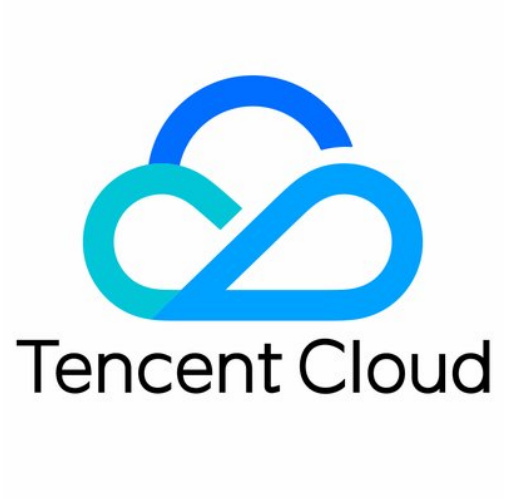
Cloud Object Storage

데이터 레이크 스토리지

제품 문서



©2013-2022 Tencent Cloud. All rights reserved.

Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice

Tencent Cloud

All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

목록:

©2013-2022 Tencent Cloud. All rights reserved.

Kerberos 인증에 GooseFS 연결

©2013-2022 Tencent Cloud. All rights reserved.

데이터 레이크 스토리지
클라우드 네이티브 데이터 레이크
시작하기

최종 업데이트 날짜: : 2023-09-18 09:55:47

소개

클라우드 네이티브 데이터 레이크 스토리지를 사용하면 TKE(Tencent Kubernetes Engine)에 COS(Cloud Object Storage) 기반 데이터 레이크 스토리지 서비스를 빠르게 배포할 수 있으며, 그 다음 TKE 또는 EKS 클러스터에서 다양한 비즈니스에 필요한 빅 데이터 및 AI 서비스 응용 프로그램을 배포할 수 있습니다. 또한 GooseFS를 사용하여 대규모 분산 스토리지 서비스에 연결할 수도 있습니다.

개념 및 용어

다음은 클라우드 네이티브 데이터 레이크 스토리지의 몇 가지 기본 **개념 및 용어**입니다.

- **환경**: 컴퓨팅 클러스터와 스토리지 서비스 간의 매핑을 유지합니다. 환경에서 컴퓨팅 클러스터 및 스토리지 서비스를 균일하게 관리하는 것이 좋습니다.

  > 주의 :
  > TKE 콘솔에서 컴퓨팅 클러스터를 삭제해야 하는 경우 먼저 데이터 레이크 환경을 지우는 것이 좋습니다.

- **컴퓨팅 클러스터**: 다양한 컴퓨팅 비즈니스를 운영하기 위한 컨테이너 클러스터입니다. TKE 또는 EKS 클러스터를 생성할 수 있습니다.
- **스토리지 서비스**: 컴퓨팅을 위해 다양한 유형의 데이터를 저장하는 COS를 말합니다.
- **애플리케이션 마켓**: Flink 및 Spark와 같은 다양한 컴퓨팅 비즈니스를 위한 애플리케이션 구성 요소가 있습니다. 환경을 생성할 때 필요에 따라 애플리케이션을 선택할 수 있습니다.

  > 주의 :
  > 컨테이너 클러스터가 종료되면 해당 클러스터에 배포된 애플리케이션도 종료됩니다. 유의하여 진행하십시오.

- **GooseFS**: 다양한 기본 버킷을 관리하고 컴퓨팅 클러스터에서 자주 액세스하는 데이터를 캐시하여 컴퓨팅을 가속화합니다.

©2013-2022 Tencent Cloud. All rights reserved.

다음 문서에서 몇 가지 기본 정보를 얻을 수 있습니다.

- COS: 시작하기에서 버킷을 생성하고 버킷에서 파일을 업로드/다운로드하는 방법을 설명합니다.
- TKE: 시작하기에서 TKE 또는 EKS 클러스터를 생성하는 방법을 설명합니다.
- 애플리케이션 시장: Application Market은 TKE 클러스터에서 애플리케이션을 생성하고 배포하는 방법을 설명합니다.
- GooseFS: GooseFS on TKE 클라우드 네이티브 활용은 클러스터에서 GooseFS를 관리하는 방법을 설명합니다.

전제 조건

- 현재 클라우드 네이티브 데이터 레이크 스토리지는 얼로우리스트를 통해 제공됩니다. 사용하시려면 문의하기에서 신청하십시오.
- 클라우드 네이티브 데이터 레이크 스토리지는 TKE 및 COS에 의존하며 **컴퓨팅 및 스토리지 서비스 작업 권한**이 필요합니다. 서브 계정으로 로그인하는 경우 서브 계정에 최소한 다음 권한이 있는지 확인하십시오.
- COS 버킷 및 파일을 조작할 수 있는 권한입니다.
  - 버킷 조작 권한: 버킷 구성을 관리해야 하는 경우 루트 계정에서 해당 권한을 얻습니다. 일반적으로 이 권한은 데이터 읽기/쓰기에 영향을 미치지 않으며 추가 구성이 필요하지 않습니다. QcloudCOSBucketConfigRead 정책과 같은 읽기 권한만 부여하면 됩니다.
  - 파일 조작 권한: 일반적으로 컴퓨팅 작업을 수행하려면 버킷에서 파일을 읽거나 써야 합니다. 루트 계정에서 QcloudCOSDataFullControl 정책과 같은 전체 액세스 권한을 얻을 수 있습니다. 또는 루트 계정은 최소 권한의 원칙 설명에 따라 권한을 부여할 수 있습니다.
- 컨테이너 클러스터 관리 권한:
  - 클러스터 작업 권한: 일반적으로 클러스터 생성 및 조작 권한을 부여해야 합니다. 자세한 지침은 Using TKE Preset Policy Authorization을 참고하십시오.
  - 클러스터 관리 권한: TKE는 Kubernetes RBAC에 연결할 수 있는 권한 부여 모드를 제공하므로 서브 계정 액세스를 세분화된 방식으로 제어할 수 있습니다. 서브 계정 작업에도 TKE Kubernetes 객체 레벨 권한 제어가 적용됩니다.
  - 애플리케이션 마켓 조작 권한: 애플리케이션 마켓은 TCR 서비스의 운영에 의존합니다. 서브 계정을 인증하는 방법에 대한 자세한 지침은 TKE Image Registry Resource-level Permission Settings를 참고하십시오.

작업 단계

다음은 환경 생성, 클러스터 연결, 컴퓨팅 애플리케이션 배포, 스토리지 서비스 연결 및 환경 관리를 포함한 단계를 자세히 설명합니다.

1. COS 콘솔에 로그인합니다.
2. 왼쪽 사이드바에서 **클라우드 네이티브 데이터 레이크 스토리지**를 클릭합니다.

©2013-2022 Tencent Cloud. All rights reserved.

3. 클라우드 네이티브 데이터 레이크 스토리지 페이지에서 기능 개요 및 배포 가이드를 볼 수 있습니다.

- 배포 가이드는 기본적으로 표시되며, 오른쪽 상단 모서리의 **가이드 닫기**를 클릭하여 사용하지 않도록 설정할 수 있습니다.
- 환경 목록 페이지에서 검색이 가능합니다. 다음과 같이 기존 환경을 조작할 수 있습니다.
  - **환경 이름**을 클릭하면 환경 세부 정보 페이지로 이동하여 환경을 관리할 수 있습니다.
  - **관련 클러스터**를 클릭하여 TKE 콘솔에서 클러스터 세부 정보 페이지로 들어갑니다.
  - 관련 버킷을 클릭하면 버킷 페이지로 이동하여 파일 정보를 볼 수 있습니다.

4. **환경 생성**을 클릭합니다.
   환경을 생성하기 전에 대상 컨테이너 컴퓨팅 클러스터를 선택하고 다음 매개변수를 구성합니다.

- **환경 이름**: 최대 63자를 포함할 수 있으며 전역적으로 고유해야 합니다.
- **리전**: 컨테이너 클러스터의 리전을 선택합니다.
- **클러스터 유형**: TKE 또는 EKS일 수 있습니다. 현재 리전에 클러스터가 없는 경우 **컨테이너 클러스터 생성**을 클릭하여 TKE 콘솔에서 클러스터를 생성할 수 있습니다.
- **클러스터**: **지정된 리전** 및 **지정된 클러스터 유형** 조건에 따라 컴퓨팅 애플리케이션을 배포하고 컴퓨팅 작업을 실행하기 위한 클러스터의 이름입니다.
- **컴퓨팅 애플리케이션**: 컴퓨팅 작업을 실행하는 데 필요한 애플리케이션 서비스를 나타냅니다. 현재 Flink, big-data-suite, colocation, airflow, pytorch 및 spark-operator 애플리케이션이 기본적으로 지원됩니다. 필요에 따라 하나 또는 여러 개의 애플리케이션을 선택할 수 있습니다. 사용자 지정 애플리케이션을 배포하려면 TKE 콘솔로 이동하여 직접 배포할 수 있습니다.

5. **다음**을 클릭하여 **버킷 구성** 페이지로 이동합니다.
   이 페이지에서 컴퓨팅 클러스터에 대해 다른 버킷을 구성할 수 있습니다. 기본적으로 GooseFS는 컴퓨팅 가속을 위해 컴퓨팅 클러스터의 로컬 노드에서 버킷 및 캐싱 데이터를 관리하는 데 사용할 수 있습니다. 다음 매개변수를 구성해야 합니다.

- 리전: 기본적으로 컴퓨팅 클러스터의 리전이며 편집할 수 없습니다. 리전에 컴퓨팅 작업에 사용할 수 있는 버킷이 없는 경우 **버킷 생성**을 클릭하여 버킷을 생성할 수 있습니다.
- 버킷: 지정된 리전에서 여러 버킷을 선택할 수 있습니다. 버킷의 지정된 파일 디렉터리만 마운트할 수도 있습니다.

> 주의 :
> 전체 버킷을 마운트하는 경우 두 번째 입력 상자를 무시할 수 있습니다. 디렉터리를 지정해야 하는 경우 `prefix/*` 형식으로 디렉터리 이름을 입력합니다.

©2013-2022 Tencent Cloud. All rights reserved.

- GooseFS 활성화: GooseFS는 컴퓨팅 작업을 가속화합니다. 기본적으로 활성화되어 있으며 수정할 수 없습니다. 추가 비용은 발생하지 않습니다.

6. **다음**을 클릭하여 **GooseFS 애플리케이션 구성** 페이지로 이동합니다.
   데이터 레이크 환경에서 모든 컴퓨팅 작업은 GooseFS를 통해 COS에 액세스해야 합니다. 따라서 지정된 버킷의 secretId 및 secretKey에 액세스할 수 있는 권한을 GooseFS에 부여해야 합니다.
7. **다음**을 클릭하고 정보를 확인합니다.
8. 구성 항목을 수정하려면 **수정**을 클릭합니다. 모든 것이 올바른지 확인한 후 **환경 생성**을 클릭합니다. 그 다음 **환경 목록으로 돌아가서 새로고침**하면 새로 생성된 환경을 볼 수 있습니다.

환경을 삭제하려면 환경 목록에서 **삭제**를 클릭하고 팝업 창에서 삭제를 확인합니다.
9. 목록에서 환경 이름을 클릭하여 **기본 정보** 페이지로 이동합니다.
환경, 컴퓨팅 클러스터 및 버킷 정보를 설명하는 세 가지 보기를 사용할 수 있습니다.

- 환경 정보: 환경의 이름, 리전, 연결된 컴퓨팅 클러스터, 스토리지 서비스, 생성 시간을 표시합니다.
- 컴퓨팅 클러스터 정보: 컴퓨팅 클러스터의 이름, 노드 수, CPU, 메모리, GPU 사용량을 표시합니다. 세부 정보 보기를 클릭하여 TKE 콘솔에 들어가 컴퓨팅 클러스터 세부 정보를 볼 수 있습니다.
- 버킷 정보: 컴퓨팅 클러스터와 연결된 버킷의 이름, 파일 URL, GooseFS 상태를 표시합니다. 세부 정보 보기를 클릭하여 스토리지 서비스의 세부 정보를 볼 수 있습니다.

이제 데이터 레이크 환경 생성을 완료했습니다.

©2013-2022 Tencent Cloud. All rights reserved.

메타데이터 가속
메타데이터 가속 기능 개요

최종 업데이트 날짜: : 2022-09-28 14:50:08

메타데이터 가속 기능은 Tencent Cloud Cloud Object Storage(COS) 서비스에서 제공하는 고성능 파일 시스템 기능입니다. 메타데이터 가속 기능의 기본 레이어는 클라우드 HDFS의 우수한 메타데이터 관리 기능을 사용하여, 사용자가 파일 시스템의 시맨틱을 통해 객체 스토리지 서비스에 액세스할 수 있도록 지원합니다. 시스템 설계 지표는 2.4Gb/s 대역폭, 10만 규모 QPS, ms 단위 딜레이에 달합니다. 메타데이터 가속 기능이 활성화되면 버킷은 빅 데이터, 고성능 컴퓨팅, 머신러닝, AI와 같은 시나리오에서 널리 사용될 수 있습니다.

메타데이터 가속 기능을 활성화하지 않는 버킷은 기본적으로 메타데이터 액세스 가속 효과가 없고, POSIX 파일 시맨틱 액세스를 지원하지 않으며, 파일 LIST 성능에 병목 현상이 있고, 파일 RENAME 작업을 지원하지 않습니다. 메타데이터 가속 기능 활성화 후 네이티브 COS API를 통해 POSIX 파일 시맨틱 체계로 버킷의 파일에 액세스하거나, 클라이언트 측에 배포된 Hadoop 툴, 콘솔 또는 SDK를 통해 POSIX 파일 시맨틱 체계로 버킷의 파일에 액세스할 수 있습니다.

> 주의 :
> 메타데이터 가속 기능은 버킷이 생성되어야만 활성화할 수 있으며, 활성화한 후에는 비활성화할 수 없습니다. 필요에 따라 활성화 여부를 **신중히 선택하시기 바랍니다.**

사용 제한

다음 표는 버킷의 메타데이터 가속 기능 활성화 후 관련 제품 기능의 지원 현황을 보여줍니다.

지표 항목	메타데이터 가속 활성화	메타데이터 가속 미활성화
버킷 생성/조회/삭제 작업	지원	지원
객체 업로드/다운로드/삭제 작업	지원	지원
버킷 권한	지원	지원
객체 권한	버킷 권한에서 상속	지원
리전 내 재해 복구	지원	지원
버킷 암호화	미적용	지원

©2013-2022 Tencent Cloud. All rights reserved.

지표 항목	메타데이터 가속 활성화	메타데이터 가속 미활성화
객체 암호화	미적용	지원
CORS	미적용	지원
링크 도용 방지	미적용	지원
버전 관리	미적용	지원
교차 버킷 복사	미적용	지원
정적 웹 사이트	미적용	지원
Origin-pull 구성	미적용	지원
라이프사이클	지원	지원
리스트	미적용	지원
버킷 태그	미적용	지원
COS Select	미적용	지원
COS Batch	미적용	지원
CI 기능	미적용	지원

과금 안내

현재 메타데이터 가속 기능은 베타 테스트 중으로, 상용화 요금은 과금되지 않습니다. 추후 과금 시 내부 메시지, 이메일, SMS 등으로 안내할 예정입니다. 내부 메시지 또는 과금 개요를 통해 최신 동향을 알아보실 수 있습니다.

©2013-2022 Tencent Cloud. All rights reserved.

HDFS 데이터를 메타데이터 가속 기능이 활성화된 버킷으로 마이그레이션하기

최종 업데이트 날짜: : 2023-03-20 15:06:18

소개

COS(Cloud Object Storage)는 메타데이터 가속화 기능을 제공하여 고성능 파일 시스템 기능을 제공합니다. 메타데이터 가속화는 기본 계층에서 Cloud HDFS(CHDFS)의 강력한 메타데이터 관리 기능을 활용하여 COS 액세스에 파일 시스템 시맨틱을 사용할 수 있도록 합니다. 설계된 시스템 메트릭은 최대 100GB/s의 대역폭, 10만개 이상의 QPS(초당 쿼리) 및 ms의 대기 시간에 도달할 수 있습니다. 메타데이터 가속이 활성화된 버킷은 빅 데이터, 고성능 컴퓨팅, 기계 학습 및 AI와 같은 시나리오에서 널리 사용될 수 있습니다. 메타데이터 가속에 대한 자세한 내용은 메타데이터 가속 개요를 참고하십시오.

COS는 메타데이터 가속 서비스를 통해 Hadoop 시맨틱을 제공합니다. 따라서 Hadoop Distcp 툴을 사용하여 COS와 다른 Hadoop 파일 시스템 간의 양방향 데이터 마이그레이션을 쉽게 구현할 수 있습니다. 이 문서에서는 Hadoop Distcp를 사용하여 로컬 HDFS의 파일을 COS의 메타데이터 가속 버킷으로 마이그레이션하는 방법을 설명합니다.

마이그레이션 환경 준비

마이그레이션 툴

1. 아래 나열된 툴의 jar 패키지를 다운로드하고 /data01/jars와 같이 클러스터에서 마이그레이션 작업을 실행하는 노드의 로컬 디렉터리에 배치합니다.

EMR 환경

설치 설명

jar 파일 이름	설명	다운로드 주소
cos-distcp-1.12-3.1.0.jar	COSN에 데이터를 복사할 COSDistCp 패키지	COSDistCp 툴을 참고하십시오
chdfs_hadoop_plugin_network-2.8.jar	OFS 플러그인	다운로드

자체 구축 Hadoop/CDH 등 환경

©2013-2022 Tencent Cloud. All rights reserved.

소프트웨어 종속성

Hadoop 2.6.0 이상 및 Hadoop-COS 8.1.5 이상이 필요합니다. cos_api-bundle 플러그인 버전은 COSN github releases에 설명된 대로 Hadoop-COS 버전과 일치해야 합니다.

설치 설명

Hadoop 환경에 다음 플러그인을 설치합니다.

jar 파일 이름	설명	다운로드 주소
cos-distcp-1.12-3.1.0.jar	COSN에 데이터를 복사할 COSDistCp 패키지	COSDistCp 툴을 참고하십시오
chdfs_hadoop_plugin_network-2.8.jar	OFS 플러그인	다운로드
Hadoop-COS	Version >= 8.1.5	Hadoop-COS 툴을 참고하십시오
cos_api-bundle	버전은 Hadoop-COS 버전과 일치해야 합니다	다운로드

주의 :

- Hadoop-cos는 버전 8.1.5부터 `cosn://bucketname-appid/` 형식의 메타데이터 가속 버킷에 대한 액세스 지원;
- 메타데이터 가속화 기능은 버킷 생성 중에만 활성화할 수 있으며 일단 활성화되면 비활성화할 수 없습니다. 따라서 비즈니스 조건에 따라 활성화할지 여부를 신중하게 고려하십시오. 또한 레거시 Hadoop-COS 패키지는 메타데이터 가속 버킷에 액세스할 수 없습니다.

2. HDFS 프로토콜을 사용하여 메타데이터 가속이 활성화된 버킷에 액세스의 '버킷 생성 및 HDFS 프로토콜 구성'의 지침에 따라 메타데이터 가속 버킷을 생성하고 이에 대한 HDFS 프로토콜을 구성합니다.
3. 마이그레이션 클러스터의 `core-site.xml` 을 수정하고 구성을 모든 노드에 배포합니다. 데이터만 마이그레이션해야 하는 경우 빅 데이터 컴포넌트를 다시 시작할 필요가 없습니다.

key	value	구성 파일
fs.cosn.trsf.fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	core-site.xn

©2013-2022 Tencent Cloud. All rights reserved.

key	value	구성 파일
fs.cosn.trsf.fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	core-site.xn
fs.cosn.trsf.fs.ofs.tmp.cache.dir	/data/emr/hdfs/tmp/ 형식	core-site.xn
fs.cosn.trsf.fs.ofs.user.appid	COS bucket의 appid	core-site.xn
fs.cosn.trsf.fs.ofs.ranger.enable.flag	false	core-site.xn
fs.cosn.trsf.fs.ofs.bucket.region	bucket region	core-site.xn

4. HDFS 프로토콜을 사용하여 메타데이터 가속이 활성화된 버킷에 액세스의 'COS에 액세스하도록 컴퓨팅 클러스터 구성'에 설명된 대로 사설망을 통해 메타데이터 가속 버킷에 액세스하여 마이그레이션을 확인할 수 있습니다. 마이그레이션 클러스터 제출자를 사용하여 COS에 성공적으로 액세스할 수 있는지 확인하십시오.

기존 데이터 마이그레이션

©2013-2022 Tencent Cloud. All rights reserved.

1. 마이그레이션할 디렉터리 결정

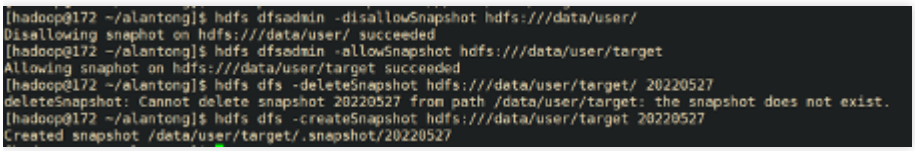
일반적으로 HDFS 스토리지 데이터가 먼저 마이그레이션됩니다. 원본 HDFS 클러스터에서 마이그레이션할 디렉터리가 선택되며 대상 경로는 원본 경로와 동일해야 합니다.

HDFS 디렉터리 `hdfs:///data/user/target` 을 `cosn://{bucketname-appid}/data/user/target` 으로 마이그레이션해야 한다고 가정합니다.

원본 디렉터리의 파일이 마이그레이션 중에 변경되지 않도록 하기 위해 HDFS의 스냅샷 기능을 사용하여 원본 디렉터리의 스냅샷(현재 날짜로 명명됨)을 생성합니다.

```
hdfs dfsadmin -disallowSnapshot hdfs:///data/user/
hdfs dfsadmin -allowSnapshot hdfs:///data/user/target
hdfs dfs -deleteSnapshot hdfs:///data/user/target {현재 날짜}
hdfs dfs -createSnapshot hdfs:///data/user/target {현재 날짜}
```

성공 예시 참고:



스냅샷을 생성하지 않으려면 원본 디렉터리에서 target 파일을 직접 마이그레이션할 수 있습니다.

2. 마이그레이션에 COSDistCp 사용

COSDistCp 작업을 시작하여 원본 HDFS에서 대상 COS 버킷으로 파일을 복사합니다.

COSDistCp 작업은 본질적으로 MapReduce 작업입니다. 인쇄된 MapReduce 작업 로그에는 MR 작업이 성공적으로 실행되었는지 여부가 표시됩니다. 작업이 실패하면 YARN 페이지를 보고 문제 해결을 위해 로그 또는 예외 정보를 COS 팀에 제출할 수 있습니다. COSDistCp를 사용하여 다음 단계에서 마이그레이션 작업을 실행할 수 있습니다.
(1) 임시 디렉터리 생성
(2) COSDistCp 작업 실행
(3) 실패한 파일을 다시 마이그레이션

(1) 임시 디렉터리 생성

```
hadoop fs -libjars /data01/jars/chdfs_hadoop_plugin_network-2.8.jar -mkdir cos
n://bucket-appid/distcp-tmp
```

(2) COSDistCp 작업 실행

©2013-2022 Tencent Cloud. All rights reserved.

```
nohup hadoop jar /data01/jars/cos-distcp-1.10-2.8.5.jar -libjars /data01/jars/chd
fs_hadoop_plugin_network-2.8.jar --src=hdfs:///data/user/target/.snapshot/{현재 날
짜} --dest=cosn://{bucket-appid}/data/user/target --temp=cosn://bucket-appid/dist
cp-tmp/ --preserveStatus=ugpt --skipMode=length-checksum --checkMode=length-check
sum --cosChecksumType=CRC32C --taskNumber 6 --workerNumber 32 --bandWidth 200 >>
./distcp.log &
```

매개변수는 아래에 자세히 설명되어 있습니다. 필요에 따라 해당 값을 조정할 수 있습니다.

- --taskNumber=VALUE: 복사 스레드 수. 예: --taskNumber=10.
- --workerNumber=VALUE: 복사 스레드 수. COSDistCp는 이 값 세트를 기반으로 각 복사 프로세스에 대해 복사 스레드 풀을 생성합니다. 예: --workerNumber=4.
- --bandWidth: 마이그레이션된 각 파일 읽기 최대 대역폭(MB/s). 기본값: -1, 읽기 대역폭에 제한이 없음을 나타냅니다. 예:--bandWidth=10.
- --cosChecksumType=CRC32C: 기본적으로 CRC32C가 사용되지만 HDFS 클러스터는 COMPOSITE_CRC32를 확인할 수 있어야 합니다. Hadoop 버전은 3.1.1+이어야 합니다. 그렇지 않으면 이 매개변수를 --cosChecksumType=CRC64로 변경해야 합니다.

> 주의 :
> COSDistCp 마이그레이션의 총 대역폭 제한을 계산하는 공식은 taskNumber x workerNumber x bandWidth입니다. workerNumber를 1로 설정하고 taskNumber 매개변수를 사용하여 동시 마이그레이션 수를 제어하고 bandWidth 매개변수를 사용하여 단일 동시 마이그레이션의 대역폭을 제어할 수 있습니다.

복사 작업이 끝나면 작업 로그는 복사본에 대한 통계를 출력합니다. 카운터는 다음과 같습니다.
여기서 FILES_FAILED는 실패한 파일의 수를 나타냅니다. FILES_FAILED 카운터가 없으면 모든 파일이 성공적으로 마이그레이션된 것입니다.

```
CosDistCp Counters
BYTES_EXPECTED=10198247
BYTES_SKIPPED=10196880
FILES_COPIED=1
FILES_EXPECTED=7
FILES_FAILED=1
FILES_SKIPPED=5
```

출력 결과의 구체적인 통계 항목은 아래와 같습니다.

통계 항목	설명

©2013-2022 Tencent Cloud. All rights reserved.

통계 항목	설명
BYTES_EXPECTED	원본 디렉터리 통계에 따라 복사할 파일의 총 크기, 단위: 바이트
FILES_EXPECTED	원본 디렉터리 통계에 따라 복사할 파일 수. 디렉터리 파일 포함
BYTES_SKIPPED	스킵할 수 있는 파일의 총 크기(바이트)(동일한 길이 또는 체크섬 값)
FILES_SKIPPED	스킵할 수 있는 원본 파일 수(동일한 길이 또는 체크섬 값)
FILES_COPIED	복사 성공한 원본 파일 수
FILES_FAILED	복사 실패한 원본 파일 수
FOLDERS_COPIED	복사 완료한 디렉터리 수
FOLDERS_SKIPPED	스킵한 디렉터리 수

(3) 실패한 파일을 다시 마이그레이션

COSDistCp는 비효율적인 파일 마이그레이션의 대부분의 문제를 해결할 뿐만 아니라 `--delete` 매개변수를 사용하여 HDFS와 COS 데이터 간의 완전한 일관성을 보장할 수 있습니다.

`--delete` 매개변수를 사용할 때 `--deleteOutput=/xxx(사용자 지정)` 매개변수를 추가해야 하지만 `--diffMode` 매개변수는 추가하지 않아야 합니다.

```
nohup hadoop jar /data01/jars/cos-distcp-1.10-2.8.5.jar -libjars /data01/jars/chd
fs_hadoop_plugin_network-2.8.jar --src=--src=hdfs:///data/user/target/.snapshot/
{현재 날짜} --dest=cosn://{bucket-appid}/data/user/target --temp=cosn://bucket-app
id/distcp-tmp/ --preserveStatus=ugpt --skipMode=length-checksum --checkMode=lengt
h-checksum --cosChecksumType=CRC32C --taskNumber 6 --workerNumber 32 --bandWidth
200 --delete --deleteOutput=/dele-xx >> ./distcp.log &
```

실행 후 HDFS와 COS 사이의 다른 데이터는 `trash` 디렉터리로 이동되고 이동된 파일 목록은 `/xxx/failed` 디렉터리에 생성됩니다. `hadoop fs -rm URL` 또는 `hadoop fs -rmr URL` 을 실행하여 `trash` 디렉터리의 데이터를 삭제할 수 있습니다.

증분 마이그레이션

나중에 증분 데이터를 마이그레이션해야 하는 경우 모든 데이터가 마이그레이션될 때까지 전체 마이그레이션 단계를 반복하기만 하면 됩니다.

©2013-2022 Tencent Cloud. All rights reserved.

HDFS 프로토콜을 사용하여 메타데이터 가속이 활성화된 버킷에 액세스

최종 업데이트 날짜: : 2023-03-20 15:06:18

메타데이터 가속 개요

메타데이터 가속 기능은 Tencent Cloud COS(Cloud Object Storage) 서비스에서 제공하는 고성능 파일 시스템 기능입니다. 메타데이터 가속 기능의 기본 레이어는 클라우드 HDFS의 우수한 메타데이터 관리 기능을 사용하여, 사용자가 파일 시스템의 시맨틱을 통해 객체 스토리지 서비스에 액세스할 수 있도록 지원합니다. 시스템 설계 지표는 100GB/s 대역폭, 10만 규모 QPS, ms 단위 딜레이에 달합니다. 메타데이터 가속 기능이 활성화되면 버킷은 빅 데이터, 고성능 컴퓨팅, 머신러닝, AI와 같은 시나리오에서 널리 사용될 수 있습니다. 메타데이터 가속 기능에 대한 자세한 내용은 메타데이터 가속 기능 개요를 참고하십시오.

HDFS 액세스의 장점

과거에는 COS 기반의 빅데이터 액세스가 주로 Hadoop-COS 툴을 통해 구현되었습니다. Hadoop-COS 도구는 내부적으로 HCFS API를 COS Restful API에 적용하여 COS의 데이터에 액세스합니다. COS와 파일 시스템 간의 메타데이터 구성의 차이로 인해 메타데이터 운영 성능이 달라지며, 이는 빅데이터 분석 성능에 영향을 미칩니다. 메타데이터 가속 기능이 활성화된 Bucket은 HCFS 프로토콜과 완벽하게 호환되며, 기본 HDFS API를 사용하여 직접 액세스할 수 있으므로 HDFS 프로토콜을 COS 프로토콜로 변환하는 오버헤드를 줄이고 효율적인 디렉터리 Rename(원자 작업), 파일 Atime, Mtime 업데이트, 효율적인 디렉터리 DU 통계, Posix ACL 권한 지원과 같은 기본 HDFS 기능을 제공합니다.

버킷 생성 및 HDFS 프로토콜 구성

1. COS 버킷 생성하고 이에 대한 메타데이터 가속을 활성화합니다.
   버킷이 생성되면 버킷의 **파일 목록** 페이지로 이동하여 파일을 업로드 및 다운로드할 수 있습니다.
2. 왼쪽 사이드바에서 **성능 구성 > 메타데이터 가속**을 클릭하면 메타데이터 가속이 활성화된 것을 확인할 수 있습니다.

메타데이터 가속을 활성화해야 하는 버킷을 생성하는 경우 메시지에 따라 해당 **권한 부여** 작업을 수행해야 합니다. 인증을 클릭하면 HDFS 프로토콜이 자동으로 활성화되고 기본 버킷 마운트 대상 정보를 볼 수 있습니다.

©2013-2022 Tencent Cloud. All rights reserved.

설명：
시스템에서 해당 HDFS 파일 시스템을 찾을 수 없다고 표시하면 티켓 제출 하시기 바랍니다.

3. HDFS 권한 구성 열에서 **권한 구성 추가**를 클릭합니다.

4. VPC 이름 열에서 컴퓨팅 클러스터가 있는 VPC를 선택하고 노드 IP 열에서 열어야 하는 IP 주소 또는 범위를 입력하고 액세스 유형으로 편집 가능 또는 읽기 전용을 선택하고 **저장**을 클릭합니다.

설명：
HDFS 권한 구성은 기본 COS 권한 시스템과 다릅니다. HDFS를 사용하여 COS 버킷에 액세스하는 경우 기본 HDFS와 동일한 권한 환경을 얻기 위해 지정된 VPC의 머신이 COS 버킷에 액세스할 수 있는 권한을 부여하도록 HDFS 권한을 구성하는 것이 좋습니다.

COS에 액세스하도록 컴퓨팅 클러스터 구성

환경 종속

종속성	chdfs-hadoop-plugin	COSN（hadoop-cos）	cos_api-bundle
버전	≥ v2.7	≥ v8.1.5	버전이 tencentyun/hadoop-cos에 나열된 COSN 버전과 일치하는지 확인하십시오.
다운로드 주소	Github	Github	Github

EMR 환경

EMR 환경은 이미 COS와 원활하게 통합되었으며 다음 단계만 완료하면 됩니다.

1. EMR 서버를 찾아 다음 명령을 실행하여 EMR 환경에 필요한 서비스 폴더의 패키지 버전이 환경 종속성에 대한 요구 사항을 충족하는지 확인합니다.

```
find / -name "chdfs*"
find / -name "temrfs_hadoop*"
```

©2013-2022 Tencent Cloud. All rights reserved.

```
[root@172 ~]# find /usr/local/service/ -name 'chdfs*'
/usr/local/service/cosranger/sbin/chdfs-ranger-service-define.sh
/usr/local/service/cosranger/sbin/chdfs-ranger.json
/usr/local/service/spark/jars/chdfs_hadoop_plugin_network-2.8.jar
/usr/local/service/hive/spark/jars/chdfs_hadoop_plugin_network-2.8.jar
/usr/local/service/hadoop/share/hadoop/common/lib/chdfs_hadoop_plugin_network-2.8.jar
/usr/local/service/apps/kylin-4.0.1/spark/jars/chdfs_hadoop_plugin_network-2.8.jar
/usr/local/service/tez/chdfs_hadoop_plugin_network-2.8.jar
/usr/local/service/ranger/ews/webapp/WEB-INF/classes/ranger-plugins/chdfs
/usr/local/service/ranger/ews/webapp/WEB-INF/classes/ranger-plugins/chdfs/chdfs-ranger-plugin-1.0-shaded.jar
/usr/local/service/trino/plugin/hive/chdfs_hadoop_plugin_network-2.8.jar
```

```
[root@172 ~]# find /usr/local/service/ -name 'temrfs_hadoop*'
/usr/local/service/spark/jars/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/hive/spark/jars/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/hadoop/share/hadoop/common/lib/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/apps/oozie-5.2.1/embedded-oozie-server/webapp/WEB-INF/lib/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/apps/kylin-4.0.1/spark/jars/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/tez/temrfs_hadoop_plugin_network-1.0.jar
/usr/local/service/trino/plugin/hive/temrfs_hadoop_plugin_network-1.0.jar
```

검색 결과에 있는 두 jar 패키지의 버전이 위의 환경 종속성 요구 사항을 충족하는지 확인합니다.

2. chdfs-hadoop-plugin을 업데이트해야 하는 경우 다음 단계를 진행합니다.

다음 위치에서 업데이트된 jar 패키지의 스크립트 파일을 다운로드합니다.

- update_cos_jar.sh
- update_cos_jar_common.sh

다음 명령을 실행하여 서버의 /root 디렉터리에 두 개의 스크립트를 넣어 update_cos_jar.sh에 대한 실행 권한을 추가합니다.

```
sh update_cos_jar.sh https://hadoop-jar-beijing-1259378398.cos.ap-beijing.myqcloud.com/hadoop_plugin_network/2.7
```

매개변수를 해당 리전의 버킷으로 교체합니다(예시: 광저우 리전의 `https://hadoop-jar-guangzhou-1259378398.cos.ap-guangzhou.myqcloud.com/hadoop_plugin_network/2.7`).
모든 jar 패키지가 교체될 때까지 각 EMR 노드에서 위의 단계를 수행합니다.

3. hadoop-cos 패키지 또는 cos_api-bundle 패키지를 업데이트해야 하는 경우 다음 단계를 진행합니다.

- /usr/local/service/hadoop/share/hadoop/common/lib/hadoop-temrfs-1.0.5.jar을 temrfs_hadoop_plugin_network-1.1.jar로 바꿉니다.
- core-site.xml에 다음 구성 항목을 추가합니다.
  - emr.temrfs.download.md5=822c4378e0366a5cc26c23c88b604b11

©2013-2022 Tencent Cloud. All rights reserved.

- emr.temrfs.download.version=2.7.5-8.1.5-1.0.6 (2.7.5를 hadoop 버전으로 바꾸고 8.1.5를 필요한 hadoop-cos 패키지 버전(8.1.5 이상이어야 함)으로 바꿉니다. cos_api-bundle 버전은 자동으로 조정됩니다)
- emr.temrfs.download.region=sh
- emr.temrfs.tmp.cache.dir=/data/emr/hdfs/tmp/temrfs

- core-site.xml에서 구성 항목 fs.cosn.impl=com.qcloud.emr.fs.TemrfsHadoopFileSystemAdapter를 수정합니다.

4. 새 구성 항목 `fs.cosn.bucket.region` 및 `fs.cosn.trsf.fs.ofs.bucket.region` 을 추가하여 버킷이 상주하는 COS 리전(예시: `ap-shanghai`)을 지정하여 EMR 콘솔에서 core-site.xml을 구성합니다.

> 주의 :
> 버킷이 상주하는 COS 리전(예시: `ap-shanghai`)을 지정하려면 `fs.cosn.bucket.region` 및 `fs.cosn.trsf.fs.ofs.bucket.region` 이 필요합니다.

5. Yarn, Hive, Presto 및 Impala와 같은 상주 서비스를 다시 시작합니다.

자체 구축된 Hadoop/CDH 환경

1. CDH 설치에 설명된 자체 구축 환경에서는 환경 종속성에서 버전 요구 사항을 충족하는 3개의 jar 패키지를 다운로드해야 합니다.

2. Hadoop 클러스터에 있는 각 서버의 'classpath' 경로에 위의 세 가지 설치 패키지를 배치합니다. 예시: `/usr/local/service/hadoop/share/hadoop/common/lib/` (컴포넌트에 따라 다를 수 있음)

3. hadoop-env.sh 파일을 수정합니다. `$HADOOP_HOME/etc/hadoop` 디렉터리에 들어가 hadoop-env.sh 파일을 편집하고, 다음 내용을 추가하여 cosn 관련 jar 패키지를 Hadoop 환경 변수에 추가합니다.

```
for f in $HADOOP_HOME/share/hadoop/tools/lib/*.jar; do
if [ "$HADOOP_CLASSPATH" ]; then
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$f
else
export HADOOP_CLASSPATH=$f
fi
done
```

4. 컴퓨팅 클러스터의 `core-site.xml` 에 다음 구성 항목을 추가합니다.

©2013-2022 Tencent Cloud. All rights reserved.

```
<!--cosn의 구현 클래스-->
<property>
<name>fs.cosn.impl</name>
<value>org.apache.hadoop.fs.CosFileSystem</value>
</property>

<!--ap-guangzhou 형식의 사용자 버킷 리전 정보-->
<property>
<name>fs.cosn.bucket.region</name>
<value>ap-guangzhou</value>
</property>

<!--ap-guangzhou 형식의 사용자 버킷 리전 정보-->
<property>
<name>fs.cosn.trsf.fs.ofs.bucket.region</name>
<value>ap-guangzhou</value>
</property>

<!-- SecretId 및 SecretKey를 얻는 방법 구성-->
<property>
<name>fs.cosn.credentials.provider</name>
<value>org.apache.hadoop.fs.auth.SimpleCredentialProvider</value>
</property>

<!--계정에 대한 API Keys 정보입니다. [액세스 관리 콘솔](https://console.tencentclou
d.com/capi)에 로그인하면 Tencent Cloud API 키를 조회할 수 있습니다. -->
<property>
<name>fs.cosn.userinfo.secretId</name>
<value>XXXXXXXXXXXXXXXXXXXXXXXXX</value>
</property>

<!--계정에 대한 API Keys 정보입니다. [액세스 관리 콘솔](https://console.tencentclou
d.com/capi)에 로그인하면 Tencent Cloud API 키를 조회할 수 있습니다. -->
<property>
<name>fs.cosn.userinfo.secretKey</name>
<value>XXXXXXXXXXXXXXXXXXXXXXXXX</value>
</property>

<!-- 계정의 appid 구성-->
<property>
<name>fs.cosn.trsf.fs.ofs.user.appid</name>
<value>125XXXXXX</value>
</property>

<!--실행 중 생성된 임시 파일을 저장하는 데 사용되는 로컬 임시 디렉터리-->
<property>
```

©2013-2022 Tencent Cloud. All rights reserved.

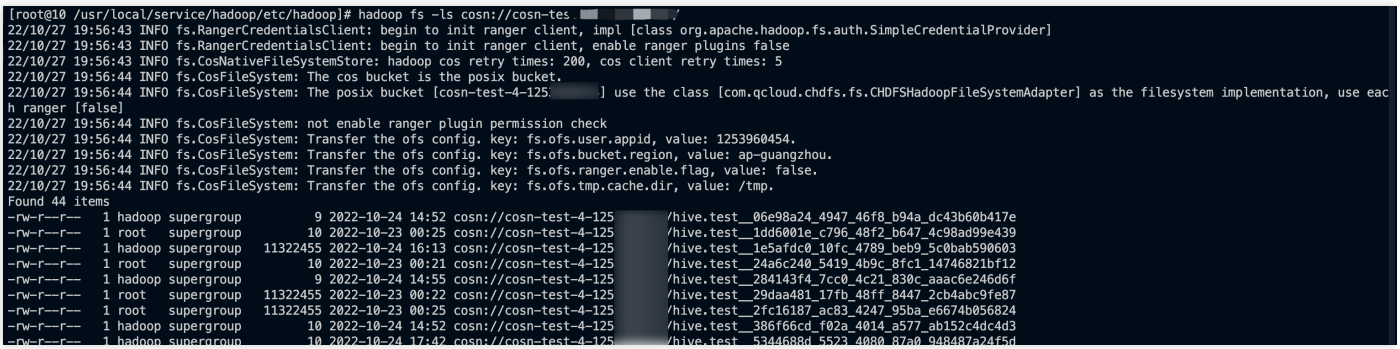
```
<name>fs.cosn.trsf.fs.ofs.tmp.cache.dir</name>
<value>/tmp</value>
</property>
```

5. Yarn, Hive, Presto 및 Impala와 같은 상주 서비스를 다시 시작합니다.

환경 확인

모든 환경 구성 단계가 완료되면 다음과 같은 방법으로 환경을 확인할 수 있습니다.

- 클라이언트에서 Hadoop 명령줄을 사용하여 마운트가 성공했는지 확인합니다.



- COS 콘솔에 로그인하여 버킷 파일 목록의 파일과 디렉터리가 일치하는지 확인합니다.

Ranger 권한 구성

기본적으로 기본 POSIX ACL 모드는 HDFS 프로토콜에 대한 인증에 채택됩니다. Ranger 인증을 사용해야 하는 경우 다음과 같이 구성합니다.

EMR 환경

1. EMR 환경에 COSRanger 서비스가 통합되어 있어 EMR 클러스터 구매 시 선택하실 수 있습니다.
2. HDFS 프로토콜의 **HDFS 인증 모드**에서 **Ranger 인증 모드**를 선택하고 해당 Ranger 주소 정보를 설정합니다.

- core-site.xml에 새 구성 항목 fs.cosn.credentials.provider를 추가하고 org.apache.hadoop.fs.auth.RangerCredentialsProvider로 설정합니다.
- Ranger에 대해 궁금한 사항은 COS Ranger 권한 시스템 솔루션을 참고하십시오.

자체 구축된 Hadoop/CDH 환경

1. HDFS 프로토콜을 통해 COS에 액세스하도록 Ranger 서비스를 구성합니다. 자세한 내용은 COS Ranger 권한 시스템 솔루션을 참고하십시오.

©2013-2022 Tencent Cloud. All rights reserved.

2. HDFS 프로토콜의 **HDFS 인증 모드**에서 **Ranger 인증 모드**를 선택하고 해당 Ranger 주소 정보를 설정합니다.

- core-site.xml에 새 구성 항목 fs.cosn.credentials.provider를 추가하고 org.apache.hadoop.fs.auth.RangerCredentialsProvider로 설정합니다.

기타

빅 데이터 시나리오에서는 다음 단계에서 HDFS 프로토콜을 통해 활성화된 메타데이터 가속을 사용하여 버킷에 액세스할 수 있습니다.

1. 버킷 생성 및 HDFS 프로토콜 구성의 설명에 따라 `core-stie.xml` 에 HDFS 마운트 대상 정보를 설정합니다.
2. Hive, MR 및 Spark와 같은 컴포넌트가 있는 버킷에 액세스합니다. 자세한 내용은 컴퓨팅 클러스터에 COS 버킷 마운트를 참고하십시오.
3. 기본적으로 기본 `POSIX ACL` 모드가 인증에 채택됩니다. `Ranger 인증` 을 사용해야 하는 경우 COS Ranger 권한 시스템 솔루션을 참고하십시오.

©2013-2022 Tencent Cloud. All rights reserved.

컴퓨팅 클러스터에 COS 버킷 마운트하기

최종 업데이트 날짜: : 2023-03-14 16:54:52

소개

COS(Cloud Object Storage)에서는 메타데이터 가속을 활성화하여 HDFS 프로토콜 액세스 기능을 얻을 수 있습니다. 메타데이터 가속이 활성화되면 COS는 버킷에 대한 마운트 포인트를 생성합니다. 그런 다음 HDFS 클라이언트를 다운로드하고 클라이언트의 마운트 포인트를 사용하여 COS를 마운트할 수 있습니다. 본 문서에서는 컴퓨팅 클러스터에 메타데이터 가속이 활성화된 버킷을 마운트하는 방법을 소개합니다.

주의 :

- Hadoop-cos는 버전 8.1.5부터 `cosn://bucketname-appid/` 메소드에서 메타데이터 가속 버킷에 대한 액세스를 지원합니다.
- 메타데이터 가속 기능은, 메모리 버킷 생성 시에만 활성화할 수 있으며 활성화 후에는 비활성화할 수 없습니다. 비즈니스 상황에 따라 활성화 여부를 **신중하게 고려**하십시오. 또한, 이전 버전의 Hadoop-cos 패키지는 일반적으로 메타데이터 가속이 활성화된 버킷에는 액세스할 수 없습니다.

전제 조건

- Java 1.8은 마운트할 컴퓨팅 클러스터의 머신 또는 컨테이너에 설치됩니다.
- 마운트할 컴퓨팅 클러스터의 머신 또는 컨테이너에 액세스 권한이 부여됩니다. HDFS 권한 구성에서 액세스할 수 있는 VPC 및 IP를 지정해야 합니다.
- 종속 JAR 패키지 설명:

1. chdfs_hadoop_plugin_network-2.8.jar verison ≥ 2.7.
2. cos_api-bundle.jar version ≥ 5.6.69.
3. Hadoop-cos version ≥ 8.1.5.
4. ofs-java-sdk.jar(version ≥ 1.0.4)는 설치 없이 자동으로 풀링하며, hadoop fs ls를 성공적으로 실행한 후 해당 버전이 fs.cosn.trsf.fs.ofs.tmp.cache.dir에 의해 구성된 디렉터리에서 예상과 부합하는지 확인합니다.

작업 단계

1. Hadoop 클라이언트 설치 패키지를 다운로드합니다.

©2013-2022 Tencent Cloud. All rights reserved.

2. POSIX Hadoop 클라이언트 설치 패키지를 다운로드합니다.
3. cos java sdk 설치 패키지를 다운로드합니다.
4. 설치 패키지를 각 노드의 classpath 아래에 두어 작업 시작이 정상적으로 로딩될 수 있도록 합니다(예: `$HADOOP_HOME/share/hadoop/common/lib/' 아래).

> 주의 :
> EMR 환경은 종속 jar 패키지와 함께 제공되므로 설치할 필요가 없습니다. POSIX semantics를 통해 메타데이터 가속 버킷에 직접 액세스할 수 있습니다. s3 프로토콜을 사용하여 액세스해야 하는 경우, fs.cosn.posix_bucket.fs.impl 구성 항목을 변경하고 자세한 내용은 아래를 참고하십시오.

5. `core-site.xml` 파일을 편집하여 다음 기본 설정을 추가합니다.

> 주의 :
> - 비즈니스 보안 향상을 위해 영구 키 대신 서브 계정 키 또는 임시 키를 사용하는 것이 좋습니다. 서브 계정 승인 시 최소 권한의 원칙 설명을 준수하여 예상치 못한 데이터 유출을 방지하시기 바랍니다.
> - 영구 키를 사용해야 하는 경우 최소 권한의 원칙 설명에 따라 권한 범위를 제한하는 것이 좋습니다. 사용 보안을 향상시키기 위해 허용된 작업, 리소스 범위 및 액세스 IP와 같은 조건을 영구 키에 포함시키는 것이 좋습니다.

```
<!--계정에 대한 API Keys 정보입니다. [액세스 관리 콘솔](https://console.tencentcloud.com/capi)에 로그인하면 Tencent Cloud API 키를 조회할 수 있습니다. -->
<!--구성 보안을 강화하기 위해 구성에 서브 계정 키 또는 임시 키를 사용하는 것이 좋습니다. 서브 계정 승인 시 [최소 권한의 원칙 설명](https://www.tencentcloud.com/document/product/436/32972)을 따릅니다. -->
<property>
<name>fs.cosn.userinfo.secretId/secretKey</name>
<value>AKIDxxxxxxxxxxxxxxxxxxxxx</value>
</property>

<!--cosn의 구현 클래스-->
<property>
<name>fs.AbstractFileSystem.cosn.impl</name>
<value>org.apache.hadoop.fs.CosN</value>
</property>

<!--cosn의 구현 클래스-->
<property>
<name>fs.cosn.impl</name>
```

©2013-2022 Tencent Cloud. All rights reserved.

```
<value>org.apache.hadoop.fs.CosFileSystem</value>
</property>

<!--ap-guangzhou 형식의 버킷 리전-->
<property>
<name>fs.cosn.bucket.region</name>
<value>ap-guangzhou</value>
</property>

<!--실행 중 생성된 임시 파일을 저장하는 데 사용되는 로컬 임시 디렉터리->
<property>
<name>fs.cosn.tmp.dir</name>
<value>/tmp/hadoop_cos</value>
</property>
```

6. `core-site.xml` 을 모든 `hadoop` 노드에 동기화 합니다.

> 설명 :
> EMR 클러스터의 경우, 상기 3, 4단계를 통해 EMR 콘솔 컴포넌트 관리에서 HDFS 설정을 수정할 수 있습니다.

7. `hadoop fs` 명령줄 도구를 사용하여 `hadoop fs -ls cosn://${bucketname-appid}/` 명령을 실행합니다. 여기서 `bucketname-appid` 는 마운트 주소, 즉 버킷 이름입니다. 파일 목록이 제대로 출력되면 COS 버킷이 성공적으로 마운트된 것입니다.
8. `hadoop` 또는 `mr` 작업의 다른 구성 항목을 사용하여 메타데이터 가속이 활성화된 버킷에서 데이터 작업을 실행할 수도 있습니다. `mr` 작업의 경우 `-Dfs.defaultFS=ofs://${bucketname-appid}/` 를 통해 작업의 기본 입력 및 출력 파일 'FS'를 해당 버킷으로 변경할 수 있습니다.

설정 항목 설명

> 설명 :
> 여기에서는 POSIX semantics 액세스 및 S3 프로토콜 액세스를 통해 메타데이터 가속 버킷에 액세스할 수 있으며 더 나은 성능을 얻기 위해 POSIX semantics 액세스를 사용하는 것이 좋습니다.

1. 일반 필수 구성 항목

©2013-2022 Tencent Cloud. All rights reserved.

주의：
메타데이터 가속 버킷에 어떤 방식으로 액세스하든 다음과 같은 일반 구성 항목을 설정해야 합니다.

구성 항목	구성 항목 콘텐츠	설명
fs.cosn.userinfo.secretId/secretKey	형식은 AKIDxxxxxxxxxxxxxxxxxxxx와 같습니다	사용자 계정의 API 키 정보를 입력 니다. CAM 콘솔에 로그인하여 Tencent Cloud API 키를 확인할 수 습니다.
fs.cosn.impl	org.apache.hadoop.fs.CosFileSystem	FileSystem에 대한 cosn 구현 유형 org.apache.hadoop.fs.CosFileSys 으로 고정됩니다.
fs.AbstractFileSystem.cosn.impl	org.apache.hadoop.fs.CosN	AbstractFileSystem에 대한 cosn 현 유형, org.apache.hadoop.fs.Co 으로 고정됩니다.
fs.cosn.bucket.region	형식은 ap-beijing과 같습니다.	액세스할 버킷의 리전 정보를 입력 니다. 열거 값은 리전 및 액세스 도 인의 도메인 약칭을 참고하십시오 예: ap-beijing, ap-guangzhou 등. 존 설정 호환: fs.cosn.userinfo.reg
fs.cosn.tmp.dir	기본값/tmp/hadoop_cos	실제 존재하는 로컬 디렉터리를 입 합니다. 실행 과정에서 생성되는 파일이 잠시 해당 디렉터리에 저장 니다. 동시에 각 노드의 디렉터리에 대해 충분한 공간과 권한을 구성하 것이 좋습니다.

2. POSIX 액세스 모드 필수 구성 항목(권장 방식)

설명：

- POSIX 액세스 모드에서는 일반 구성 항목 외에 다음과 같은 구성 내용을 추가해야 합니다. POSIX 액세스 모드의 기타 설정 항목에 "fs.cosn.trsf." 접두사를 추가하면 메타데이터 가속 버킷에 액세스할 수 있습니다.
- 기존 Hadoop cos와 관련된 구성 항목은 더 이상 적용되지 않습니다.

구성 항목	구성 항목 콘텐츠	설명

©2013-2022 Tencent Cloud. All rights reserved.

구성 항목	구성 항목 콘텐츠	설명
fs.cosn.trsf.fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	메타데이
fs.cosn.trsf.fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	메타데이
fs.cosn.trsf.fs.ofs.tmp.cache.dir	형식은 /data/emr/hdfs/tmp/posix-cosn/과 같습니다.	실제 로컬 중 생성된 저장됩니 "/data cosn/" 렉터리에 구성하는
fs.cosn.trsf.fs.ofs.user.appid	형식은 12500000000과 같습니다.	필수. 사용
fs.cosn.trsf.fs.ofs.bucket.region	형식은 ap-beijing과 같습니다.	필수. 사용

3. S3 프로토콜 액세스 모드 필수 구성 항목

S3 프로토콜 액세스 모드는 다음과 같은 설정이 필요하며, 기타 옵션은 Hadoop-cos 설정 항목을 참고하십시오.

구성 항목	구성 항목 콘텐츠	설명
fs.cosn.posix_bucket.fs.impl	org.apache.hadoop.fs.CosNFileSystem	이 매개변수는 POSIX 액세스 모드(기본 com.qcloud.chdfs.fs.CHDFSHa S3 또는 액세스 모드의 경우 org.apache.hadoop.fs.CosNFi 됩니다.

5. 주의사항

1. 이전 hadoop cos jar 패키지를 사용하여 메타데이터 가속이 활성화된 버킷에 액세스할 수 없습니다.
2. Hadoop cos ≤ 8.1.5 버전 posix를 사용하여 메타데이터 가속이 활성화된 버킷에 액세스하려면 콘솔에서 ranger 인증을 해제해야 합니다. 8.1.5 이상 버전에서는 콘솔에서 ranger 인증을 열 수 있습니다.

©2013-2022 Tencent Cloud. All rights reserved.

CDH 클러스터에서 HDFS 프로토콜을 통해 COS에 액세스

최종 업데이트 날짜: : 2022-05-05 14:14:55

소개

CDH(Cloudera's Distribution, including Apache Hadoop)는 업계에서 각광받는 Hadoop의 릴리스 버전입니다. 본 문서는 CDH 환경에서 HDFS 프로토콜을 통해 객체 스토리지(Cloud Object Storage, COS) 버킷에 액세스하는 방법을 소개합니다. 이 기능을 통해 빅 데이터 컴퓨팅과 스토리지를 분리하여 효율적인 저비용 빅 데이터 솔루션을 제공합니다.

> 주의 :
> HDFS 프로토콜을 통해 COS 버킷에 액세스하려면 먼저 메타데이터 가속 기능을 활성화해야 합니다.

빅 데이터 컴포넌트에 대한 현재 COS 지원은 다음과 같습니다.

모듈 이름	CHDFS 빅 데이터 모듈 지원 현황	서비스 모듈 재시작 필요 여부
Yarn	지원	NodeManager 재시작
Yarn	지원	NodeManager 재시작
Hive	지원	HiveServer 및 HiveMetastore 재시작
Spark	지원	NodeManager 재시작
Sqoop	지원	NodeManager 재시작
Presto	지원	HiveServer, HiveMetastore, Presto 재시작
Flink	지원	필요 없음
Impala	지원	필요 없음
EMR	지원	필요 없음
자체구축 모듈	향후 지원	없음
HBase	권장하지 않음	없음

©2013-2022 Tencent Cloud. All rights reserved.

버전 종속

본 문서에 종속된 모듈 버전은 다음과 같습니다.

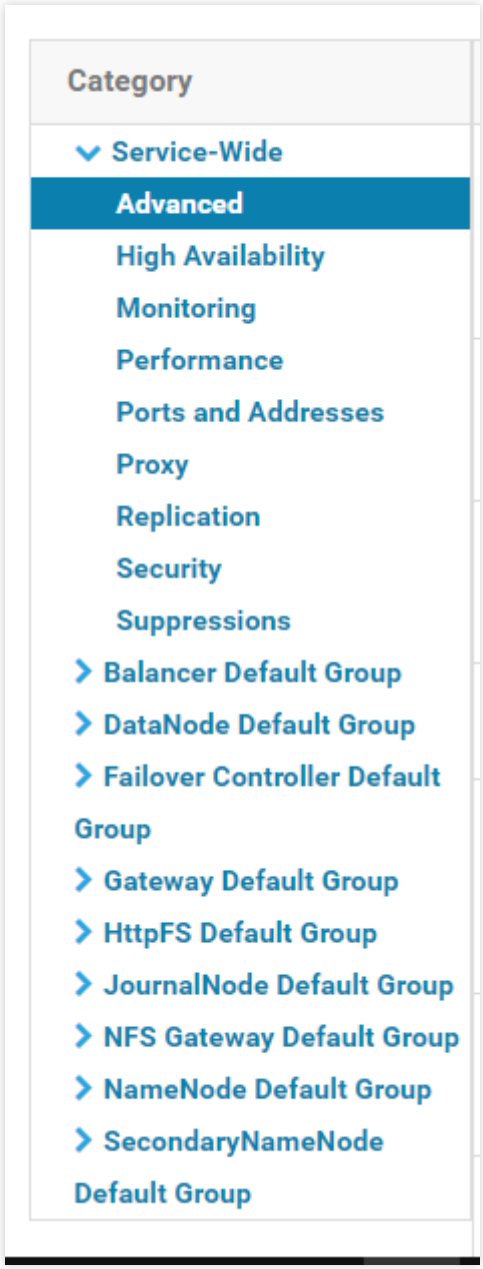
- CDH 5.16.1
- Hadoop 2.6.0

사용 방법

스토리지 환경 설정

1. CDH 관리 페이지에 로그인합니다.
2. 시스템 메인 페이지에서 **설정>서비스 범위>고급**을 선택하여 코드 스니펫 고급 설정 페이지로 이동하면 다음 그림과 같이 표시됩니다.

©2013-2022 Tencent Cloud. All rights reserved.



3. `Cluster-wide Advanced Configuration Snippet(Safety Valve) for core-site.xml` 코드 창에 COS 빅 데이터 서비스 설정을 입력합니다.

```
<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>
<!--로컬 cache의 임시 디렉터리로, 데이터 읽기/쓰기의 경우 메모리 cache가 부족할 때는 로컬 디스크에 쓰기 하고 관련 경로가 없을 때는 자동 생성합니다.-->
```

©2013-2022 Tencent Cloud. All rights reserved.

```
<property>
<name>fs.ofs.tmp.cache.dir</name>
<value>/data/emr/hdfs/tmp/chdfs/</value>
</property>
<!--appId-->
<property>
<name>fs.ofs.user.appid</name>
<value>1250000000</value>
</property>
```

다음은 필수 설정 항목(core-site.xml에 추가 필요)입니다. 기타 설정 항목은 컴퓨팅 클러스터에 COS 버킷 마운트를 참고하십시오.

설정 항목	값	의미
fs.ofs.user.appid	1250000000	사용자 appid
fs.ofs.tmp.cache.dir	/data/emr/hdfs/tmp/chdfs/	로컬 cache용 임시 디
fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	FileSystem용 chdfs 구 com.qcloud.chdfs.fs. 로 고정
fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	AbstractFileSystem용 com.qcloud.chdfs.fs. 정

4. HDFS 서비스 작업을 진행하기 위해 클라이언트 배포 설정을 클릭하면 core-site.xml 설정이 클러스터 기기에 업데이트됩니다. 5. 최신 [클라이언트 설치 패키지](https://github.com/tencentyun/chdfs-hadoop-plugin/tree/master/jar)를 CDH HDFS 서비스의 jar 패키지 경로에 넣고 실제 값에 따라 교체하십시오. 예시는 다음과 같습니다.

```
cp chdfs_hadoop_plugin_network-2.0.jar /opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16
.1.p0.3/lib/hadoop-hdfs/
```

> 주의 :
> 클러스터의 기기마다 동일한 위치에 SDK 패키지를 설치해야 합니다.

데이터 마이그레이션

Hadoop Distcp 툴을 사용해 CDH HDFS 데이터를 COS 버킷에 마이그레이션합니다. 자세한 내용은 Hadoop 파일 시스템과 COS 간의 데이터 마이그레이션을 참고하십시오.

©2013-2022 Tencent Cloud. All rights reserved.

빅 데이터 세트로 CHDFS 사용

MapReduce

작업 단계

1. 데이터 마이그레이션 섹션에 따라 HDFS 관련 설정을 완료하고, COS의 클라이언트 설치 패키지를 HDFS의 해당 디렉터리에 넣습니다.
2. CDH 시스템 메인 페이지에서 YARN을 찾아 NodeManager 서비스를 재시작합니다. TeraGen 명령어는 재시작할 필요가 없으나 TeraSort는 비즈니스 내부 로직으로 인해 NodeManger를 재시작해야 하므로, NodeManager 서비스를 일괄 재시작할 것을 권장합니다.

예시

Hadoop 표준 테스트의 TeraGen 및 TeraSort입니다.

```
hadoop jar ./hadoop-mapreduce-examples-2.7.3.jar teragen -Dmapred.map.tasks=4 109
9 ofs://examplebucket-1250000000/teragen_5/
hadoop jar ./hadoop-mapreduce-examples-2.7.3.jar terasort -Dmapred.map.tasks=4 of
s://examplebucket-1250000000/teragen_5/ ofs://examplebucket-1250000000/result14
```

설명 :
`ofs:// schema` 뒷부분을 사용자 CHDFS의 마운트 포인트 경로로 변경하십시오.

Hive

MR 엔진

작업 단계

1. 데이터 마이그레이션 섹션에 따라 HDFS 관련 설정을 완료하고, COS의 클라이언트 설치 패키지를 HDFS의 해당 디렉터리에 넣습니다.
2. CDH 메인 페이지에서 HIVE 서비스를 찾아 Hiveserver2와 HiverMetastore 역할을 재시작합니다.

예시

사용자의 실제 작업 쿼리입니다. Hive 명령 라인을 실행하여 Location을 생성하고, 이를 CHDFS의 파티션 테이블로 사용하는 경우입니다.

```
CREATE TABLE `report.report_o2o_pid_credit_detail_grant_daily`(
`cal_dt` string,
`change_time` string,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
`merchant_id` bigint,
`store_id` bigint,
`store_name` string,
`wid` string,
`member_id` bigint,
`meber_card` string,
`nickname` string,
`name` string,
`gender` string,
`birthday` string,
`city` string,
`mobile` string,
`credit_grant` bigint,
`change_reason` string,
`available_point` bigint,
`date_time` string,
`channel_type` bigint,
`point_flow_id` bigint)
PARTITIONED BY (
`topicdate` string)
ROW FORMAT SERDE
'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
STORED AS INPUTFORMAT
'org.apache.hadoop.hive.ql.io.orc.OrcInputFormat'
OUTPUTFORMAT
'org.apache.hadoop.hive.ql.io.orc.OrcOutputFormat'
LOCATION
'ofs://examplebucket-1250000000/user/hive/warehouse/report.db/report_o2o_pid_cred
it_detail_grant_daily'
TBLPROPERTIES (
'last_modified_by'='work',
'last_modified_time'='1589310646',
'transient_lastDdlTime'='1589310646')
```

SQL 쿼리 실행:

```
select count(1) from report.report_o2o_pid_credit_detail_grant_daily;
```

©2013-2022 Tencent Cloud. All rights reserved.

모니터링 결과는 다음과 같습니다.

```
hive> select count(1) from report.report_o2o_pid_credit_detail_grant_daily;
Query ID = root_20200713121818_3a911a0c-2496-4e7e-b59d-b2a26266a6ab
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1594351711155_0014, Tracking URL = http://hadoop01:8088/prox
y/application_1594351711155_0014/
Kill Command = /opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/lib/hadoop/bin
/hadoop job  -kill job_1594351711155_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-07-13 12:18:19,189 Stage-1 map = 0%,  reduce = 0%
2020-07-13 12:18:25,391 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 8.89 s
ec
2020-07-13 12:18:30,544 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.8
 sec
MapReduce Total cumulative CPU time: 10 seconds 800 msec
Ended Job = job_1594351711155_0014
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 10.8 sec   HDFS Read: 17383
HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 10 seconds 800 msec
OK
27677
Time taken: 19.128 seconds, Fetched: 1 row(s)
```

Tez 엔진

Tez 엔진은 COS의 클라이언트 설치 패키지를 Tez의 압축 패키지 내로 가져와야 합니다. 다음은 apache-tez.0.8.5를 예시로 한 설명입니다.

작업 단계

1. CDH 클러스터에 설치한 tez 패키지를 찾은 후 압축을 해제합니다. /usr/local/service/tez/tez-0.8.5.tar.gz를 예로 들 수 있습니다.
2. 압축 해제된 디렉터리에 COS 클라이언트 설치 패키지를 넣고 다시 압축하여 압축된 패키지를 출력합니다.
3. 새로 생성한 압축 패키지를 tez.lib.uris에서 지정한 경로(경로가 이미 있는 경우 변경 가능)에 업로드합니다.
4. CDH 메인 페이지에서 HIVE를 찾아 hiveserver와 hivemetastore를 재시작합니다.

Spark

작업 단계

1. 데이터 마이그레이션 섹션에 따라 HDFS 관련 설정을 완료하고, COS의 클라이언트 설치 패키지를 HDFS의 해당 디렉터리에 넣습니다.
2. NodeManager 서비스를 재시작합니다.

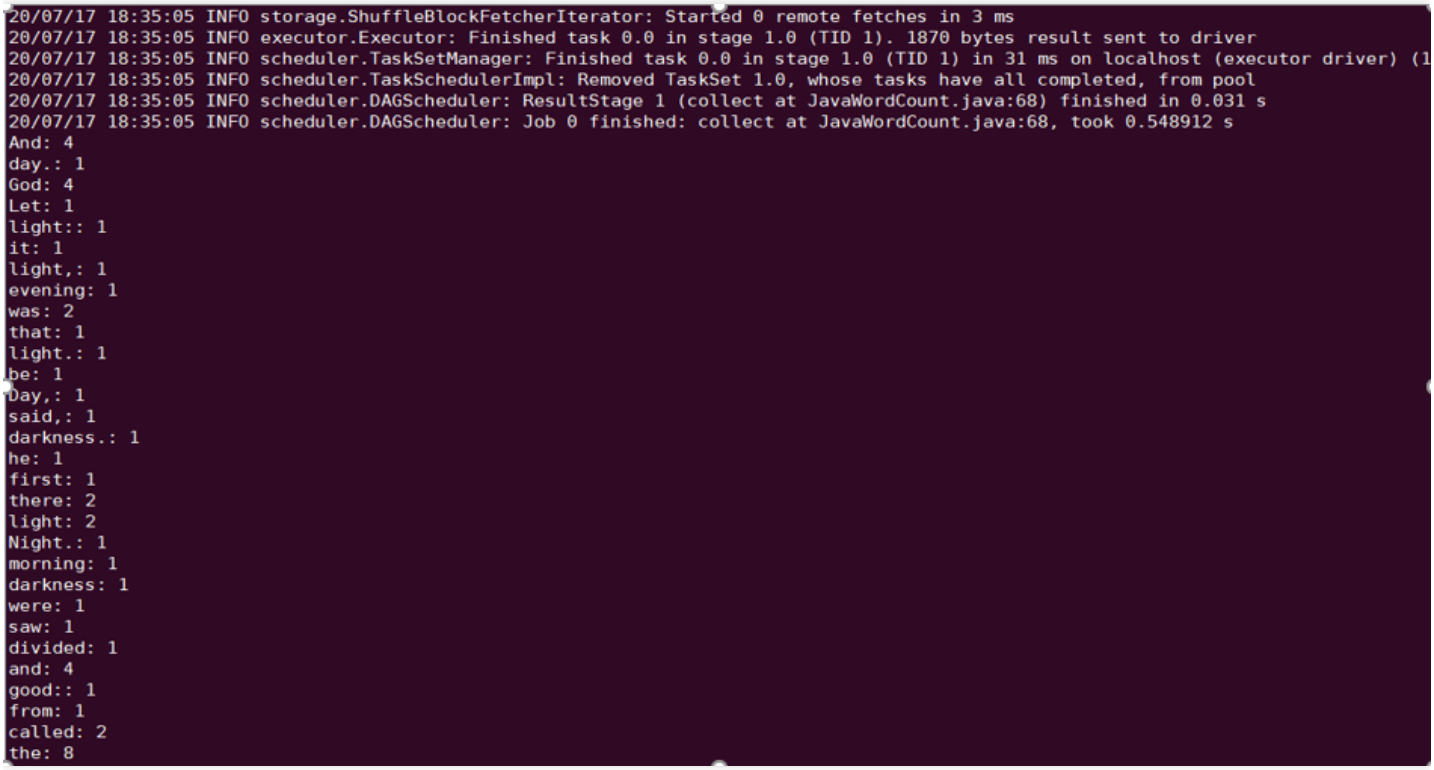
©2013-2022 Tencent Cloud. All rights reserved.

예시

Spark example word count 테스트를 진행합니다.

```
spark-submit --class org.apache.spark.examples.JavaWordCount --executor-memory 4g
--executor-cores 4 ./spark-examples-1.6.0-cdh5.16.1-hadoop2.6.0-cdh5.16.1.jar of
s://examplebucket-1250000000/wordcount
```

실행 결과는 다음과 같습니다.



Sqoop

작업 단계

1. 데이터 마이그레이션 섹션에 따라 HDFS 관련 설정을 완료하고, COS의 클라이언트 설치 패키지를 HDFS의 해당 디렉터리에 넣습니다.
2. COS의 클라이언트 설치 패키지를 sqoop 디렉터리에 넣어야 합니다(예시: /opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/lib/sqoop/).
3. NodeManager 서비스를 재시작합니다.

예시

MYSQL 테이블을 COS에 내보냅니다. 관계형 데이터베이스 및 HDFS의 가져오기/내보내기 문서를 참고하여 테스트를 진행하십시오.

©2013-2022 Tencent Cloud. All rights reserved.

```
sqoop import --connect "jdbc:mysql://IP:PORT/mysql" --table sqoop_test --username
root --password 123 --target-dir ofs://examplebucket-1250000000/sqoop_test
```

실행 결과는 다음과 같습니다.

```
20/07/17 18:48:33 INFO mapreduce.Job:  map 100% reduce 0%
20/07/17 18:48:33 INFO mapreduce.Job: Job job_1594976906551_0011 completed successfully
20/07/17 18:48:33 INFO mapreduce.Job: Counters: 35
        File System Counters
                FILE: Number of bytes read=0
                FILE: Number of bytes written=526689
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=295
                HDFS: Number of bytes written=0
                HDFS: Number of read operations=3
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=0
                OFS: Number of bytes read=0
                OFS: Number of bytes written=104
                OFS: Number of read operations=0
                OFS: Number of large read operations=0
                OFS: Number of write operations=3
        Job Counters
                Launched map tasks=3
                Other local map tasks=3
                Total time spent by all maps in occupied slots (ms)=36308
                Total time spent by all reduces in occupied slots (ms)=0
                Total time spent by all map tasks (ms)=9077
                Total vcore-milliseconds taken by all map tasks=9077
                Total megabyte-milliseconds taken by all map tasks=37179392
        Map-Reduce Framework
                Map input records=3
                Map output records=3
                Input split bytes=295
                Spilled Records=0
                Failed Shuffles=0
                Merged Map outputs=0
                GC time elapsed (ms)=277
                CPU time spent (ms)=6430
                Physical memory (bytes) snapshot=1371717632
                Virtual memory (bytes) snapshot=18832379904
                Total committed heap usage (bytes)=6655311872
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=104
20/07/17 18:48:33 INFO mapreduce.ImportJobBase: Transferred 0 bytes in 13.0961 seconds (0 bytes/sec)
20/07/17 18:48:33 INFO mapreduce.ImportJobBase: Retrieved 3 records.
20/07/17 18:48:33 INFO fs.CHDFSHadoopFileSystemAdapter: actual-file-system-close usedTime: 13
20/07/17 18:48:33 INFO fs.CHDFSHadoopFileSystemAdapter: end-close time: 1594982913402, total-used-time: 13650
```

Presto

작업 단계

1. 데이터 마이그레이션 섹션에 따라 HDFS 관련 설정을 완료하고, COS의 클라이언트 설치 패키지를 HDFS의 해당 디렉터리에 넣습니다.
2. COS의 클라이언트 설치 패키지를 presto 디렉터리에 넣어야 합니다(예시: /usr/local/services/cos_presto/plugin/hive-hadoop2).
3. presto는 hadoop common의 gson-2...jar를 로딩할 수 없으므로 gson-2...jar 또한 presto 디렉터리에 넣어야 합니다(예시: /usr/local/services/cos_presto/plugin/hive-hadoop2, COS만 gson에 종속됨).

©2013-2022 Tencent Cloud. All rights reserved.

4. HiveServer, HiveMetaStore, Presto 서비스를 재시작합니다.

예시

HIVE에서 Location이 COS인 테이블 쿼리 생성:

```
select * from chdfs_test_table where bucket is not null limit 1;
```

> 설명 :
> chdfs_test_table은 location이 ofs scheme인 테이블입니다.

쿼리 결과는 다음과 같습니다.

```
presto:inventory_search> select * from chdfs_test_table_0720 where bucket is not null limit 1;
   appid    |       bucket        |                          path                          |  size   |
------------+---------------------+--------------------------------------------------------+---------+
 125        | ssuupv80105841qq206 | 444600684/20190316/3/01a9ee49bd4045179c92e319ff03b810 | 3800424 |
(1 row)

Query 20200720 112833 00061 thb89, FINISHED, 7 nodes
```

©2013-2022 Tencent Cloud. All rights reserved.

Hadoop FileSystem API 코드를 사용하여 COS 메타데이터 가속 버킷에 액세스하기

최종 업데이트 날짜: : 2023-03-20 15:08:11

작업 시나리오

COS(Cloud Object Storage) 버킷에 대해 메타데이터 가속화가 활성화된 경우 Java 코드를 사용하여 Hadoop 명령 라인 및 빅 데이터 컴포넌트를 사용하는 것 외에도 Hadoop Filesystem API를 통해 버킷에 액세스할 수 있습니다. 본문에서는 그 방법을 설명합니다.

전제 조건

- 메타데이터 가속이 활성화되었고 환경 배포 및 HDFS 프로토콜 구성이 올바르게 수행되었는지 확인하십시오. 자세한 내용은 HDFS 프로토콜을 사용하여 메타데이터 가속이 활성화된 버킷에 액세스를 참고하십시오.
- Hadoop 환경이 있는 경우 Hadoop 명령 라인에 올바르게 액세스할 수 있는지 확인할 수 있습니다.

작업 단계

1. maven 프로젝트를 생성하고 maven의 pom.xml에 다음 종속 항목(실제 Hadoop 버전 및 환경에 따라 hadoop-common 패키지 버전 설정)을 추가합니다.

```
<dependencies>
<dependency>
<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-common</artifactId>
<version>2.8.5</version>
<scope>provided</scope>
</dependency>
</dependencies>
```

2. 다음 hadoop 코드를 참고하여 변경합니다. 구성 항목은 CHDFS 마운트에 설명된 대로 수정할 수 있습니다. **데이터 지속성 및 가시성 관련 지침에 특히 주의하십시오.**
일부 자주 사용되는 파일 시스템 작업 인터페이스는 다음과 같습니다. 다른 API는 Hadoop FileSystem API를 참고하십시오.

©2013-2022 Tencent Cloud. All rights reserved.

```
package com.qcloud.cos.demo;

import org.apache.commons.io.IOUtils;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FSDataInputStream;
import org.apache.hadoop.fs.FSDataOutputStream;
import org.apache.hadoop.fs.FileChecksum;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;

import java.io.IOException;
import java.net.URI;
import java.nio.ByteBuffer;

public class Demo {
private static FileSystem initFS() throws IOException {
Configuration conf = new Configuration();
// 구성 항목 참고: https://www.tencentcloud.com/document/product/1106/41965
// 다음 구성은 필수 항목입니다

conf.set("fs.cosn.trsf.fs.ofs.impl", "com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAd
apter");
conf.set("fs.cosn.trsf.fs.AbstractFileSystem.ofs.impl", "com.qcloud.chdfs.fs.CHDF
SDelegateFSAdapter");
conf.set("fs.cosn.trsf.fs.ofs.tmp.cache.dir", "/data/chdfs_tmp_cache");
// appid를 실제 appid로 교체
conf.set("fs.cosn.trsf.fs.ofs.user.appid", "1250000000");
// region을 실제 리전으로 교체
conf.set("fs.cosn.trsf.fs.ofs.bucket.region", "ap-beijing")
// 기타 옵션 설정 항목 참고: https://www.tencentcloud.com/document/product/1106/4196
5

String chdfsUrl = "cosn://examplebucket-12500000000/";
return FileSystem.get(URI.create(chdfsUrl), conf);
}

private static void mkdir(FileSystem fs, Path filePath) throws IOException {
fs.mkdirs(filePath);
}

private static void createFile(FileSystem fs, Path filePath) throws IOException {
// 파일 생성 (이미 있는 경우 덮어쓰기)
// if the parent dir does not exist, fs will create it!
FSDataOutputStream out = fs.create(filePath, true);
try {
```

©2013-2022 Tencent Cloud. All rights reserved.

```
// 파일 입력
String content = "test write file";
out.write(content.getBytes());
} finally {
// close가 성공적으로 반환되면 데이터 쓰기가 성공한 것이며, 예외가 발생하면 데이터 쓰기가 실패한 것입니다
out.close();
}
}

private static void readFile(FileSystem fs, Path filePath) throws IOException {
FSDataInputStream in = fs.open(filePath);
try {
byte[] buf = new byte[4096];
int readLen = -1;
do {
readLen = in.read(buf);
} while (readLen >= 0);
} finally {
IOUtils.closeQuietly(in);
}
}


private static void queryFileOrDirStatus(FileSystem fs, Path path) throws IOExcep
tion {
FileStatus fileStatus = fs.getFileStatus(path);
if (fileStatus.isDirectory()) {
System.out.printf("path %s is dir\n", path);
return;
}

long fileLen = fileStatus.getLen();
long accessTime = fileStatus.getAccessTime();
long modifyTime = fileStatus.getModificationTime();
String owner = fileStatus.getOwner();
String group = fileStatus.getGroup();


System.out.printf("path %s is file, fileLen: %d, accessTime: %d, modifyTime: %d,
owner: %s, group: %s\n",
path, fileLen, accessTime, modifyTime, owner, group);
}


// 기본 인증 유형은 COMPOSITE-CRC32C
private static void getFileCheckSum(FileSystem fs, Path path) throws IOException
```

©2013-2022 Tencent Cloud. All rights reserved.

```
{
FileChecksum checksum = fs.getFileChecksum(path);
System.out.printf("path %s, checkSumType: %s, checkSumCrcVal: %d\n",
path, checksum.getAlgorithmName(), ByteBuffer.wrap(checksum.getBytes()).getInt
());
}


private static void copyFileFromLocal(FileSystem fs, Path chdfsPath, Path localPa
th) throws IOException {
fs.copyFromLocalFile(localPath, chdfsPath);
}


private static void copyFileToLocal(FileSystem fs, Path chdfsPath, Path localPat
h) throws IOException {
fs.copyToLocalFile(chdfsPath, localPath);
}


private static void renamePath(FileSystem fs, Path oldPath, Path newPath) throws
IOException {
fs.rename(oldPath, newPath);
}


private static void listDirPath(FileSystem fs, Path dirPath) throws IOException {
FileStatus[] dirMemberArray = fs.listStatus(dirPath);


for (FileStatus dirMember : dirMemberArray) {
System.out.printf("dirMember path %s, fileLen: %d\n", dirMember.getPath(), dirMem
ber.getLen());
}
}


// 재귀 삭제 마크는 디렉터리 삭제에 사용됩니다
// 재귀가 false 이고 dir이 비어있지 않으면 작업이 수행되지 않습니다
private static void deleteFileOrDir(FileSystem fs, Path path, boolean recursive)
throws IOException {
fs.delete(path, recursive);
}


private static void closeFileSystem(FileSystem fs) throws IOException {
fs.close();
```

©2013-2022 Tencent Cloud. All rights reserved.

```
}


public static void main(String[] args) throws IOException {
// 파일 시스템 초기화
FileSystem fs = initFS();


// 파일 생성
Path chdfsFilePath = new Path("/folder/exampleobject.txt");
createFile(fs, chdfsFilePath);


// 파일 불러오기
readFile(fs, chdfsFilePath);


// 파일 또는 디렉터리 조회
queryFileOrDirStatus(fs, chdfsFilePath);


// 파일 검사합 가져오기
getFileCheckSum(fs, chdfsFilePath);


// 로컬에서 파일 복사
Path localFilePath = new Path("file:///home/hadoop/cosn_demo/data/exampleobject.t
xt");
copyFileFromLocal(fs, chdfsFilePath, localFilePath);


// 파일을 로컬로 가져오기
Path localDownFilePath = new Path("file:///home/hadoop/cosn_demo/data/exampleobje
ct.txt");
copyFileToLocal(fs, chdfsFilePath, localDownFilePath);


// 이름 변경
Path newPath = new Path("/doc/example.txt");
renamePath(fs, chdfsFilePath, newPath);


// 파일 삭제
deleteFileOrDir(fs, newPath, false);


// 디렉터리 생성
```

©2013-2022 Tencent Cloud. All rights reserved.

```
Path dirPath = new Path("/folder");
mkdir(fs, dirPath);


// 디렉터리에 파일 생성
Path subFilePath = new Path("/folder/exampleobject.txt");
createFile(fs, subFilePath);


// 디렉터리 나열
listDirPath(fs, dirPath);


// 디렉터리 삭제
deleteFileOrDir(fs, dirPath, true);


// 파일 시스템 비활성화
closeFileSystem(fs);
}
}
```

3. 프로젝트를 컴파일하고 실행합니다.

설명 :

- 코드를 실행하기 전에 메타데이터 가속 버킷에 종속된 Hadoop common 패키지 및 Jar 패키지의 경로를 포함해야 하는 classpath를 올바르게 설정해야 합니다.
- EMR 환경의 경우 HDFS 프로토콜을 사용하여 메타데이터 가속이 활성화된 버킷에 액세스에 설명된 단계를 따르면 Hadoop common 패키지는 일반적으로 `/usr/local/service/hadoop/share/hadoop/common/` 디렉터리에 있으며, 메타데이터 가속 버킷에 의존하는 Jar 패키지는 일반적으로 `/usr/local/service/hadoop/share/hadoop/common/lib/` 디렉터리에 있습니다.

©2013-2022 Tencent Cloud. All rights reserved.

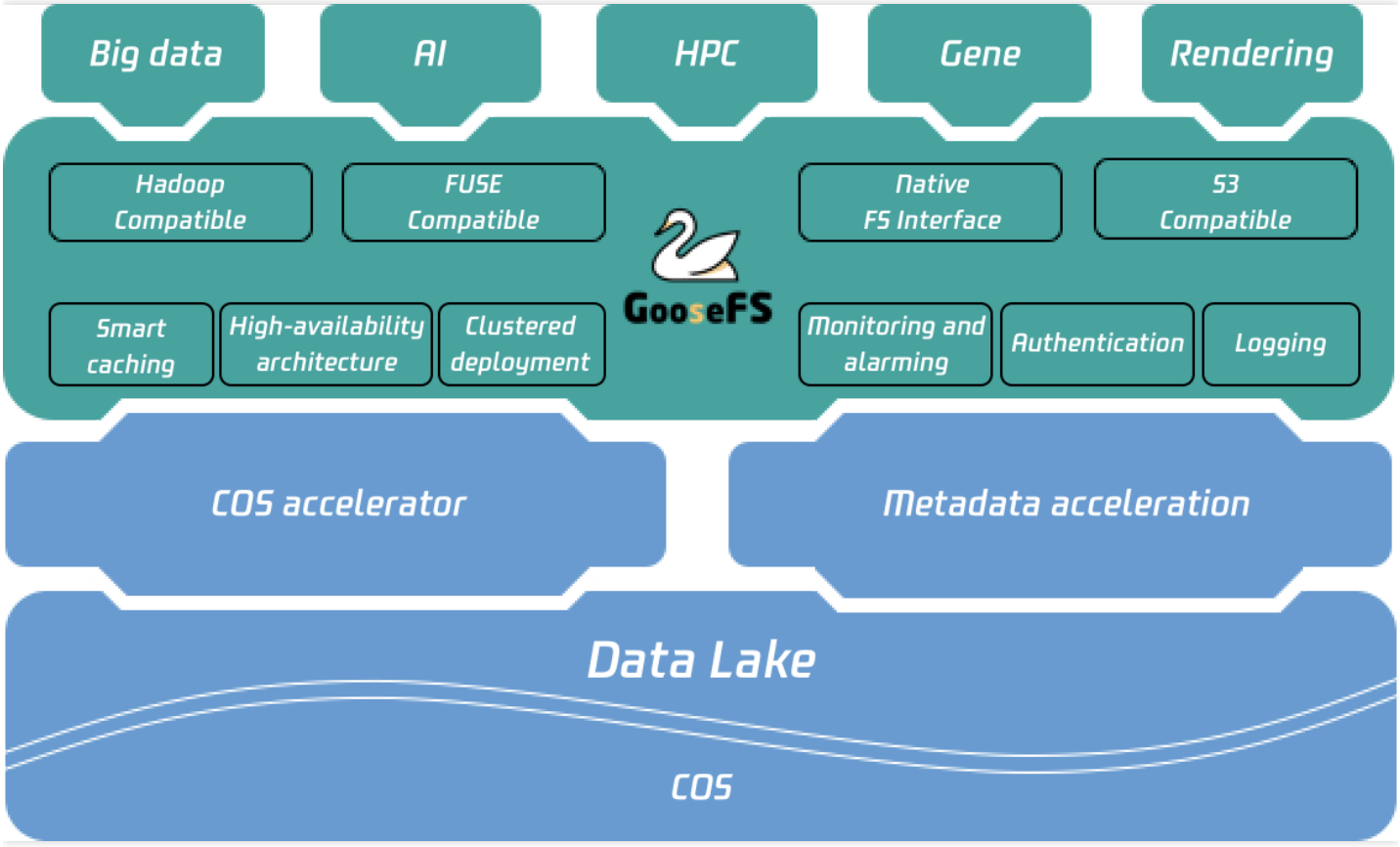
데이터 레이크 가속기 GooseFS 제품 개요

최종 업데이트 날짜: : 2022-06-09 15:04:19

Data Lake Accelerator Goose FileSystem(GooseFS)은 Tencent Cloud에서 제공하는 고가용성, 안정성, 탄력성 Data Lake 가속 서비스입니다. Cloud Object Storage(COS)를 데이터 레이크 스토리지 기반으로 사용하여 스토리지 비용을 줄이고, 데이터 레이크 생태계에서 컴퓨팅 애플리케이션에 대한 통합 엔트리를 제공하여 빅 데이터 분석, 머신러닝 및 AI와 같은 비즈니스 액세스를 가속화합니다. 강력한 분산 클러스터 아키텍처를 사용하여 상위 레이어 컴퓨팅 애플리케이션을 위한 통합 네임스페이스 및 액세스 프로토콜을 제공하므로 다양한 스토리지 시스템에서 데이터를 더 쉽게 관리하고 전송할 수 있습니다.

제품 기능

GooseFS는 원스톱 캐시 솔루션을 제공하는 것을 목표로 하며 데이터 지역성, 고속 캐시 및 통합 스토리지 액세스 구문을 활용하는 고유한 강점을 가지고 있습니다. 아래와 같이 '컴퓨팅과 스토리지'를 연결하는 커넥터로서 데이터 레이크 생태계에서 핵심적인 역할을 합니다.



©2013-2022 Tencent Cloud. All rights reserved.

GooseFS에는 다음과 같은 기능이 있습니다.

1. 캐시 가속 및 데이터 지역성(Locality): GooseFS를 컴퓨팅 노드와 함께 배포하여 데이터 지역성을 개선할 수 있습니다. 고속 캐시 기능은 스토리지 성능을 향상시키고 COS에 대한 데이터 쓰기 대역폭을 증가시킬 수 있습니다.
2. 통합 스토리지 구문: GooseFS는 COS, Hadoop, S3, K8S CSI, FUSE 등 여러 스토리지 서비스의 스토리지 구문을 지원하기 위해 UFS(Unified FileSystem) 구문을 제공하므로 다양한 생태계 및 사용 사례에 적합합니다.
3. Tencent Cloud 생태계 서비스: 로그, 인증, 모니터링 서비스를 제공하여 COS와 동일한 운영이 가능합니다.
4. Namespace 관리 기능: GooseFS는 다양한 비즈니스 및 Under File System에 대해 다양한 캐시 읽기/쓰기 및 TTL 관리 정책을 제공합니다.
5. 메타데이터 Table 감지: GooseFS Catalog는 빅 데이터 시나리오에서 메타데이터 Table을 감지하여 Table 수준에서 Cache를 미리 가져올 수 있습니다.

제품 장점

GooseFS는 데이터 레이크 시나리오에서 다음과 같은 장점을 가지고 있습니다.

데이터 I/O 성능

GooseFS는 상위 레이어 컴퓨팅 애플리케이션이 데이터 I/O를 가속화하기 위해 자주 액세스해야 하는 핫 데이터를 투명하고 효율적으로 캐시할 수 있는 컴퓨팅 노드 근처에 분산 공유 캐시를 활성화합니다. 또한 파일 데이터 쿼리 및 파일 목록 표시와 같은 빅 데이터 시나리오에서 메타데이터 작업을 더 빠르게 수행할 수 있는 메타데이터 캐시 기능이 있습니다. 빅 데이터 버킷과 함께 파일 이름 변경을 더욱 가속화할 수 있습니다. 또한 비즈니스 요구 사항에 따라 MEM, SSD, NVME 및 HDD를 비롯한 다양한 스토리지 미디어를 선택하여 비즈니스 비용과 데이터 액세스 성능의 균형을 맞출 수 있습니다.

통합 스토리지

GooseFS는 COS 뿐만 아니라 HDFS, K8S CSI, FUSE 등 다양한 스토리지 서비스의 스토리지 구문을 지원하는 통합 네임스페이스를 제공합니다. 이는 상위 레이어 비즈니스를 위한 통합 통합 스토리지 솔루션을 제공하고 비즈니스 운영 구성을 단순화합니다. 통합 스토리지는 서로 다른 데이터베이스 간의 장벽을 허물고 상위 레이어 애플리케이션에서 데이터를 더 쉽게 관리하고 전송할 수 있도록 하여 데이터 사용 효율성을 향상시킵니다.

생태계 친화성

GooseFS는 Tencent Cloud 빅 데이터 플랫폼 프레임워크와 완벽하게 호환되며 맞춤형 방식으로 온프레미스에 배포할 수도 있어 우수한 생태계 친화성을 보여줍니다. 예를 들어 Elastic MapReduce(EMR)에서 GooseFS를 사용하여 빅 데이터 비즈니스를 가속화하고 CVM 또는 자체 구축 IDC에 편리하게 배포할 수 있습니다. 또한 투명한 가속을 지원합니다. 이미 COSN 및 CHDFS를 활성화한 경우 비즈니스 코드 및 액세스 경로를 수정할 필요 없이 GooseFS를 통해 COSN 및 CHDFS 비즈니스 액세스를 자동으로 가속화하도록 구성을 수정하기만 하면 됩니다.

©2013-2022 Tencent Cloud. All rights reserved.

업데이트 로그

최종 업데이트 날짜: : 2023-05-19 15:20:02

GooseFS 버전의 업데이트 리스트는 아래와 같습니다. 기타 건의사항 및 문의사항은 문의하기를 통해 연락주시기 바랍니다.

버전 번호	업데이트 날짜	업데이트 설명	다운로드 링크
1.3.0	2022년 7월 25일	• 새로운 기능 i. 지원되는 Kerberos 인증. ii. 기본 POSIX 의미 체계를 사용하여 메타데이터 가속이 활성화된 버킷에 대한 액세스를 지원합니다. iii. 빈번한 캐시 교체를 피하기 위해 Master 노드 캐시 제거 정책에 대해 LRU 알고리즘을 지원합니다. iv. Master 노드에 대해 만료된 메타데이터 정리를 지원합니다. • 최적화 i. GooseFS Master 노드의 잠금 병목 현상 문제를 최적화하여 Master QPS를 크게 개선하고 성능을 약 35% 향상시켰습니다. ii. GooseFS Worker 노드의 성능을 향상시키기 위해 지원되는 동시 Format입니다. iii. GooseFS Fuse 클라이언트에 지원되는 덮어쓰기 작업입니다. iv. GooseFS Fuse 클라이언트에서 `ls` 명령의 메모리 사용을 최적화했습니다. v. GooseFS HDFS 클라이언트에서 ListNamespace의 성능을 최적화했습니다. • Bug 수정 i. 메모리 유출을 방지하기 위해 RocksDB 유출 및 Core 문제를 수정했습니다. ii. Zookeeper Curator 로그 출력 오류가 수정되었습니다. iii. 부정확한 UFS 대역폭 읽기 문제가 수정되었습니다. iv. DistributedLoad 동안 과도한 로그 인쇄로 인해 발생하는 LostWorker 문제를 수정했습니다.	• GooseFS: 다운로드 • GooseFS9KonaJDK 버전): 다운로드 • GooseFS 클라이언트: 다운로드 • GooseFS 클라이언트 (KonaJDK 버전): 다운로드 • GooseFS FUSE 클라이언트: 다운로드 • GooseFS FUSE 클라이언트(KonaJDK 버전): 다운로드

©2013-2022 Tencent Cloud. All rights reserved.

버전 번호	업데이트 날짜	업데이트 설명	다운로드 링크
1.2.	2022년 1월 25일	• 새로운 기능 i. GooseFS를 사용하여 CLS에 로그를 업로드하도록 지원됩니다. ii. Inode 및 Block 메타데이터를 사용하여 RocksDB의 분리된 구성을 지원합니다. iii. 재해 복구 조치를 개선하기 위해 GooseFS Client에 핫 스위치 기능을 추가했습니다. iv. 특정 전체 연결 로그에 대한 인프라를 추가했습니다. • 최적화 i. 높은 Raft 장애 조치 지연 시간을 최적화했습니다. ii. GooseFS HCFS Client의 메모리 사용량을 최적화했습니다. • Bug 수정 i. Journal 무질서 문제를 수정했습니다. ii. Ratis 교착 상태로 인한 GRPC 문제를 수정했습니다. iii. 잘못된 HDFSUnderFileSystemFactory 로드 위치를 수정했습니다. iv. log4j2의 보안 취약점을 수정했습니다. v. ufsPath 접두사 검사 오류를 수정했습니다.	• GooseFS: 다운로드 • GooseFS(KonaJDK 버전): 다운로드 • GooseFS FUSE 클라이언트: 다운로드
1.1.0	2021년 9월 1일	• 새로운 기능: i. ranger 인증을 지원합니다. ranger 플러그인을 다운로드하려면 여기를 클릭하십시오. ii. 동시 list를 지원합니다. iii. distributedLoad 디렉터리 원자성을 지원합니다. iv. namespace 캐시 얼로우리스트를 지원합니다. v. 감지할 수 없는 ofs scheme 가속을 지원합니다. • 최적화: i. cli의 각 subcmd에 대한 help 명령을 추가했습니다. ii. table 마운트 중 namespace 존재 여부를 확인하는 로직을 최적화했습니다. iii. job worker/worker metrics에 대한 모니터링이 개선되었습니다. • Bug 수정: distributedLoad 읽기 팽창 문제를 수정했습니다.	이 버전은 더 이상 유지되지 않습니다. 최신 버전을 다운로드하십시오.

©2013-2022 Tencent Cloud. All rights reserved.

버전 번호	업데이트 날짜	업데이트 설명	다운로드 링크
1.0.0	2021년 6월 1일	1. namespace 기반 읽기/쓰기 정책 및 TTL 관리. 2. Hive Table 및 Table Partition 수준에서 프리패치합니다. 3. 감지할 수 없는 가속을 달성하기 위해 기존 COSN 사용자의 경로와 호환됩니다. 4. GooseFS와의 Fluid 통합. 5. CHDFS는 통합을 지원합니다.	이 버전은 더 이상 유지되지 않습니다. 최신 버전을 다운로드하십시오.

©2013-2022 Tencent Cloud. All rights reserved.

시작하기

최종 업데이트 날짜: : 2023-04-06 14:46:09

본 문서는 GooseFS를 로컬 장치에 빠르게 배포하고, 디버깅을 수행하고, COS(Cloud Object Storage)를 원격 스토리지로 사용하는 방법을 설명합니다.

전제 조건

GooseFS 사용 전 아래 작업을 준비해야 합니다.

1. COS 서비스에서 버킷을 생성해 원격 스토리지로 만듭니다. 작업 가이드에 대한 자세한 내용은 콘솔 시작하기를 참고하십시오.
2. Java 8 또는 이후 버전을 설치합니다.
3. SSH를 통한 LocalHost 연결과 원격 로그인이 가능하도록 SSH를 설치합니다.
4. 시작하기의 안내에 따라 CVM 인스턴스를 구매하고 디스크가 인스턴스에 마운트되었는지 확인합니다.

GooseFS 다운로드 및 구성

1. 로컬 디렉터리를 생성하고 입력한 다음(필요에 따라 다른 디렉터리를 선택할 수도 있음) goosefs-1.4.1-bin.tar.gz를 다운로드합니다.

```
$ cd /usr/local
$ mkdir /service
$ cd /service
$ wget https://downloads.tencentgoosefs.cn/goosefs/1.4.1/release/goosefs-1.4.1-bin.tar.gz
```

2. 다음 명령을 실행하여 설치 패키지의 압축을 해제하고 추출된 디렉터리를 입력합니다.

```
$ tar -zxvf goosefs-1.4.1-bin.tar.gz
$ cd goosefs-1.4.1
```

압축 해제 후 gooseFS goosefs-1.4.1의 홈 디렉터리가 생성됩니다. 본문에서는 `${GOOSEFS_HOME}` 을 이 홈 디렉터리의 절대 경로로 사용합니다.

3. `${GOOSEFS_HOME}/conf` 에 `conf/goosefs-site.properties` 구성 파일을 생성합니다. GooseFS는 AI 및 빅 데이터 시나리오를 위한 구성 템플릿을 제공하며 필요에 따라 적절한 템플릿을 선택할 수 있습니다. 그 다음

©2013-2022 Tencent Cloud. All rights reserved.

편집 모드로 들어가 구성을 수정합니다.
(1) AI 템플릿을 사용합니다. 자세한 내용은 AI 시나리오의 프로덕션 환경에 대한 GooseFS 구성 사례를 참고하십시오.

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

(2) 빅 데이터 템플릿을 사용합니다. 자세한 내용은 빅 데이터 시나리오의 프로덕션 환경에 대한 GooseFS 구성 사례를 참고하십시오.

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

4. 구성 파일 `conf/goosefs-site.properties` 에서 다음 구성 항목을 수정합니다.

```
# Common properties
# Master 노드의 host 정보 수정
goosefs.master.hostname=localhost
goosefs.master.mount.table.root.ufs=${goosefs.work.dir}/underFSStorage

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

# Worker properties
# worker 노드 구성을 수정하여 로컬 캐시 미디어, 캐시 경로 및 캐시 크기 지정
goosefs.worker.ramdisk.size=1GB
goosefs.worker.tieredstore.levels=1
goosefs.worker.tieredstore.level0.alias=SSD
goosefs.worker.tieredstore.level0.dirs.path=/data
goosefs.worker.tieredstore.level0.dirs.quota=80G


# User properties
# 파일 읽기 및 쓰기에 대한 캐시 정책 지정
goosefs.user.file.readtype.default=CACHE
goosefs.user.file.writetype.default=MUST_CACHE
```

주의 :
경로 매개변수 `goosefs.worker.tieredstore.level0.dirs.path` 를 구성하기 전에 먼저 경로를 생성해야 합니다.

©2013-2022 Tencent Cloud. All rights reserved.

GooseFS 실행

1. GooseFS를 시작하기 전에 GooseFS 디렉터리에 들어가 시작 명령을 실행해야 합니다.

```
$ cd /usr/local/service/goosefs-1.4.1
$ ./bin/goosefs-start.sh all
```

이 명령을 실행하면 다음 페이지를 볼 수 있습니다.

```
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]#
```

상기 명령 실행 후 `http://localhost:9201` 과 `http://localhost:9204` 에 액세스하면 Master와 Worker의 실행 상태를 각각 볼 수 있습니다.

COS(COSN) 또는 Tencent Cloud HDFS(CHDFS)를 GooseFS에 마운트

COS(COSN) 또는 Tencent Cloud HDFS(CHDFS)를 GooseFS의 루트 디렉터리에 마운트하려면 아래와 같이 `conf/core-site.xml` 에서 COSN/CHDFS(`fs.cosn.impl` , `fs.AbstractFileSystem.cosn.impl` , `fs.cosn.userinfo.secretId` 및 `fs.cosn.userinfo.secretKey` 를 포함하되 이에 국한되지 않음)의 필수 매개변수를 구성합니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

```
<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>
```

©2013-2022 Tencent Cloud. All rights reserved.

```
<value>1250000000</value>
</property>
```

설명 :

- COSN의 완전한 구성은 Hadoop 툴을 참고하십시오.
- CHDFS의 완전한 구성은 CHDFS 마운트를 참고하십시오.

다음으로 Namespace 생성을 통해 COS 혹은 CHDFS를 마운트 하는 방법과 절차를 소개합니다.

1. 네임스페이스 Namespace 하나를 생성하여 COS를 마운트합니다.

```
$ 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 \
```

주의 :

- COSN을 마운트한 namespace를 생성할 경우 반드시 `- -secret` 매개변수를 사용하여 액세스 키를 지정해야 하며, `--attribute` 를 사용하여 Hadoop-COS (COSN)의 모든 필수 매개변수를 지정해야 합니다. 구체적인 필수 매개변수는 Hadoop 툴을 참고하십시오.
- Namespace 생성 시 읽기/쓰기 정책 (rPolicy/wPolicy)을 지정하지 않았을 경우 구성 파일 중에 지정한 read/write type 혹은 기본값 (CACHE/CACHE_THROUGH)을 사용합니다.

동일한 방법으로 네임스페이스 namespace 하나를 생성하여 Tencent Cloud HDFS 마운트에 사용할 수 있습니다.

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

2. 생성 완료 후 `ls` 명령어를 통해 클러스터에 생성된 모든 namespace를 나열할 수 있습니다.

©2013-2022 Tencent Cloud. All rights reserved.

```
$ 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.c
om/3TB 03-11-2021 11:45:12:336 CACHE_THROUGH CACHE -1 DELETE
```

3. 아래 명령어를 실행하여 namespace의 세부 정보를 지정합니다.

```
$ goosefs ns stat myNamespace

NamespaceStatus{name=myNamespace, path=/myNamespace, ttlTime=-1, ttlAction=DELET
E, ufsPath=cosn://bucketName-125xxxxxx/3TB, creationTimeMs=1615434186076, lastMod
ificationTimeMs=1615436308143, lastAccessTimeMs=1615436308143, persistenceState=P
ERSISTED, mountPoint=true, mountId=4948824396519771065, acl=user::rwx,group::rwx,
other::rwx, defaultAcl=, owner=user1, group=user1, mode=511, writePolicy=CACHE_TH
ROUGH, readPolicy=CACHE}
```

메타데이터에서 기록한 정보는 아래 내용이 포함됩니다.

번호	매개변수	설명
1	name	namespace 이름
2	path	GooseFS에서 namespace의 경로
3	ttlTime	namespace 디렉터리와 파일의 ttl 주기
4	ttlAction	namespace 디렉터리와 파일의 ttl 처리 작업에는 FREE와 DELETE 두 가지 처리 작업이 있습니다. 기본값: FREE
5	ufsPath	ufs에서 namespace의 마운트 경로
6	creationTimeMs	namespace 생성 시간, 단위: 밀리초
7	lastModificationTimeMs	namespace 디렉터리와 파일의 최종 수정 시간, 단위: 밀리초
8	persistenceState	namespace의 지속 상태
9	mountPoint	namespace의 마운트 포인트가 하나인지에 대한 여부, 값은 항상 true
10	mountId	namespace 마운트 포인트 ID
11	acl	namespace의 액세스 제어 리스트
12	defaultAcl	namespace의 기본 액세스 제어 리스트

©2013-2022 Tencent Cloud. All rights reserved.

번호	매개변수	설명
13	owner	namespace의 owner
14	group	namespace의 owner가 소속된 group
15	mode	namespace의 POSIX 권한
16	writePolicy	namespace의 쓰기 정책
17	readPolicy	namespace의 읽기 정책

GooseFS를 통한 Table의 데이터 프리패치

1. GooseFS는 Hive Table의 데이터를 GooseFS에 프리패치할 수 있습니다. 프리패치 전에 관련 DB를 GooseFS에 연결해야 하며, 관련 명령어는 아래와 같습니다.

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

> 주의 :
> 명령어의 thrift는 실제 Hive Metastore 주소를 입력해야 합니다.

2. DB 추가 후 ls 명령어를 통해 현재 연결 중인 DB와 Table 정보를 확인할 수 있습니다.

```
$ 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,
```

©2013-2022 Tencent Cloud. All rights reserved.

```
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. load 명령어를 통해 Table의 데이터를 프리패치합니다.

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

Table의 데이터 프리패치는 비동기화 작업이므로 작업 ID 하나를 반환합니다. goosefs job stat <Job Id> 명령어를 통해 프리패치 작업의 진행률을 확인할 수 있습니다. "COMPLETED" 상태가 되면 전체 프리패치 과정이 완료되었음을 의미합니다.

GooseFS를 사용하여 파일 업로드와 다운로드 작업 실행

1. GooseFS는 대부분의 파일 시스템 작업 명령어를 지원하며, 다음 명령어를 통해 현재 지원하는 명령어 리스트를 확인할 수 있습니다.

```
$ goosefs fs
```

2. `ls` 명령어를 통해 GooseFS의 파일을 나열할 수 있습니다. 다음 예시에서 루트 디렉터리의 모든 파일을 나열하는 방법을 확인할 수 있습니다.

```
$ goosefs fs ls /
```

3. `copyFromLocal` 명령어를 통해 데이터를 로컬에서 GooseFS로 복사할 수 있습니다.

©2013-2022 Tencent Cloud. All rights reserved.

```
$ 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% /LICE
NSE
```

4. `cat` 명령어를 통해 파일 내용을 확인할 수 있습니다.

```
$ 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. GooseFS는 로컬 디스크를 하부 파일 시스템으로 사용하는 것으로 기본 설정되어 있으며, 기본 파일 시스템 경로는 `./underFSStorage` 입니다. `persist` 명령어를 통해 파일을 로컬 파일 시스템에 영속 저장할 수 있습니다.

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

GooseFS를 사용하여 신속하게 파일 업로드/다운로드

1. CFS 상태를 검사하여 파일이 캐시 되었는지 확인합니다. 파일 상태가 `PERSISTED` 인 경우 파일이 메모리에 저장된 것을 의미하며, 파일 상태가 `NOT_PERSISTED` 인 경우 파일이 메모리에 없음을 의미합니다.

```
$ 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/c
os/sample_tweets_150m.csv
```

2. 파일에 단어 'tencent'가 몇 개 있는지 통계를 낸 후 작업 소요 시간을 계산합니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

3. 해당 데이터를 메모리에 캐시하여 조회 속도를 효과적으로 향상 시킬 수 있습니다. 자세한 예시는 아래와 같습니다.

```
$ 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
```

시스템 프로세스 딜레이가 1.181s에서 0.243s로 감소되어 10배 향상된 것을 알 수 있습니다.

GooseFS 비활성화

아래 명령어를 통해 GooseFS를 비활성화할 수 있습니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

주요 특징
캐시 능력

최종 업데이트 날짜: : 2023-01-29 15:19:20

캐시 능력 개요

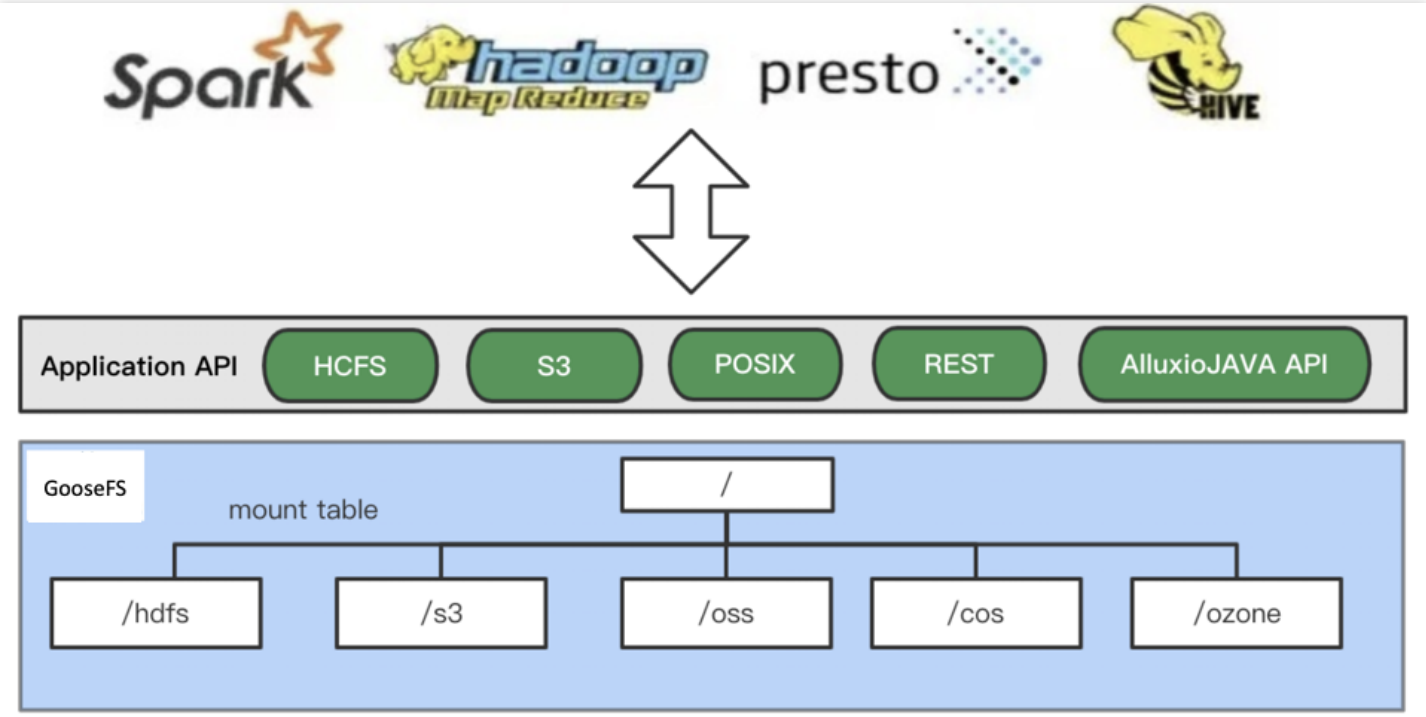
GooseFS는 오픈 소스 Alluxio 아키텍처에 기반한 강력한 분산형 Cache 능력을 제공하여 사용자가 다양한 플랫폼의 사용자 데이터를 통합하는 동시에, 사용자의 전반적인 I/O 처리량을 향상할 수 있도록 돕습니다.

주로 아래 두 가지 방법을 사용합니다.

- GooseFS NameSpace 원격 스토리지: NS 스토리지 공간 기반은 외부 스토리지로, COSN 및 HDFS를 거쳐 하나 이상의 여러 NS를 동일한 인프라 스토리지에 연결하여, 대량의 데이터를 보존할 수 있습니다. COSN 스토리지 및 컴퓨팅 분리 솔루션을 사용하면, GooseFS NameSpace는 COSN을 통해 탄력적인 확장과 고가용성 기능을 갖춘 빅 데이터 스토리지를 제공할 수 있습니다.
- GooseFS 로컬 Cache 스토리지: GooseFS는 master와 worker가 제공하는 분산형 Cache 기능을 통해, 애플리케이션 레이어에서 생성된 데이터를 애플리케이션 노드의 로컬 메모리와 디스크에 보존하며, 주로 핫 데이터와 임시 데이터를 저장합니다. 각 로컬 스토리지 노드는 사용자가 구성할 수 있으며 NameSpace에 연결된 원격 영속 레이어도 동일한 client 서비스 사용자의 읽기/쓰기 작업을 경유하게 됩니다.

사용자는 Cache Policy 작업을 할 수 있습니다. 읽기/쓰기 모드에서 로컬 Cache 스토리지와 NameSpace 원격 스토리지 사용을 결정합니다.

©2013-2022 Tencent Cloud. All rights reserved.



GooseFS는 동일한 로컬 Cache 및 원격 NameSpace 통합 기능을 제공하여 사용자가 스토리지 분리의 통합 방안을 완전히 구현하는 동시에 애플리케이션 노드에 대한 데이터 친연성를 설정할 수 있도록 지원합니다.

GooseFS 캐시 설정

사용자는 goosefs-site.properties 파일에 들어가 GooseFS 캐시 설정 상황을 볼 수 있습니다. 현재 GooseFS의 캐시 설정은 캐시 레이어, 캐시 만료 정책 등 방면에서 이해할 수 있으며, 캐시 레이어 방면에는 주로 단일 레이어 스토리지와 멀티 레이어 스토리지 두 가지가 있습니다. 캐시 만료 정책은 주로 LRU와 LRFU의 두 가지 유형으로 나뉩니다. 자세한 소개는 아래와 같습니다.

1. 캐시 레이어

GooseFS는 단일 레이어 스토리지와 멀티 레이어 두 가지 캐시 레벨을 제공합니다. 단일 레이어 스토리지에는 하나의 스토리지만 사용하고, 멀티 레이어 스토리지에는 다양한 스토리지들이 사용되며, 업무 부하 상황에 따라 어떤 스토리지 미디어를 사용할지 결정하여 맞춤형 I/O 성능을 제공할 수 있습니다. GooseFS 기본 설정은 단일 스토리지로 되어 있습니다. 멀티 레이어 시나리오에서 캐시 만료는 많은 성능 비용을 가져오므로 **단일 레이어 스토리지**를 사용하는 것을 추천합니다. I/O 성능 수준에 따라 일반적으로 MEM, SSD, HDD를 스토리지 미디어로 선택할 수 있습니다.

goosefs-site.properties 설정 파일의 levels 매개변수를 수정하여 캐시 레벨을 수정할 수 있으며, 입력 매개변수가 1이면 단일 레이어 스토리지만 사용하고 입력 매개변수가 3이면 3개 레이어 스토리지를 사용합니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

goosefs-site.properties 구성 파일의 alias 매개변수를 수정하여 캐시 레이어에 해당하는 스토리지 미디어를 수정할 수 있습니다.

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

> 주의 :
> level{x}은 스토리지 레이어 레벨을 나타냅니다. 예를 들어 level 0은 단일 레이어 스토리지를 나타냅니다.

1.1 단일 레이어 스토리지

단일 레이어 스토리지 모드에서 자주 사용하는 시스템 구성 항목은 다음과 같습니다:

```
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
```

다음은 각 구성 항목 내용에 대한 설명입니다.

- ramdisk.size: 이 구성 항목은 worker 노드가 차지하는 메모리 크기를 지정하는 데 사용되며 ramdisk의 용량은 memory보다 작아야 합니다. 설정 후 GooseFS는 각 worker 노드에 지정된 크기의 메모리 공간을 할당하고 시스템의 전체 메모리를 사용합니다.
- memory.size: 이 구성 항목은 GooseFS 시스템의 총 메모리를 지정하는 데 사용됩니다. 설정 후 GooseFS는 지정된 크기의 스토리지 미디어를 자동으로 점용합니다. 실제 물리적 스토리지 공간은 memory 값보다 커야 합니다.
- dirs.path: 이 구성 항목은 디렉터리를 지정하는 데 사용됩니다. GooseFS는 지정된 디렉터리에 대한 스토리지 미디어를 할당합니다. dirs.mediumtype과 함께 사용해야 합니다. 위의 예시와 같이 /data/GooseFSWorker를 MEM 스토리지 미디어에 마운트하고, /data4/GooseFSWorker를 SSD 스토리지 미디어에 마운트합니다. 각 디렉토리의 순서는 스토리지 미디어의 순서와 각각 일치해야 한다는 점에 유의해야 합니다.
- dirs.mediumtype: 이 구성 항목은 스토리지 미디어를 지정하는 데 사용됩니다. GooseFS는 지정된 디렉토리에 대해 스토리지 미디어를 할당합니다. dirs.path와 함께 사용해야 합니다. 사용 가능한 기본 스토리지 미디어는 MEM과 SSD가 있으며, HDD와 같은 다른 스토리지 미디어를 마운트한 경우 필요에 따라 구성을 수정할 수 있습니다.
- dirs.quota: 구성 항목을 변경하여 각 디렉터리의 사전 설정 공간을 지정합니다. 설정 후 GooseFS를 통해 지정 디렉터리에 대해 공간을 할당하고, 디렉터리 순서와 각각 대응되어야 합니다. 위의 예시는 /data/GooseFSWorker, /data1/GooseFSWorker, /data2/GooseFSWorker와 같은 디렉토리에 16GB MEM 공간을 할당하고 /data3/GooseFSWorker, /data4/GooseFSWorker 디렉터리에 100GB SSD 공간을 할당해야 함을 보여줍니다.

©2013-2022 Tencent Cloud. All rights reserved.

1.2 티어링 스토리지

티어링 스토리지 모드에서는 데이터 블록의 읽기 및 쓰기 모드와 단일 레이어 스토리지의 읽기 및 쓰기 모드 간에 차이가 있습니다. 단일 레이어 스토리지 모드에서 데이터는 스토리지 미디어에서 직접 읽고 쓰게 됩니다. 멀티 레이어 스토리지 모드에서 데이터는 기본적으로 최상위 스토리지 미디어에 먼저 기록됩니다. 읽을 때 데이터를 먼저 최상위 스토리지 미디어로 이동해야 합니다. 구체적인 내용은 다음과 같습니다.

- **데이터 쓰기**: 새 데이터 블록은 기본적으로 최상위 레이어 스토리지 미디어에 기록되며, 최상위 레이어 스토리지 미디어의 저장 공간이 가득 차면 GooseFS에서 다음 레이어 스토리지 미디어에 데이터를 순차적으로 저장합니다. 모든 스토리지 미디어에 데이터가 가득 차면 사용자가 지정한 캐시 만료 정책에 따라 GooseFS가 만료된 데이터를 제거합니다. 만료된 데이터를 제거할 수 없고 사용 가능한 저장 공간이 모두 차면 데이터를 쓸 수 없습니다.
- **데이터 읽기**: 멀티 레이어 스토리지 모드에서 콜드 데이터는 하위 수준 레이어 스토리지 미디어에 투명하게 덤프되고 데이터를 읽게되면 핫 데이터로 바뀌어 상위 수준 스토리지에 배치됩니다.

> 주의 :
>
> - 설정된 만료 정책에 의해, GooseFS는 지정된 릴리스 공간에 따라 일정량의 데이터를 제거합니다. 이 매개변수는 goosefs.worker.tieredstore.free.ahead.bytes로 지정할 수 있으며 기본값은 0입니다.
> - 티어링 스토리지 모드에서 데이터를 읽을 때 하위 레이어 스토리지 미디어에서 상위 레이어 스토리지 미디어로 데이터가 자주 덤프될 수 있으며, 이는 빈번한 캐시 제거 및 특정 성능 손실로 이어질 수 있습니다. 일반적으로 **단일 레이어 스토리지 사용을 권장합니다**.

티어링 스토리지 모드에서, 자주 사용하는 시스템 구성 항목은 다음과 같습니다.

```
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
```

이 중 x는 캐시 레이어를 의미하며 일반적으로 MEM, SSD, HDD 등 스토리지 미디어에 상응하는 3개 레이어로 설정할 수 있습니다. 2개 레이어는 스토리지 레이어가 각각 MEM, SSD이며, 레이어당 100GB 스토리지를 할당하는 것을 예로 들면, 이에 대한 설정 정보는 다음과 같습니다.

```
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
```

©2013-2022 Tencent Cloud. All rights reserved.

```
goosefs.worker.tieredstore.level1.dirs.mediumtype=SSD
goosefs.worker.tieredstore.level1.dirs.quota=100GB
```

GooseFS는 멀티 레이어 스토리지를 지원하며 레이어 수에 제한이 없습니다. 하지만 각 레이어의 이름(alias)의 유일성은 보장되어야 합니다. 일반적으로 3개 레이어 캐시를 제공하며 각 레이어가 각각 MEM, SSD, HDD에 대응되어 티어링 효과가 좋습니다.

2. 캐시 만료 정책

GooseFS는 두 가지 캐시 만료 정책을 제공합니다.

- LRUAnnotator：최근 사용 횟수가 가장 적은(least-recently-used) 순서대로 캐시를 제거합니다. 이는 GooseFS의 기본 설정 정책입니다.
- LRFUAnnotator：최근 사용 횟수가 가장 적은(least-recently-used) 순서 및 최근 사용 빈도가 가장 적은(least-frequently-used) 순서대로 캐시를 삭제합니다. 가중치 설정 goosefs.worker.block.annotator.lrfu.step.factor과 goosefs.worker.block.annotator.lrfu.attenuation.factor 를 통해 두 가지 정책의 삭제 과정 중의 작용을 조정합니다.
- 만약 least-recently-used 의 가중치를 최대로 조정하면, 해당 정책의 결과는 LRUAnnotator와 같게 됩니다.

goosefs.worker.block.annotator.class 설정 항목에서 캐시 만료 정책을 설정할 수 있습니다. 설정 시 정책 이름을 올바르게 설정해야 합니다.

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

데이터 라이프사이클 관리

이 라이프사이클는 GooseFS 중의 데이터 라이프사이클을 지칭하며, 원격 스토리지 시스템 UFS가 아닙니다. 라이프사이클 관리는 주로 아래 4가지 작업이 있습니다.

- free: 이 작업은 GooseFS에서 해당 디렉터리나 파일의 데이터를 삭제합니다. (UFS 중의 상응 데이터는 삭제하지 않습니다). 이 작업은 주로 GooseFS의 캐시 공간을 릴리스하여 더 자주 사용되는 데이터를 사용할 수 있도록 합니다.
- load：이 작업은 UFS에 상응하는 디렉터리 또는 파일의 데이터를 GooseFS에 로딩하는데 사용되며, 이러한 작업은 콜드 데이터의 핫 데이터로의 전환 및 I/O 성능 향상에 사용됩니다.
- persist：이 작업은 GooseFS 내의 데이터를 UFS 스토리지에 쓰는 데에 사용됩니다. 이러한 작업은 데이터 영속화 및 데이터 손실 방지에 사용됩니다.
- TTL(Time to Live): TTL 속성은 주로 GooseFS에서의 디렉터리나 파일의 라이프사이클을 설정하는 데 사용되며, 수명 주기가 초과되면 GooseFS 및 UFS에서 삭제됩니다. 설정을 통해 GooseFs 데이터만 삭제하거나 디스크 공간만 릴리스할 수 있습니다.

©2013-2022 Tencent Cloud. All rights reserved.

1. GooseFS에서 데이터 릴리스

free 명령을 통해 GooseFS에서 데이터를 릴리스 합니다. 다음 예시는 데이터 릴리스 후, GooseFS의 데이터 상태가 0%로 바뀌는 것을 나타냅니다.

```
$ 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
```

예시 중의 /data/test.txt는 임의의 유효한 GooseFS파일 경로일 수 있습니다. 만약 데이터를 원격 스토리지 UFS에 저장할 경우, load 명령을 통해 재로딩할 수 있습니다.

> 주의 :
> 일반적으로, 캐시 정책 설정을 통해 GooseFS 중의 과거 데이터를 자동 제거할 것을 권장합니다.

2. GooseFS로 데이터 로딩

load 명령을 통해 GooseFS에서 데이터를 로딩할 수 있습니다. 아래 예시는 데이터 릴리스 후, GooseFS 중의 데이터 상태가 100%로 바뀌는 것을 나타냅니다.

```
$ 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
```

예제의 /data/test.txt는 모든 유효한 GooseFS 파일 경로일 수 있습니다. 로컬 파일 시스템에서 데이터를 로딩하는 경우 copyFromLocal 명령을 사용해야 합니다. 로컬 파일 시스템에서 데이터를 로드하는 작업은 데이터를 원격 스토리지 UFS에 보존하지 않습니다. 기본적으로 GooseFS는 파일에 처음 액세스할 때 데이터를 GooseFS에 자동으로 캐시에 로딩하므로 일반적으로 이 명령을 사용하여 데이터를 로딩할 필요는 없지만 비즈니스에서 미리 캐시에 데이터를 로딩해야 하는 경우 , 이 명령을 통해 로딩할 수 있습니다.

3. GooseFS에서의 데이터 영속화

persist 명령을 통해 데이터를 원격 스토리지 시스템 UFS에 영속화할 수 있습니다.

```
$ 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
```

예시 중의 data/test.txt 는 임의의 유효한 GooseFS 파일 경로일 수 있습니다. 캐시 정책 설정을 통해 영속화 작업을 자동 실행할 것을 권장합니다.

©2013-2022 Tencent Cloud. All rights reserved.

4. 데이터 TTL 속성 설정

GooseFS는 네임스페이스의 모든 파일 또는 디렉터리에 대한 TTL 속성 설정을 지원합니다. 이 속성은 비즈니스 데이터가 지정된 기간에 따라 주기적으로 삭제되도록 하여 새 파일을 위한 공간을 확보하고 로컬 디스크 공간을 효과적으로 사용할 수 있도록 합니다. GooseFS 파일 및 디렉터리의 TTL 속성은 파일 메타데이터의 일부로서, 클러스터가 재시작된 후에도 TTL 속성이 여전히 유효함을 보장할 수 있으며, 백엔드 스레드의 정기 점검 프로그램은 이러한 TTL 속성을 확인하고 자동으로 만료된 파일을 삭제합니다.

정기 점검 프로그램의 점검 주기 및 점검 유형은 goosefs-site.preperties에서 설정할 수 있으며 다음 예시에서는 점검 주기를 10분으로, 점검 유형을 FREE로 설정합니다.

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

설정이 완료되면 GooseFS 백그라운드 스레드가 10분마다 점검하고 현재 검사 주기에서 발견된 만료된 파일은 다음 검사 주기 후에 캐시 리소스를 릴리스합니다. 예를 들어 00:00:00에 첫 번째 점검 중에 만료된 파일 배치가 발견되면 이 파일 배치의 캐시는 00:10:00에 지워집니다.

> 주의 :
> 기본적으로 입력 매개변수 단위는 밀리초(ms)이며, 매개변수를 입력할 때 감지 시간 주기의 단위를 초(s), 분(m), 시(h) 등과 같이 지정할 수 있습니다. 자세한 설명은 설정 설명서를 참고하십시오.

데이터 복사 관리

데이터 복사는 수동과 능동 두 가지 형식으로 나눠집니다.

1. 수동 레플리케이션

GooseFS의 각 파일에는 클러스터에 분산 저장된 하나 이상의 스토리지 블록이 포함되어 있으며 기본적으로 GooseFS는 부하 및 용량에 따라 데이터 블록의 복사 수준을 자동으로 조정할 수 있습니다. 수동 레플리케이션은 아래 시나리오에서 발생할 수 있습니다.

- 여러 클라이언트가 같은 블록을 동시에 읽게 되면 여러 block이 다른 worker에 동시에 존재하게 됩니다.
- '로컬 읽기 우선'가 활성화 된 후, 데이터가 로컬에 없는 경우, remote read가 발송되고, 로컬 worker에 저장됩니다.
- 수동 레플리케이션으로 발생된 복제본의 수가 설정된 복제본 수보다 클 경우, 비동기적으로 초과된 복제본을 삭제합니다. 위의 과정은 사용자에게 투명하게 공개됩니다.

2. 능동 레플리케이션

©2013-2022 Tencent Cloud. All rights reserved.

능동 레플리케이션은 파일의 복제본 수 설정을 통해 구현되며, 설정한 복제본 수에 맞지 않는 block은 비동기적으로 보완하고, 설정한 복제본 수를 초과하는 block은 초과된 복제본을 삭제합니다. 아래 명령을 통해 능동 레플리케이션 레벨을 조정할 수 있습니다.

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

매개변수 설정 설명

- max: 지정 파일의 최대 복제본 수로, 기본값은 -1입니다. 상한선을 설정하지 않습니다. 0으로 설정하면 지정 파일의 콜드 데이터를 GooseFS에 저장하지 않습니다. 일반적으로 자연수로 설정합니다. 설정 후 GooseFS는 파일의 복제본 수를 체크하고 잔여 복제본을 삭제합니다.
- min: 지정 파일의 최소 복제본 수로, 기본값은 0입니다. 이는 GooseFS가 데이터 콜드화 후 해당 데이터를 삭제하고 어떤 복제본도 남기지 않음을 나타냅니다. 일반적으로 자연수로 설정하며, **max값보다 작아야 합니다**. 설정 후 GooseFS는 파일 복제본 수를 검사하고 최소값보다 작으면 복제본 수를 자동으로 채웁니다.
- path: 지정된 파일 이름으로, 디렉토리 또는 특정 파일 경로가 될 수 있습니다.
- R: path의 입력 매개변수가 디렉터리인 경우 -R을 지정하면 지정된 최소 복제본 수와 최대 복제본 수에 따라 지정된 디렉터리의 모든 파일 및 하위 디렉터리를 중복 재귀 복제합니다.

지정된 파일의 복사 상태를 확인해야 하는 경우 stat 명령을 통해 볼 수 있습니다. 다음 예제는 /data/test.txt 파일의 최대 복제본 수가 -1임을 보여줍니다. 이는 해당 파일이 콜드 데이터로 변환된 후, 만료 및 삭제됨을 나타냅니다.

```
$ goosefs fs stat /data/test.txt
/data/test.txt is a file path.
FileInfo{fileId=50331647, fileIdentifier=null, name=test.txt, path=/data/test.tx
t, ufsPath=hdfs://172.16.16.16:4007/data/test.txt, length=0, blockSizeBytes=13421
7728, creationTimeMs=1618193473555, completed=true, folder=false, pinned=false, p
innedlocation=[], cacheable=true, persisted=true, blockIds=[], inMemoryPercentage
=100, lastModificationTimesMs=1616763603692, ttl=-1, lastAccessTimesMs=1616763603
692, 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) , ac
l=user::rw-,group::r--,other::r--, defaultAcl=}
This file does not contain any blocks.
```

캐시 사용량 확인

GooseFS는 로컬 캐시 용량과 사용 상황을 기록하며, 다음 명령을 통해 GooseFS의 실행 상황을 확인하여 로컬 캐시에 대해 맞춤형 관리 및 점검을 진행할 수 있습니다

- GooseFS 사용 중 캐시 확인. 단위: 바이트.

©2013-2022 Tencent Cloud. All rights reserved.

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

- GooseFS 총 캐시 용량 확인. 단위: 바이트.

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

- GooseFS의 캐시 사용 리포트 조회:

```
$ 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
```

©2013-2022 Tencent Cloud. All rights reserved.

투명 가속 기능

최종 업데이트 날짜: : 2021-11-01 14:28:34

개요

투명한 가속화 기능은 CosN의 COS 액세스 기능을 가속화하는데 사용합니다. CosN 툴은 Tencent Cloud Cloud Object Storage(COS)이 제공하는 표준 Hadoop 파일 시스템을 기반으로 구현되었으며, Hadoop, Spark 및 Tez 등 빅 데이터 컴퓨팅 프레임워크와 COS를 통합하는 기능을 지원합니다. 사용자는 Hadoop 파일 시스템 인터페이스를 구현한 CosN 플러그인을 사용하여, COS에 저장된 데이터를 읽기/쓰기할 수 있습니다. CosN 툴을 사용하여 COS에 액세스하는 기존 사용자를 위해, GooseFS는 클라이언트 경로 매핑 방법을 제공하여 사용자가 현재 Hive table 정의를 수정하지 않고도 CosN scheme을 사용하여 GooseFS에 액세스할 수 있도록 합니다. 해당 특성은 사용자로 하여금 기존 테이블 정의를 수정하지 않는 전제 하에 편리하게 GooseFS의 기능과 성능을 비교 테스트 할 수 있도록 합니다. 클라우드 HDFS 사용자(CHDFS)는, 설정 변경을 통해 OFS Scheme를 사용하여 GooseFS에 액세스하는 목적을 달성할 수 있습니다.

CosN Schema와 GooseFS Schema의 매핑 설명은 아래와 같습니다.

Namespace warehouse 에 상응하는 UFS 경로가 'cosn://examplebucket-1250000000/data/warehouse/'인 경우, CosN에서 GooseFS의 경로 매핑 관계는 다음과 같습니다.

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

GooseFS에서 CosN의 경로 매핑 관계는 다음과 같습니다.

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

CosN Scheme은 GooseFS 기능에 액세스하여, 클라이언트 측에서 GooseFS 경로와 기본 파일 시스템의 CosN 경로 간의 매핑 관계를 유지하고, CosN 경로의 요청을 GooseFS의 요청으로 변환합니다. 매핑 관계는 주기적으로 갱신되며, GooseFS 설정 파일 goosefs-site.properties 중의 설정 항목 goosefs.user.client.namespace.refresh.interval을 수정하여 갱신 주기를 조정할 수 있으며, 기본값은 60초입니다.

> 주의 :
> 액세스한 CosN 경로를 GooseFS 경로로 변환할 수 없는 경우, 해당 Hadoop API 호출에서 이상 경고가 발생합니다.

©2013-2022 Tencent Cloud. All rights reserved.

작업 예시

해당 예시에서는 Hadoop 명령 라인 및 Hive에서 gfs://, cosn://, ofs://의 3가지 Schema를 사용하여 GooseFS에 액세스하는 방법을 보여줍니다. 작업 과정은 다음과 같습니다.

1. 데이터 및 컴퓨팅 클러스터 준비

- 버킷 생성 문서를 참고하여 테스트용 버킷을 생성 합니다.
- 폴더 생성 문서를 참고하여 버킷의 루트 경로 아래에 ml-100k라는 폴더를 생성합니다.
- Grouplens에서 ml-100k 데이터 세트를 다운로드하고 u.user 파일을 '<버킷 루트 경로>/ml-100k'에 업로드합니다.
- EMR 가이드 문서를 참고하여 하나의 EMR 클러스터를 구매하고 HIVE 컴포넌트를 설정하십시오.

2. 환경 설정

i. GooseFS 클라이언트측 jar 패키지(goosefs-1.0.0-client.jar)를 share/hadoop/common/lib/ 디렉터리에 넣습니다.

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

> 주의
> 설정 변경 및 jar 패키지 추가는 클러스터의 모든 노드에 동기화되어야 합니다.

ii. Hadoop 구성 파일 etc/ hadoop/ core-site.xml을 수정합니다. GooseFS의 구현 클래스를 지정합니다.

```
<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. 다음 Hadoop 명령어를 실행하여 gfs:// Scheme을 통해 GooseFS에 액세스할 수 있는지 확인합니다.
&dxlt;MASTER_IP>는 Master 노드의 IP입니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

iv. Hive가 GooseFS Client 패키지에 로딩될 수 있도록, GooseFS의 클라이언트측 jar 패키지를 Hive의 auxlib 디렉터리에 넣습니다.

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

v. 다음 명령을 실행하여 UFS Scheme이 CosN인 Namespace를 생성하고 Namespace를 나열합니다. 이 명령어의 examplebucket-1250000000을 사용자의 COS 버킷으로 바꾸고 SecretId와 SecretKey를 사용자의 키 정보로 바꿀 수 있습니다.

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

vi. 명령을 실행하여 UFS Scheme이 OFS인 Namespace를 생성하고 Namespace를 나열합니다. 이 명령의 instance-id를 사용자의 CHDFS 인스턴스로 바꾸고 1250000000을 사용자의 APPID로 바꿀 수 있습니다.

```
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. GooseFS Schema 테이블 생성 및 데이터 조회

아래 명령을 통해 실행합니다.

```
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. CosN Schema 테이블 생성 및 데이터 조회

아래 명령을 통해 실행합니다.

©2013-2022 Tencent Cloud. All rights reserved.

```
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. CosN을 수정하여 GooseFS 호환 구현

hadoop/etc/hadoop/core-site.xml 수정

```
<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>
```

Hadoop 명령어를 실행합니다. 경로를 GooseFS의 경로로 변환할 수 없는 경우 명령 출력에 오류 메시지가 포함됩니다.

```
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-125000000
0/ml-100k/u.user
hadoop fs -ls cosn://examplebucket-1250000000/unknow-path
ls: Failed to convert ufs path cosn://examplebucket-1250000000/unknow-path to Goo
seFs path, check if namespace mounted
```

Hive를 재실행하여 구문 조회.

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

6. OFS Schema 테이블 생성 및 데이터 조회

아래 명령을 통해 실행합니다.

©2013-2022 Tencent Cloud. All rights reserved.

```
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. OFS 구현을 GooseFS의 호환 구현으로 수정

hadoop/etc/hadoop/core-site.xml 수정

```
<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>
```

Hadoop 명령어를 실행합니다. 경로를 GooseFS의 경로로 변환할 수 없는 경우 출력 결과에 오류 메시지가 포함됩니다.

```
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-guan
gzhou.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
```

Hive를 재실행하여 구문 조회.

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

©2013-2022 Tencent Cloud. All rights reserved.

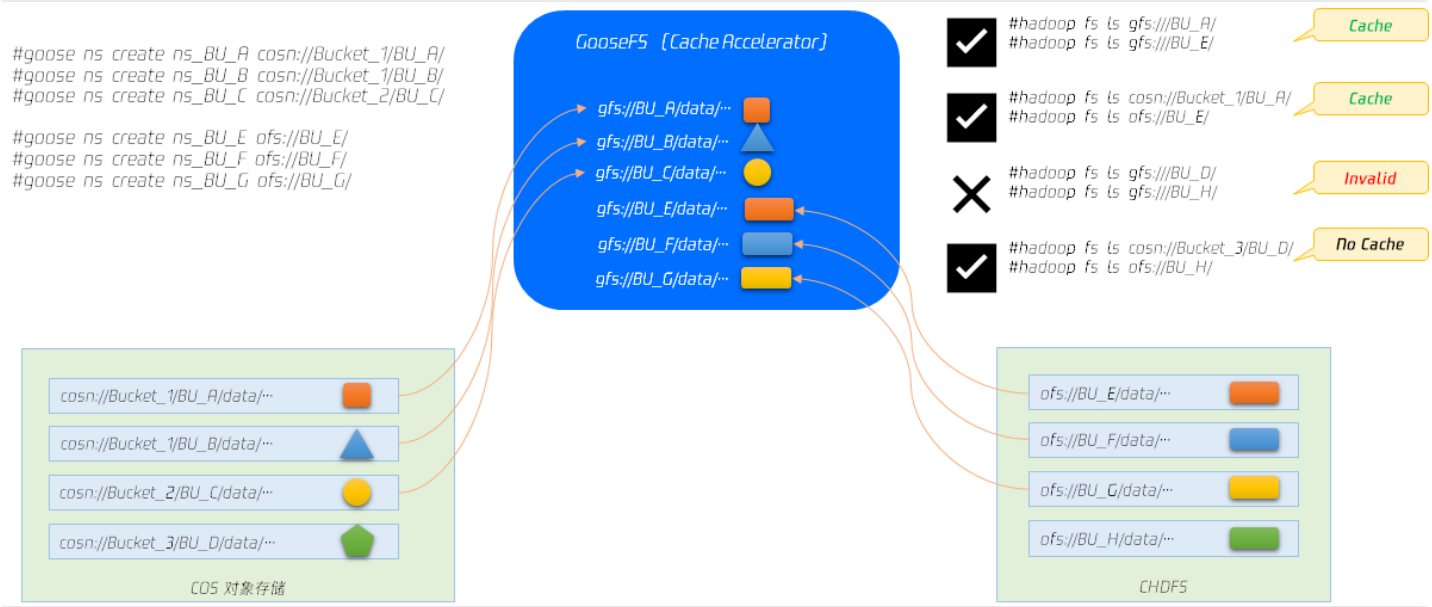
통합 네임스페이스 기능

최종 업데이트 날짜: : 2023-03-14 17:01:39

통합 네임스페이스 기능 개요

GooseFS 통합 네임스페이스(NameSpace) 기능은 투명한 네이밍 메커니즘을 통해 다양한 기본 스토리지 시스템 액세스 시맨틱을 통합하여 사용자에게 데이터 관리에 대한 통합된 인터랙티브 뷰를 제공합니다.

GooseFS는 이 기능을 활용하여 로컬 파일 시스템, Tencent Cloud Object Storage(COS), Tencent Cloud HDFS(CHDFS)와 같은 다양한 기본 스토리지 시스템과 연결 및 통신하며, 상위 계층 비즈니스를 위한 통합 액세스 API 및 파일 프로토콜을 제공합니다. 이러한 방식으로 비즈니스 측은 다양한 기본 스토리지 시스템에 저장된 데이터에 액세스하기 위해 GooseFS에서 제공하는 액세스 API만 호출하면 됩니다.



상기 이미지는 통합 네임스페이스의 작동 원리를 보여줍니다. GooseFS의 네임스페이스 생성 명령인 create ns를 사용하여 COS 및 CHDFS의 지정된 파일 디렉터리를 GooseFS에 마운트한 다음 gfs:// 통합 schema를 사용하여 데이터에 액세스할 수 있습니다. 자세한 설명은 다음과 같습니다.

- COS는 총 3개의 버킷(bucket-1, bucket-2, bucket-3)이 있으며 Bucket-1에는 BU_A와 BU_B라는 2개의 디렉터리가 있습니다. bucket-1, bucket-2는 모두 GooseFS에 마운트되어 있습니다.
- Cloud HDFS에는 BU_E, BU_F, BU_G, BU_H의 4개의 디렉터리가 있으며 BU_H를 제외한 모든 디렉터리는 GooseFS에 마운트됩니다.
- GooseFS의 파일 작업에서 두 디렉터리 BU_A, BU_E에 gfs://의 통합 schema로 액세스하면 정상적으로 액세스할 수 있으며, 파일은 GooseFS의 로컬 파일 시스템에 캐싱됩니다.

©2013-2022 Tencent Cloud. All rights reserved.

- 기본 파일 시스템(COS, Cloud HDFS)에 저장된 2개의 디렉터리 BU_A와 BU_E는 GooseFS에 마운트되어 있으며, 파일이 GooseFS에 캐싱되어 있다면 gfs://의 통합 schema로 액세스할 수 있습니다(예: hadoop fs ls gfs://BU_A). 각 원격 파일 시스템의 네임스페이스를 통해 액세스할 수도 있습니다(예: hadoop fs ls cosn://bucket-1/BU_A).
- 파일이 GooseFS에 캐시되지 않은 경우, 파일이 GooseFS의 로컬 파일 시스템에 캐시되지 않기 때문에 통합 스키마 gfs://를 통해 액세스할 수 없지만 여전히 기본 스토리지 시스템의 네임스페이스를 통해 액세스할 수 있습니다.

통합 네임스페이스 기능 사용

ns 작업을 통해 GooseFS에 네임스페이스를 생성하고, 기본 스토리지 시스템을 GooseFS에 매핑할 수 있습니다. 현재 지원되는 기본 스토리지 시스템에는 COS, CHDFS, 로컬 HDFS 등이 있습니다. 네임스페이스 생성 작업은 Linux 파일 시스템에서 파일 볼륨을 디스크를 마운트하는 절차와 유사합니다. 생성된 네임스페이스를 사용하여 GooseFS는 통일적인 액세스 의미 체계를 가진 파일 시스템을 클라이언트에 제공할 수 있습니다. GooseFS 네임스페이스에 대한 현재 작동 지침 세트는 다음과 같습니다.

```
$ goosefs ns
Usage: goosefs ns [generic options]
[create <namespace> <CosN/Chdfs path> <--wPolicy <1-6>> <--rPolicy <1-5>> [--read
only] [--shared] [--secret fs.cosn.userinfo.secretId=<AKIDxxxxxxx>] [--secret fs.
cosn.userinfo.secretKey=<xxxxxxxxxx>] [--attribute fs.ofs.userinfo.appid=12000000
00][--attribute fs.cosn.bucket.region=<ap-xxx>/fs.cosn.bucket.endpoint_suffix=<co
s.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>]
```

주의 :

- 비즈니스 보안 향상을 위해 영구 키 대신 서브 계정 키 또는 임시 키를 사용하는 것이 좋습니다. 서브 계정 승인 시 최소 권한의 원칙 설명을 준수하여 예상치 못한 데이터 유출을 방지하시기 바랍니다.
- 영구 키를 사용해야 하는 경우 사용 보안을 향상시키기 위해 최소 권한의 원칙 설명에 따라 허용된 작업, 리소스 범위 및 액세스 IP와 같은 조건을 포함한 권한 범위를 영구 키에 제한하는 것이 좋습니다.

상기 명령어 집합의 각 명령어의 기능 설명은 아래와 같습니다.

©2013-2022 Tencent Cloud. All rights reserved.

명령	설명
create	네임스페이스를 생성하고 원격 스토리지 시스템 UFS를 네임스페이스에 매핑하는 데 사용됩니다. 네임스페이스를 생성할 때 캐시 읽기 및 쓰기 정책 설정을 지원합니다. 인증된 키 정보(secretId, secretKey)를 전달해야 합니다.
delete	지정한 네임스페이스를 삭제하는 데 사용됩니다.
ls	마운트 포인트, UFS 경로, 생성 시간, 캐시 정책, TTL 정보 등 지정된 네임스페이스의 세부 정보를 나열하는 데 사용됩니다.
setPolicy	지정한 네임스페이스의 캐시 정책을 설정하는 데 사용됩니다.
setTtl	지정한 네임스페이스의 TTL를 설정하는 데 사용됩니다.
stat	마운트 포인트, UFS 경로, 생성 시간, 캐시 정책, TTL 정보, 영속화 상태, 사용자 그룹, ACL, 마지막 액세스 시간, 수정 시간 등과 같은 지정된 네임스페이스에 대한 설명 정보 제공에 사용됩니다.
unsetPolicy	지정한 네임스페이스의 캐시 정책을 재설정하는 데 사용됩니다.
unsetTtl	지정한 네임스페이스의 TTL를 재설정하는 데 사용됩니다.

네임스페이스 생성 및 삭제

GooseFS를 통해 네임스페이스 작업을 생성하면 자주 액세스하는 핫 데이터를 원격 스토리지 시스템에서 로컬 고성능 스토리지 노드로 캐시할 수 있어, 로컬 컴퓨팅 서비스에 고성능 데이터 액세스 기능을 제공합니다. 다음은 COS의 example-bucket 버킷, 버킷의 example-prefix 디렉터리 및 Cloud HDFS 파일 시스템이 각각 test_cos, test_cos_prefix, test_chdfs 등 네임스페이스에 매핑됨을 보여줍니다.

```
# COS 버킷 example-bucket을 test_cos 네임스페이스에 매핑
$ goosefs ns create test_cos cosn://example-bucket-1250000000/ --wPolicy 1 --rPol
icy 1 --secret fs.cosn.userinfo.secretId=AKIDxxxxxxx --secret fs.cosn.userinfo.se
cretKey=xxxxxxxxxx --attribute fs.cosn.bucket.region=ap-guangzhou --attribute fs.
cosn.bucket.endpoint_suffix=cos.ap-guangzhou.myqcloud.com

# COS 버킷 example-bucket의 example-prefix 디렉터리를 test_cos_prefix 네임스페이스에
매핑
$ goosefs ns create test_cos_prefix cosn://example-bucket-1250000000/example-pref
ix/ --wPolicy 1 --rPolicy 1 --secret fs.cosn.userinfo.secretId=AKIDxxxxxxx --secr
et fs.cosn.userinfo.secretKey=xxxxxxxxxx --attribute fs.cosn.bucket.region=ap-gua
ngzhou --attribute fs.cosn.bucket.endpoint_suffix=cos.ap-guangzhou.myqcloud.com

# 클라우드 HDFS 파일 시스템 f4ma0l3qabc-Xy3을 test_chdfs 네임스페이스에 매핑
$ goosefs ns create test_chdfs ofs://f4ma0l3qabc-Xy3/ --wPolicy 1 --rPolicy 1 --a
ttribute fs.ofs.userinfo.appid=1250000000
```

©2013-2022 Tencent Cloud. All rights reserved.

생성 완료 후, goosefs fs ls 명령을 통해 디렉터리 세부 사항을 조회할 수 있습니다.

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

네임스페이스를 사용하지 않을 경우 delete 명령을 통해 삭제할 수 있습니다.

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

캐시 정책 설정

사용자는 setPolicy 및 unsetPolicy를 통해 지정된 네임스페이스의 캐시 정책을 설정할 수 있습니다. 캐싱 정책을 설정하기 위한 명령어 집합은 다음과 같습니다.

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

각 매개변수의 의미는 다음과 같습니다.

- wPolicy: 캐시 쓰기 정책. 6가지 캐시 쓰기 정책을 지원합니다.
- rPolicy: 캐시 읽기 정책. 5가지 캐시 읽기 정책을 지원합니다.
- namespace: 지정된 네임스페이스.

현재 GooseFS에서 지원하는 캐시 읽기 및 쓰기 정책은 다음과 같습니다.

캐시 쓰기 정책

정책 이름	동작	해당 Write_Type	데이터 보안성	쓰기 효율
MUST_CACHE(1)	데이터는 GooseFS에만 저장되며 원격 스토리지 시스템에는 기록하지 않습니다.	MUST_CACHE	없음	높음
TRY_CACHE(2)	캐시에 공간이 있으면 GooseFS에 기록하고, 캐시에 공간이 없으면 기본 스토리지에 기록합니다.	TRY_CACHE	없음	중간
CACHE_THROUGH (3)	가능한 한 많은 데이터를 캐싱하면서 동시에 원격 스토리지 시스템에 기록합니다.	CACHE_THROUGH	있음	낮음
THROUGH (4)	GooseFS에 데이터를 저장하지 않고 원격 스토리지 시스템에 직접 기록합니다.	THROUGH	있음	중간

©2013-2022 Tencent Cloud. All rights reserved.

정책 이름	동작	해당 Write_Type	데이터 보안성	쓰기 효율
ASYNC_THROUGH(5)	데이터는 GooseFS에 기록되고 원격 스토리지 시스템에 비동기적으로 새로 고침합니다.	ASYNC_THROUGH	약함	높음

설명：
Write_Type: 사용자가 SDK 또는 API를 호출하여 GooseFS에 데이터를 쓸 때 지정하는 파일 캐싱 정책을 말하며, 이는 단일 파일에 적용됩니다.

캐시 읽기 정책

정책 이름	동작	메타데이터 동기화	해당 Read_T
NO_CACHE(1)	데이터를 캐시하지 않고 원격 스토리지 시스템에서 직접 데이터를 읽습니다.	NO	NO_CACHE
CACHE(2)	• 메타데이터 액세스 동작: 캐시가 히트되면 메타데이터는 Master를 따르며, 메타데이터는 언더 스토리지부터 능동적으로 동기화되지 않습니다. • 데이터 스트림 액세스 동작: 데이터 스트림의 ReadType은 CACHE 정책을 사용합니다.	Once	CACHE
CACHE_PROMOTE(3)	• 메타데이터 액세스 동작: CACHE 모드와 동일합니다. • 데이터 스트림 액세스 동작: 데이터 스트림의 ReadType은 CACHE_PROMOTE 정책을 사용합니다.	Once	CACHE_PRO

©2013-2022 Tencent Cloud. All rights reserved.

정책 이름	동작	메타데이터 동기화	해당 Read_T
CACHE_CONSISTENT_PROMOTE(4)	• 메타데이터 동작: 각 읽기 작업 전에 원격 스토리지 시스템 UFS의 메타데이터가 동기화됩니다. UFS가 없으면 Not Exists 예외가 발생합니다. • 데이터 스트림 액세스 동작: 데이터 스트림의 ReadType은 CACHE_PROMOTE 정책을 사용하고 히트 후에는 가장 핫한 캐시 미디어에 캐시됩니다.	Always	CACHE
CACHE_CONSISTENT(5)	• 메타데이터 동작: CACHE_CONSISTENT_PROMOTE와 동일합니다. • 데이터 스트림 액세스 동작: 데이터 스트림의 ReadType은 CACHE 정책을 사용합니다. 즉, CACHE 히트 시 다른 미디어 레이어에서 데이터를 이동하지 않습니다.	Always	CACHE_PRC

설명：
Read_Type: 사용자가 SDK 또는 API를 호출하여 GooseFS에서 데이터를 읽을 때 지정하는 파일 캐싱 정책을 말하며, 이는 단일 파일에 적용됩니다.

현재 빅 데이터 비즈니스 사례와 결합하여 다음과 같은 캐시 읽기/쓰기 정책 조합을 권장합니다.

캐시 쓰기 정책	캐시 읽기 정책	정책 조합 성능
CACHE_THROUGH(3)	CACHE_CONSISTENT(5)	캐시 및 원격 스토리지 시스템 데이터 일관성 높음.
CACHE_THROUGH(3)	CACHE(2)	쓰기 일관성 높음, 읽기 최종 일관성.
ASYNC_THROUGH(5)	CACHE_CONSISTENT(5)	쓰기 최종 일관성, 읽기 일관성 높음.
ASYNC_THROUGH(5)	CACHE(2)	읽기/쓰기 최종 일관성.
MUST_CACHE(1)	CACHE(2)	캐시에서만 데이터 쓰기함.

©2013-2022 Tencent Cloud. All rights reserved.

다음 예시는 지정된 네임스페이스 test_cos의 캐시 읽기/쓰기 정책이 CACHE_THROUGH 및 CACHE_CONSISTENT로 설정되어있습니다.

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

주의 :
네임스페이스 생성 시 캐시 정책을 지정하는 것 외에도 사용자는 파일을 읽기/쓰기할 때 지정된 파일에 대해 ReadType 또는 Write_Type을 설정하거나 properties 프로파일을 통해 전역 캐시 정책을 설정할 수도 있습니다. 여러 정책이 동시에 존재하는 경우 우선순위는 사용자 정의 우선순위 > Namespace 읽기/쓰기 정책 > 프로파일의 전역 캐시 정책 설정입니다. 이 중 읽기 정책의 경우 사용자 정의 ReadType과 Namespace의 DirReadPolicy 조합이 적용됩니다. 즉, 데이터 스트림 읽기 정책은 사용자 정의 ReadType을 채택하고 메타데이터는 Namespace 정책을 채택합니다.

예를 들어 GooseFS에 COSN 네임스페이스가 있고 읽기 정책이 CACHE_CONSISTENT인 경우 이 네임스페이스에 test.txt 파일이 있다고 가정합니다. 클라이언트가 test.txt를 읽을 때 ReadType을 CACHE_PROMOTE로 지정하면, 전체 읽기 동작은 메타데이터와 CACHE_PROMOTE를 동기화합니다.

unsetPolicy 명령을 사용하여 읽기 및 쓰기 캐싱 정책을 재설정할 수 있습니다. 다음은 test_cos 네임스페이스의 캐시 읽기/쓰기 정책 재설정 관련 내용입니다.

```
$ goosefs ns unsetPolicy test_cos
```

TTL 설정

TTL은 GooseFS의 로컬 노드에 캐시된 데이터를 관리하는 데 사용되며, TTL 매개변수를 설정하면 캐시 데이터가 지정된 시간 후에 지정된 작업을 수행할 수 있습니다. 예: delete 또는 free 작업. 현재 TTL 설정 작업 지침은 다음과 같습니다.

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

각 매개변수의 의미는 다음과 같습니다.

- action: 캐시 시간이 만료된 후 수행되는 작업으로, 현재 delete 및 free 두 가지 작업을 지원합니다. delete 작업은 캐시 및 UFS의 데이터를 삭제하고, free 작업은 캐시의 데이터만 삭제합니다.
- namespace: 지정된 네임스페이스.
- time to live: 데이터 캐시 시간(단위: 밀리초).

다음 예시는 지정된 네임스페이스 test_cos의 정책이 60초 후에 삭제되도록 설정되었음을 보여줍니다.

©2013-2022 Tencent Cloud. All rights reserved.

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

메타데이터 정보 관리

이 섹션에서는 GooseFS가 메타데이터 동기화, 업데이트 등 메타데이터를 관리하는 방법을 소개합니다. GooseFS는 사용자에게 통합된 네임스페이스 기능을 제공합니다. 사용자는 통합된 gfs:// 경로를 통해 다른 기본 스토리지 시스템의 파일에 액세스할 수 있으며, 기본 스토리지 시스템의 경로만 지정하면 됩니다. 메타데이터 정보의 일관성을 보장하기 위해, GooseFS에서 데이터를 읽고 쓸 때 GooseFS를 통합 데이터 액세스 레이어로 사용할 것을 권장합니다.

메타데이터 동기화 개요

conf/goosefs-site.properties 프로파일에서 메타데이터 동기화 주기를 수정하여 메타데이터 동기화 주기를 관리할 수 있으며, 설정 매개변수는 다음과 같습니다.

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

동기화 주기는 다음 3가지 입력 매개변수를 지원합니다.

- 입력 매개변수 값이 -1일 경우: 처음 GooseFS에 로딩된 후 메타데이터 업데이트하지 않음.
- 입력 매개변수 값이 0일 경우: GooseFS가 각 읽기 및 쓰기 작업 후에 메타데이터를 업데이트함.
- 입력 매개변수 값이 양의 정수일 경우: GooseFS가 지정된 시간 간격으로 메타데이터를 주기적으로 업데이트함.

노드 수량, GooseFS 클러스터와 기본 스토리지의 I/O 간격, 기본 스토리지 유형을 종합적으로 고려하여 적절한 동기화 기간을 선택할 수 있습니다. 일반적인 경우는 다음과 같습니다.

- GooseFS 클러스터 노드 수가 많을수록 메타데이터 동기화 딜레이가 길어집니다.
- GooseFS 클러스터와 기본 스토리지 시스템 간의 거리가 멀수록, 메타데이터 동기화 딜레이가 길어집니다.
- 기본 스토리지 시스템이 메타데이터 동기화 딜레이에 미치는 영향은 주로 시스템 요청의 QPS 부하에 따라 결정됩니다. QPS 부하가 클수록 동기화 딜레이가 짧아집니다.

메타데이터 동기화 관리 방법

설정 방법

1. 명령 라인을 통해 설정합니다.
   명령 라인을 통해 메타데이터 정보의 동기화 기간을 설정할 수 있습니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

2. 프로파일을 통해 설정합니다.
대규모 Goosefs 클러스터의 경우, goosefs-site.properties 프로파일을 통해 클러스터 내 Master 노드의 메타데이터 정보 동기화 주기를 일괄 설정할 수 있으며, 다른 노드의 동기화 주기는 기본적으로 주기값에 따라 실행됩니다.

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

주의 :
많은 서비스는 디렉터리에 따라 데이터 용도 구별을 선택하고, 각각의 디렉터리 데이터 액세스 빈도는 완전히 동일하지는 않습니다. 메타데이터 동기화 주기는 디렉터리별로 다른 주기값을 설정할 수 있으며, 디렉터리가 자주 변경되는 일부 디렉터리의 경우, 디렉터리의 메타데이터 동기화 주기를 더 짧게(예: 5분) 설정할 수 있습니다. 변화가 극도로 적거나 변화가 없는 디렉터리는 동기화 주기를 -1로 설정하여 GooseFS가 디렉터리의 메타데이터를 자동으로 동기화하지 않도록 할 수 있습니다.

권장 설정

서비스 액세스 모드에 따라 다른 메타데이터 동기화 기간을 설정할 수 있습니다.

<table>
<tr><th colspan="2">액세스 모드</th><th>메타데이터 동기화 주기</th><th>설명</th></tr>
<tr><td colspan="2">모든 파일 요청은 GooseFS를 통과합니다.</td><td>-1</td><td>-</td></tr>
<tr><td rowspan="2">대부분의 파일 요청은 GooseFS를 통과합니다.</td><td>HDFS를 UFS로 사용합니다.</td><td>핫 업데이트 또는 경로별 업데이트 사용을 권장합니다.</td><td>HDFS가 매우 자주 업데이트되는 경우, 업데이트 주기를 -1로 설정할 것을 권장합니다. 업데이트가 금지됩니다.</td></tr>
<tr><td>COS를 UFS로 사용합니다.</td><td>경로에 따라 업데이트 주기를 설정할 것을 권장합니다.</td><td rowspan="3">각 디렉터리에 다른 업데이트 주기를 설정하면, 메타데이터 동기화의 부담을 완화할 수 있습니다.</td></tr>
<tr><td rowspan="2">파일 업로드 요청은 일반적으로 GooseFS를 거치지 않습니다.</td><td>HDFS를 UFS로 사용합니다.</td><td>경로에 따라 업데이트 주기를 설정할 것을 권장합니다.</td></tr>
<tr><td>COS를 UFS로 사용합니다.</td><td>경로에 따라 업데이트 주기를 설정할 것을 권장합니다.</td></tr>
</table>

©2013-2022 Tencent Cloud. All rights reserved.

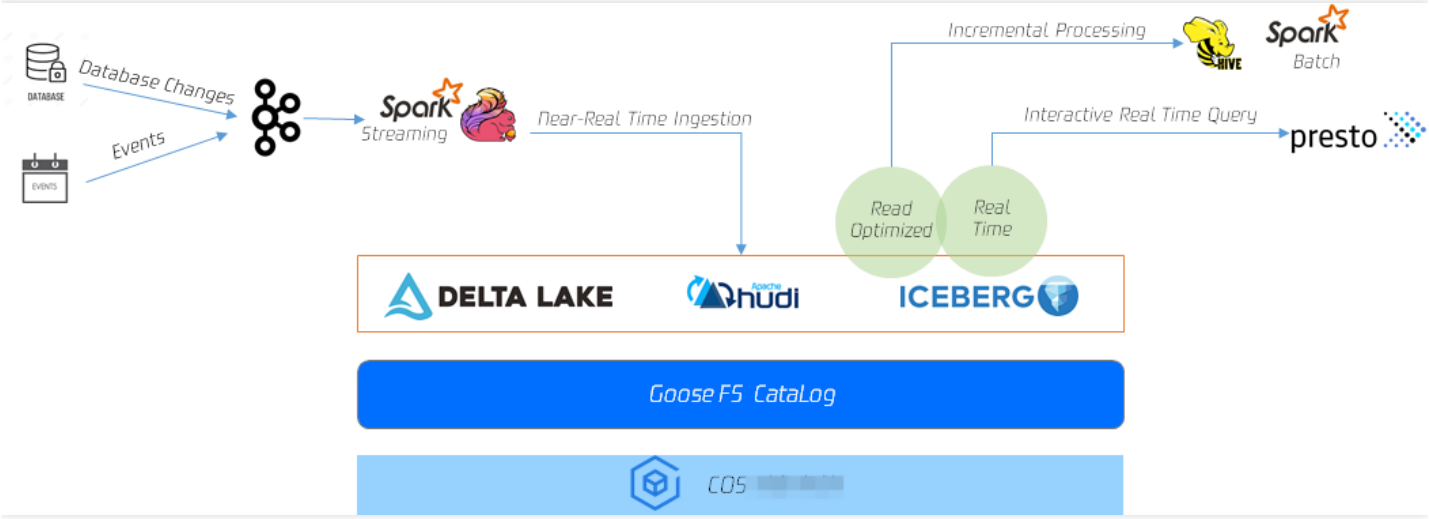
©2013-2022 Tencent Cloud. All rights reserved.

Table 관리 기능

최종 업데이트 날짜: : 2021-08-25 11:21:32

Table 관리 기능 개요

GooseFS Table 관리 능력을 통해 구조화 데이터를 관리합니다. SparkSQL, Hive, Presto 등 상위 컴퓨팅 애플리케이션에게 데이터베이스 테이블 관리 능력을 제공하고, 현재 Hive MetaStore 연결을 기본 지원합니다. Table 관리 능력은 각종 SQL 엔진이 지정한 데이터 내용을 읽게 하여 효과적으로 빅 데이터 시나리오에서 데이터의 액세스 효율을 높입니다.



GooseFS Table 관리 능력은 아래 특성을 지원합니다.

- 메타데이터 레이어의 기술 능력. GooseFS Catalog는 원격 메타데이터 서비스 (Hive MetaStore)의 메타데이터 캐시 서비스를 제공하며, SparkSQL, Hive, SQL Presto 등 SQL 엔진을 쿼리할 때, GooseFS Catalog의 메타데이터 캐시 서비스에 따라, 읽어오는 데이터의 크기, 타깃 데이터의 위치 및 데이터 구조를 확인할 수 있으며, Hive MetaStore와 동일한 능력을 갖고 있습니다.
- 테이블 레벨 데이터 프리캐싱 능력. GooseFS Catalog는 데이터 테이블과 데이터 스토리지 경로의 대응 관계를 감지할 수 있어 Table 레벨 및 Table Partition 레벨의 캐시 프리패치 능력을 제공하고, 사용자가 사전에 테이블 구조에 따라 데이터를 캐시하여 액세스 성능을 대폭 향상 시킵니다.
- 크로스 스토리지의 통합 메타데이터 서비스. GooseFS Catalog를 통해 상위 컴퓨팅 애플리케이션을 실행하면 각각 다른 기본 스토리지 시스템에 액세스 가속 능력을 동시에 제공할 수 있습니다. GooseFS Catalog는 스토리지 서비스간의 통합 메타데이터 쿼리 능력을 제공하여 하나의 GooseFS 클라이언트가 Catalog 기능을 활성화하기만 하면 각각 다른 스토리지 시스템을 확인할 수 있습니다. 예: HDFS, COS, CHDFS의 데이터.

©2013-2022 Tencent Cloud. All rights reserved.

GooseFS Table 의 관리 기능 사용

GooseFS Table 관리 기능은 goosefs table 명령어 집합을 통해 구현되며, DB의 바인딩과 바인딩 해제, DB 정보 조회, 테이블 정보 조회, 데이터 로딩, 데이터 제거 등 기능을 제공합니다. GooseFS Table 관리 명령어 집합은 아래와 같습니다.

```
$ 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>]
```

상기 명령어 집합의 각 명령어의 기능 설명은 아래와 같습니다.

- attachdb: 데이터베이스 마운트. 하나의 원격 데이터베이스를 GooseFS에 바인딩하며 현재 Hive MetaStore만 지원합니다.
- detachdb: 데이터베이스 언마운트. GooseFS에 바인딩된 데이터베이스의 바인딩을 해제합니다.
- free: 지정 DB.Table의 데이터 캐시 삭제. Partition 세분성을 지원합니다.
- load: 지정 DB.Table의 데이터 캐시. partition 세분성을 지원하며 replication을 통한 캐시의 복사본 수량 설정을 지원합니다.
- 1s: 지정 DB 혹은 DB.Table의 메타데이터 정보를 나열합니다.
- stat: 지정 DB.Table의 파일 수량, 크기, 캐시 백분율을 조회합니다.
- sync: 지정 DB의 내용을 동기화합니다.
- transform: 지정 DB에 연결된 Table을 새로운 Table로 전환합니다.
- transformStatus: Table 전환의 진행 상황을 확인합니다.

1. DB 마운트

지정 Table 데이터를 GooseFS 이전으로 프리패치하며 해당하는 DB를 GooseFS에 마운트해야 합니다. 아래 예시는 지정 주소 metastore_host:port의 데이터베이스 goosefs_db_demo를 GooseFS에 마운트 하고, GooseFS 내 해당 DB 이름을 test_db로 설정하는 것을 설명합니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

주의 :
metastore_host:port는 연결 가능한 임의의 합법 Hive MetaStore 주소로 변경할 수 있습니다.

2. Table 정보 확인

데이터베이스 바인딩 후 ls 명령어를 통해 마운트된 DB와 Table 정보를 확인할 수 있습니다. 아래 예시는 test_db의 web_page 테이블 정보를 확인하는 방법을 설명합니다.

```
$ 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. Table의 데이터 프리패치

Table 프리패치 명령어 전달 후 백그라운드에서 비동기화 작업을 전달하고, GooseFS는 작업 실행 후 하나의 작업 ID를 반환합니다. ‘job stat ’ 명령어를 통해 작업의 실행 상태를 확인할 수 있으며 ‘table stat’ 명령어를 통해 프리패치의 백분율을 확인할 수 있습니다. 프리패치 명령어는 아래와 같습니다.

©2013-2022 Tencent Cloud. All rights reserved.

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

4. Table 프리패치 상황 확인

'job stat' 명령어를 통해 Table 프리패치 작업의 진행률을 확인할 수 있습니다. COMPLETED 상태가 되면, 전체 프리패치 프로세스가 완료된 것이며, FAILED 상태일 경우 master.log 파일에서 로그 기록을 확인하여 프리패치 오류 원인을 진단할 수 있습니다.

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

Table 프리패치 완료 후 stat 명령어를 통해 지정 Table의 현황을 확인할 수 있습니다.

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

5. Table 릴리스

아래 명령어를 통해 GooseFS에서 지정 Table 데이터 캐시를 릴리스할 수 있습니다.

```
$ goosefs table ls test_db web_page
detail
```

6. DB 언마운트

아래 명령어를 통해 GooseFS에서 지정 DB를 언마운트 할 수 있습니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

GooseFS-FUSE 기능

최종 업데이트 날짜: : 2022-11-24 11:47:39

GooseFS-FUSE는 Unix 기기의 로컬 파일 시스템에 GooseFS 분산형 파일 시스템을 마운트할 수 있습니다. 이 기능을 사용하면 일부 표준 TCCLI(예: ls, cat 및 echo)가 GooseFS 분산형 파일 시스템의 데이터에 직접 액세스할 수 있습니다. 또한 더 중요한 것은 C, C++, Python, Ruby, Perl, Java와 같은 다른 언어로 구현된 애플리케이션은 모두 GooseFS의 클라이언트 통합 및 설정 없이 표준 POSIX 인터페이스(예시: open, write, read)를 통해 GooseFS를 읽고 쓸 수 있습니다.

GooseFS-FUSE는 FUSE 프로젝트를 기반으로 하며 대부분의 파일 시스템 작업을 지원합니다. 그러나 GooseFS의 일회성 불변 파일 데이터 모델과 같은 고유한 속성으로 인해 마운트된 파일 시스템은 POSIX 표준과 완전히 일치하지 않으며 여전히 제한성이 있습니다. 따라서 이 특징의 기능과 제한을 이해하려면 먼저 제한성을 읽으십시오.

설치 요구 사항

- JDK 1.8 이상
- Linux 시스템: libfuse 2.9.3 이상(버전 2.8.3은 사용 가능하나 경고 발생 가능성 있음)
- MAC 시스템: osxfuse 3.7.1 이상

사용법

GooseFS-FUSE 마운트

설정을 완료하고 GooseFS 클러스터를 시작한 후 GooseFS를 마운트해야 하는 노드에서 Shell을 실행하고 $GOOSEFS_HOME 디렉터리로 이동하여 다음 명령어를 실행합니다.

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

이 명령은 `<mount_point>` 로 지정된 경로에 해당 GooseFS 경로를 마운트하기 위해 백엔드 Java 프로세스를 실행합니다. 예를 들어 다음 명령은 GooseFS 경로 /people을 로컬 파일 시스템의 /mnt/people 디렉터리에 마운트합니다.

```
$ 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
```

GooseFS_path를 지정하지 않으면 GooseFS-FUSE는 기본적으로 GooseFS 루트 디렉터리(/)에 마운트됩니다. 이 명령을 여러 번 호출하여 GooseFS를 다른 로컬 디렉터리에 마운트할 수 있습니다. 모든 GooseFS-FUSE는

©2013-2022 Tencent Cloud. All rights reserved.

$GOOSEFS_HOME\logs\fuse.log 로그 파일을 공유합니다. 이 로그 파일은 오류 진단에 유용합니다.

> 주의 :
> `<mount_point>` 는 로컬 파일 시스템의 빈 폴더여야 하며 GooseFS-FUSE 프로세스를 실행한 사용자는 마운트 포인트 및 읽기/쓰기 권한이 있습니다.

GooseFS-FUSE 언마운트

GooseFS-FUSE를 언마운트할 때 노드에서 Shell을 실행하고 $GOOSEFS_HOME 디렉터리로 이동하여 다음 명령을 실행해야 합니다.

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

이 명령은 goosefs-fuse Java 백엔드 프로세스를 종료하고 파일 시스템을 언마운트합니다. 예시:

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

기본적으로 unmount 작업은 읽기/쓰기 작업이 완료되지 않은 경우 최대 120s 동안 기다립니다. 120s 후에도 읽기/쓰기 작업이 완료되지 않으면 Fuse 프로세스가 강제 종료되어 파일 읽기/쓰기가 실패하게 되므로 -s 매개변수를 추가하여 Fuse 프로세스가 강제 종료되는 것을 방지할 수 있습니다. 예시:

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

GooseFS-FUSE가 실행 중인지 확인

모든 마운트 포인트를 나열하려면 노드에서 Shell을 실행하고 $GOOSEFS_HOME 디렉터리로 이동하여 다음 명령을 실행해야 합니다.

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

이 명령은 pid, mount_point, GooseFS_path를 포함한 정보를 출력합니다.

예를 들어 출력은 다음 형식일 수 있습니다.

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

Goosefs-FUSE 디렉터리 구조

©2013-2022 Tencent Cloud. All rights reserved.

conf 디렉터리:

- masters: master 서버의 IP 구성 파일
- worker: worker 서버의 IP 구성 파일
- goosefs-site.properties: goosefs 구성 파일
- libexec: goosefs-fuse 종속 lib 파일 실행
- goosefs-fuse-1.4.0: goosefs-fuse 백엔드에서 실행되는 jar 패키지
- log: 로그 디렉터리

설정 옵션

GooseFS-FUSE는 표준 GooseFS-core-client-fs를 기반으로 작업합니다. 다른 애플리케이션을 사용하는 client와 같이 작업하려면, 이 GooseFS-core-client-fs의 동작을 사용자 정의하십시오.

$GOOSEFS_HOME/conf/goosefs-site.properties 구성 파일을 편집하여 클라이언트 옵션을 변경할 수 있습니다.

> 주의 :
> 모든 변경은 GooseFS-FUSE가 실행되기 전에 완료되어야 합니다.

제한성

©2013-2022 Tencent Cloud. All rights reserved.

현재 GooseFS-FUSE는 대부분의 기본 파일 시스템 작업을 지원합니다. 그러나 GooseFS의 몇 가지 특징으로 인해 다음 사항에 유의해야 합니다.

- 파일은 랜덤 및 추가로 작성될 수 없습니다.
- 파일은 한 번만 순차적으로 쓸 수 있으며 수정할 수 없습니다. 파일을 수정하려면 먼저 파일을 삭제한 다음 다시 생성하거나 O_TRUNC 식별자로 파일을 open하고 길이를 0으로 설정합니다.
- 마운트 지점에 작성 중인 파일은 읽을 수 없습니다.
- 파일 길이는 truncate할 수 없습니다.
- soft/hard link 미지원; GooseFS는 hard-link 및 soft-link의 개념이 없기 때문에 ln과 같은 관련 명령어는 지원하지 않습니다. 또한 hard-link에 대한 정보는 ll의 출력에 표시되지 않습니다.
- COS(Cloud Object Storage)가 기본 저장소로 사용되는 경우 Rename 작업은 원자가 아닙니다.
- GooseFS의 GooseFS.security.group.mapping.class 옵션이 ShellBasedUnixGroupsMapping 값으로 설정된 경우에만 파일의 사용자 및 그룹 정보가 Unix 시스템의 사용자 그룹에 해당합니다. 그렇지 않으면 chown 및 chgrp의 작업이 적용되지 않으며 ll이 반환하는 사용자 및 그룹은 GooseFS-FUSE 프로세스의 사용자 및 그룹 정보입니다.

성능 고려 사항

FUSE와 JNR을 함께 사용하기 때문에 마운트된 파일 시스템을 사용하는 경우 네이티브 파일 시스템 API를 직접 사용하는 경우보다 성능이 상대적으로 떨어집니다.

대부분의 성능 문제의 이유는 read 또는 write 작업을 할 때마다 메모리에 여러 복사본이 있고 FUSE는 쓰기의 최대 데이터 분할 정도를 128KB로 설정하기 때문입니다. kernel 3.15에 연결된 FUSE write-backs 캐시 정책을 이용하면 성능이 크게 향상될 수 있습니다(그러나 libfuse 2.x 사용자 공간 라이브러리는 현재 이 기능을 지원하지 않습니다).

GooseFS-FUSE 설정 매개변수

GooseFS-FUSE와 관련된 설정 매개변수는 다음과 같습니다.

매개변수	기본값	설명
goosefs.fuse.cached.paths.max	500	내부 GooseFS-FUSE 캐시 크기를 정의하며, 이 캐시는 로컬 파일 시스템 경로와 Alluxio 파일 URI 간의 가장 자주 사용되는 전환을 점검합니다.
goosefs.fuse.debug.enabled	false	`goosefs.logs.dir` 로 지정된 디렉터리의 `fuse.out` 로그 파일로 리디렉션되는 FUSE 디버깅 출력을 허용합니다.
goosefs.fuse.fs.name	goosefs-fuse	FUSE가 파일 시스템을 마운트하는 데 사용하는 설명형 이름입니다.

©2013-2022 Tencent Cloud. All rights reserved.

매개변수	기본값	설명
goosefs.fuse.jnifuse.enabled	true	더 나은 성능을 위해 JNI-Fuse 라이브러리를 사용합니다. 비활성화되면 JNR-Fuse가 사용됩니다.
goosefs.fuse.shared.caching.reader.enabled	false	(실험용) GooseFS JNI Fuse를 통한 다중 프로세스 파일 읽기 성능 향상을 위해 공유 grpc 데이터 리더를 사용합니다. 블록 데이터는 클라이언트에 캐시되므로 Fuse 프로세스에는 더 많은 메모리가 필요합니다.
goosefs.fuse.logging.threshold	10s	소요 시간이 임계값을 초과하면 FUSE API 호출을 기록합니다.
goosefs.fuse.maxwrite.bytes	131072	FUSE 쓰기 작업(bytes)의 데이터 분할 정도. 현재 128KB는 Linux 커널 제한의 상한선입니다.
goosefs.fuse.user.group.translation.enabled	false	FUSE API에서 GooseFS 사용자 및 그룹을 해당 Unix 사용자 및 그룹으로 전환할지 여부입니다. false로 설정하면 모든 FUSE 파일의 사용자 및 그룹이 goosefs-fuse 스레드가 마운트된 사용자 및 그룹으로 표시됩니다.

FAQ

libfuse 라이브러리 파일 누락

GooseFS-Fuse를 마운트하기 전에 libfuse를 설치해야 합니다.

```
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
```

©2013-2022 Tencent Cloud. All rights reserved.

- **방법1**
  설치 명령어:

```
yum install fuse-devel
```

설치 완료 여부 조회:

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

- **방법2**
  이전 버전 libfuse.so.2.9.2를 업데이트합니다. 설치 단계는 다음과 같습니다.

> 설명 :
> CentOS 7에 libfuse를 설치합니다. CentOS 7은 기본적으로 libfuse.so.2.9.2를 설치합니다.

  먼저 libfuse 소스 코드를 다운로드하고 libfuse.so.2.9.7을 컴파일 및 생성합니다.

```
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
```

다음으로, libfuse.so.2.9.7을 다운로드, 컴파일 및 생성한 후 아래 단계에 따라 설치 및 교체합니다.

1. 다음 명령을 실행하여 이전 버전 libfuse.so.2.9.2 라이브러리 링크를 찾습니다.

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

2. 다음 명령을 실행하여 libfuse.so.2.9.7을 이전 버전 라이브러리 libfuse.so.2.9.2 위치에 복사합니다.

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

3. 다음 명령을 실행하여 이전 libfuse.so 라이브러리에 대한 모든 링크를 삭제합니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

4. 다음 명령을 실행하여 삭제된 이전 버전의 링크와 유사한 libfuse.so.2.9.7 라이브러리를 링크합니다.

```
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
```

VIM을 사용하여 마운트 지점에서 파일을 편집하는 동안 Write error in swap file? 오류가 발생합니다.

GooseFS-Fuse 마운팅 포인트에서 파일 편집에 VIM 7.4 또는 이전 버전을 사용하도록 VIM 구성을 변경할 수 있습니다. VIM swap 파일은 시스템이 충돌하거나 다시 시작하는 경우 VIM이 swap 파일을 기반으로 저장하지 않은 수정 사항을 복원할 수 있도록 수정 사항을 유지하는 데 사용됩니다. 상기 오류는 VIM swap 파일에 대한 임의 쓰기 작업으로 인해 발생하며 GooseFS에서는 지원하지 않습니다. 솔루션: `:set noswapfile` 명령을 실행하여 swap 파일 생성을 종료하거나 구성 파일(~/.vimrc)에 `set noswapfile` 을 추가할 수 있습니다.

©2013-2022 Tencent Cloud. All rights reserved.

배포 가이드
자체 구축 클러스터로 배포

최종 업데이트 날짜: : 2023-04-06 14:28:02

본문은 주로 스탠드 얼론, 클러스터 및 Tencent Cloud EMR 클러스터(현재 GooseFS의 버전을 통합 지원하지 않습니다)에 GooseFS를 표준화적으로 배포, 설정, 실행하는 방법에 대해 설명합니다.

배포 환경 요건

하드웨어 환경

현재 GooseFS는 주요 x86/x64 아키텍처의 Linux/MacOS 실행 환경을 지원합니다. (**주의: MacOS M1 프로세서 지원은 검증되지 않았습니다**). 자세한 실행 노드 구성은 아래와 같습니다.

Master 노드

- **CPU**: 작업 클럭 속도는 1GHz 이상이어야 하며, Master 노드는 많은 양의 데이터를 처리해야 하므로 프로덕션 환경에서는 멀티 코어 프로세서를 사용하는 것이 좋습니다.
- **물리적 메모리**: 1GB 이상이어야 하며, 프로덕션 환경에서는 8GB 이상의 메모리를 사용하는 것이 좋습니다.
- **디스크**: 20GB 이상이어야 하며, 프로덕션 환경에는 고성능 NVME SSD 디스크를 메타데이터 캐시 디스크를 구비하는 것을 권장합니다(RocksDB 모드).

Worker 노드

- **CPU**: 작업 클럭 속도는 1GHz 이상이어야 하며, Worker 노드도 많은 동시 요청을 처리해야 하므로 프로덕션 환경에서는 멀티 코어 프로세서를 사용하는 것이 좋습니다.
- **물리적 메모리**: 2GB 이상이어야 하며, 성능 요구 사항에 따라 프로덕션 환경에 메모리를 할당할 수 있습니다. 예를 들어 Worker 노드 메모리에 많은 데이터 블록을 캐시해야 하거나 혼합 스토리지(MEM + SSD/HDD)를 사용해야 하는 경우 메모리에 캐시될 수 있는 데이터의 양에 따라 물리적 메모리를 할당할 수 있습니다. 사용하는 캐싱 모드에 관계없이 프로덕션 환경의 Worker에 대해 16GB의 물리적 메모리를 사용하는 것이 좋습니다.
- **디스크**: 20GB 이상의 SSD/HDD 디스크, 프로덕션 환경에 캐시해야 하는 데이터의 양을 추정하여 각 Worker 노드에 디스크 공간을 할당하는 것이 좋습니다. 또한 더 나은 성능을 위해 Worker 노드에 NVME SSD 디스크를 사용하는 것이 좋습니다.

소프트웨어 환경

- Red Hat Enterprise Linux 5.5+, Ubuntu Linux 12.04 LTS+(일괄 배포하지 않을 경우 지원 가능), CentOS 7.4+ 및 TLinux 2.x(Tencent Linux 2.x), Intel 아키텍처 기반의 MacOS 10.8.3 이상 버전. Tencent Cloud CVM 사용자는 CentOS 7.4, Tencent(TLinux 2.x) 혹은 TencentOS Server 2.4 버전의 운영 체제 사용을 권장합니다.

©2013-2022 Tencent Cloud. All rights reserved.

- OpenJDK 1.8/Oracle JDK 1.8, 주의해야 할 점은 JDK 1.9 이상의 버전에 대한 지원은 인증되지 않았습니다.
- SSH 툴 세트 (SSH와 SCP 툴 포함), Expect Shell (일괄 배포할 경우 필요), Yum 패키지 관리 툴을 지원합니다.

클러스터 네트워크 환경

- Master/Worker 노드 간 외부 네트워크 혹은 내부 네트워크 통신이 가능합니다. 일괄 배포할 경우에는 Master가 정상적으로 외부 네트워크 소프트웨어 보관소에 액세스할 수 있거나, 패키지 관리 시스템의 내부 네트워크 소프트웨어 소스를 올바르게 구성할 수 있어야 합니다.
- 클러스터는 외부 네트워크 혹은 내부 네트워크를 통해 COS 버킷이나 CHDFS 파일 시스템에 액세스 할 수 있습니다.

보안 그룹과 사용자 권한 요구

일반적으로 GooseFS는 사용자가 전용 Linux 계정을 사용하여 GooseFS의 배포 실행을 권장합니다. 예를 들어 자체 구축 클러스터와 EMR 환경에서 hadoop 사용자를 통합하여 GooseFS를 배포 실행할 수 있습니다. 일괄 배포할 경우 이래와 같은 사용자의 기능과 권한이 추가적으로 필요합니다.

- root로 전환하거나 sudo 권한을 사용할 수 있어야 합니다.
- 배포 실행 계정은 설치 디렉터리를 읽고 쓸 수 있는 권한이 있어야 합니다.
- Master 노드에는 클러스터의 모든 Worker 노드에 로그인할 수 있는 SSH 권한이 있어야 합니다.
- 클러스터에서 해당하는 노드 역할은 대응 포트를 개방해야 합니다. Master (9200과 9201), Worker (9203과 9204), Job Master (9205, 9206, 9207), Job Worker (9208, 9209, 9210), Proxy (9211), LogServer (9212).

스탠드 얼론 모드 배포 실행 (가상 분산형 아키텍처)

가상 분산형 아키텍처 배포는 주로 GooseFS의 체험과 디버깅에 적용되며 초급 사용자는 본인의 Linux 혹은 Mac 시스템 호스트에서 GooseFS를 빠르게 체험하고 디버깅할 수 있습니다.

배포

1. 먼저 GooseFS 바이너리 배포 패키지를 다운로드합니다.
2. 배포 패키지 다운로드 후 압축 해제하여 GooseFS의 디렉터리로 이동하고 아래의 작업을 실행합니다.

- conf/goosefs-site.properties.template 복사를 통해 conf/goosefs-site.properties 구성 파일을 생성합니다.

```
$ cp conf/goosefs-site.properties.template conf/goosefs-site.properties
```

- conf/goosefs-site.properties 구성 파일에서 `goosefs.master.hostname` 를 `localhost` 로 설정합니다.

©2013-2022 Tencent Cloud. All rights reserved.

- conf/goosefs-site.properties 구성 파일에서 `goosefs.master.mount.table.root.ufs` 를 로컬 파일 시스템의 디렉터리로 설정합니다. 예시: /tmp 혹은 /data/goosefs 등.

localhost의 SSH 비밀번호 없이 로그인으로 설정을 권장합니다. 그렇지 않을 경우 포맷과 실행 등 작업 시 localhost의 로그인 비밀번호를 입력해야 합니다.

실행

아래 명령어를 실행하면 하나의 RamFS 파일 시스템 마운트를 완료할 수 있습니다.

```
$ ./bin/goosefs-mount.sh SudoMount
```

다음과 같이 상기 명령을 실행하지 않고 GooseFS 클러스터를 시작할 때 직접 마운트할 수도 있습니다.

```
$ ./bin/goosefs-start.sh local SudoMount
```

실행 후 'jps' 명령어를 통해 가상 분산형 모드에서의 모든 GooseFS의 프로세스를 확인할 수 있습니다.

```
$ jps
35990 Jps
35832 GooseFSSecondaryMaster
35736 GooseFSMaster
35881 GooseFSWorker
35834 GooseFSJobMaster
35883 GooseFSJobWorker
35885 GooseFSProxy
```

그 후, 'goosefs' 명령 라인을 통해 namespace, fileSystem, job, talble의 각 작업을 실행할 수 있습니다. 예를 들어, 로컬 파일을 GooseFS에 업로드하고, 현재 GooseFS 루트 디렉터리의 파일과 디렉터리를 나열할 수 있습니다.

```
$ goosefs fs copyFromLocal test.txt /
Copied file:///Users/goosefs/test.txt to /

$ goosefs fs ls /
-rw-r--r-- goosefs staff 0 PERSISTED 04-28-2021 04:00:35:156 100% /test.txt
```

GooseFS의 명령 라인은 사용자에게 GooseFS의 모든 명령 라인 인터페이스의 관리와 액세스를 제공하며, 네임스페이스(namespace), 테이블(table), 작업 (job), 자주 사용하는 파일 시스템(fs) 작업 등을 포함합니다. 자세한 정보는 공식 문서를 참고하거나 'goosefs -h' 명령 라인 매개변수를 통하여 알 수 있습니다.

클러스터 모드 배포 실행

©2013-2022 Tencent Cloud. All rights reserved.

클러스터 배포 실행은 주로 사용자 자체구축 IDC 클러스터의 프로덕션 환경 혹은 미통합 GooseFS의 Tencent Cloud EMR 프로덕션 환경에 대한 것이며, 구체적으로 Standalone 아키텍처 배포와 고가용성 (HA) 아키텍처 배포로 나뉩니다.

GooseFS는 'scripts' 디렉터리에서 SSH 비밀번호 없이 로그인 일괄 설정 및 GooseFS의 스크립트 툴 일괄 설치 배포를 제공하며, 사용자가 빠르고 간편하게 GooseFS 대규모 클러스터의 설치 배포를 할 수 있습니다. 사용자는 이전 챕터에서 배포 환경 요건의 일괄 배포 조건을 다시 확인할 수 있으며, 일괄 배포 실행 여부를 효율적으로 선택할 수 있습니다.

일괄 배포 툴 소개

GooseFS는 'scripts' 디렉터리에서 SSH 비밀번호 없이 로그인 일괄 설정 및 GooseFS의 스크립트 툴 일괄 배포 설치를 제공하며, 사용자가 실행 조건을 만족할 경우 아래 순서대로 일괄 배포 작업을 할 수 있습니다.

- GooseFS 압축 해제 디렉터리 `cd ${GOOSEFS_HOME}` 으로 이동합니다.
- `conf/masters` 및 `conf/workers` 에서 호스트 이름 또는 IP 주소를 구성합니다.
- `scripts` 디렉터리로 이동하여 `commons.sh` 구성 파일의 `SCRIPT_EXEC_USER` 및 `SCRIPT_EXEC_PASSWORD` 를 구성합니다. 그 다음 `root` 계정으로 전환하거나 `sudo` 를 사용하여 `config_ssh.sh` 를 실행하여 전체 클러스터에 대해 암호 없는 로그인을 구성할 수 있습니다.
- '비밀번호 없이 로그인' 설정 완료 후 `validate_env.sh` 툴을 실행하여 전체 클러스터 설정 상태를 검증합니다.
- 구성 파일 디렉터리 `ln -s conf ~/.goosefs` 에 대한 소프트 링크를 생성합니다.
- 마지막으로 `goosefs copy .` 및 `goosefs copy ~/.goosefs` 를 실행하여 goosefs 바이너리 패키지와 구성 파일을 모든 노드에 배포합니다.

모든 배포 프로세스 완료 후 `./bin/goosefs-start.sh all SudoMount` 실행을 통해 전체 클러스터를 실행할 수 있습니다. 모든 실행 로그는 `${GOOSEFS_HOME}/logs` 에 기록되도록 기본 설정되어 있습니다.

Standalone 아키텍처 배포

Standalone 아키텍처는 단일 Master 노드, 다중 Worker 노드의 클러스터 배포 아키텍처를 적용했습니다. 아래 순서를 참고하여 배포를 실행합니다.

1. GooseFS 바이너리 배포 패키지를 다운로드합니다.

2. `tar zxvf goosefs-x.x.x-bin.tar.gz` 명령어를 통해 설치 경로 뒤에 압축 해제합니다. 일괄 배포 툴의 소개를 참고하여 클러스터의 일괄 배포를 설정 및 실행할 수 있으며, 아래 상세한 수동 배포 프로세스 문장을 참고할 수 있습니다.

   (1) 'conf' 디렉터리에서 'template' 파일을 복사하여 구성 파일을 생성합니다.

©2013-2022 Tencent Cloud. All rights reserved.

```
$ cp conf/goosefs-site.properties.template conf/goosefs-site.properties
```

(2) `goosefs-site.properties` 구성 파일에서 아래와 같은 설정을 지정합니다.

```
goosefs.master.hostname=<MASTER_HOSTNAME>
goosefs.master.mount.table.root.ufs=<STORAGE_URI>
```

`goosefs.master.hostname` 을 단일 master 노드의 hostname 혹은 ip로 설정합니다.
`goosefs.master.mount.table.root.ufs` 는 지정 GooseFS 루트 디렉터리가 마운트한 유닉스 파일 시스템(UFS) 경로 URI 입니다.

> 주의 :
> 해당 URI는 반드시 Master와 Worker 노드 모두 액세스 할 수 있어야 하므로 로컬 디렉터리는 지원되지 않습니다.

예를 들어, GooseFS를 루트 경로로 한 COS 경로를 마운트 할 수 있습니다.
goosefs.master.mount.table.root.ufs=cosn://bucket-1250000000/goosefs/.

'masters' 구성 파일에서 단일 Master 노드의 hostname 혹은 ip를 지정합니다. 예시:

```
# The multi-master Zookeeper HA mode requires that all the masters can access
# the same journal through a shared medium (e.g. HDFS or NFS).
# localhost
cvm1.compute-1.myqcloud.com
```

'workers' 구성 파일에서 모든 Worker 노드의 host 혹은 ip를 지정합니다. 예시:

```
# An GooseFS Worker will be started on each of the machines listed below.
# localhost
cvm2.compute-2.myqcloud.com
cvm3.compute-3.myqcloud.com
```

모든 설정 완료 후 `./bin/goosefs copyDir conf/` 을 실행하면 구성을 모든 노드에 동기화할 수 있습니다.

마지막으로 `./bin/goosefs-start.sh all` 을 실행하면 GooseFS 클러스터를 실행할 수 있습니다.

고가용성 아키텍처 배포

단일 Master 노드의 Standalone 아키텍처는 단일 문제가 쉽게 발생됨으로 실제 프로덕션 환경은 다중 Master 노드를 배포하여 고가용성 시스템 아키텍처로 사용을 권장합니다. 다수의 Master 노드 중 한 노드만 프라이머리(leader) 노드로 외부 서비스를 제공하고, 그 외 노드는 세컨더리(Standby) 노드로 공유 로그 (Journal)를 동기화하며 프라이머리

©2013-2022 Tencent Cloud. All rights reserved.

노드와 동일한 파일 시스템 상태를 유지합니다. 프라이머리 노드가 장애로 다운 됐을 경우 세컨더리 노드 중의 하나를 자동으로 선택하여 프라이머리 노드 대신 외부 서비스를 제공합니다. 이로써 시스템의 단일 장애를 제거하여 고가용성 아키텍처를 사용할 수 있습니다.

현재 GooseFS는 Raft 로그와 Zookeeper 기반의 2가지 방식으로 프라이머리/세컨더리 노드 상태의 강한 일치성을 지원합니다. 다음으로 이 2가지 배포 방식에 대하여 각각 소개하겠습니다.

Raft 임베디드 로그 기반의 고가용성 배포 설정

설정 템플릿에 따라 구성 파일을 생성합니다.

```
$ cp conf/goosefs-site.properties.template conf/goosefs-site.properties
```

```
goosefs.master.mount.table.root.ufs=<STORAGE_URI>
goosefs.master.embedded.journal.addresses=<EMBBEDDED_JOURNAL_ADDRESS>
```

> 설명 :
> 위의 구성 항목의 설명은 다음과 같습니다.

- `goosefs.master.mount.table.root.ufs` 는 GooseFS 루트 디렉터리의 기본 스토리지 URI 마운트로 설정을 합니다.
- `goosefs.master.embedded.journal.addresses` 는 모든 세컨더리 노드의 `ip:embedded_journal_port` 혹은 `host:embedded_journal_port` 를 설정합니다. embedded_journal_port는 9202로 기본 설정되어 있습니다. 예시: 192.168.1.1:9202, 192.168.1.2:9202, 192.168.1.3:9202.

Raft 임베디드 로그 기반의 배포 방식은 copycat의 Leader 선출 메커니즘에 종속됨으로 Raft의 HA 배포 아키텍처는 Zookeeper와 함께 사용할 수 없습니다.

모든 설정 완료 후 아래 명령어를 통해 모든 설정을 동기화할 수 있습니다.

```
$ ./bin/goosefs copyDir conf/
```

포맷 완료 후 바로 GooseFS 클러스터를 실행할 수 있습니다.

```
$ ./bin/goosefs format
```

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

©2013-2022 Tencent Cloud. All rights reserved.

권한이 부족하다는 메시지가 표시되면 다음과 같이 재시작합니다.

```
$ ./bin/goosefs-stop.sh all
$ ./bin/goosefs-start.sh all SudoMount
```

아래 명령어를 통해 현재 프라이머리(Leader) 노드를 확인할 수 있습니다.

```
$ ./bin/goosefs fs leader
```

아래 명령어를 통해 클러스터 상태를 확인할 수 있습니다.

```
$ ./bin/goosefs fsadmin report
```

Zookeeper 기반의 공유 로그 배포 설정

Zookeeper 서비스 구축 GooseFS의 고가용성 아키텍처 설정은 아래 조건을 만족해야 합니다.

- Zookeeper 클러스터, GooseFS Master는 Zookeeper를 통해 Leader를 선택하고, GooseFS의 클라이언트와 Worker 노드는 Zookeeper를 통해 프라이머리 Master 노드를 확인합니다.
- 모든 GooseFS의 Master 노드가 모두 액세스할 수 있는 고가용성, 고일치성 공유 스토리지 시스템. 프라이머리 Master 노드가 로그를 해당 스토리지 시스템에 쓰고 세컨더리 (Standby) 노드는 지속적으로 공유 스토리지 시스템에서 로그를 읽으며 프라이머리 로그의 상태와 일치하도록 유지합니다. 일반적으로 해당 공유 스토리지 시스템을 HDFS 혹은 COS로 설정을 권장합니다. 예시: `hdfs://10.0.0.1:9000/goosefs/journal` 혹은 `cosn://bucket-1250000000/journal` .

설정은 아래와 같습니다.

```
goosefs.zookeeper.enabled=true
goosefs.zookeeper.address=<ZOOKEEPER_ADDRESSS>
goosefs.master.journal.type=UFS
goosefs.master.journal.folder=<JOURNAL_URI>
```

ZK 모드의 'JOURNAL_URI'는 공유 스토리지 시스템의 URI여야 합니다. 그 다음 `./bin/goosefs copyDir conf/` 를 사용하여 클러스터의 모든 노드와 설정을 동기화 하고 클러스터 `./bin/goosefs-start.sh all` 을 실행합니다.

GooseFS의 프로세스 리스트

`goosefs-start.sh all` 스크립트를 실행하고 GooseFS를 실행하면 클러스터에 아래와 같은 프로세스가 포함됩니다.

©2013-2022 Tencent Cloud. All rights reserved.

프로세스	설명
GooseFSMaster	기본 RPC 포트: 9200, Web 포트: 9201
GooseFSWorker	기본 RPC 포트: 9203, Web 포트: 9204
GooseFSJobMaster	기본 RPC 포트: 9205, Web 포트: 9206
GooseFSProxy	기본 Web 포트: 9211
GooseFSJobWorker	기본 RPC 포트: 9208, Web 포트: 9210

©2013-2022 Tencent Cloud. All rights reserved.

Tencent Cloud EMR을 통한 배포

최종 업데이트 날짜: : 2022-09-16 10:11:13

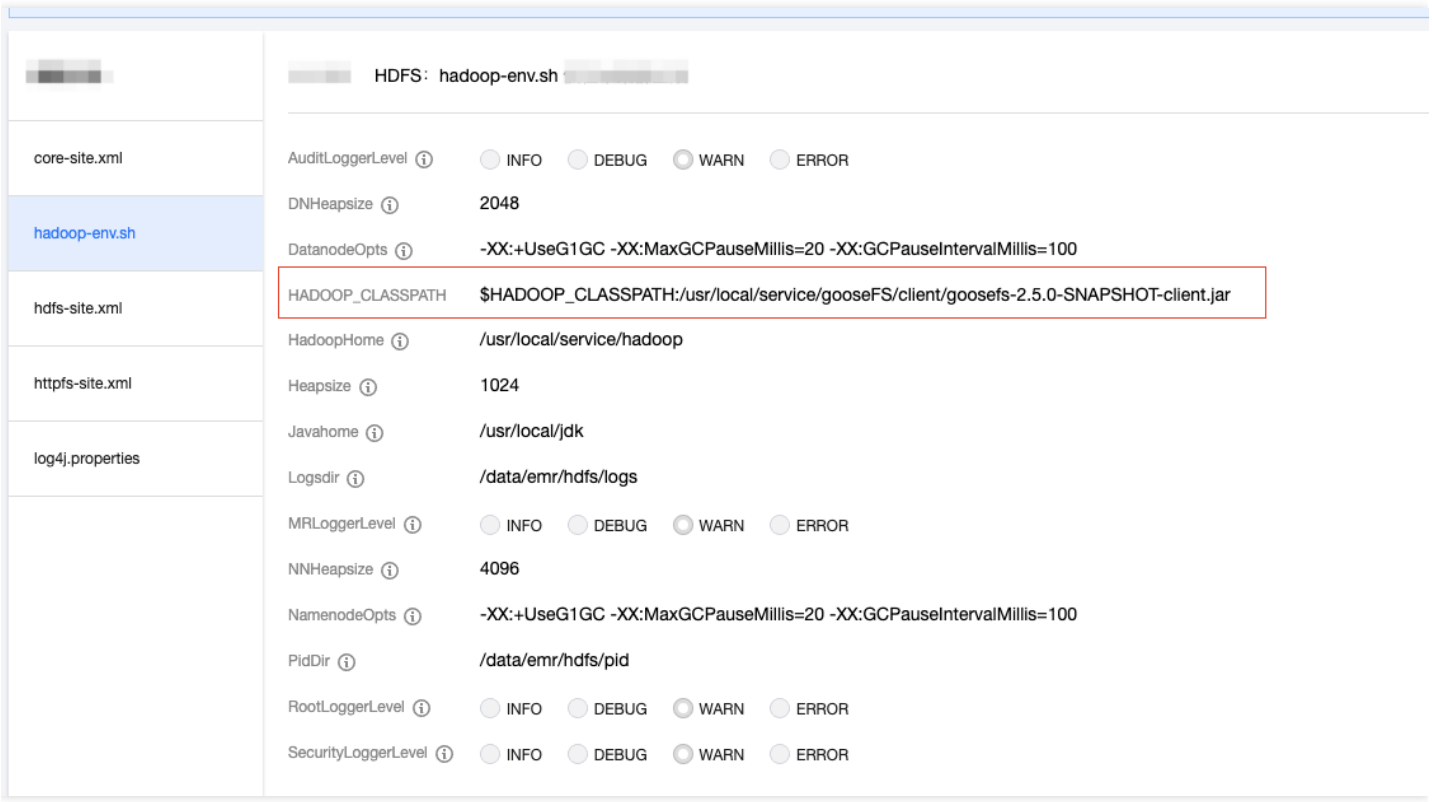
현재 GooseFS는 Tencent Cloud EMR 환경으로 통합되었으며 EMR 최신 버전에 배포될 예정입니다. 사용자는 더 이상 Tencent Cloud EMR 환경을 단독 배포하지 않아도 되며, 기타 EMR 컴포넌트를 사용하는 것과 같이 바로 GooseFS를 사용할 수 있습니다.

아래 문장은 통합되지 않은 GooseFS의 Tencent Cloud EMR 기존 저장 클러스터에 대해 GooseFS의 EMR환경 설정을 배포하는 방법에 대해 소개합니다.

우선, 클러스터 모드 배포 실행챕터의 내용을 참고하여 프로덕션 환경에 적합한 배포 아키텍처를 선택하고 클러스터를 배포합니다.
다음으로, EMR이 컴포넌트를 설정하는 것을 지원하므로 본문은 Hadoop MapReduce, Spark, Flink의 GooseFS에 대한 지원으로 설명합니다.

Hadoop MapReduce 지원

Hadoop의 MapReduce 작업이 GooseFS의 데이터를 읽고 쓰게 하기 위해 hadoop-env.sh에서 GooseFS Client의 종속 경로를 HADOOP_CLASSPATH에 추가해야 합니다. 해당 작업은 EMR의 콘솔에서 할 수 있습니다. 예시:

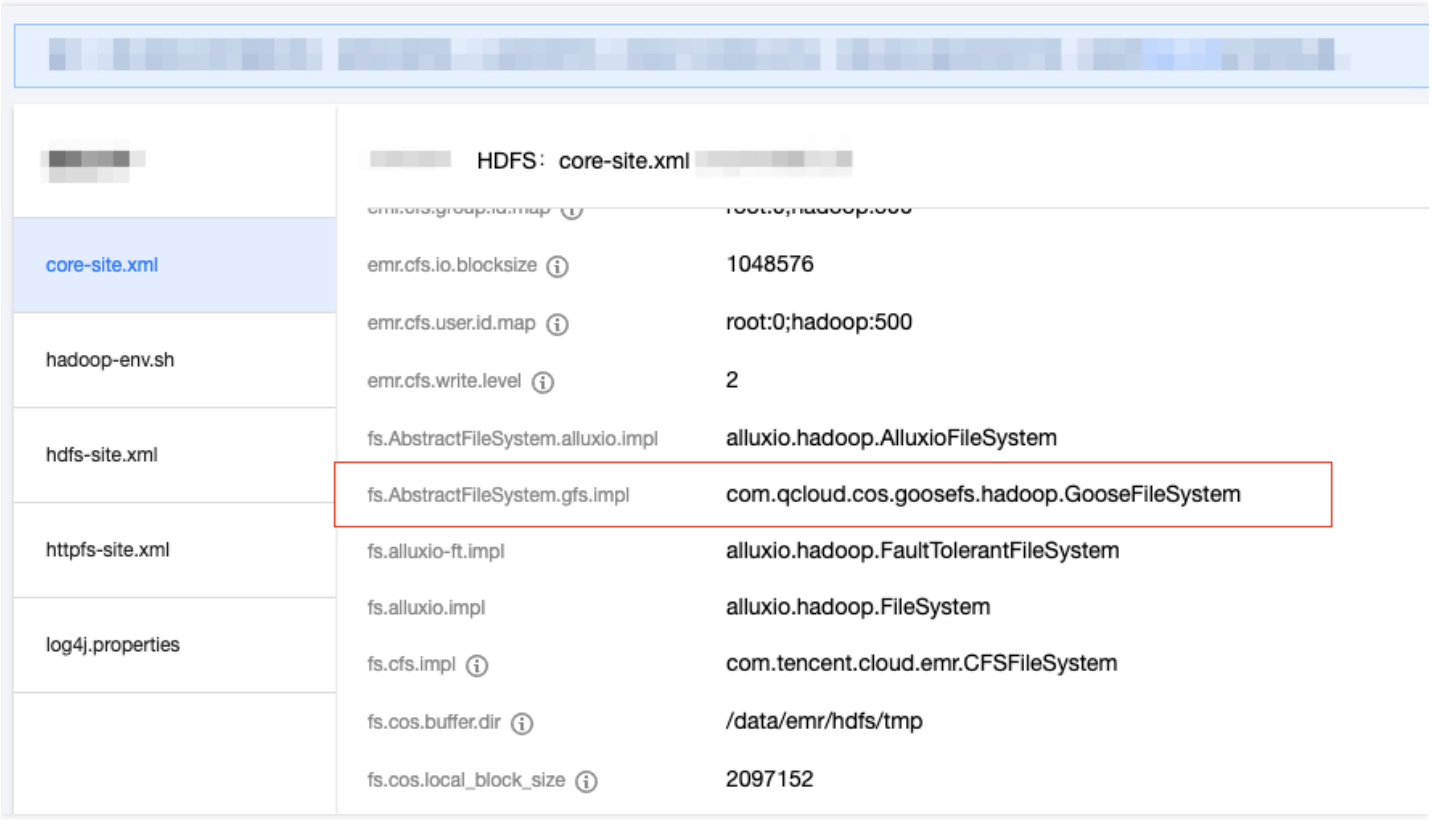


©2013-2022 Tencent Cloud. All rights reserved.

동시에 core-site.xml에서 GooseFS의 HCFS 구현을 설정해야 하며, 해당 작업도 EMR 콘솔에서 할 수 있습니다.

fs.AbstractFileSystem.gfs.impl의 설정은 아래와 같습니다.

```
com.qcloud.cos.goosefs.hadoop.GooseFileSystem
```



fs.gfs.impl의 설정은 아래와 같습니다.

```
com.qcloud.cos.goosefs.hadoop.FileSystem
```

©2013-2022 Tencent Cloud. All rights reserved.

설정 전달 후 YARN 관련 컴포넌트를 재시작하면 바로 적용됩니다.

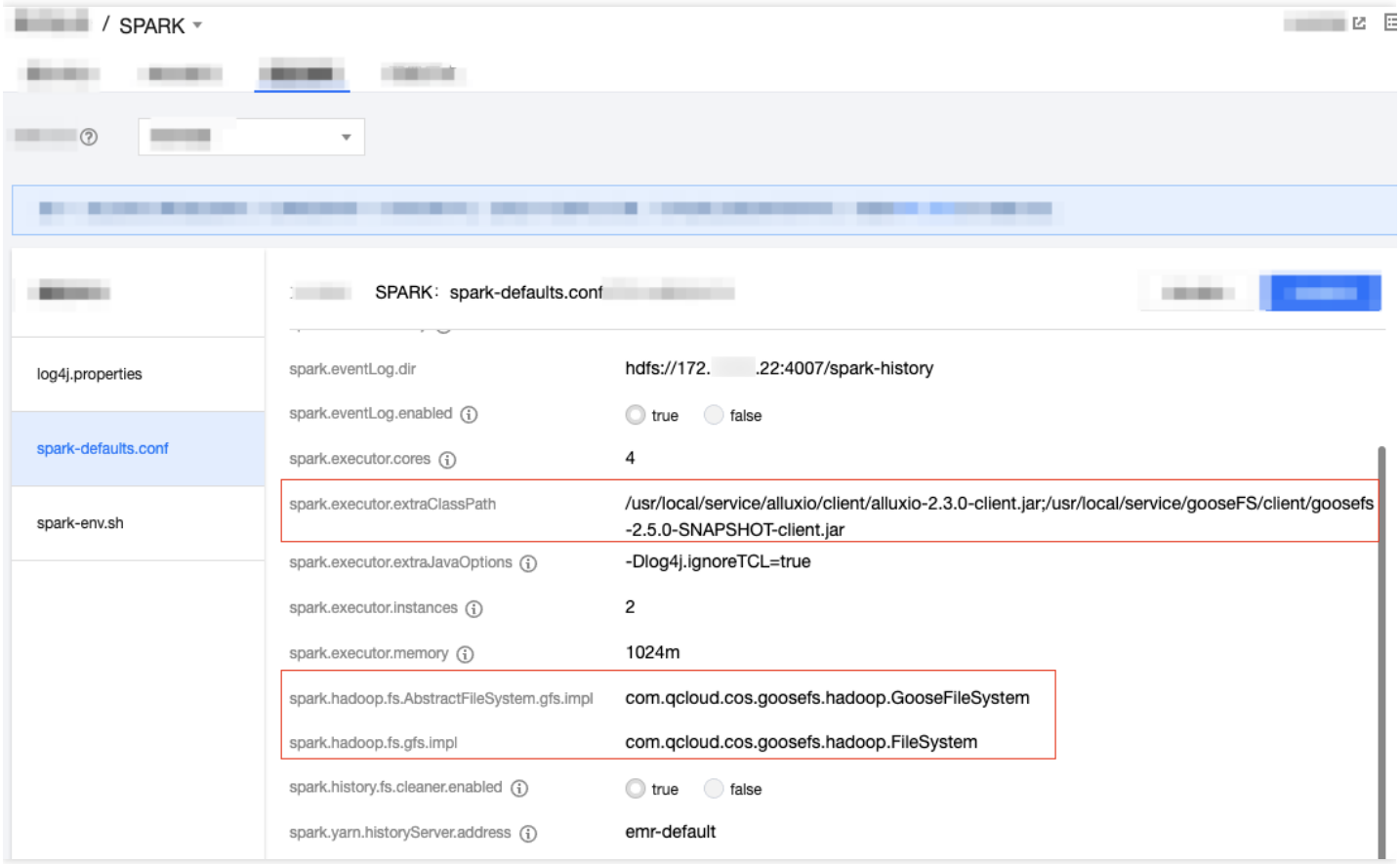
Spark 지원

Spark가 goosefs에 액세스하게 하기 위해 GooseFS의 client 종속 패키지를 spark의 executor classpath에 설정해야 하며 spark-defaults.conf에 지정해야 합니다.

```
...
spark.driver.extraClassPath ${GOOSEFS_HOME}/client/goosefs-x.x.x-client.jar
spark.executor.extraClassPath ${GOOSEFS_HOME}/client/goosefs-x.x.x-client.jar
spark.hadoop.fs.gfs.impl com.qcloud.cos.goosefs.hadoop.FileSystem
spark.hadoop.fs.AbstractFileSystem.gfs.impl com.qcloud.cos.goosefs.hadoop.GooseFi
leSystem
...
```

해당 작업도 EMR 콘솔의 Spark 컴포넌트에서 설정과 전달을 할 수 있습니다.

©2013-2022 Tencent Cloud. All rights reserved.



Flink 지원

Tencent Cloud EMR의 Flink는 Flink on YARN의 배포 모드를 적용하였기에 원칙적으로 ${FLINK_HOME}/flink-conf.yaml에서 'fs.hdfs.hadoopconf'를 hadoop의 설정 경로에 정확하게 설정만 하면 됩니다. Tencent Cloud EMR 클러스터에서 `/usr/local/service/hadoop/etc/hadoop` 이 일반적 입니다.

기타 설정 항목을 설정하지 않고 바로 Flink on YARN의 방식으로 Flink 작업을 제출하면 됩니다. 작업 중 GooseFS의 액세스 경로는 `gfs://master:port/<path>` 입니다.

> 주의 :
> Flink가 GooseFS로 액세스할 경우 반드시 master 와 port를 지정해야 합니다.

Hive, Impala, HBase, Sqoop, Oozie를 지원합니다.

©2013-2022 Tencent Cloud. All rights reserved.

Hadoop MapReduce의 환경 지원 설정 후 Hive, Impala, HBase 등 컴포넌트는 별도의 설정 지원 없이 정상 사용 가능합니다.

©2013-2022 Tencent Cloud. All rights reserved.

Tencent Cloud TKE를 통한 배포

최종 업데이트 날짜: : 2021-11-16 11:52:09

TKE를 이용한 GooseFS 배포는 오픈소스 모듈 Fluid 배포가 필요하며, TKE 애플리케이션은 이미 런칭되었습니다. 배포에는 다음 두 단계가 포함됩니다.

1. Fluid helm chart를 통한 controller 배포.
2. kubectl를 통한 Dataset와 GooseFS runtime 생성.

준비 사항

1. Tencent Cloud TKE 클러스터가 있어야 합니다.
2. kubectl v1.18 이상 버전이 설치되어 있어야 합니다.

설치 방법

1. TKE 애플리케이션 마켓에서 fluid 애플리케이션을 찾습니다.

2. Fluid Controller를 설치합니다.

3. controller 모듈을 검사합니다. 좌측 [클러스터]에서 해당 클러스터를 찾고 2개의 controller가 확인되었다면 fluid 모듈이 성공적으로 설치가 되었음을 의미합니다.

작업 예시

1. 클러스터 액세스 권한

```
[root@master01 run]# export KUBECONFIG=xxx/cls-xxx-config (tke 콘솔 페이지에서 클러스터 증명을 특정 디렉터리에 다운로드합니다.)
```

> 주의 :
> 클러스터 API Server의 외부 네트워크 액세스 권한을 활성화해야 합니다.

2. UFS 데이터 세트 Dataset 생성 (예시: COS)

©2013-2022 Tencent Cloud. All rights reserved.

암호화를 위한 secret.yaml을 생성합니다. 템플릿은 다음과 같습니다.

```
apiVersion: v1
kind: Secret
metadata:
name: mysecret
stringData:
fs.cosn.userinfo.secretKey: xxx
fs.cosn.userinfo.secretId:xxx
```

secret 생성:

```
[root@master01 ~]# kubectl apply -f secret.yaml
secret/mysecret created
```

dataset.yaml 템플릿은 아래와 같습니다.

```
apiVersion: data.fluid.io/v1alpha1
kind: Dataset
metadata:
name: slice1
spec:
mounts:
- mountPoint: cosn://{your bucket}
name: slice1
options:
fs.cosn.bucket.region: ap-beijing
fs.cosn.impl: org.apache.hadoop.fs.CosFileSystem
fs.AbstractFileSystem.cosn.impl: org.apache.hadoop.fs.CosN
fs.cosn.userinfo.appid: "${your appid}"
encryptOptions:
- name: fs.cosn.userinfo.secretKey
valueFrom:
secretKeyRef:
name: mysecret
key: fs.cosn.userinfo.secretKey
- name: fs.cosn.userinfo.secretId
valueFrom:
secretKeyRef:
name: mysecret
key: fs.cosn.userinfo.secretId
```

dataset 생성

©2013-2022 Tencent Cloud. All rights reserved.

```
[root@master01 run]# kubectl apply -f dataset.yaml
dataset.data.fluid.io/slice1 created
```

Dataset 상태 조회, NotBond 상태.

```
[root@master01 run]# kubectl get dataset
NAME UFS TOTAL SIZE CACHED CACHE CAPACITY CACHED PERCENTAGE PHASE AGE
slice1
NotBound 11s
```

3. runtime 생성

runtime.yaml 템플릿은 아래와 같습니다.

```
apiVersion: data.fluid.io/v1alpha1
kind: GooseFSRuntime
metadata:
name: slice1
spec:
replicas: 1
data:
replicas: 1
goosefsVersion:
imagePullPolicy: Always
image: ${img_uri}
imageTag: ${tag}
tieredstore:
levels:
- mediumtype: MEM
path: /dev/shm
quota: 1Gi
high: "0.95"
low: "0.7"
properties:
# goosefs user
goosefs.user.file.writetype.default: MUST_CACHE
master:
replicas: 1
journal:
volumeType: hostpath
jvmOptions:
- "-Xmx40G"
- "-XX:+UnlockExperimentalVMOptions"
- "-XX:ActiveProcessorCount=8"
worker:
```

©2013-2022 Tencent Cloud. All rights reserved.

```
jvmOptions:
- "-Xmx12G"
- "-XX:+UnlockExperimentalVMOptions"
- "-XX:MaxDirectMemorySize=32g"
- "-XX:ActiveProcessorCount=8"
resources:
limits:
cpu: 8
fuse:
imagePullPolicy: Always
image: ${fuse_uri}
imageTag: ${tag_num}
env:
MAX_IDLE_THREADS: "32"
jvmOptions:
- "-Xmx16G"
- "-Xms16G"
- "-XX:+UseG1GC"
- "-XX:MaxDirectMemorySize=32g"
- "-XX:+UnlockExperimentalVMOptions"
- "-XX:ActiveProcessorCount=24"
resources:
limits:
cpu: 16
args:
- fuse
- --fuse-opts=kernel_cache,ro,max_read=131072,attr_timeout=7200,entry_timeout=720
0,nonempty
```

runtime 생성.

```
[root@master01 run]# kubectl apply -f runtime.yaml
goosefsruntime.data.fluid.io/slice1 created
```

goosefs 모듈 상태 확인.

```
[root@master01 run]# kubectl get pods
NAME READY STATUS RESTARTS AGE
slice1-fuse-xsvwj 1/1 Running 0 37s
slice1-master-0 2/2 Running 0 118s
slice1-worker-fzpdw 2/2 Running 0 37s
```

4. 프리패치 데이터

dataload.yaml 프리패치 모듈은 아래와 같습니다.

©2013-2022 Tencent Cloud. All rights reserved.

```
apiVersion: data.fluid.io/v1alpha1
kind: DataLoad
metadata:
name: slice1-dataload
spec:
dataset:
name: slice1
namespace: default
```

이때 Dataset는 Bond 상태이며 Cached 비율은 0%입니다.

```
[root@master01 run]# kubectl get dataset
NAME UFS TOTAL SIZE CACHED CACHE CAPACITY CACHED PERCENTAGE PHASE AGE
slice1 97.67MiB 0.00B 4.00GiB 0.0% Bound 31m
```

데이터 프리패치 실행.

```
[root@master01 run]# kubectl apply -f dataload.yaml
dataload.data.fluid.io/slice1-dataload created
```

데이터 프리패치 진행률 확인.

```
[root@master01 run]# kubectl get dataset --watch
NAME UFS TOTAL SIZE CACHED CACHE CAPACITY CACHED PERCENTAGE PHASE AGE
slice1 97.67MiB 52.86MiB 4.00GiB 54.1% Bound 39m
slice1 97.67MiB 53.36MiB 4.00GiB 54.6% Bound 39m
slice1 97.67MiB 53.36MiB 4.00GiB 54.6% Bound 39m
slice1 97.67MiB 53.87MiB 4.00GiB 55.2% Bound 39m
slice1 97.67MiB 53.87MiB 4.00GiB 55.2% Bound 39m
```

데이터 프리패치 100% 완료.

```
[root@master01 run]# kubectl get dataset --watch
NAME UFS TOTAL SIZE CACHED CACHE CAPACITY CACHED PERCENTAGE PHASE AGE
slice1 97.67MiB 97.67MiB 4.00GiB 100.0% Bound 44m
```

5. 데이터 검사

```
[root@master01 run]# kubectl get pods
NAME READY STATUS RESTARTS AGE
slice1-dataload-loader-job-km6mg 0/1 Completed 0 12m
slice1-fuse-xsvwj 1/1 Running 0 17m
slice1-master-0 2/2 Running 0 19m
slice1-worker-fzpdw 2/2 Running 0 17m
```

©2013-2022 Tencent Cloud. All rights reserved.

goosefs master 컨테이너로 이동

```
[root@master01 run]# kubectl exec -it slice1-master-0 -- /bin/bash
Defaulting container name to goosefs-master.
```

goosefe 디렉터리 나열.

```
[root@VM-2-40-tlinux goosefs-1.0.0-SNAPSHOT-noUI-noHelm]# goosefs fs ls /slice1
10240 PERSISTED 06-25-2021 16:45:11:809 100% /slice1/p1
1 PERSISTED 05-24-2021 16:07:37:000 DIR /slice1/a
10000 PERSISTED 05-26-2021 19:29:05:000 DIR /slice1/p2
```

특정 파일 보기.

```
[root@VM-2-40-tlinux goosefs-1.0.0-SNAPSHOT-noUI-noHelm]# goosefs fs ls /slice1/
a/
12 PERSISTED 06-25-2021 16:45:11:809 100% /slice1/a/1.xt
```

©2013-2022 Tencent Cloud. All rights reserved.

Tencent Cloud EKS를 사용하여 배포

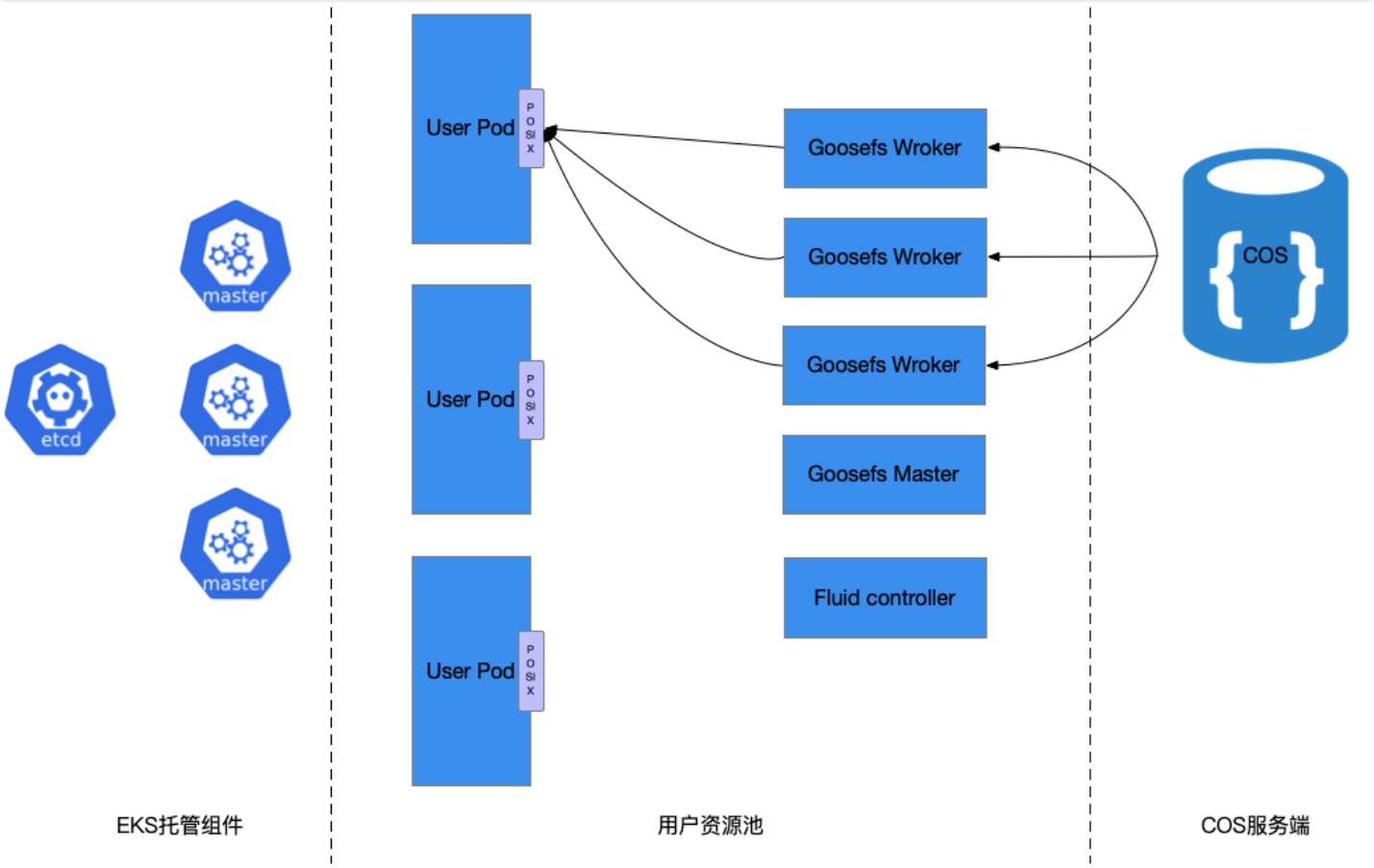
최종 업데이트 날짜: : 2022-01-27 09:19:51

Elastic Kubernetes Service(EKS)는 사용자가 노드를 구매하지 않고도 워크로드를 배포할 수 있는 Tencent Kubernetes Engine입니다. EKS는 네이티브 Kubernetes와 완벽하게 호환되며, 네이티브 방식으로 리소스를 구매 및 관리할 수 있고, 컨테이너에서 사용한 실제 리소스 양에 따라 요금이 청구됩니다. EKS는 또한 Tencent Cloud의 스토리지 및 네트워크 제품을 확장 지원하는 동시에 사용자 컨테이너의 안전한 격리와 사용 편리성을 보장합니다.

Tencent Cloud EKS를 사용하여 GooseFS를 배포하면 EKS의 탄력적인 컴퓨팅 리소스를 최대한 활용할 수 있으며, 초 단위로 과금되는 주문형 Cloud Object Storage(COS) 스토리지 액세스 가속화 서비스를 구축할 수 있습니다.

아키텍처 설명

다음 이미지는 Tencent Cloud EKS를 사용하여 GooseFS를 배포하는 일반적인 아키텍처를 보여줍니다.



그림에 표시된 것처럼 전체 아키텍처는 EKS 호스팅 컴포넌트, 사용자 리소스 풀 및 COS 스토리지 세 부분으로 구성됩니다. 그 중 사용자 리소스 풀은 주로 GooseFS 클러스터를 구축하는데 사용되고, COS 스토리지는 원격 스토리지

©2013-2022 Tencent Cloud. All rights reserved.

시스템으로 사용되며, 퍼블릭 클라우드 스토리지 서비스인 클라우드 HDFS의 대체도 지원됩니다. 구체적인 구축 과정은 다음과 같습니다.

- GooseFS Master와 Worker 모두 Kubernetes Statefulset 유형으로 리소스를 배포합니다.
- Fluid를 사용하여 GooseFS 클러스터를 풀업합니다.
- Fuse Client는 사용자 Pod의 샌드박스(Sandbox)에 통합됩니다.
- 사용 방법은 표준 Kubernetes와 일치합니다.

작업 단계

환경 준비

1. EKS 클러스터 생성 구체적인 작업은 클러스터 생성을 참고하십시오.
2. 클러스터 액세스를 활성화하고 실제 상황에 따라 내부 네트워크 또는 외부 네트워크를 선택합니다. 구체적인 설명은 클러스터 연결을 참고하십시오.
3. `kubectl get ns` 명령을 실행하여 클러스터를 사용할 수 있는지 확인합니다.

```
-&gt; goosefs kubectl get ns
NAME STATUS AGE
default Active 7h31m
kube-node-lease Active 7h31m
kube-public Active 7h31m
kube-system Active 7h31m
```

4. 'helm'을 가져오는 방법은 Helm 공식 문서를 참고하십시오.

GooseFS 설치

1. `helm install` 명령을 입력하여 chart 패키지를 설치하고, fluid를 설치합니다.

```
-&gt; goosefs helm install fluid ./charts/fluid-on-tke
NAME: fluid
LAST DEPLOYED: Tue Jul 6 17:41:20 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
```

2. `fluid` 와 관련된 pod의 상태를 확인합니다.

```
-&gt; goosefs kubectl -n fluid-system get pod
NAME READY STATUS RESTARTS AGE
alluxioruntime-controller-78877d9d47-p2pv6 1/1 Running 0 59s
```

©2013-2022 Tencent Cloud. All rights reserved.

```
dataset-controller-5f565988cc-wnp7l 1/1 Running 0 59s
goosefsruntime-controller-6c55b57cd6-hr78j 1/1 Running 0 59s
```

3. `dataset` 을 생성하고 실제 필요에 따라 관련 변수를 수정한 다음 `kubectl apply -f dataset.yaml` 명령을 실행하여 `dataset` 을 적용합니다.

```
apiVersion: data.fluid.io/v1alpha1
kind: Dataset
metadata:
name: ${dataset-name}
spec:
mounts:
- mountPoint: cosn://${bucket-name}
name: ${dataset-name}
options:
fs.cosn.userinfo.secretKey: XXXXXXX
fs.cosn.userinfo.secretId: XXXXXXX
fs.cosn.bucket.region: ap-${region}
fs.cosn.impl: org.apache.hadoop.fs.CosFileSystem
fs.AbstractFileSystem.cosn.impl: org.apache.hadoop.fs.CosN
fs.cos.app.id: ${user-app-id}
```

4. `GooseFS` 클러스터를 생성하고 다음 yaml을 사용하여 `kubectl apply -f runtime.yaml` 을 실행합니다.

```
apiVersion: data.fluid.io/v1alpha1
kind: GooseFSRuntime
metadata:
name: slice1
annotations:
master.goosefs.eks.tencent.com/model: c6
worker.goosefs.eks.tencent.com/model: c6
spec:
replicas: 6 # worker 수량. 컨트롤러가 확장을 지원하지만 goosefs는 현재 데이터의 자동 re-balance를 지원하지 않습니다.
data:
replicas: 1 # goosefs 데이터 복제본 수
goosefsVersion:
imagePullPolicy: Always
image: ccr.ccs.tencentyun.com/cosdev/goosefs # goosefs 클러스터에서 사용하는 미러 이미지 및 버전
imageTag: v1.0.1
tieredstore:
levels:
- mediumtype: MEM # 메모리, 고효율 클라우드 디스크, SSD 클라우드 디스크에 각각 해당하는 MEM, HDD, SSD 지원
```

©2013-2022 Tencent Cloud. All rights reserved.

```
path: /data
quota: 5G # 메모리와 클라우드 디스크가 모두 적용되며, 클라우드 디스크는 최소 10G입니다.
high: "0.95"
low: "0.7"
properties:
goosefs.user.streaming.data.timeout: 5s
goosefs.job.worker.threadpool.size: "22"
goosefs.master.journal.type: UFS # UFS 또는 EMBEDDED. 단일 master는 UFS 사용
# goosefs.worker.network.reader.buffer.size: 128MB
goosefs.user.block.size.bytes.default: 128MB
# goosefs.user.streaming.reader.chunk.size.bytes: 32MB
# goosefs.user.local.reader.chunk.size.bytes: 32MB
goosefs.user.metrics.collection.enabled: "false"
goosefs.user.metadata.cache.enabled: "true"
goosefs.user.metadata.cache.expiration.time: "2day"
master:
# POD에 해당하는 가상 머신의 사양을 설정합니다. 매개변수는 필수 사항이며, 미입력 시 기본값은 1c1g입니다.
resources:
requests:
cpu: 8
memory: "16Gi"
limits:
cpu: 8
memory: "16Gi"
replicas: 1
# journal:
# volumeType: pvc
# storageClass: goosefs-hdd
jvmOptions:
- "-Xmx12G"
- "-XX:+UnlockExperimentalVMOptions"
- "-XX:ActiveProcessorCount=8"
- "-Xms10G"
worker:
jvmOptions:
- "-Xmx28G"
- "-Xms28G"
- "-XX:+UnlockExperimentalVMOptions"
- "-XX:MaxDirectMemorySize=28g"
- "-XX:ActiveProcessorCount=8"
resources:
requests:
cpu: 16
memory: "32Gi"
limits:
cpu: 16
```

©2013-2022 Tencent Cloud. All rights reserved.

```
memory: "32Gi"
fuse:
jvmOptions:
- "-Xmx4G"
- "-Xms4G"
- "-XX:+UseG1GC"
- "-XX:MaxDirectMemorySize=4g"
- "-XX:+UnlockExperimentalVMOptions"
- "-XX:ActiveProcessorCount=24"
```

5. 클러스터 상태 및 PVC 상태를 확인합니다.

```
-&gt; goosefs kubectl get pod
NAME READY STATUS RESTARTS AGE
slice1-master-0 2/2 Running 0 8m8s
slice1-worker-0 2/2 Running 0 8m8s
slice1-worker-1 2/2 Running 0 8m8s
slice1-worker-2 2/2 Running 0 8m8s
slice1-worker-3 2/2 Running 0 8m8s
slice1-worker-4 2/2 Running 0 8m8s
slice1-worker-5 2/2 Running 0 8m8s
-&gt; goosefs kubectl get pvc
slice1 Bound default-slice1 100Gi ROX fluid 7m37s # PVC 이름은 dataset 이름과 같고, 100Gi는 플레이스홀더로 사용하는 가상 값입니다.
```

데이터 로딩

데이터를 미리 로딩하려면 다음 yaml을 사용하여 resource를 생성하기만 하면 됩니다. yaml의 예시는 `kubectl apply -f dataload.yaml` 입니다. 실행 후 응답 예시는 다음과 같습니다.

```
apiVersion: data.fluid.io/v1alpha1
kind: DataLoad
metadata:
name: slice1-dataload
spec:
# 데이터 로딩을 실행해야 하는 dataset 정보 설정
dataset:
name: slice1
namespace: default
```

생성 후 `kubectl get dataload slice1-dataload` 를 통해 상태를 관찰할 수 있습니다.

비즈니스 Pod 마운트 PVC

사용자 서비스 컨테이너는 k8s 표준 사용법에 따라 사용하며, 자세한 내용은 Kubernetes 공식 문서를 참고하십시오.

©2013-2022 Tencent Cloud. All rights reserved.

GooseFS 클러스터 폐기

GooseFS 클러스터는 `delete` 명령을 통해 폐기할 수 있으며, master 노드와 worker 노드를 삭제하도록 지정할 수 있습니다. **이 작업은 고위험 작업입니다. 비즈니스 pod에 Goosefs에 대한 IO 작업이 없는지 확인한 후 진행하십시오.**

```
-> goosefs kubectl get sts
NAME READY AGE
slice1-master 1/1 14m
slice1-worker 6/6 14m
-> goosefs kubectl delete sts slice1-master slice1-worker
statefulset.apps "slice1-master" deleted
statefulset.apps "slice1-worker" deleted
```

©2013-2022 Tencent Cloud. All rights reserved.

Docker를 통한 배포

최종 업데이트 날짜: : 2021-12-16 14:15:31

본문은 주로 Docker를 통해 GooseFS 배포하는 방법에 대해 설명합니다.

준비 사항

1. Docker 19.03.14 혹은 그 이상의 버전을 설치합니다.
2. CentOS 7 버전 이상이어야 합니다.
3. GooseFS docker 이미지를 획득해야 합니다. 예: goosefs:v1.0.0

설치 방법

1. ufs 디렉터리를 생성하고 로컬 기기의 디렉터리를 GooseFS 루트 디렉터리에 마운트합니다.

```
mkdir /tmp/goosefs_ufs
```

2. master 프로세스를 실행합니다.

```
docker run -d --rm \
--net=host \
--name=goosefs-master \
-v /tmp/goosefs_ufs:/opt/data \
-e GOOSEFS_JAVA_OPTS=" \
-Dgoosefs.master.hostname=localhost \
-Dgoosefs.master.mount.table.root.ufs=/opt/data" \
goosefs:v1.0.0 master
```

설명

- -Dgoosefs.master.hostname: master 주소를 설정 합니다.
- -Dgoosefs.master.mount.table.root.ufs: GooseFS 루트 디렉터리 마운트 포인트를 설정합니다.
- -v /tmp/goosefs_ufs:/opt/data: 로컬 디렉터리를 docker 컨테이너 안에 매핑합니다.
- --net=host: docker가 host 네트워크를 사용합니다.

©2013-2022 Tencent Cloud. All rights reserved.

3. worker 프로세스를 실행합니다.

```
docker run -d --rm \
--net=host \
--name=goosefs-worker1 \
--shm-size=1G \
-e GOOSEFS_JAVA_OPTS=" \
-Dgoosefs.worker.memory.size=1G \
-Dgoosefs.master.hostname=localhost" \
goosefs:v1.0.0 worker
```

작업 예시

1. 컨테이너 확인.

```
[root@VM-0-7-centos ~]# docker ps | grep goosefs
0bda1cac76f4 goosefs:v1.0.0 "/entrypoint.sh mast…" 32 minutes ago Up 32 minutes g
oosefs-master
b6260f9a0134 goosefs:v1.0.0 "/entrypoint.sh work…" About an hour ago Up About an
hour goosefs-worker1
```

2. 컨테이너로 이동.

```
docker exec -it 0bda1cac76f4 /bin/bash
```

3. COS 디렉터리 마운트.

```
goosefs fs mount --option fs.cosn.userinfo.secretId={secretId} \
--option fs.cosn.userinfo.secretKey={secretKey} \
--option fs.cosn.bucket.region=ap-beijing \
--option fs.cosn.impl=org.apache.hadoop.fs.CosFileSystem \
--option fs.AbstractFileSystem.cosn.impl=org.apache.hadoop.fs.CosN \
/cosn {cos버킷}
```

4. 디렉터리 확인.

©2013-2022 Tencent Cloud. All rights reserved.

```
[goosefs@VM-0-7-centos goosefs-1.0.0-SNAPSHOT-noUI-noHelm]$ goosefs fs ls /
drwxrwxrwx goosefs goosefs 1 PERSISTED 01-01-1970 08:00:00:000 DIR /cosn
drwxr-xr-x root root 0 PERSISTED 06-25-2021 11:01:24:000 DIR /my
```

5. worker 노드 확인.

```
[goosefs@VM-0-7-centos goosefs-1.0.0-SNAPSHOT-noUI-noHelm]$ goosefs fsadmin repor
t capacity
Capacity information for all workers:
Total Capacity: 1024.00MB
Tier: MEM Size: 1024.00MB
Used Capacity: 0B
Tier: MEM Size: 0B
Used Percentage: 0%
Free Percentage: 100%
Worker Name Last Heartbeat Storage MEM
172.31.0.7 0 capacity 1024.00MB
used 0B (0%)
```

©2013-2022 Tencent Cloud. All rights reserved.

OPS 가이드
로그 가이드
GooseFS 로그 소개

최종 업데이트 날짜: : 2021-08-03 15:18:58

GooseFS의 Master와 Worker 노드 및 Spark 등 컴퓨팅 프레임워크가 GooseFS Client를 통해 GooseFs에게 요청을 진행할 경우, 요청 로그에 모두 기록됩니다. 사용자는 해당 출력 로그를 분석하여 문제를 진단할 수 있습니다. GooseFS 로그 출력은 log4j에 기반하여 구현되며, log4j.properties 설정 파일을 변경하여 로그 저장 경로, 로그 레벨, RPC 호출 상황 기록 여부 등의 GooseFS 로그 출력을 설정할 수 있습니다. 사용자는 GooseFS의 설정 파일 디렉터리의 log4j.properties 파일을 수정할 수 있습니다.

```
$ 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
...
```

GooseFS의 로그 설정 상세 설명은 아래와 같습니다.

로그 저장 위치

GooseFS가 수집한 로그는 기본적으로 ${GOOSEFS_HOME}/logs 디렉터리에 저장됩니다. 그중, Master가 수집한 로그는 logs/master.log에 저장되고, Worker가 수집한 로그는 logs/worker.log에 저장됩니다. 주의할 것은, 노드 프로세스에서 발생된 오류에 대한 로그는 master.out 또는 worker.out에 저장되고, 일반적인 상황에서 이 두 종류의 파일은 빈 파일입니다. 시스템에서 오류가 발생되었을 시, 문제 추적을 위해 오류를 기록합니다.

Master 노드의 로그 기록 설정을 예로 든다면, 자주 사용하는 설정들은 아래와 같습니다.

```
# 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
```

©2013-2022 Tencent Cloud. All rights reserved.

```
log4j.appender.MASTER_LOGGER.layout.ConversionPattern=%d{ISO8601} %-5p %c{1} - %
m%n
```

매개변수 설정 설명은 다음과 같습니다.

- MASTER_LOGGER: MASTER의 로그 출력 설정
- MASTER_LOGGER.File: 로그의 저장 경로 지정. 경로 수정을 통해 로그 저장 위치를 사용자 정의할 수 있습니다.
- MASTER_LOGGER.MaxFileSize: 단일 로그 파일에 대한 크기 제한 설정
- MASTER_LOGGER.MaxBackupIndex: 로그 파일 수량 제한 설정
- MASTER_LOGGER.layout: 로그 출력 형식 템플릿 지정
- MASTER_LOGGER.layout.ConversionPattern: 로그 출력의 상세 형식 지정

주의：

-.log 파일은 롤링 방식으로 저장되며 UFS에 백업할 수 있습니다. .out 파일 같은 경우, 롤링 방식으로 저장되지 않기 때문에 .out 파일을 제거해야 하는 경우 수동으로 삭제 작업을 진행해야 합니다.

- log4j의 자세한 매개변수 설정은 log4j configuration 문서를 참고하십시오.
- GooseFS는 자체 생성된 로그만을 저장합니다. 상위 레이어의 컴퓨팅 애플리케이션에서 생성된 로그는 컴퓨팅 애플리케이션의 로그 설정에 따라 로그 저장 위치를 확인할 수 있습니다. 자주 사용되는 컴퓨팅 애플리케이션 로그 설정 정보는 다음과 같습니다. Apache Hadoop, Apache HBase, Apache Hive, Apache Spark。

로그 레벨

GooseFS는 아래와 같은 5개 레벨의 로그를 제공합니다.

- TRACE: 상세 호출 로그. 메소드와 클래스 호출의 디버깅에 사용됩니다.
- DEBUG: 비교적 상세한 호출 로그. DEBUG 과정 중의 문제를 진단할 수 있습니다.
- INFO: 요청 처리 과정 중의 주요 정보.
- WARN: 알람류 정보. 작업은 계속 실행될 수 있지만 발생 가능한 문제에 주의해야 합니다.
- ERROR: 시스템 오류 보고 정보. 작업 진행에 영향을 미칩니다.

위의 5개 레벨의 로그에 대한 상세 수준은 높음에서 낮음 순이며, 높은 레벨의 로그는 낮은 레벨의 로그 정보도 기록합니다. 기본적으로 GooseFS는 INFO 레벨의 로그를 구성하고 INFO, WARN, ERROR 3가지 레벨의 로그를 기록합니다.

GooseFS의 설정 파일 디렉터리로 이동하여 log4j.properties 파일을 열고 수정할 수 있습니다. 아래의 예시는 모든 GooseFS의 로그 레벨을 DEBUG 레벨로 수정하는 방법을 보여줍니다.

©2013-2022 Tencent Cloud. All rights reserved.

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

지정된 유형의 로그 레벨을 수정해야 하는 경우 설정 파일에 선언문을 추가할 수 있습니다. 아래의 예시는 GooseFSFileInStream 클래스의 로그 레벨이 DEBUG임을 보여줍니다.

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

일반적으로 로그 설정 파일에서 로그 레벨을 수정하는 것이 좋습니다. 그러나 일부 특정 시나리오에서는 사용자가 클러스터 작업 중 로그 매개변수를 수정해야 할 수 있으며 이때 명령 라인에 goosefs logLevel 명령을 입력하여 수정할 수 있습니다. 다음은 logLevel에서 지원하는 설정 옵션입니다.

```
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.Default
FileSystemMaster) 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
```

각 설정 항목의 상세 설명은 다음과 같습니다.

- level：로그 레벨. TRACE, DEBUG, INFO, WARN, ERROR 다섯 가지의 레벨.
- logName：로그 출력 logger. 예: com.qcloud.cos.goosefs.underfs.hdfs.HdfsUnderFileSystem 등.
- target: 수정 범위. Master 또는 Worker 노드(IP: PORT 방식으로 지정)를 지정하여 설정할 수 있으며 기본 값은 Master 및 모든 Worker 노드입니다.

사용자는 시스템 작동 중 문제를 해결하기 위해 필요에 따라 로그 레벨을 조정할 수 있습니다. 예를 들어 다음 예는 모든 Worker 노드의 com.qcloud.cos.goosefs.underfs.hdfs.HdfsUnderFileSystem 클래스의 로그 레벨을 DEBUG 레벨로 변경하고 디버깅이 완료된 후 다시 INFO 레벨로 수정하는 것을 보여줍니다.

```
$ goosefs logLevel --logName=com.qcloud.cos.goosefs.underfs.hdfs.HdfsUnderFileSys
tem --target=workers --level=DEBUG # DEBUG 모드로 변경
$ goosefs logLevel --logName=com.qcloud.cos.goosefs.underfs.hdfs.HdfsUnderFileSys
tem --target=workers --level=INFO # NFO 모드로 변경
```

고급 설정

GooseFS는 GC 이벤트 로그, FUSE 인터페이스 로그, RPC 호출 로그, UFS 작업 로그, 로그 분할 및 로그 필터링과 같은 작업의 설정을 지원합니다. 다음은 몇 가지 일반적인 고급 설정을 사용하는 방법을 설명합니다.

©2013-2022 Tencent Cloud. All rights reserved.

- **GC 이벤트 로그**
  GooseFS는 .out 파일에 GC 이벤트 로그를 기록합니다. conf/goosefs-env.sh에서 다음과 같은 설정을 추가할 수 있습니다.

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

  GOOSEFS_JAVA_OPTS는 모든 유형의 GooseFS 노드에 대한 Java 가상 머신 매개변수입니다. GOOSEFS_MASTER_JAVA_OPTS 및 GOOSEFS_WORKER_JAVA_OPTS를 지정하여 Master 및 Worker에 대한 가상 머신 매개변수를 각각 지정할 수도 있습니다.

- **FUSE 인터페이스 로그**
  FUSE 로그 레벨의 기록은 conf/log4j.properties 파일에서 설정할 수 있습니다.

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

  활성화한 후 FUSE 인터페이스 로그는 logs/fuse.log에서 볼 수 있습니다.

- **RPC 호출 로그 활성화**
  GooseFS는 conf/log4j.properties 구성 파일에서 Client 또는 Master의 RPC 호출 로그 설정을 지원합니다. log4j.properties 파일을 통해 클라이언트가 RPC 요청 로그를 출력하도록 설정할 수 있습니다.

```
log4j.logger.com.qcloud.cos.goosefs.client.file.FileSystemMasterClient=DEBUG # Client와 FileSystemMaster 간의 RPC 요청 로그
log34j.logger.com.qcloud.cos.goosefs.client.block.BlockSystemMasterClient=DEBUG # Client와 BlockMaster 간의 RPC 요청 로그
```

  logLevel 명령을 통해 Master가 RPC 요청 로그를 출력하도록 설정할 수 있습니다.

```
$ goosefs logLevel \--logName=com.qcloud.cos.goosefs.master.file.FileSystemMasterClientServiceHandler \--target master --level=DEBUG # 파일 관련 RPC 요청 로그
$ goosefs logLevel \--logName=com.qcloud.cos.goosefs.master.block.BlockSystemMasterClientServiceHandler \--target master --level=DEBUG # 블록 관련 RPC 요청 로그
```

- **UFS 작업 로그**
  UFS 작업 로그 출력 설정은 log4j.properties 파일에 설정 항목을 추가하거나 logLevel 명령을 사용하여 설정할 수 있습니다. 다음은 logLevel 명령의 예시입니다.

©2013-2022 Tencent Cloud. All rights reserved.

```
$ goosefs logLevel \--logName=com.qcloud.cos.goosefs.underfs.UnderFileSystemWit
hLogging \--target master --level=DEBUG # Master 노드의 UFS에 대한 작업 로그 기록
$ goosefs logLevel \--logName=com.qcloud.cos.goosefs.underfs.UnderFileSystemWit
hLogging \--target workers --level=DEBUG # Worker 노드의 UFS에 대한 작업 로그 기록
```

- **로그 분할**
  GooseFS는 지정된 유형의 로그를 지정된 저장 경로에 저장할 수 있도록 지원하며, 만약 모든 로그가 .log 파일에 기록되면 다음과 같은 문제가 발생할 수 있습니다.
  - 클러스터 크기가 크고 처리량이 클 경우 master.log 또는 worker.log 파일이 비정상적으로 커지거나 롤링에 의해 많은 수의 로그 파일이 생성될 수 있습니다.
  - 로그 정보가 많아 대상 분석을 위한 타깃성 로그 분석에 적합하지 않습니다.
  - 많은 양의 로그가 로컬 노드에 저장되어 공간을 차지합니다.

  따라서 서비스의 필요에 따라 log4j.properties 파일에 설정 항목을 추가하여 지정된 클래스에 의해 생성된 로그를 지정된 파일 경로에 저장합니다. 다음 예시는 Statelock.log에 StateLockManager 클래스 로그를 저장하는 방법을 보여줍니다.

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

- **로그 필터링**
  GooseFS는 전체 로그를 기록하는 대신 특정 조건에 따라 필터링 및 기록을 지원합니다. 예를 들어 성능 테스트 시 RPC 호출 로그를 기록해야 하지만, 서비스에서는 모든 RPC 호출 로그를 기록할 필요는 없고 대기 시간이 긴 일부 로그만 기록하면 됩니다. 이때 설정 항목을 추가할 수 있습니다. log4j.properties 파일에서 로그 필터링 조건 설정을 추가하기만 하면 필터링이 가능합니다. 다음은 RPC 호출 딜레이가 200ms를 초과하고 FUSE 호출이 1초를 초과하는 요청의 필터링을 보여줍니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

모니터링 가이드
GooseFS 모니터링 지표 가져오기

최종 업데이트 날짜: : 2022-03-09 18:06:01

Goosefs는 Coda Hale Metrics Library를 기반으로 모니터링 데이터를 기록하고 명령 라인, 콘솔, 파일 등 여러 경로를 통한 지표 획득을 지원합니다. 현재 지원되는 지표 획득 방법은 다음과 같습니다.

- MetricsServlet: Json 형식으로 사용자에게 모니터링 지표를 제공합니다.
- CsvSink: CSV 파일을 통해 모니터링 지표를 표시합니다. 설정 후 모니터링 지표를 기록하는 CSV 파일은 주기적으로 생성됩니다.
- PrometheusMetricsServlet: Prometheus에서 정의한 형식으로 사용자에게 모니터링 지표를 제공합니다.

상기 모니터링 지표의 설정은 구성 파일을 통해 지정할 수 있습니다. GooseFS 모니터링 지표 구성 파일의 기본 파일 경로는 `$ GooseFS_HOME/conf/metrics.properties` 이며, goosefs.metrics.conf.file을 통한 사용자 정의 모니터링 구성 파일 지정을 지원합니다. GooseFS는 설정 가능한 모든 속성을 포함하는 기본 템플릿 metrics.properties.template을 사용자에게 제공합니다.

모니터링 지표 가져오기

다음은 모니터링 지표를 얻는 세 가지 기본 경로를 소개합니다.

1. JSON 형식으로 모니터링 지표 가져오기

GooseFS가 모니터링 지표를 얻는 기본 방법은 MetricsServlet 설정 항목에 해당하는 JSON 형식으로 풀링하는 것입니다. 명령 라인에서 GooseFS의 Leading Master 노드에 대한 HTTP 요청을 제안하여 필요한 모니터링 지표를 풀링합니다. 그 중 master의 metrics 포트는 9201이고 worker의 metrics 포트는 9204입니다. 요청 명령어 형식은 다음과 같습니다.

```
$ curl <LEADING_MASTER_HOSTNAME>:<MASTER_WEB_PORT>/metrics/json
```

상기 예시에서 은 올바른 MASTER 노드의 IP여야 하고 는 활성화된 포트여야 합니다.

WORKER 노드의 모니터링 지표를 가져와야 하는 경우 다음과 같은 방법으로 얻을 수 있습니다.

```
$ curl <WORKER_HOSTNAME>:<WORKER_WEB_PORT>/metrics/json
```

2. CSV 파일로 모니터링 지표 가져오기

©2013-2022 Tencent Cloud. All rights reserved.

GooseFS는 CSV 형식 파일로 데이터 내보내기를 지원합니다. 이 기능을 통해 모니터링 지표를 얻으려면 먼저 모니터링 지표를 저장할 디렉터리를 준비해야 합니다.

```
$ mkdir /tmp/goosefs-metrics
```

스토리지 경로를 준비한 후 구성 파일 conf/metrics.properties를 수정하여 CsvSink 능력을 활성화합니다.

```
sink.csv.class=goosefs.metrics.sink.CsvSink # CsvSink 기능 활성화
sink.csv.period=1 # 모니터링 지표 내보내기 주기 설정
sink.csv.unit=senconds # 모니터링 지표 내보내기 주기 단위 설정
sink.csv.directory=/tmp/goosefs-metrics # 모니터링 지표 내보내기 경로 설정
```

설정이 완료된 후 설정을 적용하려면 노드를 재시작해야 합니다. 설정이 적용된 후 모니터링 지표는 주기적으로 CSV 형식으로 내보내기되고 지정된 경로에 저장됩니다.

주의 :

- GooseFS는 모니터링 설정 템플릿을 준비했습니다. conf/metrics.properties.template 파일을 참고하십시오.
- GooseFS가 클러스터에 배포된 경우 지정된 지표 스토리지 경로를 모든 노드에서 읽을 수 있도록 보장되어야 합니다.

3. Prometheus 모니터링 지표 풀링

GooseFS master와 worker의 Prometheus 모니터링 지표는 다음과 같은 명령어로 조회할 수 있습니다. 그 중 master의 metrics 포트는 9201이며, worker의 metrics 포트는 9204입니다.

```
curl <LEADING_MASTER_HOSTNAME>:<MASTER_WEB_PORT>/metrics/prometheus/
# HELP Master_CreateFileOps Generated from Dropwizard metric import (metric=Maste
r.CreateFileOps, type=com.codahale.metrics.Counter)
...
curl <WORKER_IP>:<WOKER_PORT>/metrics/prometheus/
# HELP pools_Code_Cache_max Generated from Dropwizard metric import (metric=pool
s.Code-Cache.max, type=com.codahale.metrics.jvm.MemoryUsageGaugeSet$$Lambda$51/13
7460818)
...
```

©2013-2022 Tencent Cloud. All rights reserved.

Prometheus를 기반으로 GooseFS 모니터링 시스템 구축

최종 업데이트 날짜: : 2022-03-09 16:58:15

Goosefs는 설정을 통해 지표 데이터를 Prometheus와 같은 다른 모니터링 시스템으로 출력할 수 있습니다. Prometheus는 오픈 소스 모니터링 프레임워크로 현재 Tencent Cloud 모니터링은 Prometheus를 통합하였습니다. 다음으로 Goosefs의 모니터링 지표와 자체구축한 Prometheus와 클라우드 Prometheus에 모니터링 지표를 보고하는 과정을 중점적으로 소개합니다.

준비 작업

Prometheus를 통해 모니터링 시스템을 구축하려면 다음 준비 작업이 선행되어야 합니다.

- GooseFS 클러스터 설정
- Prometheus 공식 홈페이지의 설치 패키지 또는 Tencent Cloud Prometheus 설치 패키지 다운로드
- Grafana 다운로드 및 설정

GooseFS 모니터링 지표 보고 설정 활성화

1. GooseFs를 편집하여 conf/goosefs-site.properties를 설정합니다. 다음과 같은 설정 항목을 추가하고 goosefs copyDir conf/를 모든 worker 노드로 복사하며 `./bin/goosefs-start.sh all` 클러스터를 재시작합니다.

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

2. master와 worker의 Prometheus 모니터링 지표는 다음과 같은 명령어로 조회할 수 있습니다. 그중 master의 metrics 포트는 9201이며, worker의 metrics 포트는 9204입니다.

```
curl <LEADING_MASTER_HOSTNAME>:<MASTER_WEB_PORT>/metrics/prometheus/
# HELP Master_CreateFileOps Generated from Dropwizard metric import (metric=Maste
r.CreateFileOps, type=com.codahale.metrics.Counter)
...
curl <WORKER_IP>:<WOKER_PORT>/metrics/prometheus/
# HELP pools_Code_Cache_max Generated from Dropwizard metric import (metric=pool
s.Code-Cache.max, type=com.codahale.metrics.jvm.MemoryUsageGaugeSet$$Lambda$51/13
```

©2013-2022 Tencent Cloud. All rights reserved.

```
7460818)
...
```

자체구축 Prometheus에 모니터링 지표 보고

1. Prometheus 설치 패키지를 다운로드 및 압축 해제하고, prometheus.yml을 수정합니다.

```
# 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. targets/cluster/masters/masters.yml을 생성하여 master의 IP와 port를 추가합니다.

```
- targets:
- "<TARGERTS_MASTER_IP>:<TARGERTS_MASTER_PORT>"
```

3. targets/cluster/workers/workers.yml을 생성하여 worker의 IP와 port를 추가합니다.

```
- targets:
- "<TARGERTS_WORKER_IP>:<TARGERTS_WORKER_PORT>"
```

4. Prometheus를 실행하고, 그중 --web.listen-address는 Prometheus 수신 주소를 지정합니다. 기본 포트 번호: 9090

©2013-2022 Tencent Cloud. All rights reserved.

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

5. 시각화 인터페이스를 조회합니다.

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

6. 기기 인스턴스를 조회합니다.

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

Tencent Cloud Prometheus에 모니터링 지표 보고

1. 설치 가이드의 가이드에 따라, master 기기에 Promethus agent를 설치합니다.

```
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. master와 worker의 캡처 작업을 설정합니다.

방식1:

```
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:
```

©2013-2022 Tencent Cloud. All rights reserved.

```
- /usr/local/services/prometheus/targets/cluster/workers/*.yml
refresh_interval: 1m
```

주의 :
job_name에는 빈칸이 없지만, 오프라인 Prometheus의 job_name에는 빈칸이 포함될 수 있습니다.

방식2:

```
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
```

주의 :
캡처 작업은 두 번째 방법에 따라 구성되며 targets/cluster/masters/경로에 masters.yml 및 workers.yml 파일을 생성할 필요가 없습니다.

Grafana를 사용하여 모니터링 지표 조회

1. Grafana를 실행합니다.

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

©2013-2022 Tencent Cloud. All rights reserved.

2. 로그인 페이지 http://<GRAFANA_IP>:<GRAFANA_PORT>를 엽니다. Grafana 기본 포트: 3000, username과 password는 모두 admin이며, 첫 로그인 후 비밀번호를 수정할 수 있습니다.
3. 페이지 이동 후 Prometheus의 Datasource를 추가합니다.

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

4. Goosefs의 Grafana 템플릿을 가져오고, 가져올 json 및 위에서 생성한 Datasource를 선택합니다(json 다운로드).

> 주의 :
> 클라우드 Prometheus 구매 시 비밀번호를 설정해야 합니다. 클라우드 Grafana의 시각화 모니터링 인터페이스 설정은 위와 유사하며, job_name은 일치하도록 설정해야 합니다.

5. DashBoard 수정 후 DashBoard 내보내기가 가능합니다.

©2013-2022 Tencent Cloud. All rights reserved.

클러스터 설정 사례

최종 업데이트 날짜: : 2022-05-05 14:22:45

GooseFS는 다양한 배포 방법을 제공합니다. 본 문서에서는 빅 데이터 시나리오에서 일반적으로 사용되는 클러스터 모드 배포 방법을 소개합니다.

- AI 시나리오 프로덕션 환경 구성 사례
- 빅 데이터 시나리오 프로덕션 환경 구성 사례

©2013-2022 Tencent Cloud. All rights reserved.

데이터 보안
Apache Ranger를 통한 GooseFS 액세스 권한 제어

최종 업데이트 날짜: : 2021-11-09 16:03:20

개요

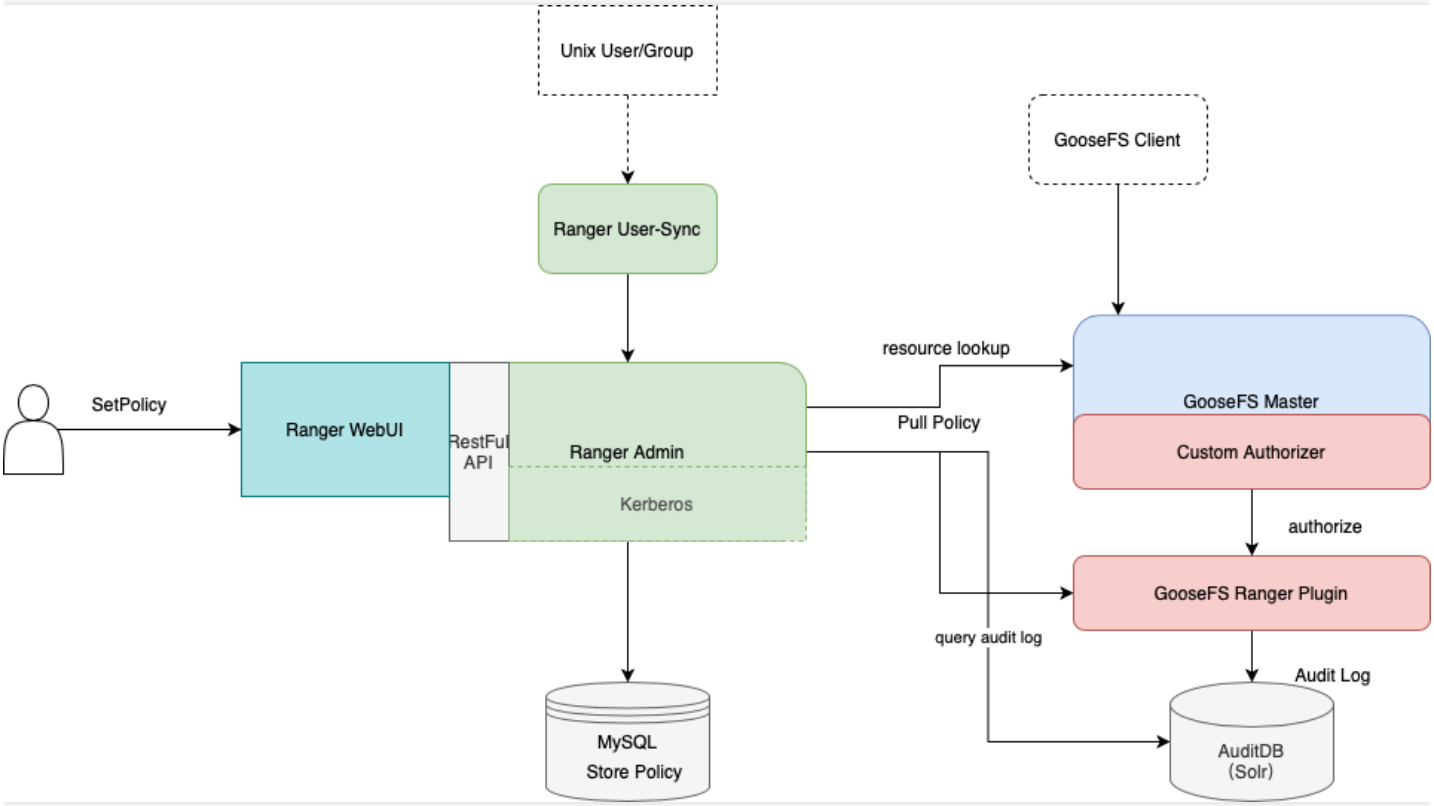
Apache Ranger는 빅 데이터 생태 시스템에서 액세스 권한을 제어하는 데 사용되는 표준 인증 모듈입니다. GooseFS는 빅 데이터 및 데이터 레이크 시나리오의 스토리지 가속화 시스템으로 Apache Ranger 통합 인증 플랫폼 액세스를 지원합니다. 본문은 Apache Ranger를 사용하여 GooseFS 리소스 액세스 권한을 제어하는 방법을 소개합니다.

강점

- GooseFS는 클라우드 네이티브 가속 스토리지 시스템으로서 Apache Ranger를 지원하여 HDFS와 거의 동일한 액세스 제어 수준에 도달했습니다. 따라서 원래 HDFS를 사용하던 빅 데이터 사용자는 매우 쉽게 GooseFS로 마이그레이션할 수 있으며 HDFS의 Ranger 권한 정책을 직접 재사용하여 일관된 사용자 경험을 얻을 수 있습니다.
- GooseFS with Ranger는 HDFS with Ranger의 인증 아키텍처와 비교하여 Ranger + 네이티브 ACL의 공동 인증 옵션도 제공합니다. Ranger 인증에 실패하면, 네이티브 ACL 인증을 사용하도록 선택할 수도 있어 일부 Ranger 인증 정책 설정의 미흡한 문제를 해결할 수 있습니다.

GooseFS with Ranger의 인증 아키텍처

©2013-2022 Tencent Cloud. All rights reserved.



GooseFS를 Ranger 인증 플랫폼에 통합할 수 있도록 GooseFS Ranger Plugin을 개발하였으며, GooseFS Master 노드와 Ranger Admin측에 동시에 배포됩니다. 아래와 같은 작업을 책임지고 완성합니다.

- GooseFS Master 노드 측:
  - 'Authorizer' 인터페이스를 제공하여 GooseFS Master의 모든 메타데이터 요청에 대한 인증 결과를 제공합니다.
  - Ranger Admin에 접속하여 사용자가 설정한 인증 정책을 획득합니다.
- Ranger Admin 측:
  - Ranger Admin에게 GooseFS의 리소스 조회(resource lookup) 기능을 제공합니다.
  - 구성 인증 기능을 제공합니다.

배포 시작

준비 과정

사용을 시작하기 전에 Ranger 관련 구성 요소(Ranger Admin 및 Ranger UserSync 포함)가 환경에 배포 및 구성되었는지, Ranger의 WebUI가 정상적으로 열리고 사용할 수 있는지 확인해야 합니다.

모듈 배포

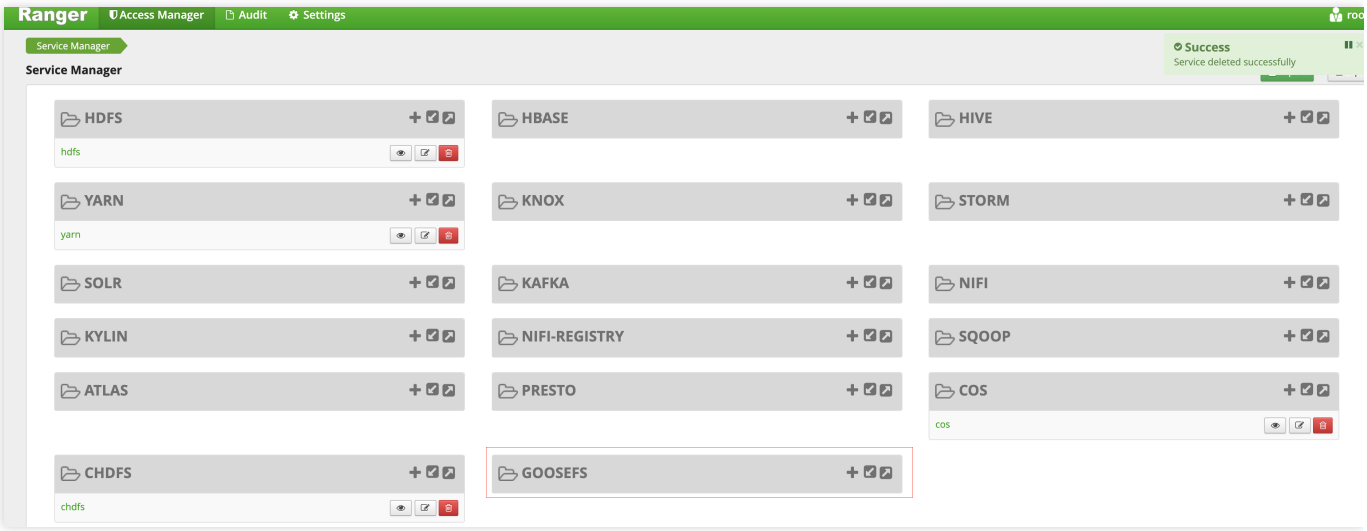
Ranger Admin 측에 GooseFS Ranger Plugin을 배포하고 해당 서비스를 등록합니다.

©2013-2022 Tencent Cloud. All rights reserved.

설명：
여기를 클릭하여 GooseFS Ranger Plugin을 다운로드하십시오.

배포 단계는 다음과 같습니다.

1. Ranger 서비스 정의 디렉터리 하위에 새로운 GooseFS의 디렉터리를 만듭니다(디렉터리 권한은 최소한 x와 r 권한을 보장합니다).
   i. a. Tencent Cloud EMR 클러스터를 사용하는 경우 Ranger의 서비스 정의 디렉터리는 `/usr/local/service/ranger/ews/webapp/WEB-INF/classes/ranger-plugins` 에 둡니다.
   ii. 자체 구축 hadoop 클러스터의 경우 ranger 디렉터리에서 hdfs 등 이미 ranger 서비스에 액세스 된 모듈을 찾아 디렉터리 위치를 찾을 수 있습니다.



2. GooseFS 디렉터리에 goosefs-ranger-plugin-${version}.jar 및 ranger-servicedef-goosefs.json을 넣고 읽기 권한을 가집니다.

3. Ranger 서비스를 재시작합니다.

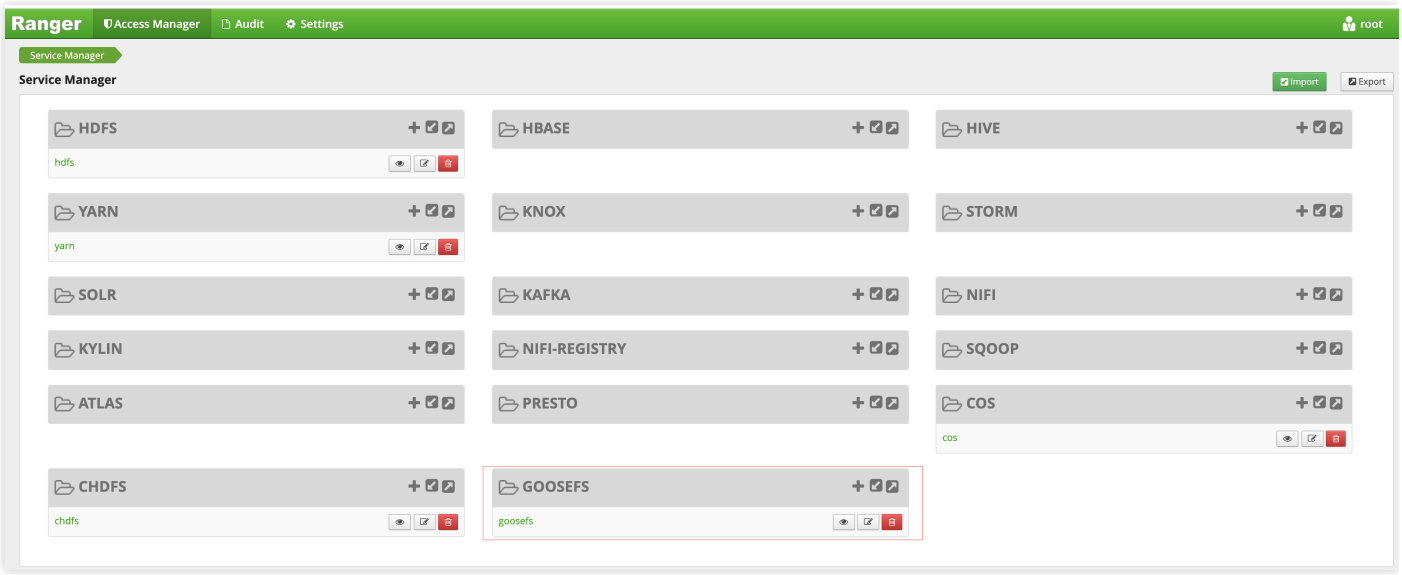
4. Ranger에서 아래 명령어에 따라 GooseFS Service에 가입합니다.

```
# 서비스를 생성하려면 Ranger 관리자 계정 및 암호, Ranger 서비스 주소를 입력해야 합니다.
# Tencent Cloud EMR 클러스터의 경우, 관리자는 root이고, 암호는 EMR 클러스터를 구축할 때 설정한 root 암호이며, ranger 서비스의 IP는 EMR 서비스의 Master IP입니다.
adminUser=root
adminPasswd=xxxx
rangerServerAddr=10.0.0.1:6080
curl -v -u${adminUser}:${adminPasswd} -X POST -H "Accept:application/json" -H
```

©2013-2022 Tencent Cloud. All rights reserved.

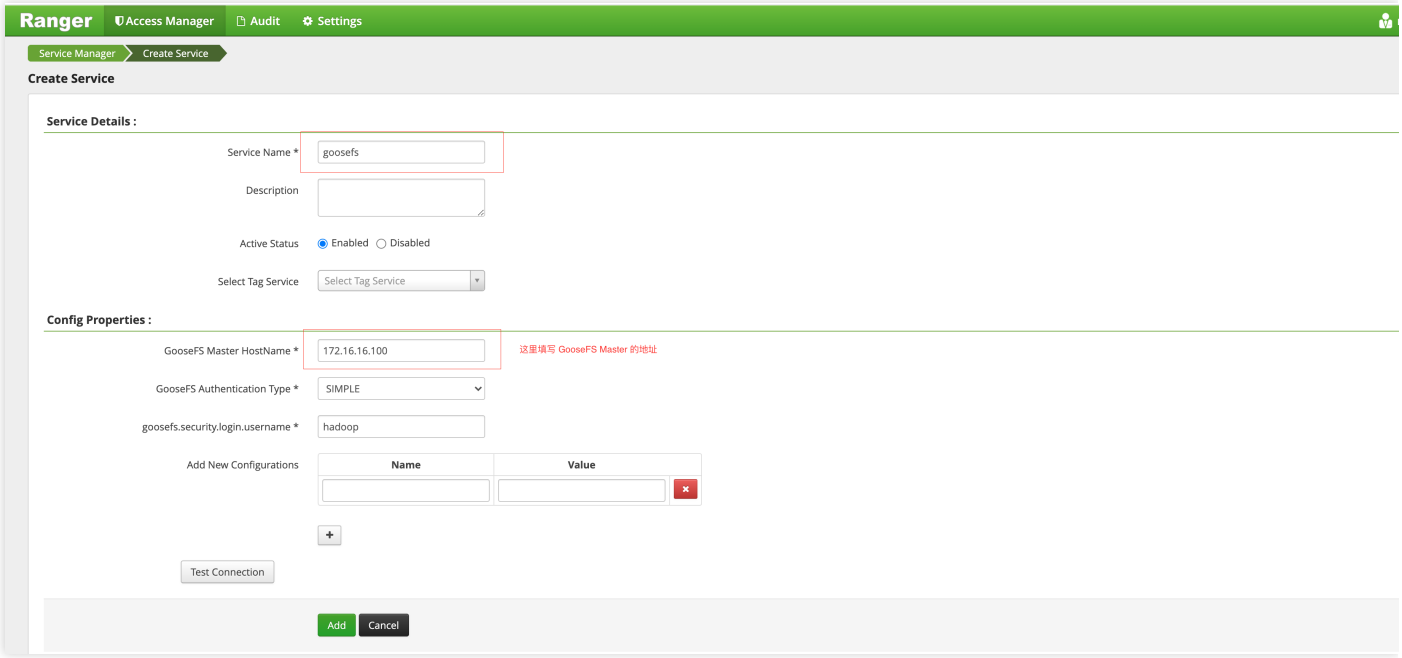
```
"Content-Type:application/json" -d @./ranger-servicedef-goosefs.json http://${r
angerServerAddr}/service/plugins/definitions
# 서비스 가입이 완료되면 서비스 ID가 반환되므로 반드시 이 ID를 기록해 두시기 바랍니다.
# GooseFS 서비스를 삭제하려면, 방금 반환된 서비스 ID를 전달하고 다음 명령을 실행합니다.
serviceId=104
curl -v -u${adminUser}:${adminPasswd} -X DELETE -H "Accept:application/json" -H
"Content-Type:application/json" http://${rangerServerAddr}/service/plugins/defi
nitions/${serviceId}
```

5. 생성에 성공하면 Ranger의 Web 콘솔에서 GooseFS 관련 서비스를 볼 수 있습니다.

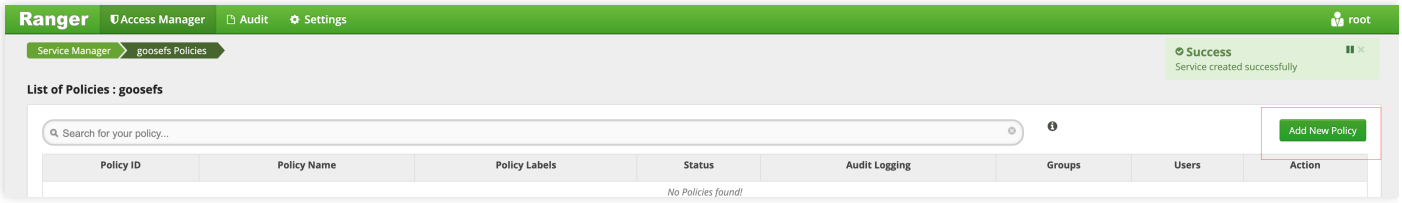


©2013-2022 Tencent Cloud. All rights reserved.

6. GooseFS 서비스 측에서 [+]를 클릭하여 goosefs 서비스 인스턴스를 정의합니다.



7. 새로 생성된 goosefs 서비스 인스턴스를 클릭하여 인증 policy를 추가합니다.



GooseFS Master측에 GooseFS Ranger 플러그인을 배포하고 Ranger 인증을 구성 및 활성화합니다.

1. goosefs-ranger-plugin-${version}.jar를 ${GOOSEFS_HOME}/lib 경로에 넣고 최소 읽기 권한을 가집니다.

2. ranger-goosefs-audit.xml, ranger-goosefs-security.xml 및 ranger-policymgr-ssl.xml 3개 파일을 `\${GOOSEFS_HOME}/conf` 경로에 넣고 필수 구성을 각각 입력합니다.

   - 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(감사 미활성화, 설정하지 않아도 됨)

- 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>
```

©2013-2022 Tencent Cloud. All rights reserved.

3. goosefs-site.xml 파일에 다음과 같은 구성을 추가합니다.

```
...
goosefs.security.authorization.permission.type=CUSTOM
goosefs.security.authorization.custom.provider.class=org.apache.ranger.authorization.goosefs.RangerGooseFSAuthorizer
...
```

4. ${GOOSEFS_HOME}/libexec/goosefs-config.sh에 goosefs-ranger-plugin-${version}.jar를 GooseFS 경로에 추가합니다.

```
...
GOOSEFS_RANGER_CLASSPATH="${GOOSEFS_HOME}/lib/ranger-goosefs-plugin-1.0.0-SNAPSHOT.jar"
GOOSEFS_SERVER_CLASSPATH=${GOOSEFS_SERVER_CLASSPATH}:${GOOSEFS_RANGER_CLASSPATH}
...
```

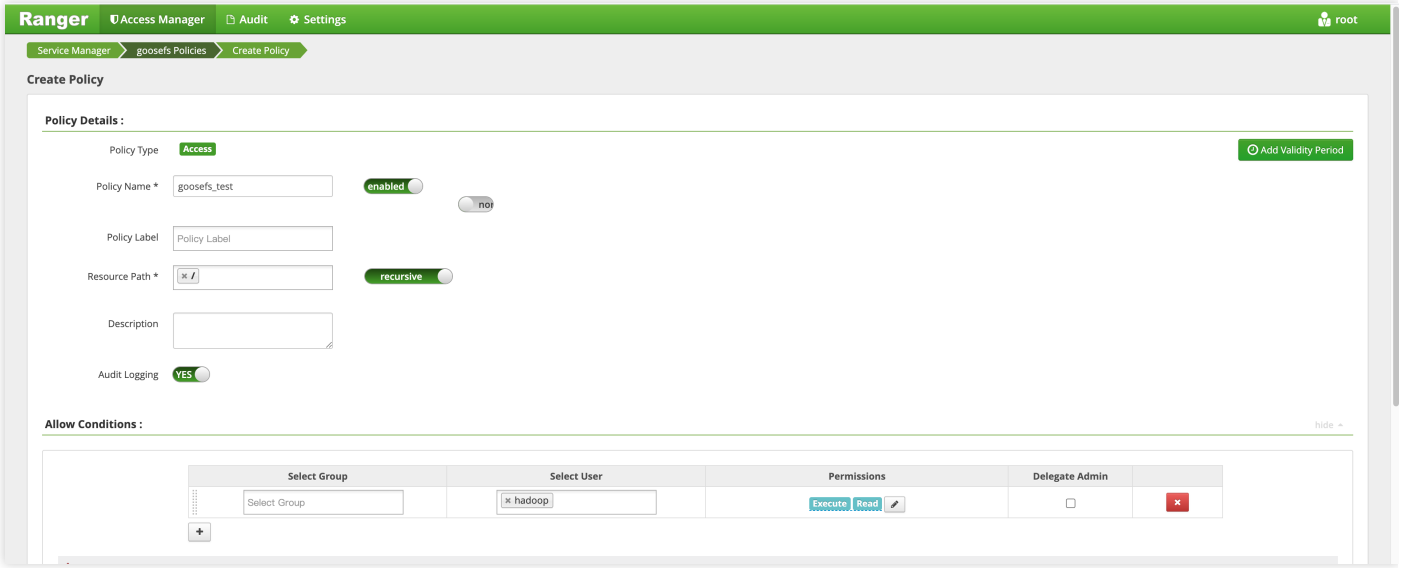
이렇게 모든 구성이 완료됩니다.

사용 인증

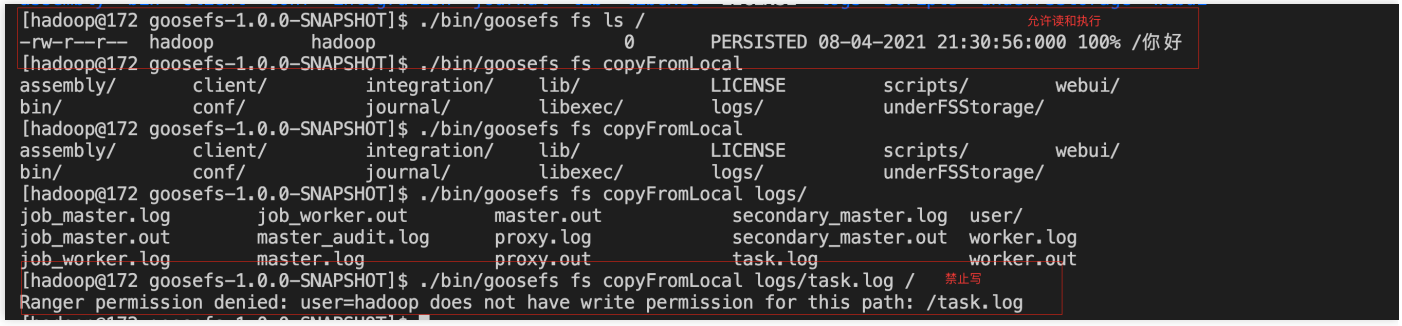
Hadoop 사용자에게 GooseFS의 루트 디렉터리에 대한 읽기 및 실행 권한은 있지만 쓰기는 허용하지 않는 정책을 추가하는 방법은 다음과 같습니다.

©2013-2022 Tencent Cloud. All rights reserved.

1. 다음과 같이 정책을 추가합니다.



2. 정책이 성공적으로 추가된 후, 정책을 인증하면 다음과 같이 정책이 적용된 것을 확인할 수 있습니다.



©2013-2022 Tencent Cloud. All rights reserved.

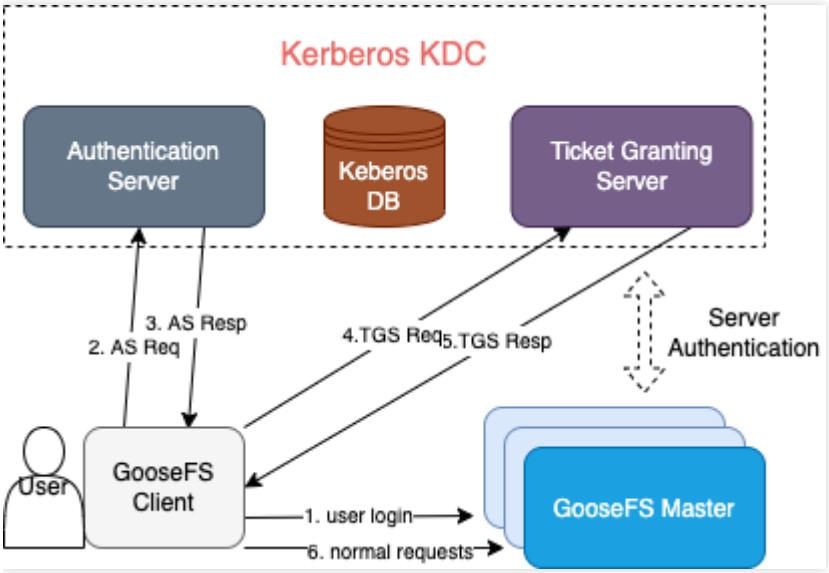
Kerberos 인증에 GooseFS 연결

최종 업데이트 날짜: : 2023-03-14 17:01:39

개요

Kerberos는 빅 데이터 에코시스템에서 널리 사용되는 통합 인증 서비스입니다. 빅 데이터 및 데이터 레이크 시나리오를 위한 스토리지 가속 서비스인 GooseFS를 사용하면 클러스터 노드와 사용자 액세스 요청을 Kerberos에 연결할 수 있습니다. 이 문서에서는 Kerberos에 대한 GooseFS 연결을 구성하는 방법과 Hadoop Delegation Token을 사용하여 인증하는 방법을 설명합니다.

Kerberos 인증 아키텍처에 GooseFS 연결



GooseFS용 Kerberos 인증의 강점

- 인증 아키텍처 및 프로세스는 기본적으로 Kerberos에 대한 HDFS 연결과 동일합니다. HDFS에서 Kerberos 인증 프로세스가 활성화된 애플리케이션은 GooseFS로 쉽게 마이그레이션할 수 있습니다.
- Hadoop Delegation Token 인증 메커니즘이 지원되므로 GooseFS는 Hadoop 기반 애플리케이션 작업과 잘 호환됩니다.

©2013-2022 Tencent Cloud. All rights reserved.

Kerberos에 대한 GooseFS 연결 구성

전제 조건

-GooseFS 1.3.0 이상.

- Kerberos KDC 서비스가 귀하의 환경에 존재하며, GooseFS 및 애플리케이션 클라이언트가 포트에 정상적으로 액세스 할 수 있는지 확인합니다.

Kerberos KDC에서 GooseFS ID 정보 생성

먼저 연결을 구성하기 전에 Kerberos KDC의 GooseFS 클러스터 Server 및 Client에 대한 Kerberos ID를 생성하십시오. 여기에서 `kadmin.local` 은 **Kerberos KDC 서버**에서 생성에 사용됩니다.

> 주의 :
> root/sudo 권한으로 `kadmin.local` 툴을 실행해야합니다.

```
$ sudo kadmin.local
```

명령이 성공적으로 실행되면 kadmin의 인터랙티브 shell 환경에 들어갑니다.

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

여기서 `kadmin.local` 은 인터랙티브 실행 환경의 명령 프롬프트를 나타냅니다.

GooseFS Server / Client ID 생성

다음은 Kerberos 구성 방법 설명을 위한 간단한 테스트 클러스터와 사용 사례 예시입니다.

1. 클러스터 환경 설명
   하나의 Master 듀얼 Worker의 Standalone 아키텍처가 사용됩니다.

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

2. `kadmin.local` 에서 Server 및 Client ID를 생성합니다.

©2013-2022 Tencent Cloud. All rights reserved.

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

주의 :
여기서 `-randkey` 를 사용하는 이유는, Server 또는 Client GooseFS 로그인 시 일반 텍스트 암호 대신 `keytab` 파일 인증을 사용합니다. password로 로그인 시 ID 정보를 사용하려면 이 필드를 제거 할 수 있습니다.

3. 각 ID에 대해 `keytab` 파일을 생성하고 내보내기합니다.

```
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
```

GooseFS Server/Client 액세스용 Kerberos 인증 구성

1. 위에서 내보내기한 `keytab` 파일을 해당 서버로 배포합니다. 여기서는 `${GOOSEFS_HOME}/conf/` 경로를 사용하는 것이 좋습니다.

```
$ 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. 해당 서버에서 Server Principal KeyTab 파일이 GooseFS Server 실행 시 사용되는 사용자/사용자 그룹에 속하는 사용자/사용자 그룹을 변경하여 GooseFS에 실행 시 필요한 읽기 권한을 부여합니다.

```
$ chown <GooseFS_USER>:<GOOSEFS_USERGROUP> goosefs_172_16_0_1.keytab
$ # Unix 액세스 권한 비트를 수정합니다
$ chmod 0440 goosefs_172_16_0_1.keytab
```

3. Client 파일 읽기 권한을 보장하기 위해 Client KeyTab 파일이 GooseFS 요청을 시작하는 클라이언트 계정에 속하는 사용자/사용자 그룹을 변경하십시오.

```
$ chown <client_user>:<client_usergroup> client_172_16_0_4.keytab
$ # Unix 액세스 권한 비트를 수정합니다
$ chmod 0440 client_172_16_0_4.keytab
```

©2013-2022 Tencent Cloud. All rights reserved.

Server 및 Client의 GooseFS 구성

1. Master/Worker Server의 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
```

GooseFS Server에서 인증을 구성한 후 구성이 적용되도록 전체 클러스터를 다시 시작합니다.

2. Client의 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_16_0_4.keytab
```

> 주의 :
> Kerberos 인증 시스템에서와 같이 KDC는 현재 Client가 액세스 한 Service를 알아야 하며, GoosFS는 Server principal을 사용하여 현재 Client가 요청한 Service를 식별해야 합니다.

이 시점에서 GooseFS는 Kerberos의 기본 인증 서비스에 연결되었으며 클라이언트가 시작한 모든 요청은 이후 Kerberos가 인증합니다.

©2013-2022 Tencent Cloud. All rights reserved.