MC900/MO644 - Programação Paralela
Laboratório 3

Professor: Guido Araújo
Monitor: Luís Felipe Mattos

Histograma

Neste laboratório, iremos fazer o exercício 4.1 de implementação do livro-texto, que é o histograma mostrado no exemplo 2.7.1 do livro. Para isso, deve-se utilizar Pthreads.

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

Enunciado

Deve-se paralelizar um algoritmo de histograma utilizando Pthreads. 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, que é o número de elementos no vetor de dados, a terceira linha que contém um inteiro que é o número de bins a serem utilizadas e pro fim n valores que serão os dados do histograma. **Os valores devem ser lidos a partir da entrada padrão (stdin) usando scanf()**.

Um exemplo de entrada a seguir:

```
2
10
5
79.75 36.12 93.66 33.77 88.09 14.91 83.34 36.74 16.07 86.72
```

Neste exemplo, o programa deve utilizar 2 threads para calcular o histograma de 10 dados e 5 bins. O resultado deve ser impresso na saída padrão (stdout) usando printf(), no seguinte formato: a primeira linha contém o vetor com os intervalos entre as bins, note que são $nbins + 1$ valores, a segunda contém o vetor de frequências calculado no histograma e a terceira linha 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:

```
14.00 30.00 46.00 62.00 78.00 94.00
2 3 0 0 5
3
```

Testes e Resultado

Quanto aos testes, serão 3 testes abertos e 3 testes fechados de mesmo nível de complexidade. Teremos entradas variando entre 1000 valores e 150000 valores e de 100 bins até 1000 bins.

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.

Observações

- O vetor que contém os intervalos entre as bins, deve ser impresso com 2 casas decimais
- O valor mínimo a ser impresso no vetor de frequências deve ser o floor do vetor de dados e o máximo o ceil do vetor de dados, para isso, deve-se incluir a biblioteca math.h
- 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:
./hist < arqX.in

Trabalho Complementar

A tarefa complementar deste lab será montar uma tabela com dados de Speedup e Eficiência, conforme tabela do slide 80 do link: `http://oxent2.ic.unicamp.br/sites/oxent2.ic.unicamp.br/files/ch2_1.pdf`.

Os arquivos que serão avaliados nesta tarefa complementar são: arq1.in, arq2.in e arq3.in. Para isso, será necessário alternar o número de threads dos 3 arquivos para colher os dados no formato descrito na Tabela .1.

Tabela .1: Tabela Speedup e Eficiência

	Threads	1	2	4	8	16
arq1.in	Speedup	1				
	Eficiência	1				
arq2.in	Speedup	1				
	Eficiência	1				
arq3.in	Speedup	1				
	Eficiência	1				

Além da Tabela, deverá também ser incluso a sua **justificativa** entre as quedas na eficiência e/ou Speedup (quando houver). Use como base para a sua justificativa o conteúdo já aplicado na disciplina e informações de profiling (perf or Vtune). De preferência, justifique todos os casos.

Por fim, mostre o percentual do programa que é paralelizável.

Todos os dados coletados deverão ser inclusos no arquivo fonte em formato de comentário.