

如何打造高可用nodejs框架

张龙 QQ浏览器

个人介绍



张龙

- 2014年加入 QQ 浏览器
- 2014年负责 QQ 浏览器首页卡片开发
- 2016年负责 QQ 浏览器 Feeds 后台开发
- 2018年负责 QQ 浏览器 Feeds Hippy 开发

内容大纲

01 思维转变

02 框架目标

03 如何打造

04 未来展望

思维转变



高性能（秒开）

稳定性（无BUG）

一致性（强一致）

可信度（信赖）



高性能（TPS高）

高可用（柔性可用）

一致性（最终一致性）

可信度（皆不可信）

■ 框架目标

支持异构

支持 C++/Nodejs/Java/PHP/Go 等多语言

解决方案

提供涉及开发、运维、测试的一整套解决方案，帮助产品或者服务快速开发、部署、测试、上线

统一应用框架

组件丰富

具备编解码协议、高性能RPC通信框架、名字路由与发现、发布监控、日志统计、配置管理等必要组件

快速构建

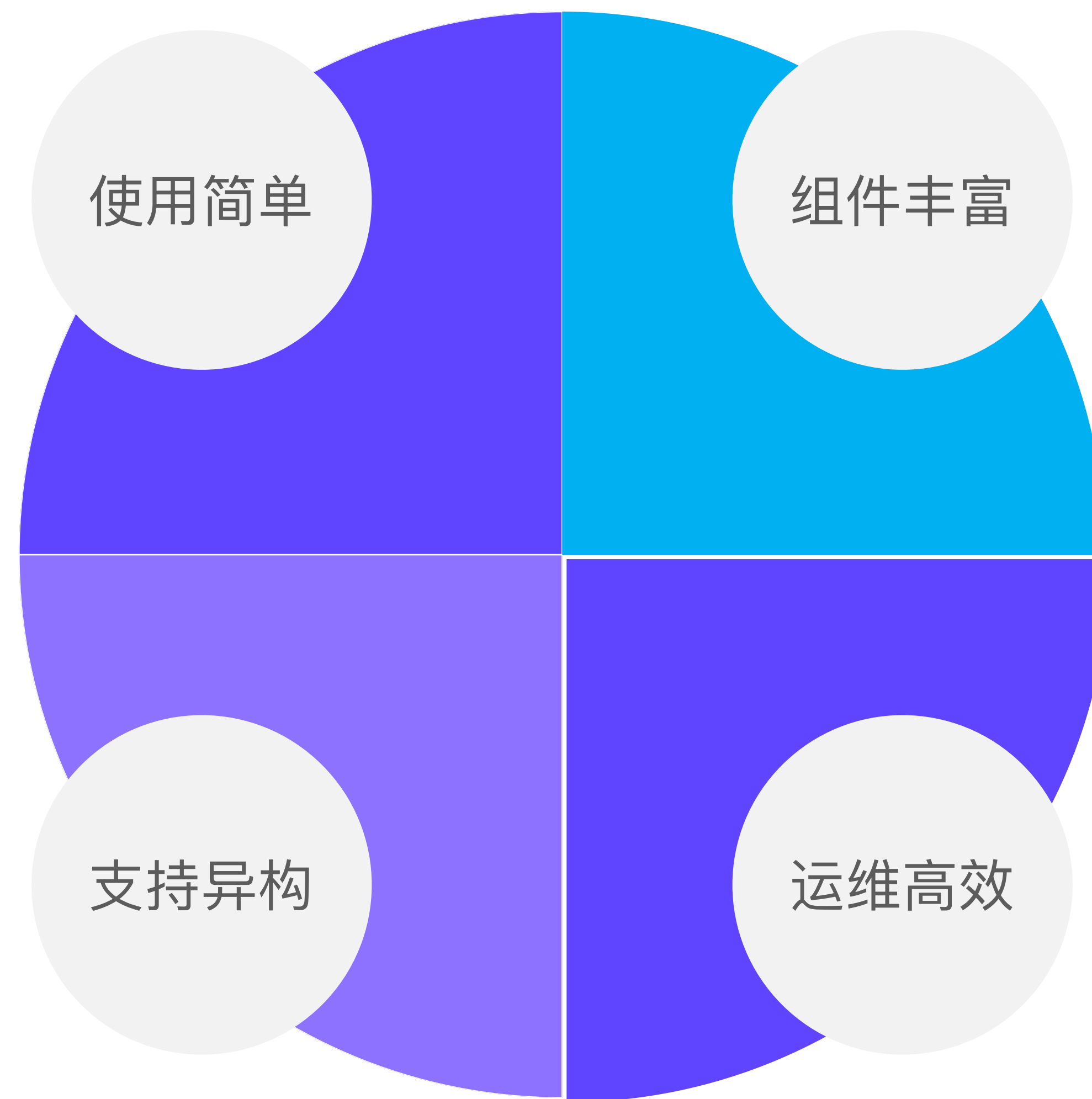
可以快速构建自己的稳定可靠的微服务

服务治理

实现完整有效的服务治理

Nodejs框架

- 上层开发只需聚焦业务逻辑
- 兼容业界常用框架，接入简单
- 一站式平台解决方案



核心模块

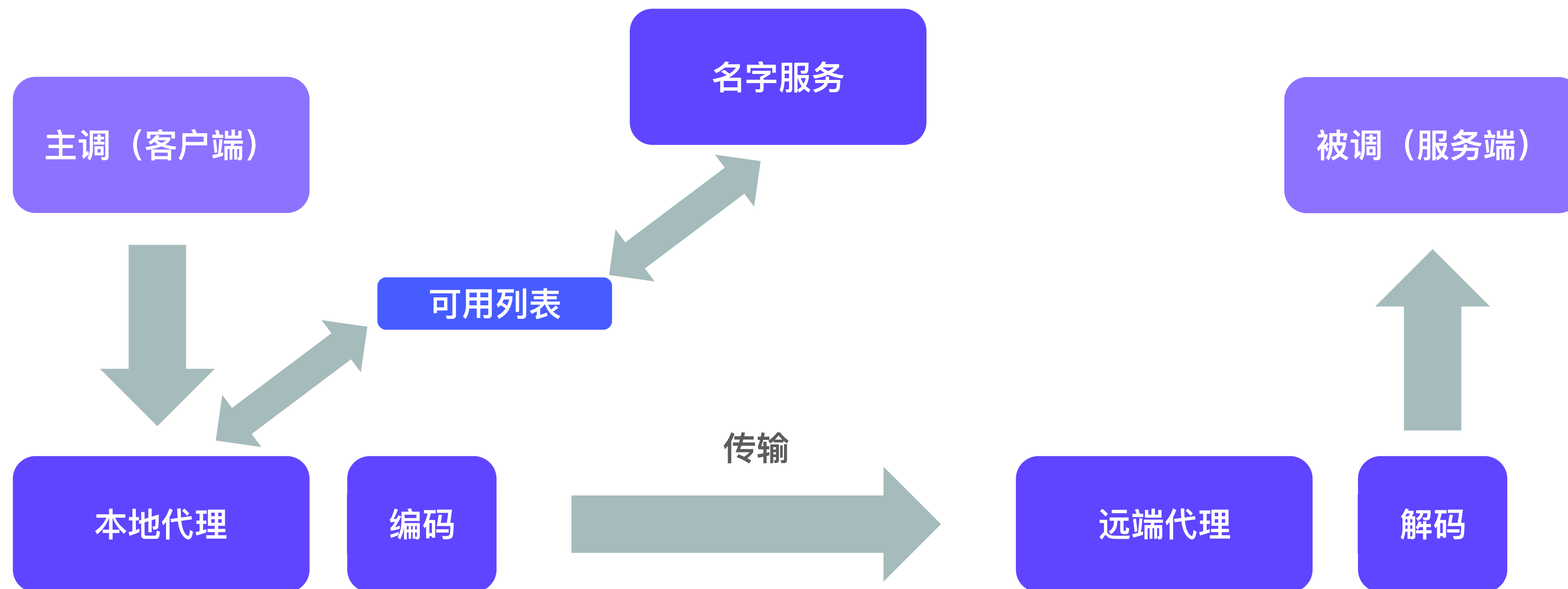
RPC调用

传输协议

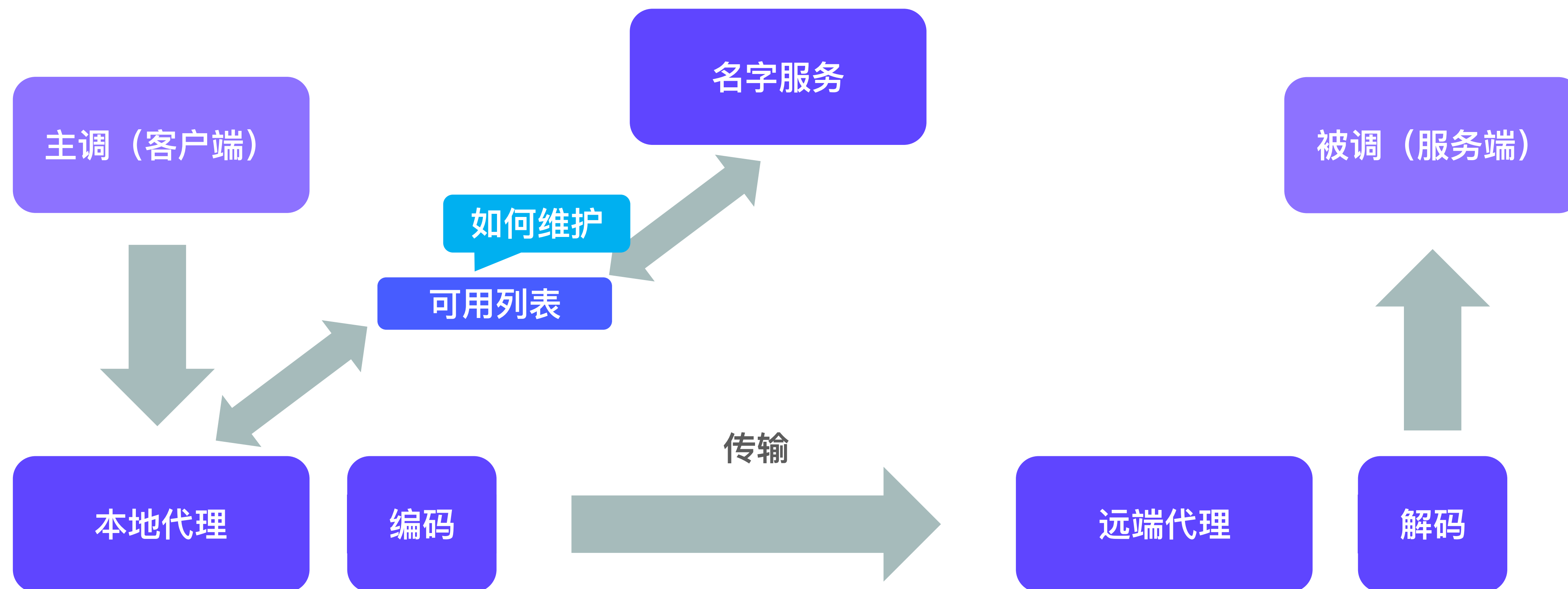
编解码

远程调用同本地调用一样方便高效

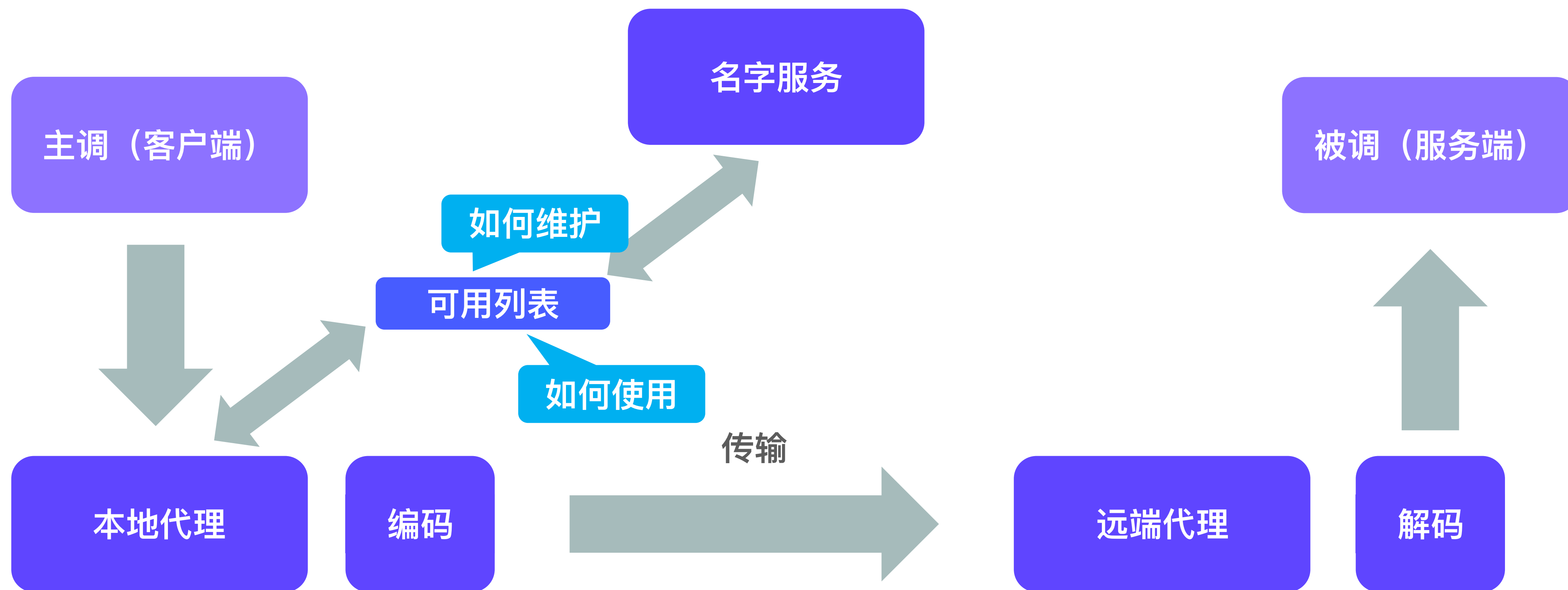
RPC调用



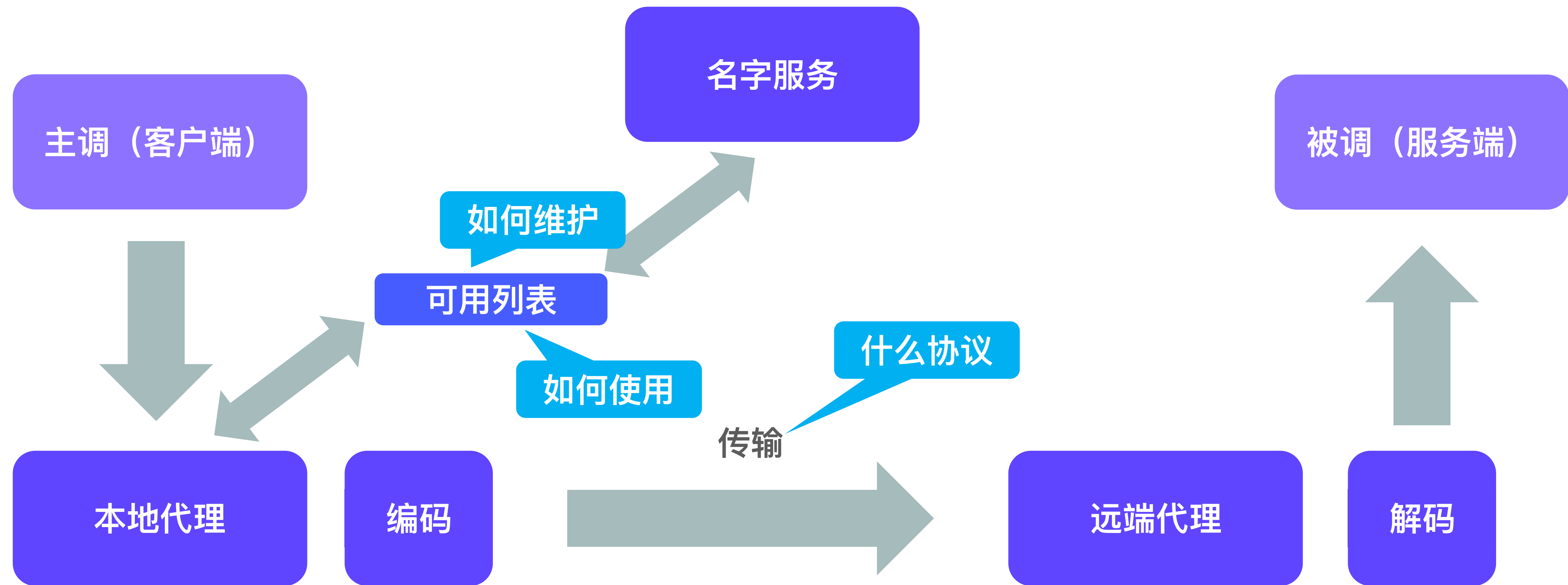
RPC调用



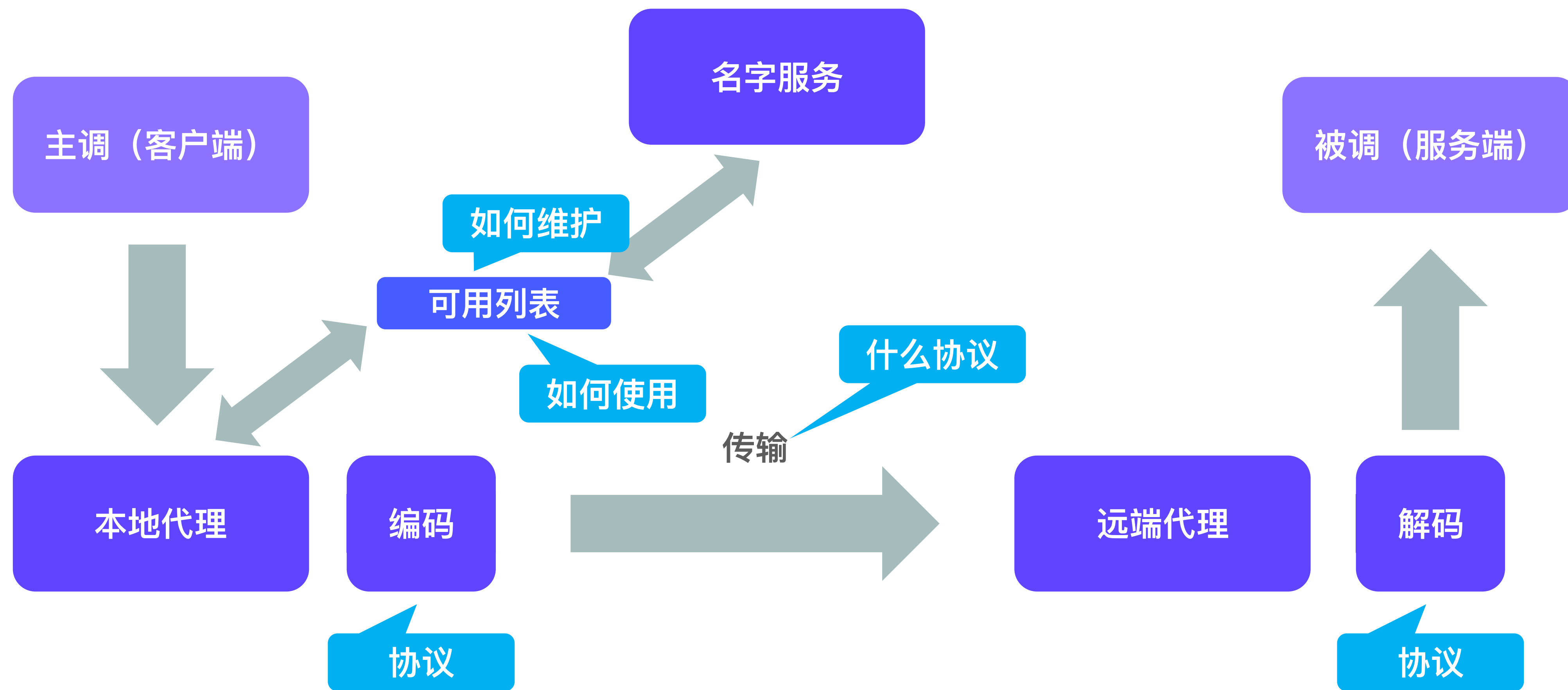
RPC调用



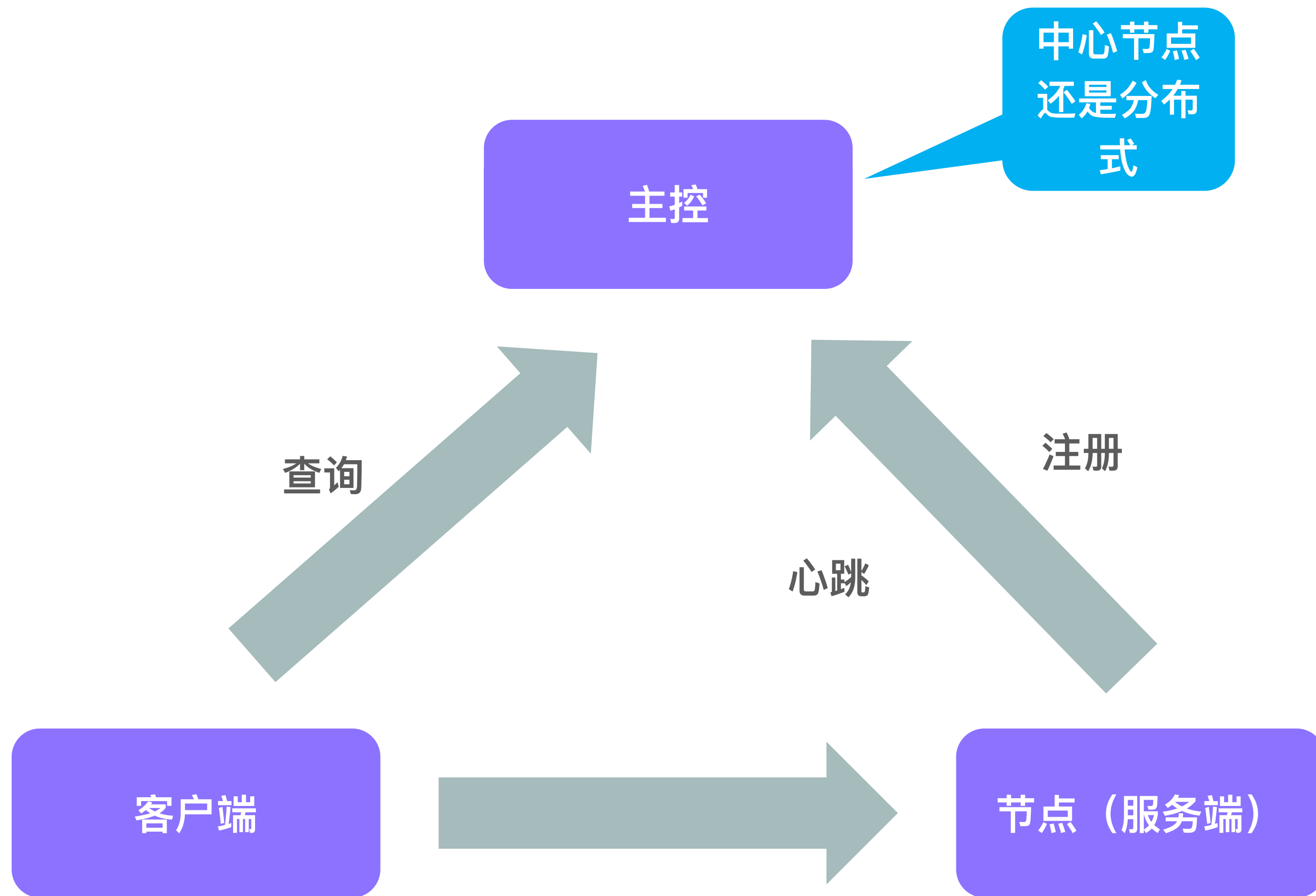
RPC调用



RPC调用



注册发现



中心节点

优势：查询高效，实现简单

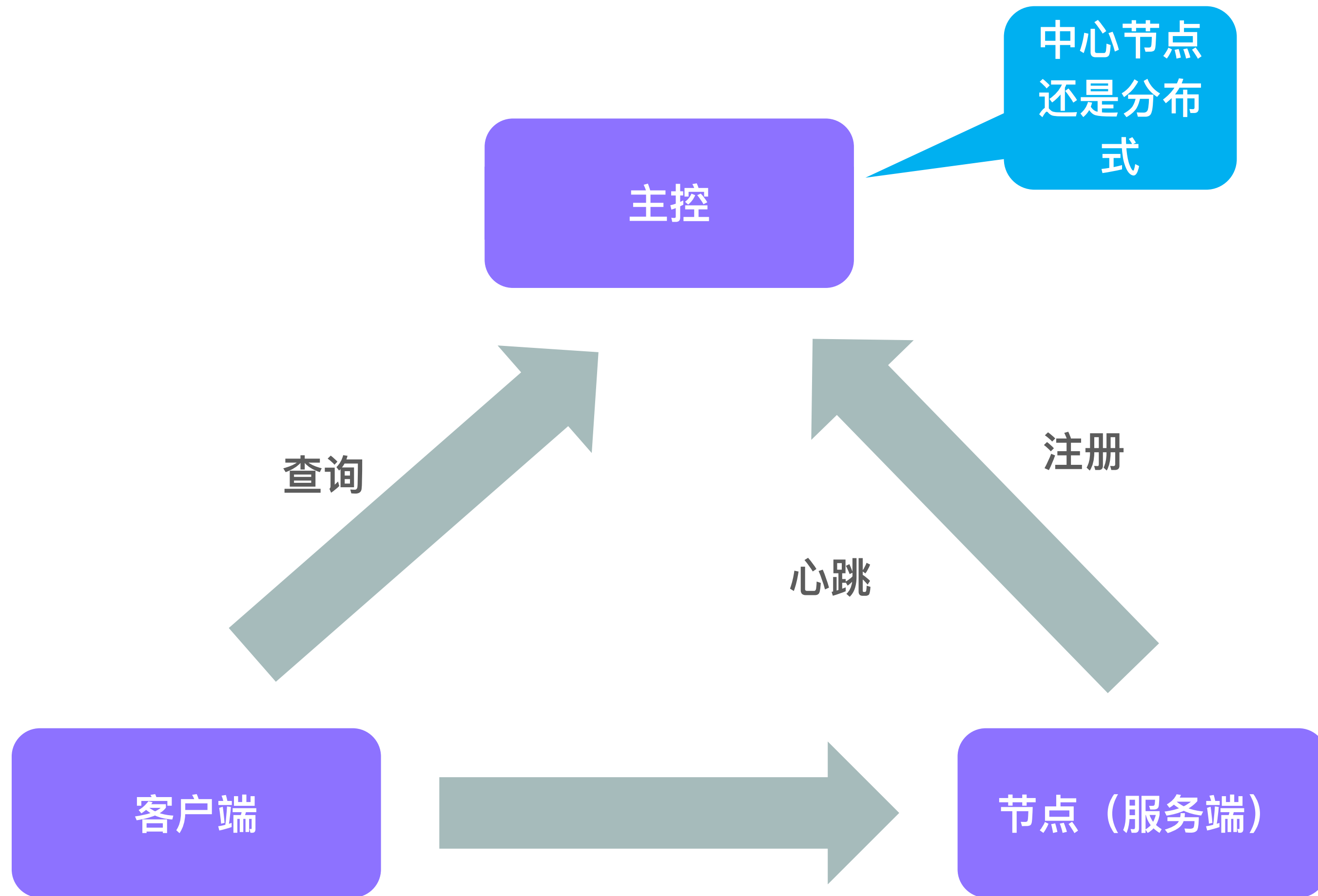
劣势：可靠性低

分布式

优势：可靠性高

劣势：性能较差，实现复杂

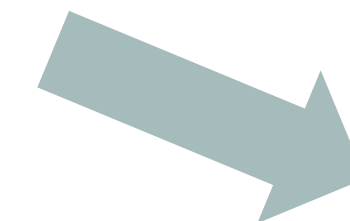
注册发现



中心节点

优势：查询高效，实现简单

劣势：可靠性低



主备

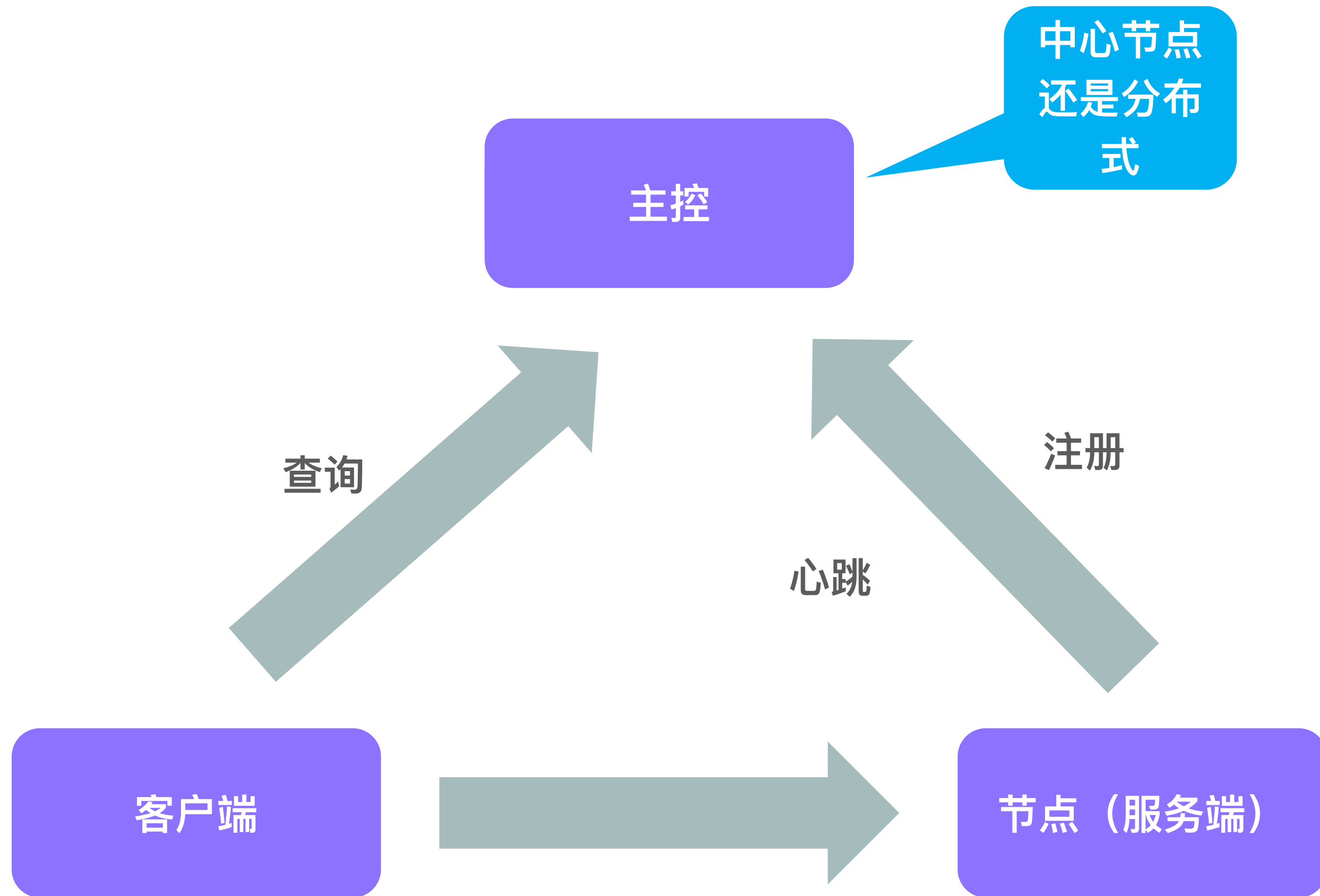
本地缓存

分布式

优势：可靠性高

劣势：性能较差，实现复杂

注册发现



中心节点

优势：查询高效，实现简单

劣势：可靠性低

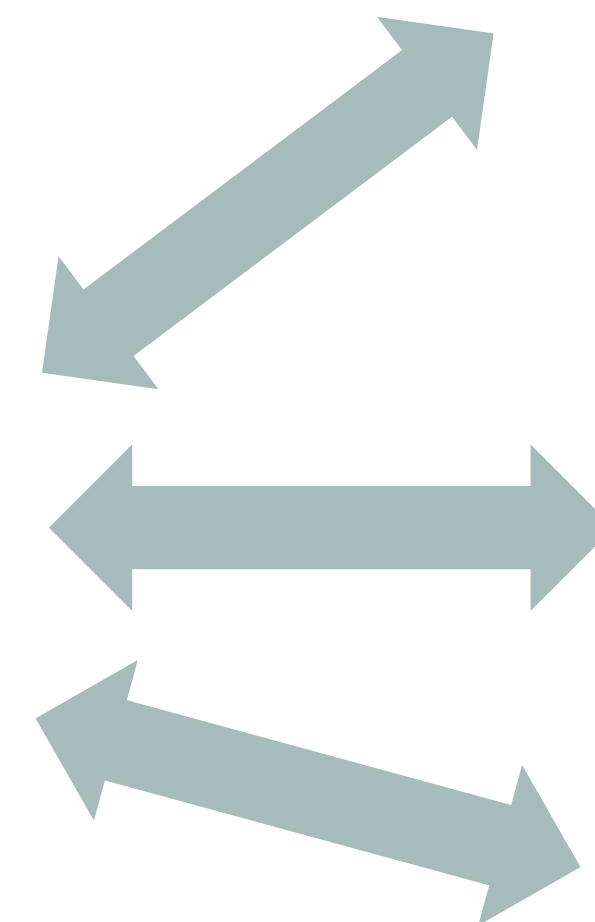
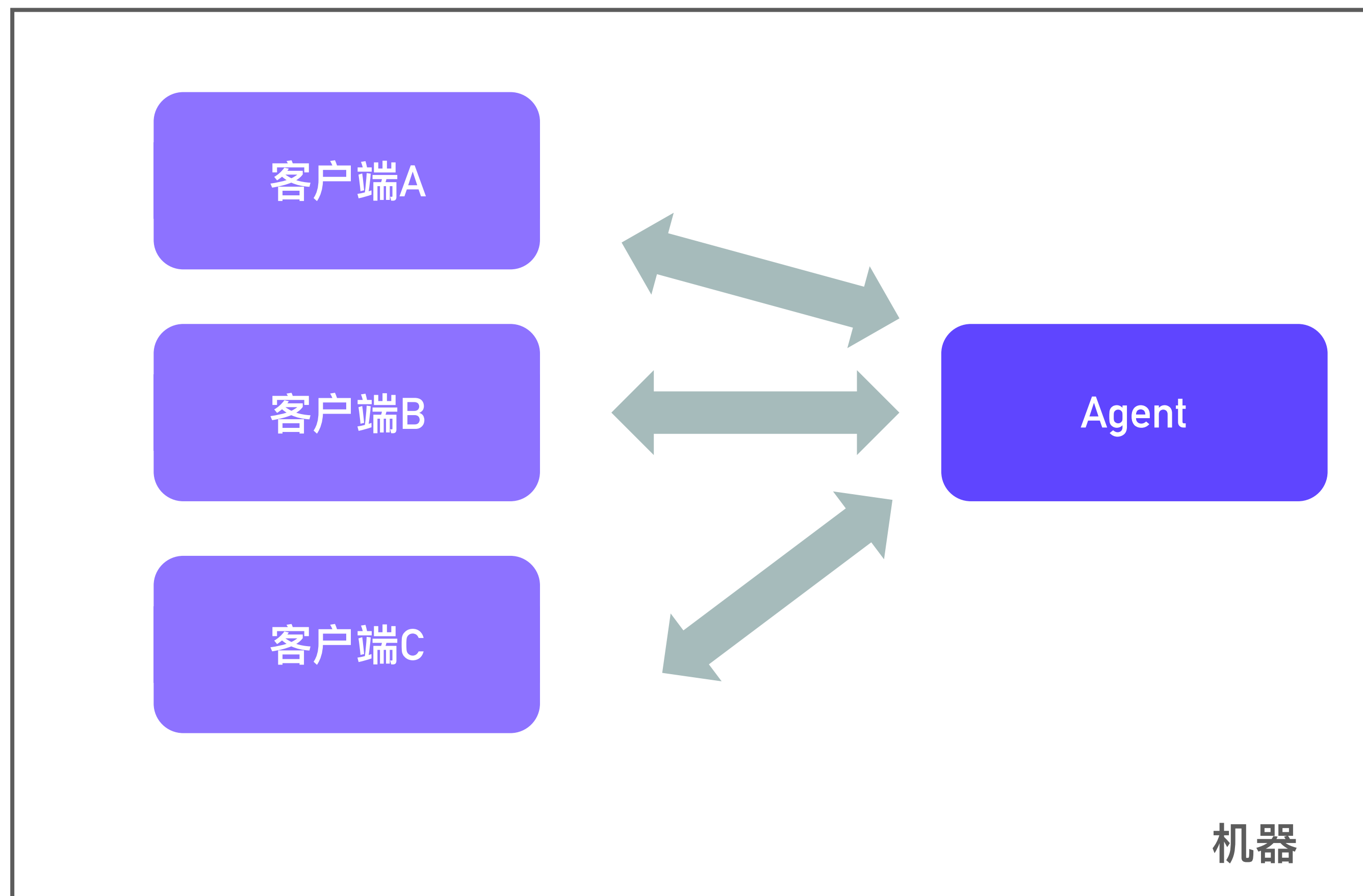
分布式

优势：可靠性高

劣势：性能较差，实现复杂



Agent模式



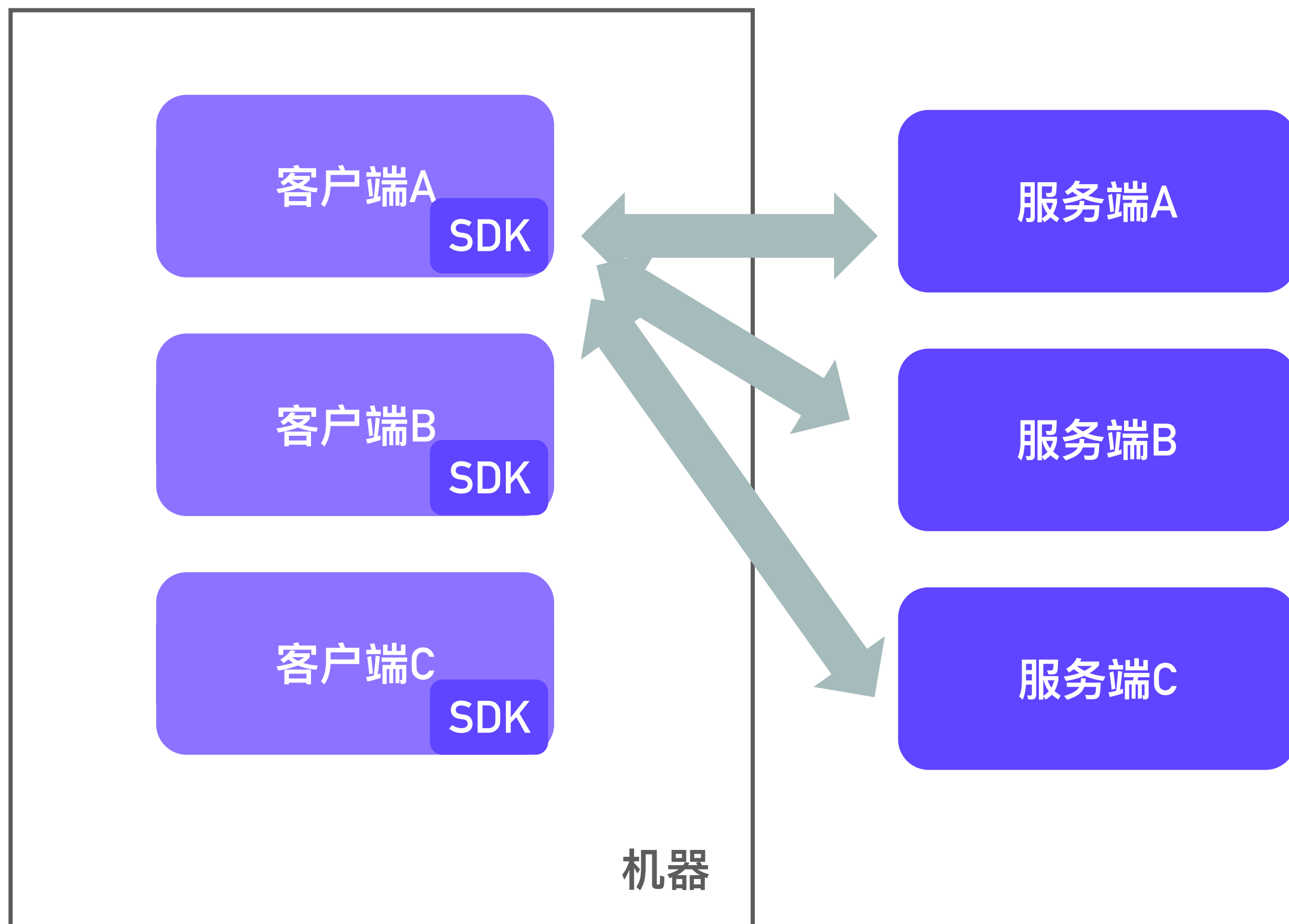
优势

1. 节省资源
2. 复用度高
3. 数据一致

劣势

1. 性能瓶颈
2. 稳定性差
3. 维护麻烦

SDK模式



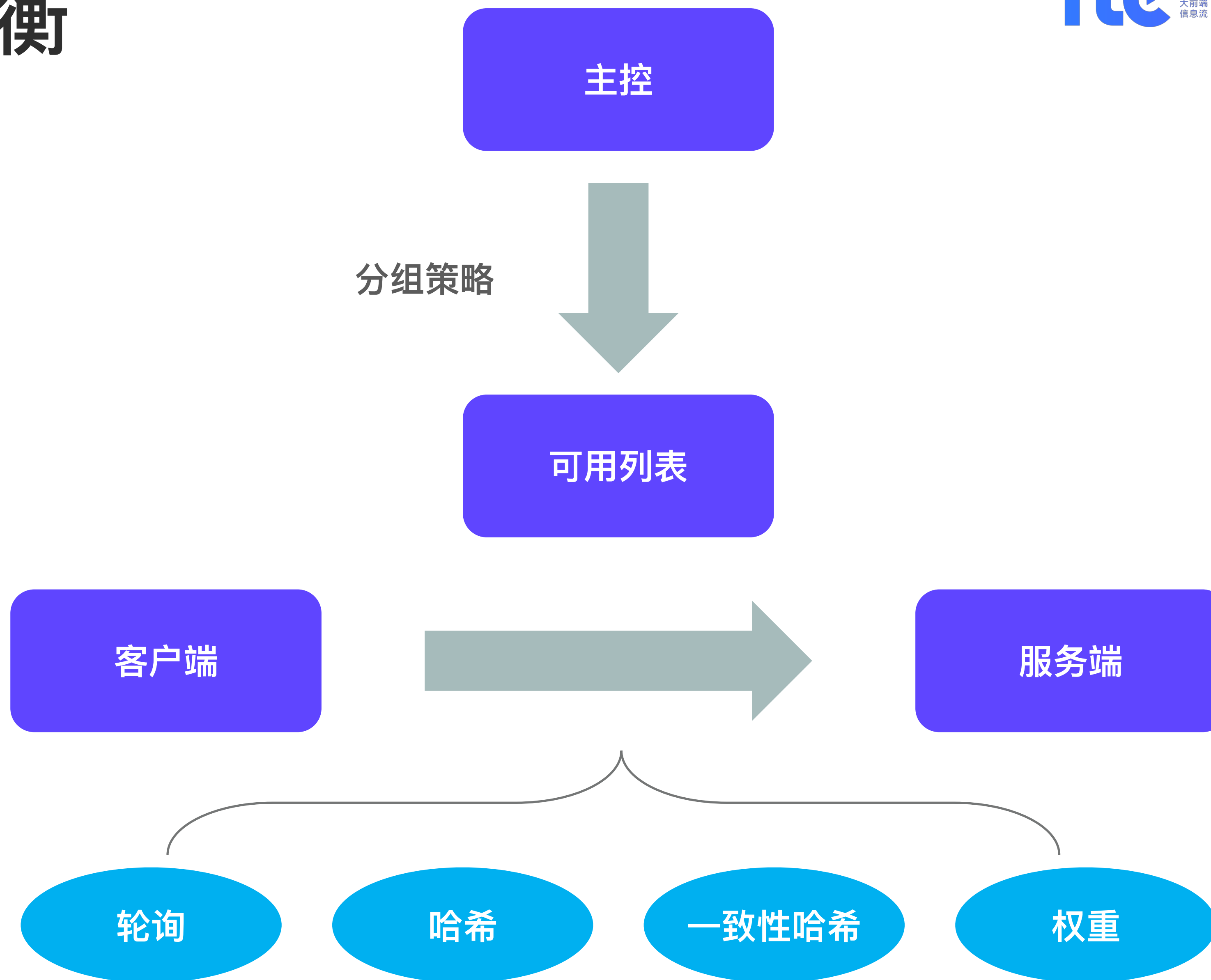
优势

1. 稳定性好
2. 性能更好
3. 部署简单

劣势

1. 浪费资源
2. 复用度低

负载均衡



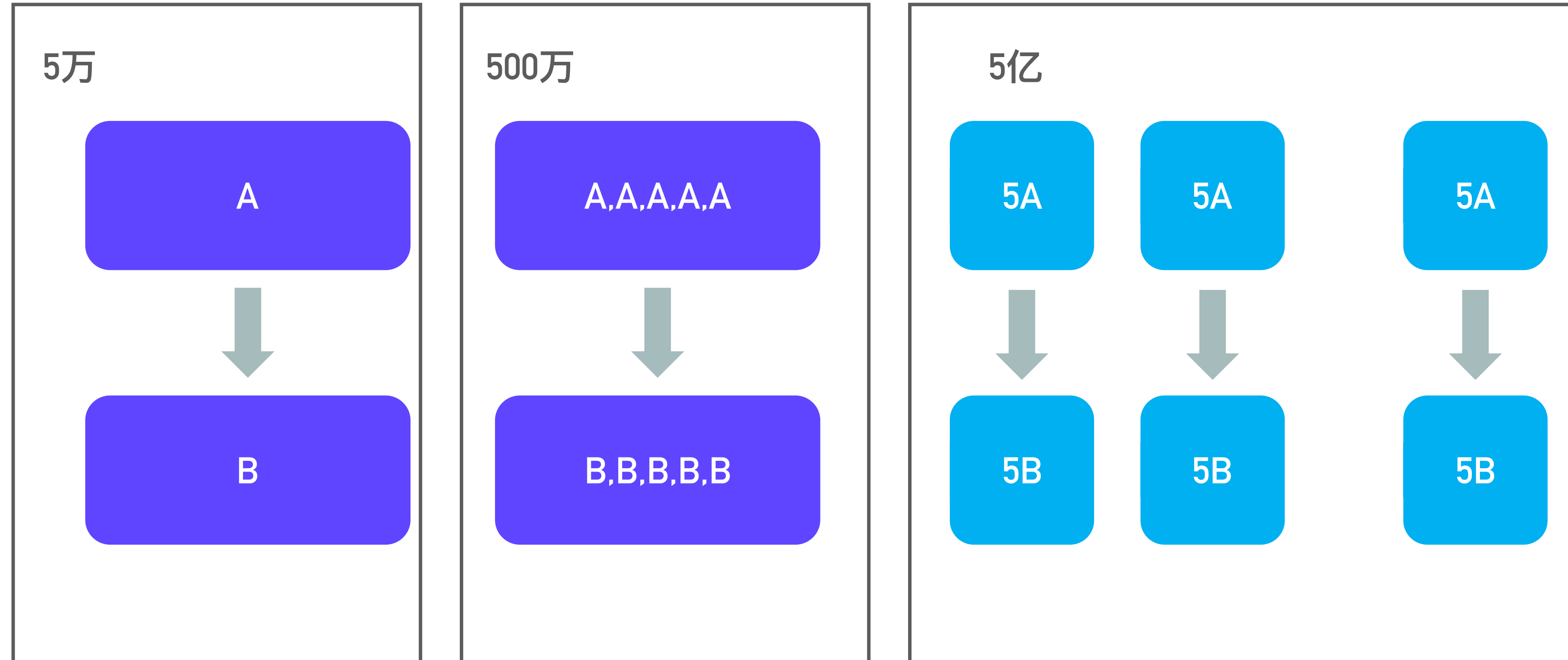
SET分组

目的

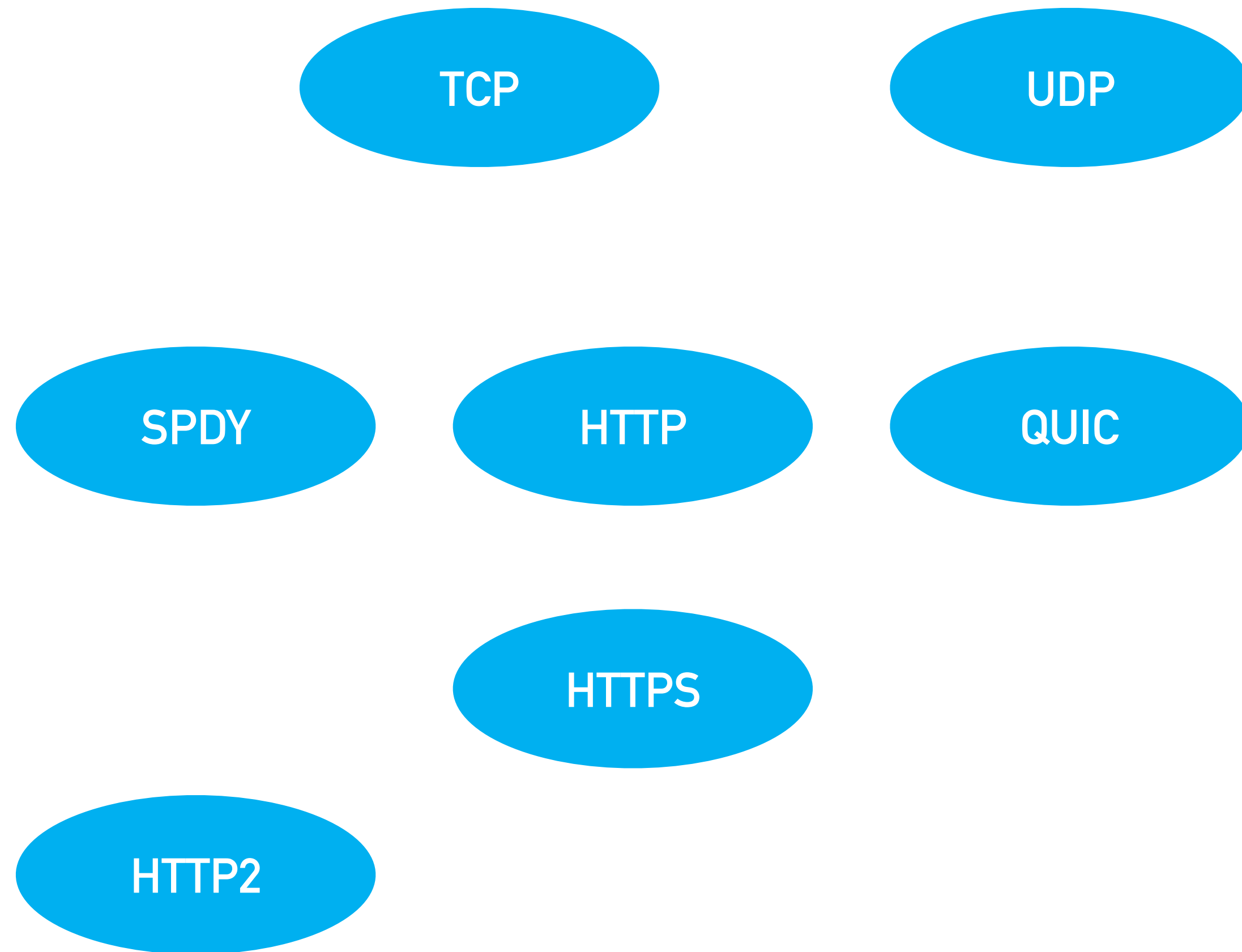
1. 方便管理
2. 分组隔离

作用

1. 服务部署管理标准化和容量化，提高运维效率
2. 各个分组之间相互隔离，互不干扰，提高服务可用性



传输协议



协议统一

分层治理

Protocol Buffer

扩展性强、体积小、编解码性能高、开源社区活跃

FlatBuffers

反序列化性能高、无额外内存消耗；可读性差，体积大，扩展性差，序列化性能低

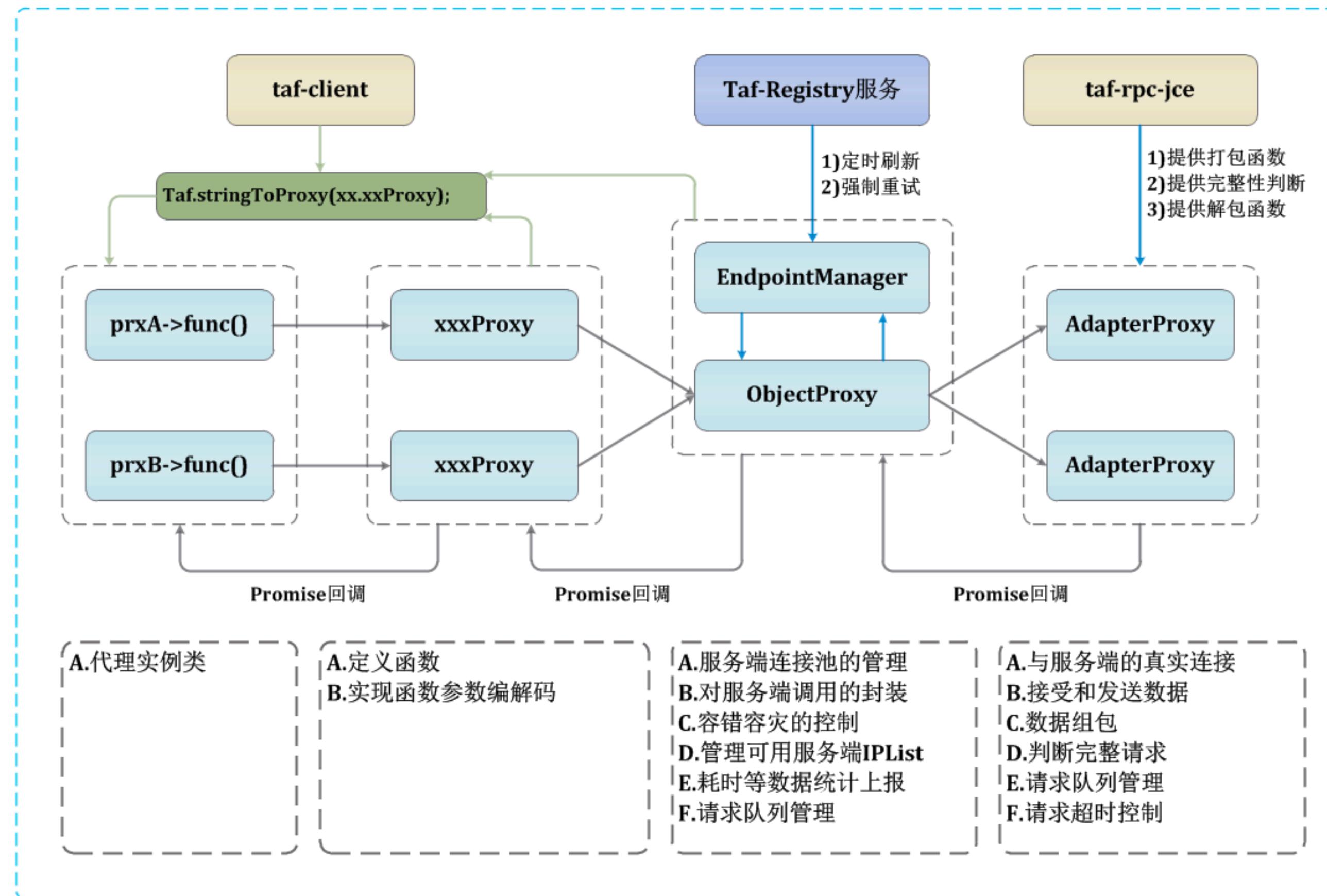
JSON

扩展性强、开发方便；体积较大，编解码性能低

JCE

扩展性强、体积小、编解码性能高；内部私有协议，不能充分利用开源力量

TAF-RPC



异常保护

容错保护

异常检测

过载保护

容错保护

节点排除

主控会根据服务的心跳情况返回对应的可用列表

主动屏蔽

客户端会根据被调服务的情况进行快速故障屏蔽

异常判定

1. 在 60s 内，超时调用次数大于等于 2，超时率大于 50%
2. 连续超时次数大于 5

异常恢复

1. 在每隔 30s 重试，调用成功则恢复

异常判定

1. 在 60s 内，超时调用次数大于等于 2，超时率大于 50%
2. 连续超时次数大于 5

以上参数
皆可配置

异常恢复

1. 在每隔 30s 重试，调用成功则恢复

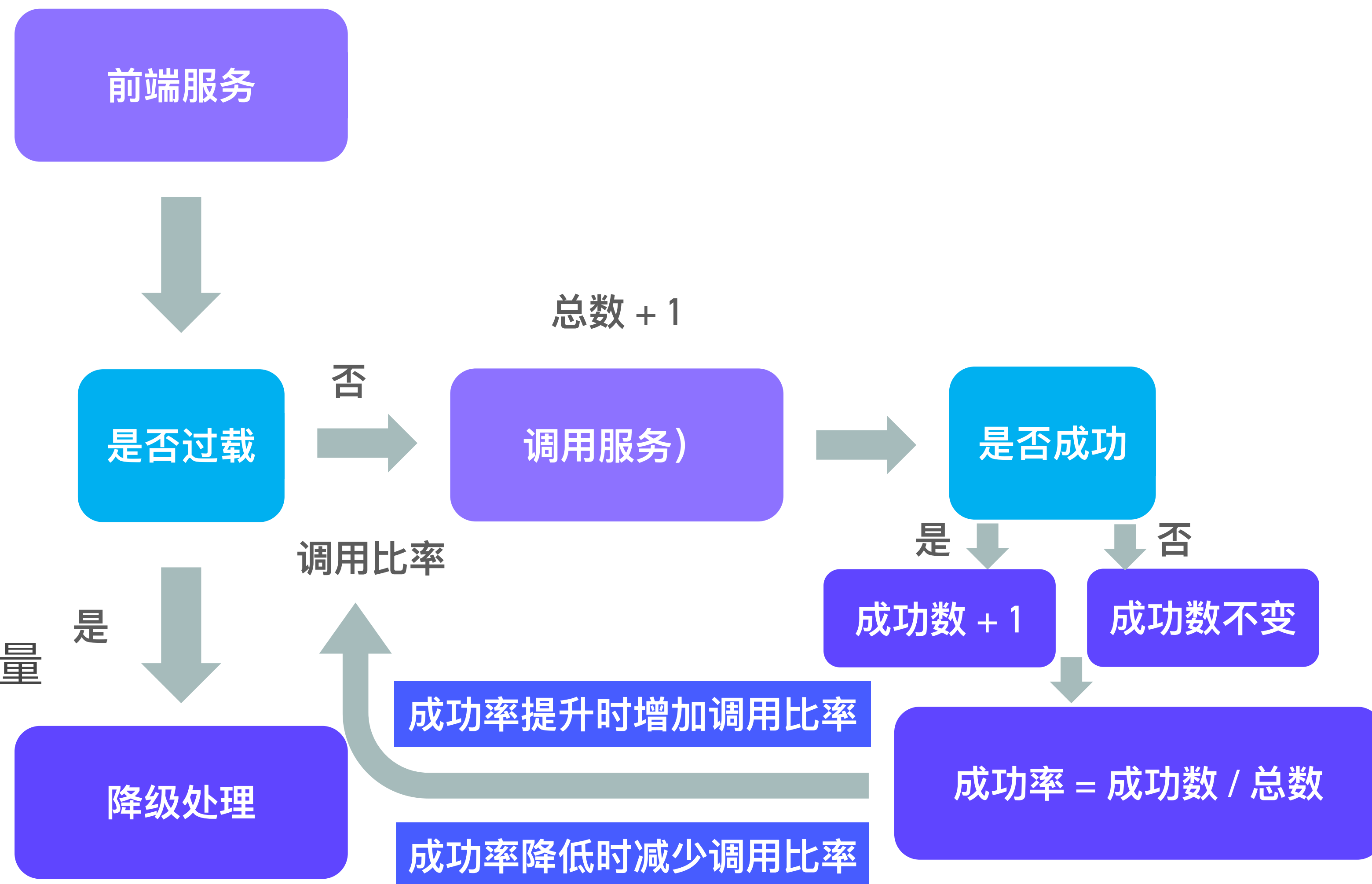
过载保护

保护自身

令牌桶

保护后端

根据请求成功率动态调整对后端流量



运维能力

监控告警

运维管理

弹性调度

监控

1. 服务监控：框架提供基本服务监控参数上报，方便了解服务运行情况
2. 特性监控：taf-nodejs 提供基础特性上报，方便追查问题
3. pp监控：提供强大多维度的业务定制化监控

告警

1. 阈值特性/波动特性
2. 多维度
3. 屏蔽策略强大

运维管理

服务管理

服务申请/发布管理/服务配置/监报告警/数据报表

运维操作

服务审批/服务部署/资源回收/服务扩容/模板管理

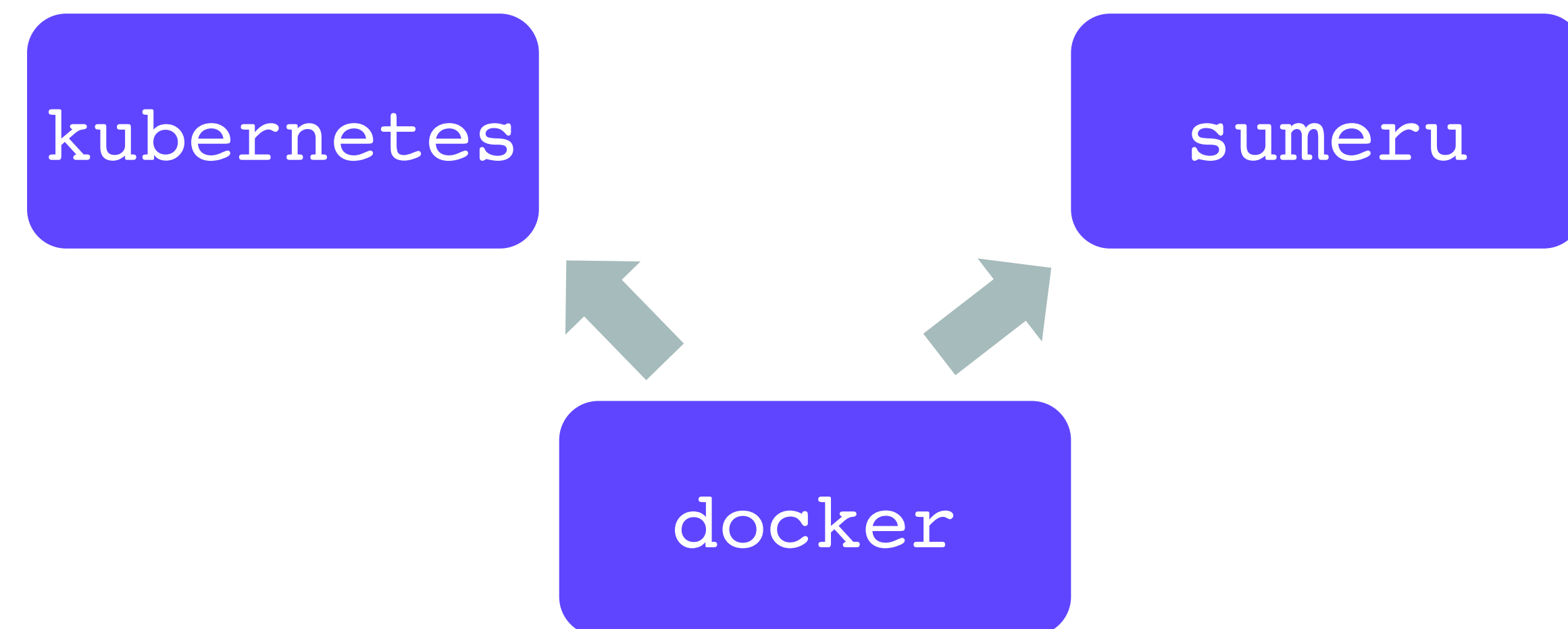
弹性调度

自动化

制定策略，自动化运维

智能化

针对不同场景使用不同策略



构建发布

持续集成

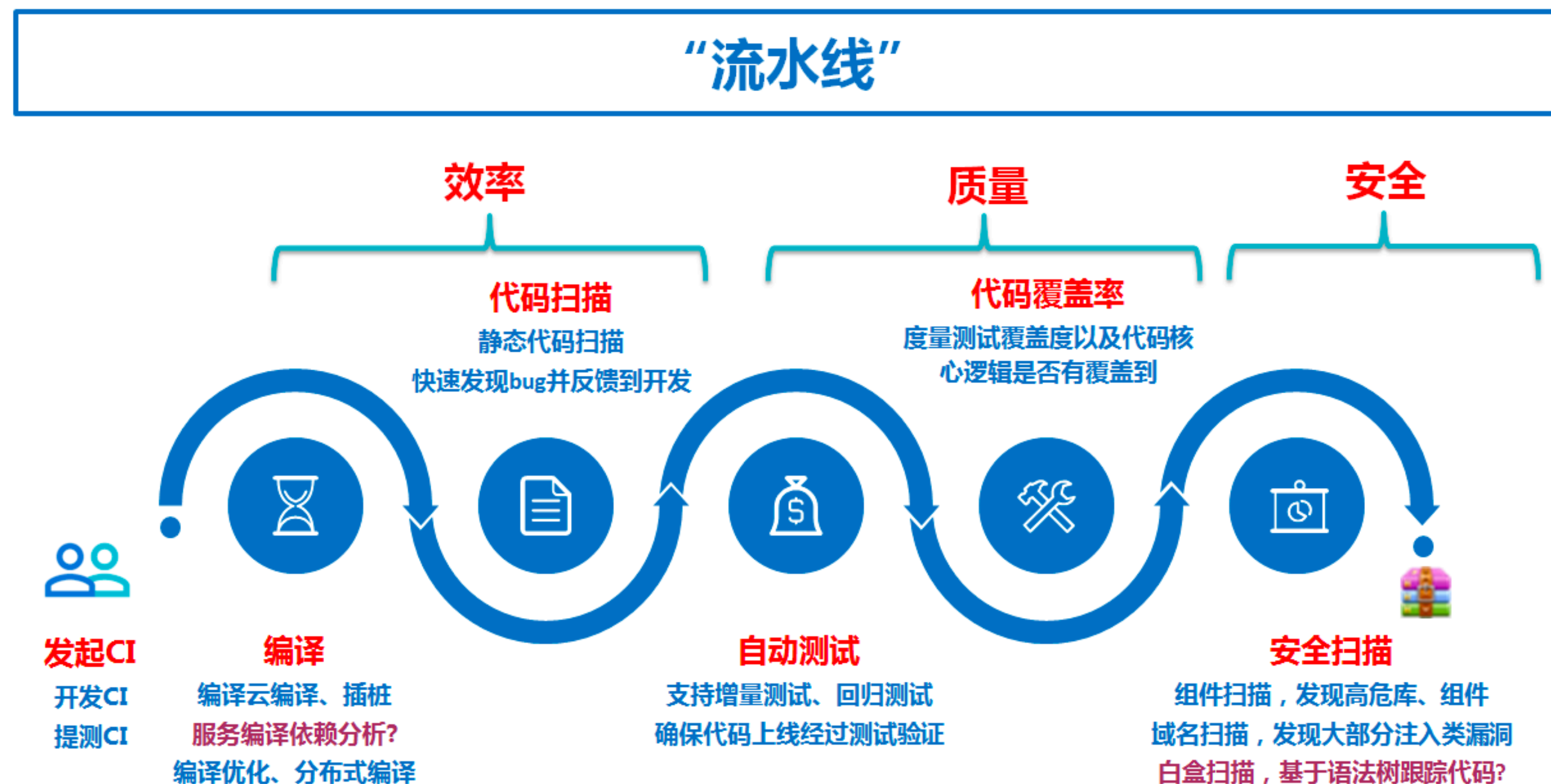
持续部署

配置管理

持续集成

CI

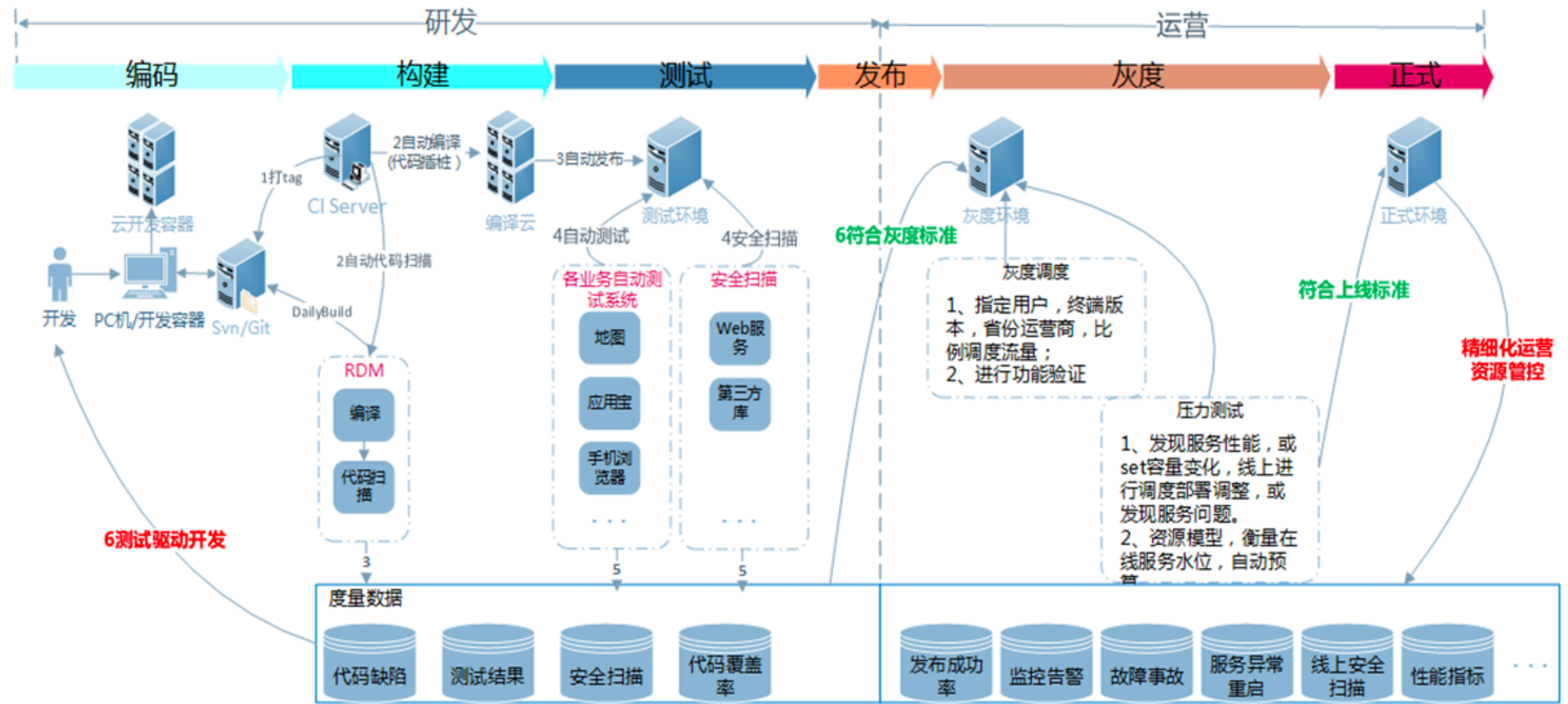
自动化构建，尽早发现集成错误



持续部署

CD

持续交付/自动部署



配置管理

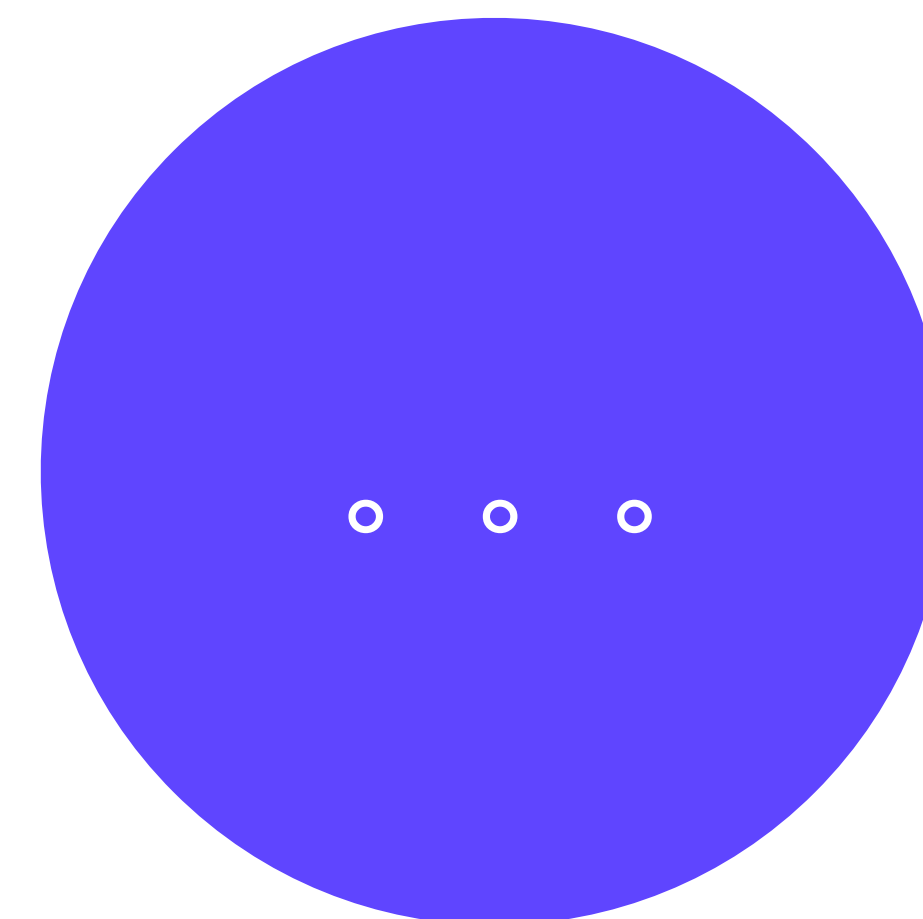
使用便捷

1. 用服务来管理配置
2. 添加、编辑、删除、回滚操作方便

功能强大

配置模板/私有配置/参数配置/动态push

公共组件



未来展望

1. 分层治理会更加完善，耦合更少
2. 公共组件更多
3. 开源共建，更快更稳定

Thank you for your attention!

pollyzhang