

# Aktoren in Java

---

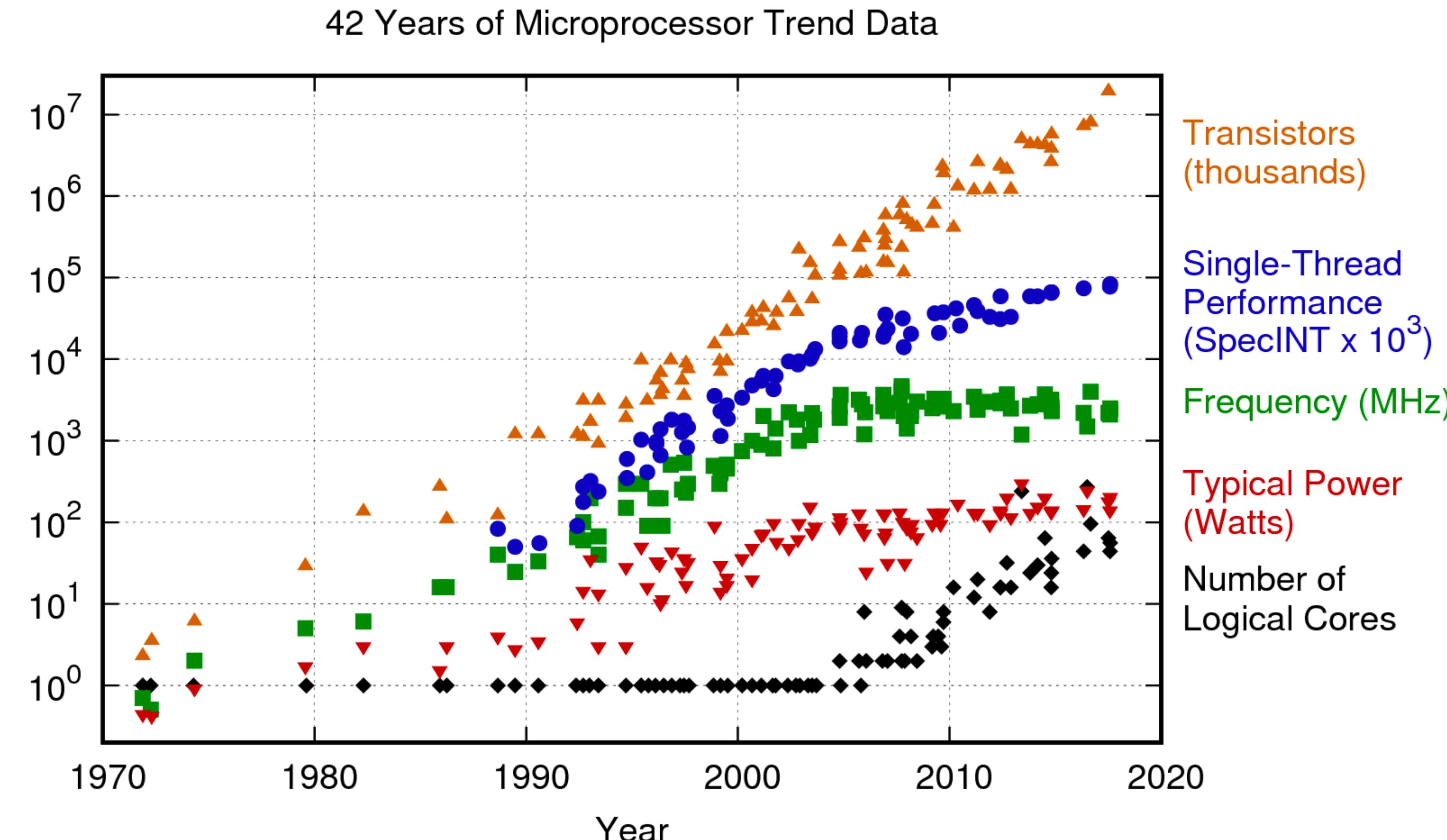
Probenvortrag  
Florian Lautenschlager  
Rosenheim, 09.11.2018

FRAGE? ANREGUNG?



Sind Akteure die besseren Threads  
zur Entwicklung nebenläufiger Anwendungen?

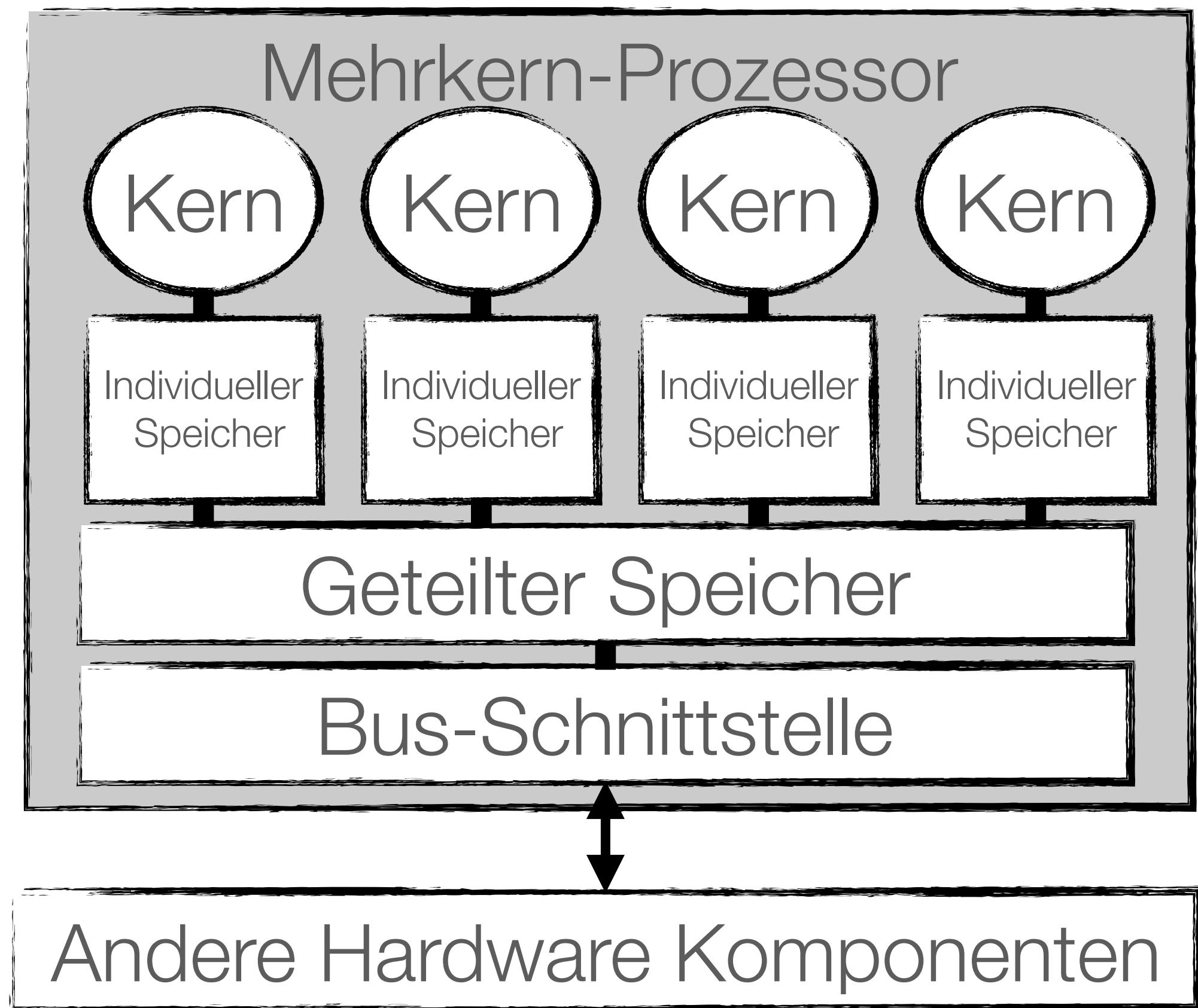
Die Taktfrequenz ist seit 2005 fast stabil und auch die Single-Thread Leistung steigt nur noch gering, während die Anzahl der Kerne steigt.



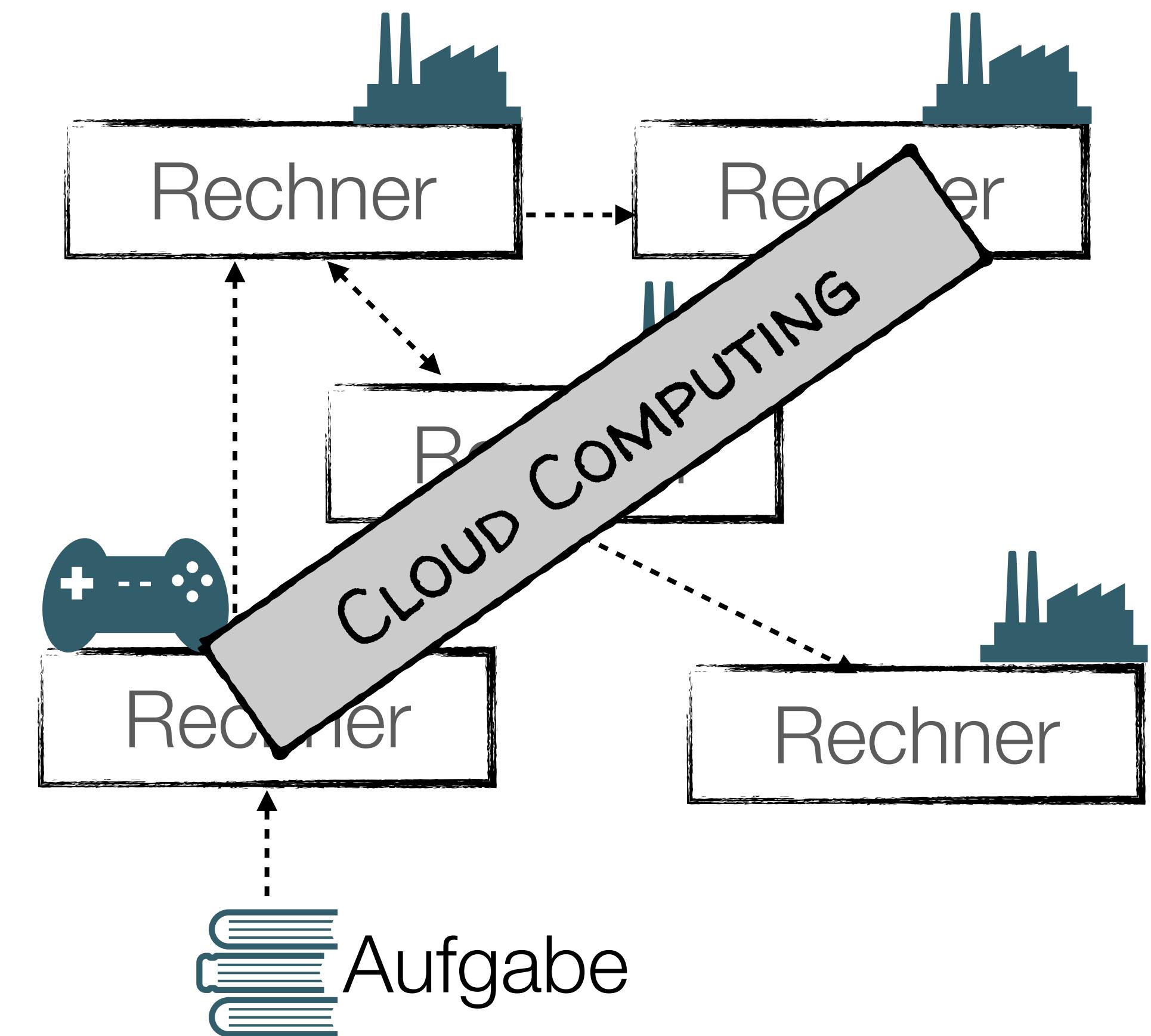
Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten  
New plot and data collected for 2010-2017 by K. Rupp

Schnelle Verarbeitung wird heutzutage über Nebenläufigkeit erzielt.  
Das geht im Großen wie auch im Kleinen.

Im Kleinen: Mehrere Kerne



Im Großen: Mehrere Rechner

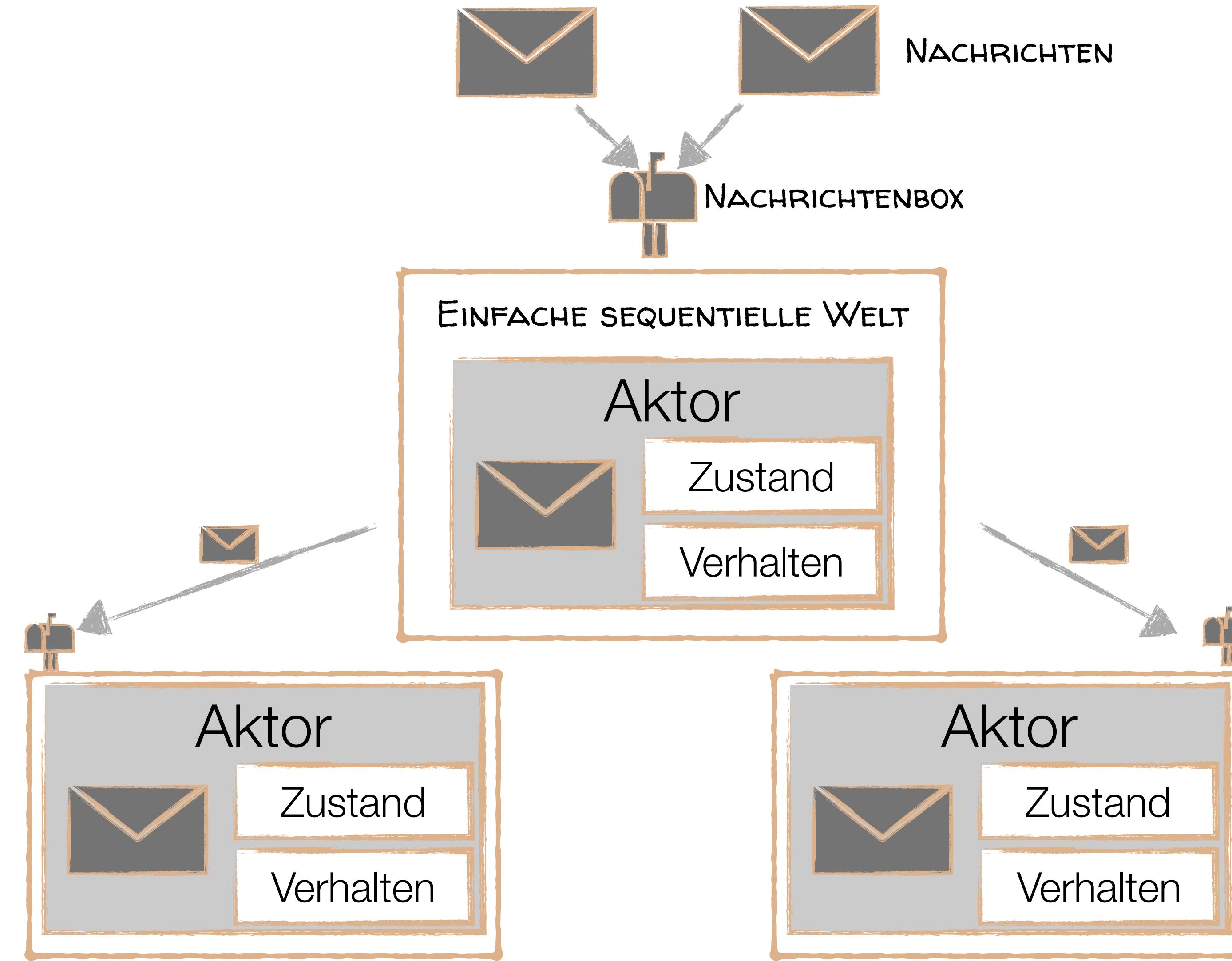


In Java ist der Grundbaustein für Nebenläufigkeit ein *Thread*.  
Parallelisierung mit Threads **IST SCHWIERIG.**

---



Das Aktorenmodell von Hewitt et al. (1973) ist ein höherwertiger Ansatz zum Umgang mit Nebenläufigkeit.



# Aktoren 101: Die Lerngruppe.

Drei Studierende (Aktoren) wollen einen Index für das Programmieren 3 Script erstellen.

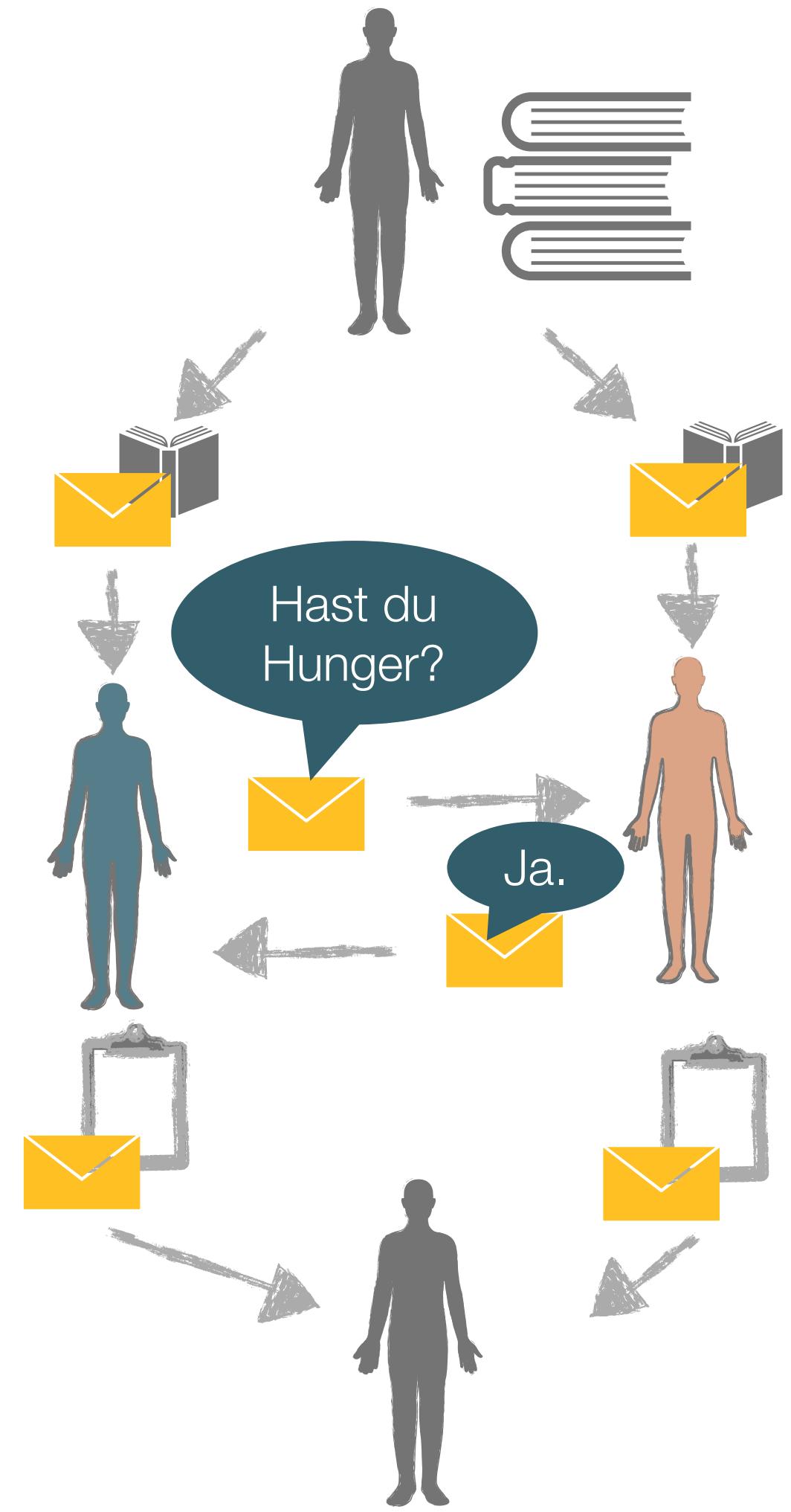
3 Regeln:

1. Mehrere Personen arbeiten nicht gleichzeitig am selben Script.
2. Niemand wird aktiv unterbrochen.
3. Ergebnisse dürfen nicht verändert werden.

**Schritt 1:**  
Eine Person muss das Script aufteilen.

**Schritt 2:**  
Jeder bearbeitet seinen Teil des Scripts (parallel). Hat jemand eine Frage, schreibt er einen Zettel. Der andere beantwortet die Frage, wenn er Zeit hat.

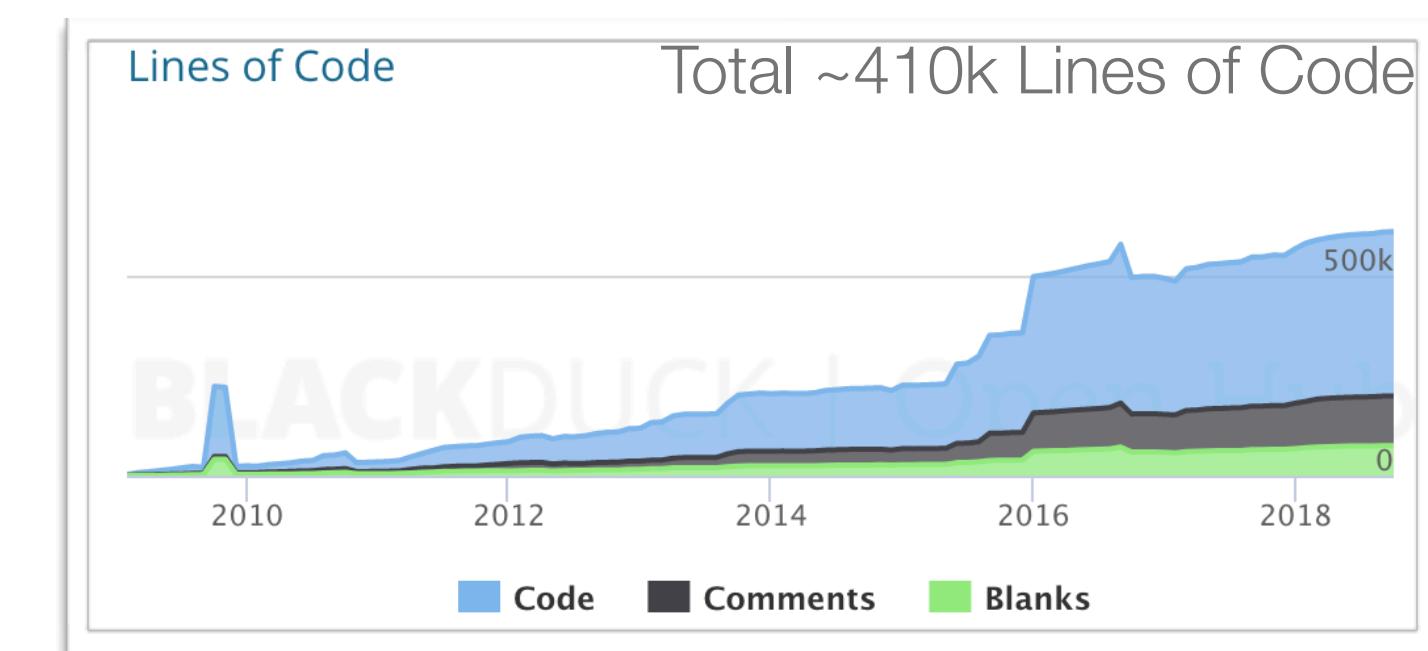
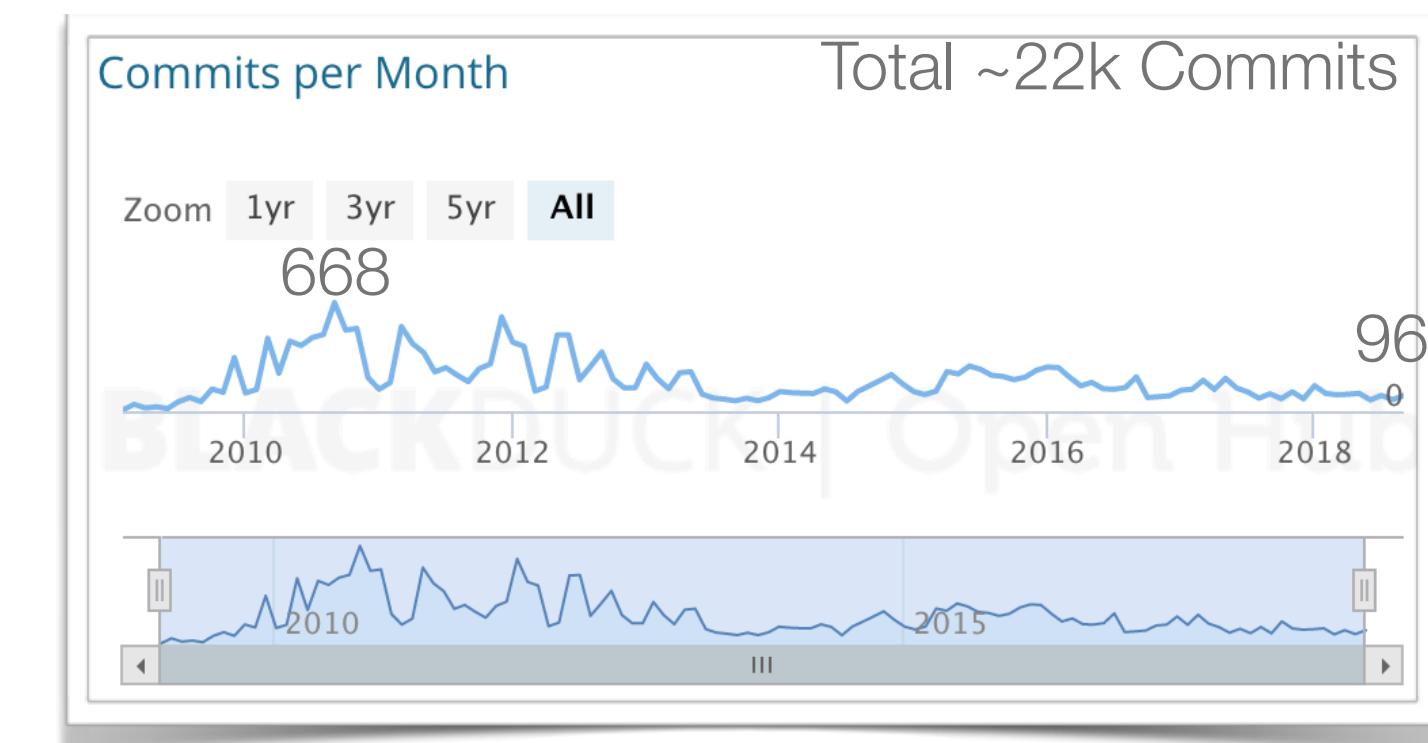
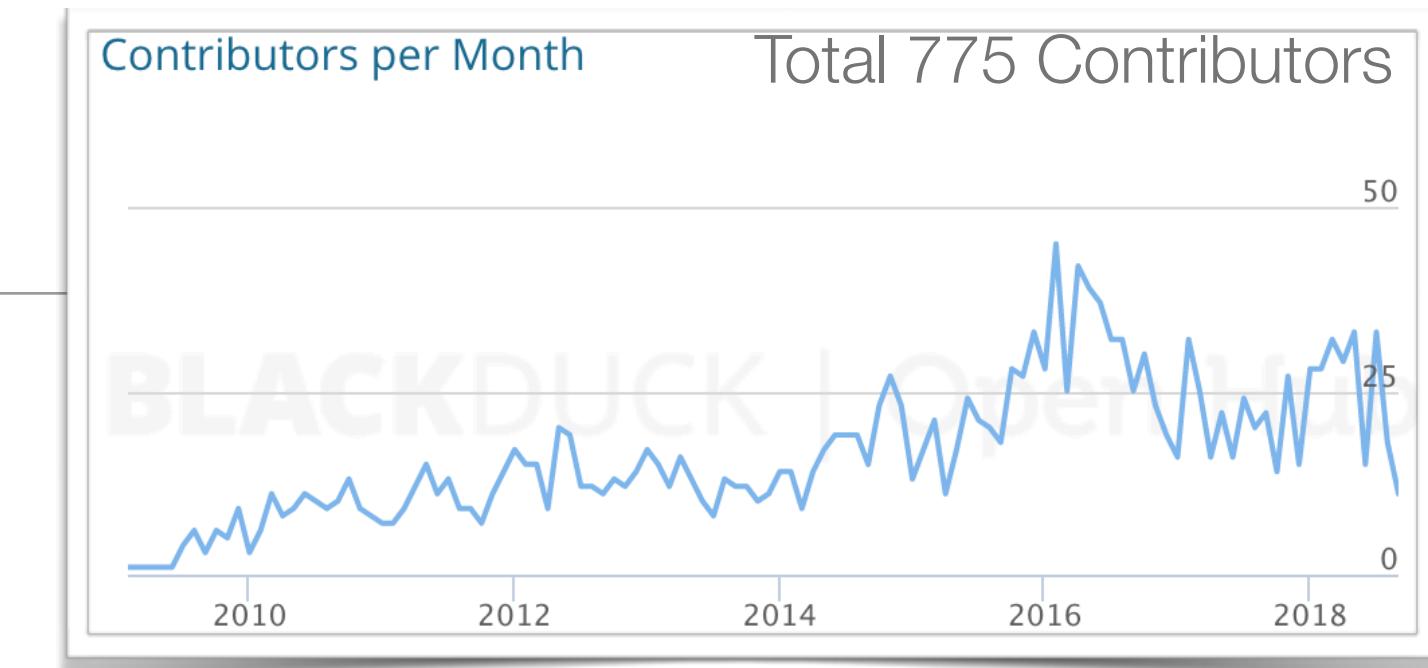
**Schritt 3:**  
Jeder schickt seinen Index als Nachricht an die Person, die das Script geteilt hat. Sie fügt beide Teile zusammen und druckt es für alle aus.



# Aktoren in Java mit dem Framework



- Implementierung nach dem Aktorenmodell
- Einfache Realisierung nebenläufiger Programme
- Transparenter Nachrichten-Kommunikationskanal
- Nebenläufigkeit im Kleinen und im Großen
- Entwicklung reaktiver Software-Systeme nach dem Reaktive Manifesto (<https://www.reactivemanifesto.org/de>).



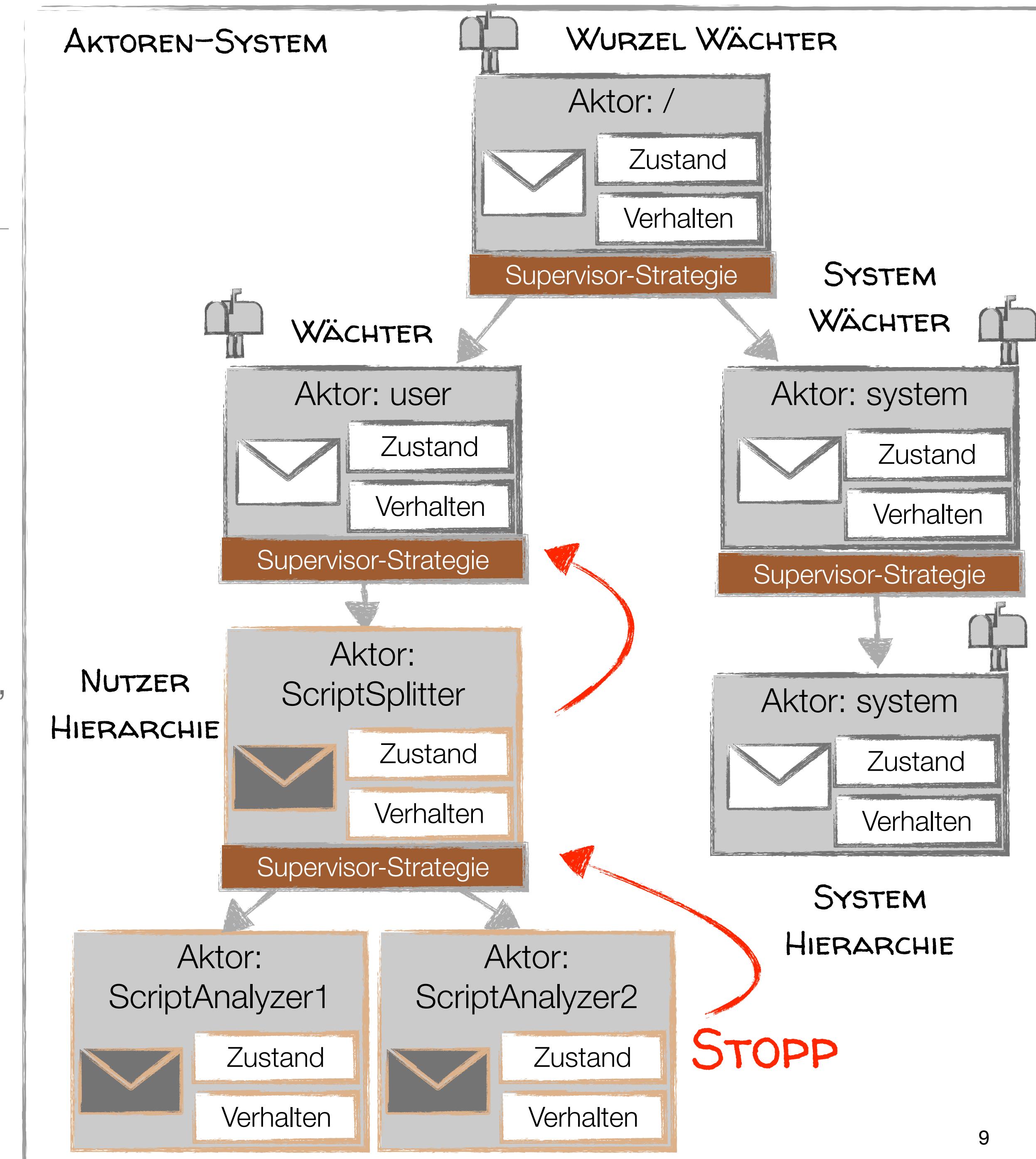
# Grundkonzepte in akka: Aktoren-System und Aktoren.

## Aktor:

- Zustand: Variablen etc.
- Verhalten: Logik zum Verarbeiten einer Nachricht
- Nachrichtenbox: Zu verarbeitende Nachrichten
- Kind-Aktoren: Mögliche Kinder eines Aktors (Supervisor)
- Aktor-Referenz: Kommunikation mit (Kind-)Aktoren
- Supervisor-Strategie: Hilfestellung für Kind-Aktoren, z. B. Fehlerbehandlung.

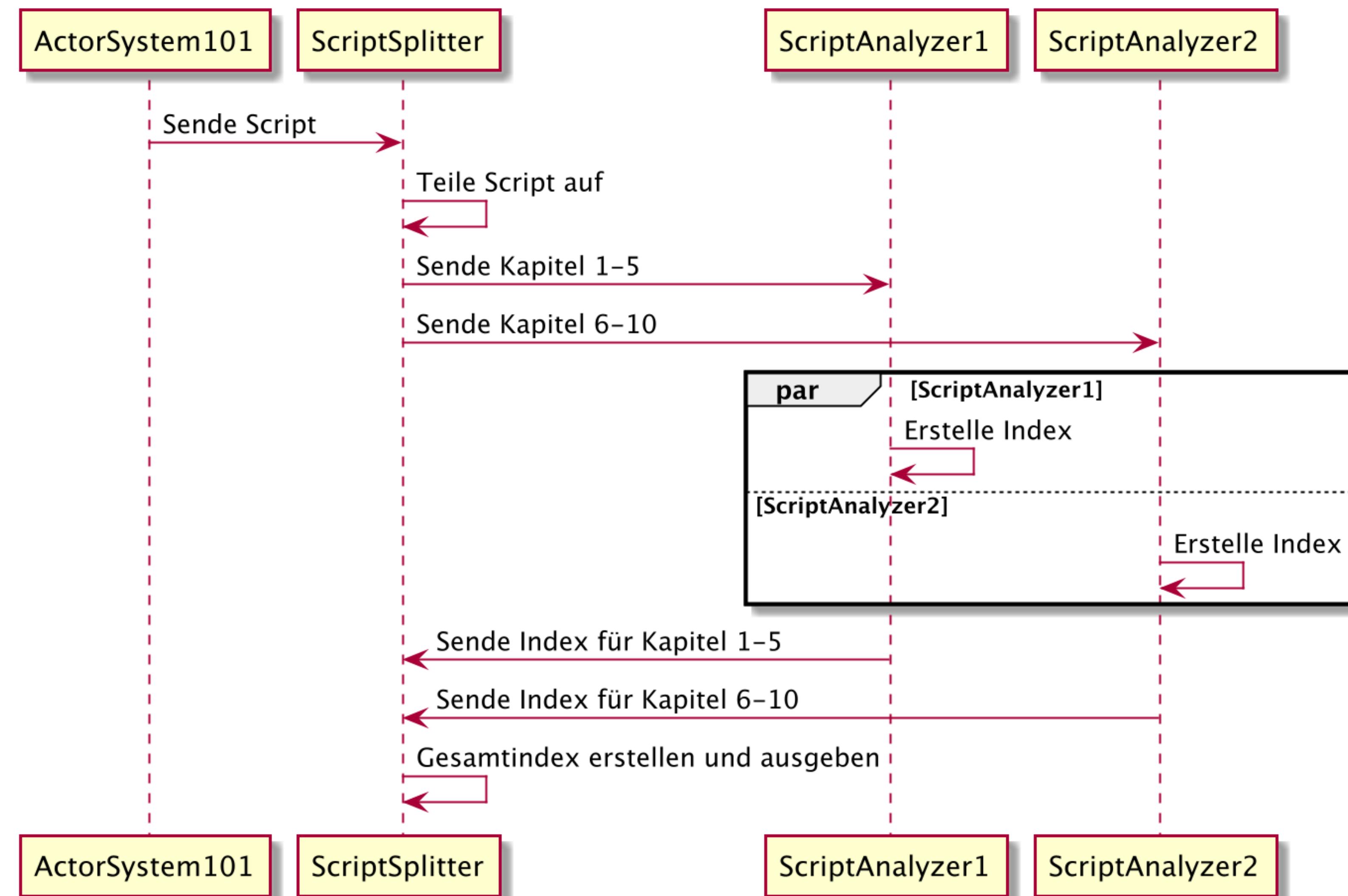
## Aktoren-System:

- Existiert einmal pro logische Anwendung
- Nutzer- und System-Aktoren
- Hierarchische Struktur
- Steuert Threads zum Ausführen der Aktoren
- Rekursives Beenden (erst Kind-Aktoren)



# Aktoren 101: Die Lerngruppe encoded.

## Run



Ladies and Gentlemen start your Engines.



```
//get it
git clone https://github.com/FlorianLautenschlager/akka-101.git

//mix it
./mvnw clean compile assembly:single

//run it
java -jar target/akka-1.0-SNAPSHOT-jar-with-dependencies.jar
```

Sind Aktoren die besseren Threads  
in nebenläufigen Anwendungen?

# Aktoren erleichtern die Realisierung nebenläufiger Anwendungen.

---

## Ja

- Einfache sequentielle Programmierung (in Aktoren)
- Unterteilung der Anwendung in mehrere Aktoren und Funktionen (ähnlich funktionaler Programmierung)
- Transparenter Nachrichtenkanal zur lokalen oder verteilten Ausführung einer Anwendung
- Trennung von A und T: Anwendungscode nicht vermischt mit Technik-Code

## Aber

- Kommunikation mittels Nachrichten teurer als geteilte Variablen
- Debugging: Nachrichtenfluss schwierig nachvollziehbar
- Keine Garantie des Zustellungszeitpunkts
- Optimierung: Durchsatz (Aktoren) vs. Antwortzeit (Threads)

<https://answergarden.ch/808743>

# Literatur und Quellen

---

## Aktorenmodell:

- Carl Hewitt, Peter Bishop, Richard Steiger: A universal modular ACTOR formalism for artificial intelligence, Proceeding IJCAI'73 Proceedings of the 3rd international joint conference on Artificial intelligence, Pages 235-245 (<https://eighty-twenty.org/files/Hewitt,%20Bishop,%20Steiger%20-%201973%20-%20A%20universal%20modular%20ACTOR%20formalism%20for%20artificial%20intelligence.pdf>)

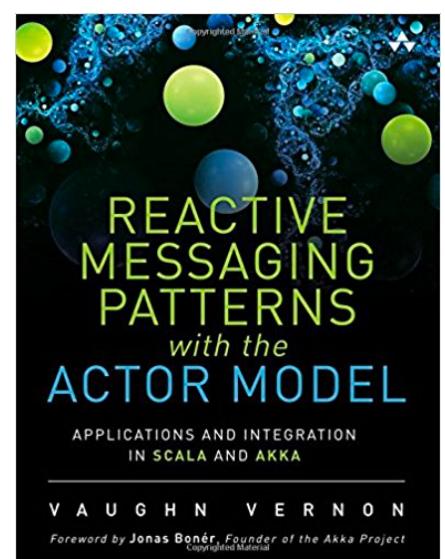
## akka:

- Framework Dokumentation: <https://doc.akka.io/docs/akka/2.5/>

- Bücher:



- Vaughn Vernon: Reactive Messaging Patterns with the Actor Model: Applications and Integration in Scala and Akka, Addison Wesley; Auflage: 01, ISBN-13: 978-0133846836
- Raymond Roestenburg, Diverse Vorträge: <https://speakerdeck.com/rayroestenburg>



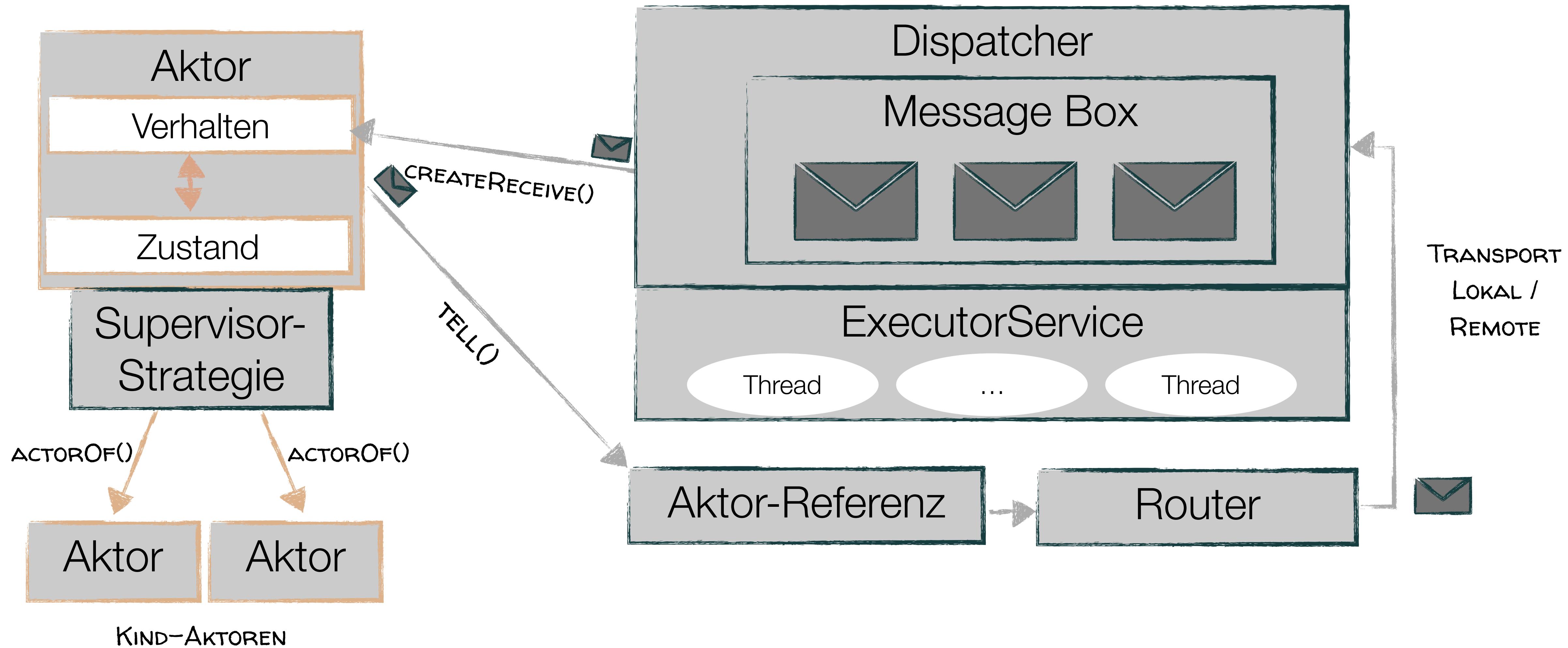
**Quellcode von Heute:** <https://github.com/FlorianLautenschlager/akka-101>

Backup

Entwicklerbaustein

# Die Bausteine von akka aus der Entwicklerbrille.

akka-Baustein



# Reaktive Anwendung: Das Reactive Manifesto

<https://www.reactivemanifesto.org/de>

## Elastisch:

Das System bleibt auch unter sich ändernden Lastbedingungen antwortbereit.

## Antwortbereit:

Das System antwortet unter allen Umständen zeitgerecht, solange dies überhaupt möglich ist.

## Widerstandsfähig:

Das System bleibt selbst bei Ausfällen von Hard- oder Software antwortbereit.

## Nachrichtenorientiert:

Das System verwendet asynchrone Nachrichtenübermittlung zwischen seinen Komponenten.

