12. Ausblick

Proseminar „Nebenläufige Programmierung“
2010-07-21

Nebenläufige Programmierung

- Hauptmöglichkeit zur weiteren Beschleunigung von Software
- Zukunft:
  - Weg von manuellen Mechnismen
  - Hin zu Patterns

Transactional Memory

- Programmieren mit *Transaktionen*
  - Code blocks: ganz oder gar nicht
  - compareAndSet() ist wie eine Mini-Transaktion
  - Zugriff auf mehrere Variablen
  - Locking wird automatisch gehandhabt
- Siehe Kapitel im Buch

Map-Reduce

- Map-reduce (Google).
  http://en.wikipedia.org/wiki/MapReduce
- Algorithmus
  - Input: vom Typ v1
  - Produziere: map : $(k1 \times v1) \rightarrow (k2 \times v2)^*$
  - Konsolidiere: Reduce : $(k2 \times v2^*) \rightarrow v2^*$
  - Ergebnis: vom Typ v2

Fork-Join-Library

- Unterstützung für paralleles Divide-and-Conquer in Java.
- Profitiert von *Closures* in Java 7.
- „Parallelism with Fork/Join in Java 7“, R. J. Lorimer. http://www.infoq.com/news/2008/03/fork_join

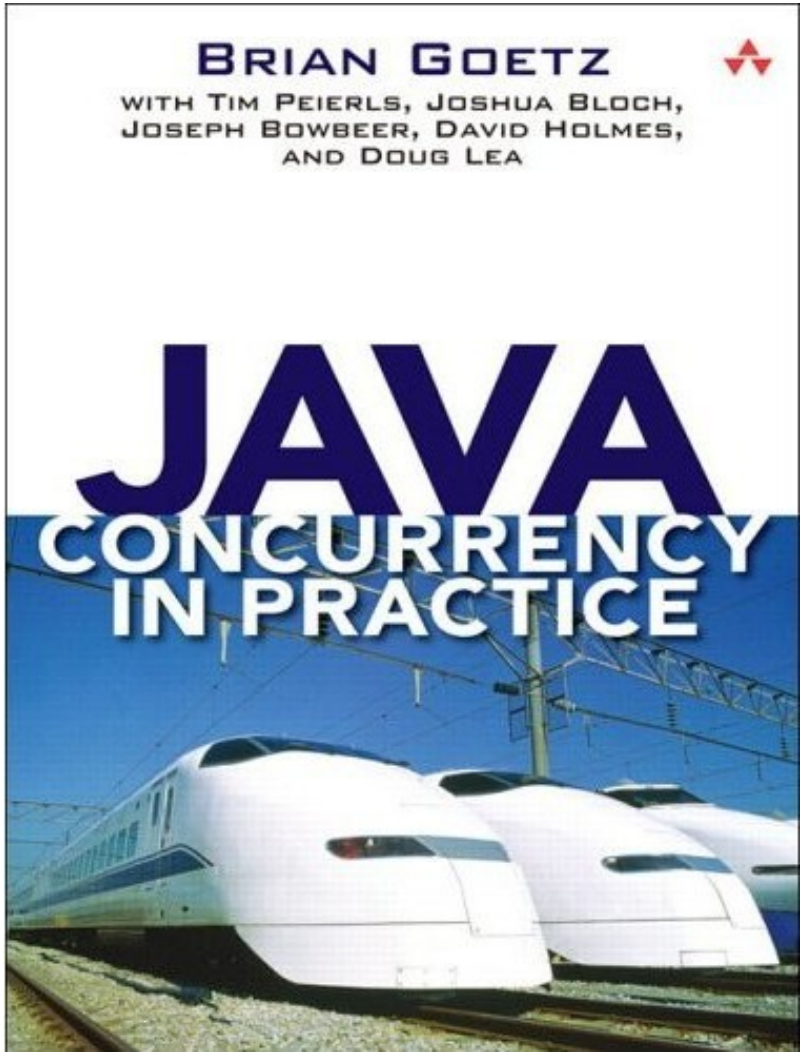
Tuple Spaces

- Für verteilte Kommunikation
- „Black Board“:
  - Produzent: Publiziere Tupel global im TS
  - Konsument: Gib Tupel-Muster an für Benachrichtigung und Extraktion
- Ursprünglich: David Gelernter, Programmiersprache „Linda“
- Java: http://java.sun.com/developer/technicalArticles/tools/JavaSpaces/

Sonstige Technologien

- Google BigTable: Verteilte, strukturierte (nicht-relationale) Datenbank.
- Apache CouchDB: Speichert JSON-Datensätze, implementiert in Erlang (s.u.)

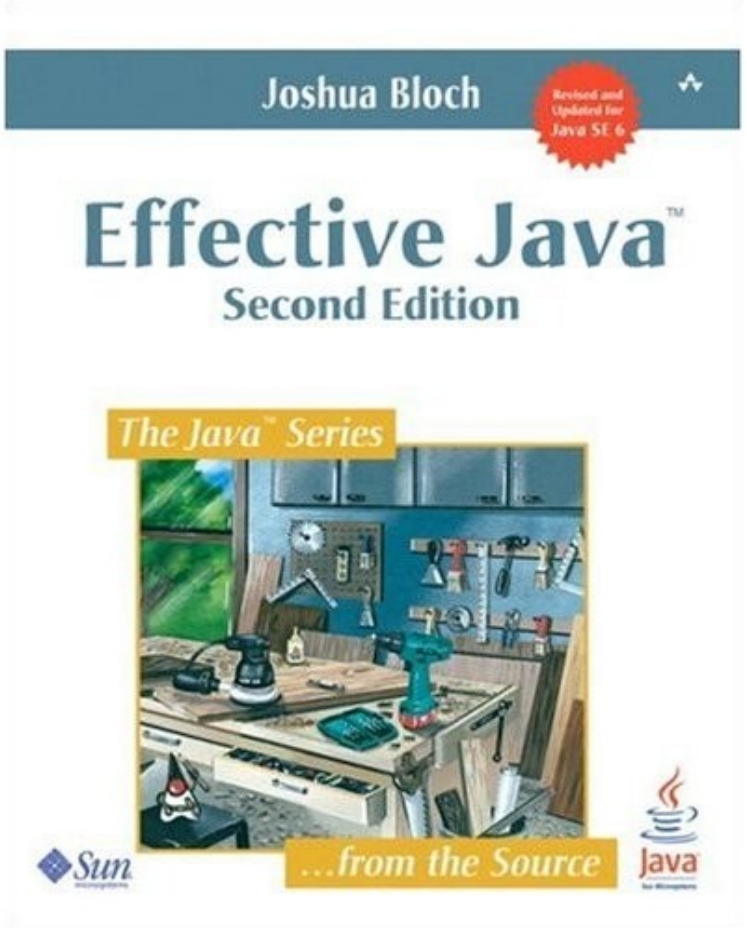
Buch: Java Concurrency in Practice



„Java Concurrency in Practice“, Brian Goetz et al.

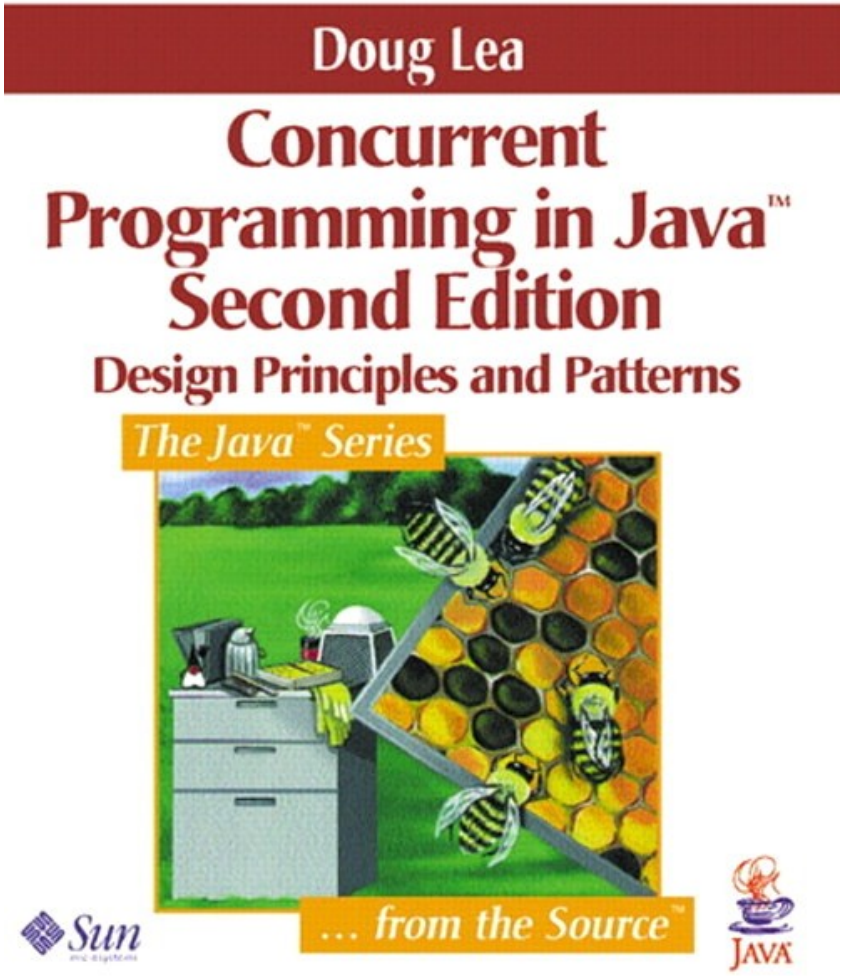
http://www.javaconcurrencyinpractice.com/

Buch: Effective Java



„Effective Java, 2nd Edition“, Joshua Bloch. http://java.sun.com/docs/books/effective/

Buch: Concurrent Programming in Java



„Concurrent Programming in Java: Design Principles and Patterns“, Doug Lea (2nd edition, 1999). http://java.sun.com/docs/books/cp/

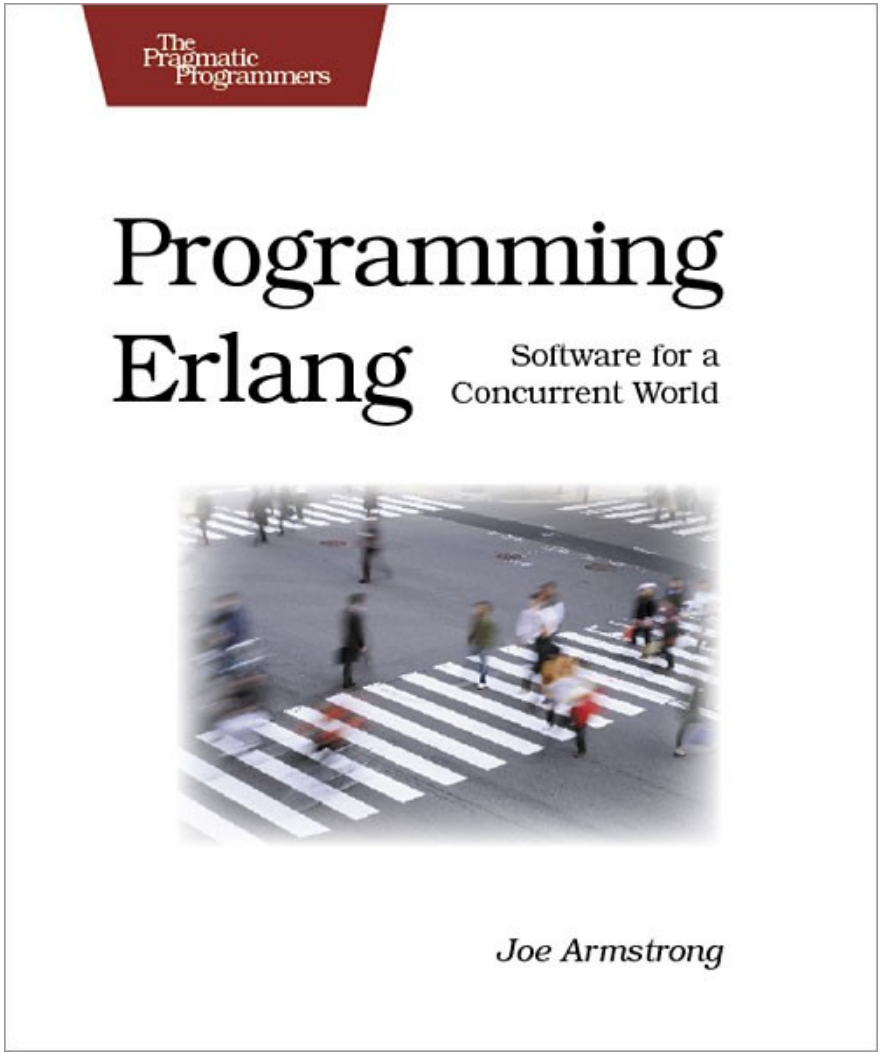
Weitere Bücher

http://www.pst.ifi.lmu.de/~rauschma/articles/software_engineering_books.html

Erlang: Die Sprache

- Für massiv-parallele Systeme (z.B. Server)
  - Fehlertolerant
  - Zur Laufzeit änderbar
- Ursprung: Ericsson, Telekommunikation
- Prozesse als zentrales Sprachkonstrukt
  - Unterhalten sich über Messages

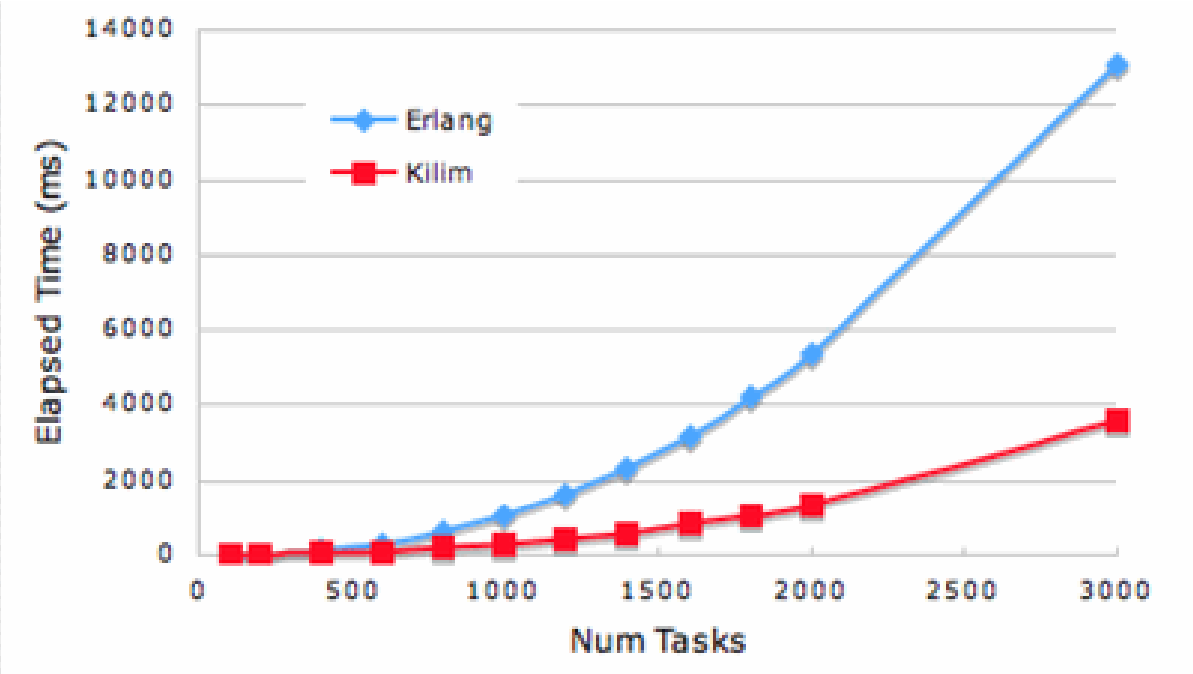
Erlang: Buch



„Programming Erlang: Software for a Concurrent World“, Joe Armstrong. http://www.pragprog.com/titles/jaerlang/programming-erlang

Erlang: Vergleichbares auf der JVM

- Scala: Funktionale Programmiersprache, die *Actors* unterstützt. http://www.scala-lang.org/
- Kilim: „Message-passing framework for Java“. http://www.malhar.net/sriram/kilim/



Clojure: Die Sprache

- http://clojure.org/
- Lisp
- Unterstützt Aktoren, Software Transactional Memory, etc.
- Gut mit Java integriert

Feedback

- Buch (vgl. Goetz)?
- Organisation des Seminars?