MC900/MO644 - Programação Paralela
Laboratório 4

Professor: Guido Araújo
Monitor: Luís Felipe Mattos

Pi de Monte Carlo

Neste laboratório, iremos fazer o exercício 4.2 de implementação do livro-texto, que é o método de Monte Carlo para calcular um valor aproximado de Pi, baseado em valores aleatórios.

O exercício do livro tem o seguinte enunciado:

Suppose we toss darts randomly at a square dartboard, whose bullseye is at the origin, and whose sides are 2 feet in length. Suppose also that there's a circle inscribed in the square dartboard. The radius of the circle is 1 foot, and it's area is π square feet. If the points that are hit by the darts are uniformly distributed (and we always hit the square), then the number of darts that hit inside the circle should approximately satisfy the equation

$$\frac{number\ in\ circle}{total\ number\ of\ tosses} = \frac{\pi}{4}$$

since the ratio of the area of the circle to the area of the square is $\pi/4$.

We can use this formula to estimate the value of π with a random number generator:

```
number in circle = 0;
for (toss = 0; toss < number of tosses; toss++) {
        x = random double between -1 and 1;
        y = random double between -1 and 1;
        distance squared = x*x + y*y;
        if (distance squared <= 1) number in circle++;
}
pi estimate = 4*number in circle/((double) number of tosses);
```

This is called a “Monte Carlo” method, since it uses randomness (the dart tosses). Write a Pthreads program that uses a Monte Carlo method to estimate π. The main thread should read in the total number of tosses and print the estimate. You may want to use long long ints for the number of hits in the circle and the number of tosses, since both may have to be very large to get a reasonable estimate of π.

Enunciado

Deve-se paralelizar o algoritmo de aproximação do valor de Pi de Monte Carlo passado. O programa deve receber como entrada 2 linhas: a primeira contém o número de threads e a segunda contém o número total de pontos que serão gerados.

Como a função rand() não é thread safe, deve ser usada a versão reentrante da função, ou seja, rand_r(unsigned int *), onde o parâmetro é um ponteiro para o estado da função em cada thread.

Os valores devem ser lidos a partir da entrada padrão (stdin) usando scanf().

Um exemplo de entrada a seguir:

```
2
10000
```

Neste exemplo, o programa deve utilizar 2 threads para calcular o valor aproximado de Pi usando 10000 pontos. O resultado deve ser impresso na saída padrão (stdout) usando printf(), no seguinte formato: a primeira linha contém o valor obtido para Pi e a segunda linha contém o tempo de execução do programa em microsegundos.

Como Pthreads não oferece uma função na biblioteca para cálculo de tempo, recomenda-se o uso da função **gettimeofday()** da biblioteca **time.h**, dependendo do sistema, a biblioteca pode ficar em **sys/time.h**. Para isso, deve-se declarar duas variáveis do tipo **struct timeval** e usar chamar a função da seguinte maneira:

```
gettimeofday(&start, NULL).
```

Para calcular o tempo de execução, usa-se o seguinte comando:

```
duracao = ((end.tv_sec * 1000000 + end.tv_usec) - \
(start.tv_sec * 1000000 + start.tv_usec));
```

O exemplo de saída a seguir:

```
3.136000
541
```

Testes e Resultado

Quanto aos testes, serão 3 testes abertos e 3 testes fechados de mesmo nível de complexidade. Teremos entradas variando entre 10000 valores e 1000000 valores.

Dada a natureza aleatória do algoritmo, para que o resultado seja considerado correto é preciso que o programa paralelo tenha algum speedup em relação ao programa serial. Assim, o código será analisado para garantir que a paralelização do algoritmo está correta.

O código deve conter comentários das passagens principais, não é preciso comentar cada linha.

Observações

- Para garantir que as variáveis tenham o tamanho correto e que não ocorra overflow, ao invés de variáveis int, use variáveis long long int
- Para os pontos aleatórios e as distâncias dos pontos até o centro da circunferência, utilize variável do tipo double
- O tempo de duração em microsegundos, deve ser um inteiro do tipo long unsigned

Submissões

O número máximo de submissões é de **25**.

Compilação

O Susy irá compilar seu programa com as seguintes flags:
-std=c99 -pedantic -Wall -lpthread -lm

Execução

A execução será feita com o seguinte comando:
./pi < arqX.in