云 HDFS

实践教程

腾讯云

文档目录

实践教程
使用 CHDFS 作为 Druid 的 Deep storage

最近更新时间：2024-10-12 20:45:01

环境依赖

- CHDFS_JAR
- Druid 版本：Druid-0.12.1

下载与安装

获取 CHDFS JAR

在官方 Github 上下载 CHDFS_JAR 。

安装 CHDFS JAR

使用 CHDFS 作为 Druid 的 Deep Storage，需要借助 Druid-hdfs-extension 实现。
下载 CHDFS JAR 后，将 `chdfs_hadoop_plugin_network-1.7.jar` 拷贝到 Druid 安装路径 `extensions/druid-hdfs-storage` 以及 `hadoop-dependencies/hadoop-client/2.x.x` 下。

使用方法

配置修改

1. 修改 Druid 安装路径的 `conf/druid/_common/common.runtime.properties` 文件，将 hdfs 的 extension 加入到 `druid.extensions.loadList` 中，同时指定 hdfs 为 Druid 的 deep storage，而路径则填写为 CHDFS 的路径：

```
properties
druid.extensions.loadList=["druid-hdfs-storage"]
druid.storage.type=hdfs
druid.storage.storageDirectory=ofs://<mountpoint>/<druid-path>
```

2. 在 `conf/druid/_common/` 这个目录下，新建一个 hdfs 的配置文件 hdfs-site.xml，填入 CHDFS 的配置信息等：

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at
 http://www.apache.org/licenses/LICENSE-2.0
  Unless required by applicable law or agreed to in writing, software
```

```
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
 <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>
 <!--本地 cache 的临时目录，对于读写数据，当内存 cache 不足时会写入本地硬盘，这个路径若
不存在会自动创建-->
 <property>
 <name>fs.ofs.tmp.cache.dir</name>
 <value>/data/chdfs_tmp_cache</value>
 </property>
 <!--appId 用户需要换成自己的 appid，可前往
https://console.cloud.tencent.com/cam/capi 获取-->
 <property>
 <name>fs.ofs.user.appid</name>
 <value>125000001</value>
 </property>
</configuration>
```

上述配置的支持项与 CHDFS 官网文档描述完全一致，详情可参见 挂载 CHDFS 文档。

开始使用

依次启动 Druid 进程，Druid 数据就可加载到 CHDFS 中。

原生 HDFS 数据迁移到腾讯云 CHDFS

最近更新时间：2024-10-12 20:45:01

准备工作

1. 在腾讯云官网创建 CHDFS 文件系统和 CHDFS 挂载点，配置好权限信息。
2. 通过腾讯云 VPC 环境的 CVM 访问创建好的 CHDFS，详情请参见 创建 CHDFS 。
3. 当挂载成功后，打开 hadoop 命令行工具，执行以下命令，验证 CHDFS 功能是否正常。

```
hadoop fs -ls ofs://f4xxxxxxxxxxxxxx.chdfs.ap-beijing.myqcloud.com/
```

如果能看到以下类似的输出，则表明云 HDFS 功能一切正常。

```
[hadoop@10 ~]$ hadoop fs -ls ofs://[redacted].chdfs.ap-beijing.myqcloud.com/
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/service/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/service/tez/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 31 items
drwxr-xr-x   - root   root          0 2019-12-30 17:03 ofs://[redacted]chdfs.ap-beijing.myqcloud.com/0x
drwxr-xr-x   - hadoop hadoop        0 2019-12-30 18:20 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/data
-rw-r--r--   1 root   root  1048576000 2019-12-13 23:20 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/dd_1G
drwxrwxrwx   - hadoop hadoop        0 2019-12-18 12:08 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/emr
drwxrwxr-x   - root   root          0 2019-12-05 17:13 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-dir-0
drwxrwxr-x   - root   root          0 2019-12-05 17:13 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-dir-1
drwxrwxr-x   - root   root          0 2019-12-05 17:13 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-dir-2
drwxrwxr-x   - root   root          0 2019-12-05 17:24 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-dir-3
drwxr-xr-x   - root   root          0 2019-12-05 17:25 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-dir-4
-rwxrwxr-x   1 root   root          0 2019-12-05 17:13 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-file-0
-rw-rw-r--   1 root   root          0 2019-12-05 17:13 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-file-1
```

迁移

使用 COSDistcp 工具迁移

COSDistcp 工具是由 COS 团队研发的适用于对象存储和 HDFS 系统之间进行数据高效传输的工具，针对对象存储系统和 HDFS 系统之间的差异，COS 团队对该工具进行了许多的优化和改进，其中包括：

- 跨系统之间的数据 CRC 在线校验
- 小文件性能
- 增量复制拷贝

更多工具详情，请参见 COSDistcp 工具文档 。
当准备工作就绪后，即可使用 COSDistcp 工具进行数据迁移。COSDistcp 是一个 Jar 包工具，依赖 Hadoop 的 MapReduce 框架来执行。
执行命令提交 COSDistcp 程序到 Hadoop 系统。例如：

```
hadoop jar cos-distcp-1.6-2.8.5.jar -Dmapred.job.queue.name=root.users.presto -
-src /user/hive/warehouse/dw.db/logbak/ --srcPrefixesFile
file:///home/hadoop/filebeat_gaotu_service0000 --dest ofs://f4xxxxxxxx-
xxxx.chdfs.ap-beijing.myqcloud.com/user/hive/warehouse/dw.db/logbak/ --
taskNumber=25 --workerNumber=10 --bandWidth=10 &

#具体参数可以参考 COSDistcp 工具的文档
```

其中 `f4xxxxxxxx-xxxx.chdfs.ap-beijing.myqcloud.com` 为挂载点域名，需要根据实际申请的挂载点信息进行替换。

使用 Distcp 工具迁移

当准备工作就绪后，也可以使用 hadoop 社区标准的 Distcp 工具实现全量或者增量的 HDFS 数据迁移，详情请参见 Distcp 官方指引文档 。

注意事项

1. 在 hadoop distcp 工具中，提供了一些 CHDFS 不兼容的参数。如果指定如下表格中的一些参数，则不生效。

参数	描述	状态
-p[rbax]	r：replication，b：block-size，a：ACL，x：XATTR	不生效

2. 由于 Hadoop 2.x 中的 HDFS 系统的 CRC 计算方式和对象存储文件的 CRC 计算方式不一致，导致在迁移过程中无法利用 crccheck 来对数据进行在线迁移校验。因此，在迁移过程中，一般都需要加上-skipcrccheck 选项。
如果需要校验迁移后的数据是否完整，需要借助 COS 研发的 COS 离线校验工具 进行离线校验。
Hadoop 3.1.1版本及以上，可以采用 COMPOSITE_CRC 算法进行在线校验，示例如下：

```
hadoop distcp  -Ddfs.checksum.combine.mode=COMPOSITE_CRC -checksum
hdfs://10.0.1.11:4007/testcp ofs://f4xxxxxxxx-xxxx.chdfs.ap-
beijing.myqcloud.com/
```

示例说明

1. 当 CHDFS 准备就绪后，执行以下 hadoop 命令进行数据迁移。

```
hadoop distcp hdfs://10.0.1.11:4007/testcp ofs://f4xxxxxxxx-xxxx.chdfs.ap-
beijing.myqcloud.com/
```

其中 `f4xxxxxxxx-xxxx.chdfs.ap-beijing.myqcloud.com` 为挂载点域名，需要根据实际申请的挂载点信息进行替换。

2. Hadoop 命令执行完毕后，会在日志中打印出本次迁移的具体详情。如下示例所示：

```
2019-12-31 10:59:31 [INFO ] [main:13300] [org.apache.hadoop.mapreduce.Job:]
[Job.java:1385]
 Counters: 38
 File System Counters
     FILE: Number of bytes read=0
     FILE: Number of bytes written=387932
     FILE: Number of read operations=0
     FILE: Number of large read operations=0
     FILE: Number of write operations=0
     HDFS: Number of bytes read=1380
     HDFS: Number of bytes written=74
```

```
        HDFS: Number of read operations=21
        HDFS: Number of large read operations=0
        HDFS: Number of write operations=6
        OFS: Number of bytes read=0
        OFS: Number of bytes written=0
        OFS: Number of read operations=0
        OFS: Number of large read operations=0
        OFS: Number of write operations=0
    Job Counters
        Launched map tasks=3
        Other local map tasks=3
        Total time spent by all maps in occupied slots (ms)=419904
        Total time spent by all reduces in occupied slots (ms)=0
        Total time spent by all map tasks (ms)=6561
        Total vcore-milliseconds taken by all map tasks=6561
        Total megabyte-milliseconds taken by all map tasks=6718464
    Map-Reduce Framework
        Map input records=3
        Map output records=2
        Input split bytes=408
        Spilled Records=0
        Failed Shuffles=0
        Merged Map outputs=0
        GC time elapsed (ms)=179
        CPU time spent (ms)=4830
        Physical memory (bytes) snapshot=1051619328
        Virtual memory (bytes) snapshot=12525191168
        Total committed heap usage (bytes)=1383071744
    File Input Format Counters
        Bytes Read=972
    File Output Format Counters
        Bytes Written=74
    org.apache.hadoop.tools.mapred.CopyMapper$Counter
        BYTESSKIPPED=5
        COPY=1
        SKIP=2
```

使用 DataX 导入或导出 CHDFS

最近更新时间：2024-10-12 20:45:01

环境依赖

- CHDFS_JAR
- DataX 版本：DataX-3.0

下载与安装

获取 CHDFS JAR

在官方 Github 上下载 CHDFS_JAR 。

获取 DataX 软件包

在官方 Github 上下载 DataX 。

安装 CHDFS JAR

下载 CHDFS JAR 后，将 `chdfs_hadoop_plugin_network-1.7.jar` 拷贝到 Datax 解压路径 `plugin/reader/hdfsreader/libs/` 以及 `plugin/writer/hdfswriter/libs/` 下。

使用方法

DataX 配置

修改 datax.py 脚本

打开 DataX 解压目录下的 bin/datax.py 脚本，修改脚本中的 CLASS_PATH 变量为如下：

```
CLASS_PATH =
("%s/lib/*:%s/plugin/reader/hdfsreader/libs/*:%s/plugin/writer/hdfswriter/libs/
*:.") % (DATAX_HOME, DATAX_HOME, DATAX_HOME)
```

在配置 JSON 文件里配置 hdfsreader 和 hdfswriter

示例 JSON 如下：

```
{
    "job": {
        "setting": {
            "speed": {
                "byte": 10485760
            },
            "errorLimit": {
                "record": 0,
                "percentage": 0.02
```

```
            }
        },
        "content": [{
            "reader": {
                "name": "hdfsreader",
                "parameter": {
                    "path": "testfile",
                    "defaultFS": "ofs://f4xxxxxxxx-hxT9.chdfs.ap-
beijing.myqcloud.com/",
                    "column": ["*"],
                    "fileType": "text",
                    "encoding": "UTF-8",
                    "hadoopConfig": {
                        "fs.AbstractFileSystem.ofs.impl":
"com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter",
                        "fs.ofs.impl":
"com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter",
                        "fs.ofs.tmp.cache.dir": "/data/chdfs_tmp_cache",
                        "fs.ofs.user.appid": "1250000000"
                    },
                    "fieldDelimiter": ","
                }
            },
            "writer": {
                "name": "hdfswriter",
                "parameter": {
                    "path": "/user/hadoop/",
                    "fileName": "testfile1",
                    "defaultFS": "ofs://f4xxxxxxxx-hxT9.chdfs.ap-
beijing.myqcloud.com/",
                    "column": [{
                            "name": "col",
                            "type": "string"
                        },
                        {
                            "name": "col1",
                            "type": "string"
                        },
                        {
                            "name": "col2",
                            "type": "string"
                        }
                    ],
                    "fileType": "text",
                    "encoding": "UTF-8",
                    "hadoopConfig": {
```

```
                        "fs.AbstractFileSystem.ofs.impl":
"com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter",
                        "fs.ofs.impl":
"com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter",
                        "fs.ofs.tmp.cache.dir": "/data/chdfs_tmp_cache",
                        "fs.ofs.user.appid": "1250000000"
                    },
                    "fieldDelimiter": ":",
                    "writeMode": "append"
                }
            }
        }]
    }
}
```

其中，hadoopConfig 配置为 CHDFS 所需要的配置，defaultFS 填写为 CHDFS 的路径。例如 `ofs://f4xxxxxxxxx-hxT9.chdfs.ap-beijing.myqcloud.com/` ，其他配置同 hdfs 配置项即可。

执行数据迁移

将配置文件保存为 hdfs_job.json，存放到 job 目录下，执行以下命令行：

```
bin/datax.py job/hdfs_job.json
```

观察屏幕正常输出如下：

```
2020-03-09 16:49:59.543 [job-0] INFO  JobContainer -
         [total cpu info] =>
                averageCpu                     | maxDeltaCpu
| minDeltaCpu
                -1.00%                         | -1.00%
| -1.00%


         [total gc info] =>
                 NAME                 | totalGCCount       | maxDeltaGCCount
| minDeltaGCCount    | totalGCTime        | maxDeltaGCTime     | minDeltaGCTime
                 PS MarkSweep         | 1                  | 1
| 1                  | 0.024s             | 0.024s             | 0.024s
                 PS Scavenge          | 1                  | 1
| 1                  | 0.014s             | 0.014s             | 0.014s

2020-03-09 16:49:59.543 [job-0] INFO  JobContainer - PerfTrace not enable!
2020-03-09 16:49:59.543 [job-0] INFO  StandAloneJobContainerCommunicator -
Total 2 records, 33 bytes | Speed 3B/s, 0 records/s | Error 0 records, 0 bytes
```

```
|  All Task WaitWriterTime 0.000s |  All Task WaitReaderTime 0.033s |
Percentage 100.00%
2020-03-09 16:49:59.544 [job-0] INFO  JobContainer -
任务启动时刻                    : 2020-03-09 16:49:48
任务结束时刻                    : 2020-03-09 16:49:59
任务总计耗时                    :                 11s
任务平均流量                    :               3B/s
记录写入速度                    :              0rec/s
读出记录总数                    :                   2
读写失败总数                    :                   0
```

使用 Python 访问腾讯云 CHDFS

最近更新时间：2024-10-12 20:45:01

背景

本文指导如何使用 Python 的工具包 pyarrow 操作 CHDFS。

部署环境

1. Python 3.7版本及以上。PyArrow 目前与 Python 3.7、3.8、3.9 和 3.10 兼容。
2. 使用如下命令，安装 pyarrow 库：

```
pip3 install pyarrow -image
pip3 install pyarrow -image -i http://mirrors.tencent.com/pypi/simple --
trusted-host mirrors.tencent.com
```

部署组件

部署 CHDFS 插件的方法，请参见 挂载 CHDFS 。

编写 Python 程序

1. 使用 pyarrow 访问 CHDFS，示例代码如下：

```
import pyarrow as pa

host = "ofs://xxx-xxx.chdfs.ap-guangzhou.myqcloud.com"
fs = pa.hdfs.connect(host, 0)
# open（path, mode）模式 w,文件不存在创建一个文件
out_file = fs.open("ofs://xxx-xxx.chdfs.ap-
guangzhou.myqcloud.com/ppyarrow.txt","wb")
out_file.write(str.encode("hello world, pyarrow")) #写
out_file.close()

in_file = fs.open("ofs://xxx-xxx.chdfs.ap-
guangzhou.myqcloud.com/ppyarrow.txt","rb")
# 将光标重置到起始位置
in_file.seek(0)
data = in_file.read() # 读
print("写入的数据为%s."%(data))
in_file.close()

# 列出文件
ls_file = fs.ls("ofs://xxx-xxx.chdfs.ap-guangzhou.myqcloud.com/")
print("目录文件为%s." %(ls_file))
```

```
# 创建目录
fs.mkdir("ofs://xxx-xxx.chdfs.ap-guangzhou.myqcloud.com/pyarrowtest")
# 移动并重命名文件
fs.mv("ofs://xxx-xxx.chdfs.ap-guangzhou.myqcloud.com/ppyarrow.txt",
"ofs://xxx-xxx.chdfs.ap-guangzhou.myqcloud.com/pyarrowtest/tina.txt")
# 列出文件
mv_file = fs.ls("ofs://xxx-xxx.chdfs.ap-guangzhou.myqcloud.com/pyarrowtest")
print("移动后的目录文件为%s." %(mv_file))

# 删除测试文件，重新列出文件
fs.delete("ofs://xxx-xxx.chdfs.ap-
guangzhou.myqcloud.com/pyarrowtest/tina.txt")
de_file = fs.ls("ofs://xxx-xxx.chdfs.ap-guangzhou.myqcloud.com/pyarrowtest/")
print("删除文件后的pyarrowtest目录下文件为%s." %(de_file))
```

2. 设置环境变量，示例如下：

```
export JAVA_HOME=/usr/local/jdk    #设置 JAVA_HOME，根据自己安装位置定
export HADOOP_HOME=/usr/local/service/hadoop  #设置 HADOOP_HOME，hadoop 的安装
位置
export CLASSPATH=`$HADOOP_HOME/bin/hadoop classpath --glob`
#参考网址 https://arrow.apache.org/docs/python/filesystems.html#hadoop-file-
system-hdfs
```

3. 执行 Python 文件：

```
python3 libtest.py
```

4. 执行过程及结果如下：

```
SLF4J: Found binding in [jar:file:/usr/local/service/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder
.class]
SLF4J: Found binding in [jar:file:/usr/local/service/hadoop/share/hadoop/kms/tomcat/webapps/kms/WEB-INF/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j
/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/service/hadoop/share/hadoop/httpfs/tomcat/webapps/webhdfs/WEB-INF/lib/slf4j-log4j12-1.7.10.jar!/or
g/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
写入的数据为b'hello world, pyarrow'.
目录文件为['ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/CosTable', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/cos_json_
apache.json', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/cos_json_openx.json', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud
.com/cos_txt.txt', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/emr', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/hadoop
.log', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/henryswang.txt', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/henrysw
ang.txt_2', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/henryswang.txt_3', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/
hivetest', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/ppyarrow.txt', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/teras
ort', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/testfile1', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/tinazzheng.tx
t', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/tmp', 'ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/user'].
移动后的目录文件为['ofs://[redacted].chdfs.ap-guangzhou.myqcloud.com/pyarrowtest/tina.txt'].
删除文件后的pyarrowtest目录下文件为[].
[hadoop@172 ~]$
```

相关参考

- Filesystem Interface
- Building Python and OpenSSL from source, but ssl module fails

CDH 配置 CHDFS 指引

最近更新时间：2025-06-05 18:48:32

简介

CDH（Cloudera's Distribution, including Apache Hadoop）是业界流行的 Hadoop 发行版本。本文指导如何在 CDH 环境下使用腾讯云 CHDFS 服务，以实现大数据计算与存储分离，提供灵活及低成本的大数据解决方案。
CHDFS 大数据组件支持情况如下：

组件名称	CHDFS 大数据组件支持情况	服务组件是否需要重启
Yarn	支持	重启 NodeManager
Yarn	支持	重启 NodeManager
Hive	支持	重启 HiveServer 及 HiveMetastore
Spark	支持	重启 NodeManager
Sqoop	支持	重启 NodeManager
Presto	支持	重启 HiveServer 及 HiveMetastore 和 Presto
Flink	支持	否
Impala	支持	否
EMR	支持	否
自建组件	后续支持	无
HBase	不推荐	无

版本依赖

本文依赖的组件版本如下：

- CDH 5.16.1
- Hadoop 2.6.0

使用方法

存储环境配置

1. 登录 CDH 管理页面。

2. 在系统主页，选择**配置** > **服务范围** > **高级**，进入高级配置代码段页面，如下图所示：

3. 在 Cluster-wide Advanced Configuration Snippet(Safety Valve) for core-site.xml 的代码框中，填入 CHDFS 配置。

```
<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>
<!--本地 cache 的临时目录，对于读写数据，当内存 cache 不足时会写入本地硬盘，这个路径若不存在会自动创建-->
<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>
```

以下为必选的 CHDFS 配置项（需添加到 core-site.xml 中），CHDFS 其他配置可参见 挂载 CHDFS 。

CHDFS 配置项	值	含义
fs.ofs.user.appid	1250000000	用户 appid
fs.ofs.tmp.cache.dir	/data/emr/hdfs/tmp/chdfs/	本地 cache 的临时目录
fs.ofs.impl	com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter	chdfs 对 FileSystem 的实现类，固定为 com.qcloud.chdfs.fs.CHDFSHadoopFileSystemAdapter
fs.AbstractFileSystem.ofs.impl	com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter	chdfs 对 AbstractFileSystem 的实现类，固定为 com.qcloud.chdfs.fs.CHDFSDelegateFSAdapter

4. 对 HDFS 服务进行操作，单击部署客户端配置，此时以上 core-site.xml 配置会更新到集群里的机器上。
5. 将 CHDFS 最新的 SDK 包，放置到 CDH HDFS 服务的 jar 包路径下，请根据实际值进行替换，示例如下：

```
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：**
> 在集群中的每台机器都需要在相同的位置放置 SDK 包。

数据迁移

使用 Hadoop Distcp 工具将 CDH HDFS 数据迁移到 CHDFS，详情请参见 原生 HDFS 数据迁移到腾讯云 CHDFS 。

大数据套件使用 CHDFS

1. MapReduce

操作步骤

（1）按照 数据迁移 章节，配置好 HDFS 的相关配置，并将 CHDFS 的 SDK jar 包，放置到 HDFS 相应的目录。
（2）在 CDH 系统主页，找到 YARN，重启 NodeManager 服务（TeraGen 命令可以不用重启，但是 TeraSort 由于业务内部逻辑，需要重启 NodeManager ，建议都统一重启 NodeManager 服务）。

示例

下面以 Hadoop 标准测试中的 TeraGen 和 TeraSort 为例：

```
hadoop jar ./hadoop-mapreduce-examples-2.7.3.jar teragen -Dmapred.map.tasks=4
1099 ofs://examplebucket-1250000000/teragen_5/

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

说明：
`ofs://schema` 后面请替换为用户 CHDFS 的挂载点路径。

2. Hive

2.1 MR 引擎

操作步骤
（1）按照 数据迁移 章节，配置好 HDFS 的相关配置，并且将 CHDFS 的 SDK jar 包，放置到 HDFS 相应的目录。
（2）在 CDH 主页面，找到 HIVE 服务， 重启 Hiveserver2 及 HiverMetastore 角色。
示例
某用户的真实业务查询，例如执行 Hive 命令行，创建一个 Location，作为在 CHDFS 上的分区表：

```
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_credit_detail_grant_dai
ly'
TBLPROPERTIES (
  'last_modified_by'='work',
  'last_modified_time'='1589310646',
  'transient_lastDdlTime'='1589310646')
```

执行 sql 查询：

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

观察结果如下：

```
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)
```

2.2 Tez 引擎

Tez 引擎需要将 CHDFS 的 jar 包导入到 Tez 的压缩包内，下面以 apache-tez.0.8.5 为例进行说明：

操作步骤

（1）找到 CDH 集群安装的 tez 包，然后解压，例如/usr/local/service/tez/tez-0.8.5.tar.gz。

（2）将 CHDFS 的 jar 包放置到解压后的目录下，然后重新压缩输出一个压缩包。

（3）将新的压缩包上传到 tez.lib.uris 指定的路径下（如果之前存在路径则直接替换即可）。
（4）在 CDH 主页面，找到 HIVE，重启 hiveserver 和 hivemetastore。

3. Spark

操作步骤

（1）按照 数据迁移 章节，配置好 HDFS 的相关配置，并且将 CHDFS 的 SDK jar 包，放置到 HDFS 相应的目录。
（2）重启 NodeManager 服务。

示例

以 CHDFS 进行 Spark example word count 测试为例。

```
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 ofs://examplebucket-1250000000/wordcount
```

执行结果如下：

```
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
```

4. Sqoop

操作步骤

（1）按照 数据迁移 章节，配置好 HDFS 的相关配置，并且将 CHDFS 的 SDK jar 包，放置到 HDFS 相应的目录。
（2）CHDFS 的 SDK jar 包还需要放到 sqoop 目录下（例如/opt/cloudera/parcels/CDH-5.16.1-1.cdh5.16.1.p0.3/lib/sqoop/）。
（3）重启 NodeManager 服务。

示例

以导出 MYSQL 表到 CHDFS 为例，可参考 关系型数据库和 HDFS 的导入导出 文档进行测试。

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

执行结果如下：

```
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
```

5. Presto

操作步骤

（1）按照 数据迁移 章节，配置好 HDFS 的相关配置，并且将 CHDFS 的 SDK jar 包，放置到 HDFS 相应的目录。

（2）CHDFS 的 SDK jar 包还需要放到 presto 目录下（例如/usr/local/services/cos_presto/plugin/hive-hadoop2）。

（3）由于 presto 不会加载 hadoop common 下的 gson-2...jar，需将 gson-2...jar 也放到 presto 目录下（例如/usr/local/services/cos_presto/plugin/hive-hadoop2，仅 CHDFS 依赖 gson）。

（4）重启 HiveServer、HiveMetaStore 和 Presto 服务。

示例

以 HIVE 创建 Location 为 CHDFS 的表查询为例：

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

> **说明：**
> chdfs_test_table 为 location 是 ofs scheme 的表。

查询结果如下：

```
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
```

CHDFS Ranger 权限体系解决方案

最近更新时间：2025-01-25 10:09:02

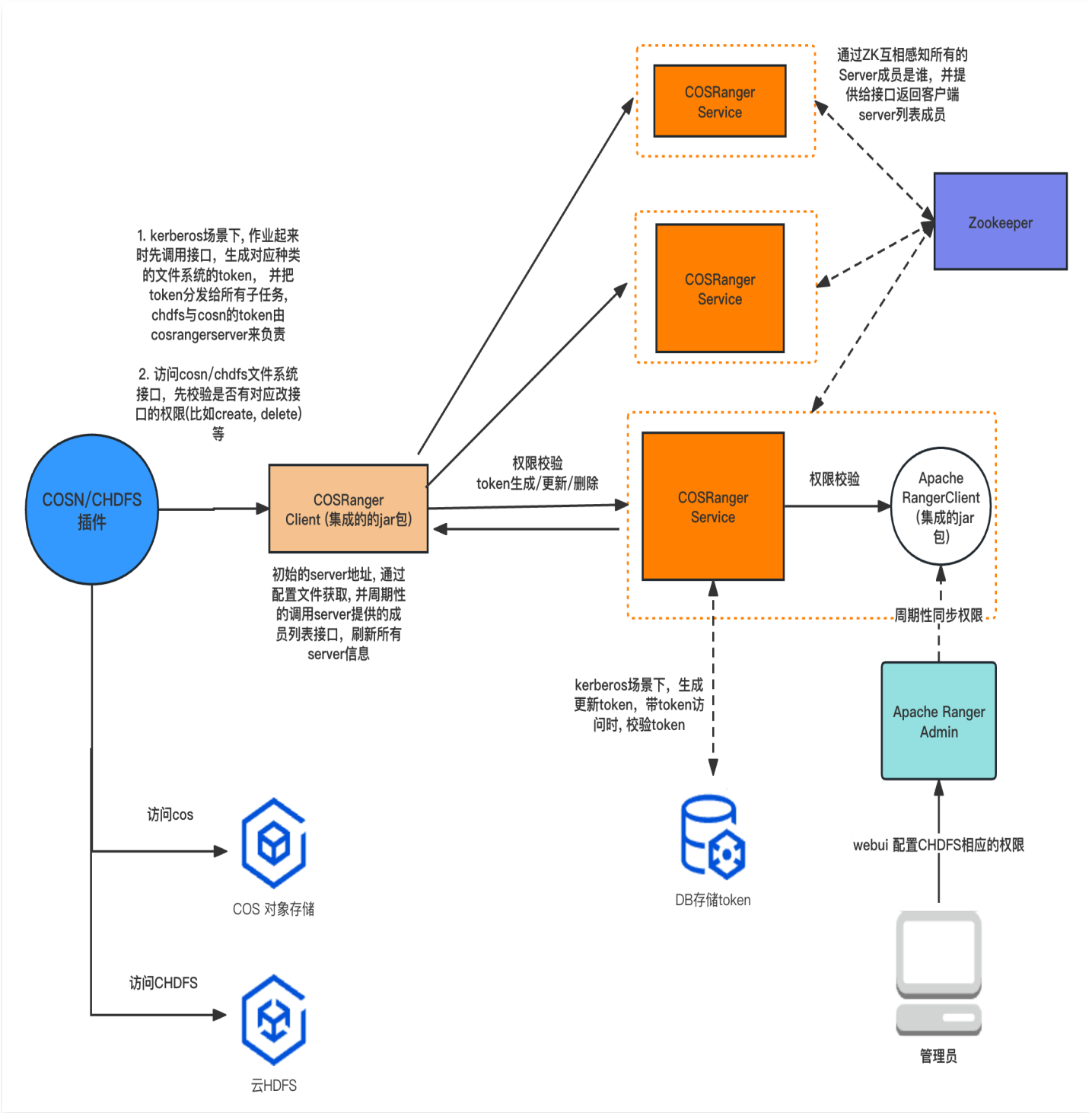
背景

大数据用户使用存算分离后，将数据托管在云 HDFS（Cloud HDFS，CHDFS）上。CHDFS 提供了类似 HDFS 的权限体系管控。Hadoop Ranger 在 HDFS 权限基础上，提供了更精细的权限管控，包括用户组权限设置，针对某个前缀的权限设置。同时 Hadoop Ranger 作为一站式的权限体系解决方案， 不仅支持存储端权限管控，还支持 YARN，Hive 等组件权限管控。因此，为了维持方便客户的使用习惯，我们提供了 CHDFS 的 Ranger 接入解决方案，方便客户使用 Ranger 来进行 CHDFS 的权限管控。

优势

- 细粒度的权限控制，兼容 Hadoop 权限习惯。
- 用户统一管理大数据组件与云端托管存储的权限。

解决方案架构



Hadoop 权限体系中，认证由 Kerberos 提供，授权鉴权由 Ranger 负责。在此基础上，我们提供以下组件，来支持 CHDFS 的 Ranger 权限方案。

- CHDFS-Ranger-Plugin：提供 Ranger 服务端的服务定义插件。它们提供了 Ranger 侧的 CHDFS 服务描述，部署了该插件后，用户即可在 Ranger 的控制页面上，填写相应的权限策略。
- COSRangerService：该服务集成了 Ranger 的客户端，周期性从 Ranger 服务端同步权限策略，在收到客户的鉴权请求后，在本地进行权限校验。 同时它提供了 Hadoop 中 DelegationToken 相关的生成，续租等接口，所有的接口都是通过 Hadoop IPC 定义。

- CosRangerClient：COSN/CHDFS 插件对其进行动态加载，把权限校验的请求转发给 CosRangerService。

部署环境

- Hadoop 环境。
- ZooKeeper、Ranger、Kerberos 服务（如果有认证需求，则部署）。

> **说明**
> 以上服务由于是成熟的开源组件，因此客户可自行安装。

部署组件

部署组件请按照 CHDFS-Ranger-Plugin、Cos-Ranger-Service、Cos-Ranger-Client、CHDFS 次序进行。

部署 CHDFS-Ranger-Plugin

CHDFS-Ranger-Plugin 拓展了 Ranger Admin 控制台上的服务种类， 用户可在 Ranger 控制台上，设置和 CHDFS 相关的操作权限。

代码地址

可前往 Github 的 ranger-plugin 目录下获取。

版本

V1.0版本及以上。

部署步骤

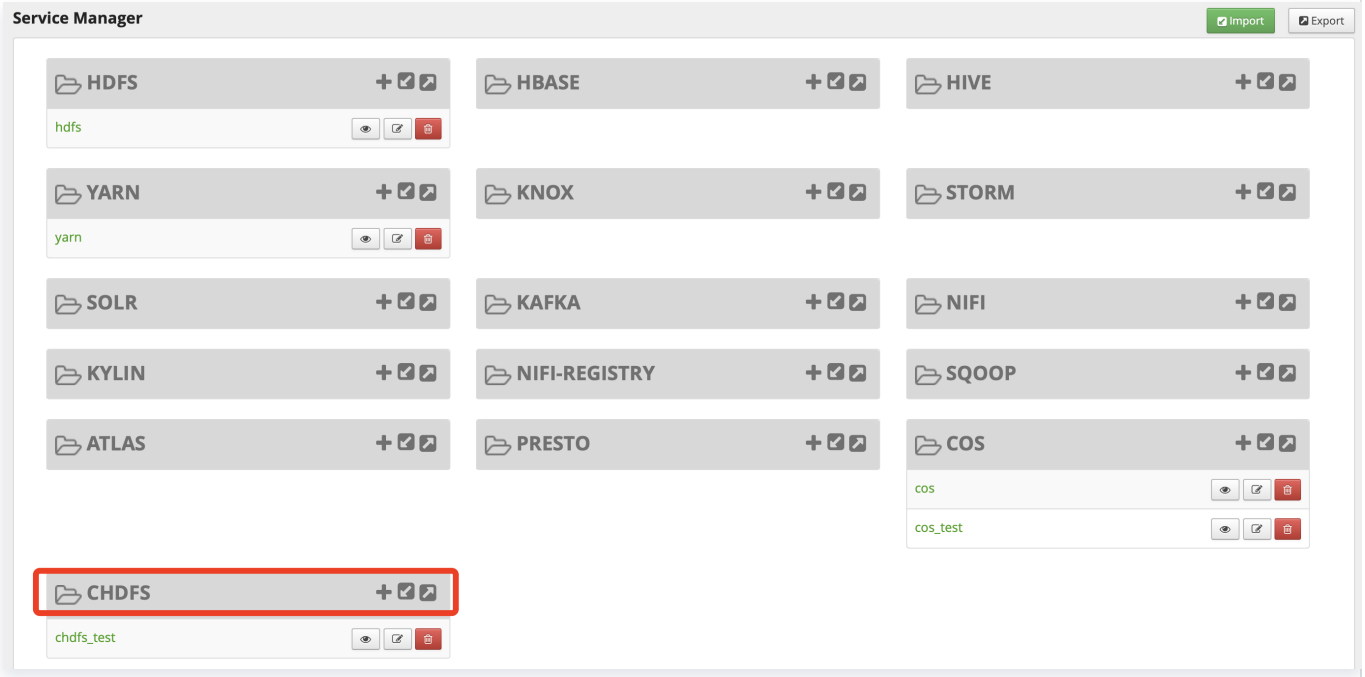
1. 在 Ranger 的服务定义目录下新建 COS 目录（注意：目录权限需要保证至少有 x 与 r 权限）。
2. 腾讯云的 EMR 环境，路径是 ranger/ews/webapp/WEB-INF/classes/ranger-plugins。
3. 自建的 hadoop 环境，可以通过在 ranger 目录下 find hdfs 等方式找到已经接入到 ranger 服务的组件，查找 ranger-plugins 目录位置。

```
[hadoop@10 ranger-plugins]$ ll
total 68
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 atlas
drwxr-xr-x 2 hadoop hadoop 4096 Dec 15 10:57 chdfs
drwxr-xr-x 2 hadoop hadoop 4096 Dec 15 10:57 cos
drwxr-xr-x 2 hadoop hadoop 4096 Feb 25  2020 hbase
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 hdfs
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 hive
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 kafka
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 kms
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 knox
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 kylin
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 nifi
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 nifi-registry
drwxr-xr-x 2 hadoop hadoop 4096 Aug  5  2020 presto
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 solr
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 sqoop
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 storm
drwxr-xr-x 2 hadoop hadoop 4096 Feb 19  2020 yarn
```

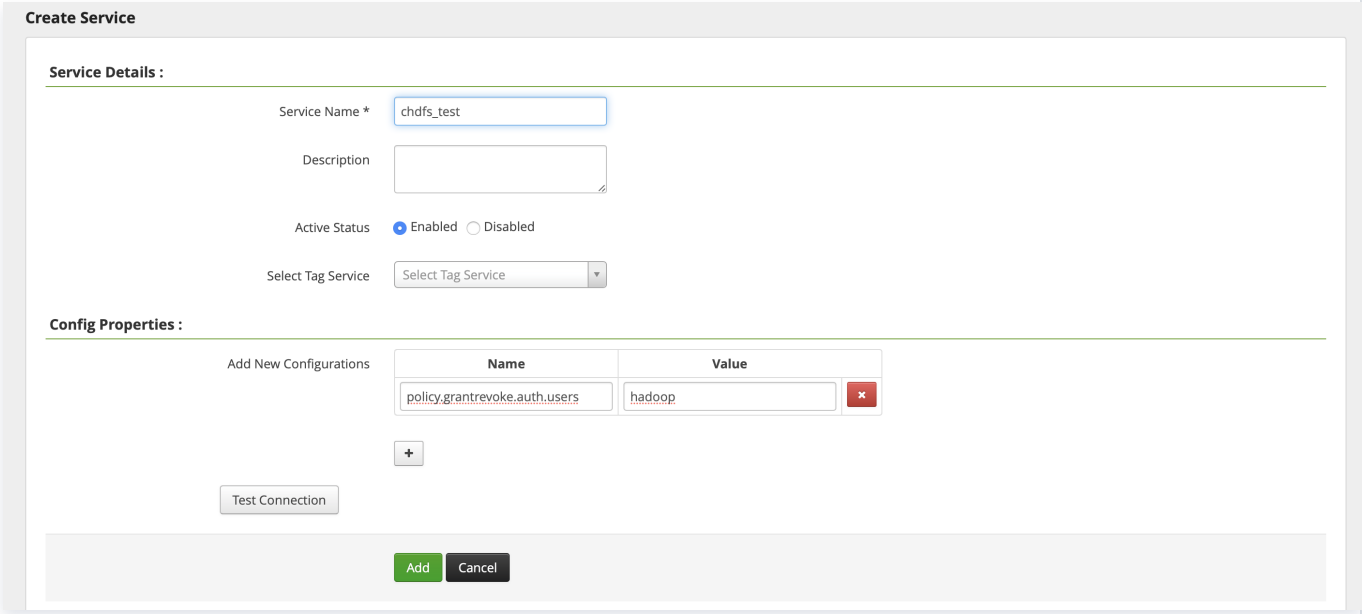
4. 在 CHDFS 目录下，放入 cos-chdfs-ranger-plugin-xxx.jar（注意：jar 包至少有 r 权限）。若使用 CHDFS 服务，需要在机器上放入 chdfs-ranger.json；若使用 COSN 服务，需要放入 cos-ranger.json；若同时使用 CHDFS 和 COSN 服务，需要同时放入 chdfs-ranger.json 和 cos-ranger.json 文件，可前往 Github 获取。
5. 重启 Ranger 服务。
6. 在 Ranger 上注册 CHDFS Service。可参考如下命令：

```
##生成服务，需传入 Ranger 管理员账号密码，以及 Ranger 服务的地址。
##对于腾讯云 EMR 集群，管理员用户是 root，密码是构建 emr 集群时设置的 root 密码，
ranger 服务的 IP 换成 EMR 的 master 节点 IP。
adminUser=root
##构建 EMR 集群时设置的密码，也是 ranger 服务 web 页面的登录密码。
adminPasswd=xxxxxx
##如果 ranger 服务有多个 master 节点，任选一个 master 即可
rangerServerAddr=10.0.0.1:6080
##命令行中 -d 指定步骤 2 中的 json 文件
curl -v -u${adminUser}:${adminPasswd} -X POST -H
"Accept:application/json" -H "Content-Type:application/json" -d @./chdfs-
ranger.json http://${rangerServerAddr}/service/plugins/definitions
##如果要删除刚定义的服务，则传入刚刚创建服务时，返回的服务 ID
serviceId=102
curl -v -u${adminUser}:${adminPasswd} -X DELETE -H
"Accept:application/json" -H "Content-Type:application/json"
http://${rangerServerAddr}/service/plugins/definitions/${serviceId}
```

7. 创建服务成功后，可在 Ranger 控制台看到 CHDFS 服务（以 CHDFS 为例，COSN 类似）。如下所示：



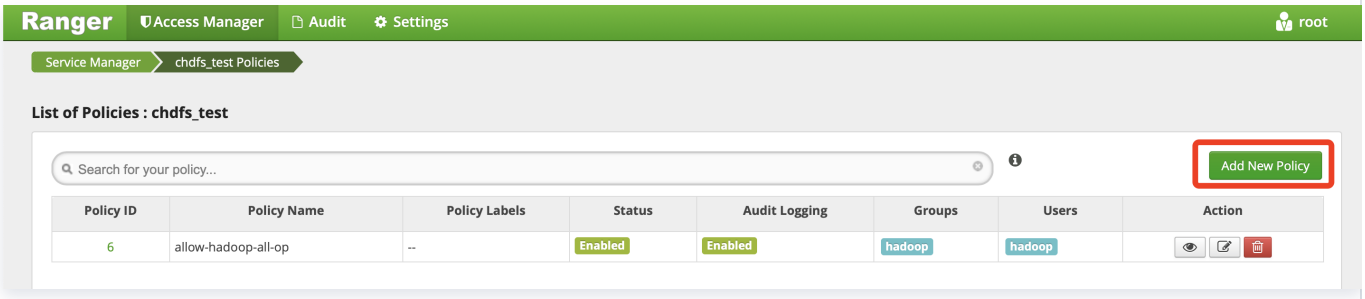
8. 在 CHDFS 服务侧单击【+】，定义新服务实例，服务实例名可自定义，例如 `chdfs` 或者 `chdfs_test` 。服务的配置如下所示：



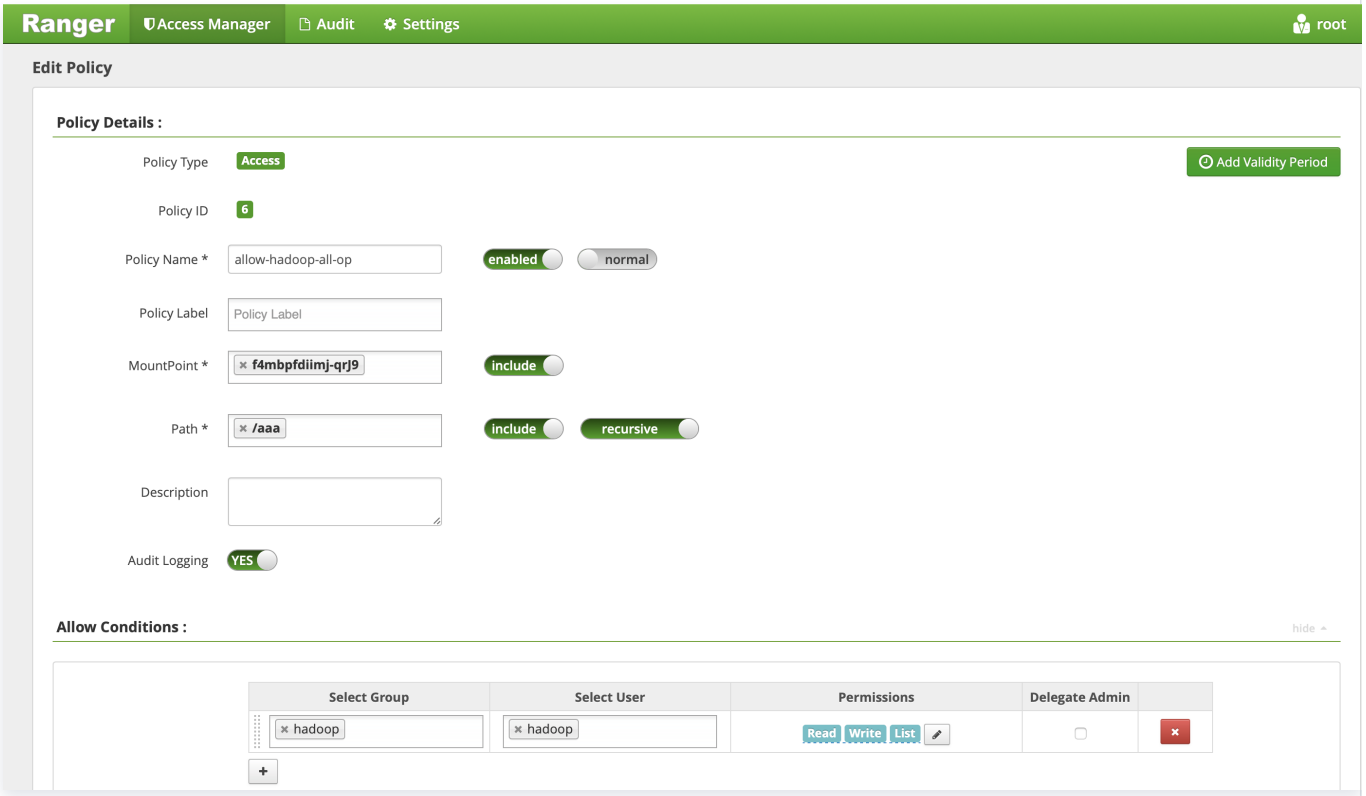
其中 policy.grantrevoke.auth.users 需设置后续启动 COSRangerService 服务的用户名。通常建议设置成 hadoop，后续 COSRangerService 可使用此用户名进行启动。

9. 单击新生成的 CHDFS 服务实例。

添加 policy，如下所示：



10.在跳转界面中，配置以下参数。具体说明如下：



- **MountPoint**：挂载点的名称，格式为 f4mxxxxxx-yyyy 等样式，可登录 CHDFS 控制台 查看。
- **path**: CHDFS 路径。注意 CHDFS 路径必须以 / 开始。
  - include：表示设置的权限适用于 path 本身，还是除了 path 以外的其他路径。
  - recursive：表示权限不仅适用于 path，还适用于 path 路径下的子成员（即递归子成员）。通常用于 path 设置为目录的情况。
- **user/group**：用户名和用户组。这里是或的关系，即用户名或者用户组满足其中一个，即可拥有对应的操作权限。
- **Permissions**：
  - Read：读操作。对应于对象存储里面的 GET、HEAD 类操作，包括下载对象、查询对象元数据等。
  - Write：写操作。对应于对象存储里面的 PUT 类等修改操作，例如上传对象。
  - Delete：删除操作。 对应于对象存储里删除 Object。对于 Hadoop 的 Rename 操作，需要有对原路径的删除操作权限，对新路径的写入操作权限。
  - List：遍历权限。对应于对象存储里面的 List Object。

部署 COS-Ranger-Service

COS-Ranger-Service 是整个权限体系的核心，负责集成 ranger 的客户端，接收 ranger client 的鉴权请求，token 生成续租请求和临时密钥生成请求。同时也是敏感信息（腾讯云密钥信息）所在的区域，通常部署在堡垒机器上，只允许集群管理员操作，查看配置等。
COS-Ranger-Service 支持多节点的HA 部署，DelegationToken 状态持久化到 DB上。通过 ZK 互相感知彼此节点的存在。通过客户端配置的任何一个的cos-ranger-server 的地址， 即可知道全量的列表。 因此可以通过平行扩容 cos-ranger-server，来提升整体的鉴权能力。

代码地址

可前往 Github 的 cos-ranger-server 目录下获取。

版本

V6.0版本及以上。

部署步骤

1. 将 COS Ranger Service 服务代码拷贝到集群的几台机器上，生产环境建议至少两台机器。因为涉及到敏感信息，建议是堡垒机或者权限严格管控的机器。
2. 如果是 kerberos 集群，则需要一个 db 来保存 Delegation Token（有关 kerberos 的 Delegation token 的作用, 搜索相关博文即可）， db 规格推荐16c32g，100g磁盘以上。在负载不高的集群上，可混用 Hive meta store 的 db。初始化 database 和表的 sql 语句可前往 Github 的 cos-ranger-server/sql 目录下获取.
3. 修改 cos-ranger.xml 文件中的相关配置，其中必须修改的配置项如下所示。配置项说明请参见文件中的注释说明（配置文件可前往 Github 的 cos-ranger-server/conf 目录下获取）。
   - qcloud.object.storage.rpc.address
   - qcloud.object.storage.status.port
   - qcloud.object.storage.enable.chdfs.ranger
   - qcloud.object.storage.zk.address（zk 地址，cos ranger service 启动后注册到 zk 上）
   - qcloud.object.storage.kerberos.principal (kerberos 集群下的 principal, 非 kerberos 请忽略)
   - qcloud.object.storage.kerberos.keytab (kerberos 集群下的 keytab 文件, 非 kerberos 请忽略)
   - sql-dt-secret-manager.connection.url (kerberos 集群, 保存 delegation token 的 db, 非 kerberos 请忽略)
   - sql-dt-secret-manager.connection.username (kerberos 集群, 访问 delegation token 的 db 的用户名, 非 kerberos 请忽略)
   - hadoop.security.credential.provider.path（kerberos 集群, 访问 delegation token 的 db 的用户密码的 jceks 文件路径, 非 kerberos 请忽略)
4. 修改 ranger-chdfs-security.xml 文件中的相关配置。其中必须修改的配置项有如下所示。配置项说明请参见文件中的注释说明（配置文件可前往 Github 的 cos-ranger-server/conf 目录下获取）。
   - ranger.plugin.chdfs.policy.cache.dir
   - ranger.plugin.chdfs.policy.rest.url
   - ranger.plugin.chdfs.service.name
5. 修改 start_rpc_server.sh 中 hadoop_conf_path 和 java.library.path 的配置。这两个配置分别指向 hadoop 配置文件所在的目录（例如 core-site.xml、hdfs-site.xml）以及 hadoop native lib 路径。
6. 执行如下命令启动服务。

```
chmod +x start_rpc_server.sh
nohup ./start_rpc_server.sh &> nohup.txt &
```

7. 如果启动失败，查看 log 下 error 日志是否有错误信息。

8. cos-ranger-service 支持展示 HTTP 端口状态（端口名为 qcloud.object.storage.status.port，默认值为 9998）。用户可通过以下命令获取状态信息（例如目前全量的 cos-ranger-server 列表、鉴权数量统计等）。

```
# 请将下面的10.xx.xx.xxx 替换为部署 ranger service 的机器 IP
# port 9998 设置为 qcloud.object.storage.status.port 配置值
curl -v http://10.xx.xx.xxx:9998/status
```

部署 COS-Ranger-Client 和 COSN-Ranger-Interface

COS-Ranger-Client 由 hadoop chdfs 插件动态加载，并代理访问 COS-Ranger-Service 的相关请求。例如获取 token、鉴权操作等。

代码地址

可前往 Github 的 cos-ranger-client 和 cosn-ranger-interface 目录下获取。

版本

cos-ranger-client 要求 V6.0版本及以上。cosn-ranger-interface 要求 v1.0.5版本及以上。

腾讯云 EMR 环境中默认安装目录在 common/lib 下，例如 /usr/local/service/hadoop/share/hadoop/common/lib 下。请根据自己的环境，放在对应的common/lib 路径下。对于 ranger-client 的包名，例如 hadoop-ranger-client-for-hadoop-2.8.5-6.0.jar，2.8.5 是 hadoop 版本号，6.0是该包的版本号。for-hadoop 是通常组件使用的版本，其他一些组件，例如 presto，impala 以及高版本的 spark(spark-3.2.0版本及以后) 等，由于对依赖的 hadoop-common做了 shade，因此 ranger-client 也必须做 shade，否则会报类找不到。这些包请下载对应的 for-presto，for-impala, for-spark 版本等。

部署方式

1. 将 cos-ranger-client jar 包和 cosn-ranger-interface jar 包拷贝到与 CHDFS 插件同一目录下（通常在 /usr/local/service/hadoop/share/hadoop/common/lib/ 目录下；请选择拷贝与自身 hadoop 大版本一致的 jar 包，最后确保 jar 包有可读权限）。
2. 在 core-site.xml 添加如下配置项：

```
```
<configuration>
        <!--******必须配置********-->
        <!-- 上一步部署的 cos ranger server 的地址 -->
        <property>
            <name>qcloud.object.storage.ranger.service.address</name>
            <value>10.0.0.8:9999,10.0.0.10:9999</value>
        </property>
```
```

```
            <!--***可选配置****-->
            <!-- 设置 cos ranger service 端用的 kerberos 凭据，参考 cos
ranger service 端的配置，须保持一致，如果没有认证的需求，不需要配置 -->
            <property>

<name>qcloud.object.storage.kerberos.principal</name>
                        <value>hadoop/_HOST@EMR-XXXX</value>
            </property>
</configuration>
```

部署 CHDFS

版本

V3.3版本及以上。

部署方式

部署 CHDFS 插件的方法请参考 挂载 CHDFS 文档，开启 ranger 通过添加以下方式进行：

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

验证

1. 使用 hadoop cmd 执行访问 chdfs 的相关操作。如下所示：

```
# 将挂载点，路径等替换为自己的实际信息。
hadoop fs -ls ofs://f4mxxxxyyyy-zzzz.chdfs.ap-guangzhou.myqcloud.com/doc
hadoop fs -put ./xxx.txt ofs://f4mxxxxyyyy-zzzz.chdfs.ap-
guangzhou.myqcloud.com/doc/
hadoop fs -get ofs://f4mxxxxyyyy-zzzz.chdfs.ap-
guangzhou.myqcloud.com/exampleobject.txt
hadoop fs -rm ofs://f4mxxxxyyyy-zzzz.chdfs.ap-
guangzhou.myqcloud.com/exampleobject.txt
```

2. 使用 MR Job 进行验证，验证前需重启相关的服务，例如 Yarn、Hive 等。

获取统计信息

可以通过以下 curl 命令手动获取 COS Ranger 的统计信息：

```
curl http://${ranger_ip}:9998/status | jq

# 返回
{
  "allMemberAddress": "10.0.0.7:9999",//集群中所有成员的地址
  "currentNodeIsLeader": true, // 当前节点是否是leader节点
  "leaderAddress": "10.0.0.7:9999", // leader地址(对于客户端v5.x版本有意义，v6.x后是全对等模式，leader信息没有意义
  "authStat": {// 自定义鉴权认证的次数统计信息(对于为实现自定义认证的服务，始终是认证成功)
    "authSuccessStat": { // 认证成功统计信息
      "qps_5m": 0,// 最近 5 分钟的每秒查询数（QPS）
      "total_1m": 0,//最近 1 分钟的总成功认证次数
      "qps_1m": 0,// 最近 1 分钟的每秒查询数
      "total_5m": 0,//最近 5 分钟的总成功认证次数
      "qps": 0//当前的每秒查询数
    },
    "authFailedStat": {// 鉴权认证失败统计信息,字段含义与 authSuccessStat 相同
      "qps_5m": 0,
      "total_1m": 0,
      "qps_1m": 0,
      "total_5m": 0,
      "qps": 0
    }
  },
  "rpcMethodStat": {//包含不同 RPC 方法的调用次数的统计信息
    "checkPermission": {//检查权限的方法统计信息
      "qps_5m": 10,
      "total_1m": 3000,
      "qps_1m": 50,
      "total_5m": 3000,
      "qps": 1000
    },
    "getAvailableService": {//获取可用服务的方法统计信息
      "qps_5m": 0,
      "total_1m": 1,
      "qps_1m": 0.02,
      "total_5m": 1,
      "qps": 0
    }
  },
```

```
"checkPermissionAllowAfterRetryCnt": 0,// 经过重试后允许的权限检查次数
"checkPermissionAllowCnt": 4000,//允许的权限检查总次数
"becomeLeaderTime": "2024-12-10T12:56:52.888Z",//当前节点成为leader的时间
"checkAuthDenyCnt": 0,//被拒绝的认证检查次数
"serviceStartTime": "2024-12-10T12:56:52.884Z",//服务启动的时间
"checkPermissionDenyCnt": 0,//被拒绝的权限检查次数
"accessStat": {//包含不同访问类型的统计信息
  "READ": {//读取操作的统计信息
    "qps_5m": 10,
    "total_1m": 3000,
    "qps_1m": 50,
    "total_5m": 3000,
    "qps": 1000
  }
},
"checkCostStat": {//checkPermission(检查权限)耗时的统计信息
  "checkFailStat": {//失败
    "avg_5m": 0,
    "min_1m": 0,
    "avg": 0,
    "min": 0,
    "max": 0,
    "max_1m": 0,
    "avg_1m": 0,
    "max_5m": 0,
    "min_5m": 0
  },
  "checkSuccessStat": {//成功
    "avg_5m": 5,//最近 5 分钟的平均耗时
    "min_1m": 0,//最近1分钟最小耗时
    "avg": 4,//平均耗时
    "min": 0,//最小耗时
    "max": 1263,//最大耗时
    "max_1m": 1263,//最近一分钟最大耗时
    "avg_1m": 5,//最近一分钟平均耗时
    "max_5m": 1263,//最近五分钟最大耗时
    "min_5m": 0//最近五分钟最小耗时
  }
},
"authCostStat": {// 权限认证耗时统计
  "authSuccessStat": {
    "avg_5m": 0,
    "min_1m": 0,
    "avg": 0,
    "min": 0,
    "max": 0,
    "max_1m": 0,
```

```
      "avg_1m": 0,
      "max_5m": 0,
      "min_5m": 0
    },
    "authFailedStat": {
      "avg_5m": 0,
      "min_1m": 0,
      "avg": 0,
      "min": 0,
      "max": 0,
      "max_1m": 0,
      "avg_1m": 0,
      "max_5m": 0,
      "min_5m": 0
    }
  },
  "rpcMethodCostStat": {//RPC接口耗时统计
    "checkPermission": {
      "avg_5m": 5,
      "min_1m": 0,
      "avg": 4,
      "min": 0,
      "max": 1263,
      "max_1m": 1263,
      "avg_1m": 5,
      "max_5m": 1263,
      "min_5m": 0
    },
    "getAvailableService": {
      "avg_5m": 280,
      "min_1m": 280,
      "avg": 280,
      "min": 280,
      "max": 280,
      "max_1m": 280,
      "avg_1m": 280,
      "max_5m": 280,
      "min_5m": 280
    }
  },
  "statsTimestamp": "2024-12-12T05:28:38.688Z",//统计时间
  "checkStat": {// check policy统计信息
    "checkFailStat": {
      "qps_5m": 0,
      "total_1m": 0,
      "qps_1m": 0,
      "total_5m": 0,
```

```
      "qps": 0
    },
    "checkSuccessStat": {
      "qps_5m": 10,
      "total_1m": 3000,
      "qps_1m": 50,
      "total_5m": 3000,
      "qps": 1000
    }
  }
}
```

接口的出参定义如下：

一级字段	二级字段	三级字段	字段含义
serviceStartTime	-	-	服务启动时间
currentNodeIsLeader	-	-	当前节点是否是 leader
becomeLeaderTime	-	-	成为 leader 时间
statsTimestamp	-	-	统计时间
leaderAddress	-	-	leader 地址
allMemberAddress	-	-	集群内所有成员地址
checkPermissionAllowCnt	-	-	check permission 成功数
checkAuthDenyCnt	-	-	认证鉴权失败数
checkPermissionDenyCnt	-	-	check permission 失败数
checkPermissionAllowAfterRetryCnt	-	-	重试后check permission 成功数
accessStat	-	-	操作统计
accessStat	LIST	-	List 操作次数
accessStat	WRITE	-	Write 操作次数
accessStat	READ	-	Read 操作次数
accessStat	DELETE	-	Delete 操作次数
authStat	-	-	认证统计

authStat	authSuccessStat	qps、qps_1m、qps_5m、count_1m、count_5m	认证成功统计
authStat	authFailedStat	qps、qps_1m、qps_5m、count_1m、count_5m	认证失败统计
authCostStat	-	-	认证耗时统计
authCostStat	authSuccessStat	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	认证成功耗时统计
authCostStat	authFailedStat	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	认证失败耗时统计
checkStat	-	-	check permission 统计
checkStat	checkSuccessStat	qps、qps_1m、qps_5m、count_1m、count_5m	check permission 成功统计
checkStat	checkFailStat	qps、qps_1m、qps_5m、count_1m、count_5m	check permission 失败统计
checkCostStat	-	-	check permission 耗时统计
checkCostStat	checkSuccessStat	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	check permission 成功耗时统计
checkCostStat	checkFailStat	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	check permission 失败耗时统计
rpcMethodStat	-	-	rpc 方法统计

rpcMethodStat	getDelegationToken	qps、qps_1m、qps_5m、count_1m、count_5m	getDelegationToken 方法统计
rpcMethodStat	renewDelegationToken	qps、qps_1m、qps_5m、count_1m、count_5m	renewDelegationToken 方法统计
rpcMethodStat	cancelDelegationToken	qps、qps_1m、qps_5m、count_1m、count_5m	cancelDelegationToken 方法统计
rpcMethodStat	verifyDelegationToken	qps、qps_1m、qps_5m、count_1m、count_5m	verifyDelegationToken 方法统计
rpcMethodStat	checkPermission	qps、qps_1m、qps_5m、count_1m、count_5m	checkPermission 方法统计
rpcMethodStat	getSTS	qps、qps_1m、qps_5m、count_1m、count_5m	getSTS 方法统计
rpcMethodStat	getRangerAuthPolicy	qps、qps_1m、qps_5m、count_1m、count_5m	getRangerAuthPolicy 方法统计
rpcMethodStat	getAvailableService	qps、qps_1m、qps_5m、count_1m、count_5m	getAvailableService 方法统计
rpcMethodCostStat			RPC 方法耗时统计
rpcMethodCostStat	getDelegationToken	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	getDelegationToken 方法统计
rpcMethodCostStat	renewDelegationToken	max、min、avg、max_1m、min_1m、	renewDelegationToken 方法统计

		avg_1m、max_5m、min_5m、avg_5m	
rpcMethodCostStat	cancelDelegationToken	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	cancelDelegationToken 方法统计
rpcMethodCostStat	verifyDelegationToken	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	verifyDelegationToken 方法统计
rpcMethodCostStat	checkPermission	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	checkPermission 方法统计
rpcMethodCostStat	getSTS	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	getSTS 方法统计
rpcMethodCostStat	getRangerAuthPolicy	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	getRangerAuthPolicy 方法统计
rpcMethodCostStat	getAvailableService	max、min、avg、max_1m、min_1m、avg_1m、max_5m、min_5m、avg_5m	getAvailableService 方法统计

常见问题

如果开启了 Ranger，但未配置任何 Policy，或者未匹配到任何 Policy，会如何操作?

如果未匹配上任何 policy，会默认拒绝该操作。

开启了 ranger 后，CHDFS 端是否还会进行 POSIX 鉴权?

Ranger 鉴权是在客户端环境进行的，经过 ranger 鉴权的请求，会发给 CHDFS 服务端，服务端默认会进行 POSIX 鉴权。因此如果权限都在 Ranger 端进行控制，请在 CHDFS 控制台关闭 POSIX 权限。

在 ranger 页面更改了 Policy 未生效怎么办?

请修改 cos-ranger-server 服务目录 conf 下的 ranger-chdfs-security.xml 文件的配置项：ranger.plugin.chdfs.policy.pollIntervalMs，调小该配置项（单位为毫秒），然后重启 cos-ranger-service 服务。Policy 相关测试结束后，建议修改回原来值（时间间隔太小导致轮训频率高，从而导致 CPU 利用率高企）。

其他认证和鉴权的问题可 点击此处 查看。

Tensorflow 读写 CHDFS 数据

最近更新时间：2024-10-12 20:45:01

CHDFS 准备工作

1. 在腾讯云官网创建 CHDFS 文件系统和 CHDFS 挂载点，配置好权限信息。
2. 通过腾讯云 VPC 环境的 CVM 机器访问创建好的 CHDFS，详情请参见 创建 CHDFS 。
3. 当挂载成功后，打开 hadoop 命令行工具，并执行以下命令，验证 CHDFS 功能是否正常。

```
hadoop fs -ls ofs://f4xxxxxxxxxxxxxx.chdfs.ap-beijing.myqcloud.com/
```

如果能看到以下类似的输出，则表示云 HDFS 功能一切正常。

```
[hadoop@10 ~]$ hadoop fs -ls ofs://[redacted].chdfs.ap-beijing.myqcloud.com/
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/service/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/service/tez/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 31 items
drwxr-xr-x   - root   root          0 2019-12-30 17:03 ofs://[redacted]chdfs.ap-beijing.myqcloud.com/0x
drwxr-xr-x   - hadoop hadoop        0 2019-12-30 18:20 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/data
-rw-r--r--   1 root   root  1048576000 2019-12-13 23:20 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/dd_1G
drwxrwxrwx   - hadoop hadoop        0 2019-12-18 12:08 ofs://[redacted]chdfs.ap-beijing.myqcloud.com/emr
drwxrwxr-x   - root   root          0 2019-12-05 17:13 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-dir-0
drwxrwxr-x   - root   root          0 2019-12-05 17:13 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-dir-1
drwxrwxr-x   - root   root          0 2019-12-05 17:13 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-dir-2
drwxrwxr-x   - root   root          0 2019-12-05 17:24 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-dir-3
drwxr-xr-x   - root   root          0 2019-12-05 17:25 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-dir-4
-rwxrwxr-x   1 root   root          0 2019-12-05 17:13 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-file-0
-rw-rw-r--   1 root   root          0 2019-12-05 17:13 ofs://[redacted].chdfs.ap-beijing.myqcloud.com/fuse-file-1
```

Tensorflow 准备工作

1. 通过 官方 Github 下载 Tensorflow。
2. 参考 腾讯云支持 CHDFS patch，修改 tensorflow 源码，本文示例采用 tensorflow 2.5版本编译。
3. 参考 tensorflow 编译教程，编译修改源码后的 tensorflow。
4. 待编译完成后，安装 tensorflow 模块，并且验证。

> **说明：**
> tensorflow 代码版本差异较大，如果您使用的 tensorflow 非2.5版本，遇到代码问题，可以寻求 CHDFS 团队协助。

Tesorflow 读写 CHDFS 验证

1. 在 CHDFS 上创建测试文件：

```
hadoop fs -copyFromLocal ./testfile ofs://f4xxxxxxxxxxxxxx.chdfs.ap-beijing.myqcloud.com/testfile

hadoop fs -cat ofs://f4xxxxxxxxxxxxxx.chdfs.ap-beijing.myqcloud.com/testfile
hello, world
```

2. 使用 TensorFlow 的 API 查看 CHDFS 上的数据。

```
➜  ~ python3
Python 3.9.6 (default, Jun 29 2021, 05:25:02)
[Clang 12.0.5 (clang-1205.0.22.9)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> with tf.gfile.Open('ofs://f4xxxxxxxxxxxxxx.chdfs.ap-
beijing.myqcloud.com/testfile') as rf:
...     rf.read()
'hello, world\n'
>>>
```

说明：

`ofs://f4xxxxxxxxxxxxxx.chdfs.ap-beijing.myqcloud.com` **为测试的挂载点信息，需要您替换成真实的挂载点信息。**

跨账户访问 CHDFS

最近更新时间：2024-12-17 20:15:52

简介

本文介绍如何实现跨账户访问云 HDFS（Cloud HDFS，CHDFS）。
在 CHDFS 的使用情景中，可能需要使用不同的主账户访问同一个 CHDFS，例如账户 B 的 EMR 集群访问账户 A 的 CHDFS。但在常用的 云 HDFS 控制台 中不同主账户之间的权限组不能绑定对方的挂载点，无法实现跨账户访问。这时可以使用 云 HDFS API：

- CHDFS 提供功能全面的云 HDFS API，只需少量代码您就可以操作 CHDFS，并且云 API 允许将账户 B 的权限组绑定账户 A 的挂载点以此实现 CHDFS 的跨账户访问。
- 在绑定的过程中，挂载点是主要资源，**会鉴权调用方是否拥有操作挂载点 ID 的权限，而权限组 ID 不做鉴权。**

前提条件

- 账户 A 已创建好文件系统，详情请参见 创建 CHDFS 。
- 账户 A 已创建好 挂载点 。

- 账户 B 已创建好 权限组 和 权限规则 。

- 账户 B 的 EMR 集群配置好 CHDFS 环境 。

操作步骤

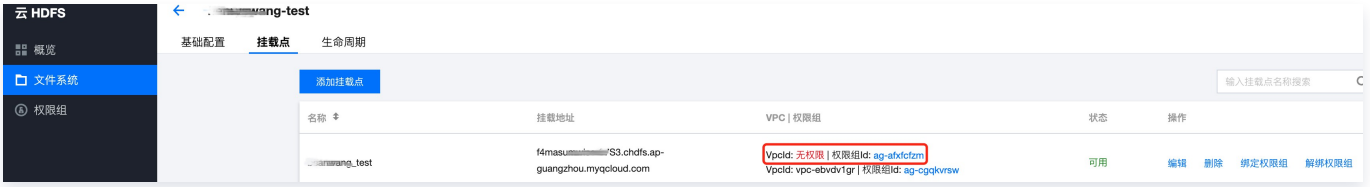
CHDFS 提供功能全面的 云HDFS API，通过云 HDFS API 可以实现 绑定权限组列表 。本文以 Golang 代码为例，实现将账户 B 的权限组绑定账户 A 的挂载点。实行绑定动作的主体为账户 A 或拥有权限的 A 的子账户。

1. 在以下代码配置账户 A 的密钥、账户 A 的挂载点 ID、账户 B 的权限组 ID。

```
package main

import (
    "fmt"

    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common"
    "github.com/tencentcloud/tencentcloud-sdk-go/tencentcloud/common/errors"
```

```
	"github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/common/profile"
	chdfs "github.com/tencentcloud/tencentcloud-sdk-
go/tencentcloud/chdfs/v20201112"
)

func main() {
	// 实例化一个认证对象，入参需要传入腾讯云账户 secretId，secretKey,此处还需注意密钥
对的保密
	// 密钥可前往 https://console.cloud.tencent.com/cam/capi网站进行获取
	credential := common.NewCredential(
		"SecretId",
		"SecretKey",
	)
	// 实例化一个 client 选项，可选的，没有特殊需求可以跳过
	cpf := profile.NewClientProfile()
	cpf.HttpProfile.Endpoint = "chdfs.tencentcloudapi.com"
	// 实例化要请求产品的 client 对象,clientProfile 是可选的
	client, _ := chdfs.NewClient(credential, "", cpf)

	// 实例化一个请求对象,每个接口都会对应一个 request 对象
	request := chdfs.NewAssociateAccessGroupsRequest()

		//账户 A 的挂载点 ID
	request.MountPointId = common.StringPtr("f4maxxxx-xxS3")
		//账户B的权限组 ID
	request.AccessGroupIds = common.StringPtrs([]string{ "ag-xxxxxzm" })

	// 返回的 resp 是一个 AssociateAccessGroupsResponse 的实例，与请求对象对应
	response, err := client.AssociateAccessGroups(request)
	if _, ok := err.(*errors.TencentCloudSDKError); ok {
		fmt.Printf("An API error has returned: %s", err)
		return
	}
	if err != nil {
		panic(err)
	}
	// 输出 json 格式的字符串回包
	fmt.Printf("%s", response.ToJsonString())
}
```

2. 运行代码后，再次查看账户 A 的挂载点，即可看到账户 B 的权限组成功绑定在本挂载点上。



3. 在账户 B 的 EMR 集群机器中通过 `hadoop fs -ls` 命令查看到文件列表，即表示挂载成功。账户 B 可以访问账户 A 的 CHDFS，成功实现跨账户访问 CHDFS。

```
hadoop fs -ls ofs://f4masu[redacted]S3.chdfs.ap-guangzhou.myqcloud.com/
Found 1 items
drwxr-xr-x   - hadoop supergroup          0 2021-0[redacted].48 ofs://f4masum[redacted]S3.chdfs.ap-guangzhou.myqcloud.com[redacted]
```

若未通过云 API 执行上述绑定操作，在账户 B 的 EMR 集群中通过 `hadoop fs -ls` 命令则查看到访问文件列表失败，显示 ***Permission denied: No access rules matched*** 错误，账户 B 无法访问账户 A 的 CHDFS。

4.
```
[redacted]/]# hadoop fs -ls ofs://f4masu[redacted]S3.chdfs.ap-guangzhou.myqcloud.com/
22/[redacted]:26 ERROR fs.CHDFSHadoopFileSystem: Permission denied: No access rules matched
22/[redacted]:26 ERROR fs.CHDFSHadoopFileSystemAdapter: initialize failed! a ioException occur!
java.io.IOException: Permission denied: No access rules matched
```

跨 VPC 访问 CHDFS

最近更新时间：2024-12-17 20:30:53

简介

本文介绍如何在不同 VPC 下访问云 HDFS（Cloud HDFS，CHDFS）的过程。

- **权限组**是 CHDFS 提供的访问控制白名单，用户可以自行创建权限组，以此实现对 CHDFS 的权限管理，权限组需要绑定且只能绑定一个 VPC 网络。
- 创建**挂载点**时候，需要将权限组绑定挂载点。

一个挂载点可以同时被多个权限组绑定，因此我们可以利用不同 VPC 的权限组绑定同一个挂载点来实现跨 VPC 访问 CHDFS。
本文以不同 VPC 下的 EMR 集群访问 CHDFS 为例。

前提条件

需要账户中已有 CHDFS 文件系统和EMR集群，并且 EMR 集群已配置好 CHDFS 环境，可参考以下创建：

- 已创建文件系统，详情请参见 创建 CHDFS 。
- 已创建 挂载点 。
- EMR 集群已配置 CHDFS 环境 。

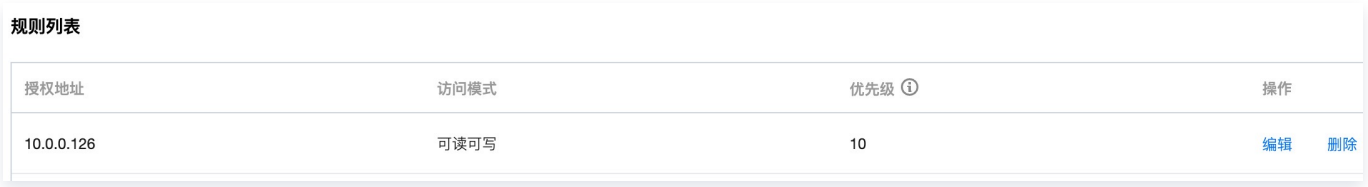
操作步骤

步骤1：创建权限组和权限规则

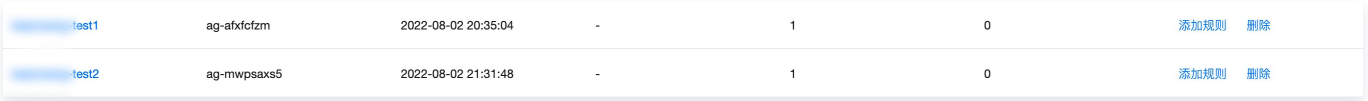
1. 登录 CHDFS 控制台 。
2. 在左侧导航栏中，单击**权限组**，选择地域（例如广州），单击**新建**。

3. 在弹出的窗口中，输入名称，依据 EMR 集群所属 VPC，设置 VPC 网络类型和 VPC 网络名称。

4. 选择创建好的权限组，单击**添加规则**，在授权地址中填入 EMR 集群所在的网段，配置访问模式和优先级。



5. 重复执行步骤2 – 步骤4，配置另一个 VPC 的权限组。



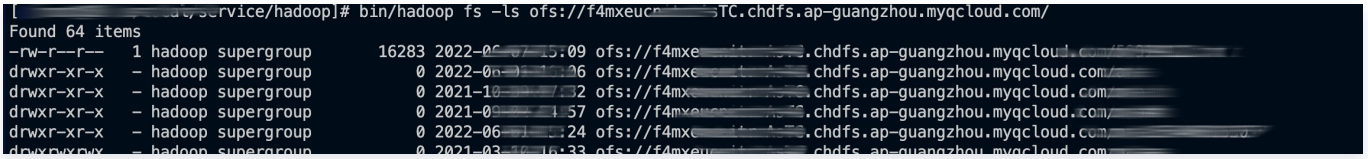
步骤2：挂载点绑定权限组

1. 选择创建好的挂载点，单击**绑定权限组**。

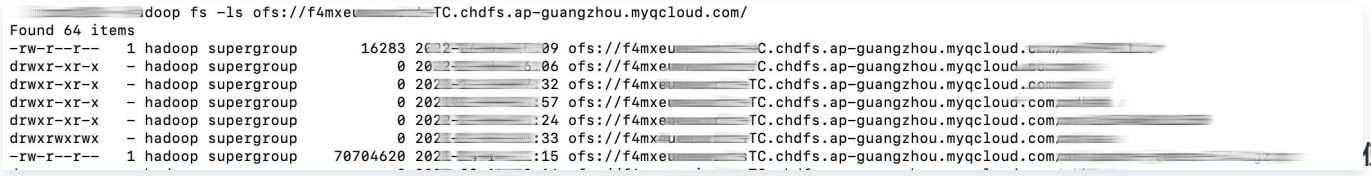
2. 在弹出的窗口中，绑定 步骤1 创建的两个不同 VPC 的权限组。

步骤3：访问 CHDFS

1. 在配置好权限组和 CHDFS 环境 的 EMR 集群机器中，执行 `hadoop fs -ls` 命令，查看到文件列表，即表示挂载成功。



2. 在另一台不同权限组的 EMR 集群机器中，重复执行上一步命令，查看到文件列表，即表示挂载成功。



使用一台权限组未添加权限规则的机器进行相同操作，挂载失败，显示 ***Permission denied: No access rules matched*** 错误。