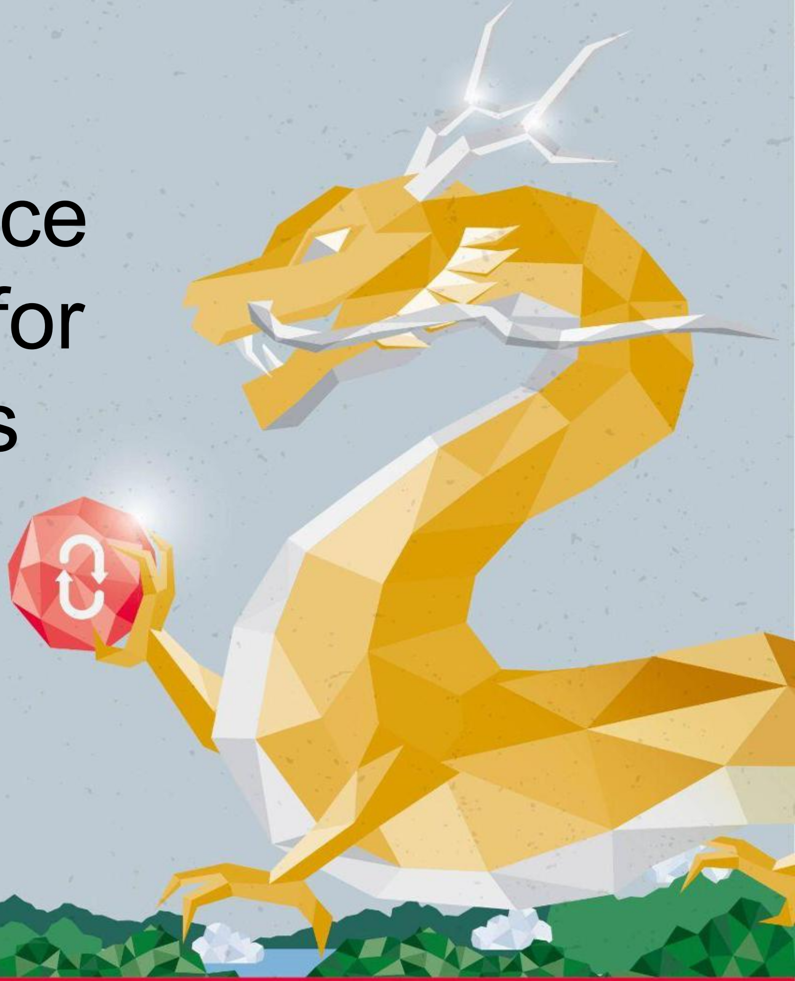
CURVE: a High-performance Software-Defined storage for Cloud Native applications

Curve

Xiang Dong
Architect, Netease Hangzhou Research Institute



Agenda

√ Why develop storage

Design objectives

Achievements in CURVE

Key designs used by CURVE

CURVE Roadmap

CURVE Community

You can take away

Why develop

Storage requirements

- Hardware requirements
  - more CPU cores, faster Network, nvme storage
- Problems for stateful apps
  - storage capacity expansion
  - capacity imbalance
  - apps bundled with data locations
- Requirements for elastic block storage
- Requirements for file system

open-source storage

- Requirements
  - Cloud Native
  - Easy operation and maintenance
  - High performance
- CEPH
  - Complex and Large amount of codes
  - system maintenance are difficult in corner cases
  - Performance requirements cannot be met

Agenda

Why develop storage

√ Design objectives

Achievements in CURVE

Key designs used by CURVE

CURVE Roadmap

CURVE Community

You can take away

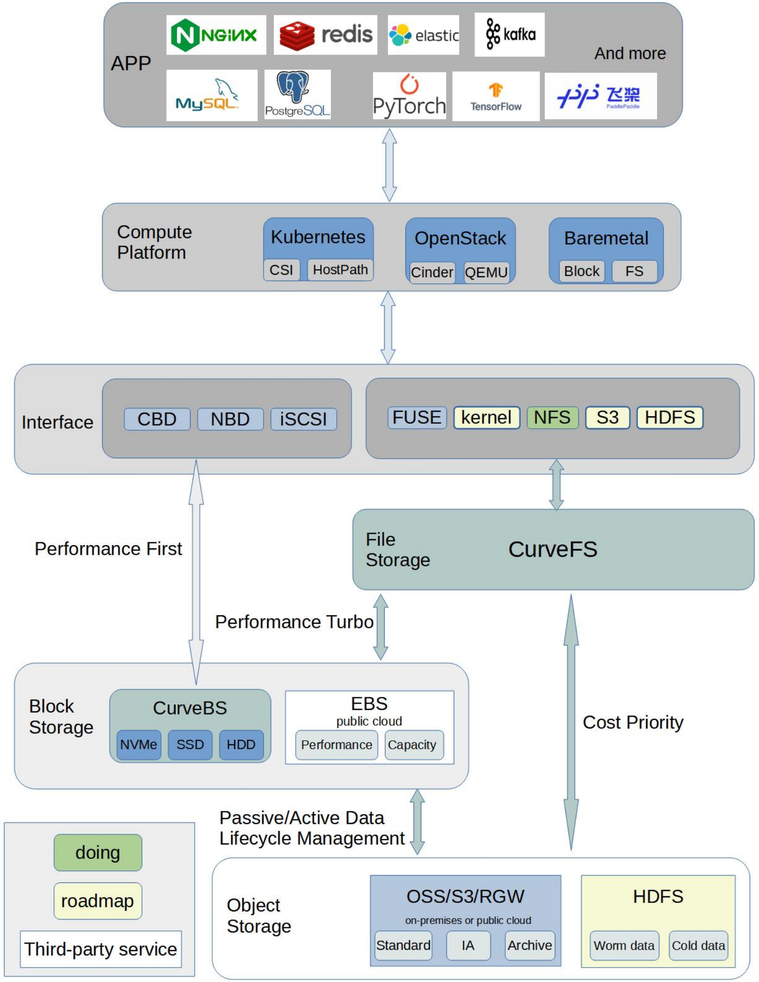
Design objectives

App scenario & requires

- Database scenario
- Elastic block Storage for KVM / Kubernetes / iSCSI

Design objectives

- Thin provisioning storage pools
- High performance
- Easy maintenance
- Cloud Native



APP
NGINX
redis
elastic
kafka
And more
MySQL
PostgreSQL
PyTorch
TensorFlow
飞桨 PaddlePaddle

Compute Platform
Kubernetes
CSI
HostPath
OpenStack
Cinder
QEMU
Baremetal
Block
FS

Interface
CBD
NBD
iSCSI
FUSE
kernel
NFS
S3
HDFS

Performance First

File Storage
CurveFS

Performance Turbo

Block Storage
CurveBS
NVMe
SSD
HDD
EBS
public cloud
Performance
Capacity

Cost Priority

doing
roadmap
Third-party service

Passive/Active Data Lifecycle Management

Object Storage
OSS/S3/RGW
on-premises or public cloud
Standard
IA
Archive
HDFS
Worn data
Cold data

Agenda

Why develop storage

Design objectives

√ Achievements in CURVE

Key designs used by CURVE

CURVE Roadmap

CURVE Community

You can take away

Performance in database scenario

Solution architecture

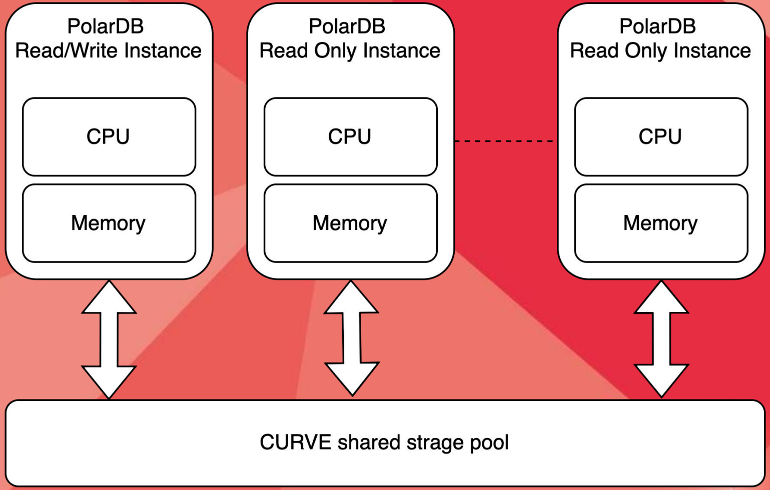
- CURVE provides underlying distributed shared storage services
- Polardb for PostgreSQL provides database services

Solution features

- seperation between storage and computation
- The call method is the same as that of the standalone database

Performance test vs CEPH

- BenchmarkSQL transactions per minute increased by 39%
- In PGBench tests, latency decreased by 21% and TPS improved by 26%


PolarDB Read/Write Instance
CPU
Memory
PolarDB Read Only Instance
CPU
Memory
PolarDB Read Only Instance
CPU
Memory
CURVE shared strage pool

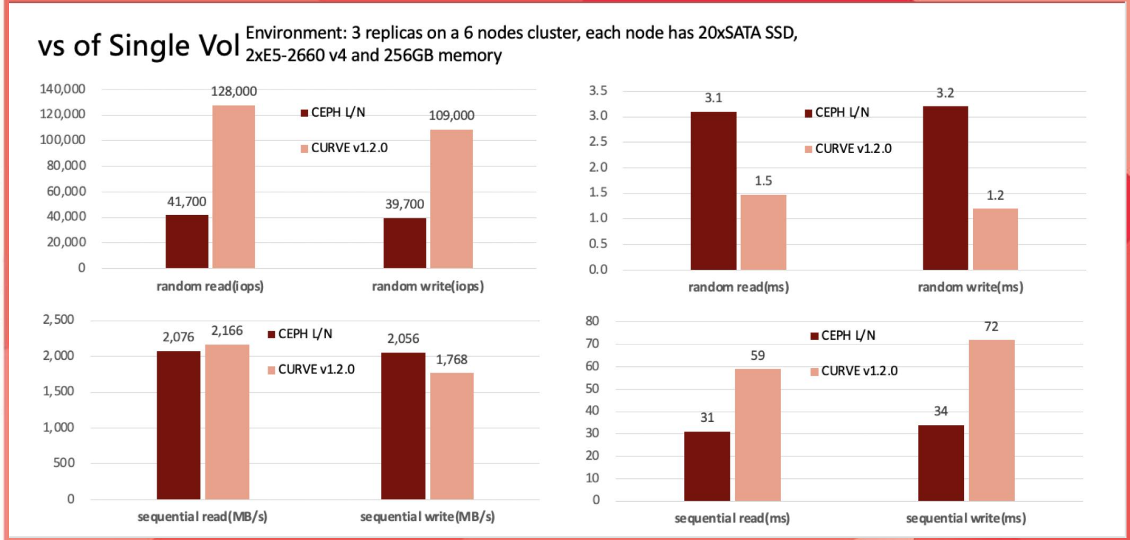
Performance compare with CEPH

The test environment

- A cluster consists of six nodes. Each node consists of 20 x SSDS, 256GB memory, and 2 x e5-2660 cpus

Performance using RDMA

- Compared with TCP, I/O throughput increases by 20%



vs of Single Vol Environment: 3 replicas on a 6 nodes cluster, each node has 20xSATA SSD, 2xE5-2660 v4 and 256GB memory

Availability testing in corner cases

The impact of different faults on I/O jitter delay

- Disk failure
- Server failure
- Server fake death
- Slow response

Faults case	curve i/o 抖动
Disk failure	4s
Server failure	4s
Server fake death	4s
Slow response	1s frequently

Data availability analysis

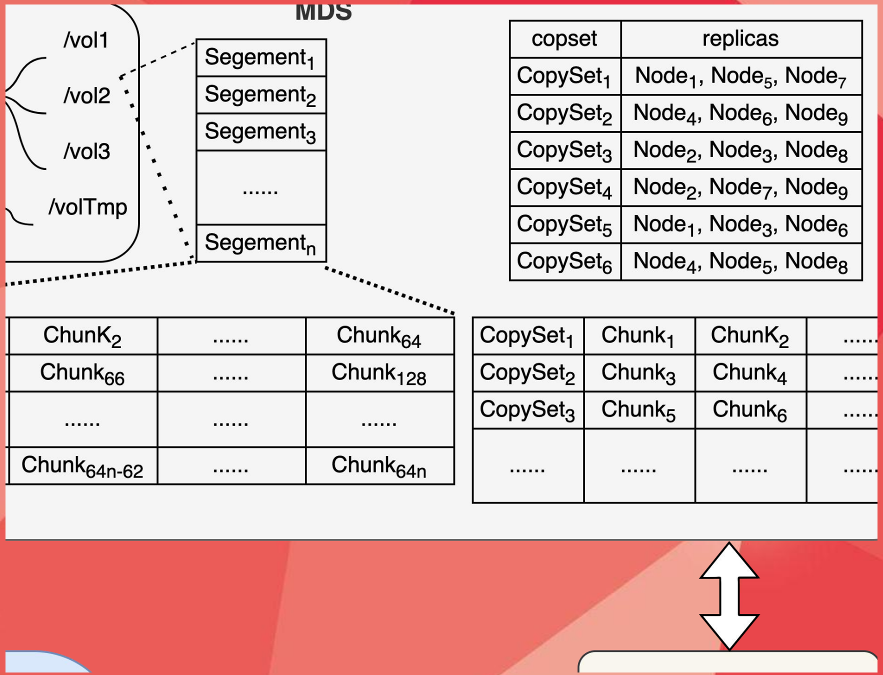
Copyset allocation algorithm

cluster with 1200 disks (each have MTBF 1.2 million hours)

3 replicas

Restore data on a disk within 5 minutes

Data availability of 6 nines can be achieved



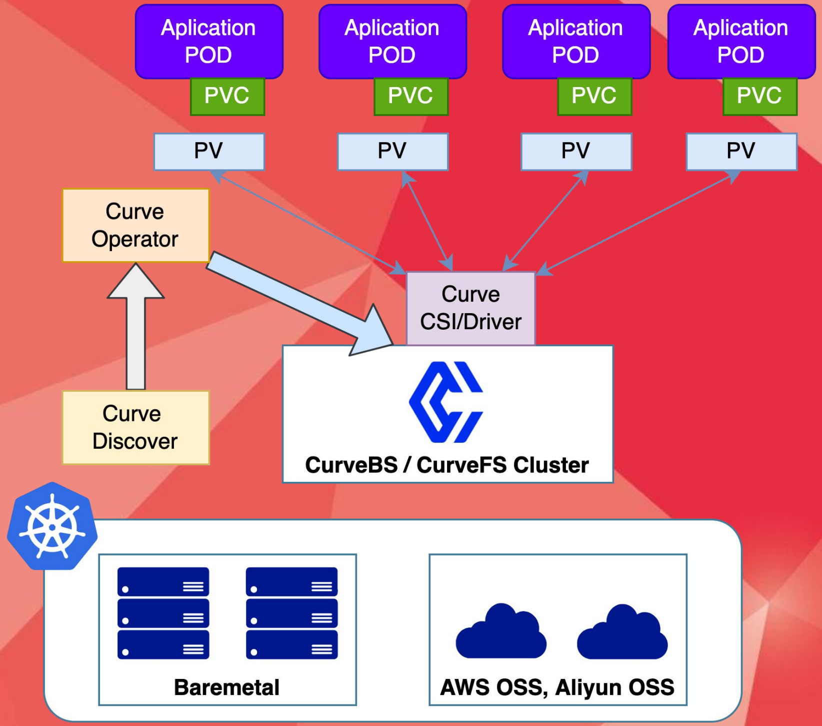
copset	replicas
CopySet$_1$	Node$_1$, Node$_5$, Node$_7$
CopySet$_2$	Node$_4$, Node$_6$, Node$_9$
CopySet$_3$	Node$_2$, Node$_3$, Node$_8$
CopySet$_4$	Node$_2$, Node$_7$, Node$_9$
CopySet$_5$	Node$_1$, Node$_3$, Node$_6$
CopySet$_6$	Node$_4$, Node$_5$, Node$_8$

ChunK$_2$	......	Chunk$_{64}$
Chunk$_{66}$	......	Chunk$_{128}$
......	......	......
Chunk$_{64n-62}$	......	Chunk$_{64n}$

CopySet$_1$	Chunk$_1$	ChunK$_2$	......
CopySet$_2$	Chunk$_3$	Chunk$_4$	......
CopySet$_3$	Chunk$_5$	Chunk$_6$	......
......	......	......	......

Cloud native Support

Currently we offer CSI Driver for block storage to provide PV/PVC resources on Kubernetes



Agenda

Why develop storage

Design objectives

Achievements in CURVE

√ Key designs used by CURVE

CURVE Roadmap

CURVE Community

You can take away

Key designs used by CURVE

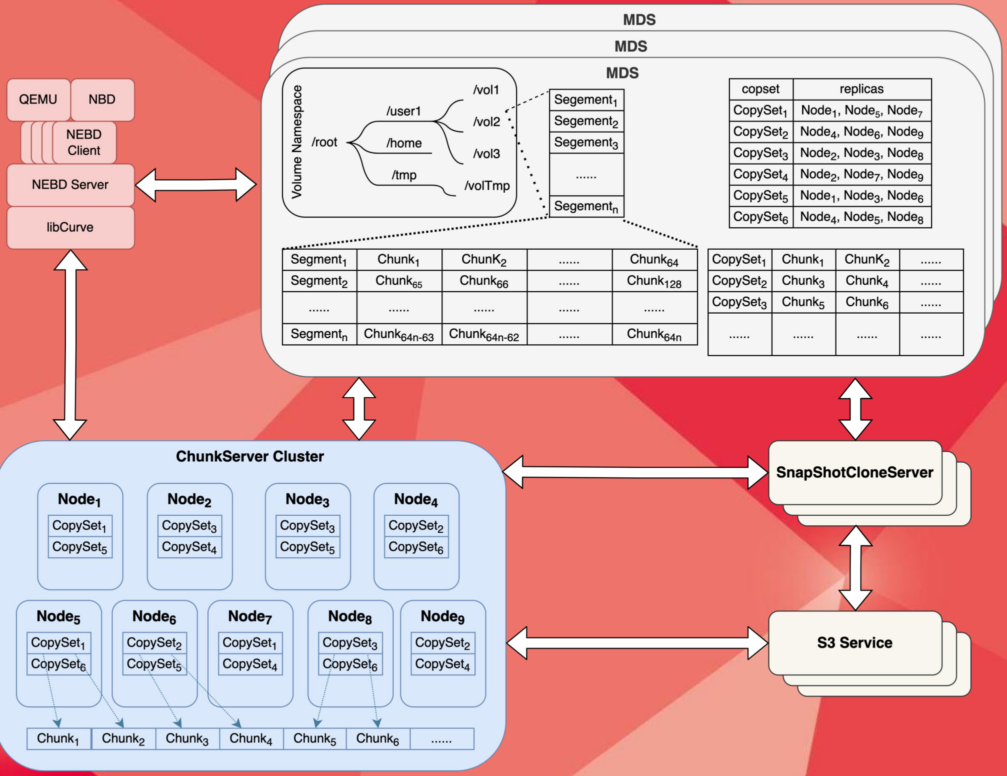
High availability and reliability

- Cluster topology
- CopySet pre-allocation algorithm
- Raft Consistency protocol

High performance

- pre-created file pool
- data strip like RAID
- Zero data copy
- RDMA

Cloud Native



Segement$_1$
Segement$_2$
Segement$_3$
......
Segement$_n$

copset	replicas
CopySet$_1$	Node$_1$, Node$_5$, Node$_7$
CopySet$_2$	Node$_4$, Node$_6$, Node$_9$
CopySet$_3$	Node$_2$, Node$_3$, Node$_8$
CopySet$_4$	Node$_2$, Node$_7$, Node$_9$
CopySet$_5$	Node$_1$, Node$_3$, Node$_6$
CopySet$_6$	Node$_4$, Node$_5$, Node$_8$

Segment$_1$	Chunk$_1$	ChunK$_2$	......	Chunk$_{64}$
Segment$_2$	Chunk$_{65}$	Chunk$_{66}$	......	Chunk$_{128}$
......	......	......	......	......
Segment$_n$	Chunk$_{64n-63}$	Chunk$_{64n-62}$	......	Chunk$_{64n}$

CopySet$_1$	Chunk$_1$	ChunK$_2$	......
CopySet$_2$	Chunk$_3$	Chunk$_4$	......
CopySet$_3$	Chunk$_5$	Chunk$_6$	......
......	......	......	......

Node$_1$	Node$_2$	Node$_3$	Node$_4$
CopySet$_1$	CopySet$_3$	CopySet$_3$	CopySet$_2$
CopySet$_5$	CopySet$_4$	CopySet$_5$	CopySet$_6$

Node$_5$	Node$_6$	Node$_7$	Node$_8$	Node$_9$
CopySet$_1$	CopySet$_2$	CopySet$_1$	CopySet$_3$	CopySet$_2$
CopySet$_6$	CopySet$_5$	CopySet$_4$	CopySet$_6$	CopySet$_4$

Chunk$_1$	Chunk$_2$	Chunk$_3$	Chunk$_4$	Chunk$_5$	Chunk$_6$	......

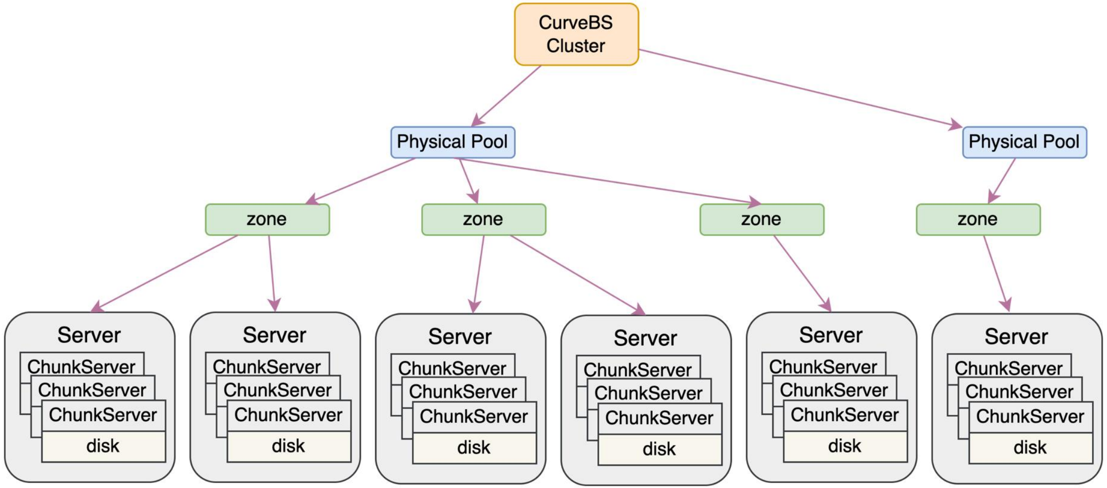
Cluster topology

The physical pool is used to physically isolate machine resources

Zone is the basic unit of fault isolation

Server Indicates a physical server

Chunkserver is a service instance on a physical server



OID + CND

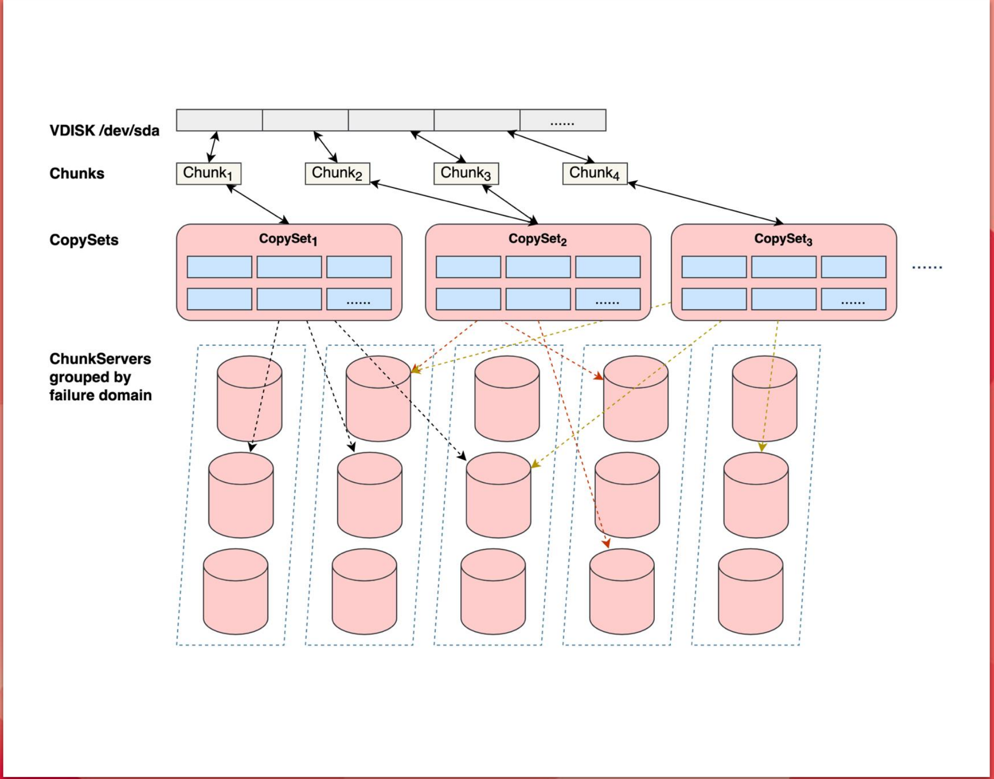
Curve metadata organization

Curve maps virtual block devices to files

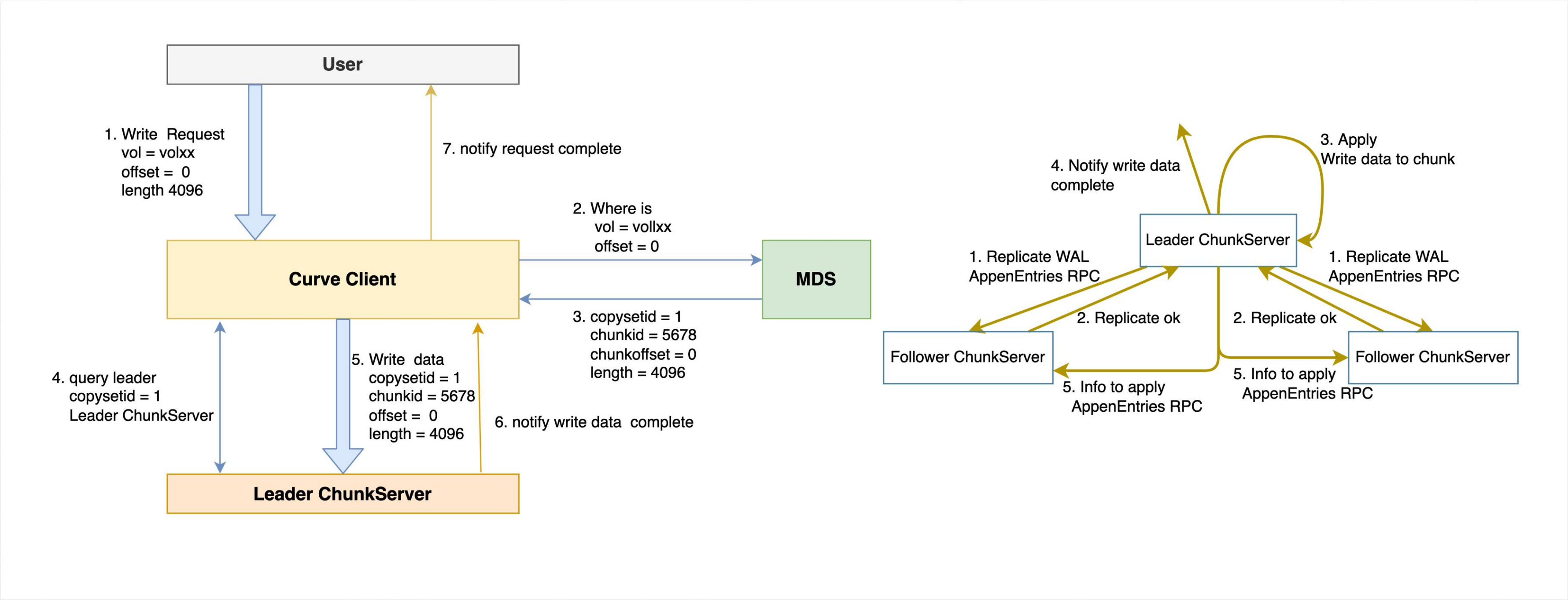
Each file contains chunks scattered across storage nodes in the cluster

Chunkservers are grouped by failure domains

Nodes in a COPYSET belong to different failure domains



CURVE IO data flow

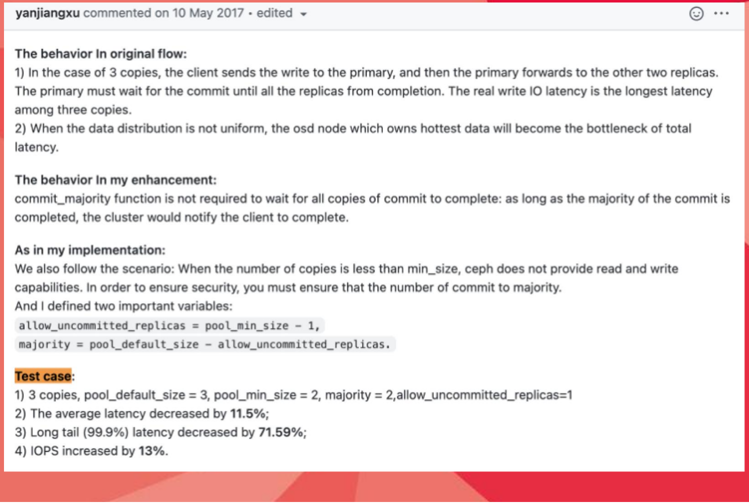


Other performance optimizations

RAFT protocol

Zero data copy

pre-created file pool



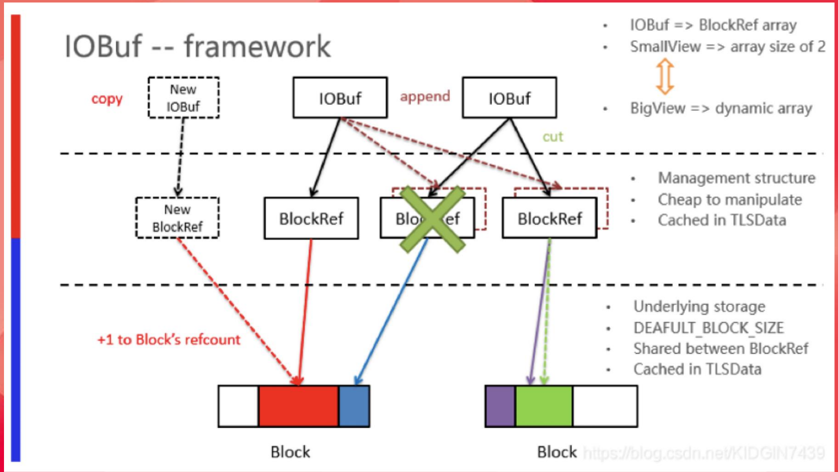
yanjiangxu commented on 10 May 2017 • edited

The behavior In original flow:
1) In the case of 3 copies, the client sends the write to the primary, and then the primary forwards to the other two replicas. The primary must wait for the commit until all the replicas from completion. The real write IO latency is the longest latency among three copies.
2) When the data distribution is not uniform, the osd node which owns hottest data will become the bottleneck of total latency.

The behavior In my enhancement:
commit_majority function is not required to wait for all copies of commit to complete: as long as the majority of the commit is completed, the cluster would notify the client to complete.

As in my implementation:
We also follow the scenario: When the number of copies is less than min_size, ceph does not provide read and write capabilities. In order to ensure security, you must ensure that the number of commit to majority.
And I defined two important variables:
`allow_uncommitted_replicas = pool_min_size - 1,`
`majority = pool_default_size - allow_uncommitted_replicas.`

Test case:
1) 3 copies, pool_default_size = 3, pool_min_size = 2, majority = 2,allow_uncommitted_replicas=1
2) The average latency decreased by **11.5%**;
3) Long tail (99.9%) latency decreased by **71.59%**;
4) IOPS increased by **13%**.



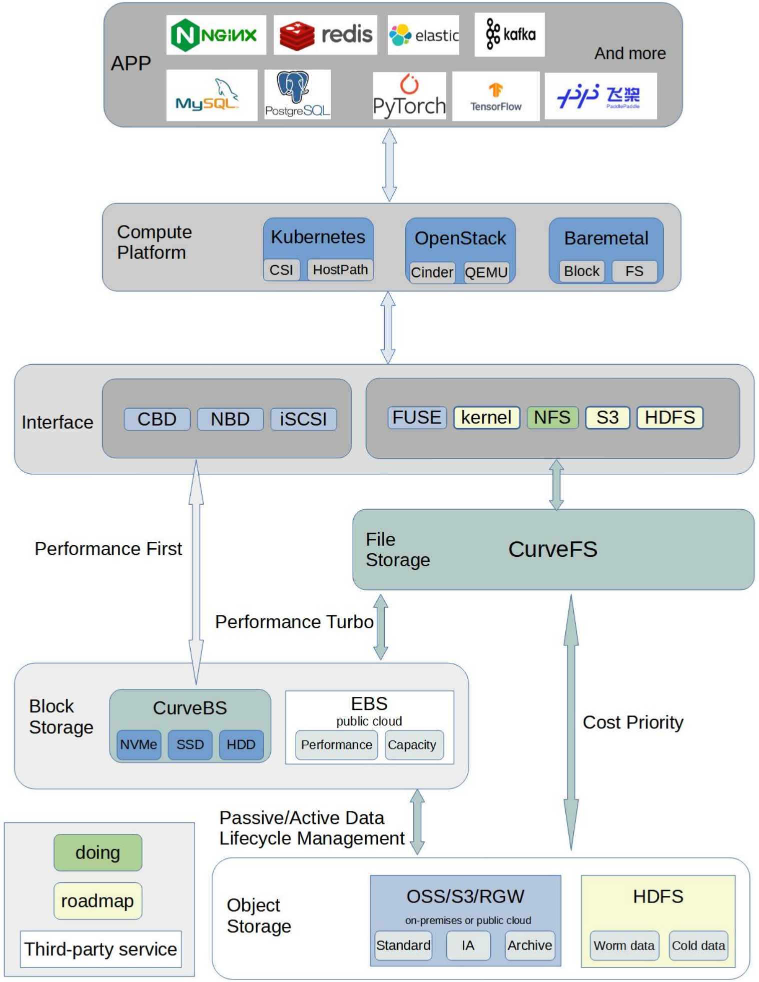
CURVE file system

File service middleware

Upper-layer applications can access file data in storage pools through interfaces such as NFS/HDFS/Posix api

Manage multiple types of storage (object storage, HDFS storage, Elastic block storage)

Support both on-premise and public clouds



Agenda

Why develop storage

Design objectives

Achievements in CURVE

Key designs used by CURVE

√ CURVE Roadmap

√ CURVE Community

√ You can take away

CURVE Roadmap

- Features
  - Supports RDMA and SPDK
  - Further reduce I/O latency and improve throughput
  - Support data lifecycle management
  - Curve block devices can be used as the underlying storage of Curve file systems
- System operation and maintenance
  - Further improve tools for installation/monitoring/maintenance
  - dashboard
- Cloud Native
  - Using CURVE file system in Kubernetes

CURVE community

The improvement and development of Curve cannot be achieved without the support and participation of community contributors.

Project address https://github.com/opencurve/curve

Release cycle: a major release every six months, and a minor release every 1-2 months

Follow up on CURVE progress: CURVE meetings are held every 2 weeks to explain CURVE progress and discuss related issues

Submit bugs and Suggestions:
https://github.com/openCURVE/CURVE/issues

Participated in the communication and discussion of Curve project: wechat group, qr code on the right hand side

You can take away

Challenges faced

- With the development of hardware and software, more and more stateful applications require to run on the clouds
- High availability/reliability
- Easy to operation and maintenance
- Cloud Native

CURVE solution

- RAFT protocol; Copyset allocation algorithm with topology-based failure domain to provide high availability/reliability
- Zero copy; Data stripe; RDMA to improve performance
- Management and monitor tools
- Support CSI deriver for upper cloud native applications access

THANKS