Laboratório 4
Implementação e avaliação de aplicações concorrentes (parte 3)

Computação Concorrente (MAB-117)
Prof. Silvana Rossetto

[1]DCC/IM/UFRJ — 5 de setembro de 2019

Introdução

O objetivo deste Laboratório é praticar os conceitos estudados para implementar e avaliar um código **concorrente** para calcular a soma de uma série de valores reais que aproxima o valor de pi. Usaremos a linguagem C e a biblioteca *Pthreads*.

Para cada atividade, siga o roteiro proposto e responda às questões colocadas.

Atividade 1

Roteiro:

1. Implemente uma função sequencial para calcular o valor de pi usando a série abaixo:
   $\pi = 4 * [1 - 1/3 + 1/5 - 1/7 + 1/9 - ...]$
   O **número de elementos (N) da série** deve ser informado pelo usuário na chamada do programa.
2. Para verificar o quanto o resultado calculado se aproxima do valor de pi, compare-o com a constante M_PI (de math.h).
3. Aumente o valor de N (10^3 a 10^9) e verifique se o valor calculado se aproxima mais do valor de pi.
4. OBS.: defina a variável N do tipo **long long int** e use a função **atoll()** para converter o valor recebido do usuário (string) para long long int.

Atividade 2

Roteiro:

1. Projete e implemente uma solução concorrente para calcular o valor de pi usando a mesma série da Atividade 1. O **número de elementos (N) e de threads (T)** deve ser informado pelo usuário. Use a função `pthread_exit()` para retornar o valor calculado pela thread para a função `main()`
2. Compare o valor calculado pela função sequencial com a solução concorrente para os **mesmos valores de N**.
3. Os resultados coincidem? Por que?
4. Implemente outra função sequencial para obter o mesmo resultado da versão concorrente (mantendo a mesma ordem de soma dos elementos da série). **Considere apenas o caso de execução concorrente com 2 threads.**
5. **Os resultados coincidem agora?**

Atividade 3

Roteiro:

1. Projete e implemente **outra solução concorrente** (com outra estratégia de divisão da tarefa entre as threads) para calcular o valor de pi usando a série dada. O **número de elementos (N) e de threads (T)** deve ser informado pelo usuário.
2. Compare o valor calculado pela função sequencial com a nova solução concorrente para os **mesmos valores de N**.
3. Implemente outra função sequencial para obter o mesmo resultado da versão concorrente (mantendo a mesma ordem de soma dos elementos da série). **Considere apenas o caso de execução concorrente com 2 threads.**
4. **Os resultados da solução sequencial e concorrente coincidem?**

Atividade 4

Objetivo: Avaliar o desempenho das diferentes versões.

Roteiro:

1. Acrescente no seu programa chamadas para a função `GET_TIME()` para medir o tempo de execução do valor de pi (versões sequencial e concorrente) que dão o mesmo resultado).
2. Varie o valor de N (10^3 a 10^{10}) e compare os tempos de execução sequencial e concorrente (com 2 threads). **A partir de qual valor de N a sua versão concorrente é mais rápida que a versão sequencial?**
3. Calcule o ganho de desempenho (*speedup*) obtido com a versão concorrente ($T_{sequencial}/T_{concorrente}$) para cada valor de N ($10^3$ a 10^{10}).