Primeiro Trabalho - Relatório Parcial

Computação Concorrente (MAB-117) — 2021/1 REMOTO

Aplicação concorrente no processo de Eliminação Gaussiana

Carlos A. Cozzolino R.J. e Thiago de Oliveira Silva

[1]*115086800 e 111466197*

1. Descrição do problema

A proposta deste primeiro trabalho de implementação é a de projetar e implementar uma aplicação que realize o cálculo de eliminação gaussiana em matrizes **N**x**N**. Neste trabalho o usuário deve informar a dimensão da matriz e a quantidade de threads que deseja utilizar para calcular o processo. Uma matriz de números aleatórios será gerada pelo programa junto com um vetor **"b"** de tamanho **N**x**1** e o processo de escalonamento será realizado pelas threads, onde cada uma irá efetuar uma operação diferente para no fim obtermos a matriz triangular superior. Por fim, o usuário poderá pedir para retornar a matriz resultante e o vetor **'x'** resultante, sabendo que **A.x = b**.

Observação: Cada thread realizará a leitura de uma linha e coluna para fazer a operação e pode ocorrer de mais de uma thread necessitar ler a mesma linha ou coluna ao mesmo tempo. Isto não é problema, uma vez que a leitura não altera os dados e cada thread armazena seu resultado em um local diferente. O que garante a independência de processamento.

2. Projeto inicial da solução concorrente

Para a resolução do trabalho, foram pensadas em três estratégias:

1. Uma thread lidaria com cada elemento de saída
2. Cada thread lidaria com um número contínuo de elementos
3. As threads iriam se intercalar entre as linhas da matriz.

Verificamos que o custo operacional da primeira forma citada seria muito alto e, a estratégia de intercalação seria a mais apropriada. Visto que poderíamos colocar cada identificador para atuar em uma linha específica da thread. Elegemos, portanto, a terceira opção.

A princípio, usaríamos apenas uma variável global para o número de threads que o usuário deseja utilizar, também teríamos uma estrutura para armazenar a matriz A, o vetor b, e sua dimensão. Uma função de pivoteamento poderia ser usada junto da função de escalonamento e por fim teríamos a função principal do programa. A função de escalonamento receberia os identificadores das threads e iria trabalhar com os valores dados pela estrutura, podendo retornar uma estrutura com os novos valores de A e b ou ter retorno nulo mas alterando essas variáveis da estrutura.

3. Projeto inicial dos casos de teste

Comparando os resultados sequenciais com os concorrentes, poderíamos verificar a funcionalidade do programa, outro método seria o de verificar se A.x – b = 0, visto que x seria o vetor solução para a matriz escalonada A e o vetor b.

Para verificarmos o ganho de desempenho, podemos utilizar o macro "timer.h" apresentado nos laboratórios para medir o tempo que a função sequencial e a concorrente levam para terminar seus cálculos, fazemos essas medições para matrizes cada vez maiores e teremos uma noção do ganho de desempenho com a versão concorrente ($T_{sequencial} / T_{concorrente}$).

4. Referências bibliográficas

Ruggiero, Márcia A. **Cálculo Numérico: Aspectos Teóricos e Computacionais.** 2ª Ed. Pearson Universidades. São Paulo. 2000.