ECE459: Programming for Performance Winter 2021

Lecture 16 — Rayon

Jeff Zarnett *2020-10-14*

Data Parallelism with Rayon

Looking back at the nbody-bins-parallel code that we discussed earlier, you may have noticed that it contains some includes of a library called Rayon. It's a data parallelism library that's intended to make your sequential computation into a parallel one. In an ideal world, perhaps you've designed your application from the ground up to be easily parallelizable, or use multiple threads from the beginning. That might not be the situation you encounter in practice; you may instead be faced with a program that starts out as serial and you want to parallelize some sections that are slow (or lend themselves well to being done in parallel, at least) without a full or major rewrite.

That's what I wanted to do with the nbody problem. I was able to identify the critical loop (it is, unsurprisingly, in `calculate_forces`). We have a vector of points, and if there are N points we can calculate the force on each one independently.

My initial approach looked at spawning threads and moving stuff into the thread. This eventually ran up against the problem of trying to borrow the `accelerations` vector as mutable more than once. I have all these points in a collection and I'm never operating on one of them from more than one thread, but a naive analysis of the borrowing semantics is that the vector is going to more than one thread. This is a super common operation, and I know the operation I want to do is correct and won't have race conditions because each element in the vector is being modified only by the one thread. I eventually learned that you can split slices but it was going to be a slightly painful process. Further research eventually told me to stop reinventing the wheel and use a library for this. Thus, Rayon.

As an aside as to why this – in previous courses, such as a concurrency course, there was a lot of expectation to do most things the hard way: write your own implementation and don't use libraries. In this course, such restrictions don't apply. In industry, you'll use libraries that have appropriate functionality, assuming the license for them is acceptable to your project. Rayon is, for the record, Apache licensed, so it should pose no issue. In the previous version of the course where we used C and C++, we taught the OpenMP functionality, which is used to direct the compiler to parallelize things in a pretty concise way.

Back on track. The line in question where we apply the Rayon library is:

```
accelerations.par_iter_mut().enumerate().for_each(|(i, current_accel)| {
```

A lot happens in this one line, so we need to take a look at it.

References