inloco

Building resiliency with Go’s concurrency primitives

Recife

2017

AGENDA

1. Language Domain
2. Concurrency

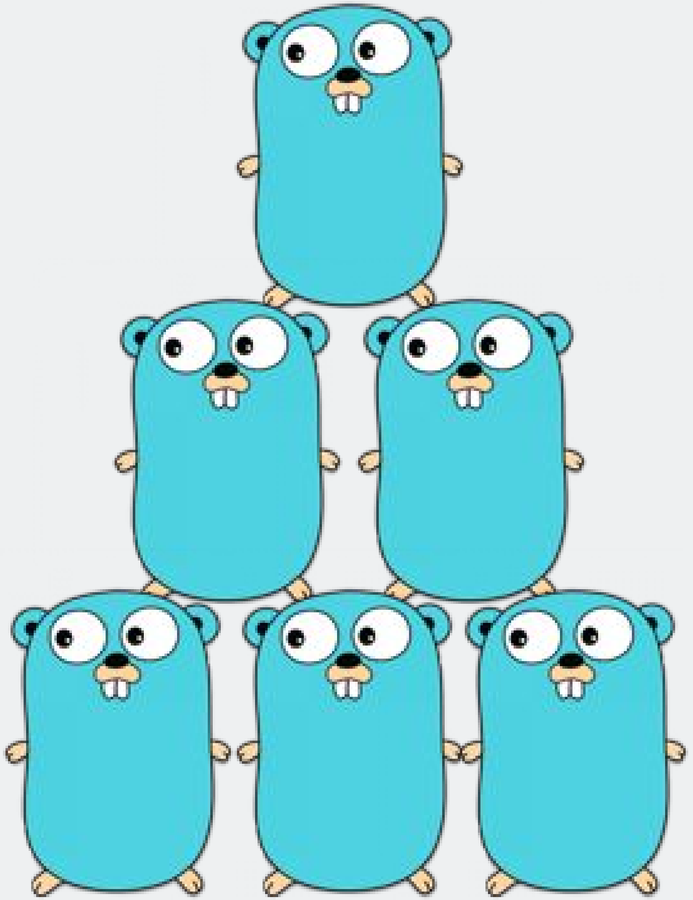
1 Language Domain

IT industry has moved towards

Scaling Out,

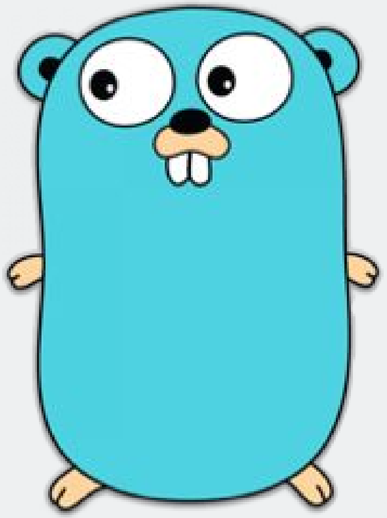
rather than

Scaling Up.



Monolithic Architecture
Database as Integrator
Defect free/Fragile

Federated Architecture
API as connectors
Antifragile



2 Concurrency

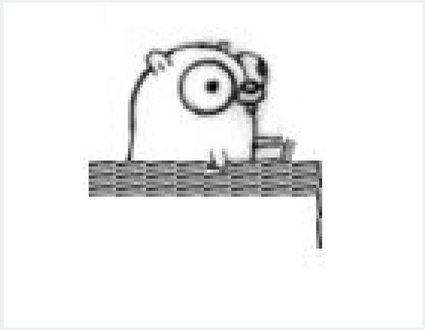
Concurrency vs. Parallelism

Concurrency is about *dealing with* many things at once.

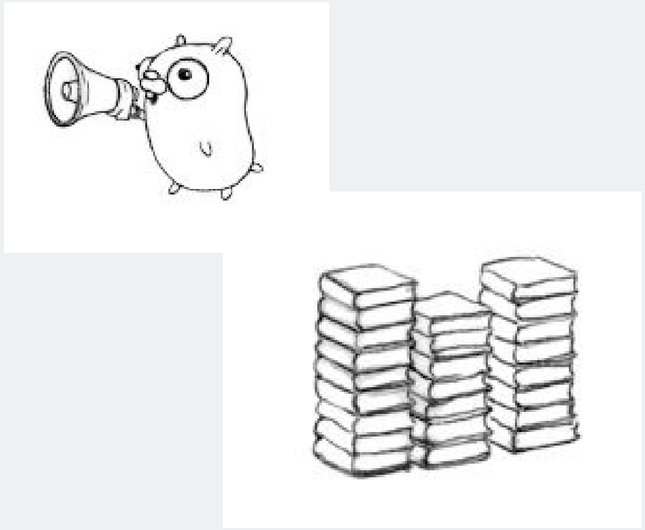
Parallelism is about *doing* many things at once.

Concurrent is the opposite of *sequential*.
Parallel is the opposite of *serial*.

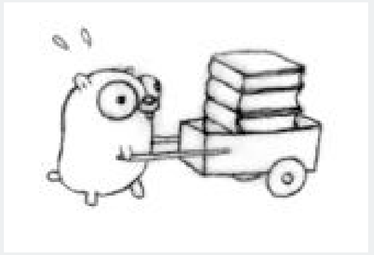
An example of a Concurrent model



Lazy Observer



Queue of Jobs



Job Incinerator

Goroutines

A goroutine is a lightweight thread managed by Go runtime.

Goroutines share the same address space.

Goroutines

A given application may have hundreds of thousands of goroutines running.

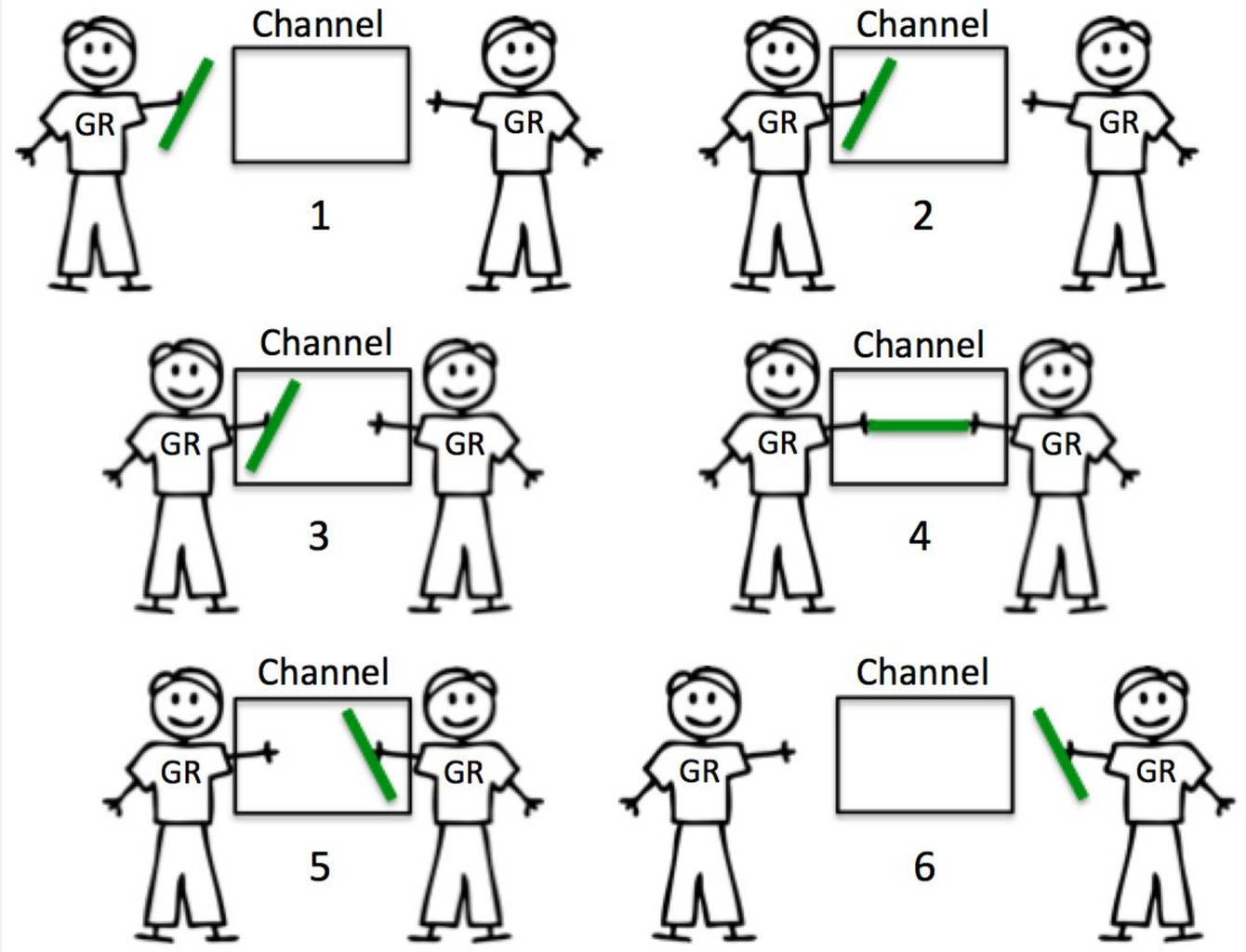
Goroutines start cheap, with a stack of 8K, and will grow as necessary.

Goroutines are mutexed automatically into threads.

- The GOMAXPROCS variable limits the number of operating system threads that can execute user-level Go code simultaneously.

https://play.golang.org/p/6PHXHha_Uv

Channels

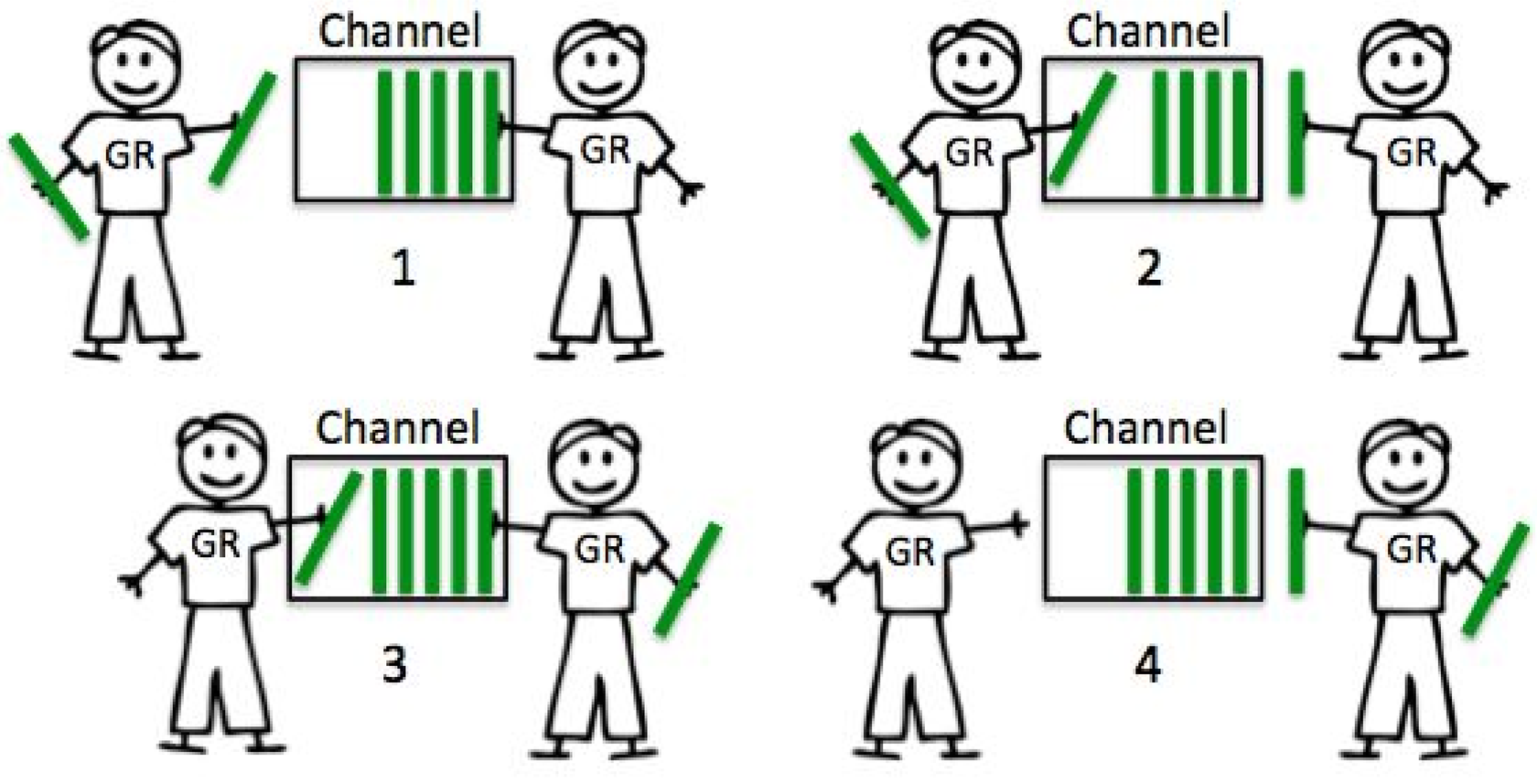

Channel
GR
GR
1
Channel
GR
GR
2
Channel
GR
GR
3
Channel
GR
GR
4
Channel
GR
GR
5
Channel
GR
GR
6

Channels

Channels are a typed conduit through which you can send and receive values with the channel operator, <-.

By default, sends and receives block until the other side is ready.
https://play.golang.org/p/NU4-hq4R51

Buffered Channels



Buffered Channels

Channels can be buffered.

Sends to a buffered channel block only when the buffer is full.
Receives block when the buffer is empty.

https://play.golang.org/p/qh_YznYAER

Select

Select will block until one of the given channels can return.

https://play.golang.org/p/_Bez50lCsT

Live Coding

Summary

Using Go primitives we came from:

Slow
Sequential
Failure Sensitive

To a program that is

Fast
Concurrent
Robust

Thank you for coming!

https://github.com/rhnasc/deepdive-livecoding