高并发、大流量解决方案

一、高并发架构相关概念

1、并发：是指并发的访问，也就是某个时间点，有多少个访问同时到来；

通常如果一个系统的日 PV 在千万以上，有可能是一个高并发的系统；

2、具体关心什么?

QPS：每秒请求或查询的数量，在互联网领域，指每秒响应请求数；

吞吐量：单位时间内处理的请求量（通常由 QPS 与并发数决定）

响应时间：从请求发出到收到响应花费的时间，例如一个系统处理一个 HTTP 请求需要 100ms，这个 100ms 就是系统的响应时间；

PV：综合浏览量，即页面浏览量或者点击量，一个访客在 24 小时内访问的页面数量；

UV：独立访客 ，即一定时间范围内相同访客多次访问网站，只计算为一个独立的访客；

带宽：计算带宽大小需要关注两个指标，峰值流量和页面的平均大小；

日网站带宽=pv/统计时间（换算到秒）*平均页面大小（单位 kB）*8

峰值一般是平均值的倍数；

QPS 不等于并发连接数，QPS 是每秒 HTTP 请求数量，并发连接数是系统同时处理的请求数量；

峰值每秒请求数（QPS） = (总 PV 书*80%)/（6 小时秒数*20%）

压力测试：测试能承受的最大并发，测试最大承受的 QPS 值

测试工具（ab）：目标是 URL，可以创建多个访问线程对同一个 URL 进行访问（Nginx）；

ab 的使用：模拟并发请求 100 次（100 个人），总共请求 5000 次（每个人请求 5000 次）

ab -c 100 -n 5000 待测试网站（内存和网络不超过最高限度的 75%）

QPS 达到 50：一般的服务器就可以应付；

QPS 达到 100：假设关系型数据库的每次请求在 0.01 秒完成（理想），假设单页面只有一个 SQL 查询，那么 100QPS 意味着 1 秒中完成 100 次请求，但此时我们不能保证数据库查询能完成 100 次；

方案：数据库缓存层、数据库的负载均衡；

QPS 达到 800：假设我们使用 百兆宽带，意味着网站出口的实际带宽是 8M 左右，假设每个页面是有 10k，在这个并发的条件下，百兆带宽已经被吃完；

方案：CDN 加速、负载均衡

QPS 达到 1000：假设使用 memcache 缓存数据库查询数据，每个页面对 memcache 请求远大于直接对 DB 的请求；

memacache 的悲观并发数在 2W 左右，但有可能之前内网带宽已经被吃光，表现出不稳定：

方案：静态 HTML 缓存

QPS 达到 2000：文件系统访问锁都成为了灾难；

方案：做业务分离，分布式存储；

二、高并发解决方案案例：

流量优化：防盗链处理（把一些恶意的请求拒之门外）

前端优化：减少 HTTP 请求、添加异步请求、启用浏览器的缓存和文件压缩、CDN 加速、建立独立的图片服务器；

服务端优化：页面静态化处理、并发处理、队列处理；

数据库优化：数据库的缓存、分库分表、分区操作、读写分离、负载均衡

web 服务器优化：负载均衡；

网站大流量高并发访问的处理解决办法

—目录——

服务器并发处理

1、什么是服务器并发处理能力

一台服务器在单位时间里能处理的请求越多,服务器的能力越高,也就是服务器并发处理能力越强.HTTP 请求通常是对不同资源的请求,也就是请求不同的 URL,有的是请求图片,有的是获取动态内容,有的是静态页面,显然这些请求所花费的时间各不相同,而这些请求再不同时间的组成比例又是不确定的.

说个题外话

假如 100 个用户同时向服务器分别进行 10 个请求，与 1 个用户向服务器连续进行 1000 次请求，对服务器的压力是一样吗？(服务器缓冲区只有 1 个和 100 个请求等待处理)

2、提高服务器的并发处理能力

提高 CPU 并发的处理能力

1.多进程:多进程的好处不仅在于 CPU 时间的轮流使用,还在于对 CPU 计算和 I/O 操作进行很好的重叠利用.

2.减少进程切换:进程拥有独立的内存空间,每个进程都只能共享 CPU 寄存器。一个进程被挂起的本质是将它在 CPU 寄存器中的数据拿出来暂存在内存态堆栈着那个，而一个进程恢复工作的本质就是把它的数据重新装入 CPU 寄存器,这段装入和移出的数据称为“硬件上下文”,当硬件上下文频繁装入和移出时,所消耗的时间是非常明显的

3.减少使用不必要的锁:服务器处理大量并发请求时,多个请求处理任务时存在一些资源抢占竞争,这时一般采用“锁”机制来控制资源的占用,当一个任务占用资源时,我们锁住资源,这时其它任务都在等待锁的释放.

4.其他:不写了.对于老板来说,买就是了 o(*￣▽￣*)o

服务器集群、负载均衡、分布式

1、什么是集群?

单机处理到达瓶颈的时候,你就把单机复制几份,这样就构成了一个“集群”.集群中的每一台服务器叫做这个集群的节点,节点构成集群(废话…).每个节点提供<相同>的业务或者服务.这样系统的处理能力就会翻倍.

那么问题来了,用户的请求到底哪一台服务器去处理执行的？必须有”领导(负载均衡器)”.这个领导的职责就是进行调度所有的请求以达到使得每一台服务器的负载均衡(是不是很熟悉=.=),不能让有的人闲着,有的人忙死.

2、集群结构的优点?

集群结构的好处就是系统扩展非常容易.如果随着你们系统业务的发展,当前的系统又支撑不住了,那么给这个集群再增加节点就行了.所有节点处于活动状态,有一台 down(宕)机，那么整个的业务还在跑(分布式的话,emmmm….)

3、集群分类:

Linux 集群主要分成三大类:(高可用集群,负载均衡集群,科学计算集群),其他两个没了解,估计原理差不多吧…

4、集群负载的原理?

DNS 轮询、HTTP 重定向、IP 欺骗（又称三角传输） (这三种实现方式都是在用户通过域名来访问目标服务器时，由 GSLB 设备(Global Server Load Balancing)进行智能决策，将用户引导到一个最佳的服务 IP)

智能 DNS 可以通过多种负载均衡策略来将客户端需要访问的域名解析到不同的数据中心不同的线路上，比如通过 IP 地理信息数据库解析到最近的线路，或者权衡不同线路

的繁忙度解析到空闲的线路等等.

下面介绍一个解析线路的过程:

1、DNS 的负载均衡:

用户访问某个网站时，需要首先通过域名解析服务（DNS）获得网站的 IP。域名解析通常不是一次性完成的，常常需要查询若干不同的域名服务器才能找到对应的 IP。如下图所示，用户首先在本地配置一个本地 DNS 服务器地址，本地 DNS 服务器收到 DNS 请求后若不能解析，会将请求转发给更高一级的 DNS 服务器直到找到域名对应的 IP 或确定域名不存在

普通的访问流程:

加入 GBLS（全局负载均衡设备）访问流程:

优点:部署容易，成本低。缺点:GSLB 只能拿到本地 DNS 的 IP（获取的地理信息或者其他信息）,不能拿到用户的 IP，或者说用户的指定的本地 DNS 如果离自己较远,那么,GSLB 就回错误的认为你在本地 DNS 处,然后返回错误的 IP。

2、HTTP 重定向负载均衡:

使用基于 HTTP 重定向方案，首先在 DNS 中将 GSLB 设备的 IP 地址登记为域名的 A 记录（既域名对应的 IP）。如上图所示，用户首先通过 DNS 得到 GSLB 设备的 IP 地址，此时用户以为这就是站点服务器的 IP，并向其发送 HTTP 请求。GSLB 设备收到 HTTP 请求后使用一定策略选择一个最合适的服务器，然后 GSLB 设备向用户发送一个 HTTP 重定向指令（HTTP302），并附上选出的服务器的 IP 地址。最后，用户根据重定向 IP 访问站点的服务器。

3、基于 IP 欺骗的负载均衡:

5、和单机结构,集群结构的区别:

从单机到集群,你项目的架构,代码基本不用动(也会改动,因为服务器之间的响应也需要你的项目中有对应的模块或者功能),因为你是通过服务器的数量和负载均衡来增加并发和大流量问题的.

举个栗子:电商网站的某一款娃娃热卖((@^_^@)),负责娃娃的模块瞬间生热,多台服务器也满足不了大家只买这一款娃娃,然而网站的其他模块或者部分基本闲着,调度者(负载均衡器)也表示爱莫能助(因为每一台服务器的局部发热严重),已经非常平衡的分配了用户请求,但是请求的是都是娃娃,这一模块卡顿甚至 500(http 状态码都 5 开头了),也不能满足用户需求. 明显是需要把所有的节点服务器中空余的资源分出一部分来给娃娃模块.

6、分布式结构工作机理:

分布式结构就是将一个完整的系统,按照业务功能,拆分成一个个独立的子系统,在分布式结构中,每个子系统就被称为“服务”.这些子系统能够独立运行在 web 容器中,它们之间通过 RPC 方式通.

如果上述的网站采用分布式结构,你的服务器器们不在做同一件事情,各司其职,娃娃卖的好,那就多用几台服务器来负责娃娃相关的模块,利用率得到提高.

7、分布式结构的优点:

<1>系统之间的耦合度大大降低,可以独立开发、独立部署、独立测试,系统与系统之间的边界非常明确,排错也变得相当容易,开发效率大大提升

<2>系统之间的耦合度降低,从而系统更易于扩展.我们可以针对性地扩展某些服务.假设这个商城要搞一次大促,下单量可能会大大提升,因此我们可以针对性地提升订单系统、产品系统的节点数量,而对于后台管理系统、数据分析系统而言,节点数量维持原有水平即可
<3>服务的复用性更高.比如,当我们将用户系统作为单独的服务后,该公司所有的产品都可以使用该系统作为用户系统,无需重复开发

CDN

1、什么是 CDN?
内容分发网络（Content Delivery Network, CDN）其目的是通过在现有的 Internet 中增加一层新的网络架构，将网站的内容发布到最接近用户的网络"边缘"，使用户可以就近取得所需的内容，解决 Internet 网络拥塞状况，提高用户访问网站的响应速度。从技术上全面解决由于网络带宽小、用户访问量大、网点分布不均等原因所造成的用户访问网站响应速度慢的问题。
2、CDN 组成有什么?
一个 CDN 网络主要由以下几部分组成：内容缓存设备、内容分发管理设备、本地负载均衡交换机、GSLB 设备和 CDN 管理系统，其网络结构如下图所示
各部分的工作:
1、内容缓存设备 Cache 用于缓存内容实体和对缓存内容进行组织和管理。当有用户访问该客户内容时，直接由各缓存服务器响应用户的请求
2、内容分发管理设备主要负责核心 Web 服务器内容到 CDN 网络内缓存设备的内容推送、删除、校验以及内容的管理、同步。
3、GSLB 设备则实现 CDN 全网各缓存节点之间的资源负载均衡，它与各节点的 SLB 设备保持通信，搜集各节点缓存设备的健康状态、性能、负载等，自动将用户指引到位于其地理区域中的服务器或者引导用户离开拥挤的网络和服务器。还可以通过使用多站点的内容和服务来提高容错性和可用性，防止因本地网或区域网络中断、断电或自然灾害而导致的故障。
3、CDN 的工作流程
用户访问某个站点的内容时，若该站点使用了 CDN 网络，则在用户会在域名解析时获得 CDN 网络 GSLB 设备的 IP 地址。GSLB 设备根据其预设的选择策略（如，地理区域、用户时间等）为用户选择最合适的内容缓存节点，并且使用某种方式（如，基于 DNS、基于 HTTP 重定向、基于 IP 欺骗的方式等）导引用户访问所选的内容缓存节点。用户继续向缓存节点发出请求，若缓存中包含请求的内容，则直接返回给用户，否则从核心 Web 服务器中获取该内容，缓存后返回给用户。这样当用户再次访问相同内容或其他用户访问相同内容时，可以直接从缓存中读取，提高了效率

写在最后:

集群强调的是任务的同一性,分布式强调的是差异性.但是不是绝对对立的.上海的服务器处理上海的文件上传事务,北京服务器处理北京文件上传事务,最终结果都完成文件上传,从全局考虑是分布式的结构,但是就北京而言,北京那边有多台服务器在做同样的事情那么北京那边就是集群结构…分布式的每一个节点都可以做集群,但是集群不一定就是分布式的.

高并发,大流量处理及解决方法

第一:确认服务器硬件是否足够支持当前的流量。

普通的 P4 服务器一般最多能支持每天 10 万独立 IP，如果访问量比这个还要大，那么必须首先配置一台更高性能的专用服务器才能解决问题，否则怎么优化都不可能彻底解决性能问题。

第二：优化数据库访问

前台实现完全的静态化当然最好，可以完全不用访问数据库，不过对于频繁更新的网站，静态化往往不能满足某些功能。

缓存就是另一个解决方案，就是将动态数据存储到缓存文件中，动态网页直接调用这些文件，而不必再访问数据库，技术如果确实无法避免对数据库的访问，那么可以尝试优化数据库的查询 SQL.避免使用 Select * from 这样的语句，每次查询只返回自己需要的结果，避免短时间内的大量 SQL 查询。最好在相同字段进行比较操作,在建立好的索引字段上尽量减少函数操作，如果要做到极致的话需要代码的优化；

第三，禁止外部的盗链。

外部网站的或者文件盗链往往会带来大量的负载压力，因此应该严格限制外部对于自身的图片或者文件盗链，好在目前可以简单地通过 refer 来控制盗链，自己就可以通过配置来禁止盗链。当然，伪造 refer 也可以通过来实现盗链，不过目前蓄意伪造 refer 盗链的还不多，可以先不去考虑，或者使用非技术手段来解决，比如在图片上增加水印。

第四，控制大文件的下载。

大文件的下载会占用很大的流量，并且对于非 SCSI 硬盘来说，大量文件下载会消耗 CPU，使得网站响应能力下降。因此，尽量不要提供超过 2M 的大文件下载，如果需要提供，建议将大文件放在另外一台服务器上。

第五，使用不同主机分流主要流量

将文件放在不同的主机上，提供不同的镜像供用户下载。比如如果觉得 RSS 文件占用流量大，那么使用 FeedBurner 或者 FeedSky 等服务将 RSS 输出放在其他主机上，这样别人访问的流量压力就大多集中在 FeedBurner 的主机上，RSS 就不占用太多资源了。

第六，使用流量分析统计软件。

在网站上一个流量分析统计软件，可以即时知道哪些地方耗费了大量流量，哪些页面需要再进行优化，因此，解决流量问题还需要进行精确的统计分析才可以。推荐使用的流量分析统计软件是 Analytics（Google 分析）。

面对网站大流量、高并发该怎么处理

所谓大流量高并发指的是：在同时或极短时间内，有大量的请求到达服务端，每个请求都需要服务端耗费资源进行处理，并做出相应的反馈

如何看待高并发问题

从服务端视角看高并发: 服务端处理请求需要耗费服务端的资源，比如能同时开启的进程数、能同时运行的线程数、网络连接数、cpu、I/O、内存等等，由于服务端资源是有限的，那么服务端能同时处理的请求也是有限的。

高并发带来的问题： 高并发问题的本质就是：资源的有限性。因此，当高并发出现时，服务端的处理和响应会越来越慢，甚至会丢弃部分请求不予处理，更严重的会导致服务端崩溃。

如何解决高并发问题

CDN（内容分发网络）负载均衡，是构建在现有网络基础之上的智能虚拟网络，依靠部署在各地的边缘服务器，通过中心平台的负载均衡、内容分发、调度等功能模块，使用户就近获取所需内容，降低网络拥塞，提高用户访问响应速度和命中率，是解决高并发的不二之选