



vs



Actor-based Message-driven Runtime

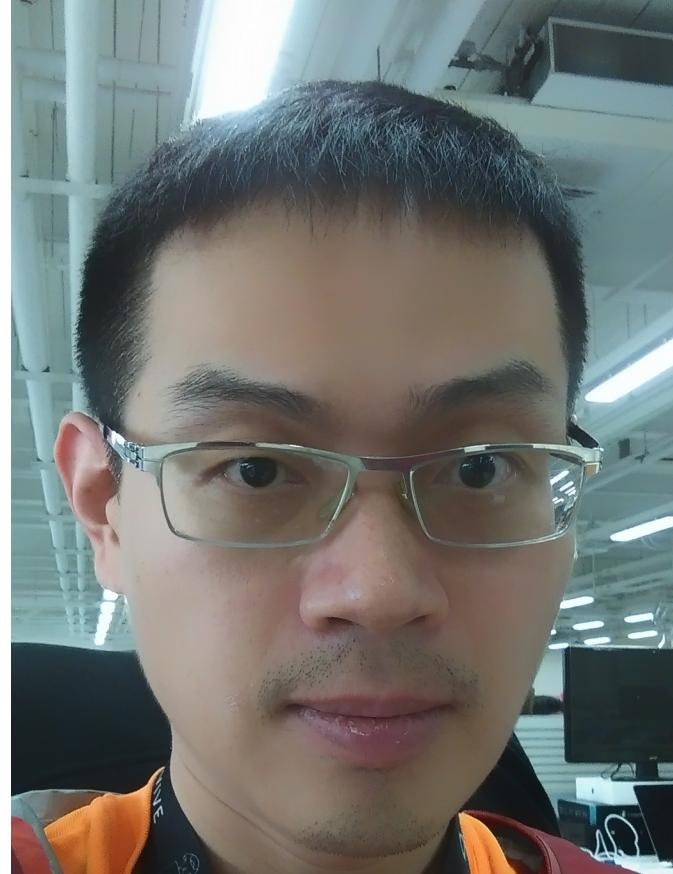
lightbend.com/akka

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

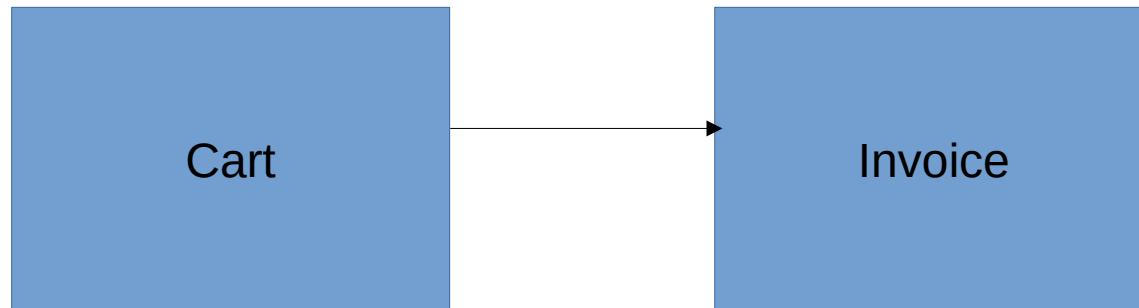
- Some essential concepts
- Finagle introduction
- Akka http introduction
- Comparison
- Discussion

Some essential concepts

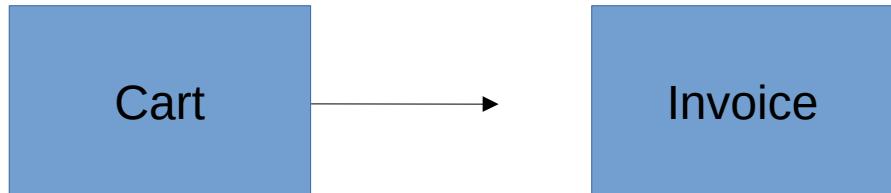
- Reactive programming
- Concurrency is not parallelism

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

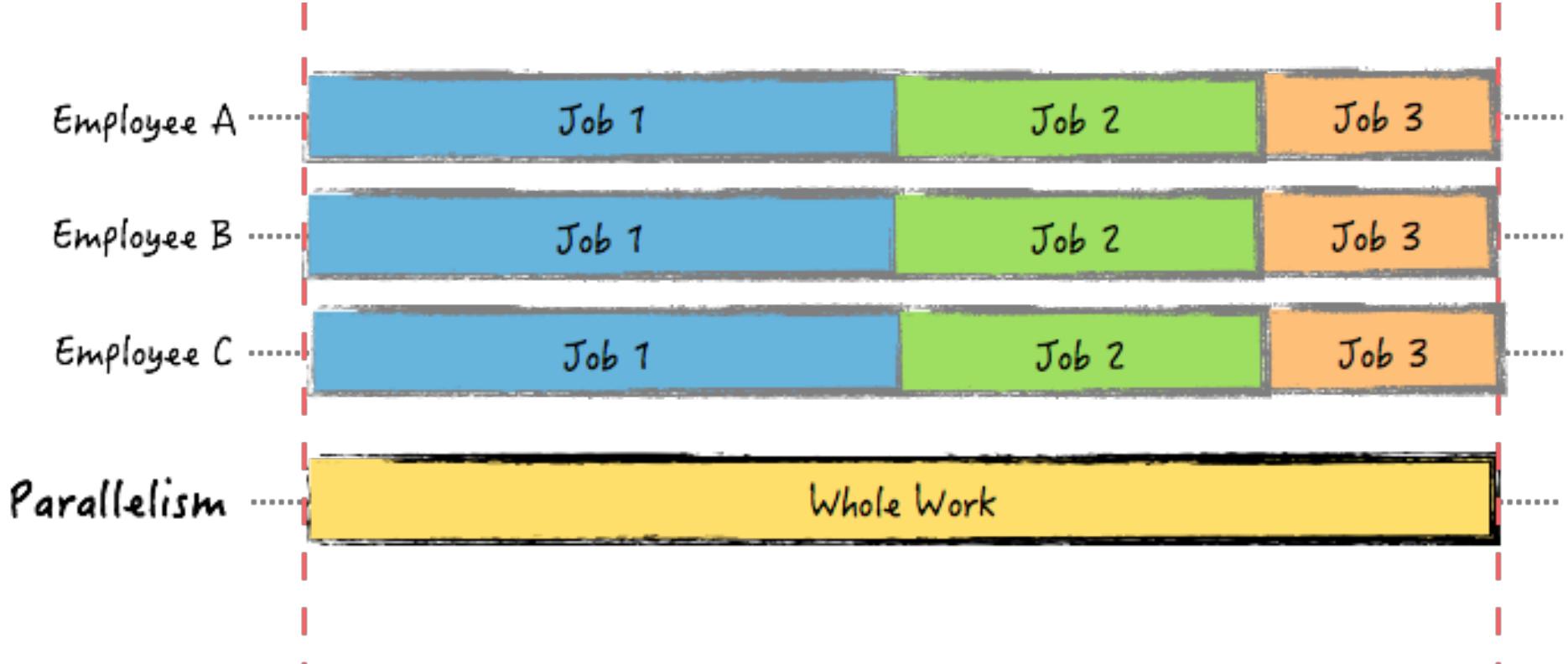


- Events, observation, and self-updates

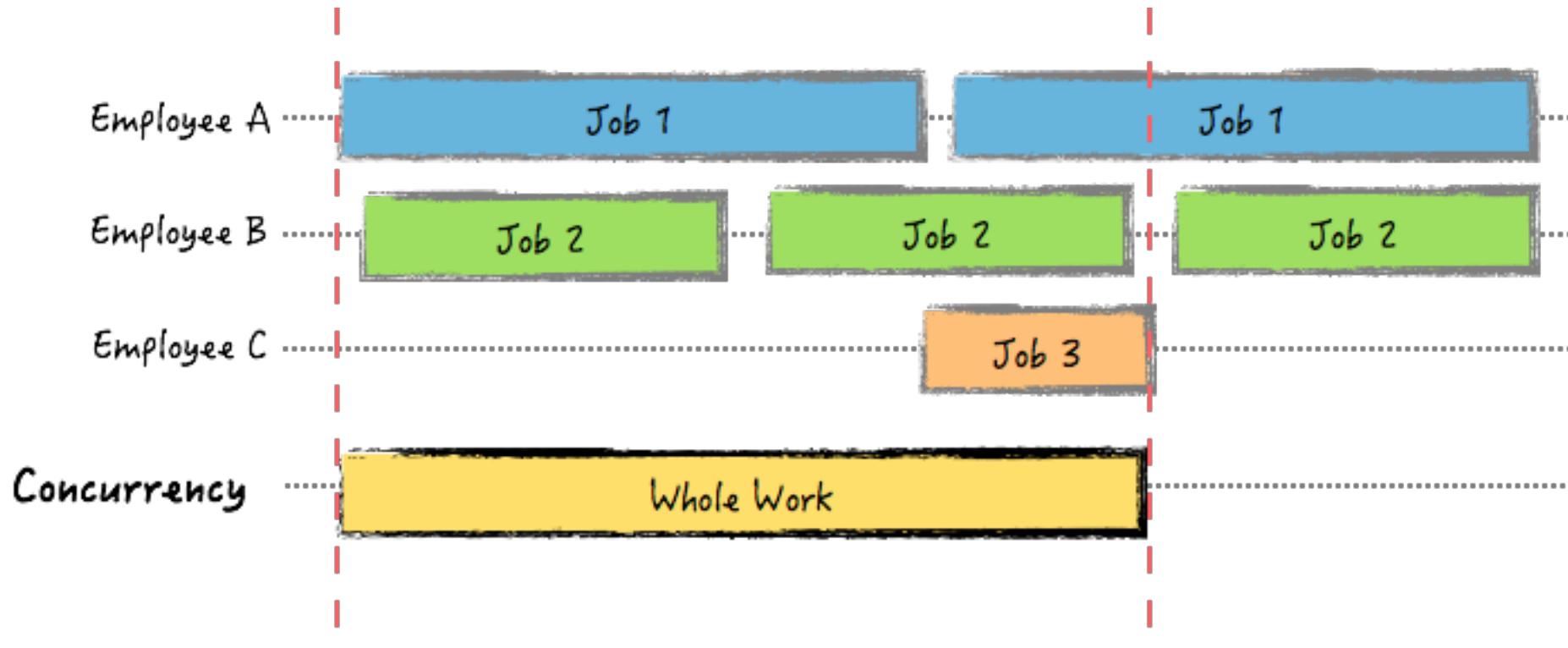
```
object Invoice {  
    ...  
    def receive = {  
        case Cart.onProductAdded(product) => {  
            ...  
            this.updateInvoicing(product)  
        }  
    }  
}
```

	Passive	Reactive
How does it work?	<i>Find usages</i>	<i>Look inside</i>
What does it affect?	<i>Look inside</i>	<i>Find usages</i>

Concurrency is not parallelism



Concurrency is not parallelism



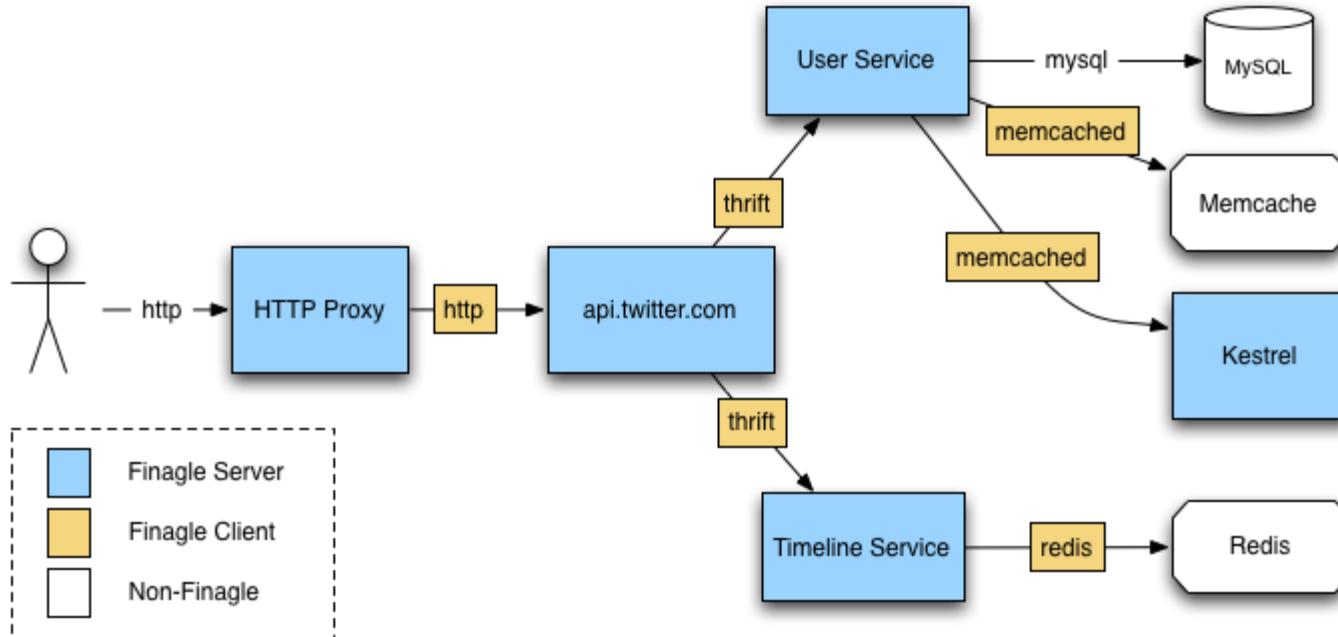
Finagle introduction



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

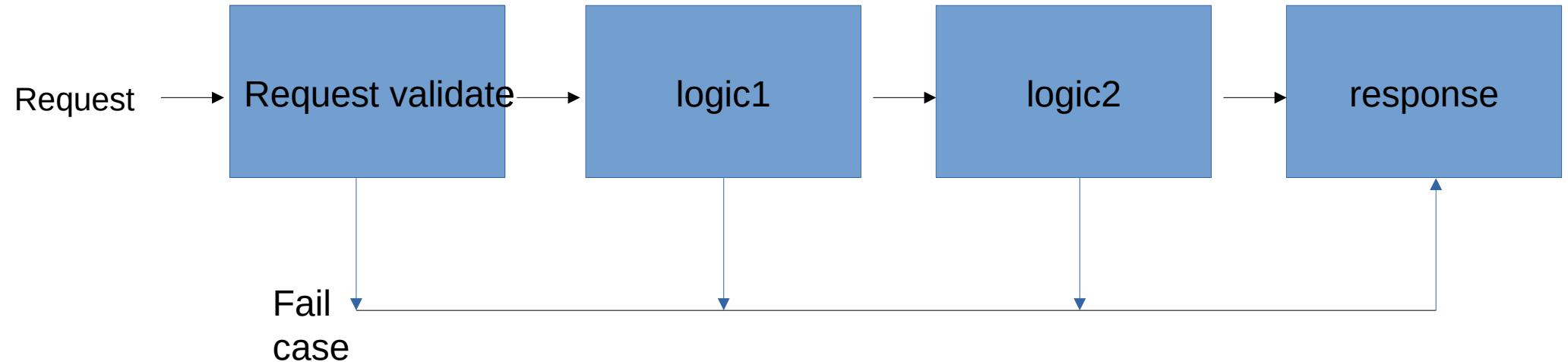


Finagle components



- 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

Summary

- Downstream Clients
- Listening Servers
- Metrics
- Misc
- Performance Profile
- Process Info
- Utilities

jvm/uptime: 32m 56s · jvm/thread/count: 24 · jvm/mem/current/used: 119.6MB · jvm/gc/msec: 340ms · Finagle Ver: 6.34.0

http

Success Rate

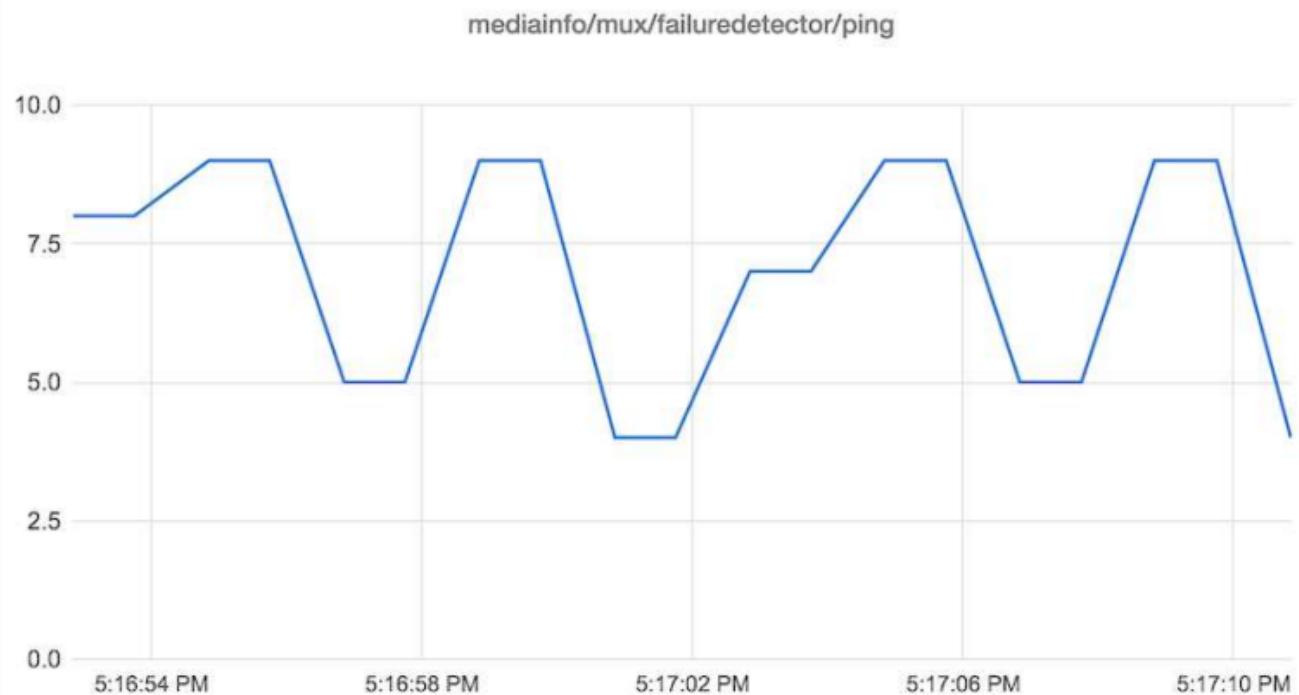
Load: 0
Failures: ...
Success: 0
Requests: 0

Least Performant Downstream Clients

zipkin-tracer	productService	userProfileService
Set{Inet(/192.168.99.100:9410,Map{})}	127.0.0.1:3001,127.0.0.1:3002,127.0.0.1:3003	127.0.0.1:4001,127.0.0.1:4002,127.0.0.1:4003
SUCCESS RATE	SUCCESS RATE	SUCCESS RATE
100.00%	100.00%	100.00%

Finalge admin UI(cont.)

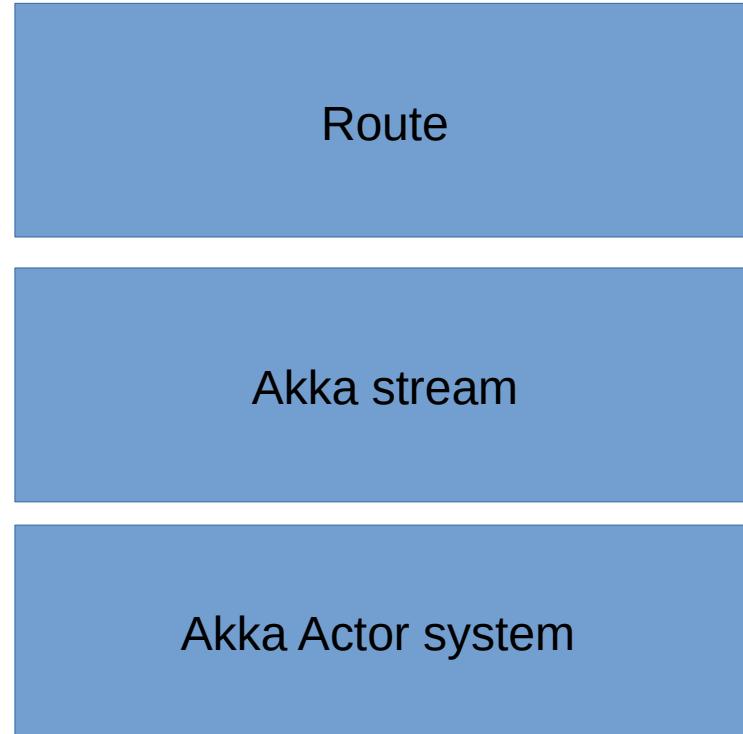
```
mediainfo/loadbalancer/size  
mediainfo/mux/current_lease_ms  
mediainfo/mux/drained  
mediainfo/mux/draining  
mediainfo/mux/failuredetector/close  
mediainfo/mux/failuredetector/failures  
mediainfo/mux/failuredetector/failures/com.twitter.fin  
mediainfo/mux/failuredetector/failures/com.twitter.fin  
mediainfo/mux/failuredetector/failures/com.twitter.util  
mediainfo/mux/failuredetector/marketed_busy  
mediainfo/mux/failuredetector/ping  
mediainfo/mux/failuredetector/revivals  
mediainfo/mux/leased  
mediainfo/namer/dtabcache/evicts  
mediainfo/namer/dtabcache/idle
```



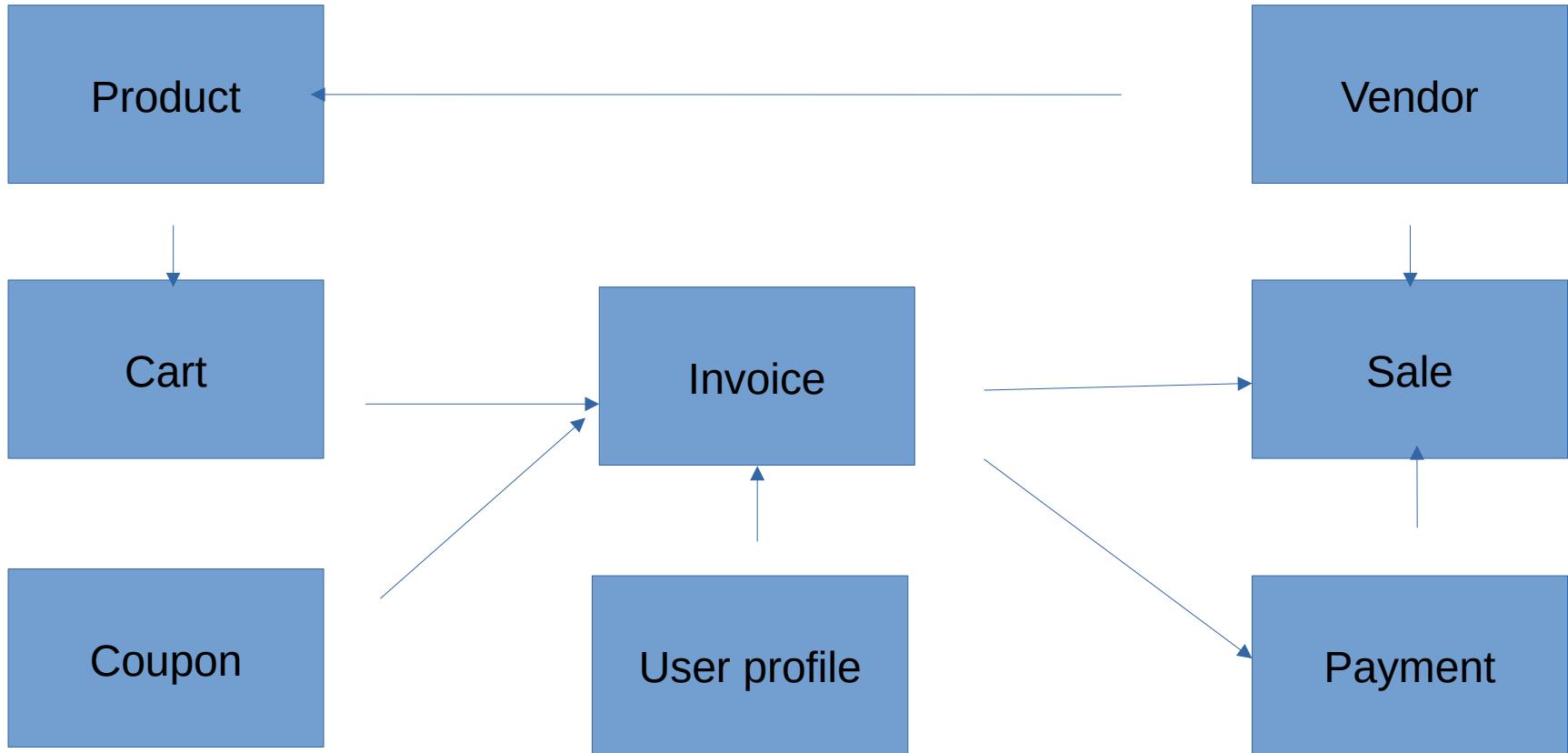
Akka http

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



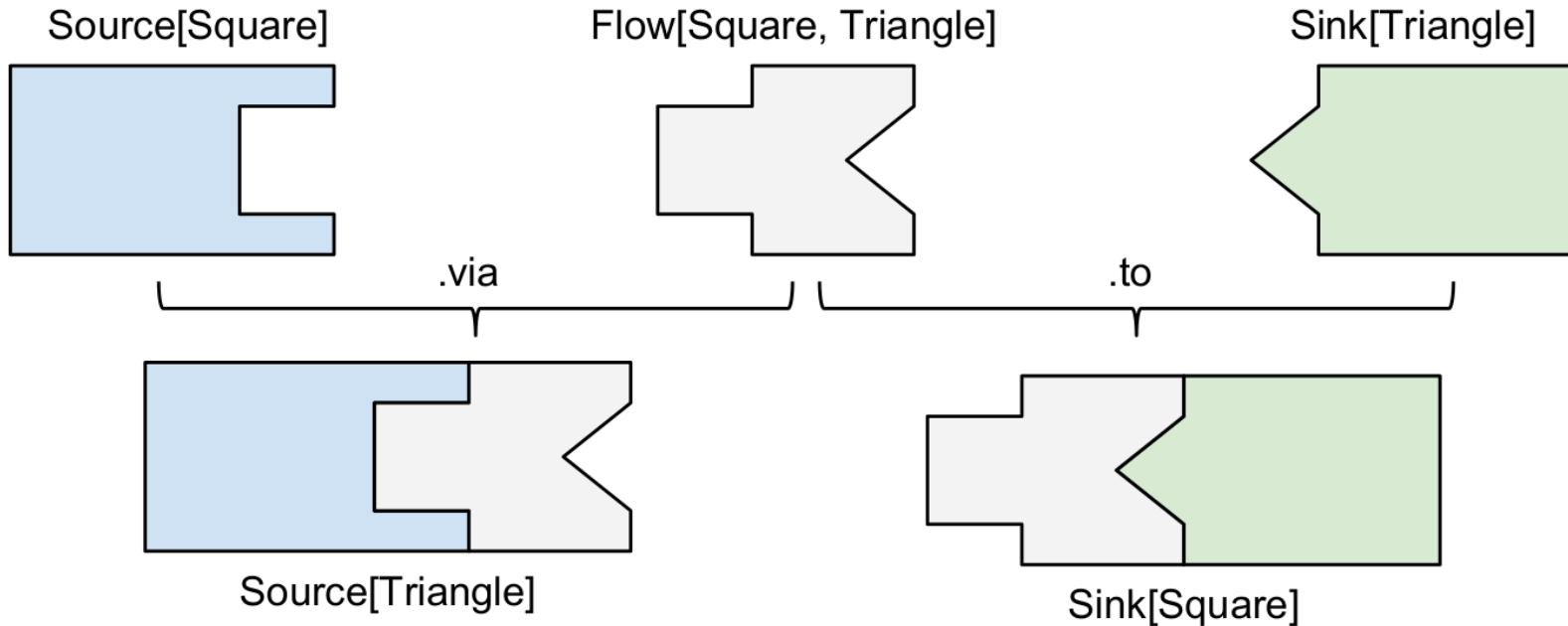
Akka actor example



Actor features

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

Akka stream



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		

Best JSON responses per second, Dell servers at ServerCentral (45 tests)

Rnk	Framework	Best performance (higher is better)		Errors	Cls	Lng	Plt	FE	Aos	IA
1	colossus	505,321	100.0% (92.9%)	0	Mcr	Sca	Akk	Non	Lin	Rea
2	gemini	492,544	97.5% (90.5%)	0	Ful	Jav	Svt	Res	Lin	Rea
3	netty	487,126	96.4% (89.5%)	0	Plt	Jav	Nty	Non	Lin	Rea
4	vertx-web	472,289	93.5% (86.8%)	0	Mcr	Jav	vtx	Non	Lin	Rea
5	vertx	465,523	92.1% (85.6%)	0	Plt	Jav	ver	Non	Lin	Rea
6	rapidoid	443,720	87.8% (81.6%)	0	Plt	Jav	Rap	Non	Lin	Rea
7	undertow	416,380	82.4% (76.5%)	0	Plt	Jav	Utw	Non	Lin	Rea
8	jooby	391,043	77.4% (71.9%)	0	Ful	Jav	Nty	Non	Lin	Rea
9	servlet	375,472	74.3% (69.0%)	0	Plt	Jav	Svt	Res	Lin	Rea
10	finatra	372,338	73.7% (68.4%)	0	Mcr	Sca	Nty	Non	Lin	Rea
11	grizzly	338,599	67.0% (62.2%)	0	Mcr	Jav	Svt	Grz	Lin	Rea
12	ilhttp	336,212	66.5% (61.8%)	0	Plt	Jav	JLH	Non	Lin	Rea
13	finagle	335,698	66.4% (61.7%)	0	Mcr	Sca	Nty	Non	Lin	Rea
14	wicket	328,387	65.0% (60.4%)	0	Ful	Jav	Svt	Res	Lin	Rea
15	fintrospect	320,366	63.4% (58.9%)	0	Mcr	Sca	Nty	Non	Lin	Rea
16	finch	298,505	59.1% (54.9%)	0	Mcr	Sca	Nty	Non	Lin	Rea
17	spark	278,400	55.1% (51.2%)	0	Mcr	Jav	Svt	Jty	Lin	Rea
18	jetty-servlet	261,972	51.8% (48.2%)	0	Plt	Jav	Jty	Non	Lin	Rea
19	aleph	239,349	47.4% (44.0%)	0	Mcr	Clj	Nty	Non	Lin	Rea
20	grizzly-jersey	223,663	44.3% (41.1%)	0	Mcr	Jav	JAX	Grz	Lin	Rea
21	jetty	212,780	42.1% (39.1%)	0	Plt	Jav	Jty	Non	Lin	Rea
22	jawn	187,453	37.1% (34.5%)	0	Ful	Jav	Svt	Utw	Lin	Rea
23	activeweb-jackson	171,607	34.0% (31.5%)	0	Ful	Jav	Svt	Non	Lin	Rea
24	undertow-jersey-hikaricp	152,381	30.2% (28.0%)	0	Plt	Jav	JAX	Non	Lin	Rea
25	bayou	148,166	29.3% (27.2%)	0	Plt	Jav	Bay	Non	Lin	Rea
26	tapestry	137,726	27.3% (25.3%)	0	Ful	Jav	Svt	Res	Lin	Rea
27	play2-scala-reactivemongo	137,540	27.2% (25.3%)	0	Ful	Sca	Akk	Non	Lin	Rea
28	curacao	120,851	23.9% (22.2%)	1	Mcr	Jav	Svt	Non	Lin	Rea
29	play2-scala/slick	118,895	23.5% (21.9%)	0	Ful	Sca	Akk	Non	Lin	Rea
30	http4s	116,048	23.0% (21.3%)	0	Mcr	Sca	NIO	bla	Lin	Rea
31	restexpress	114,787	22.7% (21.1%)	0	Mcr	Jav	Nty	Non	Lin	Rea
32	akka-http	106,054	21.0% (19.5%)	0	Mcr	Sca	Akk	Non	Lin	Rea
33	compojure	104,622	20.7% (19.2%)	0	Mcr	Clj	Svt	Res	Lin	Rea
34	dropwizard	89,702	17.8% (16.5%)	0	Ful	Jav	JAX	Jty	Lin	Rea

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

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_IA
- 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=IwAR2JZZ0AaHKn7AgIIv6xcbWu9dtx9m6EskFKao9dtc0TnSMtWUD6fQ-X27s>