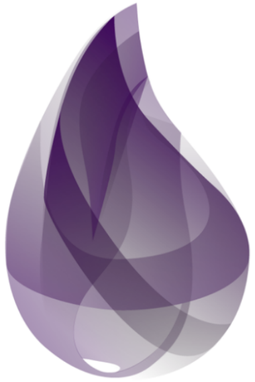
# NEBENLÄUFIGKEIT GANZ EINFACH MIT ELIXIR UND ERLANG

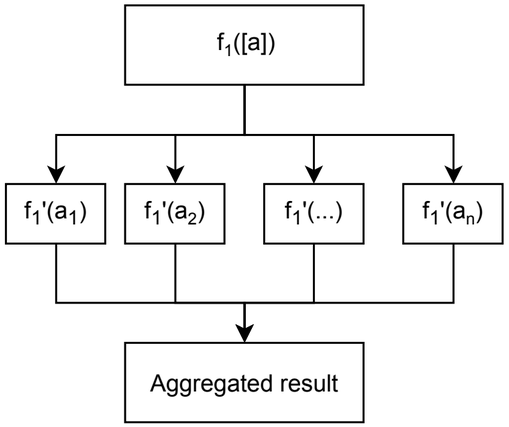



Martin Grotz, @mobilgroma, redheads Ltd.

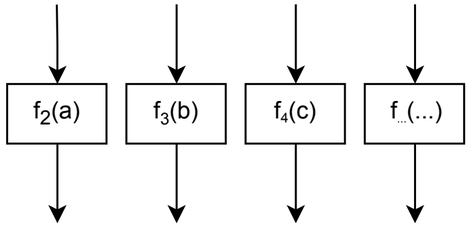
Herbstcampus 2019

# Parallelisierung vs. Nebenläufigkeit

## Parallelisierung

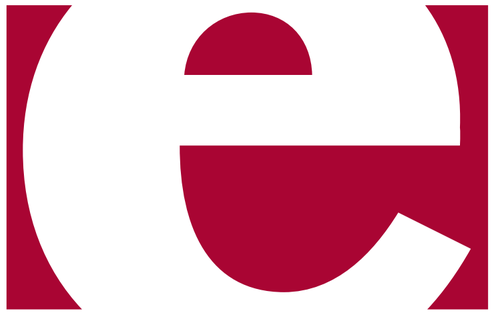



## Nebenläufigkeit



# Demo

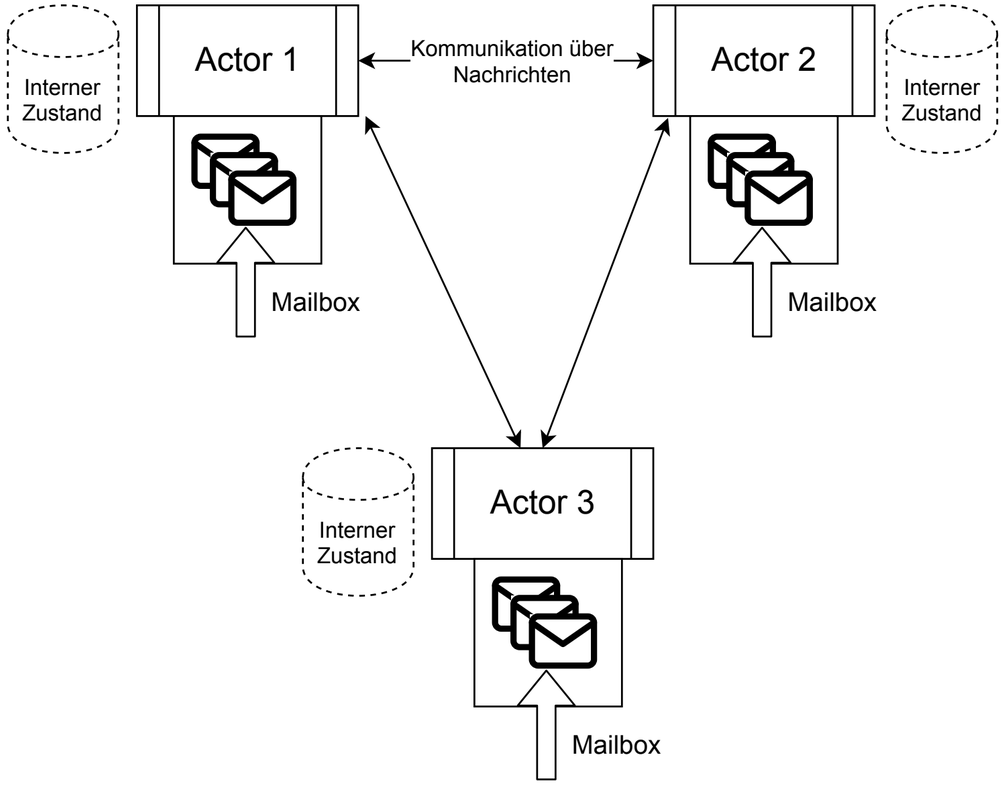
Demo-Anwendung von Saša Jurić



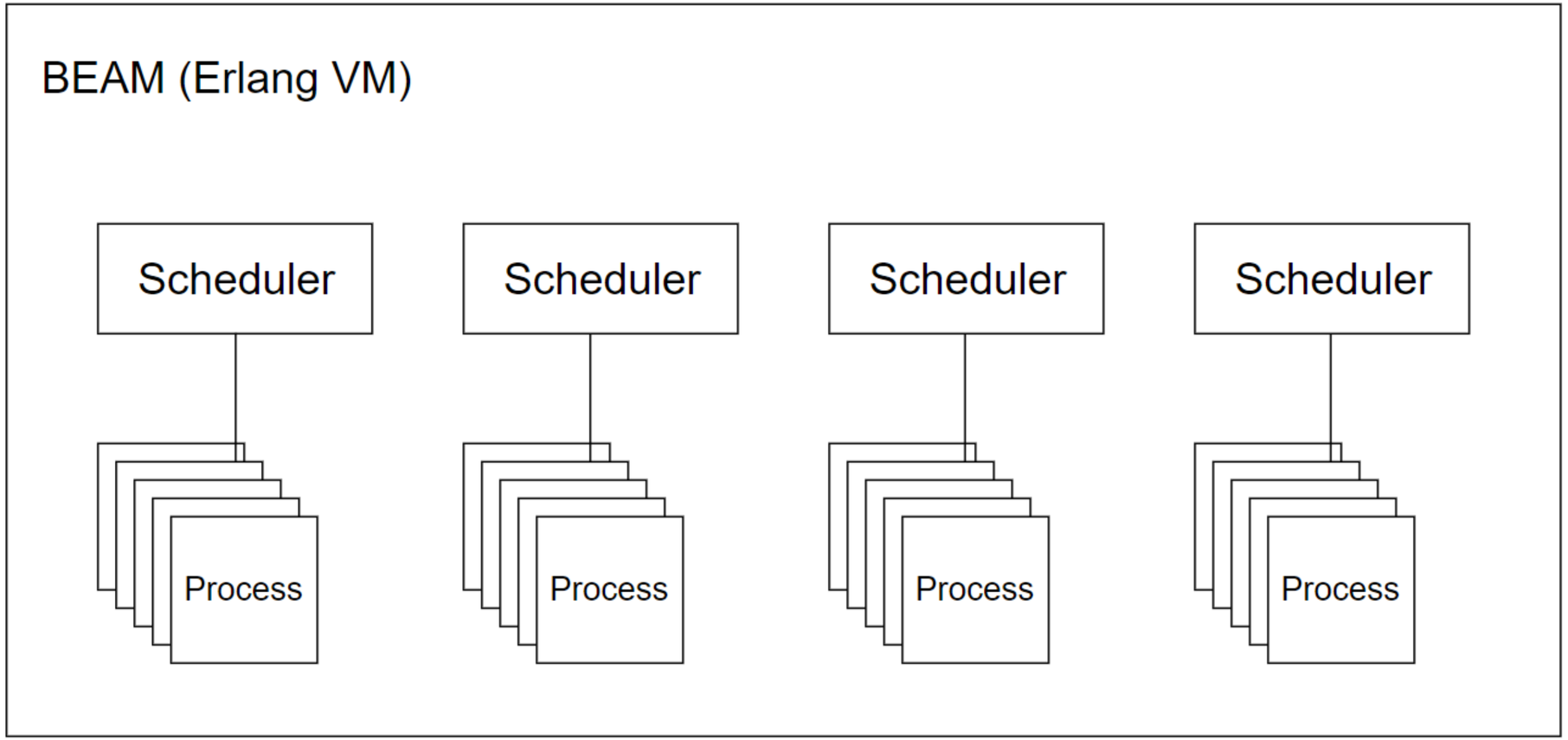
ERLANG

- "Concurrency Oriented Programming Language"
- Funktional, nebenläufig, verteilt
- Syntax angelehnt an Prolog
- "Versehentliche" Implementierung des Aktor-Modells

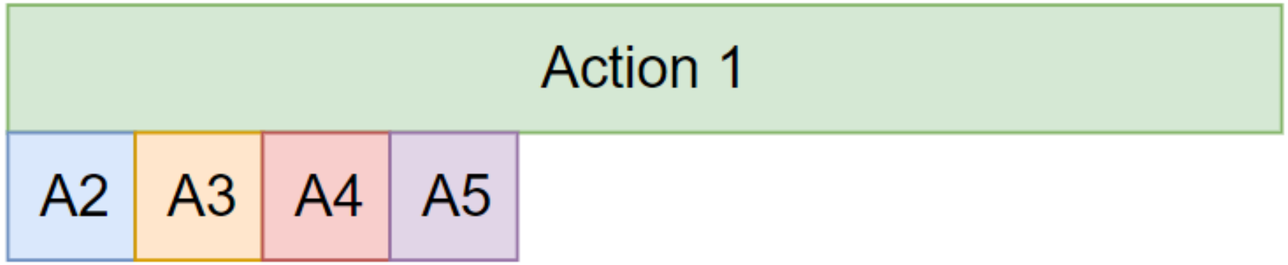
# Aktor-Modell und Prozesse




- Erlang: 1 Aktor == 1 Prozess
- Prozesse sind leichtgewichtig und vollständig voneinander isoliert
- In einem Prozess läuft alles sequentiell ab — Nachricht für Nachricht


BEAM (Erlang VM)
Scheduler
Scheduler
Scheduler
Scheduler
Process
Process
Process
Process

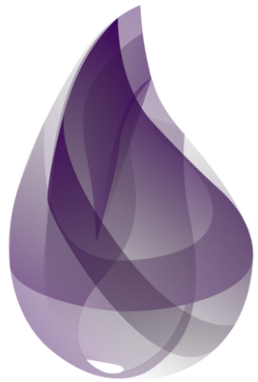
# Cooperative vs. preemptive scheduler




## Andere Runtimes (cooperative scheduler)


## Erlang VM (preemptive scheduler)

# Elixir



- Funktionale Allzweck-Programmiersprache
- Läuft auf der virtuellen Maschine von Erlang (BEAM)
- Für lang laufende Dienste ohne harte Echtzeit-Anforderungen
- Skalierbar und fehlertolerant durch Aktor-Modell und Erlang-Prozesse

# OTP

"If half of Erlang's greatness comes from its concurrency and distribution
and the other half comes from its error handling capabilities,
then the OTP framework is the third half of it."

# OTP-Bausteine

## Task

- Aufgabe einmalig ausführen
- async und await

## Agent

- Verwaltet Zustand
- get und set

## GenServer

- Standard-Abstraktion für Prozesse
- Zustand und Funktionen
- Wrapper um den "spawn, receive, send" Loop

# Supervisors

## Supervisor

- Lebenszyklus-Verwaltung
- Baumstruktur
- Wie werden Kindprozesse gestartet und gestoppt? Was passiert bei einem Crash?
- Statische Menge von Kindprozessen

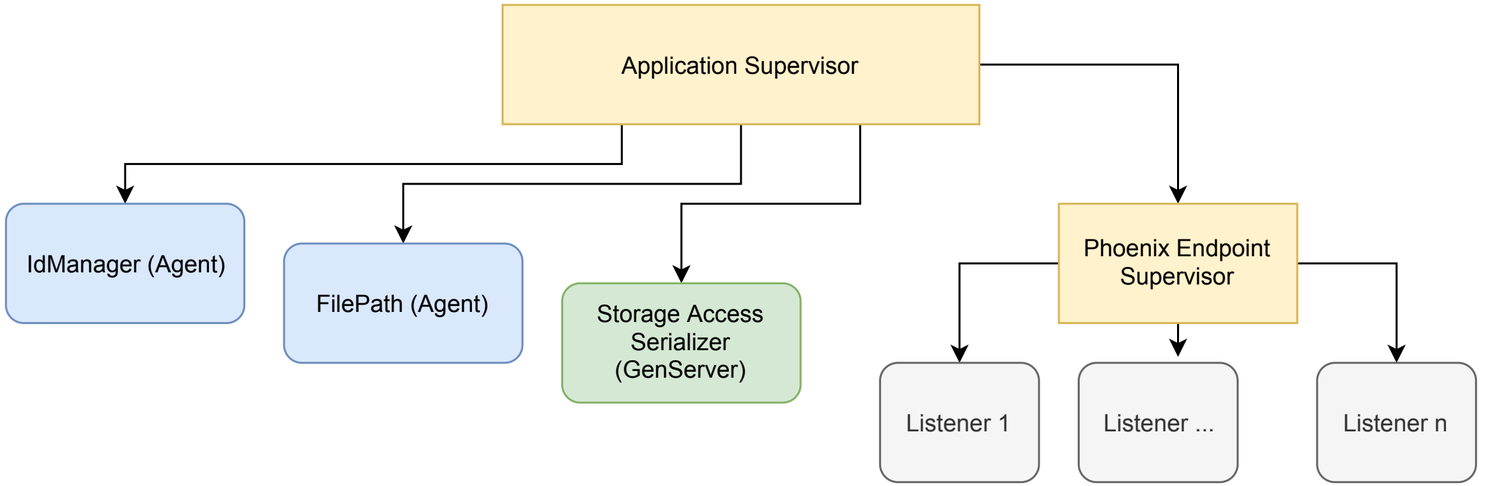
## Dynamic Supervisor

- Dynamische Menge von Kindprozessen
- Sonst identisch zum Supervisor

# Demo

## Interner Konzertkarten-Verkauf

# Architektur des Programms



# Prozesse

- Prozess == Aktor
- Leichtgewichtig (ca. 1 KB RAM pro neuem Prozess)
- Vollständig voneinander isoliert ("shared nothing")
- Innerhalb eines Prozesses läuft die Verarbeitung immer sequentiell

# Demo

## Zahlreiche gleichzeitige Game Sessions

# Fazit


- Elixir-Anwendungen dank Erlang skalierbar, verteilt und fehlertolerant
- Supervisor verändern die Art, über ein System nachzudenken
- Prozesse erlauben unkomplizierte Nebenläufigkeit

## Elixir (und die Erlang-VM) machen süchtig...

und Nebenläufigkeit in anderen Sprachen kommt einem anschließend umständlich vor

# Danke


E-Mail: martin.grotz@redheads.de
Twitter: @mobilgroma
Blog: https://elm.finde-ich-super.de
github: https://github.com/groma84

red·head·s
SOFTWARE SOLUTIONS SUPPORT