MC900/MO644 - Programação Paralela
Laboratório 2

Professor: Guido Araújo
Monitor: Luís Felipe Mattos

Produtor/Consumidor

Neste laboratório, iremos abordar uma variação do problema do produtor/consumidor. Neste caso, todas as threads devem alternar entre produtor e consumidor entre as iterações, ou seja, quando a iteração do loop principal é par, a thread é um produtor e quando é ímpar, é um consumidor.

O produtor deve receber um valor de referência da entrada, realizar uma computação e colocar o valor no buffer para que o consumidor possa ler na próxima iteração. Já o consumidor, deve ler os valores computados e somá-los. As computações são simples, porém o mecanismo do algoritmo e a ordem em que as operações realizadas são importantes.

O código da função sequencial será fornecido nos arquivos auxiliares do Susy.

Enunciado

Deve-se paralelizar um algoritmo semelhante ao do produtor/consumidor usando os pragmas de Openmp. O programa deve receber como entrada 4 linhas: a primeira que contém um inteiro que representa o número de threads que o programa deve rodar, a segunda linha que também contém um inteiro n, **sempre par**, que é o número de iterações do loop principal, ou seja, quantos ciclos de produção/consumo ocorrerão, a terceira linha que contém um inteiro que é o tamanho do buffer e pro fim n valores que servirão como base

para o cálculo que será realizado.
Os valores devem ser lidos a partir da entrada padrão (stdin) usando scanf().

Um exemplo de entrada a seguir:

```
2
10
10
6 4 8 4 5 3 2 1 6 7
```

Neste exemplo, o programa deve utilizar 2 threads para realizar 5 ciclos do produtor, 5 ciclos do consumidor (10 iterações do loop principal) e o buffer tem tamanho de 10 valores do tipo inteiro. O resultado deve ser impresso na saída padrão (stdout) usando printf(), no seguinte formato: a primeira linha contém o resultado do cálculo e a segunda linha o tempo de execução do programa em segundos. É recomendado o uso da função *omp_get_wtime*() da biblioteca do Openmp.
O exemplo de saída a seguir:

```
495
0.000143
```

Testes e Resultado

Quanto aos testes, serão 3 testes abertos e 3 testes fechados de mesmo nível de complexidade. Teremos entradas variando entre 1000 iterações com o tamanho do buffer de 1000, até 100000 iterações o com tamanho de buffer de 10000.

Para que o resultado seja considerado correto é preciso que o valor da computação seja igual valor calculado pelo sequecial e o programa paralelo deve ter algum speedup em relação ao programa serial. Serão consideradas as duas condições para que o susy considere o programa correto dada a entrada.

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

Compilação

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

Execução

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

Tarefa Complementar

1. Executar o comando **cat /proc/cpuinfo** no linux ou **lscpu**. Este comando lista informações sobre a sua máquina. **Você deve colocar no final do arquivo a ser enviado no Susy**.

2. Executar a Compilação na máquina local utilizando a flag **-g -pg** para depois usar o gprof, que é uma ferramenta de profile que vem com o gcc. Obs: Tutorial do gprof se encontra na pagina da Monitoria em Profile.

O resultado do profile deverá também ser colocado como comentário no final do arquivo. O resultado será semelhante ao que se segue:

```
Each sample counts as 0.01 seconds.
%   cumulative self              self   total
time seconds    seconds calls s/call s/call name
33.86 15.52     15.52    1    15.52  15.52  func2
33.82 31.02     15.50    1    15.50  15.50  new_func1
33.29 46.27     15.26    1    15.26  30.75  func1
0.07  46.30     0.03                       main
```

Onde você identificará qual local que está gastando mais tempo de execução. Note que tem que executar com o programa serial. Executar este mesmo processo para cada arquivo de entrada.

3 - Executar as flag -O0, -O1, -O2 e -O3 sem usar OpenMP com o programa serial e mostrar se teve algum ganho. Informações sobre as flags de otimização do gcc está disponível no link: `https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html`. Exemplo de compilação:
gcc -O0 test.c -o test
gcc -O3 test.c -o test

Incluir para cada flag de compilação o Speedup (Speedup = TempoSemFlag/TempoComFlag) no final do arquivo em forma de comentário.

Todas informações colhidas nesta seção deverá ser incluso no final do arquivo que será submetido no SuSy. Lembrando: Em forma de comentário.