



Serving millions of journals with Apache BookKeeper

Flavio Junqueira
Microsoft Research

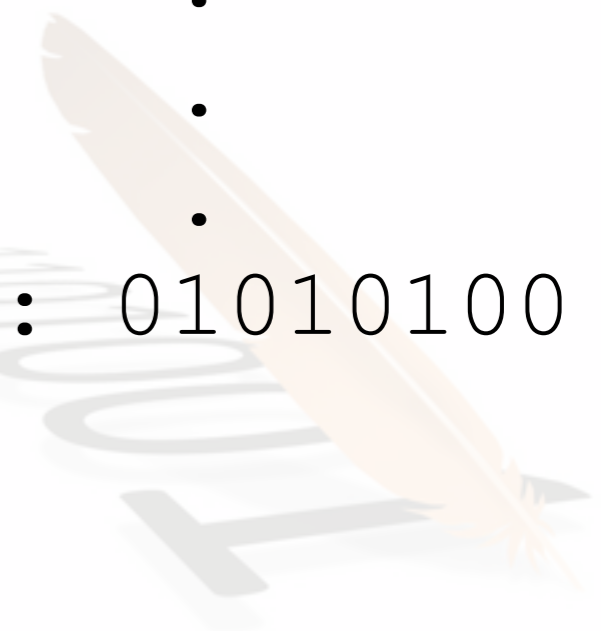
Ivan Kelly
Yahoo! Research

ZooKeeper/BookKeeper committers
ZooKeeper PMC members

What's BookKeeper?

- Storage for sequences of byte arrays
- Append only
- Single writer
- Distributed and replicated
- Tolerates crashes of storage servers
- Lots of them concurrently...

```
<1> : 00010110  
<2> : 11001101  
<3> : 01010101  
<4> : 00101011  
      .  
      .  
      .  
<n> : 01010100
```

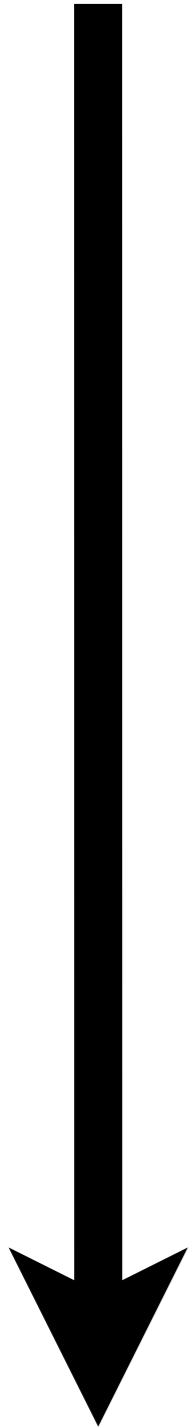


Why BookKeeper?

- Logging is a common problem
- Examples
 - ✓ Journaling (e.g., HDFS Namenode)
 - ✓ Write-ahead log (e.g., HBase)
 - ✓ Message durability (e.g., ActiveMQ, Hedwig)
- Not easy to get it right
 - ✓ Fault tolerance
 - ✓ Concurrent writes

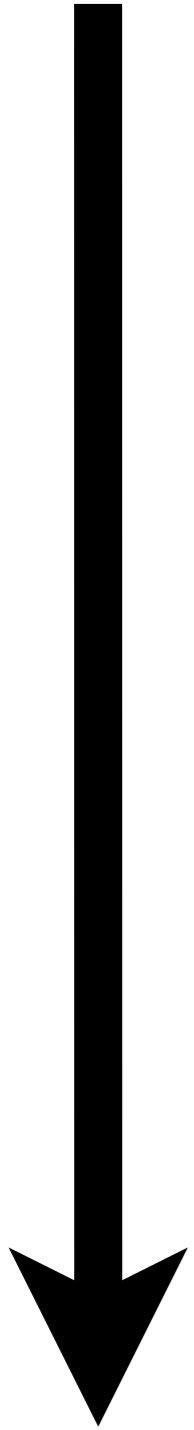


A historical perspective



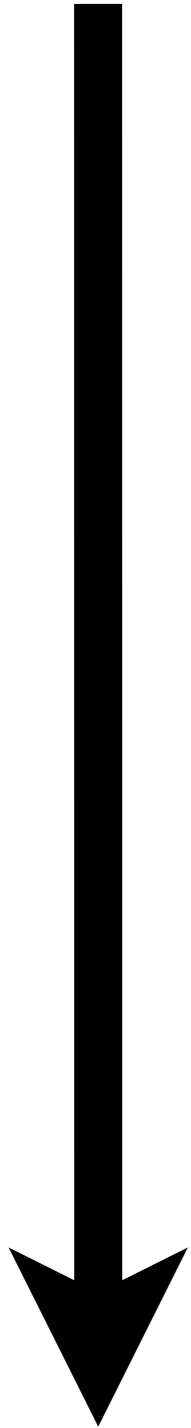
A historical perspective

- Projects starts around mid 2008
 - ▶ **Namenode** was the motivating use case



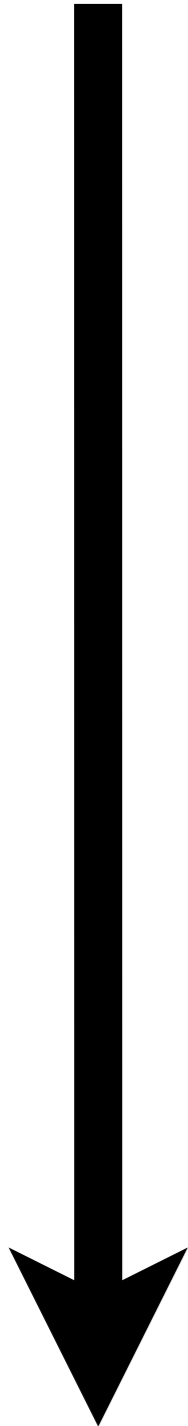
A historical perspective

- Projects starts around mid 2008
 - ▶ **Namenode** was the motivating use case
- ZooKeeper contrib by the end of 2008

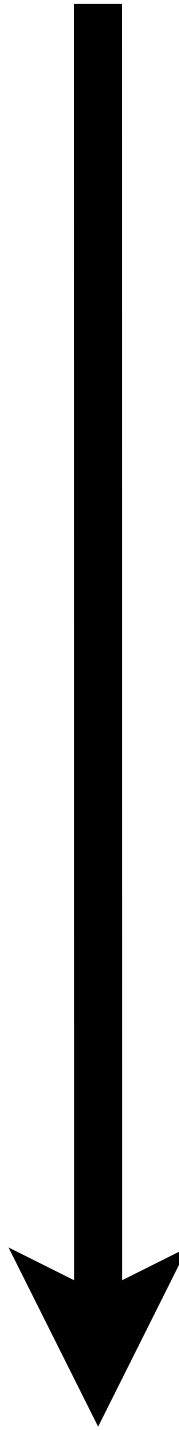


A historical perspective

- Projects starts around mid 2008
 - ▶ **Namenode** was the motivating use case
- ZooKeeper contrib by the end of 2008
- Mid 2009, new use case: **PNUTS** via **Hedwig**
 - ▶ Topic-based, guaranteed delivery pub-sub

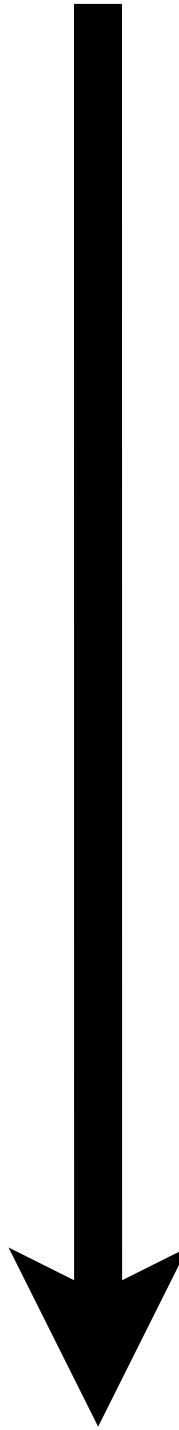


A historical perspective

- 
- Projects starts around mid 2008
 - ▶ **Namenode** was the motivating use case
 - ZooKeeper contrib by the end of 2008
 - Mid 2009, new use case: **PNUTS** via **Hedwig**
 - ▶ Topic-based, guaranteed delivery pub-sub
 - Backburner, but progress on
 - ▶ **Namenode** pluggable journal, **Hedwig**

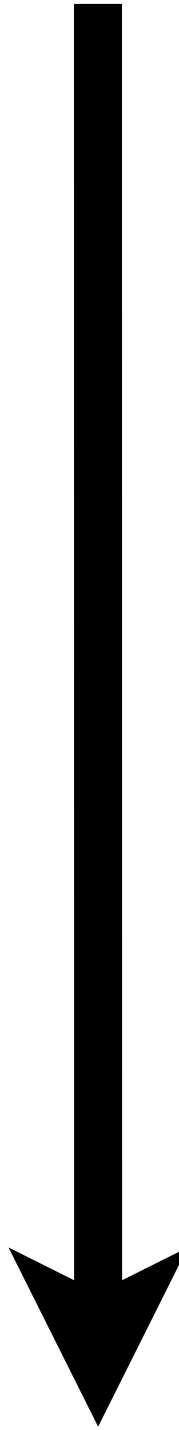


A historical perspective

- 
- Projects starts around mid 2008
 - ▶ **Namenode** was the motivating use case
 - ZooKeeper contrib by the end of 2008
 - Mid 2009, new use case: **PNUTS** via **Hedwig**
 - ▶ Topic-based, guaranteed delivery pub-sub
 - Backburner, but progress on
 - ▶ **Namenode** pluggable journal, **Hedwig**
 - Beginning of 2011 → Subproject

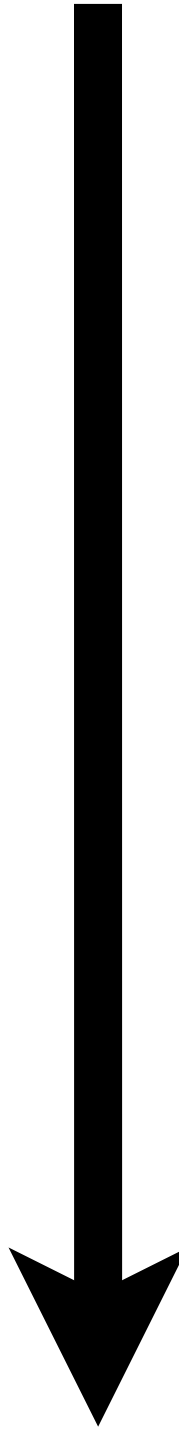



A historical perspective

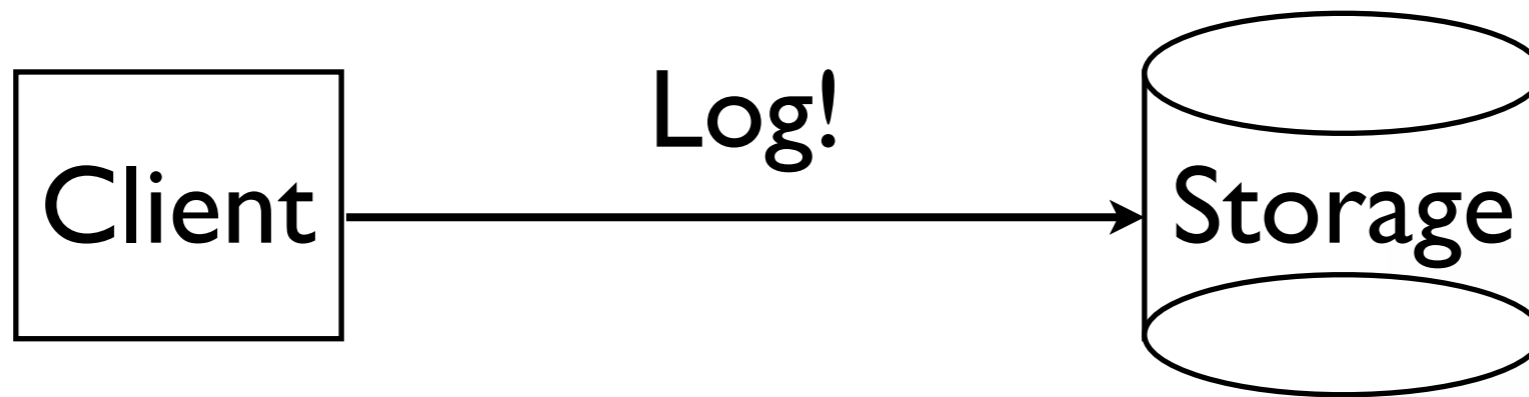
- 
- Projects starts around mid 2008
 - ▶ **Namenode** was the motivating use case
 - ZooKeeper contrib by the end of 2008
 - Mid 2009, new use case: **PNUTS** via **Hedwig**
 - ▶ Topic-based, guaranteed delivery pub-sub
 - Backburner, but progress on
 - ▶ **Namenode** pluggable journal, **Hedwig**
 - Beginning of 2011 → Subproject
 - Also in 2011, new use case: *Push notifications*



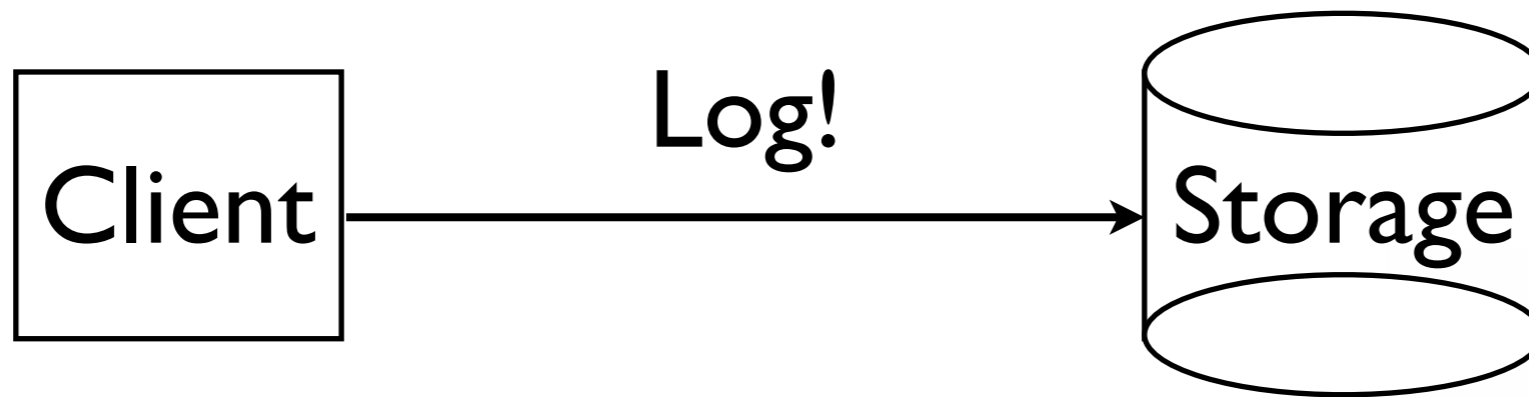
A historical perspective

- 
- Projects starts around mid 2008
 - ▶ **Namenode** was the motivating use case
 - ZooKeeper contrib by the end of 2008
 - Mid 2009, new use case: **PNUTS** via **Hedwig**
 - ▶ Topic-based, guaranteed delivery pub-sub
 - Backburner, but progress on
 - ▶ **Namenode** pluggable journal, **Hedwig**
 - Beginning of 2011 → Subproject
 - Also in 2011, new use case: *Push notifications*
 - In 2012, production (!) and new use cases...
- 

Deriving a design



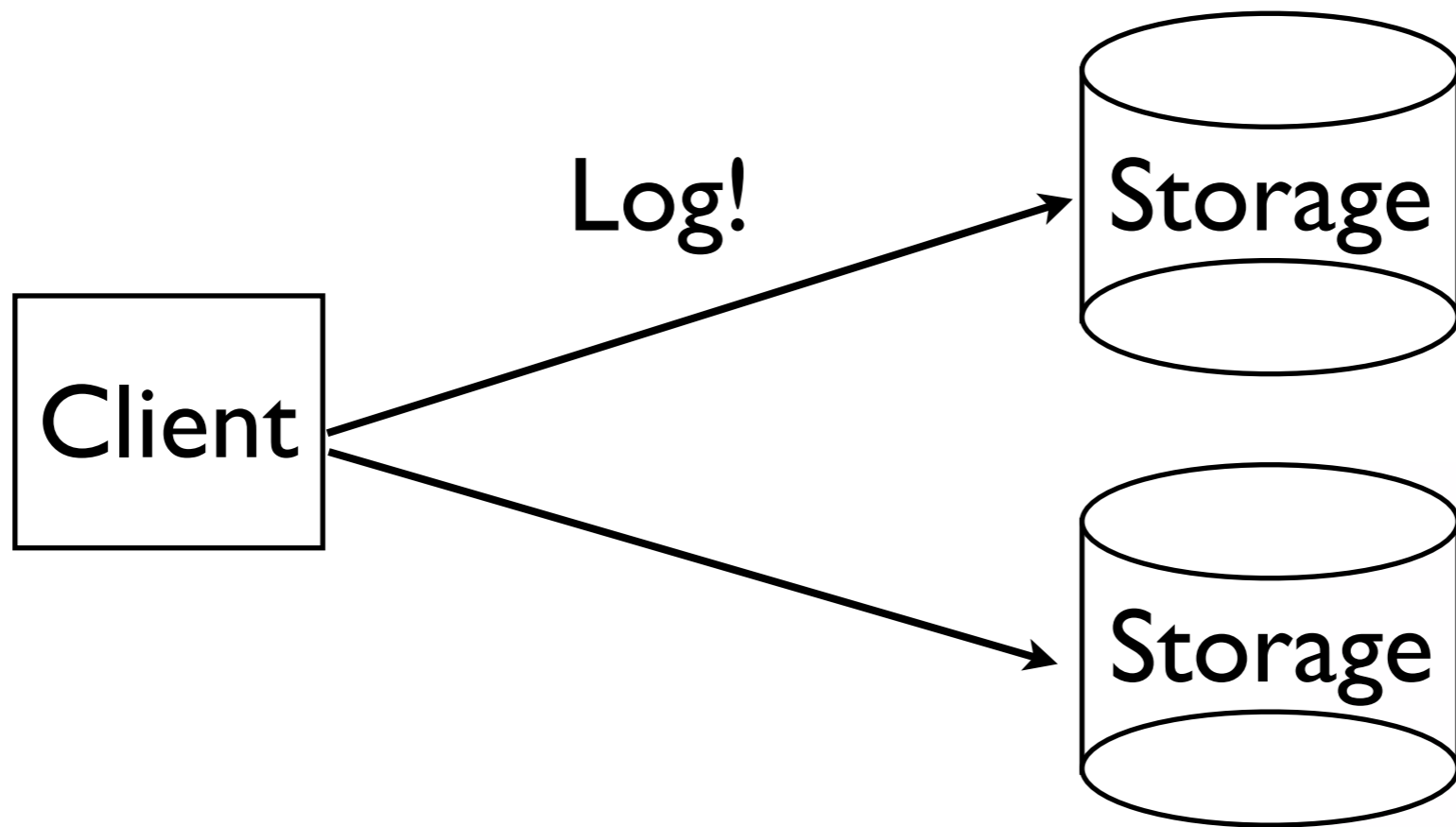
Deriving a design



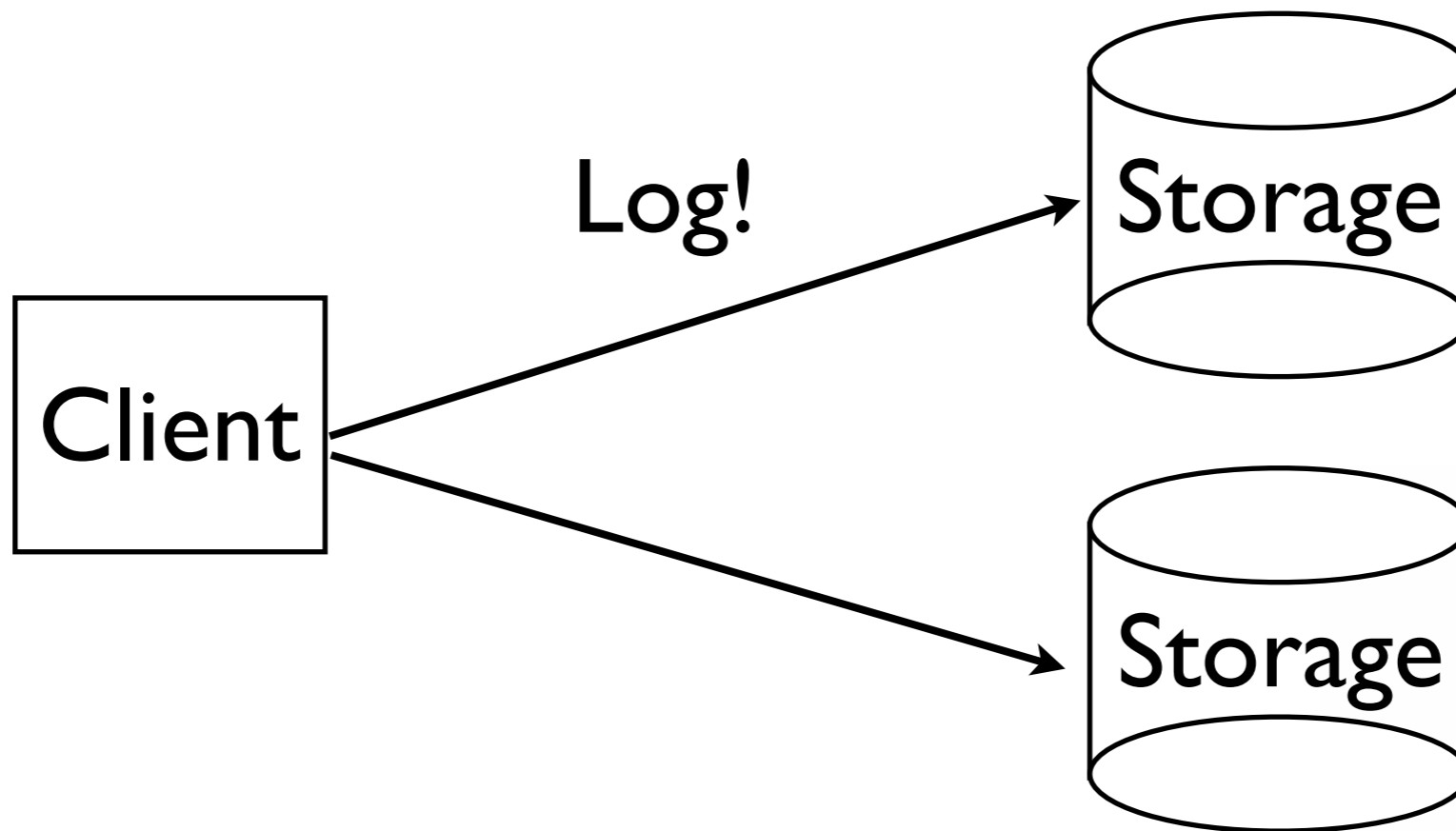
Problem: Not fault tolerant



Deriving a design

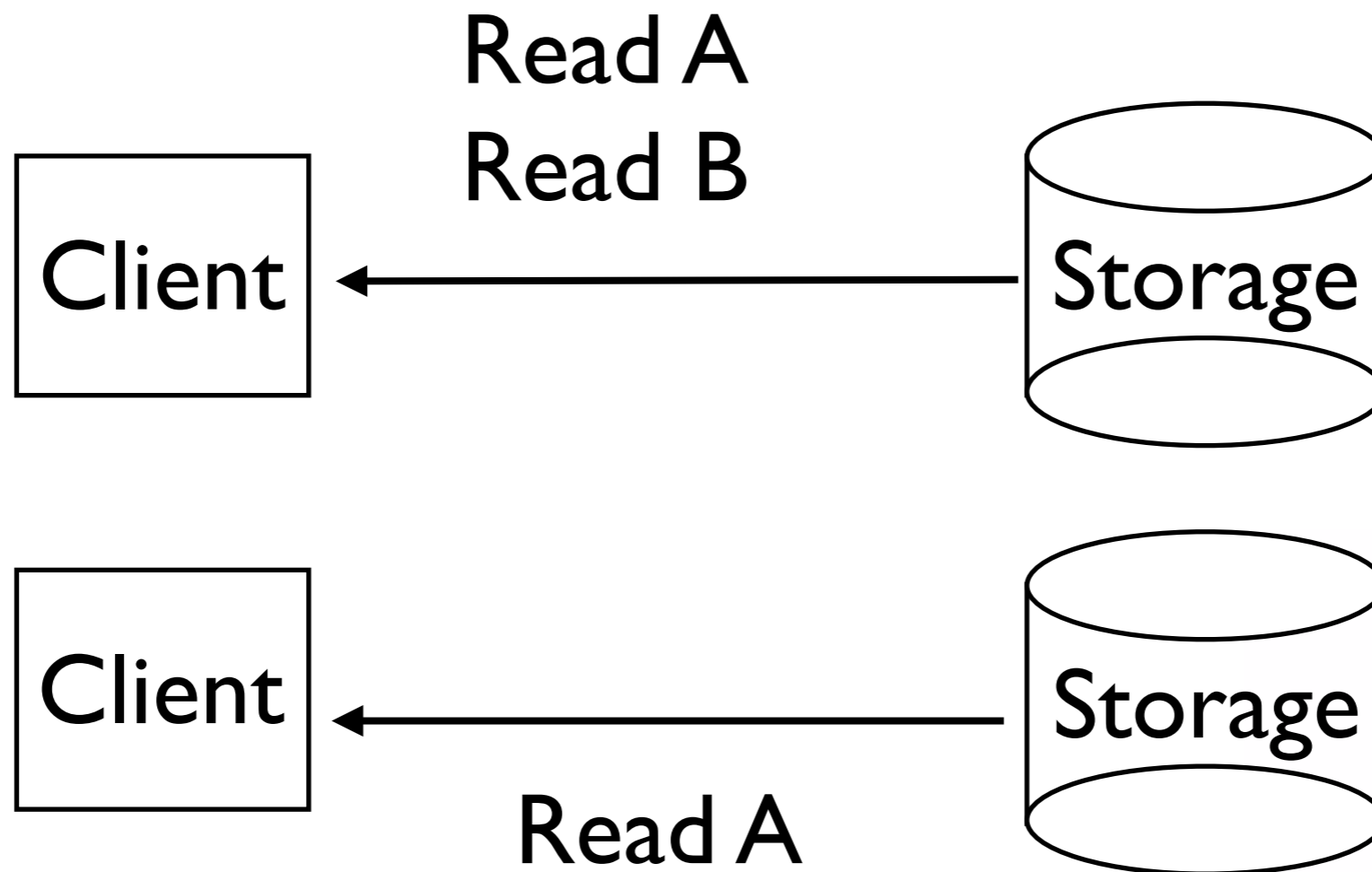


Deriving a design



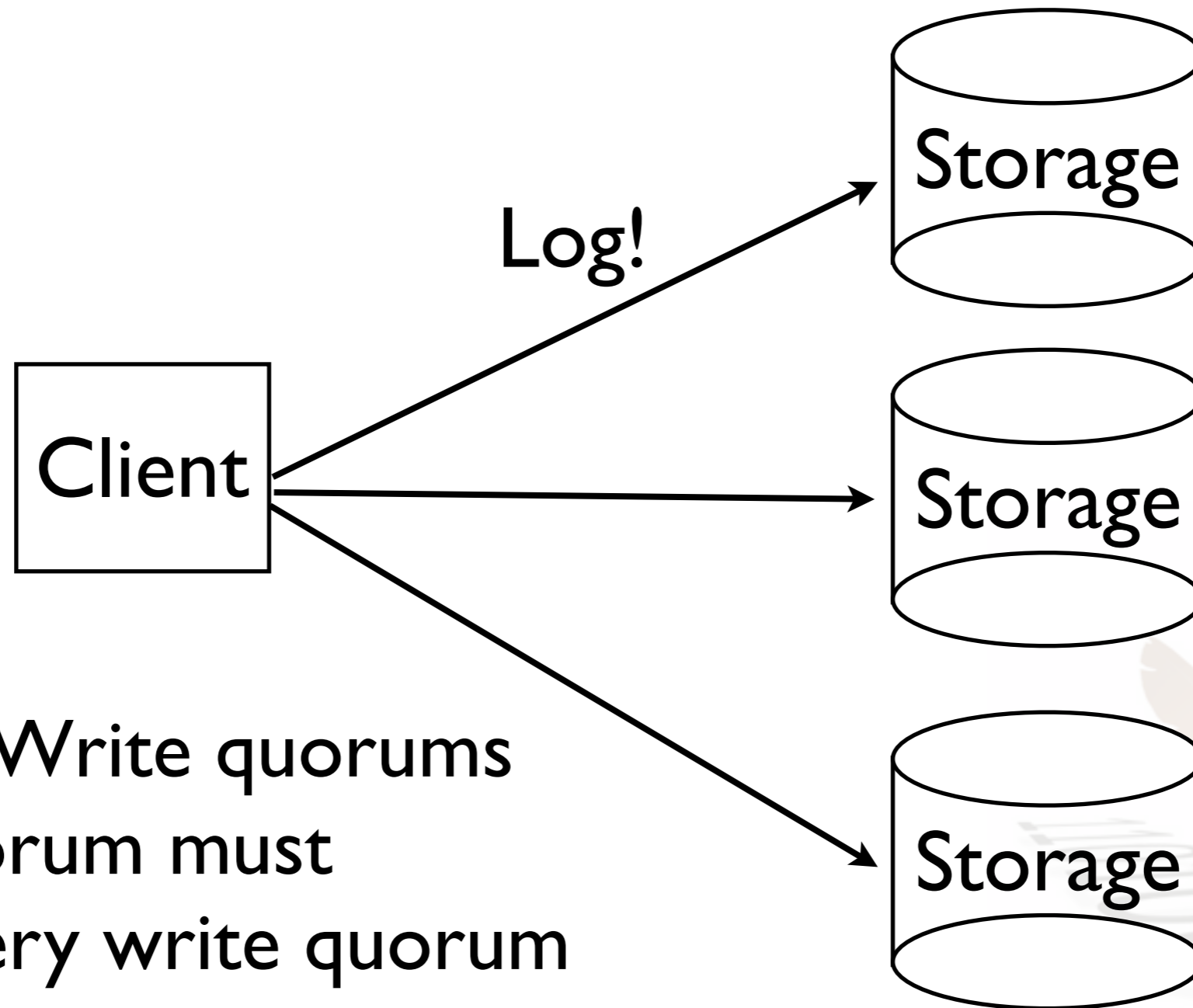
Problems are more subtle now,
Inconsistent reads are possible

Deriving a design



Problems are more subtle now,
Inconsistent reads are possible

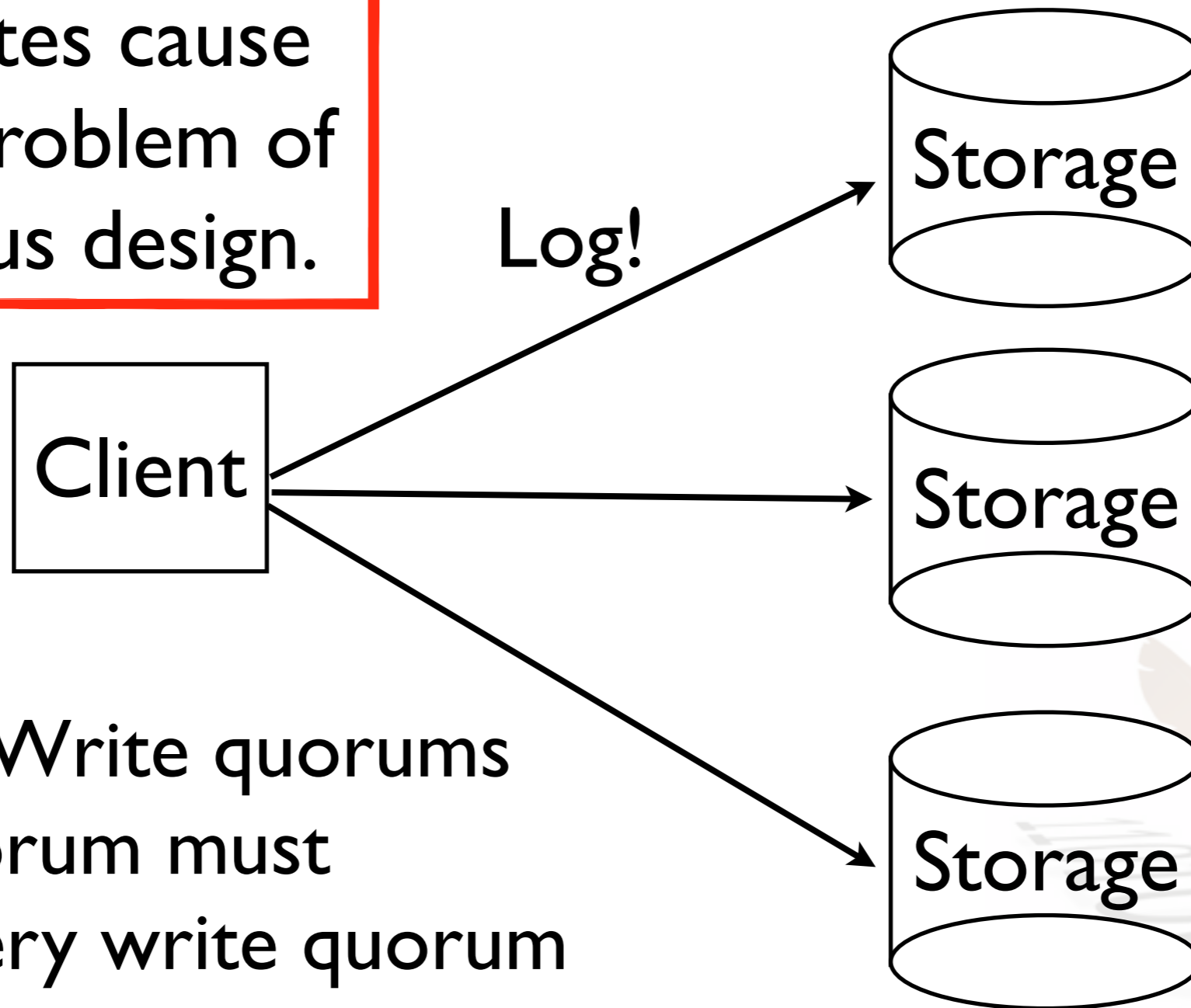
Deriving a design



- Use Read-Write quorums
- A read quorum must intersect every write quorum
- *E.g.*, quorums are majorities

Deriving a design

Partial writes cause the same problem of the previous design.



- Use Read-Write quorums
- A read quorum must intersect every write quorum
- *E.g.*, quorums are majorities

We need consensus somewhere...



Deriving a design

- Can't simply read from a quorum
 - ✓ Partial writes
 - ✓ Might not contain such writes
- Can only read what has been fully replicated
- With Paxos
 - ✓ Read from a quorum
 - ✓ Write back before it learns



Wait, do we really need to go through all this
consensus trouble?



Can I use something that already exists?



Can I use something that already exists?

Yes, ZooKeeper!

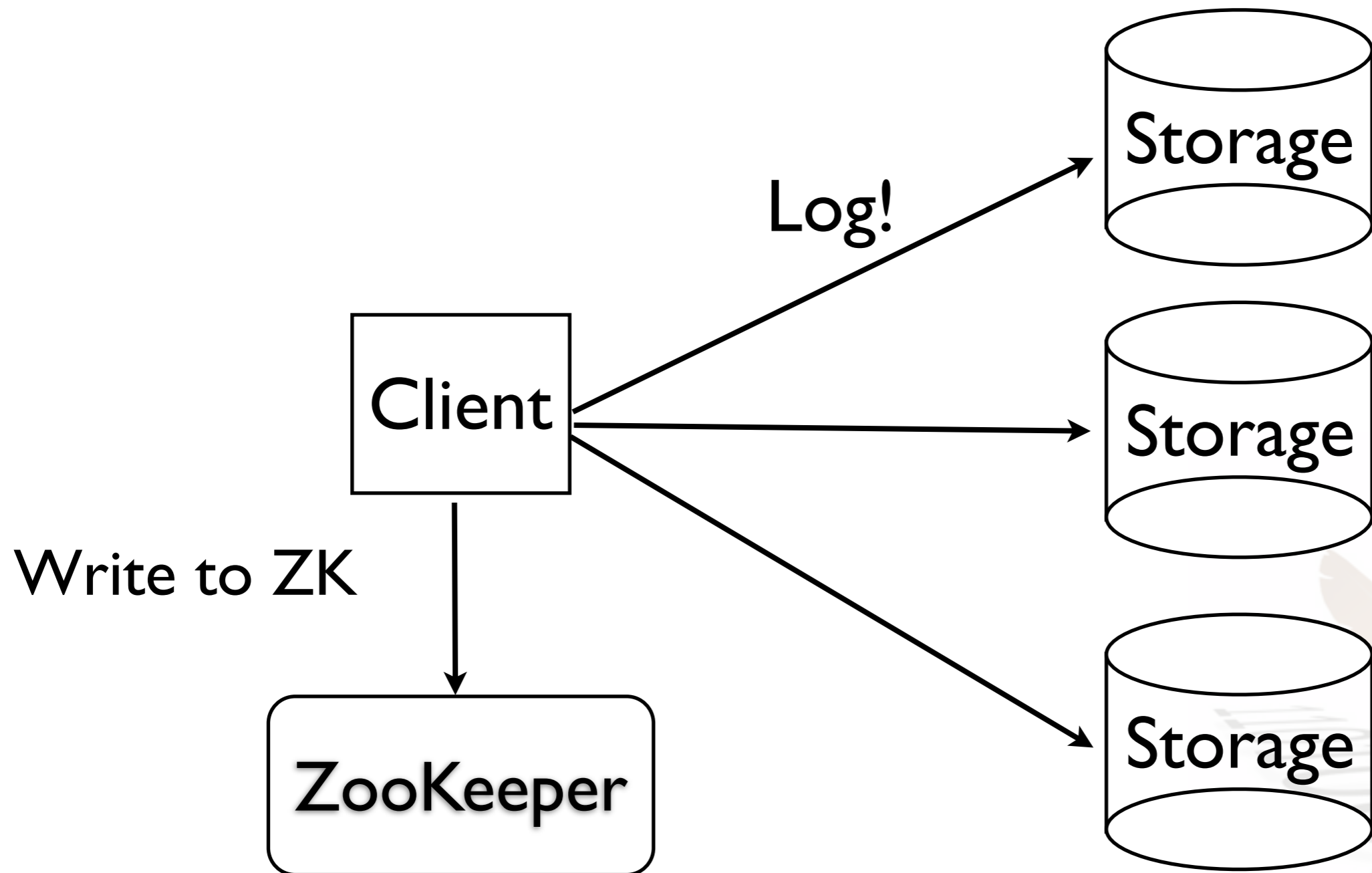


Introducing ZooKeeper

- What does ZooKeeper give me?
 - ✓ Totally ordered writes
 - ✓ Acknowledgment only when fully replicated
- Doesn't expose a "consensus" interface
- Equally powerful, though

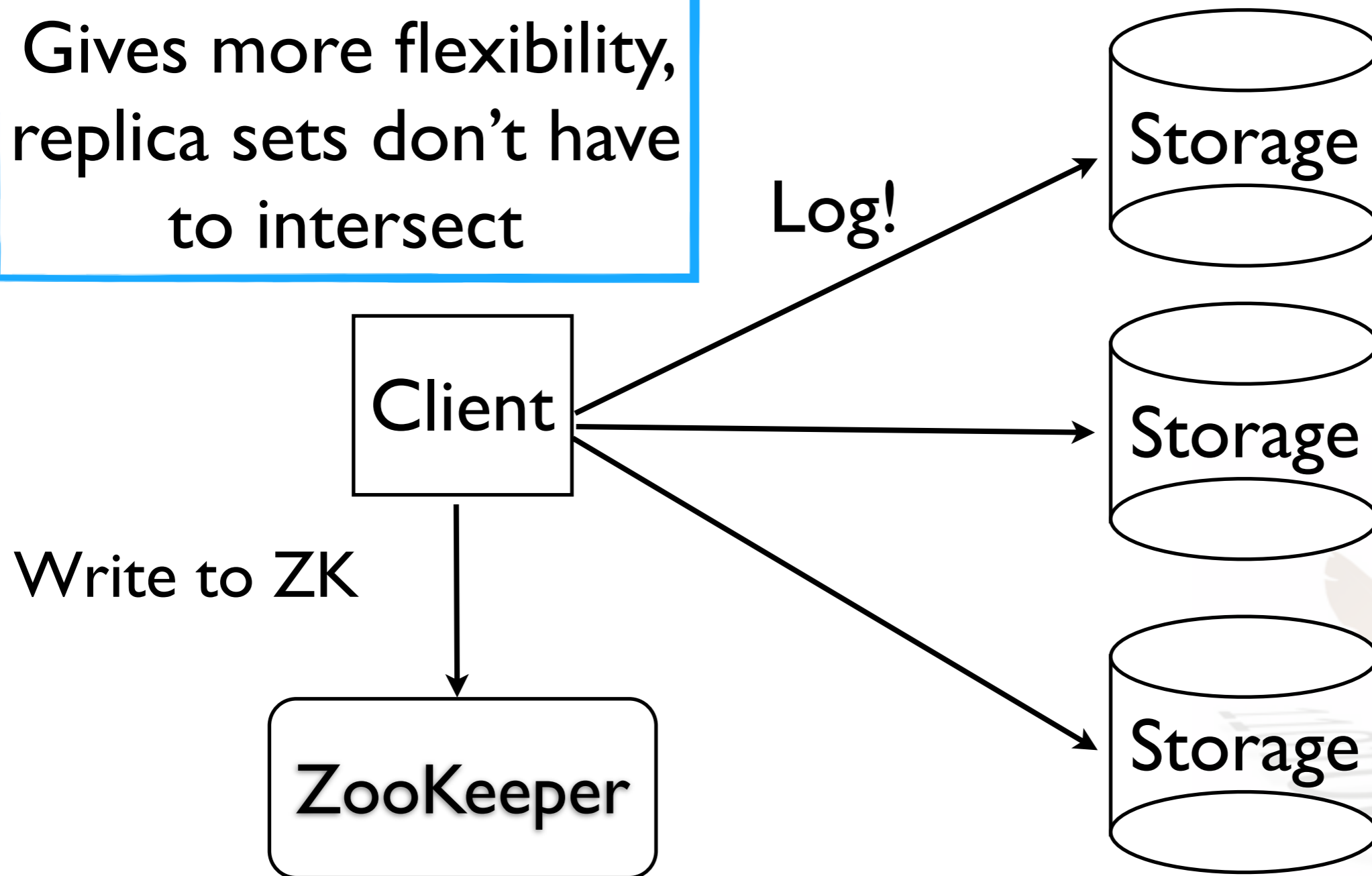


Introducing ZooKeeper

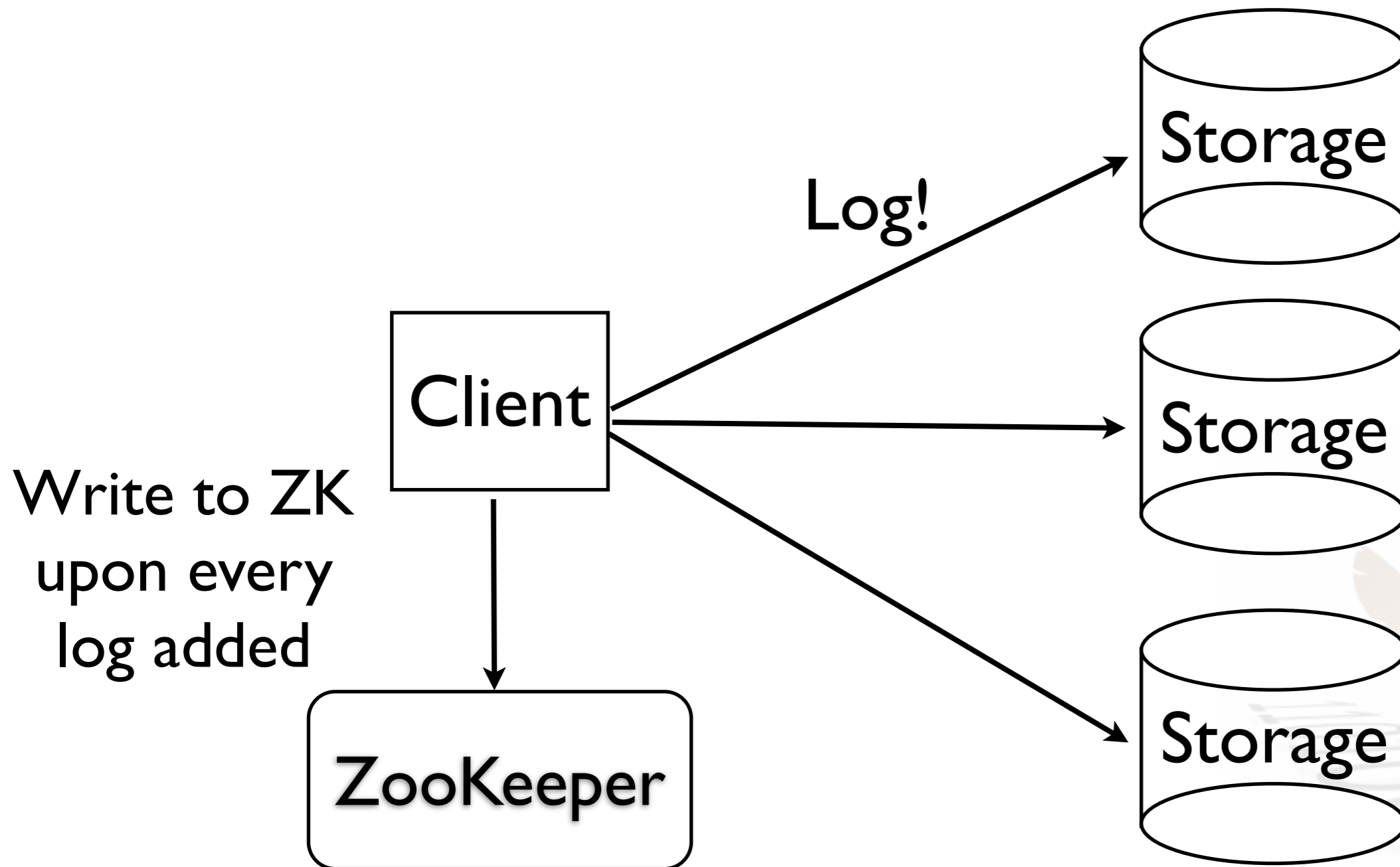


Introducing ZooKeeper

Gives more flexibility,
replica sets don't have
to intersect

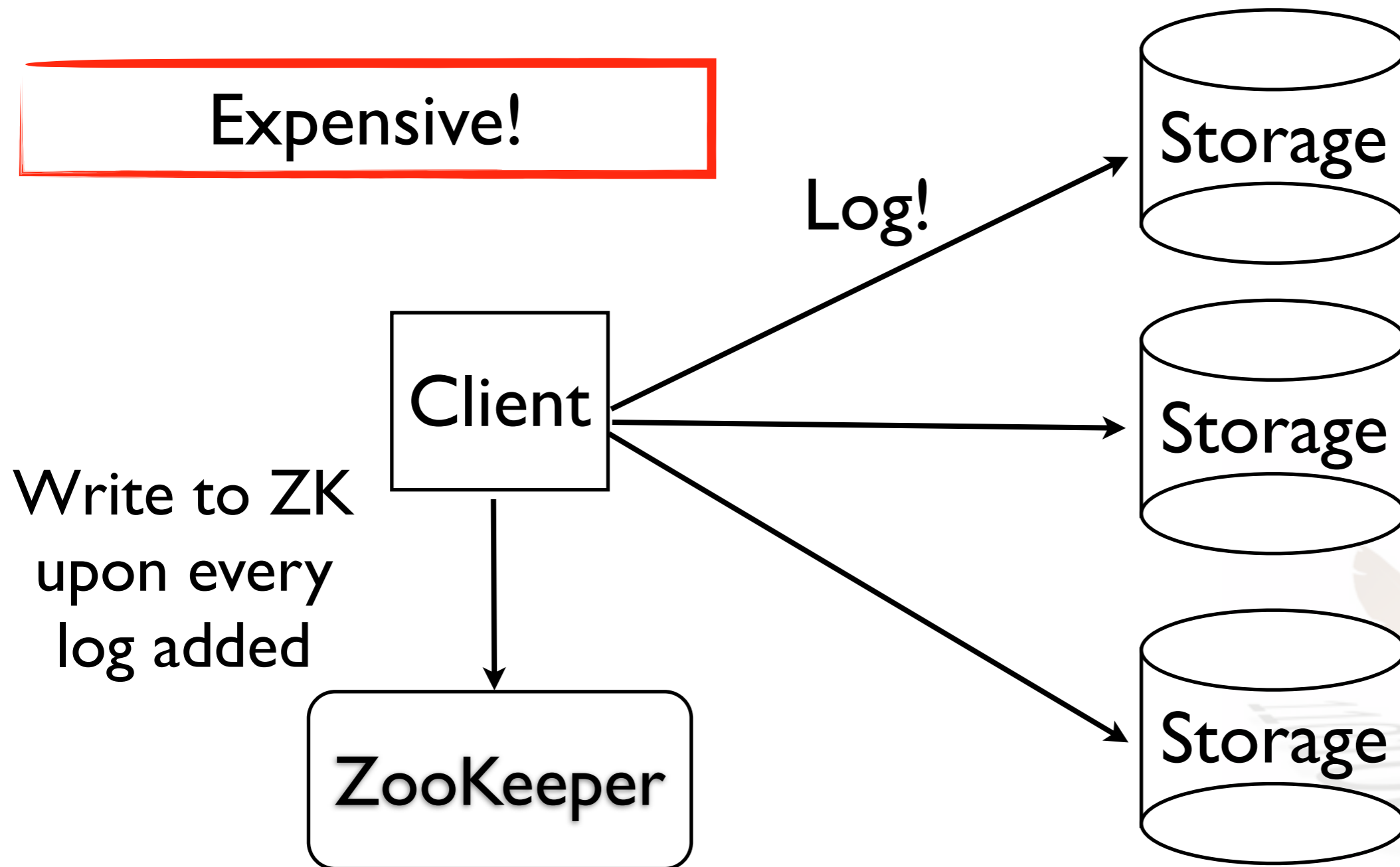


Introducing ZooKeeper

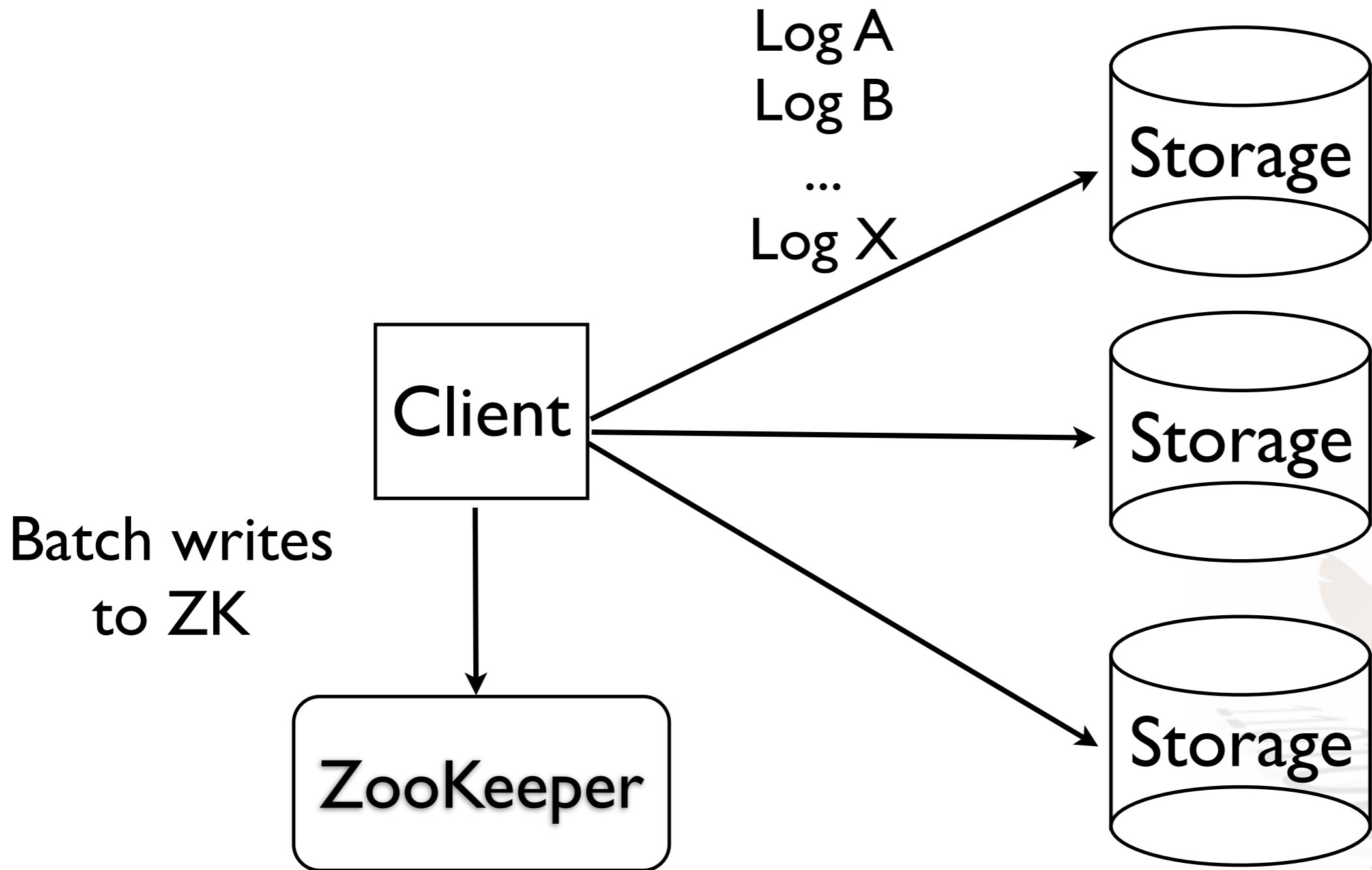


Introducing ZooKeeper

Expensive!



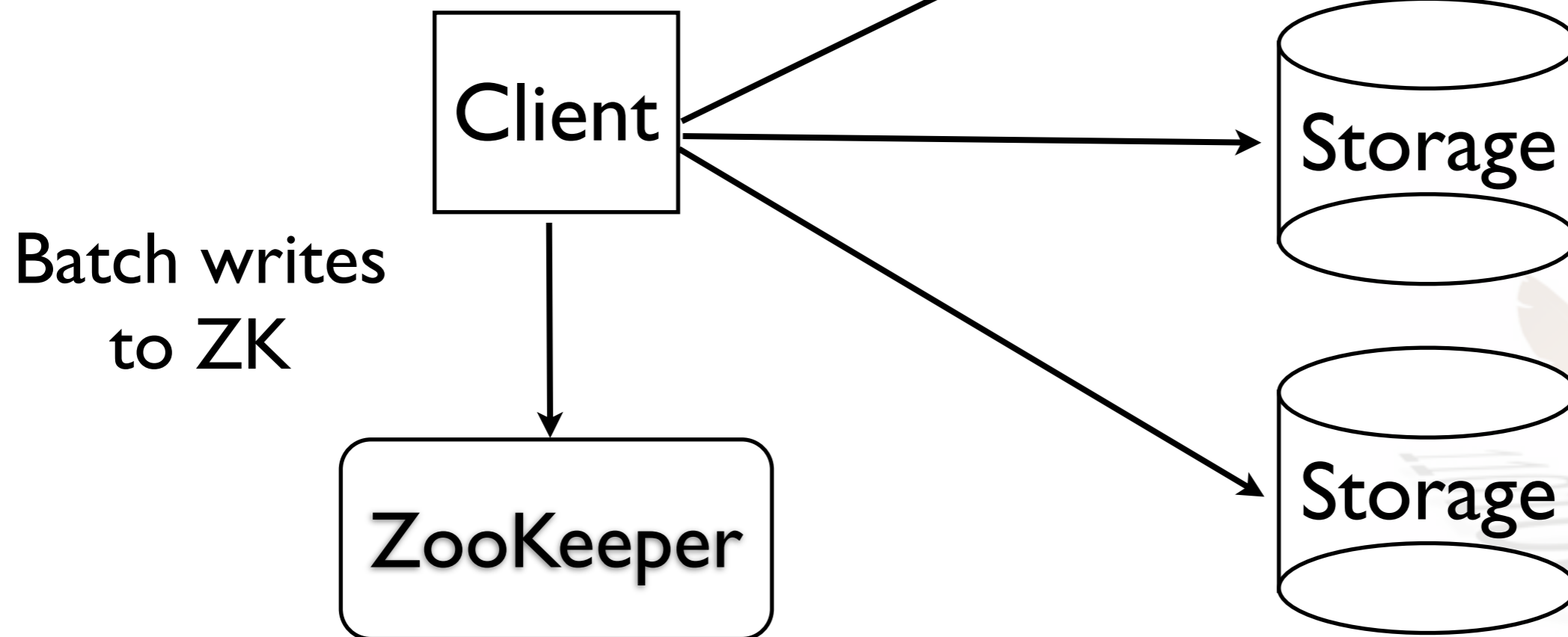
Introducing ZooKeeper



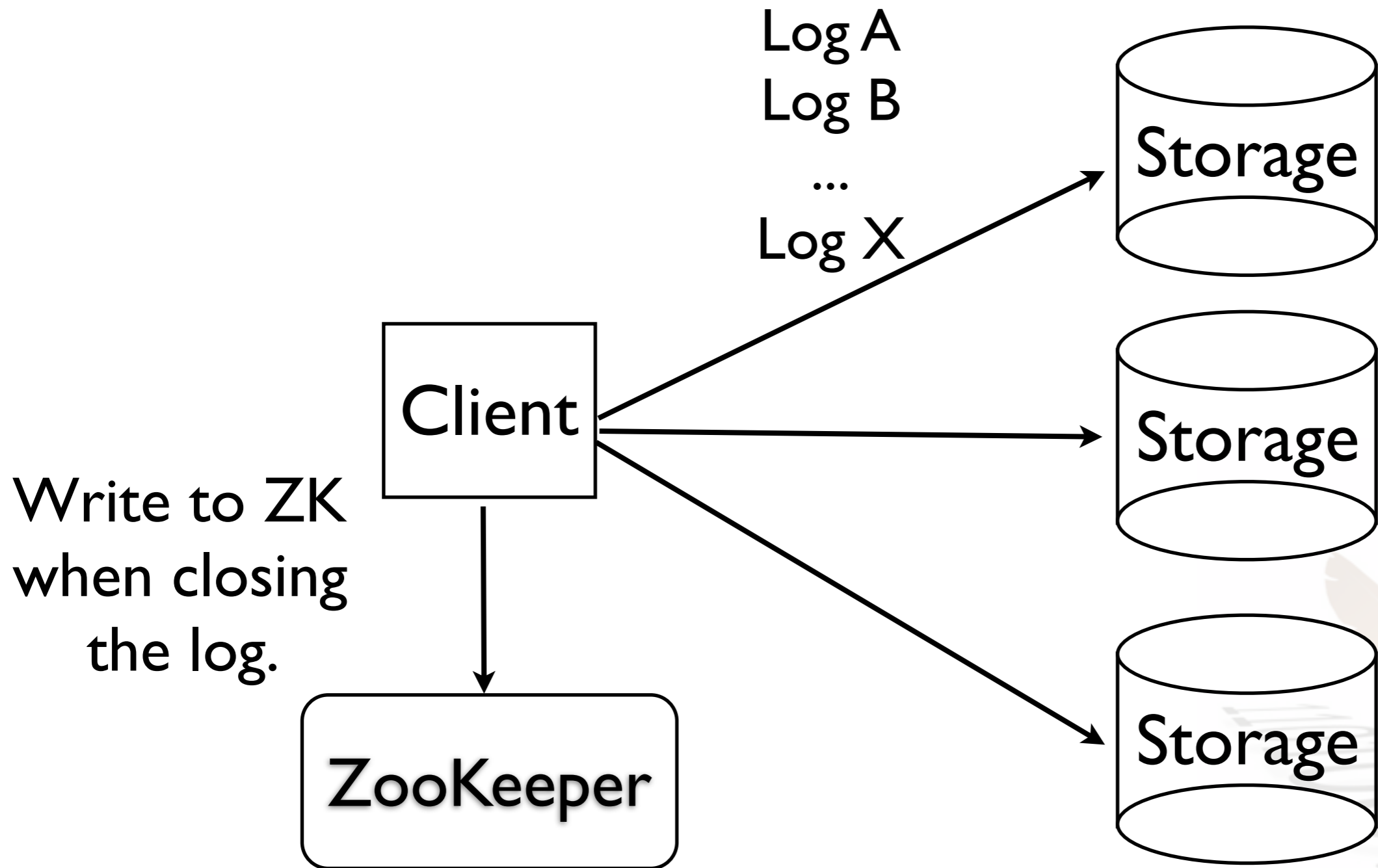
Introducing ZooKeeper

We use a variant of this technique

Log A
Log B
...
Log X



Introducing ZooKeeper

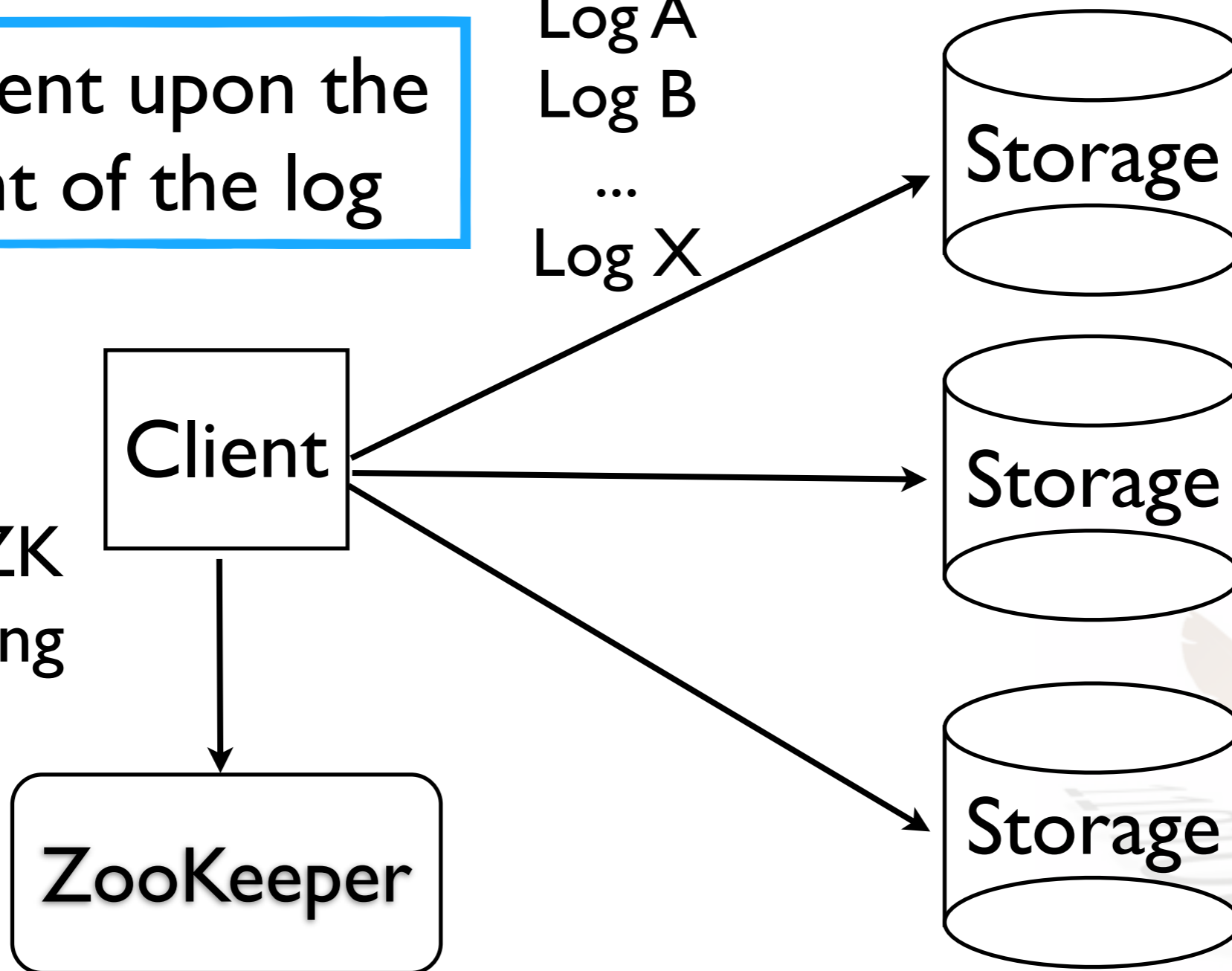


Introducing ZooKeeper

Agreement upon the content of the log

Log A
Log B
...
Log X

Write to ZK
when closing
the log.





Making it more concrete

Concepts

- Clients are *clients*
 - ✓ Access via a BookKeeper client
- Log files are *ledgers*
- Each element of a ledger is an *entry*
- Storage nodes are *bookies*



API summary

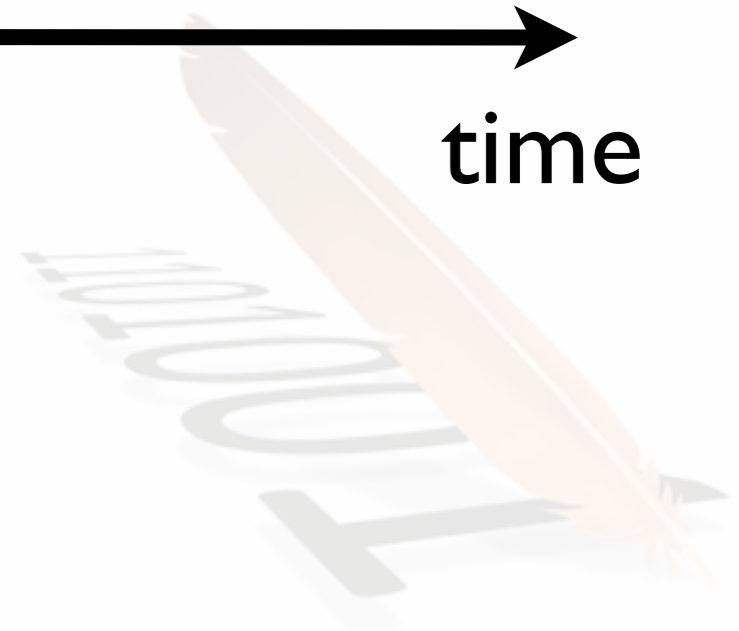
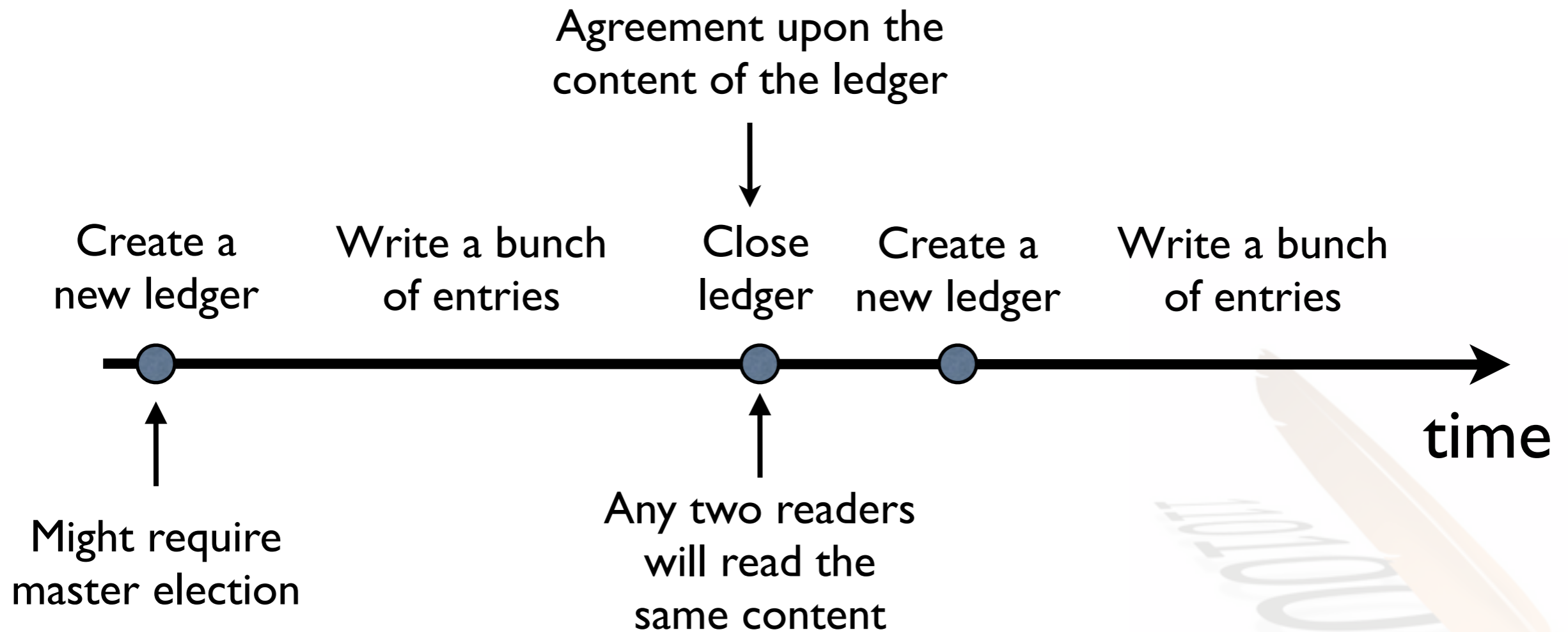
- BookKeeper class
 - ✓ Create ledger
 - ✓ Open ledger
- LedgerHandle class
 - ✓ Close ledger
 - ✓ Add to ledger
 - ✓ Read from ledger

API summary

- BookKeeper class
 - ✓ Create ledger
 - ✓ Open ledger
- LedgerHandle class
 - ✓ Close ledger
 - ✓ Add to ledger
 - ✓ Read from ledger

Synchronous
and
Asynchronous

Ledger cycle



Bookies

- Write intensive service
 - ✓ High throughput and low latency for writes
 - ✓ Writes to many ledgers at a time
- Reads
 - ✓ Separate device
 - ✓ Minimize interference with writes



BK as a Service

- Pool of bookies
 - ✓ Ability to add and remove bookies
 - ✓ Register and unregister from ZooKeeper
 - ✓ ZooKeeper is already there, right?
- Auto-replication
 - ✓ Replicate ledger entries of faulty bookies
 - ✓ New to version 4.2.0

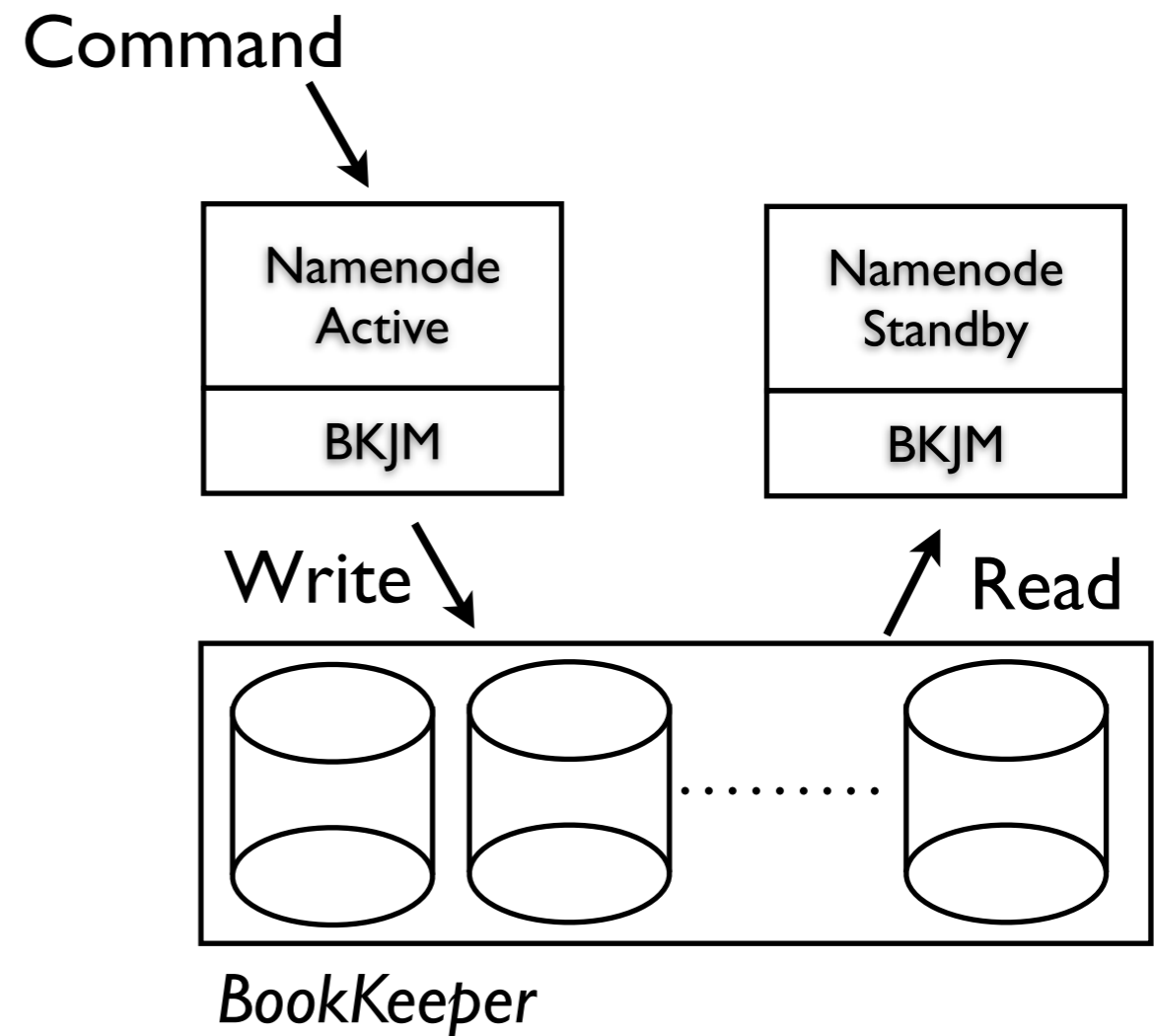




Use cases

BKJM

- Hadoop DFS
 - **Namenode**
- Datanodes
- BookKeeper Journal Manager
 - A journal manager for the Namenode
 - Replaces shared storage (e.g., filer)

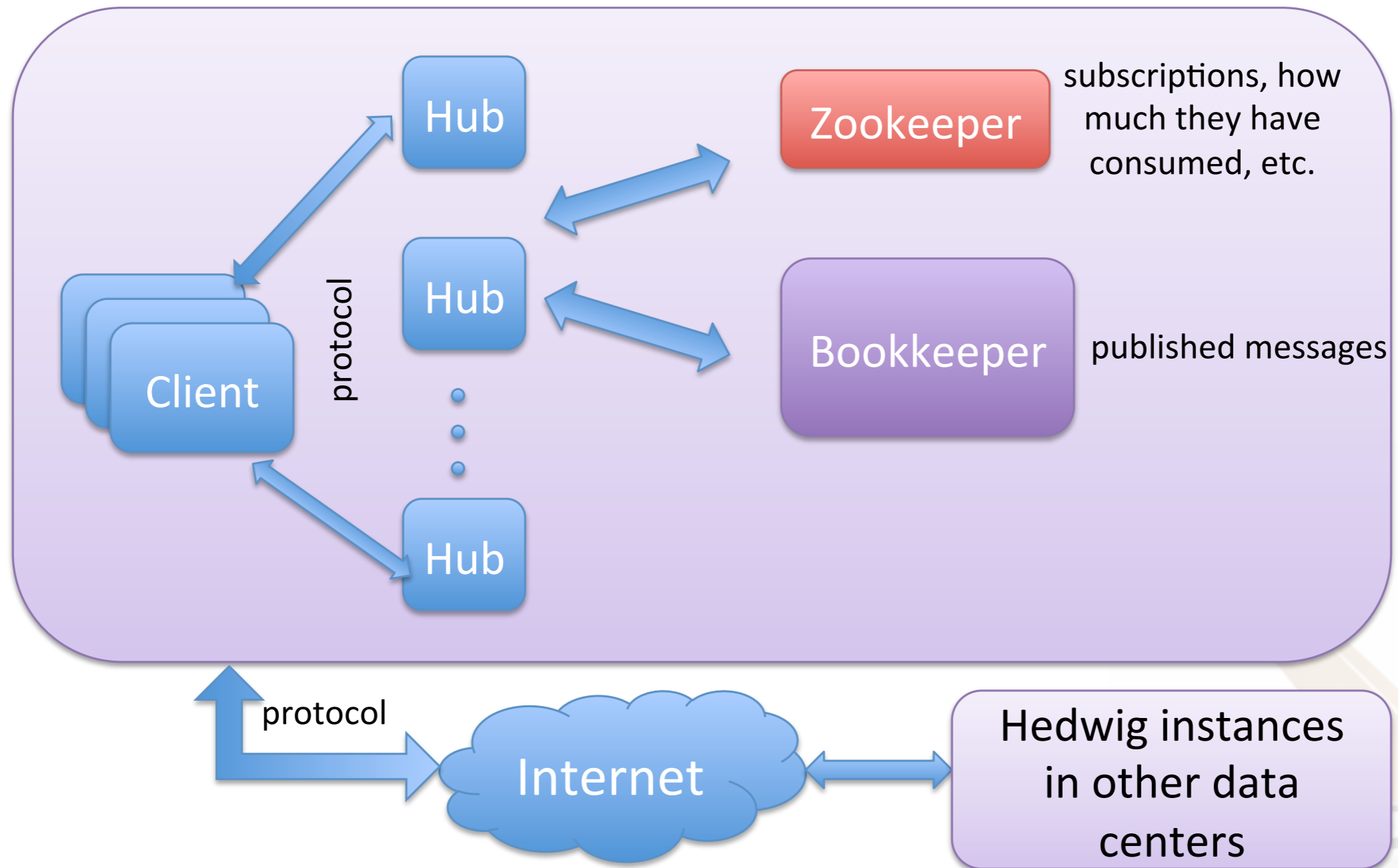


Hedwig

- Multi-region pub/sub system
 - ✓ Multiple data centers
- Guaranteed-delivery topic-based pub-sub system
- Elastically scalable
 - ✓ Deployed over commodity machines
 - ✓ Capacity can be added on-the-fly by adding machines
- Low Operational Complexity
 - ✓ Tolerate failures without manual intervention
 - ✓ Automatic load balancing



Hedwig overview

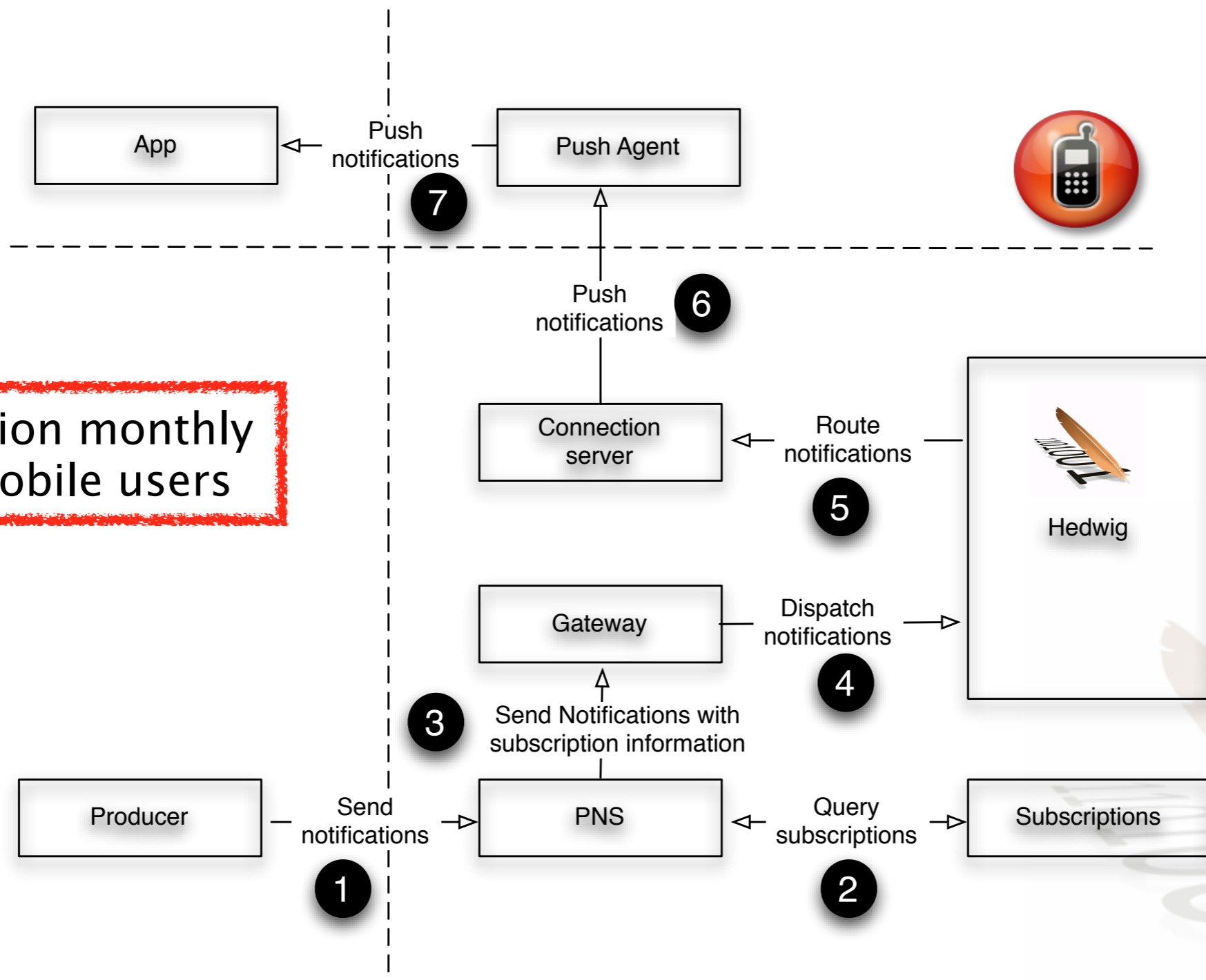


Push notifications

- Users of mobile devices
- Notifications
 - ✓ News alerts, social network updates, email, etc.
- Pushing is typically preferable over polling
 - ✓ Lower latency
 - ✓ Saves on battery

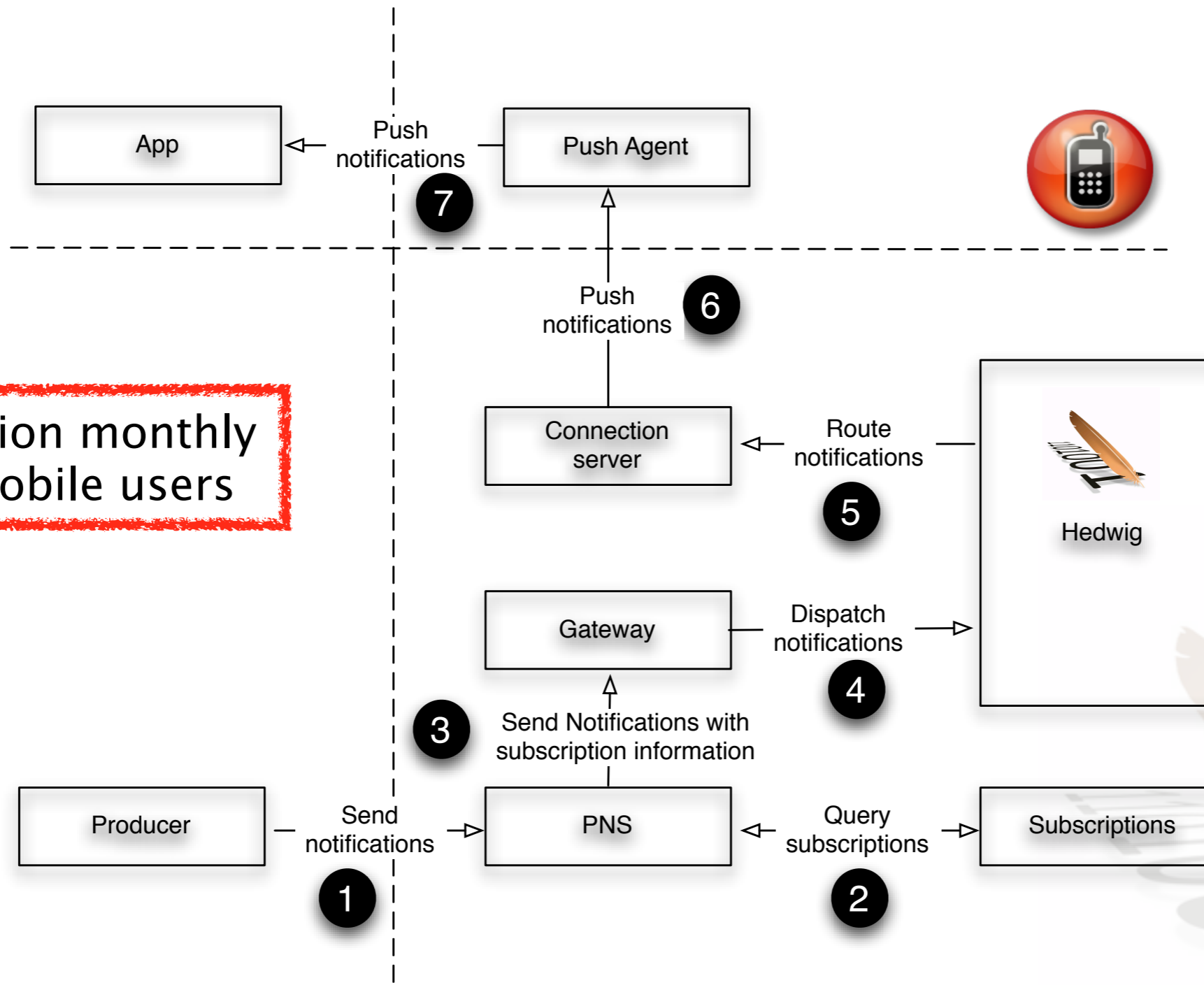


Push notifications



<http://developer.yahoo.com/blogs/ydn/posts/2012/11/bookkeeper-durability-at-scale/>

Push notifications



200 million monthly active mobile users



Designed to serve tens to hundreds of millions of users

<http://developer.yahoo.com/blogs/ydn/posts/2012/11/bookkeeper-durability-at-scale/>

Challenges

- Metadata
 - ✓ One ZooKeeper installation is not enough
 - ✓ Metadata store interface (e.g., HBase)
- Ideal workload
 - ✓ Long-lived ledgers
 - ✓ Amortized cost of metadata ops is low
 - ✓ ... the notifications use case is not always like that





Community

Active committers

- Sijie Guo (Twitter)
- Flavio Junqueira, PMC (Microsoft Research)
- Ivan Kelly, PMC (Yahoo! Research)
- Uma Maheswara Rao G (Huawei)
- Ben Reed (Facebook)
- Jiannan Wang (Yahoo!)



Companies using it

- Yahoo!
 - ✓ Push notifications
 - ✓ Cloud messaging
- Huawei
 - ✓ BKJM
- Hubspot
 - ✓ User logging

Conclusions

- Apache BookKeeper
 - ✓ Shared storage for logging
 - ✓ Targets durability
- Use cases
 - ✓ HDFS Namenode
 - ✓ Hedwig (e.g., used with push notifications)
 - ✓ Cloud Messaging



Conclusions

- Active community
 - ✓ Diverse set of committers
 - ✓ A few important use cases
 - ✓ ... looking forward to growing!





Apache BookKeeper
<http://zookeeper.apache.org/bookkeeper>