1. I/O 优化

1. 增加缓存，减少磁盘的访问次数。
2. 优化磁盘的管理系统，设计最优的磁盘方式策略，以及磁盘的寻址策略，这是在底层操作系统层面考虑的。
3. 设计合理的磁盘存储数据块，以及访问这些数据库的策略，这是在应用层面考虑的。例如，我们可以给存放的数据设计索引，通过寻址索引来加快和减少磁盘的访问量，还可以采用异步和非阻塞的方式加快磁盘的访问速度。
4. 应用合理的 RAID 策略提升磁盘 I/O。

2. Web 前端调优

1. 减少网络交互的次数（多次请求合并）
2. 减少网络传输数据量的大小(压缩)
3. 尽量减少编码（尽量提前将字符转化为字节，或者减少从字符到字节的转化过程。）
4. 使用浏览器缓存
5. 减少 Cookie 传输
6. 合理布局页面
7. 使用页面压缩
8. 延迟加载页面
9. CSS 在最上面，JS 在最下面
10. CDN
11. 反向代理
12. 页面静态化
13. 异地部署

3.服务降级（自动优雅降级）

拒绝服务和关闭服务

4.幂等性设计

有些服务天然具有幂等性，比如讲用户性别设置为男性，不管设置多少次，结果都一样。但是对转账交易等操作，问题就会比较复杂，需要通过交易编号等信息进行服务调用有效性校验，只有有效的操作才能继续执行。

（注：幂等性是系统的接口对外一种承诺(而不是实现)，承诺只要调用接口成功，外部多次调用对系统的影响是一致的. 声明为幂等的接口会认为外部调用失败是常态，并且失败之后必然会有重试.）

5.失效转移

若数据服务器集群中任何一台服务器宕机，那么应用程序针对这台服务器的所有读写操作都需要重新路由到其他服务器，保证数据访问不会失败，这个过程叫失效转移。
失效转移包括：失效确认（心跳检测和应用程序访问失败报告）、访问转移、数据恢复。
失效转移保证当一个数据副本不可访问时，可以快速切换访问数据的其他副本，保证系统可用。

6.性能优化

根据网站分层架构,性能优化可分为：web 前端性能优化、应用服务器性能优化、存储服务器性能优化。

1. web 前端性能优化
    - 浏览器访问优化：减少 http 请求;使用浏览器缓存;启用压缩;css 放在页面最上面、javaScript 放在页面最下面;减少 Cookie 传输
    - CDN 加速
    - 反向代理
2. 应用服务器性能优化
    - 分布式缓存（Redis 等）
    - 异步操作（消息队列）
    - 使用集群（负载均衡）
    - 代码优化
3. 存储性能优化
    - 机械硬盘 vs 固态硬盘
    - B+树 vs LSM 树
    - RAID vs HDFS

7. 代码优化

- 多线程（Q:怎么确保线程安全？无锁机制有哪些？）
- 资源复用（单例模式，连接池，线程池）
- 数据结构
- 垃圾回收

8. 负载均衡

- HTTP 重定向负载均衡
  当用户发来请求的时候，Web 服务器通过修改 HTTP 响应头中的 Location 标记来返回一个新的 url，然后浏览器再继续请求这个新 url，实际上就是页面重定向。通过重定向，来达到“负载均衡”的目标。例如，我们在下载 PHP 源码包的时候，点击下

载链接时，为了解决不同国家和地域下载速度的问题，它会返回一个离我们近的下载地址。重定向的 HTTP 返回码是 302。
优点：比较简单。
缺点：浏览器需要两次请求服务器才能完成一次访问，性能较差。重定向服务自身的处理能力有可能成为瓶颈，整个集群的伸缩性国模有限；使用 HTTP302 响应码重定向，有可能使搜索引擎判断为 SEO 作弊，降低搜索排名。

- DNS 域名解析负载均衡
  DNS（Domain Name System）负责域名解析的服务，域名 url 实际上是服务器的别名，实际映射是一个 IP 地址，解析过程，就是 DNS 完成域名到 IP 的映射。而一个域名是可以配置成对应多个 IP 的。因此，DNS 也就可以作为负载均衡服务。
  事实上，大型网站总是部分使用 DNS 域名解析，利用域名解析作为第一级负载均衡手段，即域名解析得到的一组服务器并不是实际提供 Web 服务的物理服务器，而是同样提供负载均衡服务的内部服务器，这组内部负载均衡服务器再进行负载均衡，将请求分发到真是的 Web 服务器上。
  优点：将负载均衡的工作转交给 DNS，省掉了网站管理维护负载均衡服务器的麻烦，同时许多 DNS 还支持基于地理位置的域名解析，即会将域名解析成举例用户地理最近的一个服务器地址，这样可以加快用户访问速度，改善性能。
  缺点：不能自由定义规则，而且变更被映射的 IP 或者机器故障时很麻烦，还存在 DNS 生效延迟的问题。而且 DNS 负载均衡的控制权在域名服务商那里，网站无法对其做更多改善和更强大的管理。
- 反向代理负载均衡
  反向代理服务可以缓存资源以改善网站性能。实际上，在部署位置上，反向代理服务器处于 Web 服务器前面（这样才可能缓存 Web 相应，加速访问），这个位置也正好是负载均衡服务器的位置，所以大多数反向代理服务器同时提供负载均衡的功能，管理一组 Web 服务器，将请求根据负载均衡算法转发到不同的 Web 服务器上。Web 服务器处理完成的响应也需要通过反向代理服务器返回给用户。由于 web 服务器不直接对外提供访问，因此 Web 服务器不需要使用外部 ip 地址，而反向代理服务器则需要配置双网卡和内部外部两套 IP 地址。
  优点：和反向代理服务器功能集成在一起，部署简单。
  缺点：反向代理服务器是所有请求和响应的中转站，其性能可能会成为瓶颈。
- LVS-NAT:修改 IP 地址
- LVS-TUN: 一个 IP 报文封装在另一个 IP 报文的技术。
- LVS-DR:将数据帧的 MAC 地址改为选出服务器的 MAC 地址，再将修改后的数据帧在与服务器组的局域网上发送。

9.缓存

缓存就是将数据存放在距离计算最近的位置以加快处理速度。缓存是改善软件性能的第一手段，现在 CPU 越来越快的一个重要因素就是使用了更多的缓存，在复杂的软件设计中，缓存几乎无处不在。大型网站架构设计在很多方面都使用了缓存设计。

- CDN: 及内容分发网络，部署在距离终端用户最近的网络服务商，用户的网络请求总是先到达他的网络服务商哪里，在这里缓存网站的一些静态资源（较少变化的数

据），可以就近以最快速度返回给用户，如视频网站和门户网站会将用户访问量大的热点内容缓存在 CDN 中。

- 反向代理：反向代理属于网站前端架构的一部分，部署在网站的前端，当用户请求到达网站的数据中心时，最先访问到的就是反向代理服务器，这里缓存网站的静态资源，无需将请求继续转发给应用服务器就能返回给用户。
- 本地缓存：在应用服务器本地缓存着热点数据，应用程序可以在本机内存中直接访问数据，而无需访问数据库。
- 分布式缓存：大型网站的数据量非常庞大，即使只缓存一小部分，需要的内存空间也不是单机能承受的，所以除了本地缓存，还需要分布式缓存，将数据缓存在一个专门的分布式缓存集群中，应用程序通过网络通信访问缓存数据。

使用缓存有两个前提条件，一是数据访问热点不均衡，某些数据会被更频繁的访问，这些数据应该放在缓存中；二是数据在某个时间段内有效，不会很快过期，否则缓存的数据就会因已经失效而产生脏读，影响结果的正确性。网站应用中，缓存处理可以加快数据访问速度，还可以减轻后端应用和数据存储的负载压力，这一点对网站数据库架构至关重要，网站数据库几乎都是按照有缓存的前提进行负载能力设计的。

10. 负载均衡算法

轮询 Round Robin
加强轮询 Weight Round Robin
随机 Random
加强随机 Weight Random
最少连接 Least Connections
加强最少连接
源地址散列 Hash

其他算法

- 最快算法(Fastest)：传递连接给那些响应最快的服务器。当其中某个服务器发生第二到第 7 层的故障，BIG-IP 就把其从服务器队列中拿出，不参加下一次的用户请求的分配，直到其恢复正常。
- 观察算法(Observed)：连接数目和响应时间以这两项的最佳平衡为依据为新的请求选择服务器。当其中某个服务器发生第二到第 7 层的故障，BIG-IP 就把其从服务器队列中拿出，不参加下一次的用户请求的分配，直到其恢复正常。
- 预测算法(Predictive)：BIG-IP 利用收集到的服务器当前的性能指标，进行预测分析，选择一台服务器在下一个时间片内，其性能将达到最佳的服务器相应用户的请求。(被 BIG-IP 进行检测)
- 动态性能分配算法(Dynamic Ratio-APM):BIG-IP 收集到的应用程序和应用服务器的各项性能参数，动态调整流量分配。
- 动态服务器补充算法(Dynamic Server Act.):当主服务器群中因故障导致数量减少时，动态地将备份服务器补充至主服务器群。
- 服务质量算法(QoS):按不同的优先级对数据流进行分配。

- 服务类型算法(ToS): 按不同的服务类型(在 Type of Field 中标识)负载均衡对数据流进行分配。
- 规则模式算法：针对不同的数据流设置导向规则，用户可自行

11. 扩展性和伸缩性的区别

扩展性：指对现有系统影响最小的情况下，系统功能可持续扩展或替身的能力。表现在系统基础设施稳定不需要经常变更，应用之间较少依赖和耦合，对需求变更可以敏捷响应。它是系统架构设计层面的开闭原则（对扩展开放，对修改关闭），架构设计考虑未来功能扩展，当系统增加新功能时，不需要对现有系统的结构和代码进行修改。

衡量网站架构扩展性好坏的主要标准就是在网站增加新的业务产品时，是否可以实现对现有产品透明无影响，不需要任何改动或者很少改动既有业务功能就可以上线新产品。不同产品之间是否很少耦合，一个产品改动对其他产品无影响，其他产品和功能不需要受牵连进行改动。

伸缩性：所谓网站的伸缩性指是不需要改变网站的软硬件设计，仅仅通过改变部署的服务器数量就可以扩大或者缩小网站的服务处理能力。

指系统能够增加（减少）自身资源规模的方式增强（减少）自己计算处理事务的能力。如果这种增减是成比例的，就被称作线性伸缩性。在网站架构中，通常指利用集群的方式增加服务器数量、提高系统的整体事务吞吐能力。

衡量架构伸缩性的主要标准就是可以用多台服务器构建集群，是否容易向集群中添加新的服务器。加入新的服务器后是否可以提供和原来服务无差别的服务、集群中的可容纳的总的服务器数量是否有限制。

12.分布式缓存的一致性 hash

具体算法过程：先构造一个长度为 2^32 的整数环（这个环被称作一致性 Hash 环）根据节点名称的 Hash 值（其分布范围为[0,2^32 - 1]）将缓存服务器阶段设置在这个 Hash 环上。然后根据需要缓存的数据的 Key 值计算得到 Hash 值（其分布范围也同样为[0,2^32 - 1]），然后在 Hash 环上顺时针查找举例这个 KEY 的 hash 值最近的缓存服务器节点，完成 KEY 到服务器的 Hash 映射查找。

优化策略：将每台物理服务器虚拟为一组虚拟缓存服务器，将虚拟服务器的 Hash 值放置在 Hash 环上，key 在换上先找到虚拟服务器节点，再得到物理服务器的信息。

一台物理服务器设置多少个虚拟服务器节点合适呢？经验值：150。

13. 网络安全

1. XSS 攻击
   跨站点脚本攻击(Cross Site Script)，指黑客通过篡改网页，注入恶意的 HTML 脚本，

在用户浏览网页时，控制用户浏览器进行恶意操作的一种攻击方式。
防范手段：消毒（XSS 攻击者一般都是通过在请求中嵌入恶意脚本大道攻击的目的，这些脚本是一般用户输入中不使用的，如果进行过滤和消毒处理，即对某些 html 危险字符转移，如“>”转译为“& gt;”）;HttpOnly(防止 XSS 攻击者窃取 Cookie).

2. 注入攻击：SQL 注入和 OS 注入
   SQL 防范：预编译语句 PreparedStatement; ORM；避免密码明文存放；处理好相应的异常。
3. CSRF（Cross Site Request Forgery，跨站点请求伪造）。听起来与 XSS 有点相似，事实上两者区别很大，XSS 利用的是站内的信任用户，而 CSRF 则是通过伪装来自受信任用户的请求来利用受信任的网站。
   防范：httpOnly;增加 token;通过 Referer 识别。
4. 文件上传漏洞
5. DDos 攻击

14. 加密技术

1. 摘要加密：MD5, SHA
2. 对称加密：DES 算法，RC 算法， AES
3. 非对称加密：RSA
   非对称加密技术通常用在信息安全传输，数字签名等场合。
   HTTPS 传输中浏览器使用的数字证书实质上是经过权威机构认证的非对称加密的公钥。

15. 流控（流量控制）

1. 流量丢弃
2. 通过单机内存队列来进行有限的等待，直接丢弃用户请求的处理方式显得简单而粗暴，并且如果是 I/O 密集型应用（包括网络 I/O 和磁盘 I/O），瓶颈一般不再 CPU 和内存。因此，适当的等待，既能够替身用户体验，又能够提高资源利用率。
3. 通过分布式消息队列来将用户的请求异步化。