

# 高性能并行计算第 2 次作业

姓名: 张子栋      学号: 2020317210101

代码地址: /home/2020317210101/work2

<https://github.com/Bluuur/MarkdownNotes/tree/main/高性能并行计算/Code2>

## 实验结果

均以多线程并行

1. 计算  $N$  维数据的最大值, 最小值, 平均值, 标准差

## 代码

```
1 //
2 // Created by zidongzh on 2022/9/26.
3 //
4
5 #include "stdio.h"
6 #include "math.h"
7 #include <omp.h>
8 #include <ntdef.h>
9 #include <profileapi.h>
10
11
12 double getMax(double array[], int len) {
13     double max = array[0];
14     int i;
15     for (i = 0; i < len; ++i) {
16         if (array[i] > max) {
17             max = array[i];
18         }
19     }
20     return max;
21 }
22
23 double getMin(double array[], int len) {
24     double min = array[0];
25     int i;
26     for (i = 0; i < len; ++i) {
27         if (array[i] <= min) {
28             min = array[i];
```

```

29     }
30 }
31 return min;
32 }
33
34 double getMean(double array[], int len) {
35     double sum = 0.0;
36     double mean = 0.0;
37     int i;
38     for (i = 0; i < len; ++i) {
39         sum += array[i];
40     }
41     mean = sum / len;
42     return mean;
43 }
44
45 double getSD(double array[], int len) {
46     double mean = getMean(array, len);
47     double SS = 0.0;
48     int i;
49     for (i = 0; i < len; ++i) {
50         SS += pow(array[i] - mean, 2);
51     }
52     double SD = sqrt(SS / (len - 1));
53     return SD;
54 }
55
56 int main() {
57     // Get the length of data
58     int i;
59     int j;
60
61     double run_time;
62     union _LARGE_INTEGER time_start;
63     union _LARGE_INTEGER time_over;
64     double dqFreq;
65     LARGE_INTEGER f;
66
67     QueryPerformanceFrequency(&f);
68     dqFreq = (double) f.QuadPart;
69
70 #pragma omp parallel for
71
72     // Mock data
73     for (int k = 0; k <= 1000; k += 50) {

```

```

74
75     double array[k];
76
77     for (int l = 0; l < k; ++l) {
78         array[l] = rand();
79     }
80
81     // tick
82     QueryPerformanceCounter(&time_start);
83
84     getMax(array, k);
85     getMin(array, k);
86     getMean(array, k);
87     getSD(array, k);
88
89     // tack
90     QueryPerformanceCounter(&time_over);
91
92     // Get time in us
93     run_time = 1000000 * (time_over.QuadPart -
time_start.QuadPart) / dqFreq;
94
95     printf("\ndata num:%d ,run_time:%fus\n", k, run_time);
96
97 }
98 }

```

## 结果

```

1 data num:100, run_time:8.200000us
2 data num:400, run_time:5.500000us
3 data num:500, run_time:14.200000us
4 data num:0, run_time:18.900000us
5 data num:200, run_time:22.300000us
6 data num:300, run_time:18.800000us
7 data num:600, run_time:8.100000us
8 data num:700, run_time:32.500000us
9 data num:950, run_time:16.400000us
10 data num:900, run_time:52.200000us
11 data num:800, run_time:55.800000us
12 data num:1000, run_time:60.600000us
13 data num:150, run_time:8.100000us
14 data num:450, run_time:29.400000us
15 data num:550, run_time:39.800000us
16 data num:50, run_time:4.600000us

```

```
17 data num:250, run_time:15.700000us
18 data num:350, run_time:19.800000us
19 data num:650, run_time:57.800000us
20 data num:750, run_time:50.900000us
21 data num:850, run_time:58.400000us
```

## 2. 计算 $N$ 维向量点乘

代码:

```
1  //
2  // Created by ZidongZh on 2022/9/26.
3  //
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  #include <profileapi.h>
9
10 int main() {
11
12     double run_time;
13     union _LARGE_INTEGER time_start;
14     union _LARGE_INTEGER time_over;
15     double dqFreq;
16     LARGE_INTEGER f;
17
18     QueryPerformanceFrequency(&f);
19     dqFreq = (double) f.QuadPart;
20
21     #pragma omp parallel for
22
23     for (int k = 0; k < 1000; k += 50) {
24
25         double array[k];
26
27         // Mock data
28         for (int l = 0; l < k; ++l) {
29             array[l] = rand();
30         }
31
32         // tick
33         QueryPerformanceCounter(&time_start);
34
35
```

```

36         // Initialize array
37         double array1[k];
38         double array2[k];
39
40         // Mock data
41         for (int i = 0; i < k; ++i) {
42             array1[i]=rand();
43         }
44         for (int i = 0; i < k; ++i) {
45             array2[i]=rand();
46         }
47
48         // Compute & Output
49         double result = 0;
50         for (int i = 0; i < k; ++i) {
51             result += (array1[i] * array2[i]);
52         }
53
54         // tack
55         QueryPerformanceCounter(&time_over);
56
57         // Get time in us
58         run_time = 1000000 * (time_over.QuadPart -
time_start.QuadPart) / dqFreq;
59
60         printf("\ndata num:%d ,run_time:%fus\n", k, run_time);
61
62     }
63
64     return 0;
65 }
66

```

## 结果

```

1 data num:100, run_time:3.500000us
2 data num:0, run_time:0.100000us
3 data num:50, run_time:1.400000us
4 data num:300, run_time:4.000000us
5 data num:400, run_time:14.800000us
6 data num:500, run_time:2.500000us
7 data num:850, run_time:9.600000us
8 data num:600, run_time:9.300000us
9 data num:700, run_time:17.900000us
10 data num:950, run_time:15.600000us

```

```
11 data num:900, run_time:28.100000us
12 data num:800, run_time:33.000000us
13 data num:150, run_time:5.500000us
14 data num:200, run_time:3.100000us
15 data num:350, run_time:9.100000us
16 data num:450, run_time:11.200000us
17 data num:550, run_time:14.200000us
18 data num:650, run_time:17.400000us
19 data num:750, run_time:28.200000us
20 data num:250, run_time:8.600000us
```

### 3. 计算 $N$ 维矩阵点乘

```
1 //
2 // Created by zidongzh on 2022/9/26.
3 //
4
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <ntdef.h>
8 #include <profileapi.h>
9
10 int main() {
11
12     double run_time;
13     union _LARGE_INTEGER time_start;
14     union _LARGE_INTEGER time_over;
15     double dqFreq;
16     LARGE_INTEGER f;
17
18     QueryPerformanceFrequency(&f);
19     dqFreq = (double) f.QuadPart;
20
21     #pragma omp parallel for
22
23     for (int k = 0; k < 100; k += 5) {
24
25         double array[k];
26
27         // Mock data
28         for (int l = 0; l < k; ++l) {
29             array[l] = rand();
30         }
31
32         // tick
```

```

33     QueryPerformanceCounter(&time_start);
34
35     // Initialize array
36     double matrix1[k][k];
37     double matrix2[k][k];
38     for (int i = 0; i < k; ++i) {
39         for (int j = 0; j < k; ++j) {
40             matrix1[i][j] = rand();
41         }
42     }
43     for (int i = 0; i < k; ++i) {
44         for (int j = 0; j < k; ++j) {
45             matrix2[i][j] = rand();
46         }
47     }
48
49     // Compute
50     double result = 0;
51     for (int i = 0; i < k; ++i) {
52         for (int j = 0; j < k; ++j) {
53             result += (matrix1[i][j] * matrix2[i][j]);
54         }
55     }
56
57     // tack
58     QueryPerformanceCounter(&time_over);
59
60     // Get time in us
61     run_time = 1000000 * (time_over.QuadPart -
time_start.QuadPart) / dqFreq;
62
63     printf("\ndata num:%d, run_time:%fus", k, run_time);
64
65 }
66 return 0;
67 }
68

```

## 结果

```

1 data num:10, run_time:0.200000us
2 data num:0, run_time:0.100000us
3 data num:5, run_time:1.300000us
4 data num:30, run_time:34.400000us
5 data num:40, run_time:111.000000us

```

```
6 data num:50, run_time:234.700000us
7 data num:70, run_time:292.600000us
8 data num:60, run_time:324.900000us
9 data num:80, run_time:500.400000us
10 data num:95, run_time:535.100000us
11 data num:90, run_time:577.800000us
12 data num:85, run_time:606.000000us
13 data num:15, run_time:6.800000us
14 data num:20, run_time:0.300000us
15 data num:35, run_time:47.100000us
16 data num:45, run_time:55.700000us
17 data num:55, run_time:83.000000us
18 data num:75, run_time:168.400000us
19 data num:65, run_time:144.400000us
20 data num:25, run_time:17.400000us
```

#### 4. 大量随机数冒泡排序

```
1 //
2 // Created by ZidongZh on 2022/9/26.
3 //
4
5 #include<stdio.h>
6 #include<stdlib.h>
7 #include <ntdef.h>
8 #include <profileapi.h>
9
10 void BubbleSort(int array[], int length) {
11     int i, j, temp;
12     for (i = 0; i < length - 1; i++) {
13         for (j = 0; j < length - i - 1; j++) {
14             if (array[j] > array[j + 1]) {
15                 temp = array[j + 1];
16                 array[j + 1] = array[j];
17                 array[j] = temp;
18             }
19         }
20     }
21 }
22
23 int main() {
24
25     double run_time;
26     union _LARGE_INTEGER time_start;
27     union _LARGE_INTEGER time_over;
```



```

28     double dqFreq;
29     LARGE_INTEGER f;
30
31     QueryPerformanceFrequency(&f);
32     dqFreq = (double) f.QuadPart;
33
34     #pragma omp parallel for
35
36     for (int k = 0; k < 1000; k += 50) {
37
38         double array[k];
39
40         // Mock data
41         for (int l = 0; l < k; ++l) {
42             array[l] = rand();
43         }
44
45         // tick
46         QueryPerformanceCounter(&time_start);
47
48         BubbleSort(array, k);
49
50         // tack
51         QueryPerformanceCounter(&time_over);
52
53         // Get time in us
54         run_time = 1000000 * (time_over.QuadPart -
time_start.QuadPart) / dqFreq;
55
56         printf("\ndata num:%d ,run_time:%fus\n", k, run_time);
57
58     }
59     return 0;
60
61 }
62

```

## 结果

```

1 data num:0, run_time:0.100000us
2 data num:100, run_time:2.300000us
3 data num:150, run_time:49.500000us
4 data num:300, run_time:23.700000us
5 data num:400, run_time:381.700000us
6 data num:500, run_time:402.000000us

```

```
7 data num:600, run_time:627.700000us
8 data num:700, run_time:879.800000us
9 data num:800, run_time:1417.500000us
10 data num:900, run_time:1704.900000us
11 data num:850, run_time:1737.000000us
12 data num:950, run_time:2012.600000us
13 data num:50, run_time:5.200000us
14 data num:200, run_time:18.100000us
15 data num:350, run_time:229.300000us
16 data num:450, run_time:328.100000us
17 data num:550, run_time:578.100000us
18 data num:650, run_time:876.000000us
19 data num:750, run_time:1055.400000us
20 data num:250, run_time:94.400000us
```