# CFS Documentation

The Chubao Authors

May 28, 2020

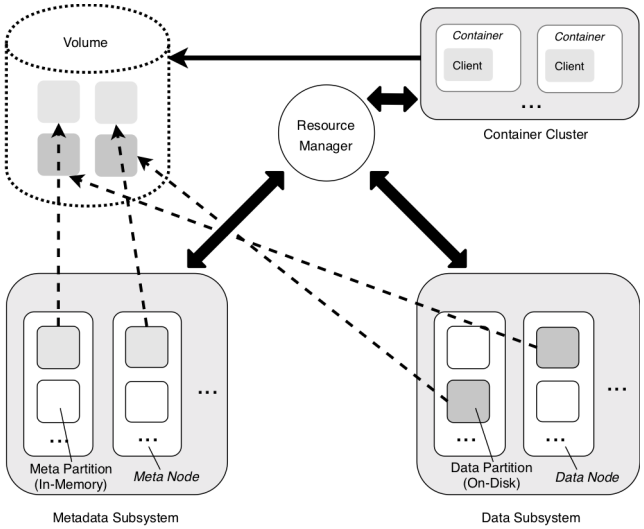
# Getting Started

CHAPTER 1

# Introduction

CFS(Chubao File System) is a distributed fle system that is designed to natively support large scale container platforms.

## 1.1 High Level Architecture




CFS consists of a *metadata subsystem*, a *data subsystem*, and a *resource manager*, and can be accessed by different *clients* (as a set of application processes) hosted on the containers through different file system instances called *volumes*.

The metadata subsystem stores the file metadata, and consists of a set of *meta nodes*. Each meta node consists of a set of *meta partitions*.

The data subsystem stores the file contents, and consists of a set of *data nodes*. Each data node consists of a set of *data partitions*.

The volume is a logical concept in CFS and consists of one or multiple meta partitions and one or multiple data partitions. Each partition can only be assigned to a single volume. From a client's perspective, the volume can be viewed as a file system instance that contains data accessible by the containers. A volume can be mounted to multiple containers so that files can be shared among different clients simultaneously, and needs to be created at the very beginning before the any file operation. A CFS cluster deployed at each data center can have hundreds of thousands of volumes, whose data sizes vary from a few gigabytes to several terabytes.

Generally speaking, the resource manager periodically communicates with the metadata subsystem and data subsystem to manage the meta nodes and data nodes, respectively. Each client periodically communicates with the resource manager to obtain the up-to-date view of the mounted volume. A file operation usually initiates the communications from the client to the corresponding meta node and data node directly, without the involvement of the resource manager. The updated view of the mounted volume, as well as the file metadata are usually cached at the client side to reduce the communication overhead.

## 1.2 Features

### 1.2.1 Scalable Meta Management

The metadata operations could make up as much as half of typical file system workloads. On our platform, this becomes even more important as there could be heavy accesses to the metadata of files by tens of thousands of clients simultaneously. A single node that stores the file metadata could easily become the performance bottleneck. As a result, we employ a distributed metadata subsystem to manage the file metadata. In this way, the metadata requests from different clients can be forwarded to different nodes, which improves the scalability of the entire system. The metadata subsystem can be considered as an in-memory datastore of the file metadata. It can have thousands of meta nodes, each of which can have hundreds of billions of meta partitions. Each meta partition on a meta node stores the file metadata in memory by maintaining a set of inodes and a set of dentries. We employ two b-trees called inodeTree and dentryTree for fast lookup of inodes and dentries in the memory. The inodeTree is indexed by the inode id, and the dentryTree is indexed by the dentry name and the parent inode id. We also maintain a range of the inode ids (denoted as start and end) stored on a meta partition for splitting.

### 1.2.2 General-Purpose Storage Engine

To reduce the storage cost, many applications and services are served from the same shared storage infrastructure (aka "multi-tenancy"). The workloads of different applications and services are mixed together, where the file size can vary from a few kilobytes to hundreds of gigabytes, and the files can be written in a sequential or random fashion. For example, the log files usually need to be written sequentially in the execution order of the code; some data analytics in the machine learning domain are based on the data stored on the underlying file system; and a database engine running on top of the file system can modify the stored data frequently. A dedicated file system needs to be able to serve for all these different workloads with excellent performance.

### 1.2.3 Strong Replication Consistency

E-commence venders who move their line of business applications to the cloud usually prefer strong consistency. For example, an image processing service may not want to provide the customer with an outdated image that does not match the product description. This can be easily achieved if there is only one copy of the file. But to ensure a distributed file system to continue operating properly in the event of machines failures, which can be caused by various reasons such as faulty hard drives, bad motherboards, etc, there are usually multiple replicas of the same file. As a result, in a desired file system, the data read from any of the replicas must be consistent with each other.

### 1.2.4 Relaxed POSIX Semantics and Metadata Atomicity

In a POSIX-compliant distributed file system, the behavior of serving multiple processes on multiple client nodes should be the same as the behavior of a local file system serving multiple processes on a single node with direct attached storage. CFS provides POSIX-compliant APIs. However, the POSIX consistency semantics, as well as the atomicity requirement between the inode and dentry of the same file, have been carefully relaxed in order to better align with the needs of applications and to improve the system performance.

CHAPTER 2

# Quick Start Guide

## 2.1 Building

### 2.1.1 Build Servers

In CFS, the server consists of the resource manager, metanode and datanode, which are compiled to a single binary for deployment convenience.

Building of CFS server depends on RocksDB, build RocksDB v5.9.2+ . Recommended installation uses *make static_lib* .

CFS server is built with the following command:

```
cd cmd; sh build.sh
```

### 2.1.2 Build Client

```
cd client; sh build.sh
```

## 2.2 Deployment

### 2.2.1 Start Resource Manager

```
nohup ./cmd -c master.json &
```

Sample *master.json* is shown as follows,

```
{
  "role": "master",
  "ip": "192.168.31.173",
  "port": "80",
  "prof":"10088",
  "id":"1",
  "peers": "1:192.168.31.173:80,2:192.168.31.141:80,3:192.168.30.200:80",
  "retainLogs":"20000",
  "logDir": "/export/Logs/cfs/master",
  "logLevel":"info",
  "walDir":"/export/Logs/cfs/raft",
  "storeDir":"/export/cfs/rocksdbstore",
  "consulAddr": "http://consul.prometheus-cfs.local",
  "exporterPort": 9510,
  "clusterName":"cfs"
}
```

For detailed explanations of *master.json*, please refer to *Resource Manager*.

### 2.2.2 Start Metanode

```
nohup ./cmd -c meta.json &
```

Sample *meta.json is* shown as follows,

```
{
    "role": "metanode",
    "listen": "9021",
    "prof": "9092",
    "logLevel": "debug",
    "metaDir": "/export/cfs/metanode_meta",
    "logDir": "/export/Logs/cfs/metanode",
    "raftDir": "/export/cfs/metanode_raft",
    "raftHeartbeatPort": "9093",
    "raftReplicatePort": "9094",
    "consulAddr": "http://consul.prometheus-cfs.local",
    "exporterPort": 9511,
    "masterAddrs": [
        "192.168.31.173:80",
        "192.168.31.141:80",
        "192.168.30.200:80"
    ]
}
```

For detailed explanations of *meta.json*, please refer to *Meta Subsystem*.

### 2.2.3 Start Datanode

1. Prepare data directories

   **Recommendation** Using independent disks can reach better performance.

   **Disk preparation**

   1.1 Check available disks

```
fdisk -l
```

1.2 Build local Linux file system on the selected devices

```
mkfs.xfs -f /dev/sdx
```

1.3 Make mount point

```
mkdir /data0
```

1.4 Mount the device on mount point

```
mount /dev/sdx /data0
```

2. Start datanode

```
nohup ./cmd -c datanode.json &
```

Sample *datanode.json* is shown as follows,

```
{
  "role": "datanode",
  "port": "6000",
  "prof": "6001",
  "logDir": "/export/Logs/datanode",
  "logLevel": "info",
  "raftHeartbeat": "9095",
  "raftReplica": "9096",
  "consulAddr": "http://consul.prometheus-cfs.local",
  "exporterPort": 9512,
  "masterAddr": [
  "192.168.31.173:80",
  "192.168.31.141:80",
  "192.168.30.200:80"
  ],
  "rack": "",
  "disks": [
     "/data0:107374182400"
  ]
}
```

For detailed explanations of *datanode.json*, please refer to *Data Subsystem*.

### 2.2.4 Create Volume

By decault, there are only a few data partitions allocated upon volume creation, and will be dynamically expanded according to actual usage. For performance evaluation, it is better to preallocate enough data partitions.

```
curl -v "http://127.0.0.1/admin/createVol?name=test&capacity=100&owner=cfs"
```

## 2.3 Mount Client

1. Run `modprobe fuse` to insert FUSE kernel module.

2. Run `yum install -y fuse` to install libfuse.
3. Run `nohup client -c fuse.json &` to start a client.

   Sample *fuse.json* is shown as follows,

```
{
  "mountPoint": "/mnt/fuse",
  "volName": "test",
  "owner": "cfs",
  "masterAddr": "192.168.31.173:80,192.168.31.141:80,192.168.30.200:80",
  "logDir": "/export/Logs/cfs",
  "profPort": "10094",
  "logLevel": "info"
}
```

For detailed explanations of *fuse.json*, please refer to *Client*.

Note that end user can start more than one client on a single machine, as long as mountpoints are different.

## 2.4 Upgrading

1. freeze the cluster

```
curl -v "http://127.0.0.1/cluster/freeze?enable=true"
```

2. upgrade each module
3. closed freeze flag

```
curl -v "http://127.0.0.1/cluster/freeze?enable=false"
```

CHAPTER 3

# Resource Manager

The resource manager manages the file system by processing different types of tasks, such as creating/deleting/updating/loading partitions and keeping track of the resource status (such as the memory/disk utilization). The resource manager is also responsible for creating new volumes and adding new meta/data nodes to the CFS cluster. It has multiple replicas, among which the consistency is maintained by a consensus algorithm such as Raft, and persisted to a key value store such as RocksDB for backup and recovery.

## 3.1 Utilization-Based Distribution/Placement

The resource manager is a utilization-based distribution strategy to places the file metadata and contents across different meta and data nodes. Because each node can have multiple partitions, and the partitions in a volume do not need to reside on the same node, this distribution can be controlled at a finer granularity, resulting a more efficient resource management. Specifically, the distribution of file metadata and contents works follows:

First, when mounting a volume, the client asks the resource manager for a set of available meta and data partitions. These partitions are usually the ones on the nodes with the lowest memory/disk utilizations. Later on, when writing a file, the client can only choose the meta and data partitions in a random fashion from the ones allocated by the resource manager.

Second, when the resource manager finds that all the partitions in a volume is about to be full, it automatically adds a set of new partitions to this volume. These partitions are usually the ones on the nodes with the lowest memory/disk utilizations. Note that, when a partition is full, or a threshold (i.e., the number of files on a meta partition or the number of extents on a data partition) is reached, no new data can be stored on this partition, although it can still be modified or deleted.

## 3.2 Replica Placement

When choosing partitions for the replicas, the resource manager ensures that two replicas of the same partition never reside on the same node.

## 3.3 Meta Partition Splitting

There is a special requirement when splitting a meta partition. In particular, if a meta partition is about to reach its upper limit of the number of stored inodes and dentries, a splitting task needs to be performed with the requirement to ensure that the inode ids stored at the newly created partition are unique from the ones stored at the original partition.

To meet this requirement, when splitting a meta partition, the resource manager cuts off the inode range of the meta partition in advance at a upper bound *end*, a value greater than highest inode id used so far (denoted as *maxInodeID*), and sends a split request to the meta node to (1) update the inode range from *1* to *end* for the original meta partition, and (2) create a new meta partition with the inode range from *end + 1* to *infinity* for this volume. As a result, the inode range for these two meta partitions becomes *[1, end]* and *[end + 1, infinity]*, respectively. If there is another file needs to be created, then its inode id will be chosen as *maxInodeID + 1* in the original meta partition, or *end + 1* in the newly created meta partition. The *maxInodeID* of each meta partition can be obtained by the periodical communication between the resource manager and the the meta nodes.

## 3.4 Exception Handling

When a request to a meta/data partition times out (e.g., due to network outage), the remaining replicas of this partition are marked as read-only. When a meta/data partition is no longer available (e.g., due to hardware failures), all the data on this partition will eventually be migrated to a new available partition manually. This unavailability is identified by the multiple failures reported by the node when operating the files.

CHAPTER 4

# Metadata Subsystem

The metadata operations could make up as much as half of typical file system workloads. this can be important as there could be heavy accesses to the metadata of files by tens of thousands of clients simultaneously. A single node that stores the file metadata could easily become the performance/storage bottleneck. As a result, we employ a distributed metadata subsystem to manage the file metadata. In this way, the metadata requests from different clients can be forwarded to different nodes, which improves the scalability of the entire system.

## 4.1 Internal Structure

The metadata subsystem can be considered as an in-memory datastore of the file metadata. It can have thousands of meta nodes, each of which can have a set of meta partitions. Each meta partition on a meta node stores the file metadata in memory by maintaining a set of *inodes* and a set of *dentries*.

Generally speaking, An inode is an object that represents the underlying file (or directory), and can be identified by an unsigned 64-bit integer called the *inode id*. A dentry is an object that represents the directory hierarchy and can be identified by a string name and the id of the parent inode. For example, if we have two directories *foo* and *bar*, where *foo* is the parent directory of *bar*, then there are two inodes: one for *foo* called *i1*, and the other for *bar* called *i2*, and one dentry to represent the hierarchy of these two directories where *i2* is the current inode and *i1* is the parent inode.

A meta partition can only store the inodes and dentries of the files from the same volume. We employ two b-trees called *inodeTree* and *dentryTree* for fast lookup of inodes and dentries in the memory. The *inodeTree* is indexed by the inode id, and the *dentryTree* is indexed by the dentry name and the parent inode id. We also maintain a range of the inode ids (denoted as *start* and *end*) stored on a meta partition for splitting (see *Resource Manager*).

## 4.2 Replication

The replication during file write is performed in terms of meta partitions. The replication consistency is ensured by a revision of the Raft consensus protocol called the MultiRaft, which has the advantage of reduced heartbeat network traffic comparing to the original version.

## 4.3 Failure Recovery

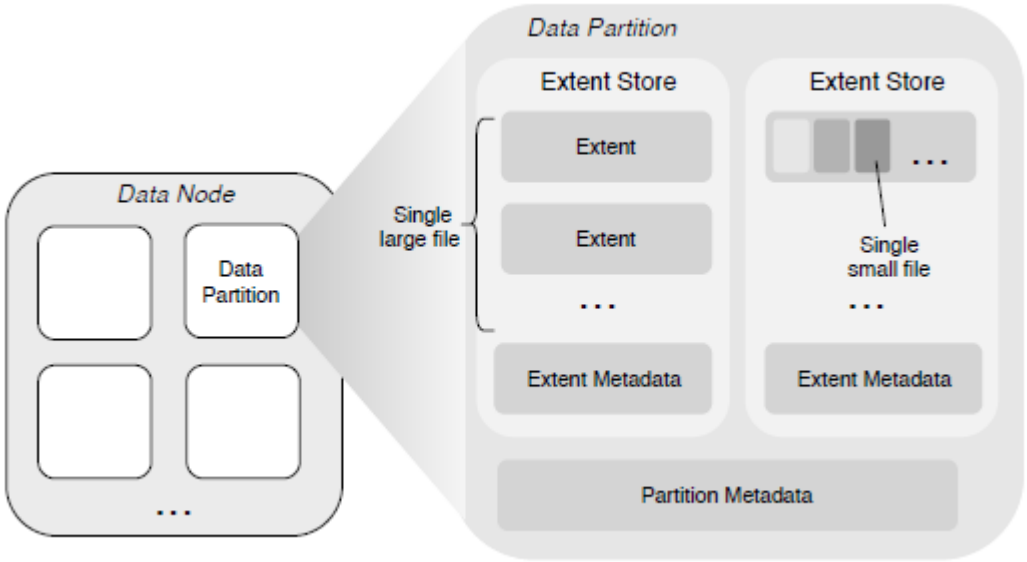
The in-memory meta partitions are persisted to the local disk by snapshots and logs for backup and recovery. Some techniques such as log compaction are used to reduce the log files sizes and shorten the recovery time.

It is worth noting that, a failure that happens during a metadata operation could result an *orphan* inode with which has no dentry to be associated. The memory and disk space occupied by this inode can be hard to free. To minimize the chance of this case to happen, the client always issues a retry after a failure until the request succeeds or the maximum retry limit is reached.

CHAPTER 5

# Data Subsystem

The data subsystem is optimized for the storage of large and small files, which can be accessed in a sequential or random fashion.




## 5.1 Features

- Large File Storage

For large files, the contents are stored as a sequence of one or multiple extents, which can be distributed across different data partitions on different data nodes. Writing a new file to the extent store always causes the data to be written at the zero-offset of a new extent, which eliminates the need for the offset within the extent. The last extent of a file does not need to fill up its size limit by padding (i.e., the extent does not have holes), and never stores the data from other files.

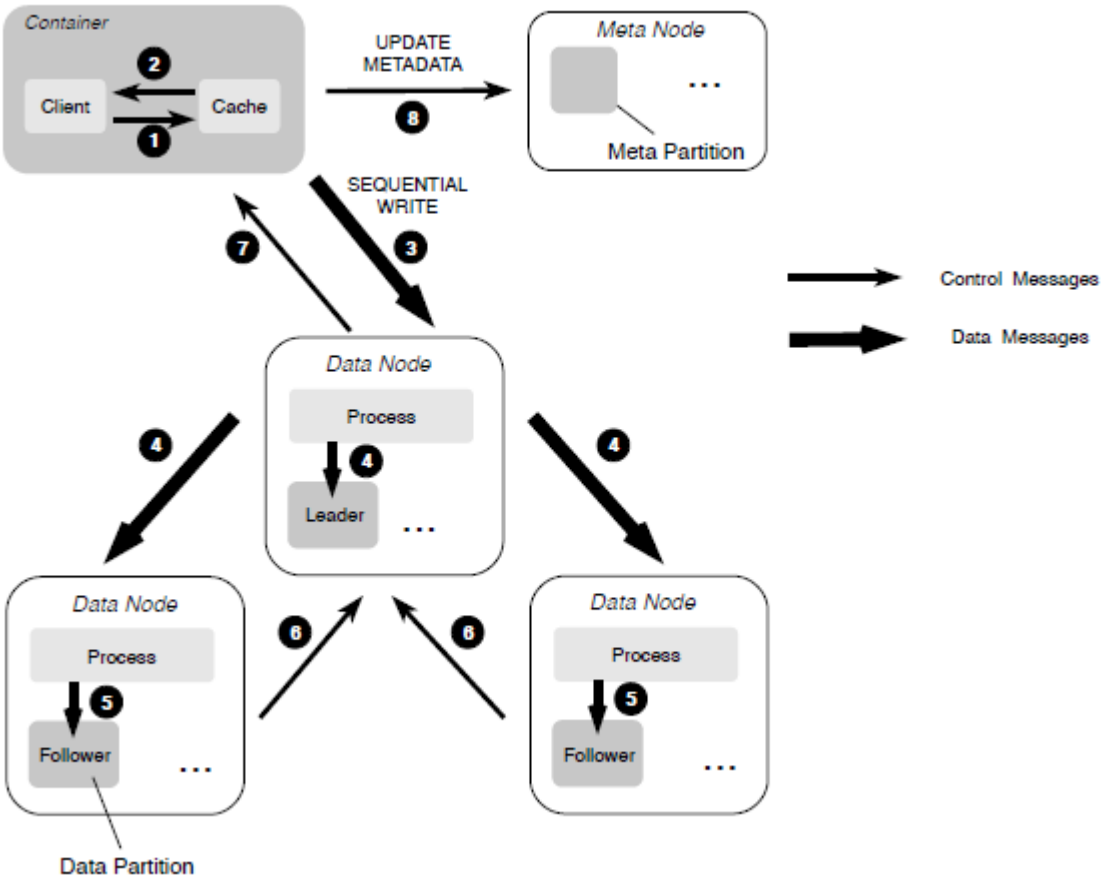
- Small File Storage

The contents of multiple small files are aggregated and stored in a single extent, and the physical offset of each file content in the extent is recorded in the corresponding meta node. CFS relies on the punch hole interface, textit{fallocate()}footnote{url{http://man7.org/linux/man-pages/man2/fallocate.2.html}}, to textit{asynchronous} free the disk space occupied by the to-be-deleted file. The advantage of this design is to eliminate the need of implementing a garbage collection mechanism and therefore avoid to employ a mapping from logical offset to physical offset in an extent~cite{haystack}. Note that this is different from deleting large files, where the extents of the file can be removed directly from the disk.

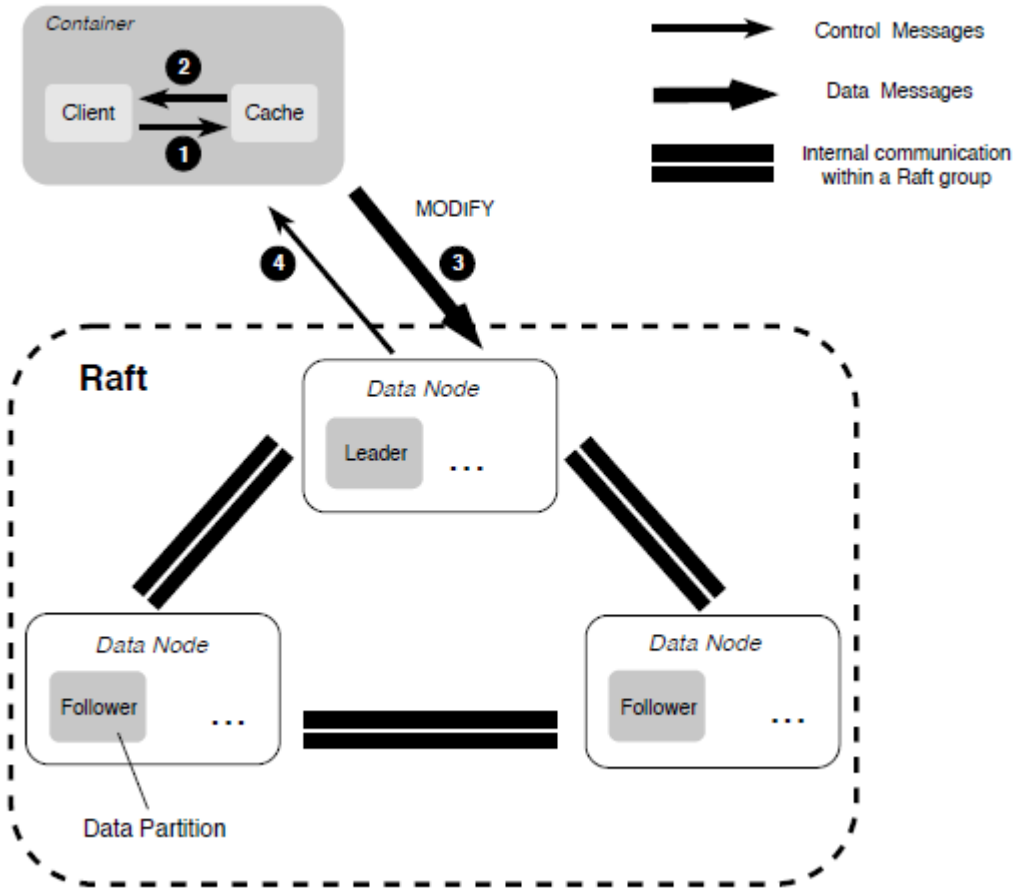
- Replication

  The replication is performed in terms of partitions during file writes. Depending on the file write pattern, CFS adopts different replication strategies.

  When a file is sequentially written into CFS, a primary-backup replication protocol is used to ensure the strong consistency with optimized IO throughput.




  When overwriting an existing file portion during random writes, we employ a MultiRaft-based replication protocol, which is similar to the one used in the metadata subsystem, to ensure the strong consistency.




- Failure Recovery

  Because of the existence of two different replication protocols, when a failure on a replica is discovered, we first start the recovery process in the primary-backup-based replication by checking the length of each extent and making all extents aligned. Once this processed is finished, we then start the recovery process in our MultiRaft-based replication.

## 5.2 HTTP APIs

| API | Method | Parameters | Description |
|---|---|---|---|
| /disks | GET | N/A | Get disk list and informations. |
| /partitions | GET | N/A | Get parttion list and infomartions. |
| /partition | GET | partitionId[int] | Get detail of specified partition. |
| /extent | GET | partitionId[int]&extentId[int] | Get extent informations. |
| /stats | GET | N/A | Get status of the datanode. |

CHAPTER 6

# Client

The client runs on each container executing application code and exposes a file system interface to applications and can access a mounted volume via a user-space file system interface such as FUSE.

## 6.1 Client Caching

The client process runs entirely in the user space with its own cache, which has been used in the following cases.

To reduce the communication with the resource manager, the client caches the addresses of the available meta and data partitions assigned to the mounted volume, which can be obtained at the startup, and periodically synchronizes this available partitions with the resource manager.

To reduce the communication with the meta nodes, the client also caches the returned inodes and dentries when creating new files, as well as the data partition id, the extent id and the offset, after the file has been written to the data node successfully. When a file is opened for read/write, the client will force the cache metadata to be synchronous with the meta node.

To reduce the communication with the data nodes, the client caches the most recently identified leader. Our observation is that, when reading a file, the client may not know which data node is the current leader because the leader could change after a failure recovery. As a result, the client may try to send the read request to each replica one by one until a leader is identified. However, since the leader does not change frequently, by caching the last identified leader, the client can have minimized number of retries in most cases.

## 6.2 Integration with FUSE

The CFS client has been integrated with FUSE to provide a file system interface in the user space. In the past, low performance is considered the main disadvantage of such user-space file systems. But over the years, FUSE has made several improvement on its performance such as multithreading and write-back cache. In the future, we plan to develop our own POSIX-compliant file system interface in the kernel space to completely eliminate the overhead from FUSE.

Currently the write-back cache feature does not work well in CFS due to the following reason. The default write behavior of FUSE is called directIO, which bypasses the kernel's page cache. This results in performance problems

on writing small files as each write pushes the file data to the user daemon. The solution FUSE implemented was to make the page cache support a write-back policy that aggregates small data first, and then make writes asynchronous. With that change, file data can be pushed to the user daemon in larger blobs at a time. However, in real production, we found that the write-back cache is not very useful, because a write operation usually invoke another process that tries to balance the dirty pages (pages in the main memory that have been modified during writing data to disk are marked as "dirty" and have to be flushed to disk before they can be freed), which incurs extra overhead. This overhead becomes more obvious when small files are continuously written through FUSE.

CHAPTER 7

# Resource Manager

The cluster contains dataNodes,metaNodes,vols,dataPartitions and metaPartitions,they are managed by master server. The master server caches the metadata in mem,persist to GoLevelDB,and ensure consistence by raft protocol. The master server manages dataPartition id to dataNode server mapping,metaPartition id to metaNode server mapping.

## 7.1 Features

- Multi-tenant, Resource Isolation
- dataNodes,metaNodes shared,vol owns dataPartition and metaPartition exclusive
- Asynchronous communication with dataNode and metaNode

## 7.2 Configurations

CFS use **JSON** as configuration file format.

Table 1: Properties

| Key | Type | Description | Mandatory |
|---|---|---|---|
| role | string | Role of process and must be set to master | Yes |
| ip | string | host ip | Yes |
| port | string | Http port which api service listen on | Yes |
| prof | string | golang pprof port | Yes |
| id | string | identy different master node | Yes |
| peers | string | the member information of raft group | Yes |
| logDir | string | Path for log file storage | Yes |
| logLevel | string | Level operation for logging. Default is *error*. | No |
| retainLogs | string | the number of raft logs will be retain. | Yes |
| walDir | string | Path for raft log file storage. | Yes |
| storeDir | string | Path for RocksDB file storage,path must be exist | Yes |
| clusterName | string | The cluster identifier | Yes |
| exporterPort | int | The prometheus exporter port | No |
| consulAddr | string | The consul register addr for prometheus exporter | No |

**Example:**

```
{
 "role": "master",
 "ip": "127.0.0.1",
 "port": "8080",
 "prof":"10088",
 "id":"1",
 "peers": "1:127.0.0.1:8080,1:127.0.0.1:8081,1:127.0.0.1:8082",
 "logDir": "/export/master",
 "logLevel":"DEBUG",
 "retainLogs":"2000",
 "walDir":"/export/raft",
 "storeDir":"/export/rocks",
 "exporterPort": 9510,
 "consulAddr": "http://consul.prometheus-cfs.local",
 "clusterName":"test"
}
```

## 7.3 Start Service

```
nohup ./master -c config.json > nohup.out &
```

# CHAPTER 8

# Meta Subsystem

Metanode is the manager of meta partitions and replicated by MultiRaft. Each metanode manages various of partitions. Each partition covers an inode range, and maintains two in-memory btrees: inode btree and dentry btree.

Table 1: Properties

| Key | Type | Description | Mandatory | |
|---|---|---|---|---|
| role | string | Role of process and must be set to *metanode* | Yes | |
| listen | string | Listen and accept port of the server | Yes | |
| prof | string | pprof port | Yes | |
| logLevel | string | Level operation for logging. Default is *error* | No | |
| metadataDir | string | metaNode store snapshot directory” | Yes | |
| logDir | string | log directory | Yes | |
| raftDir | string | raft wal directory | Yes | |
| raftHeartbeatPort | string | raft heartbeat port | Yes | |
| raftReplicaPort | string | raft replicate port | Yes | |
| consulAddr | string | Addresses of monitor system | No | |
| exporterPort | string | Port for monitor system | No | |
| masterAddrs | string | Addresses of master server | Yes | |

Example:

```
{
     "role": "metanode",
     "listen": "9021",
     "prof": "9092",
     "logLevel": "debug",
     "metadataDir": "/export/cfs/metanode_meta",
     "logDir": "/export/Logs/cfs/metanode",
     "raftDir": "/export/cfs/metanode_raft",
     "raftHeartbeatPort": "9093",
     "raftReplicaPort": "9094",
     "consulAddr": "http://consul.prometheus-cfs.local",
```

(continues on next page)

(continued from previous page)

```
    "exporterPort": 9511,
    "masterAddrs": [
        "192.168.31.173:80",
        "192.168.31.141:80",
        "192.168.30.200:80"
    ]
}
```

CHAPTER 9

# Data Subsystem

## 9.1 How To Start DataNode

Start a DataNode process by execute the server binary of CFS you built with `-c` argument and specify configuration file.

```
nohup cfs-datanode -c datanode.json &
```

## 9.2 Configurations

Table 1: Properties

| Key | Type | Description | Mandatory |
|---|---|---|---|
| role | string | Role of process and must be set to *datanode* | Yes |
| port | string | Port of TCP network to be listen | Yes |
| localIP | string | IP of network to be choose | No |
| prof | string | Port of HTTP based prof and api service | Yes |
| logDir | string | Path for log file storage | Yes |
| logLevel | string | Level operation for logging. Default is *error* | No |
| raftHeartbeat | string | Port of raft heartbeat TCP network to be listen | Yes |
| raftReplica | string | Port of raft replicate TCP network to be listen | Yes |
| raftDir | string | Path for raft log file storage | No |
| consulAddr | string | Addresses of monitor system | No |
| exporterPort | string | Port for monitor system | No |
| masterAddr | string slice | Addresses of master server | Yes |
| rack | string | Identity of rack | No |
| disks | string slice | PATH:MAX_ERRS:REST_SIZE | Yes |

**Example:**

```
{
    "role": "datanode",
    "port": "6000",
    "prof": "6001",
    "logDir": "/export/Logs/datanode",
    "logLevel": "debug",
    "raftHeartbeat": "9095",
    "raftReplica": "9096",
    "raftDir": "/export/Logs/datanode/raft",
    "consulAddr": "http://consul.prometheus-cfs.local",
    "exporterPort": 9512,
    "masterAddr": [
        "10.196.30.200:80",
        "10.196.31.141:80",
        "10.196.31.173:80"
    ],
    "rack": "main",
     "disks": [
        "/data0:107374182400",
        "/data1:107374182400"
    ]
}
```

CHAPTER 10

# Client

## 10.1 Prerequisite

Insert FUSE kernel module and install libfuse.

```
modprobe fuse
yum install -y fuse
```

## 10.2 Prepare Config File

fuse.json

```
{
  "mountPoint": "/mnt/fuse",
  "volName": "test",
  "owner": "cfs",
  "masterAddr": "192.168.31.173:80,192.168.31.141:80,192.168.30.200:80",
  "logDir": "/export/Logs/cfs",
  "logLevel": "info",
  "profPort": "10094"
}
```

Table 1: Supported Configurations

| Name | Type | Description | Mandatory |
|---|---|---|---|
| mountPoint | string | Mount point | Yes |
| volName | string | Volume name | Yes |
| owner | string | Owner name as authentication | Yes |
| masterAddr | string | Resource manager IP address | Yes |
| logDir | string | Path to store log files | No |
| logLevel | string | Log leveldebug, info, warn, error | No |
| profPort | string | Golang pprof port | No |
| exporterPort | string | Performance monitor port | No |
| consulAddr | string | Performance monitor server address | No |
| lookupValid | string | Lookup valid duration in FUSE kernel module, unit: sec | No |
| attrValid | string | Attr valid duration in FUSE kernel module, unit: sec | No |
| icacheTimeout | string | Inode cache valid duration in client | No |
| enSyncWrite | string | Enable DirectIO sync write, i.e. make sure data is fsynced in data node | No |
| autoInvalData | string | Use AutoInvalData FUSE mount option | No |

## 10.3 Mount

Use the example *fuse.json*, and client is mounted on the directory */mnt/fuse*. All operations to */mnt/fuse* would be performed on the backing distributed file system.

```
nohup ./client -c fuse.json &
```

CHAPTER 11

# Monitor

CFS use prometheus as metrics collector. It simply config as follow in mastermetanodedatanodeclient's config file

```
{
    "exporterPort": 9510,
    "consulAddr": "http://consul.prometheus-cfs.local"
}
```

- exporterPortprometheus exporter Port. when setcan export prometheus metrics from URL(http://\protect\T1\textdollarhostip:\protect\T1\textdollarexporterPort/metrics). If not set, prometheus exporter will unavailable
- consulAddr: consul register addressit can work with prometheus to auto discover deployed cfs nodes, if not set, consul register will not work.

Using grafana as prometheus metrics web front


Also, user can use prometheus alertmanager to capture and route alerts to the correct receiver. please visit prometheus alertmanger web-doc

Grafana DashBoard Config:

```
{
  "__inputs": [
    {
      "name": "DS_CFS01",
      "label": "cfs01",
      "description": "",
      "type": "datasource",
      "pluginId": "prometheus",
      "pluginName": "Prometheus"
    }
  ],
  "__requires": [
    {
      "type": "grafana",
      "id": "grafana",
      "name": "Grafana",
      "version": "5.2.4"
    },
    {
      "type": "panel",
      "id": "graph",
      "name": "Graph",
      "version": "5.0.0"
    },
    {
      "type": "datasource",
      "id": "prometheus",
      "name": "Prometheus",
      "version": "5.0.0"
```

(continues on next page)

(continued from previous page)

```
    },
    {
      "type": "panel",
      "id": "singlestat",
      "name": "Singlestat",
      "version": "5.0.0"
    }
  ],
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": "-- Grafana --",
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  "gnetId": null,
  "graphTooltip": 0,
  "id": null,
  "iteration": 1546930136099,
  "links": [
    {
      "icon": "external link",
      "tags": [],
      "targetBlank": true,
      "title": "mdc",
      "tooltip": "",
      "type": "link",
      "url": "http://mdc.jd.com/monitor/chart?ip=$hostip"
    }
  ],
  "panels": [
    {
      "gridPos": {
        "h": 1,
        "w": 24,
        "x": 0,
        "y": 0
      },
      "id": 85,
      "title": "Summary",
      "type": "row"
    },
    {
      "cacheTimeout": null,
      "colorBackground": false,
      "colorValue": false,
      "colors": [
        "#299c46",
        "rgba(237, 129, 40, 0.89)",
        "#d44a3a"
```

(continues on next page)

(continued from previous page)


```
      ],
      "datasource": "${DS_CFS01}",
      "format": "none",
      "gauge": {
        "maxValue": 100,
        "minValue": 0,
        "show": false,
        "thresholdLabels": false,
        "thresholdMarkers": true
      },
      "gridPos": {
        "h": 4,
        "w": 4,
        "x": 0,
        "y": 1
      },
      "id": 38,
      "interval": null,
      "links": [
        {
          "dashUri": "db/cfs-master",
          "dashboard": "cfs-master",
          "includeVars": false,
          "keepTime": true,
          "targetBlank": true,
          "title": "cfs-master",
          "type": "dashboard"
        }
      ],
      "mappingType": 1,
      "mappingTypes": [
        {
          "name": "value to text",
          "value": 1
        },
        {
          "name": "range to text",
          "value": 2
        }
      ],
      "maxDataPoints": 100,
      "nullPointMode": "connected",
      "nullText": null,
      "postfix": "",
      "postfixFontSize": "50%",
      "prefix": "",
      "prefixFontSize": "50%",
      "rangeMaps": [
        {
          "from": "null",
          "text": "N/A",
          "to": "null"
        }
      ],
      "sparkline": {
        "fillColor": "rgba(31, 118, 189, 0.18)",
        "full": false,
```


(continues on next page)

(continued from previous page)

```
        "lineColor": "rgb(31, 120, 193)",
        "show": true
      },
      "tableColumn": "",
      "targets": [
        {
          "expr": "count(go_info{cluster=~\"$cluster\", app=~\"$app\", role=~\"master\
↪"})",
          "format": "time_series",
          "intervalFactor": 1,
          "refId": "A"
        }
      ],
      "thresholds": "",
      "title": "master_count",
      "type": "singlestat",
      "valueFontSize": "80%",
      "valueMaps": [
        {
          "op": "=",
          "text": "N/A",
          "value": "null"
        }
      ],
      "valueName": "current"
    },
    {
      "cacheTimeout": null,
      "colorBackground": false,
      "colorValue": false,
      "colors": [
        "#299c46",
        "rgba(237, 129, 40, 0.89)",
        "#d44a3a"
      ],
      "datasource": "${DS_CFS01}",
      "format": "none",
      "gauge": {
        "maxValue": 100,
        "minValue": 0,
        "show": false,
        "thresholdLabels": false,
        "thresholdMarkers": true
      },
      "gridPos": {
        "h": 4,
        "w": 4,
        "x": 4,
        "y": 1
      },
      "id": 42,
      "interval": null,
      "links": [
        {
          "dashUri": "db/cfs-metanode",
          "dashboard": "cfs-metanode",
          "includeVars": false,
```

(continues on next page)

(continued from previous page)

```
          "keepTime": true,
          "targetBlank": true,
          "title": "cfs-metanode",
          "type": "dashboard"
        }
      ],
      "mappingType": 1,
      "mappingTypes": [
        {
          "name": "value to text",
          "value": 1
        },
        {
          "name": "range to text",
          "value": 2
        }
      ],
      "maxDataPoints": 100,
      "nullPointMode": "connected",
      "nullText": null,
      "postfix": "",
      "postfixFontSize": "50%",
      "prefix": "",
      "prefixFontSize": "50%",
      "rangeMaps": [
        {
          "from": "null",
          "text": "N/A",
          "to": "null"
        }
      ],
      "sparkline": {
        "fillColor": "rgba(31, 118, 189, 0.18)",
        "full": false,
        "lineColor": "rgb(31, 120, 193)",
        "show": true
      },
      "tableColumn": "",
      "targets": [
        {
          "expr": "count(go_info{cluster=~\"$cluster\", app=~\"$app\", role=~\
↪"metanode\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "refId": "A"
        }
      ],
      "thresholds": "",
      "title": "metanode_count",
      "type": "singlestat",
      "valueFontSize": "80%",
      "valueMaps": [
        {
          "op": "=",
          "text": "N/A",
          "value": "null"
        }
```

(continues on next page)

(continued from previous page)


```
      ],
      "valueName": "current"
    },
    {
      "cacheTimeout": null,
      "colorBackground": false,
      "colorValue": false,
      "colors": [
        "#299c46",
        "rgba(237, 129, 40, 0.89)",
        "#d44a3a"
      ],
      "datasource": "${DS_CFS01}",
      "format": "none",
      "gauge": {
        "maxValue": 100,
        "minValue": 0,
        "show": false,
        "thresholdLabels": false,
        "thresholdMarkers": true
      },
      "gridPos": {
        "h": 4,
        "w": 4,
        "x": 8,
        "y": 1
      },
      "id": 41,
      "interval": null,
      "links": [
        {
          "dashUri": "db/cfs-datanode",
          "dashboard": "cfs-datanode",
          "includeVars": false,
          "keepTime": true,
          "targetBlank": true,
          "title": "cfs-datanode",
          "type": "dashboard"
        }
      ],
      "mappingType": 1,
      "mappingTypes": [
        {
          "name": "value to text",
          "value": 1
        },
        {
          "name": "range to text",
          "value": 2
        }
      ],
      "maxDataPoints": 100,
      "nullPointMode": "connected",
      "nullText": null,
      "postfix": "",
      "postfixFontSize": "50%",
      "prefix": "",
```


(continues on next page)

(continued from previous page)

```
      "prefixFontSize": "50%",
      "rangeMaps": [
        {
          "from": "null",
          "text": "N/A",
          "to": "null"
        }
      ],
      "sparkline": {
        "fillColor": "rgba(31, 118, 189, 0.18)",
        "full": false,
        "lineColor": "rgb(31, 120, 193)",
        "show": true
      },
      "tableColumn": "",
      "targets": [
        {
          "expr": "count(go_info{cluster=~\"$cluster\", app=~\"$app\", role=~\
↪"dataNode\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "refId": "A"
        }
      ],
      "thresholds": "",
      "title": "datanode_count",
      "type": "singlestat",
      "valueFontSize": "80%",
      "valueMaps": [
        {
          "op": "=",
          "text": "N/A",
          "value": "null"
        }
      ],
      "valueName": "current"
    },
    {
      "gridPos": {
        "h": 1,
        "w": 24,
        "x": 0,
        "y": 5
      },
      "id": 40,
      "title": "Cluster",
      "type": "row"
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 6,
```

(continues on next page)

(continued from previous page)


```
      "w": 7,
      "x": 0,
      "y": 6
    },
    "id": 70,
    "legend": {
      "avg": false,
      "current": false,
      "max": false,
      "min": false,
      "show": true,
      "total": false,
      "values": false
    },
    "lines": true,
    "linewidth": 1,
    "links": [],
    "nullPointMode": "null",
    "percentage": false,
    "pointradius": 5,
    "points": false,
    "renderer": "flot",
    "seriesOverrides": [],
    "spaceLength": 10,
    "stack": false,
    "steppedLine": false,
    "targets": [
      {
        "expr": "sum(cfs_metanode_OpCreateMetaPartition{cluster=~\"$cluster\"})",
        "format": "time_series",
        "intervalFactor": 1,
        "legendFormat": "Create",
        "refId": "A"
      },
      {
        "expr": "sum(cfs_metanode_OpLoadMetaPartition{cluster=~\"$cluster\"})",
        "format": "time_series",
        "intervalFactor": 1,
        "legendFormat": "Load",
        "refId": "B"
      }
    ],
    "thresholds": [],
    "timeFrom": null,
    "timeShift": null,
    "title": "metanode_OpMetaPartition",
    "tooltip": {
      "shared": true,
      "sort": 0,
      "value_type": "individual"
    },
    "type": "graph",
    "xaxis": {
      "buckets": null,
      "mode": "time",
      "name": null,
      "show": true,
```


(continues on next page)

(continued from previous page)

```
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 6,
        "w": 7,
        "x": 7,
        "y": 6
      },
      "id": 71,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
```

(continues on next page)

(continued from previous page)

```
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "sum(cfs_metanode_OpMetaBatchInodeGet{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "BatchGet",
          "refId": "A"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaCreateInode{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Create",
          "refId": "B"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaDeleteInode{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Delete",
          "refId": "C"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaEvictInode{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Evict",
          "refId": "D"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaInodeGet{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Get",
          "refId": "E"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaLinkInode{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Link",
          "refId": "F"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "metanode_OpMetaInode",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
```

(continues on next page)

(continued from previous page)

```
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 6,
        "w": 6,
        "x": 14,
        "y": 6
      },
      "id": 45,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
```

(continues on next page)

(continued from previous page)

```
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "sum(cfs_metanode_OpMetaCreateDentry{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Create",
          "refId": "A"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaDeleteDentry{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Delete",
          "refId": "B"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaUpdateDentry{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Update",
          "refId": "C"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "metanode_OpMetaDentry",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
```

(continues on next page)

(continued from previous page)

```
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 6,
        "w": 7,
        "x": 0,
        "y": 12
      },
      "id": 79,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_CreateFile{cluster=~\"
↪$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "CreateFile",
          "refId": "A"
```

(continues on next page)

(continued from previous page)

```
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_MarkDelete{cluster=~\"
↪$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "MarkDelete",
          "refId": "B"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_Read{cluster=~\"$cluster\"})
↪",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Read",
          "refId": "C"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_Write{cluster=~\"$cluster\"})
↪",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Write",
          "refId": "D"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_RandomWrite{cluster=~\"
↪$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "RandomWrite",
          "refId": "E"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "datanode_CreateFile",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
```

(continues on next page)

(continued from previous page)

```
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 6,
        "w": 7,
        "x": 7,
        "y": 12
      },
      "id": 75,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_OpLoadDataPartition{cluster=~
↪\"$cluster\"})",
          "format": "time_series",
```

(continues on next page)

(continued from previous page)

```
          "intervalFactor": 1,
          "legendFormat": "OpLoadDataPartition",
          "refId": "G"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_OpDataNodeHeartbeat{cluster=~
↪\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "OpDataNodeHeartbeat",
          "refId": "F"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_OpGetPartitionSize{cluster=~\
↪"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "OpGetPartitionSize",
          "refId": "H"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_OpGetAppliedId{cluster=~\"
↪$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "OpGetAppliedId",
          "refId": "I"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_OpCreateDataPartition
↪{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "OpCreateDataPartition",
          "refId": "A"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "datanode_Op",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
```

(continues on next page)

(continued from previous page)

```
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 6,
        "w": 6,
        "x": 14,
        "y": 12
      },
      "id": 73,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
```

(continues on next page)

(continued from previous page)

```
          "expr": "sum(cfs_metanode_OpMetaOpen{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Open",
          "refId": "A"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaLookup{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Lookup",
          "refId": "B"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaNodeHeartbeat{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "NodeHeartbeat",
          "refId": "C"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaReadDir{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "ReadDir",
          "refId": "D"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaReleaseOpen{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "ReleaseOpen",
          "refId": "E"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaSetattr{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Setattr",
          "refId": "F"
        },
        {
          "expr": "sum(cfs_metanode_OpMetaTruncate{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Truncate",
          "refId": "G"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "metanode_OpMeta",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
```

(continues on next page)

(continued from previous page)

```
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 7,
        "w": 7,
        "x": 0,
        "y": 18
      },
      "id": 80,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
```

(continues on next page)

(continued from previous page)

```
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_ExtentRepairRead{cluster=~\"
↪$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "ExtentRepairRead",
          "refId": "B"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_GetAllExtentWatermark
↪{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "GetAllExtentWatermark",
          "refId": "C"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_NotifyExtentRepair{cluster=~\
↪"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "NotifyExtentRepair",
          "refId": "D"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "datanode_Extent",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
```

(continues on next page)

(continued from previous page)

```
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 7,
        "w": 7,
        "x": 7,
        "y": 18
      },
      "id": 83,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_streamRead{cluster=~\"
↪$cluster\"})",
```

(continues on next page)

(continued from previous page)

```
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamRead",
          "refId": "M"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_streamWrite{cluster=~\"
↪$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamWrite",
          "refId": "N"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_streamCreateFile{cluster=~\"
↪$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamCreateFile",
          "refId": "A"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_streamExtentRepairRead
↪{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamExtentRepairRead",
          "refId": "B"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_streamMarkDelete{cluster=~\"
↪$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamMarkDelete",
          "refId": "C"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_streamOpGetAppliedId
↪{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamOpGetAppliedId",
          "refId": "D"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_streamOpGetPartitionSize
↪{cluster=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamOpGetPartitionSize",
          "refId": "E"
        },
        {
          "expr": "sum(cfs_dataNode_[[cluster]]_datanode_streamNotifyExtentRepair
↪{cluster=~\"$cluster\"})",
          "format": "time_series",
```

(continues on next page)

(continued from previous page)


```
          "intervalFactor": 1,
          "legendFormat": "streamNotifyExtentRepair",
          "refId": "F"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "datanode_Stream",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "collapsed": false,
      "gridPos": {
        "h": 1,
        "w": 24,
        "x": 0,
        "y": 25
      },
      "id": 60,
      "panels": [],
      "title": "GoRuntime",
      "type": "row"
```


(continues on next page)

(continued from previous page)

```
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 6,
        "w": 7,
        "x": 0,
        "y": 26
      },
      "id": 61,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "go_goroutines{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "go_goroutines",
          "refId": "A"
        },
        {
          "expr": "go_threads{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "go_threads",
          "refId": "B"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "go info",
      "tooltip": {
```

(continues on next page)

(continued from previous page)

```
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "locale",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 6,
        "w": 7,
        "x": 7,
        "y": 26
      },
      "id": 62,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
```

(continues on next page)

(continued from previous page)

```
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "go_memstats_alloc_bytes{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "alloc_bytes",
          "refId": "A"
        },
        {
          "expr": "go_memstats_alloc_bytes_total{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "alloc_bytes_total",
          "refId": "B"
        },
        {
          "expr": "go_memstats_heap_alloc_bytes{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "heap_alloc_bytes",
          "refId": "C"
        },
        {
          "expr": "go_memstats_heap_inuse_bytes{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "heap_inuse_bytes",
          "refId": "D"
        },
        {
          "expr": "go_memstats_sys_bytes{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "sys_bytes",
          "refId": "E"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "go_memstats",
      "tooltip": {
        "shared": true,
        "sort": 0,
```

(continues on next page)

(continued from previous page)

```
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "decbytes",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 6,
        "w": 7,
        "x": 14,
        "y": 26
      },
      "id": 63,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
```

(continues on next page)

(continued from previous page)

```
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [
        {
          "alias": "gc_rate",
          "yaxis": 2
        }
      ],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "go_gc_duration_seconds{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "seconds_{{quantile}}",
          "refId": "A"
        },
        {
          "expr": "rate(go_gc_duration_seconds_count{instance=~\"$instance\"}[1m])",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "gc_rate",
          "refId": "B"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "gc_duration",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "s",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
```

(continues on next page)

(continued from previous page)

```
        },
        {
          "format": "locale",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "collapsed": false,
      "gridPos": {
        "h": 1,
        "w": 24,
        "x": 0,
        "y": 32
      },
      "id": 34,
      "panels": [],
      "title": "Master",
      "type": "row"
    },
    {
      "collapsed": false,
      "gridPos": {
        "h": 1,
        "w": 24,
        "x": 0,
        "y": 33
      },
      "id": 36,
      "panels": [],
      "title": "Metanode",
      "type": "row"
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 0,
        "y": 34
      },
      "id": 58,
      "legend": {
        "avg": false,
```

(continues on next page)

(continued from previous page)

```
      "current": false,
      "max": false,
      "min": false,
      "show": true,
      "total": false,
      "values": false
    },
    "lines": true,
    "linewidth": 1,
    "links": [],
    "nullPointMode": "null",
    "percentage": false,
    "pointradius": 5,
    "points": false,
    "renderer": "flot",
    "seriesOverrides": [],
    "spaceLength": 10,
    "stack": false,
    "steppedLine": false,
    "targets": [
      {
        "expr": "cfs_metanode_OpCreateMetaPartition{instance=~\"$instance\"}",
        "format": "time_series",
        "intervalFactor": 1,
        "legendFormat": "Create",
        "refId": "A"
      },
      {
        "expr": "cfs_metanode_OpLoadMetaPartition{instance=~\"$instance\"}",
        "format": "time_series",
        "intervalFactor": 1,
        "legendFormat": "Load",
        "refId": "B"
      }
    ],
    "thresholds": [],
    "timeFrom": null,
    "timeShift": null,
    "title": "metanode_OpMetaPartition",
    "tooltip": {
      "shared": true,
      "sort": 0,
      "value_type": "individual"
    },
    "type": "graph",
    "xaxis": {
      "buckets": null,
      "mode": "time",
      "name": null,
      "show": true,
      "values": []
    },
    "yaxes": [
      {
        "format": "ns",
        "label": null,
        "logBase": 1,
```

(continues on next page)

(continued from previous page)

```
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 8,
        "y": 34
      },
      "id": 44,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "cfs_metanode_OpMetaBatchInodeGet{instance=~\"$instance\"}",
          "format": "time_series",
```

(continues on next page)

(continued from previous page)

```
          "intervalFactor": 1,
          "legendFormat": "BatchGet",
          "refId": "A"
        },
        {
          "expr": "cfs_metanode_OpMetaCreateInode{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Create",
          "refId": "B"
        },
        {
          "expr": "cfs_metanode_OpMetaDeleteInode{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Delete",
          "refId": "C"
        },
        {
          "expr": "cfs_metanode_OpMetaEvictInode{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Evict",
          "refId": "D"
        },
        {
          "expr": "cfs_metanode_OpMetaInodeGet{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Get",
          "refId": "E"
        },
        {
          "expr": "cfs_metanode_OpMetaLinkInode{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Link",
          "refId": "F"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "metanode_OpMetaInode",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
```

(continues on next page)

(continued from previous page)


```
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 16,
        "y": 34
      },
      "id": 72,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
```


(continues on next page)

(continued from previous page)

```
      "steppedLine": false,
      "targets": [
        {
          "expr": "cfs_metanode_OpMetaCreateDentry{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Create",
          "refId": "A"
        },
        {
          "expr": "cfs_metanode_OpMetaDeleteDentry{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Delete",
          "refId": "B"
        },
        {
          "expr": "cfs_metanode_OpMetaUpdateDentry{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Update",
          "refId": "C"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "metanode_OpMetaDentry",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
```

(continues on next page)

(continued from previous page)

```
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 0,
        "y": 42
      },
      "id": 46,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "cfs_metanode_OpMetaOpen{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Open",
          "refId": "A"
        },
        {
          "expr": "cfs_metanode_OpMetaLookup{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Lookup",
          "refId": "B"
        },
```

(continues on next page)

(continued from previous page)


```
        {
          "expr": "cfs_metanode_OpMetaNodeHeartbeat{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "NodeHeartbeat",
          "refId": "C"
        },
        {
          "expr": "cfs_metanode_OpMetaReadDir{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "ReadDir",
          "refId": "D"
        },
        {
          "expr": "cfs_metanode_OpMetaReleaseOpen{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "ReleaseOpen",
          "refId": "E"
        },
        {
          "expr": "cfs_metanode_OpMetaSetattr{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Setattr",
          "refId": "F"
        },
        {
          "expr": "cfs_metanode_OpMetaTruncate{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Truncate",
          "refId": "G"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "metanode_OpMeta",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
```


(continues on next page)

(continued from previous page)

```
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 8,
        "y": 42
      },
      "id": 50,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "cfs_metanode_OpMetaExtentsAdd{instance=~\"$instance\"}",
```

(continues on next page)

(continued from previous page)


```
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Add",
          "refId": "A"
        },
        {
          "expr": "cfs_metanode_OpMetaExtentsList{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "List",
          "refId": "B"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "metanode_OpMetaExtents",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "collapsed": false,
      "gridPos": {
        "h": 1,
```


(continues on next page)

(continued from previous page)

```
        "w": 24,
        "x": 0,
        "y": 50
      },
      "id": 27,
      "panels": [],
      "title": "Datanode",
      "type": "row"
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 0,
        "y": 51
      },
      "id": 28,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_CreateFile{instance=~\"$instance\
↪"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "CreateFile",
          "refId": "A"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_MarkDelete{instance=~\"$instance\
↪"}",
          "format": "time_series",
```

(continues on next page)

(continued from previous page)


```
          "intervalFactor": 1,
          "legendFormat": "MarkDelete",
          "refId": "B"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "datanode_CreateFile",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 8,
        "y": 51
```


(continues on next page)

(continued from previous page)

```
      },
      "id": 74,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_ExtentRepairRead{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "ExtentRepairRead",
          "refId": "B"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_GetAllExtentWatermark{instance=~\
↪"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "GetAllExtentWatermark",
          "refId": "C"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_NotifyExtentRepair{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "NotifyExtentRepair",
          "refId": "D"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "datanode_Extent",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
```

(continues on next page)

(continued from previous page)

```
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 16,
        "y": 51
      },
      "id": 81,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
```

(continues on next page)

(continued from previous page)

```
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_OpLoadDataPartition{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "OpLoadDataPartition",
          "refId": "G"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_OpDataNodeHeartbeat{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "OpDataNodeHeartbeat",
          "refId": "F"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_OpGetPartitionSize{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "OpGetPartitionSize",
          "refId": "H"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_OpGetAppliedId{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "OpGetAppliedId",
          "refId": "I"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_OpCreateDataPartition{instance=~\
↪"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "OpCreateDataPartition",
          "refId": "A"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "datanode_Op",
      "tooltip": {
```

(continues on next page)

(continued from previous page)

```
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 0,
        "y": 59
      },
      "id": 76,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
```

(continues on next page)

(continued from previous page)

```
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_Read{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Read",
          "refId": "J"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_Write{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Write",
          "refId": "K"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_RandomWrite{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "RandomWrite",
          "refId": "L"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "datanode_IO",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
```

(continues on next page)

(continued from previous page)

```
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 8,
        "y": 59
      },
      "id": 77,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
```

(continues on next page)

(continued from previous page)

```
          "expr": "cfs_dataNode_[[cluster]]_datanode_streamRead{instance=~\"$instance\
↪"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamRead",
          "refId": "M"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_streamWrite{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamWrite",
          "refId": "N"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_streamCreateFile{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamCreateFile",
          "refId": "A"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_streamExtentRepairRead{instance=~
↪\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamExtentRepairRead",
          "refId": "B"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_streamMarkDelete{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamMarkDelete",
          "refId": "C"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_streamOpGetAppliedId{instance=~\"
↪$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamOpGetAppliedId",
          "refId": "D"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_streamOpGetPartitionSize
↪{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamOpGetPartitionSize",
          "refId": "E"
        },
        {
          "expr": "cfs_dataNode_[[cluster]]_datanode_streamNotifyExtentRepair
↪{instance=~\"$instance\"}",
```

(continues on next page)

(continued from previous page)


```
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "streamNotifyExtentRepair",
          "refId": "F"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "datanode_Stream",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "ns",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "collapsed": true,
      "gridPos": {
        "h": 1,
        "w": 24,
        "x": 0,
        "y": 67
      },
      "id": 66,
      "panels": [
        {
```


(continues on next page)

(continued from previous page)

```
        "aliasColors": {},
        "bars": false,
        "dashLength": 10,
        "dashes": false,
        "datasource": "${DS_CFS01}",
        "fill": 1,
        "gridPos": {
          "h": 6,
          "w": 7,
          "x": 0,
          "y": 85
        },
        "id": 64,
        "legend": {
          "avg": false,
          "current": false,
          "max": false,
          "min": false,
          "show": true,
          "total": false,
          "values": false
        },
        "lines": true,
        "linewidth": 1,
        "links": [],
        "nullPointMode": "null",
        "percentage": false,
        "pointradius": 5,
        "points": false,
        "renderer": "flot",
        "seriesOverrides": [],
        "spaceLength": 10,
        "stack": false,
        "steppedLine": false,
        "targets": [
          {
            "expr": "cfs_fuseclient_OpMetaOpen{instance=~\"$instance\"}",
            "format": "time_series",
            "intervalFactor": 1,
            "legendFormat": "Open",
            "refId": "B"
          },
          {
            "expr": "cfs_fuseclient_OpMetaExtentsAdd{instance=~\"$instance\"}",
            "format": "time_series",
            "intervalFactor": 1,
            "legendFormat": "ExtentsAdd",
            "refId": "H"
          },
          {
            "expr": "cfs_fuseclient_OpMetaExtentsList{instance=~\"$instance\"}",
            "format": "time_series",
            "intervalFactor": 1,
            "legendFormat": "ExtentsList",
            "refId": "I"
          },
          {
```

(continues on next page)

(continued from previous page)


```
            "expr": "cfs_fuseclient_OpMetaReadDir{instance=~\"$instance\"}",
            "format": "time_series",
            "intervalFactor": 1,
            "legendFormat": "ReadDir",
            "refId": "K"
          },
          {
            "expr": "cfs_fuseclient_OpMetaSetattr{instance=~\"$instance\"}",
            "format": "time_series",
            "intervalFactor": 1,
            "legendFormat": "Setattr",
            "refId": "L"
          }
        ],
        "thresholds": [],
        "timeFrom": null,
        "timeShift": null,
        "title": "fuseclient_OpMeta",
        "tooltip": {
          "shared": true,
          "sort": 0,
          "value_type": "individual"
        },
        "type": "graph",
        "xaxis": {
          "buckets": null,
          "mode": "time",
          "name": null,
          "show": true,
          "values": []
        },
        "yaxes": [
          {
            "format": "ns",
            "label": null,
            "logBase": 1,
            "max": null,
            "min": null,
            "show": true
          },
          {
            "format": "short",
            "label": null,
            "logBase": 1,
            "max": null,
            "min": null,
            "show": true
          }
        ],
        "yaxis": {
          "align": false,
          "alignLevel": null
        }
      },
      {
        "aliasColors": {},
        "bars": false,
```


(continues on next page)

(continued from previous page)


```
            "dashLength": 10,
            "dashes": false,
            "datasource": "${DS_CFS01}",
            "fill": 1,
            "gridPos": {
              "h": 6,
              "w": 7,
              "x": 7,
              "y": 85
            },
            "id": 67,
            "legend": {
              "avg": false,
              "current": false,
              "max": false,
              "min": false,
              "show": true,
              "total": false,
              "values": false
            },
            "lines": true,
            "linewidth": 1,
            "links": [],
            "nullPointMode": "null",
            "percentage": false,
            "pointradius": 5,
            "points": false,
            "renderer": "flot",
            "seriesOverrides": [],
            "spaceLength": 10,
            "stack": false,
            "steppedLine": false,
            "targets": [
              {
                "expr": "cfs_fuseclient_OpMetaBatchInodeGet{instance=~\"$instance\"}",
                "format": "time_series",
                "intervalFactor": 1,
                "legendFormat": "BatchInodeGet",
                "refId": "A"
              },
              {
                "expr": "cfs_fuseclient_OpMetaCreateInode{instance=~\"$instance\"}",
                "format": "time_series",
                "intervalFactor": 1,
                "legendFormat": "CreateInode",
                "refId": "D"
              },
              {
                "expr": "cfs_fuseclient_OpMetaDeleteInode{instance=~\"$instance\"}",
                "format": "time_series",
                "intervalFactor": 1,
                "legendFormat": "DeleteInode",
                "refId": "F"
              },
              {
                "expr": "cfs_fuseclient_OpMetaEvictInode{instance=~\"$instance\"}",
                "format": "time_series",
```


(continues on next page)

(continued from previous page)


```
            "intervalFactor": 1,
            "legendFormat": "EvictInode",
            "refId": "G"
          },
          {
            "expr": "cfs_fuseclient_OpMetaInodeGet{instance=~\"$instance\"}",
            "format": "time_series",
            "intervalFactor": 1,
            "legendFormat": "InodeGet",
            "refId": "J"
          }
        ],
        "thresholds": [],
        "timeFrom": null,
        "timeShift": null,
        "title": "fuseclient_OpMetaInode",
        "tooltip": {
          "shared": true,
          "sort": 0,
          "value_type": "individual"
        },
        "type": "graph",
        "xaxis": {
          "buckets": null,
          "mode": "time",
          "name": null,
          "show": true,
          "values": []
        },
        "yaxes": [
          {
            "format": "ns",
            "label": null,
            "logBase": 1,
            "max": null,
            "min": null,
            "show": true
          },
          {
            "format": "short",
            "label": null,
            "logBase": 1,
            "max": null,
            "min": null,
            "show": true
          }
        ],
        "yaxis": {
          "align": false,
          "alignLevel": null
        }
      },
      {
        "aliasColors": {},
        "bars": false,
        "dashLength": 10,
        "dashes": false,
```


(continues on next page)

(continued from previous page)


```
      "datasource": "${DS_CFS01}",
      "fill": 1,
      "gridPos": {
        "h": 6,
        "w": 7,
        "x": 14,
        "y": 85
      },
      "id": 68,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "cfs_fuseclient_OpMetaCreateDentry{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Create",
          "refId": "C"
        },
        {
          "expr": "cfs_fuseclient_OpMetaDeleteDentry{instance=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Delete",
          "refId": "E"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "fuseclient_OpMetaDentry",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
```


(continues on next page)

(continued from previous page)


```
            "buckets": null,
            "mode": "time",
            "name": null,
            "show": true,
            "values": []
          },
          "yaxes": [
            {
              "format": "ns",
              "label": null,
              "logBase": 1,
              "max": null,
              "min": null,
              "show": true
            },
            {
              "format": "short",
              "label": null,
              "logBase": 1,
              "max": null,
              "min": null,
              "show": true
            }
          ],
          "yaxis": {
            "align": false,
            "alignLevel": null
          }
        },
        {
          "aliasColors": {},
          "bars": false,
          "dashLength": 10,
          "dashes": false,
          "datasource": "${DS_CFS01}",
          "fill": 1,
          "gridPos": {
            "h": 6,
            "w": 7,
            "x": 0,
            "y": 91
          },
          "id": 69,
          "legend": {
            "avg": false,
            "current": false,
            "max": false,
            "min": false,
            "show": true,
            "total": false,
            "values": false
          },
          "lines": true,
          "linewidth": 1,
          "links": [],
          "nullPointMode": "null",
          "percentage": false,
```


(continues on next page)

(continued from previous page)

```
            "pointradius": 5,
            "points": false,
            "renderer": "flot",
            "seriesOverrides": [],
            "spaceLength": 10,
            "stack": false,
            "steppedLine": false,
            "targets": [
              {
                "expr": "cfs_fuseclient_OpMetaExtentsAdd{instance=~\"$instance\"}",
                "format": "time_series",
                "intervalFactor": 1,
                "legendFormat": "ExtentsAdd",
                "refId": "H"
              },
              {
                "expr": "cfs_fuseclient_OpMetaExtentsList{instance=~\"$instance\"}",
                "format": "time_series",
                "intervalFactor": 1,
                "legendFormat": "ExtentsList",
                "refId": "I"
              }
            ],
            "thresholds": [],
            "timeFrom": null,
            "timeShift": null,
            "title": "fuseclient_OpMetaExtent",
            "tooltip": {
              "shared": true,
              "sort": 0,
              "value_type": "individual"
            },
            "type": "graph",
            "xaxis": {
              "buckets": null,
              "mode": "time",
              "name": null,
              "show": true,
              "values": []
            },
            "yaxes": [
              {
                "format": "ns",
                "label": null,
                "logBase": 1,
                "max": null,
                "min": null,
                "show": true
              },
              {
                "format": "short",
                "label": null,
                "logBase": 1,
                "max": null,
                "min": null,
                "show": true
              }
```

(continues on next page)

(continued from previous page)

```
          ],
          "yaxis": {
            "align": false,
            "alignLevel": null
          }
        }
      ],
      "title": "FuseClient",
      "type": "row"
    }
  ],
  "refresh": false,
  "schemaVersion": 16,
  "style": "dark",
  "tags": [],
  "templating": {
    "list": [
      {
        "allValue": null,
        "current": {
          "selected": true,
          "text": "cfs",
          "value": "cfs"
        },
        "hide": 2,
        "includeAll": false,
        "label": "App",
        "multi": false,
        "name": "app",
        "options": [
          {
            "selected": true,
            "text": "cfs",
            "value": "cfs"
          }
        ],
        "query": "cfs",
        "type": "custom"
      },
      {
        "allValue": null,
        "current": {},
        "datasource": "${DS_CFS01}",
        "hide": 0,
        "includeAll": false,
        "label": "Cluster",
        "multi": false,
        "name": "cluster",
        "options": [],
        "query": "label_values(go_info{app=~\"$app\"}, cluster)",
        "refresh": 1,
        "regex": "",
        "sort": 0,
        "tagValuesQuery": "",
        "tags": [],
        "tagsQuery": "",
        "type": "query",
```

(continues on next page)

(continued from previous page)

```
        "useTags": false
      },
      {
        "allValue": null,
        "current": {},
        "datasource": "${DS_CFS01}",
        "hide": 0,
        "includeAll": false,
        "label": "Role",
        "multi": false,
        "name": "role",
        "options": [],
        "query": "label_values(go_info{app=~\"$app\", cluster=~\"$cluster\"}, role)",
        "refresh": 1,
        "regex": "",
        "sort": 0,
        "tagValuesQuery": "",
        "tags": [],
        "tagsQuery": "",
        "type": "query",
        "useTags": false
      },
      {
        "allValue": null,
        "current": {},
        "datasource": "${DS_CFS01}",
        "hide": 0,
        "includeAll": false,
        "label": "Instance",
        "multi": false,
        "name": "instance",
        "options": [],
        "query": "label_values(go_info{app=~\"$app\", role=~\"$role\", cluster=~\"
↪$cluster\"}, instance)",
        "refresh": 1,
        "regex": "",
        "sort": 0,
        "tagValuesQuery": "",
        "tags": [],
        "tagsQuery": "",
        "type": "query",
        "useTags": false
      },
      {
        "allValue": null,
        "current": {},
        "datasource": "${DS_CFS01}",
        "hide": 2,
        "includeAll": false,
        "label": "Host",
        "multi": false,
        "name": "hostip",
        "options": [],
        "query": "label_values(go_info{instance=~\"$instance\", cluster=~\"$cluster\"}
↪, instance)",
        "refresh": 1,
        "regex": "/([^:]+):.*/",
```

(continues on next page)

(continued from previous page)


```
        "sort": 0,
        "tagValuesQuery": "",
        "tags": [],
        "tagsQuery": "",
        "type": "query",
        "useTags": false
      }
    ]
  },
  "time": {
    "from": "now-1h",
    "to": "now"
  },
  "timepicker": {
    "refresh_intervals": [
      "5s",
      "10s",
      "30s",
      "1m",
      "5m",
      "15m",
      "30m",
      "1h",
      "2h",
      "1d"
    ],
    "time_options": [
      "5m",
      "15m",
      "1h",
      "6h",
      "12h",
      "24h",
      "2d",
      "7d",
      "30d"
    ]
  },
  "timezone": "",
  "title": "cfs-cluster",
  "uid": "tu_qnbsmk",
  "version": 140
}
```

CHAPTER 12

# Tune FUSE Performance

## 12.1 Fetch Linux kernel source code

Download the corresponding src rpm, and use the following commands to install source code.

```
rpm -i kernel-3.10.0-327.28.3.el7.src.rpm 2>&1 | grep -v exist
cd ~/rpmbuild/SPECS
rpmbuild -bp --target=$(uname -m) kernel.spec
```

The source code will be installed in `~/rpmbuild/BUILD/`

## 12.2 Optimize FUSE linux kernel module

In order to achieve maximum throughput performance, several FUSE kernel parameters have to be modified, such as `FUSE_MAX_PAGES_PER_REQ` and `FUSE_DEFAULT_MAX_BACKGROUND`.

Update source code according to the following lines.

```
/* fs/fuse/fuse_i.h */
#define FUSE_MAX_PAGES_PER_REQ 256

/* fs/fuse/inode.c */
#define FUSE_DEFAULT_MAX_BACKGROUND 32
```

## 12.3 Build against current running Linux kernel

```
yum install kernel-devel-3.10.0-327.28.3.el7.x86_64
```

(continues on next page)

(continued from previous page)

```
cd ~/rpmbuild/BUILD/kernel-3.10.0-327.28.3.el7/linux-3.10.0-327.28.3.el7.x86_64/fs/
↪fuse
make -C /lib/modules/`uname -r`/build M=$PWD
```

## 12.4 Install kernel module

```
cp fuse.ko /lib/modules/`uname -r`/kernel/fs/fuse

rmmod fuse
depmod -a
modprobe fuse
```

# CHAPTER 13

# Resource Manager API

## 13.1 Cluster

### 13.1.1 Overview

```
curl -v "http://127.0.0.1/admin/getCluster" | python -m json.tool
```

display the base information of the cluster, such as the detail of metaNode,dataNode,vol and so on.

response


```
{
    "Name": "test",
    "LeaderAddr": "127.0.0.1:80",
    "DisableAutoAlloc": false,
    "Applied": 225,
    "MaxDataPartitionID": 100,
    "MaxMetaNodeID": 3,
    "MaxMetaPartitionID": 1,
    "DataNodeStat": {},
    "MetaNodeStat": {},
    "VolStat": {},
    "MetaNodes": {},
    "DataNodes": {}
}
```


### 13.1.2 Freeze

```
curl -v "http://127.0.0.1/cluster/freeze?enable=true"
```

if cluster is freezed,the vol never allocates dataPartitions automatically

Table 1: Parameters

| Parameter | Type | Description |
|---|---|---|
| enable | bool | if enable is true,the cluster is freezed |

## 13.2 Metanode Related

### 13.2.1 GET

```
curl -v "http://127.0.0.1/metaNode/get?addr=127.0.0.1:9021"  | python -m json.tool
```

show the base information of the metaNode,such as addr,total memory,used memory and so on.

Table 2: Parameters

| Parameter | Type | Description |
|---|---|---|
| addr | string | the addr which communicate with master |

response


```
{
    "ID": 3,
    "Addr": "127.0.0.1:9021",
    "IsActive": true,
    "Sender": {
        "TaskMap": {}
    },
    "Rack": "",
    "MaxMemAvailWeight": 66556215048,
    "TotalWeight": 67132641280,
    "UsedWeight": 576426232,
    "Ratio": 0.008586377967698518,
    "SelectCount": 0,
    "Carry": 0.6645600532184904,
    "Threshold": 0.75,
    "ReportTime": "2018-12-05T17:26:28.29309577+08:00",
    "MetaPartitionCount": 1
}
```


### 13.2.2 Decommission

```
curl -v "http://127.0.0.1/metaNode/decommission?addr=127.0.0.1:9021"
```

remove the metaNode from cluster, meta partitions which locate the metaNode will be migrate other available metaNode asynchronous

Table 3: Parameters

| Parameter | Type | Description |
|---|---|---|
| addr | string | the addr which communicate with master |

### 13.2.3 Threshold

```
curl -v "http://127.0.0.1/threshold/set?threshold=0.75"
```

the used memory percent arrives the threshold,the status of the meta partitions which locate the metaNode will be read only

Table 4: Parameters

| Parameter | Type | Description |
|---|---|---|
| threshold | float64 | the max percent of memory which metaNode can use |

## 13.3 Datanode Related

### 13.3.1 GET

```
curl -v "http://127.0.0.1/dataNode/get?addr=127.0.0.1:5000"  | python -m json.tool
```

show the base information of the dataNode,such as addr,disk total size,disk used size and so on.

Table 5: Parameters

| Parameter | Type | Description |
|---|---|---|
| addr | string | the addr which communicate with master |

response


```
{
    "MaxDiskAvailWeight": 3708923232256,
    "CreatedVolWeights": 2705829396480,
    "RemainWeightsForCreateVol": 36960383303680,
    "TotalWeight": 39666212700160,
    "UsedWeight": 2438143586304,
    "Available": 37228069113856,
    "Rack": "rack1",
    "Addr": "10.196.30.231:6000",
    "ReportTime": "2018-12-06T10:56:38.881784447+08:00",
    "Ratio": 0.06146650815226848,
    "SelectCount": 5,
    "Carry": 1.0655859145960367,
    "Sender": {
        "TaskMap": {}
    },
    "DataPartitionCount": 21
}
```


### 13.3.2 Decommission

```
curl -v "http://127.0.0.1/dataNode/decommission?addr=127.0.0.1:5000"
```

remove the dataNode from cluster, data partitions which locate the dataNode will be migrate other available dataNode asynchronous

Table 6: Parameters

| Parameter | Type | Description |
|---|---|---|
| addr | string | the addr which communicate with master |

## 13.4 Volume

### 13.4.1 Create

```
curl -v "http://127.0.0.1/admin/createVol?name=test&capacity=100&owner=cfs"
```

allocate a set of data partition and a meta partition to the user.

Table 7: Parameters

| Parameter | Type | Description |
|---|---|---|
| name | string | |
| capacity | int | the quota of vol,unit is GB |
| owner | string | the owner of vol |

### 13.4.2 Delete

```
curl -v "http://127.0.0.1/vol/delete?name=test&authKey=md5(owner)"
```

Mark the vol status to MarkDelete first, then delete data partition and meta partition asynchronous,finally delete meta data from persist store

Table 8: Parameters

| Parameter | Type | Description |
|---|---|---|
| name | string | |
| authKey | string | calculates the MD5 value of the owner field as authentication information |

### 13.4.3 Get

```
curl -v "http://127.0.0.1/client/vol?name=test&authKey=md5(owner)" | python -m json.
↪tool
```

show the base information of the vol,such as name,the detail of data partitions and meta partitions and so on.

Table 9: Parameters

| Parameter | Type | Description |
|---|---|---|
| name | string | |
| authKey | string | calculates the MD5 value of the owner field as authentication information |

response

```
{
    "Name": "test",
    "VolType": "extent",
    "MetaPartitions": {},
    "DataPartitions": {}
}
```

### 13.4.4 Stat

```
curl -v http://127.0.0.1/client/volStat?name=test
```

show vol stat information

Table 10: Parameters

| Parameter | Type | Description |
|---|---|---|
| name | string | |

response

```
{
    "Name": "test",
    "TotalSize": 322122547200000000,
    "UsedSize": 15551511283278
}
```

### 13.4.5 Update

```
curl -v "http://127.0.0.1/vol/update?name=test&capacity=100&authKey=md5(owner)"
```

add the vol quota

Table 11: Parameters

| Parameter | Type | Description |
|---|---|---|
| name | string | |
| capacity | int | the quota of vol, unit is GB |
| authKey | string | calculates the MD5 value of the owner field as authentication information |

## 13.5 Meta Partition

### 13.5.1 Create

```
curl -v "http://127.0.0.1/metaPartition/create?name=test&start=10000"
```

split meta partition manually,if max meta partition of the vol which range is [0,end),end larger than start parameter,old meta partition range is[0,start], new meta partition is [start+1,end)

Table 12: Parameters

| Parameter | Type | Description |
|---|---|---|
| name | string | the name of vol |
| start | uint64 | the start value of meta partition which will be create |

### 13.5.2 Get

```
curl -v "http://127.0.0.1/client/metaPartition?id=1" | python -m json.tool
```

show base information of meta partition,such as id,start,end and so on.

Table 13: Parameters

| Parameter | Type | Description |
|---|---|---|
| id | uint64 | the id of meta partition |

response


```
{
    "PartitionID": 1,
    "Start": 0,
    "End": 9223372036854776000,
    "MaxNodeID": 1,
    "Replicas": {},
    "ReplicaNum": 3,
    "Status": 2,
    "PersistenceHosts": {},
    "Peers": {},
    "MissNodes": {}
}
```


### 13.5.3 Decommission

```
curl -v "http://127.0.0.1/metaPartition/decommission?id=13&addr=127.0.0.1:9021"
```

remove the replica of meta partition,and create new replica asynchronous

Table 14: Parameters

| Parameter | Type | Description |
|---|---|---|
| id | uint64 | the id of meta partition |
| addr | string | the addr of replica which will be decommission |

### 13.5.4 Load

```
curl -v "http://127.0.0.1/metaPartition/load?id=1"
```

send load task to the metaNode which meta partition locate on,then check the crc of each replica in the meta partition

Table 15: Parameters

| Parameter | Type | Description |
|---|---|---|
| id | uint64 | the id of data partition |

## 13.6 Data Partition

### 13.6.1 Create

```
curl -v "http://127.0.0.1/dataPartition/create?count=40&name=test"
```

create a set of data partition

Table 16: Parameters

| Parameter | Type | Description |
|---|---|---|
| count | int | the num of dataPartitions will be create |
| name | string | the name of vol |

### 13.6.2 Get

```
curl -v "http://127.0.0.1/dataPartition/get?id=100"  | python -m json.tool
```

Table 17: Parameters

| Parameter | Type | Description |
|---|---|---|
| id | uint64 | the id of data partition |

response

```
{
    "PartitionID": 100,
    "LastLoadTime": 1544082851,
    "ReplicaNum": 3,
    "Status": 2,
    "Replicas": {},
    "PartitionType": "extent",
    "PersistenceHosts": {},
    "Peers": {},
    "MissNodes": {},
    "VolName": "test",
    "RandomWrite": true,
    "FileInCoreMap": {}
}
```

### 13.6.3 Decommission

```
curl -v "http://127.0.0.1/dataPartition/decommission?id=13&addr=127.0.0.1:5000"
```

remove the replica of data partition,and create new replica asynchronous

Table 18: Parameters

| Parameter | Type | Description |
|---|---|---|
| id | uint64 | the id of data partition |
| addr | string | the addr of replica which will be decommission |

### 13.6.4 Load

```
curl -v "http://127.0.0.1/dataPartition/load?id=1"
```

send load task to the dataNode which data partition locate on,then check the crc of each file in the data partition asynchronous

Table 19: Parameters

| Parameter | Type | Description |
|---|---|---|
| id | uint64 | the id of data partition |

## 13.7 Master Management

### 13.7.1 Add

```
curl -v "http://127.0.0.1/raftNode/add?addr=127.0.0.1:80&id=3"
```

add new master node to master raft group

Table 20: Parameters

| Parameter | Type | Description |
|---|---|---|
| addr | string | the addr of master server, format is ip:port |
| id | uint64 | the node id of master server |

### 13.7.2 Remove

```
curl -v "http://127.0.0.1/raftNode/remove?addr=127.0.0.1:80&id=3"
```

remove the master node from master raft group

Table 21: Parameters

| Parameter | Type | Description |
|---|---|---|
| addr | string | the addr of master server, format is ip:port |
| id | uint64 | the node id of master server |

CHAPTER 14

# Meta Node API

## 14.1 Meta Partition

### 14.1.1 Get Partitions

```
curl -v  http://127.0.0.1:9092/getPartitions
```

Get all meta-partition base information of the metanode.

### 14.1.2 Get Partition by ID

```
curl -v http://127.0.0.1:9092/getPartitionById?pid=100
```

Get the specified partition information, this result contains: leader address, raft group peer and cursor.

Table 1: Parameters

| Parameter | Type | Description |
|---|---|---|
| pid | integer | meta-partition id |

## 14.2 Inode

### 14.2.1 Get Inode

```
curl -v http://127.0.0.1:9092/getInode?pid=100&ino=1024
```

Get inode information

Table 2: Parameters

| Parameter | Type | Description |
|---|---|---|
| pid | integer | meta-partition id |
| ino | integer | inode id |

### 14.2.2 Get Extents by Inode

```
curl -v http://127.0.0.1:9092/getExtentsByInode?pid=100&ino=1024
```

Get inode all extents information

Table 3: Parameters

| Parameter | Type | Description |
|---|---|---|
| pid | integer | meta-partition id |
| ino | integer | inode id |

### 14.2.3 Get All Inodes

```
curl -v http://127.0.0.1:9092/getAllInodes?pid=100
```

Get all inodes of the specified partition

Table 4: Parameters

| Parameter | Type | Description |
|---|---|---|
| pid | integer | meta-partition id |

## 14.3 Dentry

### 14.3.1 Get Dentry

```
curl -v 'http://127.0.0.1:9092/getDentry?pid=100&name=""&parentIno=1024'
```

Get dentry information

Table 5: Parameters

| Parameter | Type | Description |
|---|---|---|
| pid | integer | meta partition id |
| name | string | file or directory name |
| parentIno | integer | file or directory parent directory inode |

### 14.3.2 Get Directory

```
curl -v "http://127.0.0.1:9092/getDirectory?pid=100&parentIno=1024"
```

Get all files of the parent inode is 1024

Table 6: Parameters

| Parameter | Type | Description |
|---|---|---|
| pid | integer | partition id |
| ino | integer | inode id |

### 14.3.3 Get All Dentry

```
curl -v "http://127.0.0.1:9092/getAllDentry?pid=100"
```

Table 7: Parameters

| Parameter | Type | Description |
|---|---|---|
| pid | integer | partition id |

CHAPTER 15

# Performance

- Throughput
- Latency
- FIO
- IOR/MDtest

CHAPTER 16

# Integrity

- Linux Test Project / fs

CHAPTER 17

# Workload

- database backup
- Java application logs
- code git repo
- database systems

  MyRocks, MySQL Innodb, HBase,

CHAPTER 18

# Scalability

- volume scalability: tens to millions of cfs volumes
- metadata scalability: a big volume with billions of files/directories

CHAPTER 19

# FAQ