HW8：矩阵乘法的 CUDA 实现

一、 作业内容

在本次作业中，各位同学需参照课程所讲述的相关内容，利用 CUDA 来实现一个简单的矩阵乘法运算，包含如下三部分的内容，分别对应课件上所讲的三种方法。

1. Tile size 1×1 per thread

本部分对应课件 PPT 第 82 页中矩阵乘法的最简单实现，即每个线程负责输出矩阵 C 中一个 1×1 tile。该部分较为简单，请各位同学在 *matrix_multiplication.cu* 文件中的 **TODO** 部分补充对应的代码，得到判定正确的计算结果并记录 CUDA 的运行时间即可。

2. Increasing tile size per thread

本部分对应课件 PPT 第 84 页中的内容，即增大每个线程所负责的输出矩阵 C 中的 tile size 的大小以增大数据复用度。请各位同学在 *matrix_multiplication_method2.cu* 文件中的 **TODO** 部分补充对应的代码，得到判定正确的计算结果，并调整代码开头所定义的 TILE_SIZE 的大小，记录 TILE_SIZE=1, 2, 4, 8, …, 64 情况下的 CUDA 运行时间。

各位同学可以将运行时间与方法 1 进行对比，并尝试解释 CUDA 运行时间随 TILE_SIZE 的调整所带来的变化。各位同学可能会发现一些现象，可以根据你学习课程以来对体系架构的理解尝试给出简单的解释，例如结合 cache、访存等因素。

3. Optimization using shared memory

本部分对应课件 PPT 第 91 页中的内容，但请注意，作业要求的具体实现方式与课件中所讲述的并不相同。在本次作业中，每个线程块负责计算输出矩阵 C 中 BLOCK_SIZE ×BLOCK_SIZE 大小的 tile，其包含 BLOCK_SIZE×BLOCK_SIZE 个线程，每个线程负责计算输出矩阵 C 的 tile 中的**一个**元素。在每次循环中，需预先加载矩阵 A 与 B 的大小均为 BLOCK_SIZE×BLOCK_SIZE 的子矩阵 Asub 与 Bsub 至 shared memory 中，用于计算部分和并随循环所累积。每个线程均需加载 Asub 与 Bsub 中的其中一个元素。之后各个线程从 shared memory 中读取计算各自所负责的 C 矩阵中的一个输出元素所需的数据并进行乘累加运算。

请各位同学在 *matrix_multiplication_opt.cu* 文件中的 **TODO** 部分补充对应的代码，得到判定正确的计算结果，并调整代码开头所定义的 BLOCK_SIZE 的大小，记录 BLOCK_SIZE=1, 2, 4, 8, …, 32 情况下的 CUDA 运行时间。（继续增大 BLOCK_SIZE 会超出目前 GPU 平台每个线程块所能支持的最大 1024 个线程）

在 *matrix_multiplication_opt.cu* 代码中预先定义好了一些函数供同学们使用，这可以避免较为繁琐的地址索引运算。

二、 相关说明

请注意，课件的代码中矩阵 B 是以转置方式存储的，在本次作业中，所有矩阵均以 row-major 方式存储，未进行转置。此外，课件代码中出现的 lda, ldb, ldc 参数在本次作业中也未涉及，请各位同学注意即可。在作业代码中，数据类型 *Matrix* 为结构体，其定义可参见 *matrix_multiplication.h*。

请各位同学在自己的服务器上新建目录，并将助教所提供的 9 个代码文件上传至该目录下（或直接将 *MatrixMultiplication_hw* 文件夹上传）进行运行。如前所述，需要各位同学补充的源代码文件为 *matrix_multiplication.cu*、*matrix_multiplication_method2.cu* 以及 *matrix_multiplication_opt.cu*，其余文件为一些预定义内容以及判别计算结果是否正确的代码，各位同学在补充代码时如遇到问题可对应参考阅读。

每次修改完.cu 文件中的代码后，各位同学需首先运行 *make clean* 命令将之前运行产生的结果文件清除，注意助教令 *make clean* 命令可同时清除这 3 种方法所可能输出的所有结果文件，运行时遇到 rm: cannot remove 'xxx': No such file or directory 报错属正常结果。之后即可编译 CUDA 代码并进行运算。上述 3 种实现方法所对应的运行命令如下：

1. 首先运行命令 *make cuda*
   之后运行命令 *./matrix_multiplication* （输入 *./* 后使用 Tab 键可自动填充）
2. 首先运行命令 *make cuda_method2*
   之后运行命令 *./matrix_multiplication_method2*
3. 首先运行命令 *make cuda_opt*
   之后运行命令 *./matrix_multiplication_opt*

由于每次运行时 CUDA 的运行时间可能会有略微差别，各位同学在运行完 *make cuda/ cuda_method2/ cuda_opt* 命令后可连续多次运行对应的 *./matrix_*xxx 命令，例如可连续运行 5 次并取平均运行时间（如遇到偏离均值较大的运行时间可剔除后再取平均）。一个成功的运行示例如下图所示：

```
[tsinghuaee286@c4130-007 MatrixMultiplication_hw]$ make cuda
nvcc -c matrix_multiplication.cu
nvcc -c matrix_multiplication_host.cc util.o
nvcc -c util.cc
nvcc -o matrix_multiplication matrix_multiplication.o matrix_multiplication_host.o util.o
[tsinghuaee286@c4130-007 MatrixMultiplication_hw]$ ./matrix_multiplication
CUDA Elapsed time: 3.456256 ms
Success!![tsinghuaee286@c4130-007 MatrixMultiplication_hw]$ ./matrix_multiplication
CUDA Elapsed time: 3.416608 ms
Success!![tsinghuaee286@c4130-007 MatrixMultiplication_hw]$ ./matrix_multiplication
CUDA Elapsed time: 3.418688 ms
Success!![tsinghuaee286@c4130-007 MatrixMultiplication_hw]$ ./matrix_multiplication
CUDA Elapsed time: 3.441408 ms
Success!![tsinghuaee286@c4130-007 MatrixMultiplication_hw]$ ./matrix_multiplication
CUDA Elapsed time: 3.461760 ms
Success!![tsinghuaee286@c4130-007 MatrixMultiplication_hw]$ make clean
rm *.o matrix_multiplication matrix_multiplication_method2 matrix_multiplication_opt
rm: cannot remove 'matrix_multiplication_method2': No such file or directory
rm: cannot remove 'matrix_multiplication_opt': No such file or directory
make: *** [clean] Error 1
[tsinghuaee286@c4130-007 MatrixMultiplication_hw]$ make cuda_method2
```

助教提供的代码可将大家编写并利用 CUDA 运算所得的矩阵乘法结果与调用 CPU 运行所得的正确结果进行对比，各位同学首先要保证自己的运算结果正确，之后再记录对应的 CUDA 运行时间。

在本次作业中，各位同学仅需补充实现矩阵乘法的函数中的代码，其余涉及 CUDA 操作的代码助教已经帮大家给出，大家如果有兴趣可以阅读并查阅相关资料学习了解。其中各个.cu 文件中较为核心的一段 CUDA 代码与启动 CUDA 核心相关，以 *matrix_multiplication.cu* 为例，如下图中的 3 行代码所示。各位同学仅补充完成了实现矩阵乘法运算的函数 *void d_matrix_multiplication(Matrix C, Matrix A, Matrix B)*中的相关代码，使每个线程负责输出矩阵 C 中一个 1×1 tile，如下代码便是定义了每个线程块的尺寸（即包含多少个线程），以及线程网格的尺寸，并使函数 *d_matrix_multiplication* 及其参数加载到 CUDA 核心中，使得相关线程能够以 SIMD 的方式并行计算得到完整的输出矩阵 C。

```
// Launch CUDA Kernel
dim3 blockDim(8, 8);
dim3 gridDim(d_C.width / blockDim.x, d_C.height / blockDim.y);
d_matrix_multiplication<<<gridDim, blockDim>>>(d_C, d_A, d_B);
```

各位同学在提交作业时需提交一份文档，除了相关结果的记录及分析外，请各位同学将 3 种方法成功运行后的某次运行截图贴在报告里，方便助教查验。除此之外，请各位同学将 *matrix_multiplication.cu*、*matrix_multiplication_method2.cu* 以及 *matrix_multiplication_opt.cu* 这 3 份源代码文件一同打包提交。

关于服务器的使用，请各位同学参见《服务器使用说明》。值得注意的是，各位同学每次登录服务器后首先需要运行命令 *module add cuda91/toolkit/9.1.85* 来配置服务器的 CUDA 环境，请各位同学仔细阅读《服务器使用说明》中的相关内容。