题目一：

CPU 版本

代码：

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include<Windows.h>
#define ROWS 1024
#define COLS 1024
#include <time.h>
#ifdef WIN32
#include <windows.h>
#else
#include <sys/time.h>
#endif
#ifdef WIN32
int gettimeofday(struct timeval *tp, void *tzp)
{
  time_t clock;
  struct tm tm;
  SYSTEMTIME wtm;
  GetLocalTime(&wtm);
  tm.tm_year   = wtm.wYear - 1900;
  tm.tm_mon    = wtm.wMonth - 1;
  tm.tm_mday   = wtm.wDay;
  tm.tm_hour   = wtm.wHour;
  tm.tm_min    = wtm.wMinute;
  tm.tm_sec    = wtm.wSecond;
  tm. tm_isdst  = -1;
  clock = mktime(&tm);
  tp->tv_sec = clock;
  tp->tv_usec = wtm.wMilliseconds * 1000;
  return (0);
}
#endif


using namespace std;

void matrix_mul_cpu(float* M, float* N, float* P, int width)
{
```

```
    for(int i=0;i<width;i++)
        for(int j=0;j<width;j++)
        {
            float sum = 0.0;
            for(int k=0;k<width;k++)
            {
                float a = M[i*width+k];
                float b = N[k*width+j];
                sum += a*b;
            }
            P[i*width+j] = sum;
        }
}

int main()
{
    struct timeval start, end;
    gettimeofday( &start, NULL );
    float *A, *B, *C;
    int total_size = ROWS*COLS*sizeof(float);
    A = (float*)malloc(total_size);
    B = (float*)malloc(total_size);
    C = (float*)malloc(total_size);

    //CPU 一维数组初始化
    for(int i=0;i<ROWS*COLS;i++)
    {
        A[i] = 80.0;
        B[i] = 20.0;
    }

    matrix_mul_cpu(A, B, C, COLS);

    gettimeofday( &end, NULL );
    int timeuse = 1000000 * ( end.tv_sec - start.tv_sec ) + end.tv_usec - st
art.tv_usec;
    cout << "total time is " << timeuse/1000 << "ms" <<endl;
    system("pause");
    return 0;
}
```

结果：用时 5549ms

GPU 版本

代码：

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <stdio.h>
#include <math.h>
#define Row  1024
#define Col 1024
#include <time.h>

#ifdef WIN32
#include <windows.h>
#else
#include <sys/time.h>
#endif
#ifdef WIN32
int gettimeofday(struct timeval* tp, void* tzp)
{
    time_t clock;
    struct tm tm;
    SYSTEMTIME wtm;
    GetLocalTime(&wtm);
    tm.tm_year = wtm.wYear - 1900;
    tm.tm_mon = wtm.wMonth - 1;
    tm.tm_mday = wtm.wDay;
    tm.tm_hour = wtm.wHour;
    tm.tm_min = wtm.wMinute;
    tm.tm_sec = wtm.wSecond;
    tm.tm_isdst = -1;
    clock = mktime(&tm);
    tp->tv_sec = clock;
    tp->tv_usec = wtm.wMilliseconds * 1000;
    return (0);
}
#endif

__global__ void matrix_mul_gpu(int* M, int* N, int* P, int width)
{
    int i = threadIdx.x + blockDim.x * blockIdx.x;
    int j = threadIdx.y + blockDim.y * blockIdx.y;

    int sum = 0;
```

```
    for (int k = 0; k < width; k++)
    {
        int a = M[j * width + k];
        int b = N[k * width + i];
        sum += a * b;
    }
    P[j * width + i] = sum;
}

int main()
{
    struct timeval start, end;
    gettimeofday(&start, NULL);

    int* A = (int*)malloc(sizeof(int) * Row * Col);
    int* B = (int*)malloc(sizeof(int) * Row * Col);
    int* C = (int*)malloc(sizeof(int) * Row * Col);
    //malloc device memory
    int* d_dataA, * d_dataB, * d_dataC;
    cudaMalloc((void**)&d_dataA, sizeof(int) * Row * Col);
    cudaMalloc((void**)&d_dataB, sizeof(int) * Row * Col);
    cudaMalloc((void**)&d_dataC, sizeof(int) * Row * Col);
    //set value
    for (int i = 0; i < Row * Col; i++) {
        A[i] = 90;
        B[i] = 10;
    }

    cudaMemcpy(d_dataA, A, sizeof(int) * Row * Col, cudaMemcpyHostToDevice);

    cudaMemcpy(d_dataB, B, sizeof(int) * Row * Col, cudaMemcpyHostToDevice);

    dim3 threadPerBlock(16, 16);
    dim3 blockNumber((Col + threadPerBlock.x - 1) / threadPerBlock.x, (Row + threadPerBlock.y - 1) / threadPerBlock.y);
    printf("Block(%d,%d)   Grid(%d,%d).\n", threadPerBlock.x, threadPerBlock.y, blockNumber.x, blockNumber.y);
    matrix_mul_gpu << <blockNumber, threadPerBlock >> > (d_dataA, d_dataB, d_dataC, Col);
    //拷贝计算数据-一级数据指针
    cudaMemcpy(C, d_dataC, sizeof(int) * Row * Col, cudaMemcpyDeviceToHost);

    //释放内存
```

```
    free(A);
    free(B);
    free(C);
    cudaFree(d_dataA);
    cudaFree(d_dataB);
    cudaFree(d_dataC);

    gettimeofday(&end, NULL);
    int timeuse = 1000000 * (end.tv_sec - start.tv_sec) + end.tv_usec - star
t.tv_usec;
    printf("total time is %d ms\n", timeuse / 1000);

    return 0;
}
```

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。