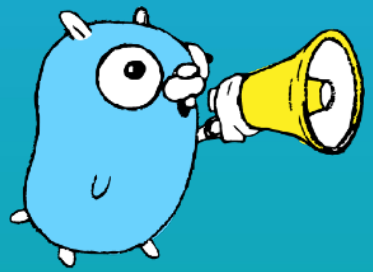
Go 1.13 Release Party, Aug 22 2019

Go 1.13 Release Notes

Hyejong Hong

GDG Golang KR

Speaker



Today's (glorious) blather.

SECTION ONE

Install Go 1.13

SECTION TWO

Changes to the language

Changes to the language

- Go 2 Number Literal Changes
  - Binary integer literals
  - Octal integer literals
  - Digit separator
  - Hexadecimal floating point
  - *Imaginary literals
- Allow signed shift counts

“

Go 2 Number Literal Changes

Go는 C언어 스타일의 숫자 리터럴(literal)을 채택하였었고
이러한 “C-numbered languages” 그룹에는
C, C++, C#, Java, JavaScript, Perl, PHP, Python, Ruby, Rust, Swift가 포함됩니다.

Go가 발표된지 10년이 흘렀고 “C-numbered languages”의 대부분의 언어들은 앞에서 제안한 4가지 제안 중 하나 이상을 추가하여 확장했습니다.

이제 Go도 같은 방식으로 확장하여 개발자들이 이러한 언어 사이를 손쉽게 이동할 수 있도록 불필요한 장벽을 제거하려고 합니다.

Binary integer literals

https://github.com/golang/go/issues/19308

```
const (
    SOME_MASK    = 0b00001111
    SOME_FLAG_A = 0b00000001
    SOME_FLAG_B = 0B00000010
    SOME_FLAG_C = 0B00000100
    SOME_FLAG_D = 0B00001000
)
```

- C++ 14 (2014)
- C# 7.0 (2017)
- Java 7 (2011)
- JavaScript ES6 (2015)
- Perl 5.005_55 (1998)
- PHP 5.4.0 (2012)
- Python 2.6 (2008)
- Ruby 1.4.0 (1999)
- Rust 0.1 (2012)
- Swift 1.0 (2014)

Octal integer literals

https://github.com/golang/go/issues/12711

```
const (
    SOME_MASK    = 000001111
    SOME_FLAG_A = 0o0000001
    SOME_FLAG_B = 0O12
)
```

- C
- C++
- Java
- Perl
- PHP
- Ruby
- Caml Light 0.5 (1992)
- JavaScript ES3 (1999)
- Python (1991)
- Rust (2012)
- Swift (2014)

Hexadecimal floating point

https://github.com/golang/go/issues/29008

```
0x1.FFFFFFFFFFFFFp1023  // double.max
0x1p-52                 // double.epsilon
1.175494351e-38F        // float.min
```

- Fortran (1956)
- Algol 68 (1968)
- PL/I (1964)
- C99 (1999)
- C++17 (2017)
- Java 5 (2004)
- Perl 5.22 (2015)
- Swift (2014)

Digit separator

https://github.com/golang/go/issues/28493

```
0__        // 0
0_77       // 77 decimal, or perhaps not permitted (TBD)
01_10      // 0110 (octal)
1_2__3___  // 123
0x1_23_0   // 0x1230
0__.0__e-0_ // 0.0e-0

_42        // an identifier, not an integer literal
42_        // invalid: _ must separate successive digits
4__2       // invalid: only one _ at a time
0_xBadFace  // invalid: _ must separate successive digits
```

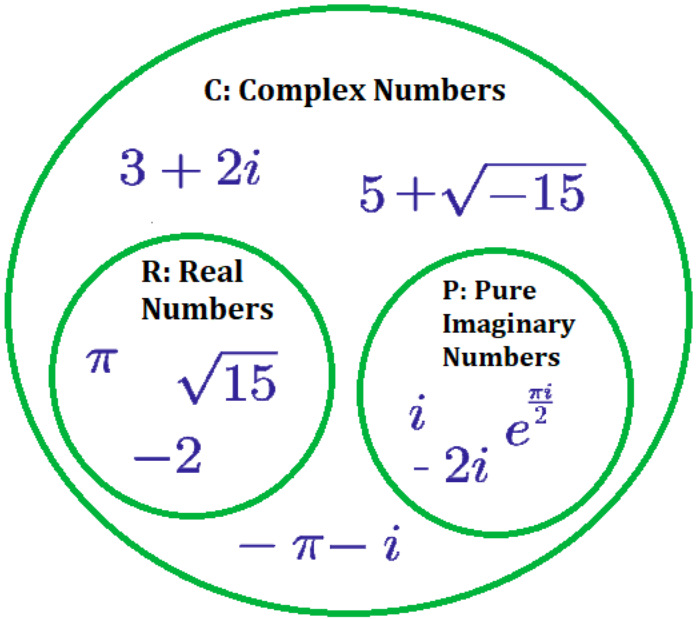
- Ada 83
- C# 7.0 (2017)
- Java 7 (2011)
- Perl 2.0 (1988)
- Python 3.6 (2016)
- Ruby 1.0 (1998)
- Rust 0.1 (2012)
- Swift 1.0 (2014)

Imaginary literals

- 이제 허수(imaginary) 리터럴이 모든 정수(int, decimal, hexadecimal), 부동 소수점과 함께 사용할 수 있습니다.

```
imaginary_lit = (decimal_digits | int_lit | float_lit) "i" .
```

```
0i
0123i         // == 123i for backward-compatibility
0o123i        // == 0o123 * 1i == 83i
0xabci        // == 0xabc * 1i == 2748i
0.i
2.71828i
1.e+0i
6.67428e-11i
1E6i
.25i
.12345E+5i
0x1p-2i       // == 0x1p-2 * 1i == 0.25i
```



“

Permit Signed Integers as Shift Counts for Go 2

처음부터 Go의, shift counts는
부호없는 정수 타입에만 적용되었습니다.

이 규칙의 기본 개념은
(a) spec에서 음수 값에 대해 발생한 일을 설명 할 필요가 없었으며
(b) 구현시 runtime에서 발생할 수 있는 음수 값을 처리 할 필요가 없었습니다.

Allow signed shift counts

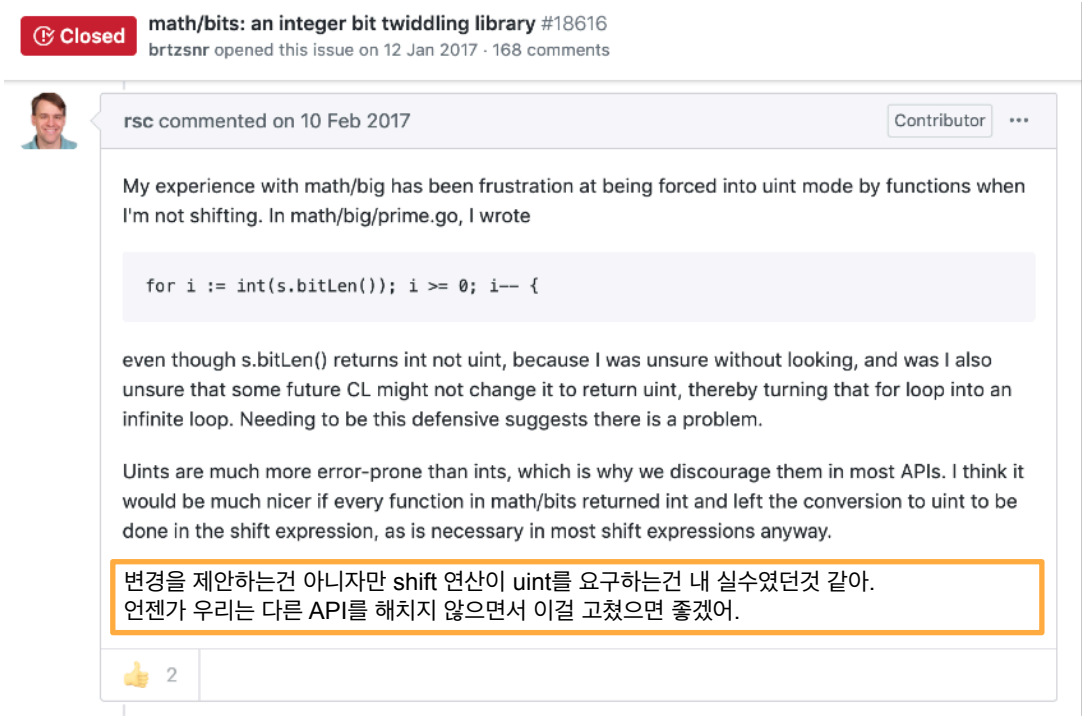
Arithmetic operators

Arithmetic operators apply to numeric values and yield a result of the same type as the first operand. The four standard arithmetic operators (+, −, ∗, /) apply to integer, floating-point, and complex types; + also applies to strings. The bitwise logical and shift operators apply to integers only.

```
+    sum                    integers, floats, complex values, strings
-    difference             integers, floats, complex values
*    product                integers, floats, complex values
/    quotient               integers, floats, complex values
%    remainder              integers

&    bitwise AND            integers
|    bitwise OR             integers
^    bitwise XOR            integers
&^   bit clear (AND NOT)    integers

<<   left shift             integer << unsigned integer
>>   right shift            integer >> unsigned integer
```

https://golang.org/ref/spec#Arithmetic_operators

Allow signed shift counts



Closed **math/bits: an integer bit twiddling library** #18616

brtzsnr opened this issue on 12 Jan 2017 · 168 comments

rsc commented on 10 Feb 2017 — Contributor

My experience with math/big has been frustration at being forced into uint mode by functions when I'm not shifting. In math/big/prime.go, I wrote

```
for i := int(s.bitLen()); i >= 0; i-- {
```

even though s.bitLen() returns int not uint, because I was unsure without looking, and was I also unsure that some future CL might not change it to return uint, thereby turning that for loop into an infinite loop. Needing to be this defensive suggests there is a problem.

Uints are much more error-prone than ints, which is why we discourage them in most APIs. I think it would be much nicer if every function in math/bits returned int and left the conversion to uint to be done in the shift expression, as is necessary in most shift expressions anyway.

변경을 제안하는건 아니자만 shift 연산이 uint를 요구하는건 내 실수였던것 같아.
언젠가 우리는 다른 API를 해치지 않으면서 이걸 고쳤으면 좋겠어.

2

https://github.com/golang/go/issues/18616#issuecomment-278852766

쓸때없는 uint casting 사례

archive/tar/strconv.go:88

```
func fitsInBase256(n int, x int64) bool {
        var binBits = uint(n-1) * 8    // <<<< uint cast
        return n >= 9 || (x >= -1<<binBits && x < 1<<binBits)
}
```

cmd/compile/internal/gc/esc.go:1460

```
shift := uint(bitsPerOutputInTag*(vargen-1) + EscReturnBits)    // <<<< uint cast
old := (e >> shift) & bitsMaskForTag
```

src/fmt/scan.go:604

```
n := uint(bitSize)    // uint cast
x := (r << (64 - n)) >> (64 - n)
```

shift 연산의 Overflow, Underflow

```
func main(){
    var ui uint32 = 0b01
    var i int32 = 0b01

    fmt.Println(ui >> 1, i >> 1)     // 0, 0
    fmt.Println(ui << 31, i << 31)   // 2147483648, -2147483648
    fmt.Println(ui << 32, i << 32)   // 0, 0
}
```

오버플로우(Overflow), 언더플로우(Underflow)가 없음.

SECTION THREE

Tools

Tools

- Modules
- Go Command
  - go env
  - go version
  - go build -trimpath, tags
- Compiler toolchain
  - add new escape analysis implementation
- remove godoc

Modules

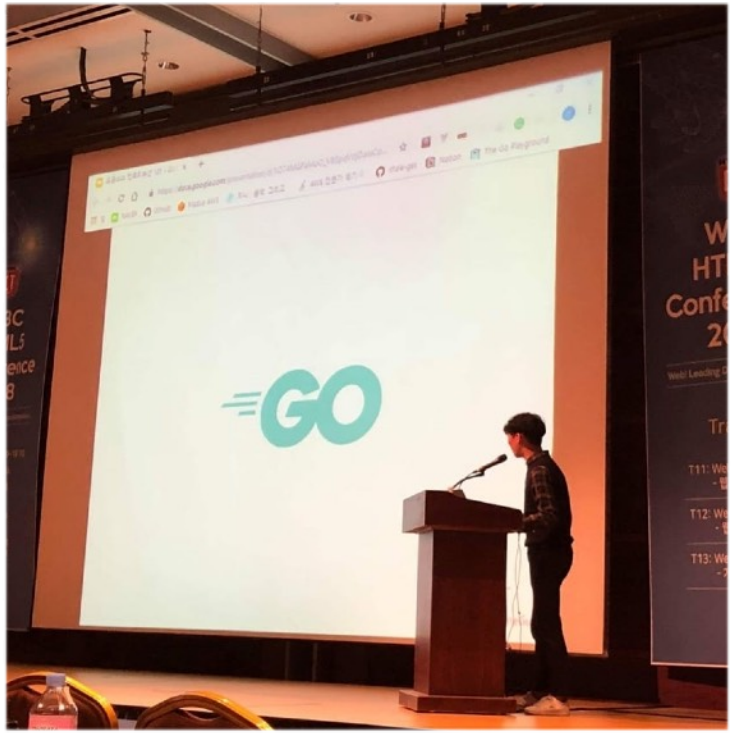
These are the posts in the February

- "Go += Package Versioning"
- "A Tour of Versioned Go (v
- "Semantic Import Versionin
- "Minimal Version Selection"
- "Reproducible, Verifiable, V
- "Defining Go Modules" [PD
- "Versioned Go Commands"

The official Go proposal is at https 1, 2018.

These are shorter followup posts:

- "The vgo proposal is accepte
- "What is Software Engineeri
- "Why Add Versions To Go?"



go env

- go 환경 변수를 좀 더 쉽게 사용하기 위한 기능 추가
- go env -m는 os.UserConfigDir() 경로에 파일로 환경변수값을 저장
- go env -u는 관련 환경 변수값 삭제

```
hyejong ~/Library/Application Support/go $ go1.13beta1 env -w GOPROXY=direct
hyejong ~/Library/Application Support/go $ cat env
GOPROXY=direct
hyejong ~/Library/Application Support/go $ go1.13beta1 env -u GOPROXY
hyejong ~/Library/Application Support/go $ cat env
hyejong ~/Library/Application Support/go $
```

go version -m

- go version -m {file}은

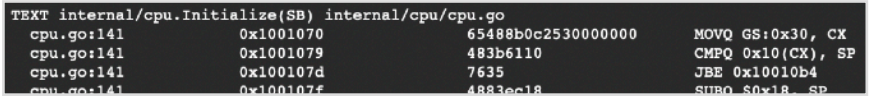
  해당 바이너리의 go, module, dependency 버전 출력

```
hyejong ~/Library/Mobile Documents/com~apple~CloudDocs/Source/go1.13 $ go1.13beta1 version -m binary
binary: go1.13beta1
        path    command-line-arguments
        mod     go1.13  (devel)
        dep     github.com/google/uuid  v1.1.1  h1:Gkbcsh/GbpXz7lPftLA3P6TYMwjCLYm83jiFQZF/3gY=
        dep     github.com/labstack/echo/v4     v4.1.10 h1:/yhIpO50CBInUbE/nHJtGIyhBv0dJe2cDAYxc3V3uMo=
        dep     github.com/labstack/gommon      v0.3.0  h1:JEeO0bvc78PKdyHxloTKiF8BD5iGrH8T6MSeGvSgob0=
        dep     github.com/mattn/go-colorable   v0.1.2  h1:/bC9yWikZXAL9uJdulbSfyVNIR3n3trXl+v8+1sx8mU=
        dep     github.com/mattn/go-isatty      v0.0.9  h1:d5US/mDsogSGW37IV293h//ZFaeajb69h+EHFsv2xGg=
        dep     github.com/valyala/bytebufferpool       v1.0.0  h1:GqA5TC/0021Y/b9FG4Oi9Mr3q7XYx6KllzawFIhcdPw=
        dep     github.com/valyala/fasttemplate v1.0.1  h1:tY9CJiPnMXf1ERmG2EyK7gNUd+c6RKGD0IfU8WdUSz8=
        dep     golang.org/x/crypto     v0.0.0-20190701094942-4def268fd1a4      h1:HuIa8hRrWRSrqYzx1qI49NNxhdi2PrY7gxVSq1JjLDc=
        dep     golang.org/x/net        v0.0.0-20190404232315-eb5bcb51f2a3      h1:0GoQqolDA55aaLxZyTzK/Y2ePZzZTUrRacwib7cNsYQ=
        dep     golang.org/x/text       v0.3.0  h1:g61tztE5qeGQ89tm6NTjjM9VPIm088od1l6aSorWRWg=
```

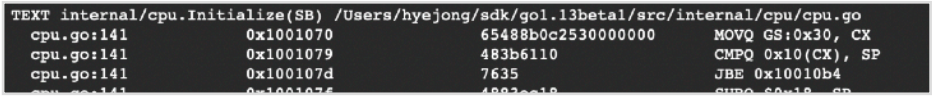
go build -trimpath

- go build -trimpath는 실행 파일에 로컬 디렉토리의 path를 기록하지 않도록 파일 경로를 제거합니다.

```
[hyejong ~/Source/go1.13 $ go tool objdump binary
```



go build -trimpath main.go



go build main.go

go build -tags

- 이제 tags에 comma-separated 가능

  ex) go build -tags dev,prd {file}

src/os/pipe_bsd.go

```
5   // +build aix darwin dragonfly js,wasm nacl solaris
6
7   package os
8
9   import "syscall"
10
11  // Pipe returns a connected pair of Files; reads from r return bytes written to w.
12  // It returns the files and an error, if any.
13  func Pipe() (r *File, w *File, err error) {
14      var p [2]int
15
16      // See ../syscall/exec.go for description of lock.
17      syscall.ForkLock.RLock()
```

src/runtime/mmap.go

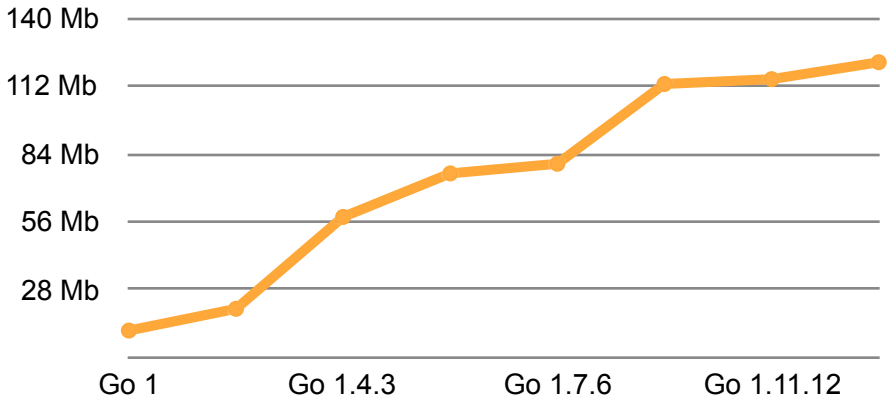
```
5   // +build !plan9
6   // +build !solaris
7   // +build !windows
8   // +build !nacl
9   // +build !linux !amd64
10  // +build !linux !arm64
11  // +build !js
12  // +build !darwin
13  // +build !aix
14
15  package runtime
16
17  import "unsafe"
18
```

cmd/compile: add new escape analysis implementation

- escape analysis가 새로운 구현체로 바뀌었습니다.
- 이 구현은 더 적은 메모리를 사용하고 일부 컴파일 시간을 단축시키며 전반적으로 escape analysis 분석 결과를 크게 개선시켰습니다.

godoc

- godoc 웹 서버는 더 이상 기본 바이너리 배포판에 포함되지 않습니다.
  (배포판 사이즈 줄이기 위해 -16MB)
- godoc 웹 서버를 실행하려면 수동으로 설치해야 합니다.




```
go get golang.org/x/tools/cmd/godoc
godoc
```

수동 설치 방법

SECTION FOUR

Runtime

Runtime

- Out of range panic messages now include the index.
- This release improves performance of most uses of defer by 30%.
- The runtime is now more aggressive at returning memory to the operating system.
- Randomize package initialization order in race mode.

Panic message

- 이제 out of range로 발생한 panic 메시지에

  범위를 벗어난 인덱스와 슬라이스의 길이(또는 용량)이 추가적으로 출력됩니다.

```
hyejong ~/Library/Mobile Documents/com~apple~CloudDocs/Source/go1.13 $ go1.13beta1 run panic.go
panic: runtime error: index out of range [4] with length 3

goroutine 1 [running]:
main.main()
        /Users/hyejong/Library/Mobile Documents/com~apple~CloudDocs/Source/go1.13/panic.go:8 +0x1d
exit status 2
```

Allocate defer records on the stack

- 함수에서 defer가 최대 한번만 실행된다면
  defer를 heap대신 stack에 할당 할 수 있음.
- 이 최적화는 cmd/go 패키지의 370의 defer 중 363개에 적용될 정도로 효과적임
- 속도는 30% 정도 향상됨

```
803 »       case ODEFER:
804 »       »       if e.loopdepth == 1 { // top level
805 »       »       »       n.Esc = EscNever // force stack allocation of defer record (see ssa.go)
806 »       »       »       break
807 »       »       }
```

src/cmd/compile/internal/gc/esc.go

```
880 »       // Top level defers arguments don't escape to heap, but they
881 »       // do need to last until end of function. Tee with a
882 »       // non-transient location to avoid arguments from being
883 »       // transiently allocated.
884 »       if where.Op == ODEFER && e.loopDepth == 1 {
885 »       »       where.Esc = EscNever // force stack allocation of defer record (see ssa.go)
886 »       »       // TODO(mdempsky): Eliminate redundant EscLocation allocs.
887 »       »       return e.teeHole(k, e.newLoc(nil, false).asHole())
888 »       }
889
890 »       return e.heapHole()
891 }
```

src/cmd/compile/internal/gc/escape.go

“

Proposal: Smarter Scavenging

Out Of Memory(OOM)는 Go의 커다란 허들 중 하나입니다.

일시적으로 필요 메모리가 급증하면 Go 런타임에서 힙 메모리는 커지지만
불필요한 메모리를 시스템으로 다시 반환하는 데는 시간이 몇 분 정도 걸립니다.

시스템에 메모리가 부족할 경우 OOM Killer가 응용 프로그램을 종료시킬 수 있습니다.

현재 이 문제의 해결법은 debug.FreeOSMemory 함수를 실행 시키는 것입니다.
하지만 이 솔루션은 매우 무겁습니다.

debug.FreeOSMemory

- 사용 가능한 모든 메모리를 한 번에 기본 시스템으로 되돌리는 것은 비용이 많이 들고 전체 프로세스에서 힙에 락을 걸어야하므로 대기 시간이 급증 할 수 있습니다.

- 침입(invasive) 솔루션입니다. 사용자가 직접 호출해야 합니다.

- 사용 가능한 메모리 청크를 다시 가져오는 것은 매우 비쌉니다.
  UNIX계열 시스템에서 extra page fault를 의미합니다 (일부 시스템에서는 놀랍게도 비쌉니다).

Background

- Go 1.11에서는 2.5 분마다 실행되는 소거 프로세스(scavenger)를 Go 런타임에 추가하였습니다.
  이 프로세스는 힙에 있는 모든 사용 가능한 스팬(메모리 조각)을 스캔하여 **최소 5분 동안 사용하지 않은 경우** 소거합니다. 주기적으로 작동하는 소거 프로세스는 응용 프로그램의 메모리 사용량 변경에 충분히 빠르게 반응하지 않았습니다.

- Go 1.12부터 주기적으로 작동하는 scavenger 외에도 Go 런타임은 **힙 메모리를 할당 하기전 소거를 추가적으로 수행** 합니다. 힙 메모리를 할당할때마다 스팬에서 최대 N 바이트까지 소거합니다. 여기서 N은 힙이 증가한 바이트 수입니다. 이 아이디어는 힙 성장 비용을 “회수”하는 것 입니다.
  이 기술은 일부 응용 프로그램의 최대 RSS를 줄이는 데 도움이 되었습니다.

Goal

- 더 스마트한 소거의 목표는 두 가지입니다.
    1. Go 애플리케이션의 평균 및 피크 RSS를 줄입니다.
    2. RSS를 낮게 유지함으로써 CPU에 미치는 영향을 최소화 하는 것.
- 한편으로, 응용 프로그램의 RSS를 사용중인 메모리 사용량에 최대한 가깝게 유지하려고 하면 CPU 시간 측면에서 비용이 많이 들며 우리가 가진 모든 여유 공간을 소거하면 모든 스팬 할당에서 하드 페이지 폴트가 발생합니다.

- 이상적인 시나리오는
    1. RSS를 실제 사용중인 힙에 가깝게 유지해야하지만 응용 프로그램에서 할당되지 않은 메모리 풀을 갖도록 충분한 버퍼를 남겨 두십시오.
    2. 힙 크기의 빠르고 일시적인 변경을 부드럽게 처리해야합니다.
- 이 제안의 목표는 설명 된 동작을 나타내도록 Go 런타임의 소거 메커니즘을 개선하는 것입니다. 오늘날의 구현과 비교할 때 이 동작은 성능에 미치는 영향을 최소화하면서 대부분의 Go 응용 프로그램의 평균 전체 RSS를 줄여야합니다.

Proposal

- 세 가지 질문은 메모리 청소 시스템을 설명하는 주요 정책 결정을 나타냅니다.

1.메모리 청소 속도는 어느 정도입니까?

2.얼마나 많은 메모리를 유지해야합니까?

3.어떤 메모리를 청소해야합니까?

Proposal

- 세 가지 질문은 메모리 청소 시스템을 설명하는 주요 정책 결정을 나타냅니다.

1.메모리 청소 속도는 어느 정도입니까? -> 메모리를 할당하는 속도에 비례하는 속도로

2.얼마나 많은 메모리를 유지해야합니까? -> C*max(heap goal, max(heap goal over the last N GCs))

3.어떤 메모리를 청소해야합니까? -> 가장 큰 메모리 청크를 먼저 선택

runtime: add background scavenger

```
// gcenable is called after the bulk of the runtime initialization,
// just before we're about to start letting user code run.
// It kicks off the background sweeper goroutine, the background
// scavenger goroutine, and enables GC.
func gcenable() {
»       // Kick off sweeping and scavenging.
»       c := make(chan int, 2)
»       go bgsweep(c)
»       go bgscavenge(c)
»       <-c
»       <-c
»       memstats.enablegc = true // now that runtime is initialized, GC is okay
}
```

gc에 백그라운드로 도는 소거 작업 추가

```
func gcPaceScavenger() {
»       // Compute our scavenging goal and align it to a physical page boundary
»       // to make the following calculations more exact.
»       retainedGoal := memstats.next_gc
»       // Add retainExtraPercent overhead to retainedGoal. This calculation
»       // looks strange but the purpose is to arrive at an integer division
»       // (e.g. if retainExtraPercent = 12.5, then we get a divisor of 8)
»       // that also avoids the overflow from a multiplication.
»       retainedGoal += retainedGoal / (1.0 / (retainExtraPercent / 100.0))
»       retainedGoal = (retainedGoal + uint64(physPageSize) - 1) &^ (uint64(physPageSize) - 1)
```

소거하는 메모리 목표 계산

runtime: add background scavenger

```
// Calculate how big we want the retained heap to be
// at this point in time.
//
// The formula is for that of a line, y = b - mx
// We want y (want),
//   m = scavengeBytesPerNS (> 0)
//   x = time between scavengeTimeBasis and now
//   b = scavengeRetainedBasis
rate := mheap_.scavengeBytesPerNS
tdist := nanotime() - mheap_.scavengeTimeBasis
rdist := uint64(rate * float64(tdist))
want := mheap_.scavengeRetainedBasis - rdist
```

소거 프로세스 실행 주기 계산

```
// On systems where we have huge pages, we want to do as much of the
// scavenging work as possible on huge pages, because the costs are the
// same per page, but we can give back more more memory in a shorter
// period of time.
if physHugePageSize != 0 {
»       // Start by computing the amount of free memory we have in huge pages
»       // in total. Trivially, this is all the huge page work we need to do.
»       hugeWork := uint64(mheap_.free.unscavHugePages * physHugePageSize)

»       // ...but it could turn out that there's more huge work to do than
»       // total work, so cap it at total work. This might happen for very large
»       // heaps where the additional factor of retainExtraPercent can make it so
»       // that there are free chunks of memory larger than a huge page that we don't want
»       // to scavenge.
»       if hugeWork >= totalWork {
»       »       hugePages := totalWork / uint64(physHugePageSize)
»       »       hugeWork = hugePages * uint64(physHugePageSize)
»       }
»       // Everything that's not huge work is regular work. At this point we
»       // know huge work so we can calculate how much time that will take
»       // based on scavengePageRate (which applies to pages of any size).
»       regularWork = totalWork - hugeWork
»       hugeTime = hugeWork / uint64(physHugePageSize) * scavengeHugePagePeriod
}
```

소거 할 가장 큰 페이지 찾기

Randomize package initialization order in race mode

- 이제 -race 모드일때 패키지들의 init 함수가 무작위로 호출 됩니다.
- 패키지 초기화 순서에 의존 하지 않도록 사람들을 강제하는 첫번째 시도입니다.
- 향후 패키지 초기화가 항상 무작위로 호출되게 바뀔 수 있습니다.

```
func doInit(t *initTask) {
        switch t.state {
        case 2: // fully initialized
                return
        case 1: // initialization in progress
                throw("recursive call during initialization - linker skew")
        default: // not initialized yet
                t.state = 1 // initialization in progress
                if raceenabled {
                        // Randomize initialization order of packages t depends on.
                        // TODO: enable always instead of just for race?
                        s := *(*[]uintptr)(unsafe.Pointer(&slice{array: add(unsafe.Pointer(t), 3*sys.PtrSize), len: int(t.ndeps), cap: int(t.ndeps)}))
                        for i := len(s) - 1; i > 0; i-- {
                                j := int(fastrandn(uint32(i + 1)))
                                s[i], s[j] = s[j], s[i]
                        }
                }
                for i := uintptr(0); i < t.ndeps; i++ {
                        p := add(unsafe.Pointer(t), (3+i)*sys.PtrSize)
                        t2 := *(**initTask)(p)
                        doInit(t2)
                }
                for i := uintptr(0); i < t.nfns; i++ {
                        p := add(unsafe.Pointer(t), (3+t.ndeps+i)*sys.PtrSize)
                        f := *(*func())(unsafe.Pointer(&p))
                        f()
                }
                t.state = 2 // initialization done
        }
}
```

src/runtime/proc.go

SECTION FIVE

Core library

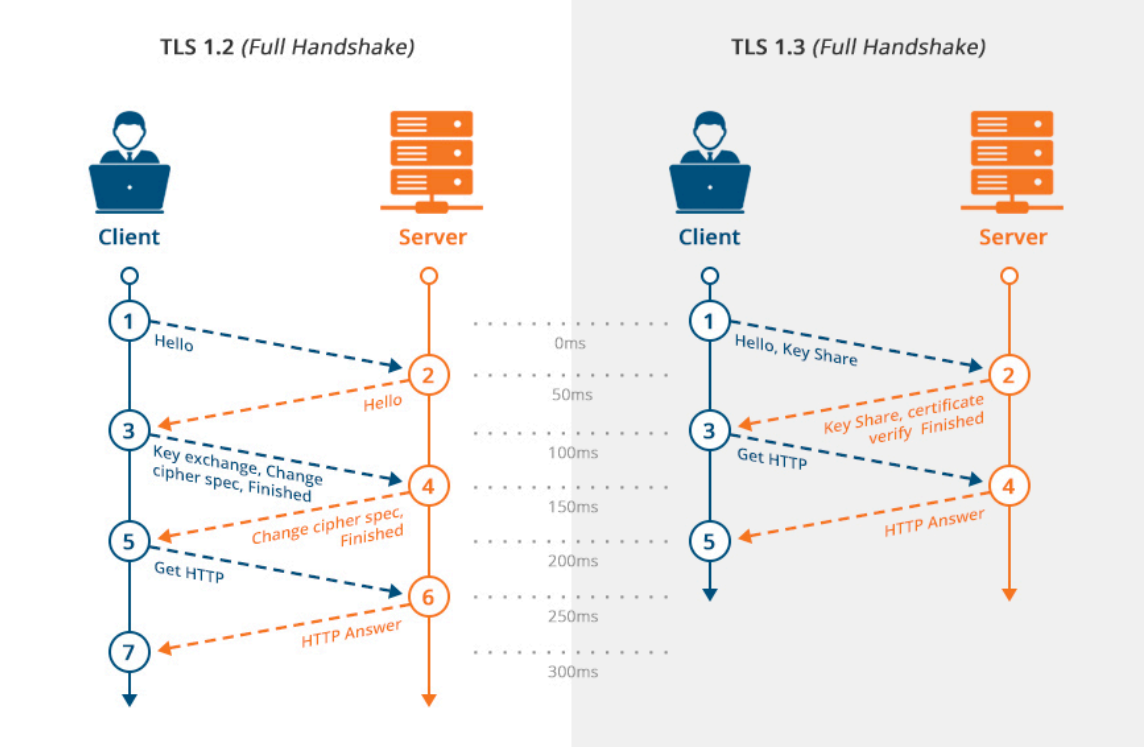
Core library

- crypto/tls 패키지에서 TLS 1.3를 디폴트로 지원
- golang.org/x/crypto/ed25519 -> crypto/ed25519
- Error
  - Un
  - %v
  - IS,

TLS 1.3 요약 1

좀 더 빠르게

좀 더 안전하게



https://www.ssl2buy.com/wiki/tls-1-3-protocol-released-move-ahead-to-advanced-security-and-privacy

TLS 1.3 요약 2

TLS 1.3에서 제거 된 기능

- Static RSA handshake
- CBC MtE modes
- RC4
- SHA1, MD5
- Compression
- Renegotiation

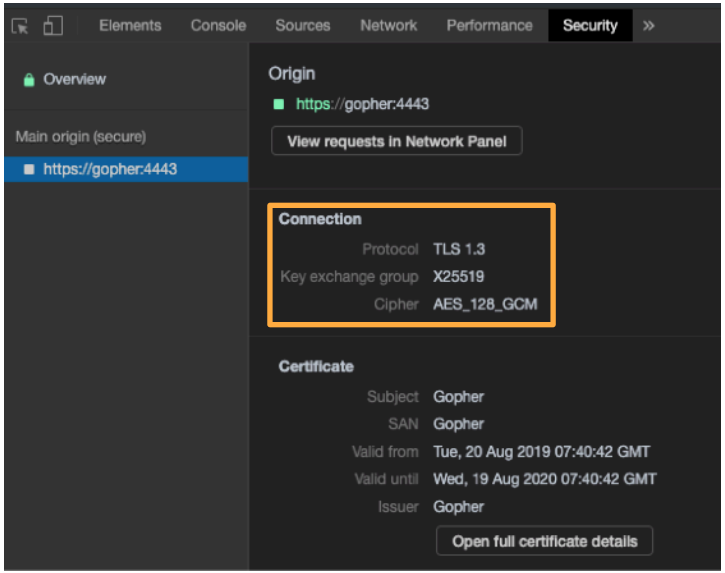
TLS 1.3에서 추가 된 기능

- Abbreviated TLS/SSL handshake
- 0-RRT session resumption
- Full handshake signature
- Downgrade protection
- Abbreviated resumption with optional (EC)DHE
- Curve 25519 and 448

https://www.cloudflare.com/learning-resources/tls-1-3/

TLS 1.3 default on!!

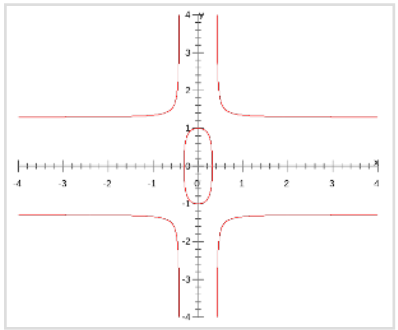
```go
func HelloServer(w http.ResponseWriter, req *http.Request) {
    w.Header().Set( key: "Content-Type", value: "text/plain")
    w.Write([]byte("This is an example server.\n"))
}

func main() {
    http.HandleFunc( pattern: "/hello", HelloServer)
    log.Fatal(
        http.ListenAndServeTLS(
            addr: "gopher:4443",
            certFile: "server.crt",
            keyFile: "server.key",
            handler: nil,
        ),
    )
}
```



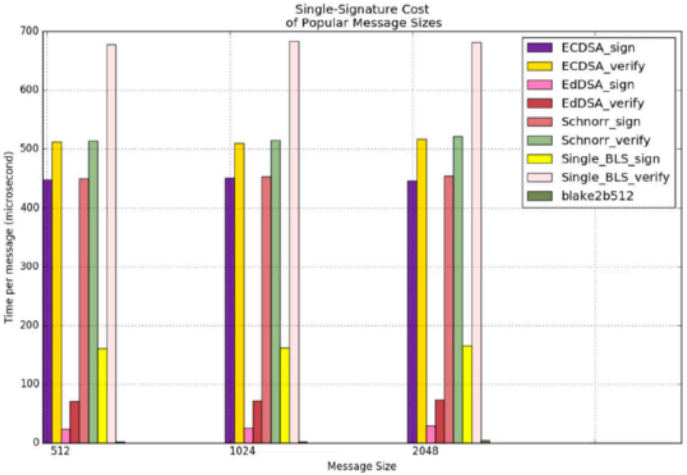
TLS 1.3를 디폴트로 지원

crypto/ed25519

- Ed25519는 공개 키 서명 알고리즘입니다.
- Ed25519는 ECDSA(Elliptic Curve Digital Signature Algorithm)가 아니라 EdDSA(Edwards-Curve Digital Signature Algorithm)기반이며 **보안을 유지하면서 기존 디지털 서명 체계보다 더 빠르도록 설계되었습니다.**
- 특징으로는 secret array indices, secret branch conditions가 없어 spectre, meltdown 공격에 영향을 받지 않고 세션 키가 해시 함수로 생성되므로 난수 생성기가 더 이상 필요로 하지 않으며 서명을 만드는데 사용한 난수 생성기가 개인키를 공개 할 위험이 없습니다.(ECDSA, Playstation 3 해킹 이슈)



Twisted Edwards curve



https://medium.com/logos-network/benchmarking-hash-and-signature-algorithms-6079735ce05

SECTION SIX

Minor changes to the library

Minor changes to the library 1

- bytes/strings
  - Add ToValidUTF8 function
- context
  - remove dependency on fmt
- database/sql
  - Add NullTime, NullInt32 type
- rerflect
  - Add Value.IsZero method
- sync
  - Large Pool no longer increase stop-the-world pause times.
  - Pool no longer needs to be completely repopulated after every GC. It now retains some objects across GCs, as opposed to releasing all objects, reducing load spikes for heavy users of Pool.

Minor changes to the library 2

- net
  - On Unix systems where use-vc is set in resolve.conf, TCP is used for DNS resolution.
  - The new field ListenConfig.KeepAlive
- net/http
  - Add ForceAttemptHTTP2 field
  - Add ReadBufferSize, WriteBufferSize field
  - When reusing HTTP/2, the Transport no longer performs unnecessary TLS handshakes.
  - TimeoutHandler's ResponseWriter now implements the Pusher and Flusher interfaces.
  - The StatusCode 103 "Early Hints" has been added.
  - Transport now uses the Request.Body's io.ReaderFrom implementation if available, to optimize writing the body.
  - On encountering unsupported transfer-encodings, http.Server now returns a "501 Unimplemented" status as mandated by the HTTP specification RFC 7230 Section 3.3.1.
  - The new Server fields BaseContext and ConnContext allow finer control over the Context values provided to requests and connections.
  - http.DetectContentType now correctly detects RAR signatures, and can now also detect RAR v5 signatures.
  - The new Header method Clone returns a copy of the receiver.
  - A new function NewRequestWithContext has been added and it accepts a Context that controls the entire lifetime of the created outgoing Request, suitable for use with Client.Do and Transport.RoundTrip.
  - Transport now silently ignores a 408 "Request Timeout" response.

Minor changes to the library 3

- testing
  - When running benchmarks, B.N is no longer rounded.
  - The new method B.ReportMetric lets users report custom benchmark metrics and override built-in metrics.
  - Testing flags are now registered in the new Init function.
- text/template: The new slice function returns the result of slicing its first argument by the following arguments.
- time
  - Day-of-year is now supported by Format and Parse.
  - The new Duration methods Microsecond, Millisecond
- unicode: Unicode 10.0 to Unicode 11.0
- math/bits: guarantee Add, Sub, Mul, RotateLeft, ReverseBytes to be constant-time operations
- cmd/compile: evaluate map initializers incrementally
- cmd/go: load packages in parallel

bytes/strings

- ToValidUTF8 함수 추가
- ToValidUTF8 함수는 유효하지 않는 부분을

  replacement 부분으로 대체해서 []byte or string의 사본을 리턴시킵니다.

```
// ToValidUTF8 treats s as UTF-8-encoded bytes and returns a copy with each run of bytes
// representing invalid UTF-8 replaced with the bytes in replacement, which may be empty.
func ToValidUTF8(s, replacement []byte) []byte {...}
```

```
// ToValidUTF8 returns a copy of the string s with each run of invalid UTF-8 byte sequences
// replaced by the replacement string, which may be empty.
func ToValidUTF8(s, replacement string) string {...}
```

context

- fmt 패키지 종속성 제거
- 이유는 net패키지의 유니코드 테이블에 종속성을 제거하기 위해서 (net: remove dependency on unicode)
- net -> context -> fmt -> unicode tables

```
// ToValidUTF8 treats s as UTF-8-encoded bytes and returns a copy with each run of bytes
// representing invalid UTF-8 replaced with the bytes in replacement, which may be empty.
func ToValidUTF8(s, replacement []byte) []byte {...}
```

```
// ToValidUTF8 returns a copy of the string s with each run of invalid UTF-8 byte sequences
// replaced by the replacement string, which may be empty.
func ToValidUTF8(s, replacement string) string {...}
```

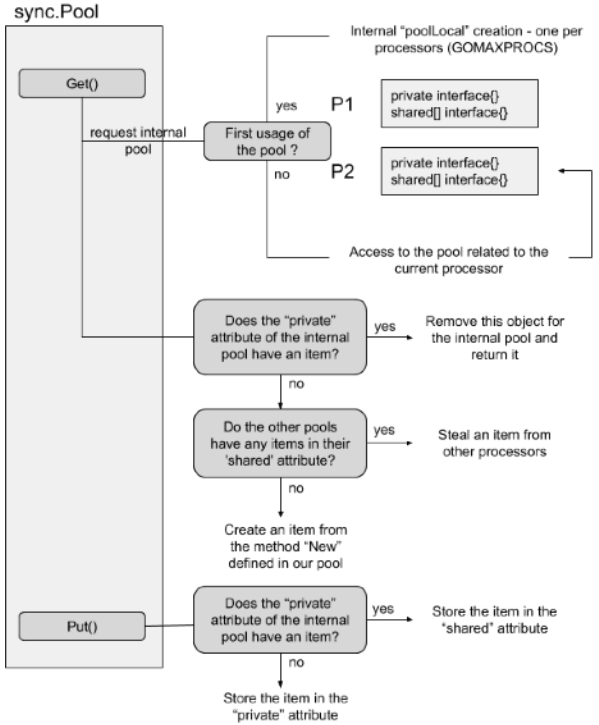
database/sql: support more nullable types

```
type NullBool
type NullFloat64
type NullInt64
type NullString
type NullTime     // + Go 1.13
type NullInt32    // + Go 1.13
```

- Unsigned integers
- Non-64 bit width integers
- Timestamps
- Runes
- Structs

(esp. for JSON SQL columns)

sync: use lock-free structure for Pool stealing



sync.Pool workflow in Go 1.12

https://medium.com/@blanchon.vincent/go-understand-the-design-of-sync-pool-2dde3024e277

- sync.Pool은 각 P(processors)마다 poolLocal을 가지고 있습니다.
- poolLocal은 자기 자신만 쓸수 있는 private와 다른 P도 가져올 수 있는 shared가 있습니다.
- 기존에는 shared의 데이터를 다른 P가 가져갈때(stealing), 풀을 지울때(clearing) Mutex를 사용합니다.
- Pool은 sweep 종료중에 지워지므로 다른 P에서 해당 Pool에 Get/Put하려고 시도하면 교착상태에 빠질 수 있습니다.

sync: use lock-free structure for Pool stealing

```
// Local per-P Pool appendix.
type poolLocalInternal struct {
        private interface{}   // Can be used only by the respective P.
        shared  []interface{} // Can be used by any P.
        Mutex                 // Protects shared.
}
```

```
// Local per-P Pool appendix.
type poolLocalInternal struct {
        private interface{} // Can be used only by the respective P.
        shared  poolChain   // Local P can pushHead/popHead; any P can popTail.
}
```

```
func (p *Pool) Get() interface{} {
        if race.Enabled {
                race.Disable()
        }
        l := p.pin()
        x := l.private
        l.private = nil
        runtime_procUnpin()
        if x == nil {
                l.Lock()
                last := len(l.shared) - 1
                if last >= 0 {
                        x = l.shared[last]
                        l.shared = l.shared[:last]
                }
                l.Unlock()
                if x == nil {
                        x = p.getSlow()
                }
        }
        if race.Enabled {
                race.Enable()
                if x != nil {
                        race.Acquire(poolRaceAddr(x))
                }
        }
        if x == nil && p.New != nil {
                x = p.New()
        }
        return x
}
```

```
func (p *Pool) Get() interface{} {
        if race.Enabled {
                race.Disable()
        }
        l, pid := p.pin()
        x := l.private
        l.private = nil
        if x == nil {
                // Try to pop the head of the local shard. We prefer
                // the head over the tail for temporal locality of
                // reuse.
                x, _ = l.shared.popHead()
                if x == nil {
                        x = p.getSlow(pid)
                }
        }
        runtime_procUnpin()
        if race.Enabled {
                race.Enable()
                if x != nil {
                        race.Acquire(poolRaceAddr(x))
                }
        }
        if x == nil && p.New != nil {
                x = p.New()
        }
        return x
}
```

```
// Put adds x to the pool.
func (p *Pool) Put(x interface{}) {
        if x == nil {
                return
        }
        if race.Enabled {
                if fastrand()%4 == 0 {
                        // Randomly drop x on floor.
                        return
                }
                race.ReleaseMerge(poolRaceAddr(x))
                race.Disable()
        }
        l := p.pin()
        if l.private == nil {
                l.private = x
                x = nil
        }
        runtime_procUnpin()
        if x != nil {
                l.Lock()
                l.shared = append(l.shared, x)
                l.Unlock()
        }
        if race.Enabled {
                race.Enable()
        }
}
```

```
// Put adds x to the pool.
func (p *Pool) Put(x interface{}) {
        if x == nil {
                return
        }
        if race.Enabled {
                if fastrand()%4 == 0 {
                        // Randomly drop x on floor.
                        return
                }
                race.ReleaseMerge(poolRaceAddr(x))
                race.Disable()
        }
        l, _ := p.pin()
        if l.private == nil {
                l.private = x
                x = nil
        }
        if x != nil {
                l.shared.pushHead(x)
        }
        runtime_procUnpin()

        if race.Enabled {
                race.Enable()
        }
}
```

src/sync/pool.go

sync: use lock-free structure for Pool stealing

	time/op		delta	
Pool-12	3.00ns ± 0%	2.21ns ±36%	**−26.32%**	(p=0.000 n=18+19)
PoolOverflow-12	600ns ± 1%	587ns ± 1%	**−2.21%**	(p=0.000 n=16+18)
PoolSTW-12	71.0µs ± 2%	5.6µs ± 3%	**−92.15%**	(p=0.000 n=20+20)
PoolExpensiveNew-12	3.14ms ± 5%	3.69ms ± 7%	**+17.67%**	(p=0.000 n=19+20)
[Geo mean]	4.48µs	2.27µs	−49.21%	

	p50-ns/STW		delta	
PoolSTW-12	70.7k ± 1%	5.5k ± 2%	**−92.25%**	(p=0.000 n=20+20)

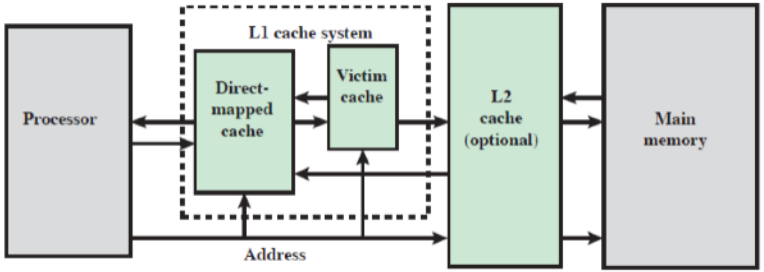
	p95-ns/STW		delta	
PoolSTW-12	73.1k ± 2%	6.7k ± 4%	**−90.86%**	(p=0.000 n=18+19)

	GCs/op		delta	
PoolExpensiveNew-12	0.38 ± 1%	0.39 ± 1%	**+2.07%**	(p=0.000 n=20+18)

	New/op		delta	
PoolExpensiveNew-12	33.9 ± 6%	40.0 ± 6%	**+17.97%**	(p=0.000 n=19+20)

(https://perf.golang.org/search?q=upload:20190311.1)

sync: smooth out Pool behavior over GC with a victim cache



Victim Cache
https://www.ece.ucsb.edu/~strukov/ece154bWinter2015/week3.pdf

- 현재 sync.Pool은 GC가 시작 될 때 모두 지워집니다.
- 풀을 비운 직후에는 할당량이 급증하여 처리량과 대기 시간에 영향을 미치므로 2차 캐시를 추가하였습니다.
- 이 캐시 메커니즘은 primary 캐시를 지우는 대신 victim 캐시로 이동합니다.

sync: smooth out Pool behavior over GC with a victim cache

```
259 var (
260 »       allPoolsMu Mutex
261
262 »       // allPools is the set of pools that have non-empty primary
263 »       // caches. Protected by either 1) allPoolsMu and pinning or 2)
264 »       // STW.
265 »       allPools []*Pool
266
267 »       // oldPools is the set of pools that may have non-empty victim
268 »       // caches. Protected by STW.
269 »       oldPools []*Pool
270 )
```

```
233 func poolCleanup() {
234 »       // This function is called with the world stopped, at the beginning of a garbage collection.
235 »       // It must not allocate and probably should not call any runtime functions.
236
237 »       // Because the world is stopped, no pool user can be in a
238 »       // pinned section (in effect, this has all Ps pinned).
239
240 »       // Drop victim caches from all pools.
241 »       for _, p := range oldPools {
242 »       »       p.victim = nil
243 »       »       p.victimSize = 0
244 »       }
245
246 »       // Move primary cache to victim cache.
247 »       for _, p := range allPools {
248 »       »       p.victim = p.local
249 »       »       p.victimSize = p.localSize
250 »       »       p.local = nil
251 »       »       p.localSize = 0
252 »       }
253
254 »       // The pools with non-empty primary caches now have non-empty
255 »       // victim caches and no pools have primary caches.
256 »       oldPools, allPools = allPools, nil
257 }
```

sync: smooth out Pool behavior over GC with a victim cache

	time/op		delta	
Pool-12	2.21ns ±36%	2.00ns ± 0%	~	(p=0.070 n=19+16)
PoolOverflow-12	587ns ± 1%	583ns ± 1%	**−0.77%**	(p=0.000 n=18+18)
PoolSTW-12	5.57µs ± 3%	4.52µs ± 4%	**−18.82%**	(p=0.000 n=20+19)
PoolExpensiveNew-12	3.69ms ± 7%	1.25ms ± 5%	**−66.25%**	(p=0.000 n=20+19)
[Geo mean]	2.27µs	1.60µs	−29.58%	

	p50-ns/STW		delta	
PoolSTW-12	5.48k ± 2%	4.53k ± 2%	**−17.32%**	(p=0.000 n=20+20)

	p95-ns/STW		delta	
PoolSTW-12	6.69k ± 4%	5.13k ± 3%	**−23.31%**	(p=0.000 n=19+18)

	GCs/op		delta	
PoolExpensiveNew-12	0.39 ± 1%	0.32 ± 2%	**−17.95%**	(p=0.000 n=18+20)

	New/op		delta	
PoolExpensiveNew-12	40.0 ± 6%	12.4 ± 6%	**−68.91%**	(p=0.000 n=20+19)

(https://perf.golang.org/search?q=upload:20190311.2)

cmd/compile: evaluate map initializers incrementally

```
m := map[int]int {
  a(): b(),
  c(): d(),
  e(): f(),
}
```

기존에는

```
t1 := a()
t2 := b()
t3 := c()
t4 := d()
t5 := e()
t6 := f()
m := map[int]int{}
m[t1] = t2
m[t3] = t4
m[t5] = t6
```

```
m := map[int]int{}
t1 := a()
t2 := b()
m[t1] = t2
t3 := c()
t4 := d()
m[t3] = t4
t5 := e()
t6 := f()
m[t5] = t6
```

cmd/compile: evaluate map initializers incrementally

```
m := map[int]int {
  a(): b(),
  c(): d(),
  e(): f(),
}
```

Go 1.13 이후부터는 이렇게 변경

```
t1 := a()
t2 := b()
t3 := c()
t4 := d()
t5 := e()
t6 := f()
m := map[int]int{}
m[t1] = t2
m[t3] = t4
m[t5] = t6
```

```
m := map[int]int{}
t1 := a()
t2 := b()
m[t1] = t2
t3 := c()
t4 := d()
m[t3] = t4
t5 := e()
t6 := f()
m[t5] = t6
```

time: add methods to convert duration to microseconds and milliseconds

```
var d time.Duration = 1000 * 1000 // 1 milliseconds

fmt.Println(d.Seconds())      // 0.001
fmt.Println(d.Milliseconds()) // 1
fmt.Println(d.Microseconds()) // 1000
fmt.Println(d.Nanoseconds())  // 1000000
```

time: add methods to convert duration to microseconds and milliseconds

Unit	`Duration` constants	`Duration` conversion
Nanosecond	`time.Nanosecond`	`func (d Duration) Nanoseconds() int64`
Microsecond	`time.Microsecond`	
Millisecond	`time.Millisecond`	
Second	`time.Second`	`func (d Duration) Seconds() float64`
Minute	`time.Minute`	`func (d Duration) Minutes() float64`
Hour	`time.Hour`	`func (d Duration) Hours() float64`

통일성을 위해 Millisecond, Microsecond 추가제안

time: add methods to convert duration to microseconds and milliseconds

```
783 // Nanoseconds returns the duration as an integer nanosecond count.
784 func (d Duration) Nanoseconds() int64 { return int64(d) }
785
786 // Microseconds returns the duration as an integer microsecond count.
787 func (d Duration) Microseconds() int64 { return int64(d) / 1e3 }
788
789 // Milliseconds returns the duration as an integer millisecond count.
790 func (d Duration) Milliseconds() int64 { return int64(d) / 1e6 }
791
```

src/time/time.go

time: add support for day-of-year in Format and Parse

```
t := time.Date( year: 2019,  month: 2,  day: 1,  hour: 1,  min: 0,  sec: 0,  nsec: 0, time.Local)
fmt.Println(t.Format( layout: "2006-01-02 002 __2 15:04:05"))     // 2019-02-01 032  32 01:00:00
fmt.Println(time.Parse( layout: "2006 002",  value: "2019 032"))   // 2019-02-01 00:00:00 +0000 UTC <nil>
```

Format, Parse 함수가 day-of-year를 지원

testing: add B.ReportMetric for custom benchmark metrics

```go
func BenchmarkSum(b *testing.B) {
    var compares int64
    for i := 0; i < b.N; i++ {
        Sum( a: 1,  b: 2)
        compares++
    }

    // This metric is per-operation, so divide by b.N and
    // report it as a "/op" unit.
    b.ReportMetric(float64(compares)/float64(b.N),  unit: "compares/op")
}
```

```
hyejong ~/Library/Mobile Documents/com~apple~CloudDocs/Source/go1.13/test $ go test -bench=.
goos: darwin
goarch: amd64
pkg: go1.13/test
BenchmarkSum-8          1000000000               0.684 ns/op             1.00 compares/op
PASS
ok      go1.13/test     0.764s
```

Q&A