**架构演进及架构演进中的问题**

1、最初所有的东西（包括应用程序、数据可、静态资源都在一台服务器上），如下图



2、随着用户越来越多、访问量越来越大（cpu等吃紧），就到了应用服务与数据服务分离



3、随着访问并发越来越高、为了降低接口访问时间、提高服务性能

（很多业务数据不需要从数据库中提取，于是采用缓存，80%的业务访问集中在20%的数据上）（问题：具有那种特点的业务数据使用本地缓存、具有那种特点的业务数据使用远程缓存、分布式缓存在扩容时会碰到什么问题、分布式的算法有哪几种及优缺点）



4、随着访问的QPS不断提高、服务器的处理能力又有瓶颈（假设是tomcat）就需要服务器的集群，这时就要负载均衡调动服务器（问题：负载均衡的调度策略有哪些、优缺点、各适合哪些场景）（比如有轮询、权重、地址散列、最少连接、加权最少连接）



采用session sticky 解决session共享（这里的session server是单点的，如何解决单点保证可用性，可以把session server做成一个集群，问题：为什么做单点登录，有哪些解决方案、各有什么优缺点、如何平衡取舍）



5、当用户量达到一定量时、数据库成为瓶颈（采用数据库的读写分离）

增加了一个数据访问模块（data access module），使得外部人员并不知道增加了读和写的数据库。这里就引出了代码层次的演变，如何支持多数据源，如何封装对业务没有侵入，如何使用目前业务使用的ORM框架完成主从的读写分离，是否需要更换ORM，各有什么优缺点，如何取舍。（主库从库的延时，跨机房同步的问题）

6、为了提高服务器，增加了CDN和反向代理服务器，使用CND可以很好的解决不同地区访问速度问题，反向代理则在服务器机房中可以缓存用户的资源。



7、文件服务器出现瓶颈（采用分布式文件集群 问题：如何不影响已部署在线上的业务访问、是否需要业务部分帮忙清洗数据、是否需要备份服务器、是否需要重新做域名解析等）



8、数据库又出现瓶颈（采用专库专用的方式、进行数据的垂直拆分、解决数据并发量大的问题）如何解决跨库的事务（使用分布式事务或者不使用事务）



9、随着数据量的增大、访问量过大、业务量过大、某个业务的访问量、业务量达到单个数据库的瓶颈，这时就需要对数据库进行水平拆分。



水平拆分需要注意哪些点，有哪些水平拆分的方式。

拆分后的问题：SQL路由的问题（用户信息存在于USER1还是USER2），分页的问题。

10、当完成以上架构时，应用服务器的搜索量飙升，把应用服务器上的搜索功能单独抽取出来，做一个搜索引擎。



11、负载均衡集群



继续发展… SOA架构、服务化、消息队列、任务调度、多机房…

高大上的项目技术架构和开发设计实现不是一蹴而就的