并行计算课程

结 题 报 告

报告名称:	多线程实现矩阵转置算法的性能分析
姓　　名:	李润泽
学　　号:	3019244266
联系电话:	15942643201
电子邮箱:	Lirz3019244266@163.com
填写日期:	2021 年 4 月 6 日

2020 年制

一、实验内容概述

本实验要求使用多线程来进行大规模矩阵转置运算，使用不同转置方式以及不同线程数并进行性能分析。旨在提升学生对并行计算的理解和认识，培养学生编写并行程序的能力，并巩固和加深学生对多线程（pthread）并行编程的理解和认识。

在进行矩阵转置的过程中，我们分别使用串行算法和并行算法。课程中并行计算提供了两种转置方式：块棋盘划分方法和直角划分方法。实验要求学生编写相应的代码，并通过改变线程数，分别进行不同方式的数据统计、时间记录以及性能分析。（代码中的 N 尽可能大）

1.计算方法：串行算法、并行算法（块棋盘划分方法、直角划分方法）

2.编程语言：C 或 C++

3.并行计算操作系统：天津大学超算平台 CentOS 7.6

4.编译环境：Intel 19.1.0.166

5.脚本编写：系统提交需要编写 PBS 脚本实现

6.数据分析要求：提供实验结果数据、加速比曲线以及效率

二、并行算法分析设计

（一）实现方法

1 输入线程数 t(1、4 或 9)，与总数据量(矩阵为 900*900、1800*1800 或 2700*2700)，记录时间

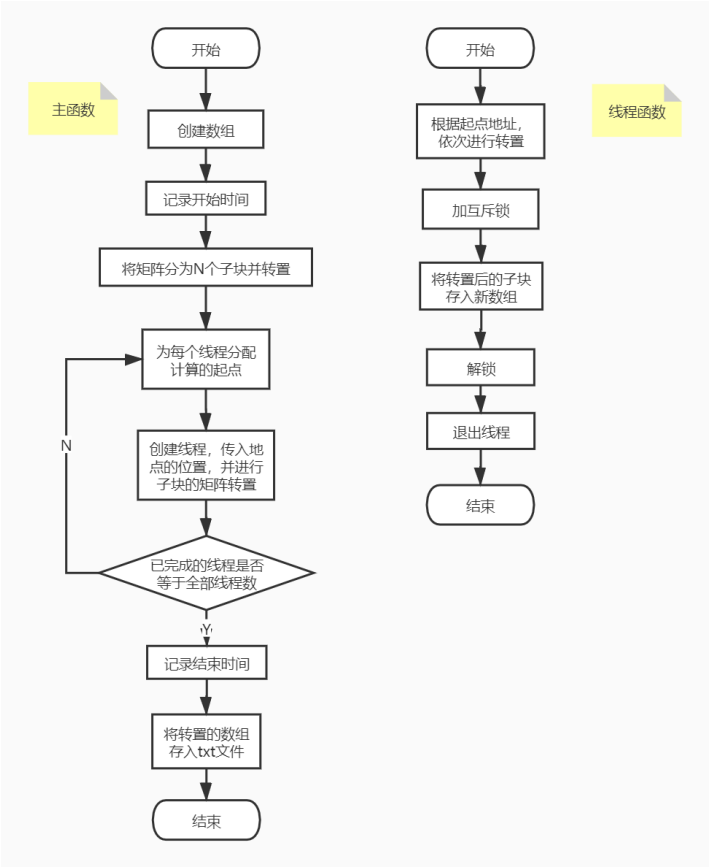
2 为每个线程分配计算的数据量

3 创建线程，每个线程计算各自部分

4 计算完成后，将各自线程计算的部分汇总得出最终结果，并再次记录时间

5 输出转置矩阵

（二）程序流程图



（三）转置算法

1 串行算法

1.1 转置思路

输入：矩阵 A_{n*n}

输出：矩阵 A_{n*n} 的转置 A^{T}_{n*n}

```
Begin
    for i=2 to n do
        for j=1 to i-1 do
            swap(a[i,j], a[j,i])
        endfor
    endfor
End
```

1.2 核心代码（完整代码位于 matrix_serial.c）

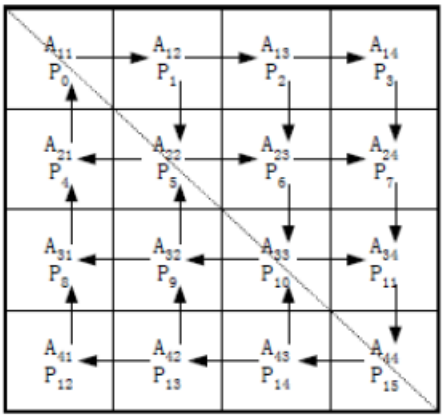
```
void serial(int row, int column, int n){
    int i,j;
    int temp;
    for(i=row; i<n; i++){
        for(j=column; j<i; j++){
            temp=matrix[i][j];
            matrix[i][j]=matrix[j][i];
            matrix[j][i]=temp;
```

```
        }
    }
}
```

2 并行算法——块棋盘划分方法

2.1 转置思路

假设线程数为 p，编号为 0,1,…,p-1，则将 n 阶矩阵 A 分成 p 个大小为 m*m 个子块，p 个子块组成一个 √p* √p 的子块阵列，如下图所示：



转置分为两步进行：第一步，子块转置；第二步，处理器内部局部转置。

2.2 核心代码（完整代码位于 matrix_chessboard.c）

```
void *thread_function(void *ID){
    int id=*(int*)ID;
    int u=id/sqrt((double)p);
```

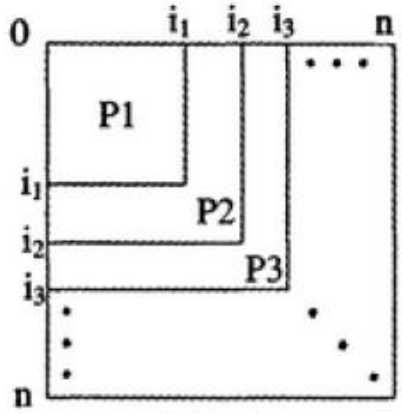
```
    int v=id%((int)sqrt((double)p));
    int m=sqrt((double)(N*N/p));
    int i,j;
    for(i=u*m;i<(u+1)*m;i++){
        for(j=v*m;j<(v+1)*m;j++){
            matrix[i][j]=temp[j][i];
        }
    }
    return NULL;
}
```

3 并行算法——直角划分方法

3.1 转置思路

转置分为两步进行：第一步，将矩阵划分为大小相近的 p 个子块；第二步，对每一个子块进行转置。

直角划分方法如下图所示：



3.2 核心代码（完整代码位于 matrix_rightAngle.c）

```
void *thread_function(void *ID){
    int id=*(int*)ID;
    int length=N/p;
    int i,j,temp;

    for(i=id*length;i<(id+1)*length&&i<N;i++){
        for(j=0;j<i;j++){
            temp=matrix[i][j];
            matrix[i][j]=matrix[j][i];
            matrix[j][i]=temp;
        }
    }
```

```
    return NULL;
}
```

（四）运行脚本（以 test_matrix_chessborad.pbs，p=9，N=900 为例）

```
#!/bin/bash
#PBS -N test
#PBS -q qstudent
#PBS -l nodes=1:ppn=9
#PBS -j oe

#cd $PBS_O_WORKDIR
date +%s.%N
./matrix_chessboard 9 900
date +%s.%N
```

三、实验数据分析

（一）实验环境

CPU：Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz　2.59 GHz

内存：16.0GB

互联网络参数：172.23.80.20（用的是校园网）

（二）实验数据综合分析

加速比：$S(n)=\frac{\text{单线程计算时间}}{\text{多线程计算时间}}=\frac{t_s}{t_p}$

效率：$E=\frac{\text{单线程计算时间}}{\text{多线程计算时间}\times\text{处理器数}}=\frac{t_s}{t_p\times n}$

1 实验数据

1.1 串行算法

N	run time
900	0.248253345
1800	0.773203373
2700	1.718780041

1.2 并行算法——块棋盘划分方法

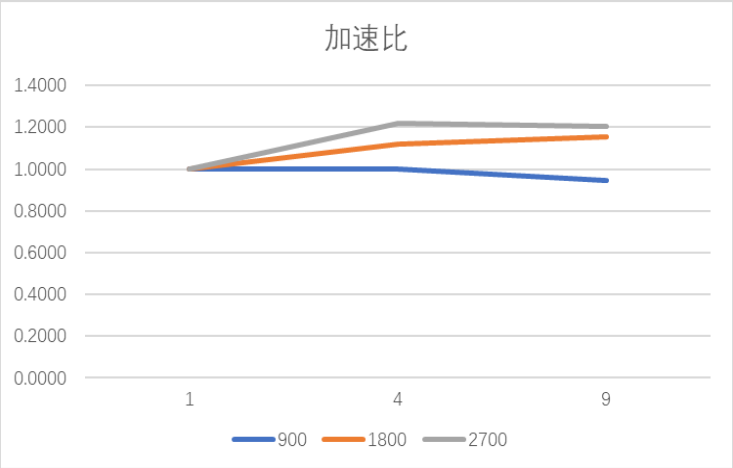
1.2.1 计算时间

N / ppn	1	4	9
900	0.240400076	0.240530014	0.254689932

1800	0.857650042	0.766340017	0.743349791
2700	2.200020075	1.797260046	1.830499887

1.2.2 加速比

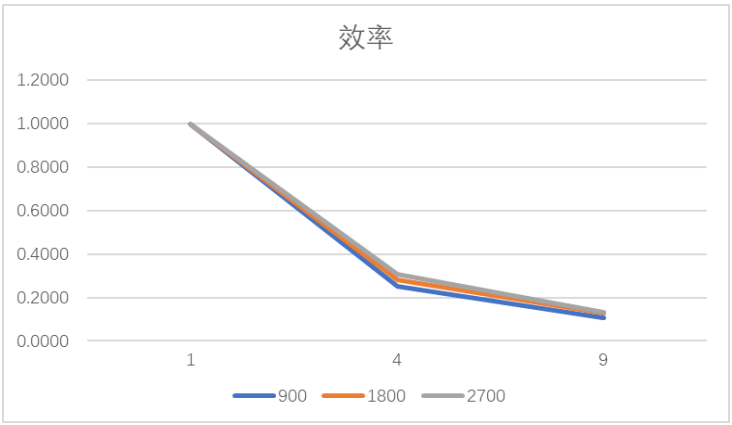
N / ppn	1	4	9
900	1.0	0.9996	0.9442
1800	1.0	1.1191	1.1538
2700	1.0	1.2200	1.2019



1.2.3 效率

N / ppn	1	4	9

900	1.0	0.2499	0.1049
1800	1.0	0.2798	0.1282
2700	1.0	0.1335	0.1335



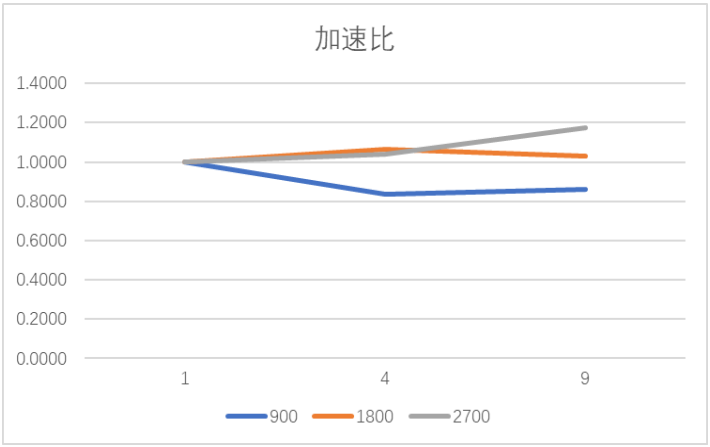
1.3 并行算法——直角划分方法

1.3.1 计算时间

N / ppn	1	4	9
900	0.207610130	0.248159885	0.241169930
1800	0.775909901	0.729039907	0.754819870
2700	1.932420015	1.861680031	1.645979881

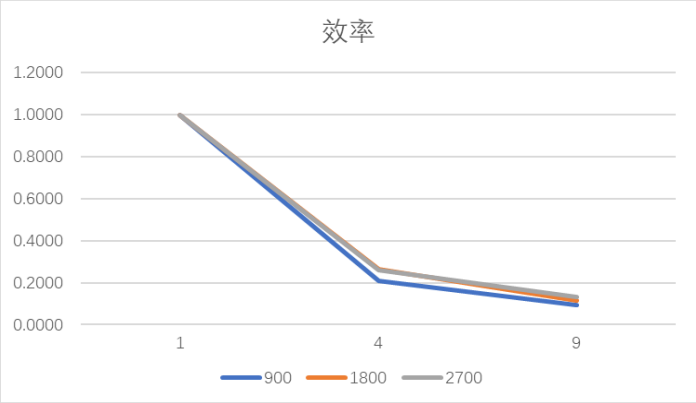
1.3.2 加速比

N ppn	1	4	9
900	1.0	0.8364	0.8607
1800	1.0	1.0643	1.0280
2700	1.0	1.0380	1.1741



1.3.3 效率

N ppn	1	4	9
900	1.0	0.2091	0.0956
1800	1.0	0.2661	0.1142
2700	1.0	0.2595	0.1305



四、实验总结

根据实验数据，我们可以分析出：

1、多线程并行的效果比串行程序结果好；随着数据规模的增大，程序耗费时间不断上升，但并行的效果更加明显。

2、随着线程量的增加，多线程并行程序的加速比不断增加，但效率不断变低。

在本次实验中，我对多线程编程又有了更加进一步的了解与认识，与第一次实验相比，本次的实验进行起来较为轻松。当然，在编程的过程中，我也遇到了一些问题。首先我需要将读取的矩阵分成子块，并把子块的首地址进行转置。一开始始终没能成功，是由于我读取的矩阵与开的矩阵规模不同导致。还有就是实现多线程的过程中，我没有做到各个线程并行，导致没有做到优化。

通过编写两种不同的并行算法来进行矩阵的转置，我对计算机并行计算有了更加深刻的认识。通过对实验中串行、并行结果的分析，我也更加清晰认识到并行计算的重要性。这会对我未来编写有效高速的代码奠定了更加坚实的基础。

五、课程总结

本次实验在有了第一次实验的基础上，继续在 Windows 系统和天津大学超算平台的环境下进行多线程编程。有了第一次实验的经验以及教训，我在编程过程中变得更加轻松，尽管在过程中遇到了或多或少的错误，但在自己的不断修改、优化以及与同学的沟通下，我算是顺利地完成了此次实验。就授课的内容而言，授课教师和助教对我的帮助很大，实验指导书讲解详细，图文并茂，在编程前可以起到很好的帮助，可谓是事半功倍。

在实验过程中，请允许我在此提出一下建议：

1，实验指导书中可以适当添加一些与并行计算相关联知识的网络链接，可以提供一个让同学们自主获取知识的渠道，这样可以更加高效率地进行该学科的学习。

2，建议实验期间进行开放式问题的探究（可以选做），这样可以促进同学们进行深入挖掘，对并行计算有更多了解。

在做实验之前，我认为我们必须要将课程上学到的理论知识完全吸收，这是进行实践过程中最为重要的基石。至此，第二次实验顺利完成，这对我在并行计算方面的学习有巨大的帮助。

附：上机实验与课程知识点分析

序号	上机实验内容	理论知识点	分析总结
1	矩阵转置的方式	两种并行计算方式详见	在块棋盘划分方法中，注意线程需要是整数的平方，否则转置过程中易出现错

		实验指导书	误。
2	加速比	$S(n)=t_s/t_p$	加速比等于串行计算时间与多线程计算时间的比值
3	效率	$E=S(n)/n$	效率等于加速比与线程数的比值
4			