Spark优化：禁止应用程序将依赖的Jar包传到HDFS

每次当你在Yarn上以Cluster模式提交Spark应用程序的时候，通过日志我们总可以看到下面的信息：

```
21 Oct 2014 14:23:22,006 INFO  [main] (org.apache.spark.Logging$class.logInfo:59)  -
Uploading file:/home/spark-1.1.0-bin-2.2.0/lib/spark-assembly-1.1.0-hadoop2.2.0.jar to
hdfs://my/user/iteblog/...../spark-assembly-1.1.0-hadoop2.2.0.jar
21 Oct 2014 14:23:23,465 INFO  [main] (org.apache.spark.Logging$class.logInfo:59)  -
Uploading file:/export1/spark/spark-1.0.1-bin-hadoop2/spark-1.0-SNAPSHOT.jar to
hdfs://my/user/iteblog/.sparkStaging/application_1413861490879_0010/spark-1.0-SNAPSHOT.j
ar
```

这是Spark自己将运行时候需要依赖的Java包上传到HDFS上，而且每次运行Spark Application的时候都会上传，这时候你会发现你的hdfs://mycluster/user/iteblog/.sparkStaging目录下面存在了大量的Jar文件，这最少存在两个不好的地方：

1、每次上传Jar文件，多少也会影响到程序的运行速度；

2、当在HDFS中上传了大量的Jar文件，这会产生大量的小文件，会对HDFS有影响。

所以我们想是否可以在HDFS上面建立一个公共的lib库存放目录，每次运行Spark的时候，只要我们程序依赖的Jar包存在HDFS中的lib库中，那么这时候我们就不上传该Jar。其实是可以的。我们可以通过配置相应的环境变量实现，步骤如下：

```
bin/hadoop fs -mkdir /home/iteblog/spark_lib
bin/hadoop fs -put  spark-assembly-1.1.0-hadoop2.2.0.jar
            /home/iteblog/spark_lib/spark-assembly-1.1.0-hadoop2.2.0.jar
```

然后编辑spark-default.conf文件，添加以下内容：

```
spark.yarn.jar=hdfs://my/home/iteblog/spark_lib/spark-assembly-1.1.0-hadoop2.2.0.jar
```

也就是使得spark.yarn.jar指向我们HDFS上的Spark lib库。
然后你再去提交应用程序

```
./bin/spark-submit --class org.apache.spark.examples.SparkPi ₩
```

```
--master yarn-cluster ₩
--num-executors 3 ₩
--driver-memory 512m ₩
--executor-memory 2g ₩
--executor-cores 1 ₩
lib/spark-examples*.jar ₩
10
```

你可以看到日志里面已经不需要上传spark-assembly-1.1.0-hadoop2.2.0.jar文件了。

但是遗憾的是，如果你的Hadoop集群配置了HA、 Federation，那么Spark 1.1.0及其之前版本，这个选项无效。这是为什么？看下代码就清楚了：

```
/** See if two file systems are the same or not. */
  private def compareFs(srcFs: FileSystem, destFs: FileSystem): Boolean = {
    val srcUri = srcFs.getUri()
    val dstUri = destFs.getUri()
    if (srcUri.getScheme() == null) {
      return false
    }
    if (!srcUri.getScheme().equals(dstUri.getScheme())) {
      return false
    }
    var srcHost = srcUri.getHost()
    var dstHost = dstUri.getHost()
    if ((srcHost != null) && (dstHost != null)) {
      try {
        srcHost = InetAddress.getByName(srcHost).getCanonicalHostName()
        dstHost = InetAddress.getByName(dstHost).getCanonicalHostName()
      } catch {
        case e: UnknownHostException =>
          return false
      }
      if (!srcHost.equals(dstHost)) {
        return false
      }
    } else if (srcHost == null && dstHost != null) {
      return false
    } else if (srcHost != null && dstHost == null) {
      return false
    }
    if (srcUri.getPort() != dstUri.getPort()) {
      false
    } else {
```

```
    true
  }
}
```

仔细看第15、16行代码，代码里面把我们传进来的HDFS路径中的srcHost当作是IP:port格式的。而如果你的Hadoop集群开启了HA、 Federation，这个srcHost是一个命名空间，所以代码运行在这里就会出现异常，从而导致了上面的选项无效。（本来我想提交这个Bug，但是发现在10月4日已经有人也发现了。）高兴的是，这个bug在Spark 1.1.1及其之后的版本已经修复了。

如果你急着用这个特性，你可以将你的compareFs函数修改成下面的实现即可

```
import com.google.common.base.Objects

/**
  * Return whether the two file systems are the same.
  */
 private def compareFs(srcFs: FileSystem, destFs: FileSystem): Boolean = {
   val srcUri = srcFs.getUri()
   val dstUri = destFs.getUri()
   if (srcUri.getScheme() == null || srcUri.getScheme() != dstUri.getScheme()) {
     return false
   }

   var srcHost = srcUri.getHost()
   var dstHost = dstUri.getHost()

   // In HA or when using viewfs, the host part of the URI may not actually
   // be a host, but the name of the HDFS namespace.
   // Those names won't resolve, so avoid even trying if they
   // match.
   if (srcHost != null && dstHost != null && srcHost != dstHost) {
     try {
       srcHost = InetAddress.getByName(srcHost).getCanonicalHostName()
       dstHost = InetAddress.getByName(dstHost).getCanonicalHostName()
     } catch {
       case e: UnknownHostException =>
         return false
     }
   }

   Objects.equal(srcHost, dstHost) && srcUri.getPort() == dstUri.getPort()
 }
```

然后重新编译一下Spark源码（不会编译？参见《用Maven编译Spark 1.1.0》）。

根据Cloudera 官方博客说明，如果你用的是Cloudera Manager，那么Spark assembly JAR会自动地上传到HDFS，比如上面的hdfs://my/home/iteblog/spark_lib/目录。但是我没安装那个，如果你安装了可以试试。