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

MySQL

Graph Search Index

(Alice lives in Oakland)

(Alice lives in Oakland)

(Alice lives in Oakland)

(Alice lives in Boston)

Alice: Change my city to Boston
Alice moves away from Oakland

MySQLe Alice: Change my city to Boston

Graph Search Index (Alice lives in Oakland)

Me: Get my friends in Oakland

Result: Alice, . . .

(Alice lives in Oakland)
Need for updates and its challenges

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

Datastores:
- RocksDB
- HDFS
- MySQL

Applications:
- News Feed Index
- TAO Cache
- Memcache
- Graph Search Index
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

Datastore

Transaction Log: 5 6 7 8 9 10 11

Subscriber

TAO Cache

Graph Search Cache

Memcache

Applications

Crash

done?_{10}
done?_{10}
done?_{10}
Applications failure and recovery
Applications failure and recovery

Transaction Log • • •

- Tradeoff: one recovery tailer versus multiple recovery tailers
Finish applications recovery

Datastore

- Recovery Tailer
- Tailer
- Publisher

Transaction Log ... 5 6 7 8 9 10 11 12 13 14 15 16 17

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

Subscriber

- TAO Cache
- Graph Search Cache
- Memcache

Applications
Tailers: I/O efficiency

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

Bytes sent from publishers to six applications

Bytes read from datastores by publishers

5x data read during failure
Tailers: I/O vs. latency tradeoff

Experiment: Send part of a 20 GB data to 10 applications
Tailers: I/O vs. latency tradeoff

Average latency (min)

I/O load (how many times data is read)

1 tailer

40% reduction

10 tailers
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

(a) Updates to Application 1
(b) Updates to Application 2

Red datastore fails
Red datastore recovers
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