**به نام خدا**

# گزارش تمرین عملی سوم سیستم های عامل

**امیررضا آذری**

**99101087**

**بخش اول**:

برای مولتی ترد کردن ضرب ماتریسی میتوان سیاست های مختلفی برای ترد ها در نظر گرفت. یک روش ساده تقسیم کردن عملیات چند ریسه ای ابتدا سطرها و سپس به ستون های ماتریس مقصد است. به این صورت که اگر تعداد ترد ها در pool-thread کمتر یا مساوی m باشد، فقط ضرب سطرها در ستون های ماتریس دوم به صورت مولتی ترد انجام شود و اگر تعداد تردها بیشتر باشد، ضرب ستون های ماتریس دوم نیز دسته بندی شود.

```c
void *multiply(void *arg) {
    int thread_id = *(int *) (arg);
    int start = thread_id * M;
    int end = (thread_id + 1) * M;
    for (int i = start / thread_pool_size;
         i < end / thread_pool_size; i++) {
        for (int j = 0; j < N; j++) {
            for (int k = 0; k < K; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    pthread_exit(NULL);
}
```

حال در بخش اول، ورودی داده شده در تمرین را با 2 اندازه thread pool متفاوت، انجام می دهیم. نتایج را در تصویر زیر مشاهده می نمایید:

```
amirreza@ubuntu:~/Documents/HW3_Practical$ ls
OSQ1_HW3_99101087.c  OSQ2_HW3_99101087.c
amirreza@ubuntu:~/Documents/HW3_Practical$ gcc OSQ1_HW3_99101087.c -o OSQ1_HW3_99101087 -lpthread
amirreza@ubuntu:~/Documents/HW3_Practical$ ./OSQ1_HW3_99101087
1
2 2
1 2
3 4
2 3
1 2 3
4 5 6
Result:
9 12 15
19 26 33
Size of the final matrix is: 2 * 3
amirreza@ubuntu:~/Documents/HW3_Practical$
```

```
amirreza@ubuntu:~/Documents/HW3_Practical$ gcc OSQ1_HW3_99101087.c -o OSQ1_HW3_99101087 -lpthread
amirreza@ubuntu:~/Documents/HW3_Practical$ ./OSQ1_HW3_99101087
7
2 2
1 2
3 4
2 3
1 2 3
4 5 6
Result:
9 12 15
19 26 33
Size of the final matrix is: 2 * 3
amirreza@ubuntu:~/Documents/HW3_Practical$
```

حال در بخش دوم تاثیر اندازه thread pool را بر زمان اجرا در نظر می گیریم، برای این کار 3 اندازه 1 و 2و 4 را تست می نماییم. نتایج را برای ورودی به اندازه 2000 مشاهده می کنید. همچنین قابل ذکر است که ماتریس ها تقریبا رندوم جنریت می شوند:

```
amirreza@ubuntu:~/Documents/HW3_Practical$ gcc OSQ2_HW3_99101087.c -o OSQ2_HW3_9
9101087 -lpthread
amirreza@ubuntu:~/Documents/HW3_Practical$ ./OSQ2_HW3_99101087
2000 2000 2000
Thread Pool Size:       1
Time of execution:      255112.3281 ms
-------------------------------------------------
Thread Pool Size:       2
Time of execution:      172785.1875 ms
-------------------------------------------------
Thread Pool Size:       4
Time of execution:      172134.9531 ms
-------------------------------------------------
amirreza@ubuntu:~/Documents/HW3_Practical$
```

```
amirreza@ubuntu:~/Documents/HW3_Practical$ gcc OSQ2_HW3_99101087.c -o OSQ2_HW3_99101087 -lpthread
amirreza@ubuntu:~/Documents/HW3_Practical$ ./OSQ2_HW3_99101087
2000 2000 2000
Thread Pool Size:       1
Time of execution:      77161.1328 ms
-------------------------------------------------
Thread Pool Size:       17
Time of execution:      49189.3711 ms
-------------------------------------------------
Thread Pool Size:       114
Time of execution:      44226.5156 ms
-------------------------------------------------
Thread Pool Size:       218
Time of execution:      42051.7109 ms
-------------------------------------------------
Thread Pool Size:       747
Time of execution:      41385.7227 ms
-------------------------------------------------
amirreza@ubuntu:~/Documents/HW3_Practical$
```

البته در صورت سوال نیز ذکر شده است که از pipeline استفاده بشود. گرچه کمی غیرمنطقی به نظر میرسد اما با استفاده از pipe حالات زیر را نیز پیاده سازی کرده ایم:

```c
int pipefd[2];

void* write_to_pipe(void* arg){
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            Index index;
            index.row = i;
            index.col = j;
            write(pipefd[1], &index, sizeof(Index));
        }
    }
}
 void* worker_thread(void* arg) {
    Index index;
    read(pipefd[0], &index, sizeof(Index));
    int row = index.row;
    int col = index.col;
    int sum = 0;
    for (int i = 0; i < k; i++)
        sum += matrix1[row][i] * matrix2[i][col];

    result[row][col] = sum;
    return NULL;
}
```