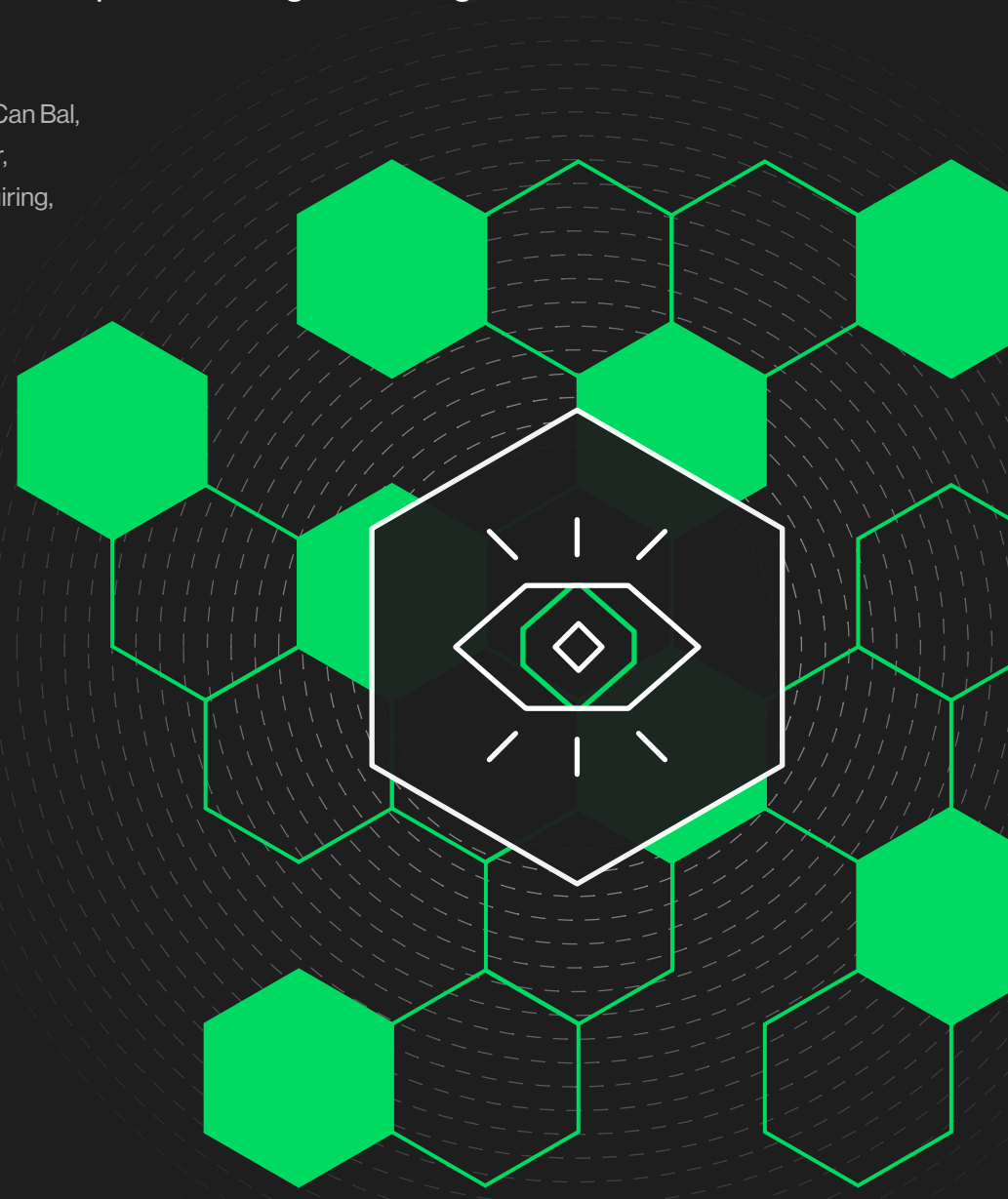
EDGE DELTA

WHITE PAPER

The Edge Delta Observability Architecture

How Edge Processing Can Help You Manage Growing Data Volumes

Ozan Unlu, Fatih Yildiz, Ohad Almog, Can Bal,
Huseyin Ozgur Batur, Taylan Isikdemir,
Matt Miller, Riley Peronto, Zachary Quiring,
Mustafa Veysi Soyvural, David Wynn



Abstract

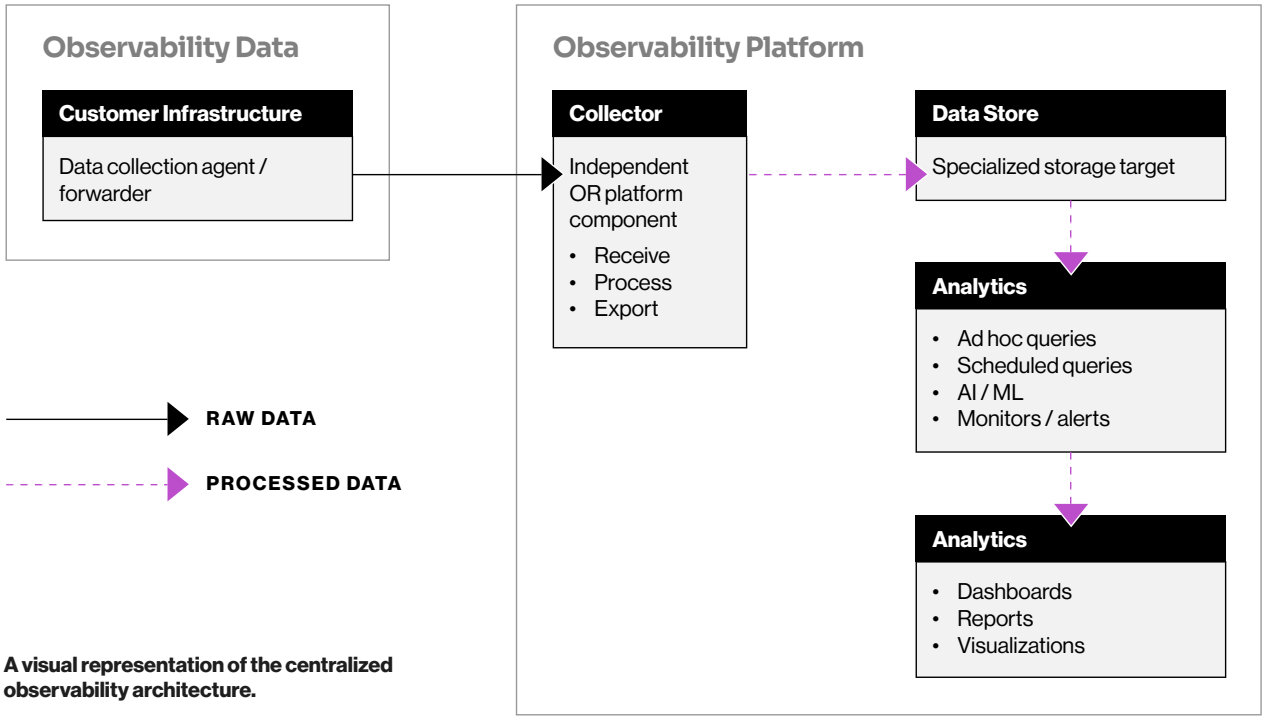
In this paper, we describe the Edge Delta observability architecture, focusing on its distributed properties. This distributed deployment computation model enables Edge Delta to horizontally scale, processing telemetry as it's created at the point of the data's creation. We will discuss the key problems this architecture addresses, namely exploding data volumes and the impact this has on cost, performance, coordination across teams, and tool sprawl. We will conclude with a discussion about opportunities for further applications.

Defining the Predecessor: Centralized Observability Processing

Since this paper focuses on the Edge Delta observability architecture, it is first important to understand the architecture it displaces. Each vendor and customer may implement slight variations of what is described below. However, the implications and pain points of this approach remain largely the same across all parties.

For the purposes of this paper, we will refer to this architecture as the "centralized observability architecture."

With this architecture, users first deploy an **agent** (or "forwarder") on their application computing resources. The agent then gathers and routes observability data downstream to a **collector**. Vendor-specific agents support a single observability platform's collector, while open source agents often have support for multiple platforms.



A visual representation of the centralized observability architecture.

There are a few common collector deployment styles, including:

- **An independent collector** that runs on dedicated hardware, serving as a "gateway" or "aggregator." It receives data from the agent, and then processes and exports the data to the observability platform.
- **A cloud component** within a larger observability platform (e.g., an "indexer") that receives, processes, and exports data.

Depending on the agent and collector, users may receive basic data processing capabilities at either level, such as data filtering, sampling, or obfuscation. The collector routes processed and unprocessed data to the observability **data store**, a specialized storage target within the larger platform (e.g., a log index). Often this data store resides in a cloud or data center that is geographically distant from the collector. This storage target is optimal for short-term retention to support analytics. However, it is often cost-prohibitive both for longer-term retention and at large data volumes.

Once the user centralizes data in the observability data store, they can run **analytics** on stored data: ad hoc queries for troubleshooting, scheduled queries to populate dashboards, anomaly detection, monitors, etc. The platform provides **visualizations** via dashboards, reports, and other views.

The Problem

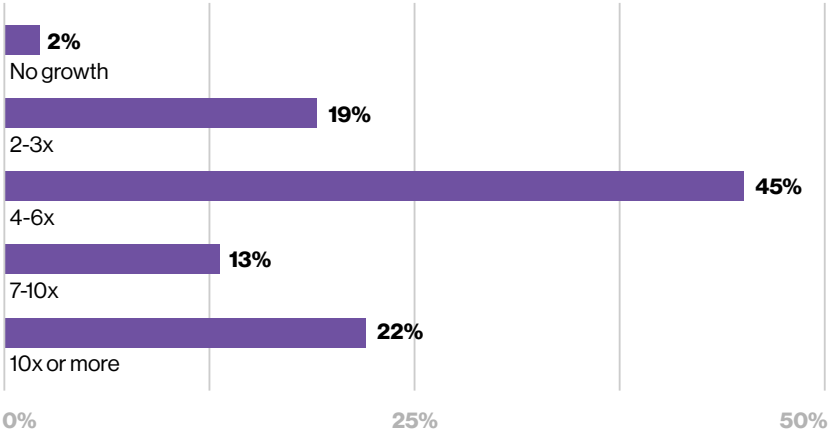
The centralized observability architecture has started to become a problem for organizations at scale for a number of reasons.

Rising Observability Data Volumes

The continued growth of data volumes in modern IT environments poses an increasing challenge for maintaining visibility into how those systems function.

Log Data Increase

Previous 3 years



Log data has grown 5x on average over the past three years.

For most organizations, this is a direct consequence of growth. If customers are served via a computing stack, that stack will need to grow to serve more customers, regardless of the growth rate.

At the same time, there has also been an effort across industries to move to microservices-oriented architectures. While bringing a number of benefits from a developer velocity and isolation boundary standpoint, this practice also increases the amount of boilerplate overhead (and subsequent observability data) that application teams generate.

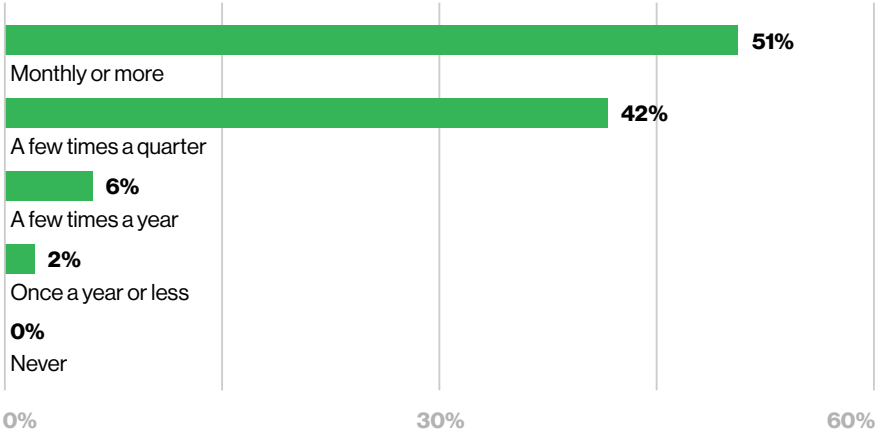
The end result is that the majority of established companies today are generating more than 1TB of log data per day alone, and most companies are growing that volume at a rate of 4x or more every 3 years[1].

The rising tide of data has directly led to a number of secondary challenges we will enumerate.

Observability Costs are Rising

The vast majority of observability products charge on the amount of data ingested into the platform (in addition to other methods). The rising cost of these tools directly reflects the reality of growing observability data volumes in the organization.

Frequency of Overages/Cost Spikes



98% experience overages or unexpected spikes in costs at least a few times a year.

If there are other price mechanisms involved in a tool, the rising data volume can trigger even more charges. If data ingestion and indexing are charged separately, if queries are charged based on the data scanned, if metrics are charged by the number of metrics in the system... rising data volumes will impact all of these costs.

[1] "Charting Observability 2023: A Voyage Into Data Growth and the ROI of Observability." Edge Delta, September 2023, edgedelta.com/charting-observability-2023.

Observability Complexity Rises to Meet Cost Burdens

Organizations have taken on increasing complexity in their implementations, processes, and standards to address growing observability costs. The increased complexity has impacted the following areas:

Performance

The most immediate impact is on the performance of the tools themselves. By definition, searching across more data slows down query times, so most organizations implement some form of partitioning in order to improve performance again. Others create an observability ingestion budget for application teams to adhere to (with varying degrees of enforcement).

In either case, the querying becomes more complex, or additional planning and risk are taken on by platform and application teams in order to keep the observability tool performing in a usable manner.

This is to say nothing of the fact that the half-life of the value of most observability data is relatively short, and if rising data volumes can't be processed quickly, that value to the organization is simply lost... or translated into other outcomes such as avoidable downtime.

Coordination

Following from the above, initiatives that coordinate across teams involve heavy coordination burdens. Teams that own the observability platform often take on roles as informal project managers to continually check in with application teams and keep their observability data volumes in check. However, with the current growth rate of observability data, even effective cost-cutting initiatives can have negative repercussions. If an organization were able to cut its data in half through high-effort mechanisms, that exercise now becomes a core part of the team's function as observability data continue to grow year over year. If there isn't a dedicated team to observability, as is frequently the case, then that means even less time to drive progress on other responsibilities.

Additionally, observability data itself comes from a mix of sources and each message has widely variable value in different contexts. The data includes stack traces that are vital to debugging issues, contextual information that can be useful in root cause analysis from time to time, and information of very low value such as health checks and low-level networking messages that provide extremely low insight per byte. Handling these data types differently, as they should be, requires an ongoing understanding of what they are, who uses them, and what manners of use are most economical.

Tooling

Finally, there is additional tooling complexity. Allowing for distributed decision-making among application teams can lead to a mix of polyglot environments, each with different observability standards and needs. If given enough authority, teams might even choose to spin up their own observability tooling just for their use cases, which can lead to multiple different vendors or open-source deployments at a single organization.

Another layer organizations will add if the data volume grows too large is a message queue like Kafka, giving a publish-subscribe (Pub/Sub) level of control over where the data goes and opening up the opportunity for some transformations, at the price of an additional deployment to run and maintain for observability data.

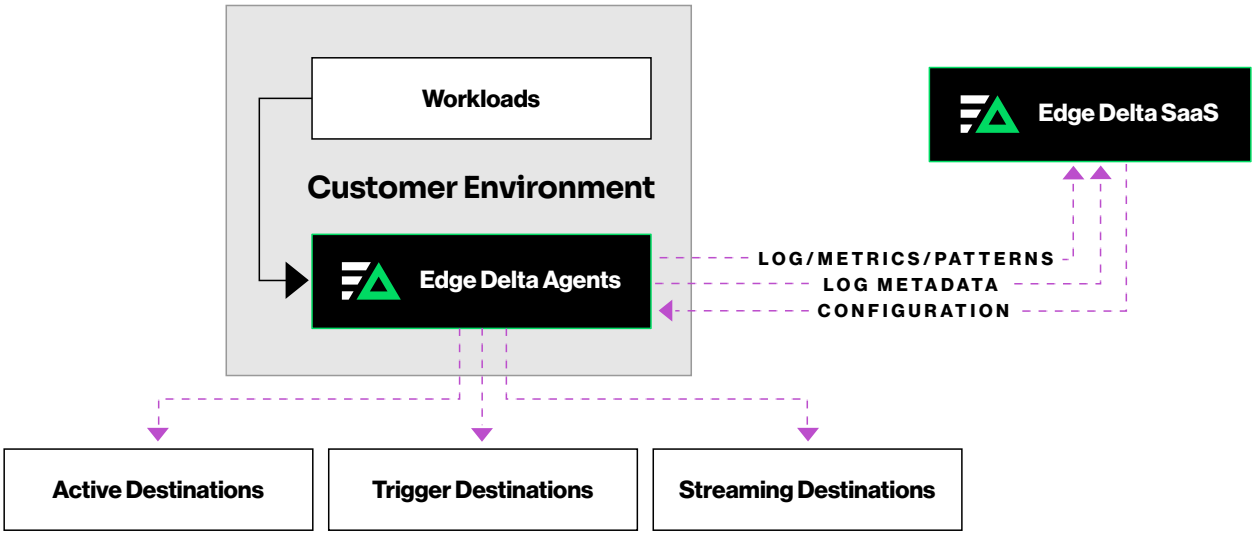
The Results

All this uncertainty and complexity leaves organizations with a number of initiatives, many of which struggle to deliver their value quickly enough before the next renewal comes about. The Edge Delta architecture addresses these challenges directly.

The Edge Delta Architecture

Overview

In this section, we will define the Edge Delta architecture and juxtapose it with the centralized approach. The Edge Delta architecture is made up of two different components: the agent and the SaaS backend.



The Edge Delta Agent

The Edge Delta agent is deployed on the customer's computing resources (e.g., server, virtual machine, Kubernetes cluster, etc.). It runs as an independent collector, though it also supports being co-deployed with other tools. Unlike other tools, the Edge Delta agent is intensely optimized for performance, enabling all the traditional functions of shipping and forwarding, while also enabling distributed processing and analytics. This allows for the data to be processed more quickly, in the context of its original environment, in a horizontally scalable manner.

CPU Utilization % by Events Per Second on AWS EC2 c5d.4xlarge

Events per second	Edge Delta	Splunk UF	Splunk HF	FluentBit	FluentD	Vector	Filebeat	Logstash	OpenTelemetry
1000	**2%**	4%	1%	1%	3%	3%	4%	34%	3%
5000	**3%**	9%	1%	2%	4%	5%	18%	39%	7%
10000	**4%**	9%	FAILED	2%	4%	5%	37%	63%	10%
20000	**5%**	29%	FAILED	4%	4%	5%	52%	69%	25%
50000	**8%**	FAILED	FAILED	10%	12%	8%	57%	92%	32%
500000	**62%**	FAILED	FAILED	79%	FAILED	83%	FAILED	FAILED	FAILED

Memory Utilization % by Events Per Second on AWS EC2 c5d.4xlarge

Events per second	Edge Delta	Splunk UF	Splunk HF	FluentBit	FluentD	Vector	Filebeat	Logstash	OpenTelemetry
1000	**78MB**	125MB	125MB	14MB	72MB	167MB	123MB	1.12GB	174MB
5000	**78MB**	125MB	125MB	14MB	74MB	145MB	132MB	1.12GB	174MB
10000	**78MB**	126MB	FAILED	14MB	76MB	152MB	132MB	1.15GB	174MB
20000	**79MB**	126MB	FAILED	14MB	76MB	158MB	145MB	1.2GB	174MB
50000	**80MB**	FAILED	FAILED	14MB	76MB	167MB	149MB	1.2GB	174MB
500000	**80MB**	FAILED	FAILED	14MB	FAILED	176MB	FAILED	FAILED	FAILED

The Edge Delta agent is optimized for performance, enabling all the traditional functions of data collection agents, as well as distributed processing and analytics.

Here are a few examples of the capabilities applied in a distributed fashion:

- Summarizing similar and repetitive logs into pattern groupings
- Converting raw logs to metrics
- Detecting known and unknown anomalies via machine learning
- Extracting message components
- Enriching data items for easier UI navigation downstream
- Distilling known metrics libraries into the data users access or alert on

If and when other vendors support any of the capabilities listed above, they do so downstream, after data has been ingested into their platform and paid for. Edge Delta pushes all these functions upstream to the data source, enabling users to populate downstream platforms with optimized datasets.

The Edge Delta agent supports routing processed data to different destinations, be they dev/null, other observability tools, or simple storage locations. Message queues are also supported, but in many cases, those queues can be abandoned by making use of the agent's processing capabilities itself, thus cutting out extra network hops and tech burden.

The Agent Properties

- Written in Go
- Supports multiple OS and processor architectures
- Self-contained binary for every platform
- Configured through the SaaS Backend by default, but supports local-only configuration

The agent also supports "trigger destinations," meaning Edge Delta can trigger alerts upstream before data leaves the customer environment, and route directly to the alerting platform.

The Edge Delta SaaS Backend

The SaaS backend provides a centralized fleet management service, creating the ability to deploy agents everywhere, but updating their configurations in a single place. These changes are tracked, so that users can see the full change history to help trace any unexpected results. Agent configurations can be deployed widely or only to groups of similar machines, easing the burden of updating entire clusters at once.

The Edge Delta SaaS backend can be used as a core observability platform as well, though that functionality is out of scope for this paper.

Challenges Addressed

Horizontally Scaling Processing with Data Volumes

At the core of this architectural shift is aligning the processing of data to the distributed nature of creating data. After all, there are both practical and theoretical limits as to what can be achieved by processing data centrally if it's created in a distributed fashion. At scale, the efficiency gains of processing the data in place outweigh the costs and bottlenecks of moving it all before analyzing it.

Enables Multiple Cost Control Models

The alignment above enables addressing the fundamentals of the observability data growth problem head-on. Not only can the appropriate data simply be routed to the most cost-appropriate destination, but raw data itself can be transformed into the appropriate shape and even aggregated for use in downstream systems. This significantly reduces the cost of downstream observability solutions.

Sample Cost Savings Enabled by the Edge Delta Observability Architecture

Daily Ingest (GB)	Original Observability Price per GB	Edge Delta Price per GB	Amazon S3 Price per GB	% Data Optimization	Total	Total Savings
100	$2	0.12	0.03	25%	$157,440	**-$47,360**
100	$2	0.12	0.03	50%	$110,080	**-$94,720**
100	$2	0.12	0.03	75%	$62,720	**-$142,080**
100	$2	0.12	0.03	90%	$34,304	**-$170,496**

Processing data upstream and routing to the optimal storage destination enables large TCO reduction.

Reduces Complexity by Using SaaS as Needed

This cost reduction comes with complexity reduction as well, which is accomplished by standardizing the collection layer for gathering observability data and easing the configuration administration. This is implemented without adding additional message queue architectures to maintain and support.

Further Discussion

Hybrid Storage

As computing power becomes more distributed and handles the majority of atomic cases very well, use cases do arise where tapping into a distributed storage system becomes necessary. This could be due to regulatory reasons, or simply to guarantee certain data locality goals. We're actively looking into expanding this architecture to include a hybrid storage model to address such cases.

Improving Tracing Through Distributed Analysis

A common complaint heard about trace spans at the time of this paper's publishing is that there are simply too many. This has led to a push for sampling in this realm, which is a very tricky thing to establish via static rules. With a bit more distributed processing power, and the ability to cache events for short periods of time and potentially recall them all at once rather than analyzing individual events in a vacuum, there becomes an interesting possibility to solve this problem in a different way. We're currently looking into ways the Edge Delta architecture could support enhanced tracing use cases.

Hot and Cold Path Processing

One of the design features of the Edge Delta architecture is that the agents mostly coordinate independently. This cuts down on coordination overhead, cuts out time sync issues, and avoids several other problems that are often inherent in distributed systems. However, as the need for more advanced processing within the isolation boundary occurs, there is an opportunity to put data on a cold path for more in-depth analysis, should the need arise. Unlike the Agents that live on the machines serving production workloads, particular events of interest could be passed off to other machines to perform some correlations, or other dynamic enrichment from elsewhere in the environment. These ideas are currently untested, but of keen interest to us.

Conclusion

In conclusion, the Edge Delta Architecture represents a much needed shift in addressing the fundamental challenges of the Observability landscape by aligning horizontally scaling data growth with horizontally scaling data processing. Its development is ongoing, but the results we've seen from customers thus far suggest that its viability today is strong, with only more promise in the future.