一、【大型网站系统特点】
1、高迸发，大流量
2、高可用
3、少量数据
4、用户分布广，网络复杂
5、安全环境恶劣
6、需求变更快，版本发布频繁
7、渐进式发展

二、【大型网站架构所经过的路径（见高访问服务架构图）】
1、CDN服务器（缓存作用）
2、反向代理服务器（缓存作用）
3、负载均衡器（分流用户到具体的应用服务器上）
4、应用服务器（处理本应用的请求）
5、分布式微服务（如用户、商品、订单等），微服务器又可以有自己的本地缓存
若本级缓存未命中，可通过统一数据访问模块进行继续请求
6、搜索引擎服务器、nosql服务器、分布式缓存服务器、分布式文件服务器、分布式数据库等

三、【大型网站架构模式】
1、分层（横向切分）
应用层（文库、贴吧、知道、百科、首页、搜索等）；
服务层（用户服务、登陆服务、session服务、购物车服务等可复用的服务）
数据层（数据库服务、缓存服务、文件服务、搜索引擎服务等）
2、分割（纵向切分）
如应用层的分割（购物、论坛、搜索、广告）
服务层的分割（随应用层的分割而做相应的分割）
数据层的分割也类似
3、分布式（分层分割的目的是便于分布式部署，即将分层分割后的模块部署在独立的服务器上），常用的分布式方案
1）分布式应用和服务：指对分层分割后的应用层和服务层进行分布式部署
2）分布式静态资源：js、css、图片等
3）分布式数据和存储：SQL 和 NOSQL等
4）分布式计算：主要是离线的如：搜索引擎的索引构建、数据分析统计等，主要方案如Hadoop
5）分布式配置：如etcd
6）分布式锁
7）分布式文件系统等
4、集群
分布式已经将分层分割后的模块进行独立部署
集群是通过负载均衡连接多台服务器，并向外提供服务

5、缓存

1）CDN：部署在离用户最近的网络服务商，主要是静态资源

2）反向代理：部署在网站前端，主要是静态资源

3）本地缓存：应用服务器本地内存中的缓存

4）分布式缓存：

6、异步

主要是通过消息队列来达到不同模块之间的异步操作

优点：提高可用性，加快响应速度，消除迸发高峰

7、冗余

特别是数据库，即使访问量很小的时候也需要主从分离，实时热备份，异地数据中心灾备等

8、自动化

发布过程自动化、代码管理自动化、测试自动化、安全检测自动化、部署自动化、监控自动化、报警自动化等

9、安全

密码和校验码进行身份认证

通信进行加密

提交数据使用验证码识别

XSS攻击、SQL注入

敏感信息过滤

四、【大型网站架构评估指标】

1、**性能**：响应时间、并发数、吞吐量（TPS、QPS、HPS）、性能计数器（主要是服务器和操作系统上的一些数据指标）。

【1】前端优化

1）浏览器优化：

a）减少http请求：合并js，css，图片，最好一次http请求就完成

b）使用浏览器缓存：使用Cache-Control和Expires设置缓存和超时

c）启用GZip压缩：服务器压缩，浏览器解压，视情况而定是否采用

d）将CSS放在页面最上面，JS在最下面，因为浏览器会等所有CSS完后才会渲染，而JS是一加载就立即执行

e）减少Cookie传输，尤其请求一些不需要Cookie的静态资源时

2）CDN加速：主要缓存静态文件

3）反向代理：即可以做缓存，也可做负载均衡

【2】应用服务器优化

1）分布式缓存：memcached、redis等

a）将读写比大的数据缓存

b）注意防止缓存穿透，如恶意高并发的请求一个不存在的数据，由于缓存中没有保存该数据，就会穿透到数据库中，简单处理方式是将不存在的key也缓存起来并设置为null值

2）异步操作：使用消息队列

3）使用集群：用负载均衡设备构建一个由多台应用服务器组成的集群
4）代码优化：采用多线程、资源利用（如单例、对象池等）
5）算法与数据结构
6）垃圾回收

【3】存储优化
1）采用SSD硬盘
2）B+树和LSM树 ： SQL一般用B+树如MYSQL，而NOSQL会用LSM树
3）RAID技术 还是 HDFS技术

2、**可用性**：即服务器的可用时间与总时间之比，大型网站如QQ可达4个9以上
1）高可用的架构：主要是分层、分割、可分布式部署
2）高可用的应用（层）
a）用负载均衡将无状态的对等点服务器构建成集群
b）集群的Session管理，以实现应用服务器的无状态。
3）高可用的服务（层）：主要是服务器构建集群，同时有以下常用策略
a）分级管理：核心服务使用更好的硬件，甚至需要部署到不同的地域
b）超时设置：当应用访问服务时，设置一个超时，超时到后返回或重试等
c）异步调用：消息队列
d）服务降级：访问高峰期拒绝或随机拒绝部分服务；也可关闭不重要服务
e）幂等次性：即重复提交相同请求 跟 只提交一次的结果是一样的
4）高可用的数据（层）：CAP原理，即存储系统无法同时满足一致性、可用性和分区耐受性（指的是分布式存储），在大型网站应用中通常放弃了一致性，只要数据能最终一致性即可
a）数据备份：分热备份和冷备份，热备份又可分同步和异步的，SQL型数据库一般采用主从，NOSQL天生就有很好的备份机制
b）失效转移：
* 失效确认：通过心跳或应用程序访问失败报告
* 访问转移：访问一台服务器失败后，要切换到对等的另一台服务器上
* 数据恢复：若某台服务器挂了，需要将副本数目恢复到指定的数量
5）高可用软件质量
a）网站发布：通常由发布脚本来完成
b）自动化测试：如Selenium工具
c）预发布验证：即发布到‘预发面服务器’上，其与线上的服务器唯一的区别就是其没有配置到负载均衡器上
d）代码控制：GIT
e）自动化发布
f）灰度发布：即分批量逐步的发布，如发现在总题可以快速回滚
6）网站运行监控（“不允许没有监控的系统上线”）
a）监控数据采集
*日志收集（Storm工具）：访问日志；页面嵌入js采集用户操作行为
*服务器性能监控（Ganglia工具）：如load， 内存，IO等
*运行数据报告：缓存命中率、平均延迟、邮件发送数、待处理任务数等

b）监控管理

*系统报警

*失效转移

*自动降级

3、**伸缩性**：是否可构建集群，是否容易向集群中添加新的服务器，并且可以提供无差别的服务，是否有总服务器数量的限制等

【1】伸缩性架构

1）不同功能进行分层分割后实现伸缩

2）单一功能通过集群实现伸缩

【2】应用服务器集群设计：设计成无状态的，可用负载均衡设备增减服务器数量

1、负载均衡技术：

1）HTTP重定向：通过算法得到ip后返回302重定向。缺点需要两次请求

2）DNS域名解析：即一个域名绑定多个A记录IP，可构成一个集群，缺点是上线下线一台服务器修改A记录生效时间长，且DNS负载均衡控制权在域名服务商那里，一般是将DNS解析作为第一级负载均衡，其解析到的是下一级负载均衡，然后才到真实的服务器

3）反向代理（应用层代理）：一般反向代理同时也是负载均衡

4）IP负载均衡（网络层代理）：即在网络层修改请求目标地址进行负载均衡，当用户请求到负载均衡器后，操作系统内核获取数据包，根据算法得到一个真实服务器的IP，然后将数据包的目的地址改为真实服务器的IP，同时将数据包的源地址改为负载均衡服务器的地址。真实服务器处理完请求后，返回到负载均衡，再由负载均衡返回给用户。由于这种均衡是在内核里发生，其性能比反向代理更好；其不完美之处是，所有响应都要返回负载均衡器，对于下载服务和视频服务会受限于负载均衡器的带宽

5）链路层均衡（如LVS方案）：这种方式也叫直接路由（DR），通过配置集群内所有真实物理服务器的虚拟IP与负载均衡的IP一致，从而在数据包到负载均衡后，不需要修改目的IP，而只要修改目的MAC地址，由于真实物理服务器IP 与数据包请求的目的IP一致，在请求处理完后，可直接将响应返回给用户，不经过负载均衡器，从而不会受负载均衡器带宽的影响

2、常用均衡算法：轮询；加权轮询；随机；最少连接；源地址散列

【3】缓存服务器集群设计：由于缓存服务器不是对等的，不能简单的通过负载均衡器实现，因此需要采取合适的算法，使添加新的缓存服务器不会使大量请求穿透到数据库服务器，必须使新加入的服务器对整个集群影响最小，

1、选择合适的路由算法

1）分布式缓存的一致性HASH算法：

构造一个长度为0~2^32个整数环（称Hash环，常用二叉查找树实现），每一个物理缓存服务器对应一组环上的点（经验值150个）也就是说一台缓存服务器对应150个整数点（即找到这150个点中的任何一个点就能找到一个缓存服务器），当要缓存一个数据的时候，将数据的Key计算出一个整数Hash值，然后沿Hash环顺时针找第一个离这个Hash值最近的点对应物理缓存服务器进行缓存。当要加入一个新的缓存服务器的

时候，也是按150个点对应一个缓存服务器的原则加入Hash环中，这样就可以使新加入的缓存服务器对再有的缓存服务器影响最小，并且新的缓存服务器也能均匀的分担所有现存服务器的压力。

【4】存储服务器集群设计：

1、SQL：关系数据库本身支持数据复制、主从热备等机制，但很难做到大规模集群伸缩，一般要在数据库之外通过路由分库分表等手段实现，将部署有多个数据库的服务器组成分布式集群，开源解决方案：Amoeba 和 Cobar

Cobar原理：

Cobar根据SQL和分库规则分解SQL语句，分发到MYSQL集群不同的实例上执行，每个实例都部署为主从结构，参见Cobar模型图，前端通信模块负责和应用程序通信，并接收SQL请求（如select * from users where id in (12,22,23)）；然后转交给SQL解析模块，解析模块获得查询条件（id in (12,22,23)）；再转交给路由模块，路由模块根据配置发现ID为偶数要路由到数据库A，ID为奇数要路由到数据库B，于是将原SQL拆分为两个SQL(select *from users where id in (12,22); 和 select * from users where id in (23))并转交给SQL执行代理模块；执行代理模块将SQL分发到对应的数据库A\B执行并取得返回结果；然后将执行结果转交给结果合并模块，合并模块将所有结果合并成一个结果并返回给前端通信模块。

Cobar伸缩：

Cobar服务器是无状态的可通过负载均衡集群，而MYSQL需要做数据迁移，将集群中原来机器的数据迁移到新添加的机器中，具体迁移哪些数据可以利用一致性Hash算法实现

2、NOSQL：对于非关系型数据库，其设计目的就是为了分布式部署，一般都有很好的伸缩性，开源方案如：HBase

HBase原理：

HBase中的数据以HRegion为单位，每个HRegion存储一段[key1, key2)区间的数据，由HRegion完成读写操作；HRegionServer是物理服务器，其可以启动多个HRegion实例，当一个HRegion存储的数据达到一个阈值时，HRegionServer会将其分裂成两个HRegion，并将HRegion在整个集群中迁移，以达到负载均衡；HMaster记录所有HRegion信息（key区间，端口号等），HBase为高可用性会启动多个HMaster并通过Zookeeper选举出一个主HMaster。

应用程序访问HBase过程：

1）先通过Zookeeper得到主HMaster地址；

2）将key传给HMaster从而得到key对应的HRegionServer地址

3）请求HRgionServer得到key所在的HRegion，从而访问key对应的数据，这些数据被存储在若干个HFile中，这些文件所使用的是HDFS分布式文件系统存储

4、**扩展性**：当网站新加业务产品时，是否可对已有业务产生影响或影响很小

主要实现方案：通过分层分割将系统分成若干个低耦合的独立组件模块，模块间通过消息传递和依赖调用聚合成一个完整的系统

聚合方式：

1）利用分布式消息队列，在各个降低耦合性模块间传递消息

a）事件驱动架构：在低耦合性模块间通过传递事件消息，即消息生产与消费关系，生产者发布消息，一个或多个消费者接收消息进行消费

b）分布式消息队列：即将消息队列部署到独立的服务器上，消息生产者和消费者都通过接口访问分布工消息队列服务器

2）利用分布式服务，打造可复用的业务平台，即将业务与可复用的服务分离出来并独立部署，分布式服务通过接口分解系统耦合性，不同子系统或应用通过相同的接口进行服务调用 （见图分布式服务架构）

a）分布式服务器的需求和特点：

*服务可注册和发现；
*服务调用接口；
*可通过负载均衡伸缩；
*当某个服务器不能用了可进行失效转移；
*有高效的远程通信；
*由于每个服务用不同语言开发，需要具备整合异构的系统；
*对应用最少侵入；
*接口要具有版本管理；
*可实时监控；

b）分布式服务架构：如阿里巴巴的Dubbo（见图Dubbo架构原理）

3）可扩展的数据结构：这里主要是数据库表的设计是否可扩展

4）利用开放平台建设网站生态圈：主要是提供一些OATH等第三方接入

5、**安全性**：针对现存和潜在的各种攻击手段，是否有可靠的应对策略

【1】网站攻击与防御

1）XSS（跨站脚本攻击）

a）对用户输入的内容做HTML危险字符转义；

b）对敏感的Cookie信息设置HttpOnly属性

2）SQL注入：

a）对用户输入做严格类型检查

b）SQL预编译和参数绑定，这样用户输入的数据就被当作参数，而不是SQL语句去执行

3）CSRF（跨站请求伪造）：需要构造用户请求的所有参数才可以

a）表单token：由于CSRF需要构造用户请求的所有参数才可以，所以只要每次在表单中加一个随机数作为token并设置其超时，就可以有效的防御CSRF攻击

b）验证码：包括图形验证码、手机验证码等

c）Http Referer验证：如在图片防盗链中的应用等

4）其他攻击和漏洞

a）错误回显：有些黑客可以利用错误信息了解系统的结构，因此防御手段可以采取当发生错误的时候跳到一个预先定义好的错误页面

b）HTML注释：有时候页面的一些HTML注释会被黑客利用，所以当

发布的时候要将那些没必要的注释消除干净

c）文件上传：当上传一个恶意的可执行文件时，就使黑客完全控制服务器，所以上传文件时要确保只能传些可靠的文件类型，重新修改文件名或使用专门的存储设备等来防御

d）路径遍历：攻击者通过URL中使用相对路径来遍历系统中未开放的文件和目录

防御方法：将静态资源部署在独立的域名独立的服务器上，其他文件不使用静态URL访问（使其不能通过URL访问，也就不会被遍历了），URL中不要包含文件路径等相关信息；

5）Web应用防火墙

可统一拦截请求，过滤恶意参数，自动消毒和添加Token等可以自动处理大多数网站攻击，解决方案如ModSecurity

6）安全漏洞扫描：许多大型网站都会有安全团队开发自己的漏洞扫描工具

【2】信息加密与密钥安全

1）单向散列加密：如MD5、SHA等，为防彩虹破解，常需要加盐散列

2）对称加密：加密和解密使用相同的密钥，如DES、RC等

3）非对称加密：加密和解密使用不同的密钥，如RSA；

数字签名——就是发送方用自己的私钥对信息进行加密，然后传给接收方，接收方用签名者的公钥进行解密，由于私钥只有签名者才有，所以发送方是不可抵赖的，起到签名了就是代表其人的作用

4）密钥安全管理

a）将密钥和算法放在独立的服务器上，统一对外提供加密和解密服务。缺点是服务器会成为应用的瓶颈，开销大

b）算法放在应用系统中，密钥放在独立服务器中，应用程序通过调用统一的加解密接口进行加解密，该接口实现了常用的加解密算法，并可任意扩展，接口通过请求密钥服务器得到密钥后，并缓存在本地。

【3】信息过滤和反垃圾

1）文本匹配

a）正则表达式：适用于敏感词少的情况，要过滤文本比较少时

b）Trie树算法：双数组Trie树，base数组存储节点，check数组状态检查

c）构造多级Hash表进行文本匹配

2）分类算法：主要是贝叶斯分类算法，主要用于反垃圾、信息自动分类

3）黑名单：可通过Hash表实现；但对于亿级别的黑名单时，可用布隆过滤器代替Hash表，布隆过滤器原理：以要处理10亿个级别的邮箱黑名单为例，初始化一个16GB个二进制bit的空间为0，当要把一个邮箱地址加入黑名单时，通过8个随机映射函数将邮箱地址映射出8个0~16GB范围内的随机数，然后将这8个位置的bit设置为1。

【4】电子商务风险控制：大型电商网站风险控制主要是机器自动识别为高风险的交易或信息后，会发送给人工审核人员审核，自动风控技术主要有：

1）规则引擎

当交易满足一些条件时被认为是高风险的，如交易来自欺诈高发地区，

金额超过某个值，和上次登陆地距离差距大，用户登录与收货地不符，用户第一次交易等；一般会创建一个规则库，由程序自动按规则库去过滤交易，缺点是当规则库庞大起来后，会影响处理性能，一般大型网站会倾向于使用统计模型

2）统计模型

一般使用分类算法或更复杂的机器学习进行智能统计，根据历史交易信息训练分类算法，然后将采集加工后的交易信息输入分类算法，即可得到风险分值