**说明**

一、背景介绍：

本次研究使用的是大数据存储的开源框架Alluxio。Alluxio是世界上第一个以内存为中心的分布式存储虚拟系统，它在大数据计算框架和底层存储系统之间建立了连接，从而统一了数据的访问方式，不同的计算框架只需要连接到Alluxio就可以访问到底层不同存储系统里的数据，另外不同的计算框架也可以因为Alluxio实现多任务的数据共享。目前Alluxio只支持MEN、SSD和HDD这三种存储介质，但是由于MEM具有易失性，所以Alluxio就无法保证在断电后的数据完整性。

所以这次项目的最终工作就是通过源码改造以及重编译，使得Alluxio在原来的基础上增加对Nvdimm的支持。

二、Mvdimm模块介绍：

在com/yh目录下，有个NvdimmUtils.java类，这个类是负责把数据从java项目里转移到Nvdimm里，这个类里有个带native的方法，没有实现。下面分为以下几个步骤生成想要的类：

1、编译java类，使用命令 javac NvdimmUtils.java 会看到在目标目录下生成文件NvdimmUtils.class

2、生成c头文件，使用命令 javah （路径）\_NvdimmUtils 我的是com\_yh\_NvdimmUtils.h，打开会发现就是一个方法的声明。



3、编写Ｃ文件，在该目录下新建Write2Nvdimm.c，并把com\_yh\_NvdimmUtils.h里的内容复制进来，然后假如自己的代码（即往Nvdimm里写数据的代码），具体的内容我已经实现，查看Write2Nvdimm.c即可。

4、使用命令 gcc -fPIC -D\_REENTRANT -I /usr/java/jdk1.8.0\_111/include -I /usr/java/jdk1.8.0\_111/include/linux -I /usr/include -c Write2Nvdimm.c

生成Write2Nvdimm.o

5、使用命令 gcc -shared Write2Nvdimm.o -lArxcisAPI -o libWrite2Nvdimm.so 生成libWrite2Nvdimm.so

6、将so库文件放到-Djava.library.path目录下 就可以全局共享。

三、Alluxio源码编译说明

1、对Alluxio源码编译使用的是Maven，这里首先得安装Maven，如何安装网上很多，<https://jingyan.baidu.com/article/1709ad808ad49f4634c4f00d.html> 这个讲的很详细，可以借鉴下。

2、安装了Maven后，进入到alluxio-1.3.0 目录下 使用命令 mvn clean install 可以重新编译源码，要是感觉比较慢的话，也可以使用 mvn clean install –Dmaven.test.skip=true

3．上述命令执行完了，就会在alluxio-1.3.0/assembly/target 目录下发现重新生成了两个新的jar文件。把这两个jar包拷到目标工程下即可使用。

四、关于Alluxio源码的分析

首先是对Alluxio已经汉化过了，汉化的部分图片在pic\_save文件夹里。

然后花了一些时间对其进行了分析，然后通过源码编译进行验证，具体的验证方式都是通过输出的形式进行验证的，具体读写的输出信息都在alluxioOutput.txt里，包括读写流程以及各个类执行的顺序还有写文件的路径，还有元数据的信息。其他的详细信息，在我论文里的项目实现章节里有讲。