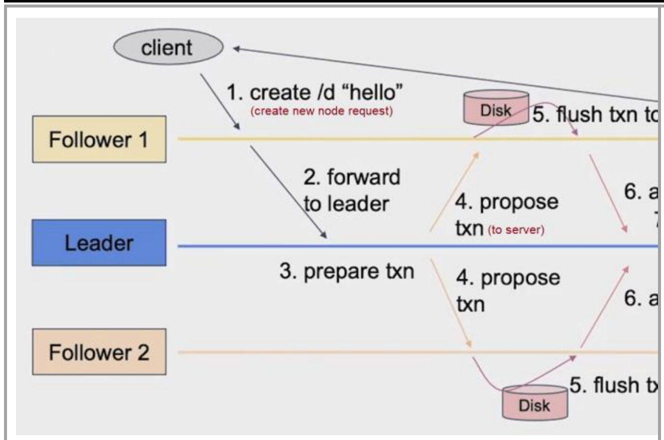
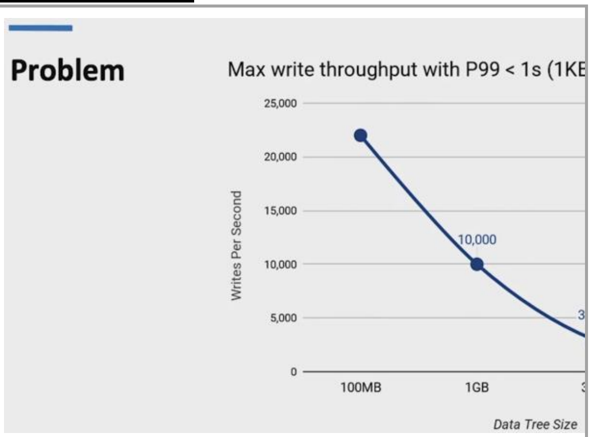
## Zookeeper @ Facebook

- Facebook runs on ZK today and has a dedicated ZK team (8 engineers as of 01/2020) that contributes to hundreds of ZK use cases across FB engineering domains.
- ZK started @ FB circa 2014 for initial canonical use case of configuration & metadata management along with service discovery. Continued & expanded usage from then onwards with heavy current usage.
- ZK also being used for distributed semaphores & locks, membership management, and leader election @ FB
- Facebook running on 3.6 (FB internal ZK stack revision) in master for more than 2 years due to ZK enhancements (Peter considers that a notable achievement and vote of confidence in the stability of that branch)
- ZK engineering team continues to upstream new features, bug fixes, and enhancements around performance and reliability to its internal ZK stack.
- Recent updates on ZK @ FB (01/2020):
  - TLS enabled for all client connections (major effort due to broad distribution of clients)
  - ZK transitioned from bare metal to FB's container platform
  - Dramatic scale & efficiency improvements
  - Upstreamed almost all features (excluding C++ facebook ZK client) in version 3.6
  - 12 inconsistency & data corruption bugs fixed as of 01/2020
  - 120 patches upstreamed between 2019-2020
- Key ZK features slated for upstream in FY20:
  - Inspect API
    - Continuously monitors performance bottlenecks across ZK stack
  - Snapshot scheduler
    - Reduces disk I/O contention in ZK and expands maximum data traffic volume
  - Batch ACK and Commit
    - Extends overall max throughput thresholds in ZK
  - Failure request short circuit
    - Improves ZK quality & efficiency
- Additional ZK features slated for upstream:
  - Server-side semaphore API
  - Service isolation & demand control
  - Live traffic migration w/ ensemble split feature
  - Global session inconsistency detector
  - OOM (memory) protection

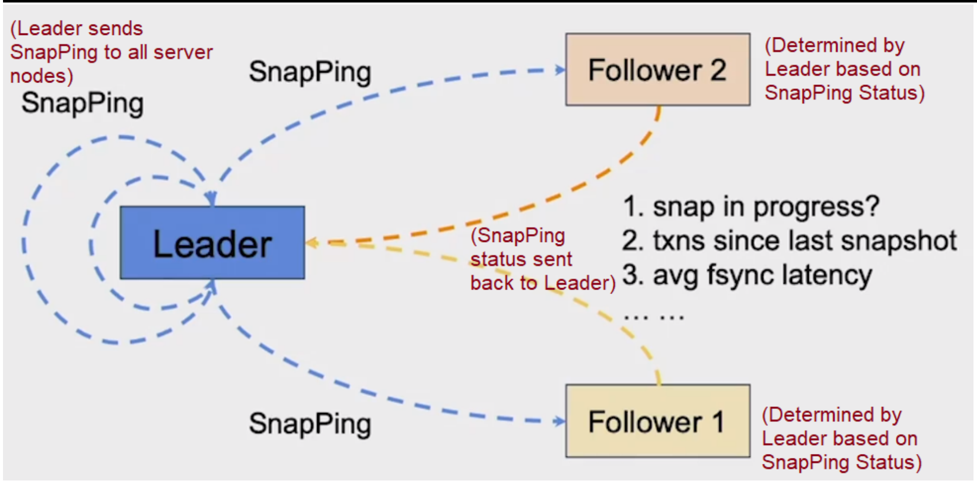
**Facebook's Zookeeper Snapshot Scheduler Improvements:**

Scheduler Write Flow (High I/O Latency Due to Disk Flush Concurrency):





Scheduler Write Flow Improvment (Simplified. Solves for High Latency via SnapPing Request/ACK):


(Leader sends SnapPing to all server nodes)
SnapPing
SnapPing
Follower 2
(Determined by Leader based on SnapPing Status)
Leader
(SnapPing status sent back to Leader)
1. snap in progress?
2. txns since last snapshot
3. avg fsync latency
... ...
SnapPing
Follower 1
(Determined by Leader based on SnapPing Status)