PROJECT PROPOSAL CHECKPOINT

Nishal Ancelette Pereira <napereir@andrew.cmu.edu>
Supradeep Rangarajan <strangar@andrew.cmu.edu>

Comparison of Multithreading between C/C++ and Rust (OpenMP vs Rayon/Crossbeam)

Introduction:

In this project we would be comparing multithreading performance in Rust to that of C/C++ against various benchmarks. We would be comparing multithreading performance by utilizing possibly various multithreading libraries such as Rayon [1], and Crossbeam [1], which are implemented specifically for Rust to that of OpenMP implementation for C/C++. The objective of the project is to get an in-depth understanding of these Multithreading abstractions and explain why one would perform better than the other.

Motivation:

Over the course of 15-618, we have learned that we as developers, need to worry about data races, and we can get higher performance if we understand such nuances. What Rust believes to provide is the abstraction to that problem with zero-cost [0], thus becoming highly popular among industries and developers (Mozilla being the strongest influence). We want to get a deeper understanding of how Rust solves this issue and what speed it provides against C/C++, as it promises to solve many of the issues a programmer faces when parallelizing C/C++ code at zero-cost.

Existing Literature:

Our initial literature survey helped us understand how different programming languages support parallel and concurrent programming [2]. Concurrency is provided with the help of various primitives such as locks, semaphores, conditional variables, channels and buffers with conditional waiting etc. Rust communicates amongst threads using channel that have buffers and conditional waiting. Furthermore, Rust performs compile time analysis on values of threads to determine potential problems. There is a blog which shows how Rust can perform better [3] and there are benchmarks that support it [4]. Rayon is the most popular package used in Rust which provides parallelism with simple abstraction. Similarly, Crossbeam provides stronger support for different styles of parallelism.

Project Goals:

Right now, we have identified only a few goals of the project, but are in further trying to see what is achievable in the course of the month. Some but not limited goals:

1. Comparison of simple multithreading performance for mandelbrot generation, reduction add, grid solver and etc. (we would like some suggestions here)
2. Comparison with more benchmarks using Rayon/Crossbeam on Rust and OpenMP on C/C++.
3. Profile overheads and describe the behavior of threads.

Major Obstacles:

1. Understanding the multi-threading paradigm of Rust: Rust, as mentioned above, inherently addresses two specific problems, Memory Safety and Safe Concurrency while having zero-abstraction cost. To explain any difference, we need to look into the assembly generated and also go through the documentation thoroughly to see the changes under the hood, which are abstracted away in Rust. Rust also provides unsafe parallelism; we should explore that too.

2. The learning curve: Though syntactically similar to C++, there are a lot of differences in the way Rust needs to be coded. The learning curve is steep for us to write code in Rust.

3. Crossbeam vs Rayon vs ThreadPool…: There are multiple Crates (libraries in Rust world) that provide parallelism. Crossbeam and Rayon being the best ones as suggested by the community. They both serve similar purposes but provide a different abstraction cost. We need to understand or even compare which libraries to use for which purpose. Rayon, with its Cilk like behavior, has a really good documentation [1]. Crossbeam on the other hand provides more tools with AtomicCell and lock-free data structures, which can give higher performance [1]. We need to choose right, and profile right.

4. Profiling Rust: Rust produces an executable from LLVM. Using Oprofiler and valgrind we can get low level information, but to get a function level timing details, we need to experiment with other tools or possibly write our own code using "std::Instant" to profile [0].

Bibliography

[0] Rust Documentation - https://doc.rust-lang.org/
[1] Official Rayon Documentation - https://docs.rs/rayon/1.0.3/rayon/; Official Crossbeam Documentation
[2] Blog post Comparing Concurrency in Various Languages
[3] http://dtrace.org/blogs/bmc/2018/09/28/the-relative-performance-of-c-and-rust/
[4] The Computer Language Game - https://benchmarksgame-team.pages.debian.net/benchmarksgame/faster/rust-gpp.html