

## 题目一：

### CPU 版本

代码：

```
1. #include <iostream>
2. #include <stdlib.h>
3. #include <time.h>
4. #include<Windows.h>
5. #define ROWS 1024
6. #define COLS 1024
7. #include <time.h>
8. #ifdef WIN32
9. #include <windows.h>
10. #else
11. #include <sys/time.h>
12. #endif
13. #ifdef WIN32
14. int gettimeofday(struct timeval *tp, void *tzp)
15. {
16.     time_t clock;
17.     struct tm tm;
18.     SYSTEMTIME wtm;
19.     GetLocalTime(&wtm);
20.     tm.tm_year = wtm.wYear - 1900;
21.     tm.tm_mon = wtm.wMonth - 1;
22.     tm.tm_mday = wtm.wDay;
23.     tm.tm_hour = wtm.wHour;
24.     tm.tm_min = wtm.wMinute;
25.     tm.tm_sec = wtm.wSecond;
26.     tm.tm_isdst = -1;
27.     clock = mktime(&tm);
28.     tp->tv_sec = clock;
29.     tp->tv_usec = wtm.wMilliseconds * 1000;
30.     return (0);
31. }
32. #endif
33.
34.
35. using namespace std;
36.
37. void matrix_mul_cpu(float* M, float* N, float* P, int width)
38. {
```

```

39.     for(int i=0;i<width;i++)
40.         for(int j=0;j<width;j++)
41.         {
42.             float sum = 0.0;
43.             for(int k=0;k<width;k++)
44.             {
45.                 float a = M[i*width+k];
46.                 float b = N[k*width+j];
47.                 sum += a*b;
48.             }
49.             P[i*width+j] = sum;
50.         }
51. }
52.
53. int main()
54. {
55.     struct timeval start, end;
56.     gettimeofday( &start, NULL );
57.     float *A, *B, *C;
58.     int total_size = ROWS*COLS*sizeof(float);
59.     A = (float*)malloc(total_size);
60.     B = (float*)malloc(total_size);
61.     C = (float*)malloc(total_size);
62.
63.     //CPU 一维数组初始化
64.     for(int i=0;i<ROWS*COLS;i++)
65.     {
66.         A[i] = 80.0;
67.         B[i] = 20.0;
68.     }
69.
70.     matrix_mul_cpu(A, B, C, COLS);
71.
72.     gettimeofday( &end, NULL );
73.     int timeuse = 1000000 * ( end.tv_sec - start.tv_sec ) + end.tv_usec - start.tv_usec;
74.     cout << "total time is " << timeuse/1000 << "ms" <<endl;
75.     system("pause");
76.     return 0;
77. }

```

结果：用时 5549ms



```

e:\Code\cpp\test\test5.exe
total time is 5549ms
请按任意键继续. . .

```

## GPU 版本

代码:

```
1. #include "cuda_runtime.h"
2. #include "device_launch_parameters.h"
3.
4. #include <stdio.h>
5. #include <math.h>
6. #define Row 1024
7. #define Col 1024
8. #include <time.h>
9.
10. #ifdef WIN32
11. #include <windows.h>
12. #else
13. #include <sys/time.h>
14. #endif
15. #ifdef WIN32
16. int gettimeofday(struct timeval* tp, void* tzp)
17. {
18.     time_t clock;
19.     struct tm tm;
20.     SYSTEMTIME wtm;
21.     GetLocalTime(&wtm);
22.     tm.tm_year = wtm.wYear - 1900;
23.     tm.tm_mon = wtm.wMonth - 1;
24.     tm.tm_mday = wtm.wDay;
25.     tm.tm_hour = wtm.wHour;
26.     tm.tm_min = wtm.wMinute;
27.     tm.tm_sec = wtm.wSecond;
28.     tm.tm_isdst = -1;
29.     clock = mktime(&tm);
30.     tp->tv_sec = clock;
31.     tp->tv_usec = wtm.wMilliseconds * 1000;
32.     return (0);
33. }
34. #endif
35.
36. __global__ void matrix_mul_gpu(int* M, int* N, int* P, int width)
37. {
38.     int i = threadIdx.x + blockDim.x * blockIdx.x;
39.     int j = threadIdx.y + blockDim.y * blockIdx.y;
40.
41.     int sum = 0;
```

```

42.     for (int k = 0; k < width; k++)
43.     {
44.         int a = M[j * width + k];
45.         int b = N[k * width + i];
46.         sum += a * b;
47.     }
48.     P[j * width + i] = sum;
49. }
50.
51. int main()
52. {
53.     struct timeval start, end;
54.     gettimeofday(&start, NULL);
55.
56.     int* A = (int*)malloc(sizeof(int) * Row * Col);
57.     int* B = (int*)malloc(sizeof(int) * Row * Col);
58.     int* C = (int*)malloc(sizeof(int) * Row * Col);
59.     //malloc device memory
60.     int* d_dataA, * d_dataB, * d_dataC;
61.     cudaMalloc((void**)&d_dataA, sizeof(int) * Row * Col);
62.     cudaMalloc((void**)&d_dataB, sizeof(int) * Row * Col);
63.     cudaMalloc((void**)&d_dataC, sizeof(int) * Row * Col);
64.     //set value
65.     for (int i = 0; i < Row * Col; i++) {
66.         A[i] = 90;
67.         B[i] = 10;
68.     }
69.
70.     cudaMemcpy(d_dataA, A, sizeof(int) * Row * Col, cudaMemcpyHostToDevice);
71.     cudaMemcpy(d_dataB, B, sizeof(int) * Row * Col, cudaMemcpyHostToDevice);
72.     dim3 threadPerBlock(16, 16);
73.     dim3 blockNumber((Col + threadPerBlock.x - 1) / threadPerBlock.x, (Row +
        threadPerBlock.y - 1) / threadPerBlock.y);
74.     printf("Block(%d,%d)  Grid(%d,%d).\n", threadPerBlock.x, threadPerBlock
        .y, blockNumber.x, blockNumber.y);
75.     matrix_mul_gpu << <blockNumber, threadPerBlock >> > (d_dataA, d_dataB, d
        _dataC, Col);
76.     //拷贝计算数据-一级数据指针
77.     cudaMemcpy(C, d_dataC, sizeof(int) * Row * Col, cudaMemcpyDeviceToHost);
78.
79.     //释放内存

```

```

80.    free(A);
81.    free(B);
82.    free(C);
83.    cudaFree(d_dataA);
84.    cudaFree(d_dataB);
85.    cudaFree(d_dataC);
86.
87.    gettimeofday(&end, NULL);
88.    int timeuse = 1000000 * (end.tv_sec - start.tv_sec) + end.tv_usec - start.tv_usec;
89.    printf("total time is %d ms\n", timeuse / 1000);
90.
91.    return 0;
92. }

```

结果：用时 1267ms，显然比 CPU 快，而且是数倍的提升。

## 题目二：

一个 warp 中的线程必然在同一个 block 中，如果 block 所含线程数目不是 warp 大小的整数倍，那么多出的那些 thread 所在的 warp 中，会剩余一些 inactive 的 thread，也就是说，即使凑不够 warp 整数倍的 thread，硬件也会为 warp 凑足，只不过那些 thread 是 inactive 状态，需要注意的是，即使这部分 thread 是 inactive 的，也会消耗 SM 资源。因此在 block 的增大过程中当 block 所含线程数量不是 warp 的整数倍的时候调用时间将会明显增加。而当 Grid 增大的时候 warp 调用时间将会增加。