论文代码运行环境配置

代码链接：点此处

关于Ubuntu的一些tips:

（1）一定使用虚拟机，并且只推荐virtualbox和vmware workstation，前者完全免费后者需要百度搜个许可证，千万不要装双系统；

（2）安装ubuntu可参照virtualbox安装或者VMware安装Ubuntu；

（3）关于文件编辑，一般使用vim，搜索一下如何用vim插入编辑、保存、编辑后不保存 这些基本操作即可；

（4）推荐使用命令行输入poweroff来关机，输入reboot来重启，如果直接用vbox或者vm自带的关机键，小概率会引起ubuntu系统崩溃

安装Ubuntu与基础配置

建议选择ubuntu22，20或18。

磁盘空间建议至少30G，内存4G，除非你用的虚拟机软件比较方便扩容（Virtualbox很难做到），否则最好开始就开大一点磁盘空间。

由于后续用到的docker都在root权限下运行，所以先创建root密码：

```
sudo passwd root
```

以后切换到root用户只需要输入：

```
su
```

从root切换到普通用户则输入：

```
su 你的名字
```

现在执行ubuntu常规操作——apt更新（如果在root用户状态下可以省略sudo）：

```
sudo apt update
```

选择1：不使用Spark，直接运行源代码

如果不想在任何大数据框架下运行，则直接按照Github上的指示来执行即可。

需要注意的是，先要安装好miniconda：

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

然后找到这个.sh文件的安装目录，执行：

```
sh ./Miniconda3-latest-Linux-x86_64.sh
```

安装过程中，有yes的一定要输入yes，其余的直接回车。安装好后再开始conda create等操作，按照普通python程序运行即可。

选择2：使用Spark

如果要写一些用到PySpark库的代码，就需要在spark框架下运行。

采用的是Docker来管理镜像和容器。关于Docker、镜像、容器的科普，可以参考这篇文章。简单来说，容器就是创建了一个新的独立小空间，可以有自己的环境，不受宿主影响，从外表上看，是虚拟机中的一个不同于宿主的命令行界面（用户名、文件都与宿主不同）。**下面的操作请确保在root用户下执行。**

步骤1：安装Docker（先安装curl）和docker-compose

```
apt install curl
curl -sSL https://get.daocloud.io/docker | sh
apt install docker-compose
```

为了正常使用共享空间，给本次任务创建了一个新的目录，例如我是在/home/lyd下新建了一个文件夹spark/（顺便安装一下vim编辑器）

```
mkdir spark
cd spark
apt install vim
vim docker-compose.yml
```

这样就进入了一个新的配置文件docker-compose.yml文件中，按下**i**开始编辑，粘贴如下内容即可：

```
version: '2'

services:
  spark:
    image: s1mplecc/spark-hadoop:3
    hostname: master
    environment:
      - SPARK_MODE=master
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    volumes:
      - /home/lyd/spark/share:/opt/share
    ports:
      - '8080:8080'
      - '4040:4040'
      - '8088:8088'
      - '8042:8042'
      - '9870:9870'
      - '19888:19888'
  spark-worker-1:
    image: s1mplecc/spark-hadoop:3
```

```
    hostname: worker1
    environment:
      - SPARK_MODE=worker
      - SPARK_MASTER_URL=spark://master:7077
      - SPARK_WORKER_MEMORY=1G
      - SPARK_WORKER_CORES=1
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    volumes:
      - /home/lyd/spark/share:/opt/share
    ports:
      - '8081:8081'
  spark-worker-2:
    image: s1mplecc/spark-hadoop:3
    hostname: worker2
    environment:
      - SPARK_MODE=worker
      - SPARK_MASTER_URL=spark://master:7077
      - SPARK_WORKER_MEMORY=1G
      - SPARK_WORKER_CORES=1
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    volumes:
      - /home/lyd/spark/share:/opt/share
    ports:
      - '8082:8081'
```

注意要把三个volumes的/home/lyd/spark改成你刚才创建的目录（后面简称spark目录），也就是这个docker-compose.yml所在的目录。编辑完之后按ESC然后输入:wq保存（注意有冒号）。

步骤2：创建容器

在建立容器之前，需要下载容器所依赖的镜像：

```
docker pull s1mplecc/spark-hadoop:3
```

在spark目录下执行：

```
docker-compose up -d
```

这样就启动了一个master节点和两个worker节点，如果要关闭节点，则输入：

```
docker-compose stop
```

这时看一下这三个容器的名称和状态：

```
docker ps
```

会得到类似以下的输出：

```
CONTAINER ID   IMAGE                     COMMAND                  CREATED
STATUS         PORTS


                                NAMES

b1f2e00c08c3   s1mplecc/spark-hadoop:3   "/opt/bitnami/script…"   47 hours ago
Up 4 seconds   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp


                                spark_spark-worker-1_1

3f55e45360e0   s1mplecc/spark-hadoop:3   "/opt/bitnami/script…"   47 hours ago
Up 4 seconds   0.0.0.0:4040->4040/tcp, :::4040->4040/tcp, 0.0.0.0:8042-
>8042/tcp, :::8042->8042/tcp, 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp,
0.0.0.0:8088->8088/tcp, :::8088->8088/tcp, 0.0.0.0:9870->9870/tcp, :::9870-
>9870/tcp, 0.0.0.0:19888->19888/tcp, :::19888->19888/tcp   spark_spark_1

1149bb8e3ba5   s1mplecc/spark-hadoop:3   "/opt/bitnami/script…"   47 hours ago
Up 4 seconds   0.0.0.0:8082->8081/tcp, :::8082->8081/tcp


                                spark_spark-worker-2_1
```

以我的输出为例，可以看到master节点（名字不带worker的，我的是中间那个）的ID是3f55e45360e0，取前两位3f即可，后面会用到。

步骤3：配置共享文件夹

之前创建容器的时候，.yml文件里有创建共享文件夹的命令（比如我的共享文件夹是spark/share/）。为了能在容器里面用conda，需要把miniconda的安装包移动到spark/share目录下，安装命令为：

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

然后把./Miniconda3-latest-Linux-x86_64.sh这个文件移动到share目录。

再把这次要跑的代码从Github下载，解压后把这个文件夹也移动到share目录。其中**需要修改environment.yml文件**，改为：

```
name: big_data
# Environment to replicate estimation results for the ULSDPB-paper.
dependencies:
  - python=3.8.5
  - numpy=1.19.2
  - scipy=1.5.0
  - numba=0.50.1
  - pip=20.2.4
```

如果按照原来的environment.yml来配置的话，会导致一些python包冲突，比较麻烦。

步骤4：进入容器

在步骤2，我们知道了三个容器的ID，现在我们先进入master容器。我的ID前两位是3f，则输入：

```
docker exec -it 3f bash
```

可以看到命令行最前面变成了root@master:/opt#，这说明成功进入了容器。

如果想退出容器，按Ctrl+D即可。

下面进入share的ulsdpb目录（我把之前移动到share的代码文件夹改名为ulsdpb），并安装miniconda：

```
cd share/ulsdpb
sh ./Miniconda3-latest-Linux-x86_64.sh
```

接下来有yes就输入yes，没有就按回车，**尤其是最后一步一定要输入yes**，否则要删除根目录下的miniconda3文件夹然后再来一遍，因为这样的容器中无法配置bashrc。

由于港中深的网几乎无法访问conda源，输入以下几个命令（可能有的用不到但是配置了总没错）添加清华源，并删除默认源：

```
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/msys2/
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/conda-forge
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch/
conda config --add channels
https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
conda config --set show_channel_urls yes
conda config --remove channels defaults
```

然后查看以下现在用的源是什么：

```
conda config --show-sources
```

如果得到如下输出，说明把源配置好了：

```
==> /.condarc <==
channels:
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/cloud/pytorch/
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/r/
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/free/
  - https://mirrors.tuna.tsinghua.edu.cn/anaconda/pkgs/main/
show_channel_urls: True
```

在ulsdpb文件夹下创建conda环境：

```
conda env create -f environment.yml
```

创建了之后我们进入这个新的环境（在environment.yml可以看到环境名称为big_data）：

```
conda activate big_data
```

因为有一个lda包没法用conda下载，所以在这里使用pip下载：

```
pip install lda==2.0.0 -i https://pypi.tuna.tsinghua.edu.cn/simple
```

至此，所需要的代码运行环境全部配置完毕。

此后若想直接跑源代码，按照Github的教程，从第4步开始执行即可。但是这样运行的程序是用不到spark框架的，需要修改一些代码才能用到。

步骤5：使用Spark

为了让其他两个容器也有这样的运行环境，先要分别对它们执行完整的”步骤4“。

（修改代码的工作，这部分待完善）

修改完后的效果是，能在spark框架下运行，因为代码中至少包含了一处sparkContext作为框架的入口。

例如，我们需要运行的python程序为test.py，则进入master容器后输入：

```
spark-submit --master spark://master:7077 /opt/share/test.py
```

在虚拟机的浏览器中输入localhost:8080即可看到spark任务的执行状态。这是spark专属的webUI，只要用docker把三个容器启动之后就可以访问。