《金融大数据处理技术》课程
实验一报告

嵇泽同 191870068

目录

1.结果代码

1.1 大数组开平方求和

```
#include <mpi.h>
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>

int main(int argc, char** argv)
{
    int N = 0;
    int myid, numprocs;
    int* data = malloc(1000000 * sizeof(int));
    double SqrtSum = 0.0;
    double inte;
    clock_t startTime = clock();
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    if (myid == 0)
    {
        printf("请输入数组大小 N: ");
        fflush(stdout);
        scanf("%d", &N);
        data = malloc(N*sizeof(int));
        for (int i = 0; i < N; ++i)
        {
            data[i] = i + 1;
        }
    }
    MPI_Bcast(&N, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(data, N, MPI_INT, 0, MPI_COMM_WORLD);
    for (int i = myid; i < N; i = i + numprocs)
    {
        SqrtSum += sqrt(data[i]);
    }
     MPI_Reduce(&SqrtSum, &inte, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
    if (myid == 0)
    {
        clock_t endTime = clock();
        double totalTime = (double)(endTime - startTime);
```

```
            printf("I am process 0. SqrtSum=%f.\nTotal Time: %f s\n", inte, totalTime/CLOCKS_PER_SEC);
        }
        free(data);
        MPI_Finalize();
    }
```

1.2 近似积分

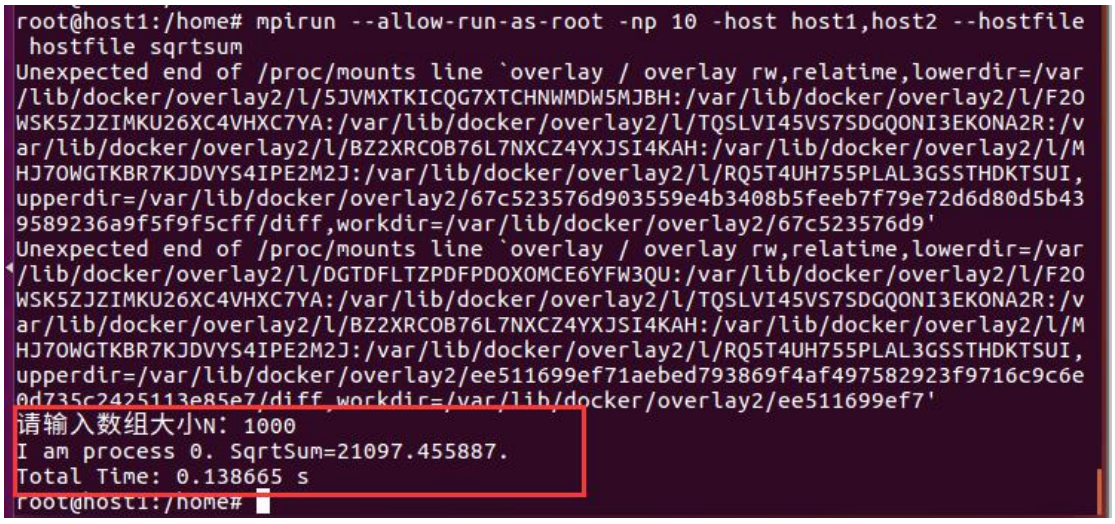
```
    #define N 100000000
    #define a 10
    #define b 100
    #include <stdio.h>
    #include <stdlib.h>
    #include <time.h>
    #include <mpi.h>
    int main(int argc, char** argv) {
        int myid, numprocs;
        MPI_Status status;
         double local = 0.0, dx = (double)(b - a) / N; /* 小矩形宽度 */
        double inte = 0.0, x, d;
        clock_t startTime = clock();
        MPI_Init(&argc, &argv);
        MPI_Comm_rank(MPI_COMM_WORLD, &myid);
        MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
        --numprocs;
        if (myid == 0)
        {
            for (int source = 1; source <= numprocs; ++source) /* 结果收集 */
            {
                MPI_Recv(&d, 1, MPI_DOUBLE, source, 99, MPI_COMM_WORLD, &status);
                inte += d;
            }
        }
        else
        {
            for (int i = myid - 1; i < N; i = i + numprocs) /* 各子节点在本地计算 */
            {
                x = a + i * dx + dx / 2;
```

```
            local += x * x * x * dx;
        }
        MPI_Send(&local, 1, MPI_DOUBLE, 0, 99, MPI_COMM_WORLD); /* 将本地计算结果传送回主节点 */
    }
    MPI_Barrier(MPI_COMM_WORLD);
    if (myid == 0)
    {
        clock_t endTime = clock();
        printf("The integal of x^3 in region [%d,%d] =%16.15f\nTotal Time: %f s\n", a, b, inte, (double)(endTime - startTime) / CLOCKS_PER_SEC);
    }
    MPI_Finalize();
}
```

（以上代码均已上传至本人 github 仓库）

2. 运行截图

2.1 大数组开平方求和



2.2 近似积分

```
root@host1:/home# mpirun --allow-run-as-root -np 10 -host host1,host2 --hostfile
 hostfile inte
Unexpected end of /proc/mounts line `overlay / overlay rw,relatime,lowerdir=/var
/lib/docker/overlay2/l/5JVMXTKICQG7XTCHNWMDW5MJBH:/var/lib/docker/overlay2/l/F2O
WSK5ZJZIMKU26XC4VHXC7YA:/var/lib/docker/overlay2/l/TQSLVI45VS7SDGQONI3EKONA2R:/v
ar/lib/docker/overlay2/l/BZ2XRCOB76L7NXCZ4YXJSI4KAH:/var/lib/docker/overlay2/l/M
HJ7OWGTKBR7KJDVYS4IPE2M2J:/var/lib/docker/overlay2/l/RQ5T4UH755PLAL3GSSTHDKTSUI,
upperdir=/var/lib/docker/overlay2/67c523576d903559e4b3408b5feeb7f79e72d6d80d5b43
9589236a9f5f9f5cff/diff,workdir=/var/lib/docker/overlay2/67c523576d9'
Unexpected end of /proc/mounts line `overlay / overlay rw,relatime,lowerdir=/var
/lib/docker/overlay2/l/DGTDFLTZPDFPDOXOMCE6YFW3QU:/var/lib/docker/overlay2/l/F2O
WSK5ZJZIMKU26XC4VHXC7YA:/var/lib/docker/overlay2/l/TQSLVI45VS7SDGQONI3EKONA2R:/v
ar/lib/docker/overlay2/l/BZ2XRCOB76L7NXCZ4YXJSI4KAH:/var/lib/docker/overlay2/l/M
HJ7OWGTKBR7KJDVYS4IPE2M2J:/var/lib/docker/overlay2/l/RQ5T4UH755PLAL3GSSTHDKTSUI,
upperdir=/var/lib/docker/overlay2/ee511699ef71aebed793869f4af497582923f9716c9c6e
0d735c2425113e85e7/diff,workdir=/var/lib/docker/overlay2/ee511699ef7'
The integal of x^3 in region [10,100] =24997499.999999746680260
Total Time: 0.069372 s
```

3. 问题总结及解决方案

3.1 操作权限不够

```
jzt@jzt-virtual-machine:~$ docker pull ubuntu:16.04
Got permission denied while trying to connect to the Docker daemon socket at uni
x:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.39/images/cr
eate?fromImage=ubuntu&tag=16.04: dial unix /var/run/docker.sock: connect: permis
sion denied
```

如上图，当按照 pdf 中给出的指令进行操作时，常常会遇到权限不够、permission denied 的问题。该问题比较容易解决，在指令前加上 sudo 即可:

```
jzt@jzt-virtual-machine:~$ sudo docker pull ubuntu:16.04
16.04: Pulling from library/ubuntu
58690f9b18fc: Pull complete
b51569e7c507: Pull complete
da8ef40b9eca: Pull complete
fb15d46c38dc: Pull complete
Digest: sha256:454054f5bbd571b088db25b662099c6c7b3f0cb78536a2077d54adc48f00cd68
Status: Downloaded newer image for ubuntu:16.04
```

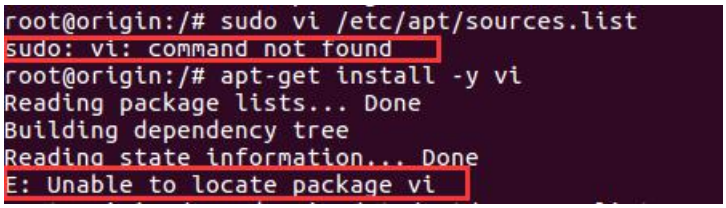
3.2 Command not found

3.2.1 Sudo

```
jzt@jzt-virtual-machine:~$ sudo docker run -dit --hostname origin --name origin ubuntu:16.04
74fc7218499c9d83e0f450951c2bfeb07843648efe4670d26fa10cf1d776fcd9
jzt@jzt-virtual-machine:~$ sudo docker exec -it origin bash
root@origin:/# sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak
bash: sudo: command not found
```

如上图，在第一次进入容器时使用 sudo 指令发现找不到。该问题同样较容易解决，调用 apt-get update 和 apt-get install sudo 指令即可。

3.2.2 Vi



如上图，同样是 command not found，但是此时调用 apt-get install 指令却无法解决问题，仍报错：Unable to locate package vi。上网查阅后得知有解决办法，但需要进行一定的操作。为了不扰乱后续的操作流程，本人在此处决定求简，尝试以 vim 来代替 pdf 教程中的 vi，而 vim 则可以顺利安装。

3.3 换源

按照教材中的指示进行换源操作修改 sources.list 并进行 update 时报出如下错误：

思路一：本人第一反应是上网查找报错信息。查阅相关资料后发现可能是因为 vim 的打开和关闭方式问题造成了文件损坏（此前提过本人以 vim 取代了教程中的 vi，虽然理论上应该没有大差异）。为了进行验证，本人将修改过的 sources.list 删除，并用 vim 新建了一个 sources.list 文件，并将此前备份的原文件复制上去并关闭。然后

再进行 update 操作。此时发现能够命令正常进行：

```
root@origin:/# sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [109 kB]
Hit:2 http://archive.ubuntu.com/ubuntu xenial InRelease
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:4 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Fetched 325 kB in 3s (104 kB/s)
Reading package lists... Done
```

于是得出结论：修改后的文件无法读取并非 vim 的原因。

思路二：阅读报错信息后（其实第一步应该就是阅读报错信息，只是当时本人有点慌乱，第一时间直接将报错信息粘贴到网上进行查找而没有先进行仔细阅读），发现信息中提到 line 3 出现了 Malformed：

```
E: Malformed line 3 in source list /etc/apt/sources.list (type)
E: The list of sources could not be read.
```

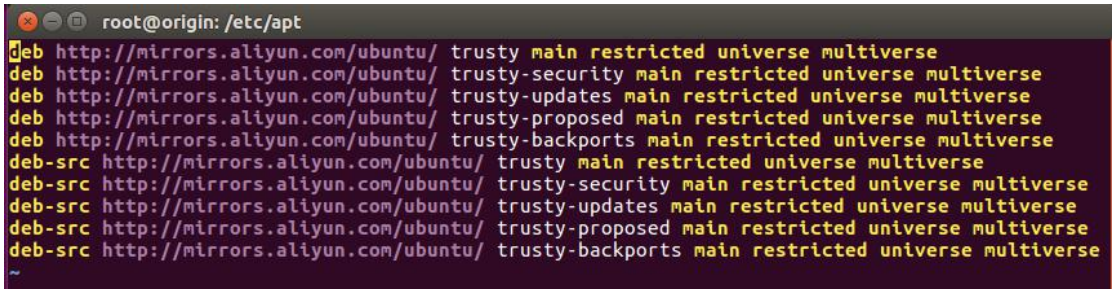
而修改过后的文件中第三行对应的内容为：

```
deb http://mirrors.aliyun.com/ubuntu/ trusty main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ trusty-security main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ trusty-updates main restricted universe
multiverse
```

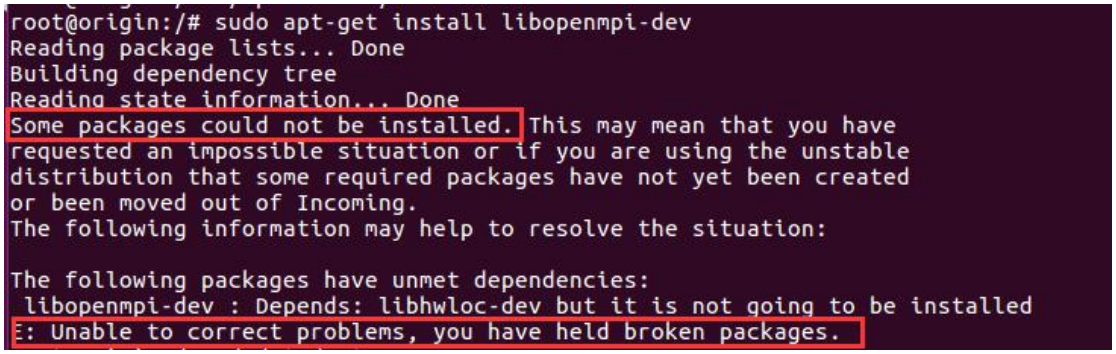
此时发现第三行的单个单词“multiverse”并非自然换行所形成，上一行（line 2）并没有达到满行顶格。而之所以会形成上述直接换行局面是因为本人在从 pdf 中直接全部选中复制换源文件到 sources.list 时没有考虑到格式问题，从而导致了换行错误：

```
deb http://mirrors.aliyun.com/ubuntu/ trusty main restricted universe multiverse
deb http://mirrors.aliyun.com/ubuntu/ trusty-security main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ trusty-updates main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ trusty-proposed main restricted universe
multiverse
deb http://mirrors.aliyun.com/ubuntu/ trusty-backports main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ trusty main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ trusty-security main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ trusty-updates main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ trusty-proposed main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ trusty-backports main restricted universe
multiverse
```

在 pdf 中 multiverse 单独成一行，直接复制过去后仍保持原来格式单独成一行，从而导致了“Malformed”的错误。手动修改换行格式成如下后即可成功运行：

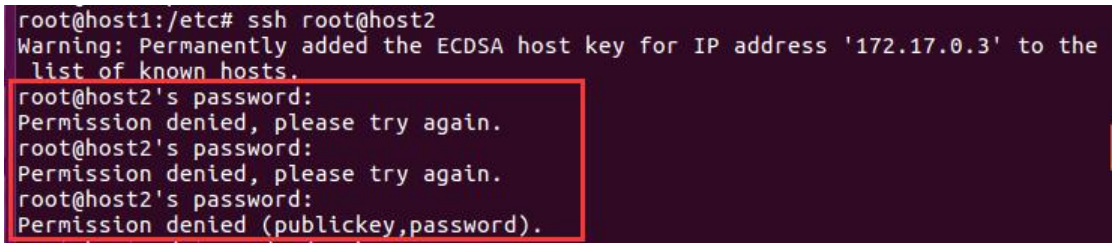


换源成功后，却在安装 MPI 时却出现了报错:



根据报错信息提示，疑似是源有问题导致文件下载失败。于是又试着换回原来的源并进行下载，结果可以成功进行。最后的结果是这一步换源其实并无必要。

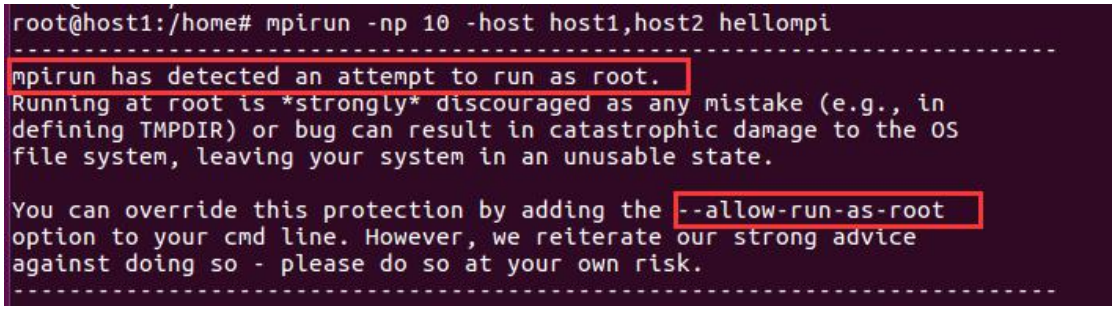
3.4 SSH 连接仍需密码



如上，在按照教程进行操作后在进行 ssh 连接时仍然需要密码。一步一步往回推，在查看 authorized_keys 文件和 id_rsa.pub 文件后发现了问题：

如上，发现原来是当时在复制文件内容的时候自作聪明，以为只要掐头去尾把中间一段字符串复制过去即可，结果导致错误。解决方法只需修改 authorized_keys 文件使其与 id_rsa.pub 文件完全一致即可，之后便可以进行免密连接。

3.5 程序运行



环境配置完成后先尝试用教材中的 hellompi 程序和指令测试一下，结果出现如上报错。由于是实验，且是干净容器，因此本人直接当了一回“root 敢死队”，直接加上--allow-run-as-root 运行，结果报出了另一错误：

```
root@host1:/home# mpirun --allow-run-as-root -np 10 -host host1,host2 hellompi
Unexpected end of /proc/mounts line `overlay / overlay rw,relatime,lowerdir=/var
/lib/docker/overlay2/l/TXR7CAP6YJACHTTBQPILJERSTJ:/var/lib/docker/overlay2/l/C25
5GPOH35KF6SH4ZDFGIR4AEW:/var/lib/docker/overlay2/l/TQSLVI45VS7SDGQONI3EKONA2R:/v
ar/lib/docker/overlay2/l/BZ2XRCOB76L7NXCZ4YXJSI4KAH:/var/lib/docker/overlay2/l/M
HJ7OWGTKBR7KJDVYS4IPE2M2J:/var/lib/docker/overlay2/l/RQ5T4UH755PLAL3GSSTHDKTSUI,
upperdir=/var/lib/docker/overlay2/81c1a0f0f3ef893e62d819d4a3f7ac62840d9c83b15888
a00ec9d2da3deb9f09/diff,workdir=/var/lib/docker/overlay2/81c1a0f0f3e'
Unexpected end of /proc/mounts line `overlay / overlay rw,relatime,lowerdir=/var
/lib/docker/overlay2/l/OFSZRVTEV6ETR3ZBSR4AEGT42T:/var/lib/docker/overlay2/l/C25
5GPOH35KF6SH4ZDFGIR4AEW:/var/lib/docker/overlay2/l/TQSLVI45VS7SDGQONI3EKONA2R:/v
ar/lib/docker/overlay2/l/BZ2XRCOB76L7NXCZ4YXJSI4KAH:/var/lib/docker/overlay2/l/M
HJ7OWGTKBR7KJDVYS4IPE2M2J:/var/lib/docker/overlay2/l/RQ5T4UH755PLAL3GSSTHDKTSUI,
upperdir=/var/lib/docker/overlay2/9bf71bca7fa08246e2c9f420a762548402780bbb49bf41
f2fccb7c41ac746805/diff,workdir=/var/lib/docker/overlay2/9bf71bca7fa'
--------------------------------------------------------------------------
There are not enough slots available in the system to satisfy the 10 slots
that were requested by the application:
  hellompi

Either request fewer slots for your application, or make more slots available
for use.
--------------------------------------------------------------------------
```

上网查阅相关资料后，得知此时需要新建一个 hostfile 文件为两个节点分别指定 slot 数量：

```
host2 slots=5
host2 slots=5
```

然后调用指令：

mpirun --allow-run-as-root -np 10 -host host1,host2 **--hostfile hostfile** hellompi

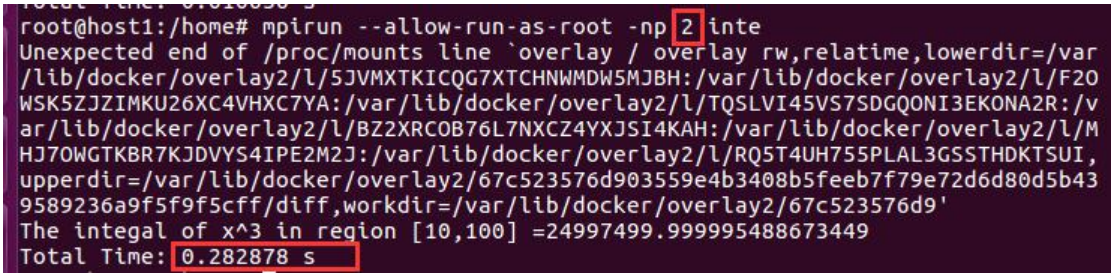
```
Hello from task 6 on host2!
Hello from task 9 on host2!
Hello from task 7 on host2!
Hello from task 5 on host2!
Hello from task 8 on host2!
Hello from task 0 on host1!
Hello from task 1 on host1!
MASTER: Number of MPI tasks is: 10
Hello from task 4 on host1!
Hello from task 2 on host1!
Hello from task 3 on host1!
```

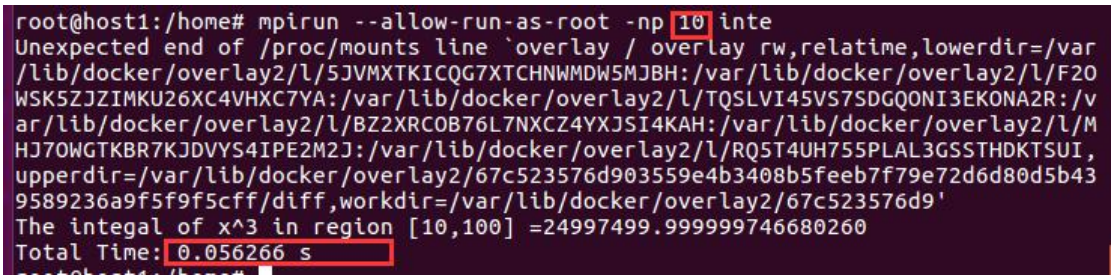
成功！

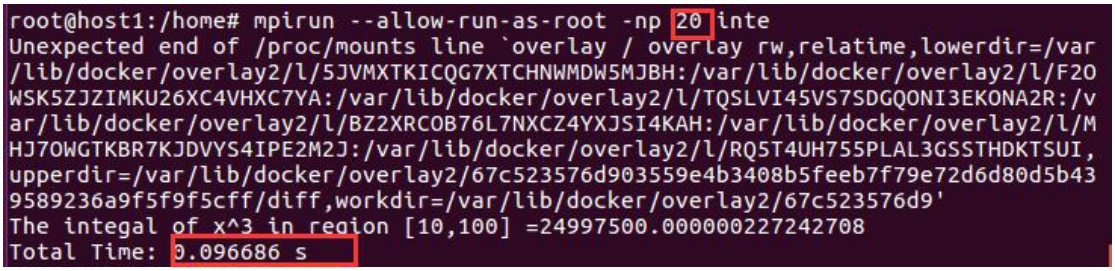
4.其它思考

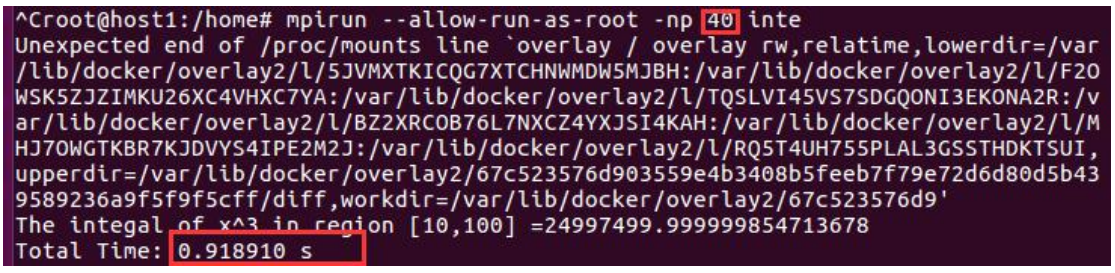
在本人此前的认识中，并行计算节点越多则速度应当越快。在此次实验中，本人采取了对比不同节点情况下运行时间的方法，得出以

下结果：









结果发现起初速度是随着节点的增加而加快，然而当节点数达到一定数量时，速度却反而随着节点的增加而减慢。在本人的理解中，此时由于节点过多而产生了反作用：节点间的通讯、连接等带来的冗余代价、负载超过了多出来的节点能够带来的计算优势。因此，在选择

节点数量时并非越多越好，而是应当适合计算量。

此外，在进行实验时，本人厘清了一些此前模糊的概念，比如多线程、多进程、多节点、多机、单机等。Host 数量=2 而 slot 数量=10 的那个案例就是一个很好的教学。在单机上可以把一个核分成多个线程来用，在多机时拥有多个核，每个核又可以再分为多个线程。而至于单机 10 线程和两机各 5 线程和 10 机各一线程，究竟哪个更好？此类问题仍然在我的心中保留着。此次实验由于时间紧迫、知识储备有限、设备问题较多等因素，这些问题未能来得及探索，故留作以后的继续探索！