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

Computação Concorrente (ICP-117) 2021.2
Prof. Silvana Rossetto

[1]Instituto de Computação/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.

As matrizes de entrada **devem** ser inicializadas dentro do programa com valores **aleatórios**. Por simplicidade, considere apenas matrizes quadradas.

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!

Roteiro para implementação e avaliação da corretude:

1. Implemente uma função que faça a multiplicação das matrizes de entrada de forma **sequencial** e armazene o resultado em uma variável a parte.
2. Projete o algoritmo concorrente para a tarefa de multiplicar duas matrizes quadradas (tomando como base a implementação do problema multiplica **matriz X vetor** apresentado nas vídeo-aulas desta semana). Você pode escolher diferentes abordagens para dividir a tarefa central entre as threads (por ex., cada thread é responsável por um bloco contínuo de linhas da matriz de saída, ou por blocos de linhas intercaladas com as demais threads).
3. Implemente o algoritmo concorrente proposto em C.
4. Verifique a corretude da sua solução (matriz de saída correta). **Implemente uma função dentro do seu programa que compare a saída sequencial e a saída concorrente**. (Como vamos trabalhar com matrizes grandes, essa função não deve imprimir as matrizes de saída mais sim varrer todas as suas células e retornar um valor único indicando se o resultado está correto ou não.)

Roteiro para avaliação de ganho de desempenho: Depois de certificar-se de que a solução do problema atende a todos os requisitos e executa corretamente, avalie o ganho de desempenho obtido. Registre as medidas de tempo coletadas e o cálculo do ganho de desempenho (aceleração) obtido com a versão concorrente ($T_{sequencial}/T_{concorrente}$).

1. Acrescente no seu programa chamadas da função `GET_TIME()` para medir separadamente os tempos de execução para: (a) a função sequencial de multiplicação das matrizes ($T_{sequencial}$); (b) de toda a parte de processamento concorrente (desde a criação das threads até o final do loop que chama `pthread_join` ($T_{concorrente}$).
2. Avalie o desempenho da sua solução ($T_{sequencial}/T_{concorrente}$), usando diferentes dimensões das matrizes de entrada e número de threads no programa:
   (a) Avalie seu programa com matrizes com as seguintes dimensões: **500, 1000 e 2000**.
   (b) **Para cada dimensão das matrizes de entrada**, avalie seu programa usando o seguinte número de threads: **1, 2, 4**.
   (c) Faça ao menos 5 execuções de cada caso e registre a de menor tempo.
   (d) Calcule o desempenho obtido em cada caso e registre todos os resultados em um arquivo PDF.
   (e) Avalie e comente no arquivo PDF se os resultados obtidos estão de acordo com o esperado. **Acrescente a informação sobre a configuração do hardware da máquina usada para a avaliação (número de processadores).**

Entrega do laboratório: 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 o arquivo PDF com os dados registrados.