ServiceComb

消费者云CSE微服务实践

李林锋

ServiceComb

关于我

- 9年电信软件平台中间件开发、设计、架构经验
- 精通Netty、Mina等网络通信框架
- 华为软件 短信/彩信/WAP网关平台SGP、API开放集成网关API Fabric首席设计师
- 华为软件分布式服务框架DSF首席设计师
- 《Netty权威指南》、《分布式服务框架原理与实践》作者

微博、微信：Nettying

公众号：Netty之家

ServiceComb

议题

- 华为消费者云业务简介
- 微服务框架技术选型
- CSE在消费者云业务的实践

华为消费者云业务简介

华为消费者云业务包括 华为应用市场、华为视频、华为钱包、华为运动健康等服务，为华为和荣耀手机提供精品云服务，提升用户体验

ServiceComb

微服务框架技术选型-业务服务化目标

- 系统解耦，功能内聚，提升需求交付效率：通过业务的拆分和解耦，让系统敏捷起来，快速、小批量的交付价值需求，提升业务的交付效率
- 践行API First：通过服务化，让服务提供者和消费者之间通过微服务API建立契约，利用Swagger OpenAPI规范，最终将微服务API规范化、标准化、在线化。系统从传统单体应用的黑盒调用（本地Java方法调用）转变成透明的API契约调用
- 服务自治：通过在线的微服务治理结合云平台，可以实现微服务的弹性伸缩、故障自动迁移、降级熔断等，保障微服务的运行质量，提升业务SLA
- 建立服务化团队：随着业务的不断拆分，大的研发团队也会被拆分成2-Pizza Team，微服务团队由3-5人组成，负责整个微服务的设计、开发、测试、部署运维和治理，通过全功能团队的建设，让业务真正敏捷起来

微服务框架技术选型-支持多语言

- 尽管现在以Java和GO语言为主，但是从架构演进角度考虑，未来会根据消费者业务自身的特点引入更适合的语言
- 服务框架不要绑定具体的语言实现，例如内部通信协议使用某种语言特定的序列化机制、发布泛型、抽象接口等

微服务框架技术选型-灵活和轻量级架构

- 当前业务服务端都是非Web应用，所以不需要运行在Web容器中，需要类似Main函数可以直接拉起来的Standalone模式
- 服务框架要足够轻量级，可以按需加载类库，防止与当前业务的三方库发生冲突
- 启停速度要快（秒级弹性伸缩）、资源占用要合理

ServiceComb

微服务框架技术选型-微服务安全

- 有些业务场景对微服务调用安全要求较高，需要微服务框架支持SSL传输、API鉴权和认证等
- 对于一些敏感信息，例如用户账号、金额等，在记录日志等落盘和采集时需要做脱敏处理、资源占用要合理
- 敏感运维操作，需要记录安全日志，例如服务上线和下线、服务的流控阈值修改等

ServiceComb

微服务框架技术选型-服务治理能力

- 服务框架不能只单单解决分布式RPC调用、服务注册&发现和路由问题，更重要的是业务微服务上线之后，需要提供实用和丰富的在线治理能力
- 流量控制、并发控制、超时控制、服务降级、服务熔断、路由权重调整...
- 常用的服务治理能力要内置到服务框架中，业务领域强相关、非通用能力可以通过扩展点实现

微服务框架技术选型-易集成

- 当前业务使用Spring MVC等传统的单体架构，希望可以较平滑、低成本的迁移到微服务架构上：
  - ✓ 从业务接受度上，希望不要翻天覆地的改变业务开发习惯，最好能够兼容原Spring MVC开发模式
  - ✓ 从集成角度看，希望可以灵活的与Spring Boot等框架集成

ServiceComb

微服务框架技术选型-高性能、低时延

- 硬件成本已经是白菜价，软件性能不重要？
  - ✓ 消费者云业务服务集群规模大，单点的性能提升能够带来巨大收益
  - ✓ 从用户体验看，端到端时延非常重要，分布式之后带来的时延增加，是一个很大的挑战
- 不是所有业务都有苛刻的性能需求，不同业务对性能的诉求不同，可以按需选择协议和传输方式，服务与传输协议、序列化方式解耦

ServiceComb

微服务框架技术选型-成熟

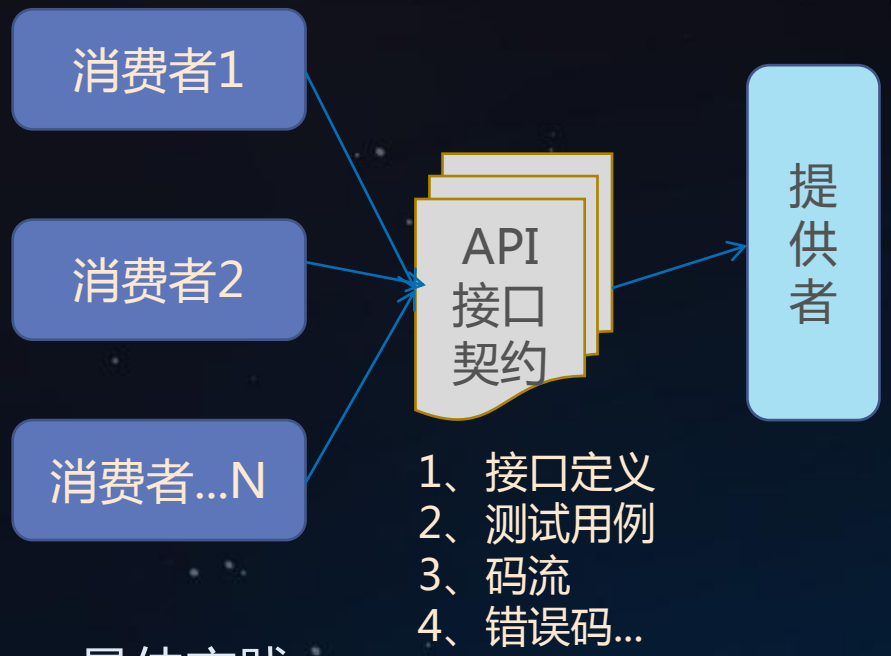
- 微服务框架采用的技术应该是经过验证、业界主流的技术，例如网络传输采用Netty
- 微服务框架本身要成熟，经过不同业务、较长时间的验证，商用发布的特性要稳定
- 无论是社区开源版本，还是购买的商用软件，或者自己构建，技术支持和保障一定要到位

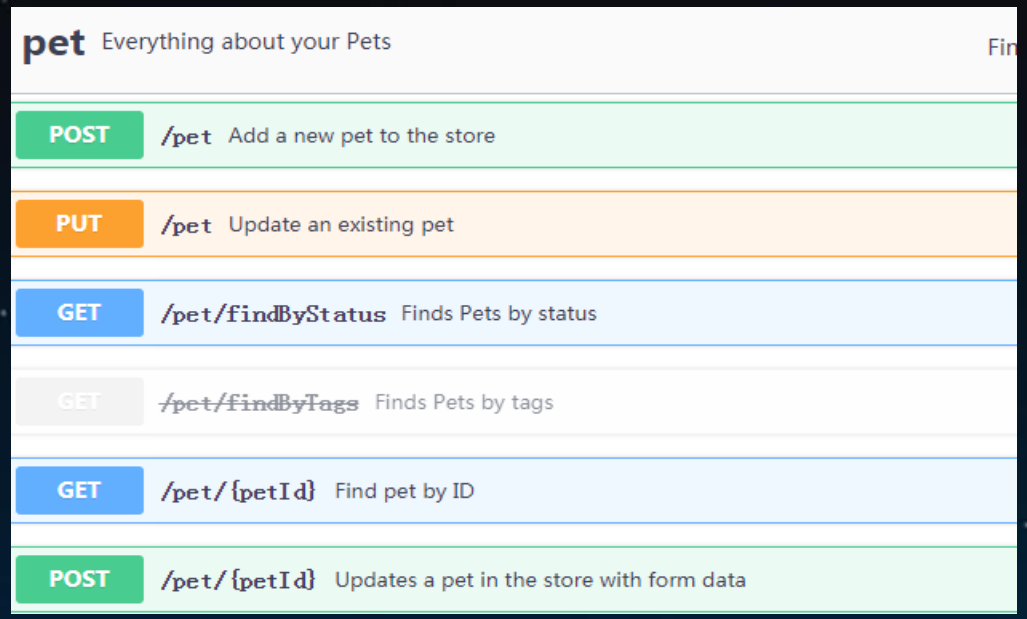
微服务框架技术选型-结果

- 为什么业务最终选择CSE作为微服务框架？

- ✓ 无论是华为内部的DSF，还是开源的Netflix、Spring Cloud等，都无法完全满足业务的选型诉求，而CSE则能够很好的满足我们的需求
- ✓ 仔细阅读了CSE的主要模块代码，包括网络通信、线程调度模型等，代码质量非常高，对细节的把握比较好
- ✓ 选型试用时，大家对CSE的接受度比较高，使用CSE改造已有的Spring MVC代码相对较容易些
- ✓ 华为内部的平台，无论是新需求接纳，还是技术支撑，各方面保障都比较给力
- ✓ 天生支持Docker容器与华为公有云，降低业务云化成本

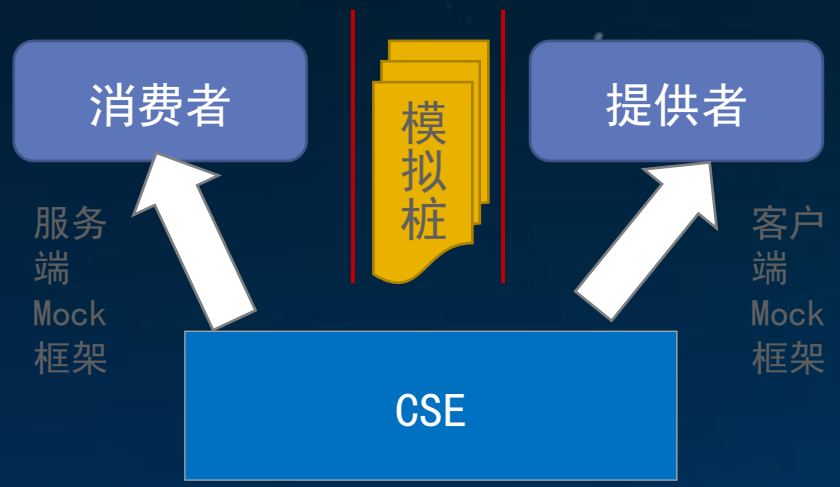
CSE在消费者云业务的实践-API First





➢ 最佳实践

- ✓ 无论Rest API，或者RPC Highway API，统一使用Swagger YAML定义API
- ✓ 服务端和客户端都基于API定义，通过CSE提供的工具生成不同语言的类库，客户端可以不导入服务端的类库定义，双方互相解耦
- ✓ 始终以在线的API定义为准，防止服务端私自修改接口代码（例如增减字段、修改字段类型）
- ✓ 每日微服务流水线构建，及时发现接口不兼容问题



CSE在消费者云业务的实践-Spring MVC模式开发微服务

1、配置spring mvc依赖

```
<dependency>
    <groupId>io.servicecomb</groupId>
    <artifactId>provider-springmvc</artifactId>
</dependency>
```

2、注解Service

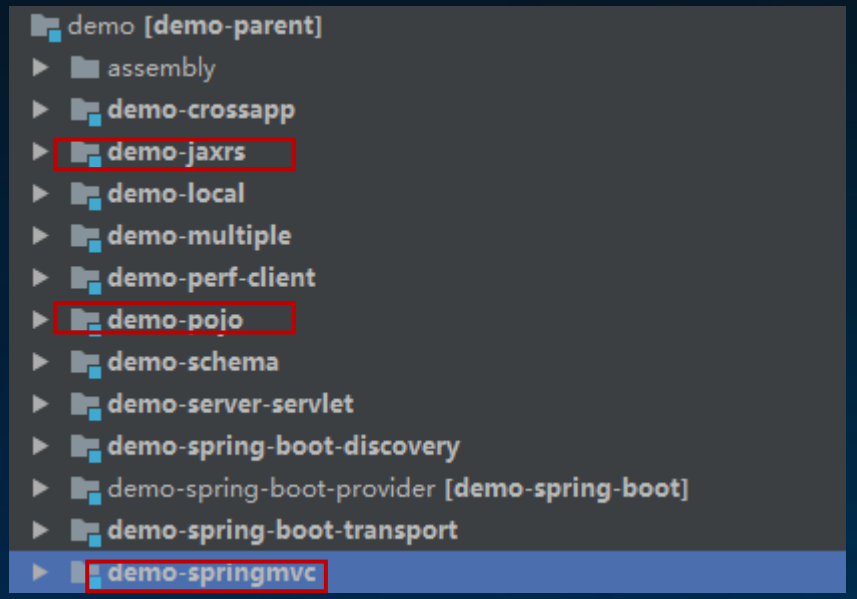
```
@RestSchema(schemaId = "controller")
@RequestMapping(path = "/controller", produces = MediaType.APPLICATION_JSON)
public class ControllerImpl {
    @RequestMapping(path = "/add", method = RequestMethod.GET)
    public int add(@RequestParam("a") int a, @RequestParam("b") int b) { return a + b; }

    @RequestMapping(path = "/sayhello/{name}", method = RequestMethod.POST)
    public String sayHello(@PathVariable("name") String name) { return "hello " + name; }
```

3、发布Service

```
APPLICATION_ID: springmvctest
service_description:
  name: springmvc
  version: 0.0.2
cse:
```

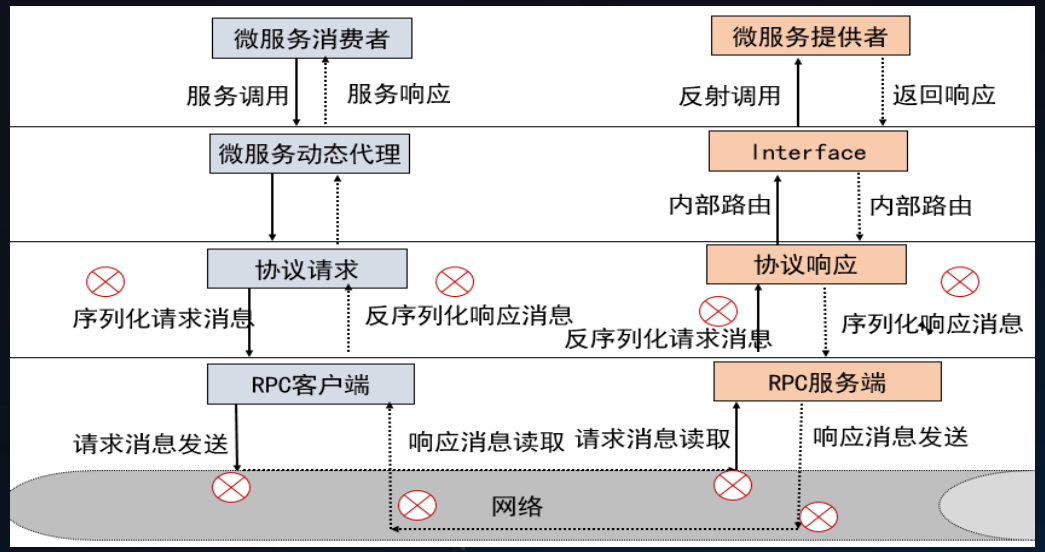
优势：除了Spring MVC开发模式，同时还支持：

- JAX-RS
- 透明RPC

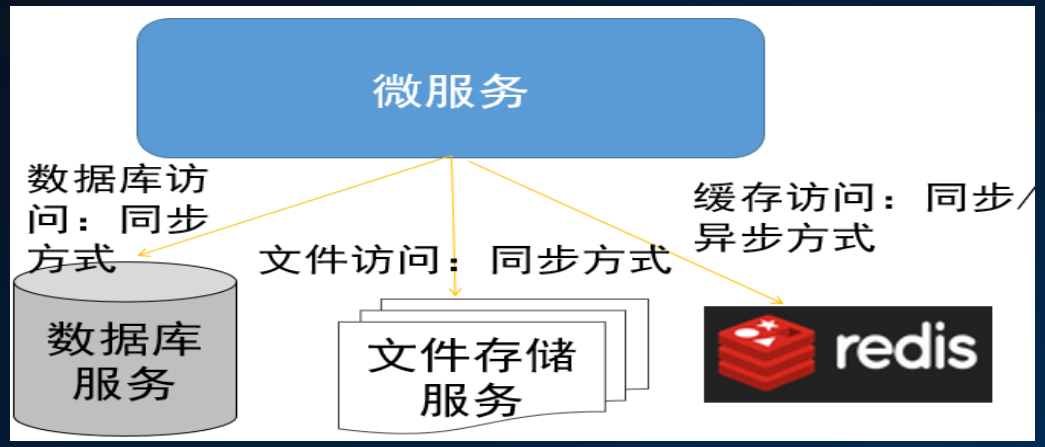


CSE在消费者云业务的实践-可靠性

1、分布式服务化本身引入的潜在故障点：

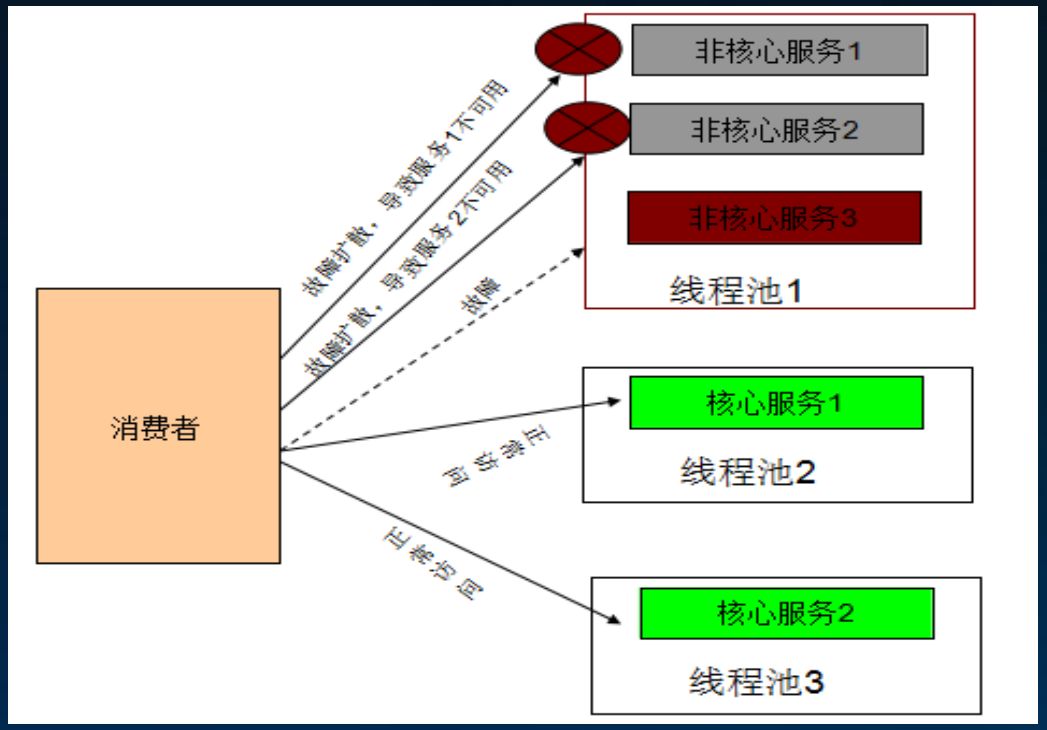


2、微服务第三方依赖潜在故障点：

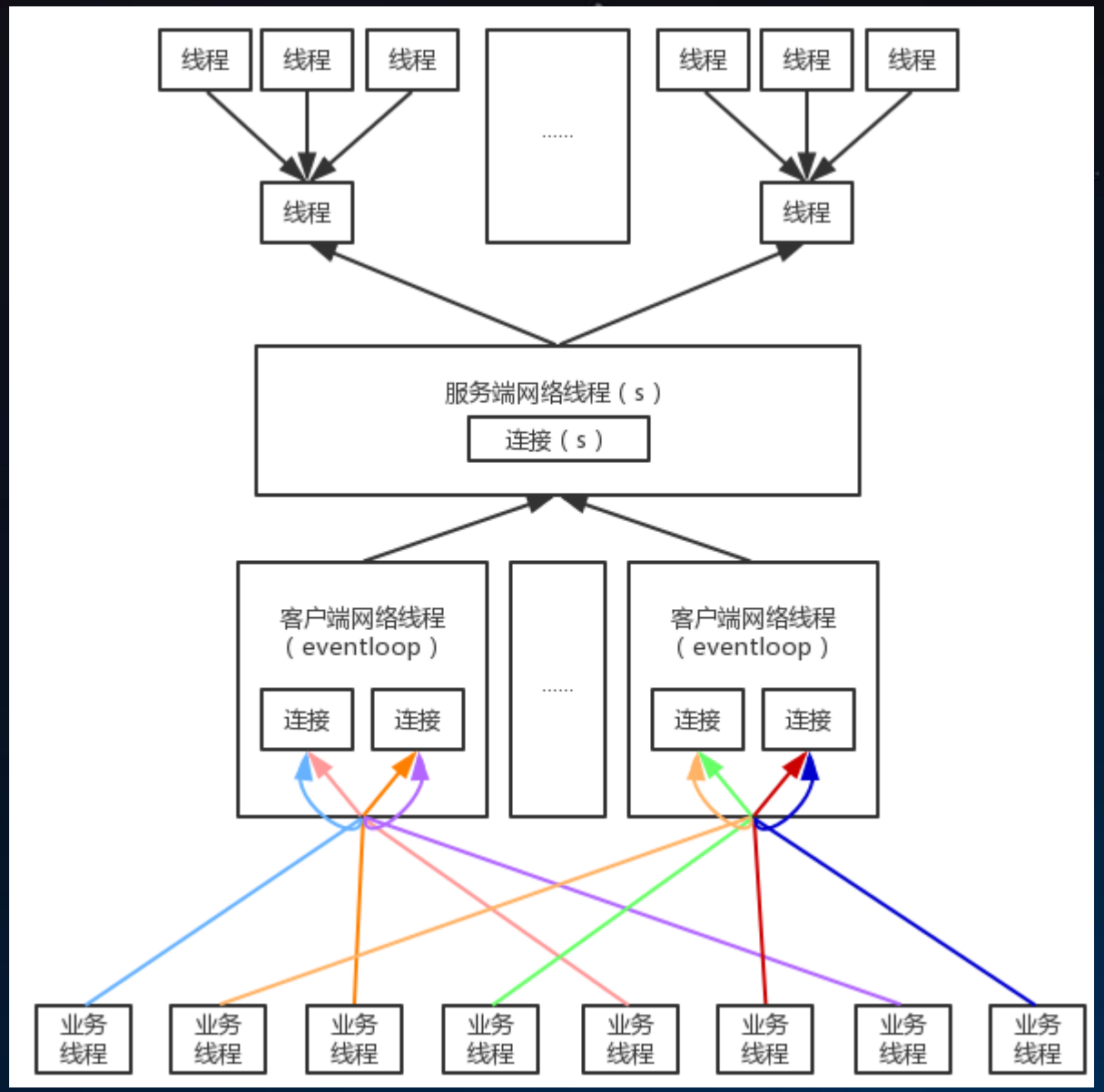


CSE的可靠性设计：

- 集群容错，自动路由
- 服务中心、配置中心无状态集群，宕机不影响已有业务
- 支持服务级故障隔离
- 支持多链路和链路级故障隔离
- 支持服务熔断和降级，以及第三方故障隔离（集成Hystrix）



CSE在消费者云业务的实践-服务调用高性能

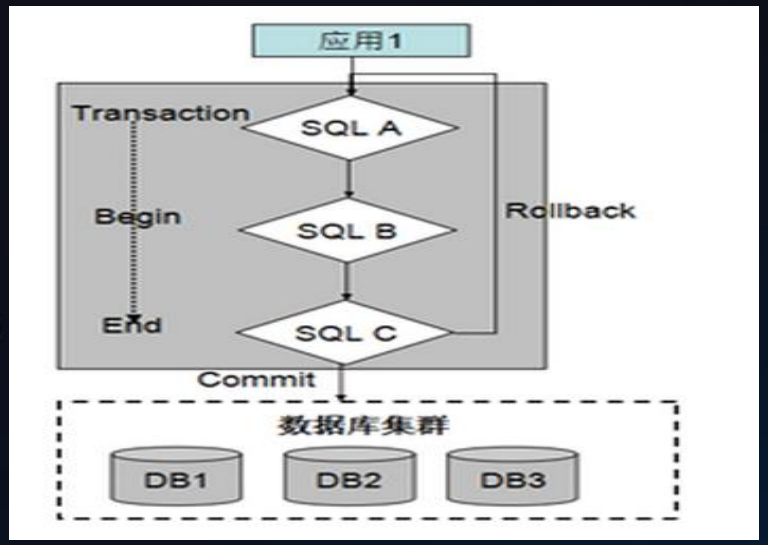


CSE的高性能设计：

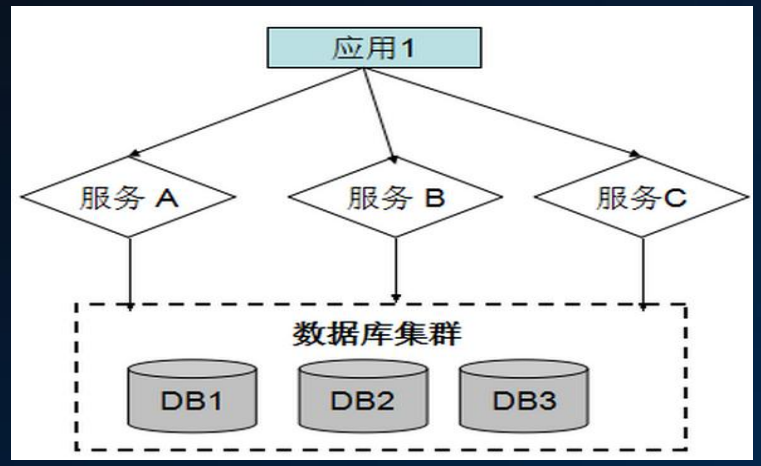
- 提供Rest和Highway RPC两种通信协议，满足不同业务场景
- 高性能的Rest：集成Vertx,底层基于Netty，性能比传统Servlet NIO性能高X倍
- Highway RPC：采用Netty + PB，既支持多语言，又保证高性能
- 高性能并发设计：线程绑定技术，网络I/O线程绑定后端的服务调度线程，最大限度减少锁竞争。采用连接池机制，重用已有的连接

CSE在消费者云业务的实践-分布式事务

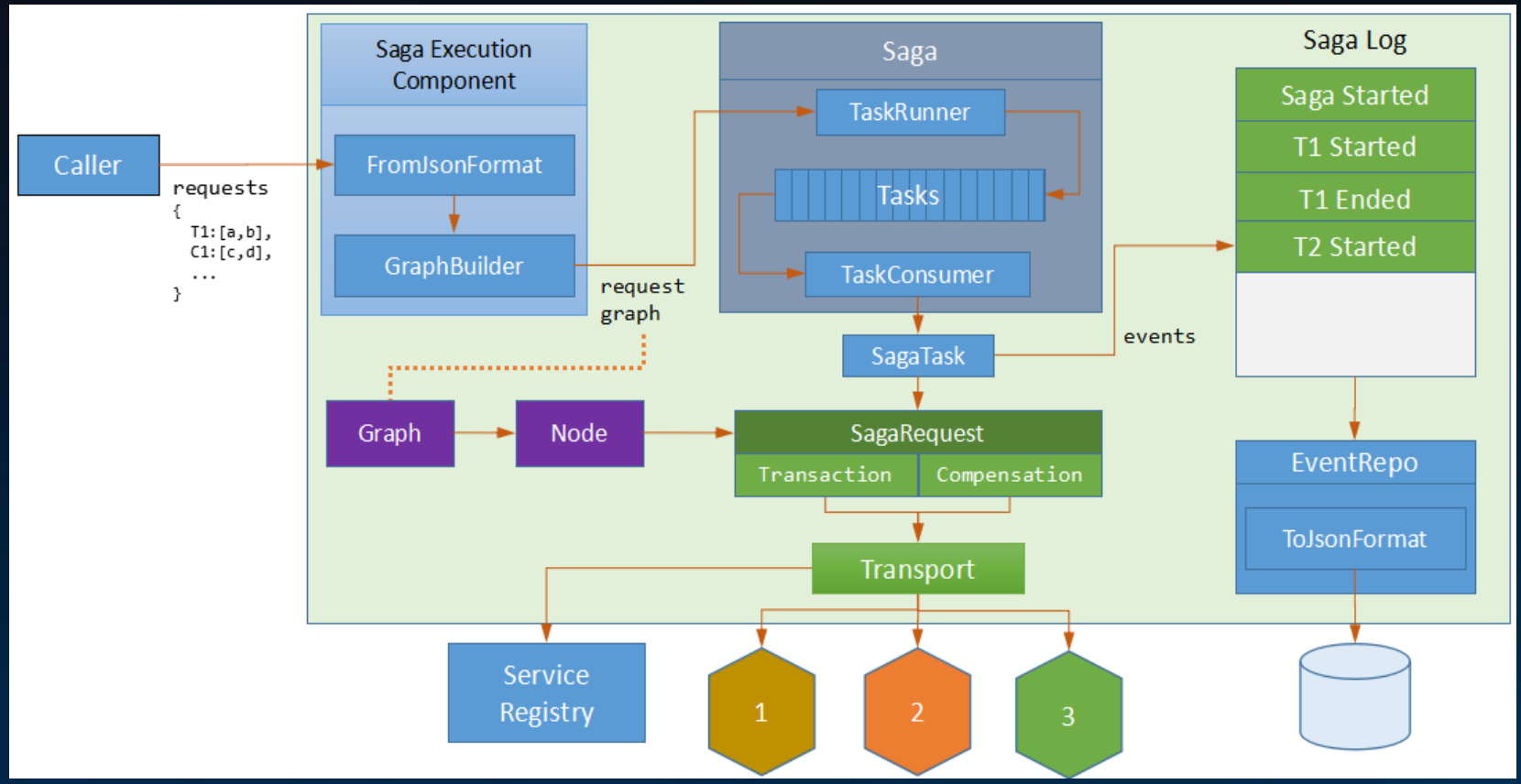
1、服务化之前本地事务：



2、服务化之后引入分布式事务：



解决方案：CSE提供数据最终一致性方案-**Saga**



CSE在消费者云业务的实践-服务治理能力

为什么需要服务治理：

- 随着业务的发展，服务越来越多，如何协调线上运行的各个服务，保障服务的SLA，对服务架构和运维人员是一个很大的挑战
- 线上业务发生故障时，需要对故障业务做服务降级、流量控制、流量迁移等，快速恢复业务
- 随着开发团队的不断扩大，服务的上线越来越随意，上线容易下线难，为了规范服务的上线和下线，在服务发布前，需要走服务预发布流程，由架构师或者项目经理对需要上线的服务做发布审核，审核通过的才能够上线

服务治理目的：满足服务上下线管控、保障微服务的高效、健康运行

部分服务治理配置项：

配置项	默认值
cse.loadbalance.NFLoadBalancerRuleClassName	com.netflix.loadbalancer.RoundRobinRule
cse.loadbalance.SessionStickinessRule.sessionTimeoutInSeconds	30
cse.loadbalance.SessionStickinessRule.successiveFailedTimes	5
cse.loadbalance.retryEnabled	FALSE
cse.loadbalance.retryOnNext	0
cse.loadbalance.retryOnSame	0
cse.loadbalance.isolation.enabled	FALSE
cse.loadbalance.isolation.enableRequestThreshold	20
cse.loadbalance.isolation.errorThresholdPercentage	20
cse.loadbalance.isolation.singleTestTime	10000
cse.loadbalance.transactionControl.policy	io.servicecomb.loadbalance.filter.SimpleTransactionControlFilter
cse.loadbalance.transactionControl.options	-

CSE在消费者云业务的实践-更多实践

调用链跟踪，与Zipkin集成，自定义调用链打点

集成Nginx做边缘服务（API接入）

本地开发与调试

分布式配置服务

HTTP/2

安全，HTTPS...

如何参与到CSE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/CSE

官网：https://www.CSE.io