Relatório Trabalho II - Otimização de Desempenho

Lucas Néia Torres
GRR20210570

Leonardo Becker de Oliveira
GRR20211779

I. Introdução

O objetivo do presente trabalho é otimizar e analisar o desempenho do trabalho 1 da matéria de Introdução à Computação Científica desenvolvido pelos autores. Trabalho este que teve como objetivo implementar um programa para calcular um polinômio de grau N que se ajuste a uma curva descrita por K pontos, utilizando aritmética intervalar para representação rigorosa dos valores reais. Para isso, utilizaremos alguns métodos para o acesso otimizado da memória, além do método de otimização de desempenho `Loop Unroll & Jam`.
Para realizar as análises de desempenho do código desenvolvido e otimizado, serão utilizadas as métricas "FLOPS DP", "FLOPS AVX DP", "Memory bandwidth", "cache miss RATIO"da ferramenta de monitoramento de desempenho chamada `LIKWID`.
No presente relatório, serão abordadas as principais alterações das versões do trabalho e a análise dessas alterações considerando as métricas citadas.

II. Principais alterações

Para otimizar o trabalho, é necessário realizar algumas alterações no código fonte do projeto, principalmente na maneira de acesso aos dados na memória.

A. Utilizar matrizes row-oriented

Foram substituídas todas as matrizes column-oriented (orientadas por coluna), que é a comumente utilizada, por row-oriented (orientadas por linha). Isso porque os endereços de memória da linha da matriz estão armazenados fisicamente próximos, então ao utilizar uma matriz row-oriented é acessada a memória de maneira contígua, aproveitando a localidade espacial e diminuindo o cache misses.

B. Iterar o maior laço externamente

Foram trocadas as ordens de iteração quando é iterado os pontos e o grau do polinômio, movimentando o `for` dos pontos para o loop externo. Isso porque a quantidade de pontos é maior que o grau do polinômio, então iterar sobre os pontos mais externamente pode aproveitar melhor a localidade espacial dos dados. Ao percorrer os pontos no loop mais externo, os acessos à memória tornam-se mais contínuos, potencialmente reduzindo a ocorrência de cache misses e melhorando o desempenho geral do código.

C. Utilizar `Loop Unroll & Jam`

Foi adicionado `Loop Unroll & Jam` nas iterações dos pontos e grau do polinômio, nas funções de calcular o vetor de resultados e de calcular o resíduo. Já que o maior valor entre a quantidade de pontos e grau do polinômio é a quantidade de pontos, o unroll é feito nessa variável, além disso, o fator de unroll utilizado foi 4, foi testado com 8 porém não houve otimização considerável. Essa técnica visa melhorar o desempenho dos loops, se aproveitando do pipelining, pois, quando são agrupadas múltiplas iterações de um loop, é feita a execução paralela de instruções, além de utilizar melhor a localidade temporal, já que os dados acessados em iterações subsequentes ainda estão presentes nos registradores ou na cache.

D. Fazer otimizações básicas

Na função de calcular a matriz de coeficientes, foram feitas divisões dos loops internos, não indo ate a quantidade de pontos no loop interno, reduzindo o cache misses e evita recálculos no preenchimento do restante da matriz. Além disso, não é feito o cálculo do segundoValor a cada iteração, já que é um valor constante naquele ponto, o que também chamava a função de potenciação, diminuindo o número operações de ponto flutuante. Por fim, Foram removidas as matrizes/vetores auxiliares da função de eliminação de gauss e uma multiplicação não utilizada.

E. Tirar `fesetround`

Ao retirar as chamadas da função `fesetround` antes das operações foi perceptível uma significativa diminuição no tempo de execução do programa, e, já que os intervalos dos resultados sem essa função não foram valores discrepantes, foi decidido retirar tais chamadas da função antes das operações.

III. Análise do desempenho

Todas os teste foram realizados utilizando um dispositivo disposto da arquitetura Intel Coffeelake, com um processador Intel Core i5-7500.
É possível encontrar todos os gráficos referentes a cada subseção ao final do documento.

A. Métrica de Tempo

Conforme esperado, houveram reduções significativas no tempo para gerar, resolver e calcular o resíduo do respectivo sistema linear construído.

B. Métrica operações aritméticas - FLOPS DP

Dado as otimizações realizadas, cujo reduziram a quantidade de tempo significativamente, o esperado era que a quantidade de operações aritméticas por segundo tivesse um acréscimo, do qual foi o resultado real apresentado. Então, por mais que uma das otimizações foi reduzir a quantidade de operações movendo o cálculo de certos paramêtros para fora do loop (mantendo a solução exata), a compensação por parte do tempo ganho fez com que tivesse um aumento na quantida de operações possíveis por segundo.

C. Métrica de operações aritméticas - FLOPS AVX

No contexto desse trabalho, nenhuma operação SIMD E AVX foi utilizada, dado que as operações aritméticas básicas, foram mediadas por intervalos, prejudicando o ambiente de aplicaçao dessas instruções. Então, conforme esperado, a quantidade de operações FLOPS AVX ficou zerada nos 3 casos de interesse.

D. L2 Miss Ratio

Conforme descrito no item B da seção 2, foi movido o laço que itera a quantidade de pontos para o loop externo, com o objetivo de aproveitar melhor a região da cache e sua localidade espacial, ocasionando portanto menor miss ratio, visto que o tamanho do sistema linear armazenado em cache necessita menos trocas em relação aos endereços dos pontos armazenados em cache.

E. L3 Bandwidth [MBytes/s]

Visto que o código foi reorganizado a fim de melhorar a localidade espacial dentro das funções a qual foram cabíveis, assim como minimizou-se chamadas de certos procedimentos, o resultado foi uma maior largura de banda, ou seja, houve um aumento do fluxo de troca de informações na execução do programa, sendo mais um ponto positivo as otimizações aplicadas.

IV. Conclusão

O trabalho realizado proporcionou melhorias significativas no tempo de execução do código, refletindo diretamente na eficiência das operações realizadas. As principais alterações, como a mudança para matrizes row-oriented, a reorganização dos loops externos e a aplicação do `Loop Unroll & Jam`, contribuíram para a redução do tempo de processamento em todas as etapas do programa, desde a geração do sistema linear até o cálculo do resíduo.

A análise das métricas de desempenho, como FLOPS DP, L2 Miss Ratio e L3 Bandwidth, demonstrou que as otimizações aplicadas resultaram não apenas em uma execução mais rápida, mas também em uma utilização mais eficiente dos recursos do sistema. A quantidade de operações por segundo aumentou, indicando uma melhor utilização das capacidades de processamento da CPU. Além disso, a redução do L2 Miss Ratio e o aumento da largura de banda da L3 cache evidenciam a melhoria na localidade espacial dos dados, resultando em um acesso mais eficiente à memória.

Em resumo, as otimizações realizadas não apenas atenderam ao objetivo de reduzir o tempo de execução do código, mas também proporcionaram benefícios em termos de utilização de recursos e desempenho.

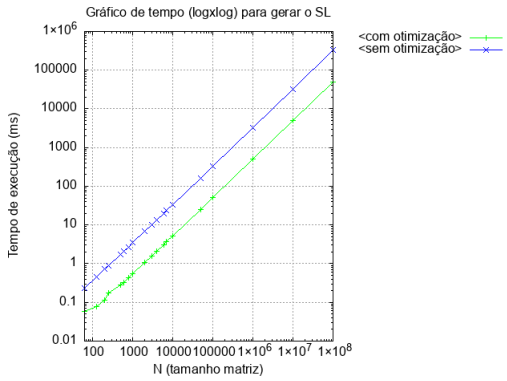

Figura 1. Tempo para gerar SL

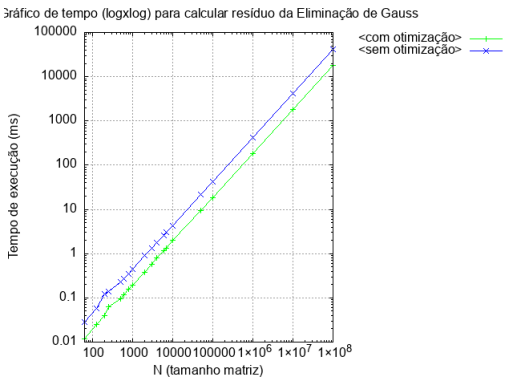

Figura 2. Tempo para cálculo do resíduo

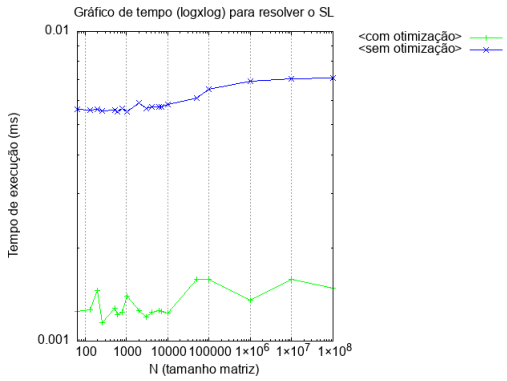

Figura 3. Tempo para resolver SL

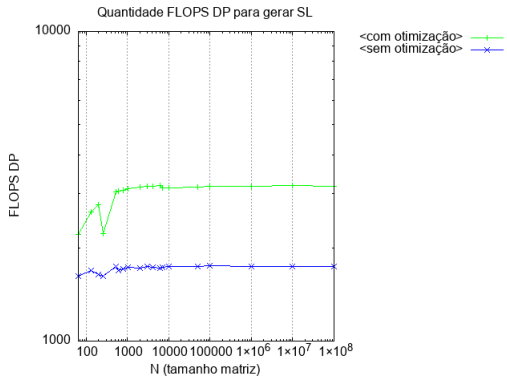

Figura 4. FLOPS DP/s para gerar SL

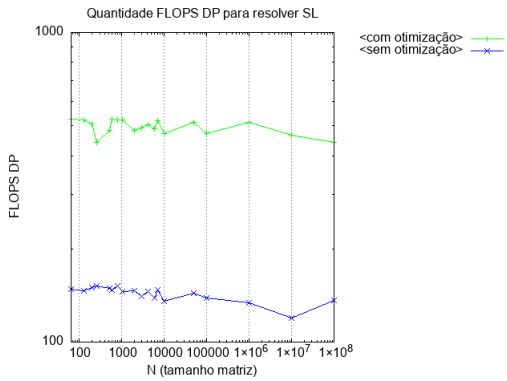

Figura 5. FLOPS DP/s para resolver SL

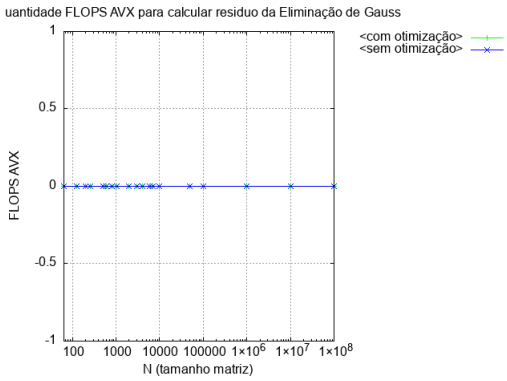

Figura 6. FLOPS DP/s para cálculo do resíduo

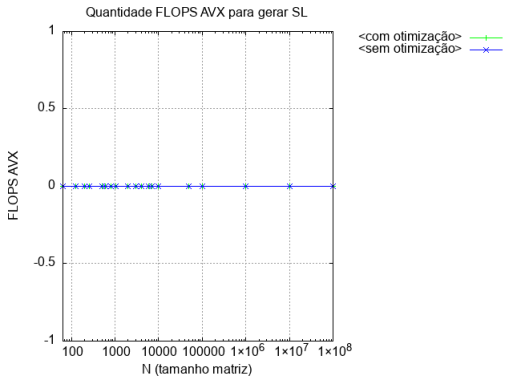

Figura 7. FLOPS AVX para gerar SL

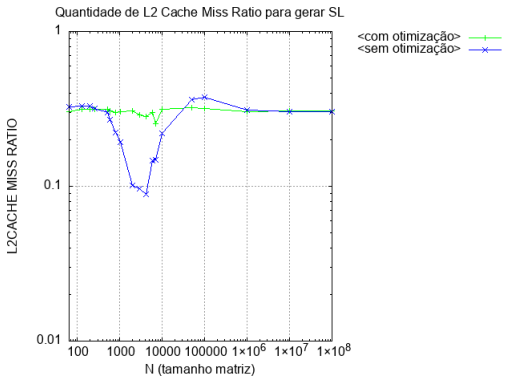

Figura 10. L2 miss ratio para gerar SL

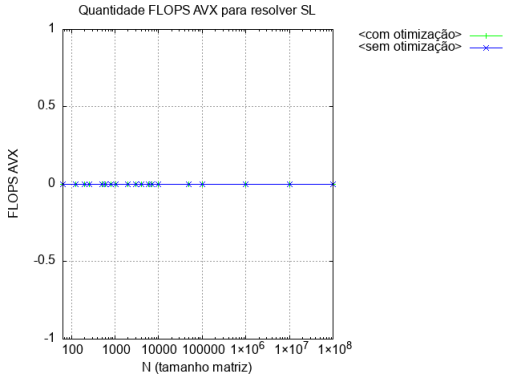

Figura 8. FLOPS AVX para resolver SL

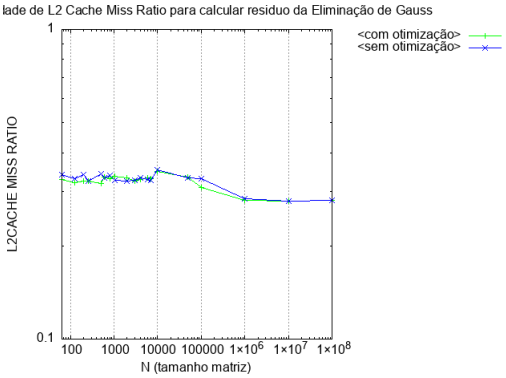

Figura 11. L2 miss ratio para cálculo do resíduo

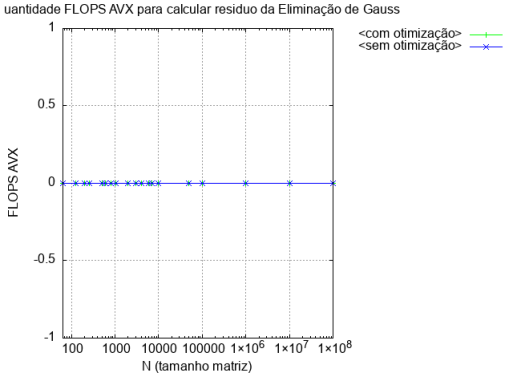

Figura 9. FLOPS AVX para cálculo do resíduo

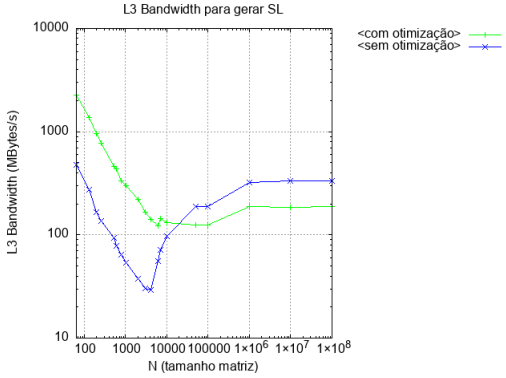

Figura 12. L3 Bandwidth ratio para gerar SL

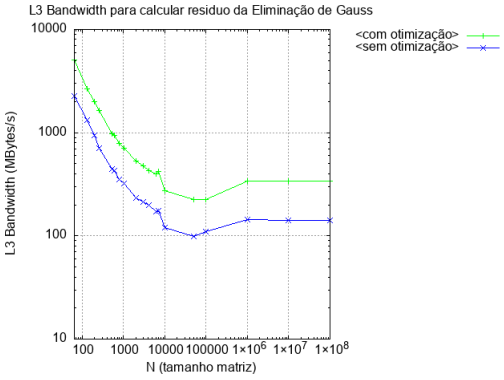

Figura 13. L3 Bandwidth para cálculo do resíduo