Metrics at Full Throttle: Intro and Deep Dive into ***Thanos***

@saswatamcode @fpetkovsky

Hello!

Filip Petkovski

- Staff Production Engineer @ Shopify
- Thanos Maintainer
- @fpetkovski on Twitter/GitHub
- Based in Munich, Germany

Hello!

Saswata Mukherjee

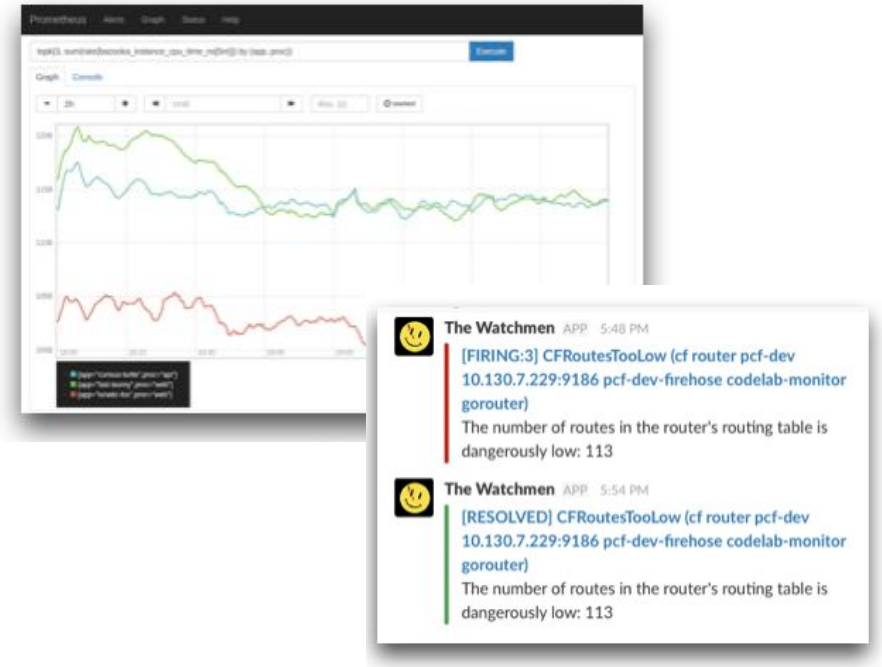
- Software Engineer @ Red Hat
- Thanos, Observatorium, mdox, other Go lib Maintainer
- GSoC'21 Thanos mentee
- @saswatamcode on Twitter/GitHub/elsewhere

Prometheus!

Prometheus

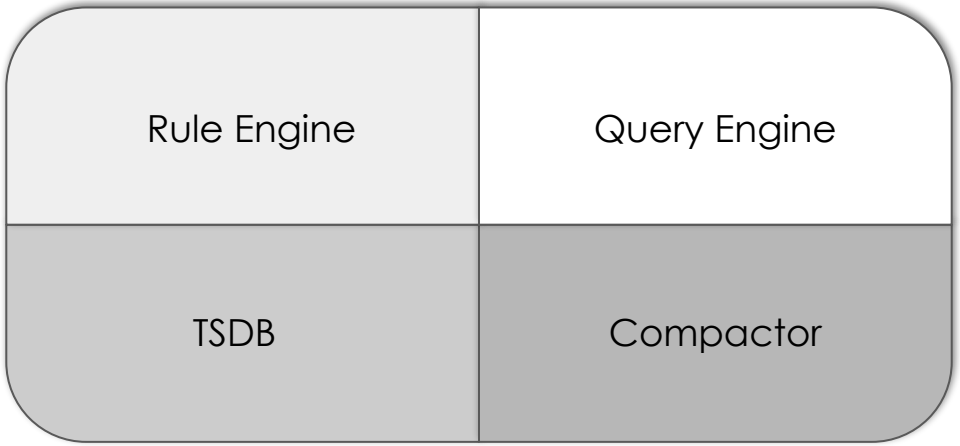
Prometheus

- Standalone monitoring server
- Deployed close to applications
- No external dependencies
- Stores data locally
- Flexible query language (PromQL)
- Yells at you when something is wrong



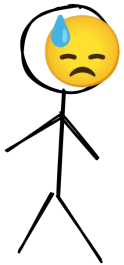
Prometheus

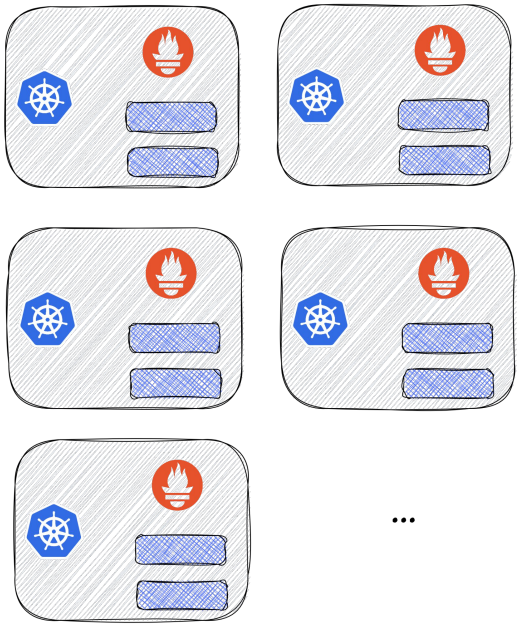
Prometheus decomposed



Limitations

- Lack of global overview of the data
- Hard to retain data for long periods of time
- Inadequate resolution for long range queries
- …and more





Thanos: “Distributed Prometheus ++”

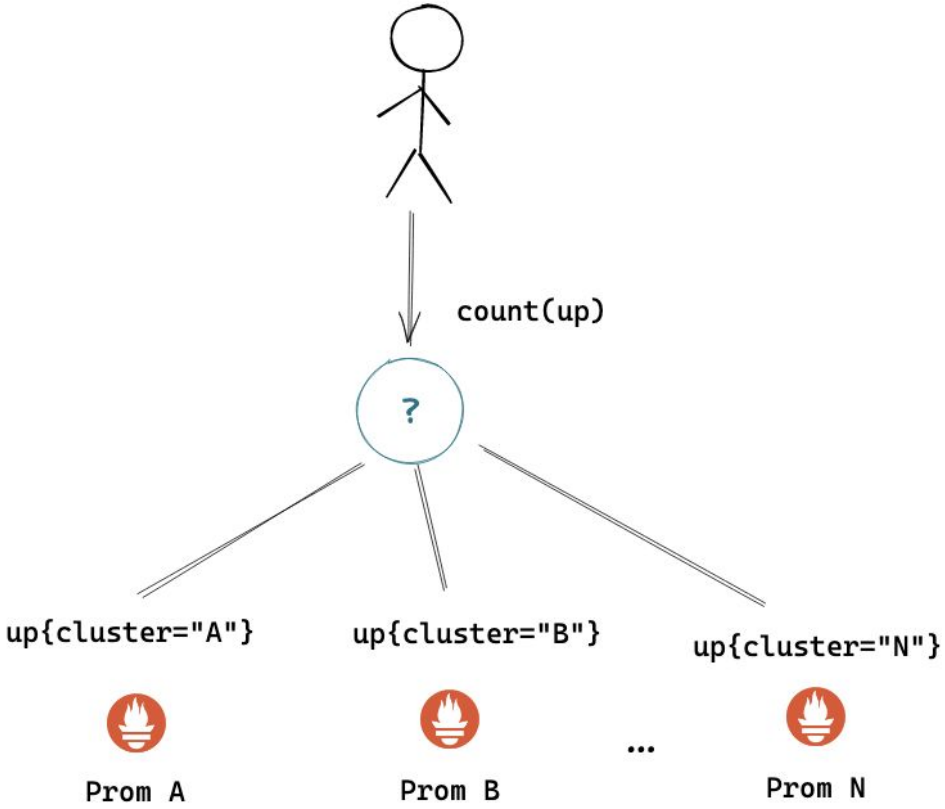
Thanos: Horizontally Scalable Metric Monitoring System

- Highly available and scalable components
- Global view
- Long term retention
- Downsampling
- Multi-Tenancy

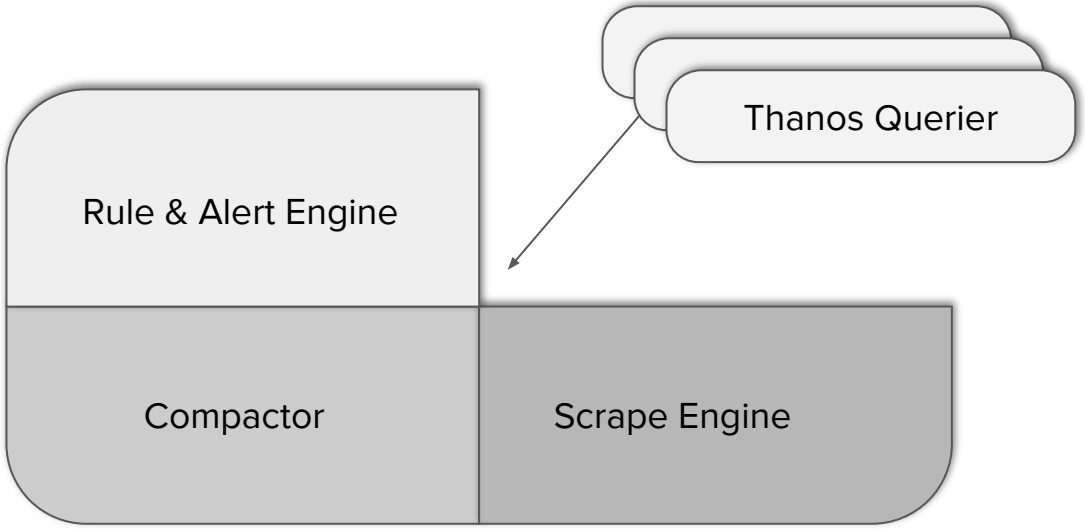
Challenge #1:

Global Query View

Global Query View



Thanos Query

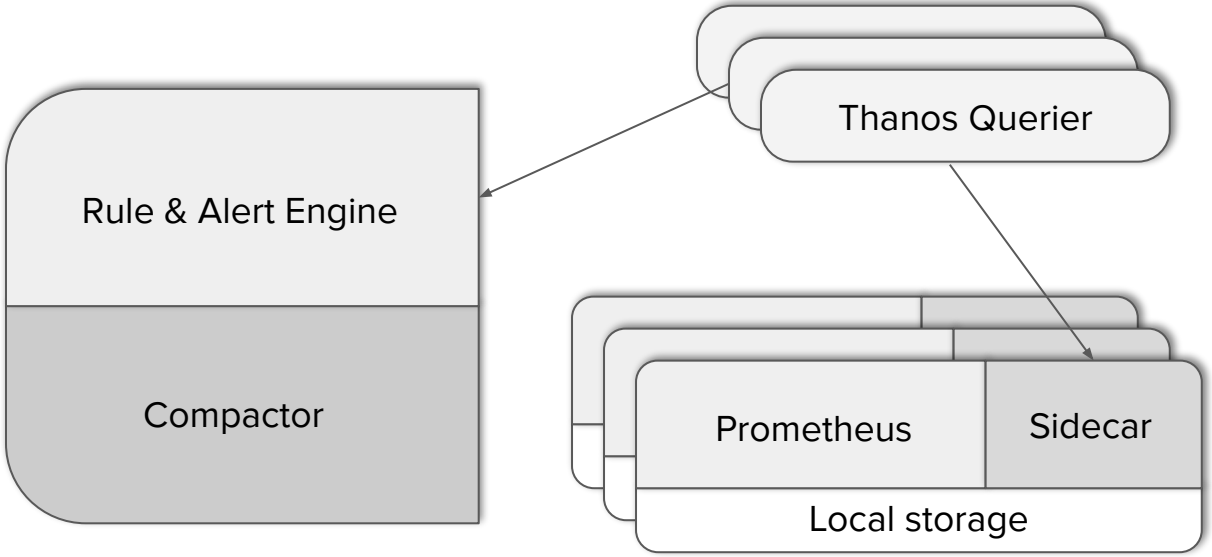


Thanos StoreAPI

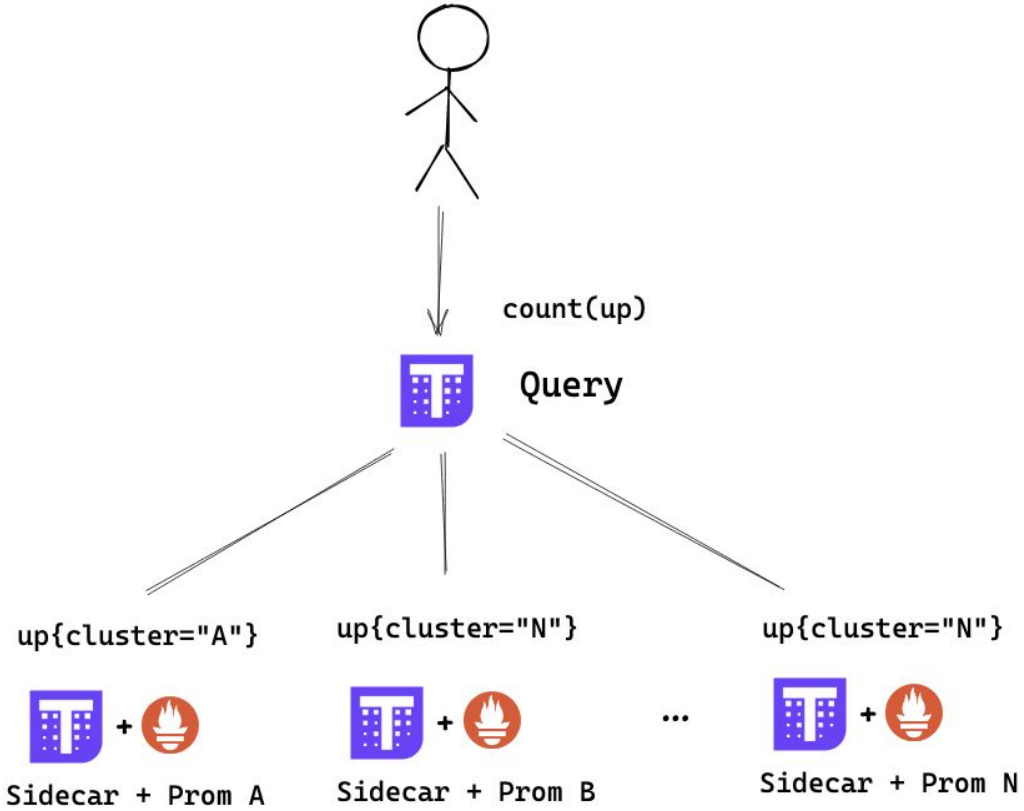
```
service Store {
  rpc Info(InfoRequest) returns (InfoResponse);
  rpc Series(SeriesRequest) returns (stream SeriesResponse);
  rpc LabelNames(LabelNamesRequest) returns (LabelNamesResponse);
  rpc LabelValues(LabelValuesRequest) returns (LabelValuesResponse);
}
```

From: rpc.proto

Thanos Sidecar



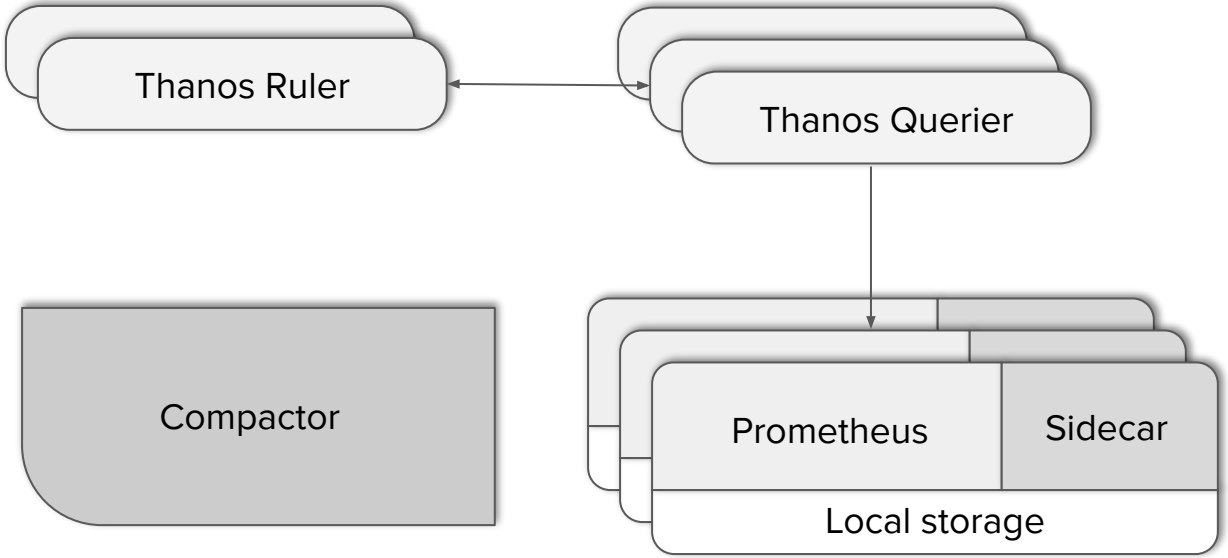
Global Query View



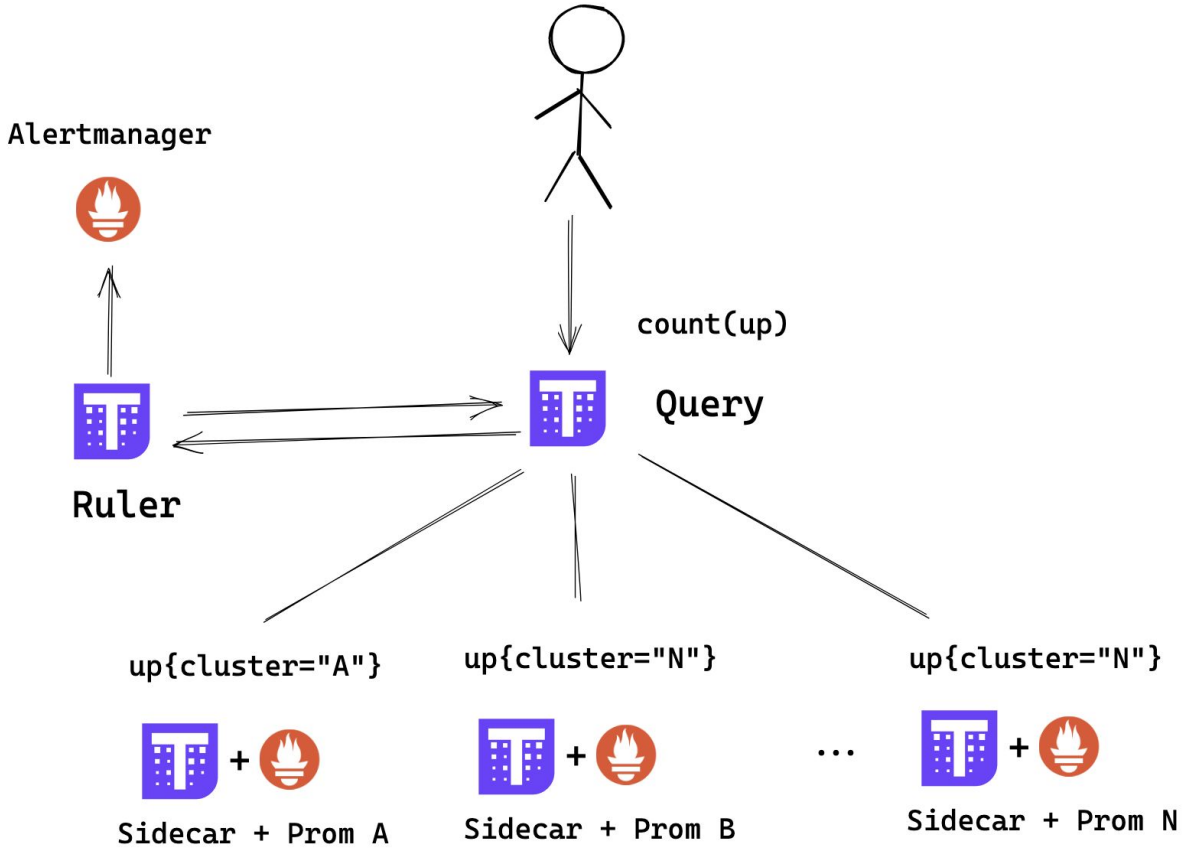
Challenge #2:

Global Rules & Alerting

Thanos Ruler



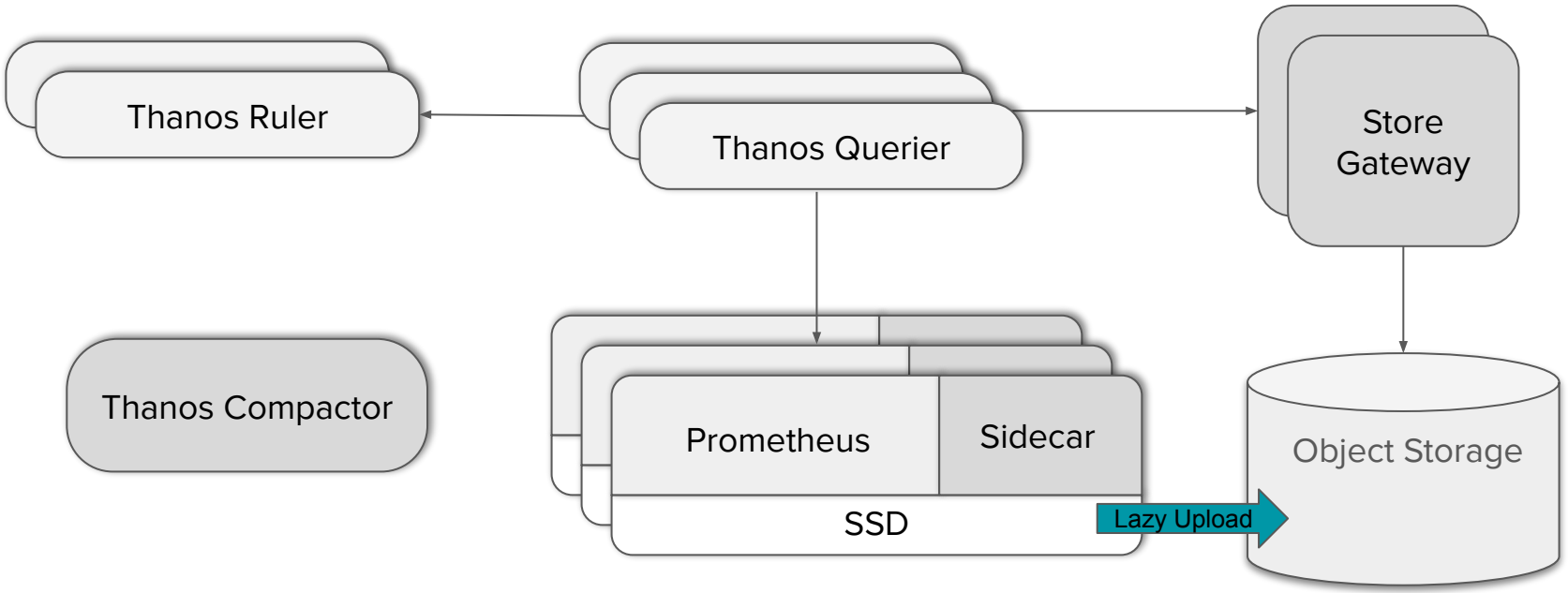
Global Rule Evaluation



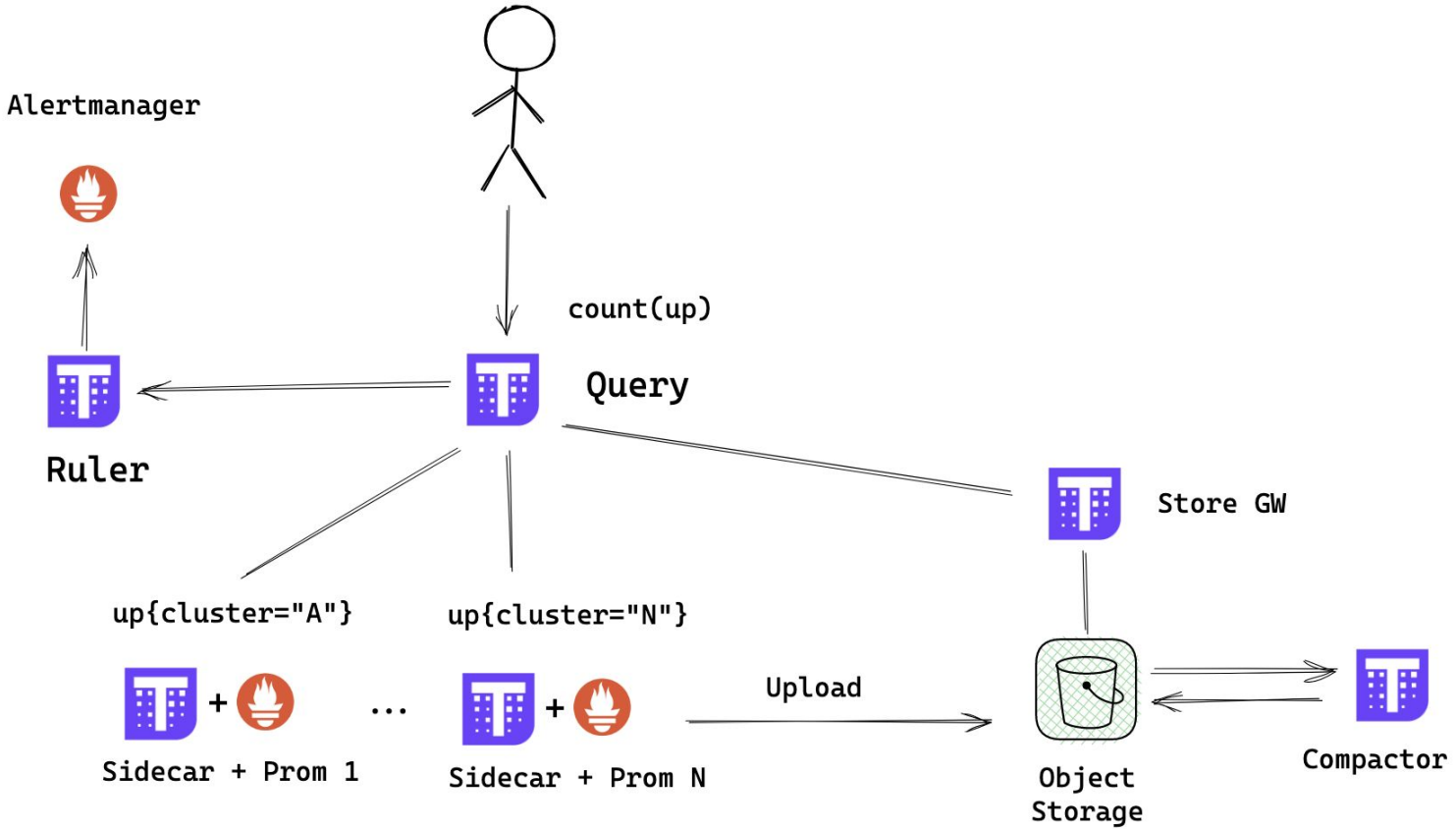
Challenge #3:

Long-term Retention

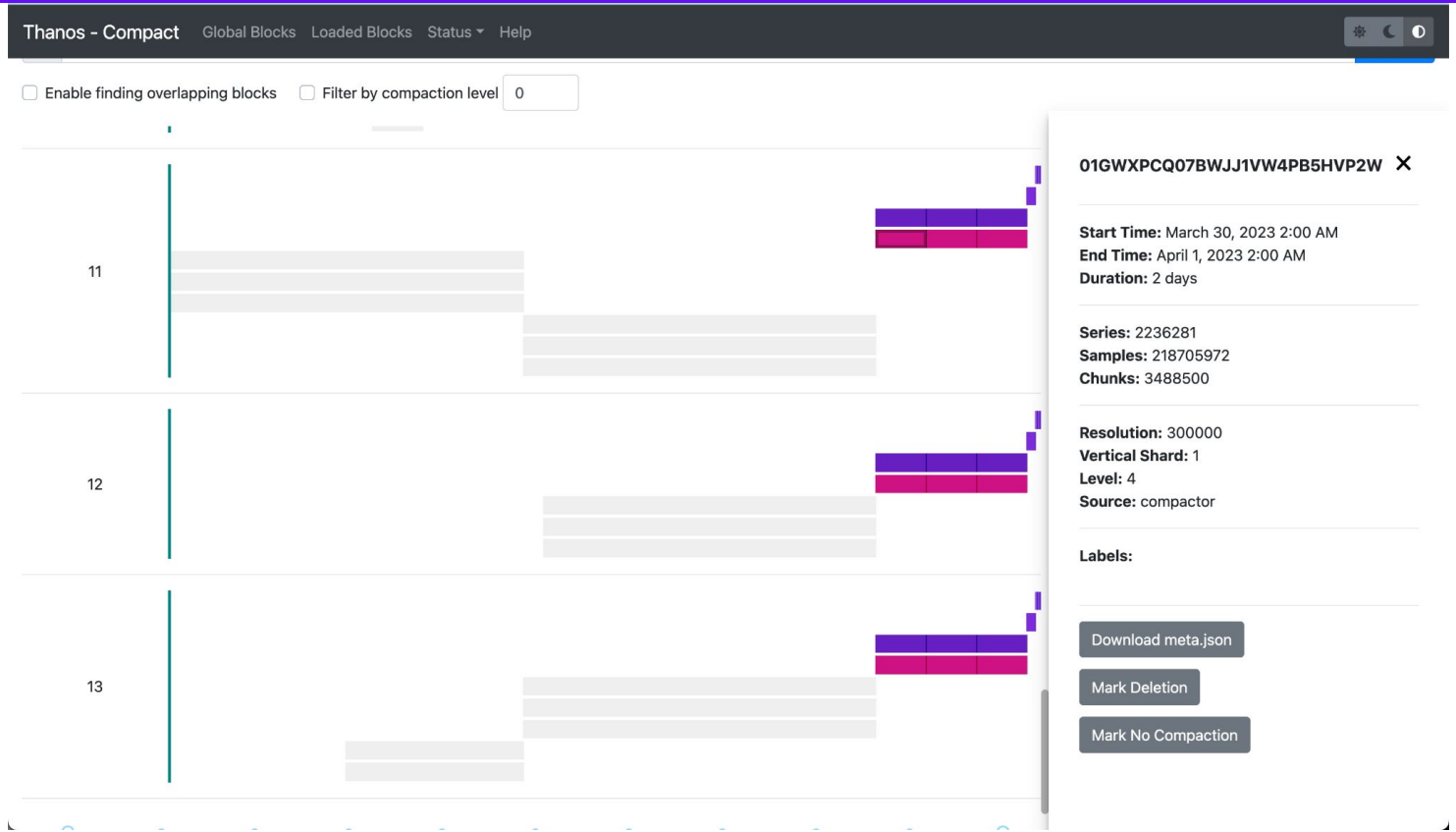
Thanos Store



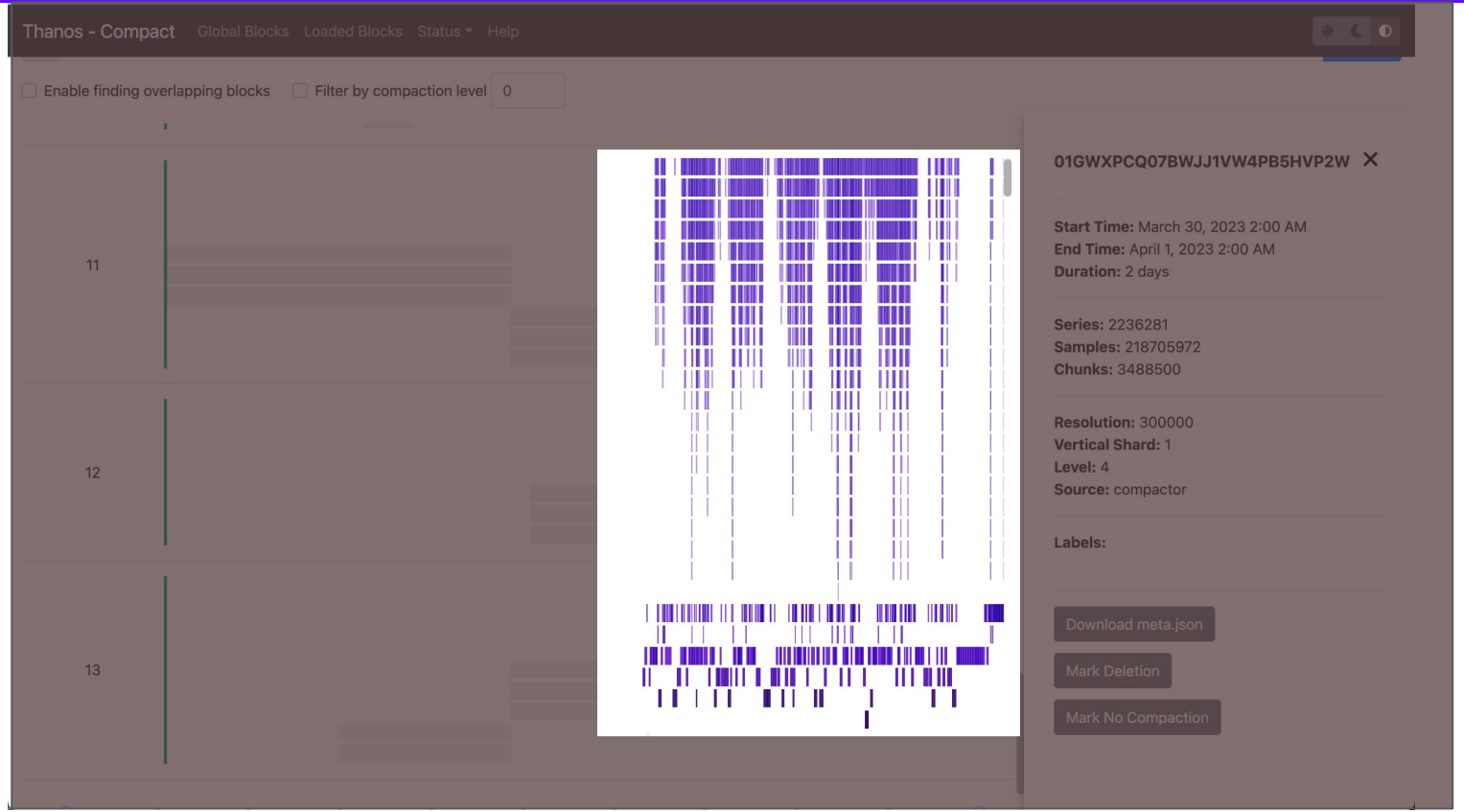
Query with Unlimited Retention



Compactor and Store UI



Compactor and Store UI



Challenge #4:

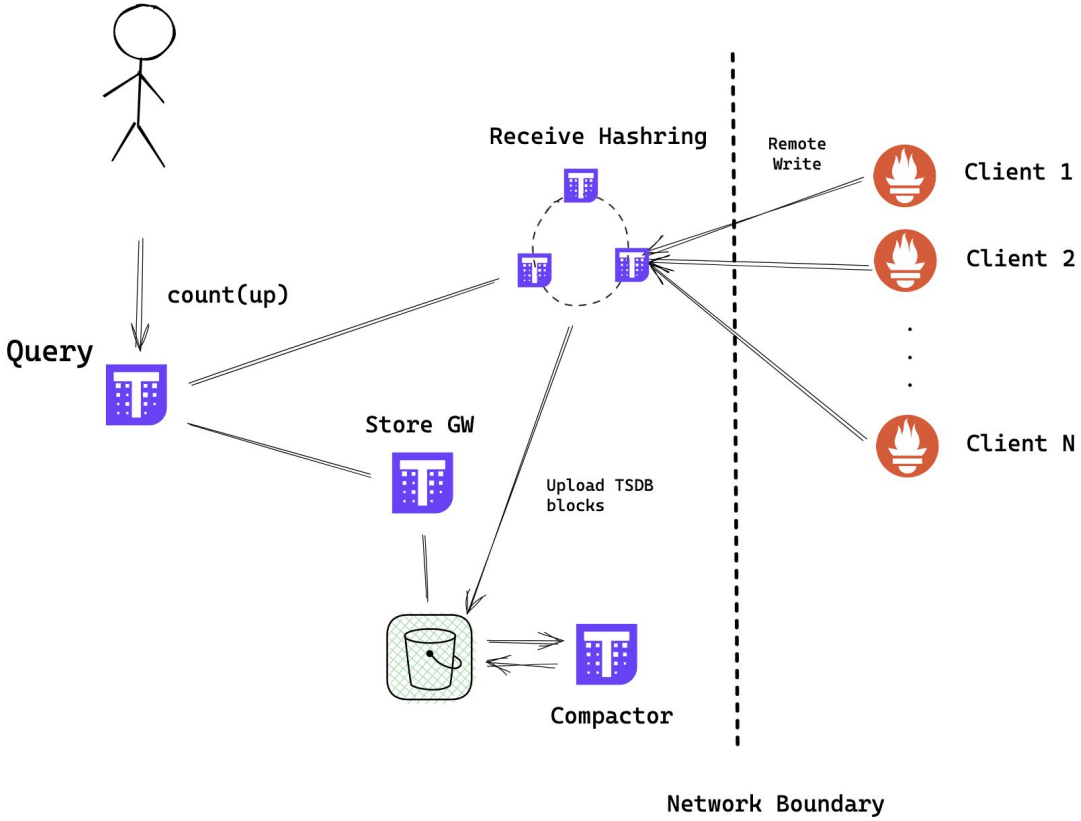
Egress-only Environments

Sidecar Architecture limitations!

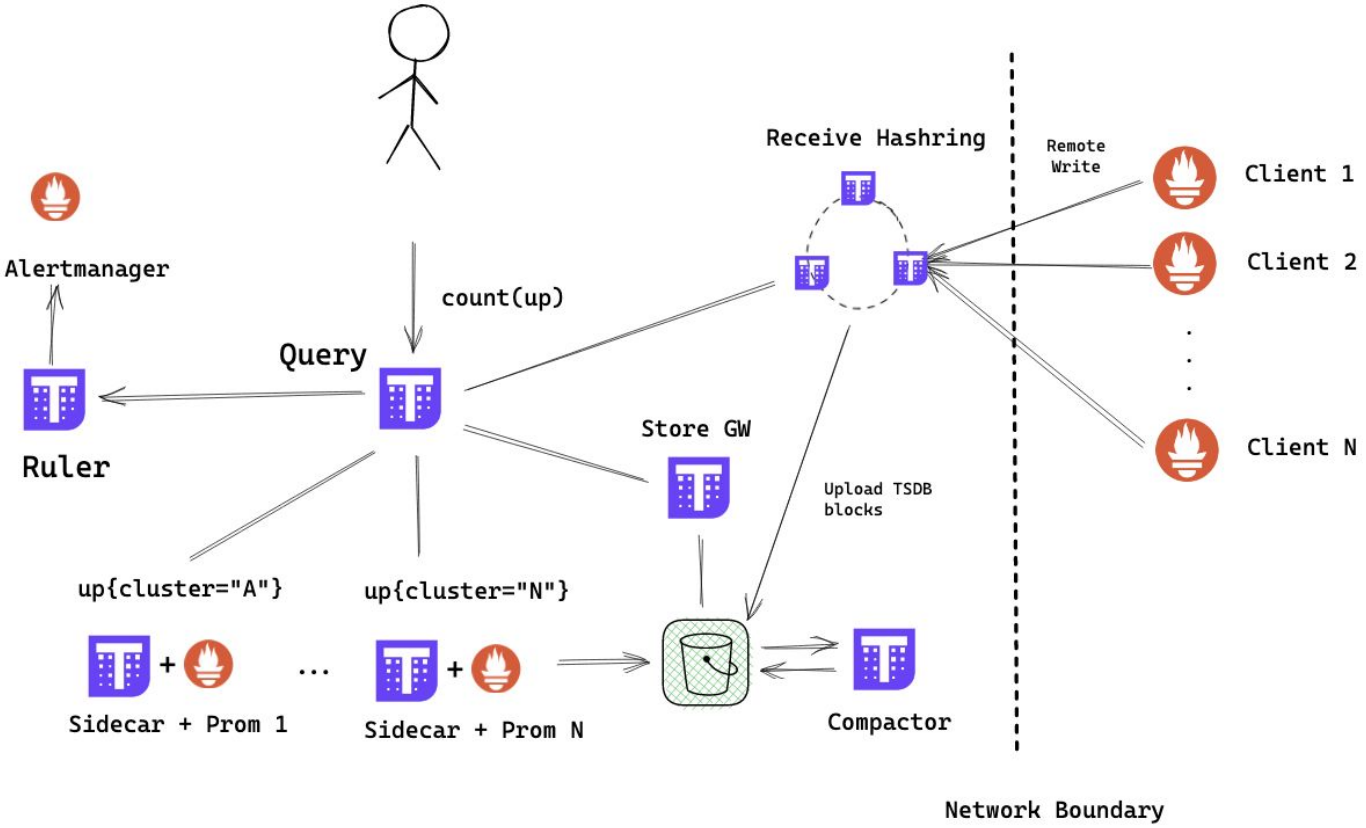
The Sidecar Architecture pattern has certain downsides:

- Needs an exposed ingress for the Querier connection
- Each sidecar needs Object Storage permissions
- Hard to design monitoring-as-a-service
- Not viable in resource constrained environments

Thanos receiver



Complex Thanos deployment

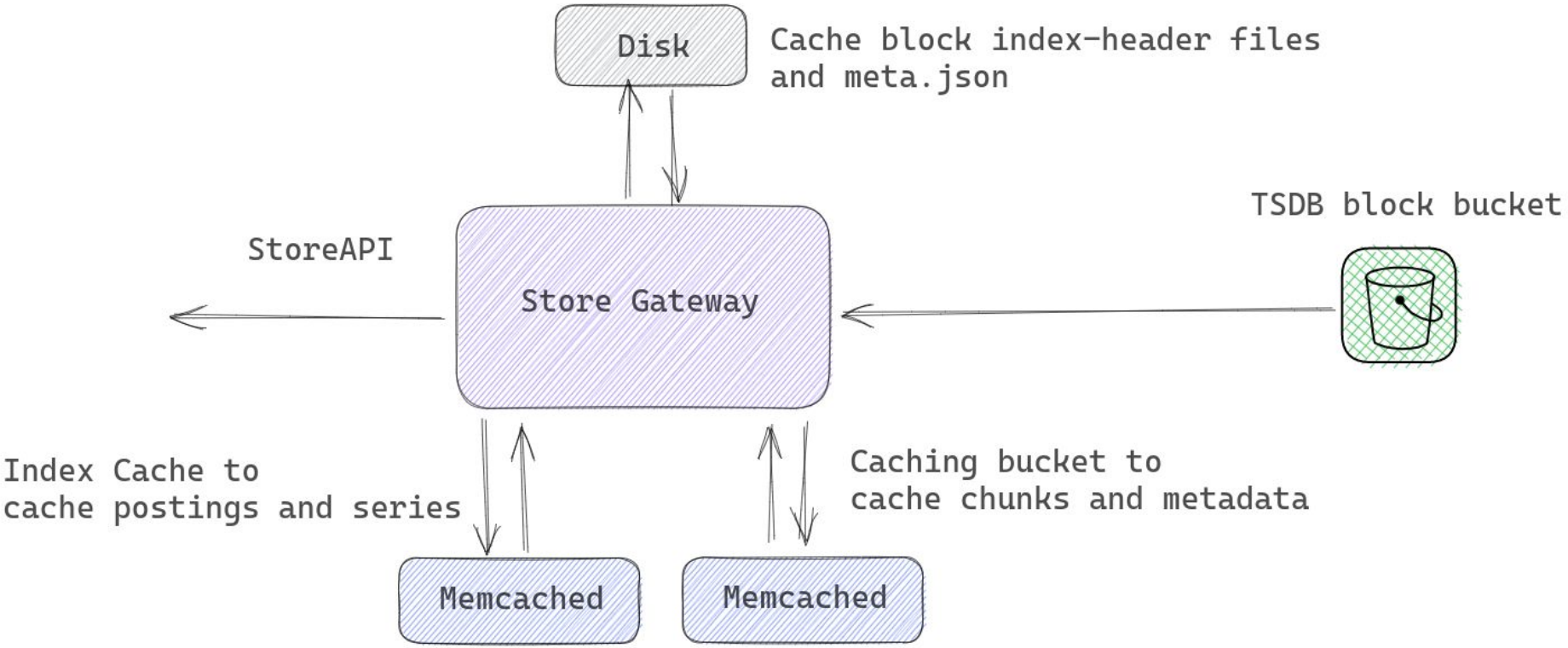


Improvements, Features & Optimizations

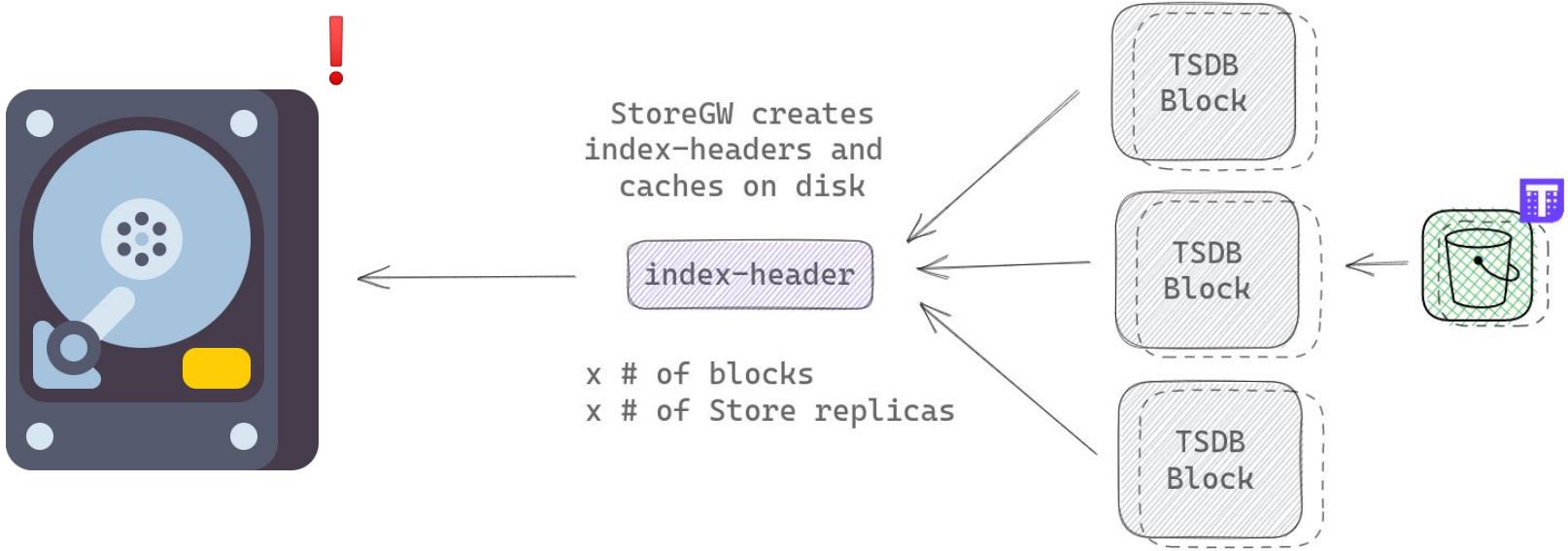
Feature Highlight #1:

Diskless Store GW!

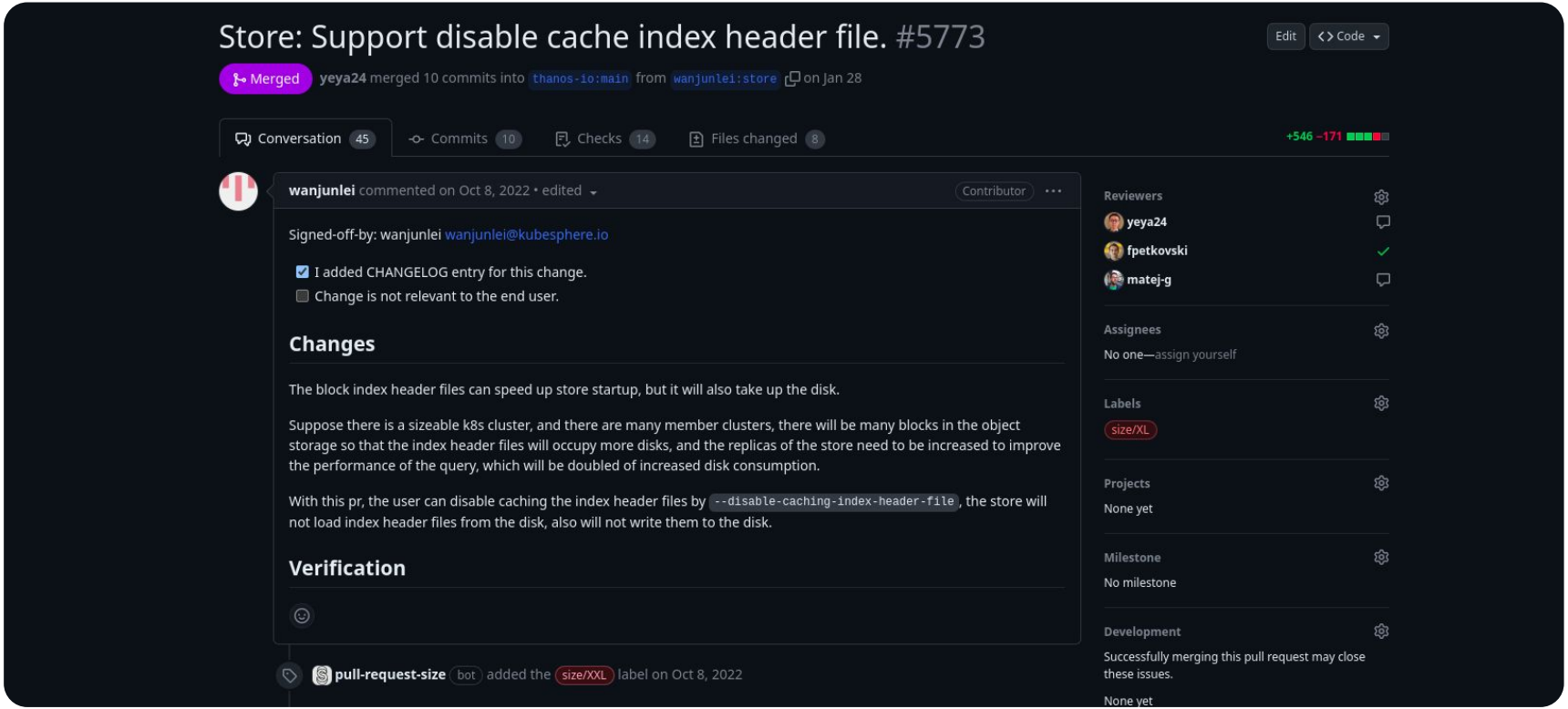
Store Gateway



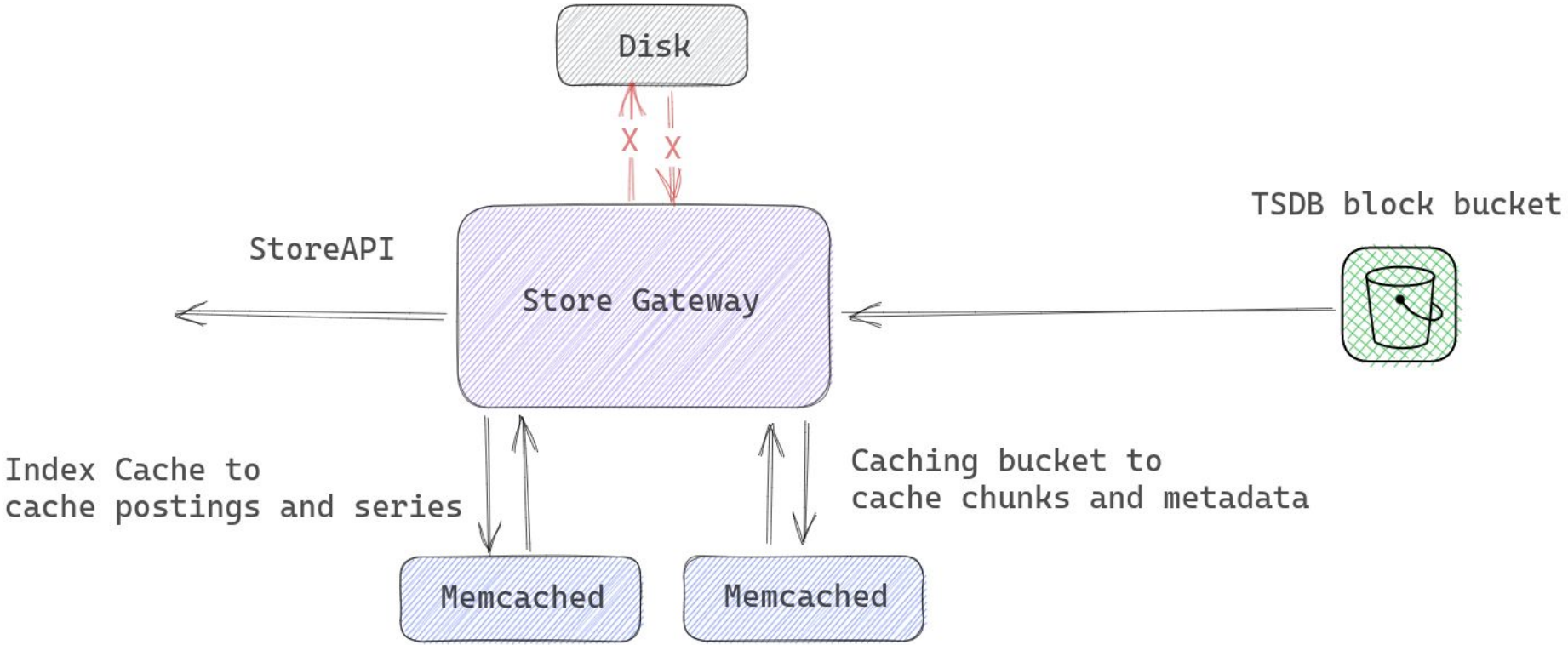
Too many index-headers!



Diskless Store Gateway



Diskless Store Gateway



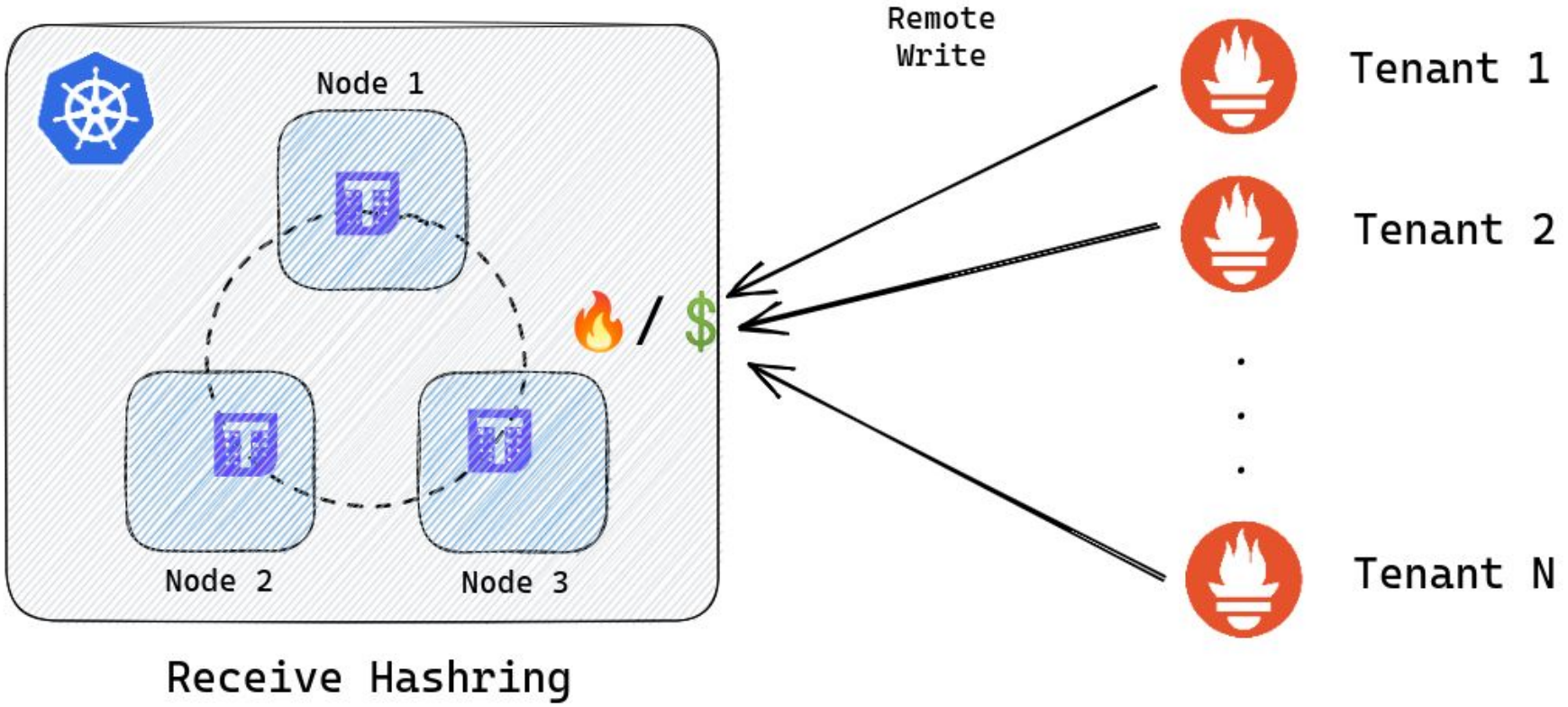
Feature Highlight #2:

QoS Improvements with Limits!

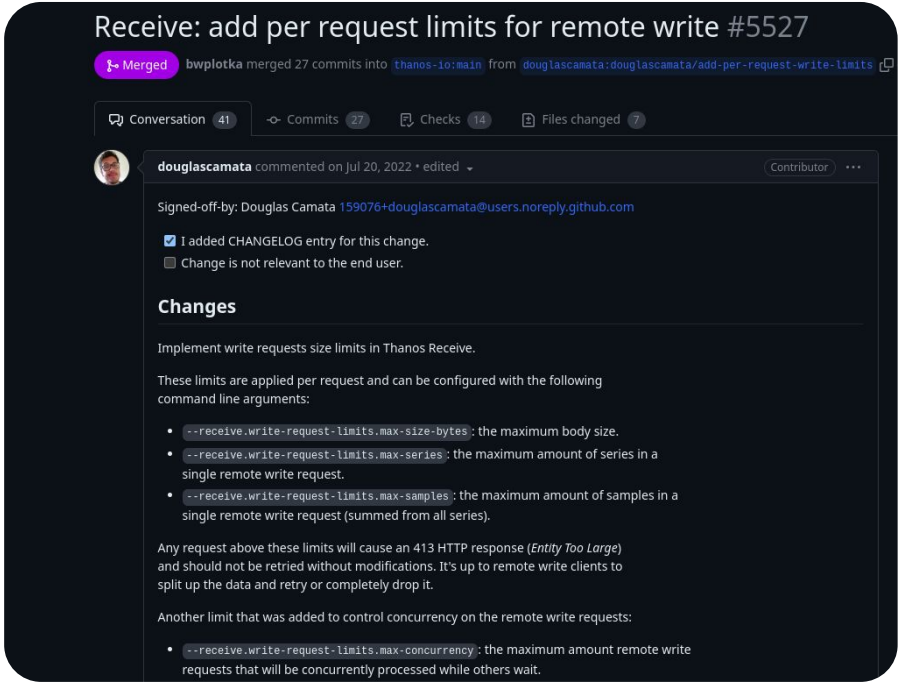
Quality Of Service

Quality of service is the ability to provide different priorities to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.

Write Path Scale



Write Path Limits



Receive: add per request limits for remote write #5527

Merged **bwplotka** merged 27 commits into `thanos-io:main` from `douglascamata:douglascamata/add-per-request-write-limits`

Conversation 41 · Commits 27 · Checks 14 · Files changed 7

douglascamata commented on Jul 20, 2022 • edited — Contributor

Signed-off-by: Douglas Camata 159076+douglascamata@users.noreply.github.com

- [x] I added CHANGELOG entry for this change.
- [ ] Change is not relevant to the end user.

Changes

Implement write requests size limits in Thanos Receive.

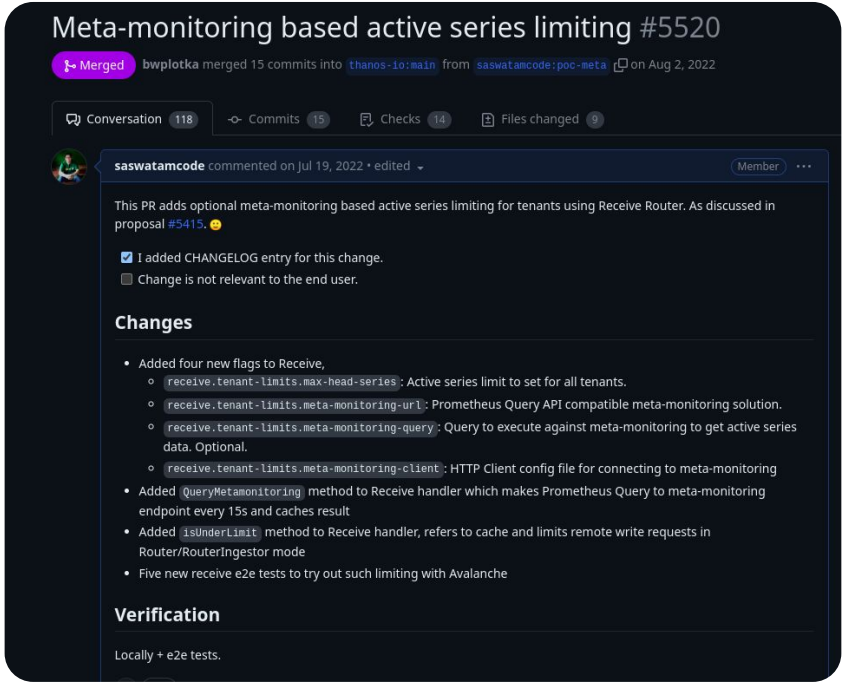
These limits are applied per request and can be configured with the following command line arguments:

- `--receive.write-request-limits.max-size-bytes`: the maximum body size.
- `--receive.write-request-limits.max-series`: the maximum amount of series in a single remote write request.
- `--receive.write-request-limits.max-samples`: the maximum amount of samples in a single remote write request (summed from all series).

Any request above these limits will cause an 413 HTTP response (*Entity Too Large*) and should not be retried without modifications. It's up to remote write clients to split up the data and retry or completely drop it.

Another limit that was added to control concurrency on the remote write requests:

- `--receive.write-request-limits.max-concurrency`: the maximum amount remote write requests that will be concurrently processed while others wait.



Meta-monitoring based active series limiting #5520

Merged **bwplotka** merged 15 commits into `thanos-io:main` from `saswatamcode:poc-meta` on Aug 2, 2022

Conversation 118 · Commits 15 · Checks 14 · Files changed 9

saswatamcode commented on Jul 19, 2022 • edited — Member

This PR adds optional meta-monitoring based active series limiting for tenants using Receive Router. As discussed in proposal #5415. 🙂

- [x] I added CHANGELOG entry for this change.
- [ ] Change is not relevant to the end user.

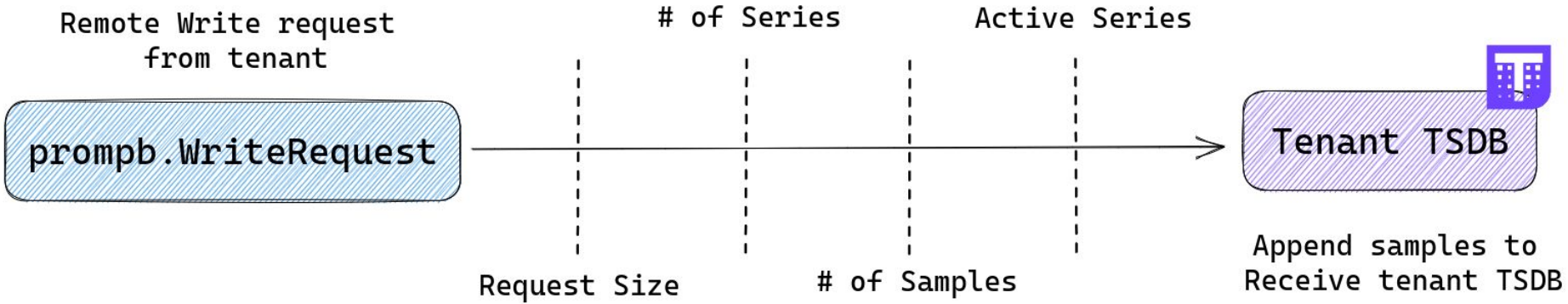
Changes

- Added four new flags to Receive,
  - `receive.tenant-limits.max-head-series`: Active series limit to set for all tenants.
  - `receive.tenant-limits.meta-monitoring-url`: Prometheus Query API compatible meta-monitoring solution.
  - `receive.tenant-limits.meta-monitoring-query`: Query to execute against meta-monitoring to get active series data. Optional.
  - `receive.tenant-limits.meta-monitoring-client`: HTTP Client config file for connecting to meta-monitoring
- Added `QueryMetamonitoring` method to Receive handler which makes Prometheus Query to meta-monitoring endpoint every 15s and caches result
- Added `isUnderLimit` method to Receive handler, refers to cache and limits remote write requests in Router/RouterIngestor mode
- Five new receive e2e tests to try out such limiting with Avalanche

Verification

Locally + e2e tests.

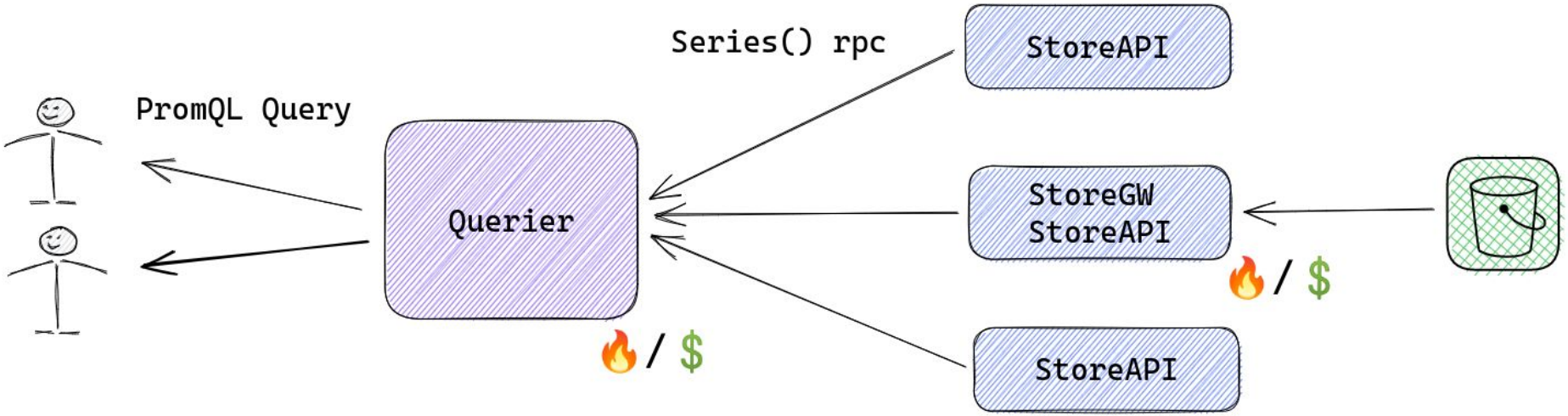
Write Path Limits



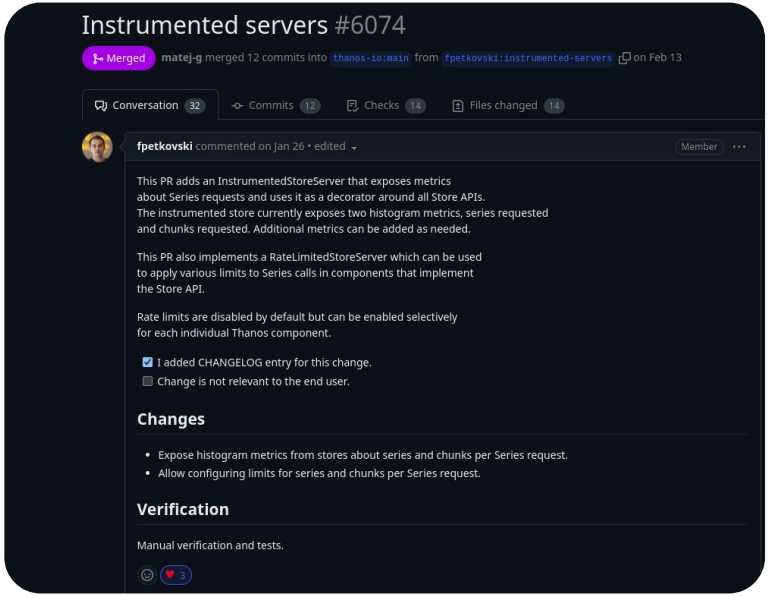
Write Path Limits

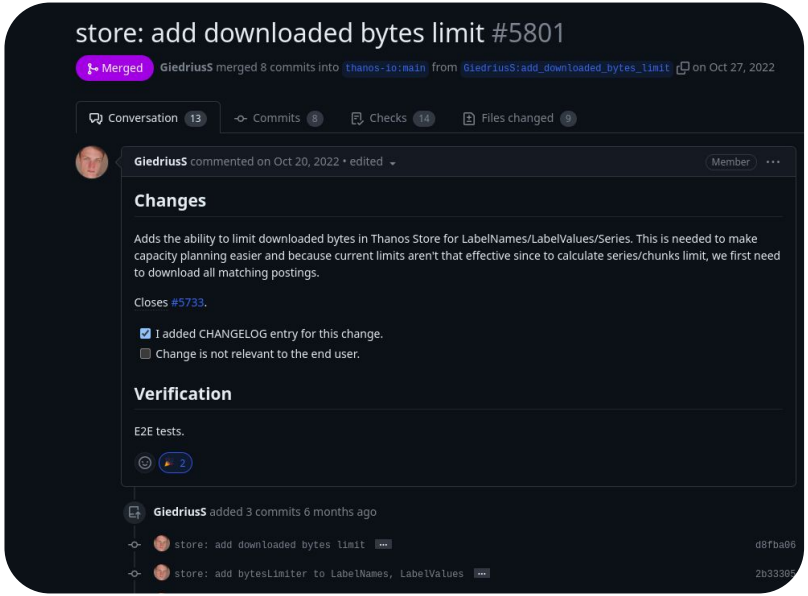
```yaml
write:
  global:
    max_concurrency: 30
    meta_monitoring_url: "http://localhost:9090"
    meta_monitoring_limit_query: "sum(prometheus_tsdb_head_series) by (tenant)"
  default:
    request:
      size_bytes_limit: 1024
      series_limit: 1000
      samples_limit: 10
    head_series_limit: 1000
  tenants:
    acme:
      request:
        size_bytes_limit: 0
        series_limit: 0
        samples_limit: 0
      head_series_limit: 2000
    ajax:
      request:
        series_limit: 50000
        samples_limit: 500
```


Query Path Scale

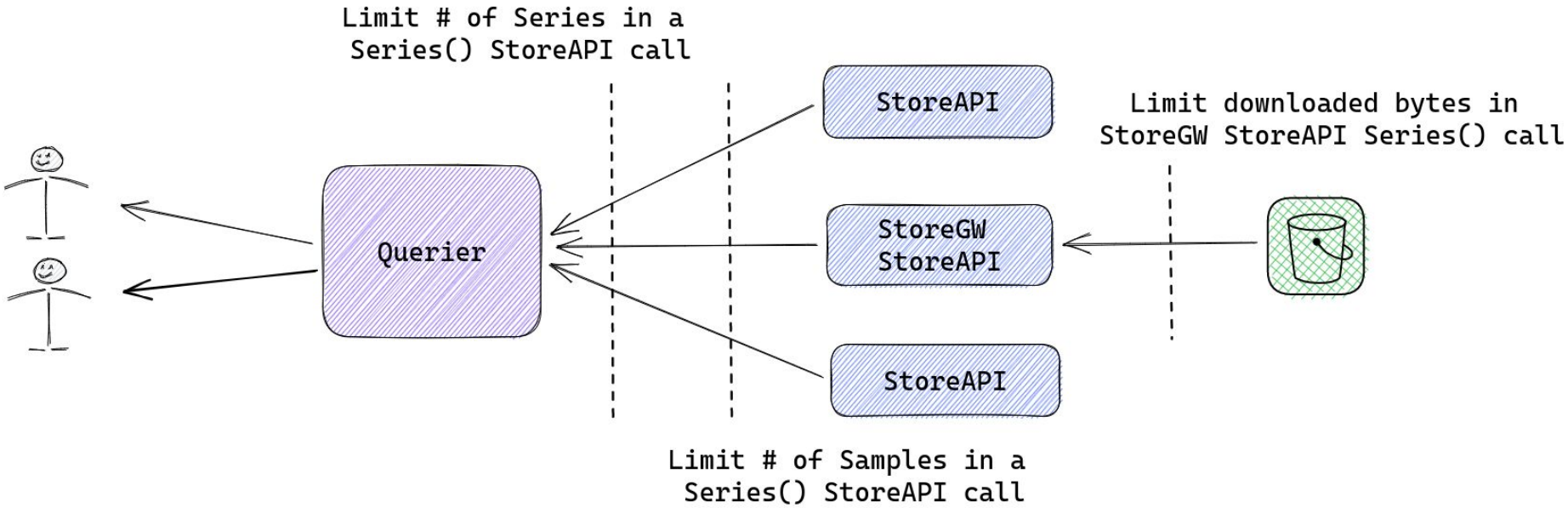


Query Path Limits





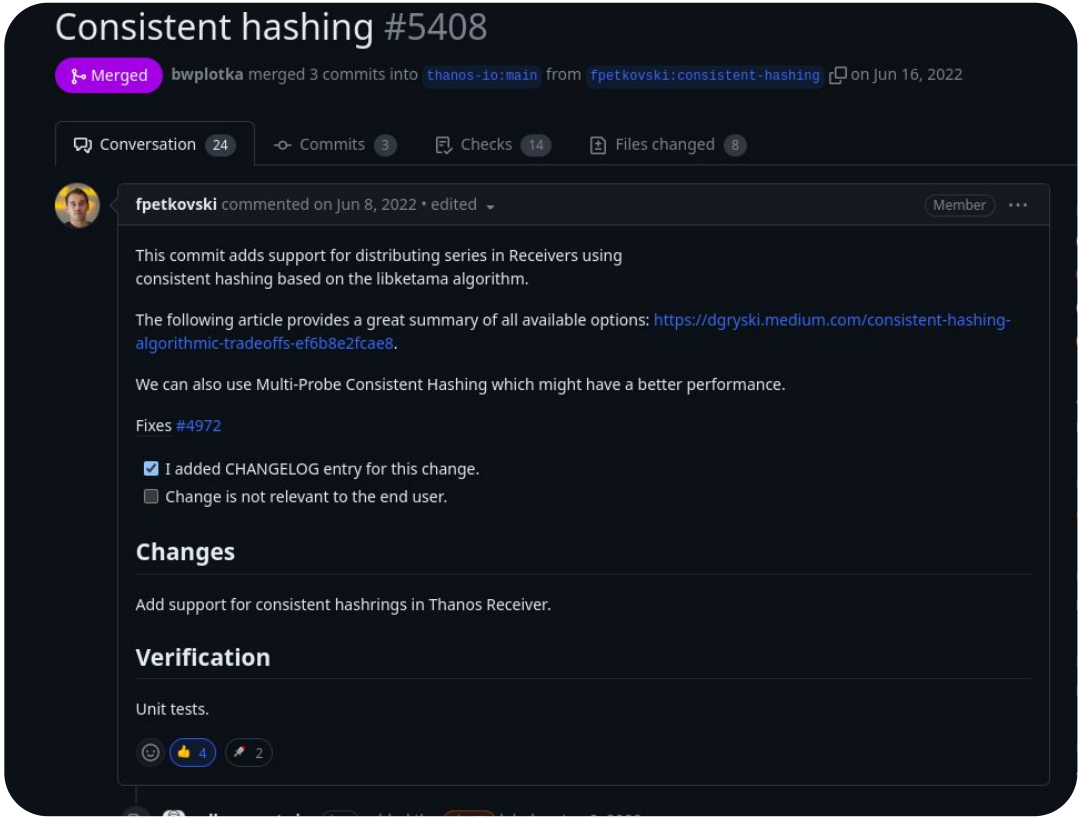
Query Path Limits



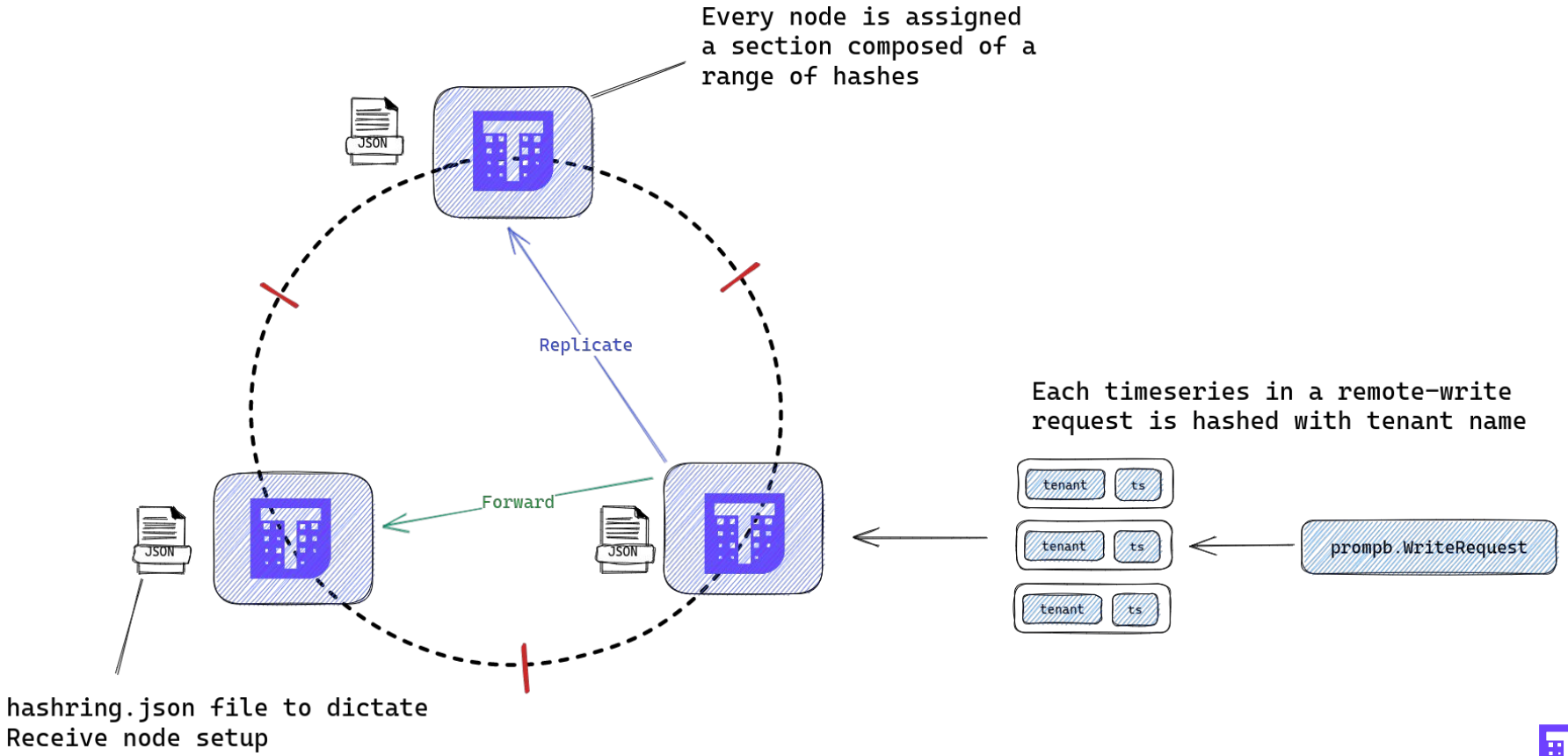
Feature Highlight #3:

Consistent Hashing in Receive

Ketama Hashring

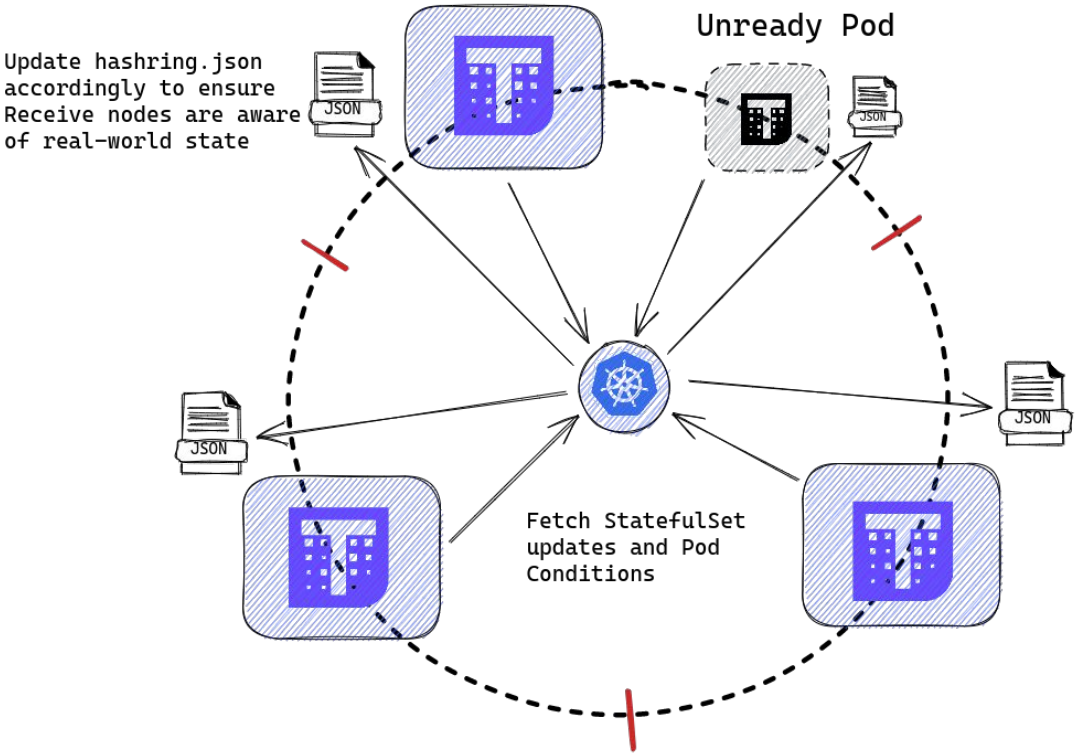


Ketama Hashring



Thanos Receive Controller

https://github.com/observatorium/thanos-receive-controller/

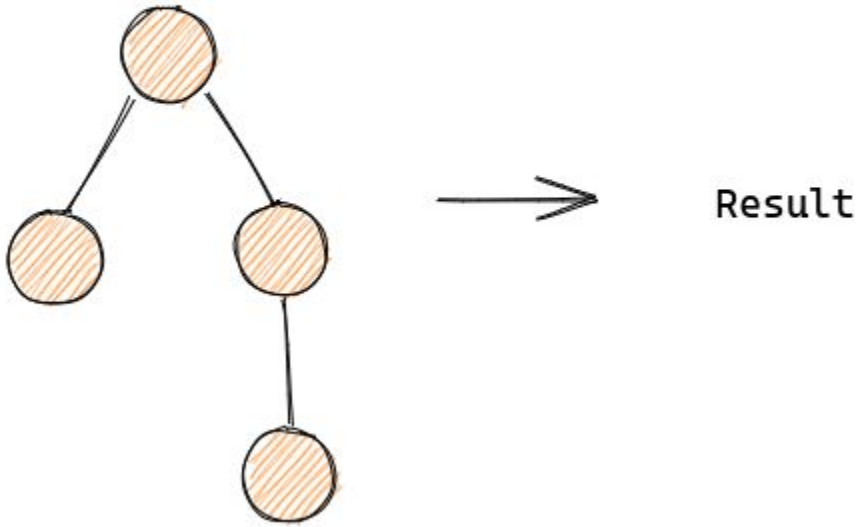


Feature Highlight #4:

New PromQL Engine!

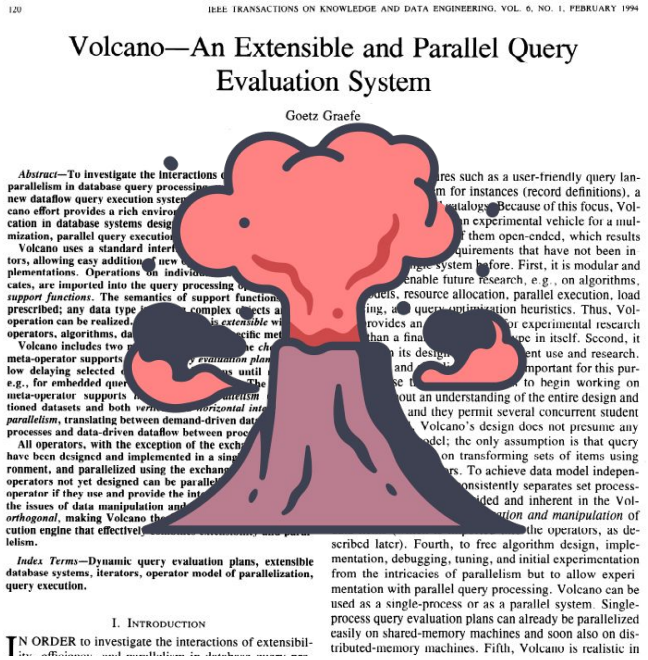
Prometheus PromQL Engine

Query Parsed into AST
Evaluate as we parse
and return value, all in a single thread



Thanos PromQL Engine: Design

https://paperhub.s3.amazonaws.com/dace52a42c07f7f8348b08dc2b186061.pdf



120 IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 6, NO. 1, FEBRUARY 1994

Volcano—An Extensible and Parallel Query Evaluation System

Goetz Graefe

Abstract—To investigate the interactions ... parallelism in database query processing ... new dataflow query execution system ... cano effort provides a rich environ... cation in database systems design ... mization, parallel query execution ...

Volcano uses a standard interf... tors, allowing easy addition of new ... plementations. Operations on individ... cates, are imported into the query processing ... *support functions*. The semantics of support function... prescribed; any data type ... complex objects a... operation can be realized. ... is *extensible* wi... operators, algorithms, da... specific me...

Volcano includes two ... *che*... meta-operator supports ... *of evaluation plan*... low delaying selected ... ...s until ... e.g., for embedded quer... The ... meta-operator supports ... ...*allelism* ... tioned datasets and both *vertical* ... *horizontal int*... *parallelism*, translating between demand-driven dat... processes and data-driven dataflow between proc...

All operators, with the exception of the exch... have been designed and implemented in a sin... ronment, and parallelized using the exchan... operators not yet designed can be parall... operator if they use and provide the inte... the issues of data manipulation and ... *orthogonal*, making Volcano the ... cution engine that effectivel... ... ...lelism.

Index Terms—Dynamic query evaluation plans, extensible database systems, iterators, operator model of parallelization, query execution.

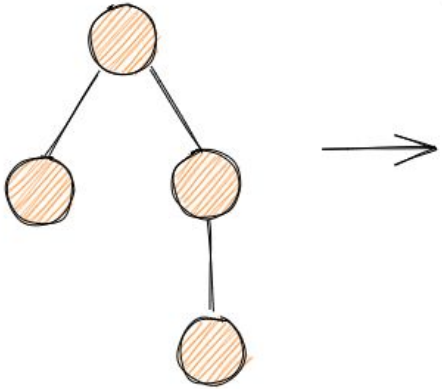
I. Introduction

IN ORDER to investigate the interactions of extensibil-...

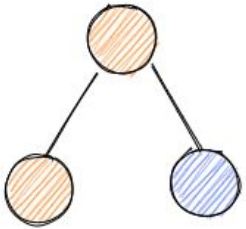
...tures such as a user-friendly query lan-...m for instances (record definitions), a ...atalog. Because of this focus, Vol-...n experimental vehicle for a mul-...f them open-ended, which results ...quirements that have not been in ...gle system before. First, it is modular and ...enable future research, e.g., on algorithms, ...dels, resource allocation, parallel execution, load ...ing, a... query optimization heuristics. Thus, Vol-...rovides an ... ...or experimental research ...than a fina... ...pe in itself. Second, it ...n its design ... ...ent use and research. ...and ... ...important for this pur-...se t... ...to begin working on ...out an understanding of the entire design and ... and they permit several concurrent student ... Volcano's design docs not presume any ...odel; the only assumption is that query ...on transforming sets of items using ...rs. To achieve data model indepen-...onsistently separates set process-...ded and inherent in the Vol-...*tion and manipulation* of ... the operators, as de-scribed later). Fourth, to free algorithm design, imple-mentation, debugging, tuning, and initial experimentation from the intricacies of parallelism but to allow experi mentation with parallel query processing. Volcano can be used as a single-process or as a parallel system. Single-process query evaluation plans can already be parallelized easily on shared-memory machines and soon also on dis-tributed-memory machines. Fifth, Volcano is realistic in

Thanos PromQL Engine: How it works?

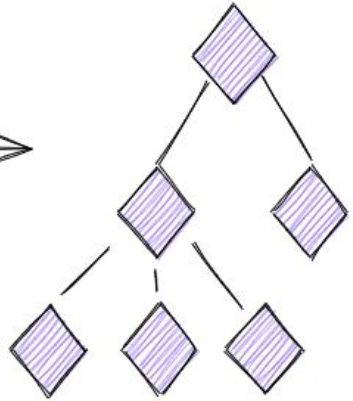
Query Parsed into AST



Logical Plan optimizes query tree
(if possible)



Physical plan converts
Query tree to Operator tree
and executes Query using multiple cores

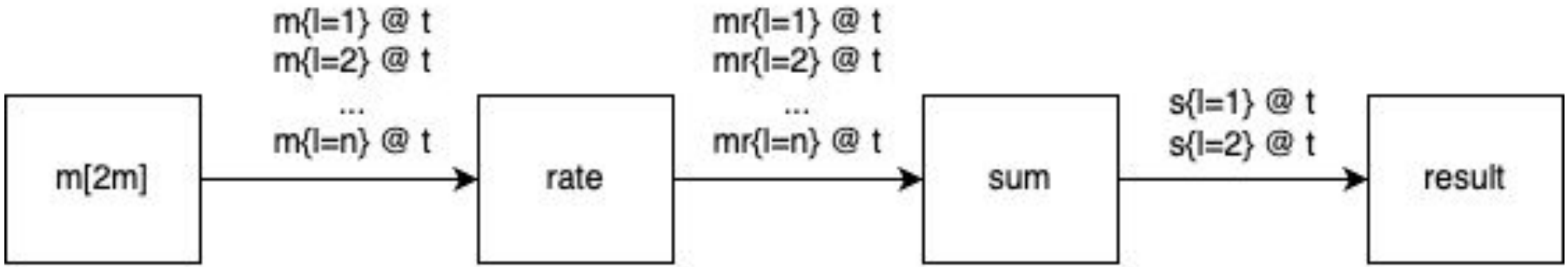


Result

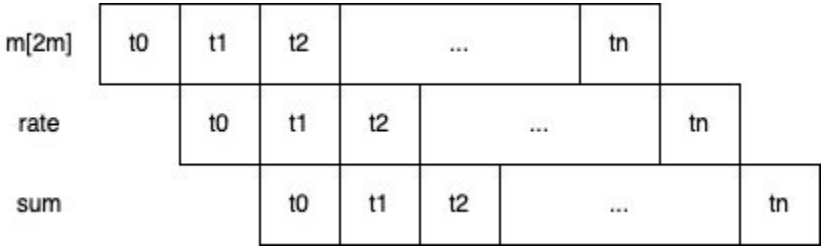
Thanos PromQL Engine: Operators

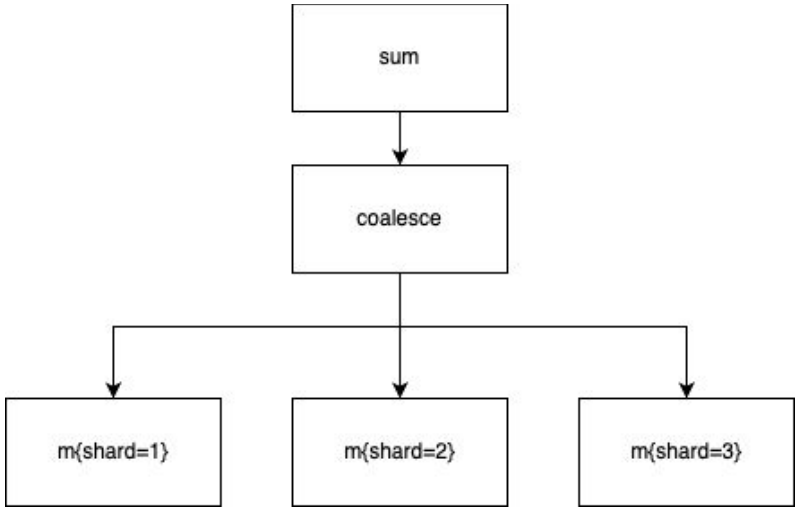
```go
// VectorOperator performs operations on series in step by step fashion.
type VectorOperator interface {
    // Next yields vectors of samples from all series for one or more execution steps.
    Next(ctx context.Context) ([]StepVector, error)

    // Series returns all series that the operator will process during Next results.
    // The result can be used by upstream operators to allocate output tables and buffers
    // before starting to process samples.
    Series(ctx context.Context) ([]labels.Labels, error)

    // GetPool returns pool of vectors that can be shared across operators.
    GetPool() *VectorPool

    // Explain returns human-readable explanation of the current operator and optional nested operators.
    Explain() (me string, next []VectorOperator)
}
```


Thanos PromQL Engine: Operator flow



Thanos PromQL Engine: Parallelism

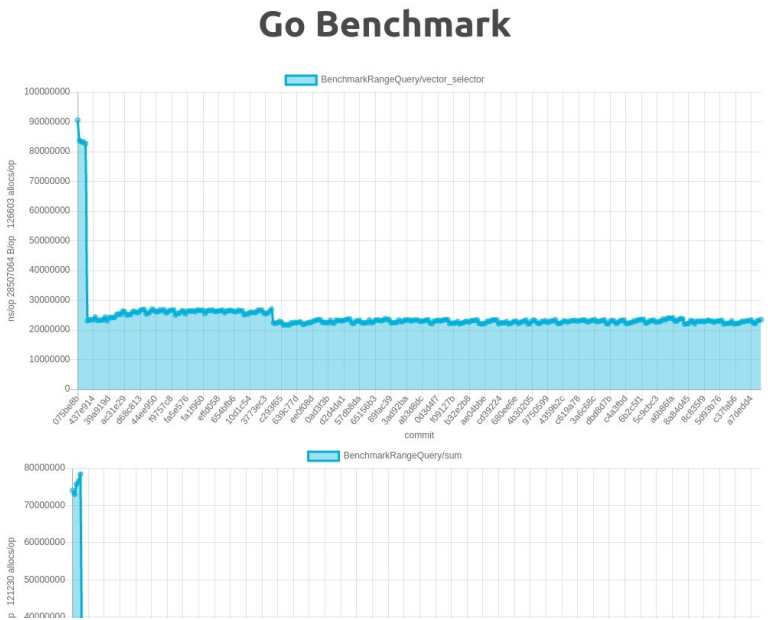




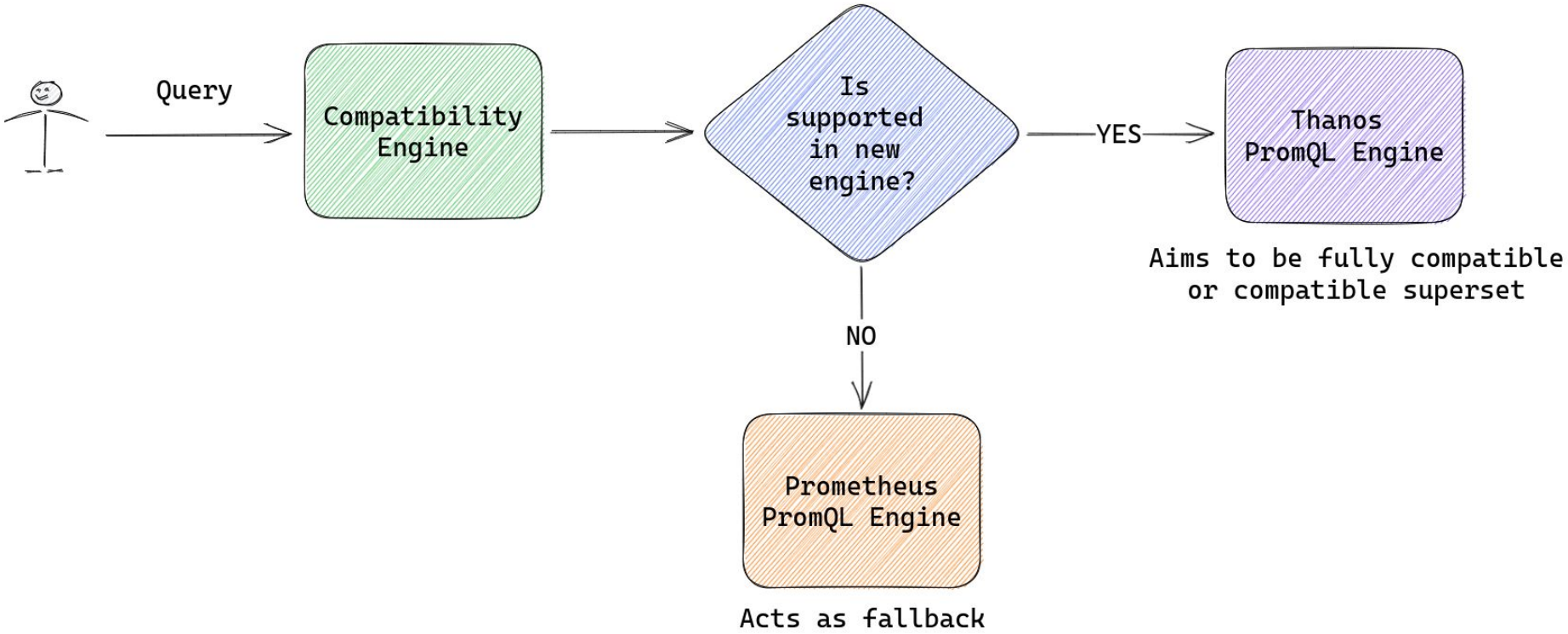
Thanos PromQL Engine: Benchmarks

https://thanos-community.github.io/promql-engine/dev/bench/

Repository: https://github.com/thanos-community/promql-engine



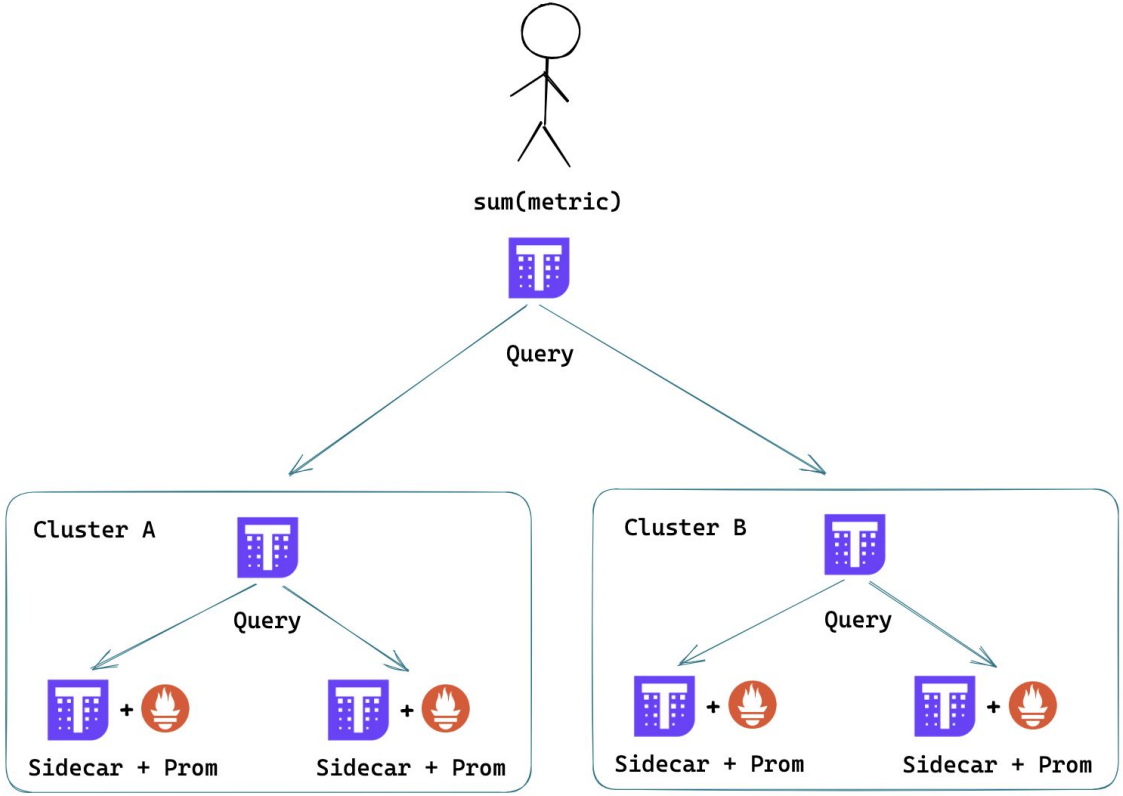
Thanos PromQL Engine: Compatibility



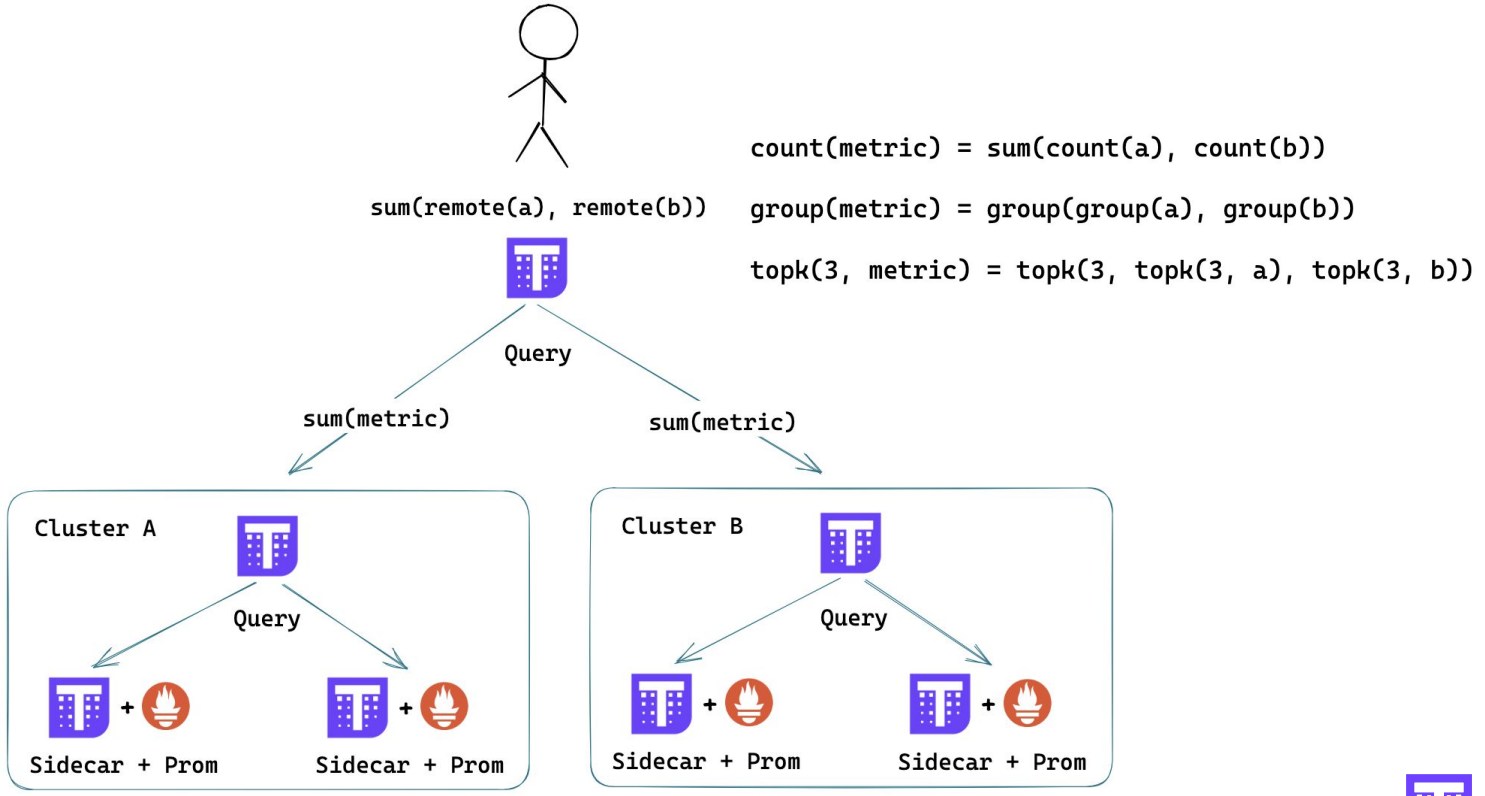
Feature Highlight #5:

Distributed Execution!

Distributed execution



Distributed execution




```
count(metric) = sum(count(a), count(b))
group(metric) = group(group(a), group(b))
topk(3, metric) = topk(3, topk(3, a), topk(3, b))
```

Native histograms

Add Native Histogram Support #5907

Open · 3 of 7 tasks · **rabenhorst** opened this issue on Nov 18, 2022 · 4 comments

rabenhorst commented on Nov 18, 2022 • edited — Contributor

Is your proposal related to a problem?

Experimental native histogram support was released with Prometheus v0.40.x. #5896 updated Thanos to Prometheus v0.40.1 but without implementing native histograms.

I propose the implementation of native histogram support (behind a feature flag) as a next step in follow up PR(s).

Describe the solution you'd like

We need to go through all Thanos components and check what needs to be changed for native histogram support. So far we identified the following changes are required (*list will be updated on new findings*):

- [x] Sidecar | **sidecar, query, receiver: Native histogram support** #6032
  - Add native histograms to storepb and prompb types
- [ ] Query | **sidecar, query, receiver: Native histogram support** #6032
  - Add native histogram support for dedup and query iterators
  - Update UI for properly displaying native histograms
- [ ] Query Frontend | **query frontend, query UI: Native histogram support** #6071
  - Update types
- [x] Receive | **sidecar, query, receiver: Native histogram support** #6032
  - Update writer to append histograms
- [ ] Compactor | w.i.p.
  - Implement native histogram downsampling
- [x] Store | **Support native histograms in store-gateway** #6056

Detailed investigation still needs to be done for the following components.

- [ ] Rule

For each component we need to make sure to have tests in place after implementation and e2e test for functionality provided by multiple components (e.g. ingestion, querying, etc.).

sidecar, query, receiver: Native histogram support #6032

Merged · **GiedriusS** merged 8 commits into `thanos-io:main` from `rabenhorst:native-histograms` on Jan 24

query frontend, query UI: Native histogram support #6071

Open · **rabenhorst** wants to merge 2 commits into `thanos-io:main` from `rabenhorst:native-histograms-qfe`

Conversation 21 · Commits 2 · Checks 14 · Files changed 23

rabenhorst commented on Jan 24 • edited — Contributor

As followup to #6032 , this PR adds native histogram support for query frontend and query UI. Additionally this PR adds native histogram support in the `ApplyCounterResetsSeriesIterator` , which is required for histogram rates.

- [ ] I added CHANGELOG entry for this change.
- [ ] Change is not relevant to the end user.

Changes

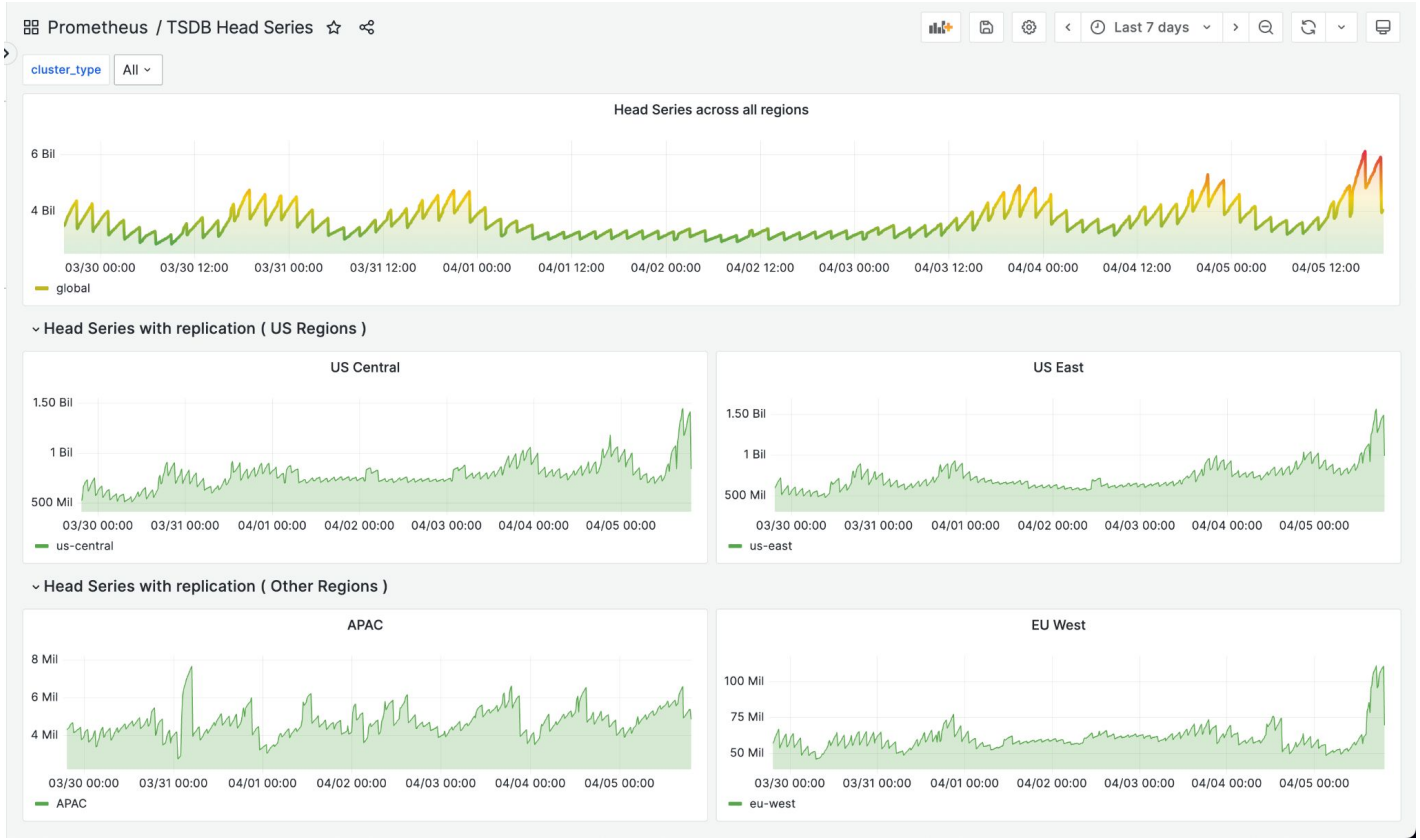
- Native histogram support for query frontend
- Native histogram support for query UI (table and graph)
- Native histogram support in `ApplyCounterResetsSeriesIterator`
- Enable native histograms rates.

Verification

Unit tests and e2e tests.

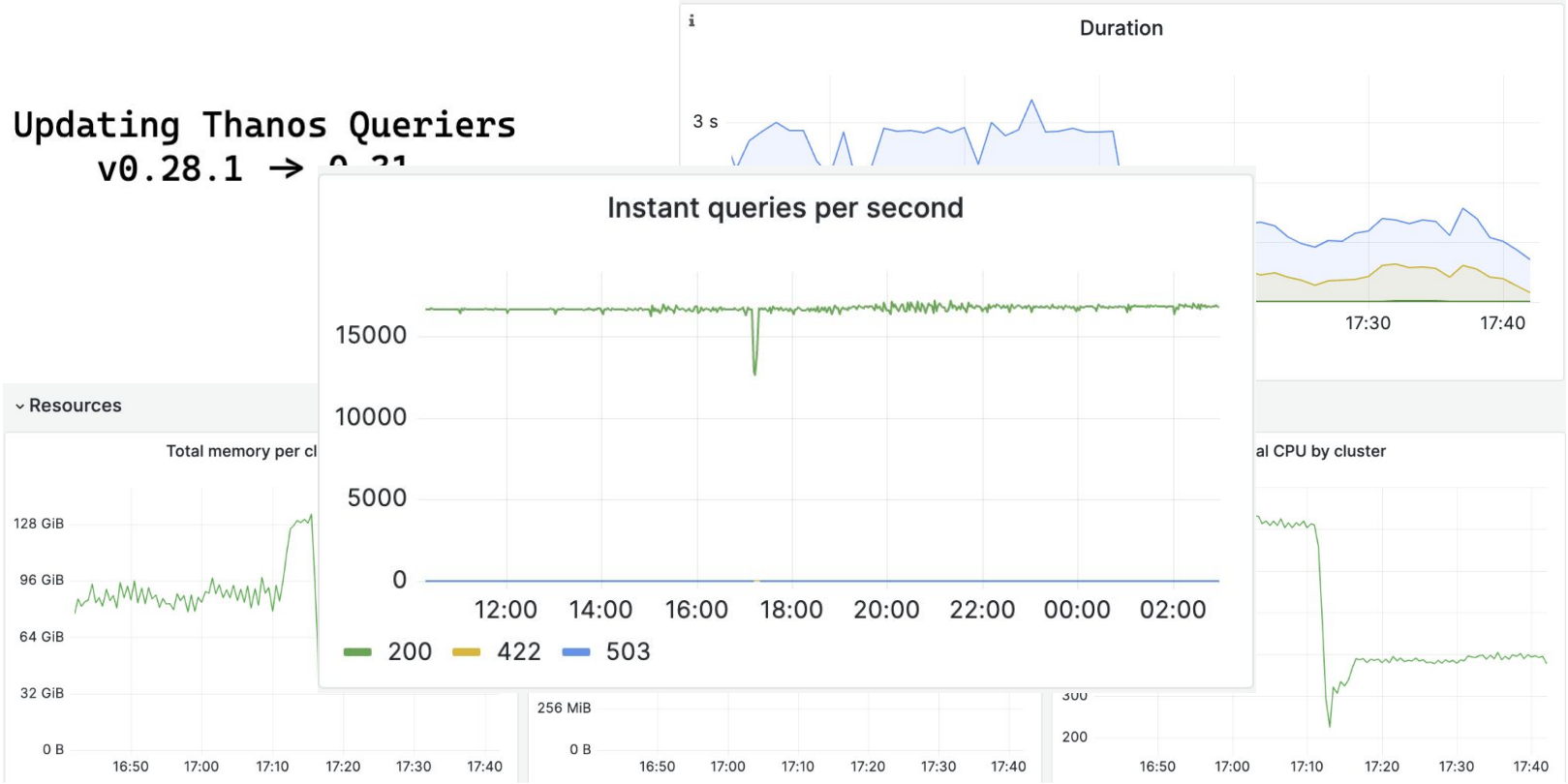
Does Thanos scale to X?

Head series


Prometheus / TSDB Head Series
Last 7 days
cluster_type
All
Head Series across all regions
6 Bil
4 Bil
03/30 00:00
03/30 12:00
03/31 00:00
03/31 12:00
04/01 00:00
04/01 12:00
04/02 00:00
04/02 12:00
04/03 00:00
04/03 12:00
04/04 00:00
04/04 12:00
04/05 00:00
04/05 12:00
global
Head Series with replication (US Regions)
US Central
1.50 Bil
1 Bil
500 Mil
03/30 00:00
03/31 00:00
04/01 00:00
04/02 00:00
04/03 00:00
04/04 00:00
04/05 00:00
us-central
US East
1.50 Bil
1 Bil
500 Mil
03/30 00:00
03/31 00:00
04/01 00:00
04/02 00:00
04/03 00:00
04/04 00:00
04/05 00:00
us-east
Head Series with replication (Other Regions)
APAC
8 Mil
6 Mil
4 Mil
03/30 00:00
03/31 00:00
04/01 00:00
04/02 00:00
04/03 00:00
04/04 00:00
04/05 00:00
APAC
EU West
100 Mil
75 Mil
50 Mil
03/30 00:00
03/31 00:00
04/01 00:00
04/02 00:00
04/03 00:00
04/04 00:00
04/05 00:00
eu-west

Constant improvements

Updating Thanos Queriers
v0.28.1 → 0.31



Thanos Community

How to get involved?

Thanos Community: Communication

Primarily happens on #thanos and #thanos-dev CNCF slack channels.

For code/bugs/issues/feature requests, GitHub issues/PRs on Thanos or other thanos-io/thanos-community repos.

Thanos Community: Meetings

Thanos Community Office Hours

Bi-weekly on Thursdays @ 2pm UTC

Agenda Doc + CNCF Calendar (not always correct!)

KubeCon ContribFest (or project meetings/booth)!

Thanos Community: Mentorships

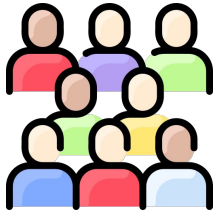
https://thanos.io/tip/contributing/mentorship.md/

CNCF Linux Foundation Mentorships: Nearly every quarter

Google Summer of Code: Once a year

Past successful projects in: https://github.com/cncf/mentoring!

Thanos Community: Share your story!



Thanos Website has new blog space: https://thanos.io/blog/!

We'd love to hear from you!

Share how you are adopting or using Thanos, and get feedback from the community!

@ThanosMetrics

Thank You!

https://thanos.io