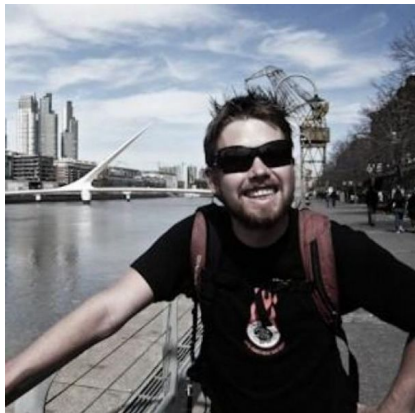




GitLab

Resource Saturation Monitoring and Capacity Planning on GitLab.com

Andrew Newdigate, GitLab



Andrew Newdigate

Scalability Team, Infrastructure Group, GitLab



@suprememoocow



gitlab.com/andrewn



Resource Saturation Incident RCA: GitLab.com Redis CPU Saturation

<https://gitlab.com/gitlab-com/gl-infra/production/issues/928>

<https://gitlab.com/gitlab-com/gl-infra/infrastructure/issues/7157>

GitLab.com Web Performance (Apdex Score)



Percentage requests completed within threshold. Higher is better





Redis Cache CPU Saturation

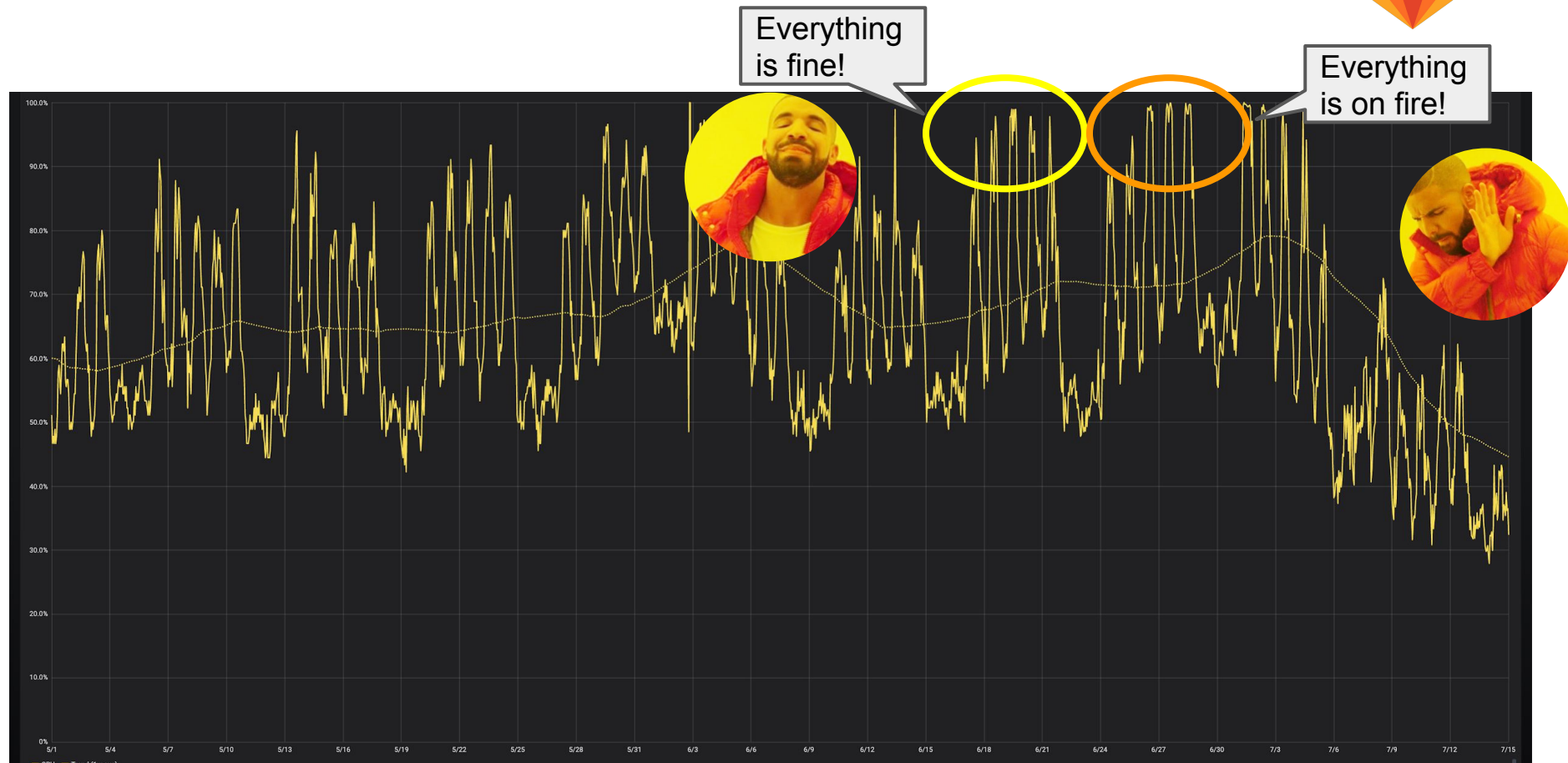
- Redis server is **single-threaded**
- Redis running on 4 core servers, 3 of the cores ~idle at any time
- Redis cache operations queuing, leading to slow down across multiple systems that relied on the cache

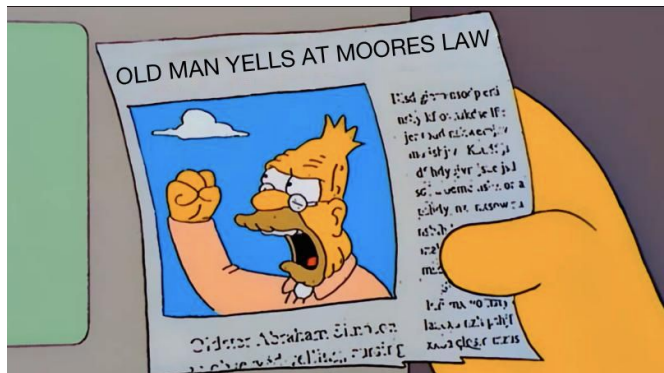


Cause?

- No single **application change** which obviously caused the problem
- No recent **infrastructure changes**
- No unusual **user activity** (eg, abuse, DDOS, etc)

Example: Redis CPU Saturation, May - Mid July





Potential Workarounds

- Faster CPUs
- Shard Redis cache
- Move to Redis Cluster
- Fixed several (old) inefficient caching operations

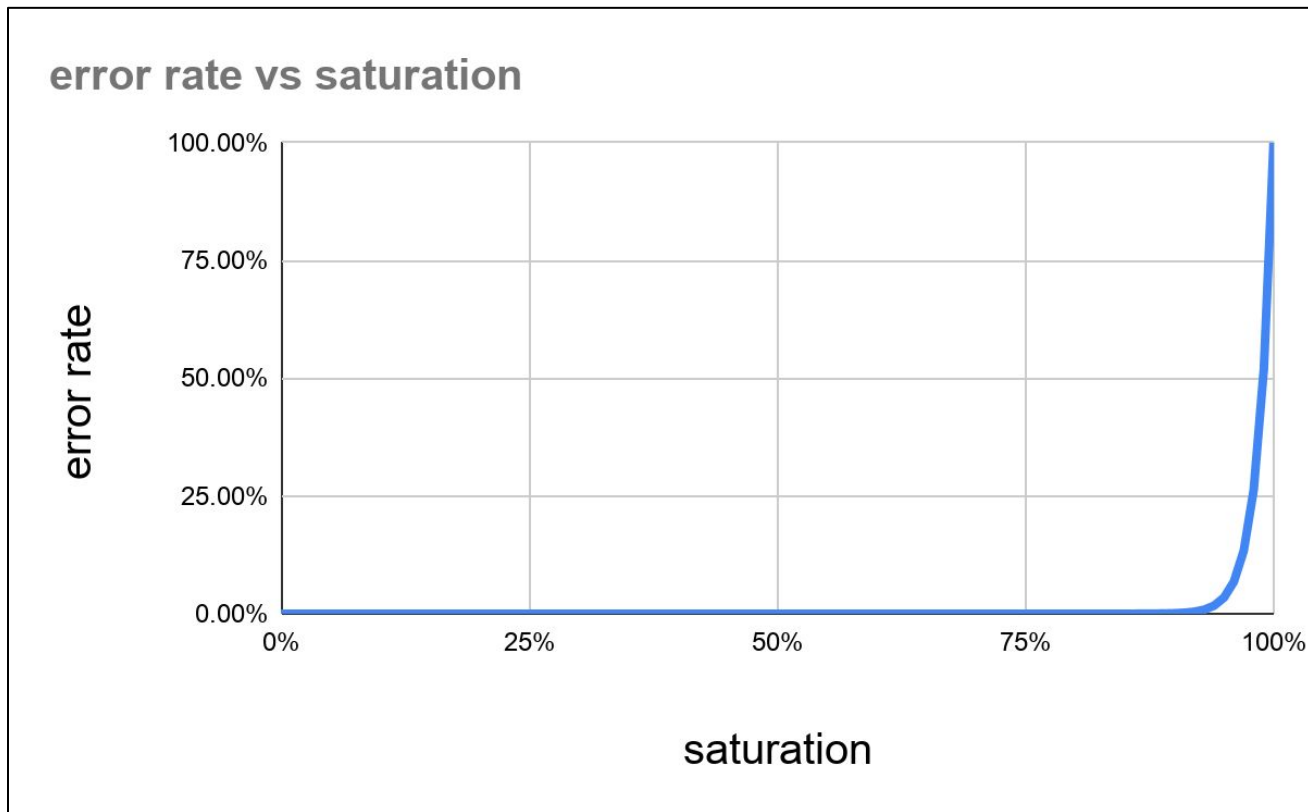


Learnings

1. Symptom-based alerting only warned us once it was too late
2. Resolving saturation problems may require time
3. Forewarning of the trend towards saturation would have helped a lot

We need better capacity planning. Can we use Prometheus for this?

Failure is not Linear





Goals

1. Model saturation as a key metric for each of our services
2. Model every potential saturation point in the application
3. Provide a forecast of resources that are most likely to breach their saturation limits in the next few weeks, giving us time to address these issues before they breach



$$\textit{Saturation} = \frac{\textit{Current Resource Usage}}{\textit{Maximum Possible Resource Usage}}$$

0: “Not Saturated”

“Completely Saturated”: 1



Saturation Measurement Recording Rules



Setup a recording rule with two fixed dimensions (labels)

`service_component:saturation:ratio`

Two Fixed Dimensions/Labels

- “**service**” the service reporting the resource
eg `service="web"` or `service="postgres"`
- “**component**” dimension - the component resource we are measuring
eg `component="memory"` or `component="cpu"`

All series report a **ratio** between 0 and 1. 0 is 0% (good). 1 = 100% Saturated (bad)



$$\textit{Saturation} = \frac{\textit{Current Resource Usage}}{\textit{Maximum Possible Resource Usage}}$$

`saturation_fds = process_open_fds / process_max_fds`

Example: File Descriptors



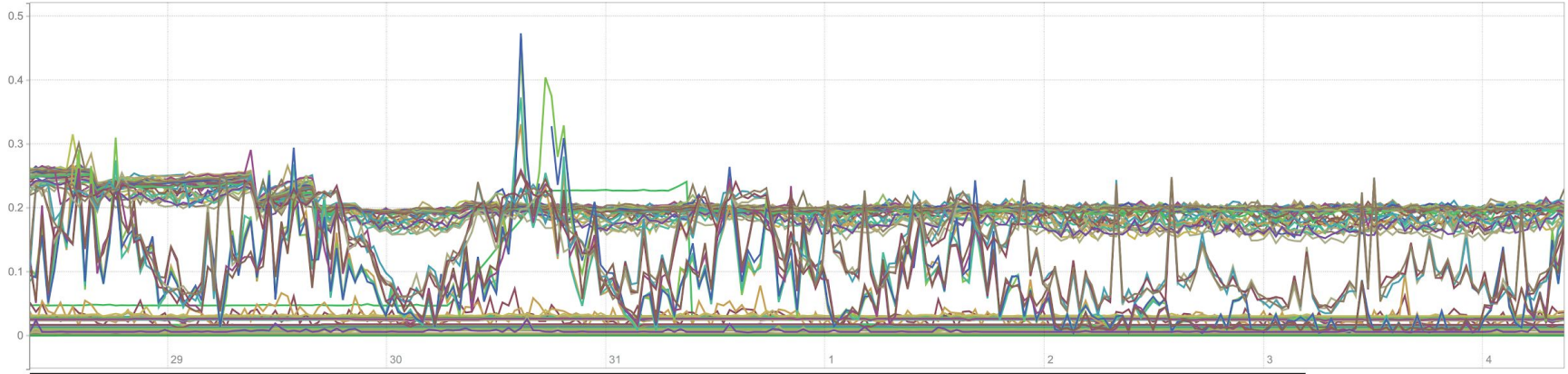
```
process_open_fds  
/  
process_max_fds
```

Load time: 23735
Resolution: 2419
Total time series:

Execute - insert metric at cursor - ▾

Graph Console

- 1w + ◀ Until ▶ Res. (s) ☐ stacked





```
saturation_fds =  
    max by (service) (  
        process_open_fds / process_max_fds  
    )
```


Example: File Descriptors



max by (service) (process_open_fds/process_max_fds)

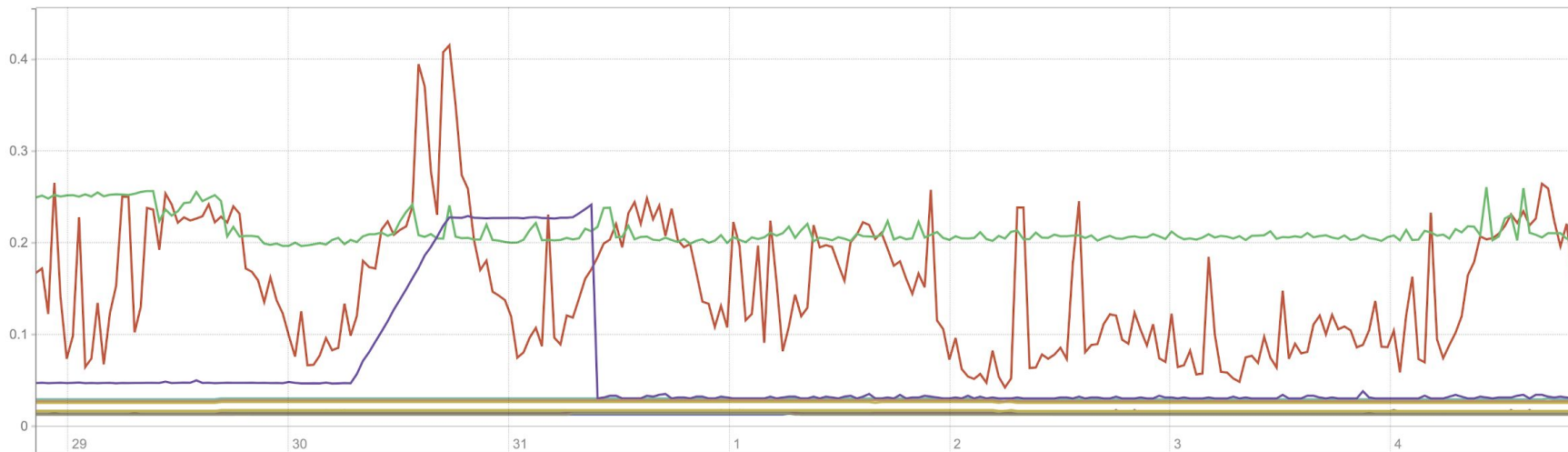
Execute

- insert metric at cursor - ↕

Load time: 13829ms
Resolution: 2419s
Total time series: 25

Graph Console

- 1w + ⏪ Until ⏩ Res. (s) ☐ stacked



File Descriptor Saturation Example



```
- record: service_component:saturation:ratio
labels:
  component: 'open_fds'
expr: >
  max by (service) (
    process_open_fds
    /
    process_max_fds
  )
```

```
# job_component:saturation:ratio{component="open_fds", service="gitaly"} 0.238
# job_component:saturation:ratio{component="open_fds", service="web"} 0.054
```



```
- record: service_component:saturation:ratio
labels:
  component: 'redis_cpu'
expr: >
  max by (service) (
    rate(redis_cpu_user_seconds_total[1m])
    +
    rate(redis_cpu_sys_seconds_total[1m])
  )
```

```
# service_component:saturation:ratio{component="redis_cpu", service="redis-cache"} 0.451
# service_component:saturation:ratio{component="redis_cpu", service="redis-sidekiq"} 0.324
```

Postgres Connection Saturation Example



```
- record: service_component:saturation:ratio
labels:
  component: 'pg_connections'
expr: >
  max by (service) (
    sum without (state, datname) (
      pg_stat_activity_count{state!="idle"}
    )
    /
    pg_settings_max_connections
  )
```

```
# service_component:saturation:ratio{component="pg_connections", service="postgres-1"} 0.2
# service_component:saturation:ratio{component="pg_connections", service="postgres-2"} 0.67
```

Other examples of saturation metrics



Server Workers: unicorn worker processes, puma threads, sidekiq worker

Disk: disk space, disk throughput, disk IOPs

CPU: compute utilization across all nodes in a service, most saturated node

Memory: node memory, cgroup memory

Database Pools: postgres connections, redis connections, pgbouncer pools

Cloud: Cloud quota limits (work-in-progress...)

Generalised alert for all saturation metrics



- alert: SaturationOutOfBounds

expr: service_component:saturation:ratio > 0.95

for: 5m

annotations:

title: |

The `{{ \$labels.service }}` service,

`{{ \$labels.component }}` component

has a saturation exceeding 95%



1 reply 1 day ago Yesterday

Slackline APP 14:10

Alert Firing: The `ci-runners` service (`main` stage), `shared_runners` component has a saturation exceeding SLO and is close to its capacity limit.


The `ci-runners` service (`main` stage), `shared_runners` component has a saturation exceeding SLO and is close to its capacity limit.

This means that this resource is running close to capacity and is at risk of exceeding its current capacity limit.

Labels:

component	env
shared_runners	grpd
environment	region
grpd	us-east
severity	stage
s3	main
tier	type
runners	ci-runners

AlertManager (90 kB)



2 replies Last reply 1 day ago


Message #alerts-general

Alert details

Embedded
Grafana
panel

Threaded
resolve
message w/
embedded
panel

Thread
#alerts-general



2 replies

Slackline APP 1 day ago

Further Investigation...

Runbook Service Dashboard

Component Dashboard Prometheus

Take Action...


Silence Open Issue

Slackline APP 1 day ago

Alert Resolved

Resolved: The `ci-runners` service (`main` stage), `shared_runners` component has a saturation exceeding SLO and is close to its capacity limit.

AlertManager (91 kB)



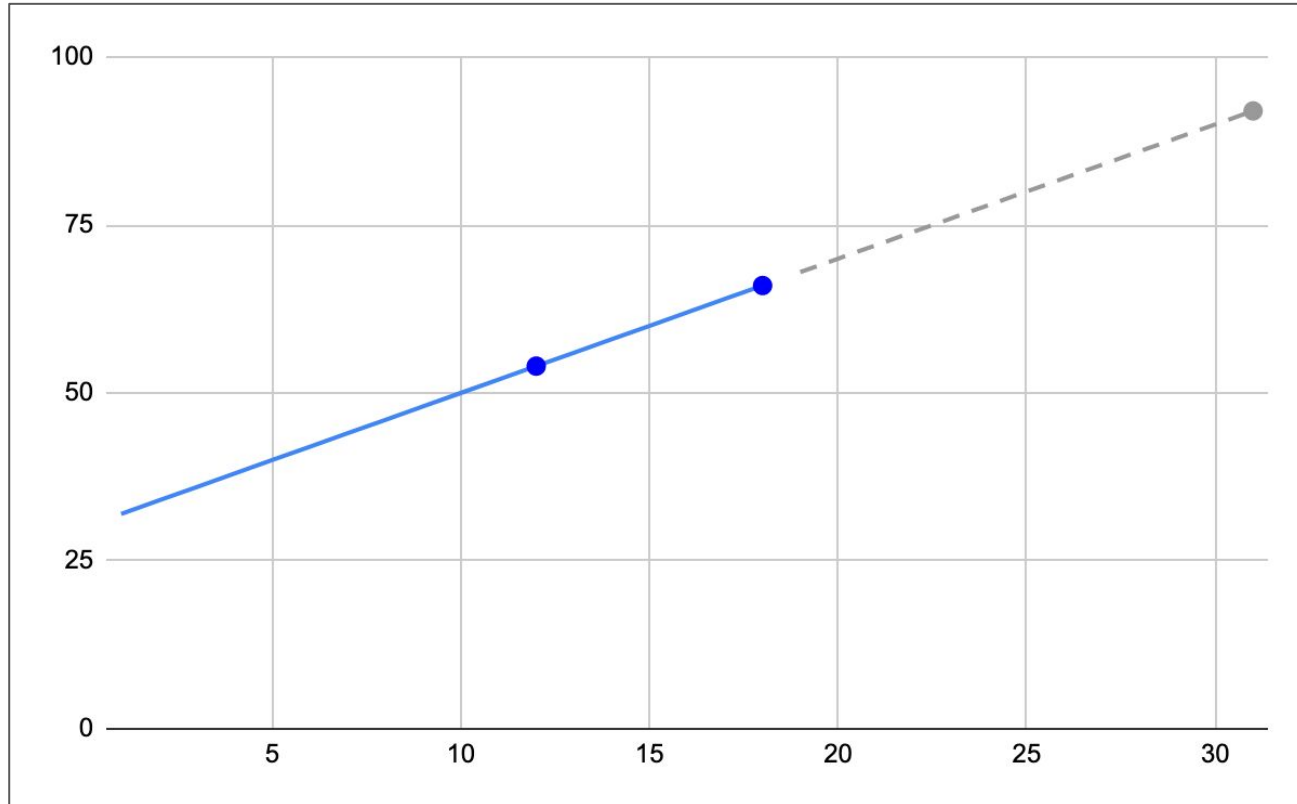
Reply...

Quick links
+ quick
actions

Capacity Planning and Forecasting



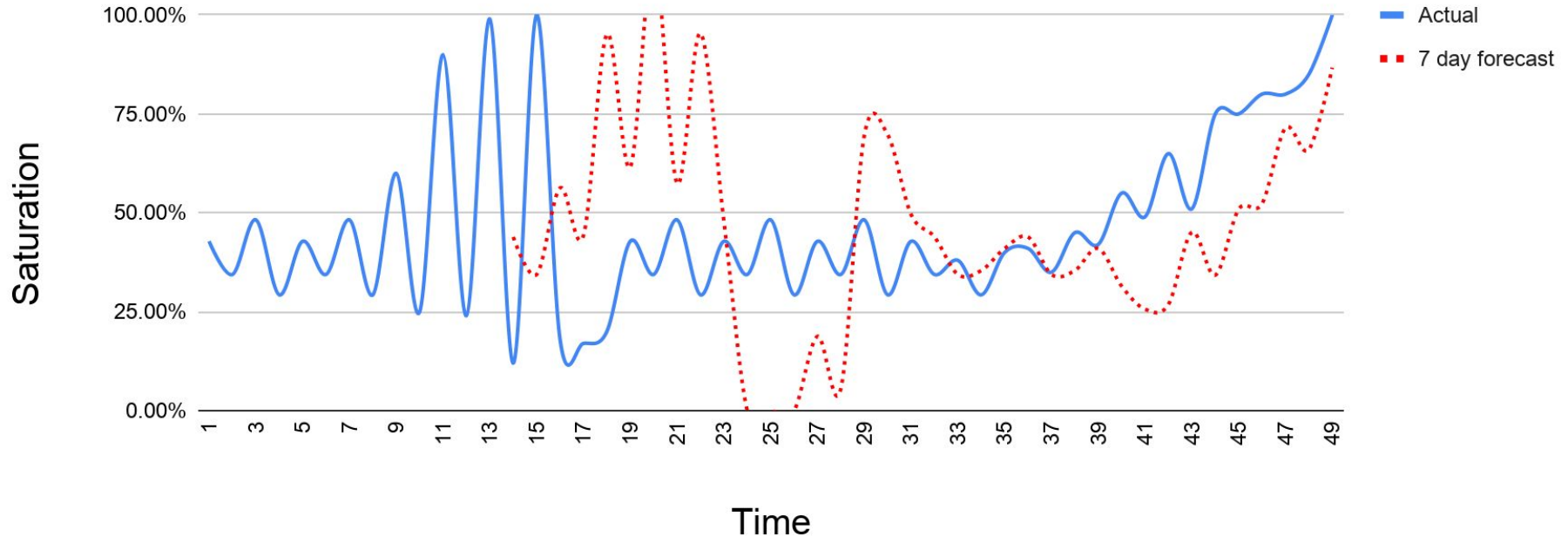
Can we use Linear Interpolation?



Linear interpolation doesn't work well on non-linear data



7 day Forecast vs Actual



Then an idea struck us...



A hurricane warning, not a weather forecast...





Estimated Worst Case Prediction Calculation:

1. **Trend Forecast:** Use linear prediction on our rolling 7 day average to extend the trend forward by 2 weeks
2. **Standard Deviation (σ):** Calculate the standard deviation for each metric for the past week
3. **Worst Case:** 2w Trend Prediction + 2σ

Estimating a worst-case with standard deviation



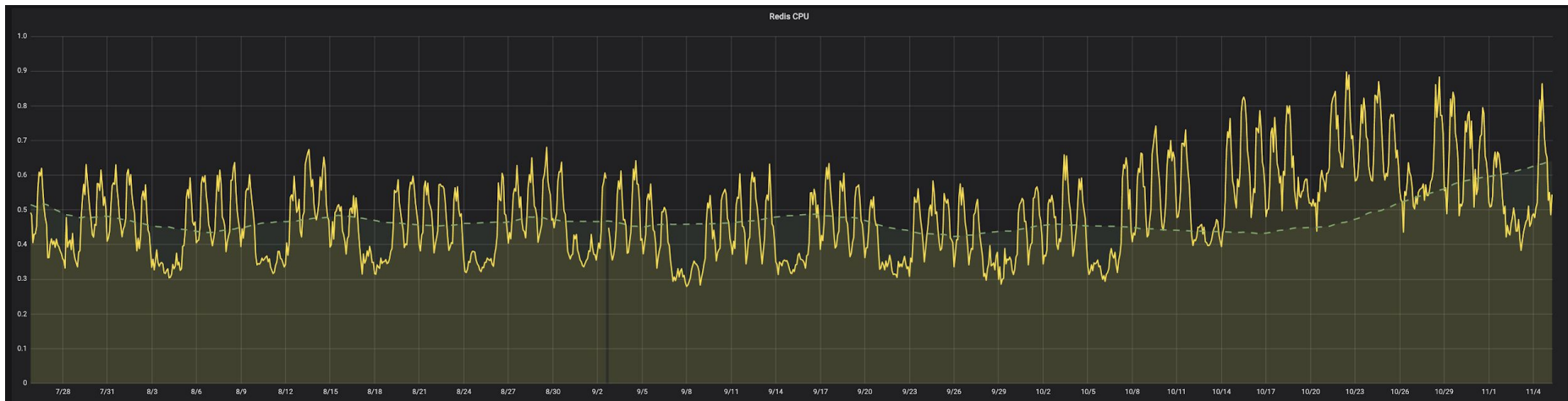
Saturation Metric: Redis CPU



Estimating a worst-case with standard deviation



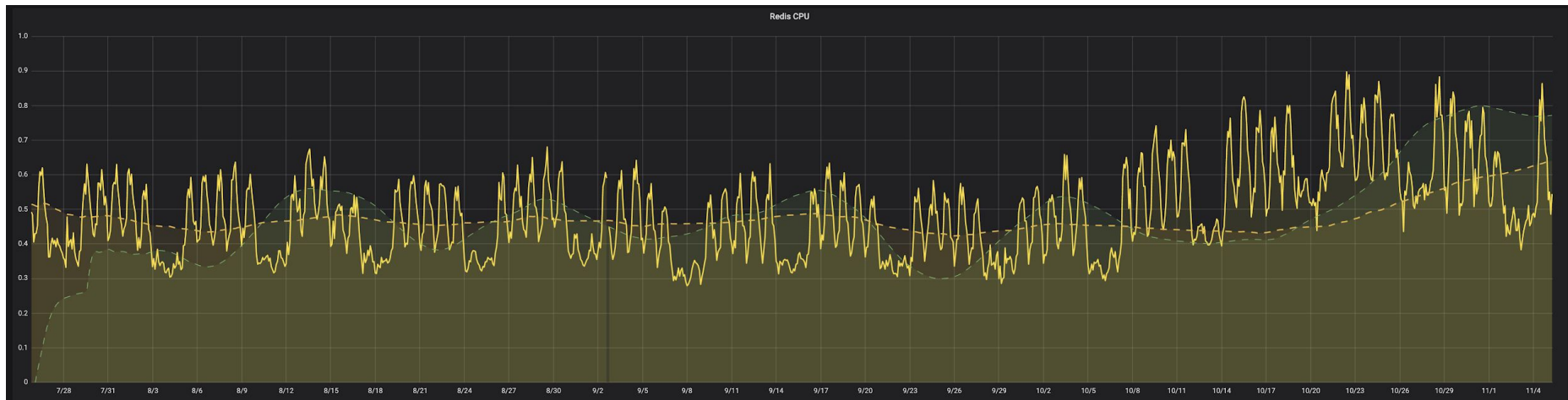
Redis CPU Trend: 7-day Rolling Average



Estimating a worst-case with standard deviation



Linear Interpolate on the Trend



Estimating a worst-case with standard deviation



Account for variance by adding 2σ





Average values for each component, over a week

- record: service_component:saturation:ratio:avg_over_time_1w

expr: >

```
avg_over_time(service_component:saturation:ratio[1w])
```

Stddev for each resource saturation component, over a week

- record: service_component:saturation:ratio:stddev_over_time_1w

expr: >

```
stddev_over_time(service_component:saturation:ratio[1w])
```



- record: service_component:saturation:ratio:predict_linear_2w

expr: >

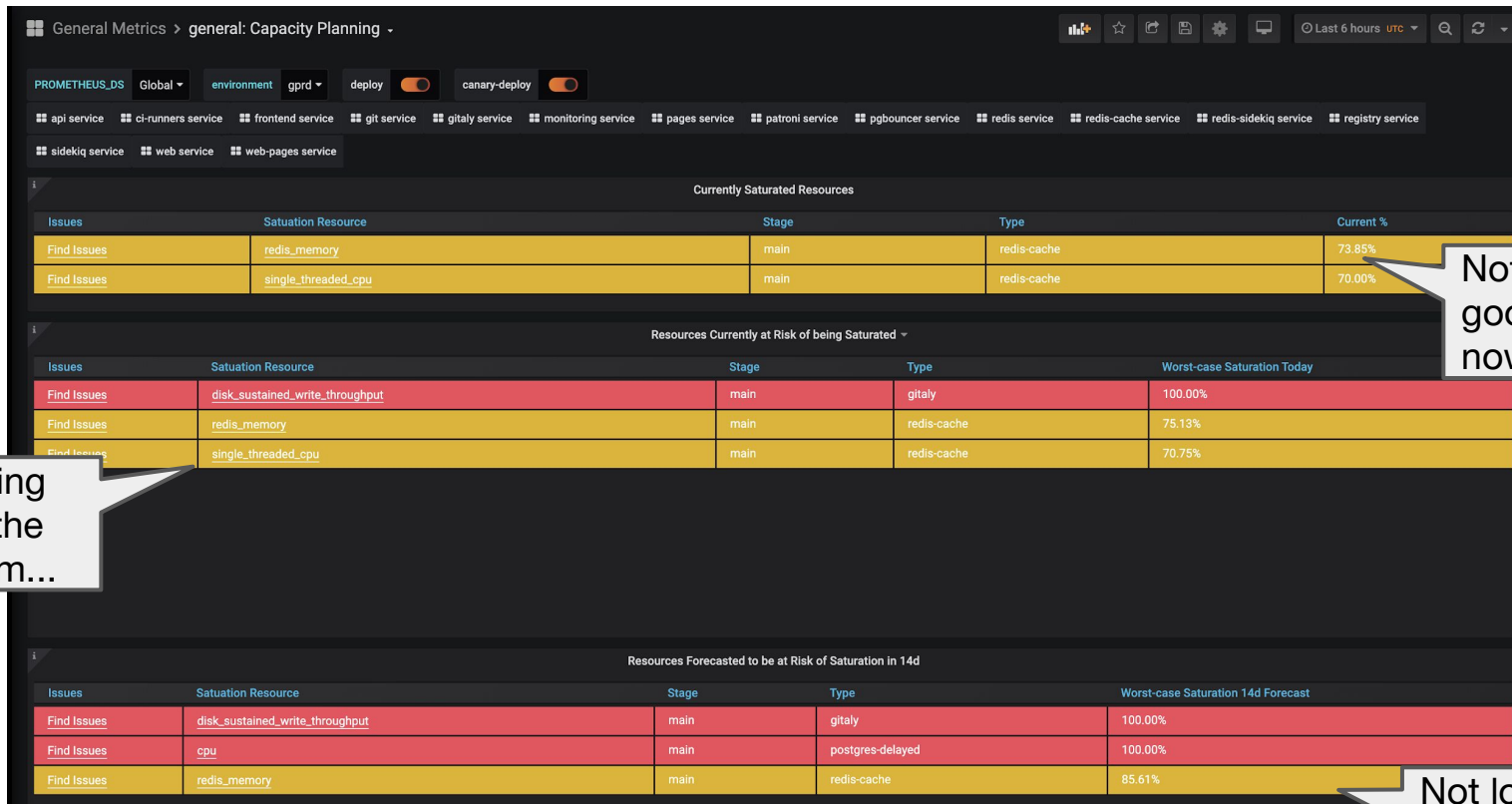
```
predict_linear(
```

```
    service_component:saturation:ratio:avg_over_time_1w[1w],
```

```
    86400 * 14 # 14 days, in seconds
```

```
)
```

Capacity Planning Report



Not looking good right now

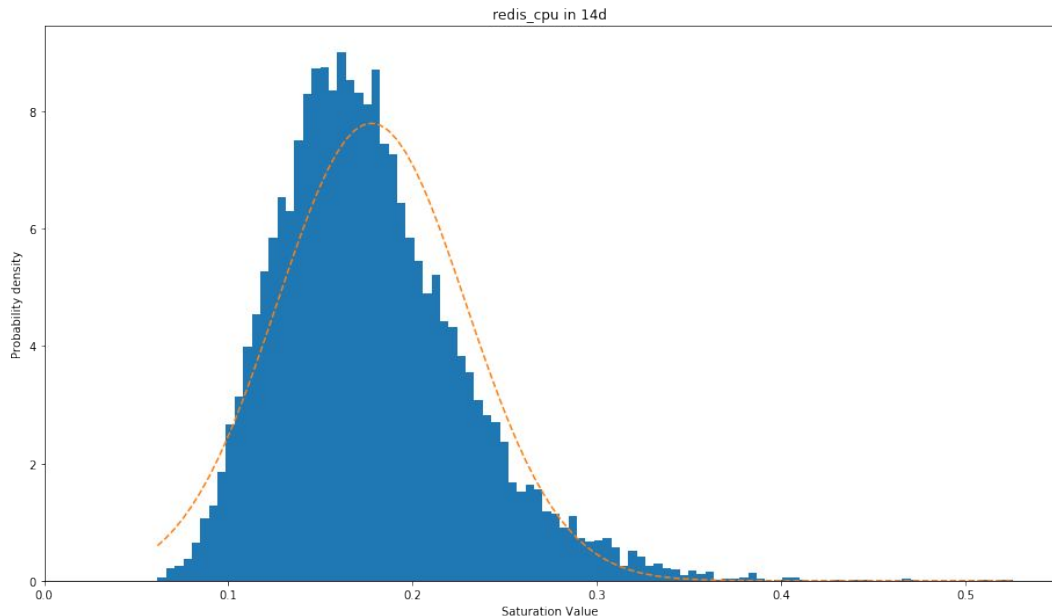
Not looking good in the short term...

Not looking good over the next few weeks

<https://dashboards.gitlab.com/d/general-capacity-planning>



Calculate the predictions outside Prometheus?



Example: using python/numpy to perform Monte-Carlo simulations to predict saturation.

Overkill much?



Capacity Planning Dashboard:

- **Reports on potential future saturation problems** based on week-on-week growth trends and volatility in our data
- **Used for further, deeper analysis and planning** - we don't alert based on this data
- **Early days** - still figuring this out. Would love to get feedback!

Questions?



GitLab.com Resource Saturation Monitoring and Capacity Planning rules at:

Saturation Metrics

https://gitlab.com/gitlab-com/runbooks/blob/master/rules/service_saturation.yml

Saturation Alerts

<https://gitlab.com/gitlab-com/runbooks/blob/master/rules/general-service-alerts.yml>

Capacity Planning Dashboard (grafonnet examples 🙌)

<https://gitlab.com/gitlab-com/runbooks/blob/master/dashboards/general/capacity-planning.jsonnet>

We're hiring!

<https://about.gitlab.com/jobs/apply/>

Questions?