**深 圳 大 学**

**实 验 报 告**

**课程名称****：并行计算**

**实验名称：矩阵乘法的MPI并行程序**

**姓 名：刘俊楠**

**学 号：2017303010**

**班 级：计科一班**

**实验日期：第12周实验课**

## 一. 实验目的

1. 学会编写简单的MPI程序；

2. 对并行程序进行简单的性能分析。

## 二. 实验环境

1. 软件环境：Microsoft Visual Studio 2013。

## 三. 实验内容

1. 实验要求：用MPI编写两个*n*阶方阵A和B的乘法程序，结果存放在方阵C中。

* 初始时，A和B都存储在进程0中，其他进程没有数据。结束时，C也存储在进程0中。
* A和B中的每个数都初始化为一个0到1之间的随机double型值（用rand()/double(RAND\_MAX)实现）。
* 添加检测计算结果是否正确的代码。
* 计算执行时间用MPI\_Wtime()函数。

2. 程序代码和说明：

#include<iostream>

#include<mpi.h>

#include<math.h>

#include<stdlib.h>

#include<time.h>

**using** **namespace** std**;**

//初始化函数initABC

//一维数组表二维 row为行 cols为列 A为一维数组

void initABC**(**double**\*** A**,** int row**,** int cols**)**

**{**

**for** **(**int i **=** 0**;** i **<** row **\*** cols**;** i**++)** **{**

A**[**i**]** **=** **(**double**)**rand**()** **/** double**(**RAND\_MAX**);**

**}**

**}**

//计算分块矩阵的矩阵乘法函数

//A、B：表示两个分块矩阵的一维数组 multiplyResult数组：为存放的结果数组

//m：数组A的行数 p：数组A的列数和数组B的行数 n：数组B的列数

//因为后续可能分块除不尽 所以必须得设置区别AB的行数和列数

void calculatedividedCube**(**double**\*** A**,** double**\*** B**,** double**\*** multiplyResult**,** int m**,** int p**,** int n**)**

**{**

**for** **(**int i **=** 0**;** i **<** m**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** n**;** j**++)** **{**

double temp **=** 0**;**

**for** **(**int k **=** 0**;** k **<** p**;** k**++)** **{**

temp **+=** A**[**i **\*** p **+** k**]** **\*** B**[**k **\*** n **+** j**];**

**}**

multiplyResult**[**i **\*** n **+** j**]** **=** temp**;**

**}**

**}**

**}**

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

**{**

//n：设置的矩阵的行数与列数 times：循环次数

int n **=** 1000**,** times **=** 5**;**

//start、end：并行时间记录

//time3：串行时间记录

//sum：并行计算总时间，用于计算并行计算平均时间

double start**,** end**,** time3**,** sum **=** 0**;**

//ABC为一维数组，存放二维矩阵

double**\*** A**,** **\*** B**,** **\*** C**;**

//b\_A：A的分块结果 b\_C：A与B分块乘结果 single\_C：串行下C矩阵存储结果 用于对比

double**\*** b\_A**,** **\*** b\_C**,** **\*** single\_C**;**

//myrank：当前进程号 proc\_num：进程数目

int myrank**,** proc\_num**;**

// 并行开始 argc为变量数目，argv为变量数组

MPI\_Init**(&**argc**,** **&**argv**);**

//设置进程数目 并且赋值给proc\_num

MPI\_Comm\_size**(**MPI\_COMM\_WORLD**,** **&**proc\_num**);**

//获取进程号 进程号赋值给myrank

MPI\_Comm\_rank**(**MPI\_COMM\_WORLD**,** **&**myrank**);**

**while** **(**times**--)**

**{**

//初始化数据

//bn：分块依据

int bn **=** n **/** proc\_num**;**

b\_A **=** **new** double**[**bn **\*** n**];**

B **=** **new** double**[**n **\*** n**];**

b\_C **=** **new** double**[**bn **\*** n**];**

A **=** **new** double**[**n **\*** n**];**

C **=** **new** double**[**n **\*** n**];**

single\_C **=** **new** double**[**n **\*** n**];**

//我们假设myrank=0为主线程

**if** **(**myrank **==** 0**)** **{**

//初始化AB矩阵

initABC**(**A**,** n**,** n**);**

initABC**(**B**,** n**,** n**);**

//当第一次运行时执行串行计算，只执行一次减少运行时间

**if** **(**times **==** 4**)**

**{**

//串行计算矩阵乘法

clock\_t start1 **=** clock**();**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** **{**

**for** **(**int j **=** 0**;** j **<** n**;** j**++)** **{**

double temp **=** 0.0**;**

**for** **(**int k **=** 0**;** k **<** n**;** k**++)** **{**

temp **+=** A**[**i **\*** n **+** k**]** **\*** B**[**k **\*** n **+** j**];**

**}**

single\_C**[**i **\*** n **+** j**]** **=** temp**;**

**}**

**}**

clock\_t end1 **=** clock**();**

time3 **=** **(**double**)(**end1 **-** start1**)** **/** 1000**;**

cout **<<** "串行时间为：" **<<** time3 **<<** "s" **<<** endl**;**

**}**

**}**

//开始并行计算 start：记录当前时间

start **=** MPI\_Wtime**();**

//MPI\_Scatter：散播函数 作用：分块，将一段array 的不同部分发送给所有的进程

//这条指令意思是A矩阵在每个进程中取出bn\*n的大小，

//类型为double，分散发给每个进程的b\_A数组，

//他们也只是接受bn\*n大小的double值，分发主要由0进程进行，发给所有进程

MPI\_Scatter**(**A**,** bn **\*** n**,** MPI\_DOUBLE**,** b\_A**,** bn **\*** n**,** MPI\_DOUBLE**,** 0**,** MPI\_COMM\_WORLD**);**

//MPI\_Bcast:广播函数

//该指令主要是将B矩阵n\*n的double值大小全部发给所有进程，发送者为0号进程，接受者为所有进程

MPI\_Bcast**(**B**,** n **\*** n**,** MPI\_DOUBLE**,** 0**,** MPI\_COMM\_WORLD**);**

//0号进程将相同的信息发送给所有的进程

//各个进程将各自的b\_A矩阵和B矩阵相乘，然后结果放在b\_C中，

//b\_A矩阵的行为bn，列为n B矩阵的行数为n，列数也为n

calculatedividedCube**(**b\_A**,** B**,** b\_C**,** bn**,** n**,** n**);** //计算C的各个分块

//MPI\_Barrier 阻止调用直到communicator中所有进程完成调用。

MPI\_Barrier**(**MPI\_COMM\_WORLD**);**

//MPI\_Gather：收集函数

//该指令意思是每个进程按顺序把他们自己的b\_C矩阵

//发送bn\*n个double大小的值给C矩阵，

//当然C矩阵也只接受bn\*n的double大小的值，

//接受者为0号进程，发送者为所有进程 发送的数据按进程标识排队

MPI\_Gather**(**b\_C**,** bn **\*** n**,** MPI\_DOUBLE**,** C**,** bn **\*** n**,** MPI\_DOUBLE**,** 0**,** MPI\_COMM\_WORLD**);** //汇总结果

//计算多余分块 当1000\*1000矩阵分块时，如果有除不尽的情况，有些块就会分不了，此时就需要处理剩余未分的块

//restid:计算当前分块是否合理 如果≠n则是分块没有分完，需要执行特殊处理

int restid **=** bn **\*** proc\_num**;**

//利用0号进程处理剩余未分的块

**if** **(**myrank **==** 0 **&&** restid **<** n**)** **{**

//remainrows：剩余未处理行

int remainRows **=** n **-** restid**;**

//处理A与B矩阵剩余未处理行，A的行为remainRows 列为n B的行列都为n

calculatedividedCube**(**A **+** restid **\*** n**,** B**,** C **+** restid **\*** n**,** remainRows**,** n**,** n**);**

**}**

**delete[]** b\_A**;**

**delete[]** B**;**

**delete[]** b\_C**;**

**if** **(**myrank **==** 0**)** **{**

//end：并行运行时间 利用0号进程计算

end **=** MPI\_Wtime**()** **-** start**;**

sum **+=** end**;**

cout **<<** proc\_num **<<** "个进程的时间为：" **<<** end **<<**"秒" **<<** endl**;**

//只判断一次 减少运行时间

**if** **(**times **==** 4**)**

**{**

**for** **(**int i **=** 0**;** i **<** n**;** i**++)** // 判断是否出错

**if** **(**C**[**i**]** **-** single\_C**[**i**]** **>=** 1e-6**)**

**{**

cout **<<** i **<<** " " **<<** C**[**i**]** **<<** " " **<<** single\_C**[**i**]** **<<** endl**;**

cout **<<** "error" **<<** endl**;**

**break;**

**}**

**}**

**delete[]** A**;**

**delete[]** C**;**

**}**

**}**

MPI\_Finalize**();** // 并行结束

**if** **(**myrank **==** 0**)** **{**

cout **<<** "平均时间为：" **<<** sum **/** 5 **<<**"秒" **<<** endl**;**

cout **<<** "加速比为：" **<<** time3 **/** **(**sum **/** 5**)** **<<** endl**;**

**}**

**return** 0**;**

**}**

3. 实验结果和分析：测试并行程序在不同进程数下的执行时间和加速比（串行执行时间/并行执行时间），并分析实验结果。其中，*n*固定为1000，进程数分别取1、2、4、8、16、32、64时，为减少误差，每项实验进行5次，取平均值作为实验结果。

表1 并行程序在不同进程数下的执行时间（秒）和加速比

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 线程数  执行时间 | 1 | 2 | 4 | 8 | 16 | 32 | 64 |
| 第1次 | 5.27609 | 2.98578 | 1.70813 | 1.19712 | 1.16763 | 1.42348 | 2.26016 |
| 第2次 | 5.32446 | 2.85986 | 1.73604 | 1.1976 | 1.16785 | 1.74787 | 3.6721 |
| 第3次 | 5.26745 | 2.84612 | 1.75575 | 1.21971 | 1.29675 | 1.98776 | 4.22285 |
| 第4次 | 5.25218 | 2.8182 | 1.71648 | 1.23534 | 1.28422 | 2.18039 | 4.88066 |
| 第5次 | 5.30478 | 2.81126 | 1.7842 | 1.23223 | 1.26462 | 1.99818 | 4.77452 |
| 平均值 | 5.28499 | 2.86424 | 1.74012 | 1.2164 | 1.23621 | 1.86754 | 3.96206 |
| 加速比 | 0.995271 | 1.84796 | 3.03945 | 4.33328 | 4.39163 | 2.90757 | 1.37378 |