B+Tree Concurrency Control

Yundi Bao, yundib
Yingqi Zhang, yingqizh

Link
https://t7nirvana.github.io/Concurrent-BPlusTree/

Current Completion
We first implemented the sequential version of B+Tree. This is basically done by a node class that preserves its own sorted list and a tree class that organizes the nodes. The list within each node class is sorted by keys and internal nodes have pointers to the children along with the sorted keys and leaf nodes have values paired with the sorted keys. The tree class has the root node as its member and traverses down the nodes to find the position to insert or read values.

The coarse-grained lock-based implementation is straightforward and we only need to add a lock to the tree class and lock accordingly every time accessing or modifying the tree. The fine-grained lock-based implementation is a bit more complicated since all nodes need an additional lock and the access methods need modification. We made some minor changes to the existing version and used inheritance to reduce the amount of code replication, which took some time.

We now leverage on the main function to do some simple correctness tests. We need further design of the dataset and access pattern before gaining some meaningful measurements.

Revised Plan and Schedule
The B+Tree implementation itself is harder than we expected and we spent some time figuring out the APIs that fit our needs best. So we decided not to implement the Scan() interface, which is a traversal of leaf nodes. Theoretically this should be just a more significant difference in performance than that of the normal reads and writes. Also we don't think we will have time to explore the SIMD execution in our lock-free implementation. Besides those bonus features(the "nice to haves") we mentioned in our proposal, everything else is still as planned.

The revised and more detailed schedule is as below:

Week	**Task**	**Status**
11.1 - 11.7	Proposal, Background Research	done
11.8 - 11.14	B+Tree, Lock-based Implementations	done
11.15 - 11.21	Lock-free Implementation	delayed
Milestone		

11.22 - 11.25	Yundi: Current B+Tree Implementation Refactor	
11.26 - 11.28	Yundi & Yingqi: Lock-free Implementation	
11.29 - 11.30	Yingqi: Benchmark Environment	
12.1 - 12.4	Yingqi: Benchmark, Evaluation	
12.5 - 12.8	Yundi & Yingqi: Final Measurement and Report	
12.9 - 12.10	Yundi & Yingqi: Poster	

Poster Session
Since our project is basically evaluations of different concurrency control mechanisms, we will show graphs during the poster session. Specifically we will provide flow charts to demonstrate our implementations of B+Tree, and plots of our performance measurements.

Concerns
So far our biggest challenges are the coding part, and we believe we are able to make it work.