Trabalho 1
Produto escalar concorrente usando threads

INE5410 - Programação Concorrente
Prof. Márcio Castro

2014/2

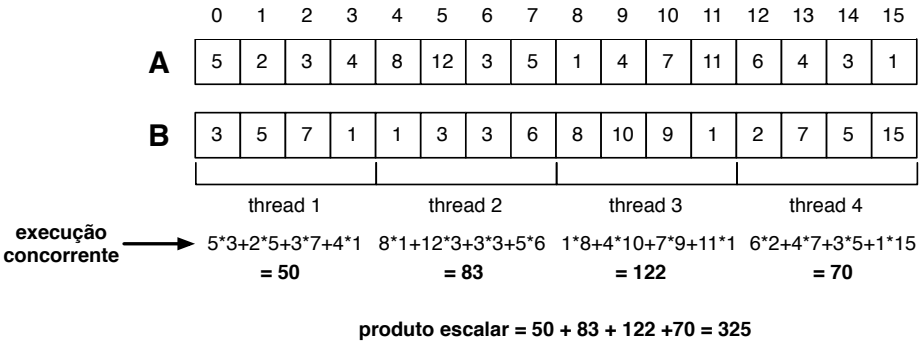
1 Definição matemática

Sejam dois vetores contendo n elementos cada um e definidos como segue: $A = \{a_1, a_2, \ldots, a_n\}$ e $B = \{b_1, b_2, \ldots, b_n\}$. O produto escalar $(\cdot)$ entre A e B é dado pela Equação 1:

$$A \cdot B = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \quad (1)$$

2 Trabalho

O primeiro trabalho da disciplina de Programação Concorrente consiste em implementar o produto escalar entre dois vetores de inteiros (obrigatoriamente do mesmo tamanho) de forma concorrente utilizando a linguagem C e a biblioteca POSIX Threads. Os dois parâmetros de entrada são o **número de threads** ($NTHREADS$) e o **tamanho dos vetores** ($VECSIZE$).



A Figura 2 mostra visualmente o funcionamento do produto escalar concorrente. Nesse exemplo, os parâmetros são $NTHREADS = 4$ e $VECSIZE = 16$.

Cada thread é responsável por realizar uma parte do cálculo. O número de elementos atribuídos a cada thread é obtido pela divisão $VECSIZE/NTHREADS$, ou seja, 4 no exemplo da Figura 2. Deverão ser tratados os casos em que o resultado da divisão $VECSIZE/NTHREADS$ não seja um número inteiro, assim como impedir a execução nos casos em que $VECSIZE < NTHREADS$.

As threads serão responsáveis por somar os seus resultados de forma concorrente em uma variável global denominada *sum*. Cada thread deverá imprimir os índices inicial e final (intervalo) atribuídos à ela, juntamente com o resultado do seu produto escalar parcial. Quando todas as threads terminarem suas execuções, a variável global *sum* deverá conter o resultado final do produto escalar. O valor da variável global *sum* deverá ser impresso na tela pela main thread ao final da execução.

3 Instruções

As seguintes instruções deverão ser **rigorosamente** seguidas para a elaboração do trabalho.

3.1 Bibliotecas

Deverão ser incluídas as seguintes bibliotecas:

```
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>
```

3.2 Parâmetros de entrada

Os parâmetros $VECSIZE$ e $NTHREADS$ deverão ser constantes definidas com #define diretamente no código como segue (exemplo com vetores de tamanho 16 e 4 threads):

```
#define VECSIZE 16
#define NTHREADS 4
```

3.3 Inicialização dos vetores

Os vetores A e B deverão ser preenchidos com **valores aleatórios entre 0 e 9** e **impressos na tela** no início da execução. Para gerar números aleatórios entre 0 e 9 em C você deverá usar duas funções:

- srand(time(NULL)): inicializa a semente (seed) do gerador de números aleatórios. Deve ser chamada uma única vez pela main thread no início da execução;
- rand() % 10: retorna um valor inteiro aleatório entre 0 e 9.

3.4 Saída

Ao final da execução, o seu programa deverá mostrar uma saída no seguinte formato (exemplo com 4 threads e vetores de tamanho 16):

A = 1, 5, 5, 7, 8, 1, 2, 9, 7, 1, 2, 3, 2, 0, 1, 4
B = 5, 2, 4, 3, 9, 4, 8, 4, 7, 0, 9, 3, 1, 4, 3, 7
Thread 2 calculou de 4 a 7: produto escalar parcial = 128
Thread 1 calculou de 0 a 3: produto escalar parcial = 56
Thread 3 calculou de 8 a 11: produto escalar parcial = 76
Thread 4 calculou de 12 a 15: produto escalar parcial = 33
Produto escalar = 293

Note que, neste caso, as threads imprimiram seus produtos parciais na seguinte ordem: 2, 1, 3 e 4. Esta ordem poderá ser diferente para cada execução, pois trata-se de um programa concorrente.

4 Grupos

O trabalho deverá ser realizado em grupos de até 2 alunos. Os alunos terão duas aulas para realizar o trabalho e deverão apresentá-lo ao professor, bem como mostrar sua solução em funcionamento. Uma apostila básica de introdução à programação em C está disponível no moodle da disciplina.

5 Avaliação

O professor irá avaliar não somente a corretude mas também o desempenho e a clareza da solução. O trabalho valerá **2,5 pontos**.