Paper Overview 3

Ivo Melse, s1088677

May 12, 2025

Summary

The two common ways to implement concurrent data structures are *lock-based* or *lock-free*. The first approach suffers from a low throughput, while the second approach is notoriously hard to get correct. In this paper, the authors look into a less common approach called *batch parallelism*. They implement an OCaml library that streamlines the process of implementing concurrency through batch parallelism.

Evidence

The researchers claim that their approach can be ported to other programming languages besides OCaml easily. They provide evidence for this in the form of a Rust implementation.

The researchers also make some claims about the performance of `OBatcher`, as compared to sequential, and course-grained parallel implementations. To substantiate this, they provide benchmarking results.

Strengths

As aforementioned, the batch parallel library `OBatcher` significantly outperformed the course-grained parallel implementations. This indicates that this approach has some potential for industrial applications. Furthermore, the approach that the researchers presented appears to have good potential to be extended and improved upon in the future.

Weaknesses

One point of criticism is that the benchmarks were only executed on one machine. Practice shows that performance increases/decreases of parallelized programs vary widely across different machines.

Evaluation

I would recommend acceptance in a conference or journal.

Quality of writing

The writing is of very high quality. The structure is clear and straight-to-the point. This makes comprehension fairly easy.

Questions

- See weaknesses. Do you think that the authors should have ran the benchmarks on multiple machines?
- Do you think that this batch parallelism approach will have applications in industry?