In-Depth-Interpretation-ChubaoFS

Speakers： Hongyin Zhu

Motivations

- Decoupling compute from storage
- Stateful containerized applications need persistent storage
- High scalability and availability
- Support diverse read/write patterns

ChubaoFS

A distributed file system for cloud native applications http://chubao.io/

Used in production for more than 200 applications/services on JD's container platform

Open sourced: https://github.com/chubaofs/chubaofs

- Apache 2.0 License

Publication

- Haifeng Liu, et al., CFS: A Distributed File System for Large Scale Container Platforms. SIGMOD '19, June 30-July 5, 2019, Amsterdam, Netherlands.
- https://arxiv.org/abs/1911.03001

Evolution

Containerized platform and storage solutions journey

Phase 1:
OpenStack + Docker

Storage solution:
local file system

Issues:
- Hard to schedule containers
- Hardware failure leads to data loss

Phase 2:
Kubernetes + Docker

Storage:
local file system + object storage

Issues:
- Suffer from storage locality as well
- Application has to use RESTful API or SDK library

Phase 3:
Kubernetes + Docker

Storage solution:
ChubaoFS

Advantages :
- Decoupling compute from storage
- High performance for heavy writes/reads

ChubaoFS

Previous Release

2019.04.02
Release v1.0.0

- Distributed file system with high availability and scalability
- Support sequential and random read/write patterns.
- Optimized for both large and small files.
- POSIX computable APIs.
  Support regular/directory/symlink file types.
- Support temporary file usage.

2019.06.11
Release v1.1.1

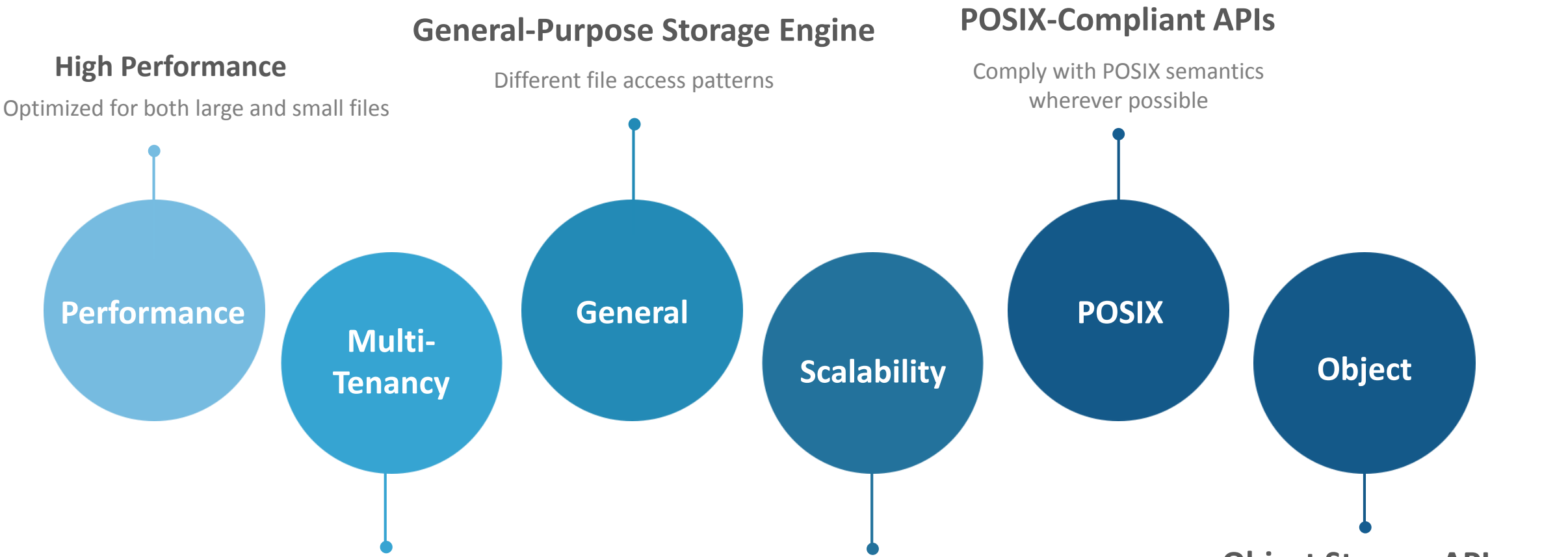
- Docker: introduce Docker compose to create a local testing ChubaoFS cluster

2020.01.08
Release v1.5.0

- add a general Authentication & Authorization framework
- object storage interface. Add ObjectNode to provide S3-compatible APIs l

ChubaoFS

Key Features



ChubaoFS

Potential Customers for ChubaoFS?

You need a generalized and persistent storage for your business

- May need to store both large files and small files
- May need to randomly or sequentially access data
- May be used for containerized or non-containerized services

You want to stick with open source solutions

- You prefer POSIX-compliant APIs to operate files

You need to handle extremely large number of files

You need high performance on file and metadata operations

ChubaoFS

Served Applications @ JD

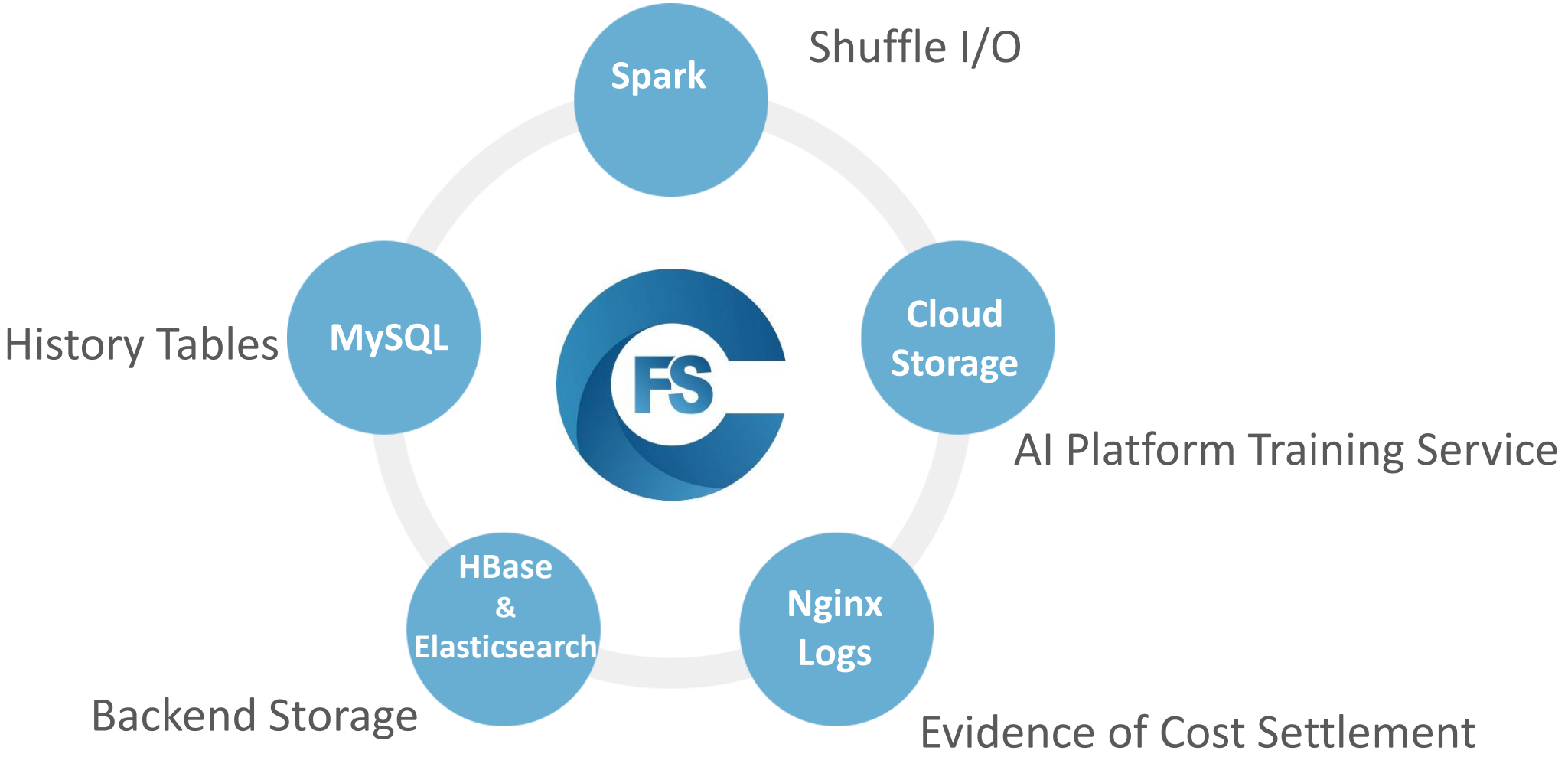
ChubaoFS served more than 200 applications at JD, such as:

- Click Stream
- Short videos
- Advertising
- Searching
- AI platform and services
- Order system
- Promotion system
- Monitoring system

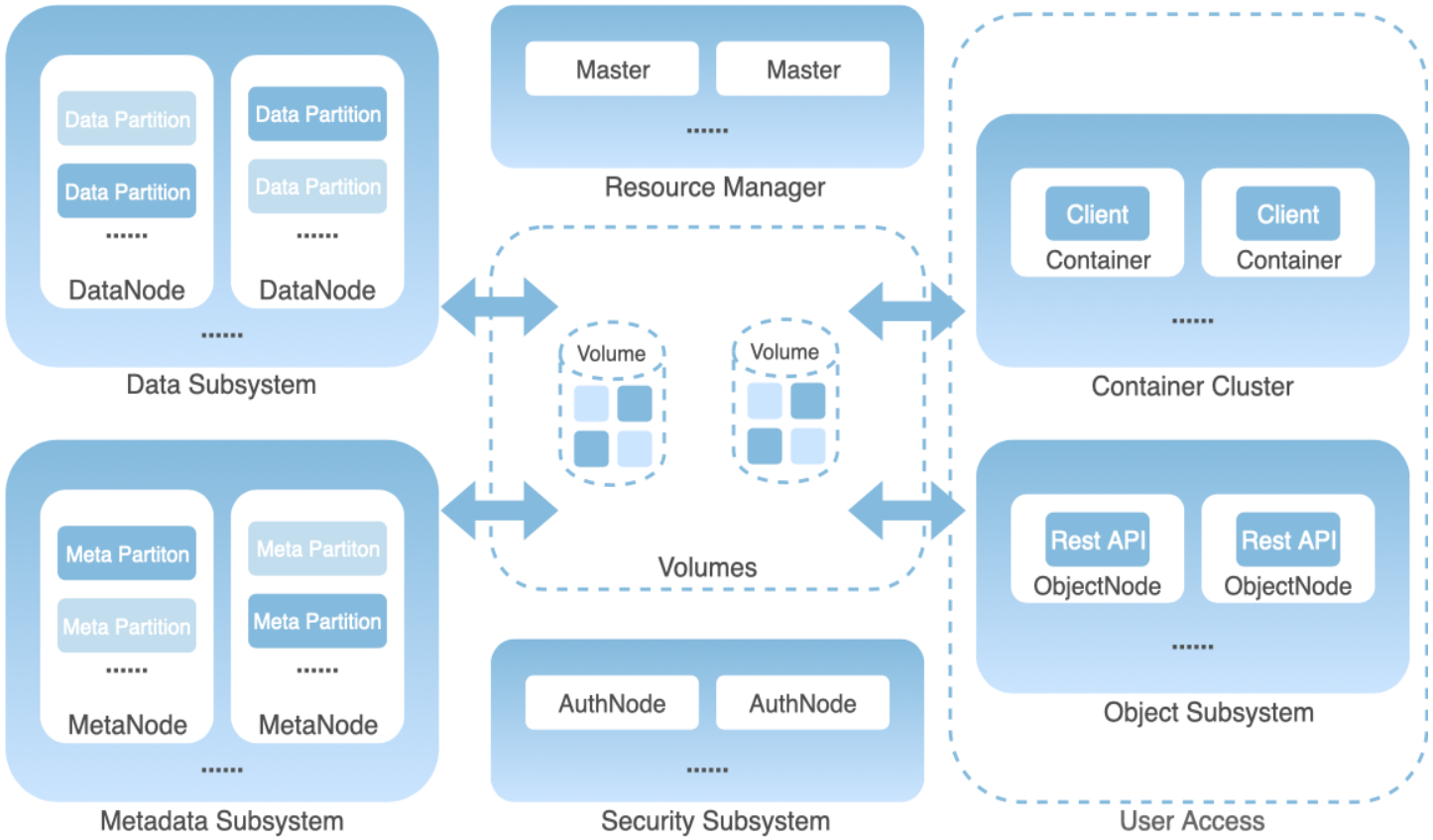
.....

ChubaoFS

Usage Scenarios



Architecture



Metadata Subsystem

- Meta Node
- Meta Partition

Data Subsystem

- Data Node
- Data Partition

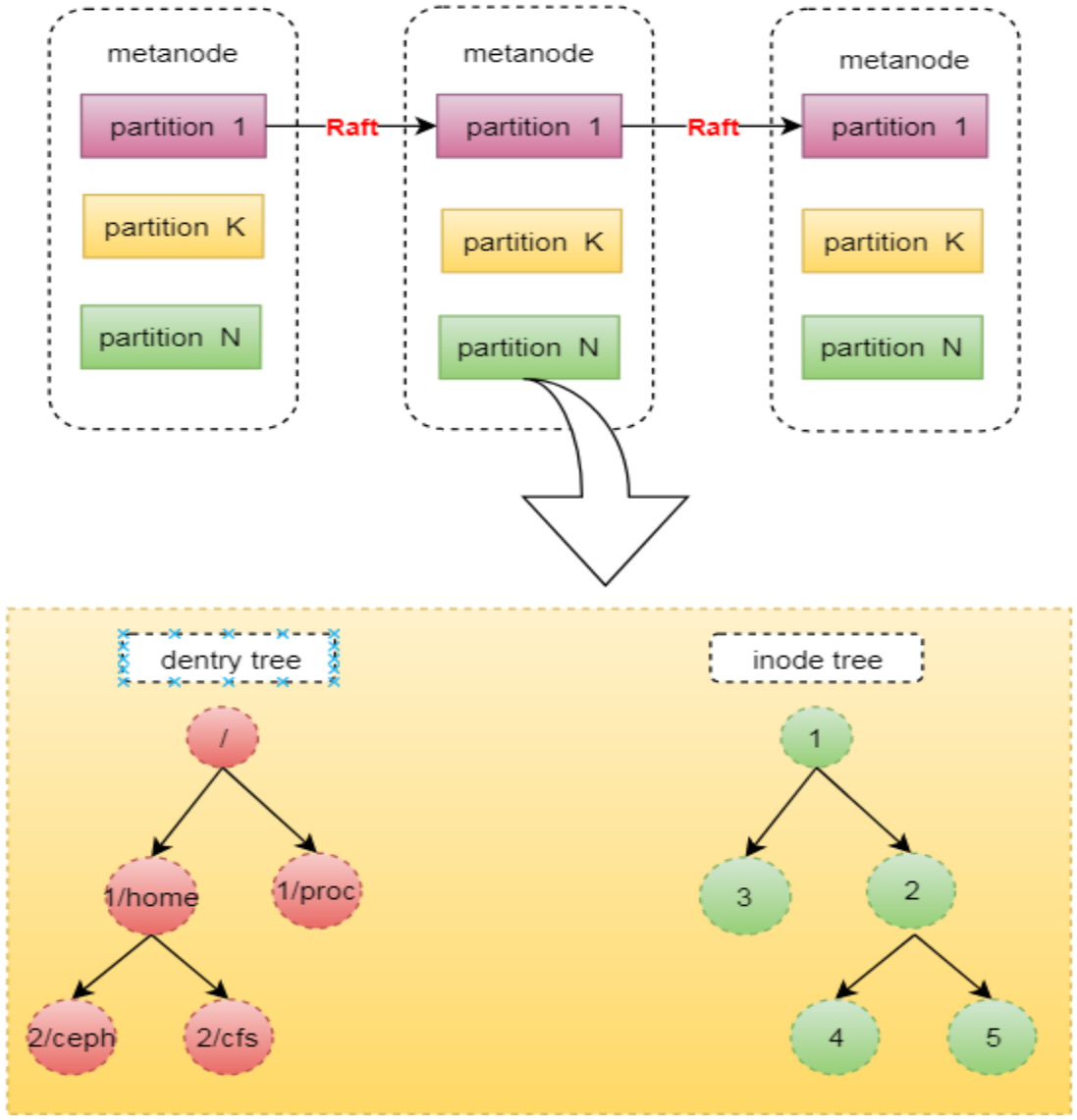
Resource Manager

- Volume

Client

ChubaoFS

Metadata Subsystem



- The subsystem has a set of meta nodes
- Each meta node has a set of meta partitions
- Each meta partition has several replicas (usually 3) in different meta nodes to form a Raft group
- Each meta partition belongs to a specific volume
- Raft logs and snapshot
- Multi-raft based (https://www.cockroachlabs.com/blog/scaling-raft/)

ChubaoFS

Data Subsystem

Place to persist the file contents

- Data Node and Data Partition
- Sequential or random access

Large files

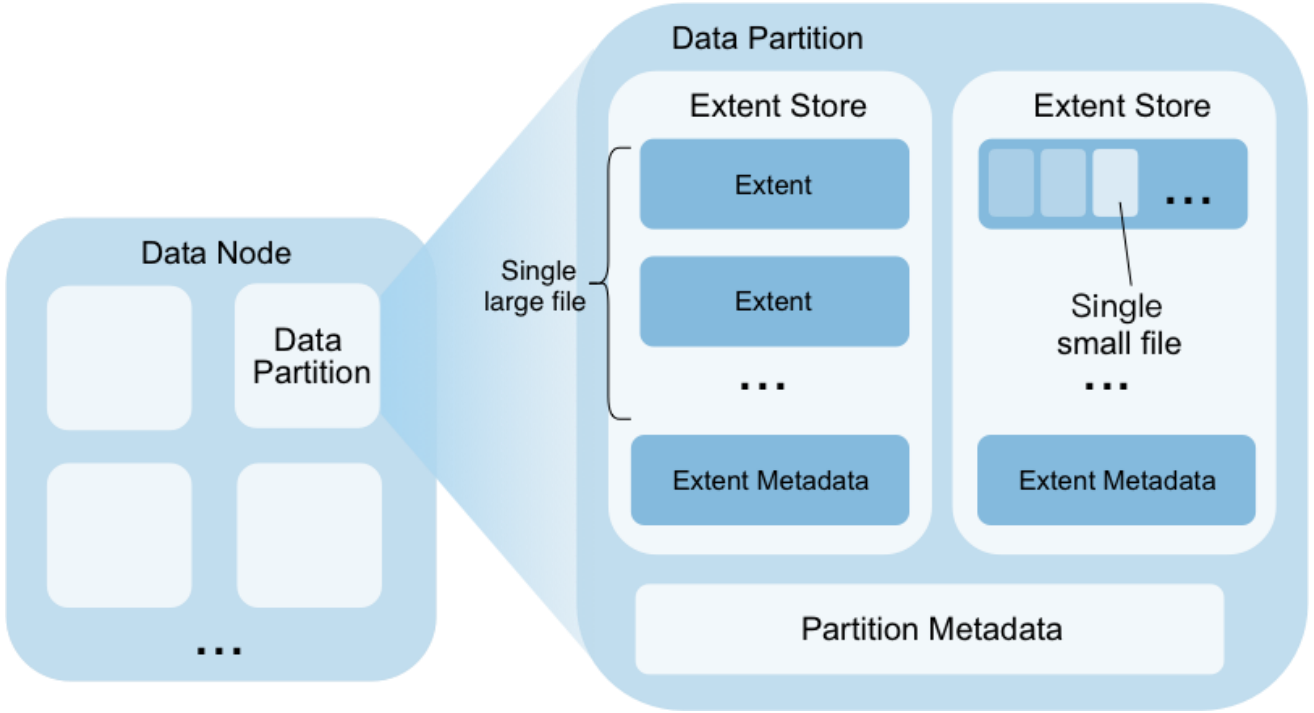
- One file -> a set of new extents
- Extent: a storage unit of data node

Small files

- One extent -> multiple small files
- Deleting a file and free disk space through punch hole

Dual replication protocol

- Primary-Backup for sequential write: better performance than Raft
- Multi-Raft for overwrite: ensure strong replication consistency at the cost of performance



ChubaoFS

Object Subsystem

Operate files like using object storage

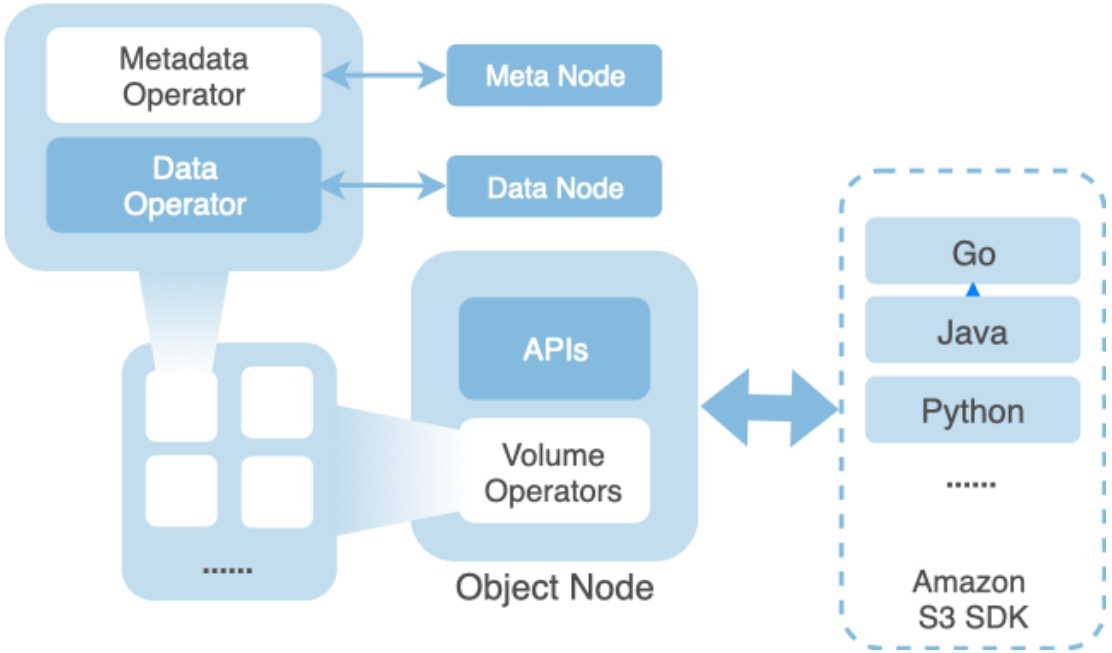
- HTTP based Rest APIs
- Compatible with native Amazon S3 SDKs

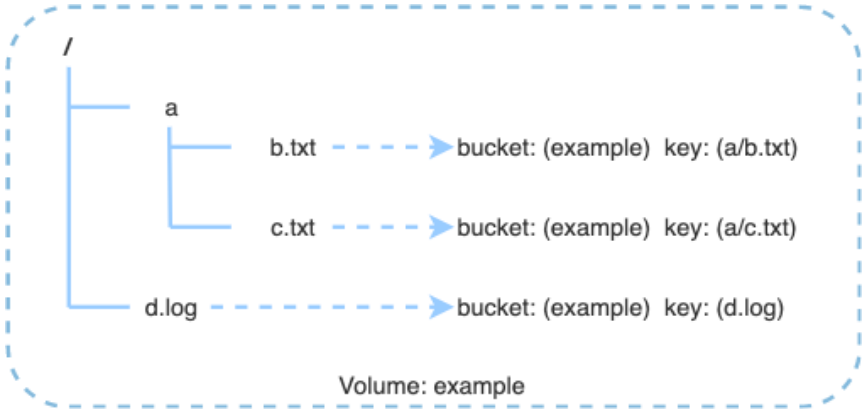
Fusion Storage

- Expose two interface (POSIX and object storage interface)

Semantic Convert

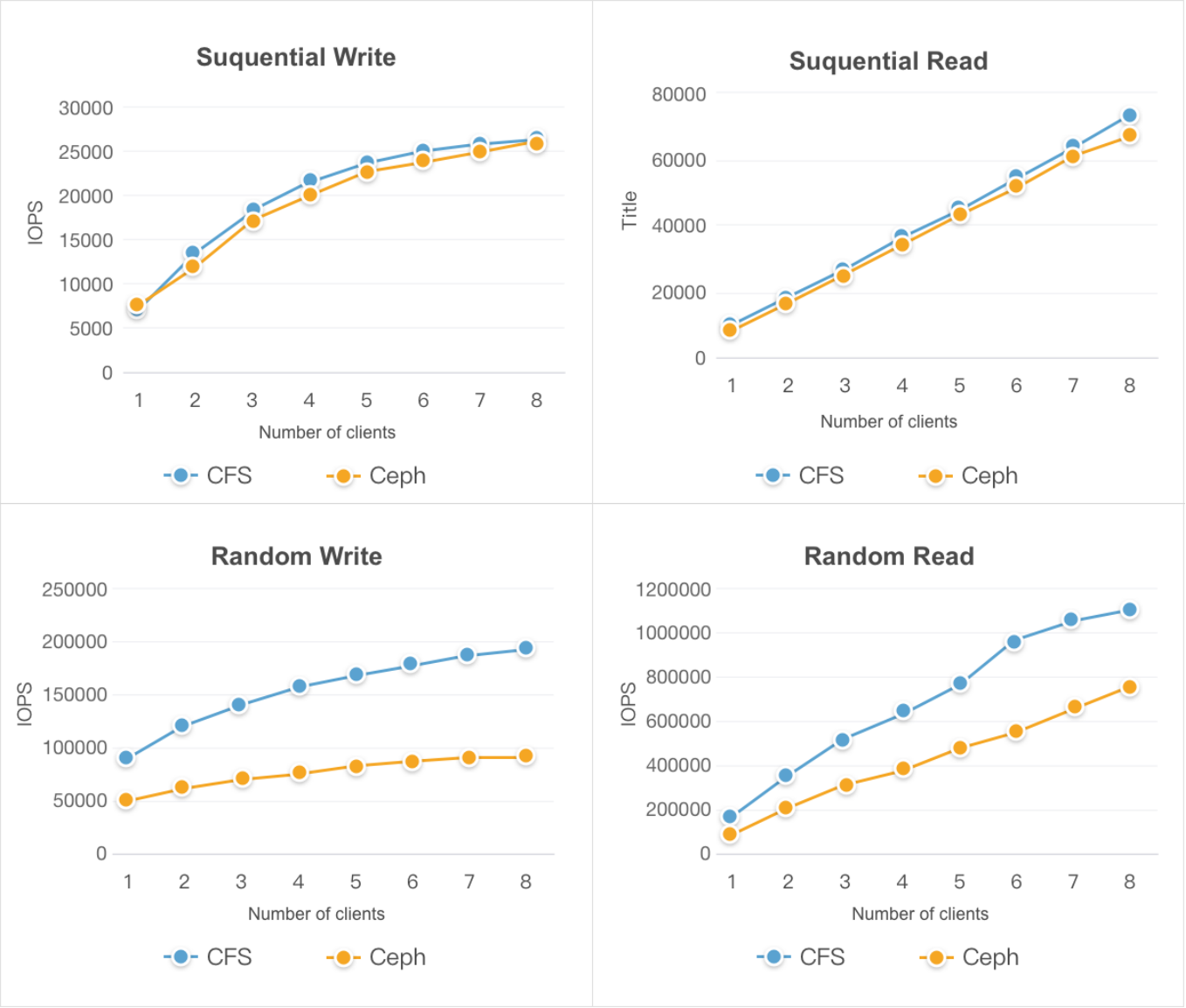
- Bucket -> Volume
- Key -> Path





ChubaoFS

Performance – Large File



Setup

- 64 processes in random read/write tests
- 16 processes in the sequential read/write tests
- Each process operates a separated 40 GB file

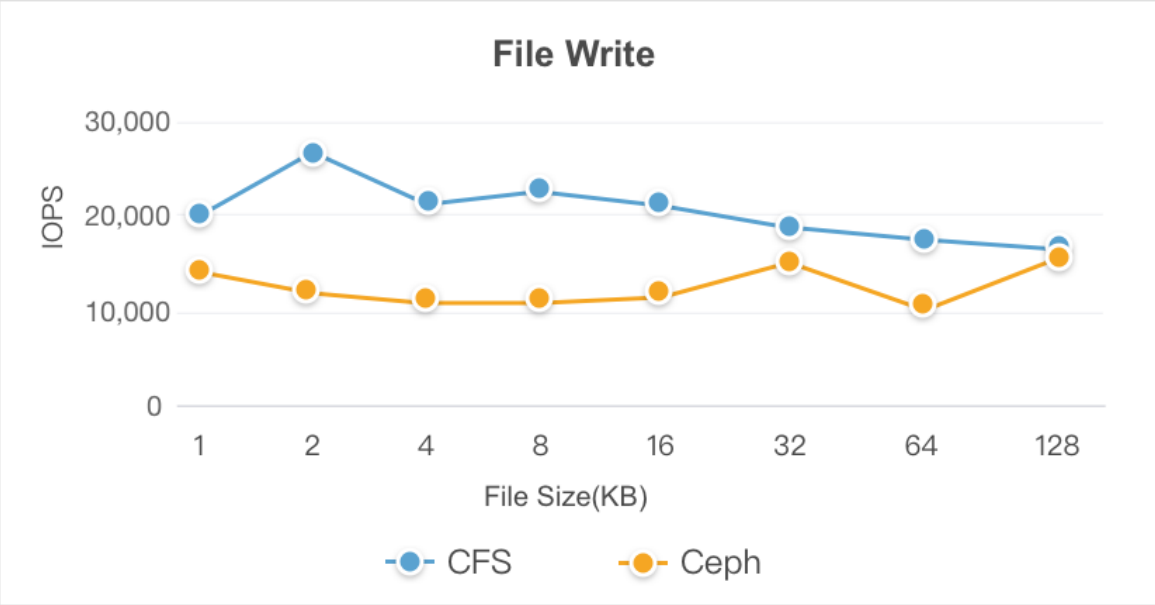
ChubaoFS outperforms CephFS in the random read/write tests

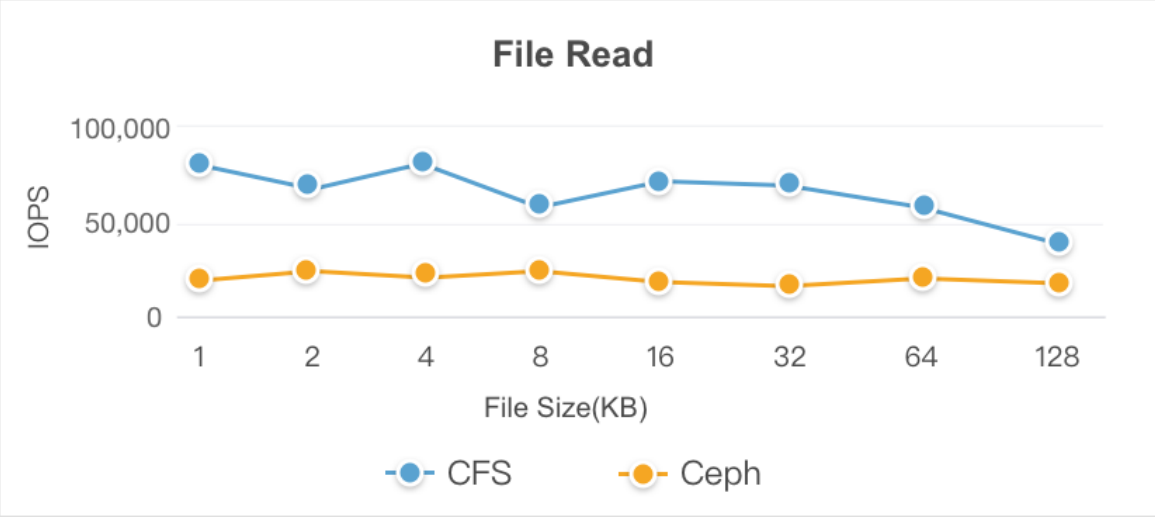
Why?

- Each meta node caches all file metadata in memory to avoid expensive disk IOs
- Overwrite in ChubaoFS is in-place (no need to update metadata)

ChubaoFS

Performance – Small File





Setup

- File size ranging from 1 KB to 128 KB

ChubaoFS outperforms CephFS in both small file read / write tests

Why?

- Small files: client does not ask for new extents (pre-allocate)
- Client sends write requests to the data node directly without extra communication

ChubaoFS

Integrates with container platform

Deploy cluster using docker compose

- Helper tool called run_docker.sh
- Build and runtime images available in hub.docker.com/u/chubaofs

Kubernetes CSI driver

- Available in github.com/chubaofs/chubaofs-csi
- Compatible with spec v0.3.0 and v1.0.0 for now

OpenSDS driver

- Used as southbound on-premise storage

Rook/Helm Chart

- deployment with Helm is ready, Rook scheduled and call for volunteers

Community

Mailing list:	chubaofs-maintainers@groups.io
Website:	http://chubao.io/
Twitter:	@ChubaoFS
Slack:	chubaofs.slack.com
Issue tracker:	Use the GitHub Issue tracker to file bugs and feature request
Contributing:	We welcome community contributions !

Thanks!

ChubaoFS Contributors:
Haifeng Liu, Wei Ding, Weilong Guo, Shuoran Liu, Tianpeng Li, Mofei Zhang, Jianxing Zhao, Hongyin Zhu, Zhengyi Zhu, Liying Zhang

Contact: chubaofs-maintainers@groups.io