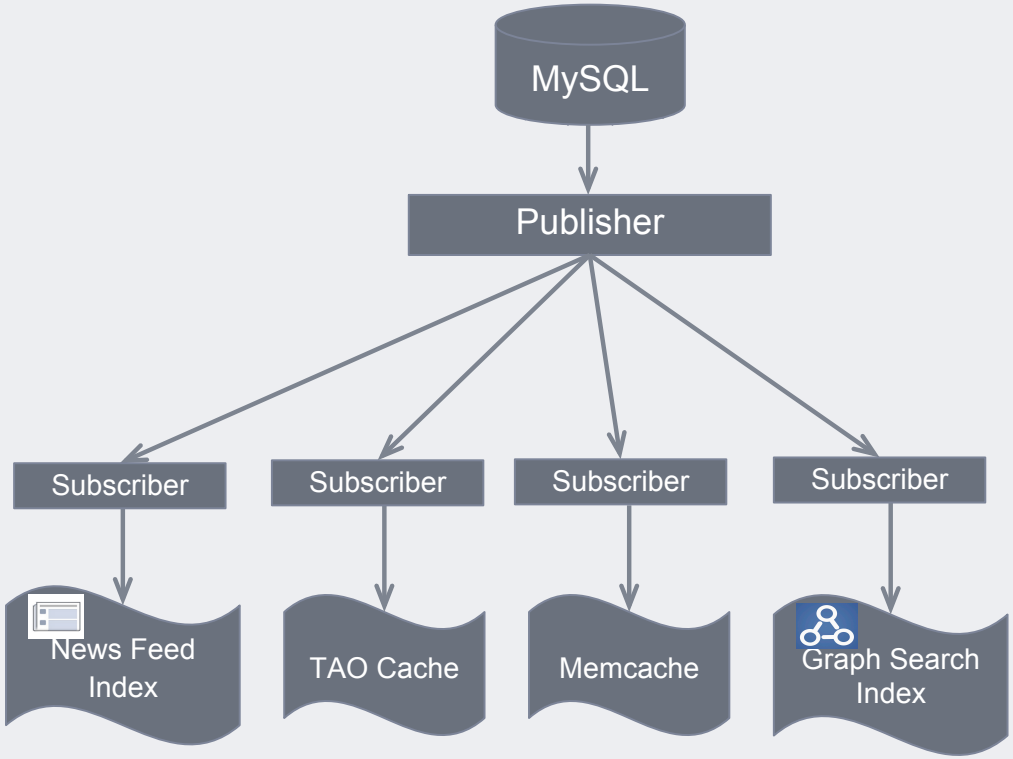
Applications

- Tens of applications
- Heterogeneous datastores
- Reliable delivery
- Varying application speeds

Each application tails updates



The publisher pushes updates



- ☑ Tens of applications
- ☐ Heterogeneous datastores
- ☐ Reliable delivery
- ☐ Varying application speeds

Wormhole – a pub-sub system

What it is:

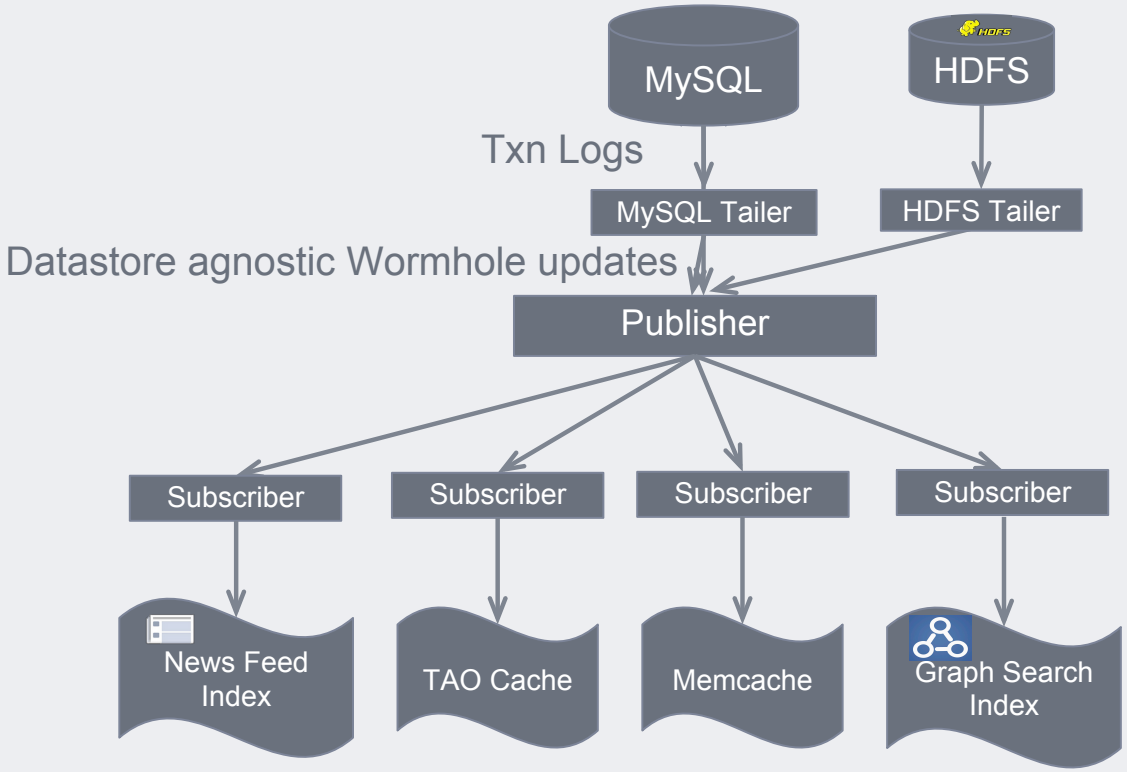
- Runs on existing heterogeneous datastores
- Delivers updates reliably – at least once, in-order
- Handles varying application speeds efficiently

Transporting over 5 trillion updates per day in Facebook

What it isn't:

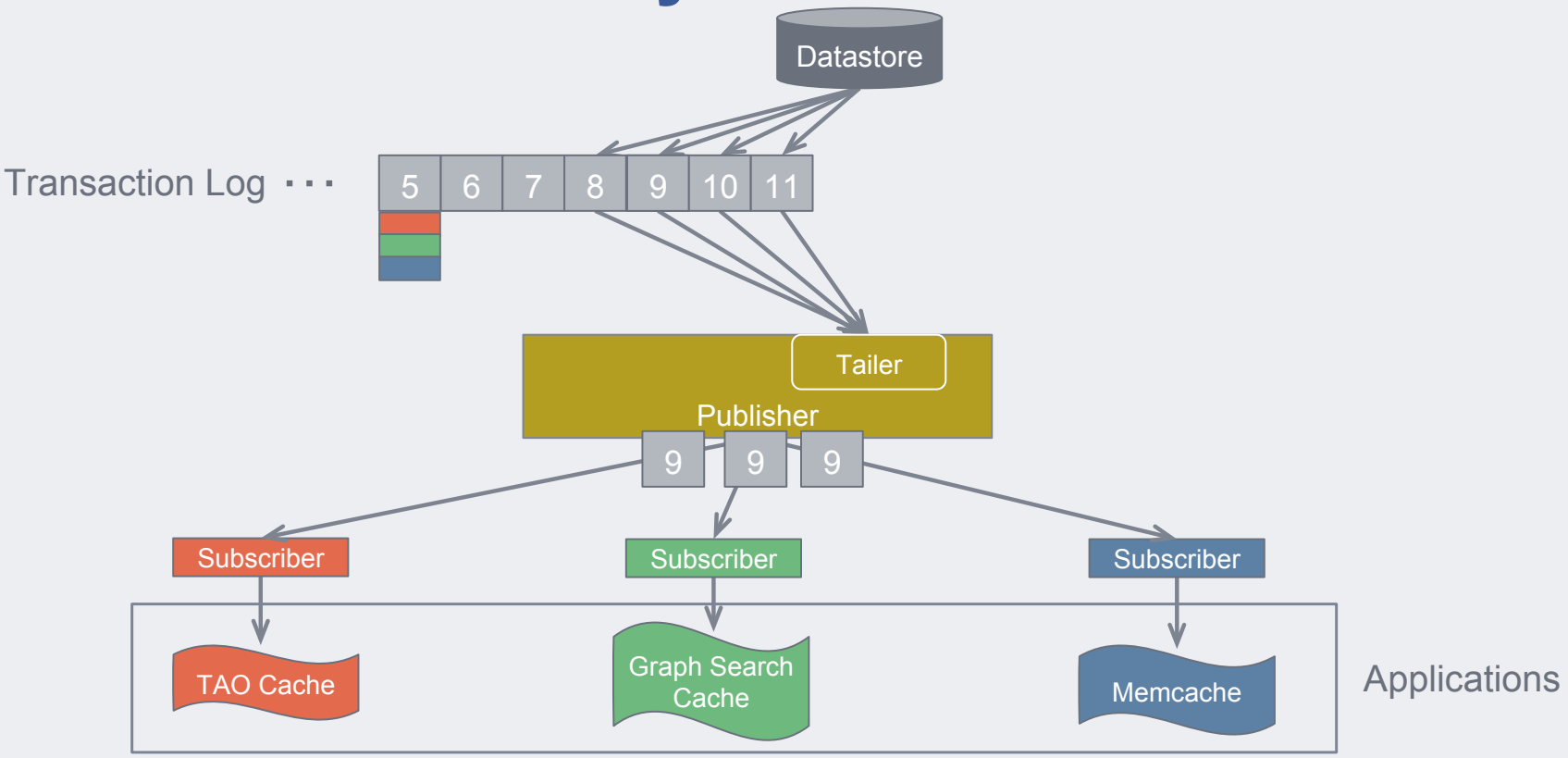
- Not exactly-once delivery
- Not a storage system
- No global ordering across different datastores

Support heterogeneous datastores

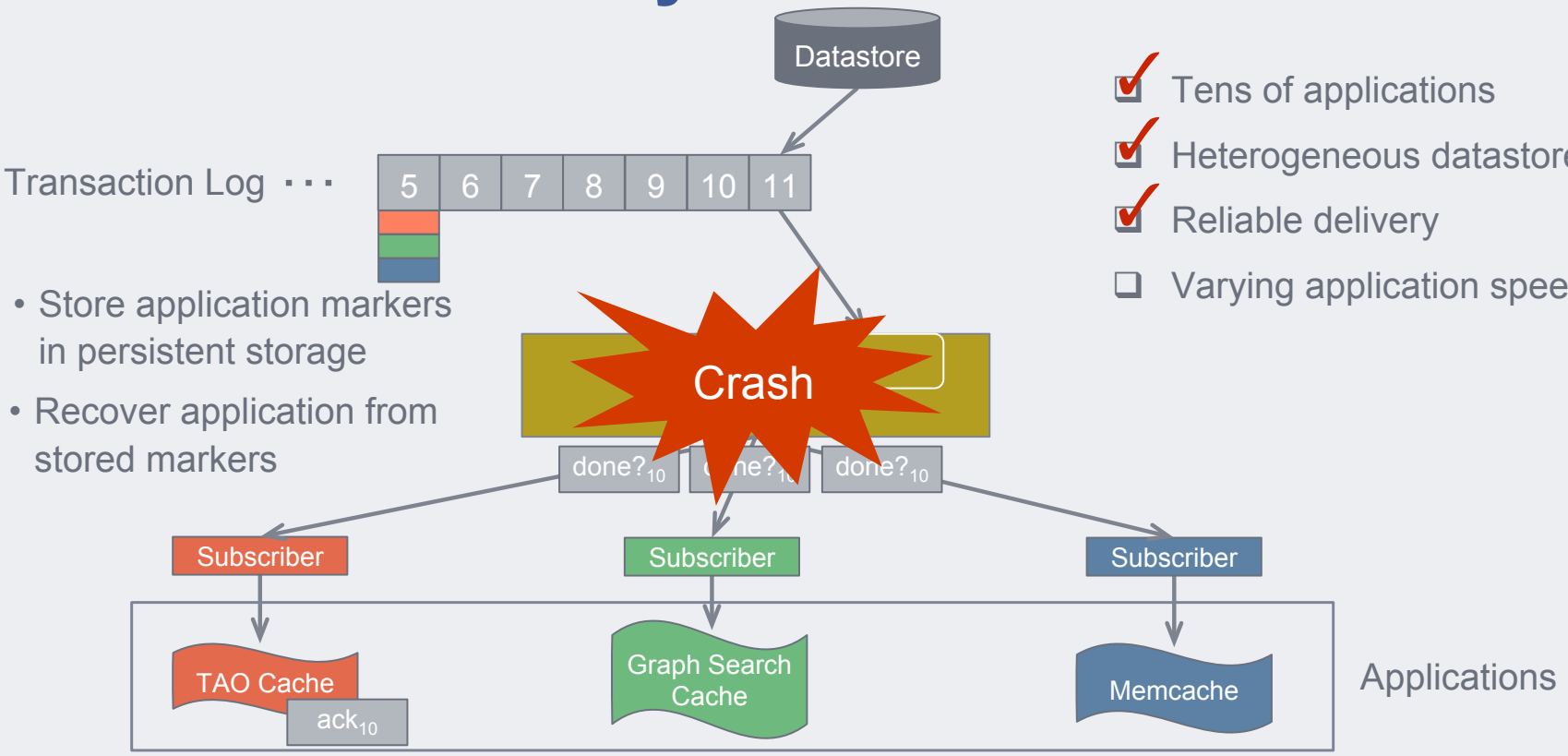


- ☑ Tens of applications
- ☑ Heterogeneous datastores
- ☐ Reliable delivery
- ☐ Varying application speeds

Reliable delivery



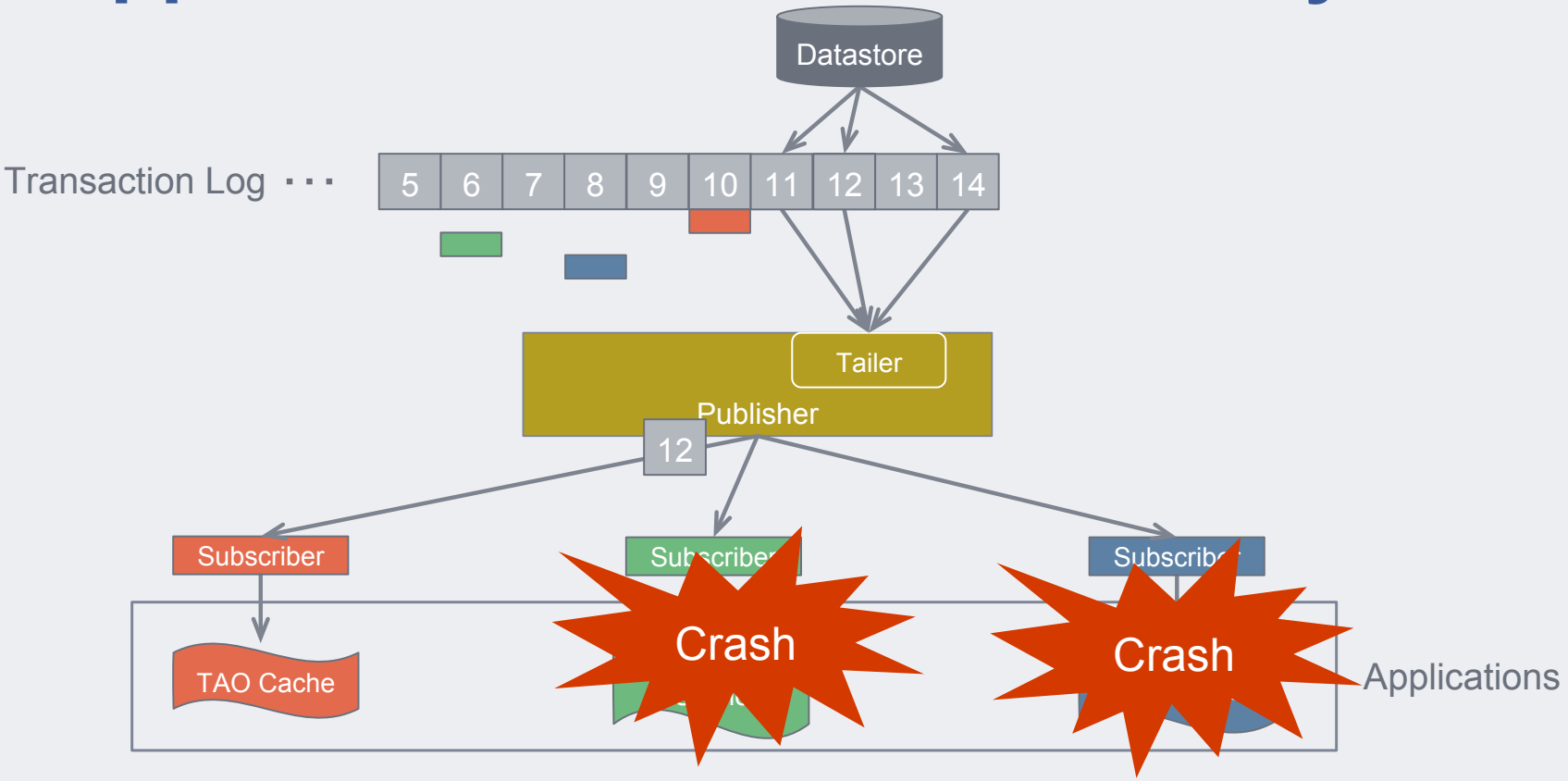
Reliable delivery



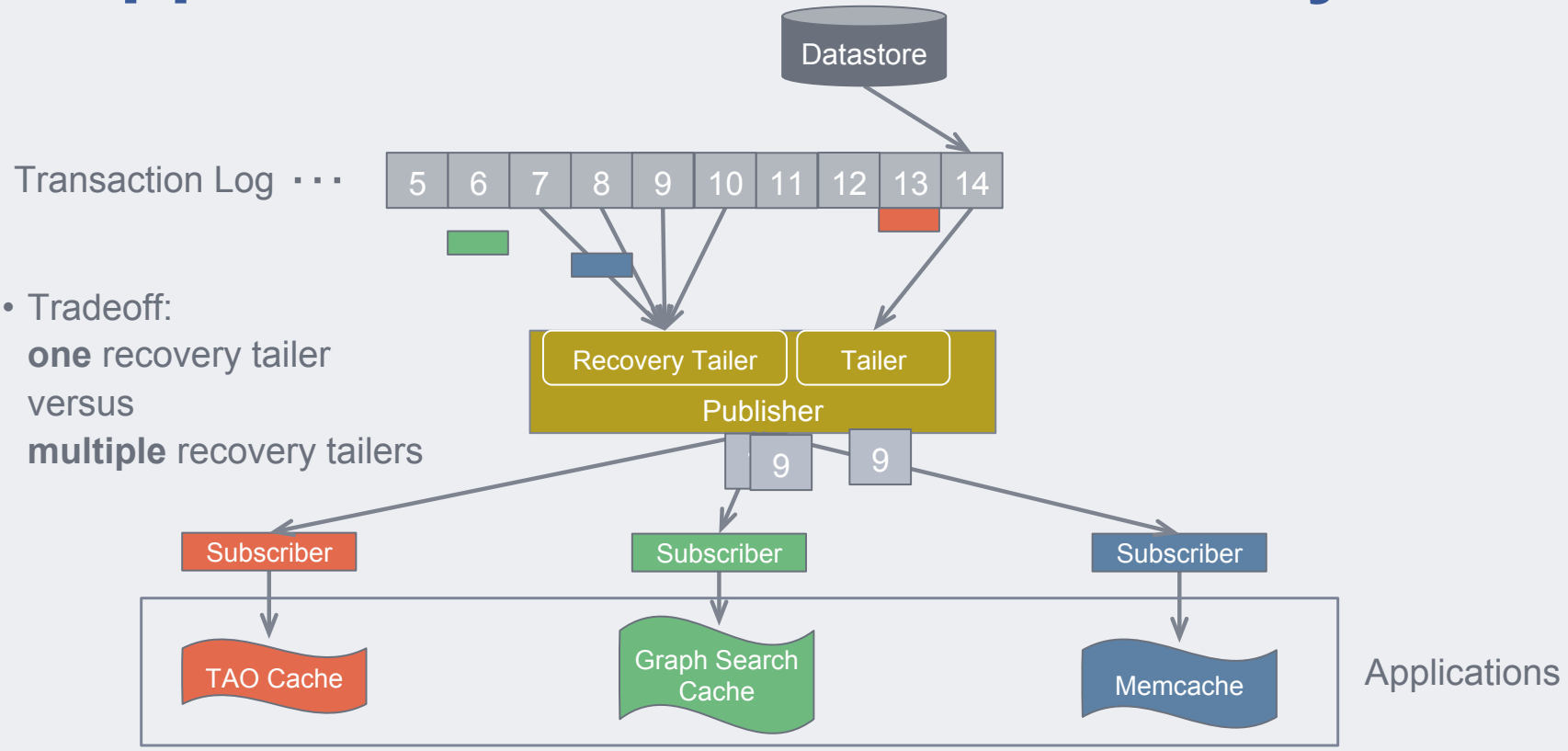
- Store application markers in persistent storage
- Recover application from stored markers

- ☑ Tens of applications
- ☑ Heterogeneous datastores
- ☑ Reliable delivery
- ☐ Varying application speeds

Applications failure and recovery

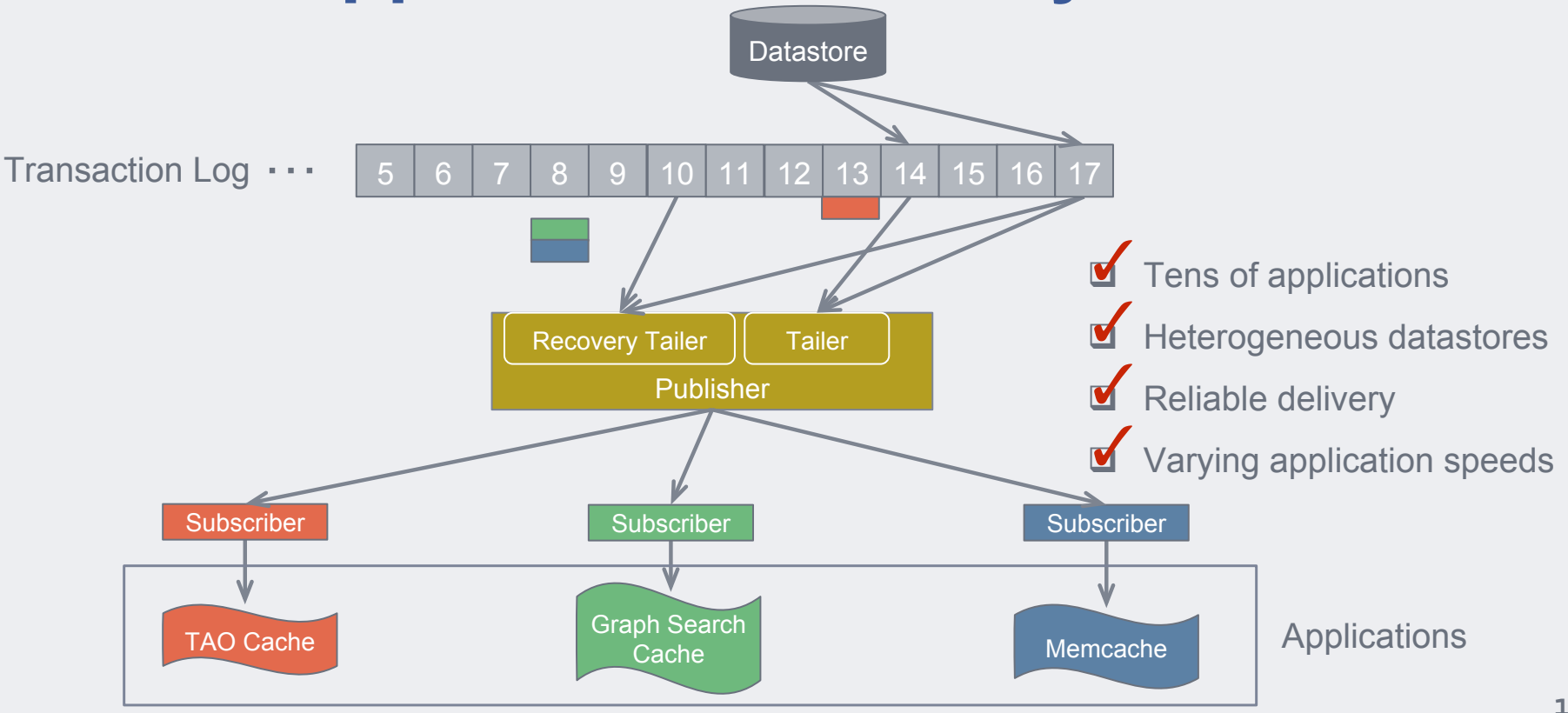


Applications failure and recovery



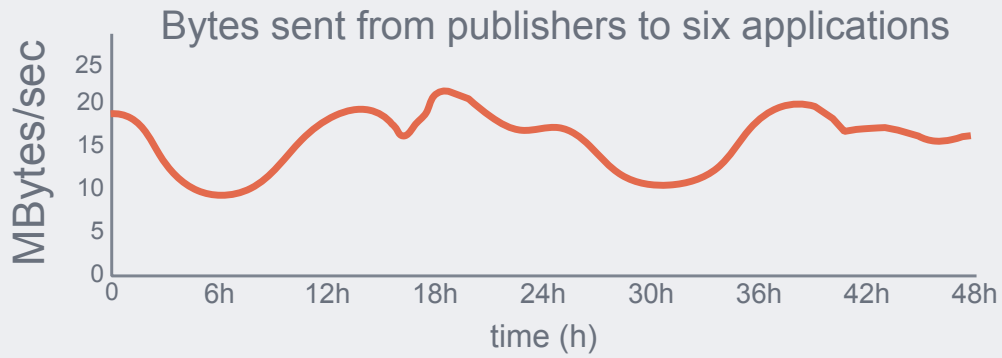
- Tradeoff:
  one recovery tailer
  versus
  multiple recovery tailers

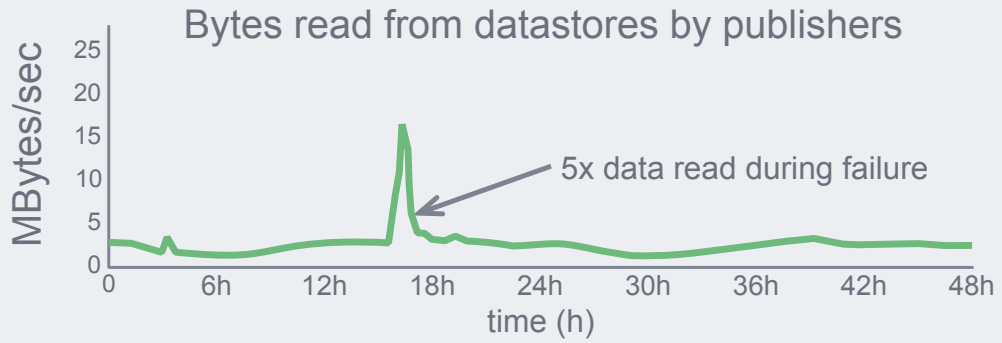
Finish applications recovery



- Tens of applications
- Heterogeneous datastores
- Reliable delivery
- Varying application speeds

Tailers: I/O efficiency

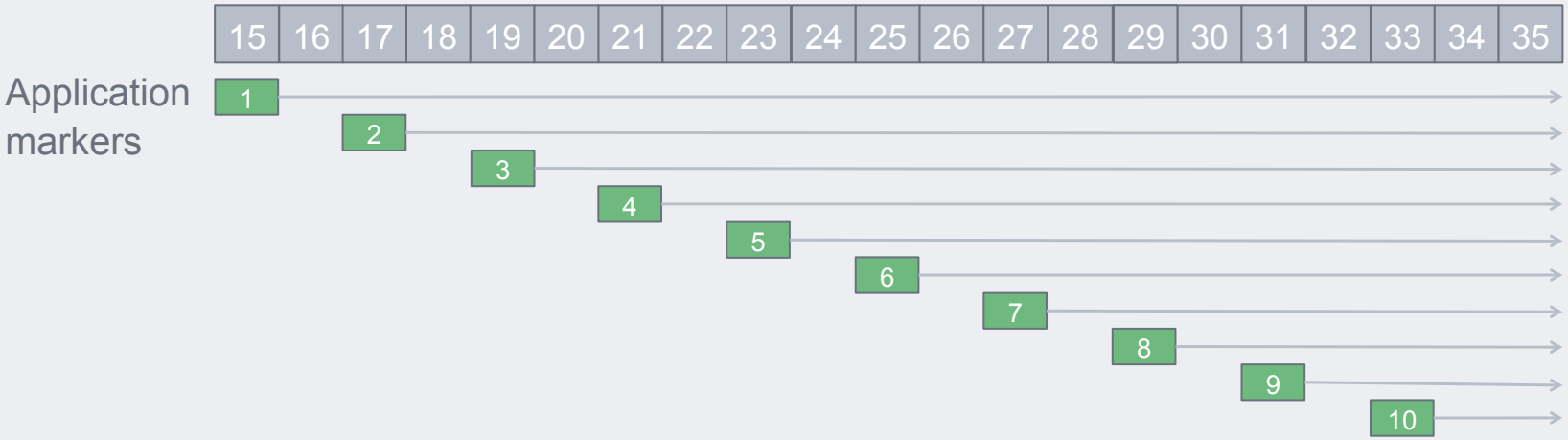




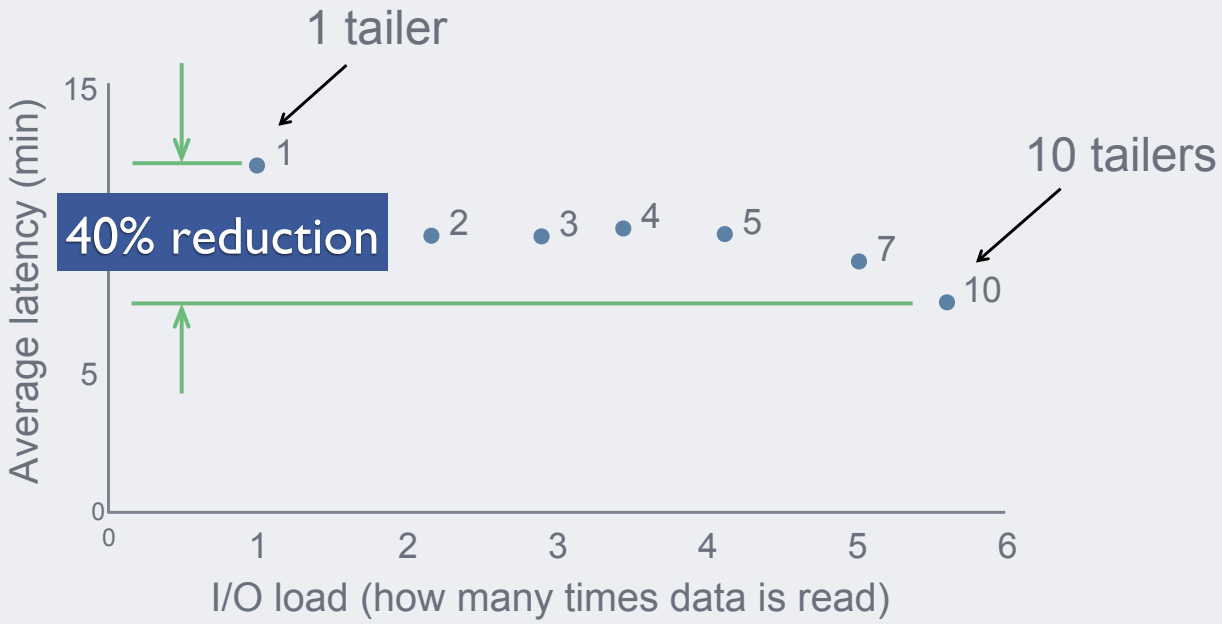
- Production deployment
- Many publishers and datastores
- Replication, 6 applications
- Metrics every 1 minute

Tailers: I/O vs. latency tradeoff

Experiment: Send part of a 20 GB data to 10 applications

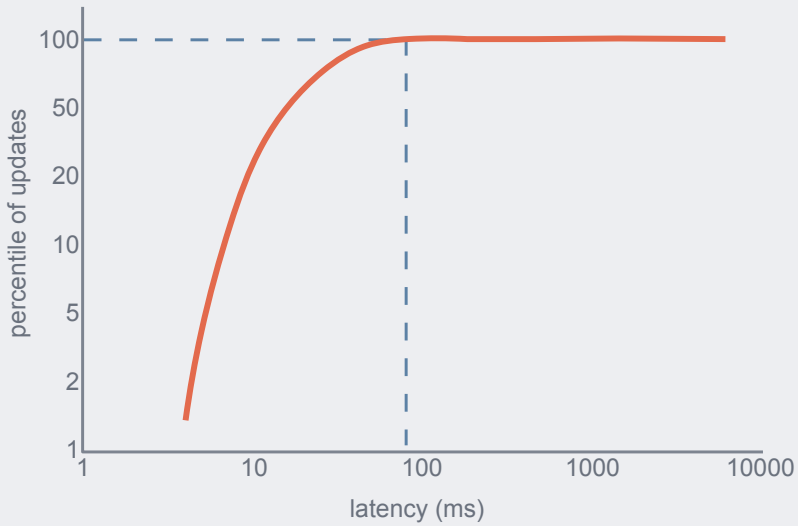


Tailers: I/O vs. latency tradeoff



Latency of updates processing

- One production publisher
- Sample of 50k updates
- Measure latency between “write to datastore” and “delivery to application”



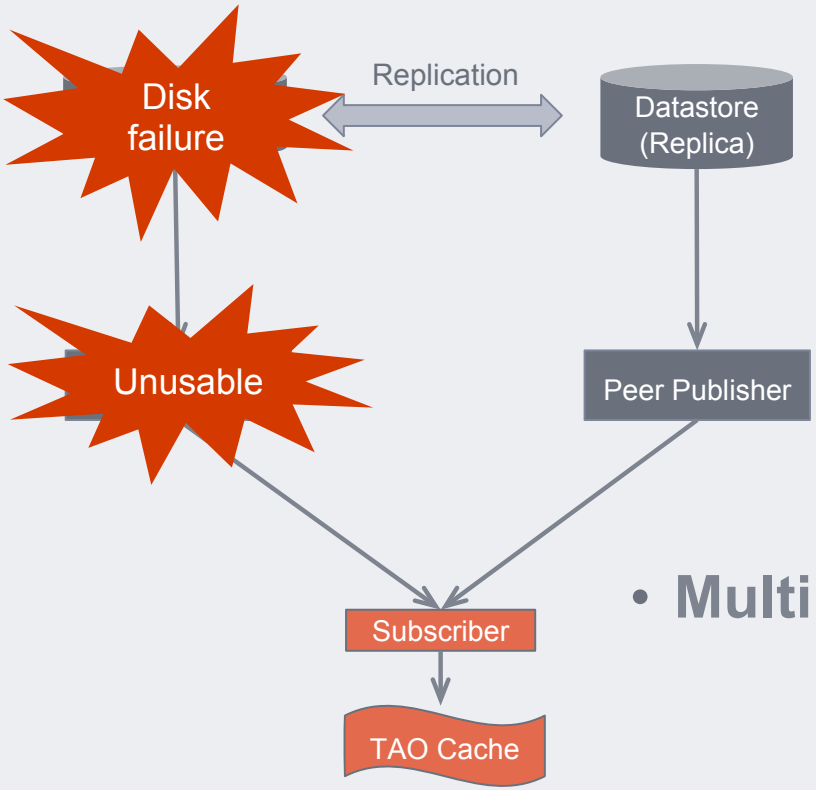
99-percentile latency ~ 81ms

What's next?

- ☑ Tens of applications
- ☑ Heterogeneous datastores
- ☑ Reliable delivery
- ☑ Varying application speeds

What if datastore disk fails?

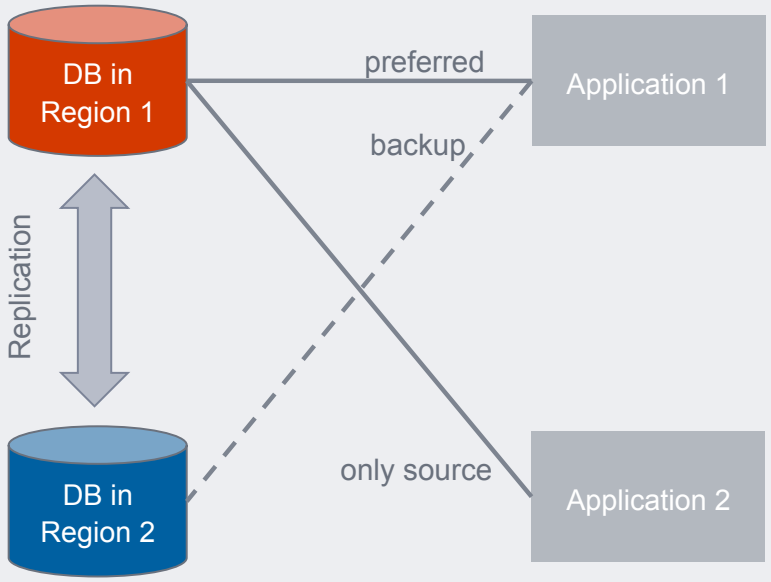
Reliable delivery despite datastore failure

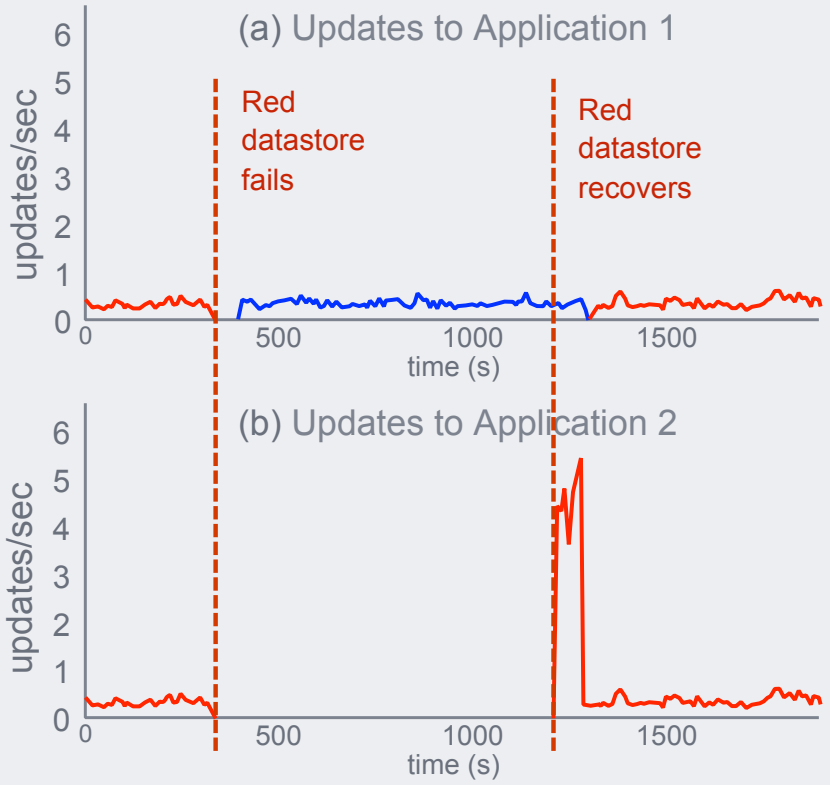


- Global application markers
- Datastore agnostic position
- Coordination mechanism

- **Multi Copy Reliable Delivery**

Multi copy reliable delivery





Conclusions

- Wormhole scalable pub-sub in production at Facebook
- Works with existing heterogeneous datastores
- Provides at-least once, in-order delivery despite failures
- Trades off latency and I/O using tailers

Thank You


facebook