
KubeCon
CloudNativeCon
North America 2024

KubeCon

CloudNativeCon

North America 2024

Cognitive and Self-Adaptive System for Effective Distributed-Tracing (Using Jaeger, Open Tracing)

Presenters - Akash Gusain & Mitul Tandon

Observability in Software Development

In a world where software complexity is the new norm, observability is our superpower, turning chaos into clarity

What is Observability?

- Provides end-to-end visibility into complex software environments.
- Enables teams to diagnose issues and enhance performance.

Why is Observability Important?

- Helps solve problems faster.
- Supports better decision-making and smarter workflows.
- Improves overall customer experiences.

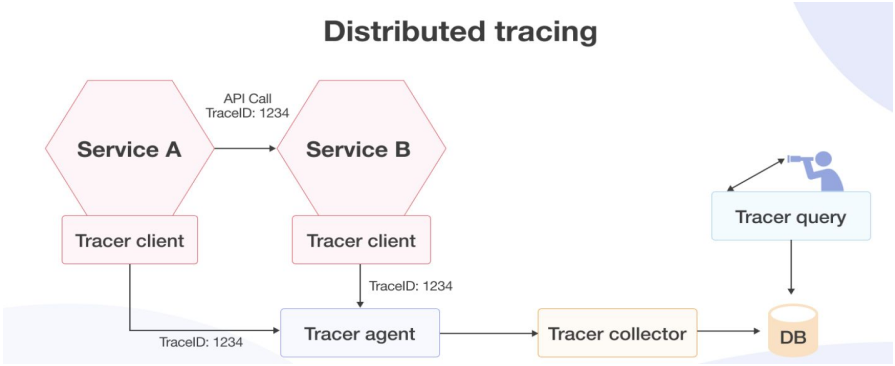
Key Types of Observability Data

- Metrics
- Events
- Logs
- **Traces**

The Role of Tracing in Observability

What are Distributed Traces?

Distributed tracing is a technique that tracks the journey of requests as they flow through microservices, providing a detailed view of interactions and performance.



Why Do We Need Tracing?

- Essential for understanding performance in cloud-based and microservices architectures.
- Provides insights into how requests are processed, helping to quickly identify bottlenecks and errors.
- Enhances the ability to maintain and optimize complex distributed applications.

Tracing the History of Distributed Tracing

2010: *Dapper* - Google releases a paper on Dapper, its internal distributed tracing infrastructure, emphasizing its value for developer and operations teams.

2012: *Zipkin* - Inspired by Dapper, Twitter launches Zipkin as an open-source tool for distributed tracing to address latency challenges.

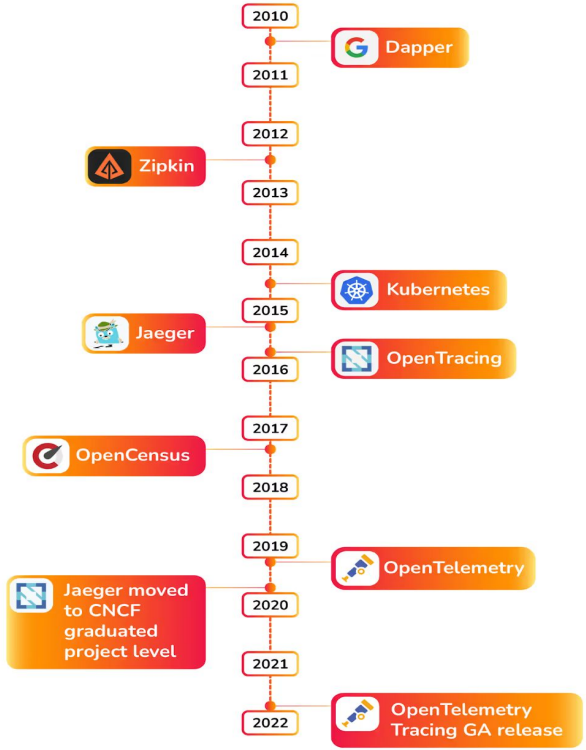
2014: *Kubernetes* - Although not a tracing tool, Kubernetes transforms cloud computing and accelerates the growth of cloud-native projects, making tracing essential for distributed systems.

2015: *Jaeger* - Uber introduces Jaeger, an open-source distributed tracing system for monitoring and profiling microservices, later accepted as a hosted project by CNCF.

2015: *OpenTracing* - CNCF welcomes OpenTracing, aimed at providing consistent, vendor-agnostic APIs for managing loosely-coupled microservices.

2017: *OpenCensus* - Google unveils OpenCensus, a library set for gathering metrics and distributed traces, allowing real-time data analysis for application health.

2019: *OpenTelemetry* - OpenCensus and OpenTracing combine to create OpenTelemetry, offering a unified framework for distributed tracing, with V1.0.0 launched in 2021 to ensure stability.



Working of Dapper

Key Terms:

- **Span**: A unit of work within a trace that records details like timestamps, IDs, and annotations.
- **Trace**: The path of a single request through a distributed system, capturing its operations and interactions.
- **Sampling -** A technique used to collect a subset of traces to reduce overhead and manage data volume, while still providing insights into performance.
- **Latency -** Time it takes trace data to propagate from instrumented application binaries to the central repository.

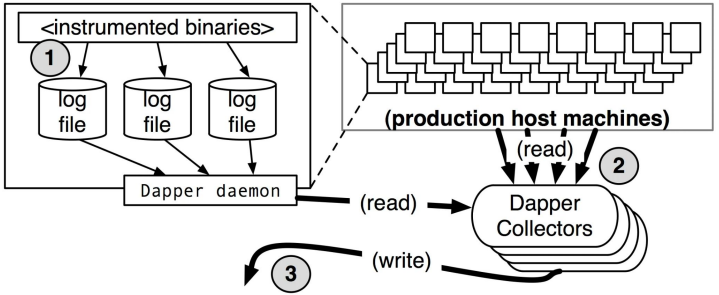
.

Dapper Sampling Strategy

Purpose of Sampling: Control overhead and manage trace data size in high-throughput environments.

Points considered:

- Head based sampling approach.
- Aggressive sampling frequency.
- Balance trace collection with performance management.
- Control size of trace data.



trace id	span 12	span 23	span 34	span 45	span 56	...
123456	*nil*	*nil*	**<data>**	**<data>**	*nil*	...
246802	**<data>**	*nil*	*nil*	*nil*	**<data>**	...
357913	*nil*	**<data>**	*nil*	*nil*	*nil*	...
...	...	...	...	...	...	...

(Central Bigtable repository for trace data)

Current Problems

- DAPPER based tracing implementations such as Zipkin and Jaeger implement a trace-sampling strategy which randomly samples only 1-5% of the traces.
- Distribution is heavily skewed towards the normal/consistent execution-traces missing out on “unusual/interesting” execution-traces.
- “Head-Based” sampling policies that adopt random sampling decision at the start of the execution flow.
- Storage overhead if sampling rate is increased.
  - *Increase Sampling Rate -> Need more storage*
  - *Lower Sampling Rate -> Lose out on capturing Error traces*
- From our analysis, over 98.1% of the saved traces returned an HTTP status code of 2xx (OK). The remaining 18 status codes collectively accounted for just 1.9% of the traced executions significantly underrepresenting the “unusual traces”.

Cognitive and Self-adaptive Solution

A “tail-based sampling” approach which makes it feasible for the tracing system to use intelligent filtering techniques in order to persist interesting and meaningful traces.

- **100% sampling rate**
  Microservices publishes all the span events to Kafka. This is done asynchronously to minimize performance overhead.

- **Span-aggregator**
  Obtains spans from kafka and stitches all matching spans to re-generate the execution trace.

- **Adaptive-Sampler**
  Each trace is assigned a specific sampling rates using the adaptive sampler engine.
  Receives the sampling rate from the Trace-Clusterer and decides whether to persist the trace in the Trace-Storage Server based on this rate.

Contd ..

Cognitive and Self-adaptive Solution

➔ **Trace-Clusterer**

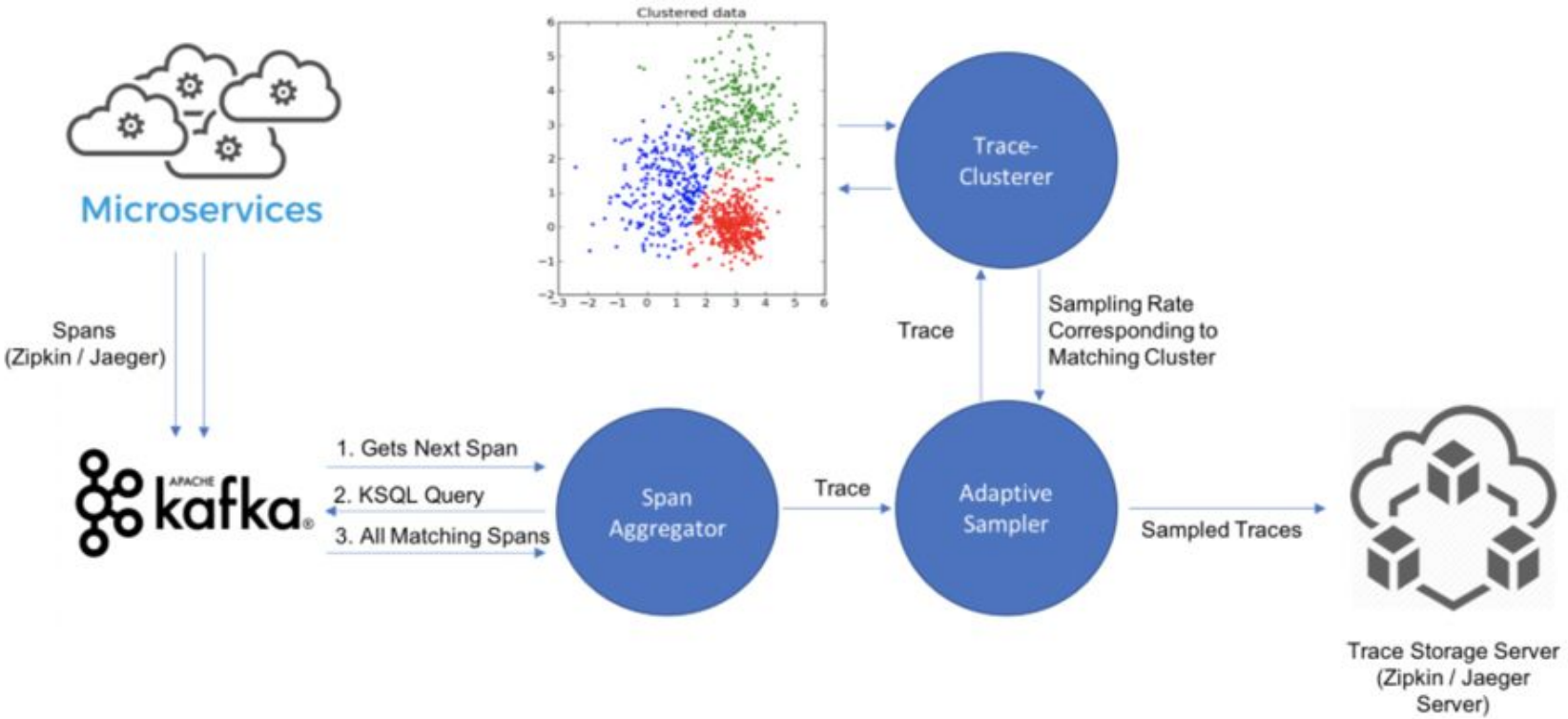
Utilizes a density-based clustering algorithm (DBSCAN) to cluster incoming traces in real-time. It groups and categorizes traces while calculating a Sampling-Rate for each cluster. This rate helps ensure meaningful traces are captured, and clusters are dynamically adjusted based on sampled traces.

Cluster data points - Maintains a list of feature-vectors and data points corresponding to the ‘N’ recent sampled Spans and assign cluster for each new datapoint using spatial clustering.

Calculate Weighted Sampling Rate (WSR) – formula which gives inverse weight to less collected data points based on the clustering due to feature vectors

Operation Method	Operation Path	Service Name	Span-1 hostname	Span-1 duration	Span-1 status code	Span-2 hostname	...

Visualizing the solution



Conclusion

- The proposed solution has the potential to significantly enhance trace quality by effectively sampling relevant data, thereby improving the representation of interesting and unusual traces.
- This solution is especially advantageous for services with high Service Level Objectives (SLOs), where traditional sampling methods frequently introduce biases towards usual/normal executions.
- Due to the loosely coupled nature of this approach, it can be further developed as a standalone service or plugin. This flexibility allows organizations to integrate and instrument their systems with OpenTelemetry-compliant architectures seamlessly.

Thank You for the time and attention !

Leave a feedback

Looking forward to connect with you to exchange more ideas, collaborations and opportunities !

Akash Gusain

aakashgusain11@gmail.com

linkedin.com/in/akash-gusain-2b375119b

Mitul Tandon

mitultandon@gmail.com

linkedin.com/in/mitul-tandon-240842177