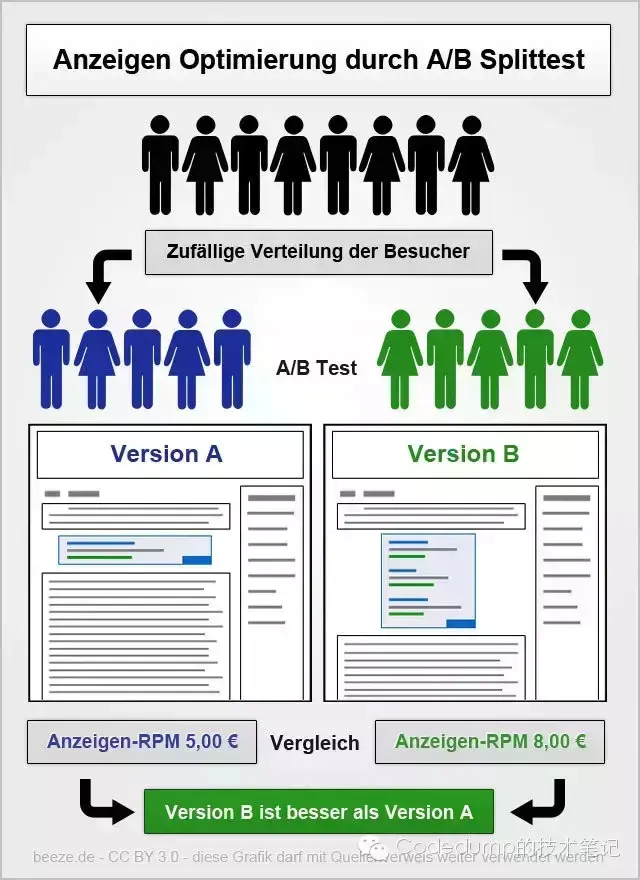
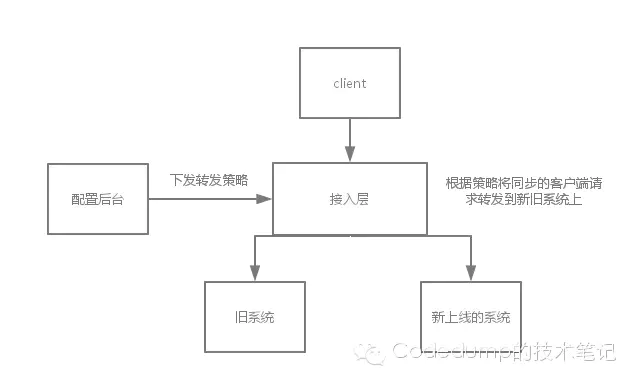
灰度发布系统的实现

灰度发布，已经不是一个很新的概念了．一个产品，如果需要快速迭代开发上线，又要保证质量，保证刚上线的系统，一旦出现问题那么可以很快的控制影响面，就需要设计一套灰度发布系统．

灰度发布系统的作用在于，可以根据自己的配置，来将用户的流量导到新上线的系统上，来快速验证新的功能修改，而一旦出问题，也可以马上的恢复，简单的说，就是一套A/BTest系统．

它大抵的架构，应该是类似这样的：



其中分为几个部分：

1. 接入层，接入客户端请求，根据下发的配置将符合条件的请求转发到新旧系统上．
2. 配置管理后台，这个后台可以配置不同的转发策略给接入层．
3. 新旧两种处理客户端请求的业务服务器．

关于接入策略的设计上，从协议层来说，需要从一开始就设计是根据哪些参数来进行转发的，而且这些参数最好跟具体的协议体内容分开，这样减少接入层对协议的解析．举个例子，如果客户端的请求是走HTTP协议的，那么将这些参数放在HEADER部分就好了，接入层不需要去具体解析body部分的数据就拿到了转发策略需要的参数．当然，放在HEADER中的数据，因为没有了加密性，又是需要考虑的另一个问题．

当然，最简单粗暴的转发策略，可以根据客户端ip地址来做，这是比较粗略的一个划分策略．

同样的，新旧服务器要对新旧客户端的协议兼容，也是能做到灰度发布的根本，如何设计一个扩展性好的应用协议，这一点就不在这里考虑了．

接下来，还需要满足如果管理后台下发了新的转发策略，接入层应该是可以马上感知到然后切换到这个新的策略来的．有好些不同的做法．假如接入层是Nginx这样的服务器，使用者只是在上面写了自己的Nginx模块来实现策略的转发，那么可能还需要在每台接入服务器上部署一个Agent的服务，主要用于：

1. 接收管理后台下发的策略，更新Nginx配置，然后优雅重启Nginx服务．
2. 定时检查本台机器的Nginx服务的状态，进行上报．

如果接入层不是Nginx这样的服务，那么也可以做一个pub-sub模型的订阅者，用ZK或者Redis都可以，订阅管理后台下发的服务进行处理即可．

<https://juejin.im/entry/6844903428053794824>

<https://www.jianshu.com/p/311009781b77>

<https://developer.aliyun.com/article/747091>

<https://juejin.im/post/6844903968137543694>

<https://www.zhihu.com/question/20584476>