程序设计实训 第三次作业

徐浩博 2020010108

编译运行

开发环境：Windows 10, 64 位操作系统

IDE：Microsoft Visual Studio 2019

编译运行方法：在 IDE 中直接编译运行

运行结果

- ijk[1]

N	第一次测试/ms	第二次测试/ms	第三次测试/ms	平均用时/ms[2]
64	7	6	8	7.0
128	64	59	59	60.7
256	466	456	458	460.0
512	3610	3615	3854	3693

- ikj

N	第一次测试/ms	第二次测试/ms	第三次测试/ms	平均用时/ms
64	9	9	9	9.0
128	74	78	76	76.0
256	945	907	934	918.7
512	8547	8638	8538	8574.3

- jik

N	第一次测试/ms	第二次测试/ms	第三次测试/ms	平均用时/ms
64	8	8	9	8.3
128	62	61	63	62.0
256	476	498	498	490.7
512	3746	3754	3948	3816.0

- jki

N	第一次测试/ms	第二次测试/ms	第三次测试/ms	平均用时/ms
64	22	20	20	20.7
128	164	168	167	166.3
256	1335	1331	1333	1333.0
512	20521	20618	20795	20644.7

[1] ijk 表示循环方式为
```
for(int i = 0; i <N; i++)
    for(int j = 0; j <N; j++)
        for(int k = 0; k <N; k++)
            c[i][j][k] = a[i][j][k] + b[i][j][k];
```

[2] 考虑到一次测试可能存在较大误差，故对于每种情况，均采用测试三次取平均值的方式获得运行时间。

❑ kij

N	第一次测试/ms	第二次测试/ms	第三次测试/ms	平均用时/ms
64	15	13	15	14.3
128	182	182	179	181.0
256	1409	1404	1431	1414.7
512	11814	11570	12087	11823.7

❑ kji

N	第一次测试/ms	第二次测试/ms	第三次测试/ms	平均用时/ms
64	20	20	20	20.0
128	191	197	190	192.7
256	1733	1734	1719	1728.7
512	21937	22101	21912	21983.3

结果分析

❑ 6 种循环方式运行时间对比

循环方式 / N	ijk	jik	ikj	kij	jki	kji
64	7.0	8.3	9.0	14.3	20.7	20.0
128	60.7	62.0	76.0	181.0	166.3	192.7
256	460.0	490.7	918.7	1414.7	1333.0	1728.7
512	3693	3816.0	8574.3	11823.7	20644.7	21983.3

考虑到由于 N 过小时，不同循环方案运行时间差别较小，可能存在一定误差。
查看计算机的 cache 大小，大约为 14MB 左右，约相当于 N=150 的三维数组，因此 N 过小时，cache 命中率对于程序运行效率的影响不显著。
基于以上两点，我们针对 N=512 的六种循环方式运行时间进行研究：

运行时间　ijk < jik < ikj < kij < jki < kji
运行效率　ijk > jik > ikj > kij > jki > kji

下面，我将针对此结果进行分析：
首先，我们考虑三重循环时

```
for(int x = 0; x <N; x++)
    for(int y = 0; y <N; y++)
        for(int z = 0; z <N; z++)
            c[x][y][z] = a[x][y][z] + b[x][y][z];
```

循环时(x,y,z)变化方式为

(0,0,0) (0,0,1) (0,0,2) … (0,0,N-1) -> (0,1,0) (0,1,1)… (0,1,N-1) ->…-> (0,N-1,0)… (0,N-1,N-1)

↓

(1,0,0) (1,0,1) (1,0,2) … (1,0,N-1) -> (1,1,0) (1,1,1)… (1,1,N-1) ->…-> (1,N-1,0)… (1,N-1,N-1)

↓

……

(N-1,0,0)… (N-1,0,N-1) -> (N-1,1,0)… (N-1,1,N-1) ->…-> (0,N-1,0)… (N-1,N-1,N-1)

我们可以看出，(x,y,z)的变化由最内层循环变量(z)引起的次数为(N-1)×N×N=N^3-N^2，中间一层循环变量(y)为(N-1)×N= N^2-N，最外层循环变量(x)为 N-1.

数组所占的储存空间在内存中是连续的。运算时，计算机需要将内存中的数据读入缓存中，然后再进行运算；而读入到缓存时，会将附近的数据一并读入。因此两个数据在储存空间内离得越近，就越可能同时被读入缓存中（称为 cache 命中率高），从而减少计算机从内存中读入数据的次数。对于数组 a[i][j][k]来说，离 a[i][j][k+1]最近，离 a[i][j+1][k]则隔着 N-1 个数，而 a[i+1][j][k]隔着的数大约是 N^2 量级的；因此三维数组末位变化时改变的距离最小，cache 命中率最高，首位变化时改变的距离最大，cache 命中率最低。

回到本题中，最内层循环变量 z 变化的次数约为 N^3 量级，远超其余两个变量，因此当 z 对应 a[i][j][k]末位的 k 时，cache 命中率高，则效率最高。此种情况对应循环 ijk 和 jik。效率中等的对应中间位 j，cache 命中率中等，对应循环 ikj 和 kij。效率最低的对应首位 i，cache 命中率中等，对应循环 jki 和 kji。综上，我们可以将循环效率初步排序为：

运行效率　ijk, jik > ikj, kij > jki, kji

下面，在中间一层循环变量 y 和最外层循环变量 x 中，最影响效率的是变化次数达 N^2 量级的 y。

1.在 ijk 和 jik 中，当 y 对应 j 时 cache 命中率中等，大于对应 i 时较低的 cache 命中率，因此效率 ijk > jik.

2.在 ikj 和 kij 中，当 y 对应 k 时 cache 命中率较高，大于对应 i 时较低的 cache 命中率，因此效率 ikj > kij.

3.在 jki 和 kji 中，当 y 对应 k 时 cache 命中率较高，大于对应 j 时中等的 cache 命中率，因此效率 jki > kji.

综上：运行效率　ijk > jik > ikj > kij > jki > kji

以上就是实验结果的理论分析。