Go Language

May 2014

Giacomo Tartari
PhD student, University of Tromsø

Go

A new language?

Why?
and what for?
Don't we have Java?
C?
C++?
C#?
D?
Haskell?
Scala?
Python?
PHP?
Ruby?
Perl?!
Brainfuck!!!!
[add random language here]?

A new language

What's wrong with all of the above?

Rob Pike's take (one of the Go instigator) (http://talks.golang.org/2012/splash.article)

Languages used at Google were not satisfactory

- These languages were developed before the multi-core revolution
- Millions of lines of code maintained by thousands of programmers
- Build times of many minutes or hours

So they (*rob*, *ken* and *gri*) designed Go

- Modern, pragmatical and language
- Easy to read, clean syntax, fast compile time and good tools
- Nothing new, but a collection of good features

Go

Compiler assistance (e.g. unused variables are errors)

Standardized formatting (all Go code looks familiar)

Small language (can use all the features without "surprises")

Easy to jump in (easy to learn, productive without an IDE)

Batteries included (rich standard library)

Fast time to "done" (garbage collection, built in maps & slices)

Easy concurrency (asynchronous code without callbacks)

Multi-core, High performance (concurrent -> parallel)

Go

Single Binary (trivial deployment)

Cross Compilation (develop on Windows deploy on linux, also ARM)

Built-in Profiling (via http of CPU, memory, lock contention etc.)

Shareable Code (small sized code, gofmt, go tool, etc..)

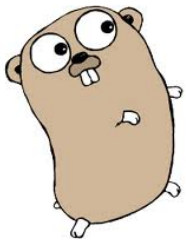
Why code in Go? (http://tech.t9i.in/2013/01/why-program-in-go/)

Go

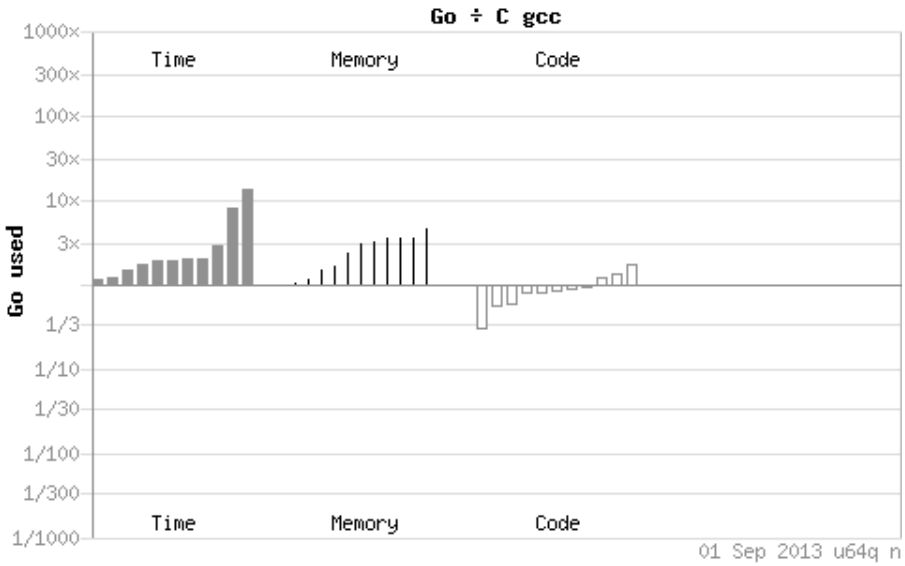
Opensourced in 2009

Current version 1.2.2

Now in feature freeze for Go 1.3

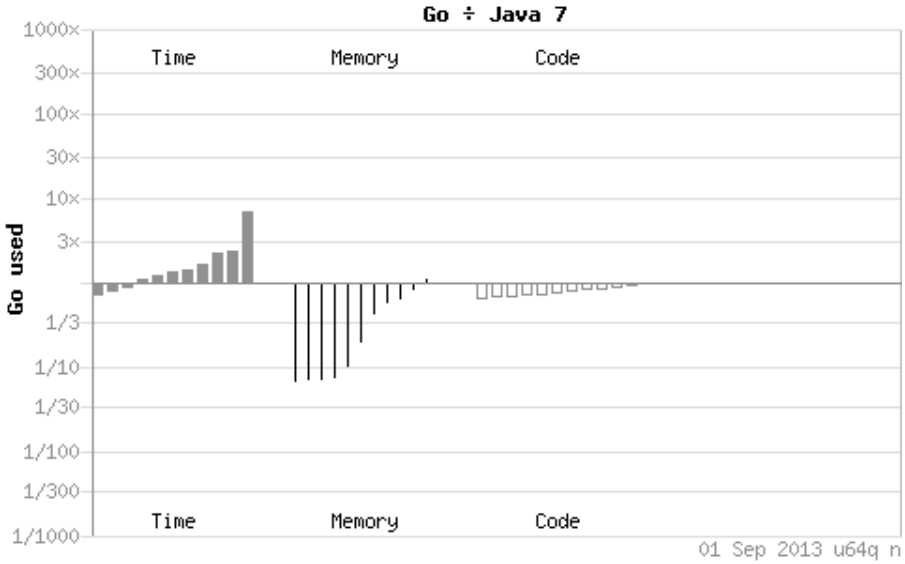


Go benchmarks (!)

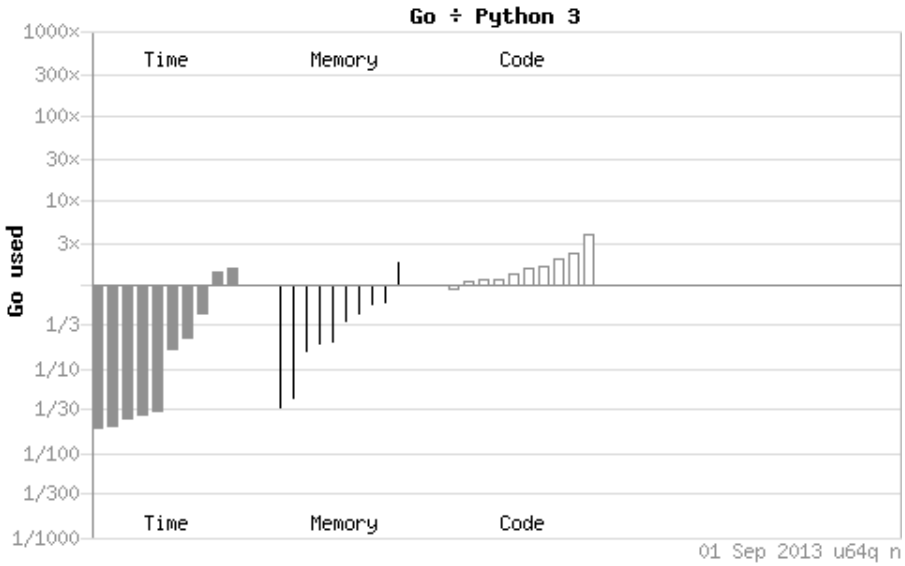


Benchmarks from here (http://benchmarksgame.alioth.debian.org/u64q/go.php)

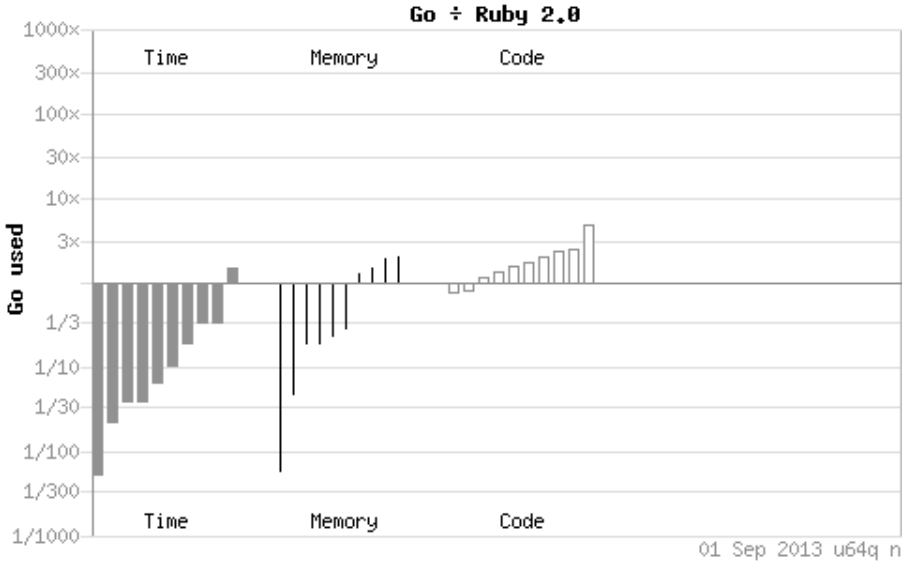
Go benchmarks (!)



Go benchmarks (!)



Go benchmarks (!)



Go benchmarks (!)

github.com/kidoman/fibrous#summary (https://github.com/kidoman/fibrous#summary)

Popularity (Success?)

- Fist Go conference GopherCon 2014 (http://gophercon.com/)
- Videos (http://confreaks.com/events/gophercon2014)
- Companies using Go (https://code.google.com/p/go-wiki/wiki/GoUsers)
- Success stories (https://code.google.com/p/go-wiki/wiki/SuccessStories)

Concurrency

Concurrency

Concurrency at language level

Concurrency enables parallelism

Goroutines provide concurrency in Go

- **go** statement allows us to run functions independently in different goroutines
- Goroutines live in the same address space
- Think of them as a very lightweight threads

Hello Goroutines World

```
package main

import (
    "fmt"
    "time"
)

func main() {
    say("world")
    say("hello")
}

func say(s string) {
    for i := 0; i < 5; i++ {
        time.Sleep(100 * time.Millisecond)
        fmt.Printf("%s\n", s)
    }
}
```

Run

Goroutine

```
func main() {
    seed := time.Now().UnixNano()
    source := rand.NewSource(seed)
    r := rand.New(source)
    t0 := time.Now()
    for i := 0; i < 5; i++ {
        do(r.Intn(1000), i)
    }
    fmt.Printf("total time %v\n", time.Now().Sub(t0))

}

func do(work, id int) {
    t0 := time.Now()
    time.Sleep(time.Duration(work) * time.Millisecond)
    fmt.Printf("done %d [%v]\n", id, time.Now().Sub(t0))
}
```

Run

Goroutine

```
func main() {
    seed := time.Now().UnixNano()
    rand.Seed(seed)
    t0 := time.Now()
    for i := 0; i < 5; i++ {
        go do(rand.Intn(1000), i)
    }
    time.Sleep(5 * time.Second)
    fmt.Printf("total time %v\n", time.Now().Sub(t0))
}

func do(work, id int) {
    t0 := time.Now()
    time.Sleep(time.Duration(work) * time.Millisecond)
    fmt.Printf("done %d [%v]\n", id, time.Now().Sub(t0))
}
```

Run

Goroutine & channels

```
func main() {
    seed := time.Now().UnixNano()
    rand.Seed(seed)
    t0 := time.Now()
    done := make(chan string)
    for i := 0; i < 5; i++ {
        go do(rand.Intn(1000), i, done)
    }
    for i := 0; i < 5; i++ {
        fmt.Println(<-done)
    }
    fmt.Printf("total time %v\n", time.Now().Sub(t0))

}

func do(work, id int, ch chan string) {
    t0 := time.Now()
    time.Sleep(time.Duration(work) * time.Millisecond)
    ch <- fmt.Sprintf("done %d [%v]", id, time.Now().Sub(t0))
}
```

Run

Goroutine & channels

```
func main() {
    seed := time.Now().UnixNano()
    rand.Seed(seed)
    t0 := time.Now()
    done := make(chan string)
    for i := 0; i < 5; i++ {
        go func(wid int) {
            res := do(rand.Intn(1000), wid)
            done <- res
        }(i)
    }
    for i := 0; i < 5; i++ {
        fmt.Println(<-done)
    }
    fmt.Printf("total time %v\n", time.Now().Sub(t0))

}

func do(work, id int) string {
    t0 := time.Now()
    time.Sleep(time.Duration(work) * time.Millisecond)
    return fmt.Sprintf("done %d [%v]", id, time.Now().Sub(t0))
}
```

Run

Goroutine & channels

```
func main() {
    seed := time.Now().UnixNano()
    rand.Seed(seed)
    timeout := time.After(3 * time.Second)
    t0 := time.Now()
    done := make(chan string)
    for i := 0; i < 5; i++ {
        go func(wid int) {
            res := do(rand.Intn(5000), wid)
            done <- res
        }(i)
    }
    for i := 0; i < 5; i++ {
        select {
        case res := <-done:
            fmt.Println(res)
        case <-timeout:
            fmt.Printf("timeout, workers done: %d\n", i)
            return
        }
    }
    fmt.Printf("total time %v\n", time.Now().Sub(t0))
}
```

Run

Practical stuff

Installation

Official binary distributions

- FreeBSD
- Linux
- Mac OS X
- NetBSD
- Windows

32 and 64 bit, ARM and x86

Installation

Or from source

- Needs a C compiler
- Needs mercurial
- Allows you to try the development branch

go tool(s)

```
$go help
Go is a tool for managing Go source code.

Usage:

    go command [arguments]

The commands are:

    build       compile packages and dependencies
    clean       remove object files
    env         print Go environment information
    fix         run go tool fix on packages
    fmt         run gofmt on package sources
    get         download and install packages and dependencies
    install     compile and install packages and dependencies
    list        list packages
    run         compile and run Go program
    test        test packages
    tool        run specified go tool
    version     print Go version
    vet         run go tool vet on packages
    ...
```

go tool(s)

go get

```
$go get github.com/golang/glog
```

go build

```
$cd $GOPATH/src/myproject
$go build
```

go run

```
$cd $GOPATH/src/myproject
$vim main.go
$go run main.go
```

go fmt (aka end of coding style war!!)

```
$go fmt .
```

godoc

Offline docs

```
$godoc
usage: godoc package [name ...]
godoc -http=:6060
...
```

Online docs for the standard library

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

Online docs for third party libraries

godoc.org/ (http://godoc.org/)

gowalker.org/ (http://gowalker.org/)

Online presentation

talks.godoc.org/ (http://talks.godoc.org/)

Integration

$ls -l $GOROOT/misc

```
IntelliJIDEA
arm
bash
bbedit
benchcmp
cgo
chrome
dashboard
dist
emacs
fraise
git
goplay
kate
linkcheck
notepadplus
pprof
swig
vim
xcode
zsh
```

Tests and benchmark

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

Put your tests/benchmark in a file ending in `_test.go`

```
import testing

func TestXxx(t *testing.T){
    ...
}

func BenchmarkXxx(b *testing.B){
    ...
}
```

```
$cd $GOPATH/src/mypackage
$go test
$go test -bench=.
```

golang.org/cmd/go/#Description_of_testing_flags (http://golang.org/cmd/go/#Description_of_testing_flags)

Profiling

software.intel.com/en-us/blogs/2014/05/10/debugging-performance-issues-in-go-programs (https://software.intel.com/en-us/blogs/2014/05/10/debugging-performance-issues-in-go-programs)

blog.golang.org/profiling-go-programs (http://blog.golang.org/profiling-go-programs)

github.com/davecheney/profile (http://github.com/davecheney/profile)

Readings

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

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

gobyexample.com/ (https://gobyexample.com/)

learnxinyminutes.com/docs/go/ (http://learnxinyminutes.com/docs/go/)

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

code.google.com/p/go-wiki/w/list (https://code.google.com/p/go-wiki/w/list)

code.google.com/p/go-wiki/wiki/GoTalks (https://code.google.com/p/go-wiki/wiki/GoTalks)

research.swtch.com/godata (http://research.swtch.com/godata)

morsmachine.dk/go-scheduler (http://morsmachine.dk/go-scheduler)

CSP (http://www.cs.ucf.edu/courses/cop4020/sum2009/CSP-hoare.pdf)

Thank you

Giacomo Tartari
PhD student, University of Tromsø
giacomo.tartari@uit.no (mailto:giacomo.tartari@uit.no)