UFFS
Curso de Ciência da Computação
Sistemas Operacionais – 2020.1
Prof. Marco Aurélio Spohn

Trabalho 04

Média iterativa uni-dimensional (*One-Dimensional Iterative Averaging*)

Esse problema consiste em resolver a recorrência da média iterativa em um vetor unidimensional. Vamos assumir índices em vetores V com N elementos no intervalo de 0 a N - 1 (assim como é na linguagem C):

V[0 .. N-1]

Considere um vetor com **N + 2** elementos, tendo-se os elementos de borda (posição 0 e N + 1) com valores fixos e iguais a:

V[0] = 0;

V[N+1] = 1

e com todos os demais elementos (da posição 1 até N) com valores zero.

Assuma que há dois vetores:

Velho[N+2] // inicializado conforme descrito anteriormente

Novo[N+2]

A cada iteração, calcula-se a seguinte recorrência para todos os elementos da posição **1** a **N**:

Novo[i] = (Velho[i - 1] + Velho[i + 1]) / 2

Finalizada a rodada para todos os elementos (de 1 a N), inverte-se os dois vetores (*swap*):

Temp <-- Novo
Novo <-- Velho
Velho <-- Temp

Ou seja, faz-se um *swap* entre os dois vetores.

Repete-se o processo para um número determinado de iterações. Para um número grande o suficiente de iterações, o valor de cada elemento ***i*** converge para a respectiva média entre os valores dos vizinhos à esquerda e à direita. De outra forma, pode-se dizer que o valor na posição ***i*** converge para o valor ***i/(N+1)***.

Veja a seguir um exemplo para um vetor com N=9 (11 elementos: 9 + os dois de borda)

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.5	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.25	0.5	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.125	0.25	0.625	1.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0625	0.125	0.375	0.625	1.0
0.0	0.0	0.0	0.0	0.0	0.03125	0.0625	0.21875	0.375	0.6875	1.0
0.0	0.0	0.0	0.0	0.015625	0.03125	0.125	0.21875	0.453125	0.6875	1.0
0.0	0.0	0.0	0.0078125	0.015625	0.0703125	0.125	0.2890625	0.453125	0.7265625	1.0
0.0	0.0	0.00390625	0.0078125	0.0390625	0.0703125	0.1796875	0.2890625	0.5078125	0.7265625	1.0
0.0	0.001953125	0.00390625	0.021484375	0.0390625	0.109375	0.1796875	0.34375	0.5078125	0.75390625	1.0
0.0	0.001953125	0.01171875	0.021484375	0.0654296875	0.109375	0.2265625	0.34375	0.548828125	0.75390625	1.0
0.0	⋮									1.0
0.0	⋮									1.0
0.0	⋮									1.0
0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0

Como a cada iteração o vetor **Novo** deriva dos valores no vetor **Velho**, pode-se facilmente paralelizar o problema (i.e., dividir em fatias o vetor e computar cada fatia em paralelo, em *threads* independentes).

Objetivo

Desenvolver uma versão paralelizada como solução do problema descrito, aplicando necessariamente, o recurso de barreiras (*barriers*). O usuário deve informar, no início da execução do programa principal, quantas *threads* (o que também determinará quantas fatias) serão empregadas na execução. Além disso, deve-se informar quantas rodadas (iterações) deverão ser executadas. Exemplo (assumindo Prog <#threads> <#iteracoes>):

Prog 10 100000 (executar com **10** *threads*/fatias por **100000** iterações).

O **tamanho do vetor** poderá ser definido estaticamente durante a compilação. Deve-se considerar vetores com, no mínimo **10000** elementos (considerando os elementos de borda).

Dicas (p/ desenvolvimento do programa)

O tamanho da fatia pode ser calculado da seguinte forma:

n = N - 2;
d = numThreads; //número de *threads*
fatia = ceil(n/d); //incluir a diretiva ***-lm*** na compilação (biblioteca de *math*)

Para N = 1000 (998 + 2 de borda), com 10 *threads*, tem-se fatias de 100 elementos, com exceção da última fatia que terá 98 elementos (veja abaixo como tratar isso).
Para calcular o índice pertinente a cada fatia, pode-se adotar a seguinte abordagem:

```
        // com X threads, assumir threads com ids de 0 a X – 1
        //      t é o id da thread
        inicio = t*fatia + 1;
        fim = inicio + fatia -1;
        if(fim > (N-2)) fim = N – 2; //no caso em que a última fatia seja menor que as demais
```

Considerando o exemplo anterior, a *thread* 0 sempre manipula o vetor **Novo** no intervalo de índices de 1 a 100, a *thread* 1 manipula os elementos de 101 a 200, e assim por diante, até a *thread* 9 que manipula os elementos de 901 a 998.

Dica para emprego das barreiras:

Assumir que todas as *threads* esperam (*wait*) inicialmente na barreira 1 (*barrier1*)

```
ThreadCode(threadId) {
        t = threadId;
        ...
  for(   ) { // laço executado pelo número de iterações informado na inicialização

        s = pthread_barrier_wait(&barrier1);

                realizam sua computação pertinente
                atualizando o vetor Novo no intervalo de índices correspondente

        s = pthread_barrier_wait(&barrier2);

        //aqui apenas uma das threads deve fazer o swap entre os dois vetores (Novo e Velho)
        // lembre que apenas uma das threads terá retorno em s igual
        // PTHREAD_BARRIER_SERIAL_THREAD
  }
        ...
}
```

OBS.: para utilizar algumas das funções matemáticas deve-se incluir

#include<math.h>

bem como a diretiva ***-lm*** na compilação