Cloud Observability: A MELTing Pot for Petabytes of Heterogenous Time Series

Suman Karumuri
Slack Technologies
skarumuri@slack-corp.com

Franco Solleza, Stan Zdonik
Brown University
{fsolleza,sbz}@cs.brown.edu

Nesime Tatbul
Intel Labs and MIT
tatbul@csail.mit.edu

On May 12, 2020 at 4:45pm PST, the cloud-based business communication platform Slack experienced a total service disruption [3]. To its millions of users, the outage lasted for 48 minutes; within Slack, the cascade of events that led to this outage had actually started several hours before, at 8:45am PST. A post-mortem analysis of this 8-hour incident is laid out in Slack's engineering blog in detail [5]. However, triaging the problem to identify its root cause on the spot, under time pressure was not a trivial endeavor. Engineers from multiple teams put "all hands on deck", exploring patterns and correlations across PBs of operational data that were visible to them through Slack's monitoring, dashboarding, and alerting infrastructure.

This incident is a prime example for why *observability* has been emerging as a critical capability for large-scale software systems and services that are built and operated on the highly distributed, heterogeneous, and complex setting of the cloud-native ecosystem [1]. Borrowed from control theory, the notion of observability aims at bringing better visibility into testing, debugging, and understanding the general behavior of software based on telemetry data collected from the internals of a system as it operates [6, 7]. Beyond simple black-box monitoring of known scenarios, observability requires deeper contextual information and insight about operational semantics of systems for root cause analysis of unforeseen problems. The main goal is to minimize *time to insight* - a critical measure of understanding what is happening in the system, and why.

Due to its data-intensive and time-sensitive nature, observability is essentially a data management problem. Large volumes of heterogeneous time series data should be collected, stored, and indexed from instrumented systems in a way that enables their low-latency search, analysis, and visualization in real time and on demand. Since observability is critical for meeting service-level objectives, there is a lot of industrial activity in this domain. However, current solutions are mostly ad hoc, custom-built solutions with many moving pieces [4]. We believe that these solutions will not scale well in the long term, as they incur high performance overheads, operational complexities, and infrastructure costs. There is a growing need to rethink the current design of data and software infrastructures to enable observability data management at scale.

11th Annual Conference on Innovative Data Systems Research (CIDR '21) January 10-13, 2021, Chaminade, USA.

Observability data typically consists of four time series categories that widely differ in terms of their storage, querying, and long-term retention needs as well as payload types and volumes: *Metrics (M)*, *Events (E)*, *Logs (L)*, and *Traces (T)*. Metrics provide quantitative (numeric) measurements of system performance and availability at a specific point in time. Events are highly structured strings that identify a finite set of possible occurrences. Logs are semi-structured strings that expose higher granularity event information with rich local context. Finally, Traces capture the end-to-end request flow of logical operations through various components of a distributed system [2].

Slack generates 100s of TBs of MELT data every day, which needs to be ingested, stored, and indexed to serve a variety of observability queries. For example, to identify the root cause of its outage incident, Slack had to run various types of analytical queries across all of its MELT data for the time period immediately preceding the incident. Despite their differences, there are also certain workload characteristics that are common across all MELT data: they are all immutable, append-heavy, and bursty, and queries over them have a bias to freshness (e.g., $> 95\%$ of all queries on MELT data in a Slack workload that we analyzed were on data that was less than 24 hours old). Understanding the complete requirements of heterogeneous MELT datasets and query workloads over them is key in building a scalable observability data management system.

Based on current experience in industry exemplified by Slack, this talk will discuss why today's approaches to cloud observability are all facing technical and practical challenges, and what guiding design principles should instead be followed while building data systems that can handle this emerging workload at scale [2].

REFERENCES

[1] C. Chan and B. Cooper. Debugging Incidents in Google's Distributed Systems. *ACM Queue*, 18(2):47–66, March-April 2020.

[2] S. Karumuri, F. Solleza, S. Zdonik, and N. Tatbul. Towards Observability Data Management at Scale, 2020. *Under submission*.

[3] R. Katkov. All Hands on Deck: What does Slack do when Slack goes down? https://slack.engineering/all-hands-on-deck/, 2020.

[4] S. Niedermaier, F. Koetter, A. Freymann, and S. Wagner. On Observability and Monitoring of Distributed Systems – An Industry Interview Study. In *International Conference on Service-Oriented Computing (ICSOC)*, pages 36–52, October 2019.

[5] L. Nolan. A Terrible, Horrible, No-Good, Very Bad Day at Slack. https://slack.engineering/a-terrible-horrible-no-good-very-bad-day-at-slack/, 2020.

[6] OpenTelemetry. The OpenTelemetry Observability Framework. https://opentelemetry.io/, 2019.

[7] C. Sridharan. *Distributed Systems Observability: A Guide to Building Robust Systems*. O'Reilly Media, July 2018.