Alluxio 底层存储系统

Alluxio是一个基于内存的分布式文件系统，其架构在底层分布式文件系统和上层分布式计算框架之间的中间件，其主要职责是以文件的形式在内存或其他存储设施中提供数据的存取服务。Alluxio的系统架构图如下所示：



# 底层文件系统

Alluxio提供UnderFileSystem接口来将各种底层存储与Alluxio整合到一起，被整合的底层存储都作为underfs模块的子模块，可以使用Alluxio同时管理多个底层文件系统，对外提供统一的名称空间达到对上层屏蔽数据存储位置的目的。目前支持的底层文件系统包括：

* GCS，使用Google Cloud Storage作为底层文件系统
* S3，使用Amazon S3作为底层文件系统
* Swift，使用Swift作为底层文件系统
* GlusterFS
* HDFS&Secure HDFS
* OSS，使用Aliyun OSS（对象存储）作为底层文件系统
* NFS
* Azure Blob Storage

Alluxio为了支持多种底层文件系统，引入文件系统Adapter，为了支撑某种底层存储，只需要开发对应的文件系统Adapter即可。底层存储的配置参数为：

*alluxio.underfs.address=${alluxio.work.dir}/underFSStorage*

例如使用HDFS的底层文件系统，配置为：

*alluxio.underfs.address=hdfs://nn/*

Alluxio为底层存储文件系统提供接口UnderFileSystem，只需要实现该接口及提供相应的UnderFileSystemFactor接口，让Alluxio完成路径URI和实现的UnderFileSystem匹配，最后在META-INF/services目录下添加对应的文件，即可整合存储系统。目前支持的底层存储系统如下所示：



下面以HdfsUnderFileSystem为例，介绍Alluxio与底层存储系统的交互过程

1. Alluxio客户端从HDFS中读取数据，并加载到内存

在Alluxio读取数据流程中已经分析数据的过程，其中最后从Worker中通过BlockReadHandler获取数据，核心方法为openBlock，执行过程如下所示：

*private void openBlock(BlockReadRequestContext context, Channel channel) throws Exception {*

*while (retryPolicy.attempt()) {*

*.....*

*//先读取Alluxio Worker内存中的Block，如果没有找到，从UFS中读取*

*if (mWorker.openUfsBlock(request.getSessionId(), request.getId(), openUfsBlockOptions)) {*

*try {*

*BlockReader reader =*

*mWorker.readUfsBlock(request.getSessionId(), request.getId(), request.getStart());*

*AlluxioURI ufsMountPointUri =*

*((UnderFileSystemBlockReader) reader).getUfsMountPointUri();*

*String ufsString = MetricsSystem.escape(ufsMountPointUri);*

*String metricName = String.format("BytesReadUfs-Ufs:%s", ufsString);*

*context.setBlockReader(reader);*

*context.setCounter(MetricsSystem.workerCounter(metricName));*

*return;*

*} .....}*

2）Alluxio将数据持久化到HDFS中

可以通过bin/alluxio fs persist命令，将Alluxio内存文件系统的数据持久化到HDFS中，调用核心方法是PersistCommand#persist

*private void persist(AlluxioURI filePath) throws AlluxioException, IOException {*

*URIStatus status = mFileSystem.getStatus(filePath);*

*if (status.isFolder()) {*

*List<URIStatus> statuses = mFileSystem.listStatus(filePath);*

*List<String> errorMessages = new ArrayList<>();*

*for (URIStatus uriStatus : statuses) {*

*AlluxioURI newPath = new AlluxioURI(uriStatus.getPath());*

*try {*

*persist(newPath);*

*} catch (Exception e) {*

*errorMessages.add(e.getMessage());*

*}*

*}*

*if (errorMessages.size() != 0) {*

*throw new IOException(Joiner.on('\n').join(errorMessages));*

*}*

*} else if (status.isPersisted()) {*

*System.out.println(filePath + " is already persisted");*

*} else { //FileSystemUtils调用mFileSystem的接口写入数据*

*FileSystemUtils.persistFile(mFileSystem, filePath);*

*System.out.println("persisted file " + filePath + " with size " + status.getLength());*

*}*

*}*

具体过程不再介绍，将persist命令通过rpc调用至AlluxioMaster，然后执行。