Acadêmica: Jênifer Vieira Goedert

1. Quais são as limitações no uso de estrutura de repetição em programação paralela e quais estruturas não podem ser utilizadas.

O uso de estrutura de repetição em programação paralela só é possível se o número de iterações for previamente conhecido e não sofrer alteração durante a execução (por exemplo, o *for*). Não se pode utilizar esse construtor em estruturas do tipo *while* ou *do-while*.

2. Explique como funciona o *#pragma omp for*.

Esse construtor faz com que as iterações da estrutura de repetição situada logo abaixo da diretiva sejam executadas em paralelo. As iterações são distribuídas entre as threads no grupo criado na região paralela onde o laço se encontra. Nele é implementado o SIMD (Single Instruction Multiple Data), ou seja, cada thread executa as mesmas instruções sob um conjunto diferente de dados.

As cláusulas que podem ser utilizadas no construtor for são as seguintes:

a) private (lista de variáveis)
b) firstprivate (lista de variáveis)
c) lastprivate (lista de variáveis)
d) reduction (operador: lista de variáveis)
e) schedule (tipo, [,tamanho do chunk])
f) nowait
g) ordered

3. Qual construtor implementa o MIMD (Multiple Instruction Multiple Data) e explique seu funcionamento.

O construtor *sections* implementa MIMD. É utilizado para dividir tarefas entre as threads em blocos de código que não possuem iterações. Dessa forma, cada thread executa uma instrução diferente sob um conjunto diferente de dados. Alem do construtor *sections*, que indica que a thread irá executar um bloco de instruções diferentes, é necessário incluir no código a diretiva *#pragma omp section*, que indica qual a instrução que cada thread irá executar.

As cláusulas que podem ser utilizadas nesse construtor são as seguintes:

a) private (lista de variáveis)
b) firstprivate (lista de variáveis)
c) lastprivate (lista de variáveis)
d) reduction (operador: lista de variáveis)
e) nowait

4. Explique como funciona o #pragma omp single.

Esse construtor indica que o trecho de código abaixo dessa diretiva deve ser executado apenas por uma thread, não necessariamente a thread mestre, mas a primeira thread que atingir esse ponto de execução. As outras threads esperam numa barreira implícita, no final do construtor single, até que a thread que encontrou o construtor termine a execução.

As cláusulas que podem ser utilizadas no construtor single são as seguintes:

a) private (lista de variáveis)
b) firstprivate (lista de variáveis)
c) copyprivate (lista de variáveis)
d) nowait

Link do GitHub para os códigos das próximas questões: https://github.com/Jenifer19IFC/OpenMP-atividades/tree/main/aula4ativ

5. **Cálculo Paralelo da Soma de um Vetor** - Implemente um programa que calcule a soma dos elementos de um vetor de inteiros utilizando a diretiva *#pragma omp for*. Divida o trabalho entre múltiplas threads para somar partes do vetor em paralelo e combine os resultados parciais em uma única soma total. Utilize a redução (reduction) com a diretiva *#pragma omp for* para acumular a soma de forma segura.

6. **Inicialização de Matriz com Tarefa Única** - Implemente um programa que inicialize uma matriz 2D de inteiros com valores aleatórios. Use a diretiva *#pragma omp single* para garantir que apenas uma thread execute a função de inicialização do gerador de números aleatórios antes de preencher a matriz em paralelo com *#pragma omp for*. Certifique-se de que a função de inicialização do gerador de números aleatórios seja executada apenas uma vez.

7. **Processamento Paralelo com Seções** - Implemente um programa que processe duas tarefas independentes em paralelo utilizando a diretiva *#pragma omp sections*. Uma seção deve calcular a soma dos elementos de um vetor, enquanto a outra seção deve calcular o produto dos elementos de outro vetor. Utilize *#pragma omp sections* para definir as seções e *#pragma omp section* para as tarefas individuais.

8. **Multiplicação de Matrizes com Divisão de Tarefas** - Implemente um programa para multiplicar duas matrizes quadradas. Use *#pragma omp sections* para dividir a multiplicação em várias tarefas, onde cada seção será responsável por calcular um subconjunto das linhas da matriz resultante. Utilize a diretiva *#pragma omp for* dentro de cada seção para paralelizar o cálculo das linhas. Assegure-se de que cada seção lide com um conjunto distinto de linhas da matriz resultante.

9. **Busca Paralela em Vetor** - Implemente um programa que realize a busca de um valor específico em um vetor de inteiros utilizando *#pragma omp for* para paralelizar a busca. Além disso, use *#pragma omp single* para imprimir a posição do valor encontrado na primeira ocorrência, garantindo que apenas uma thread execute a impressão. Utilize uma variável compartilhada para armazenar a posição do valor encontrado e uma cláusula *#pragma omp critical* para proteger a atualização desta variável.