Universidade Federal do ABC
MCZA020-13 - Programação Paralela
2018.Q3

Lista de Exercícios 5

Prof. Emílio Francesquini

2 de dezembro de 2018

Lista de termos cuja definição você **deve** saber:

- Pragmas, diretivas e cláusulas OpenMP.
- No contexto da execução: time, mestre, escravo.
- Blocos `atomic`, `critical` e *locks* OpenMp
- *Loop-carried dependences*
- Particionamento em blocos e alternativas de escalonamento (*scheduling*) disponíveis em OpenMp
  - `dynamic`, `guided`, `auto` e `runtime`
  - *Chunk size* - Tamanho do "naco"
- Escopo de variáveis em OpenMP
  - Privado vs. compartilhado
  - Diretivas para definição de escopo
- Redução: variáveis e operadores

Exercícios

1. [PP] Caso o símbolo `_OPENMP` esteja definido (verificável através de `#ifdef OPENMP`) ele terá o valor de um inteiro. Escreva um programa que imprima o seu valor. O que o valor indica?

2. [PP] O OpenMP utiliza variáveis privadas para efetuar reduções. Essas variáveis são inicializadas com o valor identidade relativo ao operador escolhido. Por exemplo, se o operador escolhido for a adição então a variável de redução é inicializada com 0 enquanto caso o operador escolhido seja a multiplicação a variável é inicializada com 1. Quais são os valores identidade para os operadores `&&`, `||`, `&`, `|` e `^`?

3. [PP] O fantástico supercomputador Chespirito® é capaz de guardar variáveis do tipo `float` com até 3 dígitos decimais. Nesta incomparável máquina, cada um dos seus registradores é capaz de armazenar até 4 dígitos decimais. Contudo, após efetuada uma operação, o resultado é arredondado para três dígitos para poder ser armazenado na memória principal. Considere a declaração de um vetor `float` abaixo:

   ```
   float a[] = {4.0, 3.0, 3.0, 1000.0};
   ```

   (a) Qual é a saida do programa abaixo quando executado no computador Chesperito®?

   ```
   int i ;
   float soma = 0.0;
   for (i = 0; i < 4; i ++)
       soma += a [ i ];
   printf ("soma = %4.1f\n", soma );
   ```

   (b) Agora considere o seguinte código:

   ```
   int i ;
   float soma = 0.0;
   #pragma omp parallel for num threads (2) \
       reduction (+: soma )
   for ( i = 0; i < 4; i ++)
       soma += a [ i ];
   printf ("soma = %4.1f\n", soma );
   ```

   Suponha que o sistema de execução atribui as iterações `i = 0, 1` para o thread 0 e `i = 2, 3` para o thread 1. Qual é a saída deste código quando executado no Chesperito®?

4. [PP] Escreva um programa OpenMP que determina a configuração padrão de escalonamento de laços `for`. A entrada do seu programa deve ser o o número de iterações e a sua saída deve ser quais iterações foram executadas por cada um dos threads. Por exemplo, se há dois threads e quatro iterações, uma possível saída seria:

   ```
   Thread 0: Iterations 0 -- 1
   Thread 1: Iterations 2 -- 3
   ```

5. [PP] Nossa primeira tentativa de paralelizar o programa para cálculo de π falhou, ela não dava valores consistentes com a versão sequencial. De fato, utilizamos a própria versão sequencial para mostrar que a versão paralelizada não estava funcionando conforme esperávamos. Explique por que podemos acreditar que a versão sequencial é a que produz resultados corretos e não a versão paralela.

6. [PP] Considere o laço abaixo:

```
a [0] = 0;
for ( i = 1; i < n ; i ++)
    a [i] = a [i - 1] + i ;
```

   Há uma dependência entre as iterações do laço (*loop-carried dependence*). É possível eliminar esta dependência e paralilizar o laço?

7. [PP] Em C uma função que recebe uma array bi-dimensional (matriz) também precisa receber o número de colunas. Por este motivo é relativamente comum que programadores avançados linearizem a matriz (ou seja, transformem a matriz em um vetor unidimensional) e escrevam código que, explicitamente, convertam os índices bi-dimensionais em um índice unidimensional para acessar a posição correta no vetor. Escreva um código em OpenMP que recebe uma matriz linearizada e um vetor e que devolva o resultado da multiplicação da matriz pelo vetor. Parelelize a sua solução usando OpenMP. Consulte o Capítulo 5 do livro do Peter Pacheco para uma versão base do algoritmo.

8. [PP] Embora a função da biblioteca padrão `strtok_r` seja thread-safe, ela tem a péssima propriedade de modificar a string de entrada. Escreva a sua própria versão de `strtok_r` que não modifique a string original e que seja thread-safe.

9. Utilizando OpenMP implemente uma fila thread-safe com múltiplos produtores e múltiplos consumidores. Meça o desempenho para 1000 inserções e para 1000 remoções utilizando 64 threads. Meça o desempenho para o mesmo conjunto de operações e o mesmo número de threads. Como os resultados se comparam? Você é capaz explicar as diferenças encontradas?

10. Utilizando OpenMP implemente um pipeline de 3 estágios onde a comunicação entre cada um dos estágios é feita através de uma fila. Considere primeiro o caso onde cada estágio do pipeline é executado por um único thread. Em seguida altere o seu código para que ele seja capaz de trabalhar com um número arbitrário de threads em cada estágio. Que mecanismos você utilizou para a sincronização? O que o fez escolher um mecanismo dentre os demais disponíveis?