Relatório da Tarefa 07

Utilização de tasks

20210093987 - Francisca Paula de Souza Braz
Data da entrega: 22/04/2025
Disciplina: DCA3703 - PROGRAMAÇÃO PARALELA - T01 (2025.1)

Introdução

Nesta tarefa, foi implementado um programa em linguagem C que cria uma lista encadeada, onde cada nó armazena o nome de um arquivo fictício. O objetivo principal foi aplicar conceitos de paralelismo com OpenMP para percorrer essa lista e processar cada nó utilizando a diretiva #pragma omp task. Cada tarefa criada imprime o nome do arquivo e o identificador da thread responsável por sua execução.

Inicialmente, a criação e o processamento das tarefas ocorreram dentro de uma região paralela sem controle adequado de acesso à estrutura encadeada. Essa abordagem resultou em comportamentos incorretos, como a repetição de nós, omissão de elementos e processamento exclusivo pela thread 0, devido a condições de corrida (race conditions).

Para resolver esses problemas e garantir que cada nó fosse percorrido apenas uma vez, foram testadas duas abordagens distintas: o uso das diretivas #pragma omp single e #pragma omp master. Ambas garantem que apenas uma thread percorra a lista e crie as tarefas, evitando conflitos de acesso ao ponteiro da lista. Com isso, foi possível distribuir corretamente as tarefas entre várias threads, garantindo que cada nó fosse processado exatamente uma vez e de forma paralela.

Durante a execução do programa, foi observado que o comportamento pode variar entre execuções, a depender da forma como o agendador do OpenMP distribui as tarefas entre as threads disponíveis. No entanto, ao garantir que a criação das tarefas ocorra de maneira controlada (por apenas uma thread), o processamento dos nós se torna estável, sem repetições ou omissões.

Metodologia

A implementação foi dividida em três etapas principais: construção da lista encadeada, paralelização com OpenMP e análise do comportamento das tarefas.

Foi definida uma estrutura de nó contendo um campo para o nome do arquivo e um ponteiro para o próximo nó. Em seguida, a função append_node foi utilizada para adicionar 10 arquivos fictícios à lista, nomeados sequencialmente como arquivo1.txt até arquivo10.txt.

Para explorar o paralelismo, foi utilizada uma região paralela com a diretiva #pragma omp parallel, onde o objetivo era percorrer a lista encadeada e criar uma tarefa para cada nó utilizando #pragma omp task.
Inicialmente, a lista foi percorrida por múltiplas threads simultaneamente, o que causou comportamento incorreto devido a condições de corrida. Posteriormente, foram testadas duas abordagens para controlar esse acesso:

- #pragma omp single: apenas uma thread qualquer percorre a lista e cria as tarefas.
- #pragma omp master: somente a thread mestre é responsável por percorrer a lista e gerar as tarefas.

O programa foi executado diversas vezes, observando o identificador da thread responsável por cada tarefa. Foram analisados:

- A quantidade de tarefas criadas e executadas;
- Se algum nó foi processado mais de uma vez ou ignorado;
- A distribuição das tarefas entre threads.

Esses testes permitiram avaliar a influência das diretivas de controle (single e master) sobre a execução correta e paralela das tarefas, além de identificar boas práticas para evitar erros em contextos paralelos com estruturas encadeadas.

Resultados

Durante os testes realizados, foi possível observar diferenças significativas no comportamento do programa dependendo da forma como as tarefas foram criadas dentro da região paralela.

Quando usamos o #pragma omp task diretamente no parallel:
As tarefas foram criadas diretamente por múltiplas threads, sem uso de single ou master, observou-se que:

- Alguns nós foram processados mais de uma vez e alguns foram ignorados;

- Todas as tarefas foram executadas pela mesma thread 0;

- Houve comportamento inconsistente entre diferentes execuções.

Isso ocorreu devido ao acesso concorrente à variável current, o que causou condições de corrida durante a iteração da lista encadeada.

Ao utilizar a diretiva single, apenas uma thread escolhida automaticamente pelo runtime percorre a lista e cria as tarefas.

- Todos os nós foram processados exatamente uma vez;
- As tarefas foram distribuídas entre diferentes threads;
- O comportamento foi mais estável e previsível entre execuções;
- A desempenho foi satisfatória, com ganho de paralelismo.

Utilizando a diretiva master, garantiu-se que apenas a thread principal fosse responsável pela criação das tarefas. Os resultados foram similares à abordagem com single, observou-se que:

- Todos os nós foram processados corretamente;
- As tarefas foram executadas por múltiplas threads;
- O desempenho foi adequado, e o controle sobre a criação foi mais explícito.

Essa análise evidencia a importância de controlar o acesso a estruturas de dados compartilhadas dentro de regiões paralelas, especialmente ao utilizar listas encadeadas, onde o ponteiro de navegação precisa ser manipulado com exclusividade.

Conclusão

A atividade proposta consistiu em implementar um programa em C que utiliza uma lista encadeada para armazenar nomes de arquivos fictícios e, dentro de uma região paralela, criar tarefas com #pragma omp task para processar cada nó individualmente. Através da execução e testes do programa, foi possível refletir sobre o comportamento do OpenMP em cenários com estruturas dinâmicas e tarefas concorrentes.

Observou-se que, sem o uso de mecanismos de controle como #pragma omp single ou #pragma omp master, o acesso concorrente à lista resultou em processamento incorreto: alguns nós foram processados mais de uma vez, enquanto outros foram ignorados. Além disso, a distribuição de tarefas entre as threads foi ineficaz, concentrando a execução em uma única thread.

Ao aplicar #pragma omp single ou #pragma omp master, foi possível garantir que cada nó fosse processado exatamente uma vez e por uma única tarefa, mantendo a integridade da iteração sobre a lista e permitindo a real paralelização do processamento.

Portanto, conclui-se que o comportamento do programa muda entre execuções quando o acesso à estrutura encadeada não é controlado. Para garantir que cada nó seja processado apenas uma vez e por apenas uma thread, é essencial restringir a criação de tarefas a uma única thread, utilizando diretivas adequadas de controle de fluxo dentro da região paralela.