Project Analysis

Lightstep Distributed Tracer

Introduction

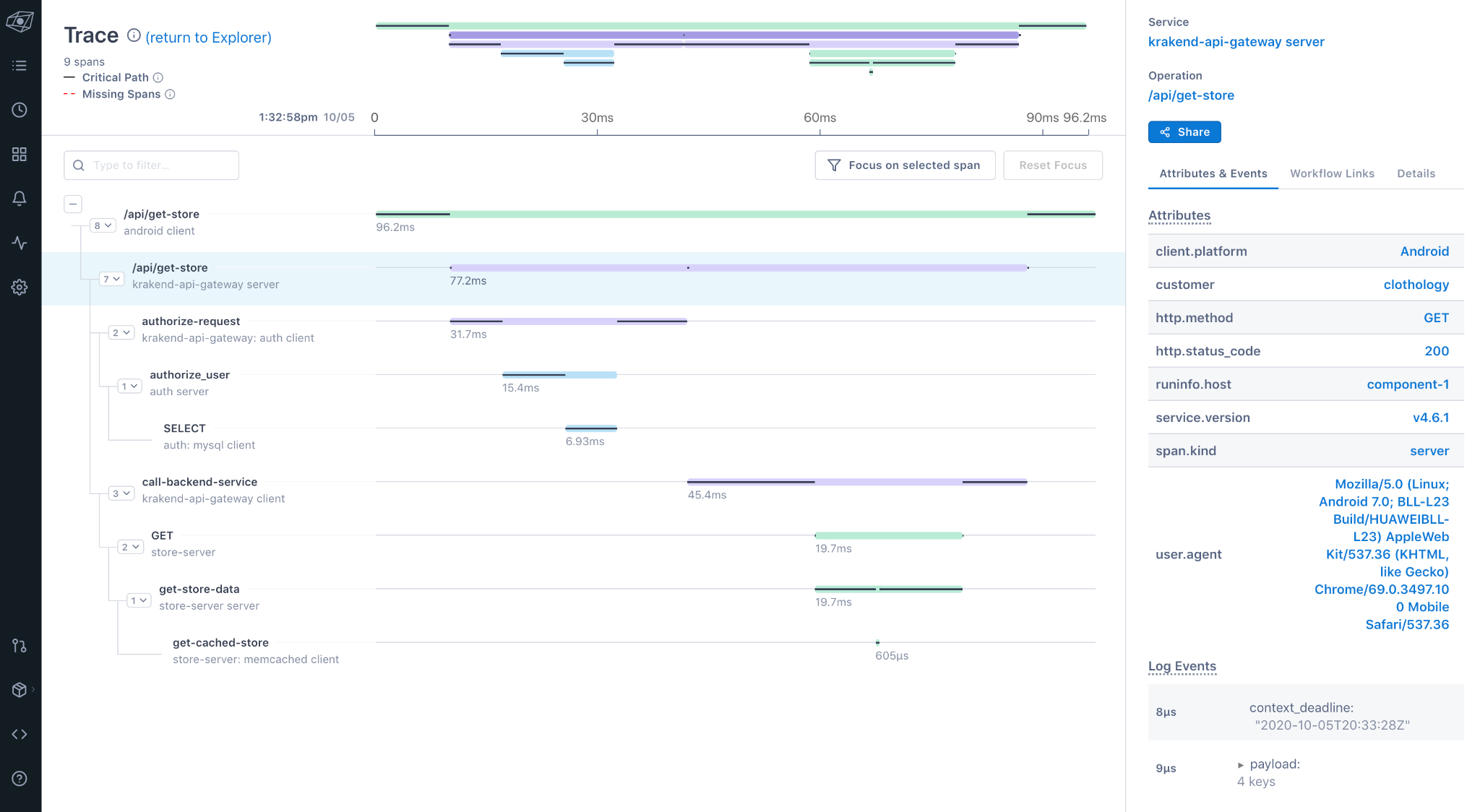
Lightstep is a cloud-native reliability platform. It collects and analyzes data across infrastructure, applications, runtime, cloud, and other third-party services. This reliability platform is the easiest way for developers and SREs to monitor system health, understand changes, respond to incidents, and accelerate release velocity in cloud-native applications.

This project analysis is on the Lightstep distributed tracer for Go. Distributed tracing provides complete visibility into a system, whether for migrating to microservices or for a complex, multi-cloud environment.

Modern software architectures built on microservices and serverless introduce advantages to application development, but there’s also the cost of reduced visibility. Metrics and logs can only give a limited amount of information- they let you know there’s an issue, but they can’t always tell you where and when. That’s where distributed tracing comes in.

With distributed tracing implemented, one has a window into performance at every step in the request. It provides end-to-end visibility and reveals service dependencies – showing how the services respond to each other. This provides visibility into exactly how separate services and parts of an application interact with each other.

By being able to visualize transactions in their entirety, it becomes possible to compare anomalous traces against performant ones to see the differences in behavior, structure, and timing. This information allows you to better understand the culprit in the observed symptoms and jump to the performance bottlenecks in your systems.Once a symptom has been observed, distributed tracing can help identify and validate hypotheses about what has caused this change.



*Lightstep dashboard-distributed tracing*

Project Summary

| Website | <https://lightstep.com/> |
| --- | --- |
| Organization/Foundation Name | LightStep which was acquired by ServiceNow on May 10, 2021 |
| License | The MIT License |
| Open/Proprietary | Open-source |
| Source Path(if open source) | <https://github.com/lightstep/lightstep-tracer-go.git> |
| Brief Description | Distributed tracing provides a view of the life of a request as it travels across multiple hosts and services communicating over various protocols. It’s a diagnostic technique that reveals how a set of services coordinate to handle individual user requests. A single trace typically shows the activity for an individual transaction or request within the application being monitored, from the browser or mobile device down through to the database and back. In aggregate, a collection of traces can show which backend service or database is having the biggest impact on performance as it affects your users’ experiences. |

Project Details

Key Features

* Easy to set up and use despite the complexity of the system

### Unlimited cardinality; pinpoint exactly what causes issues across your stack.

### Instant insights; understand why performance issues occur.

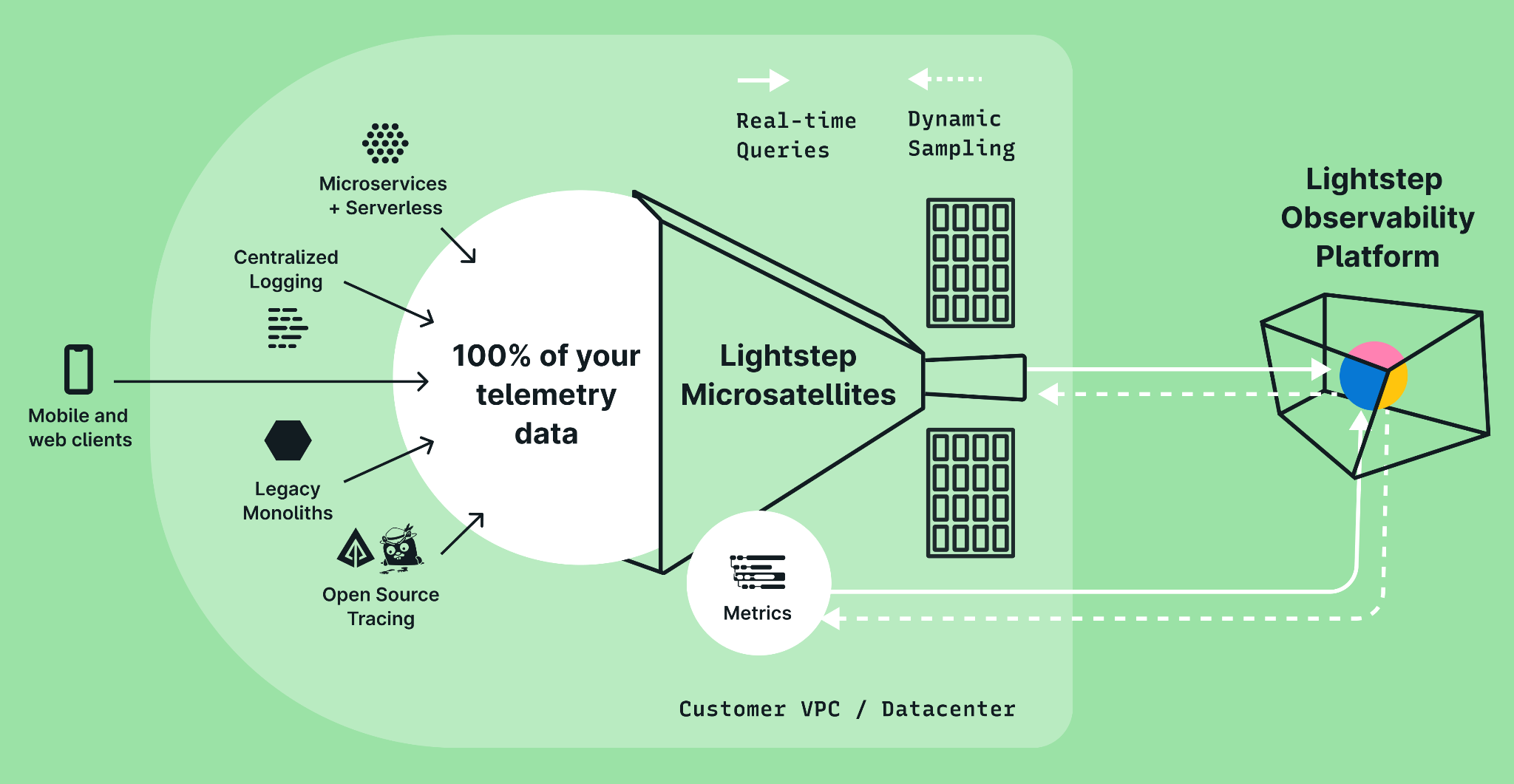
### More context; automatically surface the relevant logs and metrics for incident resolution

### Full context problem-solving

### Powered by open source

* Move seamlessly from a high-level view of dependencies to specific services, operations, traces, or any other signals contributing to issues in production
* Provides full-context root cause analysis with exact logs, metrics, and traces to simplify and solve complex investigations
* Auto-instrumentation libraries powered by OpenTelemetry
* Provides end-to-end visibility and reveals service dependencies – showing how the services respond to each other which allows visualizations of transactions in their entirety so one can compare anomalous traces against performant ones to see the differences in behavior, structure, and timing.

Architecture



Lightstep’s data collection architecture allows collection and analysis of the large volume of production data that their enterprise customers generate.

Distributed tracing relies on instrumentation of the system you’re trying to observe. You can use specifications such as [OpenTelemetry](https://opentelemetry.io/) to provide a consistent interface across a variety of languages to write this instrumentation code. Once you’ve done your instrumentation, you instantiate tracers that know how to create the spans and their associated attributes, events, and context. That [instrumentation](https://docs.lightstep.com/otel/quick-start-instrumentation) collects 100% of that data and sends it to the Lightstep [Microsatellites](https://docs.lightstep.com/docs/learn-about-micro-satellites). [Microsatellites](https://docs.lightstep.com/docs/learn-about-micro-satellites) are Lightstep components that communicate with the instrumentation to collect 100% of the telemetry data. The Microsatellites then send any data that serves as examples of application errors, high latency, or other interesting events in real time to the Lightstep Observability SaaS platform, who pieces together the spans into traces.

Key terms of distributed tracing include:

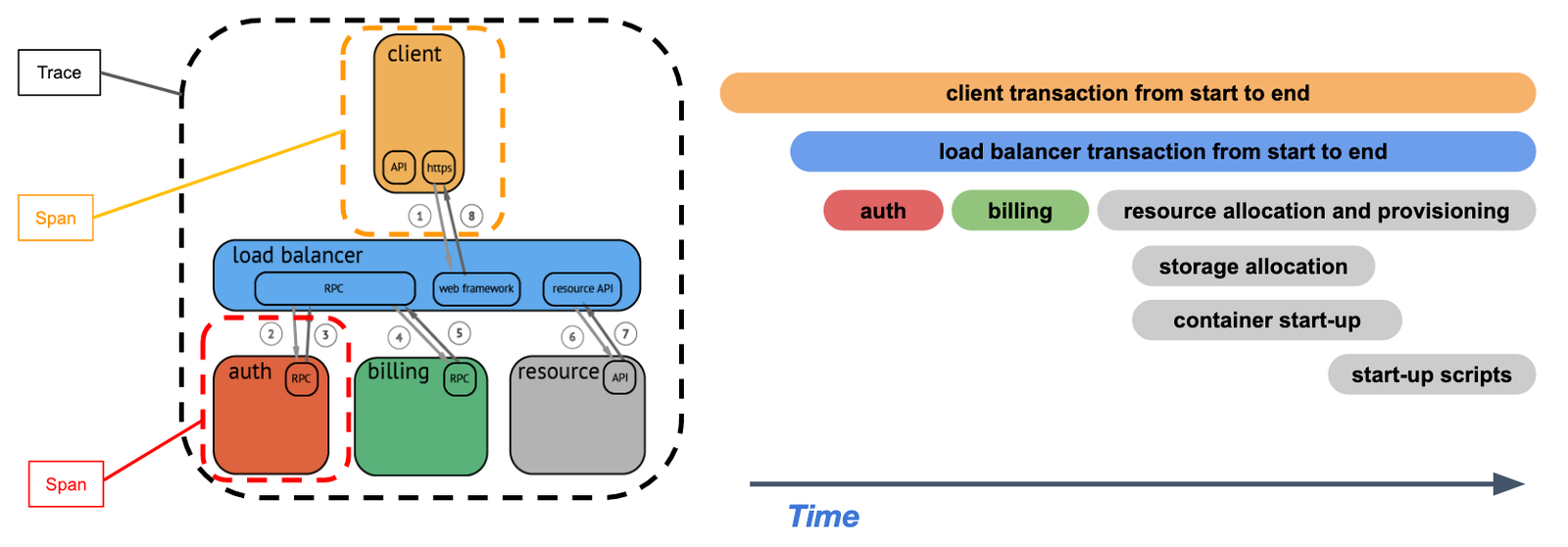
**Trace**: represents an end-to-end request; made up of single or multiple spans

**Span**: represents work done by a single-service with time intervals and associated metadata; the building blocks of a trace

**Tags**: metadata to help contextualize a span

The point of traces is to provide a request-centric view.

In distributed tracing, a single trace contains a series of tagged time intervals called spans. A span can be thought of as a single unit of work. Spans have a start and end time, and optionally may include other metadata like logs or tags that can help classify “what happened.” Spans have relationships between one another, including parent-child relationships, which are used to show the specific path a particular transaction takes through the numerous services or components that make up the application. A span may also have zero or more key/value [attributes](https://opentelemetry.lightstep.com/core-concepts/attributes-and-labels). Attributes allow you to create metadata about the span.



*simplified view of a trace, as it relates to the request*

Current Usage

**Github**

GitHub uses Lightstep daily to investigate latency, test hypotheses, and rapidly resolve performance issues across their monolithic and microservice architecture. In combination with OpenTelemetry, GitHub is able to ensure that millions of developers have a seamless experience with their platform, as well as increase the productivity of their internal developer teams.

**Plaid**

As an engineering organization building a polyglot architecture (primarily Go, Node.js, and Python), Plaid found a distributed tracing solution with high-quality client libraries that could drive their observability, monitoring, and alerting initiatives with Lightstep, which enables the team to spend less time debugging code — and more time releasing features to their customers.

**FactSet**

Their complex and highly-trafficked system includes hundreds of services, thousands of instances, and multiple languages that ultimately send Lightstep nearly five million spans/minute. Lightstep provides FactSet with full visibility into their deep system, enabling their developers to resolve incidents faster, understand dependencies, and ensure better customer experience.

**Lyft**

Lyft is a ridesharing company and its consumer mobile app has real-time transactions totaling more than one million rides per day, so performance is critically important. Lightstep Observability has played a critical part in helping Lyft minimize downtime and ensure that rider requests are quick and optimally routed.

**Cognite**

Cognite offers industrial software that unites disparate, siloed data under a single operational context. Their Industrial IoT data platform, Cognite Data Fusion, creates new opportunities for machine learning and value creation in heavy-asset industries. Lightstep enables Cognite to resolve issues faster, proactively optimize system performance, and gain complete visibility into their many service dependencies.

Technical Details

Lightstep distributed tracing allows moving seamlessly from a high-level view of dependencies to specific services, operations, traces, or any other signals contributing to issues in production. It provides full-context root cause analysis with exact logs, metrics, and traces to simplify and solve complex investigations.

The auto-instrumentation libraries are powered by OpenTelemetry. OpenTelemetry provides APIs, libraries and instrumentation resources to capture telemetry data from your applications. Any supported frameworks, protocols, libraries, and data stores are automatically instrumented with just one line of code. By providing a standardized data format for distributed traces and metrics data, OpenTelemetry eliminates the need for vendor-specific integrations.

Lightstep Observability collects and analyzes telemetry data across infrastructure, application, runtime, cloud and other third-party services. By mapping metrics to trace data from a distributed system, it can correlate root causes across traces, metrics, and logs anywhere in the system, and provide immediate insights for developers and SREs. Distributed traces on their own are just analytical data, much like raw time-series metrics or log files. So even if the right traces are captured, solutions must provide valuable insights about these traces to put them in the right context for the issues being investigated.

Project comparison

**Honeycomb**

Honeycomb, like Lightstep, also provides an easy-to-use distributed tracing solution. The Honeycomb distributed tracing tool can quickly diagnose bottlenecks and optimize performance with a waterfall view to understand how your system is processing service requests, do full-text search over trace spans and toggle to collapse and expand sections of trace waterfalls and provides Honeycomb allows for automatic definition of key pieces of trace data.

**SigNoz**

SigNoz is a full-stack open-source APM and observability tool. It captures both metrics and traces with log management currently in the product roadmap. Logs, metrics, and traces are considered to be the three pillars of observability in modern-day distributed systems. SigNoz provides a unified UI for metrics and traces so that there is no need to switch between different tools like Jaeger and Prometheus.

**Jaeger**

Jaeger is an open-source APM tool developed at Uber, later donated to Cloud Native Computing Foundation(CNCF). Inspired by Google's Dapper, Jaeger is a distributed tracing system. It is used for monitoring and troubleshooting microservices-based distributed systems. Some of its key features include:

**Zipkin**

Zipkin is an open-source APM tool used for distributed tracing. Zipkin captures timing data need to troubleshoot latency problems in service architectures. Zipikin was initially developed at Twitter and drew inspiration from Google's Dapper. Unique identifiers called Trace ID are attached to each request which then identifies that request across services.

**Dynatrace**

Dynatrace is an extensive SaaS enterprise tool targeting a broad spectrum of monitoring needs of large-scale enterprises. For distributed tracing, it provides a technology called Purepath, which combines distributed tracing with code-level insights. When a user initiates a transaction with the application, PurePath gives the transaction a unique ID.

Any other information

A distributed tracing solution is absolutely crucial for understanding the factors that affect application latency. However, modern applications are developed using different programming languages and frameworks, and they must support a wide range of mobile and web clients. To effectively measure latency, distributed tracing solutions need to follow concurrent and asynchronous calls from end-user web and mobile clients all the way down to servers and back, through microservices and serverless functions.

Reference / Acknowledgements

<https://lightstep.com/>: Official website for the Lightstep platform

<https://github.com/lightstep/lightstep-tracer-go>: Github link for Lightstep distributed tracer

<https://docs.lightstep.com/docs/how-lightstep-works>: Description of working

<https://signoz.io/blog/distributed-tracing-tools/>: Project comparison