Otimizações em multiplicações de matrizes

Marcelo Schreiber e Felipe Vieira

***Resumo*—O relatório aborda otimização de multiplicação de matrizes usando Unroll & Jam e Loop Unroll. Experimentos em um computador específico revelaram melhorias em energia, FLOPs, tempo, largura de banda de memória e L2 cache miss.**

***Index Terms*—Otimização, Unroll, Jam, Loop, Desempenho.**

I. Introdução

Neste relatório, apresentaremos a otimização da multiplicação de matrizes utilizando duas técnicas: Unroll & Jam e Loop Unroll. Também é relevante que o algoritmos de multiplicação de matrizes x matrizes e matrizes x vetores são bem semelhantes e, para diferenciar os métodos de otimização, optou-se por não usar loop blocking no algoritmo vetor x matriz, para poder observar a diferença entre: não otimizar, otimizar com unroll & jam e otimizar com unroll & jam e loop blocking.

II. Metodologia

A. Especificações

- CPU: AMD Ryzen 5 3500U @ 2.100GHz
- Sistema Operacional: Linux Pop!_OS 22.04 LTS x86_64
- Memória RAM: 5861MiB
- Compilador: GCC 11.4.0
- Tamanho das matrizes: Variando de 64x64 a 4000x4000

- Mais detalhes no arquivo likwid-topology.txt

B. Limitações

É importante ressaltar que as escolhas de fator de unroll 8 e fator de blocking como 16, quando ambas as estratégias são utilizadas, somente matrizes com tamanho multiplo de 16 funcionam corretamente.

Também, na arquitetura descrita, não há o MFLOPS AVX e nem o ENERGY isoladamente (somente ENERGY CORE e ENERGY PKG) como opções no LIKWID.

III. Figuras e Tabelas

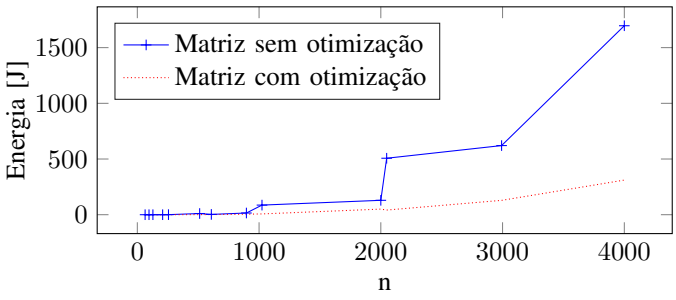

Figura 1. Gráfico com a relação Energia em Jaules por Tamanho da matriz.

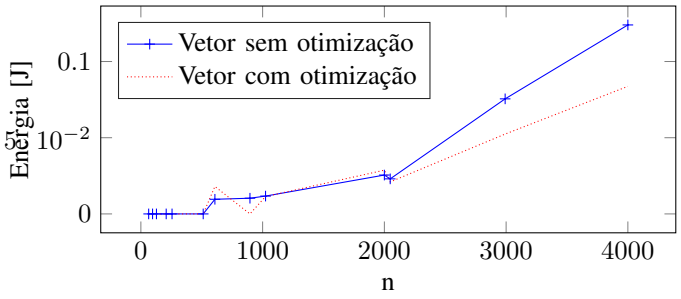

Figura 2. Gráfico com a relação Energia em Jaules por Tamanho do vetor.

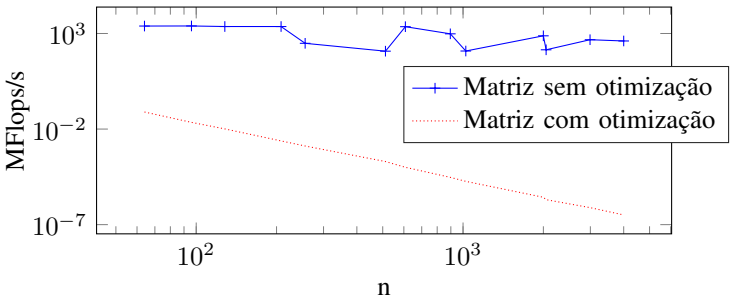

Figura 3. Gráfico com a relação MFlops/s por Tamanho da matriz.

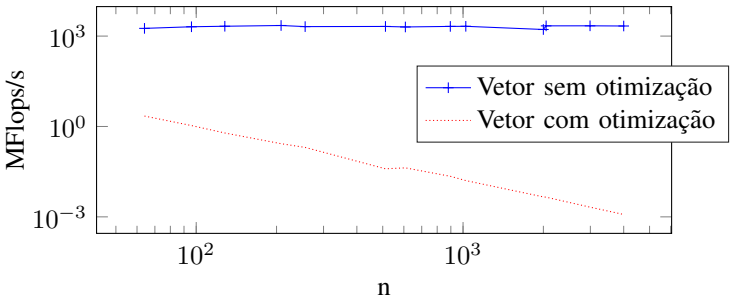

Figura 4. Gráfico com a relação MFlops/s por Tamanho do vetor.

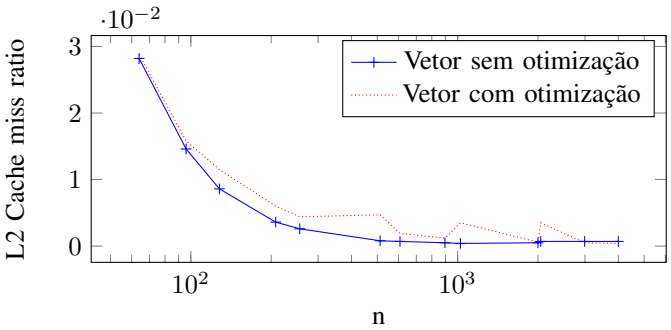

Figura 5. Gráfico com a relação l2 cache miss ratio por Tamanho do vetor.

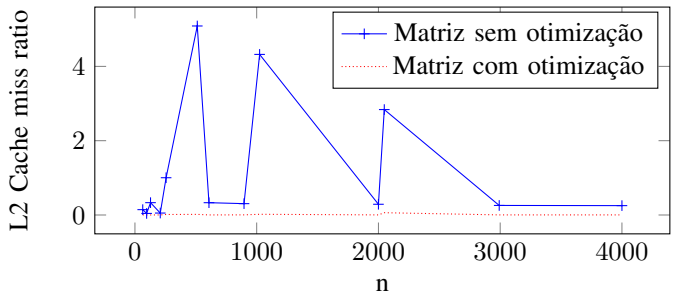

Figura 6. Gráfico com a relação l2 cache miss ratio por Tamanho da matriz.

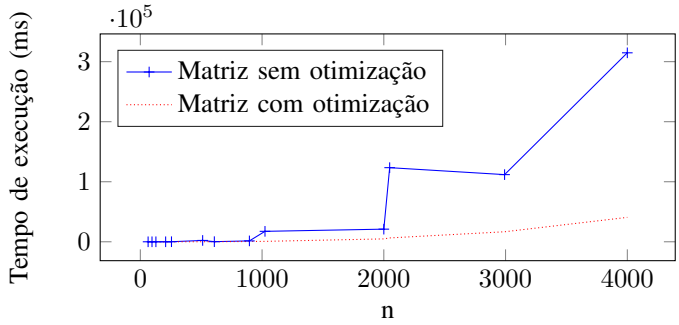

Figura 7. Gráfico com a relação do tempo de execução por Tamanho da matriz.

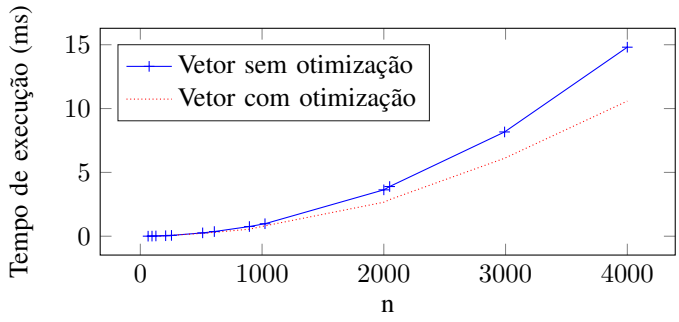

Figura 8. Gráfico com a relação do tempo de execução (ms) por Tamanho do vetor.

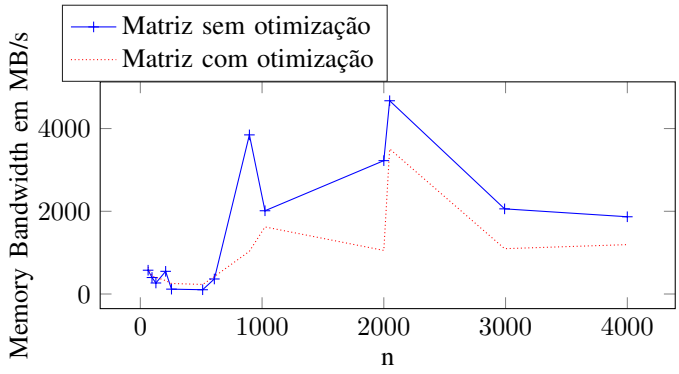

Figura 9. Gráfico com a relação da banda de memória (MBytes/s) por Tamanho da matriz.

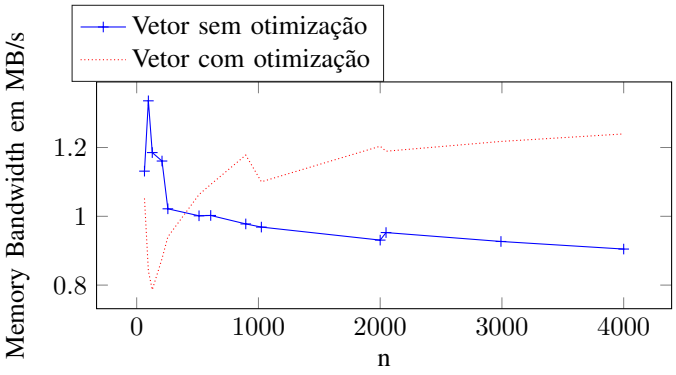

Figura 10. Gráfico com a relação da banda de memória (MBytes/s) por Tamanho do vetor.

IV. Conclusão

A otimização da multiplicação de matrizes utilizando as técnicas de Unroll & Jam e Loop Blocking demonstrou melhorias significativas no desempenho. Reduzimos o consumo de energia, diminuimos os MFLOP/s, diminuímos o tempo de execução e melhoramos o uso da largura de banda de memória. Além disso, a taxa de erros de L2 cache foi reduzida.

No entanto, uma uma anomalia importante de ressaltar é na Figura 12 e na Figura 8, onde como o loop blocking não foi realizado, a taxa de miss e largura de banda acabaram ficando piores que sem otimização. Isso ressalta a importância do loop blocking como uma estratégia de otimização.