Atividade 1 - Produto escalar com Pthreads

Programação Concorrente e Distribuída

aluna Rosangela Miyeko Shigenari

1 Problema

Implemente através de Pthreads uma solução para calcular o Produto Escalar entre 2 vetores (alocados dinamicamente do tipo double) de tamanho N (PE =

$$A_1.B_1 + A_2.B_2 + ... + A_N.B_N$$

).

Considerando N como 10^3 e 10^5 e o número de threads de 1 até 8.

Os vetores A e B devem ser preenchidos com valores aleatórios fora das threads.

Escreva um breve relatório contendo uma tabela com os tempos obtidos e descrição do equipamento utilizado (modelo de processador - descrever a quantidade de núcleos físicos existentes e a possível existência de hyperthreading, Memória principal, Sistema Operacional - incluindo a versão, Compilador - incluindo a versão).

A tabela de tempos deve conter o tempo total de processamento do programa para cada execução com um determinado número de threads, e também a medida de tempo gasto APENAS no trecho relativo ao cálculo do produto escalar.

2 Especificação da Máquina

Modelo do processador: 1,6 GHz Intel Core i5

Sistema Operacional: macOS Sierra versão 10.12.5

Memória : 8 GB 1600 MHz DDR3

3 Análise do Problema

Foi realizada a análise do algoritmo de acordo com o tamanho N do *array* e o número máximo de *threads* criadas, a tabela 1 mostra os resultados obtidos.

Tabela 1: Tempo de execução em relação ao tamanho do *array* e o o número máximo de *threads*

N/ Numero de threads	1	2	4	8
1000	5ms	6ms	9ms	11ms
100000	15ms	14ms	13ms	13ms

A Figura 1 mostra o trecho do código que calcula o produto escalar entre 2 vetores, com o uso de um *array* de 1000 posições e o máximo de 4 *threads*. A

```
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include <time.h>
#include <sys/time.h>

#define N 1000
#define MAX_THREADS 4

double *arrayA;
double *arrayB;
double *retorno;

void *produtoEscalar(void *tid) {
  int thid;
  long i;
  double resultado;
```

Figura 1: Trecho do código

Figura 2 mostra a execução do cálculo do produto escalar com os valores de 2 vetores gerados aleatoriamente. Sendo analisado para a comparação o tempo real de execução.

```
rosangela@MacBook-Air-de-Rosangela:~/downloads$ gcc -o produtoEscalar produtoEscalar.c -pthreadprodutoEscalar.c:2
4:40: warning: format specifies type 'long' but the argument has type 'int'
      [-Wformat]
  printf("thread=%ld, resultado=%f\n", thid, resultado);
                ~~~~                   ^~~~
                %d
produtoEscalar.c:27:1: warning: control reaches end of non-void function [-Wreturn-type]
}
^
produtoEscalar.c:54:51: warning: cast to 'void *' from smaller integer type 'int'
      [-Wint-to-void-pointer-cast]
    pthread_create(&t[th], NULL, &produtoEscalar, (void *) th);
                                                  ^
3 warnings generated.
rosangela@MacBook-Air-de-Rosangela:~/downloads$ time ./produtoEscalar
thread=2, resultado=3042377325170502533312.000000
thread=1, resultado=2757196359001801031168.000000
thread=0, resultado=2980096625122812231168.000000
thread=3, resultado=2842578535670600826888.000000
tempo decorrido: 0 milisegundos
produto escalar=1162224884496571826176.000000

real    0m0.009s
user    0m0.003s
sys     0m0.003s
rosangela@MacBook-Air-de-Rosangela:~/downloads$
```

Figura 2: Impressão do tempo de execução

4 Conclusão

A partir da análise da tabela criada com os tempos de execuções para determinados tamanhos de *arrays*, alterando também o número máximo de *threads*, se pode observar que ao aumentar o tamanho das *arrays*, a criação de mais *threads* se torna mais vantajosa pois a operação pode ser dividida entre as *threads* diminuindo o tempo para calcular o resultado total.