TARDIS: Go for Haxe!

A talk by Elliott Stoneham at WWX2014 on 24th May 2014

- 50 years ago: watched very first Dr Who episode
- 40 years ago: started programming (punch cards)
- 20 years ago: became a manager
- 10 years ago: stopped programming
- 1 year ago: discovered Go and Haxe…

Go is like the TARDIS

A small idiosyncratic exterior,
conceals large-scale quality engineering.

Haxe is like a sonic screwdriver

An easy to use tool, built on engineering excellence,
that is useful in a wide range of situations.

overview

- Introduction to Go
- TARDIS Go transpiler
- Go —> Haxe issues
- Haxe —> Go hopes
- ? —> Haxe —> ? dream

the birth of Go in 2007

“When the three of us got started, it was pure research. The three of us got together and decided that we hated C++.

...we started off with the idea that all three of us had to be talked into every feature in the language, so there was no extraneous garbage put into the language for any reason.”

—Ken Thompson (who designed and implemented the original Unix operating system)

the objectives of Go

- *“Readable, safe, and efficient code.*
- *A build system that scales.*
- *Good concurrency support.*
- *Tools that can operate at Google-scale."*

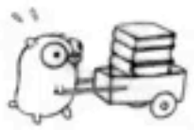
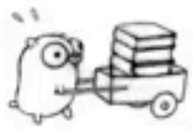
— Robert Griesemer

“Now, Go makes much more sense for the class of problems that C++ was originally intended to solve.”

–Bruce Eckel, author and
founding member of the ANSI/ISO C++ standard committee

TARDIS Go 8 Concurrent Gophers

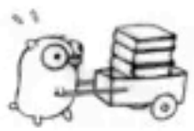
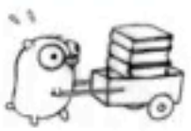
TARDIS Go example; see tardisgo.github.io

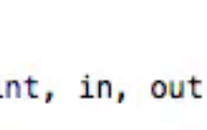

All animated gophers are running the Go code on the right. The logos show where the 2 above gophers each are in that code now. This Go code is running live, transpiled into: neko

```
func gopher(x, y *float64, state *int, in, out chan int) {
    for {
        cartLoad := pickBooks(x, y, state, in)
        pushBooks(x, y, state, cartLoad)
        fireBooks(x, y, state, cartLoad, out)
        moreBooks(x, y, state)
    }
}
```

Inspired by "Concurrency is not Parallelism (it's better)" - Rob Pike
http://concur.rspace.googlecode.com/hg/talk/concur.html



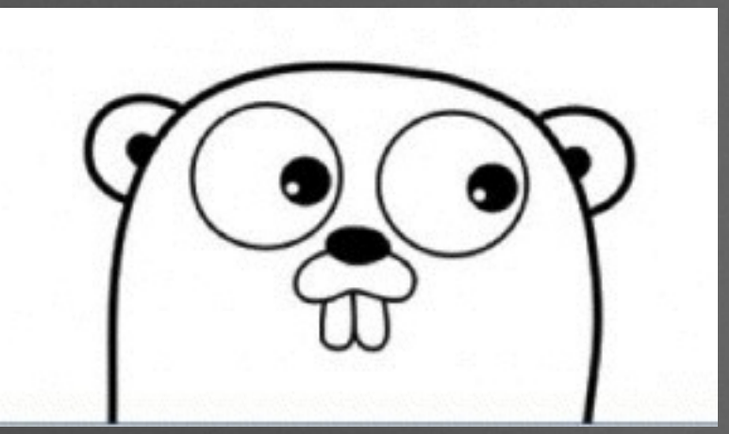



burn those C++ manuals!

Go —> HaXe + OpenFL
(running on neko target)

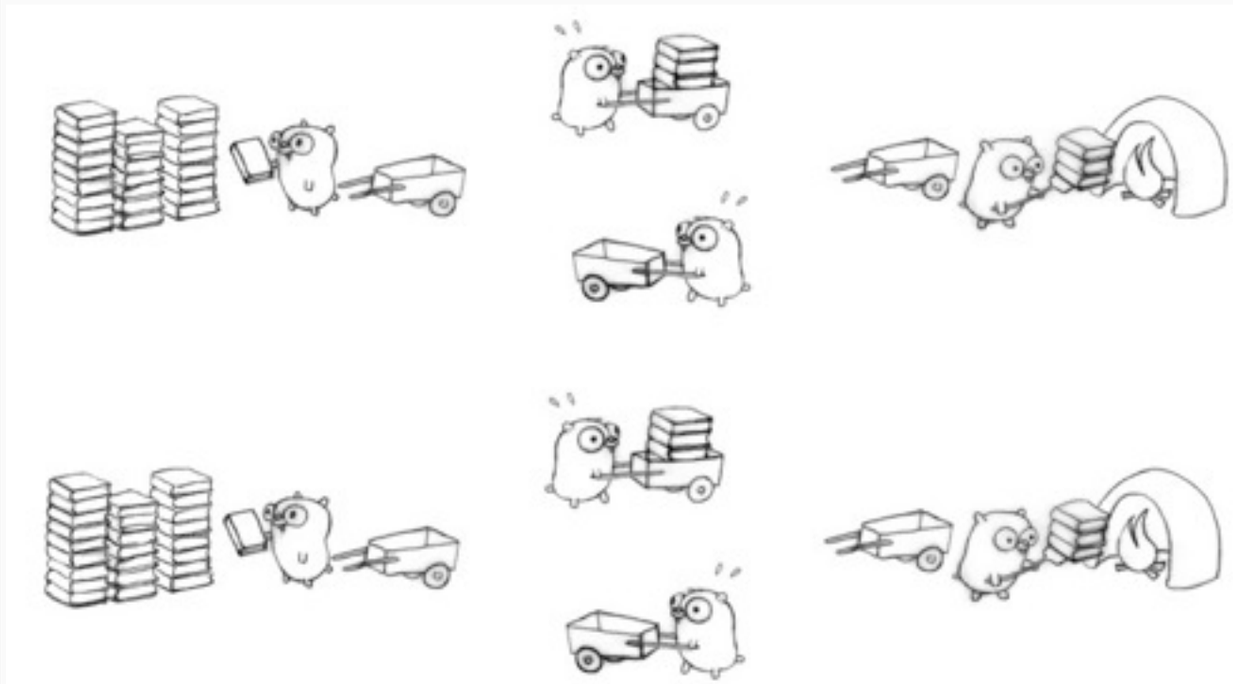
Where is Go now?

- Open-source more than 4 years, 2 official compilers
- Google-scale usage: dl.google.com, youtube.com etc.
- Non-Google: docker.com, nsq.io, juju.ubuntu.com etc.
- 800 attendees at first conference last month in Denver



Go vs Haxe

- Github repos: Go >25,600; Haxe > 1,900
- Twitter: Go >11,100; Haxe >1,700
- Google+: Go+ >13,800; Haxe >800
- TIOBE Index for March 2014: Go #36; Haxe not listed



tardisgo.github.io

TARDIS Go -> Haxe transpiler

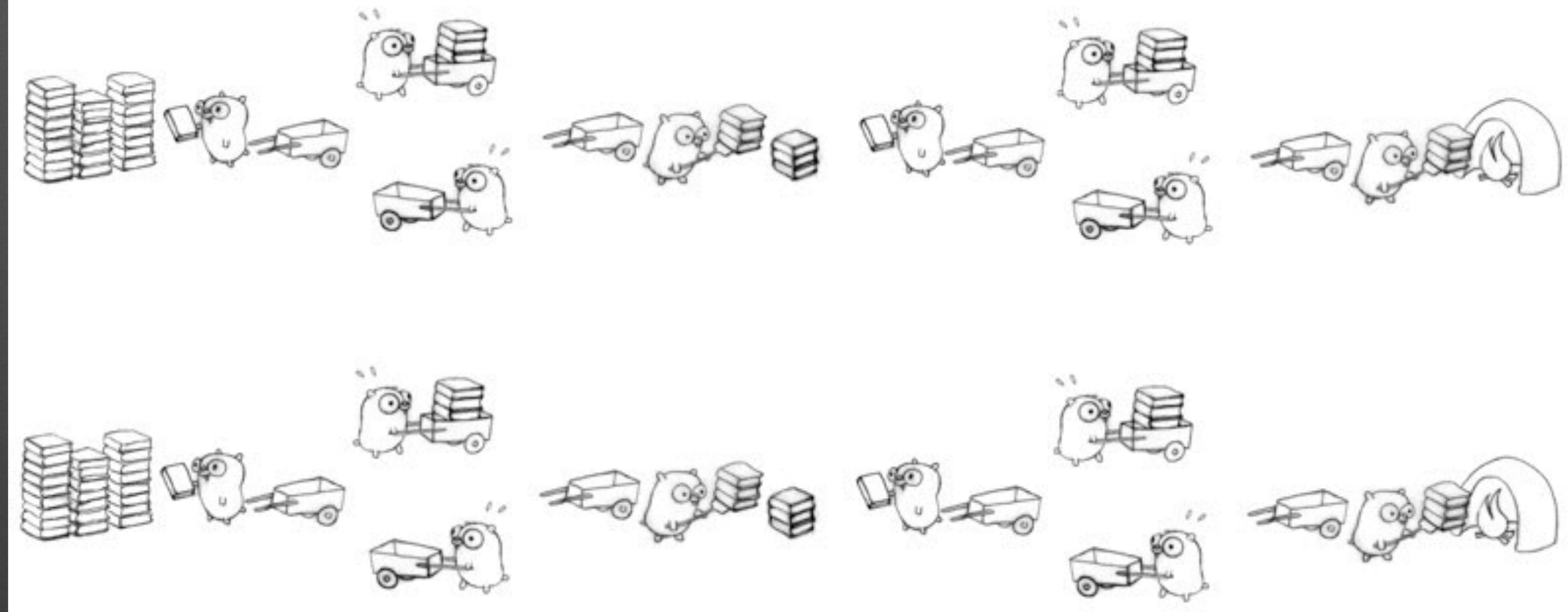
TARDIS Go objectives

- Go running on Haxe/OpenFL cross-platform targets
- Haxe calling Go libraries
- Show that Haxe is useful as an intermediate language
- Influence the future directions of Haxe and Go

concurrency

Go provides simple concurrency primitives:

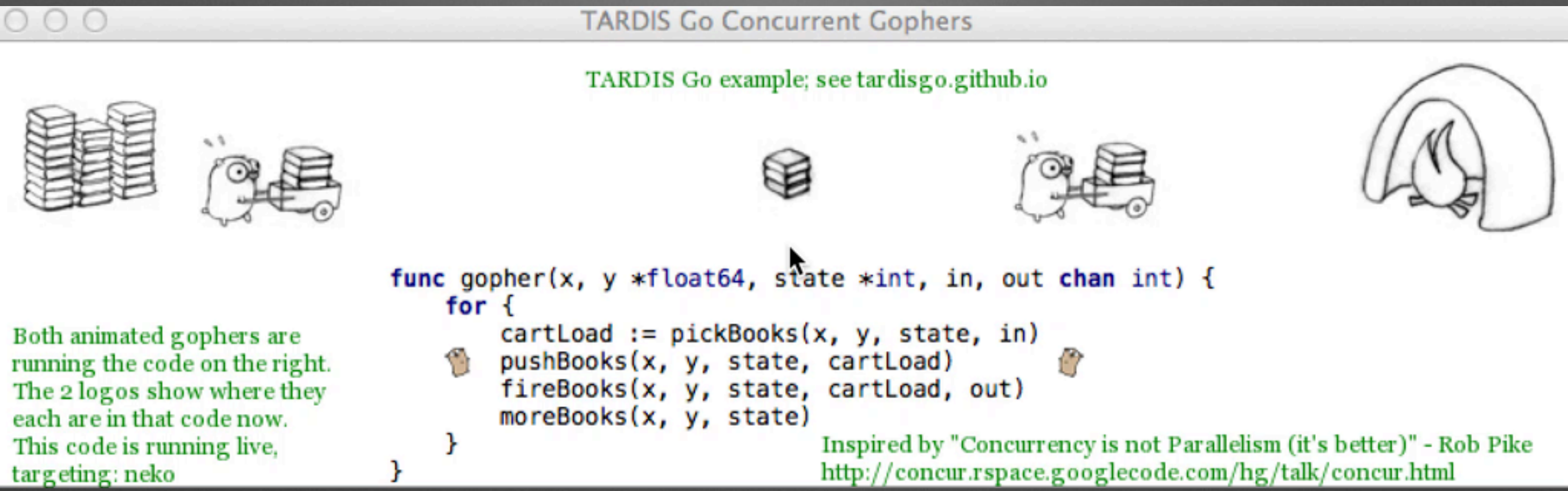
- lightweight threads (goroutines)
- typed thread-safe communication and synchronization (channels)



Concurrency is not parallelism

image from a talk by Rob Pike
http://blog.golang.org/concurrency-is-not-parallelism

tardisgo.github.io



- Each of the two sprites is a Go “goroutine"
- The pile of books in the middle is a Go “channel”

The 3 phases of the TARDIS Go transpiler

`Go program to be compiled =>`

(1) Use Go library packages to create an intermediate form that entirely describes the original Go program

`=> Single Static Assignment (SSA) form =>`

(2) Generate new code with the same behaviour

`=> Haxe =>`

(3) Use normal Haxe compilation process

The 36 SSA (Single Static Assignment) Instructions

	Value?	Instruction?
*Alloc	✔	✔
*BinOp	✔	✔
*Builtin	✔	✔
*Call	✔	✔
*ChangeInterface	✔	✔
*ChangeType	✔	✔
*Convert	✔	✔
*DebugRef		✔
*Defer		✔
*Extract	✔	✔
*Field	✔	✔
*FieldAddr	✔	✔
*Go		✔
*If		✔
*Index	✔	✔
*IndexAddr	✔	✔
*Jump		✔
*Lookup	✔	✔
*MakeChan	✔	✔
*MakeClosure	✔	✔
*MakeInterface	✔	✔
*MakeMap	✔	✔
*MakeSlice	✔	✔
*MapUpdate		✔
*Next	✔	✔
*Panic		✔
*Phi	✔	✔
*Range	✔	✔
*Return		✔
*RunDefers		✔
*Select	✔	✔
*Send		✔
*Slice	✔	✔
*Store		✔
*TypeAssert	✔	✔
*UnOp	✔	✔

```
func fact(n int) int {
    if n == 0 {
        return 1
    }
    return n * fact(n-1)
}
```

go -> ssa example

```
# Name: main.fact
# Package: main
# Location: glug.go:9:6
func fact(n int) int:
.0.entry:                                                  P:0 S:2
   t0 = n == 0:int                                        bool
   if t0 goto 1.if.then else 2.if.done
.1.if.then:                                                P:1 S:0
   return 1:int
.2.if.done:                                                P:1 S:0
   t1 = n - 1:int                                          int
   t2 = fact(t1)                                           int
   t3 = n * t2                                             int
   return t3
```

go -> ssa -> haxe

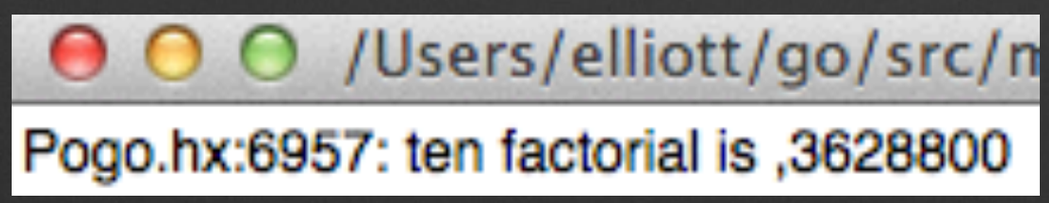
```
public function run():Pogo_main_fact {
while(true){
switch(_Next) {
case 0: // entry
this.setLatest(9,0);
this.SubFn0();
case 1: // if.then
this.setLatest(9,1);
_res= 1;
this._incomplete=false;
Scheduler.pop(this._goroutine);
return this; // return 1:int *ssa.Return @ glug.go:11:3
case 2: // if.done
this.setLatest(11,2);
this.SubFn1();
_SF1=Pogo_main_fact.call(this._goroutine,[],_t1);
_Next = -1;
return this;
case -1:
this.setLatest(13,-1);
_t2=_SF1.res();
 // _t2 = fact(t1) *ssa.Call @ glug.go:13:15
this.SubFn2();
_res= _t3;
this._incomplete=false;
Scheduler.pop(this._goroutine);
return this; // return t3 *ssa.Return @ glug.go:14:2
default: throw "Next?";}}}
private inline function SubFn0():Void {
var _t0:Bool;
_t0=(p_n==0); // _t0 = n == 0:int *ssa.BinOp @ glug.go:10:7
_Next=_t0 ? 1 : 2; // if t0 goto 1.if.then else 2.if.done *ssa.If near glug.go:10:7
}// end SubFn0
private inline function SubFn1():Void {
_t1=(p_n-1); // _t1 = n - 1:int *ssa.BinOp @ glug.go:13:17
}// end SubFn1
private inline function SubFn2():Void {
_t3=(p_n*_t2); // _t3 = n * t2 *ssa.BinOp @ glug.go:13:9
}// end SubFn2
```

```
func fact(n int) int {
    if n == 0 {
        return 1
    }
    return n * fact(n-1)
}
```

```
# Name: main.fact
# Package: main
# Location: glug.go:9:6
func fact(n int) int:
.0.entry:
   t0 = n == 0:int
   if t0 goto 1.if.then else 2.if.done
.1.if.then:
   return 1:int
.2.if.done:
   t1 = n - 1:int
   t2 = fact(t1)
   t3 = n * t2
   return t3
```

Go “fact(10)” running in JS, C++, Java, C#, PHP & Flash

```
Go->Haxe->JS (node<pogo.js):
Pogo.hx:2716: ten factorial is ,3628800

Go->Haxe->C++ (./cpp/Pogo):
Pogo.hx:2716: ten factorial is ,3628800

Go->Haxe->Java (java -jar java/java.jar):
Pogo.hx:2716: ten factorial is ,3628800

Go->Haxe->C# (mono ./cs/bin/cs.exe):
Pogo.hx:2716: ten factorial is ,3628800

Go->Haxe->PHP (php php/index.php):
Pogo.hx:2716: ten factorial is ,3628800

Go->Haxe->Flash (using flash player to test swf file):
```



Go/SSA -> Haxe issues

- Just a proof-of-concept at the moment
- Generated Haxe code is correct but very long
- Need cross-target Haxe type library, including pointers
- Need cross-target Haxe concurrency/locking API
- Need Go implementation of Haxe standard library
- ...plus some Haxe irritations

Go basic numeric types

- uint8 the set of all unsigned 8-bit integers (0 to 255)
- uint16 the set of all unsigned 16-bit integers (0 to 65535)
- uint32 the set of all unsigned 32-bit integers (0 to 4294967295)
- uint64 the set of all unsigned 64-bit integers (0 to 18446744073709551615)
- int8 the set of all signed 8-bit integers (-128 to 127)
- int16 the set of all signed 16-bit integers (-32768 to 32767)
- int32 the set of all signed 32-bit integers (-2147483648 to 2147483647)
- int64 the set of all signed 64-bit integers (-9223372036854775808 to 9223372036854775807)
- float32 the set of all IEEE-754 32-bit floating-point numbers
- float64 the set of all IEEE-754 64-bit floating-point numbers
- complex64 the set of all complex numbers with float32 real and imaginary parts
- complex128 the set of all complex numbers with float64 real and imaginary parts
- byte alias for uint8
- rune alias for int32
- uint either 32 or 64 bits
- int same size as uint
- uintptr an unsigned integer large enough to store the uninterpreted bits of a pointer value

numeric type issues

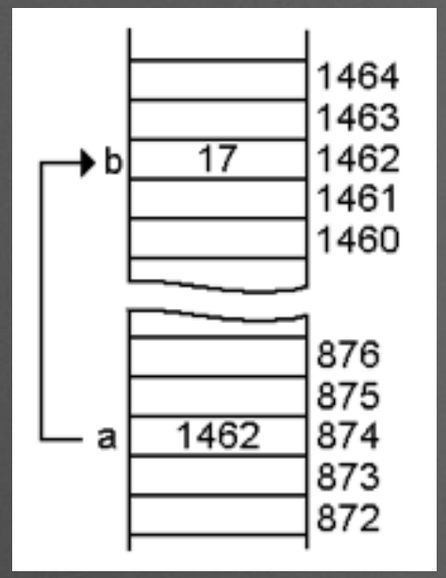
- no complex number types in Haxe
- float32 and complex64 currently modelled as float64 and complex128
- int8, int16, uint8 and uint16 each need additional processing to mask out the high-order bits and (maybe) extend the sign as part of each computation
- int64 and uint64 require emulation on platforms like JS without 64-bit arithmetic

Go non-numeric types

- Boolean types
- String types
- Array types
- Slice types
- Struct types
- Pointer types
- Function types
- Interface types
- Map types
- Channel types

string types

- Go “string” type encoding is utf-8, but Go also has the concept of a “rune” (a 32-bit full Unicode character)
- In Haxe, a string might be implemented as utf-8 or utf-16, depending on the platform
- TARDIS Go uses the native string encoding (for calling consistency) translating on the fly as required, with the runtime distinguishing between the two encodings based on the length of the ‘字’ character



pointer emulation

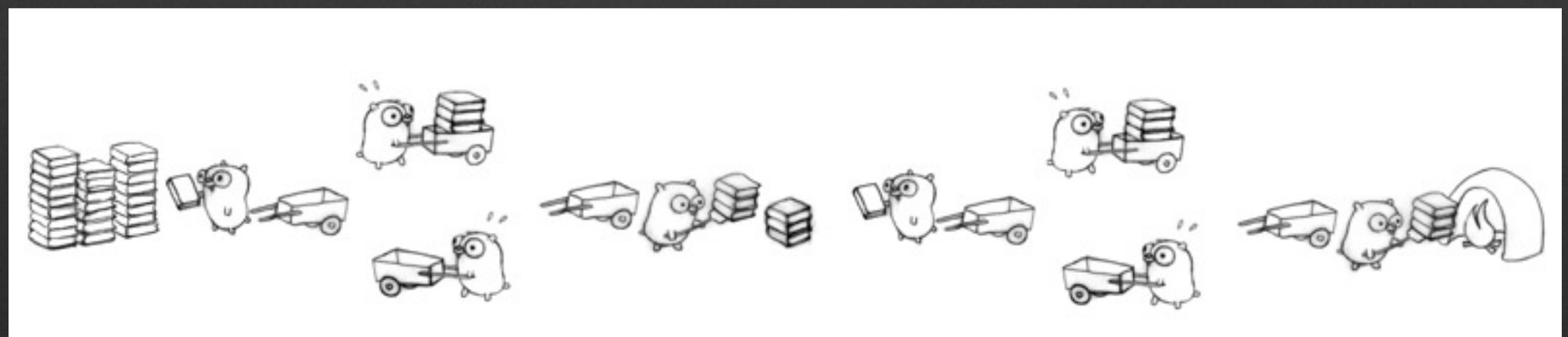
- Unlike most languages, Go pointers can’t reference outside their type, so memory can be a collection of host objects, with garbage collection by the host runtime

- Currently, emulated pointers are offsets into arrays of dynamic objects (a very simple solution that works for all targets)

- TODO: generate a new Haxe type for each possible Go pointer type so that dynamic values are not required, thus speeding up execution

goroutines

- Implemented using co-routines, emulating multiple stacks, using SSA block # as the Program Counter
- TODO: improve by using threads, for Haxe targets that support them
- TODO: Don’t implement goroutines if not used, they add a significant execution overhead and are not always required



Haxe irritations

- Auto-mapping of Int “0xFFFFFFFF” into “-1” for js and php, when they are not the same thing
- Integer overflow in PHP
- 64-bit integers were a particular issue, but may have improved since I implemented my own extensions

calling Haxe libs from Go

- Go packages beginning with an underscore define Haxe functionality; they can be auto-generated from the Haxe code or library documentation with target-specific capital letter prefixes:

```
s := string(_haxeapi.Xhaxe_Http_requestUrl(someURL))
switch tardisgolib.Platform() {
case "cpp": _haxeapi.Pcpp_Lib_println(s)
case "php": _haxeapi.Hphp_Lib_println(s)
default: println(s)
}
```

using Go vars from Haxe

- all Go globals are pointers
- all pointer targets are Dynamic, which needs to change

```
• x = Go.pkgName_varName.load();
• Go.pkgName_varName.store(y);
• aPtr = Go.pkgName_varName.addr(42);
```

calling Go code from Haxe

- `x=Go.Go_pkgNam_funcNam.callFromHaxe(y,z);`
- TODO: there is a need to build a simple "methodFromHaxe" calling interface, as using Go methods from Haxe is currently too complex, for example:
  - `_SF1=Go__star_main_dot_rect_area.call(this._goroutine,[],_t0); // by reference`
  - `_SF2=Go_main_dot_rect_perim.call(this._goroutine,[],Deep.copy(_t5)); // by value`

Haxe -> Go

- Efficient server-side operation & concurrency support
- Shared runtime library with TARDIS Go
- https://code.google.com/p/haxego/ by Lee Sylvester seems to be inactive
- Maybe write a proof of concept for the “Custom JS Generator” approach, like hx2dart by Andrew Vernon?

? -> Haxe -> ?

- TARDIS Go
- Tarwin's AS3 to Haxe Conversion Script
- Typescript to Haxe Conversion Script
- PHP to Haxe Conversion Utility
- CS2HX - C# to haXe converter
- Write a LLVM or GCC compiler Haxe back-end?

http://tardisgo.github.io

- improve TARDIS Go to be a usable tool
- write cross-target Haxe libraries to implement Go types, pointers and concurrency
- write Haxe -> Go target
- write LLVM -> Haxe target

Image Sources

- Gopher Logo by Renée French
- TARDIS image (c) BBC from http://bbc.co.uk
- http://commons.wikimedia.org/
- http://www.wikipedia.org/
- http://memegenerator.net/Your-Country-Needs-You
- Project related images from relevant project sites
- Other images self-created