Clojure
concurrency revisited

Kjell Otto

25.11.2009

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Motivation

- Höhere Taktraten für CPUs nicht mehr Energieeffizient
- Moore's Law trifft trotzdem zu - Anzahl der CPUs wird erhöht
- Aufwendige Berechnungen sind seriell zu langsam
- Multicoreprogrammierung ist die Lösung

[4]

„*Concurrency is one of the most tricky aspects of software development.*“ *[2]*

Gliederung

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Grundlagen

- Funktional - Lisp
- unveränderliche, persistente Datenstrukturen
- Läuft auf der JVM
- Java-Interop
- Geschichte zur Koordination von Parallelisierung
- OpenSource

Features

- Dynamische Entwicklung
  - REPL, compiliert direkt zu JVM bytecode
- „Primitive Datentypen“
  - Nummern, Brüche, Chars, Strings, Symbole, Keywords
- Aggregates - Listen, Maps, Vektoren und Sets
  - Alle unveränderlich, persistent und erweiterbar
- Abstrakte Sequences - Implementierung über Iterable
- Metadaten
- Lisp- Macros

Was ist das Problem?

Concurrency konventionell

- C, C++, Java, Ruby, Python...
- Absprachen
- Mutex, Semaphore etc.
- Der Programmierer entscheidet

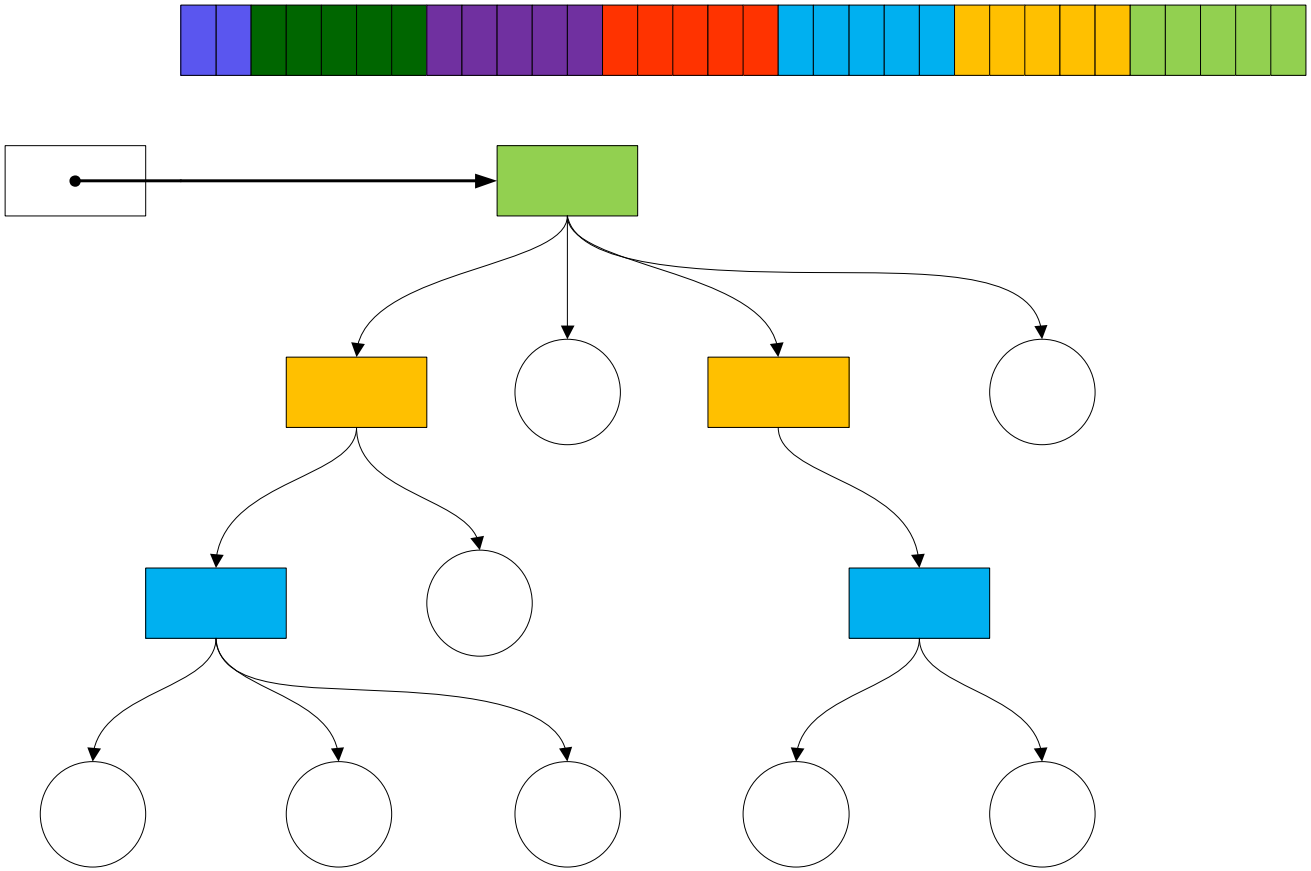
Das Problem ist nicht 20 Threads zu starten.
Das Problem ist die 20 Threads zu koordinieren.

Persistente, unveränderliche Datenstrukturen

- Was bedeutet Persistenz?
  - Historie ist bekannt
- Was bedeutet Unveränderlichkeit?
  - Kein Wert ändert sich
- Effizienz ist wichtig
- Gleiche Leistungsmerkmale
- Mit (Garbage Collection) GC kein Problem

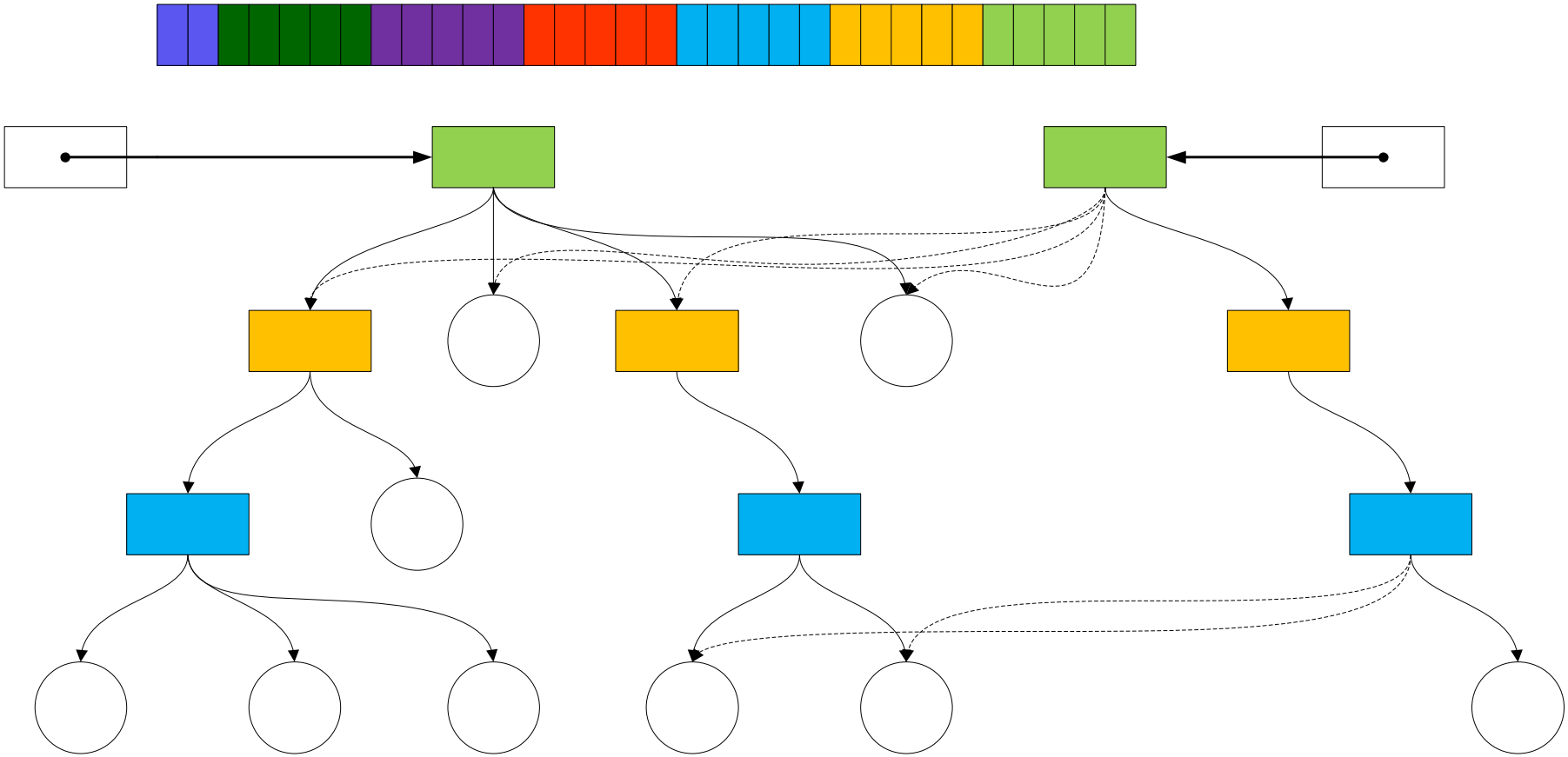
Was ist der Vorteil?

Bit partitioned hash tries



[1]

Structural sharing und Path Copying



[1]

Referenztypen

- Ref - koordinierte, synchrone Updates
- Atom - unkoordinierte, synchrone Updates
- Agent - asynchrone Updates

Werte werden änderbar!

[3]

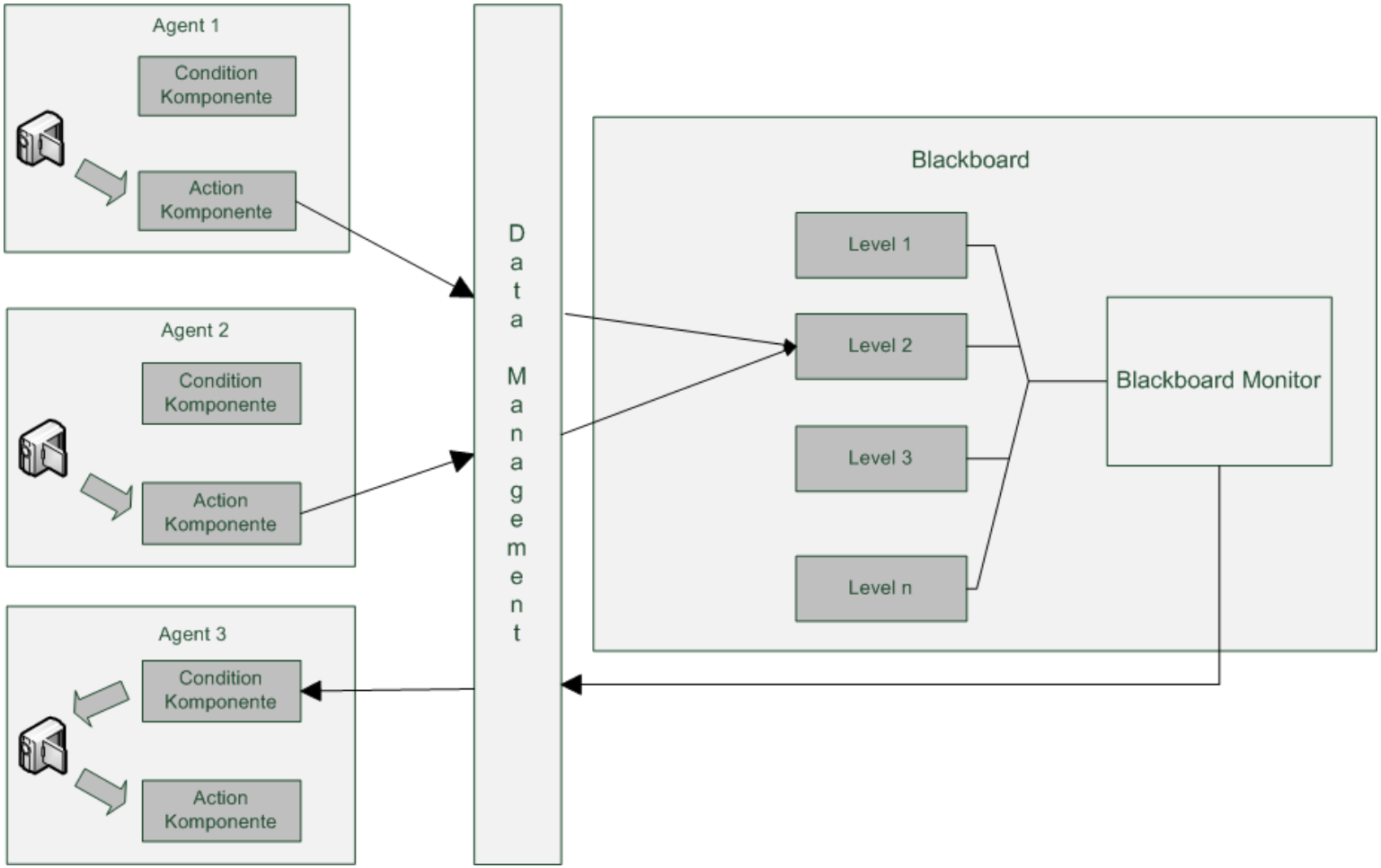
Ref

- koordinierte, synchrone Updates
- Koordination durch (Software Transactional Memory) STM
  - A - Atomic
  - C - Consistent
  - I - Isolated
  - (D - Durable)
- Arbeitsweise von (MultiVersion Concurrency Control) MVCC

Atom und Agent

- Atom
  - unkoordinierte, synchrone Updates
  - Austauschen von Inhalten
- Agent
  - asynchrone Updates
  - fire and forget

Wohin mit der Intelligenz?



Wohin mit der Intelligenz?

Server mit EventHeap

- Anzahl der Sensoren?
- Abstraktionslayer zur Erkenntnisgewinnung
- Entscheidungen treffen

Vorführung

```
0   package org.apache.commons.lang;

    ...

    public static boolean isBlank(String str) {
5     int strLen;
      if (str == null || (strLen = str.length()) == 0) {
        return true;
      }
      for (int i = 0; i < strLen; i++) {
10      if ((Character.isWhitespace(str.charAt(i)) == false)) {
          return false;
        }
      }
      return true;
15  }
```

...ab in die REPL!

Ende

Vielen Dank für die Aufmerksamkeit!

Quellen

[1] Bagwell, Phil: *Ideal Hash Trees*. 2001. – URL `http://lamp.epfl.ch/papers/idealhashtrees.pdf`. – Technical Report for the Swiss Institute of Technology Lausanne

[2] Fowler, Martin ; Rice, David ; Foemmel, Matthew ; Hieatt, Edward ; Mee, Robert ; Stanford, Randy: *Patterns of Enterprise Application Architecture*. 2005. – ISBN 0-321-12742-0

[3] Halloway, Stuart: *Programming Clojure*. 2009. – ISBN 978-1-934356-33-3

[4] Rauber, Thomas ; Rünger, Gudula: *Multicore: Parallele Programmierung*. 2008. – ISBN 978-3-540-73113-9