Hadoop分布式缓存-DistributedCache

DistributedCache是Hadoop提供的文件缓存工具，自动将指定的文件分发到各个节点上，缓存到本地，供用户程序读取使用。DistributedCache是Hadoop MapReduce计算框架的一部分，可以将应用需要的文件，只读的文本文件、jar包和压缩包等缓存后，Task执行时对所有DataNode可用。其具有以下特点：

* 缓存的文件是只读，修改这些文件没有意义
* 用户可以调整的文件可见范围，防止重复复制
* 按需复制，文件是通过HDFS作为共享中心分发到各节点上，且只发给任务到调度到的节点

Hadoop DistributedCache使用的场景：

* 分发字典文件，MapReduce需要用到一些外部字典，比如黑白名单、词表等
* Map-side Join，当多表连接时，当遇到表足以加载到内存中，可以使用DistributedCache将小表分发到各个节点上，以供Map Task加载使用
* 自动化软件部署，将特定版本的库通过DistributedCache分发到各个节点上，程序运行完后，Hadoop自动删除

# DistributedCache的使用

Hadoop提供了两种DistributedCache使用方式，一种是通过API，在程序中设置文件路径，另外一种是通过命令行（-files/-archives/-libjars）参数：

* -files，将指定的本地或者hdfs文件分发到各个task的工作目录下，不对文件进行任何处理
* -archives，将指定文件分发到各个Task的工作目录，并对名称后缀.jar,.zip和tgz文件自动解压。默认情况下，解压后的内容放到工作目录下为解压前文件名的目录中
* -libjars，指定待分发的jar包，Hadoop将这些jar包分发到各个节点，会将其自动添加到任务的CLASSPATH环境变量中。

在Hadoop中使用DistributedCache的步骤如下：

1. 在conf中正确配置被分发文件的路径（HDFS Path，要先将文件上传到HDFS上）

*DistributedCache.addCacheFile(*

*new URI("hdfs://cmhhost1.novalocal:8020/tmp/cache/test.dat"),*

*job.getConfiguration());*

1. 在定义的mapper或者reducer上获取文件下载到本地后的路径，可以通过重写setup方法

*Path path[] = DistributedCache.getLocalCacheFiles(conf);*

1. 在mapper或者reduce类中读取文件内容

*Scanner scan = new Scanner(in);*

*while(scan.hasNext()) {*

*logger.info(Thread.currentThread().getName() +" scan Context: " + scan.next());*

*}*

*scan.close();*

1. 在Map中的日志输出如下

*CacheFile:/*

*hadoop/yarn/local/usercache/hdfs/appcache/application\_1528970044550\_0005/container\_e01\_1528970044550\_0005\_01\_000002/test.dat*

*com.fys.cache.FileMapper: main scan Context: hello*

*com.fys.cache.FileMapper: main scan Context: data*

*com.fys.cache.FileMapper: main scan Context: novalLocal*

在日志中输出了缓存文件中的内容

# .DistributedCache架构及源码分析

1）Job Runner设置缓存

DistributedCache将缓存的文件复制到每个节点上，其执行代码如下所示：

*DistributedCache.addCacheFile(*

*new URI("hdfs://cmhhost1.novalocal:8020/tmp/cache/test.dat"),*

*job.getConfiguration());*

此操作在创建Job时，将conf传递给Job执行之前，否则数据文件路径不会被传到Mapper中。addCacheFile的执行源码如下：

*String files = conf.get(MRJobConfig.CACHE\_FILES);  
conf.set(MRJobConfig.CACHE\_FILES, files == null ? uri.toString() : files + ","  
 + uri.toString());*

配置CACHE\_FILES参数：*mapreduce.job.cache.files，*将以Conf的形式其参数传递到Task。

1. 在Mapper取文件URI

*Path[] paths = DistributedCache.getLocalCacheFiles(conf);*

getLocalCacheFile的执行：

*public static Path[] getLocalCacheFiles(Configuration conf)  
 throws IOException {  
 return StringUtils.stringToPath(conf.getStrings(MRJobConfig.CACHE\_LOCALFILES));  
}*

配置参数为：*mapreduce.job.cache.local.files*

1. 将文件复制到本地磁盘，通过将缓存文件添加到Task的LocalResource

*TaskAttemptImpl#createCommonContainerLaunchContext*

*{*

*.....*

*MRApps.setupDistributedCache(conf, localResources); ......*

*}*

执行如下：

*public static void setupDistributedCache(Configuration conf,*

*Map<String, LocalResource> localResources) throws IOException {*

*LocalResourceBuilder lrb = new LocalResourceBuilder();*

*lrb.setConf(conf);*

*// Cache archives*

*lrb.setType(LocalResourceType.ARCHIVE);*

*lrb.setUris(DistributedCache.getCacheArchives(conf)); //获取Archives URI*

*lrb.setTimestamps(DistributedCache.getArchiveTimestamps(conf));*

*lrb.setSizes(getFileSizes(conf, MRJobConfig.CACHE\_ARCHIVES\_SIZES));*

*lrb.setVisibilities(DistributedCache.getArchiveVisibilities(conf));*

*lrb.setSharedCacheUploadPolicies(*

*Job.getArchiveSharedCacheUploadPolicies(conf));*

*lrb.createLocalResources(localResources);*

*// Cache files*

*lrb.setType(LocalResourceType.FILE);*

*lrb.setUris(DistributedCache.getCacheFiles(conf)); //获取Cache Flles*

*lrb.setTimestamps(DistributedCache.getFileTimestamps(conf));*

*lrb.setSizes(getFileSizes(conf, MRJobConfig.CACHE\_FILES\_SIZES));*

*lrb.setVisibilities(DistributedCache.getFileVisibilities(conf));*

*lrb.setSharedCacheUploadPolicies(*

*Job.getFileSharedCacheUploadPolicies(conf));*

*lrb.createLocalResources(localResources);*

*}*

4）打开本地磁盘中的缓存文件，并使用

*public static LocalFileSystem getLocal(Configuration conf)  
 throws IOException {  
 return (LocalFileSystem)get(LocalFileSystem.NAME, conf);  
}*

*FSDataInputStream in = fsOpen.open(path[0]);*