How many dashboards does your Grafana have?

Martin Polednik
Head of DevOps @ ThreatMark

Me

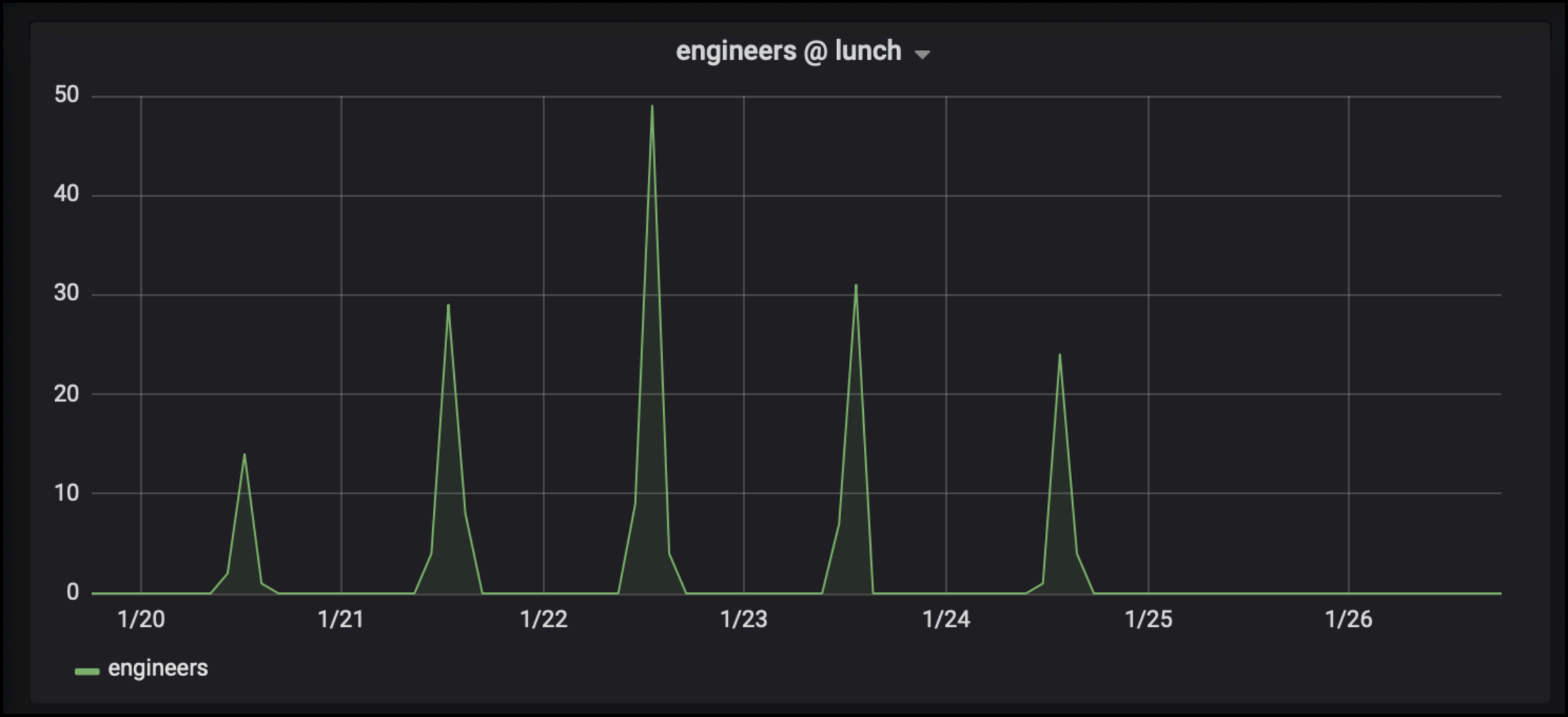
- Former Software Engineer (Infrastructure)
  - oVirt (RHEV), KubeVirt
- Enjoying the business / development / ops combination of DevOps (or SRE)
- (not only) maintaining monitoring infrastructure @ ThreatMark
- Automation & Data Geek

Why Grafana

- Prometheus is awesome
- Grafana is good fit for Prometheus
- Fits cloud native landscape
- Beautiful! (more on this later)
- But... easy to mess up

Context Matters

${startup}


engineers @ lunch
50
40
30
20
10
0
1/20
1/21
1/22
1/23
1/24
1/25
1/26
engineers

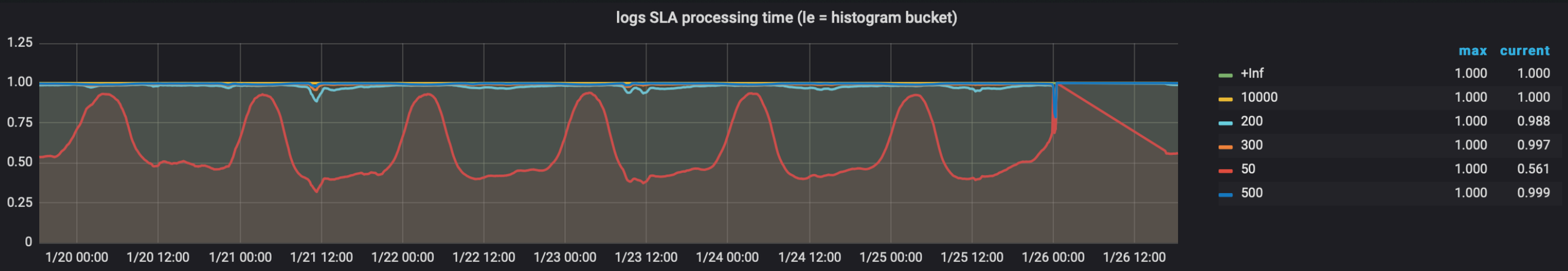
${startup}

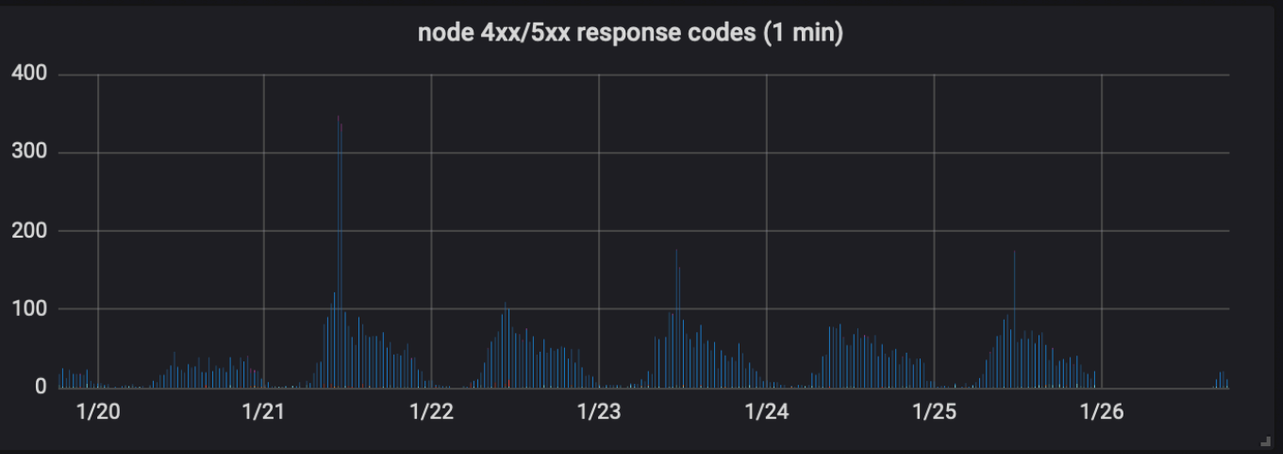
engineers @ lunch
50
40
30
20
10
0
1/20
1/21
1/22
1/23
1/24
1/25
1/26
engineers
weekend
(obviously a lie)

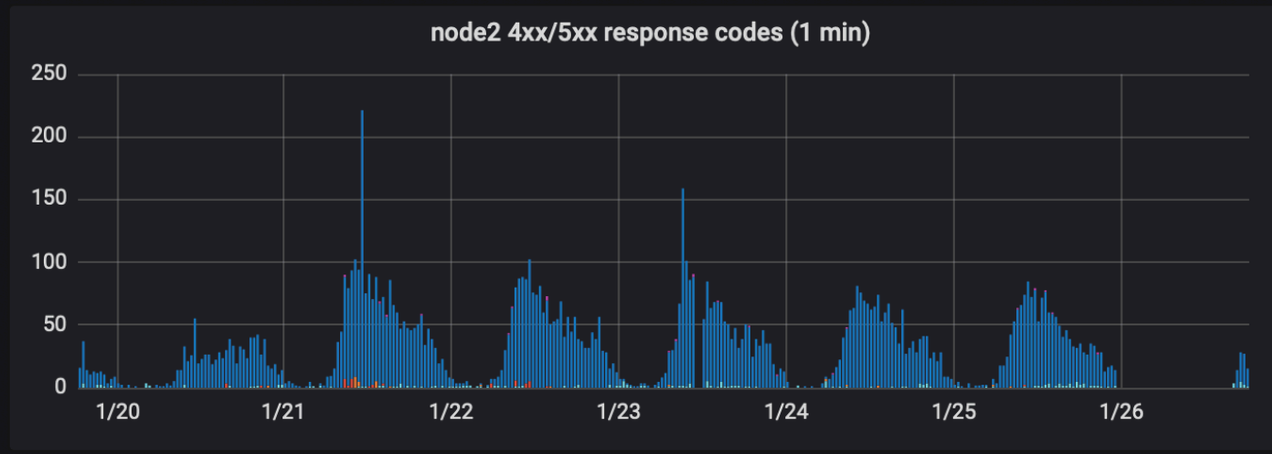
Bistro in ${tech hub}

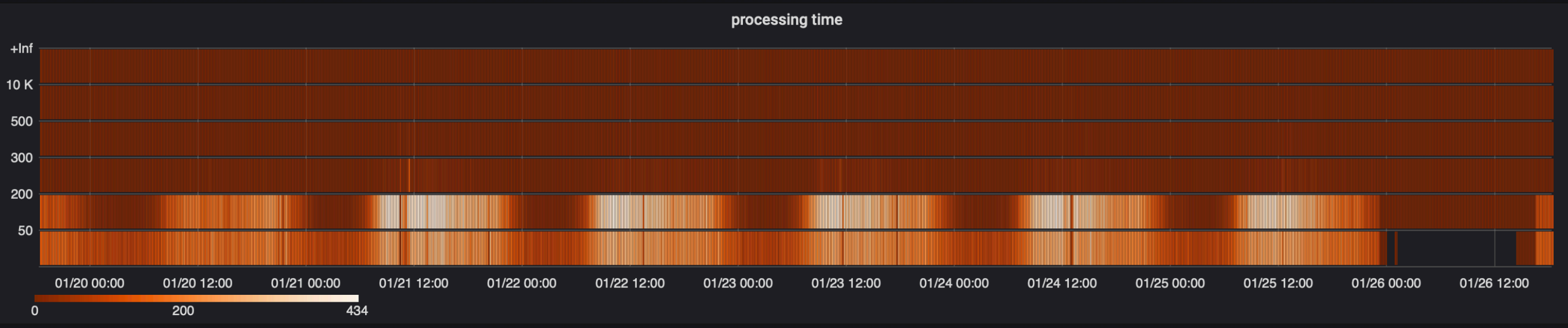
Context Matters

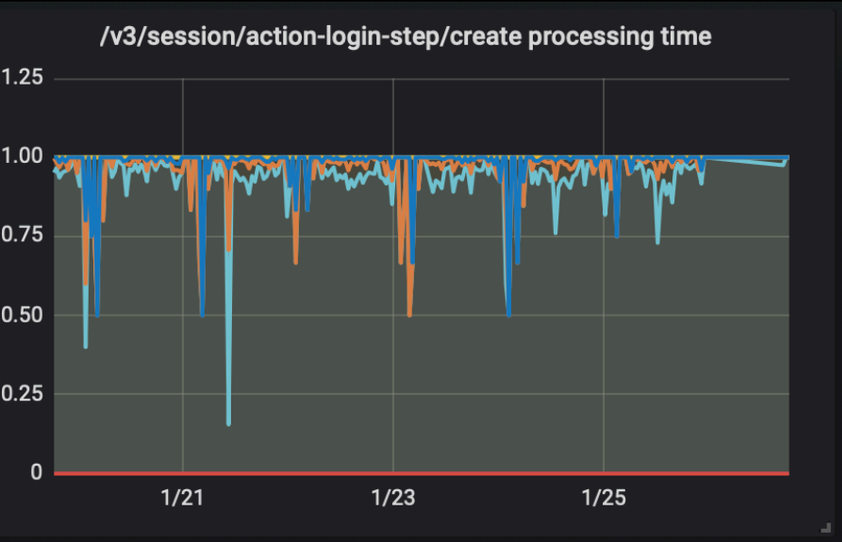
- Companies prioritize different metrics
  - Being able to function?
  - Crazy SLAs?
  - Optimizing for cost?
  - BI
- Focus on customer and business needs

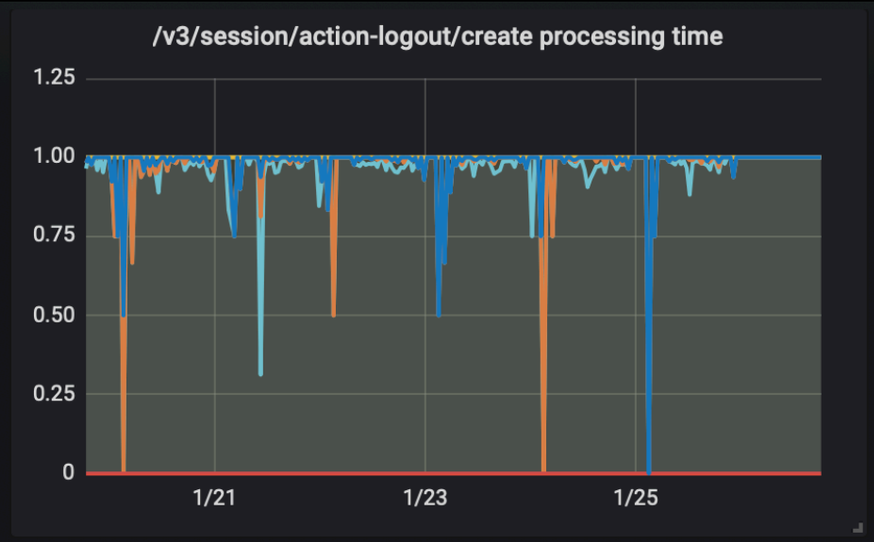

logs SLA processing time (le = histogram bucket)
1.25
1.00
0.75
0.50
0.25
0
1/20 00:00
1/20 12:00
1/21 00:00
1/21 12:00
1/22 00:00
1/22 12:00
1/23 00:00
1/23 12:00
1/24 00:00
1/24 12:00
1/25 00:00
1/25 12:00
1/26 00:00
1/26 12:00
max current
+Inf 1.000 1.000
10000 1.000 1.000
200 1.000 0.988
300 1.000 0.997
50 1.000 0.561
500 1.000 0.999

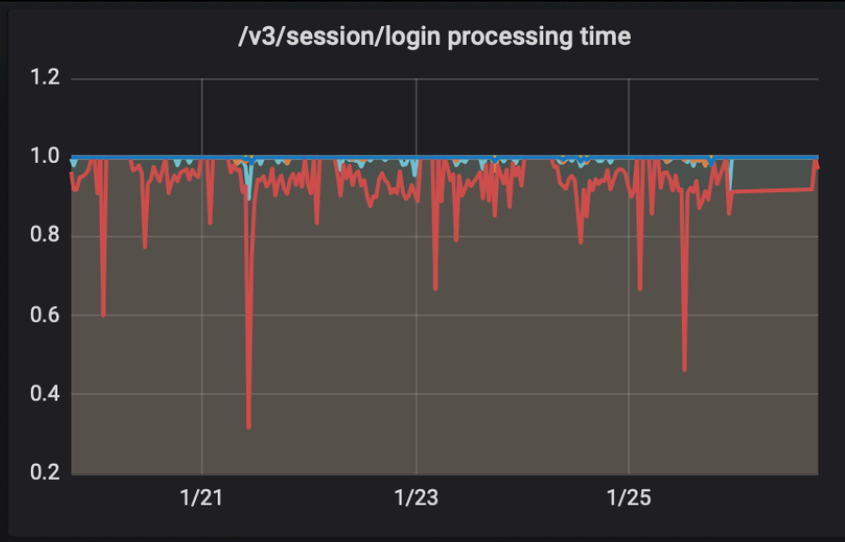

node 4xx/5xx response codes (1 min)
400
300
200
100
0
1/20
1/21
1/22
1/23
1/24
1/25
1/26

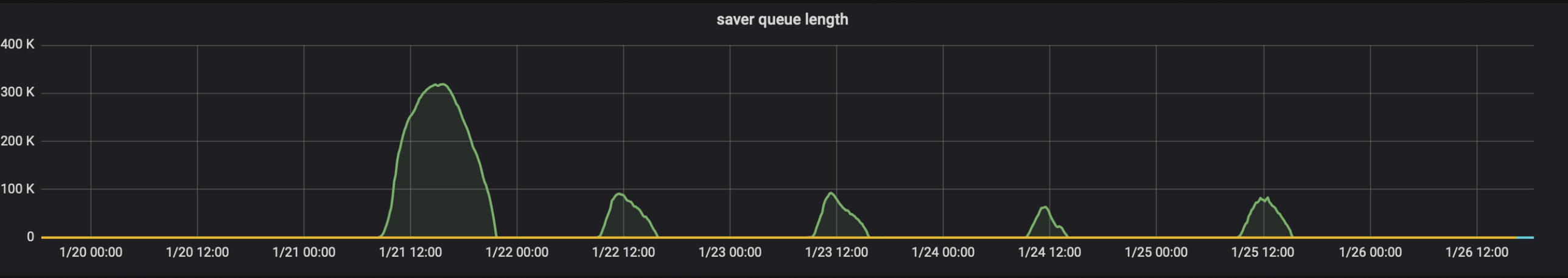

node2 4xx/5xx response codes (1 min)
250
200
150
100
50
0
1/20
1/21
1/22
1/23
1/24
1/25
1/26

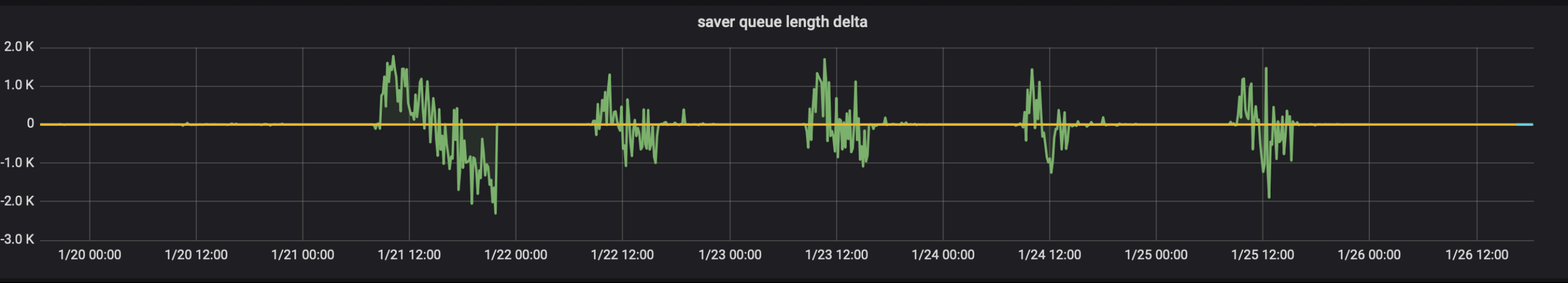

processing time
+Inf
10 K
500
300
200
50
01/20 00:00
01/20 12:00
01/21 00:00
01/21 12:00
01/22 00:00
01/22 12:00
01/23 00:00
01/23 12:00
01/24 00:00
01/24 12:00
01/25 00:00
01/25 12:00
01/26 00:00
01/26 12:00
0
200
434

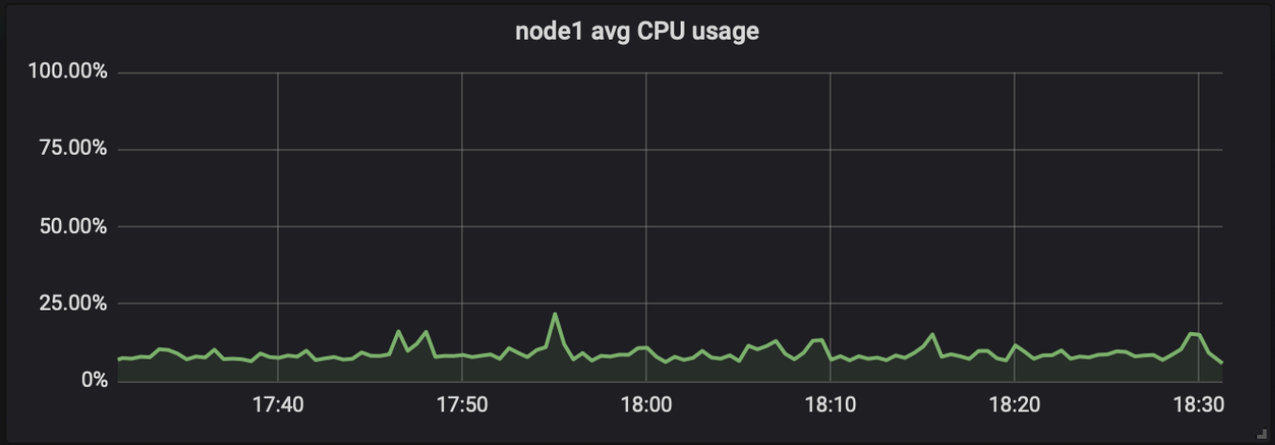

/v3/session/action-login-step/create processing time
1.25
1.00
0.75
0.50
0.25
0
1/21
1/23
1/25

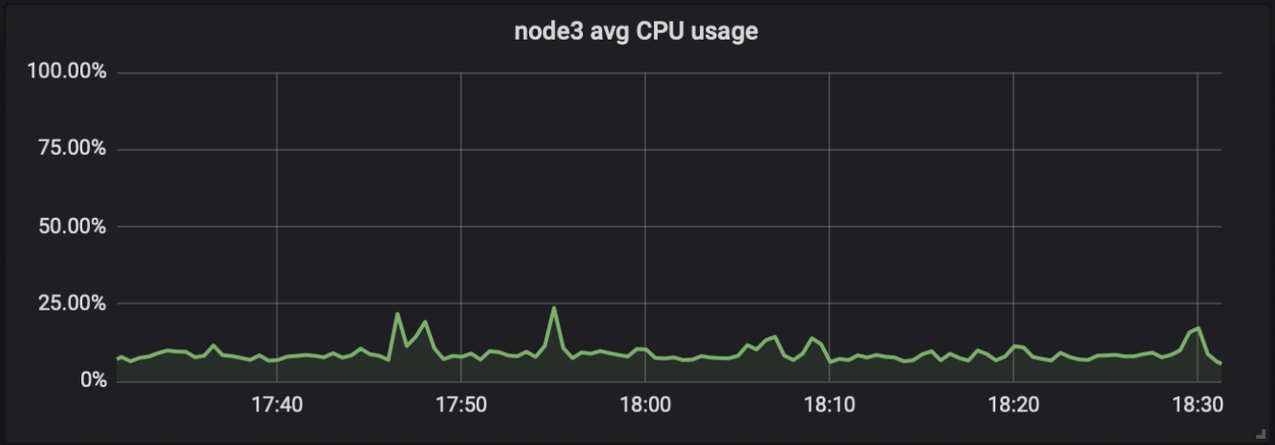

/v3/session/action-logout/create processing time
1.25
1.00
0.75
0.50
0.25
0
1/21
1/23
1/25

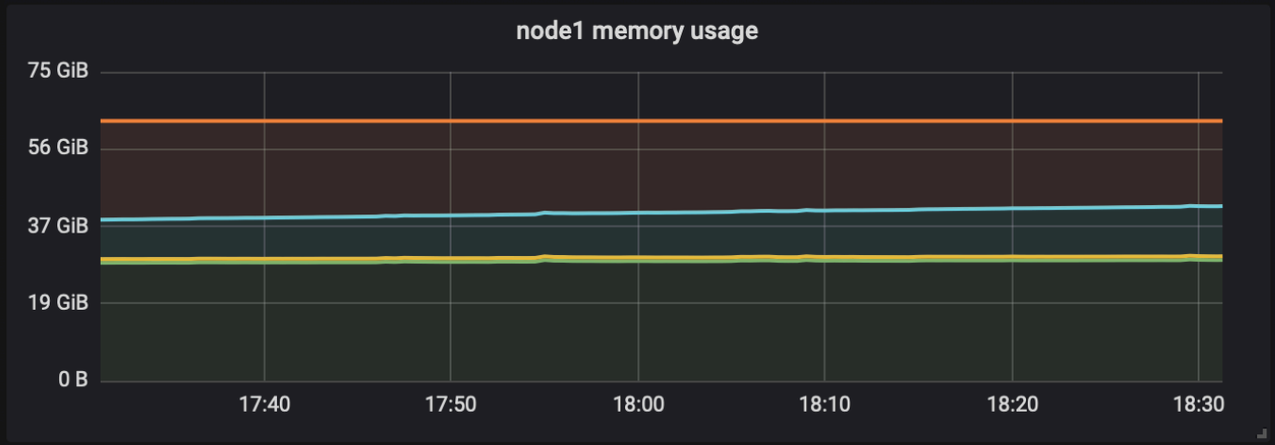

/v3/session/login processing time
1.2
1.0
0.8
0.6
0.4
0.2
1/21
1/23
1/25

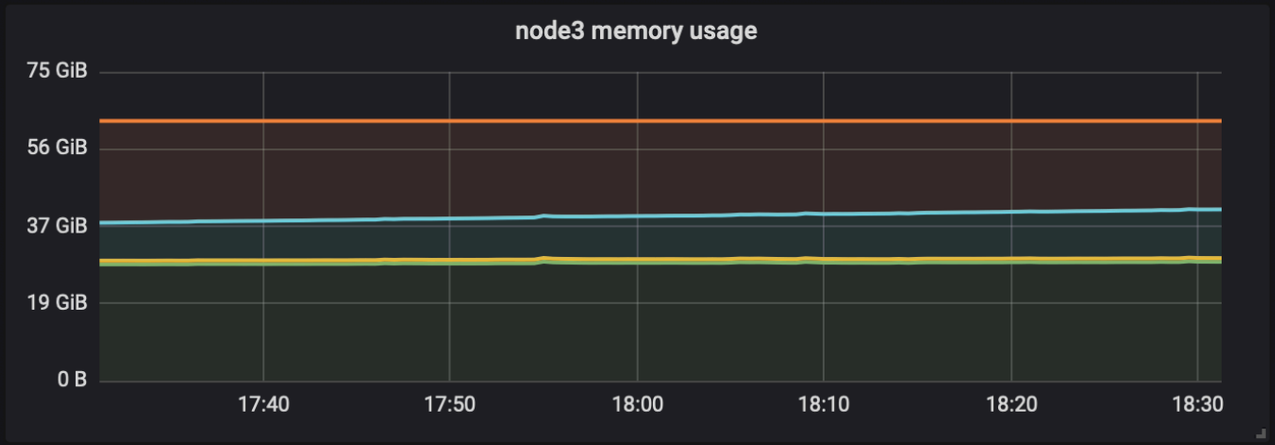

saver queue length
400 K
300 K
200 K
100 K
0
1/20 00:00
1/20 12:00
1/21 00:00
1/21 12:00
1/22 00:00
1/22 12:00
1/23 00:00
1/23 12:00
1/24 00:00
1/24 12:00
1/25 00:00
1/25 12:00
1/26 00:00
1/26 12:00

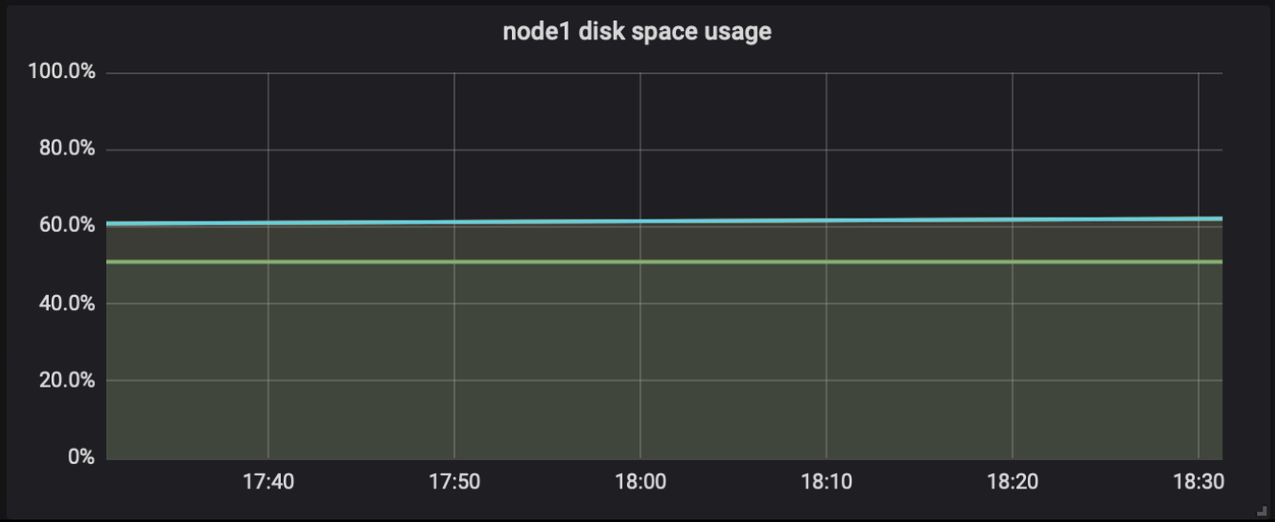

saver queue length delta
2.0 K
1.0 K
0
-1.0 K
-2.0 K
-3.0 K
1/20 00:00
1/20 12:00
1/21 00:00
1/21 12:00
1/22 00:00
1/22 12:00
1/23 00:00
1/23 12:00
1/24 00:00
1/24 12:00
1/25 00:00
1/25 12:00
1/26 00:00
1/26 12:00

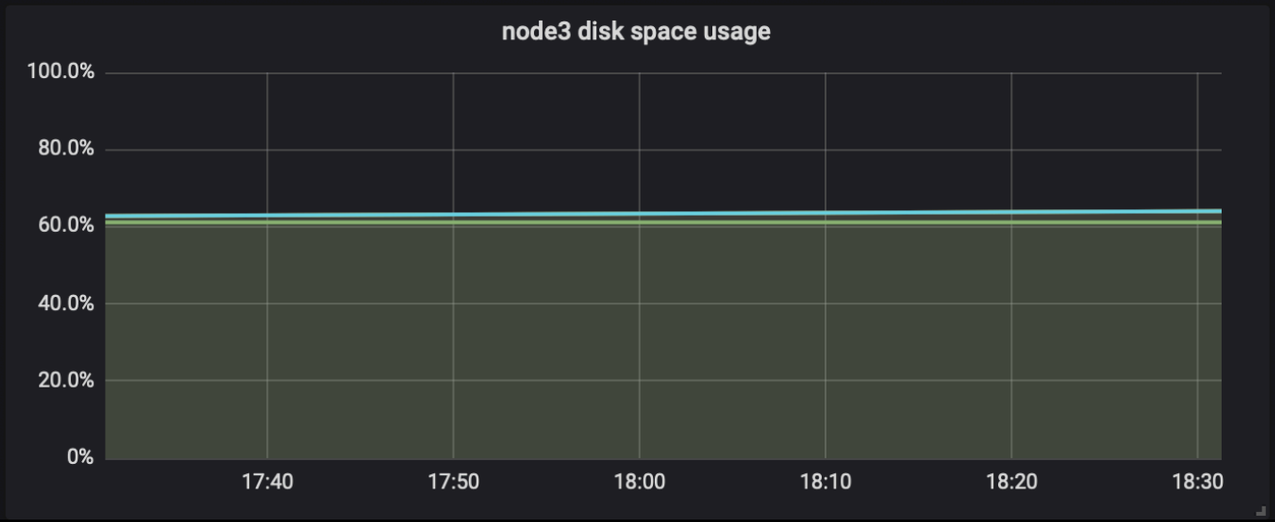

node1 avg CPU usage
100.00%
75.00%
50.00%
25.00%
0%
17:40
17:50
18:00
18:10
18:20
18:30


node3 avg CPU usage
100.00%
75.00%
50.00%
25.00%
0%
17:40
17:50
18:00
18:10
18:20
18:30

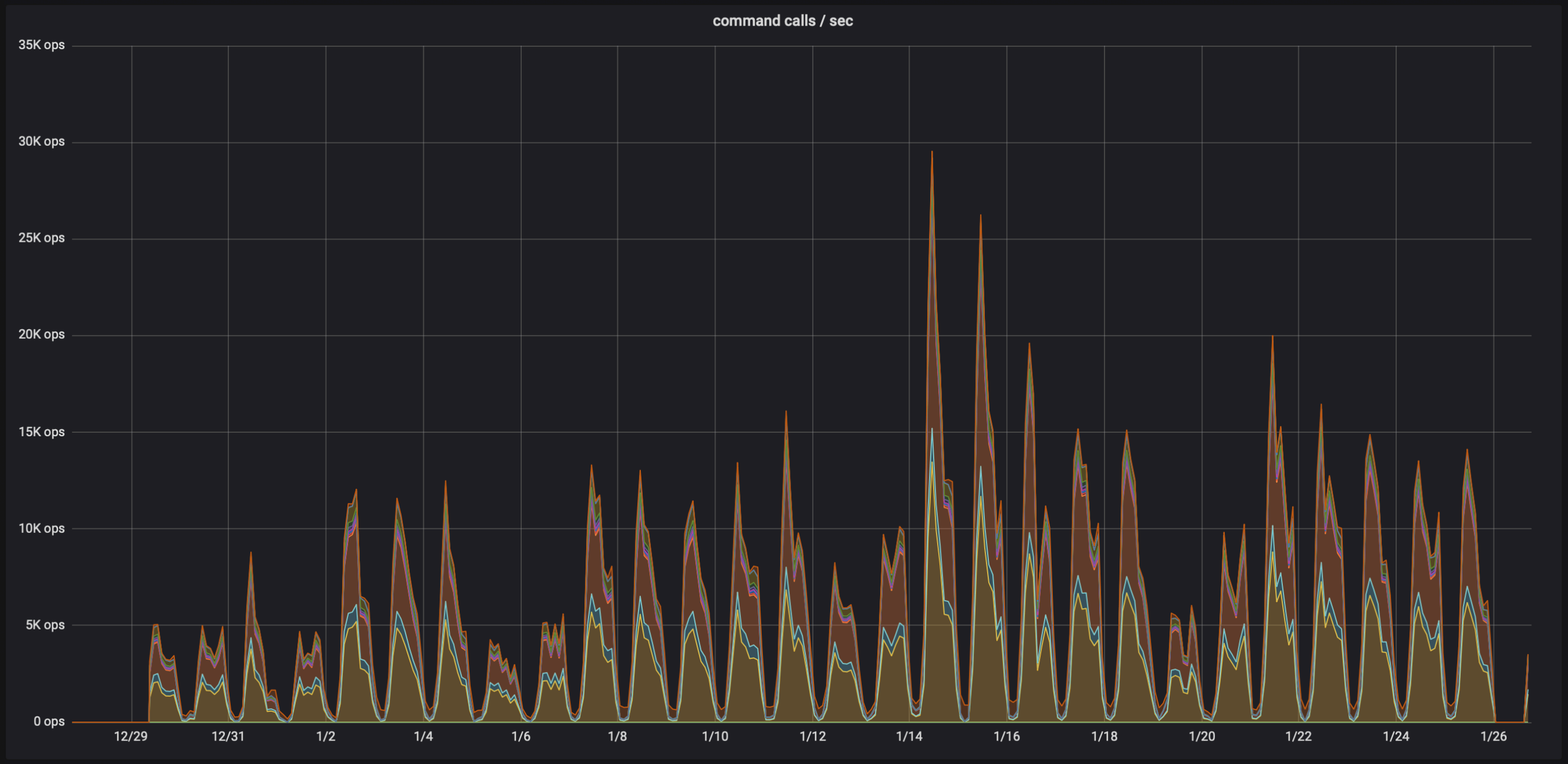

node1 memory usage
75 GiB
56 GiB
37 GiB
19 GiB
0 B
17:40
17:50
18:00
18:10
18:20
18:30

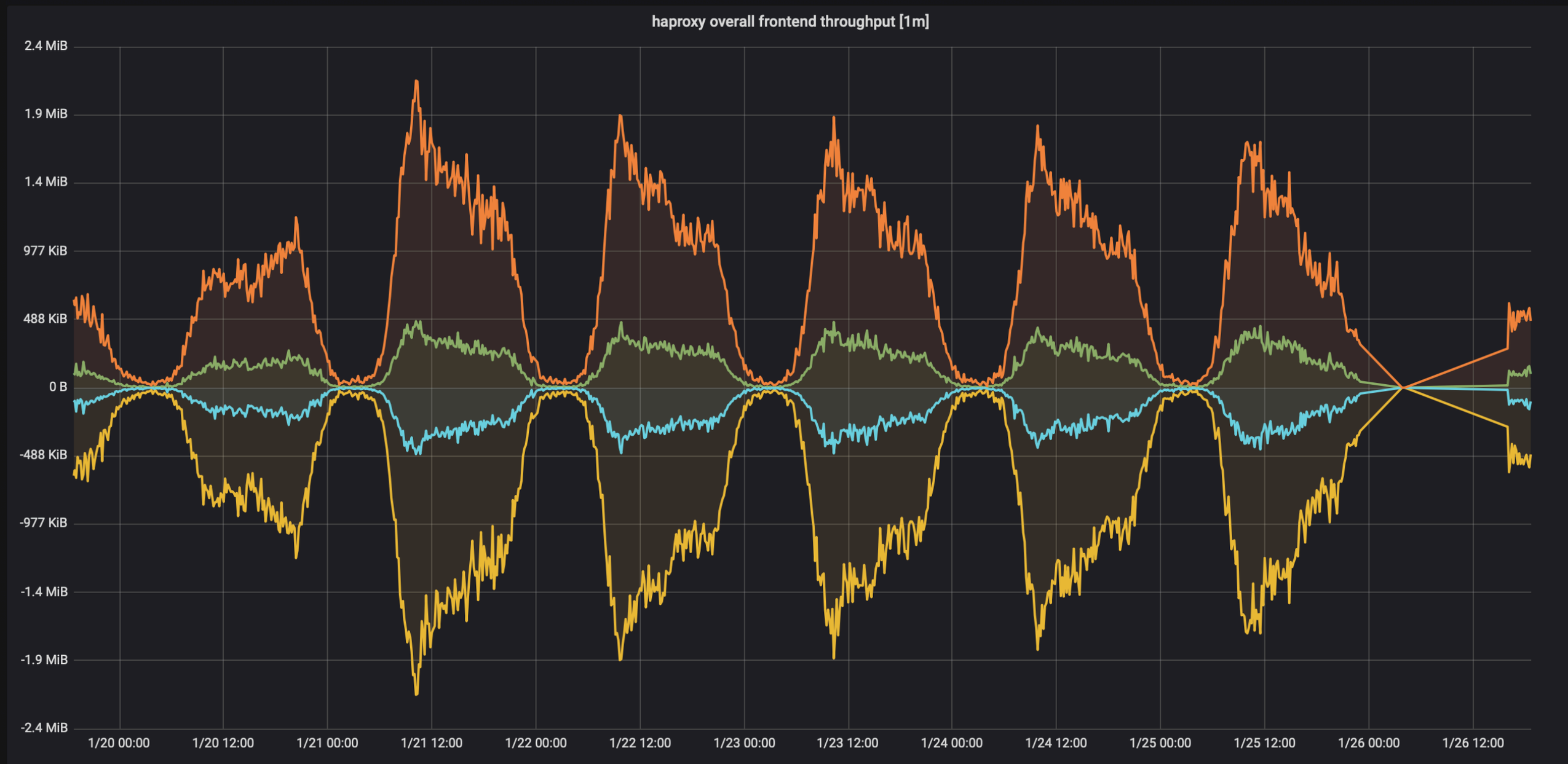

node3 memory usage
75 GiB
56 GiB
37 GiB
19 GiB
0 B
17:40
17:50
18:00
18:10
18:20
18:30

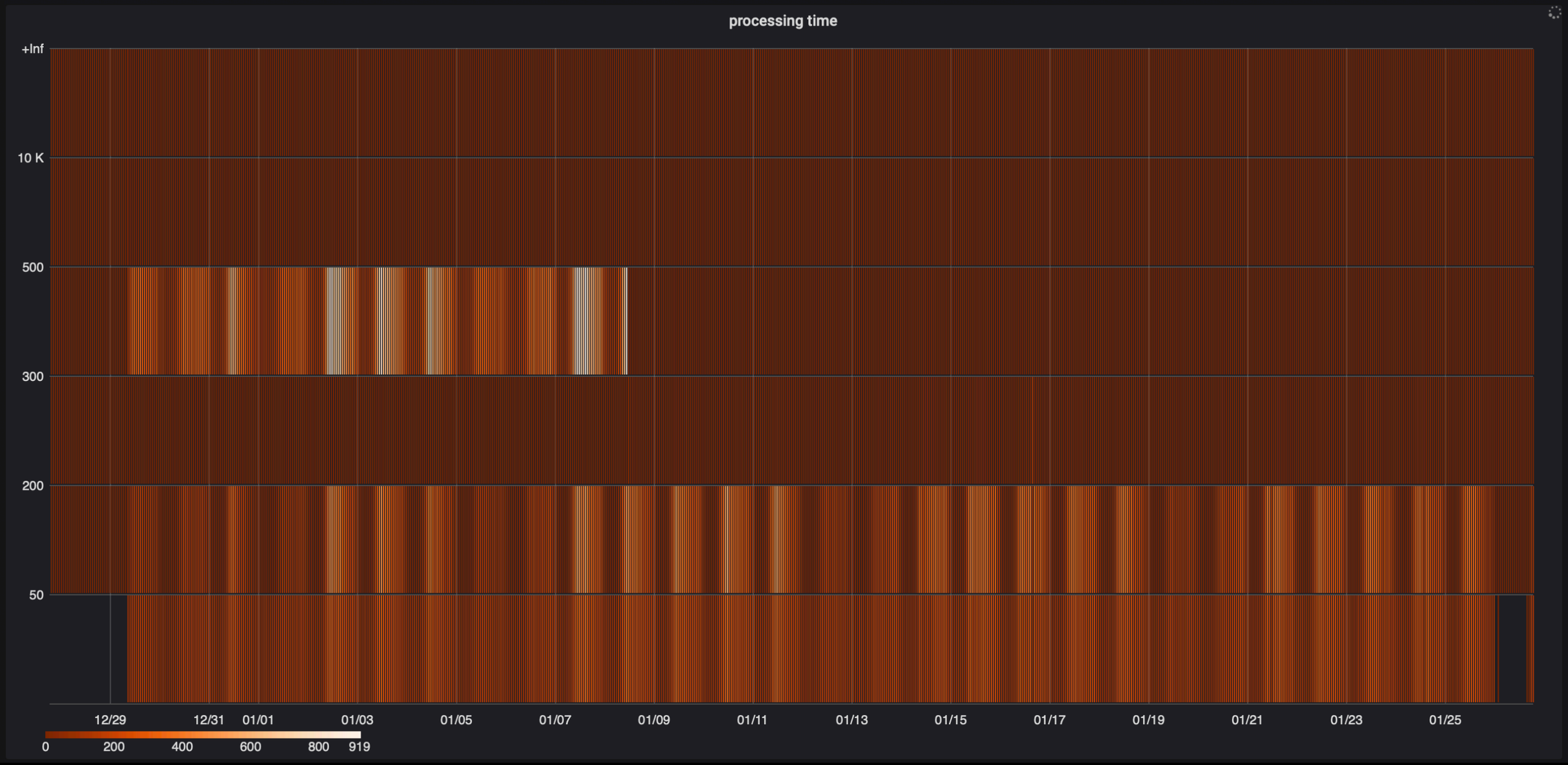

node1 disk space usage
100.0%
80.0%
60.0%
40.0%
20.0%
0%
17:40
17:50
18:00
18:10
18:20
18:30

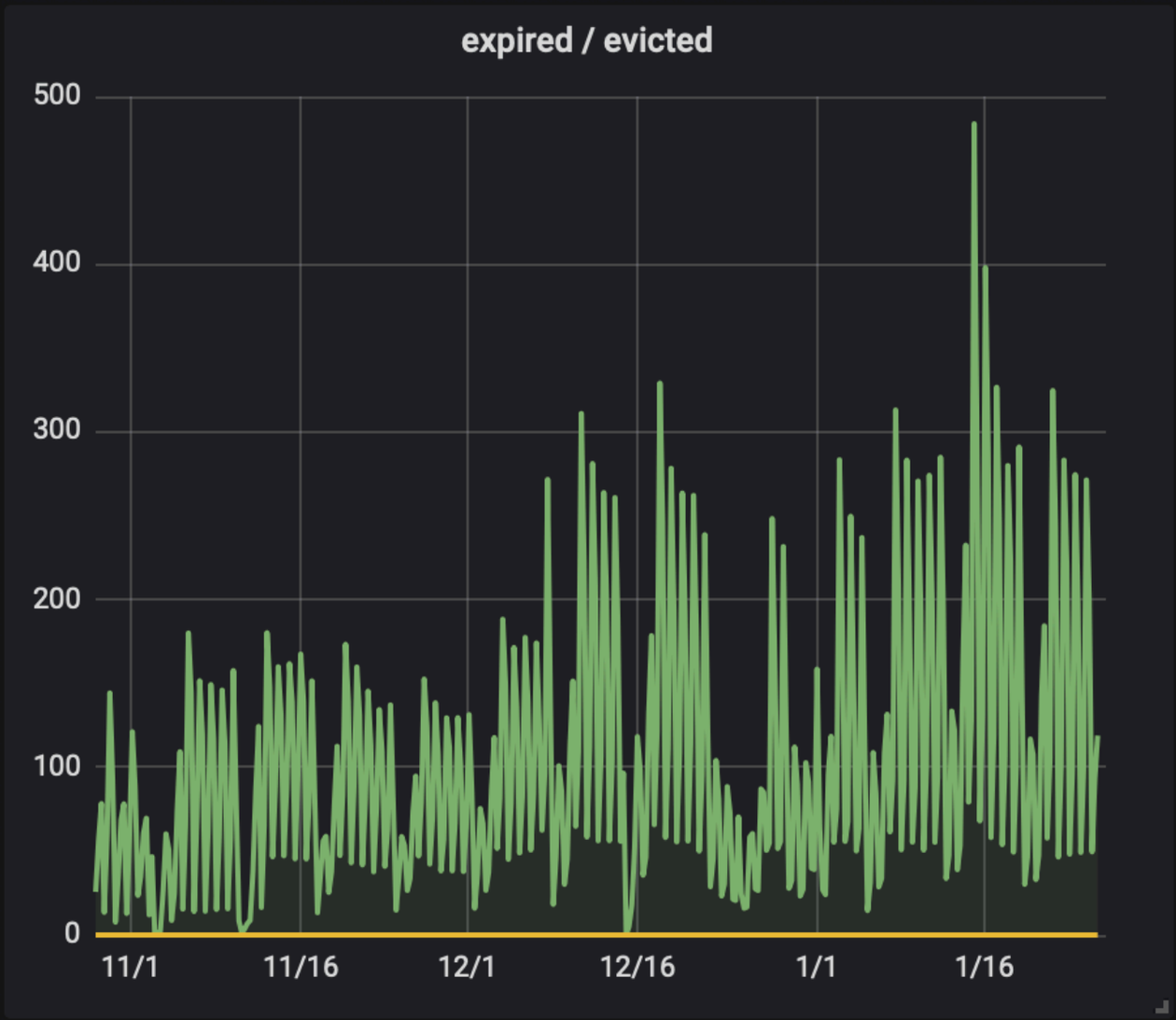

node3 disk space usage
100.0%
80.0%
60.0%
40.0%
20.0%
0%
17:40
17:50
18:00
18:10
18:20
18:30

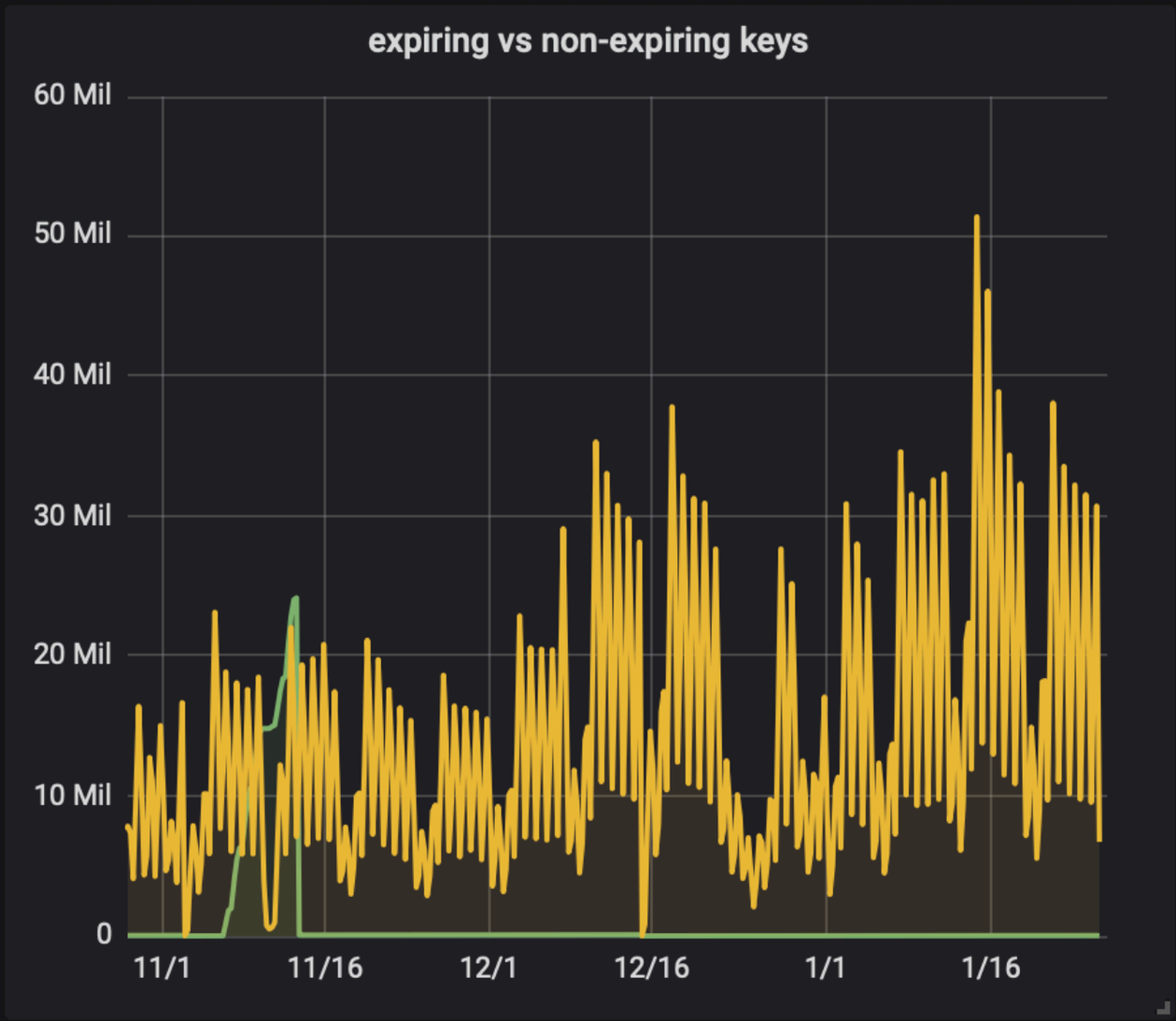
Relevant Data != Nice Graphs

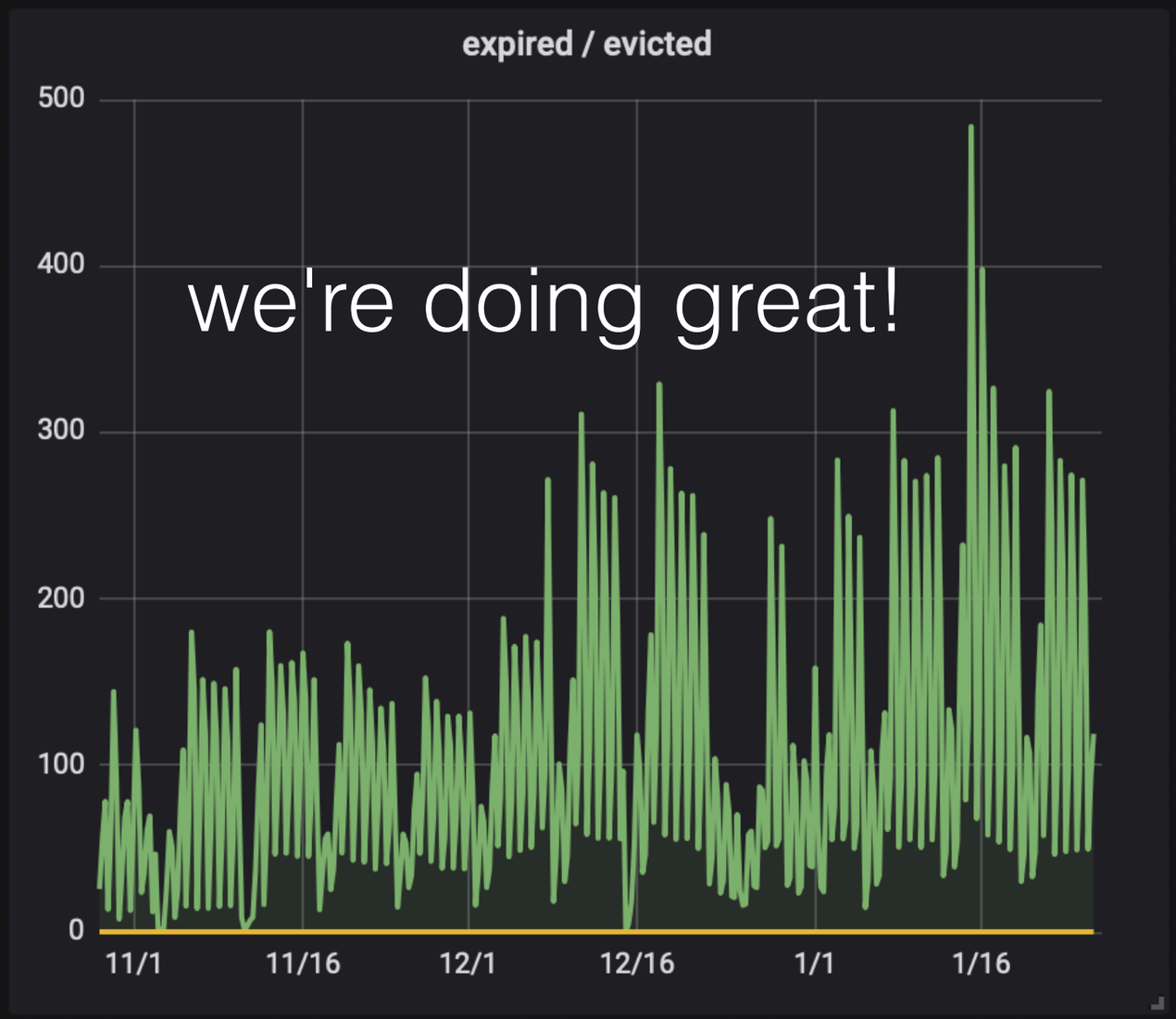

command calls / sec
35K ops
30K ops
25K ops
20K ops
15K ops
10K ops
5K ops
0 ops
12/29
12/31
1/2
1/4
1/6
1/8
1/10
1/12
1/14
1/16
1/18
1/20
1/22
1/24
1/26

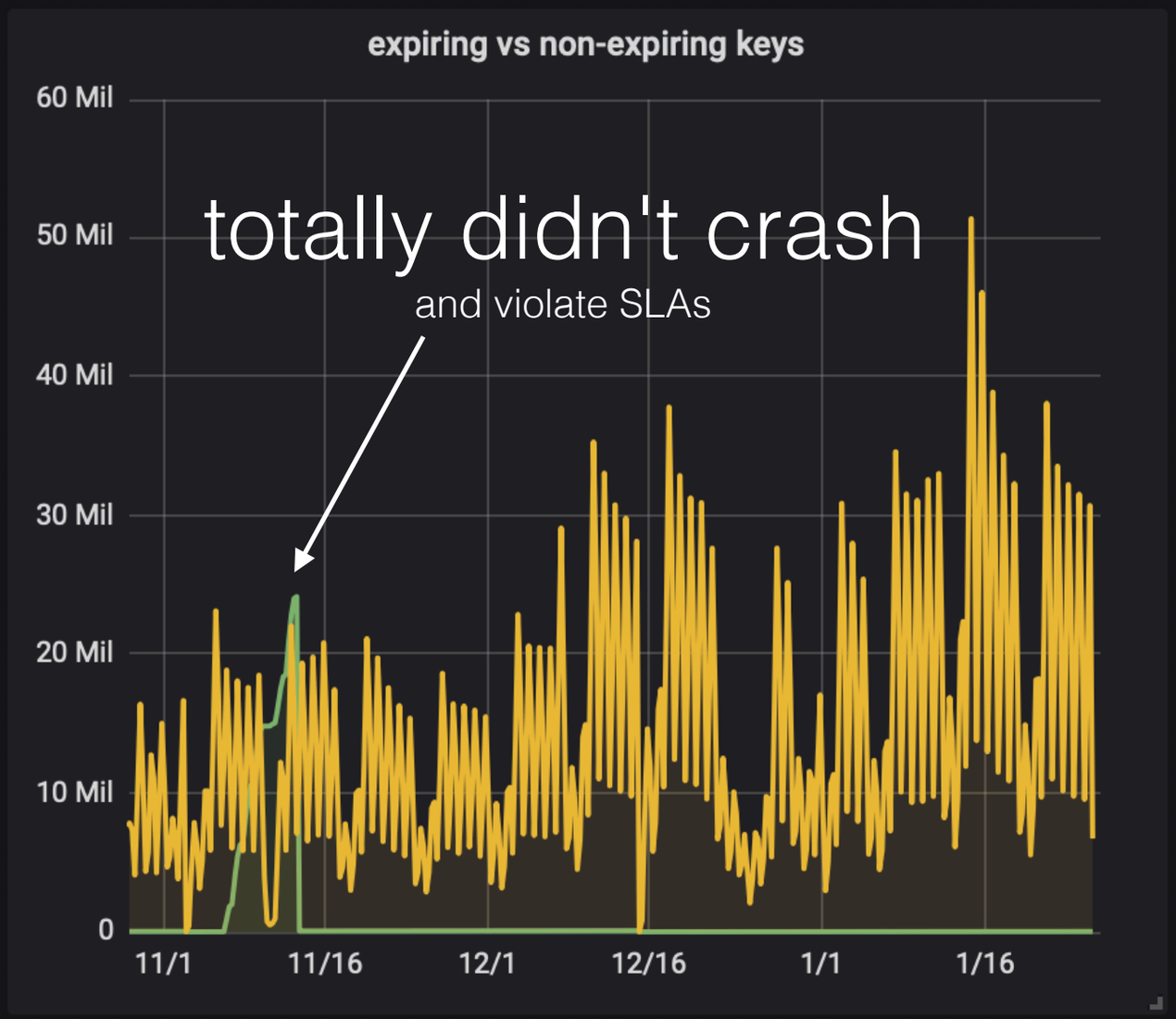

haproxy overall frontend throughput [1m]
2.4 MiB
1.9 MiB
1.4 MiB
977 KiB
488 KiB
0 B
-488 KiB
-977 KiB
-1.4 MiB
-1.9 MiB
-2.4 MiB
1/20 00:00
1/20 12:00
1/21 00:00
1/21 12:00
1/22 00:00
1/22 12:00
1/23 00:00
1/23 12:00
1/24 00:00
1/24 12:00
1/25 00:00
1/25 12:00
1/26 00:00
1/26 12:00


processing time
+Inf
10 K
500
300
200
50
12/29
12/31
01/01
01/03
01/05
01/07
01/09
01/11
01/13
01/15
01/17
01/19
01/21
01/23
01/25
0
200
400
600
800
919


expired / evicted
500
400
300
200
100
0
11/1
11/16
12/1
12/16
1/1
1/16


expiring vs non-expiring keys
60 Mil
50 Mil
40 Mil
30 Mil
20 Mil
10 Mil
0
11/1
11/16
12/1
12/16
1/1
1/16


expired / evicted
500
400
300
200
100
0
we're doing great!
11/1
11/16
12/1
12/16
1/1
1/16

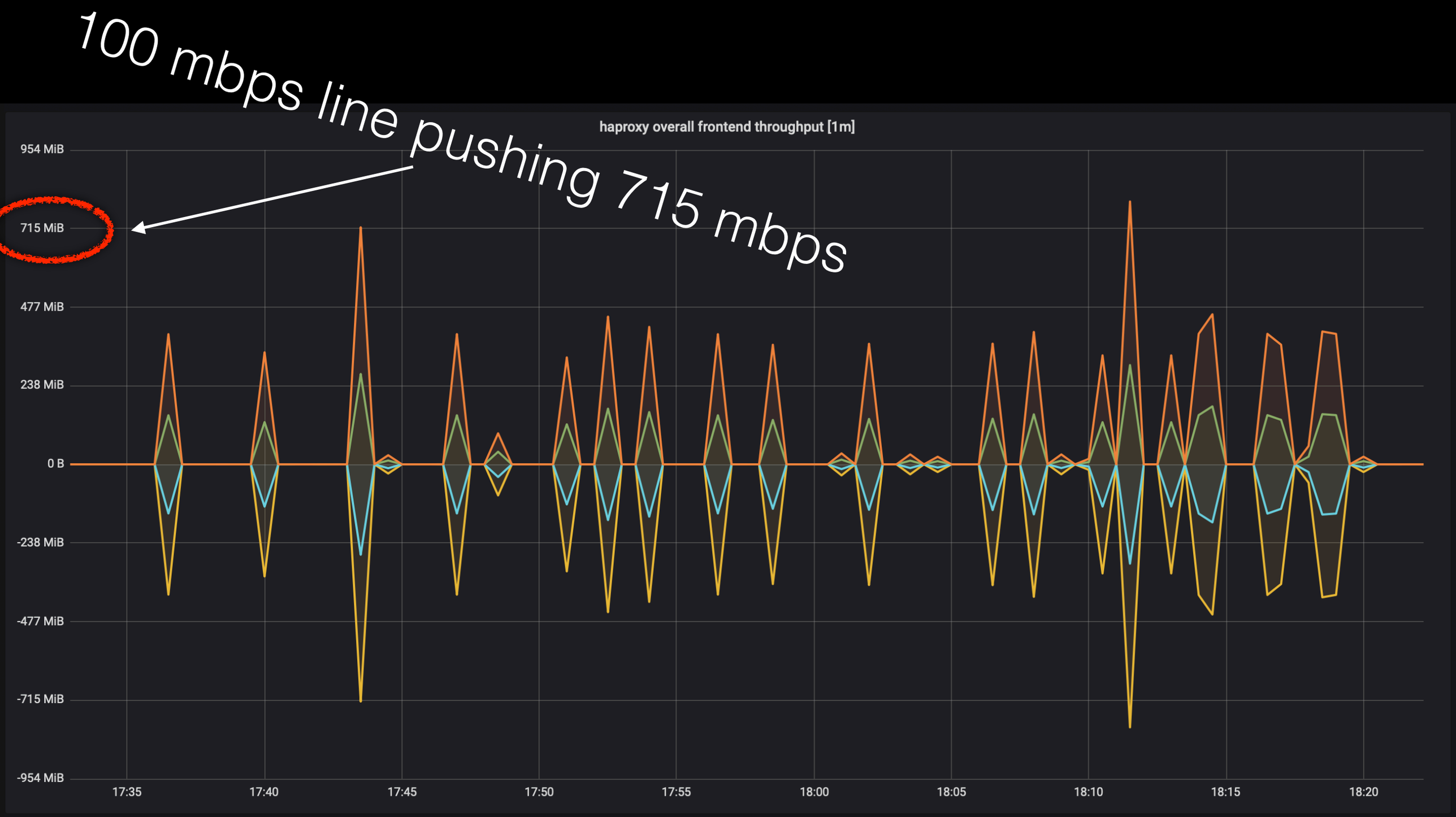

expiring vs non-expiring keys
60 Mil
50 Mil
40 Mil
30 Mil
20 Mil
10 Mil
0
totally didn't crash
and violate SLAs
11/1
11/16
12/1
12/16
1/1
1/16

Relevant Data != Nice Graphs

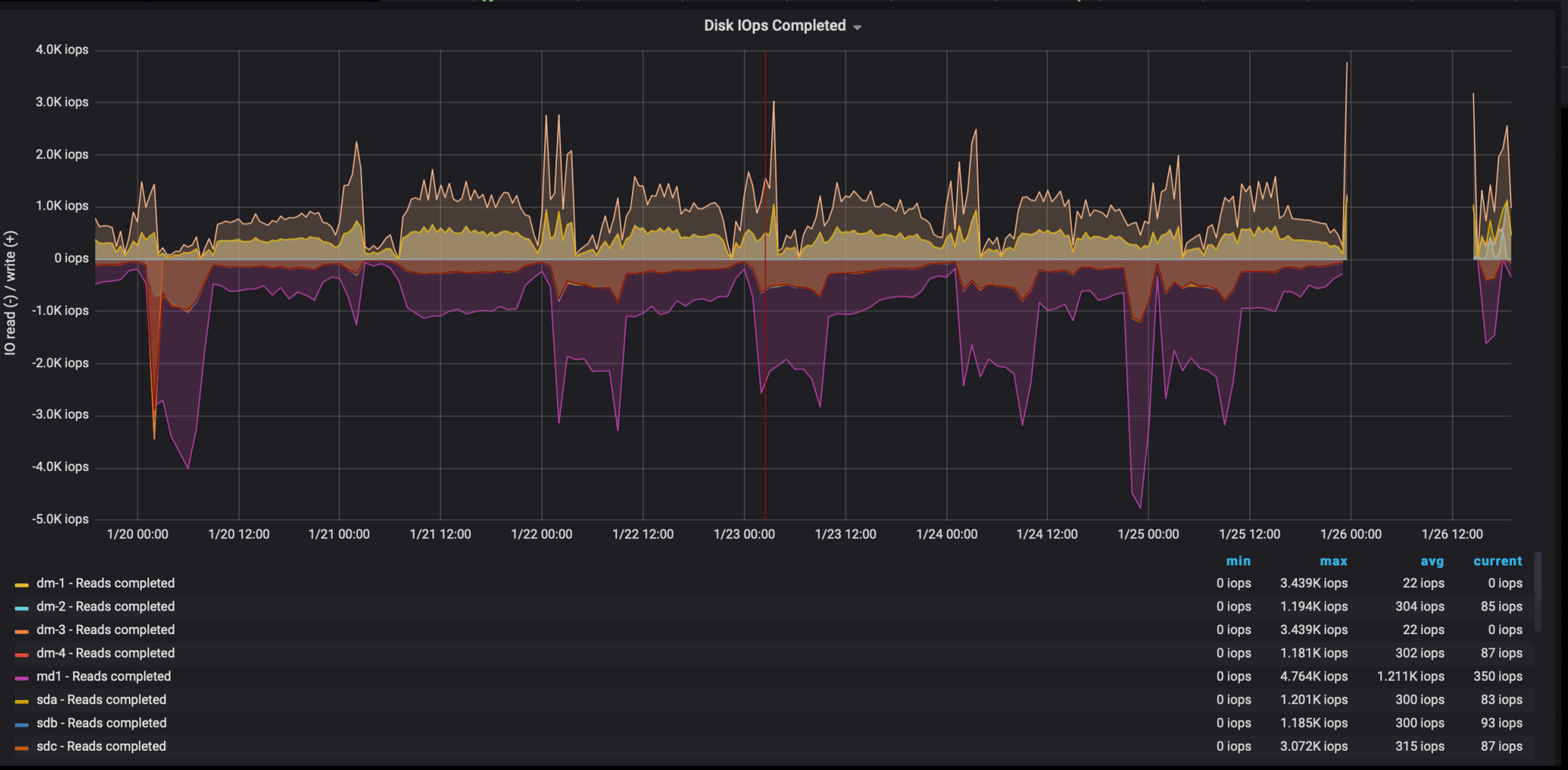
- Grafana is terrible (you might hear this few more times)
  - Because it looks beautiful
  - It makes irrelevant metrics beautiful
- Track the scary stuff

Question Your Data

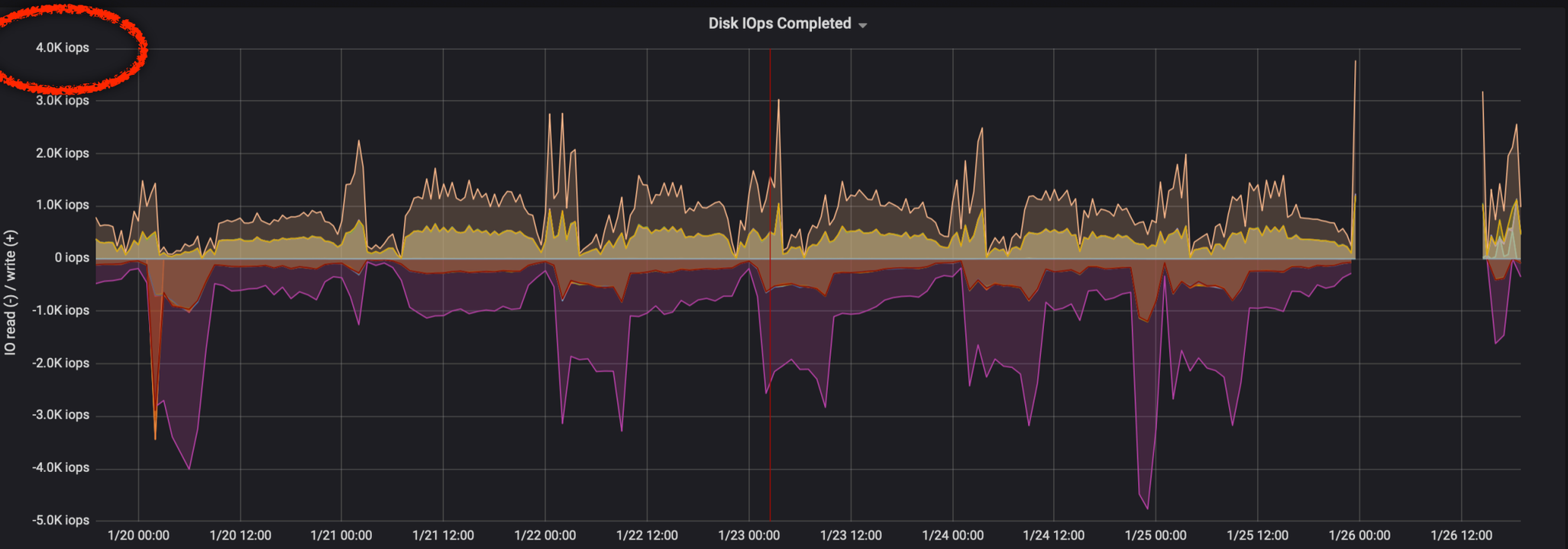
haproxy overall frontend throughput [1m]
954 MiB
715 MiB
477 MiB
238 MiB
0 B
-238 MiB
-477 MiB
-715 MiB
-954 MiB
17:35
17:40
17:45
17:50
17:55
18:00
18:05
18:10
18:15
18:20


100 mbps line pushing 715 mbps
haproxy overall frontend throughput [1m]
954 MiB
715 MiB
477 MiB
238 MiB
0 B
-238 MiB
-477 MiB
-715 MiB
-954 MiB
17:35
17:40
17:45
17:50
17:55
18:00
18:05
18:10
18:15
18:20

IOPS (- read, + write)
15 K
10 K
5 K
0
-5 K
-10 K
-15 K


Disk IOps Completed
IO read (-) / write (+)
4.0K iops
3.0K iops
2.0K iops
1.0K iops
0 iops
-1.0K iops
-2.0K iops
-3.0K iops
-4.0K iops
-5.0K iops
1/20 00:00
1/20 12:00
1/21 00:00
1/21 12:00
1/22 00:00
1/22 12:00
1/23 00:00
1/23 12:00
1/24 00:00
1/24 12:00
1/25 00:00
1/25 12:00
1/26 00:00
1/26 12:00
min max avg current
dm-1 - Reads completed 0 iops 3.439K iops 22 iops 0 iops
dm-2 - Reads completed 0 iops 1.194K iops 304 iops 85 iops
dm-3 - Reads completed 0 iops 3.439K iops 22 iops 0 iops
dm-4 - Reads completed 0 iops 1.181K iops 302 iops 87 iops
md1 - Reads completed 0 iops 4.764K iops 1.211K iops 350 iops
sda - Reads completed 0 iops 1.201K iops 300 iops 83 iops
sdb - Reads completed 0 iops 1.185K iops 300 iops 93 iops
sdc - Reads completed 0 iops 3.072K iops 315 iops 87 iops

15k vs 4k IOPS

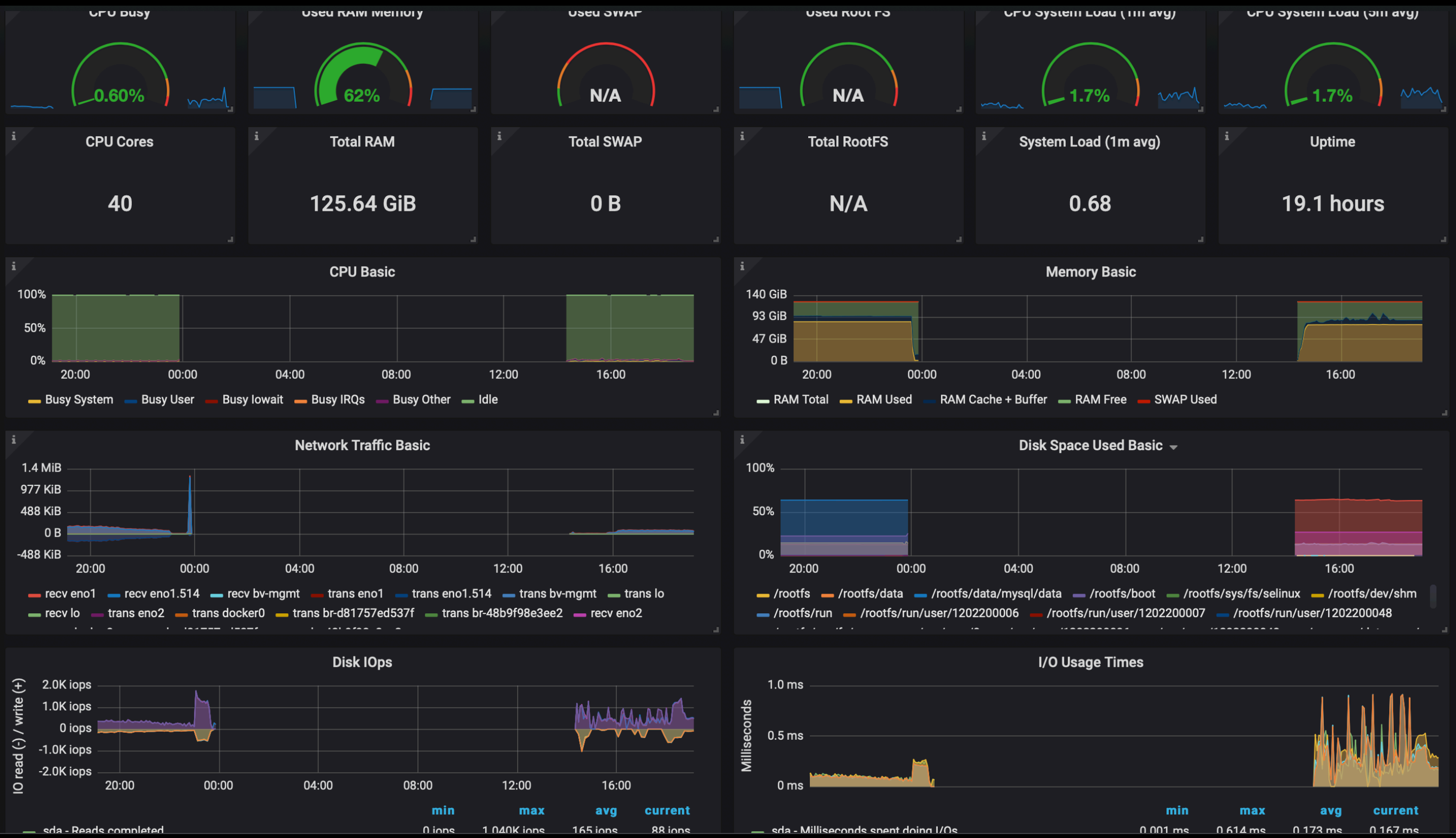


	min	max	avg	current
dm-1 - Reads completed	0 iops	3.439K iops	22 iops	0 iops
dm-2 - Reads completed	0 iops	1.194K iops	304 iops	85 iops
dm-3 - Reads completed	0 iops	3.439K iops	22 iops	0 iops
dm-4 - Reads completed	0 iops	1.181K iops	302 iops	87 iops
md1 - Reads completed	0 iops	4.764K iops	1.211K iops	350 iops
sda - Reads completed	0 iops	1.201K iops	300 iops	83 iops
sdb - Reads completed	0 iops	1.185K iops	300 iops	93 iops
sdc - Reads completed	0 iops	3.072K iops	315 iops	87 iops

Question Your Data

- When monitoring a metric, question the correctness
- Have someone review the data
- Care for double (or higher multiple) counting
- Physics can be handy!

Clutter

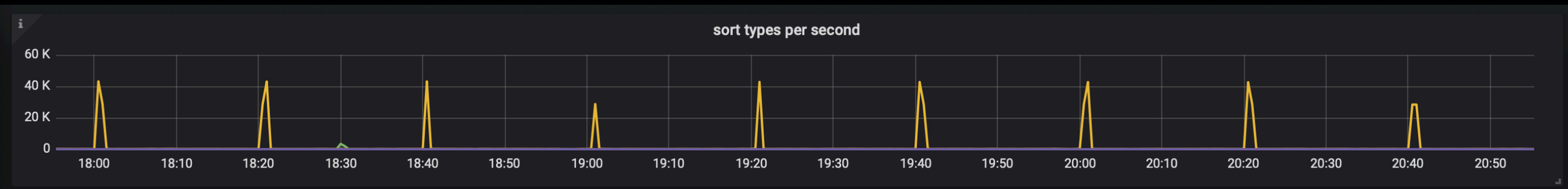


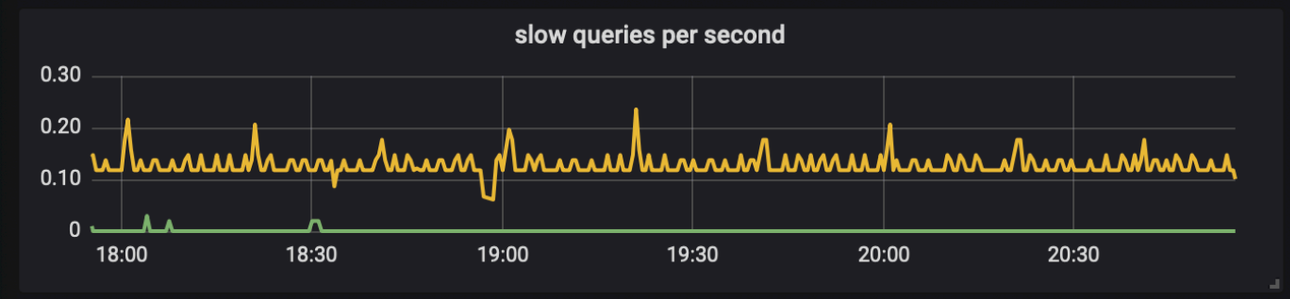
* this screenshot intentionally looks like a mess

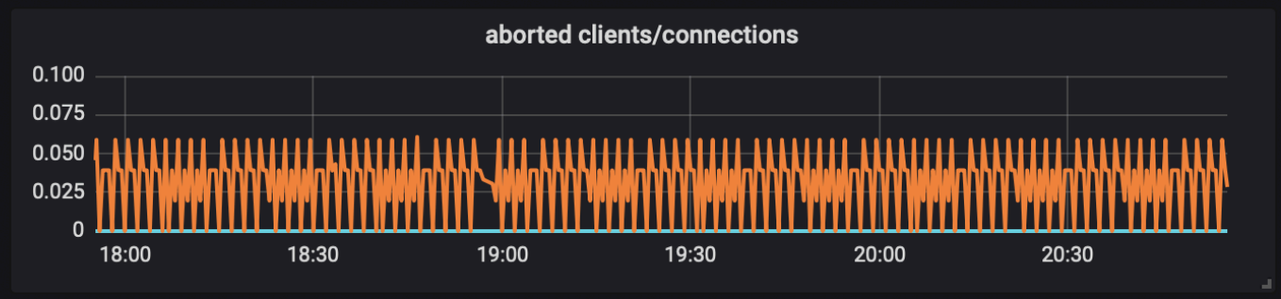
* this screenshot intentionally looks like a mess

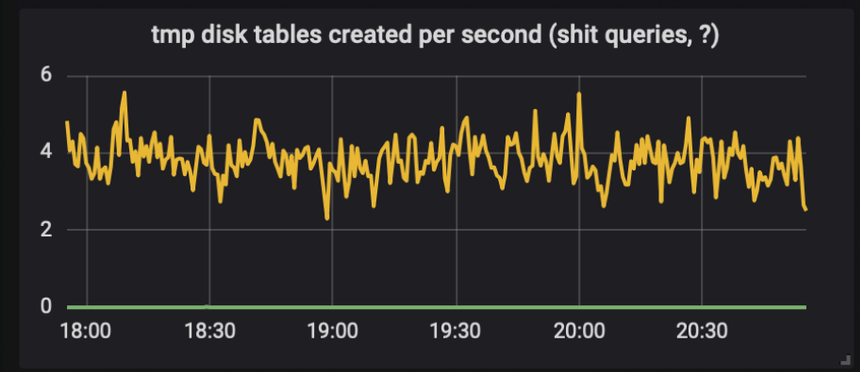
Clutter

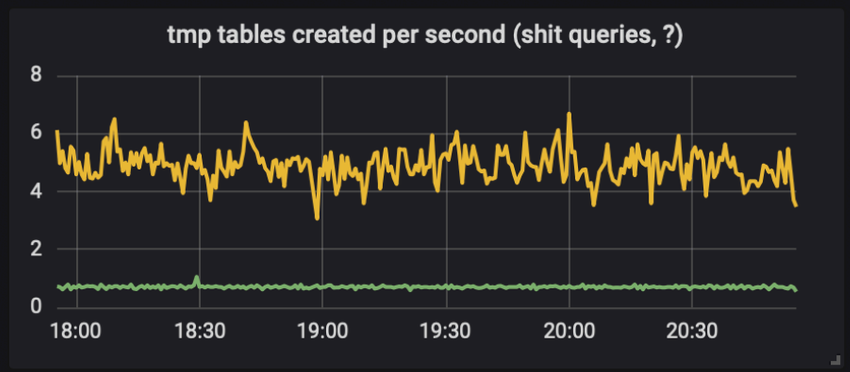
- Grafana is terrible (told you)
  - It makes adding graphs and dashboards too easy!
- The struggle...
  - Seeing absolutely ridiculous amount of data
  - Not being able to analyze the system
- Can we fix that?

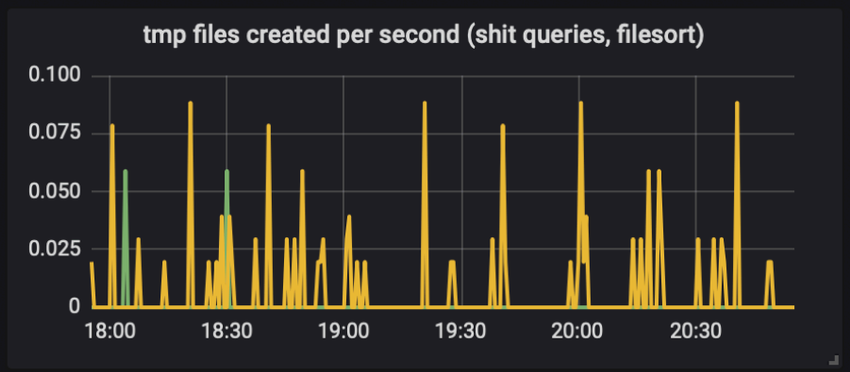

sort types per second
60 K
40 K
20 K
0
18:00
18:10
18:20
18:30
18:40
18:50
19:00
19:10
19:20
19:30
19:40
19:50
20:00
20:10
20:20
20:30
20:40
20:50


slow queries per second
0.30
0.20
0.10
0
18:00
18:30
19:00
19:30
20:00
20:30

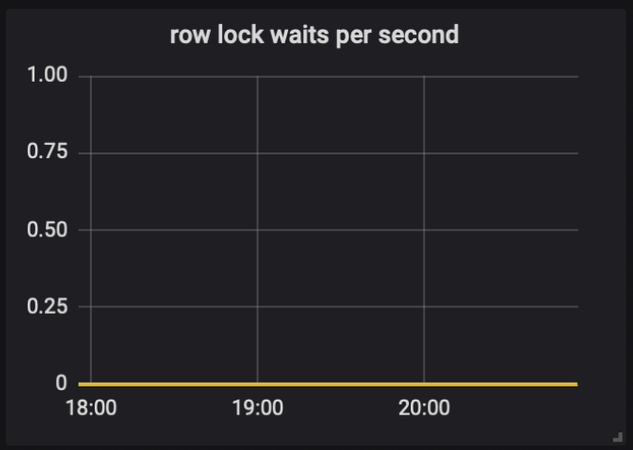

aborted clients/connections
0.100
0.075
0.050
0.025
0
18:00
18:30
19:00
19:30
20:00
20:30

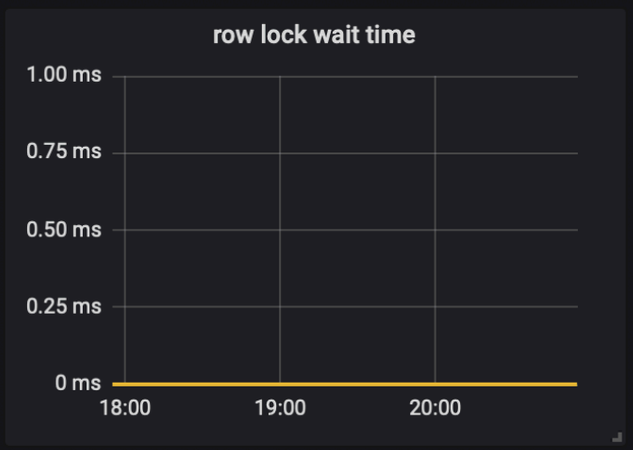

tmp disk tables created per second (shit queries, ?)
6
4
2
0
18:00
18:30
19:00
19:30
20:00
20:30

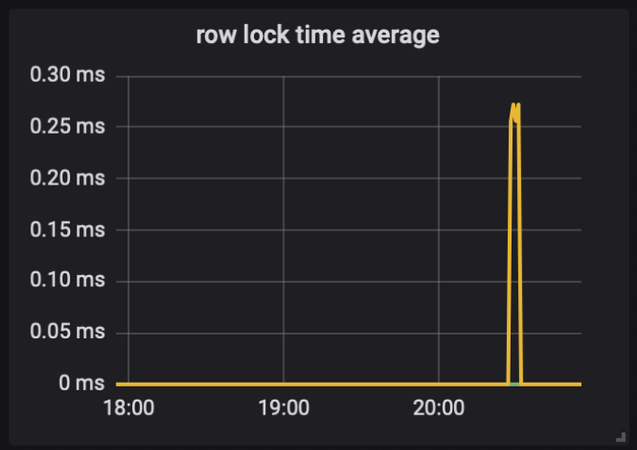

tmp tables created per second (shit queries, ?)
8
6
4
2
0
18:00
18:30
19:00
19:30
20:00
20:30


tmp files created per second (shit queries, filesort)
0.100
0.075
0.050
0.025
0
18:00
18:30
19:00
19:30
20:00
20:30

locking per second
1.00
0.75
0.50
0.25
0
18:00
19:00
20:00


row lock waits per second
1.00
0.75
0.50
0.25
0
18:00
19:00
20:00


row lock wait time
1.00 ms
0.75 ms
0.50 ms
0.25 ms
0 ms
18:00
19:00
20:00


row lock time average
0.30 ms
0.25 ms
0.20 ms
0.15 ms
0.10 ms
0.05 ms
0 ms
18:00
19:00
20:00

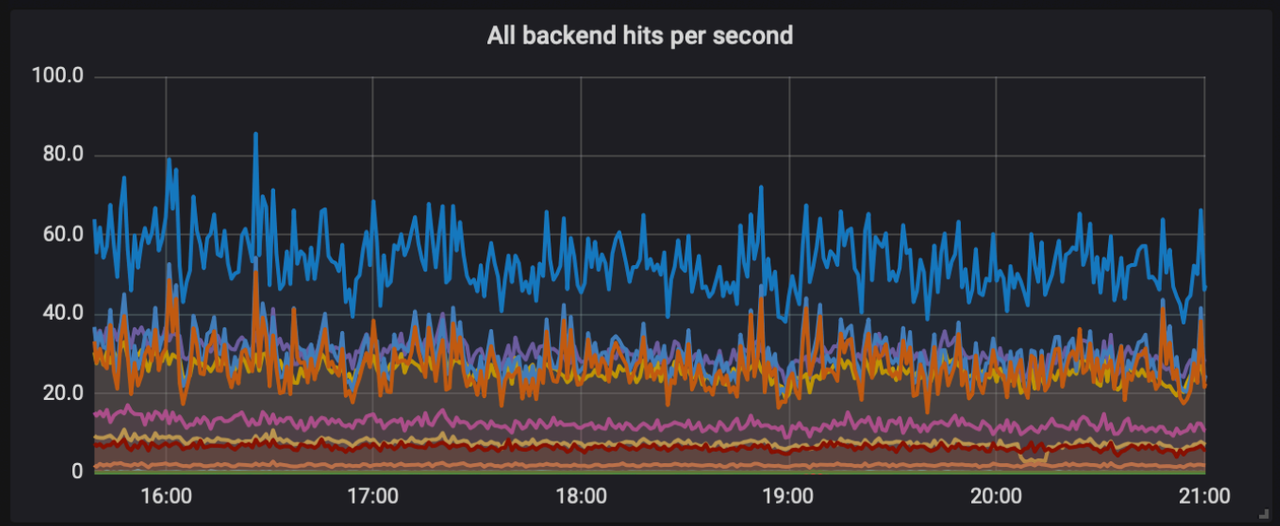
locking per second
1.00
0.75
0.50
0.25
0
19:40
19:45
19:50
19:55
20:00
20:05
20:10
20:15
20:20
20:25
20:30
20:35
row lock waits per second
0.025
0.020
0.015
0.010
0.005
0
19:40
19:50
20:00
20:10
20:20
20:30
row lock wait time
4.0 ms
3.0 ms
2.0 ms
1.0 ms
0 ms
19:40
19:50
20:00
20:10
20:20
20:30
row lock time average
0.30 ms
0.25 ms
0.20 ms
0.15 ms
0.10 ms
0.05 ms
0 ms
19:40
19:50
20:00
20:10
20:20
20:30
queries per second
800
600
400
200
0
fsync and writes per second
800
600
400
200
0

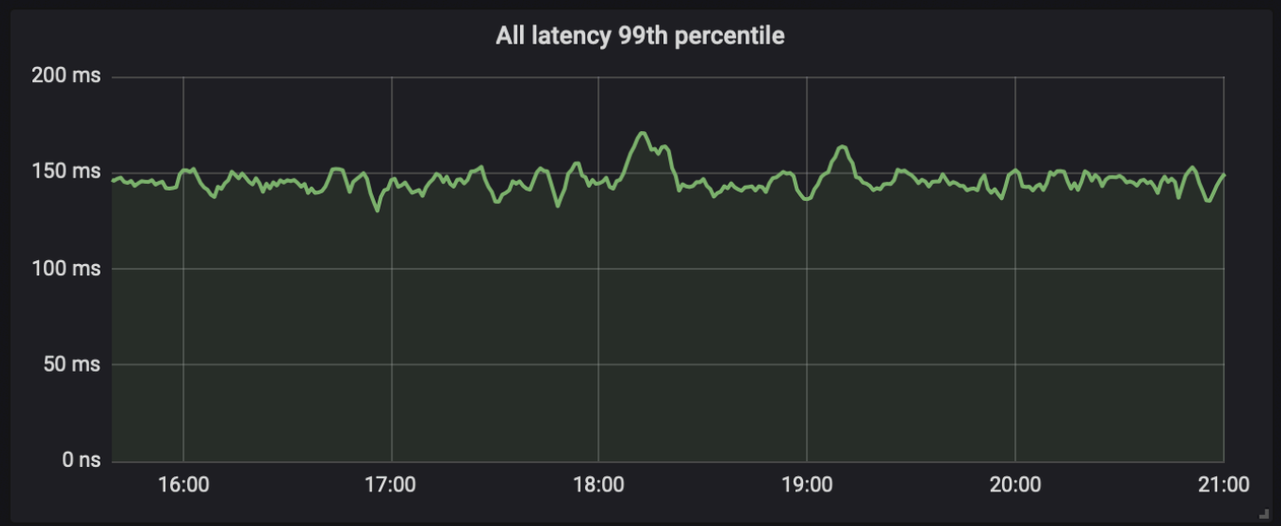
Clutter

- Think about dashboard space as limited resource
- Group correlated graphs
- Spacing
- Positioning
- Many graphs = slow dashboard (JS issue)
  - Pronounced with shared timeline selector

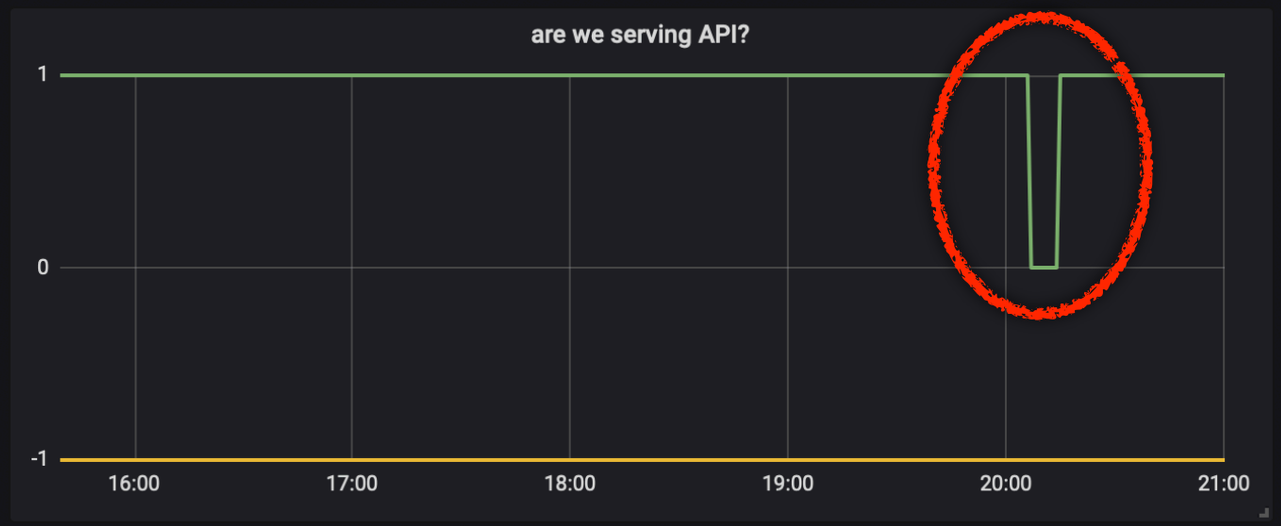
Overview at a Glance

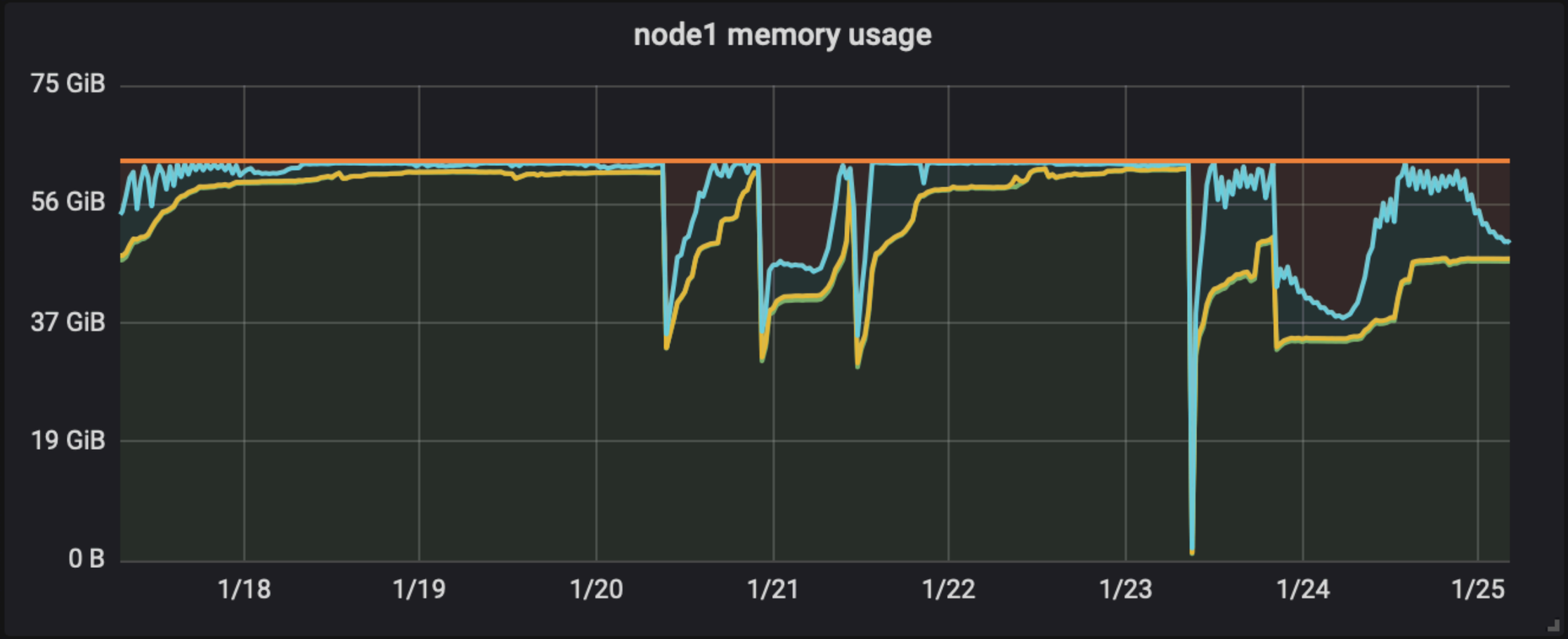
- "Think about dashboard space as limited resource"
- We need a way to
  - Quickly determine the state of a system
  - Dig deeper / root cause analysis of an issue

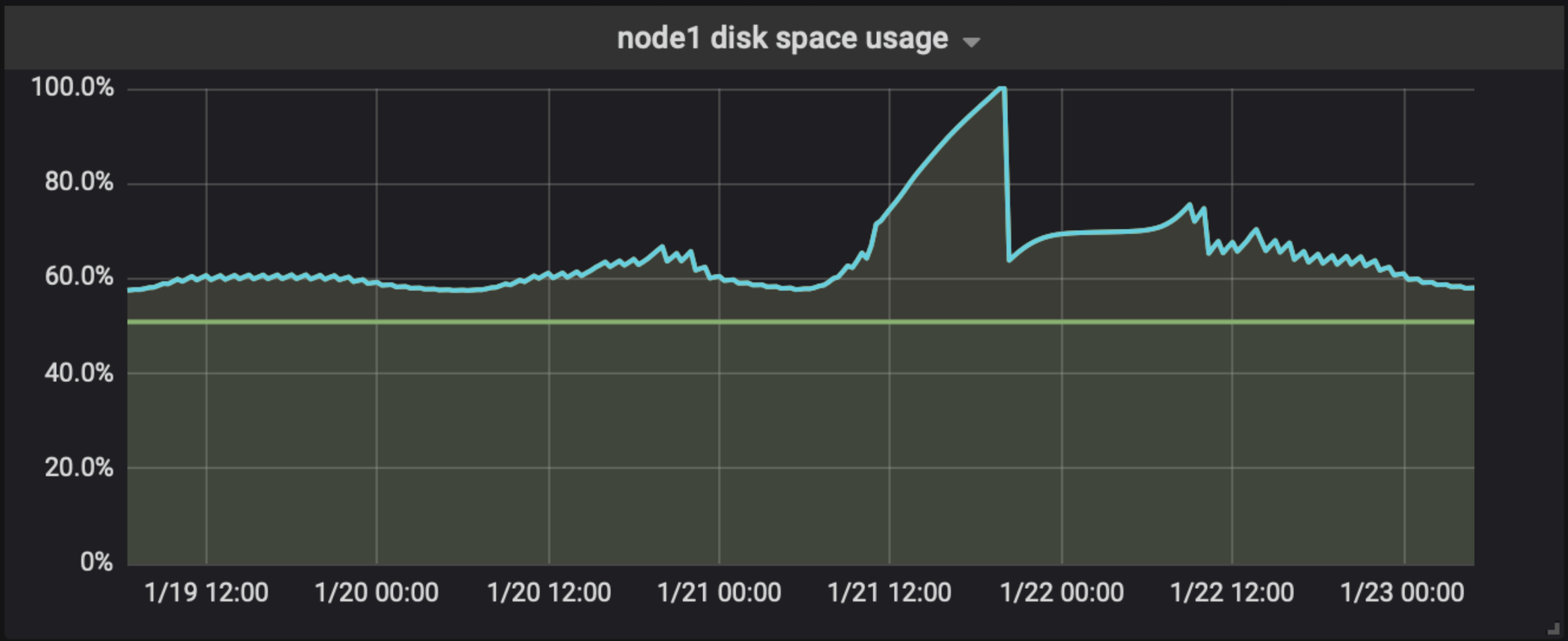

All backend hits per second
100.0
80.0
60.0
40.0
20.0
0
16:00
17:00
18:00
19:00
20:00
21:00


All latency 99th percentile
200 ms
150 ms
100 ms
50 ms
0 ns
16:00
17:00
18:00
19:00
20:00
21:00

are we serving JS?
1
0
-1
16:00
17:00
18:00
19:00
20:00
21:00


are we serving API?
1
0
-1
16:00
17:00
18:00
19:00
20:00
21:00


node1 memory usage
75 GiB
56 GiB
37 GiB
19 GiB
0 B
1/18
1/19
1/20
1/21
1/22
1/23
1/24
1/25

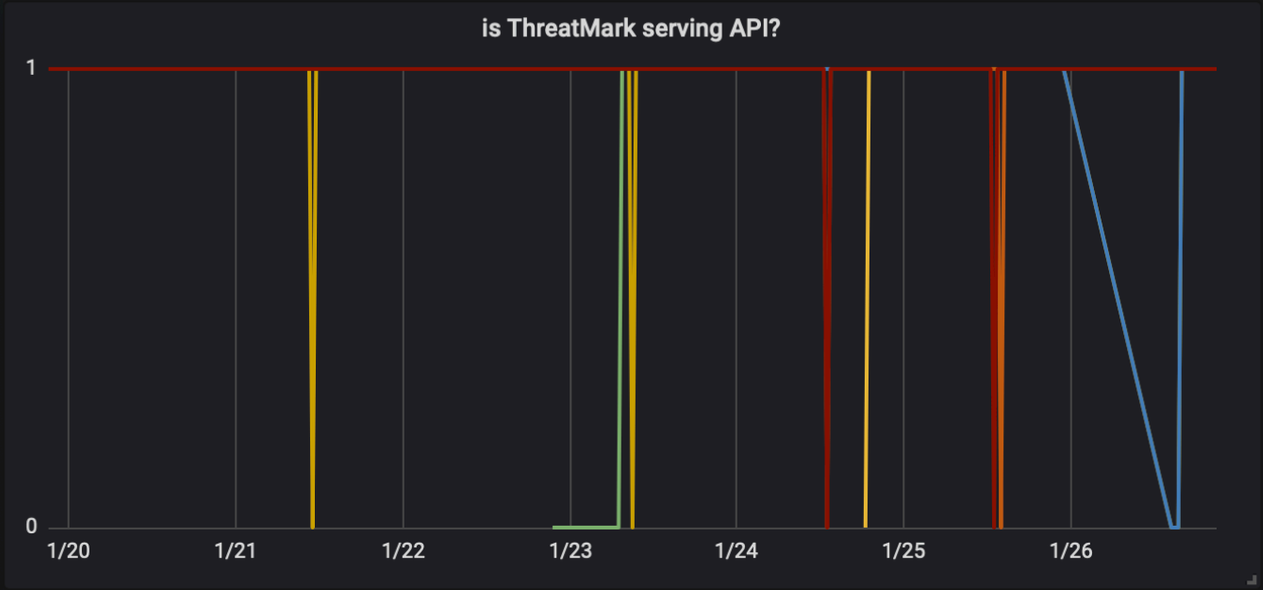

node1 disk space usage
100.0%
80.0%
60.0%
40.0%
20.0%
0%
1/19 12:00
1/20 00:00
1/20 12:00
1/21 00:00
1/21 12:00
1/22 00:00
1/22 12:00
1/23 00:00

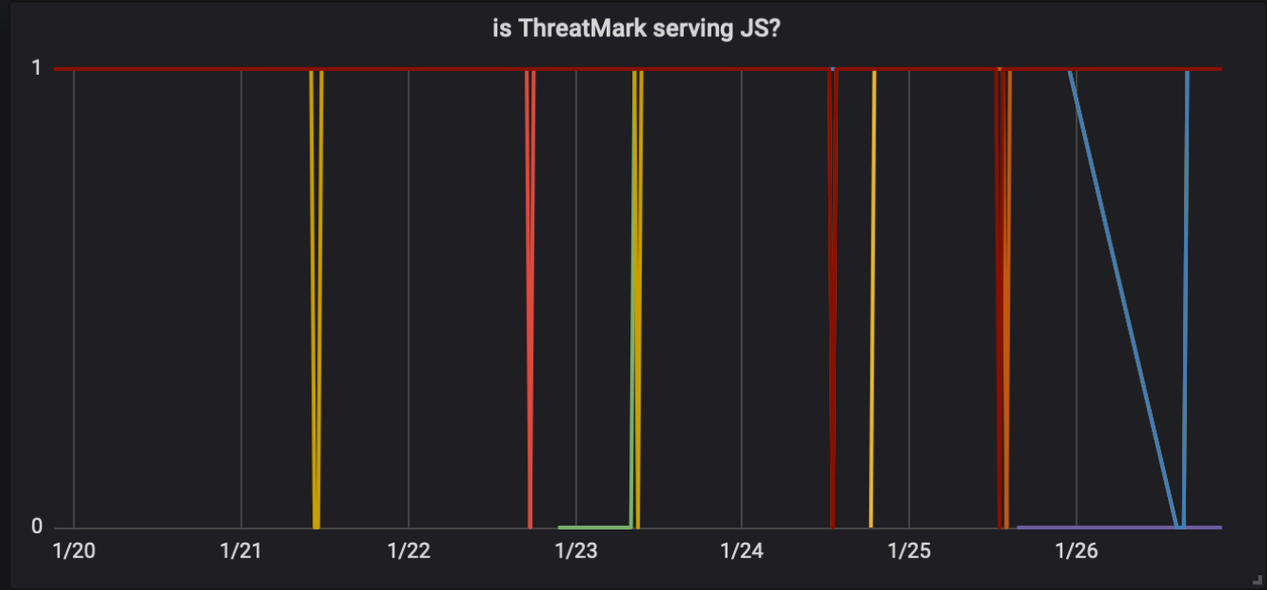
Overview at a Glance

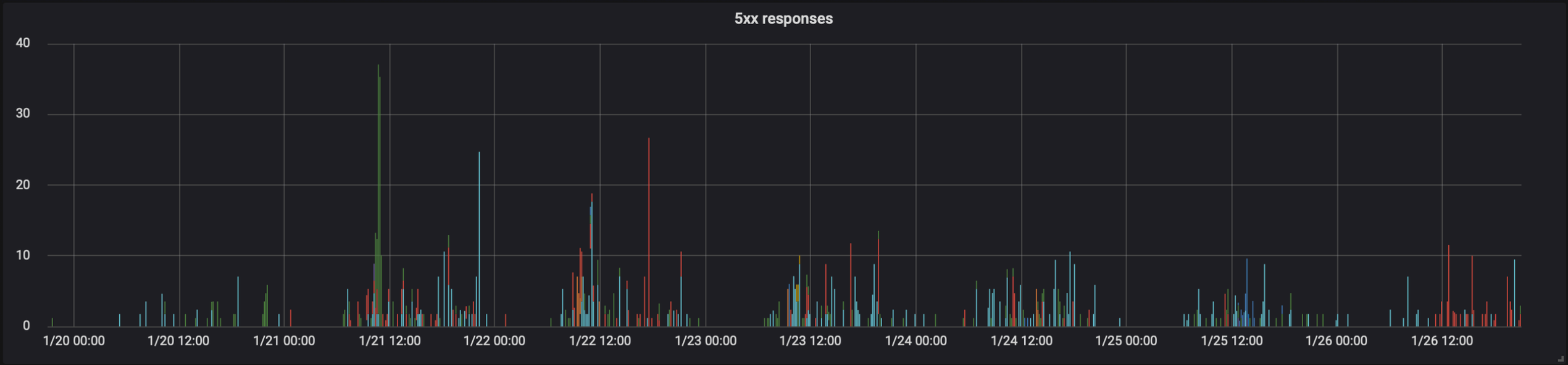
- Further analysis can be done via traces and logs
- What about multiple customers/deployments?

Is ThreatMark Running?

Last 7 days Refresh every 10s

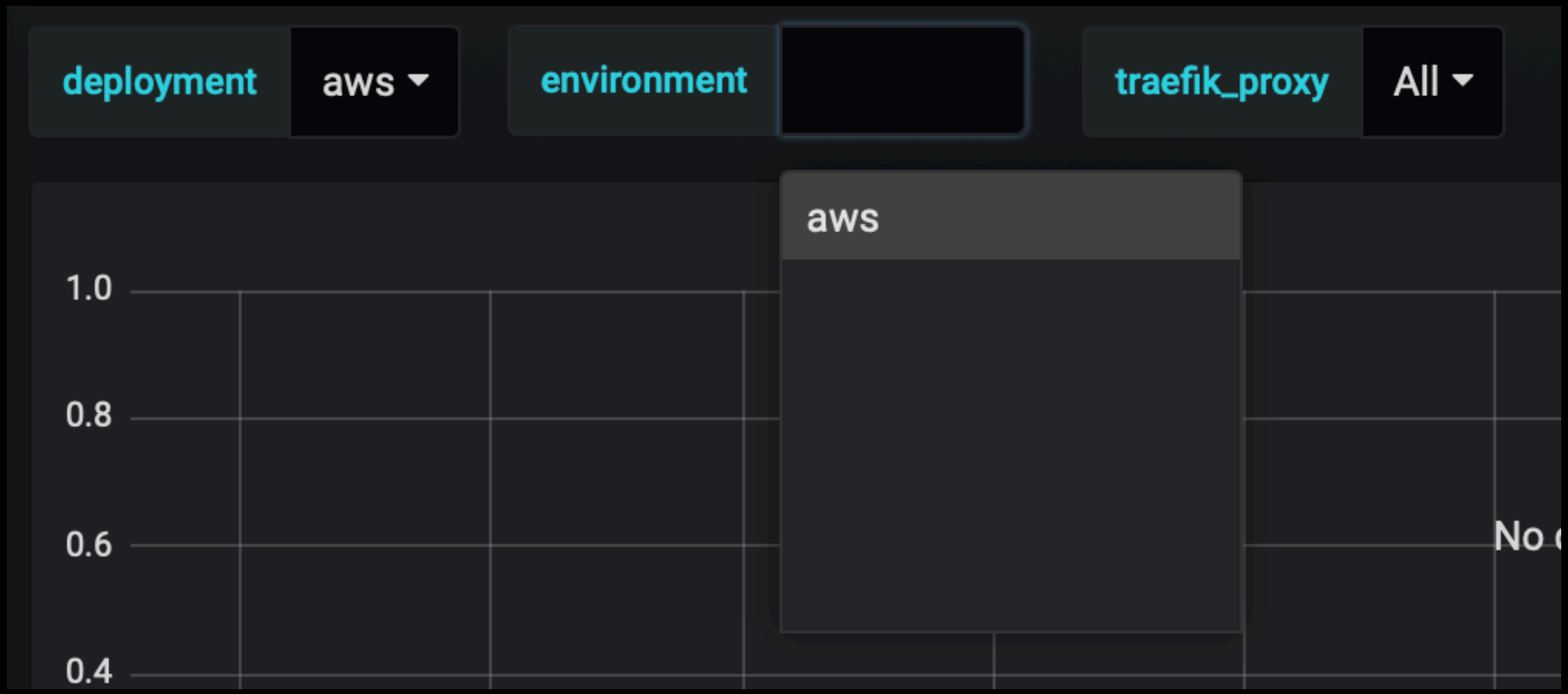






Overview at a Glance

- What about multiple customers/deployments?
  - Grafana templating to the rescue!
- Maybe even repeating graphs?



* production dashboard, showing only non-confidential stuff :)

node5 commands executed / sec
23:10
23:20
23:30
23:40
23:50

node6 commands executed / sec
800
600
400
200
0
23:00
23:10
23:20
23:30
23:40
23:50

node7 commands executed / sec
800
600
400
200
0
23:00
23:10
23:20
23:30
23:40

Don't Leave a Mess

- Experiment with your dashboards!
- Try visualizing data in a new, better way
- Do crazy things
  - ... but clean up your Grafana :(

General

- 1first
- AFS Dashboard — database, logs, redis, sanity, traefik
- Disks are on Fire!
- Is ThreatMark Running? — sanity
- MySQL — database
- Node Exporter Full
- Nodes — cadvisor, node
- Perf — logs, sanity, traefik
- Proxies — haproxy, traefik
- Redis — node, redis
- TEST - Debug dashboard
- TEST - Prometheus Redisaaaaaaaaaa — prometheus, redis
- UNUSED

Don't Leave a Mess

- See the tags?
- Organize the dashboards, namespace them
- Have a naming + tagging policies in place
- Provision mission critical dashboards

Version Control

- Dashboard is yaml file, easy to store in git
- But reviewing changes hurts
  - Try moving one graph to different position...

Open **mpolednik** wants to merge 1 commit into `master` from `test`

Conversation 0 | Commits 1 | Checks 0 | Files changed 1

Changes from **all commits** | **File filter...** | **Jump to...** | +106 −106

```
@@ -361,10 +460,10 @@
361  460          "gridPos": {
362  461            "h": 5,
363  462            "w": 12,
364       -        "x": 0,
     463  +        "x": 4,
365  464            "y": 21
366  465          },
367       -        "id": 4,
     466  +        "id": 8,
368  467          "isNew": true,
369  468          "legend": {
370  469            "alignAsTable": false,
@@ -399,7 +498,7 @@
```

Version Control

- Keep your dashboards in Python!
  - Work around Grafana limitations
  - Auto-generate graph IDs
- Code reviews are breeze (kinda)
- https://github.com/weaveworks/grafanalib

```
176     Graph(
1           title="aborted clients/connections",
2           dataSource='Prometheus',
3           targets=[
4               Target(
5                   expr="rate(mysql_global_status_aborted_connects{{{filters},instance=~'$dbnode'}}[1m]
6                   legendFormat="{{ instance }} aborted connections",
7                   refId="A",
8               ),
9               Target(
10                  expr="rate(mysql_global_status_aborted_clients{{{filters},instance=~'$dbnode'}}[1m])
11                  legendFormat="{{ instance }} aborted connections",
12                  refId="B",
13              ),
14          ],
15          fill=0,
16          legend=Legend(show=False),
17          yAxes=[
18              YAxis(format=SHORT_FORMAT),
19              YAxis(format=SHORT_FORMAT),
20          ],
21          xAxis=XAxis(mode="time")
22      ),
23  ]),
```

Version Control

- Doesn't fit nicely with pre-made dashboards
- Solution?
  - Re-implement them in Python (eww)
  - Consider them vendored (store as yaml)
  - Other ideas?

Conversation 0 | Commits 1 | Checks 0 | Files changed 1

Changes from **all commits** | **File filter...** | **Jump to...** | +11 −11

Application Monitoring

Application Monitoring

- Gather stunning amount of data about app by observing infrastructure serving it
- Grafana also supports SQL data sources

Data Sources / afs-core-db

Type: MySQL

Settings

Name	name	Default	
Type	MySQL		

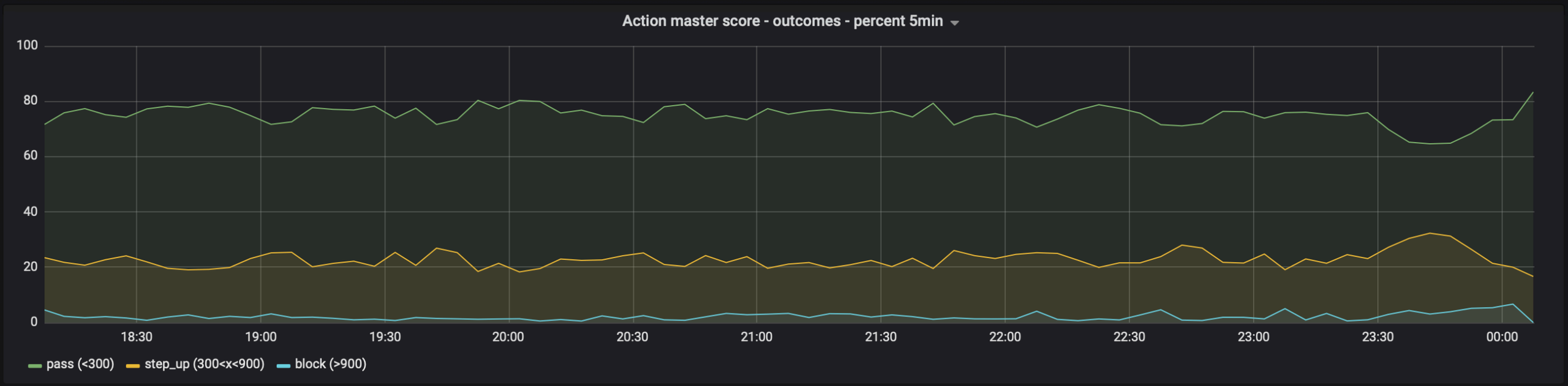
MySQL Connection

Host	localhost:3306		
Database	database name		
User	user	Password	password

User Permission

The database user should only be granted SELECT permissions on the specified database & tables you want to query. Grafana does not validate that queries are safe so queries can contain any SQL statement. For example, statements like `USE otherdb;` and `DROP TABLE user;` would be executed. To protect against this we **Highly** recommmend you create a specific MySQL user with restricted permissions. Checkout the MySQL Data Source Docs for more information.

Save & Test | Delete | Back



Graph | General | Metrics | Axes | Legend | Display | Alert | Time range

Data Source: afs-core-db

Query Inspector

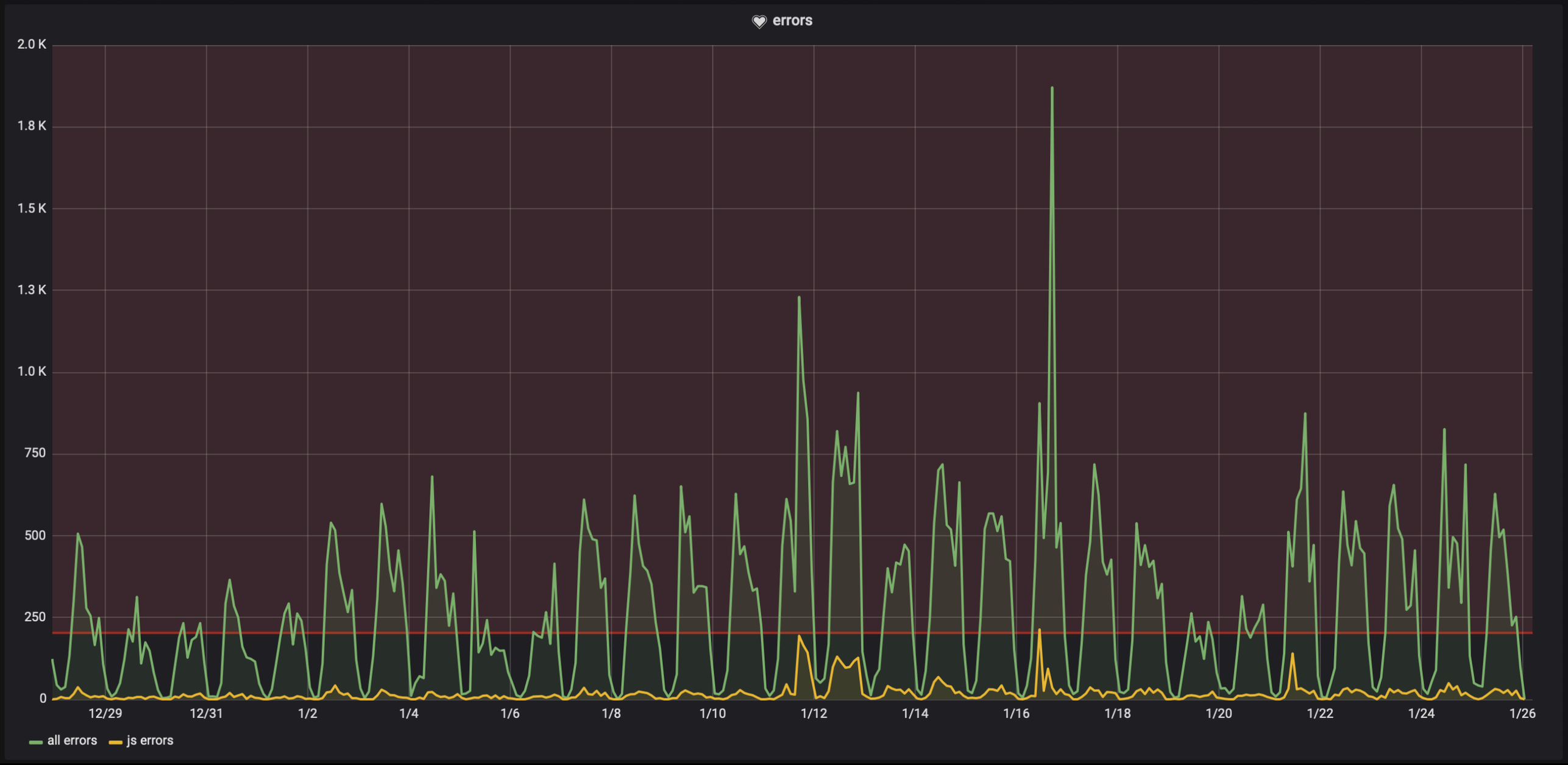
A

```
SELECT
  MIN(UNIX_TIMESTAMP(creation_time)) as time_sec,
  count(case when master_score < 300 then 1 else null end)/count(*) * 100 as "pass (<300)",
  count(case when master_score > 300 and master_score < 900  then 1 else null end)/count(*) * 100 as "step_up (300<x<900)",
    table                                                  keyword ) * 100 as "block (>900)"
FROM TABLE
WHERE $__timeFilter(creation_time)
GROUP BY $__timeGroup(creation_time,'5m')
ORDER BY creation_time ASC
```

Format as: Time series | Show Help | Generated SQL

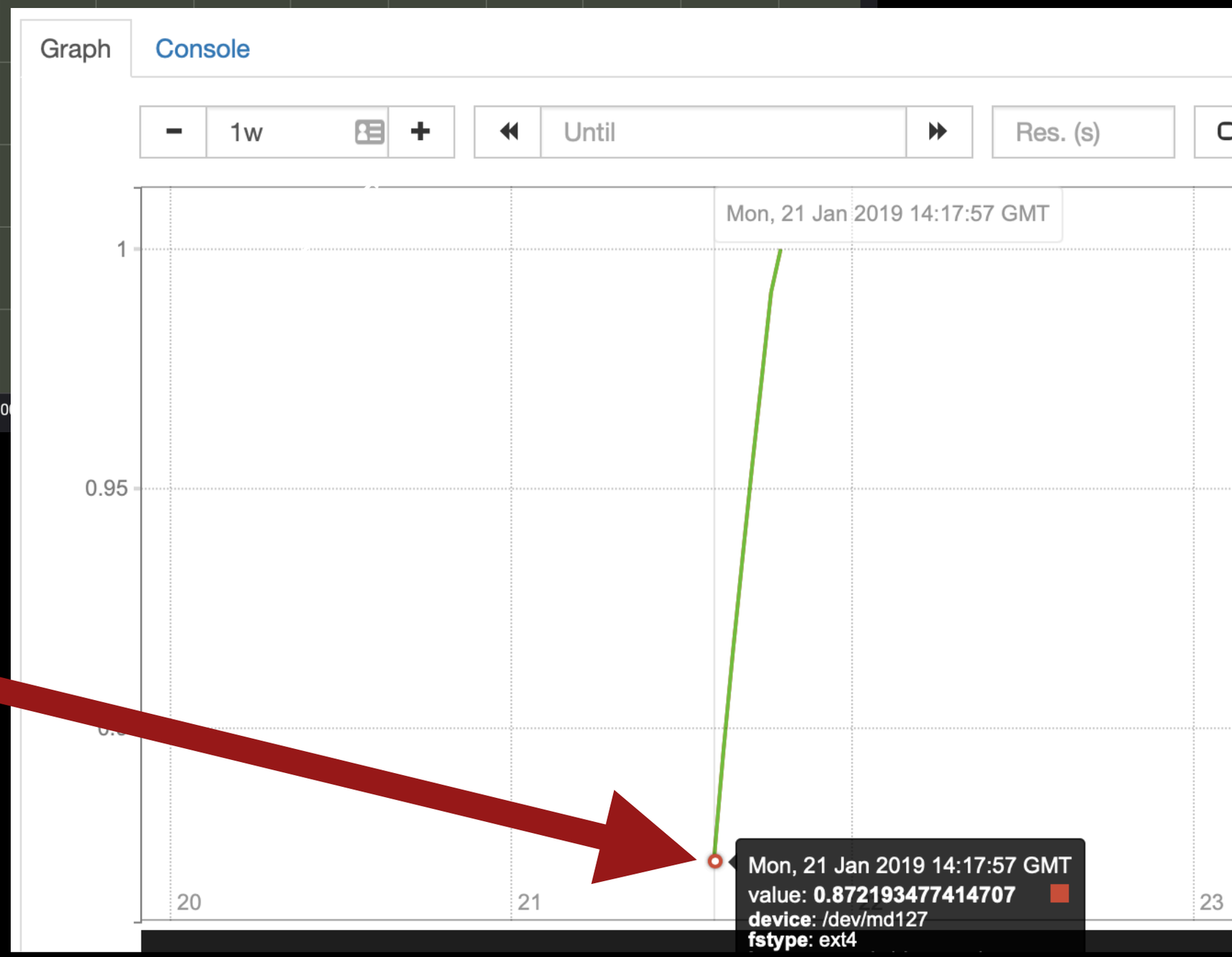
Alerting

- Grafana is terrible (...)
  - Alerting is extremely nice visually
- Not sophisticated enough to properly alert


errors
2.0 K
1.8 K
1.5 K
1.3 K
1.0 K
750
500
250
0
12/29
12/31
1/2
1/4
1/6
1/8
1/10
1/12
1/14
1/16
1/18
1/20
1/22
1/24
1/26
all errors
js errors

Alerting

- Better handled by data source, e.g. Prometheus + Alertmanager
- Alert on symptoms
  - Anomaly detection works well
    - avg + 2 * stddev
  - predict_linear (80% disk capacity vs disk capacity fill up rate)
  - Recording rules



predict_linear
+ constant

Summary

- Grafana is great
  - Maybe even good
- Don't overwhelm yourself with metrics just because it's easy
- Have fewer dashboards showing **relevant** metrics
- Keep overview and detail dashboards separated

Summary

- Focus on customer and business needs
- Monitoring new metric may be cheap
  - But further implications could be expensive

Thank you!

Q/A