Módulo 1 - Laboratório 2
Implementação e avaliação de aplicações concorrentes (parte 1)

Computação Concorrente (MAB-117) 2020.1 REMOTO
Prof. Silvana Rossetto

[1]DCC/IM/UFRJ

Introdução

O objetivo deste Laboratório é projetar e implementar uma versão concorrente para o problema de **multiplicação de matrizes**; e avaliar o desempenho da aplicação em termos de tempo de execução. Usaremos a linguagem C e a biblioteca *Pthreads*.

Acompanhe a explanação da professora nas vídeo-aulas deste laboratório. Se tiver dúvidas, entre em contato por email.

Atividade 1

Objetivo: Projetar e implementar um algoritmo concorrente para o problema de **multiplicação de matrizes** e coletar informações sobre o seu tempo de execução.

Projete um algoritmo concorrente para a tarefa de multiplicar duas matrizes quadradas (tomando como base a implementação do problema multiplica **matriz-vetor** apresentado nas vídeo-aulas deste laboratório). A entrada e saída de dados deve ser implementada do forma sequencial, apenas a parte central da multiplicação das matrizes deve ser paralelizada. Considere como **subtarefa mínima o cálculo de uma linha da matriz de saída**. Você pode escolher diferentes abordagens para distribuir as subtarefas entre as threads (por ex., cada thread é responsável por um bloco contínuo de linhas, ou por blocos de linhas intercaladas com as demais threads).

O número de threads e a dimensão das matrizes de entrada devem ser lidos da linha de comando, ou seja, o usuário do seu programa deverá poder alterar o número de threads e as dimensões das matrizes de entrada a cada execução, sem precisar recompilar o programa!

As matrizes de entrada podem ser preenchidas dentro do programa com valores fixos.

Roteiro:

1. Implemente uma solução **concorrente** do problema de multiplicação de matrizes.
2. Acrescente no seu programa chamadas da função `GET_TIME()` para medir separadamente os tempos de execução para: (a) inicialização das estruturas de dados; (b) criação das threads, execução da multiplicação, e término das threads; e (c) finalização do programa.
3. Verifique a corretude da sua solução (matriz de saída correta), usando matrizes de entrada menores.
4. Avalie o desempenho da sua solução, usando diferentes dimensões das matrizes de entrada e número de threads no programa.

Table 1. Tempo de execução com 1 e 2 threads.

Dimensão/Threads	1	2	**Aceleração**
500			
1000			
2000			

Table 2. Tempo de execução com 1 e 4 threads.

Dimensão/Threads	1	4	**Aceleração**
500			
1000			
2000			

Entrega do laboratório: Depois de certificar-se de que a solução do problema atende a todos os requisitos e executa corretamente, preencha as Tabelas (1 e 2). Registre as medidas de tempo coletadas e o cálculo do ganho de desempenho (aceleração) obtido com a versão concorrente ($T_{sequencial}/T_{concorrente}$). **Considere a execução da aplicação com uma única thread como o tempo de execução sequencial**. Faça 5 execuções de cada caso e registre a de menor tempo.

Disponibilize o código implementado na **Atividade 1** em um ambiente de acesso remoto (GitHub ou GitLab). Use o formulário de entrega desse laboratório para enviar o link do repositório do código implementado e responder às questões propostas.