分布式并行计算实验指导书

电子科技大学计算机学院

## 埃拉托斯特尼素数筛选算法并行及性能优化

## 一．实验目的：

1. 使用MPI编程实现埃拉托斯特尼筛法并行算法。

2. 对程序进行性能分析以及调优。

## 二．实验内容：

**操作系统：**Linux

### 1. 埃拉托斯特尼筛法及描述

埃拉托斯特尼是一位古希腊数学家，他在寻找整数Ｎ以内的素数时，采用了一种与众不同的方法：先将２－Ｎ的各数写在纸上：

在２的上面画一个圆圈，然后划去２的其他倍数；第一个既未画圈又没有被划去的数是３，将它画圈，再划去３的其他倍数；现在既未画圈又没有被划去的第一个数是５，将它画圈，并划去５的其他倍数……依此类推，一直到所有小于或等于Ｎ的各数都画了圈或划去为止。这时，画了圈的以及未划去的那些数正好就是小于Ｎ的素数。

这里，我们把Ｎ取120来举例说明埃拉托斯特尼筛法思想：

1. 首先将２到120写出
2. 在２上面画一个圆圈，然后划去２的其它倍数，这时划去的是除了２以外的其它偶数
3. 从２往后一个数一个数地去找，找到第一个没有被划去的数3，将它画圈，再划去３的其它倍数（以斜线划去）
4. 再从３往后一个数一个数地去找，找到第一个没有被划去的数５，将它画圈，再划去５的倍数（以交叉斜线划去）
5. 再往后继续找，可以找到9、11、13、17、19、23、29、31、37、41、43、47…将它们分别画圈，并划去它们的倍数（可以看到，已经没有这样的数了
6. 这时，小于或者等于120的各数都画上了圈或者被划去，被画圈的就是素数了

### 2. 埃拉托斯特尼筛法实现

#### 2.1 基准代码

**代码如下：**

#include "mpi.h"

#include <math.h>

#include <stdio.h>

#define MIN(a,b) ((a)<(b)?(a):(b))

int main (int argc, char \*argv[])

{

int count; /\* Local prime count \*/

double elapsed\_time; /\* Parallel execution time \*/

int first; /\* Index of first multiple \*/

int global\_count; /\* Global prime count \*/

int high\_value; /\* Highest value on this proc \*/

int i;

int id; /\* Process ID number \*/

int index; /\* Index of current prime \*/

int low\_value; /\* Lowest value on this proc \*/

char \*marked; /\* Portion of 2,...,'n' \*/

int n; /\* Sieving from 2, ..., 'n' \*/

int p; /\* Number of processes \*/

int proc0\_size; /\* Size of proc 0's subarray \*/

int prime; /\* Current prime \*/

int size; /\* Elements in 'marked' \*/

MPI\_Init (&argc, &argv);

/\* Start the timer \*/

MPI\_Comm\_rank (MPI\_COMM\_WORLD, &id);

MPI\_Comm\_size (MPI\_COMM\_WORLD, &p);

MPI\_Barrier(MPI\_COMM\_WORLD);

elapsed\_time = -MPI\_Wtime();

if (argc != 2) {

if (!id) printf ("Command line: %s <m>\n", argv[0]);

MPI\_Finalize();

exit (1);

}

n = atoi(argv[1]);

/\* Figure out this process's share of the array, as

well as the integers represented by the first and

last array elements \*/

low\_value = 2 + id\*(n-1)/p;

high\_value = 1 + (id+1)\*(n-1)/p;

size = high\_value - low\_value + 1;

/\* Bail out if all the primes used for sieving are

not all held by process 0 \*/

proc0\_size = (n-1)/p;

if ((2 + proc0\_size) < (int) sqrt((double) n)) {

if (!id) printf ("Too many processes\n");

MPI\_Finalize();

exit (1);

}

/\* Allocate this process's share of the array. \*/

marked = (char \*) malloc (size);

if (marked == NULL) {

printf ("Cannot allocate enough memory\n");

MPI\_Finalize();

exit (1);

}

for (i = 0; i < size; i++) marked[i] = 0;

if (!id) index = 0;

prime = 2;

do {

if (prime \* prime > low\_value)

first = prime \* prime - low\_value;

else {

if (!(low\_value % prime)) first = 0;

else first = prime - (low\_value % prime);

}

for (i = first; i < size; i += prime) marked[i] = 1;

if (!id) {

while (marked[++index]);

prime = index + 2;

}

if (p > 1) MPI\_Bcast (&prime, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

} while (prime \* prime <= n);

count = 0;

for (i = 0; i < size; i++)

if (!marked[i]) count++;

if (p > 1) MPI\_Reduce (&count, &global\_count, 1, MPI\_INT, MPI\_SUM,

0, MPI\_COMM\_WORLD);

/\* Stop the timer \*/

elapsed\_time += MPI\_Wtime();

/\* Print the results \*/

if (!id) {

printf ("There are %d primes less than or equal to %d\n",

global\_count, n);

printf ("SIEVE (%d) %10.6f\n", p, elapsed\_time);

}

MPI\_Finalize ();

return 0;

}

#### 2.2程序的优化：

从以下三个方面优化并行程序性能：

##### 2.2.1 去掉偶数

优化思想利用“大于２的质数都是奇数”这一知识，首先去掉所有偶数，偶数必然不是素数，这样相当于所需要筛选的数减少了一半，存储和计算性能都得到提高。

##### 2.2.2 消除广播

2.2.1的代码是通过进程0广播下一个筛选倍数的素数。进程之间需要通过MPI\_Bcast函数进行通信。通信就一定会有开销，因此我们让每个进程都各自找出它们的前sqrt(n)个数中的素数，在通过这些素数筛选剩下的素数，这样一来进程之间就不需要每个循环广播素数了，性能得到提高。

##### 2.2.3 cache优化

在上面代码基础上，可以根据所掌握的知识，探索重构循环，提高cache命中率等方式进一步优化程序性能，并给出结果和分析。

#### 2.3 实验环境

使用远程连接工具以SSH模式登录集群。注意:只能在校园网环境下连接！第一次登陆后可自行修改密码。

**IP:121.48.162.151**

**端口:22**

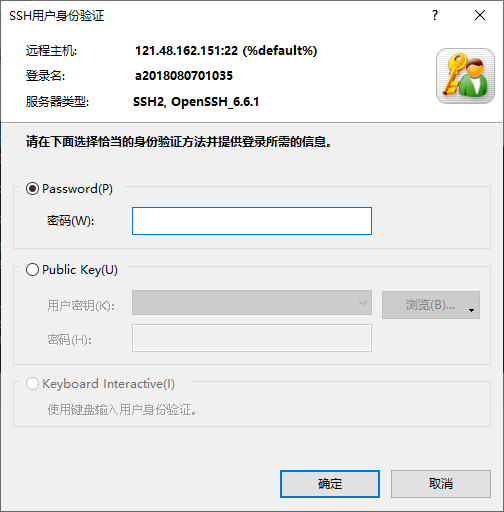
**每个学生的帐户名:std+学号**

**密码:123456**

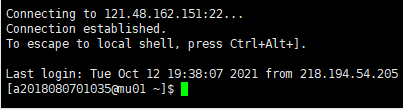
例如，在Xshell软件中：



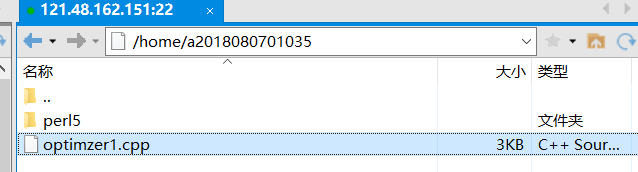
输入密码：



连接成功：



更新代码：



要运行代码，需要进入计算节点cu01或cu02。在管理节点上运行以下命令，以登录到计算节点:



编译.cpp代码文件:



运行代码：



## 三．实验要求：

实验前要做好充分准备，包括程序清单、调试步骤、调试方法，以及对程序结果的分析等。

## 四．实验评分：

本实验分为四个部分，具体评分标准如下：

1. 安装部署MPI实验环境，并调试完成基准代码，并实测在不同进程规模（1，2，4，8，16）加速比，并合理分析原因（40分）
2. 完成优化1，去除偶数优化，并实测在不同进程规模（1，2，4，8，16）加速比，并合理分析原因（10分）
3. 完成优化2，消除广播优化，并实测在不同进程规模（1，2，4，8，16）加速比，并合理分析原因（15分）
4. 完成优化3，cache优化，并实测在不同进程规模（1，2，4，8，16）加速比，并合理分析原因（10分）。
5. 性能得分：在完成优化3的基础上，可以利用课内外知识，全面优化代码性能。根据全班优化3在目标机上实测性能，最高性能（最短执行时间）得分25分，最低性能得0分，其他按执行时间进行插值。（25分）

目标机器配置：

浪潮 5280M4

CPU： E5-2660 v4 2颗

内存：64G

## 实验报告：

1．程序说明。说明程序的功能、结构。

2．调试说明。包括上机调试的情况、上机调试步骤、调试所遇到的问题是如何解决的，并对调试过程中的问题进行分析，对执行结果进行分析。

3．画出程序框图。

4．给出源程序清单并清晰注释。

5. 给出程序运行结果。

6. 实验报告需附上可见的手写(电子)签名。

### 

**电 子 科 技 大 学**

**实 验 报 告**

|  |  |
| --- | --- |
| **学生姓名：** | **学 号：** |
| **一、实验室名称：** | |
| **二、实验项目名称：** | |
| **三、实验原理：** | |
| **四、实验目的：** | |
| **五、实验内容：** | |
| **六、实验器材（设备、元器件）：** | |
| **七、实验步骤及操作：** | |
| **八、实验数据及结果分析：** | |
| **九、实验结论：** | |
| **十、总结及心得体会：** | |
| **十一、对本实验过程及方法、手段的改进建议：** | |
| **报告评分：**  **指导教师签字：** | |