# Having fun with Kotlin coroutines

A first tour of concurrency models in Kotlin

Alessandro Candolini

June 27, 2018

# Agenda

1. We live in a concurrent world
2. Blocking vs non-blocking
3. Demystifying coroutines
4. Coroutines-powered concurrency models

# We live in a concurrent world

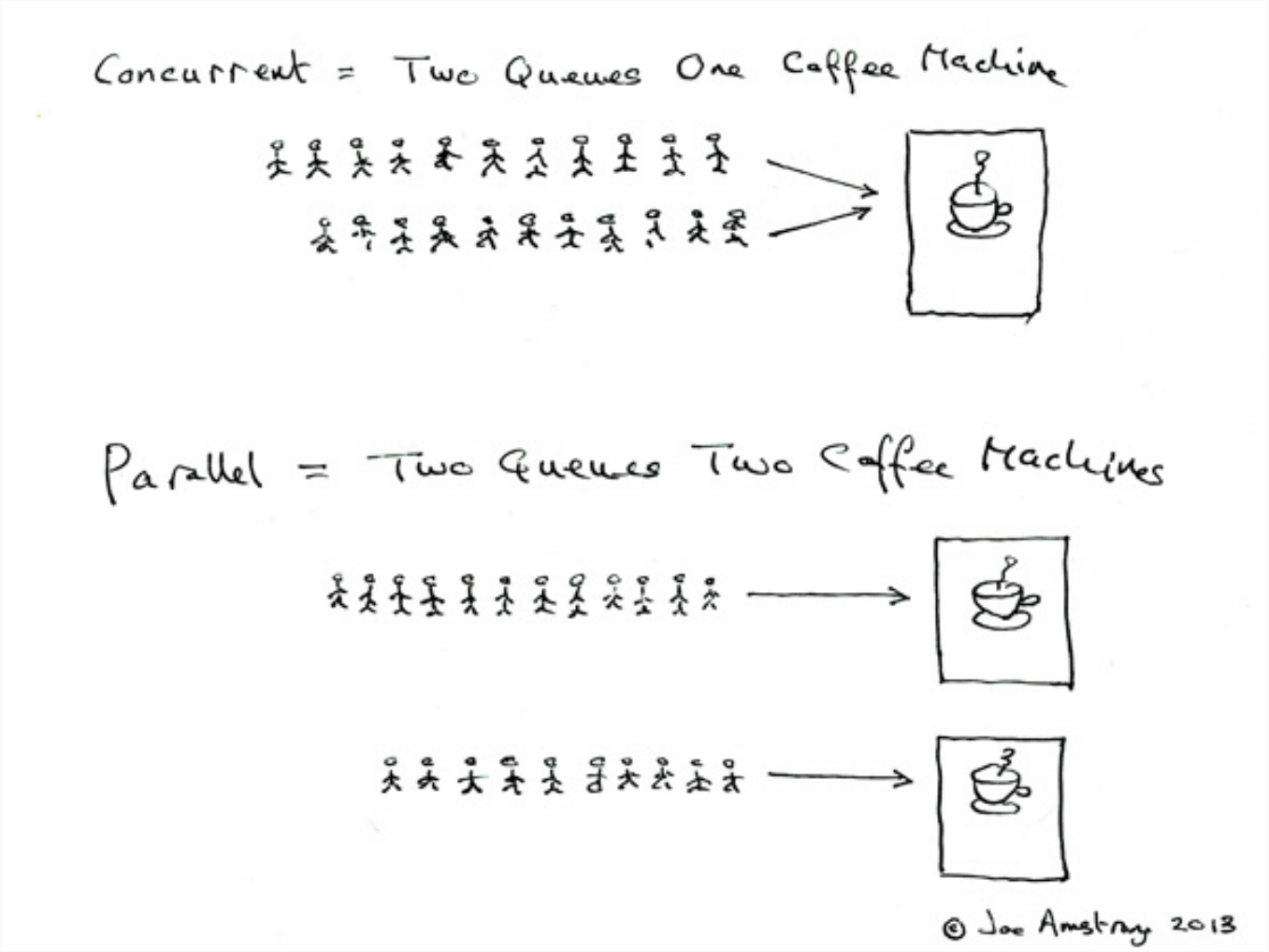



**Figure 1:** `https://joearms.github.io/published/2013-04-05-concurrent-and-parallel-programming.html`

Is concurrency relevant for mobile development?

- IO (e. g., network, etc)
- sensors (e. g., gps, etc)
- UI events
- platform lifecycle

Is concurrency relevant for mobile development?

- IO (e. g., network, etc)
- sensors (e. g., gps, etc)
- UI events
- platform lifecycle

Is concurrency relevant for mobile development?

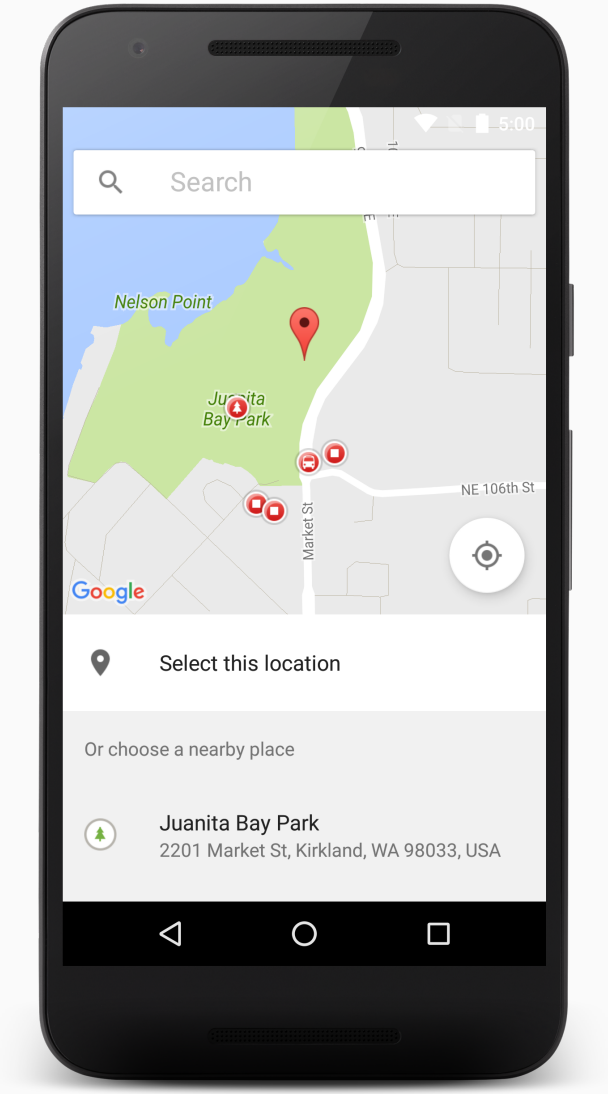
- IO (e. g., network, etc)
- sensors (e. g., gps, etc)
- UI events
- platform lifecycle

Is concurrency relevant for mobile development?

- IO (e. g., network, etc)
- sensors (e. g., gps, etc)
- UI events
- platform lifecycle

Is concurrency relevant for mobile development?

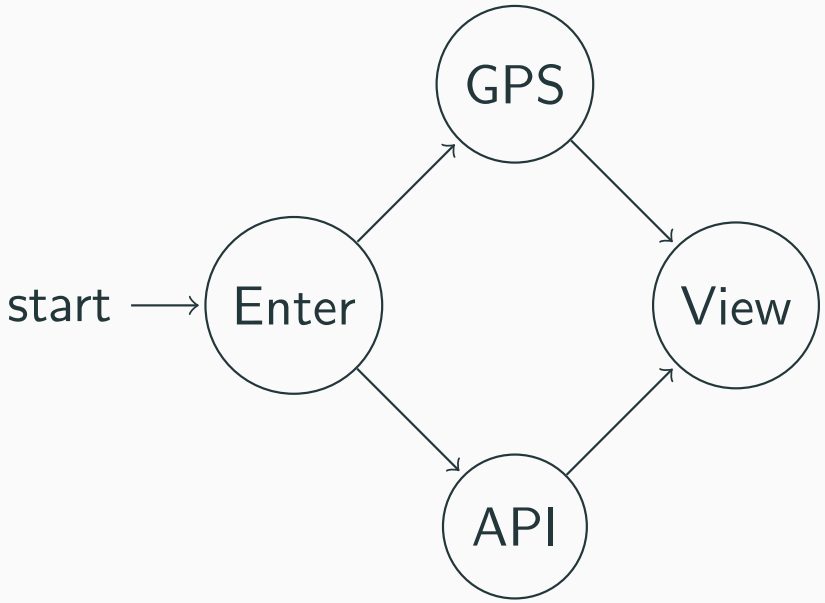
- IO (e. g., network, etc)
- sensors (e. g., gps, etc)
- UI events
- platform lifecycle

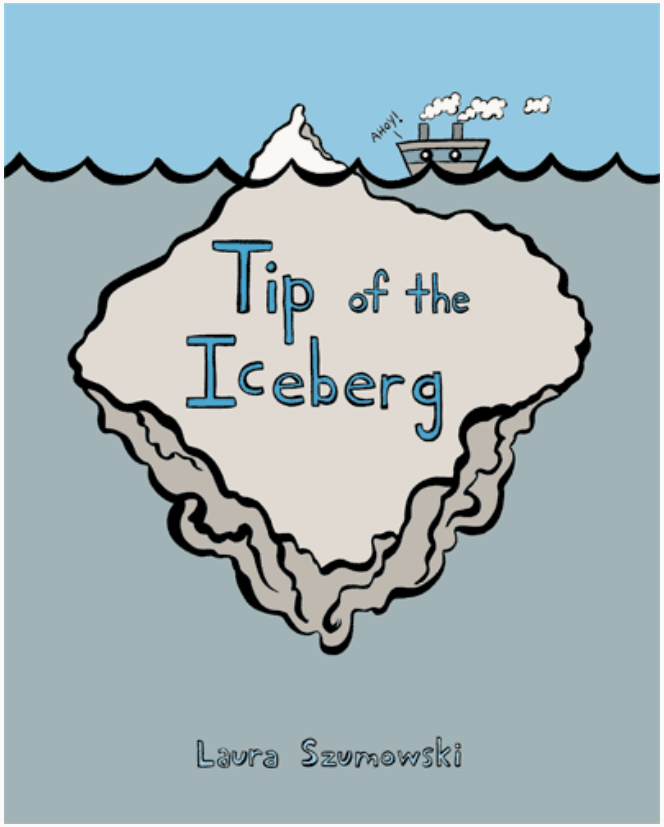



acceptance criteria:

- search by current location
- search by location name

How it looks like: simple state machine (simplified)






- GPS connection is asynchronous and might fail
- Network requests are sent in parallel at each letter
- Debouncing
- Latest takes priority (avoid overriding with old data)
- Android lifecycle, etc

"Concurrency is the composition of independently executing processes, typically functions, but they don't have to be." "Parallelism is the simultaneous execution of multiple things, possibly related, possibly not."

Concurrency is a way to structure the computer programs. Parallelism is *not* the goal of concurrency

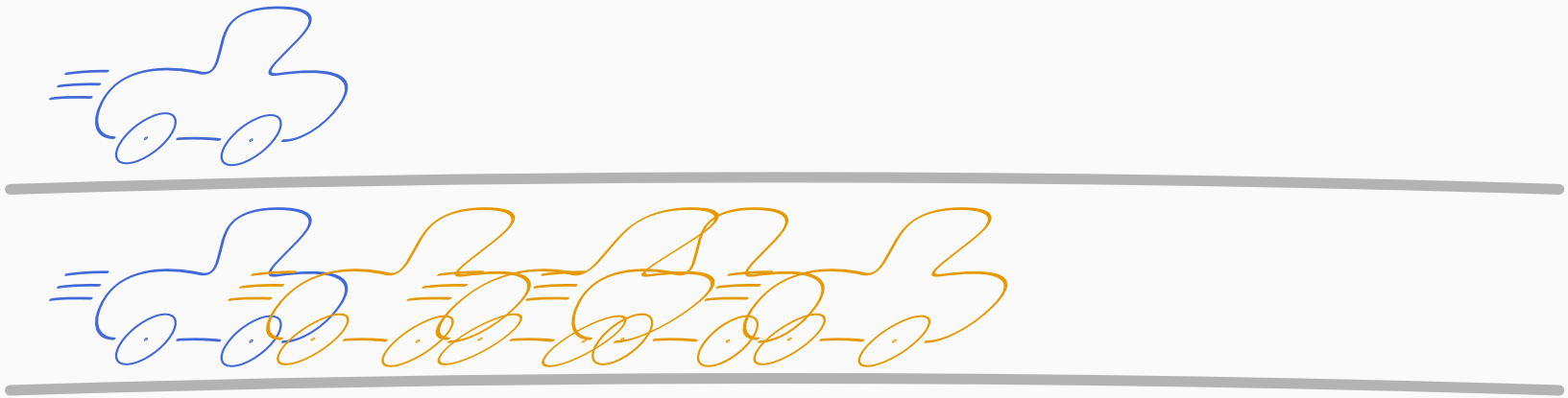
Rob Pike

Tony Horne seminal paper

Java definition of concurrency and Leslie Lamport seminal paper

Traditional picture of concurrency vs parallelism: coffe machine

Better reppresentation: driving in the desert vs driving in London



The deep problem: *communication*

Communicating, orchestrating independent processing. Software development as a *dialog* between parts.

We will just sketch the surface of this deeper problem by focusing only on a smaller technical problem in this talk: *blocking* and *non wasting unnecessary resources when waiting*

# Blocking vs non-blocking

Test

# Demystifying coroutines

What rae

# Coroutines-powered concurrency models

- CSP (aka, channels)
- actors

# Questions?