



# ServiceComb 介绍

姜宁

# 关于我



Apache CXF



# 议题

- ServiceComb是什么？
- ServiceComb包含了哪些组件？
- ServiceComb社区介绍？

# 企业应用微服务开发面临的问题

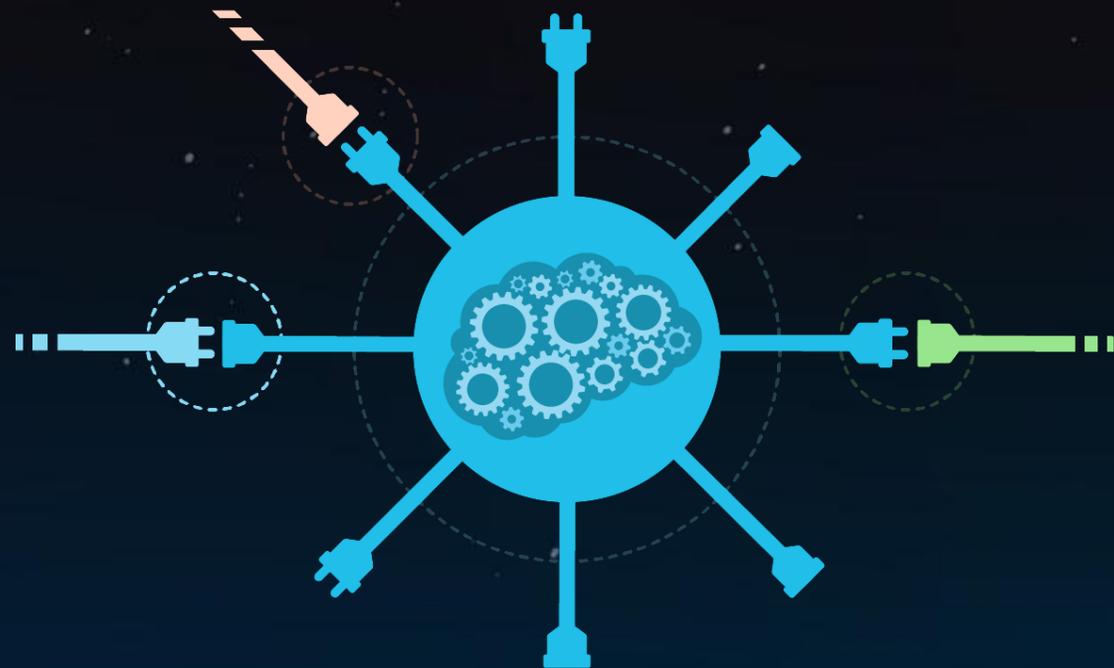
- 企业业务逻辑复杂，如何进行跨团队协作开发？
- 大量老旧系统代码，如何支持其服务化改造？
- 云化应用面临的监控已经分布调用追踪问题？

# ServiceComb

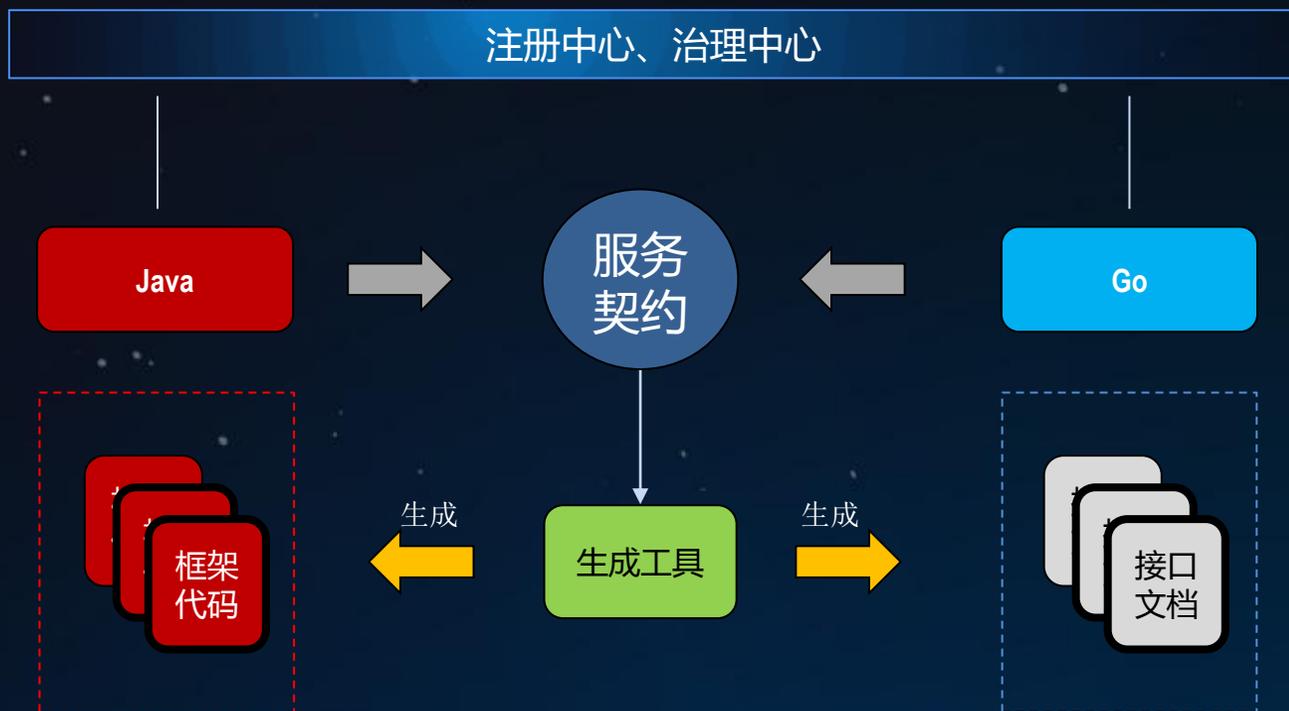


# 为什么需要服务契约

- 作为服务消费者
  - 需要明确知道如何调用服务？
  - 需要知道服务调用参数有哪些？
  - 需要知道传输的内容是什么？
- 作为服务提供方
  - 提供详细描述文档很难
  - 提供和维护标准的SDK也很难
  - 提供用户支持也不太方便



# 服务契约定义



## API First & Code First

- 面向接口契约而不是逻辑
- 解耦服务提供者和消费者的开发顺序
- 契约定义为语言中立
- 规范化系统接口，让实现与文档的同步
- 通过工具简化整个过程

# 支持多种Rest服务实现方式

## SpringMVC

Provider service:

```
import io.servicecomb.*;
import org.springframework.*;

@RestSchema(schemaId = "helloworld")
@RequestMapping(path = "/helloworld", produces =
public class HelloWorldProvider implements Hell

    @RequestMapping(path = "/sayHello", method =
    public String sayHello(@RequestParam("name")
        return "Hello " + name;
    }
}
```

Consumer service:

```
import io.servicecomb.*;
import org.springframework.*;

@Component
public class HelloWorldConsumer {
    private static RestTemplate restTemplate;

    public static void main(String[] args)
String result= restTemplate.getForObject
}
}
```

## JAXRS

Provider service:

```
import io.servicecomb.*;
import javax.ws.rs.*;

@RestSchema(schemaId = "helloworld")
@Path("/helloworld")
@Produces(MediaType.APPLICATION_JSON)
public class HelloWorldProvider implements HelloWorld {
    @Path("/sayHello")
    @GET
    public String sayHello(@PathParam("name") String name) {
        return "Hello " + name;
    }
}
```

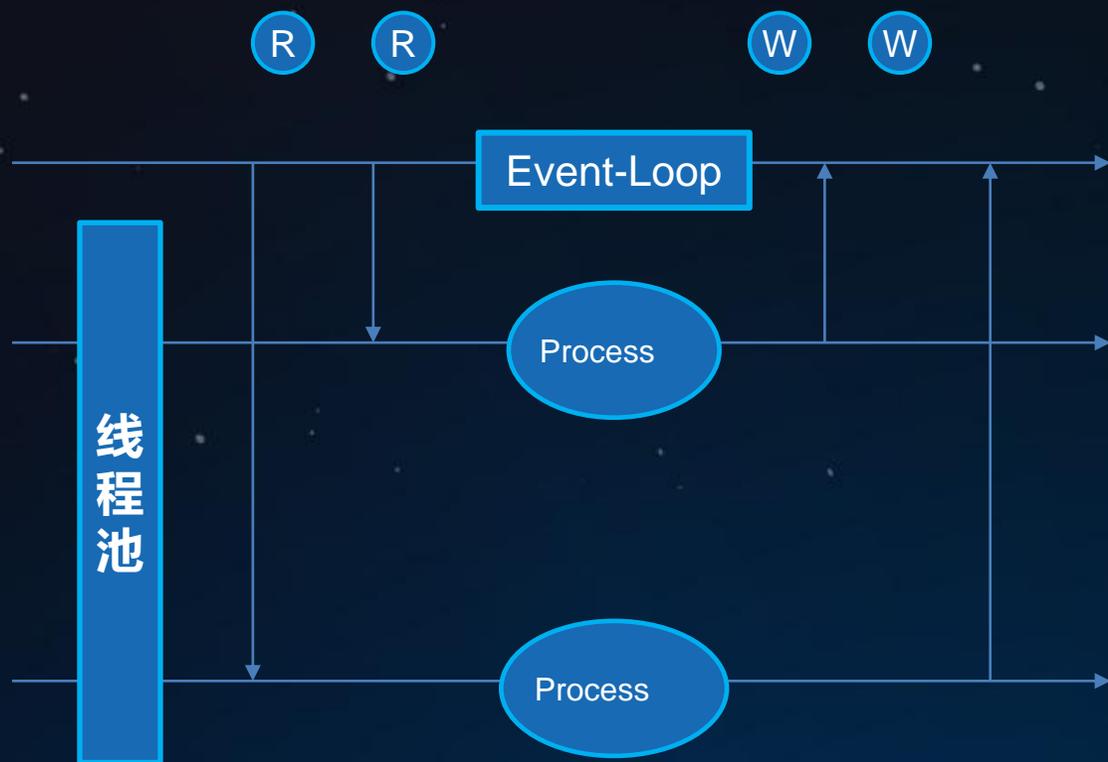
Consumer service:

```
import io.servicecomb.*;
import org.springframework.*;

@Component
public class HelloWorldConsumer {
    private static RestTemplate restTemplate = RestTemplateBuilder.create();

    public static void main(String[] args) {
String result= restTemplate.getForObject("cse://jaxrs/helloworld/syaHello?name={name}",String.class,"Tank");
}
}
```

# 异步事件操作保证系统性能



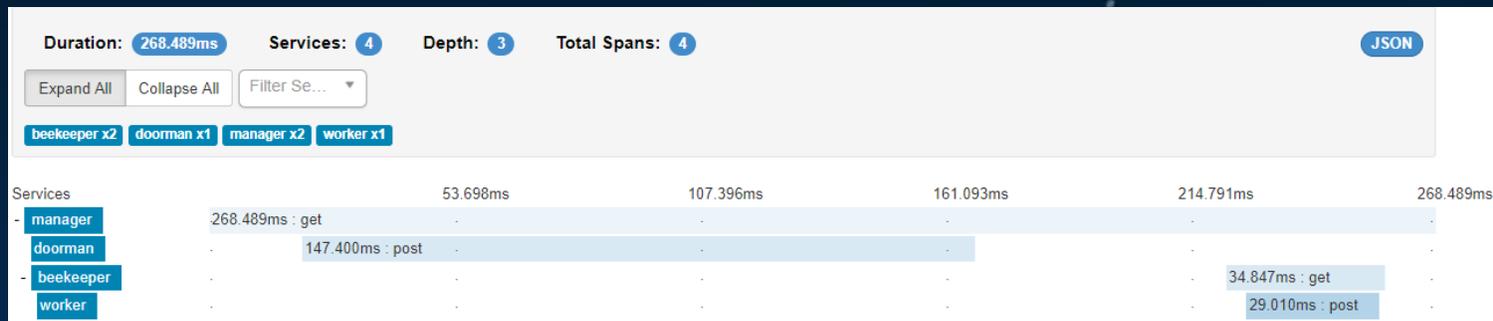
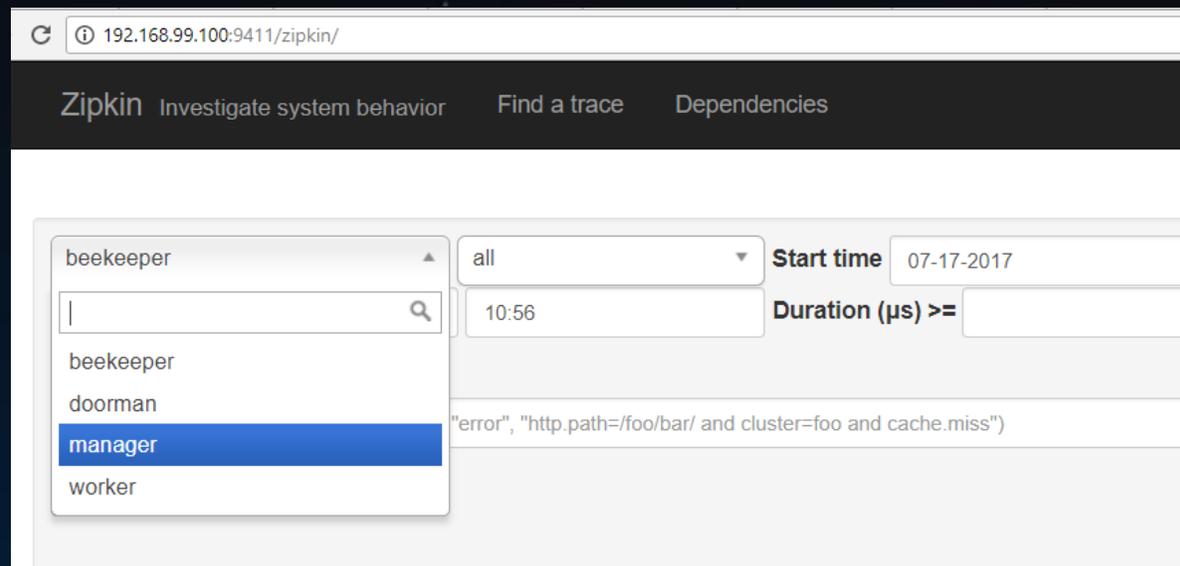
- 框架以事件驱动为基础
- 异步操作提升应用处理效率
- 支持Restful以及RPC

\*各主流微服务框架性能对比结果：\*

机型	协议	规格(CPU/内存/网卡)	EDAS	实例数	报文大小	调用线程数	TPS	时延(ms)	服
VM	RPC	8U16G1G	EDAS	1	1k	100	58431	1.73	2
物理机	RPC	24U64G1G	EDAS	1	1k	100	97586	1.01	3
机型	协议	规格(CPU/内存/网卡)	CSE SDK	实例数	报文大小	调用线程数	TPS	时延(ms)	
VM	REST	8U16G1G	CSE SDK	1	1k	100	70669	1.414	
物理机	REST	24U64G1G	CSE SDK	1	1k	100	107126	0.933	
机型	协议	规格(CPU/内存/网卡)	Tars	实例数	报文大小	调用线程数	TPS	时延(ms)	服
VM	RPC?	8U16G1G	Tars	1	1k	100	75386	1.32	21
机型	协议	规格(CPU/内存/网卡)	SpringCloud	实例数	报文大小	调用线程数	TPS	时延(ms)	
VM	REST	8U16G1G	SpringCloud	1	1k	100	22507	4.70	
物理机	REST	24U64G1G	SpringCloud	1	1k	100	51382	2.84	

# 分布式调用追踪

- 支持标准追踪协议
  - Zipkin
  - 支持@span扩展事务追踪
- 提供调用追踪扩展
  - 可以配置自定义的追踪处理器
  - 支持适配自定义追踪格式



# ServiceComb代码库信息

<http://github.com/ServiceComb>

– **Java-chassis**

- Java版的微服务框架

– **service-center**

- 基于go语言的服务中心

– **ServiceComb-Saga**

- 长时间运行事务处理框架

# ServiceComb组件的新特性

- Service-Center 0.2.0
  - 提供了管理控制台，对注册的服务实例提供管理功能
- Java-Chassis 0.3.0
  - 支持通过@Span注释的方式定义事务
  - 统一了Http请求处理，支持对Rest调用进行定制处理
- Saga 0.1.0
  - 提供了初步的Saga调用服务，支持顺序以及并发的调用处理



**WE WANT YOU!**

# 社区贡献

01/ 提交BUG

02/ 完善文档

03/ 参与邮件列表

04/ 提交Patch

# 如何参与到ServiceComb社区

- **线上：**

- 关注微信公众获取信息
- 加入微信群进行交流
- 通过邮件列表参与讨论
- 通过Github发起PR

- **线下：**

- 月度Meetup
- 不定期沙龙探讨





# ServiceComb

让云原生开发更简单

Github : <https://github.com/ServiceComb>

官网 : <https://www.servicecomb.io>

