Multiplicação de Matrizes ladrilhada sequencial e paralelo(cuda e openmp).

Victor N. Rebli
Universidade Federal do Espirito Santo

***Resumo* — Este trabalho apresenta os resultados obtidos na implementação de multiplicação de matriezes utilizando a técnica de ladrilhamento em sequencial e paralelo(cuda e openmp) com um enfoque na comparação do desempenho que foi obtido na multiplicação paralela com a sequencial.**

I. Introdução

No campo da ciência da computação, operações matriciais possuem um custo computacional elevado, sendo sua complexidade medida em O(n^3), sendo assim um problema que pode se tornar dificil de lidar com técnicas de multiplicação sequencial. Sendo a multiplicação de matrizes utilizado frequentemente em diversas áreas, a utilização de técnicas paralelas permite aumentar drasticamente a performance no tempo necessário para a multiplicação de matrizes, principalmente matrizes com dimensões grandes.
Aliado a isso, uma forma de aumentar ainda mais a performance da multiplicação de matrizes é fazer melhor uso da memória cache da CPU(no caso da multiplicação sequencial e openmp) ou do GPU(no caso da multiplicação paralela cuda), e a melhor forma de realizar isso é utilizando um conceito chamado ladrilhamento(tiled). A memória compartilhada(cache) é mais rápida que a memória global, contudo seu uso deve ficar restrito a colocação de dados que serão utilizados mais de uma vez, e é esse justamente o caso de multiplicação de matrizes, pois se for possível colocar trechos das matrizes que estão realizando as operações na memória compartilhada, o tempo necessário para a realização da multiplicação será drasticamente reduzido. Como a multiplicação de matrizes baseia-se na adição de multiplicação de produtos, não importa a ordem que as somas são feitas. Nesse caso, o ladrilhamento divide as matrizes em bloco menores e as multiplicações são feitas em submatrizes menores, permitindo que os dados das submatrizes tenham tamanho suficiente para se encaixar na memória cache.
Nsse caso, em ambientes de memória compartilhada é possível obter um grande ganho de desempenho nesses tipícos problemas envolvendo operações matriciais.

A figura 1 abaixo exemplifica o funcionamento de uma matriz ladrilhada.

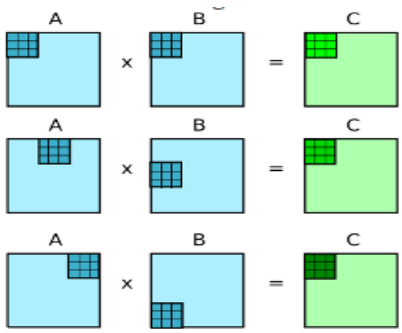


Figura 1 – Matriz ladrilhada

II. Metodologia

A implementação das multiplicações das matrizes ladrilhada foi realizada de maneira sequencial e em pararelo usando CUDA e OpenMP em precisão simples(float) e dupla(double), e foi executada nas seguintes configurações:

1) em precisão simples(float) foi utilizada matrizes 512X512 e ladrilho 64X64 e 1024X2014 e ladrilhado 64X64
2) em precisão dupla(double) foi utilizado uma matriz 1024X1024 e ladrilho 32X32.

A escolha do ladrilho de tamanho 64X64 para precisão simples e 32x32 deve-se as configurações da GPU da máquina onde foi realizado os experimentos com CUDA, cuja intenção era maximizar o tamanho da matriz e do ladrilhamento para performance máxima. Abaixo segue algumas configurações da GPU.

1)Total amount of shared memory per block: 49152 bytes
2)Maximum number of threads per multiprocessor: 2048
3) Maximum number of threads per block: 1024

Nessas configurações um ladrilho de 64X64 para precisão simples e 32X32 para precisão dupla permitiria a utilização máxima da memória compartilhada nos Sms(multiprocessadores).

Para cada configuração de tamanho de matriz e ladrilhamento foi realizado a multiplicação 1,100 e 1000 vezes e obtidos as métricas para serem computadas.

Foi realizado 3 rodadas para cada um dessas configurações e foi computada a média do tempo de execução para cada configuração, por exemplo foi executada 100 vezes a multiplicação da matriz 1024X1024 e ladrilho 64X64 para cuda, openmp e sequencial, obtido o tempo total de execução para cada um desses, sendo executado 3 rodadas dessa mesma configuração e tirada a média de execução para computar o speedup.

III. RESULTADOS

Os resultados obtidos na multiplicação de matrizes mostram uma superioridade dos métodos paralelos na multiplicação de matrizes, em especial com a utilização de CUDA, como era de se esperar.

O tempo de execução de Cuda e OpenMp superaram o tempo necessário para execução sequencial.
Abaixo segue a figura 2, onde demonstra o speedup entre a execução cuda e sequencial para matriz 512X512 e ladrilho 64X64.

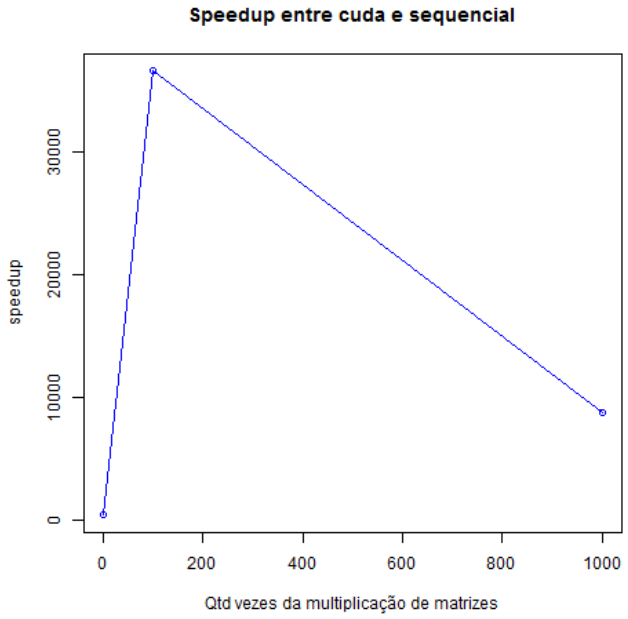


Figura 2 – 512X512 – tiled 64X64

Agora, na figura 3, o speedup entre a execução cuda e sequencial para matriz 1024X1024 e ladrilho 64x64

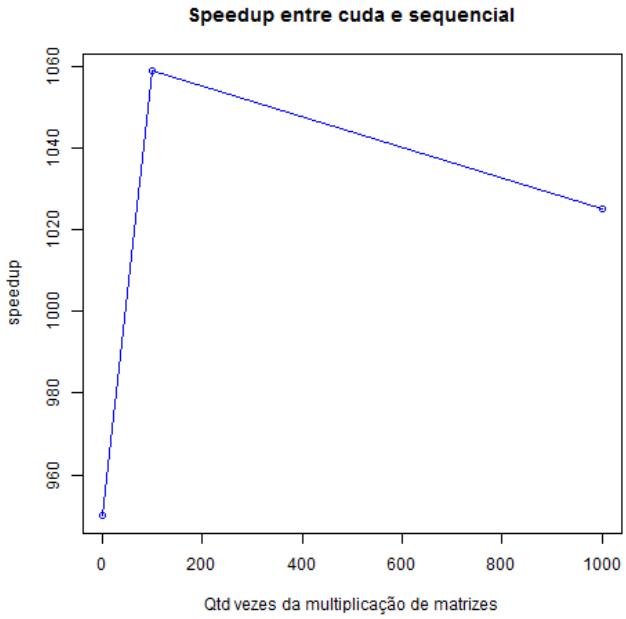


Figura 3 – Matriz 1024X1024 – tiled 64X64

Agora, na figura 4 a comparação entre a execução openmp e sequencial.

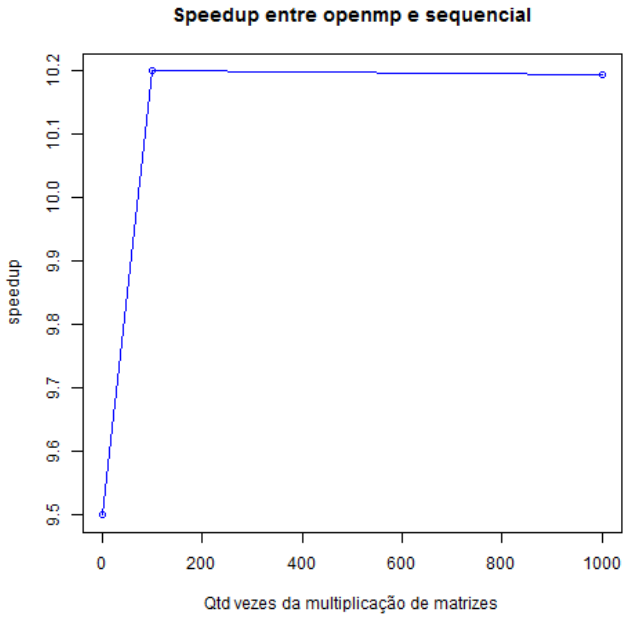
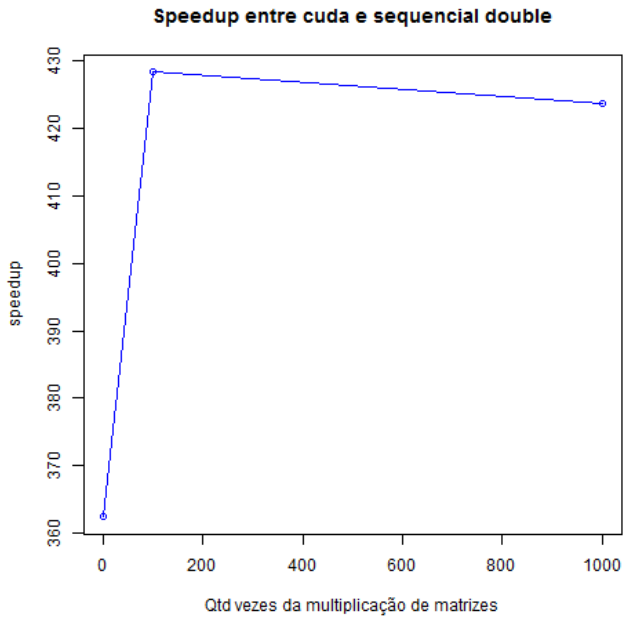


Figura 4 - Matriz 1024X1024 – tiled 64X64

E por último, na figura 5, o speedup entrre a execução cuda e sequencial em precisão dupla(double).

Figura 5 – Matriz 1024X1024 – tiled 32



Portanto, é possível constatar como era esperado a superioridade dos métodos pararelos frente a multiplicação de matrizes.