实验一：MPI Programming

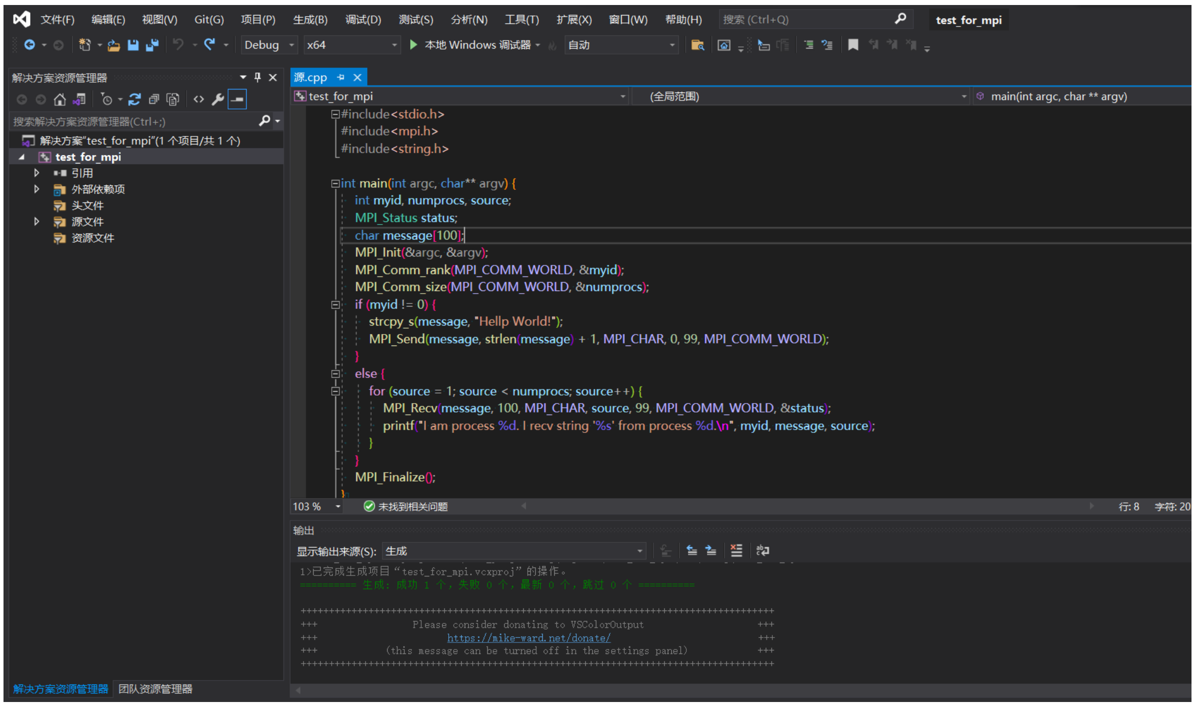
施宇 191250119

尝试在单机上安装并运行MPI环境。（MPICH或者OpenMPI等）

共安装了两套单机环境和一套多机环境：

单机环境：Visual Studio 2019 + MSMPI

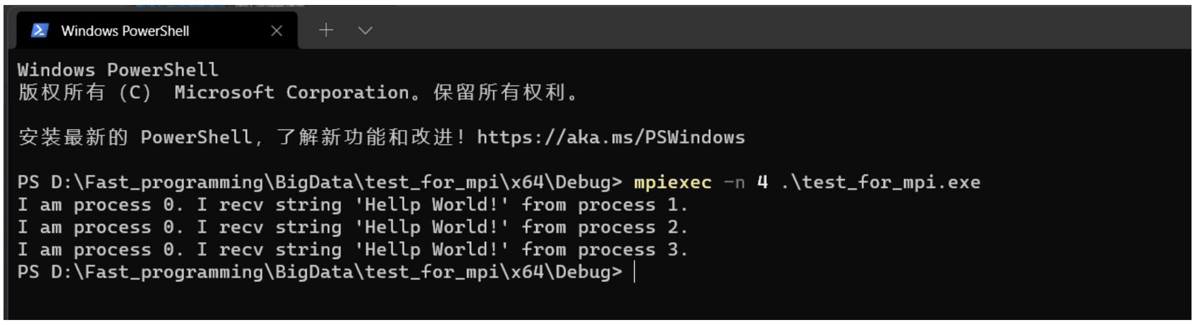
编译环境：



```
#include<stdio.h>
#include<mpi.h>
#include<string.h>

int main(int argc, char** argv) {
    int myid, numprocs, source;
    MPI_Status status;
    char message[100];
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    if (myid != 0) {
        strcpy_s(message, "Hellp World!");
        MPI_Send(message, strlen(message) + 1, MPI_CHAR, 0, 99, MPI_COMM_WORLD);
    }
    else {
        for (source = 1; source < numprocs; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source, 99, MPI_COMM_WORLD, &status);
            printf("I am process %d. I recv string '%s' from process %d.\n", myid, message, source);
        }
    }
    MPI_Finalize();
}
```

```
1>已完成生成项目“test_for_mpi.vcxproj”的操作。
========== 生成: 成功 1 个，失败 0 个，最新 0 个，跳过 0 个 ==========

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++                Please consider donating to VSColorOutput               +++
+++                     https://mike-ward.net/donate/                      +++
+++            (this message can be turned off in the settings panel)      +++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

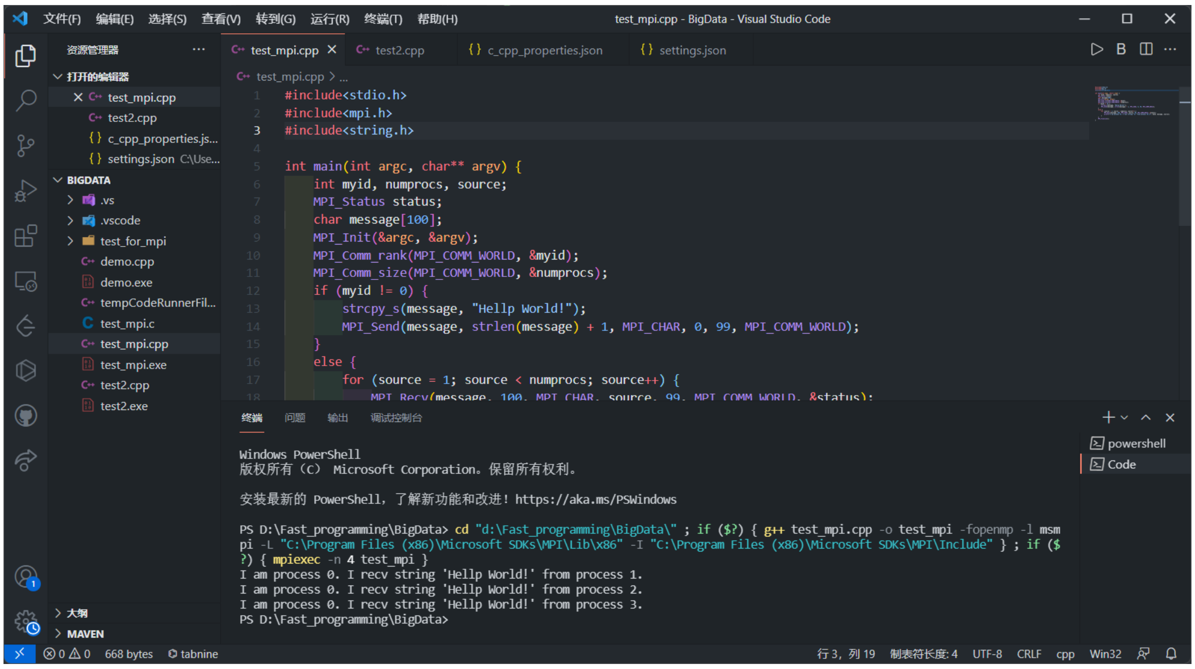
运行环境：



```
Windows PowerShell
版权所有 (C)  Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS D:\Fast_programming\BigData\test_for_mpi\x64\Debug> mpiexec -n 4 .\test_for_mpi.exe
I am process 0. I recv string 'Hellp World!' from process 1.
I am process 0. I recv string 'Hellp World!' from process 2.
I am process 0. I recv string 'Hellp World!' from process 3.
PS D:\Fast_programming\BigData\test_for_mpi\x64\Debug> 
```

单机环境：Vscode + MSMPI

编译运行：



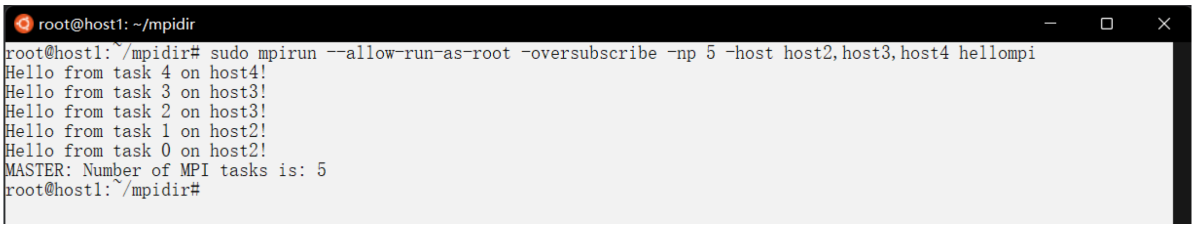
```cpp
#include<stdio.h>
#include<mpi.h>
#include<string.h>

int main(int argc, char** argv) {
    int myid, numprocs, source;
    MPI_Status status;
    char message[100];
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &myid);
    MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
    if (myid != 0) {
        strcpy_s(message, "Hellp World!");
        MPI_Send(message, strlen(message) + 1, MPI_CHAR, 0, 99, MPI_COMM_WORLD);
    }
    else {
        for (source = 1; source < numprocs; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source, 99, MPI_COMM_WORLD, &status);
```

```
Windows PowerShell
版权所有 (C)  Microsoft Corporation。保留所有权利。

安装最新的 PowerShell，了解新功能和改进！https://aka.ms/PSWindows

PS D:\Fast_programming\BigData> cd "d:\Fast_programming\BigData\" ; if ($?) { g++ test_mpi.cpp -o test_mpi -fopenmp -l msm
pi -L "C:\Program Files (x86)\Microsoft SDKs\MPI\Lib\x86" -I "C:\Program Files (x86)\Microsoft SDKs\MPI\Include" } ; if ($
?) { mpiexec -n 4 test_mpi }
I am process 0. I recv string 'Hellp World!' from process 1.
I am process 0. I recv string 'Hellp World!' from process 2.
I am process 0. I recv string 'Hellp World!' from process 3.
PS D:\Fast_programming\BigData>
```

多机环境：Windows11家庭版 + WSL2 + Docker

详细环境配置过程见**附录A**

在三个节点上运行hellompi程序：



```
root@host1:~/mpidir# sudo mpirun --allow-run-as-root -oversubscribe -np 5 -host host2,host3,host4 hellompi
Hello from task 4 on host4!
Hello from task 3 on host3!
Hello from task 2 on host3!
Hello from task 1 on host2!
Hello from task 0 on host2!
MASTER: Number of MPI tasks is: 5
root@host1:~/mpidir#
```

当并行数为5时，三个节点都参与工作，且任务数量按顺序平均分配。

编程1

编程2

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。

附录A

多机环境(Windows11家庭版 + WSL2 + Docker)配置：

安装Hyper-V

由于WSL2需要Hyper-V组件提供虚拟机支持，而该组件只有Windows专业版上才有，因此在Windows家庭版上需要手动下载安装该组件。

将下面的内容添加到记事本中：

```
pushd "%~dp0"

dir /b %SystemRoot%\servicing\Packages\*Hyper-V*.mum >hyper-v.txt

for /f %%i in ('findstr /i . hyper-v.txt 2^>nul') do dism /online /norestart
/add-package:"%SystemRoot%\servicing\Packages\%%i"

del hyper-v.txt

Dism /online /enable-feature /featurename:Microsoft-Hyper-V-All /LimitAccess
/ALL
```

然后另存为Hyper-V.cmd文件，右键该文件以管理员身份执行，待下载结束后输入Y重启，完成Hyper-V的安装。

安装WSL2

下载Linux内核更新包：

https://links.jianshu.com/go?to=https%3A%2F%2Fwslstorestorage.blob.core.windows.net%2Fwslblob%2Fwsl_update_x64.msi

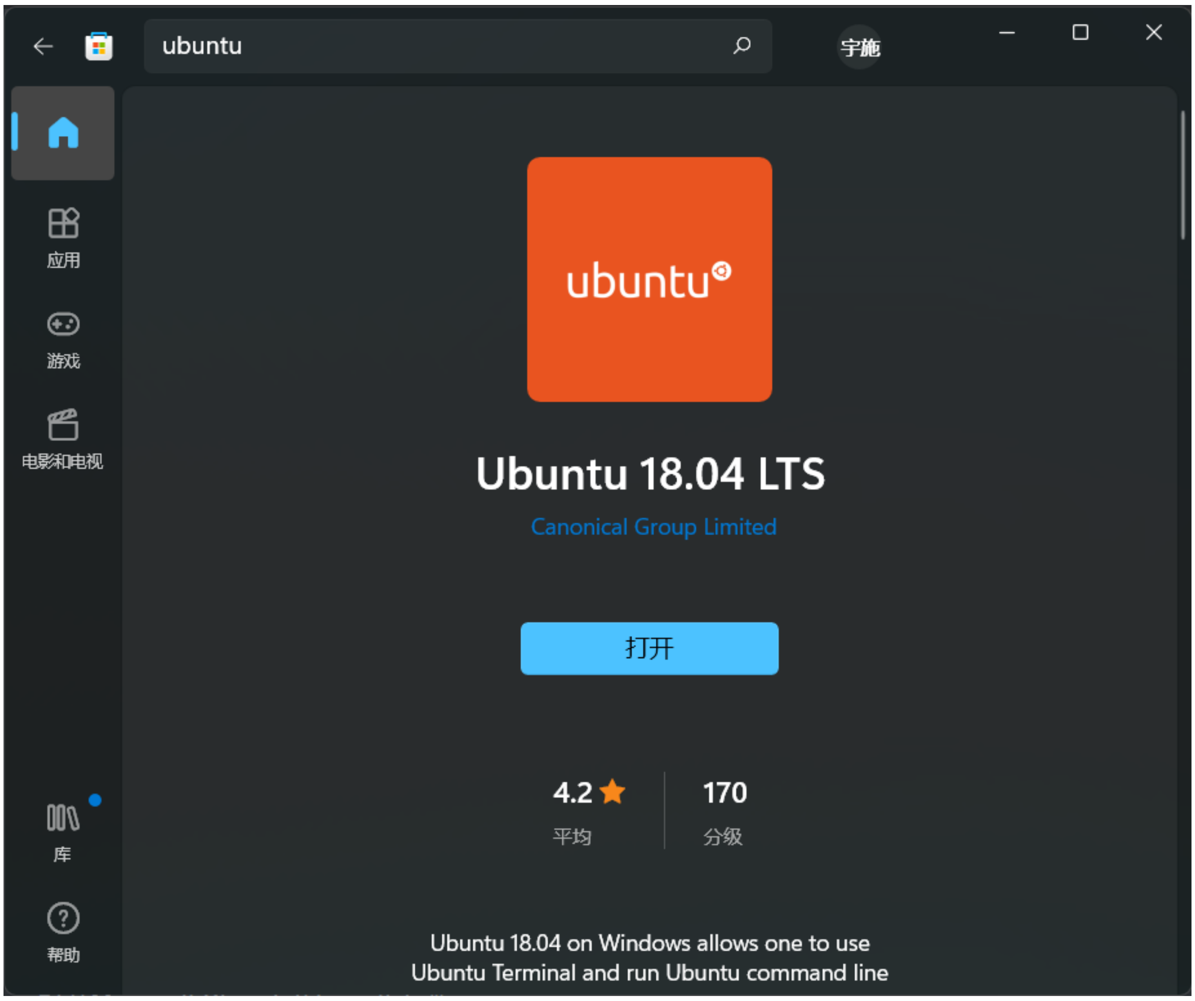
运行下载的更新包。（双击以运行 - 系统将提示提供提升的权限，选择“yes”以批准此安装。）

打开终端输入下方命令，将WSL默认版本设置为WSL2

```
wsl --set-default-version 2
```

打开Microsoft Store下载Linux发行版：

这里我选择的是Ubuntu18.04：



Ubuntu 18.04 on Windows allows one to use
Ubuntu Terminal and run Ubuntu command line

安装完成后在菜单栏找到点击运行，输入用户名密码后即可进入Ubuntu终端。

在Windows终端中输入`wsl -l -v`检查是否是WSL2版本。

安装Docker

配置阿里云镜像，`/etc/apt/sources.list`文件的内容替换如下：

```
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe
multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted
universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe
multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted
universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe
multiverse
```

```
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted
universe multiverse
```

更新源：

```
sudo apt-get update
```

安装Docker：

```
sudo apt-get remove docker docker-engine docker.io
sudo apt-get update
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo apt-key fingerprint 0EBFCD88
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
sudo apt-get update
sudo apt-get install docker-ce
```

测试docker是否安装成功：

```
docker version
```

添加用户：

```
sudo adduser $USER docker
```

启动docker服务：

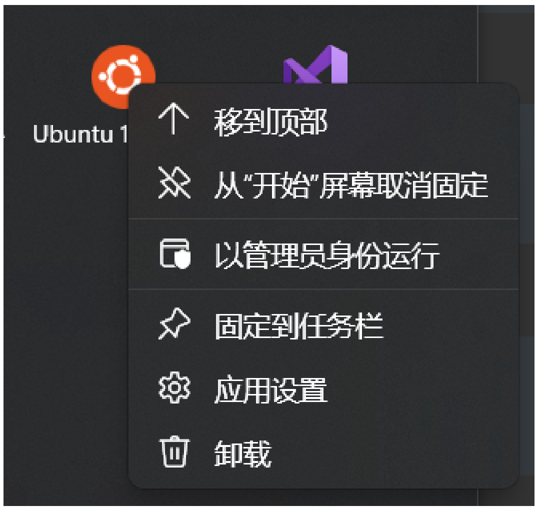
```
sudo service docker restart
```

测试docker能否正常运行：

```
sudo docker run hello-world
```

如果提示：Cannot connect to the Docker daemon at ......则还需要以下操作：

关闭wsl窗口，重新右键以管理员身份运行：



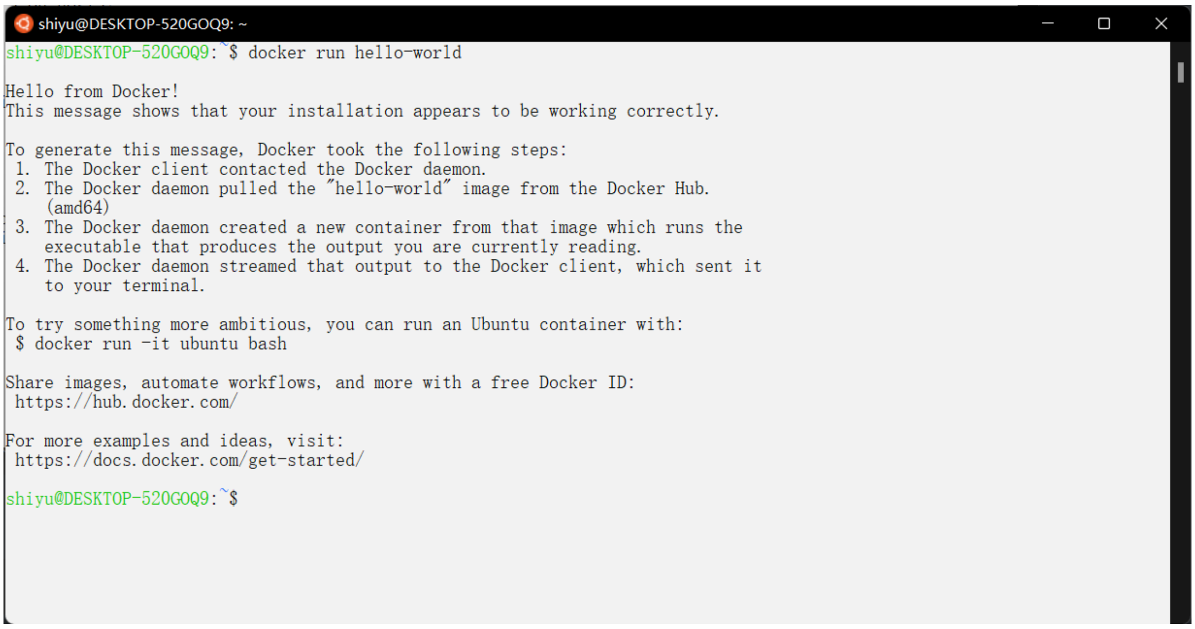
然后执行以下命令：

```
sudo apt-get install cgroupfs-mount
sudo cgroupfs-mount
sudo service docker restart
```

再次运行测试命令：

```
sudo docker run hello-world
```

运行成功：



```
shiyu@DESKTOP-520GOQ9: ~
shiyu@DESKTOP-520GOQ9:~$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

shiyu@DESKTOP-520GOQ9:~$
```

多机MPI配置

这里使用ubuntu:18.04作为节点镜像：

```
docker pull ubuntu:18.04
```

拉取镜像之后，需要对基础镜像进行环境的配置，在此之前先进行换源

开启一个容器:

```
docker run -dit --hostname origin --name origin ubuntu:18.04
```

进入容器：

```
docker exec -it origin bash
```

安装一下sudo和vim：

```
apt-get install sudo
sudo apt-get install vim
```

切换阿里云的源：

```
sudo cp /etc/apt/sources.list /etc/apt/sources.list.bak
sudo vim /etc/apt/sources.list
```

替换为以下内容：

```
deb http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic main restricted universe
multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-security main restricted
universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-updates main restricted universe
multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-proposed main restricted
universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted universe
multiverse
deb-src http://mirrors.aliyun.com/ubuntu/ bionic-backports main restricted
universe multiverse
```

更新源：

```
sudo apt-get update
```

安装MPI：

```
sudo apt-get install libopenmpi-dev
sudo apt-get install openmpi-bin
```

检查是否安装成功：

```
mpirun --version
```

安装ssh：

```
sudo apt-get install openssh-server
```

尝试ssh一下自己，应该会要求输密码，三次回车即可（拒绝访问）：

```
ssh localhost
```

到~/目录下，检查是否有.ssh文件夹：

```
ll
```

进入该文件夹，然后应该能找到authorized_keys文件。

修改authorized_keys文件权限为700：

```
sudo chmod 700 authorized_keys
```

修改.ssh目录权限为600：

```
cd ..
sudo chmod 600 .ssh
```

Ctrl+Q+P退出当前镜像。

创建新镜像：

```
docker commit -a <作者名字> origin mpi-ubuntu:latest
```

罗列出本地镜像，看是否创建成功:

```
docker images
```

通过挂在本地文件夹的方式运行容器（这里以四个为例）：

```
docker run -dit --hostname host1 --name mpi-host1 -v ~/mpidir:/root/mpidir mpi-
ubuntu:latest
docker run -dit --hostname host2 --name mpi-host2 -v ~/mpidir:/root/mpidir mpi-
ubuntu:latest
docker run -dit --hostname host3 --name mpi-host3 -v ~/mpidir:/root/mpidir mpi-
ubuntu:latest
docker run -dit --hostname host4 --name mpi-host4 -v ~/mpidir:/root/mpidir mpi-
ubuntu:latest
```

进入host1：

```
docker exec -it mpi-host1 bash
```

启动ssh服务：

```
service ssh start
```

打开hosts文件：

```
sudo vim /etc/hosts
```

可以看到文本中最后一行为：

```
172.17.0.2      host1
```

前面是ip地址（也是本容器的地址），后面是主机名（可以看作是ip的别名，可以用它来代替ip地址的书写）

在后面添加host2，host3和host4的ip地址和主机名：

```
172.17.0.3      host2
172.17.0.4      host3
172.17.0.5      host4
```

到~/.ssh目录中，在当前目录下生成密钥（输完命令后直接三次回车，不要做其他操作）：

```
ssh-keygen -t rsa
```

然后会发现生成了id_rsa.pub文件，将该文件复制到挂载目录~/mpidir中：

```
cp id_rsa.pub ~/mpidir
```

退出当前脚本镜像，分别进入host2， host3， host4，进行如下操作：

启动ssh服务：

```
service ssh start
```

进入~/mpidir目录，执行下方命令，将host1的公钥加入到其他子节点的authorized_keys中，设置免密ssh登录：

```
cat id_rsa.pub >> ~/.ssh/authorized_keys
```

退出当前镜像。

全部执行完成后，进入host1：

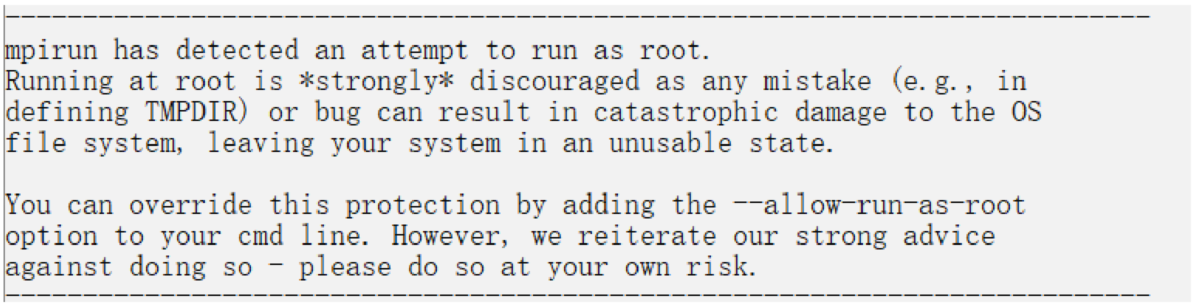
依次检查能够免密ssh连接host2，host3和host4（连接后通过Ctrl+D退出）：

```
ssh root@host2
ssh root@host3
ssh root@host4
```

如果都能成功连接，则测试hellompi程序运行：

```
mpicc hellompi.c -o hellompi
mpirun -np 10 -host host2,host3,host4 hellompi
```

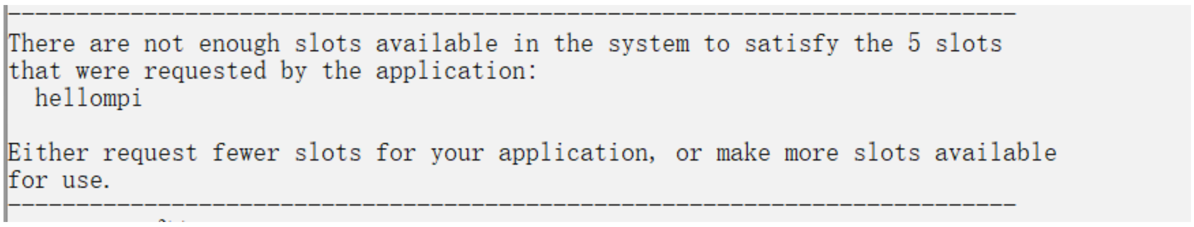
如果报错：



```
--------------------------------------------------------------------------
mpirun has detected an attempt to run as root.
Running at root is *strongly* discouraged as any mistake (e.g., in
defining TMPDIR) or bug can result in catastrophic damage to the OS
file system, leaving your system in an unusable state.

You can override this protection by adding the --allow-run-as-root
option to your cmd line. However, we reiterate our strong advice
against doing so - please do so at your own risk.
--------------------------------------------------------------------------
```

加上参数`--allow-run-as-root`。

如果报错：

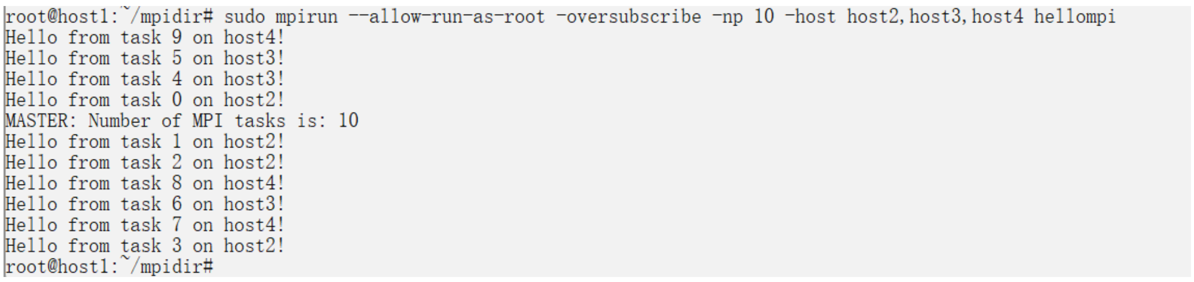


```
--------------------------------------------------------------------------
There are not enough slots available in the system to satisfy the 5 slots
that were requested by the application:
  hellompi

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

则说明mpi自动检测判断当前cpu不适合以给定的并行数执行，这里选择添加参数`-oversubscribe`来强制执行。最终的执行命令如下：

```
sudo mpirun --allow-run-as-root -oversubscribe -np 5 -host host2,host3,host4
hellompi
```

执行成功：



```
root@host1:~/mpidir# sudo mpirun --allow-run-as-root -oversubscribe -np 10 -host host2,host3,host4 hellompi
Hello from task 9 on host4!
Hello from task 5 on host3!
Hello from task 4 on host3!
Hello from task 0 on host2!
MASTER: Number of MPI tasks is: 10
Hello from task 1 on host2!
Hello from task 2 on host2!
Hello from task 8 on host4!
Hello from task 6 on host3!
Hello from task 7 on host4!
Hello from task 3 on host2!
root@host1:~/mpidir#
```

环境配置结束。