

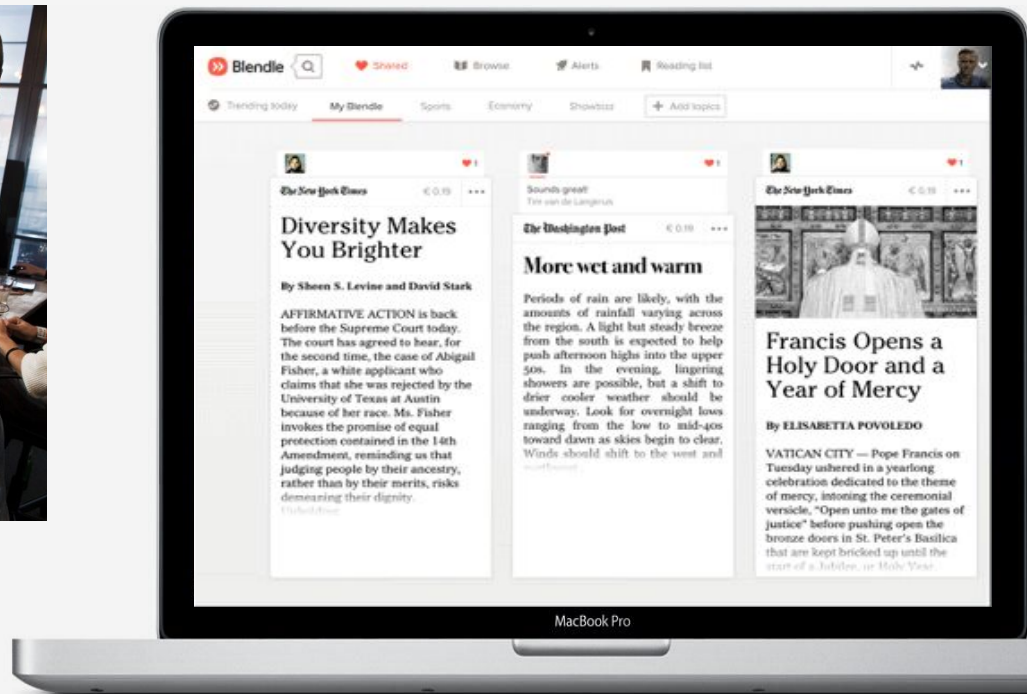
Golang at Blendle

All the e-mails! Ten minutes flat!

Table of content

- **About Blendle & Our Newsletter**
- **Enter Golang**
- **Our Challenges**
- **Go in production**
- **Tips 'n Tricks**

About Blendle



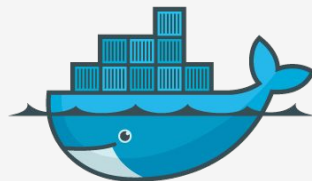
Blendle Stack (quick overview)

- **Core Backend:** Ruby & Sinatra/Webmachine
- **Frontend:** Single Page JS, Native Apps
- **Workflow:** Bash/Ruby
- **Microservices**
- **Cucumber BDD**
- **Infra:** AWS, Docker & Kubernetes



JavaScript

Cucumber



docker

The Blendle Newsletter

- Editorial Powered, Daily Curated
- All personalized e-mail:
We sort articles for your reading profile

Goedemorgen Koen,

Dit is onze selectie van vandaag.

de Volkskrant

Verrassend: een positief verhaal over Donald Trump: 'Eigenlijk is hij een nieuwe Roosevelt'



Ja, het staat er echt in *de Volkskrant*: Donald Trump heeft wél een politieke agenda die verder gaat dan xenofobie en machthonger. Sterker nog: **op veel gebieden is hij een halve Democraat.**

NRC

Op bezoek bij Zalando, waar achter alle schoenen en kleren een enorm (hip) techbedrijf schuilgaat

Iedereen gaat casual gekleed naar kantoor (dus ook de bazen), werknemers mogen hun honden en kinderen meenemen en elk jaar is er in december een 'Hack Week' waarin techneuten los mogen gaan. **Nee, dit is niet een bedrijf in Silicon Valley, maar webwinkel Zalando.**

REVU

Features of our Mailer (Brains)


- **Template Rendering**
- **Sending via Amazon Simple Email Service (SES)**
- **Selecting ± 10 articles from a curated list of around 60**
- **Some Business Logics:**
 - Allow for *Must Reads* by our staff
 - Avoid repetition of the same publisher
 - List articles for a user's subscription first



Enter Golang

Why did we choose Golang?

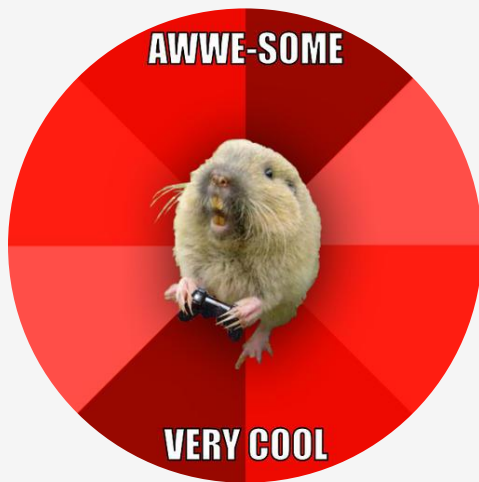
Performance...

- **Most Ruby/Sidekiq examples reach a throughput of ± 40 messages per second. Which would make us run for 83 hours.**
- **We require worst of both worlds:**
 - Sorting article == CPU intensive
 - Sending mails == I/O intensive
- **Celluloid or Eventmachine: Just** 

Then why Go?

- We needed high throughput in both sorting, rendering and sending
- Golang is a very small language, thus easy to learn and read

Then why Go?



- **It's just cool!** *(which is always a good measure for choosing tech)*
 - <insert all reasons why go lang is cool here>

Our Challenges

Scale

Emails on weekday mornings

250.000

On thursday afternoon and the sunday overview we send an additional

350.000

Time



And with the normal digest newsletter time is an important factor.

- We can't send it earlier due to the editorial part
- The longer it takes, the more people are already at work

Concurrency: First Attempt

- **Just go-routine all the things!**
 - o One go-routine for each user, then just render and send the e-mail.
- **Worked, on the Golang side of thing.**
But we had major issues with available sockets on the operating system.
- **So...**



Concurrency: Second Try

- **Basic queue system:**
 - Push users to a channel
 - Have a bunch of workers ready
 - Pass users to workers.
- **Worked for a while, but then got slower as our sorting/render speed increased**

```
type Dispatcher struct {  
    WorkerPool    chan chan Job  
    ResultChannel chan mailerResult  
    JobQueue      chan Job  
  
    maxWorkers int  
}
```

```
func (d *Dispatcher) dispatch() {  
    for {  
        select {  
            case job := <-d.JobQueue:  
                go func(job Job) {  
                    jobChannel := <-d.WorkerPool  
                    jobChannel <- job  
                }(job)  
        }  
    }  
}
```


Concurrency: Current Version

- **We separated the render queue from the sending queue.**
 - One pool of workers sort, render and prepare the SES http request (this still uses the old system)
 - A new system is in place to handle the sending and retrying of the request.
 - Dynamically growing worker count

06:59:38 mailer.go:115: info: mailer finalized, sent 248216 mails in 6m34.341153505s

06:07:03 mailer.go:115: info: mailer finalized, sent 342561 mails in 7m11.818600286s

Go in Production

CI + Deployment

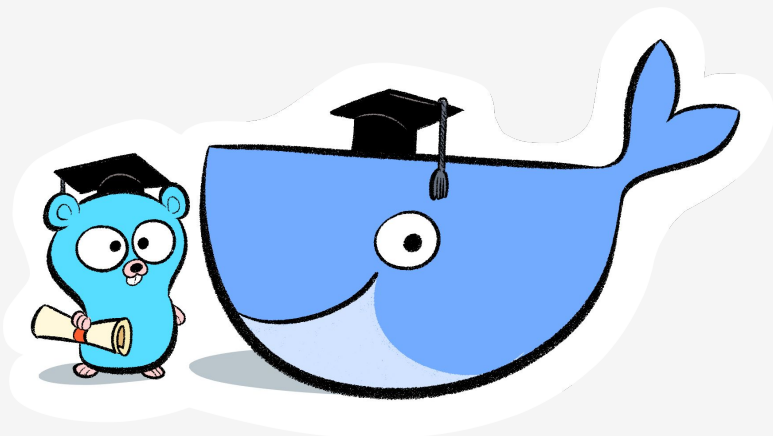
Testing

- The Go ``testing`` package for unit testing and benchmarking code.
- **Cucumber BDD** for our integration and feature tests.
 - Using Ruby + Aruba, running our Go code as commands
 - Works well with other (non-golang) colleagues at Blendle



Docker

- **Our mailer is ran in Docker containers.**
 - Sometimes multiple containers when do AB tests.
 - Fired from a Ruby mailtool
 - Progress tracked in Redis



Wercker » Quay » `docker pull`

Continuous Deployment

- Difficult, seeing we only run production once a day
- Hourly Test
- When changing the mailing process code: Manually Benchmark

Tips 'n Tricks

Some neat tricks we use

Profiling

```
```\n\npprof.StartCPUProfile(file)\ndefer pprof.StopCPUProfile()\n```\n
```

<https://github.com/uber/go-torch>



# Profiling

The visualization displays a heatmap of function calls over time. The rows are labeled with function names and file paths, such as 'fmt.Sprintf', 'runtime.concatstrings', and 'github.com/blendle/brains/vendor/github.com/aws/aws-sdk-go/aws/corehandlers.(\*validator).va..'. The columns represent different time intervals or iterations. The colors range from yellow to red, indicating the frequency or intensity of the calls. A large red 'X' is overlaid on the heatmap, spanning most of the visualization.

# Runtime Tweaks

<https://golang.org/pkg/runtime/>

```
`GOGC=400`
```

```
`CGO_ENABLED=0` (if possible)
```

```
`-ldflags "-s -extldflags -static" -a -tags netgo`
```

