系统结构实验报告

实验三 多机环境下 MPI 并行编程

学院：计算机工程与科学学院

学号：22121630

姓名：汪江豪

评分内容	评分
内容完整，有实验目的、步骤、分析、总结	
内容质量，表述清楚正确，条理清晰，分析总结到位	
格式规范，标题、段落、公式、图表、代码符合专业文献出版要求	
综合得分	

1.实验目的：

理解 MPI 的基本原理及其在并行计算中的应用。

2.实验环境：

系统：Ubuntu22.04
编程语言：C 语言
内存：2GB
逻辑 CPU 核心数：4
结点数量：2

3.实验内容：

编写 MPI 程序实现大规模向量的并行计算。

4.实验步骤：

4.1 算法设计：

1.数据分配与初始化：

- 矩阵 A:由主进程（rank 0）初始化后发送给其他进程。
- 矩阵 B:由主进程初始化后通过广播发送给所有进程。
- 矩阵 C:结果矩阵，主进程手机所有子进程的计算结果并汇总。

2.计算分配：

- 每个进程被分配矩阵 A 的一块行和整个矩阵 B。
- 每个进程独立计算其分配的矩阵 A 块和矩阵 B 的成绩，结果放在本地矩阵 C_local 中。

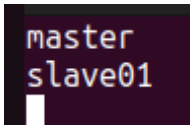
3.结果汇总：

- 主进程使用 MPI_Gather 收集所有子进程的 C_local 计算结果。
- 主进程将收集到的结果汇总，得到完整的矩阵 C.

4.2 准备工作：

1.hostfile 文件编写：

MPI 程序在运行时，可以通过-np $NP 指定进程数， -f $HOSTFILE 指定运行时使用的配置文件。因此，我编写了 mpi_hosts 文件，内容如下：



指定了运行 MPI 程序的两台主机（之前已将主机名与 IP 地址成功绑定）.

2.脚本编写：

由于在多机环境下运行 MPI 程序，需要将 MPI 程序放在同一个目录下，因此，在事先实现了两台机器的 ssh 无密码连接后，可编写脚本实现文件传输功能。我创建了 scp.sh 脚本，用于将 MPI 可执行文件传送到 slave01 主机上的相同目录下，文件名通过形如./scp.sh vec 实现将 vec 文件从 master 结点传输到 slave01 结点上的同一个目录下。Scp.sh 内容如下:

```bash
#!/bin/bash

# 目标机器的用户名和 IP 地址
REMOTE_USER="pioneer"
REMOTE_HOST="slave01"
REMOTE_DIR="~/compute_sys/mpich/"

# 检查是否提供了文件名作为参数
if [ -z "$1" ]; then
    echo "Usage: $0 <filename>"
    exit 1
fi

# 获取输入的文件名
FILE=$1

# 检查文件是否存在
if [ ! -f "$FILE" ]; then
    echo "Error: File '$FILE' does not exist."
    exit 1
fi

# 使用 scp 传输文件
echo "Transferring '$FILE' to $REMOTE_USER@$REMOTE_HOST:$REMOTE_DIR..."
scp "$FILE" "$REMOTE_USER@$REMOTE_HOST:$REMOTE_DIR"

# 检查 scp 命令是否成功
if [ $? -eq 0 ]; then
    echo "File transferred successfully!"
fi
```

其次，我编写了 run.sh 脚本，实现在应用 mpi_hosts 文件情况下，输入进程数，来运行 MPI 程序。可通过./run.sh vec 16 实现，运行 vec 程序，并指定了 16 个进程数。Run.sh 内容如下：

```bash
#!/bin/bash

HOSTFILE="/home/pioneer/compute_sys/mpich/mpi_hosts"

if [ -z "$1" ] || [ -z "$2" ]; then
    echo "Usage: $0 <filename> <num_of_processes>"
    exit 1
fi

# 获取输入的并行数
FILE=$1
NP=$2

# 检查文件是否存在
if [ ! -f "$FILE" ]; then
    echo "Error: File $FILE does not exist."
    exit 1
fi

mpirun -np $NP -f $HOSTFILE ./$FILE
# 检查MPI程序似乎否成功运行
if [ $? -eq 0 ]; then
    echo "MPI program completed successfully!"
else
    echo "Error: MPI program failed to run."
fi
```

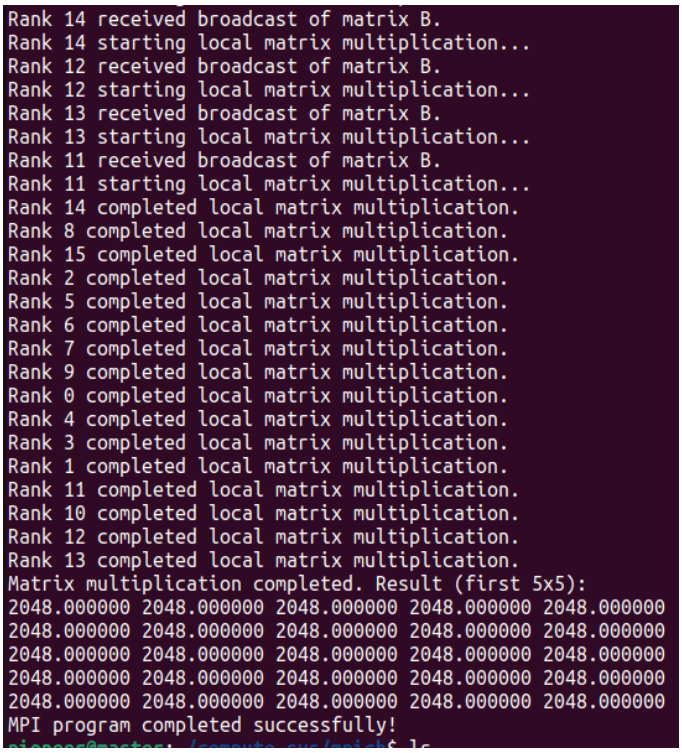
4.3 程序流程：

• MPI_Init()初始化 MPI 环境；MPI_Comm_rank 获取当前进程的编号；MPI_Comm_size 获取总的进程数。总进程数在运行时输入命令指定。

• 初始化 A,B 矩阵，A 全为 1，B 全为 2，大小均为 1024*1024,预期结果为 1024*1024 大小的矩阵，值全为 2048。
• block_size = N / size 指定了每个进程负责的矩阵行数.
• 主进程 rank==0，将 A 矩阵按行分块，通过 MPI_Send 函数分发给其他进程。
• 其他进程通过 MPI_Recv 接收主进程发送的矩阵分块。
• 主进程将 B 矩阵完整地广播给所有进程。
• 各个进程执行本地的部分矩阵乘法，将结果存到 C_local 中。
• 各个进程调用 MPI_Gather 函数，发送自己的 C_local 数据到根进程。该函数建立了一个同步点，所有参与通信的进程必须同步到达这个调用点后，才能执行后续代码。
• 根据其他进程的 rank 顺序，将 C_local 依次填充到结果矩阵 C 中。
• 在主进程中输入了结果矩阵的前 5 行前 5 列。
• 最后释放矩阵内存，MPI_Finalize()函数结束 MPI 过程。

5.实验结果

```
pioneer@master:~/compute_sys/mpich$ ./run.sh vec2 16
Rank 6 is running on host: master
Rank 14 is running on host: master
Rank 0 is running on host: master
Rank 8 is running on host: master
Rank 12 is running on host: master
Rank 4 is running on host: master
Rank 1 is running on host: slave01
Rank 3 is running on host: slave01
Rank 9 is running on host: slave01
Rank 11 is running on host: slave01
Rank 15 is running on host: slave01
Rank 5 is running on host: slave01
Rank 13 is running on host: slave01
Rank 7 is running on host: slave01
Rank 2 is running on host: master
Rank 10 is running on host: master
Rank 0 broadcasting matrix B to all processes...
Rank 9 received broadcast of matrix B.
Rank 9 starting local matrix multiplication...
Rank 10 received broadcast of matrix B.
Rank 10 starting local matrix multiplication...
Rank 8 received broadcast of matrix B.
Rank 8 starting local matrix multiplication...
Rank 7 received broadcast of matrix B.
Rank 7 starting local matrix multiplication...
Rank 6 received broadcast of matrix B.
Rank 6 starting local matrix multiplication...
Rank 5 received broadcast of matrix B.
Rank 5 starting local matrix multiplication...
Rank 4 received broadcast of matrix B.
Rank 4 starting local matrix multiplication...
Rank 3 received broadcast of matrix B.
Rank 3 starting local matrix multiplication...
Rank 2 received broadcast of matrix B.
Rank 2 starting local matrix multiplication...
Rank 0 received broadcast of matrix B.
Rank 1 received broadcast of matrix B.
Rank 1 starting local matrix multiplication...
Rank 0 starting local matrix multiplication...
Rank 15 received broadcast of matrix B.
Rank 15 starting local matrix multiplication...
```



可以看到，16 个进程成功分配到了两个结点上，各进程成功接收并执行了人物，最后结果矩阵输出前 5 行前 5 列结果正确。成功在两台机器上实现了大规模矩阵向量乘法的并行计算，实验结束。