QCon

www.qconferences.com

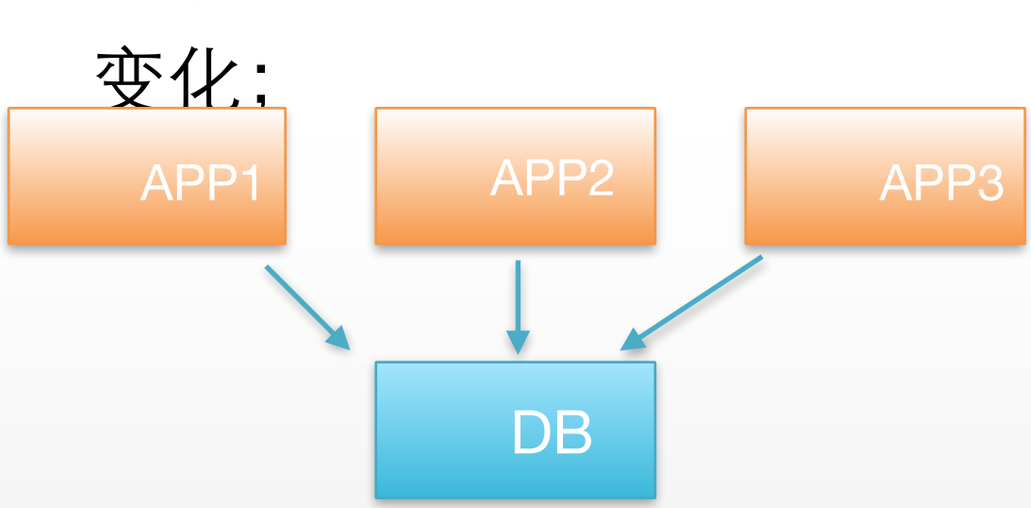
打造坚实的服务平台

——京东服务化实践

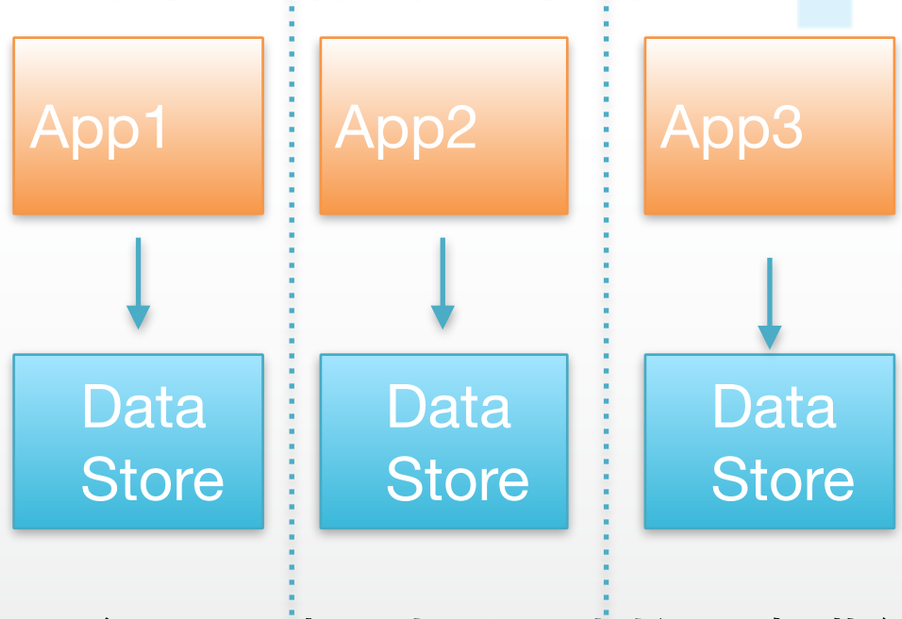
李鑫 lixininfo@jd.com

为何要服务化

- 系统规模随着业务的发展而增长，原有系统架构模式，逻辑过于耦合不再适应；
- 拆分后的子系统逻辑内聚，易于局部扩展；
- 子系统之间通过接口来进行交互，接口契约不变的情况下可独立变化；



交互通过DB来进行

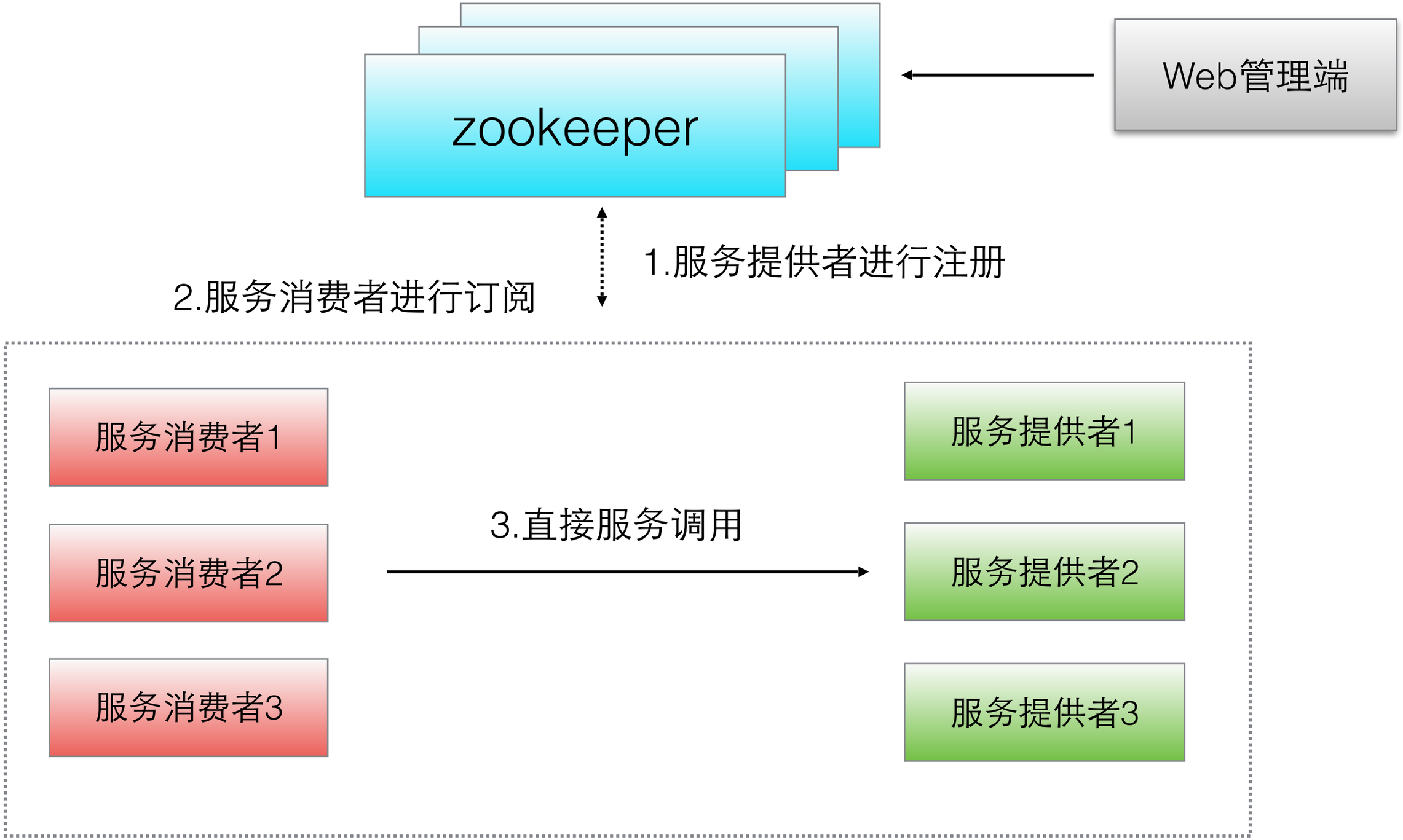


交互通过同步／异步接口来进行

为什么要打造服务平台

第一代服务框架

- 2012年初开始研发;
- zookeeper集群作为注册中心;
- base on开源的服务框架;



地址举例：WebService://172.17.3.18:20880/?interface=com.jd.arch.HelloService&group=pop&version=0.1

运营中暴露出的不足

1.客户端

- 许多逻辑放到客户端，推出新版本；有版本升级问题；
- watch时效问题；

2.注册中心

- zookeeper作为注册中心,功能定制扩展受限、性能有瓶颈；

3.服务治理

- 缺乏流控手段，大流量打爆线程池；
- 更改配置需重启，对运营不够友好；
- 缺乏调用监控，没有调用分析图表；

重装上阵！

新服务平台JSF

- 14年初开始研发;
- 自主研发以获得彻底的掌控力;
- 老版本运营经验支撑功能特性设计;
- 中文名：杰夫

Web管理端

管理数据库

注册中心数据库

Event Worker

监控数据库

index服务数据库

服务注册

服务寻址

配置管理

注册中心

Index Service

Monitor Service

定期发送性能统计数据

询问注册中心地址

注册／订阅

服务消费者1

服务消费者2

服务消费者3

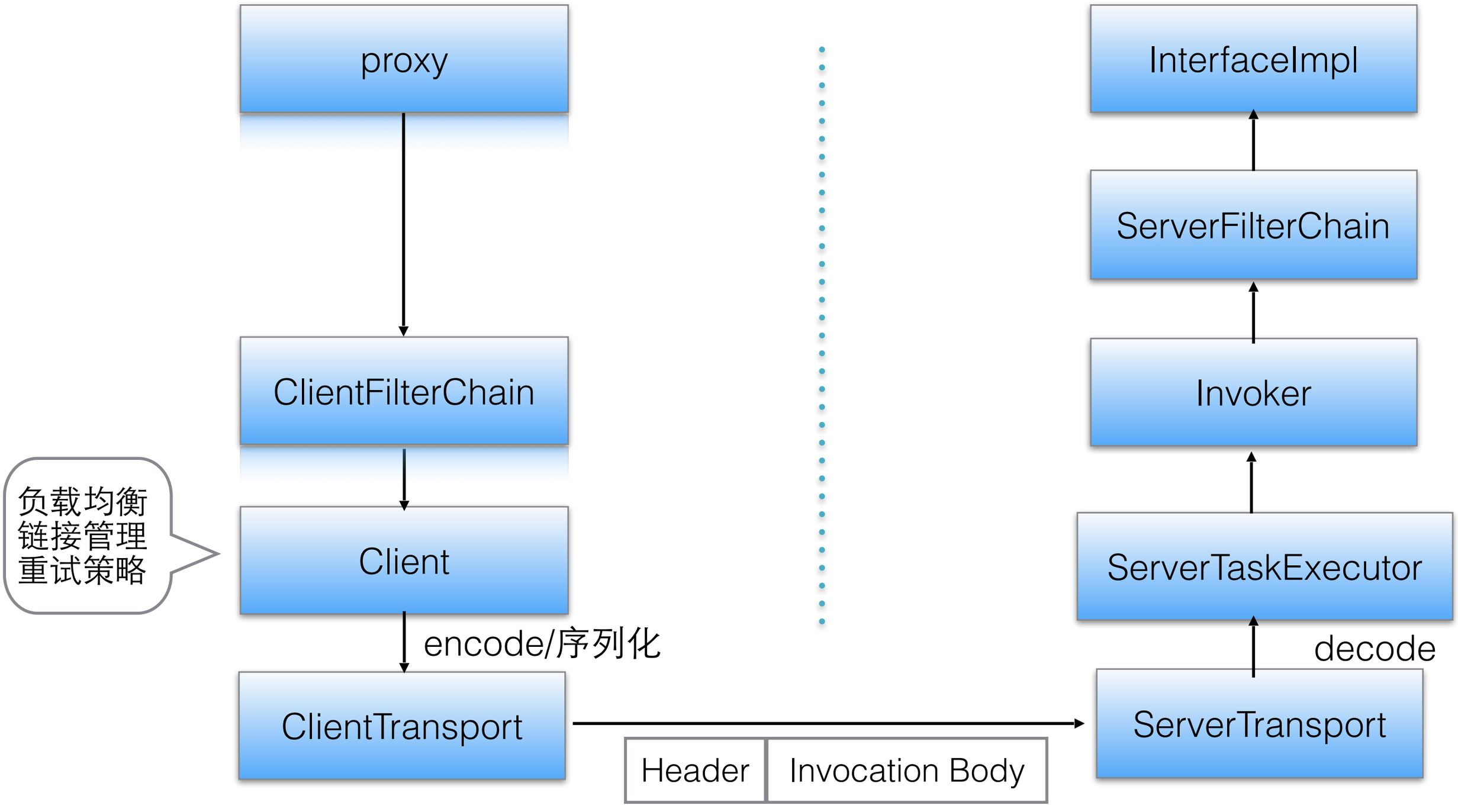
JSF协议直接调用

服务提供者1

服务提供者2

服务提供者3

JSF核心技术 – RPC示意图



NIO TCP长链接 IO-Multiplex多路复用

JSF核心技术 – 协议

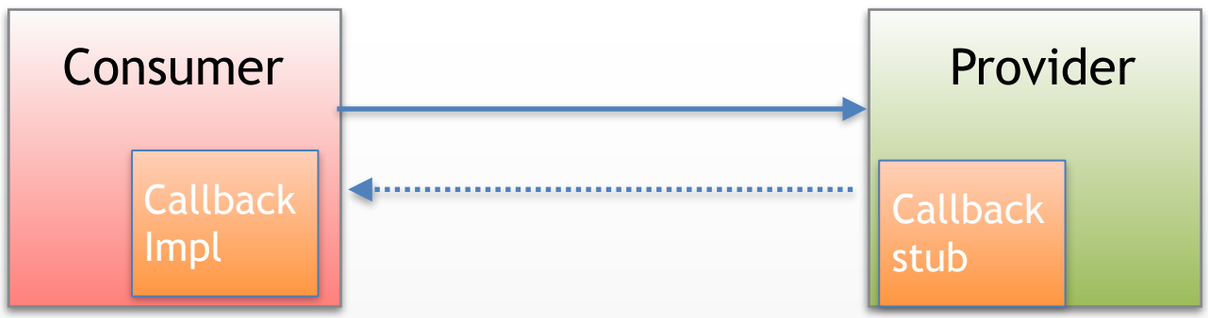
- 采用Netty来实现网络协议栈，异步事件通讯框架；
- 同一端口同时支持Http、TCP协议访问，根据数据包情况挂载不同解码器；
- TCP长链接下使用自定义二进制协议；
- HTTP网关来应对跨语言访问；

magic	full length	协议／序列化／消息..	消息ID	扩展描述	2000
ad cf	00 00 00 7f	00 0f	01 0a 01 00	00 00 0e 16	01 01 01	00 00 07 d0

97 91 c2 da 00 2f 63 6f 6d 2e 6a 64 2e 6a 6c… …

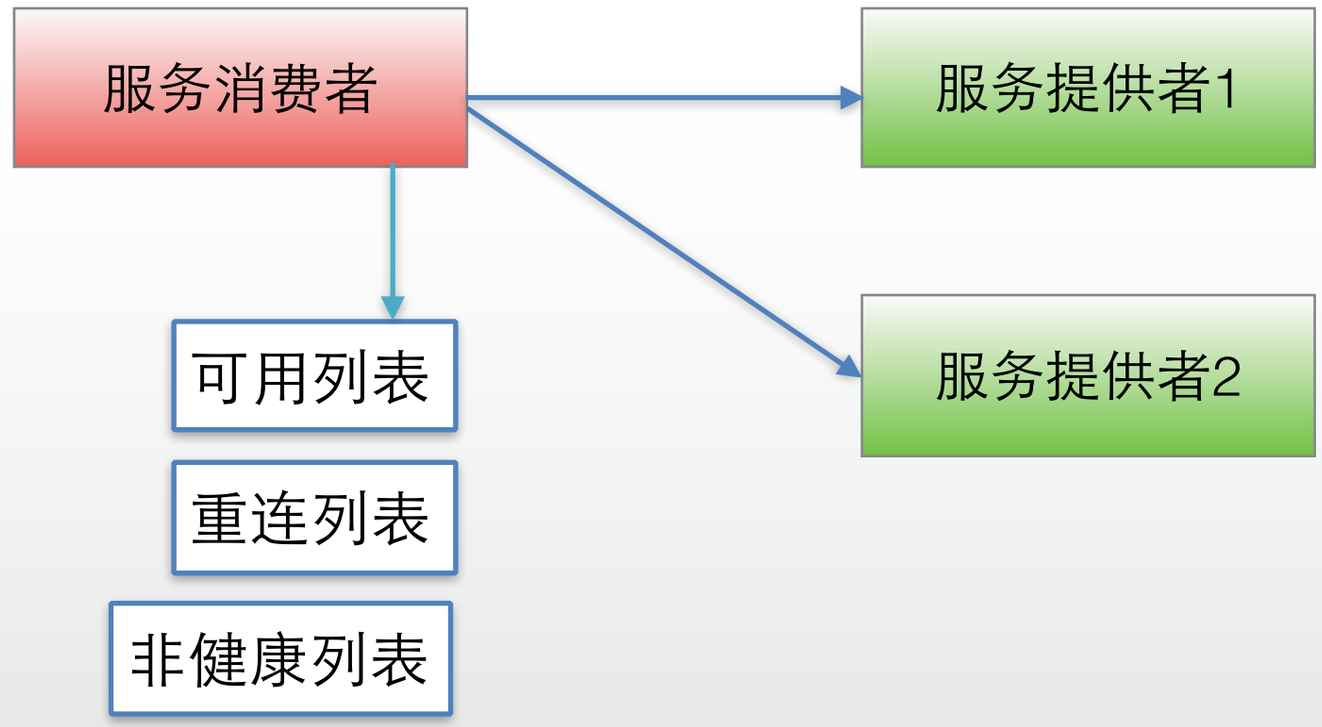
JSF核心技术 RPC – callback

- TCP长链接是双工的，服务方可以主动推送消息到调用方；
- 调用端检测到参数列表中有Callback类型，登记相应的callback对象；服务端收到调用时，生成相应的反向调用代理；
- 服务端持有此代理，并在需要时调用此代理来推送消息；



JSF核心技术 HA&负载均衡

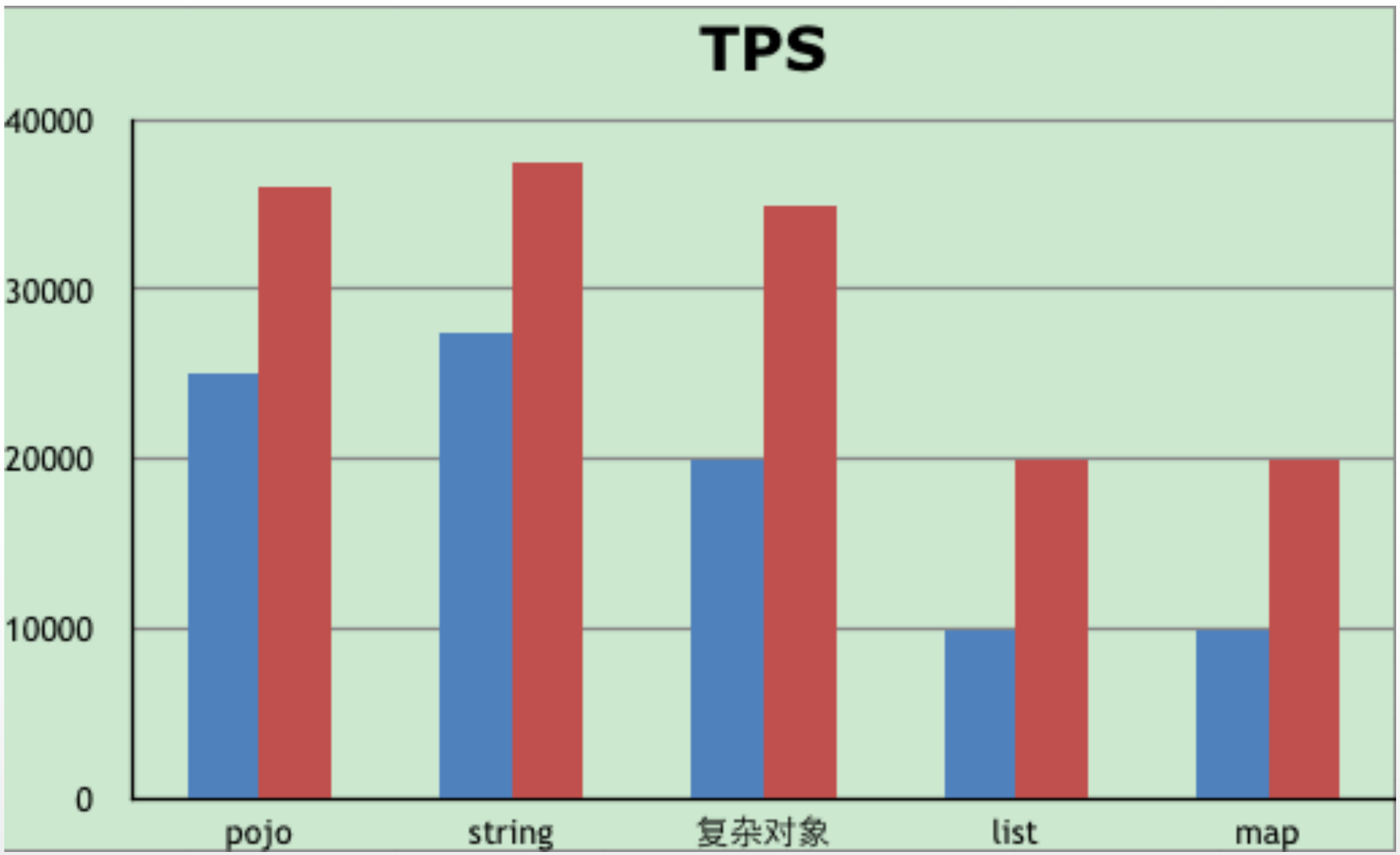
- 一个服务至少部署两个以上实例；
- 服务消费者运用负载均衡算法选择服务提供者，可以设置权重；
- 服务消费者对服务提供者有健康监测；
- 服务消费者端可以配置重试机制；



JSF核心技术 – 性能优化

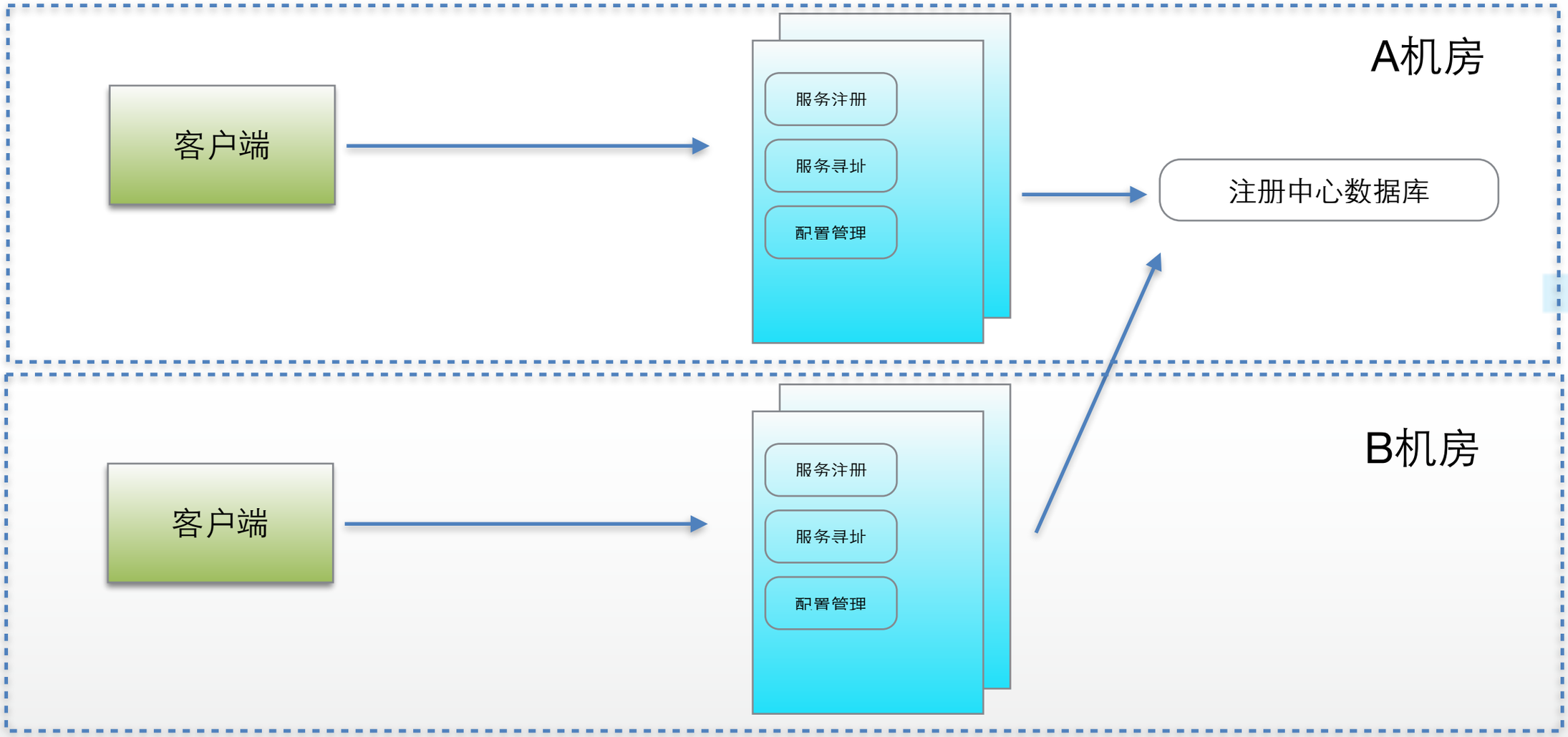
- 批量处理，请求先写入**RingBuffer**；
- 优化线程模型，将序列化与反序列化这种耗时的操作从Netty的IO线程中挪到用户线程池中；
- 启用压缩以应对大数据量的请求，默认snappy压缩算法；
- 定制msgpack序列化，序列化模版，同时还支持fast json、hessian等多种序列化协议；

JSF核心技术 – 性能优化



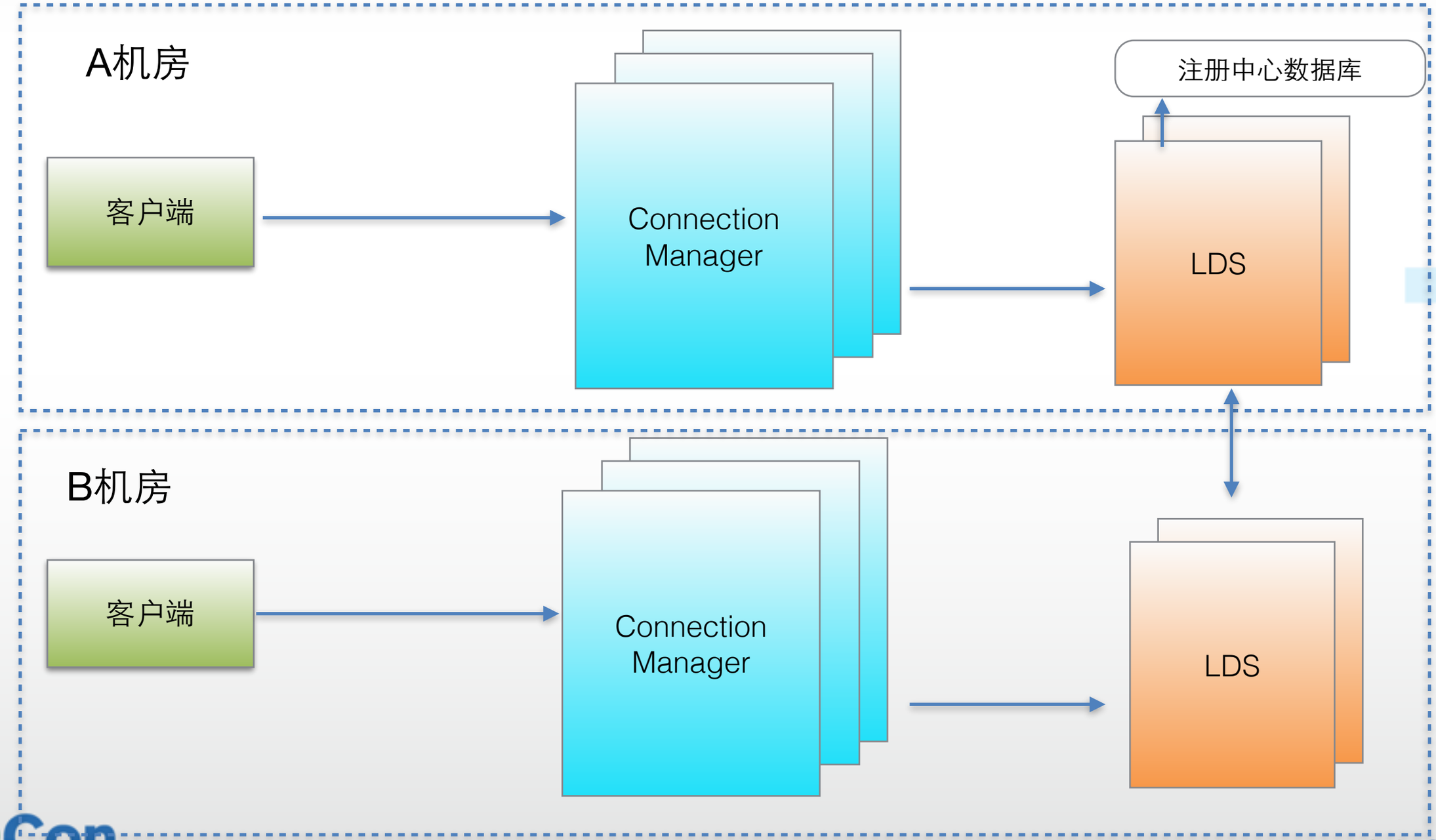
蓝色一代框架　红色二代框架

JSF核心技术 – 注册中心



优先访问本机房注册中心，各组件均有本地容灾缓存

JSF核心技术 – 注册中心



JSF核心技术 – 配置

- 服务提供者列表维护，动态推送；
- 查看当前服务生效的配置，动态下发新配置：权重／负载均衡算法／各种功能开关；
- 服务提供者动态分组无需重启；

JSF核心技术 – 限流

- 每一个服务调用者都有可能成为潜在的DDOS攻击者；
- 给服务的所有调用者带上标示，在系统环境变量中带上APPID；
- 开发计数器服务（Counter Service），限定单位时间内最大调用次数（如400次／分钟）；
- 限定服务端调用最大并发数（设定到接口 – 方法级别）；
- 服务端执行时检查请求的状态，如等待时间大于超时时间，直接丢弃；

JSF核心技术 – 降级

- 每个服务接口的每个方法都有灾备降级开关；
- 配置mock逻辑，返回的结果用json格式预先设好；
- 降级开关打开时将在consumer端短路RPC调用，直接返回JSON结果；

JSF核心技术 – 监控

服务提供者

服务消费者

调用情况报表

MonitorService

MonitorQueryService

分钟表

小时表

天表

influx DB

JSF核心技术 – 报警

- provider下线报警（心跳、telnet端口检查）；
- 某应用调用量超限额报警；
- Consumer存活报警；
- 耗时超限报警；
- 异常（Exception）捕获报警；

JSF核心技术 – 监控报表(1)

JSF动态 × 接口监控信息 × 服务监控 ×

接口名称: com.jdjr.rdp.collector.service.KeyValueService

方法名称: takeDate

提 供 者 : *

调 用 者 : *

时间维度: 原始表

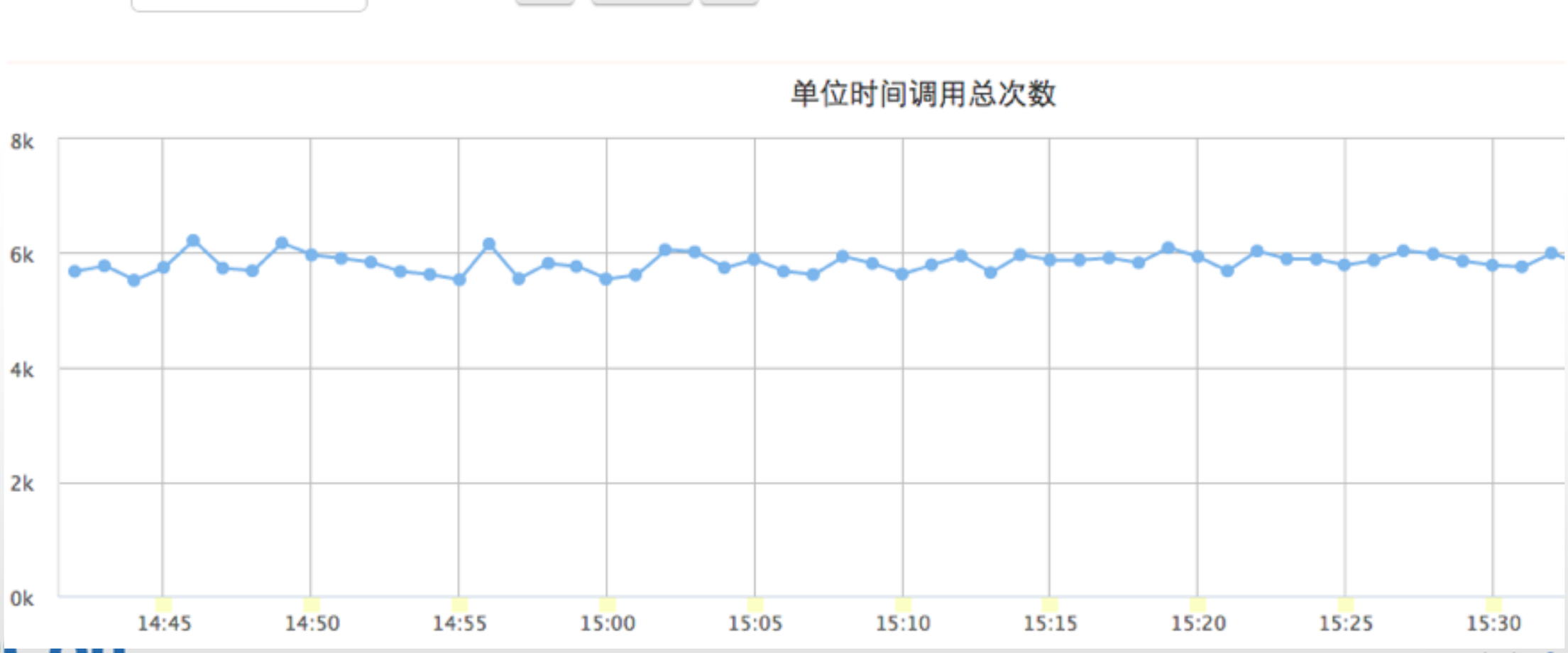
数据粒度: 1分钟

开始时间:

结束时间:

发布协议:

查看 调用分布 返回



JSF核心技术 – 监控报表(2)

JSF动态 × 接口监控信息 × 服务监控 ×

接口名称: com.jdjr.rdp.collector.service.KeyValueService

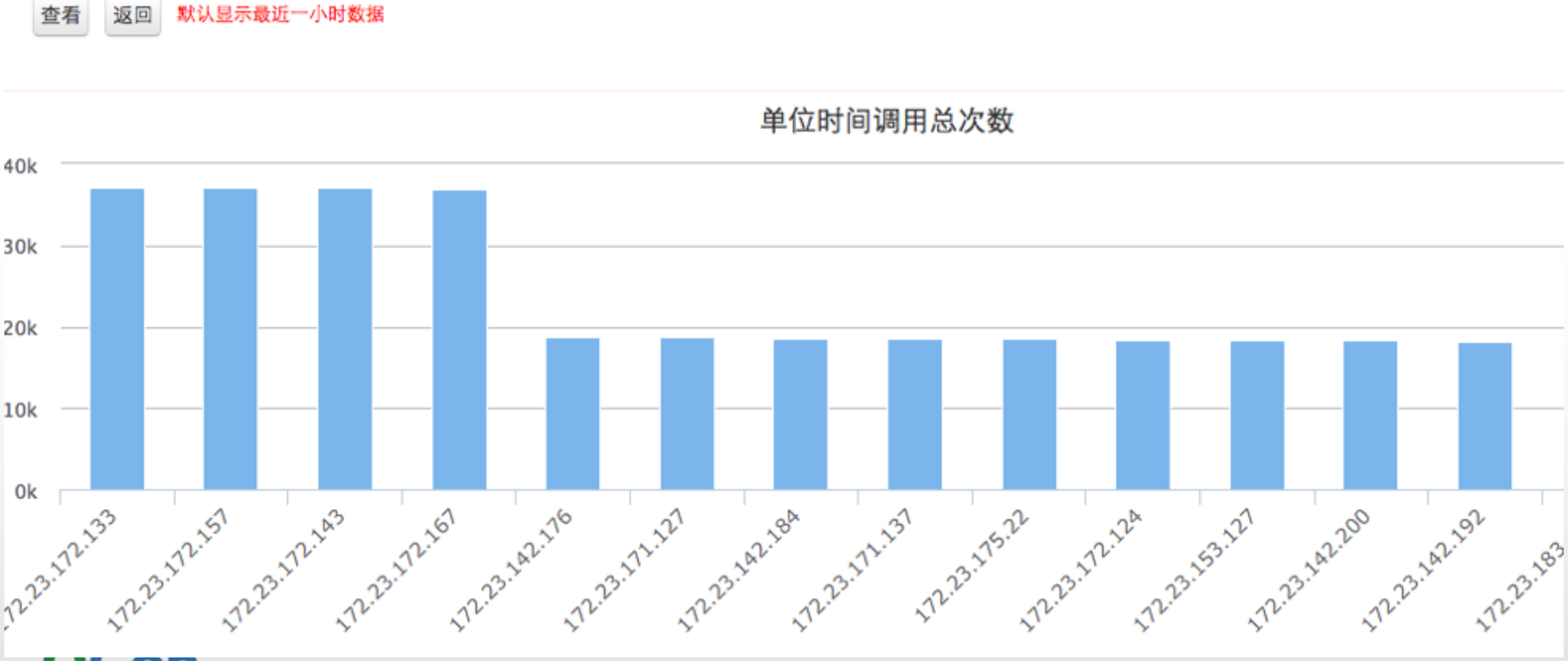
方法名称: takeDate

开始时间:

结束时间:

发布协议:

查看 返回 默认显示最近一小时数据



JSF核心技术 – 弹性云部署

- 按需自动扩展服务能力;
- CAP（Cloud Application Platform）系统自动分配物理机并创建容器;
- 自动部署系统在容器上部署业务应用;
- 应用启动并在JSF服务注册中心进行注册;

接口设计的问题

- “无缝将本地接口发布为远程接口，调用与本地一样” – 只能是理想
- 考虑超时异常的处理;
- 考虑业务逻辑粒度;
- 考虑是否幂等;
- 一些不好的接口设计举例:
  - 返回值类型为Object，实际类型被擦除
  - 接口声明中使用范型参数T
  - 嵌套层数太多

京东服务化的现状

- 接入4000余个接口（按Java Interface计算）；
- 接入的物理机 、docker按独立IP计算共 10000+；
- 每日上百亿次的调用量；
- 商品接口：500 + 多个服务实例，9000+消费实例；

2012-06-18	*2013-06-18*	*2014-06-18*
7	*60*	*700*

商品接口京东店庆日调用次数统计 单位（亿次）

研发中的一些感悟...

- 数字化运营（实时数据／阶段性统计数据）；
- 开源软件，必须彻底掌握后再使用；
- 寻找系统的关键点，详尽的日志帮助找到问题所在；
- 客户端逻辑尽量简单，逻辑做到服务端去；

研发中的一些感悟...

- 重复发明轮子不是问题，满足客户需求是第一位的；
- 线上实际环境的压测非常重要；
- 定时线上容灾演练，验证应急预案的实际执行效果；

下一步研发方向

- 服务治理，根据应用ID的一系列管理增强；
- HTTP/2等协议的支持；
- 增强跨语言支持；

Q&A

www.infoq.com/cn

InfoQueue

@InfoQ

infoqchina

软件
正在改变世界！