Trabalho 02 - Otimização de Desempenho

Caio Henrique Ramos Rufino (GRR20224386) e Frank Wolff Hannemann (GRR20224758)

I . Introdução

Este trabalho tem como objetivo principal indicar as otimizações que realizamos na versão inicial (v1) desenvolvida no trabalho 01, e realizar uma análise dos gráficos para comparar o desempenho em relação à nova versão (v2). Todos os testes de desempenho foram realizados na mesma máquina (computador do laboratório 12 do DINF), e ambas as versões foram compiladas com as mesmas flags.

II . Otimizações

Alteramos a estrutura de dados que comporta o sistema de equações para um vetor, ao invés de um vetor de vetores como estava na versão inicial.

Na geração do sistema linear houve o maior ganho de desempenho, isso porque evitamos recalcular muitos valores do sistema, que são sempre iguais nas diagonais. As equações a seguir mostram isso:

$$\alpha_1\sum(1) + \alpha_2\sum(x_i) + \alpha_3\sum(x_i^2) + \ldots + \alpha_{n+1}\sum(x_i^n) = \sum(y_i)$$
$$\alpha_1\sum(x_i) + \alpha_2\sum(x_i^2) + \alpha_3\sum(x_i^3) + \ldots + \alpha_{n+1}\sum(x_i^{n+1}) = \sum(y_i \cdot x_i)$$
$$\alpha_1\sum(x_i^2) + \alpha_2\sum(x_i^3) + \alpha_3\sum(x_i^4) + \ldots + \alpha_{n+1}\sum(x_i^{n+2}) = \sum(y_i \cdot x_i^2)$$
$$\vdots$$
$$\alpha_1\sum(x_i^n) + \alpha_2\sum(x_i^{n+1}) + \alpha_3\sum(x_i^{n+2}) + \ldots + \alpha_{n+1}\sum(x_i^{n+n}) = \sum(y_i \cdot x_i^n)$$

Os valores que possuem a mesma cor são iguais, e na versão nova do programa nós calculamos cada um deles uma única vez só, colocando esses valores em um vetor, e depois preenchemos a matriz do sistema com estes valores.

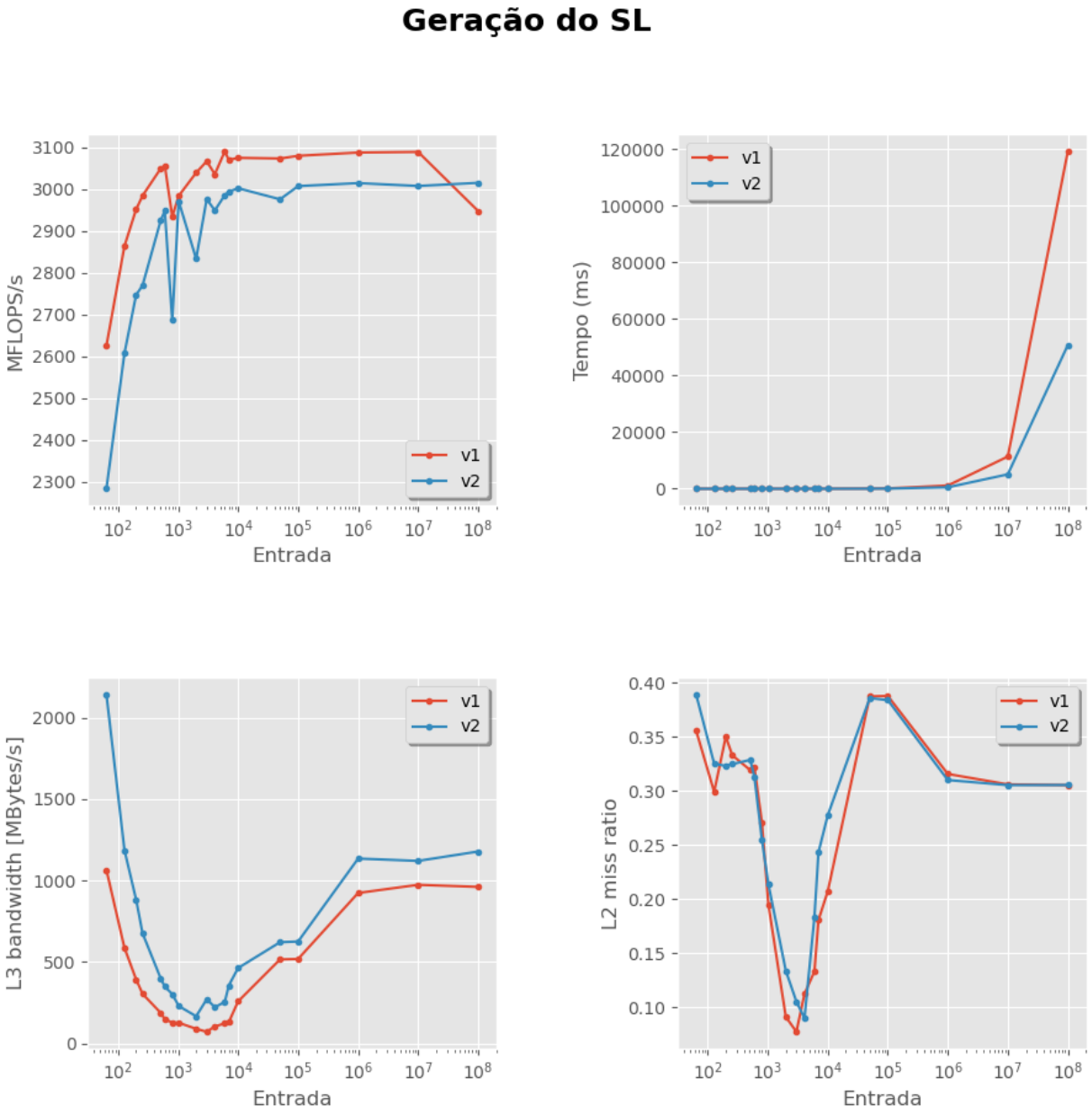
Não conseguimos fazer nenhuma otimização na solução do SL, apenas mudamos a estrutura de dados de matriz para um vetor. Como a solução da matriz é muito pequena e é só feita uma vez, mesmo otimizando a solução, não haveria um ganho significativo de performance.

No cálculo do resíduo também houve um ganho de performance significativo, pois na versão nova o uso da função expIntervalar() foi substituído por uma multiplicação cumulativa do valor de x durante as iterações que calculam o valor do polinômio. Isso é possível pois é podemos fatorar o polinômio da seguinte forma:

$$(((a_n)x + a_{n-1})x + a_{n-2})x \ldots)x + a_0$$

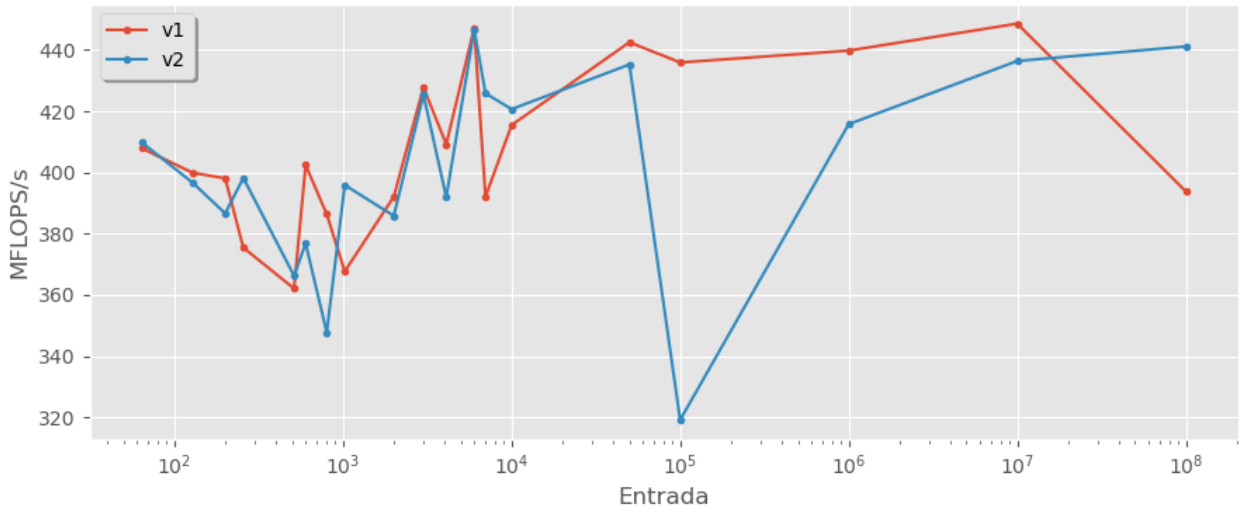
E então no programa realizamos esse cálculo em um loop partindo do último coeficiente e indo até o primeiro.

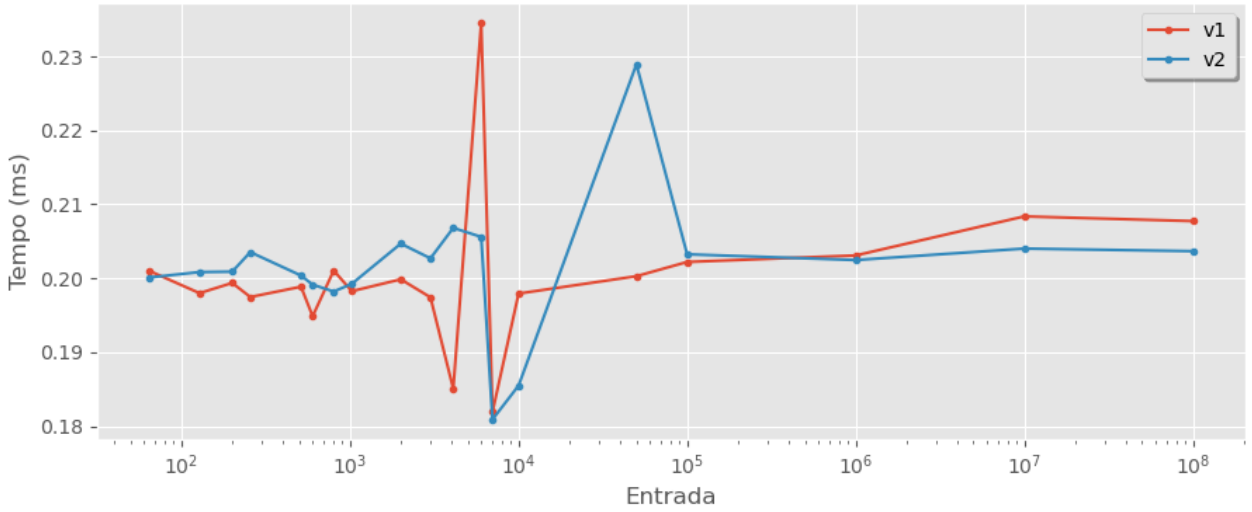
Ⅲ. Análise dos gráficos



- MFLOPS/s: não houve mudança significativa de tempo para entradas pequenas, só houve uma mudança realmente perceptível para os últimos testes.
- Tempo: podemos perceber que a redução do tempo foi significativa para entradas de tamanho grande, vide as otimizações realizadas.
- L3 Bandwidth [MBytes/s]: largura de banda melhor utilizada devido à mudança na estrutura de dados da matriz para um vetor.
- L2 miss ratio: mudanças muito pequenas, não são significativas.

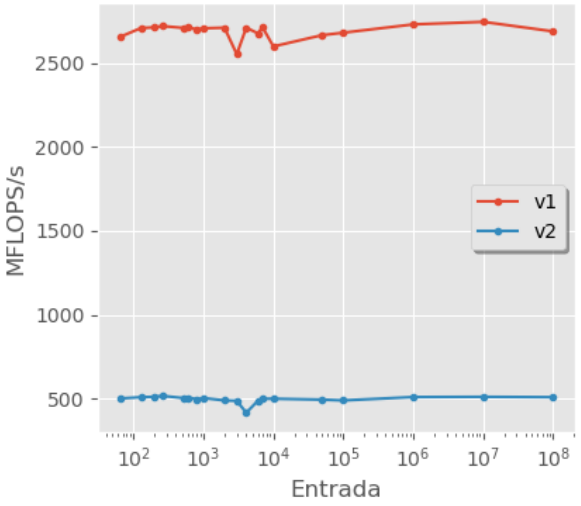
Resolução do SL

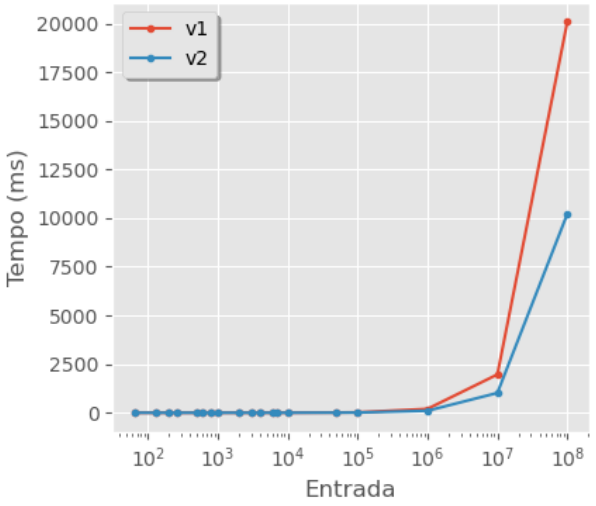


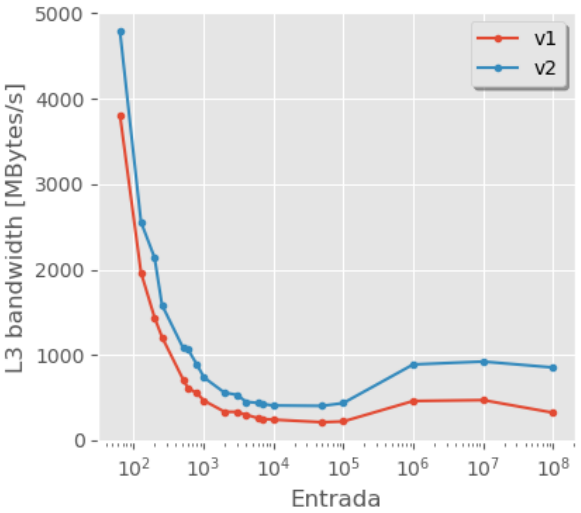


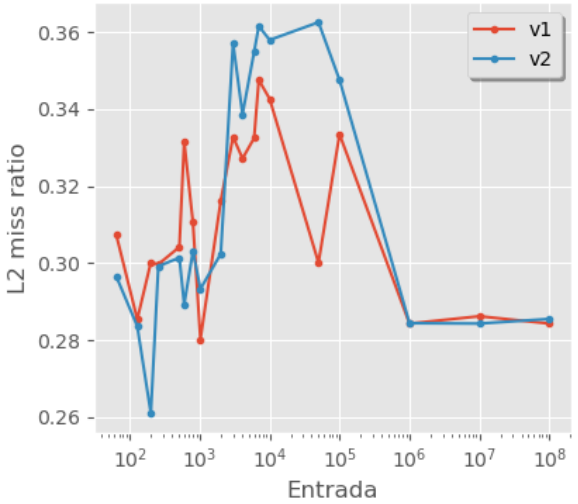
- **MFLOPS/s**: não houve nenhuma mudança significativa, afinal, essa função não sofreu mudanças.
- **Tempo**: não houve uma melhora no tempo, como dito anteriormente não conseguimos otimizar essa função.

Cálculo do resíduo









- **MFLOPS/s**: houve uma redução grande devido à alteração na forma de calcular o polinômio, que foi explicada anteriormente.
- **Tempo**: podemos perceber que a redução do tempo foi significativa para entradas de tamanho grande, vide as otimizações realizadas.
- **L3 Bandwidth [MBytes/s]**: assim como aconteceu na geração do SL, a largura de banda melhor utilizada devido à mudança na estrutura de dados da matriz para um vetor.
- **L2 miss ratio**: mudanças muito pequenas e inconstantes, não é possível tirar uma conclusão concreta.