第十二章 Spring Cloud分布式链路跟踪

需求背景

- 微服务架构下，一个请求可能会经过多个服务才会得到结果，如果在这个过程中出现了异常，就很难去定位问题。所以，必须要实现一个分布式链路跟踪的功能，直观的展示出完整的调用过程。

- 购票微服务方法增加日志功能

```
private static final Log log = LogFactory.getLog(WebController.class);
@ApiOperation(value = "远程方法：根据用户ID查询用户的方法")
@RequestMapping(value = "/order",method = RequestMethod.GET)
public String order(){
    log.info("开始调用order方法...");
    //模拟当前用户
    Integer id = 2;
    User user = userController.findById(id);
```

- 用户微服务方法增加日志跟踪代码

```
/*** 根据id查询用户     */
private static final Log log = LogFactory.getLog(UserController.class);
@ApiOperation(value = "查询主键用户")
@RequestMapping(value = "/{id}",method = RequestMethod.GET)
public User findById(@PathVariable Integer id){
    log.info("进入了UserController的findById方法");
    System.out.println("用户微服务11111");
    return userService.findById(id);    }
```

- 通过网关访问购票微服务

购票成功

- 购票微服务后台信息

```
com.example.controller.WebController      : 开始调用order方法...
s.c.a.AnnotationConfigApplicationContext : Refreshing SpringClientFactory-myshop-user
f.a.AutowiredAnnotationBeanPostProcessor : JSR-330 'javax.inject.Inject' annotation f
c.netflix.config.ChainedDynamicProperty  : Flipping property: myshop-user.ribbon.Acti
c.n.u.concurrent.ShutdownEnabledTimer    : Shutdown hook installed for: NFLoadBalance
c.netflix.loadbalancer.BaseLoadBalancer  : Client: myshop-user instantiated a LoadBal
c.n.l.DynamicServerListLoadBalancer      : Using serverListUpdater PollingServerListU
c.netflix.config.ChainedDynamicProperty  : Flipping property: myshop-user.ribbon.Acti
```

- 用户微服务后台信息

```
com.example.controller.UserController     : 进入了UserController的findById方法

money2_0_0_, user0_.password as password3_0_0_, user0_.sex as sex4_0_0_, user
c.n.d.s.r.aws.ConfigClusterResolver       : Resolving eureka endpoints via con
```

Spring Cloud Sleuth

- Spring Cloud Sleuth是一个在应用中实现日志跟踪的强有力的工具。使用Sleuth库可以应用于计划任务 、多线程服务或复杂的Web请求，尤其是在一个由多个服务组成的系统中。当我们在这些应用中来诊断问题时，即使有日志记录也很难判断出一个请求需要将哪些操作关联在一起。

- 如果想要诊断复杂操作，通常的解决方案是在请求中传递唯一的ID到每个方法来识别日志。而Sleuth可以与日志框架Logback、SLF4J轻松地集成，通过添加独特的标识符来使用日志跟踪和诊断问题。

整合Spring Cloud Sleuth

- 导入依赖（gateway，用户微服务和购票微服务）

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

- 请求微服务，然后查询日志

```
[myshop-gateway,7ed1db46f3bd838b,7ed1db46f3bd838b,false] 18564
INFO [myshop-web,7ed1db46f3bd838b,d82e6650da390731,false] 18960 ---
com.example.controller.WebController      : 开始调用order方法...
INFO [myshop-user,7ed1db46f3bd838b,1f580e165f59bbda,false] 13476
com.example.controller.UserController     : 进入了UserController的findById方法
```

日志格式

- 日志的格式为：[application name, traceId, spanId, export]
  - application name — 应用的名称，也就是application.properties中的spring.application.name参数配置的属性。
  - traceId — 为一个请求分配的ID号，用来标识一条请求链路。
  - spanId — 表示一个基本的工作单元，一个请求可以包含多个步骤，每个步骤都拥有自己的spanId。一个请求包含一个TraceId，多个SpanId
  - export — 布尔类型。表示是否要将该信息输出到类似Zipkin这样的聚合器进行收集和展示。
  - 可以看到，TraceId在两条日志中是相同的，即使消息来源于两个不同的类。这就可以在不同的日志通过寻找traceid来识别一个请求。

Spring Cloud Sleuth + Zipkin

- 单独使用Spring Cloud Sleuth依靠控制台追踪日志的方法非常不方便，不直观！这时可以使用Zipkin来收集所有日志信息，更加直观！

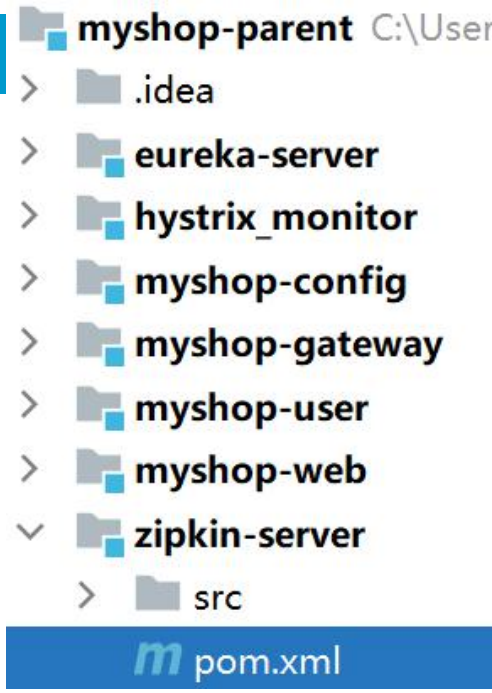
- Zipkin分布式跟踪系统；它可以帮助收集时间数据，解决在microservice架构下的延迟问题；它管理这些数据的收集和查找；Zipkin的设计是基于谷歌的Google Dapper论文。

- 每个应用程序向Zipkin报告定时数据，Zipkin UI呈现了一个依赖图表来展示多少跟踪请求经过了每个应用程序；如果想解决延迟问题，可以过滤或者排序所有的跟踪请求，并且可以查看每个跟踪请求占总跟踪时间的百分比。

搭建Zipkin服务器

- 导入依赖

```
<dependency>
        <groupId>io.zipkin.java</groupId>
        <artifactId>zipkin-server</artifactId>
        <version>2.9.4</version>
        <!--排除-->
        <exclusions>
            <exclusion>
                <groupId>org.apache.logging.log4j</groupId>
                <artifactId>log4j-slf4j-impl</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
    <dependency>
        <groupId>io.zipkin.java</groupId>
        <artifactId>zipkin-autoconfigure-ui</artifactId>
        <version>2.9.4</version>
    </dependency>
```



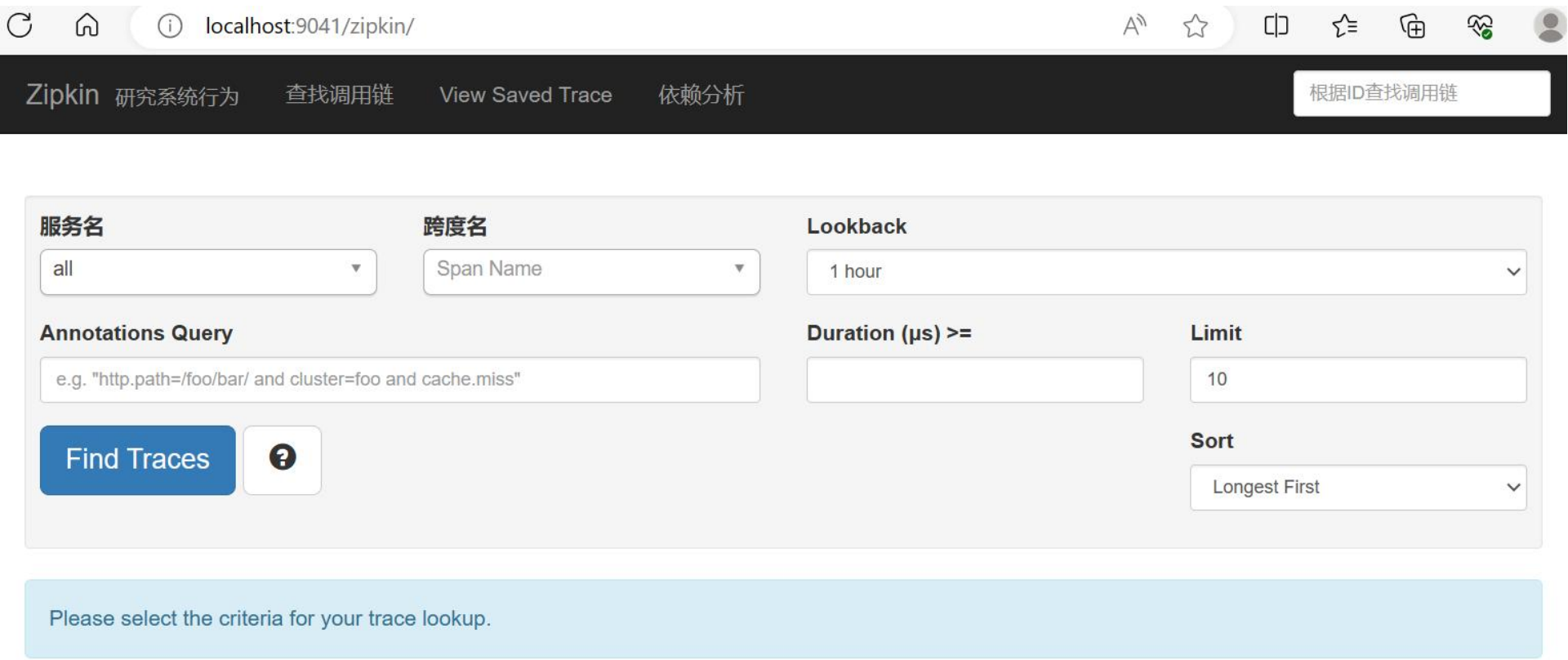
- 配置application.yml

```
server:
  port: 9041
spring:
  application:
    name: zipkin-server
#去除控制台异常
management:
  metrics:
    web:
      server:
        auto-time-requests: false
```

- 编写启动类

```
/** * Zipkin服务 */
@SpringBootApplication
@EnableZipkinServer
public class ZipkinApplication {
    public static void main(String[] args) {
        SpringApplication.run(ZipkinApplication.class,args);
```

zipkin服务端界面



注册Zipkin服务

- 修改网关及服务消费者，生产者所有微服务，让它们注册到Zipkin中，以便让这些微服务产生的日志能被Zipkin收集到！

- 导入依赖：

```
<!--sleuth-->
    <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>
```

注册Zipkin服务

添加application.yml配置

```
spring:
  application:
    name: myshop-gateway
  zipkin:
    base-url: http://localhost:9041
    sender:
      type: web
  sleuth:
    sampler:
      probability: 1
```

localhost:8222/myshop-web/web/order?token=user

购票成功

localhost:9041/zipkin/?serviceName=myshop-gateway&spanName=all&lookback=3600000...

Zipkin 研究系统行为 查找调用链 View Saved Trace 依赖分析

Service Name

myshop-gateway

Span Name

all

Lookback

1 hour

Annotations Query

e.g. "http.path=/foo/bar/ and cluster=foo and cache.miss"

Duration (μs) >=

Limit

10

Find Traces

Sort

Long

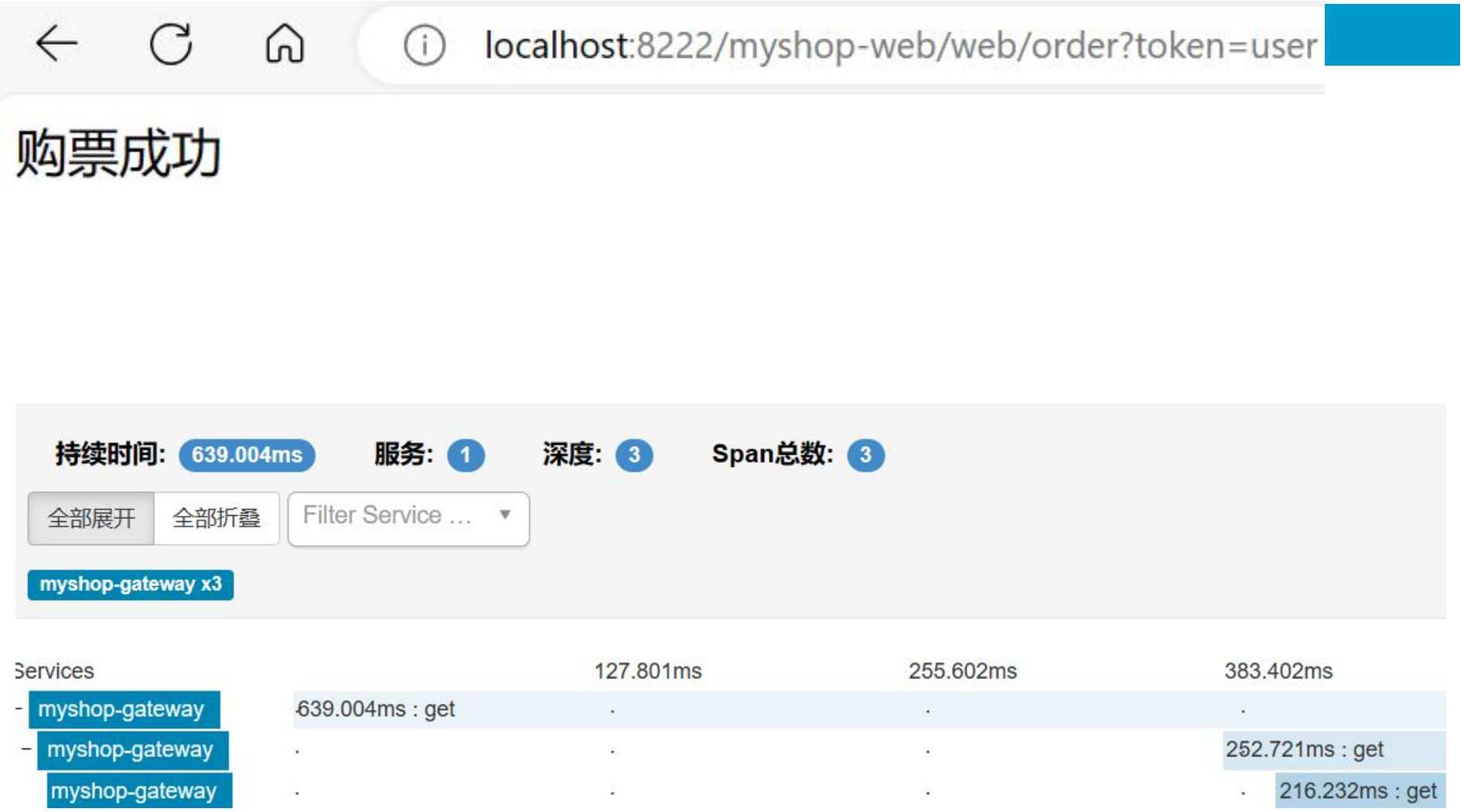
Showing: 1 of 1

Services: myshop-gateway

639.004ms 3 spans

myshop-gateway 100%

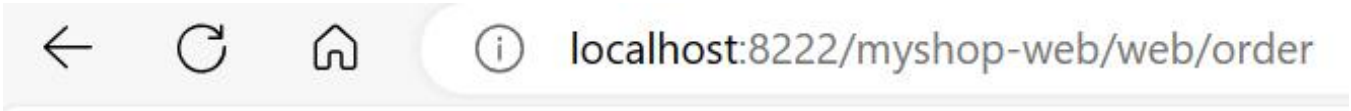
myshop-gateway x3 639ms


localhost:8222/myshop-web/web/order?token=user
购票成功
持续时间: 639.004ms
服务: 1
深度: 3
Span总数: 3
全部展开
全部折叠
Filter Service …
myshop-gateway x3
Services
127.801ms
255.602ms
383.402ms
myshop-gateway
639.004ms : get
myshop-gateway
252.721ms : get
myshop-gateway
216.232ms : get

- 用户微服务和购票微服务分别导入依赖

```
<!--sleuth-->
    <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>
```

- 用户微服务和购票微服务添加配置



购票成功

Service Name

myshop-gateway

Span Name

all

Lookback

1 hour

Annotations Query

e.g. "http.path=/foo/bar/ and cluster=foo and cache.miss"

Duration (μs) >=

Limit

10

Find Traces

Sort

Newest First

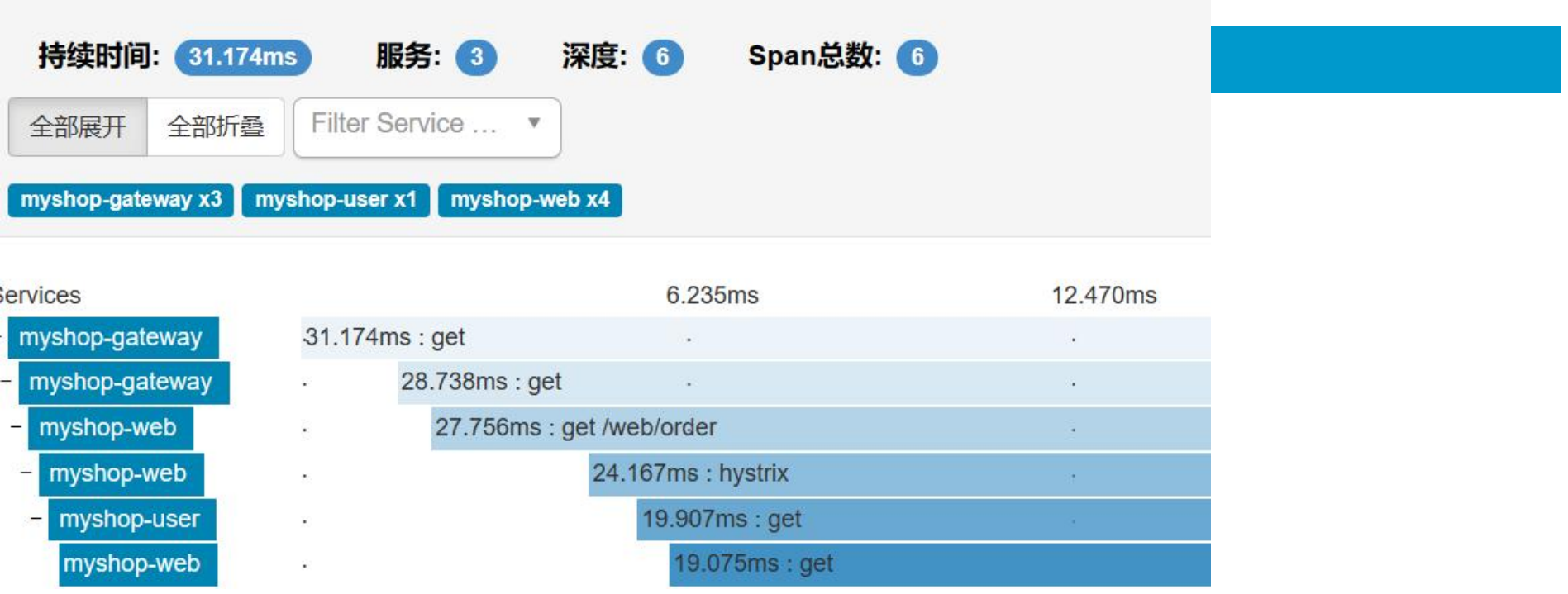
Showing: 10 of 10

Services: myshop-gateway

31.174ms 6 spans

myshop-gateway 100%

myshop-gateway x3 31ms | myshop-user x1 19ms | myshop-web x4 27ms



myshop-web.get /web/order: 27.756ms

AKA: myshop-gateway,myshop-web

Date Time	Relative Time	Annotation	Address
2023/10/31 21:27:21	2.117ms	Client Send	192.168.221.1 (myshop-gateway)
2023/10/31 21:27:21	3.067ms	Server Receive	192.168.221.1 (myshop-web)
2023/10/31 21:27:21	29.873ms	Client Receive	192.168.221.1 (myshop-gateway)
2023/10/31 21:27:21	29.926ms	Server Send	192.168.221.1 (myshop-web)

Key	Value
Client Address	[::1]:59880


Zipkin
研究系统行为
查找调用链
View Saved Trace
依赖分析
开始时间
2023-10-30
21:37
结束时间
2023-10-31
21:37
myshop-gateway
myshop-web
myshop-user