
throwing it all away
how extreme rewriting changed the way I build databases
Tyler Neely HYTRADBOI 2025

who am I

- Tyler Neely
- distributed databases since 2012
- building sled, lots of db-related projects

sled Public
the champagne of beta embedded databases
Rust 8.3k 390

rio Public
pure rust io_uring library, built on libc, thread & async friendly, misuse resistant
Rust 943 47

marble Public
garbage-collecting on-disk object store, supporting higher level KV stores and databases.
Rust 374 14

concurrent-map Public
lock-free B+ tree
Rust 280 13

tla-rust Public
writing correct lock-free and distributed stateful systems in Rust, assisted by TLA+
TLA 1.1k 26

rust-rocksdb/rust-rocksdb Public
rust wrapper for rocksdb
Rust 1.9k 750

terrors Public
ergonomic and precise error handling built atop type-level set arithmetic
Rust 215 6

tiny-lsm Public
super simple in-memory blocking LSM for constant-size keys and values
Rust 68 4

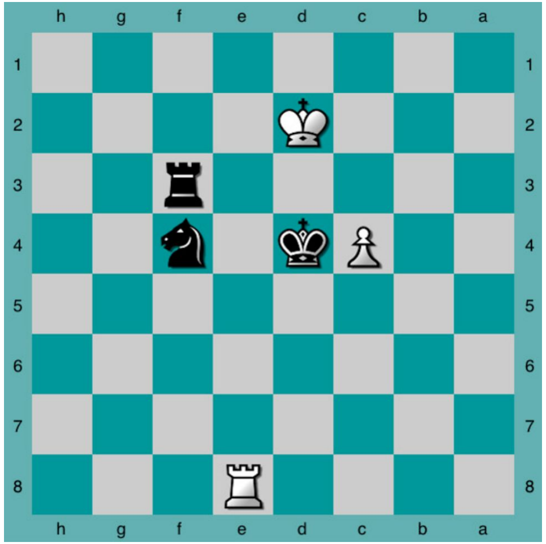
void-rs/void Public
terminal-based personal organizer
Rust 1.1k 38

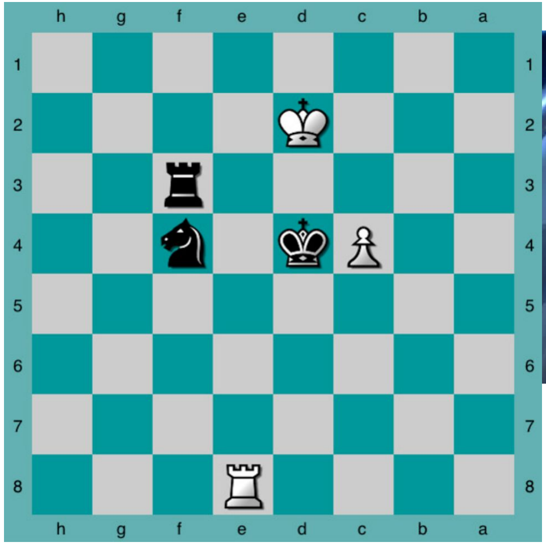
flamegraph-rs/flamegraph Public
Easy flamegraphs for Rust projects and everything else, without Perl or pipes <3
Rust 4.9k 154

cache-advisor Public
scan-resistant concurrent lazy LRU
Rust 58 1

art Public
Adaptive Radix Trie implementation for fixed-length keys
Rust 53 7

rust-crdt/rust-crdt Public
a collection of well-tested, serializable CRDTs for Rust
Rust 1.4k 60


h g f e d c b a
1 2 3 4 5 6 7 8


h g f e d c b a
1 2 3 4 5 6 7 8

sled complexity issues

- building sled since 2016
- modularity decays as concepts evolve
- productivity drops
- I hit a wall

My assumptions were challenged

- I briefly met Joe Armstrong at Erlang Factory 2015
- He told me to throw away anything you can’t finish in a day.
- I didn’t think it was good at the time.
- But it was.

the birth of komora

- I decided to try to rewrite sled in a day
- The next day, I would throw it away and start over
- I would keep the tests and high level interfaces
- Often, I would decide to just focus on a small recurring component
- komora was born

Komora

278 followers   http://komora.io

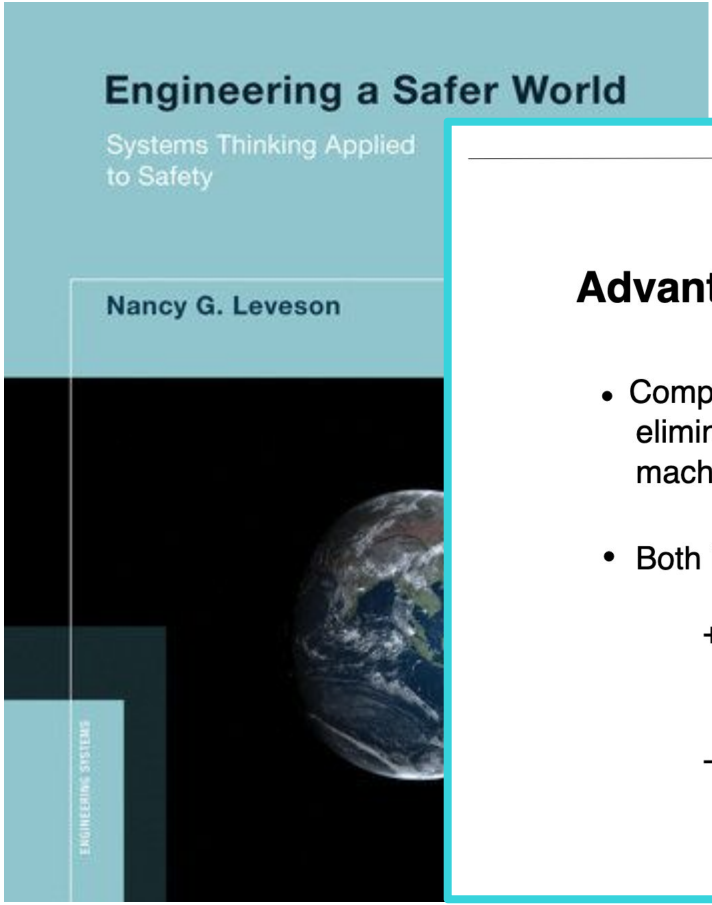
README.md

komora

key	value
wtf are these?	declassified industrial stateful system components
marble	garbage-collecting object store
concurrent-map	lock-free in-memory B+ tree
terrors	precise error handling built atop type-level set arithmetic
sharded-log	low-contention logger
tiny-lsm	KV for fixed-size items
ebr	epoch-based reclamation
pagetable	concurrent (wait-free!) 4-level pagetable
art	adaptive radix trie
fault-injection	io::Error testing triggers and source annotation
shared-local-state	manage dynamic concurrent thread state
inline-array	DB-focused low-space shared byte array container
cache-advisor	scan-resistant non-blocking sharded concurrent LRU
metadata-store	a write and recovery-only store for u64->NonZeroU64+bytes mappings
optimistic-cell	lock-like structure for highly efficient scalable concurrent access
stack-map	constant-size associative structure for composing into high-level structures

Some Connections

- Leveson’s Safety Engineering
- O’Reily’s take on Residuality Theory
- Kuhn’s theory on Scientific Revolutions



Advantages = Disadvantages

- Computer so powerful and so useful because it has eliminated many of physical constraints of previous machines.
- Both its blessing and its curse:
  - \+ No longer have to worry about physical realization of our designs.
  - – No longer have physical laws that limit the complexity of our designs.

Paper: Leveson 2003 - Applying STAMP in Accident Analysis

Systems Theoretic Safety Engineering

- Reliability-based models of root causes, component failures etc… has it upside down
- Instead, focus on control that specifically avoids undesired outcomes
- From this lens:
  - we want to avoid high complexity
  - coding over time causes complexity
  - adding a one-day max limit controls this

Human optimizations are usually the most meaningful optimizations.

Thank you :)