KubeCon CloudNativeCon North America 2026

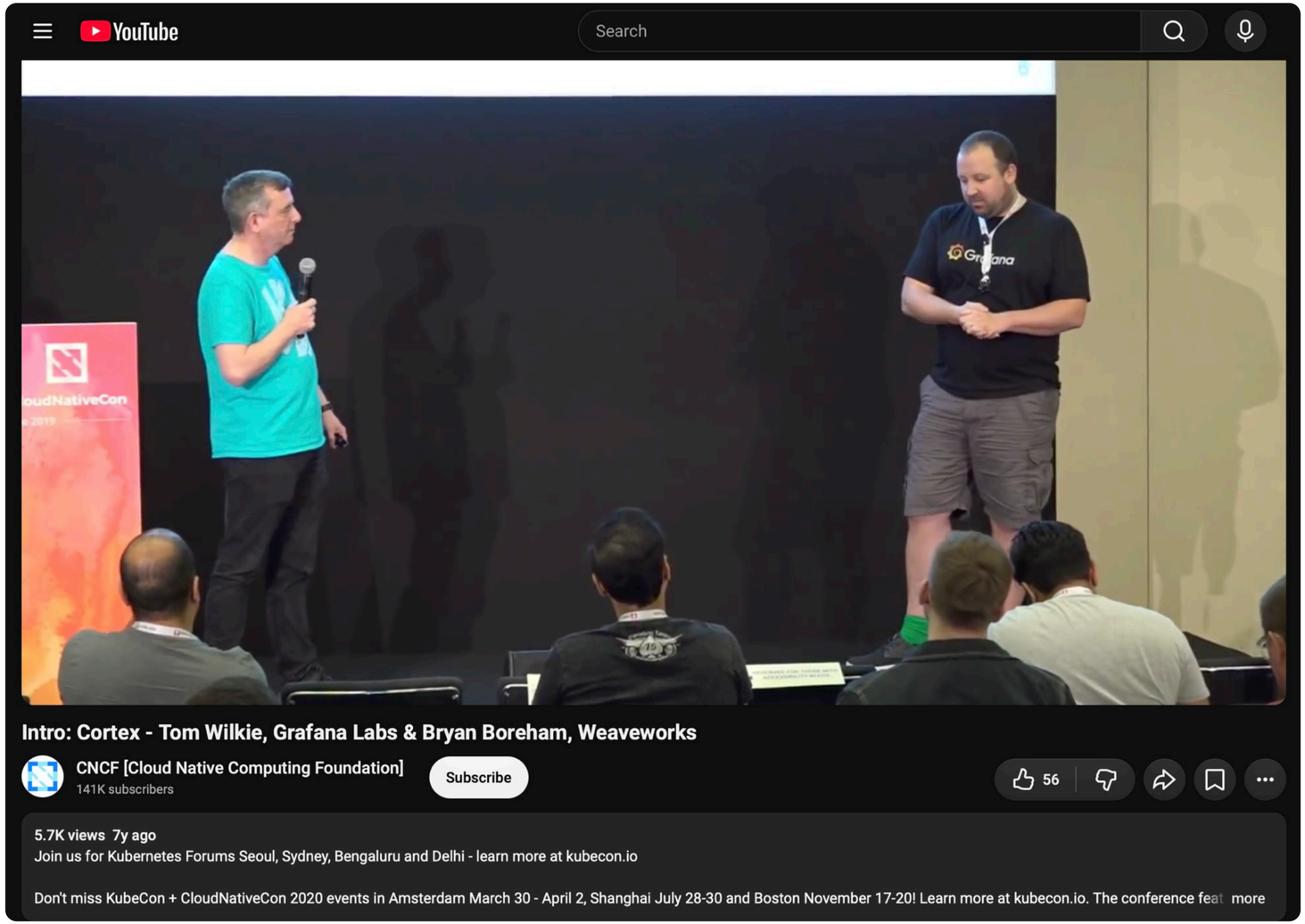
#KubeCon #CloudNativeCon

Cortex:
Surviving High Cardinality Metrics

Friedrich Gonzalez, Software Engineer at Apple

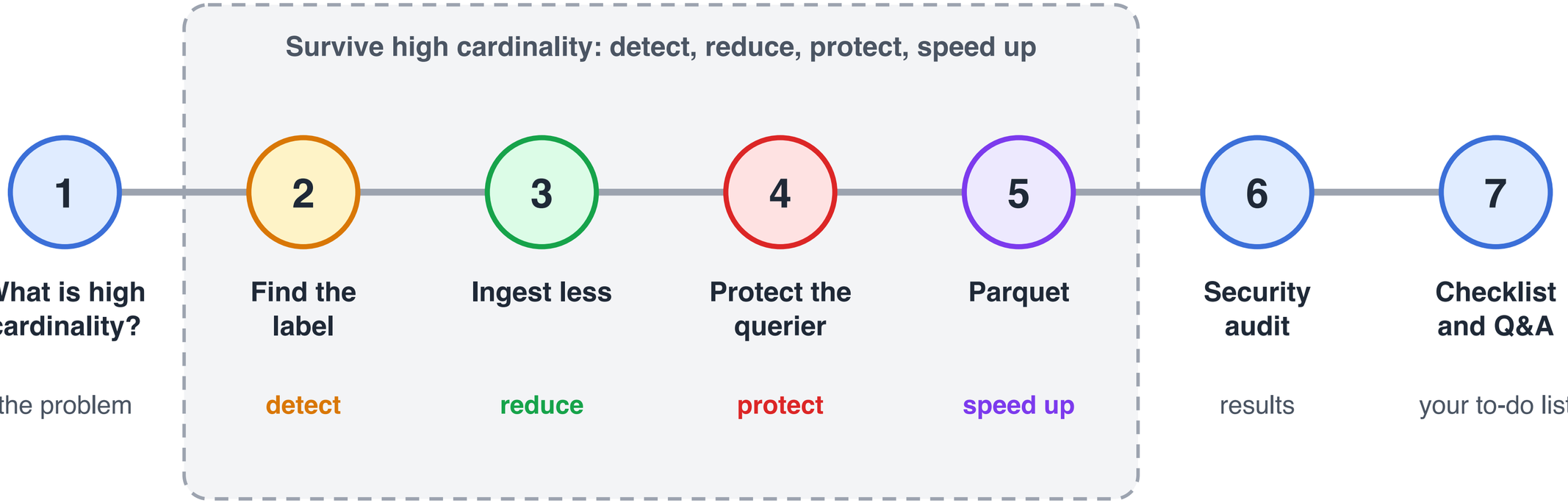


My first Cortex talk



Agenda

The path through this talk



KubeCon CloudNativeCon North America 2026

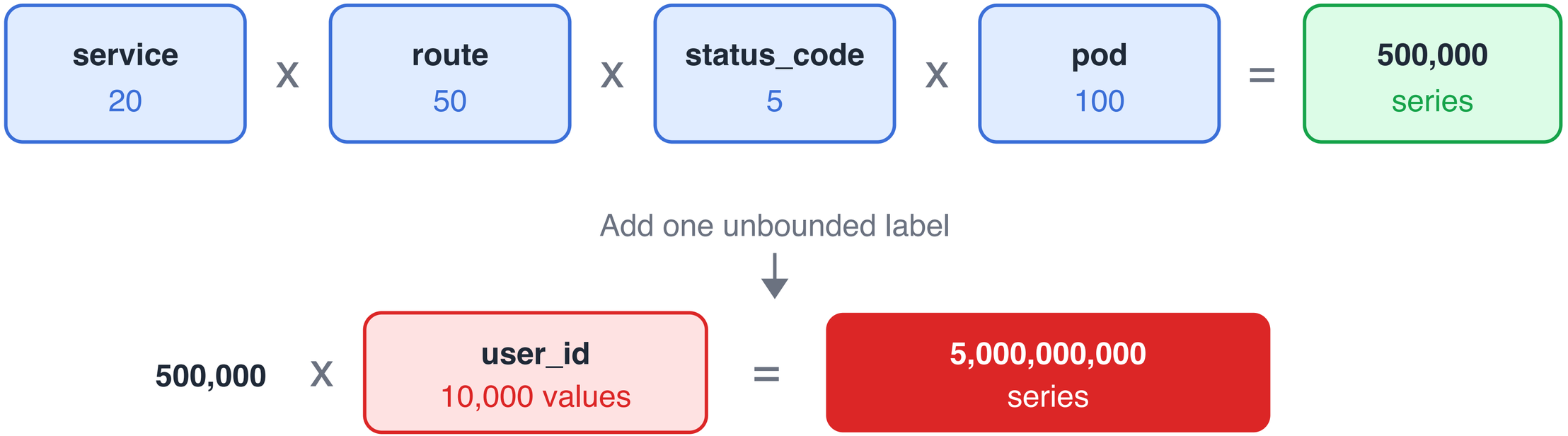
PART 1

What is high cardinality?



A series is a unique label set

Each label multiplies the series count



Labels that cause runaway cardinality

Bounded labels and unbounded labels

Bounded: good

`status_code` 5 values

`route="/users/{id}"` a template

`service` a fixed list

Unbounded: bad

`user_id`

`session_id, request_id, trace_id`

`path="/users/12345/orders/987"`

Pod UID, container ID

Timestamps or build hashes

Error messages

Churn: every deploy makes new pod series



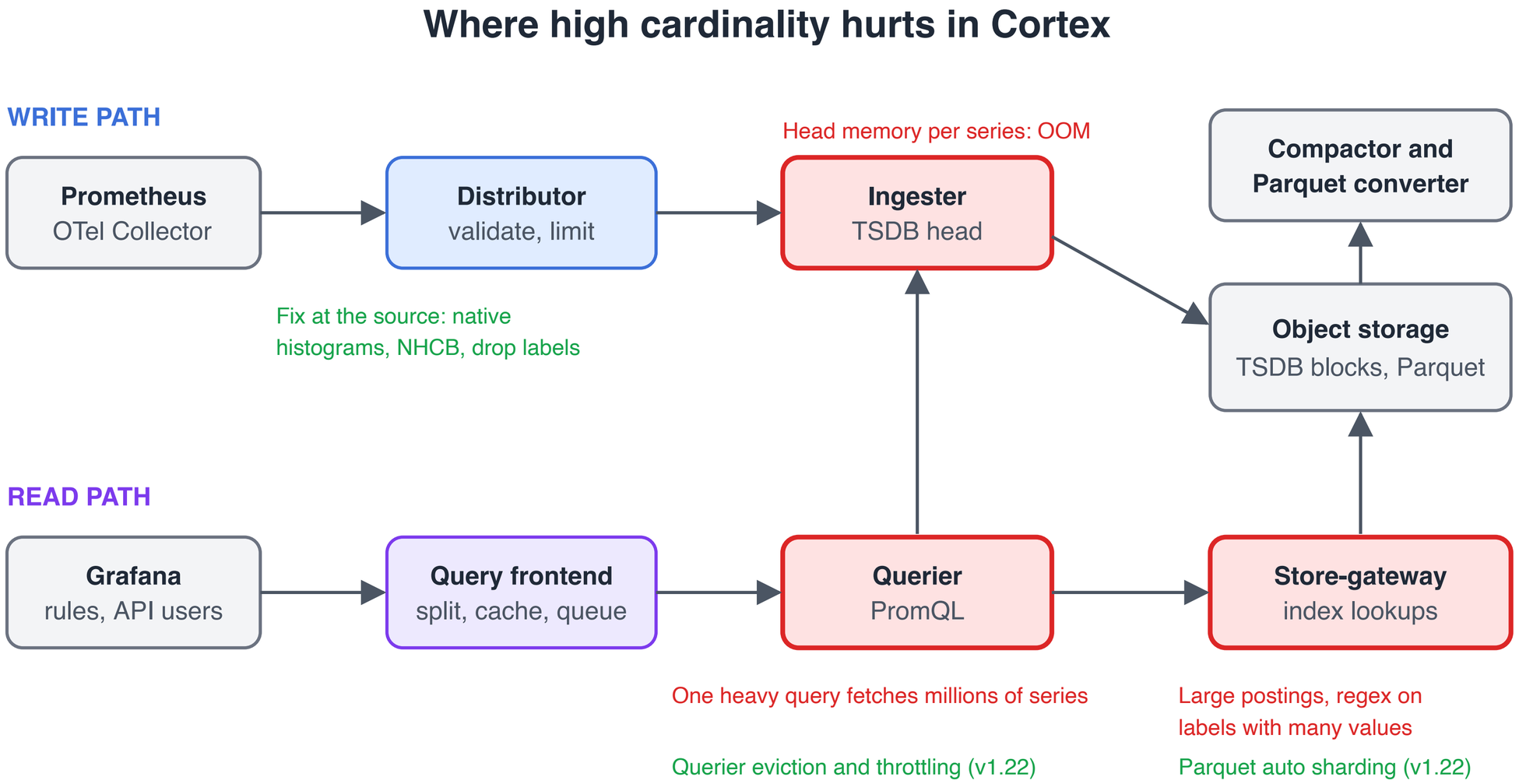
Old pod series stay in the head until head compaction.

Head memory grows during deploys.

new series old series, still in the head

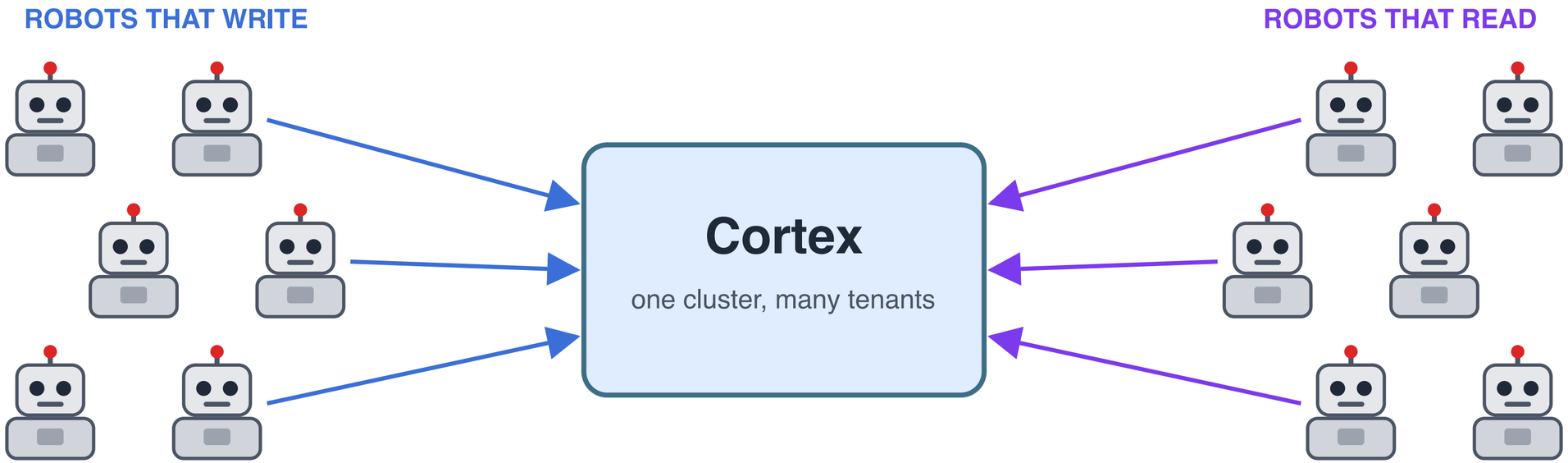
Rule: label values come from a small, bounded set. Put unbounded data in logs, traces or exemplars.

Where it hurts in Cortex



2026: many robots, one Cortex

More writers, more readers, same cluster



Write path

Each run can add new label values:
`agent_id, session_id, task_id`
More series and more churn per tenant.

Read path

Robots query all day, in parallel.
They do not know which query is wide.
More load on the querier.

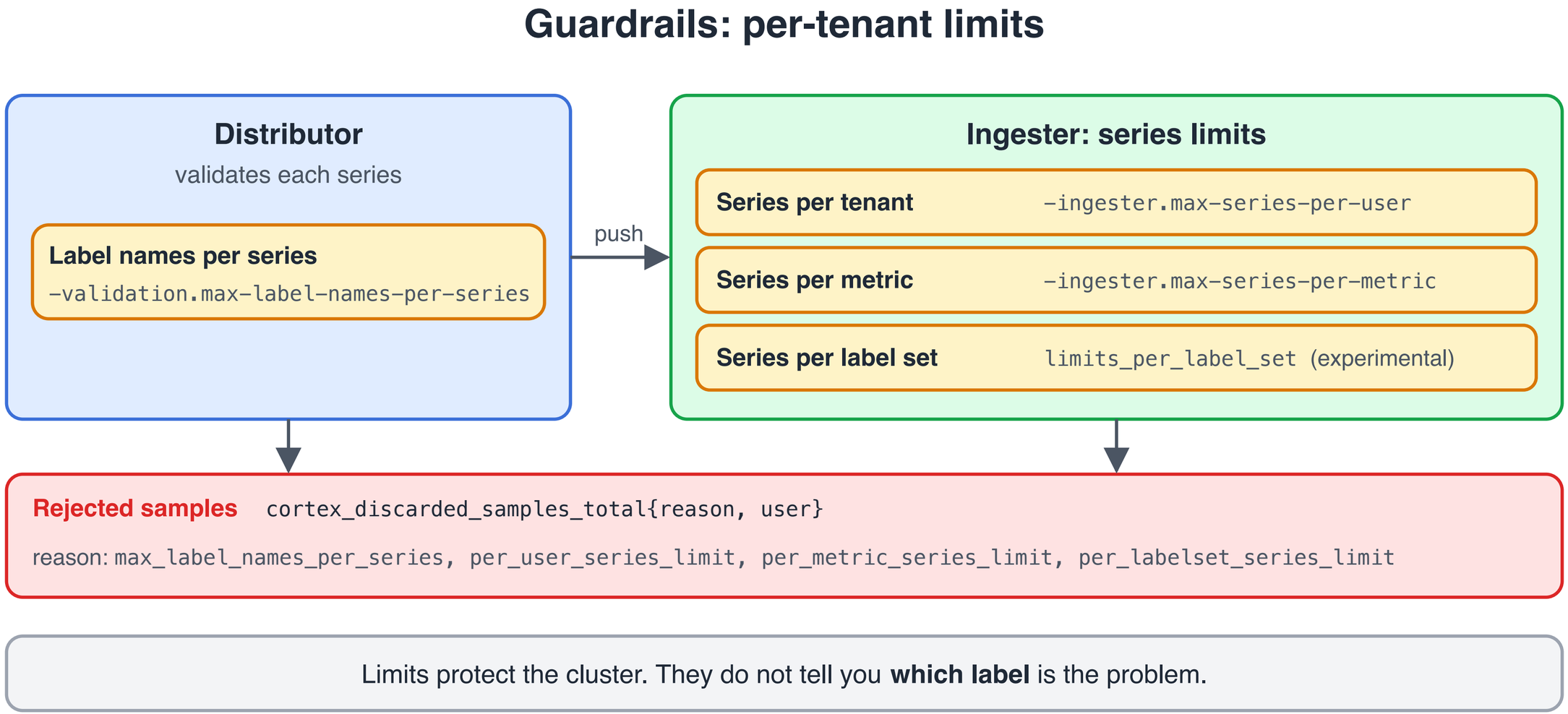
KubeCon CloudNativeCon North America 2026

PART 2

Find the label that drives growth inside a tenant



Guardrails that exist today



New in v1.22.0: active series trackers (experimental)

Active series trackers (v1.22.0, experimental)

Two trackers on http_requests_total

`http_all`
`{__name__="http_requests_total"}`

all series of the metric

`http_with_user_id`
`{__name__="http_requests_total", user_id!=""}`

series that have user_id

Compare the two bars: the red bar is what **user_id** adds.

Example bar lengths, not real data.

`cortex_ingester_active_series_per_tracker {user, name}`

Observe only. No limit.

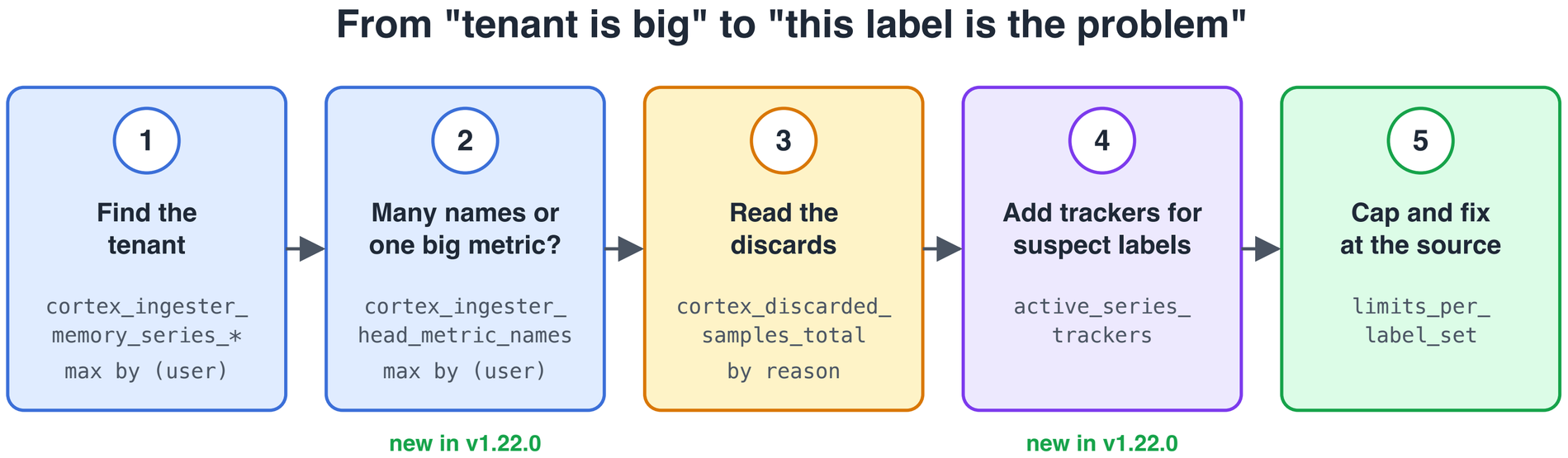
Reload at runtime. No restart.

Per ingester: divide the sum by the replication factor.

Counts head series, with no idle timeout. It does not match `cortex_ingester_active_series`.

Set in the runtime config: `overrides.<tenant>.active_series_trackers` (#7476)

Workflow: from "tenant is big" to "this label is the problem"



Step 1:

```
max by (user) (
    cortex_ingester_memory_series_created_total
  - cortex_ingester_memory_series_removed_total)
```

Step 2:

```
max by (user) (cortex_ingester_head_metric_names)
```

Coming next: a per-tenant cardinality API (#7384, in review)

```
GET /api/v1/cardinality?source=head|blocks
```



PART 3

Ingest less

Send fewer bytes and fewer series to Cortex.



Remote Write 2.0: same series, fewer bytes

Remote Write 1.0 and 2.0 payloads

Remote Write 1.0: full strings in every series

```
job="api" namespace="prod" pod="api-7f9c" le="0.1"
job="api" namespace="prod" pod="api-7f9c" le="0.5"
job="api" namespace="prod" pod="api-7f9c" le="1"
job="api" namespace="prod" pod="api-7f9c" le="+Inf"
```

The same strings repeat in every series of the batch.

Remote Write 2.0: symbols table

symbols (each string one time)

```
0:"" 1:job 2:api 3:namespace 4:prod
5:pod 6:api-7f9c 7:le 8:0.1
9:0.5 10:1 11:+Inf
```

```
labels_refs: [1,2, 3,4, 5,6, 7,8]
labels_refs: [1,2, 3,4, 5,6, 7,9]
labels_refs: [1,2, 3,4, 5,6, 7,10]
```

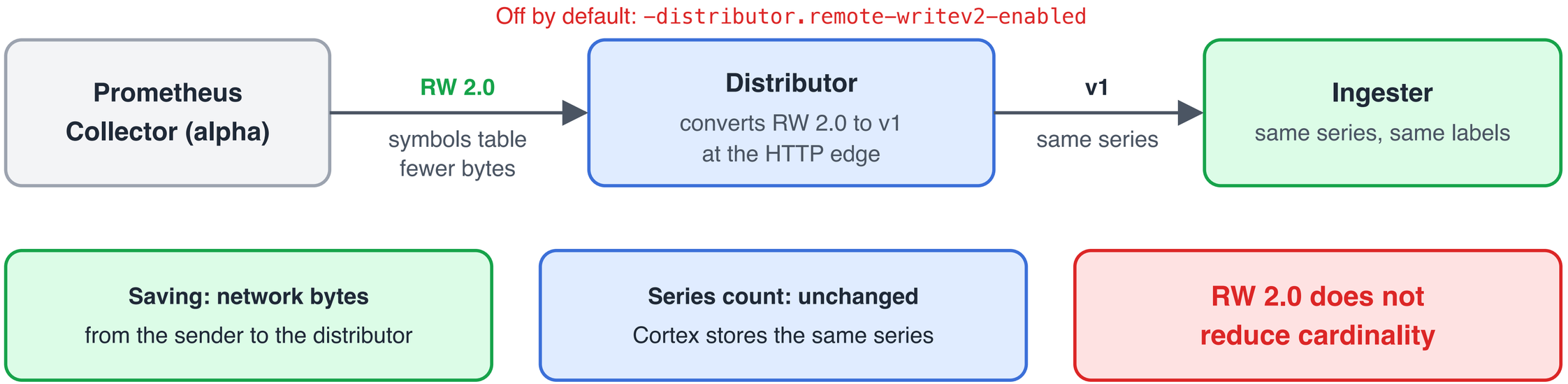
Series refer to strings by small integer index.

RW 2.0 also carries:

native histograms and NHCB, exemplars, metadata per series (type, help, unit), start timestamp

Remote Write 2.0 in Cortex

Remote Write 2.0 in Cortex: same series, fewer bytes



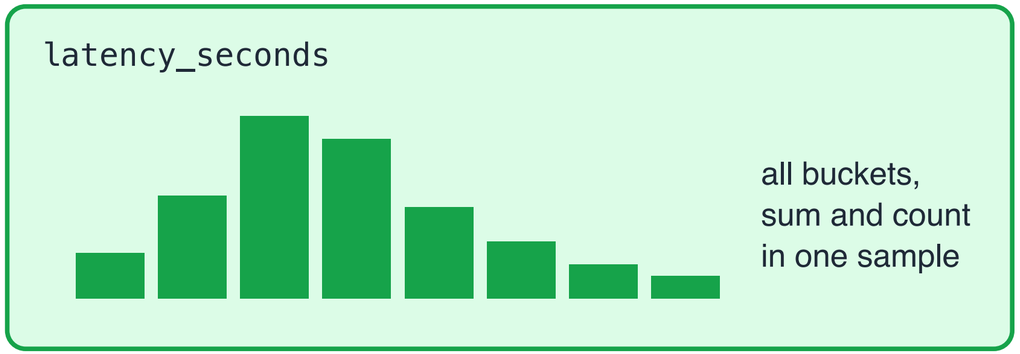
Native histograms: one series instead of many

Classic histogram and native histogram, for one label set

Classic: 13 series (10 bounds + Inf + sum + count)

```
latency_seconds_bucket{le="0.005"}
latency_seconds_bucket{le="0.01"}
latency_seconds_bucket{le="0.025"}
latency_seconds_bucket{le="0.05"}
latency_seconds_bucket{le="0.1"}
latency_seconds_bucket{le="0.25"}
latency_seconds_bucket{le="0.5"}
latency_seconds_bucket{le="1"}
latency_seconds_bucket{le="2.5"}
latency_seconds_bucket{le="5"}
latency_seconds_bucket{le="+Inf"}
latency_seconds_sum
latency_seconds_count
```

Native or NHCB: 1 series



Exponential native histogram

Buckets from a schema (-4 to 8). Sparse.

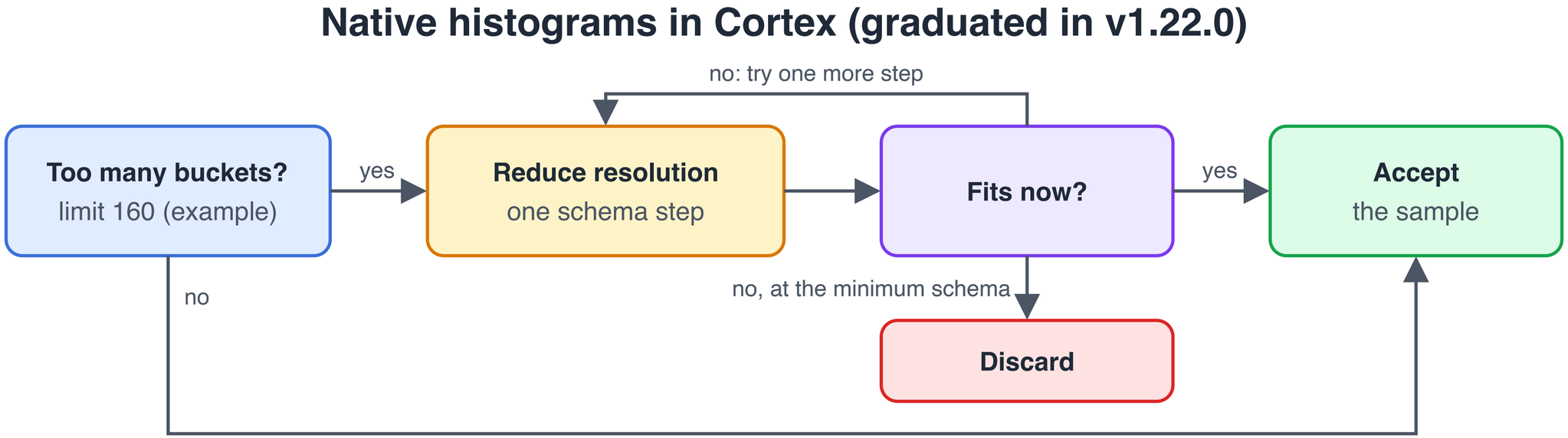
NHCB (schema -53)

Keeps your existing le bounds as custom_values.

5,000 label sets:

classic 65,000 series, native 5,000 series

Native histograms in Cortex v1.22.0



Metric for reduced samples: `cortex_reduced_resolution_histogram_samples_total`

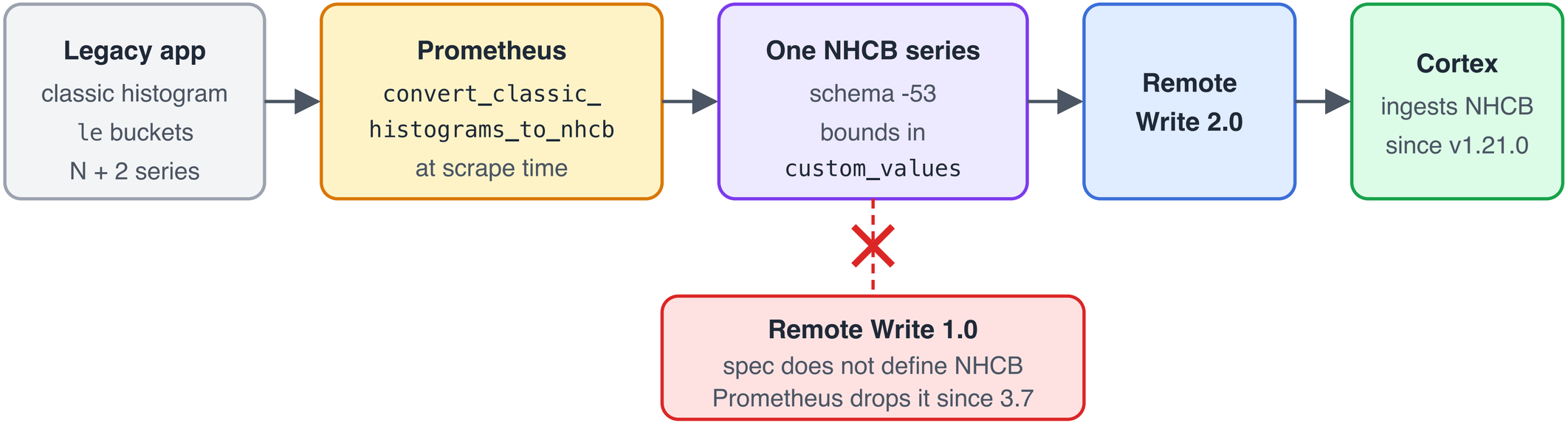
Native histograms off for the tenant: the ingester drops samples, reason `native-histogram-sample`

`-validation.max-native-histogram-size-bytes` (16 KiB) caps the raw protobuf before unmarshal (#7570)

NHCB cannot reduce resolution. An NHCB over the bucket limit is discarded.

NHCB: native histograms with custom buckets

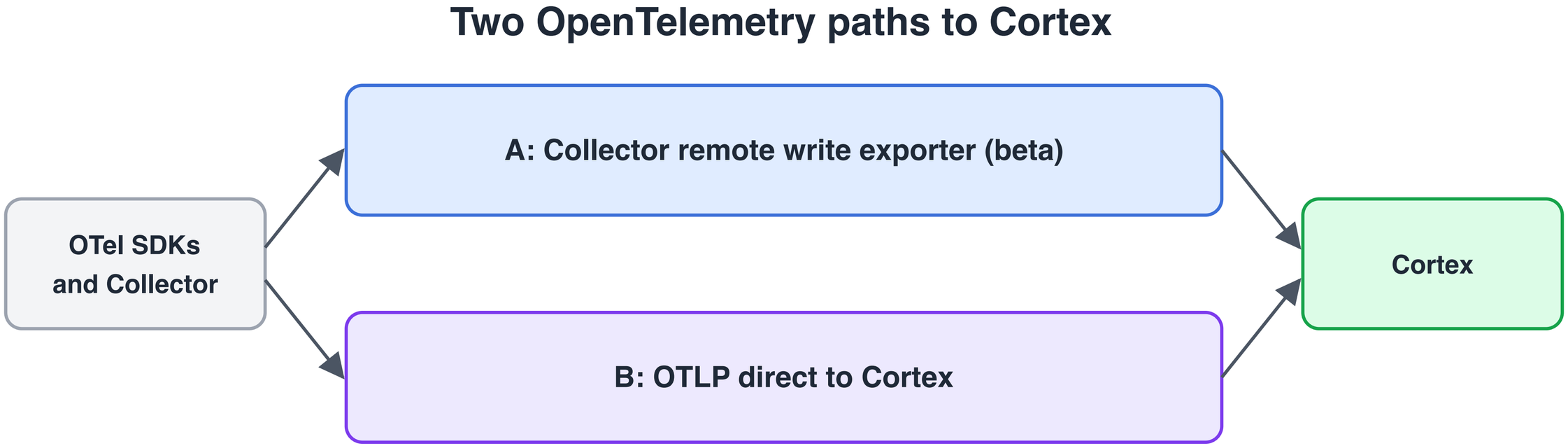
From a classic histogram to one NHCB series



Migration tip

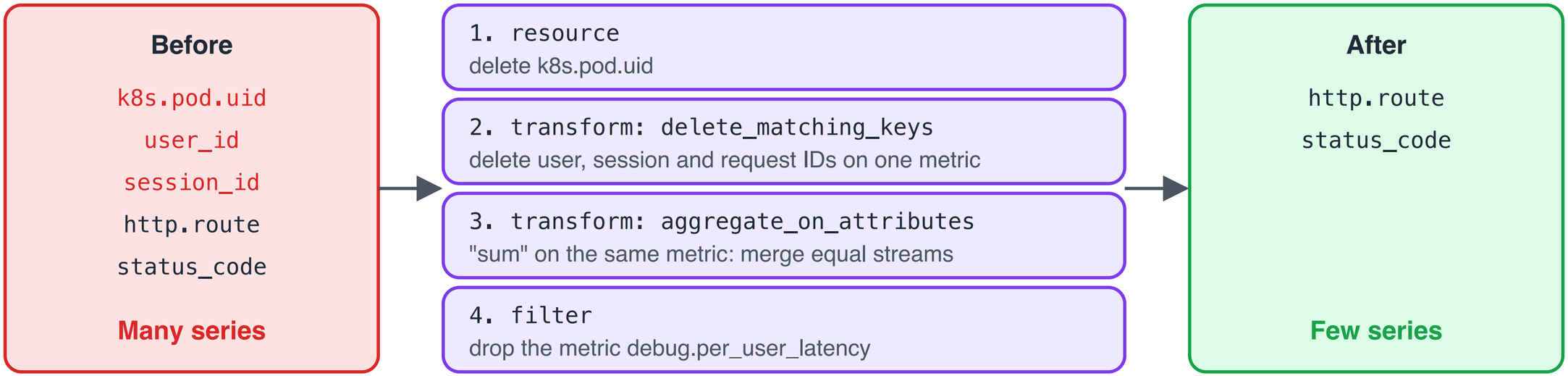
`always_scrape_classic_histograms`
1. keep both for your longest query range → 2. Move dashboards to the native form → 3. Turn it off

OpenTelemetry Collector path



Drop and aggregate labels in the Collector

Labels before and after the Collector



Caution: delete a label without re-aggregation

Label deleted, no aggregation → Many streams, same labels → Rejected, dropped, or wrong data with no error

KubeCon CloudNativeCon North America 2026

PART 4

Stop one bad query from taking down a querier



Resource monitor and throttling (experimental)

Resource monitor and querier throttling

Resource monitor

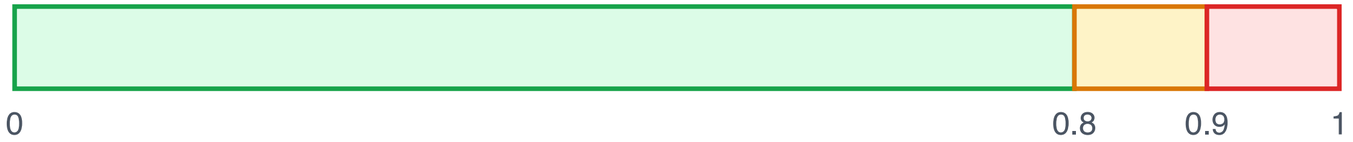
reads usage every 100ms

CPU: limit is `GOMAXPROCS`

Heap: limit is `GOMEMLIMIT`

No GOMEMLIMIT set?
Heap threshold never triggers.

Utilization (CPU or heap), 0 to 1. Example thresholds, default 0 (off)



0.8: reject new work
checks at Select, LabelNames and LabelValues

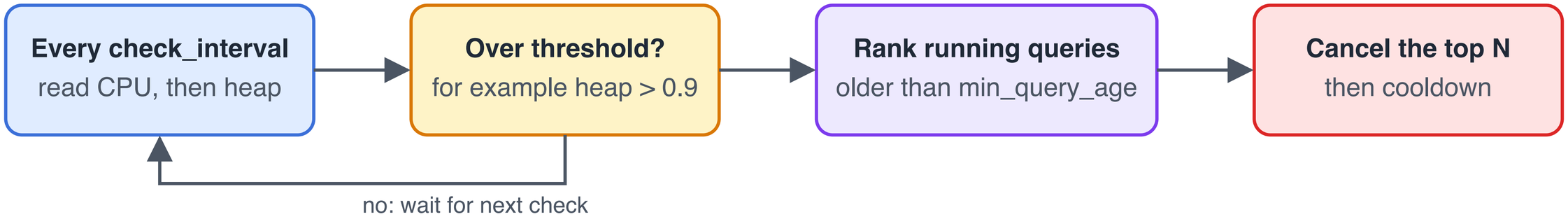
0.9: evict a query
cancel the heaviest one
(next slide)

```
cortex_querier_rejected_requests_total{reason="resource_utilization"}
```

v1.22.0 adds rejection to the querier (#7442). The ingester and store-gateway already had it.

Resource based query eviction (experimental)

Resource based query eviction (querier)



Result

One query fails with "resource limit reached".

The querier is more likely to stay up.

All other queries continue.

Metric

`cortex_query_evictions_total`

Ranked by eviction_metric (fetched_samples)

tenant-b {__name__=~".+"} 120M samples EVICT

tenant-a rate(http[5m]) 30M

tenant-c up 4M

... 1M

KubeCon CloudNativeCon North America 2026

PART 5

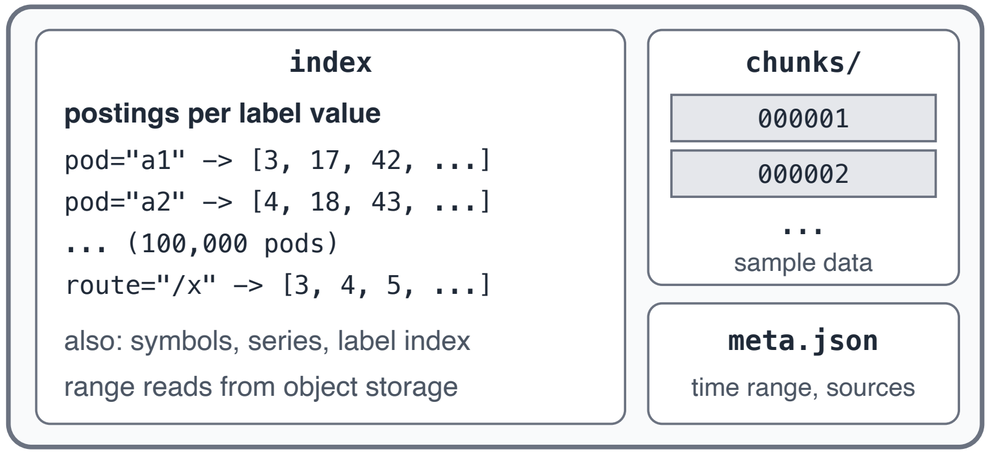
Query very large label sets with Parquet



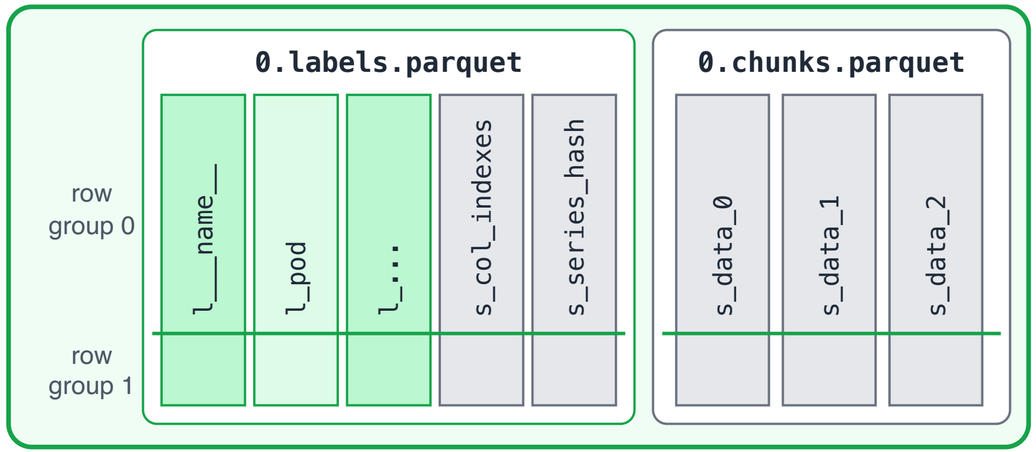
TSDB blocks and Parquet

TSDB block and sharded Parquet

TSDB block: index + chunk segment files

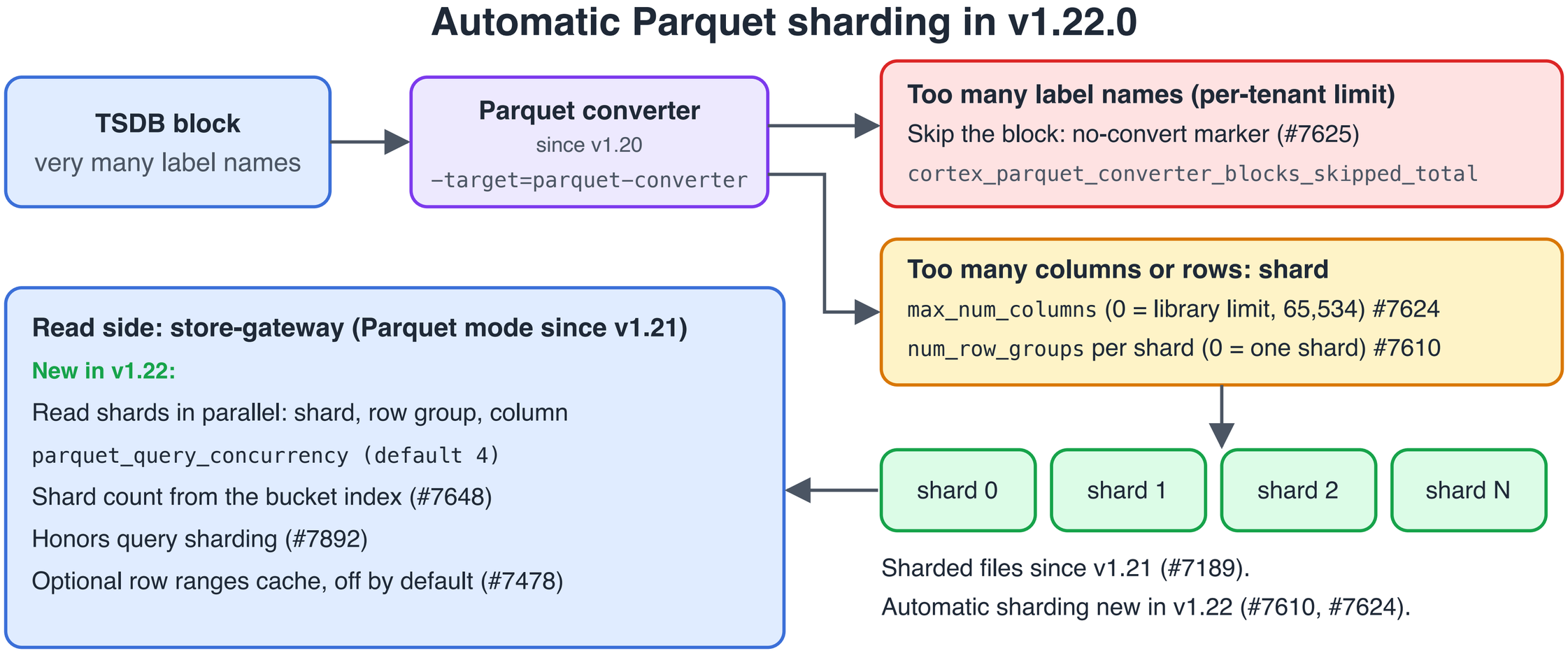


Parquet shard: one labels file + one chunks file



The converter makes the Parquet shards. The querier and the store-gateway read them.

New in v1.22.0: automatic Parquet sharding



Security audit: results

Security audit: Quarkslab, coordinated by OSTIF

Finished 2026-04-20. 7 findings.

Severity | **6 Medium** | **1 Low** | No Critical, no High

V01
Tenant impersonation on PushStream
#7475

V02
Stored XSS on status pages
#7512

V03
Secrets visible on /config
#7473

V04
Gzip decompression bomb on push, OTLP
#7515

V05
Native histogram memory amplification
#7570

V06
Unbounded reads on the gossip port
#7518

V07
Gossip payload integrity
#7474

All 7 findings fixed in v1.21.1 (2026-06-04)

V04, V05: memory amplification on the write path. A small payload grows very large in memory.

Your checklist for Cortex v1.22.0

Checklist: surviving high cardinality

Detect

- [ ] Set up all limits and alert on discards `cortex_discarded_samples_total`
- [ ] Watch head metric names and active series
- [ ] Add trackers `active_series_trackers`

Reduce at the source

- [ ] Drop unbounded labels
- [ ] Native histograms
- [ ] Convert to NHCB
- [ ] Remote Write 2.0
  NHCB, less bandwidth
- [ ] Aggregate in the OTel Collector

Protect the read path

- [ ] Set GOMEMLIMIT `resources: cpu,heap`
- [ ] Rejection and eviction thresholds
- [ ] Fetch limits per query `max_concurrent_data_bytes` (experimental)

Speed up block queries

- [ ] Parquet converter for largest tenants
- [ ] Decide if Parquet blocks should be split
- [ ] Parquet queryable or store-gateway mode

References: Cortex pull requests (v1.22.0 unless noted)

Topic	PRs	Contributors
Active series trackers	#7476	@yeya24
Head metric names gauge	#7514	@yeya24
Head queried series	#7500	@yeya24
Cardinality API	#7335 (proposal), #7384 (open)	@CharlieTLe
RW 2.0 start timestamp	#7371	@SungJin1212
RW 2.0 symbols pooling	#7398, #7401, #7392	@SungJin1212
RW 2.0 metadata dedup	#7760	@SungJin1212
Native histograms graduated	#7789	@CharlieTLe
NHCB ingestion	#7191 (v1.21)	@SungJin1212
Received histogram buckets metric	#7569	@PaurushGarg
Resource based throttling in querier	#7442	@eeldaly

Topic	PRs	Contributors
Resource based query eviction	#7488	@eeldaly
Store-gateway max concurrent data bytes	#7271	@eeldaly
Sharded Parquet	#7610 (v1.22), #7189 (v1.21)	@SungJin1212, @yeya24
Parquet max columns	#7624	@yeya24
Parquet max block label names	#7625	@siddarth2810
Parquet query concurrency	#7613	@SungJin1212
Parquet row ranges cache	#7478	@siddarth2810
Parquet shard count from bucket index	#7648	@SungJin1212
Parquet store-gateway honors query sharding	#7892	@SungJin1212
Distributed query execution	#6789 (proposal), #7018, #7019 (draft), #7688 (open)	@harry671003, @rubywtl, @eeldaly
Security audit fixes	#7473, #7474, #7475, #7512, #7515, #7518, #7570 (v1.21.1), #7587	@friedrichg, @SungJin1212, @danielblando, @PaurushGarg

KubeCon CloudNativeCon North America 2026

Q&A

- Cortex: https://cortexmetrics.io
- GitHub: https://github.com/cortexproject/cortex
- Slack: `#cortex` on the CNCF Slack

Thank you.

Cortex: Surviving High Cardinality Metrics

