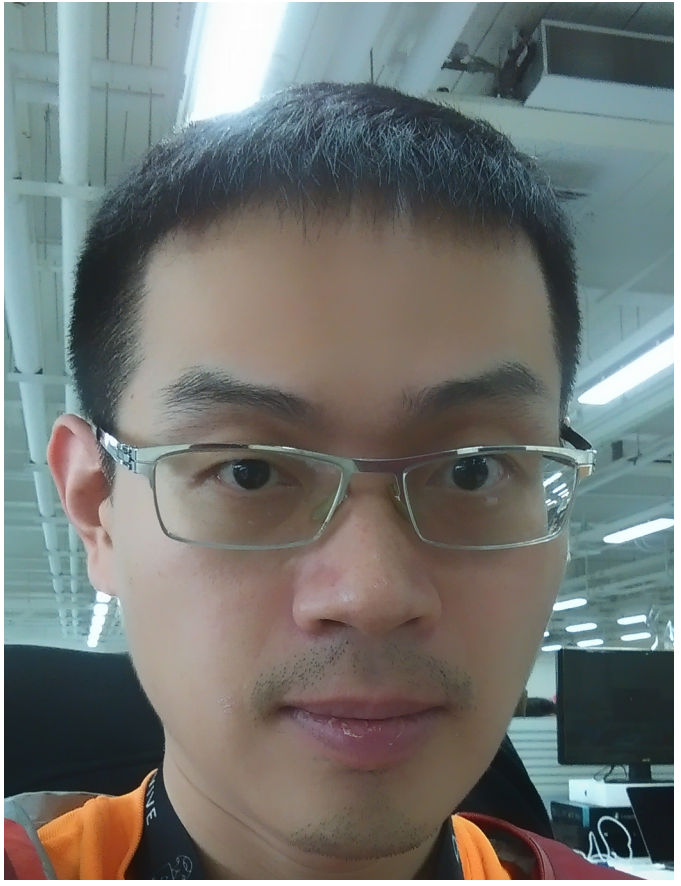
今度のご注文は？？

20191217 chenghsien wen

Who Am I

- CH Wen, Simon
- Vpon AdNext
  Sr. Engineer
- 3 years of Scala
  developer



Outline

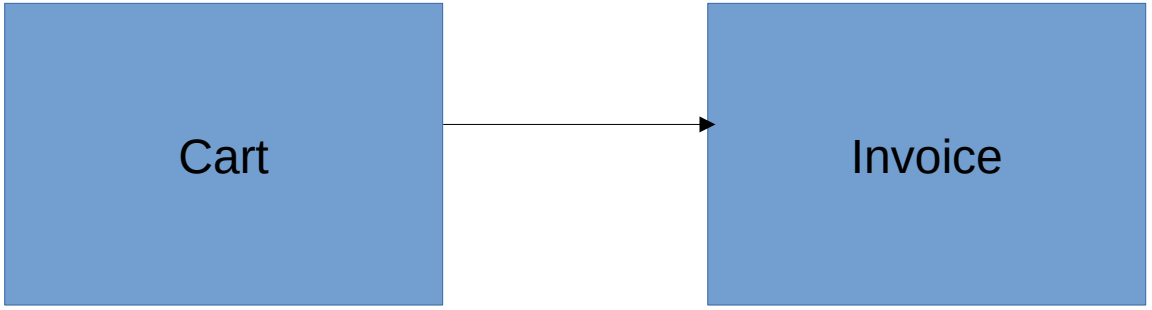
- Some essential concepts
- Finagle introduction
- Akka http introduction
- Comparison
- Discussion

Some essential concepts

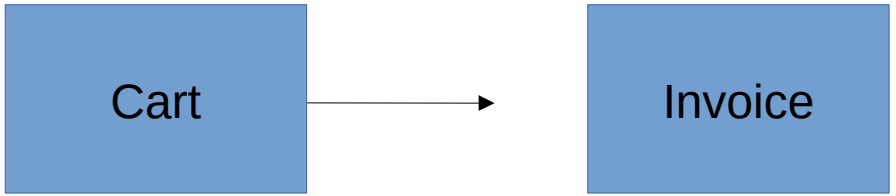
- Reactive programming
- Concurrency is not parallelism

Reactive: dynamic data flow

There is a e-commerce platform, we have cart to add products and we need to update invoice



Passive



- Remote setters and updates

```
object Cart {
    def addProduct(product: Product): Unit = {
        ...
        Invoice.updateProduct(product)
    }
}
```

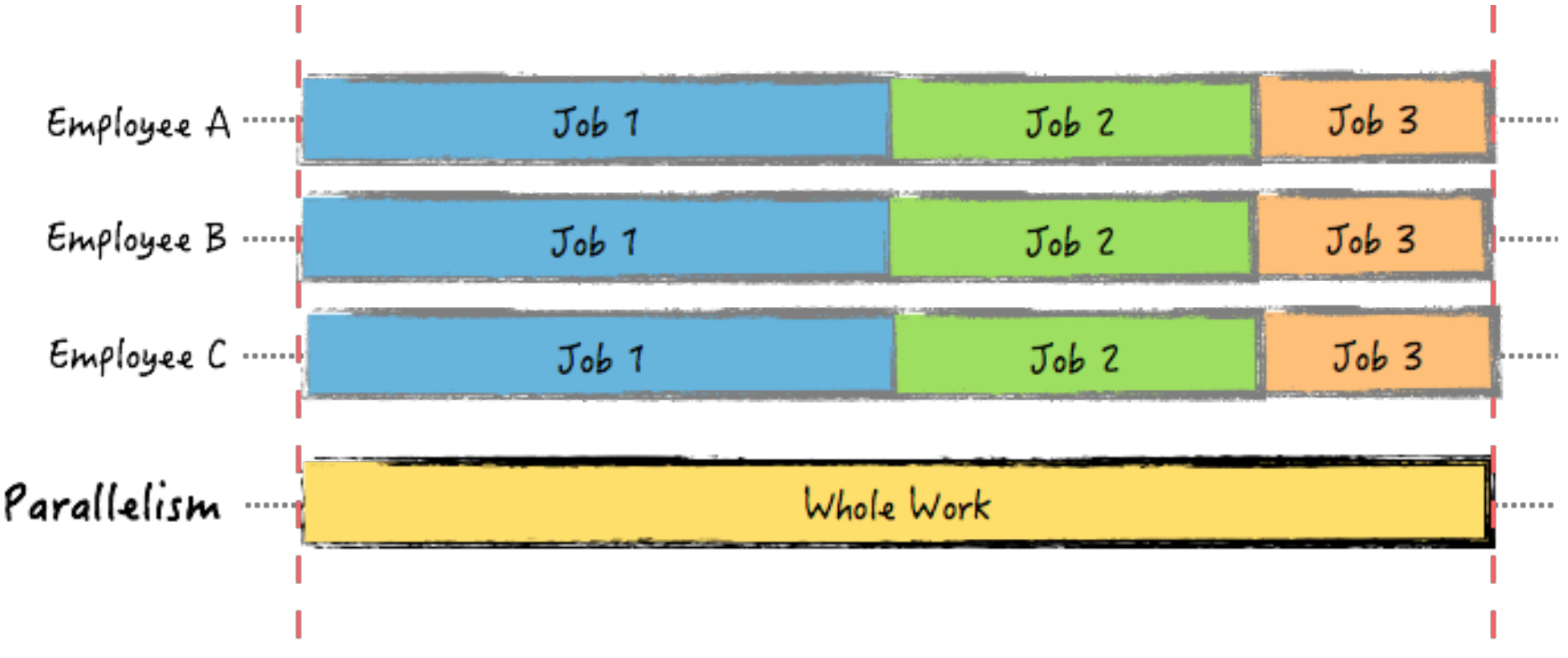
Reactive

- Events, observation, and self-updates

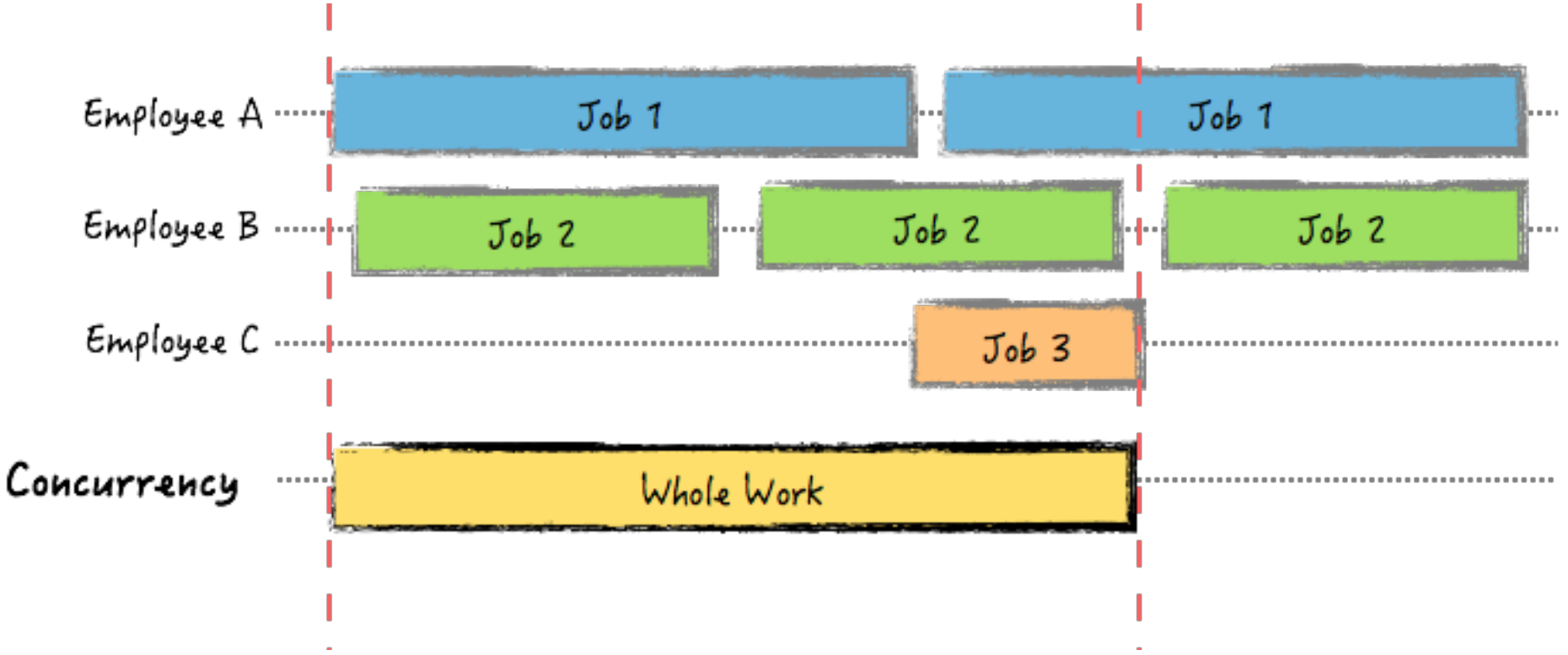
```
object Invoice {
    ...
    def receive = {
        case Cart.onProductAdded(product) => {
            ...
            this.updateInvoicing(product)
        }
    }
}
```

	Passive	Reactive
How does it work?	*Find usages*	Look inside
What does it affect?	Look inside	*Find usages*

Concurrency is not parallelism



Concurrency is not parallelism

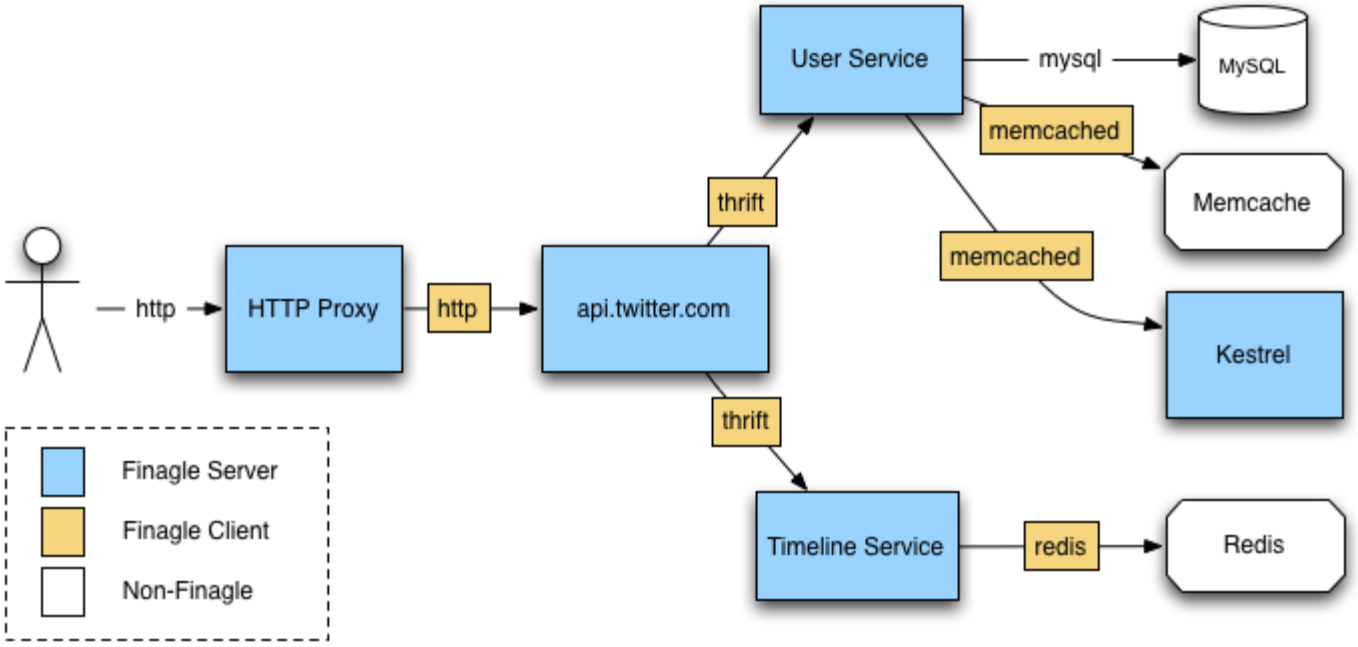


Finagle introduction

Thrift, http, memcache,
mySQL...etc

Finagle is a protocol-agnostic, asynchronous RPC system for the JVM that makes it easy to build robust clients and servers in Java, Scala, or any JVM-hosted language.

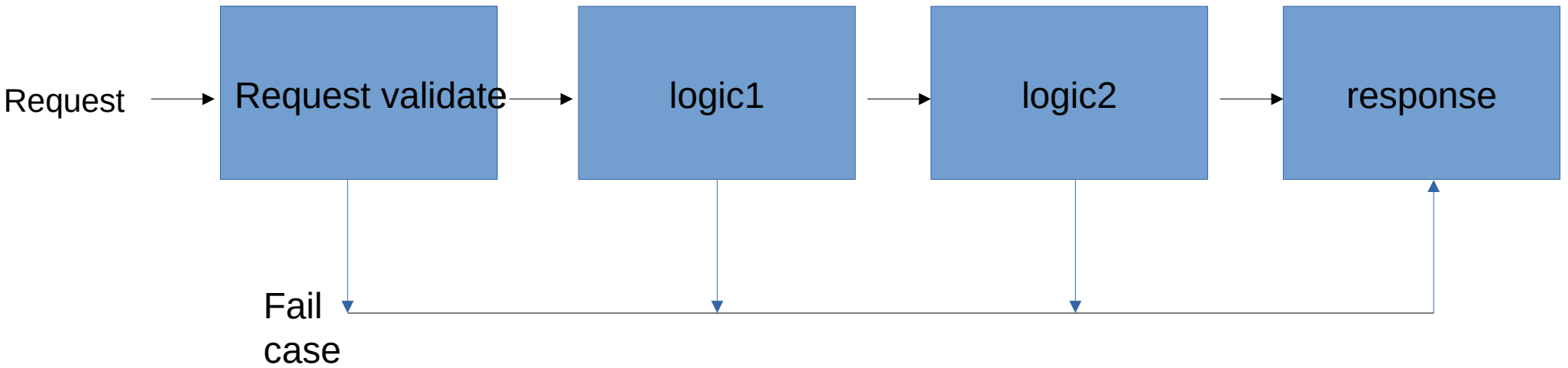
Finagle network example



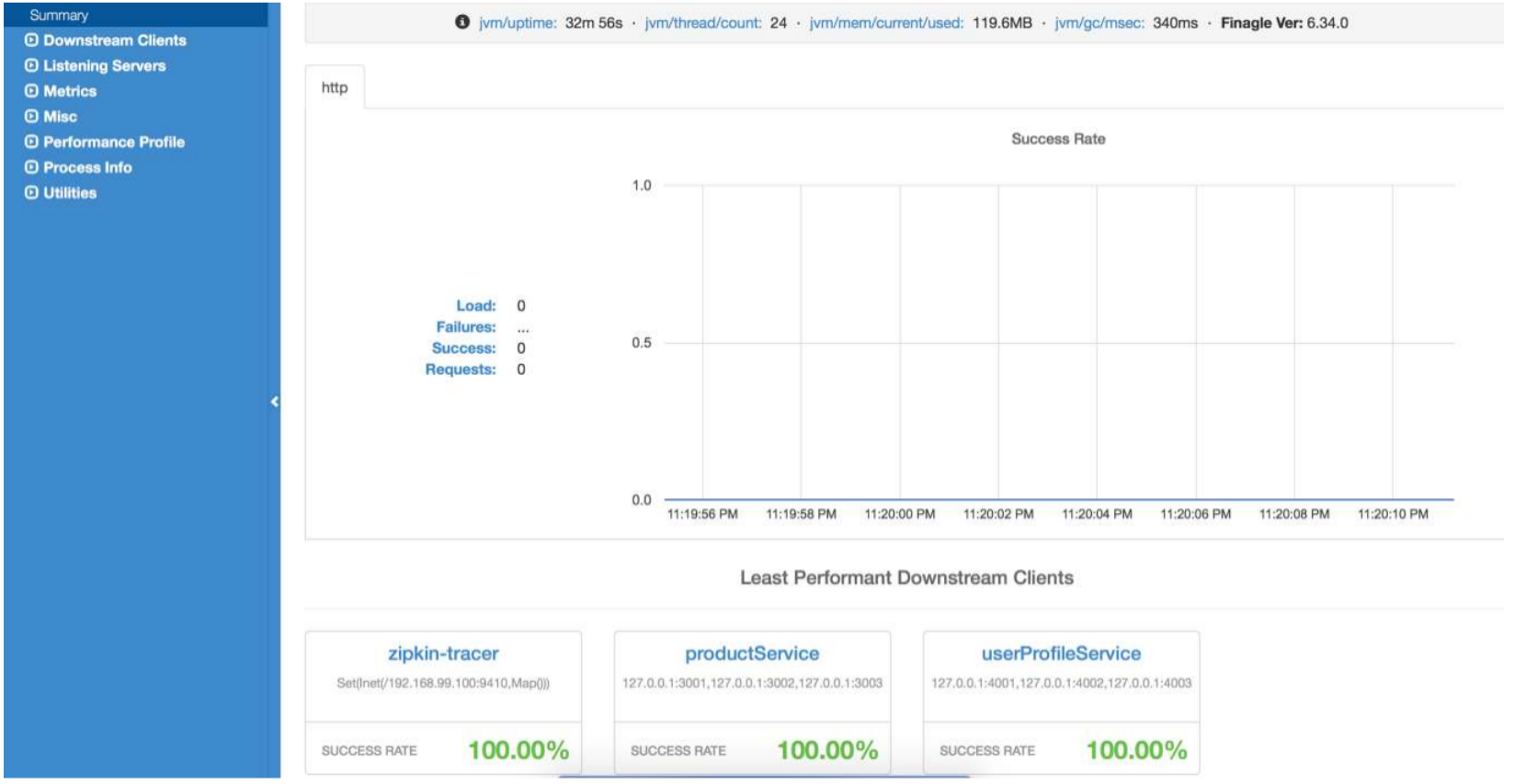
Finagle components

- Twitter server: Based on Netty
- Future: thread unit
- Service: main function
- Filter: for timeouts, retry policies, service statistics, and authentication
- Admin interface UI

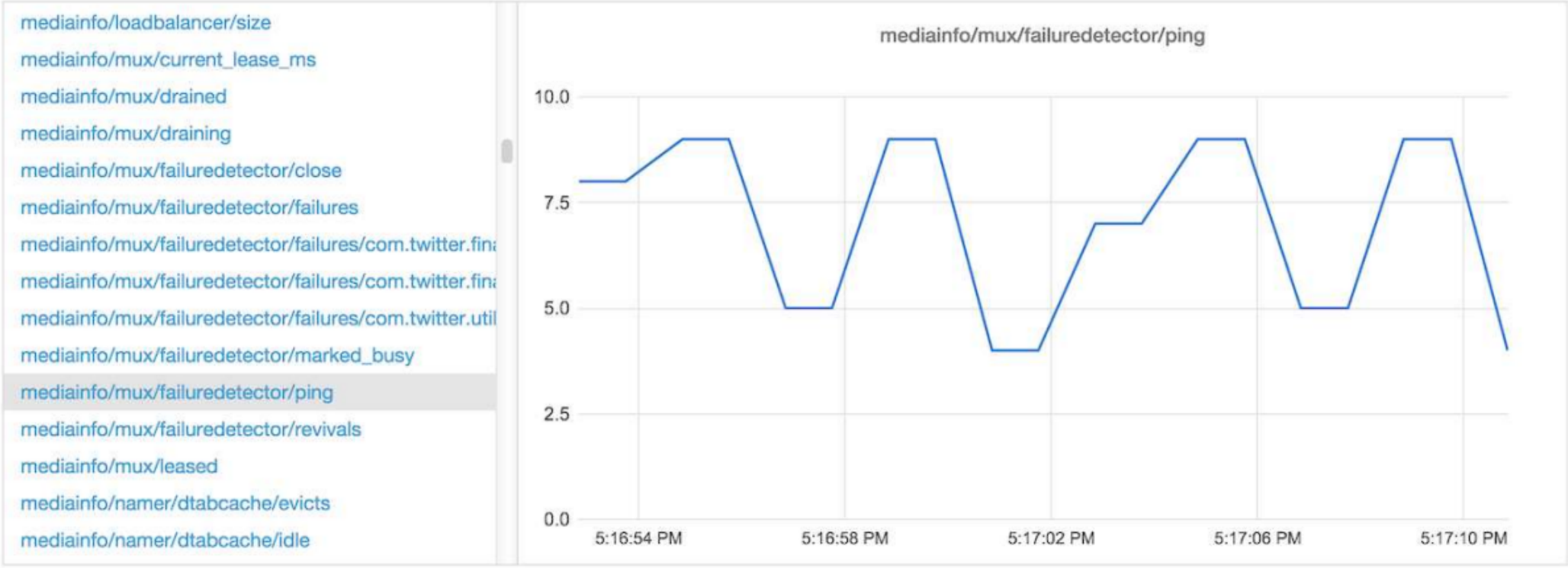
Finagle: service as function



Finalge admin UI



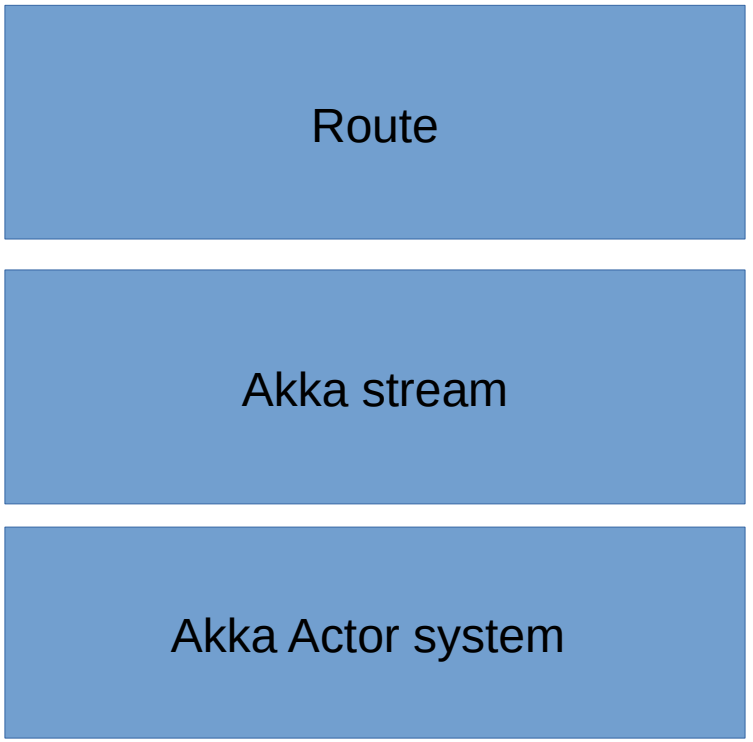
Finalge admin UI(cont.)



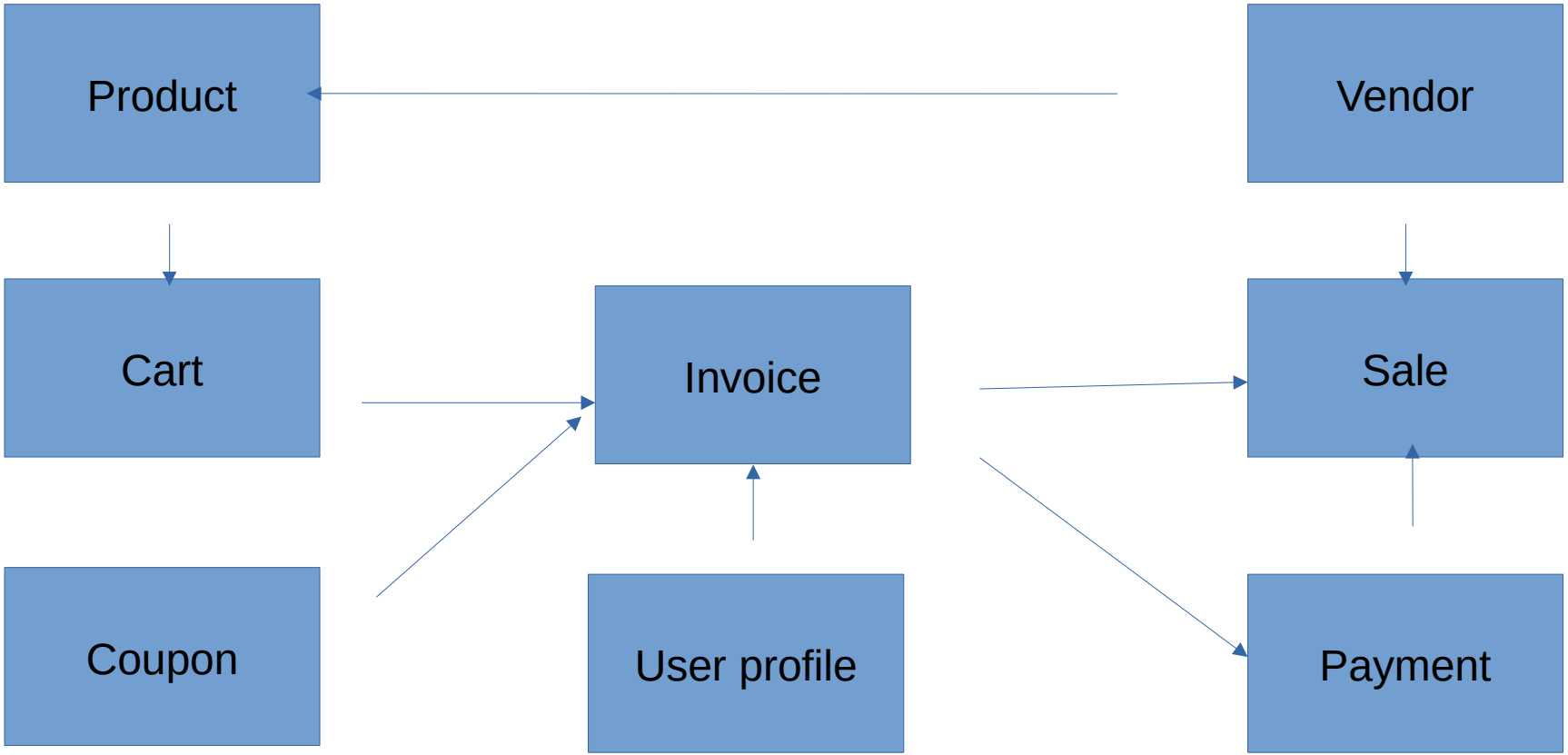
Akka http

The Akka HTTP modules implement a full server- and client-side HTTP stack on top of *akka-actor* and *akka-stream*. It's not a web-framework but rather a more general toolkit for providing and consuming HTTP-based services. While interaction with a browser is of course also in scope it is not the primary focus of Akka HTTP.

Akka http structure



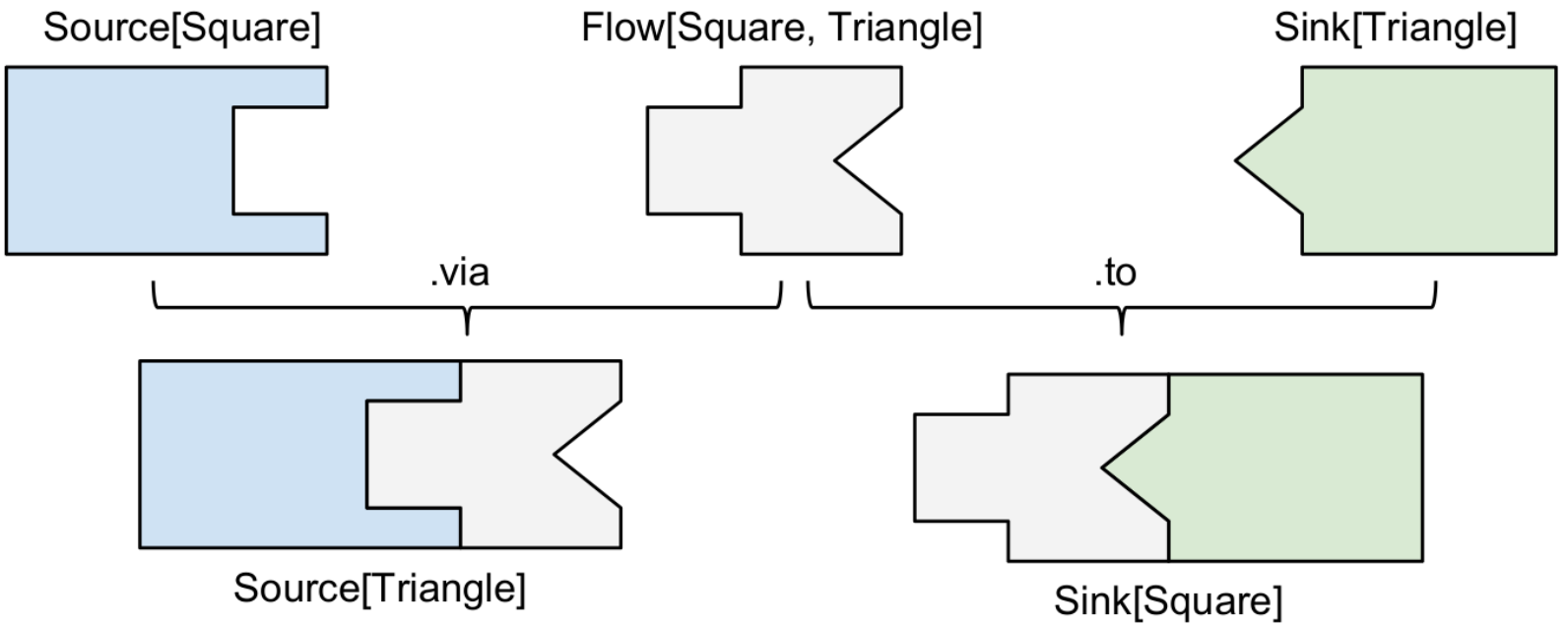
Akka actor example



Actor features

- Life cycle management:restart, stop...etc
- Life cycle monitoring: death watch
- Persistence:history record, event recovery
- fork: create child
- TestKit

Akka stream



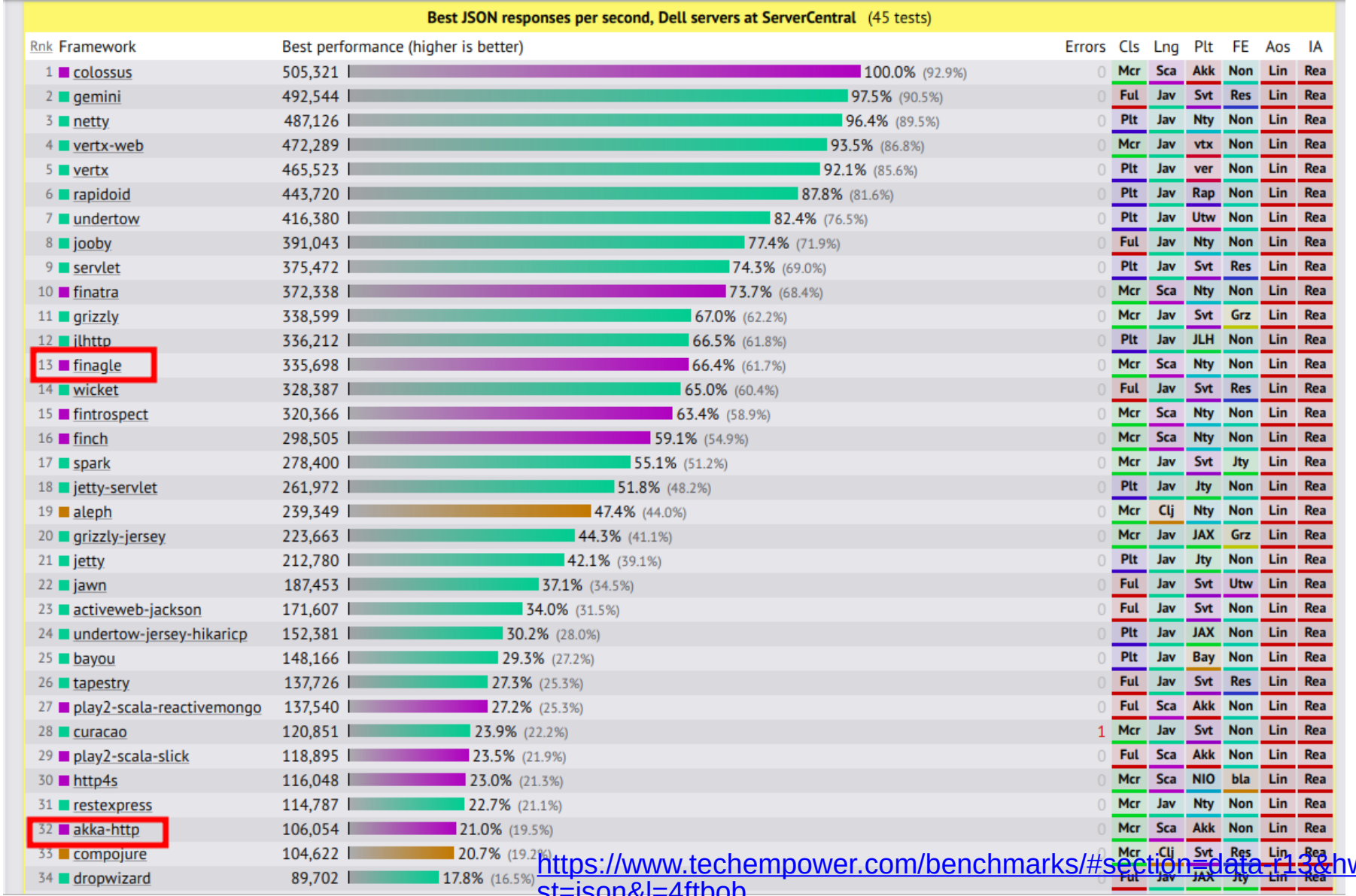
https://softwaremill.com/reactive-streams-in-scala-comparing-akka-streams-and-monix-part-1/

akka http vs spray

- Akka HTTP is the successor for spray. With the first non-experimental release of Akka HTTP, spray has reached its end-of-life. Akka HTTP is a reimplementation of HTTP based on akka-stream (former spray-can) which adds streaming support on all levels. The popular high-level routing DSL (former spray-routing) has mostly been kept but was made more consistent and simplified where possible. While underlyings have changed a lot, many of the high-level features and syntax of the routing DSL have only changed superficially so that code can hopefully be converted with little effort.

https://doc.akka.io/docs/akka-http/current/migration-guide/migration-from-spray.html

	finagle	Akka http
vendor	twitter	lightbend
Feature support		
Rest api	yes	yes
concurrency	yes(future pool)	yes(akka system)
Web ui	Admin monitoring	no
Web socket	yes(not sure stable or not...)	yes
stream	no	yes
Community support	Smaller	Higher developer population
performance	faster	slower
scalability	good	Good but more flexible
Resource consumption		



https://www.techempower.com/benchmarks/#section=data-r13&hw=ph&test=json&l=4ftbob

Discussion

- End User recommend to use finagle
- **Concurrency, Highly data flow** requirements recommend to use akka

Some sample codes by Ammonite

- Ammonite script for scala 2.12
- https://github.com/chenghsienwen/AmmSandbox
-

Reference

- Reactive design and passive design
- https://youtu.be/49dMGC1hM1o
- Concurrency is not parallelism
- https://medium.com/mr-efacani-teatime/concurrency%E8%88%87parallelism%E7%9A%84%E4%B8%8D%E5%90%8C%E4%B9%8B%E8%99%95-1b212a020e30
- Finagle: https://blog.twitter.com/engineering/en_us/a/2011/finagle-a-protocol-agnostic-rpc-system.html
- Finagle https://www.slideshare.net/TomaszKogut/exploring-twitters-finagle-technology-stack-for-microservices?from_action=save
- Akka reactive stream
- https://github.com/reactive-streams/reactive-streams-jvm/blob/v1.0.2/README.md#specification
- Reactive Streams in Scala by 張瑋修 Walter Chang - JCConf 2016 R0 Day2-2
- https://youtu.be/j3blqi3g_lA
- Akka stream and monix: https://softwaremill.com/reactive-streams-in-scala-comparing-akka-streams-and-monix-part-1/
- Spary migration: https://doc.akka.io/docs/akka-http/current/migration-guide/migration-from-spray.html
- Performance compare on web framework
- https://www.techempower.com/benchmarks/#section=data-r13&hw=ph&test=json&l=4ftbob
- Netty: https://www.baeldung.com/netty
- Spary: http://spray.io/introduction/getting-started/
- Akka-http sample :https://github.com/theiterators/akka-http-microservice
- Edx online course: https://courses.edx.org/courses/course-v1:EPFLx+scala-reactiveX+1T2019/course/
- Ammonite: https://ammonite.io/#Ammonite
- 【46選】あのサービス・アプリのアーキテクチャ・プログラミング言語・フレームワークを大調査！〔2019年始版〕

https://employment.en-japan.com/engineerhub/entry/2019/01/08/103000?fbclid=IwAR2JZZ0AaHKn7AgllV6xcbWu9dtx9m6EskFKao9dtc0TnSMtWUD6fQ-X27s