Go and Wikipedia's CDN

Golang Users Berlin

Emanuele Rocca
Wikimedia Foundation

November 11th 2019

Logging 150K+ requests per second (and break it, and fix it)

WIKIMEDIA
FOUNDATION

Outline

- Wikimedia Foundation
- Logging Wikipedia traffic
- The Bug
- The Fix

Wikimedia Foundation

Wikimedia Foundation

- Non-profit organization focusing on free, open-content, wiki-based Internet projects
- No ads, no VC money
- Entirely funded by small donors
- 350 employees (33 SRE and 80 SWE)
- Runs the CDN that serves Wikipedia and friends

The Wikimedia Family

WIKIPEDIA
The Free Encyclopedia

Wiktionary
The free dictionary

Wikiquote

WIKIBOOKS

WIKISOURCE

WIKINEWS

WIKIVERSITY

WIKISPECIES
free species directory

MediaWiki

WIKIDATA

WIKIMEDIA
COMMONS

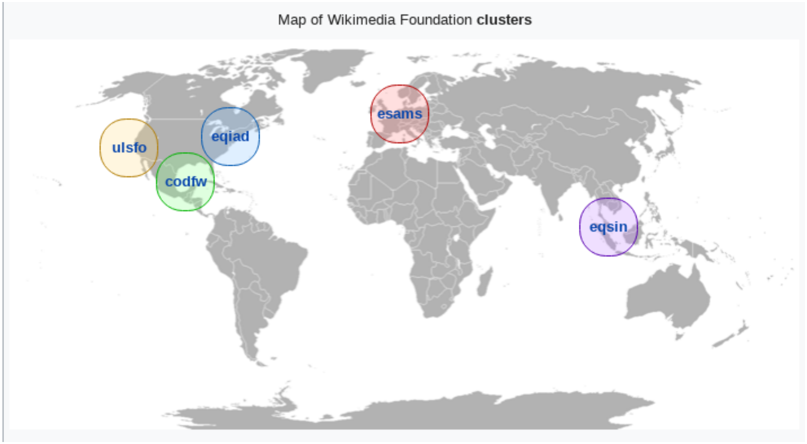
wiki
voyage

WIKIMEDIA
FOUNDATION

Build In The Open

- github.com/wikimedia
- gerrit.wikimedia.org
- phabricator.wikimedia.org
- wikitech.wikimedia.org
- grafana.wikimedia.org

Cluster Map



eqiad: Ashburn, Virginia - cp10xx
codfw: Dallas, Texas - cp20xx
esams: Amsterdam, Netherlands - cp30xx
ulsfo: San Francisco, California - cp40xx
eqsin: Singapore - cp50xx

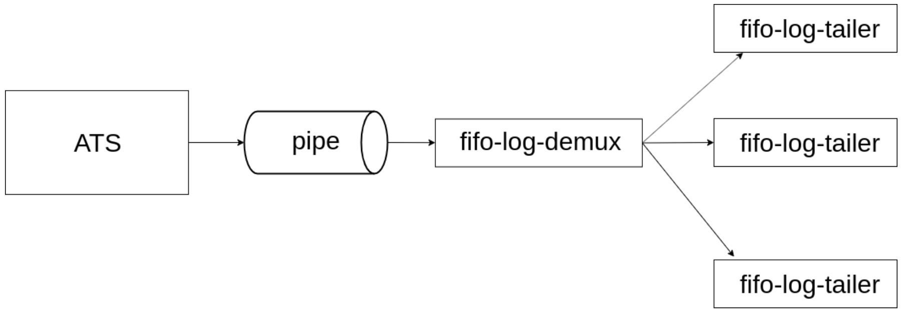
Load balancers and cache servers

- Load balancers running Linux Virtual Server
- HTTP cache proxies running Apache Traffic Server and Varnish
- TLS termination (ATS)
- In-memory transient storage (Varnish): fast, small
- On-disk persistent storage (ATS): slower, larger

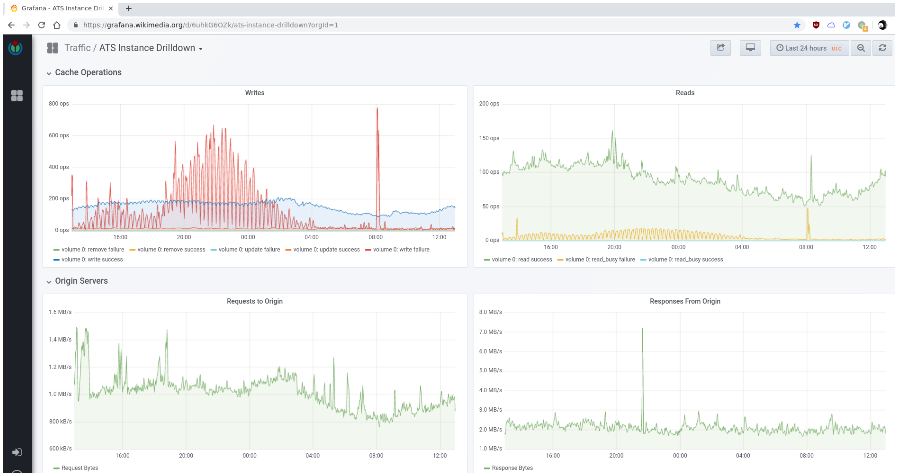
Logging loads of Traffic

Logging

- Cannot write to disk (performance, privacy)
- Logs are useful for debugging purposes
- Stats


ATS
pipe
fifo-log-demux
fifo-log-tailer
fifo-log-tailer
fifo-log-tailer

Grafana



Grafana - ATS Instance Dri
https://grafana.wikimedia.org/d/6uhkG6OZk/ats-instance-drilldown?orgId=1
Traffic / ATS Instance Drilldown
Last 24 hours UTC
Cache Operations
Writes
800 ops
600 ops
400 ops
200 ops
0 ops
16:00
20:00
00:00
04:00
08:00
12:00
volume 0: remove failure
volume 0: remove success
volume 0: update failure
volume 0: update success
volume 0: write failure
volume 0: write success
Reads
200 ops
150 ops
100 ops
50 ops
0 ops
volume 0: read success
volume 0: read_busy failure
volume 0: read_busy success
Origin Servers
Requests to Origin
1.6 MB/s
1.4 MB/s
1.2 MB/s
1.0 MB/s
800 kB/s
600 kB/s
Request Bytes
Responses From Origin
8.0 MB/s
7.0 MB/s
6.0 MB/s
5.0 MB/s
4.0 MB/s
3.0 MB/s
2.0 MB/s
1.0 MB/s
Response Bytes

fifo-log-tailer shell version

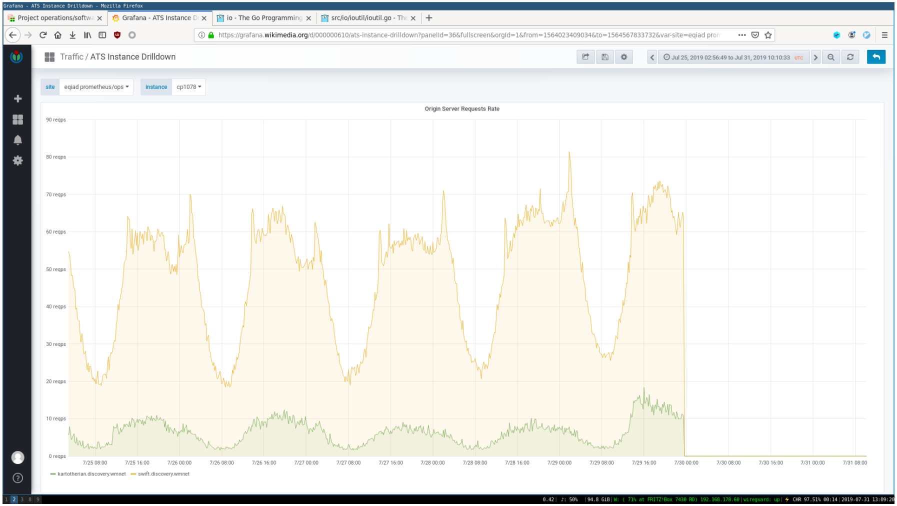
```
echo $REGEXP |
    socat UNIX-CONNECT:$LOG_SOCKET -
```

fifo-log-tailer golang version

```
// Connect to fifo-log-demux socket
c, err := net.Dial("unix", *socketPath)
[...]

// Write to standard output what is
// read from fifo-log-demux
tee := io.TeeReader(c, io.Writer(os.Stdout))
_, err = ioutil.ReadAll(tee)
```

The Bug


Grafana - ATS Instance Drilldown - Mozilla Firefox
Project operations/softwa
Grafana - ATS Instance D
io - The Go Programming
src/io/ioutil/ioutil.go - The
https://grafana.wikimedia.org/d/000000610/ats-instance-drilldown?panelId=36&fullscreen&orgId=1&from=1564023409034&to=1564567833732&var-site=eqiad prom
Traffic / ATS Instance Drilldown
Jul 25, 2019 02:56:49 to Jul 31, 2019 10:10:33 UTC
site
eqiad prometheus/ops
instance
cp1078
Origin Server Requests Rate
90 reqps
80 reqps
70 reqps
60 reqps
50 reqps
40 reqps
30 reqps
20 reqps
10 reqps
0 reqps
7/25 08:00
7/25 16:00
7/26 00:00
7/26 08:00
7/26 16:00
7/27 00:00
7/27 08:00
7/27 16:00
7/28 00:00
7/28 08:00
7/28 16:00
7/29 00:00
7/29 08:00
7/29 16:00
7/30 00:00
7/30 08:00
7/30 16:00
7/31 00:00
7/31 08:00
kartotherian.discovery.wmnet
swift.discovery.wmnet
2019-07-31 13:09:20

fifo-log-tailer crash

```
fatal error: runtime: out of memory
[...]
runtime.makeslice(0x4f6380, ...)
  /usr/lib/go-1.11/src/runtime/slice.go:70
bytes.makeSlice(0x1ffffe00, ...)
  /usr/lib/go-1.11/src/bytes/buffer.go:231
bytes.(*Buffer).grow(0xc0000f6000, ...)
  /usr/lib/go-1.11/src/bytes/buffer.go:144
bytes.(*Buffer).ReadFrom(0xc0000f6000, ...)
  /usr/lib/go-1.11/src/bytes/buffer.go:204
io/ioutil.readAll(0x540ae0, ...)
```

WIKIMEDIA FOUNDATION

pprof

```
import _ "net/http/pprof"

[...] further down in main()

go func() {
 log.Println(http.ListenAndServe(
    "localhost:6060", nil))
}()
```

pprof

```
$ curl -s localhost:6060/debug/pprof/heap > p
$ go tool pprof -top p
```

```
3.19MB of 3.19MB total (  100%)
Dropped 15 nodes (cum <= 0.02MB)
   flat  flat%   sum%        cum   cum%
 3.19MB   100%   100%     3.19MB   100%  bytes.makeSlice
      0     0%   100%     3.19MB   100%  bytes.(*Buffer).ReadFrom
      0     0%   100%     3.19MB   100%  bytes.(*Buffer).grow
      0     0%   100%     3.19MB   100%  io/ioutil.ReadAll
      0     0%   100%     3.19MB   100%  io/ioutil.readAll
      0     0%   100%     3.19MB   100%  main.main
      0     0%   100%     3.19MB   100%  runtime.main
```

pprof

```
$ while true; do curl -s localhost:6060/debug/pprof/heap > p ;
  go tool pprof -top p | grep total | ts; sleep 60; done

Jul 31 14:46:53 1712.56kB of 1712.56kB total (  100%)
Jul 31 14:47:53 3.19MB of 3.19MB total (  100%)
Jul 31 14:48:53 6.04MB of 6.04MB total (  100%)
Jul 31 14:49:53 6.04MB of 6.04MB total (  100%)
Jul 31 14:50:53 12MB of 12MB total (  100%)
```

ioutil.ReadAll

- Reads until EOF and returns the data it read using an internal buffer
- Uses bytes.Buffer.ReadFrom(), which appends to a buffer and grows it as needed
- What was I thinking

Runtime crash

The string "runtime: out of memory" comes from src/runtime/mem_linux.go:

```
func sysMap(v unsafe.Pointer, n uintptr, sysStat *uint64) {
    mSysStatInc(sysStat, n)

    p, err := mmap(v, n, _PROT_READ|_PROT_WRITE, ...)
    if err == _ENOMEM {
        throw("runtime: out of memory")
    }
    [...]
}
```

SystemTap runtime.sysMap

```
$ stap -L 'process("fifo-log-tailer").function("*sysMap*")'
process("fifo-log-tailer").function("runtime.sysMap@[...]
    /usr/lib/src/runtime/mem_linux.go:165")
        $v:void* $n:uintptr $sysStat:uint64*

$ stap -e 'probe process("fifo-log-demux").function(
    "runtime.sysMap") { printf("size=%d\n", $n) }'
size=67108864
size=134217728
size=268435456
[...]
size=8589934592
size=17179869184
```

BOOM, my laptop has 16G of memory

Buffer.grow

- ioutil.ReadAll uses bytes.(*Buffer).grow if needed
- grow allocates double the previously allocated space with makeSlice(2*c + n)
- Now we know why the OOM killer never shot anything
- Doubling the buffer size perhaps not always the best strategy?

The Fix

```
commit ddfce42ad4a549fdeb699572e35006e9b79896fc
Author: Emanuele Rocca <ema@wikimedia.org>
Date:   Wed Jul 31 15:33:49 2019 +0200

 0.4: do not use ioutil.ReadAll() in fifo-log-tailer

 Instead of using io.TeeReader and ioutil.ReadAll, which keeps
 on allocating memory forever, just use io.CopyBuffer.

 Bug: T229414
```

nope

```
tee := io.TeeReader(c, io.Writer(os.Stdout))
_, err = ioutil.ReadAll(tee)
```

yep

```
_, err = io.CopyBuffer(io.Writer(os.Stdout),
                       c, buf)
```

pprof

```
$ while true; do curl -s localhost:6060/debug/pprof/heap > p ;
  go tool pprof -top p | grep total | ts; sleep 60; done

Jul 31 14:40:28 1485.59kB of 1485.59kB total (  100%)
Jul 31 14:41:28 1485.59kB of 1485.59kB total (  100%)
Jul 31 14:42:28 1553.21kB of 1553.21kB total (  100%)
Jul 31 14:43:28 902.59kB of 902.59kB total (  100%)
Jul 31 14:44:28 902.59kB of 902.59kB total (  100%)
```

Conclusions

Conclusions

- Standard library functions are great, but consider their implementation
- The Go runtime exists and does things
- pprof is very useful and simple to use
- SystemTap likely the best debugging tool in the world?