实验1-MPI Programming报告

191840373 朱家萱

一、环境配置

1、尝试在单机上安装并运行MPI环境。（MPICH或 者OpenMPI等）

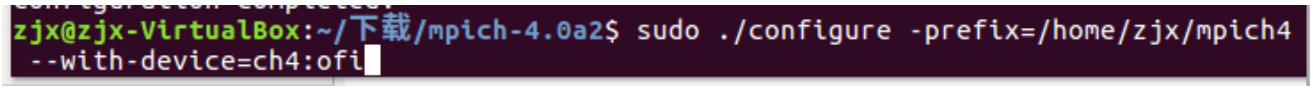
网络上的教程对于具体的Ubuntu来说非常没有普适性，常常需要根据报错提示自主添加命令。如在configure时，首先要获取管理者权限，其次，键入

```
sudo ./configure -prefix=/home/zjx/mpich4
```

会得到大串的报错提示，根据提示，需要在上一个命令末尾加上

```
--with-device=ch4:ofi
```

才能成功地进行软件配置与检查。



在接下来的编译与安装中，每次都需要获取管理者权限才可以make&&make install，但直接进入root模式是不安全的。

```
zjx@zjx-VirtualBox: ~/下载/mpich-4.0a2
  MOD       src/binding/fortran/use_mpi/mpi_constants.mod-stamp
  MOD       src/binding/fortran/use_mpi/mpi_sizeofs.mod-stamp
  MOD       src/binding/fortran/use_mpi/mpi_base.mod-stamp
  MOD       src/binding/fortran/use_mpi/mpi.mod-stamp
  GEN       lib/libmpifort.la
ar: `u' modifier ignored since `D' is the default (see `U')
  CXX       src/binding/cxx/initcxx.lo
  CXXLD     lib/libmpicxx.la
ar: `u' modifier ignored since `D' is the default (see `U')
  CC        src/env/mpichversion.o
  CCLD      src/env/mpichversion
  CC        src/env/mpivars.o
  CCLD      src/env/mpivars
cp -p src/env/mpicc.bash src/env/mpicc
cp -p src/env/mpifort.bash src/env/mpifort
cp -p src/env/mpicxx.bash src/env/mpicxx
make[2]: Leaving directory '/home/zjx/下载/mpich-4.0a2'
Making all in examples
make[2]: Entering directory '/home/zjx/下载/mpich-4.0a2/examples'
  CC        cpi.o
  CCLD      cpi
make[2]: Leaving directory '/home/zjx/下载/mpich-4.0a2/examples'
make[1]: Leaving directory '/home/zjx/下载/mpich-4.0a2'
zjx@zjx-VirtualBox:~/下载/mpich-4.0a2$ sudo make install
```

修改环境变量后，完成安装。测试运行成功：

```
zjx@zjx-VirtualBox:~/下载/mpich-4.0a2/examples$ sudo /home/zjx/mpich4/bin/mpicc
hellow.c -o hellow
zjx@zjx-VirtualBox:~/下载/mpich-4.0a2/examples$ sudo /home/zjx/mpich4/bin/mpirun
 -np 4 ./hellow
Hello world from process 2 of 4
Hello world from process 0 of 4
Hello world from process 1 of 4
Hello world from process 3 of 4
```

二、代码运行

1、题目一：用MPI_Reduce接口改写大数组各元素开平方求和 (data[N], data[i]=i*(i+1)) 的代码（可通过命令行传入 N的值，比如1000，10000，100000）；

以下截取2、3、4、5、6、10、11、12个进程数的运行结果截图（N键入10000）：

```
I am process0. I calculate that SqrtSum =49999998.349880.
It took 0.000176 seconds
```

```
I am process0. I calculate that SqrtSum =49999998.349880.
It took 0.000191 seconds
```

```
I am process0. I calculate that SqrtSum =49999998.349880.
It took 0.000134 seconds
```

```
I am process0. I calculate that SqrtSum =49999998.349880.
It took 0.000140 seconds
```

```
I am process0. I calculate that SqrtSum =49999998.349880.
It took 0.002061 seconds
```

```
I am process0. I calculate that SqrtSum =49999998.349880.
It took 0.000156 seconds
```

```
I am process0. I calculate that SqrtSum =49999998.349880.
It took 0.000159 seconds
```

```
I am process0. I calculate that SqrtSum =49999998.349880.
It took 0.000223 seconds
```

通过比较，不同进程数对结果没有影响，因为此算法一定会得到一个确切的值。比较时长，发现时长会在一些特定的值达到峰值，然后下降，递增，到达峰值，下降，递增，到达峰值，很大可能是巧合，网络搜索后了解到需要的时间与数据量，线程数，代码逻辑，电脑性能有关。

2、题目二：用MPI_Send和MPI_Receive接口计算积分：y=x^3， 求其在[10,100]区间的积分

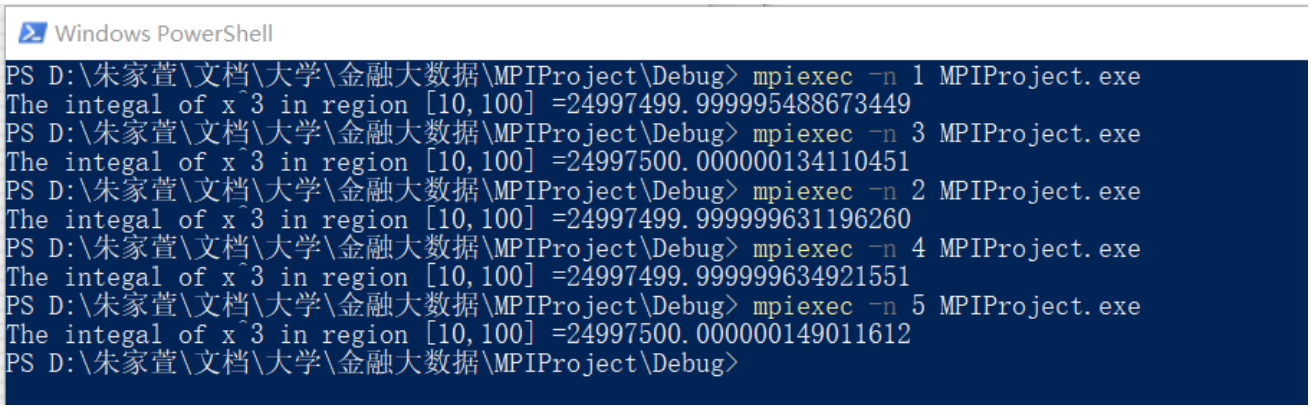
不同系统的运行结果截图：

①ubuntu

```
zjx@zjx-VirtualBox:~/BD$ mpicc jifen.cpp -o test
zjx@zjx-VirtualBox:~/BD$ mpirun -np 1 ./test
The integal of x^3 in region [10,100] =24997499.999994605779648
zjx@zjx-VirtualBox:~/BD$ mpirun -np 2 ./test
The integal of x^3 in region [10,100] =24997499.999999381601810
zjx@zjx-VirtualBox:~/BD$ mpirun -np 3 ./test
The integal of x^3 in region [10,100] =24997500.000000022351742
zjx@zjx-VirtualBox:~/BD$ mpirun -np 4 ./test
The integal of x^3 in region [10,100] =24997499.999999560415745
zjx@zjx-VirtualBox:~/BD$ mpirun -np 5 ./test
The integal of x^3 in region [10,100] =24997500.000000126659870
```

②Windows

```
Windows PowerShell
PS D:\朱家萱\文档\大学\金融大数据\MPIProject\Debug> mpiexec -n 1 MPIProject.exe
The integal of x^3 in region [10,100] =24997499.999995488673449
PS D:\朱家萱\文档\大学\金融大数据\MPIProject\Debug>
```



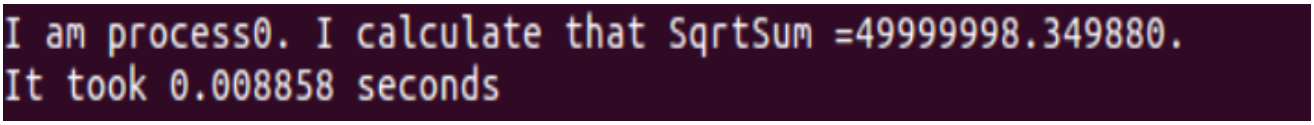
横向对比，未发现进程数与结果精确度有显著性关系；纵向对比，Linux环境下的结果比Windows下更多样。具体原因还没有研究出来，我想可能是因为编译器不同导致的，与环境关系不大。

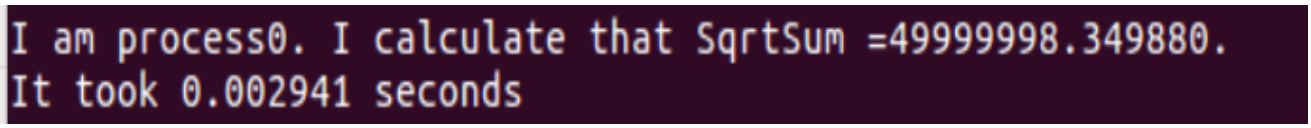
（不同进程数运行所需时间的评估在3、）

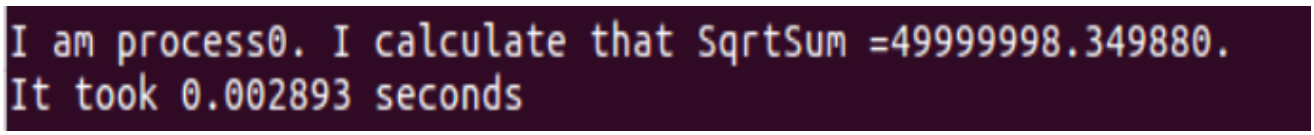
3、尝试在多个节点上运行上述MPI程序，可设置不同的进程数对结果进行比较，并评估所需时间。

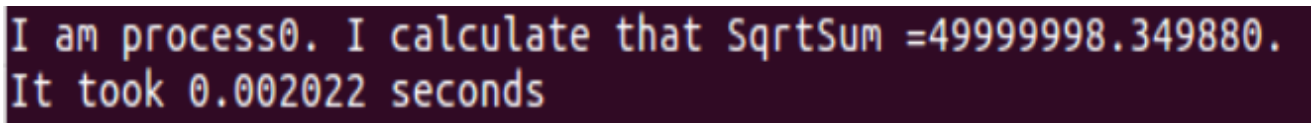
（1）各元素开平方求和的代码运行结果：

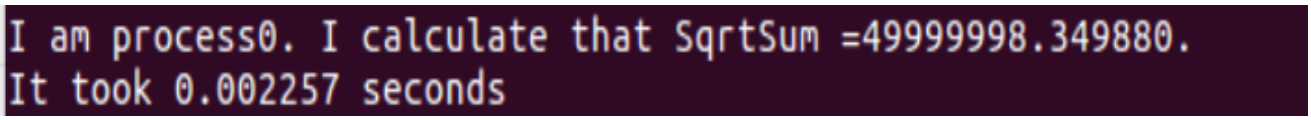
节点数为4，N=10000时，设置不同进程数3、4、5、6、10、11、100的运行结果如图：

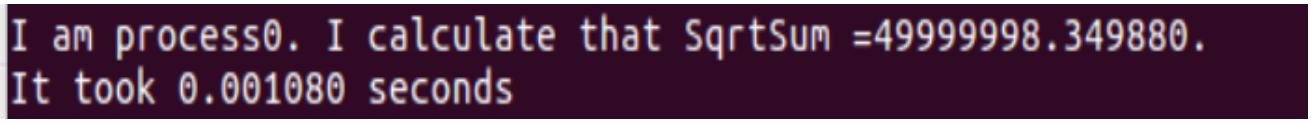


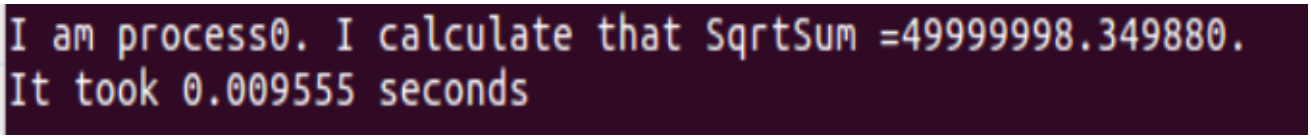












结果显示，在进程数较小时，进程越多，耗时越少，进程数较大时反而耗时更多，可见内存中进程的数量越多并不越能提高系统的并发度和效率。

线程负载均衡的话，如果线程切换时间能很好的弥补掉线程挂起等待的时间，各个线程交错执行，完全占用计算资源，计算速度才会最快。

（2）求积分的代码运行结果：

①设置不同进程数及节点数对结果精确度的影响：

5个进程4个节点：

```
root@host1:/sharee# mpirun -np 5 -host host2,host3,host4 ./test
The integal of x^3 in region [10,100] =24997500.000000126659870
```

4个进程4个节点：

```
The integal of x^3 in region [10,100] =24997499.999999560415745
```

10个进程4个节点：

```
The integal of x^3 in region [10,100] =24997499.999999873340130
```

2个进程2个节点：

```
The integal of x^3 in region [10,100] =24997499.999999381601810
```

（代码及运行结果的具体分析在下面的“问题总结及解决方案”中）

对结果的比较：进程数不同，结果有差异，进程数较小时，进程数越多，结果越精确。

②节点数相同，不同进程数1、2、3、4、5、10、11、12对运行时长的影响

```
The integal of x^3 in region [10,100] =24997499.999994605779648
It took 0.333345 seconds
```

```
The integal of x^3 in region [10,100] =24997499.999999381601810
It took 0.194265 seconds
```

```
The integal of x^3 in region [10,100] =24997500.000000022351742
It took 0.120123 seconds
```

```
The integal of x^3 in region [10,100] =24997499.999999560415745
It took 0.089985 seconds
```

```
The integal of x^3 in region [10,100] =24997500.000000126659870
It took 0.079111 seconds
```

```
The integal of x^3 in region [10,100] =24997499.999999873340130
It took 0.044805 seconds
```

```
The integal of x^3 in region [10,100] =24997500.000000186264515
It took 0.041441 seconds
```

```
The integal of x^3 in region [10,100] =24997500.000000521540642
It took 0.040660 seconds
```

在求积分的程序里，进程数与运行时长有很显著的负相关趋势，即节点数不变，进程数越多，用时越短。

③Linux与Windows下运行的效率

```
PS D:\朱家萱\文档\大学\金融大数据\MPIProject\Debug> mpiexec -n 2 MPIProject.exe
The integal of x^3 in region [10,100] =24997499.999999631196260
It took 0.460000 seconds
```

```
The integal of x^3 in region [10,100] =24997499.999999381601810
It took 0.170278 seconds
```

```
PS D:\朱家萱\文档\大学\金融大数据\MPIProject\Debug> mpiexec -n 5 MPIProject.exe
The integal of x^3 in region [10,100] =24997500.000000149011612
It took 0.213000 seconds
```

```
The integal of x^3 in region [10,100] =24997500.000000126659870
It took 0.079817 seconds
```

通过多组对比，可以看到，Linux下的运行速度明显快于Windows。

> 由于linux的轻量，而且是开源的，有很多支持者。没有图形用户界面，一切都是代码操作，注重效率。
>
> 而windows是给大众用的，关注的地方比较多，例如图形用户界面。所以程序运行效率比不上linux。

三、问题总结及解决方案、其它

1、实验一中遇到的docker使用的技巧及方法总结

（1）如何删除镜像

```
zjx@zjx-VirtualBox:~$ sudo docker rmi -f a5e94a36c541
Untagged: hello-world:latest
Untagged: hello-world@sha256:61bd3cb6014296e214ff4c6407a5a7e7092dfa8eefdbbec539e
133e97f63e09f
Deleted: sha256:a5e94a36c5413650b172bbd0957313671b6a7076ba09e0612ed26f64b3d4cb6c
```

（2）在某一镜像中访问另一镜像

```
zjx@zjx-VirtualBox:~$ sudo docker exec -it mpi-host5 bash
root@host5:/# service ssh start
 * Starting OpenBSD Secure Shell server sshd                          [ OK ]
root@host5:/# exit
exit
zjx@zjx-VirtualBox:~$ sudo docker exec -it mpi-host3 bash
root@host3:/# ssh root@host5
The authenticity of host 'host5 (172.17.0.4)' can't be established.
ECDSA key fingerprint is 6e:f3:76:6d:68:c9:1e:bf:d7:7a:3f:d0:4a:7b:9b:30.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'host5,172.17.0.4' (ECDSA) to the list of known hosts
.
Welcome to Ubuntu 14.04 LTS (GNU/Linux 4.4.0-170-generic i686)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@host5:~#
```

（3）开启容器、停止容器、查看所有容器并进行删除

关闭docker后，再次进入容器需要重新开启

```
zjx@zjx-VirtualBox:~/share$ sudo docker start mpi-host1
mpi-host1
zjx@zjx-VirtualBox:~/share$ sudo docker exec -it mpi-host1 bash
root@host1:/#
```

重启：

```
docker restart +container name
```

停止：

```
docker stop +container name
```

查看所有容器并删除：

```
zjx@zjx-VirtualBox:~$ sudo docker ps -a
```

```
zjx@zjx-VirtualBox:~$ sudo docker rm dcbaebe45d4f
dcbaebe45d4f
```

（4）将容器挂载同一本地文件夹来实现共享代码和可执行文件

```
zjx@zjx-VirtualBox:~$ sudo docker run -dit --hostname host1 --name mpi-host1 -v
/home/zjx/share:/sharee mpi-ubuntu /bin/bash
aa58890d6ec81d44c80aa292a2b77b5adec2f193896e1d7229cb6b6702d57065
```

宿主机share文件夹的内容

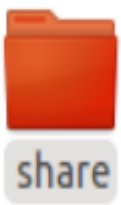
```
zjx@zjx-VirtualBox:~$ sudo docker exec -it mpi-host1 bash
root@host1:/# cd sharee
root@host1:/sharee# ls
jifen.cpp
```

成功挂载

（5）发现对宿主机用来共享的文件夹没有操作权限，进行解锁

```
zjx@zjx-VirtualBox:~$ sudo chown -R $USER /home/zjx/share
zjx@zjx-VirtualBox:~$
```

此时文件夹便可以进行修改、删除：



2、报错和解决方案

（1）报错：ssh_askpass:No such file or directory

sudo apt-get install ssh-askpass后解决

（2）报错：error trying to exec 'cc1plus': execvp: 没有该文件或目录 解决方案

执行：sudo apt-get install g++ 解决。

（3）报错：

```
zjx@zjx-VirtualBox:~/share$ mpirun -np 2 -host 172.17.0.2,172.17.0.3 jifen
ssh-askpass[347]: Trying to grab keyboard ...
ssh-askpass[347]: Could not grab keyboard (someone else already has it)
Host key verification failed.
Host key verification failed.
```

```
zjx@zjx-VirtualBox:~/share$ mpirun -np 2 -host 172.17.0.2,172.17.0.3 jifen
ssh-askpass[558]: Trying to grab keyboard ...
ssh-askpass[558]: Could not grab keyboard (someone else already has it)
Host key verification failed.
Host key verification failed.
```

```
zjx@zjx-VirtualBox:~/share$ mpirun -np 2 -host host1,host2 jifen
ssh: Could not resolve hostname host1: Name or service not known
ssh: Could not resolve hostname host2: Name or service not known
```

原因是没有好好理解多机运行的原理，应该以某一host1为主节点，设置好ssh免密和其hosts文件。

（4）尝试在容器里mpirun，报错：

```
root@host1:/sharee# mpirun -np 2 -host host1,host2 jifen
Unexpected end of /proc/mounts line `overlay / overlay rw,relatime,lowerdir=/var
/lib/docker/overlay2/l/WRDEXLEVPRQ4NKXFN7ZK2SJ3JF:/var/lib/docker/overlay2/l/EIE
W2JNDIKL56LI3V673XGKFL5:/var/lib/docker/overlay2/l/FMILJZAAKPL5ZV76QWS3LUXRCS:/v
ar/lib/docker/overlay2/l/V4FHJPC5FTYTS6QFIT4NB66ISR:/var/lib/docker/overlay2/l/V
O3VGQ5OIICOQLUDLVQO2TAXYX,upperdir=/var/lib/docker/overlay2/3087310148cd381e8df3
0d81fbdcf00bbab6ba3c0715dfe4a2d278c065409fbe/diff,workdir=/var/lib/docker/overla
y2/3087310148cd381e8df30d81fbdcf00bbab6ba3c0715dfe4a2d278c065409fbe/'
Unexpected end of /proc/mounts line `overlay / overlay rw,relatime,lowerdir=/var
/lib/docker/overlay2/l/VZOPMP3SW77O4U3OXH2VRFJRVR:/var/lib/docker/overlay2/l/EIE
W2JNDIKL56LI3V673XGKFL5:/var/lib/docker/overlay2/l/FMILJZAAKPL5ZV76QWS3LUXRCS:/v
ar/lib/docker/overlay2/l/V4FHJPC5FTYTS6QFIT4NB66ISR:/var/lib/docker/overlay2/l/V
O3VGQ5OIICOQLUDLVQO2TAXYX,upperdir=/var/lib/docker/overlay2/b289ad4f885b7c24b42d
03212cbd2661331043954ef1896a2bebd2cf922d745d/diff,workdir=/var/lib/docker/overla
y2/b289ad4f885b7c24b42d03212cbd2661331043954ef1896a2bebd2cf922d745d/'
jifen: error while loading shared libraries: libmpi.so.0: cannot open shared obj
ect file: No such file or directory
jifen: error while loading shared libraries: libmpi.so.0: cannot open shared obj
ect file: No such file or directory
--------------------------------------------------------------------------
mpirun noticed that the job aborted, but has no info as to the process
that caused that situation.
--------------------------------------------------------------------------
```

原因是在宿主机编译好了放在容器里并不能正常运行，得在容器里编译成功才可以运行。

（5）报错

```
Unexpected end of /proc/mounts line `overlay / overlay rw,relatime,lowerdir=/var
/lib/docker/overlay2/l/WRDEXLEVPRQ4NKXFN7ZK2SJ3JF:/var/lib/docker/overlay2/l/EIE
W2JNDIKL56LI3V673XGKFL5:/var/lib/docker/overlay2/l/FMILJZAAKPL5ZV76QWS3LUXRCS:/v
ar/lib/docker/overlay2/l/V4FHJPC5FTYTS6QFIT4NB66ISR:/var/lib/docker/overlay2/l/V
O3VGQ5OIICOQLUDLVQO2TAXYX,upperdir=/var/lib/docker/overlay2/3087310148cd381e8df3
0d81fbdcf00bbab6ba3c0715dfe4a2d278c065409fbe/diff,workdir=/var/lib/docker/overla
y2/3087310148cd381e8df30d81fbdcf00bbab6ba3c0715dfe4a2d278c065409fbe/'
ssh: Could not resolve hostname host1.host2: Name or service not known
--------------------------------------------------------------------------
A daemon (pid 302) died unexpectedly with status 255 while attempting
to launch so we are aborting.

There may be more information reported by the environment (see above).

This may be because the daemon was unable to find all the needed shared
libraries on the remote node. You may set your LD_LIBRARY_PATH to have the
location of the shared libraries on the remote nodes and this will
automatically be forwarded to the remote nodes.
--------------------------------------------------------------------------
--------------------------------------------------------------------------
mpirun noticed that the job aborted, but has no info as to the process
that caused that situation.
```

发现是误写了指令

[运行出来了！]

```
root@host1:/sharee# mpirun -np 2 -host host1,host2 ./test
Unexpected end of /proc/mounts line `overlay / overlay rw,relatime,lowerdir=/var
/lib/docker/overlay2/l/WRDEXLEVPRQ4NKXFN7ZK2SJ3JF:/var/lib/docker/overlay2/l/EIE
W2JNDIKL56LI3V673XGKFL5:/var/lib/docker/overlay2/l/FMILJZAAKPL5ZV76QWS3LUXRCS:/v
ar/lib/docker/overlay2/l/V4FHJPC5FTYTS6QFIT4NB66ISR:/var/lib/docker/overlay2/l/V
O3VGQ5OIICOQLUDLVQO2TAXYX,upperdir=/var/lib/docker/overlay2/3087310148cd381e8df3
0d81fbdcf00bbab6ba3c0715dfe4a2d278c065409fbe/diff,workdir=/var/lib/docker/overla
y2/3087310148cd381e8df30d81fbdcf00bbab6ba3c0715dfe4a2d278c065409fbe/'
Unexpected end of /proc/mounts line `overlay / overlay rw,relatime,lowerdir=/var
/lib/docker/overlay2/l/VZOPMP3SW77O4U3OXH2VRFJRVR:/var/lib/docker/overlay2/l/EIE
W2JNDIKL56LI3V673XGKFL5:/var/lib/docker/overlay2/l/FMILJZAAKPL5ZV76QWS3LUXRCS:/v
ar/lib/docker/overlay2/l/V4FHJPC5FTYTS6QFIT4NB66ISR:/var/lib/docker/overlay2/l/V
O3VGQ5OIICOQLUDLVQO2TAXYX,upperdir=/var/lib/docker/overlay2/b289ad4f885b7c24b42d
03212cbd2661331043954ef1896a2bebd2cf922d745d/diff,workdir=/var/lib/docker/overla
y2/b289ad4f885b7c24b42d03212cbd2661331043954ef1896a2bebd2cf922d745d/'
The integal of x^3 in region [10,100] =24997499.999999381601810
```

但会有一大堆奇怪的东西，网络搜索后知

> 这可能是由于您的/ proc / mounts文件中包含大于512个字符的行,导致OpenMPI的hwloc模块无法正确解析它. Docker倾向于将非常长的行放入/ proc / mounts.您可以在openmpi-1.10.7 / opal / mca / hwloc / hwloc191 / hwloc / src / topology-linux.c中看到该错误：1677

这可以通过将其大小增加到更大的数量来修复，也忽略这个bug。权衡之后，选择忽略。

探索规律后推断，几个节点就报错几次，与其错误产生原因吻合。

（6）发现创建新容器时，sudo docker run -it与sudo docker run -dit的不同：

-it：创建后返回root@容器名

```
zjx@zjx-VirtualBox:~$ sudo docker run -it --hostname host4 --name mpi-host4 -v /
home/zjx/share:/sharee mpi-ubuntu /bin/bash
root@host4:/# service ssh start
```

-dit：创建后自动进入容器

```
zjx@zjx-VirtualBox:~$ sudo docker run -dit --hostname host3 --name mpi-host3 -v
/home/zjx/share:/sharee mpi-ubuntu /bin/bash
71e5d01fc50343b35e57dd8a9caa0a5bceede3264db3c02aea5ca7007d2c0fe4
```

且-it和-dit创建的容器之间无法用ssh协议连接，容器必须是同样的创建方式。

（7）报错：

```
root@host1:/sharee# mpicc sqrt.c -o sqrt
/tmp/ccJs7UYG.o: In function `main':
sqrt.c:(.text+0x1a1): undefined reference to `sqrt'
collect2: error: ld returned 1 exit status
```

是shared library动态链接的问题，因为有头文件<math.h>，编译时要加-lm。

> 使用math.h中声明的库函数还有一点特殊之处，gcc命令行必须加-lm选项，因为数学函数位于libm.so库文件中（这些库文件通常位于/lib目录下），-lm选项告诉编译器，我们程序中用到的数学函数要到这个库文件里找。

（8）拉取镜像之后，需要对基础镜像进行环境的配置，在此之前千万不要进行换源，会变得不幸。

四、Github网址

https://github.com/zhua-xuan/FBDP_Programme1.git