ISSN: 2319-0124
josif 2023
15ª JORNADA CIENTÍFICA E TECNOLÓGICA
12º SIMPÓSIO DE PÓS-GRADUAÇÃO
DO IFSULDEMINAS
REALIZAÇÃO INSTITUTO FEDERAL Sul de Minas Gerais
APOIO FAPEMIG CNPq

USO DE PARALELIZAÇÃO NA MULTIPLICAÇÃO DE MATRIZES USANDO A BIBLIOTECA OPENMPI

Laura C. RATIS[1]; Yago G. OLIVEIRA[2]; João P. de T. GOMES[3]

RESUMO

Este trabalho apresenta uma investigação sobre o uso do MPI (*Message Passing Interface*) para realizar processamento paralelo em um código de multiplicação de matrizes e mensurar o desempenho computacional de um *cluster* embarcado de baixo custo implementado com *TVs Box*. Nas simulações, foi possível constatar um aumento considerável de desempenho obtido com o uso do *cluster* quando comparado com soluções sequenciais, no entanto, recomenda-se simular matrizes maiores e ampliar a quantidade de nós do *cluster* para certificar sua eficiência diante da complexidade da multiplicação de matrizes.

Palavras-chave: *OpenMPI*; *Speedup*; Eficiência; *TV Box; Cluster*.

1. INTRODUÇÃO

A multiplicação de matrizes é uma operação fundamental na álgebra linear e possui aplicações práticas em diversas áreas como computação, economia, engenharia e física. O problema da multiplicação de matrizes é um clássico problema estudado na ciência da computação, mais especificamente, na área de programação paralela. Sua relevância decorre da necessidade de otimizar seu desempenho para acelerar a execução de algoritmos que dependem dessa operação. Ademais, apesar dos processadores modernos possuírem uma alta performance, muitos problemas ainda demandam um elevado tempo de execução para computadores comuns. Uma maneira de superar tais problemas e, consequentemente, obter uma melhoria esperada em desempenho de algoritmos complexos, consiste na utilização de arquiteturas computacionais paralelas. Entretanto, computadores projetados para processamento paralelo possuem um alto custo, o que impõe uma restrição a sua utilização. Uma alternativa mais acessível é a utilização de um *cluster* de computadores pessoais e, assim, usufruir do processamento através de uma rede (WAINTRAUB, 2009).

Atualmente, existem diversas bibliotecas que auxiliam no desenvolvimento de programas paralelos e permitem que o gerenciamento de recursos compartilhados entre os *clusters* seja seguro. Uma destas bibliotecas é a *Open Message Passing Interface* (OpenMPI), que foi desenvolvido ainda na década de 1990 através de pesquisas e publicações realizadas por integrantes de um fórum

[1] Discente do curso de Bacharelado em Ciência da Computação, IFSULDEMINAS – *Campus* Passos. E-mail: laura.ratis@alunos.ifsuldeminas.edu.br.
[2] Discente do curso de Bacharelado em Ciência da Computação, IFSULDEMINAS – *Campus* Passos. E-mail: yago.oliveira@alunos.ifsuldeminas.edu.br.
[3] Professor, IFSULDEMINAS – *Campus* Passos. E-mail: joao.gomes@ifsuldeminas.edu.br.

sobre computação paralela e distribuída (PEREIRA, 2014).

O algoritmo tradicional (sequencial) de multiplicação de matrizes é amplamente utilizado para fins de estudos comparativos, ou seja, possui três laços de repetição aninhados e apresenta uma complexidade sequencial cúbica, conforme Quadro 1, onde A e B são as matrizes a serem multiplicadas e R a matriz resultante. Cada elemento R *i*, *j* é obtido através da soma dos produtos dos elementos correspondentes de posição *k* na i-ésima linha de A pelos também de posição *k* na j-ésima coluna de B (KRAUSE *et al*, 2016).

Quadro 1- Algoritmo sequencial para multiplicação de matrizes.

```
for i a N do
        for j a N do
                for k a N do
                        R[i][j] =+ A[i][k] * B[k][j] end end end
                end
        end
end
```

O MPI não é homologado por institutos de referência como, por exemplo, o *Institute of Electrical and Electronics Engineers* (IEEE), mas consiste em um padrão definido para arquiteturas de memória distribuída, promovendo rotinas e consolidação dos dados, sincronização e controle de processos entre os computadores que compõem uma infraestrutura distribuída. Uma implementação utilizando MPI oferece ao programador uma camada de abstração de alto nível que, através de uma conexão em rede, pode ser aplicada e gerenciada por um sistema operacional com o objetivo principal de fornecer uma interface simples para o desenvolvimento de aplicações com alto desempenho, principalmente, utilizando as linguagens Fortran, C e C++ (PEREIRA, 2014).

Os *clusters* são programados e controlados por *softwares*, sendo os primeiros desenvolvidos na década de 60. A implementação de um *cluster* geralmente é motivada pela necessidade de performance e geralmente são organizados de duas maneiras distintas: os centralizados, onde os computadores usados estão reunidos em uma mesma sala, geralmente em um *rack*, e os descentralizados, onde os computadores estão em salas diferentes ou até mesmo em prédios diferentes (MATOS, 2018).

Contudo, o objetivo deste trabalho é realizar uma análise de desempenho de um algoritmo de multiplicação de matrizes utilizando a biblioteca OpenMPI em um *cluster* composto por *TVs Box.*

2. MATERIAL E MÉTODOS

Para a execução do código paralelo foi utilizado um *cluster* com 8 nós formado por *TVs Box* modelo TX6 (com o sistema operacional Armbian), conectados a um *switch* de 8 portas. A formatação das *box*, configuração e montagem do *cluster*, pode ser conferida através do seguinte

endereço eletrônico: <https://www.espacomaker-passos.com/laboratorios-e-materias-de-apoio>.

Os testes de desempenho foram realizados através de matrizes aleatórias de tamanho: 1000 x 1000, 2000 x 2000 e 3000 x 3000, que passaram por rodadas contendo de 1 a 32 núcleos de processamento para mensurar o *speedup* e a eficiência de execução do algoritmo. Dessa forma, para comparar essas métricas, foi realizado um teste de execução sequencial para encontrar o tempo gasto considerando o mesmo tamanho das matrizes.

3. RESULTADOS E DISCUSSÃO

Como orientação inicial, foram realizadas 3 execuções do código de forma sequencial e obteve-se os resultados conforme Quadro 2.

Quadro 2- Tempo médio de execução sequencial

Tamanho da matriz	**1000x1000**	**2000x2000**	**3000x3000**
Tempo (s)	52,72	426,61	1260,42

Os Quadros 3, 4 e 5 apresentam os resultados obtidos através da execução de cada matriz quadrada e pode ser comparado com os resultados obtidos durante a execução sequencial.

Quadro 3 - Tempo médio de execução paralela (matriz 1000x1000)

Tamanho	**1000x1000**				
Threads	**4**	**8**	**12**	**16**	**32**
Tempo (s)	14,17	7,65	5,17	4,12	3,08
Speedup	3,72	6,89	10,20	12,80	17,12
Eficiência	0,93	0,86	0,85	0,80	0,53

O resultado apresentado na execução da matriz de tamanho 1000 x 1000 demonstra que com o aumento do número de processos o tempo de execução reduz, no entanto, a eficiência com 32 núcleos cai em torno de 50% e, apesar do tempo reduzido, isso indica que rodar esta matriz aleatória com 32 núcleos não é recomendável do ponto de vista da eficiência.

Quadro 4 - Tempo médio de execução paralela (matriz 2000x2000)

Tamanho	**2000x2000**				
Threads	**4**	**8**	**12**	**16**	**32**
Tempo (s)	315,52	168,47	116,11	85,34	44,72
Speedup	1,35	2,53	3,67	5,00	9,54
Eficiência	0,34	0,32	0,31	0,31	0,30

O resultado apresentado na execução da matriz de tamanho 2000 x 2000 demonstra que com o aumento do número de processos o tempo de execução também reduz, no entanto, percebe-se que a complexidade demanda um tempo de execução maior e o tempo consumido em 4 processos está

próximo do tempo de execução sequencial. Isso aponta que é interessante rodar este tamanho de matriz a partir de 4 processos. Contudo a eficiência cai consideravelmente e se mantém estável em torno de 30%.

Quadro 5 - Tempo médio de execução paralela (matriz 3000x3000)

Tamanho	**3000x3000**				
Threads	**4**	**8**	**12**	**16**	**32**
Tempo (s)	1115,87	751,82	553,14	481,24	316,96
Speedup	1,13	1,68	2,28	2,62	3,98
Eficiência	0,28	0,21	0,19	0,16	0,12

O Quadro 5 apresenta resultados semelhantes (guardadas as proporções de complexidade e tamanho de matriz) em relação ao apresentado pela matriz de 2000 x 2000.

4. CONCLUSÃO

Através dos resultados obtidos, conclui-se que o tempo de execução do algoritmo de multiplicação de matrizes tende a diminuir quando novos nós participam da execução e, consequentemente, um aumento no *Speedup*, no entanto, observa-se que a relação *Speedup* x Eficiência aponta caminhos opostos. Contudo, é possível verificar que em todos os cenários a execução paralela sobressai à sequencial, mas pode-se dizer que esta aplicação só é viável quando utilizado um cluster de baixo custo com reaproveitamento de *hardware*, pois a eficiência neste caso não está adequada. Como proposta futura seria analisar possíveis mudanças no algoritmo e ampliar o número de nós do *cluster* para verificar seu comportamento.

REFERÊNCIAS

KRAUSE, Arthur Mittmann; MORO, Gabriel Bronzatti; SCHNORR, Lucas Mello. **Análise de Desempenho da Multiplicação de Matrizes por Strassen contra o Método Tradicional.** Workshop on Parallel and Distributed Processing. 2016, Porto Alegre, Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS).

MATOS, Elias Rabelo. **Implementação e análise de desempenho do Framework Apache Hadoop e da Biblioteca OpenMPI para Multiplicação de Matrizes com um Cluster de Baixo Custo na Plataforma Raspberry.** 2018. 93f. Trabalho de Conclusão de Curso (Graduação)- Departamento de Computação da Universidade Federal de Sergipe, São Cristóvão, 2018.

PEREIRA, Filipe Jerônimo. **Avaliação da plataforma OpenMPI para paralelização do processo de compilação de software**. 2014. 72f. Trabalho de Conclusão de Curso (Graduação)- Universidade Federal de Santa Catarina, Araranguá, 2014.

WAINTRAUB, Marcel. **Algoritmos paralelos de otimização por enxame de partículas em problemas nucleares**. 2009. 97f. Tese (Doutorado)- Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2009.