INTERCOM

Intercom brings messaging products for sales, marketing, and customer service to one platform. More than 20K businesses use Intercom to connect with a billion people worldwide.

Environment:

- Monolithic Rails app
- External services written in Go
- 100% AWS

Need:

- An observability service that allowed them to investigate problems across high-cardinality fields such as unique customer app IDs attached to relevant events and metrics.
- Support for adding and removing data from instrumentation at will without having to reconfigure dashboards, queries, alerts, etc.

Love:

“We’ve been through a few metrics products. When you try to break data down by high-cardinality fields like customer appIDs, or even host IDs, it just kind of breaks their UIs and increases cost many times because everything is stored individually.

We've been told so many times that logging customer ID was impossible and our vendors throttle us or complain, but Honeycomb just handles that.

When investigating a problem, I want to know if it’s caused by a particular customer. Being able to break down metrics on higher-cardinality fields gives very actionable insights. We never dug into it on a per-app basis because we just couldn't.”

- Jamie Osler
Product Engineer, Intercom

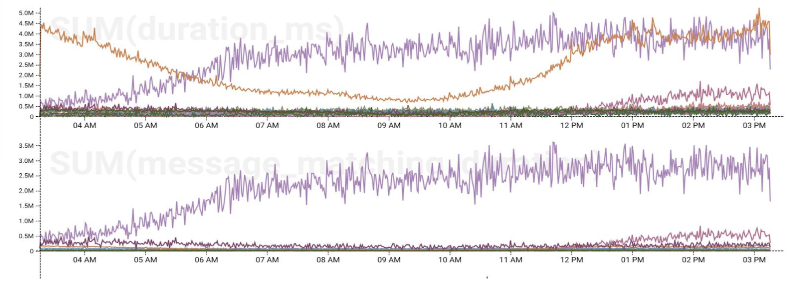
Challenge:

Intercom’s customer-centric culture drives their operations processes, but they knew they were not detecting some key error conditions and performance issues affecting individual customers. They needed to be able to pinpoint problems down to exact events and affected customers, but the tools they had been using could not handle the high-cardinality nature of their data.

For example, Intercom tracks “message matching duration”' the time it takes to check if someone should receive a message through the Messenger. Intercom’s team knew this could sometimes be a slow operation and had tried with other tools to get greater visibility into the problem. Historically, they had not been able to get insight into what/who was influencing the P50 or P99 values.

Solution:

Intercom’s first step was to add Messenger timing data to their Honeycomb dataset. This was the result of an initial query:



The top graph shows total duration of all requests across all customer `app_id_codes`. The bottom graph shows total duration of message matching across all apps. The single app in purple is taking up much more message-matching time than any other app.

Results:

“With Honeycomb, we were able to get a breakdown of message matching duration by customer and saw that one customer was a huge outlier there, accounting for 55% of all server time spent doing that operation-and over 10% of overall server CPU usage for the fleet!”

Based on what they learned from a single 15-second query in Honeycomb, Intercom is now able to plan for the future of their service. Honeycomb allows Intercom’s software owners to discover the extent of one customer’s use case, or its true impact.

Intercom also uses Honeycomb to instrument Elasticsearch services, to see how badly additional regexes in queries translate into performance, or to get insights into how well a cluster set is going to scale with a particular query load.

honeycomb.io