Evolutionary Optimization

Make it work — Make it right — Make it fast

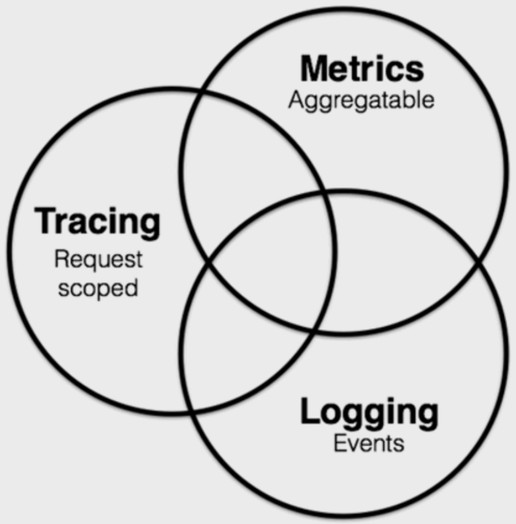

fastly

Metrics, tracing, and logging

2017 02 21

Today I had the good fortune of attending the 2017 Distributed Tracing Summit, with lots of rad folks from orgs like AWS/X-Ray, OpenZipkin, OpenTracing, Instana, Datadog, Librato, and many others I regret that I'm forgetting. At one point the discussion took a turn toward project scope and definitions. Should a tracing system also manage logging? What indeed *is* logging, when viewed through the different lenses represented in the room? And where do all of the various concrete systems fit in to the picture?

In short, I felt that we were stumbling a little bit around a shared vocabulary. I thought we could probably map out the domain of instrumentation, or observability, as a sort of Venn diagram. Metrics, tracing, and logging are definitely all parts of a broader picture, and can definitely overlap in some circumstances, but I wanted to try and identify the properties of each that were truly distinct. I had a think over a coffee break and came up with this.

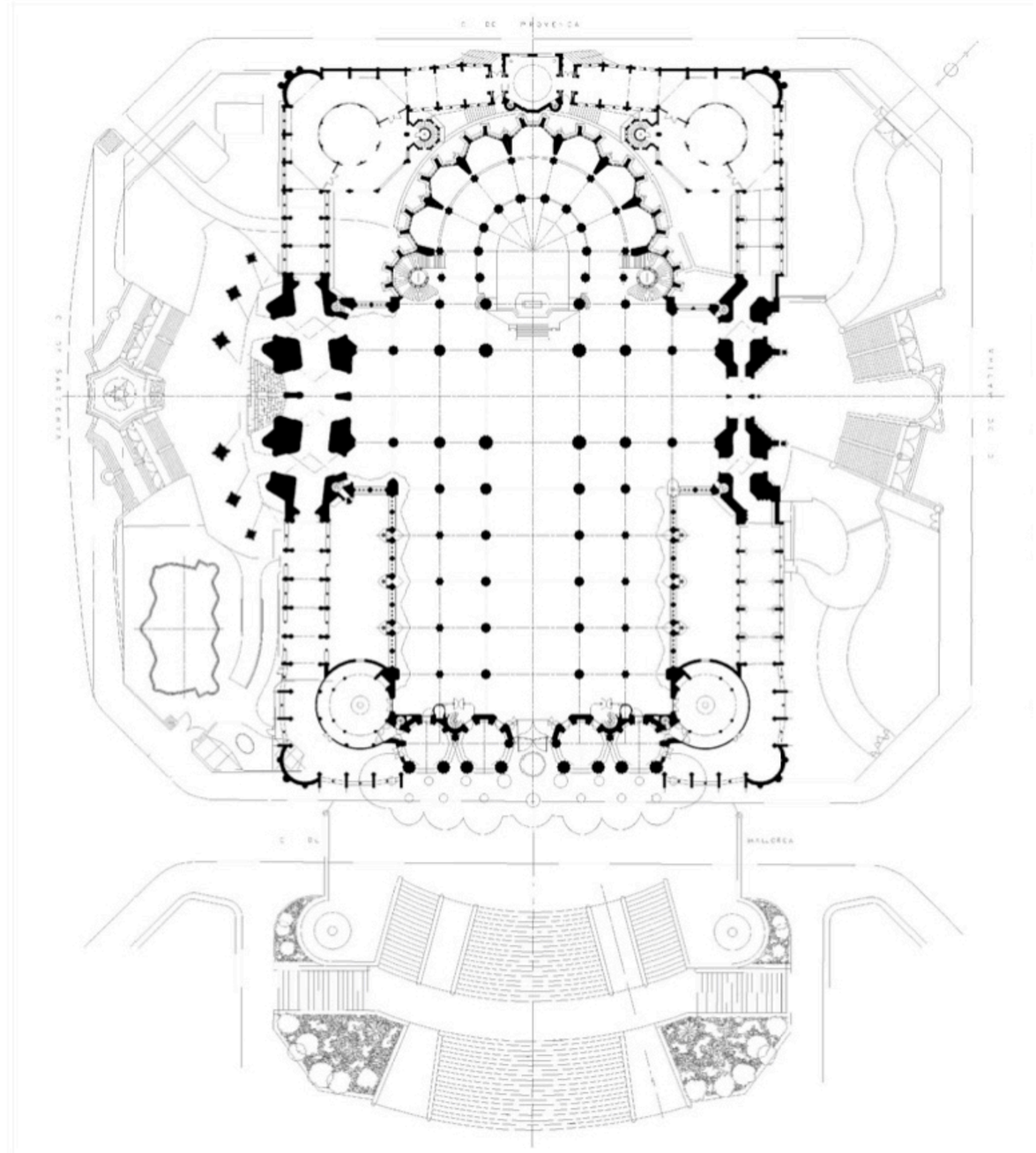


A mediocre white man presents

Building OK Log

Design

Make it work



“**Weeks** of programming can save you **hours** of planning.”

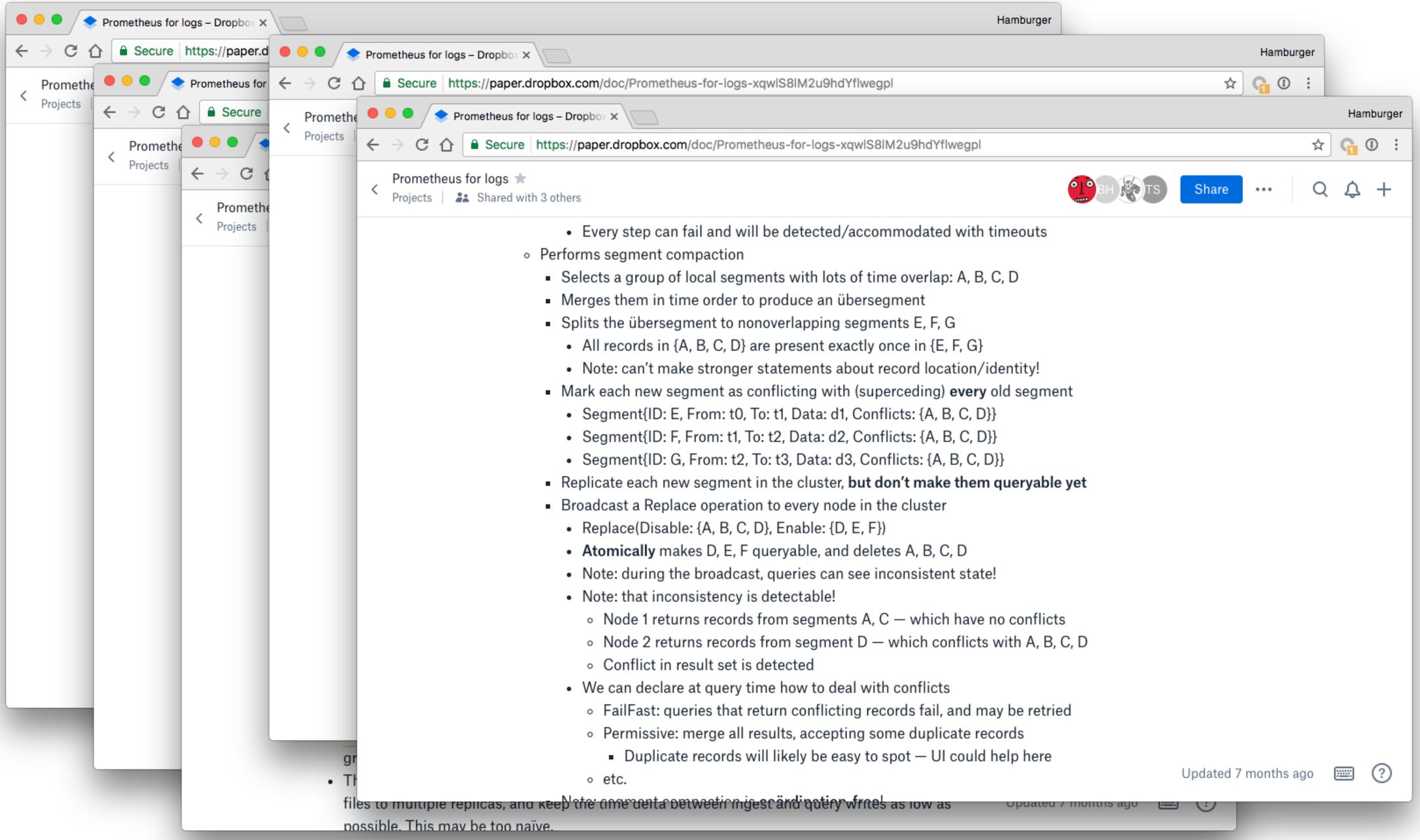
—A Hackernews

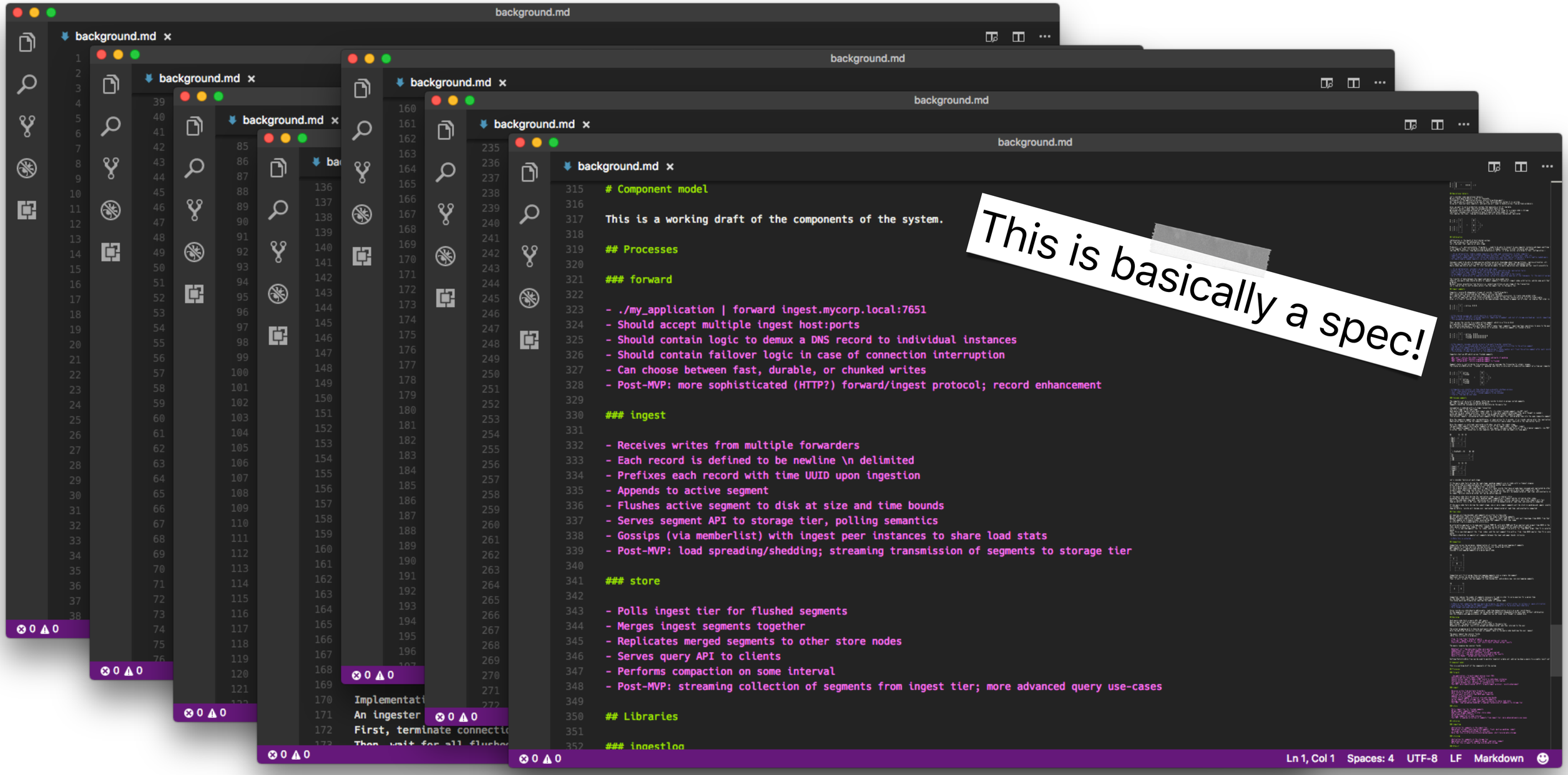
“Programming is made up of 2 activities:
making decisions (95%) and **typing (5%)**”

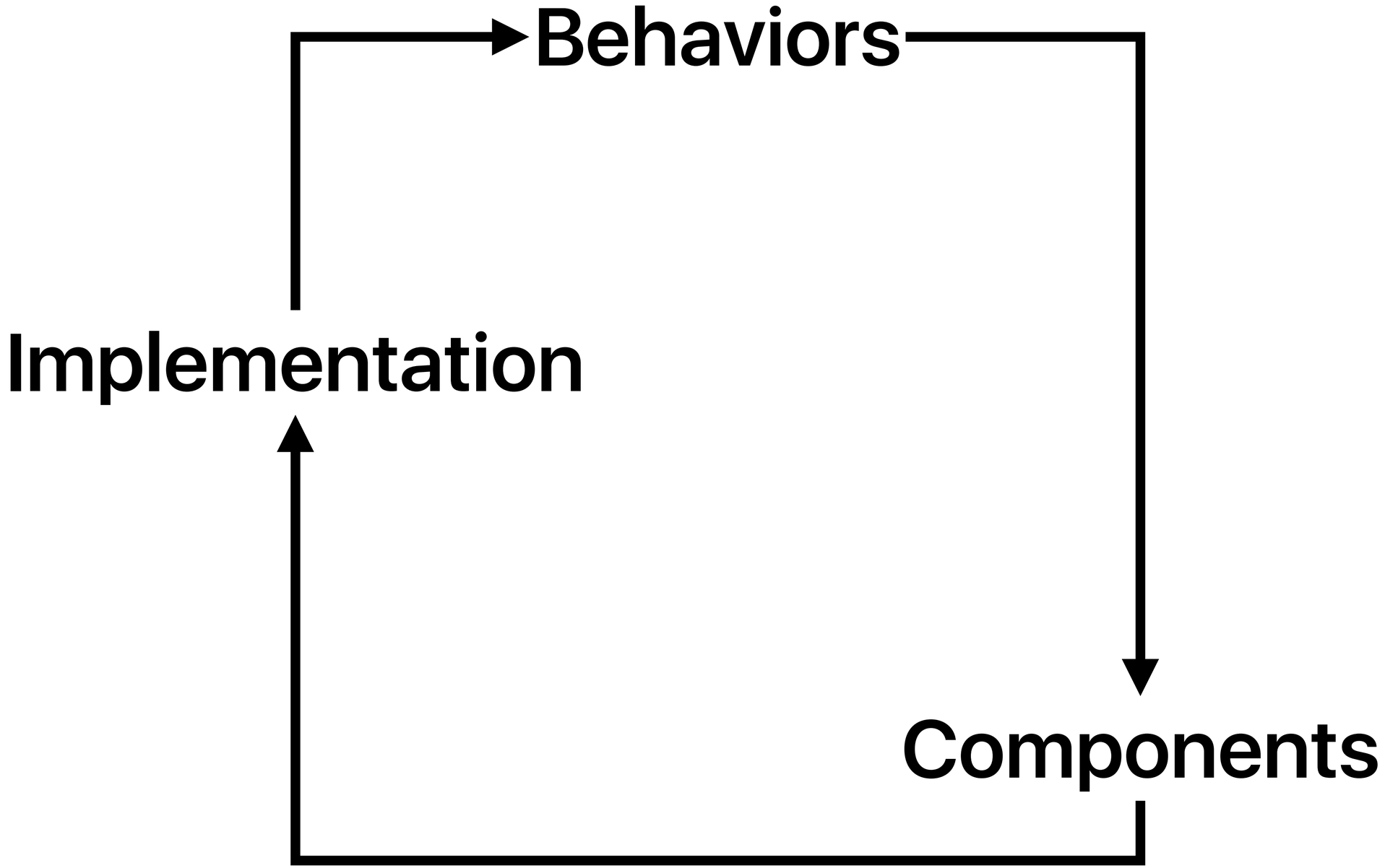
josephg.com/blog/rplace-in-a-weekend

1. Make decisions
2. Type

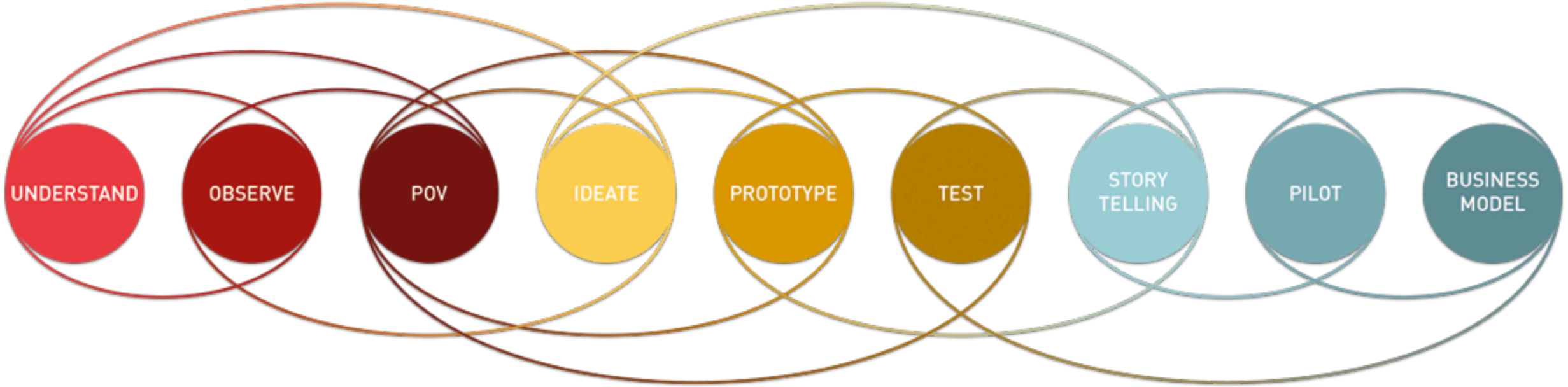
1. ~~Make decisions~~ **Design**
2. ~~Type~~ **Build**

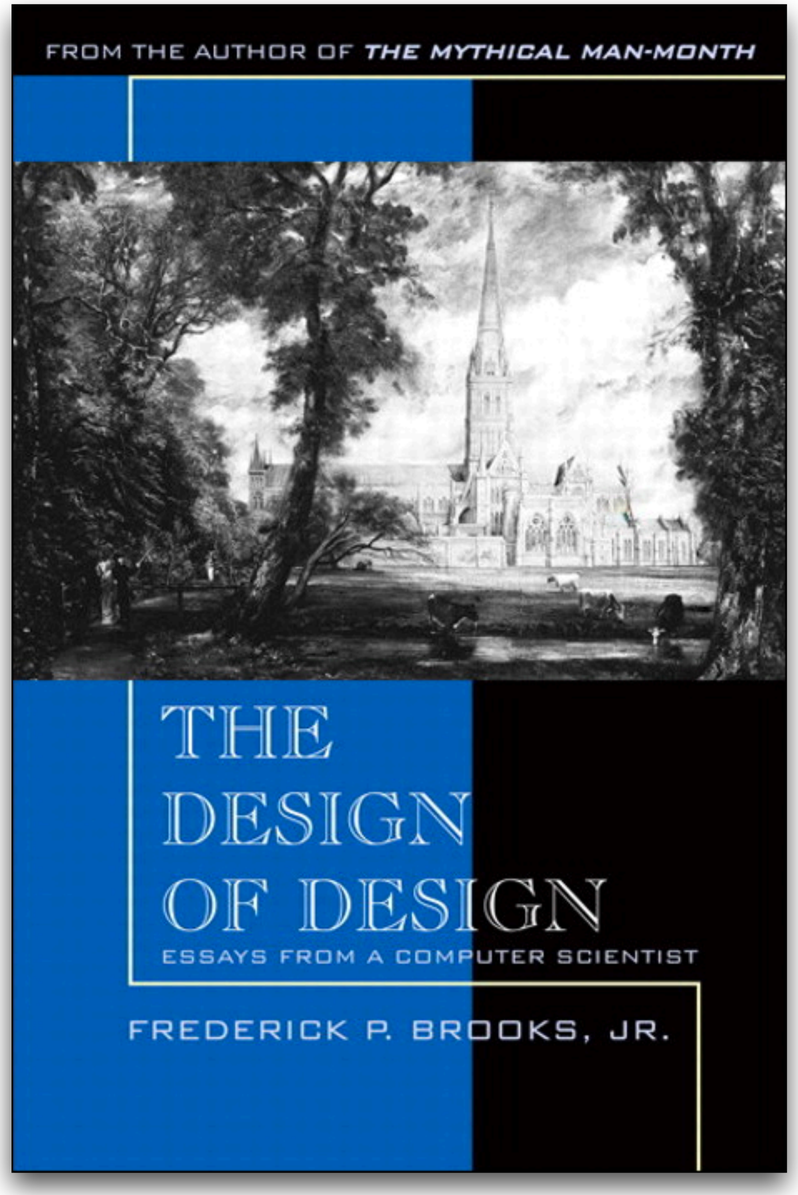

Prometheus for logs – Dropbox
Hamburger
Secure
https://paper.dropbox.com/doc/Prometheus-for-logs-xqwlS8lM2u9hdYflwegpl
Prometheus for logs
Projects
Shared with 3 others
Share
Every step can fail and will be detected/accommodated with timeouts
Performs segment compaction
Selects a group of local segments with lots of time overlap: A, B, C, D
Merges them in time order to produce an übersegment
Splits the übersegment to nonoverlapping segments E, F, G
All records in {A, B, C, D} are present exactly once in {E, F, G}
Note: can't make stronger statements about record location/identity!
Mark each new segment as conflicting with (superceding) every old segment
Segment{ID: E, From: t0, To: t1, Data: d1, Conflicts: {A, B, C, D}}
Segment{ID: F, From: t1, To: t2, Data: d2, Conflicts: {A, B, C, D}}
Segment{ID: G, From: t2, To: t3, Data: d3, Conflicts: {A, B, C, D}}
Replicate each new segment in the cluster, but don't make them queryable yet
Broadcast a Replace operation to every node in the cluster
Replace(Disable: {A, B, C, D}, Enable: {D, E, F})
Atomically makes D, E, F queryable, and deletes A, B, C, D
Note: during the broadcast, queries can see inconsistent state!
Note: that inconsistency is detectable!
Node 1 returns records from segments A, C — which have no conflicts
Node 2 returns records from segment D — which conflicts with A, B, C, D
Conflict in result set is detected
We can declare at query time how to deal with conflicts
FailFast: queries that return conflicting records fail, and may be retried
Permissive: merge all results, accepting some duplicate records
Duplicate records will likely be easy to spot — UI could help here
etc.
Updated 7 months ago


background.md
Component model
This is a working draft of the components of the system.
Processes
forward
- ./my_application | forward ingest.mycorp.local:7651
- Should accept multiple ingest host:ports
- Should contain logic to demux a DNS record to individual instances
- Should contain failover logic in case of connection interruption
- Can choose between fast, durable, or chunked writes
- Post-MVP: more sophisticated (HTTP?) forward/ingest protocol; record enhancement
ingest
- Receives writes from multiple forwarders
- Each record is defined to be newline \n delimited
- Prefixes each record with time UUID upon ingestion
- Appends to active segment
- Flushes active segment to disk at size and time bounds
- Serves segment API to storage tier, polling semantics
- Gossips (via memberlist) with ingest peer instances to share load stats
- Post-MVP: load spreading/shedding; streaming transmission of segments to storage tier
store
- Polls ingest tier for flushed segments
- Merges ingest segments together
- Replicates merged segments to other store nodes
- Serves query API to clients
- Performs compaction on some interval
- Post-MVP: streaming collection of segments from ingest tier; more advanced query use-cases
Libraries
ingestlog
This is basically a spec!
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Markdown


Behaviors
Implementation
Components

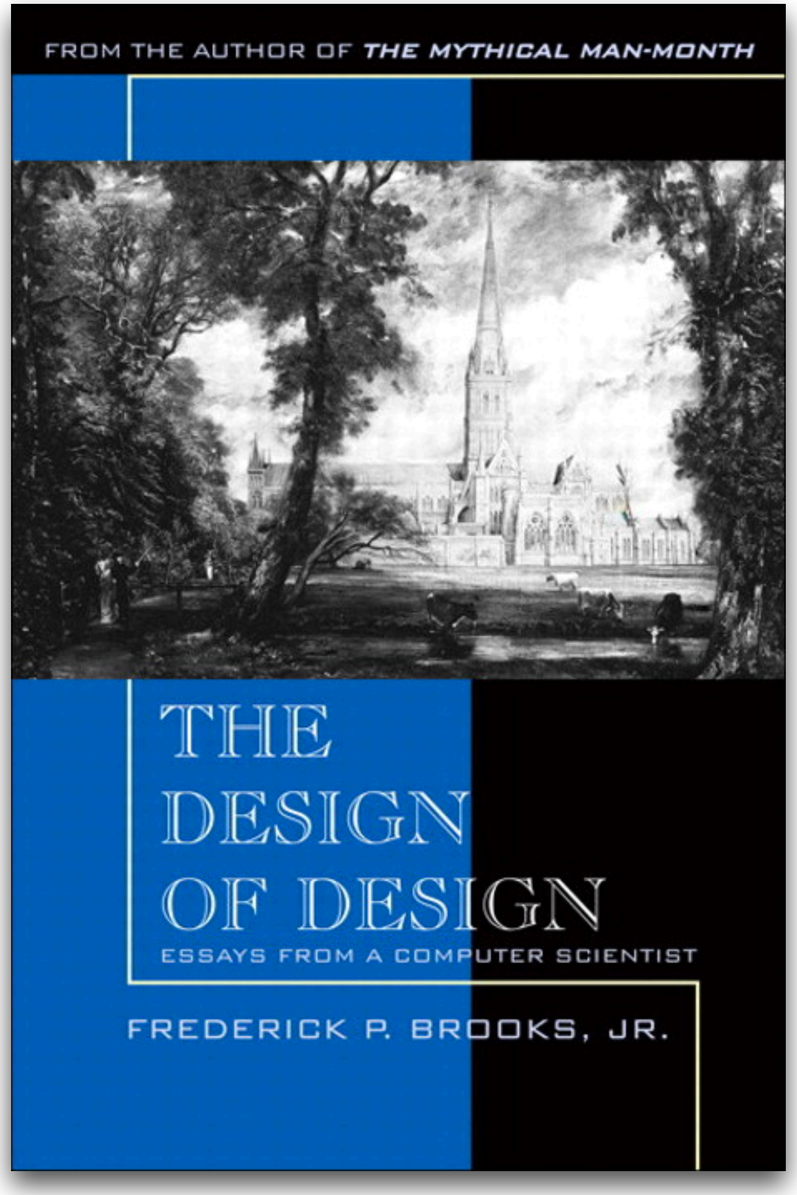
THE DESIGN THINKING PROCESS



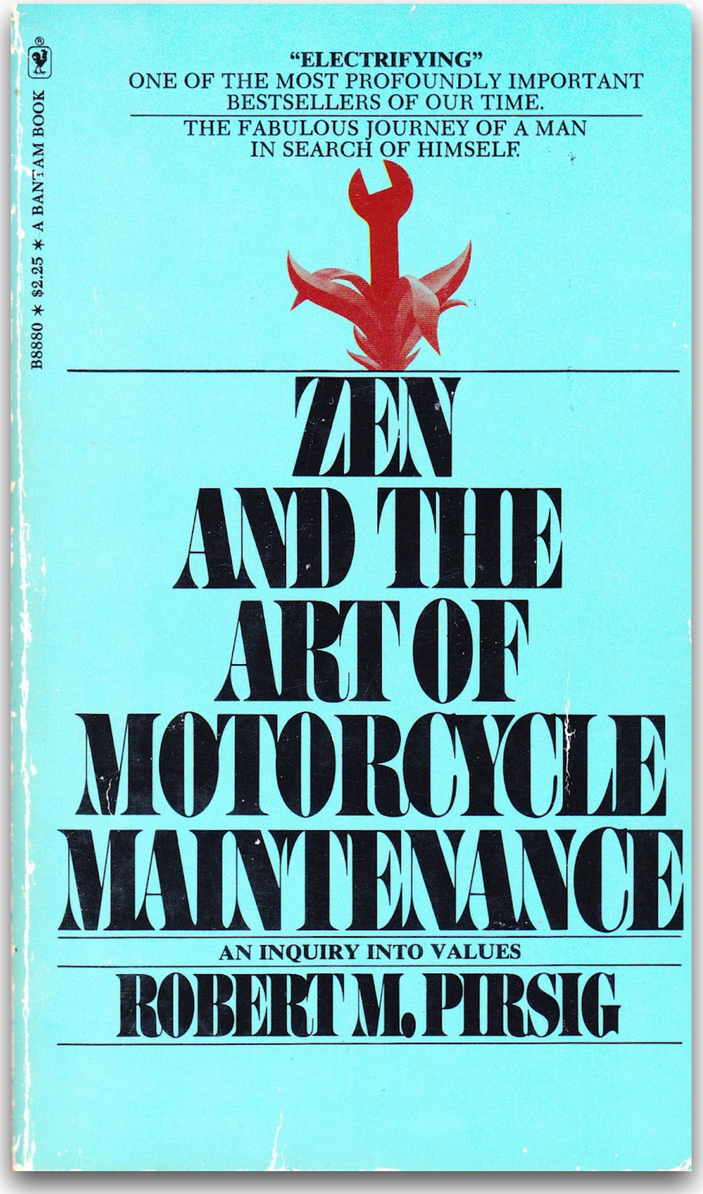


"The whole design problem [is] an **intricately interlocked interplay** of factors..."

"In practice, designing seems to proceed by **oscillating between sub-solution and sub-problem areas**..."



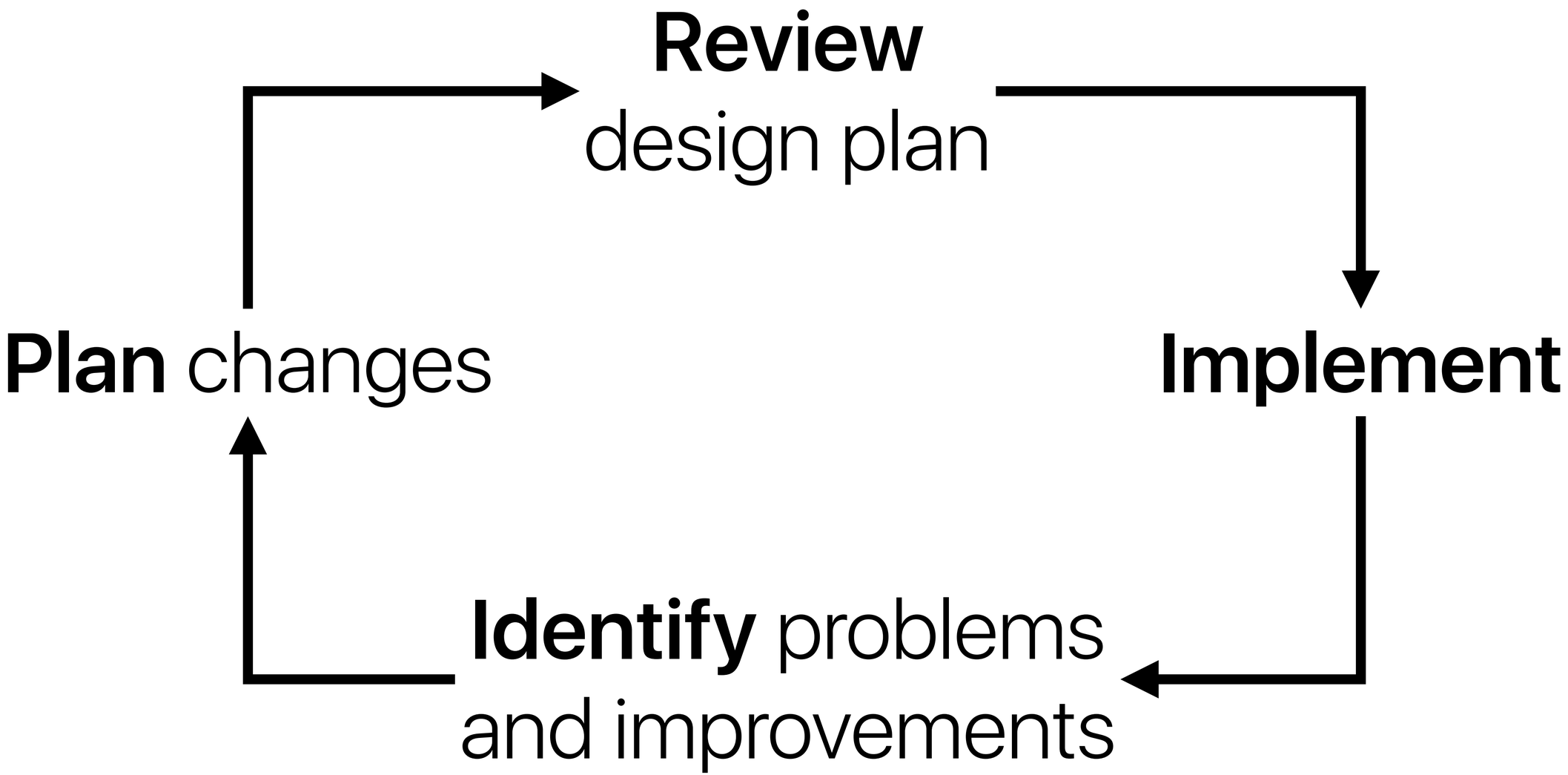

```
LET utility function = ...

UNTIL good enough OR time runs out
  DO another design, improving utility
    UNTIL design is complete
      WHILE design remains feasible
        Make another design decision
      END
      Backtrack up design tree
      Explore a path not searched before
    END
  END
  Take best design
END
```

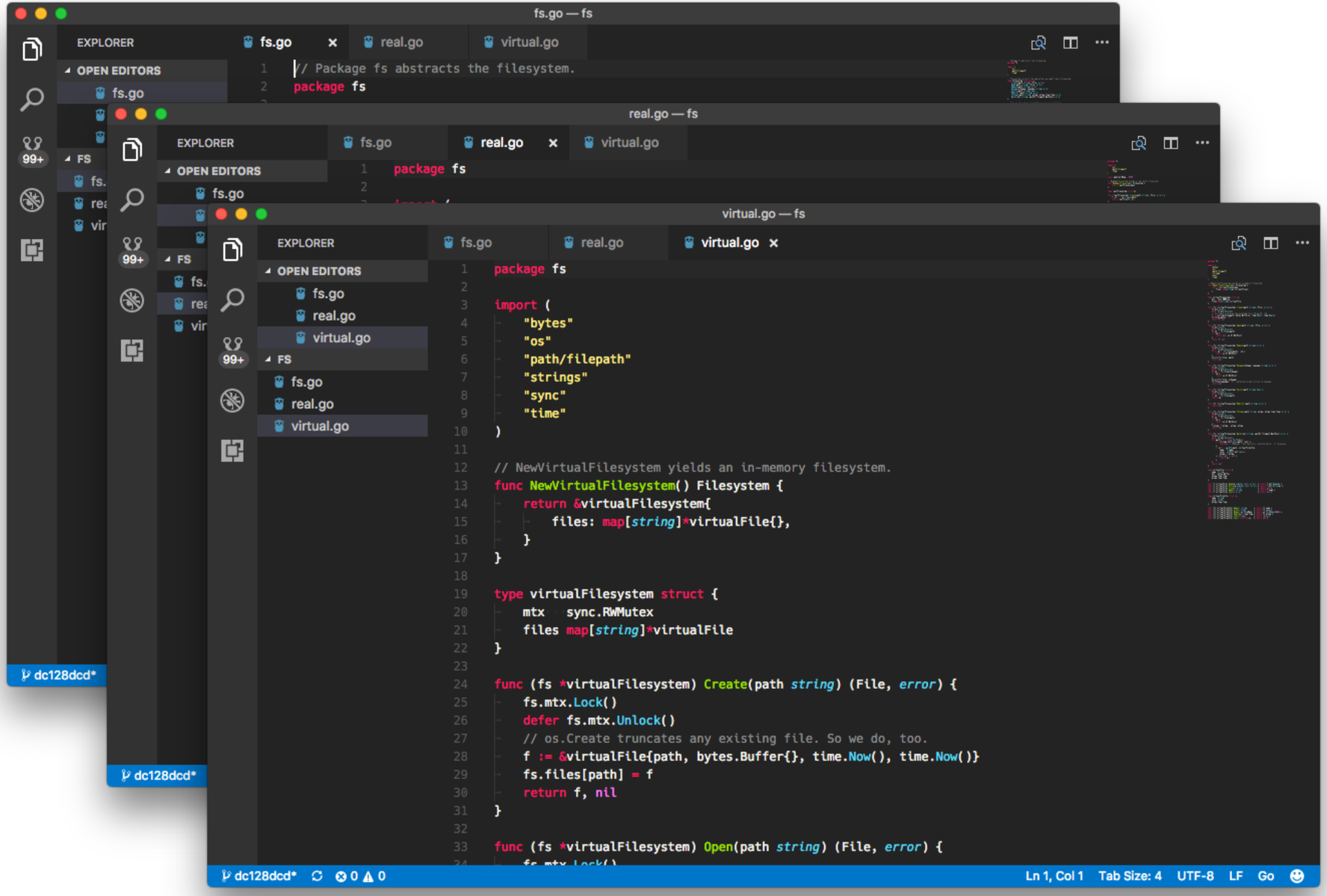


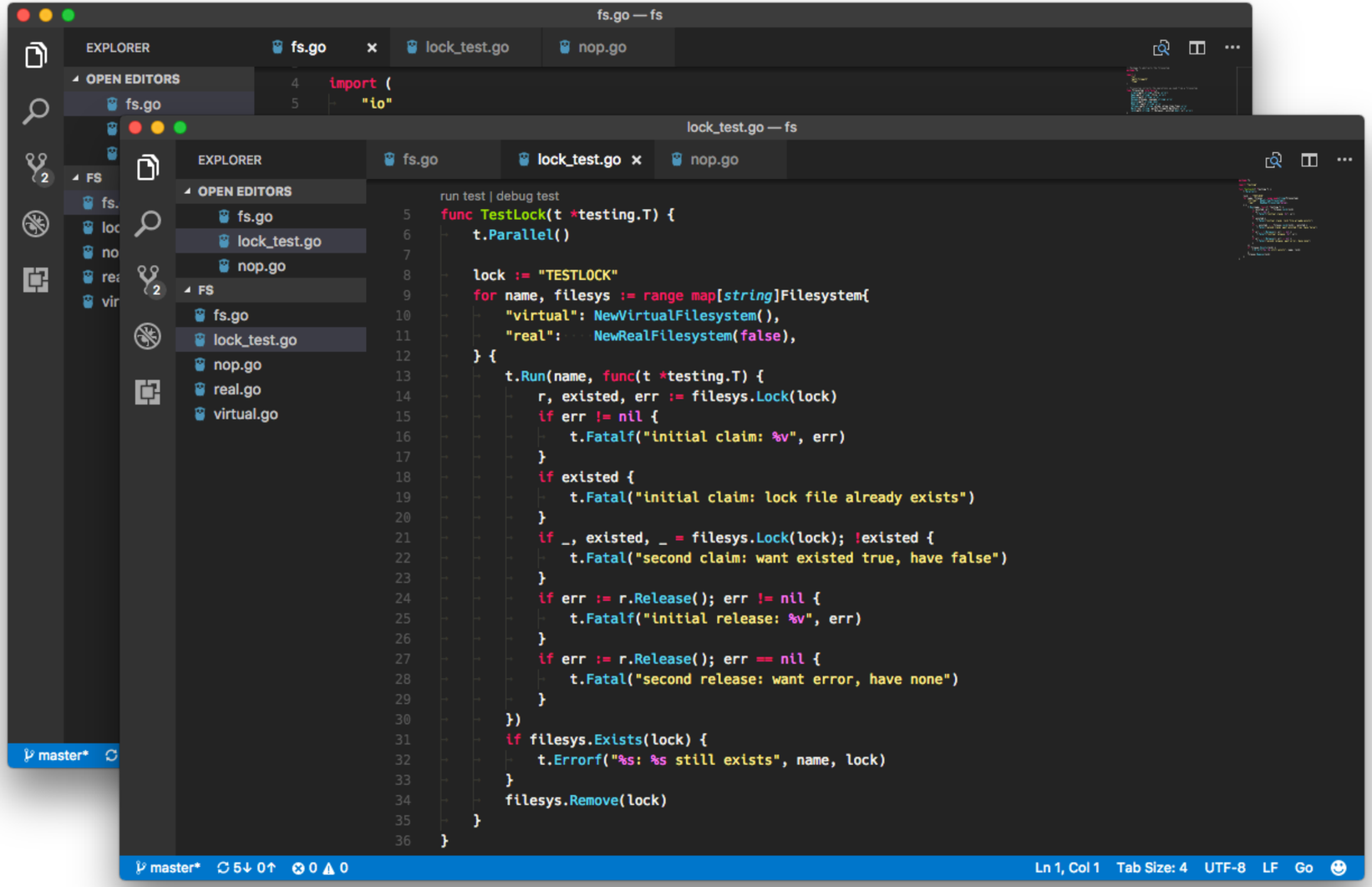
"Phædrus began to pursue ... lateral truths; no longer the frontal truths of science, those toward which the discipline pointed, but the kind of truth you see laterally, out of the corner of your eye. In a laboratory situation, **when your whole procedure goes haywire**, when everything goes wrong or is indeterminate or is so screwed up by unexpected results you can't make head or tail out of anything, **you start looking laterally.** That's a word he later used to describe a growth of knowledge that doesn't move forward like an arrow in flight, but expands sideways, like an arrow enlarging in flight, or like the archer, discovering that although he has hit the bull's eye and won the prize, his head is on a pillow and the sun is coming in the window."

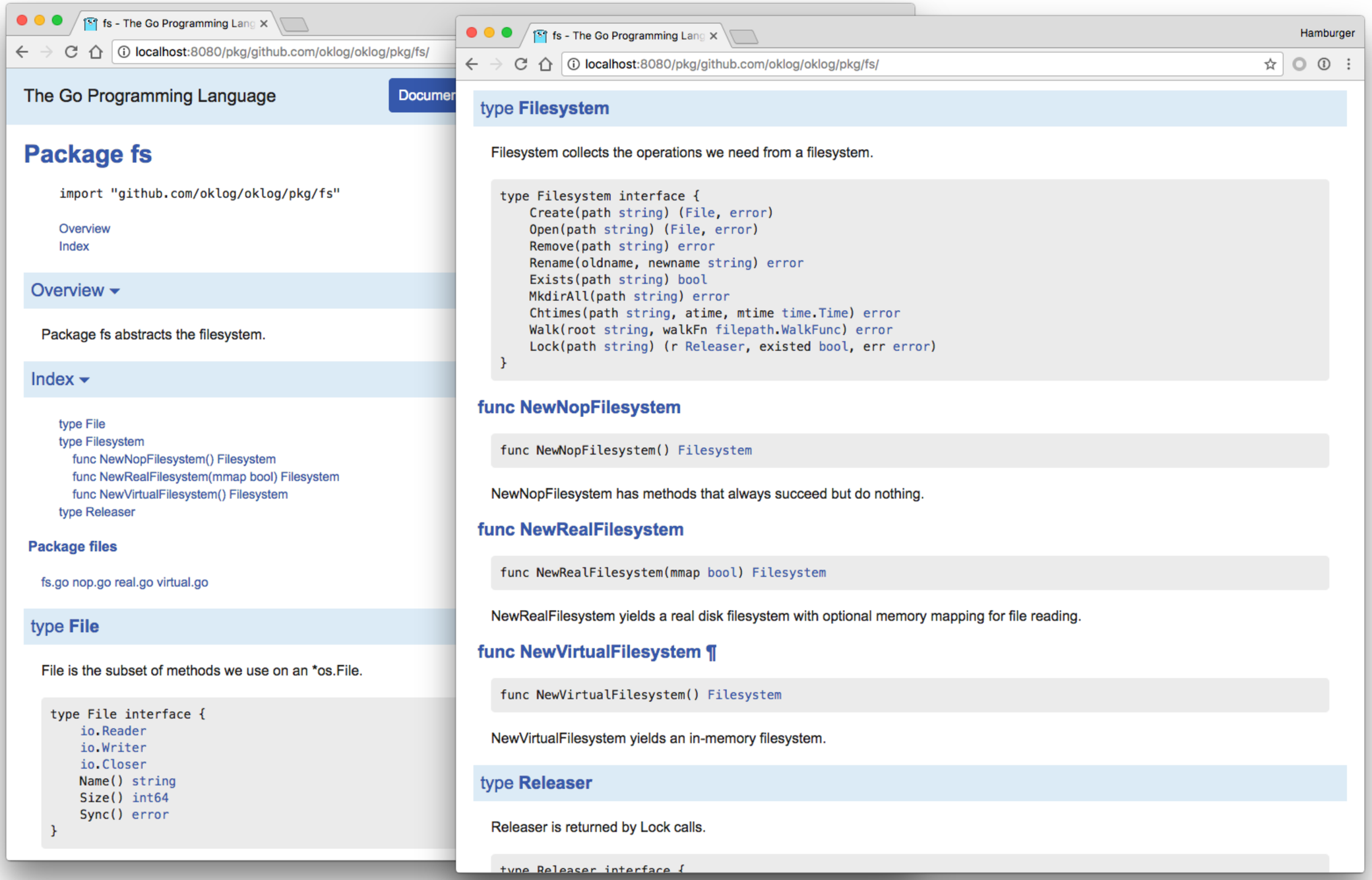
Build

Make it right


Review
design plan
Implement
Identify problems
and improvements
Plan changes

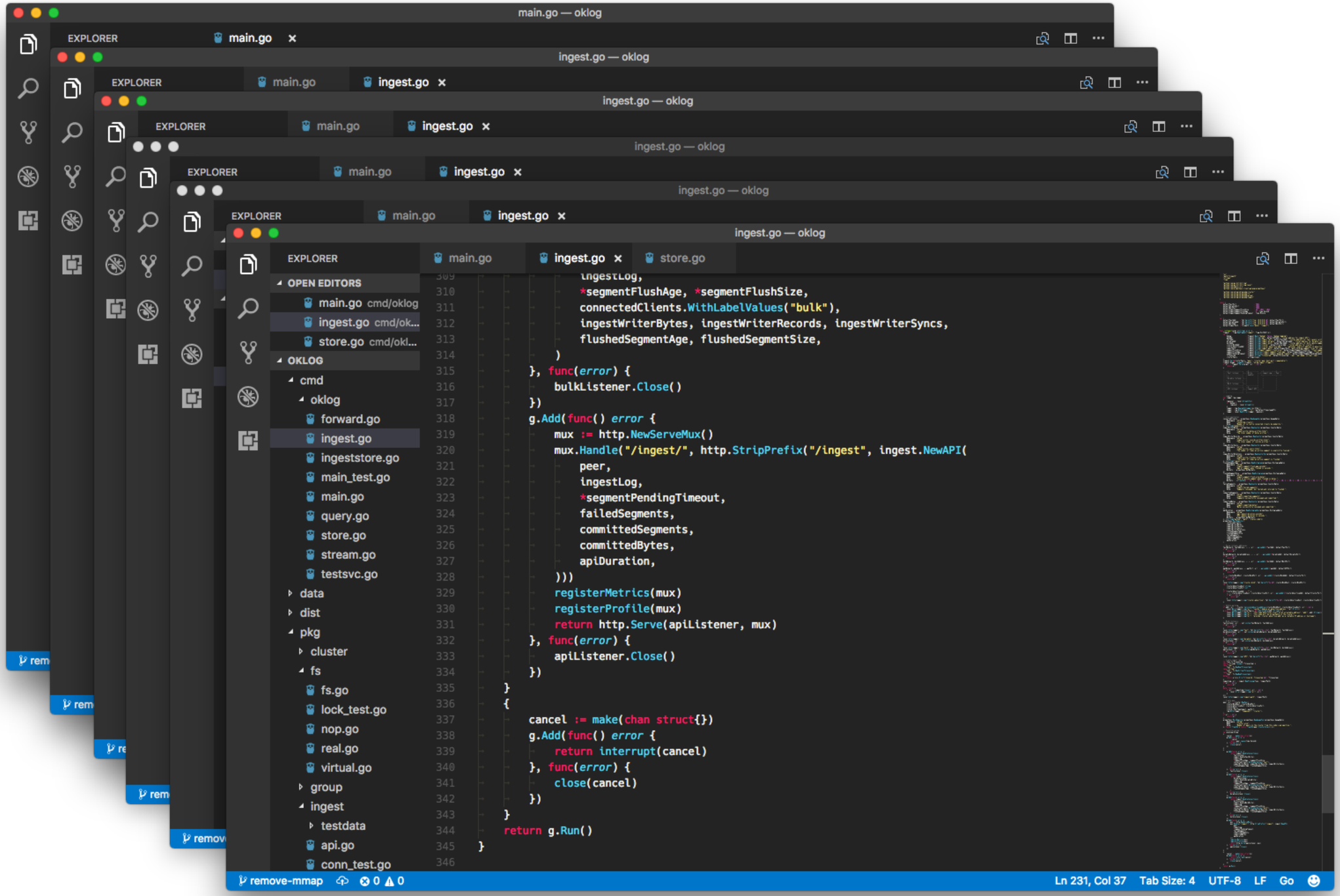

fs.go — fs
real.go — fs
virtual.go — fs
EXPLORER
OPEN EDITORS
FS
fs.go
real.go
virtual.go
// Package fs abstracts the filesystem.
package fs
import (
"bytes"
"os"
"path/filepath"
"strings"
"sync"
"time"
)
// NewVirtualFilesystem yields an in-memory filesystem.
func NewVirtualFilesystem() Filesystem {
return &virtualFilesystem{
files: map[string]*virtualFile{},
}
}
type virtualFilesystem struct {
mtx sync.RWMutex
files map[string]*virtualFile
}
func (fs *virtualFilesystem) Create(path string) (File, error) {
fs.mtx.Lock()
defer fs.mtx.Unlock()
// os.Create truncates any existing file. So we do, too.
f := &virtualFile{path, bytes.Buffer{}, time.Now(), time.Now()}
fs.files[path] = f
return f, nil
}
func (fs *virtualFilesystem) Open(path string) (File, error) {
dc128dcd*
Ln 1, Col 1
Tab Size: 4
UTF-8
LF
Go

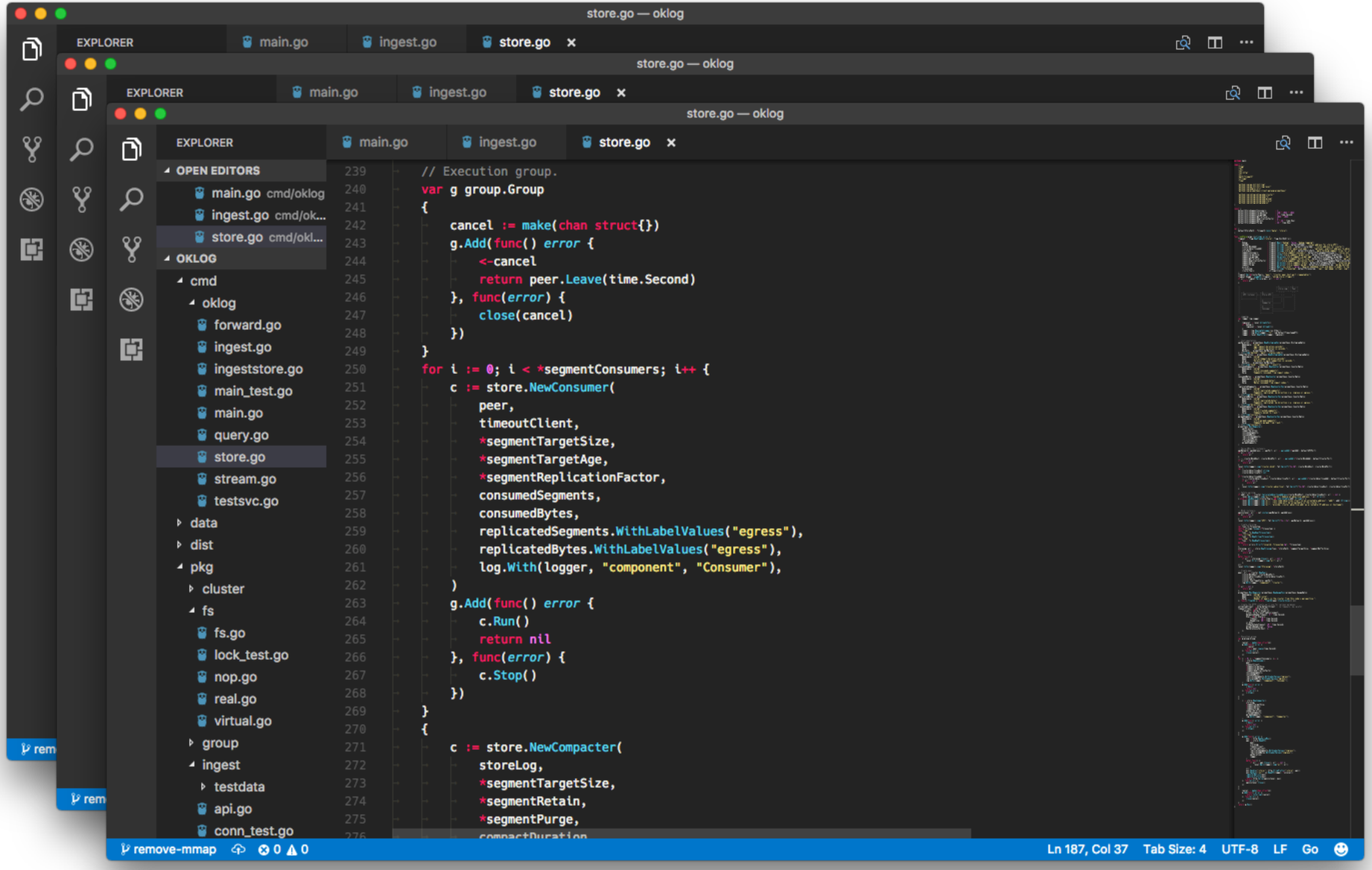

fs.go — fs
EXPLORER
fs.go
lock_test.go
nop.go
OPEN EDITORS
FS
import (
"io"
lock_test.go — fs
EXPLORER
fs.go
lock_test.go
nop.go
OPEN EDITORS
FS
fs.go
lock_test.go
nop.go
real.go
virtual.go
run test | debug test
func TestLock(t *testing.T) {
t.Parallel()
lock := "TESTLOCK"
for name, filesys := range map[string]Filesystem{
"virtual": NewVirtualFilesystem(),
"real": NewRealFilesystem(false),
} {
t.Run(name, func(t *testing.T) {
r, existed, err := filesys.Lock(lock)
if err != nil {
t.Fatalf("initial claim: %v", err)
}
if existed {
t.Fatal("initial claim: lock file already exists")
}
if _, existed, _ = filesys.Lock(lock); !existed {
t.Fatal("second claim: want existed true, have false")
}
if err := r.Release(); err != nil {
t.Fatalf("initial release: %v", err)
}
if err := r.Release(); err == nil {
t.Fatal("second release: want error, have none")
}
})
if filesys.Exists(lock) {
t.Errorf("%s: %s still exists", name, lock)
}
filesys.Remove(lock)
}
}
master*
5↓ 0↑
0 0
Ln 1, Col 1
Tab Size: 4
UTF-8
LF
Go

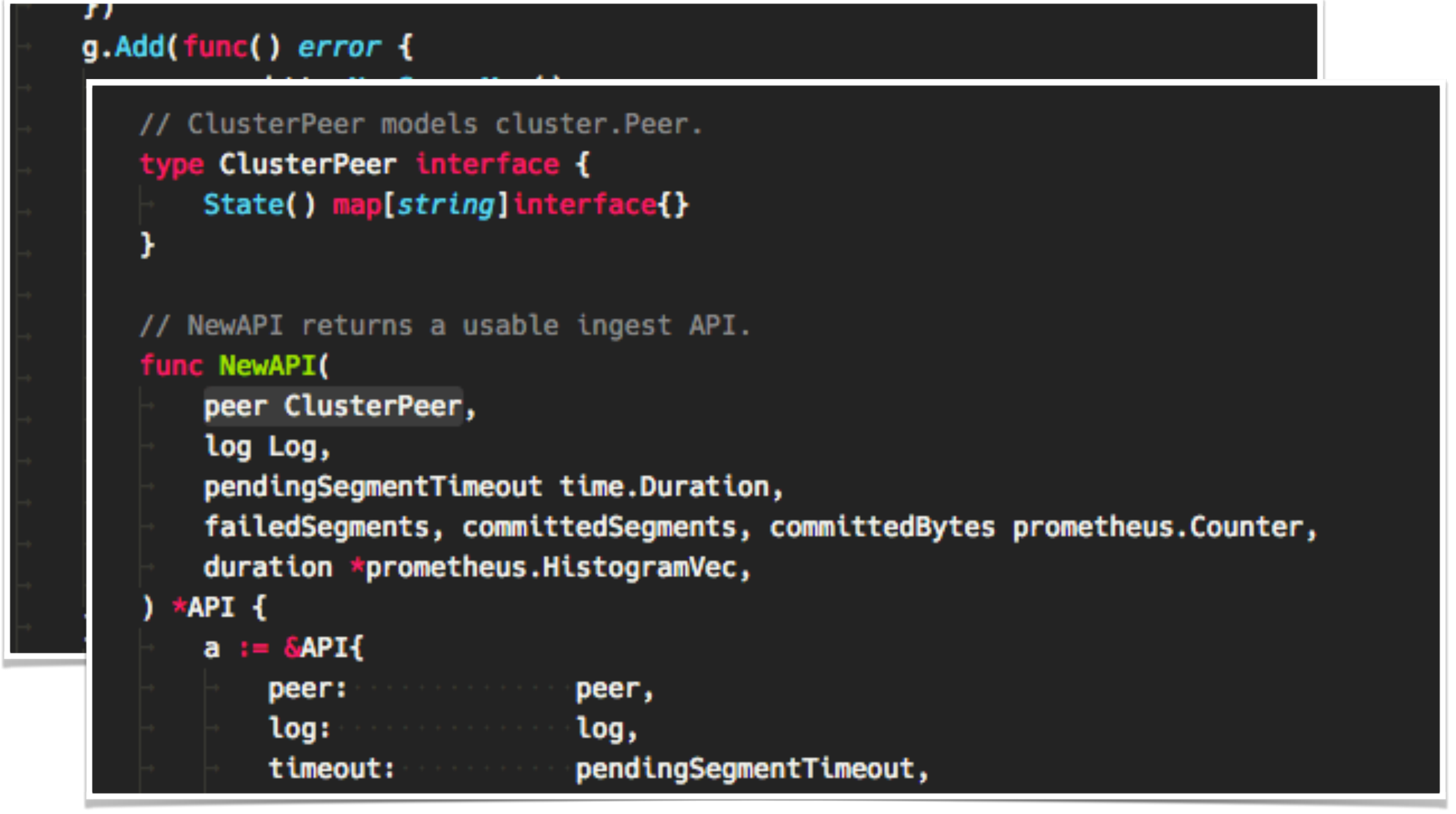

fs - The Go Programming Lang
localhost:8080/pkg/github.com/oklog/oklog/pkg/fs/
The Go Programming Language
Package fs
import "github.com/oklog/oklog/pkg/fs"
Overview
Index
Overview
Package fs abstracts the filesystem.
Index
type File
type Filesystem
func NewNopFilesystem() Filesystem
func NewRealFilesystem(mmap bool) Filesystem
func NewVirtualFilesystem() Filesystem
type Releaser
Package files
fs.go nop.go real.go virtual.go
type File
File is the subset of methods we use on an *os.File.
type File interface {
io.Reader
io.Writer
io.Closer
Name() string
Size() int64
Sync() error
}
fs - The Go Programming Lang
Hamburger
localhost:8080/pkg/github.com/oklog/oklog/pkg/fs/
type Filesystem
Filesystem collects the operations we need from a filesystem.
type Filesystem interface {
Create(path string) (File, error)
Open(path string) (File, error)
Remove(path string) error
Rename(oldname, newname string) error
Exists(path string) bool
MkdirAll(path string) error
Chtimes(path string, atime, mtime time.Time) error
Walk(root string, walkFn filepath.WalkFunc) error
Lock(path string) (r Releaser, existed bool, err error)
}
func NewNopFilesystem
func NewNopFilesystem() Filesystem
NewNopFilesystem has methods that always succeed but do nothing.
func NewRealFilesystem
func NewRealFilesystem(mmap bool) Filesystem
NewRealFilesystem yields a real disk filesystem with optional memory mapping for file reading.
func NewVirtualFilesystem ¶
func NewVirtualFilesystem() Filesystem
NewVirtualFilesystem yields an in-memory filesystem.
type Releaser
Releaser is returned by Lock calls.

A **tracer bullet** is "**a spike** with the current architecture, current technology set, current set of best practices **that results in production quality code**."

—Wikipedia on Scrum


main.go — oklog
ingest.go — oklog
EXPLORER
main.go
ingest.go
store.go
OPEN EDITORS
main.go cmd/oklog
ingest.go cmd/ok...
store.go cmd/okl...
OKLOG
cmd
oklog
forward.go
ingest.go
ingeststore.go
main_test.go
main.go
query.go
store.go
stream.go
testsvc.go
data
dist
pkg
cluster
fs
fs.go
lock_test.go
nop.go
real.go
virtual.go
group
ingest
testdata
api.go
conn_test.go
*segmentFlushAge, *segmentFlushSize,
connectedClients.WithLabelValues("bulk"),
ingestWriterBytes, ingestWriterRecords, ingestWriterSyncs,
flushedSegmentAge, flushedSegmentSize,
bulkListener.Close()
g.Add(func() error {
mux := http.NewServeMux()
mux.Handle("/ingest/", http.StripPrefix("/ingest", ingest.NewAPI(
peer,
ingestLog,
*segmentPendingTimeout,
failedSegments,
committedSegments,
committedBytes,
apiDuration,
)))
registerMetrics(mux)
registerProfile(mux)
return http.Serve(apiListener, mux)
apiListener.Close()
cancel := make(chan struct{})
return interrupt(cancel)
close(cancel)
return g.Run()
remove-mmap
Ln 231, Col 37
Tab Size: 4
UTF-8
LF
Go


store.go — oklog
EXPLORER
main.go
ingest.go
store.go
OPEN EDITORS
main.go cmd/oklog
ingest.go cmd/ok...
store.go cmd/okl...
OKLOG
cmd
oklog
forward.go
ingest.go
ingeststore.go
main_test.go
main.go
query.go
store.go
stream.go
testsvc.go
data
dist
pkg
cluster
fs
fs.go
lock_test.go
nop.go
real.go
virtual.go
group
ingest
testdata
api.go
conn_test.go
// Execution group.
var g group.Group
{
cancel := make(chan struct{})
g.Add(func() error {
<-cancel
return peer.Leave(time.Second)
}, func(error) {
close(cancel)
})
}
for i := 0; i < *segmentConsumers; i++ {
c := store.NewConsumer(
peer,
timeoutClient,
*segmentTargetSize,
*segmentTargetAge,
*segmentReplicationFactor,
consumedSegments,
consumedBytes,
replicatedSegments.WithLabelValues("egress"),
replicatedBytes.WithLabelValues("egress"),
log.With(logger, "component", "Consumer"),
)
g.Add(func() error {
c.Run()
return nil
}, func(error) {
c.Stop()
})
}
{
c := store.NewCompacter(
storeLog,
*segmentTargetSize,
*segmentRetain,
*segmentPurge,
remove-mmap
Ln 187, Col 37
Tab Size: 4
UTF-8
LF
Go

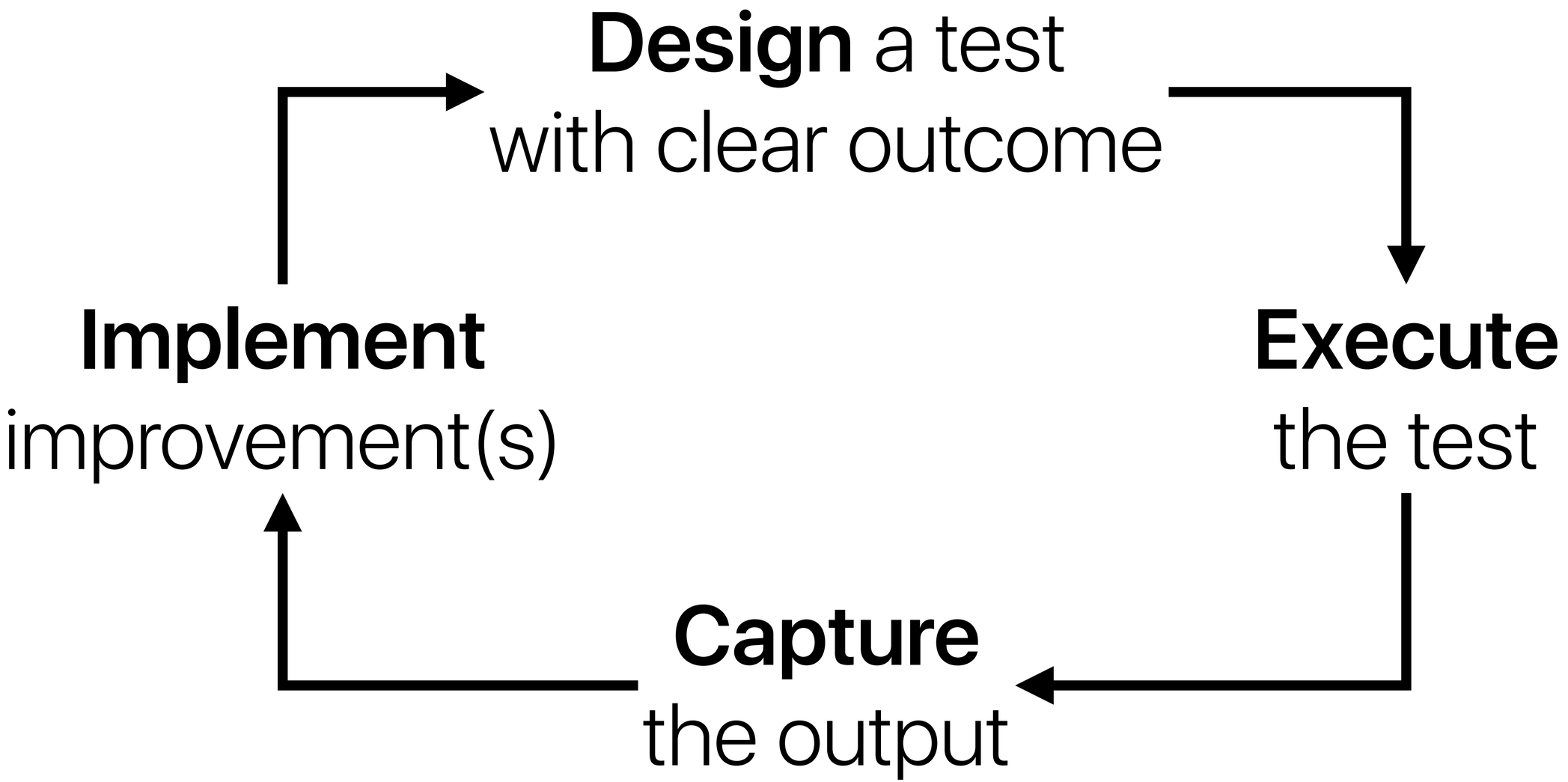


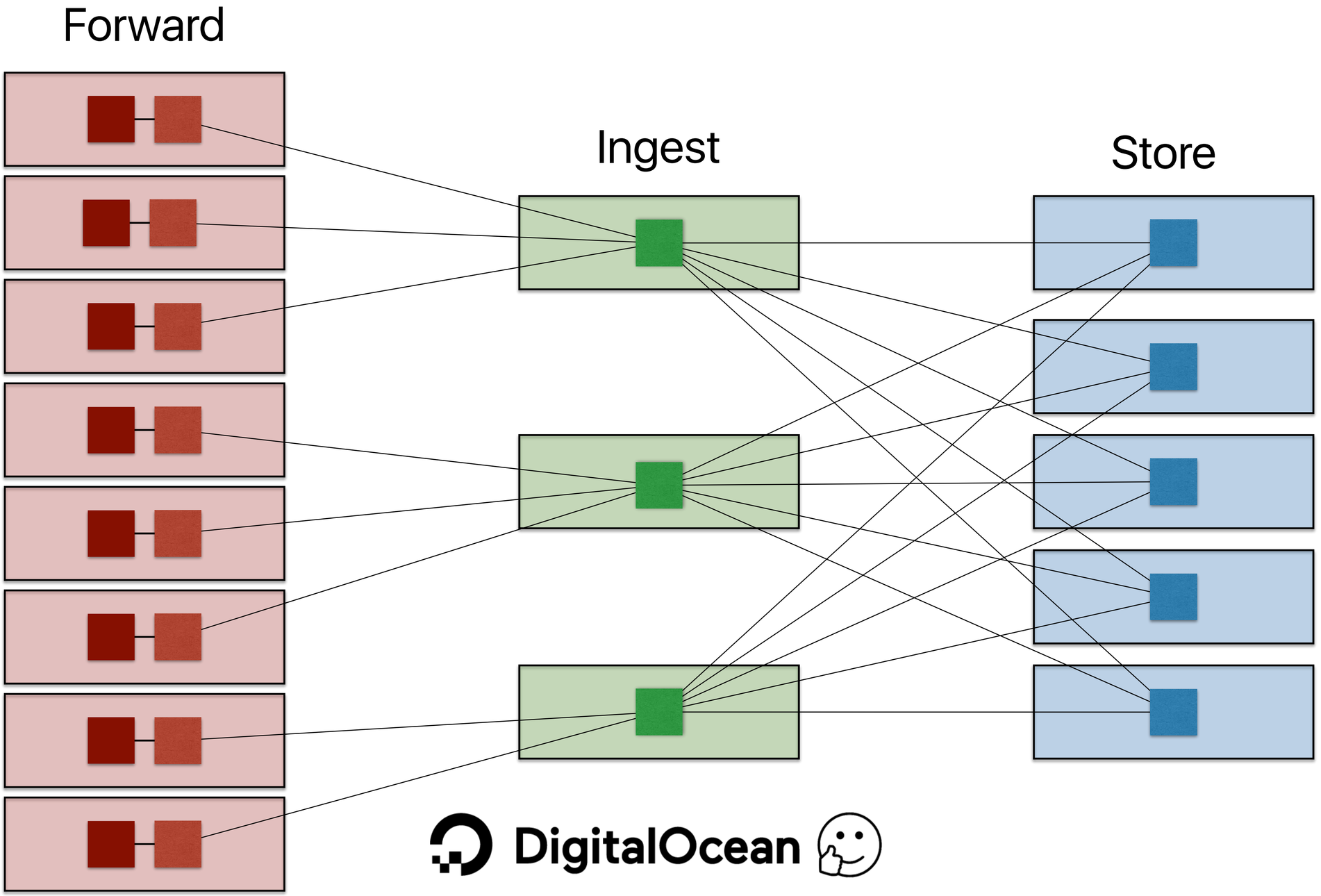
```
})
g.Add(func() error {
```

```
// ClusterPeer models cluster.Peer.
type ClusterPeer interface {
    State() map[string]interface{}
}

// NewAPI returns a usable ingest API.
func NewAPI(
    peer ClusterPeer,
    log Log,
    pendingSegmentTimeout time.Duration,
    failedSegments, committedSegments, committedBytes prometheus.Counter,
    duration *prometheus.HistogramVec,
) *API {
    a := &API{
        peer:              peer,
        log:               log,
        timeout:           pendingSegmentTimeout,
```

Optimize

Make it fast


Design a test
with clear outcome
Execute
the test
Capture
the output
Implement
improvement(s)


Forward
Ingest
Store
DigitalOcean


blep ~ _
01:fish*
0831h


blep ~ _
01:fish*
0834h

Instrumentation > Logs

USE

Brendan Gregg

Utilization
Saturation
Error rate

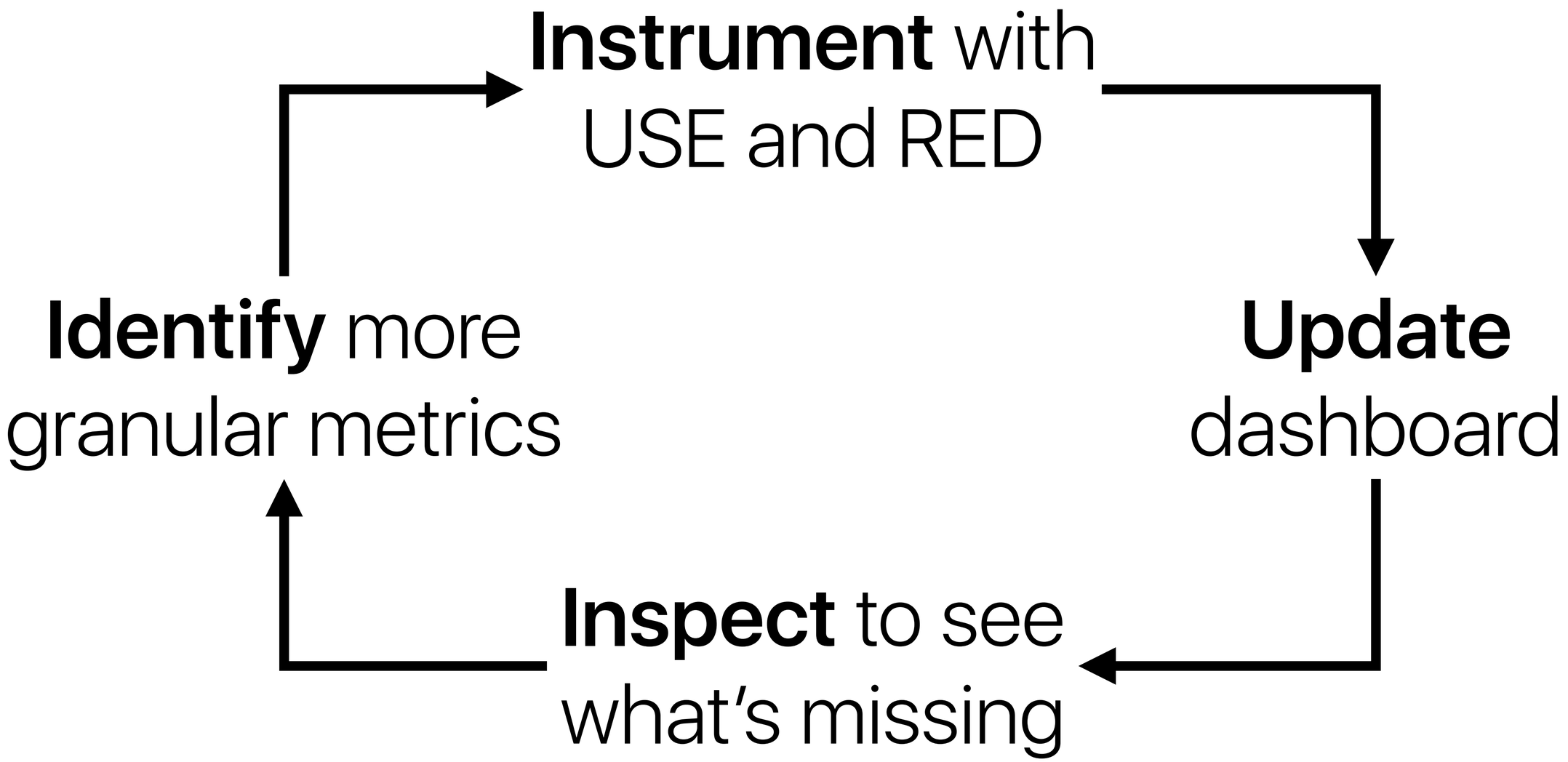
(Resources e.g. queues)

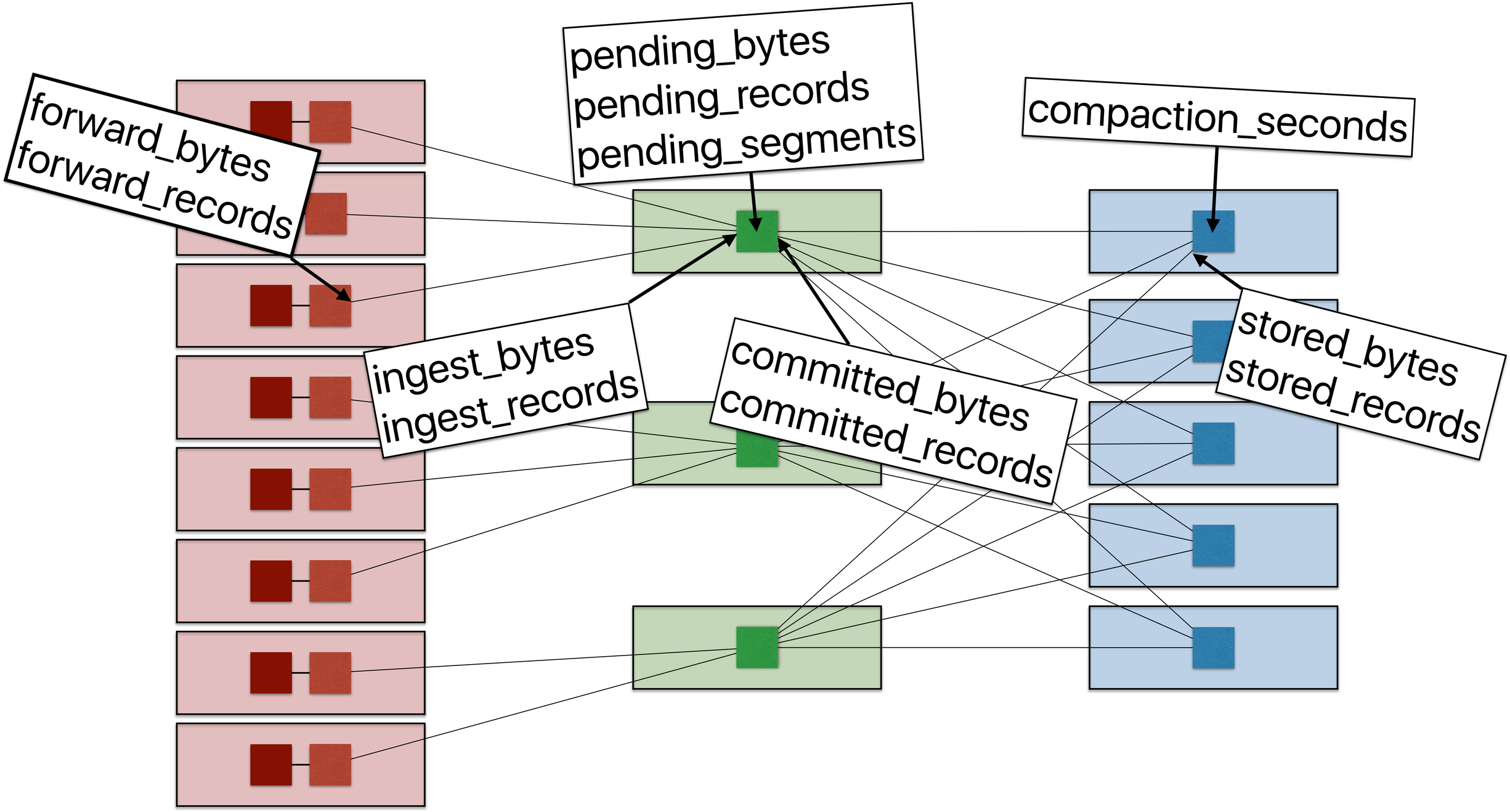
RED

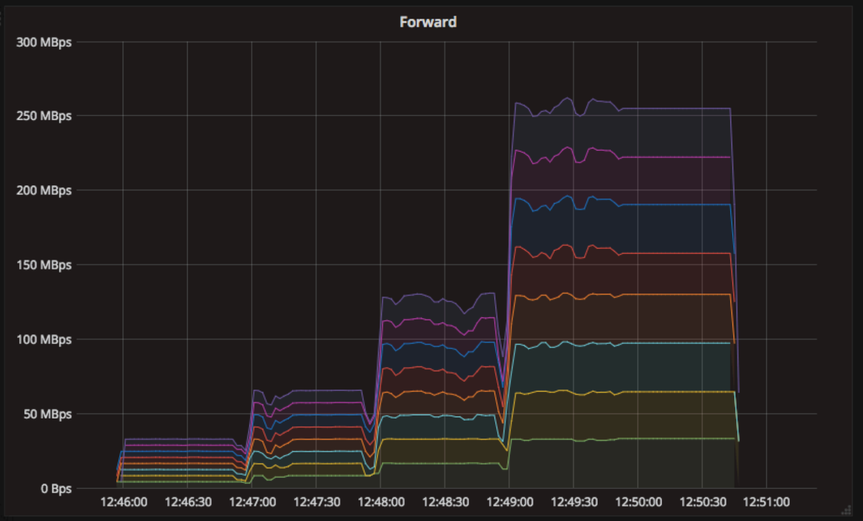
Tom Wilkie

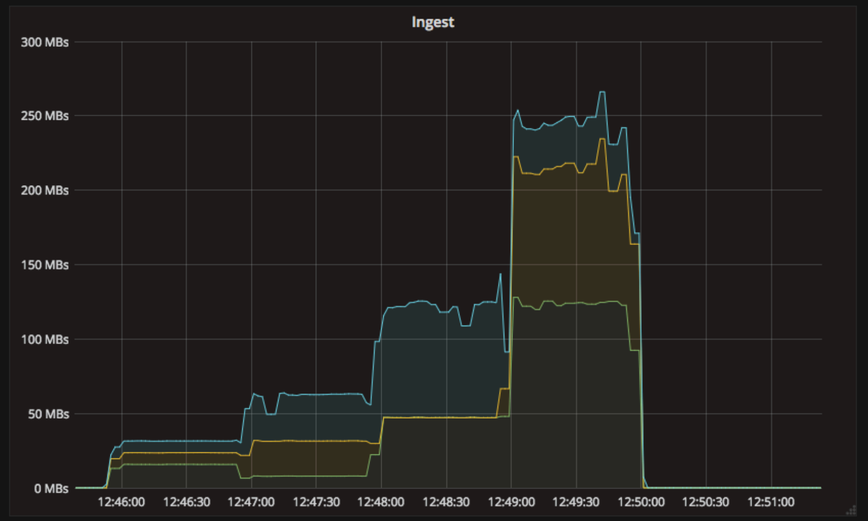
Request rate
Error rate
Duration

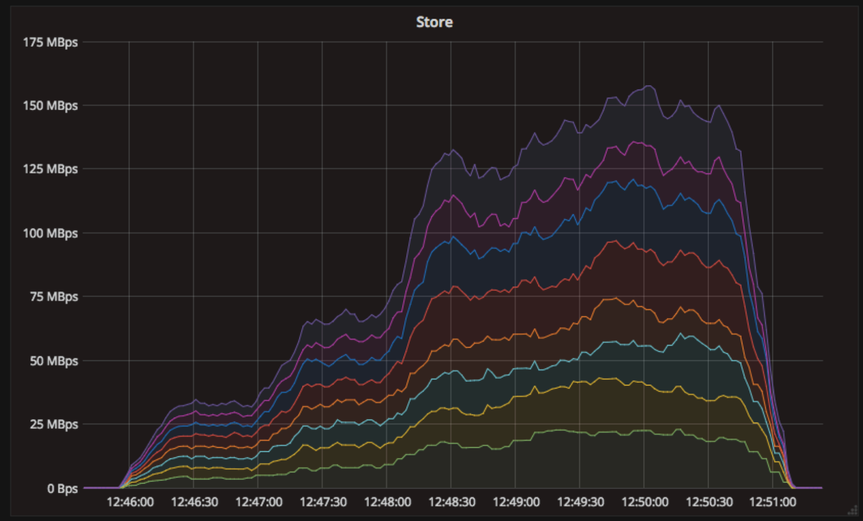
(Workers e.g. web servers)

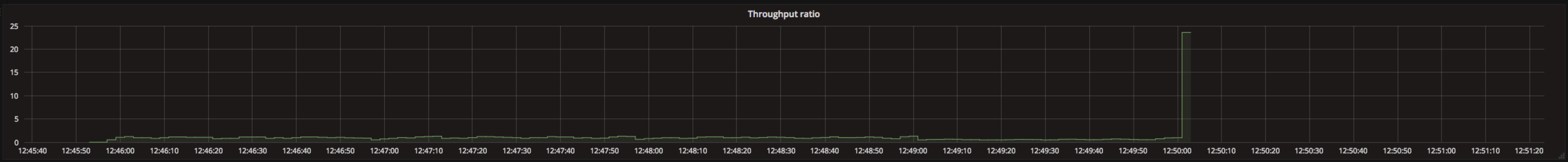

Instrument with USE and RED
Update dashboard
Inspect to see what's missing
Identify more granular metrics


forward_bytes
forward_records
pending_bytes
pending_records
pending_segments
compaction_seconds
ingest_bytes
ingest_records
committed_bytes
committed_records
stored_bytes
stored_records

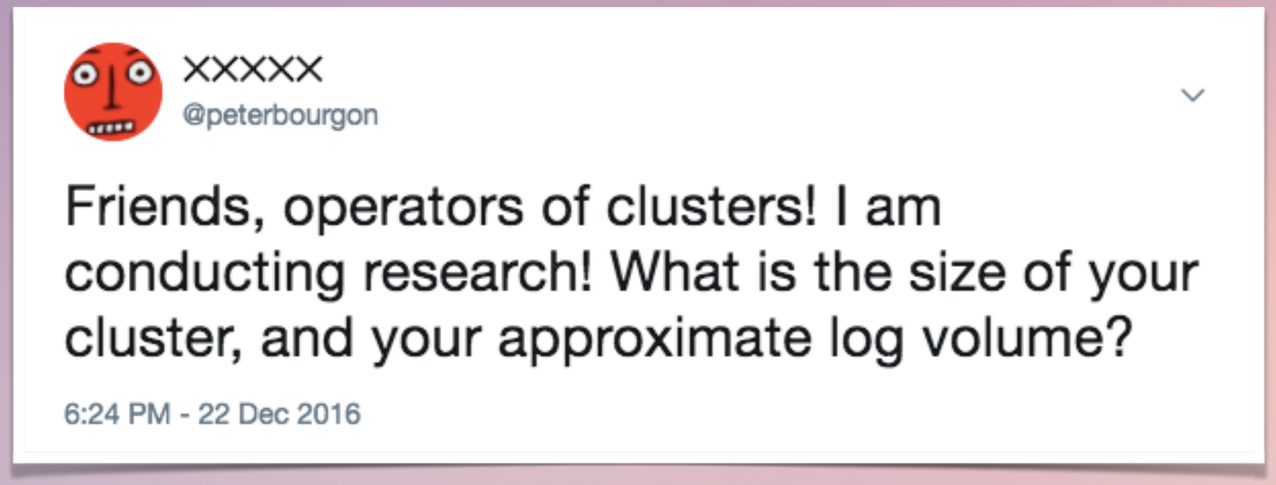

Forward
300 MBps
250 MBps
200 MBps
150 MBps
100 MBps
50 MBps
0 Bps
12:46:00
12:46:30
12:47:00
12:47:30
12:48:00
12:48:30
12:49:00
12:49:30
12:50:00
12:50:30
12:51:00


Ingest
300 MBs
250 MBs
200 MBs
150 MBs
100 MBs
50 MBs
0 MBs
12:46:00
12:46:30
12:47:00
12:47:30
12:48:00
12:48:30
12:49:00
12:49:30
12:50:00
12:50:30
12:51:00

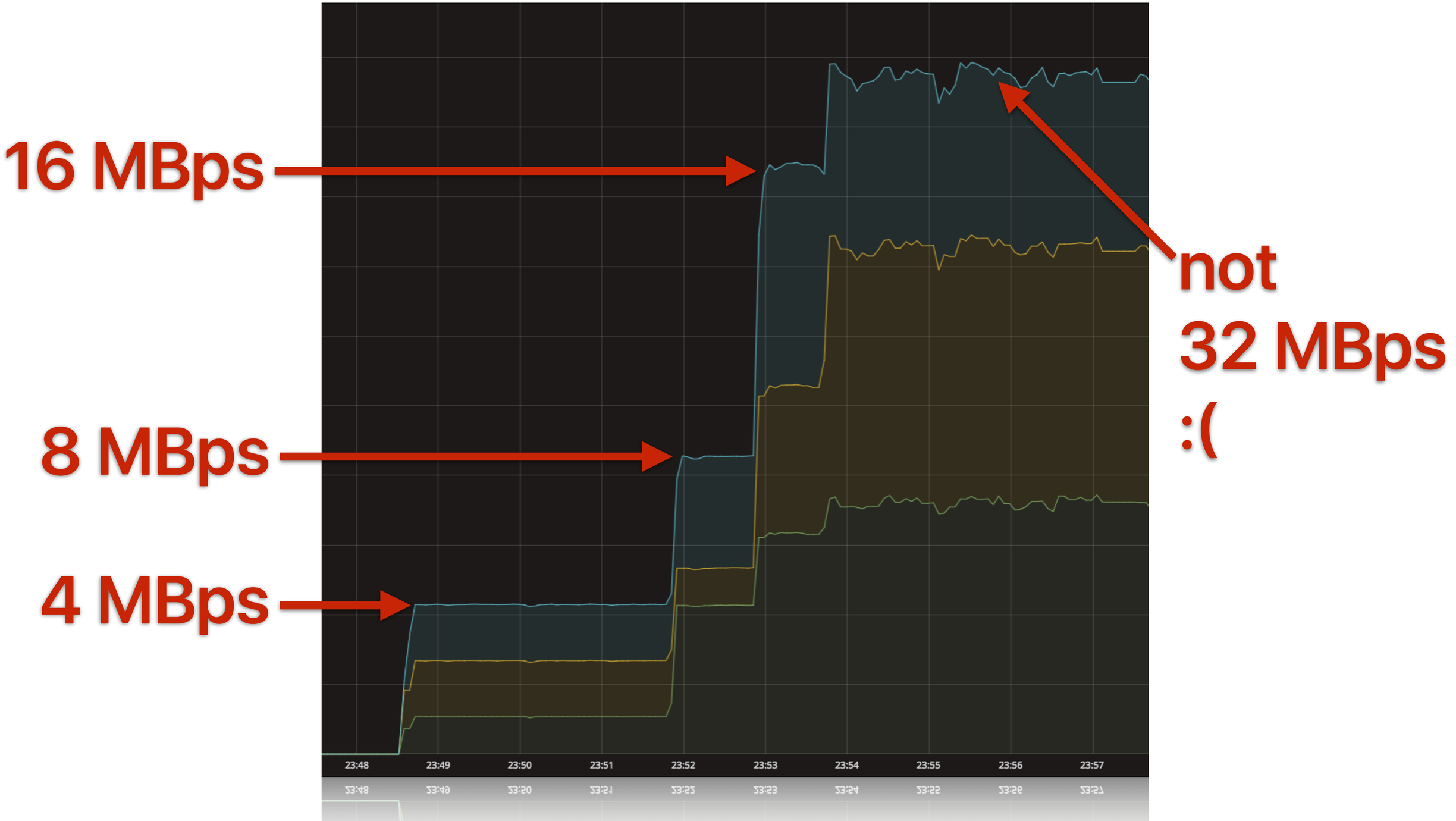

Store
175 MBps
150 MBps
125 MBps
100 MBps
75 MBps
50 MBps
25 MBps
0 Bps
12:46:00
12:46:30
12:47:00
12:47:30
12:48:00
12:48:30
12:49:00
12:49:30
12:50:00
12:50:30
12:51:00

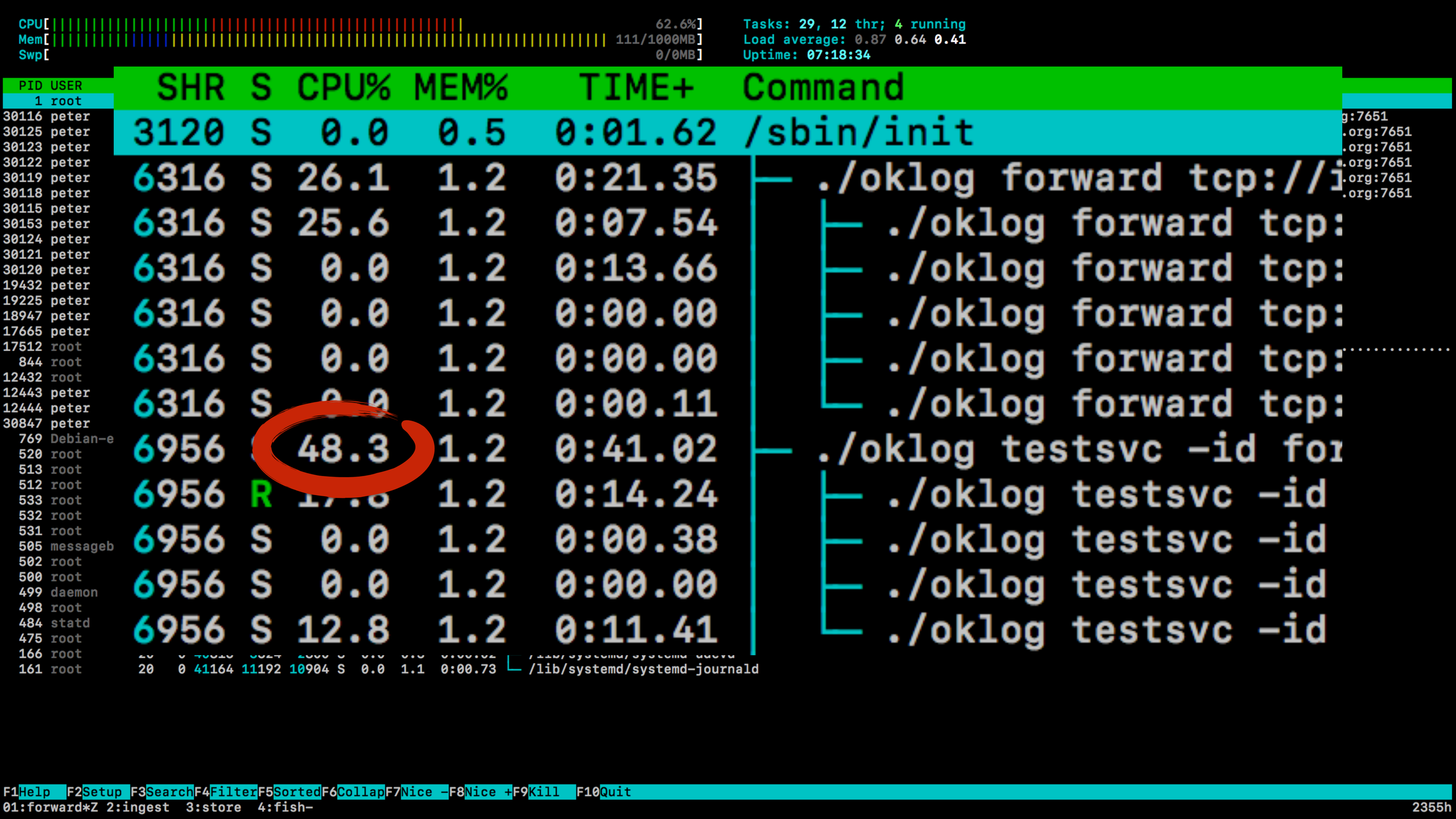

Throughput ratio
25
20
15
10
5
0
12:45:40
12:45:50
12:46:00
12:46:10
12:46:20
12:46:30
12:46:40
12:46:50
12:47:00
12:47:10
12:47:20
12:47:30
12:47:40
12:47:50
12:48:00
12:48:10
12:48:20
12:48:30
12:48:40
12:48:50
12:49:00
12:49:10
12:49:20
12:49:30
12:49:40
12:49:50
12:50:00
12:50:10
12:50:20
12:50:30
12:50:40
12:50:50
12:51:00
12:51:10
12:51:20

Optimizing ingestion



n = 8
min = 1KBps / node
mean = **~5MBps** / node
max = 20MBps / node (!)

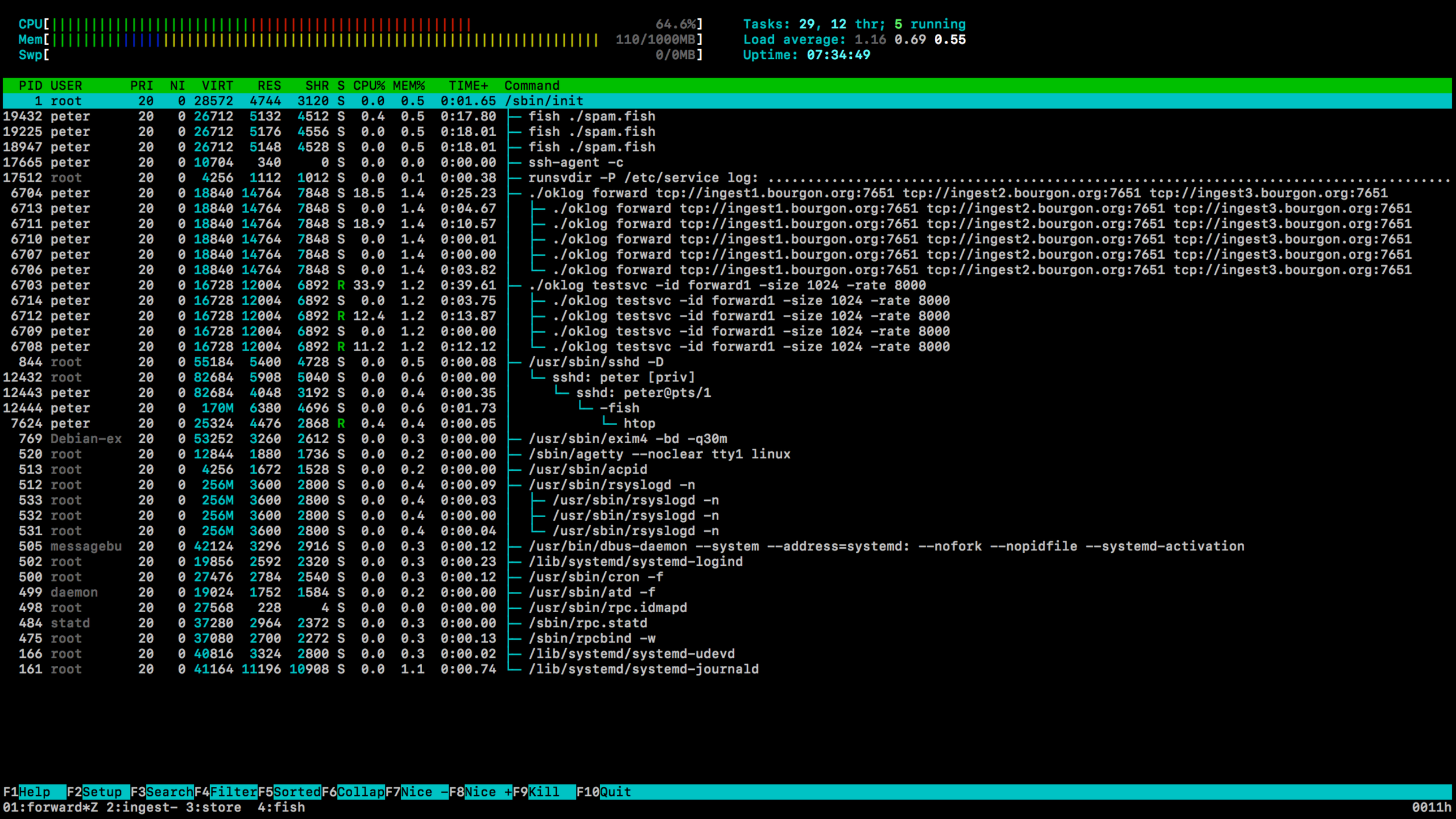

16 MBps
8 MBps
4 MBps
not
32 MBps
:(
23:48
23:49
23:50
23:51
23:52
23:53
23:54
23:55
23:56
23:57

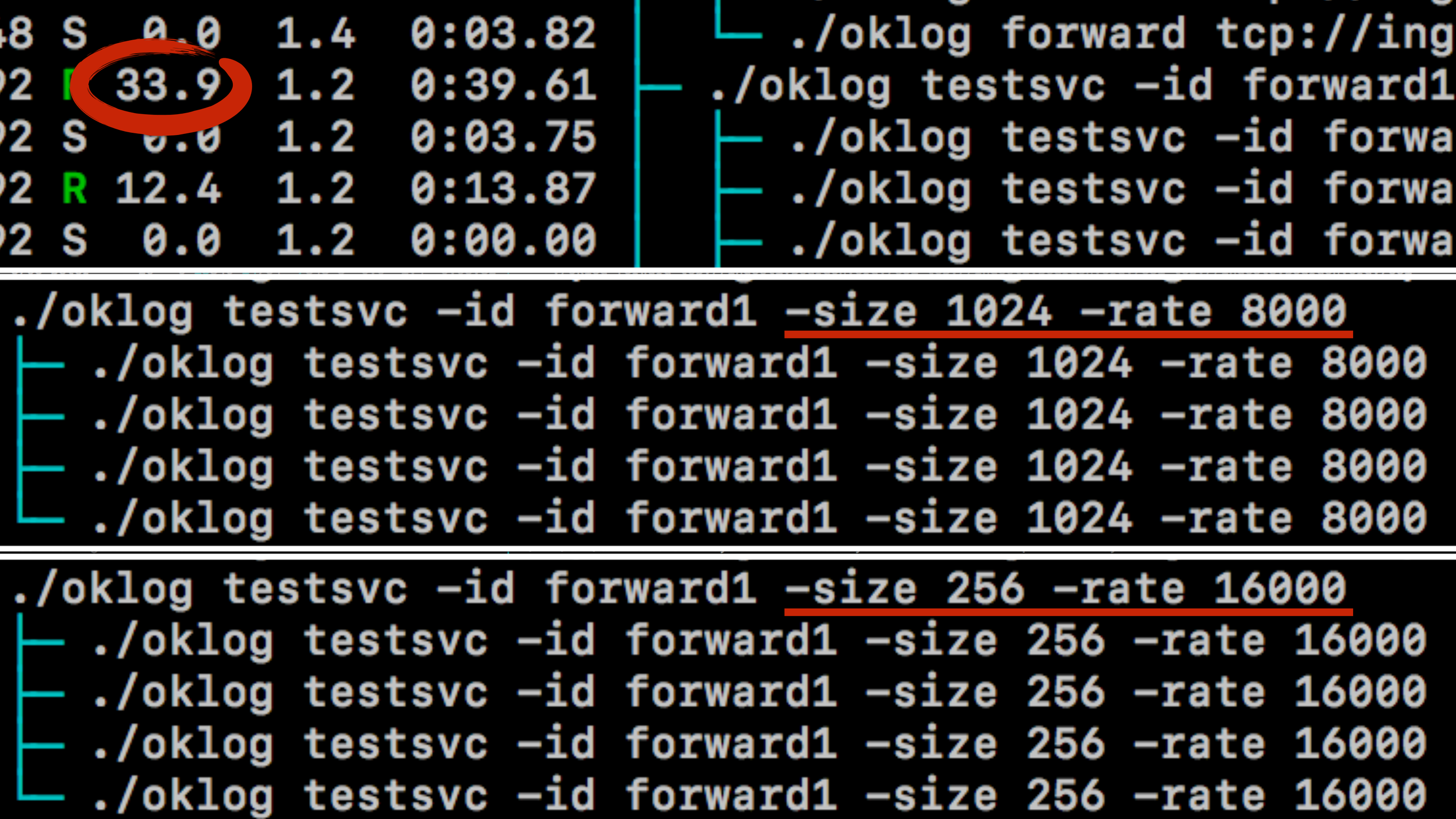

CPU[62.6%]
Mem[111/1000MB]
Swp[0/0MB]
Tasks: 29, 12 thr; 4 running
Load average: 0.87 0.64 0.41
Uptime: 07:18:34
PID USER SHR S CPU% MEM% TIME+ Command
1 root 3120 S 0.0 0.5 0:01.62 /sbin/init
30116 peter 6316 S 26.1 1.2 0:21.35 ├─ ./oklog forward tcp://i
30125 peter 6316 S 25.6 1.2 0:07.54 ├─ ./oklog forward tcp:
30123 peter 6316 S 0.0 1.2 0:13.66 ├─ ./oklog forward tcp:
30122 peter 6316 S 0.0 1.2 0:00.00 ├─ ./oklog forward tcp:
30119 peter 6316 S 0.0 1.2 0:00.00 ├─ ./oklog forward tcp:
30118 peter 6316 S 1.2 0:00.11 └─ ./oklog forward tcp:
30115 peter 6956 48.3 1.2 0:41.02 ├─ ./oklog testsvc -id for
30153 peter 6956 R 1.2 0:14.24 ├─ ./oklog testsvc -id
30124 peter 6956 S 0.0 1.2 0:00.38 ├─ ./oklog testsvc -id
30121 peter 6956 S 0.0 1.2 0:00.00 ├─ ./oklog testsvc -id
30120 peter 6956 S 12.8 1.2 0:11.41 └─ ./oklog testsvc -id
19432 peter
19225 peter
18947 peter
17665 peter
17512 root
844 root
12432 root
12443 peter
12444 peter
30847 peter
769 Debian-e
520 root
513 root
512 root
533 root
532 root
531 root
505 messageb
502 root
500 root
499 daemon
498 root
484 statd
475 root
166 root
161 root 20 0 41164 11192 10904 S 0.0 1.1 0:00.73 /lib/systemd/systemd-journald
:7651
.org:7651
.org:7651
.org:7651
.org:7651
.org:7651
F1Help F2Setup F3Search F4Filter F5Sorted F6Collap F7Nice - F8Nice + F9Kill F10Quit
01:forward*Z 2:ingest 3:store 4:fish-
2355h

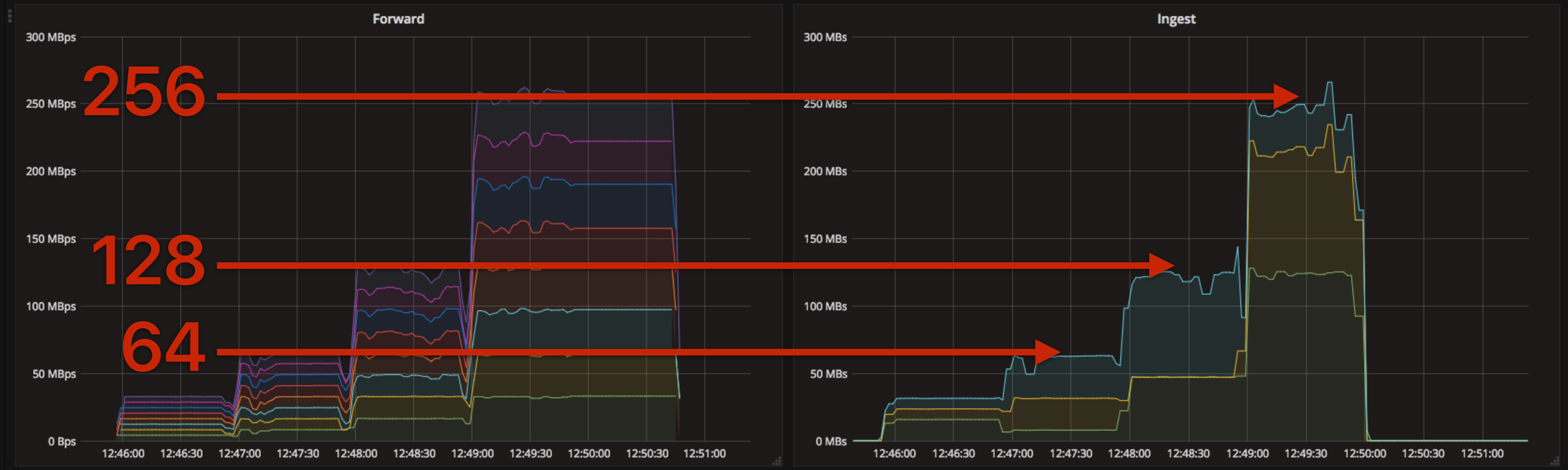
```
hz := float64(time.Second) / float64(*rate)
for range time.Tick(time.Duration(hz)) {
   fmt.Fprintf(os.Stdout,
      "%s %s\n",
      time.Now().Format(time.RFC3339),
      records[rant.Intn(len(records))],
   )
}
```

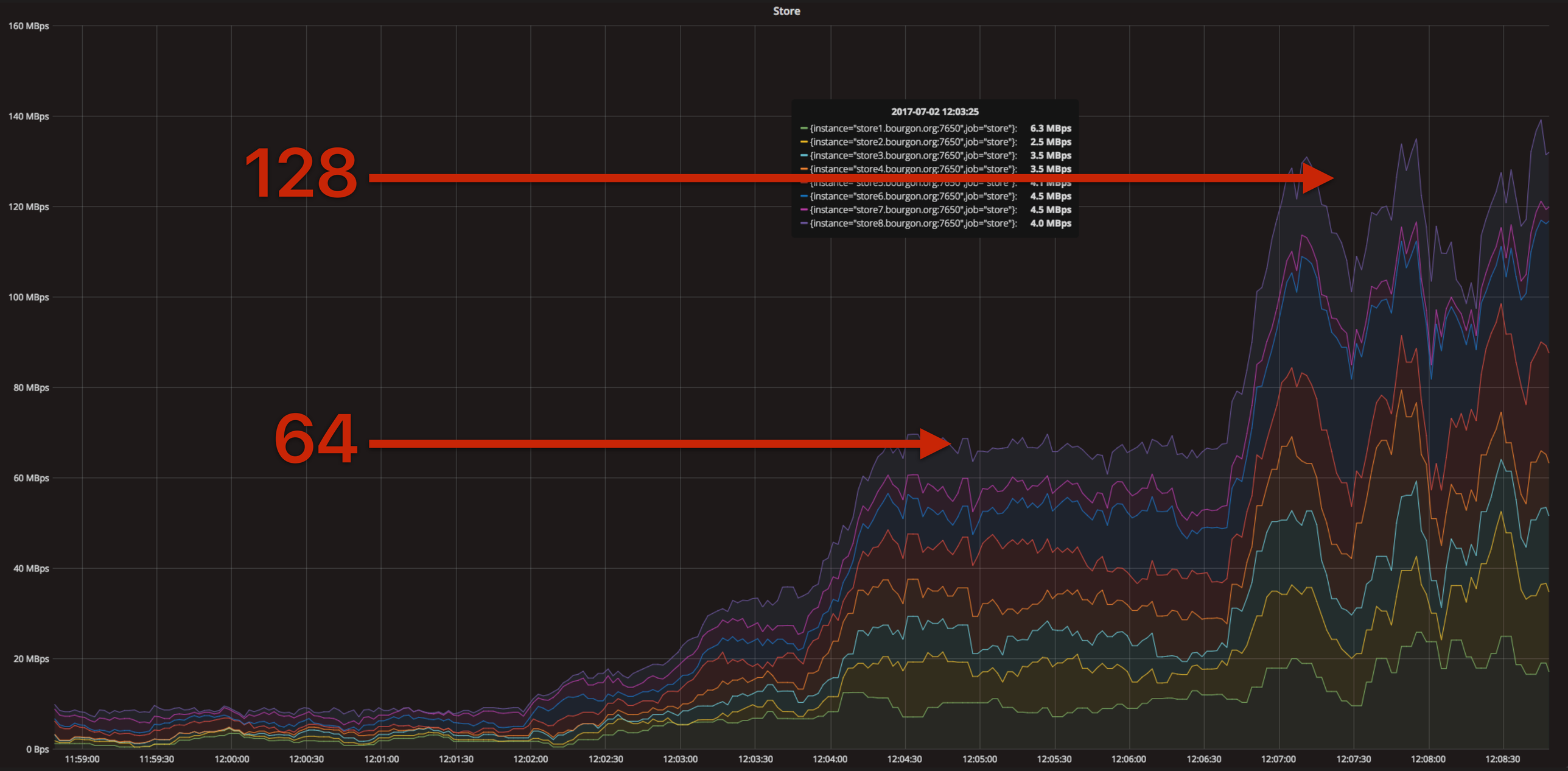
```
var (
  recordsPerCycle = 1
  timePerCycle    = time.Second / *rate
)
for timePerCycle < 50*time.Millisecond {
  recordsPerCycle *= 2
  timePerCycle *= 2
}
```

```
var count int
for range time.Tick(timePerCycle) {
  for i := 0; i < recordsPerCycle; i++ {
    fmt.Fprintf(os.Stdout,
      "%s %s\n",
      time.Now().Format(time.RFC3339),
      records[count%len(records)],
    )
    count++
  }
}
```


CPU[64.6%]
Mem[110/1000MB]
Swp[0/0MB]
Tasks: 29, 12 thr; 5 running
Load average: 1.16 0.69 0.55
Uptime: 07:34:49
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
1 root 20 0 28572 4744 3120 S 0.0 0.5 0:01.65 /sbin/init
19432 peter 20 0 26712 5132 4512 S 0.4 0.5 0:17.80 ├─ fish ./spam.fish
19225 peter 20 0 26712 5176 4556 S 0.0 0.5 0:18.01 ├─ fish ./spam.fish
18947 peter 20 0 26712 5148 4528 S 0.0 0.5 0:18.01 ├─ fish ./spam.fish
17665 peter 20 0 10704 340 0 S 0.0 0.0 0:00.00 ├─ ssh-agent -c
17512 root 20 0 4256 1112 1012 S 0.0 0.1 0:00.38 ├─ runsvdir -P /etc/service log: .....
6704 peter 20 0 18840 14764 7848 S 18.5 1.4 0:25.23 ├─ ./oklog forward tcp://ingest1.bourgon.org:7651 tcp://ingest2.bourgon.org:7651 tcp://ingest3.bourgon.org:7651
6713 peter 20 0 18840 14764 7848 S 0.0 1.4 0:04.67 │ ├─ ./oklog forward tcp://ingest1.bourgon.org:7651 tcp://ingest2.bourgon.org:7651 tcp://ingest3.bourgon.org:7651
6711 peter 20 0 18840 14764 7848 S 18.9 1.4 0:10.57 │ ├─ ./oklog forward tcp://ingest1.bourgon.org:7651 tcp://ingest2.bourgon.org:7651 tcp://ingest3.bourgon.org:7651
6710 peter 20 0 18840 14764 7848 S 0.0 1.4 0:00.01 │ ├─ ./oklog forward tcp://ingest1.bourgon.org:7651 tcp://ingest2.bourgon.org:7651 tcp://ingest3.bourgon.org:7651
6707 peter 20 0 18840 14764 7848 S 0.0 1.4 0:00.00 │ ├─ ./oklog forward tcp://ingest1.bourgon.org:7651 tcp://ingest2.bourgon.org:7651 tcp://ingest3.bourgon.org:7651
6706 peter 20 0 18840 14764 7848 S 0.0 1.4 0:03.82 │ └─ ./oklog forward tcp://ingest1.bourgon.org:7651 tcp://ingest2.bourgon.org:7651 tcp://ingest3.bourgon.org:7651
6703 peter 20 0 16728 12004 6892 R 33.9 1.2 0:39.61 ├─ ./oklog testsvc -id forward1 -size 1024 -rate 8000
6714 peter 20 0 16728 12004 6892 S 0.0 1.2 0:03.75 │ ├─ ./oklog testsvc -id forward1 -size 1024 -rate 8000
6712 peter 20 0 16728 12004 6892 R 12.4 1.2 0:13.87 │ ├─ ./oklog testsvc -id forward1 -size 1024 -rate 8000
6709 peter 20 0 16728 12004 6892 S 0.0 1.2 0:00.00 │ ├─ ./oklog testsvc -id forward1 -size 1024 -rate 8000
6708 peter 20 0 16728 12004 6892 R 11.2 1.2 0:12.12 │ └─ ./oklog testsvc -id forward1 -size 1024 -rate 8000
844 root 20 0 55184 5400 4728 S 0.0 0.5 0:00.08 ├─ /usr/sbin/sshd -D
12432 root 20 0 82684 5908 5040 S 0.0 0.6 0:00.00 │ └─ sshd: peter [priv]
12443 peter 20 0 82684 4048 3192 S 0.0 0.4 0:00.35 │ └─ sshd: peter@pts/1
12444 peter 20 0 170M 6380 4696 S 0.0 0.6 0:01.73 │ └─ -fish
7624 peter 20 0 25324 4476 2868 R 0.4 0.4 0:00.05 │ └─ htop
769 Debian-ex 20 0 53252 3260 2612 S 0.0 0.3 0:00.00 ├─ /usr/sbin/exim4 -bd -q30m
520 root 20 0 12844 1880 1736 S 0.0 0.2 0:00.00 ├─ /sbin/agetty --noclear tty1 linux
513 root 20 0 4256 1672 1528 S 0.0 0.2 0:00.00 ├─ /usr/sbin/acpid
512 root 20 0 256M 3600 2800 S 0.0 0.4 0:00.09 ├─ /usr/sbin/rsyslogd -n
533 root 20 0 256M 3600 2800 S 0.0 0.4 0:00.03 │ ├─ /usr/sbin/rsyslogd -n
532 root 20 0 256M 3600 2800 S 0.0 0.4 0:00.00 │ ├─ /usr/sbin/rsyslogd -n
531 root 20 0 256M 3600 2800 S 0.0 0.4 0:00.04 │ └─ /usr/sbin/rsyslogd -n
505 messagebu 20 0 42124 3296 2916 S 0.0 0.3 0:00.12 ├─ /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation
502 root 20 0 19856 2592 2320 S 0.0 0.3 0:00.23 ├─ /lib/systemd/systemd-logind
500 root 20 0 27476 2784 2540 S 0.0 0.3 0:00.12 ├─ /usr/sbin/cron -f
499 daemon 20 0 19024 1752 1584 S 0.0 0.2 0:00.00 ├─ /usr/sbin/atd -f
498 root 20 0 27568 228 4 S 0.0 0.0 0:00.00 ├─ /usr/sbin/rpc.idmapd
484 statd 20 0 37280 2964 2372 S 0.0 0.3 0:00.00 ├─ /sbin/rpc.statd
475 root 20 0 37080 2700 2272 S 0.0 0.3 0:00.13 ├─ /sbin/rpcbind -w
166 root 20 0 40816 3324 2800 S 0.0 0.3 0:00.02 ├─ /lib/systemd/systemd-udevd
161 root 20 0 41164 11196 10908 S 0.0 1.1 0:00.74 └─ /lib/systemd/systemd-journald
F1Help F2Setup F3SearchF4FilterF5SortedF6CollapF7Nice -F8Nice +F9Kill F10Quit
01:forward*Z 2:ingest- 3:store 4:fish
0011h


48 S 0.0 1.4 0:03.82
92 R 33.9 1.2 0:39.61
92 S 0.0 1.2 0:03.75
92 R 12.4 1.2 0:13.87
92 S 0.0 1.2 0:00.00
└─ ./oklog forward tcp://ing
├─ ./oklog testsvc -id forward1
├─ ./oklog testsvc -id forwa
├─ ./oklog testsvc -id forwa
├─ ./oklog testsvc -id forwa
./oklog testsvc -id forward1 -size 1024 -rate 8000
├─ ./oklog testsvc -id forward1 -size 1024 -rate 8000
├─ ./oklog testsvc -id forward1 -size 1024 -rate 8000
├─ ./oklog testsvc -id forward1 -size 1024 -rate 8000
└─ ./oklog testsvc -id forward1 -size 1024 -rate 8000
./oklog testsvc -id forward1 -size 256 -rate 16000
├─ ./oklog testsvc -id forward1 -size 256 -rate 16000
├─ ./oklog testsvc -id forward1 -size 256 -rate 16000
├─ ./oklog testsvc -id forward1 -size 256 -rate 16000
└─ ./oklog testsvc -id forward1 -size 256 -rate 16000


Forward
300 MBps
250 MBps
200 MBps
150 MBps
100 MBps
50 MBps
0 Bps
12:46:00
12:46:30
12:47:00
12:47:30
12:48:00
12:48:30
12:49:00
12:49:30
12:50:00
12:50:30
12:51:00
256
128
64
Ingest
300 MBs
250 MBs
200 MBs
150 MBs
100 MBs
50 MBs
0 MBs
12:46:00
12:46:30
12:47:00
12:47:30
12:48:00
12:48:30
12:49:00
12:49:30
12:50:00
12:50:30
12:51:00


Store
160 MBps
140 MBps
120 MBps
100 MBps
80 MBps
60 MBps
40 MBps
20 MBps
0 Bps
11:59:00
11:59:30
12:00:00
12:00:30
12:01:00
12:01:30
12:02:00
12:02:30
12:03:00
12:03:30
12:04:00
12:04:30
12:05:00
12:05:30
12:06:00
12:06:30
12:07:00
12:07:30
12:08:00
12:08:30
2017-07-02 12:03:25
{instance="store1.bourgon.org:7650",job="store"}: 6.3 MBps
{instance="store2.bourgon.org:7650",job="store"}: 2.5 MBps
{instance="store3.bourgon.org:7650",job="store"}: 3.5 MBps
{instance="store4.bourgon.org:7650",job="store"}: 3.5 MBps
{instance="store6.bourgon.org:7650",job="store"}: 4.5 MBps
{instance="store7.bourgon.org:7650",job="store"}: 4.5 MBps
{instance="store8.bourgon.org:7650",job="store"}: 4.0 MBps
128
64

```go
package main

import (
	"flag"
	"fmt"
	"net"
	"net/http"
	"net/http/pprof"
	"net/url"
	"os"
	"os/signal"
	"runtime"
	"strconv"
	"strings"
	"syscall"
	"text/tabwriter"
```

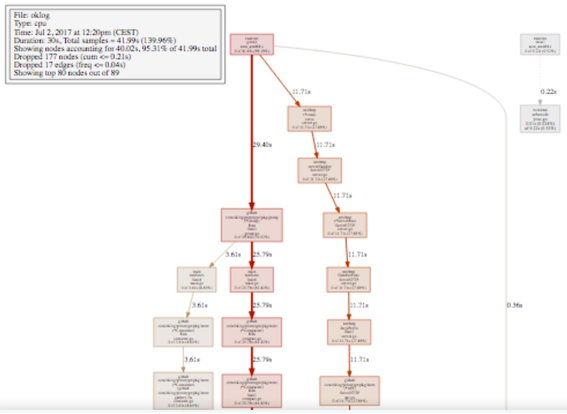
```
148
149 func registerProfile(mux *http.ServeMux) {
150     mux.HandleFunc("/debug/pprof/", pprof.Index)
151     mux.HandleFunc("/debug/pprof/cmdline", pprof.Cmdline)
152     mux.HandleFunc("/debug/pprof/profile", pprof.Profile)
153     mux.HandleFunc("/debug/pprof/symbol", pprof.Symbol)
154     mux.HandleFunc("/debug/pprof/trace", pprof.Trace)
155     mux.Handle("/debug/pprof/block", pprof.Handler("block"))
156     mux.Handle("/debug/pprof/goroutine", pprof.Handler("goroutine"))
157     mux.Handle("/debug/pprof/heap", pprof.Handler("heap"))
158     mux.Handle("/debug/pprof/threadcreate", pprof.Handler("threadcreate"
159 }
160
161 func usageFor(fs *flag.FlagSet, short string) func() {
162     return func() {
163         fmt.Fprintf(os.Stderr, "USAGE\n")
164         fmt.Fprintf(os.Stderr, "  %s\n", short)
165         fmt.Fprintf(os.Stderr, "\n")
```

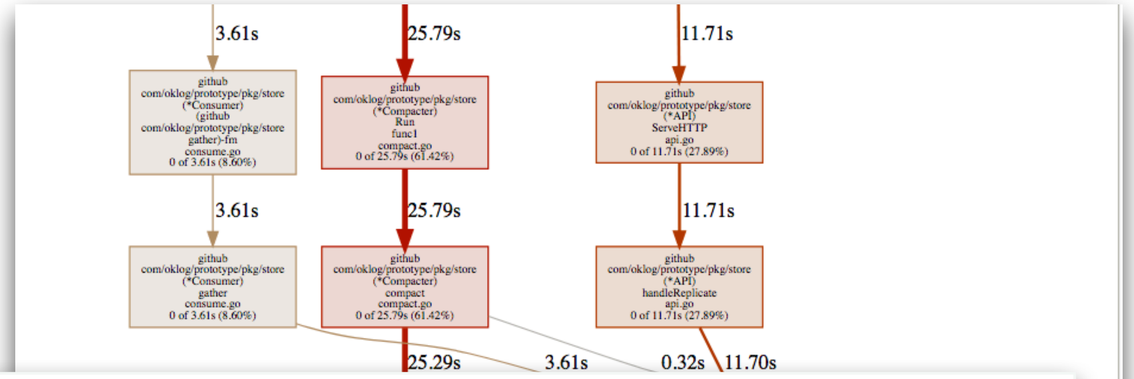

blep ~ _
01:fish*
0957h


blep ~ _
01:fish*
1011h

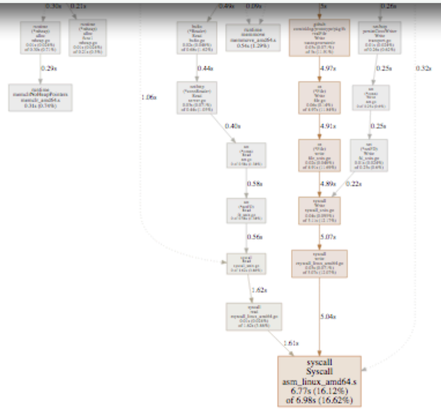
```
ingest1 ~ go tool pprof -alloc_objects bin/oklog http://ingest1.bourgon.org:7650/debug/pprof/heap
Fetching profile from http://ingest1.bourgon.org:7650/debug/pprof/heap
Saved profile in /home/peter/pprof/pprof.oklog.ingest1.bourgon.org:7650.alloc_objects.alloc_space.inuse_objects.inuse_space.007.pb.gz
Entering interactive mode (type "help" for commands)
(pprof) _
```

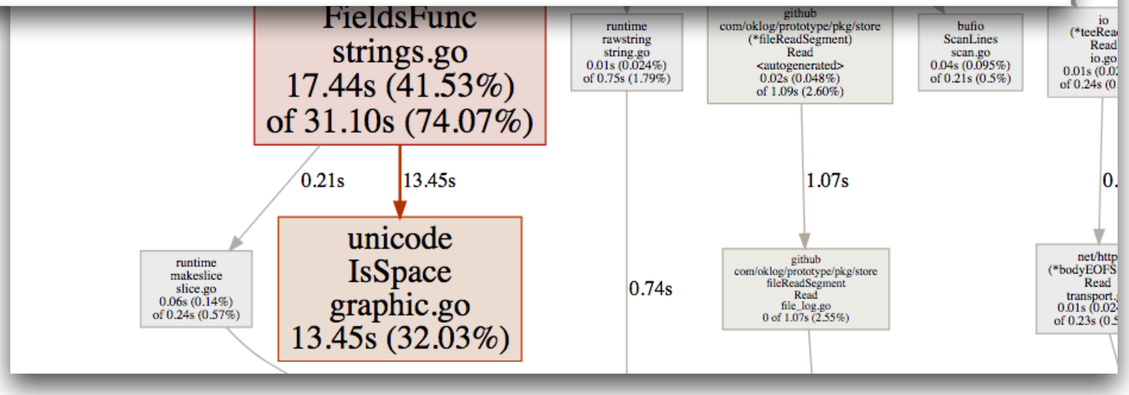
01:ssh* 1009h





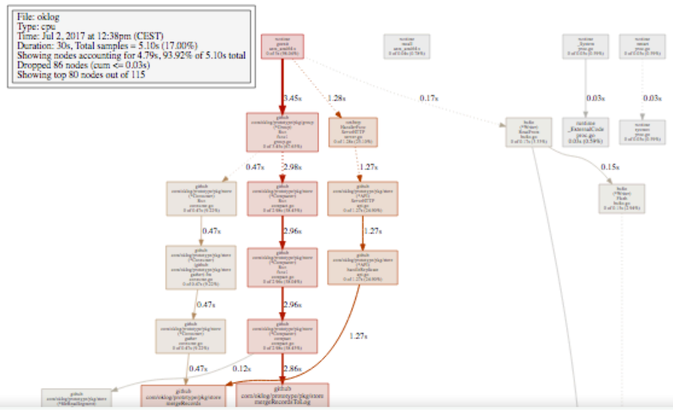

```
(pprof) top5
Showing nodes accounting for 38.51s, 91.71% of 41.99s total
Dropped 177 nodes (cum <= 0.21s)
Showing top 5 nodes out of 89
      flat  flat%   sum%        cum   cum%
    17.44s 41.53% 41.53%     31.10s 74.07%  strings.FieldsFunc
    13.45s 32.03% 73.57%     13.45s 32.03%  unicode.IsSpace
     6.77s 16.12% 89.69%      6.98s 16.62%  syscall.Syscall
     0.54s  1.29% 90.97%      0.54s  1.29%  runtime.memmove
     0.31s  0.74% 91.71%      0.31s  0.74%  runtime.memclrNoHeapPointers
```

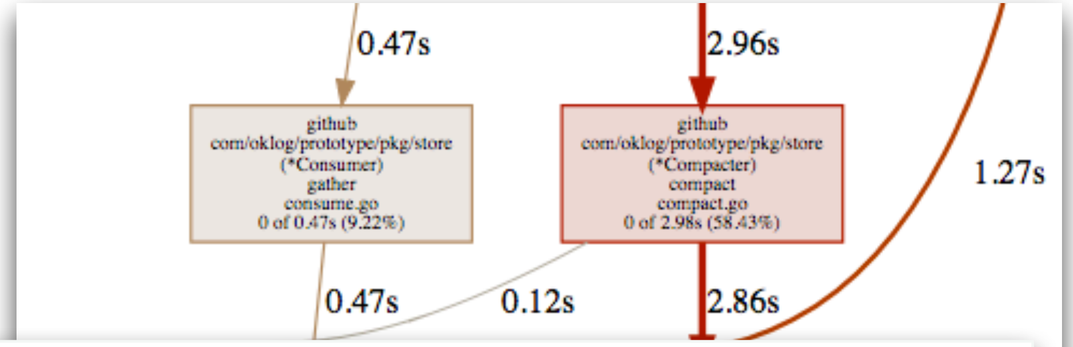




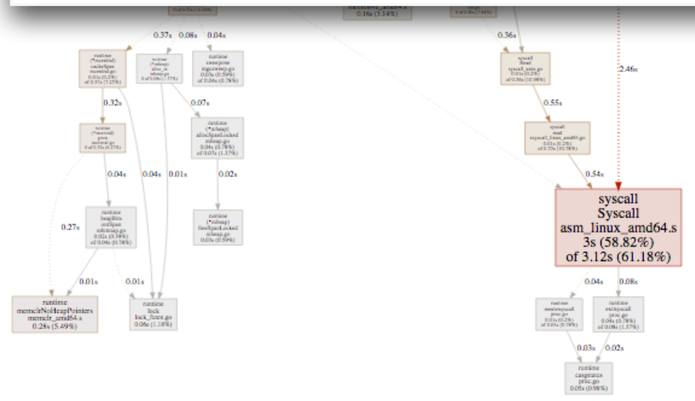
```
// Advance takes the next record from the reader.
advance := func(i int) error {
  if ok[i] = scanner[i].Scan(); ok[i] {
    record[i] = scanner[i].Bytes()
    id[i] = strings.Fields(record[i])[0]
    ...
```

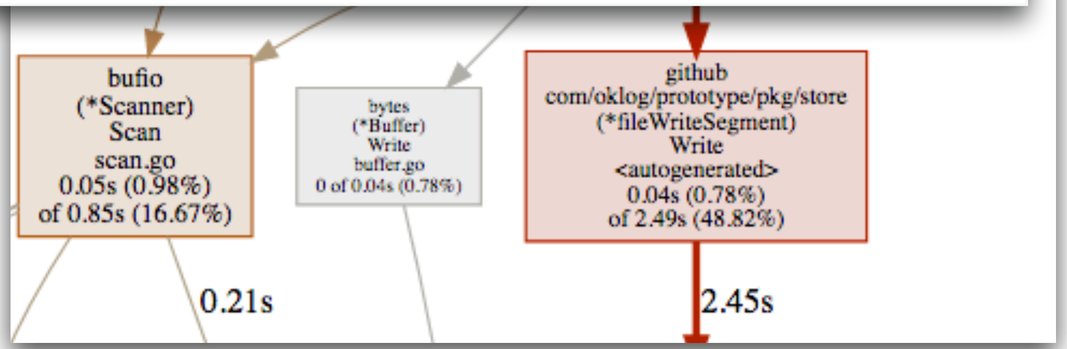
```
// Advance takes the next record from the reader.
advance := func(i int) error {
  if ok[i] = scanner[i].Scan(); ok[i] {
    record[i] = scanner[i].Bytes()
    id[i] = record[i][:ulid.EncodedSize]
    ...
```



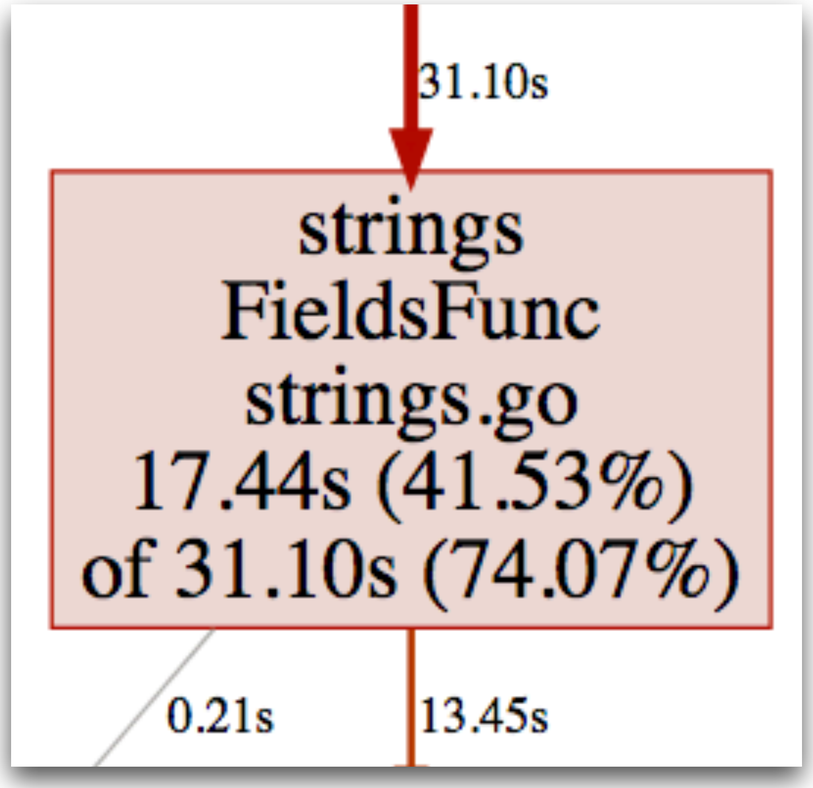



```
(pprof) top5
Showing nodes accounting for 3700ms, 72.55% of 5100ms total
Dropped 86 nodes (cum <= 25.50ms)
Showing top 5 nodes out of 115
      flat  flat%   sum%        cum   cum%
    3000ms 58.82% 58.82%     3120ms 61.18%  syscall.Syscall
     280ms  5.49% 64.31%      280ms  5.49%  runtime.memclrNoHeapPointers
     200ms  3.92% 68.24%      670ms 13.14%  runtime.mallocgc
     160ms  3.14% 71.37%      160ms  3.14%  runtime.memmove
      60ms  1.18% 72.55%       60ms  1.18%  runtime.cmpbody
```

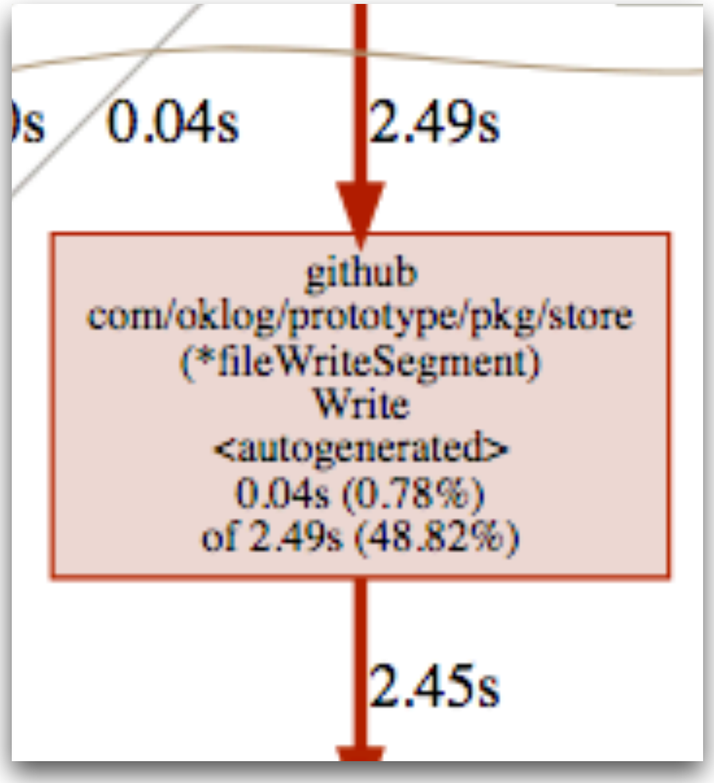


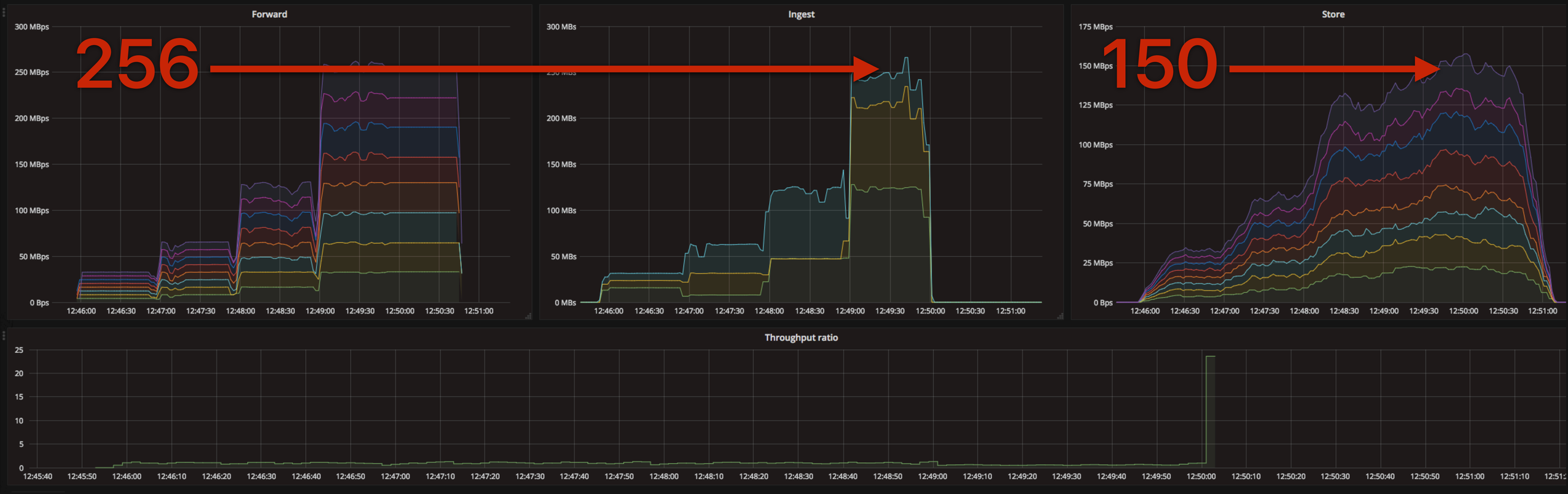


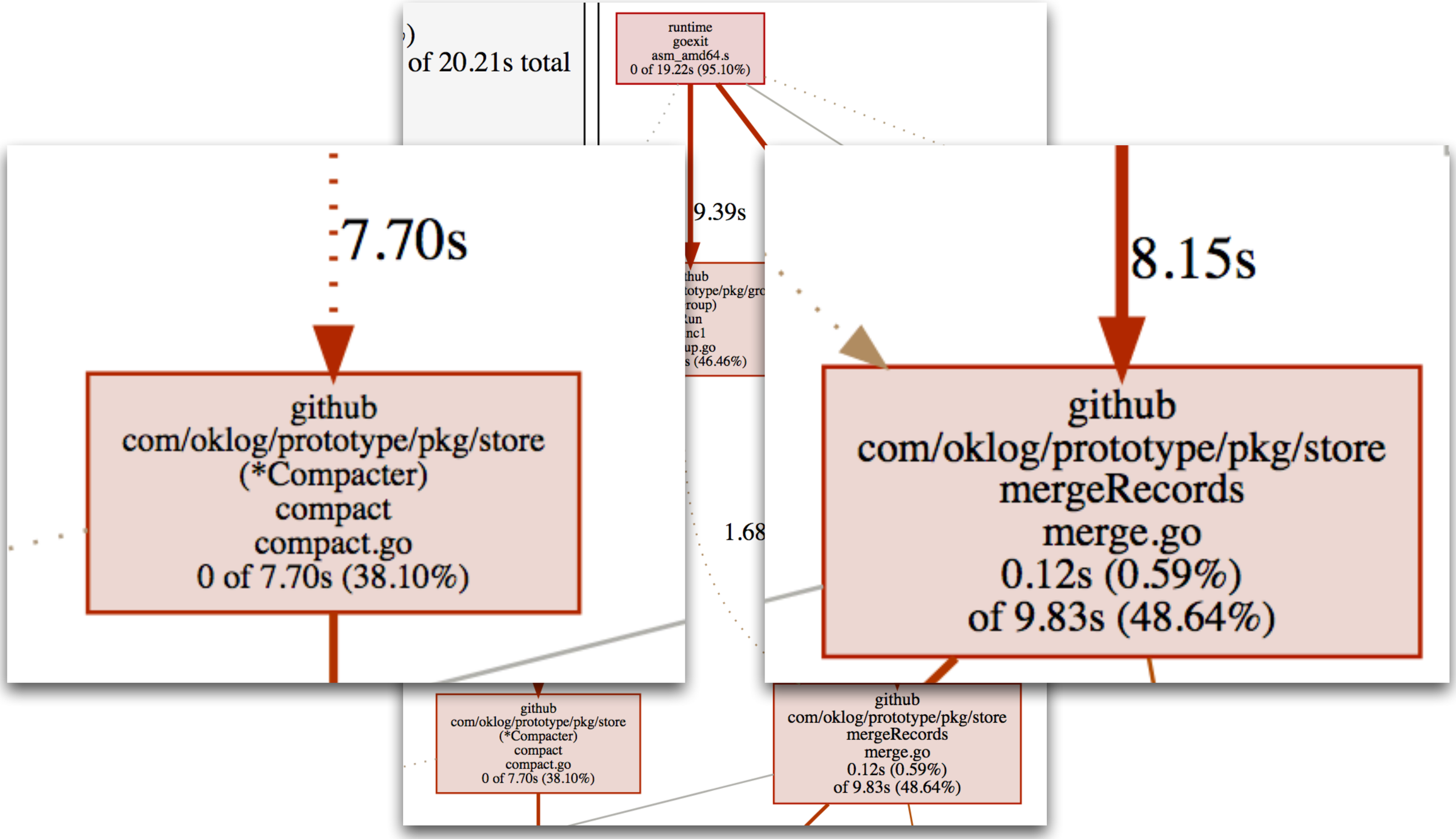
31.10s


31.10s
strings
FieldsFunc
strings.go
17.44s (41.53%)
of 31.10s (74.07%)
0.21s
13.45s

2.49s


0.04s
2.49s
github
com/oklog/prototype/pkg/store
(*fileWriteSegment)
Write
<autogenerated>
0.04s (0.78%)
of 2.49s (48.82%)
2.45s


Forward
300 MBps
250 MBps
200 MBps
150 MBps
100 MBps
50 MBps
0 Bps
12:46:00 12:46:30 12:47:00 12:47:30 12:48:00 12:48:30 12:49:00 12:49:30 12:50:00 12:50:30 12:51:00
256
Ingest
300 MBs
200 MBs
150 MBs
100 MBs
50 MBs
0 MBs
12:46:00 12:46:30 12:47:00 12:47:30 12:48:00 12:48:30 12:49:00 12:49:30 12:50:00 12:50:30 12:51:00
Store
175 MBps
150 MBps
125 MBps
100 MBps
75 MBps
50 MBps
25 MBps
0 Bps
12:46:00 12:46:30 12:47:00 12:47:30 12:48:00 12:48:30 12:49:00 12:49:30 12:50:00 12:50:30 12:51:00
150
Throughput ratio
25
20
15
10
5
0
12:45:40 12:45:50 12:46:00 12:46:10 12:46:20 12:46:30 12:46:40 12:46:50 12:47:00 12:47:10 12:47:20 12:47:30 12:47:40 12:47:50 12:48:00 12:48:10 12:48:20 12:48:30 12:48:40 12:48:50 12:49:00 12:49:10 12:49:20 12:49:30 12:49:40 12:49:50 12:50:00 12:50:10 12:50:20 12:50:30 12:50:40 12:50:50 12:51:00 12:51:10


of 20.21s total
runtime
goexit
asm_amd64.s
0 of 19.22s (95.10%)
9.39s
7.70s
8.15s
github
com/oklog/prototype/pkg/store
(*Compacter)
compact
compact.go
0 of 7.70s (38.10%)
github
com/oklog/prototype/pkg/store
mergeRecords
merge.go
0.12s (0.59%)
of 9.83s (48.64%)
1.68
github
com/oklog/prototype/pkg/store
(*Compacter)
compact
compact.go
0 of 7.70s (38.10%)
github
com/oklog/prototype/pkg/store
mergeRecords
merge.go
0.12s (0.59%)
of 9.83s (48.64%)

Optimizing queries

```
store1 ~ du -hs data
21.2G    data

store1 ~ for i in (seq 1 5)
  time rg --count 'ABCD' data >/dev/null
 end
26.66user ...
21.40user ...
18.40user ...
18.85user ...
19.01user ...

store1 ~ rg 'ABCD' data | wc -l
20287
```

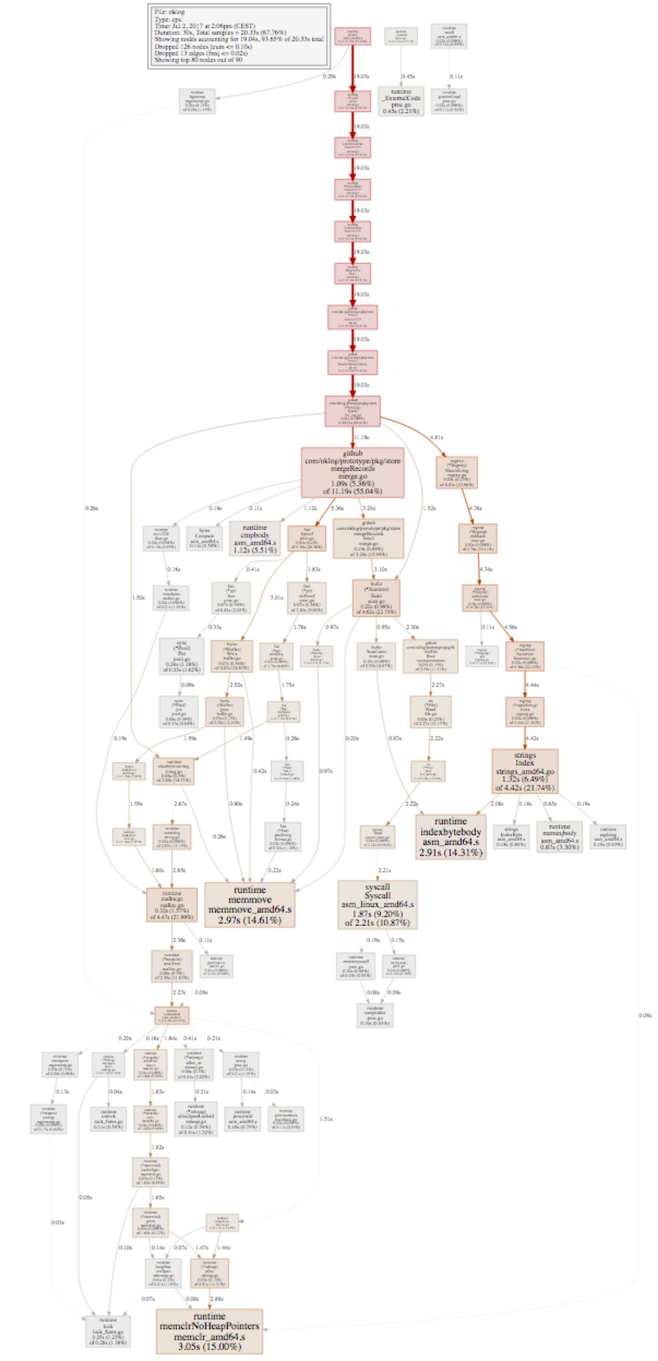
mean = 20.86s

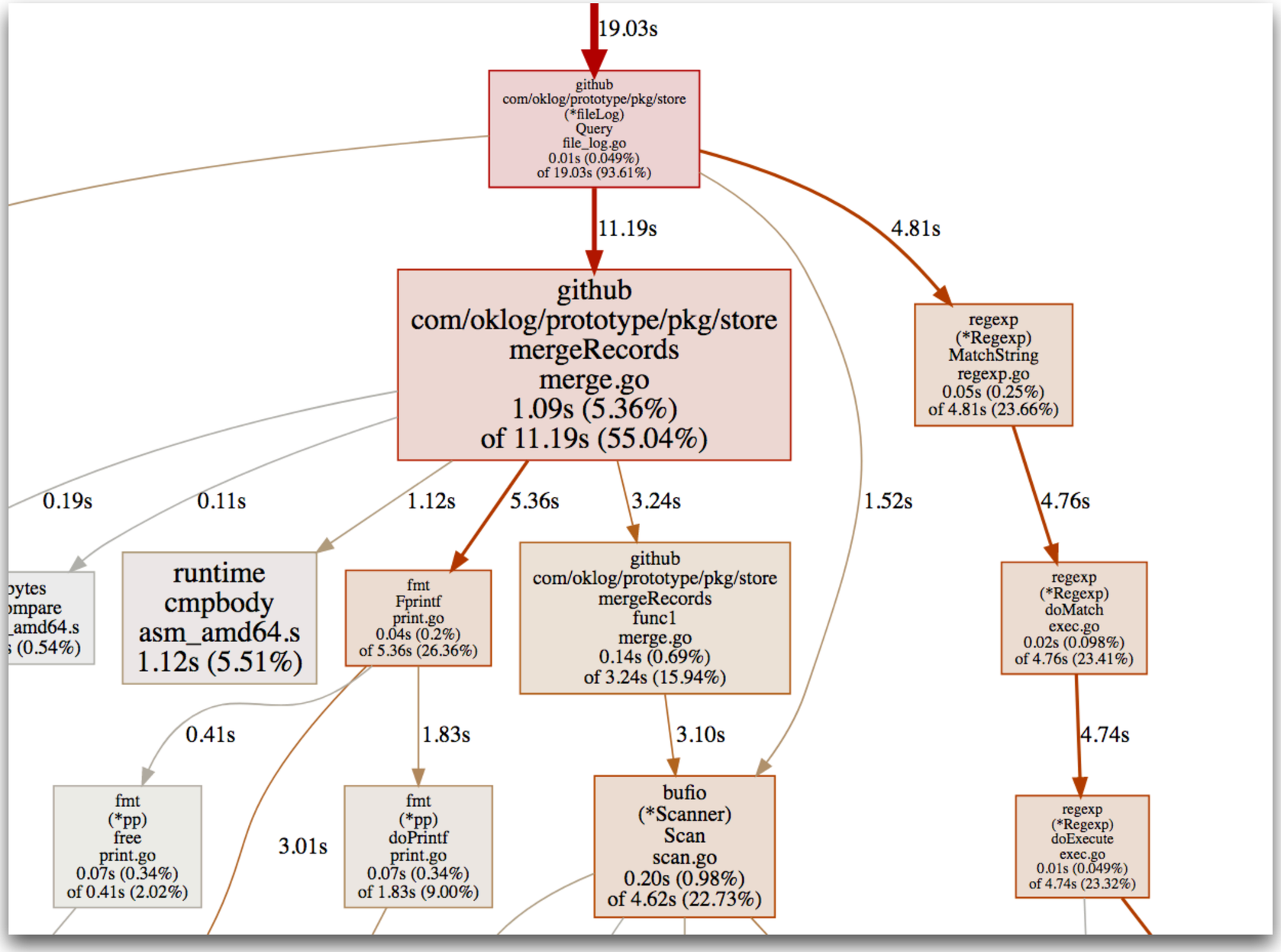
⇒ ~1.01 GBps

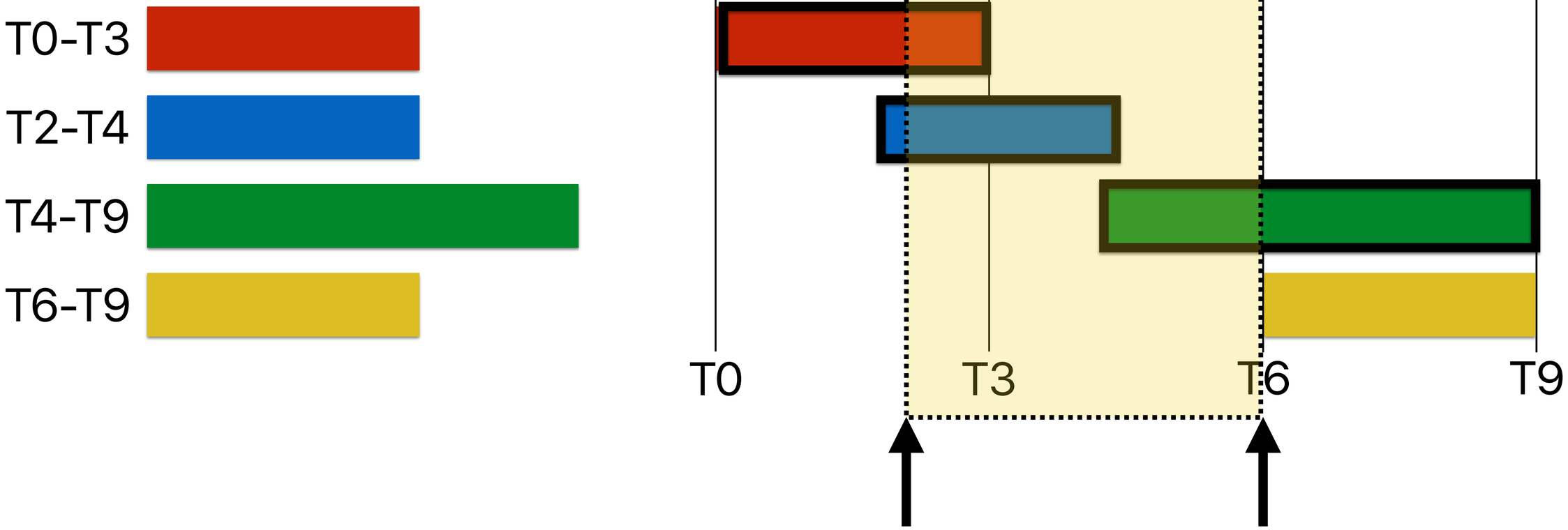
```
store1 ~ for i in (seq 1 5)
  time oklog query -q ABCD >/dev/null
 end
1:14.01user ...
1:05.51user ...
1:11.87user ...
1:12.07user ...
1:09.98user ...
```

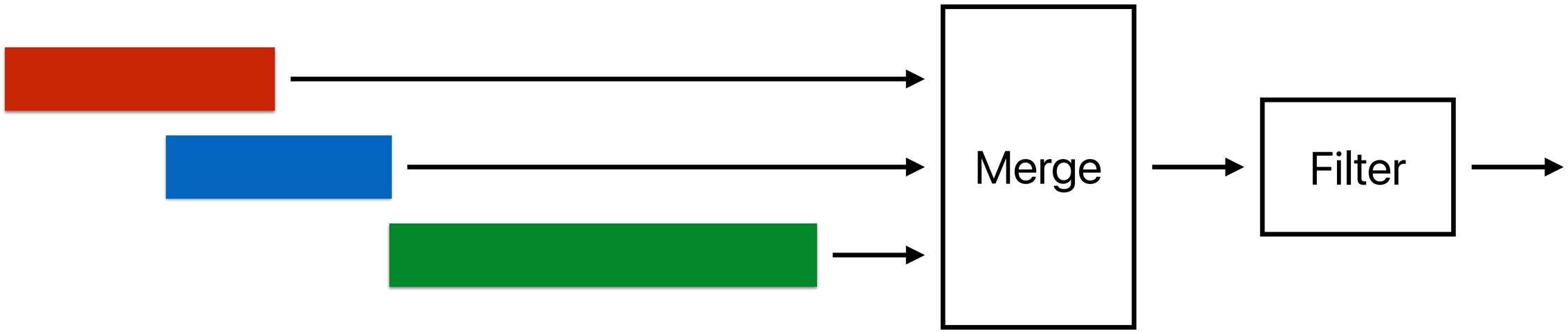
mean = 1m10s

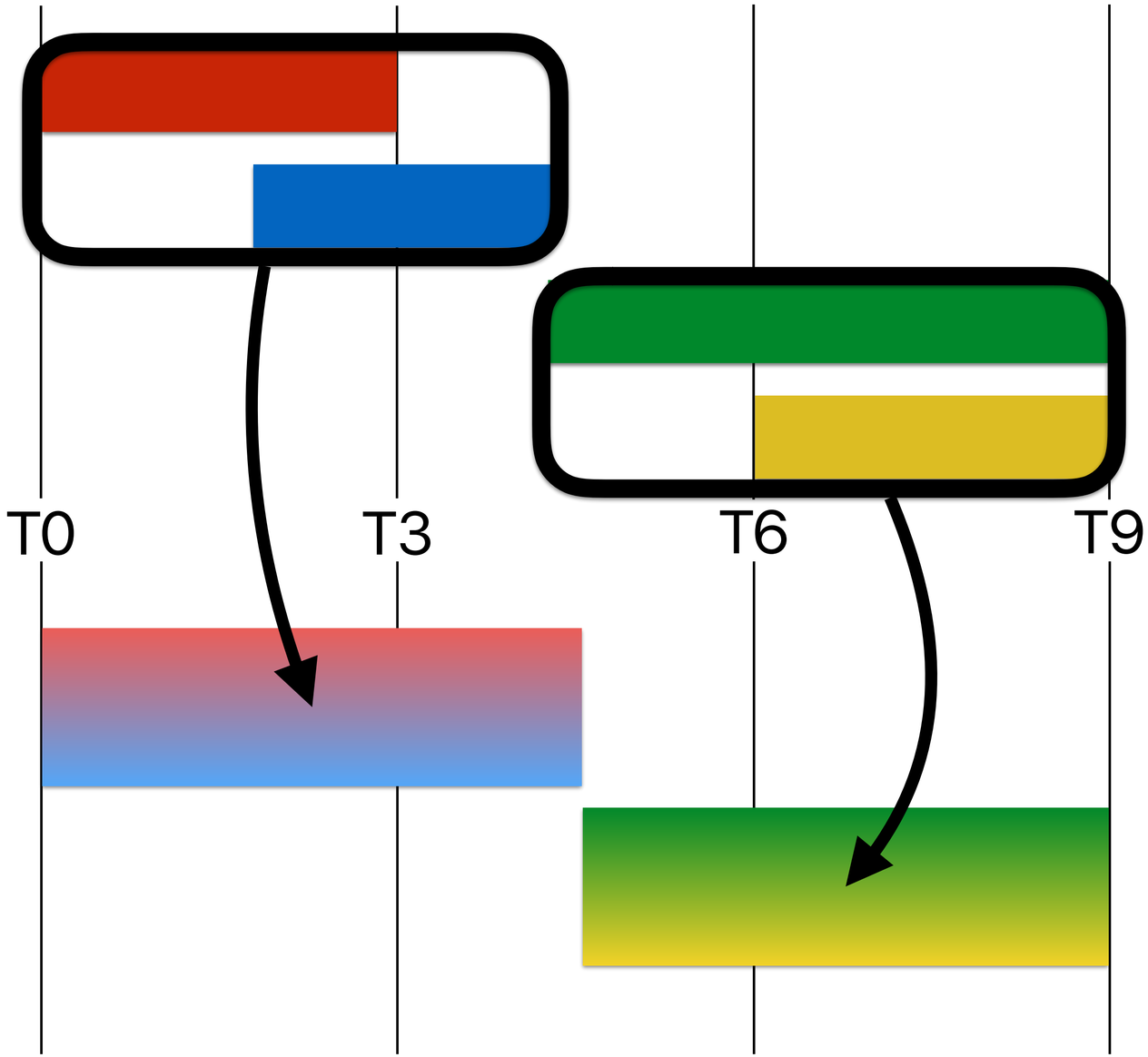
⇒ ~299 MBps

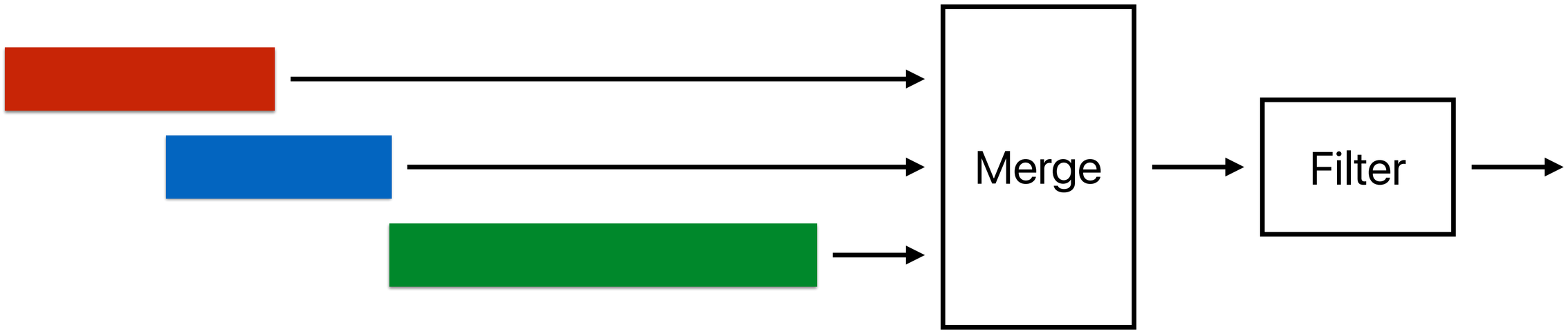


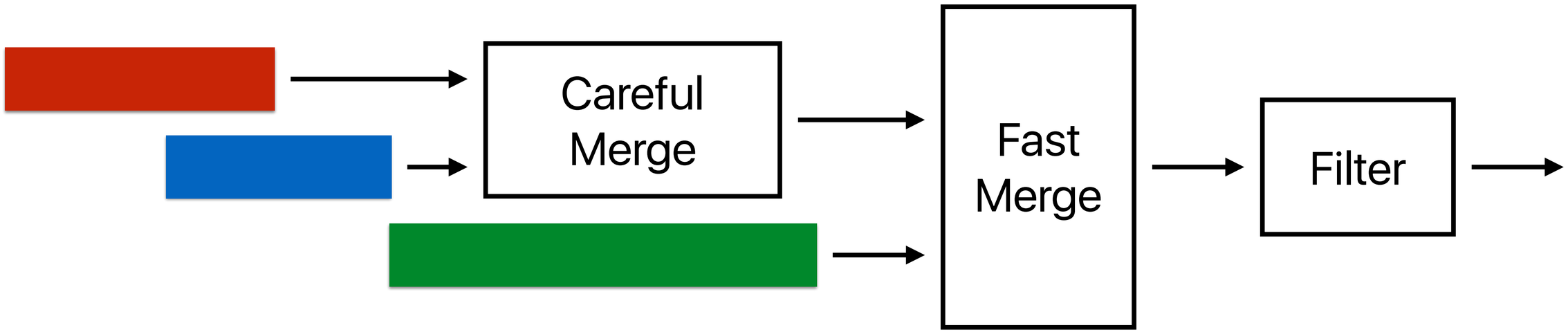

19.03s
github
com/oklog/prototype/pkg/store
(*fileLog)
Query
file_log.go
0.01s (0.049%)
of 19.03s (93.61%)
11.19s
4.81s
github
com/oklog/prototype/pkg/store
mergeRecords
merge.go
1.09s (5.36%)
of 11.19s (55.04%)
regexp
(*Regexp)
MatchString
regexp.go
0.05s (0.25%)
of 4.81s (23.66%)
0.19s
0.11s
1.12s
5.36s
3.24s
1.52s
4.76s
runtime
cmpbody
asm_amd64.s
1.12s (5.51%)
fmt
Fprintf
print.go
0.04s (0.2%)
of 5.36s (26.36%)
github
com/oklog/prototype/pkg/store
mergeRecords
func1
merge.go
0.14s (0.69%)
of 3.24s (15.94%)
regexp
(*Regexp)
doMatch
exec.go
0.02s (0.098%)
of 4.76s (23.41%)
0.41s
1.83s
3.10s
4.74s
fmt
(*pp)
free
print.go
0.07s (0.34%)
of 0.41s (2.02%)
3.01s
fmt
(*pp)
doPrintf
print.go
0.07s (0.34%)
of 1.83s (9.00%)
bufio
(*Scanner)
Scan
scan.go
0.20s (0.98%)
of 4.62s (22.73%)
regexp
(*Regexp)
doExecute
exec.go
0.01s (0.049%)
of 4.74s (23.32%)

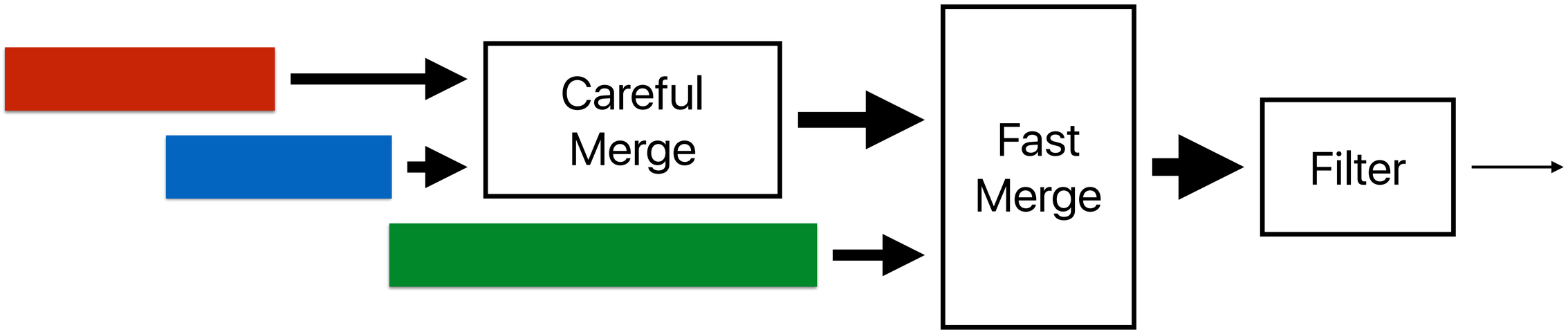

T0-T3
T2-T4
T4-T9
T6-T9
T0
T3
T6
T9

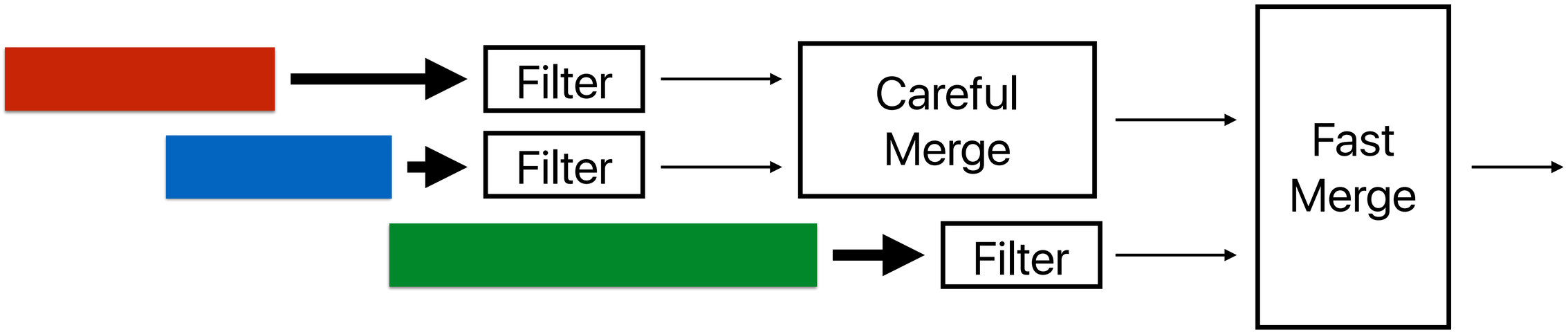

Merge
Filter

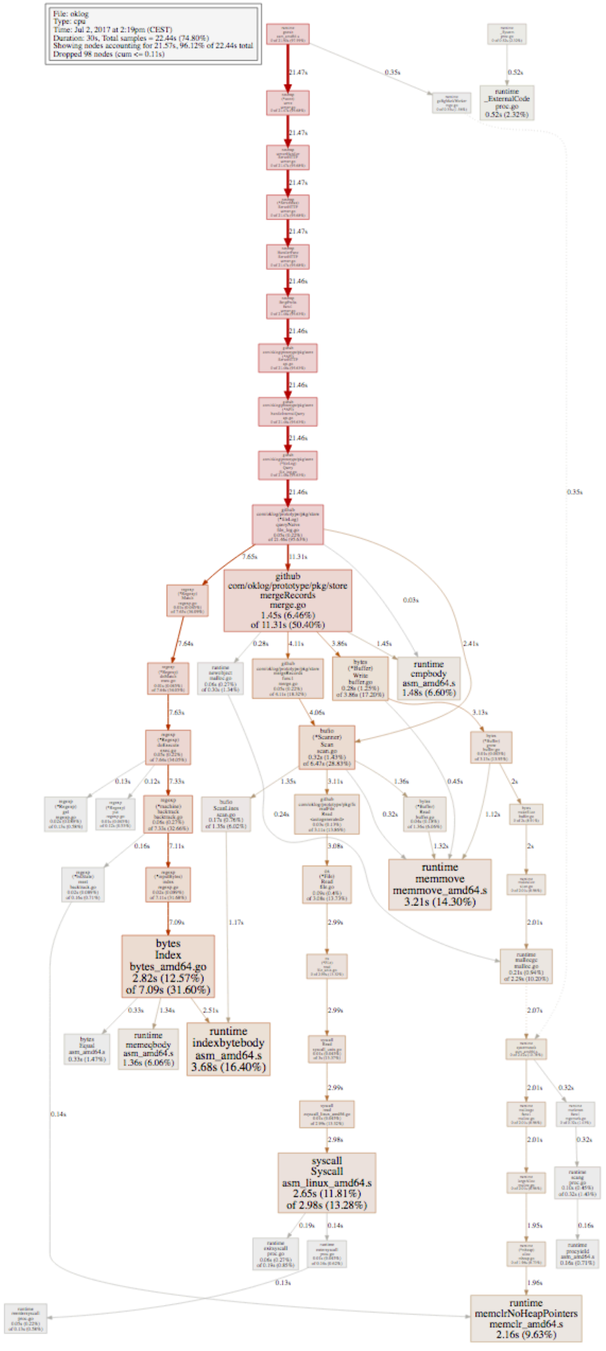

T0
T3
T6
T9

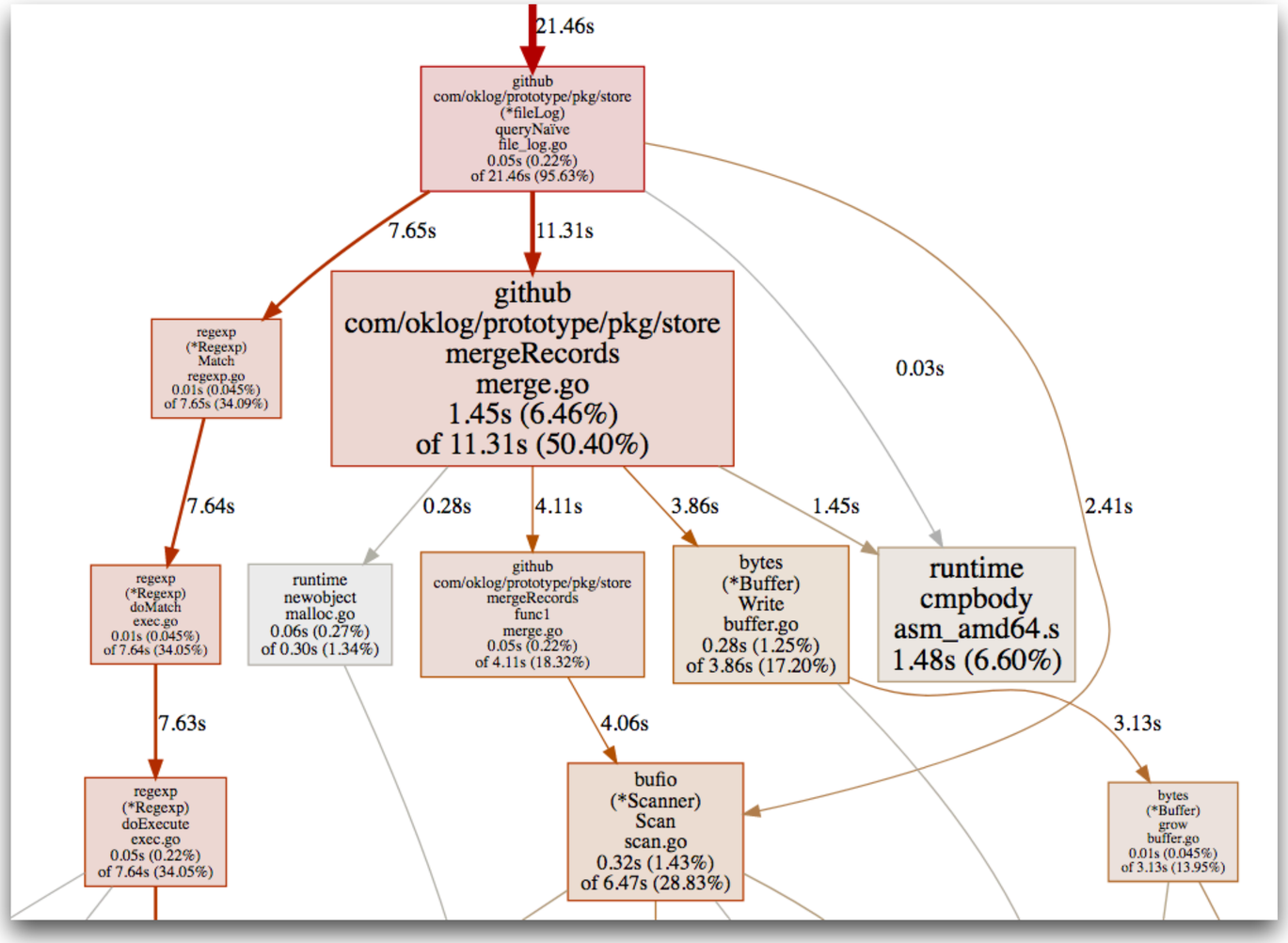

Merge
Filter

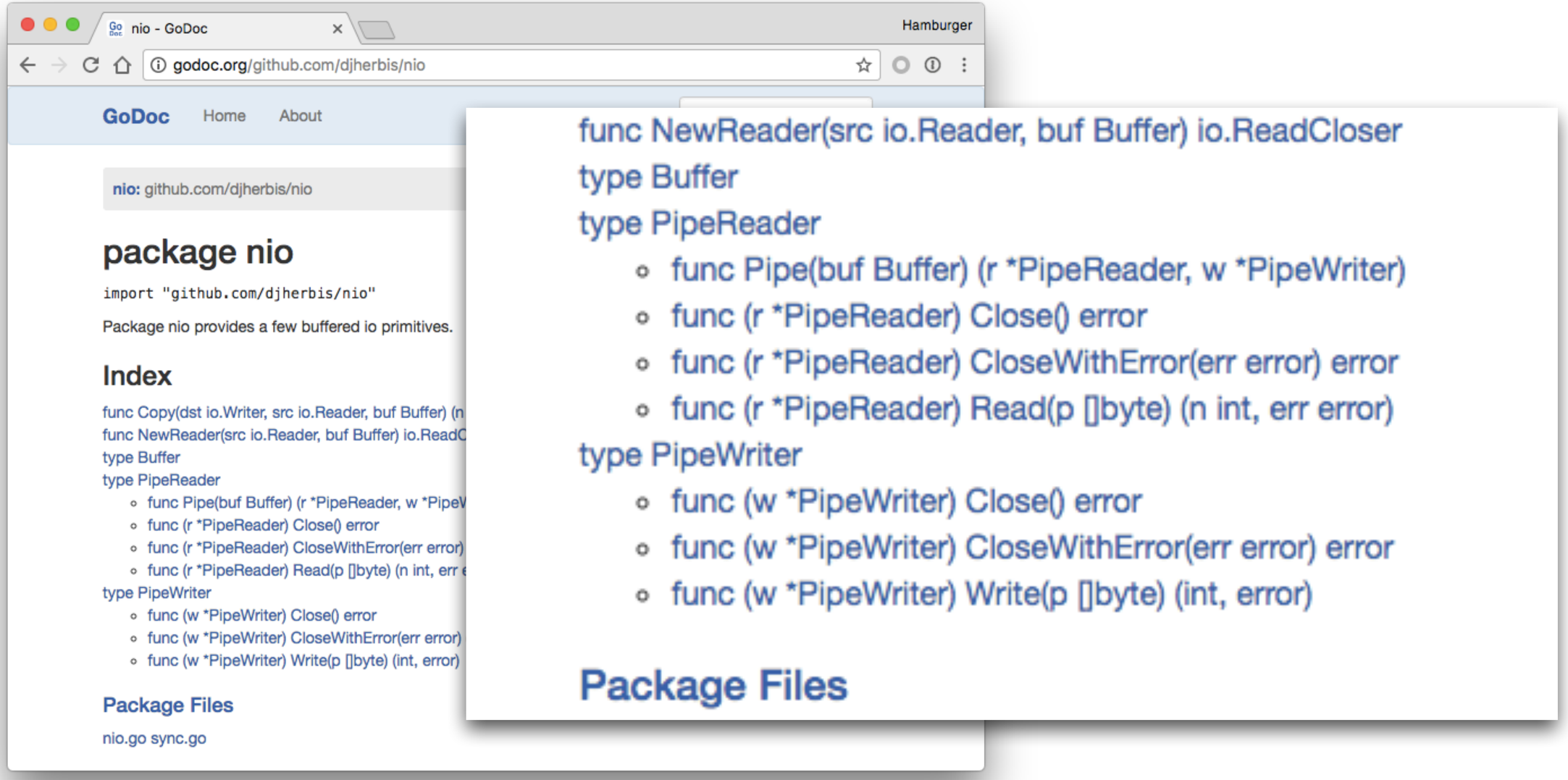

Careful
Merge
Fast
Merge
Filter


Careful
Merge
Fast
Merge
Filter

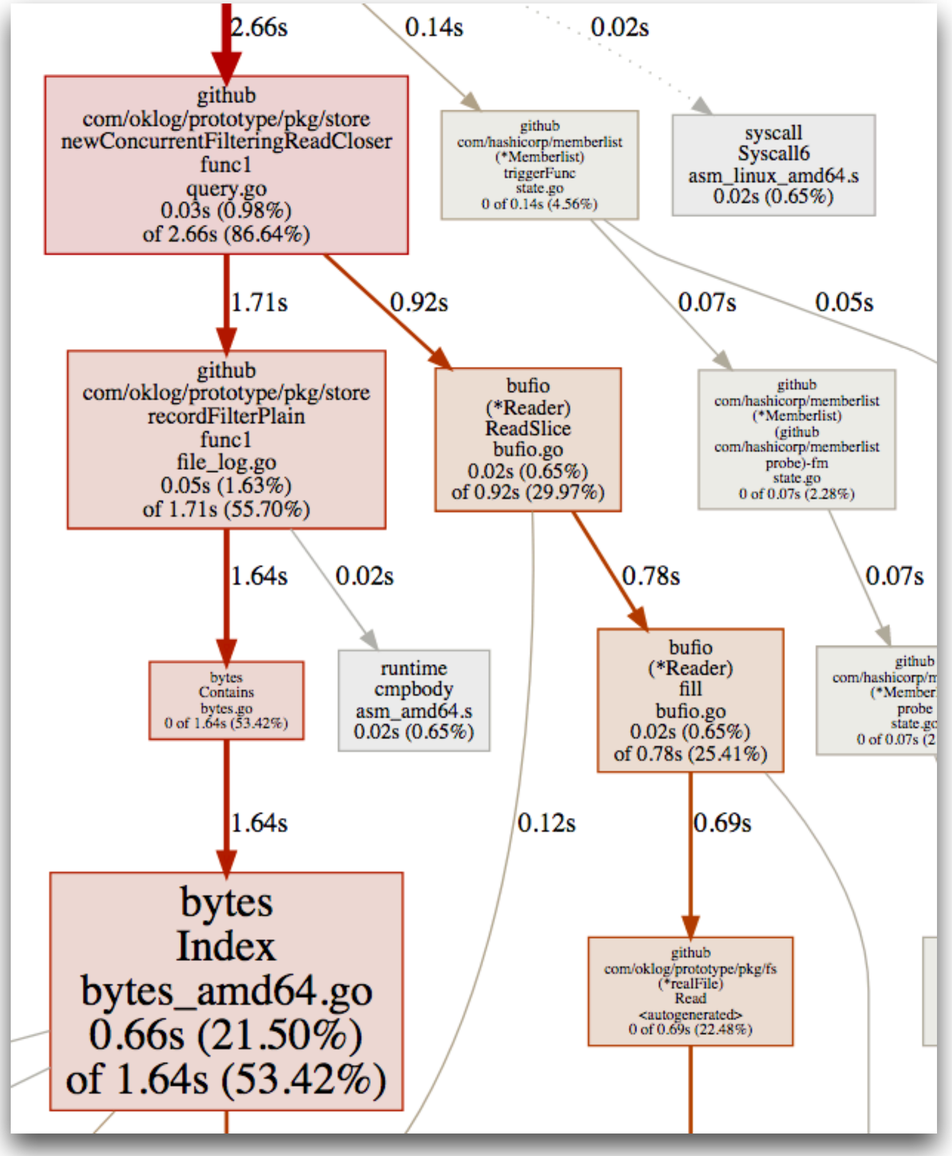

Filter
Filter
Careful
Merge
Filter
Fast
Merge


File: oklog
Type: cpu
Time: Jul 2, 2017 at 2:19pm (CEST)
Duration: 30s, Total samples = 22.44s (74.80%)
Showing nodes accounting for 21.57s, 96.12% of 22.44s total
Dropped 98 nodes (cum <= 0.11s)
runtime
_ExternalCode
proc.go
0.52s (2.32%)
github
com/oklog/prototype/pkg/store
mergeRecords
merge.go
1.45s (6.46%)
of 11.31s (50.40%)
runtime
cmpbody
asm_amd64.s
1.48s (6.60%)
bytes
(*Buffer)
Write
buffer.go
0.28s (1.25%)
of 3.86s (17.20%)
bufio
(*Scanner)
Scan
scan.go
0.32s (1.43%)
of 6.47s (28.83%)
runtime
memmove
memmove_amd64.s
3.21s (14.30%)
bytes
Index
bytes_amd64.go
2.82s (12.57%)
of 7.09s (31.60%)
runtime
indexbytebody
asm_amd64.s
3.68s (16.40%)
runtime
memeqbody
asm_amd64.s
1.36s (6.06%)
syscall
Syscall
asm_linux_amd64.s
2.65s (11.81%)
of 2.98s (13.28%)
runtime
memclrNoHeapPointers
memclr_amd64.s
2.16s (9.63%)


21.46s
github
com/oklog/prototype/pkg/store
(*fileLog)
queryNaïve
file_log.go
0.05s (0.22%)
of 21.46s (95.63%)
7.65s
11.31s
0.03s
regexp
(*Regexp)
Match
regexp.go
0.01s (0.045%)
of 7.65s (34.09%)
github
com/oklog/prototype/pkg/store
mergeRecords
merge.go
1.45s (6.46%)
of 11.31s (50.40%)
7.64s
0.28s
4.11s
3.86s
1.45s
2.41s
regexp
(*Regexp)
doMatch
exec.go
0.01s (0.045%)
of 7.64s (34.05%)
runtime
newobject
malloc.go
0.06s (0.27%)
of 0.30s (1.34%)
github
com/oklog/prototype/pkg/store
mergeRecords
func1
merge.go
0.05s (0.22%)
of 4.11s (18.32%)
bytes
(*Buffer)
Write
buffer.go
0.28s (1.25%)
of 3.86s (17.20%)
runtime
cmpbody
asm_amd64.s
1.48s (6.60%)
7.63s
4.06s
3.13s
regexp
(*Regexp)
doExecute
exec.go
0.05s (0.22%)
of 7.64s (34.05%)
bufio
(*Scanner)
Scan
scan.go
0.32s (1.43%)
of 6.47s (28.83%)
bytes
(*Buffer)
grow
buffer.go
0.01s (0.045%)
of 3.13s (13.95%)


nio - GoDoc
Hamburger
godoc.org/github.com/djherbis/nio
GoDoc
Home
About
nio: github.com/djherbis/nio
package nio
import "github.com/djherbis/nio"
Package nio provides a few buffered io primitives.
Index
func Copy(dst io.Writer, src io.Reader, buf Buffer) (n
func NewReader(src io.Reader, buf Buffer) io.ReadC
type Buffer
type PipeReader
func Pipe(buf Buffer) (r *PipeReader, w *PipeV
func (r *PipeReader) Close() error
func (r *PipeReader) CloseWithError(err error)
func (r *PipeReader) Read(p []byte) (n int, err e
type PipeWriter
func (w *PipeWriter) Close() error
func (w *PipeWriter) CloseWithError(err error)
func (w *PipeWriter) Write(p []byte) (int, error)
Package Files
nio.go sync.go
func NewReader(src io.Reader, buf Buffer) io.ReadCloser
type Buffer
type PipeReader
func Pipe(buf Buffer) (r *PipeReader, w *PipeWriter)
func (r *PipeReader) Close() error
func (r *PipeReader) CloseWithError(err error) error
func (r *PipeReader) Read(p []byte) (n int, err error)
type PipeWriter
func (w *PipeWriter) Close() error
func (w *PipeWriter) CloseWithError(err error) error
func (w *PipeWriter) Write(p []byte) (int, error)
Package Files

```
2026  // input:
2027  //    SI: data
2028  //    BX: data len
2029  //    AL: byte sought
2030  //    R8: address to put result
2031  TEXT runtime·indexbytebody(SB),NOSPLIT,$0
2032          // Shuffle X0 around so that each byte contains
2033          // the character we're looking for.
2034          MOVD AX, X0
2035          PUNPCKLBW X0, X0
2036          PUNPCKLBW X0, X0
2037          PSHUFL $0, X0, X0
2038
2039          CMPQ BX, $16
2040          JLT small
2041
2042          MOVQ SI, DI
2043
2044          CMPQ BX, $32
2045          JA avx2
2046  sse:
2047          LEAQ    -16(SI)(BX*1), AX       // AX = address of las
2048          JMP     sseloopentry
2049
2050  sseloop:
2051          // Move the next 16-byte chunk of the data into X1.
2052          MOVOU   (DI), X1
2053          // Compare bytes in X0 to X1.
2054          PCMPEQB X0, X1
2055          // Take the top bit of each byte in X1 and put the res
```



```
store1 ~ for i in (seq 1 5)
  time oklog query -q ABCD >/dev/null
 end
31.04user ...
25.41user ...
27.50user ...
23.01user ...
22.11user ...
```

mean = 25.81s

⇒ ~821 MBps

Conclusions

Think, then act

Work in cycles

NIH isn't a 4-letter word

Thanks y'all

Evolutionary Optimization
GopherCon 2017
@peterbourgon