大型分布式网站分析

一、I/O优化

1、增加缓存，减少磁盘的访问次数。

2、优化磁盘管理系统，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。

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。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。

二、Web前端调优

1、减少网络交互的次数（多次请求合并）。

2、减少网路传输数据量的大小（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。（Cookie是数据包，可以让网页具有记忆功能，在某台电脑上记忆一定的信息。Cookie的工作原理是，第一次服务器端写入客户端的系统中。以后每次访问这个网页，都是先有客户端将Cookie发送到服务器端，再由服务器端进行判断，然后再产生HTML代码返还给客户端。）。

6、合理布局页面。

7、使用页面压缩。

8、演出加载页面。

9、CSS在最上面，JS在最下面。（CSS是一种样式表语言，用于为HTML文档定义布局例如，CSS涉及字体、颜色、边框、高度、宽度、背景图像、高级定位等方面。）（JS即Javascript，Javascript是一种由Netscape的LiveScript发展而来的脚本语言，主要目的是为了解决服务器终端语言，比如Perl，遗留的速度问题。当时服务端需要对数据进行验证，由于网络速度相当缓慢，只有28.8kbps，验证步骤浪费的时间太多。于是Netscape的浏览器Navigator加入了JavaScript，提供了数据验证的基本功能）。

10、CDN。

11、反向代理。

12、页面静态化。

13、异地部署。

三、服务降级（自动优雅降级）

拒绝服务和关闭服务。

1. 幂等性设计

有些服务天然具有幂等性，比如，用户性别设置为男，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。（注意：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）。

1. 失效转移

若数据服务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。

失效转移道正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。

1. 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

浏览器范文优化：减少http请求；使用浏览器缓存；启用压缩；css放在页面最上方、JavaScript放在页面最下面；减少Cookie传输；CDN加速；反向代理。

1. 应用服务器性能优化

分布式缓存（Redis等）、异步操作（消息列队）、使用集群（负载均衡）、代码优化。

1. 存储性能优化

机械硬盘vs固态硬盘

B+树 vs LSM树

RAID vs HDFS

1. 代码优化
2. 多线程
3. 资源复用（单例模式、连接池、线程池）
4. 数据结构
5. 垃圾回收
6. 负载均衡
7. HTTP重定向负载均衡

当用户发来请求时，web服务器通过修改HTTP响应头中的Location标记返回一个新的url，然后浏览器再继续请求这个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源码包时，点击下载链接时，为了解决不同国家和不同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。

2、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的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在进行负载均衡，将请求分发到真实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哪里，网站无法对其做更多的改善和更强大的管理。

1. 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响应，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。由于微博服务器不直接对外提供访问，因此微博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模式：修改IP。

LVS-TUN模式：一个IP报文封装在另一个IP报文的技术。

LVS-DR模式：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再与服务器组的局域网上发送。

1. 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原因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是用了缓存设计。

1. CDN：即内容分发网络，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最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。例如，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。
2. 反向代理：方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，无需将请求转发给应用服务器就能返回给用户。
3. 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。
4. 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：1、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；2、数据在某个时间内有效，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的速度，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均衡能力设计的。

1. 负载均衡算法
2. 轮询 Round Robin
3. 加强轮询 Weight Round Robin
4. 随机 Random
5. 加强随机 Weight Random
6. 最少连接 Least Connections
7. 加强最少连接  源地址散列 Hash
8. 其他算法
9. 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。
10. 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。
11. 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)
12. 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。
13. 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。
14. 服务质量算法(QoS):按不同的优先级对数据流进行分配。
15. 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。
16. 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
17. 扩展性和伸缩性的区别
18. 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。
19. 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。
20. 分布式缓存的一致性hash
21. 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。
22. 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。
23. 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。
24. 网络安全
25. XSS攻击
26. 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).
27. 注入攻击
28. 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。
29. CSRF（Cross Site Request Forgery，跨站点请求伪造）
30. 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。
31. 文件上传漏洞
32. DDos攻击
33. 加密技术
34. 摘要加密：MD5, SHA
35. 对称加密：DES算法，RC算法， AES
36. 非对称加密：RSA
37. 非对称加密技术通常用在信息安全传输，数字签名等场合。
38. 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。
39. 流控（流量控制）
40. 流量丢弃
41. 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。
42. 通过分布式消息队列来将用户的请求异步化。