WARSAW, APR 25 2019

go-perftuner

Oleg Kovalov

Allegro

Twitter:
oleg_kovalov

Github: cristaloleg

Me

- Gopher for ~3 years
- Open-source contributor
- Engineer at allegro.pl core team

Twitter: @oleg_kovalov
Github: @cristaloleg

What is a performance optimization?

What is a performance optimization?

- Code changes

What is a performance optimization?

- Code changes
- But not only code

What is a performance optimization?

- Code changes
- But not only code
- Better resources utilization

What is a performance optimization?

- Code changes
- But not only code
- Better resources utilization
- Space for the new features

What is a performance optimization?

- Code changes
- But not only code
- Better resources utilization
- Space for the new features
- Much happier users

When performance optimization is needed?

When performance optimization is needed?

- Program runs slower than it should

When performance optimization is needed?

- Program runs slower than it should
- CPU usage is too high

When performance optimization is needed?

- Program runs slower than it should
- CPU usage is too high
- Uses a lot of memory

When performance optimization is needed?

- Program runs slower than it should
- CPU usage is too high
- Uses a lot of memory
- Overloads DB

When performance optimization is needed?

- Program runs slower than it should
- CPU usage is too high
- Uses a lot of memory
- Overloads DB
- Overloads a network

When performance optimization is needed?

- Program runs slower than it should
- CPU usage is too high
- Uses a lot of memory
- Overloads DB
- Overloads a network
- Requires a lot of storage

When performance optimization is needed?

- Program runs slower than it should
- CPU usage is too high
- Uses a lot of memory
- Overloads DB
- Overloads a network
- Requires a lot of storage
- Expensive to run (like in the cloud)

When performance optimization isn’t needed?

When performance optimization isn’t needed?

- Runs fast enough

When performance optimization isn’t needed?

- Runs fast enough
- Have enough of memory & storage

When performance optimization isn’t needed?

- Runs fast enough
- Have enough of memory & storage
- No performance requirements

When performance optimization isn’t needed?

- Runs fast enough
- Have enough of memory & storage
- No performance requirements
- Can run faster by 5% in a “cold” func

When performance optimization isn't needed?

- Runs fast enough
- Have enough of memory & storage
- No performance requirements
- Can run faster by 5% in a "cold" func
- 100 rps but new code will handle 1.2M rps

When performance optimization isn’t needed?

- Runs fast enough
- Have enough of memory & storage
- No performance requirements
- Can run faster by 5% in a “cold” func
- 100 rps but new code will handle 1.2M rps
- JUST GIVE ME SOMETHING TO MAKE MUCH FASTER I DON’T CARE AAAAAAAA!1111!!111 1111/1!!11!!1112:RAGE:111!@!212@!!@12

Types of performance tuning

Types of performance tuning

- CPU

Types of performance tuning

- CPU
- Memory

Types of performance tuning

- CPU
- Memory
- GC

Types of performance tuning

- CPU
- Memory
- GC
- IO

Types of performance tuning

- CPU
- Memory
- GC
- IO
- Network

Types of performance tuning

- CPU
- Memory
- GC
- IO
- Network
- Parallelism

Types of performance tuning

- CPU
- Memory
- GC
- IO
- Network
- Parallelism
- Algorithms <3

Types of performance tuning

- CPU
- Memory
- GC
- IO
- Network
- Parallelism
- Algorithms <3
- ...

What Go compiler can do?

What Go compiler can do?

- Dead code elimination

```
if 0 > 1 {
    foo()
}
```

What Go compiler can do?

- Dead code elimination
- Constant folding

```
a := 10 + 15
```

vs

```
a := 25
```

What Go compiler can do?

- Dead code elimination
- Constant folding
- Code inlining
- Bounds-checking elimination
- Escape analysis
- ...

So, welcome go-perftuner

So, welcome go-perftuner

- Pretty compiler output

So, welcome go-perftuner

- Pretty compiler output
- Shows only useful info

So, welcome go-perftuner

- Pretty compiler output
- Shows only useful info
- Simple commands

So, welcome go-perftuner

- Pretty compiler output
- Shows only useful info
- Simple commands
- JSON output

So, welcome go-perftuner

- Pretty compiler output
- Shows only useful info
- Simple commands
- JSON output
- Your feature?

go get -u github.com/cristaloleg/go-perftuner

Code inlining

Code inlining

- No func call overhead

Code inlining

- No func call overhead
- Critical for “hot” funcs

Code inlining

- No func call overhead
- Critical for “hot” funcs
- Less jumps => better performance

Code inlining

- No func call overhead
- Critical for “hot” funcs
- Less jumps => better performance
- Also impacts binary size

almostInlined example (1)

```
func ReadRuneSimple(s string) (ch rune, size int) {
	if len(s) == 0 {
		return
	}

	ch, size = utf8.DecodeRuneInString(s)
	return
}
```

almostInlined example (1)

```
func ReadRuneSimple(s string) (ch rune, size int) {
	if len(s) == 0 {
		return
	}

	ch, size = utf8.DecodeRuneInString(s)
	return
}

$ go-perftuner inl -threshold=100 old.go
```

almostInlined example (1)

```
func ReadRuneSimple(s string) (ch rune, size int) {
	if len(s) == 0 {
		return
	}

	ch, size = utf8.DecodeRuneInString(s)
	return
}

$ go-perftuner inl -threshold=100 old.go

$
```

almostInlined example (2)

```
func ReadRuneLogged(s string) (ch rune, size int) {
     if len(s) == 0 {
          return
     }
     log.Printf("we're working")
     ch, size = utf8.DecodeRuneInString(s)
     return
}

$ go-perftuner inl -threshold=100 new.go
```

almostInlined example (2)

```
func ReadRuneLogged(s string) (ch rune, size int) {
	if len(s) == 0 {
		return
	}
	log.Printf("we're working")
	ch, size = utf8.DecodeRuneInString(s)
	return
}

$ go-perftuner inl -threshold=100 new.go

inl: ./new.go:34:6: ReadRuneLogged: budget exceeded by 50
```

Bounds-checking elimination

Bounds-checking elimination

- Give the compiler hints that index-based memory access is safe

Bounds-checking elimination

- Give the compiler hints that index-based memory access is safe
- No extra checks during runtime

Bounds-checking elimination

- Give the compiler hints that index-based memory access is safe
- No extra checks during runtime
- Less checks => better performance

boundChecks example (1)

```
func PutUint32(b []byte, v uint32) {

    b[0] = byte(v)
    b[1] = byte(v >> 8)
    b[2] = byte(v >> 16)
    b[3] = byte(v >> 24)
}
```

boundChecks example (2)

```
func PutUint32(b []byte, v uint32) {

    b[0] = byte(v)
    b[1] = byte(v >> 8)
    b[2] = byte(v >> 16)
    b[3] = byte(v >> 24)
}

$ go-perftuner bce old.go

bce: ./old.go:4:7: slice/array has bound checks
bce: ./old.go:5:7: slice/array has bound checks
bce: ./old.go:6:7: slice/array has bound checks
bce: ./old.go:7:7: slice/array has bound checks

$
```

boundChecks example (3)

```
func PutUint32(b []byte, v uint32) {
    _ = b[3] // early check to guarantee safety
    b[0] = byte(v)
    b[1] = byte(v >> 8)
    b[2] = byte(v >> 16)
    b[3] = byte(v >> 24)
}
```

boundChecks example (4)

```
func PutUint32(b []byte, v uint32) {
    _ = b[3] // early check to guarantee safety
    b[0] = byte(v)
    b[1] = byte(v >> 8)
    b[2] = byte(v >> 16)
    b[3] = byte(v >> 24)
}

$ go-perftuner bce new.go

bce: ./new.go:4:7: slice/array has bound checks

$
```

boundChecks example (4)

```
func PutUint32(b []byte, v uint32) {
    if len(b) < 4 { return }
    b[0] = byte(v)
    b[1] = byte(v >> 8)
    b[2] = byte(v >> 16)
    b[3] = byte(v >> 24)
}

$ go-perftuner bce new.go

$
```

Escape analysis

Escape analysis

- Process that determines the placement of values

Escape analysis

- Process that determines the placement of values
- Determines if a value will be on the stack or “escapes” to the heap

Escape analysis

- Process that determines the placement of values
- Determines if a value will be on the stack or “escapes” to the heap
- Less escapes => less work for GC

escapedVariable example (1)

```
func getRands(size int) []int {
	nums := make([]int, size)
	for i := range nums {
		nums[i] = rand.Int()
	}
	return nums
}
```

escapedVariable example (1)

```
func getRands(size int) []int {
	nums := make([]int, size)
	for i := range nums {
		nums[i] = rand.Int()
	}
	return nums
}

$ go-perftuner esc old.go

esc: ./old.go:16:14: make([]int, size)

$
```

escapedVariable example (2)

```go
func sumRand() (total int) {
	nums := make([]int, 8191)
	for i := range nums {
		nums[i] = rand.Int()
	}
	for _, x := range nums {
		total += x
	}
	return total
}
```

escapedVariable example (3)

```
func sumRand() (total int) {
	nums := make([]int, 8191)
	for i := range nums {
		nums[i] = rand.Int()
	}
	for _, x := range nums {
		total += x
	}
	return total
}

$ go-perftuner esc 1.go

$
```

escapedVariable example (4)

```
func sumRand() (total int) {
	nums := make([]int, 8191 + 1)
	for i := range nums {
		nums[i] = rand.Int()
	}
	for _, x := range nums {
		total += x
	}
	return total
}

$ go-perftuner esc 1.go
```

escapedVariable example (4)

```
func sumRand() (total int) {
	nums := make([]int, 8191 + 1)
	for i := range nums {
		nums[i] = rand.Int()
	}
	for _, x := range nums {
		total += x
	}
	return total
}

$ go-perftuner esc 1.go

esc: ./old.go:16:14: make([]int, 8191 + 1)

$
```

A very handy tool

A very handy tool

- Called benchstat

A very handy tool

- Called benchstat
- Fast way to compare benchmarks

A very handy tool

- Called benchstat
- Fast way to compare benchmarks
- Helps a lot

go get golang.org/x/perf/cmd/benchstat

benchstat

```
$ go get golang.org/x/perf/cmd/benchstat

$ go test -bench=. -count 10 > old.txt
```

benchstat

```
$ go get golang.org/x/perf/cmd/benchstat

$ go test -bench=. -count 10 > old.txt

$ # < do some coding and magic with go-perftuner >

$ go test -bench=. -count 10 > new.txt
```

benchstat

```
$ go get golang.org/x/perf/cmd/benchstat

$ go test -bench=. -count 10 > old.txt

$ # < do some coding and magic with go-perftuner >

$ go test -bench=. -count 10 > new.txt

$ benchstat old.txt new.txt

name   old time/op   new time/op   delta
Foo    13.6ms ± 1%   11.8ms ± 1%   -13.31% (p=0.016 n=4+5)
Bar    32.1ms ± 1%   31.8ms ± 1%      ~    (p=0.286 n=4+5)
```

Performance tuning summary

Performance tuning summary

- have a correct code with tests(!)

Performance tuning summary

- have a correct code with tests(!)
- do initial bench before any changes
- should it be optimized? (y/n)

Performance tuning summary

- have a correct code with tests(!)
- do initial bench before any changes
- should it be optimized? (y/n)
- go get -u github.com/cristaloleg/go-perftuner
- <magic spells>

Performance tuning summary

- have a correct code with tests(!)
- do initial bench before any changes
- should it be optimized? (y/n)
- go get -u github.com/cristaloleg/go-perftuner
- <magic spells>
- commit & push

Performance tuning summary

- have a correct code with tests(!)
- do initial bench before any changes
- should it be optimized? (y/n)
- go get -u github.com/cristaloleg/go-perftuner
- <magic spells>
- commit & push
- drink a glass of water 

That’s all folks

Thank you
Questions?

Twitter: @oleg_kovalov
Github: @cristaloleg