Lock-free Skip List

Team Member

Chen He
Yida Wu

URL

https://supertaunt.github.io/CMU_15618_project.github.io/

Summary

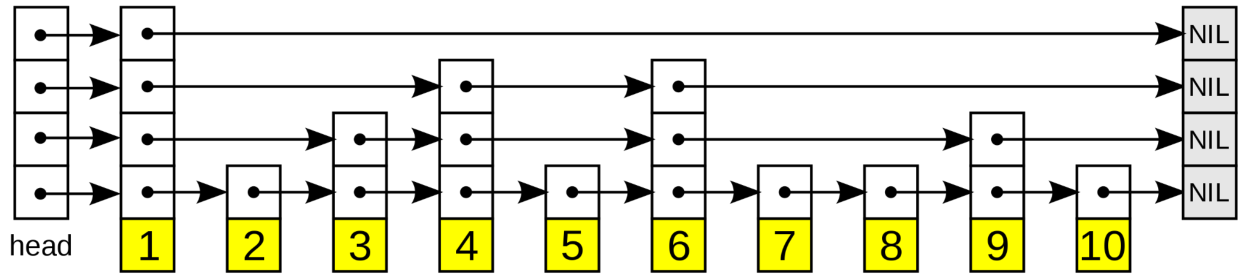
We are going to implement a coarse-grained, fine-grained and lock-free skip list and then compare the performance between them.

Background

Skip list is a relatively novel data structure and it has O (log n) search, insert and delete cost while keeping the elements in order. Skip list in a single thread version is easy to implement and programmer friendly. However, it becomes challenging when it comes to implementing a parallel versioned lock free skip list.

In this project, we will implement different versions of the parallel skip list that allow users to do search, insert, and delete in parallel. In addition, we will test our implementation on various parallel workloads and on different numbers of core count, and evaluate its advantages and disadvantages.

The updating operations (e.g. insert and delete) will benefit a lot from the parallel skip list. The reason is that, compared to many other rebalancing data structures like red black tree, the critical part of skip list is much smaller. While performing updating operations, it only requires knowledge from local nearby nodes. This should enable the skip list to maintain an excellent performance in massive parallel circumstances.



The Challenge

The main challenge we predict is to figure out an appropriate way to parallelize the updating operations. Skip list is similar to a multi-level linked list. In order to insert/delete a node at the bottom, the upper-level nodes must be handled properly as well. This will bring the main constraints for concurrency. Similar to the lock-free linked list we learned in class, lock-free delete is estimated to be harder than lock-free insert. Thus, we need to compare the ideas in the locked-free linked list and from research papers to figure out the most appropriate way. We may need to implement multiple versions to test the performance. Locality will be another challenge because the skip list, similar to linked list, can not utilize the cache efficiently. This will bring challenges when we try to improve the efficiency at the end.

Resources

We already found some research papers in this area, including "Parallelizing Skip Lists for In-memory Multi-core Database Systems", "A Provably Correct Scalable Concurrent Skip List" and "A Lock-free Skip List with Batch Updates and Snapshots." We will investigate the ideas in these research papers, compare them with what we learned about the lock-free linked list in class and figure out the most appropriate way.

Goals and Deliverables

Plan to Achieve

● A functioning coarse-grained skip list.

● A functioning fine-grained skip list.

● A functioning lock-free skip list.

● An analysis report regarding the performance and implementation.

If the progress of our project is slower than expected, we will follow the above sequence and promise to finish the coarse-grained and fine-grained skip list, and make some efforts on lock-free version.

Hope to Achieve

● Try different strategies to implement the lock-free skip list and compare the performance.

● Lock-free skip list could achieve close to linear speedup compared to single thread version in case workload operates on different parts of the list.

● The lock-free skip list could make some improvements over the fine-grained locked skip list, because CAS has less cost than a lock when contention is low.

Demo plan to show at poster session:

● We plan to show a comparison of performance across coarse-grained, fine-grained, and lock free skip list implementation for different workloads.

● We plan to show an analysis of strengths and weaknesses of lock free skip list in different operations (e.g. search, insert, delete). And list the best circumstance that lock free skip list fits in.

Platform Choice

We will implement different versions of the parallel skip list in C++. Experiments will be conducted mainly on the GHC machines.

Schedule

Week	**Task**
Nov 4 - Nov 8	Investigate on the research paper and discuss the idea.
Nov 9 - Nov 15	Implement a coarse-grained and fine-grained version of skip list.
Nov 16- Nov 24	Implement lock-free skip list and work on the milestone report
Nov 25 - Dec 2	Conduct performance evaluation and improve the lock-free version
Dec 3 - Dec 9	Analysis and finish final project report