Módulo 1 - Laboratório 4
Comunicação e sincronização de threads com memória compartilhada

Computação Concorrente (ICP-117) 2022.1
Prof. Silvana Rossetto

[1]Instituto de Computação/UFRJ

Introdução

O objetivo deste Laboratório é praticar o uso dos mecanismos de comunicação e sincronização por exclusão nmútua entre threads usando memória compartilhada. Usaremos a linguagem C e a biblioteca *Pthreads*.

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

Atividade 1

Objetivo: Dado um **vetor de entrada** de números inteiros, gerar um **vetor de saída** de números reais, calculando a **raiz quadrada** de todos os elementos do vetor de entrada que forem **primos**, como descrito na função abaixo:

```
void processaPrimos(int vetorEntrada[], float vetorSaida[], int dim) {
   for(int i=0; i<dim, i++) {
      if (ehPrimo(vetorEntrada[i])
         vetorSaida[i] = sqrt(vetorEntrada[i]);
      else
         vetorSaida[i] = vetorEntrada[i];
   }
}
```

Roteiro:

1. Implemente a função de inicialização do vetor de entrada preenchendo seus campos com **valores aleatórios** do tipo **inteiro**. O **número de elementos** (dim) **do vetor deve ser informado pelo usuário na chamada do programa.** *Sugestão: defina a variável* dim *do tipo* **long long int** *e use a função* **atoll()** *para converter o valor recebido do usuário (string) para long long int.*
2. Implemente uma **função sequencial** para resolver o problema e meça o seu tempo de execução.
3. Implemente uma **função concorrente** para resolver o problema e meça o seu tempo de execução. O **número de threads** também deve ser informado pelo usuário na chamada do programa.
4. Verifique a corretude da solução concorrente comparando seus resultados com os resultados da versão sequencial para o mesmo vetor de entrada.
5. Calcule o ganho de desempenho (aceleração) obtido com a versão concorrente: ($T_{sequencial}/T_{concorrente}$). Considere os seguintes valores de N: 10^5, 10^7, 10^8 (caso

tenha espaço de memória suficiente). Para a versão concorrente, experimente com 1, 2 e 4 threads. Repita a execução do programa ao menos **5 vezes** para cada configuração dos parâmetros de entrada. Use o menor tempo obtido nessas execuções para a versão sequencial e para a versão concorrente e então calcule a aceleração. Escreva os resultados de aceleração obtidos no README do código no GitHub ou GitLab.

Entrega do laboratório: Disponibilize o código implementado nas duas atividades 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.