Universidade Federal do Rio Grande do Norte
Instituto Metrópole Digital
IMD1116 - COMPUTAÇÃO DE ALTO DESEMPENHO - T01 (2025.1)
Tarefa 6: Escopo de variáveis e regiões críticas

Docente: SAMUEL XAVIER DE SOUZA
Discente: Iago Gabriel Nobre de Macedo (20220037927)
Link do repositório da atividade no Github: https://github.com/IagoGMacedo/IMD1116-Computacao-de-Alto-Desempenho/tree/main/exercicios/escopo_variavel_area_critica

Implementação inicial e resultado incorreto

A implementação inicial utiliza o método de Monte Carlo para estimar o valor de pi, paralelizando o laço principal apenas com a diretiva #pragma omp parallel for. No entanto, essa abordagem não trata adequadamente o acesso concorrente à variável dentro_circulo, o que resulta em uma condição de corrida. Como consequência, diferentes threads podem tentar atualizar essa variável simultaneamente, levando a erros na estimativa final de pi. O código referente a esse cenário pode ser consultado em: https://github.com/IagoGMacedo/IMD1116-Computacao-de-Alto-Desempenho/blob/main/exercicios/escopo_variavel_area_critica/Main_cenario1.c

Implementação refatorada

Na segunda abordagem, o algoritmo foi aprimorado com o uso de diretivas e cláusulas do OpenMP para garantir a correta atualização das variáveis compartilhadas e melhorar o desempenho. O uso da diretiva #pragma omp critical garante que apenas uma thread por vez possa atualizar a variável dentro_circulo, eliminando a condição de corrida observada anteriormente. A diretiva #pragma omp parallel delimita a região paralela, permitindo que múltiplas threads sejam criadas para executar o bloco de código. Já a diretiva #pragma omp for distribui as iterações do laço entre as threads, acelerando o processamento ao dividir o trabalho.

No que diz respeito ao escopo das variáveis, a cláusula private assegura que cada thread possua sua própria instância de determinadas variáveis, como a semente do gerador de números aleatórios (seed). Isso é fundamental para que cada thread produza sequências independentes de números aleatórios, evitando conflitos. A cláusula firstprivate inicializa as variáveis privadas com o valor original antes da região paralela, o que pode ser útil para garantir que todas as threads partam de um mesmo valor inicial, como o número total de pontos a serem processados. Já a cláusula lastprivate permite que o valor da variável na última iteração do loop seja preservado ao final da execução paralela. Por outro lado, a cláusula shared indica que a variável será compartilhada entre todas as threads, como ocorre com total_pontos e dentro_circulo, que representam informações globais necessárias para o cálculo.

Embora não tenha sido utilizada na implementação, vale destacar a importância da cláusula default(none). Ao exigir que o programador declare explicitamente o escopo de todas as variáveis dentro da região paralela, essa cláusula contribui para evitar ambiguidades e possíveis erros, especialmente em programas mais complexos. Dessa forma, o uso de default(none) incentiva uma análise mais criteriosa do escopo das variáveis, tornando o código mais seguro e fácil de manter.

O código correspondente à implementação aprimorada pode ser encontrado em: https://github.com/IagoGMacedo/IMD1116-Computacao-de-Alto-Desempenho/blob/main/exercicios/escopo_variavel_area_critica/Main_cenario2.c.