数据加速器 GooseFS

GooseFS

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。
您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

版权所有：腾讯云计算（北京）有限责任公司

版权所有：腾讯云计算（北京）有限责任公司

GooseFS
产品简介
产品概述

最近更新时间：2024-10-31 17:11:42

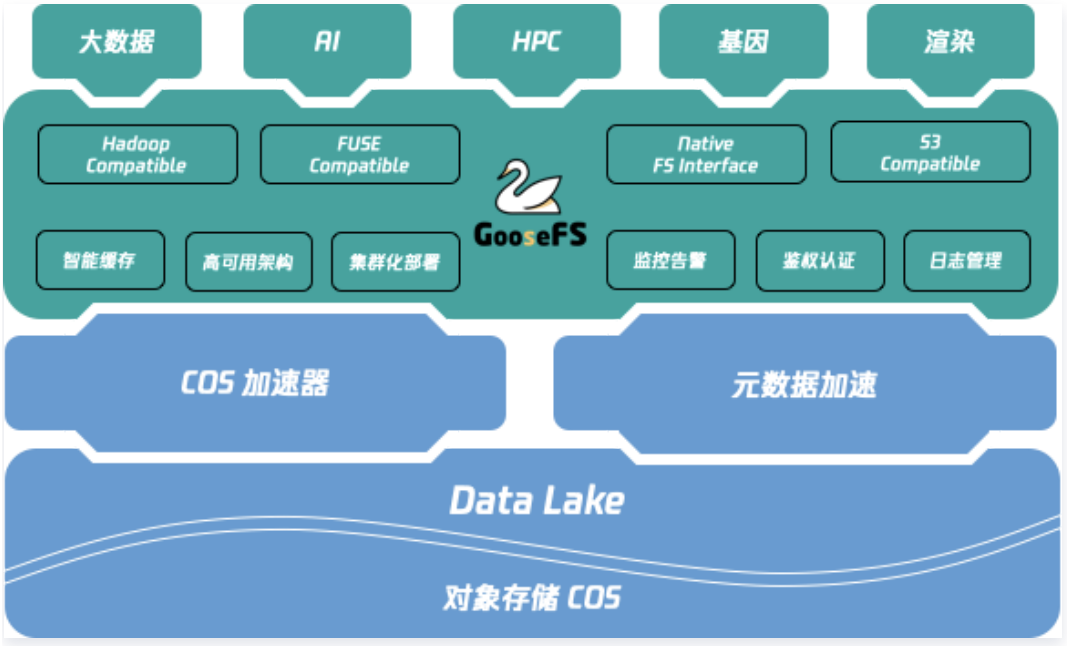
数据加速器（Data Accelerator Goose FileSystem，GooseFS），是由腾讯云推出的高可靠、高可用、弹性的数据加速服务。依靠对象存储（Cloud Object Storage，COS）作为数据湖存储底座的成本优势，为数据湖生态中的计算应用提供统一的数据湖入口，加速海量数据分析、机器学习、人工智能等业务访问存储的性能；采用了分布式集群架构，具备弹性、高可靠、高可用等特性，为上层计算应用提供统一的命名空间和访问协议，方便用户在不同的存储系统管理和流转数据。

产品功能

数据加速器有两个子产品：GooseFS 和 GooseFSx。

数据加速器 GooseFS

GooseFS 旨在提供一站式的缓存解决方案，在利用数据本地性和高速缓存，统一存储访问语义等方面具有天然的优势；GooseFS 在腾讯云数据湖生态中扮演着“上承计算，下启存储”的核心角色，如下图所示。



GooseFS 提供了以下功能：

1. 缓存加速和数据本地化（Locality）：GooseFS 可以与计算节点混合部署提高数据本地性，利用高速缓存功能解决存储性能问题，提高写入对象存储 COS 的带宽。
2. 融合存储语义：GooseFS 可以支持 HDFS 、POSIX 等协议，支持对接腾讯云 COS、云 HDFS 等多种云存储，适用于多种生态和应用场景。
3. 统一的腾讯云相关生态服务：包括日志、鉴权、监控，实现了与 COS 操作统一。
4. 提供 Namespace 管理能力：针对存储在腾讯云 COS、云 HDFS 等多种业务数据，提供不同的读写缓存策略和生命周期管理能力。

版权所有：腾讯云计算（北京）有限责任公司

5. 感知 Table 元数据功能：对于大数据场景下数据 Table，提供 GooseFS Catalog 用于感知元数据 Table ，提供 Table 级别的 Cache 预热。

版权所有：腾讯云计算（北京）有限责任公司

产品优势

最近更新时间：2023-09-20 20:30:51

数据加速器（Data Accelerator Goose FileSystem，GooseFS）在数据湖场景中具有如下几点明显的优势：

数据 I/O 性能

GooseFS 部署提供近计算端的分布式共享缓存，上层计算应用可以透明地、高效地从远端存储将需要频繁访问的热数据缓存到近计算端，加速数据 I/O 性能。GooseFS 提供了元数据缓存功能，可以加速大数据场景下查询文件数据以及列出文件列表等元数据操作的性能。配合大数据存储桶使用，还可进一步加速重命名文件的操作性能。此外，业务可以按需选择 MEM、SSD、NVME 以及 HDD 盘等不同的存储介质，平衡业务成本和数据访问性能。

存储一体化

GooseFS 提供了统一的命名空间，不仅支持了对象存储 COS 存储语义，也支持 HDFS、K8S CSI 以及 FUSE 等语义，为上层业务提供了一体化的融合存储方案，简化业务侧运维配置。存储一体化能够打通不同数据底座的壁垒，方便上层应用管理和流转数据，提升数据利用的效率。

生态亲和性

GooseFS 全兼容腾讯云大数据平台框架，也支持客户侧自定义的本地部署，具备优秀的生态亲和性。业务侧不仅可以在腾讯云弹性 MapReduce 产品中使用 GooseFS 加速大数据业务，也可以便捷地将 GooseFS 本地化部署在公有云 CVM 或者自建 IDC 内。此外，GooseFS 支持透明加速能力，对于已经使用腾讯云 COSN 和 CHDFS 的用户，只需做简单的配置修改，即可实现不修改任何业务代码和访问路径的前提下，自动使用GooseFS 加速 COSN 和 CHDFS 的业务访问。

版权所有：腾讯云计算（北京）有限责任公司

应用场景

最近更新时间：2023-09-20 20:30:51

开源生态数据湖

客户基于开源 Hadoop 生态构建大数据处理与分析，会面临计算资源与存储资源扩容速度不匹配、存储系统需对接多数据源的问题。

推荐产品

推荐数据加速器 GooseFS。

主要能力

- 计算存储分离
  通过计算与存储分离，实现计算资源弹性伸缩，满足客户对计算资源的灵活调度。
- 多数据源支持
  可对接多种数据源，允许存储任意规模的结构化、半结构化、非结构化数据。
- 高性能业务架构
  通过数据加速器（Data Accelerator Goose FileSystem，GooseFS）、元数据加速器、AZ 加速器等多级加速服务，提升计算业务访问性能。

交互式查询数据湖

客户在对象存储（Cloud Object Storage，COS）中存储了多种数据源数据，包括实时计算数据，需要对其中的数据进行 OLAP 分析并进行数据可视化展示。

主要能力

- 多数据源支持
  可对接多种数据源，允许存储任意规模的结构化、半结构化、非结构化数据。
- 性能加速
  通过数据加速器、元数据加速器、AZ 加速器等多级加速服务，实现超越本地 HDFS 的性能。

机器学习数据湖

在经典机器学习场景中，训练数据量大，同时要求很大的内网带宽。

主要能力

- 超大带宽
  可以提供超大的内网带宽，满足机器学习场景大带宽需求。
- 多数据源支持
  可对接多种数据源，允许存储任意规模的结构化、半结构化、非结构化数据。
- 性能加速
  通过数据加速器、元数据加速器、AZ 加速器等多级加速服务，实现超越本地 HDFS 的性能。

云原生数据湖

版权所有：腾讯云计算（北京）有限责任公司

通过容器服务，结合 Flink、TensorFlow 等开源应用，搭建云原生数据 ETL 集群和分析集群，实现计算资源的弹性化；通过数据加速器、元数据加速器、AZ 加速器等多级加速服务，提升计算业务访问性能；通过对象存储服务作为数据湖存储底座，实现海量异构数据的低成本存储。

主要能力

- 计算存储分离
  通过计算与存储分离，实现计算资源弹性伸缩，满足客户对计算资源的灵活调度。
- 高性能业务架构
  通过数据加速器、元数据加速器、AZ 加速器等多级加速服务，提升计算业务访问性能。
- 丰富生态支持
  可存储 Parquet、ORC 多种格式数据源，支持 Spark、Presto、Flink 等多种大数据插件。

版权所有：腾讯云计算（北京）有限责任公司

适用地域

最近更新时间：2025-04-14 15:41:02

GooseFS 的适用地域如下表所示，如果您有其他地域需求请联系 在线客服 。

管控模式	适用地域
管控面托管	北京, 广州, 上海，南京，弗吉尼亚，上海自动驾驶云
Master 托管	北京, 广州, 上海，南京，弗吉尼亚，上海自动驾驶云
全托管	北京, 广州, 上海，南京，弗吉尼亚，新加坡，上海自动驾驶云

版权所有：腾讯云计算（北京）有限责任公司

版本发布

1.4.8 版本正式发布

最近更新时间：2025-03-14 15:49:11

腾讯云对象存储团队正式发布 GooseFS 1.4.8版本，该版本实现元数据分库分表、增加非原子删除等功能，增加元数据相关监控指标，修复由于 Glibc Malloc 导致内存放大、ClearMetadata 混合部署风险等问题，提升 GooseFS 集群可用性。

重要更新点一：元数据平行扩展能力

为了提高磁盘的性能和可扩展性，GooseFS 1.4.8版本采用全新的存储架构将数据分散存储在不同的磁盘中。
之前老架构中，元数据存放在单块磁盘上。现在经过元数据分库分表后，可以使用多块物理磁盘上的节点。完成分库分表后，新架构存储空间提升数倍，集群规模大幅提升。

重要更新点二：Checkpoint 组织结构优化

在分布式存储系统 RocksDB 中，Checkpoint 机制是确保数据一致性和支持故障恢复的关键。传统的将数据目录打包成单文件的方法可能会引入数据拷贝和压缩的开销，影响性能。
GooseFS 1.4.8版本通过深度优化 Checkpoint 的组织结构，实现秒级别加载和生成 Checkpoint，从而达到秒级别恢复能力。

新增监控和告警指标

监控指标	监控描述
Master.LastAppliedCommitRaftJournalIndex	已经 apply 的 goosefs 日志的 raft index
Master.LastAppliedCommitJournalIndex	已经 apply 的 goosefs 日志的 index
Master.JournalEntriesSinceCheckPoint	新增 raft log 的条目数目
Master.JournalCheckpointWarn	未做 checkpoint 的告警
Master.EmbeddedJournalLastSnapshotDurationMs	备机做 checkpoint 的时间
Master.EmbeddedJournalLastSnapshotReplayDurationMs	冷启动 replay checkpoint 时间
Master.EmbeddedJournalLastSnapshotJournalIndex	snapshot 中包含的 goosefs log index
Master.EmbeddedJournalLastSnapshotRaftJournalIndex	snapshot 中包含的 raft log index
Master.EmbeddedJournalEntriesSinceCheckPoint	新增 goosefs log 的条目数目
Master.EmbeddedJournalLastSnapshotDownloadDurationMs	Leader 下载 snapshot 的消耗时

	间
Master.EmbeddedJournalLastSnapshotDownloadDiskSize	Leader 下载 snapshot 的大小
Master.EmbeddedJournalLastSnapshotDownloadStat	Leader 下载snapshot 的状态
Client_BLOCK_CACHE_DATA_HIT_DbName_*	Rocksdb 缓存命中次数
Client_BLOCK_CACHE_DATA_MISS_DbName_*	Rocksdb 缓存丢失次数
Master_rocksdb_size_all_mem_tables_Column_Family_Tag:Dbname_*	Rocksdb 中 Memtable 内存占用
Master_rocksdb_estimate_table_readers_mem_Column_Family_Tag:DbName_*	Rocksdb 中 Index 内存占用
Master_rocksdb_block_cache_capacity_Column_Family_Tag:DbName_*	Rocksdb 中 BlockCache 内存占用
Master_RocksInodeRemoveTimer	remove 分位图和 qps
Master_RocksInodeWriteInodeTimer	writeInode 分位图和 qps
Master_RocksInodeAddChildTimer	AddChild 分位图和 qps
Master_RocksInodeRemoveChildTimer	RemoveChild 分位图和 qps
Master_RocksInodeGetMutableTimer	GetMutable 分位图和 qps
Master_RocksInodeGetChildIdsTimer	GetChildIds 分位图和 qps
Master_RocksInodeGetChildIdTimer	GetChildId 分位图和 qps
Master_RocksInodeGetChildTimer	GetChild 分位图和 qps
Master_RocksInodeHasChildrenTimer	HasChildren 分位图和 qps
Master_RocksInodeAllEdgesTimer	AllEdges 分位图和 qps
Master_RocksInodeAllInodesTimer	AllInodes 分位图和 qps
Master_RocksInodeWriteBatchTimer	WriteBatch 分位图和 qps
Master_RocksBlockGetBlockTimer	GetBlock 分位图和 qps
Master_RocksBlockPutBlockTimer	PutBlock 分位图和 qps
Master_RocksBlockRemoveBlockTimer	RemoveBlock 分位图和 qps
Master_RocksBlockGetLocationsTimer	GetLocation 分位图和 qps
Master_RocksBlockAddLocationTimer	AddLocation 分位图和 qps
Master_RocksBlockRemoveLocationTimer	RemoveLocation 分位图和 qps

告警指标	监控描述
Master.JournalCheckpointWarn	超过12小时未完成做 checkpoint

1.4.7 版本正式发布

最近更新时间：2024-08-15 15:10:42

腾讯云对象存储团队正式发布了 GooseFS 1.4.7 版本，该版本针对低延迟数据仓库以及 OLAP 场景提供了全新的数据存储模型，同时针对故障容错与降级做了统一的优化和重构。

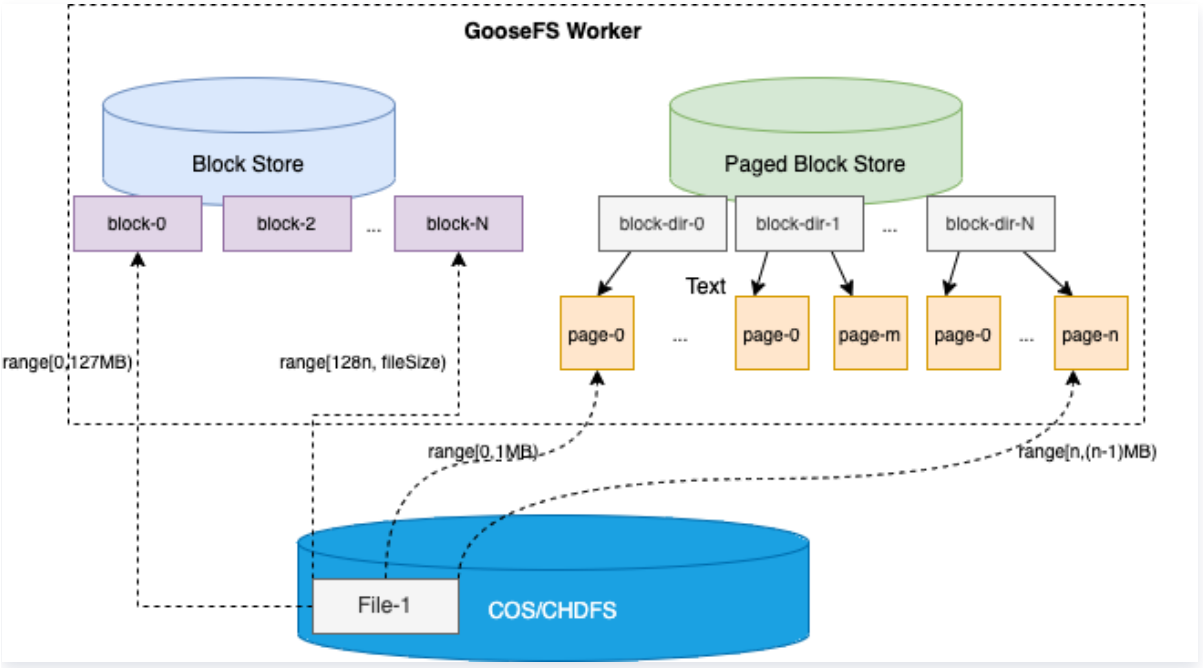
重要更新点一：Page Store 存储模型

在以 ORC / Parquet 格式文件构建的数据仓库以及其他 OLAP 数据库的应用场景中，大量的随机读 IO 会带来数据缓存的放大问题：

- 数据以 Block 形式缓存时，即使只需要 Block 中一小段的热数据，也会对 UFS 产生整个 Block 级别的读取带宽放大，对响应延迟有极大地影响；
- 包含热点数据区段的 Block 都会被缓存到 GooseFS 中，GooseFS 的缓存空间被明显放大，导致缓存空间利用率降低。

上述问题尤其是在数据文件本身比较大，但是随机 IO 却相对比较小数据查询分析场景中，尤为凸显。

因此，GooseFS 在 1.4.7 版本中全新引入了 Page Store 的存储模型设计，在不增加 Master 节点的元数据管理负担的前提下，允许 GooseFS 的数据节点以 Page 这种更小粒度存储单元来缓存数据，极大地缓解了海量随机访问场景下的数据放大问题，下图是原有 Block Store 与新的 Page Store 的设计对比。

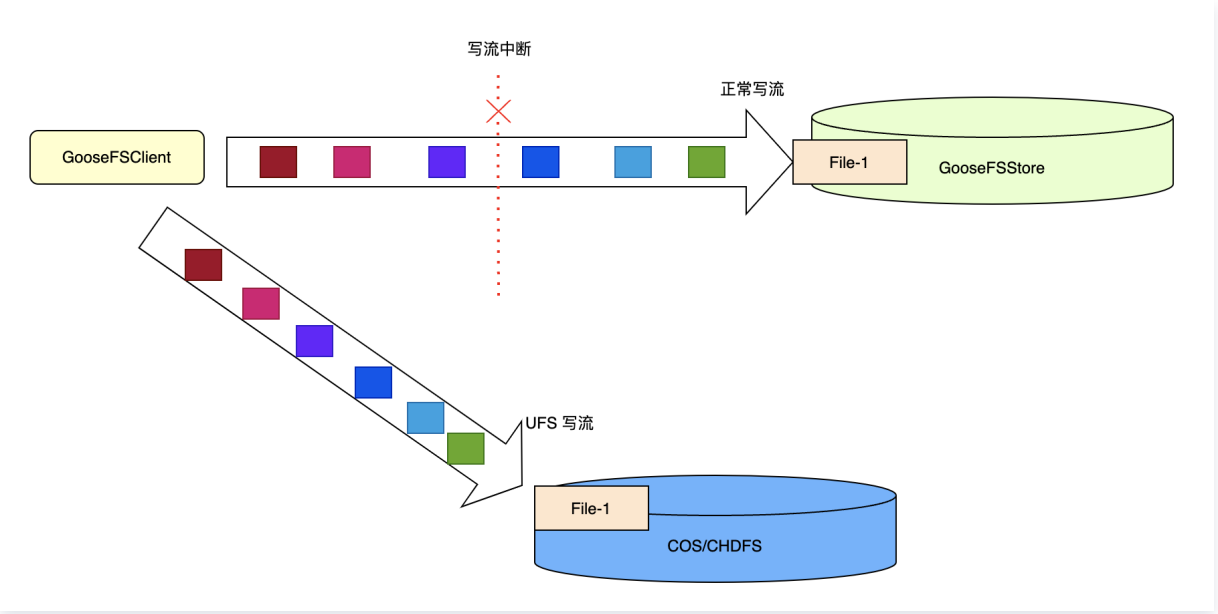


在引入了 PageStore 的设计以后，对于存放在 COS/CHDFS 等 UFS 上的数据，我们都可以按需请求与缓存。同时，在 Page Store 的设计中实现了单数据节点的缓存击穿优化，即使多个请求同时访问了一个热点数据区段，也能够有效地降低对 COS/CHDFS 的带宽访问冲击，提升了整个存算分离架构的稳定性。

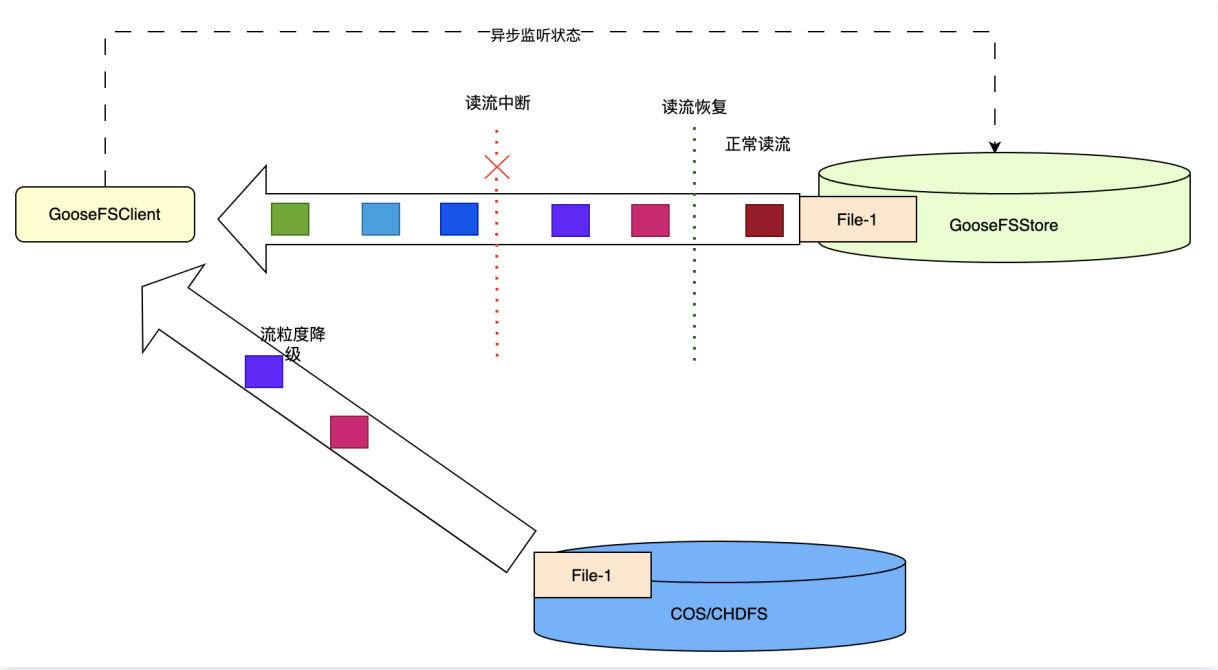
重点更新点二：统一故障降级能力

GooseFS 在 v1.4.7 版本中增强了全场景读写流的容灾能力，GooseFS 客户端会持续观测读写流状态，在状态异常时自动将流切到 Under FileSystem（一般为 COS/CHDFS），同时在 GooseFS 状态恢复后自动切回来。

参考下图，写 GooseFS 过程中如果出现异常，客户端写流会自动触发降级写，最终只要 COS/CHDFS 能够写成功，整体写流就认为成功。GooseFS 不会缓存异常点之后的数据，初次读时会从 COS/CHDFS 重新 Load。



参考下图，读 GooseFS 过程中读流如果出现异常，GooseFSClient 会重新建立 COS/CHDFS 的读流，并从故障 Position 位置续读，同时会启动异步线程监听 GooseFSStore 状态，故障恢复时会自动升级。



读写降级能力对客户屏蔽了 GooseFS 的故障情况，极大提高了 GooseFS 的容灾能力。

新增配置项说明

以下是在 GooseFS 1.4.7 版本中新增的一些配置项。

参数	默认值	描述
goosefs.worker.block.store.type	FILE	• 指定 Worker 上的存储类型，可选项为 FILE 和 PAGE。 • 默认为 FILE，即传统的 Block 存储模式，指定 PAGE 则为 Page 存储模式。
goosefs.worker.page.store.page.size	1MB	指定每个 Page 页的大小。默认大小为 1MB，这里可以按需指定，例如 128KB 或 256KB 亦可。
goosefs.worker.page.store.dirs	/tmp/goosefs-cache	指定 Page Store 的数据目录。例如：`/data/goosefs-data/paged-block` 。
goosefs.worker.page.store.size	512MB	指定 Page Store 的数据目录的大小，默认为 512MB，如果超出了容量限制，则会触发 Page 粒度的淘汰。
goosefs.worker.cache.request.pending.timeout	500ms	• 用来优化高并发冷读场景下的缓存击穿问题的超时等待选项，默认是 500 毫秒。 • 如果在同一 Worker 节点上发生了缓存击穿的并发读取，则后到的请求会尝试等待 500ms，以便于直接从缓存中直接返回数据，而不是穿透到底层存储。 • 若等待超时，才会从底层存储读取并返回。 • 当该值设置为小于等于 0 的值时候，则相当于关闭了缓存击穿优化。
goosefs.worker.page.store.overhead	0.1	Page 存储空间的预留分位。默认值为 0.1，表示会预留百分之十的空间作为保留空间。达到水位以后，开始触发淘汰动作。
goosefs.worker.page.store.evictor.class	com.qcloud.cos.goosefs.client.file.cache.evictor.LRUCacheEvictor	Page 存储的淘汰算法，支持的选项为： • com.qcloud.cos.goosefs.client.file.cache.evictor.LRUCacheEvictor； • com.qcloud.cos.goosefs.client.file.cache.evictor.LFUCacheEvictor。
goosefs.worker.page.store.eviction.retries	10	最大淘汰尝试次数，默认为 10 次。
goosefs.worker.page.store.evictor.lfu.logbase	2.0	指定 LFU 淘汰算法的 LogBase。
goosefs.worker.page.store.local.store.file.buckets	1000	存放 Paged Block 目录的 Hash 桶数目，默认为 1000。

版权所有：腾讯云计算（北京）有限责任公司

1.4.0 版本正式发布

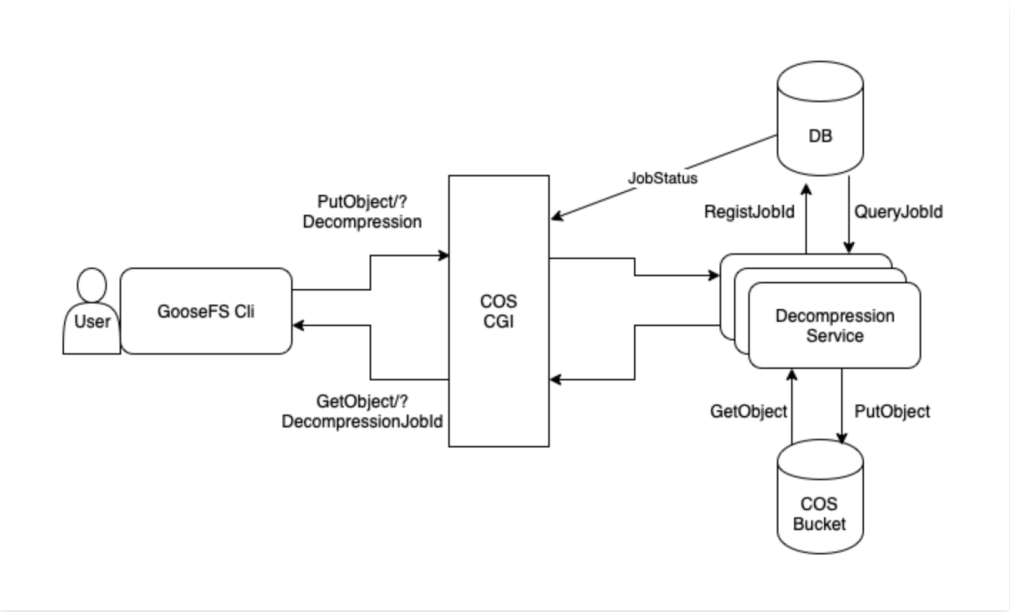
最近更新时间：2023-10-17 19:02:52

腾讯云存储团队正式发布数据加速器 GooseFS 1.4.0 版本，该版本针对 AI、大数据场景提供了文件解压缩等便捷易用的工具，同时针对海量文件读写下的集群性能和稳定性问题进行了针对性优化，提升了产品竞争力。

重要更新点一：提供文件解压缩能力

AI 场景下，业务团队可能会将大量用于训练或者学习的文件打包成一个压缩包并上传到对象存储中；在执行训练或者学习任务时，再将压缩包文件下载到本地并解压。这一流程会对底层对象存储服务产生较大的读带宽，每次启动任务时，无论需要读取多少文件，都需要将文件所处的压缩包整包下载才可以执行。

GooseFS 在本次更新中联合 COS 服务提供了服务端的解压缩能力，支持通过解压缩工具向 COS 服务端发起解压缩请求，提升文件访问性能。GooseFS 支持文件解压缩能力的基本框架如下：



整体流程上：

1. 首先通过 GooseFS 解压缩指令 `goosefs fs decompress` 向 COS 服务发起指定文件的解压缩请求。
2. COS 服务收到解压缩请求后，会向解压缩服务模块提交解压缩任务，由文件解压缩模块管理任务进度。
3. 解压缩过程中，用户可以通过 `goosefs fs queryDecompress` 指令查询解压缩任务的状态。
4. 解压缩任务完成后，完成解压后的文件会输出至用户指定的文件目录中。
5. 支持通过 `goosefs fs listDecompressJobs <namespace>` 指令查阅指定命名空间的解压缩任务进展。

> **说明**
> GooseFS 提供的解压缩能力目前仍然在**公测阶段**，公测阶段仅支持上海自动驾驶专区，暂不进行收费，如需使用请提交工单 申请。

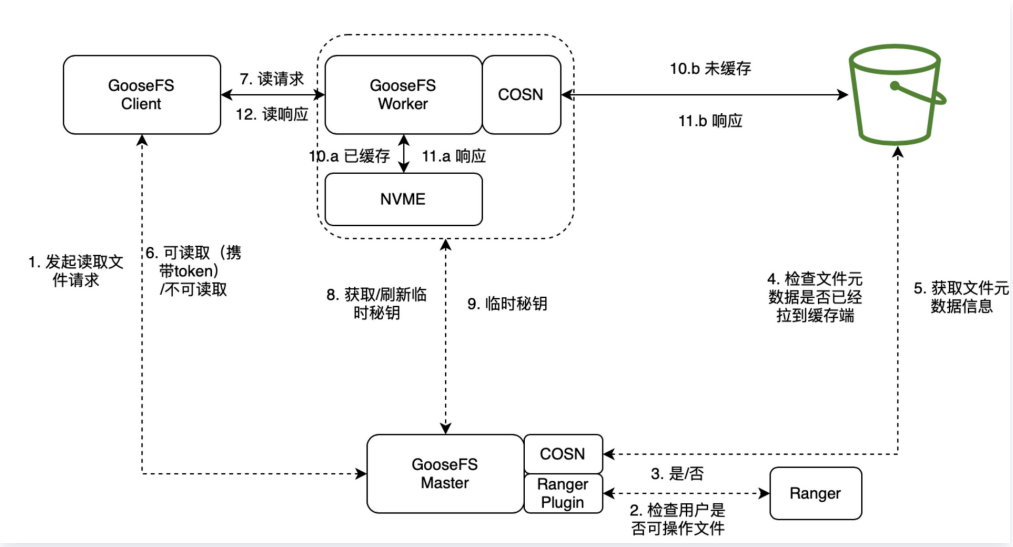
使用 GooseFS 文件解压缩能力的优势如下：

1. 避免文件读放大问题，减少底层对象存储服务的读带宽。用户在服务端侧完成解压缩后，可按需读取需要用到的文件，无需读取整个压缩包。

2. 减少客户端侧的 CPU 压力。用户无需在客户端侧执行解压缩操作，可以让宝贵的计算资源聚焦在 AI 计算任务上。

重要更新点二：支持临时密钥主动热更新

GooseFS 通过托管在集群中的密钥访问远端的对象存储服务。腾讯云的永久密钥具备永续的有效期，而临时密钥的有效期则可以由用户自行指定，最长不超过 2 小时。在 GooseFS 集群中托管永久密钥存在一定的安全风险，例如当永久密钥泄露时，对象存储服务中的文件将持续存在泄露的风险。此前，GooseFS 已经支持在 CVM、EMR 等环境中自动更新临时密钥，但是对于非上述场景中（例如裸金属、容器）却无法动态更新密钥，导致临时密钥过期后需要卸载后重新挂载。这个过程不仅繁琐，同时也会影响业务代码的运行。因此在本次更新中，GooseFS 团队进一步增强了临时密钥托管功能，允许用户主动地热更新临时密钥，并立即生效，确保密钥的有效性。
下面是对临时密钥托管的介绍，通过临时密钥托管服务，用户可以只在 Worker 节点上缓存从 Master 节点拉取的临时密钥信息，并通过临时密钥访问远端对象存储服务，获取业务所需数据。GooseFS 支持临时密钥托管服务的整体框架如下所示：



对于 CVM、EMR、CPM 等环境，只需要指定对应的 InstanceCredentialsProvider 即可，如在 CVM 机器可以用 CVMInstanceCredentialsProvider。
而对于容器来说，使用流程如下：
1. 在 Leader Master 节点中，可以周期性地通过以下指令，变更节点上的临时密钥信息。

```
goosefs ns update <namespace> [--secret <key=value>] [--attribute
fs.cosn.userinfo.sessionToken=xxx]
```

2. 客户端读取文件时，如果文件未缓存在 Worker 节点上，Worker 节点可以通过临时密钥访问远端对象存储服务拉取文件。
   临时密钥更新命令示例如下：

```
goosefs ns update testNamespace --wPolicy 1 --rPolicy 1 --secret
fs.cosn.userinfo.secretId=AKXXXX--secret fs.cosn.userinfo.secretKey=XXXXX --
secret fs.cosn.userinfo.sessionToken=XXXX --attribute fs.cosn.bucket.region=ap-
guangzhou --attribute
fs.cosn.credentials.provider=org.apache.hadoop.fs.auth.SessionTokenCredentialProv
ider
```

> **注意**
> update 需要填写完整的参数，否则缺失的参数会使用默认参数填充。

使用 GooseFS 临时密钥托管主要可以减少密钥泄露带来的安全风险。GooseFS 集群中可能管理成百上千台 Worker 节点，每一台 Worker 节点中都持久化永久密钥大大增加了密钥泄露的概率，使用临时密钥可以极大缓解此类风险。

重要更新点三：GooseFS-FUSE 客户端支持降级读

GooseFS-FUSE 可以在一台 Unix 机器上的本地文件系统中挂载一个 GooseFS 分布式文件系统。通过使用该特性，一些标准的命令行工具（例如 ls、cat 以及 echo）可以直接访问 GooseFS 分布式文件系统中的数据。GooseFS-FUSE 在访问 GooseFS 时，需要先到 GooseFS 集群中获取缓存文件，如果文件不存在，GooseFS 会到远端对象存储服务上拉取文件。如果 GooseFS 集群的 Master 节点异常（例如 Standalone 模式的 Master 节点宕机，HA 模式的多节点主备切换），导致集群整体不可用时，GooseFS-FUSE 将无法读取到文件，导致客户端也不可用。
GooseFS-FUSE 客户端在本期更新中新增了降级读能力，可以在 Master 节点异常时透传 FUSE 客户端的请求到远端对象存储服务上，这一能力有助于提升客户端整体的可用性。
整体流程上：

1. 默认情况下，GooseFS-FUSE 默认会去 Master 节点获取文件元数据信息，并读取 GooseFS 集群中的文件。
2. 节点异常的情况下，GooseFS-FUSE 会启用降级读模式，直接去远端对象存储中读取文件。

其他更新点

除了上述更新之外，我们在本次版本中优化了 GooseFS 的产品性能和稳定性，进一步提升 GooseFS 在大数据、AI 场景下的集群稳定性。主要更新点如下：

1. GooseFS distributedLoad 能力支持层级遍历能力,支持递归拉取指定目录下的元数据信息。
2. FUSE 随机读性能优化。
3. 增加 Master 查询/更新 RocksDB 的分位耗时监控，提升元数据服务的监控灵敏度。
4. 优化了 GooseFS HA 模式下的集群恢复时间，提升了集群可用性。
5. CosN 依赖版本升级，支持通过原生 HDFS 协议访问开启元数据加速的存储桶，提升大数据场景下的文件操作性能。
6. GooseFS 配置精简优化，减少了不必要的配置项，提升了配置易用性。
7. listInfo 精简优化。
8. 大文件顺序读优化。

同时，GooseFS 1.4.0 版本还修复了若干问题，其中存在潜在稳定性风险的重要修复点如下：

1. 修复 Worker 接收大量无效 async block 的请求。
2. 优化 Worker 上报时对孤立 block 的处理逻辑。

如果您想了解数据加速器 GooseFS 的更多信息，或者上手使用 GooseFS ，请参见 数据加速器 GooseFS 产品文档 。

1.3.0 版本正式发布

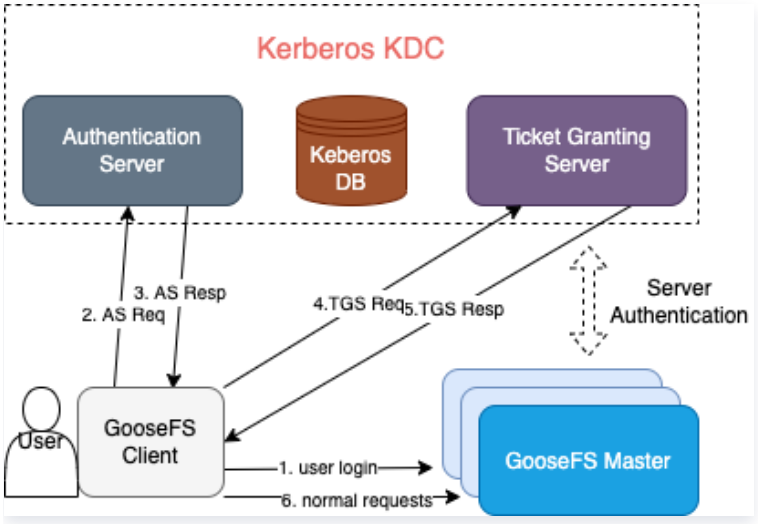
最近更新时间：2023-09-26 18:46:01

为了满足云上数据湖存储对安全、高性能的要求，腾讯云存储团队正式发布数据加速器 GooseFS 1.3.0 版本。该版本总结并收敛了 GooseFS 在过往大规模生产环境实践中遇到的性能、稳定性和安全问题，全面提升产品稳定性。

重要更新点一：支持 Kerberos 认证

Kerberos 用于非安全网络中，对个人通信以安全的手段进行身份认证。软件设计上采用客户端/服务器结构，并且能够进行相互认证，即客户端和服务器端均可对对方进行身份认证。可以用于防窃听、防重放攻击、保护数据完整性等场合，是一种应用对称密钥体制进行密钥管理的系统。目前 Kerberos 已经在大数据、AI 场景下被广泛应用于集群和文件系统之间的身份认证。

GooseFS 在本次更新中支持了 Kerberos 认证，支持将集群节点和用户访问接入 Kerberos 认证服务中，提供更安全的访问。GooseFS 支持 Kerberos 认证的基本框架如下：



GooseFS 集成 Kerberos 认证的主要优势点如下：

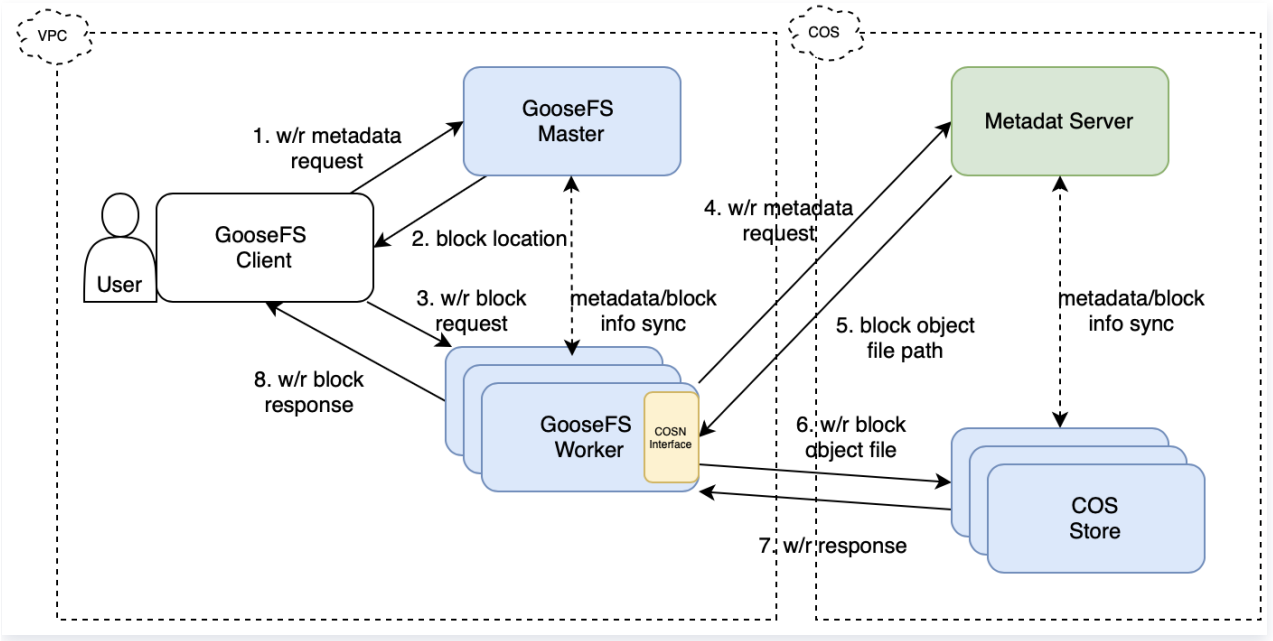
1. 可以保障 GooseFS 集群中的数据访问安全。
2. 与 HDFS 接入 Kerberos 的认证架构和流程基本一致，在 HDFS 上启用了 Kerberos 认证流程的应用可以很容易地迁移到 GooseFS。
3. 支持 Hadoop 的 Delegation Token 认证机制，因此可以很好地兼容 Hadoop 生态的应用作业。

如果需要了解 GooseFS 接入 Kerberos 的详细配置说明，可参见 GooseFS 接入 Kerberos 认证 。

重要更新点二：支持通过原生 POSIX 语义访问对象存储服务

对象存储服务通过 元数据加速能力 提供了原生的 POSIX 语义接口，支持用户通过文件系统语义访问对象存储服务，提供原生的元数据操作能力。系统设计指标可以达到 Gb 级单链接带宽、10万级 QPS 以及 ms 级延迟。启用元数据加速功能后，可以提升集群对元数据的操作性能，例如 List、Rename 等操作，可以广泛应用于大数据、高性能计算、机器学习、AI 等场景。

GooseFS 在本次更新中集成了最新版本的 COSN interface（COSN 8.14版本），支持了原生 POSIX 语义访问对象存储服务。整体的读写流程框架如下：



通过本次更新版本 GooseFS 以原生 POSIX 语义访问对象存储服务的主要步骤如下：

1. 确保您的存储桶已经开启元数据加速服务能力，元数据加速能力只能在创建存储桶时开启。
2. 安装最新版本的 GooseFS 客户端和服务端安装包。
3. 安装完成后，在 `core-site.properties` 文件中修改访问协议的配置，就可以通过原生的 POSIX 协议访问指定存储桶。

如果您需要了解更详细的操作步骤，可参见 数据加速器 GooseFS 最佳实践文档 。

重要更新点三：缓存淘汰能力优化

缓存淘汰能力是 GooseFS 为计算业务提供实时热数据的关键特性。在本次更新中，GooseFS 服务针对缓存淘汰能力提供了两项重点优化。

1. 提供了元数据清理工具
   元数据一般存储在 GooseFS Master 节点。在生产环境中，随着时间推移，线上元数据的数据量必然越来越大，但 Master 节点的存储容量始终有限，增长的元数据最终会导致 OOM。因此 GooseFS 在本次更新中提供了一个元数据清理工具，可以基于 `inode` 的 `expiretime` 检索出过期的文件元数据并执行清理动作。
   元数据的操作流程如下：
   1.1 从 Master 节点提供的 `Journal` 目录中提取 `Journal` 信息，并将其回放成 `InodeStore` 。
   1.2 检索 `FileInode` 进行过期检查，对过期 `Inode` 进行删除操作。
   1.3 检索 `DirectoryInode` 进行过期检查，对过期 `Inode` 进行删除操作。
2. Master 节点提供了 LRU 淘汰策略
   默认淘汰策略下， GooseFS Master 节点中的元数据增大时， `concurrentHashMap` 遍历一个 `segment` 的时间变长，淘汰的数据会越来越热，进而陷入缓存频繁置入置出的陷阱。在这种陷阱下，会出现缓存命中率下滑和缓存淘汰率增加的情况，进而影响 GC 性能，导致 GC 无法回收的情况。

因此，本次更新提供了基于 LRU 的淘汰策略。在该淘汰策略下，元数据缓存逃出频繁置换陷阱，能够提升缓存命中率，减少缓存淘汰速率，同时 GC 操作恢复正常，进一步减缓了元数据占用的内存增速。

其他更新点

除了上述更新之外，我们在本次版本中优化了 GooseFS 的产品性能，进一步提升 GooseFS 在大数据、AI 场景下的性能表现。主要更新点如下：

1. GooseFS Master 节点优化了锁瓶颈问题，显著提升 Master QPS，带来35%左右的性能优化效果。
2. GooseFS Worker 节点支持并发 Format，提升操作性能。
3. GooseFS Fuse 客户端支持覆盖写操作。
4. GooseFS Fuse 客户端优化了 `ls` 命令的内存占用问题。
5. GooseFS HDFS 客户端优化 ListNamespace 的性能。

同时，GooseFS 1.3.0 版本还修复了若干问题，其中存在潜在稳定性风险的重要修复点如下：

1. 修复了 RocksDB 泄漏和 Core 的问题，避免内存泄露。
2. 修复了 Zookeeper Curator 误打日志的问题。
3. 修复了 UFS 读带宽不准的问题。
4. 修复了 DistributedLoad 的时候由于日志打印过多导致的 LostWorker问题。

如果您想了解数据加速器 GooseFS 的更多信息，或者上手使用 GooseFS ，请参见 数据加速器 GooseFS 产品文档 。

版权所有：腾讯云计算（北京）有限责任公司

1.2.0 版本正式发布

最近更新时间：2024-10-12 11:37:31

腾讯云存储团队正式对外发布了数据加速器 GooseFS 1.2.0 版本，该版本总结并收敛了 GooseFS 在过往大规模生产环境实践中遇到的性能、稳定性和安全问题，全面提升产品稳定性。

重要更新点一：透明加速热开关

透明加速能力 可以让大数据用户能够使用 CosN scheme 访问 GooseFS，该特性方便用户在不修改已有表定义的前提下，使用 GooseFS 的功能，提升业务访问性能。

透明加速热开关主要用于提升系统的可运维性。在生产环境中使用 GooseFS 集群进行访问加速时，可能出现集群节点故障等各种问题，当集群无法自愈，并且需要尽快恢复现网业务时，需要有手段可以将访问流量在分钟级迅速切换到底层存储服务，然后在不影响计算作业的前提下，运维和管理 GooseFS 集群。

在具体使用过程中，可以通过如下指令启停透明加速热开关：

```
goosefs.user.client.transparent_acceleration.enabled = true | false
```

其中，`false` 代表开启透明加速能力，开启后所有访问请求会优先经过 GooseFS；`true` 代表关闭透明加速能力，关闭后所有访问请求会直接透传到底层。

重要更新点二：集成 CHDFS 认证和 Ranger 鉴权体系

Apache Ranger 是大数据生态系统中用于控制访问权限的一个标准鉴权组件，GooseFS 作为大数据和数据湖场景下的加速存储系统，也已经支持接入 Apache Ranger 的统一鉴权平台中；CHDFS 则是公有云原生的 HDFS 服务，本期重点更新主要集成了 CHDFS 认证和 Ranger 鉴权体系，方便大数据业务尽可能提升业务安全管控能力。

在具体使用过程中，可以通过配置文件方便快捷地将 CHDFS 认证和 Ranger 鉴权体系集成到 GooseFS 中。
步骤如下：

1. 当部署 GooseFS 完成后，在 goosefs-env.sh 中配置 hadoop-ranger-client-for-hadoop-${hadoop.version}-${version}.jar 和 cosn-ranger-interface-${version}.jar 所在的路径：

```
GOOSEFS_CLASSPATH=${GOOSEFS_HOME}/lib/goosefs-underfs-
chdfs-${version}.jar:/path/to/cosn-ranger-
interface-${version}.jar:/path/to/hadoop-ranger-client-for-
hadoop-${hadoop.version}-${version}.jar
```

> **说明**
> 对于 EMR 的环境，可以查看 `/usr/local/service/hadoop/share/hadoop/common/lib` 这个路径是否存在上述两个依赖包，如果存在的话，将这两个包按照上述方法配置到 GooseFS 即可：

```
GOOSEFS_CLASSPATH=${GOOSEFS_HOME}/lib/goosefs-underfs-
chdfs-${version}.jar:/usr/local/service/hadoop/share/hadoop/common/lib/cosn-
ranger-
interface-${version}.jar:/usr/local/service/hadoop/share/hadoop/common/lib/had
oop-ranger-client-for-hadoop-${hadoop.version}-${version}.jar
```

2. 确保在 core-site.xml 配置文件中，已开启 ranger 相关的配置选项：

```
  <property>
 <name>fs.ofs.ranger.enable.flag</name>
 <value>true</value>
  </property>
```

3. 在 goosefs-site.properties 中需要将 core-site.xml 的配置文件路径指定到 goosefs underfs hdfs 的配置路径中，同时开启 security authorization，这样才能保证身份认证信息能够通过 GooseFS 传递到 UFS 层：

```
goosefs.underfs.hdfs.configuration=/usr/local/service/hadoop/etc/hadoop/hdfs-
site.xml:/usr/local/service/hadoop/etc/hadoop/core-site.xml
# Security properties
goosefs.security.authorization.permission.enabled=true
goosefs.security.authentication.type=SIMPLE
```

4. 以上配置至少需要同步到所有的 Master 节点上，然后重启 Master 即可开启 CHDFS 的鉴权。有关 GooseFS Ranger 的详细介绍，可参见 使用 Apache Ranger 控制 GooseFS 的访问权限 。

其他更新点

除了上述更新之外，我们在本次版本中还优化了 GooseFS 依赖的组件：

- 升级了 RocksDB 的依赖版本到6.15.2（从5.15.10 升级到6.15.2）
- 更新了依赖的 Linux/MacOS libjnifuse 的动态链接库

同时，根据生产环境下大规模使用后的反馈，我们也修复了如下问题：

- 修复 Journal 乱序的问题
- Ratis 死锁导致的 GRPC 问题
- 修复了HDFSUnderFileSystemFactory 加载位置不正确的问题
- 修复了 log4j2 的安全漏洞问题
- 修复了ufsPath 前缀检查错误的问题

如果您想了解数据加速器 GooseFS 的更多信息，或者上手使用 GooseFS ，请参见 数据加速器 GooseFS 产品文档。

快速入门
控制台快速入门

最近更新时间：2025-04-14 15:41:02

本文档主要提供 GooseFS 快速部署的相关指引，用户无需编写代码或运行程序，直接在GooseFS 控制台上创建 GooseFS 集群，并将对象存储（Cloud Object Storage，COS）作为远端存储的步骤指引。

准备工作

账号与权限准备

1. 注册腾讯云账号。

   在使用腾讯云数据加速器 GooseFS 服务前，您需要先注册一个腾讯云账号。请点击下方按钮开始注册。（如果您已注册，请跳过该步骤。）

   开始注册

2. 完成实名认证。

   账号注册完成后，使用该账号登录 腾讯云控制台，开始实名认证。详细操作指引请参见 实名认证介绍。（如果您已完成，请跳过该步骤。）

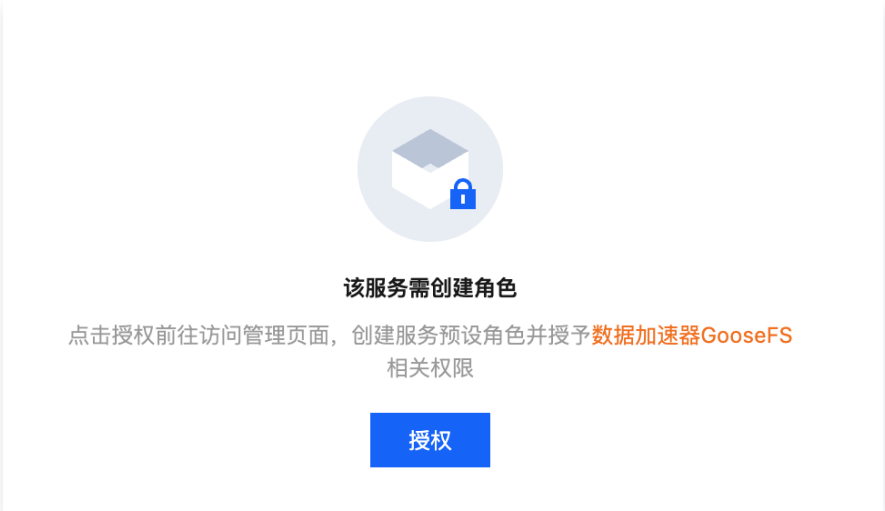
   开始实名认证

3. 开通 GooseFS 服务。

   在登录控制台使用 GooseFS 前，您需要 联系我们 开通 GooseFS 服务。

4. GooseFS 授权。

   开通 GooseFS 服务后需要预设角色并授予数据加速器 GooseFS 相关权限，请单击**授权**按钮进行授权。



   单击**同意授权**，如果您没有授权权限请将该链接复制给主账号或有授权权限的子账号进行角色授权。

授权完成后，即可正常创建 GooseFS 集群。

5. 关联存储桶和节点。

   使用 GooseFS 创建命名空间需要挂载 COS 存储桶，请在 对象存储控制台 创建 COS 存储桶。
   如果您使用管控面托管或 Master 托管模式，需要您自行准备 Master 或 Worker 节点，请在 云服务器控制台 创建云服务器。

集群部署

步骤1：创建GooseFS集群

1. 登录 数据加速器 GooseFS 控制台。在左侧导航栏中选择 **GooseFS > 实例列表**，进入 GooseFS 集群列表页面。
2. 单击**新建**，进入 GooseFS 创建流程。
3. 填写 GooseFS 集群信息，需要配置如下参数项。

- **集群名称**：集群唯一标记，账号下全局唯一，集群名称需要遵循命名规范，可参考 GooseFS 集群概述 中的描述。
- **集群描述**：用于表述集群的属性、用途等信息。
- **集群类别**：集群不同类型或应用场景。分为腾讯公有云集群和 IDC 云集群。
- **地域**：集群所在地域，一般跟计算业务所在集群的域一致。
- **可用区**：集群所在的可用区，一般跟计算集群同 AZ。
- **所属 VPC**：集群所在的 VPC，一般跟计算集群同 VPC，尽量避免跨 VPC 部署的情况。
- **所属子网**：集群所在的子网。
- **集群标签**：集群标签信息。

4. 集群资源用于配置 GooseFS 集群需要管理的缓存存储规模。

   GooseFS 部署模式包括全托管、Master 托管、管控面托管三种模式。
   - 全托管模式下，用户无需自行购买 Master 节点和 Worker 节点，所有节点均由腾讯云托管。腾讯云团队将负责集群的整体可用性和稳定性，确保用户可以专注于核心业务。
   - Master 托管模式下，用户在 GooseFS 平台购买 Master 实例；Worker 节点需要由用户自己购买 CVM 实例 作为节点，在完成集群创建后通过 添加节点 加入集群。
   - 管控面托管模式下，GooseFS 的 Master 和 Worker 节点均需要由用户自己购买 CVM 实例 作为节点。

   下面以全托管集群为例创建集群：
   全托管模式下用户只需要选择需要的**实例容量**（10TB-1000TB，步长10TB）。

5. 按需填入上述各项参数后，单击**下一步**，确认信息无误后，单击**确定**执行。
6. 进入集群创建流程，可在**任务中心**查看集群部署的详细进展。

步骤2：新增节点

了解 GooseFS 节点详情可以查看 GooseFS 集群概述，下面开始介绍新增节点的流程。

1. 选择需要新增节点的 GooseFS 集群，进入集群详情页面，在侧边栏中选择节点管理选项，进入节点管理子页面。
2. 在节点管理页面，单击**新增节点**，填写如下参数。

- **实例类型**：当前仅支持添加 CVM 实例。
- **节点类型**：支持 Worker 节点和 Client 节点，全托管只支持 Client 节点。
- **节点 IP**：选择需要添加的节点。
- **格式化挂载**：用于格式化磁盘并进行文件系统挂载点。用户自选是否进行格式化挂载。如果不启用，那么在创建流程时不需要设置数据盘挂载选项，可手动或者使用脚本挂载；如果启用，那么需要填写设备名称，格式化系统，挂载点等参数。
  - **设备名称**：需要挂载的硬件设备，一般为特定的磁盘分区，需要登录到机器上查看，一般在/dev 路径下。
  - **格式化系统**：Linux 文件系统类型。
  - **挂载点**：文件系统挂载点。

3. 按需填入各项操作所需参数，确认无误后单击**确定**执行，即可新增节点。

步骤3：创建命名空间

GooseFS 通过命名空间组织和管理缓存数据。GooseFS 的命名空间是 GooseFS 集群访问数据的统一访问接口，可以挂载多种底层存储系统，包括对象存储 COS、云 HDFS 等。

1. 选择需要创建命名空间的 GooseFS 集群，进入集群详情页面，在侧边栏中选择**命名空间**选项，进入命名空间子页面。
2. 在命名空间页面，单击**新增命名空间** ，填写如下字段。

- **存储桶来源**：可选择本账号或其他账号的存储桶。注：选择其他账号存储桶请保证该账号有对应存储桶读写权限。
- **COS 存储桶**：选择需要挂载的存储桶。
- **空间名称**：命名空间的名称。
- **挂载范围**：可选择整个存储桶/指定目录前缀。
- **UFS URI**：用于访问和操作UFS文件系统中文件的地址。
- **读策略**：用于管理从GooseFS读数据时，数据在GooseFS集群中的缓存策略。当前只支持 CACHE。
- **写策略**：用于管理通过GooseFS写数据时，数据在GooseFS集群中的缓存策略。当前支持CACHE_THROUGH和THROUGH。
- **UFS 属性**：用于设置访问底层存储的配置信息，如访问密钥等。
  - fs.cosn.bucket.region：所选存储桶地域，如ap-beijing。
  - fs.cosn.flush.enabled：是否启用数据刷新机制，默认为true。
  - fs.cosn.upload.part.size：分块上传的单个分片大小，默认为 8MB。
  - fs.cosn.userinfo.secretId：用于标识 API 调用者身份。一个 APPID 可以创建多个云 API 密钥。
  - fs.cosn.userinfo.secretKey：腾讯云账户的加密签名密钥，用于验证请求合法性。

> ⚠ **注意：SecretId 和 SecretKey 合称为云 API 密钥，是用户访问腾讯云 API 进行身份验证时需要用到的安全凭证， 请在 访问管理 页面创建。在左侧导航栏选择访问密钥 > API 密钥管理，单击新建密钥获取。SecretKey 只能在新建密钥页面获取，请注意保存。**

3. 按需填入各项操作所需参数，确认无误后单击**确定**执行， 即可创建命名空间。

步骤4：添加客户端

新增节点并创建命名空间后，需要给新增的节点添加 GooseFS-FUSE 客户端，访问命名空间。

1. 选择需要添加客户端的 GooseFS 集群，进入集群详情页面，在侧边栏中选择客户端管理选项，进入客户端管理子页面。
2. 在客户端管理页面中，进行**新增客户端**操作，用于在一个指定节点中部署 GooseFS 客户端，可选参数如下。

- **实例类型**：当前仅支持添加 CVM 实例。
- **客户端类型**：当前仅支持 FUSE 类型，有关 GooseFS FUSE 客户端介绍可参考 GooseFS FUSE 工具 。
- **节点 IP**：需要添加客户端的节点。

> **注意：**
> - 管控面托管和 Master 托管模式下，可以将客户端部署在 GooseFS 集群的 Worker 节点和 Client 节点上。

- 全托管模式下，只能将客户端部署在 GooseFS 集群的 Client 节点上。

  - **挂载信息**：用于添加本地文件系统挂载点，添加后用户可以通过客户端访问 GooseFS 上的缓存文件。可选参数如下：
    - **本地挂载目录**：FUSE 客户端本地挂载目录，用于访问 GooseFS 集群上的缓存文件，默认为/data/goosefs。
    - **GooseFS 目录**：GooseFS 文件系统目录，即 GooseFS 命名空间中的路径，例如/0-aaa-int-1250000000/data。若未指定，默认挂载命名空间根路径，例如/0-aaa-int-1250000000。
    - **挂载参数**：FUSE 客户端的挂载参数，默认挂载参数 allow_other 表示允许其他用户访问；direct_io 表示启用直接 I/O。更多参数可参考 FUSE 挂载参数列表 。
  - **批次并发度**：控制同时执行的任务数（默认1），适用于批量批量挂载客户端。
  - **批次失败策略**：任务失败后策略，可选择继续执行或种植后续批次的执行。

3. 按需填入各项操作所需参数，确认无误后单击**确定**执行， 即可添加客户端。

步骤5：挂载验证

现在存储桶已经挂载到了对应的 Client 节点上，登录节点 CVM，进入节点本地挂载目录/data/goosefs，即可查看命名空间关联存储桶中的所有文件。

```
[root@VM-0-15-tencentos ~]# cd /data/goosefs
[root@VM-0-15-tencentos goosefs]# ls
2.txt  4.txt            bj_test  dir_test  emr                                   goosefs-1.1.0-SNAPSHOT-client.jar  LICENSE  nnbench  test0217.txt  testfile    test.txt  user         燃数-ranger升级1.0.5
3.txt  authorizedKeys.sh  bollu    ee        goosefs-1.1.0-SNAPSHOT-client-hadoop.jar  keygen.sh                          new      test01   test3         testfile02  tmp       vdbench_test
[root@VM-0-15-tencentos goosefs]#
```

版权所有：腾讯云计算（北京）有限责任公司

独立部署快速入门

最近更新时间：2024-12-26 19:09:22

本文档主要提供 GooseFS 快速部署、调试的相关指引，提供在本地机器上部署 GooseFS，并将对象存储（Cloud Object Storage，COS）作为远端存储的步骤指引。

前提条件

在使用 GooseFS 之前，您还需要准备以下工作：

1. 在 COS 服务上创建一个存储桶以作为远端存储，操作指引请参见 控制台快速入门。
2. 安装 Java 8 或者更高的版本。
3. 安装 SSH，确保能通过 SSH 连接到 LocalHost，并远程登录。
4. 在 CVM 服务上购买一台实例，操作指引详见 购买云服务器，并确保磁盘已经挂载到实例上。

下载并配置 GooseFS

1. 新建本地目录并进入该目录下(您也可以按需选择其他目录)，从官方仓库下载 GooseFS 安装包到本地。安装包 Github 地址：goosefs-1.4.5-bin.tar.gz 。

```
$ cd /usr/local
$ mkdir /service
$ cd /service
$ wget https://downloads.tencentgoosefs.cn/goosefs/1.4.5/release/goosefs-1.4.5-bin.tar.gz
```

2. 执行如下命令，对安装包进行解压，解压后进入安装包目录下。

```
$ tar -zxvf goosefs-1.4.5-bin.tar.gz
$ cd goosefs-1.4.5
```

解压后，得到 goosefs-1.4.5，即 GooseFS 的主目录。下文将以 `${GOOSEFS_HOME}` 代指该目录的绝对路径。

3. 在 `${GOOSEFS_HOME}/conf` 的目录下，创建 `conf/goosefs-site.properties` 的配置文件。GooseFS 提供 AI 和大数据两个场景的配置模板，可以使用任意上述内置模板，然后进入编辑模式修改配置：
   - 使用 AI 场景模板，更多信息可参考 GooseFS AI 场景生产环境配置实践。

```
$ cp conf/goosefs-site.properties.ai_template conf/goosefs-site.properties
$ vim conf/goosefs-site.properties
```

   - 使用大数据场景模板，更多信息可参考 GooseFS 大数据场景生产环境配置实践。

```
$ cp conf/goosefs-site.properties.bigdata_template conf/goosefs-site.properties
$ vim conf/goosefs-site.properties
```

版权所有：腾讯云计算（北京）有限责任公司

4. 在配置文件 `conf/goosefs-site.properties` 中，调整如下配置项：

- AI 场景模板

```
# Master properties
goosefs.master.rpc.addresses=<master1>:9200,<master2>:9200,<master3>:9200
goosefs.master.embedded.journal.addresses=<master1>:9202,<master2>:9202,
<master3>:9202
goosefs.master.journal.folder=<path>
goosefs.master.metastore.dir=<path>
# 建议与goosefs.master.metastore.dir放在不同的nvme盘
goosefs.master.metastore.block.locations=<path>

# Worker properties
goosefs.master.worker.register.lease.count=4
goosefs.worker.tieredstore.level0.dirs.path=<path>,<path>,<path>
goosefs.worker.tieredstore.level0.dirs.quota=<capacity>,<capacity>,
<capacity>

# Client properties
goosefs.user.file.delete.unchecked=true
goosefs.user.file.passive.cache.enabled=false
goosefs.user.file.master.client.pool.size.max=100
goosefs.client.list.status.executor.pool.size=200

# Common Properties
goosefs.network.ip.address.used=true

# Custom Properties
```

- 大数据场景模板

```
# Master properties
goosefs.master.rpc.addresses=<master1>:9200,<master2>:9200,<master3>:9200
goosefs.master.embedded.journal.addresses=<master1>:9202,<master2>:9202,
<master3>:9202
goosefs.master.journal.folder=<path>

# Zookeeper Connection
#goosefs.zookeeper.enabled=true
#goosefs.zookeeper.address=<zk_quorum_1>:<zk_client_port>,<zk_quorum_2>:
<zk_client_port>,<zk_quorum_3>:<zk_client_port>
#goosefs.master.journal.type=UFS
#goosefs.master.journal.folder=<hdfs_path>

goosefs.master.metastore.dir=<path>
# 建议与goosefs.master.metastore.dir放在不同的nvme盘
```

```
goosefs.master.metastore.block.locations=<path>
goosefs.underfs.hdfs.configuration=${HADOOP_HOME}/etc/hadoop/core-
site.xml:${HADOOP_HOME}/etc/hadoop/hdfs-site.xml

# Worker properties
goosefs.worker.tieredstore.level0.dirs.path=<path>,<path>,<path>
goosefs.worker.tieredstore.level0.dirs.quota=<capacity>,<capacity>,
<capacity>

# Security properties
goosefs.security.authorization.permission.enabled=true
goosefs.security.authentication.type=SIMPLE

# Client properties
goosefs.user.client.transparent_acceleration.scope=GFS
goosefs.user.client.transparent_acceleration.enabled=true

# Custom Properties
```

注意：
配置 `goosefs.worker.tieredstore.level0.dirs.path` 该路径参数前，需要先新建这一路径。

启用 GooseFS

启用 GooseFS 前，需要进入到 GooseFS 目录下执行启动指令：

```
$ cd /usr/local/service/goosefs-1.4.5
$ ./bin/goosefs-start.sh all
```

执行该指令后，可以看到如下页面：

```
Executing the following command on all worker nodes and logging to /usr/local/service/goosefs
rt.sh -a proxy
Waiting for tasks to finish...
All tasks finished
----------------------------------------------
Starting to monitor all remote services.
----------------------------------------------
--- [ OK ] The master service @ VM-0-5-centos is in a healthy state.
--- [ OK ] The job_master service @ VM-0-5-centos is in a healthy state.
--- [ OK ] The worker service @ VM-0-5-centos is in a healthy state.
--- [ OK ] The job_worker service @ VM-0-5-centos is in a healthy state.
--- [ OK ] The proxy service @ VM-0-5-centos is in a healthy state.
[root@VM-0-5-centos goosefs-1.3.0]#
```

该命令执行完毕后，可以访问 `http://localhost:9201` 和 `http://localhost:9204`，分别查看 Master 和 Worker 的运行状态。

使用 GooseFS 挂载 COS（COSN）或腾讯云 HDFS（CHDFS）

如果 GooseFS 需要挂载 COS（COSN）或腾讯云 HDFS（CHDFS）到 GooseFS 的根路径上，则需要先在 `conf/core-site.xml` 配置中指定 COSN 或 CHDFS 的必需配置项，其中包括但不限于：`fs.cosn.impl` 、

`fs.AbstractFileSystem.cosn.impl` 以及 `fs.cosn.userinfo.secretId` 和 `fs.cosn.userinfo.secretKey` 等，如下所示：

```
<!-- COSN related configurations -->
<property>
  <name>fs.cosn.impl</name>
  <value>org.apache.hadoop.fs.CosFileSystem</value>
</property>

<property>
   <name>fs.AbstractFileSystem.cosn.impl</name>
   <value>com.qcloud.cos.goosefs.CosN</value>
</property>

<property>
    <name>fs.cosn.userinfo.secretId</name>
    <value></value>
</property>

<property>
    <name>fs.cosn.userinfo.secretKey</name>
    <value></value>
</property>

<property>
    <name>fs.cosn.bucket.region</name>
    <value></value>
</property>

<!-- CHDFS related configurations -->
<property>
   <name>fs.AbstractFileSystem.ofs.impl</name>
   <value>com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter</value>
</property>

<property>
   <name>fs.ofs.impl</name>
   <value>com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter</value>
</property>
```

```
<property>
    <name>fs.ofs.tmp.cache.dir</name>
    <value>/data/chdfs_tmp_cache</value>
</property>


<!--appId-->
<property>
    <name>fs.ofs.user.appid</name>
    <value>1250000000</value>
</property>
```

说明：

- COSN 的完整配置可参考：Hadoop 工具。
- CHDFS 的完整配置可参考：挂载 CHDFS。

下面将介绍一下如何通过创建 Namespace 来挂载 COS 或 CHDFS 的方法和步骤。

1. 创建一个命名空间 namespace 并挂载 COS：

```
$ goosefs ns create myNamespace cosn://bucketName-1250000000/ \
--secret fs.cosn.userinfo.secretId=AKXXXXXXXXXXX \
--secret fs.cosn.userinfo.secretKey=XXXXXXXXXXX \
--attribute fs.cosn.bucket.region=ap-xxx \
```

注意：

- 创建挂载 COSN 的 namespace 时，必须使用 `--secret` 参数指定访问密钥，并且使用 `--attribute` 指定 Hadoop-COS（COSN）所有必选参数，具体的必选参数可参考 Hadoop 工具。
- 创建 Namespace 时，如果没有指定读写策略（rPolicy/wPolicy），默认会使用配置文件中指定的 read/write type，或使用默认值（CACHE/CACHE_THROUGH）。

同理，也可以创建一个命名空间 namespace 用于挂载腾讯云 HDFS：

```
goosefs ns create MyNamespaceCHDFS ofs://xxxxx-xxxx.chdfs.ap-
guangzhou.myqcloud.com/ \
--attribute fs.ofs.user.appid=1250000000
--attribute fs.ofs.tmp.cache.dir=/tmp/chdfs
```

2. 创建成功后，可以通过 `ls` 命令列出集群中创建的所有 namespace：

```
$ goosefs ns ls
```

```
namespace          mountPoint           ufsPath
creationTime                wPolicy          rPolicy         TTL
ttlAction
myNamespace    /myNamespace   cosn://bucketName-125xxxxxx/3TB  03-11-2021
11:43:06:239      CACHE_THROUGH   CACHE        -1      DELETE
myNamespaceCHDFS /myNamespaceCHDFS ofs://xxxxx-xxxx.chdfs.ap-
guangzhou.myqcloud.com/3TB 03-11-2021 11:45:12:336 CACHE_THROUGH   CACHE  -1
DELETE
```

3. 执行如下命令，指定 namespace 的详细信息。

```
$ goosefs ns stat myNamespace

NamespaceStatus{name=myNamespace, path=/myNamespace, ttlTime=-1,
ttlAction=DELETE, ufsPath=cosn://bucketName-125xxxxxx/3TB,
creationTimeMs=1615434186076, lastModificationTimeMs=1615436308143,
lastAccessTimeMs=1615436308143, persistenceState=PERSISTED, mountPoint=true,
mountId=4948824396519771065, acl=user::rwx,group::rwx,other::rwx, defaultAcl=,
owner=user1, group=user1, mode=511, writePolicy=CACHE_THROUGH,
readPolicy=CACHE}
```

元数据中记录的信息包括如下内容：

序号	参数	描述
1	name	namespace 的名字
2	path	namespace 在 GooseFS 中的路径
3	ttlTime	namespace 下目录和文件的 ttl 周期
4	ttlAction	namespace 下目录和文件的 ttl 处理动作，有两种处理动作：FREE 和 DELETE，默认是 FREE
5	ufsPath	namespace 在 ufs 上的挂载路径
6	creationTimeMs	namespace 的创建时间，单位是毫秒
7	lastModificationTimeMs	namespace 下目录和文件的最后修改时间，单位是毫秒
8	persistenceState	namespace 的持久化状态
9	mountPoint	namespace 是否是一个挂载点，始终为 true
10	mountId	namespace 挂载点 ID

11	acl	namespace 的访问控制列表
12	defaultAcl	namespace 的默认访问控制列表
13	owner	namespace 的 owner
14	group	namespace 的 owner 所属的 group
15	mode	namespace 的 POSIX 权限
16	writePolicy	namespace 的 写策略
17	readPolicy	namespace 的 读策略

使用 GooseFS 预热 Table 中的数据

1. GooseFS 支持将 Hive Table 中的数据预热到 GooseFS 中，在预热之前需要先将相关的 DB 关联到 GooseFS 上，相关命令如下：

```
$ goosefs table attachdb --db test_db hive thrift://
172.16.16.22:7004 test_for_demo
```

> **注意：**
> 命令中的 thrift 需要填写实际的 Hive Metastore 的地址。

2. 添加完 DB 后，可以通过 ls 命令查看当前关联的 DB 和 Table 的信息：

```
$ goosefs table ls test_db web_page

OWNER: hadoop
DBNAME.TABLENAME: testdb.web_page (
wp_web_page_sk bigint,
wp_web_page_id string,
wp_rec_start_date string,
wp_rec_end_date string,
wp_creation_date_sk bigint,
wp_access_date_sk bigint,
wp_autogen_flag string,
wp_customer_sk bigint,
wp_url string,
wp_type string,
wp_char_count int,
wp_link_count int,
wp_image_count int,
wp_max_ad_count int,
)
PARTITIONED BY (
```

```
)
LOCATION (
gfs://172.16.16.22:9200/myNamespace/3000/web_page
)
PARTITION LIST (
{
partitionName: web_page
location: gfs://172.16.16.22:9200/myNamespace/3000/web_page
}
)
```

3. 通过 load 命令预热 Table 中的数据：

```
$ goosefs table load test_db web_page
Asynchronous job submitted successfully, jobId: 1615966078836
```

预热 Table 中的数据是一个异步任务，因此会返回一个任务 ID。可以通过 `goosefs job stat <Job Id>` 命令查看预热作业的执行进度。当状态为 "COMPLETED" 后，则整个预热过程完成。

使用 GooseFS 进行文件上传和下载操作

1. GooseFS 支持绝大部分文件系统操作命令，可以通过以下命令来查询当前支持的命令列表：

```
$ goosefs fs
```

2. 可以通过 `ls` 命令列出 GooseFS 中的文件，以下示例展示如何列出根目录下的所有文件：

```
$ goosefs fs ls /
```

3. 可以通过 `copyFromLocal` 命令将数据从本地拷贝到 GooseFS 中：

```
$ goosefs fs copyFromLocal LICENSE /LICENSE
Copied LICENSE to /LICENSE
$ goosefs fs ls /LICENSE
-rw-r--r--  hadoop          supergroup                20798       NOT_PERSISTED
03-26-2021 16:49:37:215   0% /LICENSE
```

4. 可以通过 `cat` 命令查看文件内容：

```
$ goosefs fs cat /LICENSE
Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
```

```
...
```

5. GooseFS 默认使用本地磁盘作为底层文件系统，默认文件系统路径为 `./underFSStorage`，可以通过 `persist` 命令将文件持久化存储到本地文件系统中：

```
$ goosefs fs persist /LICENSE
persisted file /LICENSE with size 26847
```

使用 GooseFS 加速文件上传和下载操作

1. 检查文件存储状态，确认文件是否已被缓存。文件状态 `PERSISTED` 代表文件已在内存中，文件状态 `NOT_PERSISTED` 则代表文件不在内存中：

```
$ goosefs fs ls /data/cos/sample_tweets_150m.csv
-r-x------ staff  staff 157046046 NOT_PERSISTED 01-09-2018 16:35:01:002   0%
/data/cos/sample_tweets_150m.csv
```

2. 统计文件中有多少单词 “tencent”，并计算操作耗时：

```
$ time goosefs fs cat /data/s3/sample_tweets_150m.csv | grep-c tencent
889
real    0m22.857s
user    0m7.557s
sys     0m1.181s
```

3. 将该数据缓存到内存中可以有效提升查询速度，详细示例如下：

```
$ goosefs fs ls /data/cos/sample_tweets_150m.csv
-r-x------ staff  staff 157046046
ED 01-09-2018 16:35:01:002   0% /data/cos/sample_tweets_150m.csv
$ time goosefs fs cat /data/s3/sample_tweets_150m.csv | grep-c tencent
889
real    0m1.917s
user    0m2.306s
sys     0m0.243s
```

可见，系统处理延迟从1.181s减少到了0.243s，得到了10倍的提升。

关闭 GooseFS

通过如下命令可以关闭 GooseFS：

```
$ ./bin/goosefs-stop.sh local
```

版权所有：腾讯云计算（北京）有限责任公司

控制台指南
GooseFS 集群概述

最近更新时间：2025-04-11 18:41:42

简介

GooseFS 是腾讯云推出的多协议、高性能、大吞吐的数据缓存加速服务。GooseFS 集群是该服务在公有云上的一种部署和运维形态。通过 GooseFS 集群，用户可以方便地管理和运维计算节点的本地盘作为缓存节点，为上层计算应用提供统一的命名空间和访问协议，方便用户在不同的存储系统管理和流转数据。

GooseFS 集群不仅可以加速海量数据分析、机器学习、人工智能等业务访问存储的性能，还可以帮助业务实现冷热数据自动分层，平衡业务架构的性能表现和成本支出。

基本概念

使用 GooseFS 集群，会涉及到以下基本概念：

- 集群：指数据湖存储 GooseFS 运行所需云资源的集合，包含了云服务器、裸金属设备等云资源。
- 节点：指组成 GooseFS 集群的各类云服务器资源实例，主要包括如下几类节点：
  - Master 节点：Master 节点是集群主控节点，至少需要使用1台实例部署；高可用模式下，至少需要3台以上实例部署。
  - Worker 节点：Worker 节点负责缓存热数据，至少需要使用1台实例作为 Worker 节点；Worker 节点数量可以跟随需要缓存的存储容量线性扩展。
  - Client 节点：Client 节点一般和计算或者 Worker 节点同机部署，用于读写 Worker 节点中的数据。
- 客户端：指访问 GooseFS 集群的终端，用于和计算应用对接，当前仅支持 GooseFS-FUSE 客户端。
- 命名空间：指 GooseFS 组织和管理缓存数据的空间，是 GooseFS 集群访问数据的统一访问接口。
- 部署形态：指数据湖存储 GooseFS 的部署模式，主要分为管控面托管、Master 托管模式。
  - 管控面托管：使用客户已购买的 CVM 资源部署 GooseFS，并交由 GooseFS 云平台代运维。
  - Master 托管：GooseFS 集群的 Master 节点托管在 GooseFS 云平台，Worker 节点仍由客户自行购买 CVM 完成部署。
  - 全托管：GooseFS 集群的 Master 节点和 Worker 节点都托管在 GooseFS 云平台，无需客户自行运维。
- 配置：指运行 GooseFS 集群的各类配置文件，主要包括：
  - `goosefs-site.properties`：GooseFS 服务配置文件，用于管理 GooseFS 集群的数据加载、缓存、淘汰策略等配置。
  - `goosefs-env.sh`：GooseFS 服务依赖环境变量脚本文件，用于管理 GooseFS 集群的各类环境变量。
  - `log4j.properties`：GooseFS 服务日志输出配置，用于管理 GooseFS 运行过程中的访问日志、系统日志等各类日志、监控信息。
  - `core-site.xml`：GooseFS 服务对接的底层存储系统配置，用于管理 GooseFS 集群对接的底层存储系统，例如对象存储、云 HDFS 等。

使用流程

版权所有：腾讯云计算（北京）有限责任公司

GooseFS 集群使用流程主要分为如下几个环节：

1. 角色授权：注册并登录 数据加速器 GooseFS 控制台，完成服务授权获取相关资源操作权限，即可开始使用数据加速器 GooseFS 产品。
2. 创建集群：按需自定义新建集群。
3. 管理集群：集群新建后，可对集群进行扩缩容、节点管理、配置变更、命名空间管理、客户端管理等操作。
4. 管理任务：集群管理操作会注册为集群任务，用户可通过查看任务进展了解集群操作状态。

产品定价

腾讯云数据湖存储 GooseFS 针对不同部署形态提供了不同的计费体系。如需了解该产品详细的收费模式和具体价格，可 提交工单 咨询。

相关服务

- GooseFS 只提供数据缓存能力，持久化数据存储可使用腾讯云对象存储服务。对象存储的使用说明请参见 对象存储产品文档。
- GooseFS 支持管理云服务器自带的本地盘或者云盘，用于将数据缓存到本地节点。云服务器的使用说明请参见 云服务器产品文档。
- GooseFS 集群可以建立在私有网络下，集群内主机可以分配在不同可用区的子网下。私有网络产品使用指南请参见 私有网络产品文档。
- GooseFS 集群支持将监控数据上报到腾讯云可观测平台或者日志服务 CLS 中。腾讯云可观测平台产品使用指南请参见 腾讯云可观测平台产品文档，日志服务 CLS 产品使用指南请参见 日志服务产品文档。

创建 GooseFS 集群

最近更新时间：2025-04-14 15:41:02

简介

您可以通过 GooseFS 控制台，在 GooseFS 列表页面创建 GooseFS 集群。关于 GooseFS 集群的概念，请参见 GooseFS 集群概述 。下面将为您详细介绍如何创建 GooseFS 集群。

> **注意：**
> GooseFS 集群具有地域概念，一般情况下，推荐您优先选择在您的计算业务所在的地域部署。例如，大数据分析作业在广州六区，那么 GooseFS 集群的部署地域也推荐在广州六区，以减少数据 IO 路径。

前提条件

在登录控制台使用 GooseFS 前，您需要 联系我们 开通 GooseFS 服务。

操作步骤

1. 登录 数据加速器 GooseFS 控制台 。
2. 在左侧导航中，选择 **GooseFS** > **实例列表**，进入 GooseFS 集群列表页面。
3. 单击**新建**，进入 GooseFS 集群创建流程。GooseFS 集群的新建流程主要分为配置集群信息、配置集群资源和确认信息三个主要步骤。
4. 集群信息主要用于描述 GooseFS 集群的基本属性，需要配置如下参数项。

- **集群名称**：集群唯一标记，账号下全局唯一，集群名称需要遵循命名规范，可参考 GooseFS 集群概述中的描述。
- **集群描述**：用于表述集群的属性、用途等信息。

- **集群类别**：集群不同类型或应用场景。分为腾讯公有云集群和 IDC 云集群。
- **地域**：集群所在地域，一般跟计算业务所在集群的地域一致。
- **可用区**：集群所在的可用区，一般跟计算集群同 AZ。
- **所属 VPC**：集群所在的 VPC，一般跟计算集群同 VPC，尽量避免跨 VPC 部署的情况。
- **所属子网**：集群所在的子网。
- **标签**：集群标签信息。

5. 集群资源主要用于配置 GooseFS 集群需要管理的缓存存储规模。GooseFS 部署模式包括全托管、Master 托管和管控面托管三种模式，以下分别展开三种模式的集群创建说明。

全托管

全托管模式下，用户无需自行购买 Master 节点和 Worker 节点，所有节点均由腾讯云托管。腾讯云团队将负责集群的整体可用性和稳定性，确保用户可以专注于核心业务。

全托管模式用户只需选择**实例容量**（10TB-1000TB，步长10TB）即可。

Master 托管

Master 托管模式下，用户在 GooseFS 平台购买 Master 实例；Worker 节点需要由用户自己购买 CVM 实例 作为节点，在完成集群创建后通过 添加节点 加入集群。在创建 Master托管模式的 GooseFS 集群时，相关参数说明如下：

- **部署模式**：选择 Master 托管。
- **Master 规格**：可以选择 Medium 版、Large 版、XLarge 版。
  不同 Master 机型的规格如下：

机型规格	文件数
Medium	1亿
Large	50亿 – 70亿
XLarge	100亿 – 120亿

- **实例类型**：选择 CVM/EMR。
- **缓存配置**：
  - **缓存路径**：GooseFS 集群 Worker 节点数据在宿主机上的存储路径。
  - **缓存容量**：集群单个 Worker 节点数据容量大小。

管控面托管

管控面托管模式下，GooseFS 的 Master 和 Worker 节点均需要由用户自己购买 CVM 实例作为节点。在创建管控面托管模式的 GooseFS 集群时，相关参数说明如下：

- **部署模式**：选择管控面托管。
- **类型**：当前支持 CVM/EMR。
- **实例版本**：GooseFS 版本号信息。
- **高可用模式**：若启用高可用模式，则至少需要部署3个 Master 节点。
- **Master 节点**：选择 Master 节点。若未启用高可用模式，则选择1个 Master 节点即可。
- **Worker 节点**：选择 Worker 节点。

- **Master 格式化挂载**：用于格式化磁盘并进行文件系统挂载点。用户自选是否进行格式化挂载。若未启用，则在创建流程时不需要设置数据盘挂载选项，可手动或者使用脚本挂载；若启用，则需要填写设备名称，格式化系统，挂载点等参数。
  - **挂载设备名称**：需要挂载的硬件设备，一般为特定的磁盘分区，需要登录到机器上查看，一般在 `/dev` 路径下。
  - **格式化系统**：Linux 文件系统类型。
  - **挂载点**：文件系统挂载点。

  > **示例：**
  > 如果您需要将 `/dev/vdb` 这块设备格式化成 `ext4`，并挂载到 `/var/lib/docker` 目录下，可以这样设置：设备名称 `/dev/vdb`，格式化系统 `ext4`，挂载点 `/var/lib/docker`。

- **Worker 格式化挂载**：配置说明同上述 Master 格式化挂载。
- **缓存配置**：主要用于配置Worker 节点。
  - 缓存路径：GooseFS 集群 Worker 节点数据在宿主机上的存储路径。
  - 缓存存储容量：GooseFS 集群单个 Worker 节点数据容量大小。
- **挂载点配置**：主要用于设置挂载点的详细配置信息，最关键的是单盘缓存容量。
  - Journal 路径：GooseFS 集群的 Journal 日志在宿主机上的存储路径。
  - Block 路径： GooseFS 集群的 Block 元数据在宿主上的存储路径。
  - Metastore 路径：GooseFS 集群的元数据在宿主机上的存储路径。

6. 按需填入上述各项参数后，单击**下一步**，将开始检查节点状态。系统检查无误后，单击**确定**执行。
7. 进入集群创建流程，可在**任务中心**中查看集群部署的详细进展。

后续步骤

创建好 GooseFS 集群后，可以根据业务需要登录集群进行下一步的节点管理、命名空间管理、客户端管理、配置管理等操作。具体操作可以参考下述文档：

- 节点管理
- 命名空间管理
- 客户端管理
- 配置管理

管理 GooseFS 集群
节点管理

最近更新时间：2025-04-14 15:41:02

简介

您可以通过 GooseFS 控制台，在 GooseFS 集群详情页面管理 GooseFS 集群节点。关于 GooseFS 集群节点的概念，请参见 GooseFS 集群概述 。下面将为您详细介绍如何管理 GooseFS 集群节点。

注意：

- GooseFS 集群中的节点需要保证在同一个可用区中，尽可能保证在同一个 VPC 下，避免流量穿梭导致数据请求延迟较长的问题。
- GooseFS 集群中包括 Master 节点，Worker 节点，Client 节点三大类节点，不同节点的作用如下：
  - Master 节点：Master 节点是集群主控节点，至少需要使用1台实例部署；高可用模式下，至少需要3台以上实例部署。
  - Worker 节点：Worker 节点负责缓存热数据，至少需要使用1台实例作为 Worker 节点；Worker 节点数量可以跟随需要缓存的存储容量线性扩展。
  - Client 节点：Client 节点用于读写 Worker 节点中的数据。

操作步骤

1. 登录 数据加速器 GooseFS 控制台 。
2. 在左侧导航中，选择 **GooseFS** > **实例列表**，进入 GooseFS 集群列表页面。
3. 选择需要管理的 GooseFS 集群，进入集群详情页面，在侧边栏中单击**节点管理**，进入节点管理子页面。
4. 在节点管理页面中，可以进行如下操作。

- 新增节点：用于往 GooseFS 集群中新增一个节点，可选参数如下：
  - **实例类型**：当前仅支持添加 CVM 实例。
  - **节点类型**：支持 Worker 节点和 Client 节点，全托管模式只支持 Client 节点。
  - **节点 IP**：需要添加的节点。
  - **关联配置组**：节点需要关联的配置组
  - **格式化挂载**：用于格式化磁盘并进行文件系统挂载点。用户自选是否进行格式化挂载。如果不启用，那么在创建流程时不需要设置数据盘挂载选项，可手动或者使用脚本挂载；如果启用，那么需要填写设备名称，格式化系统，挂载点等参数。
    - 设备名称：需要挂载的硬件设备，一般为特定的磁盘分区，需要登录到机器上查看，一般在 `/dev` 路径下。
    - 格式化系统：Linux 文件系统类型。
    - 挂载点：文件系统挂载点。
- 删除节点：用于删除 GooseFS 集群中的指定节点。

> **注意：**
> - 删除节点前必须先停用节点上的 GooseFS 服务，避免影响业务。
> - 删除集群中的最后一个 Master 节点等效于删除集群，如需删除请直接参考 删除 GooseFS 集群 。

5. 按需填入各项操作所需参数，确认无误后单击**确定**执行。
6. 进入节点管理流程，可以在**任务中心**详情中查看各类节点操作的详细进展。

后续步骤

在 GooseFS 集群中创建好节点后，可以根据业务需要添加命名空间、管理节点配置、在节点上安装客户端等操作。具体操作请参见下述文档：

- 配置管理
- 客户端管理
- 命名空间管理

网络配置管理

最近更新时间：2025-04-14 15:41:02

简介

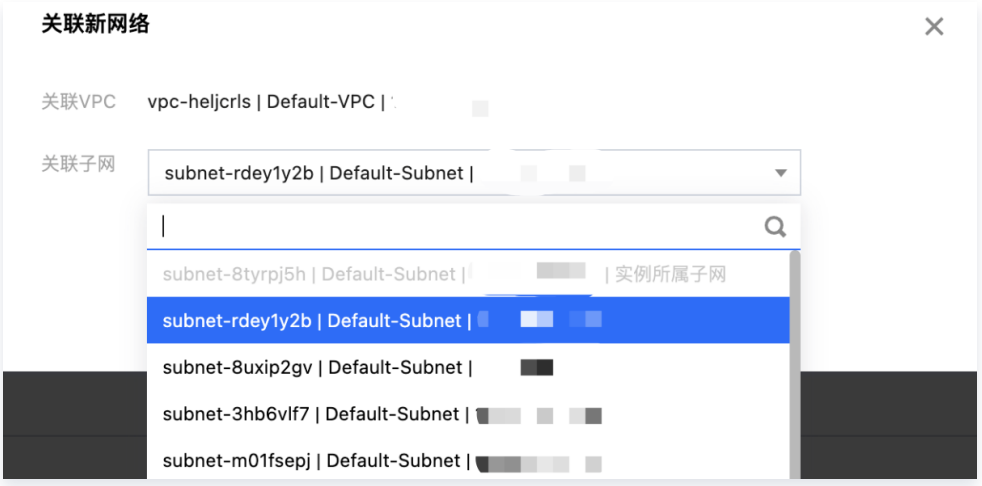
GooseFS 在集群创建后，支持关联集群 VPC 下的其余子网，下面将介绍关联新子网的流程。

操作步骤

1. 登录 数据加速器 GooseFS 控制台 。
2. 在左侧导航中，选择 **GooseFS** > **实例列表**，进入 GooseFS 集群列表页面。
3. 选择需要管理的 GooseFS 集群，在侧边栏中选择**基本信息**页面。
4. 在**基本信息** > **网络配置**模块中，默认网络配置为创建集群时关联的 VPC 和子网。

网络配置模块支持进行新增子网和解除关联操作。

- 新增子网：单击**关联新网络**，选择**关联子网**，关联默认 VPC 下的其他子网。



关联新网络后可在网络配置列表中查看到新关联的子网

- 解除关联：单击子网列表**操作**中的**解除关联**，跳转到解除关联后单击**确定**即可解除子网关联。

> **注意：**
> 创建集群时选择的关联子网为默认网络配置，不可解除。

命名空间管理

最近更新时间：2025-04-14 15:41:02

简介

GooseFS 通过命名空间组织和管理缓存数据。GooseFS 的命名空间是 GooseFS 集群访问数据的统一访问接口，可以挂载多种底层存储系统，包括对象存储 COS、云 HDFS 等。

操作步骤

通过命名空间管理支持创建、更新、删除命名空间，为 GooseFS 集群关联 COS 存储桶并设置缓存读写策略。

1. 登录 数据加速器 GooseFS 控制台 。
2. 在左侧导航中，选择 **GooseFS** > **实例列表**，进入 GooseFS 集群列表页面。
3. 选择需要创建命名空间的 GooseFS 集群，进入集群详情页面，在侧边栏中单击**命名空间**，进入命名空间子页面。
4. 在命名空间页面，单击**新增命名空间** ，填写如下字段。

- **存储桶来源**：可选择本账号或其他账号的存储桶。

  > **注意：**
  > 选择其他账号存储桶请保证该账号有对应存储桶读写权限。

- **COS 存储桶**：选择需要挂载的存储桶。
- **空间名称**：命名空间的名称。
- **挂载范围**：可选择整个存储桶/指定目录前缀。

- **UFS URI**：用于访问和操作UFS文件系统中文件的地址。
- **读策略**：用于管理从 GooseFS 读数据时，数据在 GooseFS 集群中的缓存策略。当前只支持 CACHE。
- **写策略**：用于管理通过 GooseFS 写数据时，数据在 GooseFS 集群中的缓存策略。当前支持 CACHE_THROUGH 和 THROUGH。
- **UFS 属性**：用于设置访问底层存储的配置信息，如访问密钥等。
  - fs.cosn.bucket.region：所选存储桶地域，如 ap-beijing。
  - fs.cosn.flush.enabled：是否启用 数据刷新机制，默认为 true。
  - fs.cosn.upload.part.size：分块上传的单个分片大小，默认为8MB。
  - fs.cosn.userinfo.secretId：用于标识 API 调用者身份。一个 APPID 可以创建多个云 API 密钥。
  - fs.cosn.userinfo.secretKey：腾讯云账户的加密签名密钥，用于验证请求合法性。

> **说明：**
> SecretId 和 SecretKey 合称为云 API 密钥，是用户访问腾讯云 API 进行身份验证时需要用到的安全凭证， 请在 访问管理 页面创建。在左侧导航栏选择**访问密钥** > **API 密钥**管理，单击**新建密钥**获取。

5. 按需填入各项操作所需参数，确认无误后单击**确定**执行， 即可创建命名空间。
6. 命名空间管理。

   进入命名空间页面，在对应命名空间的操作项中进行管理，支持**更新**或**删除**。

- 更新操作：支持修改**写策略**和 **UFS 属性**，命名空间名称创建后无法修改。

- 删除操作：支持删除当前命名空间。

后续步骤

创建命名空间后，您可以通过新增客户端在您的节点上管理命名空间，详细请参见 客户端管理 。

服务管理

最近更新时间：2024-10-30 11:43:43

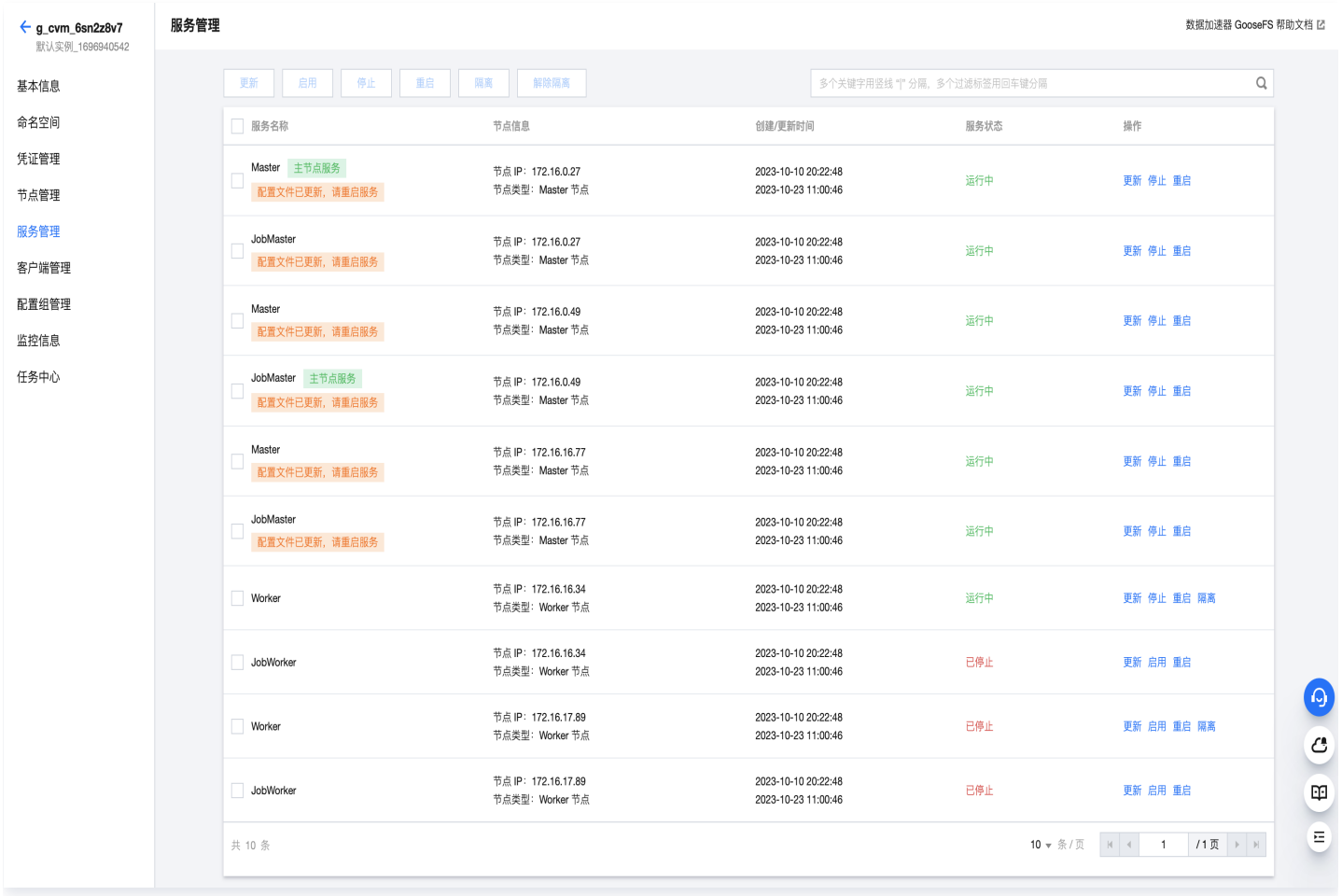
服务管理能力为运维 GooseFS 集群提供了可视化界面，支持 GooseFS 集群中各服务的更新、启用和停止、隔离和解除隔离等运维操作。 GooseFS 集群目前支持的服务有以下几大类：

1. Master 服务：Master 服务主要用于提供元数据管理的能力。用户读写文件时，可以通过 Master 服务获取文件的元数据信息，用于访问文件的数据块。
2. Worker 服务：Worker 服务主要用于存储 Block 数据。用户获取文件元数据信息后，会根据元数据中披露的 Block 数据信息到对应的 Worker 服务节点获取数据。
3. Fuse 服务：Fuse 服务为业务提供文件系统客户端。业务可以通过 GooseFS Fuse 客户端，以文件协议访问 GooseFS 上的文件。
4. Job Master 服务：Job Master 服务用于管控 GooseFS 集群上的任务，如元数据加载（LoadMetadata），预热（DistributedLoad），元数据淘汰（TTL）。
5. Job Service 服务：Job Worker 服务用于执行 GooseFS 集群上的任务，受到 Job Master 服务管控。

操作说明

1. 登录 数据湖存储 GooseFS 控制台 。
2. 在左侧导航中，单击 GooseFS 集群列表，进入 GooseFS 集群列表页面。
3. 选择需要管理的 GooseFS 集群，进入集群详情页面，在侧边栏中选择服务管理选项，进入服务管理子页面。
4. 在服务管理页面中，可以进行如下操作：
   - 更新服务：更新服务上的保活状态，可用于启停节点上的服务保活能力；保活能力开启时，如果节点异常会尝试自动拉起服务。
   - 启用服务：启用节点上的服务，异常节点或者暂停服务节点上的服务可通过该操作重新拉起。
   - 停止服务：可用于暂停节点上的服务，用于临时排障用。
   - 重启服务：可用于重启节点上的服务，用于生效配置。
   - 隔离服务：可用于将异常节点隔离出集群，例如 Worker 服务异常时将节点隔离出集群，避免流量路由到异常节点上。
   - 解除隔离：可用于将已恢复健康状态的节点上的服务重新加入集群。

版权所有：腾讯云计算（北京）有限责任公司


g_cvm_6sn2z8v7
默认实例_1696940542
基本信息
命名空间
凭证管理
节点管理
服务管理
客户端管理
配置组管理
监控信息
任务中心
服务管理
数据加速器 GooseFS 帮助文档
更新
启用
停止
重启
隔离
解除隔离
多个关键字用竖线 "|" 分隔，多个过滤标签用回车键分隔
服务名称
节点信息
创建/更新时间
服务状态
操作
Master 主节点服务
配置文件已更新，请重启服务
节点 IP：172.16.0.27
节点类型：Master 节点
2023-10-10 20:22:48
2023-10-23 11:00:46
运行中
更新 停止 重启
JobMaster
配置文件已更新，请重启服务
节点 IP：172.16.0.27
节点类型：Master 节点
2023-10-10 20:22:48
2023-10-23 11:00:46
运行中
更新 停止 重启
Master
配置文件已更新，请重启服务
节点 IP：172.16.0.49
节点类型：Master 节点
2023-10-10 20:22:48
2023-10-23 11:00:46
运行中
更新 停止 重启
JobMaster 主节点服务
配置文件已更新，请重启服务
节点 IP：172.16.0.49
节点类型：Master 节点
2023-10-10 20:22:48
2023-10-23 11:00:46
运行中
更新 停止 重启
Master
配置文件已更新，请重启服务
节点 IP：172.16.16.77
节点类型：Master 节点
2023-10-10 20:22:48
2023-10-23 11:00:46
运行中
更新 停止 重启
JobMaster
配置文件已更新，请重启服务
节点 IP：172.16.16.77
节点类型：Master 节点
2023-10-10 20:22:48
2023-10-23 11:00:46
运行中
更新 停止 重启
Worker
节点 IP：172.16.16.34
节点类型：Worker 节点
2023-10-10 20:22:48
2023-10-23 11:00:46
运行中
更新 停止 重启 隔离
JobWorker
节点 IP：172.16.16.34
节点类型：Worker 节点
2023-10-10 20:22:48
2023-10-23 11:00:46
已停止
更新 启用 重启
Worker
节点 IP：172.16.17.89
节点类型：Worker 节点
2023-10-10 20:22:48
2023-10-23 11:00:46
已停止
更新 启用 重启 隔离
JobWorker
节点 IP：172.16.17.89
节点类型：Worker 节点
2023-10-10 20:22:48
2023-10-23 11:00:46
已停止
更新 启用 重启
共 10 条
10 条 / 页
1
/ 1 页

配置管理

最近更新时间：2024-10-12 11:37:31

简介

您可以通过 GooseFS 控制台，在 GooseFS 集群详情页中管理 GooseFS 配置。有关 GooseFS 集群配置实践，可以参见 GooseFS 集群配置实践。下面为您详细介绍如何管理 GooseFS 集群配置。

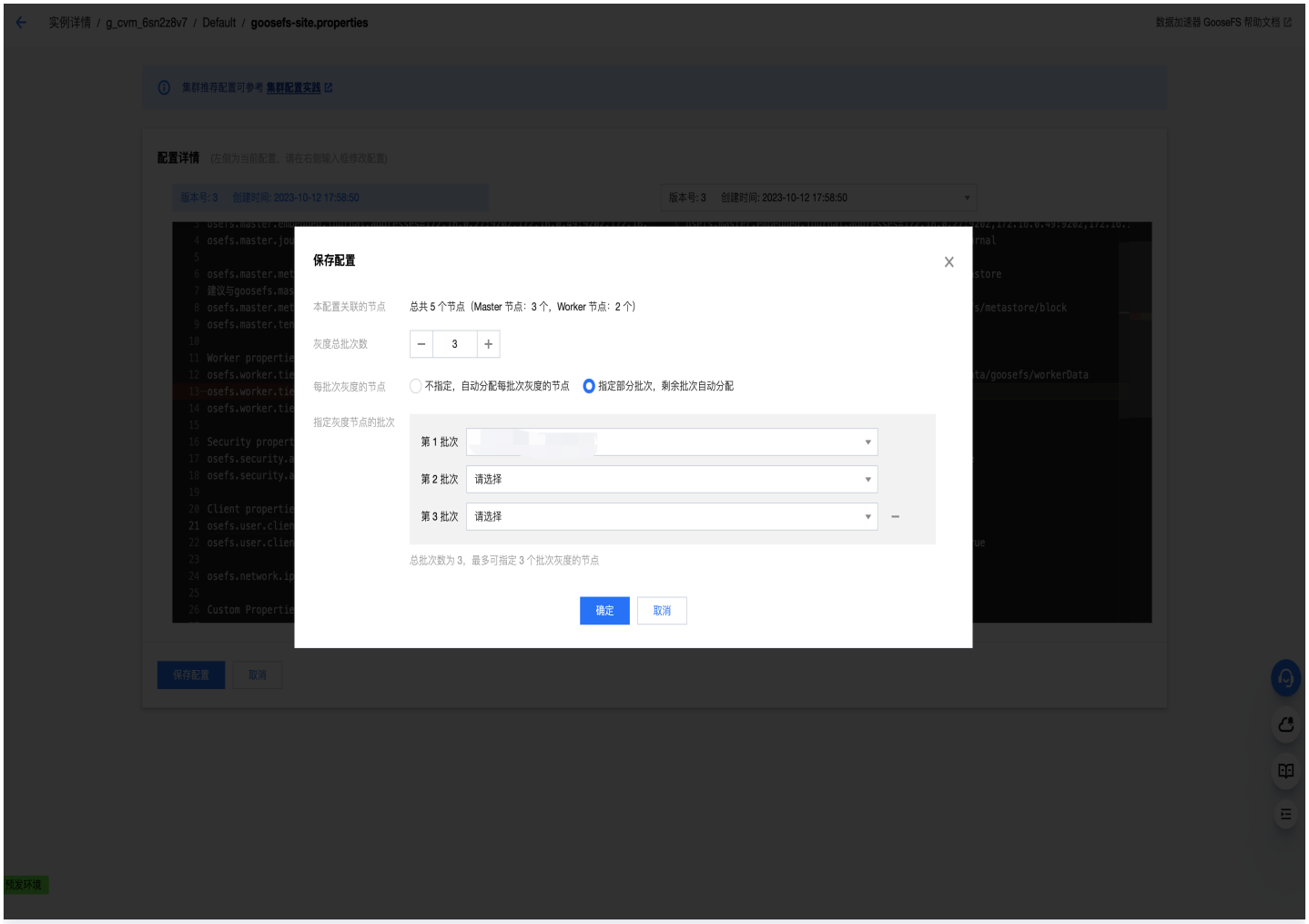
操作步骤

1. 登录 数据湖存储 GooseFS 控制台。
2. 在左侧导航中，单击 GooseFS 集群列表，进入 GooseFS 集群列表页面。
3. 选择需要管理的 GooseFS 集群，进入集群详情页面，在侧边栏中选择**配置管理**选项，进入配置管理子页面。
4. GooseFS 支持4种配置文件修改，单击修改配置后即可进入配置修改页面进行编辑，编辑后单击保存即可完成配置下发，配置项说明如下：
   - `goosefs-site.properties`：GooseFS 服务配置文件，用于管理 GooseFS 集群的数据加载、缓存、淘汰策略等配置。
   - `goosefs-env.sh`：GooseFS 服务依赖环境变量脚本文件，用于管理 GooseFS 集群的各类环境变量。
   - `log4j.properties`：GooseFS 服务日志输出配置，用于管理 GooseFS 运行过程中的访问日志、系统日志等各类日志、监控信息。
   - `core-site.xml`：GooseFS 服务对接的底层存储系统配置，用于管理 GooseFS 集群对接的底层存储系统，例如对象存储、云 HDFS等。
5. 单击保存后，配置会下发到集群中所有节点，下发配置完成后需要重启服务才可生效。

> ⚠ **注意：**
> - 配置下发后会覆盖已有的历史配置，历史配置覆盖后暂不支持回滚，请谨慎操作。
> - 配置下发后需要在节点管理页面重启服务，观察重启服务后配置是否可以生效，如未生效可尝试重新下发配置。

6. 线上业务发布配置时，可能需要逐个节点灰度配置，以观察下发的配置是否符合业务预期。GooseFS 提供了配置灰度的能力，在保存配置的过程中，可以设置灰度批次，并设置每一批次可灰度的节点。

版权所有：腾讯云计算（北京）有限责任公司


实例详情 / g_cvm_6sn2z8v7 / Default / goosefs-site.properties
数据加速器 GooseFS 帮助文档
集群推荐配置可参考 集群配置实践
配置详情 (左侧为当前配置，请在右侧输入框修改配置)
版本号:3 创建时间: 2023-10-12 17:58:50
保存配置
本配置关联的节点
总共 5 个节点（Master 节点：3 个，Worker 节点：2 个）
灰度总批次数
3
每批次灰度的节点
不指定，自动分配每批次灰度的节点
指定部分批次，剩余批次自动分配
指定灰度节点的批次
第 1 批次
第 2 批次
请选择
第 3 批次
请选择
总批次数为 3，最多可指定 3 个批次灰度的节点
确定
取消
保存配置
取消

客户端管理

最近更新时间：2025-04-14 15:41:02

简介

您可以通过 GooseFS 控制台，在 GooseFS 集群详情页面管理 GooseFS 客户端。关于 GooseFS 客户端的概念，请参见 GooseFS 集群概述 。下面将为您详细介绍如何管理 GooseFS 集群的客户端。

操作步骤

1. 登录 数据加速器 GooseFS 控制台 。
2. 在左侧导航中，选择 **GooseFS** > **实例列表**，进入 GooseFS 集群列表页面。
3. 单击需要管理的 GooseFS 集群 **ID/名称**，进入集群详情页面，在侧边栏中单击**客户端管理**，进入客户端管理子页面。
4. 在客户端管理页面中，可以进行如下操作。

- 新增客户端：用于在一个指定节点中部署 GooseFS 客户端，可以配置如下参数，配置完成后单击**确定**即可。
  - **节点 IP**：需要新增客户端的节点。
  - **挂载信息**：
    - **本地挂载目录**：FUSE 客户端本地挂载目录，用于访问 GooseFS 集群上的缓存文件，默认为/data/goosefs。
    - **GooseFS 目录**：GooseFS 文件系统目录，即 GooseFS 命名空间中的路径，例如/0-aaa-int-1250000000/data。若未指定，默认挂载命名空间根路径，例如/0-aaa-int-1250000000。
    - **挂载参数**：FUSE 客户端的挂载参数，默认挂载参数 allow_other 表示允许其他用户访问；direct_io 表示启用直接 I/O。更多参数可参考 FUSE 挂载参数列表 。
  - **批次并发度**：控制同时执行的任务数（默认 1），适用于批量挂载客户端。
  - **批次失败策略**：任务失败后策略，可选择继续执行或终止后续批次的执行。

- 更新客户端：在客户端管理页面，选择需要操作的客户端，在**操作**列单击**更新**，跳转到弹窗中单击**确定**。用于启用服务进程保活，节点异常重启时自动拉起服务，GooseFS 集群在节点异常重启时会自动拉起客户端，保证客户端的可用性。
- 重启客户端：在客户端管理页面，选择需要操作的客户端，在**操作**列单击**重启**，跳转到弹窗中单击**确定**。用于客户端异常后重启，可以支持修改挂载用户、JVM 参数和挂载参数。
- 卸载客户端：在客户端管理页面，选择需要操作的客户端，在**操作**列单击**卸载**，跳转到弹窗中单击**确定**。用于卸载本地文件系统挂载点，卸载后用户无法再通过客户端访问 GooseFS 上的缓存文件。

  > **注意：**
  > 卸载客户端不影响 GooseFS 上已经存在的缓存文件，但如果卸载客户端时有文件正在写入到 GooseFS 中，有可能导致文件写入失败。

- 复制客户端：在客户端管理页面，选择需要操作的客户端，在**操作**列单击**复制**，跳转到弹窗中选择需要新建客户端的**节点 IP** 后单击**确定**。复制当前客户端的配置信息，用于在新节点上创建客户端。

任务管理

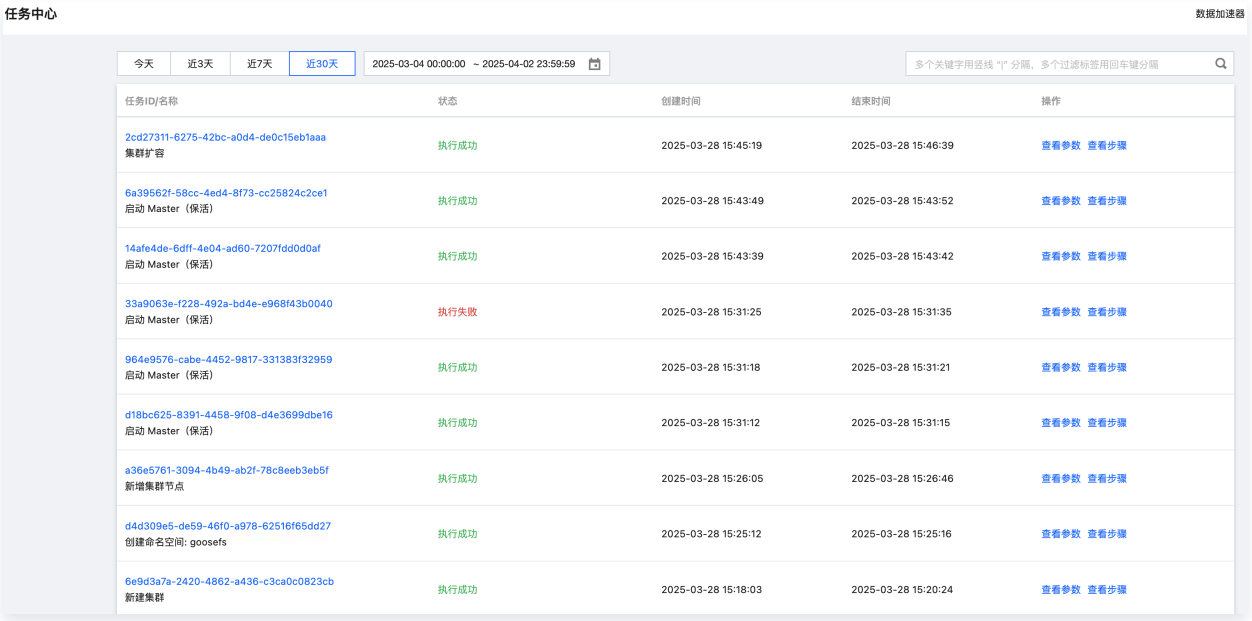
最近更新时间：2025-04-11 18:41:42

简介

GooseFS 集群的创建、集群管理和删除等操作都被视为异步任务处理。您可以通过 GooseFS 控制台，在 GooseFS 集群详情页中管理所有集群相关的任务。下面为您详细介绍如何管理 GooseFS 集群任务。

操作步骤

1. 登录 数据加速器 GooseFS 控制台 。
2. 在左侧导航中，单击 **GooseFS** > **实例列表**，进入 GooseFS 集群列表页面。
3. 单击需要管理的 GooseFS 集群 **ID/名称**，进入集群详情页面，在侧边栏中单击**任务中心**选项，进入任务管理子页面。
4. 根据所需查询的日期，选择对应日期范围内的任务列表，可选参数包括。
   - 特定天数内：支持查询特定天数内的任务情况，如今天、近3天、近7天、近30天内的任务列表。
   - 自选时间范围：支持查询指定时间窗口内的任务列表。
5. 在任务列表中，可以单击**任务 ID/名称**或者**查看步骤**按钮，通过控制台侧拉抽屉查看任务步骤。



6. 在任务步骤中，可以单击**具体流程**，在弹窗中查看每个步骤的详细流程的执行状态、创建时间和结束时间。

注意：

任务中心暂不支持暂停任务或者重启任务，如需重启任务，请回到对应操作的交互页面重新执行。例如重新进行一个扩容节点的任务，需要重新回到节点管理页面重新执行添加节点的操作。

删除 GooseFS 集群

最近更新时间：2024-10-12 11:37:31

简介

您可以通过 GooseFS 控制台，在 GooseFS 列表页面删除 GooseFS 集群。关于 GooseFS 集群的概念，请参见 GooseFS 集群概述 。下面将为您详细介绍如何删除 GooseFS 集群。

操作步骤

1. 登录 数据湖存储 GooseFS 控制台。
2. 在左侧导航中，单击 GooseFS 集群列表，进入 GooseFS 集群列表页面。
3. 在指定的 GooseFS 集群展示行，单击**删除按钮**。
4. 在弹窗中输入指定的实例名称，进行二次确认。
5. 单击确认后，集群进入删除流程。

> **注意：**
> - 集群删除后，集群配置以及集群中缓存的文件无法恢复，请谨慎执行集群删除操作。
> - GooseFS 集群中缓存的文件，如果有存储在持久化存储层如对象存储 COS 上，那么不会影响文件的持久化存储；正在写入的文件有可能因为集群删除导致文件写入失败。
> - GooseFS 集群删除操作会被记录在操作审计中，如需审计对应的操作记录，可登录 操作审计控制台 查阅。

GooseFS 集群监控

最近更新时间：2024-06-05 09:29:41

简介

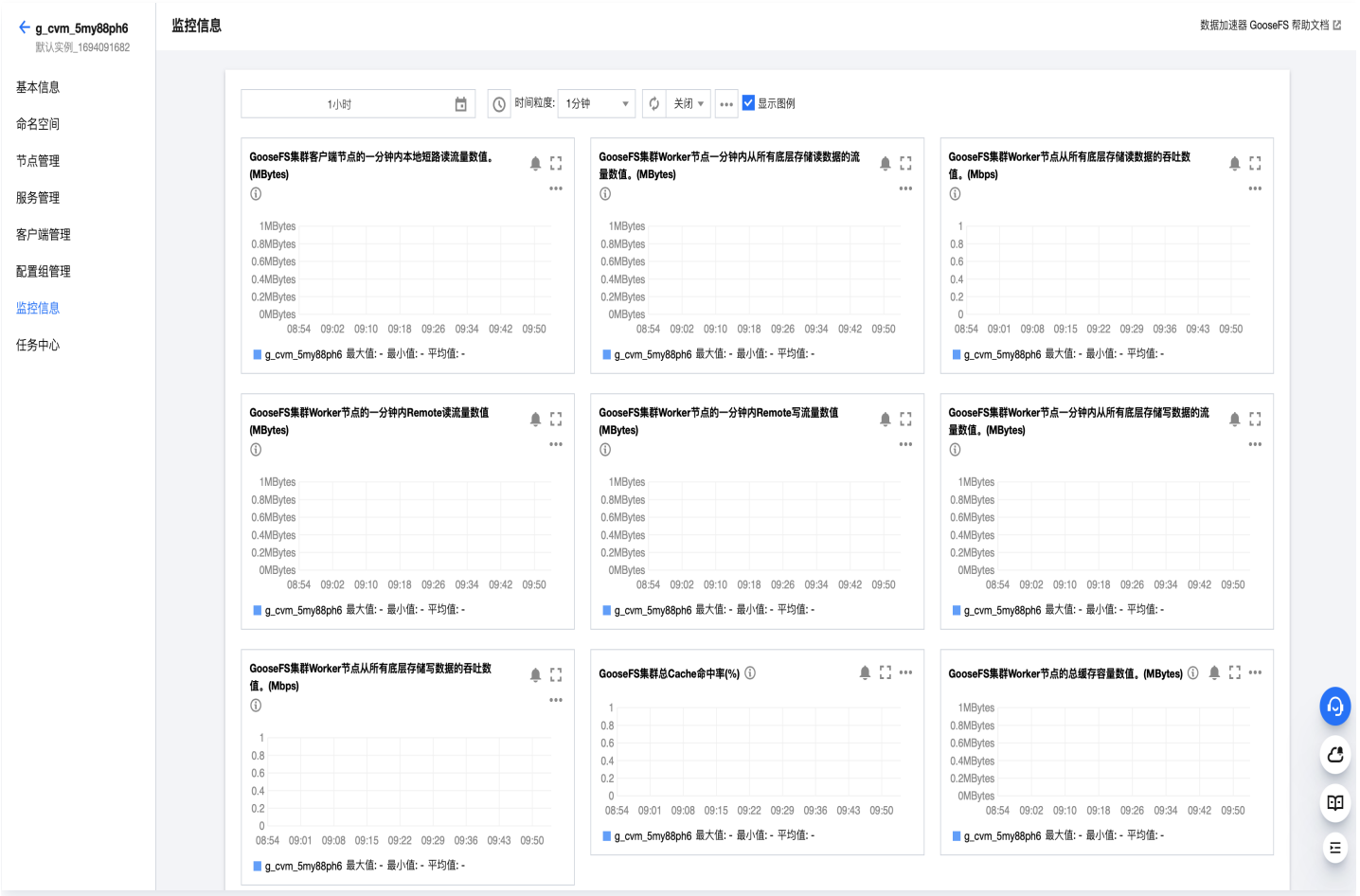
GooseFS 为用户提供了集群级别的运行指标监测页面，您可以在该页面中查看 GooseFS 集群的缓存容量，缓存命中率，读写请求次数等关键监控数据。

> **注意：**
> - 如需通过子账号登录腾讯云 GooseFS 控制台并查阅监控数据，需要有腾讯云监控 获取监控数据的接口权限，可前往腾讯云访问管理页面，参考文档 配置相应策略。
> - 集群级别监控指标采集并上报有分钟级延迟，监控指标中涉及的用量数据仅作为实时监测参考，不作为计量计费依据。

操作步骤

1. 登录 数据湖存储 GooseFS 控制台 。
2. 在左侧导航中，单击 GooseFS 集群列表，进入 GooseFS 集群列表页面。
3. 点击需要监测的 GooseFS 集群实例 ID，进入集群详情页，点击集群左侧侧边栏中的监控信息入口，进入监控信息详情页。
4. 监控信息详情页中，可按需选取指定的监控周期、时间粒度等过滤条件，设置是否自动刷新。
5. 监控信息详情页默认展示当前已有的全量监控指标，可以按需点击监控指标卡片页右上方的按钮 icon，针对特定监控指标发起告警配置、全屏展示、导出监控图表等操作。

版权所有：腾讯云计算（北京）有限责任公司



指标介绍

基础指标名	中文含义	指标解释
cluster_bytes_read_local	集群本地短路读流量数值（MBytes/Min）	客户端节点和 Worker 节点同节点部署时，如果 Worker 节点上有数据，客户端会直接从本地节点上读取数据。该指标用于记录本地读取数据的流量数值。
cluster_bytes_read_remote	集群Remote读流量数值（MBytes/Min）	客户端节点和 Worker 节点不在一个节点时，客户端从 Worker 节点中读取缓存数据会产生 Remote 读流量。该指标用于记录 Remote 读数据流量数值。
cluster_bytes_read_ufs_all	集群读 UFS 流量数值（MBytes/Min）	文件不在 GooseFS 集群缓存中时，GooseFS 会到底层存储引擎拉取数据。该指标用于记录 GooseFS 集群从底层存储引擎上读取数据的流量。
cluster_bytes_read_ufs_throughput	集群读 UFS 吞吐数值（MBytes/Min）	GooseFS 集群从底层存储引擎上读取数据的吞吐。

cluster_bytes_write_ufs_throughput	集群写 UFS 吞吐数值（MBytes/Min）	GooseFS 集群往底层存储引擎写入数据的吞吐。
cluster_bytes_written_remote	集群 Remote 写流量数值（MBytes/Min）	客户端往 Worker 节点写入数据的流量。
cluster_bytes_written_ufs_all	集群写 UFS 流量数值（MBytes/Min）	在写缓存模式下，客户端会往底层存储引擎和 Worker 节点写入数据；GooseFS 集群往底层存储引擎写入数据的流量。
cluster_cache_hit	集群总 Cache 命中率（%）	缓存命中率 =（本地短路读流量 + 集群 Remote读流量）/集群总流量
cluster_capacity_total	集群总缓存容量数值（MBytes）	GooseFS 集群管理的 Worker 节点的总物理容量，例如每个 Worker 节点有4块3.68TB的磁盘，总共10个 Worker 节点，那么总容量为147.2TB。
cluster_capacity_used	集群总已有缓存容量数值（MBytes）	GooseFS 集群中缓存的文件总容量。
cluster_local_cache_hit	集群本地 Cache 命中率（%）	本地 Cache 命中率 = 本地短路读流量 / 集群总流量
cluster_remote_cache_hit	集群远程 Cache 命中率（%）	远程 Cache 命中率 = 集群 Remote 读流量 / 集群总流量
cluster_space_used	集群空间使用率（%）	集群空间使用率 = 集群总已有缓存容量数值 / 集群总缓存容量数值
cluster_workers	集群总 Worker 节点数量（个）	集群中已管控的 Worker 点数量。
master_complete_file_ops	集群 CompleteFile 操作总请求次数（个）	CompleteFile 操作用于关闭文件句柄。
master_create_directory_ops	集群 CreateDirectory 操作总请求次数（个）	CreateDirectory 操作用于创建目录。
master_delete_path_ops	集群 Delete 操作总请求次数（个）	Delete 操作用于删除文件。
master_directories_created	集群 CreateDirectory 操作总成功请求次数（个）	CreateDirectory 操作的成功请求次数。
master_file_infos_got	集群 GetFileInfo 操作总成功请求次数（个）	GetFileInfo 操作用于获取文件元数据信息。
master_files_freed	集群 FreeFile 操作总成功请求次数（个）	FreeFile 操作用于释放已缓存的文件。
master_files_pinned	集群当前已锁定的缓存文件数（个）	锁定的缓存文件不会被 GooseFS 集群淘汰。

版权所有：腾讯云计算（北京）有限责任公司

master_mount_ops	集群 Mount 操作总请求次数（个）	Mount 操作用于在客户端节点建立文件系统挂载点。
master_total_paths	集群总文件（含目录）数（个）	集群元数据信息中记录的总文件数目，该数值包含目录。
master_unmount_ops	集群 Unmount 操作总请求次数（个）	Unmount 操作用于在客户端节点解除文件系统挂载点。

版权所有：腾讯云计算（北京）有限责任公司

核心特性
缓存能力

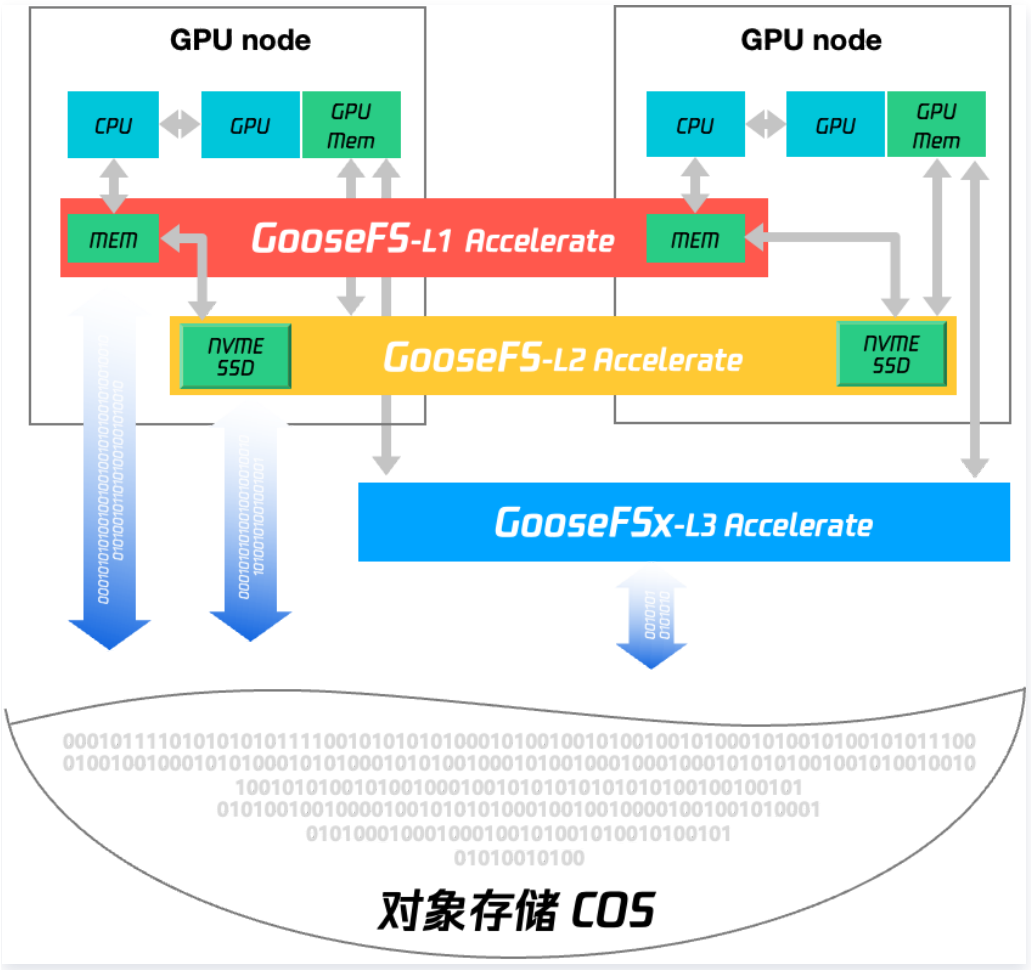
最近更新时间：2024-10-09 16:50:01

缓存能力概述

GooseFS 通过数据亲和性调度能力，将热点数据从海量分布式存储系统中调度到近计算端节点来缩短数据 IO 链路，提升数据吞吐。在了解 GooseFS 缓存能力的之前，需要先了解缓存能力关联的几个基础概念：

- 命名空间 NameSpace：GooseFS 通过命名空间组织和管理缓存数据。命名空间和外部存储是多对一的关系，一个或者多个命名空间可以关联到同一个外部存储服务；GooseFS 支持 COS、CHDFS、TStor OneCOS 和本地 HDFS 等多种不同类型的外部存储服务，可以利用这些外部存储的持久化存储能力存储海量数据并保证数据的可靠性。
- 本地缓存存储：GooseFS 通过 Worker 节点的物理介质提供分布式缓存能力，支持将计算产生的数据缓存计算节点本地的内存、磁盘或者可用区级别的云盘上，每个 Worker 节点的缓存容量可由用户自己配置。

用户可以操作 Cache Policy，决定数据 IO 过程中和本地缓存存储交互或者命名空间关联的外部存储交互。



GooseFS 缓存配置

用户可以进入 goosefs-site.properties 文件中查看 GooseFS 缓存配置情况。目前 GooseFS 的缓存设置可以从缓存层级、缓存淘汰策略等方面了解，缓存层级方面主要有单层存储和多层存储两种，缓存淘汰策略主要区分为 LRU 和 LRFU 两种。以下进行详细介绍。

1. 缓存层级

GooseFS 提供单层存储和多层存储两种缓存层级。单层存储只使用一种存储介质，多层存储使用多种存储，可以根据业务负载情况来决定使用何种存储介质，以提供匹配的 I/O 性能。GooseFS 默认配置为单层存储，在多层存储的场景下，缓存淘汰会带来较多的性能开销，因此推荐使用**单层存储**。根据 I/O 性能的高低，一般可以选用 MEM（内存）、SSD（固态存储）和 HDD（硬盘存储）作为存储介质。

通过修改 goosefs-site.properties 配置文件中的 levels 参数可以修改缓存层级，入参为 1 时代表只使用单层存储，入参为 3 代表使用三层存储：

```
goosefs.worker.tieredstore.levels=1
```

通过修改 goosefs-site.properties 配置文件中的 alias 参数可以修改缓存层对应的存储介质：

```
goosefs.worker.tieredstore.level{x}.alias=MEM
```

> **注意：**
> level{x} 代表第几层存储，例如 level0 代表单层存储。

1.1 单层存储

单层存储模式下，常用的系统配置项如下：

```
goosefs.worker.tieredstore.levels=1
goosefs.worker.tieredstore.level0.alias=HDD
goosefs.worker.ramdisk.size=16GB
goosefs.worker.memory.size=100GB
goosefs.worker.tieredstore.level0.dirs.path=/data/GooseFSWorker,/data1/GooseFSWor
ker,/data2/GooseFSWorker,/data3/GooseFSWorker,/data4/GooseFSWorker
goosefs.worker.tieredstore.level0.dirs.mediumtype=MEM,MEM,MEM,SSD,SSD
goosefs.worker.tieredstore.level0.dirs.quota=16GB,16GB,16GB,100GB,100GB
```

以下逐项介绍配置项内容。

- ramdisk.size：该配置项用于指定 worker 节点占用内存的大小。ramdisk 的容量必须比 memory 小。设置后 GooseFS 会为每个 worker 节点分配指定大小的内存空间，并消耗系统总内存。
- memory.size：该配置项用于指定 GooseFS 系统总内存，设置后 GooseFS 将会自动占用指定大小的存储介质，实际物理存储空间必须比 memory 数值大。
- dirs.path：该配置项用于指定目录，由 GooseFS 为指定目录分配存储介质，需要结合 dirs.mediumtype 使用，如上示例展示将 /data/GooseFSWorker 挂载到 MEM 存储介质上，将/data4/GooseFSWorker 挂载到 SSD 存储介质上。需要注意，每个目录的顺序需要和存储介质的顺序一一匹配。

- dirs.mediumtype：该配置项用于指定存储介质，由 GooseFS 为指定目录分配存储介质，需要结合 dirs.path 使用。默认可选的存储介质包括 MEM，SSD，如果挂载了其他存储介质，如 HDD，可按需修改配置。
- dirs.quota：该配置项用于指定每个目录的预置空间，配置后由 GooseFS 为指定目录分配好空间，需要和目录的顺序一一对应。如上示例展示了需要为 /data/GooseFSWorker,/data1/GooseFSWorker,/data2/GooseFSWorker 等目录分配 16GB MEM 空间，为 /data3/GooseFSWorker,/data4/GooseFSWorker 目录分配 100GB SSD 空间。

1.2 分层存储

分层存储模式下，数据块的读写模式与单层存储的读写模式存在差异。单层存储模式下数据会直接从一种存储介质中读写，多层存储模式下数据会默认优先写入最顶层存储介质，读取的时候则需要将数据先挪到最顶层存储介质。具体地：

- **数据写入**：新的数据块会被默认写入最顶层的存储介质；如果最顶层存储介质的存储空间已经满了，GooseFS 会把数据顺位存储到下一层的存储介质中；如果所有存储介质都已经存满数据，那么 GooseFS 会按照用户指定的缓存淘汰策略淘汰过期数据；如果过期数据无法淘汰，并且所有可用存储空间已满，则无法写入数据。
- **数据读取**：多层存储模式下，会将冷数据透明地转储到更低层次的存储介质中，读取数据会将数据加热并放置到顶层存储中。

> ⚠ **注意：**
> - 在既定的淘汰策略下，GooseFS 会按照指定的释放空间来清理一定量的数据，该参数可以通过 goosefs.worker.tieredstore.free.ahead.bytes 指定，默认值为0。
> - 分层存储模式下，读取数据时可能会频繁将数据从低层存储介质转储到顶层存储介质中，进而导致频繁的缓存淘汰情况，对性能带来一定损耗，一般情况下，**推荐使用单层存储**。

分层存储模式下，常用的系统配置项如下：

```
goosefs.worker.tieredstore.levels
goosefs.worker.tieredstore.level{x}.alias
goosefs.worker.tieredstore.level{x}.dirs.path
goosefs.worker.tieredstore.level{x}.dirs.mediumtype
goosefs.worker.tieredstore.level{x}.dirs.quota
```

其中，x 代表缓存层级，一般而言可以设置 3 层存储，分别对应 MEM，SSD，HDD等存储介质。以 2 层存储，存储介质分别为 MEM，SSD，每一层分配100GB存储为例，对应的配置信息如下：

```
goosefs.worker.tieredstore.levels=2
goosefs.worker.tieredstore.level0.alias=MEM
goosefs.worker.tieredstore.level0.dirs.path=/data/GooseFSWorker
goosefs.worker.tieredstore.level0.dirs.mediumtype=MEM
goosefs.worker.tieredstore.level0.dirs.quota=100GB
goosefs.worker.tieredstore.level1.alias=SSD
goosefs.worker.tieredstore.level1.dirs.path=/data1/GooseFSWorker
goosefs.worker.tieredstore.level1.dirs.mediumtype=SSD
goosefs.worker.tieredstore.level1.dirs.quota=100GB
```

版权所有：腾讯云计算（北京）有限责任公司

GooseFS 支持配置多层存储，层数没有限制，但是每一层的命名（alias）均需要保证唯一性。一般来说，提供 3 层缓存，每一层分别对应 MEM，SSD，HDD就起到较好的分层效果。

2. 缓存淘汰策略

GooseFS 提供了两种缓存淘汰策略：

- LRUAnnotator：按照近期使用次数最少（least-recently-used）的顺序淘汰缓存，是 GooseFS 的默认策略。
- LRFUAnnotator：按照近期使用次数最少（least-recently-used）和近期使用频次最少（least-frequently-used）的顺序淘汰缓存，通过权重配置 goosefs.worker.block.annotator.lrfu.step.factor 和 goosefs.worker.block.annotator.lrfu.attenuation.factor 来调整两种策略在淘汰过程中起的作用。
  - 如果将 least-recently-used 的权重调整至最大，那么该策略的表现与 LRUAnnotator一致。

可以在 goosefs.worker.block.annotator.class 这一配置项中配置缓存淘汰策略，配置时需要正确指定策略名称：

```
goosefs.worker.block.annotator.LRUAnnotator
goosefs.worker.block.annotator.LRFUAnnotator
```

数据生命周期管理

这里的生命周期是指在 GooseFS 中的数据生命周期，而不是远端存储系统 UFS 的，生命周期管理主要有如下四种操作：

- free：此操作用于从 GooseFS 中删除对应目录或者文件的数据（不会删除 UFS 中的对应数据），此类操作主要是用来释放 GooseFS 的缓存空间，供其他更热的数据使用。
- load：此操作用于将 UFS 对应目录或者文件的数据加载到 GooseFS 中，此类操作主要用于冷数据转热，从而提高 I/O 性能。
- persist：此操作用于将 GooseFS 内的数据写入到 UFS 存储中，此类操作主要用于持久化数据，从而避免数据丢失。
- TTL(Time to Live)：TTL 属性主要用于设置目录或者文件在 GooseFS 中的生命周期，在超过其生命周期后，会从 GooseFS 和 UFS 中删除。通过配置，也可支持仅删除 GooseFs 数据或仅释放磁盘空间。

1. 从 GooseFS 中释放数据

通过 free 指令可以从 GooseFS 中释放数据，如下示例展示了释放数据后，GooseFS 中数据的状态变成 0%：

```
$ goosefs fs free /data/test.txt
/data/test.txt was successfully freed from GooseFS space.
$ goosefs fs ls /data/test.txt
-rw-rw-rw-  hadoop         hadoop                        14       PERSISTED 03-11-
2021 11:46:15:000 0% /data/test.txt
```

示例中的 /data/test.txt 可以是任何合法的 GooseFS 文件路径，如果数据存储于远端存储 UFS 中，则可以通过 load 指令重新加载。

> ⚠ **注意：**
> 一般而言，推荐通过配置缓存策略自动清理 GooseFS 中的历史数据。

2. 往 GooseFS 中加载数据

通过 load 指令可以从 GooseFS 中加载数据，如下示例展示了释放数据后，GooseFS 中数据的状态变成 100%：

```
$ goosefs fs load /data/test.txt
/data/test.txt loaded
$ goosefs fs ls /data/test.txt
-rw-rw-rw-  hadoop         hadoop                         14       PERSISTED 03-11-
2021 11:46:15:000 100% /data/test.txt
```

示例中的 /data/test.txt 可以是任何合法的 GooseFS 文件路径。如果是从本地文件系统中加载数据，则需要使用 copyFromLocal 指令，需要注意的是，从本地文件系统中加载数据的操作并不会将数据持久化到远端存储 UFS 中。在默认情况下， GooseFS 能在首次访问文件时自动加载数据到 GooseFS 到缓存中，因此一般无需使用该指令加载数据；但如果业务需要提前加载数据到缓存中，可以通过该指令加载。

3. 在 GooseFS 中持久化数据

通过 persist 指令可以将数据持久化到远端存储系统 UFS 中：

```
$ goosefs fs persist /data/test.txt
$ goosefs fs ls /data/test.txt
-rw-rw-rw-  hadoop         hadoop                         14       PERSISTED 03-11-
2021 11:46:15:000 100% /data/test.txt
```

示例中的 /data/test.txt 可以是任何合法的 GooseFS 文件路径。推荐通过缓存策略配置来自动执行持久化的操作。

4. 设置数据 TTL 属性

GooseFS 支持为命名空间中的任意文件或者目录设置 TTL 属性。该属性可以保证业务数据可以按照指定的时间周期定期删除，为新文件释放空间，保证本地磁盘空间的有效利用。 GooseFS 文件和目录的 TTL 属性可以作为文件元数据的一部分，可以保证在集群重启后 TTL 属性依然生效，后端线程的定期巡检程序会检查这些 TTL 属性并自动清理过期文件。

定期巡检程序的巡检周期和巡检类型可以在 goosefs-site.preperties 中设置，以下示例将巡检周期设置为 10 分钟，巡检类型设置为 FREE：

```
goosefs.master.ttl.checker.interval=10m
goosefs.user.file.create.ttl.action=FREE
```

设置完毕后，GooseFS 后台线程会每隔 10 分钟巡检一遍，当次巡检周期内发现的过期文件会在下一次巡检周期过完后释放缓存资源。如 00:00:00 进行了第一次巡检发现一批过期文件，这批文件缓存会在 00:10:00 清除。

> **注意：**
> 默认情况下该入参单位为毫秒（ms），可以在入参时指定好检测时间周期的单位，如秒（s）,分钟（m），小时（h）等，更多说明可以查看配置说明的介绍。

数据复制管理

数据复制主要分为被动复制和主动复制两种形式。

1. 被动复制

GooseFS中的每个文件都包含一个或多个分布在集群中存储的存储块，默认情况下 GooseFS 可以根据负载和容量情况自动调整数据分块的复制级别。被动复制会发生在以下场景：

- 多个客户端同时读取相同的块，会导致多个 block 同时在不同的 worker 上存在。
- 当开启优先本地读时，数据不在本地，则会发起 remote read，并保存在本地 worker 上。
- 当被动复制产生的副本数大于设定的副本数目时，其会异步的去删除多余的副本。以上过程对用户完全透明。

2. 主动复制

主动复制是通过设置文件的副本数目实现的，对于不满足设定副本数的 block，其会异步的补足，对于超过设定副本数的 block，其会删除多余的副本。具体的，可以通过如下指令调整主动复制的级别：

```
$ goosefs fs setReplication  [-R] [--max | --min ] <path>
```

相关参数说明如下：

- max：指定文件的最大副本数，默认值为 -1，表示不设置上限；如果设置为 0 代表不在 GooseFS 中存储指定文件的冷数据。一般设置为正整数，设置后 GooseFS 会检查文件副本数量并删除多余的副本。
- min：指定文件的最小副本数，默认值为0，代表 GooseFS 会在数据变冷后删除该数据不留存任何副本。一般设置为正整数，并且**需要小于 max 值**，设置后 GooseFS 会检查文件副本数量，如果副本数量比最小值小会自动补齐副本数。
- path：指定的文件名，可以为目录或者具体的文件路径。
- R：如果 path 中入参为目录，则指定 -R 可以按照指定的最小副本数和最大副本数，重复递归复制指定目录下的所有文件和子目录。

如果需要查看指定文件的复制情况，可以通过 stat 指令查看，如下示例展示 /data/test.txt 这个文件的最大副本数为 -1，意味着该文件变冷后会被过期删除：

```
$ goosefs fs stat /data/test.txt
/data/test.txt is a file path.
FileInfo{fileId=50331647, fileIdentifier=null, name=test.txt,
path=/data/test.txt, ufsPath=hdfs://172.16.16.16:4007/data/test.txt, length=0,
blockSizeBytes=134217728, creationTimeMs=1618193473555, completed=true,
folder=false, pinned=false, pinnedlocation=[], cacheable=true, persisted=true,
blockIds=[], inMemoryPercentage=100, lastModificationTimesMs=1616763603692,
ttl=-1, lastAccessTimesMs=1616763603692, ttlAction=DELETE, owner=hadoop,
group=supergroup, mode=420, persistenceState=PERSISTED, mountPoint=false,
replicationMax=-1, replicationMin=0, fileBlockInfos=[], mountId=1,
inGooseFSPercentage=100, ufsFingerprint=TYPE|FILE UFS|hdfs OWNER|hadoop
GROUP|supergroup MODE|420 CONTENT_HASH|(len:0,_modtime:1616763603692) ,
acl=user::rw-,group::r--,other::r--, defaultAcl=}
This file does not contain any blocks.
```

查看缓存用量

GooseFS 会记录本地缓存的容量和使用情况，可以通过如下指令查看 GooseFS 的运行情况，针对性地管理和维护本地缓存。

- 查看 GooseFS 正在使用的缓存，单位为字节：

```
$ goosefs fs getUsedBytes
Used Bytes: 0
```

- 查看 GooseFS 总缓存容量，单位为字节：

```
$ goosefs fs getCapacityBytes
Capacity Bytes: 1610612736000
```

- 查看 GooseFS 的缓存使用报告：

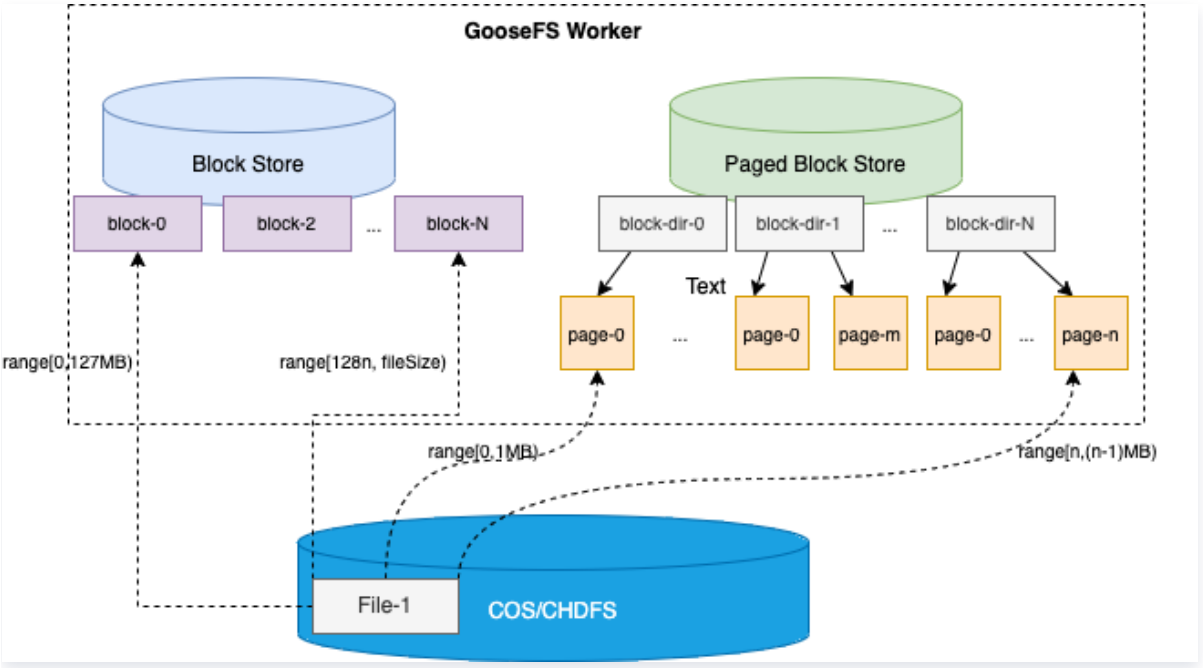
```
$ goosefs fsadmin report
GooseFS cluster summary:
    Master Address: 172.16.16.16:19998
    Web Port: 19999
    Rpc Port: 19998
    Started: 04-12-2021 10:52:05:255
    Uptime: 0 day(s), 1 hour(s), 28 minute(s), and 57 second(s)
    Version: 2.5.0-SNAPSHOT
    Safe Mode: false
    Zookeeper Enabled: false
    Live Workers: 3
    Lost Workers: 0
    Total Capacity: 1500.00GB
        Tier: HDD  Size: 1500.00GB
    Used Capacity: 0B
        Tier: HDD  Size: 0B
    Free Capacity: 1500.00GB
```

Page Store 缓存

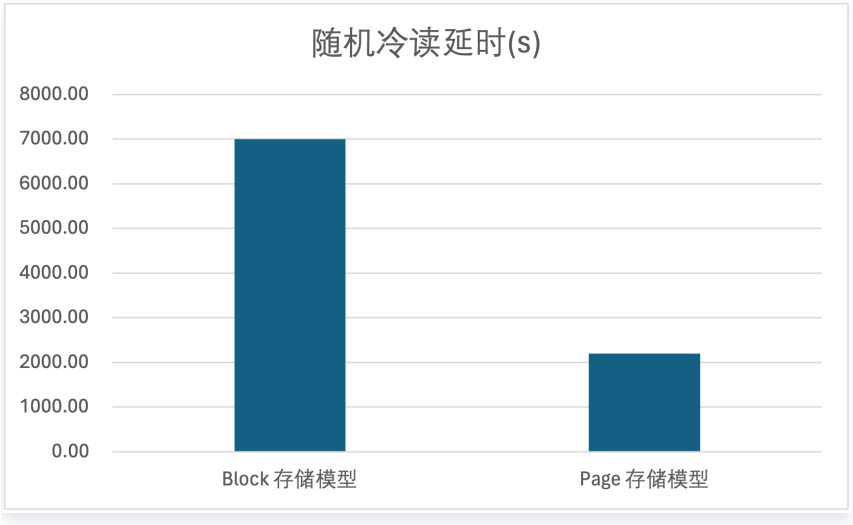
最近更新时间：2024-06-11 10:54:21

Page Store 特性概述

从 GooseFS-1.7.0 以上的版本开始支持 Page Store 缓存新特性。相较于传统的 Block Store 缓存，前者可以极大地优化离散 IO 访问模型的缓存空间利用率以及冷读效率。Page Store 与 Block Store 之间的架构比较如下。



Page Store 存储架构打破了原先 Worker 节点上每个块默认 128MB 作为缓存最小单位的限制，在不额外增加任何 Master 节点上的元数据负担的基础上，允许以默认 1MB 大小的页形式在 Worker 节点上缓存数据。
这种存储架构，一方面可以极大地提高缓存空间的利用率；另一方面也可以在单文件随机冷读场景中，可以非常明显地降低请求响应与缓存生效时延。
下面给出了一个 OLAP 查询的随机读场景中，Block 模式与 Page 模式的冷读延时对比。



Page Store 使用方法

为了使用 Page Store 模式，需要将 Worker 的数据存储增加如下配置：

```
# ...

goosefs.worker.block.store.type=PAGE
# Worker page storage type.
goosefs.worker.page.store.dirs=/Users/yuyang/goosefs-data/paged-block
goosefs.worker.page.store.page.size=2MB
goosefs.worker.page.store.size=6MB
goosefs.worker.page.store.overhead=0
goosefs.worker.network.reader.buffer.size=1MB

# ...
```

然后将配置同步分发到所有节点以后，重启所有节点的 Worker 进程即可生效。

Page Store 与 Block Store 之间的目录结构说明

在 Page Store 的数据目录下，采用了四级目录结构：

- 第一级子目录是默认的 LOCAL，不可更改；
- 第二级子目录表征该目录下的 Page Size。例如，图中的子目录 2097152 表示该目录下每个 Page 的大小都是 2MB；
- 第三级是一个哈希桶值，该值是使用每个 Block 目录名的哈希值与 ${goosefs.worker.page.store.local.store.file.buckets} 计算得到；
- 第四级子目录就是 Block 目录了，该目录下存放的都就是属于该 Block 的所有缓存 Page 页，该目录的命名规则为：paged_block_${blockId}。

> ⚠ **注意：**
> 在第二级子目录下，通常还会有一个 .TEMP 目录，该目录下会存放所有正在写入的 Block 数据。.TEMP 目录下每个 Block 目录的命名规则为 paged_block_${blockId}session{sessionId}。

关于 Page Store 与 Block Store 之间的互相切换

目前，在默认未开启短路读写的情况下，是允许通过更改配置的方式来自由切换存储类型，即修改配置项 goosefs.worker.block.store.type 为对应的存储类型，然后重启 Worker Server 即可生效。 Worker 将会根据对应的存储类型所指定的数据路径来加载和缓存数据。

在开启短路读的情况下，也可以通过上述方式修改生效，但是必须同时重启 Client 所在进程，否则可能会出现短路读写异常的问题。

PageStore 的配置参数

参数	默认值	描述
goosefs.worker.block.store.type	FILE	• 指定 Worker 上的存储类型，可选项为 FILE 和 PAGE。 • 默认为 FILE，即传统的 Block 存储模式，指定 PAGE 则为 Page 存储模式。
goosefs.worker.page.store.page.size	1MB	指定每个 Page 页的大小。默认大小为 1MB，这里可以按需指定，例如 128KB 或 256KB 亦可。
goosefs.worker.page.store.dirs	/tmp/goosefs-cache	指定 Page Store 的数据目录。例如：`/data/goosefs-data/paged-block` 。
goosefs.worker.page.store.size	512MB	指定 Page Store 的数据目录的大小，默认为 512MB，如果超出了容量限制，则会触发 Page 粒度的淘汰。
goosefs.worker.cache.request.pending.timeout	500ms	• 用来优化高并发冷读场景下的缓存击穿问题的超时等待选项，默认是 500 毫秒。 • 如果在同一 Worker 节点上发生了缓存击穿的并发读取，则后到的请求会尝试等待 500ms，以便于直接从缓存中直接返回数据，而不是穿透到底层存储。 • 若等待超时，才会从底层存储读取并返回。 • 当该值设置为小于等于 0 的值时候，则相当于关闭了缓存击穿优化。
goosefs.worker.page.store.overhead	0.1	Page 存储空间的预留分位。默认值为 0.1，表示会预留百分之十的空间作为保留空间。达到水位以后，开始触发淘汰动作。
goosefs.worker.page.store.evictor.class	com.qcloud.cos.goosefs.client.file.cache.evictor.LRUCacheEvictor	Page 存储的淘汰算法，支持的选项为： • com.qcloud.cos.goosefs.client.file.cache.evictor.LRUCacheEvictor； • com.qcloud.cos.goosefs.client.file.cache.evictor.LFUCacheEvictor。
goosefs.worker.page.store.eviction.retries	10	最大淘汰尝试次数，默认为 10 次。

goosefs.worker.page.store.evictor.lfu.logbase	2.0	指定 LFU 淘汰算法的 LogBase。
goosefs.worker.page.store.local.store.file.buckets	1000	存放 Paged Block 目录的 Hash 桶数目，默认为 1000。

关于分层存储的说明

GooseFS 在 Page Store 存储类型下，不支持分层存储，即只有默认的第一层存储介质可用。这样设计的原因是在于，离散的 Page 页在多层存储之间索引和流动的开销会很大，对 Worker 节点的资源占用和性能都有极大的影响。

透明加速能力

最近更新时间：2023-09-05 17:48:44

概述

透明加速能力用于加速 CosN 访问 COS 的性能。CosN 工具 是基于腾讯云对象存储（Cloud Object Storage，COS）提供的标准的 Hadoop 文件系统实现，可以为 Hadoop、Spark 以及 Tez 等大数据计算框架集成 COS 提供支持。用户可使用实现了 Hadoop 文件系统接口的 CosN 插件，读写存储在 COS 上的数据。对于已经使用 CosN 工具访问 COS 的用户，GooseFS 提供了一种客户端路径映射方式，让用户可以在不修改当前 Hive table 定义的前提下，仍然能够使用 CosN scheme 访问 GooseFS，该特性方便用户在不修改已有表定义的前提下，对 GooseFS 的功能和性能进行对比测试。对于云 HDFS 用户（CHDFS），也可以通过修改配置，实现使用 OFS Scheme 访问 GooseFS 的目的。

CosN Schema 和 GooseFS Schema 的映射说明如下：

假设 Namespace warehouse 对应的 UFS 路径为 `cosn://examplebucket-1250000000/data/warehouse/`，则 CosN 到 GooseFS 的路径映射关系如下：

```
cosn://examplebucket-1250000000/data/warehouse -> /warehouse/
cosn://examplebucket-1250000000/data/warehouse/folder/test.txt -
>/warehouse/folder/test.txt
```

GooseFS 到 CosN 的路径映射关系如下：

```
/warehouse ->cosn://examplebucket-1250000000/data/warehouse/
/warehouse/ -> cosn://examplebucket-1250000000/data/warehouse/
/warehouse/folder/test.txt -> cosn://examplebucket-
1250000000/data/warehouse/folder/test.txt
```

CosN Scheme 访问 GooseFS 特性，通过在客户端维持 GooseFS 路径和底层文件系统 CosN 路径之间的映射关系，并将 CosN 路径的请求转换为 GooseFS 的请求。映射关系周期性刷新，您可以通过修改 GooseFS 配置文件 goosefs-site.properties 中的配置项 goosefs.user.client.namespace.refresh.interval 调整刷新间隔，默认值为60s。

> **注意**
> 如果访问的 CosN 路径无法转换为 GooseFS 路径，对应的 Hadoop API 调用会抛出异常。

操作示例

该示例演示了 Hadoop 命令行以及 Hive 中，如何使用 gfs://、 cosn://、ofs:// 三种 Schema 访问 GooseFS。操作流程如下：

1. 准备数据和计算集群

- 参考 创建存储桶 文档，创建一个测试用途的存储桶。
- 参考 创建文件夹 文档，在存储桶根路径下创建一个名为 ml-100k 的文件夹。

- 从 Grouplens 下载 ml-100k 数据集，并将文件 u.user 上传到 `<存储桶根路径>/ml-100k` 下。
- 参考 EMR 指引文档，购买一个 EMR 集群并配置 HIVE 组件。

2. 环境配置

i. 将 GooseFS 的客户端 jar 包（goosefs-1.0.0-client.jar）放入 share/hadoop/common/lib/ 目录下：

```
cp goosefs-1.0.0-client.jar  hadoop/share/hadoop/common/lib/
```

> **注意**
> 配置变更和添加 jar 包，需同步到集群上所有节点。

ii. 修改 Hadoop 配置文件 etc/hadoop/core-site.xml，指定 GooseFS 的实现类：

```
<property>
  <name>fs.AbstractFileSystem.gfs.impl</name>
  <value>com.qcloud.cos.goosefs.hadoop.GooseFileSystem</value>
</property>
<property>
  <name>fs.gfs.impl</name>
  <value>com.qcloud.cos.goosefs.hadoop.FileSystem</value>
</property>
```

iii. 执行如下 Hadoop 命令，检查是否能够通过 gfs:// Scheme 访问 GooseFS，其中 <MASTER_IP> 为 Master 节点的 IP：

```
hadoop fs -ls gfs://<MASTER_IP>:9200/
```

iv. 将 GooseFS 的客户端 jar 包放到 Hive 的 auxlib 目录下，使得 Hive 能加载到 GooseFS Client 包：

```
cp goosefs-1.0.0-client.jar  hive/auxlib/
```

v. 执行如下命令，创建 UFS Scheme 为 CosN 的 Namespace，并列出 Namespace。您可将该命令中的 examplebucket-1250000000 替换为您的 COS 存储桶，SecretId 和 SecretKey 替换为您的密钥信息：

```
goosefs ns create ml-100k cosn://examplebucket-1250000000/ml-100k  --secret
fs.cosn.userinfo.secretId=SecretId --secret fs.cosn.userinfo.secretKey=SecretKey-
-attribute fs.cosn.bucket.region=ap-guangzhou --attribute
fs.cosn.credentials.provider=org.apache.hadoop.fs.auth.SimpleCredentialProvider
goosefs ns ls
```

vi. 执行命令，创建 UFS Scheme 为 OFS 的 Namespace，并列出 Namespace。您可将该命令中的 instance-id 替换为您的 CHDFS 实例，1250000000 替换为您的 APPID：

```
goosefs ns create ofs-test  ofs://instance-id.chdfs.ap-
guangzhou.myqcloud.com/ofs-test --attribute fs.ofs.userinfo.appid=1250000000
goosefs ns ls
```

3. 创建 GooseFS Schema 表和查询数据

通过如下指令执行：

```
create database goosefs_test;

use goosefs_test;
CREATE TABLE u_user_gfs (
userid INT,
age INT,
gender CHAR(1),
occupation STRING,
zipcode STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION 'gfs://<MASTER_IP>:<MASTER_PORT>/ml-100k';

select sum(age) from u_user_gfs;
```

4. 创建 CosN Schema 表和查询数据

通过如下指令执行：

```
CREATE TABLE u_user_cosn (
userid INT,
age INT,
gender CHAR(1),
occupation STRING,
zipcode STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION 'cosn://examplebucket-1250000000/ml-100k';

select sum(age) from u_user_cosn;
```

5. 修改 CosN 的实现为 GooseFS 的兼容实现

修改 hadoop/etc/hadoop/core-site.xml：

```
 <property>
```

```
    <name>fs.AbstractFileSystem.cosn.impl</name>
    <value>com.qcloud.cos.goosefs.hadoop.CosN</value>
</property>
<property>
    <name>fs.cosn.impl</name>
    <value>com.qcloud.cos.goosefs.hadoop.CosNFileSystem</value>
</property>
```

执行 Hadoop 命令，如果路径无法转换为 GooseFS 中的路径，命令的输出中会包含报错信息：

```
hadoop fs -ls  cosn://examplebucket-1250000000/ml-100k/

Found 1 items
-rw-rw-rw-   0 hadoop hadoop      22628 2021-07-02 15:27 cosn://examplebucket-
1250000000/ml-100k/u.user

hadoop fs -ls cosn://examplebucket-1250000000/unknow-path
ls: Failed to convert ufs path cosn://examplebucket-1250000000/unknow-path to
GooseFs path, check if namespace mounted
```

重新执行 Hive 查询语句：

```
select sum(age) from u_user_cosn;
```

6. 创建 OFS Schema 表和查询数据

通过如下命令执行：

```
CREATE TABLE u_user_ofs (
userid INT,
age INT,
gender CHAR(1),
occupation STRING,
zipcode STRING)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY '|'
STORED AS TEXTFILE
LOCATION 'ofs://instance-id.chdfs.ap-guangzhou.myqcloud.com/ofs-test/';

select sum(age) from u_user_ofs;
```

7. 修改 OFS 的实现为 GooseFS 的兼容实现

修改 hadoop/etc/hadoop/core-site.xml：

```
<property>
```

```
    <name>fs.AbstractFileSystem.ofs.impl</name>
    <value>com.qcloud.cos.goosefs.hadoop.CHDFSDelegateFS</value>
</property>
<property>
    <name>fs.ofs.impl</name>
    <value>com.qcloud.cos.goosefs.hadoop.CHDFSHadoopFileSystem</value>
</property>
```

执行 Hadoop 命令，如果路径无法转换为 GooseFS 中的路径，则输出结果中会包含报错信息：

```
hadoop fs -ls  ofs://instance-id.chdfs.ap-guangzhou.myqcloud.com/ofs-test/

Found 1 items
-rw-r--r--   0 hadoop hadoop      22628 2021-07-15 15:56 ofs://instance-
id.chdfs.ap-guangzhou.myqcloud.com/ofs-test/u.user

hadoop fs -ls  ofs://instance-id.chdfs.ap-guangzhou.myqcloud.com/unknown-path
ls: Failed to convert ufs path ofs://instance-id.chdfs.ap-
guangzhou.myqcloud.com/unknown-path to GooseFs path, check if namespace mounted
```

重新执行 Hive 查询语句：

```
select sum(age) from u_user_ofs;
```

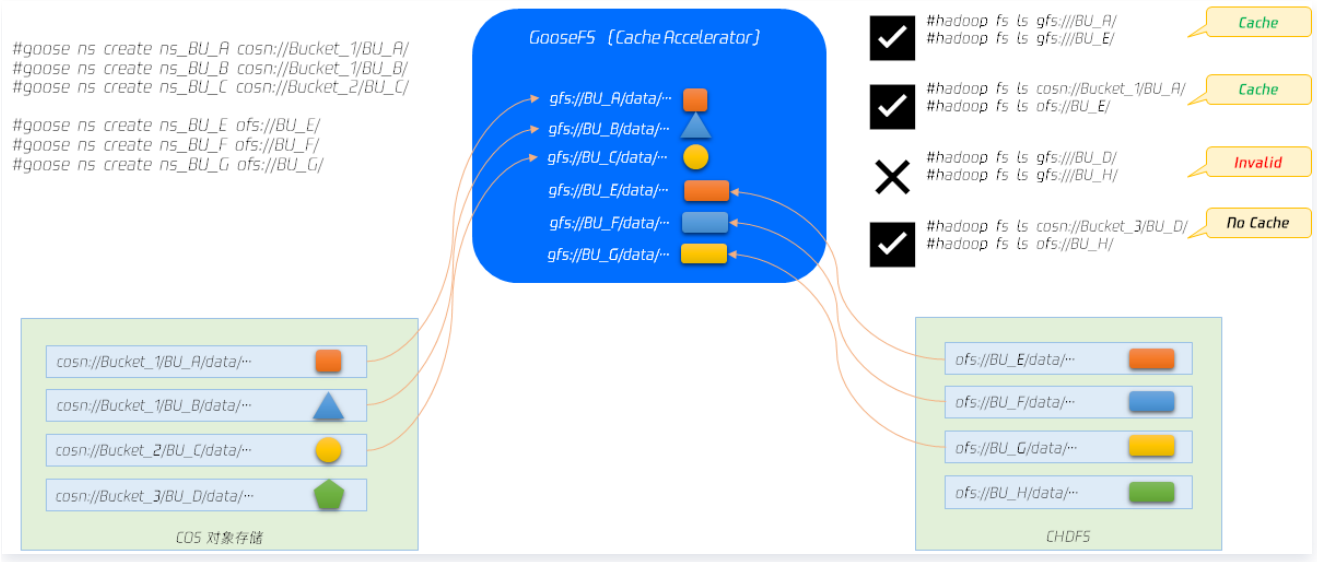
统一命名空间能力

最近更新时间：2024-11-12 17:52:12

统一命名空间能力概述

GooseFS 统一命名空间（NameSpace）能力通过透明的命名机制，可以融合多种不同的底层存储系统访问语义，为用户提供了一个统一的数据管理交互视图。

GooseFS 通过统一命名空间能力对接不同的底层存储系统，如本地文件系统、腾讯云对象存储（Cloud Object Storage，COS）、腾讯云云 HDFS（Cloud HDFS，CHDFS）等，与这些底层存储系统进行通信，并为上层业务提供统一的访问接口和文件协议。业务侧只需使用 GooseFS 的访问接口，即可访问存储在不同底层存储系统中的数据。



上图显示了统一命名空间的工作原理。您可以通过 GooseFS 创建命名空间的指令 create ns ，将 COS 和 云 HDFS 的指定文件目录挂载到 GooseFS 中，然后通过 gfs:// 的这一统一的 schema 访问数据。详细说明如下所示：

- COS 总共有3个存储桶，分别是 bucket-1、bucket-2、bucket-3，其中 bucket-1 有 BU_A 和 BU_B 两个目录，bucket-1 和 bucket-2 均挂载在 GooseFS 中。
- 云 HDFS 中有 BU_E、BU_F、BU_G 和 BU_H 共 4 个目录，其中除了 BU_H 其余目录均挂载到 GooseFS 上。
- 在 GooseFS 的文件操作中，如果以 gfs:// 这一统一的 schema 访问 BU_A 和 BU_E 这两个目录，均可正常访问，且文件缓存在 GooseFS 的本地文件系统中。
- BU_A 和 BU_E 这两个存储在底层文件系统（COS、云 HDFS）中的目录已经挂载到 GooseFS 中，如果文件已经缓存在 GooseFS 的上，可以通过 gfs:// 这一统一的 schema 访问（例如 hadoop fs ls gfs://BU_A ）；也可以通过各个远端文件系统的命名空间访问（例如 hadoop fs ls cosn://bucket-1/BU_A ）。
- 如果文件未被缓存在 GooseFS 上，此时通过 gfs:// 这一形式访问则会失败，因为文件未被缓存在本地文件系统中，但仍然可以通过底层存储系统的命名空间访问。

使用统一命名空间能力

您可以通过 ns 操作在 GooseFS 中创建命名空间，并将底层存储系统映射到 GooseFS 上，目前支持的底层存储系统包括 COS、云 HDFS、本地 HDFS 等。创建命名空间的操作与 Linux 文件系统中挂载文件卷盘类似。创建命名空间后，GooseFS 可以为客户端提供一个具有统一访问语义的文件系统。目前 GooseFS 命名空间的操作指令集如下：

注意：

- 建议用户尽量避免在配置中使用永久密钥，采取配置子账号密钥或者临时密钥的方式有助于提升业务安全性。为子账号授权时建议按需授权子账号可执行的操作和资源，避免发生预期外的数据泄露。
- 如果您一定要使用永久密钥，建议对永久密钥的权限范围进行限制，可通过限制永久密钥的可执行操作、资源范围和条件（访问 IP 等），提升使用安全性。

```
$ goosefs ns
Usage: goosefs ns [generic options]
         [create <namespace> <CosN/Chdfs path> <--wPolicy <1-6>> <--rPolicy <1-
5>> [--readonly] [--shared] [--secret fs.cosn.userinfo.secretId=
<***************************>] [--secret fs.cosn.userinfo.secretKey=
<xxxxxxxxxx>] [--attribute fs.ofs.userinfo.appid=1200000000][--attribute
fs.cosn.bucket.region=<ap-xxx>/fs.cosn.bucket.endpoint_suffix=<cos.ap-
xxx.myqcloud.com>]]
         [delete <namespace>]
         [help [<command>]]
         [ls [-r|--sort=option|--timestamp=option]]
         [setPolicy [--wPolicy <1-6>] [--rPolicy <1-5>] <namespace>]
         [setTtl [--action delete|free] <namespace> <time to live>]
         [stat <namespace>]
         [unsetPolicy <namespace>]
         [unsetTtl <namespace>]
```

上述指令集中各项指令的能力简述如下：

指令	说明
create	用于创建命名空间，将一个远端存储系统（UFS）映射到命名空间中；支持在创建命名空间时设置读写缓存策略；需要传入有权限的密钥信息（secretId、secretKey）。
delete	用于删除指定命名空间。
ls	用于列出指定命名空间的详细信息，例如挂载点、UFS 路径、创建时间、缓存策略、TTL 信息等。
setPolicy	用于设置指定命名空间的缓存策略。
setTtl	用于设置指定命名空间的 TTL。
stat	用于提供指定命名空间的描述性信息，例如挂载点、UFS 路径、创建时间、缓存策略、TTL 信息、持久化状态、用户组、ACL、最后一次访问时间、修改时间等。
unsetPolicy	用于重置指定命名空间的缓存策略。

unsetTtl	用于重置指定命名空间的 TTL。

创建和删除命名空间

通过 GooseFS 创建命名空间操作，可以将频繁访问的热数据从远端存储系统缓存到本地高性能存储节点中，为本地计算业务提供高性能的数据访问。如下指令展示了将 COS 中的存储桶 example-bucket、存储桶中的 example-prefix 目录以及云 HDFS 文件系统分别映射到 test_cos、test_cos_prefix 和 test_chdfs 等命名空间下。

```
# 将 COS 存储桶 example-bucket 映射到 test_cos 命名空间中
$ goosefs ns create test_cos cosn://example-bucket-1250000000/ --wPolicy 1 --
rPolicy 1 --secret fs.cosn.userinfo.secretId=*************************** --
secret fs.cosn.userinfo.secretKey=xxxxxxxxxx --attribute
fs.cosn.bucket.region=ap-guangzhou --attribute
fs.cosn.bucket.endpoint_suffix=cos.ap-guangzhou.myqcloud.com

# 将 COS 存储桶 example-bucket 的 example-prefix 目录映射到 test_cos_prefix 命名空间中
$ goosefs ns create test_cos_prefix cosn://example-bucket-1250000000/example-
prefix/ --wPolicy 1 --rPolicy 1 --secret
fs.cosn.userinfo.secretId=*************************** --secret
fs.cosn.userinfo.secretKey=xxxxxxxxxx --attribute fs.cosn.bucket.region=ap-
guangzhou --attribute fs.cosn.bucket.endpoint_suffix=cos.ap-
guangzhou.myqcloud.com

# 将 云HDFS 文件系统 f4ma0l3qabc-Xy3 映射到 test_chdfs 命名空间中
$ goosefs ns create test_chdfs ofs://f4ma0l3qabc-Xy3/ --wPolicy 1 --rPolicy 1 --
attribute fs.ofs.userinfo.appid=1250000000
```

创建成功后，可以通过 goosefs fs ls 指令查看目录详情：

```
$ goosefs fs ls /test_cos
```

对于不需要使用的命名空间，可以通过 delete 指令来删除：

```
$ goosefs ns delete test_cos
Delete the namespace: test_cos
```

设置缓存策略

用户可通过 setPolicy 和 unsetPolicy 设置指定命名空间的缓存策略。设置缓存策略的指令集如下：

```
$goosefs ns setPolicy [--wPolicy <1-6>] [--rPolicy <1-5>] <namespace>
```

其中各项参数的含义如下：

- wPolicy：写缓存策略，支持6种写缓存策略。
- rPolicy：读缓存策略，支持5种读缓存策略。

- namespace：指定的命名空间。

目前 GooseFS 支持的读写缓存策略分别如下：

写缓存策略

策略名字	行为	对应 Write_Type	数据安全性	写效率
MUST_CACHE（1）	数据仅存储在 GooseFS，不会写入远端存储系统中。	MUST_CACHE	不可靠	高
TRY_CACHE（2）	缓存有空间时就写入 GooseFS 中，缓存如果没空间则直接写入到底层存储中。	TRY_CACHE	不可靠	中
CACHE_THROUGH（3）	尽量缓存数据，同时同步写入远端存储系统。	CACHE_THROUGH	可靠	低
THROUGH（4）	数据不存储在 GooseFS，直接写远端存储系统。	THROUGH	可靠	中
ASYNC_THROUGH（5）	数据写入 GooseFS 中，并异步刷新到远端存储系统。	ASYNC_THROUGH	弱可靠	高

说明：

- Write_Type：指用户调用 SDK 或者 API 向 GooseFS 中写入数据时指定的文件缓存策略，对单个文件生效。
- 写缓存策略设置后如果需要变更，需要审慎评估缓存数据的重要性，如果数据重要，建议先保证缓存数据已经持久化，否则缓存数据可能出现丢失情况。例如将写缓存从 `MUST_CACHE` 变更到 `CACHE_THROUGH` 后，如果未调用 `persist` 指令持久化数据，那么即将淘汰的数据无法写入到底层，会出现丢失的情况。

读缓存策略

策略名字	行为	元数据同步	对应 Read_Type	数据一致性	读效率	是否缓存数据
NO_CACHE（1）	不缓存数据，直接从远端存储系统中读数据。	NO	NO_CACHE	强一致	低	否
CACHE（2）	• 元数据访问行为：如果命中缓存时，元数据以 Master 中的为准，不会主动从底层同步元数据。 • 数据流访问行为：数据流的 ReadType 采用 CACHE 策略。	Once	CACHE	弱一致	命中：高 未命中：低	是

CACHE_PROMOTE（3）	• 元数据访问行为：与 CACHE 模式相同。 • 数据流访问行为：数据流的 ReadType 采用 CACHE_PROMOTE 策略。	Once	CACHE_PROMOTE	弱一致	命中：高 未命中：低	是
CACHE_CONSISTENT_PROMOTE（4）	• 元数据行为：每次读取操作前均先同步远端存储系统 UFS 上的元数据，如果 UFS 中不存在，则抛出异常 Not Exists。 • 数据流访问行为：数据流的 ReadType 采用 CACHE_PROMOTE 策略，命中以后，缓存到最热的缓存介质中。	Always	CACHE	强一致	命中：中 未命中：低	是
CACHE_CONSISTENT（5）	• 元数据行为：与 CACHE_CONSISTENT_PROMOTE 相同。 • 数据流访问行为：数据流的 ReadType 采用 CACHE 策略，即 CACHE 命中，不会在不同的介质层中移动数据。	Always	CACHE_PROMOTE	强一致	命中：中 未命中：低	是

说明：
Read_Type：指用户调用 SDK 或者 API 从 GooseFS 中读取数据时指定的文件缓存策略，对单个文件生效。

结合目前大数据的业务实践，我们推荐的读写缓存策略组合主要如下：

写缓存策略	读缓存策略	策略组合表现
CACHE_THROUGH（3）	CACHE_CONSISTENT（5）	缓存和远端存储系统数据强一致。
CACHE_THROUGH（3）	CACHE（2）	写强一致性，读最终一致性。
ASYNC_THROUGH（5）	CACHE_CONSISTENT（5）	写最终一致性，读强一致性。
ASYNC_THROUGH（5）	CACHE（2）	读写最终一致性。
MUST_CACHE（1）	CACHE（2）	只从缓存中读数据。

如下示例展示了将指定命名空间 test_cos 的读写缓存策略分别设置为 CACHE_THROUGH 和 CACHE_CONSISTENT：

```
$ goosefs ns setPolicy --wPolicy 3 --rPolicy 5 test_cos
```

> **注意：**
> 除了在创建命名空间时指定缓存策略，用户还可以通过在读写文件时，针对指定文件设置 ReadType 或者 Write_Type，或者通过 properties 配置文件配置全局缓存策略。多个策略同时存在的时候，优先级为用户自定义优先级 > Namespace 读写策略 > 配置文件的全局缓存策略配置。其中，针对读策略会采用用户自定义 ReadType 和 Namespace 的 DirReadPolicy 的组合生效，即数据流读策略采用用户自定义 ReadType，元数据采用 Namespace 的策略。
>
> 例如，GooseFS 中存在一个 COSN 命名空间，读策略为 CACHE_CONSISTENT；假设在该命名空间中存在一个 test.txt 的文件。客户端读取 test.txt 时，ReadType 指定了 CACHE_PROMOTE。那么整个读取行为就是同步元数据并且 CACHE_PROMOTE。

如果需要重置读写缓存策略，可以通过 unsetPolicy 指令实现，如下策略展示了重置 test_cos 命名空间的读写缓存策略：

```
$ goosefs ns unsetPolicy test_cos
```

设置 TTL

TTL 用于管理缓存在 GooseFS 本地节点的数据，配置 TTL 参数可以让缓存数据在指定的时间后执行指定的操作，例如 delete 或者 free 操作。目前设置 TTL 的操作指令如下：

```
$ goosefs ns setTtl [--action delete|free] <namespace> <time to live>
```

其中各项参数的含义如下：

- action：缓存时间到期后执行的操作，目前支持 delete 和 free 两种操作。delete 操作会删除缓存和 UFS 上的数据，free 操作只会删除缓存上的数据。
- namespace：指定的命名空间。
- time to live：数据缓存时间，单位毫秒。

如下示例展示了将指定命名空间 test_cos 的策略设置为60秒到期后删除：

```
$ goosefs ns setTtl --action free test_cos 60000
```

元数据信息管理

本小节介绍 GooseFS 如何管理元数据，包括元数据的同步、更新等内容。GooseFS 为用户提供了统一命名空间能力，用户可以通过统一的 gfs:// 的路径来访问不同底层存储系统上的文件，只需要指定好底层存储系统的路径即可。我们推荐您使用 GooseFS 作为统一的数据接入层，统一从 GooseFS 进行数据读写，保障元数据信息的一致性。

元数据同步概述

您可以通过在 conf/goosefs-site.properties 配置文件中修改元数据同步周期来管理元数据同步周期，配置参数如下：

```
goosefs.user.file.metadata.sync.interval=<INTERVAL>
```

同步周期支持如下3种入参：

- 入参数值为-1：代表元数据在首次加载到 GooseFS 中后就不再进行更新。
- 入参数值为0：代表 GooseFS 会在每次读写操作后都更新元数据。
- 入参数值为正整数：代表 GooseFS 会按照指定的时间间隔，周期性地更新元数据。

您可以综合考虑您的节点数量、GooseFS 集群和底层存储的 I/O 距离、底层存储类型等因素，选择合适的同步周期。通常：

- GooseFS 集群节点数量越多，元数据同步延迟越大。
- GooseFS 集群所处的机房和底层存储的物理距离越远，元数据同步延迟越大。
- 底层存储系统对元数据同步延迟的影响主要视系统请求 QPS 负载情况而定，QPS 负载越高则同步延迟相对更低。

元数据同步管理方式

配置方式

1. 通过命令行配置
   您可以通过命令行的方式设置元数据信息同步周期：

   ```
   goosefs fs ls -R -Dgoosefs.user.file.metadata.sync.interval=0 <path to sync>
   ```

2. 通过配置文件配置
   对于大规模的 Goosefs 集群而言，您可以通过 goosefs-site.properties 配置文件批量配置集群中 Master 节点的元数据信息同步周期，其他节点的同步周期会默认按照该周期值执行。

   ```
   goosefs.user.file.metadata.sync.interval=1m
   ```

> ⚠ **注意：**
> 很多业务会选择按照目录来区分数据的用途，不同目录的数据访问频次并不全部相同。元数据同步周期可以按照不同目录设置不同的周期值，对于一些经常变化的目录，可以将该目录的元数据同步周期设置为较短的时间（如5分钟），对于极少变化或者不变化的目录，可以将同步周期设置为-1，这样 GooseFS 不会自动同步该目录的元数据。

推荐配置

根据业务访问模式的差异，您可以设置不同的元数据同步周期：

访问模式		元数据同步周期	说明
所有文件请求都经过 GooseFS		-1	-
绝大部分文件请求都经过 GooseFS	使用 HDFS 作为 UFS	推荐使用热更新或者按路径更新	如果 HDFS 的更新特别频繁，推荐将更

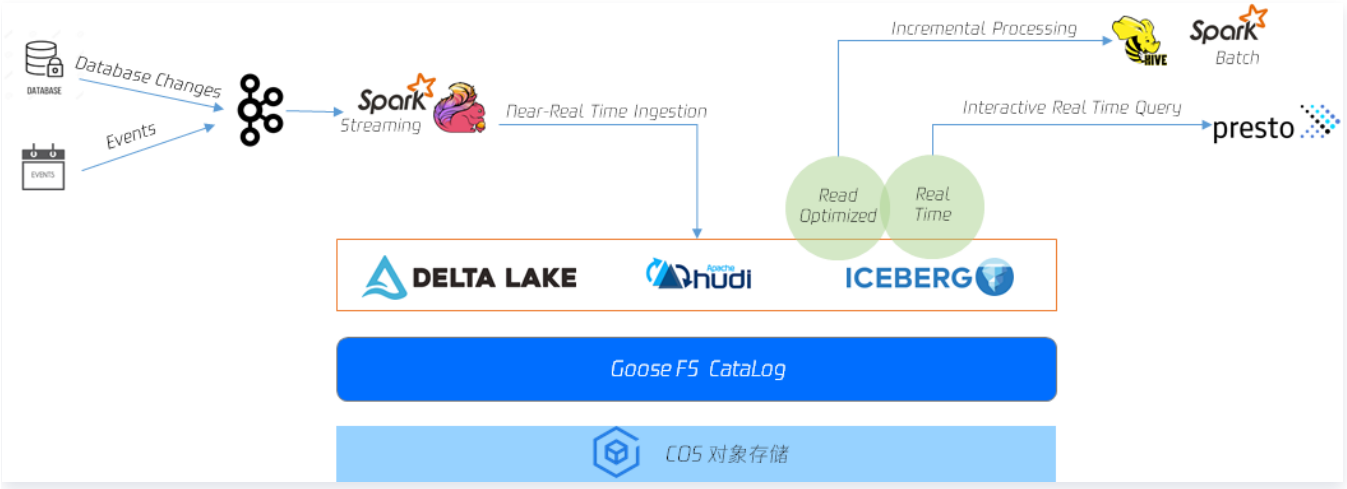
			新周期设置为-1，禁止更新
上传文件请求一般不经过 GooseFS	使用 COS 作为 UFS	推荐按照路径配置更新周期	按照不同目录配置不同的更新周期，可以缓解元数据同步的压力
	使用 HDFS 作为 UFS	推荐按照路径配置更新周期	
	使用 COS 作为 UFS	推荐按照路径配置更新周期	

Table 管理能力

最近更新时间：2023-09-20 20:30:52

Table 管理能力概述

GooseFS Table 管理能力用于管理结构化数据，为 SparkSQL、Hive、Presto 等上层计算应用提供数据库表管理能力，目前底层支持对接 Hive MetaStore。Table 管理能力能够帮助各类 SQL 引擎读取指定的数据内容，能够有效提升大数据场景下对数据的访问效率。



GooseFS Table 管理能力目前主要支持了以下特性：

- 元数据层面的描述能力。GooseFS Catalog 提供源自远程元数据服务（Hive MetaStore）的元数据缓存服务，针对 SparkSQL，Hive，SQL Presto 等 SQL 引擎做查询时，可以根据 GooseFS Catalog 中的元数据缓存服务来确定读取数据大小、目标数据位置以及数据结构，具备与 Hive MetaStore 相同的能力表现。
- 表级数据预缓存能力。GooseFS Catalog 能够感知数据表和数据存储路径的对应关系，进而可以提供 Table 级别以及 Table Partition 级别的缓存预热能力，帮助用户提前按照表结构缓存数据，极大提高访问性能。
- 跨存储服务的统一元数据服务。通过 GooseFS Catalog 运行上层计算应用，可以同时对不同的底层存储系统提供访问加速能力。同时 GooseFS Catalog 可以提供跨越存储服务的统一元数据查询能力，只需要一个 GooseFS 客户端开启 Catalog 功能，即可查询不同存储系统，例如 HDFS、COS、CHDFS 中的数据。

使用 GooseFS Table 管理能力

GooseFS Table 管理能力通过 goosefs table 指令集实现，提供了 DB 的绑定和解绑、查询 DB 信息、查询表信息、数据加载、数据淘汰等能力。GooseFS Table 管理指令集如下所示：

```
$ goosefs table
Usage: goosefs table [generic options]
     [attachdb [-o|--option <key=value>] [--db <goosefs db name>] [--ignore-sync-
errors] <udb type> <udb connection uri> <udb db name>]
     [detachdb <db name>]
     [free <dbName> <tableName> [-p|--partition <partitionSpec>]]
     [load <dbName> <tableName> [-g|--greedy] [--replication <num>] [-p|--
partition <partitionSpec>]]
```

```
      [ls [<db name> [<table name>]]]
      [stat <dbName> <tableName>]
      [sync <db name>]
```

上述指令集中各项指令的能力简述如下：

- attachdb：挂载数据库，将一个远端数据库绑定到 GooseFS 上，目前仅支持 Hive MetaStore。
- detachdb：卸载数据库，将 GooseFS 上绑定的数据库解绑。
- free：清除指定 DB.Table 的数据缓存，可支持 Partition 粒度。
- load：缓存指定 DB.Table 的数据，可支持 partition 粒度，支持通过 replication 设置缓存的副本数。
- ls：列出指定 DB 或 DB.Table 的元数据信息。
- stat：查询指定 DB.Table 的文件数目、总大小、以及缓存百分比。
- sync：同步指定 DB 的内容。
- transform：将指定 DB 关联的 Table 转换为新的 Table。
- transformStatus：Table 转换的进度情况。

1. 挂载 DB

预热指定 Table 数据到 GooseFS 之前，需要将对应的 DB 挂载到 GooseFS 上。如下指令展示了将指定地址 metastore_host:port 中的数据库 goosefs_db_demo 挂载到 GooseFS 中，并将该 DB 在 GooseFS 中命名为 test_db：

```
$ goosefs table attachdb --db test_db hive thrift://metastore_host:port
goosefs_db_demo

response of attachdb
```

> **注意**
> metastore_host:port 可以替换为任意合法可连接的 Hive MetaStore 地址。

2. 查看 Table 信息

绑定完数据库后，可以通过 ls 指令查看已挂载的 DB 和 Table 信息，如下指令展示了如何查询 test_db 中的 web_page 表信息：

```
$ goosefs table ls test_db web_page

OWNER: hadoop
DBNAME.TABLENAME: testdb.web_page (
   wp_web_page_sk bigint,
   wp_web_page_id string,
   wp_rec_start_date string,
   wp_rec_end_date string,
   wp_creation_date_sk bigint,
   wp_access_date_sk bigint,
```

```
    wp_autogen_flag string,
    wp_customer_sk bigint,
    wp_url string,
    wp_type string,
    wp_char_count int,
    wp_link_count int,
    wp_image_count int,
    wp_max_ad_count int,
)
PARTITIONED BY (
)
LOCATION (
    gfs://metastore_host:port/myiNamespace/3000/web_page
)
PARTITION LIST (
    {
    partitionName: web_page
    location: gfs://metastore_host:port/myNamespace/3000/web_page
    }
)
```

3. 预热 Table 中的数据

预热 Table 的指令下发后会在后台发起一个异步作业，GooseFS 会在启动作业后返回一个作业 ID，可以通过 `job stat <ID>` 指令查询任务的运行状态，同时可以通过 `table stat` 指令查看预热百分比。预热指令如下：

```
$ goosefs table load test_db web_page
Asynchronous job submitted successfully, jobId: 1615966078836
```

4. 查看 Table 预热情况

通过 `job stat` 指令可以查看预热 Table 作业的执行进度。当状态为 COMPLETED 时，整个预热过程完成，如果状态为 FAILED，可以在 master.log 文件中查看日志记录，排查预热错误的原因：

```
$ goosefs job stat 1615966078836
COMPLETED
```

当 Table 完成预热后，可以通过 stat 指令查看指定 Table 的概况。

```
$ goosefs table stat test_db web_page
detail
```

5. 释放 Table

通过以下指令可以从 GooseFS 中释放指定 Table 数据缓存：

```
$ goosefs table free test_db web_page
detail
```

6. 卸载 DB

通过以下指令可以从 GooseFS 中卸载指定 DB:

```
$ goosefs table detachdb test_db
detail
```

GooseFS-FUSE 能力

最近更新时间：2024-08-12 14:14:11

GooseFS-FUSE 可以在一台 Unix 机器上的本地文件系统中挂载一个 GooseFS 分布式文件系统。通过使用该特性，一些标准的命令行工具指令（例如 ls、cat 以及 echo）可以直接访问 GooseFS 分布式文件系统中的数据。此外更重要的是使用不同语言实现的应用程序，例如 C、C++、Python、Ruby、Perl、Java 都可以通过标准的 POSIX 接口（例如 open、write、read）来读写 GooseFS，而不需要任何 GooseFS 的客户端整合与设置。

GooseFS-FUSE 是基于 FUSE 这个项目，并且都支持大多数的文件系统操作。但是由于 GooseFS 固有的属性，例如它的一次性、不可改变的文件数据模型，该挂载的文件系统与 POSIX 标准不完全一致，尚有一定的局限性。因此，请先阅读 局限性，从而了解该特性的作用以及局限。

局限性

目前，GooseFS-FUSE 支持大多数基本文件系统的操作。然而，由于 GooseFS 某些内在的特性，您需要注意以下几点：

- 不支持对文件进行随机写入和追加写入操作。
- 文件只能顺序地写入一次，并且无法修改。这意味着如果要修改一个文件，您需要先删除该文件，或者 open 操作时，携带 O_TRUNC 标志符，将文件长度置为0。
- 不支持读取挂载点中正在写入的文件。
- 不支持对文件长度进行 truncate 操作。
- 不支持 soft/hard link；GooseFS 没有 hard-link 和 soft-link 的概念，所以不支持与之相关的命令，例如 ln。此外关于 hard-link 的信息也不在 ll 的输出中显示。
- 以对象存储作为底层存储时，Rename 操作非原子。
- 只有当 GooseFS 的 GooseFS.security.group.mapping.class 选项设置为 ShellBasedUnixGroupsMapping 的值时，文件的用户与分组信息才与 Unix 系统的用户分组对应。否则 chown 与 chgrp 的操作不生效，而 ll 返回的用户与分组为启动 GooseFS-FUSE 进程的用户与分组信息。

性能考虑

由于 FUSE 和 JNR 的配合使用，与直接使用原生文件系统 API 相比，使用挂载文件系统的性能会相对较差。

大多数性能问题的原因在于，每次进行 read 或 write 操作时，内存中都存在若干个副本，并且 FUSE 将写操作的最大粒度设置为128KB。其性能可以利用 kernel 3.15 引入的 FUSE 回写（write-backs）缓存策略从而得到大幅提高（但 libfuse 2.x 用户空间库目前尚不支持该特性）。

安装要求

- JDK 1.8及以上
- Linux 系统：libfuse 2.9.3 及以上（可以使用 2.8.3 版本，但会提示一些警告）
- MAC 系统：osxfuse 3.7.1 及以上

版权所有：腾讯云计算（北京）有限责任公司

可选配置

GooseFS-FUSE 基于标准的 GooseFS-core-client-fs 进行操作。如果您希望它像使用其他应用的 client 一样，自定义该 GooseFS-core-client-fs 的行为。可通过编辑 $GOOSEFS_HOME/conf/goosefs-site.properties 配置文件来更改客户端选项。

> ⚠ **注意：**
> 所有的更改应该在 GooseFS-FUSE 启动之前完成。

GooseFS-FUSE 配置参数

以下是 GooseFS-FUSE 相关的配置参数：

参数	默认值	描述
goosefs.fuse.cached.paths.max	500	用于定义 GooseFS-FUSE 缓存路径的上限；路径越多，缓存的命中率越高。
goosefs.fuse.debug.enabled	false	允许 FUSE 调试输出，该输出会被重定向到 `goosefs.logs.dir` 指定目录中的 `fuse.out` 日志文件。
goosefs.fuse.fs.name	goosefs-fuse	FUSE 挂载文件系统使用的描述性名称。
goosefs.fuse.jnifuse.enabled	true	使用 JNI-Fuse 库以获得更好的性能。 如果禁用，将使用 JNR-Fuse。
goosefs.fuse.shared.caching.reader.enabled	false	（实验性）使用共享 grpc 数据读取器，通过 GooseFS JNI Fuse 在多进程文件读取中获得更好的性能。块数据将缓存在客户端，因此 Fuse 进程需要更多内存。
goosefs.fuse.logging.threshold	10s	当IO 延迟超过这一阈值时，FUSE 会记录 API 调用情况。
goosefs.fuse.maxwrite.bytes	131072	FUSE 写操作的粒度（bytes），注意目前 128KB 是 Linux 内核限制的上界。
goosefs.fuse.user.group.translation.enabled	false	是否在 FUSE API 中将 GooseFS 的用户与组转化为对应的 Unix 用户与组。当设为 false 时，所有 FUSE 文件的用户与组将会显示为挂载 goosefs-fuse 线程的用户与组。

用法

挂载 GooseFS-FUSE

完成配置以及启动 GooseFS 集群后，在需要挂载 GooseFS 的节点上启动 Shell，并进入 $GOOSEFS_HOME 目录执行以下命令：

```
$ integration/fuse/bin/goosefs-fuse mount mount_point [GooseFS_path]
```

该命令会启动一个后台 Java 进程，用于将对应的 GooseFS 路径挂载到 `<mount_point>` 指定的路径。例如以下命令，将 GooseFS 路径 /people 挂载到本地文件系统的 /mnt/people 目录下。

```
$ integration/fuse/bin/goosefs-fuse mount /mnt/people /people
Starting goosefs-fuse on local host.
goosefs-fuse mounted at /mnt/people. See /lib/GooseFS/logs/fuse.log for logs
```

当 GooseFS_path 没有给定时，GooseFS-FUSE 会默认挂载到 GooseFS 根目录下(/)。您可以多次调用该命令，将 GooseFS 挂载到不同的本地目录下。所有的 GooseFS-FUSE 会共享 $GOOSEFS_HOME\logs\fuse.log 日志文件。该日志文件对于错误排查很有帮助。

> **注意：**
> `<mount_point>` 必须是本地文件系统中的一个空文件夹，并且启动 GooseFS-FUSE 进程的用户拥有该挂载点及对其的读写权限。

卸载 GooseFS-FUSE

卸载 GooseFS-FUSE 时，需在该节点上启动 Shell，并进入 $GOOSEFS_HOME 目录执行以下命令：

```
$ integration/fuse/bin/goosefs-fuse umount mount_point
```

该命令将终止 goosefs-fuse java 后台进程，并卸载该文件系统。例如：

```
$ integration/fuse/bin/goosefs-fuse umount /mnt/people
Unmount fuse at /mnt/people (PID: 97626).
```

默认情况下，如果有任何读写操作未完成，unmount 操作会等待最多120s。如果120s后读写操作仍未完成，那么 Fuse 进程会被强行结束，这会导致正在读写的文件失败，您可以添加 -s 参数来避免 Fuse 进程被强行结束。例如：

```
$ ${GOOSEFS_HOME}/integration/fuse/bin/goosefs-fuse unmount -s /mnt/people
```

检查 GooseFS-FUSE 是否在运行

罗列所有的挂载点，需在该节点上启动 Shell，并进入 $GOOSEFS_HOME 目录执行以下命令：

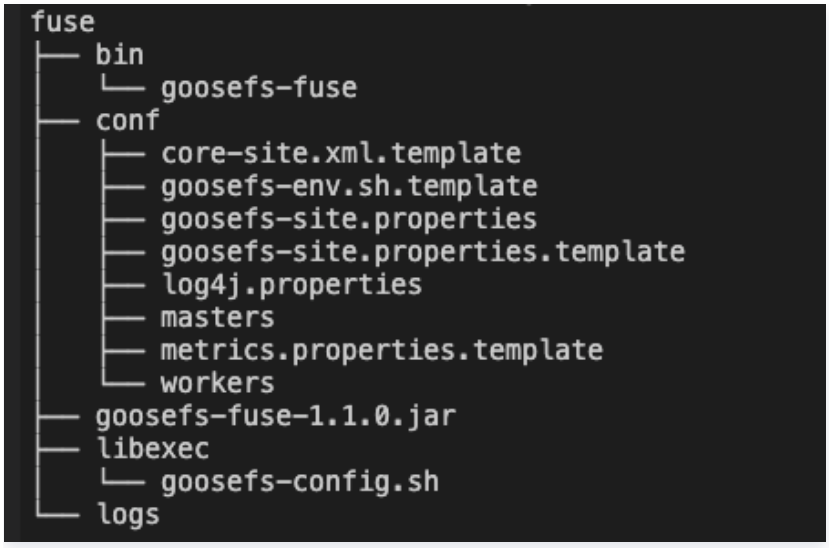
```
$ integration/fuse/bin/goosefs-fuse stat
```

该命令会输出包括 pid、mount_point、GooseFS_path 在内的信息。

例如输出可以是以下格式：

```
$ pid    mount_point     GooseFS_path
80846  /mnt/people     /people
80847  /mnt/sales       /sales
```

Goosefs-FUSE 目录结构



conf 目录下：

- masters：master 服务器的 IP 配置文件
- workers：worker 服务器的 IP 配置文件
- goosefs-site.properties：goosefs 配置文件
- libexec：goosefs-fuse 运行依赖的 lib 库文件
- goosefs-fuse-1.4.2：goosefs-fuse 后台运行的 jar 包
- log：日志目录

常见问题

缺少 libfuse 库文件

在您执行 GooseFS-Fuse 挂载之前，需要安装 libfuse。

```
2021-10-13 20:04:42,970 INFO  TieredIdentityFactory - Initialized tiered identity TieredIdentity(node=10.91.27.82, rack=null)
2021-10-13 20:04:43,560 ERROR GooseFSFuse - launch fuse failed:
java.io.IOException: Failed to mount GooseFS file system
        at com.qcloud.cos.goosefs.fuse.GooseFSFuse.launchFuse(GooseFSFuse.java:192)
        at com.qcloud.cos.goosefs.fuse.GooseFSFuse.main(GooseFSFuse.java:126)
Caused by: java.lang.UnsatisfiedLinkError: /private/var/folders/5_/sp1cwpdd1xn9w96x5xkw7qnc0000gn/T/libjnifuse957879065968834264.jnilib: dlopen(/private/var/fo
5_/sp1cwpdd1xn9w96x5xkw7qnc0000gn/T/libjnifuse957879065968834264.jnilib, 1): Library not loaded: /usr/local/lib/libfuse.2.dylib
  Referenced from: /private/var/folders/5_/sp1cwpdd1xn9w96x5xkw7qnc0000gn/T/libjnifuse957879065968834264.jnilib
  Reason: image not found
        at java.lang.ClassLoader$NativeLibrary.load(Native Method)
        at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1934)
        at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1817)
        at java.lang.Runtime.load0(Runtime.java:810)
        at java.lang.System.load(System.java:1086)
        at com.qcloud.cos.goosefs.jnifuse.utils.NativeLibraryLoader.loadLibraryFromJar(NativeLibraryLoader.java:105)
        at com.qcloud.cos.goosefs.jnifuse.utils.NativeLibraryLoader.loadLibrary(NativeLibraryLoader.java:83)
        at com.qcloud.cos.goosefs.jnifuse.LibFuse.loadLibrary(LibFuse.java:48)
        at com.qcloud.cos.goosefs.jnifuse.LibFuse.<clinit>(LibFuse.java:32)
        at com.qcloud.cos.goosefs.jnifuse.AbstractFuseFileSystem.<clinit>(AbstractFuseFileSystem.java:41)
        at com.qcloud.cos.goosefs.fuse.GooseFSFuse.launchFuse(GooseFSFuse.java:154)
        ... 1 more
```

- **方式一**

  安装命令：

```
yum install fuse-devel
```

查看是否安装成功：

```
find / -name libfuse.so*
```

- **方式二**

  更新旧版本 libfuse.so.2.9.2，安装步骤如下：

> **说明：**
> 在 CentOS 7安装 libfuse，CentOS 7默认安装的是 libfuse.so.2.9.2。

首先，下载 libfuse 源码，并编译生成 libfuse.so.2.9.7。

```
tar -zxvf fuse-2.9.7.tar.gz
cd fuse-2.9.7/ && ./configure && make && make install
echo -e '\n/usr/local/lib' >> /etc/ld.so.conf
ldconfig
```

其次，下载、编译及生成 libfuse.so.2.9.7 后，然后按照以下步骤进行安装替换。

1. 执行以下命令，查找旧版本 libfuse.so.2.9.2 库链接。

```
find / -name libfuse.so*
```

2. 执行以下命令，将 libfuse.so.2.9.7 拷贝至旧版本库 libfuse.so.2.9.2 所在位置。

```
cp /usr/local/lib/libfuse.so.2.9.7 /usr/lib64/
```

3. 执行以下命令， 删除旧版本 libfuse.so 库的所有链接。

```
rm -f /usr/lib64/libfuse.so
rm -f /usr/lib64/libfuse.so.2
```

4. 执行以下命令，建立与被删除旧版本链接类似的 libfuse.so.2.9.7 库链接。

```
ln -s /usr/lib64/libfuse.so.2.9.7 /usr/lib64/libfuse.so
ln -s /usr/lib64/libfuse.so.2.9.7 /usr/lib64/libfuse.so.2
```

VIM 编辑挂载点中的文件报错 Write error in swap file?

您可以通过更改 VIM 的配置，使用 VIM 7.4 及之前的版本，对 GooseFS-Fuse 挂载点中的文件进行编辑。VIM 的 swap 文件用于暂存用户对文件的修改内容，方便 VIM 在机器宕掉重启后，可以根据 swap 文件，对未保存的修改内容进行恢复。上面的报错是由于 VIM 对 swap 文件，涉及随机写入操作，而 GooseFS 不支持对文件的随机写入。解决方法：您可以通过在 VIM 编辑窗口中，执行如下的 VIM 命令关闭 swap 文件生成， `:set noswapfile` ，或者在配置文件 ~/.vimrc 中添加配置行 `set noswapfile` 。

开发者指南
命令行指南
指令概览

最近更新时间：2024-10-12 10:20:41

GooseFS 命令行接口为用户提供 GooseFS 的命名空间、缓存管理、文件读写等操作能力，可以通过如下指令获取所有子命令：

```
Usage: GooseFS COMMAND [GENERIC_COMMAND_OPTIONS] [COMMAND_ARGS]
```

指令总览如下：

指令名	指令作用
extensions	用于管理 GooseFS 的扩展组件。
format [-s]	用于格式化所有 GooseFS 的节点，包括 Master 和 Worker 节点；该操作会清理所有缓存和元数据信息，需要谨慎操作。如果指定了 `-s` ，仅格式化不存在的 UFS 存储路径。
formatMaster	格式化所有 GooseFS Master 节点。
formatWorker	格式化所有 GooseFS Worker 节点。
bootstrapConf	快速生成 GooseFS 配置文件。
ns	GooseFS 命名空间指令行工具。
fs	GooseFS 文件系统指令行工具。
fsadmin	GooseFS 文件系统管理员指令行工具。
table	GooseFS 库表操作指令行工具。
getConf [k]	用于快速获取指定 GooseFS 配置文件的内容。
job	GooseFS 任务服务指令行工具。
logLevle	用于设置或者获取 GooseFS 日志级别。
runClass	用于运行 GooseFS 类的主方法。
runTest	用于运行指定 GooseFS 集群端到端的测试用例集。
runTests	用于运行所有 GooseFS 集群端到端的测试用例集。
runHmsTest	用于运行 GooseFS 和 Hive 的集成测试用例集。

runHdfsMountTests	用于运行 GooseFS 和 HDFS 的集成测试用例集。
runJournalCrashTest	用于运行 GooseFS Master 的故障注入用例集。
runUfsIOTest	用于运行 GooseFS 集成 UFS 的数据吞吐用例集。
runUfsTests	用于运行 GooseFS 和 UFS 的集成测试用例集。
readJournal	用于读取 GooseFS Journal 文件。
upgradeJournal	用于更新 GooseFS Journal 版本。
killAll [-s] <WORD>	用于中断 GooseFS 指定进程，进程名称通过 WORD 参数指定。
copyDir <PATH>	用于将指定路径拷贝到所有 Master/Worker 节点，路径通过 PATH 参数指定
clearCache	用于清理机器节点上的所有系统缓冲区（OS buffer cache）。
docGen	用户自动生成 GooseFS 指导文档。
version	用于打印 GooseFS 版本信息。
validateConf	用于校验 GooseFS 配置信息。
validateEnv	用于校验 GooseFS 环境信息。
collectInfo	用于收集 GooseFS 集群信息。
collectLogs	用于收集 Master 和 Worker 节点日志到指定路径，路径通过 PATH 参数制定。

版权所有：腾讯云计算（北京）有限责任公司

命名空间指令介绍

最近更新时间：2024-11-15 21:51:43

GooseFS 命名空间是 GooseFS 缓存文件的基本管理单元，一个命名空间一般对应一个 UFS 上的一个空间或者一个路径，例如，对象存储 COS 上某个指定存储桶或者其指定目录。可以通过如下指令获取命名空间所有指令列表：

```
goosefs ns [generic options]
```

命名空间各项指令说明如下表：

指令操作	指令说明
cacheRestrict -apply <config>	用于为指定命名空间应用指定的缓存策略配置。
cacheRestrictList [-all] [<namespace>]	用于为指定命名空间设置所有缓存策略配置。
create <namespace> <CosN/Chdfs path> <--wPolicy <1-6>> <--rPolicy <1-3>> [--readonly] [--shared] [--secret fs.cosn.userinfo.secretId=<**********>] [--secret fs.cosn.userinfo.secretKey=**********] [--attribute fs.ofs.userinfo.appid=1200000000][--attribute fs.cosn.bucket.region=<ap-xxx>/fs.cosn.bucket.endpoint_suffix=<cos.ap-xxx.myqcloud.com>]	用于创建指定名称的命名空间。
delete <namespace>	用于删除指定名称的命名空间。
help [<command>]	获取命名空间指令帮助。
ls [-r|--sort=option|--timestamp=option]	用于按照筛选条件，列出当前集群上的命名空间列表。
setPolicy [--wPolicy <1-6>] [--rPolicy <1-3>] <namespace>	用于设置指定命名空间的缓存策略。
setTtl [--action delete_only_goosefs|delete|free] <namespace> <time to live>	用于设置指定命名空间的 TTL 策略。
stat <namespace>	用于统计指定命名空间的信息。
unsetPolicy <namespace>	用于重置指定命名空间的缓存策略。
unsetTtl <namespace>	用于重置指定命名空间的 TTL 策略。
update <namespace> [--readonly] [--shared] [--secret <key=value>] [--attribute <key=value>]	用于更新指定命名空间的属性。

版权所有：腾讯云计算（北京）有限责任公司

文件系统指令介绍

最近更新时间：2024-10-12 10:20:41

GooseFS 命令行接口为用户提供了基本的文件系统操作，可以使用以下命令来得到所有子命令：

```
goosefs fs [generic options]
```

文件系统各项指令列表如下：

指令操作	指令说明
cat <path>	打印指定的 GooseFS 路径中的文件内容。
checkConsistency [-r][-t|--threads] <GooseFS path>	检查 GooseFS 和底层存储的元数据一致性。
checksum <GooseFS path>	计算指定 GooseFS 文件路径的 md5 校验码。
chgrp [-R]<group><path>	修改指定 GooseFS 文件或者文件夹的所属用户组信息，支持通过 -R 递归修改指定目录下所有文件的所属用户组。
chmod [-R]<mode><path>	修改指定 GooseFS 文件或者文件夹的访问权限信息，支持通过 -R 递归修改指定目录下所有文件的访问权限。
chown [-R]<owner>[:<group>]<path>	修改指定 GooseFS 文件或者文件夹的所有者信息，支持通过 -R 递归修改指定目录下所有文件的所有者。
copyFromLocal [--thread <num>] [--buffersize <bytes>] <src> <remoteDst>	将指定的本地路径拷贝到指定的 GooseFS 文件路径上，可以设置该指令的并发数和缓冲区大小，调整拷贝速率。
copyToLocal [--buffersize <bytes>] <src> <localDst>	将指定的 GooseFS 文件路径拷贝到指定的本地路径，可以设置缓冲区大小。
count [-h]<path>	统计指定 GooseFS 路径的文件及文件夹总数。
cp [-R][--buffersize] <src> <dst>	将指定的 GooseFS 路径拷贝到另一个指定的 GooseFS 路径上，支持递归拷贝，支持调整缓冲区大小。
decompress [-a]<path>[-C <targetDir>]	将指定 GooseFS 路径上的文件解压缩到目标目录下。
distributedCp [--active-jobs <num>] [--batch-size <num>] <src> <dst>	分布式拷贝数据，将指定的 GooseFS 路径并发拷贝到另一个指定的 GooseFS 路径上，支持设置作业数和每批次的拷贝数量。

distributedLoad [-ignoreCap][-dryrun][-A][--replication][--active-jobs <num>] [--batch-size <num>] [--expire-time] <path>	分布式预热数据，将指定 GooseFS 路径的数据从底层存储中预热到缓存集群中，支持设置缓存副本数、作业数和每批次的拷贝数等参数。
distributedMv <src><dst>	分布式移动数据，将指定的 GooseFS 路径并发移动到另一个指定的 GooseFS 路径上。
du [-h|-s|-g|-m]<path>	输出指定 GooseFS 文件/文件夹的大小。
free [-f]<path>	释放指定 GooseFS 文件/文件夹的缓存数据，该操作不会删除底层存储中的数据。
getCapacityBytes	获取 GooseFS 的集群总容量大小。
getSyncPathList	获取当前活跃的自动同步进程列表。
getUsedBytes	获取 GooseFS 已缓存的容量大小。
getfacl <path>	获取指定 GooseFS 文件路径的 Posix acl。
head [-c <bytes>]<path>	输出指定 GooseFS 文件开头的指定长度内容，默认输出开头的1KB内容。
help [<command>]	获取指定文件系统指令的帮助信息。
leader	获取当前 GooseFS 集群的主 Master 节点信息。
listDecompressJobs <namespace>[-f <Running|Success|Failed|Pending>][-s <asc|desc>][-n <resultNums>][-p <pageIndex>]	列出指定 GooseFS 命名空间下正在解压缩的任务列表，支持状态、升降序、每页打印的任务数以及任务索引等参数。
load [--local] <path>	预热数据，将指定 GooseFS 路径的数据从底层存储加载到集群中。
loadMetadata [-R][-F] <path>	预热元数据，将指定 GooseFS 路径的元数据信息加载到集群中，如果路径为目录，支持递归加载。
location <path>	输出包含指定 GooseFS文件的节点列表。
ls [-d|-f|-p|-R|-h|--sort=option|--timestamp=option|-r|-a] <path> ...	输出指定 GooseFS 路径下的所有文件和目录信息，支持递归列出等操作。
masterInfo	输出 GooseFS Master 节点的容灾信息，例如 leader 节点信息、所有 master 节点列表以及配置的 zookeeper 地址（如有配置）等。
mkdir <path1>[path2]...	在制定的路径下创建文件夹，多个路径用空格或者 tab 键分隔；如果其中有任意路

[pathn]	径已存在，那么该操作会抛出异常。
mount [--readonly] [--shared] [--option <key=val>] <goosefsPath> <ufsURI>	将指定的底层存储路径挂载到 GooseFS 命名空间的指定路径下。如果指定路径已存在，那么挂载操作会抛出异常。挂载成功后，对该挂载点下的文件操作会同时作用于底层存储上的对应路径。
mv <src><dst>	将指定的 GooseFS 路径并发移动到另一个指定的 GooseFS 路径上。如果目标路径已存在，该操作会抛出异常。
persist [-p|--parallelism <#>] [-t|--timeout <milliseconds>] [-w|--wait <milliseconds>] <path> [<path> ...]	将仅存在于 GooseFS 上的文件或者文件夹持久化到底层存储上，支持设置并发度、超时时间等参数。
pin <path> media1 media2 media3 ...	将指定 GooseFS 上的文件或者文件夹锁定到 GooseFS 集群中，避免被淘汰。如果指定路径为文件夹则默认递归锁定文件夹下所有文件。
queryDecompress <path> [<jobId>]	查询指定 GooseFS 文件路径、指定任务id的解压缩任务。
rm [-R][-U][--goosefsOnly] <path>	删除指定 GooseFS 文件路径的文件，支持递归删除，支持设定是否只删除缓存中的文件。
setReplication [--max <num> | --min <num>] <path>	设置指定 GooseFS 上文件路径的副本数，支持设定副本数的最大值和最小值。
setTtl [--action delete_only_goosefs|delete|free] <path> <time to live>	设置指定 GooseFS 上文件路径的TTL，支持设置到期删除操作类型，包括删除底层存储文件，删除本地文件，释放缓存。
setfacl [-d][-R][--set | -m | -x <acl_entries> <path>] | [-b | -k <path>]	设置指定GooseFS上文件路径的权限信息，支持递归设置。
startSync <path>	启动指定 GooseFS 文件路径的自动同步进程。
stat [-f <format>]<path>	统计指定 GooseFS 路径的文件信息。
stopSync <path>	停止指定 GooseFS 文件路径的自动同步进程。
tail [-c <bytes>]<path>	输出指定 GooseFS 文件末尾的指定长度内容，默认输出最后的1KB内容。
test [-d|-f|-e|-s|-z]<path>	测试指定 GooseFS 路径的属性，如果属性正确返回0，否则返回1。
touch <path>	在指定的 GooseFS 路径上创建一个空文件。
unmount <goosefsPath>	解除指定的文件挂载点。

版权所有：腾讯云计算（北京）有限责任公司

unpin <path>	将指定 GooseFS 上的文件或者文件夹解锁，解锁后可以被集群淘汰。如果指定路径为文件夹则默认递归解锁文件夹下所有文件。该操作为 pin 操作的反向操作。
unsetTtl <path>	删除指定 GooseFS 文件路径的TTL信息。
updateMount [--readonly] [--shared] [--option <key=val>] <goosefsPath>	更新指定 GooseFS 文件路径的挂载信息。

管理员操作指令介绍

最近更新时间：2024-10-12 10:20:41

GooseFS 命令行接口为用户提供了基本的文件系统管理员操作，可以使用以下命令来得到所有子命令：

```
goosefs fsadmin [generic options]
```

文件系统管理员各项指令列表如下：

指令操作	指令说明
backup [directory] [--local]	备份 GooseFS 元数据信息到 Master 节点上注册的 directory 路径上。
doctor [category]	展示 GooseFS 报错和警告信息。
getBlockInfo [blockId]	获取指定 BlockId 的信息，如所处的节点信息等。
help [<command>]	获取 fsadmin 参数的帮助信息。
journal [checkpoint] [quorum]	主动触发 Master 节点 Checkpoint，支持通过 quorum 修改集群的选主配置。
metrics [clear]	提供 GooseFS 集群监控指标，支持清理 GooseFS 集群监控指标，支持指定 master 节点和 worker 节点清理。
pathConf [add] [show] [list] [remove]	添加、展示、删除或者动态变更默认配置项。
report [category] [category args]	输出 GooseFS 集群的运行状态信息。
statelock	提供有关 GooseFS statelock 的 waiter 和 holder 的信息，用于诊断耗时较长的请求。
ufs [--mode <noAccess/readOnly/readWrite>] <ufsPath>	更新 GooseFS 集群中指定底层存储路径的挂载点信息。

版权所有：腾讯云计算（北京）有限责任公司

客户端工具
GooseFS-FUSE 工具

最近更新时间：2023-07-11 16:09:11

GooseFS-FUSE 可以在一台 Unix 机器上的本地文件系统中挂载一个 GooseFS 分布式文件系统。通过使用该特性，一些标准的命令行工具（例如 ls、cat 以及 echo）可以直接访问 GooseFS 分布式文件系统中的数据。此外更重要的是使用不同语言实现的应用程序，例如 C、C++、Python、Ruby、Perl、Java 都可以通过标准的 POSIX 接口（例如 open、write、read）来读写 GooseFS，而不需要任何 GooseFS 的客户端整合与设置。

GooseFS-FUSE 是基于 FUSE 这个项目，并且都支持大多数的文件系统操作。但是由于 GooseFS 固有的属性，例如它的一次性、不可改变的文件数据模型，该挂载的文件系统与 POSIX 标准不完全一致，尚有一定的局限性。因此，请先阅读 局限性，从而了解该特性的作用以及局限。

安装要求

- JDK 1.8及以上
- Linux 系统：libfuse 2.9.3及以上（可以使用 2.8.3版本，但会提示一些警告）
- MAC 系统：osxfuse 3.7.1及以上

用法

挂载 GooseFS-FUSE

完成配置以及启动 GooseFS 集群后，在需要挂载 GooseFS 的节点上启动 Shell，并进入 $GOOSEFS_HOME 目录执行以下命令：

```
$ integration/fuse/bin/goosefs-fuse mount mount_point [GooseFS_path]
```

该命令会启动一个后台 Java 进程，用于将对应的 GooseFS 路径挂载到 `<mount_point>` 指定的路径。例如以下命令，将 GooseFS 路径 /people 挂载到本地文件系统的 /mnt/people 目录下。

```
$ integration/fuse/bin/goosefs-fuse mount /mnt/people /people
Starting goosefs-fuse on local host.
goosefs-fuse mounted at /mnt/people. See /lib/GooseFS/logs/fuse.log for logs
```

当 GooseFS_path 没有给定时，GooseFS-FUSE 会默认挂载到 GooseFS 根目录下(/)。您可以多次调用该命令，将 GooseFS 挂载到不同的本地目录下。所有的 GooseFS-FUSE 会共享 $GOOSEFS_HOME\logs\fuse.log 日志文件。该日志文件对于错误排查很有帮助。

> **注意**
> `<mount_point>` 必须是本地文件系统中的一个空文件夹，并且启动 GooseFS-FUSE 进程的用户拥有该挂载点及对其的读写权限。

卸载 GooseFS-FUSE

卸载 GooseFS-FUSE 时，需在该节点上启动 Shell，并进入$GOOSEFS_HOME 目录执行以下命令：

```
$ integration/fuse/bin/goosefs-fuse umount mount_point
```

该命令将终止 goosefs-fuse java 后台进程，并卸载该文件系统。例如：

```
$ integration/fuse/bin/goosefs-fuse umount /mnt/people
Unmount fuse at /mnt/people (PID: 97626).
```

默认情况下，如果有任何读写操作未完成，unmount 操作会等待最多120s。如果120s后读写操作仍未完成，那么 Fuse 进程会被强行结束，这会导致正在读写的文件失败，您可以添加 -s 参数来避免 Fuse 进程被强行结束。例如：

```
$ ${GOOSEFS_HOME}/integration/fuse/bin/goosefs-fuse unmount -s /mnt/people
```

检查 GooseFS-FUSE 是否在运行

罗列所有的挂载点，需在该节点上启动 Shell，并进入 $GOOSEFS_HOME 目录执行以下命令：

```
$ integration/fuse/bin/goosefs-fuse stat
```

该命令会输出包括 pid、mount_point、GooseFS_path 在内的信息。
例如输出可以是以下格式：

```
$ pid    mount_point     GooseFS_path
80846  /mnt/people     /people
80847  /mnt/sales       /sales
```

Goosefs-FUSE 目录结构

```
fuse
├── bin
│   └── goosefs-fuse
├── conf
│   ├── core-site.xml.template
│   ├── goosefs-env.sh.template
│   ├── goosefs-site.properties
│   ├── goosefs-site.properties.template
│   ├── log4j.properties
│   ├── masters
│   ├── metrics.properties.template
│   └── workers
├── goosefs-fuse-1.1.0.jar
├── libexec
│   └── goosefs-config.sh
└── logs
```

conf 目录下：

- masters：master 服务器的 IP 配置文件
- workers：worker 服务器的 IP 配置文件
- goosefs-site.properties：goosefs 配置文件
- libexec：goosefs-fuse 运行依赖的 lib 库文件
- goosefs-fuse-1.2.0：goosefs-fuse 后台运行的 jar 包
- log：日志目录

可选配置

GooseFS-FUSE 基于标准的 GooseFS-core-client-fs 进行操作。如果您希望它像使用其他应用的 client 一样，自定义该 GooseFS-core-client-fs 的行为。

可通过编辑 $GOOSEFS_HOME/conf/goosefs-site.properties 配置文件来更改客户端选项。

> **注意**
> 所有的更改应该在 GooseFS-FUSE 启动之前完成。

局限性

目前，GooseFS-FUSE 支持大多数基本文件系统的操作。然而，由于 GooseFS 某些内在的特性，您需要注意以下几点：

- 文件只能顺序地写入一次，并且无法修改。这意味着如果要修改一个文件，您需要先删除该文件，然后再重新创建。例如当目标文件存在时，拷贝命令 cp 会失败。
- GooseFS 没有 hard-link 和 soft-link 的概念，所以不支持与之相关的命令，例如 ln。此外关于 hard-link 的信息也不在 ll 的输出中显示。
- 只有当 GooseFS 的 GooseFS.security.group.mapping.class 选项设置为ShellBasedUnixGroupsMapping 的值时，文件的用户与分组信息才与 Unix 系统的用户分组对应。否则 chown 与 chgrp 的操作不生效，而 ll 返回的用户与分组为启动 GooseFS-FUSE 进程的用户与分组信息。

性能考虑

由于 FUSE 和 JNR 的配合使用，与直接使用原生文件系统 API 相比，使用挂载文件系统的性能会相对较差。

大多数性能问题的原因在于，每次进行 read 或 write 操作时，内存中都存在若干个副本，并且 FUSE 将写操作的最大粒度设置为128KB。其性能可以利用 kernel 3.15 引入的 FUSE 回写（write-backs）缓存策略从而得到大幅提高（但 libfuse 2.x 用户空间库目前尚不支持该特性）。

GooseFS-FUSE 配置参数

以下是 GooseFS-FUSE 相关的配置参数：

参数	默认值	描述
goosefs.fuse.cached.paths.max	500	定义内部 GooseFS-FUSE 缓存的大小，该缓存维护本地文件系统路径和 GooseFS 文件路径之间最常用的转换。
goosefs.fuse.debug.enabled	false	允许 FUSE 调试输出，该输出会被重定向到 `goosefs.logs.dir` 指定目录中的 `fuse.out` 日志文件。
goosefs.fuse.fs.name	goosefs-fuse	FUSE 挂载文件系统使用的描述性名称。
goosefs.fuse.jnifuse.enabled	true	使用 JNI-Fuse 库以获得更好的性能。 如果禁用，将使用 JNR-Fuse。
goosefs.fuse.shared.caching.reader.enabled	false	（实验性）使用共享 grpc 数据读取器，通过 GooseFS JNI Fuse 在多进程文件读取中获得更好的性能。块数据将缓存在客户端，因此 Fuse 进程需要更多内存。
goosefs.fuse.logging.threshold	10s	当花费的时间超过阈值时，记录 FUSE API 调用。
goosefs.fuse.maxwrite.bytes	131072	FUSE 写操作的粒度（bytes），注意目前 128KB 是 Linux 内核限制的上界。
goosefs.fuse.user.group.translation.enabled	false	是否在 FUSE API 中将 GooseFS 的用户与组转化为对应的 Unix 用户与组。当设为 false 时，所有 FUSE 文件的用户与组将会显示为挂载 goosefs-fuse 线程的用户与组。

常见问题

缺少 libfuse 库文件，需要安装 libfuse。

```
2021-10-13 20:04:42,970 INFO  TieredIdentityFactory - Initialized tiered identity TieredIdentity(node=10.91.27.82, rack=null)
2021-10-13 20:04:43,560 ERROR GooseFSFuse - launch fuse failed:
java.io.IOException: Failed to mount GooseFS file system
        at com.qcloud.cos.goosefs.fuse.GooseFSFuse.launchFuse(GooseFSFuse.java:192)
        at com.qcloud.cos.goosefs.fuse.GooseFSFuse.main(GooseFSFuse.java:126)
Caused by: java.lang.UnsatisfiedLinkError: /private/var/folders/5_/sp1cwpdd1xn9w96x5xkw7qnc0000gn/T/libjnifuse957879065968834264.jnilib: dlopen(/private/var/fo
5_/sp1cwpdd1xn9w96x5xkw7qnc0000gn/T/libjnifuse957879065968834264.jnilib, 1): Library not loaded: /usr/local/lib/libfuse.2.dylib
  Referenced from: /private/var/folders/5_/sp1cwpdd1xn9w96x5xkw7qnc0000gn/T/libjnifuse957879065968834264.jnilib
  Reason: image not found
        at java.lang.ClassLoader$NativeLibrary.load(Native Method)
        at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1934)
        at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1817)
        at java.lang.Runtime.load0(Runtime.java:810)
        at java.lang.System.load(System.java:1086)
        at com.qcloud.cos.goosefs.jnifuse.utils.NativeLibraryLoader.loadLibraryFromJar(NativeLibraryLoader.java:105)
        at com.qcloud.cos.goosefs.jnifuse.utils.NativeLibraryLoader.loadLibrary(NativeLibraryLoader.java:83)
        at com.qcloud.cos.goosefs.jnifuse.LibFuse.loadLibrary(LibFuse.java:48)
        at com.qcloud.cos.goosefs.jnifuse.LibFuse.<clinit>(LibFuse.java:32)
        at com.qcloud.cos.goosefs.jnifuse.AbstractFuseFileSystem.<clinit>(AbstractFuseFileSystem.java:41)
        at com.qcloud.cos.goosefs.fuse.GooseFSFuse.launchFuse(GooseFSFuse.java:154)
        ... 1 more
```

- **方式一**

  安装命令：

  ```
  yum install fuse-devel
  ```

  查看是否安装成功：

  ```
  find / -name libfuse.so*
  ```

- **方式二**

  更新旧版本 libfuse.so.2.9.2，安装步骤如下：

  > **说明**
  > 在 CentOS 7安装 libfuse，CentOS 7默认安装的是 libfuse.so.2.9.2。

  首先，下载 libfuse 源码，并编译生成 libfuse.so.2.9.7。

  ```
  tar -zxvf fuse-2.9.7.tar.gz
  cd fuse-2.9.7/ && ./configure && make && make install
  echo -e '\n/usr/local/lib' >> /etc/ld.so.conf
  ldconfig
  ```

  其次，下载、编译及生成 libfuse.so.2.9.7 后，然后按照以下步骤进行安装替换。

  1.1 执行以下命令，查找旧版本 libfuse.so.2.9.2 库链接。

  ```
  find / -name libfuse.so*
  ```

  1.2 执行以下命令，将 libfuse.so.2.9.7拷贝至旧版本库 libfuse.so.2.9.2 所在位置。

版权所有：腾讯云计算（北京）有限责任公司

```
cp /usr/local/lib/libfuse.so.2.9.7 /usr/lib64/
```

1.3 执行以下命令， 删除旧版本 libfuse.so 库的所有链接。

```
rm -f /usr/lib64/libfuse.so
rm -f /usr/lib64/libfuse.so.2
```

1.4 执行以下命令，建立与被删除旧版本链接类似的 libfuse.so.2.9.7 库链接。

```
ln -s /usr/lib64/libfuse.so.2.9.7 /usr/lib64/libfuse.so
ln -s /usr/lib64/libfuse.so.2.9.7 /usr/lib64/libfuse.so.2
```

版权所有：腾讯云计算（北京）有限责任公司

运维指南
监控指南
GooseFS 监控指标

最近更新时间：2023-06-27 15:11:42

监控指标概述

Goosefs 监控指标记录了客户端、集群、Master 节点、 Worker 节点、进程的运行状态。

1. 按照统计维度，监控指标可以分成以下四大类：
   - **Gauge**：记录一个事件的瞬时数值，该数值可增可减，一般用来反映系统运行状态。
   - **Meter**：统计事件频率，即固定时间周期内事件的发生次数，如每分钟、每5分钟，每15分钟内的请求数。
   - **Counter**：统计事件发生的总次数，与 Gauge 不同的点在于该数值只增不减，一般用来记录访问请求次数等数据。
   - **Timer**：统计事件的频率和分布情况，该指标可以认为 Histogram 指标和 Meter 指标的结合，Histogram 指标用于统计耗时分布，Meter 指标用于统计 QPS；如果需要同时统计频率和耗时的时候可以使用该指标。

> ⚠ **注意**
> 更多监控指标分类可参见 Metrics Library 文档。GooseFS 目前提供的监控指标只有上述4种类型。

2. 按照数据采集来源，监控指标可以分为以下两大类：
   - **集群状态指标**：集群状态指标包括了 Cluster、Master 和 Worker 维度的监控数据，这些监控指标可以反应整个集群及集群下的每个节点的运行状态。这类指标的记录格式如下：

```
Master.[metricName].[tag1].[tag2]...
[Worker_ip].[metricName].[tag1].[tag2]...
```

   - **进程处理指标**：进程处理指标用于详细记录集群中的请求次数、延迟、CPU 消耗、RPC 调用次数等信息，这类指标可以推送到第三方监控工具中展示。这类指标的记录格式如下：

```
[processType].[metricName].[tag1].[tag2]...[hostName]
```

Cluster 监控指标

GooseFS 集群化部署时，客户端和 Worker 节点会周期性地通过心跳将监控指标发送到 Master 节点，可以通过 goosefs.master.worker.heartbeat.interval 和 goosefs.user.metrics.heartbeat.interval 分别设置 Worker 节点和客户端监控指标的心跳周期。

Cluster 级别的监控指标列表如下：

指标名称	指标类型	指标描述

Cluster.BytesReadDomain	COUNTER	GooseFS集群Worker节点的总DomainSocket读流量数值。
Cluster.BytesReadDomainThroughput	GAUGE	GooseFS集群Worker节点的总DomainSocket读带宽数值。
Cluster.BytesReadLocal	COUNTER	GooseFS集群客户端节点的总本地短路读流量数值。
Cluster.BytesReadLocalThroughput	GAUGE	GooseFS集群客户端节点的总本地短路读吞吐数值。
Cluster.BytesReadPerUfs	COUNTER	GooseFS集群Worker节点从某个指定的底层存储读数据的流量数值。
Cluster.BytesReadRemote	COUNTER	GooseFS集群Worker节点的总Remote读流量数值，该数值等于只包含GooseFS集群的缓存读流量和从底层存储读流量，不包含短路读流量及DomainSocket读流量。
Cluster.BytesReadRemoteThroughput	GAUGE	GooseFS集群Worker节点的总Remote读带宽数值，该数值等于只包含GooseFS集群的缓存读带宽和从底层存储读带宽，不包含短路读带宽及DomainSocket读带宽。
Cluster.BytesReadUfsAll	COUNTER	GooseFS集群Worker节点从所有底层存储读数据的流量数值。
Cluster.BytesReadUfsThroughput	GAUGE	GooseFS集群Worker节点从所有底层存储读数据的吞吐数值。
Cluster.BytesWrittenDomain	COUNTER	GooseFS集群Worker节点的总DomainSocket写流量数值。
Cluster.BytesWrittenDomainThroughput	GAUGE	GooseFS集群Worker节点的总DomainSocket写带宽数值。
Cluster.BytesWrittenLocal	COUNTER	GooseFS集群Worker节点的总本地短路写流量数值。

Cluster.BytesWrittenLocalThroughput	GAUGE	GooseFS集群Worker节点的总本地短路写吞吐数值。
Cluster.BytesWrittenPerUfs	COUNTER	GooseFS集群Worker节点往某个指定的底层存储写数据的流量数值。
Cluster.BytesWrittenRemote	COUNTER	GooseFS集群Worker节点的总Remote写流量数值，该数值等于只包含GooseFS集群的写缓存流量和往底层存储写数据流量，不包含短路写流量及DomainSocket写流量。
Cluster.BytesWrittenRemoteThroughput	GAUGE	GooseFS集群Worker节点的总Remote写吞吐数值，该数值等于只包含GooseFS集群的写缓存吞吐和往底层存储写数据吞吐，不包含短路写吞吐及DomainSocket写吞吐。
Cluster.BytesWrittenUfsAll	COUNTER	GooseFS集群Worker节点从所有底层存储写数据的流量数值。
Cluster.BytesWrittenUfsThroughput	GAUGE	GooseFS集群Worker节点从所有底层存储写数据的吞吐数值。
Cluster.CapacityFree	GAUGE	GooseFS集群Worker节点的总可用缓存容量数值。
Cluster.CapacityTotal	GAUGE	GooseFS集群Worker节点的总缓存容量数值。
Cluster.CapacityUsed	GAUGE	GooseFS集群Worker节点的总已用缓存容量数值。
Cluster.RootUfsCapacityFree	GAUGE	GooseFS集群远端存储的总可用容量数值。
Cluster.RootUfsCapacityTotal	GAUGE	GooseFS集群远端存储的总容量数值。
Cluster.RootUfsCapacityUsed	GAUGE	GooseFS集群远端存储的总已用容量数值。
Cluster.Workers	GAUGE	GooseFS集群的总Worker节点数量。

Master 监控指标

Master 监控指标有两大类，其一是默认指标，Master 运行过程中会默认记录这些指标；其二是动态监控指标。默认的 Master 监控指标如下：

指标名称	指标类型	指标描述
Master.CompleteFileOps	COUNTER	CompleteFile操作的总请求次数。
Master.CreateDirectoryOps	COUNTER	CreateDirectory操作的总请求次数。
Master.CreateFileOps	COUNTER	CreateFile操作的总请求次数。
Master.DeletePathOps	COUNTER	Delete操作的总请求次数。
Master.DirectoriesCreated	COUNTER	CreateDirectory操作的总成功请求次数。
Master.EdgeCacheEvictions	GAUGE	已清理的元数据EdgeCache。EdgeCache负责管理元数据从(parentId，childName)到childId的映射。
Master.EdgeCacheHits	GAUGE	已命中的元数据EdgeCache。 EdgeCache负责管理元数据从(parentId，childName)到childId的映射。
Master.EdgeCacheLoadTimes	GAUGE	元数据EdgeCache的加载次数。 EdgeCache负责管理元数据从(parentId，childName)到childId的映射。
Master.EdgeCacheMisses	GAUGE	未命中的元数据EdgeCache。EdgeCache负责管理元数据从(parentId，childName)到childId的映射。
Master.EdgeCacheSize	GAUGE	元数据EdgeCache的总大小。 EdgeCache负责管理元数据从(parentId，childName)到childId的映射。
Master.EdgeLockPoolSize	GAUGE	Master节点EdgeLockPool的大小。
Master.FileBlockInfosGot	COUNTER	GetFileBlockInfo操作的总请求次数
Master.FileInfosGot	COUNTER	GetFileInfo操作的总成功请求次数。
Master.FilesCompleted	COUNTER	CompleteFile的总成功请求次数。

Master.FilesCreated	COUNTER	CreateFile的总成功请求次数。
Master.FilesFreed	COUNTER	FreeFile的总成功请求次数。
Master.FilesPersisted	COUNTER	已持久化到底层存储的总文件数。
Master.FilesPinned	GAUGE	当前已锁定的缓存文件数。
Master.FreeFileOps	COUNTER	FreeFile操作的总请求次数。
Master.GetFileBlockInfoOps	COUNTER	GetFileBlockInfo的总请求次数。
Master.GetFileInfoOps	COUNTER	GetFileInfo的总请求次数。
Master.GetNewBlockOps	COUNTER	GetNewBlock的总请求次数。
Master.InodeCacheEvictions	GAUGE	元数据缓存中inodes的总淘汰次数。
Master.InodeCacheHits	GAUGE	元数据缓存中inodes的总命中次数。
Master.InodeCacheLoadTimes	GAUGE	元数据缓存中inodes的总加载次数。
Master.InodeCacheMisses	GAUGE	元数据缓存中inodes的总Miss次数。
Master.InodeCacheSize	GAUGE	已缓存的Inodes总数。
Master.InodeLockPoolSize	GAUGE	Master inode 的lock pool大小。
Master.JournalFlushFailure	COUNTER	journal flush 的总失败次数。
Master.JournalFlushTimer	TIMER	journal flush 的计时器统计次数。

Master.JournalGainPrimacyTimer	TIMER	journal gain primacy 的计时器统计次数。
Master.LastBackupEntriesCount	GAUGE	元数据备份的元数据日志中所记录的元数据条目总数。该参数用于记录元数据的备份情况，并在需要时作为恢复节点的参考。
Master.LastBackupRestoreCount	GAUGE	元数据恢复时记录的元数据条目总数。该参数用于记录元数据的恢复情况。
Master.LastBackupRestoreTimeMs	GAUGE	元数据恢复的处理时长。
Master.LastBackupTimeMs	GAUGE	元数据备份的处理时长。
Master.ListingCacheSize	GAUGE	Master节点的listing cache大小。
Master.MountOps	COUNTER	Mount操作的总请求次数。
Master.NewBlocksGot	COUNTER	GetNewBlock操作的总成功请求次数。
Master.PathsDeleted	COUNTER	Delete操作的总成功请求次数。
Master.PathsMounted	COUNTER	Mount操作的总成功请求次数。
Master.PathsRenamed	COUNTER	Rename操作的总成功请求次数。
Master.PathsUnmounted	COUNTER	Unmount操作的总成功请求次数。
Master.RenamePathOps	COUNTER	Rename操作的总请求次数。
Master.SetAclOps	COUNTER	SetAcl操作的总请求次数。
Master.SetAttributeOps	COUNTER	SetAttribute的总请求次数。
Master.TotalPaths	GAUGE	GooseFS命名空间中的总文件和目录数量。

Master.UfsJournalCatchupTimer	TIMER	GooseFS底层存储的journal catchup计时统计次数。
Master.UfsJournalFailureRecoverTimer	TIMER	GooseFS底层存储的journal failure recover计时统计次数。
Master.UfsJournalInitialReplayTimeMs	GAUGE	GooseFS底层存储的journal initial replay处理时长。
Master.UnmountOps	COUNTER	Unmount操作的总请求次数。

动态指标如下：

指标名称	指标类型	指标描述
Master.CapacityTotalTier	GAUGE	GooseFS各缓存层级的容量大小。
Master.CapacityUsedTier	GAUGE	GooseFS各缓存层级的已用容量大小。
Master.CapacityFreeTier	GAUGE	GooseFS各缓存层级的可用容量大小。
Master.UfsSessionCount-Ufs:	COUNTER	GooseFS 当前已打开的 UFS 会话数量。
Master.PerUfsOp.UFS:	TIMER	GooseFS 主 Master 节点对 UFS 的所有操作数量。

Worker 监控指标

Worker 监控指标有两大类，其一是默认指标，Worker 运行过程中会默认记录这些指标；其二是动态监控指标。默认的 Worker 监控指标如下：

指标名称	指标类型	指标描述
Worker.AsyncCacheDuplicateRequests	COUNTER	指定Worker节点收到的重复异步缓存请求的总数。
Worker.AsyncCacheFailedBlocks	COUNTER	指定Worker节点异步缓存失败的block数量。

Worker.AsyncCacheRemoteBlocks	COUNTER	需要从远端资源中异步缓存的block数量。
Worker.AsyncCacheRequests	COUNTER	指定Worker节点收到的异步缓存请求总数。
Worker.AsyncCacheSucceededBlocks	COUNTER	指定Worker节点收到的成功的异步缓存请求总数。
Worker.AsyncCacheUfsBlocks	COUNTER	需要从本地资源中异步缓存的block数量。
Worker.BlockRemoverBlocksToRemovedCount	COUNTER	指定Worker节点被asynchronous block remover清理的block数量。
Worker.BlockRemoverRemovingBlocksSize	GAUGE	指定Worker节点正在被asynchronous block remover清理的block数量。
Worker.BlockRemoverTryRemoveBlocksSize	GAUGE	指定Worker节点已经被asynchronous block remover清理的block数量。
Worker.BlockRemoverTryRemoveCount	COUNTER	指定Worker节点即将被asynchronous block remover清理的block数量。
Worker.BlocksAccessed	COUNTER	指定Worker节点所有数据块被访问的总次数。
Worker.BlocksCached	GAUGE	指定Worker节点中缓存的数据块总数。
Worker.BlocksCancelled	COUNTER	指定Worker节点中释放的临时数据块总数。
Worker.BlocksDeleted	COUNTER	指定Worker节点中被外部请求删除的数据块总数。
Worker.BlocksEvicted	COUNTER	指定Worker节点中淘汰的数据块总数。

版权所有：腾讯云计算（北京）有限责任公司

Worker.BlocksLost	COUNTER	指定Worker节点中丢失的数据块总数，包含。
Worker.BlocksPromoted	COUNTER	指定Worker节点中数据块转移到不同缓存层的次数。
Worker.BytesReadDomain	COUNTER	指定Worker节点中被domain read的总读流量。
Worker.BytesReadDomainThroughput	METER	指定Worker节点中被domain read的总读带宽。
Worker.BytesReadPerUfs	COUNTER	指定Worker节点从底层存储中读取文件的流量数值。
Worker.BytesReadRemote	COUNTER	指定Worker节点的Remote读流量数值，该数值等于只包含该节点的读缓存流量和从底层存储读数据流量，不包含短路读流量及DomainSocket的读流量。
Worker.BytesReadRemoteThroughput	METER	指定Worker节点的Remote读吞吐数值，该数值等于只包含该节点的读缓存吞吐和从底层存储读数据吞吐，不包含短路读吞吐及DomainSocket的读吞吐。
Worker.BytesReadUfsThroughput	METER	指定Worker节点从所有底层存储中读数据吞吐数值。
Worker.BytesWrittenDomain	COUNTER	指定Worker节点中，通过Domain socket方式写入GooseFS的写流量数值。
Worker.BytesWrittenDomainThroughput	METER	指定Worker节点中，通过Domain socket方式写入GooseFS的写吞吐数值。
Worker.BytesWrittenPerUfs	COUNTER	指定Worker节点往指定底层存储中写数据吞吐数值。
Worker.BytesWrittenRemote	COUNTER	指定Worker节点的Remote写流量数值，该数值等于只包含该节点的写缓存流量和往底层存储写数据流量，不包含短路写流量及DomainSocket的写流量。
Worker.BytesWrittenRe	METER	指定Worker节点的Remote写吞吐数值，该数值等于只包含该节点的写缓存吞吐和往底层存储写数据吞吐，不包含短路写吞吐及DomainSocket的写吞吐。

moteThroughput		
Worker.BytesWrittenUfsThroughput	METER	指定Worker节点往所有底层存储中写数据吞吐数值。
Worker.CapacityFree	GAUGE	指定Worker节点上的可用缓存容量数值。
Worker.CapacityTotal	GAUGE	指定Worker节点上的总缓存容量数值。
Worker.CapacityUsed	GAUGE	指定Worker节点上的已用缓存容量数值。

动态监控指标如下：

指标名称	指标类型	指标描述
Worker.UfsSessionCount-Ufs	GAUGE	指定Worker节点上已打开的UFS会话数量。

客户端监控指标

客户端监控指标会通过指定的用户 ID 或者客户端的 IP 维度汇聚，如果已经在 GooseFS 中注册好用户 ID，则会优先按照用户 ID 维度汇聚。用户 ID 可以通过 goosefs.user.app.id 指定。客户端监控指标如下：

指标名称	指标类型	指标描述
Client.BytesReadLocal	COUNTER	指定Client节点从本地存储短路读流量数值。
Client.BytesReadLocalThroughput	METER	指定Client节点从本地存储短路读吞吐数值。
Client.BytesWrittenLocal	COUNTER	指定Client节点往本地存储短路写流量数值。
Client.BytesWrittenLocalThroughput	METER	指定Client节点往本地存储短路写吞吐数值。
Client.BytesWrittenUfs	COUNTER	指定Client节点往GooseFS底层存储的写流量数值。
Client.CacheBytesEvicted	METER	指定Client节点淘汰的数据量。

Client.CacheBytesReadCache	METER	指定Client节点缓存命中，从Client节点中读取的缓存数据量。
Client.CacheBytesReadExternal	METER	指定Client节点缓存未命中，从其他存储中读取的数据量。
Client.CacheBytesRequestedExternal	METER	指定Client节点缓存未命中，从其他存储中请求的数据量。由于chunk读情况的存在，该数值小于等于Client.CacheBytesReadExternal。
Client.CacheBytesWrittenCache	METER	往指定Client节点缓存中写入数据的流量。
Client.CacheCleanupGetErrors	COUNTER	清理失败的读缓存请求时的失败次数。
Client.CacheCleanupPutErrors	COUNTER	清理失败的写缓存请求时的失败次数。
Client.CacheCreateErrors	COUNTER	指定Client节点创建缓存的失败次数。
Client.CacheDeleteErrors	COUNTER	指定Client节点删除缓存的失败次数。
Client.CacheGetErrors	COUNTER	指定Client节点读取缓存的失败次数。
Client.CacheGetNotReadyErrors	COUNTER	指定Client节点读取页时，由于缓存未就绪导致的失败次数。
Client.CacheHitRate	GAUGE	指定Client节点的缓存命中率，该参数等于从缓存中读请求次数/总请求次数。
Client.CachePages	COUNTER	指定Client节点的页缓存数量。
Client.CachePagesEvicted	METER	指定Client节点中已淘汰的页缓存数量。
Client.CachePutErrors	COUNTER	指定Client节点写入缓存数据的失败次数。
Client.CacheSpaceAvailable	GAUGE	指定Client节点中可用的字节数。

Client.CacheSpaceUsed	GAUGE	指定Client节点中已用的字节数。
Client.CacheSpaceUsedCount	COUNTER	指定Client节点中可用的字节计数值。
Client.CacheState	COUNTER	指定Client节点中的缓存状态，0（不可用），1（只读），2（只写）
Client.CacheUnremovableFiles	COUNTER	指定Client节点中不可用的缓存数量。

进程监控指标

进程监控指标可以在 Cluster、Master 和 Worker 中收集并汇聚。主要包括 JVM 信息、垃圾回收信息、内存占用信息三大类。
JVM 信息包括以下内容：

指标名称	指标描述
name	JVM 名称
uptime	JVM 更新时间
vendor	当前的 JVM vendor

垃圾回收监控指标包括以下内容：

指标名称	指标描述
PS-MarkSweep.count	Mark 和 Sweep 的总数。
PS-MarkSweep.time	Mark 和 Sweep 的耗时。
PS-Scavenge.count	Scavenge 次数。
PS-Scavenge.time	Scavenge 时间。

内存占用监控指标记录了内存使用情况的概述和详情信息，部分监控指标如下：

指标名称	指标描述
total.committed	JVM 可保证使用的内存量。
total.init	JVM 可用的内存量。
total.max	JVM 的最大可用内存量。
total.used	当前已用的 JVM 内存量。

版权所有：腾讯云计算（北京）有限责任公司

heap.committed	JVM 可保证使用的堆内存大小。
heap.init	JVM 可用的堆内存大小。
heap.max	JVM 的最大可用堆内存大小。
heap.usage	当前正在使用的 JVM 堆内存量。
heap.used	当前已用的 JVM 堆内存量。
pools.Code-Cache.used	内存池中用于编译和存储本地代码的内存大小。
pools.Compressed-Class-Space.used	Used memory of collection usage from the pool from which memory is use for class metadata 内存池中用于收集用途的已用内存大小，该内存用于类元数据。
pools.PS-Eden-Space.used	内存池中用于收集用途的已用内存大小，该内存用于大部分对象的初始化。
pools.PS-Survivor-Space.used	内存池中用于收集用途的已用内存大小，该内存池包含在Eden空间垃圾回收后幸存的对象。

基于 Prometheus 搭建 GooseFS 监控体系

最近更新时间：2024-10-12 10:20:41

Goosefs 可以通过配置将指标数据输出到不同的监控系统中，Prometheus 是其中之一。Prometheus 是一个开源的监控框架，目前腾讯云监控已集成了 Prometheus，下文将重点介绍 Goosefs 监控指标，以及将监控指标上报到自建的 Prometheus 和云上 Prometheus 的流程。

准备工作

通过 Prometheus 构建监控体系需要先做如下准备工作：

- 配置 GooseFS 集群
- 下载 Prometheus 官方安装包或腾讯云 Prometheus 安装包
- 下载和配置 Grafana

启用 GooseFS 监控指标上报配置

1. 编辑 GooseFs 配置 conf/goosefs-site.properties， 添加如下配置项，并使用 goosefs copyDir conf/ 拷贝到所有 worker节点，并重启集群 `./bin/goosefs-start.sh all` 。

```
goosefs.user.metrics.collection.enabled=true
goosefs.user.metrics.heartbeat.interval=10s
```

2. master 和 worker 的 Prometheus 的监控指标可用如下的命令查看，其中 master 的 metrics 端口为9201，worker 的 metrics 端口为 9204：

```
curl <LEADING_MASTER_HOSTNAME>:<MASTER_WEB_PORT>/metrics/prometheus/
# HELP Master_CreateFileOps Generated from Dropwizard metric import
(metric=Master.CreateFileOps, type=com.codahale.metrics.Counter)
...

curl <WORKER_IP>:<WOKER_PORT>/metrics/prometheus/
# HELP pools_Code_Cache_max Generated from Dropwizard metric import
(metric=pools.Code-Cache.max,
type=com.codahale.metrics.jvm.MemoryUsageGaugeSet$$Lambda$51/137460818)
...
```

上报监控指标到自建 Prometheus

1. 下载 Prometheus 安装包并解压，修改 prometheus.yml：

```
# prometheus.yml
global:
    scrape_interval:     10s
    evaluation_interval: 10s
scrape_configs:
```

```
    - job_name: 'goosefs masters'
      metrics_path: /metrics/prometheus
      file_sd_configs:
      - refresh_interval: 1m
      files:
      - "targets/cluster/masters/*.yml"
    - job_name: 'goosefs workers'
      metrics_path: /metrics/prometheus
      file_sd_configs:
      - refresh_interval: 1m
      files:
      - "targets/cluster/workers/*.yml"
```

2. 创建 targets/cluster/masters/masters.yml，添加 master 的 IP 和 port：

```
- targets:
  - "&lt;TARGERTS_MASTER_IP>:&lt;TARGERTS_MASTER_PORT>"
```

3. 创建 targets/cluster/workers/workers.yml，添加 worker 的 IP 和 port：

```
- targets:
  - "&lt;TARGERTS_WORKER_IP>:&lt;TARGERTS_WORKER_PORT>"
```

4. 启动 Prometheus，其中 --web.listen-address 指定 Prometheus 监听地址，默认端口号 9090：

```
nohup ./prometheus --config.file=prometheus.yml --web.listen-address="
<LISTEN_IP>:<LISTEN_PORT>" > prometheus.log 2>&1 &
```

5. 查看可视化界面：

```
http://<PROMETHEUS_BI_IP>:<PROMETHEUS_BI_PORT>
```

6. 查看机器实例：

```
http://<PROMETHEUS_BI_IP>:<PROMETHEUS_BI_PORT>/targets
```

上报监控指标到腾讯云 Prometheus

1. 按照安装指南中的指引，在 master 机器上安装 Promethus agent：

```
wget https://rig-1258344699.cos.ap-guangzhou.myqcloud.com/prometheus-
agent/agent_install && chmod +x agent_install && ./agent_install prom-12kqy0mw
agent-grt164ii ap-guangzhou <secret_id> <secret_key>
```

2. 配置 master 和 worker 的抓取任务：

方式一：

```
job_name: goosefs-masters
honor_timestamps: true
metrics_path: /metrics/prometheus
scheme: http
file_sd_configs:
- files:
  - /usr/local/services/prometheus/targets/cluster/masters/*.yml
  refresh_interval: 1m
job_name: goosefs-workers
honor_timestamps: true
metrics_path: /metrics/prometheus
scheme: http
file_sd_configs:
- files:
  - /usr/local/services/prometheus/targets/cluster/workers/*.yml
  refresh_interval: 1m
```

> ⚠ **注意：**
> job_name 中没有空格，而单机的 Prometheus 的 job_name 中可以包含空格。

方式二：

```
job_name: goosefs masters
honor_timestamps: true
metrics_path: /metrics/prometheus
scheme: http
static_configs:
- targets:
 - "<TARGERTS_MASTER_IP>:<TARGERTS_MASTER_PORT>"
 refresh_interval: 1m

job_name: goosefs workers
honor_timestamps: true
metrics_path: /metrics/prometheus
scheme: http
static_configs:
- targets:
 - "<TARGERTS_WORKER_IP>:<TARGERTS_WORKER_PORT>"
 refresh_interval: 1m
```

> ⚠ **注意：**

抓取任务按方式二配置，则无需在 targets/cluster/masters/ 路径下创建 masters.yml 和 workers.yml 文件。

使用 Grafana 查看监控指标

1. 启动 Grafana：

```
nohup ./bin/grafana-server web > grafana.log 2>&1 &
```

2. 打开登录页面 http://<GRAFANA_IP>:<GRAFANA_PORT>，Grafana 的默认端口为 3000，username 和 password 都是 admin，首次登录后可修改密码。
3. 进入页面后，添加 Prometheus 的 Datasource：

```
<PROMETHEUS_IP>:<PROMETHEUS_PORT>
```

4. 导入 Goosefs 的 Grafana 模板，选择 json 导入（点此下载 json），并选择上面创建的 Datasource。

注意：
云上 Prometheus 购买时需设置密码，云上 Grafana 的可视化监控界面配置和上面类似，注意 job_name 需要配置成一致。

5. 修改 DashBoard 以后，可以将 DashBoard 导出来。

版权所有：腾讯云计算（北京）有限责任公司

日志指引
GooseFS 日志介绍

最近更新时间：2024-10-12 10:20:41

GooseFS 的 Master 和 Worker 节点，以及 Spark 等计算框架通过 GooseFS Client 请求 GooseFS 时，都会记录请求日志，用户可对输出日志进行分析，进行问题排查。GooseFS 日志输出基于 log4j 实现，可以通过修改 log4j.properties 配置文件来调整 GooseFS 的日志输出，例如日志存储路径，日志级别，是否记录 RPC 调用情况等。用户可以到 GooseFS 的配置文件目录下，打开并修改 log4j.properties 文件：

```
$ cd /usr/local/service/goosefs/conf
$ cat log4j.properties

# May get overridden by System Property

log4j.rootLogger=INFO, ${goosefs.logger.type}, ${goosefs.remote.logger.type}

log4j.category.goosefs.logserver=INFO, ${goosefs.logserver.logger.type}
log4j.additivity.goosefs.logserver=false

log4j.logger.AUDIT_LOG=INFO, ${goosefs.master.audit.logger.type}
log4j.additivity.AUDIT_LOG=false

...
```

下文将详细介绍 GooseFS 的日志配置：

日志存储位置

GooseFS 采集的日志默认存储在 ${GOOSEFS_HOME}/logs 目录下。其中，Master 采集的日志存储在 logs/master.log 中，Worker 采集的日志存储在 logs/worker.log 中。需要注意的是， 节点进程异常抛出的日志会记录在 master.out 或者 worker.out 中，正常情况下这两类文件均为空文件，系统有异常时会记录异常信息以便追溯。

以 Master 节点的日志存储配置为例，以下为常用的几项配置：

```
# Appender for Master
log4j.appender.MASTER_LOGGER=org.apache.log4j.RollingFileAppender
log4j.appender.MASTER_LOGGER.File=${goosefs.logs.dir}/master.log
log4j.appender.MASTER_LOGGER.MaxFileSize=10MB
log4j.appender.MASTER_LOGGER.MaxBackupIndex=100
log4j.appender.MASTER_LOGGER.layout=org.apache.log4j.PatternLayout
log4j.appender.MASTER_LOGGER.layout.ConversionPattern=%d{ISO8601} %-5p %c{1} -
%m%n
```

配置参数介绍如下：

- MASTER_LOGGER：指定配置 MASTER 的日志输出。
- MASTER_LOGGER.File：指定日志存储路径，可以通过修改路径来自定义日志存储位置。
- MASTER_LOGGER.MaxFileSize：指定单个日志文件大小的上限。
- MASTER_LOGGER.MaxBackupIndex：指定日志文件数上限。
- MASTER_LOGGER.layout：指定日志输出格式模板。
- MASTER_LOGGER.layout.ConversionPattern：指定日志输出的具体格式。

> ⚠ 注意
> - .log 文件是滚动存储的，您可以将其备份到 UFS，例如对象存储中；.out 文件不滚动存储，如果需要清理 .out 文件需要手动发起删除操作。
> - 更多 log4j 的参数配置可以参考 log4j configuration 文档 。
> - GooseFS 仅存储本身产生的日志，上层计算应用产生的日志可以根据计算应用的日志配置，查看日志存储位置。常见计算应用的日志配置信息可见：Apache Hadoop，Apache HBase，Apache Hive，Apache Spark 。

日志级别

GooseFS 提供了以下5种级别的日志：

- TRACE: 详细的调用日志，适用于调试方法和类的调用。
- DEBUG: 较详细的调用日志，适用于 DEBUG 过程中排查问题。
- INFO: 请求处理过程中的关键信息。
- WARN: 警告类信息，任务可以继续执行，但需要注意可能存在问题。
- ERROR: 系统报错信息，影响任务进行。

上述5种级别日志的详细程度从高到低，配置高等级的日志级别会一并记录低等级的日志信息。默认情况下 GooseFS 配置 INFO 级别的日志，记录 INFO、WARN、ERROR三个级别的日志。
可以到 GooseFS 的配置文件目录下打开并修改 log4j.properties 文件，如下示例展示了将所有 GooseFS 的日志级别修改为 DEBUG 级别：

```
log4j.rootLogger=DEBUG, ${goosefs.logger.type}, ${goosefs.remote.logger.type}
```

如果需要修改指定类的日志级别，可以在配置文件中添加声明，如下示例展示指定 GooseFSFileInStream 这个类的日志级别为 DEBUG：

```
log4j.logger.com.qcloud.cos.goosefs.client.file.GooseFSFileInStream=DEBUG
```

一般而言，推荐在日志配置文件中修改日志级别。但在一些特定的场景下，用户可能需要在集群运行过程中修改日志参数，此时可以通过在命令行中输入 goosefs logLevel 指令进行调整。如下为 logLevel 支持的配置选项：

```
usage: logLevel [--level <arg>] --logName <arg> [--target <arg>]
    --level <arg>     The log level to be set.
```

```
    --logName <arg>   The logger's name(e.g.
com.qcloud.cos.goosefs.master.file.DefaultFileSystemMaster) you want to get or
set level.
    --target <arg>    <master|workers|host:webPort>. A list of targets separated
by, can be specified. host:webPort pair must be one of workers. Default target is
master and all workers
```

各项配置的详细说明如下：

- level：日志级别，支持 TRACE、DEBUG、INFO、WARN、ERROR 五种级别。
- logName：日志输出 logger，如 com.qcloud.cos.goosefs.underfs.hdfs.HdfsUnderFileSystem 等。
- target：修改范围，可以设置为指定的 Master 或者 Worker 节点（通过 IP：PORT 方式指定），默认为 Master 和所有 Worker 节点。

用户可以按需在系统运行过程中调整日志级别，以便排查系统运行过程中的问题。如以下示例展示了在运行过程中将所有 Worker 节点上 com.qcloud.cos.goosefs.underfs.hdfs.HdfsUnderFileSystem 这个类的日志级别调整为 DEBUG 级别，并在调试完成后调整回 INFO 级别：

```
$  goosefs logLevel --
logName=com.qcloud.cos.goosefs.underfs.hdfs.HdfsUnderFileSystem --target=workers
--level=DEBUG # 调整为 DEBUG 模式

$  goosefs logLevel --
logName=com.qcloud.cos.goosefs.underfs.hdfs.HdfsUnderFileSystem --target=workers
--level=INFO # 调整为 INFO 模式
```

高级配置

GooseFS 支持配置 GC 事件日志，FUSE 接口日志，RPC 调用日志，UFS 操作日志，以及进行日志分割和日志筛选等操作。以下介绍部分常用高级配置的使用方式。

- **GC 事件日志**
  GooseFS 将 GC 事件日志记录在 .out 文件中， 可以通过在 conf/goosefs-env.sh 中添加如下配置：

  ```
  GOOSEFS_JAVA_OPTS+=" -XX:+PrintGCDetails -XX:+PrintTenuringDistribution -
  XX:+PrintGCTimeStamps"
  ```

  GOOSEFS_JAVA_OPTS 为所有类型 GooseFS 节点的 Java 虚拟机参数，也可以通过指定 GOOSEFS_MASTER_JAVA_OPTS 和 GOOSEFS_WORKER_JAVA_OPTS 分别指定 Master 和 Worker上的虚拟机参数。
- **FUSE 接口日志**
  可以在 conf/log4j.properties 文件中配置记录 FUSE 日志级别：

  ```
  goosefs.logger.com.qcloud.cos.goosefs.fuse.GoosefsFuseFileSystem=DEBUG
  ```

  启用后 FUSE 接口日志可以在 logs/fuse.log 查看。

- **启用 RPC 调用日志**
  GooseFS 支持在 conf/log4j.properties 配置文件中，配置 Client 端或 Master 端的 RPC 调用日志。
  可以通过在 log4j.properties 文件中，配置客户端输出 RPC 请求日志：

```
log4j.logger.com.qcloud.cos.goosefs.client.file.FileSystemMasterClient=DEBUG # Client 与 FileSystemMaster 之间的 RPC 请求日志
log34j.logger.com.qcloud.cos.goosefs.client.block.BlockSystemMasterClient=DEBUG # Client 与 BlockMaster 之间的 RPC 请求日志
```

  可以通过 logLevel 指令来，配置 Master 输出 RPC 请求日志：

```
$ goosefs logLevel \--logName=com.qcloud.cos.goosefs.master.file.FileSystemMasterClientServiceHandler \--target master --level=DEBUG # 文件相关的 RPC 请求日志
$ goosefs logLevel \--logName=com.qcloud.cos.goosefs.master.block.BlockSystemMasterClientServiceHandler \--target master --level=DEBUG # 块相关别的 RPC 请求日志
```

- **UFS 操作日志**
  UFS 操作日志输出配置，可以通过在 log4j.properties 文件中添加配置项，或者通过 logLevel 指令进行操作。下面以 logLevel 指令为例：

```
$ goosefs logLevel \--logName=com.qcloud.cos.goosefs.underfs.UnderFileSystemWithLogging \--target master --level=DEBUG # 记录 master 节点对 UFS的操作日志
$ goosefs logLevel \--logName=com.qcloud.cos.goosefs.underfs.UnderFileSystemWithLogging \--target workers --level=DEBUG # 记录 worker 节点对 UFS的操作日志
```

- **分割日志**
  GooseFS 支持将指定类型日志存储在指定存储路径，如果将所有日志都记录在 .log 文件，可能产生以下问题：
- 当集群规模大，吞吐量较多时，master.log 或者 worker.log 文件可能变得异常庞大，或者滚动产生大量日志文件。
- 日志信息较多，不利于针对性的进行日志分析。
- 大量日志存储在本地节点，占用空间。

因此业务可以根据需要在 log4j.properties 文件中添加配置项，将指定类产生的日志存储至指定文件路径，如下示例展示了将 StateLockManager 类的日志存储在 statelock.log 中：

```
log4j.category.com.qcloud.cos.goosefs.master.StateLockManager=DEBUG,State_LOCK_LOGGER
log4j.additivity.com.qcloud.cos.goosefs.master.StateLockManager=false
log4j.appender.State_LOCK_LOGGER=org.apache.log4j.RollingFileAppender
log4j.appender.State_LOCK_LOGGER.File=<GOOSEFS_HOME>/logs/statelock.log
log4j.appender.State_LOCK_LOGGER.MaxFileSize=10MB
log4j.appender.State_LOCK_LOGGER.MaxBackupIndex=100
```

版权所有：腾讯云计算（北京）有限责任公司

```
log4j.appender.State_LOCK_LOGGER.layout=org.apache.log4j.PatternLayout
log4j.appender.State_LOCK_LOGGER.layout.ConversionPattern=%d{ISO8601} %-5p %c{1}
- %m%
```

- **筛选日志**
  GooseFS 支持按照一定的条件筛选并记录日志，而不是全量记录所有日志。例如在进行性能测试时需要记录 RPC 调用日志，但业务侧并不需要记录所有 RPC 调用日志，只需要记录某些延迟较高的日志即可，此时可以通过在 log4j.properties 文件中添加配置项添加日志筛选条件即可，如下示例分别展示了筛选 RPC 调用延迟超过200ms的请求和 FUSE 调用超过1s的请求：

  ```
  goosefs.user.logging.threshold=200ms
  goosefs.fuse.logging.threshold=1s
  ```

GooseFS 日志上报

最近更新时间：2024-08-16 16:37:31

简介

GooseFS 日志上报功能支持将 GooseFS 运行日志上报到腾讯云的日志服务（Cloud Log Service，CLS）或 Elasticsearch Service（ES）中。本文将介绍如何将 GooseFS 的运行日志上报到这两种日志系统中。

准备工作

部署 GooseFS 集群，GooseFS 的部署请参考 部署环境要求 。

上报日志

GooseFS 上报日志到 CLS

创建 CLS 主题

GooseFS 日志上报依赖于第三方采集系统平台，此处以 CLS 为例进行说明。
创建 CLS 主题，详情请参见 CLS 一分钟入门指南 。

配置 filebeat

1. 配置日志采集目录，编辑 GooseFS 部署根目录下的 $GOOSEFS_HOME/conf/filebeat.yml 文件。

```
- type: log
  enabled: true
  paths:
    - ${path.home}/../logs/job_master.log*

  fields:
    type: "master"
  exclude_files: ['.gz$']

  multiline.pattern: '^[[:space:]]+(at|\.{3})[[:space:]]+\b|^Caused by:'
  multiline.negate: false
  multiline.match: after
```

> **说明：**
> - paths：配置要采集的日志路径，多个日志文件使用通配符*。
> - fields.type：自定义类型名。
> - multiline.pattern：多行合并规则。

2. 配置 CLS 日志采集平台账号。

```
output.kafka:
  hosts: ["sh-producer.cls.tencentcs.com:9096"]
  topic: "a99cf1de-81d4-47a-97xxxxx-xxxx"
  version: "0.11.0.0"
  compression: "none"
  username: "cc098474-b387-381xxxx-xxxxx"
  password: "secretId#secretKey"
```

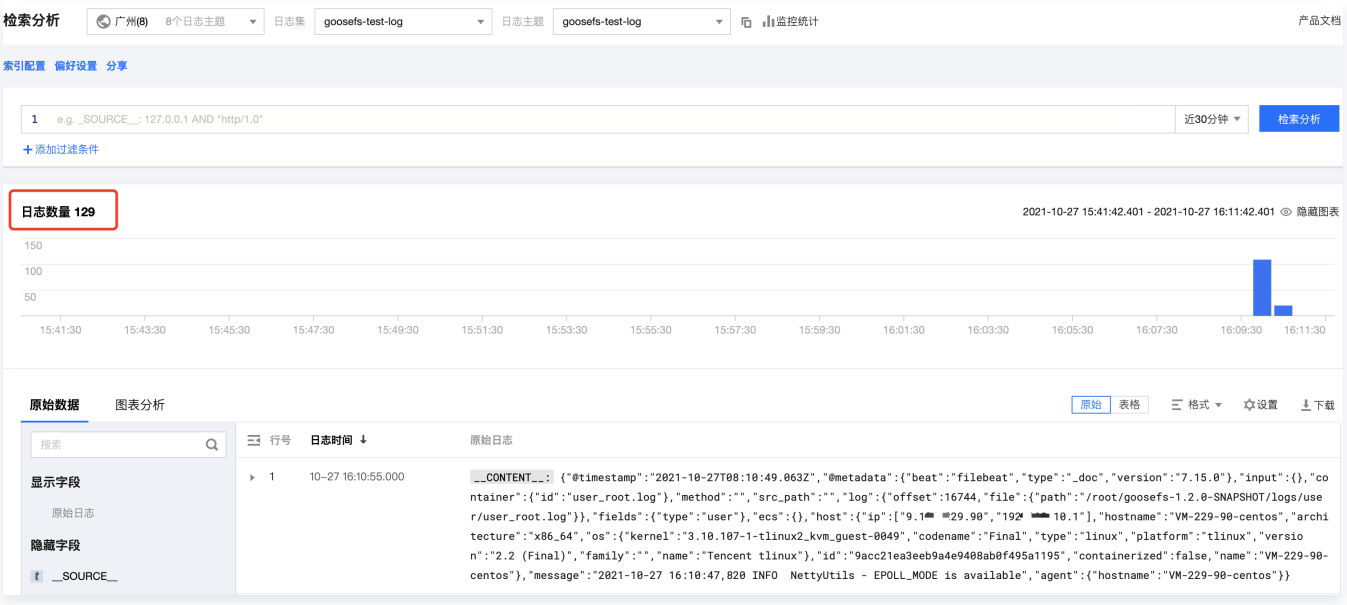
说明

- hosts：CLS 平台的可用地域。具体地址可参考CLS平台配置：CLS日志服务-使用 Kafka 协议上传日志 。
- topic：CLS 平台创建日志主题时分配的日志主题 ID。
- version：CLS 平台服务支持 kafka 的版本，默认值：0.11.0.0。
- username：CLS 平台创建日志主题时分配的日志集 ID。
- password：SecretId#SecretKey，SecretId 和 SecretKey 是云管理账号下 App 密钥管理中分配的密钥。

3. 进入 $GOOSEFS_HOME/filebeat 目录，执行如下命令，启动 filebeat。

```
./goosefs-filebeat -c filebeat.yml
```

4. filebeat 启动完成之后，会实时采集 GooseFS 服务产生的日志上报到 CLS 平台，通过 CLS 平台可以查看具体上报的日志信息。如下图所示：



GooseFS 上报日志到 ES

创建 ES

GooseFS 日志上报依赖于第三方采集系统平台，下面以 ES 为例进行说明：

- 创建 ES 集群，具体创建方式请参见：创建 ES 集群 。

- 访问 ES 集群，具体访问方式请参见：访问 ES 集群 。

配置 filebeat

1. 配置日志采集目录，编辑 GooseFS 部署根目录下的 $GOOSEFS_HOME/conf/filebeat.yml 文件。

```
- type: log
  enabled: true
  paths:
    - ${path.home}/../logs/job_master.log*

  fields:
    type: "master"
  exclude_files: ['.gz$']

  multiline.pattern: '^[[:space:]]+(at|\.{3})[[:space:]]+\b|^Caused by:'
  multiline.negate: false
  multiline.match: after
```

> **说明：**
> - paths：配置要采集的日志路径，多个日志文件使用通配符*。
> - fields.type：自定义类型名。
> - multiline.pattern：多行合并规则。

2. 配置 ES 日志采集平台账号。

```
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["es-nli7xxxxx.public.tencentelasticsearch.com"]

  protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  username: "elasticxxx"
  password: "costestxxx"

  index: "GooseFS-%{[fields.type]}-%{+yyyy.MM.dd}"
```

> **说明**
> - hosts：ES 对外提供服务地址。
> - username：ES 登录用户名。
> - password：ES 登录密码。

- index：ES 日志输出索引。例如，master 的 index 是 GooseFS-master-2022.01.18，worker 的 index 是 GooseFS-worker-2022.01.18等。

3. 进入 $GOOSEFS_HOME/filebeat 目录，执行如下命令，启动 filebeat。

```
./goosefs-filebeat -c filebeat.yml
```

4. filebeat 启动完成后，会实时采集 GooseFS 服务产生的日志上报到 ES 平台，通过 ES 平台可以查看具体上报的日志信息，如下图所示：

开启 GooseFS 审计日志上报

如果需要上报 GooseFS 服务的审计日志，则需要开启审计日志配置，操作如下：

1. 编辑 GooseFS 部署目录下的 $GOOSEFS_HOME/conf/goosefs-site.properties 文件，添加如下配置项（可选）。

```
goosefs.master.audit.logging.enabled=true
```

2. 执行如下命令，将文件 $GOOSEFS_HOME/conf/goosefs-site.properties 拷贝到所有 worker 节点。

```
goosefs copyDir conf/
```

3. 启动 GooseFS 集群。

```
./bin/goosefs-start.sh all
```

集群配置实践
AI 场景生产环境配置实践

最近更新时间：2023-09-26 18:46:03

简介

数据加速器 GooseFS 提供了多种部署方式，支持 自建部署，在 TKE 上部署 以及在 EMR 上部署 等。在大数据场景中，通常使用**集群模式**部署，并且采用**高可用架构**以满足业务连续性的要求。

高可用架构指多个 Master 节点的主备多活架构。多个 Master 节点中只有一个会作为主（Leader）节点对外提供服务，其他的备（Standby）节点通过同步共享日志（Journal）来保持与主节点相同的文件系统状态。当主节点故障宕机后，会从当前的备节点中自动选择一个接替主节点继续对外提供服务，这样就消除了系统的单点故障，实现了整体高可用架构。目前 GooseFS 支持基于 Raft 日志和 Zookeeper 两种方式来实现主备节点状态的强一致性，在容器场景下，我们推荐基于 Raft 日志模式部署您的高可用架构。本文重点介绍基于 Raft 的高可用部署配置，并区分了顺序读和随机读两种不同场景。

基于 Raft 的高可用架构部署配置（顺序读场景）

在顺序读的场景下，您可以参考如下推荐配置，并将该配置项复制粘贴到 goosefs-site.properties 文件中，完成您的高可用架构配置：

```
goosefs.master.embedded.journal.addresses=<master1>:9202,<master2>:9202,
<master3>:9202

goosefs.master.metastore=ROCKS
# 主要是在不确定 rocksdb 是否修复的情况下使用
goosefs.master.metastore.block=HEAP

# 根据内存大小而定
goosefs.master.metastore.inode.cache.max.size=10000000

# rocksdb 数据存放的地方
goosefs.master.metastore.dir=/meta-1/metastore

# 根目录的挂载路径，需要放在安全目录中，防止被误删除
goosefs.master.mount.table.root.ufs=/meta-1/underFSStorage

# raft 日志存放的地方
goosefs.master.journal.folder=/meta-1/journal

# 触发切主的超时时间，不易过小（jvm gc 会导致切主震荡），也不易过大（影响恢复可用性的时间）
goosefs.master.embedded.journal.election.timeout=20s

# 在大数据量情况下，强烈建议关闭
goosefs.master.startup.block.integrity.check.enabled=false
```

```
# 触发 checkpoint 的时机，不易过小（做checkpoint会很频繁，在做 checkpoint 期间不会参与选
主），也不易过大（影响服务重启耗时），可根据checkpoint加载时间评估。
goosefs.master.journal.checkpoint.period.entries=20000000

# acl 鉴权开关，根据场景定
goosefs.security.authorization.permission.enabled=false

# 建议打开，不然会采用 hostname，而 hostname 可能相同。
goosefs.network.ip.address.used=true

# Worker properties
goosefs.worker.tieredstore.levels=1
goosefs.worker.tieredstore.level0.alias=HDD
goosefs.worker.tieredstore.level0.dirs.quota=7TB,7TB
goosefs.worker.tieredstore.level0.dirs.path=/data-1,/data-2

# worker 重启的超时时间，在大数量情况下，尽量调大。
goosefs.worker.registry.get.timeout.ms=3600s

# 读取数据响应的超时时间，默认1h
goosefs.user.streaming.data.timeout=60s

# 写策略，默认是 LocalFirstPolicy，有可能导致数据不均衡
goosefs.user.block.write.location.policy.class=com.qcloud.cos.goosefs.client.bloc
k.policy.RoundRobinPolicy

# 影响 distributedLoad 的速度，在不考虑影响在线读的情况下，建议设置为 cpu 数目 * 2，
gosefs.job.worker.threadpool.size=50
```

基于 Raft 的高可用架构部署配置（随机读场景）

在随机读的场景下，您可以参考如下推荐配置，并将该配置项复制粘贴到 goosefs-site.properties 文件中，完成您的高可用架构配置：

```
goosefs.master.embedded.journal.addresses=<master1>:9202,<master2>:9202,
<master3>:9202

goosefs.master.metastore=ROCKS
# 主要是在不确定 rocksdb 是否修复的情况下使用
goosefs.master.metastore.block=HEAP

# 根据内存大小而定
goosefs.master.metastore.inode.cache.max.size=10000000

# rocksdb 数据存放的地方
goosefs.master.metastore.dir=/meta-1/metastore
```

```
# 根目录的挂载路径，需要放在安全目录中，防止被误删除
goosefs.master.mount.table.root.ufs=/meta-1/underFSStorage

# raft 日志存放的地方
goosefs.master.journal.folder=/meta-1/journal

# 触发切主的超时时间，不易过小（jvm gc 会导致切主震荡），也不易过大（影响恢复可用性的时间）
goosefs.master.embedded.journal.election.timeout=20s

# 在大数据量情况下，强烈建议关闭
goosefs.master.startup.block.integrity.check.enabled=false

# 触发 checkpoint 的时机，不易过小（做 checkpoint 会很频繁，在做 checkpoint 期间不会参与选主），也不易过大（影响服务重启耗时），可根据checkpoint加载时间评估。
goosefs.master.journal.checkpoint.period.entries=20000000

# acl 鉴权开关，根据场景定
goosefs.security.authorization.permission.enabled=false

# 建议打开，不然会采用 hostname，而 hostname 可能相同。
goosefs.network.ip.address.used=true

# Worker properties
goosefs.worker.tieredstore.levels=1
goosefs.worker.tieredstore.level0.alias=HDD
goosefs.worker.tieredstore.level0.dirs.quota=7TB,7TB
goosefs.worker.tieredstore.level0.dirs.path=/data-1,/data-2

# worker 重启的超时时间，在大数量情况下，尽量调大。
goosefs.worker.registry.get.timeout.ms=3600s

# 读取数据响应的超时时间，默认1h
goosefs.user.streaming.data.timeout=60s

# 写策略，默认是 LocalFirstPolicy，有可能导致数据不均衡
goosefs.user.block.write.location.policy.class=com.qcloud.cos.goosefs.client.block.policy.RoundRobinPolicy

# 对于随机读情况，建议调小(默认为1MB)，防止读膨胀
goosefs.user.streaming.reader.chunk.size.bytes=256KB
goosefs.user.local.reader.chunk.size.bytes=256KB

# 等待 worker 读流关闭的时间，在大量小文件读或者随机读情况，建议调小（默认5s），避免长尾导致的性能降低
goosefs.user.streaming.reader.close.timeout=100ms
```

版权所有：腾讯云计算（北京）有限责任公司

大数据场景生产环境配置实践

最近更新时间：2024-10-12 10:20:41

简介

数据加速器 GooseFS 提供了多种部署方式，支持 自建部署，在 TKE 上部署 以及在 EMR 上部署 等。在大数据场景中，通常使用**集群模式**部署，并且采用**高可用架构**以满足业务连续性的要求，本文重点介绍基于 Zookeeper 和基于 Raft 的高可用部署配置。

高可用架构指多个 Master 节点的主备多活架构。多个 Master 节点中只有一个会作为主（Leader）节点对外提供服务，其他的备（Standby）节点通过同步共享日志（Journal）来保持与主节点相同的文件系统状态。当主节点故障宕机后，会从当前的备节点中自动选择一个接替主节点继续对外提供服务，这样就消除了系统的单点故障，实现了整体高可用架构。目前 GooseFS 支持基于 Raft 日志和 Zookeeper 两种方式来实现主备节点状态的强一致性。

基于 Zookeeper 的高可用架构部署配置

配置 Zookeeper 服务搭建 GooseFS 的高可用架构需要满足如下条件：

- 已建立 Zookeeper 集群。GooseFS Master 使用 Zookeeper 进行 Leader 选取，GooseFS 的客户端和 Worker 节点则通过 Zookeeper 查询主 Master 节点。
- 已准备好高可用强一致的共享存储系统，并且保证所有 GooseFS 的 Master 节点均可以访问到。主 Master 节点会将日志写入到该存储系统上，备（Standby）节点则会不断地从共享存储系统上读取日志，并且重放来保持与主节点的状态一致性。一般情况，该共享存储系统，推荐设置为 HDFS 或 COS。例如，hdfs://10.0.0.1:9000/goosefs/journal 或 cosn://bucket-1250000000/journal。

当您已经完成前置条件后，您可以参考如下推荐配置，并将该配置项复制粘贴到 goosefs-site.properties 文件中，完成您的高可用架构配置：

```
# GooseFS Master HA 部署配置
goosefs.zookeeper.enabled=true
goosefs.zookeeper.address=<zk_quorum_1>:<zk_client_port>,<zk_quorum_2>:
<zk_client_port>,<zk_quorum_3>:<zk_client_port>
goosefs.underfs.hdfs.configuration=${HADOOOP_HOME}/etc/hadoop/core-
site.xml:${HADOOP_HOME}/hadoop/etc/hadoop/hdfs-site.xml
goosefs.master.journal.type=UFS
goosefs.master.journal.folder=hdfs://HDFSXXXX/goosefs

# Master 元数据存储方式，推荐使用 Heap + RocksDB 方式，可以支撑上亿规模的元数据
goosefs.master.metastore=ROCKS
goosefs.master.metastore.block=ROCKS
goosefs.master.metastore.block.locations=ROCKS
# GooseFS 的元数据存储目录建议选择高 IOPS 存储介质的目录
goosefs.master.metastore.dir=/data/goosefs/metastore
#元数据交换方式，默认是 RANDOM，如果有明显最近热数据访问的，可设置为考虑 LRU；
# goosefs.master.metastore.cache.type=LRU
# 关闭启动时候校验孤儿 block 的流程，可以降低选主时间
```

```
goosefs.master.startup.block.integrity.check.enabled=false
# 同时可以根据实际情况关闭周期性地校验孤儿 block 的逻辑
# goosefs.master.periodic.block.integrity.check.interval=-1
# 如果不使用 TTL 功能，也可以考虑关闭周期性的文件过期检查
goosefs.master.ttl.checker.interval.ms=-1
# 可以考虑关闭数据副本检查，减小 Master 的开销
goosefs.master.replication.check.interval=-1

# Worker 相关的配置
goosefs.worker.tieredstore.levels=1
goosefs.worker.tieredstore.level0.alias=SSD
goosefs.worker.tieredstore.level0.dirs.path=/data1/goosefsWorker,/data2/goosefsWo
rker
# 以下 Quota 值根据实际情况设置
# goosefs.worker.tieredstore.level0.dirs.quota=2000G,2000G
goosefs.worker.block.heartbeat.interval.ms=10sec
goosefs.worker.tieredstore.free.ahead.bytes=134217728
goosefs.user.block.worker.client.pool.max=512

# 安全认证和用户模拟相关
goosefs.security.authorization.permission.enabled=true
goosefs.security.authentication.type=SIMPLE
# goosefs.security.login.username=hadoop
# goosefs.master.security.impersonation.hadoop.users=*
# goosefs.security.login.impersonation.username=_HDFS_USER_

# Client 相关的配置
goosefs.user.client.transparent_acceleration.scope=GFS_UFS
goosefs.user.client.transparent_acceleration.enabled=true
goosefs.user.file.readtype.default=CACHE
goosefs.user.file.writetype.default=CACHE_THROUGH

goosefs.user.metrics.collection.enabled=true
```

基于 Raft 的高可用架构部署配置

基于 Raft 嵌入式日志的部署方案依赖于 copycat 的 Leader 选举机制。因此，Raft 的高可用部署架构不可与 Zookeeper 相互混用。如果您计划基于 Raft 嵌入式日志搭建高可用架构，您可以参考如下推荐配置，并将该配置项复制粘贴到 goosefs-site.properties 文件中，完成您的高可用架构配置：

```
# GooseFS Master Raft 部署配置
goosefs.master.rpc.addresses=<master1>:9200,<master2>:9200,<master3>:9200
goosefs.master.embedded.journal.addresses=<master1>:9202,<master2>:9202,
<master3>:9202
#元数据 checkpoint 间隔，默认为2000000，实际速率可根据实际生产环境元数据生产速度设置
goosefs.master.journal.checkpoint.period.entries=xxxx
# GooseFS 的 Journal 数据存储位置
```

```
goosefs.master.journal.folder=/data/goosefs/journal

# Master 元数据存储方式，推荐使用 Heap + RocksDB 方式，可以支撑上亿规模的元数据
goosefs.master.metastore=ROCKS
goosefs.master.metastore.block=ROCKS
goosefs.master.metastore.block.locations=ROCKS
# GooseFS 的元数据存储目录建议选择高 IOPS 存储介质的目录
goosefs.master.metastore.dir=/data/goosefs/metastore
#元数据交换方式，默认是 RANDOM，如果有明显最近热数据访问的，可设置为考虑 LRU；
# goosefs.master.metastore.cache.type=LRU
# 关闭启动时候校验孤儿 block 的流程，可以降低选主时间
goosefs.master.startup.block.integrity.check.enabled=false
# 同时可以根据实际情况关闭周期性地校验孤儿 block 的逻辑
# goosefs.master.periodic.block.integrity.check.interval=-1
# 如果不使用 TTL 功能，也可以考虑关闭周期性的文件过期检查
goosefs.master.ttl.checker.interval.ms=-1
# 可以考虑关闭数据副本检查，减小 Master 的开销
goosefs.master.replication.check.interval=-1

# Worker 相关的配置
goosefs.worker.tieredstore.levels=1
goosefs.worker.tieredstore.level0.alias=SSD
goosefs.worker.tieredstore.level0.dirs.path=/data1/goosefsWorker,/data2/goosefsWo
rker
# 以下 Quota 值根据实际情况设置
# goosefs.worker.tieredstore.level0.dirs.quota=2000G,2000G
goosefs.worker.block.heartbeat.interval.ms=10sec
goosefs.worker.tieredstore.free.ahead.bytes=134217728
goosefs.user.block.worker.client.pool.max=512

# 安全认证和用户模拟相关
goosefs.security.authorization.permission.enabled=true
goosefs.security.authentication.type=SIMPLE
# goosefs.security.login.username=hadoop
# goosefs.master.security.impersonation.hadoop.users=*
# goosefs.security.login.impersonation.username=_HDFS_USER_

# Client 相关的配置
goosefs.user.client.transparent_acceleration.scope=GFS_UFS
goosefs.user.client.transparent_acceleration.enabled=true
goosefs.user.file.readtype.default=CACHE
goosefs.user.file.writetype.default=CACHE_THROUGH
goosefs.user.metrics.collection.enabled=true
```

数据安全
使用 Apache Ranger 控制 GooseFS 的访问权限

最近更新时间：2024-10-12 11:37:31

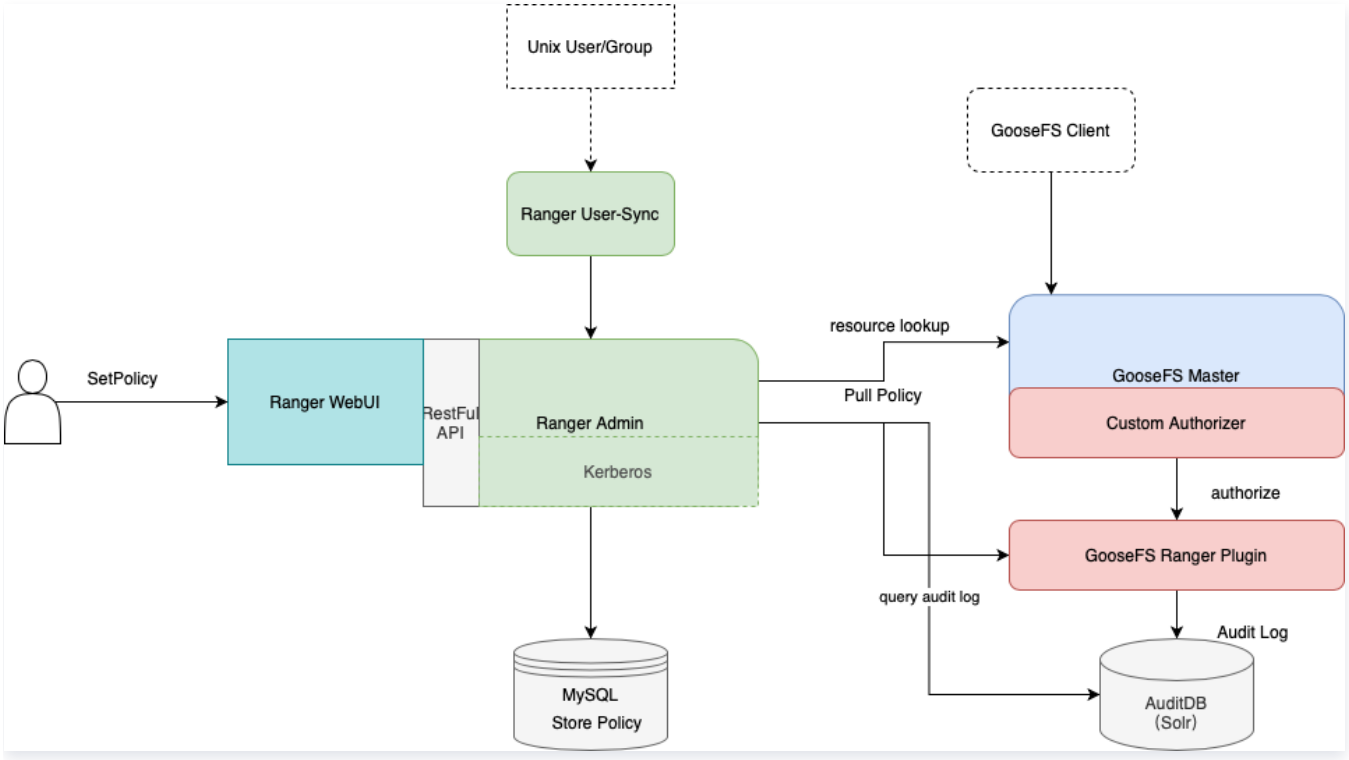
概述

Apache Ranger 是大数据生态系统中用于控制访问权限的一个标准鉴权组件，GooseFS 作为大数据和数据湖场景下的加速存储系统，也已经支持接入 Apache Ranger 的统一鉴权平台中，本文将介绍使用 Apache Ranger 控制 GooseFS 的资源访问权限。

优势

- GooseFS 作为一款云原生加速存储系统，在 Apache Ranger 支持上已经做到了与 HDFS 近乎一致的访问控制行为。因此，原先使用 HDFS 的大数据用户可以非常轻松地迁移到 GooseFS 上来，并且直接复用 HDFS 的 Ranger 权限策略，即可获得一致的使用体验。
- GooseFS with Ranger 相比 HDFS with Ranger 的鉴权架构，还额外提供了 Ranger + 原生 ACL 的联合鉴权选项，在 Ranger 鉴权失效时，还可选择使用原生 ACL 鉴权，可解决一些 Ranger 鉴权策略配置不完善的问题。

GooseFS with Ranger 的鉴权架构



为了支持将 GooseFS 集成到 Ranger 鉴权平台中，我们开发了 GooseFS Ranger Plugin，它同时部署在 GooseFS Master 节点和 Ranger Admin 侧。负责完成如下工作：

- GooseFS Master 节点侧：

- 提供 `Authorizer` 接口，为 GooseFS Master 上的每一次元数据请求提供鉴权结果。
- 连接 Ranger Admin 获取用户配置的鉴权策略。

- Ranger Admin 侧：
  - 为 Ranger Admin 提供 GooseFS 的资源查找（resource lookup）的能力。
  - 提供配置校验的能力。

开始部署

准备工作

在开始使用前，需确保环境中已部署并配置了 Ranger 相关的组件（即包括：Ranger Admin 和 Ranger UserSync），并且确保 Ranger 的 WebUI 可以正常打开和使用。

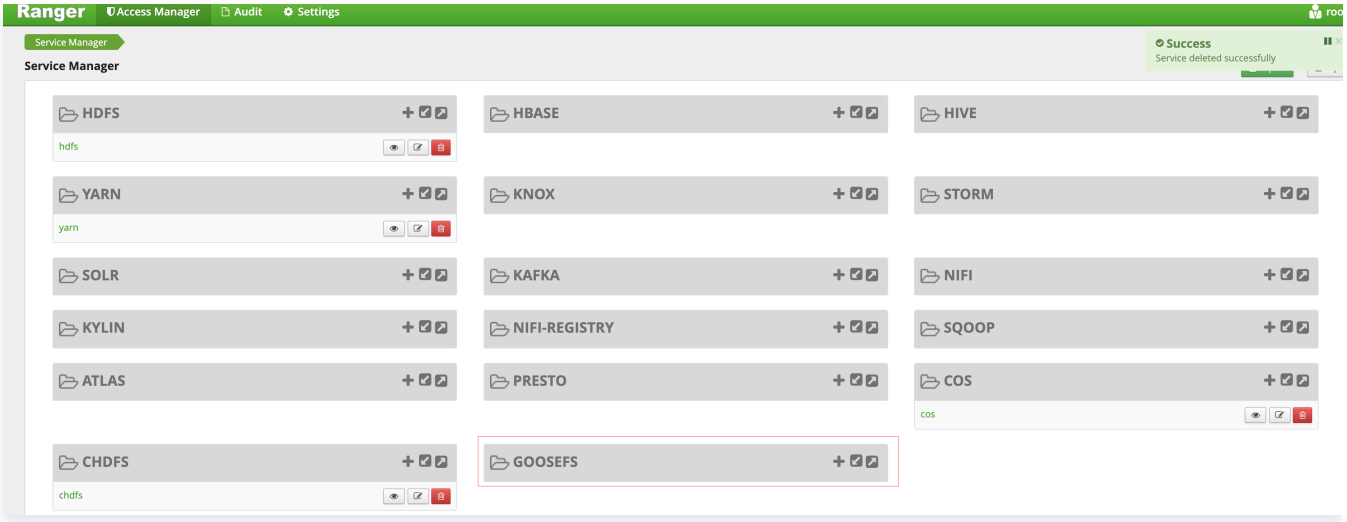
部署组件

在 Ranger Admin 侧部署 GooseFS Ranger Plugin 并注册对应服务

> **说明**
> 单击 此处 下载 GooseFS Ranger Plugin。

部署步骤如下：

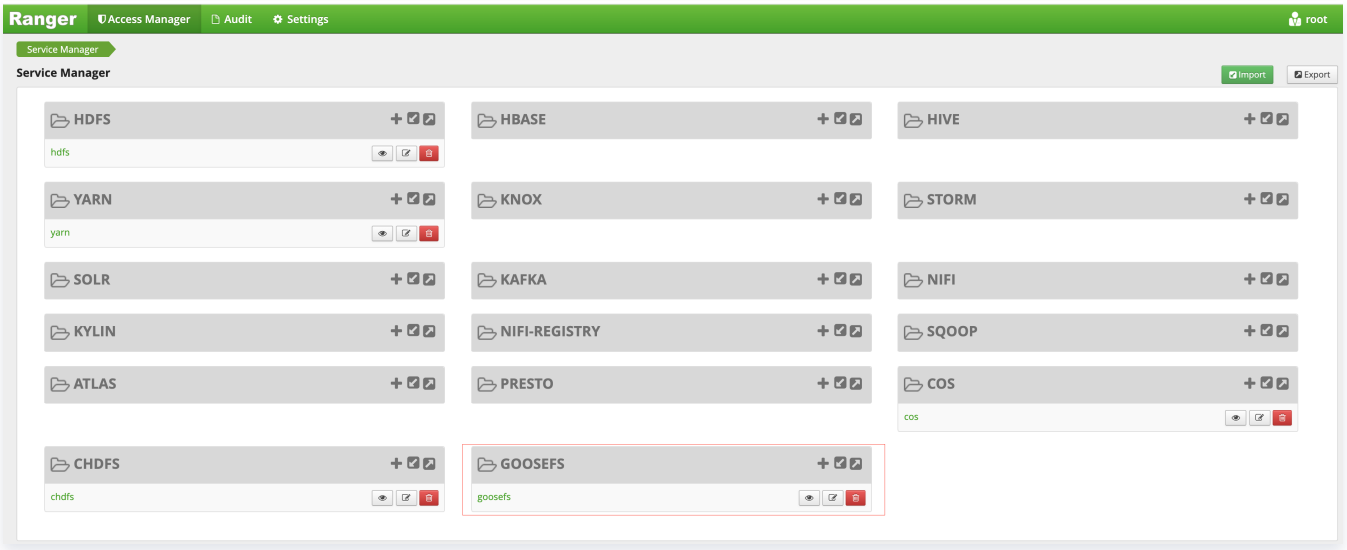
1. 在 Ranger 服务定义目录下新建 GooseFS 的目录（注意，目录权限至少保证 x 与 r 的权限）。
   1.1 如果使用的是腾讯云 EMR 集群，则 Ranger 的服务定义目录在：`/usr/local/service/ranger/ews/webapp/WEB-INF/classes/ranger-plugins`。
   1.2 如果是自建 Hadoop 集群，则可以通过在 ranger 目录下查找 hdfs 等已经接入到 ranger 服务的组件，查找目录位置。



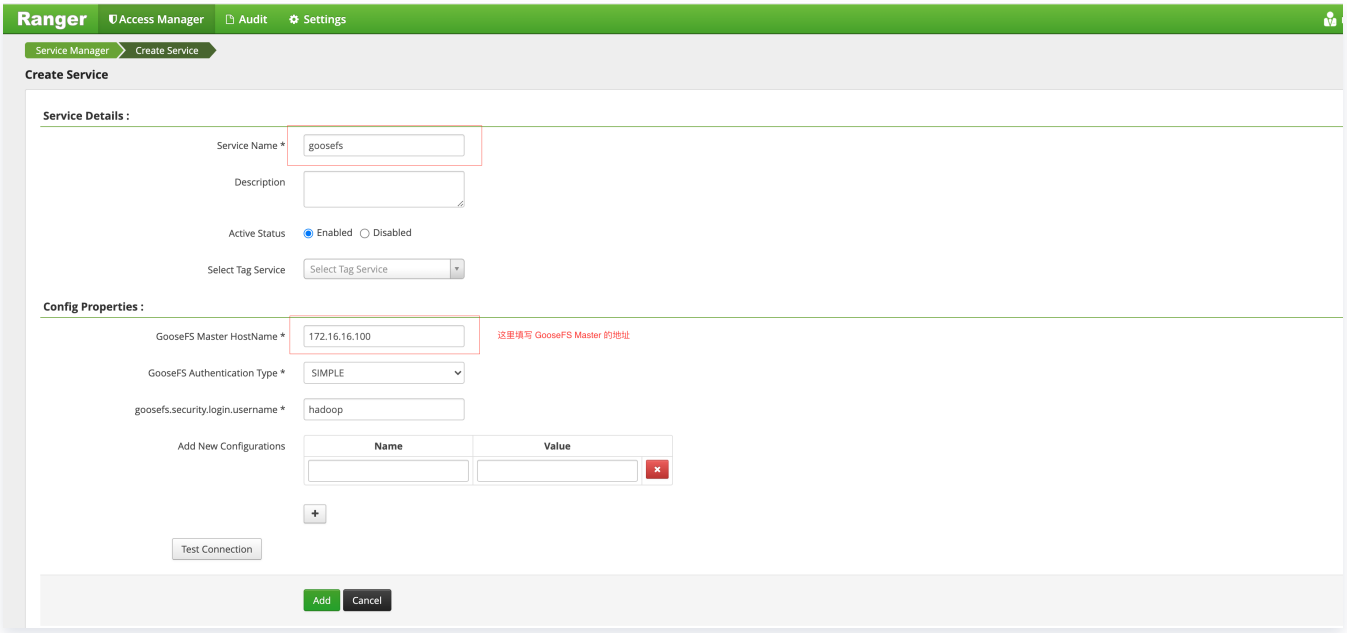
2. 在 GooseFS 的目录下，放入 goosefs-ranger-plugin-${version}.jar 和 ranger-servicedef-goosefs.json，并且具备读权限。
3. 重启 Ranger 服务。
4. 在 Ranger 上，按照如下命令，注册 GooseFS Service。

```
# 生成服务，需要传入 Ranger 管理员的账号和密码，以及 Ranger 的服务地址
# 对于腾讯云 EMR 集群，管理员用户是 root，密码是构建 EMR 集群时设置的 root 密码，ranger 服务的 IP 就是 EMR 服务的 Master IP
adminUser=root
adminPasswd=xxxx

rangerServerAddr=10.0.0.1:6080

curl -v -u${adminUser}:${adminPasswd} -X POST -H "Accept:application/json" -H "Content-Type:application/json" -d @./ranger-servicedef-goosefs.json http://${rangerServerAddr}/service/plugins/definitions

# 服务注册成功后，会返回一个服务 ID，请务必记录下这个ID
# 如果要删除 GooseFS 的服务，则传入刚刚返回的服务 ID，执行如下命令即可：
serviceId=104
curl -v -u${adminUser}:${adminPasswd} -X DELETE -H "Accept:application/json" -H "Content-Type:application/json" http://${rangerServerAddr}/service/plugins/definitions/${serviceId}
```

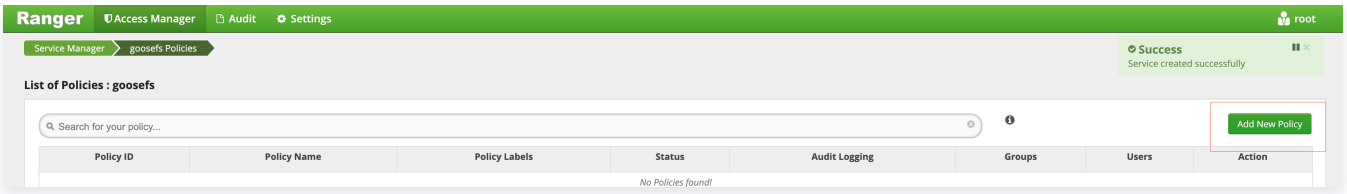
5. 创建成功后，在 Ranger 的 Web 控制台上即可看到 GooseFS 相关的服务：



6. 在 GooseFS 服务侧单击 +，定义 goosefs 服务实例。



7. 单击新生成的 goosefs 服务实例，即可添加鉴权 policy。



在 GooseFS Master 侧部署 GooseFS Ranger Plugin 并配置启用 Ranger 鉴权

1. 将 goosefs-ranger-plugin-${version}.jar 放入 ${GOOSEFS_HOME}/lib 路径下，并且至少具备读权限。
2. 将 ranger-goosefs-audit.xml、ranger-goosefs-security.xml 以及 ranger-policymgr-ssl.xml 三个文件放入 `\${GOOSEFS_HOME}/conf` 路径下，并分别填写其必要配置：
   - ranger-goosefs-security.xml：

```
<configuration xmlns:xi="http://www.w3.org/2001/XInclude">
  <property>
    <name>ranger.plugin.goosefs.service.name</name>
    <value>goosefs</value>
  </property>

  <property>
    <name>ranger.plugin.goosefs.policy.source.impl</name>
    <value>org.apache.ranger.admin.client.RangerAdminRESTClient</value>
  </property>

  <property>
    <name>ranger.plugin.goosefs.policy.rest.url</name>
    <value>http://10.0.0.1:6080</value>
  </property>
```

```
  <property>
    <name>ranger.plugin.goosefs.policy.pollIntervalMs</name>
    <value>30000</value>
  </property>

  <property>
    <name>ranger.plugin.goosefs.policy.rest.client.connection.timeoutMs</name>
    <value>1200</value>
  </property>

  <property>
    <name>ranger.plugin.goosefs.policy.rest.client.read.timeoutMs</name>
    <value>30000</value>
  </property>
</configuration>
```

- ranger-goosefs-audit.xml（不开启审计，可不配置）
- ranger-policymgr-ssl.xml

```
<configuration>
  <property>
    <name>xasecure.policymgr.clientssl.keystore</name>
    <value>hadoopdev-clientcert.jks</value>
  </property>

  <property>
    <name>xasecure.policymgr.clientssl.truststore</name>
    <value>cacerts-xasecure.jks</value>
  </property>

  <property>
    <name>xasecure.policymgr.clientssl.keystore.credential.file</name>
    <value>jceks://file/tmp/keystore-hadoopdev-ssl.jceks</value>
  </property>

  <property>
    <name>xasecure.policymgr.clientssl.truststore.credential.file</name>
    <value>jceks://file/tmp/truststore-hadoopdev-ssl.jceks</value>
  </property>
</configuration>
```

3. 在 goosefs-site.properties 文件中，添加如下配置：

```
...
goosefs.security.authorization.permission.type=CUSTOM
```

```
goosefs.security.authorization.custom.provider.class=org.apache.ranger.authori
zation.goosefs.RangerGooseFSAuthorizer
...
```

4. 在 ${GOOSEFS_HOME}/libexec/goosefs-config.sh 中，将 goosefs-ranger-plugin-${version}.jar 添加到 GooseFS 的类路径中：

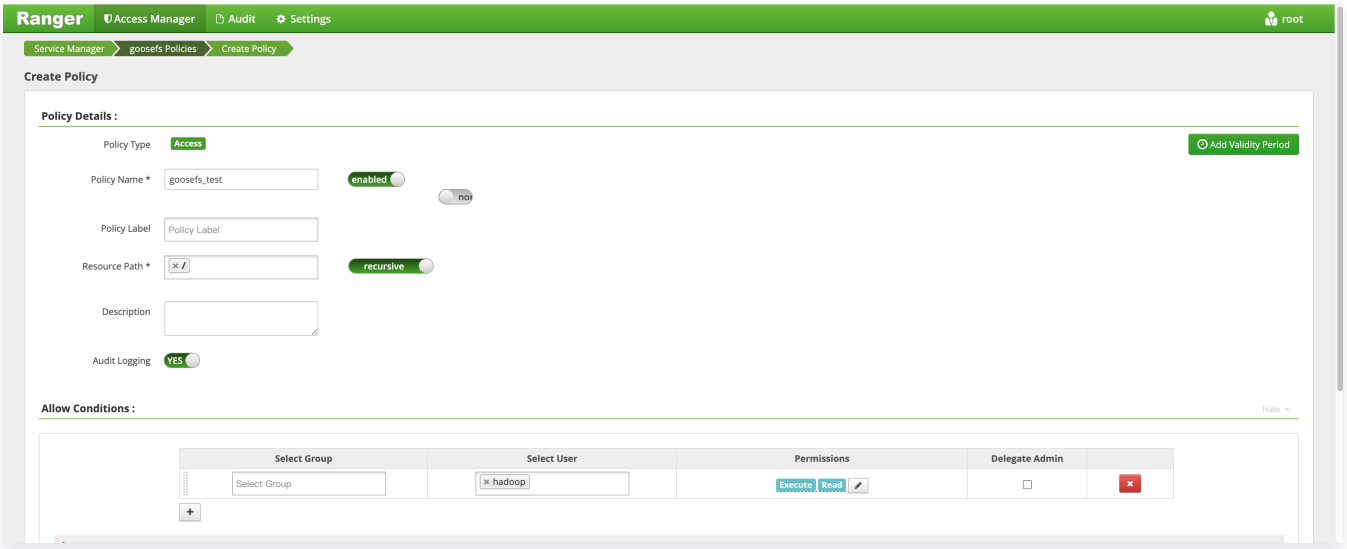
```
...
GOOSEFS_RANGER_CLASSPATH="${GOOSEFS_HOME}/lib/ranger-goosefs-
plugin-${version}.jar"
GOOSEFS_SERVER_CLASSPATH=${GOOSEFS_SERVER_CLASSPATH}:${GOOSEFS_RANGER_CLASSPAT
H}
...
```

至此，所有配置完成。

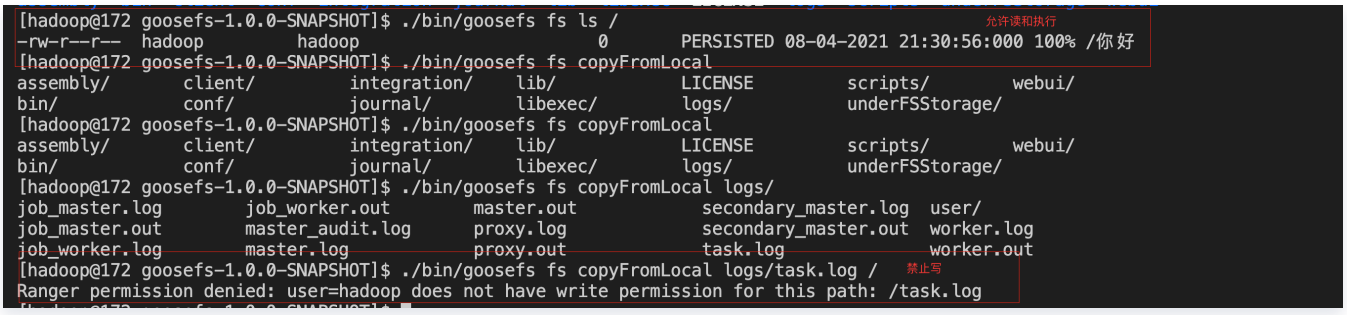
验证使用

例如，添加一条允许 hadoop 用户对 GooseFS 的根目录具备读取和执行权限，但是不允许写的策略，方法如下：

1. 添加策略，如下所示：



2. 策略添加成功后，对策略进行验证，即可看到策略已经生效，如下所示：



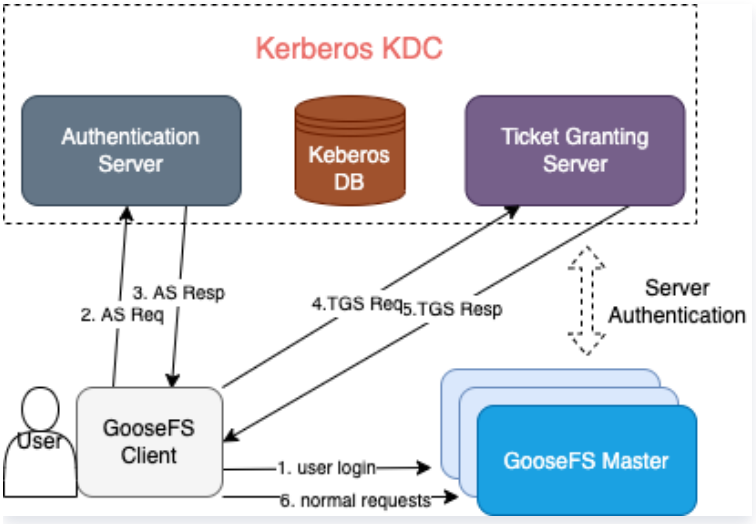
GooseFS 接入 Kerberos 认证

最近更新时间：2023-11-16 09:48:42

概述

Kerberos 是大数据生态系统中广泛应用的统一认证服务，GooseFS 作为大数据和数据湖场景下的加速存储服务，支持集群节点和用户访问接入 Kerberos 认证服务中。本文将详细介绍如何配置 GooseFS 接入 Kerberos 认证服务以及如何使用 Hadoop Delegation Token 认证。

GooseFS 接入 Kerberos 的认证架构



GooseFS Kerberos 认证优势

- 与 HDFS 接入 Kerberos 的认证架构和流程基本一致，在 HDFS 上启用了 Kerberos 认证流程的应用可以很容易地迁移到 GooseFS。
- 支持 Hadoop 的 Delegation Token 认证机制，因此可以很好地兼容 Hadoop 生态的应用作业。

配置 GooseFS 接入 Kerberos 认证

前提条件

- GooseFS 1.3.0 及以上版本；
- JDK 1.8 的环境，JDK 11 及以上暂时不兼容；
- 确保环境中已存在 Kerberos KDC 服务，并且 GooseFS 以及应用客户端能够正常访问 Kerberos KDC 服务相关端口。

在 Kerberos KDC 中创建 GooseFS 相关的身份信息

首先，我们需要在 Kerberos KDC 中创建 GooseFS 集群 Server 与 Client 相关的 Kerberos 身份信息，然后才能继续后续的接入配置。这里我们在 **Kerberos KDC 服务器**上借助 `kadmin.local` 交互式工具来完成创建：

> ⚠ **注意**

> `kadmin.local` 工具需要 root/sudo 权限执行。

```
$ sudo kadmin.local
```

如果执行成功，则进入了 kadmin 的交互式 shell 环境中：

```
Authenticating as principal root/admin@GOOSEFS.COM with password.
kadmin.local:
```

其中，`kadmin.local` 表示该交互式执行环境的命令提示符。

创建 GooseFS Server / Client 相关的身份信息

如下通过一个简单测试集群以及应用场景样例来介绍整个 Kerberos 的配置过程。

1. 集群环境说明

   采用单 Master 双 Worker 的 Standalone 部署架构：

   - Master（JobMaster）：172.16.0.1
   - Worker1（JobWorker1）：172.16.0.2
   - Worker2（JobWorker2）：172.16.0.3
   - Client：172.16.0.4

2. 在 `kadmin.local` 中创建 Server 和 Client 身份认证相关信息：

   ```
   kadmin.local: addprinc -randkey goosefs/172.16.0.1@GOOSEFS.COM
   kadmin.local: addprinc -randkey client/172.16.0.4@GOOSEFS.COM
   ```

   > ⚠ **注意**
   >
   > 此处使用 `-randkey` 选项的原因在于 GooseFS 无论 Server 还是 Client 登录均使用 `keytab` 文件认证，不使用明文密码。若身份信息需要用于 password 登录场景，则可以去掉该选项。

3. 生成导出每个身份对应的 `keytab` 文件：

   ```
   kadmin.local: xst -k goosefs_172_16_0_1.keytab goosefs/172.16.0.1@GOOSEFS.COM
   kadmin.local: xst -k client_172_16_0_4.keytab client/172.16.0.4@GOOSEFS.COM
   ```

配置 GooseFS Server/Client 接入使用 Kerberos 认证

1. 将上述导出 `keytab` 文件分发到对应的机器上，此处建议的路径是 `${GOOSEFS_HOME}/conf/`：

   ```
   $ scp goosefs_172_16_0_1.keytab <username>@172.16.0.1:${GOOSEFS_HOME}/conf/
   $ scp goosefs_172_16_0_1.keytab <username>@172.16.0.2:${GOOSEFS_HOME}/conf/
   $ scp goosefs_172_16_0_1.keytab <username>@172.16.0.3:${GOOSEFS_HOME}/conf/
   $ scp client_172_16_0_4.keytab <username>@172.16.0.4:${HOME}/.goosefs/
   ```

2. 在对应机器上，将 Server Principal KeyTab 文件的所属用户/用户组修改为 GooseFS Server 启动时所使用的用户及用户组（其目的是为了让 GooseFS 启动时有足够的权限读取）。

```
$ chown <GooseFS_USER>:<GOOSEFS_USERGROUP> goosefs_172_16_0_1.keytab
$ # 同时修改 Unix 访问权限位
$ chmod 0440 goosefs_172_16_0_1.keytab
```

3. 将 Client 的 KeyTab 的所属用户/用户组修改为发起 GooseFS 请求的客户端账户。其目的同样是为了保证 Client 有足够的权限读取该文件。

```
$ chown <client_user>:<client_usergroup> client_172_16_0_4.keytab
$ # 同时修改 Unix 访问权限位
$ chmod 0440 client_172_16_0_4.keytab
```

Server 和 Client 端 GooseFS 配置

1. Master/Worker Server 的 goosefs-site.properties

```
# Security properties
# Kerberos properties
goosefs.security.authorization.permission.enabled=true
goosefs.security.authentication.type=KERBEROS
goosefs.security.kerberos.unified.instance.name=172.16.0.1
goosefs.security.kerberos.server.principal=goosefs/172.16.0.1@GOOSEFS.COM
goosefs.security.kerberos.server.keytab.file=${GOOSEFS_HOME}/conf/goosefs_172_
16_0_1.keytab
```

配置完成 GooseFS Server 端的认证配置后，重启整个集群以使得配置生效。

2. Client 的 goosefs-site.properties

```
# Security properties
# Kerberos properties
goosefs.security.authorization.permission.enabled=true
goosefs.security.authentication.type=KERBEROS
goosefs.security.kerberos.unified.instance.name=172.16.0.1
goosefs.security.kerberos.server.principal=goosefs/172.16.0.1@GOOSEFS.COM
goosefs.security.kerberos.client.principal=client/172.16.0.4@GOOSEFS.COM
goosefs.security.kerberos.client.keytab.file=${GOOSEFS_HOME}/conf/client_172_1
6_0_4.keytab
```

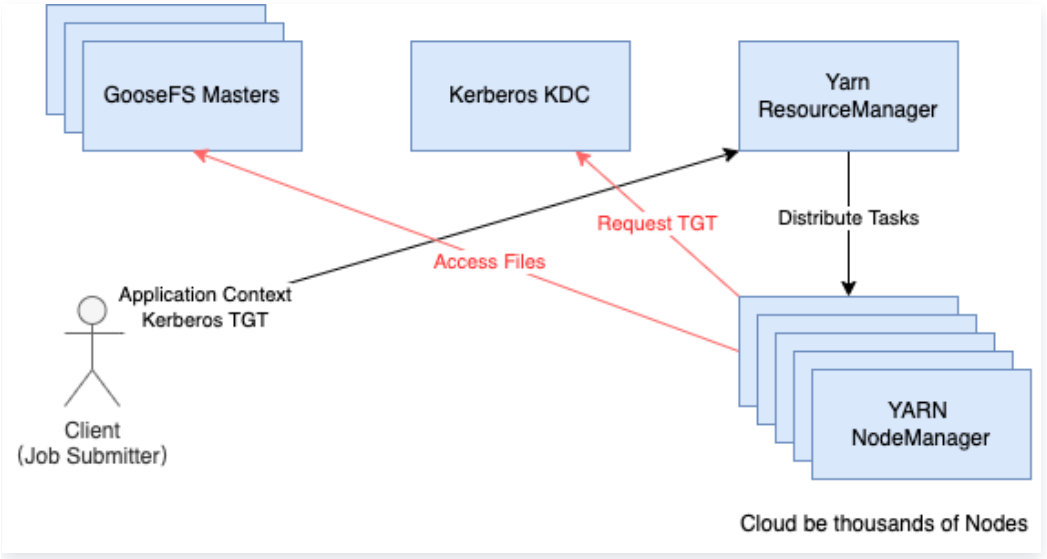
> ⚠ 注意

> Client 端需要指定 Server 的 principal。其原因为在 Kerberos 认证体系中，KDC 需要知道当前 Client 所访问的 Service，而 GooseFS 是通过 Server 的 principal 区分 Client 当前请求的 Service。

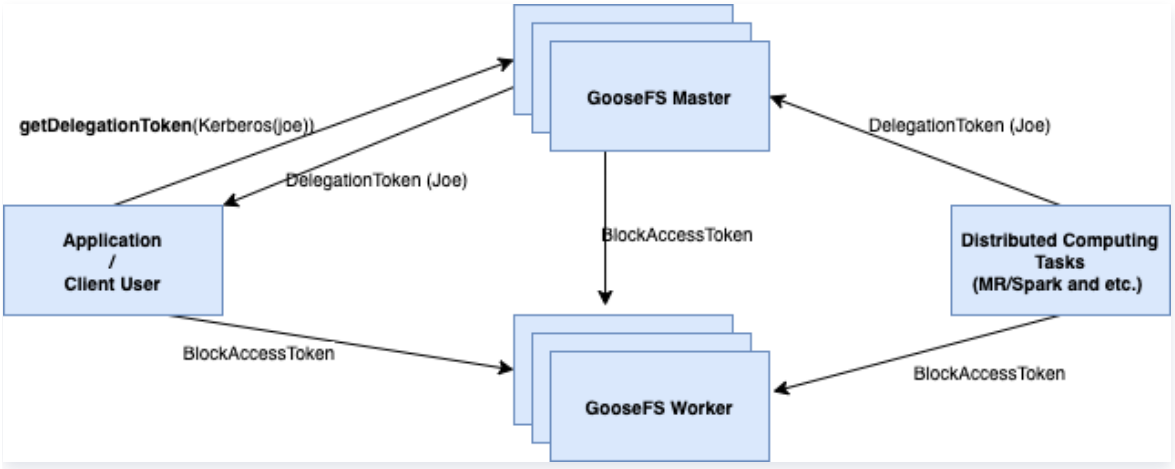
至此，GooseFS 接入 Kerberos 的基础认证完毕，后续客户端发起的请求都将经过 Kerberos 进行身份认证。

GooseFS 接入 Hadoop Delegation Token 认证

虽然理论上可以单独使用 Kerberos 进行身份认证，但在 Hadoop 这样的大规模分布式系统中，如果对于每个 MapReduce 作业，所有工作任务都需要请求 TGT，那么 Kerberos 的 KDC 一定会成为整个系统运行的瓶颈。如下图所示：



如果一个 YARN 集群中 NodeManager 节点数达到几百上千个时，同时并行运行了多个 IO 密集型作业任务，那么每个执行节点在正常访问 GooseFS 中的文件所需向 Kerberos KDC 发起的凭据请求会产生巨大的 DDos 攻击。
因此，Hadoop 社区补充设计了 Delegation Token 这种轻量级的认证机制来避免这种大规模且频繁的 Kerberos 认证请求。Kerberos 是三方认证，而 Delegation Token 在完成提交作业时的首次 Kerberos 认证以后，Master 会颁发一个 Delegation Token 给客户端缓存住，后续的每个请求都可以通过出示 Delegation Token 来认证，因此 Delegation Token 认证机制只涉及两方。具体的认证过程如下：



1. 在提交向 YARN 提交一个计算作业的时候，YARN Client（这里面也加载 GooseFS 的Client）会通过 Kerberos 认证以后，向 GooseFS 的 Master 申请一个 Delegation Token，然后添加到 YARN 的 Resource Manager 的 Delegation Token 更新服务中；

2. 当 YARN 的 ResourceManager（实际是 ApplicationMaster ）把分派给 NodeManager 创建的 Container 开始执行以后，每个 Task 都会携带这个 Delegation Token 去访问 GooseFS 集群。值的注意的是，由于是两方认证，因此 Client 不仅需要与 GooseFS Master 完成认证，还会隐式携带 BlockAccessToken去访问 Worker 集群，BlockAccessToken 是在每次访问一个文件时，由 Master 返回，其中会携带认证所需的身份信息和权限信息，以便于 Worker 判断是否允许 Client 读写指定的 Block；
3. YARN 的 Delegation Token 更新服务会定时更新 Token，直到 Token 的最大生命周期结束。

注意：不同于 Kerberos 等通用的三方认证方案，Delegation Token 认证机制一定是 Hadoop 生态所特有的，因此只能使用在 Hadoop 生态的大数据环境中，且每个 Delegation Token 都有最大生命周期限制，因此对于常驻作业（例如实时流计算作业），应当谨慎设置 Delegation Token 的最大生命周期，一般默认是 7 天。

Delegation Token 相关的配置

如果需要使用 Delegation Token，除了需要基本的 Hadoop 的 Kerberos 和 Delegation Token 的认证环境以外，**还需要满足使用的是 GooseFS 1.4.5 及以上的版本，同时在** `${GOOSEFS_HOME}/conf/goosefs-site.properties` **中添加如下配置：**

```
# Security properties
# Kerberos properties
goosefs.security.authorization.permission.enabled=true
goosefs.security.authentication.type=KERBEROS
goosefs.security.kerberos.unified.instance.name=172.16.0.1
goosefs.security.kerberos.server.principal=goosefs/172.16.0.1@GOOSEFS.COM
goosefs.security.kerberos.client.principal=client/172.16.0.1@GOOSEFS.COM
goosefs.security.kerberos.client.keytab.file=${GOOSEFS_HOME}/conf/client_172_16_0
_1.keytab

# hadoop token相关配置
goosefs.security.authentication.block.access.token.enabled=true
goosefs.security.delegation.token.lifetime.ms=7d
goosefs.security.delegation.token.renew.interval.ms=1d
```

然后，将所有配置分发到 Master/Worker 和 Client 的节点上即可启用 Hadoop Delegation Token 的认证配置。

> ⚠ **注意：**在开启了 Hadoop Delegation Token 认证的环境中，如果中途关掉 Hadoop Delegation Token，由于 Master 已经将 DelegationTokenManager 的信息写入到元数据的 Checkpoint 和 Journal 中，因此需要重新 Format 集群，才能完成关闭 Hadoop Delegation Token 重启 GooseFS 集群生效。

使用示例

以下是一个示例集群和作业简单演示一下 Hadoop Delegation Token 生效和生命周期的测试。
测试环境：

- EMR V3.6.0 环境，默认开启了 Kerberos 认证，开启高可用架构；
- GooseFS-1.4.5 版本，GooseFS-1.4.5 的 client 放到了 ${HADOOP_HOME}/share/hadoop/common/lib 下面；

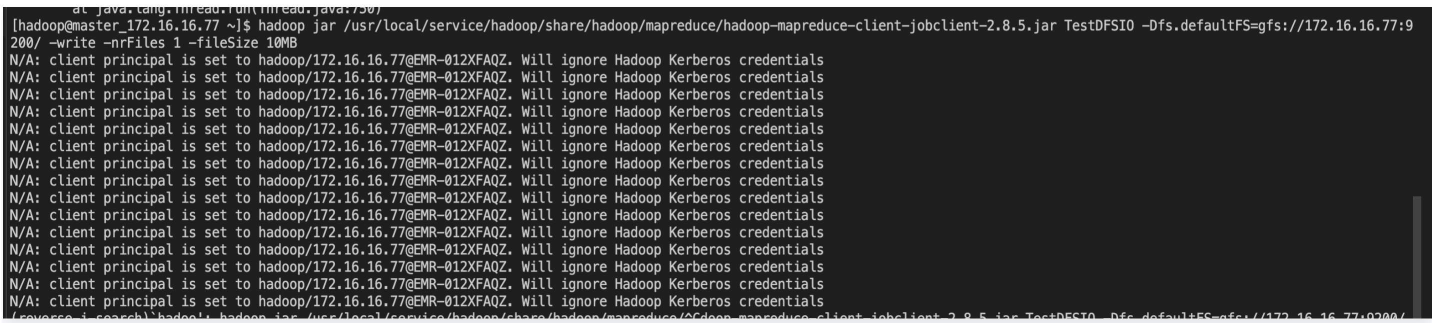
配置如下：

```
goosefs.security.authorization.permission.enabled=true
# kerberos
goosefs.security.authentication.type=KERBEROS
goosefs.security.kerberos.unified.instance.name=172.16.16.77
goosefs.security.kerberos.server.principal=hadoop/172.16.16.77@EMR-012XFAQZ
goosefs.security.kerberos.server.keytab.file=/usr/local/service/goosefs/conf/emr.
keytab
goosefs.security.kerberos.client.principal=hadoop/172.16.16.77@EMR-012XFAQZ
goosefs.security.kerberos.client.keytab.file=/usr/local/service/goosefs/conf/emr.
keytab
# delegation token
goosefs.security.authentication.block.access.token.enabled=true
goosefs.security.delegation.token.lifetime.ms=7d
goosefs.security.delegation.token.renew.interval.ms=1d
```

然后，可以执行一个 TestDFSIO：

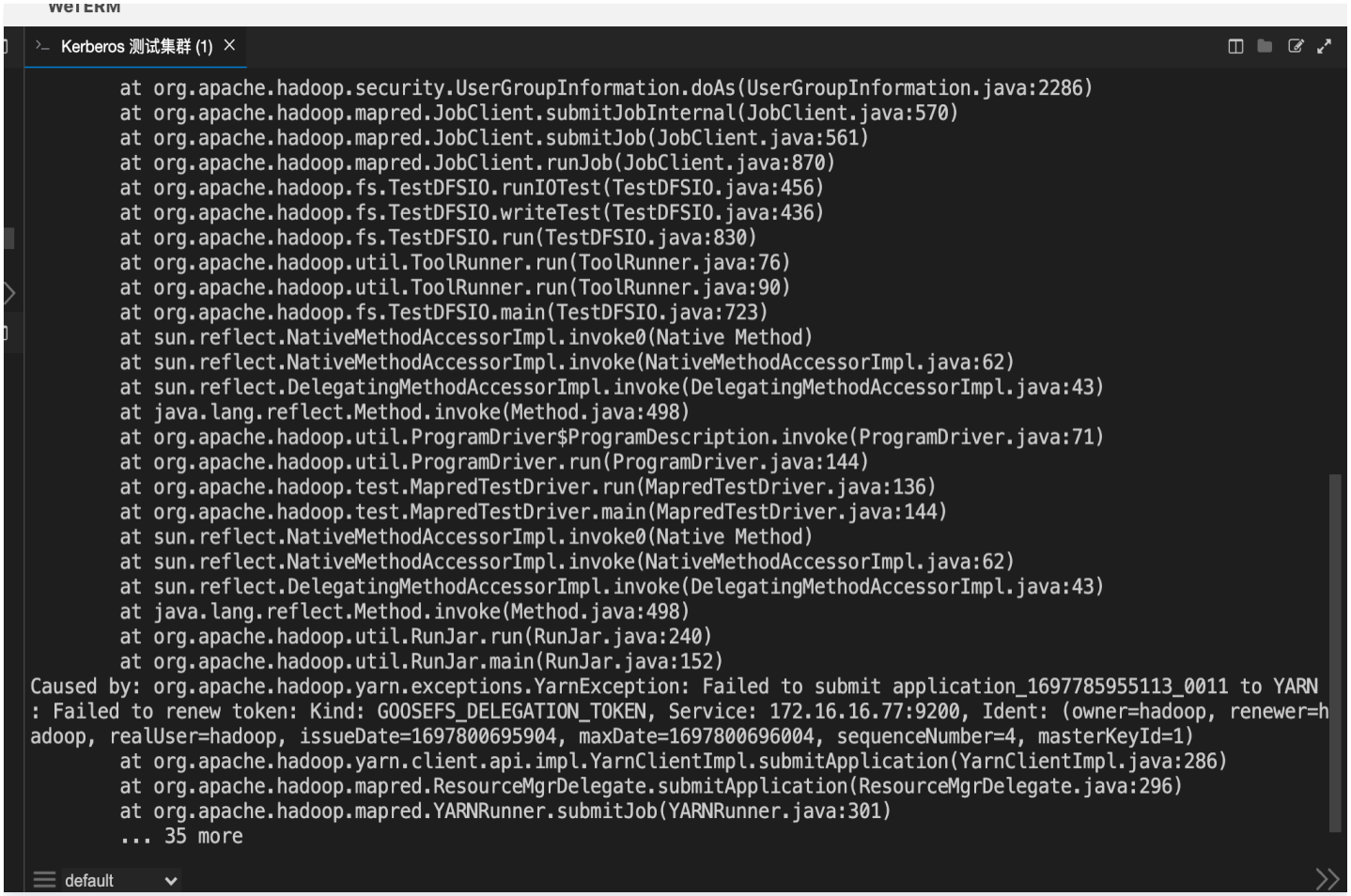
```
hadoop jar /usr/local/service/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-
client-jobclient-2.8.5.jar TestDFSIO -Dfs.defaultFS=gfs://172.16.16.77:9200/ -
write -nrFiles 1 -fileSize 10MB
```

由于 Token 的生命周期和刷新周期都是默认的，且在整个作业的运行周期范围内，因此预期是可以正常运行的，如下所示：



```
[hadoop@master_172.16.16.77 ~]$ hadoop jar /usr/local/service/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-2.8.5.jar TestDFSIO -Dfs.defaultFS=gfs://172.16.16.77:9
200/ -write -nrFiles 1 -fileSize 10MB
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
N/A: client principal is set to hadoop/172.16.16.77@EMR-012XFAQZ. Will ignore Hadoop Kerberos credentials
```

然后修改 Token 的生命周期，修改到一个很小的值，例如100ms，注意需要重启所有的 GooseFS Server，然后再执行上述测试命令，预期是会报错：



如上图所示，因为 Token 的生命周期被设置到了一个很小的值，不足以运行完测试作业，因此报了 Token 过期，且不可再续期的错误。

服务等级协议

最近更新时间：2024-10-14 10:28:11

详细信息，请参见 数据加速器 GooseFS 服务等级协议 。