Introduction To Multithreaded Programming With GPars

Concurrent programing in Groovy by examples

Object
Partners
Inc.

Jon DeJong
Chief Software Technologist
@jondejong

Who am I?

Jon DeJong
Object Partners
Chief Software Technologist

Slinging code since 1999 (15!) Years
Slinging code into production on the JVM since 2002
Groovy in production since 2007
With OPI since 2006 (8!) Years

What is OPI?

- Flat consulting firm with 100(ish) consultants in 6 states
- Historically on the JVM
- GR8 Tech, Open Source
- Changing with the world (JS, Mobile, Big Data, Microservices, DevOps)

https://github.com/jondejong/GR8ConfUS2014-GPars

All That Good Demo Code

What is GPars?

GPars solves problems

- Simple solutions for common threading issues
- Frameworks for designing solutions to complex problems

Five Problem Types

working examples for each

Five Problem Types

- Repetitive independent processes

Five Problem Types

- Repetitive independent processes
- Asynchronous method invocation

Five Problem Types

- Repetitive independent processes
- Asynchronous method invocation
- Serial Algorithm Optimization

Five Problem Types

- Repetitive independent processes
- Asynchronous method invocation
- Serial Algorithm Optimization
- Shared Mutual State

Five Problem Types

- Repetitive independent processes
- Asynchronous method invocation
- Serial Algorithm Optimization
- Shared Mutual State
- Greenfield Concurrent System Designed

Repetitive Independent Processes

- Parallel collection methods
- Map-Reduce

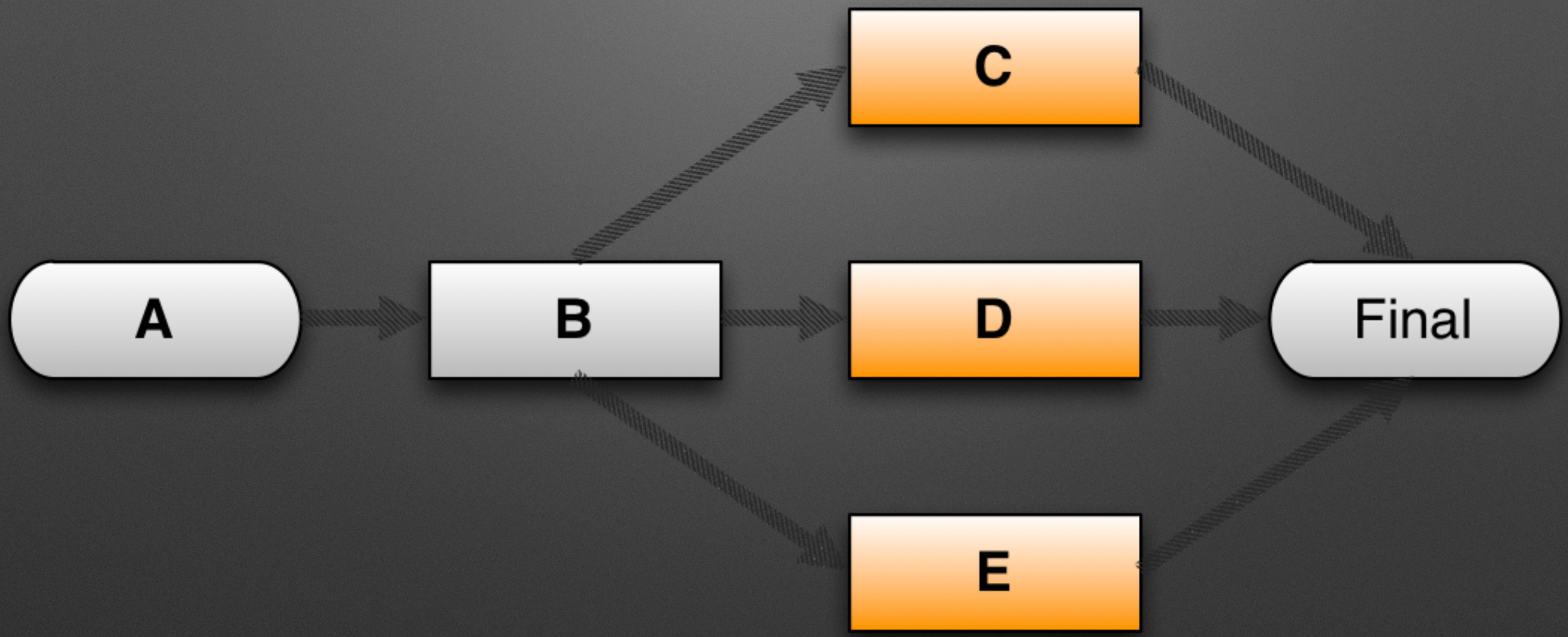
Code

Asynchronous Method Invocation

- Do not care about the result
- Do care about the result, but don’t care when
- Do care about the result, and will block for it in the future

Code

Serial Algorithm Optimization



Serial Algorithm Optimization

- Chain asynchronous call with Promises
- Dataflow variables

Code

Shared Mutual State

Thy not to do this…

A lot of the concurrency problems disappear when you eliminate the need for Shared Mutable State with your architecture. Indeed, concepts like actors, CSP or dataflow concurrency avoid or isolate mutable state completely. In some cases, however, sharing mutable data is either inevitable or makes the design more natural and understandable

- Agent

Agents hide the data and protect it from direct access. Clients can only send commands (functions) to the agent. The commands will be serialized and processed against the data one-by-one in turn. With the commands being executed serially the commands do not need to care about concurrency and can assume the data is all theirs when run.

Code

Greenfield Design

- Actors
- CSP

Actors

Actors allow for a message passing-based concurrency model: programs are collections of independent active objects that exchange messages and have no mutable shared state. Actors can help developers avoid issues such as deadlock, live-lock and starvation.

Actors

Actors always guarantee that at most one thread processes the actor's body at any one time and also, under the covers, that the memory gets synchronized each time a thread gets assigned to an actor so the actor's state can be safely modified by code in the body without any other extra (synchronization or locking) effort .

Actors

- afterStart()
- act()
- terminate()
- loop({})
- react({})

Communicating Sequential Processes

The CSP (Communicating Sequential Processes) abstraction builds on independent composable processes, which exchange messages in a synchronous manner. GPars leverages the JCSP library developed at the University of Kent, UK.

Communicating Sequential Processes

- CSP Processes
- CSP Channels
- CSP Alternative

Communicating Sequential Processes

- CSP Processes

```
def t = group.task {
    println "I am a process"
}
t.join()
```

- CSP Channels
- CSP Alternative

Communicating Sequential Processes

- CSP Processes
- CSP Channels

```
final def buffer = new DataflowQueue()
…
buffer << value
…
println buffer.val
```

- CSP Alternative

Communicating Sequential Processes

- CSP Processes
- CSP Channels
- CSP Alternative

To introduce non-determinist GPars offers the Select class with its select and prioritySelect methods

Code

Software Transactional Memory

Adds transactional access to in memory data.

Thank you

@jondejong

https://github.com/jondejong/GR8ConfUS2014-GPars