Relatório: Trabalho 2 – Otimização de Desempenho

Gabriel Lisboa Conegero – GRR20221255
Pedro Folloni Pesserl – GRR20220072
Departamento de Informática
Universidade Federal do Paraná – UFPR
Curitiba, Brasil
glc22@inf.ufpr.br, pfp22@inf.ufpr.br

Resumo

Este relatório documenta o processo de otimização de um programa que realiza ajuste polinomial de curvas, utilizando o método dos **Mínimos Quadrados** e **Eliminação de Gauss**. Também apresenta a comparação entre as duas versões do programa, obtida a partir da ferramenta LIKWID.

1 Metodologia da análise

A análise do programa de ajuste polinomial de curvas foi feita considerando três seções principais do código, que realizam, respectivamente:

1. Geração do sistema linear pelo método dos Mínimos Quadrados;
2. Solução do sistema linear pelo método da Eliminação de Gauss;
3. Cálculo de resíduos do polinômio encontrado.

Tanto a seção de geração do sistema linear quanto a de cálculo dos resíduos do polinômio foram avaliadas com as seguintes métricas: tempo de execução, número de operações aritméticas de ponto flutuante por segundo (FLOP/s), com e sem uso de SIMD, banda de memória utilizada e taxa de *miss* na *cache* de dados. A seção de solução do sistema linear teve seu desempenho avaliado em tempo de execução e FLOP/s, apenas.

2 Otimizações realizadas

2.1 Geração do sistema linear

Originalmente, a versão 1 do programa utilizava uma tabela de *lookup* para armazenar as potências, de 0 a $2m$, dos pontos de entrada, onde m é o grau do polinômio a ser ajustado. Porém, isso precisou ser modificado, porque a entrada agora pode conter até 10^8 pontos, o que impossibilita o armazenamento das potências de todos os pontos na memória. Então, a versão 1 agora utiliza a função `pow_inter()` para calcular as potências dos x's a cada iteração.

A otimização empregada na versão 2 foi inverter a ordem dos laços no cálculo das potências, de forma que iteramos sobre o vetor de pontos de entrada apenas uma vez, possibilitando que ele seja mantido (ainda que em parte) em *cache*. Dessa maneira, percorremos a primeira linha e última coluna da matriz do sistema linear e o vetor de termos independentes múltiplas vezes, calculando a próxima potência do ponto atual que estamos somando a cada iteração com apenas uma multiplicação sobre a potência anterior.

Como o vetor de pontos é muito maior do que a matriz do sistema linear, que é sempre de ordem 5, espera-se que isso diminua a taxa de *miss* na *cache* por ser necessário recarregá-lo em *cache* menos vezes.

3 Gráficos