Fixing build times with rubicon

A somewhat reasonable use of dynamic linking

fasterthanlime for Self-Directed Research

building large rust projects is **slow**.

why?

lots to **parse**
lots to **typecheck**
lots to **borrowcheck**
lots to **codegen**
lots to **optimize**
lots to **link**

incremental builds
are **not enough**

why?

not incremental enough
proc macros aren't cached
static linking takes time (hundreds of MBs)
LTO takes time
there's simply a **lot of work**

note: other build systems do it better (bazel/buck/etc.)

crate-type = "dylib"
does not help much

why?

still "one big graph" (1graph)
lots of work for no-op builds

(cool on-disk hashtable though)

"**prefer**-dynamic"
compiler picks boundaries

(docs were wrong for 4 years)

crate-type = ["rlib", "dylib"] dependencies need to opt-in

(all of them)

monomorphization =
change **app**, rebuild **libtokio**

(libtokio contains all instantiations of its generics. TODO: fact check.)

the fix?
compose smaller projects

pick your own boundaries

trivial for CLI binaries
trivial for HTTP servers
trivial for GRPC
trivial for IPC (SHM etc.)
real tricky for **dlopen**

why tricky?

no stable ABI
globals duplication (that one's hard)

A starts a tokio runtime
calls into B
B says "there's no runtime"
they're both right

it gets trickier

A installs **tracing subscriber**
B, C, D's log events go nowhere

A installs **panic handler**
B, C, D panics don't call it

A passes **tokio runtime handle**
B, C, D hang forever!

parking_lot has globals =
the wrong thread gets awakened

8 weeks of debugging

mostly memory corruption + hangs

(...last 2 were a tokio bug)

rubicon

= import/export globals

(across shared objects: **libfoo.so**, **libbar.so** etc.)

globals are

- thread-locals
- process-locals (*statics*)

how does it work?
source-level patching

(all deps need to play nice, too!)

patches ready for:

- parking_lot
- tokio
- tracing
- eyre

(just wrapping statics / thread-locals with macros)

README goes into specific set-up

(app + librubicon_exports.so + mods)

I'm having fun working on my site again

I'm able to ship lots of small changes

Untouched modules keep their docker layer = **fast deploys**

future steps?

rubicon is a "polyfill"

let's kill it

e.g. `-C globals-linkage=[import,export]`