Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital
IMD1116 - COMPUTAÇÃO DE ALTO DESEMPENHO - T01 (2025.1)
Tarefa 1: Memória cache

Docente: SAMUEL XAVIER DE SOUZA
Discente: Iago Gabriel Nobre de Macedo (20220037927)

Link do repositório da atividade no Github:
https://github.com/IagoGMacedo/IMD1116-Computacao-de-Alto-Desempenho/blob/main/exercicios/multiplacao_matrizes/MultiplicacaoMatrizes.c

Implementação dos métodos de multiplicação

Para a realização desta atividade, desenvolvi inicialmente o programa "MultiplicacaoMatrizes", que contém métodos para alocar, preencher com valores aleatórios e multiplicar matrizes utilizando as diferentes abordagens descritas. Logo no início do programa, os tamanhos das matrizes são definidos de forma parametrizável, permitindo ajustes práticos e rápidos.

Cada abordagem de multiplicação possui um método específico, responsável pelo processamento e pelo registro do tempo de execução no console.

Resultados obtidos

Para garantir uma análise consistente, comparei o tempo de processamento de cada algoritmo multiplicando matrizes de diferentes dimensões: 64×64, 128×128, 256×256, 512×512 e 1024x1024. Os tempos de execução obtidos (**em segundos**) foram os seguintes:

	64×64	**128x128**	**256x256**	**512x512**	**1024x1024**
acesso por linhas	0.001735	0.017264	0.075261	0.639508	7.393510
acesso por colunas	0.001670	0.013220	0.069054	0.626917	6.885076

Considerações

Nesse sentido, observa-se que os tempos de execução divergem significativamente em matrizes de tamanho 1024×1024. Nesse ponto, o método de acesso por colunas passa a ser notavelmente mais rápido que o acesso por linhas, com uma diferença de aproximadamente 0.5 segundos.

A hierarquia de memória (L1, L2, L3) tem um papel crítico para esse acontecimento. Nos casos em que a matriz é pequena (valores menores que 1024), seus dados cabem totalmente na cache, minimizando o custo de acesso à memória RAM. Porém, para matrizes maiores, os dados ultrapassam a capacidade da cache, levando a cache misses frequente.

Nesse sentido, a abordagem de acesso por colunas se torna mais vantajosa por permitir que o processador otimize melhor o pré-carregamento dos blocos de memória e reutilize os dados que já estão na cache, mesmo quando os dados totais da matriz ultrapassam a capacidade dos níveis L1, L2 e L3. Ou seja, apesar de as matrizes grandes provocarem cache misses frequentes, o acesso por colunas pode reduzir a sobrecarga de recarregamento dos dados, aproveitando melhor os blocos de dados já presentes na hierarquia de memória, o que reflete na melhoria do desempenho em cenários com matrizes maiores.