Trabalho de Programação Concorrente

Eduardo Souza Paixão

1 Introdução

A fim de praticar os conceitos expostos durante o decorrer da disciplina, será apresentada a implementação da estrutura de dados lista encadeada de forma concorrente, utilizando a linguagem *C++*.

2 Descrição do problema

A estrutura a ser implementada consiste em uma sequência de nós encadeados formando uma lista, estes nós são compostos por um dado, e um ponteiro contendo a referência para um nó seguinte.

De acordo com as especificações do projeto, a lista deverá executar as operações de inserção, remoção e busca, utilizando grupos de threads específicos para cada uma das operações. Bem como mecanismos sincronização, visando eliminar os problemas de starvation e deadlock em consequência do compartilhamento de recurso pelas threads.

3 Implementação

Para a implementação, foi criada uma classe "Linked_list" reponsável por implementar uma lista encadeada simples. Em sequida foi desenvolvida a classe "Concurrent_linked_list" que herda Linked_list e encapsula seus métodos aplicando as threads e os mecanismos de sincornização.

A sincronizção das threads se deu pelo uso do método de exclusão mutua, que foi utilizado por meio da classe "lock_guard" que atua protegendo uma região crítica até o final do seu escopo de execução. Assim, evitando problemas como deadlock e starvation.

3.1 Inserção

Esta operação se dá por meio da adição de um nó no final da lista. Para evitar que duas threads possam realizar esta operação ao mesmo tempo foi feito o uso do lock_guard, restringindo o acesso a esse metodo a uma threads por vez.

```
void Concurrent_linked_list::concurrent_insert(int data) {
  std::lock_guard<std::mutex> lock(m_mutex);
  insert(data);
}
```

3.2 Remoção

Ao passar um número aleatório para esta função, se este número representar a posição de um elemento na lista, este elemento será removido. Semelhante ao caso da inserção, neste método também foi aplicado o lock_guard.

```
void Concurrent_linked_list::concurrent_remove(int position) {
  std::lock_guard<std::mutex> lock(m_mutex);
  remove(position);
}
```

3.3 Busca

Essa operação consiste na busca por um determinado elemento no campo de dados dos nós da lista encadeada. Diferentemente dos dos métodos anteriores, a busca não altera o estado da lista, fazendo com que não seja necessário aplicar nenhuma restrição nas threads que executrão esta função.

```
void Concurrent_linked_list::concurrent_search(int data) {
  search(data);
}
```

4 Considerações

Eu não entendi bem, como funcionaría a política de justiça descrita entre as threads dos grupos inserção e remoção na descrição do projeto, mas pelo que eu pude apurar, ambos os grupos de threads estão cumprindo suas funções sem interferir umas nas outras.

No mais, foi bastante interessante ver como os mecanismos de sincronização funcionam e como a disputadas das threads por recursos compartilhados pode gerar incosistencias graves no funcionamento de um software.