chronosphere

chronosphere.io

Streaming Recording Rules for Prometheus, Thanos & Cortex using M3 Coordinator

Who are we?

Gibbs Cullen
Developer Advocate @ Chronosphere

→ M3 contributor
→ CNCF O11y TAG member
→ Twitter: @gibbscullen

Rob Skillington
Co-Founder and CTO @ Chronosphere

→ M3DB creator
→ OpenMetrics contributor
→ Twitter: @roskilli

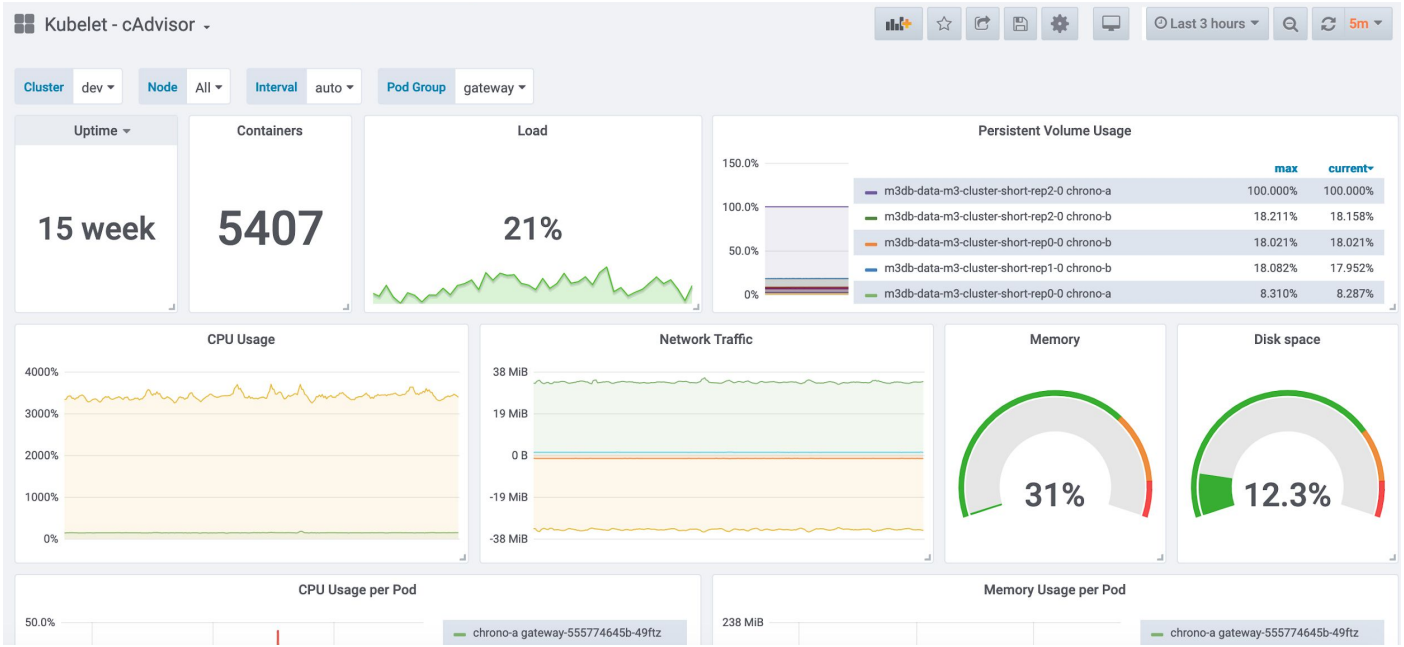
Agenda

- The problem
- Aggregation with Prometheus Recording Rules
- Streaming aggregation with Prometheus and M3 coordinator
- Live demos
- Q&A

Querying high cardinality metrics taking too long?

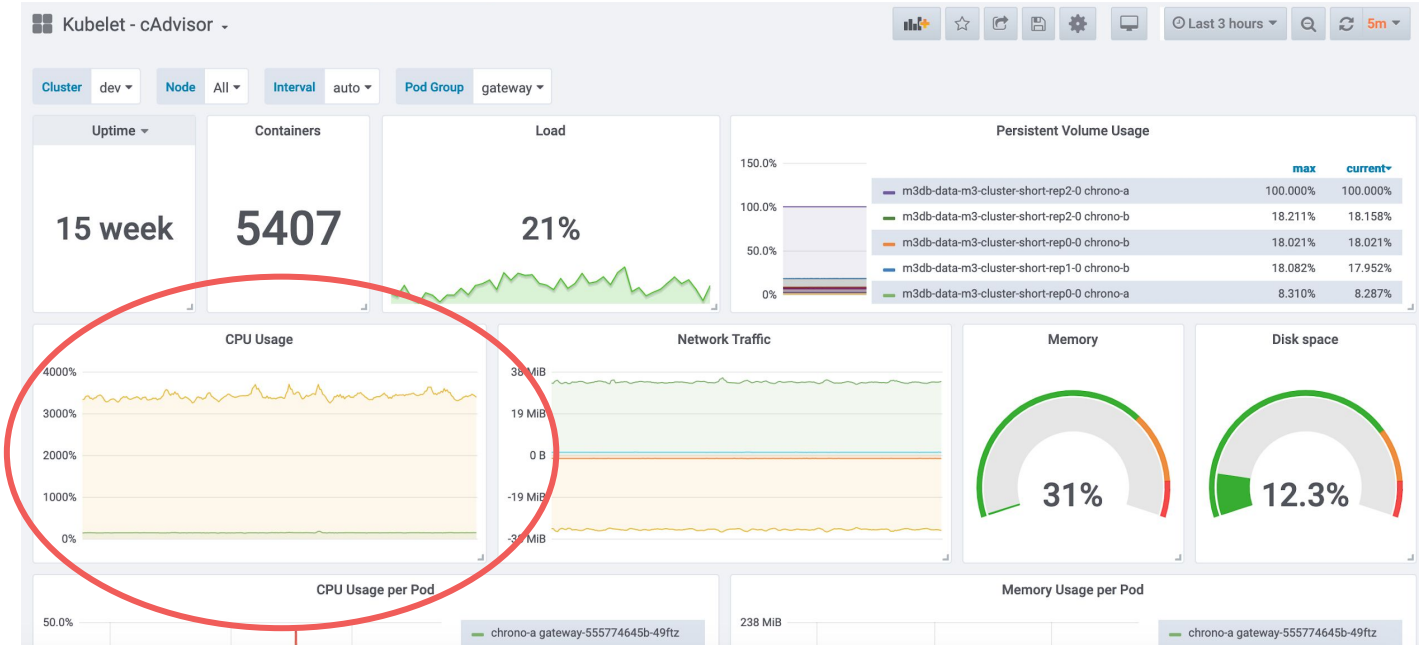
High Cardinality Metrics Example

cAdvisor – resource usage and performance metrics of running containers



High Cardinality Metrics Example

cAdvisor – resource usage and performance metrics of running containers

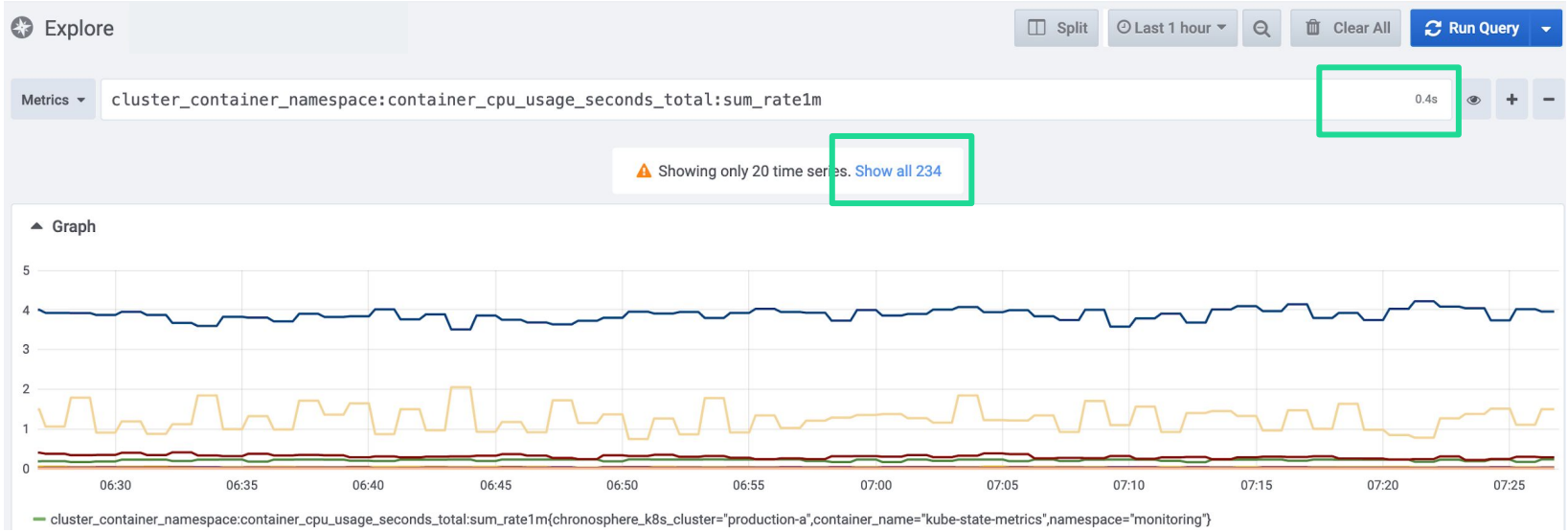


Querying without aggregation

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

Versus querying with aggregation

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



Aggregation with Prometheus Recording Rules

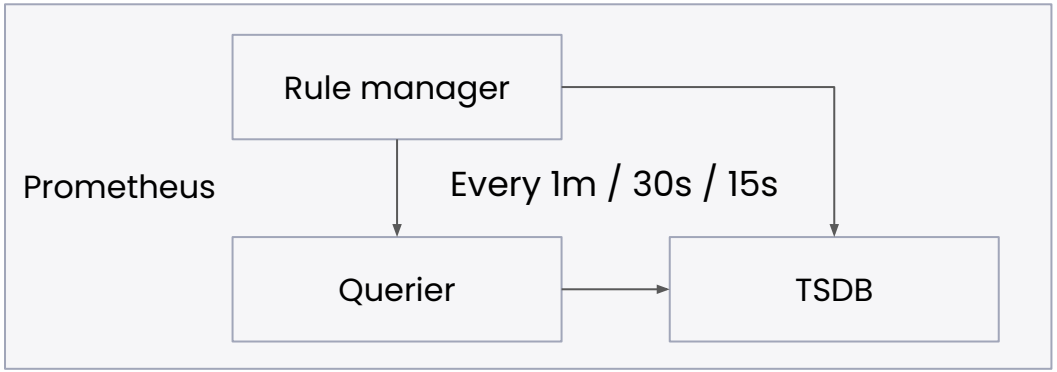
What are Prometheus recording rules?

- Allows pre-computing queries and storing back aggregate time series to the TSDB
- Execute and pre-compute the query at regular intervals; cron-job type processes
- Useful for dashboards, which need to query the same expression repeatedly every time they refresh

```
- record: cluster_container_namespace:container_cpu_usage_seconds_total:sum_rate1m
  expr: sum(rate(container_cpu_usage_seconds_total[1m])) by (container_name, namespace)
```

How does it work in Prometheus?

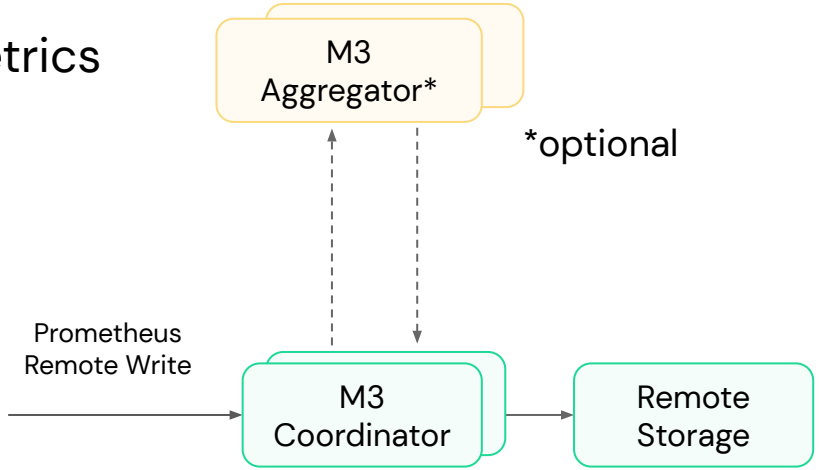
```
- record: cluster_container_namespace:container_cpu_usage_seconds_total:sum_rate1m
  expr: sum(rate(container_cpu_usage_seconds_total[1m])) by (container_name, namespace)
```



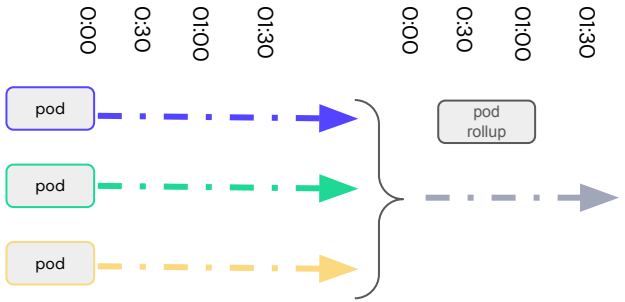
What about streaming aggregation with M3?

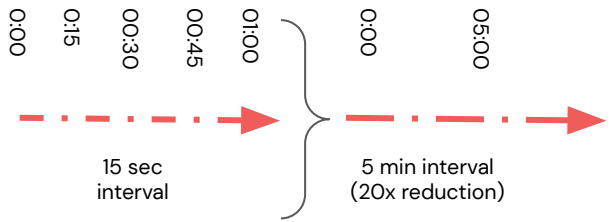
Yes... let's do it using M3 aggregation!

- M3 is a remote storage for Prometheus
- Move aggregations from recording rule to streaming aggregation with M3
- Rollup rules allow aggregating metrics
- Mapping rules allow downsampling metrics



Two ways to aggregate with M3 coordinator





Rollup with rollup rules

- Aggregates across multiple time series to reduce query-time cardinality

Downsampling with mapping rules

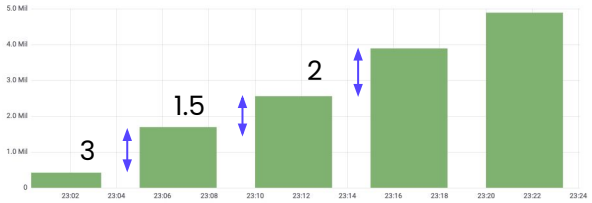
- Aggregates within a single time series, to make querying over long period efficient (fewer samples)

Let’s walk through an example

M3 Rollup Rules

Rollup rules contain a series of transforms applied in order.
The filter acts as a PromQL metric selector defined using glob syntax.

Step 1: Take the increases



Step 2: Sum by label

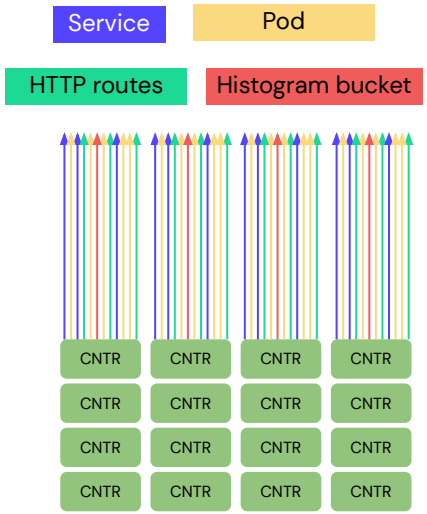
Step 3: Create monotonic cumulative counter.

```
  - name: "cAdvisor CPU usage aggregate"
    filter: "__name__:container_cpu_usage_seconds_total namespace:*
le:* name:* instance:* container_name:*"
    transforms:
    - transform:
        type: "Increase"
    - rollup:
        metricName: "container_cpu_usage_seconds_total"
        groupBy: ["container_name", "namespace"]
        aggregations: ["Sum"]
    - transform:
        type: "Add"
    storagePolicies:
    - resolution: 30s
      retention: 720h
```

How are the different approaches different?

Aggregation example

Single aggregation for one service's HTTP/gRPC endpoints



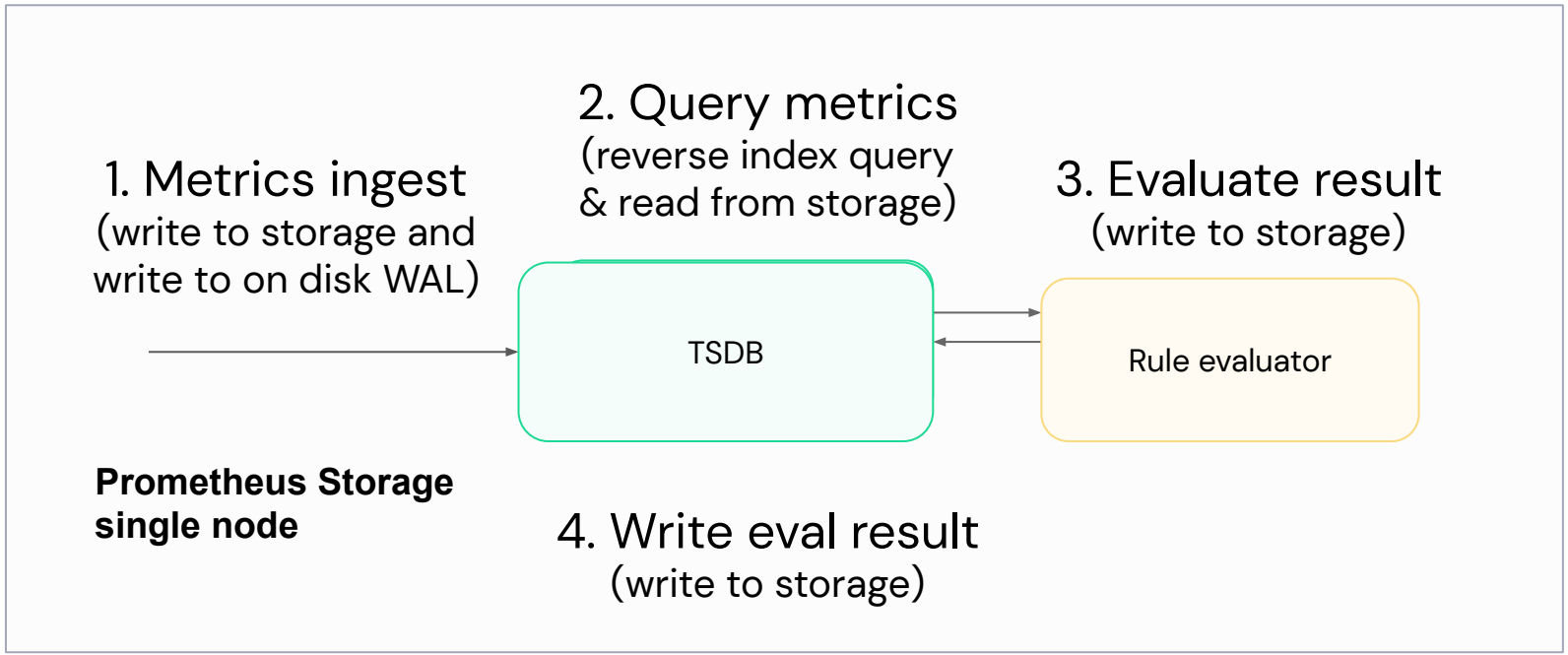
1 service or application
200 pods
20 HTTP endpoints or gRPC methods
5 common status codes
30 histogram buckets

= 600 **thousand** unique time series

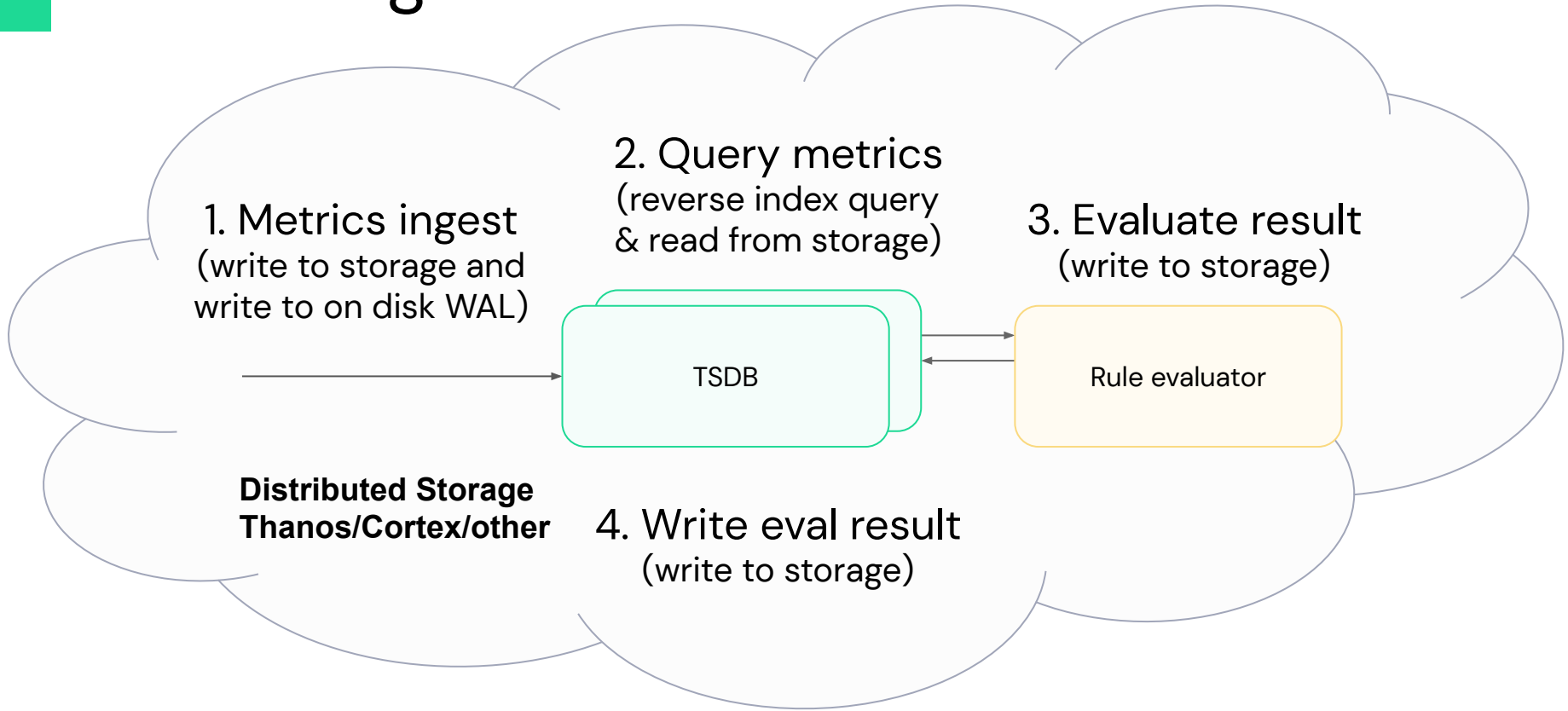
And wouldn't it be nice to slice and dice by:

- Server versions by git revision (at least 2x so 1.2 million series)
- Mobile client or web bundle versions (at least 2x so 2.4 million series)

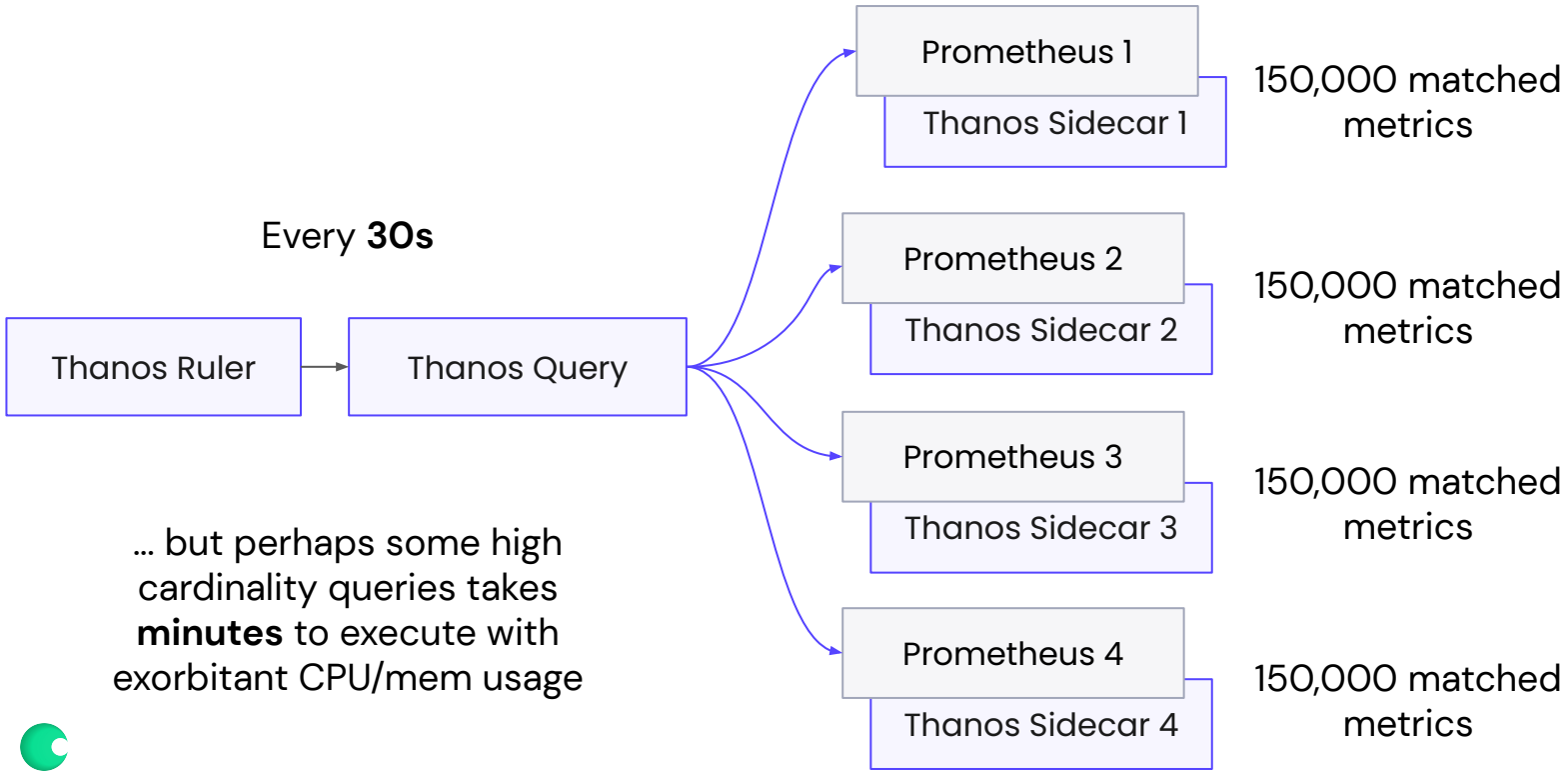
Recording rules



Recording rules

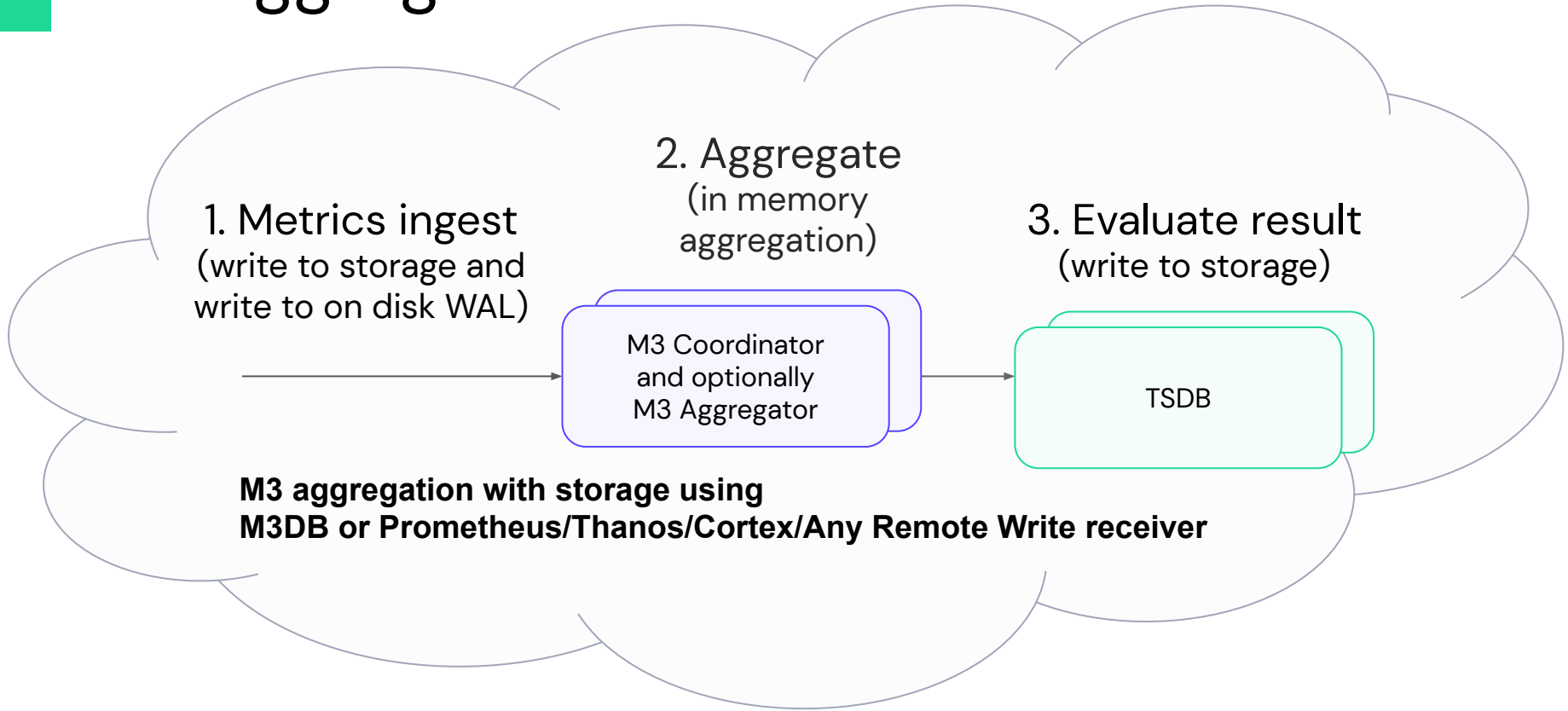


Recording rule resource usage example



... but perhaps some high cardinality queries takes **minutes** to execute with exorbitant CPU/mem usage

M3 aggregation

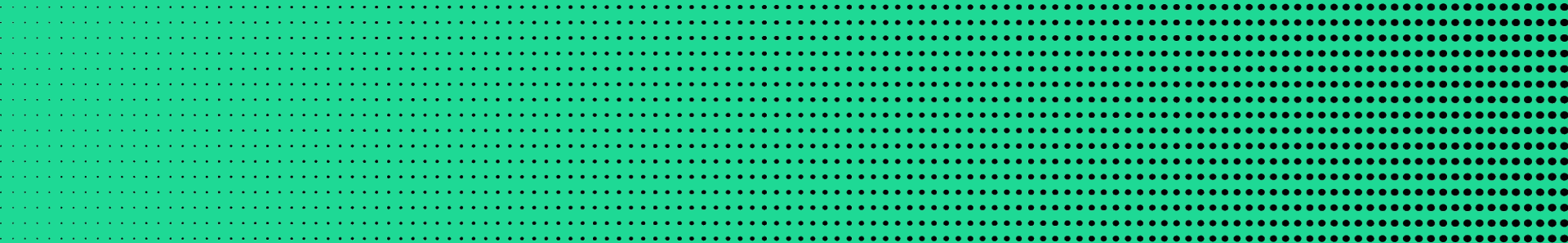


M3 aggregation supplementing recording rules

1. Save frequently on expensive step with high cardinality reverse index query and read from storage
2. Scale aggregation independent of TSDB query resources
3. Spread aggregation of single individual rules over nodes
4. Use template aggregation rule to make life easier

```
metricName: “{{ .MetricName }}:without_instance”
excludeBy: [“instance”]
...
```

Demo time



Thank you

https://m3db.io/docs/integrations/prometheus_aggregation/