# 《并行计算》实验指导书

# 实验1：基于华为云的并行环境搭建和使用

## 实验1.1华为云实验环境说明

### 实验目的

1．了解华为云环境的使用过程；

2．掌握利用云环境搭建小型集群环境的过程；

### 实验内容

实验步骤：

步骤一：登录华为云。

打开浏览器，输入华为云的域名：<https://www.huaweicloud.com>，点击右上角登录按钮，输入用户名和密码。



步骤二：购买弹性云服务器

选择产品——>计算——>弹性云服务器ECS——>购买





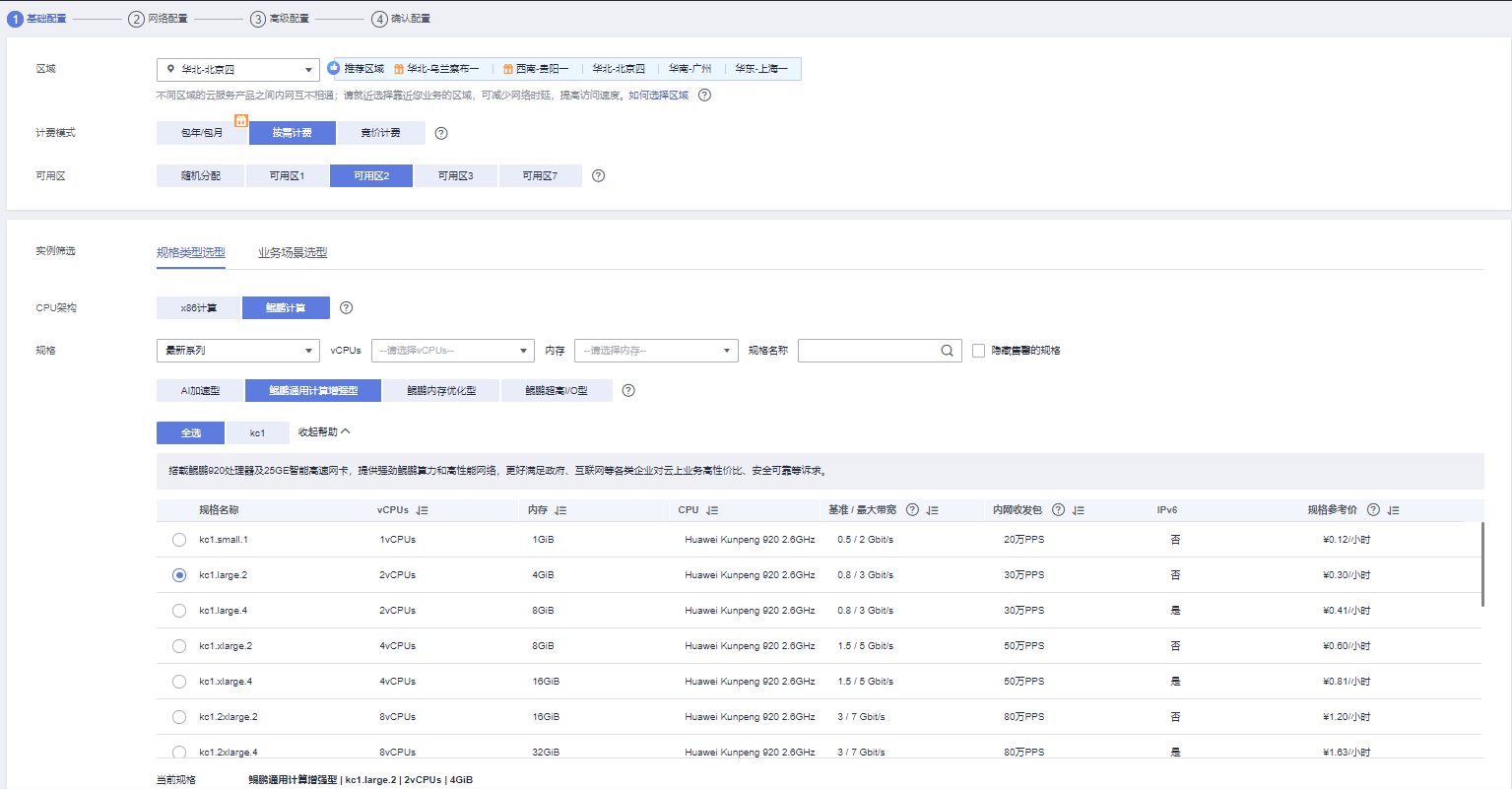
步骤三：基础配置

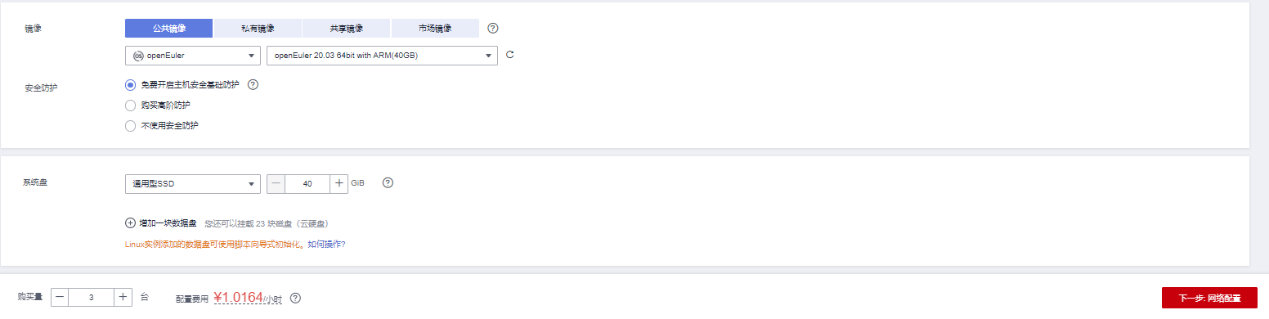
按照下表进行购买，一共购买3台。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **计费模式** | **区域** | **CPU架构** | **规格** | **镜像** | **系统盘** |
| 按需计费 | 华北-北京四 | 鲲鹏计算 | kc1.large.2 | 公共镜像：openEuler 20.03 | 至少40GB |

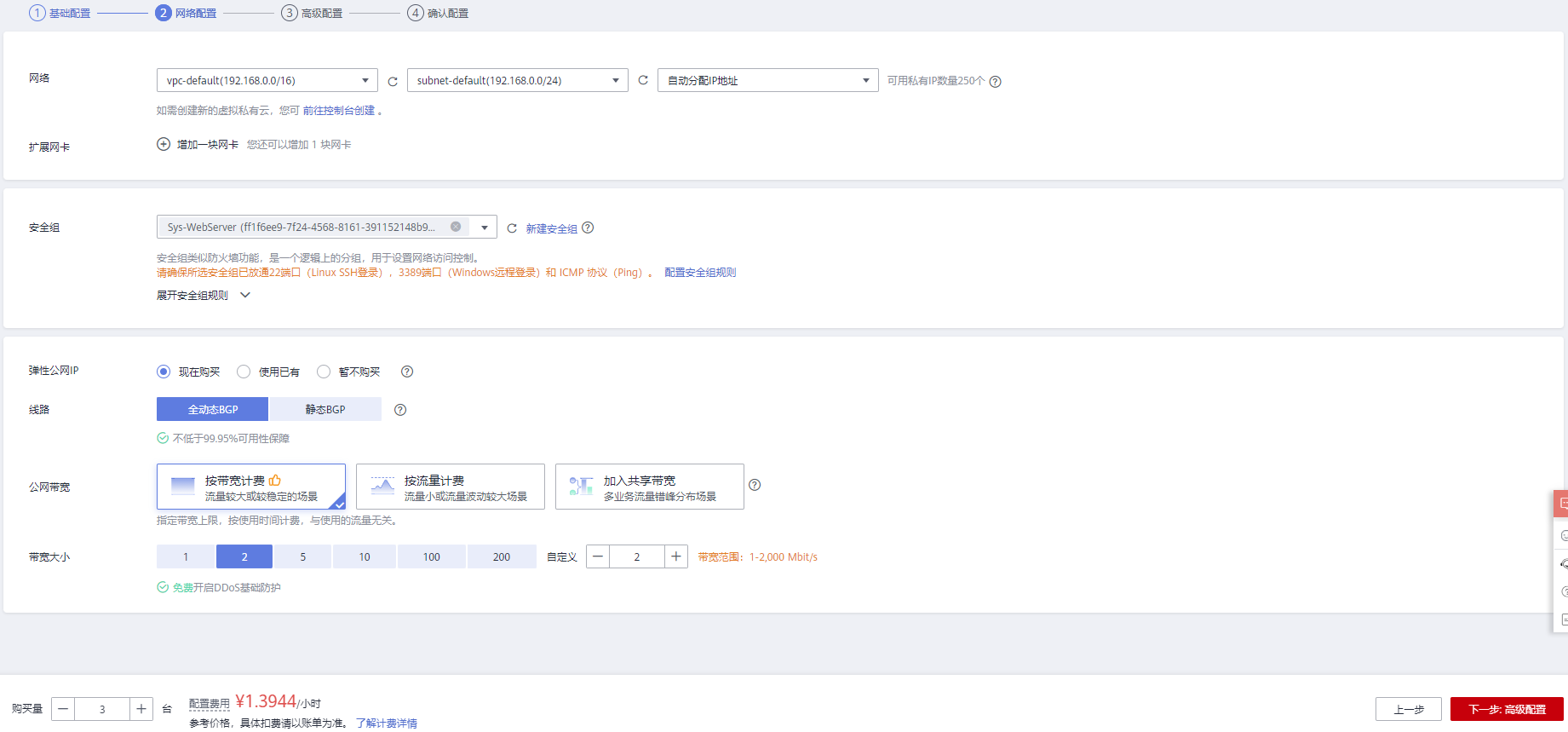
注意：是openEuler，不是其他镜像，否则安装软件包会有问题。

参考截图：



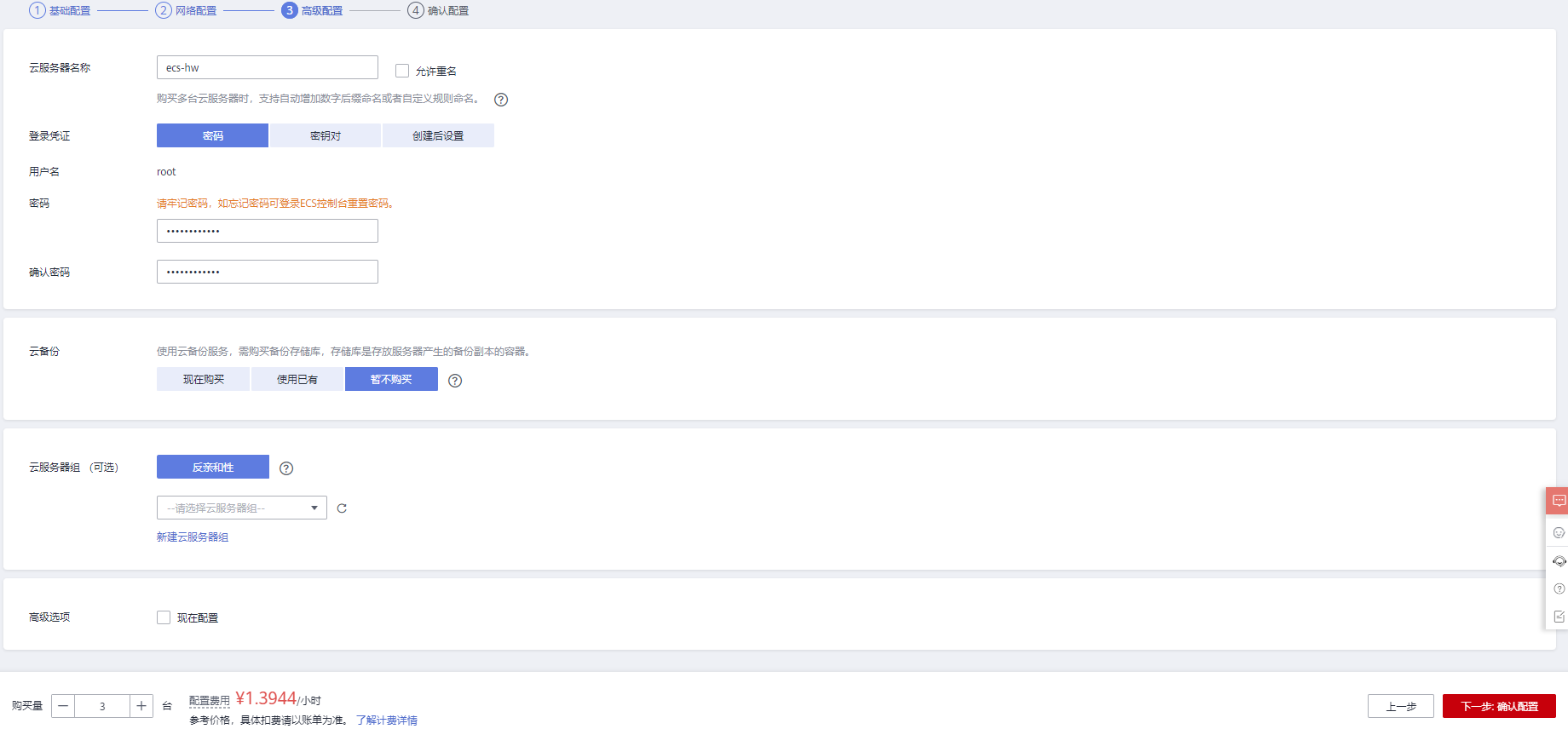


步骤四：网络配置



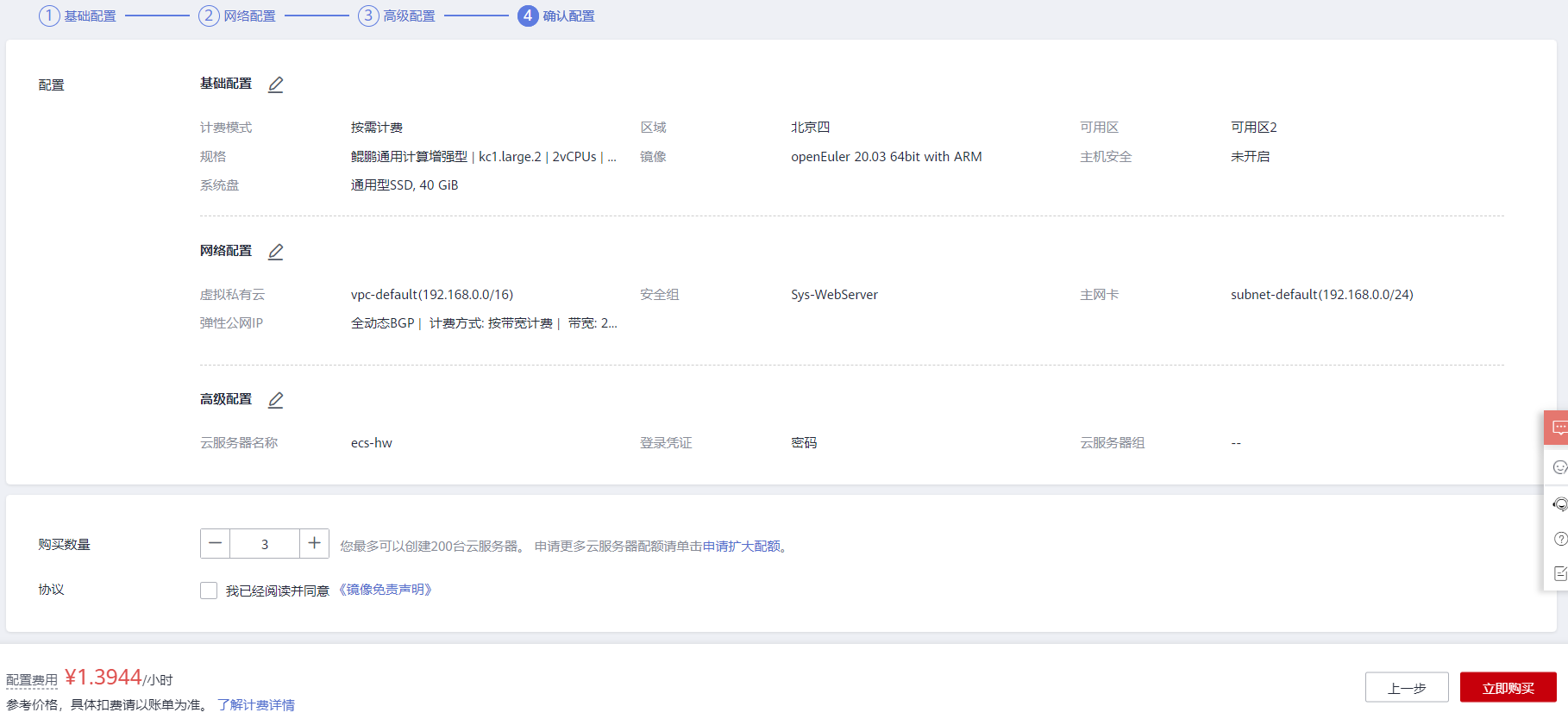
步骤五：高级配置

设置云服务器名称（eg：ecs-hw）、密码（Parallel2022），云备份选择暂不购买。



步骤六：确认配置

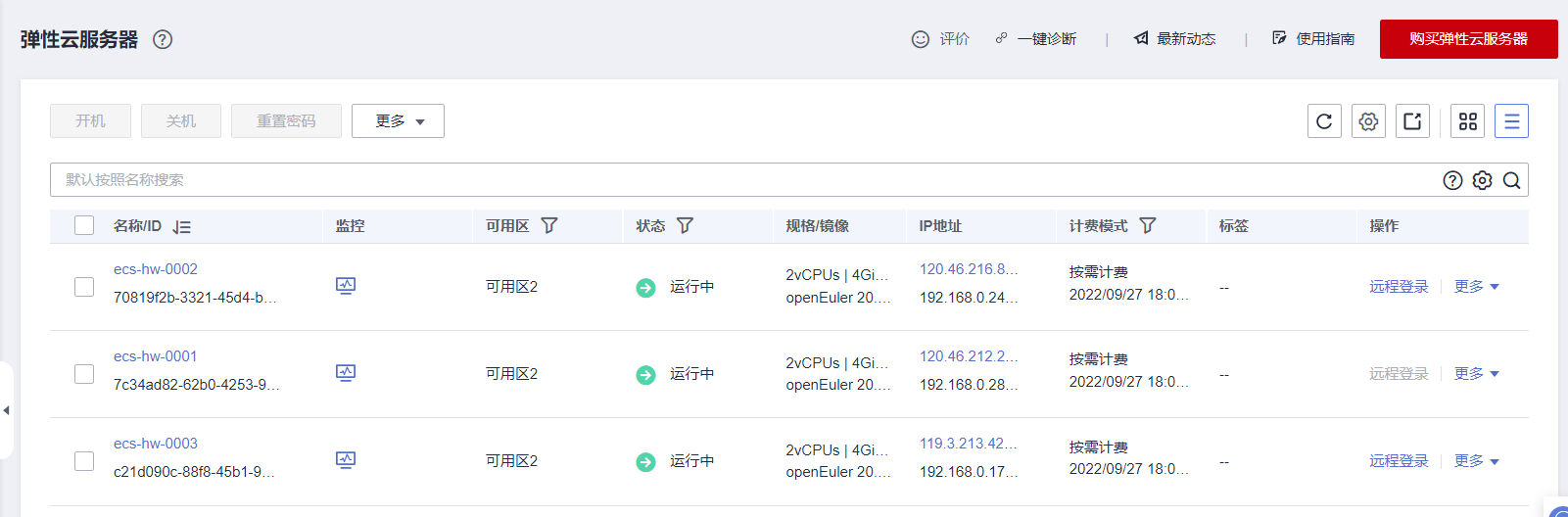
勾选我已阅读并同意《华为镜像免责声明》，点击确认配置



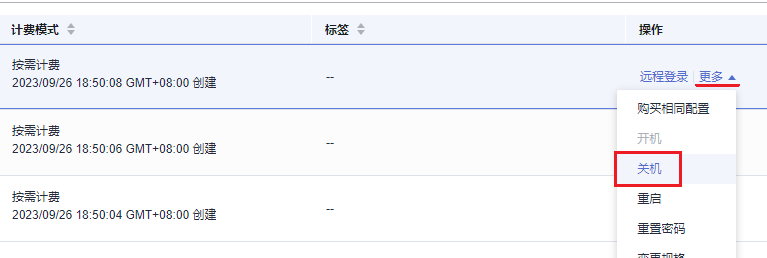
步骤七：打开控制台





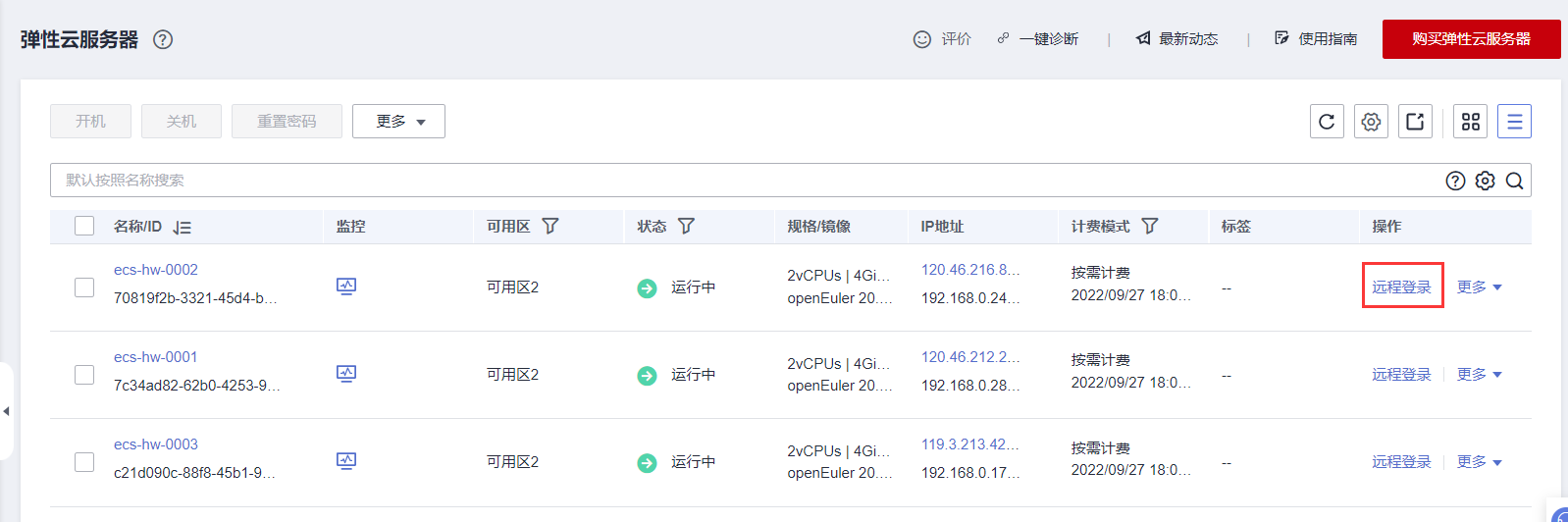


注意：实验结束后，要关闭虚拟机，以免继续扣费。



步骤八：远程登录ECS

点击远程登录，选择CloudShell登录。或者系统中安装SSH工具，如XShell，输入公网IP、用户名和密码，即可登录。





步骤九：环境配置说明

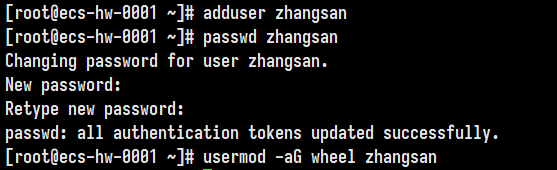
为了防⽌⼤家的⽂件混乱，建议⼤家在每⼀台机器下都建⽴个⼈账户，不建议统⼀使⽤root账户，下⾯以⽤户名zhangsan 为例，三台机器名分别为ecs-hw-0001, ecs-hw-0002, ecs-hw-0003，三台机器的私网IP为192.168.0.28, 192.168.0.247, 192.168.0.17，以下所有的操作均在ecs-hw-0001主机上，环境配置时在每⼀台主机上都需要重复执行（步骤十和步骤十一，三台机器都需要执行，IP需要根据实际分配进行相应调整）。

步骤十：创建用户

每台主机都需要在root用户下建立个人的账户，命令如下：

|  |
| --- |
| adduser zhangsan  passwd zhangsan  usermod -aG wheel zhangsan |

说明：zhangsan密码也设置为Parallel2022



步骤十一：免密配置

以下所有步骤在三台机器上均需要重复执行。

1. 配置三台机器主机名和IP解析（各主机IP可以通过ifconfig或者控制台界面查看）

|  |
| --- |
| vim /etc/hosts |

注释文件原先本身的信息并添加以下信息（注释的目的是因为会对本实验程序的运行产生报错）：

|  |
| --- |
| 192.168.0.28 ecs-hw-0001  192.168.0.247 ecs-hw-0002  192.168.0.17 ecs-hw-0003 |

添加完后，信息如下：



1. 登录新账户

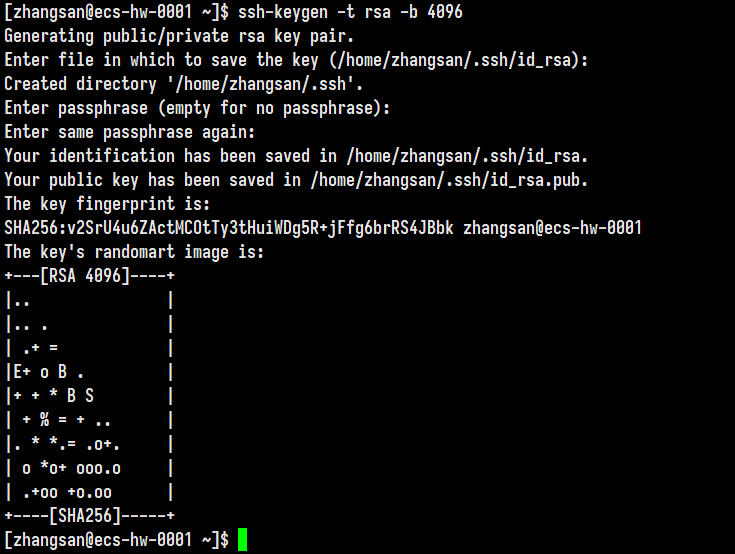
首先退出root账户，重新登录到新建立的账户下：

|  |
| --- |
| su - zhangsan |

1. 生成本地密钥

|  |
| --- |
| ssh-keygen -t rsa -b 4096 |

说明：会让输入信息，保持默认，回车即可。



1. 添加公钥至所有主机

|  |
| --- |
| ssh-copy-id zhangsan@ecs-hw-0001  ssh-copy-id zhangsan@ecs-hw-0002  ssh-copy-id zhangsan@ecs-hw-0003 |

1. 安装依赖包

|  |
| --- |
| sudo yum -y install gcc-gfortran --nogpgcheck  sudo yum -y install mpich mpich-devel mpich-doc --nogpgcheck |

1. 配置环境变量

|  |
| --- |
| vim ~/.bashrc |

在.bashrc文件后面加上

MPI\_ROOT=/usr/lib64/mpich

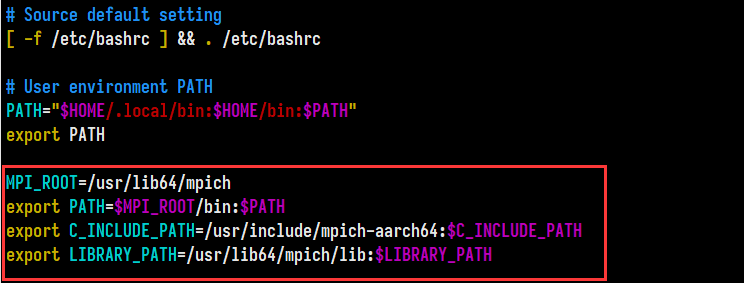
export PATH=$MPI\_ROOT/bin:$PATH

export C\_INCLUDE\_PATH=/usr/include/mpich-aarch64:$C\_INCLUDE\_PATH

export LIBRARY\_PATH=/usr/lib64/mpich/lib:$LIBRARY\_PATH

并保存。

文件截图如下：



执行source命令，使之生效。

|  |
| --- |
| source ~/.bashrc |

## 实验1.2 并行环境下程序的编译和运行

### 实验目的

通过在华为鲲鹏上编译运行并行程序，掌握集群环境并行计算的配置以及加深对并行计算的了解；

### 实验内容

实验步骤：

以下步骤均在ecs-hw-0001上zhangsan用户下执行。

步骤一：创建程序源码

执行以下命令，创建hello目录存放该程序的所有文件, 并进入hello目录。

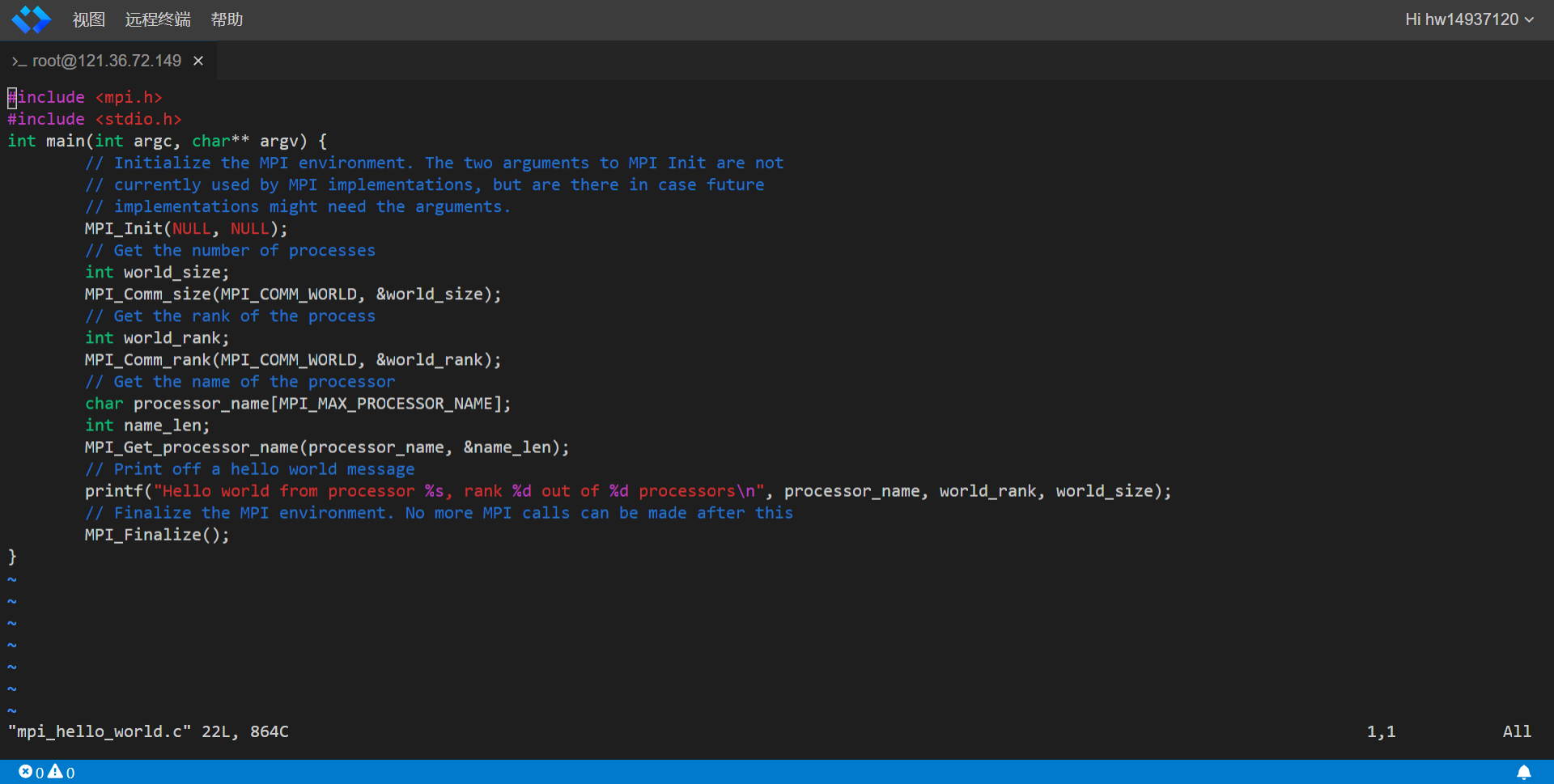
|  |
| --- |
| mkdir /home/zhangsan/hello  cd /home/zhangsan/hello |

执行以下命令，创建示例程序源码mpi\_hello\_world.c

|  |
| --- |
| vim mpi\_hello\_world.c |

代码内容如下：

|  |
| --- |
| #include <mpi.h>  #include <stdio.h>  int main(int argc, char\*\* argv) {  // Initialize the MPI environment. The two arguments to MPI Init are not  // currently used by MPI implementations, but are there in case future  // implementations might need the arguments.  MPI\_Init(NULL, NULL);  // Get the number of processes  int world\_size;  MPI\_Comm\_size(MPI\_COMM\_WORLD, &world\_size);  // Get the rank of the process  int world\_rank;  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &world\_rank);  // Get the name of the processor  char processor\_name[MPI\_MAX\_PROCESSOR\_NAME];  int name\_len;  MPI\_Get\_processor\_name(processor\_name, &name\_len);  // Print off a hello world message  printf("Hello world from processor %s, rank %d out of %d processors\n", processor\_name, world\_rank, world\_size);  // Finalize the MPI environment. No more MPI calls can be made after this  MPI\_Finalize();  } |



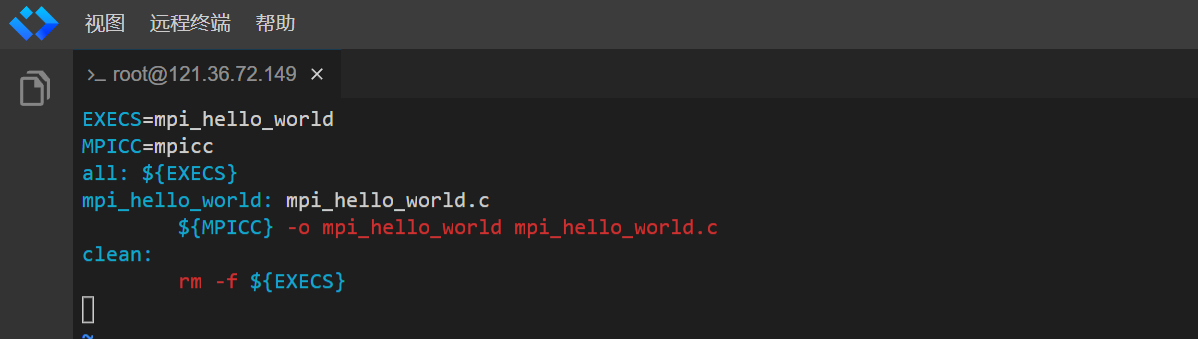
步骤二：创建makefile（也可以不创建，直接利用mpicc进行编译）

执行以下命令，创建makefile

|  |
| --- |
| vim makefile |

代码内容如下：

|  |
| --- |
| EXECS=mpi\_hello\_world  MPICC=mpicc  all: ${EXECS}  mpi\_hello\_world: mpi\_hello\_world.c  ${MPICC} -o mpi\_hello\_world mpi\_hello\_world.c  clean:  rm -f ${EXECS} |



可以直接使用mpicc命令进行编译，编译命令如下：

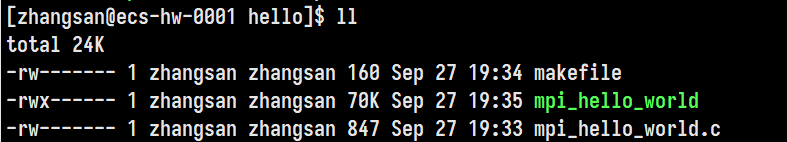
|  |
| --- |
| mpicc mpi\_hello\_world.c -o mpi\_hello\_world |

步骤三：源码编译

执行以下命令，进行编译

|  |
| --- |
| cd /home/zhangsan/hello  make |

编译的文件列表如下：



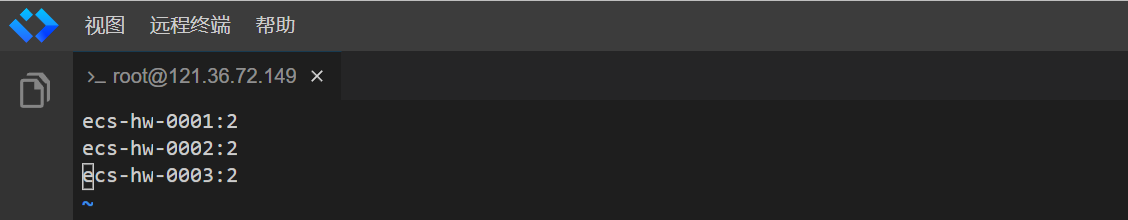
步骤四：建立主机配置文件

执行以下命令，建立主机配置文件

|  |
| --- |
| vim /home/zhangsan/hello/config |

添加内容如下（格式说明，主机名:进程数）：

|  |
| --- |
| ecs-hw-0001:2  ecs-hw-0002:2  ecs-hw-0003:2 |



步骤五：将目录拷贝到其他节点

|  |
| --- |
| scp -r /home/zhangsan/hello/ zhangsan@ecs-hw-0002:/home/zhangsan/  scp -r /home/zhangsan/hello/ zhangsan@ecs-hw-0003:/home/zhangsan/ |

步骤六：运行监测

执行以下命令，查看运行结果（只需要在ecs-hw-0001上执行）

|  |
| --- |
| mpirun -np 6 -f /home/zhangsan/hello/config /home/zhangsan/hello/mpi\_hello\_world |

**要求：**查看结果，并截图保存。

通过上述代码运行，可以看出，编写的hello-wolrd程序已经在华为鲲鹏上运行起来，程序在集群之间并行计算处理。

## 实验1.3 PI程序的编译和运行

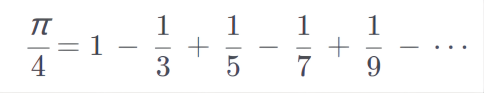
### 实验目的

集群环境中并行程序编译实战，了解在不同进程下程序的性能变化，体会并行计算的优势。

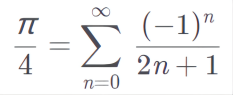
### 实验内容

程序原理介绍：

在数学领域，π的莱布尼茨公式说明



右边的展式是一个无穷级数，被称为莱布尼茨级数，这个级数收敛到π/4。它通常也被称为格雷戈里-莱布尼茨级数，用以纪念莱布尼茨同时代的天文学家兼数学家詹姆斯·格雷戈里。使用求和符号可记作：



利用该公式，可以对π进行求解，项数越多，π值的求解越精确。

串行代码（pi-c.c）如下：

|  |
| --- |
| /\* File: pi-c.c  \*  \* Purpose: Estimates pi using the Leibniz formula (in C).  \*  \*  \* Compile: gcc -o pi-c pi-c.c -lm  \* Run: ./pi-c  \*/  #include <stdio.h>  #include <math.h>  #include <time.h>  /\* We define pi here so we can check and see how accurate our computation is. \*/  #define PI 3.141592653589793238462643  int main(int argc, char \*\*argv) {  int intervals;  printf("Number of intervals: ");  fflush(stdout);  scanf("%d", &intervals);  clock\_t time1 = clock();  int i;  double sum, total = 0;  for (i = 0; i < intervals; ++i) {  total += pow(-1, i) / (2 \* i + 1);  }  clock\_t time2 = clock();  total = total \* 4;  printf("Result: %.10lf\n", total);  printf("Accuracy: %.10lf\n", PI - total);  printf("Time: %.10lf\n", (double)(time2 - time1)/CLOCKS\_PER\_SEC);  } |

并行代码（MPI实现，pi-mpi.c）如下：

|  |
| --- |
| /\* File: pi-mpi.c  \*  \* Purpose: Estimates pi using the Leibniz formula parallelized with MPI  \*  \* Compile: mpicc -o pi-mpi pi-mpi.c -lm  \* Run: mpirun -np <number of processes> pi-mpi  \*/  #include <stdio.h>  #include <math.h>  #include <mpi.h>  /\* We define pi here so we can check and see how accurate our computation is. \*/  #define PI 3.141592653589793238462643  int main(int argc, char \*\*argv) {  MPI\_Init(&argc, &argv);  int processes, pe;  MPI\_Comm\_size(MPI\_COMM\_WORLD, &processes);  MPI\_Comm\_rank(MPI\_COMM\_WORLD, &pe);  /\* Let's prompt for the number of intervals. We'll broadcast whatever  \* process 0 reads to the other processes. We could use command line  \* arguments instead, but then there'd be no reason to broadcast! \*/  int intervals;  if (pe == 0) {  printf("Number of intervals: ");  fflush(stdout);  scanf("%d", &intervals);  }  double time1 = MPI\_Wtime();  MPI\_Bcast(&intervals, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);  int count = intervals / processes;  int start = count \* pe;  int end = count \* pe + count;  int i;  double subtotal, total = 0;  for (i = start; i < end; ++i) {  subtotal += pow(-1, i) / (2 \* i + 1);  }  MPI\_Reduce(&subtotal, &total, 1, MPI\_DOUBLE, MPI\_SUM,  0, MPI\_COMM\_WORLD);  double time2 = MPI\_Wtime();  if (pe == 0) {  total = total \* 4;  printf("Result: %.10lf\n", total);  printf("Accuracy: %.10lf\n", PI - total);  printf("Time: %.10lf\n", time2 - time1);  }  MPI\_Finalize();  } |

串行代码编译命令：

|  |
| --- |
| gcc -o pi-c pi-c.c -lm |

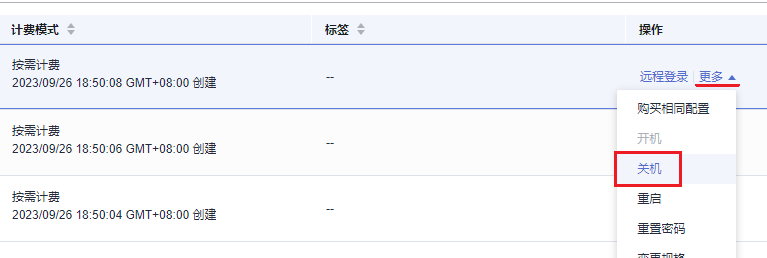
并行代码编译命令：

|  |
| --- |
| mpicc -o pi-mpi pi-mpi.c -lm |

**要求：**

1. 在集群环境中分别编译和执行串行代码和并行代码，记录执行时间并截图保存。
2. 逐步增加并行代码的进程数，看看时间如何变化，并截图保存。

注意：实验结束后，要关闭虚拟机，以免继续扣费。



### 实验评分标准

一、课堂表现（10分）

二、实验结果（50分）

三、实验报告（40分）

### 思考题

思考题1：简述华为云环境下集群的搭建过程。

思考题2：集群之间如果彼此不配置信任秘钥，程序能否正常运行？

思考题3：hello-world程序多次运行，结果一样吗？

思考题4：PI并行代码，增加更多的进程数（超过每个机器的CPU核数），性能如何变化？