# Lock-free Skip List

## Team Member

Chen He
Yida Wu

## Schedule

| Week | Task | Progress | Assigned |
|---|---|---|---|
| Nov 4 - Nov 8 | Investigate on the research paper and discuss the idea. | Done | Together |
| Nov 9 - Nov 22 | Implement a coarse-grained version of skip list. | Done | Chen He |
| | Implement  the contain function in the fine-grained skip list. | Done | Yida Wu |
| | Implement  the insert function in the fine-grained skip list. | Done | Yida Wu |
| | Implement  the delete function in the fine-grained skip list. | In Progress | Yida Wu |
| | Implement insert function in lock-free skip list | Done | Chen He |
| | Implement delete function in lock-free skip list | In Progress | Chen He |
| | Implement contain function in lock-free skip list | Done | Chen He |
| | Finish milestone report | Done | Together |
| Nov 23 - Nov 26 | Finish remaining part of delete function in lock-free skip list | | Chen He |
| | Finish the remaining part of the delete function in the fine-grained skip list. | | Yida Wu |
| Nov 27 - Nov 28 | Verify the correctness for the implementation and fix problems | | Together |
| Nov 29 - Dec 3 | Design test cases to evaluate performance | | Together |

| | | |
|---|---|---|
| | Conduct performance evaluation | Together |
| | Improve performance if we still have time | Together |
| Dec 4 - Dec 9 | Finish final project report | Together |
| | Finish poster | Together |

## Summary and Completed Work

We have finished the coarse-grained skip list implementation with a global lock.

For the fine-grained skip list, we have completed the contains(search) and insert operations. The delete operation still has some small problems in concurrent context. The reason why our fine grained skip list is not fully finished yet based on what we planned in our proposal will be detailed explained in the concern section at the end of the report.

For the lock-free skip list, we have completed the contains(search) and insert operations. The delete operation is still in progress because there's some new challenges that were not foreseen. These new challenges are mentioned in the later parts. It is expected to be fixed within several days.

## Goals and Deliverables

We will stick to our original plan and we will finish a functioning coarse-grained skip list, fine-grained skip list, lock-free skip list and an analysis report regarding their performance. We originally plan to try to improve the performance of the lock-free version and try to make it perform better than the locked version. However, we found that some of the features in the lock-free research paper are not available in C++ standard library. Thus, it's hard to tell whether a lock-free or locked version will have better performance. We will try our best on each version to improve its individual performance and we will discuss our strategy in our final report.

### 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.

### Hope to Achieve

● Optimized fine-grained skip list using strategies from different research papers.

● Optimized lock-free skip list using strategies from different research papers.

### Something further

● We seek to think creatively. So, if time allows, we would like to go beyond the research paper that already exists. We would like to optimize our skip list with batch operations if possible.

### 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.

## Preliminary Results

At this time, we have no preliminary results to demonstrate regarding the performance of our implementations. We have developed tests that focus on functionality on single/multiple threads. Once we finish our implementation and prove its correctness under multi-thread conditions, we will develop performance test cases.

## Challenges and Concerns

The reason why our delete function in the fine grained skip list is not finished yet is quite related to our fine grained skip list implementation details: our fine grained skip list adds locks on every level of each node in the list plus a big local lock on each node. Assume we have a six level skip list, the number of locks a single node could hold is 7, which means on delete operation on a single node, we need to simultaneously manage possible 14 locks at the same time. This greatly adds complexity and difficulty to our lock management, and now we still have some small problems with locks in delete operations.

The research paper we follow uses some atomic features (e.g. markable reference) that are not available in the C++ atomic standard library on the ghc machine. We already found possible alternative solutions to solve this problem and are still working on the designing and coding. This may potentially lower the performance. We also have to modify some of the program structures and design our own procedures due to the lack of features. This brings additional challenges in the designing and implementation.

Another problem is that many research papers in this area do not mention memory deallocation. This is not a problem when the program has a garbage collector. However, we are implementing the skip list in C++ so we have to figure out our own ways. We initially try to free the node when it is physically deleted from the skip list. However, we found that some threads may still have access to the deleted node, which will lead to segment fault. We can only free the node's memory when there's no place that can access it. We are planning to use a share pointer to solve this problem but worrying about whether it will further degrade the performance.

One of the strengths of skip list outweighs other data structures like red black tree is batch operations, since skip list does not need to acquire global information like root in red black tree in its design. Thus, if time allows, we would dive deeper into batch operations of the skip list which add a bunch of nodes single time instead of performing continuous insert operations.