中山大学计算机院本科生实验报告

（2024 学年春季学期）

课程名称：并行程序设计　　批改人：

实验	**3-Pthreads 并行矩阵乘法与数组求和**	专业（方向）	**计算机科学与技术**
学号	21307174	姓名	**刘俊杰**
Email	liujj255@mail2.edu.cn	完成日期	2024/4/10

1. 实验目的

1.1 并行矩阵乘法

使用 Pthreads 实现并行矩阵乘法，并通过实验分析其性能。

输入：m, n, k三个整数，每个整数的取值范围均为$[128, 2048]$

问题描述：随机生成$m \times n$的矩阵A及$n \times k$的矩阵B，并对这两个矩阵进行矩阵乘法运算，得到矩阵C.

输出：A, B, C三个矩阵，及矩阵计算所消耗的时间t。

要求：1. 使用 Pthread 创建多线程实现并行矩阵乘法，调整线程数量（1-16）及矩阵规模（128-2048），根据结果分析其并行性能（包括但不限于，时间、效率、可扩展性）。2. 选做：可分析不同数据及任务划分方式的影响。

1.2 并行数组求和

使用 Pthreads 实现并行数组求和，并通过实验分析其性能。

输入：整数n，取值范围为$[1\text{M}, 128\text{M}]$

问题描述：随机生成长度为n的整型数组A，计算其元素和$s = \sum_{i=1}^{n} A_i$。

输出：数组A，元素和s，及求和计算所消耗的时间t。

要求：1. 使用 Pthreads 实现并行数组求和，调整线程数量（1-16）及数组规模（1M, 128M），根据结果分析其并行性能（包括但不限于，时间、效率、可扩展性）。2. 选做：可分析不同聚合方式的影响。

2. 实验过程和核心代码

2.1 并行矩阵乘法

2.1.1 实验思路

①将矩阵 C 每一行的计算平均分给每一个线程。

②矩阵 A、B、C 作为共享的全局变量方便每一个线程访问和修改。

③由于每个线程负责修改的是矩阵 C 不同的行，所以不会发生线程竞争的问题，所以不需要使用锁、信号量等实现互斥。

2.1.2 实验过程

①首先接收运行的线程数目和矩阵的维度，为矩阵开创空间并使用随机数初始化矩阵。
(线程数目、矩阵维度、矩阵 A、B、C 均为全局共享变量)

②pthread_t threads[num_threads]，用来存储多个线程的标识符。

③pthread_create()创建新进程,每个线程执行 multiply_matrices()函数，每个线程的标识符作为参数传递。

④每个线程根据线程数目和矩阵维度计算出平均每个线程需要负责计算的行数,根据传入参数的线程标识符计算出每个线程负责的对应的行，每个线程再对对应的行实现计算。

⑤使用 pthread_join()等待每个线程执行完毕，主线程最后输出结果并释放空间。

2.1.3 核心代码

①首先接收运行的线程数目和矩阵的维度，为矩阵开创空间并使用随机数初始化矩阵。
(线程数目、矩阵维度、矩阵 A、B、C 均为全局共享变量)

```
    matrix_size = atoi(argv[1]);
    num_threads = atoi(argv[2]);



    // 设置随机数种子
    srand(time(NULL));

    // 分配矩阵内存
    A = allocate_matrix(matrix_size);
    B = allocate_matrix(matrix_size);
    C = allocate_matrix(matrix_size);

    // 生成矩阵随机值
    generate_random_matrix(A, matrix_size);
    generate_random_matrix(B, matrix_size);
```

②pthread_t threads[num_threads]，用来存储多个线程的标识符。

```
// 创建线程
pthread_t threads[num_threads];
```

③pthread_create()创建新进程,每个线程执行 multiply_matrices()函数，每个线程的标识符作为参数传递。

```
for (int i = 0; i < num_threads; i++) {
    pthread_create(&threads[i], NULL, multiply_matrices,(void*)i);
}
```

④每个线程根据线程数目和矩阵维度计算出平均每个线程需要负责计算的行数,根据传入参数的线程标识符计算出每个线程负责的对应的行,每个线程再对对应的行实现计算。

```c
// 在指定行范围内对矩阵A和矩阵B进行乘法运算的函数
void *multiply_matrices(void* rank) {
    int my_rank = (long)rank;
    int rows = matrix_size / num_threads;

    for (int i = my_rank * rows; i < (my_rank + 1)* rows; i++) {
        for (int j = 0; j < matrix_size; j++) {
            C[i][j] = 0;
            for (int k = 0; k < matrix_size; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return NULL;
}
```

⑤使用 pthread_join()等待每个线程执行完毕，主线程最后输出结果并释放空间。

```c
// 等待线程结束
for (int i = 0; i < num_threads; i++) {
    pthread_join(threads[i], NULL);
}

// 获取结束时间
gettimeofday(&end_time, NULL);

// 计算消耗的时间
double elapsed_time = (end_time.tv_sec - start_time.tv_sec) * 1000.0; // 将秒转换为毫秒
elapsed_time += (end_time.tv_usec - start_time.tv_usec) / 1000.0; // 将微秒转换为毫秒

// 输出矩阵计算所消耗的时间
printf("Computing time:  %.2f ms\n", elapsed_time);
printf("A:\n");
print_matrix(A);
printf("B\n");
print_matrix(B);
printf("C\n");
print_matrix(C);
// 释放矩阵内存
free_matrix(A, matrix_size);
free_matrix(B, matrix_size);
free_matrix(C, matrix_size);
```

2.2 并行数组求和

2.2.1 实验思路

①将数组设为共享全局变量方便每个线程访问。

②将数组的每个元素平均地分给每个线程计算出局部和。

③每个线程将计算的局部和加到全局的数组的和 total_sum,由于此时可能会发生线程竞争访问 total_sum，所以这里使用锁来实现互斥。

2.2.2 实验过程

①首先接收运行的线程数目和数组的大小，为数组开创空间并使用随机数初始化数组。
(线程数目、数组大小、结果 total_sum、数组均为全局共享变量)

②pthread_t threads[num_threads]，用来存储多个线程的标识符。

③pthread_create()创建新进程,每个线程执行 calculate_partial_sum()函数，每个线程的标识符作为参数传递。

④每个线程根据线程数目和数组长度计算出平均每个线程需要负责计算的元素的数目,根据传入参数的线程标识符计算出每个线程负责的对应的起始位置和结束位置，每个线程再对对应的局部元素实现计算，并使用锁将局部变量加入到 total_sum 中。

⑤使用 pthread_join()等待每个线程执行完毕，主线程最后输出结果并释放空间。

2.2.3 核心代码

①首先接收运行的线程数目和数组的大小，为数组开创空间并使用随机数初始化数组。
(线程数目、数组大小、结果 total_sum、数组均为全局共享变量)

```
array_size = atoll(argv[1]);
num_threads = atoi(argv[2]);

if (array_size < ARRAY_SIZE_MIN || array_size > ARRAY_SIZE_MAX) {
    printf("数组大小必须在%d和%d之间\n", ARRAY_SIZE_MIN, ARRAY_SIZE_MAX);
    return 1;
}

// 使用当前时间作为随机数种子
srand(time(NULL));
// 初始化互斥锁
pthread_mutex_init(&sum_mutex, NULL);

// 分配数组内存并生成随机数组
array = allocate_array(array_size);
generate_random_array(array, array_size);
```

②pthread_t threads[num_threads]，用来存储多个线程的标识符。

```
// 创建线程
pthread_t threads[num_threads];
```

③pthread_create()创建新进程,每个线程执行 calculate_partial_sum()函数，每个线程的标识符作为参数传递。

```
for (int i = 0; i < num_threads; i++) {
    pthread_create(&threads[i], NULL, calculate_partial_sum, (void*)i);
}
```

④每个线程根据线程数目和数组长度计算出平均每个线程需要负责计算的元素的数目,根据传入参数的线程标识符计算出每个线程负责的对应的起始位置和结束位置，每个线程再对对应的局部元素实现计算，并使用锁将局部变量加入到 total_sum 中。

```c
// 线程函数，计算数组部分的和
void *calculate_partial_sum(void *rank) {
    int my_rank = (long)rank;
    long long rows = array_size / num_threads;
    long long start = rows * my_rank;
    long long end = rows * (my_rank + 1);

    long long partial_sum = 0;
    for (long long i = start; i < end; i++) {
        partial_sum += array[i];
    }

    // 将局部和累加到总和中
    // 使用互斥锁保护临界区
    pthread_mutex_lock(&sum_mutex);
    total_sum += partial_sum;
    pthread_mutex_unlock(&sum_mutex);
    return NULL;
}
```

⑤使用 pthread_join()等待每个线程执行完毕，主线程最后输出结果并释放空间。

```c
// 获取结束时间
gettimeofday(&end_time, NULL);

// 计算消耗的时间
double elapsed_time = (end_time.tv_sec - start_time.tv_sec) * 1000.0; // 将秒转换为毫秒
elapsed_time += (end_time.tv_usec - start_time.tv_usec) / 1000.0; // 将微秒转换为毫秒

// 输出数组和及消耗的时间
printf("The size of array: %lld\n",array_size);
//for(int i = 0 ;i < array_size ;i++)printf("%d ",array[i]);
printf("\n");
printf("sum: %lld\n", total_sum);
printf("Computing time: %.2f ms\n", elapsed_time);

// 销毁互斥锁
pthread_mutex_destroy(&sum_mutex);


// 释放数组内存
free_array(array);
```

3. 实验结果

3.1 并行矩阵乘法

3.1.1 验证实验正确性

```
======== OUTPUT ========
Computing time: 0.34 ms
A:
1 0 5 5
8 9 7 8
0 3 6 2
4 4 0 2
B
1 4 6 8
6 3 3 8
7 0 8 4
5 9 7 1
C
61 49 81 33
151 131 187 172
70 27 71 50
38 46 50 66

======== REPORT ========
```

可以看到计算出来的矩阵 C 是正确的。

3.1.2 实验结果

①矩阵维度为 128

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 1 128
Computing time:  14.48 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 2 128
Computing time:  14.62 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 4 128
Computing time:  16.85 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 8 128
Computing time:  17.78 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 16 128
Computing time:  15.19 ms
```

②矩阵维度为 256

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 1 256
Computing time:  37.72 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 2 256
Computing time:  42.38 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 4 256
Computing time:  33.95 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 8 256
Computing time:  29.67 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 16 256
Computing time:  34.63 ms
```

③矩阵维度为 512

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 1 512
Computing time:  59.15 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 2 512
Computing time:  65.72 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 4 512
Computing time:  59.53 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 8 512
Computing time:  56.84 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 16 512
Computing time:  57.87 ms
```

④矩阵维度为 1024

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 1 1024
Computing time:  194.07 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 2 1024
Computing time:  187.67 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 4 1024
Computing time:  137.17 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 8 1024
Computing time:  122.75 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 16 1024
Computing time:  126.54 ms
```

⑤矩阵维度为 2048

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 1 2048
Computing time:  400.17 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 2 2048
Computing time:  389.47 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 4 2048
Computing time:  271.18 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 8 2048
Computing time:  238.64 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_1 16 2048
Computing time:  271.62 ms
```

线程数	矩阵规模				
	128	256	512	1024	2048
1	14.48ms	37.72ms	59.15ms	194.07ms	400.17ms
2	14.62.ms	42.38ms	65.72ms	187.67ms	389.47ms
4	16.85ms	33.95ms	59.53ms	137.17ms	271.18ms
8	17.78ms	29.7ms	56.84ms	122,75ms	238.64ms
16	15.19ms	34.63ms	57.87ms	126.54ms	217.62ms

3.2 并行数组求和

3.2.1 验证实验正确性

```
======== OUTPUT ========
The size of array: 16
5 6 5 0 7 6 0 3 1 0 0 2 1 4 7 0
sum: 47
Computing time: 0.38 ms

======== REPORT ========
```

可以看到计算出来的数组和是正确的。

3.2.2 实验结果

①数组长度为 1M

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 1048576 1
The size of array: 1048576

sum: 4717726
Computing time: 3.30 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 1048576 2
The size of array: 1048576

sum: 4716683
Computing time: 2.17 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 1048576 4
The size of array: 1048576

sum: 4717511
Computing time: 2.57 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 1048576 8
The size of array: 1048576

sum: 4718309
Computing time: 3.33 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 1048576 16
The size of array: 1048576

sum: 4715672
Computing time: 2.66 ms
```

②数组长度为 2M

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 2097152 1
The size of array: 2097152

sum: 9440953
Computing time: 7.12 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 2097152 2
The size of array: 2097152

sum: 9442066
Computing time: 5.26 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 2097152 4
The size of array: 2097152

sum: 9438802
Computing time: 4.96 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 2097152 8
The size of array: 2097152

sum: 9431897
Computing time: 3.71 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 2097152 16
The size of array: 2097152

sum: 9427646
Computing time: 12.65 ms
```

③数组长度为 4M

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 4194304 1
The size of array: 4194304

sum: 18870691
Computing time: 11.98 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 4194304 2
The size of array: 4194304

sum: 18878176
Computing time: 8.20 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 4194304 4
The size of array: 4194304

sum: 18873018
Computing time: 4.87 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 4194304 8
The size of array: 4194304

sum: 18868286
Computing time: 5.36 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 4194304 16
The size of array: 4194304

sum: 18864234
Computing time: 5.11 ms
```

④数组长度为 8M

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 8388608 1
The size of array: 8388608

sum: 37765747
Computing time: 18.82 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 8388608 2
The size of array: 8388608

sum: 37755127
Computing time: 10.53 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 8388608 4
The size of array: 8388608

sum: 37740248
Computing time: 16.02 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 8388608 8
The size of array: 8388608

sum: 37741921
Computing time: 8.51 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 8388608 16
The size of array: 8388608

sum: 37740382
Computing time: 10.29 ms
```

⑤数组长度为 16M

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 16777216 1
The size of array: 16777216

sum: 75494524
Computing time: 41.86 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 16777216 2
The size of array: 16777216

sum: 75504365
Computing time: 23.44 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 16777216 4
The size of array: 16777216

sum: 75490632
Computing time: 16.32 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 16777216 8
The size of array: 16777216

sum: 75480171
Computing time: 17.19 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 16777216 16
The size of array: 16777216

sum: 75481464
Computing time: 15.71 ms
```

⑥数组长度为 32M

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 33554432 1
The size of array: 33554432

sum: 150993173
Computing time: 77.43 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 33554432 2
The size of array: 33554432

sum: 150979150
Computing time: 49.39 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 33554432 4
The size of array: 33554432

sum: 151000215
Computing time: 31.69 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 33554432 8
The size of array: 33554432

sum: 150988670
Computing time: 32.95 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 33554432 16
The size of array: 33554432

sum: 150993711
Computing time: 32.33 ms
```

⑦数组长度为 64M

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 67108864 1
The size of array: 67108864

sum: 301987091
Computing time: 172.53 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 67108864 2
The size of array: 67108864

sum: 301960475
Computing time: 88.82 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 67108864 4
The size of array: 67108864

sum: 302015417
Computing time: 62.09 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 67108864 8
The size of array: 67108864

sum: 302002838
Computing time: 57.59 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 67108864 16
The size of array: 67108864

sum: 301950588
Computing time: 60.25 ms
```

⑧数组长度为 128M

```
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 134217728 1
The size of array: 134217728

sum: 604004462
Computing time: 315.21 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 134217728 2
The size of array: 134217728

sum: 604009626
Computing time: 177.19 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 134217728 4
The size of array: 134217728

sum: 603952743
Computing time: 115.00 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 134217728 8
The size of array: 134217728

sum: 603921910
Computing time: 117.35 ms
ljj@ljj-virtual-machine:~/parrallel_programming$ ./Lab3_2 134217728 16
The size of array: 134217728

sum: 603975830
Computing time: 103.53 ms
```

线程数	数组长度							
	1M	2M	4M	8M	16M	32M	64M	128M
1	3.30ms	7.12ms	11.98ms	10.82ms	41.86ms	77.43.ms	172.53ms	315.21md
2	2.17ms	5.26ms	8.20ms	10.53ms	23.44ms	49.39.ms	88.82ms	177.19ms
4	2.57ms	4.96ms	4.87ms	16.02ms	16.32ms	31.69ms	62.09ms	115.00ms
8	3.33ms	3.71ms	5.36ms	8.51ms	17.19ms	32.95ms	57.59ms	117.35ms

16	2.66ms	12.65ms	5.11ms	10.29ms	15.71ms	32.33ms	60.25ms	103.53ms

4. 实验感想

并行计算是一种重要的性能优化手段，能够充分利用多核处理器的性能。

在实现并行算法时，需要考虑线程创建和销毁的开销、任务分配的平衡性、数据共享与同步等问题。通过实验可以更好地理解并行计算的原理，并对系统性能进行优化。