Spring Cloud Sleuth + Zipkin实现分布式链路跟踪

博客分类： spring cloud(F版)

Spring Cloud Sleuth为SpringCloud应用实现了一种分布式链路跟踪解决方案，通过Sleuth可以很清楚了解到一个服务请求经过了哪些服务，每个服务处理花费了多长时间。

Sleuth术语

Span

Span是基本的工作单元。Span包括一个64位的唯一ID，一个64位trace码，描述信息，时间戳事件，key-value 注解(tags)，span处理者的ID（通常为IP）。

每个trace中会调用若干个服务，为了记录调用了哪些服务，以及每次调用的消耗时间等信息，在每次调用服务时，埋入一个调用记录，称为一个span。

Trace

包含一系列的span，它们组成了一个树型结构。

从客户发起请求（request）抵达被追踪系统的边界开始，到被追踪系统向客户返回响应（response）为止的过程，称为一个 trace。

Annotation

用于及时记录存在的事件。常用的Annotation如下：

cs - Client Sent：客户端发送一个请求，表示span的开始

sr - Server Received：服务端接收请求并开始处理它。(sr-cs)等于网络的延迟

ss - Server Sent：服务端处理请求完成，开始返回结果给服务端。(ss-sr)表示服务端处理请求的时间

cr - Client Received：客户端完成接收返回结果，此时span结束。(cr-sr)表示客户端接收服务端数据的时间

Sleuth的用途：

耗时分析: 通过Sleuth可以很方便的了解到每个采样请求的耗时，从而分析出哪些服务调用比较耗时;

可视化错误: 对于程序未捕捉的异常，可以通过集成Zipkin服务界面上看到;

链路优化: 对于调用比较频繁的服务，可以针对这些服务实施一些优化措施。

Spring Cloud Sleuth 可以结合Zipkin，将信息发送到Zipkin，利用Zipkin的存储来存储信息，利用Zipkin UI来展示数据。

Zipkin是Twitter的一个开源项目，它基于Google Dapper实现，它致力于收集服务的定时数据，以解决微服务架构中的延迟问题，包括数据的收集、存储、查找和展现。

Zipkin原理：基本思路是在服务调用的请求和响应中加入ID，标明上下游请求的关系。利用这些信息，可以可视化地分析服务调用链路和服务间的依赖关系。

Zipkin提供了可插拔数据存储方式：In-Memory（默认）、MySql、Cassandra 以及 Elasticsearch。

Zipkin主要由4个核心组件构成：

Collector：收集器组件，它主要用于处理从外部系统发送过来的跟踪信息，将这些信息转换为Zipkin内部处理的Span格式，以支持后续的存储、分析、展示等功能。

Storage：存储组件，它主要对处理收集器接收到的跟踪信息，默认会将这些信息存储在内存中，我们也可以修改此存储策略，通过使用其他存储组件将跟踪信息存储到数据库中。

RESTful API：API组件，它主要用来提供外部访问接口。比如给客户端展示跟踪信息，或是外接系统访问以实现监控等。

Web UI：UI组件，基于API组件实现的上层应用。通过UI组件用户可以方便而又直观地查询和分析跟踪信息。

Zipkin分为两端，一个是Zipkin服务端，一个是Zipkin客户端，客户端也就是微服务应用。客户端会配置服务端的URL地址，一旦发生服务间的调用时，会被配置在微服务里面的Sleuth监听器监听，并生成相应的Trace和Span信息发送给服务端。发送的方式主要有两种，一种是HTTP报文的方式，另一种是消息总线的方式如RabbitMQ。

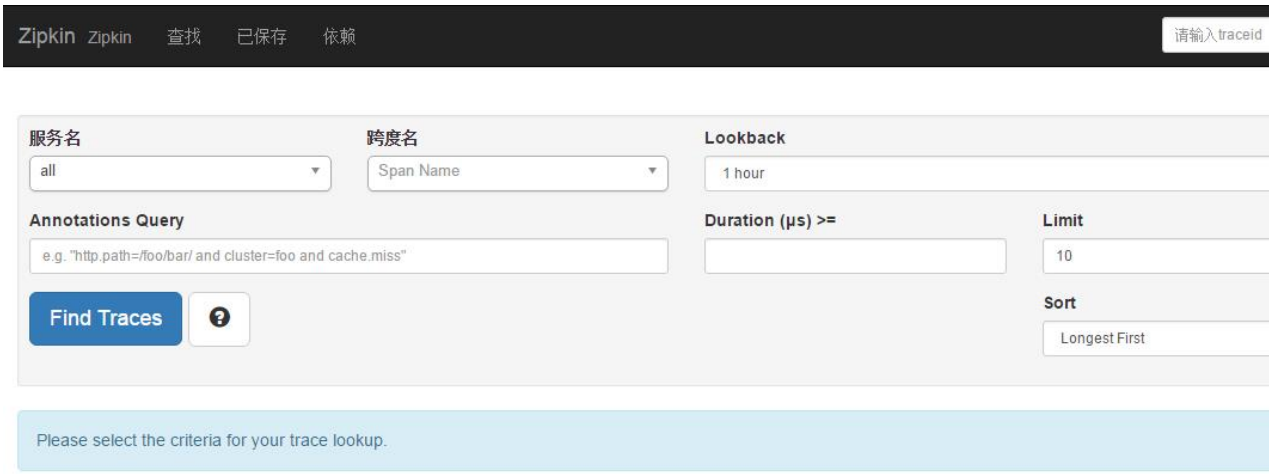
Zipkin服务端：

使用Spring Boot 2.x版本后，官方直接提供了编译好的jar包来给我们使用，比如本范例使用的jar包是zipkin-server-2.9.4-exec.jar

到 https://dl.bintray.com/openzipkin/maven/io/zipkin/java/zipkin-server/ 下载相应的jar包。

执行 java -jar zipkin-server-2.9.4-exec.jar 命令启动Zipkin Server，端口默认是9411

在浏览器访问 http://localhost:9411，显示效果如下图：



Zipkin客户端：

在pom.xml文件添加以下依赖：

Xml代码

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

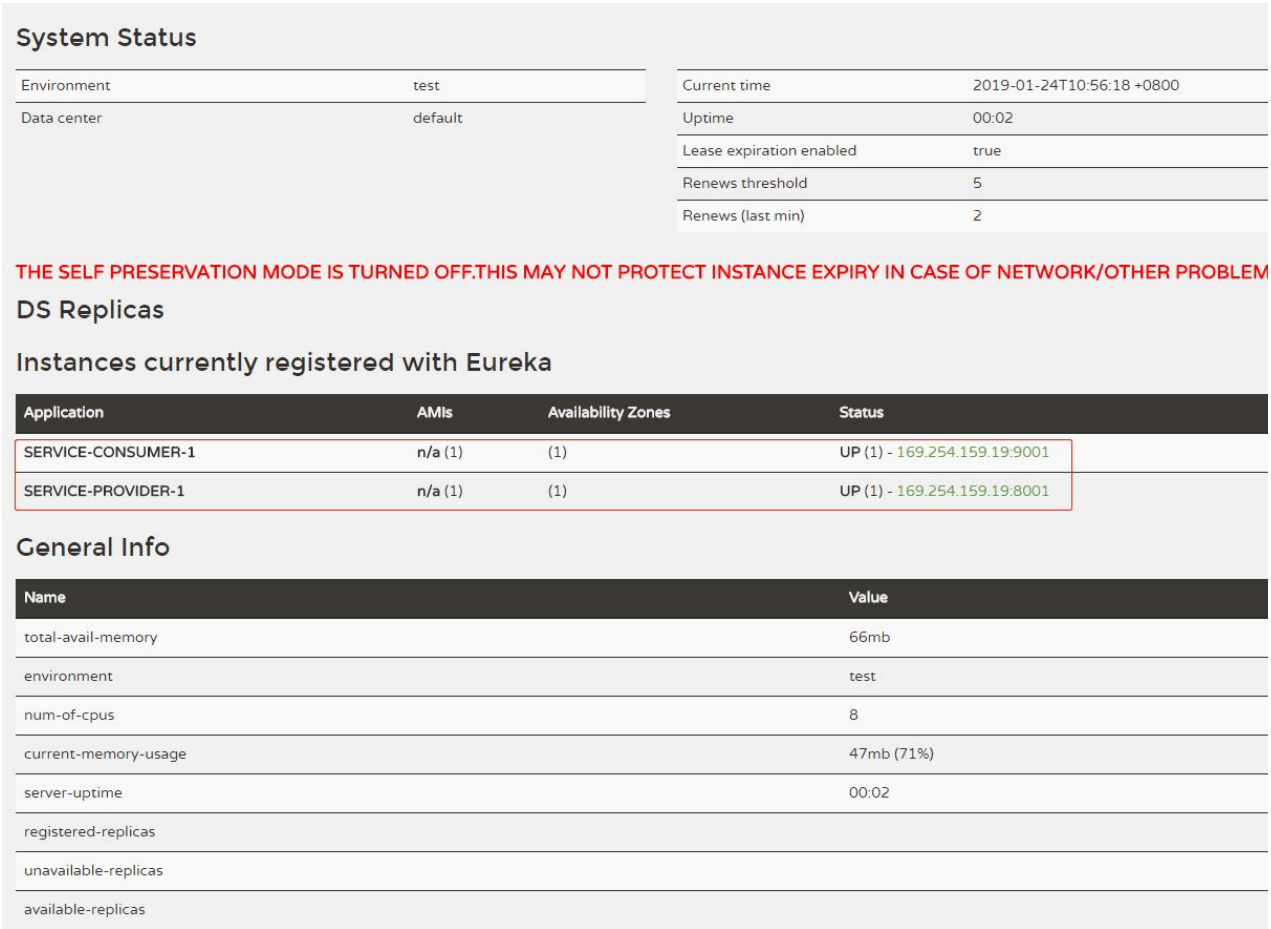
在application.properties文件添加以下配置：

Java代码

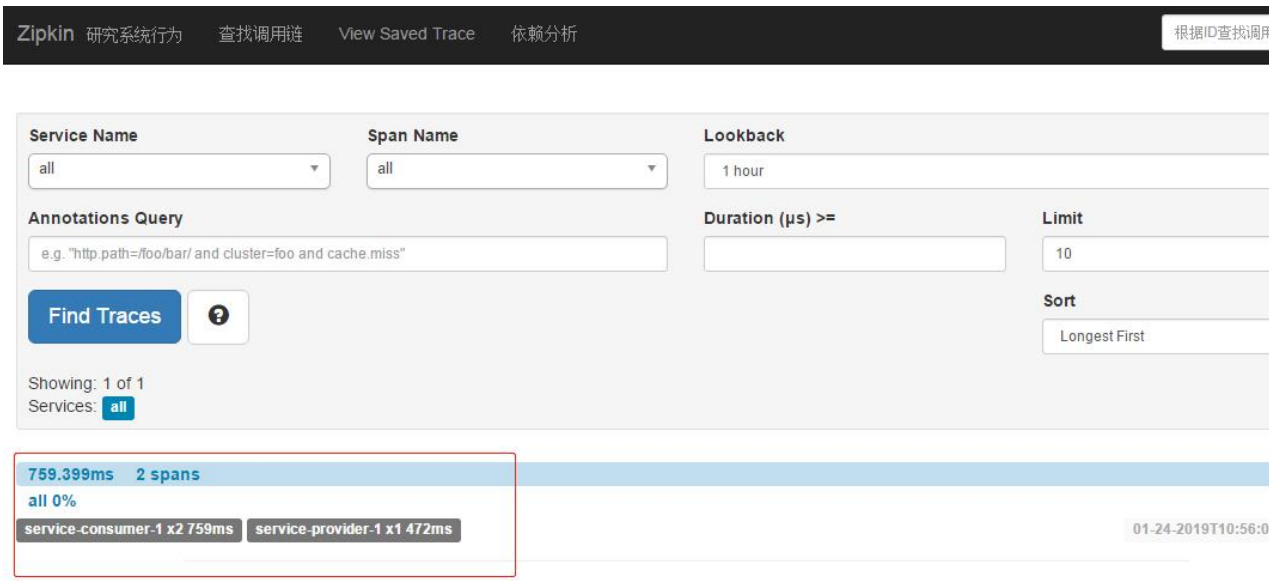
```
#设置采样比例为1.0。默认是0.1
spring.sleuth.sampler.probability=1.0

#Zipkin服务器的地址
spring.zipkin.base-url=http://localhost:9411/
```

启动注册中心和各个微服务应用，注册中心的主界面最终如下：



访问微服务URL后，在Zipkin Server即可查看到服务的调用关系：



Zipkin Server改用Mysql存储数据：

到 https://github.com/openzipkin/zipkin/tree/master/zipkin-storage 下载mysql的建表脚本

手动创建一个名为zipkin的数据库，执行建表脚本

执行以下命令启动zipkin-server：

Java代码

```
1.  java -jar zipkin-server-2.9.4-exec.jar --STORAGE_TYPE=mysql --MYSQL_HOST=localhost --MYSQL_TCP_PORT=3306 --MYSQL_USER=root --MYSQL_PASS=root
```

查看图片附件

用Spring Cloud Stream构建消息驱动的微服 ... | Spring Cloud Bus实现配置的动态更新

分享到:

2019-01-24 10:42 | 浏览 469 | 评论(0) | 分类:互联网 | 查看更多

评论

发表评论

您还没有登录,请您登录后再发表评论