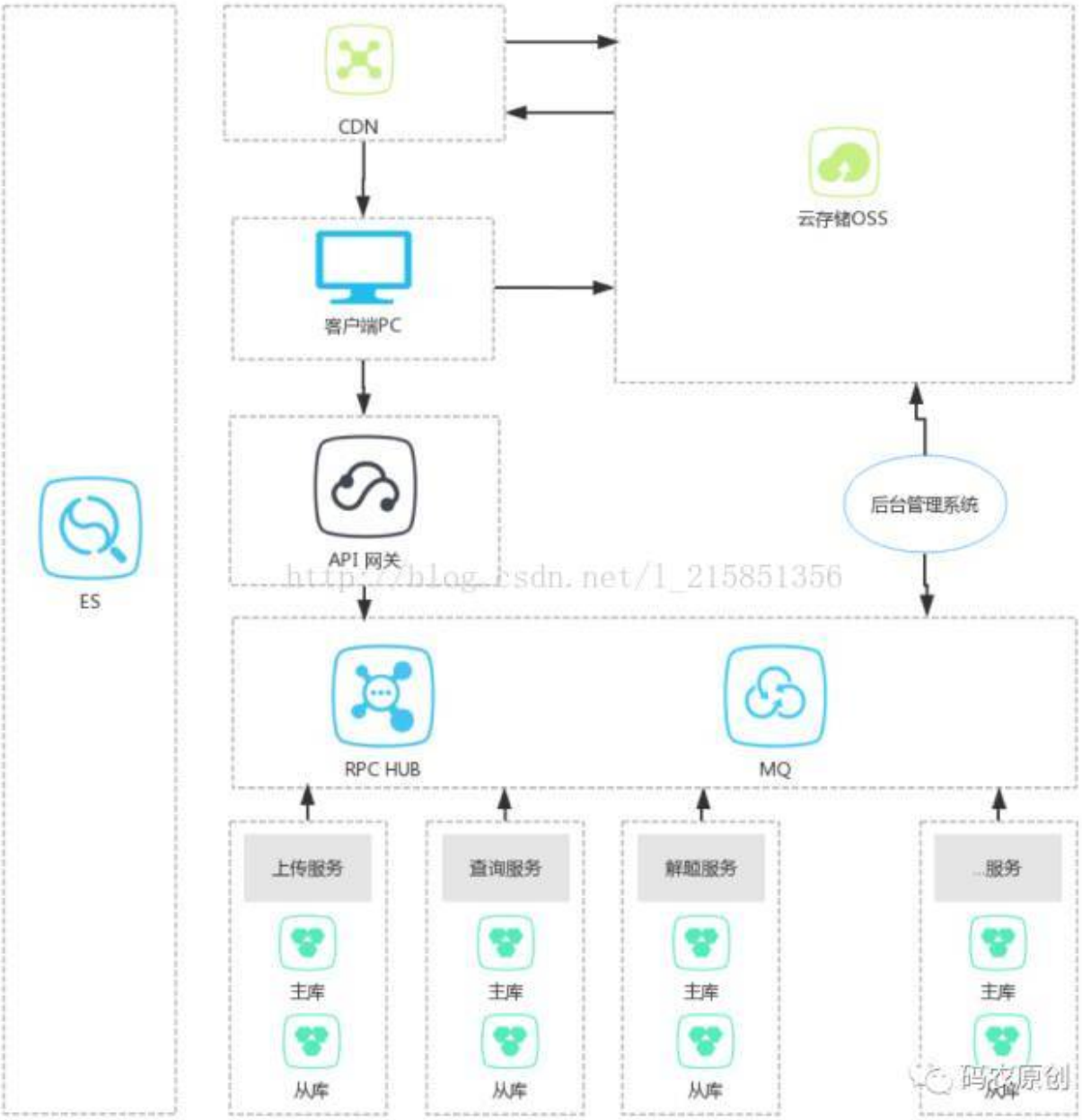
查询分离模式看似不错，解决了性能问题，我可以不用露宿街头了、老婆还是我的，哈哈。但是
软件系统天生的复杂性决定了，除了性能，还有其他诸如高可用、健壮性等大量问题等待我们解决，再加上各个部门间的撕逼、扯皮，更让我们码农雪上加霜，所以
继续吧.....

微服务模式可以说是最近的热点，花花绿绿、**小小、国内国外的公司都在鼓吹，实践这个模式，可是大部分都没有弄清楚为什么要这么做，也并不知道这么做有什么好处、坏处，在这里，我将以我自己的亲身实践说一下我对这个模式的看法，不喜勿喷！随着业务与人员的增加，遇到了如下的问题：

1. 单机数据库写请求量大量增加，导致数据库压力变大
2. 数据库一旦挂了，那么整个业务都挂了
3. 业务代码越来越多，都在一个GIT里，越来越难以维护
4. 代码腐化严重、臭味越来越浓
5. 上线越来越频繁，经常是一个小功能的修改，就要整个大项目要重新编译
6. 部门越来越多，该哪个部门改动大项目中的哪个东西，撕逼的厉害
7. 其他一些外围系统直接连接数据库，导致一旦数据库结构发生变化，所有的相关系统都要通知，甚至对修改不敏感的系统也要通知
8. 每个应用服务器需要开通所有的权限、网络、FTP、各种各样的，因为每个服务器部署的应用都是一样的
9. 作为架构师，我已经失去了对这个系统的把控.....

为了解决上述问题，我司使用了微服务模式，这种模式的一般设计见下图：



如上图所示，我把业务分块，做了垂直切分，切成一个个独立的系统，每个系统各自衍化，有自己的库、缓存、ES等辅助系统，系统之间的实时交互通过RPC，异步交互通过MQ，通过这种组合，共同完成整个系统功能。

那么，这么做是否真的解决上述问题了呢？不玩虚的，一个个来说。对于问题一，由于拆分成了多个子系统，系统的压力被分散了，而各个子系统都有自己的数据库实例，所以数据库的压力变小。

对于问题二，一个子系统A的数据库挂了，只是影响到系统A和使用系统A的那些功能，不会所有的功能不可用，从而解决一个数据库挂了，导致所有功能不可用的问题。

问题三、四，也因为拆分得到了解决，各个子系统有自己独立的GIT代码库，不会相互影响。通用的模块可通过库、服务、平台的形式解决。

问题五，子系统A发生改变，需要上线，那么我只需要编译A，然后上线就可以了，不需要其他系统做同样的事情。

问题六，顺应了康威定律，我部门该干什么事、输出什么，也通过服务的形式暴露出来，我部只管把我部的职责、软件功能做好就可以。

问题七，所有需要我部数据的需求，都通过接口的形式发布出去，客户通过接口获取数据，从而屏蔽了底层数据库结构，甚至数据来源，我部只需保证我部的接口契约没有发生变化即可，新的需求增加新的接口，不会影响老的接口。

问题八，不同的子系统需要不同的权限，这个问题也优雅的解决了。

问题九，暂时控制住了复杂性，我只需控制**的方面，定义好系统边界、接口、大的流程，然后再分而治之、逐个击破、合纵连横。

目前来说，所有问题得到解决！bingo!

但是，还有许多其他的副作用会随之产生，如RPC、MQ的超高稳定性、超高性能，网络延迟，数据一致性等问题，这里就不展开来讲了，太多了，一本书都讲不完。

另外，对于这个模式来说，最难把握的是度，切记不要切分过细，我见过一个功能一个子系统，上百个方法分成上百个子系统的，真的是太过度了。实践中，一个较为可行的方法是：能不分就不分，除非有非常必要的理由！。

优点：相对高性能，可扩展性强，高可用，适合于中等以上规模公司架构。
缺点：复杂、度不好把握。指不仅需要一个能在高层把控大方向、大流程、总体技术的人，还需要能够针对各个子系统有针对性的开发。把握不好度或者滥用的话，这个模式适得其反！