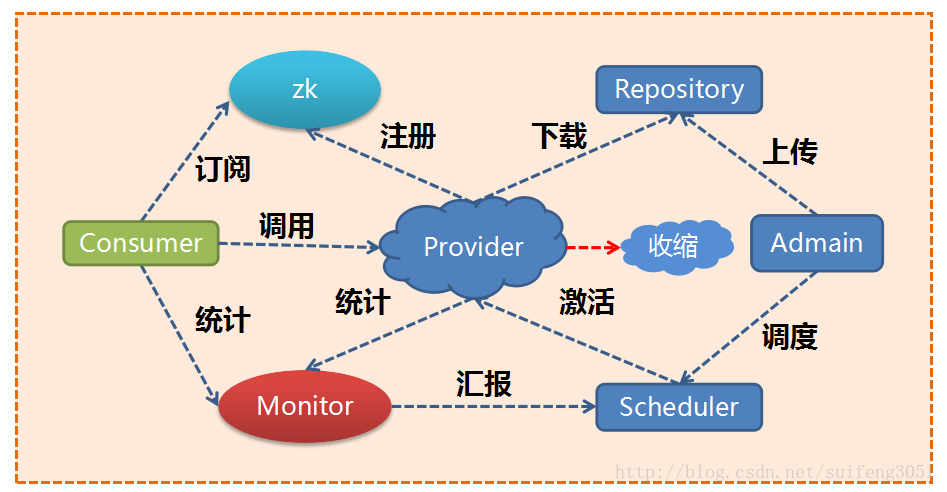
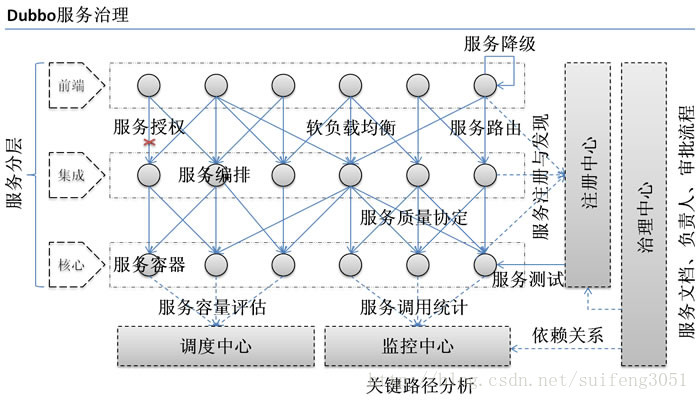
这里简单介绍一下弹性云的概念，微服务要想提供好服务，保证API不能挂掉并且有好的性能，需要很高的运维要求。这里的弹性云便是自动化运维解决方案，对访问压力进行监控，根据监控解决调度应用的发布和回收。

**Repository:仓库中心。Scheduler：调度中心。Monitor：监控中心，ZK：注册中心。**

**Admin：治理中心**

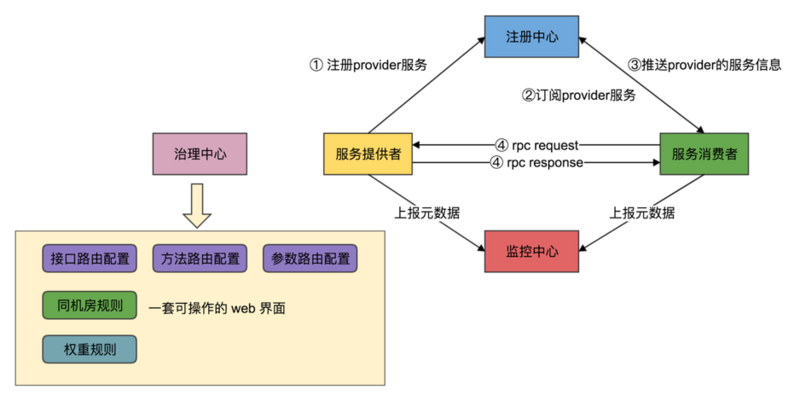
## 服务治理

上面提到了服务化，其实要想服务化，**服务治理**是关键。那么有没有好的服务治理方案呢？答案是有的，而且很多人都在用这个框架，他就是-dubbo。dubbo就是一个**带有服务治理功能的RPC框架**。

dubbo提供了一套较为完整的服务治理方案，所以企业如果要实现服务化的话，dubbo 是很好的一个选择。这里简单介绍一下dubbo服务治理相关方案。

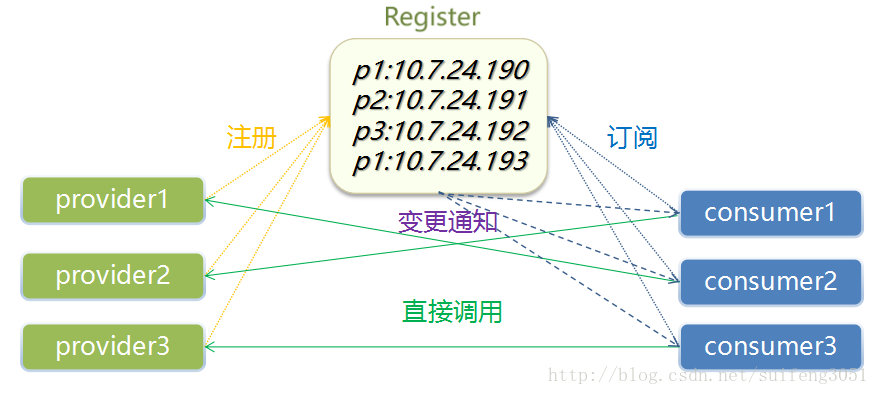
## 要想服务化，**服务治理**是关键.服务治理领域最重要的问题就是服务发现与注册。dubbo中引入了一个注册中心的概念，服务的注册与发现主要就依赖这个服务中心。

1. 我们服务**治理中心**，**需要一个注册中心**，统计都有哪些人提供了哪些服务，然后消费者，在启动服务的时候，像注册中心发送请求，我们需要哪些服务，注册中心推送提供者的服务信息。
2. 我们服务**治理中心**，**需要一个监控中心**，统计各个服务提供的次数、服务响应的时间、服务的健康状态。
3. 服务提供者和服务消费者之间通信，我们就别走 http 了，我们改成自定义协议，自己封装一套rpc 协议才是我们的良药，这样我们就像在使用本地方法一样调用远程的方法了,最好是服务提供者和服务消费者之间使用长连接，减少每次请求连接的时间消耗和网络I/O。这个rpc协议也由我们服务治理还给大家指定吧。



#### 服务发现注册

服务治理领域最重要的问题就是服务发现与注册。dubbo中引入了一个**注册中心**的概念，服务的注册与发现主要就依赖这个服务中心。



dubbo注册中心服务注册发现的具体过程：

**1.服务提供者启动，向注册中心注册自己提供的服务**

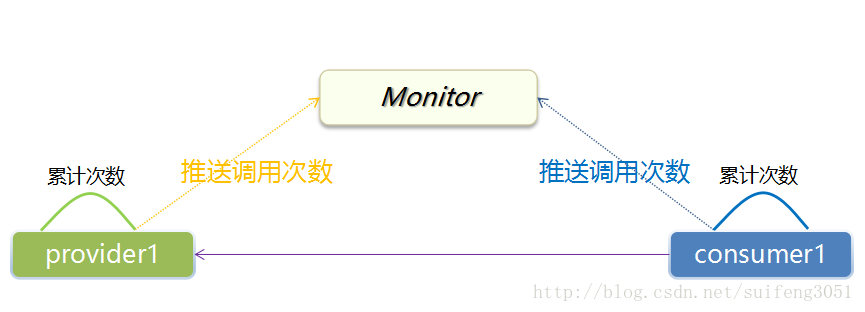
**2.消费者启动，向注册中心订阅自己需要的服务**

**3.注册中心返回服务提供者的列表给消费者**

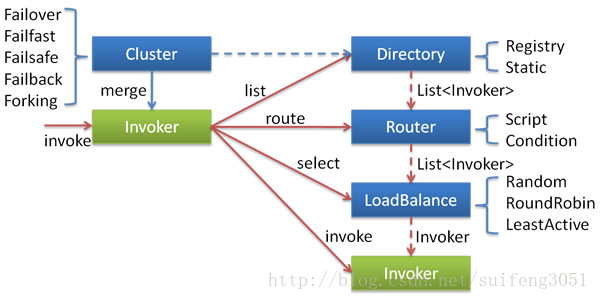
**4.消费者从服务提供者列表中，按照软负载均衡算法，选择一台发起请求**

**（N个一模一样的服务器，让客户端按照不同的机制进行分发，达到负载均衡的效果。我们约定两个子项目名称：**cloud-hyh-service-1**端口号：8071；**cloud-hyh-service-2**端口号：8072）**

#### 服务监控



#### 集群容错



#### 负载均衡

Random Loadbalance RoundRobin LeastActive ConsistentHash

#### dubbo服务治理优势

* 注册中心只负责注册查找，不负责请求转发，压力小
* 注册中心宕机不影响消费者，消费者本地缓存服务地址列表
* **注册中心对等集群，宕掉一台自动切换到另外一台**
* 服务提供者无状态，可动态部署，注册中心负责推送
* 统计无压力，本地内存中累计次数，每分钟发送注册中心
* 消费者调用服务者，自动软负载均衡
* 通过服务中心可追踪依赖关系
* 监控中心为扩容和降级提供依据
* 可启用acl机制进行鉴权
* 与Spring整合，接入简单松耦合
* 多种序列化协议支持

#### dubbo的不足

* 消费者仍需要依赖配置中心
* 消费者仍需要依赖jar包配置provider
* 提供者文档管理功能缺失
* 无统一入口
* 不支持OAuth2.0
* 内部鉴权不方便管理
* 无外部应用鉴权
* 接口基本裸奔，无法直接对外暴露服务
* IT治理不方便