# **什么是高并发**

<https://blog.csdn.net/sanyaoxu_2/article/details/78992113>

**定义：**高并发(High Concurrency)是使用技术手段使系统可以并行处理很多请求。

**关键指标：**

-响应时间(Response Time)

-吞吐量(Throughput)

-每秒查询率QPS(Query Per Second)

-每秒事务处理量TPS(Transaction Per Second)

-同时在线用户数量

**关键指标的维度：**

-平均，如：小时平均、日平均、月平均

-Top百分数TP(Top Percentile)，如：TP50、TP90、TP99、TP4个9

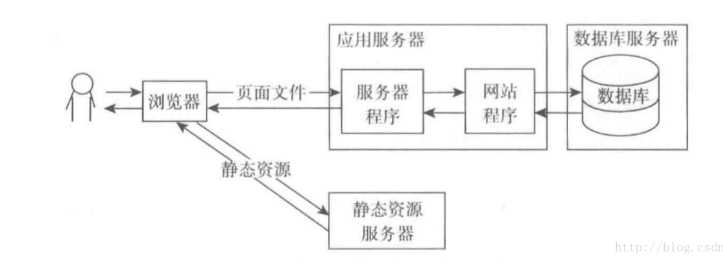
-最大值

-趋势

<https://www.cnblogs.com/xiexj/p/10399385.html>

# 高并发的解决策略

## 1.应用和静态资源分离



## 2.页面缓存

使用Nginx服务器就可以使用它自带的缓存功能，当然也可以使用专门的Squid 服务器。页面缓存的默认失效机制一班都是按缓存时间处理的，当然也可以在修改数据之后手动让相应的缓存失效。

经验

比如说一个显示文章的页面，正常来说完全可以静态化，但是如果文章后面有“顶”和“踩”的功能而且显示的有响应的数量，这个数据的变化频率就比较高了，这就会影响静态化。这个问题可以用先生成静态页面然后使用Ajax来读取并修改响应的数据，这样就可以一举两得来，既可以使用页面缓存也可以实时显示一些变化频率高的数据来。

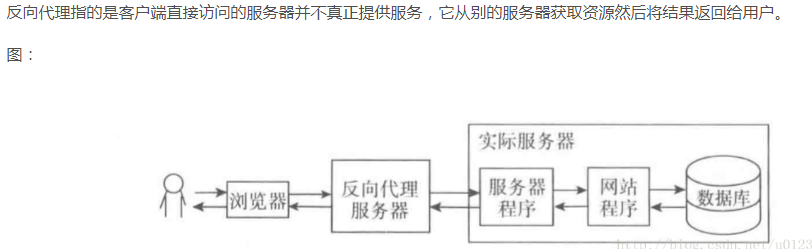
但是对于大量内容并且频繁更新的网站，我们无法全部手动去挨个实现，于是出现了我们常见的信息发布系统CMS（content management system），像我们常访问的各个门户站点的新闻频道，甚至他们的其他频道，都是通过信息发布系统来管理和实现的，信息发布系统可以实现最简单的信息录入自动生成静态页面，还能具备频道管理、权限管理、自动抓取等功能，对于一个大型网站来说，拥有一套高效、可管理的CMS是必不可少的。

## 3.集群与分布式

集群有两个方式：一种是在静态资源集群。另一种是应用程序集群。静态资源集群比较简单。应用程序集群在处理过程中最核心的问题就是Session 同步问题。

Session 同步有两种处理方式：一种是在Session 发生变化后自动同步到其他服务器，另一种就是用个程序统一管理Session。所有集群的服务器都使用同一个Session，Tomcat 默认使用就是第一种方式，通过简单的配置就可以实现，第二种方式可以使用专门的服务器安装Mencached等高效的缓存程序统一来管理session，然后再应用程序中通过重写Request并覆盖getSession 方法来获取制定服务器中的Session。

## 4. 反向代理



代理服务器的作用是代我门获取想要的资源然后将结果返回给我们，所要获取的资源是我门主动告诉代理服务器的，比如，我门想访问Facebook，但是直接访问不了，这时就可以让代理服务器访问，然后将结果返回给我们。

反向代理服务器是我门正常访问一台服务器的时候，服务器自己去调用了别的服务器资源并将结果返回给我们，我门自己并不知道。

代理服务器是我们主动使用的，是为我们服务的，他不需要有自己的域名；反向代理服务器是服务器自己试用的，我门并不知道，它有自己的域名，我门访问它和访问正常的网址没有任何区别。

反向代理服务器主要有三个作用：

1. 可以作为前端服务器跟实际处理请求的服务器集成；

2. 可以做负载均衡

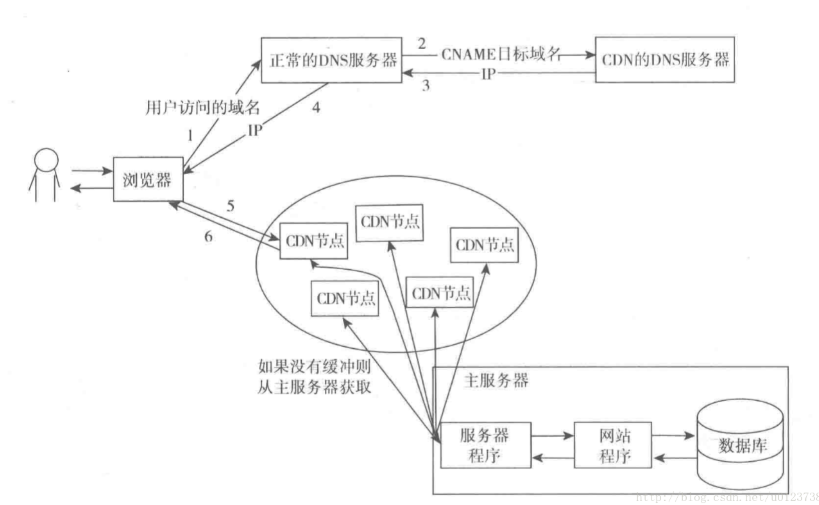
3. 转发请求，比如说可以将不同类型的资源请求转发到不同的服务器去处理。

## 5. CDN

cdn其实是一种特殊的集群页面缓存服务器，他和普通集群的多台页面缓存服务器相比，主要是它存放的位置和分配请求的方式有点特殊。CDN 服务器是分布在全国各地的，当接收到用户请求后会将请求分配到最合适的CDN服务器节点获取数据。比如联通的用户分配到联通的节点，上海的用户分配到上海的节点。

CDN的每个节点其实就是一个页面缓存服务器，如果没有请求资源的缓存就会从主服务器获取，否则直接返回缓存的页面。

CDN分配请求（负载均衡）的方式是用专门的CDN域名解析服务器在解析域名的时候就分配好的。一般的做法是在ISP哪里试用CNAME将域名解析到一个特定的域名，然后再将解析到的那个域名用专门的CDN服务器解析道相应的CDN节点。如图。



第二步访问CDN的DNS服务器是应为CNAME记录的目标域名使用NS记录指向了CDN的DNS服务器。CDN的每个节点可能也是集群了多台服务器。

## 6. 底层的优化

## 7.数据库集群和库表散列

# 总结

## 分析问题的思路

海量数据和高并发经常被连在一块说事儿，虽然他们完全是两回事儿。海量数据纯指的是数据库的海量数据，而并发指的却包括数据库和服务器的高访问量。

那么问题来了，既然是数据库的数据量大，那怎么办呢？要想解决问题，首先要知道问题是什么！！！那么海量数据会给我带来什么样的问题呢？

海量数据带来的问题无非就是增删改查的问题，除了之外还能有啥问题呢？总不能是带来安全问题吧（打脸一，还真有可能是安全问题）

1 数据库访问缓慢

2 插入更新缓慢，这个问题只能通过分库分表解决

要解决数据库访问缓慢的问题还有几种方法，既然访问数据库慢的话，在逻辑允许的情况下可以不访问数据库呢？

1 使用缓存

2 使用页面静态化

既然不访问数据库逃不过去了，那我们就对数据库进行优化

3 优化数据库（包含的内容非常多，比如参数配置，索引优化，sql优化等等）

4 分离数据库中活跃的数据

5 读写分离

6 批量读取和延迟修改；

7 使用搜索引擎搜索数据库中的数据

8 使用NoSQL和Hadoop等技术；

9 进行业务的拆分；

## 高并发的解决方案

其实这个问题必须结合上面的海量数据来讨论，什么情况下会出现高并发呢？一定是平时访问量就比较大的情况，那么平时访问量比较大相应的数据存储也就越来越多，这都是相辅相成的，当然也有个例，比如刚需，比如12306，这里的高并发相比于它的数据来说已经不算海量了。那么平时访问量大如何解决呢？因为这里牵扯到服务器和数据库的问题，所以要从这两方面来进行优化

1 增加web服务器数量，也就是做集群，做负载均衡。既然一台服务器无法完成任务，那就多用几台，几台不够用机房

 在通向第二种解决方法之前，还有没有除了数据库服务器之外能做的一些优化手段呢？当然有

1.1 页面缓存

1.2 cdn

1.3 反向代理

1.4 应用程序和静态资源分离（比如专供下载的资源单独放在一起，给这台服务器提供很高的带宽资源）

2 增加数据库服务器数量，同样做集群，做负载均衡。

## 海量数据的解决方案

1 使用缓存

好多事情都是相辅相成的，相比来说使用缓存更多是用来解决高并发问题的，因为海量数据导致了访问的缓慢，容易造成高并发问题的严重性，又因为数据库一般是web访问的瓶颈，所以我们在业务逻辑允许的情况下尽量先避免操作数据库，于是，就有了缓存。将必要的数据存放在内存中，而不必每次都去数据库中读取造成不必要的性能浪费和加快访问速度---这就是缓存带来的好处。那使用缓存以及选用管理缓存软件时应该注意些什么东西呢？

2 页面静态化---不想解释，还有什么值得去解释呢？

3 数据库优化

3.1 数据库表结构涉及

3.2 数据类型的选用

3.3 sql优化

3.4 索引优化

3.5 配置优化

4 分离数据库中的活跃数据

为什么要分离呢？说一个我实际环境中遇到的问题吧！有一个表只有10几个字段，表有130万条数据，但大小已经到了5G的数据，这本身是不太合理的，这么少的数据占用了太多的数据，说明其中有些字段存储了大量的字符串（比如说文章内容等），每次检索这个表时大部分是用不到这些大字段内容的，但却需要耗时比较长，产生很多的慢日志。这时我们可以考虑将表进行垂直切分，将活跃数据分离开来，这样能大大加快访问速度

5 读写分离