ZBS : SmartX Distributed Block Storage

Kyle Zhang
Cofounder & CTO
SmartX

2018-04-19

About Me

- Kyle Zhang 张凯
- Computer Science, Tsinghua University
- Software Engineer, Infrastructure Team, Baidu
- Cofounder & CTO, SmartX
- Contributor, Sheepdog/InfluxDB

About SmartX

smartx
MAKE IT SIMPLE

- Founded by three ***geeks*** in 2013
- Focuses on ***distributed storage*** and ***virtualization***
- *Currently* ***leading*** in distributed block storage technology
- Product has been deployed on ***thousands*** of hosts and running for 2 years
- Cooperates with Chinese ***leading*** hardware venders and cloud venders
- Has now raised ***10s of millions of dollars***

What is Block Storage Used For

□ Virtual Machine

□ Database

□ Container

KVM
openstack™
Xen Server™
docker
MySQL®
MariaDB
kubernetes

Outlines

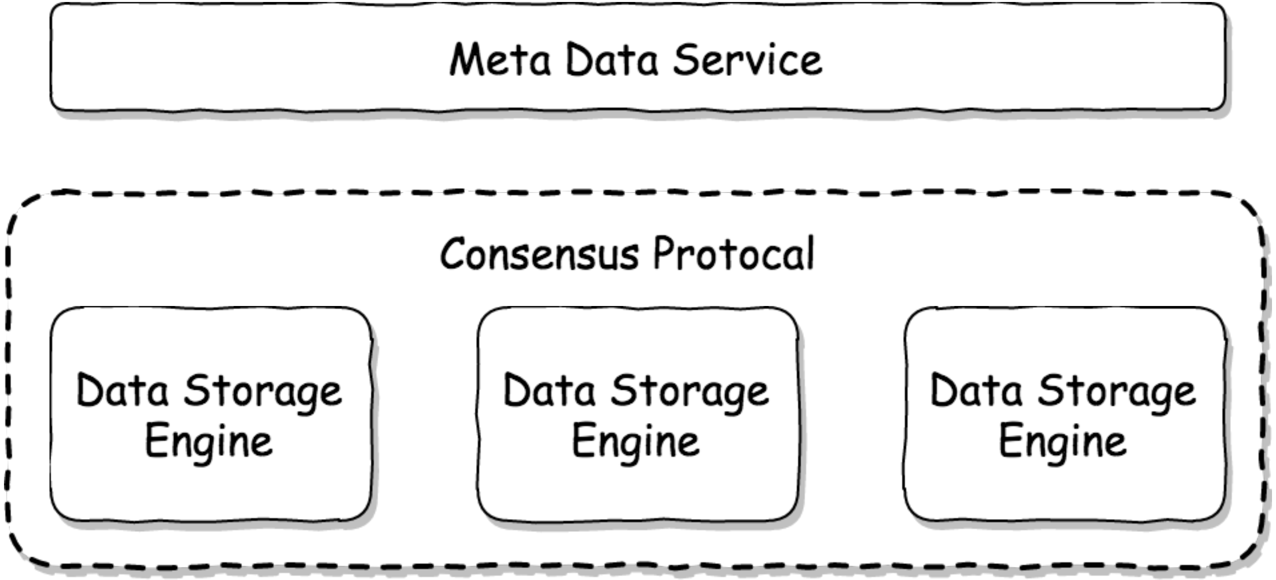
❑ How SmartX Build ZBS

❑ Roadmap of ZBS

Outlines

- ❑ How SmartX Build ZBS
- ❑ Roadmap of ZBS

How to Build a Distributed Storage



Meta Data Service

Consensus Protocal

Data Storage Engine

Data Storage Engine

Data Storage Engine

How to Build a Distributed Storage

- □ Meta data service
- □ Data storage engine
- □ Consensus protocol

How to Build a Distributed Storage

- ❑ **Meta data service**
- ❑ Data storage engine
- ❑ Consensus protocol

Meta Data Service Requirements

- Reliable
  - Data should be protected by replication
  - Failover
- High performance
  - Average latency per request should be less than 5ms
  - No necessary for high throughput
- Lightweight
  - Easy to operation

How to Build Meta Data Service

- MySQL, LevelDB/RocksDB ...
  - No data protection
  - No failover

- MongoDB/Cassandra
  - No ACID
  - Hard to operation

How to Build Meta Data Service

- Paxos/RAFT
  - Not available in 2013
- Zookeeper
  - Limited storage capacity
- DHT (Distributed Hash Table)
  - Loss control replica location
  - Hard for load balance

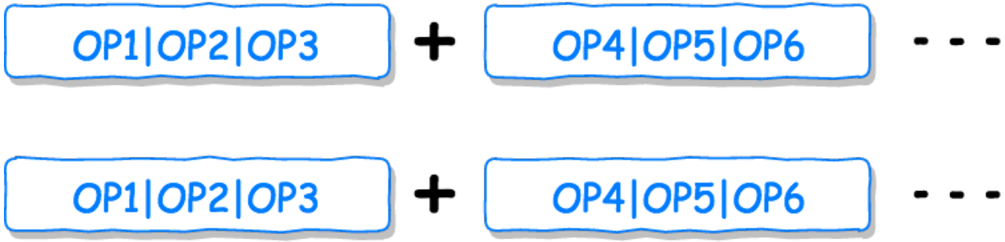
Meta Data Service Solution

❑ **Combined LevelDB and Zookeeper**

  ❑ Lightweight and stable

❑ **Log replication**

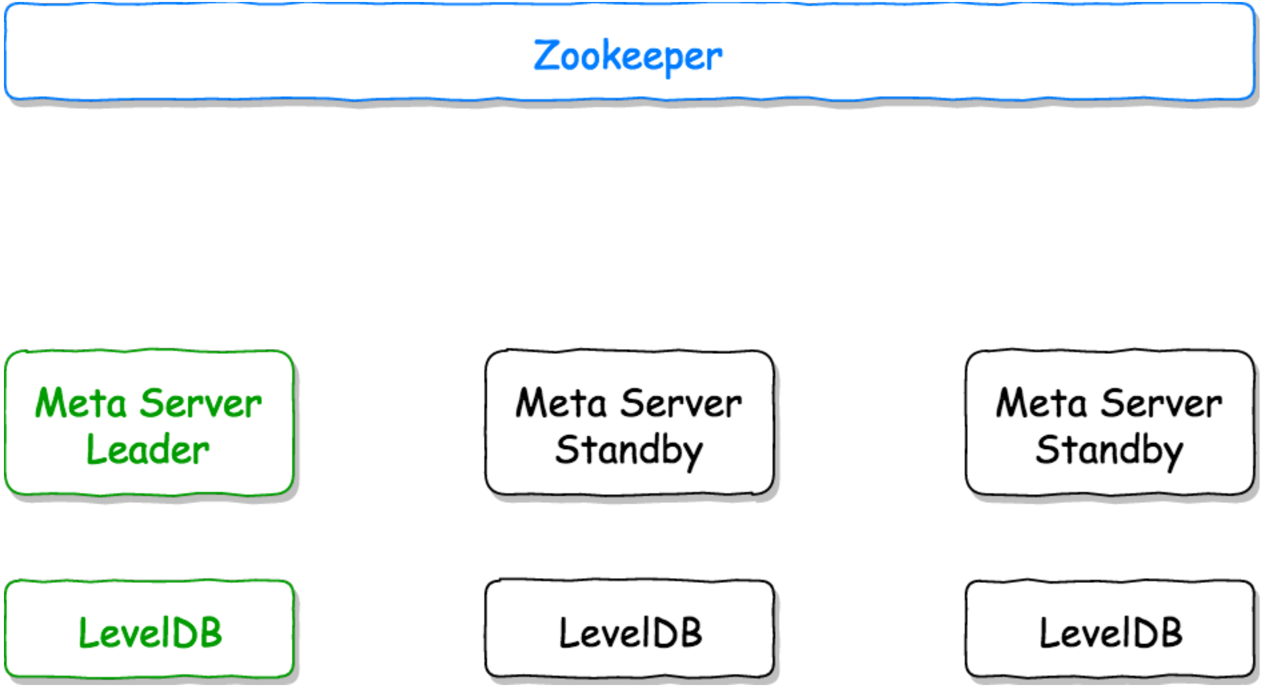
Meta Data =



OP1|OP2|OP3 + OP4|OP5|OP6 - - -

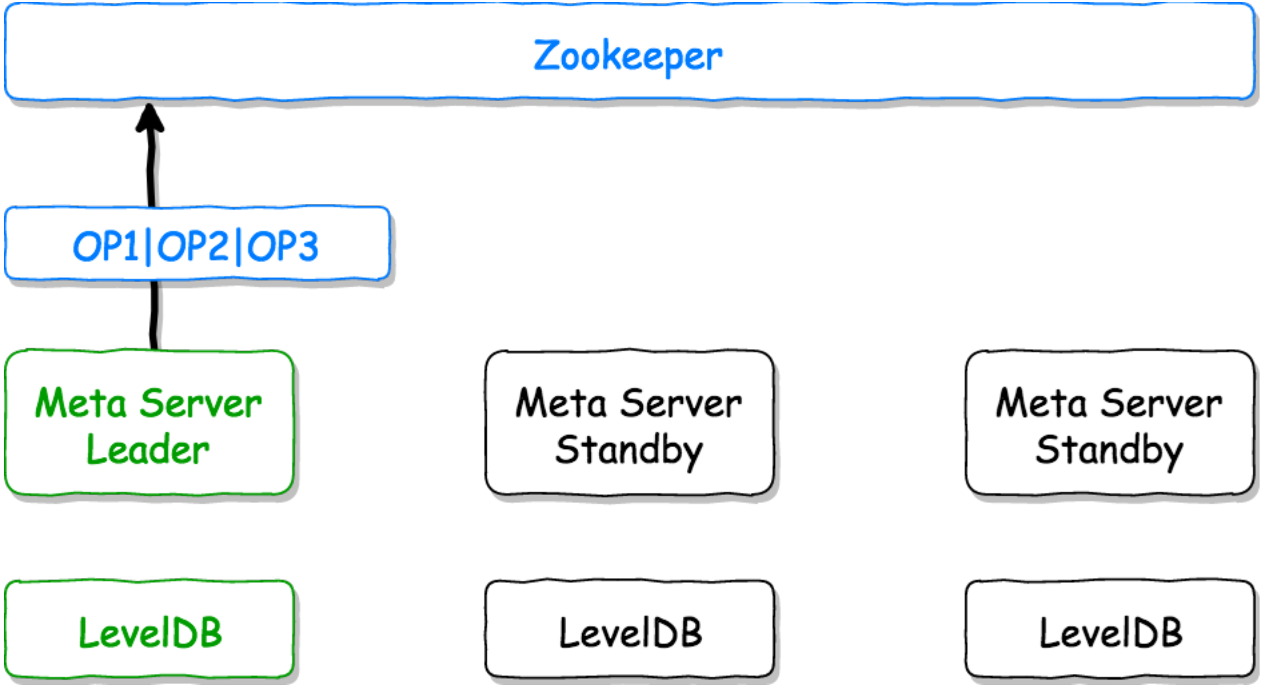
OP1|OP2|OP3 + OP4|OP5|OP6 - - -

Meta Server Cluster



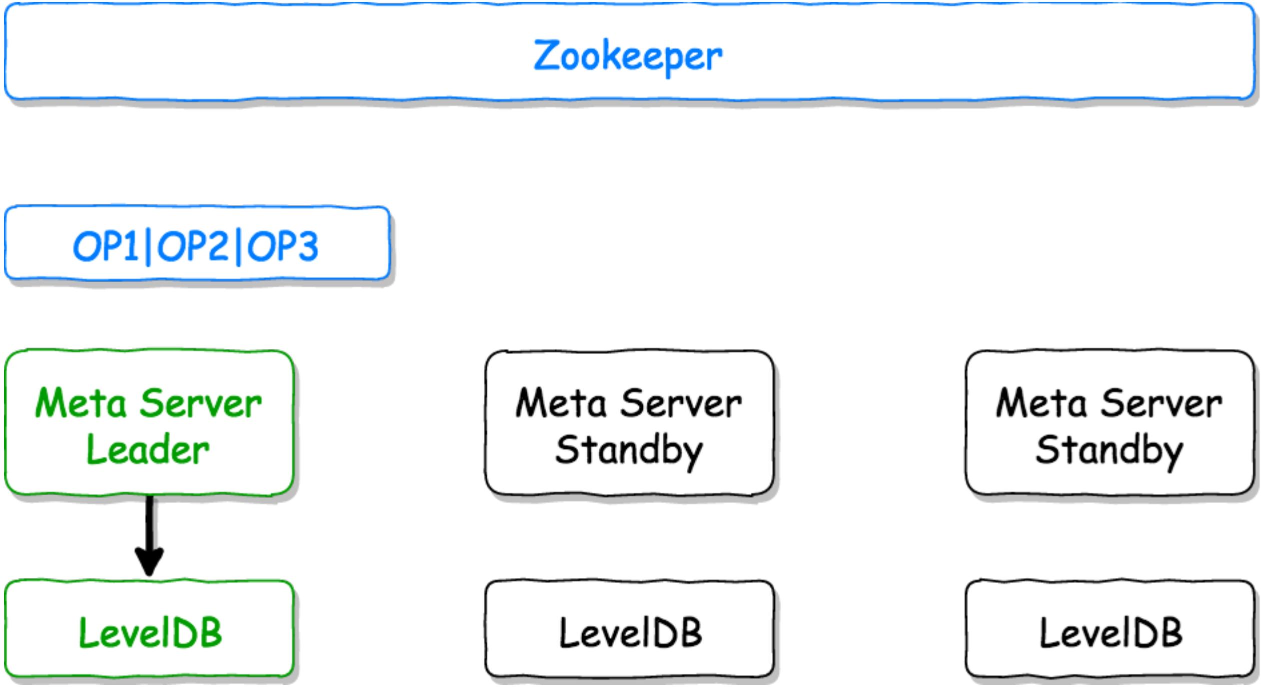
Elect leader through Zookeeper

Meta Server Cluster



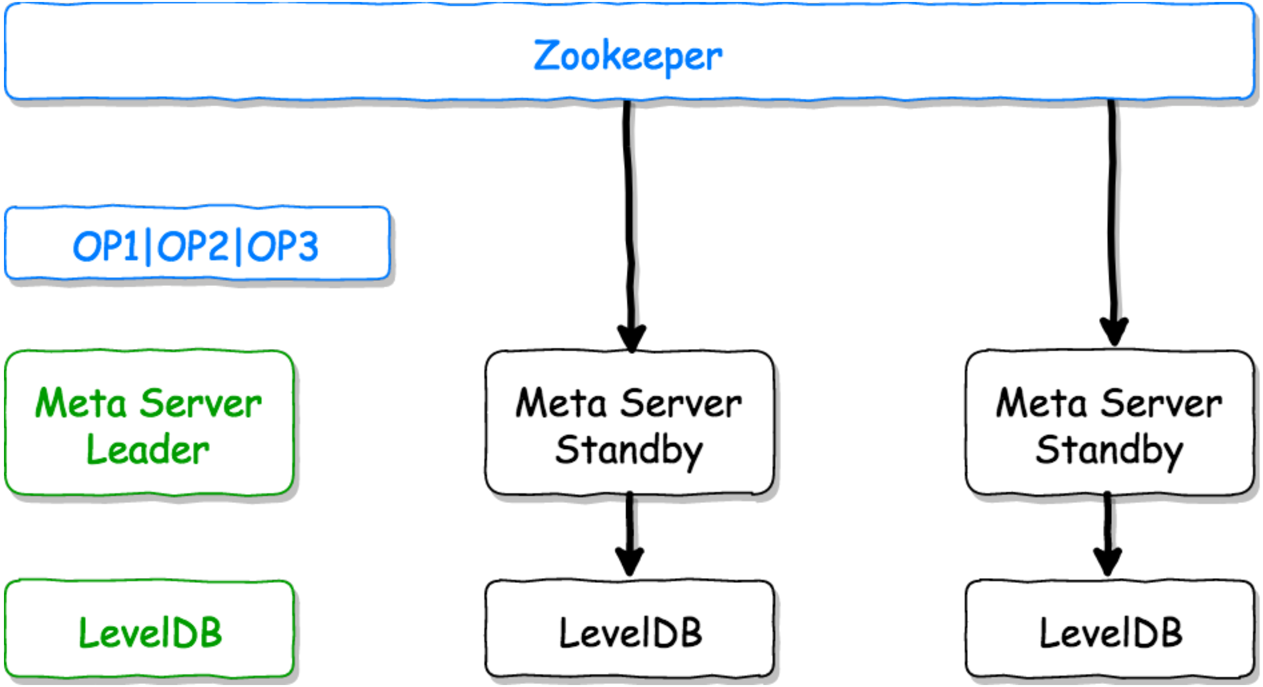
Submit operation log to Zookeeper

Meta Server Cluster



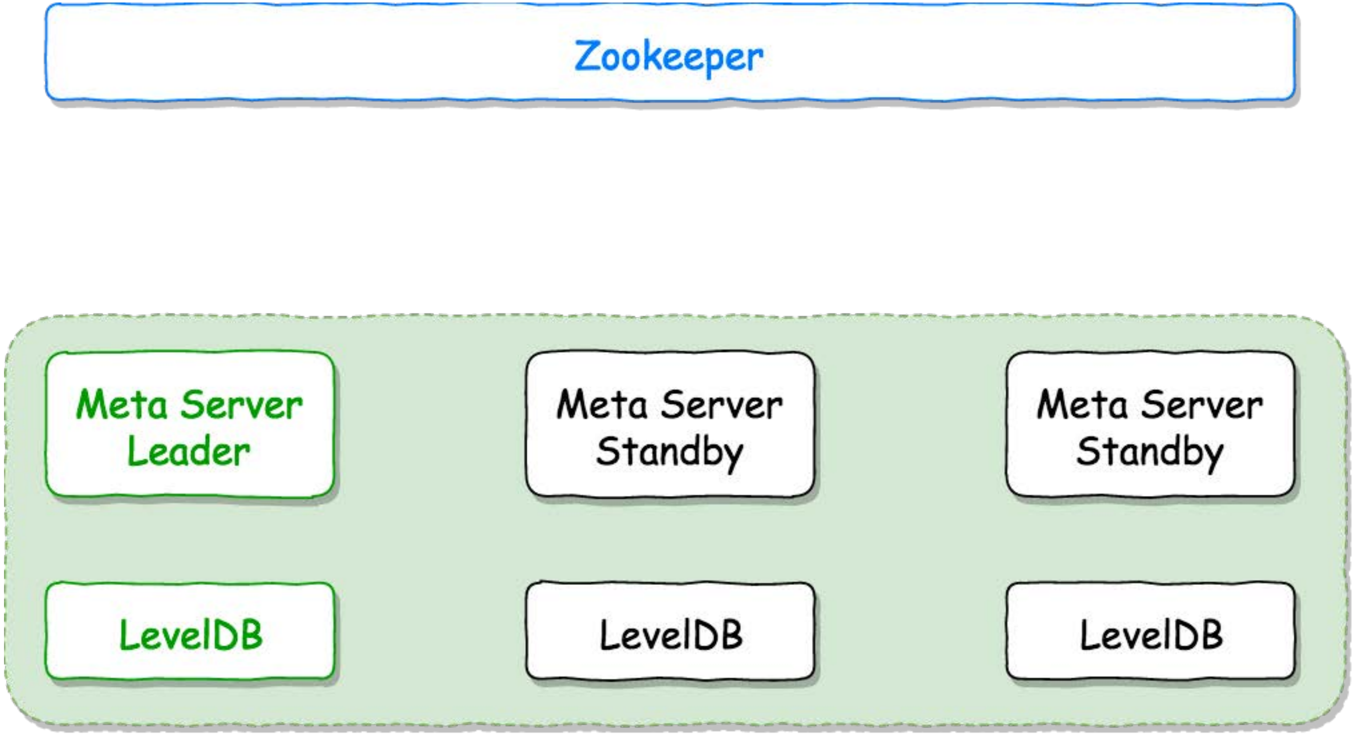
Apply changes to local DB

Meta Server Cluster



Learn logs from
Zookeeper and apply
to local DB

Meta Server Cluster



Operation logs is cleaned after learned by all meta servers

Meta data is always consistent

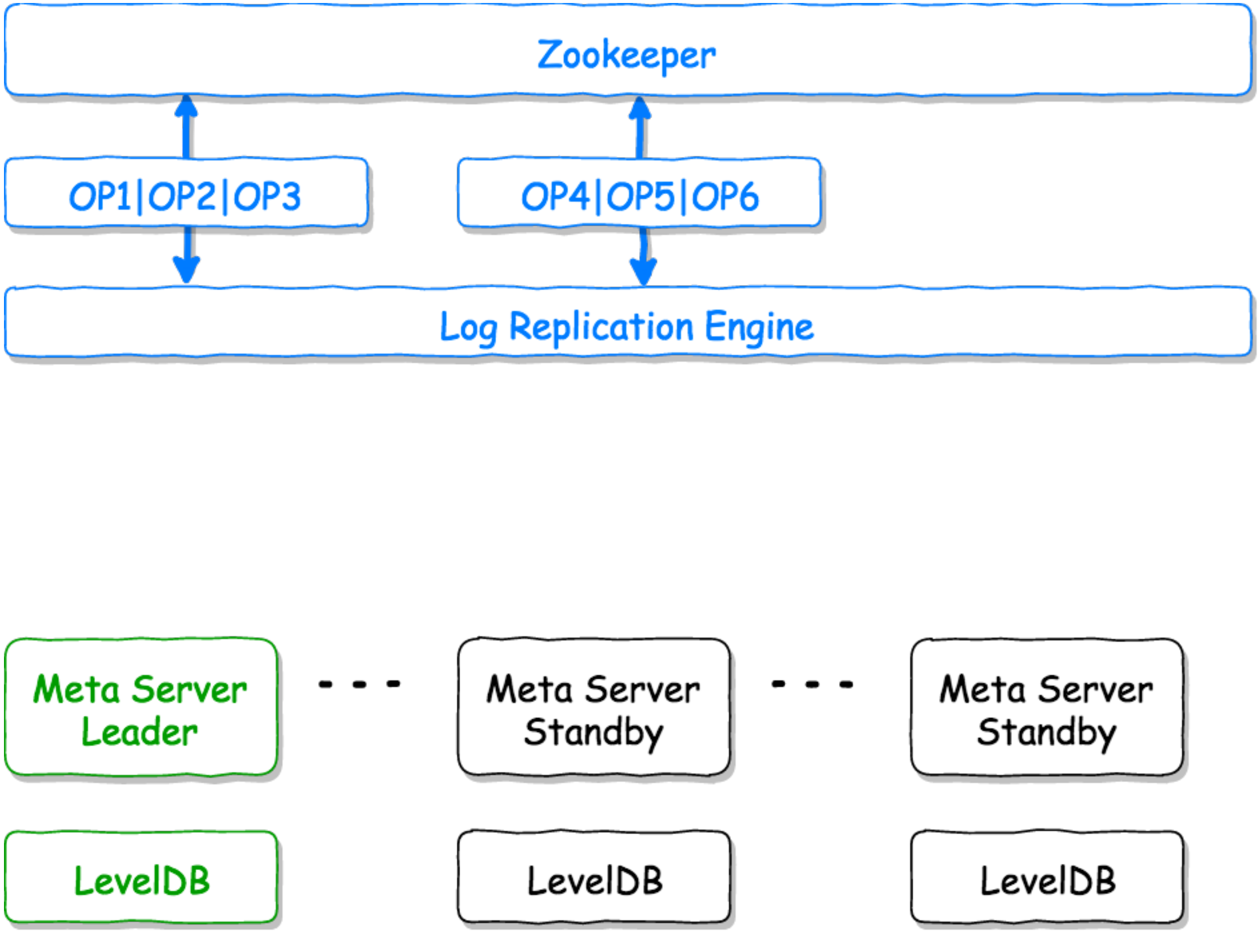
Meta Server Failover

- ❑ Elect new leader through Zookeeper
- ❑ Consume all logs from Zookeeper
- ❑ Provide meta data service

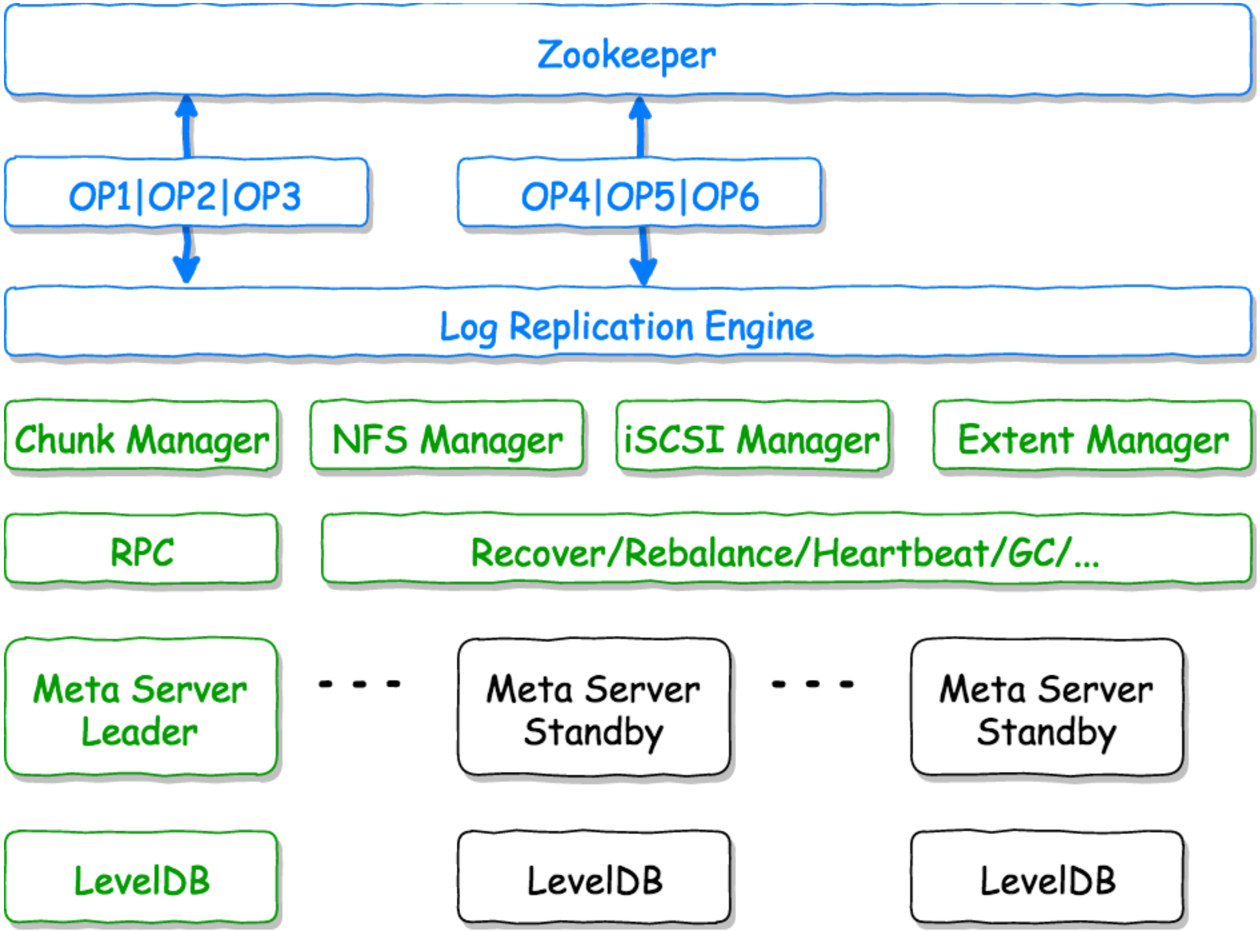
Meta Server Summary

- Easy to understand and implement
  - Zookeeper: leader election and log replication
  - LevelDB: meta data storage
- Fast enough for meta data service
- Failover
  - Failover time: leader election + log consumption
- Deployment
  - 3~5 Zookeeper and Meta Server in one cluster

Meta Server Summary



Meta Server Summary



How to Build a Distributed Storage

❑ Meta data service

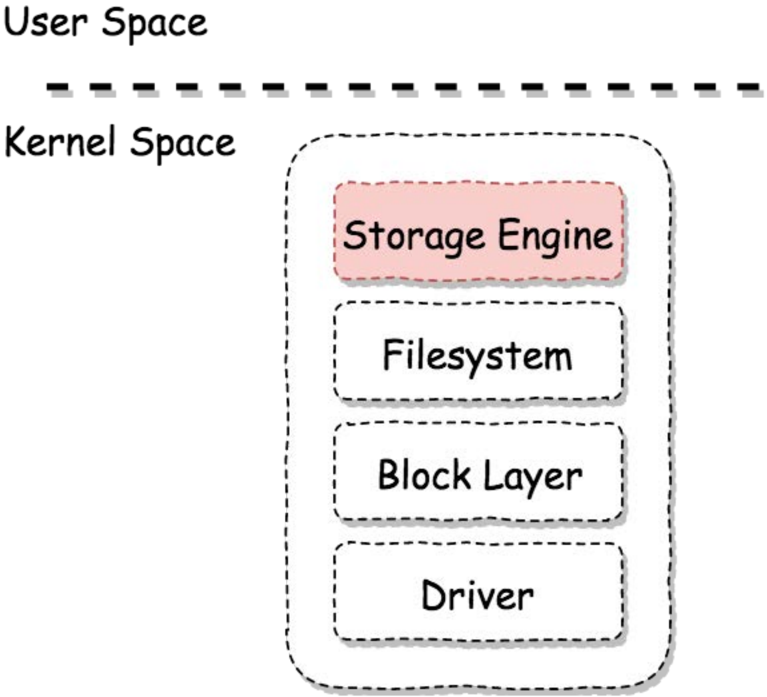
❑ Data storage engine

❑ Consensus protocol

Data Storage Engine Requirements

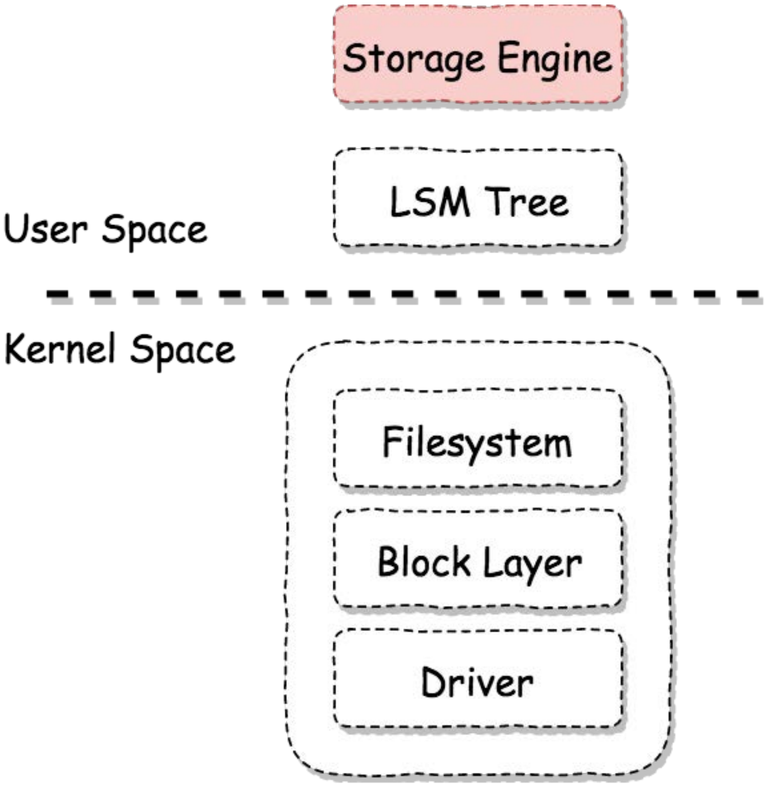
- Reliable
- Performance
- Efficiency
  - CPU, Memory, IO Bandwidth ...
  - Space
- Easy to debug and upgrade

Kernel Space



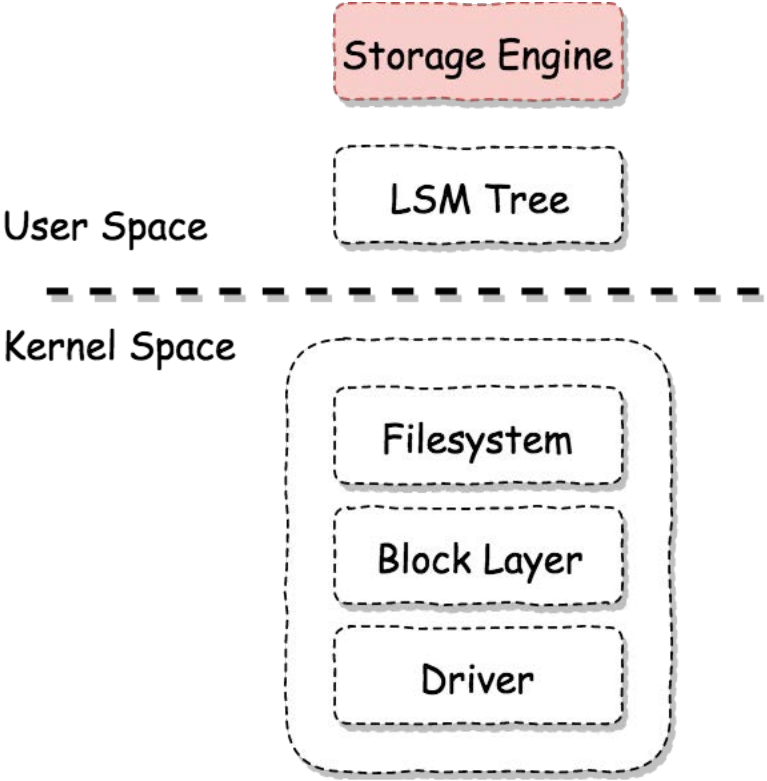
- ❑ **Kernel space**
  - ❑ Hard for debugging and management
  - ❑ Upgrading is costly: host restart
  - ❑ Large failure domain: kernel panic!
  - ❑ No context switch

Kernel Space vs. User Space



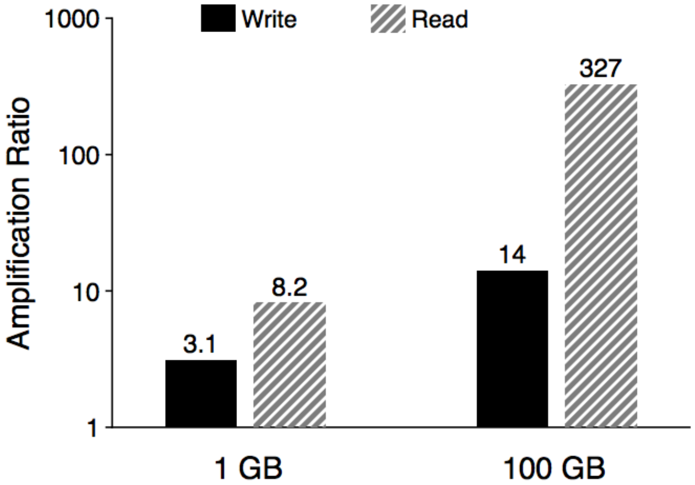
- ❑ **User space Storage Engine**
  - ❑ Flexible and easy for debugging
  - ❑ Upgrading is cheap: process restart
  - ❑ Isolated failure domain: process crash
  - ❑ Context switch is costly

LSM Tree

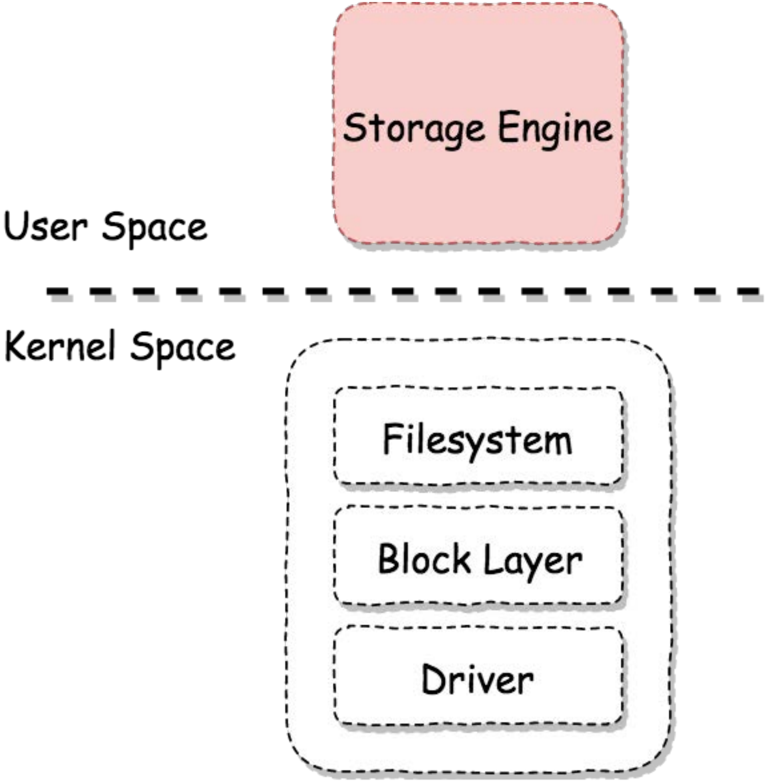


❑ LSM Tree

- ❑ Easy to implement
- ❑ Optimized for small object
- ❑ Read/Write amplification

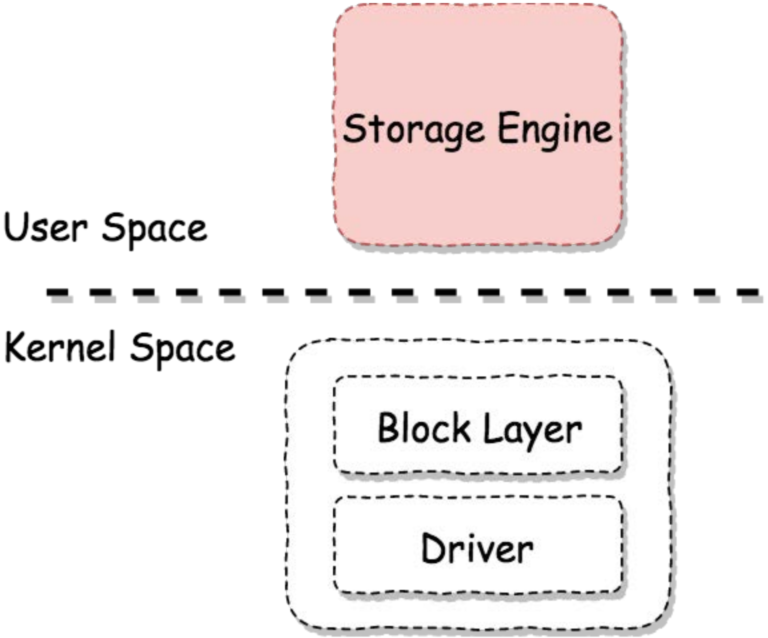


Kernel Filesystems

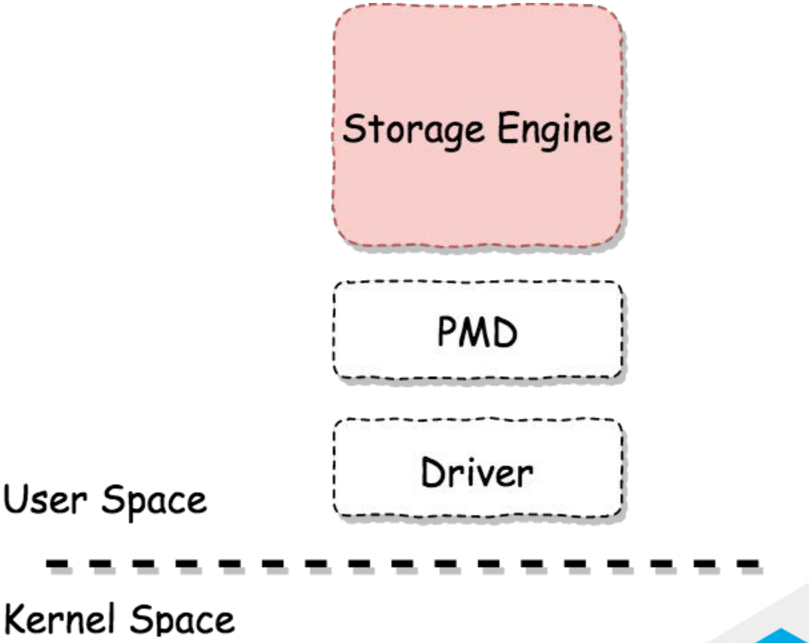


- ❑ **Kernel filesystems**
  - ❑ Too much features not necessary for building storage engine: performance overhead
  - ❑ Filesystem is designed for single disk
  - ❑ Bad support for asynchronous IO
  - ❑ Write amplification of duplicated journaling

Userspace Storage Engine

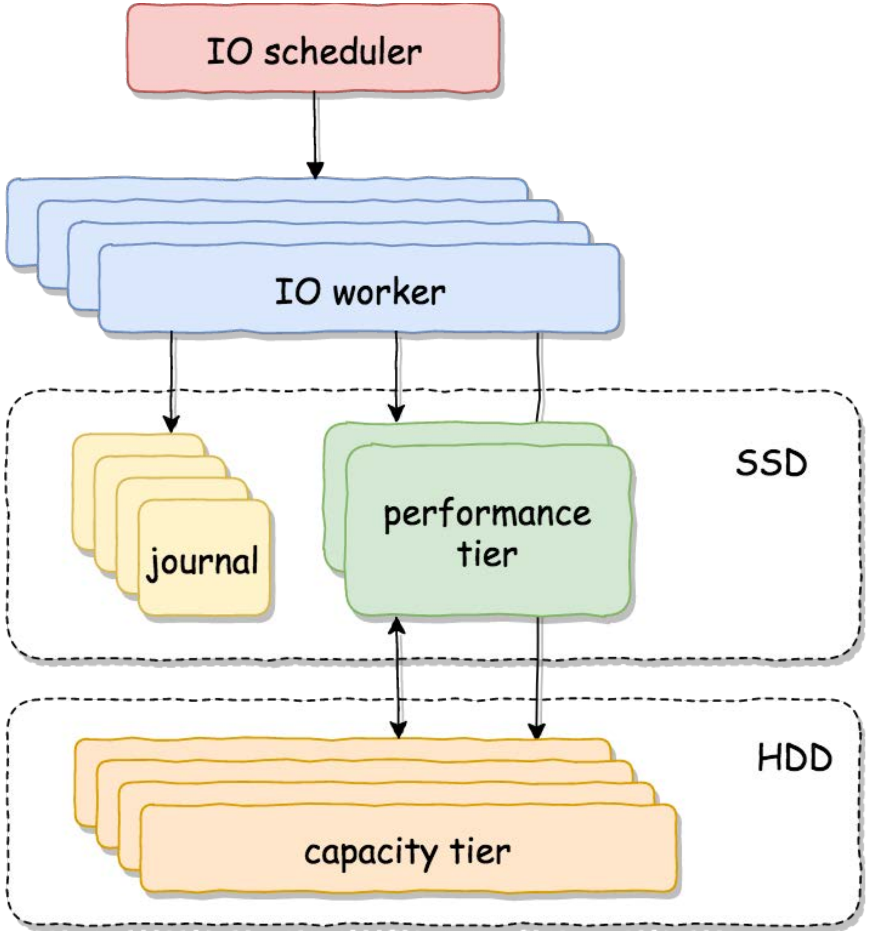


Context switch



No context switch

Userspace Storage Engine



- ❑ **IO Scheduler**
  - ❑ Construct IO transaction and submit to specified IO worker
- ❑ **IO Worker**
  - ❑ Execute IO transaction
  - ❑ Sequentialize overlapping IO requests
- ❑ **Journal**
  - ❑ Providing transaction implementation

How to Build a Distributed Storage

- ❑ Meta data service
- ❑ Data storage engine
- ❑ Consensus protocol

Consensus Protocol Requirements

- Strong consistency
- Performance
  - Fast enough for flash devices

How to Build Consistent Protocol

❑ Basic-Paxos

❑ Multi-Paxos

❑ Raft

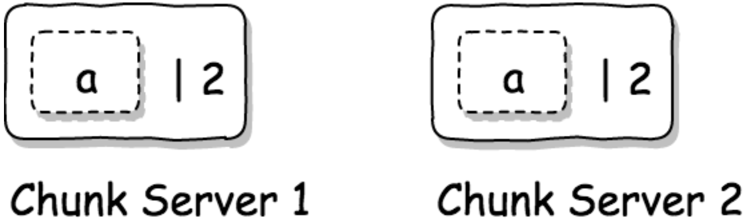
❑ ...

Generation-Based Protocol

- Generation is the version of data
  - Write increase data generation by 1
  - Generation is persist along with data
  - Request is valid only when generation match
- 1 leader can access data
  - Leader is chosen by Meta Server
- 2 phases
  - Prepare: only necessary for first IO
  - Commit: update both data and generation atomically

Prepare 1/3

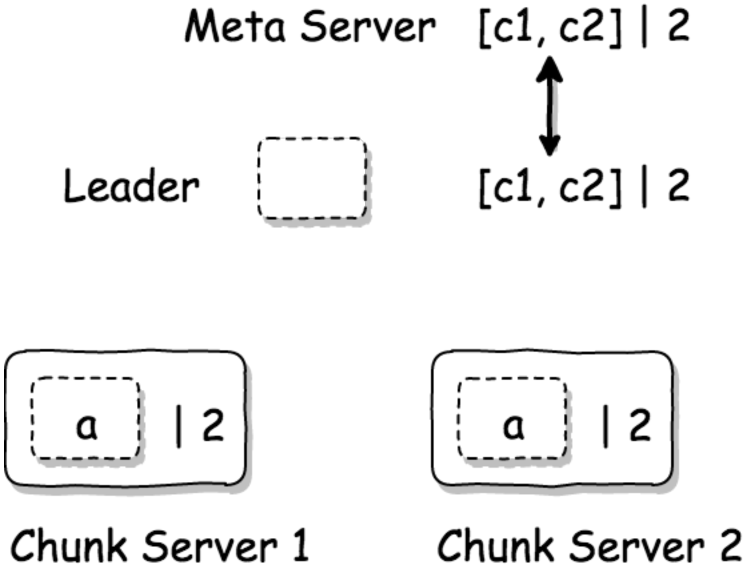
Meta Server [c1, c2] | 2



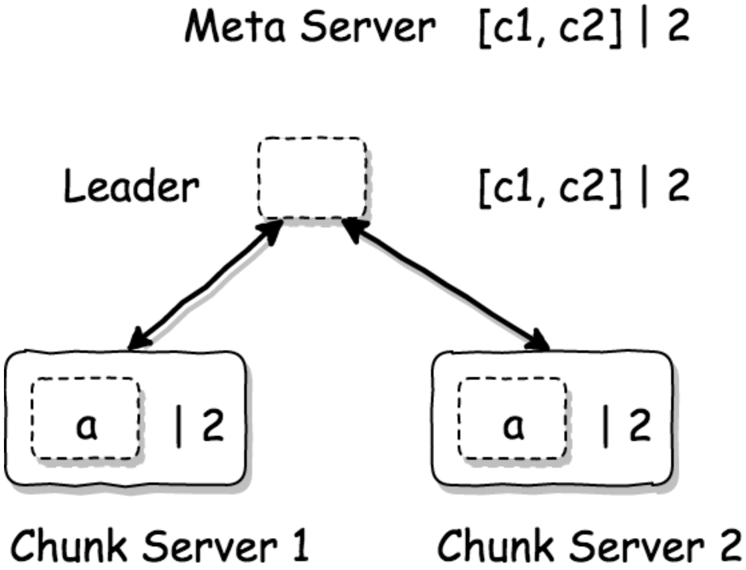
Chunk Server 1

Chunk Server 2

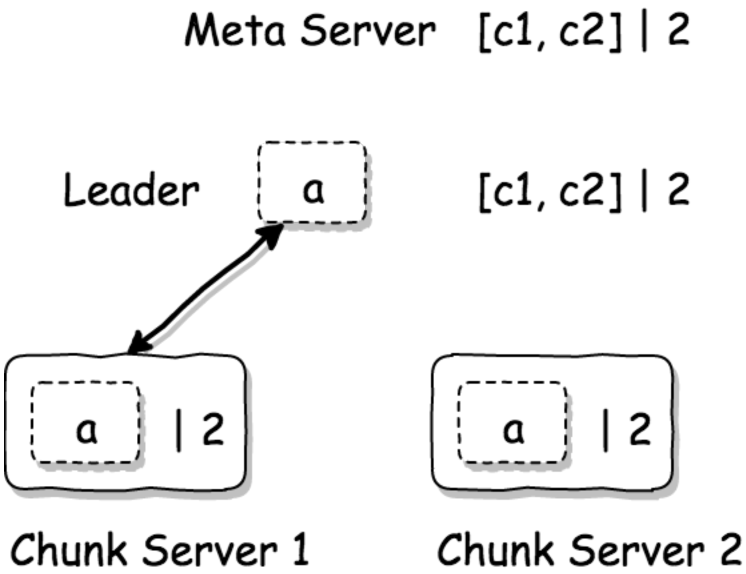
Prepare 2/3


Meta Server [c1, c2] | 2
Leader
[c1, c2] | 2
a | 2
a | 2
Chunk Server 1
Chunk Server 2

Prepare 3/3


Meta Server [c1, c2] | 2
Leader
[c1, c2] | 2
a | 2
a | 2
Chunk Server 1
Chunk Server 2

Read



Meta Server [c1, c2] | 2

Leader a [c1, c2] | 2

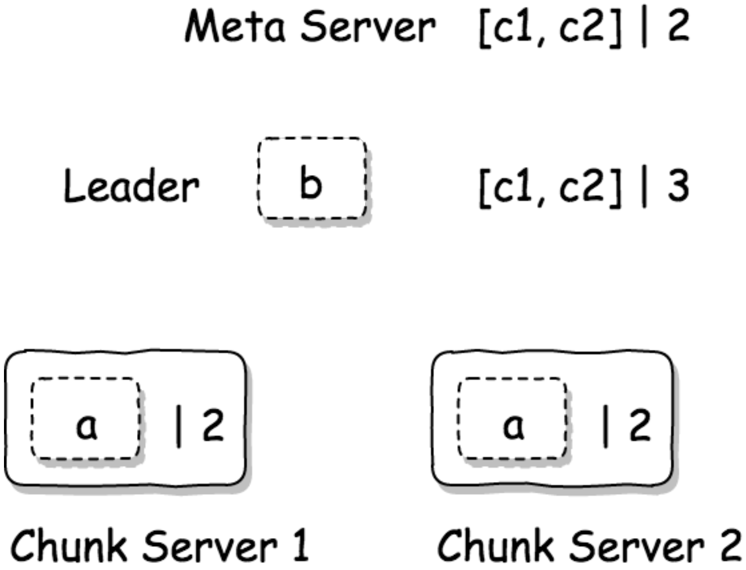
a | 2

a | 2

Chunk Server 1

Chunk Server 2

Write 1/2



Meta Server [c1, c2] | 2

Leader b [c1, c2] | 3

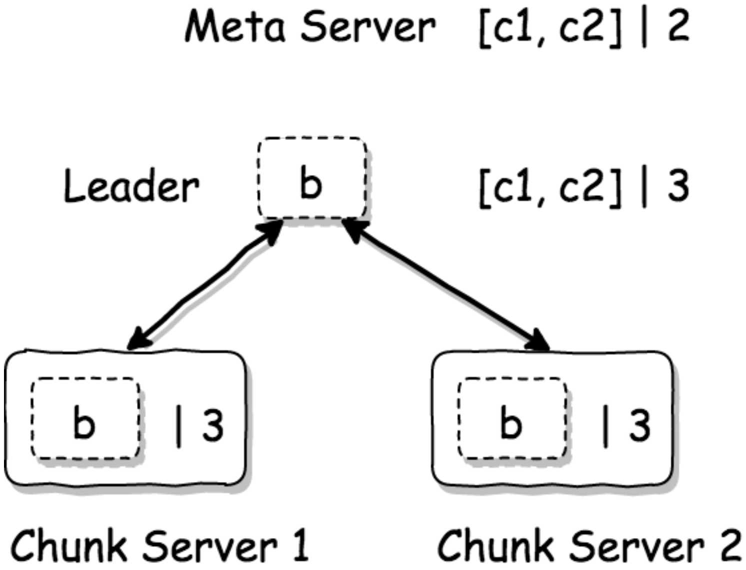
a | 2

a | 2

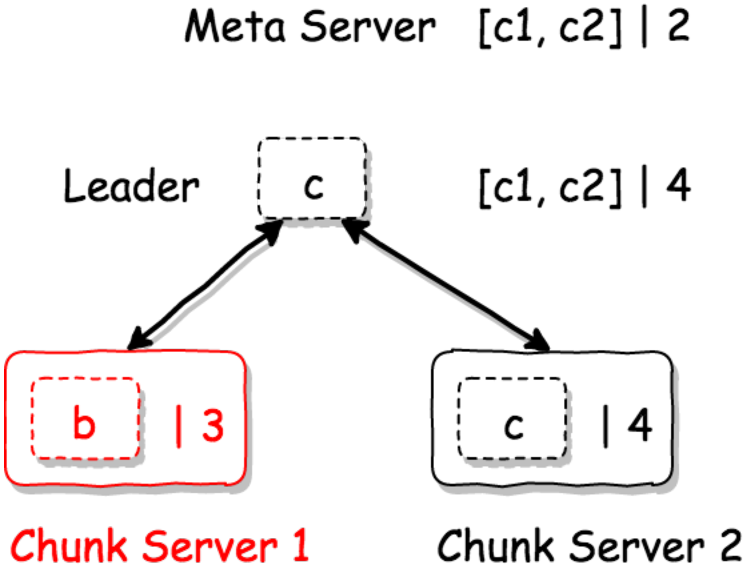
Chunk Server 1

Chunk Server 2

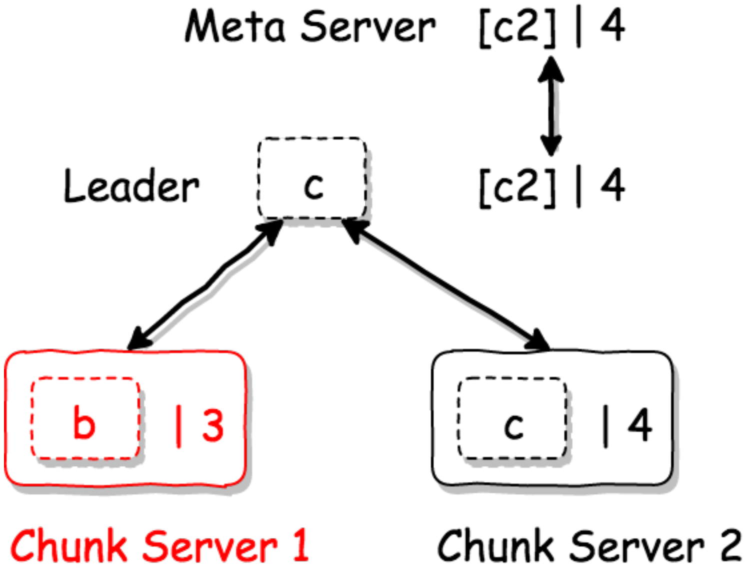
Write 2/2


Meta Server [c1, c2] | 2
Leader
b
[c1, c2] | 3
b
| 3
b
| 3
Chunk Server 1
Chunk Server 2

Membership Change 1/2



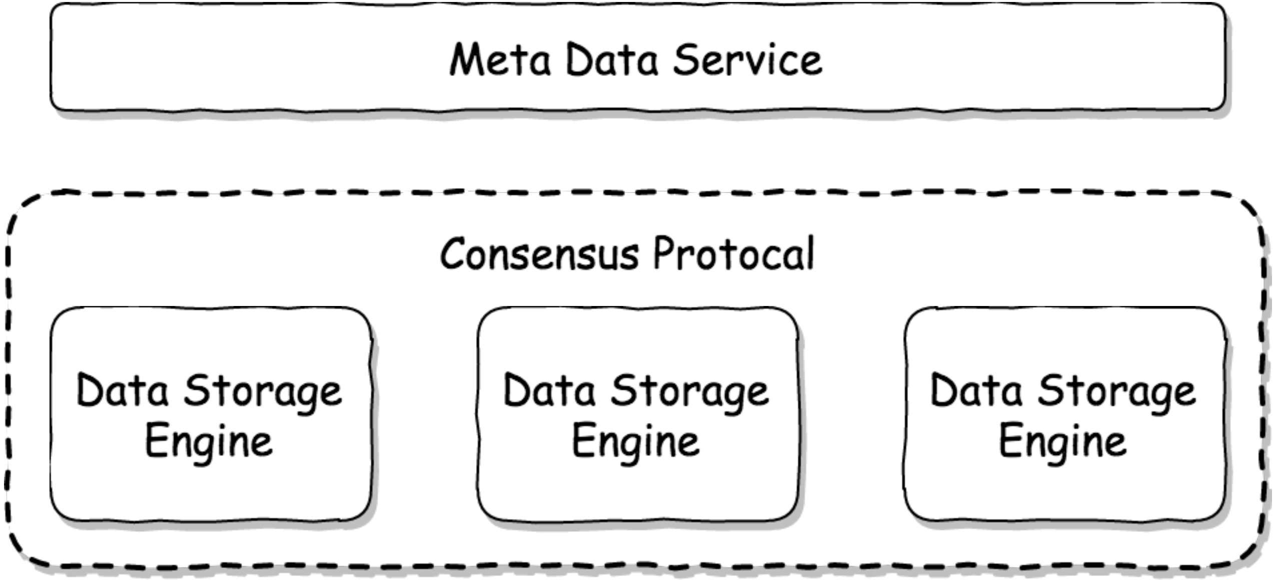
Membership Change 2/2



Consistency Protocol Summary

- Easy to understand and implement
  - Like Multi-Paxos and Raft: single leader (proposer)
  - Leader is chosen by leader of Meta Server leader
  - Maintain leadership by updating lease
- About Generation
  - Valid membership and least valid generation is persist to Meta Data Service
  - Transaction of updating data and generation is ensured by Storage Engine
- Performance
  - Less RTT than Basic Paxos

How to Build a Distributed Storage



Meta Data Service

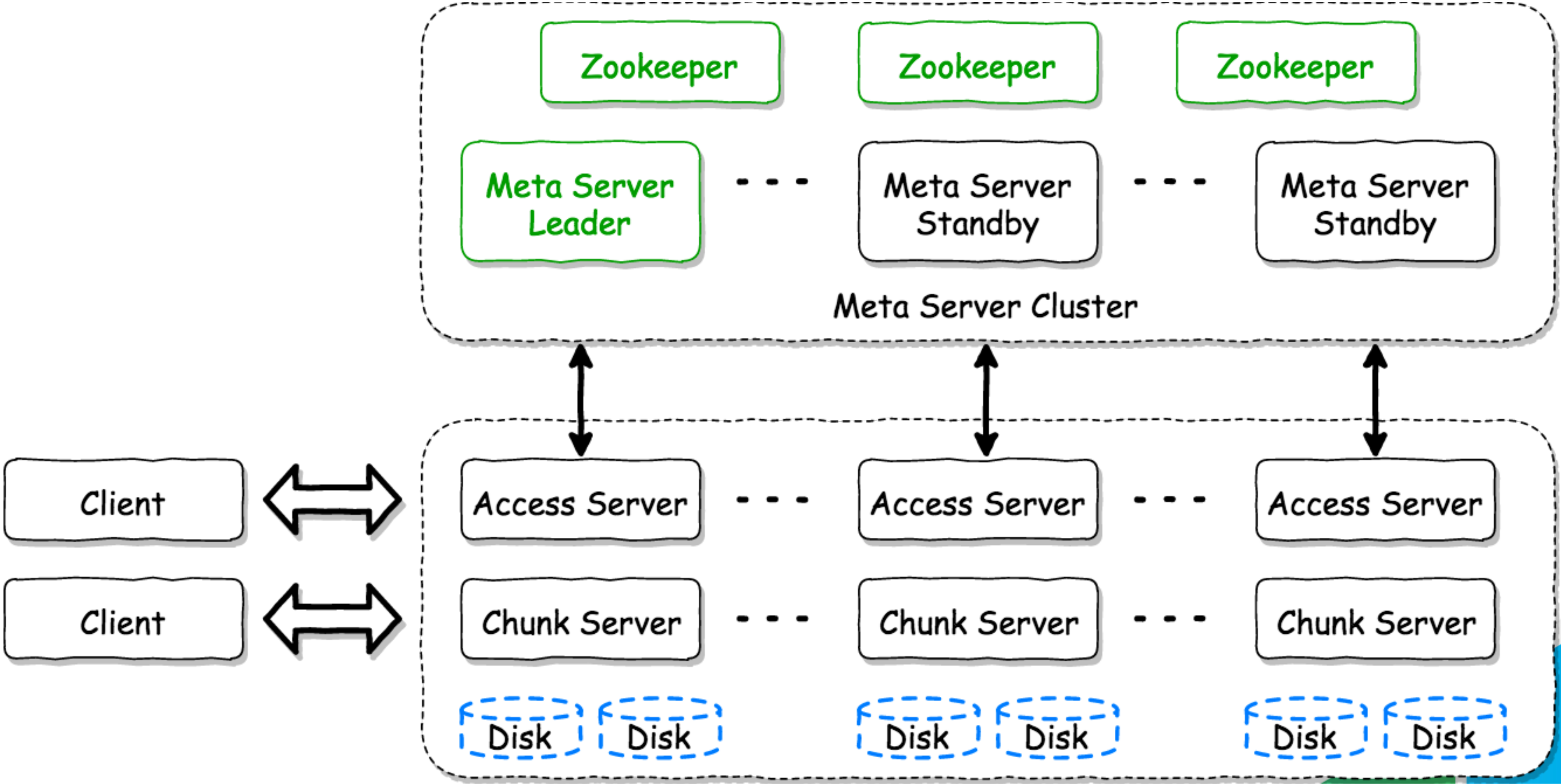
Consensus Protocal

Data Storage Engine

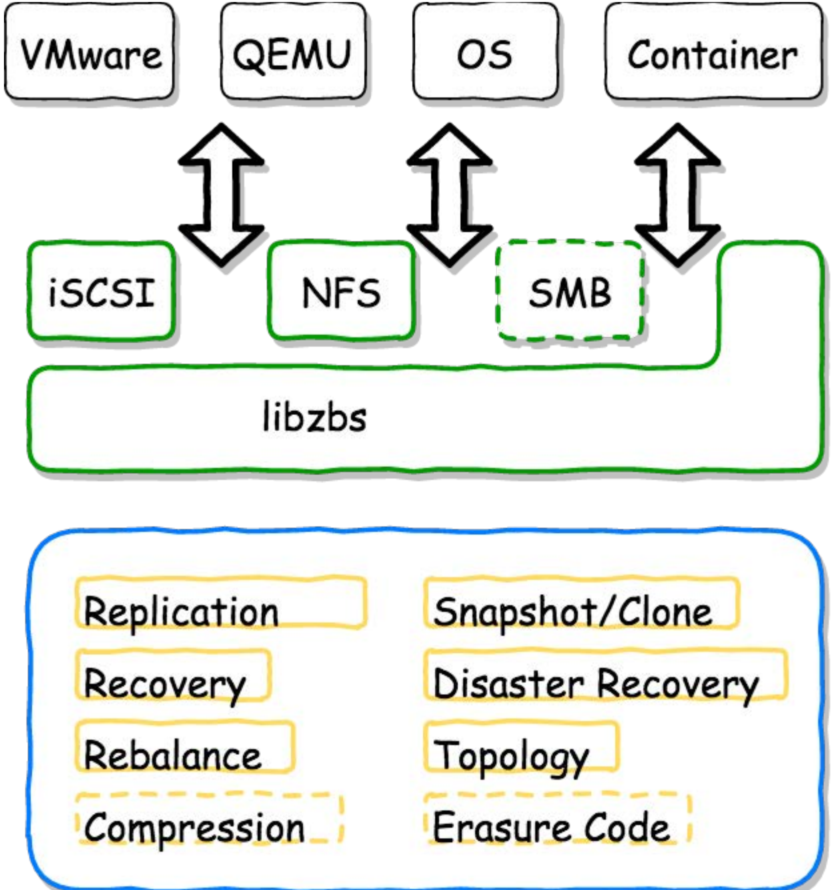
Data Storage Engine

Data Storage Engine

Architecture of ZBS

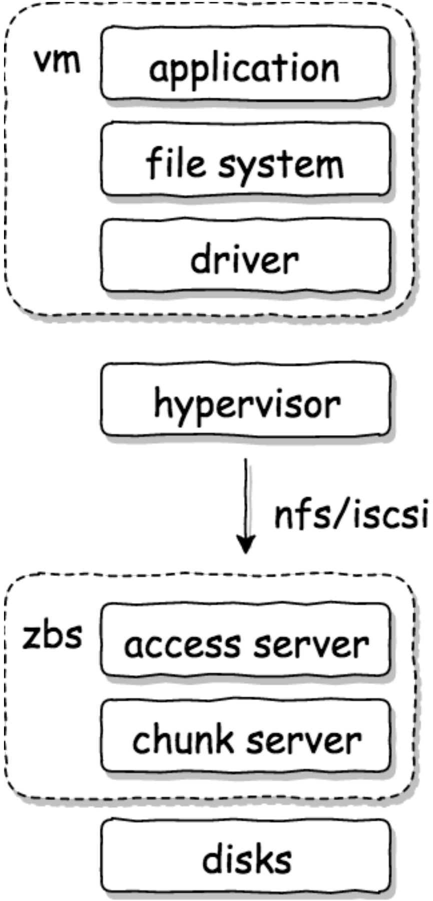


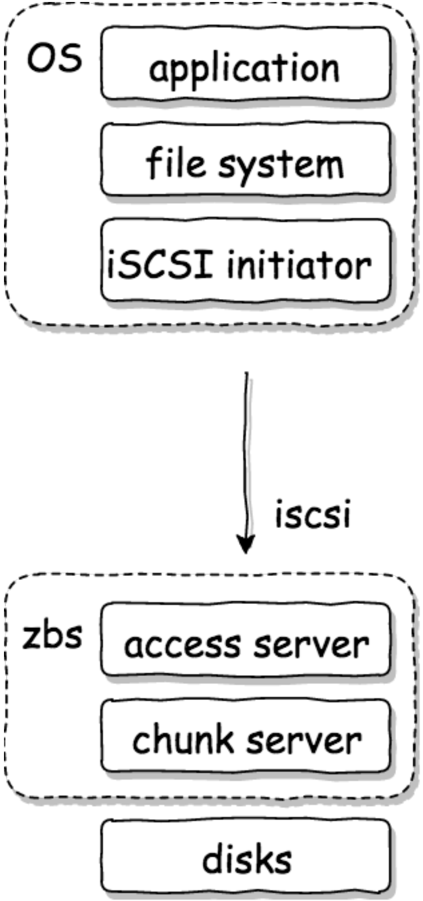
ZBS Interface



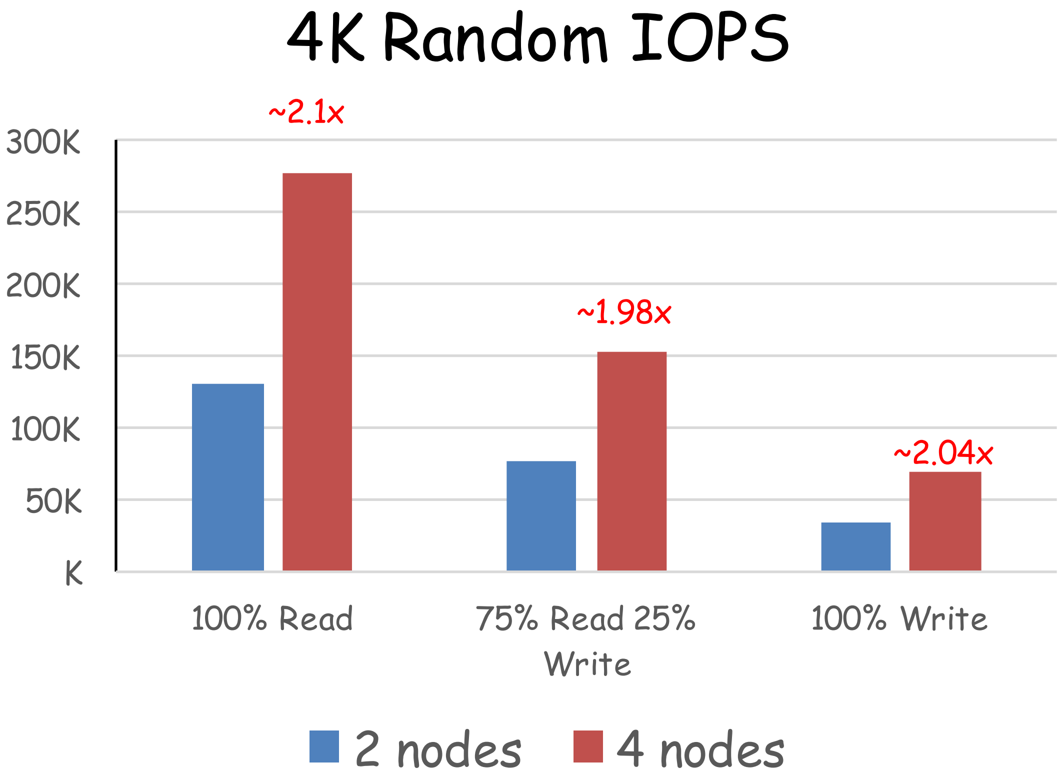
- ❑ **libzbs (c library)**
  - ❑ Optimized for performance
- ❑ **iSCSI/NFS**
  - ❑ Based on libzbs
  - ❑ Friendly for operating system, hypervisor...

IO Path


vm
application
file system
driver
hypervisor
nfs/iscsi
zbs
access server
chunk server
disks


OS
application
file system
iSCSI initiator
iscsi
zbs
access server
chunk server
disks

Scalability Evaluation



- ❑ **Node Configuration**
  - ❑ SATA SSD * 2
  - ❑ SATA HDD * 4
- ❑ **Replica factor: 2**

Outlines

smartx
MAKE IT SIMPLE

- ❑ Deep Dive into SmartX ZBS
- ❑ Roadmap of ZBS

Roadmap of ZBS

- ❑ **More data protection**
  - ❑ Erasure code
  - ❑ Cloud backup
  - ❑ ...

- ❑ **More efficiency**
  - ❑ Compression
  - ❑ Deduplication
  - ❑ ...

Roadmap of ZBS

- ❑ **More intelligence**
  - ❑ Failure prediction
  - ❑ Cache algorithm
  - ❑ ...
- ❑ **More performance**
  - ❑ SPDK
  - ❑ RDMA
  - ❑ ...