ALIBABA CLOUD

Alibaba Cloud

Tracing Analysis
Product Introduction

Document Version: 20201222

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

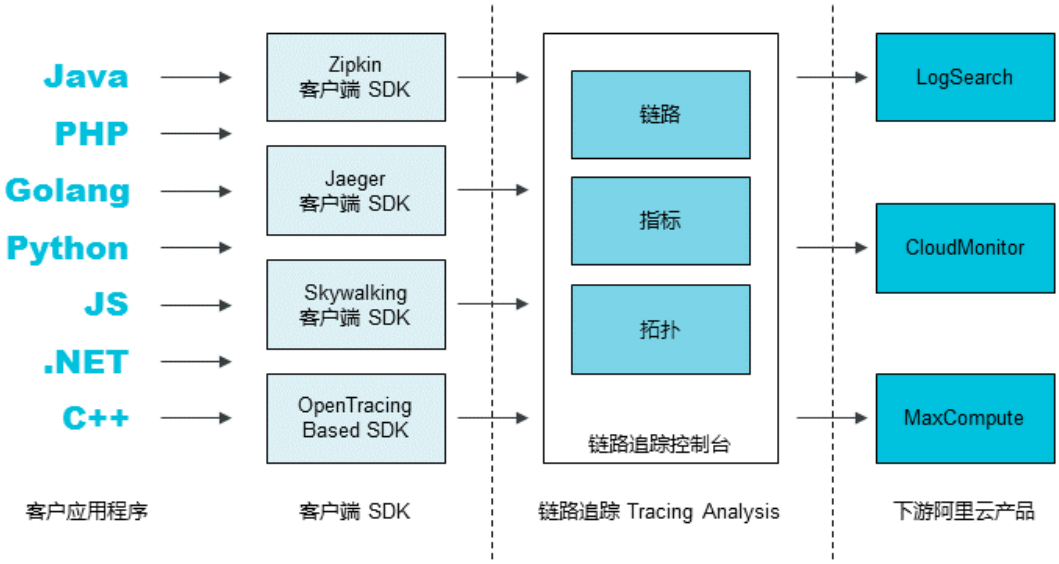
1.What is Tracing Analysis?

Tracing Analysis provides a variety of tools for distributed applications, including trace mapping, request counting, trace topology, and application dependency Analysis. With these tools, developers can quickly identify the performance bottlenecks of a distributed application architecture and improve the efficiency of microservice development and diagnosis.

Architecture

The following figure shows the product architecture of link tracing.

Product Architecture of tracing analysis



The main work process is as follows:

- The customer application reports service calls through the integrated tracing analysis SDK. Tracing analysis supports SDKs from various open source communities and the OpenTracing standard.
- After the tracing data is reported to the tracing analysis console, the tracing analysis component aggregates, computes, and persists the data in real time to form monitoring data such as the trace details, Performance Overview, and real-time topology data. You can then troubleshoot and diagnose the problem.
- You can use the trace data to connect to downstream Alibaba cloud products, such as LogSearch, CloudMonitor, and MaxCompute, for offline analysis and alert management.

Features

The main features of link tracing include:

- Query and diagnostics of distributed traces: This feature tracks microservice user requests in the distributed architecture and summarizes these requests into distributed traces.
- Real-time collection of application performance data: This feature tracks all user requests for an application and collects and analyzes in real time the performance data of the services and resources that constitute the application.
- Dynamic Discovery of distributed topology: In this way, you can obtain the distributed call information collected by tracing analysis for your distributed micro-application and PaaS products.

- Programming for multiple languages: This service is based on OpenTracing and fully compatible with open-source communities such as Jaeger and Zipkin.
- Various downstream integration scenarios: uses collected traces for log analysis and connects to downstream analysis platforms such as MaxCompute.

2.Terms

This topic describes basic concepts before using tracing analysis, including the role of the distributed tracing system, what a call chain is, the OpenTracing data model that tracing links depend on, and how data is reported to the tracing system.

Why is a distributed tracing system needed?

In order to cope with various complex business, development engineers began to adopt Agile development, continuous integration and other development methods. The system architecture has evolved from a stand-alone software system to a microservices model. Microservices are built on different sets of software, which may be developed by different teams, implemented in different programming languages, or released on multiple servers. Therefore, if a service error occurs, it may cause service exceptions in dozens of applications.

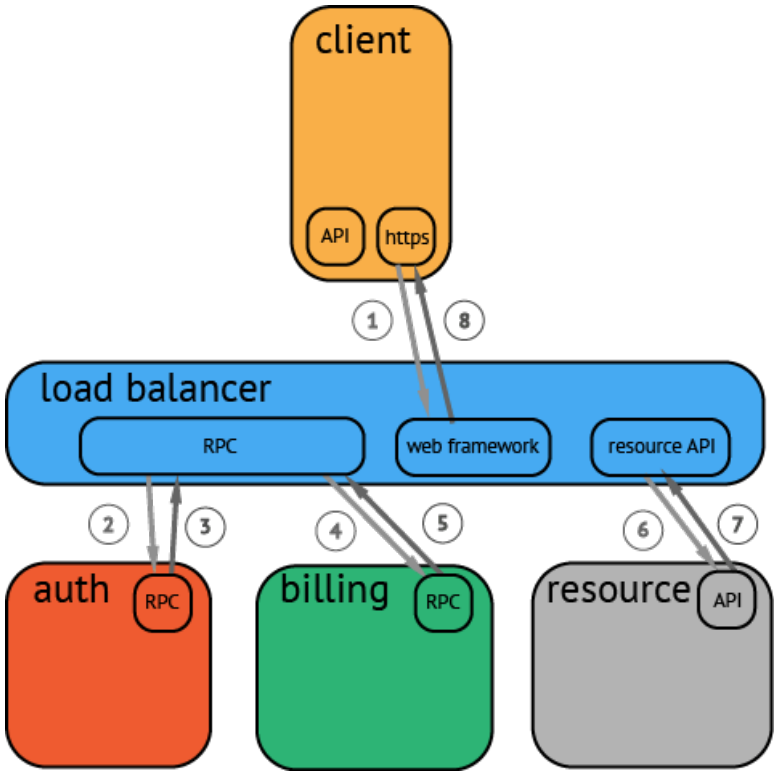
The distributed tracing system can record request information, such as the execution process and time consumption of a remote method call. It is an important tool for troubleshooting system problems and system performance.

What is a Trace?

Broadly speaking, a trace of call represents the execution process of a transaction or process in a (distributed) system. In the OpenTracing standard, a call chain is a Directed Acyclic Graph (Directed Acyclic Graph) composed of multiple spans. Each Span represents a named and timed continuous execution segment in the call chain.

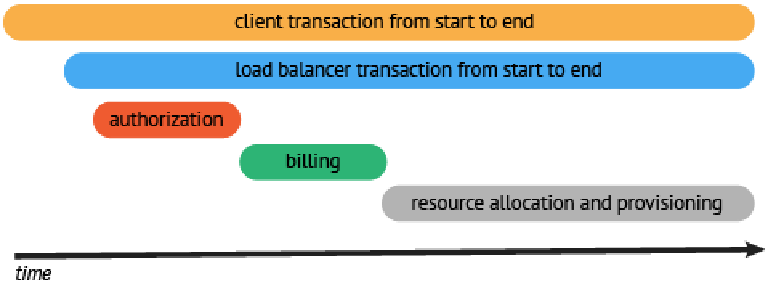
The following example shows a distributed call. When a client initiates a request, the request first goes to the Server Load Balancer, passes through the authentication service, billing service, and finally the resource, and finally returns the result.

Example of a distributed call



After the system collects and stores the data, the distributed tracing system can be used to generate a timing diagram that contains a timeline.

Link diagram containing the timeline



Data Model

Overall concept

In OpenTracing, a Trace is implicitly defined by the Span in this Trace. A trace can be considered as a directed acyclic graph (DAG) that consists of multiple spans. The relationship between spans is named References. The call chain in the following example consists of eight spans.

```
Causal relationship between spans in A single Trace [Span A] ← ←(The root Span) | + ------ + ------ + | | [span B] [Span C] ← ←(Span C is A child node of Span A, ChildOf) | [Span D] + --- + ------- + | | [Span E] [Span F] |result > [Span G] |result> [Span H] // Trigger function (Span G is called after Span F, FollowsFrom)
```

In some cases, a timing diagram based on a timeline can be used to better display the call chain.

```
time relationship between spans in A single Trace -- | ------- | ------- | ------- | ------- | ------- | ------- | ------- |-> time [Span A&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip; ..&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip; ..] [Span C&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip;&hellip; ..] [Span E&hellip;&hellip;&hellip;&hellip;] [Span F.] [span G ・] [Span H ・]
```

Link

You can call this operation to create tracers (startSpan), Extract (Extract), and Inject (passthrough). It has the following capabilities:

- Create a new Span or set the Span property

```
/**Create and start a span. On the page that appears, specify the name of the operation and options. * *For example,** create a Span: **sp := tracer. Without parentSpan. StartSpan(" GetFeed ") ** create a Span with parentSpan * sp := tracer.StartSpan("GetFeed",opentracing.ChildOf(parentSpan.Context())) example */ StartSpan(operationName string, opts ...StartSpanOption) Span
```

Each Span contains the following objects:

- Operation name: Operation name (also known as Span name).
- Start timestamp: The Start time of the day period.
- Finish timestamp: indicates the end time.
- Span tag: a set of Span tags. It is composed of a set of key-value pairs. In a key-value pair, the key must be a String and the value can be a String, Boolean, or numeric value.
- Span log: A Collection of Span logs. Each Log operation contains one key-value pair and one timestamp. In a key-value pair, the key must be a String and the value can be of any type.
- SpanContext: The pan context object. Each SpanContext contains the following states:
  - To implement any OpenTracing service, it must rely on a unique Span to transmit the status of the current call chain across process boundaries (for example, the Trace and Span ids).
  - Baggage Items are the data accompanying a Trace and a collection of key-value pairs. They are stored in a Trace and must be transmitted across process boundaries.
- References (relationship between spans): Zero or multiple related spans. Spans establish the relationship based on the SpanContext.

- Pass-through data

  Passthrough data is divided into two steps:

  i. Parses the SpanContext object from the request.

  ```
  // Inject() takes the `sm` SpanContext instance and injects it for // propagation within `carrier`. The actual type of `carrier` depends on // the value of `format`. /**Parse the SpanContext (including traceId, spanId, and bagged) from the Carrier based on the format parameter. **Example:** carrier := opentracing.HTTPHeadersCarrier(httpReq.Header) // * clientContext, err := tracer.Extract(opentracing.HTTPHeaders, carrier) // */ Extract(format interface parameter parameter, carrier interface{}) (spanccontext, error)
  ```

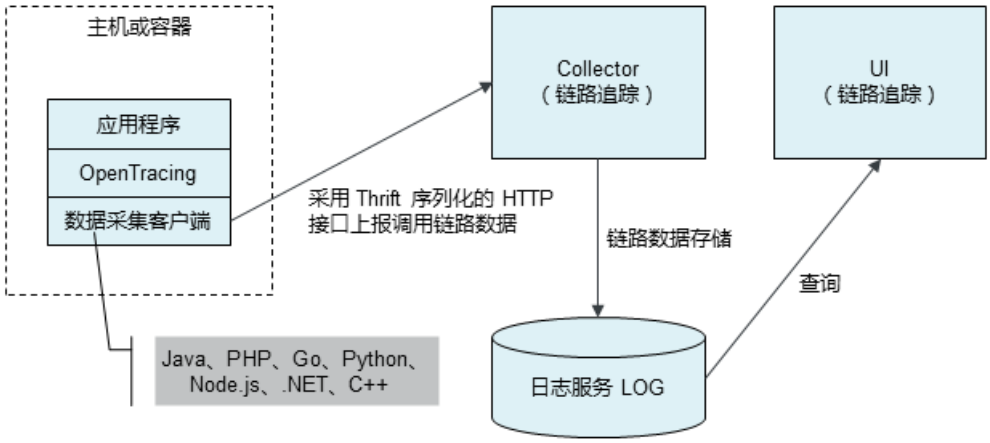
  ii. Inject SpanContext to the request.

  ```
  /**Inject *** inject traceId,spanId, and Baggage in SpanContext to the request (Carrier) according to the format parameter. **e.g** carrier := opentracing.HTTPHeadersCarrier(httpReq.Header) then * err := tracer.Inject(span.Context(), opentracing.HTTPHeaders, carrier) producer */ Inject(sm SpanContext, format interface response parameter, carrier interface{}) error
  ```

How is the data reported?

The following figure shows how data is reported without an Agent.

Data Reporting



The following figure shows how the data is reported by an Agent.

Report data through Agent