# 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

```c
//
// Created by ZidongZh on 2022/9/26.
//

#include "stdio.h"
#include "math.h"
#include <omp.h>
#include <ntdef.h>
#include <profileapi.h>


double getMax(double array[], int len) {
    double max = array[0];
    int i;
    for (i = 0; i < len; ++i) {
        if (array[i] > max) {
            max = array[i];
        }
    }
    return max;
}

double getMin(double array[], int len) {
    double min = array[0];
    int i;
    for (i = 0; i < len; ++i) {
        if (array[i] <= min) {
            min = array[i];
        }
    }
    return min;
}

double getMean(double array[], int len) {
    double sum = 0.0;
    double mean = 0.0;
    int i;
    for (i = 0; i < len; ++i) {
        sum += array[i];
    }
    mean = sum / len;
    return mean;
}

double getSD(double array[], int len) {
    double mean = getMean(array, len);
    double SS = 0.0;
    int i;
    for (i = 0; i < len; ++i) {
        SS += pow(array[i] - mean, 2);
    }
    double SD = sqrt(SS / (len - 1));
    return SD;
}

int main() {
    // Get the length of data
    int i;
    int j;

    double run_time;
    union _LARGE_INTEGER time_start;
    union _LARGE_INTEGER time_over;
    double dqFreq;
    LARGE_INTEGER f;

    QueryPerformanceFrequency(&f);
    dqFreq = (double) f.QuadPart;

#pragma omp parallel for

    // Mock data
    for (int k = 0; k <= 1000; k += 50) {

        double array[k];

        for (int l = 0; l < k; ++l) {
            array[l] = rand();
        }

        //  tick
        QueryPerformanceCounter(&time_start);

        getMax(array, k);
        getMin(array, k);
        getMean(array, k);
        getSD(array, k);

        // tack
        QueryPerformanceCounter(&time_over);

        // Get time in us
        run_time = 1000000 * (time_over.QuadPart - 
time_start.QuadPart) / dqFreq;

        printf("\ndata num:%d ,run_time:%fus\n", k, run_time);

    }
}
```

**结果**

```
data num:100, run_time:8.200000us
data num:400, run_time:5.500000us
data num:500, run_time:14.200000us
data num:0, run_time:18.900000us
data num:200, run_time:22.300000us
data num:300, run_time:18.800000us
data num:600, run_time:8.100000us
data num:700, run_time:32.500000us
data num:950, run_time:16.400000us
data num:900, run_time:52.200000us
data num:800, run_time:55.800000us
data num:1000, run_time:60.600000us
data num:150, run_time:8.100000us
data num:450, run_time:29.400000us
data num:550, run_time:39.800000us
data num:50, run_time:4.600000us
data num:250, run_time:15.700000us
data num:350, run_time:19.800000us
data num:650, run_time:57.800000us
data num:750, run_time:50.900000us
data num:850, run_time:58.400000us
```

2. 计算 $N$ 维向量点乘

**代码**:

```
//
// Created by ZidongZh on 2022/9/26.
//

#include <stdio.h>
#include <stdlib.h>

#include <profileapi.h>

int main() {

    double run_time;
    union _LARGE_INTEGER time_start;
    union _LARGE_INTEGER time_over;
    double dqFreq;
    LARGE_INTEGER f;

    QueryPerformanceFrequency(&f);
    dqFreq = (double) f.QuadPart;

#pragma omp parallel for

    for (int k = 0; k < 1000; k += 50) {

        double array[k];

        // Mock data
        for (int l = 0; l < k; ++l) {
            array[l] = rand();
        }

        //  tick
        QueryPerformanceCounter(&time_start);


        // Initialize array
        double array1[k];
        double array2[k];

        // Mock data
        for (int i = 0; i < k; ++i) {
            array1[i]=rand();
        }
        for (int i = 0; i < k; ++i) {
            array2[i]=rand();
        }

        // Compute & Output
        double result = 0;
        for (int i = 0; i < k; ++i) {
            result += (array1[i] * array2[i]);
        }

        // tack
        QueryPerformanceCounter(&time_over);

        // Get time in us
        run_time = 1000000 * (time_over.QuadPart - 
time_start.QuadPart) / dqFreq;

        printf("\ndata num:%d ,run_time:%fus\n", k, run_time);

    }

    return 0;
}

```

## 结果

```
data num:100, run_time:3.500000us
data num:0, run_time:0.100000us
data num:50, run_time:1.400000us
data num:300, run_time:4.000000us
data num:400, run_time:14.800000us
data num:500, run_time:2.500000us
data num:850, run_time:9.600000us
data num:600, run_time:9.300000us
data num:700, run_time:17.900000us
data num:950, run_time:15.600000us
data num:900, run_time:28.100000us
data num:800, run_time:33.000000us
data num:150, run_time:5.500000us
data num:200, run_time:3.100000us
data num:350, run_time:9.100000us
data num:450, run_time:11.200000us
data num:550, run_time:14.200000us
data num:650, run_time:17.400000us
data num:750, run_time:28.200000us
data num:250, run_time:8.600000us
```

3. 计算 $N$ 维矩阵点乘

```c
//
// Created by ZidongZh on 2022/9/26.
//

#include <stdio.h>
#include <stdlib.h>
#include <ntdef.h>
#include <profileapi.h>

int main() {

    double run_time;
    union _LARGE_INTEGER time_start;
    union _LARGE_INTEGER time_over;
    double dqFreq;
    LARGE_INTEGER f;

    QueryPerformanceFrequency(&f);
    dqFreq = (double) f.QuadPart;

#pragma omp parallel for

    for (int k = 0; k < 100; k += 5) {

        double array[k];

        // Mock data
        for (int l = 0; l < k; ++l) {
            array[l] = rand();
        }

        //  tick
        QueryPerformanceCounter(&time_start);

        // Initialize array
        double matrix1[k][k];
        double matrix2[k][k];
        for (int i = 0; i < k; ++i) {
            for (int j = 0; j < k; ++j) {
                matrix1[i][j] = rand();
            }
        }
        for (int i = 0; i < k; ++i) {
            for (int j = 0; j < k; ++j) {
                matrix2[i][j] = rand();
            }
        }

        // Compute
        double result = 0;
        for (int i = 0; i < k; ++i) {
            for (int j = 0; j < k; ++j) {
                result += (matrix1[i][j] * matrix2[i][j]);
            }
        }

        // tack
        QueryPerformanceCounter(&time_over);

        // Get time in us
        run_time = 1000000 * (time_over.QuadPart - 
time_start.QuadPart) / dqFreq;

        printf("\ndata num:%d, run_time:%fus", k, run_time);

    }
    return 0;
}

```

### 结果

```
data num:10, run_time:0.200000us
data num:0, run_time:0.100000us
data num:5, run_time:1.300000us
data num:30, run_time:34.400000us
data num:40, run_time:111.000000us
data num:50, run_time:234.700000us
data num:70, run_time:292.600000us
data num:60, run_time:324.900000us
data num:80, run_time:500.400000us
data num:95, run_time:535.100000us
data num:90, run_time:577.800000us
data num:85, run_time:606.000000us
data num:15, run_time:6.800000us
data num:20, run_time:0.300000us
data num:35, run_time:47.100000us
data num:45, run_time:55.700000us
data num:55, run_time:83.000000us
data num:75, run_time:168.400000us
data num:65, run_time:144.400000us
data num:25, run_time:17.400000us
```

4. 大量随机数冒泡排序

```c
//
// Created by ZidongZh on 2022/9/26.
//

#include<stdio.h>
#include<stdlib.h>
#include <ntdef.h>
#include <profileapi.h>

void BubbleSort(int array[], int length) {
    int i, j, temp;
    for (i = 0; i < length - 1; i++) {
        for (j = 0; j < length - i - 1; j++) {
            if (array[j] > array[j + 1]) {
                temp = array[j + 1];
                array[j + 1] = array[j];
                array[j] = temp;
            }
        }
    }
}

int main() {

    double run_time;
    union _LARGE_INTEGER time_start;
    union _LARGE_INTEGER time_over;
    double dqFreq;
    LARGE_INTEGER f;

    QueryPerformanceFrequency(&f);
    dqFreq = (double) f.QuadPart;

#pragma omp parallel for

    for (int k = 0; k < 1000; k += 50) {

        double array[k];

        // Mock data
        for (int l = 0; l < k; ++l) {
            array[l] = rand();
        }

        //  tick
        QueryPerformanceCounter(&time_start);

        BubbleSort(array, k);

        // tack
        QueryPerformanceCounter(&time_over);

        // Get time in us
        run_time = 1000000 * (time_over.QuadPart - time_start.QuadPart) / dqFreq;

        printf("\ndata num:%d ,run_time:%fus\n", k, run_time);

    }
    return 0;

}

```

**结果**

```
data num:0, run_time:0.100000us
data num:100, run_time:2.300000us
data num:150, run_time:49.500000us
data num:300, run_time:23.700000us
data num:400, run_time:381.700000us
data num:500, run_time:402.000000us
data num:600, run_time:627.700000us
data num:700, run_time:879.800000us
data num:800, run_time:1417.500000us
data num:900, run_time:1704.900000us
data num:850, run_time:1737.000000us
data num:950, run_time:2012.600000us
data num:50, run_time:5.200000us
data num:200, run_time:18.100000us
data num:350, run_time:229.300000us
data num:450, run_time:328.100000us
data num:550, run_time:578.100000us
data num:650, run_time:876.000000us
data num:750, run_time:1055.400000us
data num:250, run_time:94.400000us
```