Go
Crash course
16 January 2014

Rob Figueiredo
Engineering Manager, Yext
Why Go? (Motivation)

"Go is more about software engineering than programming language research. Or to rephrase, it is about language design in the service of software engineering."
Why Go?

Go is like a better C, from the guys that didn't bring you C++

-- Ikai Lan
Go = C + concurrency + ...
Go = [C] + concurrency + ...

- [Procedural]
- Pointers and Values
- Structs, Arrays, Functions

```c
#include <stdio.h>

int main(void) {
    printf("hello, world\n");
}
```

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello, 世界")
}
```
Go = [C] + concurrency + ...

- Procedural
- [Pointers and Values]
- Structs, Arrays, Functions

```go
package main

import "fmt"

type foo struct { a, b int }

func zeroB(f foo) {
    f.b = 0
}

func main() {
    var f = foo{1, 2}
    zeroB(f)
    fmt.Printf("%#v", f)
}
```
Go = [C] + concurrency + ...

- Procedural
- Pointers and Values
- [Structs, Arrays, Functions]

type Person struct {
    Name    string
    Age     func() int
    Skills  []string
}

func main() {
    var person = Person{
        Name:    "Rob",
        Age:     func() int { return 31 },
        Skills:  []string{"Foosball", "Cornhole"},
    }

    fmt.Printf("Hello %v (%d), \n I see you excel at %v\n", 
        person.Name, person.Age(), person.Skills)
}
Go = C + [concurrency] + ...  

- Goroutines

```go
var query = ...
go WebSearch(query)       // run multiple
go ImageSearch(query)      // .. search functions
go VideoSearch(query)      // .. concurrently
```

- Channels

```go
c := make(chan SearchResult)  // Initialize an unbuffered channel
c <- SearchResult{..}        // Send a result on the channel
result = <-c                  // Receive a result from the channel
```

- Select

```go
select {
  case result := <-c: fmt.Println("Got result:", result)
  case result := <-c2: fmt.Println("Got other result:", result)
  default:          fmt.Println("Both channels are empty.")
}
```
Go = C + [concurrency] + ...

A concurrency example: Merging search results.

c := make(chan Result)
go func() { c <- Web(query) }()    
go func() { c <- Image(query) }()    
go func() { c <- Video(query) }()    

timeout := time.After(80 * time.Millisecond)
for i := 0; i < 3; i++ {
    select {
    case result := <-c:
        fmt.Println(result)
        results = append(results, result)
    case <-timeout:
        fmt.Println("timed out")
        return
    }
}
Go = C + concurrency + [...]

- Unicode-friendly (string, rune)
- Structural typing

```go
type Engine interface {
    Search(q string) Result
}
```

- Embedded types

```go
type Student struct {
    Person
}
```

- Type inference

```go
var tests = map[string][]string{
    "":     {""},
    "a":     {"a"},
    "ab":    {"ab", "ba"},
    "abc":  {"abc", "acb", "bac", "bca", "cab", "cba"},
}
```

Why Go?

Go doesn't implicitly anything.

-- Steven in golang-nuts
Why Go?

Go doesn't implicitly anything.
-- Steven in golang-nuts

For example...

- No implicit type conversions
- No operator overloading
- No method overloading
- No syntactic sugar
Why Go?

PSA: Go optimizes for readability, not LOC.
-- Dave Cheney on Twitter
Why Go? (Readability)

Go has a small number of orthogonal features

- 25 keywords -- (C has 50, C++/Java have 75+)
- Short, readable spec -- (C++ spec is 1300 pages, costs CHF238)
- For example: no "static volatile friend virtual const <Template>" functions
def gzipResult(result: PlainResult): Result = result match {
  case simple @ SimpleResult(header, content) => SimpleResult(header.copy(
    headers = (header.headers - "Content-Length") + ("Content-Encoding" -> "gzip")
  ), content &> Enumeratee.map(a => simple.writeable.transform(a)) &> Gzip.gzip())
}

Understanding the Play! Filter API (http://jazzy.id.au/default/2013/02/16/understanding_the_play_filter_api.html)
        case Input.EOF => Done(Iteratee.flatten(it.feed(Input.EOF)), Input.EOF)
        case Input.Empty => Cont(step(it))
        case Input.El(e) => {
            val promise = e >> Cont(removeEof(it))
            val next = Iteratee.flatten(promise.flatMap(__.run))
            next.flatMap(
                (v, l) => Done(next, in),
                (_) => Cont(step(next)),
                (msg, input) => Error(msg, in))
        }
    }
}

Iteratees in Big Data at Klout (http://engineering.klout.com/2013/01/iteratees-in-big-data-at-klout/)
func foo() error {
    f, err := os.Open("filename")
    if err != nil {
        return
    }
    defer f.Close()

    contents, err := ioutil.ReadAll(f)
    if err != nil {
        return err
    }
    ...

    return nil
}
Why Go? (Practice)

Go is not meant to innovate programming theory. It's meant to innovate programming practice.

-- Samuel Tesla
Tooling
Tooling

Google

why is eclipse so slow
why is eclipse slow
why is eclipse juno so slow
why is eclipse so slow on mac

Eclipse Community Forums: Eclipse 4 » Eclipse 4.2 GUI is very slow
www.eclipse.org/forums/index.php/t/367243/
Jul 10, 2012 - 40 posts - 16 authors
I still find it strange how the new theme can slow down the UI so much. Unfortunately I'm gone, Juno is very slow and the bad question is that...

Eclipse Community Forums: Eclipse Platform » Juno...
Eclipse Community Forums: Subversive » Subversive compare...
Eclipse Community Forums: EGit » Egit very slow at dealing big...
Eclipse Community Forums: Eclipse Platform » Why is Eclipse so...

Why is Eclipse so freaking slow?! - Ubuntu Forums
ubuntuforums.org » ... » Programming Talk
Feb 9, 2011 — I recently got a new laptop - Asus G73JW. This thing is fast and has 8 GB of RAM. So why does it feel 5x slower running Eclipse than on my...
Tooling

why is emacs
why is emacs better than vim
why is emacs so popular
why is emacs so good
why is emacs good
Tooling

- Hypersonic builds
- Programmatic formatting
- Testing, benchmarking, profiling
- Documentation browser
- Statically linked binaries
- The Playground
- C interop
Hypersonic Builds

How fast?

$ cd ~/go/src # the Go source tree
$ time ./make.bash # build the compiler, runtime, standard library

# Building C bootstrap tool.
# Building compilers and Go bootstrap tool for host, darwin/amd64.
# Building packages and commands for darwin/amd64.
...

real 0m19.269s

How much code?

$ cloc --not-match-f _test.go ~/go/src

<table>
<thead>
<tr>
<th>Language</th>
<th>files</th>
<th>blank</th>
<th>comment</th>
<th>code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Go</td>
<td>895</td>
<td>27144</td>
<td>42126</td>
<td>198965</td>
</tr>
<tr>
<td>C</td>
<td>118</td>
<td>3539</td>
<td>3330</td>
<td>24175</td>
</tr>
<tr>
<td>Assembly</td>
<td>160</td>
<td>2158</td>
<td>2606</td>
<td>12434</td>
</tr>
</tbody>
</table>
Programmatic formatting

Gone:

- Deciding on a style guide
- Configuring your editor
- Half of code review comments
Testing, benchmarking

Function to benchmark:

```go
func permute(todo string) []string
```

Benchmark:

```go
func BenchmarkPermute5(b *testing.B) {
    for i := 0; i < b.N; i++ {
        permute("12345")
    }
}
```

Result:

```
$ go test -test.benchmem -test.bench=.  
PASS
BenchmarkPermute5  10000   136397 ns/op  30779 B/op  1007 allocs/op
```
## Profiling

```
$ go test -test.cpuprofile=cpu.out
$ go tool pprof permute.test cpu.out
Welcome to pprof! For help, type 'help'.
(pprof) top
Total: 134 samples
   23 17.2% 17.2%  79 59.0% runtime.appendslice
   13  9.7% 26.9%  13  9.7% runtime.memmove
   11  8.2% 35.1%  16 11.9% sweepspan
   10  7.5% 42.5%  10  7.5% runtime.nanotime
    9  6.7% 49.3%  76 56.7% growslice1
    8  6.0% 55.2%  54 40.3% runtime.mallocgc
    7  5.2% 60.4% 131 97.8% ~/Users/robfig/Dropbox/Revel/permute.helper
    7  5.2% 65.7%   8  6.0% runtime.settype_flush
    6  4.5% 70.1%  22 16.4% runtime.MCache_Alloc
    6  4.5% 74.6%  12  9.0% runtime.slicerunetostring
```
Profiling
### Documentation

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>gdoc strings Split</code></td>
<td># Look up function</td>
</tr>
<tr>
<td><code>gdoc strings Split*</code></td>
<td># Look up functions named like..</td>
</tr>
<tr>
<td>`gdoc strings</td>
<td>less`</td>
</tr>
<tr>
<td>`gdoc github.com/robfig/cron</td>
<td>less`</td>
</tr>
<tr>
<td><code>gdoc -http=:8080</code></td>
<td># Browse all documentation locally</td>
</tr>
</tbody>
</table>
Deployment

"Statically-linked binary" means "copy this file and you're done"

$ go build my/program # contains all dependencies
$ GOOS=windows GOARCH=386 go build my/program # simple cross compilation
$ scp bin/program server:/srv/bin/ # copy and run
Python package managers

“What’s pip?”
“A python package manager”
“How do I install it?”
“easy_install pip”
“What’s easy_install?”
“A python package manager”

-Twitter
Performance
Performance

Framework Benchmarks

- Query a random row by ID
- Serialize the result to JSON

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 31
Date: Mon, 17 Jun 2013 20:20:25 GMT

{"id":8082,"randomNumber":7888}
### Results from EC2 m1.large

#### Peak database-access responses per second, EC2 large, single query

<table>
<thead>
<tr>
<th>Framework</th>
<th>Peak performance (higher is better)</th>
<th>Cls</th>
<th>Lng</th>
<th>Plt</th>
<th>FE</th>
<th>DB</th>
<th>Orm</th>
<th>IA</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>go</td>
<td>8,479</td>
<td>100.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>openresty</td>
<td>8,137</td>
<td>96.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>servlet</td>
<td>8,079</td>
<td>95.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>onion</td>
<td>7,559</td>
<td>89.1%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>134</td>
</tr>
<tr>
<td>vertx</td>
<td>6,408</td>
<td>75.6%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>servlet</td>
<td>5,878</td>
<td>69.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>http-kit</td>
<td>5,850</td>
<td>69.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nodejs</td>
<td>4,399</td>
<td>51.9%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>elli</td>
<td>4,198</td>
<td>49.5%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ringo</td>
<td>3,890</td>
<td>45.9%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cowboy</td>
<td>3,593</td>
<td>42.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>php</td>
<td>2,767</td>
<td>32.6%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tornado</td>
<td>1,473</td>
<td>17.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Performance (Windows)

#### Results from Windows on EC2 m1.large

<table>
<thead>
<tr>
<th>Framework</th>
<th>Peak performance (higher is better)</th>
<th>Cls</th>
<th>Lng</th>
<th>Plt</th>
<th>FE</th>
<th>DB</th>
<th>Orm</th>
<th>IA</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>go</td>
<td>8,486 I</td>
<td>100.0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nodejs</td>
<td>2,732 I</td>
<td>32.2%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>php</td>
<td>1,805 I</td>
<td>21.3%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aspnet-mvc</td>
<td>1,727 I</td>
<td>20.4%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Peak database-access responses per second, Windows on EC2 large, single query.
Performance (Anecdote)

Blog

TUESDAY, MARCH 12, 2013

How We Went from 30 Servers to 2: Go

When we built the first version of IronWorker, about 3 years ago, it was written in Ruby and the API was built on Rails. It didn’t take long for us to start getting some pretty heavy load and we quickly reached the limits of our Ruby setup. Long story short, we switched to Go. For the long story, keep reading, here’s how things went down.
Summary

Go is like a better C, from the guys that didn't bring you C++
-- Ikai Lan

Go doesn't implicitly anything.
-- Steven in golang-nuts

PSA: Go optimizes for readability, not LOC.
-- Dave Cheney on Twitter

Go is not meant to innovate programming theory. It's meant to innovate programming practice.
-- Samuel Tesla

Next steps:


Interactive tour (http://tour.golang.org/)
Thank you

Rob Figueiredo
Engineering Manager, Yext
robfig@yext.com
@robfig