418 Final Project Milestone Report

Current Project Title

A Lock Free B+ Tree Implementation (Chenrui Shao and Cassidy Bolio)

URL

https://cbolio.github.io/15-418-Lock-Free-B-Plus-Tree/

Changes From Original Proposal

Originally, we were planning to implement a parallel version of the CCSD(T) method. However, it was determined that there were too few intermittent interdependencies within calculations, meaning that parallelization might be too trivial. Now, our new project idea is to implement a B+ tree data structure that supports lock free search, insertion, and deletion. We have written a new proposal for this idea and began working on it a week ago. While we are making progress, we still need to catch up after spending the first two weeks on the original CCSD(T) implementation. Additionally, we have been spending more time on the initial sequential implementation than anticipated. To make our eventual parallel implementation easier, we plan to start with two different structures based on distinct research papers. After spending 3-4 days working with each framework, we will evaluate which one has fewer complications and better opportunities for future optimization, using that one as our final version to fully implement and benchmark.

Revised Project Schedule

4/16 - 4/19:
- Design project schedule (Chenrui)
- Write milestone report (Cassidy)
- Update website (Cassidy)
- Meet with Professor Railing (Both)
- Implement serial code (Chenrui)
- Write code for conducting unit testing on both serial and parallel code (Chenrui)
- Begin parallel implementation using 2012 paper (Cassidy)

4/19 - 4/22:
- Test and debug the serial code (Chenrui)
- Finish initial parallel code for 2012 paper (Cassidy)

4/22 - 4/26:
- Attempt implementation according to the PALM paper (Chenrui)
- Test and debug parallel implementation of 2012 paper (Cassidy)

4/26 - 4/29:

- Compile challenges and highlights of each implementation (Both)
- Merge the two versions of implementation based on above factors (Both)

4/29 - 5/3:

- Unit testing and debugging of the parallel implementation (Chenrui)
- Gather performance data for final report (Cassidy)
- Compare performance of the parallel implementation with implementation based on locks and serial version we implemented (Cassidy)

5/3 - 5/6:

- Write half of final report (Chenrui)
- Write half of final report (Cassidy)
- Prepare demo and poster for presentation (Both)
- Present at poster session (Both)

Work Completed

At this point in time, we have written our revised proposal and had it approved. We have also updated our project website to reflect the change in project idea. We have found two research papers that describe different potential implementations of lock free B+ trees, which we will use as a basis for our implementation. For the past week, we have been focusing on implementing a serial version that can be used as the framework for our parallel version. We have also been consolidating unit tests from a variety of Github repositories to use with our future unit testing code. Lastly, we have created a new, updated schedule to allow us to finish on time despite our late start, and we have written the project milestone report to both submit to the class and add a new page to our website with the updated milestone check-in.

Goals and Deliverables

We are almost finished with our sequential implementation and will begin debugging and unit testing this weekend. We believe we will be able to complete our core deliverable, a correct lock free implementation, by the end of the month. We may or may not have time to optimize it further using techniques described in additional papers depending on how complicated implementation and debugging end up being. Originally, we were considering implementing our own lock-based version to use for performance comparison. However, with the challenges we are already seeing with the sequential version, this is not likely to be accomplished. Instead, we will use an open-source lock-based implementation for comparison. Additionally, our original goals included creating an interactive visual representation of the performance benchmarking for the poster session. Due to time constraints, we will instead be using graphs and charts to demonstrate differences in performance and speedup based on different actions (insert, delete, search) under various levels of contention and data size.

Our updated goals include:

- Implement, debug, and unit test a serial version of a B+ tree
- Implement, debug, and unit test a parallel lock free version of a B+ tree
- Evaluate and compare the performance of the lock free version to our serial version and an open-source lock-based version under different conditions
- Consolidate and analyze performance data
- Prepare graphs and charts for the final report and poster session

Poster Session

We plan to show a variety of graphs comparing the performance of our lock free implementation to our serial implementation and an open-source lock-based implementation. We will have tables containing basic information about the runtime of fundamental functions (search, insert, delete) for each implementation. The graphs will be used to compare the runtime and speedup of a series of operations on each of the different versions. We will also outline the results of using performance measuring tools such as perf to hypothesize why any unanticipated results occur, specifically if our lock free version does not outperform the lock-based version under high contention.

Primary Concerns

Our major concerns at this time are debugging and time itself. The sequential version is presenting some challenges, but we hope to alleviate this issue for the parallel versions by using more invariants and intermittent testing during implementation. We also hope that between the two possible parallel implementations, one will appear to be clearer to finalize and optimize. The biggest issue for us is time. We are starting the parallel implementations very late, which does not give us a lot of buffer room if neither implementation is going well. We are attempting to mitigate this possibility by meeting bi-weekly to evaluate our progress and make modified plans if needed; this Friday, we will ensure our sequential version is solid and that parallel version one has no apparent roadblocks. Similarly, we are meeting the following week to ensure that both parallel implementations are successful on a small scale, even if they do not yet support all operations. While starting with two versions may seem like a time risk, we are hoping to avoid a situation where we find ourselves completely stuck and needing to start over with less than three weeks until the poster session. Having small versions of each possible structure will give us a better idea of which one we can implement more quickly. If we do not see a discernible difference between the two, we will just pick one to proceed with for the remaining two weeks.