Actor Model

#SCBCN14

sponsored by

Ignasi Marimon-Clos (@ignasi35)

thanks!

@ignasi35

Questions!

@ignasi35

about you

about me

@ignasi35

1) problem solver, Garbage Collector, mostly scala, java 8, agile for tech teams

2) kayaker

3) under construction

4) all things JVM

The Pragmatic Programmers

Seven Concurrency Models in Seven Weeks

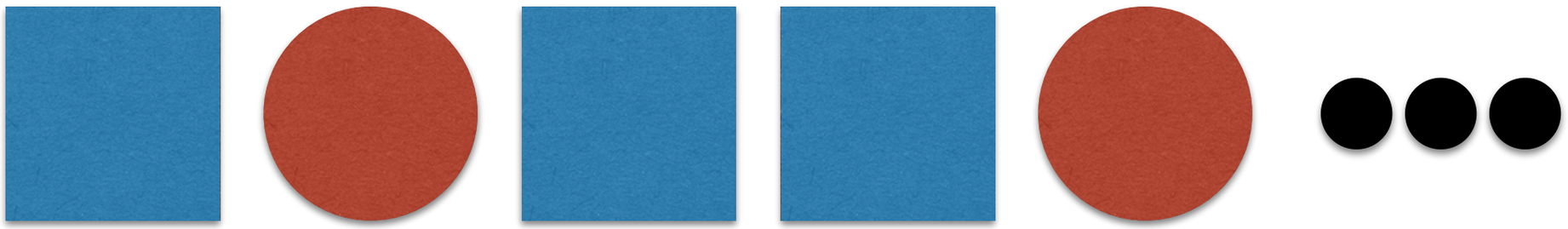
When Threads Unravel

Paul Butcher

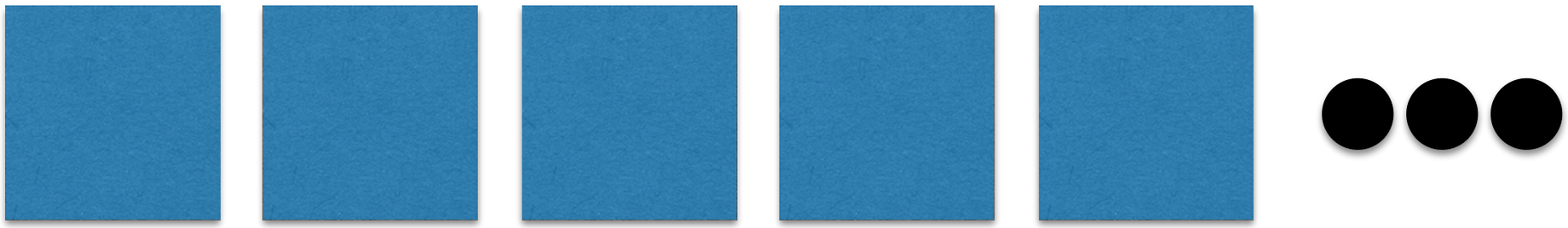
Series editor: *Bruce Tate*
Development editor: *Jacquelyn Carter*

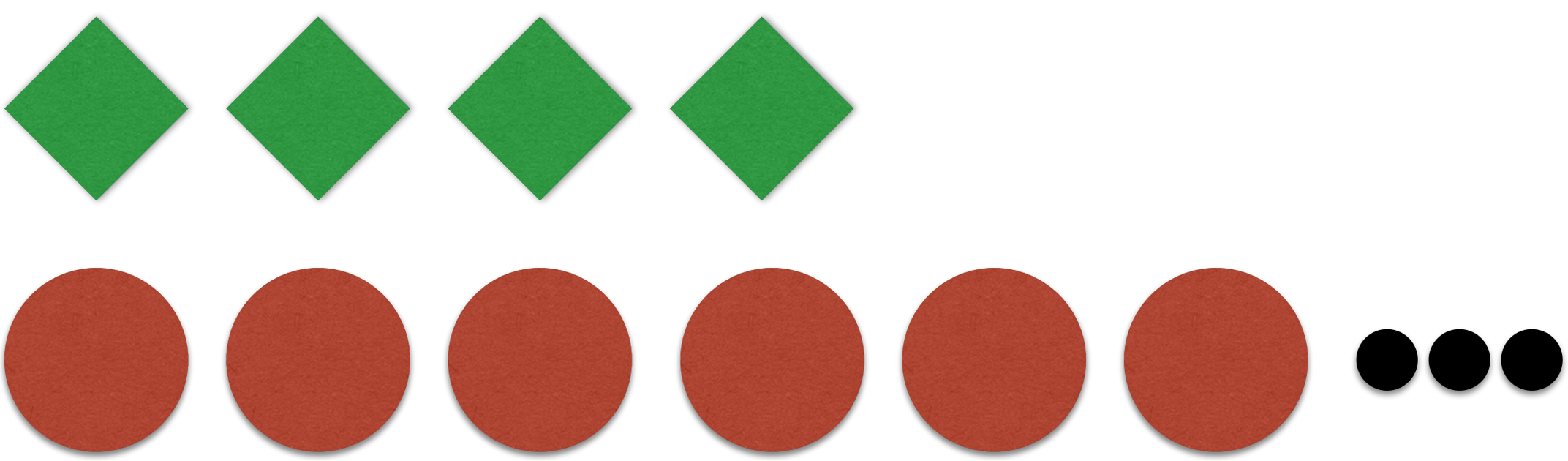
@ignasi35

Concurrency

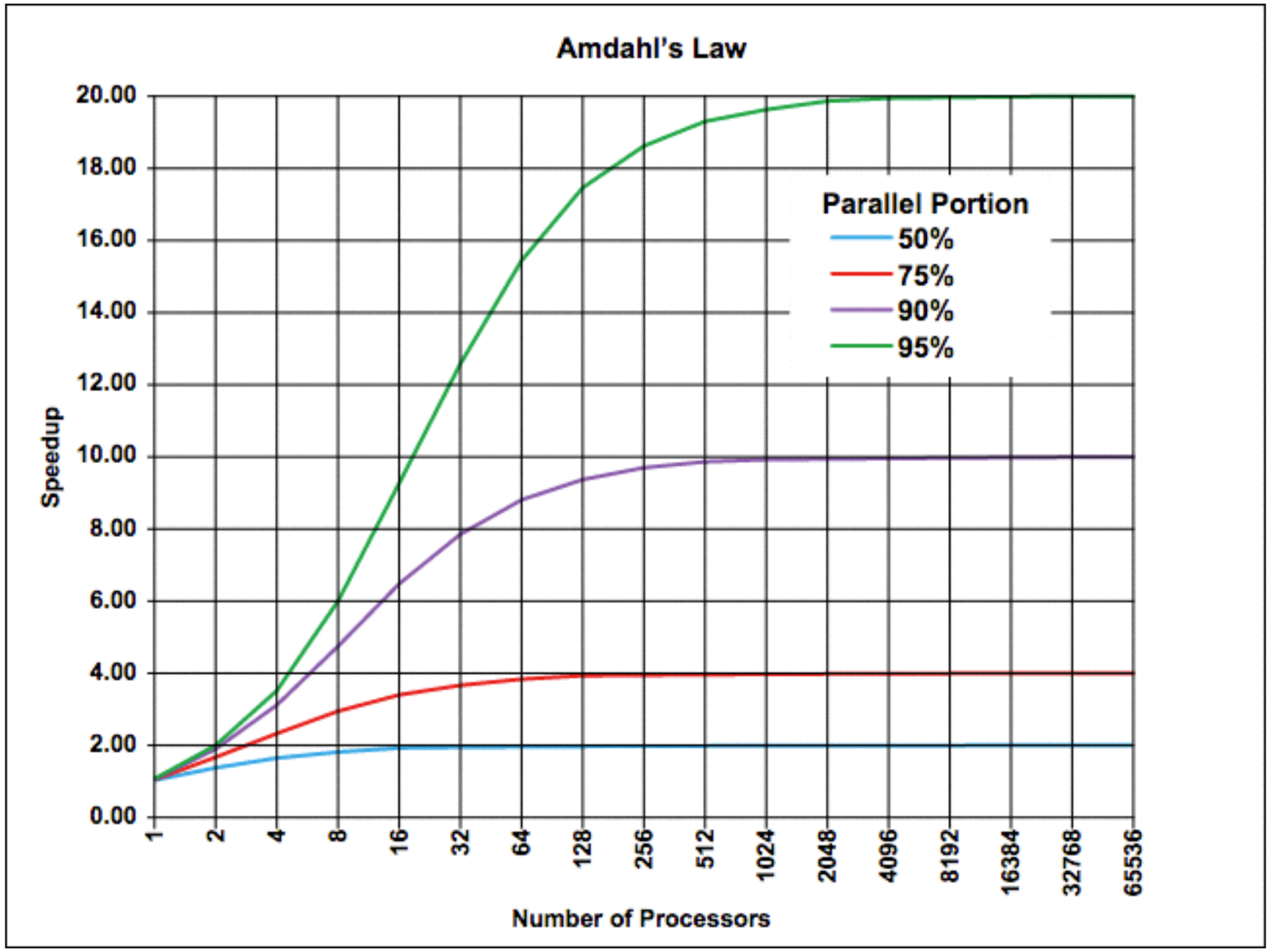


Parallelism



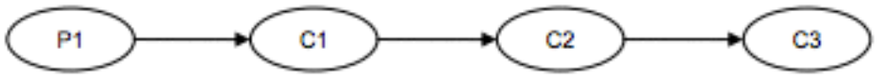


The Problem

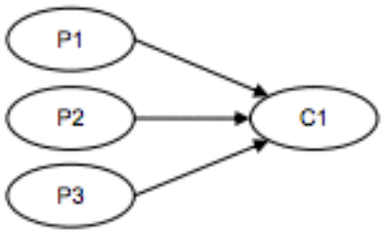

Amdahl's Law
Speedup
Number of Processors
Parallel Portion
50%
75%
90%
95%
20.00
18.00
16.00
14.00
12.00
10.00
8.00
6.00
4.00
2.00
0.00
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536

@ignasi35

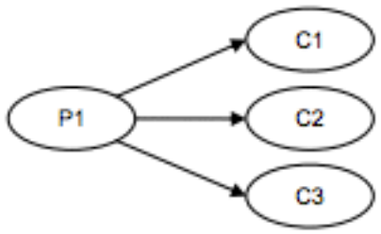
Unicast: 1P – 1C



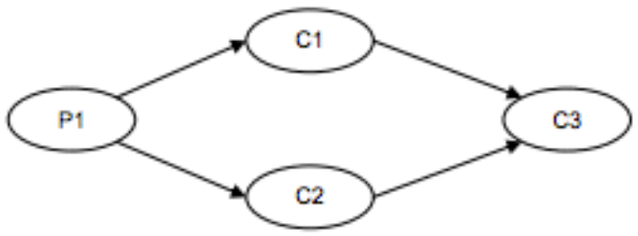
Three Step Pipeline: 1P – 3C



Sequencer: 3P – 1C



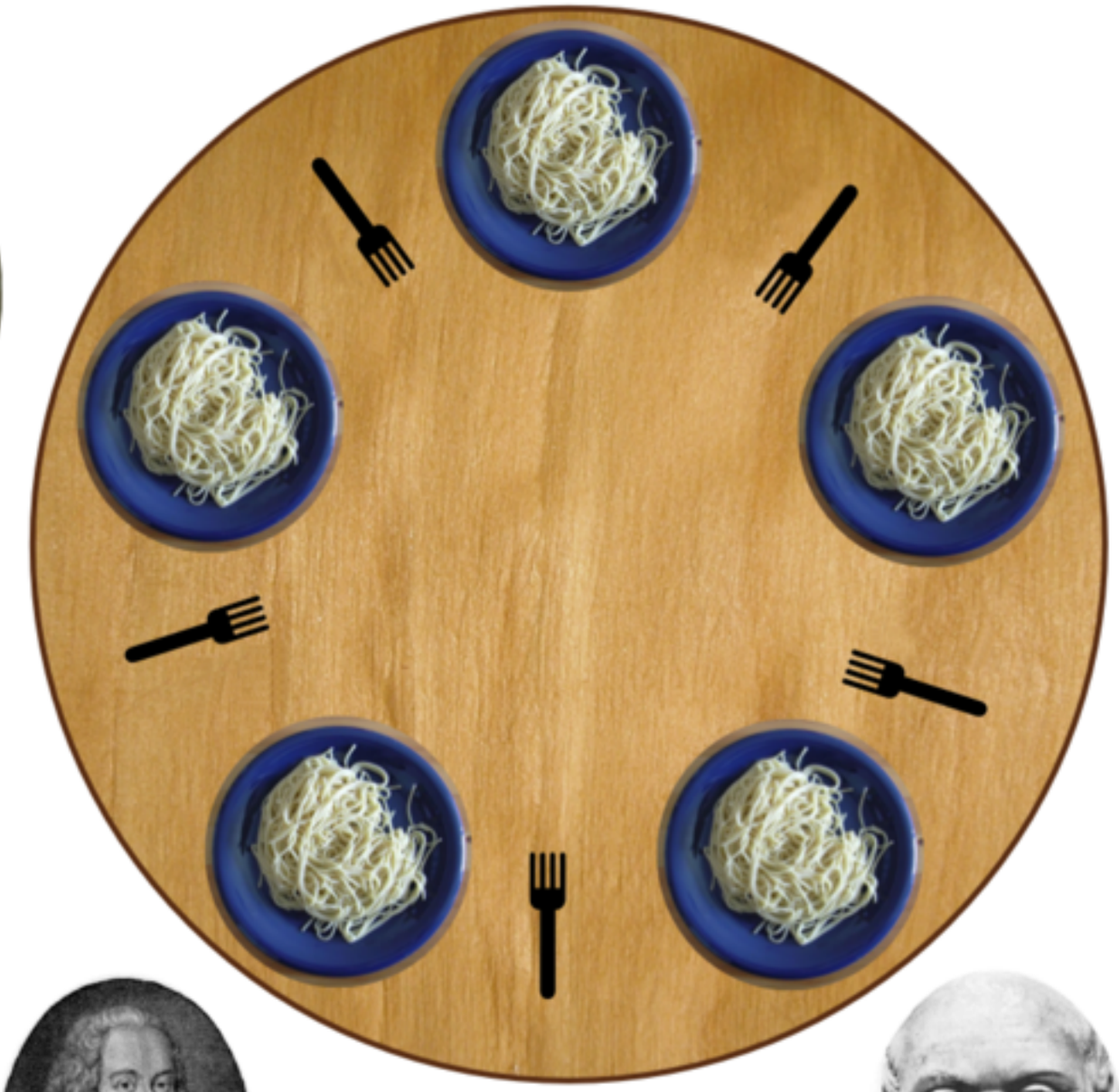
Multicast: 1P – 3C

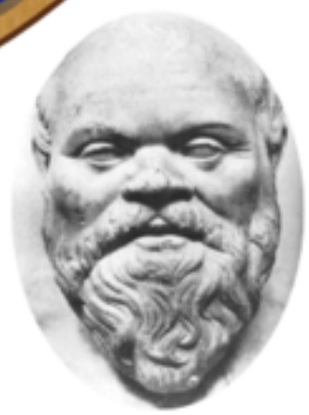


Diamond: 1P – 3C

@ignasi35

- CPU-bound or IO bound
- Latency or Thoughput
- memory footprint





@ignasi35

Threads and Locks

@ignasi35

Threads and Locks

- low level
- debugging
- ThreadPool

Threads and Locks

- threads are heavy
- locks expensive
- low level
- no recovery
- locking granularity
- visibility

Threads and Locks

Mutual Exclusion
Lock Ordering
Memory Model

@ignasi35


HD

Event Loop

@ignasi35

Event Loop

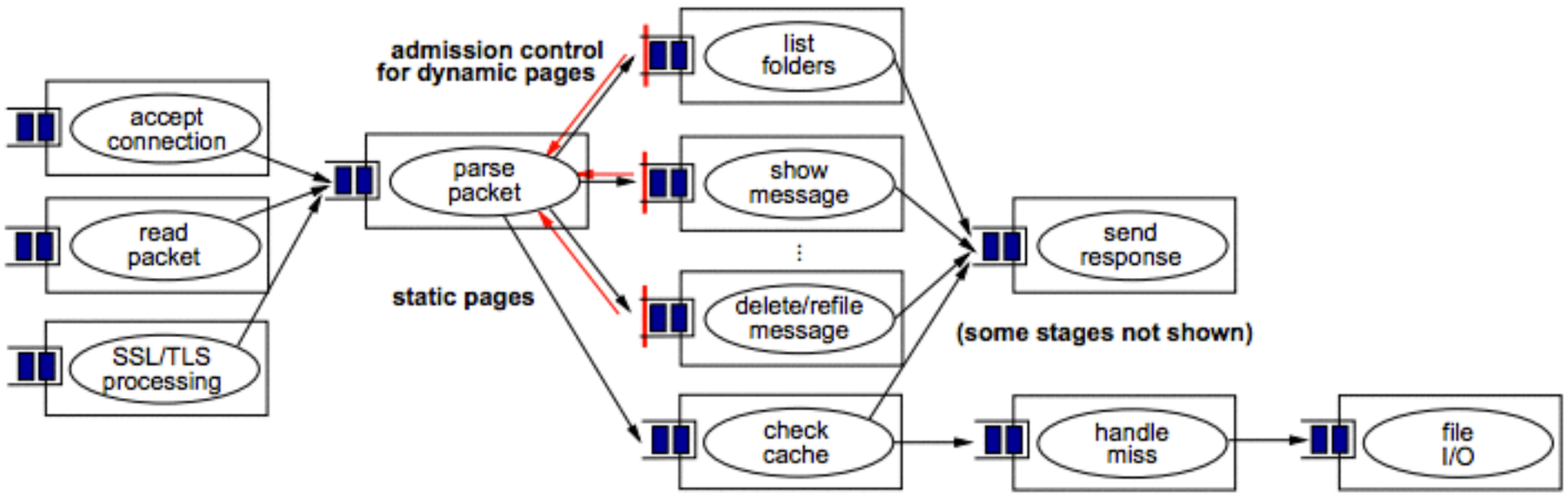
@ignasi35


filmax

SEDA

@ignasi35

SEDA

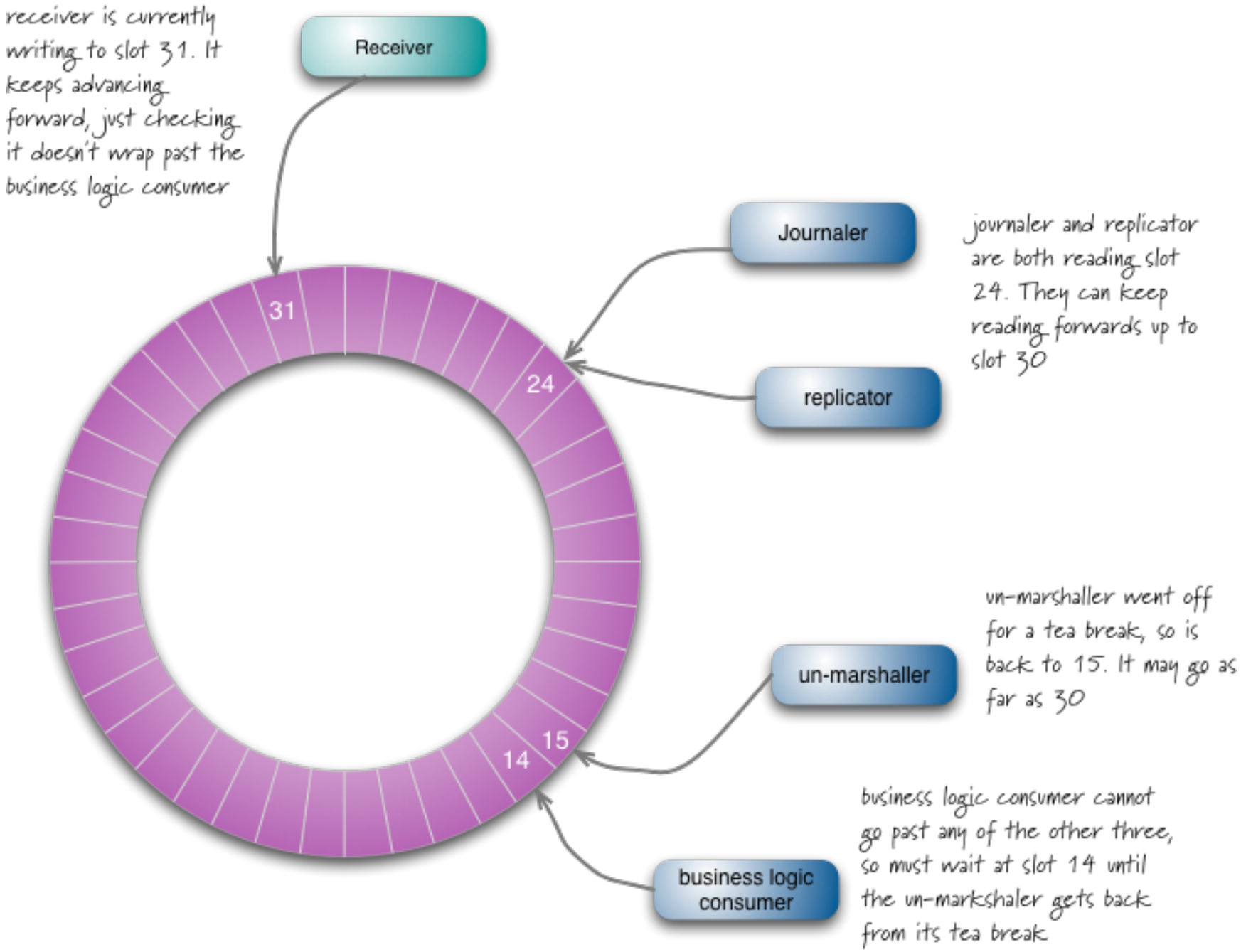


@ignasi35

Disruptor

@ignasi35

Disruptor



@ignasi35


ANCING ELK
1974

Functional Programming

@ignasi35

Functional Programming

Referential Transparency
Immutability

Persistent Datastructures

@ignasi35

Concurrency models

- Shared state
  - Threads and locks
  - Disruptor
  - FP
- Message passing
  - SEDA
  - Actors

WARNING!!! Very incomplete list!!!

Actor Model

@ignasi35

Actor Model

- OOP was always about passing messages
  - State encapsulation
- Actors run concurrently
- Actors *DO* send messages

What is ?

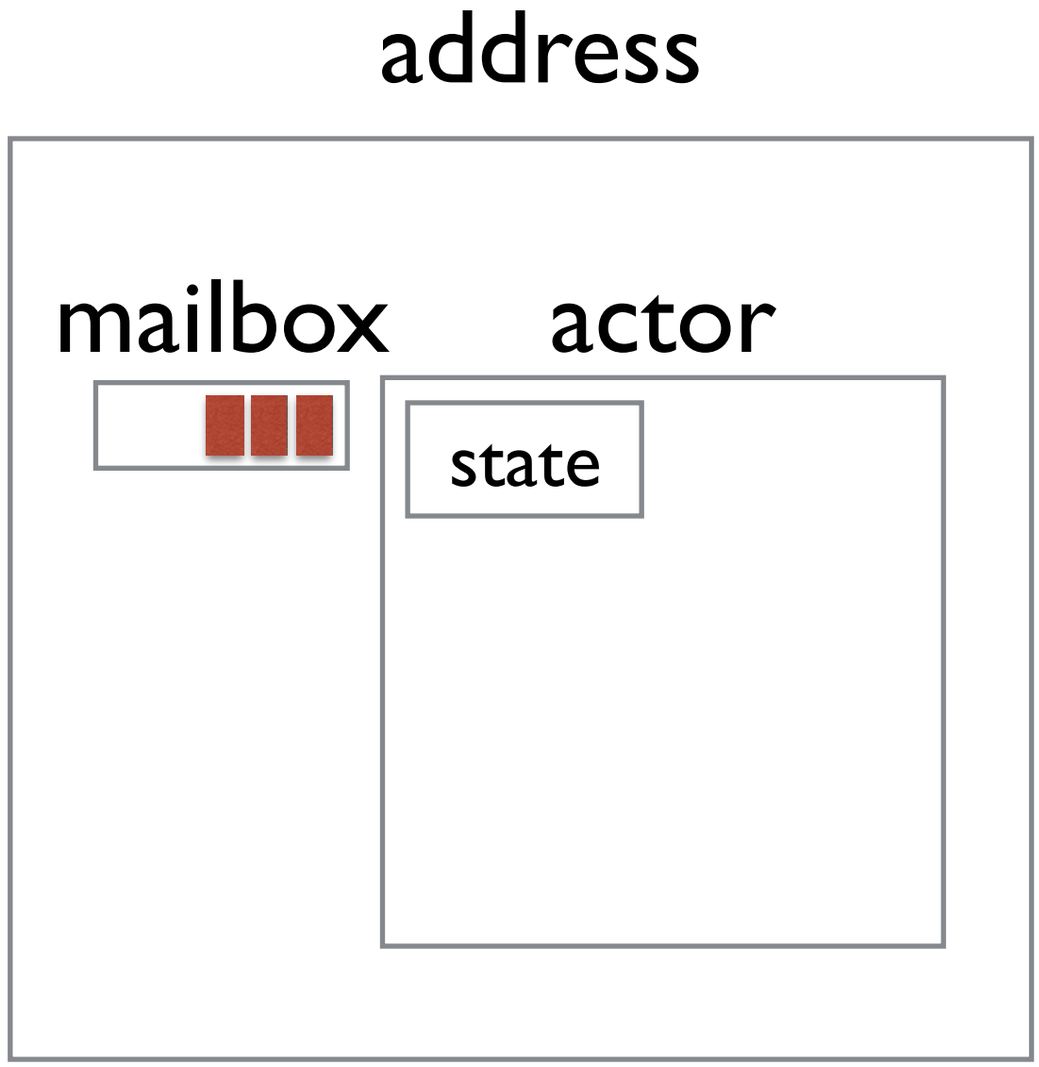
- fundamental unit of computation
- process
- store
- communicate

Axioms

- An Actor […], in response to a message it receives, can concurrently:
  - send messages to addresses of Actors that it has
  - create new Actors
  - designate how to handle the next message it receives

Higher abstraction

- everything runs on a ‘system’
- forget about locks, queues,
- programmer doesn’t know anything about where it’s running. It can be adapted at environment.


address
mailbox
actor
state

This is you now
(Only picture without actors)

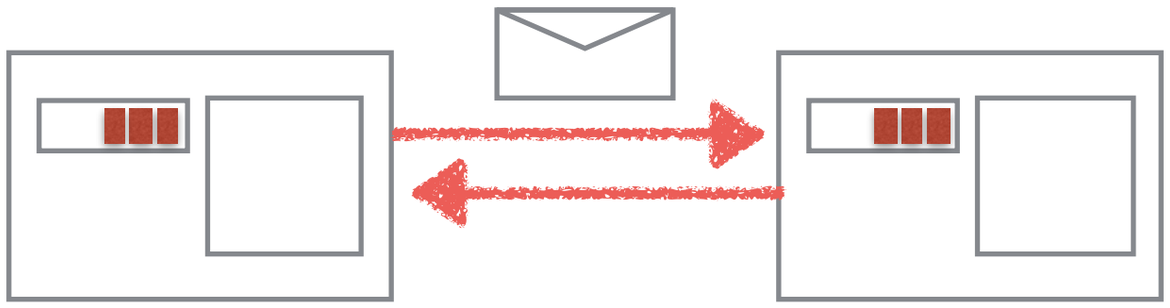
Actors/hello_actors/hello_actors.exs

```
def loop do
  receive do
    {:greet, name} -> IO.puts("Hello #{name}")
    {:praise, name} -> IO.puts("#{name}, you're amazing")
    {:celebrate, name, age} -> IO.puts("Here's to another #{age} years, #{name}")
  end
  loop
end
```

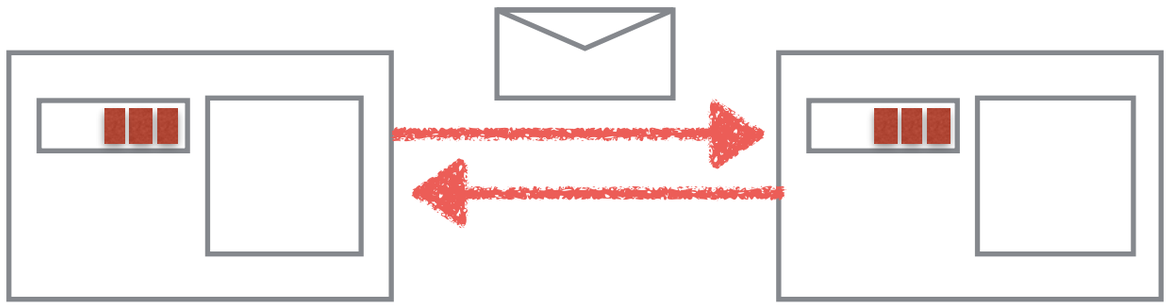
source: 7 concurrency models in 7 weeks

fire and forget

invoke

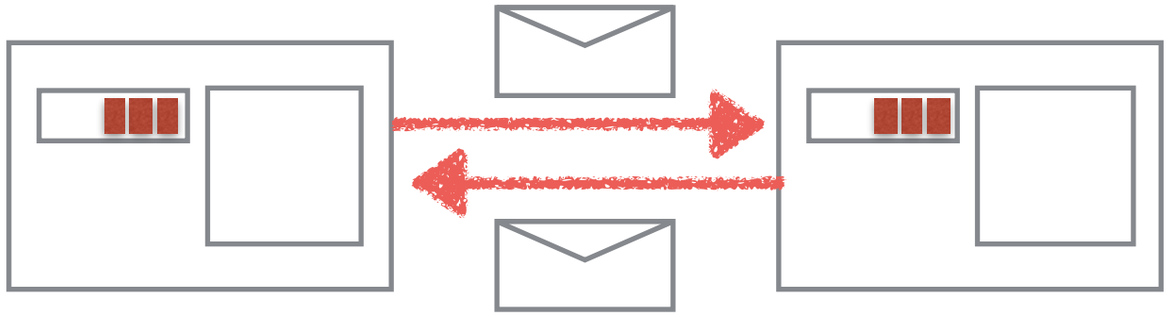


ask

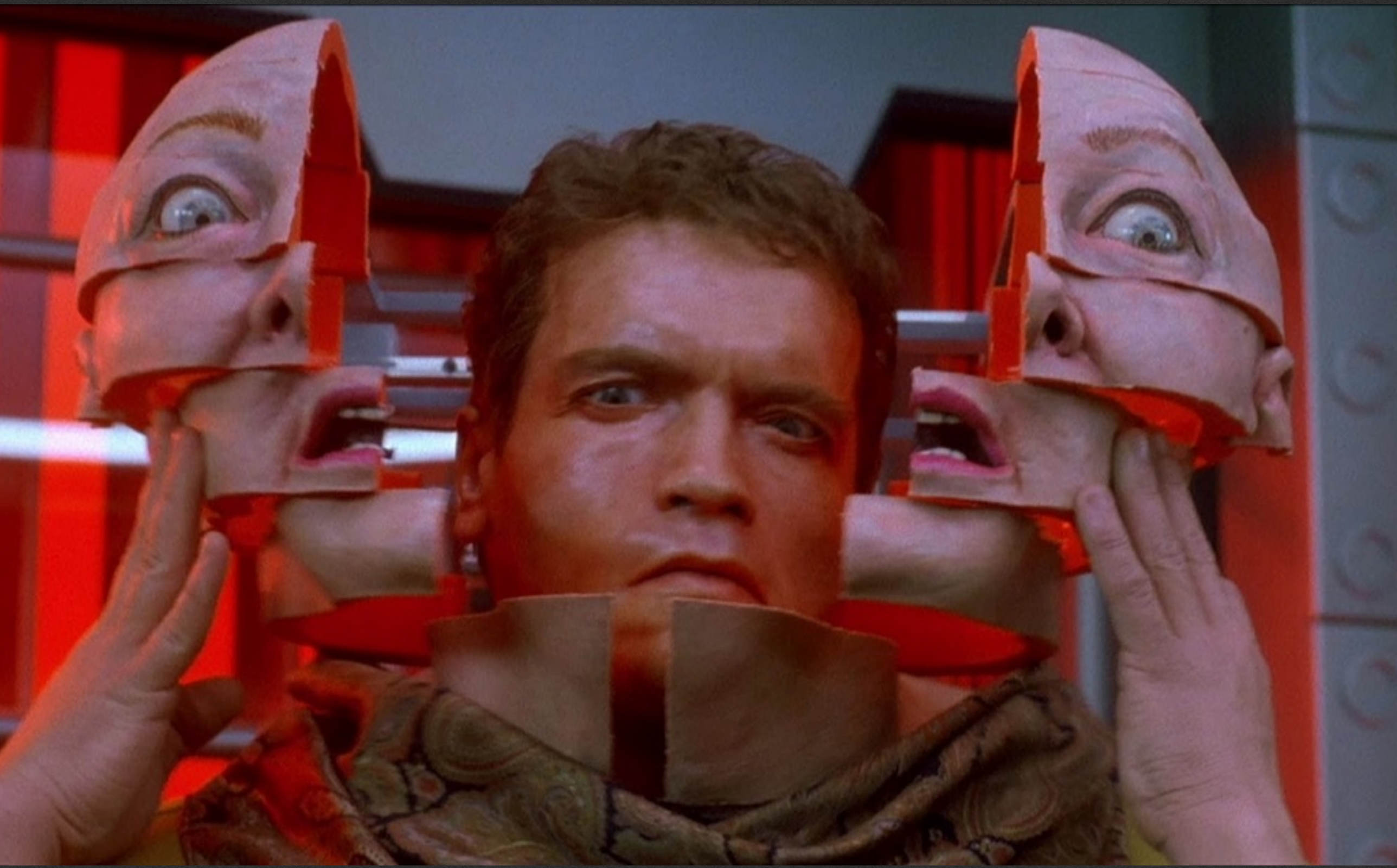


- actors
- futures (for timeouts)
- receiving actor *must* reply

ff & ff



@ignasi35

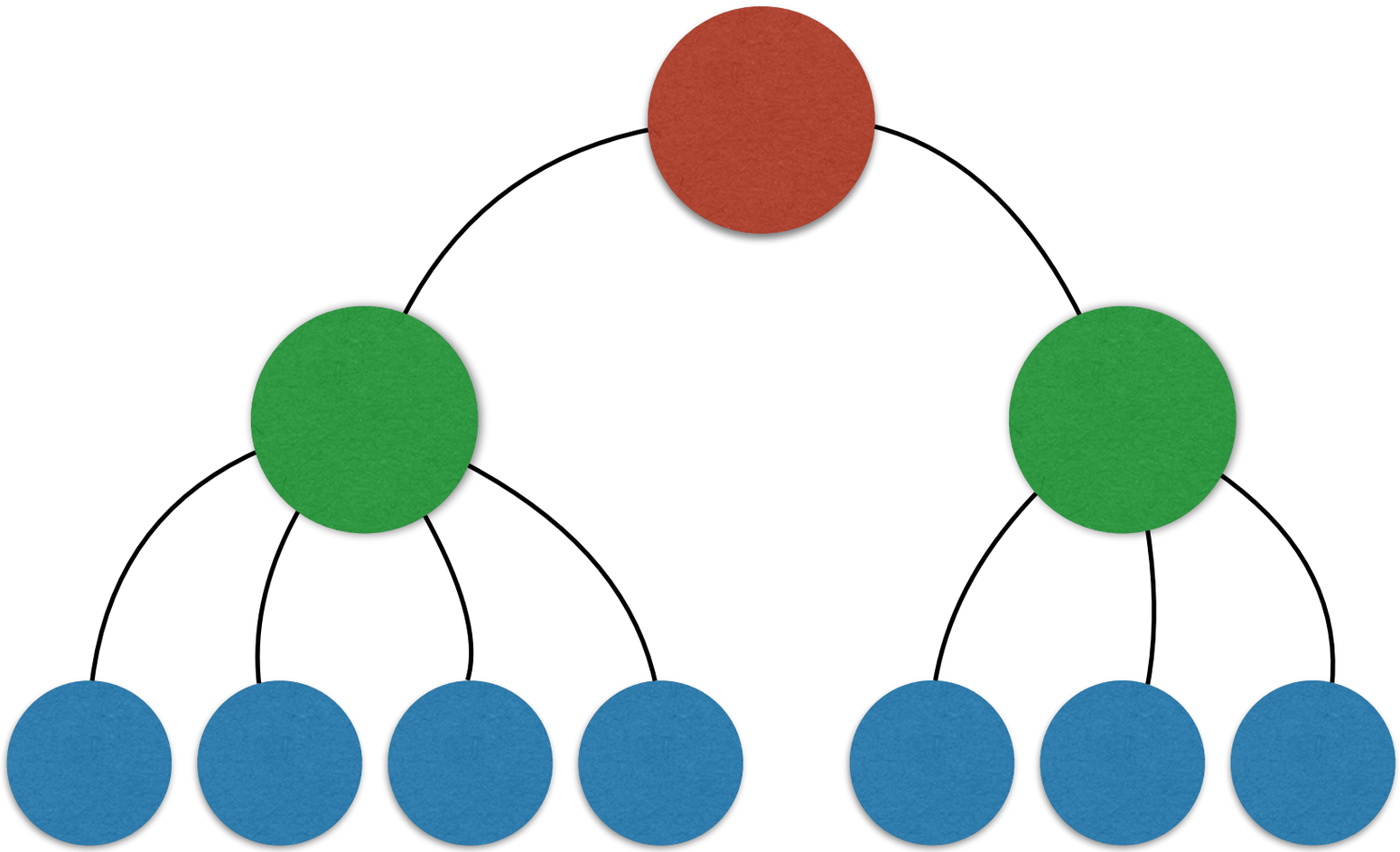


Actors/counter/counter2.ex

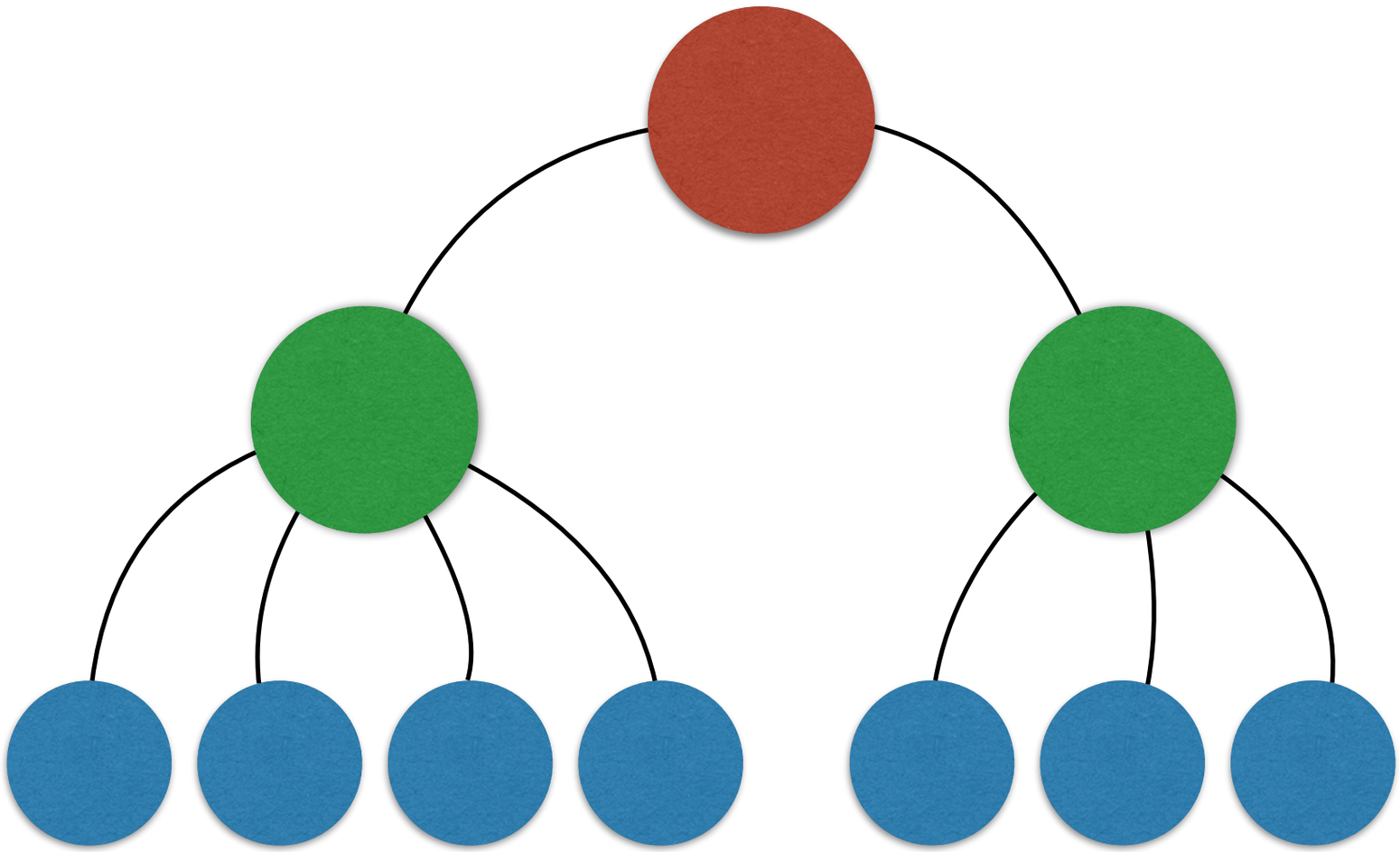
```
defmodule Counter do
  def start(count) do
    spawn(__MODULE__, :loop, [count])
  end
  def next(counter) do
    ref = make_ref()
    send(counter, {:next, self(), ref})
    receive do
      {:ok, ^ref, count} -> count
    end
  end
  def loop(count) do
    receive do
      {:next, sender, ref} ->
        send(sender, {:ok, ref, count})
        loop(count + 1)
    end
  end
end
```

source: 7 concurrency models in 7 weeks

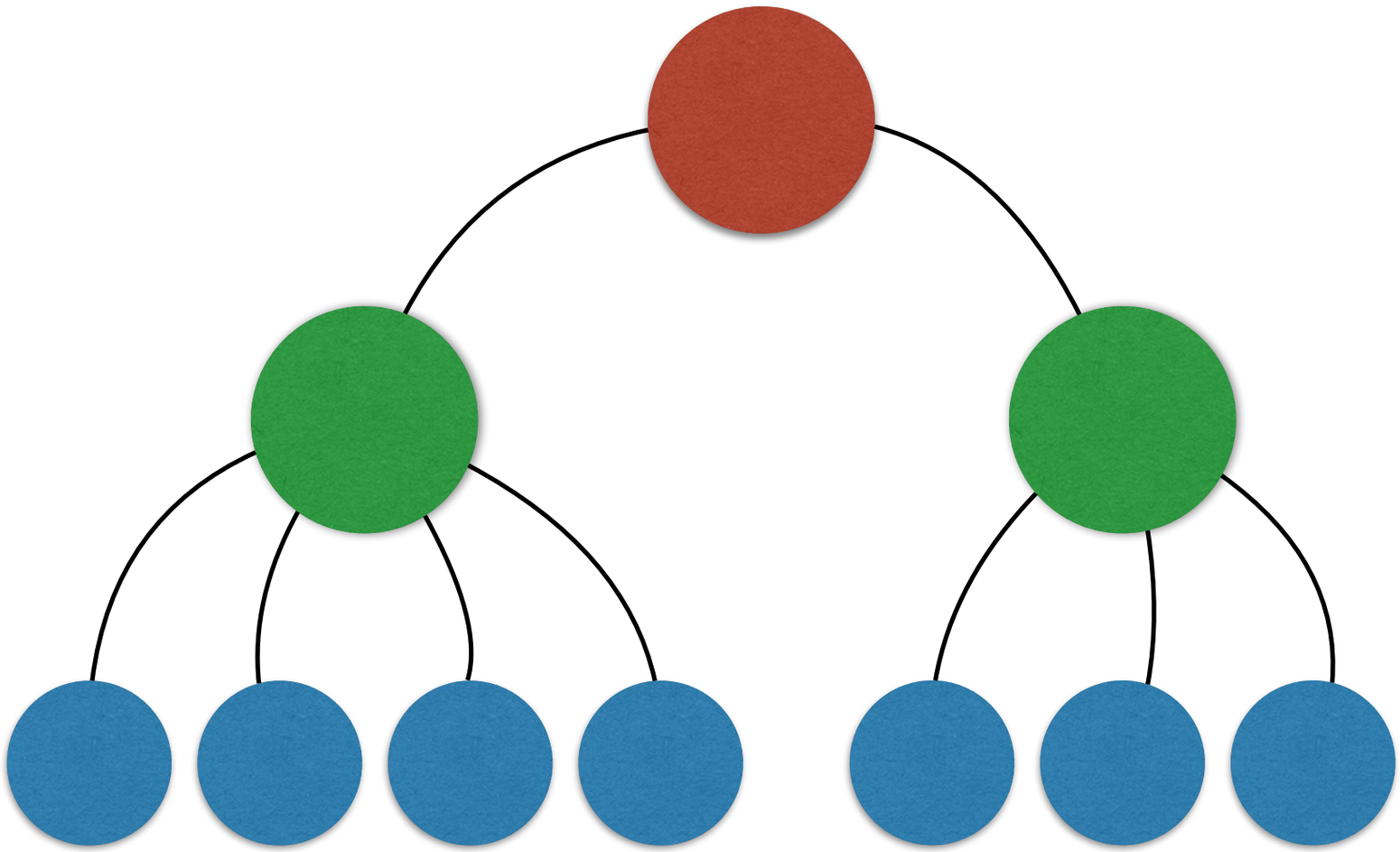
Supervision



One for One



One for One

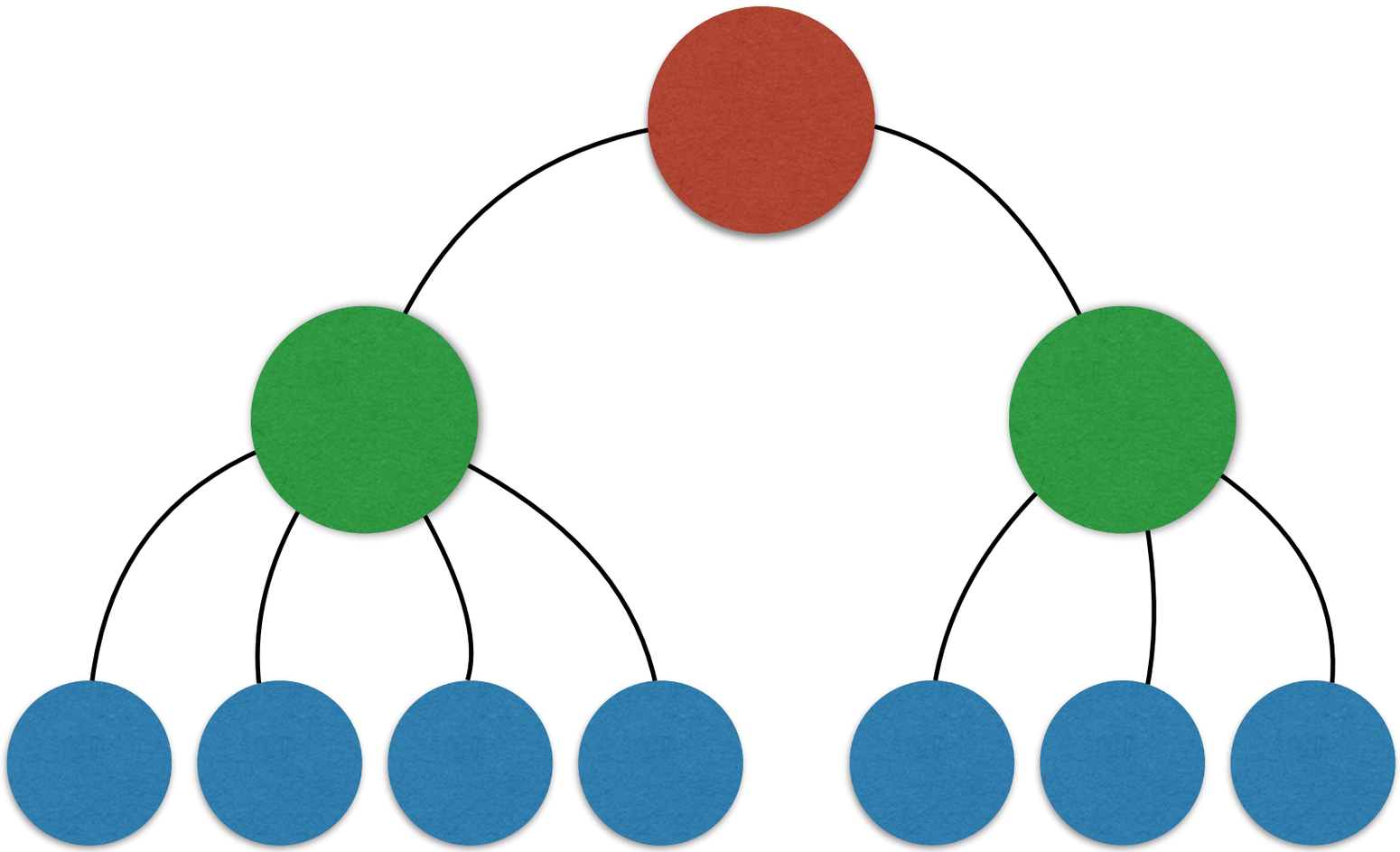


One for One

```
4.
5.  override val supervisorStrategy =
6.    OneForOneStrategy(maxNrOfRetries = 10, withinTimeRange = 1 minute) {
7.      case _: ArithmeticException       => Resume
8.      case _: NullPointerException      => Restart
9.      case _: IllegalArgumentException => Stop
10.     case _: Exception                 => Escalate
11.   }
```

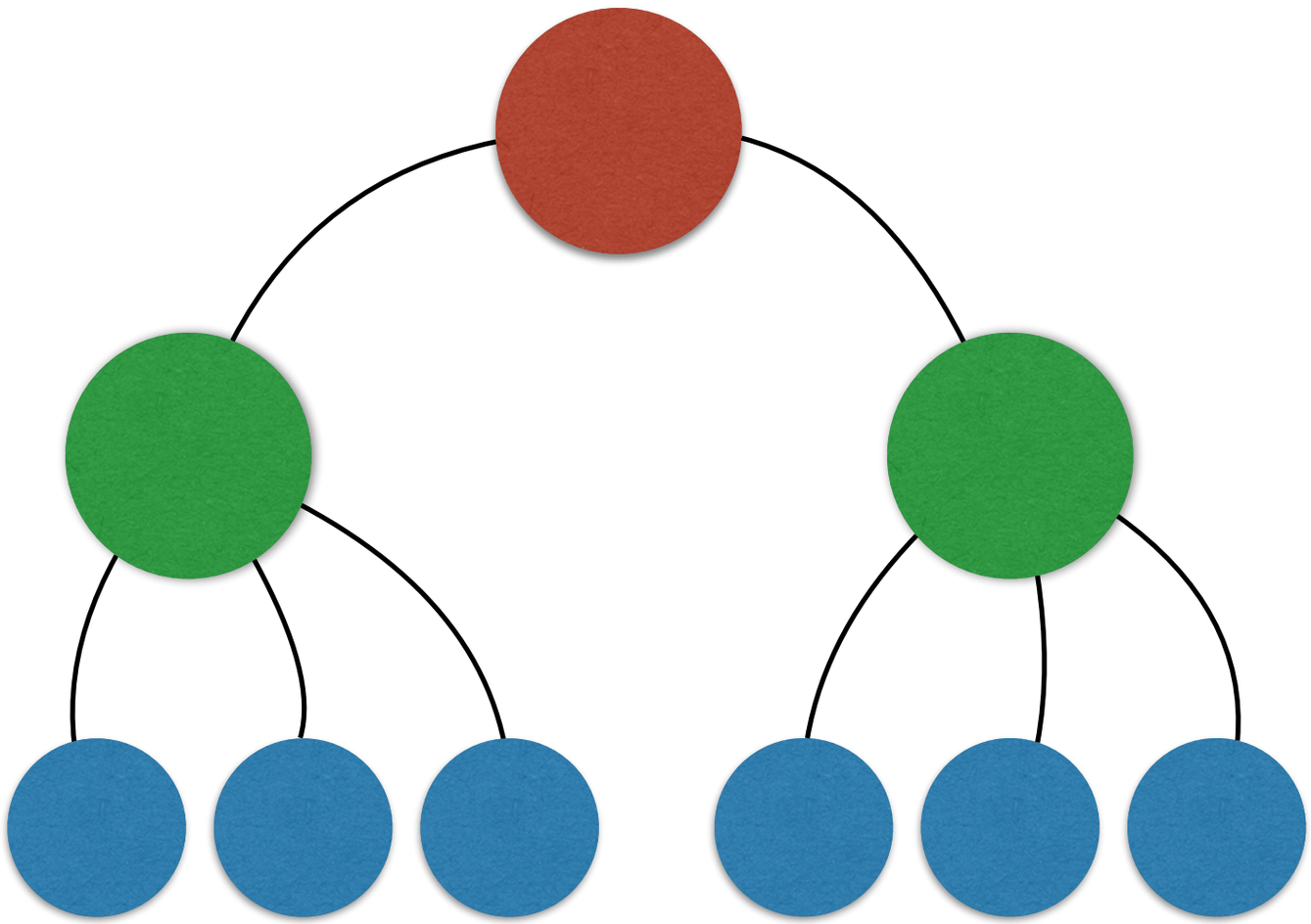
source: akka.io (old version)

All for One



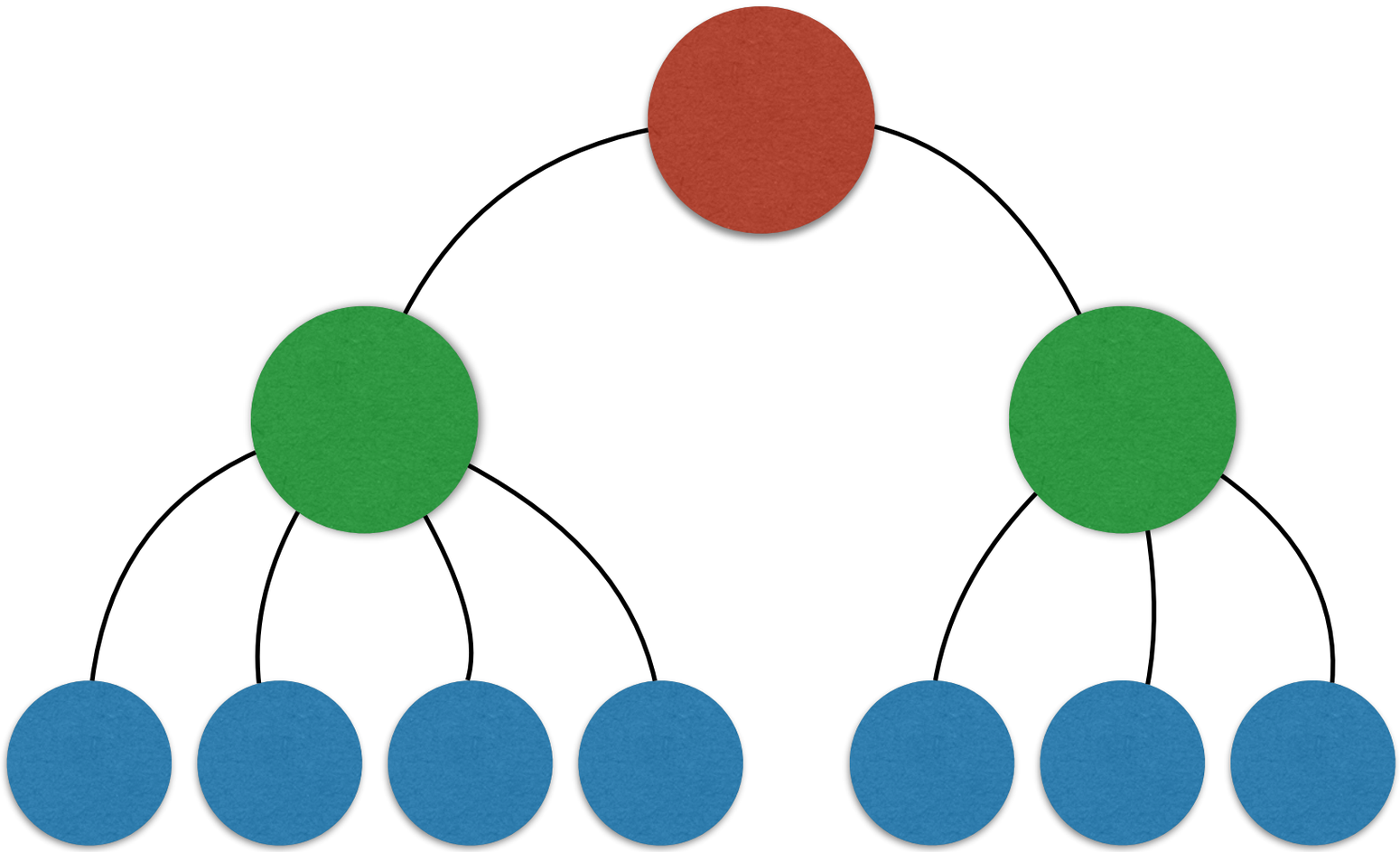
@ignasi35

All for One



@ignasi35

All for One



@ignasi35

Delivery

at-most-once

Tuning

Dispatcher *
PinnedDispatcher
BalancingDispatcher
CallingThreadDispatcher

Tuning

Dispatcher *

- one MB per actor
- many actors bound to a ThreadPool
- Bulkheading

Tuning

PinnedDispatcher

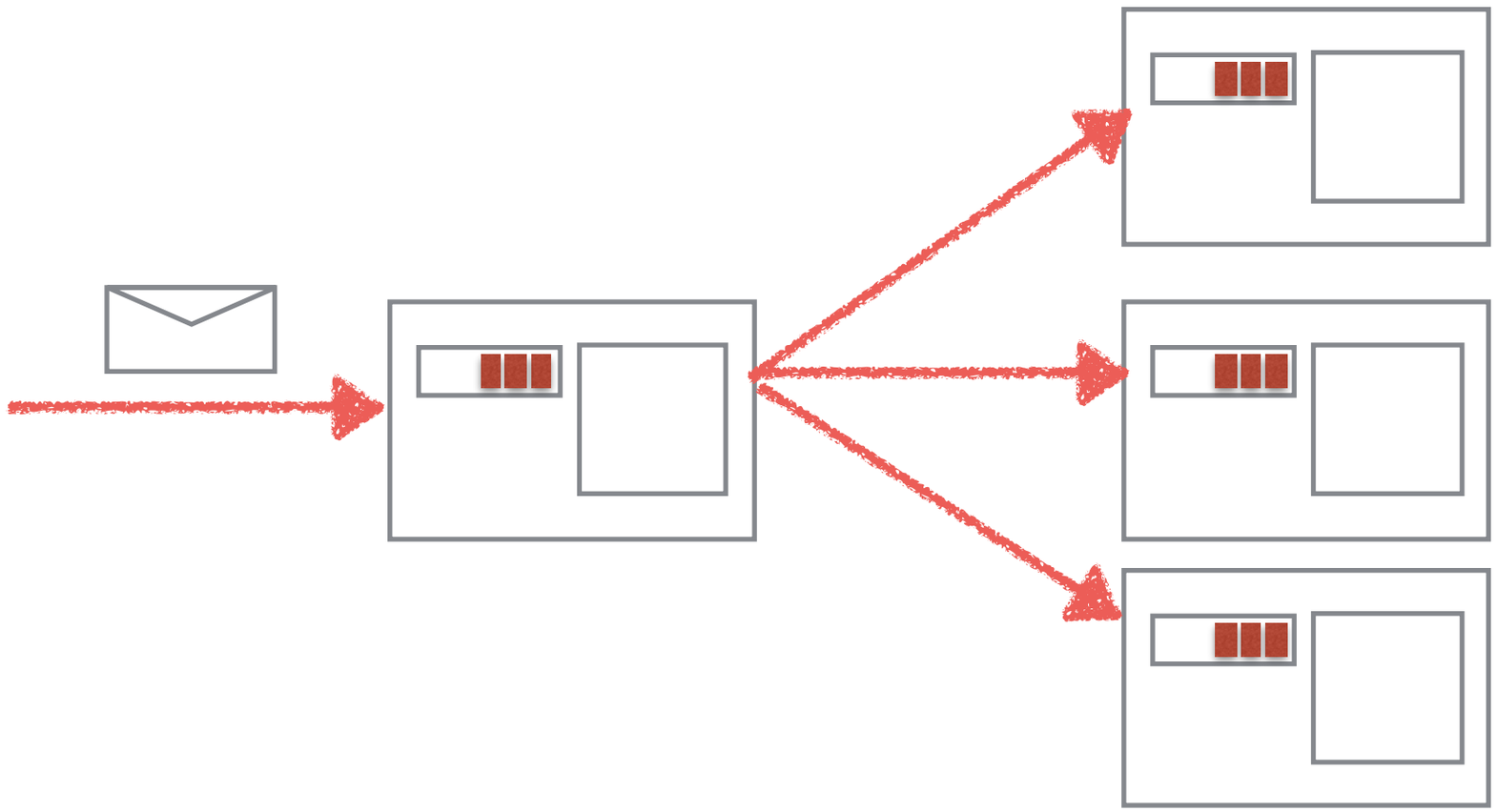
- one MB per actor
- dedicated Thread per actor
- Bulkheading

Tuning

BalancingDispatcher

- one MB - many actors
- ThreadPool
- work-sharing

Patterns



Patterns

- tailchopping
- balancing
- work pulling
- receptionist
- routing
- message forwarding
- …

Conclusions

- I/O bound vs CPU bound
- YMMV
- know your tools
- actors
  - distributed
  - at-most-once
  - supervision

@ignasi35


URGER

Questions?

@ignasi35

thanks!

@ignasi35

References

http://docs.oracle.com/cd/E19455-01/806-5257/6je9h032b/index.html

https://www.parleys.com/play/51c0bc58e4b0ed877035680a/chapter0/about

http://www.slideshare.net/legendofklang/concurrency-scaladays

http://dspace.mit.edu/handle/1721.1/6952#files-area

http://lmax-exchange.github.io/disruptor/

http://martinfowler.com/articles/lmax.html

http://gotocon.com/dl/goto-chicago-2014/slides/CharlieHunt_TheFundamentalsOfGCPerformance.pdf

https://www.youtube.com/watch?v=D6jqVS5JFpM&index=7&list=PLEx5khR4g7PJPfVDB1nUpsY2OG2wlf7hW