Comparação do Desempenho Computacional na Mutiplicação de Matrizes Utilizando Ferramentas de Computação de Alto Desempenho

Maria da Penha de Andrade Abi Harb

Programa de Pós-Graduação em Informática

Universidade Federal do Espírito Santo

Vitória, Brasil

mpenha@gmail.com

*Resumo*—Este trabalho foi desenvolvido no contexto da disciplina de Arquitetura de Computadores do Programa de Pós-Graduação em Informática da UFES ministrada pelo professor Alberto Ferreira de Souza. O objetivo do trabalho foi verificar o desempenho de um programa de multiplicação de matrizes ladrilhada sequencialmente e em paralelo utilizando as tecnologias CUDA e OpenMP. A tarefa consistiu em executar o programa uma (01), cem (100) e trezentas (300) vezes, comparando o desempenho de execução das tecnologias em uma média de três (03) rodadas.

Os resultados obtidos com a rede Hybrid-CNN depois do ajuste fino colocariam o método proposto em quarto lugar da competição IJCNN 2011 com uma acurácia total de 96,92%.

Keywords— paralelismo; memória compartilhada; CUDA; OpenMP.

# Introdução

Problemas na engenharia e tecnologia vêm se tornando cada vez mais complexos e para solucioná-los novas técnicas computacionais têm sido desenvolvidas. Essas técnicas podem ser acopladas á códigos já existentes ou receberem informações de saída de algoritmos, para deixarem o sistema mais eficiente e diminuir o tempo de execução.

Assim, uma forma viável de reduzir o tempo da computação é através da paralelização da execução das instruções, ou seja, buscar executar mais de uma instrução ao mesmo tempo. Para que isso seja possível, é preciso executar o código em um ou mais computadores paralelos. Um computador paralelo pode ser um conjunto de processadores capazes de trabalhar cooperativamente para resolver um dado problema [3] e a programação paralela consiste em solucionar um problema dividindo-o em partes, de maneira que  
essas partes possam ser executadas em paralelo. O que é muito eficaz para aplicações que necessitam de  
mais desempenho, como previsão do tempo, simulações físicas, bioinformática, etc.; que levariam muitos  
dias, ou até meses, se fossem executadas sequencialmente [6].

É possível utilizar programação em paralelo em ambientes de memória compartilhada com o uso da arquitetura Multi-core, através do uso do OpenMP (Open Multi-Processing), ou com CUDA (Compute Unified Device Architecture) utilizando GPUs (Graphic Processing Units) em placas gráficas da Nvidia.

Neste trabalho procuramos realizar um comparativo, através da métrica Speed-Up, entre algumas metodologias de computação de alto desempenho na operação de multiplicação de matrizes quadradas de diversas ordens em ambiente de memória compartilhada. Pois operações matriciais são comuns nas práticas de engenharia, como, na resolução de problemas de elementos finitos. Tais operações também são conhecidas pelo elevado custo computacional por se tratarem de operações de ordem de complexidade quadrática e cúbica

O restante deste relatório é organizado da seguinte forma. Após a introdução, na seção II é apresentada a metodologia para a realização deste trabalho. E na seção III são apresentados os experimentos realizados e os resultados obtidos.

# metodologia

Este trabalho foi desenvolvido segundo as seguintes etapas:

1. Estudo da literatura sobre as tecnologias de paralelismo;
2. Desenvolvimento de código para realização dos experimentos;
3. Realização de experimentos;
4. Escrita do relatório.

A etapa 1, envolveu, o estudo da literatura das areas de OpenMP e CUDA. A tecnologia OpenMP [1], é uma biblioteca que fornece um modelo escalável e  
portável para o desenvolvimento de programas com múltiplas *threads* para memória compartilhada e é  
disponível para as linguagens de programação C, C++ e Fortran. Já CUDA é uma plataforma de software, exposta em 15 de fevereiro de 2007, pelas GPU’s (Unidade de Processamento Gráfico) da NVIDIA, que são placas gráficas tem um enorme potencial computacional. É uma arquitetura de abstração com um modelo de programação embutido, desenvolvida para suportar a computação estrita e altamente paralelizada [2]. As aplicações aceleradas com CUDA passaram a permitir o máximo de paralelismo possível, dando suporte para o uso de vários processadores ao mesmo tempo [5]. Na Tabela 1 (abaixo) é exibido a arquitetura de um sistema com memória compartilhada, classificada pelos critérios.

|  |  |  |
| --- | --- | --- |
| **Critério** | **OpenMP** | **CUDA** |
| **Execução** | Thread | Thread |
| **Metodologia de** **programação** | API, C, Fortran | API,Extensão de C |
| **Gerenciamento de** **trabalho** | Implícito | Implícito |
| **Particionamento da** **carga de trabalho** | Implícito | Explícito |
| **Mapeamento entre** **tarefa e a thread** | Implícito | Explícito |
| **Sincronização** | Implícito/Explícito | Implícito |
| **Modelo de** **comunicação** | Espaço de memória compartilhado | Espaço de memória compartilhado |

**Tabela 1 - Arquitetura de sistema com memória compartilhada [4] - Adaptada**

Para medir o desempenho do algoritmos em OpenMP e CUDA pesquisou-se sobre Speed-up, que é um termo utilizado pela Computação de Alto Desempenho para avaliar o ganho de desempenho de programas paralelos e é obtido pela divisão do tempo de execução serial pelo tempo de execução paralela. A etapa 2, envolveu o desenvolvimento de códigos em C e em CUDA para a realização dos experimentos. Primeiramente pesquisou na internet, apostilas e livros códigos prontos para estudos. Após esse processo, escolheu-se um código como ponto de partida e foram feitas as modificações necessárias para se adequar aos experimentos. Foram desenvolvidos 03 códigos para testes. O primeiro realizando as instruções de forma sequencial. O segundo de forma paralela utilizando OpenMP, e o terceiro utilizando CUDA. A etapa 3, envolveu a realização dos experimentos elaborados e análise dos resultados obtidos. Os experimentos da multiplicação foram executados em diferentes configurações:

– 1 vez

– 100 vezes

– 1000 vezes

E ajustar o tamanho das matrizes e do ladrilho para máximo desempenho paralelo e comparar com o desempenho sequencial, em uma média de 3 rodadas. Os experimentos foram executados em um computador com especificações >>>>>>, localizada no Laboratório de Alto Desempenho, da Universidade Federal do Pará. Para acesso externo foi utilizado as ferramentas PuTTy [5] (www.putty.org), que é um programa cliente para protocolos de rede SSH , e WinSCP [7] (https://winscp.net/eng/download.php), que é um cliente SFTP e FTP, que permite acessar, transferir e manipular arquivos remotamente. A etapa 4, envolveu a escrita do relatório e publicação no github.

# resultados

Os resultados foram apresentados divididos em experimentos de execução de 01, 100 e 300 vezes, exibidos em dois modelos de gráficos:

- Gráfico de Tempo de execução (em segundos) da  
multiplicação de matrizes, aumentando o  
número de threads, usando política de  
distribuição de iterações dynamic.

- Gráfico de Speedups obtidos com política de Distribuição de Iterações dynamic, conforme varia-se o número de threads.

Para todos os experimentos foram testados o algoritmo sequencialmente e com 04, 08, 128 ou 256 threads, para matrizes quadradas de dimensões de 100 a 800. Foi testado a matriz quadrada de dimensão 1000, porém só executou no algoritmo sequencial com apenas uma multiplicação de matriz. Os demais casos ocorreu erro abortado.

## Experimento 1- execução de uma vez

Os gráficos 01 e 02 mostram o resultado das execuções do código da ocorrência de uma vez da multiplicação das matrizes. Percebendo que a medida aumenta o numero de threads diminui o tempo de processamento e aumenta o speedup.

Gráfico 1 – Execução de uma multiplicação de matriz.

Gráfico 2 – Desempenho da média de Processamento - Speedup para uma execução.

## Experimento 2- execução 100 vezes

Os gráficos 03 e 04 mostram o resultado das execuções do código da ocorrência da multiplicação das matrizes 100 vezes (média). Percebendo que a medida aumenta o numero de threads diminui o tempo de processamento e aumenta o speedup.

Gráfico 3 – Execução multiplicação de matriz 100 vezes.

Gráfico 4 – Desempenho da média de Processamento - Speedup para 100 vezes de execução.

## Experimento 3- execução 300 vezes

Os gráficos 05 e 06 mostram o resultado das execuções do código da ocorrência da multiplicação das matrizes 300 vezes (média). Percebendo que a medida aumenta o numero de threads diminui o tempo de processamento e aumenta o speedup.

Gráfico 3 – Execução multiplicação de matriz 300 vezes.

Gráfico 6 – Desempenho da média de Processamento - Speedup para 300 vezes de execução.

Operações matriciais apresentam um alto custo computacional associado, porém com o uso de programação em paralelo em ambiente com memória compartilhada, utilizando OpenMP ou em GPUs utilizando CUDA, é possível obter resultados extraordinários em relação ao desempenho das operações. Esses resultados são apresentados aqui usando a métrica Speed-Up, que relaciona o tempo gasto em processamento serial com o tempo gasto em processamento paralelo.

Realizando o produto de matrizes sequencialmente e variando as dimensões das matrizes de 100 a 800, o tempo de processamento chegou até 76.1 segundos de processamento para a maior dimensão. Na versão em que se usou OpenMP foi observado um Speed-up de 5,93, sendo utilizadas 256 Threads.

O desempenho na GPU usando a biblioteca CUBLAS foi o mais eficiente sendo obtido um discrepante Speed-up de 317. Desta forma, o desempenho na GPU ficou quase 41 vezes mais rápido do que em OpenMP.

##### referências

1. Chapman, B., Jost, G., and Van Der Pas, R. Using OpenMP: portable shared memory parallel programming,  
   volume 10. MIT press, 2008.
2. Corporation, N. NVIDIA CUDA Architecture, 2009.
3. Foster, I. T. Designing and building parallel programs:concepts and tools for parallel software engineering. Reading: Addison-Wesley, 1995. 379 p.
4. Kasim, H. et al., S. Survey on Parallel Programming Model, IFIP International Conference on Network and Parallel Computing (IFIP 2008), 18-20 October 2008, Shanghai, China. Disponivel em: <http://apstc.sun.com.sg/content.php?l1=research&l2=resources&l3=pubs>. Acesso em: out. 2016.
5. PuTTy, PuTTY - a free SSH and telnet client for Windows. Disponivel em: < http://www.putty.org/>. Acesso em: out. 2016.
6. Schepke, C.; Lima, J. V. F. Programação Paralela em Memória Compartilhada e Distribuída. In: DE ROSE, C.;  
   SCHNORR, L. M.; PASIN, M., ed. Anais da ERAD 2015. SBC, 2015. p 45-70.
7. WinSCp, Free SFTP, SCP and FTP client for Windows. Disponivel em: < https://winscp.net/eng/download.php>. Acesso em: out. 2016.