New Relic.

White Paper

Gain an Edge with Distributed Tracing

Faster Troubleshooting for Distributed Systems

Table of Contents

From Monolith to Microservices: Challenges and Opportunities

Traditionally, monolithic software environments—where single-tiered software applications combined different components into one program—were standard. For example, a simple e-commerce application could include several functions bundled together, such as inventory, payment, and shipping. When something went wrong, and issues cropped up, it was relatively easy to identify which part of the code was at fault and why. You could dig through transactions inside that part of the application to find bottlenecks or errors.

Many organizations continue to find monolithic environments serve their purposes well. But as more engineering organizations move from monoliths to microservices, containers, and serverless architectures, what they gain in speed and flexibility is countered by increased complexity. Microservices environments can include dozens or hundreds of services, and seeing how they connect and how requests flow through them to diagnose an issue can be challenging—but also critical.

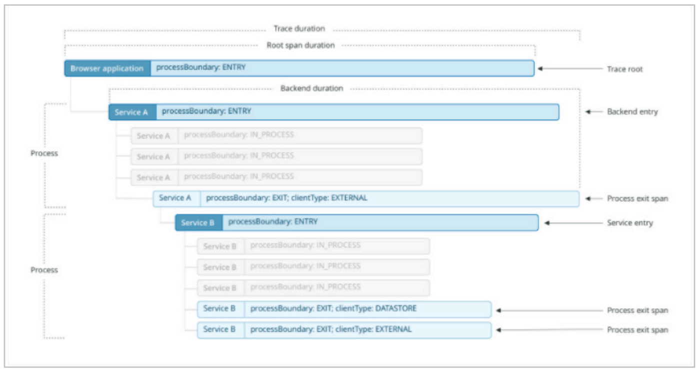
Not only that, the increased adoption of DevOps and site reliability engineering (SRE) practices coupled with technologies like orchestration, automation, and CI/CD for frequent software deployments in highly distributed environments, also introduce greater complexity for application monitoring. There are more points of failure in a distributed system, not to mention the added complexity of having various teams managing different parts of the system.

When issues occur, if you don't have the right monitoring instrumentation in place, you risk wasting a significant amount of precious time searching across your distributed systems, increasing mean time to resolution (MTTR). The time squandered while searching for answers leaves you with less time for innovating and developing new software or features.

Tracing the Path Through Complex Distributed Systems

When you monitor software and system performance for observability, you need four fundamental telemetry data types: metrics, events, logs, and traces. For large, modern systems, which are often distributed across many microservices, "tracing" often refers to distributed tracing, which is a way to monitor and analyze requests as they propagate through a complex, distributed environment and hop from service to service.

When you instrument systems for distributed tracing, all transactions generate trace telemetry, from the frontend user to the backend database calls. Referring back to the e-commerce example, when you click on a cart to make a purchase, that request now travels through several distinct frontend and backend services. The request might include the inventory service to ensure there's inventory available, the payment service, the shipping service, and ultimately the request completes and comes back to the user. Every time a request hops from one service to another, it emits a span with tracing telemetry. Once the request has finished, spans are stitched together to create a complete trace of the request's journey through the system.



New Relic distributed tracing

With distributed tracing, you can:

- Trace the path of a request as it travels across a complex system
- Understand upstream and downstream service dependencies
- Discover the latency of the components along that path
- Understand where bottlenecks are occurring in the request path
- See and analyze where errors happen in the transaction at the individual service level

Besides e-commerce companies, technology, media, and financial services organizations are among those that have thousands of customers using their systems, requiring the use of distributed tracing for troubleshooting complex architectures.

Distributed tracing requires reporting and processing vast amounts of tracing telemetry. For this reason, many organizations use sampling to capture a representative sample of trace activity. Ideally, the sampled data represents the characteristics of the larger data population.

This white paper outlines the different types of sampling methods used in distributed tracing. It shows why software teams need the flexibility to choose head-based sampling or fully managed tail-based sampling to meet the monitoring requirements for each application.

Efficient Head-Based Sampling

Traditional distributed tracing solutions use head-based sampling to track and analyze what happens to a transaction across all of the services it touches. Typically, head-based sampling is done within the agent that's responsible for collecting trace telemetry by randomly selecting which traces will be sampled for analysis. The sampling decisions are made before traces are complete. Because there's no way to know which trace might encounter an issue, you may miss traces that contain unusually slow processes or errors.

Head-based sampling works well for giving you an overall statistical sampling of requests through a distributed system. It does a good job of catching traces with errors or latency in applications that have a lower volume of transactions and in environments that have a mix of monolith and microservices-based architectures. Head-based sampling is an efficient way to sample a vast amount of trace data in real time, and there is little to no impact on application performance.

For New Relic customers, distributed tracing with head-based sampling is included with New Relic APM Pro. And with New Relic Trace API, you can ingest trace data from open source instrumentation tools such as Zipkin, Istio, OpenTelemetry, and OpenCensus and use New Relic distributed tracing as a backend for visualization, troubleshooting, and analysis.

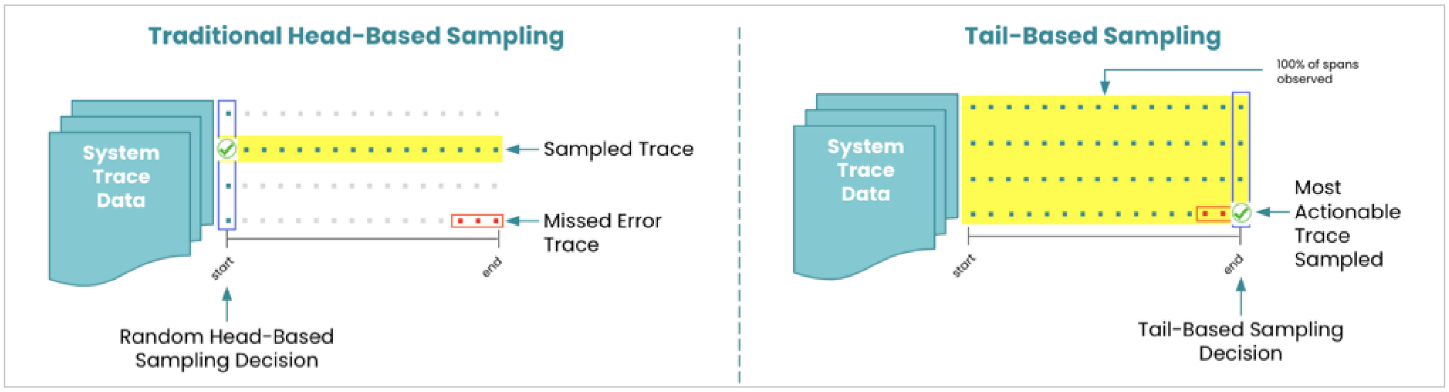
Surfacing Actionable Traces with Tail-Based Sampling

Distributed tracing with tail-based sampling helps software teams troubleshoot issues in highly distributed, high-volume microservices-based systems where teams must observe all trace telemetry and sample the traces that contain errors or unusual latency.

You could be cloud-native, or you may have "lifted and shifted" and completed a cloud migration. Needless to say, if you need the highest level of granularity in troubleshooting, tail-based sampling is less of a "nice to have" and more of a requirement.

Some organizations need their distributed tracing tool to observe and analyze every span—every hop between services—and surface the most actionable traces for troubleshooting, because any downtime could cost millions of dollars, especially during peak events. For example, one company has an average span load of 3 million spans per minute, but when a new product launches, it sees spikes of 300 million spans per minute. For this type of organization with a high transaction volume, traditional head-based sampling is inadequate.

Some vendors offer distributed tracing solutions with tail-based sampling, but require customers to deploy, manage, support, and scale gateways or satellites in their environments for data collection. What's more, you're required to take on additional infrastructure costs to run the gateways or satellites. You also have to think about the networking and scaling requirements when connecting the instrumentation that's generating telemetry data to the vendor's software, along with considering additional data egress costs. All of this toil adds additional operational burden for software teams. The bottom line is that software teams shouldn't have to manage vendor gateways or satellites when there's a better alternative.


Traditional Head-Based Sampling
System Trace Data
Sampled Trace
Missed Error Trace
start
end
Random Head-Based Sampling Decision
Tail-Based Sampling
100% of spans observed
System Trace Data
Most Actionable Trace Sampled
start
end
Tail-Based Sampling Decision

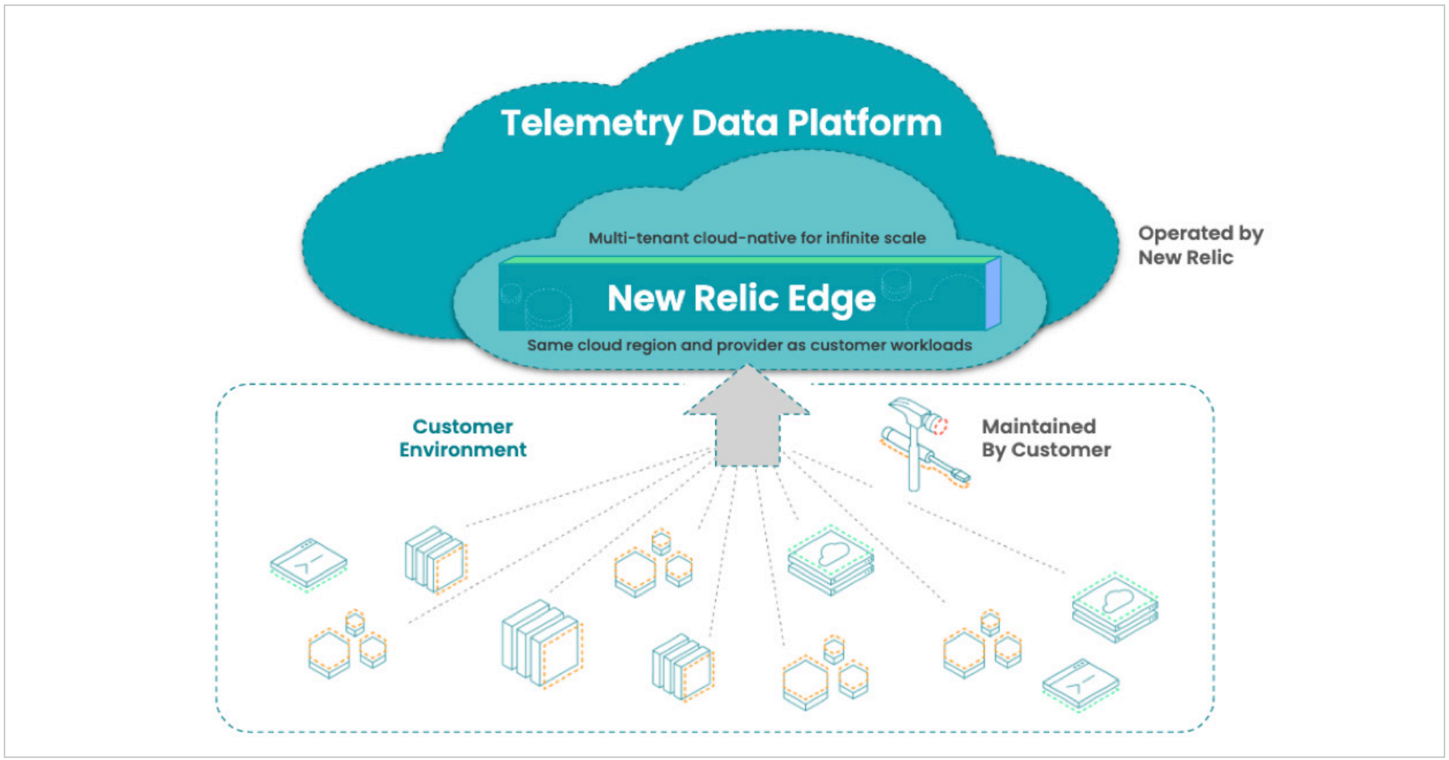
Traditional head-based sampling (left) and tail-based sampling (right)

Removing the Management Burden

New Relic Edge with Infinite Tracing offers a fully managed tail-based sampling service that observes and analyzes 100% of spans across a distributed system and provides visualizations for traces with errors or unusual latency, so software teams can quickly identify and troubleshoot issues. There's no infrastructure to manage, you don't need to staff or plan for operating the service, and it provides on-demand scalability.

The service is located in the same region and provider as your cloud-based workloads. This capability allows for low-latency and low-cost data transfer from New Relic agents, instrumentation within serverless functions, or any other data source, including third-party instrumentation. Because New Relic Edge is a fully managed SaaS solution, it allows you to eliminate the burden of deploying, managing, and scaling third-party gateways or satellites for data collection.

New Relic Edge observes every span and gives you the metrics, error data, and the essential traces you need. It provides critical insights by saving the most actionable data to New Relic. The result is unparalleled visibility into your distributed systems, allowing you to easily understand the impact of downstream latency or errors with detailed metrics, and then drill down to the saved trace data for the most relevant traces.



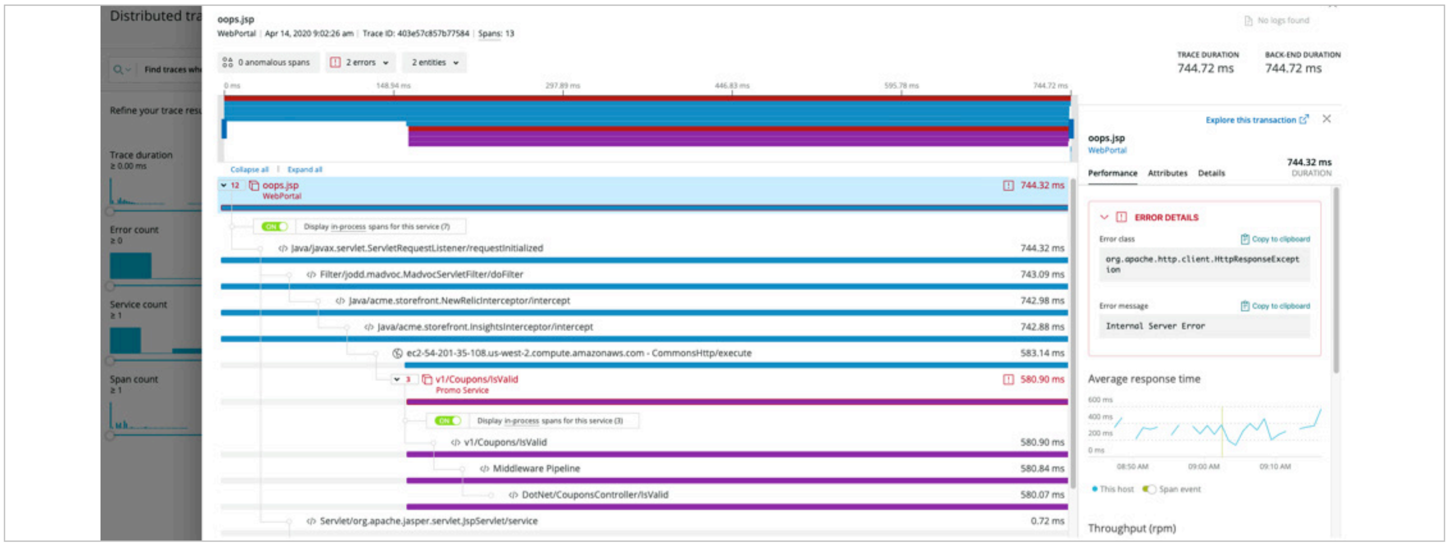
New Relic Edge with Infinite Tracing

Troubleshooting distributed systems is a classic "needle in a haystack" problem. New Relic Edge automatically eliminates the noise, based on what's important to you, such as focusing on errors or long-running traces, giving you an inventory of communication paths across your distributed system.

New Relic Edge sends the most actionable and relevant traces to NRDB, the world's most powerful telemetry database, and you can leverage enhanced distributed tracing capabilities such as anomaly detection, condensed trace views, deployment markers, trace groupings, and global trace search for quick troubleshooting across your distributed systems.

With New Relic Edge, you can:

- Enjoy a fully managed service that's cloud-local and scales on demand
- Observe and analyze 100% of the traces across your distributed system
- Visualize the most actionable traces that contain errors or unusual latency
- Eliminate the toil of deploying, managing, supporting, and scaling third-party gateways or satellites in your environments
- Leverage New Relic's full support of open instrumentation and standards for trace telemetry
- Reduce the cost of data egress charges due to proximity to your cloud workloads
- Troubleshoot more efficiently
- Reduce mean time to detection (MTTD) and MTTR with high-fidelity actionable traces
- Empower engineers and developers to focus on more important work, such as developing new features



oops.jsp
WebPortal | Apr 14, 2020 9:02:26 am | Trace ID: 403e57c857b77584 | Spans: 13

TRACE DURATION 744.72 ms | BACK-END DURATION 744.72 ms

```
12 oops.jsp  WebPortal                                                744.32 ms
   ON  Display in-process spans for this service (7)
   </> Java/javax.servlet.ServletRequestListener/requestInitialized   744.32 ms
     </> Filter/jodd.madvoc.MadvocServletFilter/doFilter              743.09 ms
       </> Java/acme.storefront.NewRelicInterceptor/intercept         742.98 ms
         </> Java/acme.storefront.InsightsInterceptor/intercept       742.88 ms
           ec2-54-201-35-108.us-west-2.compute.amazonaws.com - CommonsHttp/execute   583.14 ms
             3 v1/Coupons/IsValid  Promo Service                      580.90 ms
               ON  Display in-process spans for this service (3)
               </> v1/Coupons/IsValid                                 580.90 ms
                 </> Middleware Pipeline                              580.84 ms
                   </> DotNet/CouponsController/IsValid               580.07 ms
     </> Servlet/org.apache.jasper.servlet.JspServlet/service         0.72 ms
```

oops.jsp
WebPortal
744.32 ms DURATION

ERROR DETAILS
Error class
```
org.apache.http.client.HttpResponseException
```
Error message
```
Internal Server Error
```

Quickly see exactly which spans have high latency and errors within a trace

Heads or Tails? You Don't Need to Flip a Coin

Because New Relic offers flexible options for distributed tracing, you can make head- or tail-based sampling decisions at the application level. For critical applications for which you need to observe and analyze every trace, you can choose tail-based sampling, without having to worry about managing sampling infrastructure.

New Relic is the only vendor that gives software teams the flexibility to choose distributed tracing with head-based sampling or fully managed tail-based sampling. With less to manage, there's more room for innovation and gaining a competitive advantage.

Getting started with New Relic Edge with Infinite Tracing is easy. Visit New Relic Documentation to learn more.

New Relic.