fdb-rs FoundationDB support for Tokio/Rust

https://fdb-rs.github.io

Rajiv Ranganath

About Me

- Currently working on an application backend for India market using FoundationDB
- Interested in Functional Programming, Databases, Distributed Systems, Operating Systems, Blockchain
- Not active on social media
- Reachable at firstname.lastname@{gmail,atihita}.com, on GitHub and FoundationDB forums

Why FoundationDB?

- Accidental (Arbitrary) Complexity vs. Essential Complexity
- Accidental complexity refers to problems imposed by the particular tools and processes you have chosen
- Essential complexity refers to problems inherent in the domain that you are working on
- With FoundationDB, you still have to deal with complexity but to a large extent it is essential complexity
- There is very little accidental complexity when using FoundationDB

Why FoundationDB?

- **Strict Serializable Transaction Model**
- Managing state

F1: A Distributed SQL Database That Scales VLDB (2013)

The AdWords product ecosystem requires a data store that supports ACID transactions. We store financial data and have hard requirements on data integrity and consistency. We also have a lot of experience with eventual consistency systems at Google.

In all such systems, we find developers spend a significant fraction of their time building **extremely complex and error-prone mechanisms to cope with eventual consistency and handle data that may be out of date**. We think this is an unacceptable burden to place on developers and that consistency problems should be solved at the database level.

Full transactional consistency is one of the most important properties of F1.

Strict Serializability reduces Accidental Complexity

- Begin and commit transaction T1, which writes to item x
- Later you begin and commit transaction T2, which reads from x
- Database providing strict serializability will place T1 before T2 in the serial ordering **and** T2 will read T1's write
- Database will not reorder the transactions, and place T2 before T1
- Strict serializability matches the programmer's intuition of how transactions work

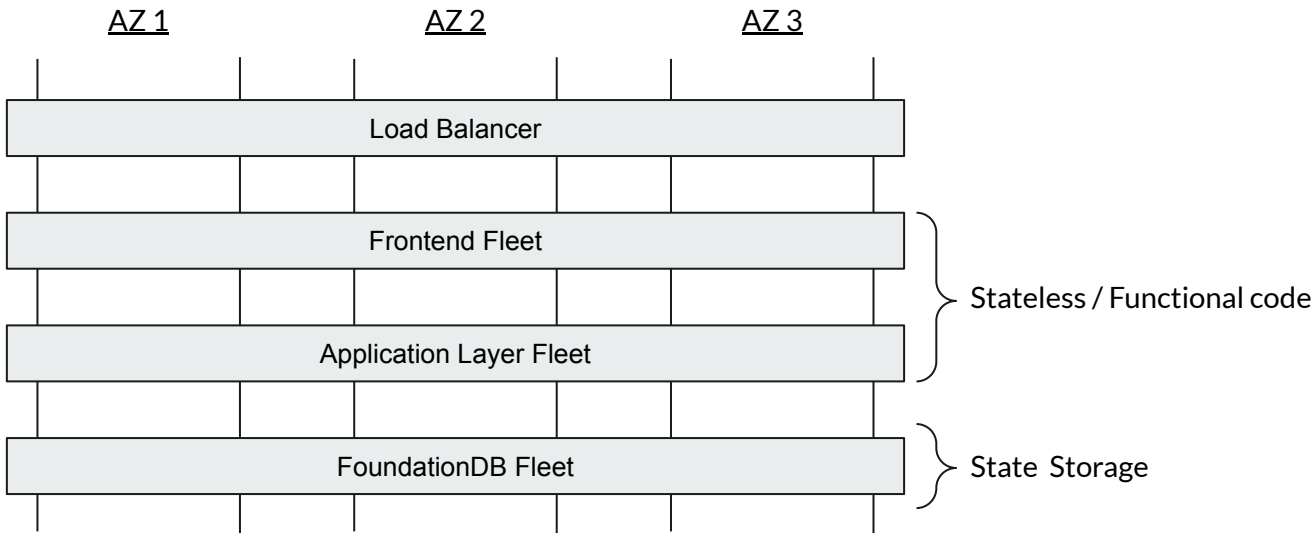
Strictly serializable databases are very rare!

- Proven, horizontally scalable, distributed, fault tolerant, strictly serializable databases are very rare!
- Two well known ones are
  - FoundationDB (open source)
  - Spanner (Google proprietary)
- Additional resources
  - Peter Balis - Linearizability versus Serializability
  - Daniel J. Abadi - Demystifying Database Systems, Part 2: Correctness Anomalies Under Serializable Isolation
  - Jepsen - Strict Serializability

Why FoundationDB?

- Strictly Serializable Transaction Model
- **Managing state**

Managing state

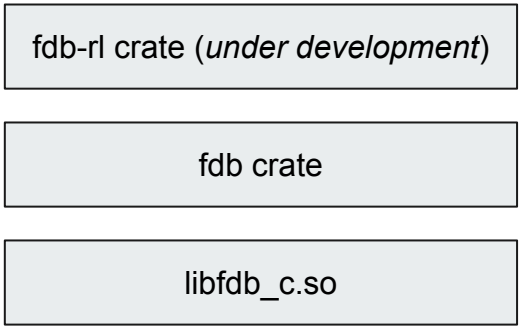


fdb-rs Project

- To develop FoundationDB Rust crates that is designed to work well with Tokio ecosystem crates
- Focused on getting interoperability and ergonomics with the Tokio ecosystem
  - Tokio Async Runtime
  - Bytes
  - Tracing/Metrics
  - Hyper
  - Tower
  - Axum
  - Smithy (AWS)
- foundationdb-rs project maintains runtime agnostic crate, if that is something you prefer!

fdb Crate

- Provides idiomatic Rust/Tokio bindings for communicating with FoundationDB cluster
- APIs are inspired by Java bindings
- Uses unsafe code to communicate with C library



fdb Crate - Types and Traits

- `FdbError`, `FdbResult<T>`
- `Key`, `Value`
- `FdbFuture<T>`, `FdbStreamKeyValue`
- `ReadTransaction`, `Transaction`
- `FdbReadTransaction`, `FdbTransaction`
- `FdbDatabase`

fdb Crate - Transaction retry loop

```
pub async fn run<T, F, Fut>(&self, mut f: F) -> FdbResult<T>
 where
     F: FnMut(FdbTransaction) -> Fut,
     Fut: Future<Output = FdbResult<T>>;

pub async fn read<T, F, Fut>(&self, mut f: F) -> FdbResult<T>
 where
     F: FnMut(FdbReadTransaction) -> Fut,
     Fut: Future<Output = FdbResult<T>>,
```

fdb Crate - Transaction retry loop

```
fdb_database
    .run(|tr| async move {
        // Retryable idempotent transaction logic
    })
    .await?;
```

Demo

Additional Resources

- https://github.com/fdb-rs
- https://fdb-rs.github.io/docs/getting-started/introduction/
- https://forums.foundationdb.org/
- Bug reports, feedback and contributions are very welcome!