Why every programmer should learn Go

Mateusz Dymiński
Nokia

github.com/mateuszdyminski/whygo (github.com/mateuszdyminski/whygo)

Whoami

Mateusz Dymiński:

- Software Developer at Nokia
- 5+ Golang exp
- One of organizer GoWroc - Golang Wroclaw Meetup (https://www.meetup.com/GoWroc)
- Page: mateuszdyminski.com (https://mateuszdyminski.com)
- Github: github.com/mateuszdyminski (http://github.com/mateuszdyminski)
- Twitter: @m_dyminski (http://twitter.com/m_dyminski)
- LinkedIn: linkedin.com/in/mdyminski (http://linkedin.com/in/mdyminski)

Agenda

1. What's Go?
2. Why Google created Golang
3. Go characteristics
4. Go vs Java
5. Why Go is awesome
6. Summary

What's Go?

"Go is an open source programming language that makes it easy to build simple, reliable, and efficient software."

Characteristics:
- Compiled
- Static typing
- Concurrency
- Simple
- Productive

golang.org (http://golang.org)

History

Conceptual work started in late 2007.

- Robert Griesemer, Rob Pike and Ken Thompson.
- Ian Lance Taylor and Russ Cox.

Open sourced in 2009.

Version 1.0 published in 2012

Current stable release 1.13.5 - 09.12.2019

Next release 1.14 - 01.2020

Why Google created Golang?

Go is the answer on the problem of engineering scale at Google

In 2011:

- 5000+ developers across 40+ offices
- 20+ changes per minute
- 50% of code base changes every month
- 50 million test cases executed per day
- single code tree

Solution: design the language for large code bases

Google - observations

- The less code we need read the better - inheritance hierarchy is bad, operators overloading is bad etc.
- We read code far more often than we write it
- Code lives longer that we expect
- Usually maintainer of the code is not the person who wrote that code
- In big organizations programming skills may be vary
- Processors speed is not increasing as expected(Moore's law is failing)

Go

Main assumptions at the :

- Simplicity
- Readability
- Easy to learn
- Build-in concurrency

Go - specs

- Java 8 spec - **788** pages PDF -> http://docs.oracle.com/javase/specs/jls/se8/jls8.pdf
- Scala spec - **191** pages PDF -> http://www.scala-lang.org/docu/files/ScalaReference.pdf
- Python spec - **150** pages PDF -> https://docs.python.org/3/reference/
- Golang spec - **51** pages PDF -> https://golang.org/ref/spec

Go vs Java

Go and Java have much in common

- C family (imperative, braces)
- Statically typed
- Garbage collected
- Memory safe (nil references, runtime bounds checks)
- Variables are always initialized (zero/nil/false)
- Methods
- Interfaces
- Type assertions (instanceof)
- Reflection

Go differs from Java in several ways

- Programs compile to single binary. There's no VM.
- Statically linked binaries
- Control over memory layout
- Function values and lexical closures
- Built-in strings (UTF-8)
- Built-in generic maps and arrays/slices
- Built-in concurrency

Go intentionally leaves out many features

- No classes
- No constructors
- No inheritance
- No final
- No exceptions
- No annotations
- No user-defined generics

Why does Go leave out those features?

Clarity is critical.

When reading code, it should be clear what the program will do.

When writing code, it should be clear how to make the program do what you want.

Sometimes this means writing out a loop instead of invoking an obscure function.

(Don't DRY out.)

For more background on design:

- Less is exponentially more (Pike, 2012) (http://commandcenter.blogspot.com/2012/06/less-is-exponentially-more.html)
- Go at Google: Language Design in the Service of Software Engineering (Pike, 2012) (http://talks.golang.org/2012/splash.article)

Who uses Go?

Who uses Go?

Google, Adobe, AeroFS, Aerospike, Apcera, Appbase, Basecamp, BBC, BBC Worldwide, **Bitbucket**, bitly, Booking.com, Canonical, CloudFlare, Cloud Foundry, Cockroach Labs, Codeship, Comcast, **CoreOS**, Couchbase, Crowdstrike, Dailymotion, **Datadog**, Datascale, Dell, **DigitalOcean**, Disqus, DNSimple, **Docker**, drone.io, **Dropbox**, **eBay**, Economist, **Facebook**, Fastly, General Electric Software, **GitHub**, Hailo, **HashiCorp**, Heroku, IBM, Imgur, **InfluxData**, Intel, Iron.io, JelloLabs, Keybase.io, Koding, Lovoo, Lyft, MaxCDN, Medium, Mesosphere, Microcosm, Minio, **MongoDB**, Mozilla, **Netflix**, New Relic, New York Times, Novartis, OpenShift, Parse.com, Percona, Pinterest, Pivotal, PocketList, pool.ntp.org, Rackspace, Reddit, Riot Games, SendGrid, SendHub, Shutterfly, **SoundCloud**, SoundHound, Sourcegraph, Space Monkey, **SpaceX**, Splice, Springer, **Square**, Stack Exchange, SteelSeries, Thomson Reuters Eikon, TIBCO, Tumblr, **Twitch**, Twitter, TweetQureet, **Uber**, VMware, Weave, Weaveworks, Wercker, Wikia, Workiva, **Yahoo**, Yandex, Yik Yak, Zalando, Zumba, Zynga

Those were big players

Rest

Google, 6Wunderkinder, 99designs, Abot, ActiveState, Acquia, adeven, Adobe, Adori, AeroFS, Aerospike, AgileBits, Airbrake, Airware, Apcera, Apeiron, Appbase, Append, Appoxy, AppsCode, Arista, Ardan, Aruba, Ascendant, Atatus, Avocet, Awake, Axioms, Basecamp, BBC, BBC, Beachfront, Beam, Beehively, Beme, Betable, Benzinga, BigCommerce, Bitbucket, bitly, Blippar, Blink, Bolt, bol, botvs, Booking, Bread, Bridgevine, Brightcove, Bringhub, BuySellAds, BuzzFeed, Byndr, Canonical, Carbon, Clearblade, Clever, CloudFlare, Cloud, CloudWalk, clypd, Cockroach, Codeship, Comcast, Conformal, Copper, CoreOS, Couchbase, Crashlytics, Crowdstrike, Cupcake, CURT, CustomerIO, Dailymotion, Datadog, Datascale, DeferPanic, Dell, DigitalOcean, Disqus, DNSimple, Docker, domainr, DotDashPay, DoubleDutch, Doximity, DramaFever, drone, Dropbox, eBay, Economist, Embedly, EMC, ERNIT, Everything, Facebook, Factom, Fastly, Foize, Flipboard, FLXOne, Fullstory, Gatherer, Gamegos, Gemfury, General, Gengo, Getty, GitHub, GlobalSign, Granify, GrayMeta, Grovo, Hailo, Happy, HashiCorp, HER,

Rest

Heroku, Home24, Hooklift, HyperDev, IBM, Igneous, Imagefly, Imgix, Imgur, InfluxData, Instela, Intel, identakid, Ionic, ISDC, Iron, JelloLabs, Jimdo, Jive, Jive, Karma, Kayako, Keybase, Kingsoft, Kochava, Koding, LabStack, Leader, Lean, Lincoln, LocalRaces, LogPacker, Lovoo, Luckie, Lyft, Magic, Maldive, MalwareBytes, MaxCDN, Medium, MediaMath, Mesosphere, Meta, Microcosm, Minio, Modcloth, Moovweb, MongoDB, Monsoon, Mozilla, MROffice, Namely, Netflix, New, New, Newspaper, Next, Nexway, Ninchat, Novartis, Odoscope, Ooyala, Opendoor, OpenShift, Ottemo, Outdoorsy, OvrClk, Oyster, Pachyderm, Packet, Pagoda, Pantheon, Parse, Partner, Percona, pet, Pinshape, Pinterest, Pivotal, PocketList, Points, pool, Poptip, Pressly, Pronovix, Public, Rackspace, Raintank, Rakuten, RapidLoop, Rawstream, Raygun, Reddit, ReelDx, Remember, Remind101, Rendered, Replicated, Repustate, ReverbNation, ReviewTrackers, Revolving, Riot, Room, Rounds, RueBaRue, Runscope, Sagan, Secret, Segment, SendGrid, SendHub, SessionM, Shopify, Showyou, Shutterfly, SignalFx, SmartyStreets, SmugMug,

Rest

Skimlinks, SoundCloud, SoundHound, Sourcegraph, Space, SpaceX, Splice, Springer, Square, StackEngine, Stack, Staffjoy, StatHat, Steals, SteelSeries, StorReduce, SumoLogic, SuperHuman, SyndicatePro, Tamber, Tapglue, Tapjoy, Teamwork, Teespring, Telit, Tendermint, TF2Stadium, The, Thisissoon, Thomson, thoughtbot, Thumbtack, TIBCO, Timehop, TinkerCad, Toggl, Torbit, Total, Transloadit, Treetop, Trisoft, Tumblr, Tune, TurboBytes, Twitch, Twitter, TweetQureet, Uber, Ulele, Umbel, Upskill, Undisclosed, Vertamedia, VerveMobile, VividCortex, VMware, VSCO, Weave, Weaveworks, Wercker, Whim, Wikia, Wireless, Workiva, Yahoo, Yandex, Yik, Zalando, Zenoss, ZITEC, Zumba, Zynga, Atlassian, Buildkite, Dgraph, OneConfig, Sajari, Gatherer, AppsCode, Telenor, Beauty, Benefício, Coderockr, Compufácil, FourTwo, GissOnline, Globo, Hotel, Jexia, Magazine, Mendelics, Moip, Neoway, Nic, Nuveo, Planrockr, Resultados, ServiceNet, SiBBr, TOTVS, Ulife, Universo, Walmart, Pronovix, Be, clouWay, Go2Mobi, Koho, Pressly, ThinkSquare, Wattpad, 500px, Prey, 163yun, Qiniu, Yeeuu, Nivas,

Rest

Rentlio, Wawandco, SYBO, Digikong, Objenious, Rive, Batch, IRI, Teads, Algolia, Bürkert, Fraugster, HaCon, HelloFresh, JustWatch, Streetspotr, Sixt, Weaveworks, PassKit, Pronovix, Betacraft, BookMyShow, C42, CodeIgnition, Company, Exotel, Fastah, GeekTrust, Gloop, Ithaka, Jabong, Josh, JusTickets, Kayako, Qwinix, Sahaj, Siminars, SoStronk, Synerzip, Techequity, Testbook, ThoughtWorks, Unbxd, Auditsi, Brodo, Bukalapak, Codemi, GO, Ice, HappyFresh, Kelir, Kudo, Lazada, LionJobs, Matahari, Magicsoft, Michael, Midtrans, Nodeflux, Pinjam, President, Qasico, RajaMall, Raydar, RebelWorks, RedtreeMobile, Rimbun, Sale, SAP, Tokopedia, Zelos, AlAlam, Aparat, Kikojas, Intercom, 353Solutions, BigPanda, Dragontail, Elastifile, EyeSight, Fiverr, Gett, ironSource, Jewish, JFrog, Nexar, PushApps, Protected, Rimoto, Rounds, StreamRail, Surge, TechedUp, uponit, Yodas, Yotpo, Hastega, Qurami, Trenìt, Cybozu, DeNA, eureka, Fenrir, HDE, Hatena, Mackerel, Intelligence, KAYAC, LINE, mediba, Mercari, Gaurun, Preferred, SensorBee, SAKURA, Arukas, Web, Sakura, Souzoh, Mercari,

Rest

Nulab, Bengo4, CloudSign, HEARTBEATS, happo, VOYAGE, Digitalidea, DATA, Segundamano, aspros, Emphatic, Avito, Jexia, localsensor, MessageBird, Nextail, Pocket, Poki, The, Wercker, Xebia, Yonego, Hapara, Vend, Weta, ULAPPH, EngageSPARK, allegro, hostit, Brainhub, CloudThing, Husar, AppGeneration, Uniplaces, Trisoft, ITooLabs, PostmanQ, Visionect, Grab, engageSPARK, Tokopedia, Pocketmath, Dwarves, alea, Bugfender, Cabify, jobandtalent, Ximdex, puzzlopia, Kiliaro, Min, Slagkryssaren, TV4, Greta, Booli, AgFlow, Centralway, Perron2, Proton, AIS, DTAC, Insightera, GramGames, Insider, ElifTech, GOV, HM, New, XTX, Badoo, Geckoboard, Restorepoint, Songkick, Sainsbury, Ryanair, Belua, Intern, Pusher, Tyk, Fortifi, Monzo, Weaveworks, StorageOS

Tools written in Go

- Docker
- Kubernetes
- Etcd
- InfluxDB
- Prometheus
- Grafana
- Traefik
- Caddy
- Consul

Everyone of you uses Go

- SPDY proxy in Chrome
- Servers dl.google.com: for Chrome, ChromeOS, Android SDK, Earth, etc.
- YouTube Vitess MySQL balancer

Why Go is awesome

Basics

Go - Hello world

```
package main

import "fmt"

func main() {
    fmt.Println("Hello Wroclaw!")
}
```

Run

Go - Hello world

```
package main

import "fmt"

func main() {
    fmt.Println("Hello Wroclaw!")
}
```

How to run from cmd:

```
go run hello.go
```

It's translated into:

```
go build hello.go && ./hello
```

Go - Hello world http

```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/hello", helloHandler)
    http.ListenAndServe(":8080", nil)
}

func helloHandler(w http.ResponseWriter, req *http.Request) {
    fmt.Fprint(w, "Hello Wroclaw!")
}
```

Run

Go - Hello world json

```
package main

import (
    "encoding/json"
    "net/http"
)

func main() {
    http.HandleFunc("/hello", jsonHandler)
    http.ListenAndServe(":8090", nil)
}

func jsonHandler(w http.ResponseWriter, req *http.Request) {
    js, err := json.Marshal(map[string]string {"hello": "wroclaw!"})
    if err != nil {
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }

    w.Header().Set("Content-Type", "application/json")
    w.Write(js)
}
```

Run

Go - concurrency

Go - concurrency

```
package main

import (
    "fmt"
)

func main() {
    go fmt.Println("Hello from goroutine")
}
```

Run

Go - concurrency

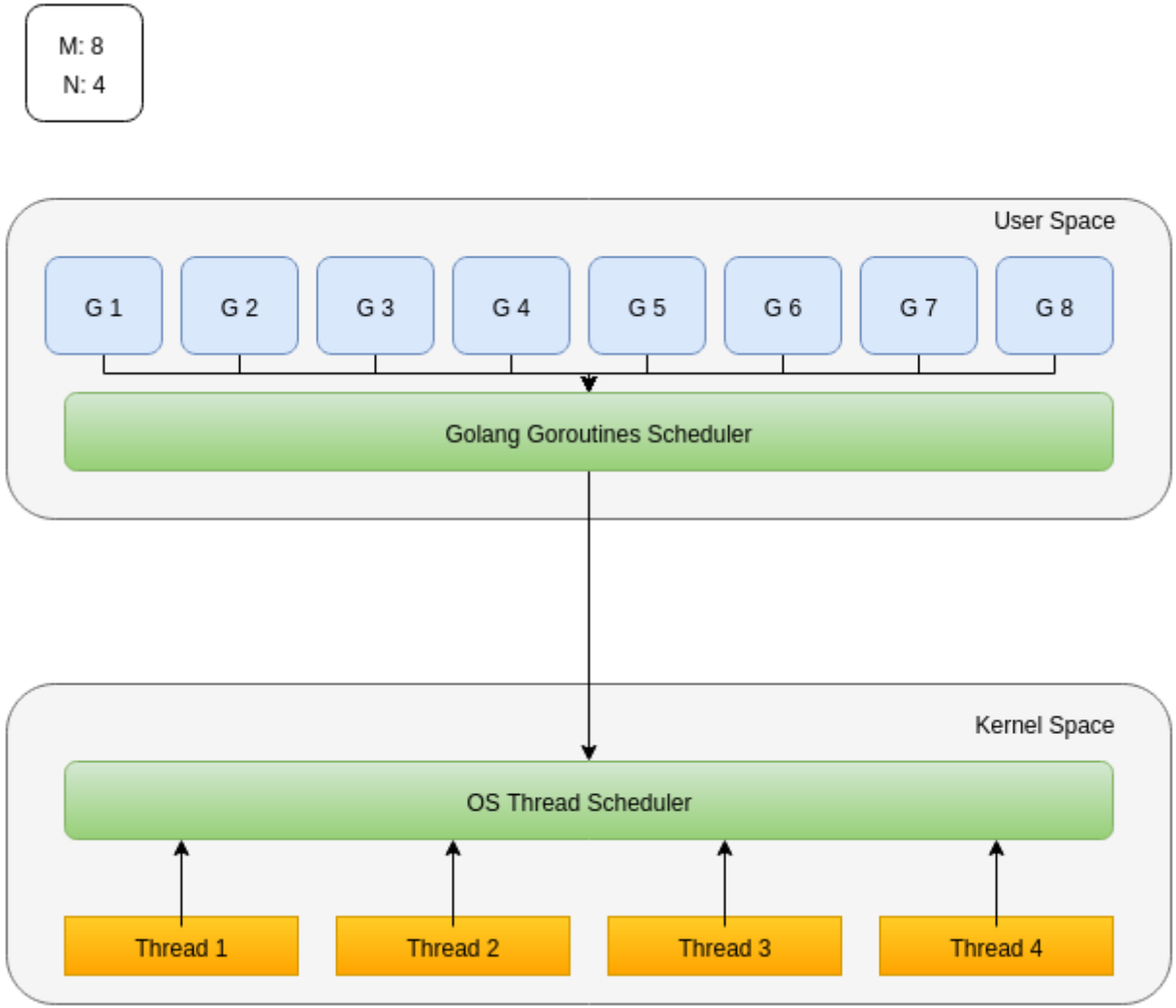
```
package main

import (
    "fmt"
    "time"
)

func main() {
    go fmt.Println("Hello from goroutine")
    time.Sleep(1 * time.Second)
}
```

Run

Go - concurrency

- What's goroutines? It's part of our program which runs independently, it's created by keyword go
- Goroutine has it's own call stack which increases or shrinks when needed
- Goroutine is very light comparing to thread - we could have millions of them
- Goroutine is not a thread.
- Our application might have 1 thread and millions of goroutines.
- Goroutines have their own scheduler which is much faster than system thread scheduler
- Goroutine scheduler switch goroutine when notices that goroutine is blocked(for example on I/O)
- We could call goroutine as cheap/light thread

Go - concurrency - threads vs goroutines



Go - concurrency - communication

- Our main goroutines doesn't communicate with new goroutine. We made assumption that job in new goroutine will be finished before our main goroutine ends.
- Creators of Go implemented solution for problems with communitaions betweeen goroutines.
- Channels are invented to face that problem.

Go - concurrency - channels

Channel provides communication between goroutines

```
// Declaring and initializing.
var c chan int
c = make(chan int)

// or
c := make(chan int)

// Sending on a channel.
c <- 1

// Receiving from a channel.
// The "arrow" indicates the direction of data flow.
value = <-c
```

Go - concurrency - channels

```
package main

import (
    "fmt"
)

func main() {
    done := make(chan bool) // create channel
    go func() {
        fmt.Println("Hello Wroclaw")
        done <- true // send information that our goroutine finished its work
    }()

    <-done // wait until something occurs in channel
}
```

Run

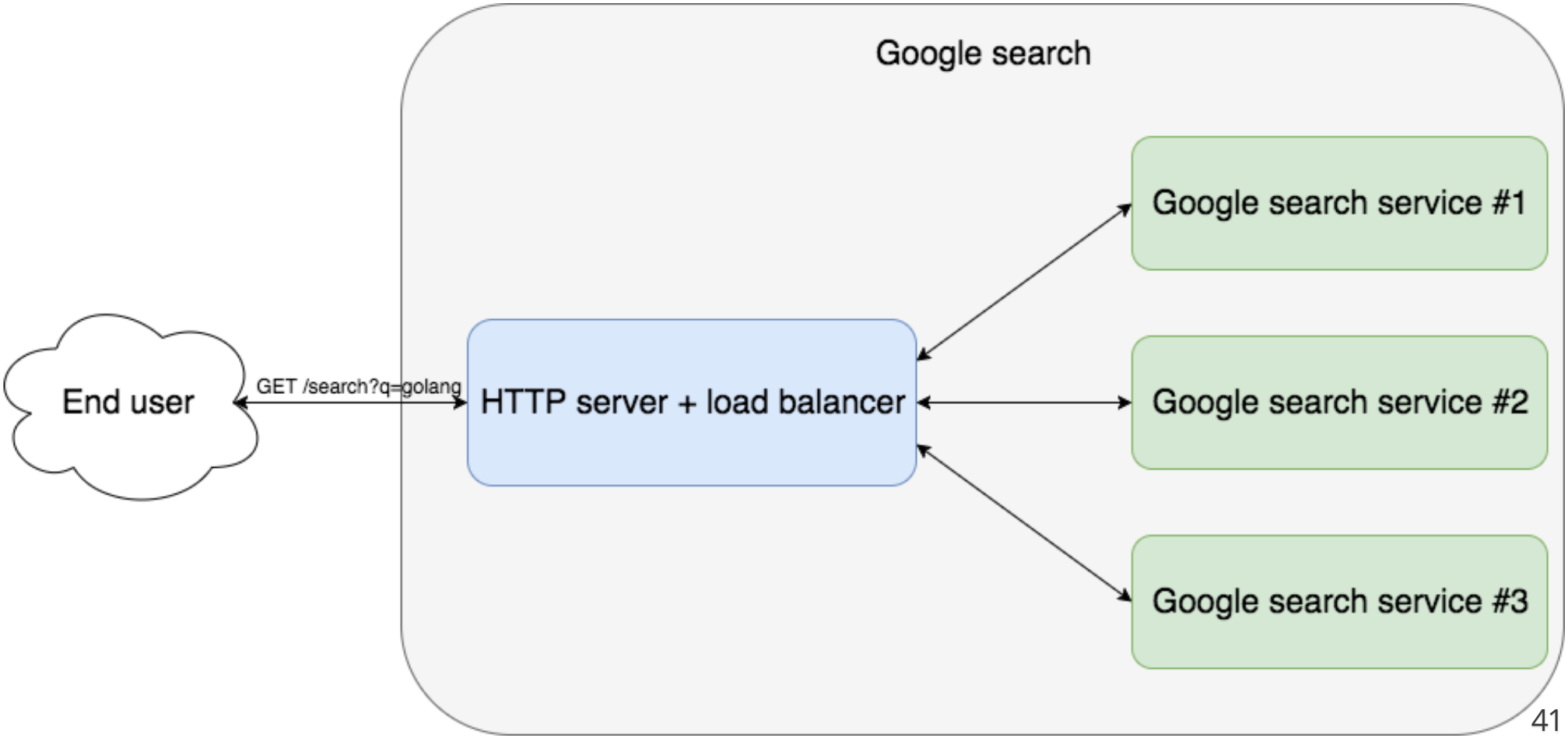
Go - concurrency - select

```
package main

import (
    "fmt"
    "time"
)

func main() {
    done := time.NewTimer(time.Second * 10) // create channel which puts value into it after 10s

    // infinite loop
    for {
        select {
        case <-time.After(time.Second * 1): // get value from channel each second
            fmt.Println("Hello Wroclaw")
        case <-done.C: // wait 10s until channel returns its value
            fmt.Println("Quitting")
            return
        }
    }
}
```

Run

Go - goroutines + channels + select

Google Search Service



Google Search Service - http server

```
func main() {
    // configure http paths
    http.HandleFunc("/search", searchFor)

    // start http server
    fmt.Println("Listening on port 9000")
    http.ListenAndServe(":9000", nil)
}
```

Google Search Service - http handler

```
func searchFor(w http.ResponseWriter, r *http.Request) {
    // get search term from url query string
    query := r.URL.Query().Get("q")
    fmt.Printf("Start searching for string=%s\n", query)

    // create a channel for responses
    resultChan := make(chan response)

    // run queries to all backend services
    for k := range backendServers {
        go func() {
            resultChan <- searchInBackend(k, query)
        }()
    }

    // get only first result and write answer to response
    res := <-resultChan
    fmt.Printf("Got response from server with id: %d\n", res.serverID)
    fmt.Fprint(w, string(res.payload))
}
```

Google Search Service - search in particular backend

```
func searchInBackend(id int, query string) response {
    // make GET request
    resp, err := http.Get(backendServers[id] + query)
    if err != nil {
        log.Fatalf("can't make request to: %s, err: %v", backendServers[id]+query, err)
    }

    if resp.Status != "200 OK" {
        log.Fatalf("wrong http status: %s", resp.Status)
    }

    // read response and return bytes
    bytes, _ := ioutil.ReadAll(resp.Body)
    return response{id, bytes}
}
```

Google Search Service

Demo:

```
func main() {
    // configure http paths
    http.HandleFunc("/search", searchFor)

    // start http server
    fmt.Println("Listening on port 9000")
    http.ListenAndServe(":9000", nil)
}
```

Run

Test:

```
http://localhost:9000/search?q=google
```

or:

```
hey -n 100 -c 10 http://localhost:9000/search?q=google
```

Go - performance

Go vs other languages

Faster:

- Fortran
- C++
- C
- Rust
- Ada

Comparable speed:

- C#
- Java
- Scala
- Swift

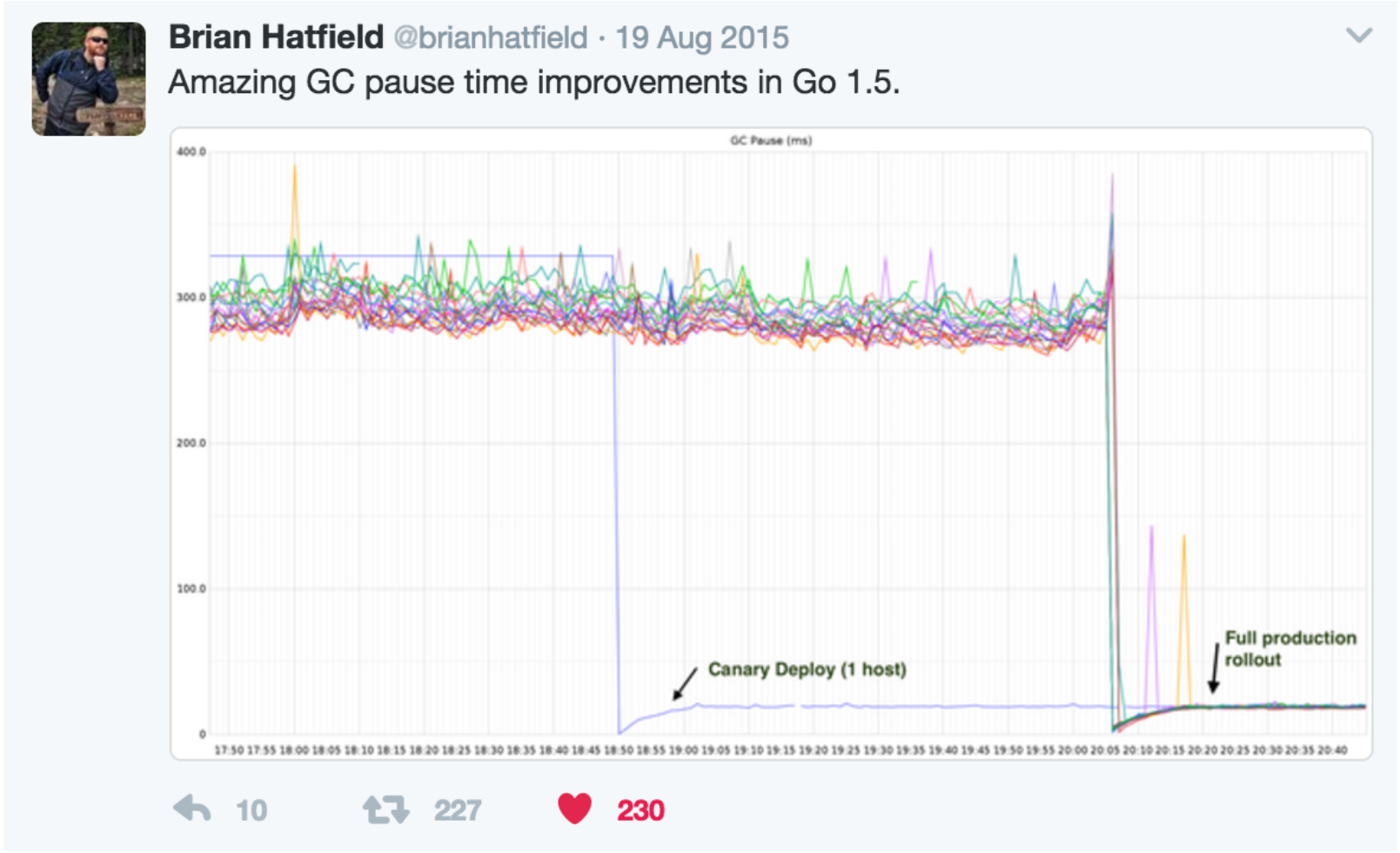
Source: http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=nbody

(http://benchmarksgame.alioth.debian.org/u64q/performance.php?test=nbody)

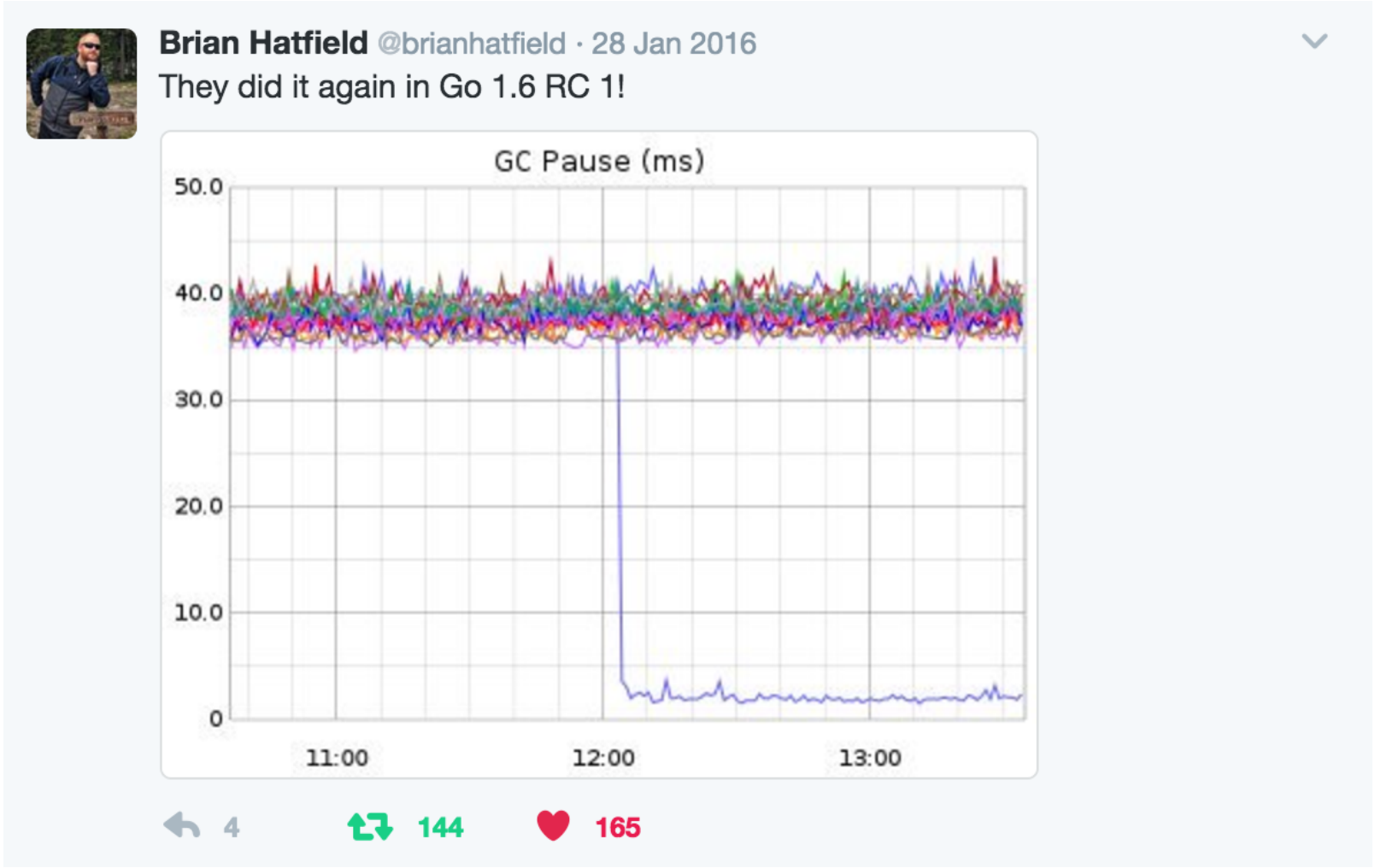
Go - performance - gettings faster

Go - performance - GC

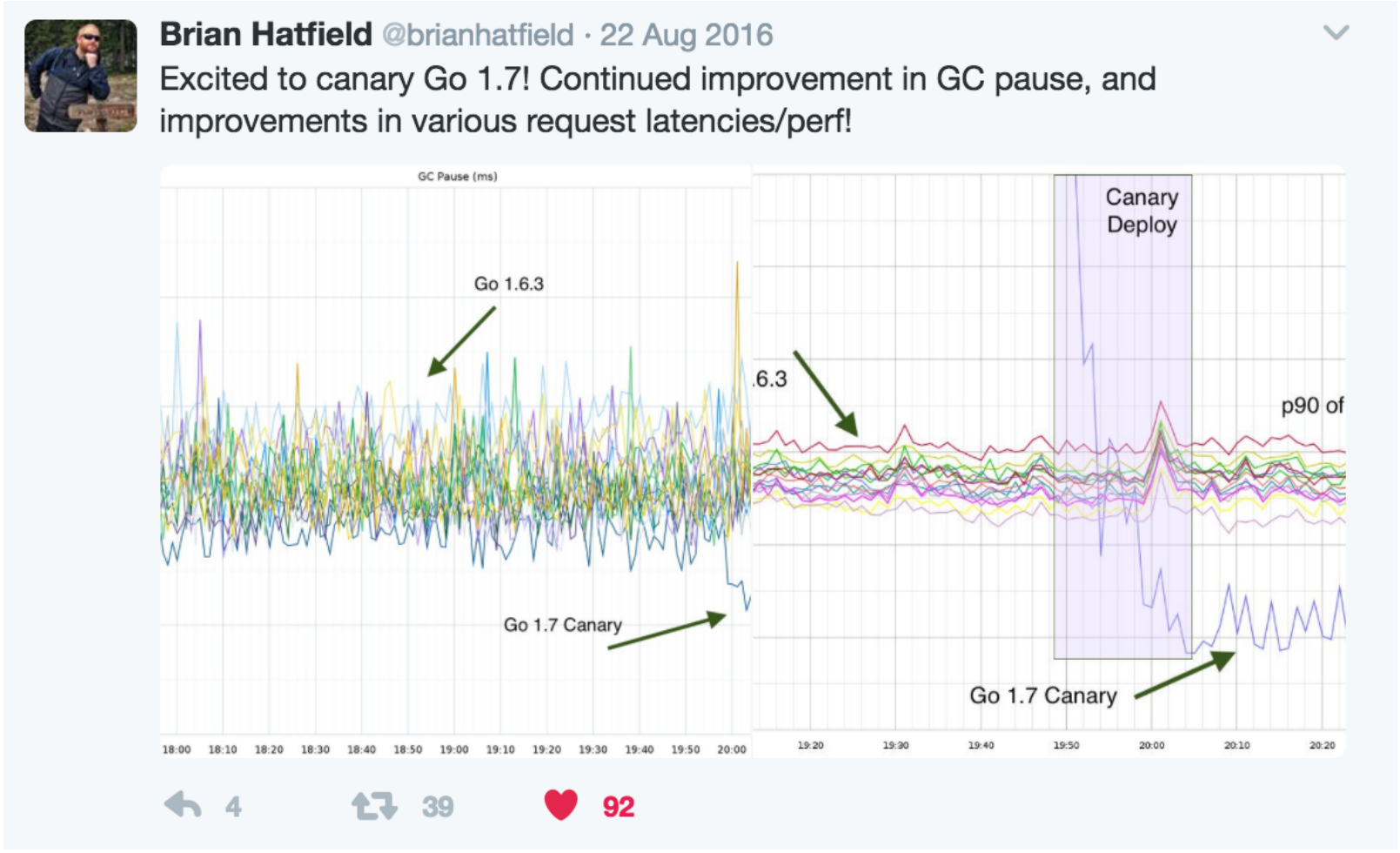
go 1.5


Brian Hatfield @brianhatfield · 19 Aug 2015
Amazing GC pause time improvements in Go 1.5.
GC Pause (ms)
400.0
300.0
200.0
100.0
0
Canary Deploy (1 host)
Full production rollout
10
227
230

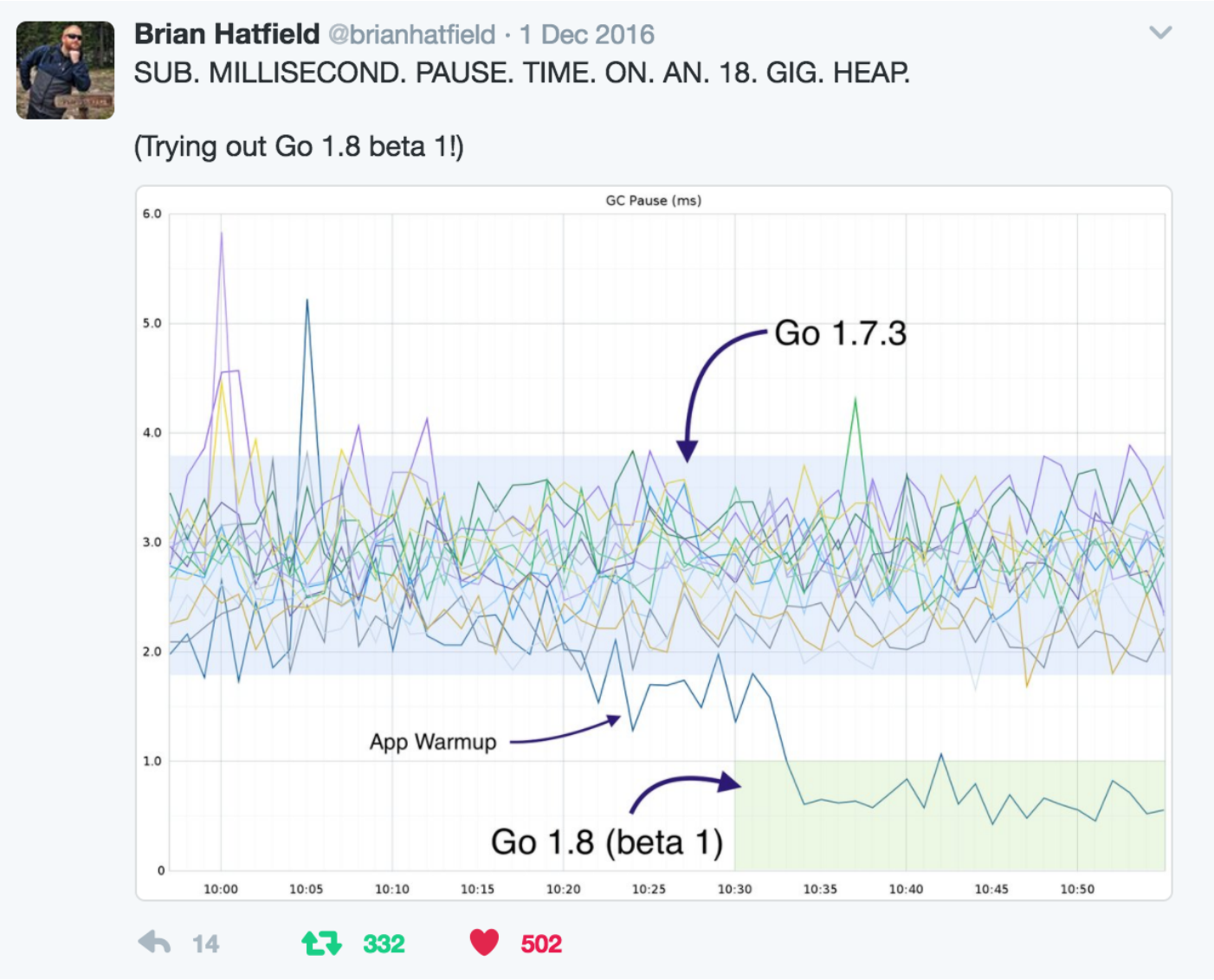
go 1.6


Brian Hatfield @brianhatfield · 28 Jan 2016
They did it again in Go 1.6 RC 1!
GC Pause (ms)
50.0
40.0
30.0
20.0
10.0
0
11:00
12:00
13:00
4
144
165

go 1.7


Brian Hatfield @brianhatfield · 22 Aug 2016
Excited to canary Go 1.7! Continued improvement in GC pause, and improvements in various request latencies/perf!
GC Pause (ms)
Go 1.6.3
Go 1.7 Canary
18:00 18:10 18:20 18:30 18:40 18:50 19:00 19:10 19:20 19:30 19:40 19:50 20:00
Canary Deploy
.6.3
p90 of
Go 1.7 Canary
19:20 19:30 19:40 19:50 20:00 20:10 20:20
4
39
92

go 1.8 (beta 1)


Brian Hatfield @brianhatfield · 1 Dec 2016
SUB. MILLISECOND. PAUSE. TIME. ON. AN. 18. GIG. HEAP.
(Trying out Go 1.8 beta 1!)
GC Pause (ms)
6.0
5.0
4.0
3.0
2.0
1.0
0
Go 1.7.3
App Warmup
Go 1.8 (beta 1)
10:00
10:05
10:10
10:15
10:20
10:25
10:30
10:35
10:40
10:45
10:50
14
332
502

Go - deployment is extra dope

Go - static binaries

Commend:

```
go build hello.go
```

Creates single binary file, which we could launch:

```
ls -alh
```

Result:

```
-rwxr-xr-x   1 md  staff   1.5M Feb  9 22:47 hello
-rw-r--r--   1 md  staff    75B Feb  9 22:32 hello.go
```

Go - cross-compilation

Command:

```
go build hello.go
```

Compiles out applications for default OS and architecture

On my computer:

```
GOARCH="amd64"
GOOS="darwin"
```

If we want compile app for Windows OS:

```
GOOS=windows GOARCH=amd64 go build hello.go
```

Result:

```
-rwxr-xr-x   1 md  staff  1715200 Feb  9 23:05 hello.exe
-rw-r--r--   1 md  staff       75 Feb  9 22:32 hello.go
```

Go - small memory footprint

We could dockerize that with following Dockerfile:

```
FROM golang

RUN mkdir /fanout
ADD fan-out-docker.go /app/
WORKDIR /app
RUN go build -o fanout .

EXPOSE 9000

CMD ["/app/fanout"]
```

Docker build & run:

```
docker build -t google .
docker run -d --link dummy1 --link dummy2 --link dummy3 -p 9000:9000 google
```

Demo:

```
curl http://localhost:9000/search?q=test
docker ps & docker stats
hey -c 10 -n 100 http://localhost:9000/search?q=test
```

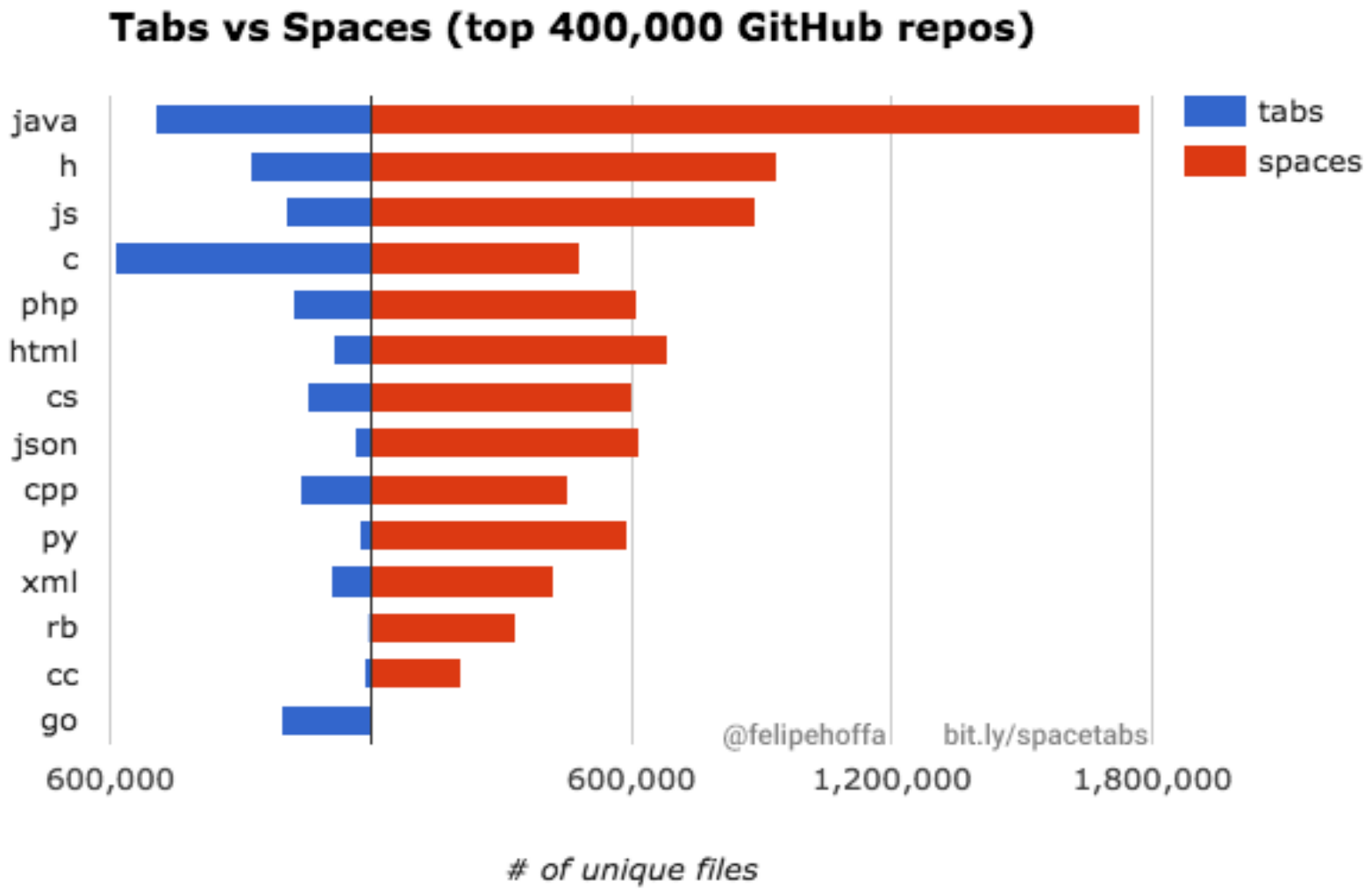
Go - ecosystem

Go - ecosystem - code formatting

- One tool
- There is no quarrel about code formatting

```
gofmt -w .
```

Go - ecosystem - code formatting



Go - ecosystem - play golang

play.golang.org/ (https://play.golang.org/)

Go - ecosystem - golang tour

tour.golang.org/ (https://tour.golang.org/)

Go - ecosystem - dependencies

```
package main

import (
    "os"
    "path"

    "go.uber.org/zap"
)

func main() {
    logger := zap.New(
        zap.NewJSONEncoder(zap.RFC3339Formatter("key")),
        zap.Fields(zap.Int("pid", os.Getpid()),
            zap.String("exe", path.Base(os.Args[0]))),
    )

    logger.Info("Hello Wroclaw")
}
```

Dependencies are downloaded with:

```
go get github.com/uber-go/zap
```

Go - ecosystem - benchmark

Code:

```
package main

func Fib(n int) int {
    if n < 2 {
        return n
    }
    return Fib(n-1) + Fib(n-2)
}
```

Benchmark:

```
package main

import "testing"

func BenchmarkFib10(b *testing.B) {
    // run the Fib function b.N times
    for n := 0; n < b.N; n++ {
        Fib(10)
    }
}
```

Go - ecosystem - benchmark

To run benchmark:

```
go test -bench=. --benchmem
```

Result:

```
md@md fib (master *+)$ go test -bench=. --benchmem
testing: warning: no tests to run
BenchmarkFib10-8        3000000           403 ns/op           0 B/op           0 allocs/op
PASS
ok      github.com/mateuszdyminski/whygo/code/fib    1.622s
```

Go - ecosystem - godoc

All documentations are in one place:

godoc.org/ (https://godoc.org/)

In popular repositories there is link to godoc.org:

github.com/uber-go/zap (https://github.com/uber-go/zap)

Go - ecosystem - present

godoc.org/golang.org/x/tools/present (https://godoc.org/golang.org/x/tools/present)

This presentation is run with tool written in Go: `present`

To run presentation:

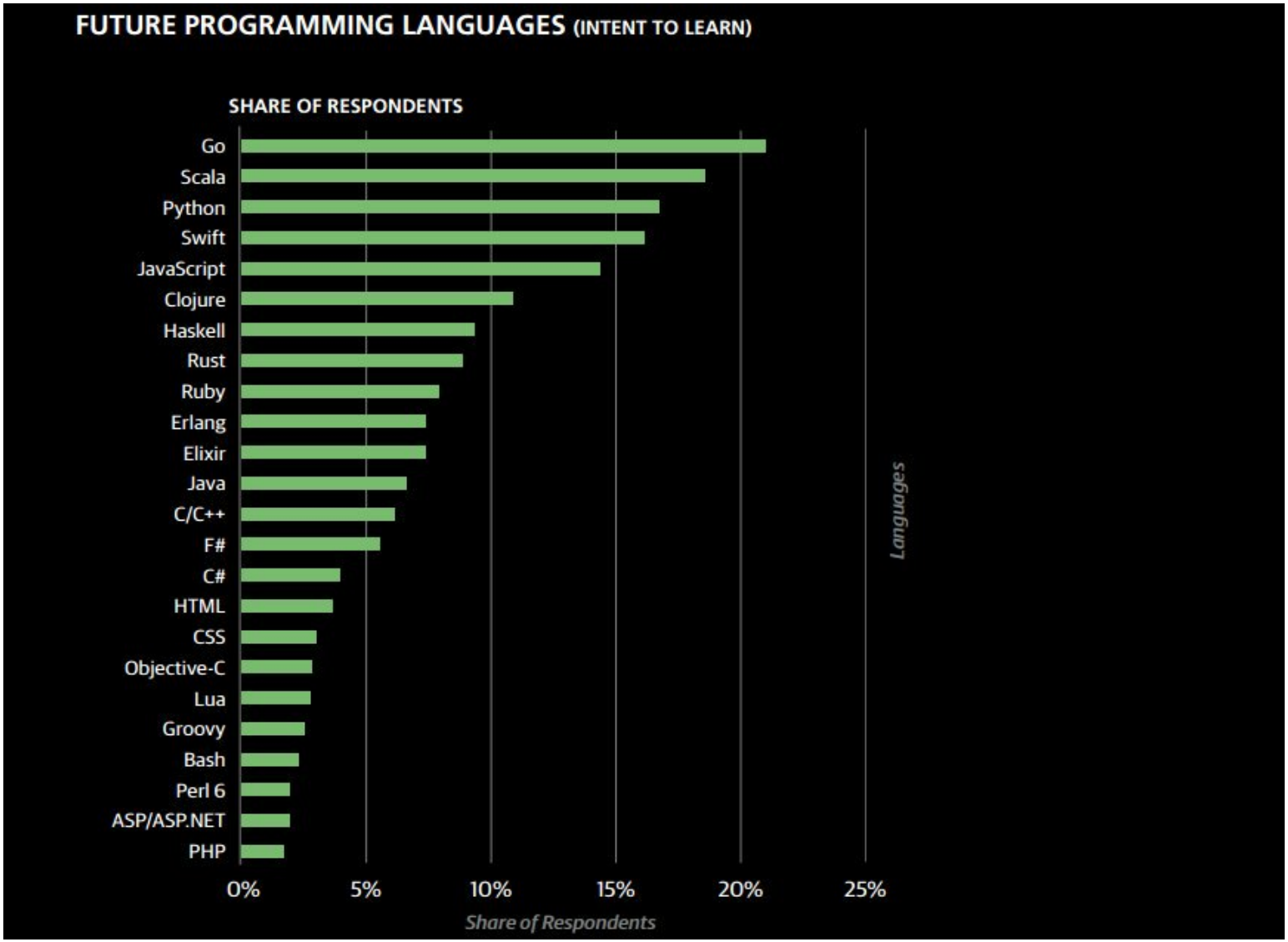
```
present
```

Go - ecosystem - IDEs

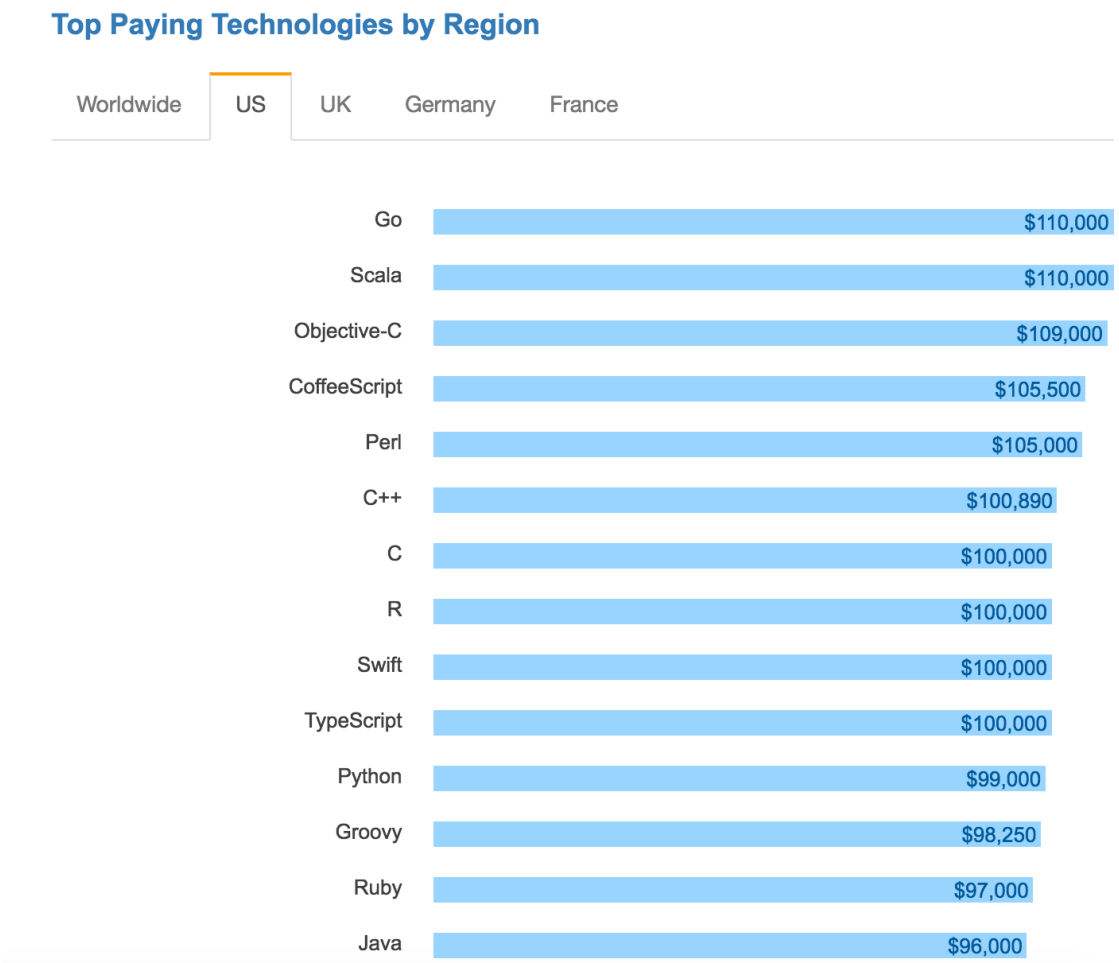
- VIM
- VisualStudio Code
- Golang
- Atom

https://golang.org/doc/editors.html (https://golang.org/doc/editors.html)

Go has potential



And Go developer salary is rather high



Source: https://stackoverflow.com/insights/survey/2017/#top-paying-technologies

(https://stackoverflow.com/insights/survey/2017/#top-paying-technologies)

Go - what's next?

Golang tour

- tour.golang.org (http://tour.golang.org)

Tutorials/resources to learn Go

- golang.org/wiki/Learn (http://golang.org/wiki/Learn)
- golang.org/doc/code.html (https://golang.org/doc/code.html)
- golang.org/doc/effective_go.html (https://golang.org/doc/effective_go.html)
- github.com/golang/go/wiki/CodeReviewComments (https://github.com/golang/go/wiki/CodeReviewComments)
- gobyexample.com/ (https://gobyexample.com/)

Wroclaw GoWroc

- meetup.com/GoWroc (https://meetup.com/GoWroc)

Questions ?

Thank you

Mateusz Dymiński
Nokia

github.com/mateuszdyminski/whygo (github.com/mateuszdyminski/whygo)
@m_dyminski (http://twitter.com/m_dyminski)