

# BEAT

Where next?

# POINTERS VS VALUES

*...DEBUGGED*



**Vangelis Katikaridis | Sr. Backend Engineer**

[github.com/drakos74](https://github.com/drakos74)

Twitter: @vangkos

# BEAT

# Using pointers or values as Method arguments

`"This is how far code review can bring you!"`

# The Theory

## **\*pointers**

- Allow for struct mutation
- Avoid copying overhead

But ...

- Not concurrently safe
- Might escape to the heap
  - Gc overhead
  - Heap expansion cost

## **values**

- Immutability
- Lives in the stack (escape analysis)

But ...

- Processing time overhead (copying)

# The Baseline

## model

```
type obj interface {
    GetI() int64
    GetF() float64
}

type tiny struct {
    a int64
    aa float64
    aaa string
}

type mini struct {
    a int64
    b int64
    c int64
    aa float64
    bb float64
    cc float64
    aaa string
    bbb string
    ccc string
}

type tera struct {
    ...
}
```

## methods

```
func calcTiny(o tiny) (float64, error) {
    return float64(o.a) / o.aa, nil
}

func calcTinyP(o *tiny) (float64, error) {
    return float64(o.a) / o.aa, nil
}

func calcTera(o tera) (float64, error) {
    return float64(o.a) / o.aa, nil
}

func calcTeraP(o *tera) (float64, error) {
    return float64(o.a) / o.aa, nil
}

func calcIcfc(o obj) (float64, error) {
    return float64(o.GetI()) / o.GetF(), nil
}

...
```

## benchmarks

```
func BenchmarkTiny(b *testing.B) {
    o := createTiny()

    for i := 0; i < b.N; i++ {
        avoidCompilerOptimisation(calcTiny(o))
    }
}

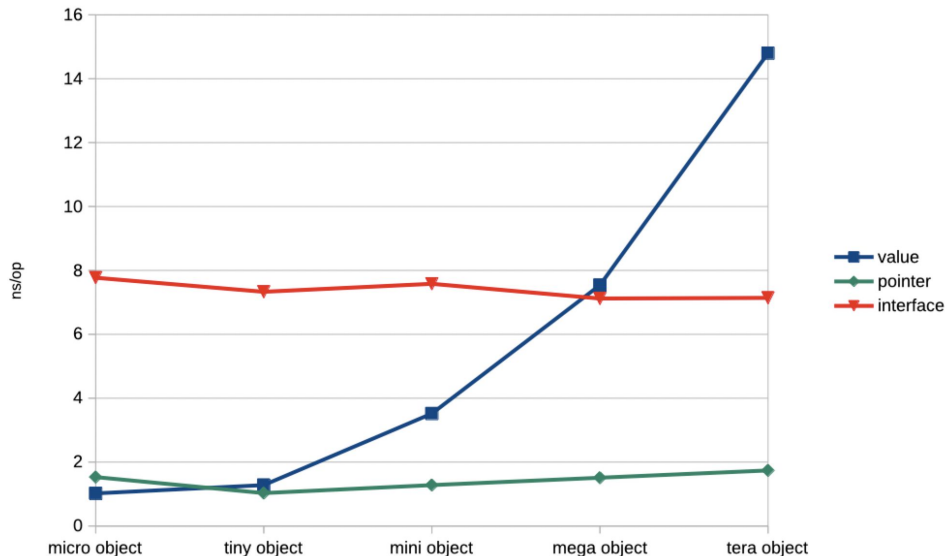
func BenchmarkTinyP(b *testing.B) {
    o := createTiny()

    for i := 0; i < b.N; i++ {
        avoidCompilerOptimisation(calcTinyP(&o))
    }
}

// avoid compiler optimisations
var F = 0.0
func avoidCompilerOptimisation(f float64, err error) {
    F += f
}
```

# ns/op

- **Pointers & Interfaces** seem constant in performance.  
(constant access pattern)
- **Value** performance is degrading “fast”.
- **Pointer** and **Value** performance is very much comparable for small object sizes \*.



“Copying objects within a cache line is the roughly equivalent to copying a single pointer” <https://segment.com/blog/allocation-efficiency-in-high-performance-go-services/>

\* One thing to note though, is even if the object is less than 64 bytes, it might spread across 2 cache lines, whereas the pointer always occupies only one.

```
type micro struct {  
  a int64  
  aa float64  
}
```

```
type tiny struct {  
  a int64  
  aa float64  
  aaa string  
}
```

```
type mini struct {  
  a int64  
  b int64  
  c int64  
  aa float64  
  bb float64  
  cc float64  
  aaa string  
  bbb string  
  ccc string  
}
```

```
type mega struct {  
  a int64 // ( x 11 )  
  ...  
  aa float64 // ( x 11 )  
  ...  
  aaa string // ( x 11 )  
  ...  
}
```

```
type tera struct {  
  a int64 // ( x 26 )  
  ...  
  aa float64 // ( x 26 )  
  ...  
  aaa string // ( x 26 )  
  ...  
}
```

# Increasing Iterations

```
const iterations = 100000

func BenchmarkTiny(b *testing.B) {
    o := createTiny()

    for i := 0; i < b.N; i++ {
        for a := 0; a < iterations; a++ {
            avoidCompilerOptimisation(calcTiny(o))
        }
    }
}
```

- **Performance** scales consistently (almost linearly) ... as the iterations increase
- **Allocations** for *pointer* and *value* are the same ... for *interfaces* they become considerably more relevant

value	ns/op	B/op	allocs/op
BenchmarkTiny	1.28	0	0
BenchmarkTiny_x_100_000	219334	0	0
BenchmarkTiny_x_100_000_000	227079453	6	0
pointer			
BenchmarkTinyP	1.03	0	0
BenchmarkTinyP_x_100_000	216990	0	0
BenchmarkTinyP_x_100_000_000	218180127	6	0
interface			
BenchmarkTinyIfc	7.33	0	0
BenchmarkTinyIfc_x_100_000	723065	0	0
BenchmarkTinyIfc_x_100_000_000	716009788	32	1

**BEAT**

# Assembly Analysis

go tool compile -S pointer\_vs\_value/code/\* > assembly.s

MOVQ : **copy** quadrant  
MOVSD : **move** doubleword  
XORPS : **clear register** (usually what gcc does in C++)  
PCDATA : **garbage collection** metadata

\*pointer

```
✓ "" .calcMicroObjectWithPointer STEXT size=178 args=0x20 locals=0x18
(args/pointer_receiver.go:135) TEXT "" .calcMicroObjectWithPointer(SB), ABIInternal, $24-32
(args/pointer_receiver.go:135) MOVQ (TLS), CX
(args/pointer_receiver.go:135) CMPQ SP, 16(CX)
(args/pointer_receiver.go:135) JLS 168
(args/pointer_receiver.go:135) SUBQ $24, SP
(args/pointer_receiver.go:135) MOVQ BP, 16(SP)
(args/pointer_receiver.go:135) LEAQ 16(SP), BP
(args/pointer_receiver.go:135) FUNCDATA $0, gcllocals·305be4e7b8a18f2d72ac305e38ac4e7b(SB)
(args/pointer_receiver.go:135) FUNCDATA $1, gcllocals·7d2d5fca880364273f07d5820a76fef4(SB)
(args/pointer_receiver.go:135) FUNCDATA $2, gcllocals·bfec7e55b3f043d1941c093912808913(SB)
(args/pointer_receiver.go:136) PCDATA $0, $1
(args/pointer_receiver.go:136) PCDATA $1, $1
(args/pointer_receiver.go:136) MOVQ "" .o+32(SP), AX
(args/pointer_receiver.go:136) MOVSD 8(AX), X0
(args/pointer_receiver.go:136) XORPS X1, X1
(args/pointer_receiver.go:136) UCOMISD X1, X0
(args/pointer_receiver.go:136) JNE 129
(args/pointer_receiver.go:136) JPS 129
(args/pointer_receiver.go:137) PCDATA $0, $0
(<unknown line number>) NOP
```

value

```
» 1 1 << "" .calcMicroObject STEXT size=176 args=0x28 locals=0x18
2 2 (args/pointer_receiver.go:91) TEXT "" .calcMicroObject(SB), ABIInternal, $24-40
3 3 (args/pointer_receiver.go:91) MOVQ (TLS), CX
4 4 (args/pointer_receiver.go:91) CMPQ SP, 16(CX)
5 5 (args/pointer_receiver.go:91) JLS 166
6 6 (args/pointer_receiver.go:91) SUBQ $24, SP
7 7 (args/pointer_receiver.go:91) MOVQ BP, 16(SP)
8 8 (args/pointer_receiver.go:91) LEAQ 16(SP), BP
9 9 (args/pointer_receiver.go:91) FUNCDATA $0, gcllocals·4284da0ffddacd9b3f4010ab6a9b04d6(SB)
10 10 (args/pointer_receiver.go:91) FUNCDATA $1, gcllocals·69c1753bd5f81501d95132d08af04464(SB)
11 11 (args/pointer_receiver.go:91) FUNCDATA $2, gcllocals·bfec7e55b3f043d1941c093912808913(SB)
12 12 (args/pointer_receiver.go:92) PCDATA $0, $0
13 13 (args/pointer_receiver.go:92) PCDATA $1, $0
14 14 (args/pointer_receiver.go:92) MOVSD "" .o+40(SP), X0
15 15 (args/pointer_receiver.go:92) XORPS X1, X1
16 16 (args/pointer_receiver.go:92) UCOMISD X1, X0
17 17 (args/pointer_receiver.go:92) JNE 125
18 18 (args/pointer_receiver.go:92) JPS 125
19 19 (args/pointer_receiver.go:92) JPS 125
20 20 (<unknown line number>) NOP
21 21
```

X0/X1 : registers

AX : primary accumulator (register)

SP : stack pointer

- **Pointer** assembly code does an extra copy call  
+ adds some gc overhead

**BEAT**



## interface

```
00000000 .calcWithPointer STEXT size=248 args=0x28 locals=0x20
0x0000 00000 (args/pointer_receiver.go:128) TEXT    ""_.calcWithPointer(SB), ABIInternal, $32-40
0x0000 00000 (args/pointer_receiver.go:128) MOVQ     (TLS), CX
0x0009 00009 (args/pointer_receiver.go:128) CMPQ     SP, 16(CX)
0x000d 00013 (args/pointer_receiver.go:128) JLS     238
0x0013 00019 (args/pointer_receiver.go:128) SUBQ     $32, SP
0x0017 00023 (args/pointer_receiver.go:128) MOVQ     BP, 24(SP)
0x001c 00028 (args/pointer_receiver.go:128) LEAQ     24(SP), BP
0x0021 00033 (args/pointer_receiver.go:128) FUNCDATA $0, gcllocals·b3e6cef2da4123bb4fb758b92dba3371(SB)
0x0021 00033 (args/pointer_receiver.go:128) FUNCDATA $1, gcllocals·7d2d5fca80364273fb07d5820a76fef4(SB)
0x0021 00033 (args/pointer_receiver.go:128) FUNCDATA $2, gcllocals·457447a82af07995fb3f7c66ee908834(SB)
0x0021 00033 (args/pointer_receiver.go:129) PCDATA $0, $0
0x0021 00033 (args/pointer_receiver.go:129) PCDATA $1, $0
0x0021 00033 (args/pointer_receiver.go:129) MOVQ     ""_.o+40(SP), AX
0x0026 00038 (args/pointer_receiver.go:129) MOVQ     24(AX), CX
0x002a 00042 (args/pointer_receiver.go:129) PCDATA $0, $1
0x002a 00042 (args/pointer_receiver.go:129) MOVQ     ""_.o+48(SP), DX
0x002f 00047 (args/pointer_receiver.go:129) PCDATA $0, $0
0x002f 00047 (args/pointer_receiver.go:129) MOVQ     DX, (SP)
0x0033 00051 (args/pointer_receiver.go:129) CALL     CX
0x0035 00053 (args/pointer_receiver.go:129) MOVSD    8(SP), X0
0x003b 00059 (args/pointer_receiver.go:129) XORPS    X1, X1
0x003e 00062 (args/pointer_receiver.go:129) UCOMISD   X1, X0
0x0042 00066 (args/pointer_receiver.go:129) JNE     145
0x0044 00068 (args/pointer_receiver.go:129) JPS     145
0x0046 00070 (args/pointer_receiver.go:130) PCDATA $1, $1
0x0046 00070 (<unknown line number>) NOP
```

## value

```
>> 1 1 << ""_.calcMicroObject STEXT size=176 args=0x28 locals=0x18
2 2 (args/pointer_receiver.go:91) TEXT    ""_.calcMicroObject(SB), ABIInternal, $24-40
3 3 (args/pointer_receiver.go:91) MOVQ     (TLS), CX
4 4 (args/pointer_receiver.go:91) CMPQ     SP, 16(CX)
5 5 (args/pointer_receiver.go:91) JLS     166
6 6 (args/pointer_receiver.go:91) SUBQ     $24, SP
7 7 (args/pointer_receiver.go:91) MOVQ     BP, 16(SP)
8 8 (args/pointer_receiver.go:91) LEAQ     16(SP), BP
9 9 (args/pointer_receiver.go:91) FUNCDATA $0, gcllocals·4284da0ffddacd9b3f4010ab6a9b04d6(SB)
10 10 (args/pointer_receiver.go:91) FUNCDATA $1, gcllocals·69c1753bd5f81501d95132d08af04464(SB)
11 11 (args/pointer_receiver.go:91) FUNCDATA $2, gcllocals·bfec7e55b3f043d1941c093912808913(SB)
12 12 (args/pointer_receiver.go:92) PCDATA $0, $0
13 13 (args/pointer_receiver.go:92) PCDATA $1, $0
14 14
15 15
16 16
17 17
18 18
19 19
20 20
21 21 (args/pointer_receiver.go:92) MOVSD    ""_.o+40(SP), X0
22 22 (args/pointer_receiver.go:92) XORPS    X1, X1
23 23 (args/pointer_receiver.go:92) UCOMISD   X1, X0
24 24 (args/pointer_receiver.go:92) JNE     125
25 25 (args/pointer_receiver.go:92) JPS     125
26 26
27 27 (<unknown line number>) NOP
```

- **Interfaces** need by default 2 pointers

See the "iface" struct <https://github.com/golang/go/blob/bf86aec25972f3a100c3aa58a6abcbcc35bdea49/src/runtime/runtime2.go#L143-L146>

- **Pointer** handling for interfaces must be "by the book"

## interface

```
0x006a 00106 ($GOROOT/src/errors/errors.go:59) PCDATA $0, $2
0x006a 00106 ($GOROOT/src/errors/errors.go:59) MOVQ CX, (AX)
0x006d 00109 (args/pointer_receiver.go:130) XORPS X0, X0
0x0070 00112 (args/pointer_receiver.go:130) MOVSD X0, [r1+56(SP)]
0x0076 00118 (args/pointer_receiver.go:130) PCDATA $0, $3
0x0076 00118 (args/pointer_receiver.go:130) PCDATA $1, $2
0x0076 00118 (args/pointer_receiver.go:130) LEAQ go.itab.*errors.errorString,error(SB), CX
0x007d 00125 (args/pointer_receiver.go:130) PCDATA $0, $2
0x007d 00125 (args/pointer_receiver.go:130) MOVQ CX, [r2+64(SP)]
0x0082 00130 (args/pointer_receiver.go:130) PCDATA $0, $0
0x0082 00130 (args/pointer_receiver.go:130) MOVQ AX, [r2+72(SP)]
0x0087 00135 (args/pointer_receiver.go:130) MOVQ 24(SP), BP
0x008c 00140 (args/pointer_receiver.go:130) ADDQ $32, SP
0x0090 00144 (args/pointer_receiver.go:130) RET
0x0091 00145 (args/pointer_receiver.go:132) PCDATA $1, $0
0x0091 00145 (args/pointer_receiver.go:132) MOVQ [r0+48(SP), AX]
0x0096 00150 (args/pointer_receiver.go:132) MOVQ 32(AX), CX
0x009a 00154 (args/pointer_receiver.go:132) PCDATA $0, $1
0x009a 00154 (args/pointer_receiver.go:132) MOVQ [r0+48(SP), DX]
0x009f 00159 (args/pointer_receiver.go:132) PCDATA $0, $0
0x009f 00159 (args/pointer_receiver.go:132) MOVQ DX, (SP)
0x00a3 00163 (args/pointer_receiver.go:132) CALL CX
0x00a5 00165 (args/pointer_receiver.go:132) MOVQ 8(SP), AX
0x00aa 00170 (args/pointer_receiver.go:132) MOVQ AX, [autotmp_12+16(SP)]
0x00af 00175 (args/pointer_receiver.go:132) MOVQ [r0+48(SP), CX]
0x00b4 00180 (args/pointer_receiver.go:132) MOVQ 24(CX), CX
0x00b8 00184 (args/pointer_receiver.go:132) PCDATA $0, $1
0x00b8 00184 (args/pointer_receiver.go:132) PCDATA $1, $1
0x00b8 00184 (args/pointer_receiver.go:132) MOVQ [r0+48(SP), DX]
0x00bd 00189 (args/pointer_receiver.go:132) PCDATA $0, $0
0x00bd 00189 (args/pointer_receiver.go:132) MOVQ DX, (SP)
0x00c1 00193 (args/pointer_receiver.go:132) CALL CX
0x00c3 00195 (args/pointer_receiver.go:132) MOVQ [autotmp_12+16(SP), AX]
0x00c8 00200 (args/pointer_receiver.go:132) XORPS X0, X0
0x00cb 00203 (args/pointer_receiver.go:132) CVTSQ2SD AX, X0
0x00d0 00208 (args/pointer_receiver.go:132) DIVSD 8(SP), X0
0x00d6 00214 (args/pointer_receiver.go:132) MOVSD X0, [r1+56(SP)]
0x00dc 00220 (args/pointer_receiver.go:132) PCDATA $1, $2
0x00dc 00220 (args/pointer_receiver.go:132) XORPS X0, X0
0x00df 00223 (args/pointer_receiver.go:132) MOVUPS X0, [r2+64(SP)]
0x00e4 00228 (args/pointer_receiver.go:132) MOVQ 24(SP), BP
0x00e9 00233 (args/pointer_receiver.go:132) ADDQ $32, SP
0x00ed 00237 (args/pointer_receiver.go:132) RET
0x00ee 00238 (args/pointer_receiver.go:132) NOP
0x00ee 00238 (args/pointer_receiver.go:128) PCDATA $1, $-1
0x00ee 00238 (args/pointer_receiver.go:128) PCDATA $0, $-1
0x00ee 00238 (args/pointer_receiver.go:128) CALL runtime.morestack_noctxt(SB)
0x00f3 00243 (args/pointer_receiver.go:128) JMP 0
```

## value

```
38 38 ($GOROOT/src/errors/errors.go:59) PCDATA $0, $1
39 39 ($GOROOT/src/errors/errors.go:59) MOVQ CX, (AX)
40 40 ($GOROOT/src/errors/errors.go:59) MOVQ CX, (AX)
41 41 (args/pointer_receiver.go:93) XORPS X0, X0
42 42 (args/pointer_receiver.go:93) MOVSD X0, [r1+48(SP)]
43 43 (args/pointer_receiver.go:93) PCDATA $0, $2
44 44 (args/pointer_receiver.go:93) PCDATA $1, $1
45 45 (args/pointer_receiver.go:93) LEAQ go.itab.*errors.errorString,error(SB), CX
46 46 (args/pointer_receiver.go:93) PCDATA $0, $1
47 47 (args/pointer_receiver.go:93) MOVQ CX, [r2+56(SP)]
48 48 (args/pointer_receiver.go:93) PCDATA $0, $0
49 49 (args/pointer_receiver.go:93) MOVQ AX, [r2+64(SP)]
50 50 (args/pointer_receiver.go:93) MOVQ 16(SP), BP
51 51 (args/pointer_receiver.go:93) ADDQ $24, SP
52 52 (args/pointer_receiver.go:93) RET
53 53 (args/pointer_receiver.go:95) PCDATA $1, $0
54 54 (args/pointer_receiver.go:95) MOVQ [r0+32(SP), AX]
55 55
56 56
57 57
58 58
59 59
60 60
61 61
62 62
63 63
64 64
65 65
66 66
67 67
68 68
69 69
70 70
71 71
72 72 (args/pointer_receiver.go:95) XORPS X1, X1
73 73 (args/pointer_receiver.go:95) CVTSQ2SD AX, X1
74 74 (args/pointer_receiver.go:95) DIVSD X0, X1
75 75 (args/pointer_receiver.go:95) MOVSD X1, [r1+48(SP)]
76 76 (args/pointer_receiver.go:95) PCDATA $1, $1
77 77 (args/pointer_receiver.go:95) XORPS X0, X0
78 78 (args/pointer_receiver.go:95) MOVUPS X0, [r2+56(SP)]
79 79 (args/pointer_receiver.go:95) MOVQ 16(SP), BP
80 80 (args/pointer_receiver.go:95) ADDQ $24, SP
81 81 (args/pointer_receiver.go:95) RET
82 82 (args/pointer_receiver.go:95) NOP
83 83 (args/pointer_receiver.go:91) PCDATA $1, $-1
84 84 (args/pointer_receiver.go:91) PCDATA $0, $-1
85 85 (args/pointer_receiver.go:91) CALL runtime.morestack_noctxt(SB)
86 86 (args/pointer_receiver.go:91) JMP 0
```

- **Interfaces** also must do more work at runtime while their methods are being invoked. This seems to be the main differentiator.

# Escape Analysis

go test pointer\_vs\_value/code/\* -gcflags "-m -m" -run none -benchmem -memprofile mem.out &> escape.out

- **Pointers** do not escape to the heap in this implementation
- **Interfaces** always do ...

```
// value
func calcTera(o tera) (float64, error) {
    return float64(o.a) / o.aa, nil
}                                     // calcTera o does not escape

// interface
func calcIfc(o obj) (float64, error) {
    return float64(o.GetI()) / o.GetF(), nil
}                                     // leaking param: o

// pointer
func calcTeraP(o *teraObject) (float64, error) {
    return float64(o.a) / o.aa, nil
}                                     // calcTeraP o does not escape
```

# Garbage Collection

... measuring with un-certainty

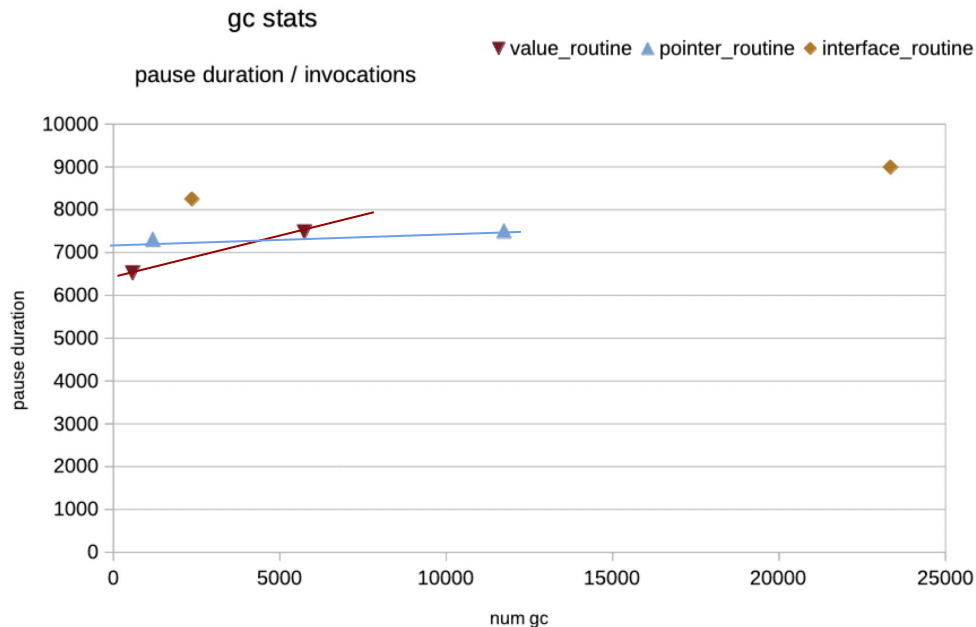
```
func trackGC(caller string) {  
    runtime.GC()  
  
    stats := debug.GCStats{}  
    debug.ReadGCStats(&stats)  
  
    println(fmt.Sprintf("\n%s , %v , %v , %v , %v\n",  
        caller,  
        stats.NumGC,  
        sum(stats.Pause),  
        stats.PauseQuantiles,  
        stats.PauseTotal))  
    //debug.FreeOSMemory() // consider cleaning up for the upcoming test  
}  
  
func sum (durations []time.Duration) int64 {  
    var f int64 = 0  
    for _, d := range durations {  
        f += d.Microseconds()  
    }  
    return f  
}
```

```
const timeout = 100 * time.Second
```

```
func runRoutine(exec func(o tera), name string) {  
    ch := make(chan tera, 1)  
    ctx, cnl := context.WithCancel(context.Background())  
  
    go func() {  
        <-time.After(timeout)  
        cnl()  
    }()  
  
    go func() {  
        start := time.Now()  
        for {  
            select {  
            case <-ctx.Done():  
                duration := time.Now().Sub(start).Milliseconds()  
                fmt.Println(fmt.Sprintf("Time passed [%v]", duration))  
                close(ch)  
                return  
            default:  
                ch <- createTeraObject()  
            }  
        }  
    }()  
  
    i := 0  
    for o := range ch {  
        exec(o)  
        i++  
    }  
  
    println(fmt.Sprintf("invocations = %v", i))  
    trackGC(name)  
}
```

BEAT

Test duration (sec)	method	time passed	invocations	num GC	pause duration (ms)
10	value	10001	2723703	580	6522
100	value	100001	26778210	5742	7476
10	pointer	10001	2835459	1188	7308
100	pointer	100000	27850534	11740	7503
10	interface	10000	2604248	2362	8249
100	interface	100000	25550879	23350	8994



- Invocation count and duration are consistent across the tests
- Value receivers seem to degrade in performance faster (slope)
- Although invocations of gc rise fast, total pause duration stays under control
- Need a broader result set

# So... who won?

*... It's all about the implementation*

## Escape analysis

Not all pointers escape to the heap.  
It depends on invocation details and object sizes.  
(Even value receivers can escape to the heap)

see [https://golang.org/doc/faq#stack\\_or\\_heap](https://golang.org/doc/faq#stack_or_heap)

## Assembly correlation

There are some expected differences.  
But in the end, it all boils down to the cpu intrinsics.

see <https://segment.com/blog/allocation-efficiency-in-high-performance-go-services/>

## Design

Interfaces provide abstraction (no tight coupling)

But harm performance ...

NOTE : With large amount of data and longer running processes, this penalty is amortised well enough.

## Performance

Pointers seem to behave optimally regarding performance.

Due to modern cpu optimizations (pre-fetch etc...).

see [https://github.com/teh-cmc/go-internals/tree/master/chapter2\\_interfaces#overhead](https://github.com/teh-cmc/go-internals/tree/master/chapter2_interfaces#overhead)

## Garbage collection

The garbage collector is “far too sophisticated” to care  
... in the end.

see <https://blog.golang.org/ismmkeynote>

**BEAT**

# Additional Considerations

## skipped points ...

- Avoid compiler optimisations  
`gcflags "-n"`  
=> No considerable difference
- Avoid compiler inlining  
`gcflags "-l"`  
=> No considerable difference  
(mostly regarding the error handling code)
- Track gc on benchmarks  
=> Inconclusive results



**BEAT**

# Where next?

## Further investigation ...

- Simplify benchmark code by using arrays  
(Thanks to Erik for the tip 😊)
- Investigate with different method bodies  
(See Appendix A)
- Investigate breaking conditions i.e. OOM  
boundary  
(TBD in Appendix B)
- Compare detailed escape analysis output  
`gcflags "-m -m -m -m"`
- Run benchmarks on different cpus
- Investigate the same for method return  
elements





# References & Further Readings

[The code]

- [https://github.com/drakos74/go-bench/tree/master/pointer\\_vs\\_value](https://github.com/drakos74/go-bench/tree/master/pointer_vs_value)

Benchmarks

- <https://dave.cheney.net/2013/06/30/how-to-write-benchmarks-in-go>
- <https://stackimpact.com/blog/practical-golang-benchmarks/>

Go assembly

- <https://www.tutorialdocs.com/article/go-assembly-code.html>
- [https://github.com/teh-cmc/go-internals/blob/master/chapter1\\_assembly\\_primer/README.md](https://github.com/teh-cmc/go-internals/blob/master/chapter1_assembly_primer/README.md)
- [https://class.ece.uw.edu/469/hauck/lectures/02\\_Assembly.pdf](https://class.ece.uw.edu/469/hauck/lectures/02_Assembly.pdf)

Escape Analysis & Memory Allocation 👍

- <http://www.agardner.me/golang/garbage/collection/gc/escape/analysis/2015/10/18/go-escape-analysis.html>
- <https://www.ardanlabs.com/blog/2017/05/language-mechanics-on-escape-analysis.html>
- <https://segment.com/blog/allocation-efficiency-in-high-performance-go-services/>
- <https://blog.learngoprogramming.com/a-visual-guide-to-golang-memory-allocator-from-ground-up-e132258453ed>

Garbage collector

- <https://dave.cheney.net/high-performance-go-workshop/dotgo-paris.html>
- <https://medium.com/@vCabbage/go-are-pointers-a-performance-optimization-a95840d3ef85>

**BEAT**

# Appendix A

Extend implementation to  
access more object properties

// access ALL object properties

```
func calcGiga(o giga) (float64, error) {  
    return float64(  
        o.a+o.b+o.c+o.d+o.e+o.f+o.g+o.h+o.i+  
        o.j+o.k+o.l+o.m+o.n+o.o+o.p+o.q+  
        o.r+o.s+o.t+o.u+o.v+o.y+o.x+o.w+o.z) /  
        (o.aa+o.bb+o.cc+o.dd+o.ee+o.fff+o.gg+o.hh+  
        o.ii+o.jj+o.kk+o.ll+o.mm+o.nn+o.oo+o.pp+o.qq+  
        o.rr+o.ss+o.tt+o.uu+o.vv+o.yy+o.xx+o.ww+o.zz),  
        nil  
    }
```

```
func calcGigaP(o *giga) (float64, error) {  
    // we skip error checking here for simplicity  
    return float64(  
        o.a+o.b+o.c+o.d+o.e+o.f+o.g+o.h+o.i+  
        o.j+o.k+o.l+o.m+o.n+o.o+o.p+o.q+  
        o.r+o.s+o.t+o.u+o.v+o.y+o.x+o.w+o.z) /  
        (o.aa+o.bb+o.cc+o.dd+o.ee+o.fff+o.gg+o.hh+o.ii+  
        o.jj+o.kk+o.ll+o.mm+o.nn+o.oo+o.pp+o.qq+  
        o.rr+o.ss+o.tt+o.uu+o.vv+o.yy+o.xx+o.ww+o.zz),  
        nil  
    }
```

\*pointer

```
***.calcGigaObjectWithPointer STEX nospit size=373 args=0x108 locals=0x0  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:182) TEXT ***.calcGigaObjectWithPointer(S0)  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:182) FUNCDATA $0, gclicals:305be4e7b0a18f2  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:182) FUNCDATA $1, gclicals:7d2d5fca0836427  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:182) FUNCDATA $2, gclicals:9fb7f9986f647f1  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:185) PCDATA $0, $1  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:185) PCDATA $1, $1  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:185) MOVQ ***.o+8(SP), AX  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:185) MOVQ (AX), CX  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:185) ADDQ 8(AX), CX  
0x0000 00012 (pointer_vs_value/pointer_receiver/func.g0:185) ADDQ 16(AX), CX  
0x0010 00016 (pointer_vs_value/pointer_receiver/func.g0:185) ADDQ 24(AX), CX  
...  
0x013a 00314 (pointer_vs_value/pointer_receiver/func.g0:190) ADDSD 392(AX), X0  
0x0142 00322 (pointer_vs_value/pointer_receiver/func.g0:190) ADDSD 384(AX), X0  
0x014a 00330 (pointer_vs_value/pointer_receiver/func.g0:190) ADDSD 400(AX), X0  
0x0152 00338 (pointer_vs_value/pointer_receiver/func.g0:190) PCDATA $9, $8  
0x0152 00338 (pointer_vs_value/pointer_receiver/func.g0:190) ADDSD 408(AX), X0  
0x015a 00346 (pointer_vs_value/pointer_receiver/func.g0:184) XORPS X1, X1  
0x015d 00349 (pointer_vs_value/pointer_receiver/func.g0:184) CVTSQ2SD CX, X1  
0x0162 00354 (pointer_vs_value/pointer_receiver/func.g0:187) DIVSD X0, X1  
0x0166 00358 (pointer_vs_value/pointer_receiver/func.g0:184) MOVSD X1, ***.-r1+16(SP)  
0x016c 00364 (pointer_vs_value/pointer_receiver/func.g0:184) PCDATA $1, $2  
0x016c 00364 (pointer_vs_value/pointer_receiver/func.g0:184) XORPS X0, X0  
0x0167 00367 (pointer_vs_value/pointer_receiver/func.g0:184) MOVUPS X0, ***.-r2+24(SP)  
0x0174 00372 (pointer_vs_value/pointer_receiver/func.g0:184) RET  
0x0000 48 8b 24 08 48 8b 08 48 03 48 08 48 03 48 10 H.DS.H.H.H.H.H.H  
0x0010 48 03 48 18 08 03 48 20 48 03 48 28 08 03 48 30 H.H.H.H.H.H.H.H  
0x0020 48 03 48 38 08 03 48 40 48 03 48 48 08 03 48 50 H.H.H.H.H.H.H.H  
0x0030 48 03 48 58 08 03 48 60 48 03 48 68 08 03 48 70 H.H.H.H.H.H.H.H  
0x0040 48 03 48 78 08 03 48 88 08 08 08 48 03 48 88 88 08 H.H.H.H.H.H.H.H  
0x0050 00 48 03 48 98 08 08 08 08 08 08 08 08 08 08 08 H.H.H.H.H.H.H.H  
0x0060 48 03 88 08 08 08 48 03 88 08 08 08 08 08 08 08 H.H.H.H.H.H.H.H  
0x0070 88 08 08 08 48 03 88 08 08 08 08 48 03 88 08 08 H.H.H.H.H.H.H.H  
0x0080 00 08 08 08 08 48 03 88 08 08 08 08 08 08 08 08 H.H.H.H.H.H.H.H  
0x0090 00 08 f2 0f 58 80 d8 08 08 08 f2 0f 58 80 c0 08 H.X.X.X.X.X.X.X  
0x00a0 00 08 f2 0f 58 80 e8 08 08 08 f2 0f 58 80 f0 08 H.X.X.X.X.X.X.X  
0x00b0 00 08 f2 0f 58 80 f8 08 08 08 f2 0f 58 80 00 08 H.X.X.X.X.X.X.X  
0x00c0 00 08 f2 0f 58 80 08 01 08 08 f2 0f 58 80 10 01 H.X.X.X.X.X.X.X  
0x00d0 00 08 f2 0f 58 80 18 01 08 08 f2 0f 58 80 20 01 H.X.X.X.X.X.X.X  
0x00e0 48 03 88 08 08 08 48 03 88 08 08 08 08 08 08 08 H.X.X.X.X.X.X.X  
0x00f0 00 08 f2 0f 58 80 28 01 08 08 f2 0f 58 80 30 01 H.X.X.X.X.X.X.X  
0x0100 00 08 f2 0f 58 80 38 01 08 08 f2 0f 58 80 40 01 H.X.X.X.X.X.X.X  
0x0110 00 08 f2 0f 58 80 48 01 08 08 f2 0f 58 80 50 01 H.X.X.X.X.X.X.X  
0x0120 00 08 f2 0f 58 80 58 01 08 08 f2 0f 58 80 60 01 H.X.X.X.X.X.X.X  
0x0130 00 08 f2 0f 58 80 68 01 08 08 f2 0f 58 80 70 01 H.X.X.X.X.X.X.X  
0x0140 00 08 f2 0f 58 80 78 01 08 08 f2 0f 58 80 80 01 H.X.X.X.X.X.X.X  
0x0150 00 08 f2 0f 58 80 88 01 08 08 f2 0f 58 80 90 01 H.X.X.X.X.X.X.X  
0x0160 2a c8 f2 0f 5c 80 f2 0f 11 4c 2a 10 0f 57 c0 0f X.X.X.X.X.X.X.X  
0x0170 11 4c 2a 10 c3 D.S...
```

value

```
***.calcGigaObject STEX nospit size=430 args=0x108 locals=0x0  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:171) TEXT ***.calcGigaObject(S0), NOSPLIT  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:171) FUNCDATA $0, gclicals:42dd1bc3808cb7c5  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:171) FUNCDATA $1, gclicals:69c1753bd5f8150  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:171) FUNCDATA $2, gclicals:33cdeccecb803  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:174) PCDATA $0, $0  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:174) PCDATA $1, $0  
0x0000 00000 (pointer_vs_value/pointer_receiver/func.g0:174) MOVQ ***.o+8(SP), AX  
0x0005 00005 (pointer_vs_value/pointer_receiver/func.g0:174) ADDQ ***.o+16(SP), AX  
0x000a 00010 (pointer_vs_value/pointer_receiver/func.g0:174) ADDQ ***.o+24(SP), AX  
0x000f 00015 (pointer_vs_value/pointer_receiver/func.g0:174) ADDQ ***.o+32(SP), AX  
...  
0x0169 00361 (pointer_vs_value/pointer_receiver/func.g0:179) ADDSD ***.o+400(SP), X0  
0x0172 00370 (pointer_vs_value/pointer_receiver/func.g0:179) ADDSD ***.o+392(SP), X0  
0x017b 00379 (pointer_vs_value/pointer_receiver/func.g0:179) ADDSD ***.o+408(SP), X0  
...  
0x0184 00388 (pointer_vs_value/pointer_receiver/func.g0:179) ADDSD ***.o+416(SP), X0  
0x018d 00397 (pointer_vs_value/pointer_receiver/func.g0:173) XORPS X1, X1  
0x0190 00400 (pointer_vs_value/pointer_receiver/func.g0:173) CVTSQ2SD AX, X1  
0x0195 00405 (pointer_vs_value/pointer_receiver/func.g0:176) DIVSD X0, X1  
0x0199 00409 (pointer_vs_value/pointer_receiver/func.g0:173) MOVSD X1, ***.-r1+424(SP)  
0x01a2 00418 (pointer_vs_value/pointer_receiver/func.g0:173) PCDATA $1, $1  
0x01a2 00418 (pointer_vs_value/pointer_receiver/func.g0:173) XORPS X0, X0  
0x01a5 00421 (pointer_vs_value/pointer_receiver/func.g0:173) MOVUPS X0, ***.-r2+432(SP)  
0x01ad 00429 (pointer_vs_value/pointer_receiver/func.g0:173) RET  
0x0000 48 8b 24 08 48 03 44 2a 18 48 03 44 2a 18 48 H.DS.H.DS.H.DS.H  
0x0010 03 44 2a 08 03 44 2a 28 08 03 44 2a 30 48 03 DS.H.DS.H.DS.H  
0x0020 44 2a 38 08 03 44 2a 40 48 03 44 2a 48 03 44 DSH.DSH.DS.H.DS  
0x0030 24 58 03 44 2a 58 08 03 44 2a 60 08 03 44 2a SPH.DSH.DS.H.DS  
0x0040 68 08 03 44 2a 78 08 03 44 2a 78 08 03 84 2a 80 HH.DSH.DSH.DS  
0x0050 00 08 48 03 84 2a 88 08 08 08 48 03 84 2a 90 H.H.S.H.S.H.S  
0x0060 00 08 48 03 84 2a 98 08 08 08 48 03 84 2a a0 H.H.S.H.S.H.S  
0x0070 00 08 48 03 84 2a ab 08 08 48 03 84 2a b0 H.H.S.H.S.H.S  
0x0080 00 08 48 03 84 2a c0 08 08 48 03 84 2a b8 H.H.S.H.S.H.S  
0x0090 00 08 48 03 84 2a c8 08 08 48 03 84 2a d0 H.H.S.H.S.H.S  
0x00a0 00 08 f2 0f 10 84 2a e8 08 08 08 f2 0f 58 84 X.X.X.X.X.X.X  
0x00b0 2a d8 08 08 f2 0f 58 84 2a e8 08 08 08 f2 0f S.X.X.X.X.X.X  
0x00c0 58 84 2a f0 08 08 f2 0f 58 84 2a f0 08 08 08 X.S.X.X.X.X.X  
0x00d0 f2 0f 58 84 2a 01 08 08 f2 0f 58 84 2a 01 X.S.H.S.H.S.H.S  
0x00e0 00 08 f2 0f 58 84 2a 01 08 08 f2 0f 58 84 2a X.S.H.S.H.S.H.S  
0x00f0 18 01 08 08 f2 0f 58 84 2a 01 08 08 f2 0f 58 X.S.X.S.X.S.X  
0x0100 84 2a 01 08 08 f2 0f 58 84 2a 01 08 08 f2 S.X.S.X.S.X.S  
0x0110 0f 58 84 2a 01 08 08 f2 0f 58 84 2a 40 01 X.S.H.S.H.S.H.S  
0x0120 0f 58 84 2a 40 01 08 08 f2 0f 58 84 2a 50 X.S.H.S.H.S.H.S  
0x0130 01 08 08 f2 0f 58 84 2a 58 01 08 08 f2 0f 58 X.X.X.X.X.X  
0x0140 2a 60 01 08 08 f2 0f 58 84 2a 60 01 08 08 f2 S.X.S.H.S.H.S  
0x0150 58 84 2a 70 01 08 08 f2 0f 58 84 2a 78 01 08 X.S.H.S.H.S.H.S  
0x0160 f2 0f 58 84 2a 80 01 08 08 f2 0f 58 84 2a 90 X.S.H.S.H.S.H.S  
0x0170 08 08 f2 0f 58 84 2a 88 01 08 08 f2 0f 58 84 X.S.H.S.H.S.H.S  
0x0180 98 01 08 08 f2 0f 58 84 2a a0 01 08 08 f2 0f X.S.H.S.H.S.H.S  
0x0190 f2 48 0f 2a c8 f2 0f 5c 80 f2 0f 11 8c 2a a8 01 H.X.X.X.X.X.S  
0x01a0 00 08 0f 57 c0 0f 11 84 2a b0 01 08 08 c3 X.X.X.X.X.X.S
```

BEAT

# Extended Results

## What does it mean?

- Difference in performance becomes much more balanced.
- Value implementation beats interfaces.

## Extra Tip!

- If you won't access many of your object's properties consider seriously using pointers or interfaces!

```
goos: darwin
goarch: amd64
pkg: go-bench/pointer_vs_value/pointer_receiver
BenchmarkTera_Calc-16      77776372      14.8 ns/op
BenchmarkTeraP_Calc-16    678546289      1.74 ns/op
BenchmarkTeraIfc_Calc-16  166455222      7.14 ns/op
PASS
```



```
goos: darwin
goarch: amd64
pkg: go-bench/pointer_vs_value/pointer_receiver
BenchmarkGiga_Calc-16      46147950      29.9 ns/op
BenchmarkGigaP_Calc-16    57532897      19.2 ns/op
BenchmarkGigaIfc_Calc-16  10571948      98.9 ns/op
PASS
```

**BEAT**

# THANK YOU!

**Vangelis Katikaridis | Sr. Backend Engineer**

[github.com/drakos74](https://github.com/drakos74)

Twitter: @vangkos

**BEAT**