链路追踪| Sleuth+Zipkin

通过链路追踪，可以记录请求在整个调用链路的日志信息、对应用性能进行监控、显示服务调用情况。

- trace：调用链路，有一个链路 ID（traceID）。
- span：工作单元，即单次调用。

Sleuth+Zipkin

Sleuth 全称 SpringCloud Sleuth，主要为分布式系统提供链路追踪提>供解决方案，兼容 Zipkin。如果想对服务调用链路进行追踪，则需要为整个链路上的所有服务都添加 Sleuth 和 Zipkin。

打印日志 Sleuth

引入 Sleuth 依赖：

```XML
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-sleuth</artifactId>
</dependency>
```

配置 Sleuth：

```YAML
logging:
 level:
   root: INFO
   org.springframework.web.servlet.DispatcherServlet: DEBUG
   org.springframework.cloud.sleuth: DEBUG
```

配置好后，运行服务，日志信息就将在控制台被打印出来，但是打印出来的日志不是特别直观。

聚合日志 Zipkin

Zipkin 可以将上面的日志进行日志聚合，将日志信息可视化展示、并且提供全文检索。

- 服务端：聚合所有客户端的日志信息，但不提供持久化。
- 客户端：采集目标服务的日志信息，最终统一到服务端。

默认可以通过 http、Kafka、RabbitMQ 传输收集数据。 zipkin 公开了 Api 接口：https://zipkin.io/zipkin-api/#/default/post_spans

Zipkin Server

- 下载：https://search.maven.org/remote_content?g=io.zipkin.java&a=zipkin-server&v=LATEST&c=exec
- 启动 Zipkin Server：默认端口 9411

```Bash
# 启动zipkin
java  -jar zipkin-server-2.12.9-exec.jar
```

- 管理后台： http://127.0.0.1:9411

Zipkin Client

引入 zipkin 依赖：

```xml
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-zipkin</artifactId>
</dependency>
```

配置 zipkin 信息：

```yaml
spring:
 zipkin:              #聚合日志配置
   base-url: http://127.0.0.1:9411/ #zipkin server的请求地址
   sender:
     type: web            #请求方式,默认以http的方式向zipkin server发送追踪数据
 sleuth:
   sampler:
     probability: 1.0         #采集的百分比，1表示全部采集
```

服务端启动：启动时指定 mysql 来持久化采集的数据，持久化看下一节。

```ini
java -jar zipkin-server-2.12.9-exec.jar
--STORAGE_TYPE=mysql       # 存储类型
--MYSQL_HOST=127.0.0.1     # mysql主机地址
--MYSQL_TCP_PORT=3306      # mysql端口
--MYSQL_DB=zipkin          # mysql数据库名称
--MYSQL_USER=root          # mysql用户名
--MYSQL_PASS=root          # mysql密码
```

日志持久化

采集到的日志信息可以持久化到 MySQL、Elasticsearch、Cassandra 中.

持久化到 mysql 需要先创建一张存储数据的表，表结构如下：

```sql
CREATE DATABASE IF NOT EXISTS zipkin;

CREATE TABLE IF NOT EXISTS zipkin_spans (
  `trace_id_high` BIGINT NOT NULL DEFAULT 0 COMMENT 'If non zero, this means the trace uses 128 bit traceIds instead of 64 bit',
  `trace_id` BIGINT NOT NULL,
  `id` BIGINT NOT NULL,
  `name` VARCHAR(255) NOT NULL,
  `parent_id` BIGINT,
  `debug` BIT(1),
  `start_ts` BIGINT COMMENT 'Span.timestamp(): epoch micros used for endTs query and to implement TTL',
  `duration` BIGINT COMMENT 'Span.duration(): micros used for minDuration and maxDuration query'
) ENGINE=InnoDB ROW_FORMAT=COMPRESSED CHARACTER SET=utf8 COLLATE utf8_general_ci;

ALTER TABLE zipkin_spans ADD UNIQUE KEY(`trace_id_high`, `trace_id`, `id`) COMMENT 'ignore insert on duplicate';
ALTER TABLE zipkin_spans ADD INDEX(`trace_id_high`, `trace_id`, `id`) COMMENT 'for joining with zipkin_annotations';
ALTER TABLE zipkin_spans ADD INDEX(`trace_id_high`, `trace_id`) COMMENT 'for getTracesByIds';
ALTER TABLE zipkin_spans ADD INDEX(`name`) COMMENT 'for getTraces and getSpanNames';
ALTER TABLE zipkin_spans ADD INDEX(`start_ts`) COMMENT 'for getTraces ordering and range';

CREATE TABLE IF NOT EXISTS zipkin_annotations (
  `trace_id_high` BIGINT NOT NULL DEFAULT 0 COMMENT 'If non zero, this means the trace uses 128 bit traceIds instead of 64 bit',
  `trace_id` BIGINT NOT NULL COMMENT 'coincides with zipkin_spans.trace_id',
  `span_id` BIGINT NOT NULL COMMENT 'coincides with zipkin_spans.id',
  `a_key` VARCHAR(255) NOT NULL COMMENT 'BinaryAnnotation.key or Annotation.value if type == -1',
  `a_value` BLOB COMMENT 'BinaryAnnotation.value(), which must be smaller than 64KB',
  `a_type` INT NOT NULL COMMENT 'BinaryAnnotation.type() or -1 if Annotation',
  `a_timestamp` BIGINT COMMENT 'Used to implement TTL; Annotation.timestamp or zipkin_spans.timestamp',
  `endpoint_ipv4` INT COMMENT 'Null when Binary/Annotation.endpoint is null',
  `endpoint_ipv6` BINARY(16) COMMENT 'Null when Binary/Annotation.endpoint is null, or no IPv6 address',
  `endpoint_port` SMALLINT COMMENT 'Null when Binary/Annotation.endpoint is null',
  `endpoint_service_name` VARCHAR(255) COMMENT 'Null when Binary/Annotation.endpoint is null'
) ENGINE=InnoDB ROW_FORMAT=COMPRESSED CHARACTER SET=utf8 COLLATE utf8_general_ci;

ALTER TABLE zipkin_annotations ADD UNIQUE KEY(`trace_id_high`, `trace_id`,`span_id`, `a_key`, `a_timestamp`) COMMENT 'Ignore insert on dupli
ALTER TABLE zipkin_annotations ADD INDEX(`trace_id_high`, `trace_id`, `span_id`) COMMENT 'for joining with zipkin_spans';
ALTER TABLE zipkin_annotations ADD INDEX(`trace_id_high`, `trace_id`) COMMENT 'for getTraces/ByIds';
ALTER TABLE zipkin_annotations ADD INDEX(`endpoint_service_name`) COMMENT 'for getTraces and getServiceNames';
ALTER TABLE zipkin_annotations ADD INDEX(`a_type`) COMMENT 'for getTraces';
ALTER TABLE zipkin_annotations ADD INDEX(`a_key`) COMMENT 'for getTraces';
ALTER TABLE zipkin_annotations ADD INDEX(`trace_id`, `span_id`, `a_key`) COMMENT 'for dependencies job';

CREATE TABLE IF NOT EXISTS zipkin_dependencies (
  `day` DATE NOT NULL,
  `parent` VARCHAR(255) NOT NULL,
  `child` VARCHAR(255) NOT NULL,
  `call_count` BIGINT
) ENGINE=InnoDB ROW_FORMAT=COMPRESSED CHARACTER SET=utf8 COLLATE utf8_general_ci;

ALTER TABLE zipkin_dependencies ADD UNIQUE KEY(`day`, `parent`, `child`);
```

执行上面的sql脚本，并启动zipkin server：

```ini
java -jar zipkin-server-2.12.9-exec.jar
--STORAGE_TYPE=mysql
--MYSQL_HOST=127.0.0.1
--MYSQL_TCP_PORT=3306
--MYSQL_DB=zipkin
--MYSQL_USER=root
--MYSQL_PASS=root
```

将采集到的数据持久化到Elasticsearch：

- 下载Elasticsearch：https://www.elastic.co/cn/downloads/past-releases/elasticsearch-7-9-1
- 启动Elasticsearch：运行bin/elasticsearch.bat
- 启动zipkin server：

```ini
java -jar zipkin-server-2.12.9-exec.jar  --STORAGE_TYPE=elasticsearch --ES-HOST=localhost:9200
```

数据采集 RabbitMQ

使用消息中间件在客户端、服务端直接添加消息队列，使同步的http采集变为异步采集，从而优化Zipkin的采集效率。

安装rabbitmq-server-3.7.4.exe、启动zipkin server：

```ini
java -jar zipkin-server-2.12.9-exec.jar  --RABBIT_ADDRESSES=127.0.0.1:5672
```

- RABBIT_ADDRESSES： 指定RabbitMQ地址
- RABBIT_USER： 用户名（默认guest）
- RABBIT_PASSWORD： 密码（默认guest）

引入依赖：

```xml
<!--消息中间件-->
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-sleuth-zipkin</artifactId>
</dependency>
<dependency>
   <groupId>org.springframework.amqp</groupId>
   <artifactId>spring-rabbit</artifactId>
</dependency>
```

添加配置：

```yaml
spring:
 application:
   name: api-gateway     #指定服务名
 zipkin: #zipkin配置
   #base-url: http://127.0.0.1:9411/  #zipkin server的请求地址
   sender:
     type: rabbit      #请求方式：web→rabbit
 sleuth:  #springcloud sleuth配置
```

```
    sampler:
      probability: 1.0      #采样的百分比
  rabbitmq:  #rabbitmq配置
    host: localhost
    port: 5672
    username: guest
    password: guest
    listener:         # 重试策略配置
      direct:
        retry:
          enabled: true
      simple:
        retry:
          enabled: true
```

Skywalking

Skywalking 的特点是与项目独立，不用额外编写代码就可以实现检测，它是一个可观测性分析平台（OAP）也是一个应用性能管理系统（APM）。skywalking 实现链路追踪需要使用探针来采集数据，只需要在需要监测的服务上安装探针即可。

- 链路追踪
- 服务网格(Service Mesh)遥测分析
- 度量(Metric)聚合和可视化一体化

Skywalking 下载：http://skywalking.apache.org/downloads/

更多

https://cloud.tencent.com/developer/article/1676282