Borrowing FoundationDB's Simulator

for Layer Development 🧪

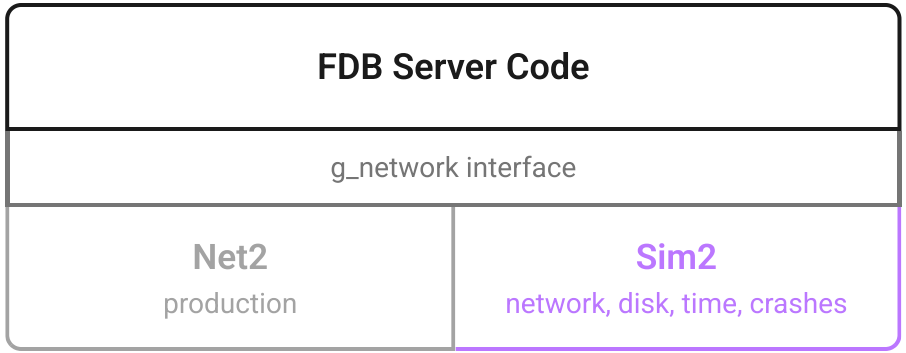
Pierre Zemb — Staff Engineer @ Clever Cloud 🇫🇷

Maintainer of foundationdb-rs

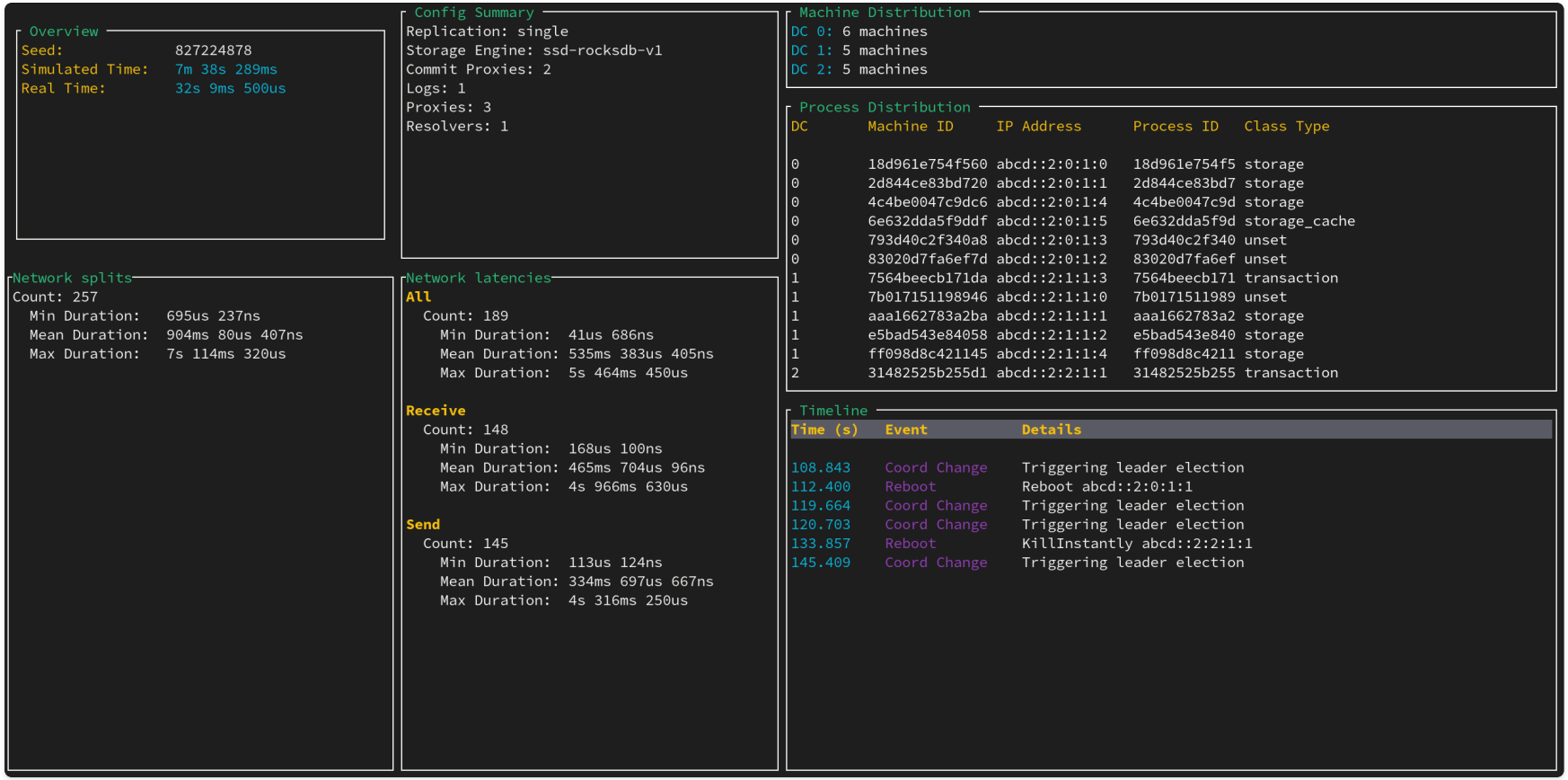
From HBase pain to FDB layers 🏗️

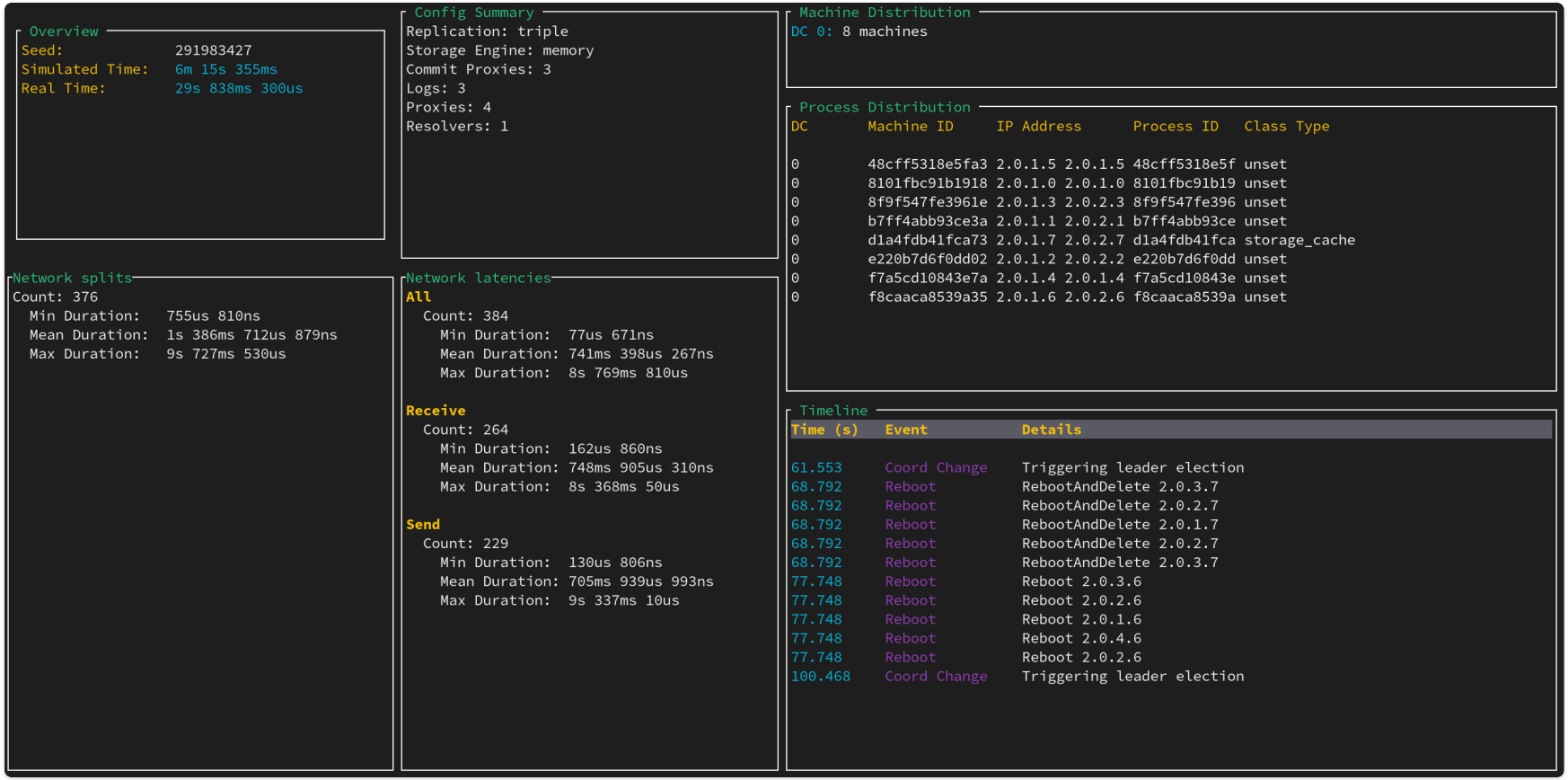
- 👨‍🚀 Software engineer who learned **dist-sys pitfalls the hard way**
- 😱 Years of **Apache HBase trauma** (270 machines, **2.5M w/s**, **6.5M r/s**)
  - Network split during region split → **inconsistent cluster**
  - Repair tool (`hbck`) moved partitioned data out of the keyspace to make it "consistent" 🙃
- 🔍 I discovered **FoundationDB**, a distributed key-value store
  - For **developers**, a toolbox to write distributed systems
  - For **operators**, a highly scalable / production-ready system

FDB's simulation framework 🔬



- 🎭 Every fallible interaction behind **one interface**: network, disk, time, randomness
- 💥 Swap the implementation for **fakes** → inject failures at will
- 🎲 Deterministic → same seed = **same bugs, every time**


Overview
Seed: 827224878
Simulated Time: 7m 38s 289ms
Real Time: 32s 9ms 500us
Config Summary
Replication: single
Storage Engine: ssd-rocksdb-v1
Commit Proxies: 2
Logs: 1
Proxies: 3
Resolvers: 1
Machine Distribution
DC 0: 6 machines
DC 1: 5 machines
DC 2: 5 machines
Process Distribution
DC Machine ID IP Address Process ID Class Type
0 18d961e754f560 abcd::2:0:1:0 18d961e754f5 storage
0 2d844ce83bd720 abcd::2:0:1:1 2d844ce83bd7 storage
0 4c4be0047c9dc6 abcd::2:0:1:4 4c4be0047c9d storage
0 6e632dda5f9ddf abcd::2:0:1:5 6e632dda5f9d storage_cache
0 793d40c2f340a8 abcd::2:0:1:3 793d40c2f340 unset
0 83020d7fa6ef7d abcd::2:0:1:2 83020d7fa6ef unset
1 7564beecb171da abcd::2:1:1:3 7564beecb171 transaction
1 7b017151198946 abcd::2:1:1:0 7b0171511989 unset
1 aaa1662783a2ba abcd::2:1:1:1 aaa1662783a2 storage
1 e5bad543e84058 abcd::2:1:1:2 e5bad543e840 storage
1 ff098d8c421145 abcd::2:1:1:4 ff098d8c4211 storage
2 31482525b255d1 abcd::2:2:1:1 31482525b255 transaction
Network splits
Count: 257
Min Duration: 695us 237ns
Mean Duration: 904ms 80us 407ns
Max Duration: 7s 114ms 320us
Network latencies
All
Count: 189
Min Duration: 41us 686ns
Mean Duration: 535ms 383us 405ns
Max Duration: 5s 464ms 450us
Receive
Count: 148
Min Duration: 168us 100ns
Mean Duration: 465ms 704us 96ns
Max Duration: 4s 966ms 630us
Send
Count: 145
Min Duration: 113us 124ns
Mean Duration: 334ms 697us 667ns
Max Duration: 4s 316ms 250us
Timeline
Time (s) Event Details
108.843 Coord Change Triggering leader election
112.400 Reboot Reboot abcd::2:0:1:1
119.664 Coord Change Triggering leader election
120.703 Coord Change Triggering leader election
133.857 Reboot KillInstantly abcd::2:2:1:1
145.409 Coord Change Triggering leader election


Overview
Seed: 291983427
Simulated Time: 6m 15s 355ms
Real Time: 29s 838ms 300us
Config Summary
Replication: triple
Storage Engine: memory
Commit Proxies: 3
Logs: 3
Proxies: 4
Resolvers: 1
Machine Distribution
DC 0: 8 machines
Process Distribution
DC Machine ID IP Address Process ID Class Type
0 48cff5318e5fa3 2.0.1.5 2.0.1.5 48cff5318e5f unset
0 8101fbc91b1918 2.0.1.0 2.0.1.0 8101fbc91b19 unset
0 8f9f547fe3961e 2.0.1.3 2.0.2.3 8f9f547fe396 unset
0 b7ff4abb93ce3a 2.0.1.1 2.0.2.1 b7ff4abb93ce unset
0 d1a4fdb41fca73 2.0.1.7 2.0.2.7 d1a4fdb41fca storage_cache
0 e220b7d6f0dd02 2.0.1.2 2.0.2.2 e220b7d6f0dd unset
0 f7a5cd10843e7a 2.0.1.4 2.0.1.4 f7a5cd10843e unset
0 f8caaca8539a35 2.0.1.6 2.0.2.6 f8caaca8539a unset
Network splits
Count: 376
Min Duration: 755us 810ns
Mean Duration: 1s 386ms 712us 879ns
Max Duration: 9s 727ms 530us
Network latencies
All
Count: 384
Min Duration: 77us 671ns
Mean Duration: 741ms 398us 267ns
Max Duration: 8s 769ms 810us
Receive
Count: 264
Min Duration: 162us 860ns
Mean Duration: 748ms 905us 310ns
Max Duration: 8s 368ms 50us
Send
Count: 229
Min Duration: 130us 806ns
Mean Duration: 705ms 939us 993ns
Max Duration: 9s 337ms 10us
Timeline
Time (s) Event Details
61.553 Coord Change Triggering leader election
68.792 Reboot RebootAndDelete 2.0.3.7
68.792 Reboot RebootAndDelete 2.0.2.7
68.792 Reboot RebootAndDelete 2.0.1.7
68.792 Reboot RebootAndDelete 2.0.2.7
68.792 Reboot RebootAndDelete 2.0.3.7
77.748 Reboot Reboot 2.0.3.6
77.748 Reboot Reboot 2.0.2.6
77.748 Reboot Reboot 2.0.1.6
77.748 Reboot Reboot 2.0.4.6
77.748 Reboot Reboot 2.0.2.6
100.468 Coord Change Triggering leader election

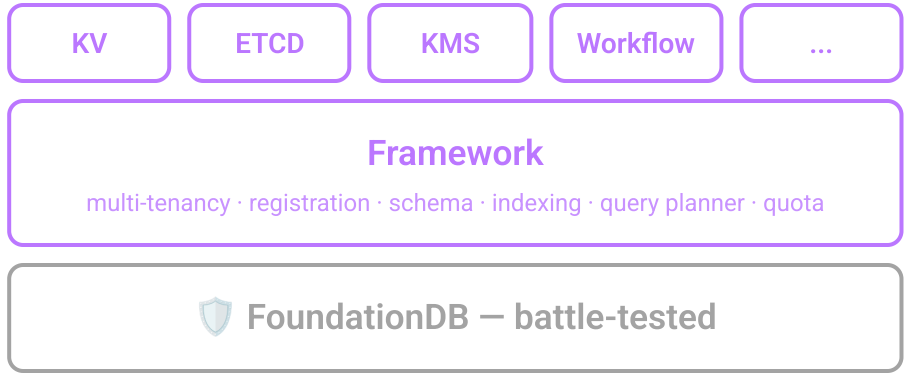
So, we started building 😰

Framework

multi-tenancy · registration · schema · indexing · query planner · quota

🛡️ FoundationDB — battle-tested

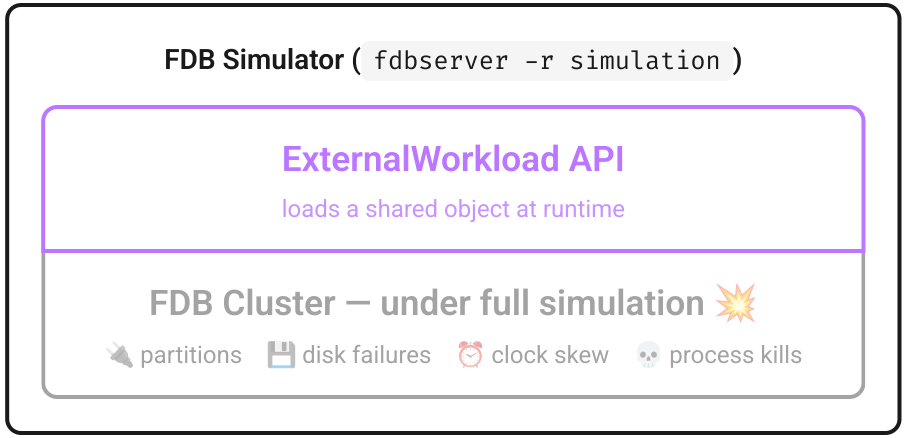
Team grew from 1 to 6 🚀



How do we test this? 🙃

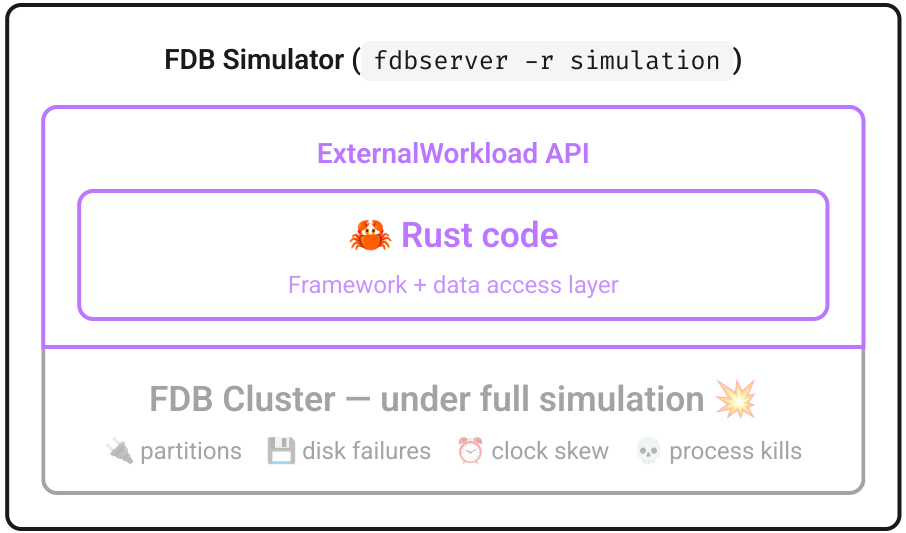
Can we inject our code inside FoundationDB? 🤔

ExternalWorkload API 🔌



Your code talks to FDB normally — but FDB runs with **fakes underneath**

Getting Rust inside the simulator 🦀



At first: boring bugs 🪳

- 🧪 Simple workloads on the framework
  - KV integrity checks, statistics up-to-date…
- 🤷 Essentially just testing FDB's transactional promises
  - The team was not convinced of the benefits, until…

Then workloads got richer 🔥

- 🧪 As workloads got richer, simulation found bugs everywhere:
  - 📊 Query execution returning wrong data
  - 🗺️ Query planning picking the wrong index
  - 💥 Data corruption during reindexing
  - 👑 Dual leader election under clock skew
  - 🗜️ ETCD compaction deleting live data
  - ⏰ Bad TTL handling in the Redis shim
  - …

Simulation finds unknown unknowns 🔮

- 🧠 Once each team member:
  - Found a bug in **their own code**, or
  - Contributed a **low-level piece of engineering** quickly
- 💡 They understood the **power of simulation**
  - Simulation finds what developers **don’t know**
- 🏗️ All our work is now **simulation-first**

Contributing back to FoundationDB 🦀

- 🦀 First language bindings (outside of C++) to ship simulation support
  - open-sourced in foundationdb-rs
- 🔁 Full circle: from happy users to upstream contributors
  - 🛠️ Contributed a pure C API upstream — no more C++ ABI nightmares
  - ⏱️ Added the `delay()` API for time-dependent simulation

Try it on your FDB layers 🎁

crates.io/crates/foundationdb

crates.io/crates/foundationdb-simulation

pierrezemb.fr

Questions? 💬