Term Project - rbtrist

과목명	리눅스시스템응용설계
교수님 성함	손용석 교수님
학과	소프트웨어학부
학번	20186085
이름	Team4

1. 아이디어 배경

- rbtree 는 root 에서 시작하기 때문에 전체가 critical section 이 되어야 함.
- 즉, 트리의 변경이 일어날 때 tree 전체가 lock 이 잡힘.
  (이유: recoloring 이 끝나야만 새로운 노드를 삽입해야 하기 때문)
- 따라서, rbtree 에 바로 multi threading 을 적용하면 locking overhead 때문에 오히려 성능이 좋지 않음.

■ **rbtree**

```
[  198.530749] insert time: 11746449 ns
[  198.530754] search time: 25715793 ns
[  198.530756] delete time: 13201085 ns
[  198.530757] Bye Module
```

■ **multi threading rbtree**

```
[  439.706617] rb insert time: 17213454 ns
[  439.706622] rb search time: 24186052 ns
[  439.706624] rb delete time: 9334076 ns

[  439.706625] Bye Module
```

➔ parallelism 효과를 얻기 위해서 critical section 을 tree 전체에서 더욱 작은 단위로 축소할 수 있는 방법을 모색

➔ tree 의 leaf node 는 그 자체로 구간을 나누는 역할을 수행할 수 있고 이를 이용하면 데이터를 삽입할 때, fine grained 하게 lock 을 잡을 수 있을 것이라 판단

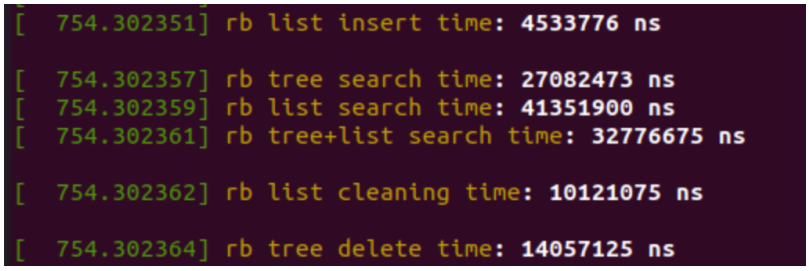
➔ insert 연산 시에 recoloring 을 미루고, 적절한 leaf node 찾아 linked list 로 node 를 연결하는 것으로 해당 leaf node 만 lock 을 잡도록 함.

➔ 새로 추가되는 insert workload 가 leaf node 에 따라 분산되고, 각 workload 들은 자신이 들어갈 leaf node 에 대한 lock 만 획득하면 되므로, multithreading 의 이점을 얻을 수 있음

```
39 struct my_tree {
40     struct rb_node node;
41     struct list_head entry;  // insert node 를 연결해줄 linked list
42     spinlock_t insert_lock;
43
44     int key;
45     bool check;              // leaf node 에 linked list 가 달려있는지에 대한 check variable
46 };
```

2. 결과

■ **multi threading rbtrist** (thread 생성 시간이 포함된 insert time 이다.)

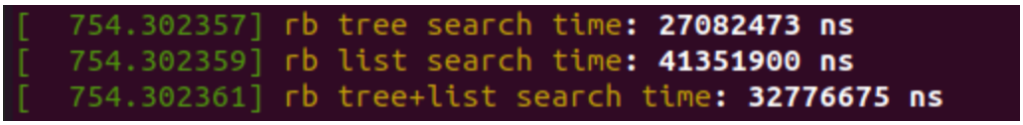


- 기존 rbtree insert time 보다는 약 2.6 배 성능 개선
- multi threading rbtree insert time 보다는 약 3.8 배 성능 개선

3. rbtrist implementation detail

A. insert 시간을 줄이기 위해서 search time 의 희생이 크지 않음
leaf node 들이 구간을 나누어 data 를 분배하는 역할을 수행하고 실제로 리스트의 길이가 크게 길어지지 않기 때문.

B. rbtrist 에서 tree 안에 있는 data 만 search 하는 것과 비교했을 때,
list 안에 있는 data 만 search 하는 것은 1.6 배의 시간이 소요되고
list 와 tree 에 있는 data 를 반반 search 하는 것은 1.28 배만 증가됨.
list 안의 data 만 찾는 것이 최악의 경우이고, 실제로 그럴 확률은 높지 않음
따라서 rbrist 는 search 시간을 약간만 희생하는 것으로 큰 insert 시간을 향상시킬 수 있음



- 하지만 이러함에도, 혹시나 너무 리스트가 길어지는 것을 방지하기 위해서 cleaning 을 구현.
- 일정 threshold 를 넘어갔을 때, rb_cleaning 함수를 호출하여 list 가 지나치게 길어지는 것을 방지하고자 함.
- recoloring 의 시간이 길지 않아서 cleaning 에 드는 시간도 상대적으로 길지 않고, 이 cleaning 을 data structure 가 idle 할 때 미리미리 수행한다면 더욱 효율적인 활용 가능.