chronosphere

chronosphere.io

Stream vs. batch: Leveraging M3 and Thanos for real-time aggregation

Gibbs Cullen, Developer Advocate @ Chronosphere

October 14, 2021

Who am I?

Gibbs Cullen

Developer Advocate @ Chronosphere

→ M3 contributor

→ CNCF O11y TAG member

→ Former AWS PM

→ Twitter: @gibbscullen

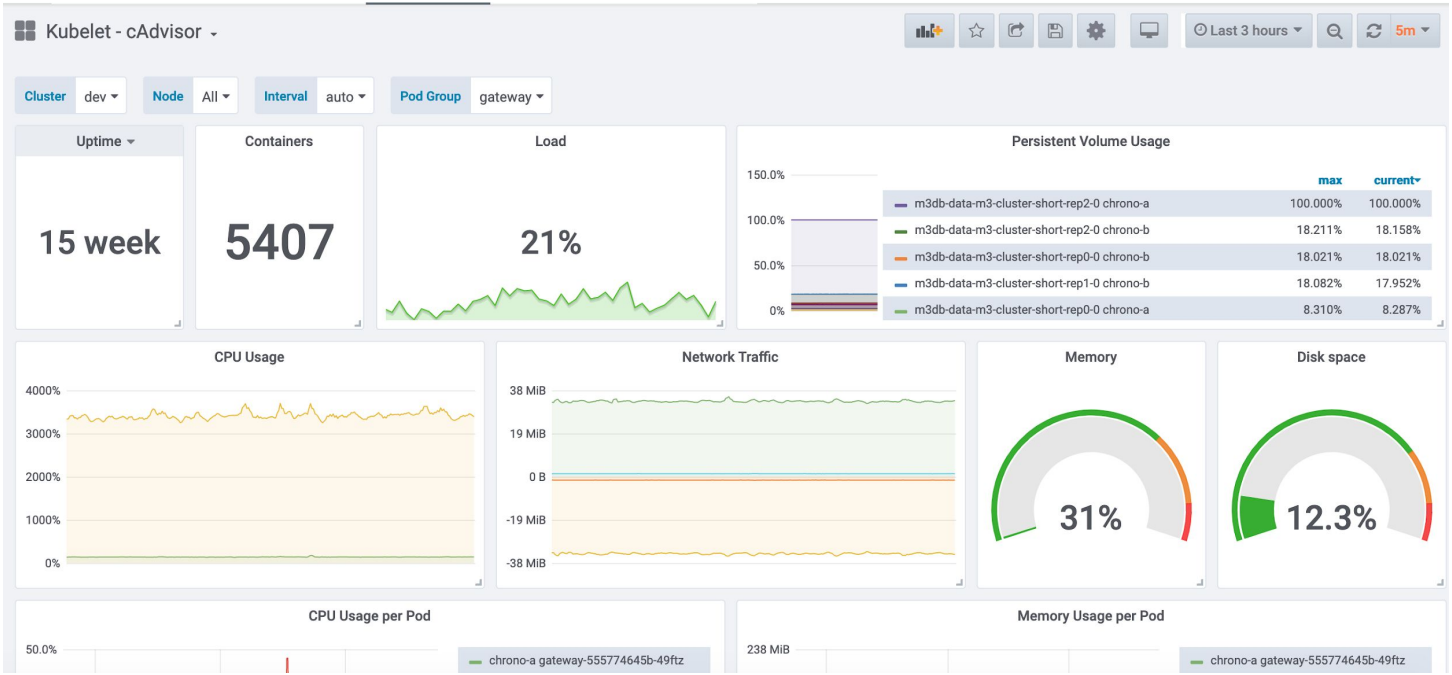
Agenda

- Overview of stream and batch
- Stream aggregation with M3
- Batch aggregation with Thanos
- Stream vs. batch: How to choose?

Why aggregation matters for real time

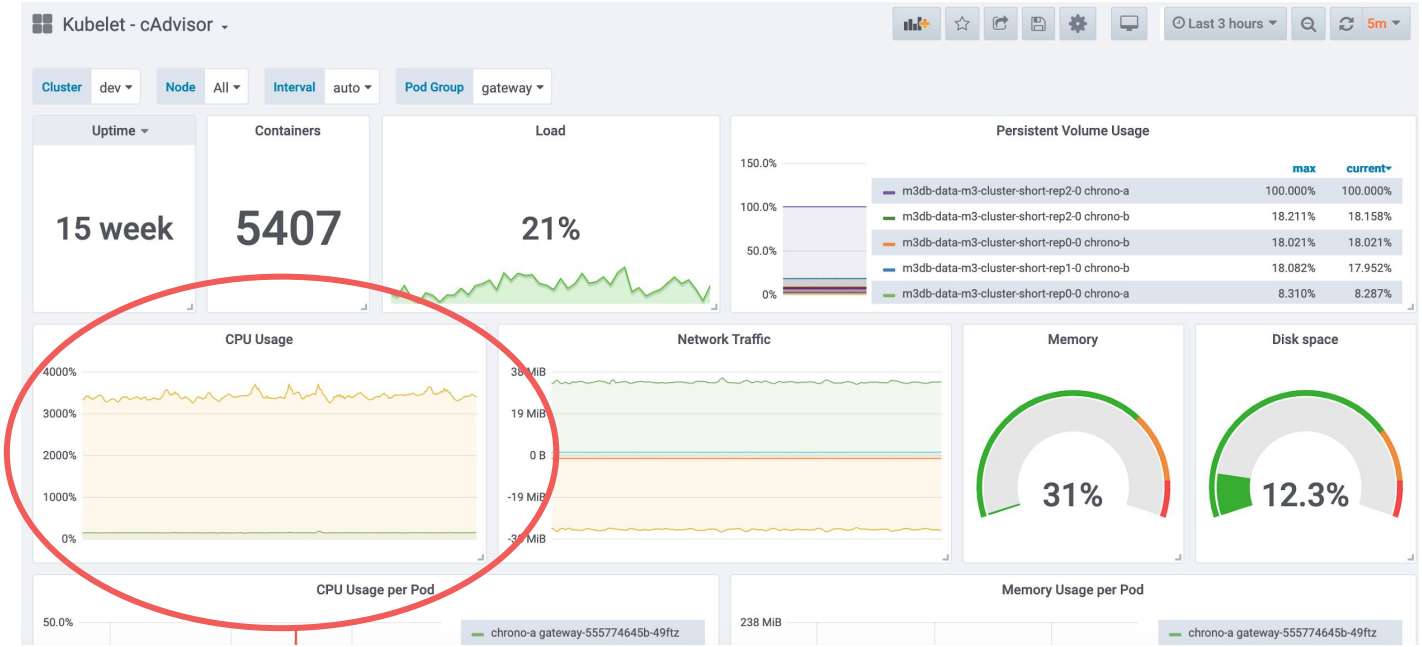
High Cardinality Metrics Example

cAdvisor – resource usage and performance metrics of running containers



High Cardinality Metrics Example

cAdvisor – resource usage and performance metrics of running containers

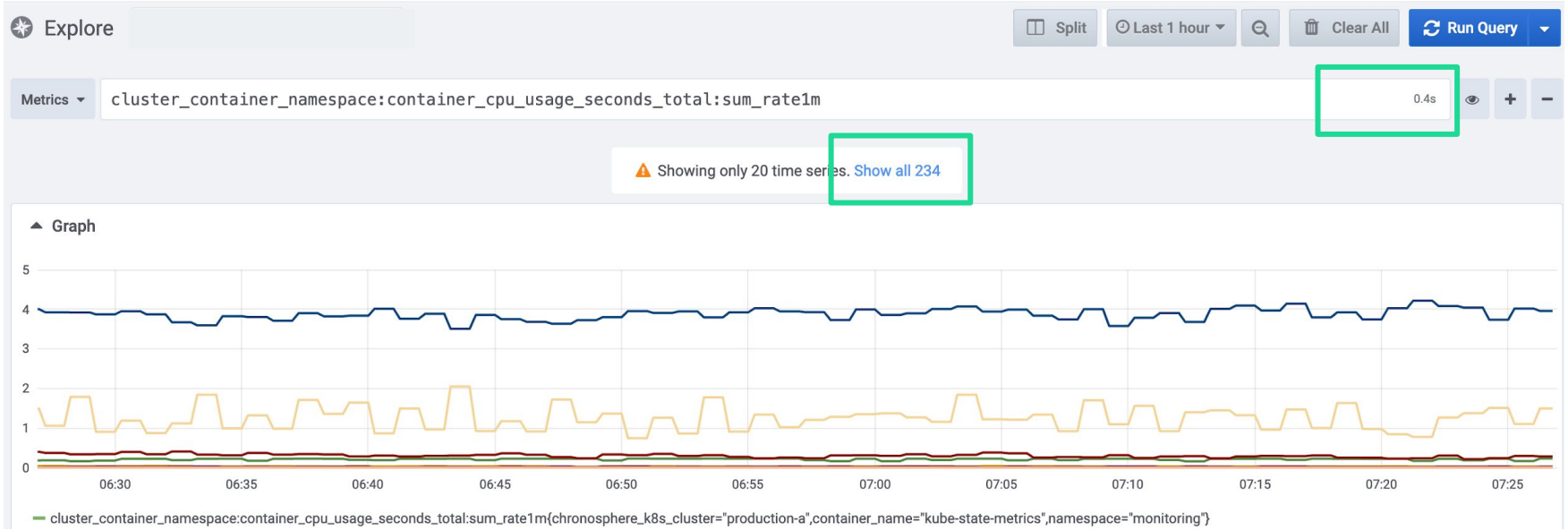


Querying without aggregation

Container CPU usage has ~51k series and takes 20s to query...

Vs. querying with aggregation

Same metric but aggregated to just two labels - ~230 series, 0.4s to query



What is stream and batch aggregation?

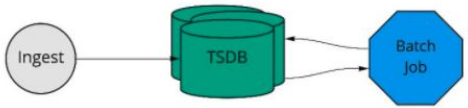
Stream and batch aggregation

Stream

- Data streams continuously
- Aggregation performed in memory before writing to TSDB
- Typically meant for information that's needed immediately
- Data aggregated in real-time and immediately available

Batch

- Data collected over time
- Aggregation performed by reading raw metrics from TSDB and writing back aggregated metric data
- Typically meant for large quantities of information that aren't time-sensitive
- Data aggregated in batches over time

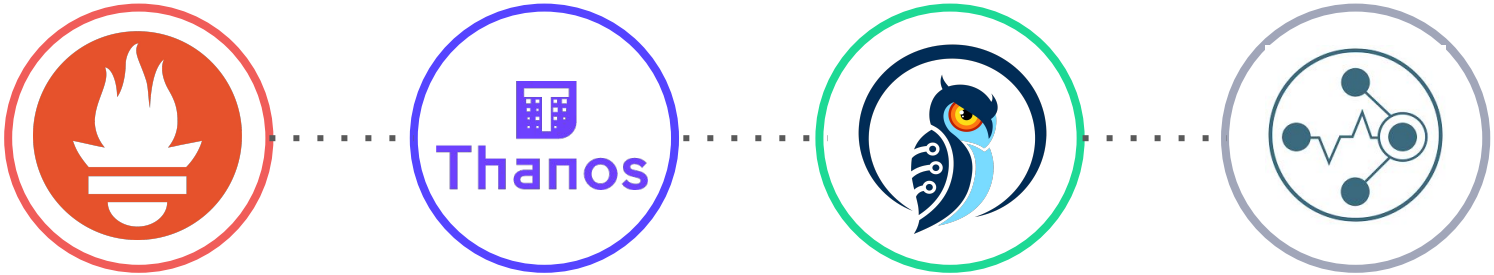


Aggregation with Prometheus

Prometheus Recording Rules

- Allows pre-computing queries and storing back aggregate time series to the TSDB
- Execute and pre-compute the query at regular intervals in-memory; cron-job type processes
- Useful for dashboards, which need to query the same expression repeatedly every time they refresh
- Full access to Prometheus Query Language (PromQL)

Open source metrics solutions



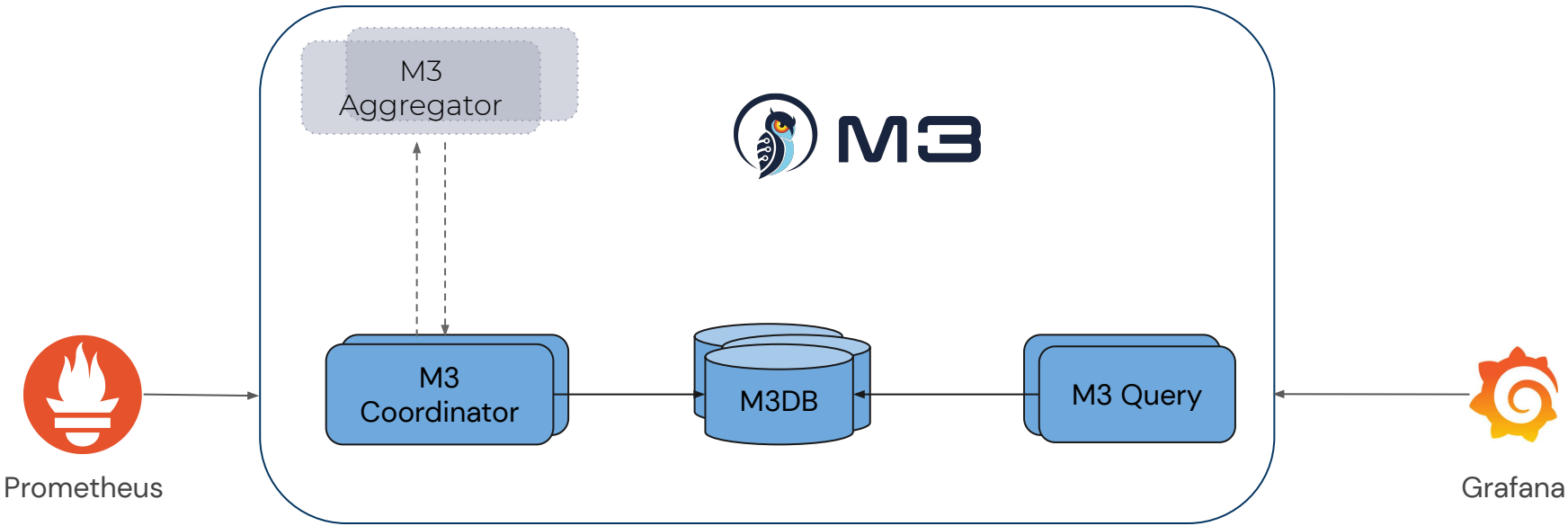
- Prometheus Remote Storage
- PromQL compatible
- Batch and stream aggregation of recording rules

Stream aggregation with

What is M3?

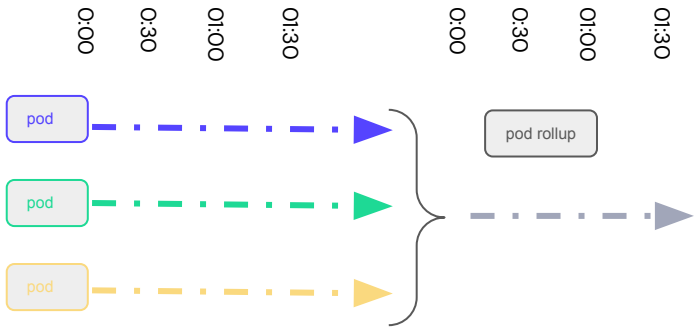
- Open source metrics engine
- Main components:
  - M3DB: Custom built TSDB
  - M3 Coordinator: Optimized ingest and downsampling tier
  - M3 Aggregator: Distributed streaming aggregation tier
  - M3 Query: Optimized query engine
- Built in open source at Uber in 2016
- Prometheus Remote Storage and PromQL Compatible

What is M3?

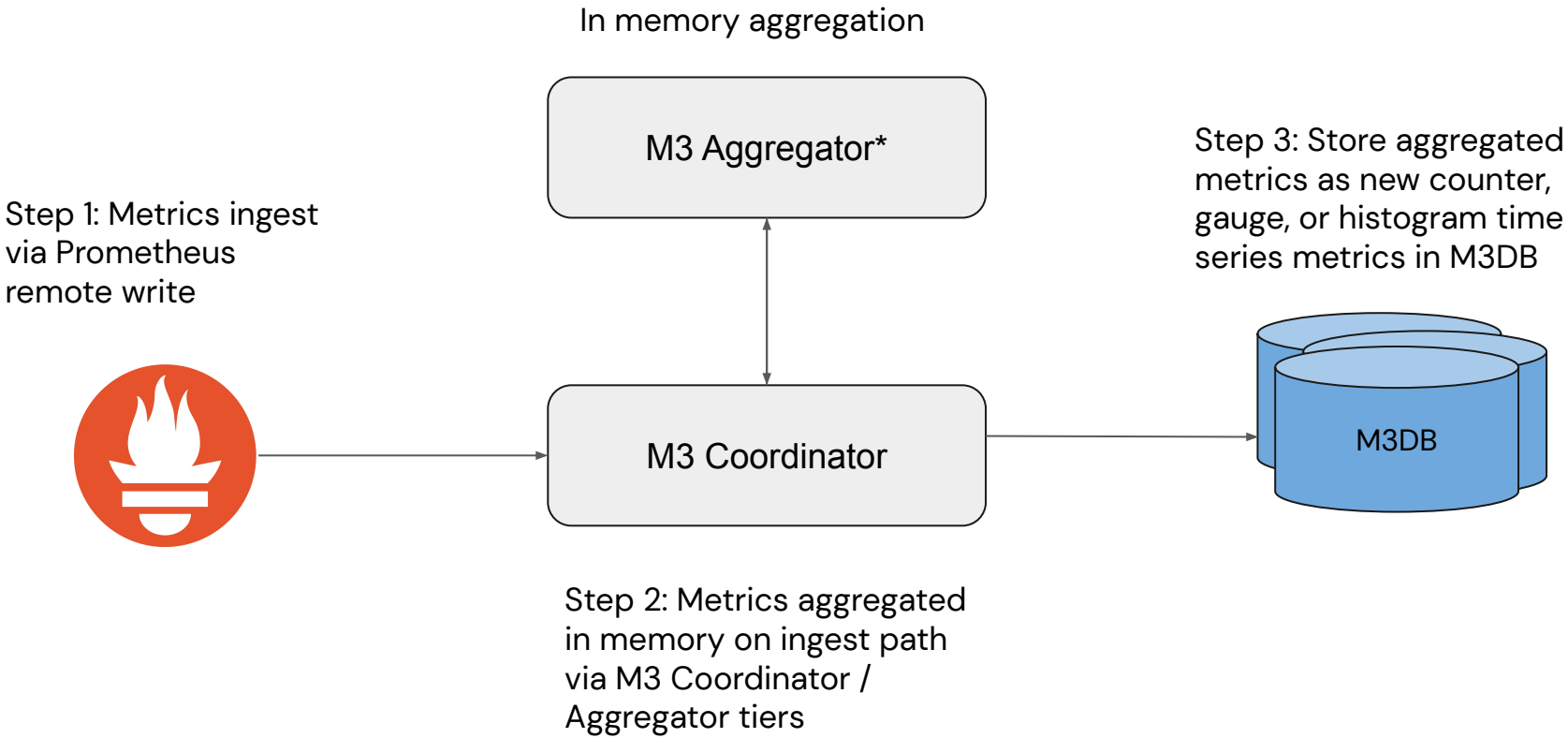

M3
Aggregator
M3
M3
Coordinator
M3DB
M3 Query
Prometheus
Grafana

Streaming aggregation with M3

- Roll up rules - M3's approach to aggregation of high cardinality metrics
- Aggregations are performed in-memory upon ingest via the M3 Aggregator and Coordinator tiers at specified time intervals
- M3 Coordinator writes the aggregated data as reconstituted counter, gauge, or histogram metrics to M3DB as soon as the time window has passed



Streaming aggregation with M3



Pros and cons of streaming aggregation

Pros

- Quick query returns and fast to load dashboards -- real time data and alerting
- Ability to alleviate query requirements for your TSDB, and scale to higher number of alerts and recording rules

Cons

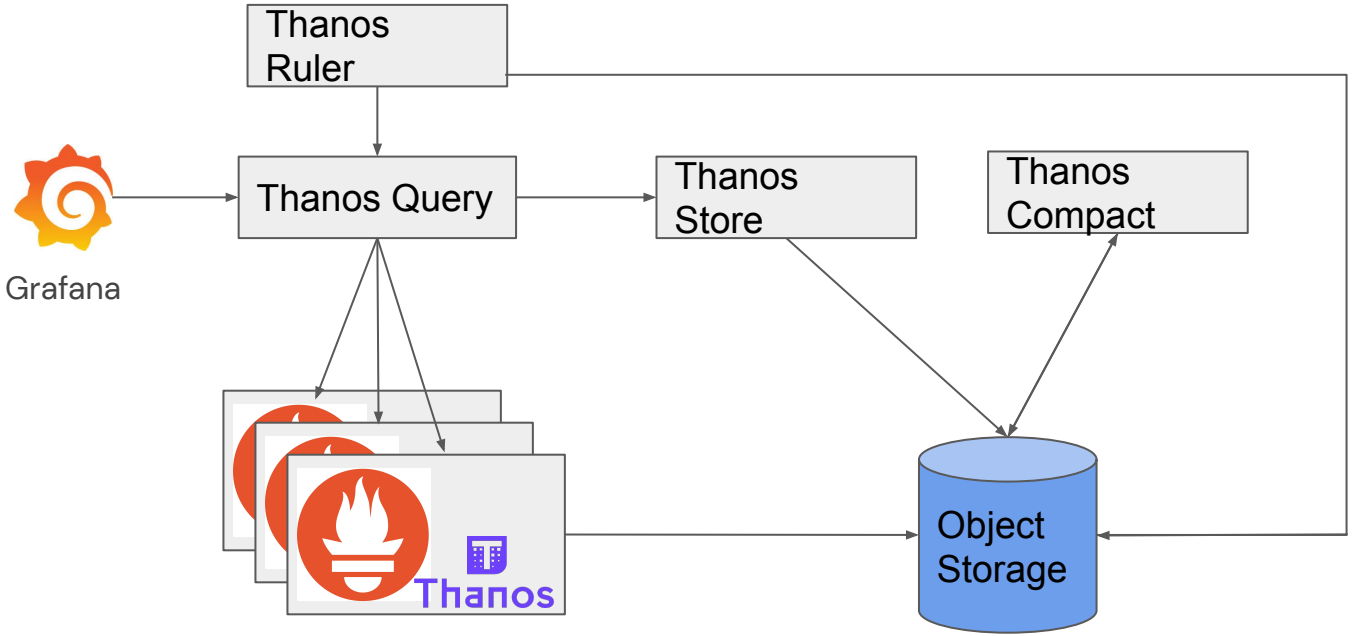
- Complex to operate and deploy
- Doesn't support arbitrary PromQL; instead builds aggregate counter/gauge/histogram metrics

Batch aggregation with

What is Thanos?

- CNCF incubating project
- Originally built by Improbable in open source in 2017
- Main components:
  - Store / Store API: gateway to object store
  - Querier: horizontally scalable and stateless query, aggregation, and deduplication tier
  - Sidecar: proxy for Prometheus via remote write/read
  - Compactor: downsampling and block compaction
  - Rule/Ruler: evaluates Prometheus recording and alerting rules
- Prometheus remote storage and PromQL compatible

What is Thanos?



Batch aggregation with Thanos

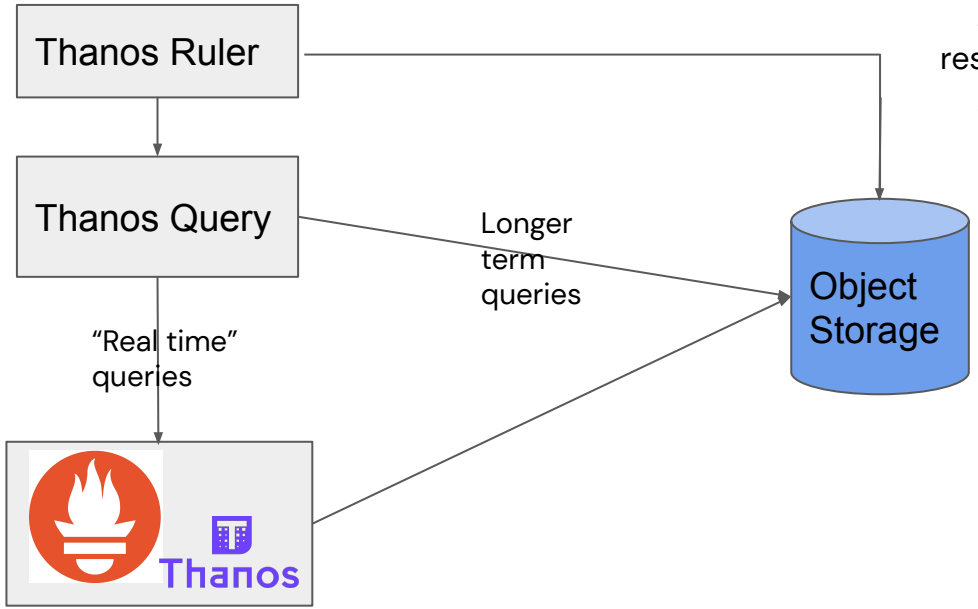
- Raw metrics data collected by Prometheus instances prior to query and aggregation
- Metrics aggregation and PromQL queries performed via Thanos Query
- Thanos Ruler implements Prometheus Recording Rules before writing new aggregated time series to object store

Batch aggregation with Thanos

Step 2: Ruler issues query and metrics are pulled (using reverse index query & reading from storage)

Step 3: Query result evaluated on the metrics pulled

Step 1: Metrics collected by Sidecar store (then to object store in blocks)



Step 4: Aggregated results written to object / long term storage

Pros and cons of batch aggregation

Pros

- Supports full PromQL
- Simple to operate, especially when scaling up or down resources

Cons

- Can lead to large resource consumption
- Can be slow to query metrics, especially when evaluating Recording Rules (or cron-job style queries)

Overview

Recap - How to choose?

Stream with M3

- Alleviate query requirements for your TSDB, and scale to higher number of alerts and recording rules
- Complex to operate and deploy
- Doesn't support arbitrary PromQL; instead builds aggregate counter/gauge/histogram metrics

Batch with Thanos

- Simple to operate, especially when scaling up or down resources
- Supports full PromQL
- Larger resource consumption
- Can be slow to query metrics

...but what about both together? Check out "Streaming Recording Rules for Prometheus, Thanos & Cortex using M3 Coordinator" from PromCon

Thank you