作业四

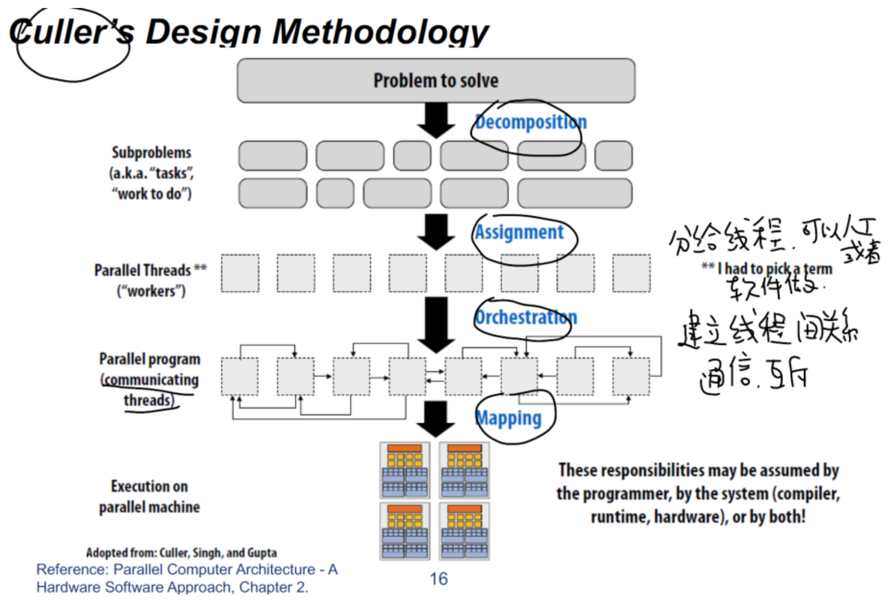
19335015 陈恩婷

一、问题描述

利用 Culler 并行程序设计方法计算 1000x1000 的矩阵与 1000x1 的向量之间的乘 积，要求清晰地呈现 Culler 并行程序设计的四个步骤，并比较程序在不同阶段 具有不同配置时如不同的子任务数量、不同的线程数量、不同的映射方案的性能差别。

二、解决方案

1. Culler 并行程序方法


Culler's Design Methodology
Problem to solve
Decomposition
Subproblems (a.k.a. "tasks", "work to do")
Assignment
Parallel Threads ** ("workers")
** I had to pick a term
分给线程. 可以人工或者软件做.
Orchestration
建立线程间联系 通信. 互斥
Parallel program (communicating threads)
Mapping
Execution on parallel machine
These responsibilities may be assumed by the programmer, by the system (compiler, runtime, hardware), or by both!
Adopted from: Culler, Singh, and Gupta
Reference: Parallel Computer Architecture - A Hardware Software Approach, Chapter 2.
16

2. 矩阵乘以向量

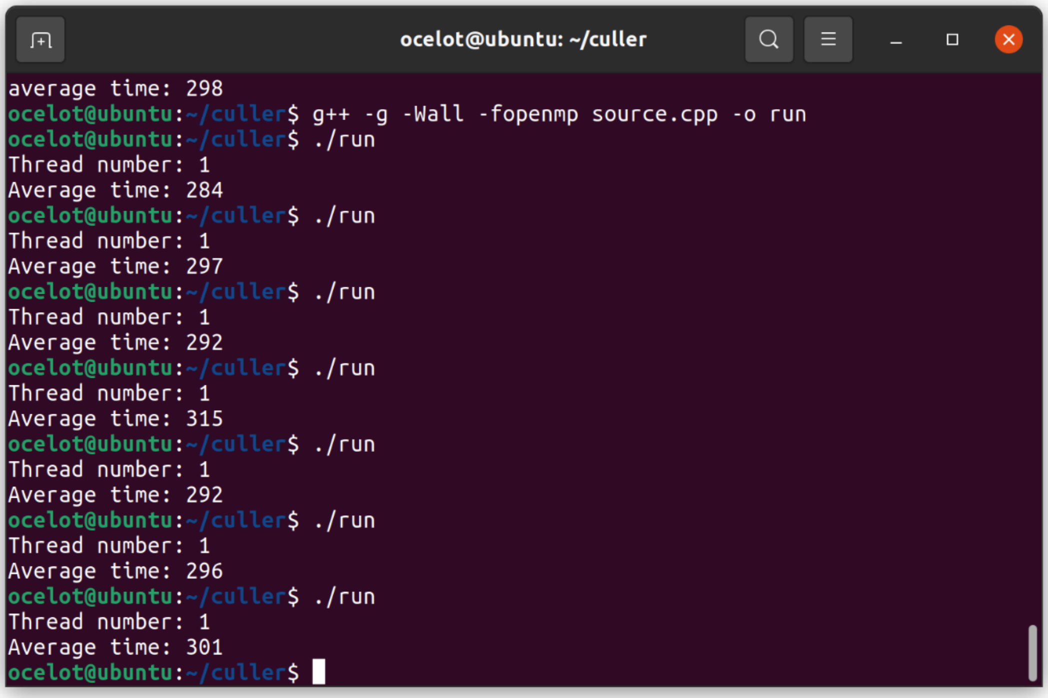
本次实验选用 openMP 对程序进行并行化，只需要加上"pragma omp parallel for num_threads(ThreadNumber)"即可。openMP 使用共享内存，所以用起来相对方便。

三、实验结果

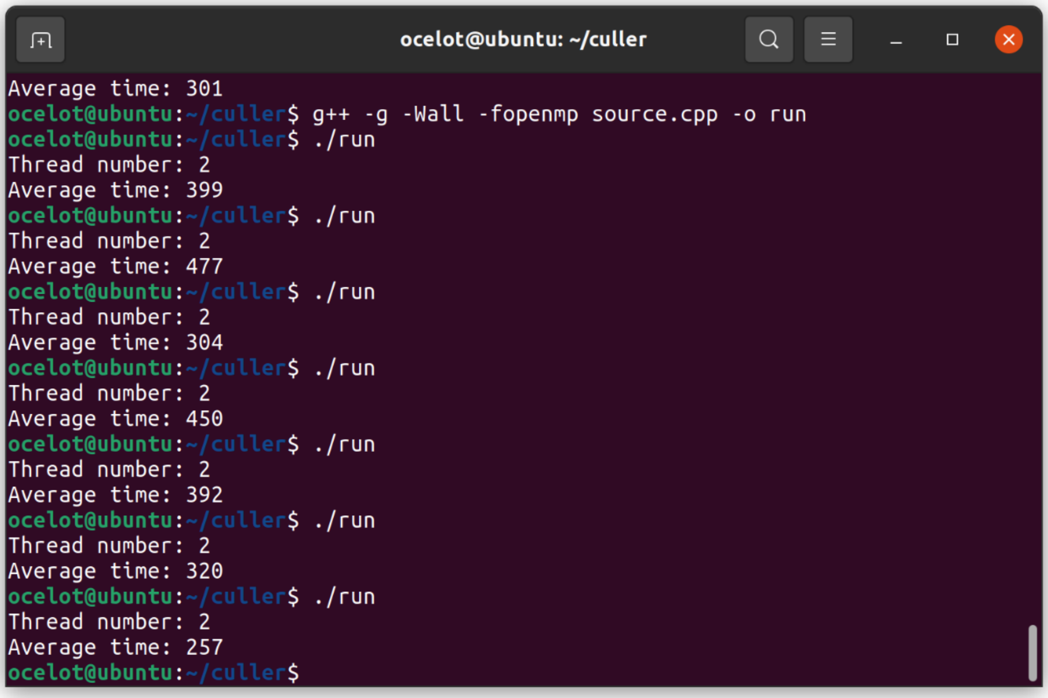
根据 openMP 和本程序的具体情况，本实验主要讨论程序可能的不同配置包括不同的线程数量和不同的调度方案等。

1. 不同的线程数量

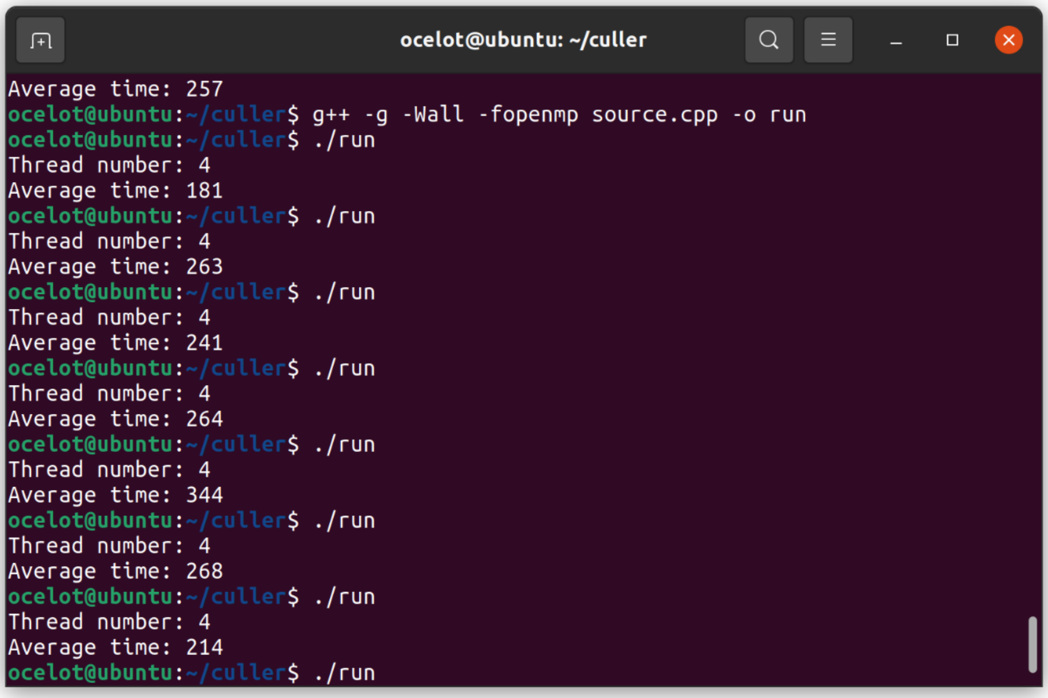
   本人通过修改 Multiplication 函数中 #pragma omp parallel for num_threads (ThreadNumber) 一行中的 ThreadNumber 的值来探究线程数量对程序运行时间的影响：



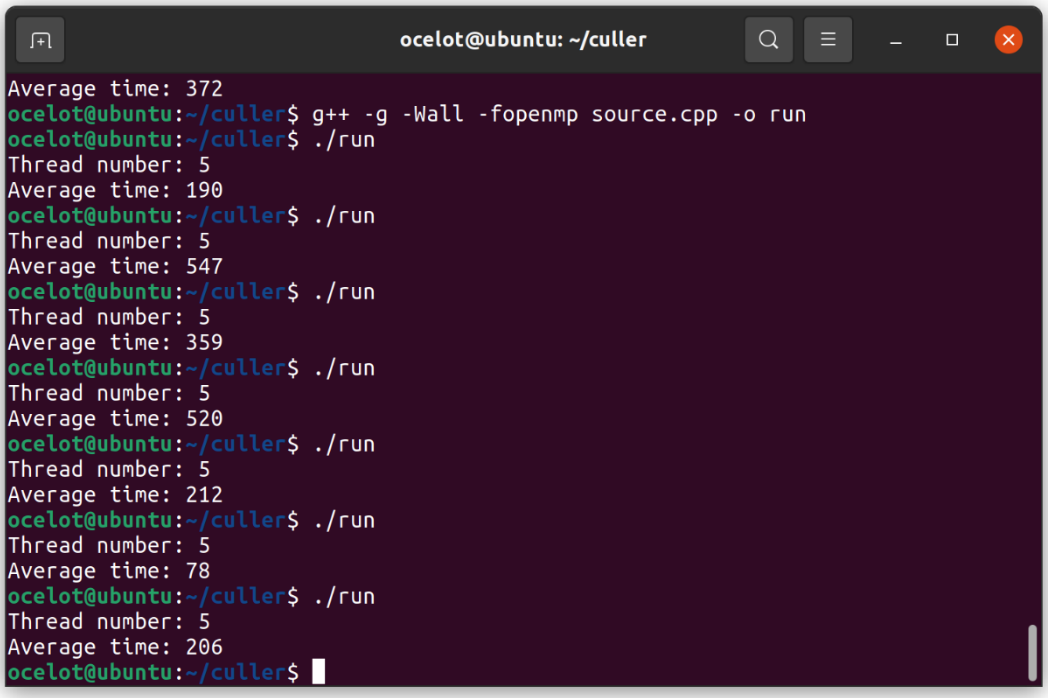
```
average time: 298
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
ocelot@ubuntu:~/culler$ ./run
Thread number: 1
Average time: 284
ocelot@ubuntu:~/culler$ ./run
Thread number: 1
Average time: 297
ocelot@ubuntu:~/culler$ ./run
Thread number: 1
Average time: 292
ocelot@ubuntu:~/culler$ ./run
Thread number: 1
Average time: 315
ocelot@ubuntu:~/culler$ ./run
Thread number: 1
Average time: 292
ocelot@ubuntu:~/culler$ ./run
Thread number: 1
Average time: 296
ocelot@ubuntu:~/culler$ ./run
Thread number: 1
Average time: 301
ocelot@ubuntu:~/culler$
```



```
Average time: 301
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
ocelot@ubuntu:~/culler$ ./run
Thread number: 2
Average time: 399
ocelot@ubuntu:~/culler$ ./run
Thread number: 2
Average time: 477
ocelot@ubuntu:~/culler$ ./run
Thread number: 2
Average time: 304
ocelot@ubuntu:~/culler$ ./run
Thread number: 2
Average time: 450
ocelot@ubuntu:~/culler$ ./run
Thread number: 2
Average time: 392
ocelot@ubuntu:~/culler$ ./run
Thread number: 2
Average time: 320
ocelot@ubuntu:~/culler$ ./run
Thread number: 2
Average time: 257
ocelot@ubuntu:~/culler$
```

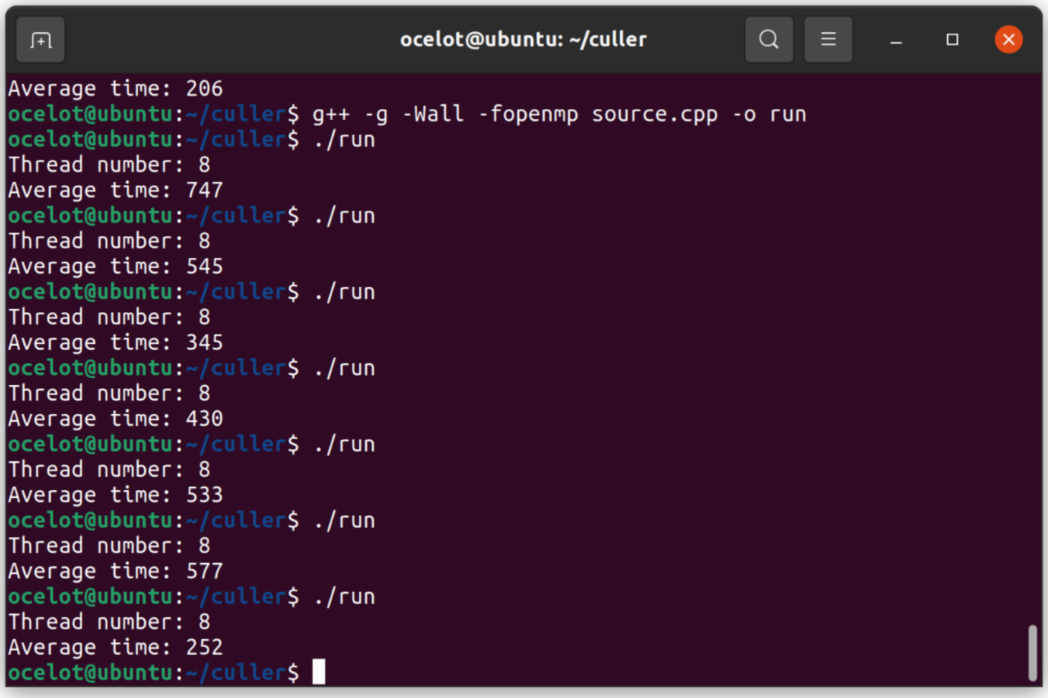


```
Average time: 257
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
ocelot@ubuntu:~/culler$ ./run
Thread number: 4
Average time: 181
ocelot@ubuntu:~/culler$ ./run
Thread number: 4
Average time: 263
ocelot@ubuntu:~/culler$ ./run
Thread number: 4
Average time: 241
ocelot@ubuntu:~/culler$ ./run
Thread number: 4
Average time: 264
ocelot@ubuntu:~/culler$ ./run
Thread number: 4
Average time: 344
ocelot@ubuntu:~/culler$ ./run
Thread number: 4
Average time: 268
ocelot@ubuntu:~/culler$ ./run
Thread number: 4
Average time: 214
ocelot@ubuntu:~/culler$ ./run
```

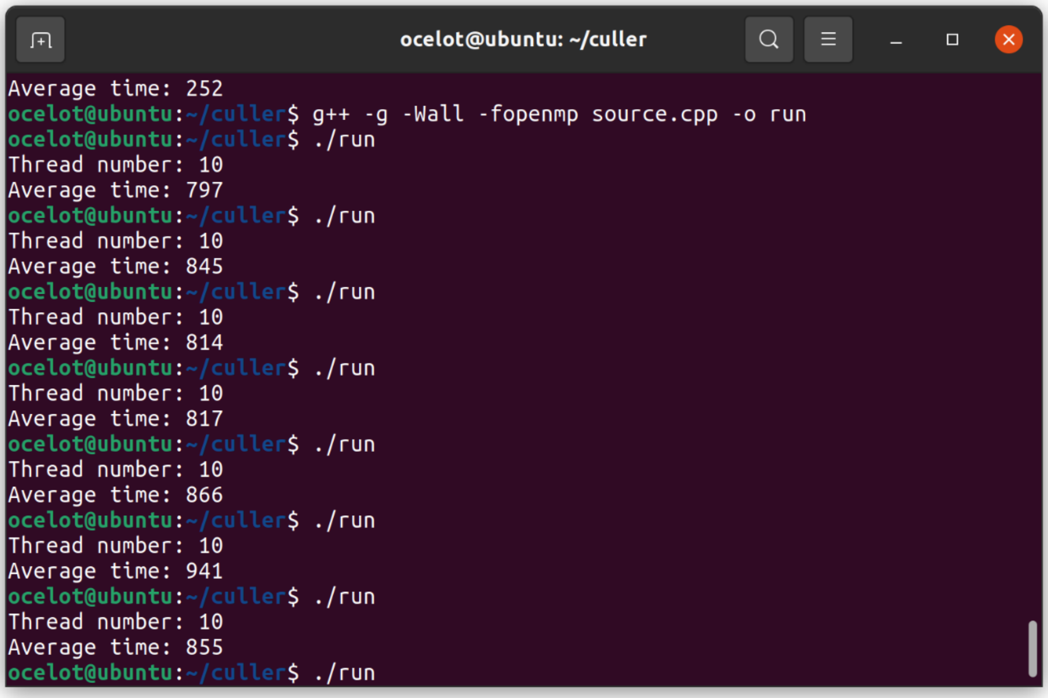


```
Average time: 372
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 190
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 547
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 359
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 520
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 212
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 78
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 206
ocelot@ubuntu:~/culler$
```

反而明显增加。


ocelot@ubuntu: ~/culler

```
Average time: 206
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
ocelot@ubuntu:~/culler$ ./run
Thread number: 8
Average time: 747
ocelot@ubuntu:~/culler$ ./run
Thread number: 8
Average time: 545
ocelot@ubuntu:~/culler$ ./run
Thread number: 8
Average time: 345
ocelot@ubuntu:~/culler$ ./run
Thread number: 8
Average time: 430
ocelot@ubuntu:~/culler$ ./run
Thread number: 8
Average time: 533
ocelot@ubuntu:~/culler$ ./run
Thread number: 8
Average time: 577
ocelot@ubuntu:~/culler$ ./run
Thread number: 8
Average time: 252
ocelot@ubuntu:~/culler$
```


ocelot@ubuntu: ~/culler

```
Average time: 252
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
ocelot@ubuntu:~/culler$ ./run
Thread number: 10
Average time: 797
ocelot@ubuntu:~/culler$ ./run
Thread number: 10
Average time: 845
ocelot@ubuntu:~/culler$ ./run
Thread number: 10
Average time: 814
ocelot@ubuntu:~/culler$ ./run
Thread number: 10
Average time: 817
ocelot@ubuntu:~/culler$ ./run
Thread number: 10
Average time: 866
ocelot@ubuntu:~/culler$ ./run
Thread number: 10
Average time: 941
ocelot@ubuntu:~/culler$ ./run
Thread number: 10
Average time: 855
ocelot@ubuntu:~/culler$ ./run
```

可见在同样的线程数量下，每一次运行的时间都还是有较大波动，而且线程数量并不是越大越好。线程数量在八个以下时运行时间较短，线程数量在八个或者十个时运行时间反而明显增加。

2. 不同的调度方案

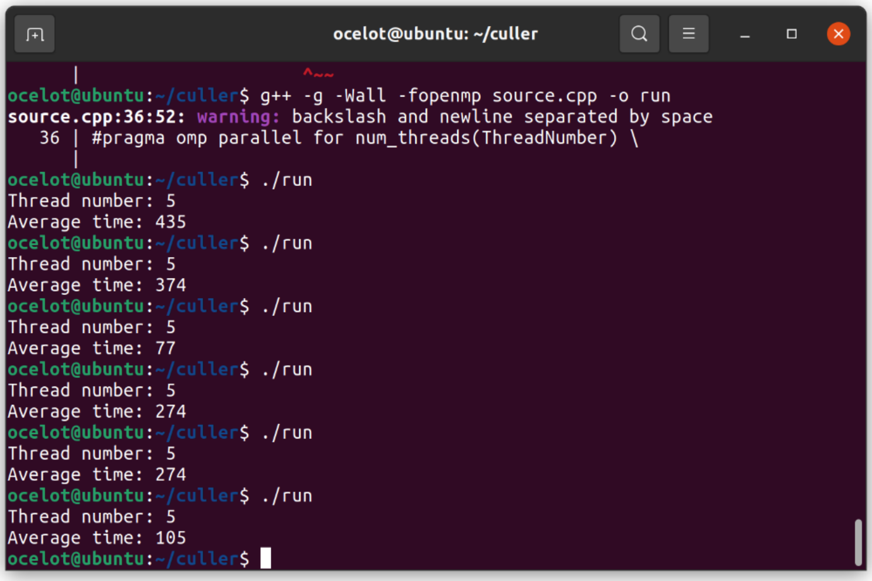
OpenMP 提供了比较方便的设定线程的调度方案的方法，即添加一个 schedule 子句到 parallel for 指令中，一般 schedule 子句有如下形式：

schedule(<type> [, <chunksize>])

以下讨论几种调度类型：

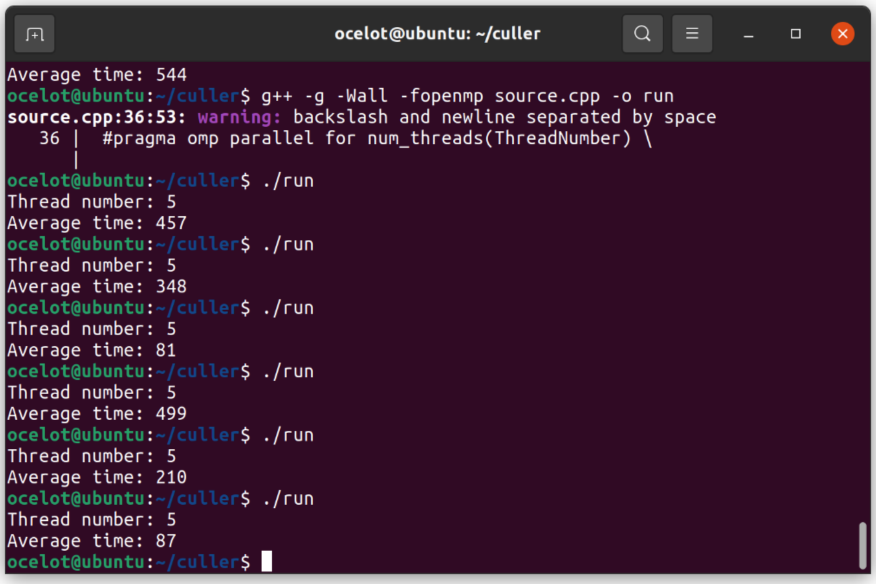
(1) Static 调度类型

a. schedule (static, 1)



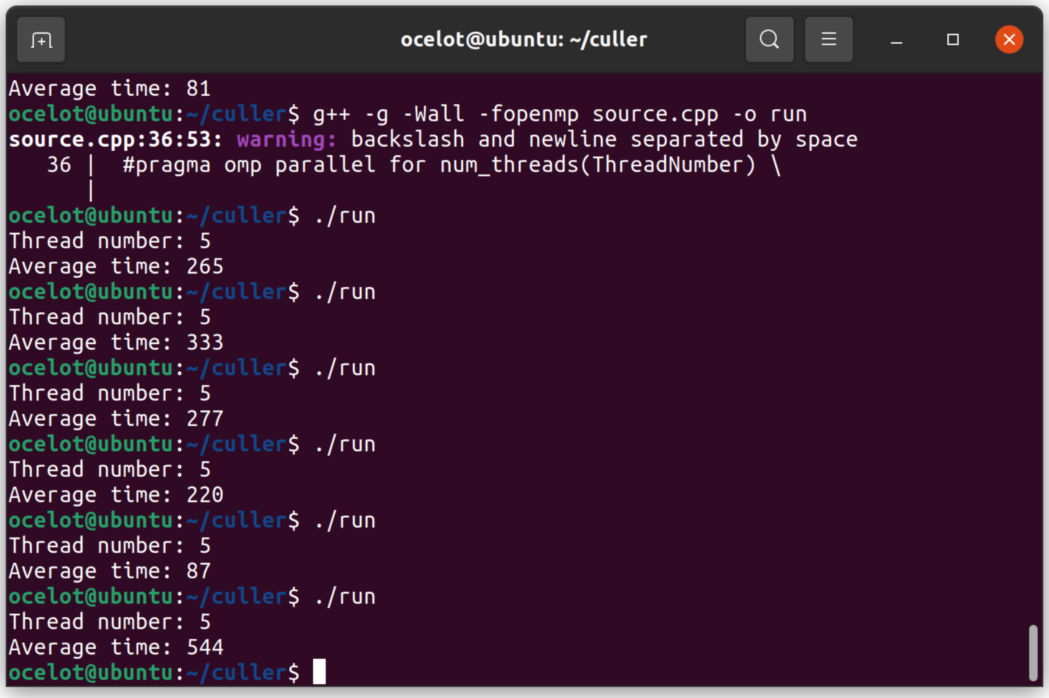
```
      |                  ^~~
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
source.cpp:36:52: warning: backslash and newline separated by space
   36 | #pragma omp parallel for num_threads(ThreadNumber) \
      |
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 435
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 374
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 77
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 274
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 274
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 105
ocelot@ubuntu:~/culler$
```

b. schedule (static, 10)



```
Average time: 544
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
source.cpp:36:53: warning: backslash and newline separated by space
   36 |  #pragma omp parallel for num_threads(ThreadNumber) \
      |
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 457
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 348
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 81
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 499
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 210
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 87
ocelot@ubuntu:~/culler$
```

c. schedule (static, 200)

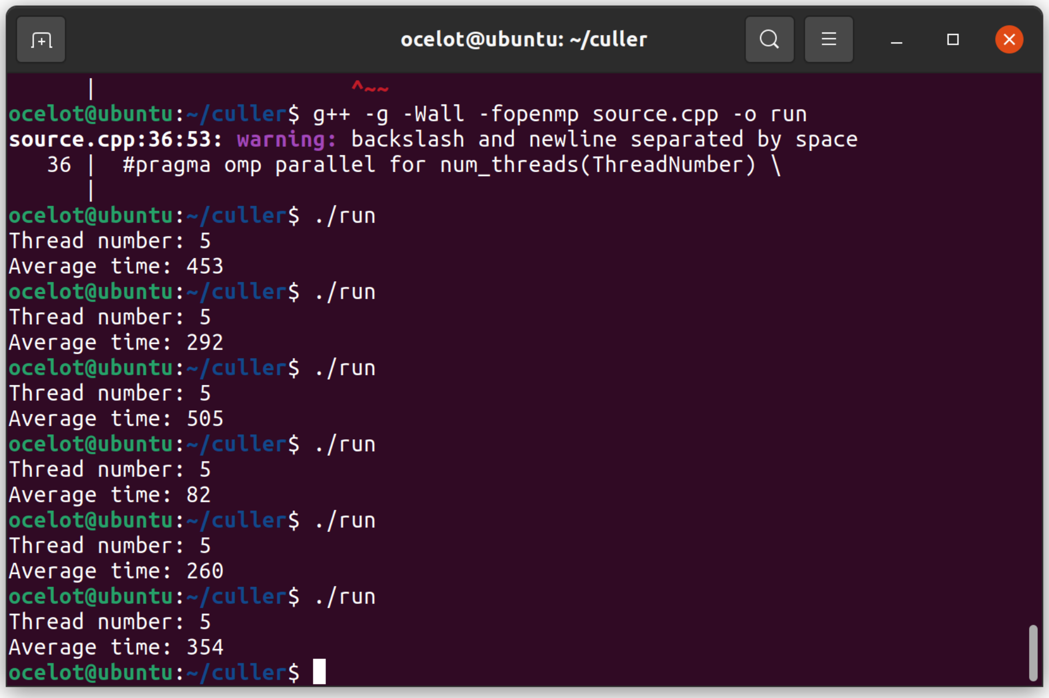


```
Average time: 81
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
source.cpp:36:53: warning: backslash and newline separated by space
   36 |   #pragma omp parallel for num_threads(ThreadNumber) \
      |
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 265
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 333
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 277
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 220
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 87
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 544
ocelot@ubuntu:~/culler$ 
```

可以看到在 static 调度方案下，设置不同 chunksize 后运算时间波动依然很大，总体来看将 chunksize 设置得较小比较好。

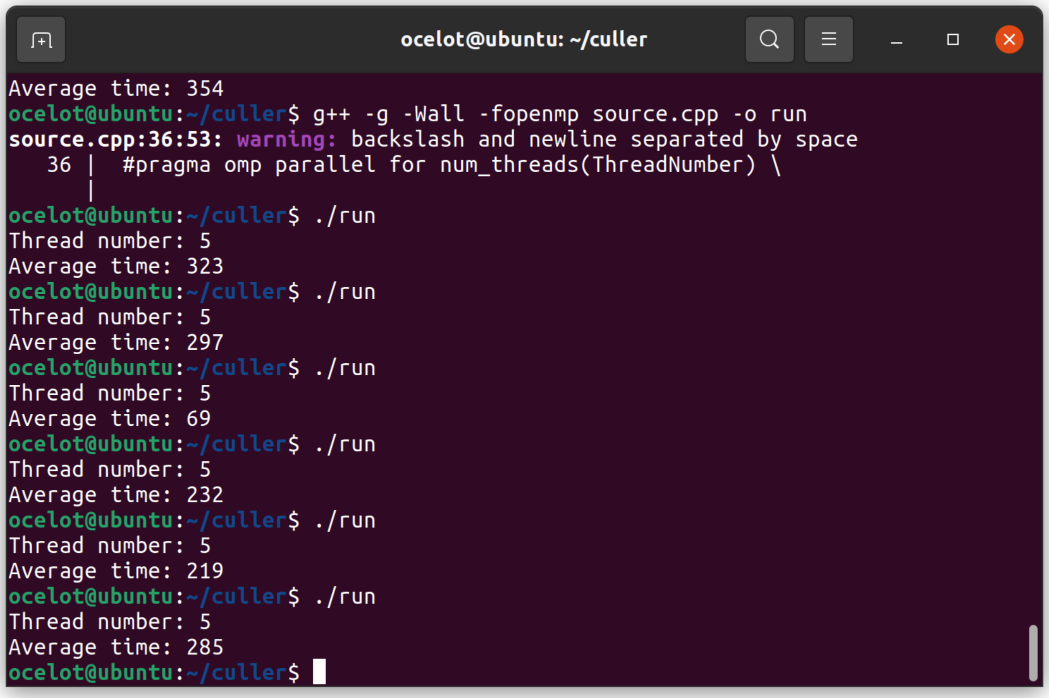
(2) dynamic 调度

a. schedule (dynamic, 1)



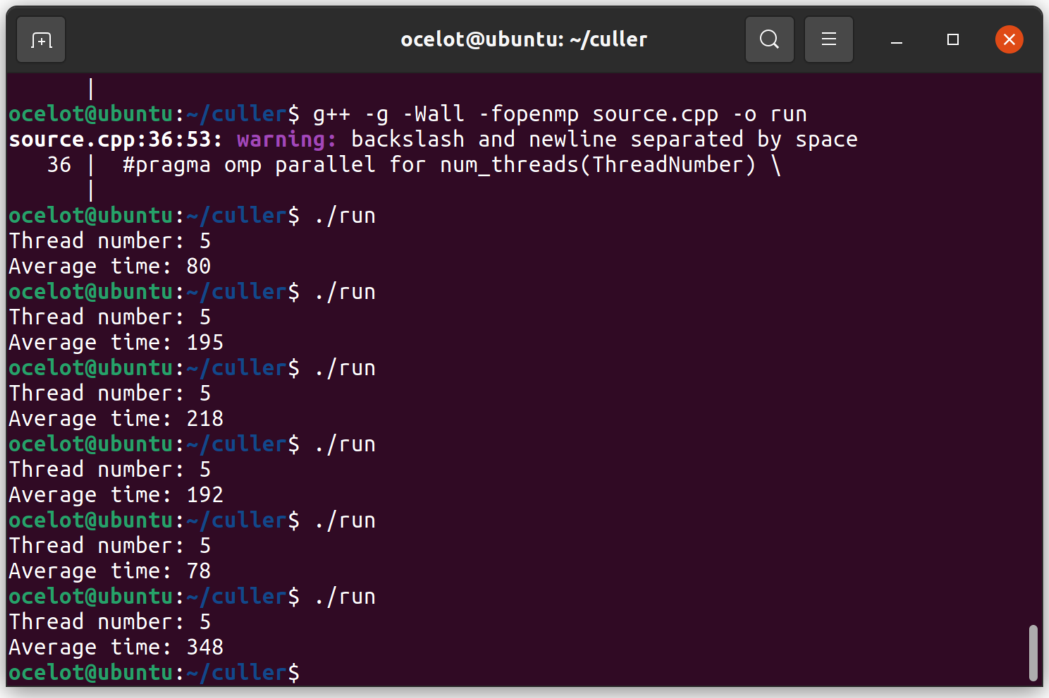
```
      |                          ^~~
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
source.cpp:36:53: warning: backslash and newline separated by space
   36 |   #pragma omp parallel for num_threads(ThreadNumber) \
      |
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 453
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 292
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 505
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 82
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 260
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 354
ocelot@ubuntu:~/culler$ 
```

b. schedule (dynamic, 10)



```
Average time: 354
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
source.cpp:36:53: warning: backslash and newline separated by space
   36 |   #pragma omp parallel for num_threads(ThreadNumber) \
      |
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 323
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 297
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 69
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 232
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 219
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 285
ocelot@ubuntu:~/culler$
```

c. schedule (dynamic, 100)



```
      |
ocelot@ubuntu:~/culler$ g++ -g -Wall -fopenmp source.cpp -o run
source.cpp:36:53: warning: backslash and newline separated by space
   36 |   #pragma omp parallel for num_threads(ThreadNumber) \
      |
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 80
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 195
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 218
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 192
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 78
ocelot@ubuntu:~/culler$ ./run
Thread number: 5
Average time: 348
ocelot@ubuntu:~/culler$
```

可以看到，dynamic 调度方式下运行时间波动较小，当把调度方式设置为 dynamic, 100 时运行速度有了很大的提升。

四、实验总结

本次实验复习了 culler 并行程序设计方法中的各个步骤，并探究了各个阶段采用不同配置导致程序的性能差别，让我对并行程序设计有了进一步的了解。在做实验的过程中我也注意到程序的运行时间不仅仅是代码本身和机器就可以完全决定的，不同代码多跑几次的运行时间也会差别很大。