为什么要做系统拆分：

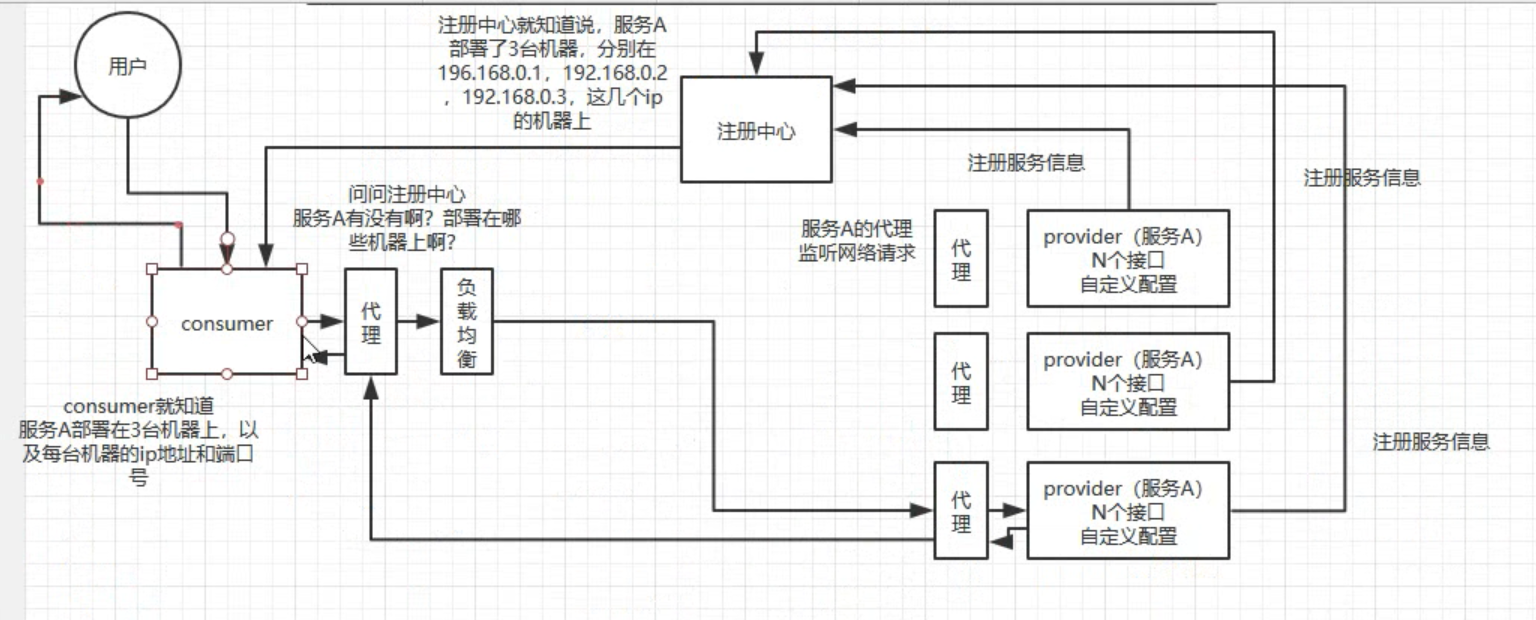
1. 降低维护成本-解耦

怎么拆：

1. 拆很多轮

Dubbo是一种RPC框架

面向接口的远程方法调用，智能容错和负载均衡，超时重试等等，以及服务自动注册和发现



Dubbo原理：(十层)

1. service层：provider与consumer接口 自己实现
2. config层：配置文件
3. proxy层：代理层 给provider与consumer生成代理 代理之间通信
4. registry层：注册层 provider在注册中心注册为一个服务，consumer就可以从注册中心找到并调用
5. cluster层：provider可以注册在多台机器上 组成一个集群
6. monitor层： 监控层 信息统计
7. protocol层：负责具体的provider与consumer之间的通信
8. exchange层： 信息交换层
9. transport层：网络传输层 抽象mina与netty为统一接口
10. seriallze层：序列化层

工作流程;

1. provider注册到注册中心
2. consumer从注册中心订阅服务
3. consumer调用provider
4. consumer与provider都将信息异步同步给监控中心

dubbo支持的通信协议与序列化协议：

1. 默认是dubbo协议 单一长期连接 NIO异步通信 序列化：hessian 支持跨语言
   1. 适用于：并发高 传递信息小(100k以内)

Dubbo的负载均衡;

1. provider是一个集群时，consumer调用就需要负载均衡策略
2. random load balance 随机调用 可以配置权重 实现负载均衡(默认)
3. roundom robin 轮询 也可以配置权重（根据机器性能调整）
4. leastactive 自动感知 机器性能越差 接收的请求越少
5. consistanthash:相同的参数请求 分发到同一个provider上

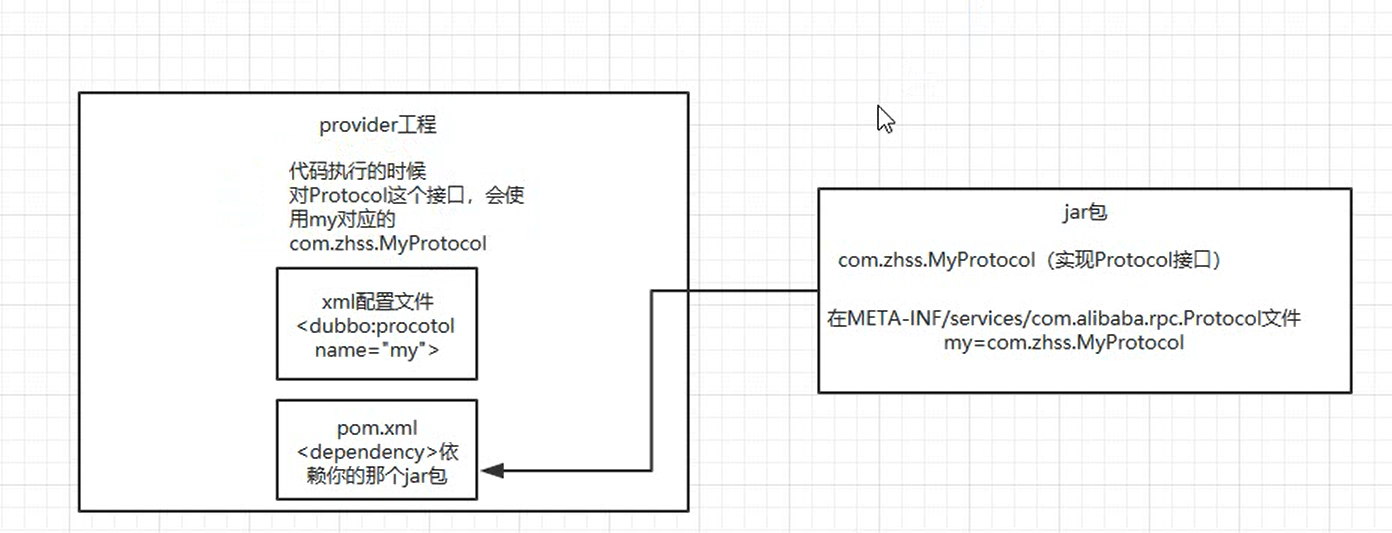
dubbo集群容错;

1. failover cluster：失败自动切换 自动重试其他机器(默认规则)
2. failfast：一次调用失败 就立即失败
3. failsafe cluster: 调用失败就忽略 常用在不怎么重要的接口
4. failback cluster 调用失败后自动记录 定时重试 可以用在消息队列中
5. forking cluster 并行调用多个provider 只要有一个返回成功

dubbo设计理念：微内核 可插拔 多插件

dubbo SPI机制

dubbo对很多组件都留了一个接口 以及多个实现，然后在系统运行的时候根据配置动态的去找到对应的实现类



Dubbo 服务治理

多应用拓扑分析 响应时长与占比分析 监控等等

Dubbo降级：

基于Mock做的

分布式顺序性保障：(是否需要100%保证顺序)

否：

1. 使用一致性hash分发负载
2. 内存队列承接

是：

1. 引入分布式锁 zk 比较重的框架
2. 同一批的请求 有顺序标记 获取分布式锁 操作数据库的时候 进行验证

如何设计一个RPC框架？

1. 注册中心 消费者 提供者 代理层 通信
2. 负载均衡 序列化协议 长连接 NIO SPI 容错降级