Byzer-lang文件系统插件介绍

Byzer-lang 支持多种类型，例如本地文件系统，HDFS，Amazon S3等。但在数据处理时，通常需要访问多个文件系统，这时 Byzer-Lang的文件系统插件就很有用了。简单配置后，既可以访问各类文件系统。下面 Azure Blob 和 Amazon S3 的读写为例介绍其使用方式。

开始体验之前，请部署最新 Byzer-lang spark3.1.1-hadoop 3.2 。

Amazon S3

为了读写Amazon S3，请将 hadoop-aws jar 包部署至 $SPARK_HOME/jars 或 Byzer-lang 安装目录的 libs 子目录，并重启 Byzer-lang。

Shell

```
wget https://repo1.maven.org/maven2/com/amazonaws/aws-java-sdk-bundle/1.11.375/aws-java-sdk-bundle-1.11.375.jar
wget https://repo1.maven.org/maven2/org/apache/hadoop/hadoop-aws/3.2.0/hadoop-aws-3.2.0.jar
```

我们可以开始读写 Amazon S3了。

Shell

```
set rawData='''
{"jack":1,"jack2":2}
{"jack":2,"jack2":3}
''';
load jsonStr.`rawData` as table1;

SAVE OVERWRITE table1 as FS.`s3a://zjc-test-ap-northeast-1/test.csv` where
`fs.s3a.endpoint`="s3.ap-northeast-3.amazonaws.com"
and `fs.s3a.access.key`="access_key"
and `fs.s3a.secret.key`="secret_key"
and `fs.s3a.impl`="org.apache.hadoop.fs.s3a.S3AFileSystem"
and `fs.s3a.buffer.dir`="/tmp/oss"
and implClass="parquet";

load FS.`s3a://zjc-test-ap-northeast-1/test.csv` where
`fs.s3a.endpoint`="s3.ap-northeast-3.amazonaws.com"
and `fs.s3a.access.key`="access_key"
and `fs.s3a.secret.key`="secret_key"
and `fs.s3a.impl`="org.apache.hadoop.fs.s3a.S3AFileSystem"
and `fs.s3a.buffer.dir`="/tmp/oss"
and implClass="parquet"
as table1;
```

脚本中包含 S3 路径，AWS区域 (fs.s3a.endpoint)，用户鉴权信息 文件格式 (access key secret key) 。S3 路径格式为

Shell

```
s3a://<bucket_name>/<name>
```

Azure Blob

使用前，请编译部署 hadoop-azure 3.2.0 shade jar 至$SPARK_HOME/jars 或 Byzer-lang 安装目录的 libs 子目录，并重启 Byzer-lang。

Ruby

```
set rawData='''
{"jack":1,"jack2":2}
{"jack":2,"jack2":3}
''';
load jsonStr.`rawData` as table1;

SAVE overwrite table1 as FS.`wasb://container@account.blob.core.chinacloudapi.cn/tmp/json_names_1`
where `fs.azure.account.key.account.blob.core.chinacloudapi.cn`="NXuooh8o07gaIcJxUm2iHkaciQ9TxQg+5MtHSKkIACRZZqZtag99rZoSolsi4RnZoEcXWrEZFULrjChqT7gYAA=="
and `fs.AbstractFileSystem.wasb.impl`="org.apache.hadoop.fs.azure.Wasb"
and `fs.wasb.impl`="org.apache.hadoop.fs.azure.NativeAzureFileSystem"
and implClass="parquet";

load FS.`wasb://ncov2019vjpn-storage@ncovjepurheee2vry.blob.core.chinacloudapi.cn/tmp/json_names`
where `fs.azure.account.key.ncovjepurheee2vry.blob.core.chinacloudapi.cn`="key_secret"
and `fs.AbstractFileSystem.wasb.impl`="org.apache.hadoop.fs.azure.Wasb"
and `fs.wasb.impl`="org.apache.hadoop.fs.azure.NativeAzureFileSystem"
and implClass="parquet"
as table2;
```

执行结果如下：

Result | Job Details | Log Message

jack	jack2
1	2
2	3

上面的脚本提供了 Azure Blob 路径，格式如下，假如你使用 Azure Global，请将 core.chinacloudapi.cn 替换为 core.windows.net 。

Shell

```
wasb://<container_name>@<storage_account_name>.blob.core.chinacloudapi.cn/<path>/<file_name>
```

Where 子句 包括三类信息: Azure 鉴权信息, Azure 文件系统参数和文件格式。Azure 鉴权信息格式为

```Shell
"fs.azure.account.key.<storage_account_name>.blob.core.chinacloudapi.cn`="key_secret"
```

fs.AbstractFileSystem.wasb.impl 和 fs.wasb.impl fs.wasb 声明了 wasb 类型的URI 处理入口类为 NativeAzureFileSystem。NativeAzureFileSystem 兼容 HDFS API，若你感兴趣，以上配置以core-site.xml 部署到 $HADOOP_CONF_DIR，部署 hadoop-azure 3.2.0 shade jar，就能使用 hdfs dfs 命令愉快地操作 Azure Blob的数据了。

最后，implClass 声明了文件格式为 parquet; 也可以是 text orc等。

文件系统插件的实现

熟悉了使用方式，我们了解下实现方式。文件系统插件的实现类为

```CSS
streaming.core.datasource.CustomFS
```

save 和 load 分别实现数据读取和保存，依托 Spark DataSource API 强大的扩展能力，以简单优雅的方式支持各种文件存储。