PUC-Rio – Departamento de Informática
Ciência da Computação
Introdução à Arquitetura de Computadores
Prof.: Alexandre Meslin

Trabalho 2 – 2025-1

Instruções Gerais

Leia com atenção o enunciado do trabalho e as instruções para a entrega. Em caso de dúvidas, não invente. Pergunte!

- O trabalho deve ser entregue até a data marcada no EaD.
- Trabalhos entregues com atraso perderão um ponto por dia (ou fração) de atraso.
- Trabalhos que não compilem (isto é, que não produzam um executável) não serão considerados! Ou seja, receberão grau zero.
- Os trabalhos podem ser feitos em grupos de no máximo dois alunos.
- Alguns grupos poderão ser chamados para apresentações orais / demonstrações dos trabalhos entregues.
- Importante: Somente ***UM*** integrante do grupo deve fazer a carga dos arquivos fontes no EAD.

Objetivo

O objetivo deste trabalho é implementar a operação do produto de matrizes a partir da implementação da versão otimizada do produto de matrizes. Nenhuma mudança deve ser realizada no módulo `matrix_lib_test.c` do programa base já implementado e testado. Apenas as funções `matrix_matrix_mult` e `scalar_matrix_mult` do módulo `matrix_lib.c` devem sofrer alterações.

O programa de teste obtém o tempo antes da chamada de uma função da biblioteca e depois do retorno da função para calcular os tempos parciais da execução de cada função. Todas as tomadas de tempo e impressão das medidas de tempo devem ser realizadas na função `main` do programa de teste.

Parte I:

Aprimorar o módulo escrito utilizando a plataforma CUDA da Nvidia, chamado `matrix_lib.c`, implementado em sala de aula, com a utilização de processamento. As duas funções de operações aritméticas com matrizes estão descritas a seguir.

Crie um módulo escrito em linguagem C, chamado `matrix_lib.c`, que implemente duas funções para fazer operações aritméticas com matrizes, conforme descrito abaixo.

a. Função `int scalar_matrix_mult(float scalar_value, struct matrix *matrix, struct matrix *result)`

   Essa função recebe um valor escalar e uma matriz como argumentos de entrada e calcula o produto do valor escalar pela matriz e o armazena na matriz recebida como terceiro parâmetro. Em caso de sucesso, a função deve retornar o valor 0. Em caso de erro, a função deve retornar o código de erro correspondente. Esta função irá executar no host e chamar uma função no device.

b. Função `int matrix_matrix_mult(struct matrix *matrixA, struct matrix * matrixB, struct matrix * matrixC)`

Essa função recebe três matrizes como argumentos de entrada e calcula o valor do produto da matriz A pela matriz B *utilizando o algoritmo otimizado apresentado em aula*. O resultado da operação deve ser armazenado na matriz C. Em caso de sucesso, a função deve retornar o valor 0. Em caso de erro, a função deve retornar o código de erro correspondente. Esta função irá executar no host e chamar uma função no device.

c. O tipo estruturado `matrix` é definido da seguinte forma:

```
typedef struct {
     unsigned long int rows;
     unsigned long int cols;
     float *values;
} matrix;
```

Onde:

- `rows` = número de linhas da matriz (múltiplo de 8)
- `cols` = número de colunas da matriz (múltiplo de 8)
- `values` = sequência de linhas da matriz (`rows*cols` elementos)

d. Ambas funções devem chamar funções no *device* para realizar a computação necessária. O módulo principal irá criar duas variáveis globais `threadsPerBlock` e `blocksPerGrid`, representando respectivamente o número de *threads* por bloco e o número de blocos para ser utilizado na chamada das funções que irão executar no *device*.

Obs.: o número de threads é múltiplo ou divisível pelo número de linhas ou colunas das matrizes.

Parte II:

Use o programa em linguagem C, chamado `matrix_lib_test.c`, que implementa um código para testar a biblioteca `matrix_lib.c`. Esse programa recebe um valor escalar `float`, a dimensão da primeira matriz (A), a dimensão da segunda matriz (B) e o nome de quatro arquivos binários de `floats` na linha de comando de execução. O programa inicializa as duas matrizes (A e B) respectivamente a partir dos dois primeiros arquivos binários de `floats`.

A função `scalar_matrix_mult` é chamada com os seguintes argumentos:

- o valor escalar fornecido
- a primeira matriz (A)
- a primeira matriz resultado não inicializada

A função `main` armazena o resultado em um arquivo binário usando o nome do terceiro arquivo de `floats`.

Depois, a função `matrix_matrix_mult` é chamada com os seguintes argumentos:

- a primeira matriz (A)
- a segunda matriz (B)
- a terceira matriz (C)
- Novamente, a função `main` armazena o resultado (retornado na matriz C) em um arquivo binário com o nome do quarto arquivo de `floats`.

Exemplo de linha de comando:

```
$ matrix_lib_test -s 5.0 -r 8 -c 16 -C 24 -m floats1.dat -M floats2.dat -o result1.dat -O result2.dat -t 256 -g 4096
```

Onde,

- 5.0 é o valor escalar que multiplicará a primeira matriz;
- 8 é o número de linhas da primeira matriz;
- 16 é o número de colunas da primeira matriz e o número de linhas da segunda matriz;
- 24 é o número de colunas da segunda matriz;
- floats1.dat é o nome do arquivo de `floats` que será usado para carregar a primeira matriz;
- floats2.dat é o nome do arquivo de `floats` que será usado para carregar a segunda matriz;
- result1.dat é o nome do arquivo de `floats` onde o primeiro resultado será armazenado;
- result2.dat é o nome do arquivo de `floats` onde o segundo resultado será armazenado;
- 256 é a quantidade de threads por bloco;
- 4096 é a quantidade de blocos por grid.

A função principal cronometra o tempo de execução das funções `scalar_matrix_mult` e `matrix_matrix_mult`. Para marcar o início e o final do tempo em cada uma das situações, usamos a função padrão `clock` disponível em `<time.h>`. Essa função retorna o tempo aproximado de processador usado pelo programa. Para calcular a diferença de tempo (delta) entre duas marcas de tempo `t0` e `t1`, usamos a macro `timedifference_msec`, definida no módulo `matrix_lib.h`, fornecido.

Entregável:

Você deve entregar um relatório informado o que funcionou e o que não funcionou no seu programa. Esse relatório deve conter também um ou mais gráficos comparando essa implementação usando a GPU Nvidia para realizar tanto a multiplicação entre matrizes quanto a multiplicação de uma matriz com um escalar com:

- Para o cálculo envolvendo matriz e escalar:
  - A função que realiza a computação indexando o vetor, percorrendo o vetor coluna por coluna (como visto em sala de aula, sem uso de instruções vetoriais)
  - A função que realiza a computação indexando o vetor, percorrendo o vetor linha por linha (como visto em sala de aula, sem uso de instruções vetoriais)
  - A função que realiza a computação incrementando o vetor (como visto em sala de aula)
  - A função usando threads no host
- Para o cálculo envolvendo duas matrizes:
  - A função que multiplica as duas matrizes usando o algoritmo convencional (como visto em sala de aula, sem uso de instruções vetoriais)
  - A função que multiplica as duas matrizes usando o algoritmo que as percorre estritamente linha por linha (como visto em sala de aula, sem uso de instruções vetoriais)
  - A função usando threads no host

Realize testes com diversos tamanhos de matrizes, desde matrizes muito pequenas até matrizes realmente enormes, do tamanho máximo que a máquina de testes permitir. Mas sempre com matrizes com tamanhos múltiplos de 8.

Além do relatório, você deve entregar também o arquivo fonte `matrix_lib.c`. Lembre-se que você não modificou os arquivos `matrix_lib_test.c` e `matrix_lib.h`, logo, esses arquivos não precisam e nem devem ser entregues.

Observação 1:

Todas as práticas da disciplina serão executadas no ambiente Linux. Este trabalho deve ser compilado e executado no ambiente Linux. Para fazer a compilação do programa no Linux, deve-se usar o NVCC, com os seguintes argumentos:

```
$ nvcc -o matrix_lib_test matrix_lib_test.cu matrix_lib.cu
```

Onde,

- `matrix_lib_test` = nome do programa executável.
- `matrix_lib_test.cu` = nome do programa fonte que tem a função main().
- `matrix_lib.cu` = nome do programa fonte do módulo de funções de matrizes.

Uma máquina virtual padrão VMware com o sistema Linux está disponível na área de download do site da Equipe de Suporte do DI, em:

URL: `http://suporte.inf.puc-rio.br/download/vms/VMCCPP-FC23-64-DI-PUC-Rio-V1.0.zip`

Para executar a máquina virtual, basta baixar o VMware Workstation Player gratuitamente a partir do site da VMware.

URL: `https://www.vmware.com/br/products/workstation-player/workstation-player-evaluation.html`

Observação 2:

Apenas o programa fonte `matrix_lib.c` deve ser carregado no site de EAD da disciplina até o prazo de entrega.

Observação 3:

Crie e documente no seu relatório os códigos de erro que você usou na sua biblioteca.

Observação 4:

Documente o seu código no estilo Javadoc (Doxygen).