**The Effects of Rotating Skiplist on LevelDB**

Jesse Galloway and Sam Paleologopoulos

Our Github Repository containing source code, tests, LevelDB builds: <https://github.com/jsngalloway/leveldb_rotating_skiplist>

**Topic**

The aim of this project is focused around LevelDB (<https://github.com/google/leveldb>), an open-source, in-memory database designed and used by Google. Our project is an attempt to replace the underlying skiplist data structure it employs as an index with a rotating skiplist, as outlined in the paper “A Skip List For Multicore”, by Dick, Fekete, and Gramoli.

**Background**

LevelDB is a fast key-value storage that provides an ordered mapping of string keys to string pairs. Basic operations include Put(key, value), Get(key), Delete(key).

Rotating Skiplist is a type of skiplist that claims to be the fastest concurrent skiplist to date. It aims to improve throughput performance by rotating the head of the skiplist for minimizing contention, using wheels instead of towers in the internal structure, and no locks. According to the paper, Rotating skiplist outperforms all other novel logarithmic skiplists/trees (Fraser’s skiplist, Crain’s no hotspot skiplist, Optimistic skiplist, Transaction-based skiplist, Speculation-Friendly BST, Citrus Tree, Non-blocking BST) across 0%, 10%, 30%, and 90% update transactions in terms of throughput.

**Motivation**

The rotating skiplist we aim to implement boasts fast speeds.

**Major Challenges**

The first major challenge we encountered in this project was getting an instance of LevelDB to build. Initially, we worked with a Windows port of LevelDB with limited documentation. After little success with this, and acknowledging that a system built on Linux would have more replicability we switched to a Ubuntu Docker environment. Container specifics can be found in our Github repository. In this development container we were able to successfully built LevelDB and Synchrobench in the same environment. Once running on the same machine we were able to modify the source and compare the performance of the two codebases.

Our first test intended to measure the performance of LevelDB’s existing skiplist against Synchrobench’s implementation of the Rotating skiplist. This proved to be difficult, however, as Synchrobench’s results were not originally comparable to our custom LevelDB tests. The test structure that Synchrobench uses in its test files involves a pre-populated skiplist and then only executes updates and lookups – for a set duration. In contrast the pre-built tests for LevelDB populated the skiplist and performed only lookups in its testing script.

After producing a replicable, maintainable version of LevelDB, we wrote tests measuring LevelDB’s performance in accordance with the tests described in “A Skiplist for Multicore”. We tested monotonically increasing keys/values, across single and multithreaded instances, using different proportions of updates. In order to compare the performance of the Rotating Skiplist and the LevelDB Skiplist, we created custom tests which could operate on both codebases performing lookups and updates for configurable durations. The results of these performance assessments can be seen below.

Because the LevelDB skiplist must be externally synchronized we only performed these performance tests using a single thread. It from the concurrent tests from LevelDB and the rotating skiplist’s documented performance is likely that that Rotating Skiplist will continue to outperform the LevelDB skiplist by an increasing margin with the further addition of threads.

LevelDB build on windows

Understanding leveldb’s structure

* Arena
* Skiplist implementation
* Locking mechanisms

Understanding synchrobench

* Performing tests on data structures
* Isolating rotating skiplist
* Understanding results

Porting Rotating skiplist to leveldb

* File structure
* Locking
* Includes
* Deletes

Testing Results

**Implementation Details**

In an ideal world, this implementation would be different.

**Results**

What we think it would do

Might not have the hugh performance bonus we expected

**References**

I. Dick, A. Fekete, V. Gramoli. A Skip List for Multicore. Concurrency and Computation: Practice and Experience. 2016.

V. Gramoli. More than You Ever Wanted to Know about Synchronization. PPoPP 2015

K. Fraser. Practical lock freedom. PhD thesis, Cambridge University, 2003.

T. Crain, V. Gramoli and M. Raynal. No hot spot non-blocking skip list. In ICDCS, 2013.

M. Herlihy, Y. Lev, V. Luchangco and N. Shavit. A Simple Optimistic Skiplist Algorithm. In SIROCCO, p.124-138, 2007.

T. Crain, V. Gramoli and M. Raynal. A speculation-friendly search tree. In PPoPP, p.161–170, 2012.

M. Arbel and H. Attiya. Concurrent updates with RCU: Search tree as an example. In PODC, 2014.

P. Felber, V. Gramoli and R. Guerraoui. Elastic Transactions. In DISC 2009.