Best practices for handling Black Friday traffic

Tobiasz Heller
Ingrid

Agenda

- Black week traffic explanation

- Preparing load tests suite

- Finding bottlenecks in your system

E-commerce and Black Week

- +/- 10 days of very high traffic

- E-commerce merchants prepare marketing plan since summer

- New marketing strategies (influencers)

Ingrid checkout

- January 2020 - 50 RPS

- Load tests - 3000 RPS

- Black Friday 2019 - 200 RPS

Ingrid architecture

- +/- 80 microservices

- Almost all written in Go

- K8S on GCP GKE

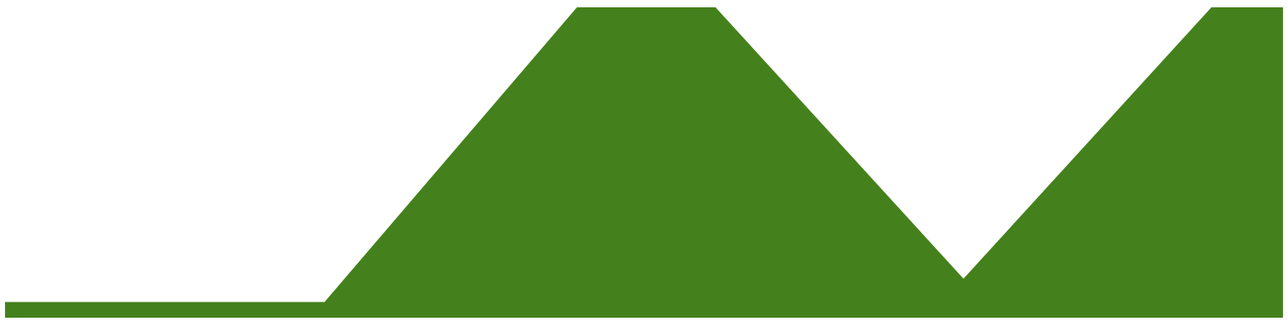
- gRPC + google Pub/Sub for internal comm

- REST API via gRPC gateway

Load/traffic types

High stable (no spikes) traffic

Traffic spikes



Simulating high traffic

- Prepare test scheme

- Bootstrap tests on stage

- Real tests on production

- Observability!

Observability

- Prometheus + Thanos

- Metrics system need to scale with load

- Sharding of metrics

Alerting

- Separate oncall (immediate action required) and other alerts
- Alerts need to be easily **configurable** and adjustable

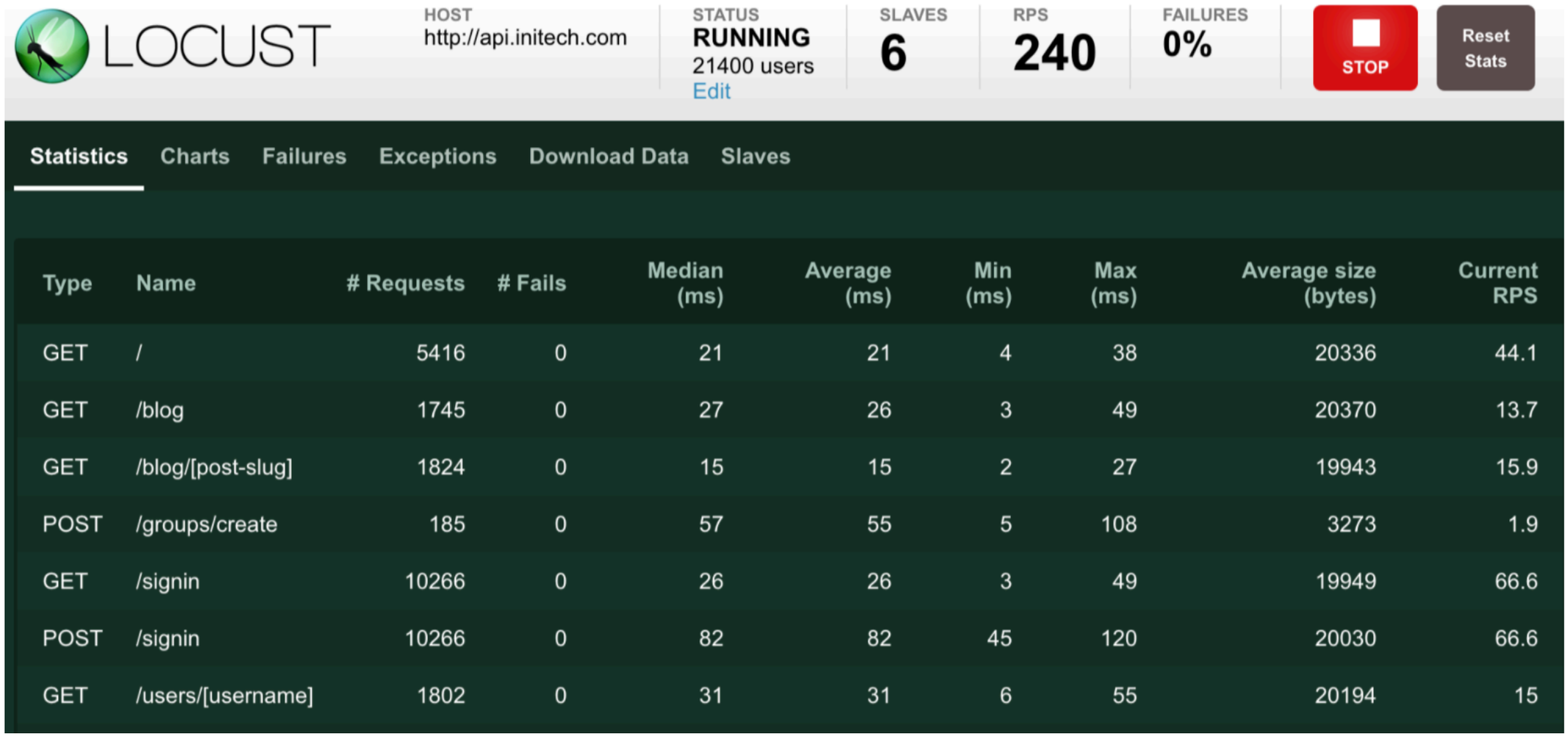
Simulating high traffic

locust.io

- Open source load testing tool
- Define user behaviour in python code
- Distributed & scalable
  - Helm chart with locust cluster
- Web-based UI

Simulating high traffic

locust.io



Type	Name	# Requests	# Fails	Median (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS
GET	/	5416	0	21	21	4	38	20336	44.1
GET	/blog	1745	0	27	26	3	49	20370	13.7
GET	/blog/[post-slug]	1824	0	15	15	2	27	19943	15.9
POST	/groups/create	185	0	57	55	5	108	3273	1.9
GET	/signin	10266	0	26	26	3	49	19949	66.6
POST	/signin	10266	0	82	82	45	120	20030	66.6
GET	/users/[username]	1802	0	31	31	6	55	20194	15

K8s in action

- Horizontal pod autoscaling

- Node autoscaling
  - Scaling from 10 to 100 nodes in few minutes

- Resources placeholder (pod priority)

K8s pod CPU/mem resources

- Requests resources - scheduled on a node that can give pod given resources
  - Min 100m of CPU for Go apps (on GCP)
- Limits resources - above limit value pod will be killed
  - Need to be a 20-40% bigger than requests in order to handle spike before autoscaling new pod

DB setup

- Mostly MySQL

- Some PostgreSQL

- Redis

DB bottlenecks

CPU limits

- Separate instance per service
- Monitor metrics github.com/luna-duclos/instrumentedsql
    - Review your indexes (move analytics one to other system)
- Vertical scaling

DB bottlenecks

Number of connections

- Cache connections to DB
- Separate instance per service
- Bigger pods = less amount of active connections

DB bottlenecks

GCP limits

- GCP limit number of new connections per 100s

- Bigger pods = less amount of active connections

In-memory cache in Go

- Optimised for fast reads

- Builtin sync and async refresh of entries

- Plans to open-source it

Profiling Go apps

- pprof and flame graphs are your friends!
  - Json encoding & jsoniter
  - Building traces span was taking most of CPU in one service
  - Creating new connection to DB

Demo

github.com/tobiaszheller/talks/tree/master/black-friday/demo

Pair programming

- Less probability of errors
- Better code quality
- Also a lot of fun

Others

- Reduce components on critical path

- Async communication if possible

Future plans

- Distributed DB

Thank you