Centro de Informática - UFPE
Lista de Exercícios — Concorrência — IF686 (23.1)

(0,4) 1. Defina uma classe ArvoreBusca que implementa uma árvore de busca onde é possível realizar inserções de elementos. Essa estrutura de dados deve funcionar com várias threads. Faça o que é pedido:
   (a) Implemente um método main() que cria 50 threads onde cada uma insere 2000 números aleatórios nessa árvore. Seu programa deve informar a quantidade de nós total da árvore após todas as inserções
   (b) Meça o tempo de execução do seu programa, comparando-o com o de uma execução puramente sequencial.

(0,4) 2. Modifique o programa do exercício anterior para tornar mais fina a granularidade do travamento. Em outras palavras, faça com que o travamento seja feito por nó, ao invés de afetar a árvore inteira. Compare o desempenho desta versão com o da sequencial e o da que usa apenas uma trava.

(0,4) 3. Modifique a sua implementação de árvore para que também inclua uma operação de remoção de nós. Faça com que as threads que você cria, que antes faziam apenas operações de inserção, agora também façam operações de remoção. Meça o desempenho.

(0,4) 4. Modifique sua abordagem de travamento para garantir que, se duas threads estão tentando percorrer, inserir ou remover nós em sub-árvores diferentes, elas podem fazer isso em paralelo (caso o hardware permita). Em outras palavras, não pode mais usar travamento global.

(0,4) 5. Implemente uma classe chamada CountDownLatch. Essa classe deve ter dois métodos, await() e countDown(). Seu construtor recebe um número inteiro positivo. Se uma thread chama o método await() de um objeto desse tipo, ela fica bloqueada esperando até que o contador chegue a zero. Chamadas a countDown() decrementam o contador interno do objeto, caso ele seja maior que zero. Se o contador chegar a zero, todas as threads que chamaram (ou chamarem) await() são desbloqueadas.