Homework #3: Concurrent Data Structures 개발/디버깅

Due on Monday, Dec 4 at 10:30am (before the lecture)

(Part 1) (이후 수정되거나 part 2가 추가될 수 있으며, part 2가 추가되는 경우 due date는 바뀌지 않습니다)

이번 숙제에서는 fine-grained lock (hand-over-hand lock)을 사용하여 여러 thread가 동시에 접근할 수 있는 Binary Search Tree를 구현하도록 합니다. 숙제에 첨부된 single-thread용 BST 에 hand-over-hand lock을 insert, delete, search 함수에 대해서 구현하세요. 구현한 concurrent BST에 대한 성능 분석을 하고 이를 리포트에 설명하세요.

- **제출**

완성된 코드는 gitlab 의 본인 repository에 "ParBST" 로 submit 하세요. 리포트는 수업시간에 제출하도록 하세요.

- **리포트**

리포트에는 아래 내용이 기술되어 있어야 합니다.

1. BST에 구현한 fine-grained lock에 대해서 operation 별로 (insert, delete, search) 간략히 설명하세요.
2. 구현한 BST를 어떤 식으로 테스트 했는지 기술하세요. 구현한 BST에 여러 thread가 동시에 접근하여 insert, delete, search 를 할 때 올바르게 동작하는지 어떻게 테스트 하였는지 설명하세요.
3. 구현한 BST의 성능을 다음과 같이 분석하세요. 본인이 실험한 컴퓨터의 스펙 (core 갯수, 메모리 크기, 캐쉬 크기 등)을 기술하고 스펙이 어떻게 성능에 영향을 미치는지 함께 분석하세요.
    a. 100만개의 랜덤한 숫자를 BST에 insert할 때 thread 1개, 2개, 4개, 8개로 나누어서 insert할 경우 실행시간이 어떻게 되는지 그래프로 그리고 설명하세요.
    b. 100만개의 랜덤한 숫자를 BST에 insert한 후 추가로 100만개의 insert/search operation을 thread 1,2,4,8개로 실행할 때 실행시간에 대해서 그래프로 그리고 설명하세요. Insert 와 search 비율은 1:1, 1:4, 1:9로 해서 실험하세요 (search 가 더 많게).
    c. 구현에 사용한 lock을ReadWriteLock으로 바꿔서 b의 실험을 반복하고 성능이 어떻게 달라지는지 설명하세요.

(Part 2)

Part 2에서는 lock-free data structure를 구현하도록 합니다. Part 2는 두가지 옵션이 있는데, 수업시간에 배운 lock-free (sorted) linked list를 구현하거나, lock-free binary search tree를 구현하도록 합니다. Lock-free BST는 수업시간에 배운 leaf-oriented BST로 구현하면 됩니다. 둘중 하나를 구현하도록 하고, insert, delete, search 함수를 구현해야 합니다. 세 함수는 동시에 실행 가능해야 합니다 – 예를 들어 insert 나 delete 도중에 search가 있더라도 잘못된 결과를 리턴하거나 하면 안됩니다. 구현한 내용에 대해서 part 1과 마찬가지로 성능 분석을 하고 리포트를 작성하세요.

- **제출**

완성된 코드는 gitlab 의 본인 repository에 "LF_LL"(linked-list인 경우) 혹은 "LF_BST" (bst 인 경우) 로 submit 하세요. 리포트는 수업시간에 제출하도록 하세요.

- **리포트**

리포트에는 아래 내용이 기술되어 있어야 합니다.

1. 구현한 함수들(insert, delete, search)에 대해서 간략히 설명하세요.
2. 구현한 data structure를 어떤 식으로 테스트 했는지 기술하세요. 여러 thread가 동시에 접근하여 insert, delete, search 를 할 때 올바르게 동작하는지 어떻게 테스트 하였는지 설명하세요.
3. 구현한 data structure의 성능을 다음과 같이 분석하세요. 본인이 실험한 컴퓨터의 스펙 (core 갯수, 메모리 크기, 캐쉬 크기 등)을 기술하고 스펙이 어떻게 성능에 영향을 미치는지 함께 분석하세요.
    a. 100만개의 랜덤한 숫자를 insert할 때 thread 1개, 2개, 4개, 8개로 나누어서 insert할 경우 실행시간이 어떻게 되는지 그래프로 그리고 설명하세요.
    b. 100만개의 랜덤한 숫자를 insert한 후 추가로 100만개의 insert/search operation을 thread 1,2,4,8개로 실행할 때 실행시간에 대해서 그래프로 그리고 설명하세요. Insert 와 search 비율은 1:1, 1:4, 1:9로 해서 실험하세요 (search 가 더 많게).
    c. BST를 구현한 경우, part 1에서 구현한 fine-grained BST와 비교해서 insert와 search 비율이 달라질때 성능에 대해서 분석하고 설명하세요.

리포트는 한글로 작성하며, 총 **7**장이 넘지 않도록 합니다.