Introduction to Akka with Scala

Piotr Trzpil @ Scala45s

The plan

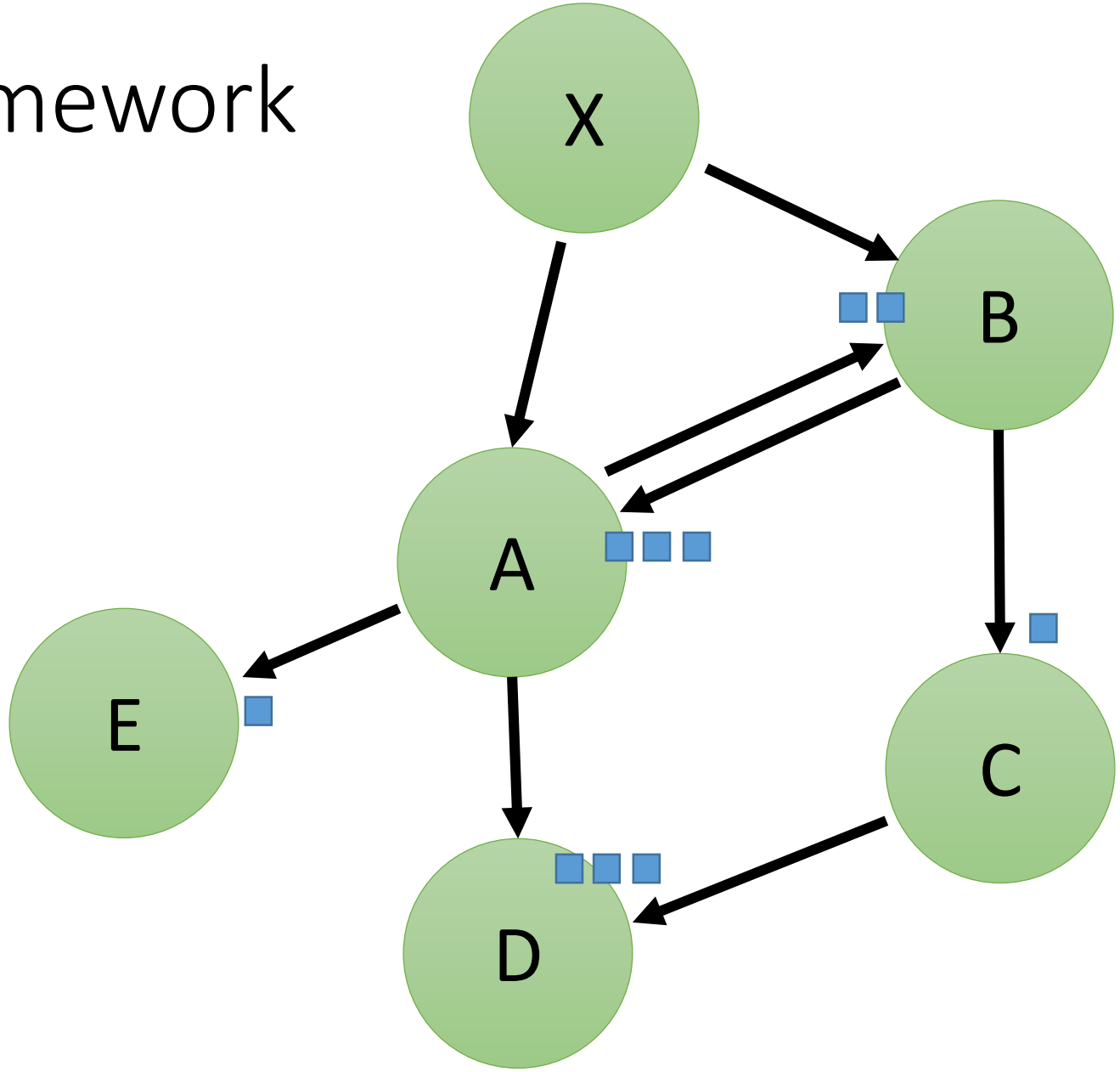
- Akka overview
- Part 1: Actor basics
- Part 2: Actor hierarchy
- A problem to solve: processing logs
- Part 3: Actor lifecycle

Akka – an actor framework

Actor system:

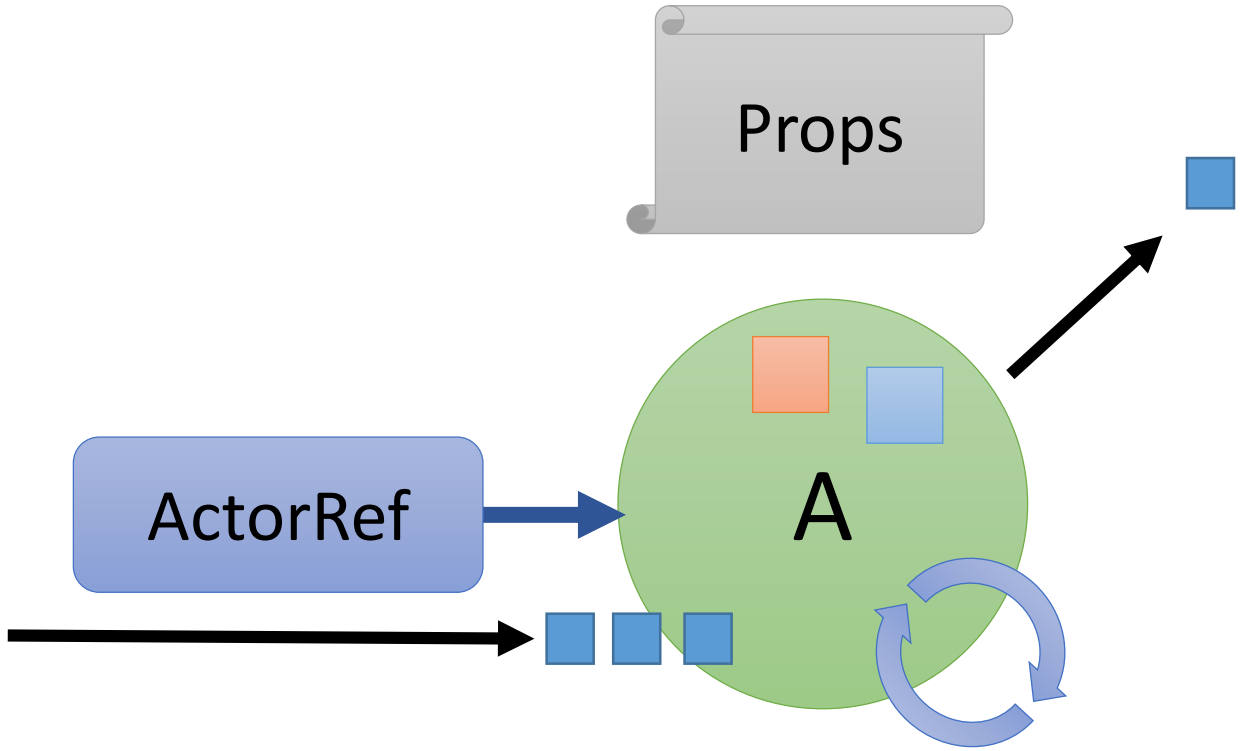
- An object model composed of actors
- Communication by asynchronous message passing
- Message delivery (in general) not guaranteed
- Java & Scala API

<u>Microservices in one app</u>



An actor

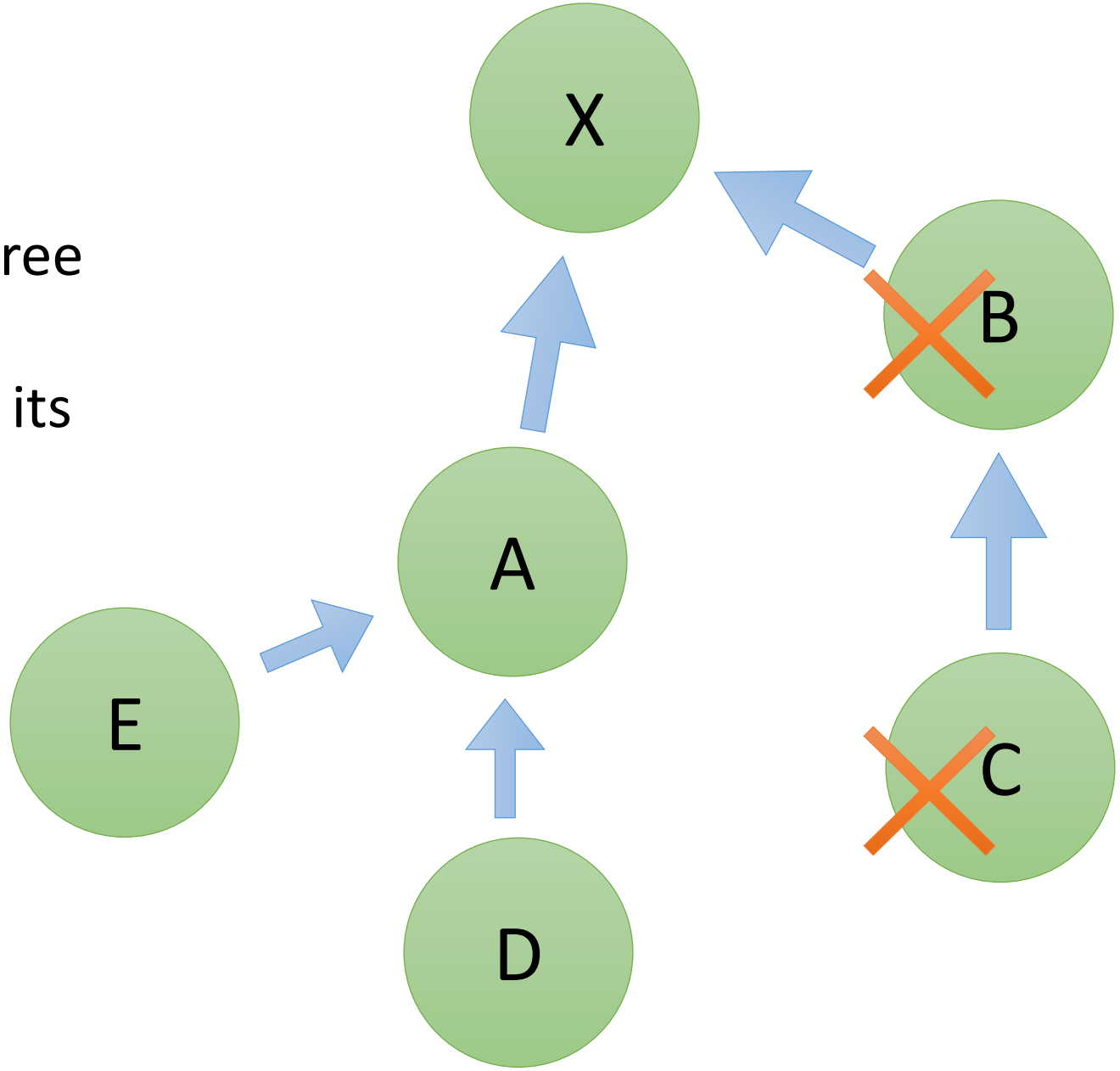
- Reacts to messages
- Is a ‘living’ object
- Has a:
  - Name
  - Location
  - Mutable state
  - Lifecycle
  - A mailbox
  - A parent
  - Optionally, children
  - Changeable behavior



Internally synchronous!

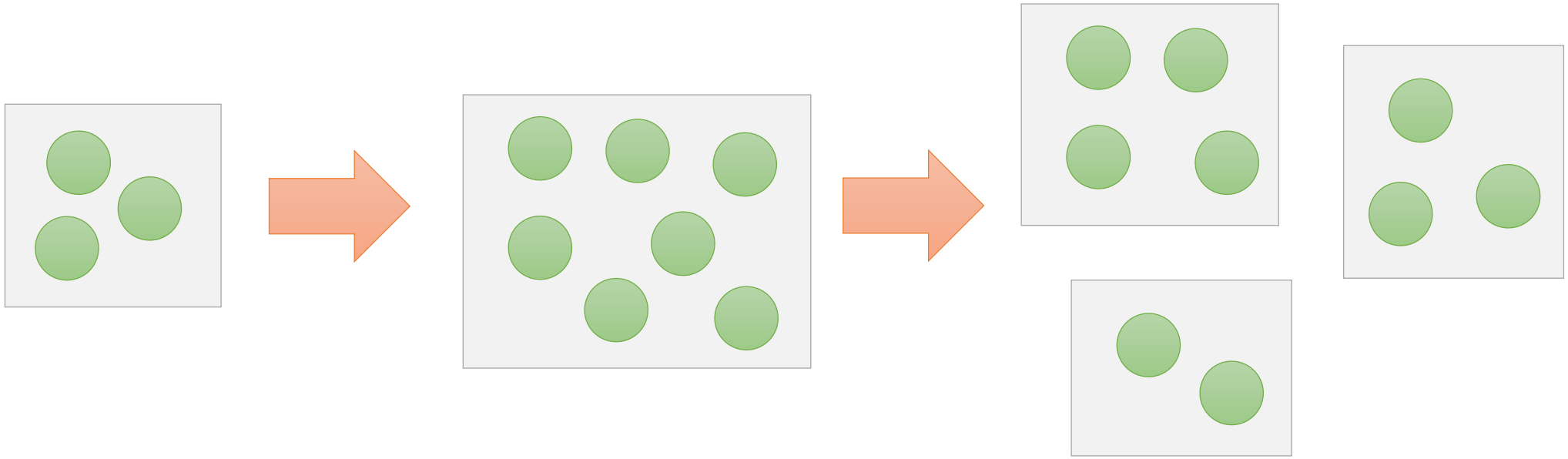
Actor hierarchy

- Actors form a supervision tree
- A child's life is governed by its parent
- Failure handling is independent from cause
- Failure is local
- Failure will happen…

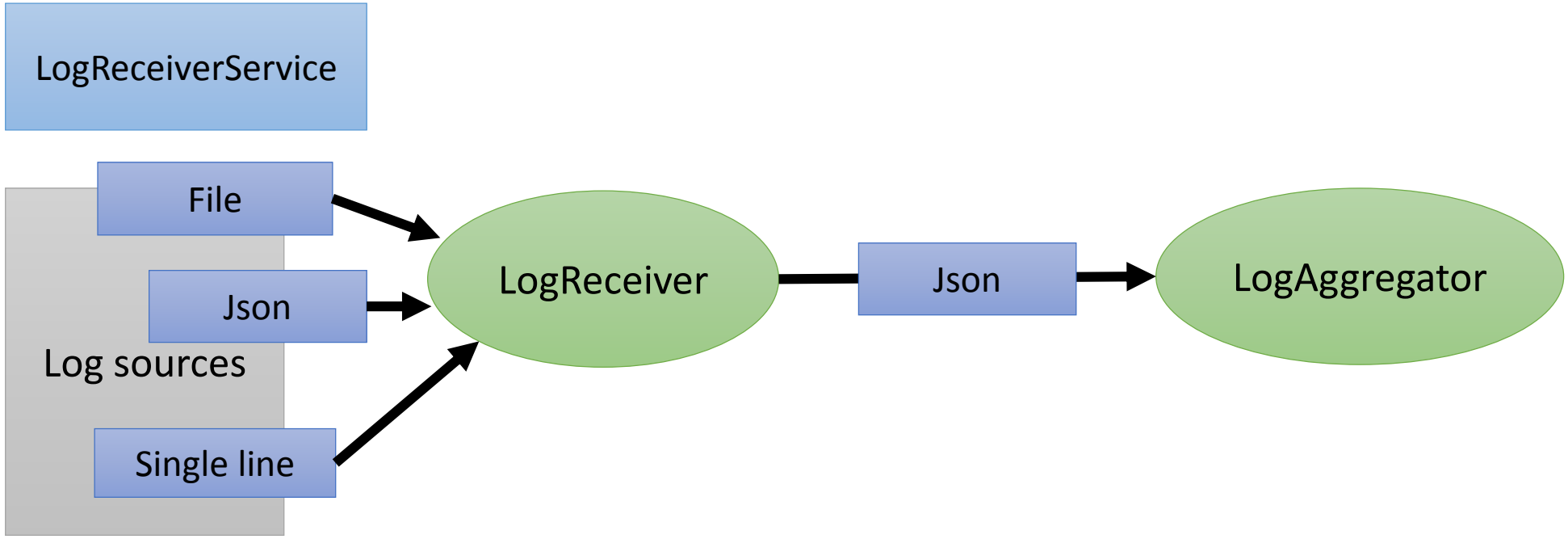


What does it all give?

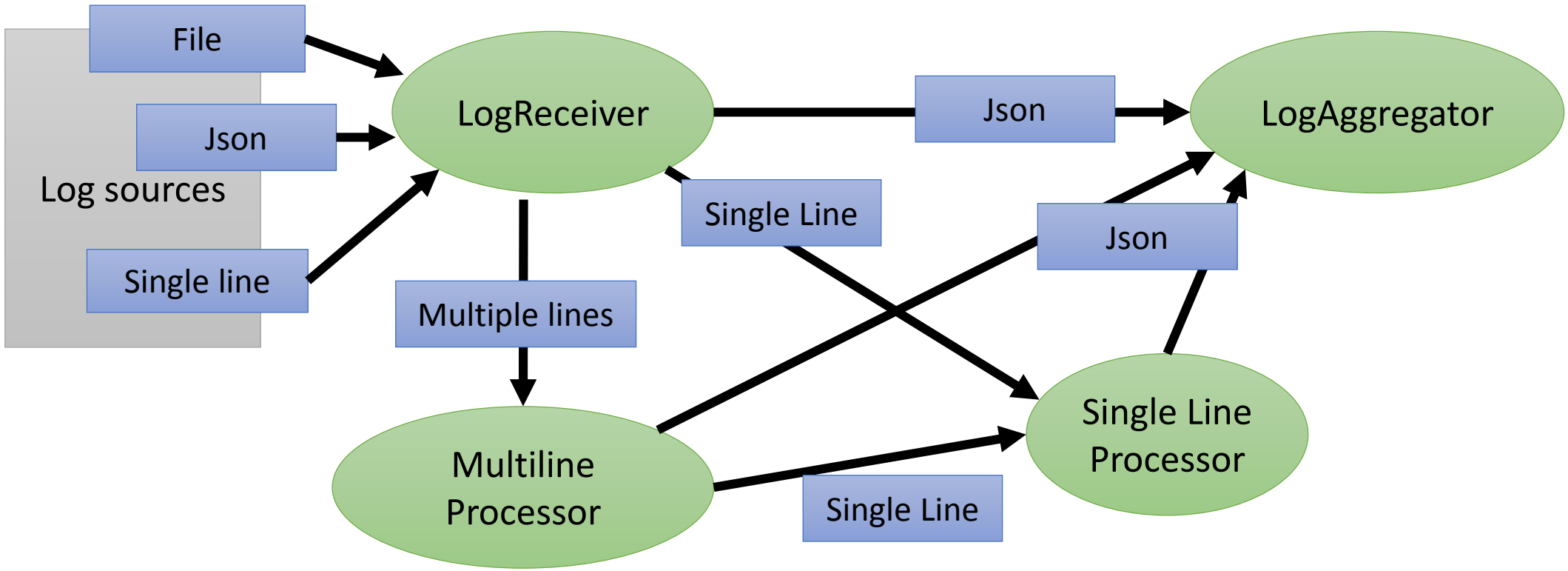
- Actors are independent processing cells
- It’s fast, non blocking
- It gives a horizontally scalable application architecture
- Composes well with regular objects



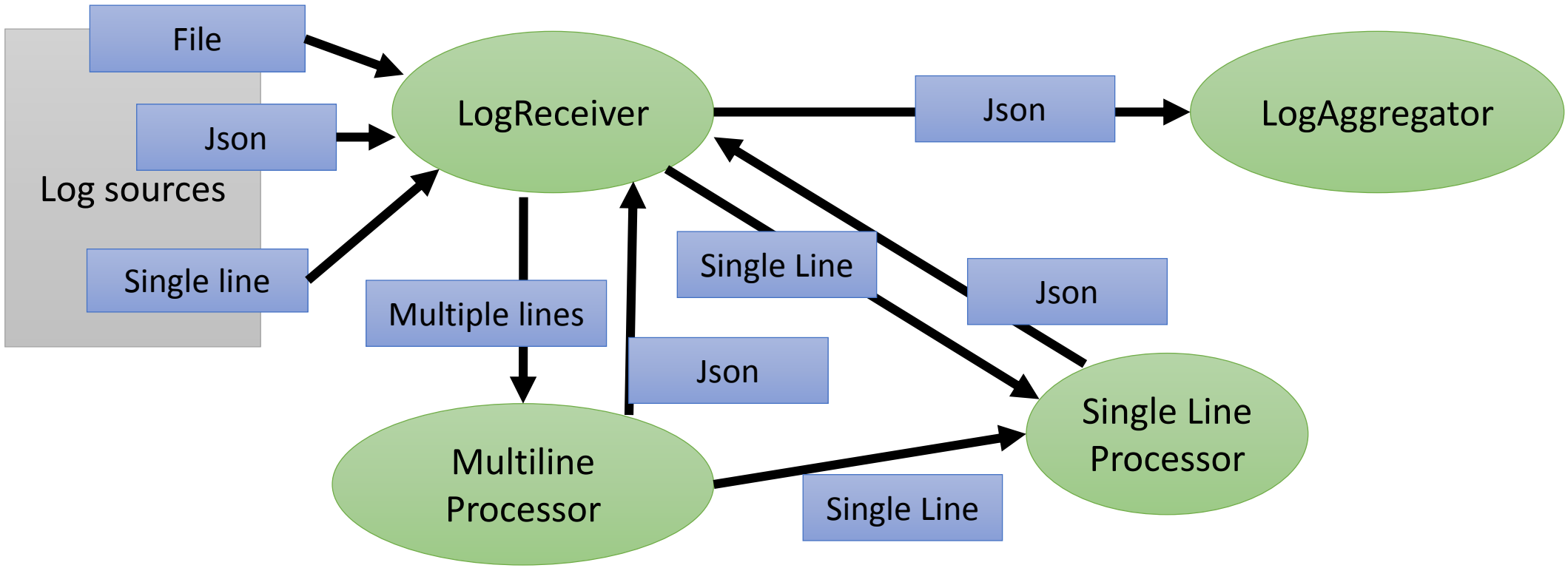
The problem to crunch – log processing



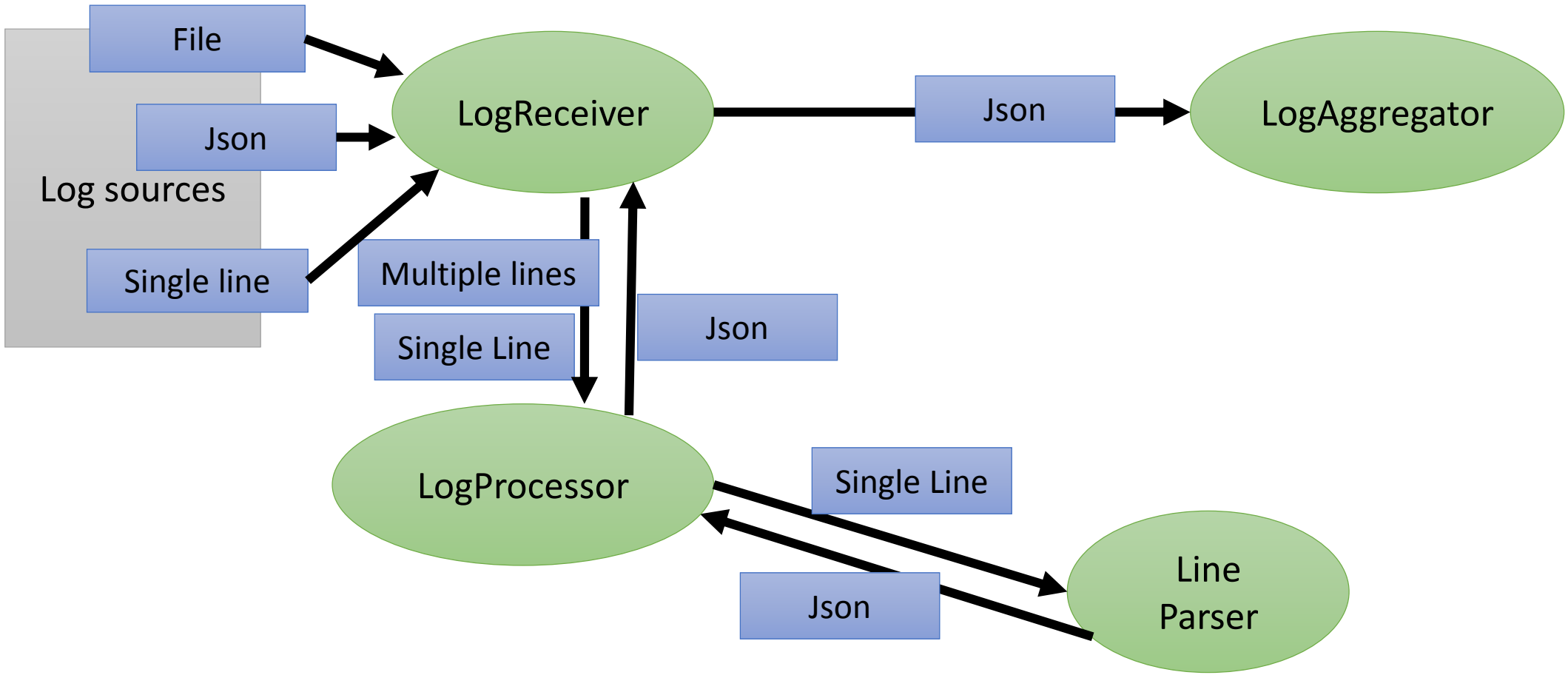
Possible Architecture



Another possibility

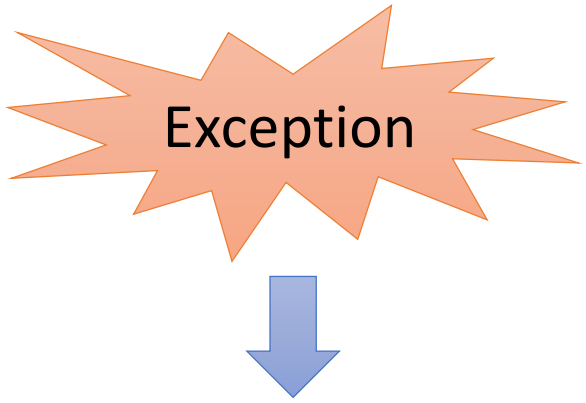


Another one – concepts changed

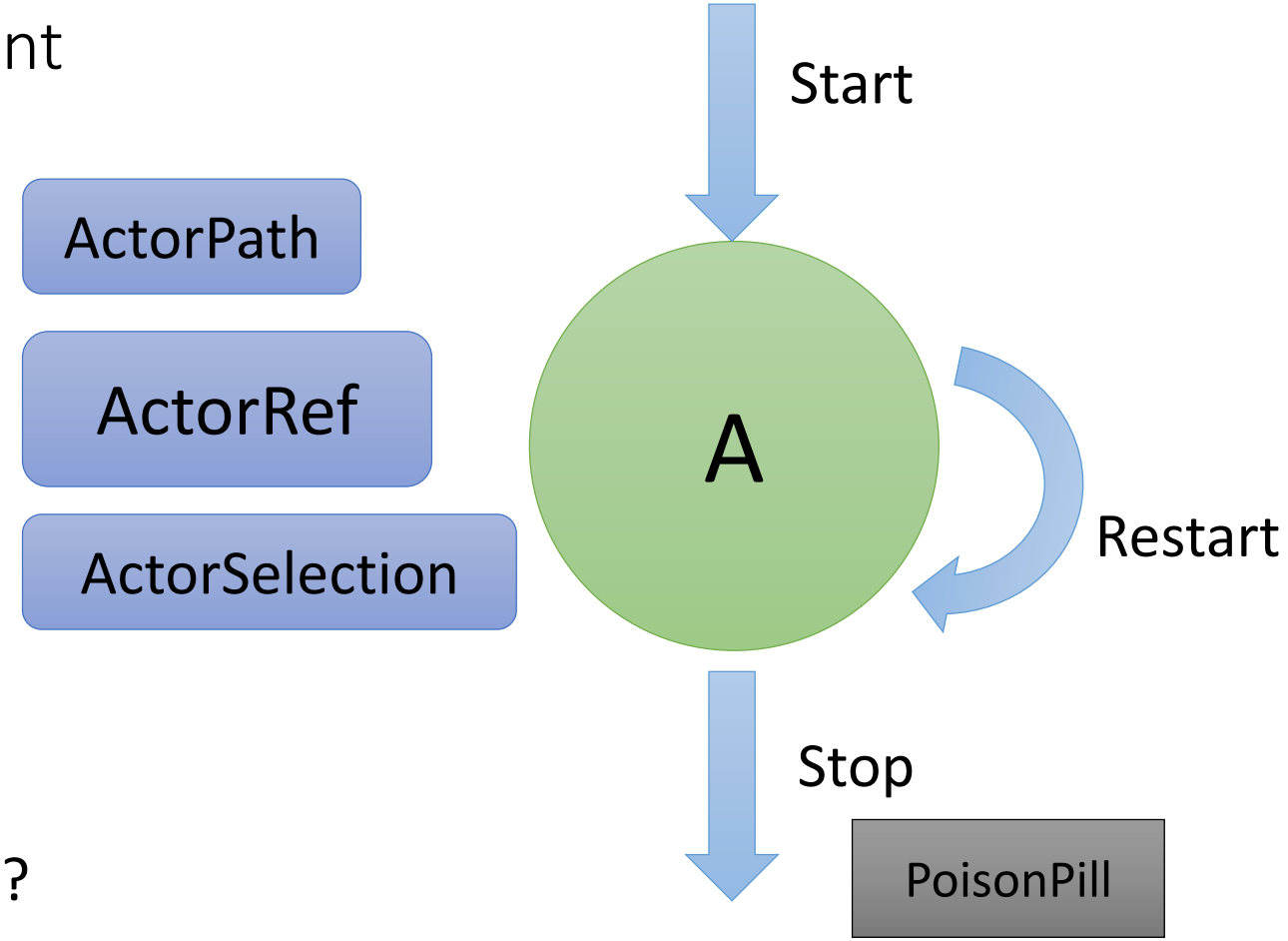


Actor lifecycle

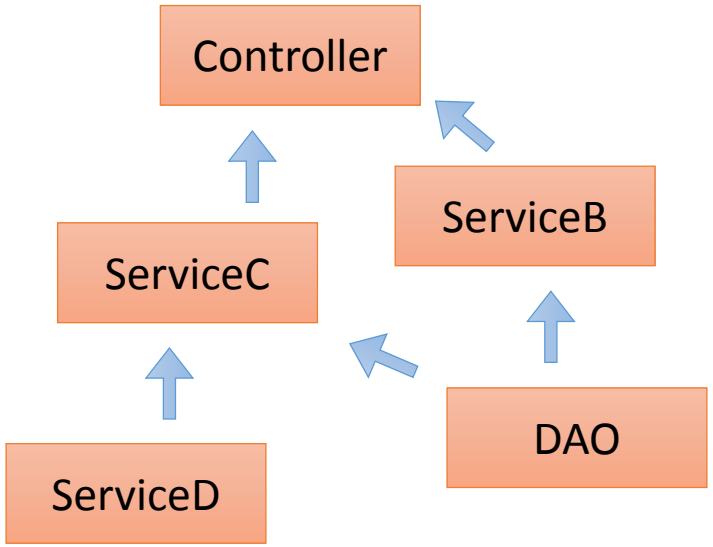
Supervised by actor's parent

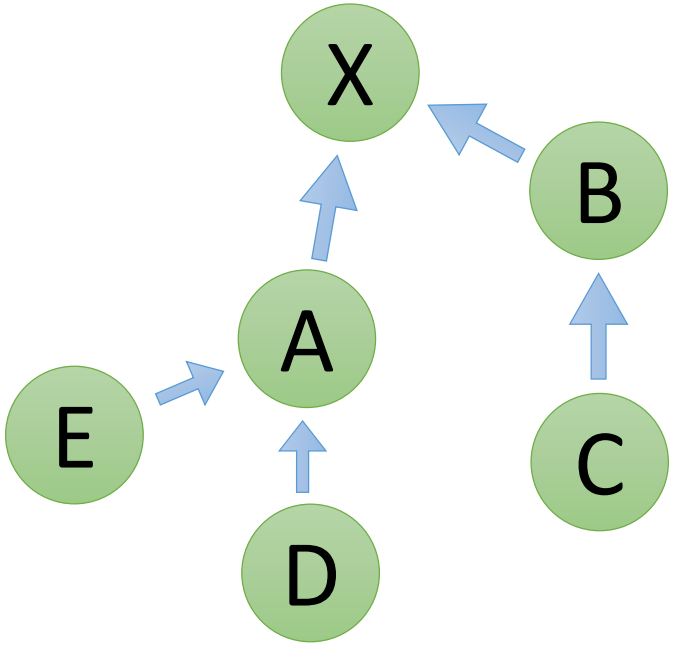


- Stop the child?
- Restart the child?
- Restart all children?
- Escalate (throw to parent)?



Old-school web application vs actor hierarchy



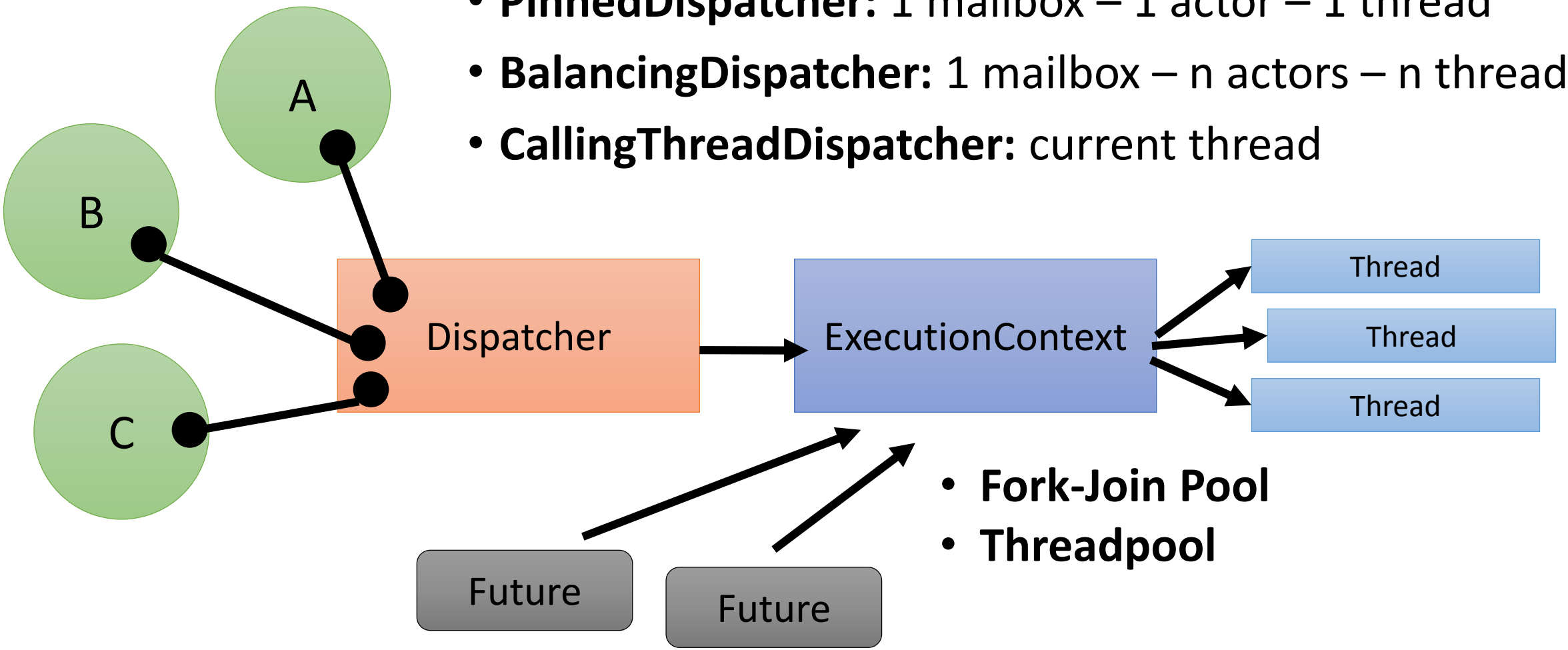


Stateless
Static

Stateful
Dynamic

Internals

- **Dispatcher:** n mailboxes – n actors – n threads
- **PinnedDispatcher:** 1 mailbox – 1 actor – 1 thread
- **BalancingDispatcher:** 1 mailbox – n actors – n threads
- **CallingThreadDispatcher:** current thread



- **Fork-Join Pool**
- **Threadpool**

Best practices

- Think of the hierarchy up-front
  - Avoid “singleton” root actors
  - Use mainly ActorRefs, not ActorSelection
- Always be prepared for failure
  - Set timeouts
- Monitor, log, measure everything
- Don’t optimize the performance (yet)

Summary

- Scalability at the core
- No threads
- No blocking
- If you can, delegate it
- Ensure single responsibility

Readings

- http://doc.akka.io/docs/akka/2.3.7/scala.html
- http://letitcrash.com/
- Akka Concurency
- Akka in Depth
- Akka in Action - MEAP
- Functional and Reactive Domain Modeling – MEAP
- **Code samples via Activator**