Sistemas Operacionais
Multiplicação de Matrizes

Geffté L. S. Caetano[1] **, Amanda B. M. P. Ribeiro**[2]

[1]Faculdade de Computação – Universidade Federal de Mato Grosso do Sul (UFMS)
Caixa Postal 549 – 79.070-900 – Campo Grande – MS – Brazil

{geffte.caetano, b.amanda}@ufms.br

Abstract. *This report presents the concept of threads, their variations, and the application of multithreading in the calculation of the product of matrices, with implementations in C++ and Java, in addition to performance analysis.*

Resumo. *Este relatório apresenta o conceito de threads, suas variações e a aplicação de multithreading no cálculo do produto de matrizes, com implementações em C++ e Java, além de análise de desempenho.*

1. Thread

Thread é a menor unidade de execução dentro de um processo. Cada thread compartilha o mesmo espaço de endereçamento e recursos do processo, mas possui seu próprio contador de programa, pilha e registradores. As threads permitem que múltiplas tarefas sejam realizadas concorrentemente dentro do mesmo processo, otimizando o uso da CPU e reduzindo overhead de criação comparado a processos independentes [Tanenbaum and Bos 2014].

1.1. SingleThread

Single-thread refere-se a programas ou processos que executam uma única thread de controle. A execução é sequencial, uma instrução após a outra, sem qualquer paralelismo interno. Essa abordagem é simples de implementar, depurar e testar, mas limita o desempenho em sistemas multicore, pois não aproveita a capacidade de execução paralela [Silberschatz et al. 2018].

Exemplo: programas de linha de comando que executam tarefas simples como cópia de arquivos ou cálculo de expressões aritméticas.

1.2. MultiThread

Multithread é a técnica que permite que múltiplas threads sejam executadas concorrentemente dentro do mesmo processo. Utiliza-se para melhorar a responsividade e o desempenho, especialmente em sistemas multicore, onde múltiplas threads podem ser executadas simultaneamente em diferentes núcleos [Butenhof 1997]. Multithreading é amplamente utilizado em aplicações como servidores web, sistemas operacionais, e softwares interativos.

Exemplo: servidores HTTP como Apache ou Nginx, que criam múltiplas threads para atender conexões simultâneas.

2. Problema das Matrizes

O produto de matrizes é uma operação fundamental em computação científica. Dadas duas matrizes, A (M×K) e B (K×N), o produto resulta na matriz C (M×N), onde cada elemento C_{ij} é calculado como:

$$C_{ij} = \sum_{n=1}^{K} A_{i,n} \times B_{n,j}$$

Esta operação tem complexidade $O(MKN)$, e pode ser otimizada com abordagens paralelas [Cormen et al. 2009].

2.1. Abordagem convencional

A abordagem convencional implementa o produto de matrizes usando três laços aninhados, percorrendo as linhas de A, colunas de B e acumulando os produtos correspondentes. Essa estratégia é simples, mas não explora paralelismo, levando a baixo desempenho para matrizes grandes.

2.2. Abordagem Multithreads

Na abordagem multithread, a computação de diferentes elementos ou blocos da matriz C é distribuída entre múltiplas threads. Cada thread calcula uma parte da matriz, reduzindo o tempo total de execução e aumentando o aproveitamento dos recursos computacionais. Modelos populares incluem paralelismo por linhas, colunas ou blocos [Quinn 2004].

3. Implementação

3.1. C++

Em C++, a biblioteca padrão oferece suporte a threads via ¡thread¿, permitindo criar e gerenciar múltiplas threads para computar o produto de matrizes de forma concorrente. Nesta implementação, a sincronização com std::mutex não é necessária, pois cada thread opera em elementos distintos da matriz de resultado, evitando condições de corrida.

4. Resultados e discussões

Table 1. Resultados de desempenho da multiplicação de matrizes em serial e em paralelo.

Matriz	Tempo (s) Serial	Tempo (s) Paralelo	SpeedUp
[2×3] × [3×4]	0.000629856	0.000563412	1,11793146
[4×5] × [5×6]	0.00141285	0.00120173	1,175680061
[9×10] × [10×11]	0.00408349	0.00361097	1,130856806
[49×50] × [50×51]	0.0434599	0.0421714	1,0305
[99×100] × [100×101]	0.37051	0.352976	1,0496
[499×500] × [500×501]	6.392	0.868378	7.36085
[999×1000] × [1000×1001]	50.7602	7.36985	7.3042
[4999×5000] × [5000×5001]	1320.1523	188.5714	7

Dada a tabela acima, observa-se que, com o aumento do tamanho das matrizes, o tempo de execução do algoritmo serial cresce significativamente, enquanto o algoritmo paralelo mantém um desempenho muito mais eficiente. Esse comportamento evidencia a escalabilidade da abordagem paralela frente à limitação do processamento sequencial. Por outro lado, ao analisarmos matrizes de menor dimensão, a diferença entre os tempos de execução torna-se pouco expressiva, a ponto de o ganho de desempenho ser praticamente imperceptível — fato confirmado pelos baixos valores observados na métrica de speedup. Isso indica que o paralelismo só se torna vantajoso a partir de um certo limiar de complexidade computacional.

5. Conclusões

O uso de multithreading no produto de matrizes reduz significativamente o tempo de execução, especialmente em sistemas multicore. A escolha da linguagem e modelo de paralelismo influencia o desempenho, como demonstrado pelos resultados experimentais comparando a implementação Serial e Paralela.

References

Butenhof, D. R. (1997). *Programming with POSIX Threads*. Addison-Wesley.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to Algorithms*. MIT Press, 3 edition.

Quinn, M. J. (2004). *Parallel Programming in C with MPI and OpenMP*. McGraw-Hill.

Silberschatz, A., Galvin, P. B., and Gagne, G. (2018). *Operating System Concepts*. John Wiley & Sons.

Tanenbaum, A. S. and Bos, H. (2014). *Modern Operating Systems*. Pearson.