The Rust Programming Language Blog

Fearless Concurrency in Firefox Quantum

Nov 14, 2017 • Manish Goregaokar

These days, Rust is used for all kinds of things. But its founding application was Servo, an experimental browser engine.

Now, after years of effort, a major part of Servo is shipping in production: Mozilla is releasing Firefox Quantum!

Rust code began shipping in Firefox last year, starting with relatively small pilot projects like an MP4 metadata parser to replace some uses of libstagefright. These components performed well and caused effectively no crashes, but browser development had yet to see large benefits from the full power Rust could offer. This changes today.

Stylo: a parallel CSS engine

Firefox Quantum includes Stylo, a pure-Rust CSS engine that makes full use of Rust's "Fearless Concurrency" to speed up page styling. It's the first major component of Servo to be integrated with Firefox, and is a major milestone for Servo, Firefox, and Rust. It replaces approximately 160,000 lines of C++ with 85,000 lines of Rust.

When a browser is loading a web page, it looks at the CSS and parses the rules. It then determines which rules apply to which elements and their precedence, and "cascades" these down the DOM tree, computing the final style for each element. Styling is a top-down process: you need to know the style of a parent to calculate the styles of its children, but the styles of its children can be calculated independently thereafter.

This top-down structure is ripe for parallelism; however, since styling is a complex process, it's hard to get right. Mozilla made two previous attempts to parallelize its style system in C++, and both of them failed. But Rust's fearless concurrency has made parallelism practical! We use rayon —one of the hundreds of crates Servo

uses from Rust's ecosystem — to drive a work-stealing cascade algorithm. You can read more about that in Lin Clark's post. Parallelism leads to a lot of performance improvements, including a 30% page load speedup for Amazon's homepage.

Fearless concurrency

An example of Rust preventing thread safety bugs is how style information is shared in Stylo. Computed styles are grouped into "style structs" of related properties, e.g. there's one for all the font properties, one for all the background properties, and so on. Now, most of these are shared; for example, the font of a child element is usually the same as its parent, and often sibling elements share styles even if they don't have the same style as the parent. Stylo uses Rust's atomically reference counted `Arc<T>` to share style structs between elements. `Arc<T>` makes its contents immutable, so it's thread safe — you can't accidentally modify a style struct when there's a chance it is being used by other elements.

We supplement this immutable access with `Arc::make_mut()`; for example, this line calls `.mutate_font()` (a thin wrapper around `Arc::make_mut()` for the font style struct) to set the font size. If the given element is the only element that has a reference to this specific font struct, it will just mutate it in place. But if it is not, `make_mut()` will copy the entire style struct into a new, unique reference, which will then be mutated in place and eventually stored on the element.

```
context.builder.mutate_font().set_font_size(computed);
```

On the other hand, Rust guarantees that it is impossible to mutate the style of the *parent* element, because it is kept behind an immutable reference. Rayon's scoped threading functionality makes sure that there is no way for that struct to even obtain/store a mutable reference if it wanted to. The parent style is something which one thread was allowed to write to to create (when the parent element was being processed), after which everyone is only allowed to read from it. You'll notice that the reference is a zero-overhead "borrowed pointer", *not* a reference counted pointer, because Rust and Rayon let you share data across threads without needing reference counting when they can guarantee that the data will be alive at least as long as the thread.

Personally, my "aha, I now fully understand the power of Rust" moment was when thread safety issues cropped up on the C++ side. Browsers are complex beings, and despite Stylo being Rust code, it needs to call back into Firefox's C++ code a lot. Firefox has a single "main thread" per process, and while it does use other

threads they are relatively limited in what they do. Stylo, being quite parallel, occasionally calls into C++ code off the main thread. That was usually fine, but would regularly surface thread safety bugs in the C++ code when there was a cache or global mutable state involved, things which basically never were a problem on the Rust side.

These bugs were not easy to notice, and were often very tricky to debug. And that was with only the *occasional* call into C++ code off the main thread; It feels like if we had tried this project in pure C++ we'd be dealing with this far too much to be able to get anything useful done. And indeed, bugs like these have thwarted multiple attempts to parallelize styling in the past, both in Firefox and other browsers.

Rust's productivity

Firefox developers had a great time learning and using Rust. People really enjoyed being able to aggressively write code without having to worry about safety, and many mentioned that Rust's ownership model was close to how they implicitly reason about memory within Firefox's large C++ codebase. It was refreshing to have fuzzers catch mostly explicit *panics* in Rust code, which are much easier to debug and fix than segfaults and other memory safety issues on the C++ side.

A conversation amongst Firefox developers that stuck with me — one that was included in Josh Matthews' talk at Rust Belt Rust — was

> *<heycam> one of the best parts about stylo has been how much easier it has been to implement these style system optimizations that we need, because Rust*
>
> *<heycam> can you imagine if we needed to implement this all in C++ in the timeframe we have*
>
> *<heycam> yeah srsly*
>
> *<bholley> heycam: it's so rare that we get fuzz bugs in rust code*
>
> *<bholley> heycam: considering all the complex stuff we're doing*
>
> **heycam remembers getting a bunch of fuzzer bugs from all kinds of style system stuff in gecko*
>
> *<bholley> heycam: think about how much time we could save if each one of those*

> *annoying compiler errors today was swapped for a fuzz bug tomorrow :-)*
>
> *<heycam> heh*
>
> *<njn> you guys sound like an ad for Rust*

Wrapping up

Overall, Firefox Quantum benefits significantly from Stylo, and thus from Rust. Not only does it speed up page load, but it also speeds up interaction times since styling information can be recalculated much faster, making the entire experience smoother.

But Stylo is only the beginning. There are two major Rust integrations getting close to the end of the pipeline. One is integrating Webrender into Firefox; Webrender heavily uses the GPU to speed up rendering. Another is Pathfinder, a project that offloads font rendering to the GPU. And beyond those, there remains Servo's parallel layout and DOM work, which are continuing to grow and improve. Firefox has a very bright future ahead.

As a Rust team member, I'm really happy to see Rust being successfully used in production to such great effect! As a Servo and Stylo developer, I'm grateful to the tools Rust gave us to be able to pull this off, and I'm happy to see a large component of Servo finally make its way to users!

Experience the benefits of Rust yourself — try out Firefox Quantum!

More about Rust — Jump straight in