[[1]](#footnote-1)

Trabalho 1 de Arquitetura de computadores

Matheus de Araújo Nogueira

*Resumo*— Esse relatório traz os resultados do primeiro trabalho de arquitetura de computadores, no qual a atividade consistia em comparar o tempo de execução da multiplicação de matrizes em *CUDA* e *OpenMP*.

# Introdução

O trabalho pede a implementação de um programa de multiplicação de matrizes ladrilhada, comparando o tempo de execução sequencial e em paralelo utilizando CUDA e OpenMP em precisão simples(*float*) e dupla(*double*). Esse tipo de trabalho demostra que a habilidade de programar em paralelo é extremamente essencial para os dias atuais. Pois os processadores modernos estagnaram a velocidade de processamento e investiram em aumentar a quantidade de processadores em um único computador.

Esse relatório esta organizado em duas sessões, a primeira descreve a metodologia utilizada para a execução desse trabalho. Na segunda sessão apresento os resultados desse trabalho.

# Metodologia

Para executar essa comparação, foi criado dois programas, um para processamento em CPU utilizando o OpenMP e outro em GPU utilizando CUDA. O programa em CPU foi desenvolvido em C++ e compilado com o parâmetro **–O3** para otimizar e sem desenrolar ***loops***. Já o programa para GPU, foi aproveitado o código de exemplo de multiplicação de matriz que a própria NVIDIA disponibiliza.

O tipo de paralelismo utilizado para esse tipo de aplicação é o SIMD (*Single Instruction, Multiple Data*), nele uma única instrução atua em vários dados.

O computador utilizado foi do próprio laboratório, o LCAD54, com as seguintes especificações.

|  |  |
| --- | --- |
| Especificações do processador | |
| Modelo | Intel Core i7-4770 3.40 GHz |
| Quantidade de núcleos | 4 |
| Quantidade de threads | 8 |
| Cache | 8 Megabytes |
| RAM | 16 Gigabytes |

|  |  |
| --- | --- |
| Especiações da placa gráfica | |
| Modelo | Tesla K40c |
| VRAM | 12 Gigabytes |
| Frequência | 0.745 GHz |
| Quantidade de WARPS | 32 |
| CUDA cores | 2880 |
| Max de threads por SMP | 2048 |
| Max de threads por bloco | 1024 |

Durante o desenvolvimento desse trabalho foi definido alguns tamanhos múltiplos de 32 de matrizes e dos ladrilhos para CPU e GPU.

|  |  |
| --- | --- |
| Parâmetros para a CPU e GPU | |
| Tamanho da matriz | Tamanho do ladrilho |
| - | 128 |
| 256 | 256 |
| 512 | 512 |
| 1024 | 1024 |
| 2048 | 2048 |
| 4096 | 4096 |
| 8192 | 8192 |
| 16384 | 16384 |

A escolha desses valores se deu ao fato de tanto o cache da CPU e da CPU são múltiplos de 32. Assim, pude melhor utilizar eles.

Na experimentação de dados com dupla precisão, o tamanho máximo da matriz na GPU e na CPU foi 16.384, pois acima desse valor, as matrizes não poderiam ser alocadas em memória.

# Resultados

Comparando o tempo de execução em CPU e GPU nas maiores matrizes instancias, chegamos à conclusão que a GPU saiu bem melhor. Dado o fato de possuir uma arquitetura que favoreça esse tipo de problema.

Como mostra a figura 1.

Figura 1: Melhores tempos da matriz 16384 X 138684

Analisando o tamanho de ladrilho na GPU, obtivemos ganhos de tempo em ladrilhos que foram maiores ou iguais que 256x256. Esse ganho se deu ao fato de melhor utilizar o cache.

Como mostra a figura 2.

Figura 2: Mostra os melhores tamanhos de matrizes e ladrilhos na GPU em float e double.

# Conclusão

Com esses experimentos fica claro que o uso de GPU’s para processamento paralelo é muito eficiente. Porem, elas possuem grandes desvantagens em relação a CPU como a quantidade de memória disponível, o preço elevado e a disponibilidade de memória na CPU, para evitar o aumento de complexidade do programa para carga e descarga da matriz na memória.

Tanto CUDA como o OpenMP necessitam que o programador pense de forma paralela e conheça o *hardware* com que trabalha, para obter o maior desempenho com os seus programas.

1. [↑](#footnote-ref-1)