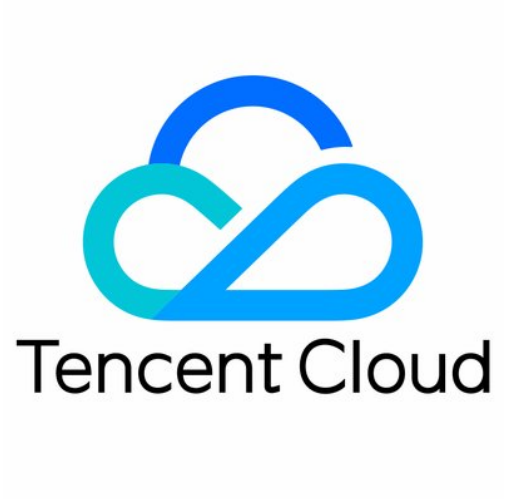
Cloud Object Storage

データレークストレージ

製品ドキュメント



カタログ：

データレークストレージ クラウドネイティブデータレイク クイックスタート

最終更新日：：2023-09-18 09:55:29

概要

クラウドネイティブデータレイクストレージサービスは、Tencent Kubernetes Engine（TKE）上で、Cloud Object Storage（COS）をベースにしたデータレイクストレージサービスをスピーディーにデプロイする上で役立ちます。クラウドネイティブデータレイクストレージサービスによって、業務に必要な各種ビッグデータサービスアプリケーションおよびAIサービスアプリケーションをTKEまたはEKSクラスター上に素早くデプロイすることができるほか、データアクセラレーターGooseFSによって大量の分散型ストレージサービスを連結することもできます。

概念と用語

クラウドネイティブデータレイクストレージサービスのご利用にあたり、次の説明をお読みいただくことで、関連の**概念と用語**を一通り理解することができます。

- **環境**：クラウドネイティブデータレイクストレージサービスにおいて、コンピューティングクラスターとストレージサービス間のマッピング関係を維持するために用いられます。コンピューティングクラスターおよびストレージサービスはこのエントリーから一元的に管理することを推奨します。

  > 注意：
  > コンピューティングクラスターを削除したい場合は、先にクラウドネイティブデータレイクストレージ環境をクリアしてから、TKEコンソール上でコンピューティングクラスターを削除することをお勧めします。

- **コンピューティングクラスター**：各種のコンピューティング業務の実行に用いるコンテナクラスターです。TKEクラスターまたはEKSクラスターを作成することができます。
- **ストレージサービス**：特にCOSサービスを指します。コンピューティング業務に用いる各種データを保存するために用いられます。
- **アプリケーションマーケット**：Flink、Sparkなど、各種のコンピューティング業務の実行に用いるアプリケーションコンポーネントです。環境作成の際に、ニーズに合わせて必要なアプリケーションを選択できます。

> 注意：
> コンテナクラスターが破棄されると、デプロイしたアプリケーションも破棄されます。削除操作は慎重に行ってください。

- **データアクセラレーターGooseFS**：さまざまな下層バケットの格納管理に用いることができ、ホットデータをコンピューティングクラスターにキャッシュすることで、コンピューティング業務のアクセラレーションを可能にします。

次のドキュメントによって、基本的な情報をあらかじめご理解いただくことも可能です。

- COSサービス：クイックスタートで、バケットを作成して、ファイルをバケットにアップロード、およびバケットからダウンロードする方法についてご説明しています。
- Tencent Kubernetes Engine：クイックスタートで、TKEクラスターまたはEKSクラスターを作成する方法についてご説明しています。
- アプリケーションマーケット：Tencent Kubernetes Engineのアプリケーションマーケットで、クラスター内にアプリケーションを作成してデプロイする方法についてご説明しています。
- データアクセラレーターGooseFS：このサービスはさまざまな下層バケットの一元的な格納管理に用いることができ、業務アクセスのアクセラレーションも可能にします。

前提条件

- 現在クラウドネイティブデータレイクストレージサービスはホワイトリスト機能に属しています。ホワイトリストを有効化して使用するには、お問い合わせください。
- クラウドネイティブデータレイクストレージサービスはTKEおよびCOSサービスに依存しています。ご利用の過程で、**コンピューティングサービスとストレージサービスの操作権限を有する**ことを確実にしておく必要があります。サブアカウントを使用してログインする場合は、そのサブアカウントが少なくとも次の権限を有することを確認してください。
- COSサービスのバケットとファイルの操作権限：
  - バケット操作権限：バケット設定の管理が必要な場合は、ルートアカウントを通じて対応するバケット設定の操作権限を取得してください。通常、このバケット設定権限はデータの読み取り/書き込みには影響せず、追加設定も必要ありません。最大で設定読み取り操作権限までを承認します（例えば、QcloudCOSBucketConfigReadポリシーセットなど）。
  - ファイル操作権限：通常、コンピューティング業務にはバケット内のファイルの読み取りと書き込みが必要なため、ルートアカウントを通じてファイルの全読み取り書き込み権限（例えば、QcloudCOSDataFullControl)）の承認を行うか、またはルートアカウントが最小権限の原則に従って権限承認を行うことができます。
- コンテナクラスターの管理権限：

- クラスター操作権限：通常はクラスターの作成および操作権限の承認が必要です。TKEプリセットポリシーを使用した権限承認を参照して設定を行うことができます。
- クラスター管理権限：TKEはKubernetes RBACに連結した権限承認方式を提供しており、サブアカウントに対し粒度の細かいアクセス権限制御を行うことができます。サブアカウントでの操作の際は、TKE Kubernetesのオブジェクトレベルでの権限制御も必ずご参照ください。
- アプリケーションマーケットの操作権限：アプリケーションマーケットはイメージウエアハウスの操作に依存しています。サブアカウントへの権限承認は、TKEイメージウエアハウスリソースレベルの権限設定を参照して行うことができます。

操作手順

操作手順全体は、環境の作成、クラスターのバインド、コンピューティングアプリケーションのデプロイ、ストレージサービスのバインド、環境の管理などの重要手順に大きく分けられます。具体的な操作ガイドは次のとおりです。

1. COSコンソールにログインします。
2. 左側ナビゲーションバーで、**クラウドネイティブデータレイクストレージ**をクリックし、クラウドネイティブデータレイクストレージサービス画面に進みます。
3. クラウドネイティブデータレイクストレージサービス画面では、ページビューに機能紹介、デプロイガイドという2つの内容が表示されます。

- デプロイガイドはデフォルトで表示されます。右上の**ガイドを折りたたむ**をクリックすると、ガイドナビゲーションを閉じることができます。
- クラウドネイティブデータレイクストレージサービス環境リストのページでは検索が可能です。すでに存在する環境について、次の操作を行うことができます。
  - **環境名**をクリックして、環境詳細ページに進み、環境の管理を行います。
  - **クラスターのバインド**をクリックし、TKEコンソールを開いて、対応するクラスターの詳細ページに進みます。
  - バケットのバインドをクリックし、バケットページに進んでバケット内のファイル情報を確認します。

4. **環境の作成**をクリックし、環境作成フローに進みます。
   環境を作成するには、先に対応するコンテナコンピューティングクラスターを選択し、それに次のパラメータを設定する必要があります。

- **環境名**：環境情報の表示に使用します。最大63文字までの、グローバルで一意の名前です。
- **リージョン**：コンテナクラスターのリージョン情報を選択します。
- **クラスタータイプ**：TKEクラスターとEKSクラスターから選択できます。現在のリージョンにクラスターがない場合は、**コンテナクラスターの作成**をクリックし、TKEコンソールでクラスターを新規作成することができ

ます。

- **クラスター：リージョン指定**および**クラスタータイプ指定**の条件下で、コンピューティングアプリケーションサービスのデプロイ、コンピューティング作業の実行に用いるクラスターの名前です。
- **コンピューティングアプリケーション**：コンピューティング作業の実行に必要なアプリケーションサービスです。現在はデフォルトでFlink、big-data-suite、colocation、airflow、pytorch、spark-operatorなどのアプリケーションをサポートしており、ニーズに応じて選択できます。カスタムアプリケーションをデプロイしたい場合は、TKEコンソールでご自身でデプロイすることができます。アプリケーションは複数選択が可能です。

5. **次のステップ**をクリックし、**バケット設定**ページビューに進みます。
このページでコンピューティングクラスターにさまざまなバケットを設定することができます。データアクセラレーターGooseFSサービスをデフォルトで提供しており、さまざまなバケットを格納管理してデータをコンピューティングクラスターのローカルノードにキャッシュし、コンピューティング作業のアクセラレーションに用いることができます。これには次のパラメータを設定する必要があります。

- リージョン情報：編集はできず、コンピューティングクラスターの選択したリージョンにデフォルトで追従します。このリージョンに選択可能なバケットがない場合は、**バケットの作成**をクリックし、コンピューティングタスクに用いるバケットを新規作成することができます。
- バケット：指定のリージョンで複数のバケットを選択できます。バケット内の特定のファイルディレクトリのみのマウントもサポートしています。

> 注意：
> バケット全体をマウントする場合は、2つ目の入力ボックスへの入力は不要です。ディレクトリを指定したい場合は、ディレクトリ名を入力します。形式は `prefix/*` のようにします。

- GooseFSの有効化：GooseFSサービスはコンピューティング作業パフォーマンスのアクセラレーションに用いられ、デフォルトで有効になっており、変更はできません。これによって追加料金が発生することはありません。

6. **次のステップ**をクリックし、**GooseFSアプリケーション設定**ページビューに進みます。
データレイク環境下では、すべてのコンピューティングタスクはGooseFSサービスを介してCOSにアクセスする必要があるため、GooseFSに対し、アクセス権限のある指定のバケットのsecretIdおよびsecretKeyを設定する必要があります。
7. **次のステップ**をクリックし、情報を確認します。
8. 設定項目を変更したい場合は、**変更**をクリックして、設定情報を変更することができます。誤りがないことを確認後、**環境の作成**をクリックすると、環境作成操作は完了です。**環境リストに戻って更新**すると、新規作成したクラウドネイティブデータレイクストレージサービス環境を確認できます。

環境を削除したい場合は、環境リストで**削除**をクリックし、ポップアップウィンドウでその削除操作を確定すれば削除できます。
9. リストの環境名をクリックすると、**基本情報**ページに進むことができます。
3つのカードビューを使用し、環境情報、コンピューティングクラスター情報、バケット情報についてそれぞれ記述しています。

- データレイク環境情報：環境の名前、リージョン、バインドしたコンピューティングクラスター、ストレージサービスおよび作成時間などの情報の表示に用います。
- コンピューティングクラスター情報：コンピューティングクラスターの名前、ノード数、CPU、メモリ、GPU使用量などの基本情報の表示に用います。コンピューティングクラスターの詳細を知りたい場合は、「詳細を見る」をクリックすると、TKEコンソールにリダイレクトされ、そこで確認することができます。
- バケット情報：コンピューティングクラスターにバインドしたバケットの名前、ファイルパスおよびGooseFSの使用状態の表示に用います。ストレージサービスの詳細を確認したい場合は、「詳細を見る」をクリックして確認できます。

上記の手順がすべて完了すると、データレイク環境作成のフローは完了です。

メタデータアクセラレーション
メタデータアクセラレーション機能の概要

最終更新日：：2022-09-28 14:50:08

メタデータアクセラレーション機能はTencent Cloud COS（Cloud Object Storage）サービス向けにハイパフォーマンスなファイルシステム機能をご提供します。メタデータアクセラレーション機能の基盤にはCloud HDFSの優れたメタデータ管理機能を採用し、ユーザーがファイルシステムのセマンティクスによってCOSサービスにアクセスできるようサポートします。システム設計指標は帯域幅2.4Gb/s、10万レベルのQPSおよびミリ秒レベルの遅延を実現可能です。バケットでメタデータアクセラレーション機能を有効にすることで、ビッグデータ、ハイパフォーマンスコンピューティング、機械学習、AIなどのシーンに幅広く応用できます。

メタデータアクセラレーション機能を有効にしていないバケットでは、デフォルトでメタデータアクセスのアクセラレーション効果はなく、POSIXファイルのセマンティクスによるアクセスがサポートされないため、ファイルLISTのパフォーマンスが制限されるほか、ファイルのRENAME操作もサポートされません。メタデータアクセラレーションを有効にすると、ネイティブのCOS APIと、クライアントにデプロイしたHadoopツール、コンソールまたはSDKなどの方式のどちらによっても、POSIXファイルのセマンティクスでバケット内のファイルにアクセスできるようになります。

> 注意：
> メタデータアクセラレーション機能はバケットの作成時にのみ有効にすることができます。有効後の無効化はサポートしていないため、有効にするかどうかは業務の状況を踏まえて**慎重に検討**してください。

使用制限

現時点でバケットのメタデータアクセラレーション機能を有効にした場合の、関連製品機能のサポート状況については次の表のとおりです。

指標項目	メタデータアクセラレーション有効時	メタデータアクセラレーション無効時
バケット作成/照会/削除操作	サポートあり	サポートあり
オブジェクトアップロード/ダウンロード/削除操作	サポートあり	サポートあり
バケット権限	サポートあり	サポートあり

指標項目	メタデータアクセラレーション有効時	メタデータアクセラレーション無効時
オブジェクト権限	デフォルトでバケット権限を継承	サポートあり
同一都市内障害復旧	サポートあり	サポートあり
バケット暗号化	非該当	サポートあり
オブジェクト暗号化	非該当	サポートあり
クロスドメインアクセス CORS	非該当	サポートあり
リンク不正アクセス防止	非該当	サポートあり
バージョン管理	非該当	サポートあり
バケットコピー	非該当	サポートあり
静的ウェブサイト	非該当	サポートあり
back-to-origin設定	非該当	サポートあり
ライフサイクル	サポートあり	サポートあり
リスト機能	非該当	サポートあり
バケットタグ	非該当	サポートあり
COS Select	非該当	サポートあり
COS Batch	非該当	サポートあり
Cloud Infinite機能	非該当	サポートあり

課金説明

現在メタデータアクセラレーション機能はパブリックベータ版機能のため、現時点では無料です。今後商用化により有料となる場合は、サイト内メッセージ、メール、SMSなどの方法でお知らせします。課金に関する最新の状況については、サイト内メッセージまたは課金概要をご覧ください。

メタデータアクセラレーション機能を有効にしたバケットへのHDFSデータの移行

最終更新日：：2023-03-20 15:06:18

概要

メタデータアクセラレーターはTencent Cloud COS（Cloud Object Storage）サービス向けにハイパフォーマンスなファイルシステム機能をご提供します。メタデータアクセラレーターの基盤にはCloud HDFSの優れたメタデータ管理機能を採用し、ユーザーがファイルシステムのセマンティクスによってCOSサービスにアクセスできるようサポートします。システム設計指標は100GBレベルの帯域幅、10万レベルのQPSおよびミリ秒レベルの遅延を実現可能です。バケットでメタデータアクセラレーターを有効にすることで、ビッグデータ、ハイパフォーマンスコンピューティング、機械学習、AIなどのシーンに幅広く応用できます。メタデータアクセラレーターの詳細な説明に関しては、メタデータアクセラレーターをご参照ください。

COSはメタデータアクセラレーションサービスによってHadoopファイルシステムのセマンティクスを提供しているため、Hadoop Distcpツールを利用すると、Cloud Object Storage（COS）と他のHadoopファイルシステム間の双方向データマイグレーションを便利に行うことができます。ここではHadoop Distcpツールによってローカル HDFS内のファイルをCOSのメタデータアクセラレーションバケットに移す方法について重点的にご説明します。

移行環境の準備

マイグレーションツールの準備

1. 下の表にあるjarパッケージツールをダウンロードし、マイグレーションクラスターの送信先マシンのローカルディレクトリ下に配置します（例：/data01/jars）。

Tencent Cloud EMR環境

インストール説明

jarパッケージファイル名	説明	ダウンロードアドレス
cos-distcp-1.12-3.1.0.jar	COSDistCp関連パッケージです。データをCOSNにコピーします	COSDistCpツールをご参照ください
chdfs_hadoop_plugin_network-2.8.jar	OFSプラグイン	クリックしてダウンロード

自作Hadoop/CDHなどの環境

ソフトウェア依存

Hadoop-2.6.0およびそれ以上のバージョン、Hadoop-COSプラグイン8.1.5およびそれ以上のバージョンとし、またcos_api-bundleプラグインのバージョンはHadoop-COSバージョンに対応するものとします。詳細については、COSN github releasesでご確認ください。

インストール説明

Hadoop環境下で、次のプラグインをインストールします。

jarパッケージファイル名	説明	ダウンロードアドレス
cos-distcp-1.12-3.1.0.jar	COSDistCp関連パッケージです。データをCOSNにコピーします	COSDistCpツールをご参照ください
chdfs_hadoop_plugin_network-2.8.jar	OFSプラグイン	クリックしてダウンロード
Hadoop-COS	Version >= 8.1.5	Hadoop-COSツールをご参照ください
cos_api-bundle	バージョンはHadoop-COSに対応している必要があります	クリックしてダウンロード

注意：

- Hadoop-cosは、バージョン8.1.5以降、`cosn://bucketname-appid/` メソッドによるメタデータアクセラレーションバケットへのアクセスをサポートしています。
- メタデータアクセラレーション機能は、バケット作成時にのみオンにすることができ、一度オンにするとオフにすることはできません。お客様のビジネスシーンに応じて、オンにするかどうか慎重にご検討ください。また、旧バージョンのHadoop-cosパッケージは、メタデータアクセラレーション機能がオンになっているバケットには正常にアクセスできませんのでご注意ください。

2. メタデータアクセラレーションバケットを作成し、メタデータアクセラレーションバケットのHDFSプロトコルを設定します。詳細な手順については、メタデータアクセラレーターを有効にしたバケットへのHDFSプロトコルを使用したアクセスの、「バケットの作成とHDFSプロトコルの設定」の章をご参照ください。
3. マイグレーションクラスター `core-site.xml` を変更し、変更完了後にすべてのノード上に送信して設定します。データの移行のみの場合は、ビッグデータコンポーネントの再起動は必要ありません。

key	value	設定ファ

key	value	設定フ
fs.cosn.trsf.fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	core-s
fs.cosn.trsf.fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	core-s
fs.cosn.trsf.fs.ofs.tmp.cache.dir	形式は/data/emr/hdfs/tmp/のようになります	core-s
fs.cosn.trsf.fs.ofs.user.appid	お客様のCOS bucketに対応するappid	core-s
fs.cosn.trsf.fs.ofs.ranger.enable.flag	false	core-s

key	value	設定フ
fs.cosn.trsf.fs.ofs.bucket.region	bucketに対応するregion	core-s

4. マイグレーションクラスターがプライベートネットワークを経由してメタデータアクセラレーションバケットにアクセス可能かどうかを検証します。詳細な手順については、メタデータアクセラレーターを有効にしたバケットへのHDFSプロトコルを使用したアクセスの、「コンピューティングクラスターを設定してCOSにアクセス」の章をご参照ください。マイグレーションクラスターを送信して、COSに正常にアクセスできるかどうかを検証します。

既存データの移行

1. 移行ディレクトリの決定

通常、データ移行はまずHDFSストレージデータの移行から開始します。ソースHDFSクラスターの移行対象のディレクトリを選定し、ターゲット側とソース側のパスは同一にします。以下に示します。

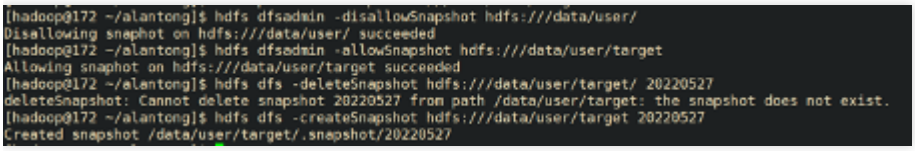
HDFSパス `hdfs:///data/user/target` を `cosn://{bucketname-appid}/data/user/target` に移行したいと仮定します。

移行の過程で、ソース側ディレクトリのファイルが変更されないようにするため、HDFSのスナップショット機能を使用し、先に移行対象のディレクトリのスナップショットを作成します。現在の日付をスナップショットのファイル名とします。

```
hdfs dfsadmin -disallowSnapshot hdfs:///data/user/
hdfs dfsadmin -allowSnapshot hdfs:///data/user/target
```

```
hdfs dfs -deleteSnapshot hdfs:///data/user/target {現在の日付}
hdfs dfs -createSnapshot hdfs:///data/user/target {現在の日付}
```

成功の例：



スナップショットを使用したくない場合は、ソース側のtargetファイルを直接移行することができます。

2. COSDistCpを使用して移行

COSDistCpタスクを起動し、ファイルをソースHDFSからターゲットCOSバケットにコピーします。

COSDistCpはMapReduceタスクであり、MapReduceタスクのプリントログにはMRタスクの実行が成功したかどうかが表示されます。失敗した場合はYARNページを確認し、ログまたはエラー情報をCOSに提供してトラブルシューティングを行うことができます。COSDistCpツールによる移行タスクの実行は次のいくつかの手順に分かれています。

（1）一時ディレクトリの作成
（2）COSDistCpタスクの実行
（3）失敗したファイルの再移行

（1）一時ディレクトリの作成

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

（2）COSDistCpタスクの実行

```
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/{現在の
日付} --dest=cosn://{bucket-appid}/data/user/target --temp=cosn://bucket-appid/di
stcp-tmp/ --preserveStatus=ugpt --skipMode=length-checksum --checkMode=length-che
cksum --cosChecksumType=CRC32C --taskNumber 6 --workerNumber 32 --bandWidth 200 >
> ./distcp.log &
```

パラメータの説明は次のとおりです。実際の状況に応じて調整できます。

- --taskNumber=VALUE コピープロセス数を指定します。例：--taskNumber=10。
- --workerNumber=VALUE コピースレッド数を指定します。COSDistCpは、各コピープロセスでこのパラメータサイズのコピースレッドプールを作成します。例：--workerNumber=4。

- --bandWidth 移行する各ファイルの読み込み帯域幅を制限します。単位はMB/s、デフォルトは-1で、読み込み帯域幅は制限しません。例：--bandWidth=10。
- --cosChecksumType=CRC32C、デフォルトではCRC32Cを使用しますが、HDFSクラスターはCOMPOSITE_CRC32チェックをサポートする必要があり、Hadoop バージョン3.1.1+に依存します。HDFSのバージョンが3.1.1より低い場合は、このパラメータを--cosChecksumType=CRC64に変更する必要があります。

> 注意：
> COSDistCpでは移行の総帯域幅制限計算式は、taskNumber x workerNumber x bandWidthとなります。workerNumberを1に設定し、パラメータtaskNumberによる移行の同時実行数の制御、およびパラメータbandWidthによる単一の同時実行帯域幅の制御を行うことができます。

コピータスクの終了時に、タスクログによってファイルのコピーに関する統計情報が出力されます。関連するカウンターは次のとおりです。
このうち、FILES_FAILEDは失敗の数を表します。FILES_FAILEDの統計項目がない場合は、すべての移行が成功したことになります。

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

具体的な出力結果統計項目の説明は次のとおりです。

統計項目	説明
BYTES_EXPECTED	ソースディレクトリの統計に基づいてコピーが必要なファイルの合計サイズ。単位：バイト
FILES_EXPECTED	ディレクトリファイルを含む、ソースディレクトリの統計に基づいてコピーが必要なファイル数
BYTES_SKIPPED	長さまたはチェックサム値が等しく、コピーされないファイルサイズの合計。単位：バイト
FILES_SKIPPED	長さまたはチェックサム値が等しく、コピーされないソースファイル数

統計項目	説明
FILES_COPIED	コピーに成功したソースファイル数
FILES_FAILED	コピーに失敗したソースファイル数
FOLDERS_COPIED	コピーに成功したディレクトリ数
FOLDERS_SKIPPED	スキップしたディレクトリ数

（3）失敗したファイルの再移行

COSDistCpツールはファイルの移行効率の問題の大部分を解決することができるほか、 `--delete` パラメータによってHDFSとCOSのデータを完全に一致させることもできます。

`--delete` パラメータを使用する場合は、 `--deleteOutput=/xxx(カスタム)` パラメータを含める必要があります。ただし `--diffMode` パラメータを含めることはできません。

```
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/
{現在の日付} --dest=cosn://{bucket-appid}/data/user/target --temp=cosn://bucket-ap
pid/distcp-tmp/ --preserveStatus=ugpt --skipMode=length-checksum --checkMode=leng
th-checksum --cosChecksumType=CRC32C --taskNumber 6 --workerNumber 32 --bandWidth
200 --delete --deleteOutput=/dele-xx >> ./distcp.log &
```

実行完了後、HDFSとCOSの差分データは `trash` ディレクトリに移動し、 `/xxx/failed` ディレクトリ下に移動ファイルリストが生成されます。 `trash` ディレクトリ下のデータの削除には `hadoop fs -rm URL` または `hadoop fs -rmr URL` を用いることができます。

増分の移行

各回の移行が完了した後に、更新された増分データが存在し、その移行も必要な場合は、データの移行がすべて完了するまで、全量移行の手順を繰り返し実行します。

メタデータアクセラレーターを有効にしたバケットへのHDFSプロトコルを使用したアクセス

最終更新日：：2023-03-20 15:06:18

メタデータアクセラレーターの概要

メタデータアクセラレーターはTencent Cloud COS（Cloud Object Storage）サービス向けにハイパフォーマンスなファイルシステム機能をご提供します。メタデータアクセラレーターの基盤にはCloud HDFSの優れたメタデータ管理機能を採用し、ユーザーがファイルシステムのセマンティクスによってCOSサービスにアクセスできるようサポートします。システム設計指標は100GBの帯域幅、10万レベルのQPSおよびミリ秒レベルの遅延を実現可能です。バケットでメタデータアクセラレーターを有効にすることで、ビッグデータ、ハイパフォーマンスコンピューティング、機械学習、AIなどのシーンに幅広く応用できます。メタデータアクセラレーターの詳細な説明に関しては、メタデータアクセラレーターをご参照ください。

HDFSプロトコルを使用したアクセスのメリット

これまでのCOSをベースにしたビッグデータアクセスには主にHadoop-COSツールが使用されてきました。Hadoop-COSツールの内部ではHCFSインターフェースをCOSのRestfulインターフェースに適用することで、COS上のデータへのアクセスを可能にしています。COSとファイルシステムではメタデータの構成方法に違いがあるため、メタデータの操作性能にも差異があり、ビッグデータの分析性能に影響しています。メタデータアクセラレーターを有効にしたBucketは、HCFSプロトコルとの間に完全な互換性を有し、ネイティブのHDFSインターフェースを使用して直接アクセスできます。HDFSプロトコルをオブジェクトのプロトコルに変換するコストが不要なだけでなく、高効率なディレクトリのアトミックRename、ファイルのAtime、Mtime更新、高効率なディレクトリのDU統計、Posix ACL権限サポートなどの、ネイティブHDFSのいくつかの機能もご提供可能です。

バケットの作成とHDFSプロトコルの設定

1. COSバケットの作成を行い、メタデータアクセラレーターを有効化します。
   バケットの作成完了後、バケットの**ファイルリスト**のページに進み、このページでファイルのアップロードおよびダウンロード操作を行うことができます。
2. 左側のメニューバーで、**パフォーマンス設定 > メタデータアクセラレーション機能**をクリックすると、メタデータアクセラレーション機能が有効になっていることを確認できます。

メタデータアクセラレーションを有効にしたいバケットを初めて作成した場合は、表示に従って対応する**権限承認**操作を行う必要があります。クリックして権限承認を完了すると、HDFSプロトコルが自動的に有効化され、デフォルトのバケットマウントポイント情報を見ることができるようになります。

> 説明：
> 対応するHDFSファイルシステムが見つからないと表示される場合は、チケットを提出をクリックしてお問い合わせいただければ、サポートを受けることができます。

3. HDFS権限設定バーで、**権限設定の追加**をクリックします。

4. VPCネットワーク名の列でコンピューティングクラスターのあるVPCネットワークアドレスを選択し、ノードのIPアドレス列に、VPCネットワークセグメント下で許可したいIPアドレスまたはIPセグメントを入力します。アクセスのタイプは読み取り/書き込みまたは読み取り専用を選択し、設定完了後に**保存**をクリックします。

> 説明：
> **HDFSの権限設定**とネイティブのCOS権限システムとの間には違いがあります。HDFSプロトコルを使用してアクセスする際は、ネイティブHDFSと同じ権限を取得するのに便利なため、HDFS権限設定によってVPC内のマシンを指定してCOSバケットにアクセスすることをお勧めします。

コンピューティングクラスターを設定してCOSにアクセス

環境の依存

依存項目	chdfs-hadoop-plugin	COSN（hadoop-cos）	cos_api-bundle
必要なバージョン	≥ バージョン2.7	≥ バージョン8.1.5	COSNのバージョンに対応します。COSN github releasesをご確認ください
オープンソースダウンロードアドレス	Githubアドレス	Githubアドレス	Githubアドレス

Tencent Cloud EMR環境

Tencent Cloud EMR環境 COSがすでにシームレスに統合されているため、次の手順を行うだけで完了します。

1. EMRマシンを見つけ、このマシン上で次のコマンドを実行します。EMR環境下で必要なサービスのフォルダにあるパッケージのバージョンが環境依存要件に適合するかどうかをチェックします。

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

```
[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
```

検索結果の2つのjarパッケージのバージョンが上記の環境依存の要件に適合していることを確認します。

2. chdfs-hadoop-pluginバージョンのパッケージを更新する必要がある場合は、次の手順を実行し、更新します。

   jarパッケージのスクリプトファイルをダウンロードして更新します。ダウンロードアドレスは次のとおりです。

   - update_cos_jar.sh
   - update_cos_jar_common.sh

   2つのスクリプトをサーバーの/rootディレクトリ下に置き、update_cos_jar.shに実行権限を追加し、次のコマンドを実行します。

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

パラメータを対応するリージョンのバケットのものに置き換えます。例えば広州リージョンの場合は、`https://hadoop-jar-guangzhou-1259378398.cos.ap-guangzhou.myqcloud.com/hadoop_plugin_network/2.7` に置き換えます。

各EMRノード上で上記の手順を実行します。これを、マシン上のjarパッケージの置き換えがすべて完了するまで行います。

3. hadoop-cosパッケージまたはcos_api-bundleバージョンのパッケージを更新する必要がある場合は、次の手順を実行し、更新します。

- /usr/local/service/hadoop/share/hadoop/common/lib/hadoop-temrfs-1.0.5.jarをtemrfs_hadoop_plugin_network-1.1.jarに置き換えます。
- core-site.xmlに次の設定項目を追加します。
  - emr.temrfs.download.md5=822c4378e0366a5cc26c23c88b604b11
  - emr.temrfs.download.version=2.7.5-8.1.5-1.0.6 （2.7.5をご自身のhadoopのバージョンに置き換え、8.1.5を必要なhadoop-cosパッケージのバージョンに置き換えます。ただし、バージョンは8.1.5より低くてはなりません。cos_api-bundleバージョンは自動的に適用されます）
  - emr.temrfs.download.region=sh
  - emr.temrfs.tmp.cache.dir=/data/emr/hdfs/tmp/temrfs
- core-site.xmlで設定fs.cosn.impl=com.qcloud.emr.fs.TemrfsHadoopFileSystemAdapterを変更します。

4. EMRコンソールでcore-site.xmlを設定し、設定項目 `fs.cosn.bucket.region` 、 `fs.cosn.trsf.fs.ofs.bucket.region` を追加します。このパラメータは、 `ap-shanghai` のようなバケットの所在COSリージョンを指定するために用いられます。

> 注意：
> `fs.cosn.bucket.region` と `fs.cosn.trsf.fs.ofs.bucket.region` は必ず設定する必要があります。このパラメータは、 `ap-shanghai` のようなバケットの所在COSリージョンを指定するために用いられます。

5. Yarn、Hive、Presto、Impalaなどの常駐サービスを再起動します。

自作Hadoop/CDHなどの環境

1. 自作環境では、環境依存の中の、バージョン要件に適合する3つのjarパッケージをダウンロードする必要があります。

2. ダウンロード後、上記の3つのインストールパッケージをHadoopクラスターの各サーバーの `classpath` パスに正しく配置します。例えば、 `/usr/local/service/hadoop/share/hadoop/common/lib/` となります（実際の状況に応じて配置してください。コンポーネントによって配置位置が異なる可能性があります）。

3. hadoop-env.shファイルを変更します。 `$HADOOP_HOME/etc/hadoop` ディレクトリに移動し、hadoop-env.shファイルを編集して、以下の内容を追加し、cosn関連のjarパッケージをHadoop環境変数に追加します。

```
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. コンピューティングクラスターに `core-site.xml` を設定し、次の設定を追加します。

```
<!--cosnの実装クラス-->
<property>
<name>fs.cosn.impl</name>
<value>org.apache.hadoop.fs.CosFileSystem</value>
</property>

<!--ユーザーバケットのリージョン情報です。形式はap-guangzhou-->のようになります
<property>
<name>fs.cosn.bucket.region</name>
<value>ap-guangzhou</value>
</property>

<!--ユーザーバケットのリージョン情報です。形式はap-guangzhou-->のようになります
<property>
<name>fs.cosn.trsf.fs.ofs.bucket.region</name>
<value>ap-guangzhou</value>
</property>

<!--SecretIdおよびSecretKeyの取得方法の設定-->
<property>
<name>fs.cosn.credentials.provider</name>
<value>org.apache.hadoop.fs.auth.SimpleCredentialProvider</value>
</property>

<!--アカウントのAPIキー情報です。[CAMコンソール](https://console.tencentcloud.com/capi)にログインすると、Tencent Cloud APIキーを確認することができます。-->
<property>
<name>fs.cosn.userinfo.secretId</name>
<value>XXXXXXXXXXXXXXXXXXXXXXXX</value>
</property>
```

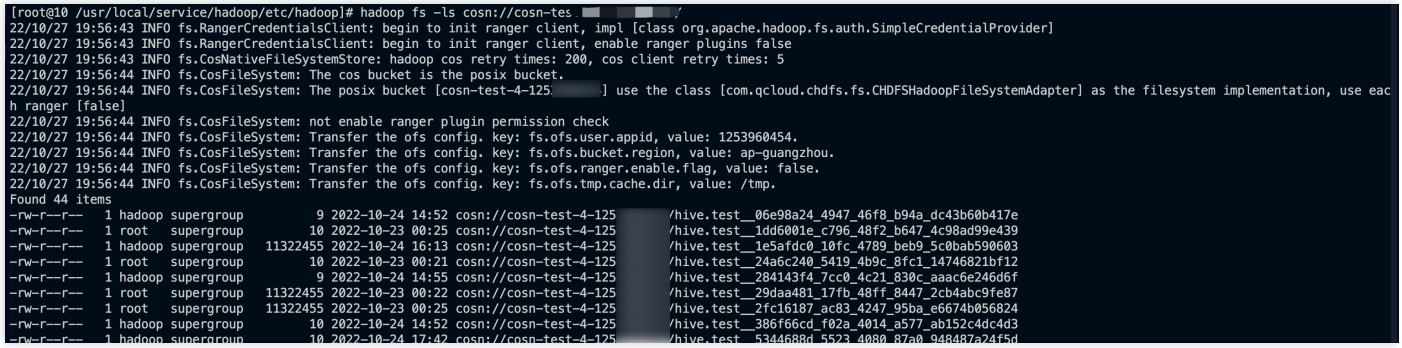
```
<!--アカウントのAPIキー情報です。[CAMコンソール](https://console.tencentcloud.com/capi)にログインすると、Tencent Cloud APIキーを確認することができます。-->
<property>
<name>fs.cosn.userinfo.secretKey</name>
<value>XXXXXXXXXXXXXXXXXXXXXXXX</value>
</property>

<!--アカウントのappidの設定-->
<property>
<name>fs.cosn.trsf.fs.ofs.user.appid</name>
<value>125XXXXXX</value>
</property>

<!--プロセス実行中に生成される一時ファイル保存用のローカル一時ディレクトリ-->
<property>
<name>fs.cosn.trsf.fs.ofs.tmp.cache.dir</name>
<value>/tmp</value>
</property>
```

5. Yarn、Hive、Presto、Impalaなどの常駐サービスを再起動します。

環境の検証

環境設定がすべて完了した後、次の操作によって検証を行うことができます。

- 下図のように、クライアントでHadoopコマンドラインを使用して、マウントが成功したかどうかを確認します。



- COSコンソールにログインしてバケットファイルリストを確認し、ファイルとディレクトリが一致しているかどうかを確認します。

Ranger権限設定

HDFSプロトコルは、デフォルトではネイティブPOSIX ACL方式を採用して認証を行います。Ranger認証を使用したい場合は、次のフローを参照して設定を行うことができます。

EMR環境

1. EMR環境にはCOSRangerサービスが統合されています。EMRクラスターの購入時にCOSRangerサービスにチェックを入れます。
2. HDFSプロトコルの**HDFS認証モード**で**Ranger認証モード**を選択し、Rangerに対応するアドレス情報を設定します。

- core-site.xmlに設定項目fs.cosn.credentials.providerを追加し、org.apache.hadoop.fs.auth.RangerCredentialsProviderに設定します。
- Rangerに関するご質問がありましたら、Rangerの概要説明をご参照ください。

自作Hadoop/CDHなどの環境

1. Rangerサービスを設定し、RangerサービスによってHDFSプロトコルでCOSにアクセスすることができます。詳細については、COS Ranger権限システムソリューションのドキュメントをご参照ください。
2. HDFSプロトコルの**HDFS認証モード**で**Ranger認証モード**を選択し、Rangerに対応するアドレス情報を設定します。

- core-site.xmlに設定項目fs.cosn.credentials.providerを追加し、org.apache.hadoop.fs.auth.RangerCredentialsProviderに設定します。

その他

ビッグデータのシーンでは、次の手順を参照して、メタデータアクセラレーション機能を有効にしたバケットにHDFSプロトコルを使用してアクセスすることができます。

1. バケットの作成とHDFSプロトコルの設定 で示したように、 `core-stie.xml` でHDFSプロトコルに関連するマウントポイントの情報を設定します。
2. Hive、MR、Sparkなどのコンポーネントによってバケットにアクセスします。コンピューティングクラスターにおけるCOSバケットのマウントをご参照ください。
3. デフォルトでは、ネイティブ `POSIX ACL` 方式を採用して認証を行います。 `Ranger認証` を使用したい場合は、 COS Ranger権限システムソリューションをご参照ください。

コンピューティングクラスターでのCOSバケットのマウント

最終更新日：：2023-03-14 16:54:52

概要

Cloud Object Storage（COS）はメタデータアクセラレーション機能を有効にすることで、HDFSプロトコルによるアクセスが可能になります。メタデータアクセラレーション機能を有効にすると、COSはバケットにマウントポイントを作成し、お客様はHDFSクライアントをダウンロードすることで、クライアントにこのマウントポイントを入力してCOSをマウントできます。ここでは、コンピューティングクラスターに、メタデータアクセラレーションを有効にしたバケットをマウントする方法について詳しくご説明します。

注意：

- Hadoop-cosは、バージョン8.1.5以降、`cosn://bucketname-appid/` メソッドによるメタデータアクセラレーションバケットへのアクセスをサポートしています。
- メタデータアクセラレーション機能は、バケット作成時にのみオンにすることができ、一度オンにするとオフにすることはできません。お客様のビジネスシーンに応じて、オンにするかどうか**慎重にご検討ください**。また、旧バージョンのHadoop-cosパッケージは、メタデータアクセラレーション機能がオンになっているバケットには正常にアクセスできませんのでご注意ください。

前提条件

- コンピューティングクラスター内のマウントしたいマシンまたはコンテナ内に、Java 1.8がインストールされていることをご確認ください。
- コンピューティングクラスター内のマウントしたいマシンまたはコンテナがアクセス権限を有していることをご確認ください。HDFS権限設定で、アクセス可能なVPCネットワークおよびIPアドレスを指定する必要があります。
- 依存するJARパッケージの説明

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) はインストールを必要とせず、自動的にプルされます。 hadoop fs lsを実行し成功すると、fs.cosn.trsf.fs.ofs.tmp.cache.dirで設定したディレクトリにおいて、対応するバージョンが想定どおりになっているかどうかを確認できます。

操作手順

1. Hadoopクライアントツールインストールパッケージをダウンロードします。
2. POSIX Hadoopクライアントツールインストールパッケージをダウンロードします。
3. cos java sdkインストールパッケージをダウンロードします。
4. タスクの起動時に正しくロードされるように、各ノードのclasspathにインストールパッケージを設定します。例えば、 `$HADOOP_HOME/share/hadoop/common/lib/` 下などです。

> 注意：
> EMR環境には独自の依存関係のあるjarパッケージがあるため、インストールする必要はなく、POSIXセマンティクスを介してメタデータアクセラレーションバケットに直接アクセスすることができます。s3プロトコルを使用してアクセスする必要がある場合は、fs.cosn.posix_bucket.fs.impl設定項目を変更します。詳細については、下記をご参照ください。

5. `core-site.xml` ファイルを編集し、次の基本設定を追加します。

> 注意：
> - 設定の際にパーマネントキーはできるだけ使用しないことをお勧めします。サブアカウントキーまたは一時キー方式を用いると業務の安全性が向上します。サブアカウントへの権限承認の際は、最小権限の原則についてのガイドに従い、データが意図せず漏洩しないようにしてください。
> - どうしてもパーマネントキーを使用したい場合は、パーマネントキーの権限範囲を限定することをお勧めします。最小権限の原則についてのガイドを参照し、パーマネントキーで実行できるアクション、リソースの範囲および条件（アクセスIPなど）を限定することで、使用上の安全性を向上させることができます。

```
<!--アカウントのAPIキー情報です。[CAMコンソール](https://console.tencentcloud.com/capi)にログインすると、Tencent Cloud APIキーを確認することができます。-->
<!--設定の安全性向上のため、サブアカウントキーまたは一時キー方式を使用して設定を行うことをお勧めします。サブアカウントへの権限承認の際は、[最小権限の原則についてのガイド](https://www.tencentcloud.com/document/product/436/32972)に従ってください。-->
```

```
<property>
<name>fs.cosn.userinfo.secretId/secretKey</name>
<value>AKIDxxxxxxxxxxxxxxxxxxxx</value>
</property>

<!--cosnの実装クラス-->
<property>
<name>fs.AbstractFileSystem.cosn.impl</name>
<value>org.apache.hadoop.fs.CosN</value>
</property>

<!--cosnの実装クラス-->
<property>
<name>fs.cosn.impl</name>
<value>org.apache.hadoop.fs.CosFileSystem</value>
</property>

<!--ユーザーバケットのリージョン情報です。形式はap-guangzhou-->のようになります
<property>
<name>fs.cosn.bucket.region</name>
<value>ap-guangzhou</value>
</property>

<!--プロセス実行中に生成される一時ファイル保存用のローカル一時ディレクトリ->
<property>
<name>fs.cosn.tmp.dir</name>
<value>/tmp/hadoop_cos</value>
</property>
```

6. `core-site.xml` をすべての `hadoop` ノードに同期します。

> 説明：
> EMRクラスターの場合、EMRコンソールのコンポーネント管理で上記の手順3と4において、HDFSの設定を変更することができます。

7. `hadoop fs` コマンドラインツールを使用して、 `hadoop fs -ls cosn://${bucketname-appid}/` コマンドを実行します。この `bucketname-appid` はマウントアドレス、すなわちバケット名です。ファイルリストが正常にリストアップされている場合は、COSバケットが正常にマウントされたことを意味します。
8. ユーザーは `hadoop` の他の設定項目または `mr` タスクを使用して、メタデータアクセラレーション機能を有効にしたCOSバケット上でデータタスクを実行することもできます。 `mr` タスクの場合， `-`

`Dfs.defaultFS=ofs://${bucketname-appid}/` によって、このタスクのデフォルトの入力・出力 `FS` を対応するバケットに変更することができます。

設定項目の説明

説明：
ここではPOSIXセマンティックによるアクセス、S3プロトコルによるアクセスの2つの方式によってメタデータアクセラレーションバケットにアクセスすることができます。ここではより良い性能を得るために、POSIXセマンティックによるアクセス方式を使用することをお勧めします。

1. 一般的な入力必須設定項目

注意：
どの方式でメタデータアクセラレーションバケットにアクセスするかに関わらず、以下の共通設定項目を設定する必要があります。

設定項目	設定項目内容	説明
fs.cosn.userinfo.secretId/secretKey	フォーマット例：AKIDxxxxxxxxxxxxxxxxxxxx	お客様のアカウントのAPIキー情を入力します。CAMコンソールにグインすれば、Tencent Cloud APキーを確認することができます。
fs.cosn.impl	org.apache.hadoop.fs.CosFileSystem	FileSystem用のcosn実装クラスです。org.apache.hadoop.fs.CosFileSysに固定されます。
fs.AbstractFileSystem.cosn.impl	org.apache.hadoop.fs.CosN	AbstractFileSystem用のcosn実装ラスです。org.apache.hadoop.fs.CosNに固れます。

設定項目	設定項目内容	説明
fs.cosn.bucket.region	フォーマット例：ap-beijing	アクセスするバケットのリージョ情報を入力してください。列挙値ついては、リージョンとアクセスメイン名のリージョンの略称をご照ください。例えば、ap-beijing、ap-guangzhouなどです。元の設定fs.cosn.userinfo.regionと互換性がります。
fs.cosn.tmp.dir	デフォルト/tmp/hadoop_cos	実際に存在するローカルディレクリを設定してください。プロセス行中に生成された一時ファイルは一時的にここに保存されます。また、各ノードでこのディレクトリ十分なスペースと権限を割り当てことをお勧めします。

2. POSIXアクセスメソッド入力必須設定項目（推奨方式）

説明：

- POSIXアクセス方式は共通設定項目以外に、さらに以下の内容を追加する必要があります。POSIX アクセス方式の その他のオプション設定項目 から「fs.cosn.trsf.」 というプレフィックスを追加することでメタデータアクセラレーションバケットにアクセスすることができます。
- 元のHadoop cosに関連する設定項目は今後適用されなくなることに注意してください。

設定項目	設定項目内容	説明
fs.cosn.trsf.fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	メタデー
fs.cosn.trsf.fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	メタデー
fs.cosn.trsf.fs.ofs.tmp.cache.dir	フォーマット例：/data/emr/hdfs/tmp/posix-cosn/	実際に存設定して成された に保存さ ディレク 割り当て 例： "/ cosn/"

設定項目	設定項目内容	説明
fs.cosn.trsf.fs.ofs.user.appid	フォーマット例：12500000000	入力必須
fs.cosn.trsf.fs.ofs.bucket.region	フォーマット例：ap-beijing	入力必須 応します

3. S3プロトコルアクセスメソッド入力必須設定項目

S3プロトコルによるアクセス方式は以下の設定が必要です。その他のオプションについては、Hadoop-cos 設定項目をご参照ください。

設定項目	設定項目内容	説明
fs.cosn.posix_bucket.fs.impl	org.apache.hadoop.fs.CosNFileSystem	POSIXメソッドのアクセス設定は `com.qcloud.chdfs.fs.CHDFSH` `S3` プロトコルメソッドのアクセス設 `org.apache.hadoop.fs.CosNFi` POSIXメソッドでのアクセスです。

5. 注意事項

1. 古いhadoop cos jarパッケージを使用してメタデータアクセラレーションのバケットにアクセスすることはできません。
2. Hadoop cos ≤ 8.1.5バージョンのposix方式を使用してメタデータアクセラレーションのバケットへのアクセスが有効な場合は、コンソールでranger検証を無効にする必要があります。8.1.5以上のバージョンはコンソールでranger検証を有効にすることをサポートしています。

CDHクラスターのHDFSプロトコルからCOSへのアクセス

最終更新日：：2022-06-07 14:13:31

概要

CDH(Cloudera's Distribution, including Apache Hadoop)は、業界で流行りのHadoopディストリビューションです。ここでは、CDH環境において、HDFSプロトコルを使用してCOS（Cloud Object Storage）バケットへアクセスし、ビッグデータの計算とストレージの分離を実現し、フレキシブルかつ低コストのビッグデータソリューションを提供する方法についてご説明します。

> 注意：
> HDFSプロトコルを使用してCOSバケットへアクセスし、最初にメタデータアクセラレーション機能をオンにする必要があります。

現在ビッグデータコンポーネントに対するCOSのサポート状況は次のとおりです。

コンポーネント名	CHDFSビッグデータコンポーネントのサポート状況	サービスコンポーネントの再起動
Yarn	サポート	NodeManagerの再起動
Yarn	サポート	NodeManagerの再起動
Hive	サポート	HiveServerおよびHiveMetastoreの動
Spark	サポート	NodeManagerの再起動
Sqoop	サポート	NodeManagerの再起動
Presto	サポート	HiveServerおよびHiveMetastoreとPrestoの再起動
Flink	サポート	いいえ
Impala	サポート	いいえ
EMR	サポート	いいえ

コンポーネント名	CHDFSビッグデータコンポーネントのサポート状況	サービスコンポーネントの再起動
セルフビルドコンポーネント	フォローアップサポート	なし
HBase	推奨しない	なし

バージョンの依存関係

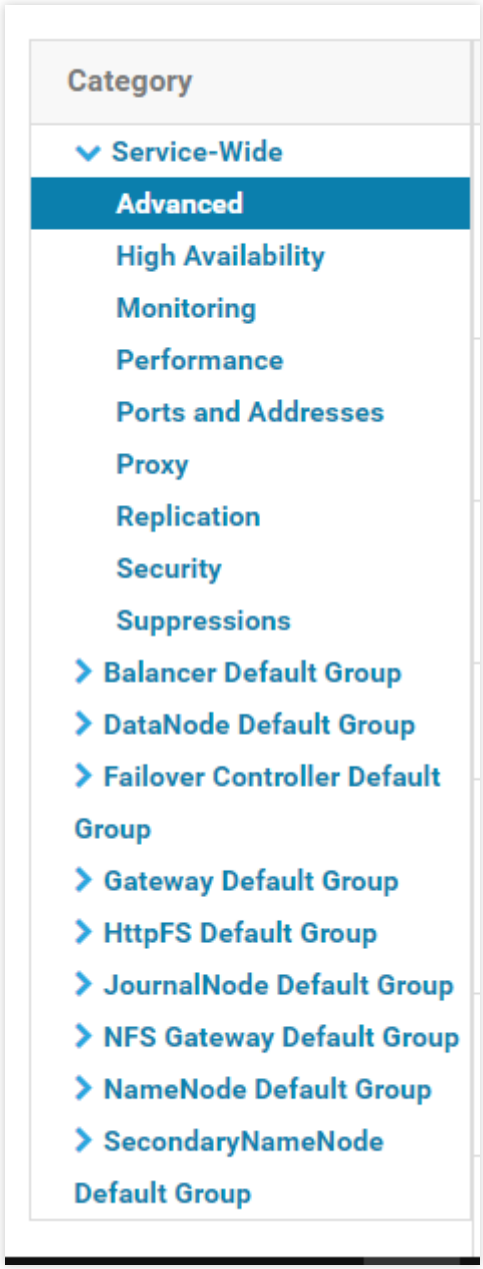
依存するコンポーネントのバージョンは次のとおりです。

- CDH 5.16.1
- Hadoop 2.6.0

利用方法

ストレージ環境の設定

1. CDH管理ページにログインします。
2. 下図に示すとおり、**設定** > *サービス範囲** > **高度**を選択して、コードセグメントの高度な設定ページに進みます。



3. Cluster-wide Advanced Configuration Snippet(Safety Valve) for core-site.xmlのコードボックスに、COSビッグデータサービス設定を入力します。

```
<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>
<!--ローカルcacheの一時ディレクトリ。データの読み取りと書き込みの場合、メモリcacheが不足すると、ローカルディスクに書き込まれます。このパスが存在しない場合は、自動的に作成されます-->
```

```
<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>
```

以下は、入力必須の設定項目です（core-site.xmlに追加する必要があります）。その他の設定については、コンピューティングクラスターにおけるCOSバケットのマウントをご参照ください。

構成アイテム	値	意味
fs.ofs.user.appid	1250000000	ユーザーappid
fs.ofs.tmp.cache.dir	/data/emr/hdfs/tmp/chdfs/	ローカルcacheの一
fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	chdfsのFileSystemに com.qcloud.chdfs.fs. に固定されます
fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	chdfsのAbstractFileS com.qcloud.chdfs.fs. されます

4. HDFSサービスを操作し、「クライアント設定のデプロイ」をクリックします。この時点で、上記のcore-site.xmlの設定がクラスター内のマシンに更新されます。 5.最新の[クライアントインストールパッケージ](https://github.com/tencentyun/chdfs-hadoop-plugin/tree/master/jar)を、CDH HDFSサービスのjarパッケージパスに配置します。実際の値に従って置き換えてください。次に例を示します。

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

注意：
クラスター内の各マシンは、SDKパッケージを同じ場所に配置する必要があります。

データマイグレーション

Hadoop Distcpツールを使用して、CDH HDFSデータをCOSバケットに移行します。詳細は、Hadoopファイルシステムと COS間のデータ移行をご参照ください。

ビッグデータスイートによるCHDFSの使用

MapReduce

操作手順

1. [データマイグレーション]#1の章に従って、HDFSの関連設定を行い、COSのクライアントインストールパッケージをHDFSの対応するディレクトリに配置します。
2. CDHシステムのホームページでYARNを見つけてNodeManagerサービスを再起動します（TeraGen コマンドを再起動する必要はありませんが、TeraSortは内部ビジネスロジックのためにNodeMangerを再起動する必要があります。NodeManagerサービスを一律に再起動することをお勧めします）。

事例

以下に、Hadoop標準テストでのTeraGenとTeraSortの例を示します。

```
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
```

> 説明：
> `ofs://schema` の後ろを、ユーザーCHDFSのマウントポイントパスに置き換えてください。

Hive

MRエンジン

操作手順

1. データマイグレーションの章に従って、HDFSの関連設定の配置を行い、COSのクライアントインストールパッケージをHDFSの対応するディレクトリに配置します。
2. CDHメインページでHIVEサービスを見つけ、Hiveserver2およびHiverMetastoreのロールを再起動します。

事例

例えば、あるユーザーの実際の業務の照会では、Hiveコマンドラインを実行してLocationを1つ作成し、CHDFSのパーティションテーブルとします。

```
CREATE TABLE `report.report_o2o_pid_credit_detail_grant_daily`(
`cal_dt` string,
`change_time` string,
`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照会を実行します。

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

観察の結果は次のとおりです。

```
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エンジン

Tezエンジンは、COSのクライアントインストールパッケージをTezの圧縮パッケージにインポートする必要があります。以下は、apache-tez.0.8.5を例として説明しています。

操作手順

1. CDHクラスターによってインストールされたtezパッケージを見つけて、解凍します。
   例：/usr/local/service/tez/tez-0.8.5.tar.gz。
2. COSのクライアントインストールパッケージを、解凍したディレクトリ下に置き、圧縮パッケージを再圧縮して出力します。
3. tez.lib.urisで指定されたパス下に、新しい圧縮パッケージをアップロードします（以前にパスがある場合は、直接置き換えてください）。
4. CDHメインページでHIVEを見つけて、hiveserverおよびhivemetastoreを再起動します。

Spark

操作手順

1. データマイグレーションの章に従って、HDFSの関連設定の配置を行い、COSのクライアントインストールパッケージをHDFSの対応するディレクトリに配置します。
2. NodeManagerサービスを再起動します。

事例

例として、Spark example word countテストの実施を取り上げます。

```
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
```

実行結果は次のとおりです。

```
20/07/17 18:35:05 INFO storage.ShuffleBlockFetcherIterator: Started 0 remote fetches in 3 ms
20/07/17 18:35:05 INFO executor.Executor: Finished task 0.0 in stage 1.0 (TID 1). 1870 bytes result sent to driver
20/07/17 18:35:05 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 1.0 (TID 1) in 31 ms on localhost (executor driver) (1
20/07/17 18:35:05 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 1.0, whose tasks have all completed, from pool
20/07/17 18:35:05 INFO scheduler.DAGScheduler: ResultStage 1 (collect at JavaWordCount.java:68) finished in 0.031 s
20/07/17 18:35:05 INFO scheduler.DAGScheduler: Job 0 finished: collect at JavaWordCount.java:68, took 0.548912 s
And: 4
day.: 1
God: 4
Let: 1
light:: 1
it: 1
light,: 1
evening: 1
was: 2
that: 1
light.: 1
be: 1
Day,: 1
said,: 1
darkness.: 1
he: 1
first: 1
there: 2
light: 2
Night.: 1
morning: 1
darkness: 1
were: 1
saw: 1
divided: 1
and: 4
good:: 1
from: 1
called: 2
the: 8
```

Sqoop

操作手順

1. データマイグレーションの章に従って、HDFSの関連設定の配置を行い、COSのクライアントインストールパッケージをHDFSの対応するディレクトリに配置します。
2. COSのクライアントインストールパッケージもsqoopディレクトリに配置する必要があります（例：/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/lib/sqoop/）。
3. NodeManagerサービスを再起動します。

事例

例として、MYSQLテーブルのCOSへのエクスポートを取り上げます。リレーショナルデータベースとHDFSのインポート・エクスポートドキュメントを参照し、テストを実施してください。

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

実行結果は次のとおりです。

```
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

操作手順

1. データマイグレーションの章に従って、HDFSの関連設定の配置を行い、COSのクライアントインストールパッケージをHDFSの対応するディレクトリに配置します。

2. COSのクライアントインストールパッケージもprestoディレクトリ下に配置する必要があります（例：/usr/local/services/cos_presto/plugin/hive-hadoop2）。
3. prestoはhadoop commonのgson-2...jarをロードしないため、gson-2...jarもprestoディレクトリ下に配置する必要があります（例： /usr/local/services/cos_presto/plugin/hive-hadoop2、COSのみがgsonに依存します）。
4. HiveServer、HiveMetaStoreおよびPrestoサービスを再起動します。

事例

例として、HIVEによってLocationをCOSとして作成したテーブルの照会を取り上げます。

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

> 説明：
> chdfs_test_tableはlocationがofs schemeのテーブルです。

確認の結果は次のとおりです。

```
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
```

Hadoop Filesystem APIコードを使用したCOSメタデータアクセラレーションバケットへのアクセス

最終更新日：：2023-03-20 15:08:51

ユースケース

Cloud Object Storage（COS）バケットでメタデータアクセラレーションを有効にしている場合は、Hadoopコマンドライン、ビッグデータコンポーネントなどの方法による操作のほか、Hadoop Filesystem APIによって、Javaコードを使用してメタデータアクセラレーションバケットにアクセスすることができます。ここではJavaコードによってメタデータアクセラレーションバケットにアクセスする方法についてご説明します。

前提条件

- メタデータアクセラレーションをアクティブ化済みであり、かつ環境デプロイとHDFSプロトコルの設定が正しく行われている必要があります。具体的なデプロイと設定の詳細については、メタデータアクセラレーターを有効にしたバケットへのHDFSプロトコルを使用したアクセスをご参照ください。
- Hadoop環境がある場合は、Hadoopコマンドラインによって正しくアクセスできるかどうかを検証することができます。

操作手順

1. mavenプロジェクトを新規作成し、mavenのpom.xmlに次の依存項目を追加します（実際のHadoopバージョンおよび環境に応じてhadoop-commonパッケージのバージョンを設定してください）。

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

2. 以下のHadoopのコードを参照して変更します。設定項目については設定項目の説明のドキュメントを参照して変更できます。**また、その中のデータの永続化と可視性に関する説明に特にご注意ください。**
以下に示すのは、一般的なファイルシステムの一部の操作のみです。その他のインターフェースについては、Hadoop FileSystemインターフェースドキュメントをご参照ください。

```
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();
// 設定項目についてはhttps://www.tencentcloud.com/document/product/1106/41965をご参照ください
// 以下の設定は、入力必須項目です

conf.set("fs.cosn.trsf.fs.ofs.impl", "com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter");
conf.set("fs.cosn.trsf.fs.AbstractFileSystem.ofs.impl", "com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter");
conf.set("fs.cosn.trsf.fs.ofs.tmp.cache.dir", "/data/chdfs_tmp_cache");
// appidは実際のappidに置き換えます
conf.set("fs.cosn.trsf.fs.ofs.user.appid", "1250000000");
// regionは実際のリージョンに置き換えます
conf.set("fs.cosn.trsf.fs.ofs.bucket.region", "ap-beijing")
// その他のオプションの設定項目については、https://www.tencentcloud.com/document/product/1106/41965をご参照ください

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 {
// ファイルを作成します（存在する場合は上書きします）
// if the parent dir does not exist, fs will create it!
FSDataOutputStream out = fs.create(filePath, true);
try {
// ファイルに書き込みます
String content = "test write file";
out.write(content.getBytes());
} finally {
// closeが返された場合はデータの書き込みに成功したことを表します。エラーがスローされた場合は
データ書き込みに失敗したことを表します
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);
}


// デフォルトのチェックタイプはCOMPOSITE-CRC32Cです
private static void getFileCheckSum(FileSystem fs, Path path) throws IOException
{
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());
}
}


// 再帰的削除フラグは、ディレクトリを削除するために用いられます
// 再帰がfalseで、dirが空でない場合、操作は失敗します
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();
}


public static void main(String[] args) throws IOException {
// ファイルシステムの初期化
FileSystem fs = initFS();


// ファイルの作成
Path chdfsFilePath = new Path("/folder/exampleobject.txt");
createFile(fs, chdfsFilePath);


// ファイルの読み取り
readFile(fs, chdfsFilePath);


// ファイルまたはディレクトリの照会
queryFileOrDirStatus(fs, chdfsFilePath);


// ファイルのチェックサムの取得
getFileCheckSum(fs, chdfsFilePath);


// ローカルからファイルをコピーする
Path localFilePath = new Path("file:///home/hadoop/cosn_demo/data/exampleobject.t
xt");
copyFileFromLocal(fs, chdfsFilePath, localFilePath);


// ファイルをローカルで取得する
Path localDownFilePath = new Path("file:///home/hadoop/cosn_demo/data/exampleobje
ct.txt");
copyFileToLocal(fs, chdfsFilePath, localDownFilePath);


// リネーム
Path newPath = new Path("/doc/example.txt");
renamePath(fs, chdfsFilePath, newPath);
```

```
// ファイルの削除
deleteFileOrDir(fs, newPath, false);


// ディレクトリの作成
Path dirPath = new Path("/folder");
mkdir(fs, dirPath);


// ディレクトリにファイルを作成する
Path subFilePath = new Path("/folder/exampleobject.txt");
createFile(fs, subFilePath);


// ディレクトリのリストアップ
listDirPath(fs, dirPath);


// ディレクトリの削除
deleteFileOrDir(fs, dirPath, true);


// ファイルシステムを閉じる
closeFileSystem(fs);
}
}
```

3. コンパイルと実行。

説明：

- 実行する前に、classpathが正しく設定されていることを確認してください。classpathには、Hadoop commonパッケージおよびメタデータアクセラレーションバケットが依存するjarパッケージのパスが含まれる必要があります。
- EMR環境の場合、メタデータアクセラレーターを有効にしたバケットへのHDFSプロトコルを使用したアクセスの手順に従って操作した場合、Hadoop commonパッケージは通常、`/usr/local/service/hadoop/share/hadoop/common/` ディレクトリ下にあり、メタデータアクセラレーションバケットが依存するjarパッケージは通常、`/usr/local/service/hadoop/share/hadoop/common/lib/` ディレクトリ下にあります。

データアクセラレーター GooseFS クイックスタート

最終更新日：：2023-04-06 14:37:42

ここでは主に、GooseFSのスピーディーなデプロイ、デバッグに関するガイドをご提供します。ローカルマシン上にGooseFSをデプロイし、Cloud Object Storage（COS）をリモートストレージとして使用する手順のガイドをご提供します。具体的な手順は次のとおりです。

前提条件

GooseFSを使用する前には、次の準備作業が必要です。

1. COSサービスでバケットを1つ作成してリモートストレージとします。操作ガイドについては、コンソールクイックスタートをご参照ください。
2. Java 8またはそれより上のバージョンをインストールします。
3. SSHをインストールします。SSHによってLocalHostに接続し、リモートログインを行えることを確認します。
4. CVMサービスでインスタンスを1台購入します。操作ガイドについては、CVMの購入をご参照ください。ディスクがすでにインスタンスにマウントされていることを確認します。

GooseFSのダウンロードと設定

1. ローカルディレクトリを新規作成し、そのディレクトリ下に進み（必要に応じて他のディレクトリも選択できます）、公式リポジトリからGooseFSインストールパッケージをローカルにダウンロードします。インストールパッケージのGithubアドレスはgoosefs-1.4.1-bin.tar.gzです。

```
$ 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. 次のコマンドを実行して、インストールパッケージを解凍し、解凍後にインストールパッケージディレクトリに入ります。

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

解凍すると、goosefs-1.4.1、すなわちGooseFSのホームディレクトリが得られます。以下では `${GOOSEFS_HOME}` がこのディレクトリの絶対パスを表します。

3. `${GOOSEFS_HOME}/conf` のディレクトリ下に `conf/goosefs-site.properties` の設定ファイルを作成します。GooseFSはAIとビッグデータという2つのシナリオ向けの設定テンプレートを提供しており、上記のうち任意の内蔵テンプレートを使用できます。その後編集モードに進み、設定を変更します。
（1）AIシナリオテンプレートの使用に関するその他の情報については、GooseFS AIシナリオ本番環境設定の実践をご参照ください。

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

（2）ビッグデータシナリオテンプレートの使用に関するその他の情報については、GooseFSビッグデータシナリオ本番環境設定の実践をご参照ください。

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

4. 設定ファイル `goosefs-site.properties` 内で次の設定項目を調整します。

```
# Common properties
# Masterノードhost情報の調整
goosefs.master.hostname=localhost
goosefs.master.mount.table.root.ufs=${goosefs.work.dir}/underFSStorage

# Security properties
# 権限設定の調整
goosefs.security.authorization.permission.enabled=true
goosefs.security.authentication.type=SIMPLE

# Worker properties
# workerノード設定の調整、ローカルキャッシュメディア、キャッシュパスおよびキャッシュ容量サイズの指定
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
# ファイル読み取り書き込みキャッシュポリシーの指定
```

```
goosefs.user.file.readtype.default=CACHE
goosefs.user.file.writetype.default=MUST_CACHE
```

注意：
パス `goosefs.worker.tieredstore.level0.dirs.path` のパラメータを設定する前に、このパスを新規作成する必要があります。

GooseFSの有効化

1. GooseFSを有効化する前に、GooseFSディレクトリ下で次の起動コマンドを実行する必要があります。

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

これらのコマンドを実行すると、次のようなページが表示されます。

```
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]#
```

このコマンドの実行が完了すると、 `http://localhost:9201` および `http://localhost:9204` にアクセスして、MasterとWorkerの実行状態をそれぞれ確認できるようになります。

GooseFSを使用してCOS（COSN）またはTencent Cloud HDFS（CHDFS）をマウント

GooseFSでCOS（COSN）またはTencent Cloud HDFS（CHDFS）をGooseFSのルートパス上にマウントしたい場合は、先に `conf/core-site.xml` 設定の中でCOSNまたはCHDFSの必須設定項目を指定する必要があります。これには次に示すように、 `fs.cosn.impl` 、 `fs.AbstractFileSystem.cosn.impl` 、およ

び `fs.cosn.userinfo.secretId` と `fs.cosn.userinfo.secretKey` などが含まれますが、これらに限定されません。

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

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

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

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

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

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

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

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


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

説明：

- COSNの完全な設定については、Hadoopツールをご参照ください。
- CHDFSの完全な設定については、CHDFSのマウントをご参照ください。

次に、Namespaceを作成してCOSまたはCHDFSをマウントする方法および手順についてご説明します。

1. ネームスペース（namespace）を作成してCOSをマウントします。

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

注意：

- COSNをマウントするnamespaceを作成する際は、必ず `--secret` パラメータを使用してアクセスキーを指定し、なおかつ `--attribute` を使用してHadoop-COS（COSN）のすべての入力必須パラメータを指定しなければなりません。具体的な入力必須パラメータについては、Hadoopツールをご参照ください。
- Namespaceを作成する際、読み取り書き込みポリシー（rPolicy/wPolicy）を指定しない場合は、デフォルトで設定ファイル内の指定のread/write typeを使用するか、またはデフォルト値（CACHE/CACHE_THROUGH）を使用します。

Tencent Cloud HDFSのマウントに用いるnamespaceも同様に作成できます。

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

2. 作成に成功すると、クラスター内に作成したすべてのnamespaceを `ls` コマンドによってリストアップすることができます。

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

3. 次のコマンドを実行し、namespaceの詳細情報を指定します。

```
$ goosefs ns stat myNamespace

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

メタデータ内に記録する情報には次のものが含まれます。

番号	パラメータ	説明
1	name	namespaceの名前
2	path	namespaceのGooseFSにおけるパス
3	ttlTime	namespace下のディレクトリおよびファイルのttl期間
4	ttlAction	namespace下のディレクトリおよびファイルのttl処理動作。FREEとDELETEの2つの処理動作があり、デフォルトではFREE
5	ufsPath	namespaceのufs上でのマウントパス

番号	パラメータ	説明
6	creationTimeMs	namespaceの作成時間。単位はミリ秒
7	lastModificationTimeMs	namespace下のディレクトリおよびファイルの最終変更時刻。単位はミリ秒
8	persistenceState	namespaceの永続化の状態
9	mountPoint	namespaceがマウントポイントかどうか。常にtrue
10	mountId	namespaceのマウントポイントID
11	acl	namespaceのアクセス制御リスト
12	defaultAcl	namespaceのデフォルトのアクセス制御リスト
13	owner	namespaceのowner
14	group	namespaceのownerが所属するgroup
15	mode	namespaceのPOSIX権限
16	writePolicy	namespaceの書き込みポリシー
17	readPolicy	namespaceの読み取りポリシー

GooseFSを使用してTable内のデータをプリフェッチ

1. GooseFSはHive Table内のデータのGooseFSへのプリフェッチをサポートしています。プリフェッチの前に関連のDBをGooseFSにバインドしておく必要があります。関連のコマンドは次のとおりです。

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

2. `16.16.22:7004 test_for_demo`

注意：
コマンド内のthriftには実際のHive Metastoreのアドレスを入力する必要があります。

2. DBの追加が完了すると、現在バインドしているDBとTableの情報をlsコマンドによって確認できるようになります。

```
$ goosefs table ls test_db web_page

OWNER: hadoop
DBNAME.TABLENAME: testdb.web_page (
wp_web_page_sk bigint,
wp_web_page_id string,
wp_rec_start_date string,
wp_rec_end_date string,
wp_creation_date_sk bigint,
wp_access_date_sk bigint,
wp_autogen_flag string,
wp_customer_sk bigint,
wp_url string,
wp_type string,
wp_char_count int,
wp_link_count int,
wp_image_count int,
wp_max_ad_count int,
)
PARTITIONED BY (
)
LOCATION (
gfs://172.16.16.22:9200/myNamespace/3000/web_page
)
PARTITION LIST (
{
partitionName: web_page
location: gfs://172.16.16.22:9200/myNamespace/3000/web_page
}
)
```

3. loadコマンドによってTable内のデータをプリフェッチします。

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

Table内のデータのプリフェッチは非同期タスクのため、タスクIDが返されます。goosefs job stat <Job Id>コマンドによってプリフェッチ作業の実行進捗状況を確認できます。ステータスが"COMPLETED"になると、プリフェッチのプロセス全体が完了しています。

GooseFSを使用したファイルのアップロードおよびダウンロード操作

1. GooseFSはほとんどのファイルシステムの操作コマンドをサポートしています。現在サポートしているコマンドのリストは次のコマンドによって照会できます。

```
$ goosefs fs
```

2. `ls` コマンドによってGooseFS内のファイルをリストアップすることができます。ルートディレクトリ下のすべてのファイルをリストアップする方法について、以下の例で示します。

```
$ goosefs fs ls /
```

3. `copyFromLocal` コマンドによって、データをローカルからGooseFSにコピーできます。

```
$ 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` コマンドによってファイルの内容を確認できます。

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

5. GooseFSはデフォルトでローカルディスクを基盤ファイルシステムとして使用します。デフォルトのファイルシステムパスは `./underFSStorage` であり、 `persist` コマンドによってファイルをローカルファイルシステムに永続的に保存することができます。

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

GooseFSを使用したファイルのアップロードおよびダウンロード操作のアクセラレーション

1. ファイルの保存ステータスをチェックし、ファイルがキャッシュ済みかどうかを確認します。ファイルのステータスが `PERSISTED` であれば、ファイルはすでにメモリ内にあることを表し、ファイルのステータスが `NOT_PERSISTED` であれば、ファイルはメモリ内にないことを表します。

```
$ 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. ファイル内に単語「tencent」がいくつあるかを集計し、操作にかかった時間を計算します。

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

3. このデータをメモリ内にキャッシュすると、照会速度を効果的に向上させることができます。詳細な例は次のとおりです。

```
$ 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
```

このように、システム処理の遅延が1.181sから0.243sに減少し、スピードが10倍向上しました。

GooseFSの無効化

次のコマンドによってGooseFSを無効化できます。

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

デプロイガイド
自作クラスターを使用したデプロイ

最終更新日：：2023-04-06 14:26:08

ここでは主に、スタンドアローン、クラスターおよびTencent Cloud EMRクラスター（現時点ではGooseFSをサポートするバージョンは統合していません）におけるGooseFSのデプロイ、設定および実行を標準化する方法についてご説明します。

デプロイ環境要件

ハードウェア環境

現在GooseFSは主流のx86/x64アーキテクチャのLinux/MacOS実行環境をサポートしています（**注意：MacOS M1プロセッサのサポートは未検証です**）。詳細な実行ノード構成は次のとおりです。

Masterノード

- **CPU**：動作クロック周波数1GHz以上とします。Masterが大量の処理を必要とすることを考慮し、本番環境ではマルチコアアーキテクチャプロセッサを使用することをお勧めします。
- **物理メモリ**：1GB以上とします。本番環境では8GB以上の物理メモリ構成を使用することをお勧めします。
- **ディスク**：20GB以上とします。本番環境ではハイパフォーマンスなNVME SSDディスクをメタデータキャッシュディスクとすることをお勧めします（RocksDBモード）。

Workerノード

- **CPU**：動作クロック周波数1GHz以上とします。Workerノードでも大量の同時実行リクエストを処理する必要があることを考慮し、同様に本番環境ではマルチコアアーキテクチャプロセッサを使用することをお勧めします。
- **物理メモリ**：2GB以上とします。本番環境ではパフォーマンス要件に応じて適切な物理メモリ構成を選択することをお勧めします。例えば、本番環境で大量のデータブロックをWorkerノードのメモリ内にキャッシュする必要がある場合、またはMEM + SSD/HDDの混合ストレージを使用している場合は、実際にメモリにキャッシュする可能性のあるデータ量に応じて物理メモリを配備する必要があります。どのキャッシュモードを使用する場合でも、本番環境ではWorkerノードに16GB以上の物理メモリを配備することをお勧めします。
- **ディスク**：20GB以上のSSD/HDDディスクとします。実際の本番環境でプリフェッチキャッシュを行う必要があるデータ量のサイズに基づいて、各Workerノードに設定する必要があるディスク容量を推計することをお勧めします。また、より良いパフォーマンスを体験するためには、NVMEインターフェースを備えたSSDディスクをWorkerノードのデータディスクとすることをお勧めします。

ソフトウェア環境

- Red Hat Enterprise Linux 5.5+、Ubuntu Linux 12.04 LTS+（バッチデプロイを使用しない場合はサポート可）、CentOS 7.4+およびTLinux 2.x（Tencent Linux 2.x）、Intelベースのアーキテクチャの MacOS 10.8.3以上のバージョン。Tencent Cloud CVMのユーザーにはCentOS 7.4、Tencent（TLinux 2.x）またはTencentOS Server 2.4バージョンのOSの使用を推奨します。
- OpenJDK 1.8/Oracle JDK 1.8。JDK 1.9以上のバージョンのサポートは未検証ですのでご注意ください。
- SSHツールセット（SSHおよびSCPツールを含む）、Expect Shellツール（バッチデプロイを使用する場合に必要）およびYumパッケージ管理ツールをサポートしています。

クラスターネットワーク環境

- Master/Workerノード間はパブリックネットワークまたはプライベートネットワークで相互接続します。バッチデプロイを使用する場合は、Masterがパブリックネットワークのソフトウェアソースに正常にアクセスできるようにするか、またはパッケージ管理システムのプライベートネットワークソフトウェアソースを正確に設定する必要があります。
- クラスターはパブリックネットワークまたはプライベートネットワーク経由でCOSバケットまたはCHDFSファイルシステムにアクセスできるようにします。

セキュリティグループとユーザー権限の要件

通常、GooseFSではユーザーが専用のLinuxアカウントを使用してGooseFSのデプロイと実行を行うことを推奨しています。例えば、自作のクラスターおよびEMR環境では統一してhadoopユーザーを使用し、GooseFSのデプロイと実行を行うことなどが可能です。バッチデプロイでは、以下のユーザー機能および権限を補足する必要があります。

- rootに切り替える、またはsudo権限を使用する機能を備えること
- アカウントにディレクトリの読み取り、書き込み、インストールの権限をデプロイし実行すること
- Masterノードがクラスター内の全WorkerノードにSSHログインできる権限を備えること
- クラスター内の対応するノードのロールは以下の対応するポートを許可する必要があります。Master（9200および9201）、Worker（9203および9204）、Job Master(9205、9206、9207)、Job Worker（9208、9209、9210）、Proxy（9211）、LogServer（9212）。

スタンドアローンモードでのデプロイと実行（疑似分散型アーキテクチャ）

疑似分散型アーキテクチャデプロイは主にGooseFSの体験とデバッグに適しています。初級ユーザーは自身のLinuxまたはMacシステムのホスト上でGooseFSのクイック体験およびデバッグを行うことができます。

デプロイ

1. 初めにGooseFSのバイナリーディストリビューションパッケージをダウンロードします。
2. ディストリビューションパッケージをダウンロードして解凍し、GooseFSのディレクトリに進み、次の操作を実行します。

- conf/goosefs-site.properties.templateをコピーしてconf/goosefs-site.properties設定ファイルを作成します。

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

- conf/goosefs-site.properties設定ファイル内で `goosefs.master.hostname` を `localhost` に設定します。

- conf/goosefs-site.properties設定ファイル内で `goosefs.master.mount.table.root.ufs` をローカルファイルシステム内のディレクトリ（例：/tmpまたは/data/goosefsなど）に設定します。

localhostのパスワードなしでのSSHログインを設定しておくことをお勧めします。これを行わなければ、フォーマット化および起動などの操作の際にlocalhostのログインパスワードを入力する必要があります。

実行

次のコマンドを実行すると、RamFSファイルシステムのマウントが完了します。

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

同様に、上記のコマンドを実行せず、GooseFSクラスターの起動時に直接マウントすることもできます。

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

起動後、 `jps` コマンドを使用すると、疑似分散型モードでGooseFSの全プロセスを表示することができます。

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

その後、 `goosefs` コマンドラインを使用し、namespace、fileSystem、job、tableの各操作を実行できます。例えば、あるローカルファイルをGooseFSにアップロードし、現在のGooseFS内のルートディレクトリ下のファイルおよびディレクトリをリストアップするなどです。

```
$ 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
```

GooseFSのコマンドラインは、GooseFSの管理とアクセスを行うすべてのコマンドラインインターフェースをユーザーに提供しています。それにはネームスペース（namespace）、テーブル（table）、ジョブ（job）および一般的なファイルシステム（fs）操作などが含まれます。具体的には公式ドキュメントをご参照いただくか、または `goosefs -h` コマンドラインパラメータを使用して詳細なヘルプ情報を取得することが可能です。

クラスターモードでのデプロイと実行

クラスターデプロイと実行は主にユーザーが自作したIDCクラスターの本番環境、またはGooseFSを統合していないTencent Cloud EMRの本番環境向けのものです。具体的にはStandaloneアーキテクチャデプロイと高可用性（HA）アーキテクチャデプロイがあります。

GooseFSは `scripts` ディレクトリ下でパスワードなしSSHログインの一括設定およびGooseFSの一括インストールとデプロイを行うスクリプトツールを提供し、ユーザーがGooseFSの大規模クラスターのインストールとデプロイを便利でスピーディーに行えるようにしています。ユーザーは前章で述べたデプロイ環境要件のバッチデプロイ条件をご自身でチェックし、バッチデプロイが実行可能かどうかを柔軟に選択することができます。

バッチデプロイツールの紹介

GooseFSは `scripts` ディレクトリ下でパスワードなしSSHログインの一括設定およびGooseFSのバッチデプロイとインストールを行うスクリプトツールを提供し、ユーザーが実行条件を満たす前提で、次の手順に従ってバッチデプロイタスクを完了できるようにしています。

- GooseFS解凍ディレクトリ、 `cd ${GOOSEFS_HOME}` に進みます。
- `conf/masters` および `conf/workers` 内のホスト名またはIPアドレスを設定します。
- その後 `scripts` ディレクトリに進み、 `commons.sh` というスクリプトファイルの `SCRIPT_EXEC_USER` および `SCRIPT_EXEC_PASSWORD` を正しく入力します。再び `root` アカウントに切り替えるか、または `sudo` 権限で `config_ssh.sh` を実行し、クラスター全体のパスワードなしログイン設定を完了します。
- パスワードなしログイン設定の完了後、 `validate_env.sh` ツールを実行してクラスター全体の設定状態をチェックすることができます。
- 設定ファイルディレクトリのソフトリンク、 `ln -s conf ~/.goosefs` を作成します。
- 最後に `goosefs copy .` および `goosefs copy ~/.goosefs` を実行し、goosefsのバイナリーパッケージおよび設定ファイルを全ノードに配信します。

デプロイ全体のフローがすべて完了した後、 `./bin/goosefs-start.sh all SudoMount` を実行してクラスター全体を起動し実行することができます。デフォルト設定では、すべての実行ログは `${GOOSEFS_HOME}/logs` に記録されます。

Standaloneアーキテクチャデプロイ

Standaloneアーキテクチャはシングルmasterノード、マルチWorkerノードのクラスターデプロイアーキテクチャを採用しています。具体的には下記の手順を参照してデプロイおよび実行することができます。

1. GooseFSのバイナリーディストリビューションパッケージをダウンロードします。
2. `tar zxvf goosefs-x.x.x-bin.tar.gz` コマンドを使用して、インストールパスの後に解凍します。
   バッチデプロイツールの紹介を参照してクラスターのバッチデプロイを設定および実行することができます。
   もしくは下記の詳細な手動デプロイフローを引き続き参照することも可能です。

（1） `conf` ディレクトリ下から `template` ファイルをコピーして設定ファイルを作成します。

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

（2） `goosefs-site.properties` 設定ファイル内で次の設定を指定します。

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

このうち、 `goosefs.master.hostname` はmasterノードのhostnameまたはipとして設定します。 `goosefs.master.mount.table.root.ufs` はGooseFSルートディレクトリがマウントする基盤ファイルシステム（UFS）のパスURIを指定します。

> 注意：
> このURIは必ずMasterおよびWorkerノードのどちらからもアクセス可能でなければならないため、ローカルディレクトリはサポートしていません。

例えば、あるCOSパスをGooseFSのルートパス：goosefs.master.mount.table.root.ufs=cosn://bucket-1250000000/goosefs/にマウントすることができます。

`masters` 設定ファイルでシングルMasterノードのhostnameまたはipを指定します。例えば次のようになります。

```
# 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` 設定ファイルで全Workerノードのhostまたはipを指定します。例えば次のようになります。

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

すべての設定が完了してから `./bin/goosefs copyDir conf/` を実行すると、全ノードに設定を同期することができます。

最後に `./bin/goosefs-start.sh all` を実行すると、GooseFSクラスターを起動できます。

高可用性アーキテクチャのデプロイ

シングルMasterノードのStandaloneアーキテクチャでは単一点障害の問題が発生しやすいため、実際の本番環境ではマルチMasterノードをデプロイして可用性の高いシステムアーキテクチャを得ることをお勧めします。複数のMasterノードのうち1つだけがリーダー（Leader）ノードとして外部にサービスを提供し、その他のスタンバイ（Standby）ノードは共有ログであるジャーナル（Journal）を同期することでリーダーノードと同一のファイルシステムの状態を維持します。リーダーノードに障害が発生してダウンすると、現在のスタンバイノードの中から1つが自動的に選択され、リーダーノードに代わって外部へのサービス提供を行います。こうすることでシステムの単一点障害を解消し、全体的な高可用性アーキテクチャを実現することができます。

現在GooseFSはRaftログとZookeeperという2種類の方式をベースにしてリーダーノードとスタンバイノードの状態の強整合性を実現しています。この2つのデプロイ方式について以下でそれぞれご説明します。

Raftの埋め込みログに基づく高可用性デプロイ設定

初めに設定テンプレートに従って設定ファイルを作成します。

```
$ 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>
```

説明：
上記の設定選択項目の説明は次のとおりです。

- `goosefs.master.mount.table.root.ufs` はGooseFSルートディレクトリにマウントする基盤ストレージURIとして設定します。
- `goosefs.master.embedded.journal.addresses` はすべてのスタンバイノードの `ip:embedded_journal_port` または `host:embedded_journal_port` として設定します。このうち、embedded_journal_portはデフォルトでは9202です。例えば、192.168.1.1:9202,192.168.1.2:9202,192.168.1.3:9202のようになります。

Raftの埋め込みログに基づくデプロイ方法はcopycatのLeader選出メカニズムに依存します。このため、RaftのHAデプロイアーキテクチャはZookeeperと相互に併用することはできません。

すべての設定が完了した後、同様に以下のコマンドを使用してすべての設定を同期します。

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

最後にフォーマット化を完了すると、GooseFSクラスターの起動が可能になります。

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

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

権限が不足していると表示された場合は、次の方法での再起動をお試しください。

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

次のコマンドを使用すると、現在のリーダー（Leader）ノードを確認できます。

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

次のコマンドを使用してクラスターの状態を確認することもできます。

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

Zookeeperのジャーナルに基づくデプロイ設定

Zookeeperサービスを設定してGooseFSの高可用性アーキテクチャを構築するには、次の条件を満たしている必要があります。

- Zookeeperクラスター、GooseFS MasterはZookeeperを使用してLeaderを選出し、GooseFSのクライアントとWorkerノードはZookeeperによってリーダーMasterノードを照会していること。
- 高可用性、強整合性の共有ストレージシステムで、なおかつすべてのGooseFSのMasterノードがアクセス可能であること。リーダーMasterノードはこのストレージシステムにログを書き込み、スタンバイ（Standby）

ノードは常に共有ストレージシステムからログを読み取り、リプレイを行うことでリーダーノードの状態との整合性を維持します。通常、この共有ストレージシステムはHDFSまたはCOSに設定することを推奨します。例えば、`hdfs://10.0.0.1:9000/goosefs/journal` または `cosn://bucket-1250000000/journal` のようになります。

設定は次のとおりです。

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

ZKモードの `JOURNAL_URI` は必ず共有ストレージシステムのURIでなければならない点に注意が必要です。その後、`./bin/goosefs copyDir conf/` を使用して、設定をクラスター内の全ノードに同期配信します。最後にクラスター `./bin/goosefs-start.sh all` を起動すれば完了です。

GooseFSのプロセスリスト

`goosefs-start.sh all` スクリプトを実行してGooseFSを起動後、クラスターには次のプロセスが含まれます。

プロセス	説明
GooseFSMaster	デフォルトのRPCポートは9200、Webポートは9201
GooseFSWorker	デフォルトのRPCポートは9203、Webポートは9204
GooseFSJobMaster	デフォルトのRPCポートは9205、Webポートは9206
GooseFSProxy	デフォルトのWebポートは9211
GooseFSJobWorker	デフォルトのRPCポートは9208、Webポートは9210