SuperComp - Atividade 0

Giovana Cassoni Andrade.

Começando com uma matriz de 200x200 nos 3 exemplos de código, é feita uma alteração para aumentar o tamanho das matrizes para 300x300, 900x900 e 1300x1300.

	200 x 200	**300 x 300**	**900 x 900**	**1300 x 1300**
Python	0.88 s	2.98 s	83.75 s	257.80 s
C++	0.0607062 s	0.210061 s	6.39031 s	19.327 s
Paralelismo em C++	0.0602256 s	0.207801 s	6.14615 s	18.937 s

Tabela 1 - Tempo de execução para multiplicação de matrizes.

Em relação à Tabela 1, foi elaborado um gráfico que relaciona a complexidade do problema (tamanho da matriz) com o tempo de execução para cada implementação.

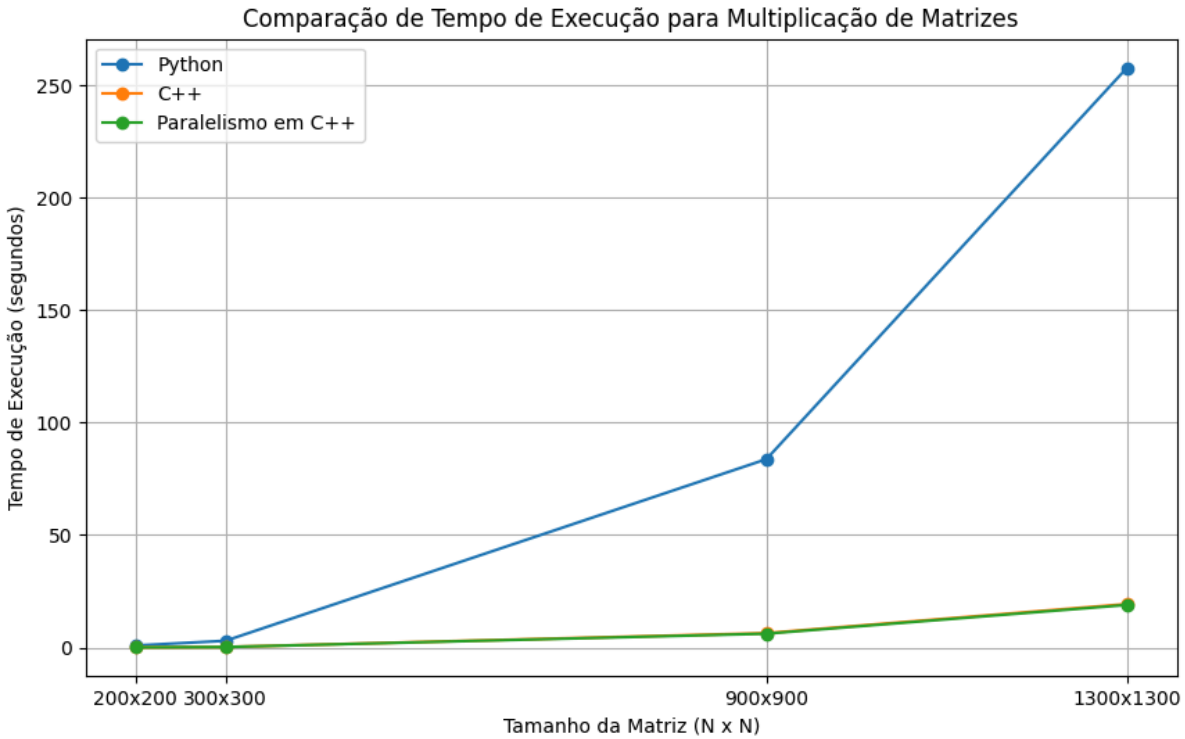

Gráfico 1 - Comparação dos 4 tamanhos de matrizes analisados para todos os programas.

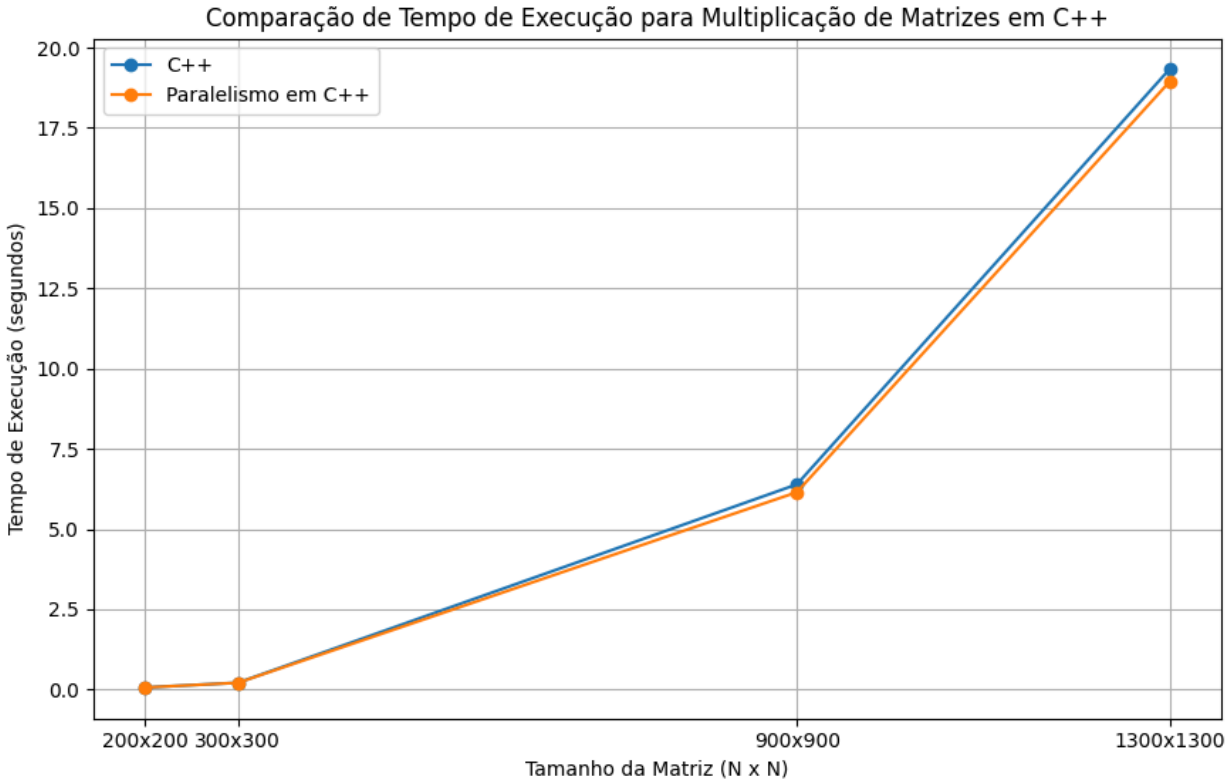

Gráfico 2 - Comparação dos 4 tamanhos de matrizes analisados para os programas em C++.

Fazendo uma análise comparativa sobre o impacto do paralelismo no desempenho de acordo com a complexidade do problema, pode-se dizer que, em um momento inicial, o paralelismo não apresenta um grande impacto (Gráfico 1). Porém, conforme a complexidade do problema aumenta, a diferença entre o tempo de execução vai se tornando maior, com um maior impacto do paralelismo no desempenho (Gráfico 2). E vale notar que o código em python obteve a pior performance em todos os casos (Gráfico 1).

Este problema com as matrizes poderia ser abordado em um ambiente de High-Performance Computing usando programação paralela ou alocação de recursos, também realizando profiling ou otimização do código, tendo uma performance melhor do que abordagens sequenciais. Um bom exemplo complexo seria o treinamento de redes neurais em deep learning.