Node 调用 dubbo 服务的探索及实践

应杲臻

2019年04月11日

自我介绍

应杲(gǎo)臻

贝贝集团开发人员

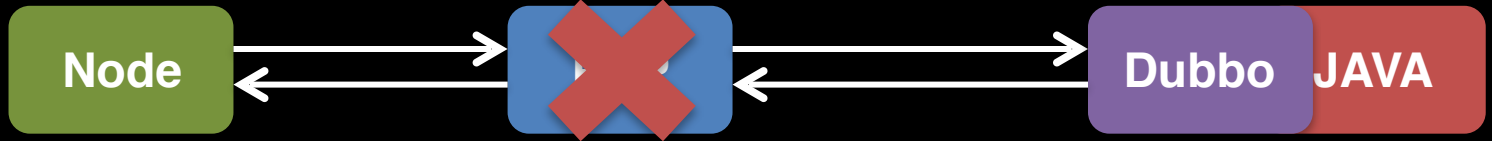
目前王者荣耀本赛季荣耀王者50星

欢迎开黑^_^

beibei.com

现状分析

现状分析



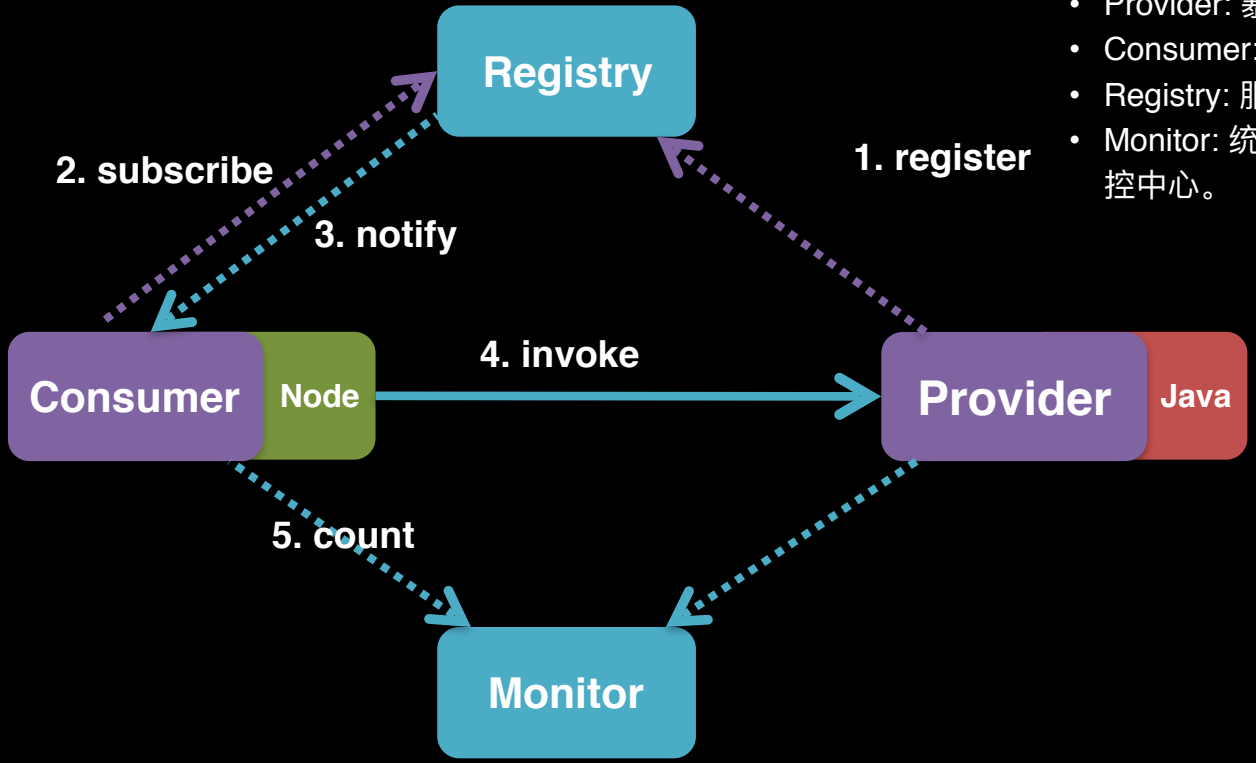
数据展示层　　数据聚合层　　基础服务层

19年的贝贝技术生态逐渐拥抱JAVA，而PHP在公司内将逐渐被下线
而PHP所承担的功能将会往Node以及Java两侧分流

数据展示/聚合层　　基础服务层

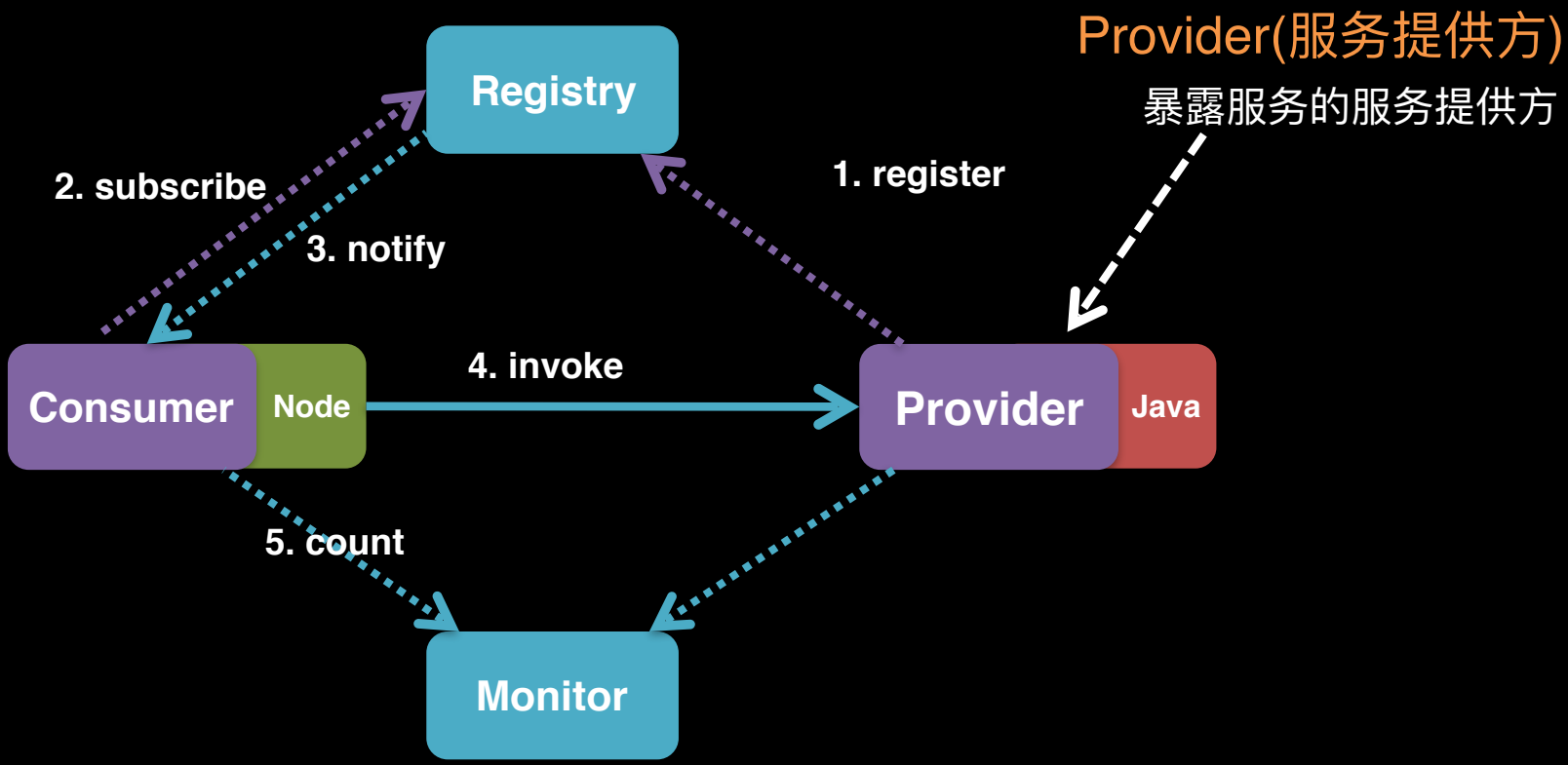
Dubbo简介

Dubbo是一款高性能、轻量级的开源Java RPC框架

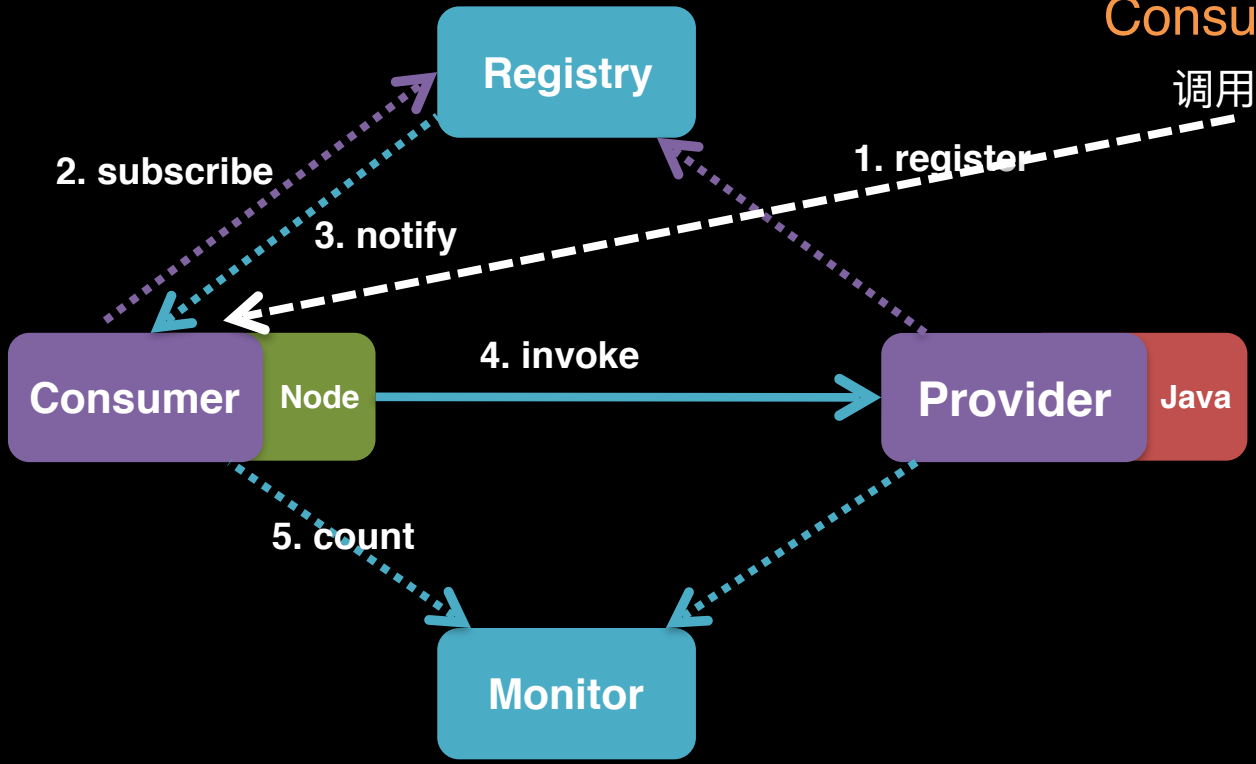


- Provider: 暴露服务的服务提供方
- Consumer: 调用远程服务的服务消费方。
- Registry: 服务注册与发现的注册中心。
- Monitor: 统计服务的调用次调和调用时间的监控中心。

Dubbo是一款高性能、轻量级的开源Java RPC框架



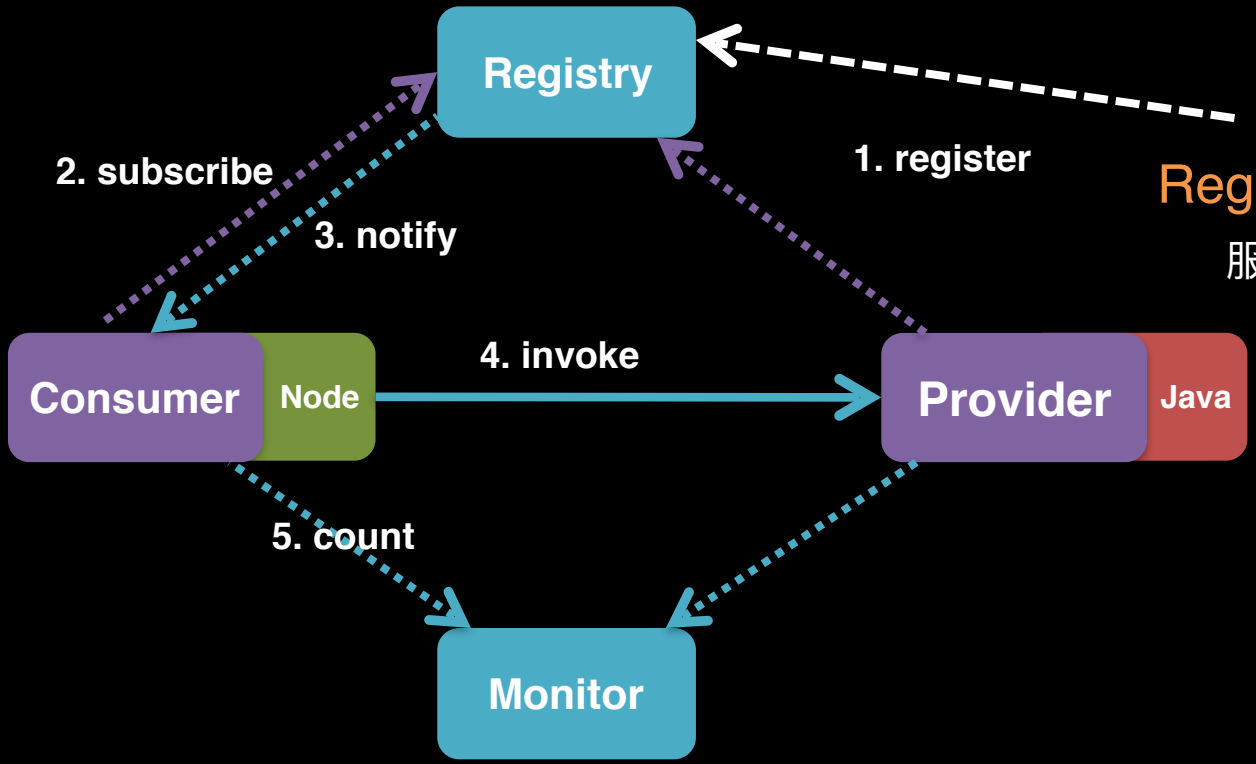
Dubbo是一款高性能、轻量级的开源Java RPC框架



Consumer(服务消费方)

调用远程服务的服务消费方

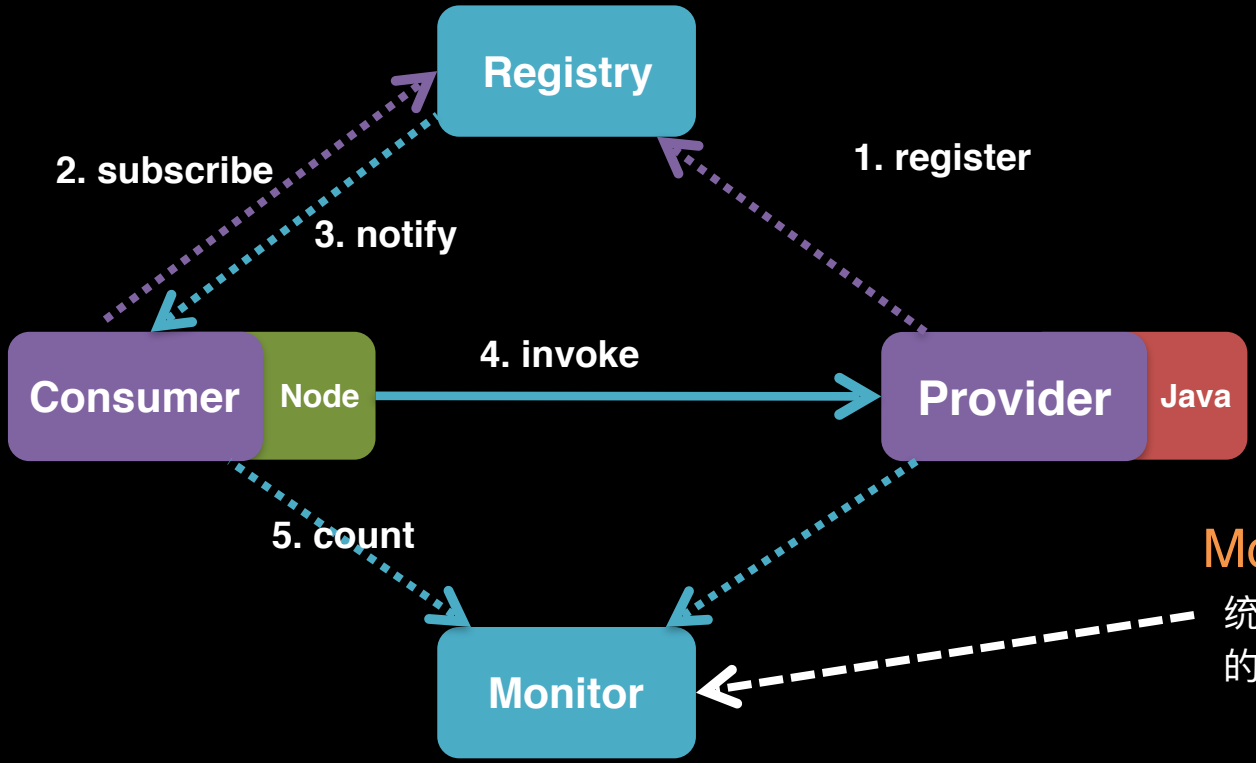
Dubbo是一款高性能、轻量级的开源Java RPC框架



Registry(服务注册中心)

服务注册与发现的注册中心

Dubbo是一款高性能、轻量级的开源Java RPC框架



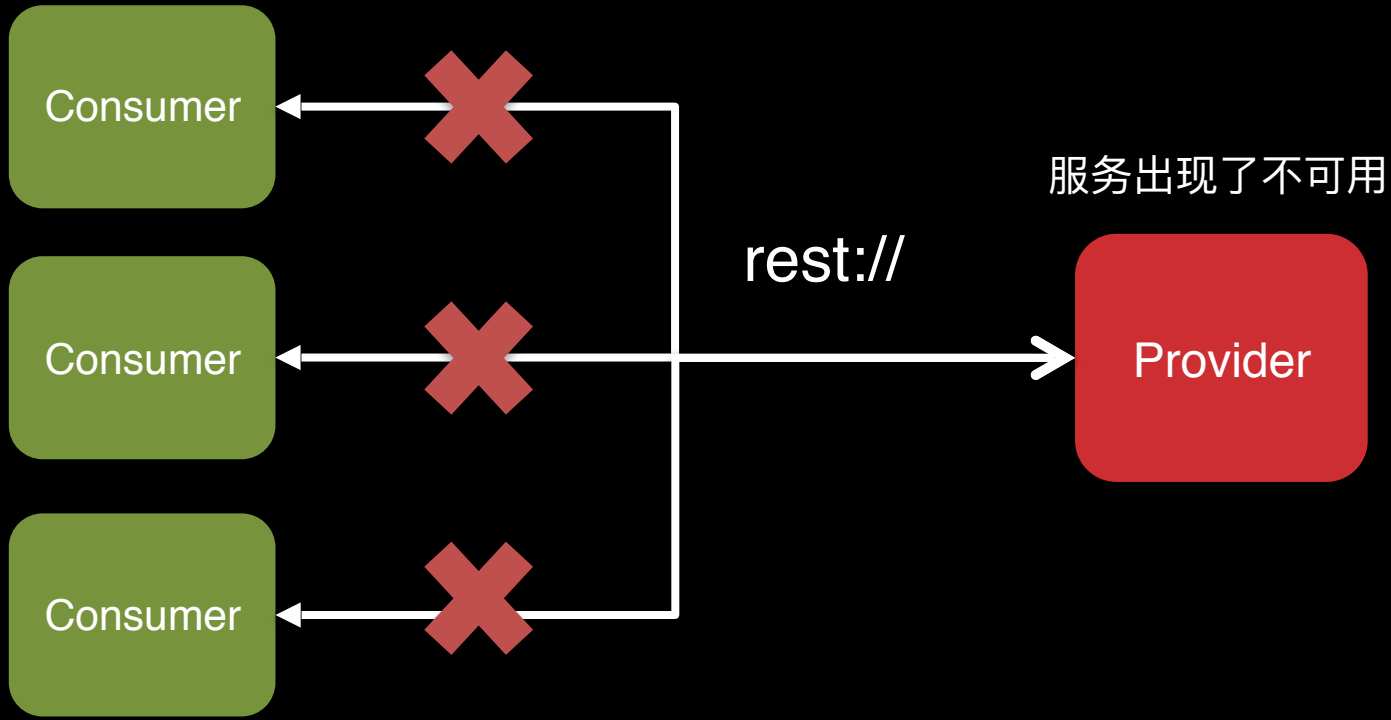
Monitor(监控中心)

统计服务的调用次数 / 调用时间的监控中心

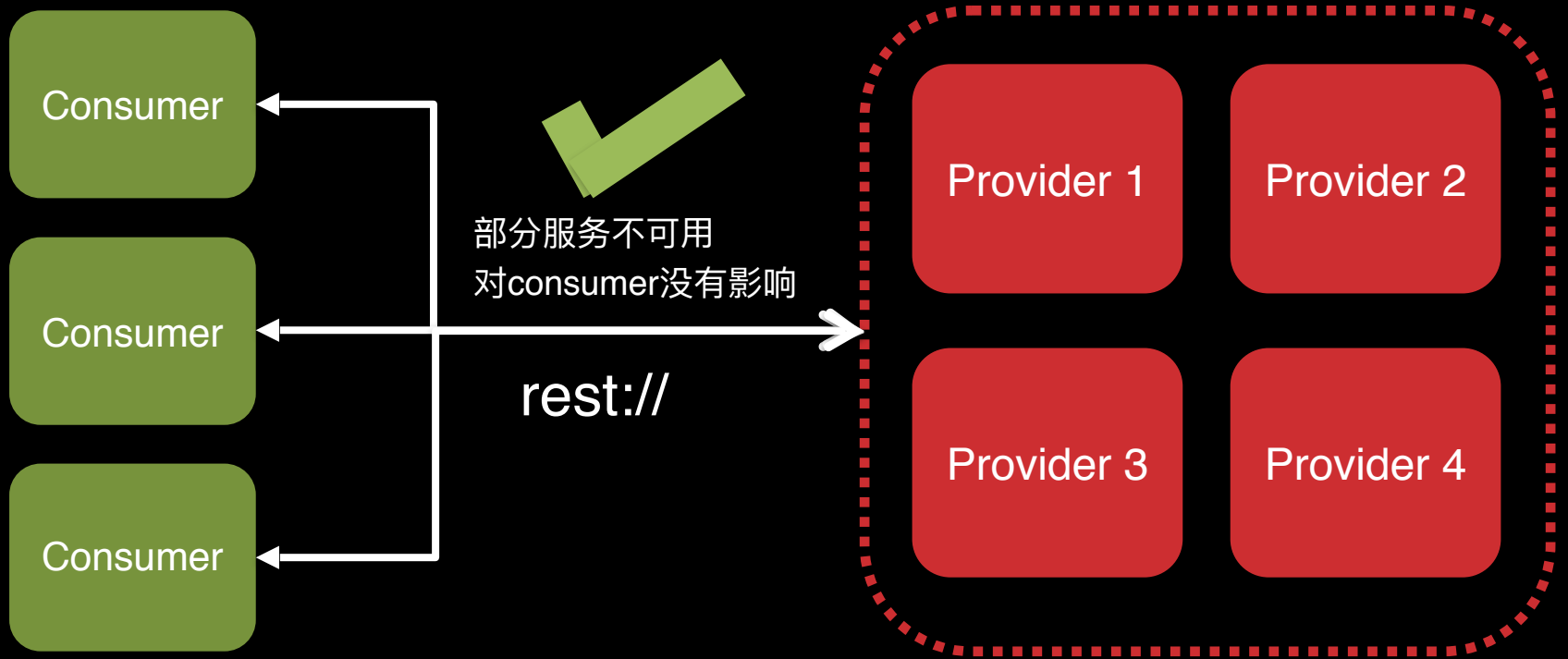
使用情况

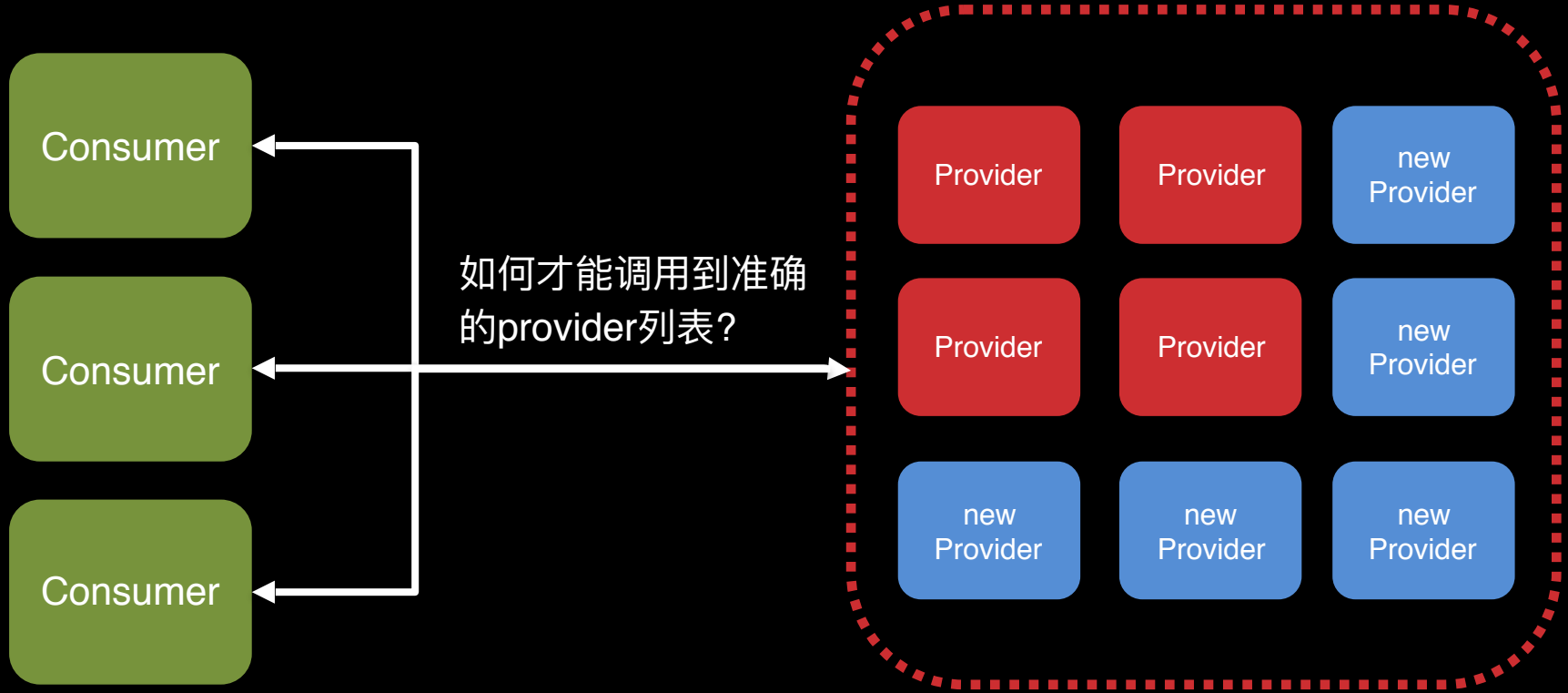
场景	使用对象	使用服务数	调用情况
• 渠道页面 • 广告场景 • 中台工程 • 内部工具	• 前端同学	• 已达上千	• 2亿+/每日

1.0 尝试


Consumer
Consumer
Consumer
rest://
服务出现了不可用
Provider

2.0 稳定

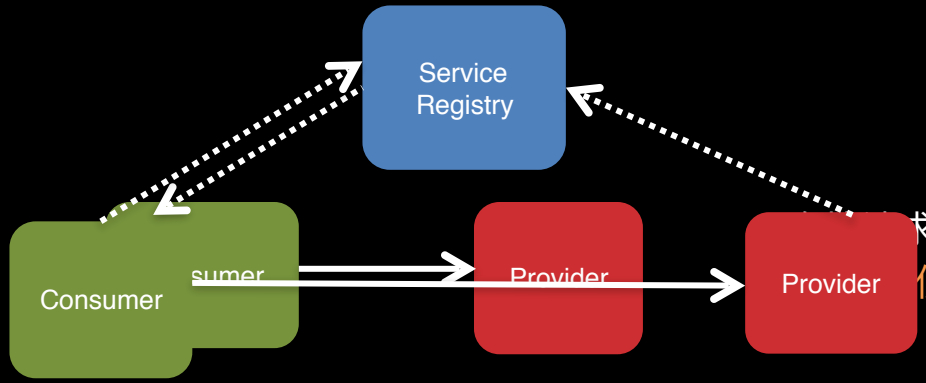

Consumer
Consumer
Consumer
部分服务不可用
对consumer没有影响
rest://
Provider 1
Provider 2
Provider 3
Provider 4


Consumer
Consumer
Consumer
如何才能调用到准确的provider列表?
Provider
Provider
new Provider
Provider
Provider
new Provider
new Provider
new Provider
new Provider

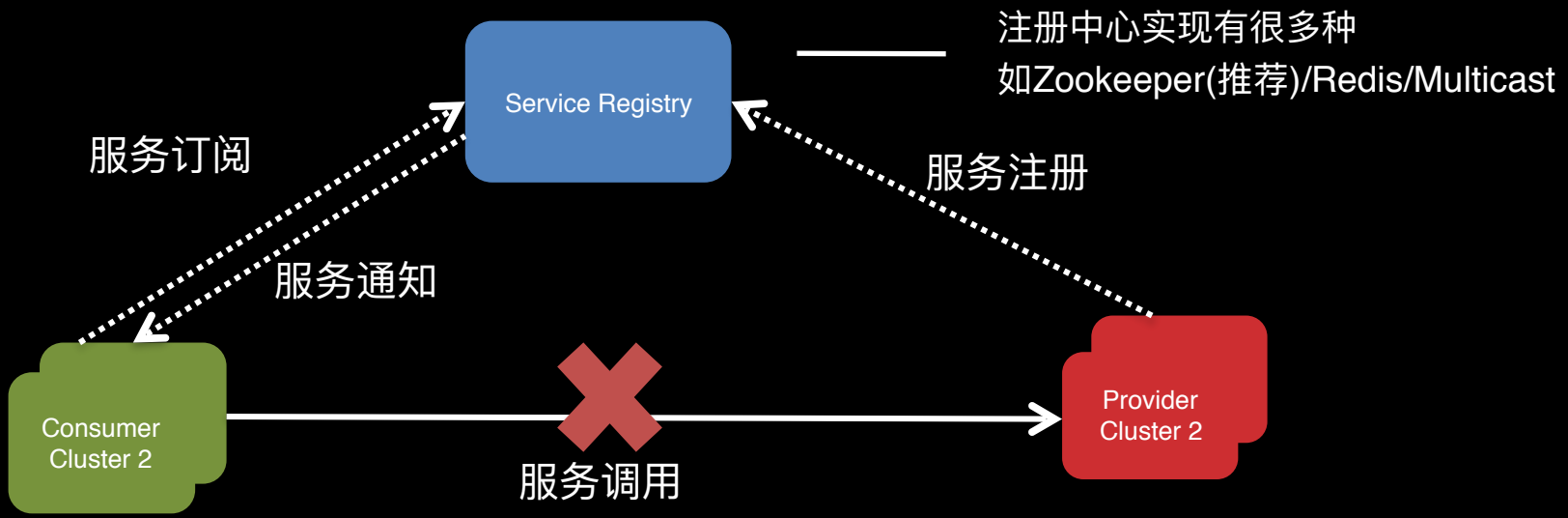
目前引用服务有两种方式

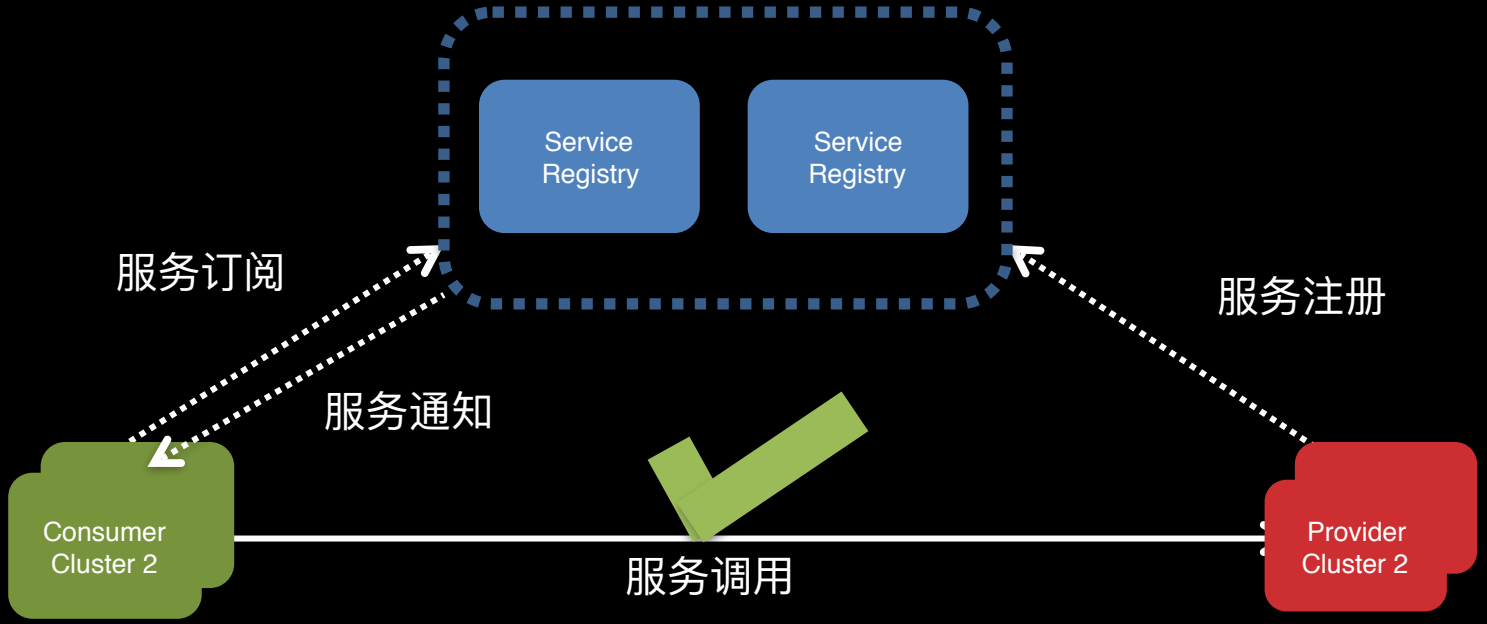
不通过注册中心
直接引用服务

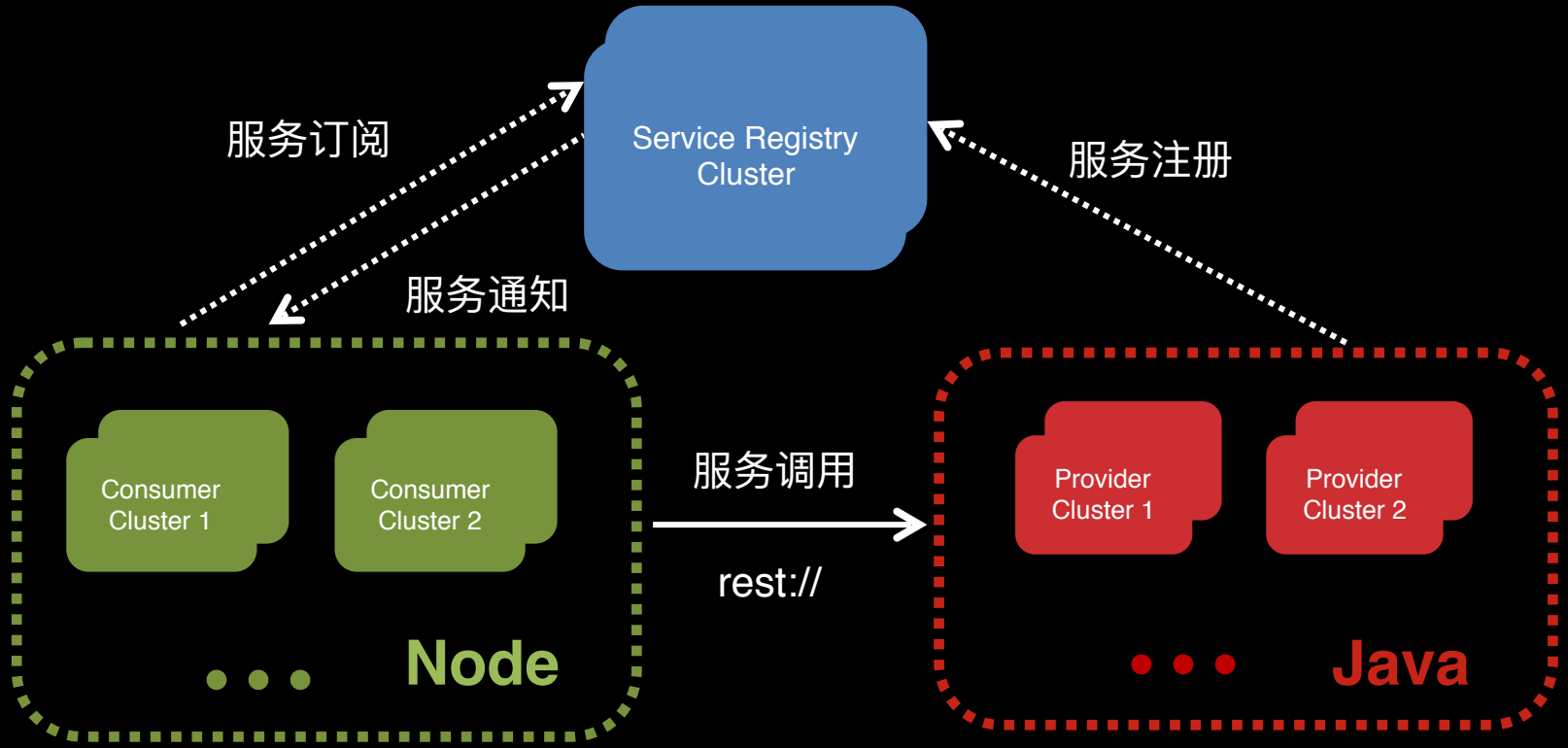
通过注册中心
发现引用服务



通过注册中心发现引用服务，Dubbo常用的引
[illegible]务提供者,无法做到高可用.所以生产环境调
使用此模式的.但是在开发环境可以尝试使用


Service Registry
注册中心实现有很多种
如Zookeeper(推荐)/Redis/Multicast
服务订阅
服务注册
服务通知
Consumer
Cluster 2
Provider
Cluster 2
服务调用


Service
Registry
Service
Registry
服务订阅
服务注册
服务通知
Consumer
Cluster 2
服务调用
Provider
Cluster 2


Service Registry
Cluster
服务订阅
服务通知
服务注册
Consumer
Cluster 1
Consumer
Cluster 2
Node
服务调用
rest://
Provider
Cluster 1
Provider
Cluster 2
Java

3.0 改进

协议选择

rest

hessian

rmi

webservice

dubbo

连接个数：

多连接

连接方式：

短连接

传输协议：

HTTP

传输方式：

同步传输

序列化：

表单序列化

适用范围：

传入传出参数数据包大小混合，提供者比消费者个数多，可用浏览器查看，可用表单或URL传入参数

适用场景：

需同时给应用程序和浏览器 JS 使用的服务。

优点：

- 对开发者友好
- 无语言使用门槛

缺点：

- 短连接会反复建连
- 协议体较大，性能较差

改进：

- 开启keepalive
- 可切换其他协议，如dubbo://

rest

hessian

rmi

webservice

dubbo

连接个数：

多连接

连接方式：

短连接

传输协议：

HTTP

传输方式：

同步传输

序列化：

Hessian二进制序列化

适用范围：

传入传出参数数据包较大，提供者比消费者个数多，提供者压力较大，可传文件。

适用场景：

页面传输，文件传输，或与原生hessian服务互操作。

总结：

- HTTP短连性能相对较差
- 适用场景不太符合我们的业务

rest

hessian

rmi

webservice

dubbo

连接个数：

多连接

连接方式：

短连接

传输协议：

TCP

传输方式：

同步传输

序列化：

Java 标准二进制序列化

适用范围：

传入传出参数数据包大小混合，消费者与提供者个数差不多，可传文件。

适用场景：

常规远程服务方法调用，与原生RMI服务互操作。

总结：

- HTTP短连性能相对较差
- 通常用于Java服务之间调用

rest

hessian

rmi

webservice

dubbo

连接个数：

多连接

连接方式：

短连接

传输协议：

HTTP

传输方式：

同步传输

序列化：

SOAP 文本序列化

适用场景：

系统集成，跨语言调用。

总结：

- HTTP短连性能相对较差
- 适用场景不太符合我们的业务

rest

hessian

rmi

webservice

dubbo

连接个数：

单连接

连接方式：

长连接

传输协议：

TCP

传输方式：

NIO 异步传输

序列化：

Hessian 二进制序列化

适用范围：

传入传出参数数据包较小，消费者比提供者个数多，单一消费者无法压满提供者。

适用场景：

常规远程服务方法调用。

优点：

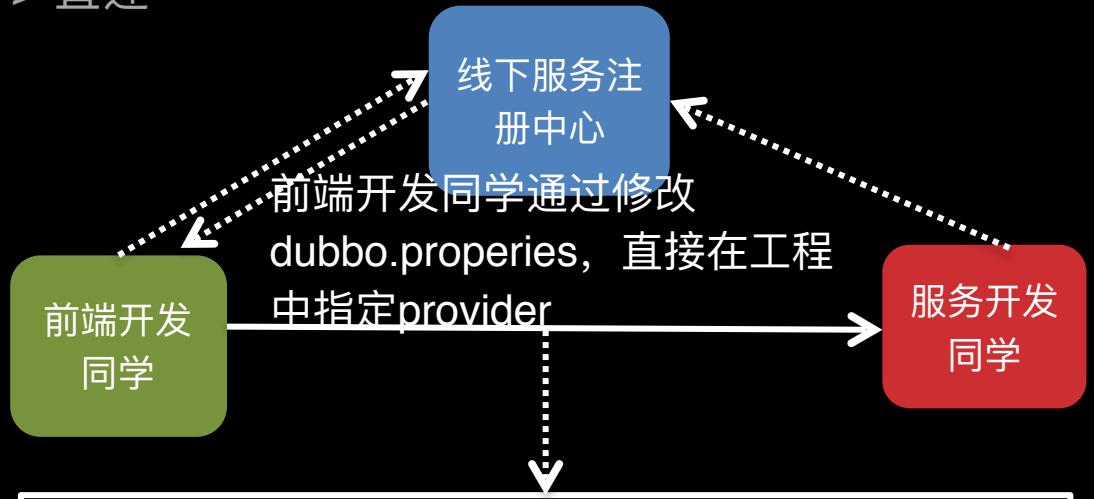
- 协议性能较好

缺点：

- 需维护socket连接状态

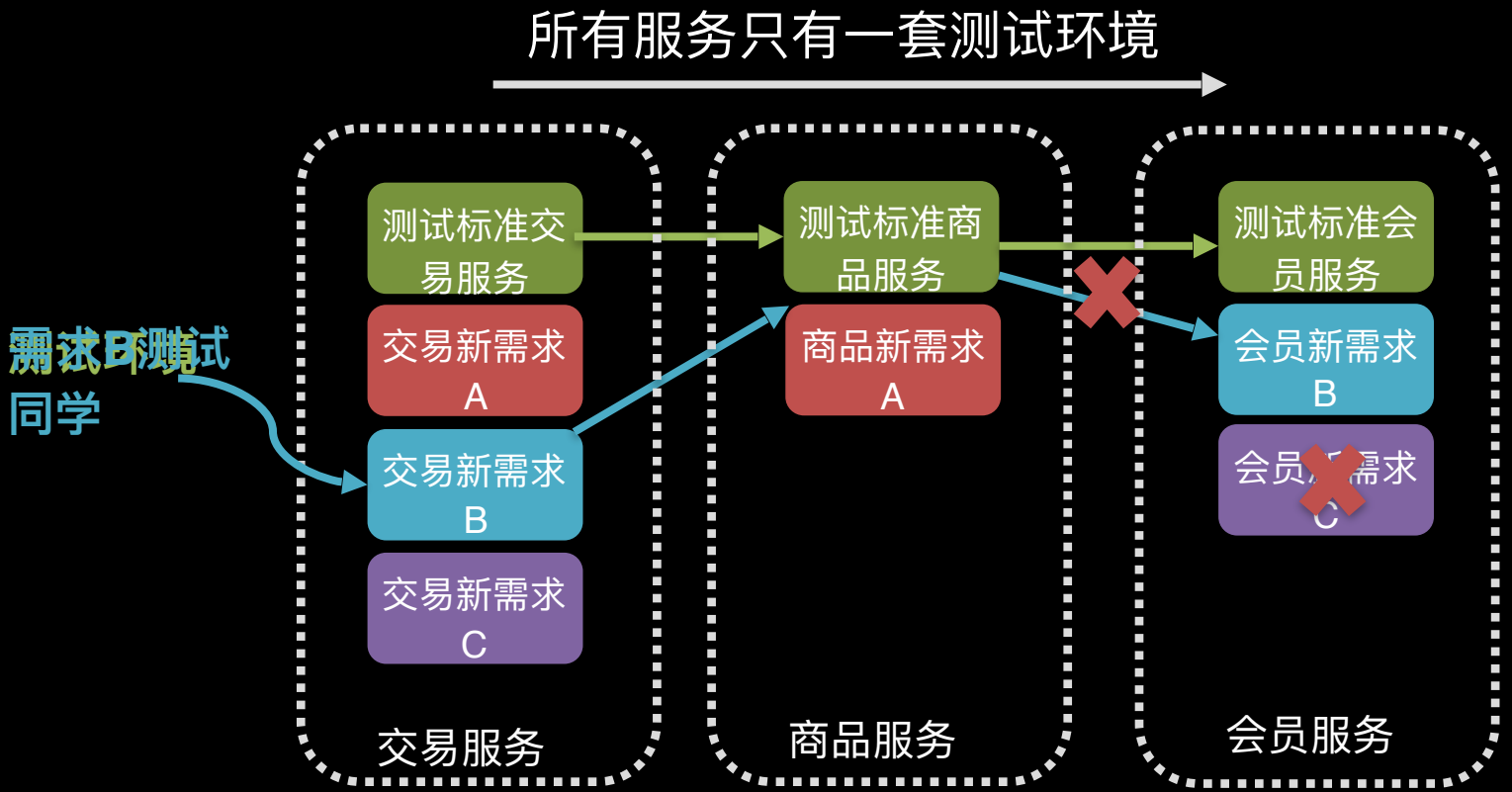
	连接个数	连接方式	传输协议	传输方式	序列化	适用范围	适用场景
dubbo	单连接	长连接	TCP	NIO异步传输	Hessian 二进制序列化	传入传出参数数据包较小，消费者比提供者个数多，单一消费者无法压满提供者	常规远程服务方法调用
rmi	多连接	短连接	TCP	同步传输	Java 标准二进制序列化	传入传出参数数据包大小混合，消费者与提供者个数差不多，可传文件。	常规远程服务方法调用，与原生RMI服务互操作
hessian	多连接	短连接	HTTP	同步传输	Hessian二进制序列化	传入传出参数数据包较大，提供者比消费者个数多，提供者压力较大，可传文件。	页面传输，文件传输，或与原生hessian服务互操作
http	多连接	短连接	HTTP	同步传输	表单序列化	传入传出参数数据包大小混合，提供者比消费者个数多，可用浏览器查看，可用表单或URL传入参数	需同时给应用程序和浏览器 JS 使用的服务。
rest	多连接	短连接	HTTP	同步传输	表单序列化	同http，适用于更加符合rest规范的服务	同http

如何联调



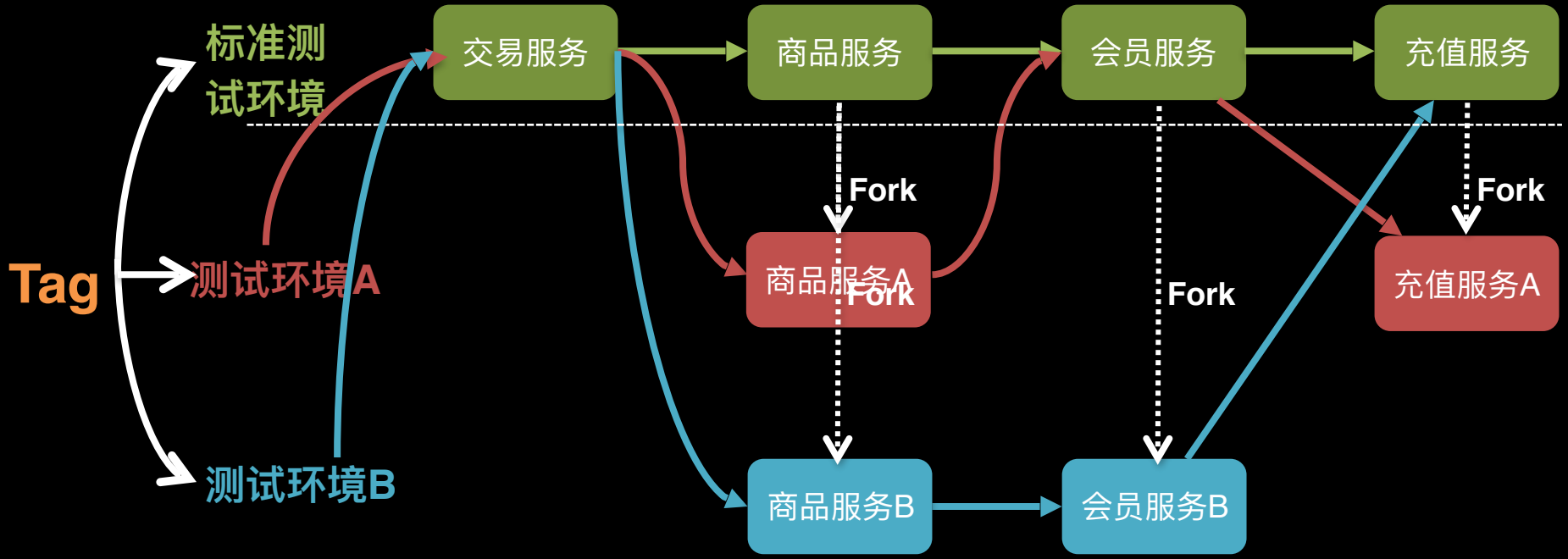

```
[\com\ProductService]
dubbo.service.provider=dubbo://10.1.1.1:10880/
[\com\MemberService]
dubbo.service.provider=dubbo://10.2.1.2:10880/



"config/development/dubbo.properties"
```

dubbo.properties不会加入到工程的版本控制当中



如果放置在一个环境下，业务之间会互相影响

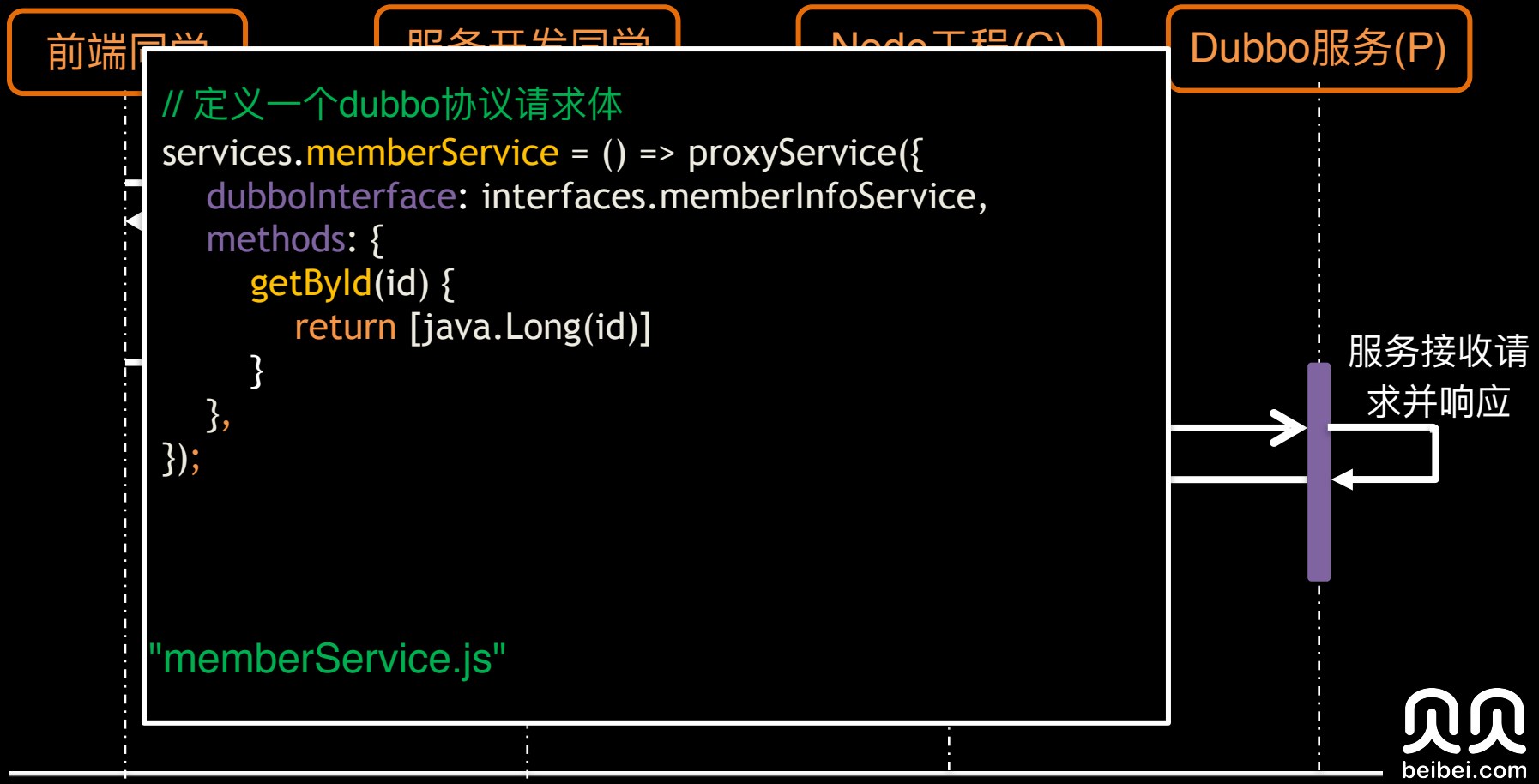
环境工厂


标准测试环境
交易服务
商品服务
会员服务
充值服务
Fork
Fork
Fork
Tag
测试环境A
商品服务A
充值服务A
测试环境B
商品服务B
会员服务B

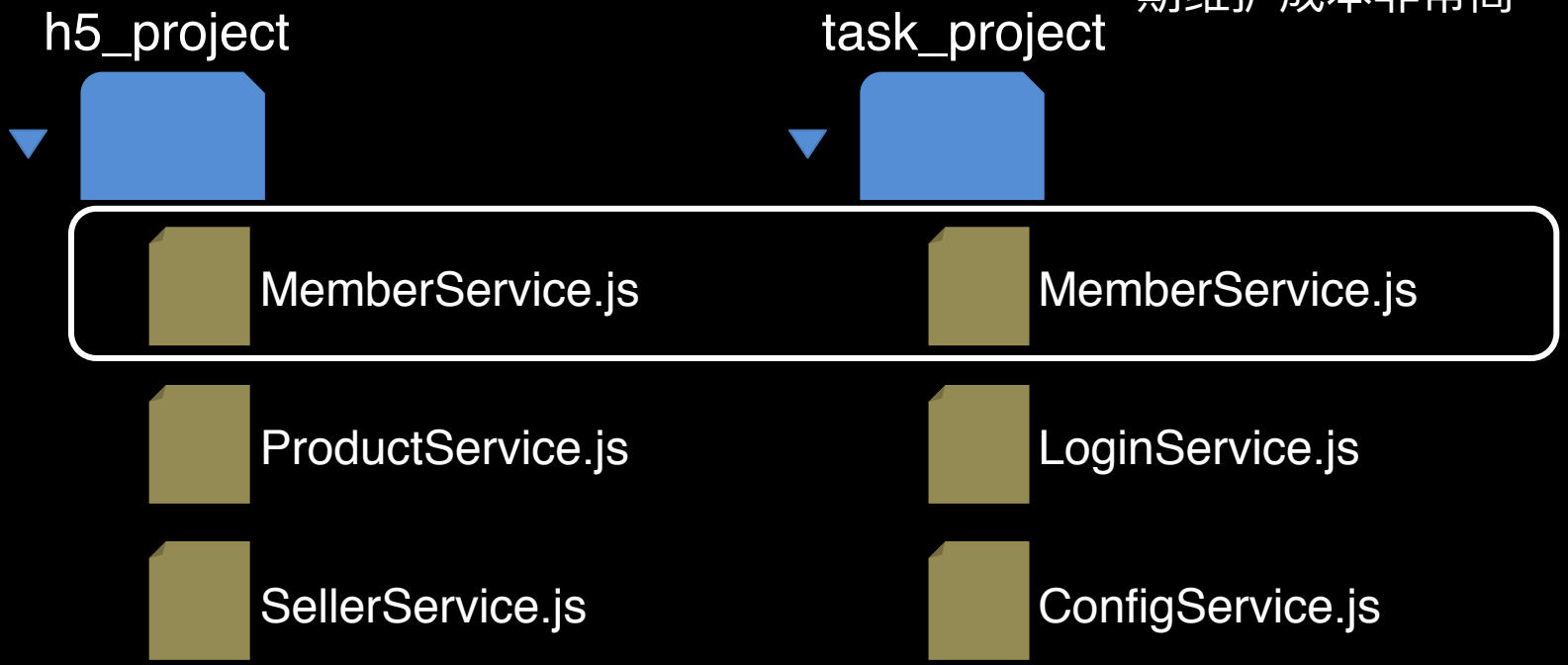
4.0 统一

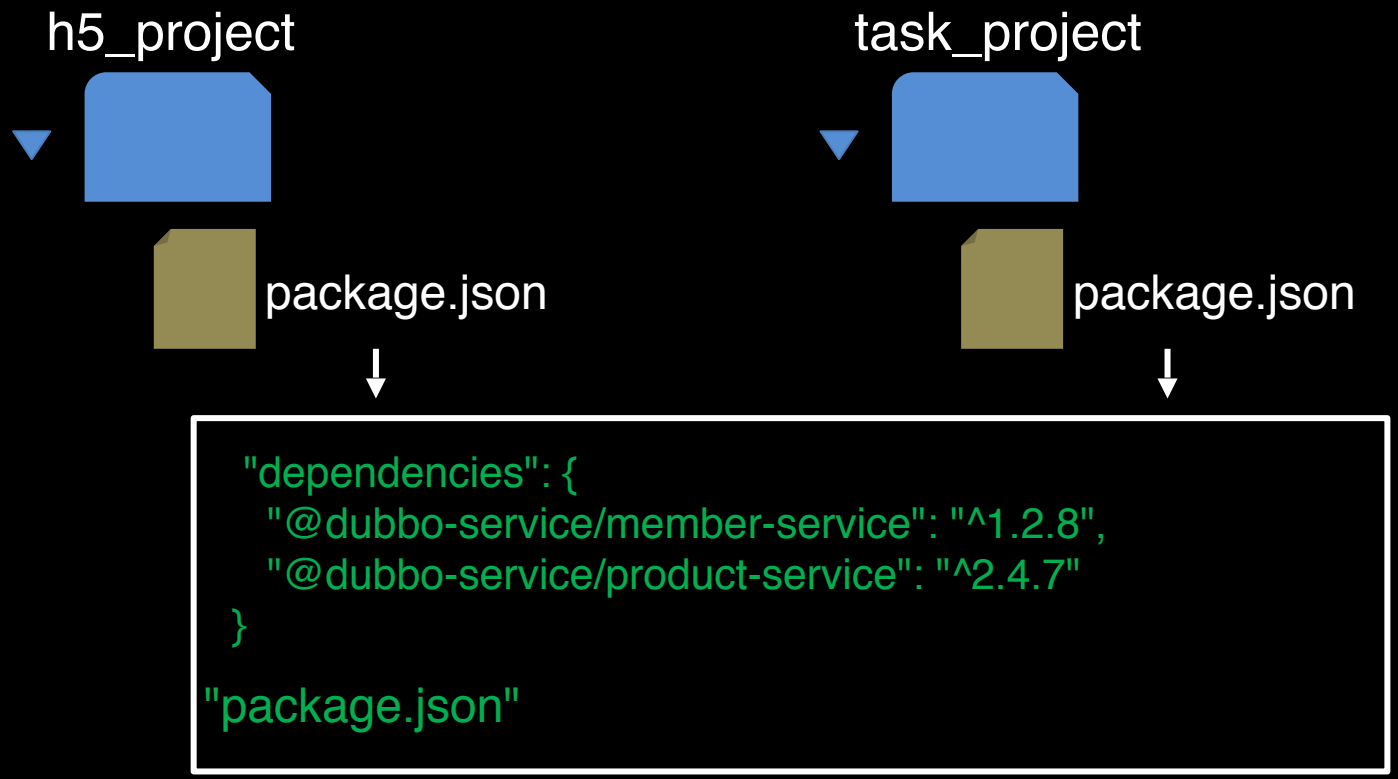
在贝贝，如果一位前端同学要用Node请求一个dubbo服务

Ta该怎么做呢？


前端同学
服务开发同学
Node工程(C)
Dubbo服务(P)
// 定义一个dubbo协议请求体
services.memberService = () => proxyService({
dubboInterface: interfaces.memberInfoService,
methods: {
getById(id) {
return [java.Long(id)]
}
},
});
"memberService.js"
服务接收请求并响应

请求的定义分散在不同工程，而且维护人员不统一，导致后期维护成本非常高




h5_project
task_project
package.json
package.json
"dependencies": {
"@dubbo-service/member-service": "^1.2.8",
"@dubbo-service/product-service": "^2.4.7"
}
"package.json"

事情到这里，我们已经解决了服务如何统一定义的问题，但是仍然没有解决统一管理与维护的问题．如：

- 项目迁移成本问题．由于是涉及到项目迁移问题，我们需要初始化很多现有的Java服务，而手动去定义服务工程量巨大，所以我们需要考虑迁移成本问题.
- 维护人员职责划分问题．NPM包的维护工作该交给服务提供方还是服务调用方?

JAVA

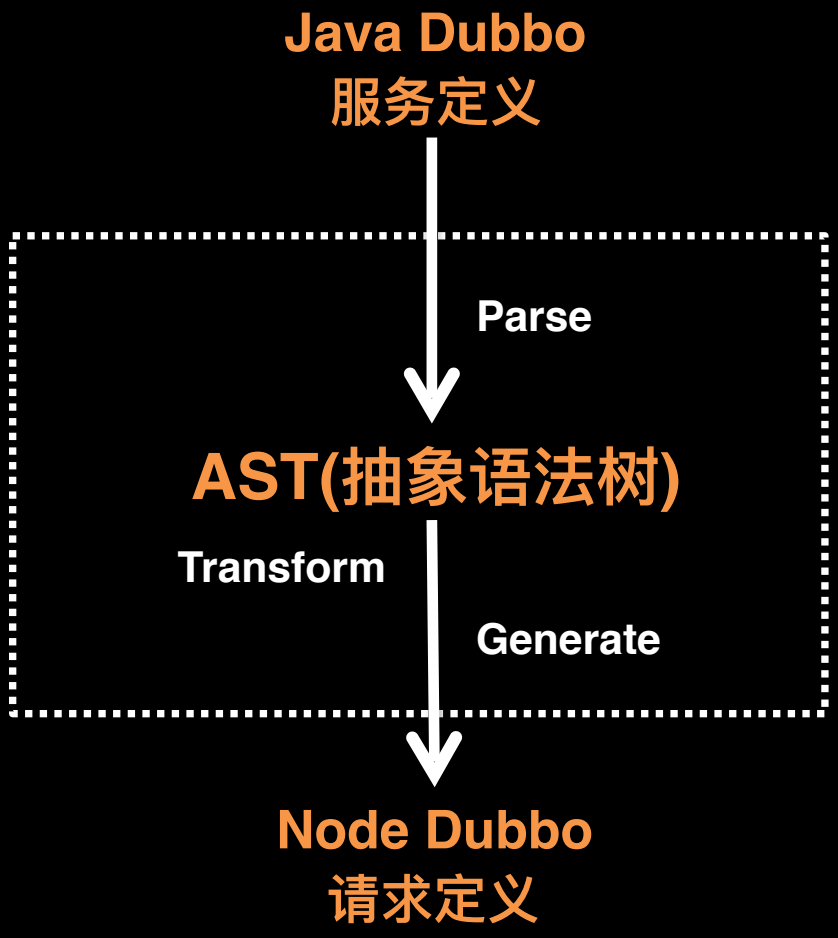
```java
package com.service;

public interface TestService {

    /**
     * 给定id,获取所有对应信息
     * @param id 测试id
     */
    TestTO getByid(Long id);

    /**
     * 给定ids,获取所有对应信息列表
     * @param ids 测试id
     */
    List<TestTO> listByIds(List<Long> ids);
}
```

>> JTJ >>

转换工具

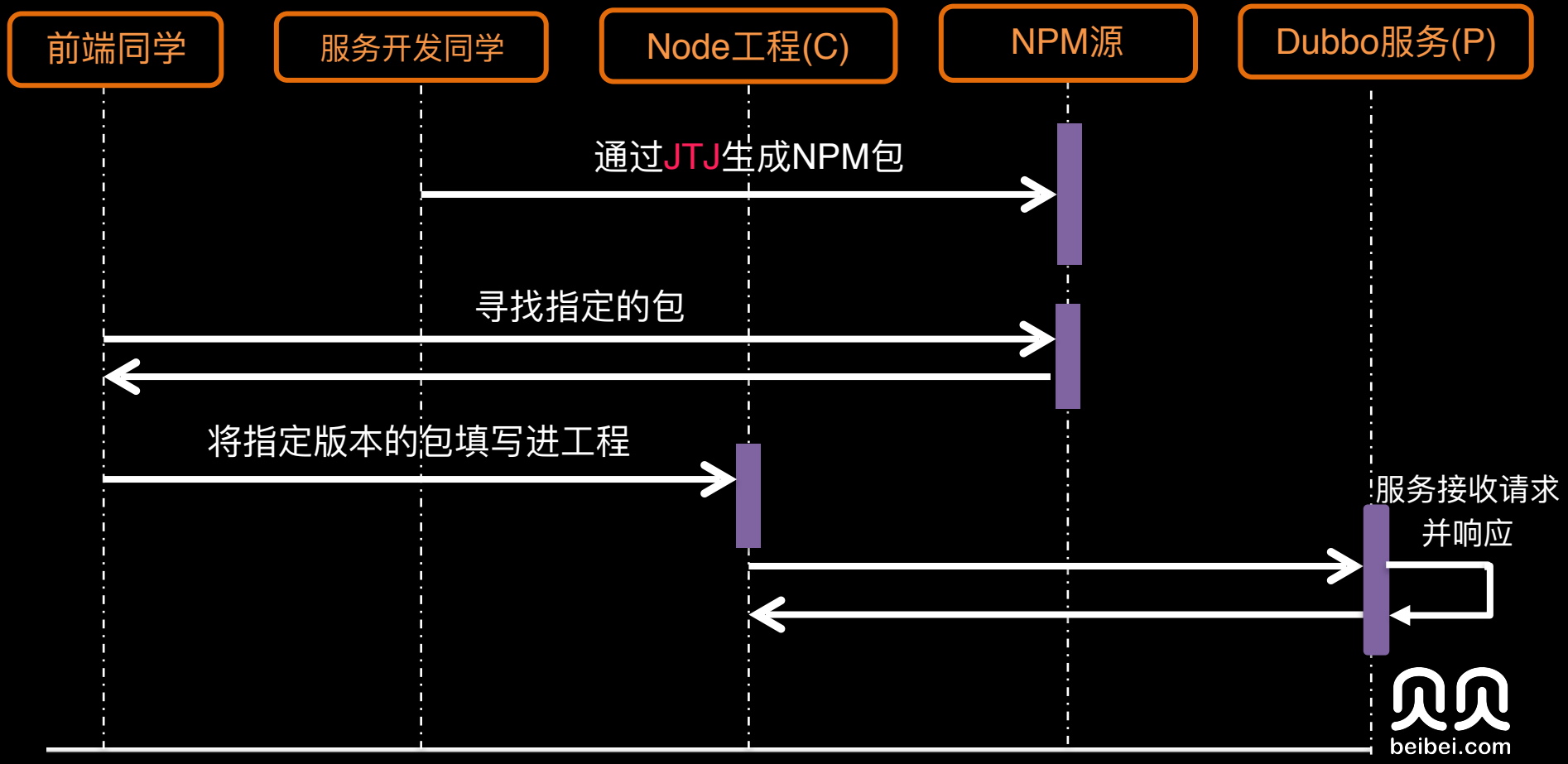
Node

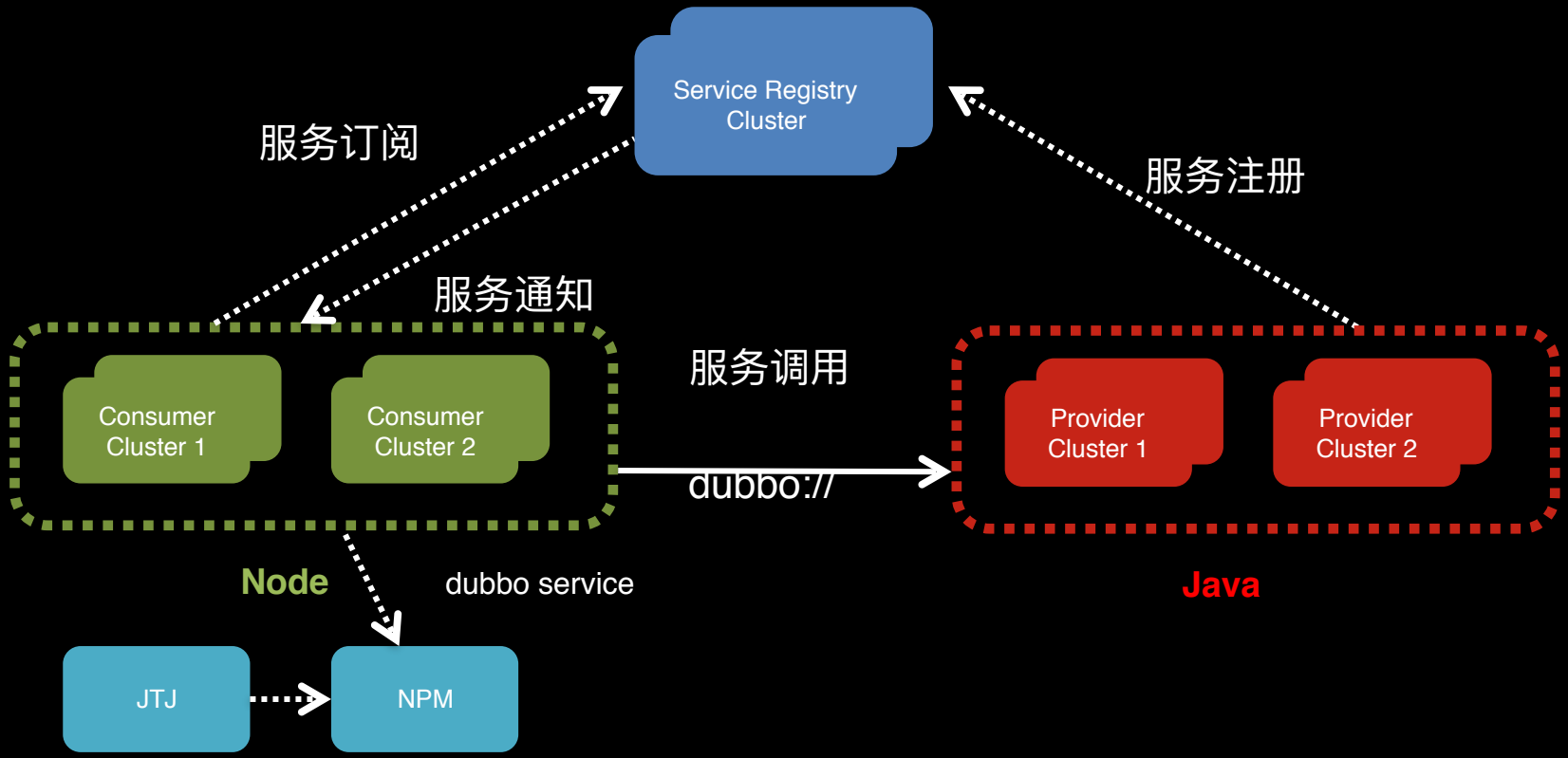
```node
services.TestService = () => proxyService({
        dubboInterface: 'com.service.TestService',
        methods: {
            /**
             * 给定id,获取所有对应信息
             * @param id 测试id
             */
            getByid(id) {
                return [
                    java.Long(id)
                ];
            },
            /**
             * 给定ids,获取所有对应信息列表
             * @param ids 测试id
             */
            listByIds(ids) {
                ids = ids.map((v) => java.List<Long>(v))
                return ids;
            },
        },
});
```


Java Dubbo
服务定义
Parse
AST(抽象语法树)
Transform
Generate
Node Dubbo
请求定义

JTJ

服务定义转换


前端同学
服务开发同学
Node工程(C)
NPM源
Dubbo服务(P)
通过JTJ生成NPM包
寻找指定的包
将指定版本的包填写进工程
服务接收请求
并响应
beibei.com


Service Registry
Cluster
服务订阅
服务注册
服务通知
服务调用
Consumer
Cluster 1
Consumer
Cluster 2
Provider
Cluster 1
Provider
Cluster 2
dubbo://
Node
dubbo service
Java
JTJ
NPM

5.0 规划

优点：
- 对开发者友好
- 无语言使用门槛

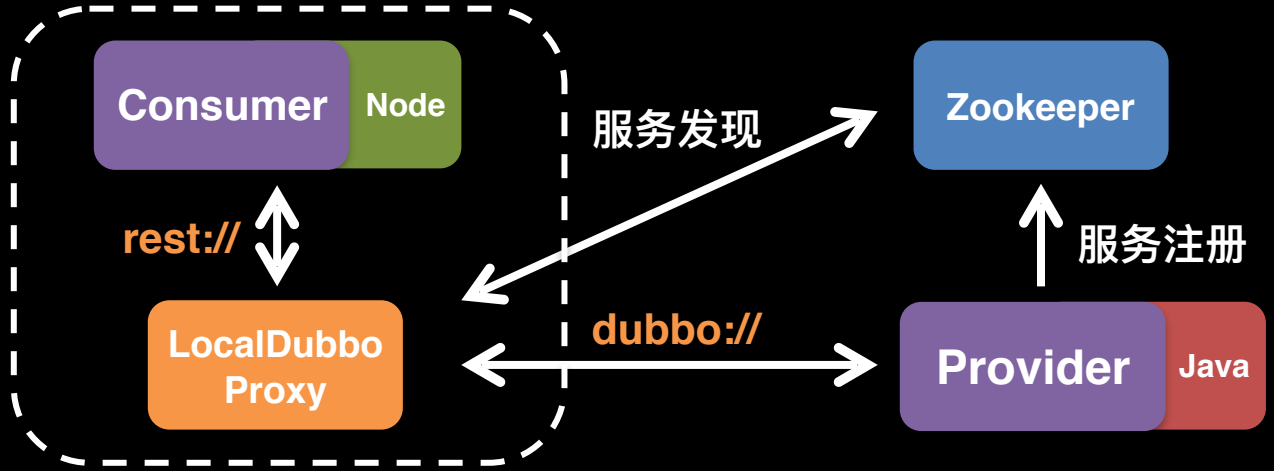
缺点：
- 短连接会反复建连
- 协议体较大，速度较慢

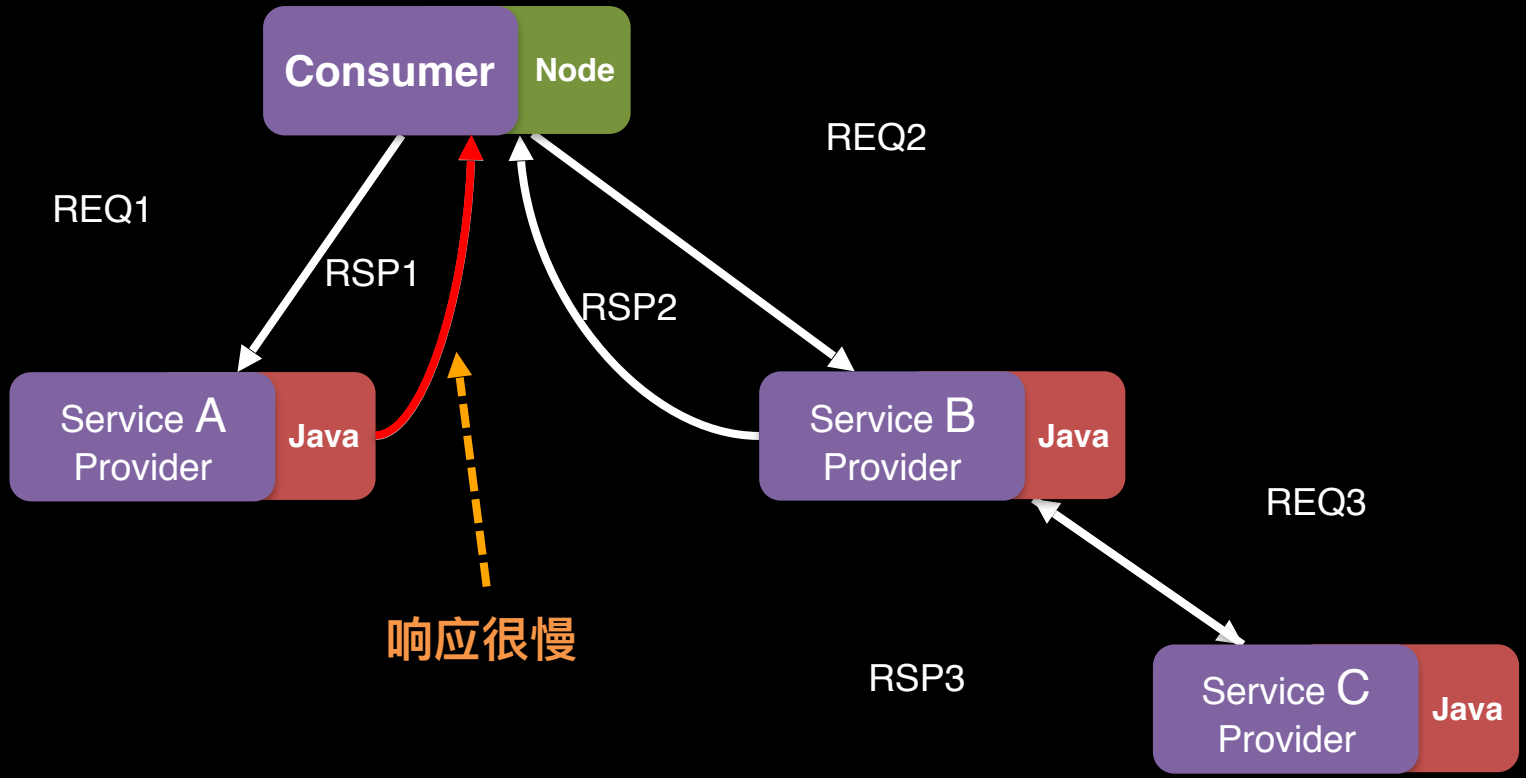
rest://

优点：
- 比Http性能更好

缺点：
- Node层承担的职能较重，例如socket维护，hessian解析.

dubbo://

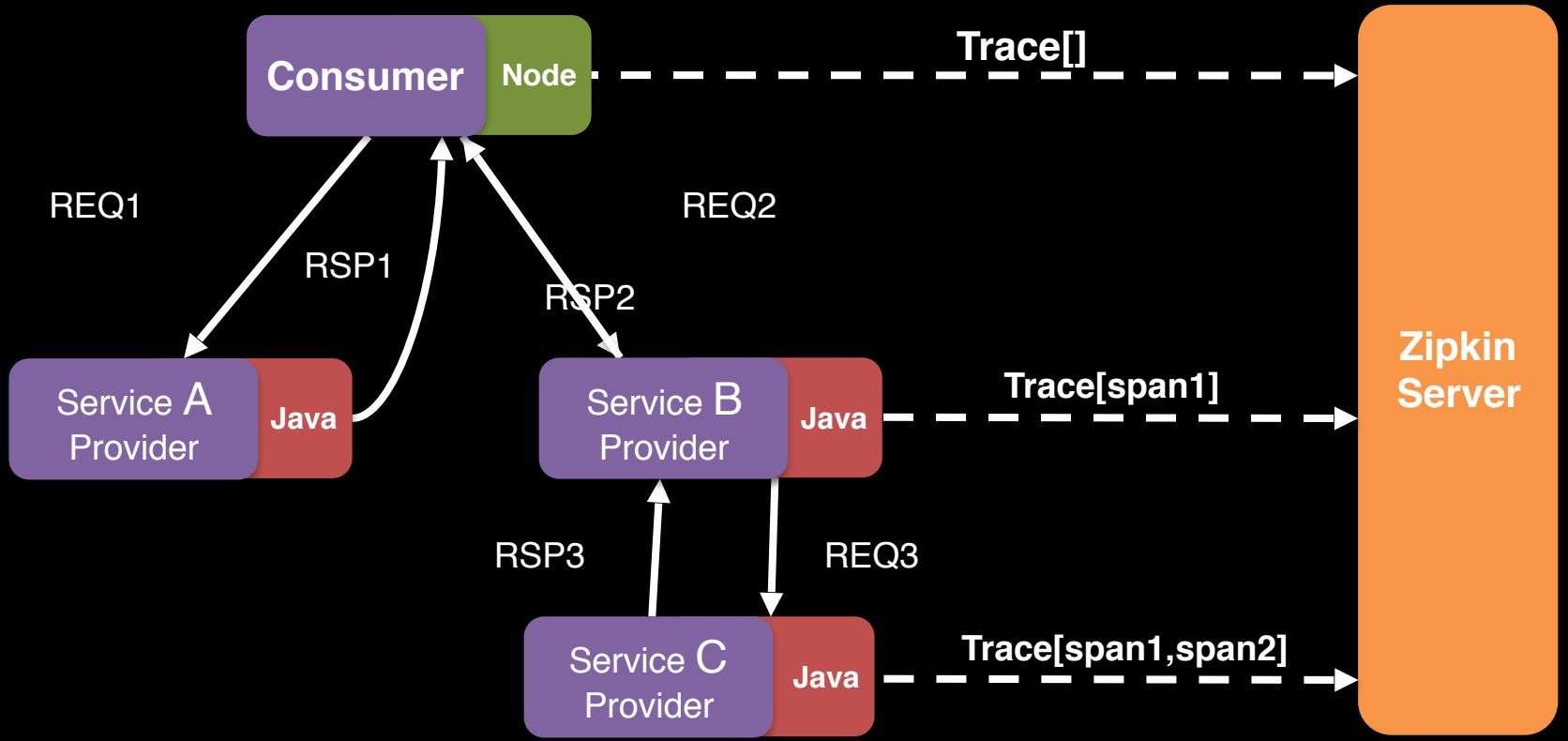



Consumer
Node
REQ1
RSP1
REQ2
RSP2
Service A
Provider
Java
Service B
Provider
Java
REQ3
响应很慢
RSP3
Service C
Provider
Java

进程数据 | 系统监控数据 | 异常日志 | 慢 HTTP 日志 | Trace (beta) | 模块依赖

ⓘ 链路追踪日志，展示您应用的慢 http 请求的具体慢调用原因

服务方	服务 & 操作名称	请求信息	响应用时	时间
+ localhost	**测试链路:** 根模块	"GET /delay 200"	12.08s	1 分钟前
	测试链路: 子模块 1: 随机并发延迟		9.07s	1 分钟前
	测试链路: 子模块 1: 随机并发延迟		6.14s	1 分钟前
	测试链路: 子模块 2: 延迟 3s		3.00s	1 分钟前
+ localhost	**测试链路:** 根模块	"GET /favicon.ico 200"	9.46s	1 分钟前


Consumer
Node
Trace[]
REQ1
RSP1
REQ2
RSP2
Service A Provider
Java
Service B Provider
Java
Trace[span1]
Zipkin Server
RSP3
REQ3
Service C Provider
Java
Trace[span1,span2]

Q & A

谢谢观看

可爱HR，在线招人

买母婴上贝贝