储宝文件系统（CubaoFS）v2.2 使用手册

ChubaoFS(储宝文件系统)是为大规模容器平台设计的分布式文件系统。

目　录

致谢

当前文档 《储宝文件系统（CubaoFS）v2.2 使用手册》 由 进击的皇虫 使用 书栈网 (BookStack.CN) 进行构建，生成于 2020-11-01。

书栈网仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈网难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常工作、生活和学习中遇到有价值有营养的知识文档，欢迎分享到书栈网，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到书栈网获取最新的文档，以跟上知识更新换代的步伐。

内容来源：chubaofs https://github.com/chubaofs/docs-zh

文档地址：http://www.bookstack.cn/books/chubaofs-2.2-zh

书栈官网：https://www.bookstack.cn

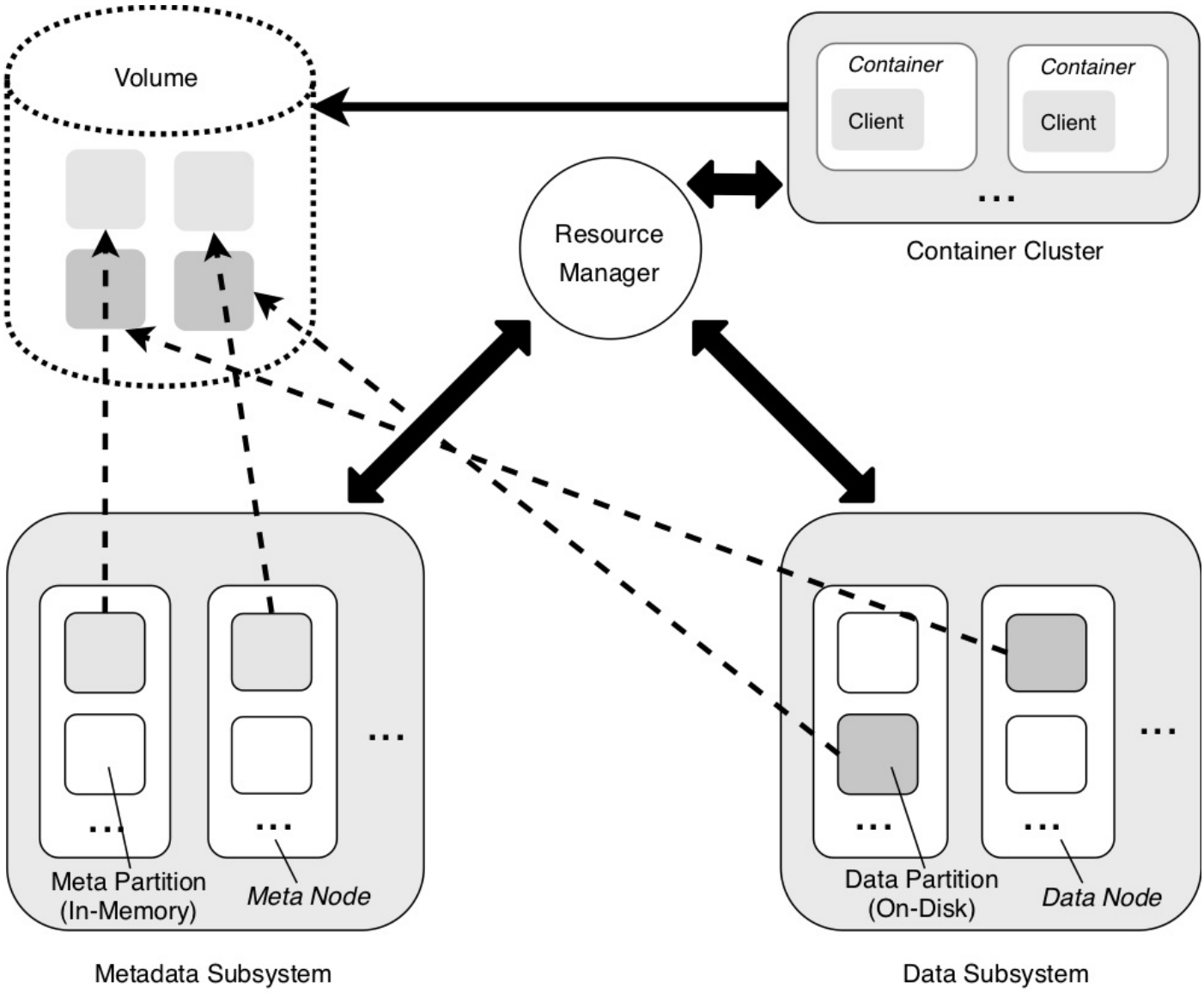
书栈开源：https://github.com/TruthHun

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

概述

ChubaoFS(储宝文件系统)是为大规模容器平台设计的分布式文件系统。

整体架构



ChubaoFS由 元数据子系统 ，数据子系统 和 资源管理节点 组成，可以通过 客户端 访问不同文件系统实例， 卷 。

元数据子系统由元数据节点组成，每个节点可以管理一组 元数据分片 。

数据子系统由数据节点组成，每个节点管理一组 数据分片 。

在ChubaoFS中，卷是一个逻辑概念，由多个元数据和数据分片组成。 从客户端的角度看，卷可以被看作是可被容器访问的文件系统实例。 一个卷可以在多个容器中挂载，使得文件可以被不同客户端同时访问。 一个ChubaoFS集群可以有上百个卷，大小从几GB至几TB不等。

概括来说，资源管理节点定期获取元数据和数据子系统信息，客户端则定期从资源管理器拉取元数据和数据分片信息，并且进行缓存。通常来讲，文件操作由客户端发起，直接与数据和元数据节点通信，无需资源管理节点介入。

系统特性

可扩展元数据管理

元数据操作有时候会成为文件系统的性能瓶颈。在我们的平台中，由于可能会有成百上千的客户端同时访问文件，这个问题变得非常突出。单独节点存储元数据很容易成为性能瓶颈。所以，ChubaoFS中使用了分布式元数据子系统，以便提供高可扩展性。 元数据子系统可以认为是内存元数据存储。我们使用了两个B-Tree，inodeTree和dentryTree，来加快索引速度。每个元数据分片根据inode id范围进行划分。

多租户

为了降低存储成本，很多应用和服务都使用了共享的存储基础设施。不同负载交织在一起，文件大小可以从几KB至几百GB，读写模型从顺序到随机。一个成熟的文件系统应该可以高性能地服务于这些负载。ChubaoFS的存储引擎同时提供了对大文件和小文件的随机/顺序读写的支持。

强一致性的复制协议

出于性能考虑，ChubaoFS根据文件写入的方式的不同采用不同的复制协议来保障副本之间的一致性。

POSIX兼容

兼容POSIX接口，可以使得上层应用的开发变得简单，并且大大降低新用户的学习难度。同时，ChubaoFS在实现时放松了对POSIX语义的一致性要求来兼顾文件和元文件操作的性能。

手动部署集群

编译构建

使用如下命令同时构建server，client及相关的依赖：

```
$ git clone http://github.com/chubaofs/chubaofs.git
$ cd chubaofs
$ make build
```

如果构建成功，将在`build/bin` 目录中生成可执行文件`cfs-server`和`cfs-client`。

集群部署

启动资源管理节点

```
nohup ./cfs-server -c master.json &
```

示例 `master.json` ：注意：master服务最少应该启动3个节点实例

```
{
  "role": "master",
  "ip": "10.196.59.198",
  "listen": "17010",
  "prof":"17020",
  "id":"1",
  "peers": 1:10.196.59.198:17010,2:10.196.59.199:17010,3:10.196.59.200:17010",
  "retainLogs":"20000",
  "logDir": "/cfs/master/log",
  "logLevel":"info",
  "walDir":"/cfs/master/data/wal",
  "storeDir":"/cfs/master/data/store",
  "consulAddr": "http://consul.prometheus-cfs.local",
  "exporterPort": 9500,
  "clusterName":"chubaofs01",
  "metaNodeReservedMem": "1073741824"
}
```

详细配置参数请参考 资源管理节点 。

启动元数据节点

```
nohup ./cfs-server -c metanode.json &
```

示例 `meta.json` ：注意：metanode服务最少应该启动3个节点实例

```
{
    "role": "metanode",
    "listen": "17210",
    "prof": "17220",
    "logLevel": "info",
    "metadataDir": "/cfs/metanode/data/meta",
    "logDir": "/cfs/metanode/log",
    "raftDir": "/cfs/metanode/data/raft",
    "raftHeartbeatPort": "17230",
    "raftReplicaPort": "17240",
    "totalMem":  "8589934592",
    "consulAddr": "http://consul.prometheus-cfs.local",
    "exporterPort": 9501,
    "masterAddr": [
        "10.196.59.198:17010",
        "10.196.59.199:17010",
        "10.196.59.200:17010"
    ]
}
```

启动元数据节点

```
nohup ./cfs-server -c metanode.json &
```

示例 `meta.json` ：注意：metanode服务最少应该启动3个节点实例

```
{
    "role": "metanode",
    "listen": "17210",
    "prof": "17220",
    "logLevel": "info",
    "metadataDir": "/cfs/metanode/data/meta",
    "logDir": "/cfs/metanode/log",
```

```
    "raftDir": "/cfs/metanode/data/raft",
    "raftHeartbeatPort": "17230",
    "raftReplicaPort": "17240",
    "totalMem":  "8589934592",
    "consulAddr": "http://consul.prometheus-cfs.local",
    "exporterPort": 9501,
    "masterAddr": [
        "10.196.59.198:17010",
        "10.196.59.199:17010",
        "10.196.59.200:17010"
    ]
}
```

启动动 ObjectNode

```
nohup ./cfs-server -c objectnode.json &
```

示例 `objectnode.json` 内容如下

```
{
    "role": "objectnode",
    "domains": [
        "object.cfs.local"
    ],
    "listen": 17410,
    "masterAddr": [
       "10.196.59.198:17010",
       "10.196.59.199:17010",
       "10.196.59.200:17010"
    ],
    "logLevel": "info",
    "logDir": "/cfs/Logs/objectnode"
}
```

配置文件的详细信息 *objectnode.json*, 请参阅 对象存储(ObjectNode).

启动管理平台（非必须）

```
nohup ./cfs-server -c console.json &
```

示例 `console.json` 内容如下

```
{
    "role": "console",
    "logDir": "/cfs/log/",
    "logLevel": "debug",
    "listen": "80",
    "masterAddr": [
        "192.168.0.11:17010",
        "192.168.0.12:17010",
        "192.168.0.13:17010"
    ],
    "objectNodeDomain": "object.chubao.io",
    "master_instance": "192.168.0.11:9066",
    "monitor_addr": "http://192.168.0.102:9090",
    "dashboard_addr": "http://192.168.0.103",
    "monitor_app": "cfs",
    "monitor_cluster": "cfs"
}
```

配置文件的详细信息 *console.json*, 请参阅 Console.

详细配置参数请参考 元数据节点.

启动数据节点

1. 准备数据目录

   推荐 使用单独磁盘作为数据目录，配置多块磁盘能够达到更高的性能。

   磁盘准备

   1.1 查看机器磁盘信息，选择给ChubaoFS使用的磁盘

   ```
   fdisk -l
   ```

   1.2 格式化磁盘，建议格式化为XFS

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

   1.3 创建挂载目录

   ```
   mkdir /data0
   ```

1.4 挂载磁盘

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

2. 启动数据节点

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

示例 `datanode.json` :注意：datanode服务最少应该启动4个节点实例

```
{
  "role": "datanode",
  "listen": "17310",
  "prof": "17320",
  "logDir": "/cfs/datanode/log",
  "logLevel": "info",
  "raftHeartbeat": "17330",
  "raftReplica": "17340",
  "raftDir":"/cfs/datanode/log",
  "consulAddr": "http://consul.prometheus-cfs.local",
  "exporterPort": 9502,
  "masterAddr": [
     "10.196.59.198:17010",
     "10.196.59.199:17010",
     "10.196.59.200:17010"
  ],
  "disks": [
     "/data0:10737418240",
     "/data1:10737418240"
  ]
}
```

详细配置参数请参考 数据节点.

启动对象管理节点

```
nohup ./cfs-server -c objectnode.json &
```

示例 *objectnode.json is* 如下:

```
{
    "role": "objectnode",
    "domains": [
        "object.cfs.local"
    ],
    "listen": 17410,
    "masterAddr": [
       "10.196.59.198:17010",
       "10.196.59.199:17010",
       "10.196.59.200:17010"
    ],
    "logLevel": "info",
    "logDir": "/cfs/Logs/objectnode"
}
```

关于 *object.json* 的更多详细配置请参考 对象存储(ObjectNode).

创建Volume卷

```
curl -v "http://10.196.59.198:17010/admin/createVol?name=test&capacity=10000&owner=cfs"

如果执行性能测试，请调用相应的API，创建足够多的数据分片（data partition）,如果集群中有8块磁盘，那么需要创建80个datapartition
```

挂载客户端

1. 运行 `modprobe fuse` 插入FUSE内核模块。
2. 运行 `yum install -y fuse` 安装libfuse。
3. 运行 `nohup client -c fuse.json &` 启动客户端。

   样例 *fuse.json* ,

```
{
  "mountPoint": "/cfs/mountpoint",
  "volName": "ltptest",
  "owner": "ltptest",
  "masterAddr": "10.196.59.198:17010,10.196.59.199:17010,10.196.59.200:17010",
  "logDir": "/cfs/client/log",
```

```
  "profPort": "17510",
  "exporterPort": "9504",
  "logLevel": "info"
}
```

详细配置参数请参考 客户端.

用户可以使用不同的挂载点在同一台机器上同时启动多个客户端

升级注意事项

集群数据节点和元数据节点升级前，请先禁止集群自动为卷扩容数据分片.

1. 冻结集群

```
curl -v "http://10.196.59.198:17010/cluster/freeze?enable=true"
```

1. 升级节点
2. 开启自动扩容数据分片

```
curl -v "http://10.196.59.198:17010/cluster/freeze?enable=false"
```

注：升级节点时不能修改各节点配置文件的端口。

- 资源管理子系统
- 元数据子系统
- 数据子系统
- 对象存储 (ObjectNode)
- 客户端
- 鉴权节点

资源管理子系统

Master负责异步的处理不同类型的任务，比如 创建/删除/更新/比对副本是否一致等数据分片和元数据分片的操作,管理数据节点和元数据节点的存活状态，创建和维护卷信息。 Master有多个节点，它们之间通过raft算法保证元数据一致性，并且把元数据持久化到RocksDB。

基于利用率的分布策略

基于利用率的分布策略放置文件元数据和内容是Master最主要的特征，此分布策略能够更高效的利用集群资源。 数据分片和元数据分片的分布策略工作流程： 1. 创建卷的时候，master根据剩余磁盘/内存空间加权计算，选择权重最高的数据节点创建数据/元数据分片，写文件时，客户端随机选择数据分片和元数据分片。 2. 如果卷的大部分数据分片是只读，只有很少的数据分片是可读写的时候，master会自动创建新的数据分片来分散写请求。

基于利用率的分布策略能带来两点额外的好处： 1. 当新节点加入时，不需要重新做数据均衡，避免了因为数据迁移带来的开销。 2. 因为使用统一的分布策略， 显著降低了产生热点数据的可能性。

副本放置

Master确保一个分片的多个副本都在不同的机器上.

拆分元数据分片

满足下面任意一个条件，元数据分片将会被拆分 1. 元数据节点内存使用率达到设置的阈值，比如总内存是64GB，阈值是0.75，如果元数据节点使用的内存达到48GB，该节点上的所有元数据分片都将会被拆分. 2. 元数据分片占用的内存达到16GB

只有数据分片ID是卷所有数据分片中ID最大的，才会真正被拆分。假设数据分片A符合拆分条件，其inode范围是[0,正无穷)， 则拆分后A的范围为[0,A.MaxInodeID+step),新生成的B分片的范围是[A.MaxInodeID+step+1，正无穷)，其中step是步长，默认是2的24方.MaxInodeID是由元数据节点汇报。

异常处理

如果数据/元数据分片某个副本不可用 (硬盘失败、硬件错误等)， 该副本上的数据最终会被迁移到新的副本上.

元数据子系统

元数据子系统是一个以内存为中心的分布式数据结构，是由一个或多个元数据分片组成。元数据子系统数据使用multiraft来充分使用服务器资源和保障数据高可用及强一致性，可以很方便的迁移资源，并通过分裂实现横向扩展。

元数据内部设计

> 每个元数据可以包含成百上千的元数据分片，每个分片由InodeTree（BTree）和DentryTree（BTree）组成。每个Inode代表文件系统中的一个文件或目录， 每个dentry代表一个目录项，dentry由parentId和name组成。在DentryTree中，以PartentId和name组成索引，进行存储和检索；在InodeTree中，则以inode id进行索引。使用multiRaft协议保障高可用性和数据一致性复制,且每个节点集合会包含大量的分片组，每个分片组对应一个raft group；每个分片组隶属于某个volume；每个分片组都是某个volume的一段元数据范围（inode id范[100-20000）);元数据子系统通过分裂来完成动态扩容；当性一分片组的性能（包含如下指标：内存）紧接临近值时，资源管理器服务会预估一个结束点，并通知此组节点设备，只服务到此点之前的数据，同时也会新选出一组节点，并动态加入到当前业务系统中，新节点组其实点刚好是上个节点组的结束点位置。

复制

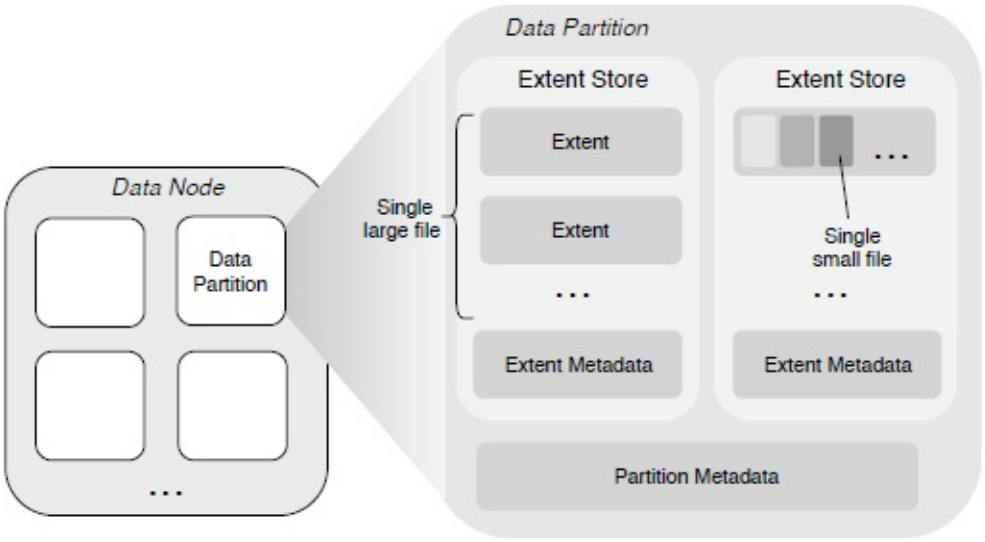
> 元数据更新的复制是以元数据分片为单位的。复制强一致性是通过Raft改良版本，MultiRaft来实现的。MultiRaft减少了心跳通信的负担。

故障恢复

> 内存元数据分片通过快照的方式持久化到磁盘以作备份和恢复使用。日志压缩技术被用来减小日志文件大小和恢复时间。 值得一提的是，元数据操作有可能会导致孤儿inode，即只有inode但是没有对应的dentry。为了减少这种情况的发生，首先，元数据节点通过Raft保证高可用，单点故障后可以迅速恢复；其次，客户端保证在一定时间内进行重试。

数据子系统

数据子系统的设计是为了满足大、小文件支持顺序随机访问的多租户需求。采用两种不同的复制协议，以确保副本之间的强一致性，并在性能和代码可用性上进行一些权衡。



系统特性

- 大文件存储

  对于大文件，内容存储为一个或多个扩展数据块的序列，这些扩展数据块可以分布在不同数据节点上的不同数据分区中。将新文件写入扩展数据块存储区始终会导致数据以新扩展数据块的零偏移量写入，这样就不需要在扩展数据块内进行偏移。文件的最后一个范围不需要通过填充来补齐其大小限制（即该范围没有空洞），并且不会存储来自其他文件的数据。

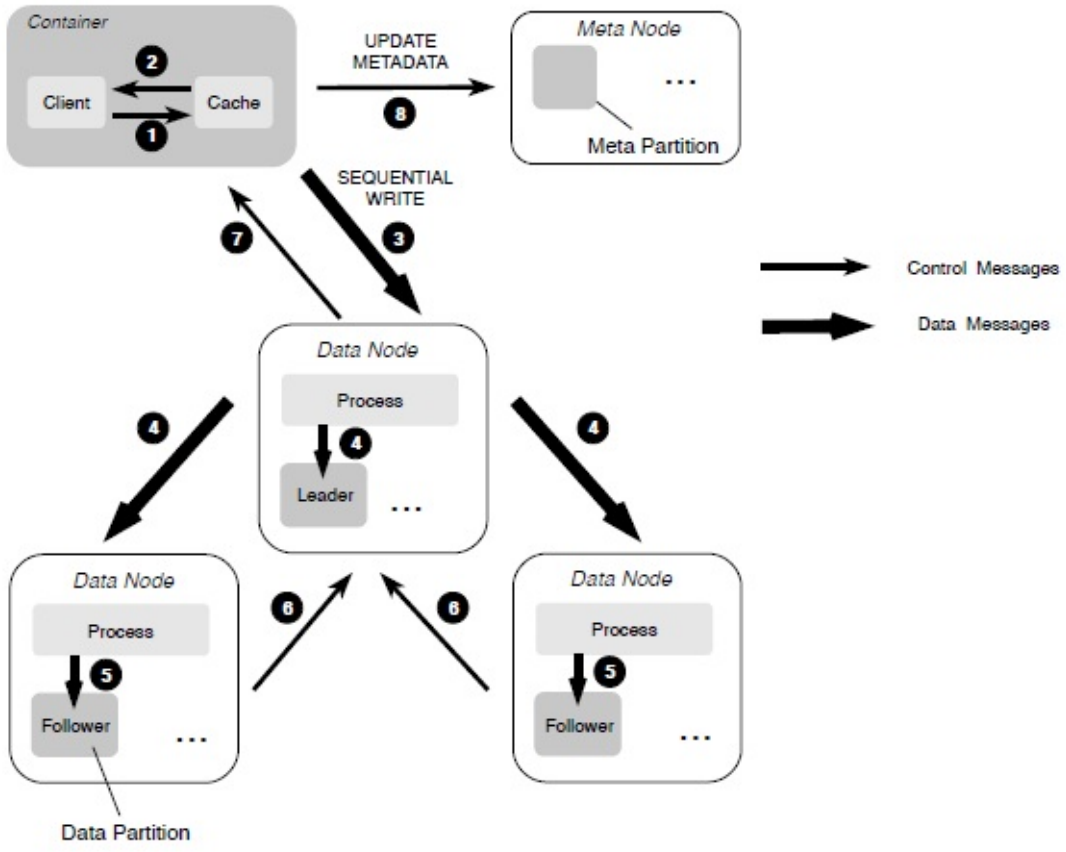
- 小文件存储

  将多个小文件的内容聚合存储在一个文件内，并将每个文件内容的物理偏移量记录在相应的元数据中。删除文件内容（释放此文件占用的磁盘空间）是通过底层文件系统提供的文件穿洞接口（`fallocate()`）实现的。这种设计的优点是不需要实现垃圾回收机制，因此在一定程度上避免使用从逻辑偏移到物理偏移的映射。

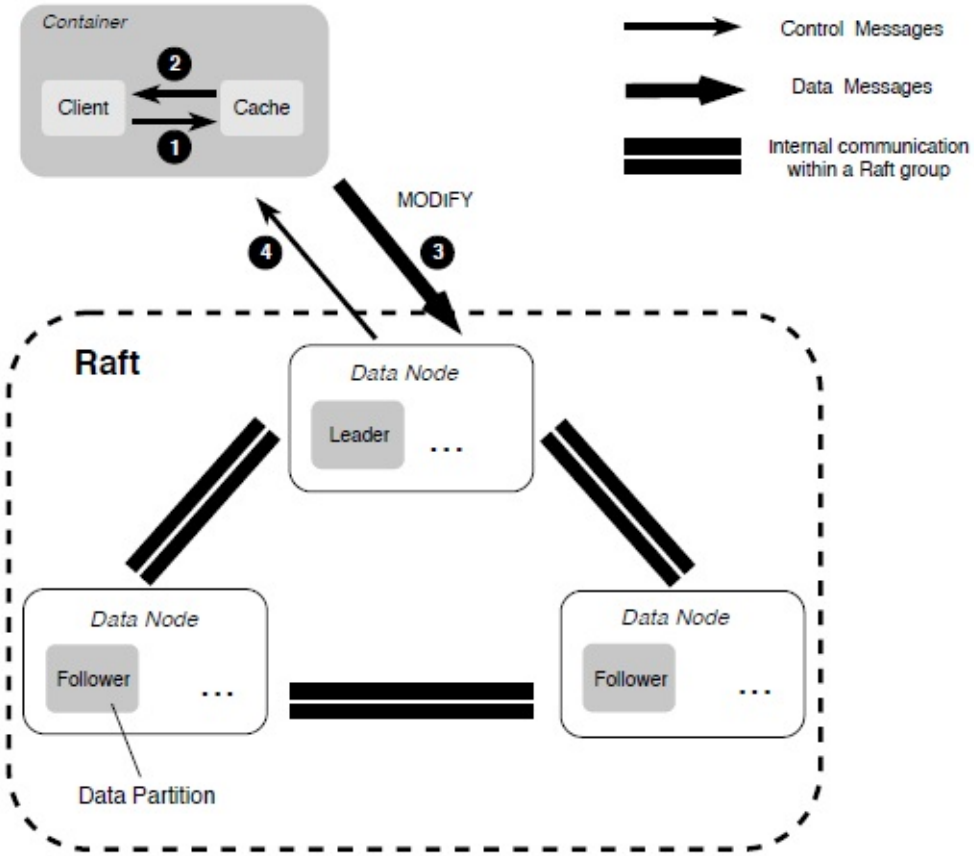
- 复制

  成员间的文件复制，根据文件写入模式，ChubaoFS采用不同的复制策略。

  当文件按顺序写入ChubaoFS时，使用主备份复制协议来确保与优化的IO吞吐量的强一致性。



在随机写入时覆盖现有的文件内容时，我们采用了一种基于Multi-Raft的复制协议，该协议类似于元数据子系统中使用的协议，以确保强一致性。



- 故障恢复

  由于存在两种不同的复制协议，当发现复制副本上的故障时，我们首先通过检查每个数据块的长度并使所有数据块对齐，启动基于主备份的复制协议的恢复。一旦这个处理完成，我们就开始在我们的基于Multi-Raft的恢复。

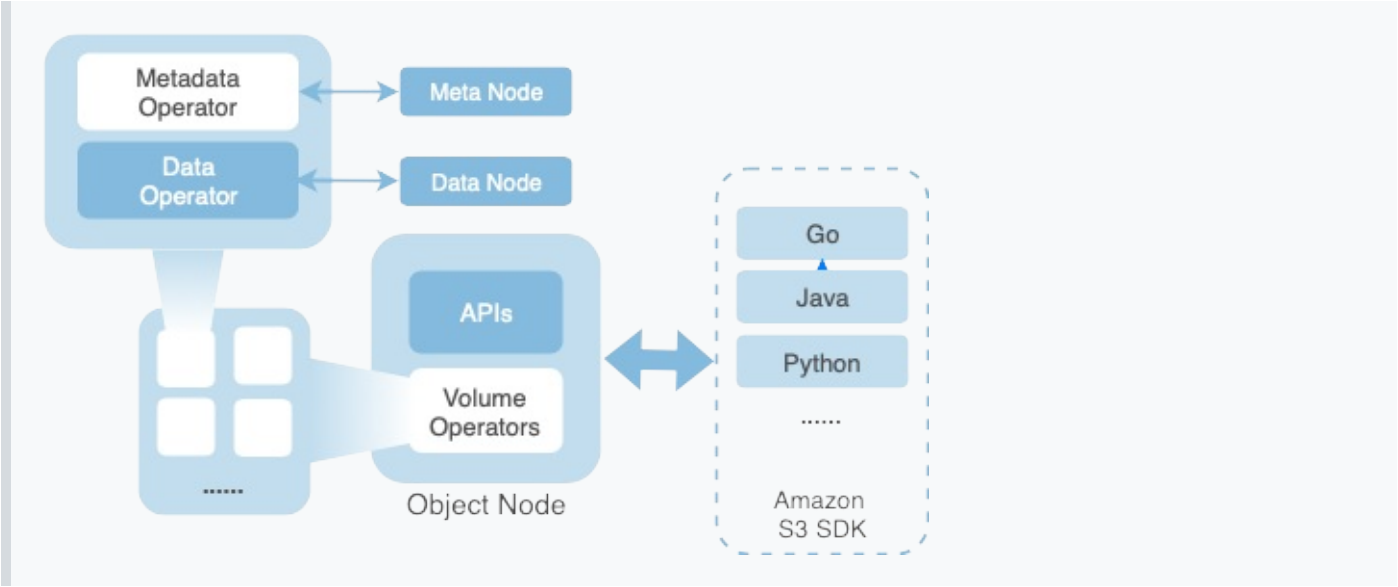
HTTP接口

API	方法	参数	描述
/disks	GET	N/A	获取磁盘的列表和信息。
/partitions	GET	N/A	获取所有数据组的信息。
/partition	GET	partitionId[int]	获取特定数据组的详细信息。
/extent	GET	partitionId[int]&extentId[int]	获取特定数据组里面特定extent文件的信息。
/stats	GET	N/A	获取DATA节点的信息。

对象存储（ObjectNode）

对象存储系统提供兼容S3的对象存储接口。它使得ChubaoFS成为一个可以将两种通用类型接口进行融合的存储（POSIX和S3兼容接口）。可以使用户使用原生的Amazon S3 SDK操作ChubaoFS中的文件。

框架



ObjectNode是一个功能性的子系统节点。它根据需要从资源管理器（Master）获取卷视图（卷拓扑）。 每个ObjectNode直接与元数据子系统（MetaNode）和数据子系统（DataNode）通信。

ObjectNode是一种无状态设计，具有很高的可扩展性，能够直接操作ChubaoFS集群中存储的所有文件，而无需任何卷装入操作。

特性

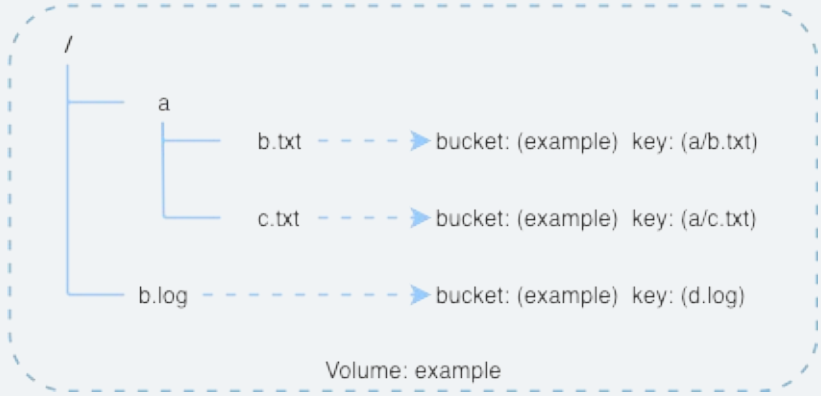
- 支持原生的Amazon S3 SDKs的对象存储接口
- 支持两种通用接口的融合存储（POSIX和S3兼容接口）
- 无状态和高可靠性

语义转换

基于原有POSIX兼容性的设计。每个来自对象存储接口的文件操作请求都需要对POSIX进行语义转换。

POSIX	Object Storage
`Volume`	`Bucket`
`Path`	`Key`

示例：



Put object *'example/a/b.txt'* will be create and write data to file *'/a/b.txt'* in volume *'example'*.

用户

在使用对象存储功能前，需要先通过资源管理器创建用户。创建用户的同时，会为每个用户生成 *AccessKey* 和 *SecretKey* ，其中 *AccessKey* 是整个ChubaoFS集群中唯一的16个字符的字符串。

ChubaoFS以卷的 **Owner** 字段作为用户ID。创建用户的方式有两种：

1. 通过资源管理器的API创建卷时，如果集群中没有与该卷的Owner同名的用户时，会自动创建一个用户ID为Owner的用户
2. 调用资源管理器的用户管理API创建用户，链接： 用户管理命令

授权与鉴权

对象存储接口中的签名验证算法与Amazon S3服务完全兼容。用户可以通过管理API获取用户信息，请参见 **Get User Information** ，链接： 用户管理命令 。从中获取 *AccessKey* 和 *SecretKey* 后，即可利用算法生成签名来访问对象存储功能。

用户对于自己名下的卷，拥有所有的访问权限。用户可以授予其他用户指定权限来访问自己名下的卷。权限分为以下三类：

- 只读或读写权限；
- 单个操作的权限，比如GetObject、PutObject等；
- 自定义权限。

当用户使用对象存储功能进行某种操作时，ChubaoFS会鉴别该用户是否拥有当前操作的权限。

临时隐藏数据

以原子方式在对象存储接口中进行写操作。每个写操作都将创建数据并将其写入一个不可见的临时对象。ObjectNode中的volume运算符将文件数据放入临时文件，临时文件的元数据中只有'**inode**'而没有'**dentry**'。当所有文件数据都成功存储时，volume操作符在元数据中创建或更新'**dentry**'使其对用户可见。

对象名称冲突（重要）

POSIX和对象存储是两种不同类型的存储产品，对象存储是一种键-值对存储服务。所以在对象存储中，名称为'*a/b/c*'和名称为'*a/b*'的对象是两个完全没有冲突的对象。

不过ChubaoFS是基于POSIX设计的。根据语义转换规则，对象名'*a/b/c*'中的'*b*'部分转换为文件夹'*a*'下的文件夹'*b*'，对象名'*a/b*'中的'*b*'部分转换为文件夹'*a*'下的文件'*b*'。

类似于上面这样的对象名称在ChubaoFS中是冲突的。

支持的S3兼容接口

桶接口

API	Reference
HeadBucket	https://docs.aws.amazon.com/AmazonS3/latest/API/API_HeadBucket.ht
GetBucketLocation	https://docs.aws.amazon.com/AmazonS3/latest/API/API_GetBucketLoca

对象接口

API	Reference
HeadObject	https://docs.aws.amazon.com/AmazonS3/latest/API/API_HeadObject.html
PutObject	https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html
GetObject	https://docs.aws.amazon.com/AmazonS3/latest/API/API_GetObject.html
ListObjects	https://docs.aws.amazon.com/AmazonS3/latest/API/API_ListObjects.html
ListObjectsV2	https://docs.aws.amazon.com/AmazonS3/latest/API/API_ListObjectsV2.htm
DeleteObject	https://docs.aws.amazon.com/AmazonS3/latest/API/API_DeleteObject.html
DeleteObjects	https://docs.aws.amazon.com/AmazonS3/latest/API/API_DeleteObjects.htm
CopyObject	https://docs.aws.amazon.com/AmazonS3/latest/API/API_CopyObject.html

并发上传接口

API	Reference
`CreateMultipartUpload`	https://docs.aws.amazon.com/AmazonS3/latest/API/API_CreateMu
`ListMultipartUploads`	https://docs.aws.amazon.com/AmazonS3/latest/API/API_ListMult
`AbortMultipartUpload`	https://docs.aws.amazon.com/AmazonS3/latest/API/API_AbortMul
`CompleteMultipartUpload`	https://docs.aws.amazon.com/AmazonS3/latest/API/API_Complete
`ListParts`	https://docs.aws.amazon.com/AmazonS3/latest/API/API_ListPart
`UploadPart`	https://docs.aws.amazon.com/AmazonS3/latest/API/API_UploadPa
`UploadPartCopy`	https://docs.aws.amazon.com/AmazonS3/latest/API/API_UploadPa

支持的SDK

Object Node提供兼容S3的对象存储接口，所以可以直接使用原生的Amazon S3 SDKs来操作文件。

Name	Language	Link
AWS SDK for Java	`Java`	https://aws.amazon.com/sdk-for-java/
AWS SDK for JavaScript	`JavaScript`	https://aws.amazon.com/sdk-for-browser/
AWS SDK for JavaScript in Node.js	`JavaScript`	https://aws.amazon.com/sdk-for-node-js/
AWS SDK for Go	`Go`	https://docs.aws.amazon.com/sdk-for-go/
AWS SDK for PHP	`PHP`	https://aws.amazon.com/sdk-for-php/
AWS SDK for Ruby	`Ruby`	https://aws.amazon.com/sdk-for-ruby/
AWS SDK for .NET	`.NET`	https://aws.amazon.com/sdk-for-net/
AWS SDK for C++	`C++`	https://aws.amazon.com/sdk-for-cpp/
Boto3	`Python`	http://boto.cloudhackers.com

客户端

客户端以用户态可执行程序的形式可以运行在容器中，并且通过FUSE将挂载的卷及文件系统接口提供给其它用户态应用。

客户端缓存

客户端进程在以下几种情况下会使用客户端缓存。

客户端为了减少与资源管理节点的通信负担，会在挂载启动时获取该挂载卷中所有元数据和数据节点的地址，并且进行缓存，后续会定期从资源管理节点进行更新。

客户端为了减少与元数据节点的通信，会缓存inode，dentry以及extent元数据信息。通常意义上，读请求应该能够读到之前所有的写入，但是客户端元数据缓存可能会导致多客户端写同一个文件时的一致性问题。所以，ChubaoFS的设计中，不同客户端，或者说挂载点，可以同时读一个文件，但是不能够同时写一个文件（注意：不同进程可以从同一个挂载点并发写一个文件）。打开文件时，客户端会强制从元数据节点更新文件元数据信息。

由于故障恢复时，raft复制组的主节点有可能发生变化，导致客户端缓存的主节点地址无效。因此，客户端在发送请求收到not leader回复时，会轮询重试该复制组的所有节点。重试成功后识别出新的主节点，客户端会缓存新的主节点地址。

对接FUSE接口

ChubaoFS客户端通过对接FUSE为提供用户态文件系统接口。之前，性能较低被认为是用户态文件系统最大的缺点。但是经过多年的发展，FUSE已经在性能上有了很大提高。后续，ChubaoFS会着手开发内核态文件系统客户端。

目前来看，FUSE的writeback cache特性并未达到预期的性能提升。FUSE默认的写流程走的是directIO接口，使得每次写入长度较小时会有性能问题，因为每次的写请求都会被推送至后端存储。FUSE的解决方案是writeback cache，即小写入写到缓存页即返回，由内核根据回刷策略推送至后端存储。这样，顺序的小请求会被聚合。但是，在实际生产中，我们发现writeback cache特性作用非常有限，原因是走writecache的写操作触发了内核balance dirty page的流程，使得本应该是响应时间非常短的写操作仍然会等较长时间才返回。这个问题在小的写入时尤其明显。

鉴权节点

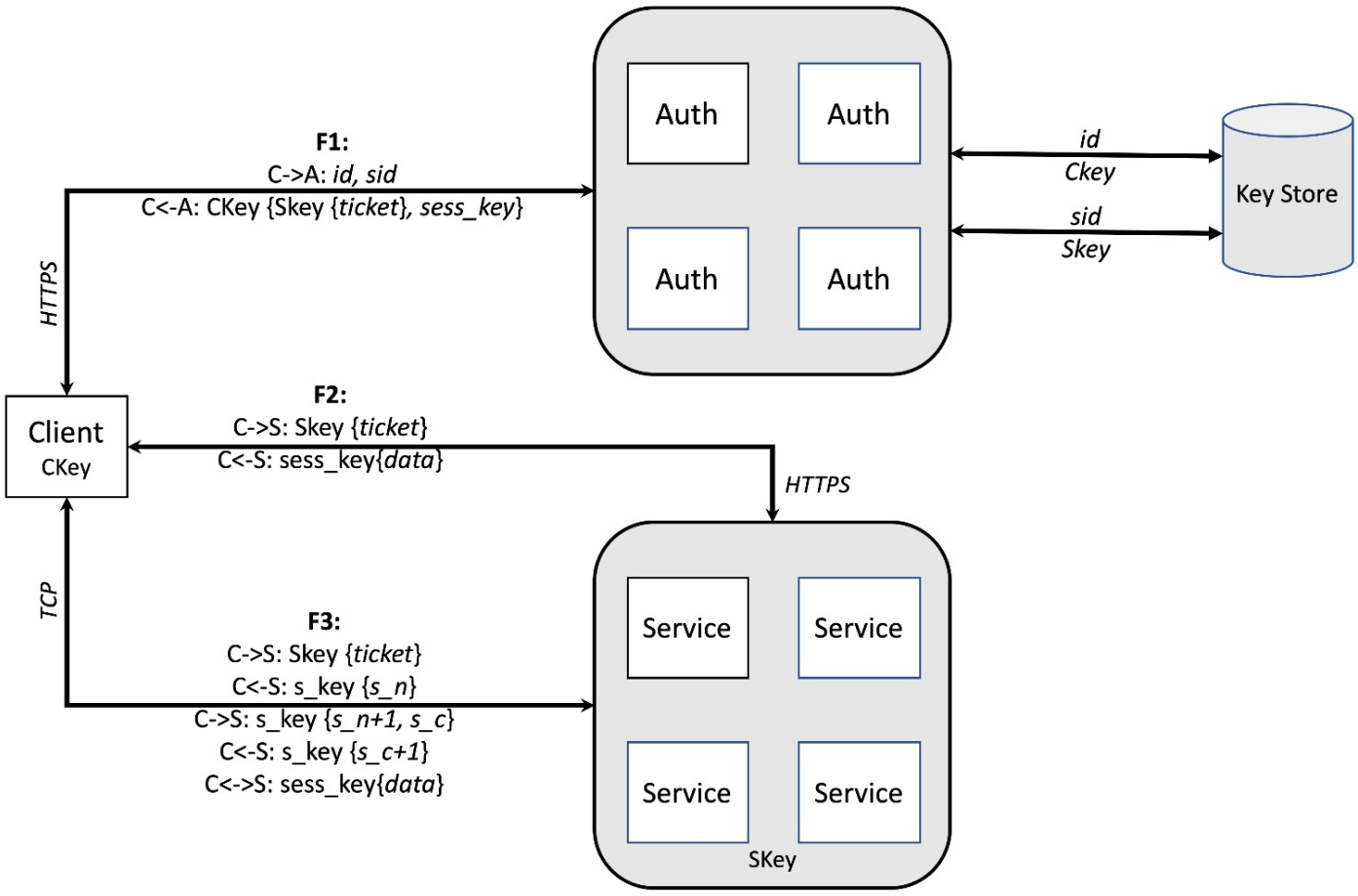
众所周知，Internet（国际互联网）和intranet（企业内部网）都是不安全的地方，黑客通常可以使用工具在网络上嗅探到很多敏感信息。更糟糕的是，由于无法确认对端是否诚实的表达了自己的身份，客户端与服务端之间没有办法建立可信的连接。因此，如果没有鉴权机制，ChubaoFS一旦部署在了网络中，便会存在一些常见的安全问题。

安全问题

- 未认证的节点可能会访问敏感信息，如restful API，volume信息等
- 通信信道可能受到 Man-in-the-middle (MITM) 攻击

系统架构

Authnode 是为 ChubaoFS 提供通用认证和授权框架的安全节点。此外， Authnode 还充当对称密钥和非对称密钥的集中密钥存储。 Authnode 采用并定制了基于票证的 Kerberos 认证思想。具体来说，当客户端节点（ Master 、 Meta 、 Data 或 client 节点）访问服务时，首先需要在 Authnode 中展示用于身份验证的共享密钥。如果认证成功， Authnode 将专门为该服务颁发一个限时票证。出于授权的目的，功能嵌入到票证中，以指示 谁可以在什么资源上做什么 。



在 Authnode 的上下文中，我们将负责初始化一个服务请求的节点定义为 Client ，而响应该请求的节点定义为 Server 。这样，任何 ChubaoFS 节点都可以充当 Client 或者 Server 。

Client 和 Server 之间通过 HTTPS 或者 TCP 进行通信。 Authnode 的工作流程如上图所示，简要描述如下：

凭据请求（F1）

在任何服务请求之前，需要持有密钥 *Ckey* 的客户端为服务（ target service ）从 Authnode 获取服务的认证信息。

- C->A: 客户端发送一个认证请求，请求中包含一个代表客户端的客户端ID（*id*）和一个代表目标服务的服务ID（*sid*）。
- C<-A: 服务端从自己的 key store 中查找客户端秘钥（*CKey*）和服务端秘钥（*SKey*）。如果认证成功，服务端会向客户端回应一条*CKey*加密的消息。消息中会包含会话秘钥（*sess_key*）和目标服务的凭据。

在获取凭据并使用*Ckey*处理一些安全检查后，客户端拥有*sess_key*和Skey{*ticket*}以备将来的服务请求。

HTTPS中的服务请求（F2）

如果服务请求是通过HTTPS协议发送的，那么它将按如下步骤进行：

- C->S: 客户端发送一个包含SKey {*ticket*}的请求。
- C<-S: 服务端进行以下操作，（1）执行消息解密并获取凭据，（2）验证功能，（3）使用从凭据中获取的*sess_key*，加密*data*返回给客户端。

客户端使用*sess_key*解密从服务器返回的消息并验证其有效性。

TCP中的服务请求（F3）

如果服务请求是通过TCP协议发送的，那么它将按如下步骤进行：

- C->S: 客户端发送一个包含SKey {*ticket*}的请求。
- C<-S: 服务端（1）解密凭据并验证其功能，（2）提取*sess_key*，（3）生成一个随机数*s_n*，（4）用*sess_key*加密的这个数后响应给客户端。
- C->S: 客户端解密回复的消息，并向服务端发送另一条消息，其中包括随机生成的数字*s_c*和*s_n*+1，这两个数字都用*sess_key*加密。
- C<-S: 服务器将验证*s_n*是否增加了1。如果验证成功,服务端将发送一条包含*s_c*+1的消息，该消息由*sess_key*加密。
- C<->S: 客户机在消息解密后验证*s_c*是否增加了一个。如果成功，则已在客户端和服务器之间建立经过身份验证的通信通道。基于此通道，客户端和服务器可以执行进一步的通信。

未来工作

Authnode 支持ChubaoFS急需的一般的身份验证和授权。未来 ChubaoFS 的安全增强有两个方向：

特性丰富

当前的 Authnode 实现不支持某些高级功能：

- 密钥轮换：共享密钥在客户端和服务器中硬编码，不会更改。它增加了安全风险，攻击会破坏加密并找到密钥。定期轮换秘钥有助于降低此类风险。
- 凭据撤销：出于性能考虑，凭据将有效一段时间（如几个小时）。如果客户端不幸地泄漏了它的票证，恶意方可以在有效期内将票证用于服务请求。凭据撤销机制可以通过在发生泄漏时撤销凭据来预防此类问题。
- HSM支持： Authnode 是ChubaoFS中的安全瓶颈。破坏 Authnode 意味着破坏整个系统，因为它管理密钥存储。硬件安全模块（HSM）为密钥管理提供物理保护。让HSM（例如*SGX*）保护 Authnode ，可以降低 Authnode 被破坏的风险。

端到端数据加密

当前的 Authnode 实现并不系统地支持对传输中和静止数据的加密，即使我们可以在通信期间使用会话密钥加密数据。保护数据的一种更安全的方法是使用 端到端 数据 加密 。特别是，加密密钥由 Authnode 管理和分发，数据在客户端节点加密，通过网络发送并存储在服务端中。与基于现有工具（ fscrypt 、 ecryptfs 和 dm-crypt ）的服务端加密相比， 端到端 数据 加密 至少具有以下优点：

-由于数据解码密钥存储在 Authnode 中，一旦数据服务器（例如 data Node ）被攻击者入侵，它就可以减少数据泄漏。 -它提供对加密密钥的集中管理（轮换、撤销和生成）。

- 资源管理节点
- 元数据节点
- 数据节点
- 对象存储(ObjectNode)
- Console
- 客户端
- 授权节点
- 性能监控
- 优化配置FUSE参数
- yum工具自动部署集群
- 启动docker集群
- CSI插件支持

资源管理节点

Master负责管理ChubaoFS整个集群，主要存储5种元数据，包括：数据节点、元数据节点、卷、数据分片、元数据分片。所有的元数据都保存在master的内存中，并且持久化到RocksDB。 多个Master之间通过raft协议保证集群元数据的一致性。注意：master的实例最少需要3个

系统特性

- 多租户，资源隔离
- 多个卷共享数据节点和元数据节点,每个卷独享各自的数据分片和元数据分片
- 与数据节点和元数据节点即有同步交互，也有异步交互，交互方式与任务类型相关。

配置参数

ChubaoFS 使用 **JSON** 作为配置文件的格式.

属性

配置项	类型	描述	是否必需	默认值
role	字符串	进程的角色，值只能是master	是	
ip	字符串	主机ip	是	
listen	字符串	http服务监听的端口号	是	
prof	字符串	golang pprof 端口号	是	
id	字符串	区分不同的master节点	是	
peers	字符串	raft复制组成员信息	是	
logDir	字符串	日志文件存储目录	是	
logLevel	字符串	日志级别	否	error
	字			

retainLogs	字符串	保留多少条raft日志.	是	
walDir	字符串	raft wal日志存储目录.	是	
storeDir	字符串	RocksDB数据存储目录.此目录必须存在，如果目录不存在，无法启动服务	是	
clusterName	字符串	集群名字	是	
exporterPort	整型	prometheus获取监控数据端口	否	
consulAddr	字符串	consul注册地址，供prometheus exporter使用	否	
metaNodeReservedMem	字符串	元数据节点预留内存大小，单位：字节	否	1073741824
heartbeatPort	字符串	raft心跳通信端口	否	5901
replicaPort	字符串	raft数据传输端口	否	5902
nodeSetCap	字符串	NodeSet的容量	否	18
missingDataPartitionInterval	字符串	当此时间段内没有收到副本的心跳，该副本被认为已丢失，单位：s	否	24h
dataPartitionTimeOutSec	字符串	当此时间段内没有收到副本的心跳，该副本被认为非存活，单位：s	否	10min
numberOfDataPartitionsToLoad	字符串	一次最多检查多少数据分片	否	40
secondsToFreeDataPartitionAfterLoad	字符串	在多少秒之后开始释放由加载数据分片任务占用的内存	否	300
tickInterval	字符串	检查心跳和选举超时的计时器间隔，单位：ms	否	500
electionTick	字符串	在计时器重置多少次时，选举超时	否	5

Example:

```
1.  {
2.    "role": "master",
```

```
  "id":"1",
  "ip": "10.196.59.198",
  "listen": "17010",
  "prof":"17020",
  "peers": "1:10.196.59.198:17010,2:10.196.59.199:17010,3:10.196.59.200:17010",
  "retainLogs":"20000",
  "logDir": "/cfs/master/log",
  "logLevel":"info",
  "walDir":"/cfs/master/data/wal",
  "storeDir":"/cfs/master/data/store",
  "exporterPort": 9500,
  "consulAddr": "http://consul.prometheus-cfs.local",
  "clusterName":"chubaofs01",
  "metaNodeReservedMem": "1073741824"
}
```

启动服务

```
nohup ./cfs-server -c master.json > nohup.out &
```

元数据节点

元数据节点是由多个元数据分片(meta partition)和基于multiRaft的对应个数的raft实例组成；每个元数据分片(meta partition)都是一个inode范围，且包含两个内存BTrees： inode BTree和dentry BTree。注意：MetaNode的实例最少需要3个

Properties

配置项	类型	描述	是否必需
role	字符串	进程角色： *MetaNode*	是
listen	字符串	监听和接受请求的端口	是
prof	字符串	调试和管理员API接口	是
logLevel	字符串	日志级别，默认： *error*	否
metadataDir	字符串	元数据快照存储目录	是
logDir	字符串	日志存储目录	是
raftDir	字符串	raft wal日志目录	是
raftHeartbeatPort	字符串	raft心跳通信端口	是
raftReplicaPort	字符串	raft数据传输端口	是
consulAddr	字符串	prometheus注册接口	否
exporterPort	字符串	prometheus获取监控数据端口	否
masterAddr	字符串	master服务地址	是
totalMem	字符串	最大可用内存，此值需高于master配置中metaNodeReservedMem的值，单位：字节	是
localIP	字符串	本机ip地址	否，如果不填写该选项，则使用和master通信的ip地址
zoneName	字符串	指定区域	否，默认分配至 `default` 区域
deleteBatchCount	int64	一次性批量删除多少inode节点，默认 `500`	否

样例：

```
{
      "role": "metanode",
      "listen": "17210",
      "prof": "17220",
      "logLevel": "debug",
      "localIP":"10.196.59.202",
      "metadataDir": "/cfs/metanode/data/meta",
      "logDir": "/cfs/metanode/log",
```

```
    "raftDir": "/cfs/metanode/data/raft",
    "raftHeartbeatPort": "17230",
    "raftReplicaPort": "17240",
    "consulAddr": "http://consul.prometheus-cfs.local",
    "exporterPort": 9501,
    "totalMem":  "8589934592",
    "masterAddr": [
        "10.196.59.198:17010",
        "10.196.59.199:17010",
        "10.196.59.200:17010"
    ]
}
```

启动服务

```
nohup ./cfs-server -c metanode.json > nohup.out &
```

注意事项

> - listen、raftHeartbeatPort、raftReplicaPort这三个配置选项在程序首次配置启动后，不能修改；
> - 相关的配置信息被记录在metadataDir目录下的constcfg文件中，如果需要强制修改，需要手动删除该文件；
> - 上述三个配置选项和MetaNode在master的注册信息有关。如果修改，将导致master无法定位到修改前的metanode信息；

数据节点

启动数据节点

通过执行ChubaoFS的二进制文件并用“-c”参数指定的配置文件来启动一个DATANODE进程。注意datanode的实例最少需要 4 个。

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

配置参数

Properties

关键字	参数类型	描述	是否必要
role	string	Role必须配置为“datanode”	是
listen	string	数据节点作为服务端启动TCP监听的端口	是
localIP	string	数据节点作为服务端选用的IP	否
prof	string	数据节点提供HTTP接口所用的端口	是
logDir	string	调测日志存放的路径	是
logLevel	string	调测日志的级别。默认是error	否
raftHeartbeat	string	RAFT发送节点间心跳消息所用的端口	是
raftReplica	string	RAFT发送日志消息所用的端口	是
raftDir	string	RAFT调测日志存放的路径。默认在二进制文件启动路径	否
consulAddr	string	监控系统的地址	否
exporterPort	string	监控系统的端口	否
masterAddr	string slice	集群管理器的地址	是
localIP	string	本机ip地址	否，如果不填写该选项，则使用和master通信的ip地址
zoneName	string	指定区域	否，默认分配至 `default` 区域
disks	string slice	格式：磁盘挂载路径:预留空间 预留空间配置范围[20G,50G]	是

举例：

```
1.  {
2.      "role": "datanode",
```

```
    "listen": "17310",
    "prof": "17320",
    "logDir": "/cfs/datanode/log",
    "logLevel": "info",
    "raftHeartbeat": "17330",
    "raftReplica": "17340",
    "raftDir": "/cfs/datanode/log",
    "consulAddr": "http://consul.prometheus-cfs.local",
    "exporterPort": 9502,
    "masterAddr": [
        "10.196.59.198:17010",
        "10.196.59.199:17010",
        "10.196.59.200:17010"
    ],
    "disks": [
        "/data0:10737418240",
        "/data1:10737418240"
    ]
}
```

注意事项

> - listen、raftHeartbeat、raftReplica这三个配置选项在程序首次配置启动后，不能修改；
> - 相关的配置信息被记录在raftDir目录下的constcfg文件中，如果需要强制修改，需要手动删除该文件；
> - 上述三个配置选项和datanode在master的注册信息有关。如果修改，将导致master无法定位到修改前的datanode信息；

对象存储(ObjectNode)

如何部署对象存储服务

通过执行ChubaoFS的二进制文件并用“-c”参数指定的配置文件来启动一个ObjectNode进程。

```
nohup cfs-server -c objectnode.json &
```

如果不打算使用对象存储功能，无需启动ObjectNode节点。

配置

对象管理节点使用 JSON 合适的配置文件

属性

参数	类型	描述	是否必需
role	string	进程角色，必须设置为 `objectnode`	是
listen	string	http服务监听的IP地址和端口号. 格式： `IP:PORT` 或者 `:PORT` 默认： `:80`	是
domains	string slice	为S3兼容接口配置域名以支持DNS风格访问资源 格式： `DOMAIN`	否
logDir	string	日志存放路径	是
logLevel	string	日志级别. 默认： `error`	否
masterAddr	string slice	格式： `HOST:PORT` . HOST: 资源管理节点IP（Master）. PORT: 资源管理节点服务端口（Master）	是
exporterPort	string	prometheus获取监控数据端口	否
prof	string	调试和管理员API接口	是

示例：

```
{
      "role": "objectnode",
      "listen": "17410",
      "domains": [
          "object.cfs.local"
      ],
      "logDir": "/cfs/Logs/objectnode",
```

```
    "logLevel": "info",
    "masterAddr": [
            "10.196.59.198:17010",
        "10.196.59.199:17010",
        "10.196.59.200:17010"
    ],
    "exporterPort": 9503,
    "prof": "7013"
}
```

获取鉴权密钥

鉴权秘钥由各用户所有，存储于用户信息中。

创建用户可以参见链接：用户管理命令。

如已创建用户，用户可以通过链接中的相关API获取用户信息，以获取鉴权密钥 *Access Key* 和 *Secret Key* 。

对象存储接口使用方法

对象子系统（ObjectNode）提供S3兼容的对象存储接口，所以可以直接使用原生的Amazon S3 SDKs来使用系统。

对象存储功能中，使用的 `Region` 变量为 集群名称 。

通过 **Supported S3-compatible APIs** 获取更详细的信息，地址： 对象存储 (ObjectNode)

通过 **Supported SDKs** 获取详细的SDK信息，地址： 对象存储 (ObjectNode)

Console

如何启动

通过执行使用-c参数构建的ChubaoFS的服务器二进制文件来启动控制台进程，并指定配置文件

```
1. nohup cfs-server -c console.json &
```

Configurations

Properties

Key	Type	Description	Mandatory
role	string	Role of process and must be set to *console*	Yes
logDir	string	Path for log file storage	Yes
logLevel	string	Level operation for logging. Default is *error*	No
listen	string	Port of TCP network to be listen, default is 80	Yes
masterAddr	string slice	Addresses of master server	Yes
objectNodeDomain	string	object domain for sign url for down	Yes
master_instance	string	the tag for monitor	Yes
monitor_addr	string	Prometheus the address	Yes
dashboard_addr	string	console menu forward to Grafana	Yes
monitor_app	string	the tag for monitor, it same as master config	Yes
monitor_cluster	string	the tag for monitor, it same as master config	Yes

Example:

```
1. {
2.   "role": "console",
3.   "logDir": "/cfs/log/",
4.   "logLevel": "debug",
5.   "listen": "80",
6.   "masterAddr": [
7.     "192.168.0.11:17010",
8.     "192.168.0.12:17010",
```

```
    "192.168.0.13:17010"
  ],
  "master_instance": "192.168.0.11:9066",
  "monitor_addr": "http://192.168.0.102:9090",
  "dashboard_addr": "http://192.168.0.103",
  "monitor_app": "cfs",
  "monitor_cluster": "spark"
}
```

Notice

> - 你可以通过这个网址来访问console http://127.0.0.1:80
> - 在console里默认用户名是 root 密码 ChubaoFSRoot
> - 如果你的系统是升级来的可能发生密码格式不兼容, 你可以通过 curl -H “Content-Type:application/json” -X POST –data ‘{“id”:”testuser”,”pwd”:”12345”,”type”:2}’ “http://10.196.59.198:17010/user/create” 创建一个新用户来登陆

客户端

环境依赖

- 插入内核FUSE模
- 安装libfuse

```
1. modprobe fuse
2. yum install -y fuse
```

配置文件

fuse.json

```
1. {
2.   "mountPoint": "/cfs/mountpoint",
3.   "volName": "ltptest",
4.   "owner": "ltptest",
5.   "masterAddr": "10.196.59.198:17010,10.196.59.199:17010,10.196.59.200:17010",
6.   "logDir": "/cfs/client/log",
7.   "logLevel": "info",
8.   "profPort": "27510"
9. }
```

配置选项

名称	类型	描述	必需
mountPoint	string	挂载点	是
volName	string	卷名称	是
owner	string	所有者	是
masterAddr	string	Master节点地址	是
logDir	string	日志存放路径	否
logLevel	string	日志级别: debug, info, warn, error	否
profPort	string	golang pprof调试端口	否
exporterPort	string	prometheus获取监控数据端口	否
consulAddr	string	监控注册服务器地址	否
lookupValid	string	内核FUSE lookup有效期, 单位: 秒	否
attrValid	string	内核FUSE attribute有效期, 单位: 秒	否

icacheTimeout	string	客户端inode cache有效期，单位：秒	否
enSyncWrite	string	使能DirectIO同步写，即DirectIO强制数据节点落盘	否
autoInvalData	string	FUSE挂载使用AutoInvalData选项	否
ronly	bool	以只读方式挂载，默认为false	否
writecache	bool	利用内核FUSE的写缓存功能，需要内核FUSE模块支持写缓存，默认为false	否
keepcache	bool	保留内核页面缓存。此功能需要启用writecache选项，默认为false	否
token	string	如果创建卷时开启了enableToken，此参数填写对应权限的token	否
readRate	int	限制每秒读取次数，默认无限制	否
writeRate	int	限制每秒写入次数，默认无限制	否
followerRead	bool	从follower中读取数据，默认为false	否
accessKey	string	卷所属用户的鉴权密钥	否
secretKey	string	卷所属用户的鉴权密钥	否
disableDcache	bool	禁用Dentry缓存，默认为false	否
subdir	string	设置子目录挂载	否
fsyncOnClose	bool	文件关闭后执行fsync操作，默认为true	否
maxcpus	int	最大可使用的cpu核数，可限制client进程cpu使用率	否
enableXattr	bool	是否使用*xattr*，默认是false	否

挂载

执行如下命令挂载客户端：

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

如果使用示例的``fuse.json``，则客户端被挂载到``/mnt/fuse``。所有针对``/mnt/fuse``的操作都将被作用于ChubaoFS。

卸载

建议使用标准的Linux `umount` 命令终止挂载。

授权节点

authnode负责授权客户端对ChubaoFS的Master节点的访问。通过此文档可以创建一个authnode docker-compose试用集群。

authnode功能的整体流程是：创建key -> 使用key获取访问指定服务的ticket -> 使用ticket访问服务。

编译构建

使用如下命令构建authtool及相关的依赖：

```
$ git clone http://github.com/chubaofs/chubaofs.git
$ cd chubaofs
$ make build
```

如果构建成功，将在 build/bin 目录中生成可执行文件 cfs-authtool 。

配置文件

- 创建anthnode的key：

  ```
  $ ./cfs-authtool authkey
  ```

  执行命令后，将在当前目录下生成 `authroot.json` 和 `authservice.json` 两个key文件。

  示例 `authservice.json` ：

  ```
  {
      "id": "AuthService",
      "key": "9h/sNq4+5CUAyCnAZM927Y/gubgmSixh5hpsYQzZG20=",
      "create_ts": 1573801212,
      "role": "AuthService",
      "caps": "{\"*\"}"
  }
  ```

- 在 docker/conf 目录下，编辑 `authnode.json` 配置文件：

  将 `authroot.json` 文件中的 `key` 值作为 `authRootKey` 的值。

将 `authservice.json` 文件中的 `key` 值作为 `authServiceKey` 的值。

示例 `authnode.json` ：

```
{
      "role": "authnode",
      "ip": "192.168.0.14",
      "port": "8080",
      "prof":"10088",
      "id":"1",
      "peers":
"1:192.168.0.14:8080,2:192.168.0.15:8081,3:192.168.0.16:8082",
      "retainLogs":"2",
      "logDir": "/export/Logs/authnode",
      "logLevel":"info",
      "walDir":"/export/Data/authnode/raft",
      "storeDir":"/export/Data/authnode/rocksdbstore",
      "exporterPort": 9510,
      "consulAddr": "http://consul.prometheus-cfs.local",
      "clusterName":"test",
      "authServiceKey":"9h/sNq4+5CUAyCnAZM927Y/gubgmSixh5hpsYQzZG20=",
      "authRootKey":"MTExMTExMTExMTExMTExMTExMTExMTExMTExMTExMTE=",
      "enableHTTPS":false
}
```

启动集群

在 docker/authnode 目录下，执行以下命令创建authnode集群

```
$ docker-compose up -d
```

使用授权功能

- 授权准备
  - 获取authService的ticket

```
$ ./cfs-authtool ticket -host=192.168.0.14:8080 -
keyfile=authservice.json -output=ticket_auth.json getticket
AuthService
```

输入：

> host：authnode的访问地址
>
> keyfile：需要获取ticket的用户key文件路径，是“创建key”操作输出的key文件

输出：

> output：存放ticket的文件路径

示例 `ticket_auth.json` ：

```
{
    "id": "AuthService",
    "session_key": "A9CSOGEN9CFYhnFnGwSMd4WFDBVbGmRNjaqGOhOinJE=",
    "service_id": "AuthService",
    "ticket":
"RDzEiRLX1xjoUyp2TDFviE/eQzXGlPO83siNJ3QguUrtpwiHIA3PLv4edyKzZdKcEb3wik
}
```

- 创建管理员用户

```
$ ./cfs-authtool api -host=192.168.0.14:8080 -
ticketfile=ticket_auth.json -data=data_admin.json -
output=key_admin.json AuthService createkey
```

输入：

> ticketfile：上一步骤所得ticket文件的路径，使用ticket才能访问相关服务
>
> data：需要注册的管理员用户信息

示例 `data_admin.json` ：

```
{
  "id": "admin",
  "role": "client",
  "caps": "{\"API\":[\"*:*:*\"]}"
}
```

输出：

> output: 管理员用户的key文件路径，key文件格式同前述操作所输出的key文件

- 管理员授权用户
  - 管理员获取ticket

```
$ ./cfs-authtool ticket -host=192.168.0.14:8080 -
keyfile=key_admin.json -output=ticket_admin.json getticket
AuthService
```

  - 管理员创建新的授权用户

```
$ ./cfs-authtool api -host=192.168.0.14:8080 -
ticketfile=ticket_admin.json -data=data_client.json -
output=key_client.json AuthService createkey
```

  - 授权用户获取访问服务的ticket

    例如，访问MasterService，可以执行以下命令获取ticket:

```
$ ./cfs-authtool ticket -host=192.168.0.14:8080 -
keyfile=key_client.json -output=ticket_client.json getticket
MasterService
```

在ChubaoFS集群中添加授权功能

- 为Master节点创建key

```
$ ./cfs-authtool api -host=192.168.0.14:8080 -ticketfile=ticket_admin.json
-data=data_master.json -output=key_master.json AuthService createkey
```

示例 `data_master` ：

```
{
    "id": "MasterService",
    "role": "service",
    "caps": "{\"API\":[\"*:*:*\"]}"
}
```

执行命令后，将 `key_master.json` 中 `key` 的值作为 `masterServiceKey` 的值写入配置文件 `master.json` 中。

- 为客户端创建key

```
$ ./cfs-authtool api -host=192.168.0.14:8080 -ticketfile=ticket_admin.json
-data=data_client.json -output=key_client.json AuthService createkey
```

示例 `data_client` ：

```
{
    "id": "ltptest",
    "role": "client",
    "caps": "{\"API\":[\"*:*:*\"]}"
}
```

参数说明：

> id: volname名称。
>
> role: 有client和service两种。
>
> caps: 格式为”{“API”:[“master:getVol:access”]}”，设为*表示所有API均可访问。

执行命令后，将 `key_client.json` 中 `key` 的值作为 `clientKey` 的值写入配置文件 `client.json` 中。

示例 `client.json` ：

```
{
    "masterAddr":
"192.168.0.11:17010,192.168.0.12:17010,192.168.0.13:17010",
    "mountPoint": "/cfs/mnt",
    "volName": "ltptest",
    "owner": "ltptest",
    "logDir": "/cfs/log",
    "logLevel": "info",
    "consulAddr": "http://192.168.0.100:8500",
    "exporterPort": 9500,
    "profPort": "17410",
    "authenticate": true,
    "ticketHost": "192.168.0.14:8080,192.168.0.15:8081,192.168.0.16:8082",
    "clientKey": "jgBGSNQp6mLbu7snU8wKIdEkytzl+pO5/OZOJPpIgH4=",
    "enableHTTPS": "false"
}
```

参数说明：

> authenticate：是否需要权限认证。设为true表示当前Vol需要进行权限认证。
>
> ticketHost：authnode集群的节点信息。
>
> clientKey：分发给client的key。
>
> enableHTTPS：是否使用https协议传输。

- 启动ChubaoFS集群

```
1.  $ docker/run_docker.sh -r -d /data/disk
```

在客户端的启动过程中，会先使用clientKey从authnode节点处获取访问Master节点的ticket，再使用ticket访问Master API。因此，只有被受权的客户端才能成功启动并挂载。

性能监控

ChubaoFS 集成了prometheus作为性能监控指标采集模块。在各模块配置文件中增加如下配置参数来启用该模块：

```
{
    "exporterPort": 9505,
    "consulAddr": "http://consul.prometheus-cfs.local"
}
```

- exporterPort: prometheus获取监控数据端口。设置后，可通过URL(http://$hostip:$exproterProt/metrics) 暴露prometheus监控指标。若不设置，prometheus指标监控模块将不会工作。
- consulAddr: consul注册服务器地址。设置后，可配合prometheus的自动发现机制实现ChubaoFS节点exporter的自动发现服务。若不设置，将不会启用consul自动注册服务。

可使用grafana作为prometheus 监控指标的展示前端，如下图所示：

可以通过prometheus alertmanager组件进行配置，来实现ChubaoFS系统的监控指标报警通知服务，详细可参考 alertmanager文档。

相关链接：

1. prometheus 安装配置

2. consul 安装配置
3. grafana 安装配置

监控指标

- 集群（Cluster）
  - 各节点数量：`MasterCount`，`MetaNodeCount`，`DataNodeCount`，`ObjectNodeCount`
  - 客户端数量：`ClientCount`
  - 卷数量：`VolumeCount`
  - 节点使用情况：`DataNodeSize`，`MetaNodeSize`
  - 节点使用率：`DataNodeUsedRatio`，`MetaNodeUsedRatio`
  - 非活动节点数量：`DataNodeInactive`，`MetaNodesInactive`
  - 卷总容量统计列表：`VolumeTotalSize`
  - 卷使用率统计列表：`VolumeUsedRatio`
  - 坏盘数量：`DiskError`
- 卷（Volume）
  - 卷使用量：`VolumeUsedSize`
  - 卷使用率：`VolumeUsedRatio`
  - 卷总容量变化率：`VolumeSizeRate`
- 资源管理节点（Master）
  - 无效master节点数量：`master_nodes_invalid`
  - 非活动元数据节点数量：`metanode_inactive`
  - 非活动数据节点数量：`datanode_inactive`
  - 非活动客户端数量：`fuseclient_inactive`
- 元数据节点（MetaNode）
  - 元数据节点上各操作的时长（Time）与每秒操作次数（Ops），可从`MetaNodeOp`下拉列表中选取监控指标。
- 数据节点（DataNode）
  - 数据节点上各操作的时长（Time）与每秒操作次数（Ops），可从`DataNodeOp`下拉列表中选取监控指标。
- 对象存储节点（ObjectNode）

> - 对象存储节点上各操作的时长（Time）与每秒操作次数（Ops），可从 `objectNodeOp` 下拉列表中选取监控指标。

- 客户端（FuseClient）

> - 客户端上各操作的时长（Time）与每秒操作次数（Ops），可从 `fuseOp` 下拉列表中选取监控指标。

推荐关注指标：集群状态相关、节点及磁盘故障相关、数据量、增长量等。

Grafana 监控面板配置模板

```
{
  "__inputs": [
    {
      "name": "DS_ChubaoFS01",
      "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"
    },
    {
      "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"
      ],
      "datasource": "${DS_ChubaoFS01}",
      "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
        },
        {
```

```
123.           "name": "range to text",
124.           "value": 2
125.         }
126.       ],
127.       "maxDataPoints": 100,
128.       "nullPointMode": "connected",
129.       "nullText": null,
130.       "postfix": "",
131.       "postfixFontSize": "50%",
132.       "prefix": "",
133.       "prefixFontSize": "50%",
134.       "rangeMaps": [
135.         {
136.           "from": "null",
137.           "text": "N/A",
138.           "to": "null"
139.         }
140.       ],
141.       "sparkline": {
142.         "fillColor": "rgba(31, 118, 189, 0.18)",
143.         "full": false,
144.         "lineColor": "rgb(31, 120, 193)",
145.         "show": true
146.       },
147.       "tableColumn": "",
148.       "targets": [
149.         {
             "expr": "count(go_info{cluster=~\"$cluster\", app=~\"$app\",
150. role=~\"master\"})",
151.           "format": "time_series",
152.           "intervalFactor": 1,
153.           "refId": "A"
154.         }
155.       ],
156.       "thresholds": "",
157.       "title": "master_count",
158.       "type": "singlestat",
159.       "valueFontSize": "80%",
160.       "valueMaps": [
161.         {
162.           "op": "=",
163.           "text": "N/A",
```

```
            "value": "null"
          }
        ],
        "valueName": "current"
      },
      {
        "cacheTimeout": null,
        "colorBackground": false,
        "colorValue": false,
        "colors": [
          "#299c46",
          "rgba(237, 129, 40, 0.89)",
          "#d44a3a"
        ],
        "datasource": "${DS_ChubaoFS01}",
        "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,
            "keepTime": true,
            "targetBlank": true,
            "title": "cfs-metanode",
            "type": "dashboard"
          }
        ],
```

```
206.       "mappingType": 1,
207.       "mappingTypes": [
208.         {
209.           "name": "value to text",
210.           "value": 1
211.         },
212.         {
213.           "name": "range to text",
214.           "value": 2
215.         }
216.       ],
217.       "maxDataPoints": 100,
218.       "nullPointMode": "connected",
219.       "nullText": null,
220.       "postfix": "",
221.       "postfixFontSize": "50%",
222.       "prefix": "",
223.       "prefixFontSize": "50%",
224.       "rangeMaps": [
225.         {
226.           "from": "null",
227.           "text": "N/A",
228.           "to": "null"
229.         }
230.       ],
231.       "sparkline": {
232.         "fillColor": "rgba(31, 118, 189, 0.18)",
233.         "full": false,
234.         "lineColor": "rgb(31, 120, 193)",
235.         "show": true
236.       },
237.       "tableColumn": "",
238.       "targets": [
239.         {
             "expr": "count(go_info{cluster=~\"$cluster\", app=~\"$app\",
240. role=~\"metanode\"})",
241.           "format": "time_series",
242.           "intervalFactor": 1,
243.           "refId": "A"
244.         }
245.       ],
246.       "thresholds": "",
```

```
      "title": "metanode_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_ChubaoFS01}",
      "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": "",
      "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_ChubaoFS01}",
          "fill": 1,
          "gridPos": {
            "h": 6,
            "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": [],
```

```
412.       "timeFrom": null,
413.       "timeShift": null,
414.       "title": "metanode_OpMetaPartition",
415.       "tooltip": {
416.         "shared": true,
417.         "sort": 0,
418.         "value_type": "individual"
419.       },
420.       "type": "graph",
421.       "xaxis": {
422.         "buckets": null,
423.         "mode": "time",
424.         "name": null,
425.         "show": true,
426.         "values": []
427.       },
428.       "yaxes": [
429.         {
430.           "format": "ns",
431.           "label": null,
432.           "logBase": 1,
433.           "max": null,
434.           "min": null,
435.           "show": true
436.         },
437.         {
438.           "format": "short",
439.           "label": null,
440.           "logBase": 1,
441.           "max": null,
442.           "min": null,
443.           "show": true
444.         }
445.       ],
446.       "yaxis": {
447.         "align": false,
448.         "alignLevel": null
449.       }
450.     },
451.     {
452.       "aliasColors": {},
453.       "bars": false,
```

```
454.       "dashLength": 10,
455.       "dashes": false,
456.       "datasource": "${DS_ChubaoFS01}",
457.       "fill": 1,
458.       "gridPos": {
459.         "h": 6,
460.         "w": 7,
461.         "x": 7,
462.         "y": 6
463.       },
464.       "id": 71,
465.       "legend": {
466.         "avg": false,
467.         "current": false,
468.         "max": false,
469.         "min": false,
470.         "show": true,
471.         "total": false,
472.         "values": false
473.       },
474.       "lines": true,
475.       "linewidth": 1,
476.       "links": [],
477.       "nullPointMode": "null",
478.       "percentage": false,
479.       "pointradius": 5,
480.       "points": false,
481.       "renderer": "flot",
482.       "seriesOverrides": [],
483.       "spaceLength": 10,
484.       "stack": false,
485.       "steppedLine": false,
486.       "targets": [
487.         {
             "expr":
488.  "sum(cfs_metanode_OpMetaBatchInodeGet{cluster=~\"$cluster\"})",
489.           "format": "time_series",
490.           "intervalFactor": 1,
491.           "legendFormat": "BatchGet",
492.           "refId": "A"
493.         },
494.         {
```

```
          "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",
        "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_ChubaoFS01}",
        "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",
      "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
        },
        {
          "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_ChubaoFS01}",
      "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=~\"$cluster\"})",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "CreateFile",
          "refId": "A"
        },
        {
          "expr":
"sum(cfs_dataNode_[[cluster]]_datanode_MarkDelete{cluster=~\"$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": [],
```

```
741.       "timeFrom": null,
742.       "timeShift": null,
743.       "title": "datanode_CreateFile",
744.       "tooltip": {
745.         "shared": true,
746.         "sort": 0,
747.         "value_type": "individual"
748.       },
749.       "type": "graph",
750.       "xaxis": {
751.         "buckets": null,
752.         "mode": "time",
753.         "name": null,
754.         "show": true,
755.         "values": []
756.       },
757.       "yaxes": [
758.         {
759.           "format": "ns",
760.           "label": null,
761.           "logBase": 1,
762.           "max": null,
763.           "min": null,
764.           "show": true
765.         },
766.         {
767.           "format": "short",
768.           "label": null,
769.           "logBase": 1,
770.           "max": null,
771.           "min": null,
772.           "show": true
773.         }
774.       ],
775.       "yaxis": {
776.         "align": false,
777.         "alignLevel": null
778.       }
779.     },
780.     {
781.       "aliasColors": {},
782.       "bars": false,
```

```
783.       "dashLength": 10,
784.       "dashes": false,
785.       "datasource": "${DS_ChubaoFS01}",
786.       "fill": 1,
787.       "gridPos": {
788.         "h": 6,
789.         "w": 7,
790.         "x": 7,
791.         "y": 12
792.       },
793.       "id": 75,
794.       "legend": {
795.         "avg": false,
796.         "current": false,
797.         "max": false,
798.         "min": false,
799.         "show": true,
800.         "total": false,
801.         "values": false
802.       },
803.       "lines": true,
804.       "linewidth": 1,
805.       "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=~\"$cluster\"}
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "OpLoadDataPartition",
          "refId": "G"
        },
        {
          "expr":
"sum(cfs_dataNode_[[cluster]]_datanode_OpDataNodeHeartbeat{cluster=~\"$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",
          "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_ChubaoFS01}",
      "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": [
        {
          "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"
      },
      "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_ChubaoFS01}",
        "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,
        "links": [],
        "nullPointMode": "null",
        "percentage": false,
        "pointradius": 5,
        "points": false,
        "renderer": "flot",
        "seriesOverrides": [],
        "spaceLength": 10,
```

```
246.       "stack": false,
247.       "steppedLine": false,
248.       "targets": [
249.         {
             "expr":
250.  "sum(cfs_dataNode_[[cluster]]_datanode_ExtentRepairRead{cluster=~\"$cluster\"})",
251.           "format": "time_series",
252.           "intervalFactor": 1,
253.           "legendFormat": "ExtentRepairRead",
254.           "refId": "B"
255.         },
256.         {
             "expr":
257.  "sum(cfs_dataNode_[[cluster]]_datanode_GetAllExtentWatermark{cluster=~\"$cluster\
258.           "format": "time_series",
259.           "intervalFactor": 1,
260.           "legendFormat": "GetAllExtentWatermark",
261.           "refId": "C"
262.         },
263.         {
             "expr":
264.  "sum(cfs_dataNode_[[cluster]]_datanode_NotifyExtentRepair{cluster=~\"$cluster\"})
265.           "format": "time_series",
266.           "intervalFactor": 1,
267.           "legendFormat": "NotifyExtentRepair",
268.           "refId": "D"
269.         }
270.       ],
271.       "thresholds": [],
272.       "timeFrom": null,
273.       "timeShift": null,
274.       "title": "datanode_Extent",
275.       "tooltip": {
276.         "shared": true,
277.         "sort": 0,
278.         "value_type": "individual"
279.       },
280.       "type": "graph",
281.       "xaxis": {
282.         "buckets": null,
283.         "mode": "time",
284.         "name": null,
285.         "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_ChubaoFS01}",
      "fill": 1,
      "gridPos": {
        "h": 7,
        "w": 7,
        "x": 7,
        "y": 18
      },
      "id": 83,
      "legend": {
        "avg": false,
        "current": false,
```

```
328.         "max": false,
329.         "min": false,
330.         "show": true,
331.         "total": false,
332.         "values": false
333.       },
334.       "lines": true,
335.       "linewidth": 1,
336.       "links": [],
337.       "nullPointMode": "null",
338.       "percentage": false,
339.       "pointradius": 5,
340.       "points": false,
341.       "renderer": "flot",
342.       "seriesOverrides": [],
343.       "spaceLength": 10,
344.       "stack": false,
345.       "steppedLine": false,
346.       "targets": [
347.         {
            "expr":
348.  "sum(cfs_dataNode_[[cluster]]_datanode_streamRead{cluster=~\"$cluster\"})",
349.           "format": "time_series",
350.           "intervalFactor": 1,
351.           "legendFormat": "streamRead",
352.           "refId": "M"
353.         },
354.         {
            "expr":
355.  "sum(cfs_dataNode_[[cluster]]_datanode_streamWrite{cluster=~\"$cluster\"})",
356.           "format": "time_series",
357.           "intervalFactor": 1,
358.           "legendFormat": "streamWrite",
359.           "refId": "N"
360.         },
361.         {
            "expr":
362.  "sum(cfs_dataNode_[[cluster]]_datanode_streamCreateFile{cluster=~\"$cluster\"})",
363.           "format": "time_series",
364.           "intervalFactor": 1,
365.           "legendFormat": "streamCreateFile",
366.           "refId": "A"
367.         },
```

```
368.         {
               "expr":
369.   "sum(cfs_dataNode_[[cluster]]_datanode_streamExtentRepairRead{cluster=~\"$cluster
370.           "format": "time_series",
371.           "intervalFactor": 1,
372.           "legendFormat": "streamExtentRepairRead",
373.           "refId": "B"
374.         },
375.         {
               "expr":
376.   "sum(cfs_dataNode_[[cluster]]_datanode_streamMarkDelete{cluster=~\"$cluster\"})",
377.           "format": "time_series",
378.           "intervalFactor": 1,
379.           "legendFormat": "streamMarkDelete",
380.           "refId": "C"
381.         },
382.         {
               "expr":
383.   "sum(cfs_dataNode_[[cluster]]_datanode_streamOpGetAppliedId{cluster=~\"$cluster\'
384.           "format": "time_series",
385.           "intervalFactor": 1,
386.           "legendFormat": "streamOpGetAppliedId",
387.           "refId": "D"
388.         },
389.         {
               "expr":
390.   "sum(cfs_dataNode_[[cluster]]_datanode_streamOpGetPartitionSize{cluster=~\"$clust
391.           "format": "time_series",
392.           "intervalFactor": 1,
393.           "legendFormat": "streamOpGetPartitionSize",
394.           "refId": "E"
395.         },
396.         {
               "expr":
397.   "sum(cfs_dataNode_[[cluster]]_datanode_streamNotifyExtentRepair{cluster=~\"$clust
398.           "format": "time_series",
399.           "intervalFactor": 1,
400.           "legendFormat": "streamNotifyExtentRepair",
401.           "refId": "F"
402.         }
403.       ],
404.       "thresholds": [],
405.       "timeFrom": null,
```

```
406.       "timeShift": null,
407.       "title": "datanode_Stream",
408.       "tooltip": {
409.         "shared": true,
410.         "sort": 0,
411.         "value_type": "individual"
412.       },
413.       "type": "graph",
414.       "xaxis": {
415.         "buckets": null,
416.         "mode": "time",
417.         "name": null,
418.         "show": true,
419.         "values": []
420.       },
421.       "yaxes": [
422.         {
423.           "format": "ns",
424.           "label": null,
425.           "logBase": 1,
426.           "max": null,
427.           "min": null,
428.           "show": true
429.         },
430.         {
431.           "format": "short",
432.           "label": null,
433.           "logBase": 1,
434.           "max": null,
435.           "min": null,
436.           "show": true
437.         }
438.       ],
439.       "yaxis": {
440.         "align": false,
441.         "alignLevel": null
442.       }
443.     },
444.     {
445.       "collapsed": false,
446.       "gridPos": {
447.         "h": 1,
```

```
        "w": 24,
        "x": 0,
        "y": 25
      },
      "id": 60,
      "panels": [],
      "title": "GoRuntime",
      "type": "row"
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_ChubaoFS01}",
      "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": {
        "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,
```

```
532.           "show": true
533.         },
534.         {
535.           "format": "short",
536.           "label": null,
537.           "logBase": 1,
538.           "max": null,
539.           "min": null,
540.           "show": true
541.         }
542.       ],
543.       "yaxis": {
544.         "align": false,
545.         "alignLevel": null
546.       }
547.     },
548.     {
549.       "aliasColors": {},
550.       "bars": false,
551.       "dashLength": 10,
552.       "dashes": false,
553.       "datasource": "${DS_ChubaoFS01}",
554.       "fill": 1,
555.       "gridPos": {
556.         "h": 6,
557.         "w": 7,
558.         "x": 7,
559.         "y": 26
560.       },
561.       "id": 62,
562.       "legend": {
563.         "avg": false,
564.         "current": false,
565.         "max": false,
566.         "min": false,
567.         "show": true,
568.         "total": false,
569.         "values": false
570.       },
571.       "lines": true,
572.       "linewidth": 1,
573.       "links": [],
```

```
574.       "nullPointMode": "null",
575.       "percentage": false,
576.       "pointradius": 5,
577.       "points": false,
578.       "renderer": "flot",
579.       "seriesOverrides": [],
580.       "spaceLength": 10,
581.       "stack": false,
582.       "steppedLine": false,
583.       "targets": [
584.         {
585.           "expr": "go_memstats_alloc_bytes{instance=~\"$instance\"}",
586.           "format": "time_series",
587.           "intervalFactor": 1,
588.           "legendFormat": "alloc_bytes",
589.           "refId": "A"
590.         },
591.         {
592.           "expr": "go_memstats_alloc_bytes_total{instance=~\"$instance\"}",
593.           "format": "time_series",
594.           "intervalFactor": 1,
595.           "legendFormat": "alloc_bytes_total",
596.           "refId": "B"
597.         },
598.         {
599.           "expr": "go_memstats_heap_alloc_bytes{instance=~\"$instance\"}",
600.           "format": "time_series",
601.           "intervalFactor": 1,
602.           "legendFormat": "heap_alloc_bytes",
603.           "refId": "C"
604.         },
605.         {
606.           "expr": "go_memstats_heap_inuse_bytes{instance=~\"$instance\"}",
607.           "format": "time_series",
608.           "intervalFactor": 1,
609.           "legendFormat": "heap_inuse_bytes",
610.           "refId": "D"
611.         },
612.         {
613.           "expr": "go_memstats_sys_bytes{instance=~\"$instance\"}",
614.           "format": "time_series",
615.           "intervalFactor": 1,
```

```
          "legendFormat": "sys_bytes",
          "refId": "E"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "go_memstats",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "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_ChubaoFS01}",
      "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,
      "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
```

```
        },
        {
          "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"
    },
    {
```

```
783.       "aliasColors": {},
784.       "bars": false,
785.       "dashLength": 10,
786.       "dashes": false,
787.       "datasource": "${DS_ChubaoFS01}",
788.       "fill": 1,
789.       "gridPos": {
790.         "h": 8,
791.         "w": 8,
792.         "x": 0,
793.         "y": 34
794.       },
795.       "id": 58,
796.       "legend": {
797.         "avg": false,
798.         "current": false,
799.         "max": false,
800.         "min": false,
801.         "show": true,
802.         "total": false,
803.         "values": false
804.       },
805.       "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,
          "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_ChubaoFS01}",
      "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=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "BatchGet",
          "refId": "A"
        },
        {
          "expr": "cfs_metanode_OpMetaCreateInode{instance=~\"$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": []
      },
      "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_ChubaoFS01}",
      "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,
      "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
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_ChubaoFS01}",
      "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"
        },
```

```
        {
          "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,
          "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_ChubaoFS01}",
      "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=~\"$instance\"}",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "Add",
          "refId": "A"
        },
        {
          "expr": "cfs_metanode_OpMetaExtentsList{instance=~\"$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
      }
    },
    {
```

```
503.       "collapsed": false,
504.       "gridPos": {
505.         "h": 1,
506.         "w": 24,
507.         "x": 0,
508.         "y": 50
509.       },
510.       "id": 27,
511.       "panels": [],
512.       "title": "Datanode",
513.       "type": "row"
514.     },
515.     {
516.       "aliasColors": {},
517.       "bars": false,
518.       "dashLength": 10,
519.       "dashes": false,
520.       "datasource": "${DS_ChubaoFS01}",
521.       "fill": 1,
522.       "gridPos": {
523.         "h": 8,
524.         "w": 8,
525.         "x": 0,
526.         "y": 51
527.       },
528.       "id": 28,
529.       "legend": {
530.         "avg": false,
531.         "current": false,
532.         "max": false,
533.         "min": false,
534.         "show": true,
535.         "total": false,
536.         "values": false
537.       },
538.       "lines": true,
539.       "linewidth": 1,
540.       "links": [],
541.       "nullPointMode": "null",
542.       "percentage": false,
543.       "pointradius": 5,
544.       "points": false,
```

```
545.       "renderer": "flot",
546.       "seriesOverrides": [],
547.       "spaceLength": 10,
548.       "stack": false,
549.       "steppedLine": false,
550.       "targets": [
551.         {
             "expr":
552.  "cfs_dataNode_[[cluster]]_datanode_CreateFile{instance=~\"$instance\"}",
553.           "format": "time_series",
554.           "intervalFactor": 1,
555.           "legendFormat": "CreateFile",
556.           "refId": "A"
557.         },
558.         {
             "expr":
559.  "cfs_dataNode_[[cluster]]_datanode_MarkDelete{instance=~\"$instance\"}",
560.           "format": "time_series",
561.           "intervalFactor": 1,
562.           "legendFormat": "MarkDelete",
563.           "refId": "B"
564.         }
565.       ],
566.       "thresholds": [],
567.       "timeFrom": null,
568.       "timeShift": null,
569.       "title": "datanode_CreateFile",
570.       "tooltip": {
571.         "shared": true,
572.         "sort": 0,
573.         "value_type": "individual"
574.       },
575.       "type": "graph",
576.       "xaxis": {
577.         "buckets": null,
578.         "mode": "time",
579.         "name": null,
580.         "show": true,
581.         "values": []
582.       },
583.       "yaxes": [
584.         {
585.           "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_ChubaoFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 8,
        "y": 51
      },
      "id": 74,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
```

```
628.       },
629.       "lines": true,
630.       "linewidth": 1,
631.       "links": [],
632.       "nullPointMode": "null",
633.       "percentage": false,
634.       "pointradius": 5,
635.       "points": false,
636.       "renderer": "flot",
637.       "seriesOverrides": [],
638.       "spaceLength": 10,
639.       "stack": false,
640.       "steppedLine": false,
641.       "targets": [
642.         {
             "expr":
643.  "cfs_dataNode_[[cluster]]_datanode_ExtentRepairRead{instance=~\"$instance\"}",
644.           "format": "time_series",
645.           "intervalFactor": 1,
646.           "legendFormat": "ExtentRepairRead",
647.           "refId": "B"
648.         },
649.         {
             "expr":
650.  "cfs_dataNode_[[cluster]]_datanode_GetAllExtentWatermark{instance=~\"$instance\"}
651.           "format": "time_series",
652.           "intervalFactor": 1,
653.           "legendFormat": "GetAllExtentWatermark",
654.           "refId": "C"
655.         },
656.         {
             "expr":
657.  "cfs_dataNode_[[cluster]]_datanode_NotifyExtentRepair{instance=~\"$instance\"}",
658.           "format": "time_series",
659.           "intervalFactor": 1,
660.           "legendFormat": "NotifyExtentRepair",
661.           "refId": "D"
662.         }
663.       ],
664.       "thresholds": [],
665.       "timeFrom": null,
666.       "timeShift": null,
667.       "title": "datanode_Extent",
```

```
668.       "tooltip": {
669.         "shared": true,
670.         "sort": 0,
671.         "value_type": "individual"
672.       },
673.       "type": "graph",
674.       "xaxis": {
675.         "buckets": null,
676.         "mode": "time",
677.         "name": null,
678.         "show": true,
679.         "values": []
680.       },
681.       "yaxes": [
682.         {
683.           "format": "ns",
684.           "label": null,
685.           "logBase": 1,
686.           "max": null,
687.           "min": null,
688.           "show": true
689.         },
690.         {
691.           "format": "short",
692.           "label": null,
693.           "logBase": 1,
694.           "max": null,
695.           "min": null,
696.           "show": true
697.         }
698.       ],
699.       "yaxis": {
700.         "align": false,
701.         "alignLevel": null
702.       }
703.     },
704.     {
705.       "aliasColors": {},
706.       "bars": false,
707.       "dashLength": 10,
708.       "dashes": false,
709.       "datasource": "${DS_ChubaoFS01}",
```

```
      "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,
      "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,
```

```
751.           "legendFormat": "OpDataNodeHeartbeat",
752.           "refId": "F"
753.         },
754.         {
             "expr":
755.  "cfs_dataNode_[[cluster]]_datanode_OpGetPartitionSize{instance=~\"$instance\"}",
756.           "format": "time_series",
757.           "intervalFactor": 1,
758.           "legendFormat": "OpGetPartitionSize",
759.           "refId": "H"
760.         },
761.         {
             "expr":
762.  "cfs_dataNode_[[cluster]]_datanode_OpGetAppliedId{instance=~\"$instance\"}",
763.           "format": "time_series",
764.           "intervalFactor": 1,
765.           "legendFormat": "OpGetAppliedId",
766.           "refId": "I"
767.         },
768.         {
             "expr":
769.  "cfs_dataNode_[[cluster]]_datanode_OpCreateDataPartition{instance=~\"$instance\"}
770.           "format": "time_series",
771.           "intervalFactor": 1,
772.           "legendFormat": "OpCreateDataPartition",
773.           "refId": "A"
774.         }
775.       ],
776.       "thresholds": [],
777.       "timeFrom": null,
778.       "timeShift": null,
779.       "title": "datanode_Op",
780.       "tooltip": {
781.         "shared": true,
782.         "sort": 0,
783.         "value_type": "individual"
784.       },
785.       "type": "graph",
786.       "xaxis": {
787.         "buckets": null,
788.         "mode": "time",
789.         "name": null,
790.         "show": true,
```

```
791.         "values": []
792.       },
793.       "yaxes": [
794.         {
795.           "format": "ns",
796.           "label": null,
797.           "logBase": 1,
798.           "max": null,
799.           "min": null,
800.           "show": true
801.         },
802.         {
803.           "format": "short",
804.           "label": null,
805.           "logBase": 1,
806.           "max": null,
```

```
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    },
    {
      "aliasColors": {},
      "bars": false,
      "dashLength": 10,
      "dashes": false,
      "datasource": "${DS_ChubaoFS01}",
      "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
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
```

```
42.       "spaceLength": 10,
43.       "stack": false,
44.       "steppedLine": false,
45.       "targets": [
46.         {
            "expr":
47. "cfs_dataNode_[[cluster]]_datanode_Read{instance=~\"$instance\"}",
48.           "format": "time_series",
49.           "intervalFactor": 1,
50.           "legendFormat": "Read",
51.           "refId": "J"
52.         },
53.         {
            "expr":
54. "cfs_dataNode_[[cluster]]_datanode_Write{instance=~\"$instance\"}",
55.           "format": "time_series",
56.           "intervalFactor": 1,
57.           "legendFormat": "Write",
58.           "refId": "K"
59.         },
60.         {
            "expr":
61. "cfs_dataNode_[[cluster]]_datanode_RandomWrite{instance=~\"$instance\"}",
62.           "format": "time_series",
63.           "intervalFactor": 1,
64.           "legendFormat": "RandomWrite",
65.           "refId": "L"
66.         }
67.       ],
68.       "thresholds": [],
69.       "timeFrom": null,
70.       "timeShift": null,
71.       "title": "datanode_IO",
72.       "tooltip": {
73.         "shared": true,
74.         "sort": 0,
75.         "value_type": "individual"
76.       },
77.       "type": "graph",
78.       "xaxis": {
79.         "buckets": null,
80.         "mode": "time",
81.         "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_ChubaoFS01}",
      "fill": 1,
      "gridPos": {
        "h": 8,
        "w": 8,
        "x": 8,
        "y": 59
      },
      "id": 77,
      "legend": {
        "avg": false,
```

```
124.         "current": false,
125.         "max": false,
126.         "min": false,
127.         "show": true,
128.         "total": false,
129.         "values": false
130.       },
131.       "lines": true,
132.       "linewidth": 1,
133.       "links": [],
134.       "nullPointMode": "null",
135.       "percentage": false,
136.       "pointradius": 5,
137.       "points": false,
138.       "renderer": "flot",
139.       "seriesOverrides": [],
140.       "spaceLength": 10,
141.       "stack": false,
142.       "steppedLine": false,
143.       "targets": [
144.         {
             "expr":
145. "cfs_dataNode_[[cluster]]_datanode_streamRead{instance=~\"$instance\"}",
146.           "format": "time_series",
147.           "intervalFactor": 1,
148.           "legendFormat": "streamRead",
149.           "refId": "M"
150.         },
151.         {
             "expr":
152. "cfs_dataNode_[[cluster]]_datanode_streamWrite{instance=~\"$instance\"}",
153.           "format": "time_series",
154.           "intervalFactor": 1,
155.           "legendFormat": "streamWrite",
156.           "refId": "N"
157.         },
158.         {
             "expr":
159. "cfs_dataNode_[[cluster]]_datanode_streamCreateFile{instance=~\"$instance\"}",
160.           "format": "time_series",
161.           "intervalFactor": 1,
162.           "legendFormat": "streamCreateFile",
163.           "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
          "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": [
        {
          "aliasColors": {},
          "bars": false,
          "dashLength": 10,
          "dashes": false,
          "datasource": "${DS_ChubaoFS01}",
          "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,
```

```
286.           "targets": [
287.             {
288.               "expr": "cfs_fuseclient_OpMetaOpen{instance=~\"$instance\"}",
289.               "format": "time_series",
290.               "intervalFactor": 1,
291.               "legendFormat": "Open",
292.               "refId": "B"
293.             },
294.             {
                   "expr":
295.  "cfs_fuseclient_OpMetaExtentsAdd{instance=~\"$instance\"}",
296.               "format": "time_series",
297.               "intervalFactor": 1,
298.               "legendFormat": "ExtentsAdd",
299.               "refId": "H"
300.             },
301.             {
                   "expr":
302.  "cfs_fuseclient_OpMetaExtentsList{instance=~\"$instance\"}",
303.               "format": "time_series",
304.               "intervalFactor": 1,
305.               "legendFormat": "ExtentsList",
306.               "refId": "I"
307.             },
308.             {
309.               "expr": "cfs_fuseclient_OpMetaReadDir{instance=~\"$instance\"}",
310.               "format": "time_series",
311.               "intervalFactor": 1,
312.               "legendFormat": "ReadDir",
313.               "refId": "K"
314.             },
315.             {
316.               "expr": "cfs_fuseclient_OpMetaSetattr{instance=~\"$instance\"}",
317.               "format": "time_series",
318.               "intervalFactor": 1,
319.               "legendFormat": "Setattr",
320.               "refId": "L"
321.             }
322.           ],
323.           "thresholds": [],
324.           "timeFrom": null,
325.           "timeShift": null,
326.           "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,
          "dashLength": 10,
          "dashes": false,
          "datasource": "${DS_ChubaoFS01}",
```

```
369.           "fill": 1,
370.           "gridPos": {
371.             "h": 6,
372.             "w": 7,
373.             "x": 7,
374.             "y": 85
375.           },
376.           "id": 67,
377.           "legend": {
378.             "avg": false,
379.             "current": false,
380.             "max": false,
381.             "min": false,
382.             "show": true,
383.             "total": false,
384.             "values": false
385.           },
386.           "lines": true,
387.           "linewidth": 1,
388.           "links": [],
389.           "nullPointMode": "null",
390.           "percentage": false,
391.           "pointradius": 5,
392.           "points": false,
393.           "renderer": "flot",
394.           "seriesOverrides": [],
395.           "spaceLength": 10,
396.           "stack": false,
397.           "steppedLine": false,
398.           "targets": [
399.             {
                 "expr":
400.  "cfs_fuseclient_OpMetaBatchInodeGet{instance=~\"$instance\"}",
401.               "format": "time_series",
402.               "intervalFactor": 1,
403.               "legendFormat": "BatchInodeGet",
404.               "refId": "A"
405.             },
406.             {
                 "expr":
407.  "cfs_fuseclient_OpMetaCreateInode{instance=~\"$instance\"}",
408.               "format": "time_series",
409.               "intervalFactor": 1,
```

```
410.                "legendFormat": "CreateInode",
411.                "refId": "D"
412.              },
413.              {
                    "expr":
414.  "cfs_fuseclient_OpMetaDeleteInode{instance=~\"$instance\"}",
415.                "format": "time_series",
416.                "intervalFactor": 1,
417.                "legendFormat": "DeleteInode",
418.                "refId": "F"
419.              },
420.              {
                    "expr":
421.  "cfs_fuseclient_OpMetaEvictInode{instance=~\"$instance\"}",
422.                "format": "time_series",
423.                "intervalFactor": 1,
424.                "legendFormat": "EvictInode",
425.                "refId": "G"
426.              },
427.              {
428.                "expr": "cfs_fuseclient_OpMetaInodeGet{instance=~\"$instance\"}",
429.                "format": "time_series",
430.                "intervalFactor": 1,
431.                "legendFormat": "InodeGet",
432.                "refId": "J"
433.              }
434.            ],
435.            "thresholds": [],
436.            "timeFrom": null,
437.            "timeShift": null,
438.            "title": "fuseclient_OpMetaInode",
439.            "tooltip": {
440.              "shared": true,
441.              "sort": 0,
442.              "value_type": "individual"
443.            },
444.            "type": "graph",
445.            "xaxis": {
446.              "buckets": null,
447.              "mode": "time",
448.              "name": null,
449.              "show": true,
450.              "values": []
```

```
451.           },
452.           "yaxes": [
453.             {
454.               "format": "ns",
455.               "label": null,
456.               "logBase": 1,
457.               "max": null,
458.               "min": null,
459.               "show": true
460.             },
461.             {
462.               "format": "short",
463.               "label": null,
464.               "logBase": 1,
465.               "max": null,
466.               "min": null,
467.               "show": true
468.             }
469.           ],
470.           "yaxis": {
471.             "align": false,
472.             "alignLevel": null
473.           }
474.         },
475.         {
476.           "aliasColors": {},
477.           "bars": false,
478.           "dashLength": 10,
479.           "dashes": false,
480.           "datasource": "${DS_ChubaoFS01}",
481.           "fill": 1,
482.           "gridPos": {
483.             "h": 6,
484.             "w": 7,
485.             "x": 14,
486.             "y": 85
487.           },
488.           "id": 68,
489.           "legend": {
490.             "avg": false,
491.             "current": false,
492.             "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"
```

```
534.           },
535.           "type": "graph",
536.           "xaxis": {
537.             "buckets": null,
538.             "mode": "time",
539.             "name": null,
540.             "show": true,
541.             "values": []
542.           },
543.           "yaxes": [
544.             {
545.               "format": "ns",
546.               "label": null,
547.               "logBase": 1,
548.               "max": null,
549.               "min": null,
550.               "show": true
551.             },
552.             {
553.               "format": "short",
554.               "label": null,
555.               "logBase": 1,
556.               "max": null,
557.               "min": null,
558.               "show": true
559.             }
560.           ],
561.           "yaxis": {
562.             "align": false,
563.             "alignLevel": null
564.           }
565.         },
566.         {
567.           "aliasColors": {},
568.           "bars": false,
569.           "dashLength": 10,
570.           "dashes": false,
571.           "datasource": "${DS_ChubaoFS01}",
572.           "fill": 1,
573.           "gridPos": {
574.             "h": 6,
575.             "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,
          "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
            }
          ],
          "yaxis": {
            "align": false,
            "alignLevel": null
          }
        }
      ],
      "title": "FuseClient",
```

```
659.       "type": "row"
660.     }
661.   ],
662.   "refresh": false,
663.   "schemaVersion": 16,
664.   "style": "dark",
665.   "tags": [],
666.   "templating": {
667.     "list": [
668.       {
669.         "allValue": null,
670.         "current": {
671.           "selected": true,
672.           "text": "cfs",
673.           "value": "cfs"
674.         },
675.         "hide": 2,
676.         "includeAll": false,
677.         "label": "App",
678.         "multi": false,
679.         "name": "app",
680.         "options": [
681.           {
682.             "selected": true,
683.             "text": "cfs",
684.             "value": "cfs"
685.           }
686.         ],
687.         "query": "cfs",
688.         "type": "custom"
689.       },
690.       {
691.         "allValue": null,
692.         "current": {},
693.         "datasource": "${DS_ChubaoFS01}",
694.         "hide": 0,
695.         "includeAll": false,
696.         "label": "Cluster",
697.         "multi": false,
698.         "name": "cluster",
699.         "options": [],
700.         "query": "label_values(go_info{app=~\"$app\"}, cluster)",
```

```
        "refresh": 1,
        "regex": "",
        "sort": 0,
        "tagValuesQuery": "",
        "tags": [],
        "tagsQuery": "",
        "type": "query",
        "useTags": false
      },
      {
        "allValue": null,
        "current": {},
        "datasource": "${DS_ChubaoFS01}",
        "hide": 0,
        "includeAll": false,
        "label": "Role",
        "multi": false,
        "name": "role",
        "options": [],
        "query": "label_values(go_info{app=~\"$app\", cluster=~\"$cluster\"},
role)",
        "refresh": 1,
        "regex": "",
        "sort": 0,
        "tagValuesQuery": "",
        "tags": [],
        "tagsQuery": "",
        "type": "query",
        "useTags": false
      },
      {
        "allValue": null,
        "current": {},
        "datasource": "${DS_ChubaoFS01}",
        "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": "${DS_ChubaoFS01}",
        "hide": 2,
        "includeAll": false,
        "label": "Host",
        "multi": false,
        "name": "hostip",
        "options": [],
        "query": "label_values(go_info{instance=~\"$instance\",
cluster=~\"$cluster\"}, instance)",
        "refresh": 1,
        "regex": "/([^:]+):.*/",
        "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
  }
```

优化配置FUSE参数

适当调整内核FUSE参数，能够在顺序写及高并发情况下获得更好的性能。具体可参考如下步骤：

1. 获取Linux内核源码

   下载对应的Linux内核源码包并且安装源码，源码安装目录为 `~/rpmbuild/BUILD/`

```
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
```

2. 优化Linux FUSE内核模块参数

   为了达到最优的性能，可以修改内核FUSE的参数 `FUSE_MAX_PAGES_PER_REQ` 和 `FUSE_DEFAULT_MAX_BACKGROUND` ，优化后的参考值如下：

```
/* fs/fuse/fuse_i.h */
#define FUSE_MAX_PAGES_PER_REQ 256
/* fs/fuse/inode.c */
#define FUSE_DEFAULT_MAX_BACKGROUND 32
```

3. 编译对应版本Linux内核模块

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

4. 插入内核模块

```
cp fuse.ko /lib/modules/`uname -r`/kernel/fs/fuse
rmmod fuse
depmod -a
modprobe fuse
```

yum工具自动部署集群

可以使用yum工具在CentOS 7+操作系统中快速部署和启动ChubaoFS集群. 该工具的rpm依赖项可通过以下命令安装:

```
$ yum install http://storage.jd.com/chubaofsrpm/latest/cfs-install-latest-el7.x86_64.rpm
$ cd /cfs/install
$ tree -L 3
 .
 ├── install_cfs.yml
 ├── install.sh
 ├── iplist
 ├── src
 └── template
     ├── client.json.j2
     ├── console.json.j2
     ├── create_vol.sh.j2
     ├── datanode.json.j2
     ├── grafana
     │   ├── grafana.ini
     │   ├── init.sh
     │   └── provisioning
     ├── master.json.j2
     ├── metanode.json.j2
     └── objectnode.json.j2
```

可根据实际环境，在 **iplist** 文件中修改ChubaoFS集群的参数.

- **[master]** , **[datanode]** , **[metanode]** , **[objectnode]**, **[console]**, **[monitor]** , **[client]** 包含了每个模块的成员IP地址。
- **[cfs:vars]** 模块定义了所有节点的ssh登陆信息，需要事先将集群中所有节点的登录名和密码进行统一。
- **#master config** 模块定义了每个Master节点的启动参数。

参数	类型	描述	是否必需
master_clusterName	字符串	集群名字	是

master_listen	字符串	http服务监听的端口号	是
master_prof	字符串	golang pprof 端口号	是
master_logDir	字符串	日志文件存储目录	是
master_logLevel	字符串	日志级别，默认为 *info*	否
master_retainLogs	字符串	保留多少条raft日志	是
master_walDir	字符串	raft wal日志存储目录	是
master_storeDir	字符串	RocksDB数据存储目录.此目录必须存在，如果目录不存在，无法启动服务	是
master_exporterPort	整型	prometheus获取监控数据端口	否
master_metaNodeReservedMem	字符串	元数据节点预留内存大小，如果剩余内存小于该值，MetaNode变为只读。单位：字节， 默认值：1073741824	否

- **#datanode config** 模块定义了每个DataNode的启动参数。

参数	类型	描述	是否必需
datanode_listen	字符串	数据节点作为服务端启动TCP监听的端口	是
datanode_prof	字符串	数据节点提供HTTP接口所用的端口	是
datanode_logDir	字符串	日志存放的路径	是
datanode_logLevel	字符串	日志的级别。默认是 *info*	否
datanode_raftHeartbeat	字符串	RAFT发送节点间心跳消息所用的端口	是
datanode_raftReplica	字符串	RAFT发送日志消息所用的端口	是
datanode_raftDir	字符串	RAFT调测日志存放的路径。默认在二进制文件启动路径	否
datanode_exporterPort	字符串	监控系统采集的端口	否
	字符	格式：*PATH:RETAIN*. PATH：磁盘挂载路径.	

	组	RETAIN：该路径下的最小预留空间，剩余空间小于该值即认为磁盘已满，单位：字节。（建议值：20G~50G）		

- **#metanode config** 模块定义了MetaNode的启动参数。

参数	类型	描述	是否必需	
metanode_listen	字符串	监听和接受请求的端口	是	
metanode_prof	字符串	调试和管理员API接口	是	
metanode_logLevel	字符串	日志级别，默认：*info*	否	
metanode_metadataDir	字符串	元数据快照存储目录	是	
metanode_logDir	字符串	日志存储目录	是	
metanode_raftDir	字符串	raft wal日志目录	是	
metanode_raftHeartbeatPort	字符串	raft心跳通信端口	是	
metanode_raftReplicaPort	字符串	raft数据传输端口	是	
metanode_exporterPort	字符串	prometheus获取监控数据端口	否	
metanode_totalMem	字符串	最大可用内存，此值需高于master配置中metaNodeReservedMem的值，单位：字节	是	

- **#objectnode config** 模块定义了ObjectNode的启动参数。

参数	类型	描述	是否必需
objectnode_listen	字符串	http服务监听的IP地址和端口号	是
objectnode_domains	字符串数组	为S3兼容接口配置域名以支持DNS风格访问资源格式： `DOMAIN`	否
objectnode_logDir	字符串	日志存放路径	是
objectnode_logLevel	字符串	日志级别. 默认： `error`	否
objectnode_exporterPort	字符串	prometheus获取监控数据端口	No

objectnode_enableHTTPS	字符串	是否支持 HTTPS协议	Yes

- **#console config** 模块定义了Console控制台的启动参数。

参数	类型	描述	是否必需
console_logDir	字符串	日志存放路径	是
console_logLevel	字符串	日志级别. 默认 *info*	否
console_listen	字符串	控制台服务端口，默认 80	是

- **#client config** 模块定义了fuse客户端的启动参数

参数	类型	描述	是否必需
client_mountPoint	字符串	挂载点	是
client_volName	字符串	卷名称	否
client_owner	字符串	卷所有者	是
client_SizeGB	字符串	如果卷不存在，则会创建一个该大小的卷，单位：GB	否
client_logDir	字符串	日志存放路径	是
client_logLevel	字符串	日志级别：*debug*, *info*, *warn*, *error*，默认 *info*	否
client_exporterPort	字符串	prometheus获取监控数据端口	是
client_profPort	字符串	golang pprof调试端口	否

```
[master]
10.196.59.198
10.196.59.199
10.196.59.200
[datanode]
...
[cfs:vars]
ansible_ssh_port=22
ansible_ssh_user=root
ansible_ssh_pass="password"
...
#master config
...
#datanode config
...
datanode_disks =  '"/data0:10737418240","/data1:10737418240"'
...
#metanode config
...
metanode_totalMem = "28589934592"
```

```
...
#objectnode config
...
#console config
...
```

更多配置介绍请参考 资源管理节点; 元数据节点; 数据节点; 对象存储(ObjectNode); 客户端; 性能监控; Console. 。

用 **install.sh** 脚本启动ChubaoFS集群，并确保首先启动Master。

```
$ bash install.sh -h
Usage: install.sh -r | --role [datanode | metanode | master | objectnode |
console | monitor | client | all | createvol ]
$ bash install.sh -r master
$ bash install.sh -r metanode
$ bash install.sh -r datanode
$ bash install.sh -r objectnode
$ bash install.sh -r console
$ bash install.sh -r monitor
$ bash install.sh -r client
```

全部角色启动后，可以登录到 **client** 角色所在节点验证挂载点 **/cfs/mountpoint** 是否已经挂载ChubaoFS文件系统。

在浏览器中打开链接http://consul.prometheus-cfs.local 查看监控系统(监控系统的IP地址已在 **iplist** 文件的 **[monitor]** 模块定义).

启动docker集群

在docker目录下，run_docker.sh工具用来方便运行ChubaoFS docker-compose试用集群。

执行下面的命令，可完全重新创建一个最小的ChubaoFS集群。注意的是**/data/disk**是数据根目录，至少需要30GB大小空闲空间。

```
1.  $ docker/run_docker.sh -r -d /data/disk
```

客户端启动成功后，在客户端docker容器中使用`mount`命令检查目录挂载状态：

```
1.  $ mount | grep chubaofs
```

在浏览器中打开http://127.0.0.1:3000，使用\`admin/123456\`登录，可查看chubaofs的grafana监控指标界面。

或者使用下面的命令分步运行：

```
1.  $ docker/run_docker.sh -b
2.  $ docker/run_docker.sh -s -d /data/disk
3.  $ docker/run_docker.sh -c
4.  $ docker/run_docker.sh -m
```

更多命令：

```
1.  $ docker/run_docker.sh -h
```

监控的Prometheus和Grafana相关配置位于`docker/monitor`目录下。

CSI插件支持

Chubaofs基于Container Storage Interface (CSI) (https://kubernetes-csi.github.io/docs/) 接口规范开发了CSI插件，以支持在Kubernetes集群中使用云存储。

cfscsi	kubernetes
v0.3.0	v1.12
v1.0.0	v1.15

Kubernetes v1.12

在Kubernetes v1.12集群中使用ChubaoFS CSI。

Kubernetes配置要求

为了在kubernetes集群中部署cfscsi插件，kubernetes集群需要满足以下配置。

kube-apiserver启动参数:

```
1.  --feature-gates=CSIPersistentVolume=true,MountPropagation=true
2.  --runtime-config=api/all
```

kube-controller-manager启动参数:

```
1.  --feature-gates=CSIPersistentVolume=true
```

kubelet启动参数:

```
    --feature-
1.  gates=CSIPersistentVolume=true,MountPropagation=true,KubeletPluginsWatcher=true
2.  --enable-controller-attach-detach=true
```

获取插件源码及脚本

```
1.  $ git clone -b csi-spec-v0.3.0 https://github.com/chubaofs/chubaofs-csi.git
2.  $ cd chubaofs-csi
```

拉取官方CSI镜像

```
1. docker pull quay.io/k8scsi/csi-attacher:v0.3.0
2. docker pull quay.io/k8scsi/driver-registrar:v0.3.0
3. docker pull quay.io/k8scsi/csi-provisioner:v0.3.0
```

获取cfscsi镜像

有两种方式可以实现。

- 从docker.io拉取镜像

```
1. docker pull docker.io/chubaofs/cfscsi:v0.3.0
```

- 根据源码编译镜像

```
1. make cfs-image
```

创建kubeconfig

```
1. kubectl create configmap kubecfg --from-file=pkg/cfs/deploy/kubernetes/kubecfg
```

创建RBAC和StorageClass

```
1. kubectl apply -f pkg/cfs/deploy/dynamic_provision/cfs-rbac.yaml
2. kubectl apply -f pkg/cfs/deploy/dynamic_provision/cfs-sc.yaml
```

部署cfscsi插件

- 方式一：将cfscsi ControllerServer和NodeServer绑定在同一个sidecar容器

修改 `pkg/cfs/deploy/dynamic_provision/sidecar/cfs-sidecar.yaml` 文件，将环境变量 `MASTER_ADDRESS` 设置为Chubaofs的实际Master地址，将 `<NodeServer IP>` 设置为kubernetes集群任意IP（如果被调度到该IP的pod需要动态挂载Chubaofs网盘，则必须为该IP部署cfscsi sidecar容器）。

```
1. kubectl apply -f pkg/cfs/deploy/dynamic_provision/sidecar/cfs-sidecar.yaml
```

- 方式二：将cfscsi插件ControllerServer和NodeServer分别部署为statefulset和daemonset（推荐此种）

修改 `pkg/cfs/deploy/dynamic_provision/independent` 文件夹下 `csi-controller-`

`statefulset.yaml` 和 `csi-node-daemonset.yaml` 文件，将环境变量 `MASTER_ADDRESS` 设置为Chubaofs的实际Master地址 ，将 `<ControllerServer IP>` 设置为kubernetes集群中任意节点IP。

为Kubernetes集群中的节点添加标签，拥有 `csi-role=controller` 标签的节点为ControllerServer。拥有 `csi-role=node` 标签的节点为NodeServer，也可以删除 `csi-node-daemonset.yaml` 文件中的 `nodeSelector` ，这样kubernetes集群所有节点均为NodeServer。

```
1. kubectl label nodes <ControllerServer IP> csi-role=controller
2. kubectl label nodes <NodeServer IP1> csi-role=node
3. kubectl label nodes <NodeServer IP2> csi-role=node
4. ...
```

部署:

```
   kubectl apply -f pkg/cfs/deploy/dynamic_provision/independent/csi-controller-
1. statefulset.yaml
   kubectl apply -f pkg/cfs/deploy/dynamic_provision/independent/csi-node-
2. daemonset.yaml
```

创建PVC

```
1. kubectl apply -f pkg/cfs/deploy/dynamic_provision/cfs-pvc.yaml
```

nginx动态挂载Chubaofs示例

```
1. docker pull nginx
2. kubectl apply -f pkg/cfs/deploy/dynamic_provision/pv-pod.yaml
```

Kubernetes v1.15+

在Kubernetes v1.15+ 集群中使用ChubaoFS CSI。

Kubernetes配置要求

为了在kubernetes集群中部署cfscsi插件，kubernetes api-server需要设置 `--allow-privileged=true` 。

从Kubernetes 1.13.0开始， `allow-privileged=true` 成为kubelet启动的默认值。参考CSI

官方github: https://kubernetes-csi.github.io/docs/deploying.html

准备一个ChubaoFS集群

ChubaoFS集群部署可参考 https://github.com/chubaofs/chubaofs.

获取插件源码及脚本

```
$ git clone https://github.com/chubaofs/chubaofs-csi.git
$ cd chubaofs-csi
```

ChubaoFS CSI插件部署

```
$ kubectl apply -f deploy/csi-controller-deployment.yaml
$ kubectl apply -f deploy/csi-node-daemonset.yaml
```

创建StorageClass

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: chubaofs-sc
provisioner: csi.chubaofs.com
reclaimPolicy: Delete
parameters:
  masterAddr: "master-service.chubaofs.svc.cluster.local:8080"
  owner: "csi-user"
  consulAddr: "consul-service.chubaofs.svc.cluster.local:8500"
  logLevel: "debug"
```

参数 `provisioner` 指定插件名称。这里设置为 `csi.chubaofs.com` ，kubernetes会将PVC的创建、挂载等任务调度给 `deploy/csi-controller-deployment.yaml` 和 `deploy/csi-node-daemonset.yaml` 中定义的ChubaoFS CSI插件去处理。

参数名	描述
MasterAddr	ChubaoFS Master地址
consulAddr	监控地址

```
$ kubectl create -f deploy/storageclass-chubaofs.yaml
```

创建PVC

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: chubaofs-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
  storageClassName: chubaofs-sc
```

`storageClassName` 需要和刚刚创建的StorageClass的 `metadata` 中的name属性保持一致。这样就会根据 `chubaofs-sc` 中定义的参数来创建存储卷。

```
$ kubectl create -f examples/pvc.yaml
```

在应用中挂载PVC

接下来就可以在你自己的应用中挂载刚刚创建的PVC到指定目录了。

```
...
spec:
  containers:
    - name: csi-demo
      image: alpine:3.10.3
      volumeMounts:
        - name: mypvc
          mountPath: /data
  volumes:
    - name: mypvc
      persistentVolumeClaim:
        claimName: chubaofs-pvc
...
```

```
$ kubectl create -f examples/deployment.yaml
```

- 资源管理节点

资源管理节点

Master

    - 查询
    - 下线副本
    - 比对副本文件
    - 磁盘下线
- 资源管理命令
    - 增加节点
    - 删除节点
- 用户管理命令
    - 创建用户
    - 删除用户
    - 查询用户信息
    - 查询用户列表
    - 更新用户信息
    - 用户授权
    - 移除用户权限
    - 转交卷

元数据节点

Metanode

- 元数据分片管理命令
    - 获取当前设备上所有分片信息
    - 获取指定分片ID的当前状态信息
- Inode管理命令
    - 获取指定Inode基本信息
    - 获取指定Inode的数据存储信息
    - 获取指定元数据分片的全部inode信息
- Dentry调试命令
    - 获取Dentry信息
    - 获取指定目录下全部文件
    - 获取指定分片的全部目录信息

使用CLI工具管理集群

CLI

- CLI工具配置及使用方法
    - 编译及配置
    - 使用方法

集群管理命令

概述

```
curl -v "http://10.196.59.198:17010/admin/getCluster" | python -m json.tool
```

展示集群基本信息，比如集群包含哪些数据节点和元数据节点，卷等。

响应示例

```
{
    "Name": "test",
    "LeaderAddr": "10.196.59.198:17010",
    "DisableAutoAlloc": false,
    "Applied": 225,
    "MaxDataPartitionID": 100,
    "MaxMetaNodeID": 3,
    "MaxMetaPartitionID": 1,
    "DataNodeStatInfo": {},
    "MetaNodeStatInfo": {},
    "VolStatInfo": {},
    "BadPartitionIDs": {},
    "BadMetaPartitionIDs": {},
    "MetaNodes": {},
    "DataNodes": {}
}
```

冻结集群

```
curl -v "http://10.196.59.198:17010/cluster/freeze?enable=true"
```

如果启用了冻结集群功能，卷就不再自动地创建数据分片。

参数列表

参数	类型	描述
enable	bool	如果设置为true，则集群被冻结

获取集群空间信息

```
curl -v "http://10.196.59.198:17010/cluster/stat"
```

按区域展示集群的空间信息。

响应示例

```
{
    "DataNodeStatInfo": {
        "TotalGB": 1,
        "UsedGB": 0,
        "IncreasedGB": -2,
        "UsedRatio": "0.0"
    },
    "MetaNodeStatInfo": {
        "TotalGB": 1,
        "UsedGB": 0,
        "IncreasedGB": -8,
        "UsedRatio": "0.0"
    },
    "ZoneStatInfo": {
        "zone1": {
            "DataNodeStat": {
                "TotalGB": 1,
                "UsedGB": 0,
                "AvailGB": 0,
                "UsedRatio": 0,
                "TotalNodes": 0,
                "WritableNodes": 0
            },
            "MetaNodeStat": {
                "TotalGB": 1,
                "UsedGB": 0,
                "AvailGB": 0,
                "UsedRatio": 0,
                "TotalNodes": 0,
                "WritableNodes": 0
            }
        }
    }
}
```

获取集群的拓扑信息

```
curl -v "http://10.196.59.198:17010/topo/get"
```

按区域展示集群的拓扑信息。

响应示例

```
[
    {
        "Name": "zone1",
        "Status": "available",
        "NodeSet": {
            "700": {
                "DataNodeLen": 0,
                "MetaNodeLen": 0,
                "MetaNodes": [],
                "DataNodes": []
            }
        }
    },
    {
        "Name": "zone2",
        "Status": "available",
        "NodeSet": {
            "800": {
                "DataNodeLen": 0,
                "MetaNodeLen": 0,
                "MetaNodes": [],
                "DataNodes": []
            }
        }
    }
]
```

更新可用区状态

```
curl -v "http://10.196.59.198:17010/zone/update?name=zone1&enable=false"
```

更新可用区的状态为可用或不可用。

参数列表

参数	类型	描述
name	string	可用区名称
enable	bool	true表示可用，false为不可用

获取所有可用区信息

```
curl -v "http://10.196.59.198:17010/zone/list"
```

获取所有可用区的名称及可用状态。

响应示例

```
[
    {
        "Name": "zone1",
        "Status": "available",
        "NodeSet": {}
    },
    {
        "Name": "zone2",
        "Status": "available",
        "NodeSet": {}
    }
]
```

获取节点信息

```
curl -v "http://192.168.0.11:17010/admin/getNodeInfo"
```

获取metanode、datanode节点信息

响应示例

```
{
    "code": 0,
    "msg": "success",
    "data": {
        "batchCount": 0,
        "markDeleteRate": 0
```

```
    }
}
```

设置节点信息

```
curl -v "http://192.168.0.11:17010/admin/setNodeInfo?batchCount=100&markDeleteRate=100"
```

设置metanode、datanode节点信息

参数列表

参数	类型	描述
batchCount	uint64	metanode 删除批量大小
markDeleteRate	uint64	datanode批量删除限速设置. 0代表未做限速设置

元数据节点管理命令

查询

```
curl -v "http://10.196.59.198:17010/metaNode/get?addr=10.196.59.202:17210"  |
python -m json.tool
```

展示元数据节点的详细信息，包括地址、总的内存大小、已使用内存大小等等。

参数列表

参数	类型	描述
addr	string	元数据节点和master的交互地址

响应示例

```
{
    "ID": 3,
    "Addr": "10.196.59.202:17210",
    "IsActive": true,
    "Zone": "zone1",
    "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,
    "NodeSetID": 2,
    "PersistenceMetaPartitions": {}
}
```

下线节点

```
curl -v "http://10.196.59.198:17010/metaNode/decommission?
addr=10.196.59.202:17210"
```

从集群中下线某个元数据节点， 该节点上的所有元数据分片都会被异步的迁移到集群中其它可用的元数

据节点。

参数列表

参数	类型	描述
addr	string	元数据节点和master的交互地址

设置阈值

```
curl -v "http://10.196.59.198:17010/threshold/set?threshold=0.75"
```

如果某元数据节点内存使用率达到这个阈值，则该节点上所有的元数据分片都会被设置为只读。

参数列表

参数	类型	描述
threshold	float64	元数据节点能使用本机内存的最大比率

数据节点管理命令

查询

```
curl -v "http://10.196.59.198:17010/dataNode/get?addr=10.196.59.201:17310"  |
python -m json.tool
```

显示数据节点的详情，包括数据节点的地址、总的容量、已使用空间等等。

参数列表

参数	类型	描述
addr	string	数据节点和master的交互地址

响应示例

```
{
    "TotalWeight": 39666212700160,
    "UsedWeight": 2438143586304,
    "AvailableSpace": 37228069113856,
    "ID": 2,
    "Zone": "zone1",
    "Addr": "10.196.59.201:17310",
    "ReportTime": "2018-12-06T10:56:38.881784447+08:00",
    "IsActive": true
    "UsageRatio": 0.06146650815226848,
    "SelectTimes": 5,
    "Carry": 1.0655859145960367,
    "DataPartitionReports": {},
    "DataPartitionCount": 21,
    "NodeSetID": 3,
    "PersistenceDataPartitions": {},
    "BadDisks": {}
}
```

下线节点

```
curl -v "http://10.196.59.198:17010/dataNode/decommission?
addr=10.196.59.201:17310"
```

从集群中下线某个数据节点，该数据节点上的所有数据分片都会被异步的迁移到集群中其它可用的数据节点

参数列表

参数	类型	描述
addr	string	数据节点和master的交互地址

卷管理命令

创建

```
curl -v "http://10.196.59.198:17010/admin/createVol?
name=test&capacity=100&owner=cfs&mpCount=3"
```

为用户创建卷，并分配一组数据分片和元数据分片． 在创建新卷时，默认分配10个数据分片和3个元数据分片。

ChubaoFS以 **Owner** 参数作为用户ID。在创建卷时，如果集群中没有与该卷的Owner同名的用户时，会自动创建一个用户ID为Owner的用户；如果集群中已存在用户ID为Owner的用户，则会自动将该卷的所有权归属于该用户。详情参阅： 用户管理命令

参数列表

参数	类型	描述	是否必需	默认值
name	string	卷名称	是	无
capacity	int	卷的配额,单位是GB	是	无
owner	string	卷的所有者，同时也是用户ID	是	无
mpCount	int	初始化元数据分片个数	否	3
size	int	数据分片大小，单位GB	否	120
followerRead	bool	允许从follower读取数据	否	false
crossZone	bool	是否跨区域，如设为true，则不能设置zoneName参数	否	false
zoneName	string	指定区域	否	如果crossZone设为false，则默认值为default
enableToken	bool	是否开启token控制读写权限	否	false

删除

```
curl -v "http://10.196.59.198:17010/vol/delete?name=test&authKey=md5(owner)"
```

首先把卷标记为逻辑删除（status设为1）， 然后通过周期性任务删除所有数据分片和元数据分片,最终从持久化存储中删除。

在删除卷的同时，将会在所有用户的信息中删除与该卷有关的权限信息。

参数列表

参数	类型	描述
name	string	卷名称
authKey	string	计算vol的所有者字段的32位MD5值作为认证信息

查询

```
curl -v "http://10.196.59.198:17010/client/vol?name=test&authKey=md5(owner)" |
python -m json.tool
```

展示卷的基本信息，包括卷的名字、所有的数据分片和元数据分片信息等。

参数列表

参数	类型	描述
name	string	卷名称
authKey	string	计算vol的所有者字段的32位MD5值作为认证信息

响应示例

```
{
    "Name": "test",
    "Owner": "user",
    "Status": "0",
    "FollowerRead": "true",
    "MetaPartitions": {},
    "DataPartitions": {},
    "CreateTime": 0
}
```

统计

```
curl -v http://10.196.59.198:17010/client/volStat?name=test
```

展示卷的总空间大小、已使用空间大小及是否开启读写token控制的信息。

参数列表

参数	类型	描述
name	string	卷名称

响应示例

```
{
    "Name": "test",
    "TotalSize": 322122547200000000,
    "UsedSize": 155515112832780000,
    "UsedRatio": "0.48",
    "EnableToken": true
}
```

更新

```
curl -v "http://10.196.59.198:17010/vol/update?name=test&capacity=100&authKey=md5(owner)"
```

增加卷的配额，也可调整其它相关参数。

参数列表

参数	类型	描述	是否必需
name	string	卷名称	是
authKey	string	计算vol的所有者字段的32位MD5值作为认证信息	是
capacity	int	扩充后卷的配额,单位是GB	是
zoneName	string	更新后所在区域，若不设置将被更新至default区域	是
enableToken	bool	是否开启token控制读写权限，默认设为 `false`	否
followerRead	bool	允许从follower读取数据	否

获取卷列表

```
curl -v "http://10.196.59.198:17010/vol/list?keywords=test"
```

获取全部卷的列表信息，可按关键字过滤。

参数列表

参数	类型	描述	是否必需
keywords	string	获取卷名包含此关键字的卷信息	否

响应示例

```
[
    {
        "Name": "test1",
        "Owner": "cfs",
```

```
        "CreateTime": 0,
        "Status": 0,
        "TotalSize": 155515112832780000,
        "UsedSize": 155515112832780000
    },
    {
        "Name": "test2",
        "Owner": "cfs",
        "CreateTime": 0,
        "Status": 0,
        "TotalSize": 155515112832780000,
        "UsedSize": 155515112832780000
    }
]
```

添加token

```
curl -v "http://10.196.59.198:17010/token/add?
name=test&tokenType=1&authKey=md5(owner)"
```

添加控制读写权限的token。

参数列表

参数	类型	描述
name	string	卷名称
authKey	string	计算vol的所有者字段的32位MD5值作为认证信息
tokenType	int	1代表只读token，2代表读写token

更新token

```
curl -v "http://10.196.59.198:17010/token/update?
name=test&token=xx&tokenType=1&authKey=md5(owner)"
```

更新token类型。

参数列表

参数	类型	描述
name	string	卷名称
authKey	string	计算vol的所有者字段的32位MD5值作为认证信息
tokenType	int	1代表只读token，2代表读写token

token	string	token值

删除token

```
curl -v "http://10.196.59.198:17010/token/delete?name=test&token=xx&authKey=md5(owner)"
```

删除指定token。

参数列表

参数	类型	描述
name	string	卷名称
authKey	string	计算vol的所有者字段的32位MD5值作为认证信息
token	string	待删除的token值

获取token类型

```
curl -v "http://10.196.59.198:17010/token/get?name=test&token=xx"
```

获取指定token的类型。

参数列表

参数	类型	描述
name	string	卷名称
token	string	token值

响应示例

```
{
    "TokenType":2,
    "Value":"siBtuF9hbnNqXzJfMTU48si3nzU4MzE1Njk5MDM1NQ==",
    "VolName":"test"
}
```

元数据分片管理命令

创建

```
curl -v "http://10.196.59.198:17010/metaPartition/create?name=test&start=10000"
```

手动切分元数据分片，如果卷的最大的元数据分片 **inode** 的范围是 `[0, end)` ， **end** 大于 **start** 参数，原来最大的元数据分片的inode范围变为 `[0, start]` ，新创建的元数据分片的范围是 `[start+1,end)` 。

参数列表

参数	类型	描述
name	string	卷的名字
start	uint64	根据此值切分元数据分片

查询

```
curl -v "http://10.196.59.198:17010/metaPartition/get?id=1" | python -m
json.tool
```

展示元数据分片的详细信息，包括分片ID，分片的起始范围等等。

参数列表

参数	类型	描述
id	uint64	元数据分片ID

响应示例

```
{
    "PartitionID": 1,
    "Start": 0,
    "End": 9223372036854776000,
    "MaxNodeID": 1,
    "VolName": "test",
    "Replicas": {},
    "ReplicaNum": 3,
    "Status": 2,
    "IsRecover": true,
    "Hosts": {},
```

```
    "Peers": {},
    "Zones": {},
    "MissNodes": {},
    "LoadResponse": {}
}
```

下线副本

```
curl -v "http://10.196.59.198:17010/metaPartition/decommission?id=13&addr=10.196.59.202:17210"
```

下线元数据分片的某个副本，并且创建一个新的副本。

参数列表

参数	类型	描述
id	uint64	元数据分片ID
addr	string	要下线副本的地址

比对副本

```
curl -v "http://10.196.59.198:17010/metaPartition/load?id=1"
```

发送比对副本任务到各个副本，然后检查各个副本的Crc是否一致。

参数列表

参数	类型	描述
id	uint64	元数据分片ID

数据分片管理命令

创建

```
curl -v "http://10.196.59.198:17010/dataPartition/create?count=400&name=test"
```

创建指定数量的数据分片。

参数列表

参数	类型	描述
count	int	创建多少个数据分片
name	string	卷的名字

查询

```
curl -v "http://10.196.59.198:17010/dataPartition/get?id=100"  | python -m
json.tool
```

展示数据分片的详细信息，包括副本数量、卷信息等。

参数列表

参数	类型	描述
id	uint64	数据分片的ID

响应示例

```
{
    "PartitionID": 100,
    "LastLoadedTime": 1544082851,
    "ReplicaNum": 3,
    "Status": 2,
    "Replicas": {},
    "Hosts": {},
    "Peers": {},
    "Zones": {},
    "MissingNodes": {},
    "VolName": "test",
    "VolID": 2,
    "FileInCoreMap": {},
```

```
14.     "FilesWithMissingReplica": {}
15. }
```

下线副本

```
    curl -v "http://10.196.59.198:17010/dataPartition/decommission?
1.  id=13&addr=10.196.59.201:17310"
```

移除数据分片的某个副本，并且创建一个新的副本。

参数列表

参数	类型	描述
id	uint64	数据分片的ID
addr	string	要下线的副本的地址

比对副本文件

```
1.  curl -v "http://10.196.59.198:17010/dataPartition/load?id=1"
```

给数据分片的每个副本都发送比对副本文件的任务，然后异步的检查每个副本上的文件crc是否一致。

参数列表

参数	类型	描述
id	uint64	数据分片的ID

磁盘下线

```
    curl -v "http://10.196.59.198:17010/disk/decommission?
1.  addr=10.196.59.201:17310&disk=/cfs1"
```

同步下线磁盘上的所有数据分片，并且为每一个数据分片在集群内创建一个新的副本。

参数列表

参数	类型	描述
addr	string	要下线的副本的地址
disk	string	故障磁盘

资源管理命令

增加节点

```
curl -v "http://10.196.59.198:17010/raftNode/add?addr=10.196.59.197:17010&id=3"
```

增加新的master节点到raft复制组。

参数列表

参数	类型	描述
addr	string	master的ip地址, 格式为ip:port
id	uint64	master的节点标识

删除节点

```
curl -v "http://10.196.59.198:17010/raftNode/remove?
addr=10.196.59.197:17010&id=3"
```

从raft复制组中移除某个节点。

参数列表

参数	类型	描述
addr	string	master的ip地址, 格式为ip:port
id	uint64	master的节点标识

用户管理命令

创建用户

```
    curl -H "Content-Type:application/json" -X POST --data
    '{"id":"testuser","pwd":"12345","type":3}'
1.  "http://10.196.59.198:17010/user/create"
```

在集群中创建用户，用于访问对象存储功能。在集群启动时，会自动创建root用户(type值为0x1)。

ChubaoFS将卷的 **Owner** 字段看作一个用户ID。例如，创建卷时Owner取值为 *testuser* 的话，则该卷自动归为用户 *testuser* 的名下。

如果创建卷时不存在与Owner取值相同的用户ID，则创建卷时会自动创建用户ID取值为Owner的用户。

body参数列表

参数	类型	描述	取值范围	是否必需	默认值
id	string	用户ID	由字母、数字及下划线组成，不超过20个字符	是	无
pwd	string	用户密码	无限制	否	ChubaoFSUser
ak	string	用于对象存储功能的Access Key	由16位字母及数字组成	否	系统随机生成
sk	string	用于对象存储功能的Secret Key	由32位字母及数字组成	否	系统随机生成
type	int	用户类型	2（管理员）/3（普通用户）	是	无

删除用户

```
1.  curl -v "http://10.196.59.198:17010/user/delete?user=testuser"
```

在集群中删除指定的用户。

参数列表

参数	类型	描述
user	string	用户ID

查询用户信息

展示的用户基本信息,包括用户ID、Access Key、Secret Key、名下的卷列表、其他用户授予的权

限列表、用户类型、创建时间等。

用户信息中 **policy** 字段表示该用户拥有权限的卷，其中 **own_vols** 表示所有者是该用户的卷，**authorized_vols** 表示其他用户授权给该用户的卷，及所拥有的权限限制。

有以下两种方式获取:

通过用户ID查询

```
curl -v "http://10.196.59.198:17010/user/info?user=testuser" | python -m
json.tool
```

参数列表

参数	类型	描述
user	string	用户ID

通过Access Key查询

```
curl -v "http://10.196.59.198:17010/user/akInfo?ak=0123456789123456" | python -
m json.tool
```

参数列表

参数	类型	描述
ak	string	该用户的16位Access Key

响应示例

```
{
      "user_id": "testuser",
      "access_key": "gDcKaBvqky4g8StT",
      "secret_key": "ZVY5RHlrnOrCjImW9S3MajtYZyxSegcf",
      "policy": {
          "own_vols": ["vol1"],
          "authorized_vols": {
              "ltptest": [
                  "perm:builtin:ReadOnly",
                  "perm:custom:PutObjectAction"
              ]
          }
      },
      "user_type": 3,
      "create_time": "2020-05-11 09:25:04"
```

```
}
```

查询用户列表

```
curl -v "http://10.196.59.198:17010/user/list?keywords=test" | python -m
json.tool
```

查询集群中包含某关键字的所有用户的信息。

参数列表

参数	类型	描述
keywords	string	查询用户ID包含此关键字的用户信息

更新用户信息

```
curl -H "Content-Type:application/json" -X POST --data
'{"user_id":"testuser","access_key":"KzuIVYCFqvu0b3Rd","secret_key":"iaawlCchJeeu
"http://10.196.59.198:17010/user/update"
```

更新指定UserID的用户信息，可修改的内容包括Access Key、Secret Key和用户类型。

body参数列表

参数	类型	描述	是否必需
user_id	string	待更新信息的用户ID	是
access_key	string	更新后的Access Key取值	否
secret_key	string	更新后的Secret Key取值	否
type	int	更新后的用户类型	否

用户授权

```
curl -H "Content-Type:application/json" -X POST --data
'{"user_id":"testuser","volume":"vol","policy":
["perm:builtin:ReadOnly","perm:custom:PutObjectAction"]}'
"http://10.196.59.198:17010/user/updatePolicy"
```

更新指定用户对于某个卷的访问权限。 **policy** 的取值有三类:

- 授予只读或读写权限，取值为 `perm:builtin:ReadOnly` 或 `perm:builtin:Writable` ；
- 授予指定操作的权限，格式为 `action:oss:XXX` ，以 *GetObject* 操作为例，policy取值为

action:oss:GetObject ;

- 授予自定义权限，格式为 `perm:custom:XXX` ，其中 *XXX* 由用户自定义。

指定权限后，用户在使用对象存储功能时，仅能在指定权限范围内对卷进行访问。如果该用户已有对此卷的权限设置，则本操作会覆盖原有权限。

body参数列表

参数	类型	描述	是否必需
user_id	string	待设置权限的用户ID	是
volume	string	待设置权限的卷名	是
policy	string slice	待设置的权限	是

移除用户权限

```
    curl -H "Content-Type:application/json" -X POST --data
    '{"user_id":"testuser","volume":"vol"}'
1.  "http://10.196.59.198:17010/user/removePolicy"
```

移除指定用户对于某个卷的所有权限。

body参数列表

参数	类型	描述	是否必需
user_id	string	待删除权限的用户ID	是
volume	string	待删除权限的卷名	是

转交卷

```
    curl -H "Content-Type:application/json" -X POST --data
    '{"volume":"vol","user_src":"user1","user_dst":"user2","force":"true"}'
1.  "http://10.196.59.198:17010/user/transferVol"
```

转交指定卷的所有权。此操作将指定卷从源用户名下移除，并添加至目标用户名下；同时，卷结构中的Owner字段的取值也将更新为目标用户的用户ID。

body参数列表

参数	类型	描述	是否必需
volume	string	待转交权限的卷名	是
user_src	string	该卷原来的所有者，必须与卷的Owner字段原取值相同	是
user_dst	string	转交权限后的目标用户ID	是
force	bool	是否强制转交卷。如果该值设为true，即使user_src的取值与卷的Owner取值不等，也会将卷变更至目标用户名下	否

元数据分片管理命令

获取当前设备上所有分片信息

```
curl -v  http://10.196.59.202:17210/getPartitions
```

获取指定分片ID的当前状态信息

```
curl -v http://10.196.59.202:17210/getPartitionById?pid=100
```

获取指定分片id的当前状态信息，包含当前分片组的raft leader地址，raft组成员，inode分配游标等信息

请求参数：

参数	类型	描述
pid	整型	元数据分片的ID

Inode管理命令

获取指定Inode基本信息

```
1. curl -v http://10.196.59.202:17210/getInode?pid=100&ino=1024
```

请求参数说明:

参数	类型	描述
pid	整型	分片id
ino	整型	inode的id

获取指定Inode的数据存储信息

```
1. curl -v http://10.196.59.202:17210/getExtentsByInode?pid=100&ino=1024
```

请求参数:

参数	类型	描述
pid	整型	分片id
ino	整型	inode id

获取指定元数据分片的全部inode信息

```
1. curl -v http://10.196.59.202:17210/getAllInodes?pid=100
```

请求参数:

参数	类型	描述
pid	整型	分片 id

Dentry调试命令

获取Dentry信息

```
curl -v 'http://10.196.59.202:17210/getDentry?
pid=100&name="aa.txt"&parentIno=1024'
```

Parameters

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

获取指定目录下全部文件

```
curl -v 'http://10.196.59.202:17210/getDirectory?pid=100&parentIno=1024'
```

Parameters

Parameter	Type	Description
pid	integer	partition id
ino	integer	inode id

获取指定分片的全部目录信息

```
curl -v 'http://10.196.59.202:17210/getAllDentry?pid=100'
```

Parameters

Parameter	Type	Description
pid	integer	partition id

CLI工具配置及使用方法

使用命令行界面工具（CLI）可以实现方便快捷的集群管理。利用此工具，可以查看集群及各节点的状态，并进行各节点、卷及用户的管理。

随着CLI的不断完善，最终将会实现对于集群各节点接口功能的100%覆盖。

编译及配置

下载ChubaoFS源码后，在 `chubaofs/cli` 目录下，运行 `build.sh` 文件 ，即可生成 `cfs-cli` 可执行程序。

同时，在 `root` 目录下会生成名为 `.cfs-cli.json` 的配置文件，修改master地址为当前集群的master地址即可。也可使用命令 `./cfs-cli config info` 和 `./cfs-cli config set` 来查看和设置配置文件。

使用方法

在 `chubaofs/cli` 目录下，执行命令 `./cfs-cli --help` 或 `./cfs-cli -h` ，可获取CLI的帮助文档。

CLI主要分为六类管理命令:

命令	描述
cfs-cli cluster	集群管理
cfs-cli metanode	元数据节点管理
cfs-cli datanode	数据节点管理
cfs-cli datapartition	数据分片管理
cfs-cli metapartition	元数据分片管理
cfs-cli config	配置管理
cfs-cli completion	生成自动补全命令脚本
cfs-cli volume, vol	卷管理
cfs-cli user	用户管理
cfs-cli compatibility	兼容性测试

集群管理命令

```
    ./cfs-cli cluster info          #获取集群信息，包括集群名称、地址、卷数量、节点数量及使
1.  用率等
```

```
./cfs-cli cluster stat          #按区域获取元数据和数据节点的使用量、状态等
```

```
./cfs-cli cluster freeze [true/false]        #是否冻结集群，设置为 `true` 冻结后，当
partition写满，集群不会自动分配新的partition
```

```
./cfs-cli cluster threshold [float]     #设置集群中每个MetaNode的内存阈值
```

元数据节点管理命令

```
./cfs-cli metanode list          #获取所有元数据节点的信息，包括id、地址、读写状态及存活
状态
```

```
./cfs-cli metanode info [Address]     #展示元数据节点基本信息，包括状态、使用量、承载的
partition ID等，
```

```
./cfs-cli metanode decommission [Address] #将该元数据节点下线，该节点上的partition将
自动转移至其他可用节点
```

数据节点管理命令

```
./cfs-cli datanode list          #获取所有数据节点的信息，包括id、地址、读写状态及存活状
态
```

```
./cfs-cli datanode info [Address]     #展示数据节点基本信息，包括状态、使用量、承载的
partition ID等，
```

```
./cfs-cli datanode decommission [Address] #将该数据节点下线，该节点上的data
partition将自动转移至其他可用节点
```

数据分片管理命令

```
./cfs-cli datapartition info [VOLUME] [Partition ID]        #获取指定data
partition的信息
```

```
./cli datapartition decommission [Address] [Partition ID]   #将目标节点上的指定
data partition分片下线，并自动转移至其他可用节点
```

```
./cfs-cli datapartition add-replica [Address] [Partition ID]    #在目标节点新增一个data partition分片
```

```
./cfs-cli datapartition del-replica [Address] [Partition ID]    #删除目标节点上的data partition分片
```

```
./cfs-cli datapartition check    #故障诊断，查找多半分片不可用和分片缺失的data partition
```

元数据分片管理命令

```
./cfs-cli metapartition info [VOLUME] [Partition ID]        #获取指定meta partition的信息
```

```
./cli metapartition decommission [Address] [Partition ID]   #将目标节点上的指定meta partition分片下线，并自动转移至其他可用节点
```

```
./cfs-cli metapartition add-replica [Address] [Partition ID]    #在目标节点新增一个meta partition分片
```

```
./cfs-cli metapartition del-replica [Address] [Partition ID]    #删除目标节点上的meta partition分片
```

```
./cfs-cli metapartition check    #故障诊断，查找多半分片不可用和分片缺失的meta partition
```

配置管理

```
./cfs-cli config info     #展示配置信息
```

```
./cfs-cli config set     #设置配置信息
root@fa27e115a0ba:/cfs#$ cfs-cli config set
Please input master host:
test.chubaofs.com
Config has been set successfully!
```

自动补全管理

```
./cfs-cli completion      #生成命令自动补全脚本
```

卷管理命令

```
./cfs-cli volume create [VOLUME NAME] [USER ID] [flags]      #创建所有者是[USER ID]的卷[VOLUME NAME]
Flags:
    --capacity uint                                          #指定卷的容量，单位GB（默认为10）
    --dp-size  uint                                          #指定数据分片的大小，单位GB（默认为120）
    --follower-read                                          #启用从follower副本中读取数据的功能（默认为true）
    --mp-count int                                           #指定初始元数据分片的数量（默认为3）
    -y, --yes                                                #跳过所有问题并设置回答为"yes"
```

```
./cfs-cli volume delete [VOLUME NAME] [flags]                #删除指定卷[VOLUME NAME]
Flags:
    -y, --yes                                                #跳过所有问题并设置回答为"yes"
```

```
./cfs-cli volume info [VOLUME NAME] [flags]                  #获取卷[VOLUME NAME]的信息
Flags:
    -d, --data-partition                                     #显示数据分片的详细信息
    -m, --meta-partition                                     #显示元数据分片的详细信息
```

```
./cfs-cli volume add-dp [VOLUME] [NUMBER]                    #创建并添加个数为[NUMBER]的数据分片至卷[VOLUME]
```

```
./cfs-cli volume list                                        #获取包含当前所有卷信息的列表
```

```
./cfs-cli volume transfer [VOLUME NAME] [USER ID] [flags]    #将卷[VOLUME NAME]转交给其他用户[USER ID]
Flags：
    -f, --force                                              #强制转交
```

```
    -y, --yes                                    #跳过所有问题并设置回答
4. 为"yes"
```

用户管理命令

```
1. ./cfs-cli user create [USER ID] [flags]          #创建用户[USER ID]
2. Flags:
3.     --access-key string                  #指定用户用于对象存储功能的access key
4.     --secret-key string                  #指定用户用于对象存储功能的secret key
5.     --password string                    #指定用户密码
       --user-type string                   #指定用户类型，可选项为normal或
6. admin（默认为normal）
7.     -y, --yes                            #跳过所有问题并设置回答为"yes"
```

```
1. ./cfs-cli user delete [USER ID] [flags]          #删除用户[USER ID]
2. Flags:
3.     -y, --yes                            #跳过所有问题并设置回答为"yes"
```

```
1. ./cfs-cli user info [USER ID]                    #获取用户[USER ID]的信息
```

```
1. ./cfs-cli user list                              #获取包含当前所有用户信息的列表
```

```
    ./cfs-cli user perm [USER ID] [VOLUME] [PERM]   #更新用户[USER ID]对于卷[VOLUME]
1. 的权限[PERM]
                                            #[PERM]可选项为"只
2. 读"（READONLY/RO）、"读写"（READWRITE/RW）、"删除授权"（NONE）
```

```
1. ./cfs-cli user update [USER ID] [flags]          #更新用户[USER ID]的信息
2. Flags:
3.     --access-key string                  #更新后的access key取值
4.     --secret-key string                  #更新后的secret key取值
       --user-type string                   #更新后的用户类型，可选项为normal或
5. admin
6.     -y, --yes                            #跳过所有问题并设置回答为"yes"
```

兼容性测试

```
    ./cfs-cli cptest meta [Snapshot Path] [Host] [Partition ID]         #meta data
1. 兼容性测试
```

```
Parameters:
        [Snapshot Path] string                  #快照文件存放路径
        [Host] string                           #生成快照文件的MetaNode地址
        [Partition ID] string                   #需要测试对比的meta partition
ID
```

例:

1. 使用旧版本server生成meta data, 停止meta data服务, 五分钟后会生成meta data快照数据, 然后拷贝快照文件到本地目录
2. 在本地机器执行 cfs-cli cptest meta 命令对刚刚拷贝的旧版本快照数据和线上的新数据对比验证

```
[Verify result]
All dentry are consistent
All inodes are consistent
All meta has checked
```

使用案例

ChubaoFS是一个分布式文件系统，兼容绝大部分POSIX文件系统语义,挂载后可以像使用本地文件系统一样简单。 基本上可以用在任何需要文件系统的场合，替换本地文件系统，实现可无限扩展的、无物理边际的存储. 已经应用在多种场景，下面是摘取的部分场景

机器学习

使用本地磁盘存储训练数据集的缺点

- 本地磁盘空间小，有多个模型，每个模型的训练数据集到达TB级别，使用本地磁盘存储训练数据集，需要缩减训练数据集大小
- 训练数据集需要经常更新，需要更多的磁盘空间
- 如果机器故障，存在训练数据集丢失风险

使用chubaofs存储点击流日志优势

- 磁盘空间不受限制，易扩容，根据磁盘使用百分比自动扩展磁盘容量，能够实现存储系统按需扩容，极大的节省存储成本
- 数据有多个副本，保证数据高可靠，不用担心丢失数据
- 兼容posix文件系统接口，应用程序无需任何改动

ElasticSearch

使用本地磁盘存储数据会经常遇到下面的问题:

- 磁盘使用率不均匀，磁盘IO无法充分利用
- 本地磁盘空间大小受限制

使用chubaofs作为后端存储的优势

- 磁盘空间不受限制，易扩容，根据磁盘使用百分比自动扩展磁盘容量，能够实现存储系统按需扩容，极大的节省存储成本
- 磁盘IO使用率均匀，磁盘IO得到充分利用
- 保证数据高可靠，不用担心丢失数据

Nginx日志存储

你是不是经常为本地磁盘空间已满问题而发愁，使用chubaofs，可以将数据存储在分布式文件系统中，不用再担心磁盘空间不够用问题。

使用本地磁盘存储日志的问题

- docker本地磁盘空间小
- docker容器故障，日志丢失且不可恢复
- 物理机和docker机器混合部署，难以管理，运维成本高

使用chubaofs存储Nginx日志优势

- 磁盘空间不受限制，易扩容，根据磁盘使用百分比自动扩展磁盘容量，能够实现存储系统按需扩容，极大的节省存储成本
- 保证数据高可靠，不用担心丢失数据
- 多副本，可解决磁盘级和datanode节点故障导致日志无法写入问题
- 兼容posix文件系统接口，应用程序无需任何改动
- chubaofs运维简单，一个人就可以轻松的管理上万台机器的集群

Spark

在大数据集场景下，你是否为存储Spark中间计算结果需要精心计算每个task的数据量而发愁，可以将shuffle结果存储到cfs，不用再担心磁盘没有可用空间而导致任务失败问题，实现存储和计算分离。

使用本地磁盘存储shuffle中间结果的痛点

- 磁盘空间不足
- 临时目录文件过多，无法创建新文件

使用chubaofs优势

- 磁盘空间不受限制，易扩容，根据磁盘使用百分比自动扩展磁盘容量，能够实现存储系统按需扩容，极大的节省存储成本
- Metanode管理文件元数据，可以水平扩容，文件个数不受限制

MySQL数据库备份

使用云存储备份MySQL数据库的缺点

- 需要利用云存储SDK或者RESTful API开发备份程序，增加运维难度。
- 备份文件失败，排查问题比较困难
- 备份文件到云存储后，不方便查看文件是否上传成功
- 备份文件经过多层服务处理，影响性能

使用chubaofs备份MySQL数据库的优点

- 简单易用，兼容POSIX文件接口，可以当作本地文件系统使用

- 完整且详尽的操作记录存储在本地文件系统中，排查问题简单方便
- 只需执行ls命令，即可验证文件是否上传成功
- 支持PageCache和WriteCache，与云存储相比，文件读写性能显著提升

性能评估

环境准备

集群信息

节点类型	节点数	CPU	内存	存储	网络	备注
管理节点	3	32	32 GB	120 GB SSD	10 Gb/s	
元数据节点	10	32	32 GB	16 x 1TB SSD	10 Gb/s	混合部署
数据节点	10	32	32 GB	16 x 1TB SSD	10 Gb/s	混合部署

卷设置

参数	默认值	推荐值	说明
是否开启FollowerRead	True	True	
容量	10 GB	300 000 000 GB	
数据副本	3	3	
元数据副本数	3	3	
数据分区大小	120 GB	120 GB	只是理论值上限 并不预分配空间
数据分区数	10	1500	
元数据分区数	3	10	
是否跨zone	False	False	

设置方法：

```
$ cfs-cli volume create test-vol {owner} --capacity=300000000 --mp-count=10
Create a new volume:
  Name                : test-vol
  Owner               : ltptest
  Dara partition size : 120 GB
  Meta partition count: 10
  Capacity            : 300000000 GB
  Replicas            : 3
  Allow follower read : Enabled

Confirm (yes/no)[yes]: yes
Create volume success.

$ cfs-cli volume add-dp test-vol 1490
```

client配置

参数	默认值	推荐值
rate limit	-1	-1

```
#查看当前iops：
$ http://[ClientIP]:[ProfPort]/rate/get
#设置iops，默认值-1代表不限制iops
$ http://[ClientIP]:[ProfPort]/rate/set?write=800&read=800
```

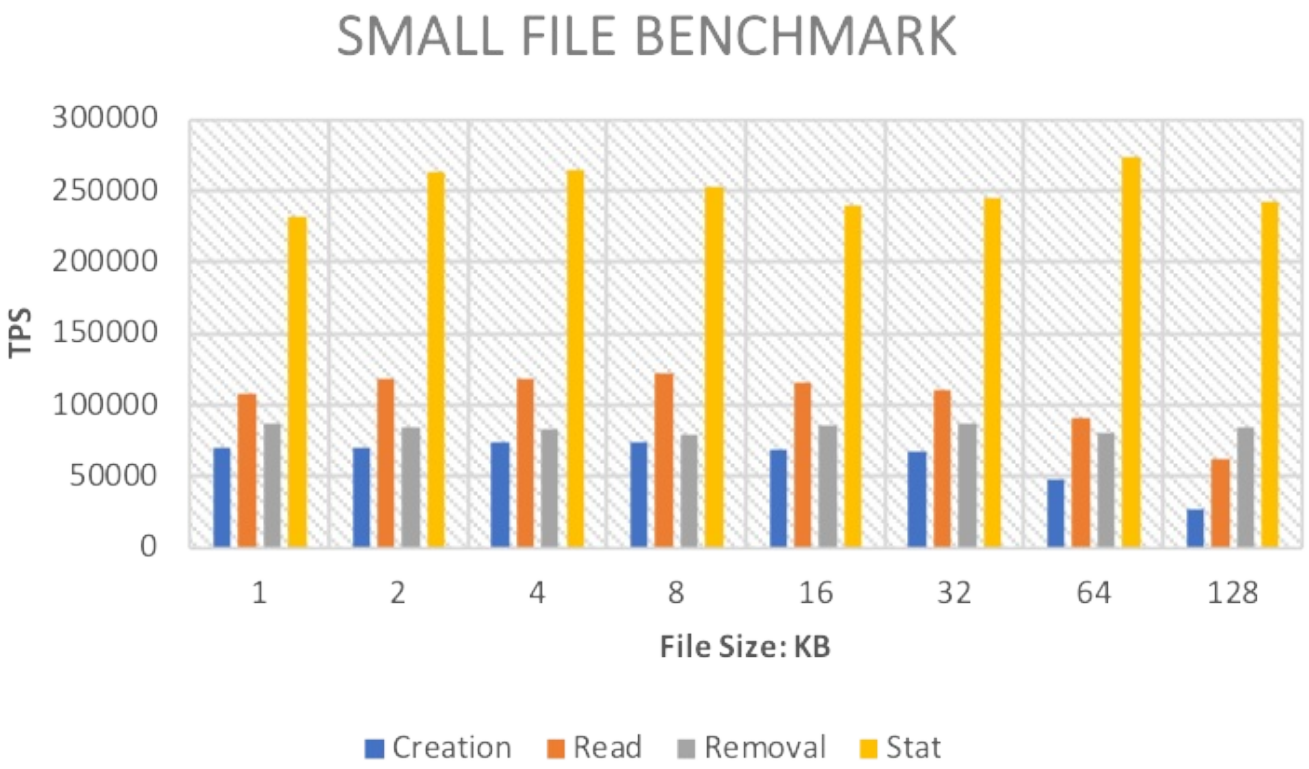
小文件性能评估

通过 mdtest 进行小文件性能测试的结果如下：

配置

```
#!/bin/bash
set -e
TARGET_PATH="/mnt/test/mdtest" # mount point of ChubaoFS volume
for FILE_SIZE in 1024 2048 4096 8192 16384 32768 65536 131072 # file size
do
mpirun --allow-run-as-root -np 512 --hostfile hfile64 mdtest -n 1000 -w $i -e
$FILE_SIZE -y -u -i 3 -N 1 -F -R -d $TARGET_PATH;
done
```

测试结果



文件大小(KB)	1	2	4	8	16	32	64	128
创建操作(TPS)	70383	70383	73738	74617	69479	67435	47540	27147
读取操作(TPS)	108600	118193	118346	122975	116374	110795	90462	62082
删除操作(TPS)	87648	84651	83532	79279	85498	86523	80946	84441
信息查看(TPS)	231961	263270	264207	252309	240244	244906	273576	242930

IO性能评估

通过 `fio` 进行IO性能测试的结果如下： （注：其中多个客户端挂载同一个卷，进程指 *fio* 进程）

1. 顺序读

工具设置

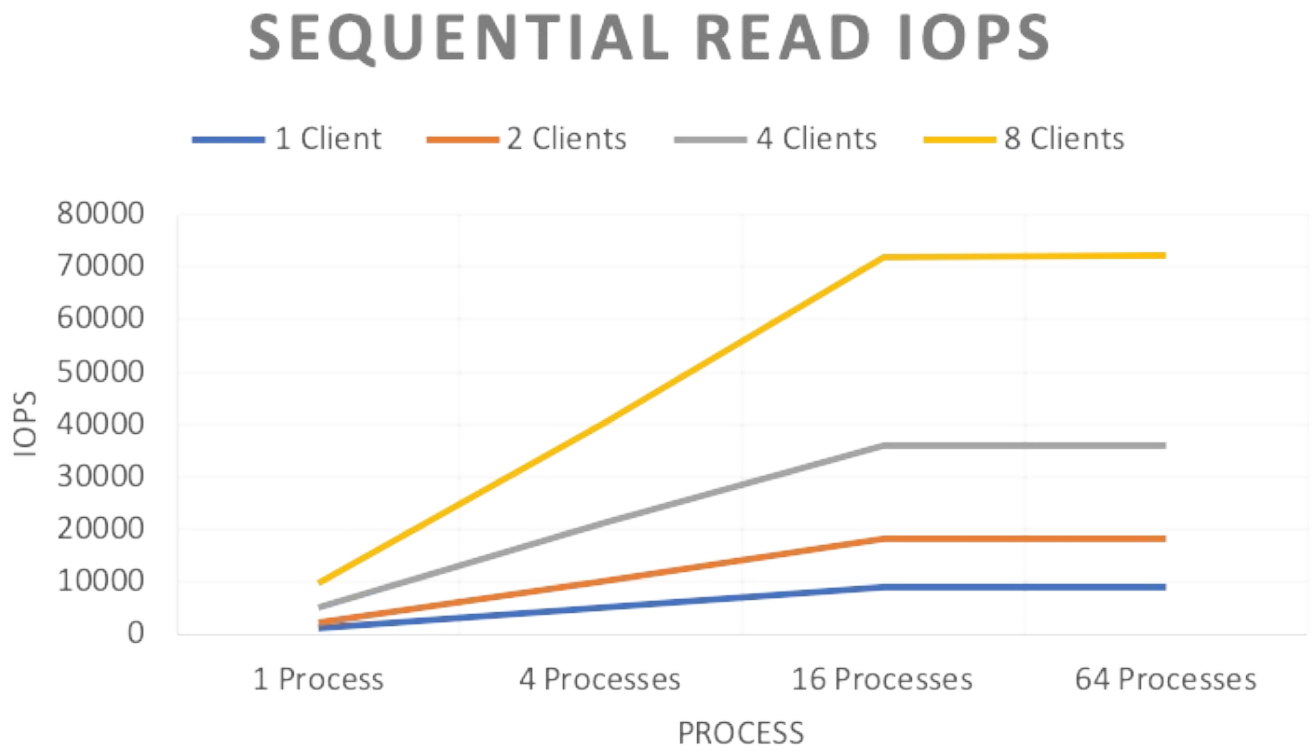
```
#!/bin/bash
fio -directory={} \
    -ioengine=psync \
    -rw=read \  # sequential read
    -bs=128k \  # block size
```

```
    -direct=1 \ # enable direct IO
    -group_reporting=1 \
    -fallocate=none \
    -time_based=1 \
    -runtime=120 \
    -name=test_file_c{} \
    -numjobs={} \
    -nrfiles=1 \
    -size=10G
```

带宽(MB/s)

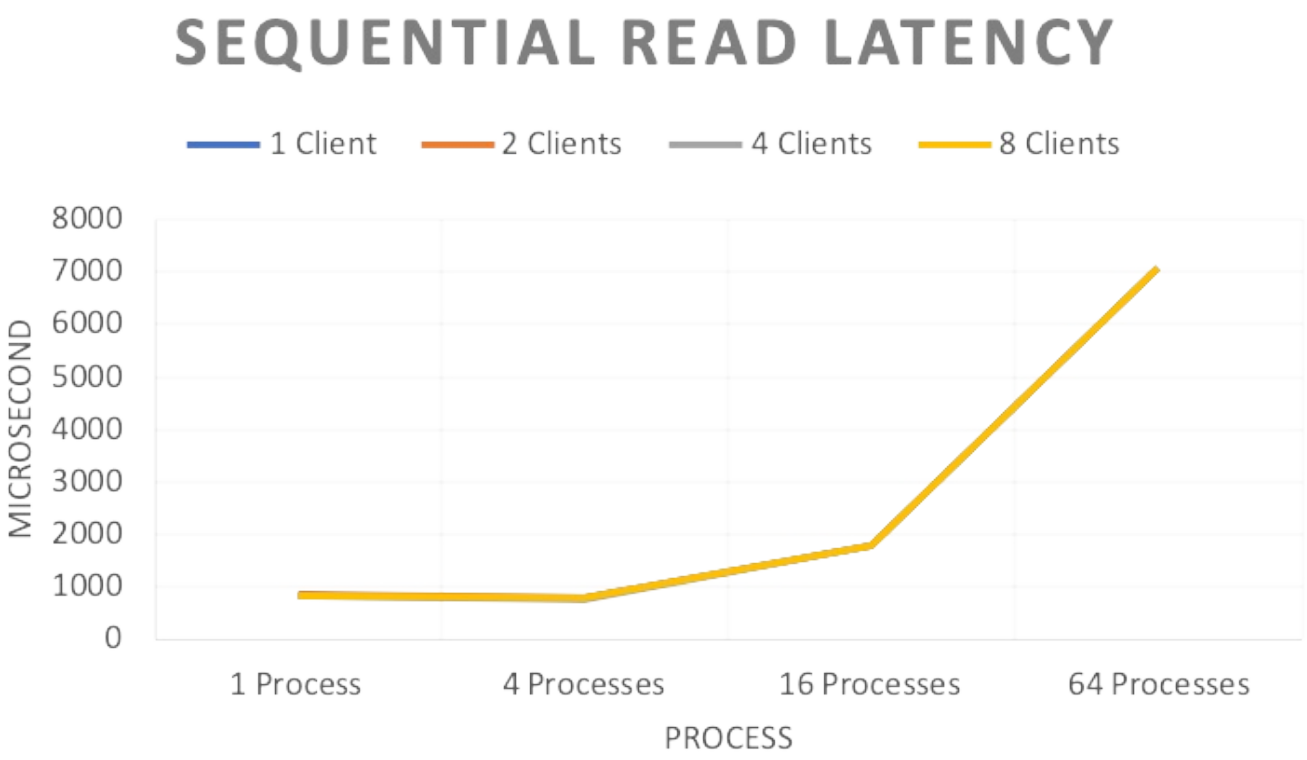
	1 进程	4 进程	16 进程	64 进程
1 客户端	148.000	626.000	1129.000	1130.000
2 客户端	284.000	1241.000	2258.000	2260.000
4 客户端	619.000	2640.000	4517.000	4515.000
8 客户端	1193.000	4994.000	9006.000	9034.000

IOPS



	1 进程	4 进程	16 进程	64 进程
1 客户端	1180.000	5007.000	9031.000	9040.000
2 客户端	2275.000	9924.000	18062.000	18081.000
4 客户端	4954.000	21117.000	36129.000	36112.000
8 客户端	9531.000	39954.000	72048.000	72264.000

延迟(微秒)



	1 进程	4 进程	16 进程	64 进程
1 客户端	842.200	794.340	1767.310	7074.550
2 客户端	874.255	801.690	1767.370	7071.715
4 客户端	812.363	760.702	1767.710	7077.065
8 客户端	837.707	799.851	1772.620	7076.967

2. 顺序写

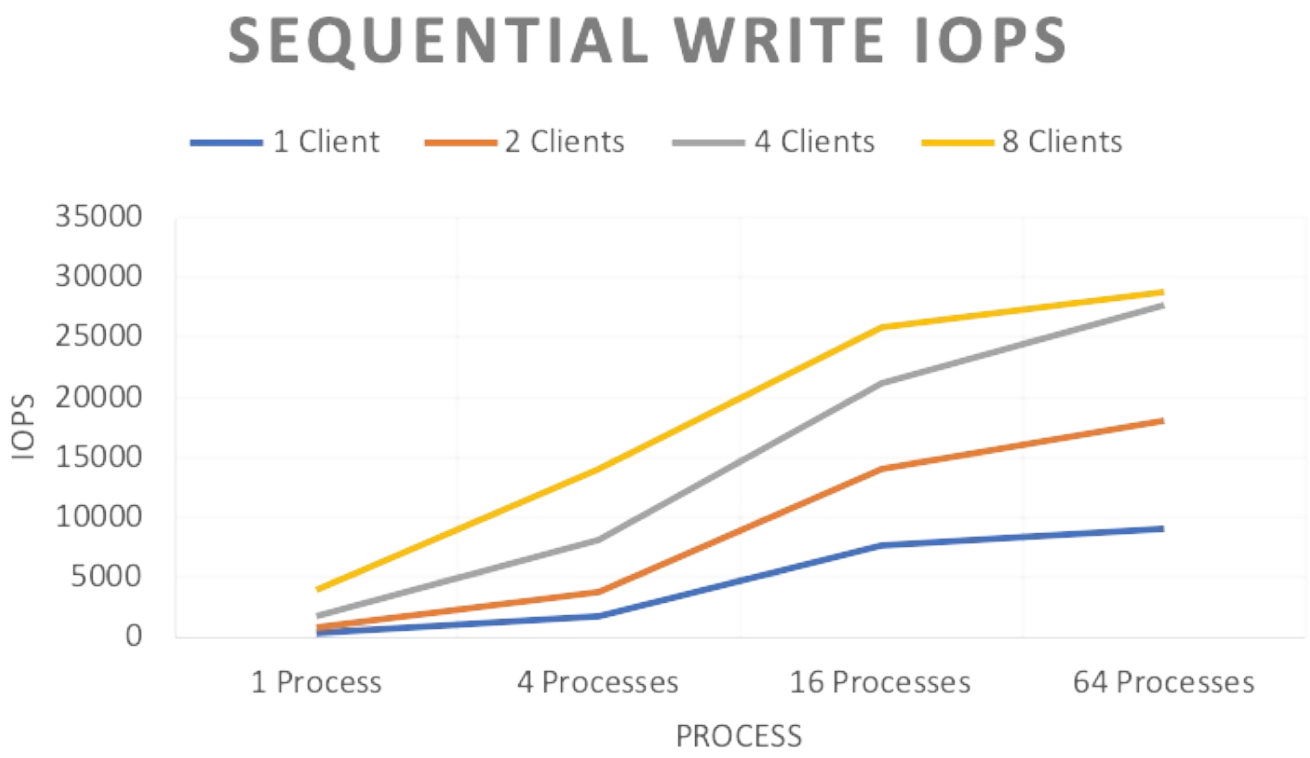
工具设置

```
#!/bin/bash
fio -directory={} \
    -ioengine=psync \
    -rw=write \ # sequential write
    -bs=128k \  # block size
    -direct=1 \ # enable direct IO
    -group_reporting=1 \
    -fallocate=none \
    -name=test_file_c{} \
    -numjobs={} \
    -nrfiles=1 \
    -size=10G
```

带宽(MB/s)

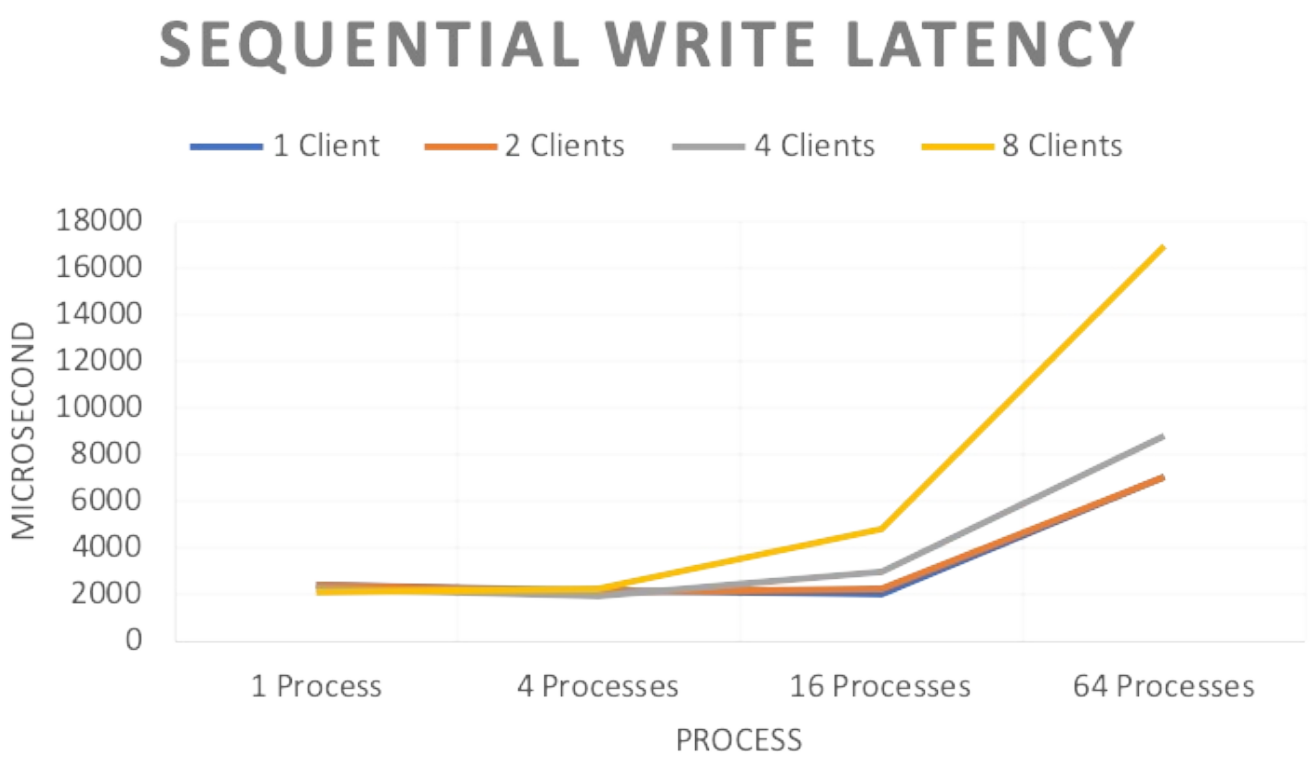
	1 进程	4 进程	16 进程	64 进程
1 客户端	52.200	226.000	956.000	1126.000
2 客户端	104.500	473.000	1763.000	2252.000
4 客户端	225.300	1015.000	2652.000	3472.000
8 客户端	480.600	1753.000	3235.000	3608.000

IOPS



	1 进程	4 进程	16 进程	64 进程
1 客户端	417	1805	7651	9004
2 客户端	835	3779	14103	18014
4 客户端	1801	8127	21216	27777
8 客户端	3841	14016	25890	28860

延迟(微秒)



	1 进程	4 进程	16 进程	64 进程
1 客户端	2385.400	2190.210	2052.360	7081.320
2 客户端	2383.610	2081.850	2233.790	7079.450
4 客户端	2216.305	1947.688	2946.017	8842.903
8 客户端	2073.921	2256.120	4787.496	17002.425

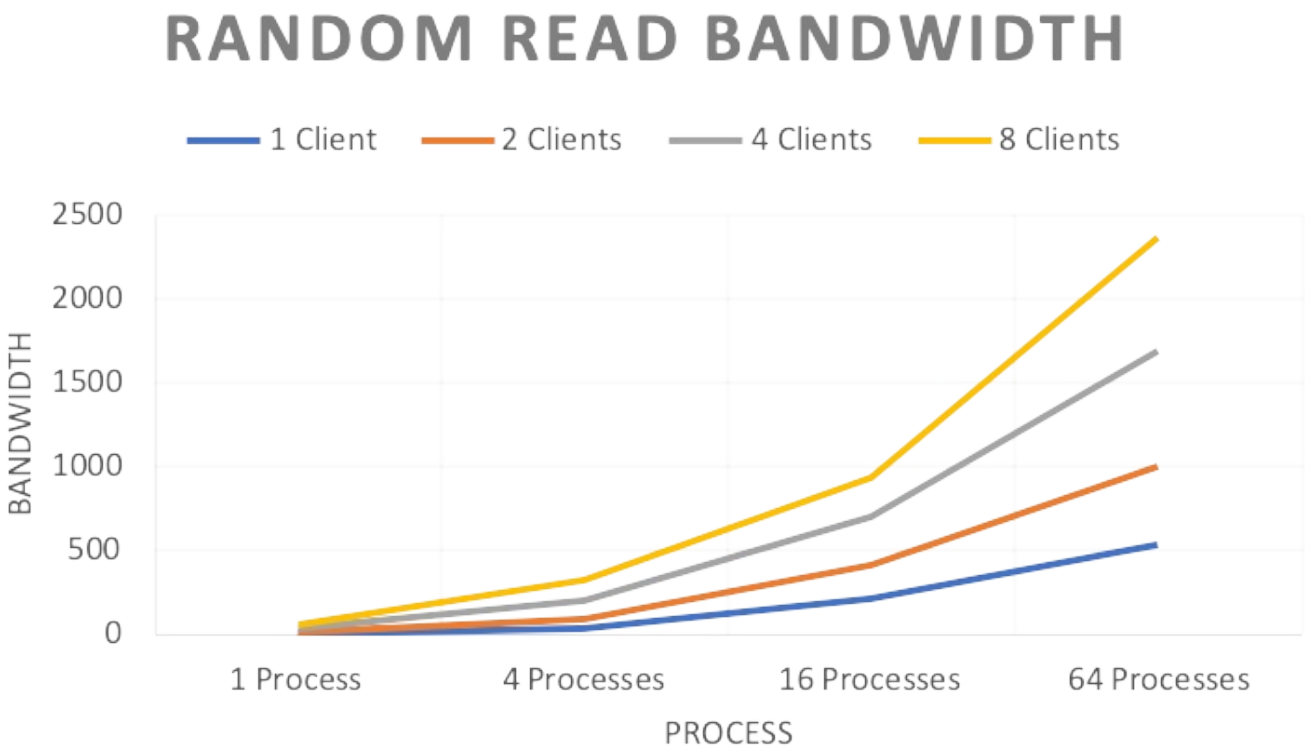
3. 随机读

工具设置

```
#!/bin/bash
fio -directory={} \
    -ioengine=psync \
    -rw=randread \ # random read
    -bs=4k \       # block size
    -direct=1 \    # enable direct IO
    -group_reporting=1 \
    -fallocate=none \
    -time_based=1 \
    -runtime=120 \
    -name=test_file_c{} \
    -numjobs={} \
    -nrfiles=1 \
```

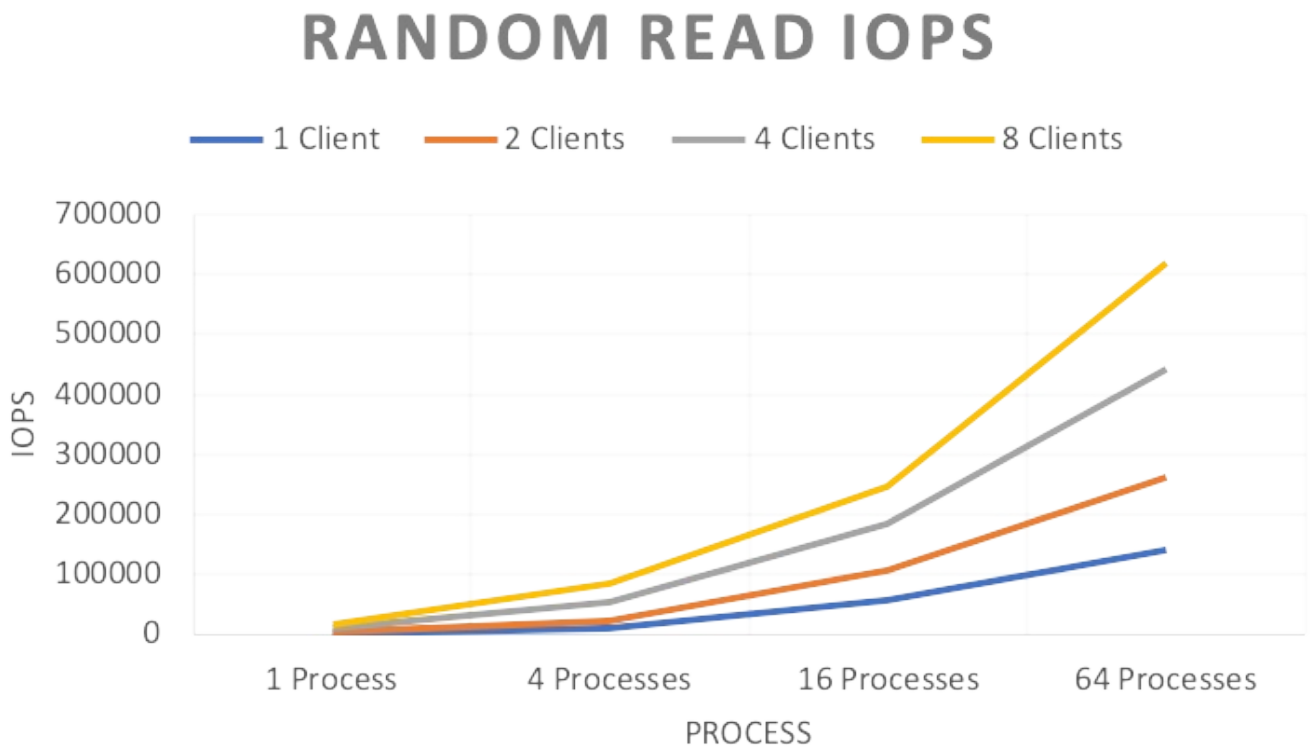
```
14.      -size=10G
```

带宽(MB/s)



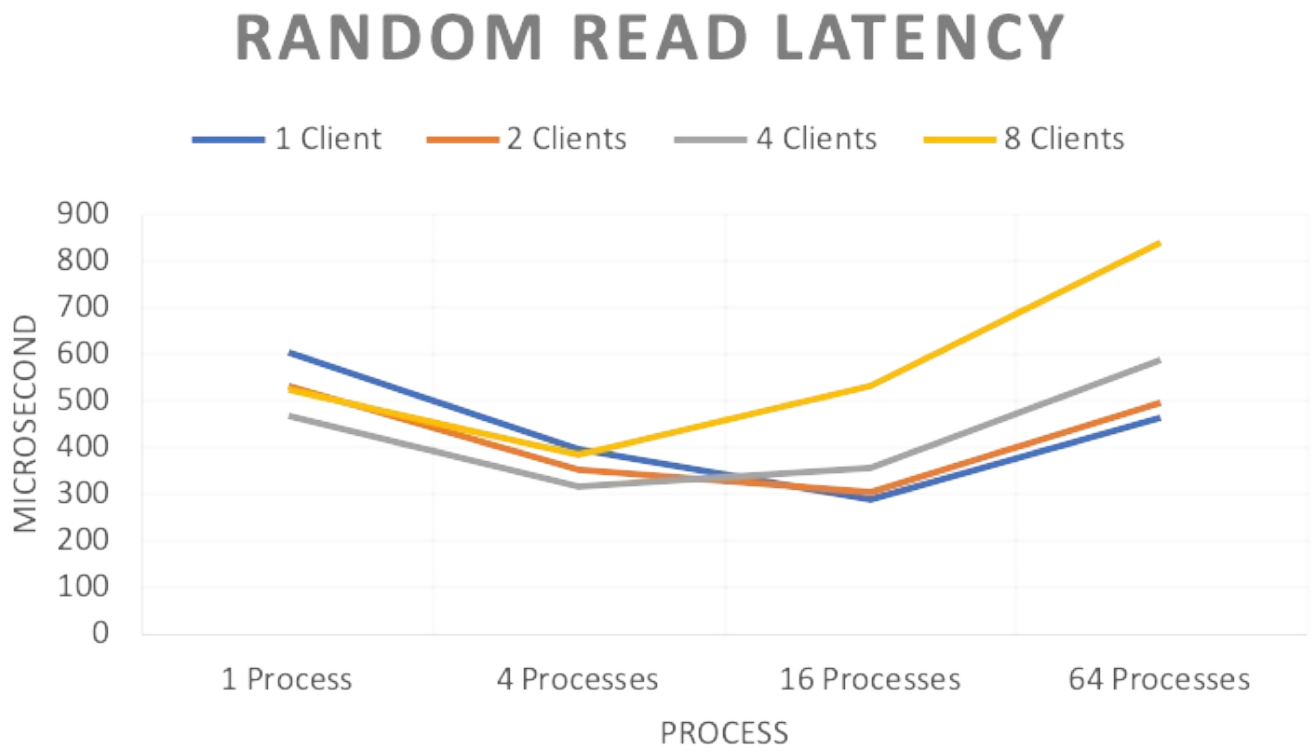
	1 进程	4 进程	16 进程	64 进程
1 客户端	6.412	39.100	216.000	534.000
2 客户端	14.525	88.100	409.000	1002.000
4 客户端	33.242	200.200	705.000	1693.000
8 客户端	59.480	328.300	940.000	2369.000

IOPS



	1 进程	4 进程	16 进程	64 进程
1 客户端	1641	10240	56524.800	140288
2 客户端	3718	23142.4	107212.8	263168
4 客户端	8508	52428.8	184627.2	443392
8 客户端	15222	85072.8	246681.6	621056

延迟(微秒)



	1 进程	4 进程	16 进程	64 进程
1 客户端	603.580	395.420	287.510	466.320
2 客户端	532.840	351.815	303.460	497.100
4 客户端	469.025	317.140	355.105	588.847
8 客户端	524.709	382.862	530.811	841.985

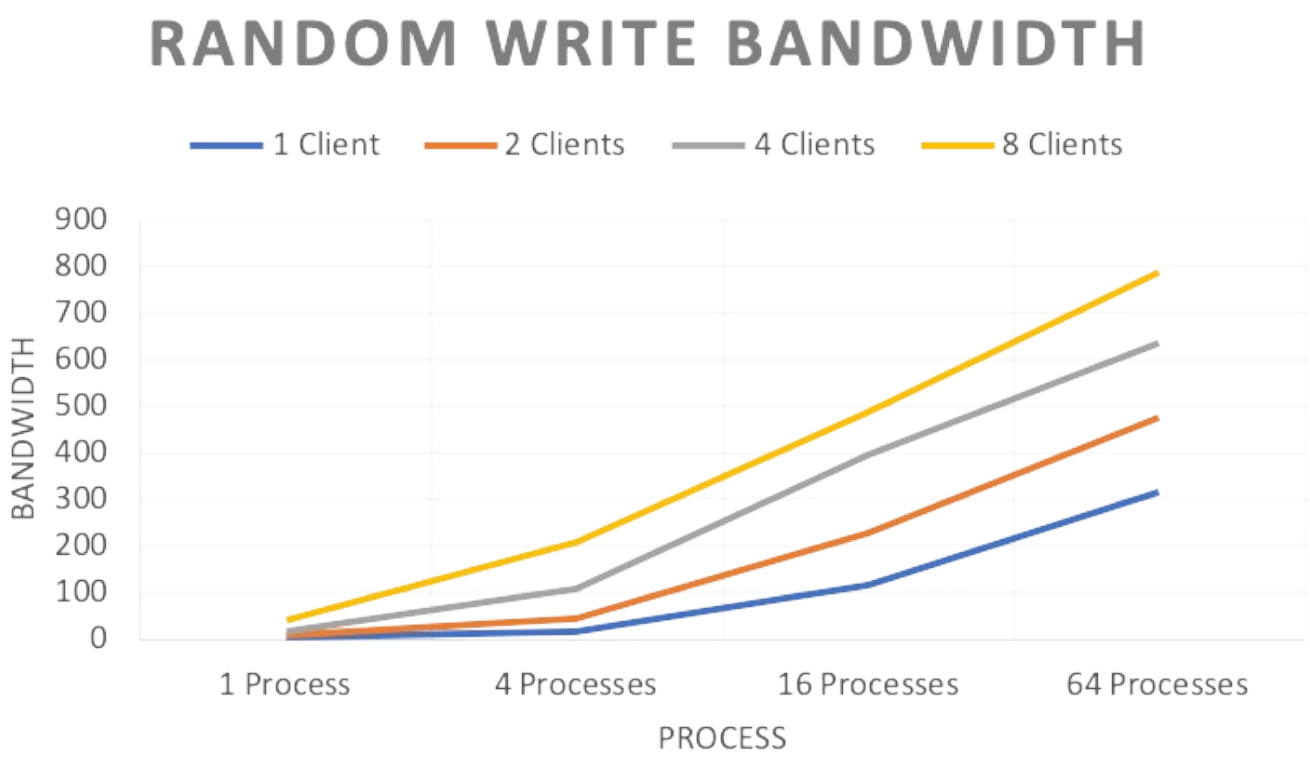
4. 随机写

工具设置

```
#!/bin/bash
fio -directory={} \
    -ioengine=psync \
    -rw=randwrite \ # random write
    -bs=4k \        # block size
    -direct=1 \     # enable direct IO
    -group_reporting=1 \
    -fallocate=none \
    -time_based=1 \
    -runtime=120 \
    -name=test_file_c{} \
    -numjobs={} \
    -nrfiles=1 \
```

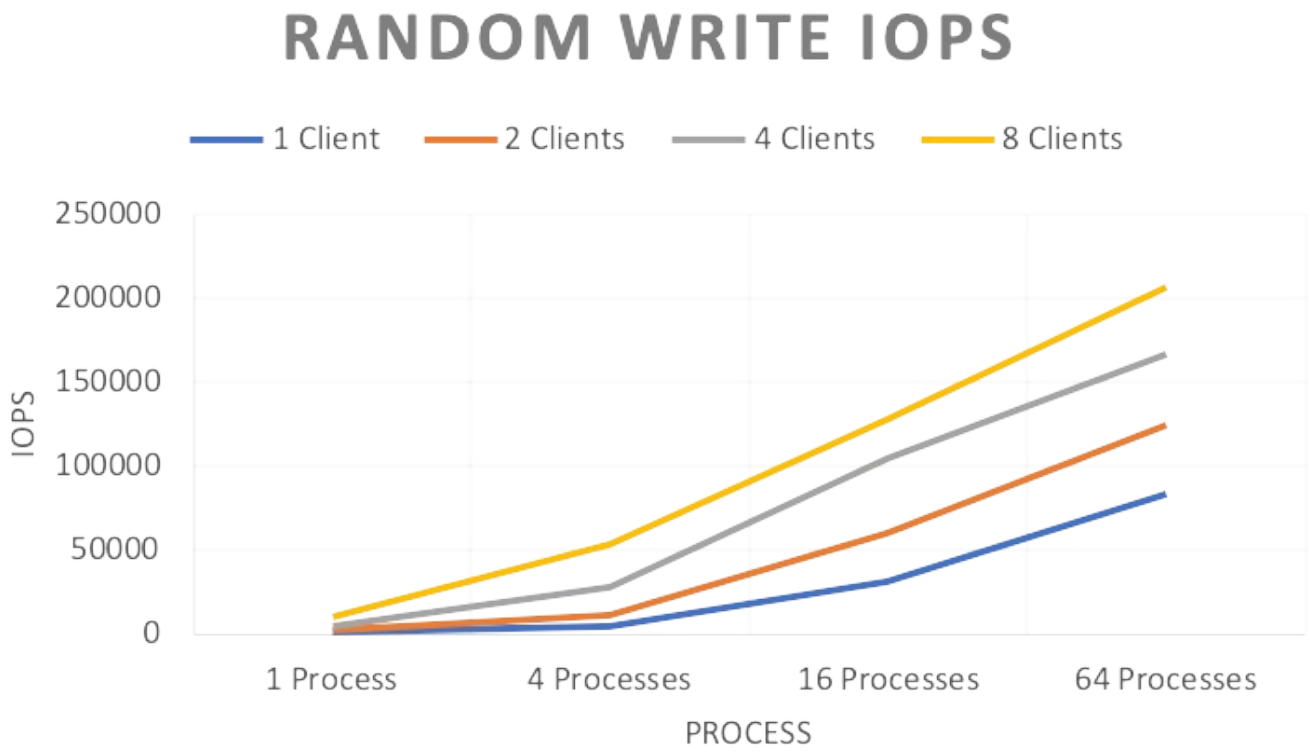
```
14.      -size=10G
```

带宽(MB/s)



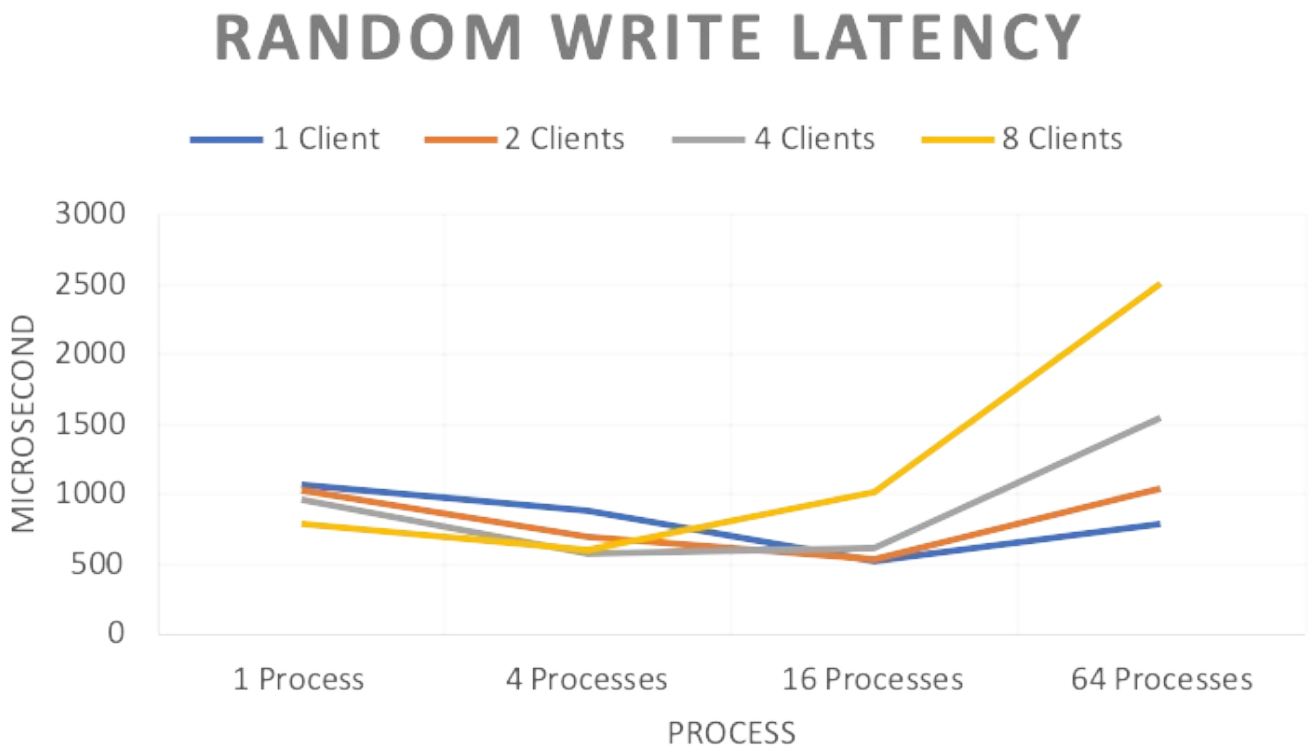
	1 进程	4 进程	16 进程	64 进程
1 客户端	3.620	17.500	118.000	318.000
2 客户端	7.540	44.800	230.000	476.000
4 客户端	16.245	107.700	397.900	636.000
8 客户端	39.274	208.100	487.100	787.100

IOPS



	1 进程	4 进程	16 进程	64 进程
1 客户端	926.000	4476.000	31027.200	83251.200
2 客户端	1929.000	11473.000	60313.600	124620.800
4 客户端	4156.000	27800.000	104243.200	167014.400
8 客户端	10050.000	53250.000	127692.800	206745.600

延迟(微秒)



	1 进程	4 进程	16 进程	64 进程
1 客户端	1073.150	887.570	523.820	784.030
2 客户端	1030.010	691.530	539.525	1042.685
4 客户端	955.972	575.183	618.445	1552.205
8 客户端	789.883	598.393	1016.185	2506.424

元数据性能评估

通过 `mdtest` 进行元数据性能测试的结果如下：

工具设置

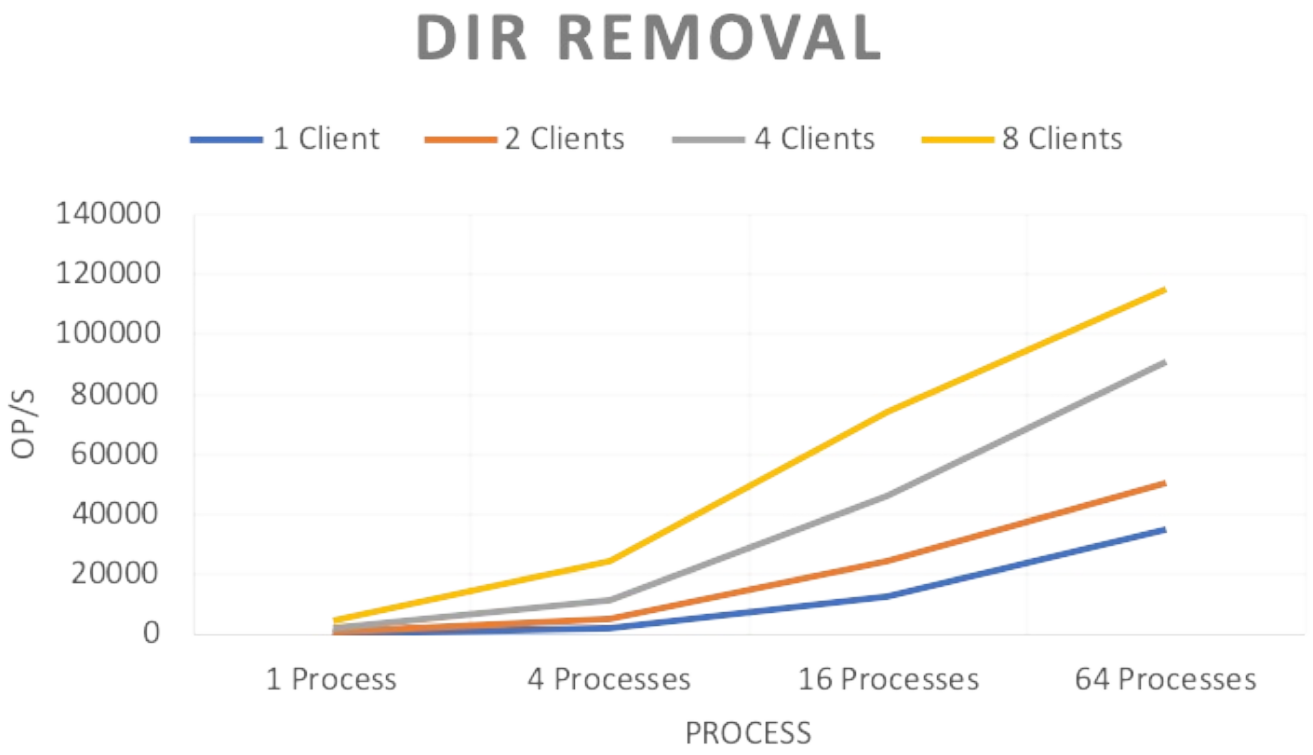
```
#!/bin/bash
TEST_PATH=/mnt/cfs/mdtest # mount point of ChubaoFS volume
for CLIENTS in 1 2 4 8 # number of clients
do
mpirun --allow-run-as-root -np $CLIENTS --hostfile hfile01 mdtest -n 5000 -u -z
2 -i 3 -d $TEST_PATH;
done
```

目录创建

目录创建评估结果

	1 进程	4 进程	16 进程	64 进程
1 客户端	448.618	2421.001	14597.97	43055.15
2 客户端	956.947	5917.576	28930.431	72388.765
4 客户端	2027.02	13213.403	54449.056	104771.356
8 客户端	4643.755	27416.904	89641.301	119542.62

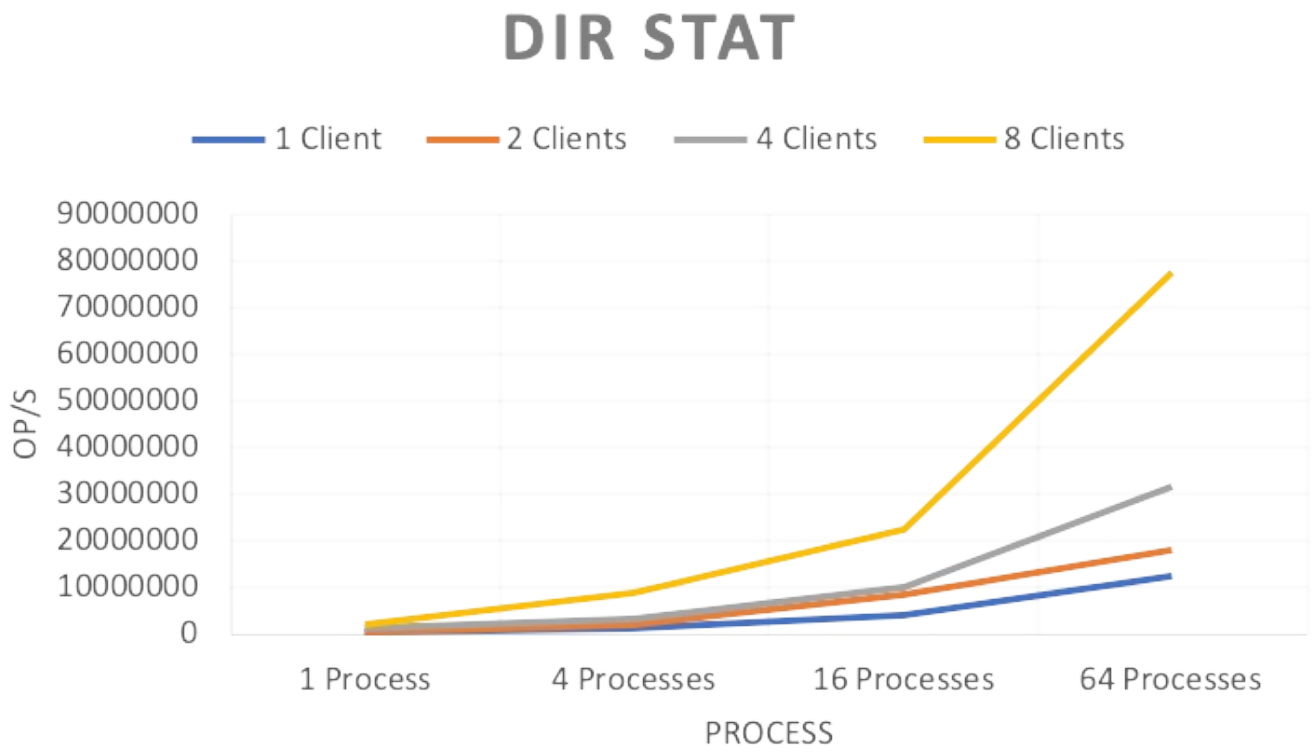
目录删除



目录删除评估结果

	1 进程	4 进程	16 进程	64 进程
1 客户端	399.779	2118.005	12351.635	34903.672
2 客户端	833.353	5176.812	24471.674	50242.973
4 客户端	1853.617	11462.927	46413.313	91128.059
8 客户端	4441.435	24133.617	74401.336	115013.557

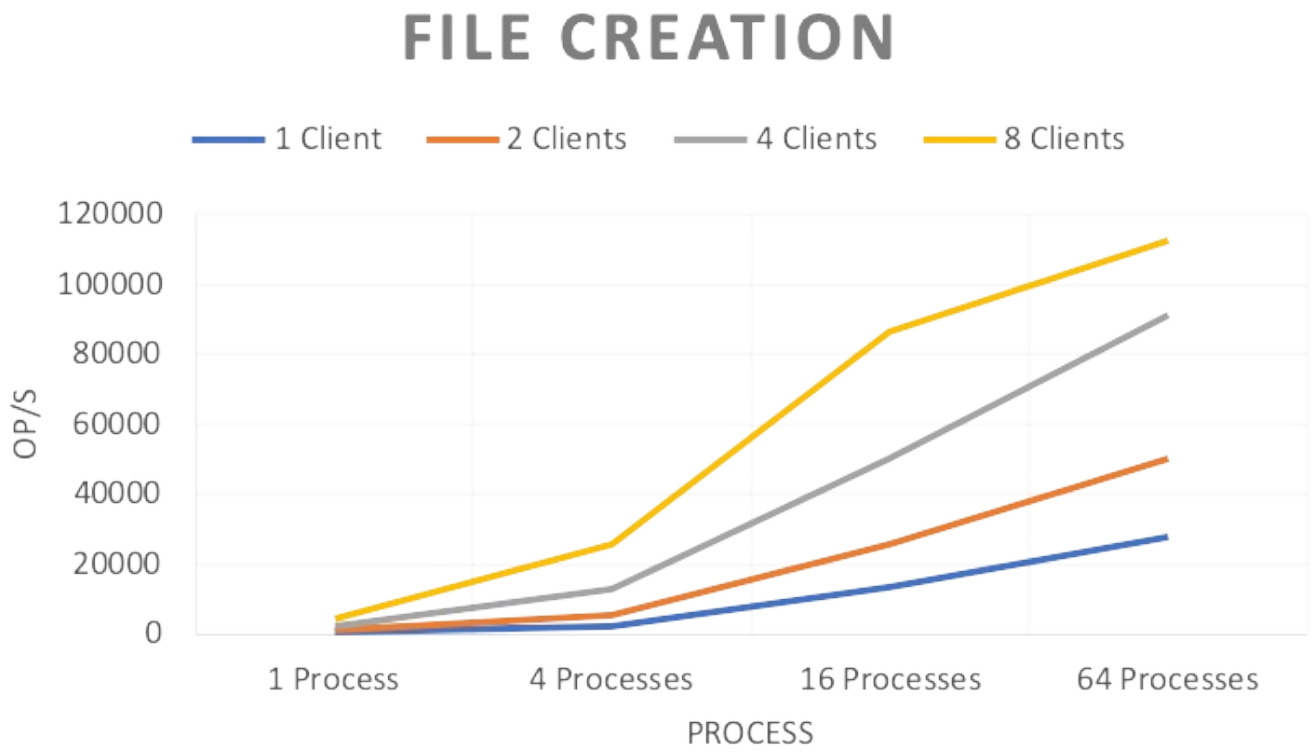
目录状态查看



目录状态查看评估结果

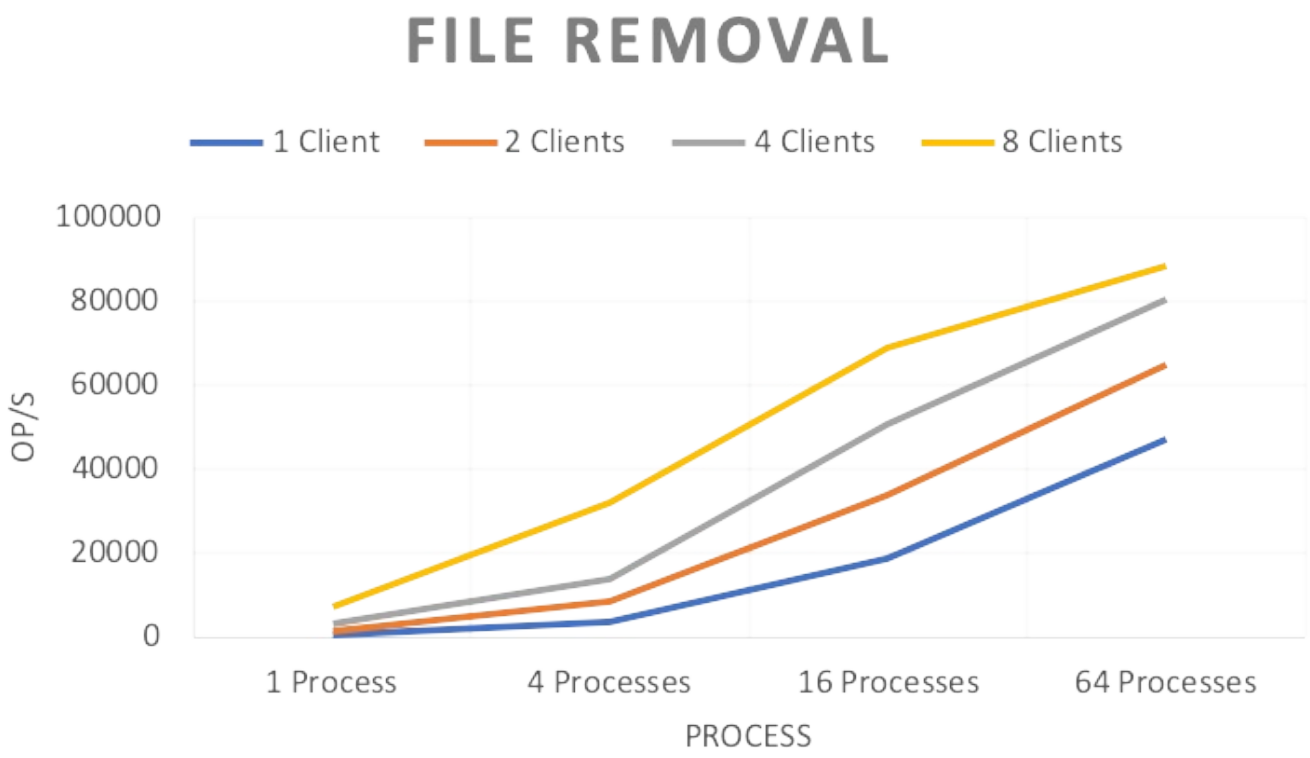
	1 进程	4 进程	16 进程	64 进程
1 客户端	283232.761	1215309.524	4231088.104	12579177.02
2 客户端	572834.143	2169669.058	8362749.217	18120970.71
4 客户端	1263474.549	3333746.786	10160929.29	31874265.88
8 客户端	2258670.069	8715752.83	22524794.98	77533648.04

文件创建



	1 进程	4 进程	16 进程	64 进程
1 客户端	448.888	2400.803	13638.072	27785.947
2 客户端	925.68	5664.166	25889.163	50434.484
4 客户端	2001.137	12986.968	50330.952	91387.825
8 客户端	4479.831	25933.437	86667.966	112746.199

文件删除



	1 进程	4 进程	16 进程	64 进程
1 客户端	605.143	3678.138	18631.342	47035.912
2 客户端	1301.151	8365.667	34005.25	64860.041
4 客户端	3032.683	14017.426	50938.926	80692.761
8 客户端	7170.386	32056.959	68761.908	88357.563

Tree创建

	1 进程	4 进程	16 进程	64 进程
1 客户端	305.778	229.562	86.299	23.917
2 客户端	161.31	211.119	76.301	24.439
4 客户端	260.562	223.153	81.209	23.867
8 客户端	350.038	220.744	81.621	17.144

Tree删除

	1 进程	4 进程	16 进程	64 进程
1 客户端	137.462	70.881	31.235	7.057
2 客户端	217.026	113.36	23.971	7.128
4 客户端	231.622	113.539	30.626	7.299
8 客户端	185.156	83.923	20.607	5.515

功能完整性评估

- Linux Test Project / fs

多种负载评估

- Database backup
- Java application logs
- Code git repo
- Database systems

  MyRocks, MySQL Innodb, HBase,

可扩展性评估

- 卷扩展性：单集群可以支持百万级别的cfs卷
- 元数据扩展性：单卷可以支持十亿级别文件或者目录

对ChubaoFS作出贡献

报告问题

请合理的使用关键字搜索 查找问题 ，以确保问题没有提交。然后通过 打开问题 提交问题描述和复现方法。

补丁指南

为了使补丁更可能被接受，您可能需要注意以下几点：

- 提交请求前需要先进行 文件系统压力测试

```
1.  runltp -f fs -d [MOUNTPOINT]
```

- 提供描述错误修复或新功能的提交消息
- DCO 是必须的，请在commit中添加 Signed-off-by

感谢

文中部分内容转载自 CoreDNS 和 Fluentd

环境及容量规划

环境要求

下面的表格列举了性能测试环境和生产环境的系统及硬件要求，您也可以参考容量规划章节，根据您的集群实际容量规划来精确订制部署方案。 注意，由于DataNode用到了CentOS7的特性，因此DataNode的内核版本必须高于CentOS7。

为了加快元数据读取速度，元数据都放在内存中，而DataNode数据主要占用磁盘资源，如果希望最大化利用节点资源，可以采用DataNode和MetaNode在同一节点混合部署的方式。

角色	规格	性能测试环境	生产环境
Master	CPU	>=4C	>=8C
	内存	>=4G	>=16G
	内核	CentOS 7	CentOS 7
	数量	3	3
DataNode	CPU	>=4C	>=4C
	内存	>=4G	>=8G
	内核	CentOS 7	CentOS 7
	硬盘容量	>=1TB	>=2TB
	硬盘类型	sata | ssd	sata | ssd
	文件系统	xfs | etx4	xfs | etx4
	数量	>=3	100~1000
MetaNode	CPU	>=4C	>=8C
	内存	>=8G	>=16G
	内核	CentOS 7	CentOS 7
	数量	>=4	100~1000
Client	CPU	>=2C	>=2C
	内存	>=4G	>=1G
	内核	CentOS 7	CentOS 7

容量规划

首先你要预估集群在未来相当长的一段时间内，最高预期文件数量和存储容量。 其次你还要对目前拥有的机器资源有清晰地了解。知道每台机器的内存大小、CPU核心数、磁盘容量。 如果您对以上数据了解清楚了，可以通过第二小节给出的经验参考值来看看自己的当前环境属于哪一种规模，能承载怎样的文件体量，或者需要针对当前文件体验需求应该准备多少资源，以防止频繁扩充机器资源。

文件总数量	文件总储量	集群总内存	集群总磁盘空间
10亿	10PB	2048 GB	10PB

大文件占比越高，MetaNode压力会越大。

当然，如果您觉得目前的资源足够使用，不需要一次性满足容量增长需求。那么可以及时关注MetaNode/ DataNode的容量预警信息。当内存或者磁盘即将使用完时，动态增加MetaNode/DataNode进行容量的调整。也就是说，如果发现磁盘空间不够了，可以增加磁盘或者增加DataNode，如果发现全部MetaNode内存过满，可以增加MetaNode来缓解内存压力。

多机房部署

如果你希望集群需要支持机房容错性，可以部署跨机房的ChubaoFS集群。同时需要注意，由于机房之间的通信延迟高于单机房，所以如果对于高可用的要求大于低延迟，可以选择跨机房部署方案。如果对性能要求更高，则建议单机房部署集群。 配置方案：在DataNode/MetaNode配置文件中修改zoneName参数，指定为所在机房名称，然后启动DataNode/MetaNode进程，则该机房会随着DataNode/MetaNode的注册而被Master存储并记录。

创建单机房volume:

```
1.  $ cfs-cli volume create {name} --zone-name={zone}
```

为了防止单机房volume初始化失败，请保证单个机房的DataNode不少于3，MetaNode不少于4

创建跨机房volume:

```
1.  $ cfs-cli volume create {name} --cross-zone=true
```

Q&A

- 如果你刚接触ChubaoFS，希望快速熟悉，请参考 启动docker集群
- 如果你对ChubaoFS感兴趣，希望使用ChubaoFS到您的生产环境，但是又要做性能测试来全方位体验和测评ChubaoFS，请参考 性能评估
- 如果你已经完成对ChubaoFS的测试评估，希望正式投入生产环境，想了解如何进行容量规划及环境配置，请参考 环境及容量规划
- 如果您希望了解ChubaoFS所使用的业务场景，可以参考 使用案例
- 如果您在实际生产环境中遇到了一些问题，不知道如何解决，接下来的内容可能会对你有所帮助。

为了描述方便，定义以下几个关键词缩写

全称	缩写	说明
Data Partition	dp	数据分区
Meta Partition	mp	元数据分区
Data Partition Replica	dpr	数据分区副本
Meta Partition Replica	mpr	元数据分区副本
NodeSet	ns	节点集
DataNode	dn	数据节点
MetaNode	mn	元数据节点

编译

1. 本机编译ChubaoFS，部署到其它机器上无法启动

首先请确认使用 `PORTABLE=1 make static_lib` 命令编译rocksdb，然后使用ldd命令查看依赖的库，在机器上是否安装，安装缺少的库后，执行 `ldconfig` 命令

1. undefined reference to ‘ZSTD_versionNumber’ 类似问题

可以使用下面两种方式解决

> - CGO_LDFLAGS添加指定库即可编译，
>
> 例如: `CGO_LDFLAGS="-L/usr/local/lib -lrocksdb -lzstd"` 这种方式，要求其他部署机器上也要安装 `zstd` 库
>
> - 删除自动探测是否安装zstd库的脚本
>
> 文件位置示例: rockdb-5.9.2/build_tools/build_detect_platform

> 删除的内容如下

```
# Test whether zstd library is installed
    $CXX $CFLAGS $COMMON_FLAGS -x c++ - -o /dev/null 2>/dev/null  <<EOF
      #include <zstd.h>
      int main() {}
EOF
    if [ "$?" = 0 ]; then
        COMMON_FLAGS="$COMMON_FLAGS -DZSTD"
        PLATFORM_LDFLAGS="$PLATFORM_LDFLAGS -lzstd"
        JAVA_LDFLAGS="$JAVA_LDFLAGS -lzstd"
    fi
```

节点或磁盘故障

如果节点或者磁盘发生故障，可以通过decommission操作将故障节点或者磁盘下线。

1. Datanode/Metanode下线

```
$ cfs-cli metanode/datanode decommission 192.168.0.21:17210
```

- 什么情况下下线mn/dn？

如果某节点发生了故障无法正常提供服务，为了提升系统可用性而不得不将该节点从集群中摘除，并且让这台机器上的数据自动迁移到其他健康的节点。

- 下线节点后，引起部分节点磁盘io和网络io突然增加

下线操作会触发数据自动迁移，消耗大量网络资源，因此在数据量较大的情况下，请尽量在低峰期进行，并且尽量避免同时下线多台节点。

- 如何判断下线完成？

> ```
> $ cfs-cli datapartition check
> ```
>
> bad partition ids没有信息，说明下线完成了。
>
> - 常见error1：没有可用的mn，此时所有mn内存或者磁盘已满，需要增加新的mn到集群中
> - 常见error2：端口号错误，每台mn的标识符应该是ip+port的组合，port为mn配置文件中的listen端口，不能写错

2. 磁盘下线

如果磁盘发生了故障而节点还能正常提供服务，此时建议你直接下线该磁盘。同样的，和dn/mn下线一

样，请尽量在低峰期、避免多次下线操作同时进行。

```
1.  $ cfs-cli disk decommission { disk } 192.168.0.11:17310
```

操作正确的话，会提示下线成功，并且该磁盘上的dp均被转移到了其他磁盘。

常见错误和mn/dn下线类似.

1. data partition/meta partition下线

    ```
    1. $ cfs-cli datapartition decommission 192.168.0.11:17310 {Partition ID}
    ```

操作正确的话，会提示下线成功，并且该dp被转移到了其他地址。

> - 什么情况下线partition？
>
>   > - 节点partition过多，下线部分partition减缓压力
>   > - 防止下线节点或磁盘时集群抖动明显
>
> - 常见错误和mn/dn下线类似

1. 如果磁盘写满了，会不会发生爆盘

通常在dn启动参数中建议用户配置disk参数中 {disk path}:{reserved space} 的 reserved space ，防止磁盘写满的情况。当剩余空间小于 reserved space，dn被设置为只读状态，避免了爆盘。

数据及元数据性能

1. 业务存在大量小文件，元数据规模庞大，怎样提升集群性能

ChubaoFS的元数据存储在内存中，提升mn机身内存或者横向扩展mn节点都会明显提升元数据性能，支撑海量小文件。

1. 如果集群中新增了dn/mn，后台是否会自动rebalance，将旧节点上的dp/mp自动转移到新节点？

不会的。考虑到rebalance会增加系统负载，而且增加数据丢失的风险，不会自动rebalance。如果希望新节点能承载更多的dp/mp，分散旧节点压力，可以自行给volume创建dp，或者将旧节点上的dp进行decommission下线操作。

1. 有很多批量删除文件的业务造成集群负载过高

设置和查看后台文件删除速率，默认值0，代表不限速。建议设置markdeleterate=1000,然后根据集群中节点的cpu状况动态进行调整。

```
$ cfs-cli cluster info 查看当前删除限速数值
$ cfs-cli cluster delelerate -h
 Set delete parameters

 Usage:
   cfs-cli cluster delelerate [flags]

 Flags:
       --auto-repair-rate string         DataNode auto repair rate
       --delete-batch-count string       MetaNode delete batch count
       --delete-worker-sleep-ms string   MetaNode delete worker sleep time with
millisecond. if 0 for no sleep
   -h, --help                            help for delelerate
       --mark-delete-rate string         DataNode batch mark delete limit rate.
if 0 for no infinity limit
```

容量管理

1. Volume空间不够用了怎么办?

```
$ cfs-cli volume expand {volume name} {capacity / GB} 增加volume容量
```

1. 如何提升Volume读写性能？

可读写的dp数量越多，数据就会越分散，volume的读写性能会有响应提升。ChubaoFS采取动态空间分配机制，创建volume之后，会为volume预分配一定的数据分区dp，当可读写的dp数量少于10个，会自动扩充dp数量。而如果希望手动提升可读写dp数量可以用以下命令:

```
$ cfs-cli volume create-dp {volume name} {number}
```

一个dp的默认大小为120GB，请根据volume实际使用量来创建dp，避免透支所有dp。

1. 如何回收Volume多余的空间

```
$ cfs-cli volume shrink {volume name} {capacity / GB} 减少volume容量
```

该接口会根据实际使用量计算，当设定值<已使用量的%120， 操作失败

1. 集群空间不够用了怎么办？

准备好新的dn和mn，启动配置文件配置现有master地址即可自动将新的节点添加到集群中。

分区（zone）相关

设置集群分区可以防止单个分区故障而引发整个集群不可用。每台节点启动的时候设置 cell 将自动加入该分区。

1. 查看分区信息

```
$ cfs-cli zone list
```

1. 不小心错误设置了volume分区，希望改变分区

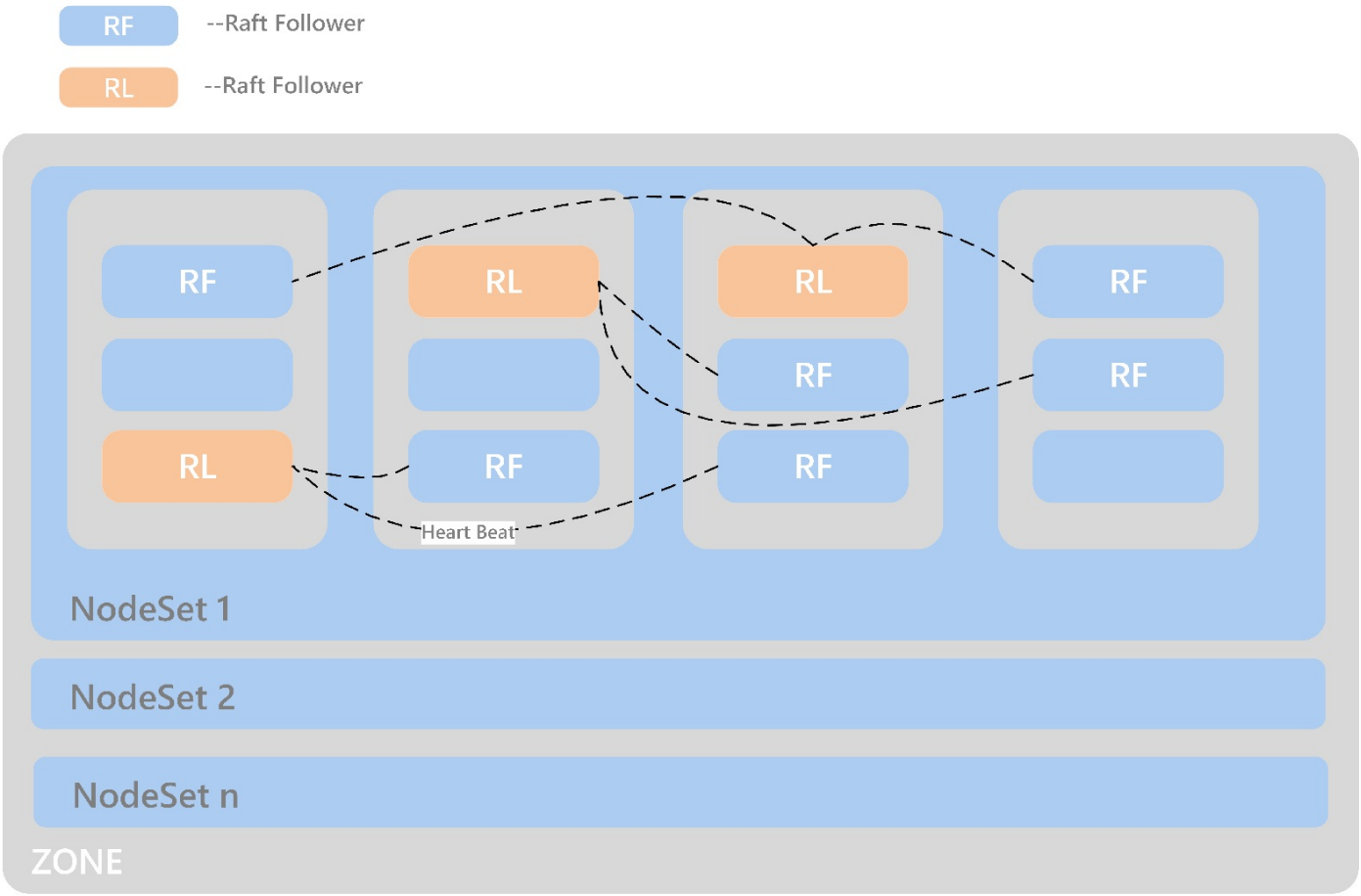
```
$ cfs-cli volume update {volume name}  --zone-name={zone name}
```

1. MetaNde和DataNode没有设置分区会怎么样？

集群中大部分参数都是有默认值的，默认分区名字为default。需要注意的是一个分区内必须同时有足够的dn和mn，否则在该分区创建volume，要么数据分片初始化失败，要么元数据分片初始化失败。

1. NodeSet的意义？

每个zone会有若干nodeset，每个nodeset的默认容量为18个节点。因为ChubaoFS实现了multi-raft，每个node启动了一个raft server进程， 每个raft server管理该节点上的m个raft实例，如果这些raft实例的其他复制组成员分布在n个node上，raft实例之间会发送raft心跳，那么心跳会在n个节点之间传递，随着集群规模的扩大，n也会变得比较大。而通过nodeset限制，心跳在nodeset内部相对独立，避免了集群维度的心跳风暴,我们是使用了multi raft和nodeset机制一起来避免产生raft心跳风暴问题。



1. 一个volume的dp/mp在NodeSet中如何分布？

dp/mp在ns中均匀分布，每创建一个dp/mp，都会从上一个dp/mp所在的ns开始轮询，查找可用的ns进行创建。

1. NodeSet数量如何规划？

对于3副本的dp/mp，只有当一个ns中存在至少3个可用节点时，dp/mp才会选择该ns。count(ns) >= 18*n + 3

节点状态异常

通过cli工具查看节点状态信息

```
$ cfs-cli datanode list
[Data nodes]
ID        ADDRESS               WRITABLE    STATUS
7         192.168.0.31:17310    No          Inactive
8         192.168.0.32:17310    No          Inactive
9         192.168.0.33:17310    Yes         Active
10        192.168.0.35:17310    Yes         Active
11        192.168.0.34:17310    Yes         Active
```

```
$ cfs-cli metanode list
[Meta nodes]
ID        ADDRESS               WRITABLE    STATUS
2         192.168.0.21:17210    No          Inactive
3         192.168.0.22:17210    No          Inactive
4         192.168.0.23:17210    Yes         Active
5         192.168.0.25:17210    Yes         Active
6         192.168.0.24:17210    Yes         Active
```

1. Datanode可写状态 WRITABLE=No 原因排查

> - 节点正在等待下线操作完成
> - 节点磁盘已满
> - 节点刚刚启动正在从本地恢复数据

1. Metanode可写状态 WRITABLE=No 原因

> - 节点正在等待下线操作完成
> - 节点内存空间已经达到totalmemory设定的值已满
> - 节点刚刚启动正在从本地恢复数据

1. 由三台master组成的集群中坏掉了一台，剩余两台重启能否正常提供服务？

可以。由于Master使用了RAFT算法，在剩余节点数量超过总节点数量50%时，均可正常提供服务。

1. 节点 STATUS=Inactive 原因排查

> - 节点和master的网络连接中断，需要检查网络状况，恢复网络连接
> - 节点进程挂掉，需要查看节点的server进程是否异常终止，此时重启进程即可恢复

系统升级

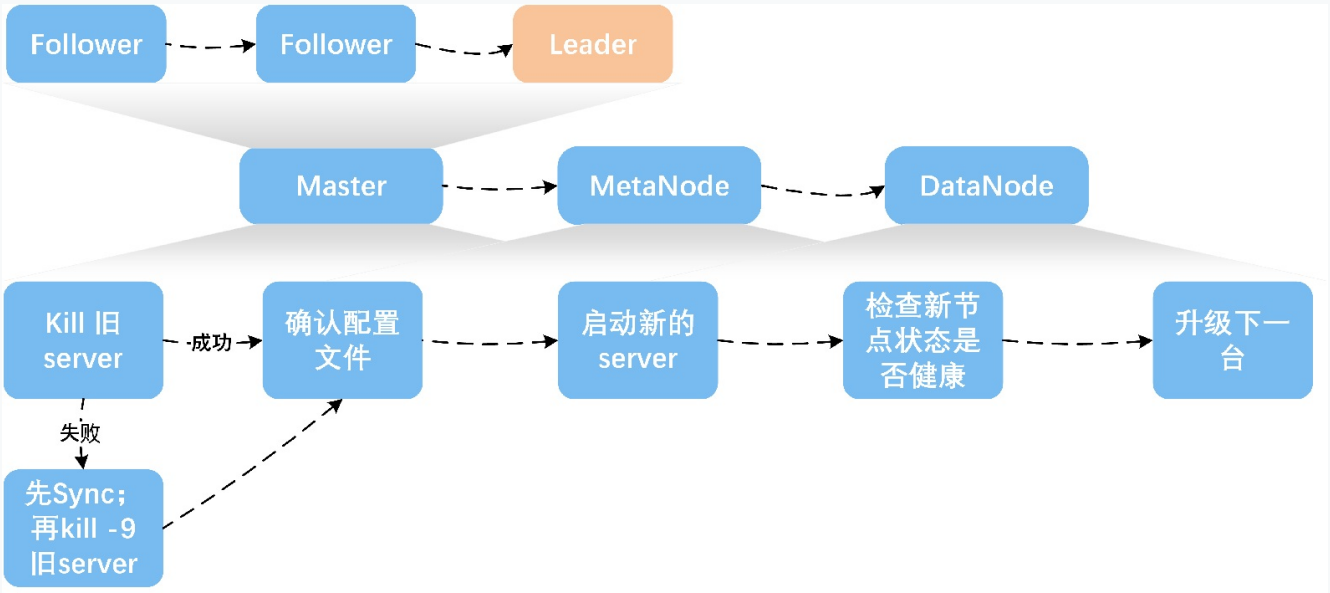
1. 升级步骤

> 1. 从ChubaoFS官方网站下载最新二进制文件压缩包 https://github.com/chubaofs/chubaofs/releases,解压得到二进制server
> 2. 冻结集群
>
> ```
> $ cfs-cli cluster freeze true
> ```
>
> 1. 确认启动配置文件，不要更改配置文件中的数据目录、端口等重要信息
> 2. 停止旧的server进程
> 3. 启动新的server进程

4. 检查确认升级后节点状态恢复健康 IsActive: Active

```
$ cfs-cli datanode info 192.168.0.33:17310
[Data node info]
  ID                  : 9
  Address             : 192.168.0.33:17310
  Carry               : 0.06612836801123345
  Used ratio          : 0.0034684352702178426
  Used                : 96 GB
  Available           : 27 TB
  Total               : 27 TB
  Zone                : default
  IsActive            : Active
  Report time         : 2020-07-27 10:23:20
  Partition count     : 16
  Bad disks           : []
  Persist partitions  : [2 3 5 7 8 10 11 12 13 14 15 16 17 18 19 20]
```

1. 升级下一节点(为了减少对客户端的影响，尤其是在比较大的用户体量下，需要逐一升级MetaNode节点),升级顺序如图所示



1. 升级一台Master之后，发现监控系统中没有及时显示？

检查这台master节点的配置信息是否正确，尤其是id号；查看master error日志是否大量报错 no leader，同时在master warn日志中查询关键字leaderChange查看leader变更原因，然后查看raft warn日志进一步分析。

1. 升级时能否修改配置文件端口号？

不能，ip+端口号 构成mn和dn实例的唯一标识符，修改之后会被当成新的节点。

在线修改配置

1. 修改mn threshold

```
$ cfs-cli cluster set threshold { value }
```

1. 修改集群配置
2. 修改volume配置

```
$ cfs-cli volume set -h
Set configuration of the volume
Usage:
  cfs-cli volume set [VOLUME NAME] [flags]
Flags:
      --authenticate string    Enable authenticate
      --capacity uint          Specify volume capacity [Unit: GB]
      --enable-token string    ReadOnly/ReadWrite token validation for fuse
client
      --follower-read string   Enable read form replica follower
  -h, --help                   help for set
      --replicas int           Specify volume replicas number
  -y, --yes                    Answer yes for all questions
      --zonename string        Specify volume zone name
```

1. 修改日志级别

提供了在线修改master、MetaNode、DataNode日志级别的接口

```
$ http://127.0.0.1:{profPort}/loglevel/set?level={log-level}
```

支持的 log-level 有 debug,info,warn,error,critical,read,write,fatal

离线修改配置

1. 修改master IP地址

三节点master的ip地址更换之后，需要将所有的mn、dn以及其他引用了master ip地址的应用在修改配置后重启

1. 修改DataNode MetaNode端口

不建议修改dn/mn的端口。因为dn/mn在master中是通过ip:port进行注册的。如果修改了端口，

master则会认为其为全新节点，旧节点是 Inactive 状态。

1. 修改MetaNode totalmemory

Total memory是指MetaNode总内存大小，当MetaNode的内存占用高于此值，MetaNode变为只读状态。通常该值要小于节点内存，如果MetaNode和DataNode混合部署，则需要给DataNode预留内存空间。

1. 修改DataNode 保留空间

dn启动配置文件中，disk参数后半部分的数字即为 Reserved Space 值，单位byte，修改完后启动即可。

```
{ ...
"disks": [
    "/cfs/disk:10737418240"
  ],
  ...
  }
```

1. 其他配置请参考 资源管理节点 数据节点 元数据节点客户端.

日志处理

1. 每天产生几十GB日志，占用过多的磁盘空间怎么办？

如果您是开发及测试人员，希望进行调试，可以将日志级别设置为Debug或者info， 如果生产环境，可以将日志级别设置为warn或者error,将大大减少日志的量

```
$ http://127.0.0.1:{profPort}/loglevel/set?level={log-level}
```

支持的 log-level 有 debug,info,warn,error,critical,read,write,fatal

1. Datanode warn日志

```
checkFileCrcTaskErr clusterID[xxx] partitionID:xxx File:xxx badCrc On xxx:
```

日志分析：在Master的调度下，dn会每隔几个小时进行crc数据校验。此报错说明crc校验未通过，文件数据出错了。此时需要根据报错信息中的partitionID和File并借助Datanode日志分析文件数据出错的原因。

1. Datanode error日志
2. Master error 日志

```
clusterID[xxx] addr[xxx]_op[xx] has no response util time out
```

日志分析：Master向mn或dn发送[Op]命令时响应超时，检查Master和mn/dn网络连通性；查看dn/mn服务进程是否还在。

1. Master warn 日志
2. Metanode error日志

```
Error metaPartition(xx) changeLeader to (xx):
```

日志分析：切换leader。正常行为

```
inode count is not equal, vol[xxx], mpID[xx]
```

日志分析：inode数量不一致。因为写入数据时，三副本中只要有2个副本成功就算成功了，所以会存在三副本不一致的情况。查看日志了解具体原因。

1. Metanode warn 日志
2. Client warn日志

```
operation.go:189: dcreate:
packet(ReqID(151)Op(OpMetaCreateDentry)PartitionID(0)ResultCode(ExistErr))
mp(PartitionID(1) Start(0) End(16777216) Members([192.168.0.23:17210
192.168.0.24:17210 192.168.0.21:17210]) LeaderAddr(192.168.0.23:17210)
Status(2)) req({ltptest 1 1 16777218 test.log 420}) result(ExistErr)
```

日志分析：ExistErr说明在rename操作中，文件名已存在。属于上层业务操作问题。Client运维人员可以忽略此日志。

```
extent_handler.go:498: allocateExtent: failed to create extent, eh(ExtentHan
ResultCode NOK,
packet(ReqID(xxxxx)Op(OpCreateExtent)Inode(0)FileOffset(0)Size(86)PartitionI
datapartionHosts(1.1.0.0:17310) ResultCode(IntraGroupNetErr))
```

日志分析：client向一个mp发送创建extent的请求返回失败，会尝试请求其他mp。

1. Client error日志

```
appendExtentKey: packet(%v) mp(%v) req(%v) result(NotExistErr)
```

日志分析：该错误说明写入文件时文件被删除了，属于上层业务操作问题。Client运维人员可以忽略此日志。

```
conn.go:103:sendToMetaPartition: retry failed
req(ReqID(xxxx)Op(OpMetaInodeGet)PartitionID(0)ResultCode(Unknown
ResultCode(0)))mp(PartitionID(xxxx) Start(xxx) End(xxx) Members([xxx xxxx
xxxx]) LeaderAddr(xxxx) Status(2)) mc(partitionID(xxxx) addr(xxx))
err([conn.go 129] Failed to read from conn,
req(ReqID(xxxx)Op(OpMetaInodeGet)PartitionID(0)ResultCode(Unknown
ResultCode(0))) :: read tcp 10.196.0.10:42852->11.196.1.11:9021: i/o
timeout) resp(<nil>)
```

日志分析1：client和metanode网络连接异常，根据报错信息“10.196.0.10:42852->11.196.1.11:9021”，检查这两个ip地址之间通路是否正常

日志分析2：检查“11.196.1.11:9021” 上的metanode进程是否挂了

1. Raft warn日志
2. Raft error日志

```
raft.go:446: [ERROR] raft partitionID[1105] replicaID[6] not active
peer["nodeID":"6","peerID":"0","priority":"0","type":"PeerNormal"]
```

日志分析：该错误是因为网络压力过大而导致延迟增加，超过raft选举时间间隔，raft复制组失去leader。网络恢复后，重新选举leader，该报错会自行消失。

数据丢失及一致性

1. 单个dn/mn数据全部丢失

该情况可以等同于dn/mn故障，可以通过decommission下线节点，然后重启节点重新注册节点到Master，则Master将其视为新的成员。

1. 不小心删除了dn中某个dp目录下的文件数据

dn有自动修复数据的功能，如果长时间数据仍未修复，可以手动重启当前dn进程，会触发数据修复流程。

Fuse客户端问题

1. 内存及性能优化问题

- Fuse客户端占用内存过高，超过了2GB，对其他业务影响过大

  离线修改：在配置文件中设置readRate和writeRate参数，重启客户端

> 在线修改: http://{clientIP}:{profPort} /rate/set?write=800&read=800

- Fuse客户端性能优化请参考(https://chubaofs.readthedocs.io/zh_CN/latest/user-guide/fuse.html)

1. 挂载问题

- 支持子目录挂载吗？

支持。配置文件中设置subdir即可

- 挂载失败的原因有哪些

  - 挂载失败后，输出以下信息

```
$ ... err(readFromProcess: sub-process: fusermount: exec:
"fusermount": executable file not found in $PATH)
```

查看是否已安装fuse，如果没有则安装

```
$ rpm –qa|grep fuse
  yum install fuse
```

  - 检查挂载目录是否存在
  - 检查挂载点目录下是否为空
  - 检查挂载点是否已经umount
  - 检查挂载点状态是否正常，若挂载点 mnt 出现以下信息,需要先umount，再启动client

```
$ ls -lih
ls: cannot access 'mnt': Transport endpoint is not connected
total 0
6443448706 drwxr-xr-x 2 root root 73 Jul 29 06:19 bin
 811671493 drwxr-xr-x 2 root root 43 Jul 29 06:19 conf
6444590114 drwxr-xr-x 3 root root 28 Jul 29 06:20 log
         ? d????????? ? ?    ?     ?            ? mnt
 540443904 drwxr-xr-x 2 root root 45 Jul 29 06:19 script
```

  - 检查配置文件是否正确，master地址 、volume name等信息
  - 如果以上问题都不存在，通过client error日志定位错误，看是否是metanode或者master服务导致的挂载失败

1. IO问题

- IOPS过高导致客户端占用内存超过3GB甚至更高，有没有办法限制IOPS?

通过修改客户端rate limit来限制客户端响应io请求频率。

```
#查看当前iops：
$ http://[ClientIP]:[profPort]/rate/get
#设置iops，默认值-1代表不限制iops
$ http://[ClientIP]:[profPort]/rate/set?write=800&read=800
```

- ls等操作io延迟过高

  - 因为客户端读写文件都是通过http协议，请检查网络状况是否健康
  - 检查是否存在过载的mn，mn进程是否hang住，可以重启mn，或者扩充新的mn到集群中并且将过载mn上的部分mp下线以缓解mn压力

1. 多客户端并发读写是否强一致？

不是。ChubaoFS放宽了POSIX一致性语义，它只能确保文件/目录操作的顺序一致性，并没有任何阻止多个客户写入相同的文件/目录的leasing机制。这是因为在容器化环境中，许多情况下不需要严格的POSIX语义，即应用程序很少依赖文件系统来提供强一致性保障。并且在多租户系统中也很少会有两个互相独立的任务同时写入一个共享文件因此需要上层应用程序自行提供更严格的一致性保障。

1. Fsync chunk
2. 能否直接杀死client进程，来停止client服务

不建议，最好走umount流程，umount后，client进程会自动停止。