Thunderdome

Ian Davis, Protocol Labs

IPFS Camp

Thunderdome?

Thunderdome

A gladiatorial arena where ~~conflicts are resolved by a fight to the death~~ performance of different IPFS Gateway configurations and implementations can be compared

IPFS Camp

What does Thunderdome do?

- Runs experiments on demand and records metrics for the things being tested
- Currently it supports testing IPFS gateway implementations with HTTP traffic
- Designed to run for long periods - days
- Given an experiment definition it starts all the necessary infrastructure, applies load and monitors

IPFS Camp

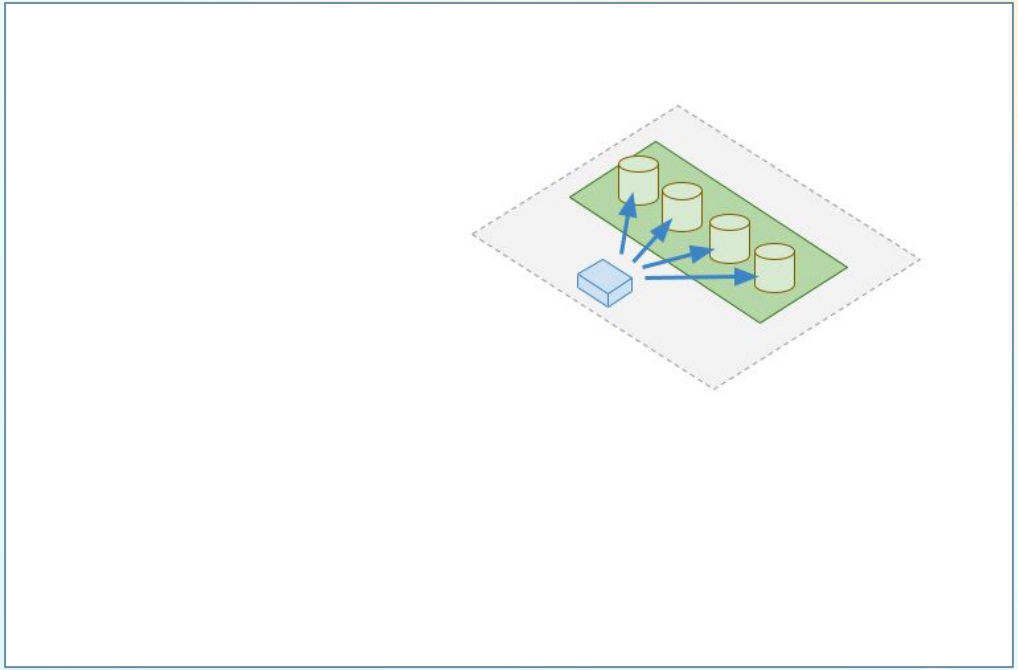
Thunderdome

Infrastructure

IPFS Camp

What is an experiment?

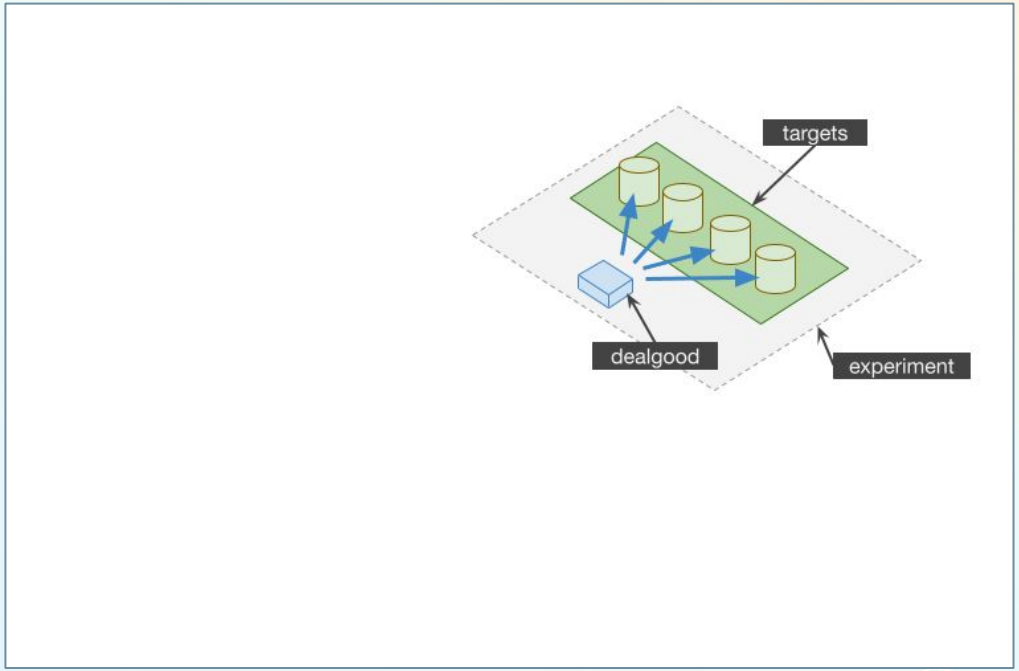
- At its simplest: fire requests at a bunch of ipfs nodes
- Today these are HTTP requests intended to test IPFS gateways



IPFS Camp

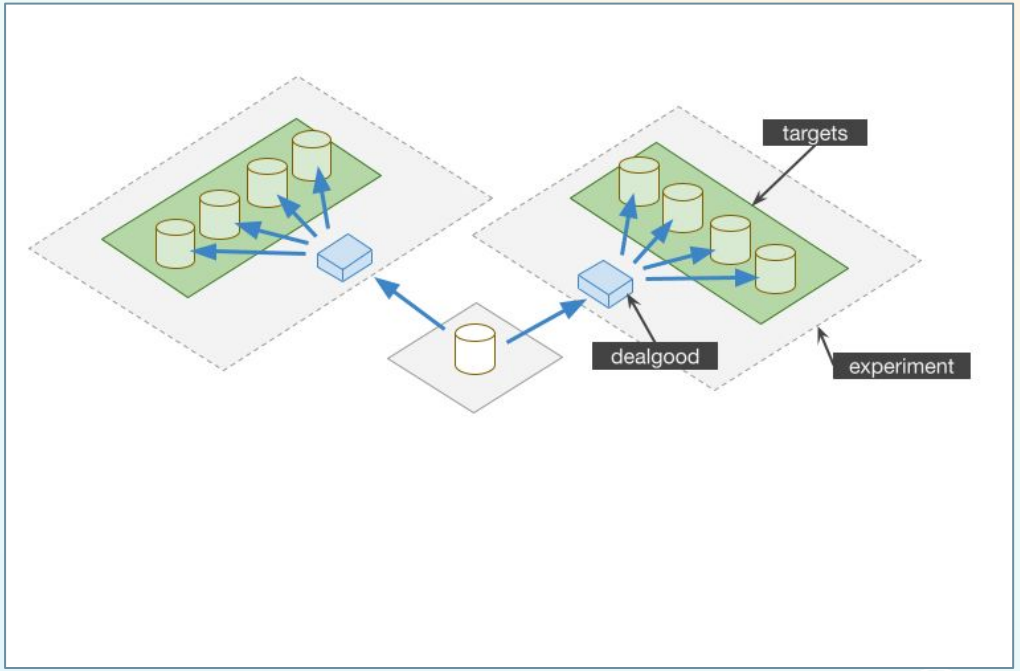
What goes into an experiment?

- experiment defines
  - request rate
  - concurrency
  - target config
- target - a docker container in ECS
- dealgood - applies load evenly to targets



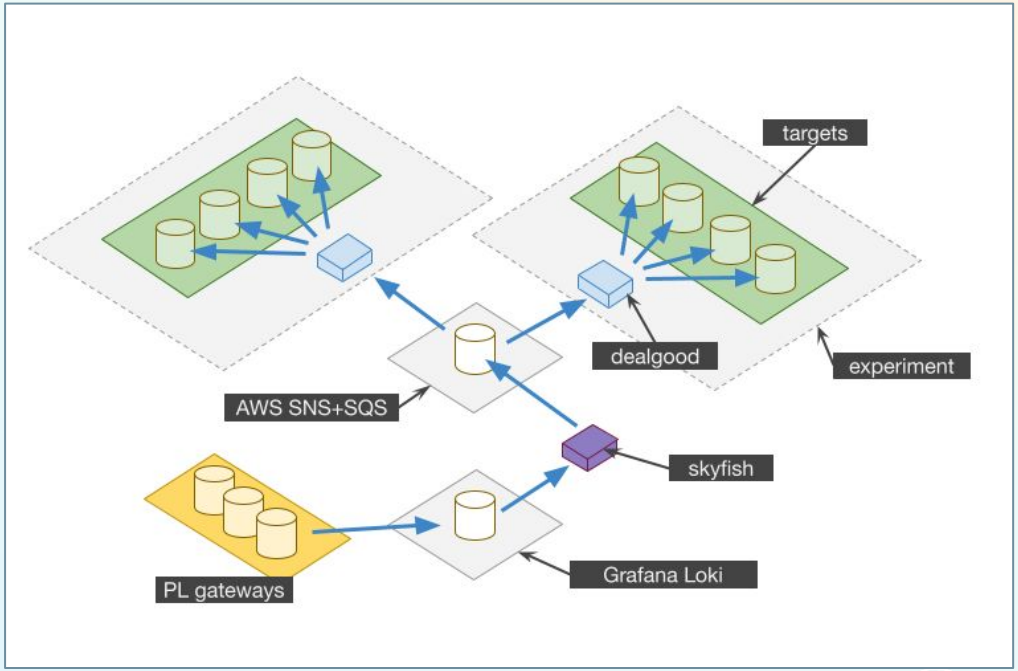
Multiple experiments

- Each experiment is independent
- Each instance of dealgood controls the load applied to that experiment's targets



Where do requests come from?

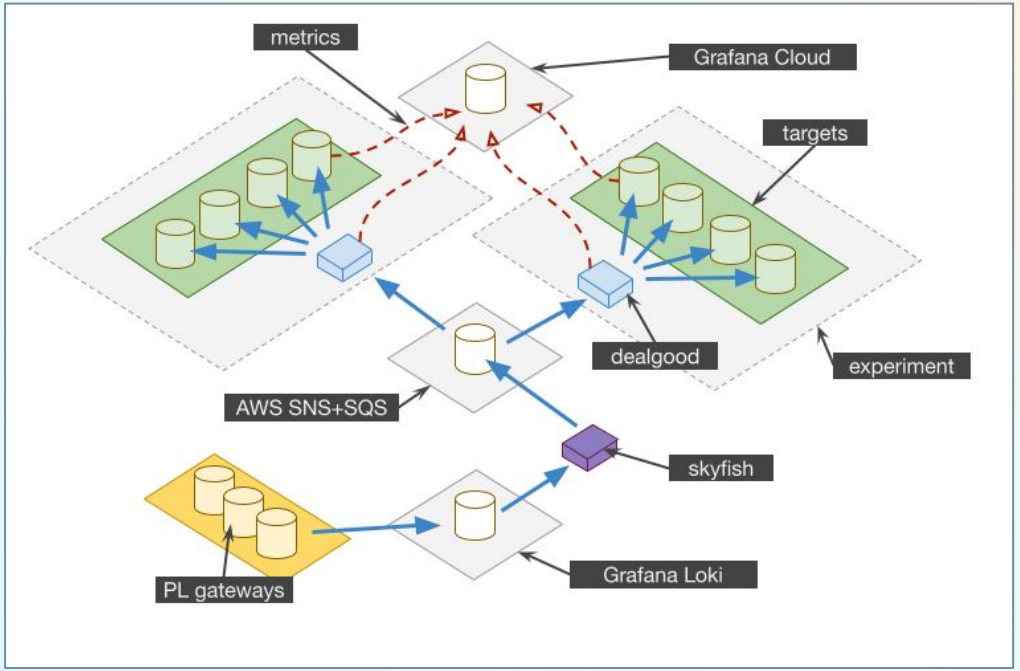
- HTTP requests are sampled from ipfs.io gateways into Loki
- skyfish - bridges Loki and AWS SNS
- dealgood - listens to its own SQS queue fed by SNS



IPFS Camp

Metric collection

- Metrics are read from target containers and dealgood
- All available in Grafana cloud for dashboarding

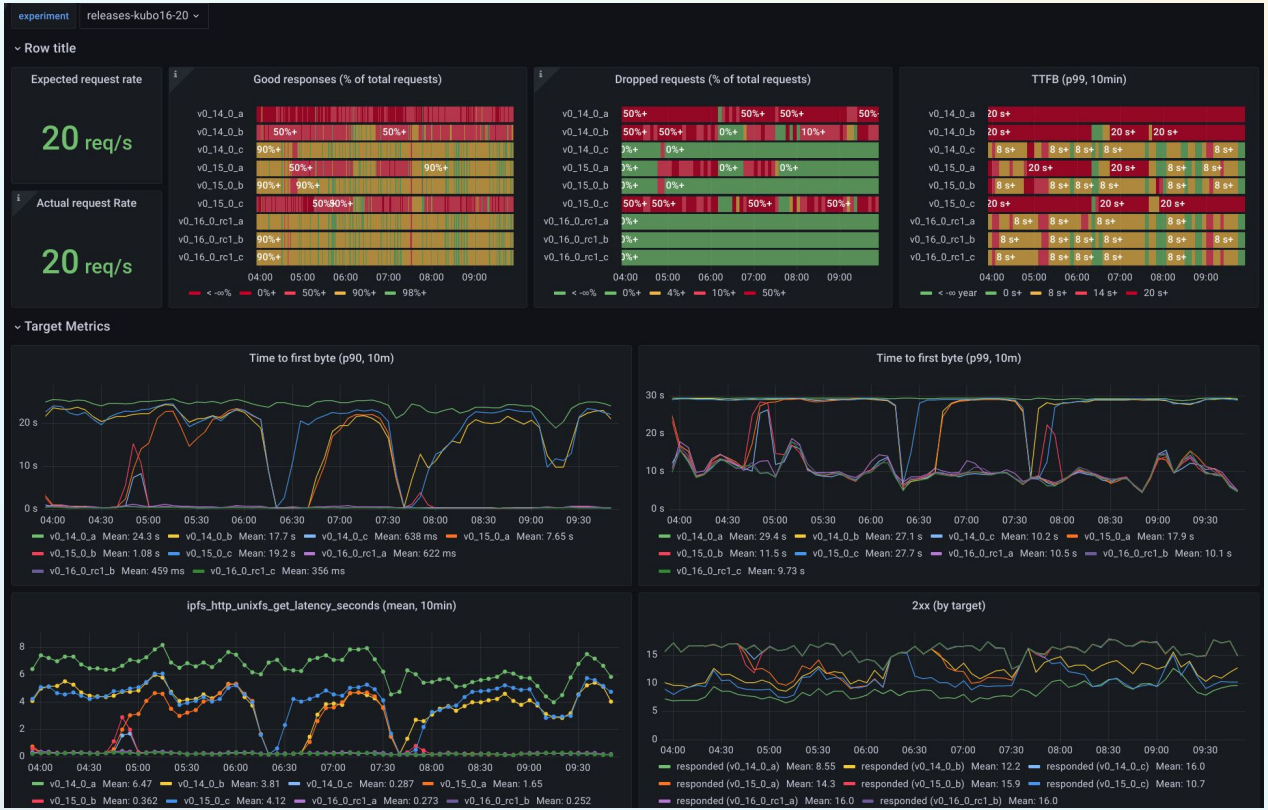


IPFS Camp

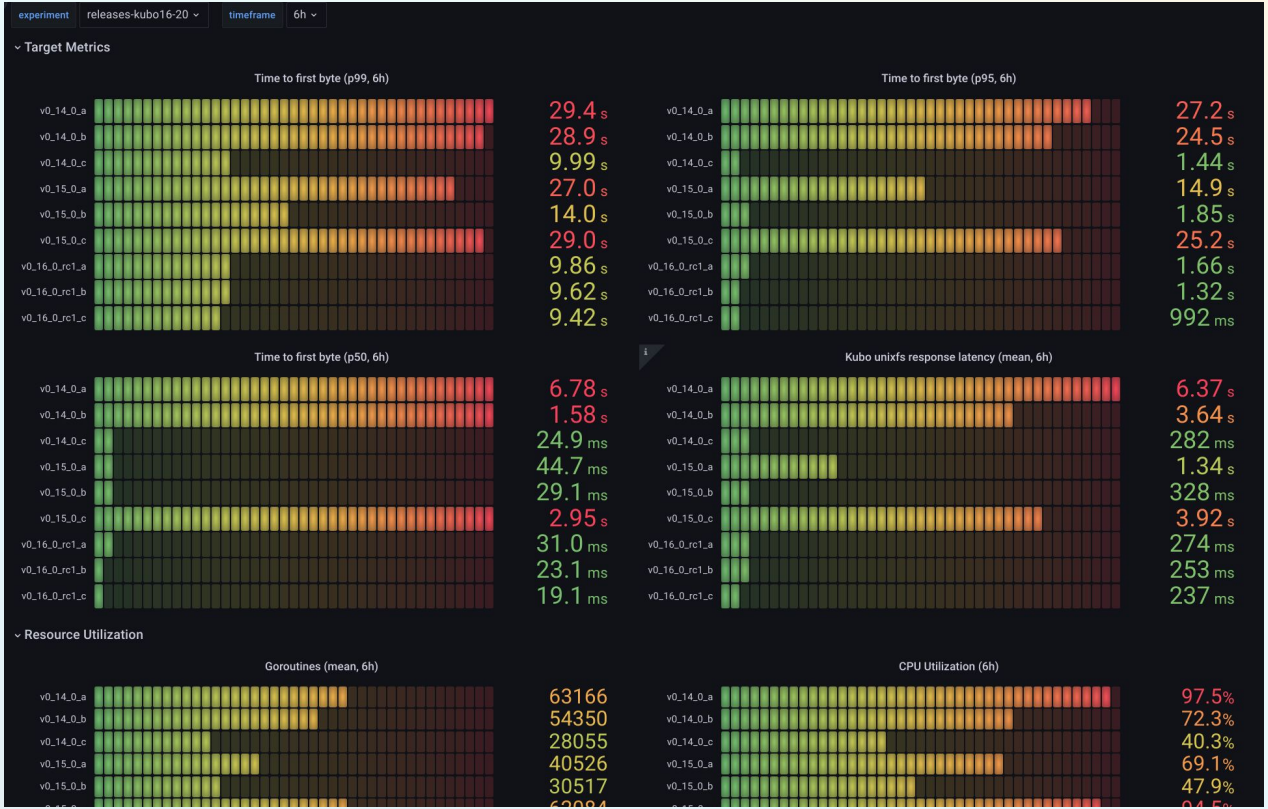
Experiment results

- Grafana holds all relevant metrics from an experiment
- Collected via prometheus from:
  - targets
  - dealgood
  - skyfish
- Tagged by experiment and source (job)

Experiment timelines

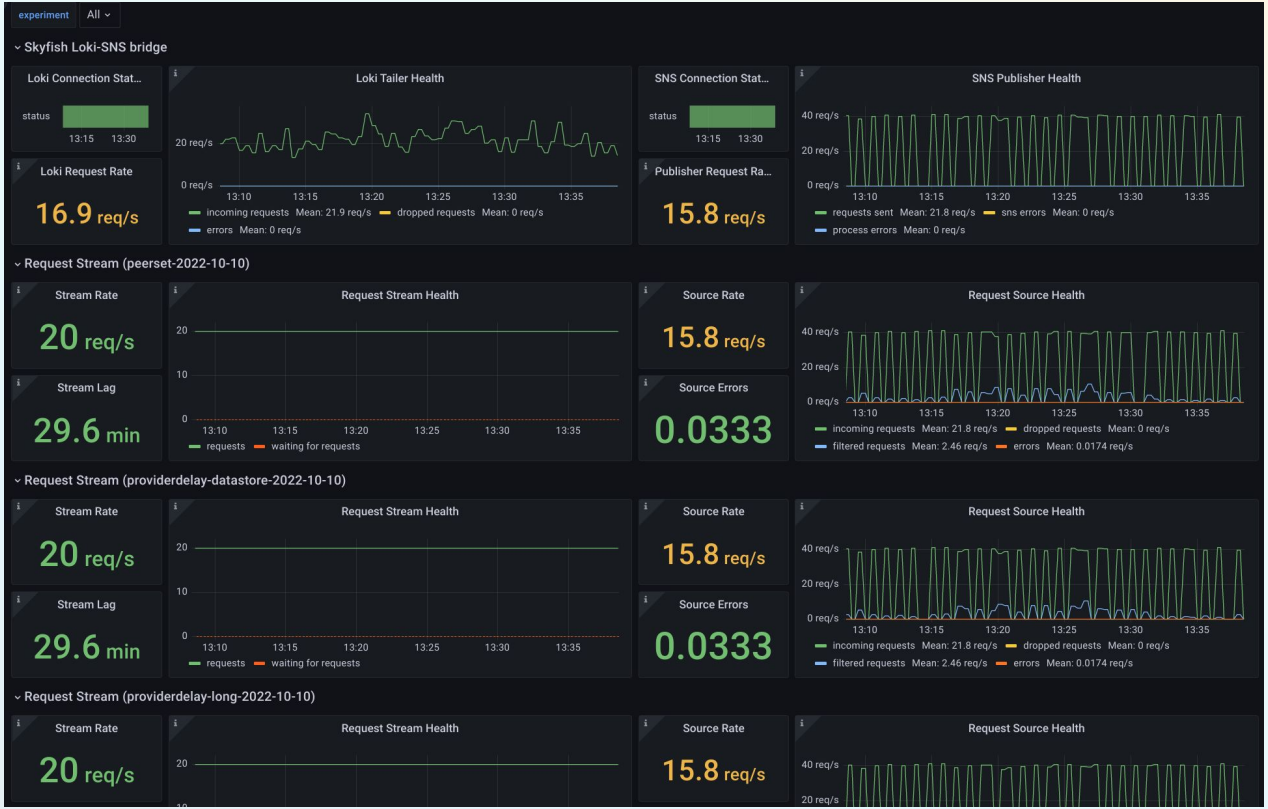


Experiment summaries



IPFS Camp

Experiment health



IPFS Camp

Infrastructure challenges

- The nature of ipfs and p2p makes it hard to test instances in isolation
- Nodes like to chat to their neighbors
- At a minimum we want to avoid test targets from fetching from one another
- Network ACLs isolate each target
- Tracking IPFS work on peer connection rules

IPFS Camp

Thunderdome

Experiments

IPFS Camp

Initial experiments

- Our focus has been on proving Thunderdome
  - Ensuring no bias in infrastructure
  - Scaling request streams
- Picked some easy experiments:
  - Baseline experiment
  - Comparing kubo releases
  - Bitswap provider delays

IPFS Camp

Experiment: tweedles

- Baseline experiment
- Two identical Kubo instances
  - tweedle-dee and tweedle-dum
  - Same configuration
  - Same load
  - Should perform identically
- Detect bias in Thunderdome infrastructure

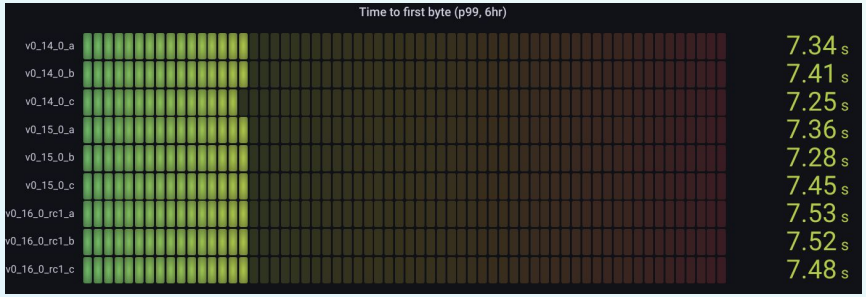
IPFS Camp

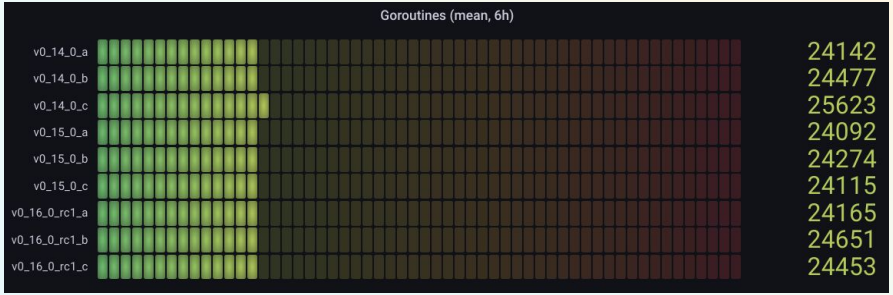
Experiment: kubo v0.16 release

- Version comparison
- Two pairs of each Kubo version:
  - v0.14, v0.15 and v0.16 rc1
  - Same configuration
  - 10 rps and 20 rps
- Detect potential performance regressions or improvements

Comparison at 10 rps

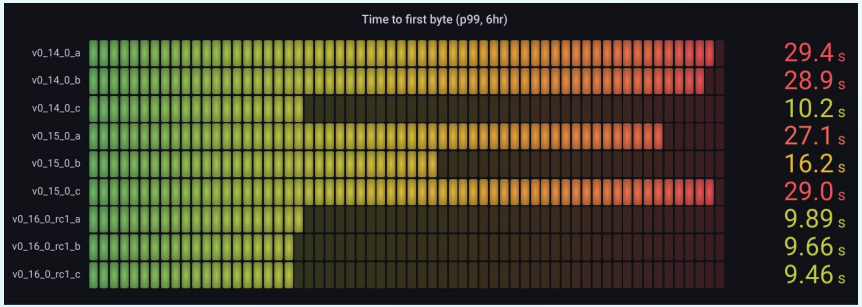
All quiet…

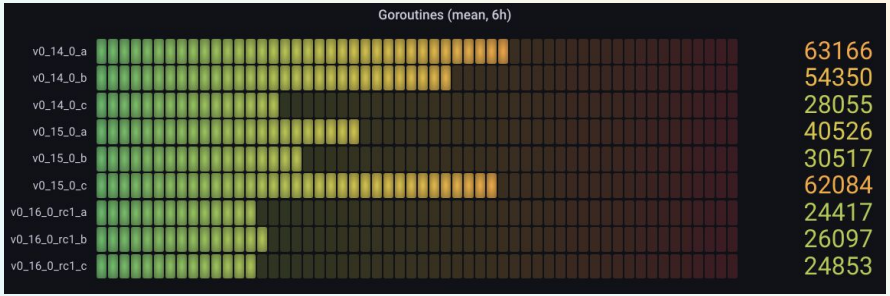


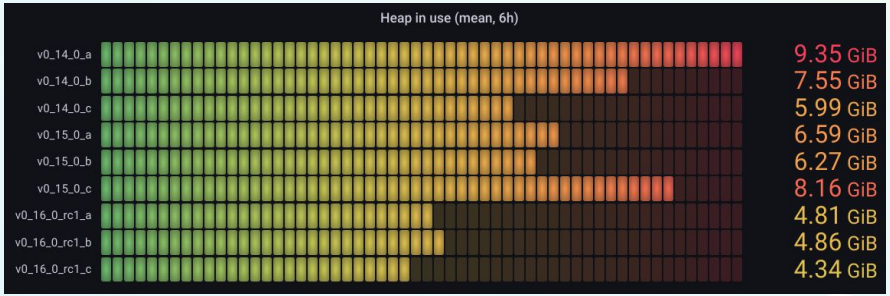


Comparison at 20 rps

Some things blew up…



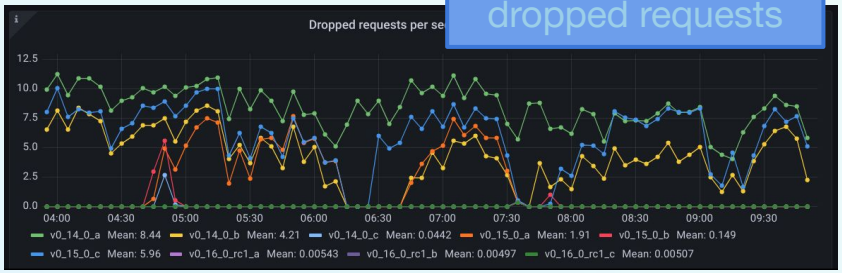




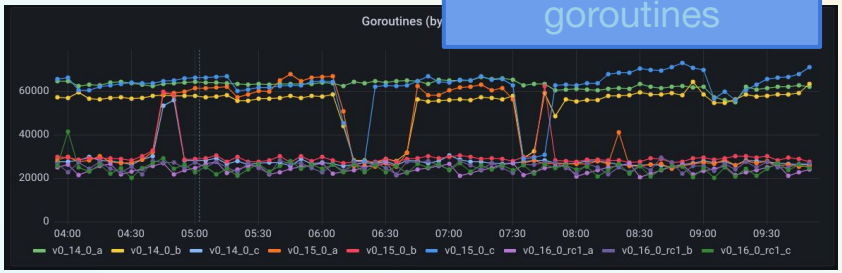
IPFS Camp

Comparison at 20 rps

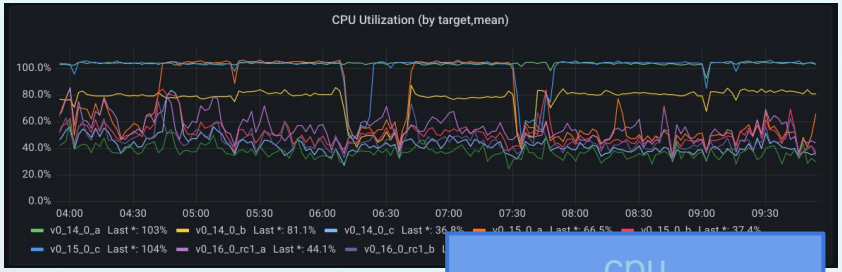
Over time…



dropped requests



goroutines



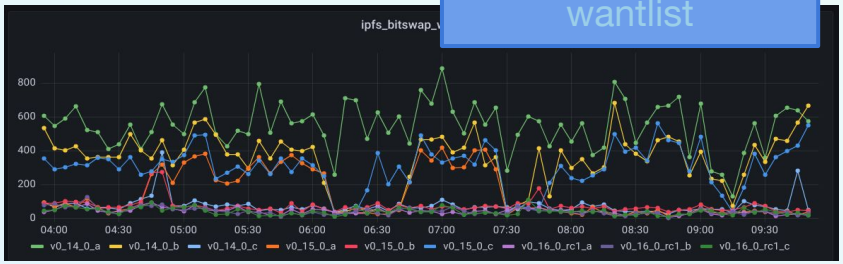
cpu

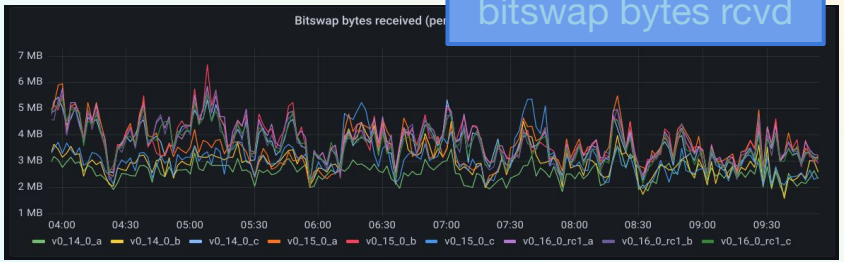
heap

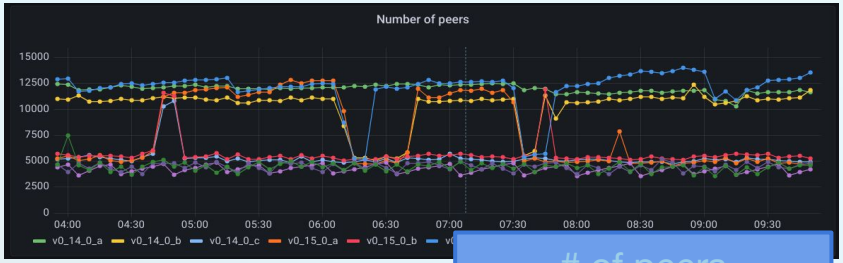
IPFS Camp

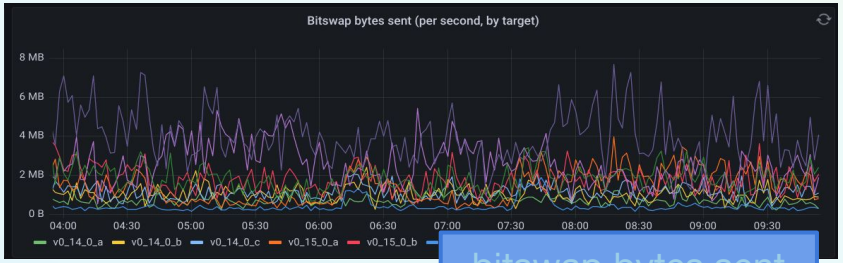
Comparison at 20 rps

Bitswap behavior - elevated peer lists





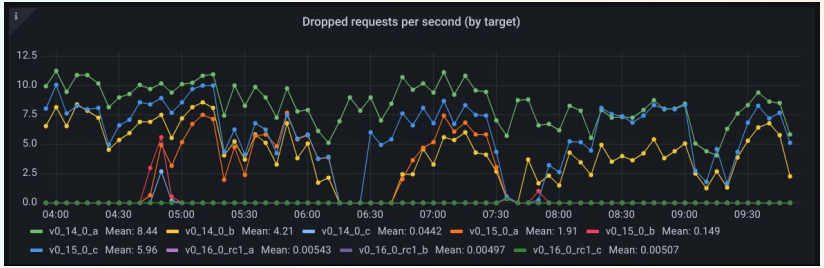


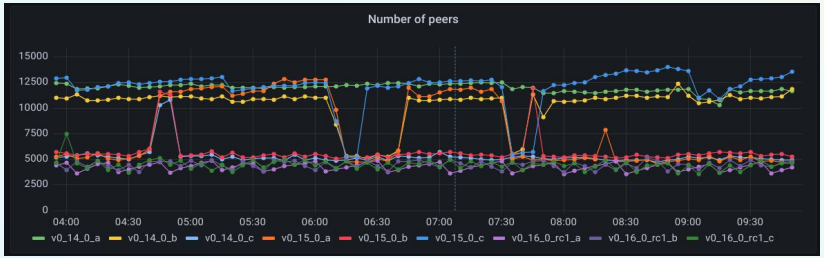


IPFS Camp

Observations

- Kubo can get trapped in a state with elevated numbers of peers which causes:
  - Large heap
  - High goroutines + CPU
  - Larger wantlists
  - High TTFB
  - Poor throughput

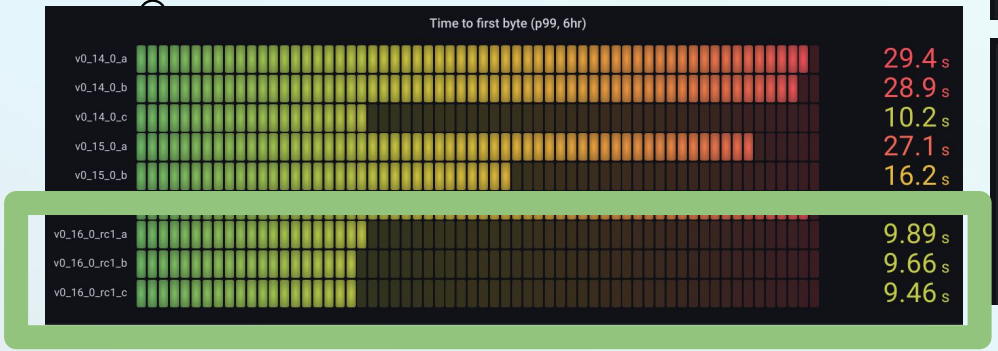


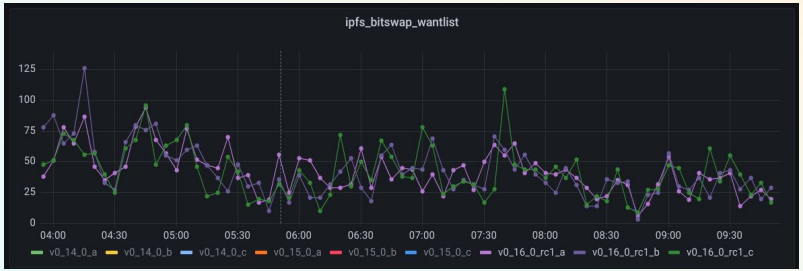


IPFS Camp

Conclusion

- But… Kubo 0.16.0 rc1 seemed to avoid falling into this state





Follow up

- Followed up with comparison of various commits between v0.15 and v0.16
- Narrowed down point at which Kubo appears to behave better - routing logic changes
- Read the experiment summaries
  - https://tinyurl.com/45mhhf89

Other Experiments

- **gwblock** — Compared performance of Kubo when instances are blocked from connecting to public gateway instances
- **providerdelay-base** — Compared various bitswap provider delay settings
- **providerdelay-peerset** — Compared various provider delays with high water of peer connections of 600
- **providerdelay-acceldht** — Same as base provider search delay experiment but accelerated dht is disabled
- **providerdelay-datastore** — Same as base provider search delay experiment but the datastore is limited to 1GB to force block retrieval
- **providerdelay-long** — Same as base provider search delay experiment but using much longer delays (10/20/30s)
- **peerset** — Latest kubo comparing settings for number of peer connections

Thunderdome

Future

IPFS Camp

Roadmap

- Q4 2022 - Automatic Release Comparisons
  - Test every Kubo release candidate against previous release
- Q1 2023 - Single Click Experiments
  - Allow collaborators to create and run experiments
- Q2 2023 - Alternate Gateway Implementations
  - Experiments with iroh, pomegranate etc.

IPFS Camp

Long term vision

- More target types - not just IPFS gateway implementations
- More load sources - support protocols other than HTTP such as bitswap
- Bring your own infra - enable deployment on non-AWS infrastructure
- More experiments… more data…

IPFS Camp

Thunderdome

Get Involved

IPFS Camp

Got an experiment?

- Find me on Filecoin slack in #prodeng
- Or on IPFS discord in #probe-lab
- Raise an issue in the thunderdome Github repo:

  https://github.com/ipfs-shipyard/thunderdome/issues

(soon you can run your own)

Get the source code

- It's open source on Github

  https://github.com/ipfs-shipyard/thunderdome

IPFS Camp

Thank you

IPFS Camp