Taking control of metrics growth and cardinality: Tips for maximizing your observability function

Rob Skillington, Co-Founder and CTO @ Chronosphere
About me

Rob Skillington, Co-Founder and CTO at Chronosphere
- M3 Open Source Creator
- OpenMetrics Contributor
- Twitter: @roskilli
Agenda

- Observability in a cloud-native world
- Taking control of metrics growth and cardinality
- Evaluating your observability function
- Key takeaways
Cloud-native observability

Beyond metrics + logs + traces
Our mission: help customers get to remediation as quickly as possible

01. How quickly do I get notified when something is wrong? Is it BEFORE a user/customer has a bad experience?

02. How easily and quickly can I triage it to know what the impact is?

03. How do I find the underlying cause so I can fix the problem?
Growth in monitoring data at Uber

- 1.5B datapoints/s
- 10X Cost Efficiency
- 99.99% Reliability

Metrics & Monitoring Team
- Founded 2015

Data Growth at Uber

- 1 Product in 3 Cities
- 1 Monolith
- 10s Hosts

- 10s Products in 100 Cities
- 200 Services
- 1000s VMs

- 100s Products in 600 Cities
- 4,000 Microservices
- 1,000,000s Containers
Taking control of metrics growth and cardinality
High cardinality runs wild in cloud-native environments

Virtual-machine based environment

- 10 HTTP routes
- 5 services
- 300 VMs

= 150 thousand possible unique time series

Cloud-native environment

- 10 HTTP routes
- 5 services
- 30,000 pods (10x VMs)
- 100 experiments

= 150 million possible unique time series
Scenarios for taming data growth and cardinality

Scenario 1: Tensions between too much and not enough information

Scenario 2: Cardinality of metrics is too much to manage at micro level

Scenario 3: Ownership needed beyond the Observability team – it’s a team effort!
Scenario 1:
Tensions between too much and not enough information

Tips for how to reduce these tensions:

- Remember that more data is not more better
- Create internal framework on how and which metrics will use tags or labels
- Find ways to control data flow (e.g. Rate and Query Limiters)
Scenario 2: Cardinality of metrics is too much to manage at micro level.

Tips for managing metrics at a more macro level:

- “Monitor the monitor” – Metadata dashboards for macro-level overview of your metrics
- Alert on your metrics system uptime and availability, and deep dive only when needed
- Take a programmatic approach by utilizing your platform’s aggregation functionality (e.g. roll up rules)
Scenario 3:
Ownership needed beyond the Observability team – it’s a team effort!

Tips for how to make observability a team effort:

- Set company or team wide parameters, and put onus on respective teams to stay within them
- Get buy-in from leadership and automate where possible
- Don’t build if you don’t have to!
- Encourage safe experimentation and iteration of tools and processes
Evaluating your observability function
Internal KPIs and metrics – meta metrics

- Core function of SRE and DevOps
- Initially 2 in 100, then 5 in 500 and eventually grew to 50 in 2500

- Not a lot of good benchmarks out there
- At Uber it grew to 8% of infrastructure cost at its peak, then was hyper optimized to 3%

- Are there reasonable SLO/SLIs in place and are they being met?
- Internal and external NPS
- Error rate and speed of mitigation

How many FTEs on the team?

How much should we be investing?

How do you measure success?
Key takeaways

With cardinality on the rise, your observability practice should focus on:

- How do I get notified when something is wrong?
- How easily and quickly can I triage it to know what the impact is?
- How do I find the underlying cause so I can fix the problem?

More data is not more better

Know when (and when not) to deep dive your metrics

Uplevel your function with automation, safe experimentation, and top-down support

Don’t build if you don’t have to!
Thank you