多核平台下的并行计算课程实验报告

姓名：李润泽

学号：201922301282

班级：2020级软件工程1班

1. **填写下表，给出所使用的软硬件环境参数**

|  |  |
| --- | --- |
| 操作系统版本 | ubuntu-20.04.1-desktop-amd64 |
| 并行编程模型 | openMP |
| 编译器版本 | gcc 7.5.0 |
| CPU型号、主频及核数 | Intel(R) Core(TM) i5-9300H CPU @ 2.40GHz 2.40 GHz 8核 |
| 内存型号，大小及主频 | 16G 2666Hz |

1. **填写下表，给出程序使用不同线程数目时的运行时间与加速比（也可以自己设计图表描述不同线程数时程序的运行时间及加速比）**

|  |  |  |
| --- | --- | --- |
| 线程数目 | 运行时间（秒） | 加速比 |
| 1 | 29.440664000 | 1.0 |
| 2 | 17.681733000 | 1.67 |
| 3 | 15.779900000 | 1.87 |
| 4 | 18.640029000 | 1.58 |
| 8 | 68.246178000 | 0.43 |
| …… | …… | …… |

1. **并行算法设计和优化思路**

1.对算法整体结构的优化，因为原程序中采用递归方法，递归的实现是通过调用函数本身，函数调用的时候，每次调用时要做地址保存，参数传递等，这是通过一个递归工作栈实现的。具体是每次调用函数本身要保存的内容包括：局部变量、形参、调用函数地址、返回值。那么，如果递归调用N次，就要分配N\*局部变量、N\*形参、N\*调用函数地址、N\*返回值。这势必是影响效率的。

所以重新编写该方法，使其结构从递归变成循环。

原方法经过-O3优化大概60s左右，经过重新编写变为30s左右

2.编译参数-O3优化

当优化标识被启用之后，gcc编译器将会试图在不改变程序语义的情况下改变程序的结构，以满足代码大小最小或运行速度更快等目标。考虑到-O2会牺牲部分编译速度，除了执行-O1所执行的所有优化之外，还会采用几乎所有的目标配置支持的优化算法，用以提高目标代码的运行速度。-O3除了执行-O2所有的优化之外，一般还会采取很多向量化算法，提高代码的并行执行程度，比如利用现代CPU中的流水线，Cache等。

经过查询资料了解到还有一种优化选项是-Ofast，但是-Ofast优化将不会严格遵循语言标准，除了启用所有的-O3优化选项之外，也会针对某些语言启用部分优化，-Ofast优化比之前的几个优化选项都要激进，可能会发生丢失精度造成坏点的情况，所以最后为了保证结果的正确性，我们采用了-O3优化选项。

3.自动向量化

自动向量化（automatic vectorization）是自动并行化（automatic parallelization）的一种特殊情况，它将一次处理一对的标量运算转换为一次并行处理多对的向量运算。因此向量化可以显着加速一些带循环的程序运算，尤其是在大型数据集上。根据arch信息，编译器优化的目标可以是Intel或AMD处理器中的SSE\*、AVX/AVX2或更高级的指令，或ARM处理器中的NEON指令。

  默认情况下，GCC、ICC或AOCC/LLVM编译器中都启用了部分自动向量化功能。这些指令受运行时检查保护，即运行时检查机器是否支持该指令集，如果不支持则使用其它的指令集。

4.循环调度

考虑到OpenMP 使用paralle for指令时，只是粗略的使用了块分割，因此采用schedule字句实现循环划分，轮流分配线程的工作。我们尝试了static、dynamic等类型，最后采用了效果最好的dynamic调度类型，使用了64个 chunksize。

5.对求切比雪夫距离的一点改进（不完善）

求切比雪夫距离和可以转化为求曼哈顿距离和

切比雪夫距离：

Dis（A,B）=max(|XA−XB|,|YA−YB|)

Dis（A,B）=max(|XA−XB|,|YA−YB|)

曼哈顿距离：

Dis(A,B)=|XA−XB|+|YA−YB|

Dis(A,B)=|XA−XB|+|YA−YB|

转化一：

先介绍一下去max的方法：max(X,Y)=(X+Y+|X−Y|)/2max(X,Y)=(X+Y+|X−Y|)/2

相应的，去绝对值的方法：|X−Y|=max(X−Y,Y−X)|X−Y|=max(X−Y,Y−X)

由此我们可以开始推导公式：

令X′=X+Y,Y′=X−YX′=X+Y,Y′=X−Y

max(|X′A−X′B|,|Y′A−Y′B|)

max(|XA′−XB′|,|YA′−YB′|)

=max(|(XA+YA)−(XB+YB)|,|(XA−YA)−(XB−YB)|)

=max(|(XA+YA)−(XB+YB)|,|(XA−YA)−(XB−YB)|)

=max((XA+YA)−(XB+YB),(XB+YB)−(XA+YA)),(XA−YA)−(XB−YB),(XB−YB)−(XA−YA))

=max((XA+YA)−(XB+YB),(XB+YB)−(XA+YA)),(XA−YA)−(XB−YB),(XB−YB)−(XA−YA))

=max((XA−XB)+(YA−YB),(XB−XA)+(YB−YA),(XA−XB)+(YB−YA),(XB−XA)+(YA−YB))

=max((XA−XB)+(YA−YB),(XB−XA)+(YB−YA),(XA−XB)+(YB−YA),(XB−XA)+(YA−YB))

=|XA−XB|+|YA−YB|

=|XA−XB|+|YA−YB|

由于X′=X+Y,Y′=X−YX′=X+Y,Y′=X−Y

同理可得X′+Y′=2X,X′−Y′=2YX′+Y′=2X,X′−Y′=2Y

于是切比雪夫距离可以转化为曼哈顿距离

但是此方法仅能适用于二维即k=2时的情况，更高维度的转换好像暂时还没有得到解答，所以很遗憾没有找到通用的解决方法

6.优化全部的无依赖for循环

方法：采用Open MP 并行模型，对for循环进行优化，主要使用的优化命令是：

# pragma omp parallel for num\_threads(thread\_count)

thread\_count 代表线程数

1. Unlikely和likely的一点小小的优化尝试

\_\_builtin\_expect是编译器内建函数，原型为long \_\_builtin\_expect (long exp, long c)。该函数并不会改变exp的值，但是可以对if-else分支或者if分支结构进行优化。likely代表if分支大概率会发生，unlikely代表if分支大概率不会发生。

<https://blog.csdn.net/weixin_44873133/article/details/107302688>

1. 对排序算法的一点小小的改进

适合并行处理的排序算法：一、快速排序；二、合并排序；三、双调排序。其中，快速排序可以轻松并行化，因为它具有分而治之的特性.单个就地分区操作难以并行化，但是一旦划分，列表的不同部分可以并行排序。

但是考虑循环次数过多，如果每次都进行并行化排序，前面几次效率会大大增加，可是当到了最后的循环，最大的一千个数据和最小的一千个数据基本保持不变，所以保留了原有的排序方法，只不过加了一个break，及时进行终止操作