Relatório - Introdução à Computação Científica

Henrique Luiz Rieger e Leon Augusto Okida Gonçalves

Dezembro de 2021

1 Introdução

Este trabalho foi feito para a disciplina de Introdução à Computação Científica, ministrada pelos professores Armando Luiz Nicolini Delgado e Guilherme Alex Derenievicz.
Ele consiste em otimizações feitas em cima do código do primeiro trabalho da disciplina e em um experimento, onde comparamos os desempenhos das duas versões.
Neste texto, relataremos as melhorias feitas e exibiremos os resultados do experimento.

2 Máquina usada nos experimentos

A CPU da máquina usada nos experimentos é um Intel i7-7700 Coffeelake, com frequência de 3.60 GHz. Ela conta com 1 socket, 4 cores por socket e 2 threads por core, totalizando 8 threads. Tem 3 níveis de cache, com L1 com espaço de 32 kB, L2 com 256 kB e L3 com 8 MB. A máquina também tem 16 GB de memória RAM.

3 Como executar os experimentos

Para executar os experimentos, basta executar `./script.sh` no diretório principal do trabalho. Após isso, deve-se inserir a senha do super-usuário, o que fará com que os experimentos sejam iniciados. Por fim, deve-se aguardar o término dos experimentos. Os gráficos e resultados serão gerados automaticamente.

4 Alteração feita na contagem de tempo

Foi feita uma mudança na contagem de tempo do cálculo das derivadas parciais, para desconsiderar o tempo gasto com a alocação dinâmica da matriz. Essa alteração foi feita tanto no código antigo bem como no código otimizado.

5 Otimizações feitas

Foram feitas as seguintes otimizações:

5.1 Estrutura da matriz de derivadas parciais

No trabalho 1, ela era estruturada como um vetor de ponteiros de linhas separadas. Isso foi mudado, sendo agora um vetor único, com os acessos sendo feitos por meio de aritmética de ponteiros. Isso favorece o acesso rápido à memória, pois seus elementos estão em posições contíguas, tendo uma boa localidade de referência.
É importante notar que o tipo da matriz mudou de `void***` para `void**`. Essa otimização exigiu também a mudança de diversas partes do código, tanto em argumentos de funções bem como no acesso à matriz. A estrutura da matriz está em `vetoresOpt.c`.

5.2 Considerando a tridiagonalidade

Além das mudanças estruturais na matriz de derivadas parciais, foram feitas alterações para considerar que essa sempre seria uma matriz tridiagonal. Dessa forma, apenas são registrados dados das 3 diagonais mais internas, salvando espaço em memória.
Os cálculos da primeira linha e da última linha foram feitos fora do laço, que engloba apenas as linhas intermediárias. Isso elimina a necessidade de desvios dentro do mesmo, otimizando o uso do pipeline do processador. Esses cálculos são feitos nos arquivos `vetoresOpt.c` e `newtonOpt.c`, embora no último o endereçamento da matriz seja feito de maneira tradicional.

5.3 Padding

Foi feito padding em vetores/matrizes de tamanho da forma 2^k, a fim de evitar cache misses. Isso foi implementado com uma macro em `utils.h`.

5.4 Restrict Pointers

Foram usados restrict pointers quando possível, que fazem com que o processador entenda que não há dependências implícitas, permitindo que ele faça otimizações adicionais.

5.5 Inversão no cálculo de F(X)

Durante o cálculo de F(X), a inversão é feita *inline*, para evitar alterações desnecessárias depois. Isso é feito no arquivo `newtonOpt.c`.

5.6 Valores abaixo da diagonal principal

Os valores que estão abaixo da diagonal principal não são zerados, pois são desnecessários para cálculos com matrizes tridiagonais. Isso poupa acessos à memória.

5.7 Endereços repetidos

Os valores de endereços específicos da matriz de derivadas parciais são salvos em variáveis durante os cálculos, para evitar FLOPS desnecessários.

6 Gráficos gerados

6.1 Cálculo das derivadas parciais

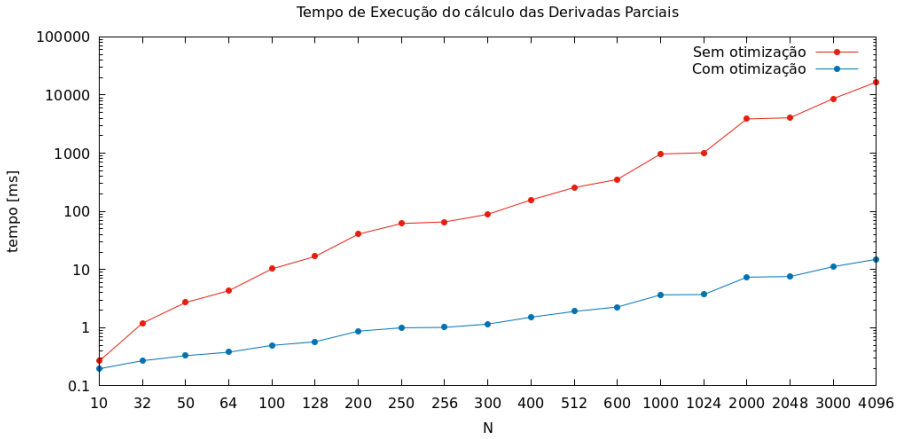

Figura 1: Gráfico do tempo do cálculo da matriz de derivadas parciais

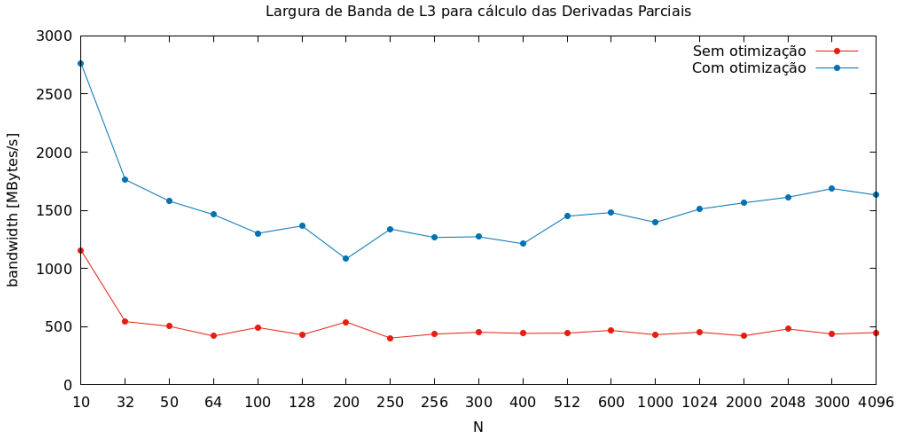

Figura 2: Gráfico da banda de memória do cálculo da matriz de derivadas parciais

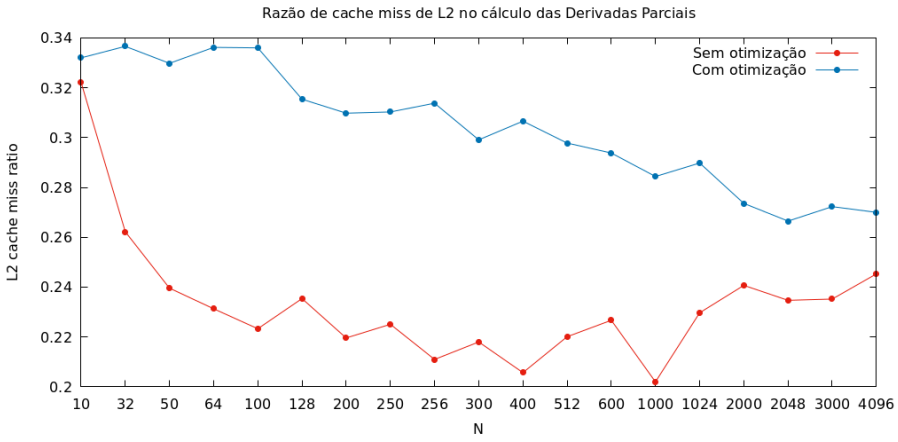

Figura 3: Gráfico de cache misses do cálculo da matriz de derivadas parciais

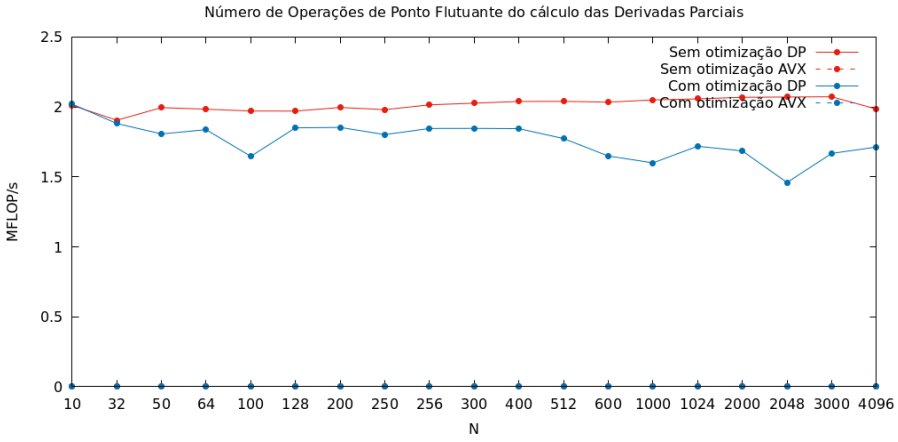

Figura 4: Gráfico de operações de ponto flutuante do cálculo da matriz de derivadas parciais

6.2 Método de Newton

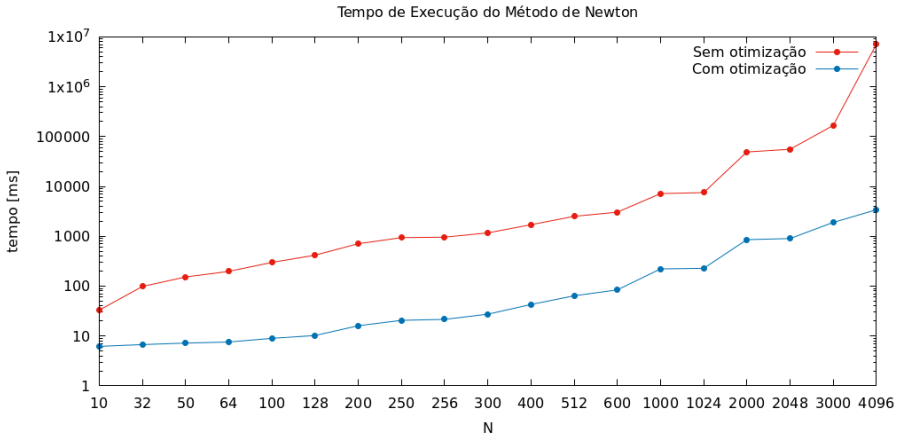

Figura 5: Gráfico do tempo do método de Newton

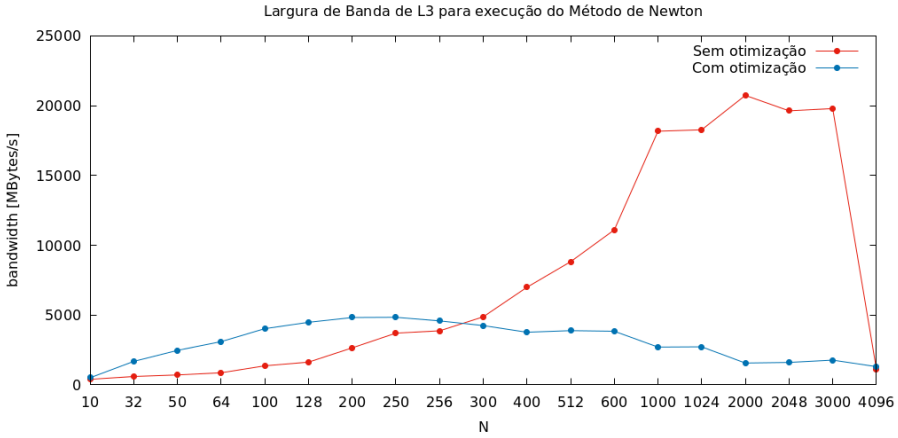

Figura 6: Gráfico da banda de memória do método de Newton

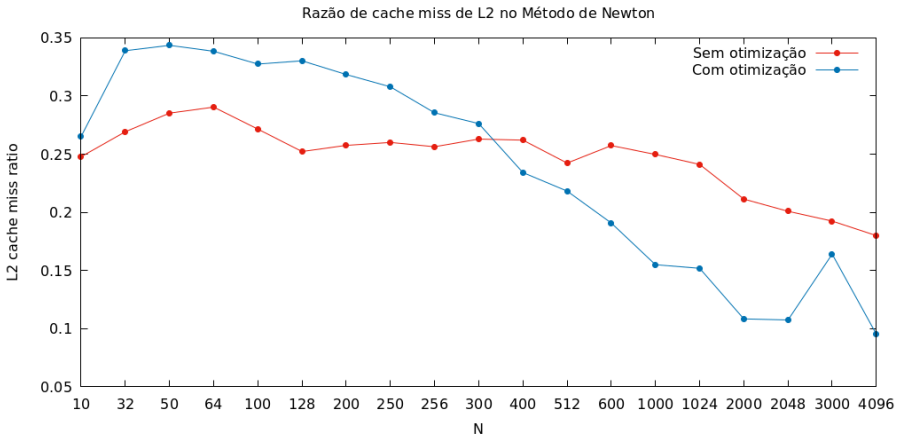

Figura 7: Gráfico de cache misses do método de Newton

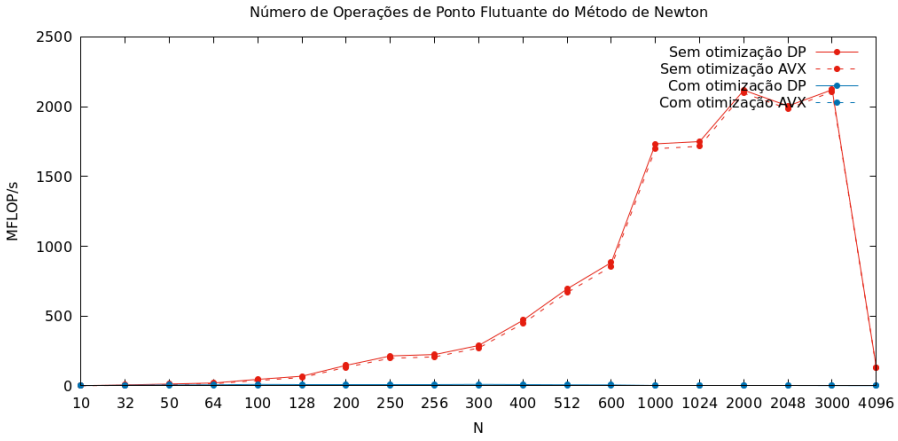

Figura 8: Gráfico de operações de ponto flutuante do método de Newton

6.3 Cálculo das Matrizes Jacobianas

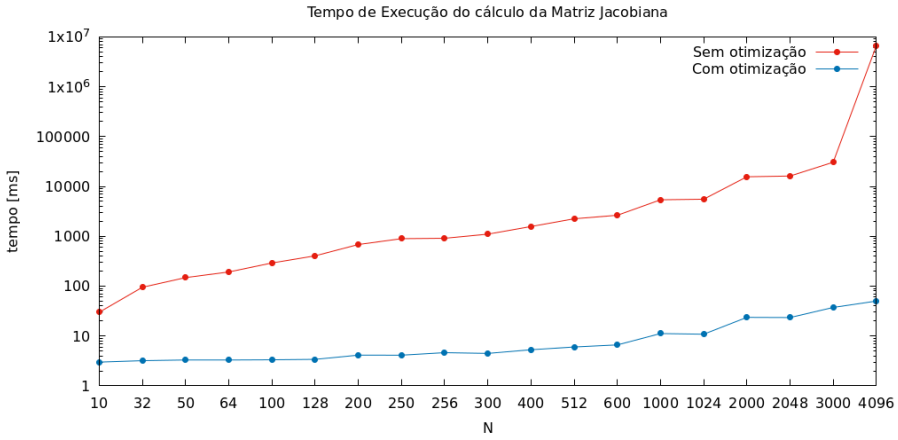

Figura 9: Gráfico do tempo do cálculo das Matrizes Jacobianas

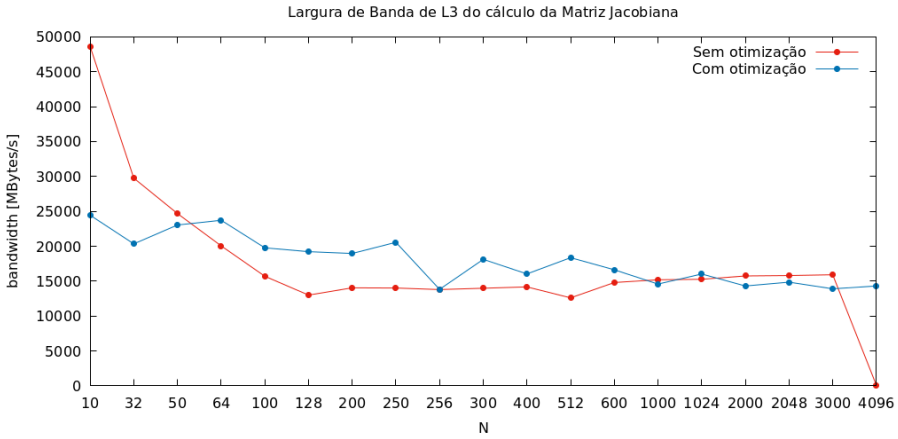

Figura 10: Gráfico da banda de memória do cálculo das Matrizes Jacobianas

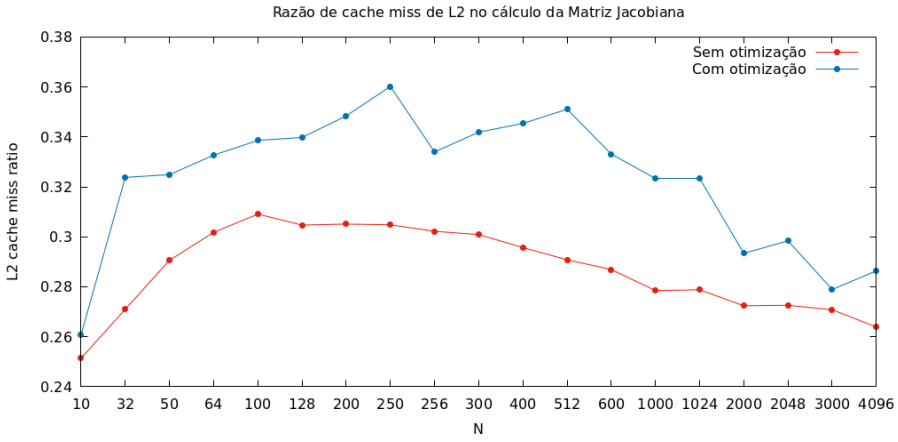

Figura 11: Gráfico de cache misses do cálculo das Matrizes Jacobianas

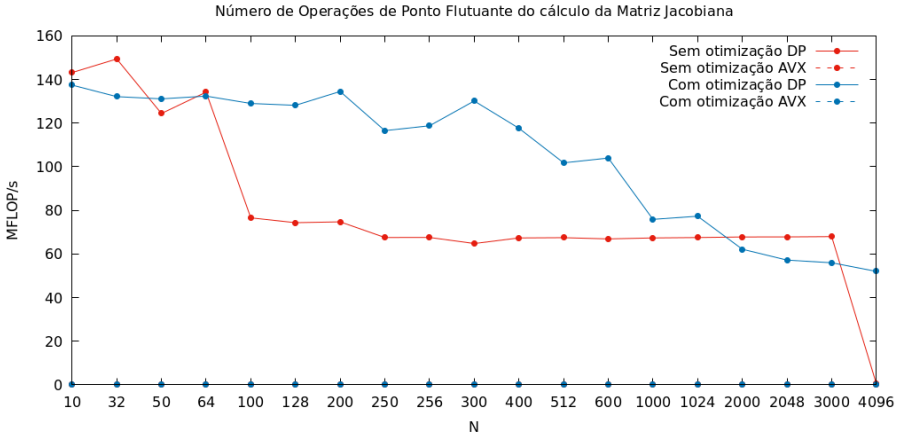

Figura 12: Gráfico de operações de ponto flutuante do cálculo das Matrizes Jacobianas

6.4 Resolução dos Sistemas Lineares

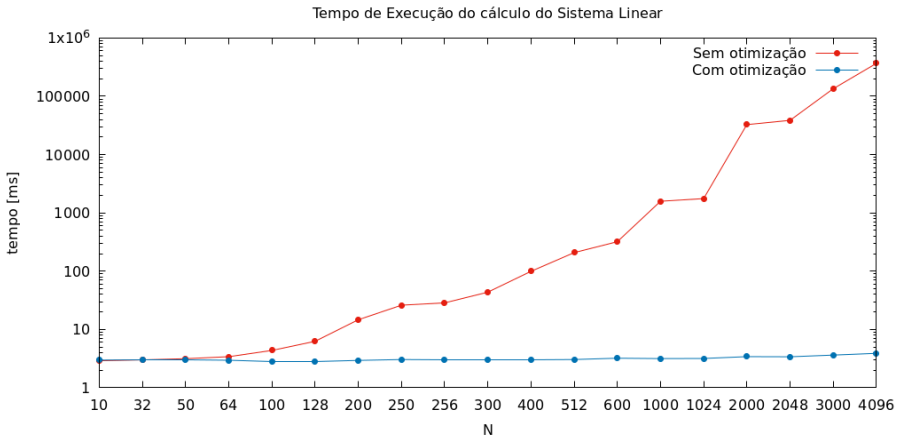

Figura 13: Gráfico do tempo da resolução dos Sistemas Lineares

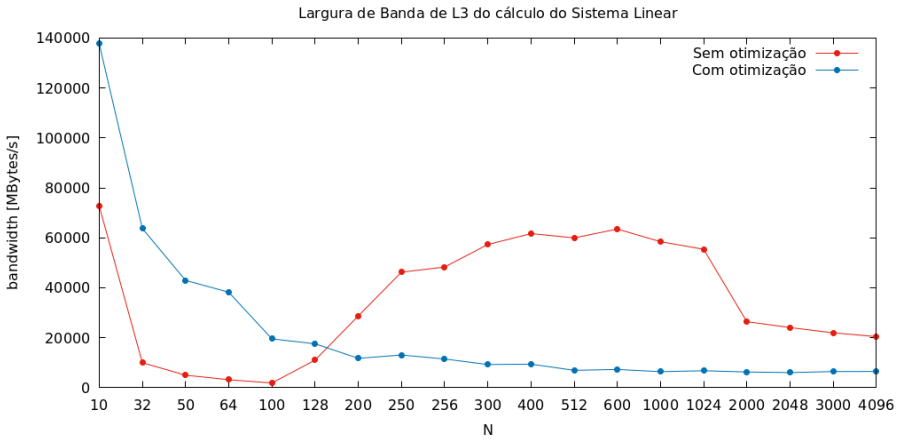

Figura 14: Gráfico da banda de memória da resolução dos Sistemas Lineares

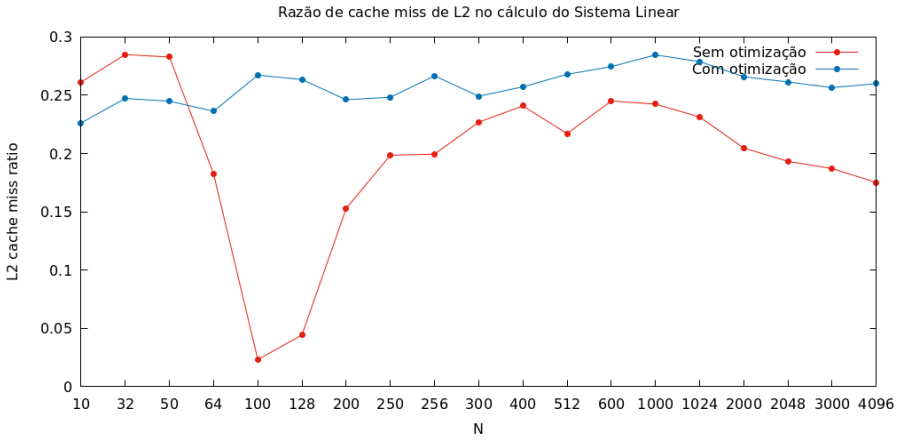

Figura 15: Gráfico de cache misses da resolução dos Sistemas Lineares

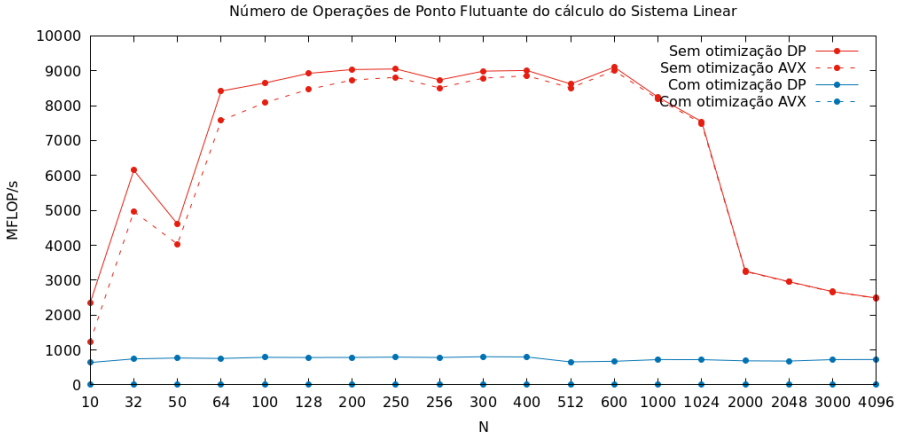

Figura 16: Gráfico de operações de ponto flutuante da resolução dos Sistemas Lineares