Pipelines and Streaming gRPC

Background

UTD

TOYOTA
connected

Agenda

- An introduction to gRPC
- Pipelines in Go
- A basic gRPC example
- A Streaming gRPC example
- Next Steps
- Resources
- Q&A

What is gRPC?

- An RPC framework
- HTTP/2 transport
- Interfaces defined via Protocol Buffers
    - Service Descriptions
    - Payload Structure Definition
- Tooling for generating client/server stubs in a number of languages
    - C++, C#, Dart, Go, Java, Node.js, Objective-C, PHP, Python, Ruby

Why use gRPC?

- Efficiency of Protocol buffers
  - Compact on the wire
  - Generally quicker (de)serialization times than JSON
- Support for bidirectional streaming
- Protocol buffer definitions and code generation allow for the easy distribution of SDKs or interface specifications

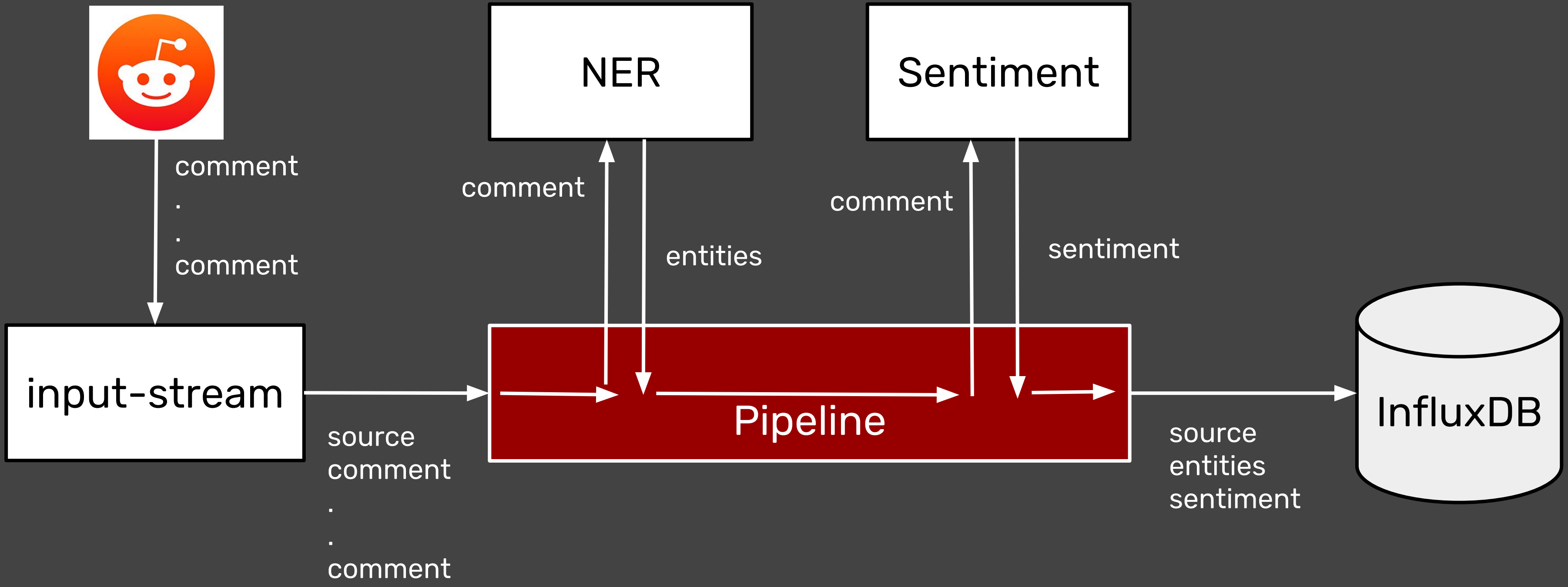
Pipelines

- Goroutines and channels facilitate development of streaming pipelines
- Pairs well with streaming gRPC APIs
- Typically, each step in a pipeline receives a message on an input channel performs a transformation or operation, then puts the result on an output channel

Pipeline Example

- Goal: As a wanna-be day trader, I’d like to improve my knowledge of market conditions, so I can trade more effectively.
- Hypothesis: I can find correlations between the sentiment of companies among internet users, and the value of those companies. Using that information, I can make predictions about the future value of companies.

Pipeline Example



Next Steps

- Better Error Handling
- Support Cancellation
- Incorporate more sources
- Incorporate more information, such as security prices

Resources

- gRPC Website
  - https://grpc.io/docs/
- Concurrency in Go
  - https://www.oreilly.com/library/view/concurrency-in-go/9781491941294/
- This Project
  - https://github.com/datravis/go-meetup-solution
- Go Pipelines Blog
  - https://blog.golang.org/pipelines