Asynchrones Programmieren in Clojure

mit Hilfe der Bibliothek core.async

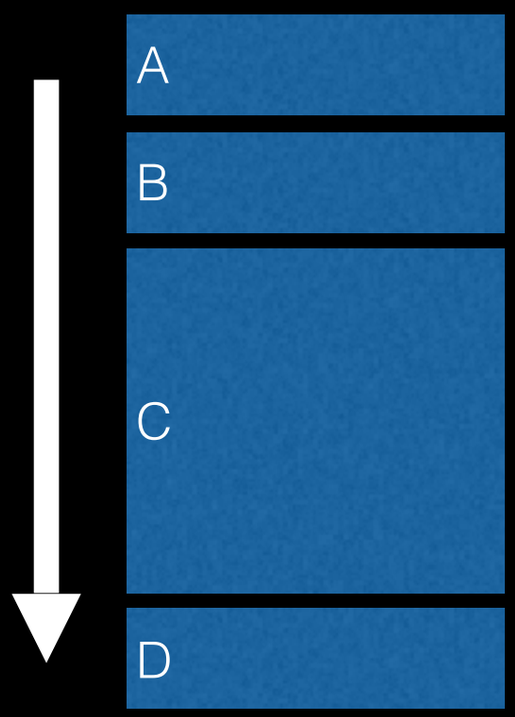
Vincent Elliott Wagner
Tobias Schwalm

Inhalt

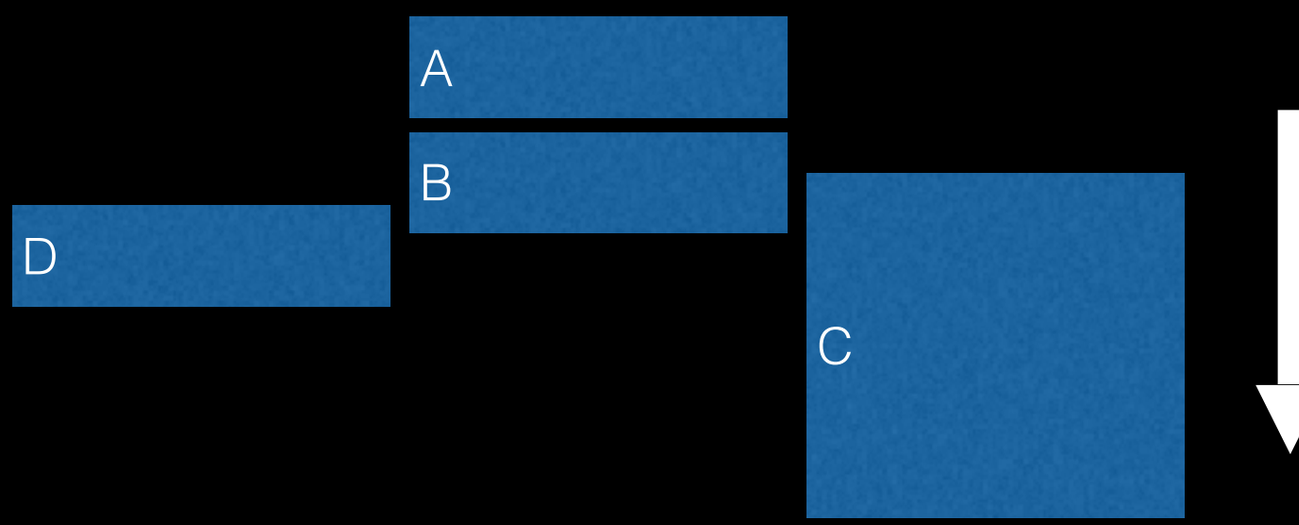
1. Was bedeutet asynchrone Programmierung?
2. Blockierende und nichtblockierende I/O
3. Asynchrone Programmierung in Clojure
4. ClojureScript
5. Fazit

1. Was bedeutet asynchrone Programmierung?

Annahme: Programmcode wird sequentiell abgearbeitet
B und C müssen nach A erfolgen, D muss nach A erfolgen

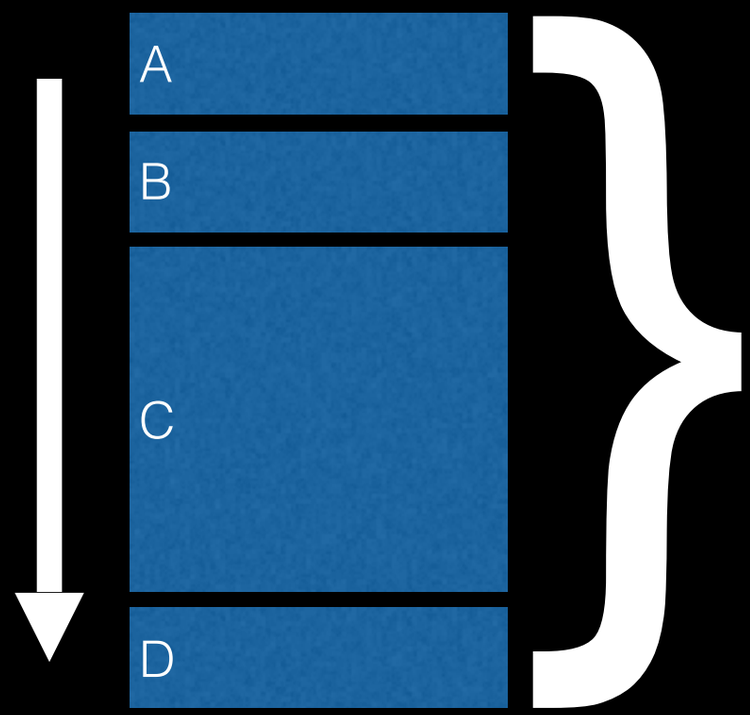


synchrone Abarbeitung



asynchrone Abarbeitung

2. Blockierende und nichtblockierende I/O

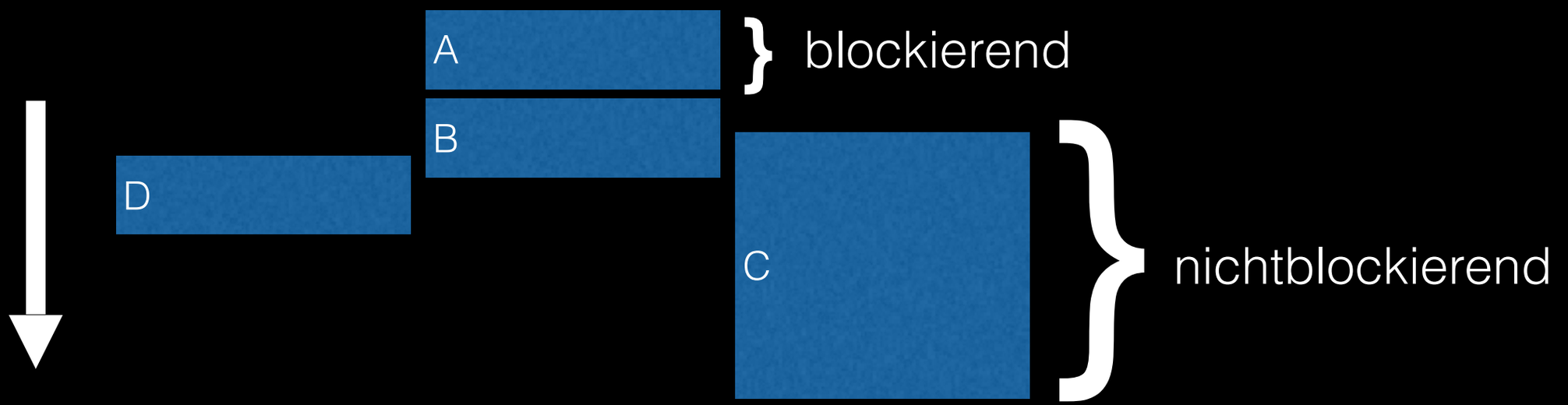


ausschließlich blockierende Aufrufe

Befehlsfolge: A, B, C, D

Solange A abgearbeitet wird, wird das System blockiert. B muss warten.

2. Blockierende und nichtblockierende I/O



Mögliche Befehlsfolge: A, B, D, C

Solange A abgearbeitet wird, wird das System blockiert. B, C, D werden angestoßen.

3. Asynchrone Programmierung in Clojure

1. Clojure spezifische APIs (future-call bzw. future)
2. Java spezifische APIs (Executor- ForkJoin-Framework, Threads)

Problematik

- Threads
  - schwergewichtig
  - oft existiert nur ein Thread (z. B. bei JavaScript)
- Kommunikation über einen shared Storage
  - Race-Conditions
  - schwer zu handhaben
  - Deadlocks

4. ClojureScript

„ClojureScript is a compiler for Clojure that targets JavaScript. It is designed to emit JavaScript code which is compatible with the advanced compilation mode of the Google Closure optimizing compiler.“

Quelle: http://clojure.org/clojurescript

Fazit

Fragen?