Abhishek Pathak *apathak2*
Rohan Aggarwal *rohanagg*

Project Proposal

15-618
November 1, 2019

Summary

We will implement a non-blocking chromatic tree, which is a relaxed version of a red-black tree. This modification of a red-black tree is based on a generic template which applies to a variety of tree data structures. If time permits, we will extend this template to a non-blocking version of an AVL tree.

Background

We are implementing a parallel, non-blocking version of a self-balancing tree using shared-memory parallelism. Self-balancing trees are, in general, important data structures used for efficient lookup. Since many processes may want to modify and query the data in a tree at once, it makes sense to try and parallelize a tree. Various approaches are tried to parallelize self-balancing trees. In all these approaches, what is required is to ensure the structural integrity of the tree when lookup happens. This integrity can be endangered by various processes racing to make their own changes to this shared data structure. Therefore, many approaches use locks to protect certain parts of the tree from being accessed by other threads while they are being modified. However, locks can cause lots of waiting between threads. Therefore, implementing a lock-free or non-blocking parallelized version of a self-balancing tree would be very beneficial.

The challenge

Search trees are difficult to parallelize because it is not a "linear" data structure whose layout can be predicted and used to control synchronization, and the insert, delete and balance operations can interfere with each other when called from different threads - each one can cause large changes in tree structure.

Therefore, we will implement non-blocking parallel self-balancing trees, in which the balance operation is applied subsequently rather than immediately after an insert or delete operation. It has been proven that this subsequent balancing will eventually lead to a balanced tree.

Lock-free data structures are in general difficult to get right and make efficient, so we will have to ensure that we are able to surpass the base performance of the data structure in question.

Resources

We will be using the paper "A General Technique for Non-blocking Trees" by Trevor Brown, Faith Ellen and Eric Ruppert as a reference, which describes a template to implement various

kinds of tree data structures in a lock-free manner.

Goals and deliverables

1. Understand the template and how it applies to various kinds of trees. (50%)
2. Implement a non-blocking chromatic tree - a relaxed version of a red-black tree described in the paper. (80%)
3. Benchmark the performance of the non-blocking chromatic tree and observed speedup. (100%)
4. Optionally, extend the template described in the paper to another kind of tree and prove its correctness using the template as described in the paper. (120%)‘

Platform Choice

We will work on Linux OS and code in C++. This is because C++ is better for writing flexible code with complex data structures. Also we will be able to make use of pthreads. We will need a machine with a large number of hardware threads and relatively large amount of memory. For this we may use the GHC machine or latedays, or any other machines determined by consulting the instructors.

Schedule

Nov 9 - Understand the template described in the paper

Nov 14 - Implement a sequential version of the chromatic tree

Nov 23 - Convert the sequential implementation to a parallel implementation

Nov 28 - Do all checks and performance comparisons

Dec 5 - Implement a parallel AVL tree (Optional)