Lessons learned from deploying Rust apps in production

Leon Oosterwijk
Akuna Capital
Oct 2024

Overview

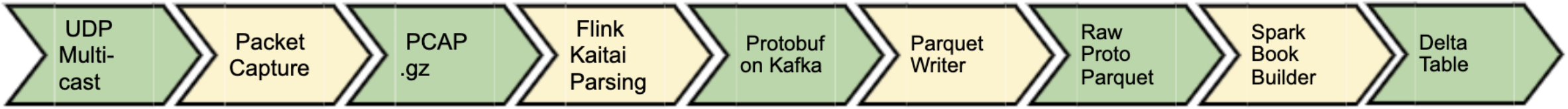
- Introduction
- Use-case
- Outcome
- Implementation details
- Lessons learned
- Things I wish for

Introduction

- Akuna Capital
  - HFT Trading firm
  - HQ in Chicago
  - Trading APAC markets from Sydney
  - Focused on Index Option Market making
  - Participant on NSE, the world's largest exchange by volume
  - Large technology footprint in C++
- Leon Oosterwijk
  - Tech lead for 'Core Systems'
  - Experience across telco, ecommerce, banking and trading
  - Mostly focused on 'business' languages (Java, Python)
  - 2 years experience with Rust -- 1 in production

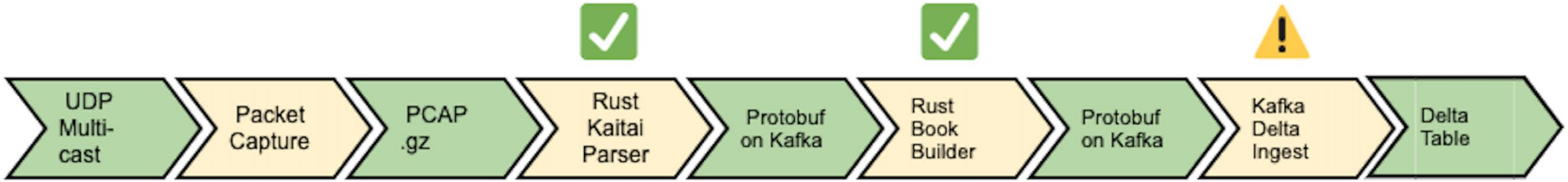
Use Case

- Data pipeline for NSE Market data
  - Approx 7-10 Billion events per day (1TB compressed)
- Existing pipeline was brittle, slow and expensive
- Book building requires a lot of state



Outcome

- Deployed PCAP Parser
- Deployed Book Builder
- Kafka Delta Ingest in dev – pending prod deploy
- Resource usage approx 1/3 of previous solution (parser)
- No more delay from realtime



Implementation Details

- Libraries
  - Etherparse, Flatbuffers, Kaitai for pcap parsing
  - Rust-rdkafka for kafka
  - Protofish and protobuf
  - Tokio and rayon for async / parallelism
  - Grafana for observability (metrics-rs)
- Deployment
  - Container in EKS (Exploring graviton deployment)
  - Cron and Lag based scaling

Lessons learned

- Many rust libraries are dormant or immature
- `async` is infectious
- If it compiles, it runs
- Controlling memory footprint with librdkafka is tricky
- For long running apps, leaking to `'static` is sometimes best
- `.clone()` first, optimize later
- Only store references in structs if you really really have to

Things I wish for. . .

- Flink style as-of join capability
- Actor-like framework
- Faster compiles (or is it the linking that’s slow?)
- Reflection