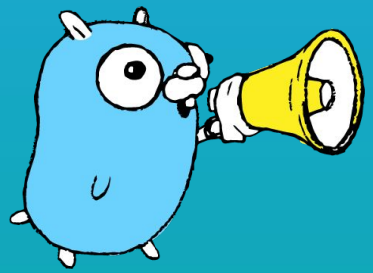
SHEFFIELD, UK - 4th APR 2019

EVENT-DRIVEN PROGRAMMING

JACK ADAMS

U Account

jack.adams@uaccount.uk



ABOUT ME

- Developer for ~10 years
- Using Go sporadically for ~6 months
- LAST MINUTE - I WROTE THESE SLIDES AND CODE TODAY

- Get in touch: jack.adams@uaccount.uk

EVENT-DRIVEN PROGRAMMING

Some programs are inherently event-driven (think web-server, webpage or UIs) but that's not the end of it

Go itself IS event-driven, GoRoutines are scheduled based on readiness of data, Go networking stack uses epoll

When thinking about true concurrency your code MUST be event-driven, the "event" signals the availability of data

CONCURRENCY SUMMARY

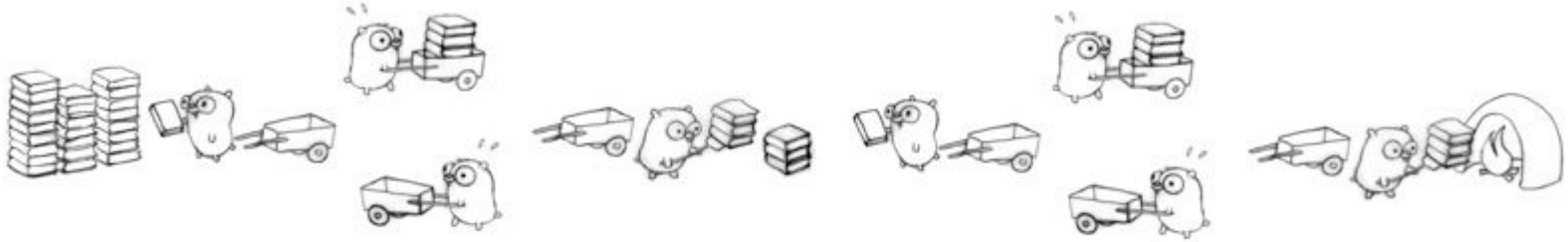
LET THE SYSTEM HANDLE THE PARALLELISM

THINK SMALL

GOROUTINES ARE CHEAP, MAKE MORE OF THEM

CHANNELS PAIRED WITH SELECT ARE YOUR FRIEND

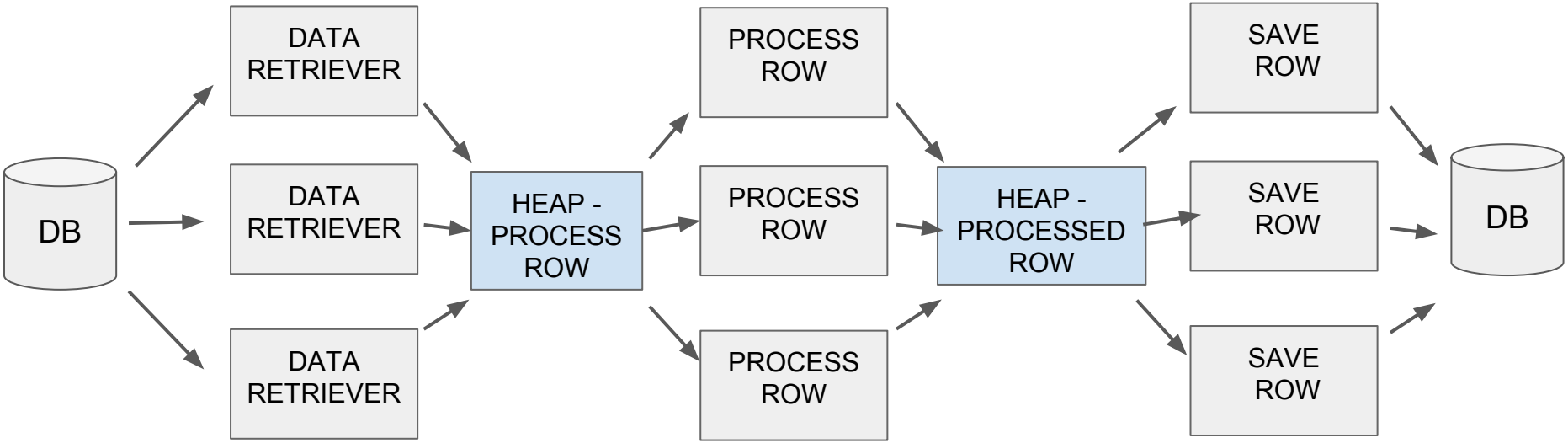
BY EXAMPLE - GOPHERS - CONCURRENCY!



- Job is broken into many smaller tasks, allowing more gophers to work at once.
- None of these gophers are waiting on each other

With thanks to Rob Pike

BY EXAMPLE - CONCURRENCY



- GET, PROCESS and SAVE can all run concurrently
- Bottlenecks are an infrastructure issue, the program is scalable by design

DO WE NEED MUTEXES?

Sometimes, unfortunately

Can you pass the object through your channels instead?

If you need to use a mutex you might not be managing your concurrency properly

GOROUTINES

```
go func() {
    time.Sleep(deltaT)
    fmt.Println(“DONE!”)
}()

// don’t run this!
for {
    go func() {
        fmt.Println(“HELLO!”)
    }
}
```

NOT THREADS!

But think of them like much cheaper threads

Like using & in shell
- go do this, but let me carry on

CONCURRENCY IN GO

CHANNELS

```
timerChan := make(chan time.Time)

go func() {
    time.Sleep(deltaT)
    timerChan <- time.Now()
}()

go func() {
    Time := <-timerChan
}
```

Communication between goroutines

Allow data to be passed between workers

SELECT

```
for {
    select {

    case v := <-ch1:
        fmt.Println("channel 1 sends", v)

    case v := <-ch2:
        fmt.Println("channel 2 sends", v)

    default: // optional
        fmt.Println("neither ready")
    }
}
```

Select an appropriate channel based on its ability to communicate

Think ‘load balancer’ when coupled with a heap of ‘channels’?

SOME CODE

1. https://play.golang.org/p/mw15CvfV6R2
2. https://play.golang.org/p/EnX58BekQfa
3. https://play.golang.org/p/uErIMgPlPUt
4. https://play.golang.org/p/1QxhpcpsflU

If you'd like to see a non-string based example give me an email at:
jack.adams@uaccount.uk