QCon 全球软件开发大会
INTERNATIONAL SOFTWARE DEVELOPMENT CONFERENCE

BEIJING 2017

头条

Go构建日请求千亿级微服务实践

项超

主办方 Geekbang. 极客邦科技 InfoQ

Go微服务历程

2015年6月，使用Go开发后端核心服务，并且整体开始转向微服务架构。

2016年6月，Feed流后端服务从Python切换到Go语言，节省大量机器，稳定性提高。

Go微服务现状

微服务框架
kite框架，基于Thrift协议

服务发现
基于Consul实现

配置管理和服务降级
基于ETCD实现

监控和追踪
Metrics系统，日志链系统

Go微服务规模

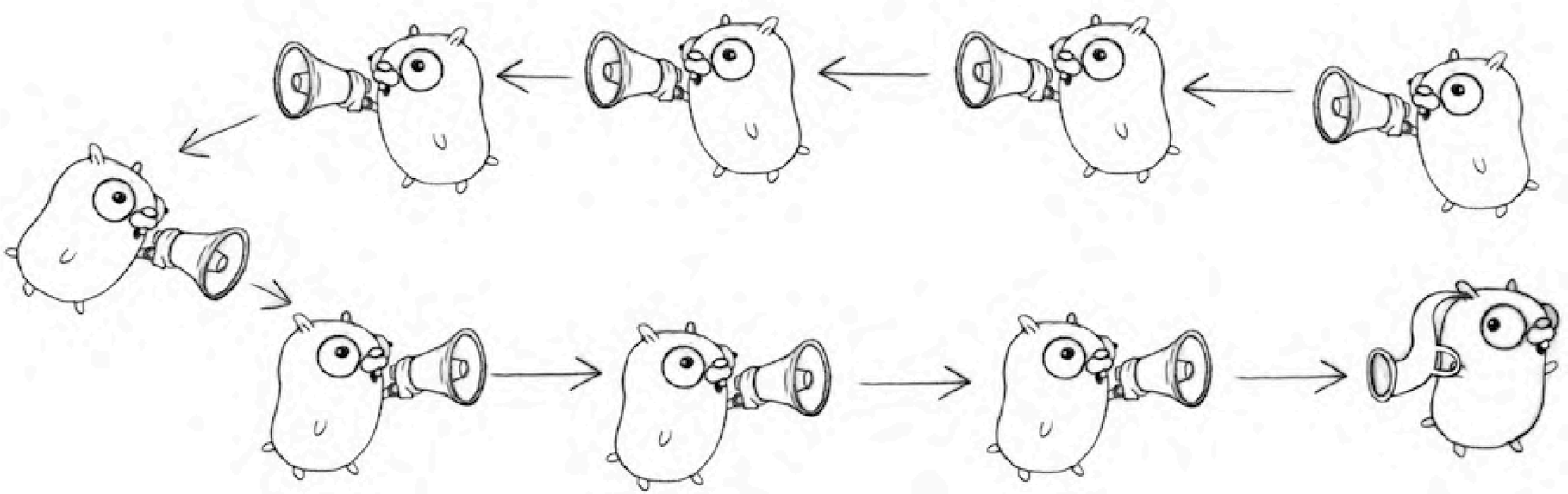
Go微服务数量***100+***，高峰每秒处理请求数超过***700万***，每天处理请求数超过***3000亿***

概览

- 并发
- 性能
- 监控
- 编程思维
- 工程性

并发

Goroutine & Channel

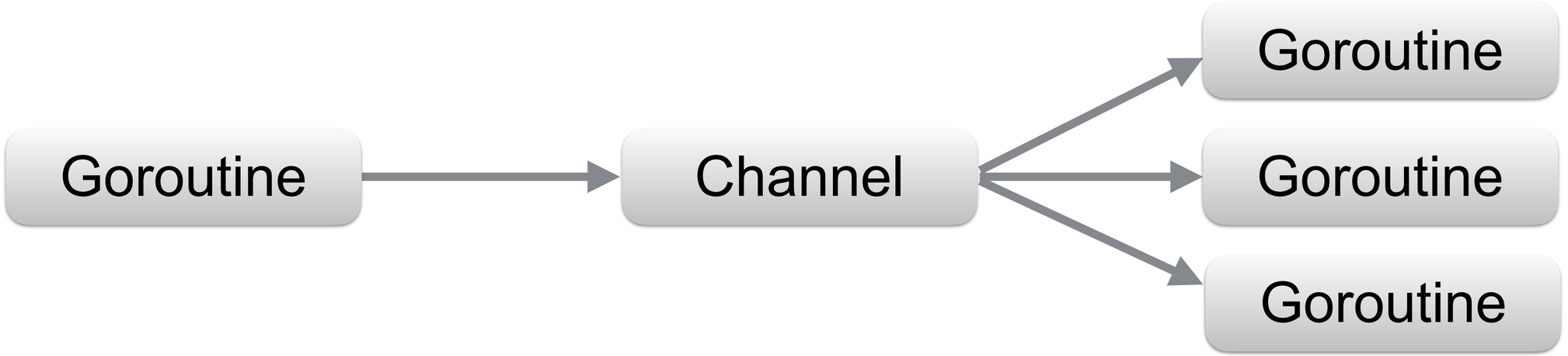


一对一


```
func gen() chan int {
    out := make(chan int)
    go func(){
        for i:=0; i<100; i++ {
            out <- i
        }
    }()
    return out
}
```

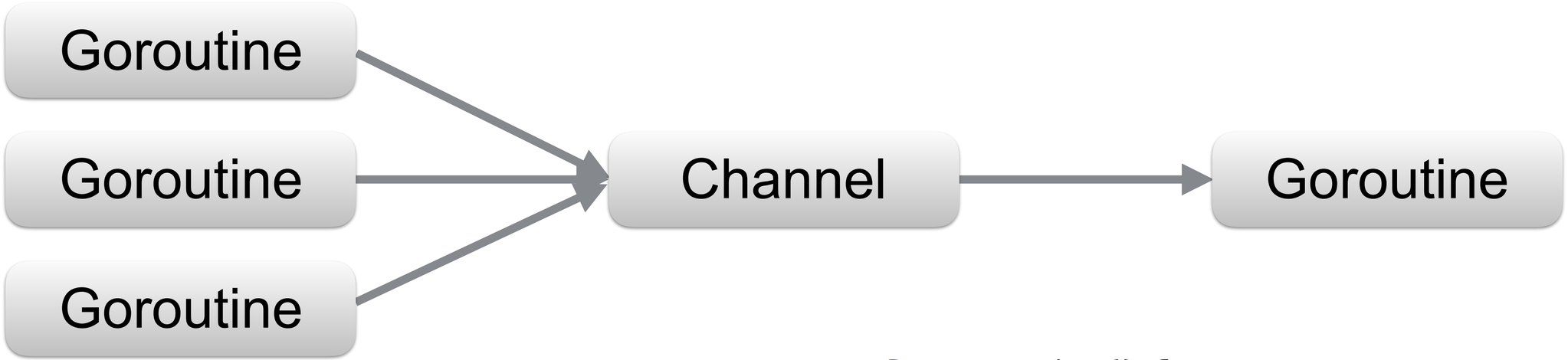
```
func seq(input chan int) {
    for num := range input {
        fmt.Println(num)
    }
}

func main(){
    seq(gen())
}
```

一对多




```
func main(){
    in := gen()
    go seq(in)
    go sep(in)
    go seq(in)
}
```

多对一

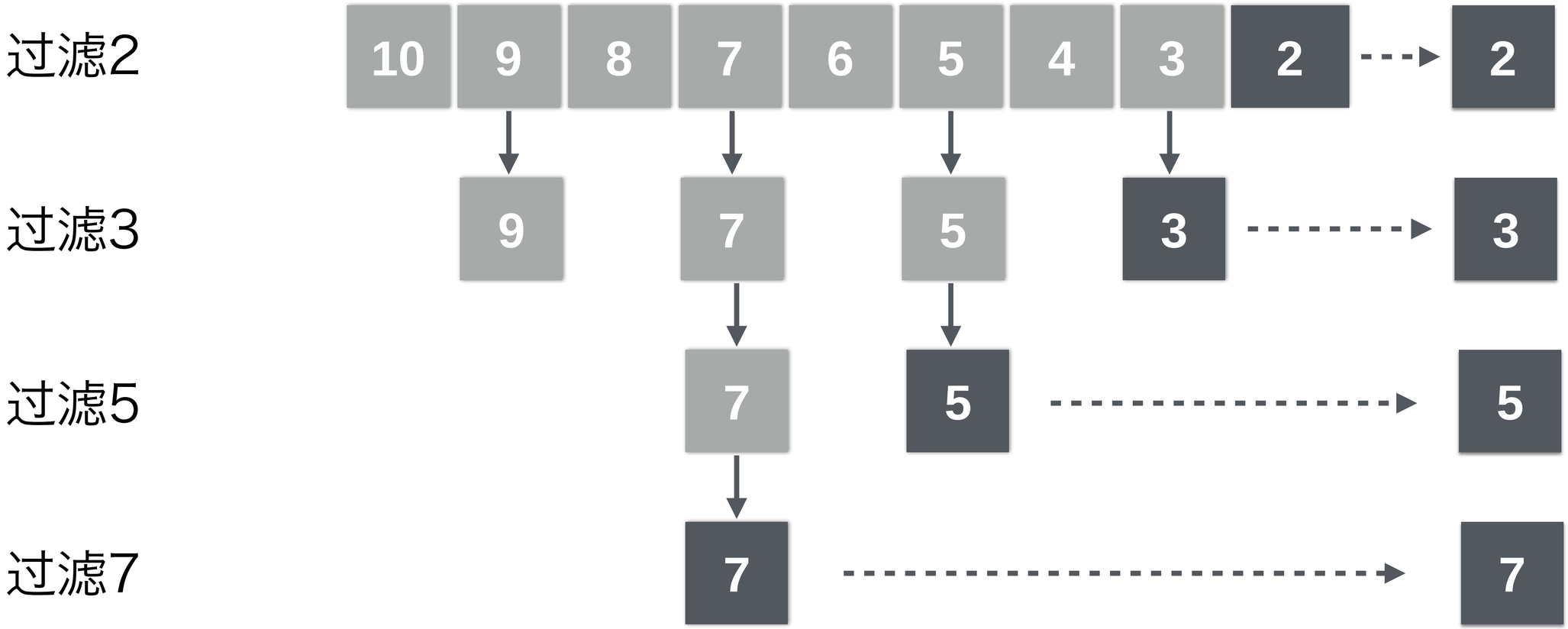



```
func spout(out chan int) {
    for i:=0; i<100; i++ {
        out<-i
    }
}
```

```
func main(){
    out := make(chan int)
    go spout(out)
    go spout(out)
    go spout(out)
    seq(out)
}
```

“Problem: To print in ascending order all primes less than 10000. Use an array of processes, SIEVE, in which each process inputs a prime from its predecessor and prints it. The process then inputs an ascending stream of numbers from its predecessor and passes them on to its successor, suppressing any that are multiples of the original prime.”

–Communicating Sequential Processes

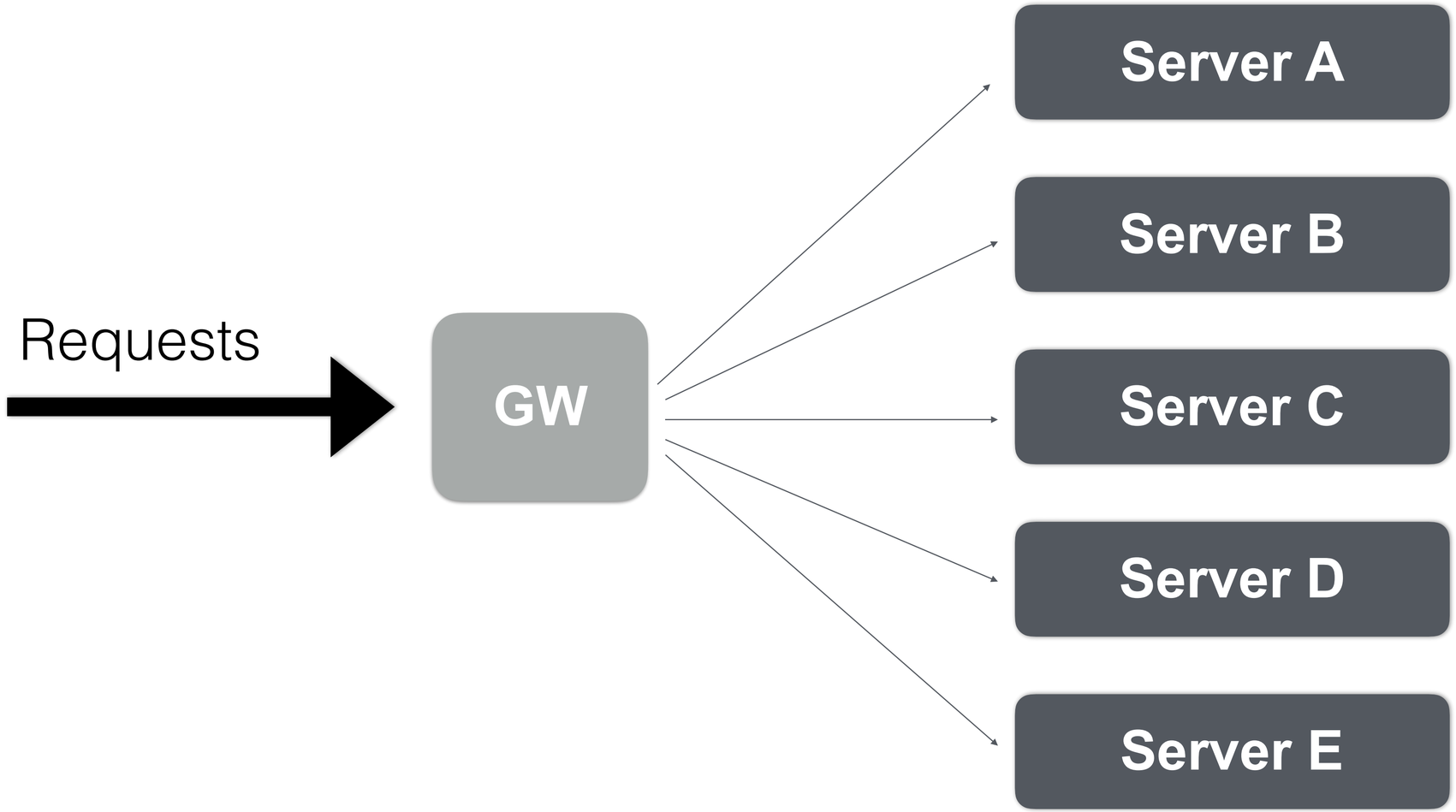

过滤2
10
9
8
7
6
5
4
3
2
2
过滤3
9
7
5
3
3
过滤5
7
5
5
过滤7
7
7

```
func main() {
    origin, wait := make(chan int), make(chan struct{})
    Processor(origin, wait)
    for num := 2; num < 10000; num++ {
        origin <- num
    }
    close(origin)
    <-wait
}
```

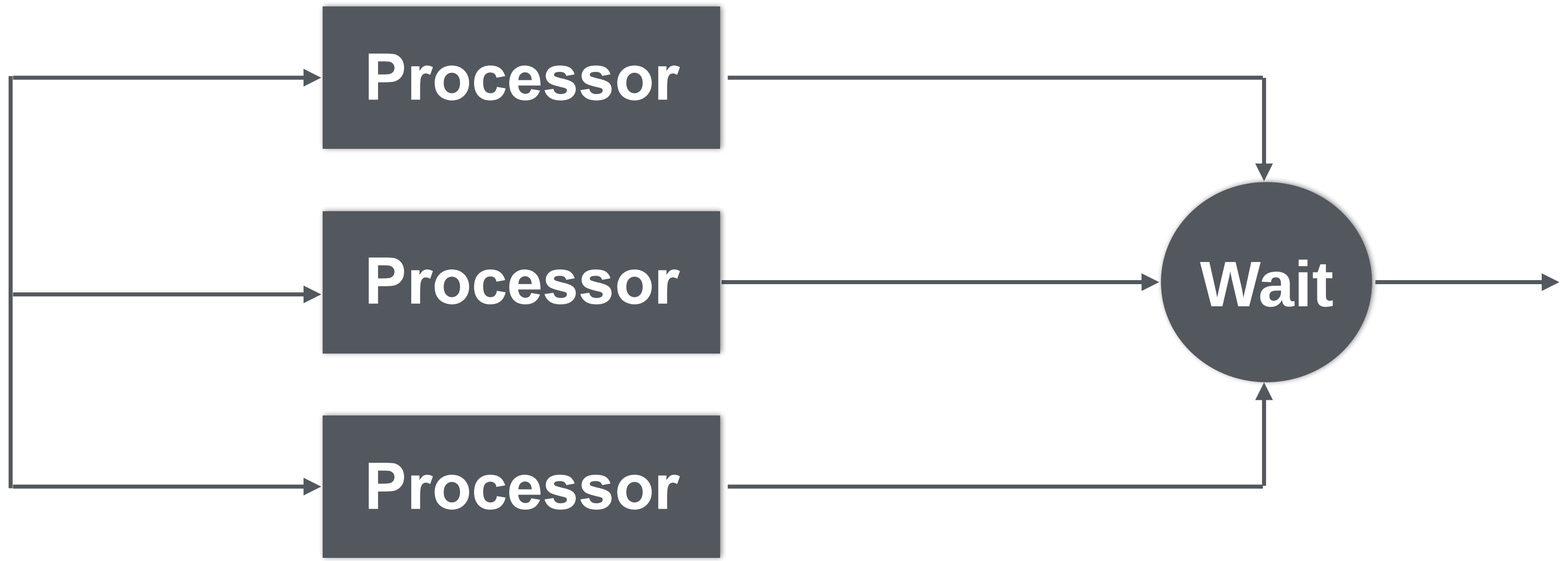
```
func Processor(seq chan int, wait chan struct{}) {
    go func() {
        prime, ok := <-seq
        if !ok {
            close(wait)
            return
        }
        fmt.Println(prime)
        out := make(chan int)
        Processor(out, wait)
        for num := range seq {
            if num%prime != 0 {
                out <- num
            }
        }
        close(out)
    }()
}
```

并发控制

- Wait
- Cancel


Requests
GW
Server A
Server B
Server C
Server D
Server E

Wait

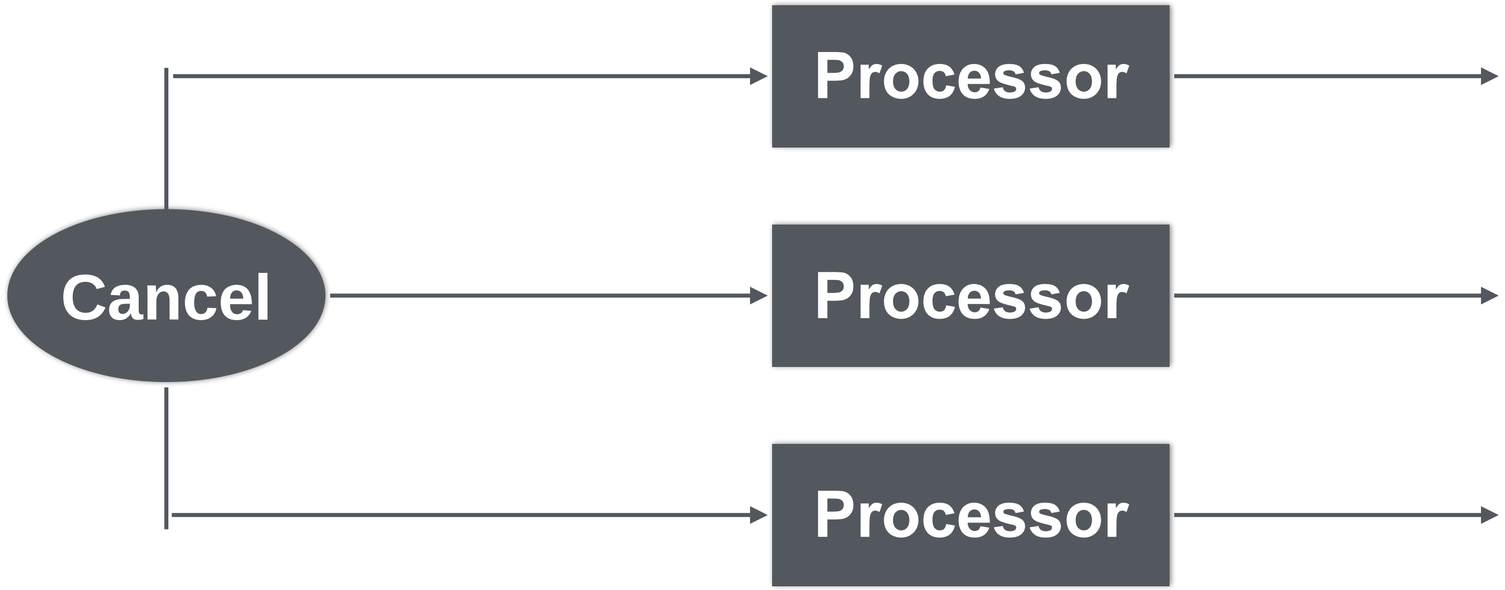


Wait

```
wg := sync.WaitGroup{}
wg.Add(3)

go func() {
    defer wg.Done()
    // do ...
}()
```

```
go func() {
    defer wg.Done()
    // do ...
}()
go func() {
    defer wg.Done()
    // do ...
}()
wg.Wait()
```

Cancel

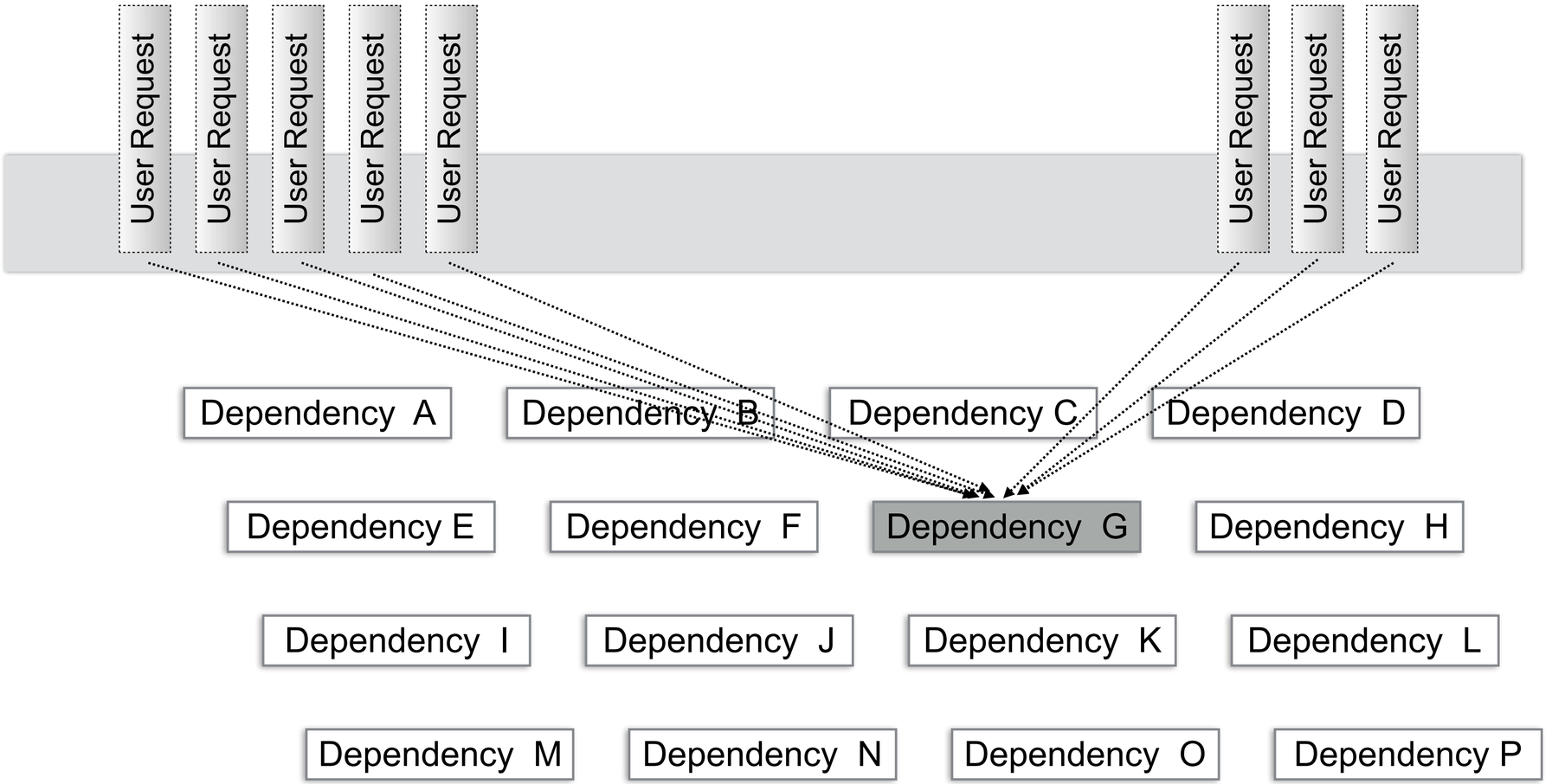


Cancel

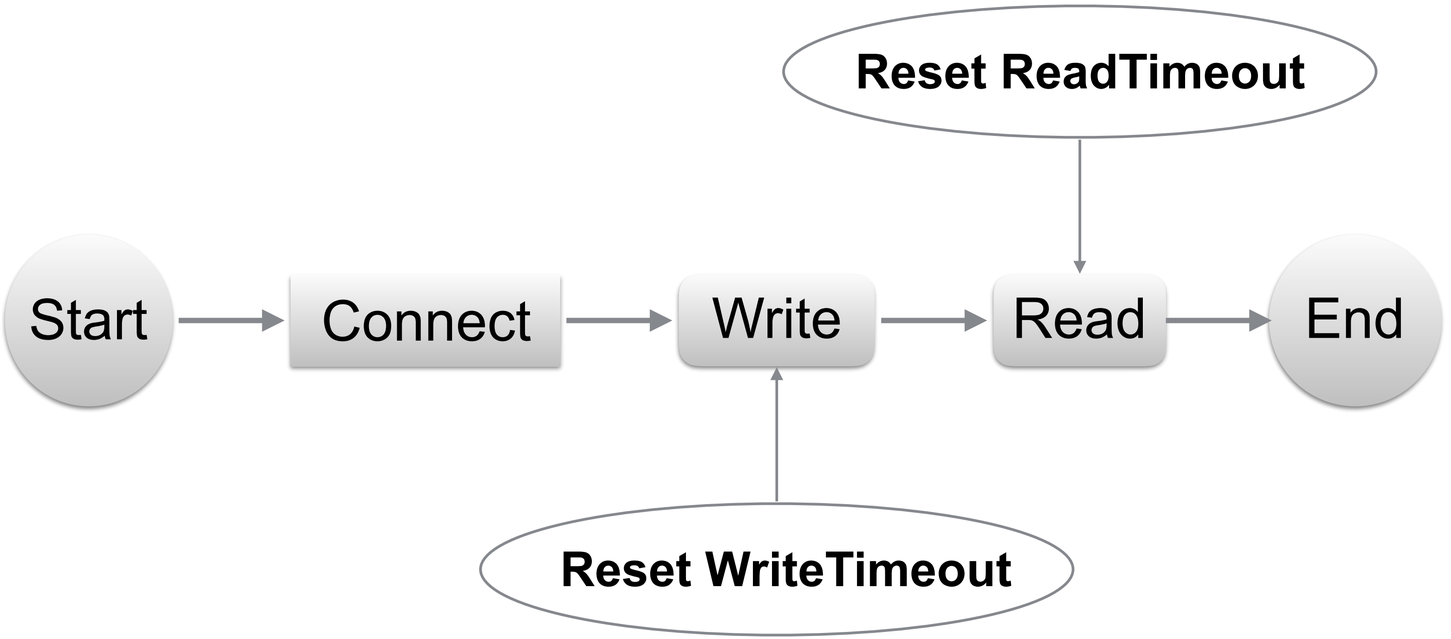
```
type SIG struct{}

func Processor(exit chan SIG) {
    for {
        select {
        case <-exit:
            return
        default:
            // do ...
        }
    }
}
```

```
exit := make(chan SIG)

go Processor(exit)
go Processor(exit)
go Processor(exit)

// Cancel after 1s
time.Sleep(time.Second)
close(exit)
```

超时控制

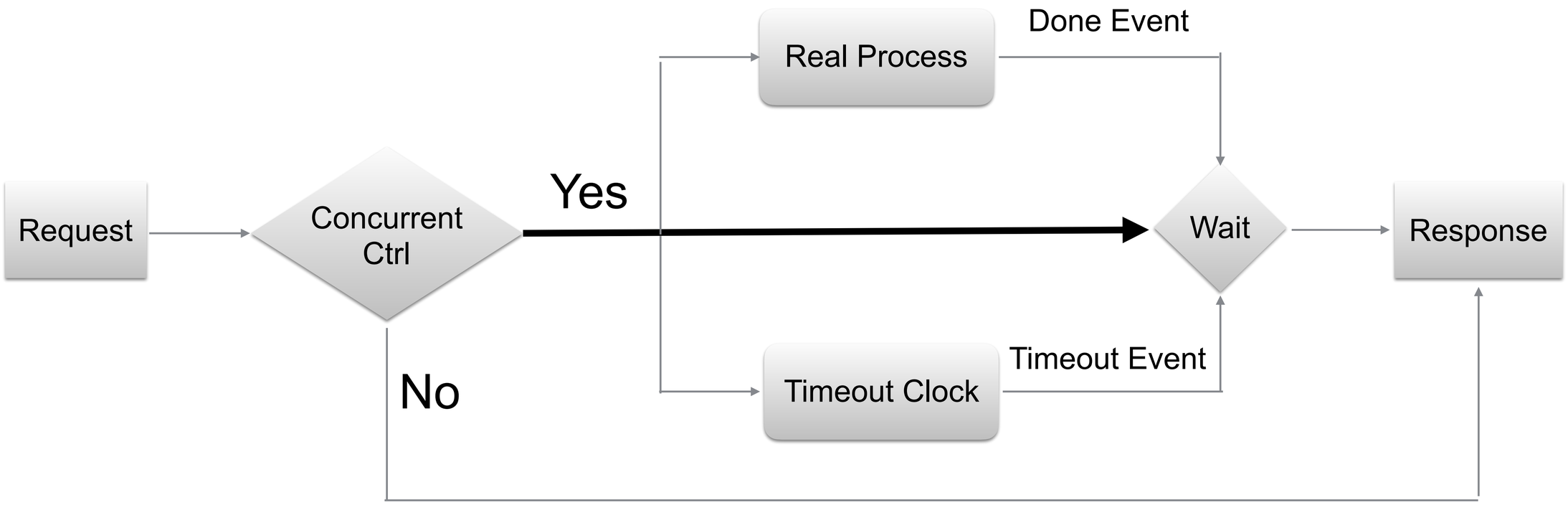

User Request
User Request
User Request
User Request
User Request
User Request
User Request
User Request
Dependency A
Dependency B
Dependency C
Dependency D
Dependency E
Dependency F
Dependency G
Dependency H
Dependency I
Dependency J
Dependency K
Dependency L
Dependency M
Dependency N
Dependency O
Dependency P

网络超时



连接超时　　写超时　　读超时

并发超时控制


Request
Concurrent Ctrl
Yes
No
Real Process
Done Event
Timeout Clock
Timeout Event
Wait
Response

Go1.7 “context”

```
// A Context carries a deadline, cancelation signal, and request-scoped values
// across API boundaries. Its methods are safe for simultaneous use by multiple
// goroutines.
type Context interface {
    // Done returns a channel that is closed when this Context is canceled
    // or times out.
    Done() <-chan struct{}

    // Err indicates why this context was canceled, after the Done channel
    // is closed.
    Err() error

    // Deadline returns the time when this Context will be canceled, if any.
    Deadline() (deadline time.Time, ok bool)

    // Value returns the value associated with key or nil if none.
    Value(key interface{}) interface{}
}
```

```
import (
    "context"
)

func Handler(r *Request) {
    timeout := r.Value("timeout")
    ctx, cancel := context.WithTimeout(context.Background(), timeout)
    defer cancel()
    done := make(chan struct{}, 1)
    go func() {
        Do(ctx, ...)
        done <- struct{}
    }()
    select{
    case <-done:
        // nice ...
    case <-ctx.Done():
        // timeout ...
    }
}
```

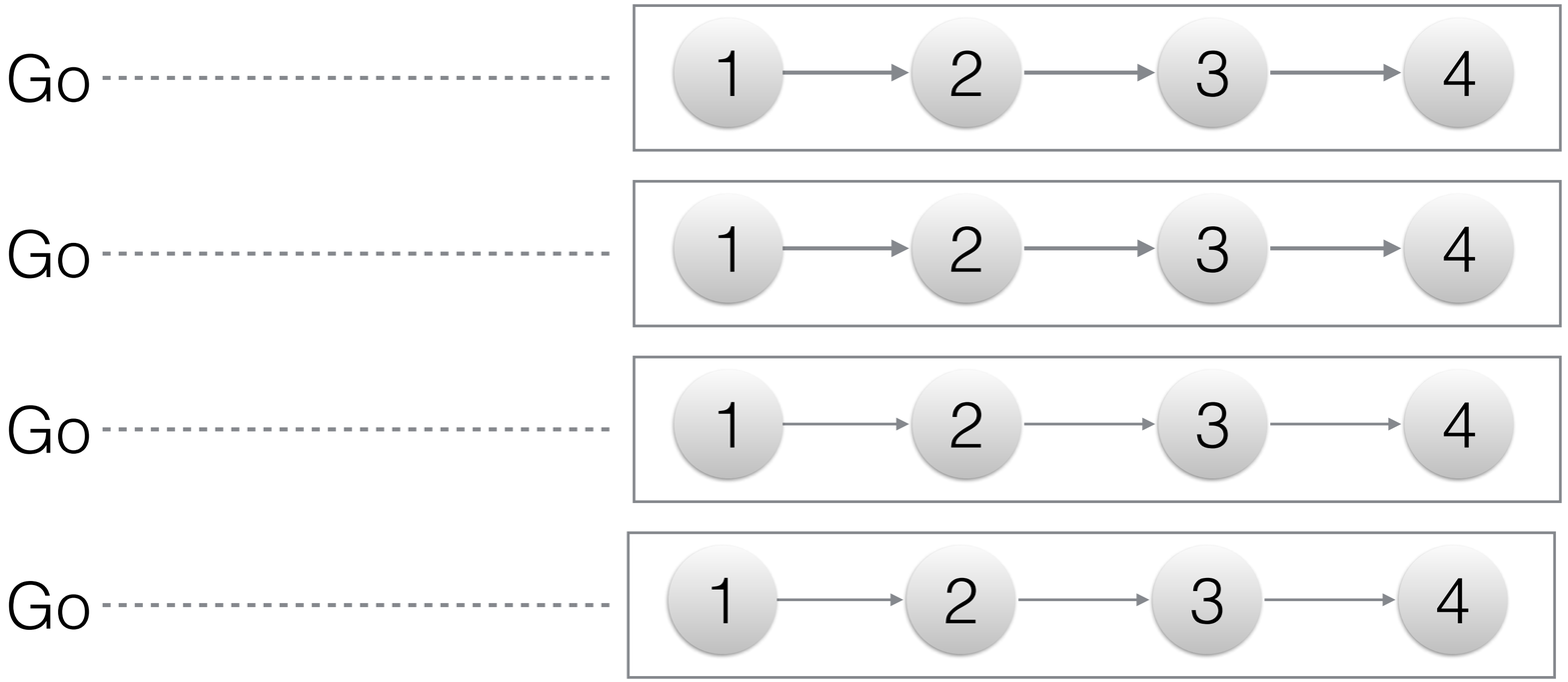
熔断器状态	实时并发度	十秒内成功数	十秒内失败数	十秒内超时数	最近连续错误数	错误率
CLOSED	88	2331	0	0	0	0.0000%
CLOSED	88	2033	0	0	0	0.0000%
CLOSED	64	2056	0	0	0	0.0000%
CLOSED	70	1997	0	0	0	0.0000%
CLOSED	70	1996	0	0	0	0.0000%
CLOSED	76	2051	0	0	0	0.0000%

并发超时控制

关键点

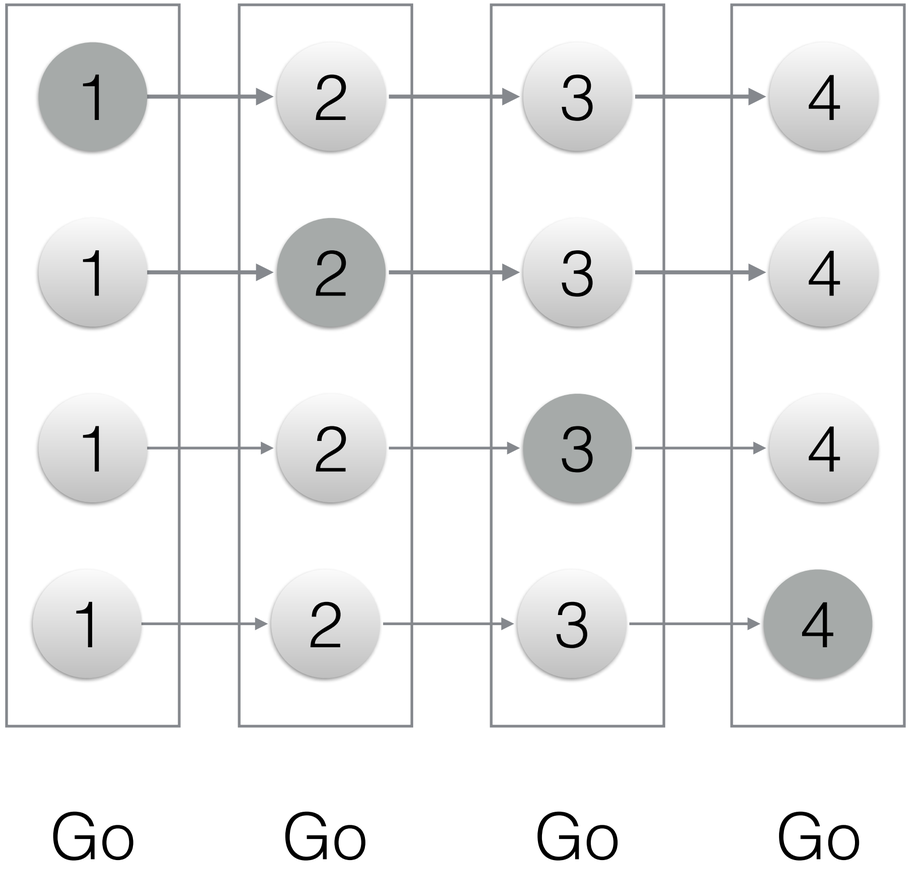
- **精确控制服务响应时间**
- **严格控制并发协程数量**

流水线处理



① 消费消息队列
② 查询数据库
③ 逻辑计算
④ 写数据库

流水线处理



① 消费消息队列
② 查询数据库
③ 逻辑计算
④ 写数据库

并发总结

- 并发提高效率
- 协程需要控制
- 精确控制超时
- 流水线处理模式

性能

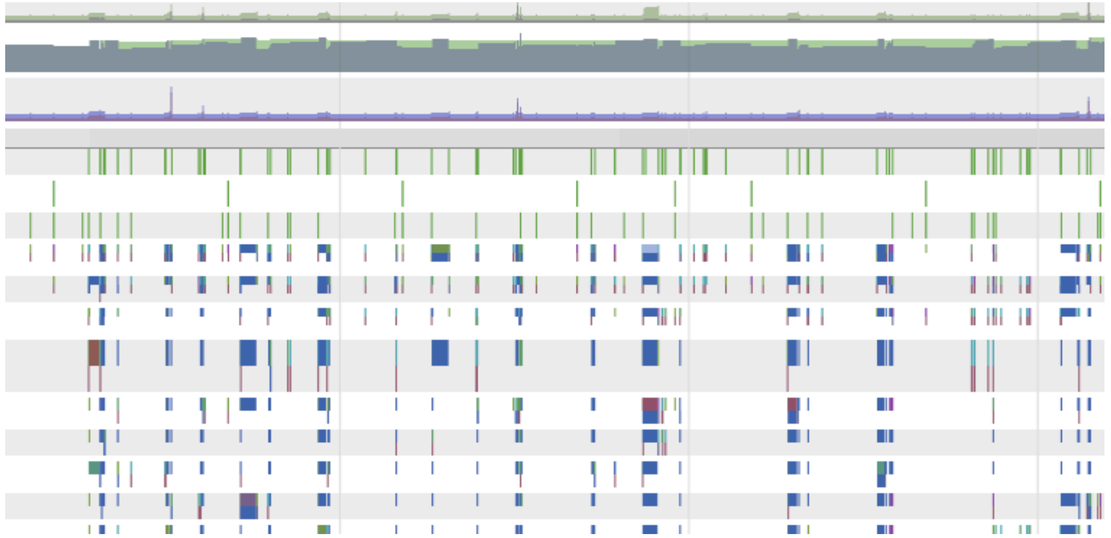
性能分析方法

- pprof工具
- 查看协程栈
- GC日志
- Trace分析

```
(pprof) top
1.11GB of 1.12GB total (99.69%)
Dropped 340 nodes (cum <= 0.01GB)
      flat  flat%   sum%        cum   cum%
    1.11GB 99.69% 99.69%     1.11GB 99.69%  code.byted.org/video/urls/vendor/code.byted.org/gopkg/t
         0     0% 99.69%     1.11GB 99.73%  code.byted.org/video/urls/thrift_gen/toutiao/video/urls
         0     0% 99.69%     1.11GB 99.69%  code.byted.org/video/urls/vendor/code.byted.org/gopkg/t
         0     0% 99.69%     1.11GB 99.73%  code.byted.org/video/urls/vendor/code.byted.org/kite/ki
         0     0% 99.69%     1.11GB 99.78%  code.byted.org/video/urls/vendor/code.byted.org/kite/ki
         0     0% 99.69%     1.12GB 99.91%  runtime.goexit
(pprof) list readStringBody
Total: 1.12GB
ROUTINE ======================== code.byted.org/video/urls/vendor/code.byted.org/gopkg/thrift.(*TBi
    1.11GB     1.11GB (flat, cum) 99.69% of Total
         .          .    475:   }
         .          .    476:   var buf []byte
         .          .    477:   if size <= len(p.buffer) {
         .          .    478:           buf = p.buffer[0:size]
         .          .    479:   } else {
    1.11GB     1.11GB    480:           buf = make([]byte, size)
         .          .    481:   }
         .          .    482:   _, e := io.ReadFull(p.trans, buf)
         .          .    483:   return string(buf), NewTProtocolException(e)
         .          .    484:}
```

```
gc 8 @1.183s 0%: 0.098+1.3+0.27 ms clock, 2.8+0/8.0/10+8.0 ms cpu, 1
gc 9 @1.403s 0%: 0.074+2.2+0.34 ms clock, 2.3+0.043/8.6/12+10 ms cpu
gc 10 @3.587s 0%: 0.022+1.6+0.31 ms clock, 0.72+0/7.5/15+10 ms cpu,
gc 11 @6.199s 0%: 0.021+2.8+0.28 ms clock, 0.67+0/10/12+9.2 ms cpu,
gc 12 @8.703s 0%: 0.017+1.9+0.27 ms clock, 0.54+0/7.2/15+8.6 ms cpu,
gc 13 @11.235s 0%: 0.018+2.1+0.29 ms clock, 0.59+0/8.0/17+9.5 ms cpu
gc 14 @13.754s 0%: 0.022+2.1+0.30 ms clock, 0.71+0/11/13+9.7 ms cpu,
gc 15 @16.160s 0%: 0.054+1.8+1.0 ms clock, 1.7+0.083/16/11+33 ms cpu
gc 16 @18.458s 0%: 0.018+2.8+0.29 ms clock, 0.58+0.24/22/3.2+9.4 ms
gc 17 @20.664s 0%: 0.025+1.7+0.33 ms clock, 0.81+0/8.8/18+10 ms cpu,
gc 18 @22.115s 0%: 0.042+1.8+0.32 ms clock, 1.3+0.16/13/12+10 ms cpu
gc 19 @23.604s 0%: 0.020+1.8+0.32 ms clock, 0.65+0.27/15/14+10 ms cp
gc 20 @25.080s 0%: 0.017+3.0+0.37 ms clock, 0.57+0/11/13+11 ms cpu,
gc 21 @26.263s 0%: 0.020+2.0+0.35 ms clock, 0.64+0.056/12/12+11 ms c
gc 22 @27.304s 0%: 0.019+1.8+0.36 ms clock, 0.63+0.015/10/19+11 ms c
gc 23 @28.391s 0%: 0.067+1.9+0.37 ms clock, 1.9+0.76/15/13+10 ms cpu
gc 24 @29.473s 0%: 0.029+2.1+0.30 ms clock, 0.88+1.9/17/4.6+9.1 ms c
gc 25 @30.557s 0%: 0.019+2.0+0.31 ms clock, 0.60+2.2/17/5.9+10 ms cp
```

```
goroutine profile: total 317
1 @ 0x8afa98 0x8af873 0x8ab164 0x719d2e 0x719f40 0x5eb2ba 0x5ecb2d 0x5ed5ae 0x5e
#       0x8afa98        runtime/pprof.writeRuntimeProfile+0xb8   /data04/home/xia
#       0x8af873        runtime/pprof.writeGoroutine+0x93        /data04/home/xia
#       0x8ab164        runtime/pprof.(*Profile).WriteTo+0xd4    /data04/home/xia
#       0x719d2e        net/http/pprof.handler.ServeHTTP+0x37e   /data04/home/xia
#       0x719f40        net/http/pprof.Index+0x200               /data04/home/xia
#       0x5eb2ba        net/http.HandlerFunc.ServeHTTP+0x3a      /data04/home/xia
#       0x5ecb2d        net/http.(*ServeMux).ServeHTTP+0x17d     /data04/home/xia
#       0x5ed5ae        net/http.serverHandler.ServeHTTP+0x19e   /data04/home/xia
#       0x5ea01e        net/http.(*conn).serve+0xf2e             /data04/home/xia

1 @ 0x430ff3 0x42b99e 0x42ae60 0x6b7a6a 0x6b7ad6 0x6bb76c 0x6d95ed 0x6d982d 0x5e
#       0x42ae60        net.runtime_pollWait+0x60                /data04/home/xia
#       0x6b7a6a        net.(*pollDesc).Wait+0x3a                /data04/home/xia
#       0x6b7ad6        net.(*pollDesc).WaitRead+0x36            /data04/home/xia
#       0x6bb76c        net.(*netFD).accept+0x27c                /data04/home/xia
#       0x6d95ed        net.(*TCPListener).AcceptTCP+0x4d        /data04/home/xia
#       0x6d982d        net.(*TCPListener).Accept+0x3d           /data04/home/xia
```



性能优化

锁粒度

- 锁变量而不要锁过程
- 使用CAS
- 热点代码重点优化

GC

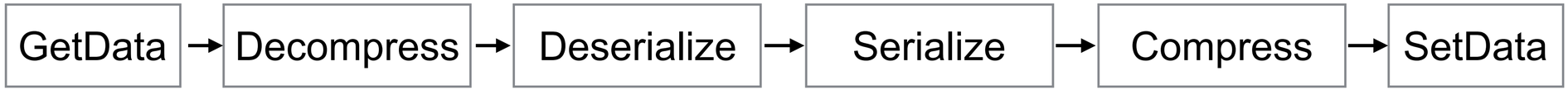
- 对象复用
- 避免反射
- GOGC

性能优化实践

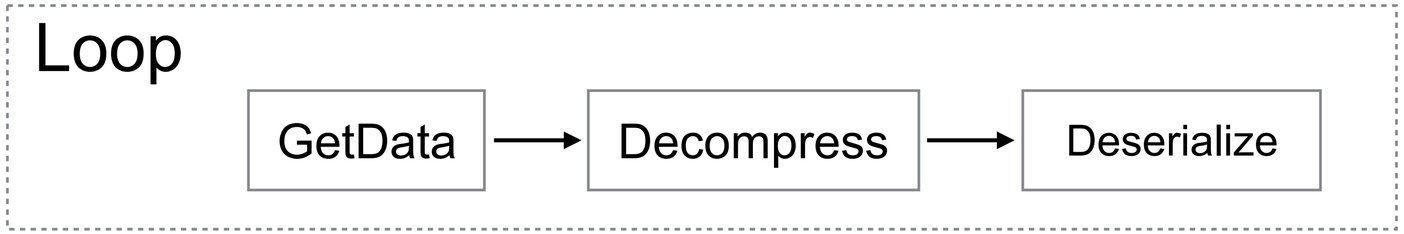
- 服务功能
  - 信息流历史存储服务
- **问题**
  - 高峰延迟PCT99高
- 服务接口列表
  - SetHistory
  - GetHistoryByRange

性能分析实践

- SetHistory



- GetHistoryByRange



性能分析

问题点

- GC
- Deserialize

优化思路

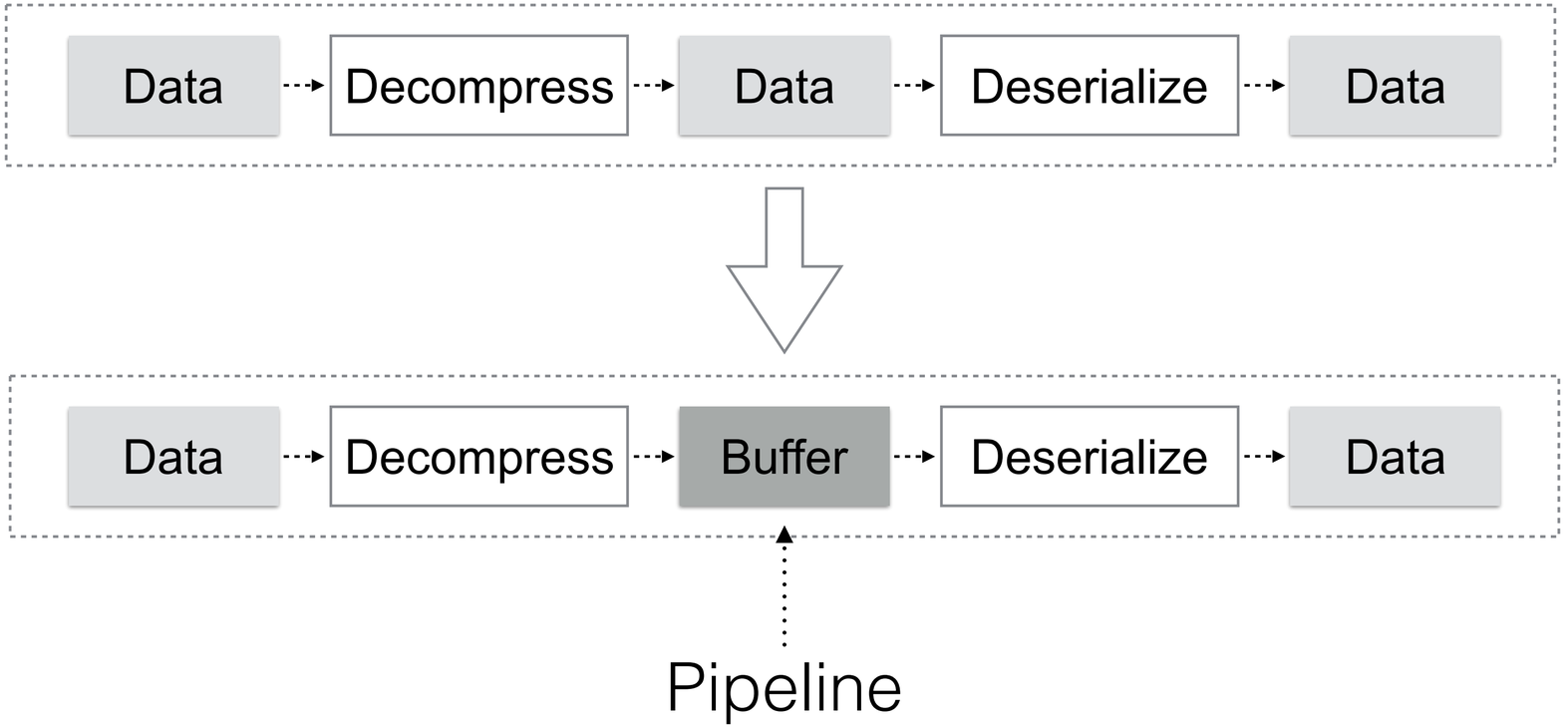
1. 降低内存使用
2. 减少对象申请
3. 优化序列化方式

Pipeline

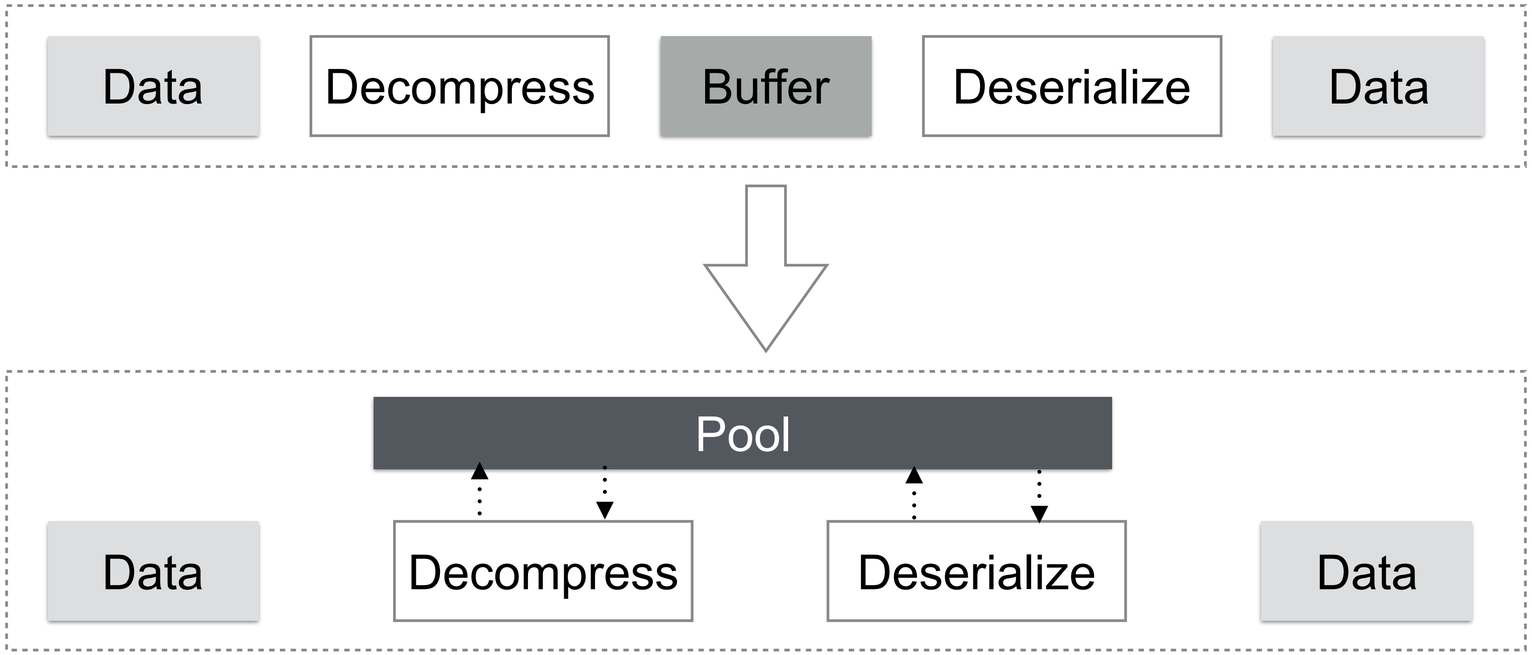
```
type Reader interface {
        Read(p []byte) (n int, err error)
}
```

```
type Writer interface {
        Write(p []byte) (n int, err error)
}
```

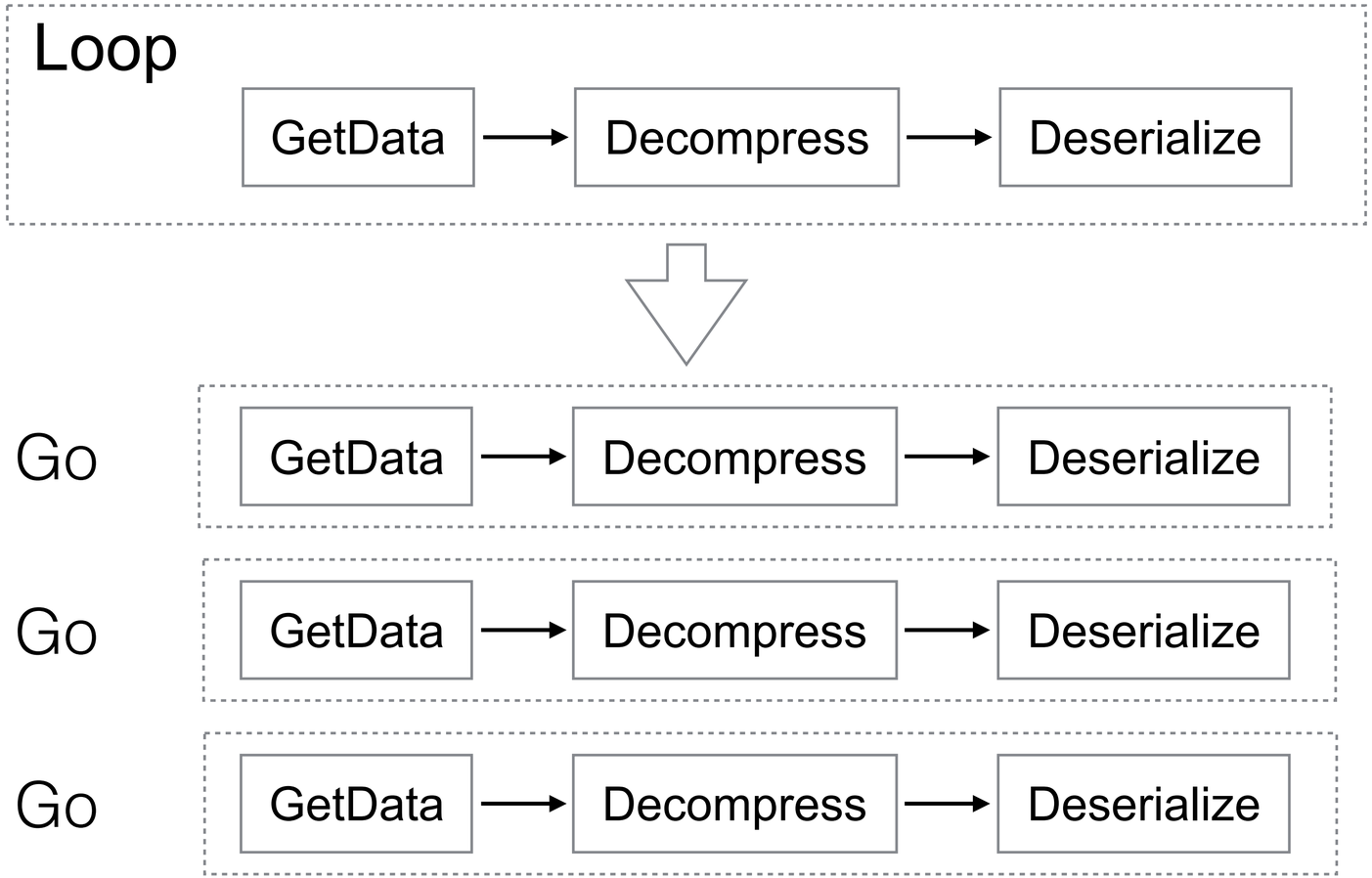
减少内存


Data
Decompress
Data
Deserialize
Data
Data
Decompress
Buffer
Deserialize
Data
Pipeline

减少对象


Data
Decompress
Buffer
Deserialize
Data
Pool
Data
Decompress
Deserialize
Data

提高并发度


Loop
GetData
Decompress
Deserialize
Go
GetData
Decompress
Deserialize
Go
GetData
Decompress
Deserialize
Go
GetData
Decompress
Deserialize

性能优化

- 提高并发
- 对象复用

监控

监控

- 协程数量和状态
- GC停顿时间，GC频率
- 堆栈内存使用
- 快照运行状态

编程思维

编程思维

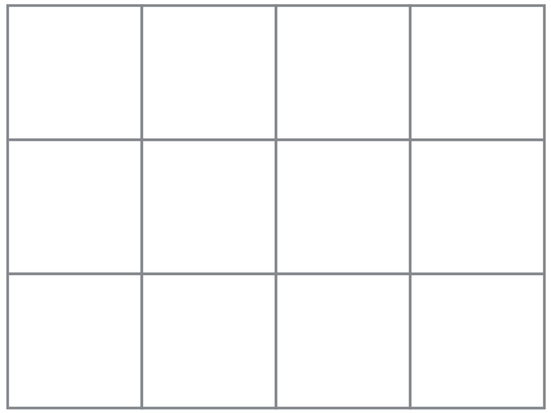
- 独立进程，没有Thread local
- 需要考虑Panic，Recover
- 并发是常态，共享资源保护

工程性

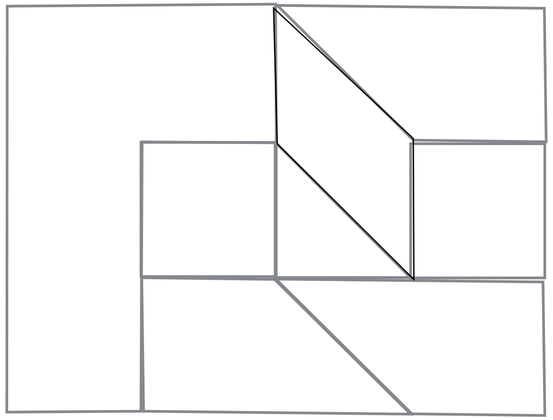
工程性

- 语言使用的效率和语言本身的可管理性的权衡

工程性



Golang



Python

工程性

借助Go的AST包实现自定义检测代码

```
import (
    "go/parser"
    "go/token"
)

fset := token.NewFileSet() // positions are relative to fset
f, err := parser.ParseFile(fset, “example.go”, nil, parser.AllErrors)
```

总结

- 并发-可控
- 性能-持续关注
- 监控-尽量全面
- 编程思维-转变
- 工程性-无限想象

Thanks !