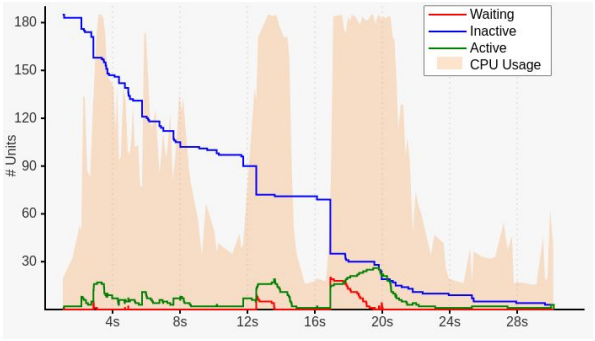
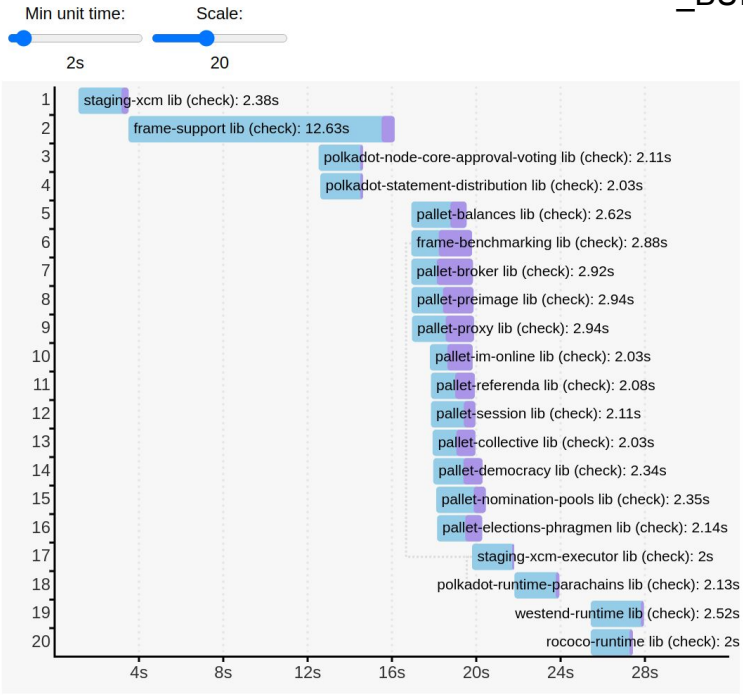
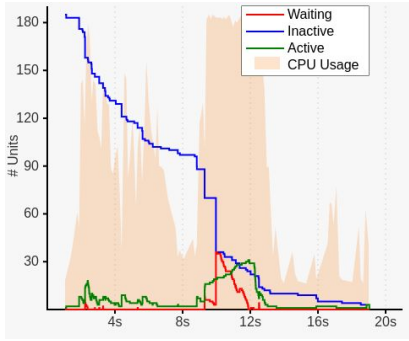


Original Rust compiler

30 sec

for an incremental build (SKIP_WASM _BUILD=1)

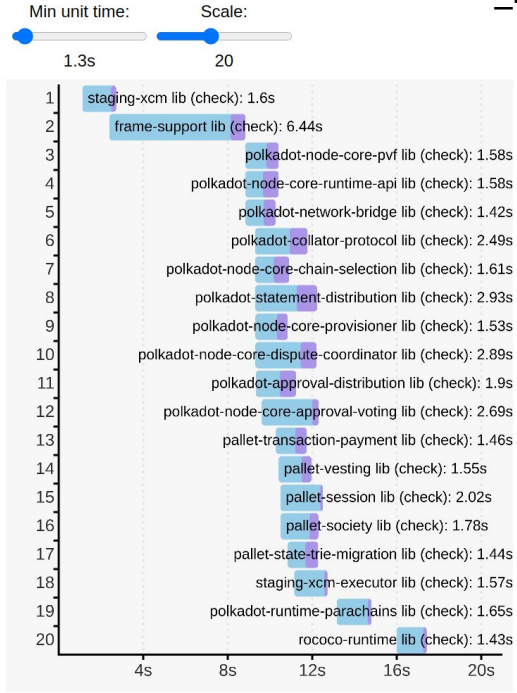




Modded Rust compiler

19 sec

for an incremental build (SKIP_WASM _BUILD=1)



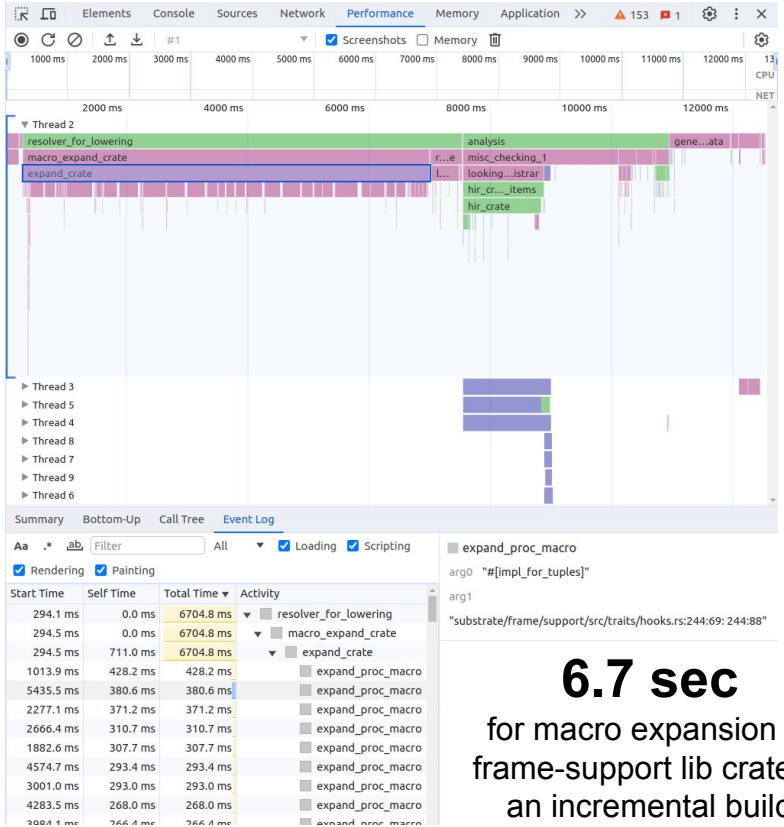
Our compiler is **50% faster than the default compiler** for incremental builds. Note: it is *not* faster for clean builds.

How? We implemented macro expansion caching.

The next page is a case study examining the slowest crate (frame-support) using the Rust self-profiler

Original Rust compiler:

Proc macros are always re-expanded



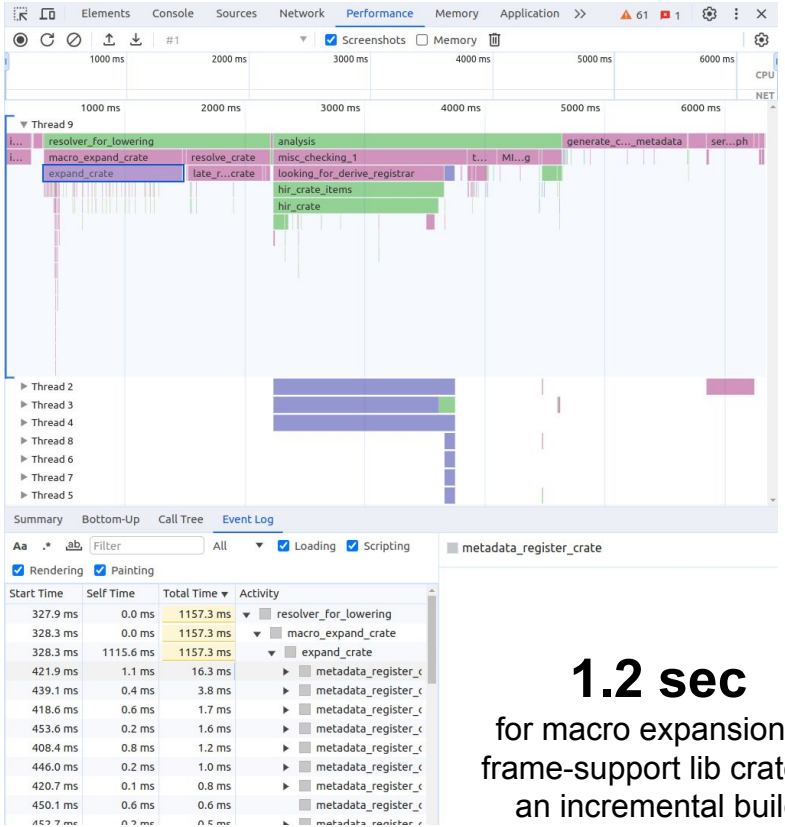
6.7 sec

for macro expansion of frame-support lib crate in an incremental build

Your old compiler: On **any** code change, **all** of the ~250 macros in this crate are expanded (and 1000s of macros across crates)

Modded Rust compiler:

Cache proc macro expansion



1.2 sec

for macro expansion of frame-support lib crate in an incremental build

Our modded compiler: Here we cache all the macro expansions! This effectively solves IDE lags when expanding macros