Clojure并发编程

刘家财
http://liujiacai.net/

2017-12-16

大纲

- 为什么需要并发
  - 并发 vs 并行
- 并发的难点
  - 传统语言(Java/C++/Python...)解决方式
  - Clojure 解决方式 State vs. Identity
- 四种引用类型：var/atom/agent/ref
- STM

为什么需要并发

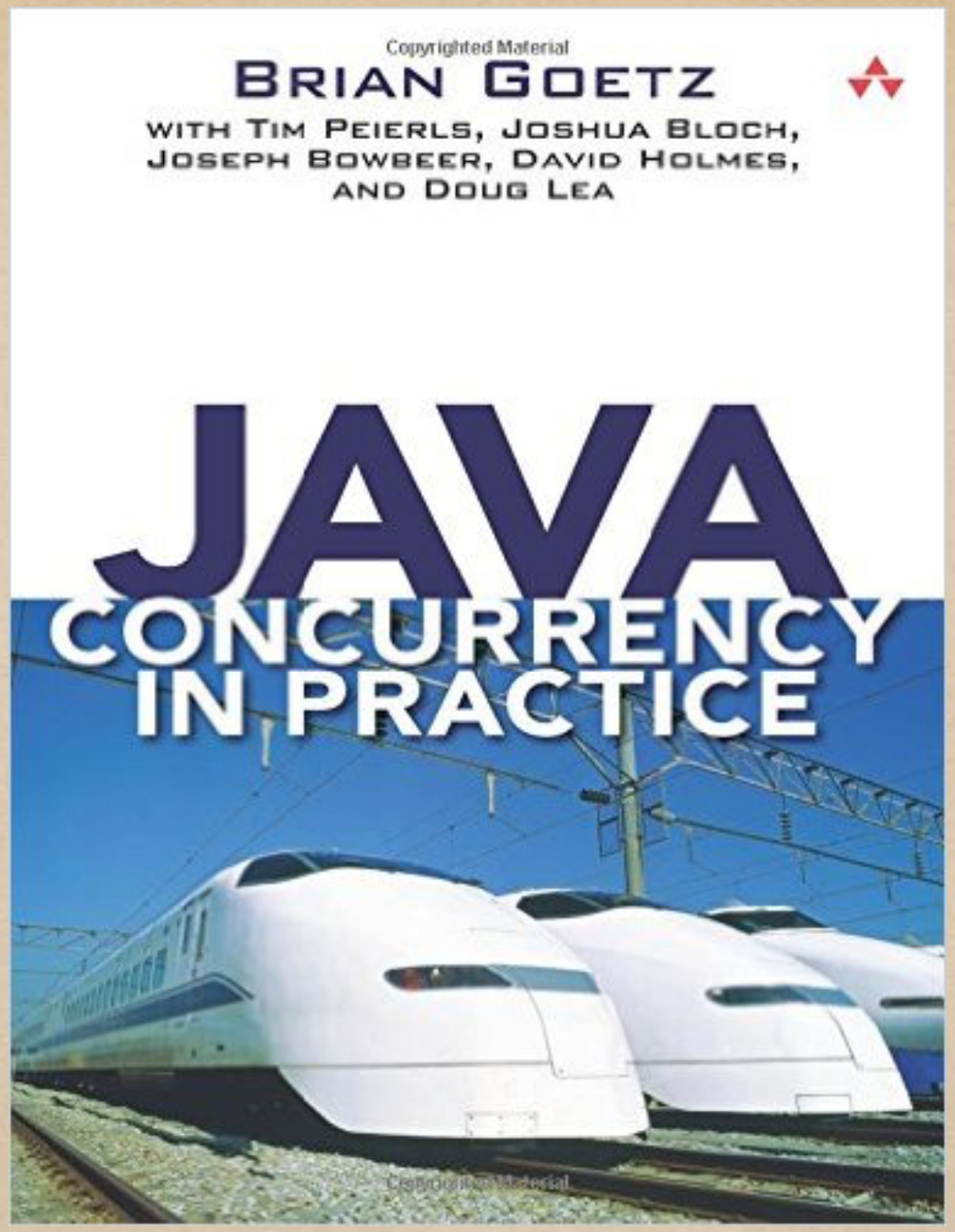
- Multicore
- Networks
- Clouds of CPUs
- Loads of users

并发 vs 并行

- Concurrency(并发) is not parallelism(并行)
- 并发一般指同时运行多个不同的任务，任务间有共享资源，任务之间相互影响，结果具有不确定性
- 并行一般指把一个任务分解成几个独立的子任务，子任务之间相互不影响，结果具有确定性
- 摘抄自 《Joy Of Clojure》第十章

传统语言中的并发

- 面向对象
  - 用对象封装，提供修改状态的方法
  - Lock/Semaphore/Barriers
- 问题
  - 死锁、饿死、Race condition….
- 问题根源
  - 直接操作共享内存
  - 没有严格区分时间概念



Java Concurrency in Practice

豆瓣评分：**9.5**

Clojure 中的并发

- 区分 State 与 Identity
- 不可变数据结构
- 提供原子类型的引用
- STM

State vs. Identity

- Identity
  - Stable logical entity
    - associated with a series of different values over time
- State
  - value currently associated with this identity
  - a true value, i.e. it never changes. 不可变数据

State vs. Identity

Identity：某人

State：幼儿/儿童/青年/中年/老年

State vs. Identity

- Clojure's answer to identity semantics
  - var
  - agent
  - atom
  - ref

State vs. Identity

	var	atom	agent	ref
Coordinated				✅
Asynchronous			✅	
Retriable		✅		✅
Thread-local	✅			

Var

- 最常用的引用类型
- 可以为 var 在一个命名空间中命名，interned
- dynamic 提供线程内的 state

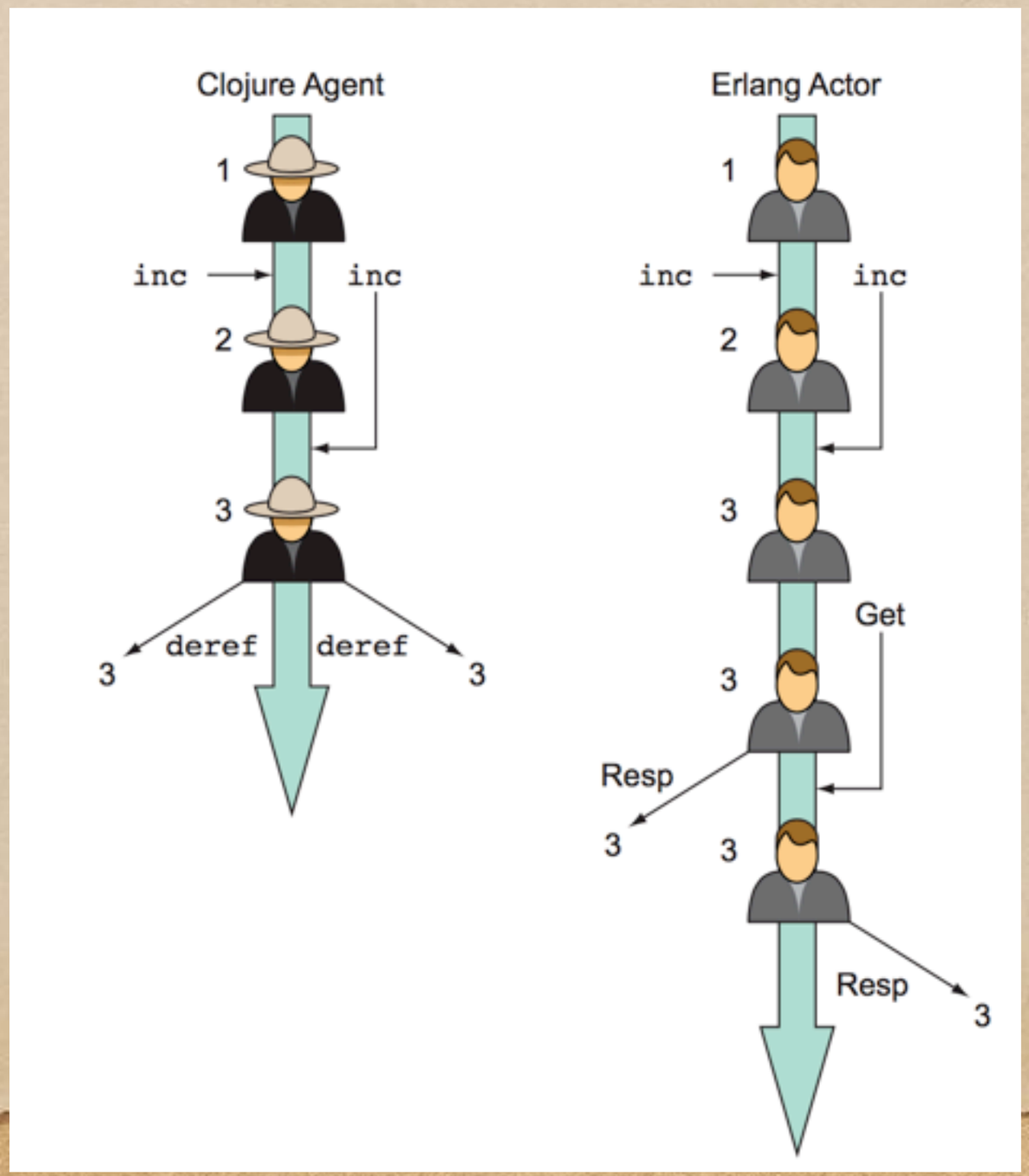
Atom

- java.util.concurrent.atomic 的封装
- 线程间共享变量
- 常用于 CAS（compare-and-swap）操作

Agent

- 与 Atom 类似，区别在于异步执行

Clojure Agent vs. Erlang Actor

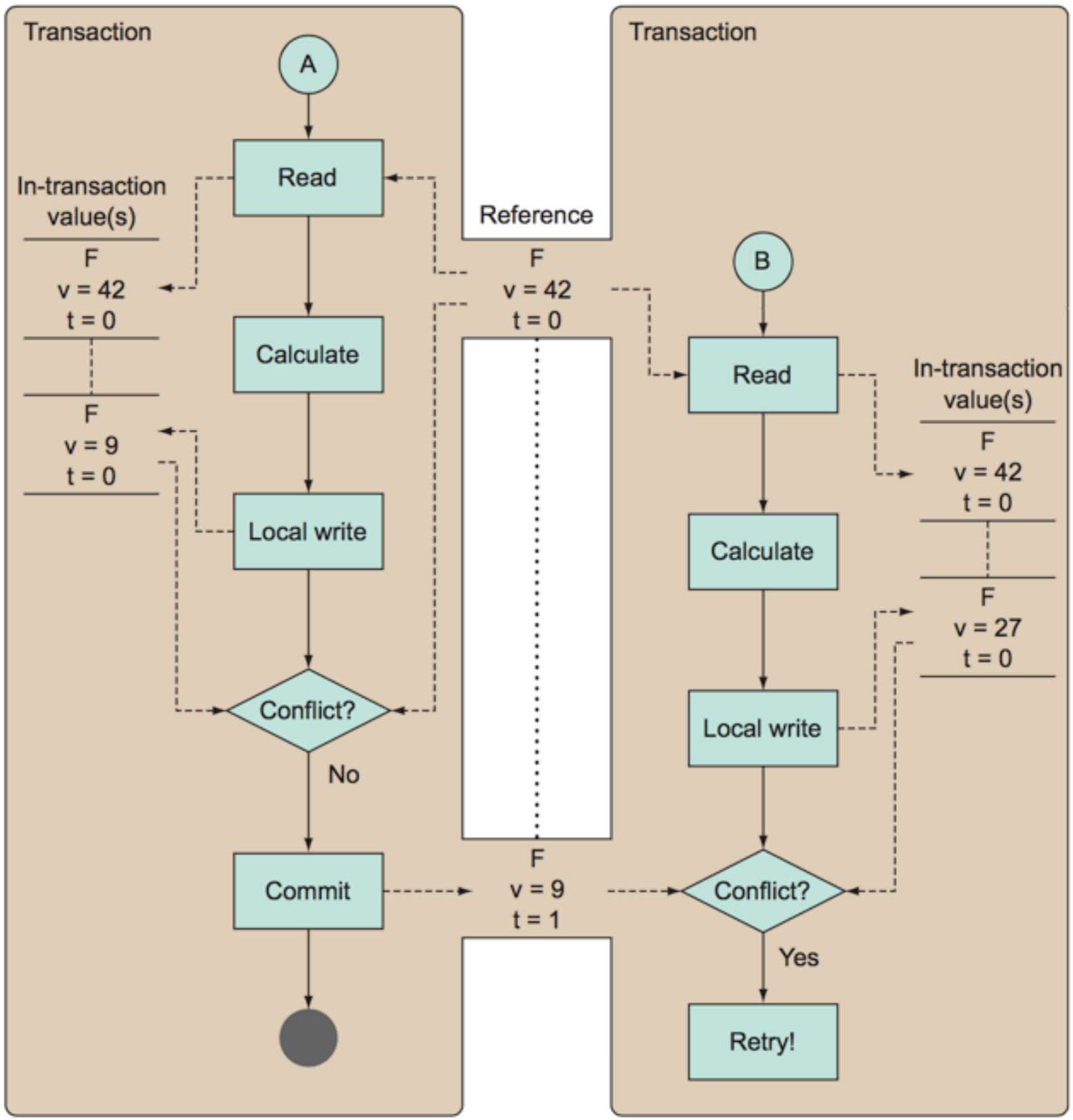


Ref

- 与 Atom 类似，区别在于具有协调性，可以同时修改多个引用类型

STM

- Software Transaction Memory
- 使用 MVCC（multiversion concurrency control）保证 snapshot 隔离


Transaction
A
Read
In-transaction value(s)
F
v = 42
t = 0
F
v = 9
t = 0
Calculate
Local write
Conflict?
No
Commit
Reference
F
v = 42
t = 0
F
v = 9
t = 1
Transaction
B
Read
In-transaction value(s)
F
v = 42
t = 0
F
v = 27
t = 0
Calculate
Local write
Conflict?
Yes
Retry!

STM 优势

- 使用简单
- No Need For Locks. 有锁程序就很可能会死锁，比如异常时相应锁没释放
- ACI
  - atomicity, consistency, and isolation
  - 没有D durability，在内存中

STM 劣势

- WRITE SKEW 事务依赖一个 ref 但是不进行写操作(使用ensure)

- 事务不能有副作用（IO）
- Live Lock 一个事务不断重试，一般是由于事务逻辑太多造成

More

- future/promise/pmap/pvalues/pcalls
- 1.5 引进的 reducer/fold.
- core.async
- http://clojure-doc.org/articles/language/concurrency_and_parallelism.html
- https://docs.oracle.com/javase/8/docs/technotes/guides/concurrency/changes8.html
- Clojure STM 实现剖析
- 本次课程演示代码

Thank You.

群名称：SICP读书群

群　号：119845407

公众号