**LPW使用手册**

1. **简介**

为更好的选择Apache Spark框架运行时的缓存数据，本项目提出了LPW(Least partition weight) 模型，LPW是一个面向分布式计算框架Spark的缓存替换系统，可以为用户提供数据缓存替换功能。由于应用负载特征的多样性、内存资源需求的变化性以及内存资源的持续变化性，LPW根据数据块计算代价、数据块引用次数、数据块所占内存空间等因素建立模型，通过权重识别缓存数据的必要性，并与Spark无缝集成，在应用执行时提供更合理的数据进行替换，提升应用执行的效率。

1. **环境准备**

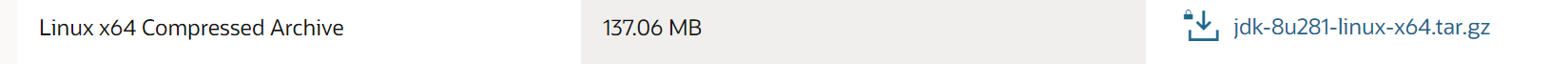
使用LPW需要搭建基础环境，包括Java、Scala、Maven组件的安装。同时，因为LPW是基于Spark扩展的，所以还需要下载spark2.2.3源码。通过安装的Maven对spark源码进行编译。后续脚本的执行和提交与原生spark的使用方式相同。

* 1. **安装JAVA**

下载地址：

<https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>

注：这里需要有oracle账户登录。



配置环境变量。

* 1. **安装Scala**

下载地址：<https://downloads.lightbend.com/scala/2.12.11/scala-2.12.11.tgz>

配置环境变量。

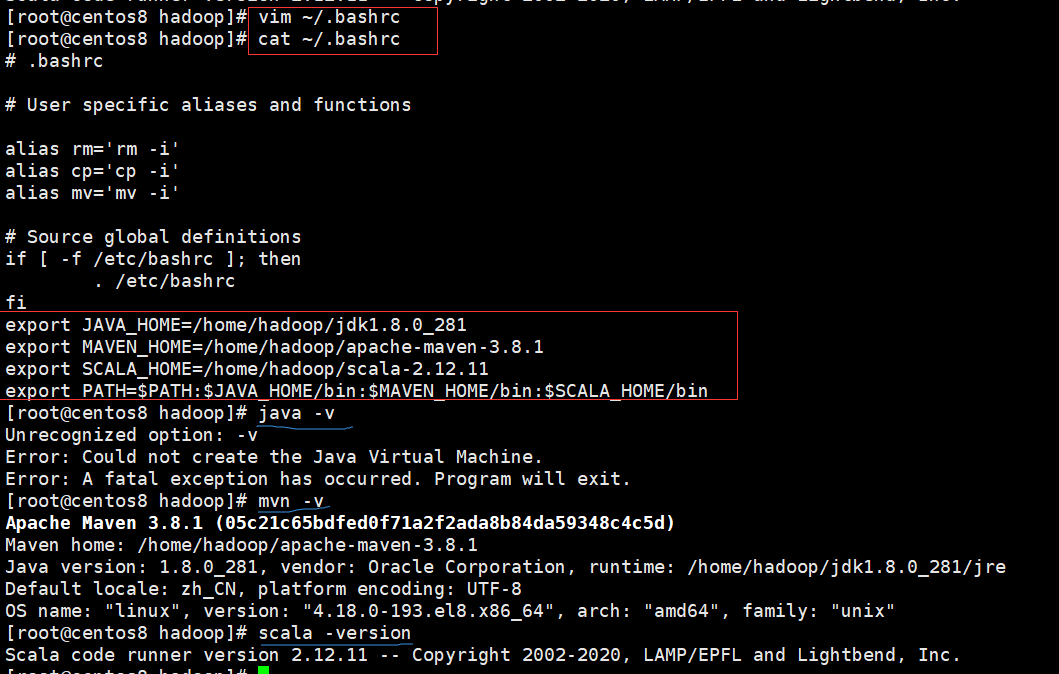
* 1. **安装MAVEN**

Maven下载地址：

<https://archive.apache.org/dist/maven/maven-3/3.6.2/binaries/>



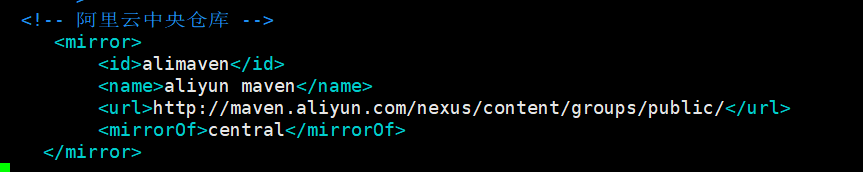
配置环境变量



环境变量配置后一定要保存并更新，**source ~/.bashrc**

* 配置阿里云中央仓库

/home/hadoop/apache-maven-3.8.1/conf/settings.xml 配置文件添加以下内容

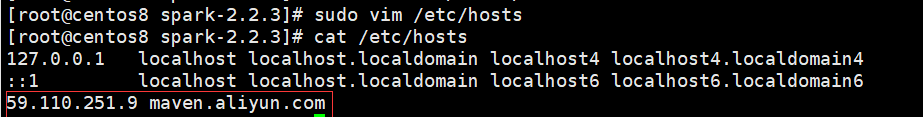


测试阿里云仓库是否能正确访问：



如果不能正确访问，解决办法如下：

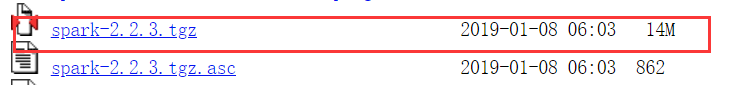
本机host配置maven.aliyun.com的域名，截图如下：



* 1. **编译Spark2.2.3源码**

下载源码：<https://archive.apache.org/dist/spark/spark-2.2.3/>

wget https://archive.apache.org/dist/spark/spark-2.2.3/spark-2.2.3.tgz



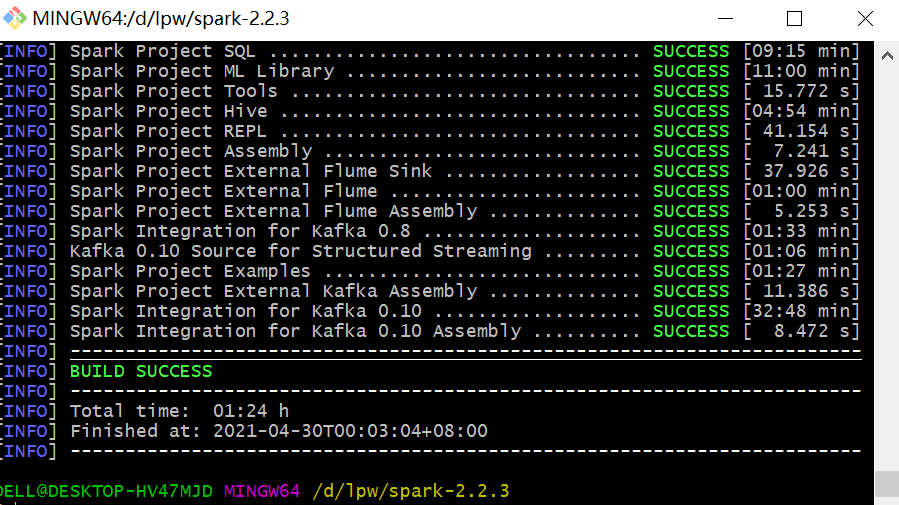
进入源码的根目录（pom.xml所在的目录）

* 编译命令

mvn -DskipTests clean package



* 验证编译是否成功



以上截图以windows平台为测试平台，linux平台同样适合。

1. **编译安装**

用本项目中Instrument文件夹下的scala文件分别替换原生spark中的同名文件。之后在spark根目录下使用mvn clean package -DskipTests进行编译。编译完成后即可将该修改过的spark，开启的lpw使用之旅。本工程中添加了许多便于测试及观察缓存替换效果的额外日志输出。用户可根据自己的需求将这些输出注释掉。

* 1. **文件更新列表**

替换该工程目录下的文件，文件更新列表如下：

|  |  |
| --- | --- |
| **文件名** | **原文件位置** |
| MemoryStore.scala | core/src/main/scala/org/apache/spark/storage/memory/MemoryStore.scala |
| BlockId.scala | core/src/main/scala/org/apache/spark/storage/BlockId.scala |
| BlockManagerMaster.scala | core/src/main/scala/org/apache/spark/storage/BlockManagerMaster.scala |
| BlockManagerMasterEndpoint.scala | core/src/main/scala/org/apache/spark/storage/BlockManagerMasterEndpoint.scala |
| BlockManagerMessages.scala | core/src/main/scala/org/apache/spark/storage/BlockManagerMessages.scala |
| BlockManagerSlaveEndpoint.scala | core/src/main/scala/org/apache/spark/storage/BlockManagerSlaveEndpoint.scala |
| RDD.scala | core/src/main/scala/org/apache/spark/rdd/RDD.scala |
| DAGScheduler.scala | core/src/main/scala/org/apache/spark/scheduler/DAGScheduler.scala |
| ResultTask.scala | core/src/main/scala/org/apache/spark/scheduler/ResultTask.scala |
| ShuffleMapTask.scala | core/src/main/scala/org/apache/spark/scheduler/ShuffleMapTask.scala |

1. **测试样例**

在本项目中的cacheProcessExample文件夹中包含一个motivationWordCount.scala文件。该文件为测试样例,打包后得到cacheProcessExample-1.0.jar。你可以使用以下两种方法中的任意一种来运行该样例：

* 1. **使用方式1 （推荐）**

创建一个maven项目，将该文件的代码放入该maven项目中，之后将其打包为jar文件。通过spark-submit的方式来运行。运行时需要传入两个命令行参数，第一个为spark的master url，第二个为一个英文文本文件位置（请确保文件中包含i和id开头的单词。如果没有，请将代码第28行中item.\_1.startsWith("i")里的参数改为某个出现在文件中的单词的首字母）。

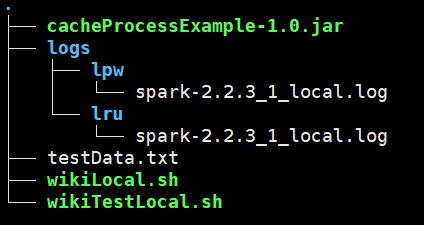
1. **授予权限**

* Spark-submit授予权限。如lru/spark-2.2.3/bin 目录、lpw/spark-2.2.3/bin添加执行权限,具体命令如下：



* 脚本授予权限。
* 待测应用授予权限。

整个目录截图如下：



上面的示例假设你当前正位于编译好的spark的根文件夹下，并且你将包含

样例代码：cacheProcessExample-1.0.jar

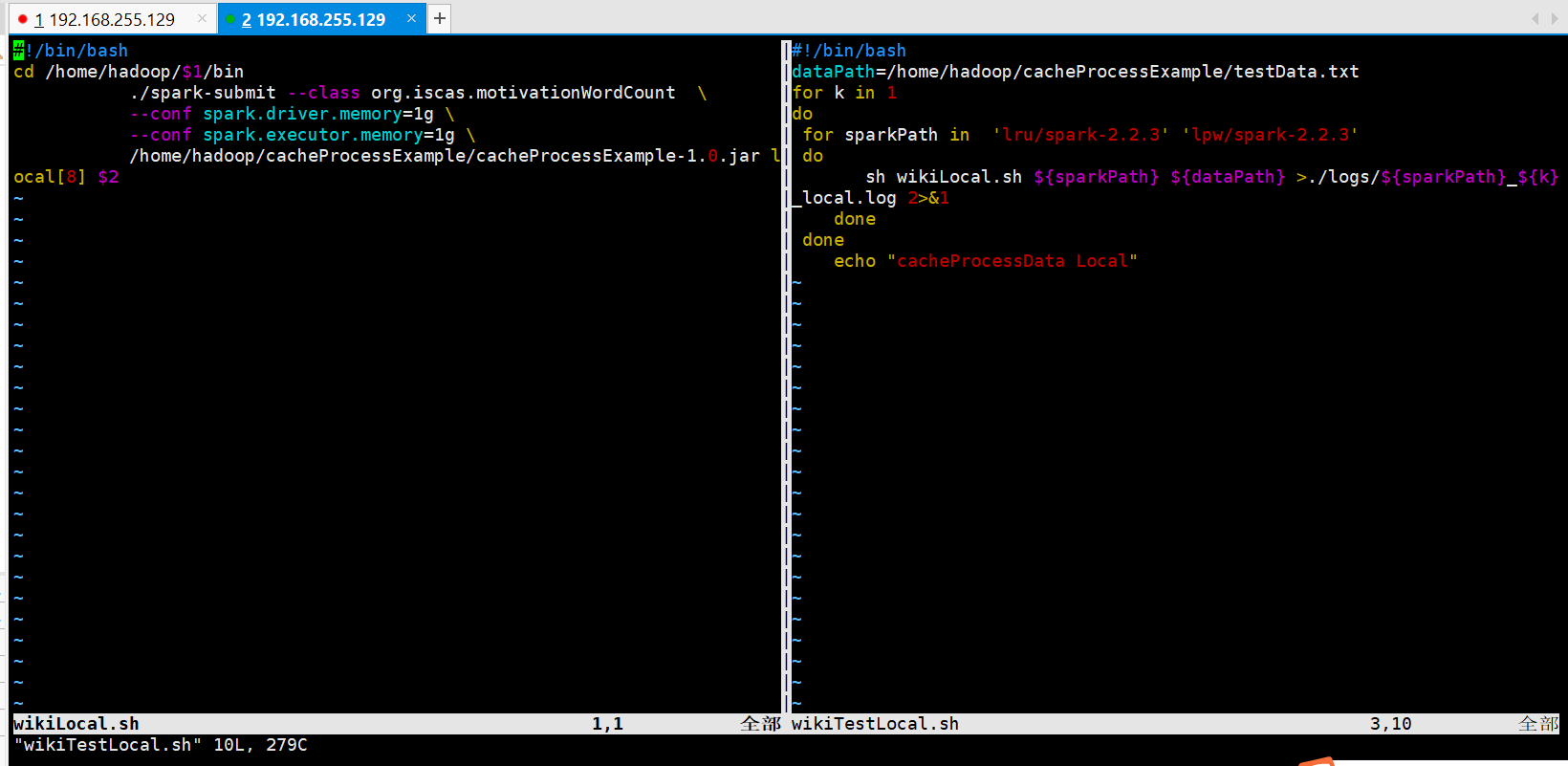
测试数据文件：testData.txt

脚本：wikiTestLocal.sh 和wikiLocal.sh

注：如果需要使用集群模式来测试，需要修改master的输入参数和输入数据路径hdfs路径，并同时将测试数据文件testData.txt上传到hdfs。

1. **运行脚本**

脚本内容如下：



Local[8] ： 表示本地模式8个线程运行。

提交并运行脚本 sh wikiTestLocal.sh

1. **结果查看**

详细日志见lpw/spark-2.2.3\_1\_local.log 和lru/spark-.2.3\_1\_local.log.

环境配置信息如下：内存设置driver.memory=1g executor.memory=1g，应用以local[8]模式运行，结果统计如下：

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **详细内容** | **LRU** | **LPW** |
| 1 | 执行时间(/ms) | 108995 | 83641 |
| 2 | 替换block的频率(/次数)  [根据”Removed rdd\_”搜索] | 50 | 55 |
| 3 | 缓存block的频率(/次数)  [根据” (Added rdd\_”搜索) | 62 | 58 |