TinySTM B plus Tree

Name: Beiming CUI

1. Test report of the TinySTM

1.1 Bank

First, I test all default setting for the bank program. Then I changed the number of threads. I run the threads number such as 4, 8 and 16.

As the number of threads grows the txs is smaller and smaller. So we could find that different threads run concurrently well.

Thread numbers	time
1	11060205
4	11016022
8	8231760
16	5599689

1.2 Intset

First, I test all default setting for the bank program. Then I changed the number of threads. I run the threads number such as 4, 8 and 16 with different implementations.

Thread numbers	Intset-hashset	Intset-linkedlist	Intset-red black tree	Intset-skiplist
1	145587573	9588823	55877277	40729158
4	180956952	8211852	64252205	35213659

8	156777894	4320833	65916300	56975560
16	130093645	3160749	65495965	38096035

1.3 Regression

First, I test all default setting for the regression program. Then I changed the number of threads. I run the threads number such as 4, 8 and 16. Second, I changed the rate irrevocable percent from 25%, 50% and 75%. I found that with the same number of threads, the ratio of Number of successful irrevocable-serial executions to the Number of successful irrevocable-parallel executions is become smaller and smaller. It changed from 3.0504, 2.9873, 2.8124.

1 Introduction to B Plus Tree

Because of I do not have enough time to implement the concurrency B plus tree. I only implement the serial B plus tree.

Object is to implement a B plus tree which has search(), insert(), and remove() 3 API.

- insert(x) adds x to the tree. Returns true if x was absent, and false otherwise.
- remove(x) removes x from the tree. Returns true if x was present, and false otherwise.

- search(x) returns true if x is present, and false otherwise.

2 Implementation of a B Plus Tree

A B plus tree is contained by a class called BTNode. There are two kinds of BTNode. The first one is BTInternal and the second one is called BTLeaf. BTInternal is the internal node in the tree. It has the pointers point to its child nodes and the keys which represent the child's value. And it also has a field called parent, which is used to point to its parent.

BTLeaf is the leaf node in the tree, which contains all the key values in the tree.

For the three functions, search, insert and remove, we use recursion way to implement. We find the target from the tree root and use the recursive way to find the leaf node which contains the target node.

For search, we just find whether the target is in the leaf node.

For insert, we need to find the target leaf node position to find where we should insert the node, then in some case we need to split the node to make sure that the elements in the node is correct.

For remove, we need to find the target leaf node position to find where we should remove the node, then in some case we need to delete the whole node or we need to merge the target node with its neighbor to make sure that the elements in the node is correct.

3 Data Structure

BTInternal has two field, the first one is an array called all_keys and the second one is a array of pointers called all_pointers.

BTLeaf has one field, which is an array of keys called all_data.

4 Test Performance:

Based on the test driver, I test the three functions search(), insert() and remove(). Each time, I will check the print out of the tree to check whether the function is working right as predicted.

For delete function, I have following several test cases.

For leaf node:

Case#1 delete in leaf, which leaf has enough elements

Case#2.1 leaf does not have enough elements, borrow from neighbor and neighbor has enough nodes to borrow

Case#2.2 leaf does not have enough elements and neighbor does not have enough nodes to borrow, merge two leaf nodes

For internal node:

Case#1 delete key in internal node, and internal node has enough elements

Case#2.1 internal node does not have enough elements and borrow from neighbor

Case#2.2 internal node does not have enough elements and borrow from neighbor, and neighbor also does not have enough, we need to merge.

For root:

Case#1: root is become empty

Case#2: a internal node become a new root

For insert function, I have following several test cases.

For leaf node and internal node:

Case#1 add in leaf or internal, which leaf and internal has enough space

Case#2.1 do not have enough space, split into two nodes and parent has space

Case#2.2 do not have enough space, split into two leaf and parent does not have space, need do the recursive