ServiceComb 微服务框架

杨波

ServiceComb 社区 / 华为开源软件能力中心

ServiceComb微服务解决方案

ServiceComb子系统

服务中心

ServiceCenter 是一个使用Go构造的、建立在etcd存储上的高性能、高可用服务中心。

Java微服务SDK

Java Chassis是一个由编程模型、运行模型、通信模型和服务契约四个部分组成的微服务框架。

Saga
分布式事务解决方案

Saga 是一个微服务数据一致性解决方案。

Go微服务SDK

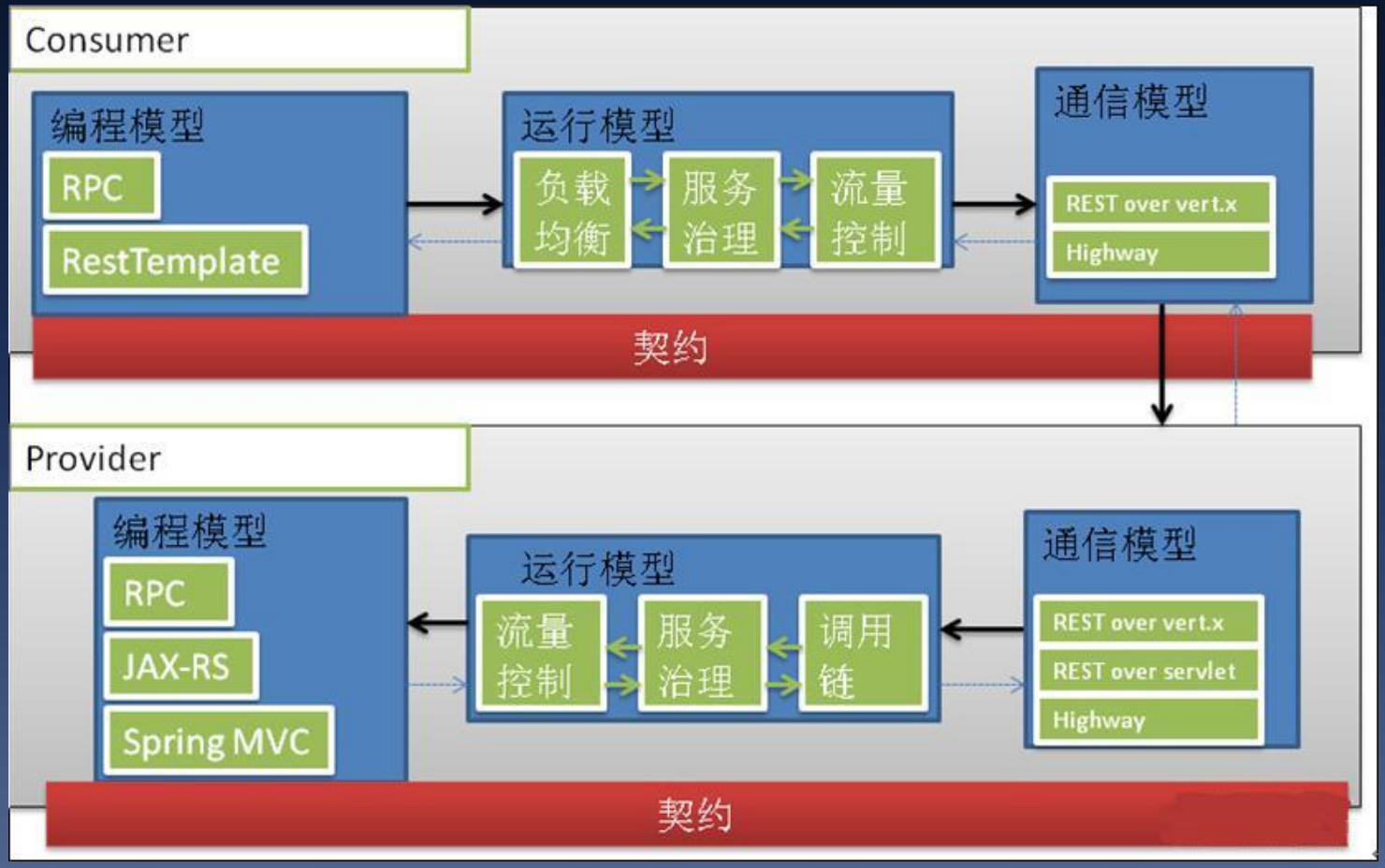
ServiceMesh

ServiceComb的开放性设计

•Provider与Consumer具有完全一致的开发体验

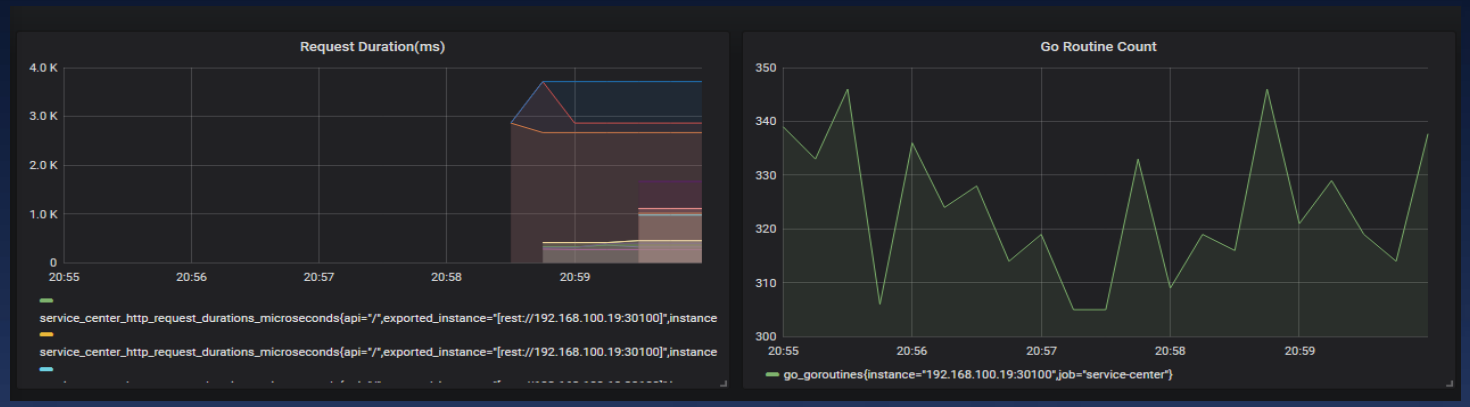
•通信模型与编程模型隔离，适应不同业务场景需求

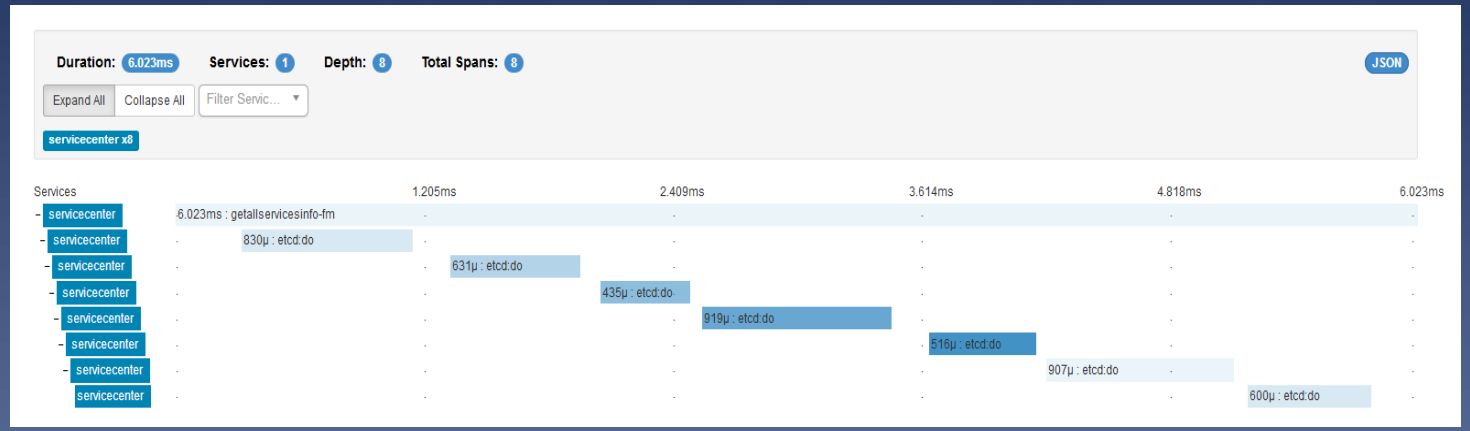
•运行模型支持灵活扩展，便于对接外部系统



特性介绍： Service Center

- 服务注册/服务发现
- 微服务元数据与依赖管理
- 支持监听服务实例状态变更
- 服务实例管理
- 高性能、高可靠
- 性能监控与调用追踪



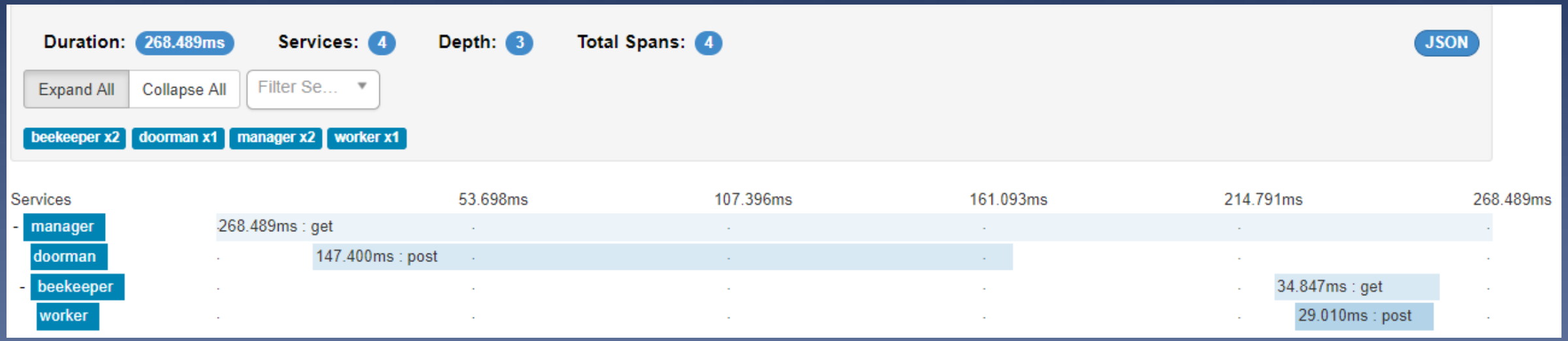


特性介绍： Java-Chassis

分布式追踪

- 支持标准的分布式调用追踪Zipkin
- 支持通过@span扩展自定义追踪
- 通过调用链支持自定义追踪扩展
- 支持Skywalking

```
@Override
@Span
public double calculate(double height, double weight) {
  if (height <= 0 || weight <= 0) {
    throw new IllegalArgumentException("Arguments must be
above 0");
  }
  double heightInMeter = height / 100;
  double bmi = weight / (heightInMeter * heightInMeter);
  return roundToOnePrecision(bmi);
}
```



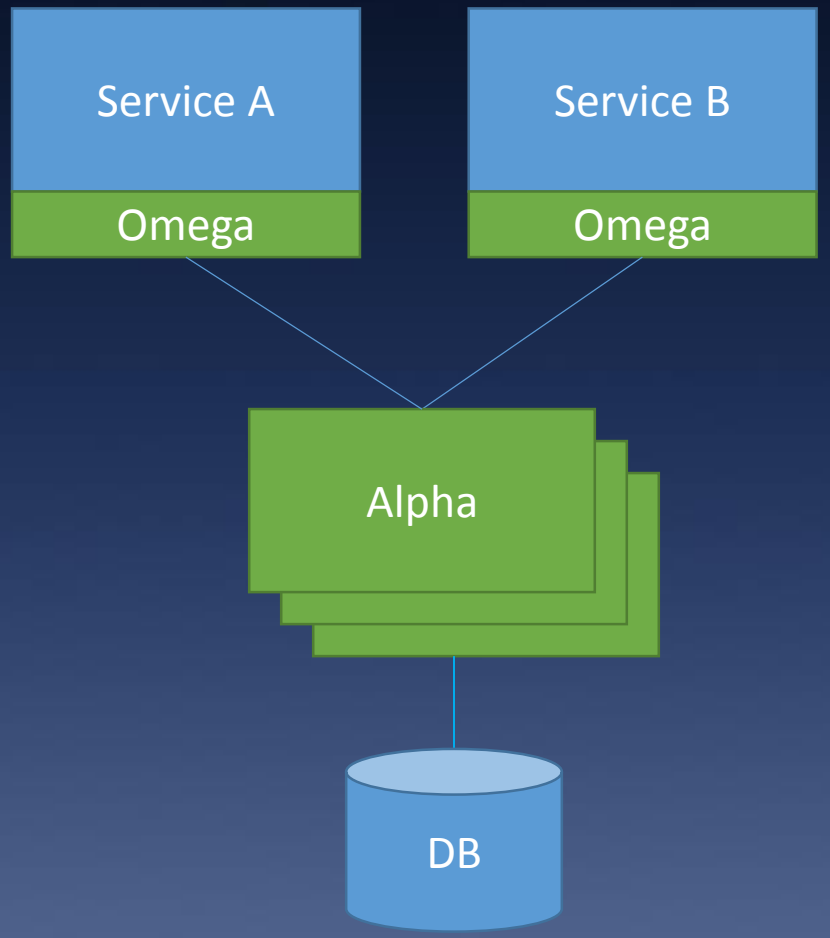
特性介绍： Saga

基于Pack模型的新构架

由 Alpha 和 Omega组成

•Alpha充当协调者的角色，主要负责对事务的事件进行持久化存储以及协调子事务的状态，使其最终得以与全局事务的状态保持一致，即保证事务中的子事务**全执行，或全不执行**。

•Omega是用户程序侧代理，负责对网络请求进行拦截并向Alpha上报事务事件，并在异常情况下根据Alpha下发的指令执行相应的**补偿**或**重试**操作。



特性介绍： Saga

```
@SpringBootApplication
@EnableOmega
public class Application {
  public static void main(String[] args) {
    SpringApplication.run(Application.class, args);
  }
}
```

```
@SagaStart
@PostMapping("/booking/{name}/{rooms}/{cars}")
public String order(@PathVariable String name,
@PathVariable Integer rooms, @PathVariable Integer
cars) {
    template.postForEntity(
        carServiceUrl + "/order/{name}/{cars}",
        null, String.class, name, cars);
```

Pack模型

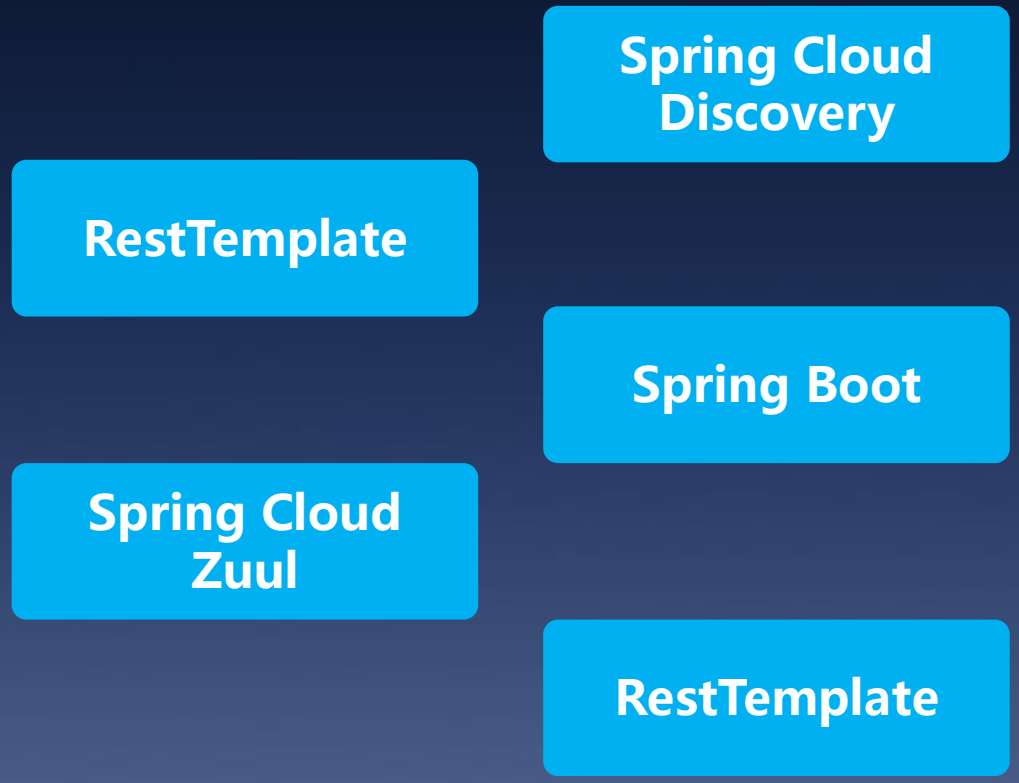
```
@Compensable(compensationMethod = "cancel")
 void order(CarBooking booking) {
   booking.confirm();
   bookings.put(booking.getId(), booking);
 }

 void cancel(CarBooking booking) {
   Integer id = booking.getId();
   if (bookings.containsKey(id)) {
     bookings.get(id).cancel();
   }
 }
```

ServiceComb与Spring Cloud的集成

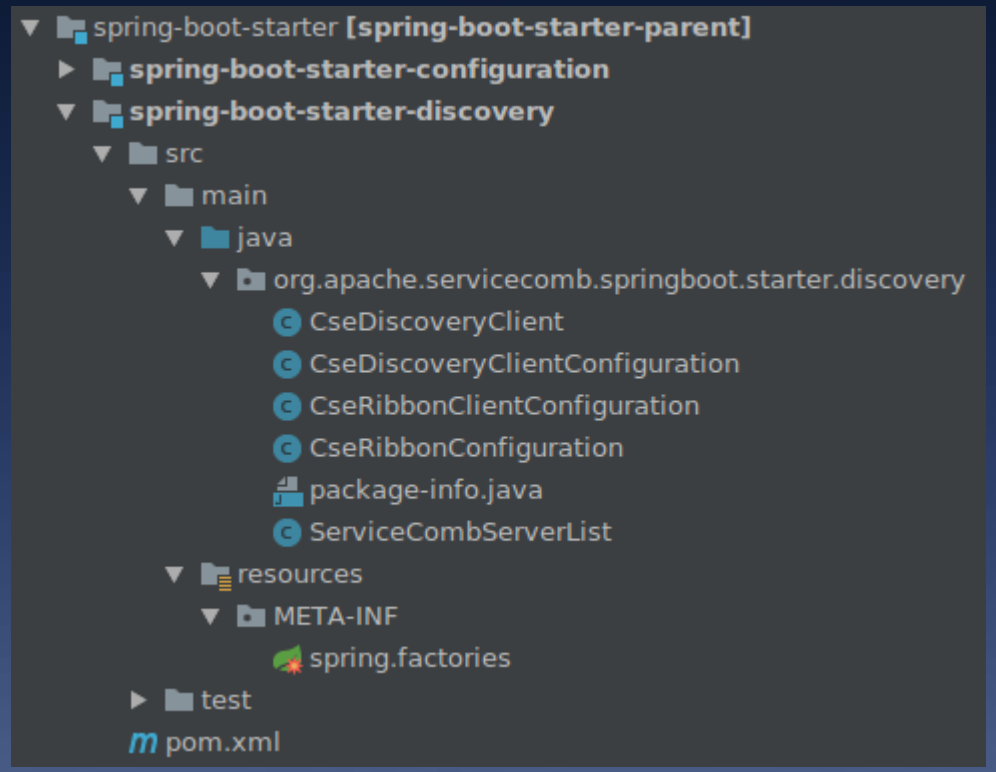
- Java-Chassis与Saga都对Spring Boot进行了深度集成

- 通过Starter形式，Java-Chassis提供了Spring Cloud Discovery, Zuul等组件的集成

- 根据需要，后期会继续集成更多的组件



集成Spring Cloud组件



服务发现

- Spring Cloud通过@EnableDiscoveryClient来支持服务发现
- 需要实现DiscoveryClient接口，提供获取服务实例的实现
- 通过@Configuration来设置相关服务实例
- 设置完DiscoveryClient后，可无缝集成到Spring Cloud Zuul

加入我们

- 线上
  - 关注ServiceComb微信小助手
  - 在官网获取快速入门以及相关教程
  - 加入微信群与开发人员进行交流
  - 通过邮件列表讨论 dev@servicecomb.apache.org
  - Github发起PR
- 线下
  - 月度Meetup
  - 不定期的技术沙龙研讨