¿QUIÉNES SOMOS?

Digital Lab

ADL DIGITAL LAB

¿Quiénes somos?

Somos una compañía 100% digital. Combinamos la innovación y el diseño para lograr la transformación digital de nuestros usuarios, mediante soluciones que van más allá de la tecnología y que atienden los retos del mercado y se apalancan de las nuevas oportunidades.

Nuestros servicios

- Digitalización e innovación
- Agilidad
- Diseño (UX/UI)
- Arquitectura
- DevOps
- Cloud Service
- Seguridad
- Marketing Digital
- Data

Resultados

- A la fecha tenemos 20 MVPs en digitalización
- Lanzamiento de ***dale!*** La SEDPE de las entidades aval
- Procesos de Transformación Cultural

Digital Lab

¿QUÉ ES GUILD GOLANG?

Digital Lab

GUILD GOLANG es un grupo de investigacion motivado por el auto aprendizaje del lenguaje de programación Golang, queremos proponer formas eficientes de implementar soluciones que automaticen nuestras tareas del dia a dia y den valor agregado al rendimiento de nuestras APIs.

Digital Lab

Manejo de *Streams* en Golang con YoMo

Jheison Alejandro Morales
jheison.morales@avaldigitallabs.com
Desarrollador

Digital Lab

Agenda ~ 50 min

- Motivación ~2.5 min
- Repaso modelo de concurrencia en Golang ~17.5 min
- Introducción a YoMo ~10 min
- Revisión del repo ~15 min
- Preguntas: ~5 min

Digital Lab

Descargo de responsabilidad

En esta presentación nos vamos a concentrar en:

- Golang + YoMo (etapas trempanas de desarrollo)

Digital Lab

Motivación

- Estudiar características de Golang útiles para el procesamiento de flujos de eventos.

- Aprender leyendo código.

Digital Lab

Modelos de concurrencia en Golang

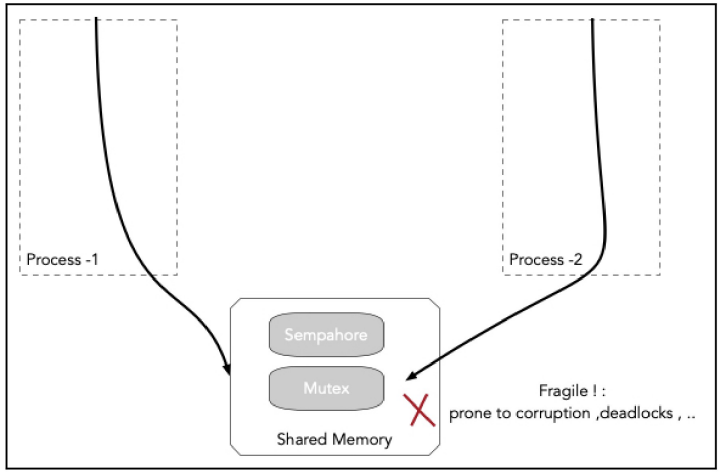
- Goroutines
- Channels
- RxGo
- Streams

Digital Lab

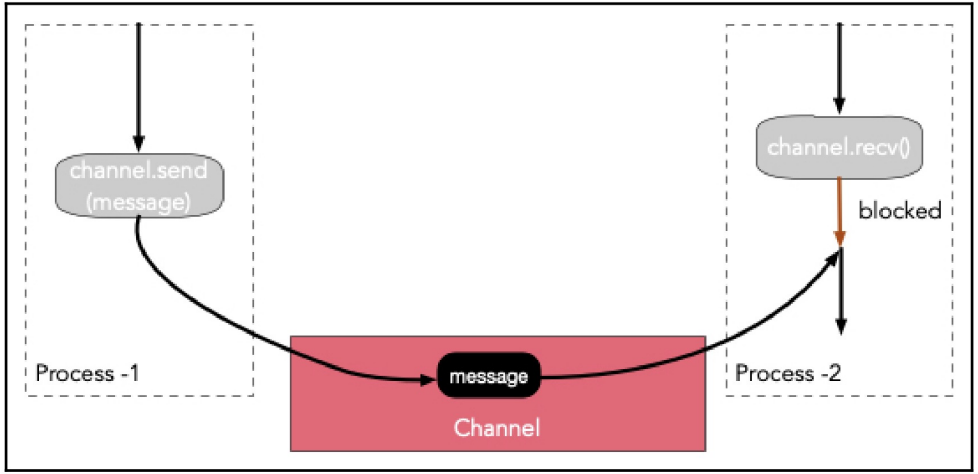
Modelos de concurrencia en Golang

Goroutines: Modelo CSP

Antes



Después

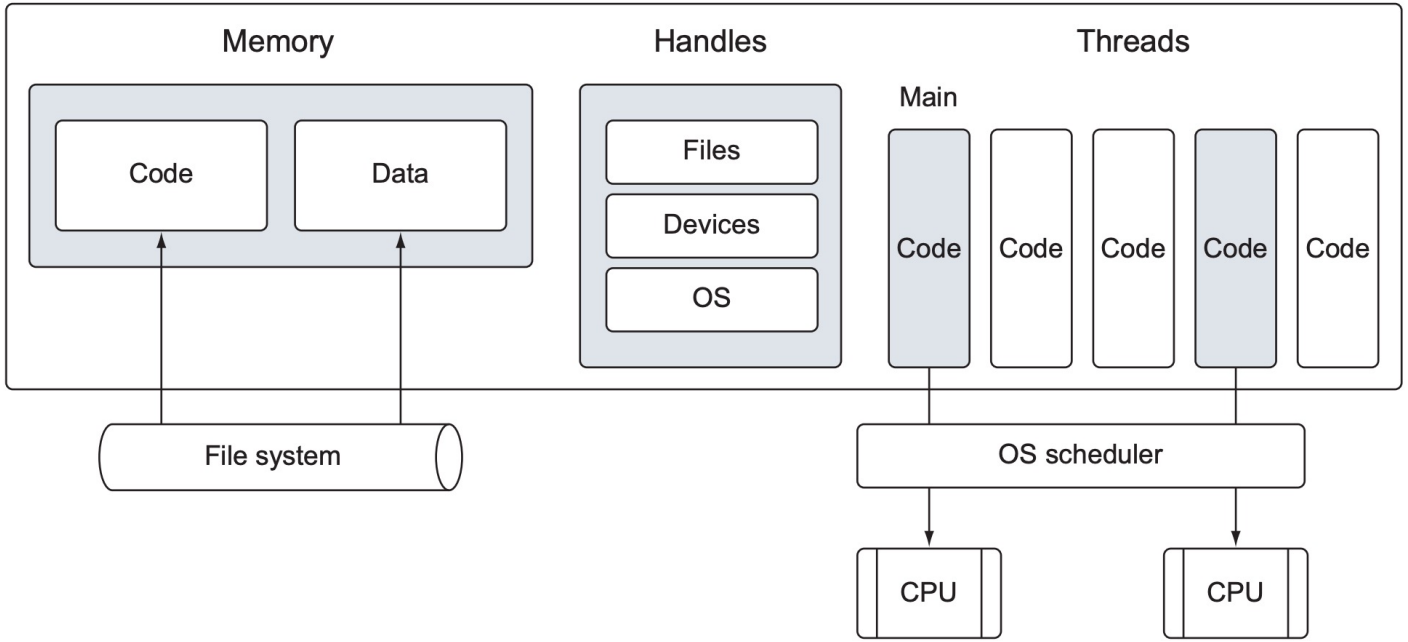


Tomado de: *Hands-On Software Architecture with Golang*, Jyotiswarup Raiturkar, Packt Publishing, 2018. Chapter 1, pag 33-34

Digital Lab

Modelos de concurrencia en Golang

Goroutines (I)



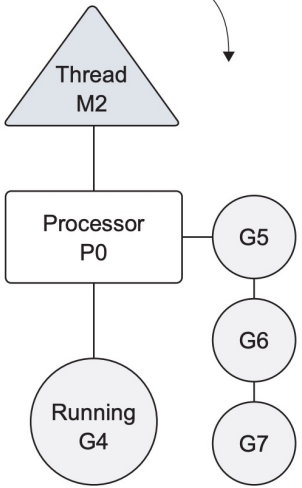
Tomado de: Go in Action, William Kennedy, Brian Ketelsen, Erik St. Martin, Manning, 2016. Chapter 6. Concurrency

Digital Lab

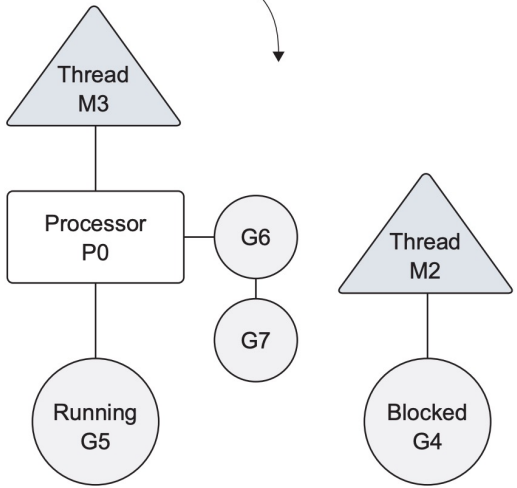
Modelos de concurrencia en Golang

Goroutines (II)

The Go runtime schedules goroutines to run in a logical processor that is bound to a single operating system thread. When goroutines are runnable, they are added to a logical processor's run queue.



When a goroutine makes a blocking syscall, the scheduler will detach the thread from the processor and create a new thread to service that processor.



Tomado de: *Go in Action*, William Kennedy, Brian Ketelsen, Erik St. Martin, Manning, 2016. Chapter 6. Concurrency

Digital Lab

Modelos de concurrencia en Golang

Channels (I)

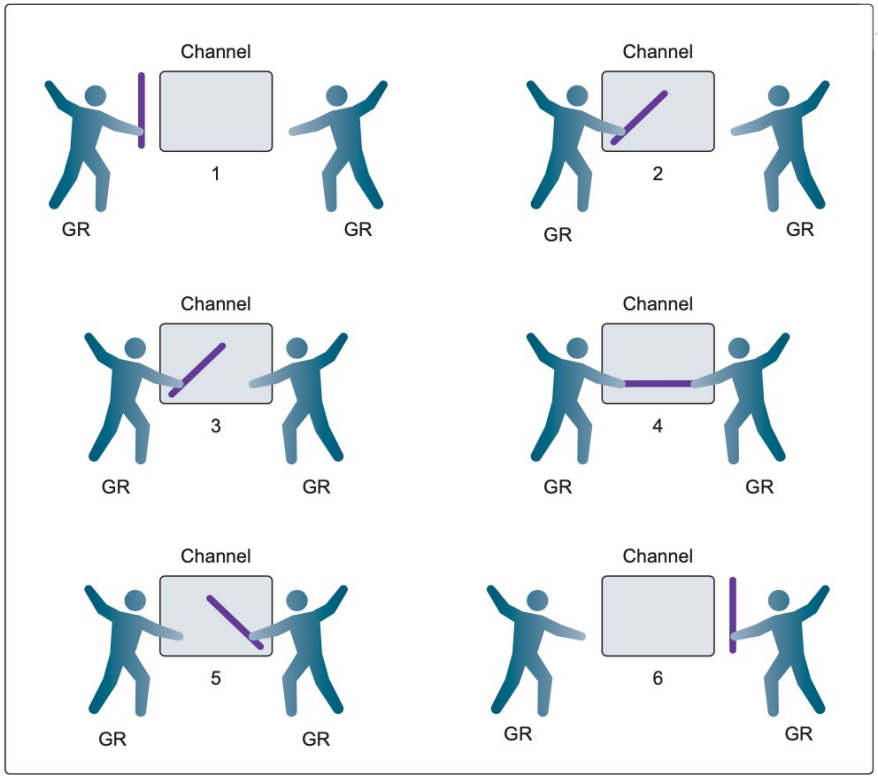


Figure 6.6 Synchronization between goroutines using an unbuffered channel

Tomado de: *Go in Action*, William Kennedy, Brian Ketelsen, Erik St. Martin, Manning, 2016. Chapter 6. Concurrency

Digital Lab

Modelos de concurrencia en Golang

Channels (II)

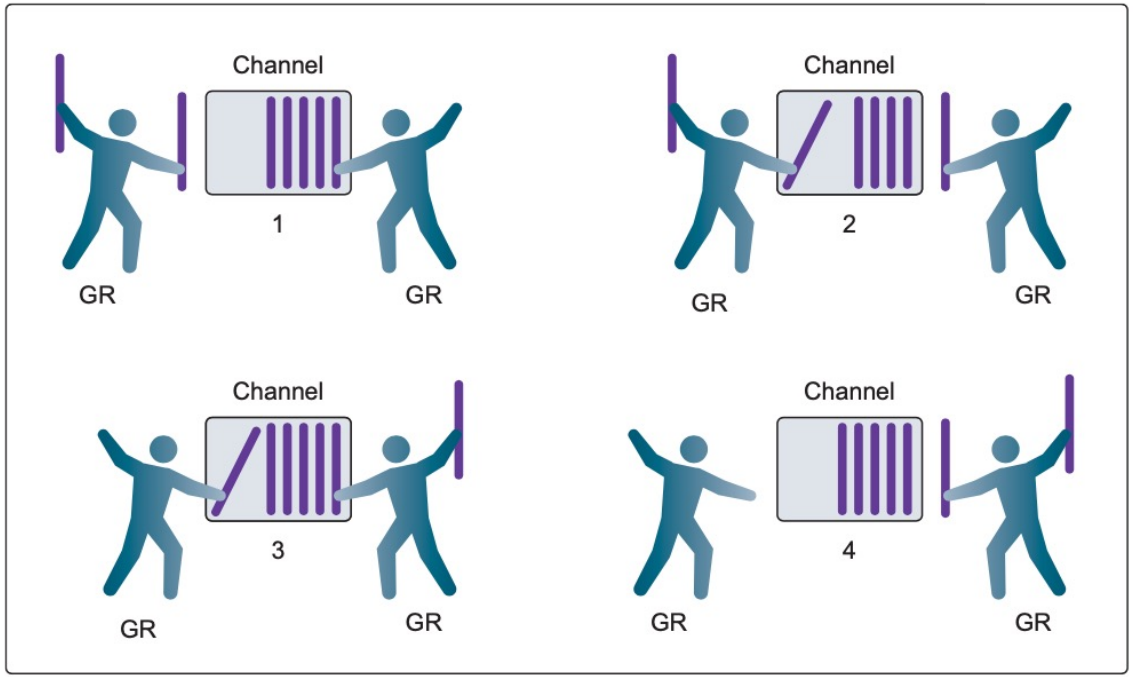


Figure 6.7 Synchronization between goroutines using a buffered channel

Tomado de: *Go in Action*, William Kennedy, Brian Ketelsen, Erik St. Martin, Manning, 2016. Chapter 6. Concurrency

Digital Lab

Modelos de concurrencia en Golang

Streams (I)

Modelos de concurrencia en Golang

a Store raw events

=History of what happened

b Store aggregated data

=Current state / end result

a Raw events

=DB writes =Buttons

b Aggregated data

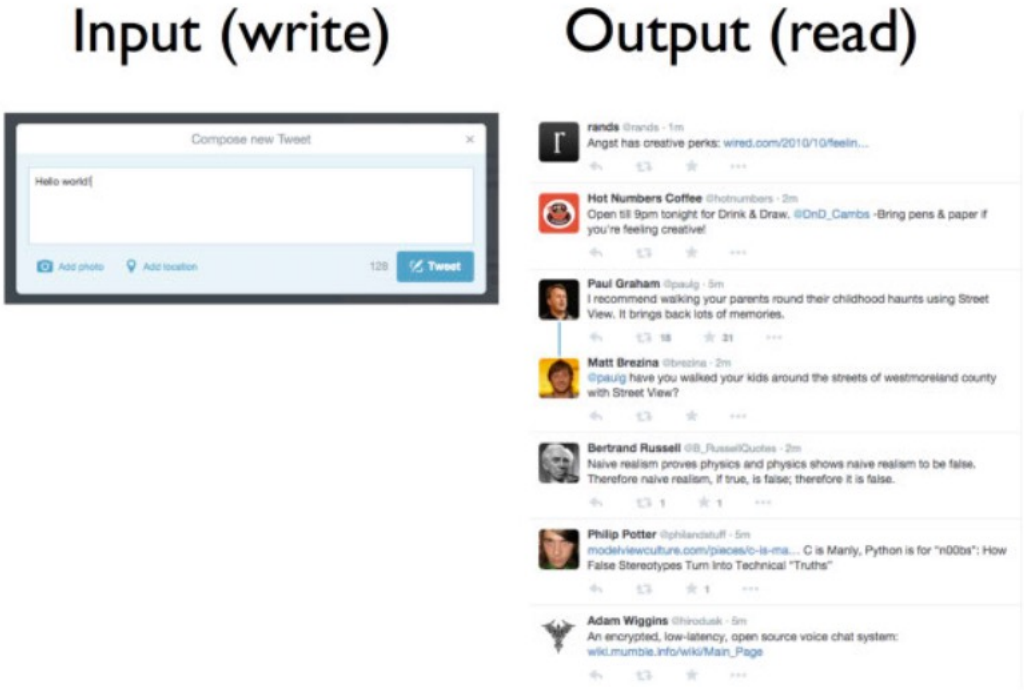
=DB reads =Screens

Tomado de: *Making Sense of Stream Processing*, Martin Kleppman, O'Reilly Media, 2016. Chapter 1

Digital Lab

Modelos de concurrencia en Golang

Streams (II)

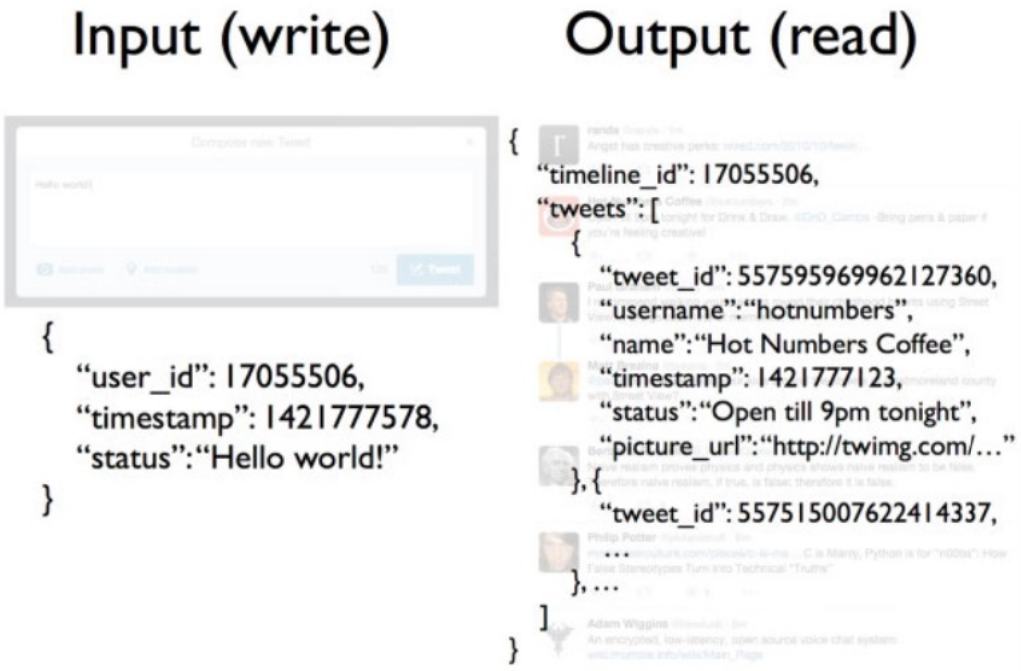


Tomado de: *Making Sense of Stream Processing*, Martin Kleppman, O'Reilly Media, 2016. Chapter 1

Digital Lab

Modelos de concurrencia en Golang

Streams (II)

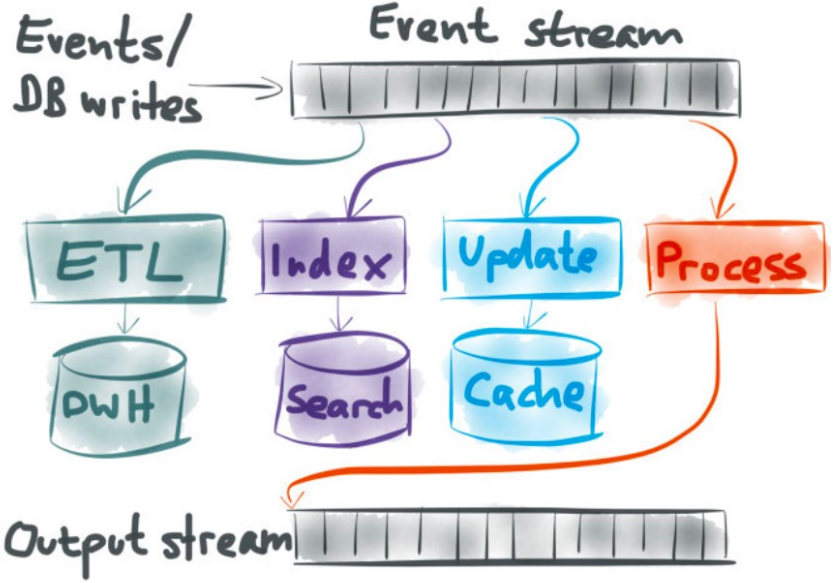


Tomado de: *Making Sense of Stream Processing*, Martin Kleppman, O'Reilly Media, 2016. Chapter 1

Digital Lab

Modelos de concurrencia en Golang

Streams (III)

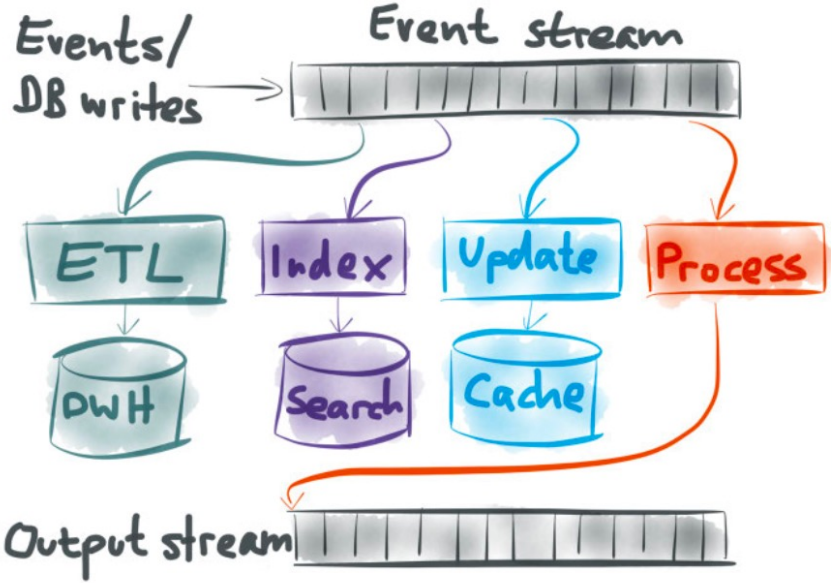


Tomado de: *Making Sense of Stream Processing*, Martin Kleppman, O'Reilly Media, 2016. Chapter 1

Digital Lab

Modelos de concurrencia en Golang

Streams (III)



Beneficios:

- Bajo acoplamiento
- Alto desempeño escritura & lectura
- Escalabilidad
- Flexibilidad
- Auditabilidad/recuperación de errores

Tomado de: *Making Sense of Stream Processing*, Martin Kleppman, O'Reilly Media, 2016. Chapter 1

Digital Lab

Modelos de concurrencia en Golang

RxGo

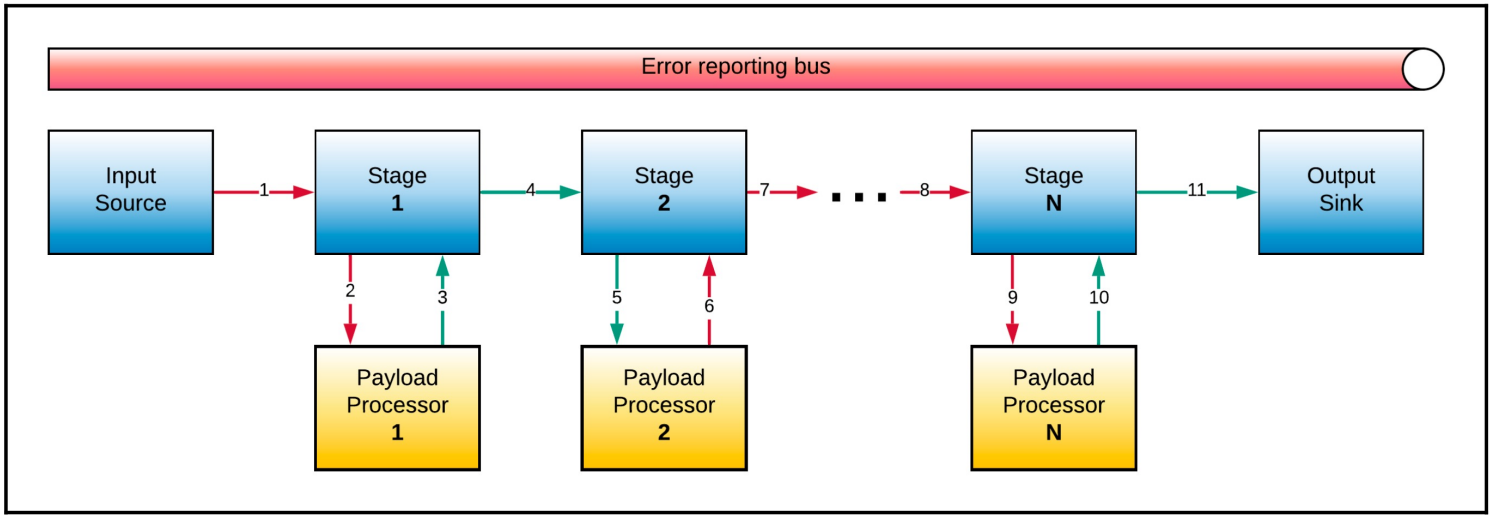


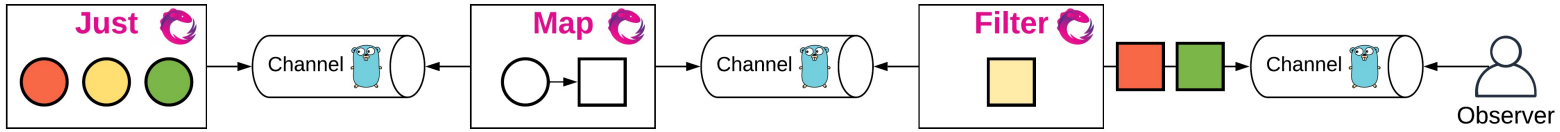
Figure 1: A generic, multistage pipeline

Tomado de: *Hands-On Software Architecture with Golang*, Jyotiswarup Raiturkar, Packt Publishing, 2018.

Digital Lab

Modelos de concurrencia en Golang

RxGo



Conceptos:

- Backpressure
- Hot vs Cold
- Lazy vs. Eager Observation
- Sequential vs. Parallel Operators

Operadores:

- Creación
- Transformación
- Filtrado
- Condicionales/Operaciones booleanas
- Combinación
- Agregación
- Manejo de errores

Tomado de: https://github.com/ReactiveX/RxGo

Digital Lab

Introducción a YoMo

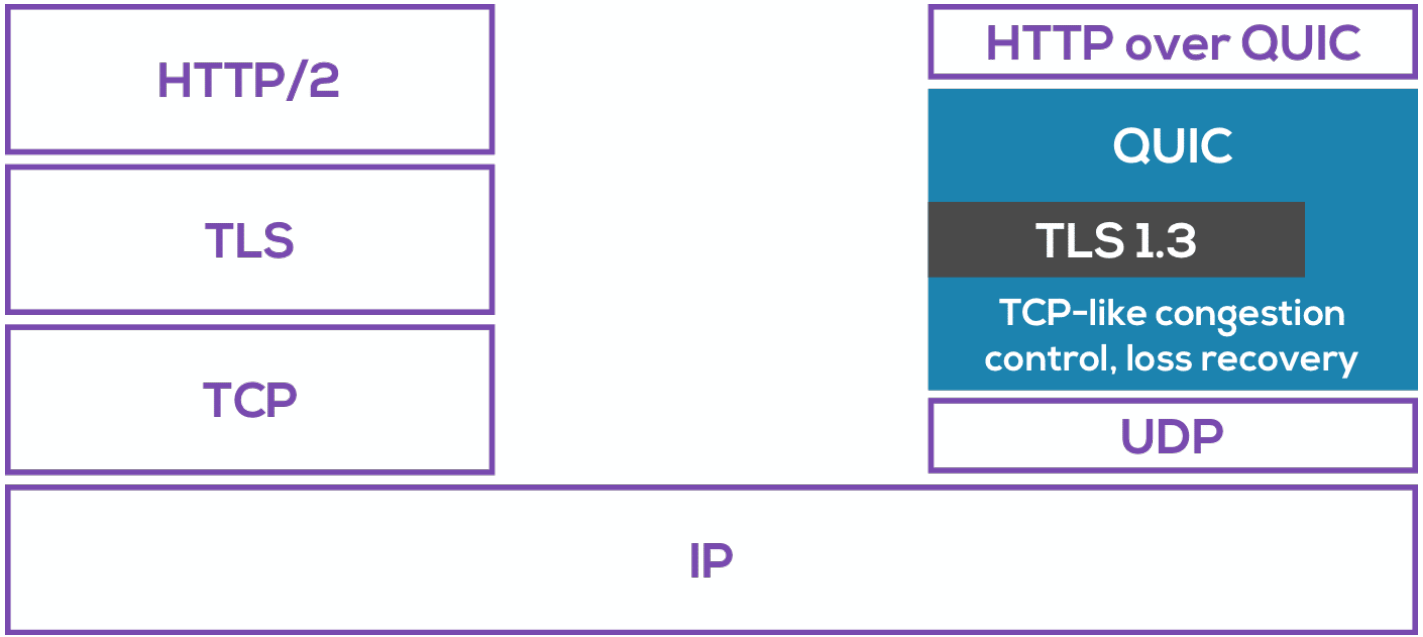
- QUIC
- Arquitectura

Digital Lab

Introducción a YoMo

QUIC (I)

Quick UDP Internet Connection

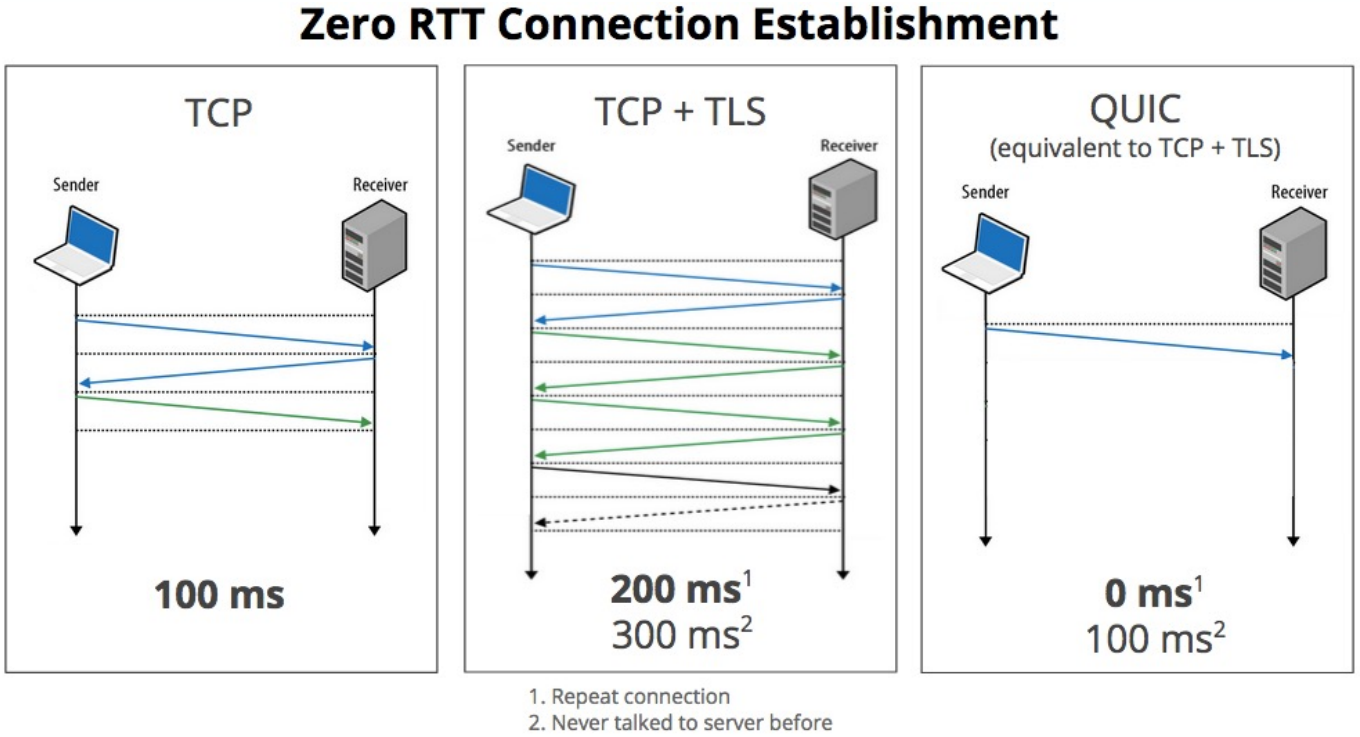


Tomado de: https://www.callstats.io/blog/2017/02/03/web-protocols-http2-quic

Digital Lab

Introducción a YoMo

QUIC (II)



Tomado de: https://blog.chromium.org/2015/04/a-quic-update-on-googles-experimental.html

Digital Lab

Introducción a YoMo

QUIC (III)

QUIC como capa de transporte puede usarse en otros lenguajes:

- https://www.apiref.com/nodejs/quic.html
- https://kachayev.github.io/quiche4j/
- etc.

Digital Lab

Introducción a YoMo

Arquitectura

Objetivos de diseño:

- Facilitar la construcción de aplicaciones **sensibles a la latencia**.
- Manejar **datos continuos de alta frecuencia con procesamiento de flujo**.
- Creación de sistemas complejos con **arquitectura sin servidor.**

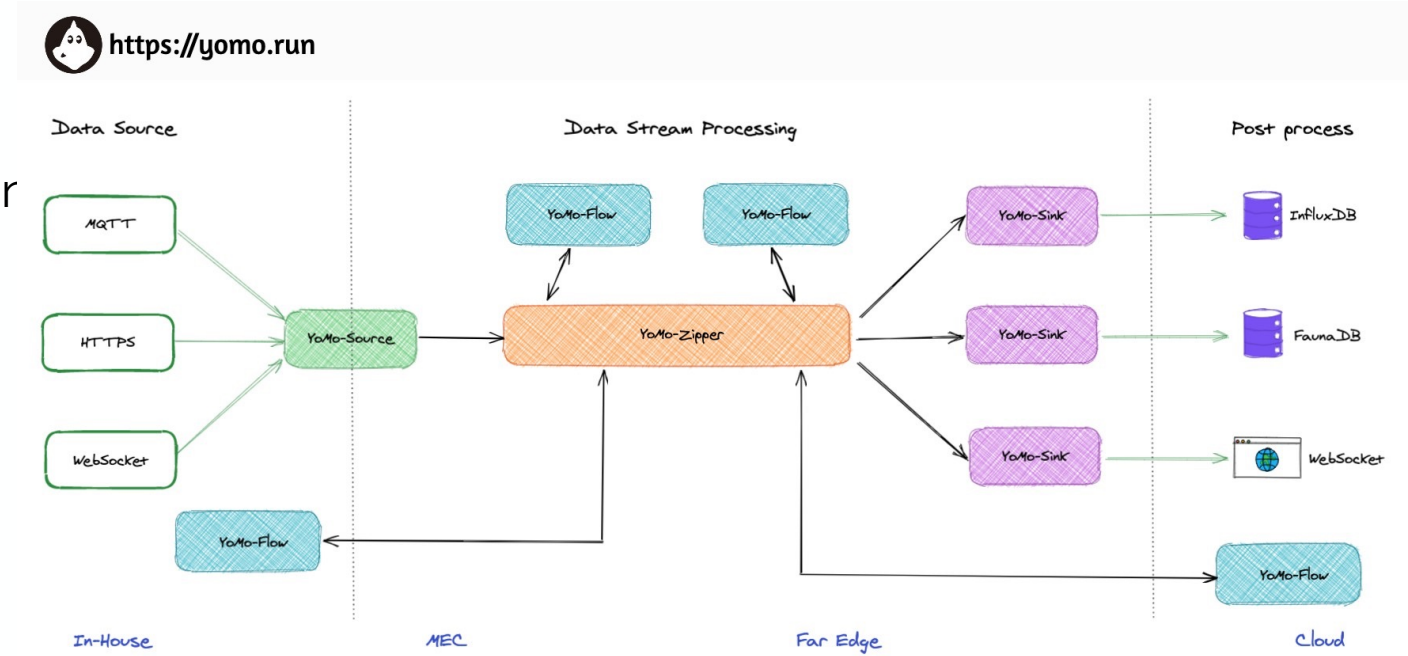
Fuente https://github.com/yomorun/yomo

Introducción a YoMo

Arquitectura

Componentes:

- YoMo Source
- YoMo Zipper
- Stream Functior

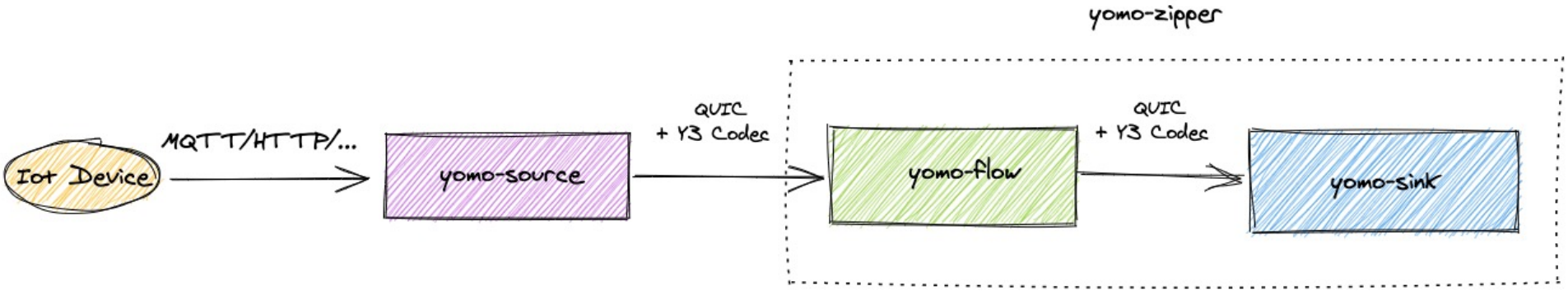


Tomado de: https://github.com/yomorun/yomo

Digital Lab

Introducción a YoMo

YoMo Source

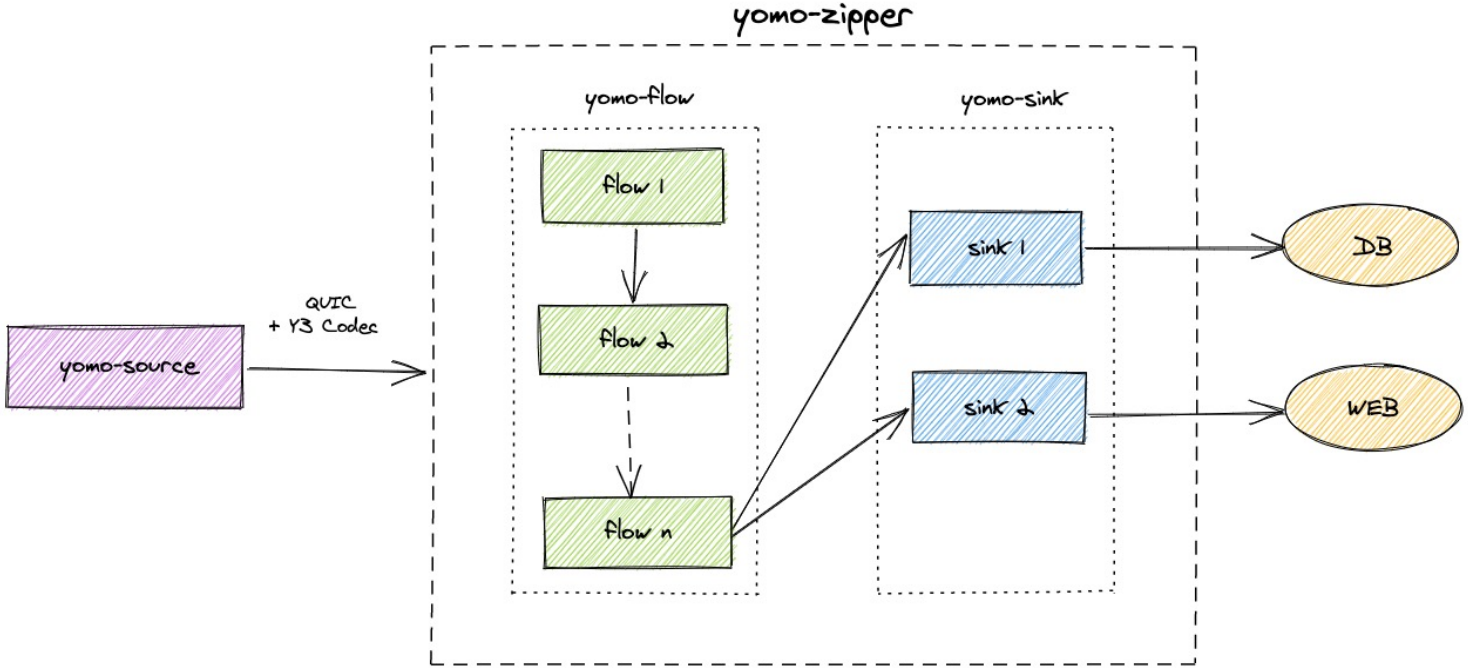


Tomado de: https://docs.yomo.run/source

Digital Lab

Introducción a YoMo

YoMo Zipper



Tomado de: https://docs.yomo.run/zipper

Digital Lab

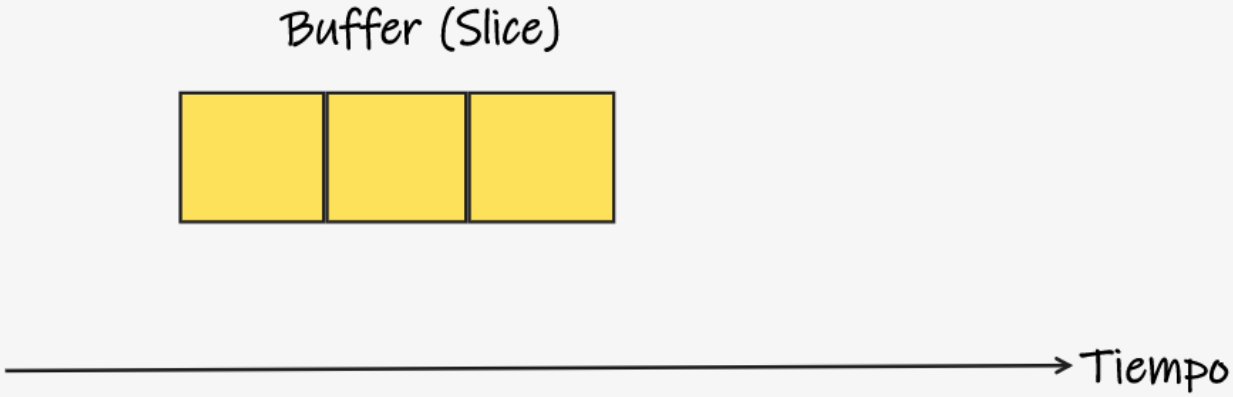
Repositorio de apoyo

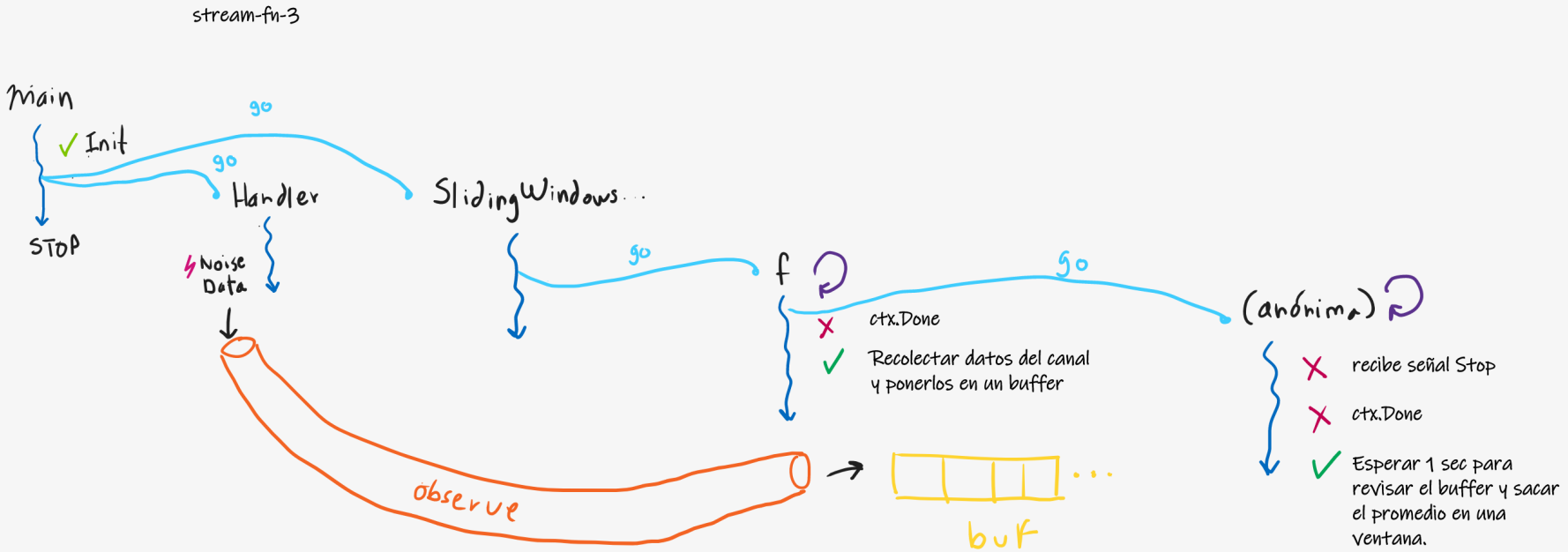
Ahora veamos a revisar el repo

https://github.com/alejandro56664-adl/talk-intro-streams-golang

Digital Lab

Filtro de media móvil




stream-fn-3
main
Init
go
go
Handler
SlidingWindows...
STOP
Noise Data
go
f
go
(anónima)
ctx.Done
Recolectar datos del canal y ponerlos en un buffer
recibe señal Stop
ctx.Done
Esperar 1 sec para revisar el buffer y sacar el promedio en una ventana.
observe
buf

Digital Lab

Referencias

- *Go in Action*, William Kennedy, Brian Ketelsen, Erik St. Martin, Manning, 2016
- *Hands-On Software Architecture with Golang*, Jyotiswarup Raiturkar, Packt Publishing, 2018
- *Making Sense of Stream Processing*, Martin Kleppman, O'Reilly Media, 2016
- https://github.com/ReactiveX/RxGo, Revisado el 2 de noviembre 2021
- https://www.callstats.io/blog/2017/02/03/web-protocols-http2-quic , Revisado el 2 de noviembre 2021
- https://blog.chromium.org/2015/04/a-quic-update-on-googles-experimental.html , Revisado el 2 de noviembre 2021
- https://docs.yomo.run/ , Revisado el 2 de noviembre 2021

Digital Lab

¿Preguntas?

Digital Lab

Gracias

Digital Lab