苏亦凡　　计算机科学与技术学院　　200111229

实验五

实验内容

使用OpenMP实现多线程计算矩阵乘法，并与单线程计算，pthread实现的多线程计算进行比较。

设计方案

本次实验所用的CPU为i7-1065G7，规格为四核心。使用pthread实现多线程计算与实验四一致。使用OpenMP实现只需在单线程的基础上在 `for` 循环前加上预处理指令，即：

```
#pragma omp parallel for
```

核心代码如下

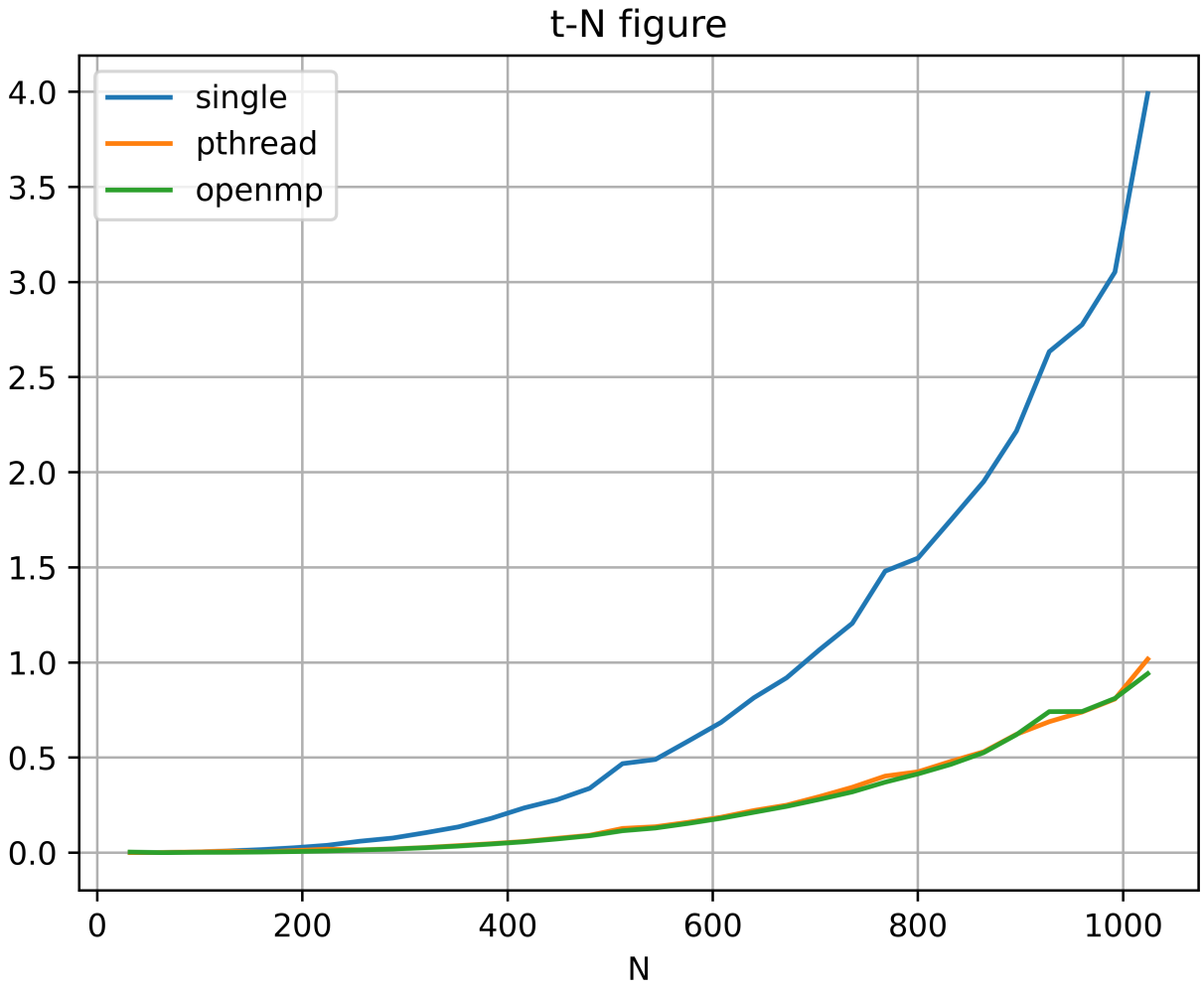
```
void omp_matmul(int N)
{
    #pragma omp parallel for shared(A, B, C)
    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            for (int k = 0; k < N; ++k)
                c(i, j) += a(i, k) * b(k, j);
}
```

在[0,1024]以步长为32取矩阵规模，比较单线程串行计算与两种多线程计算的时间，并按照格式写入data.csv。最后利用python进行数据的处理。

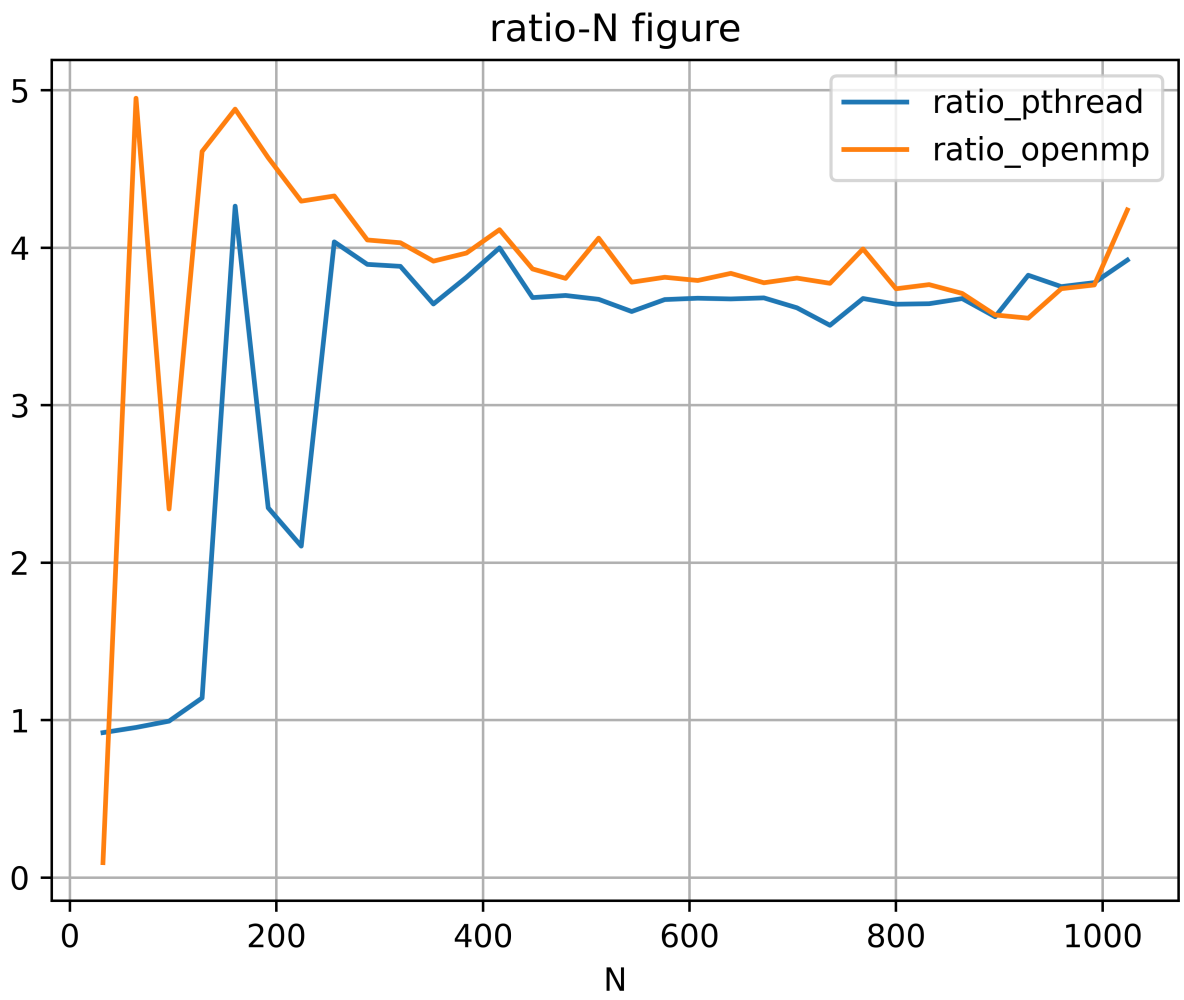
实验数据

原始数据[1]

三种计算方法的时间比较：



两种多线程实现方法的加速比率：



OpenMP和pthread的区别

不需要手动实现线程的创建与同步，操作简便，代码可读性更好，但要求代码可并行且符合要求。两种方法对时间效率的提升区别不大。

1. ↩

N	single	pthread	openmp
32	0.000313	0.000340	0.003299
64	0.001168	0.001225	0.000236
96	0.003631	0.003654	0.001551
128	0.009098	0.007976	0.001973

160	0.015977	0.003747	0.003274
192	0.025806	0.010990	0.005645
224	0.038670	0.018369	0.009002
256	0.060160	0.014902	0.013898
288	0.076850	0.019734	0.018979
320	0.105005	0.027051	0.026046
352	0.135394	0.037168	0.034585
384	0.180309	0.047295	0.045461
416	0.235072	0.058783	0.057133
448	0.278006	0.075475	0.071925
480	0.338344	0.091524	0.088912
512	0.468064	0.127456	0.115266
544	0.489942	0.136288	0.129574
576	0.586360	0.159738	0.153804
608	0.684697	0.186091	0.180568
640	0.814230	0.221574	0.212196
672	0.919583	0.249789	0.243431
704	1.066980	0.294868	0.280252
736	1.205828	0.343792	0.319504
768	1.480039	0.402458	0.370725
800	1.548641	0.425311	0.414149
832	1.748357	0.479740	0.464277
864	1.950671	0.530488	0.525825
896	2.216566	0.622340	0.620382
928	2.634195	0.688646	0.741486
960	2.775800	0.739612	0.742244

992	3.052660	0.807923	0.811112
1024	3.990788	1.017640	0.941204