Los Alamos NATIONAL LABORATORY

MarFS is Getting Rusty

Benjamin Schlueter

Mentors: Dave Bonnie, Garrett Ransom

08/07/25

ROSY ID: 00fbd36a
LA-UR-25-28204

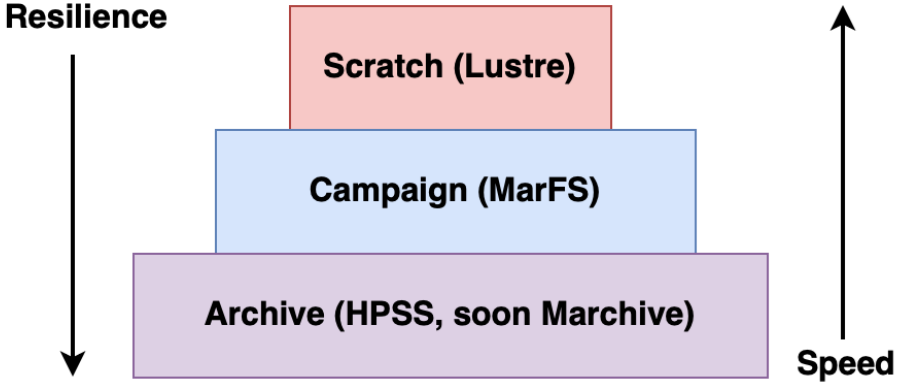
```
Segmentation fault (core dumped)
```

Rust vs C

- Rust enforces strict rules at compile time to catch bugs
- Nearly as fast as C: No garbage collection

MarFS

- LANL's homegrown Campaign storage system
- Serves as a middle tier between scratch and the archive
- Resilience is highest priority
- Only realistic to rewrite modules over time
  - need proof of concept

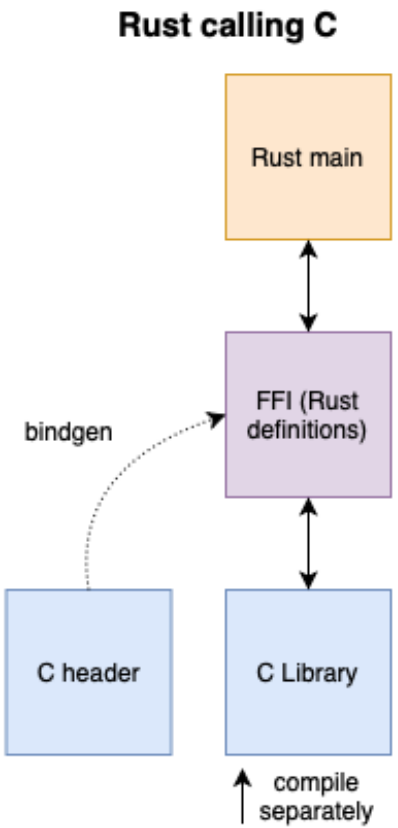


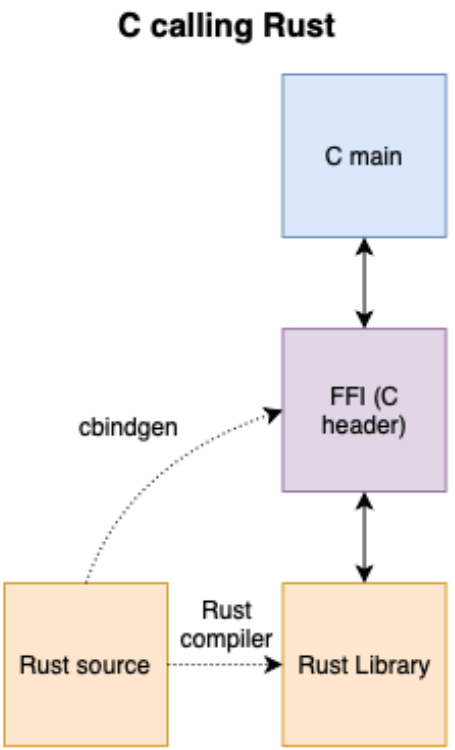
Integrating Foreign Code with Rust

- Rust has a Foreign Function Interface feature
- Two integration steps:
  - Generate FFI bindings (definitions)
  - Link foreign library (implementations)

Proof of Concept

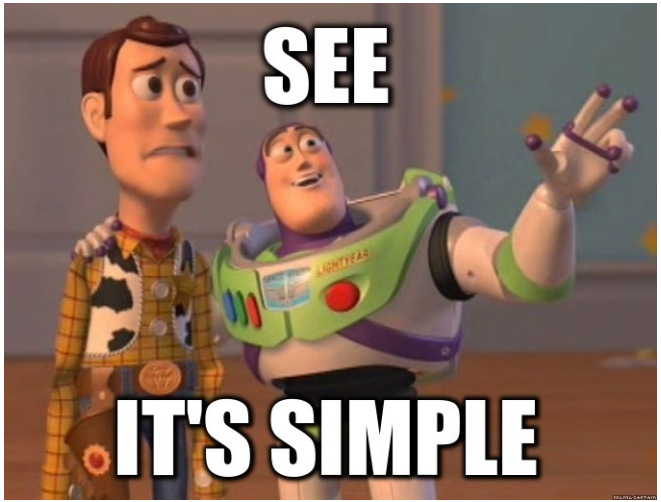
- Proof of concepts:
  - streamutil (Rust calling C)
  - Data Abstraction Layer (Rust called by C)
- Two integration steps:
  - definitions: bindgen
  - implementations: compiler vars
- Automate and never think about it again!





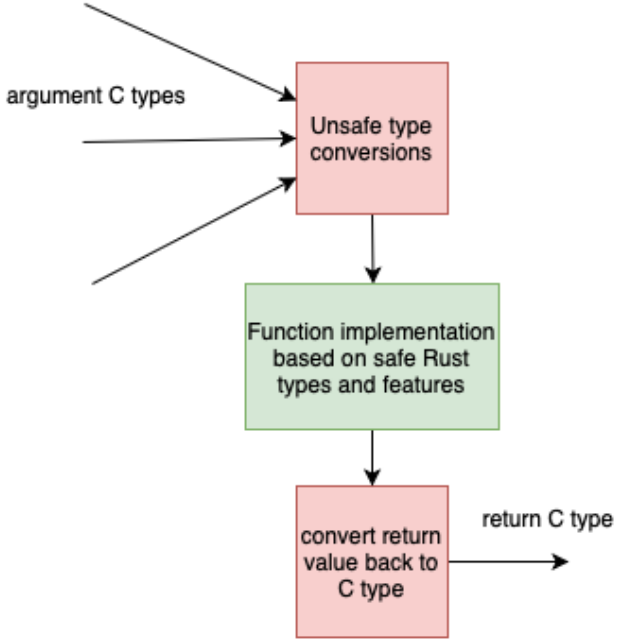
Calling C From Rust

- Easy: Just use FFI Rust definition
- Sometimes have to do gross unsafe conversions to FFI C types



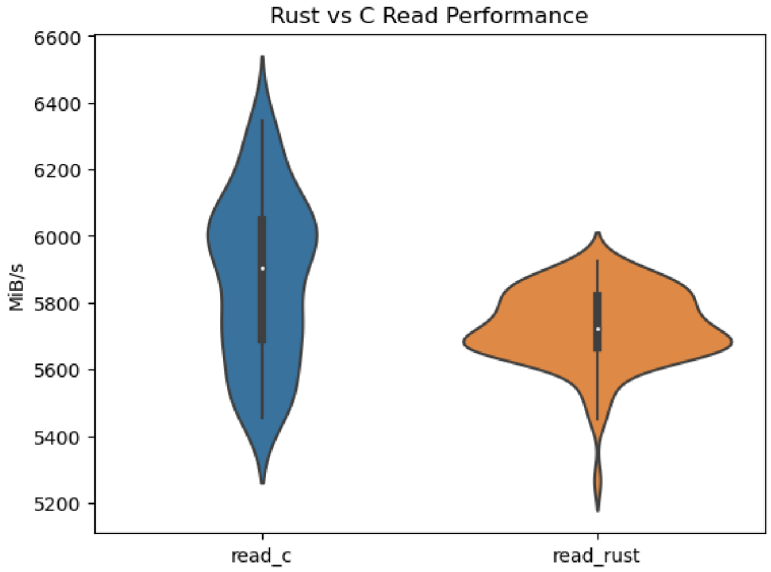
Calling Rust from C

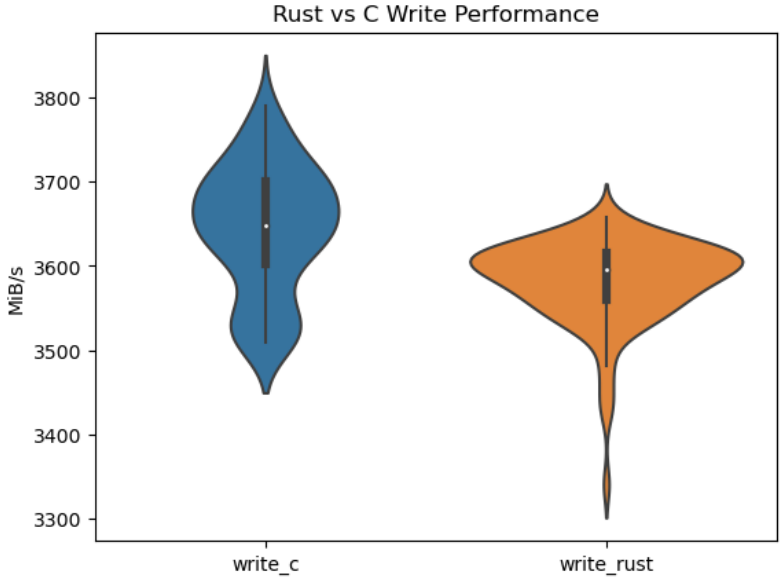
- Only C compatible components can be in the FFI
- Any Rust features can be used in function implementations
- Cannot use high level Rust features like Traits
- Summary: think of C when designing



Performance

- No significant performance difference between Rust and C I/O (on VM)
- Used nix: thin safe syscall wrapper





Conclusion

Integrating Rust and C is performant and easy once you learn the tricks

I encourage anyone looking to add safety and niceties to try Rust


WOOHOO!!
MY PRESENTATION IS DONE!
makeameme.org