

Fundamentos de Sistemas Paralelos e Distribuídos - 2020/1

Alunos:

- Felipe dos Santos Leão Ribeiro
- Francisco de Paula Dias Neto

Trabalho Prático 1

Proposta

A proposta deste trabalho consistia na implementação de uma versão paralela eficiente do [algoritmo de árvores binárias do professor Nívio Ziviani](#). Além disto, foi requisitado que implementássemos uma versão adaptada dos testes fornecidos por este, assim como consta na função `main` do código anteriormente citado. Os testes seriam, então divididos em três fases: inserção de elementos; remoção de um elemento + pesquisas na árvore + reinserção do elemento removido; e remoção final dos elementos. Cada uma dessas fases deve ser controlada com uma barreira, fazendo com que o programa só prossiga para a próxima fase após a execução de todas as threads da fase atual. Além disto, é necessário imprimir o cálculo de operações por segundo ao final de cada fase.

Implementação

Enquanto a lógica da árvore ficou definida nos arquivos do `binaryTree`, a lógica dos testes e a instância das threads foi definida no arquivo `main`. Para a lógica da árvore:

- Cada nó da árvore possui seu próprio mutex. Isso possibilita que nós individuais sejam travados, permitindo o fluxo normal de processamento nos outros nós enquanto um deles é modificado.
- Uma versão adaptada do algoritmo de leitores e escritores foi utilizada: cada nó tem um contador da quantidade de leitores simultâneos e um mutex para acesso a esse contador, além do mutex geral para escrita. Nenhum leitor precisa acessar o mutex geral para acessar os dados do nó, mas acessam o mutex do contador para realizar incremento e decremento da quantidade de nós simultâneos acessando aquele nó. Quando o primeiro leitor acessa um nó, ele aumenta o contador e bloqueia o mutex geral. Quando o último leitor está prestes a sair do nó, ele diminui o contador (voltando a 0) e libera o mutex geral, de forma que, se houvesse um escritor esperando para remover ou inserir alguma informação daquele nó, ele teria que acessar

esse mutex geral e esperar que todos os leitores terminassem sua leitura antes de poder escrever. Este método foi implementado com prioridade para leitores.

- Para garantir o desempenho e o acesso paralelo, o mutex não bloqueia a árvore toda ao realizar operações de escrita. Ele percorre cada nó realizando um lock, verificando se o nó atual é diretamente afetado na operação e, caso negativo, realiza um unlock e passa para o próximo. O unlock é realizado antes de percorrer a árvore uma vez que percorrer antes de dar unlock, por ser uma função recursiva, iria resultar no bloqueio de uma branch inteira da árvore. A única situação em que a árvore é percorrida antes de realizar o unlock é quando um filho de um nó folha deve ser acessado, uma vez que não haverá um mutex instanciado para este filho. Neste caso, o nó folha é bloqueado para que a operação seja realizada nele, impedindo que inserções no nó folha possam ser sobrescritas por outras inserções ou uma remoção.

Para a lógica dos testes e da barreira, temos:

- A barreira foi implementada em cima da barreira existente na biblioteca de pthreads.
- Um vetor com MAX posições é criado para armazenar os dados que serão inseridos na árvore, sendo um vetor com valores exclusivos de 1 a MAX cuja ordenação é randomizada a cada execução do código.
- Existem NUM_THREADS threads definidas para o programa.
- Cada thread, em cada fase, fica responsável por MAX/NUM_THREADS elementos do vetor. A função que a thread chama utiliza o índice da thread para descobrir qual partição dos dados ela deve acessar.
- Ao lidar com os MAX/NUM_THREADS dados, a thread chama a barreira, que irá esperar com que todas as threads tenham finalizado para prosseguir com o encerramento de sua execução.
- Antes de iniciar a execução de uma thread é iniciado um contador, que é finalizado ao final da execução da thread. Este contador é utilizado no cálculo do número de operações por segundo de cada fase. Na primeira fase é considerado que são realizadas MAX operações de inserção. Na segunda fase é considerado que são realizadas MAX operações de inserção, MAX operações de remoção e MAX * MAX operações de pesquisa, totalizando MAX * (MAX + 2) operações. Na última fase é considerado que são realizadas MAX operações de remoção.