高并发和大流量解决方案

高并发的问题，应关注

1. QPS：每秒钟请求或查询数量，在互联网领域指每秒响应的请求数（指 HTTP 请求）
2. 吞吐量：单位时间内处理的请求数量（通常由 QPS 和并发数决定）
3. 响应时间：从请求发出到收到响应花费时间
4. PV：综合浏览量（Page View），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。同一个人浏览你的网站同一个页面，只记作一次 PV
5. UV：独立访客（UniQue Visitor）,即一定时间范围内相同访客多次访问网站，只能计算为 1 个独立访客
6. 带宽：计算带宽大小需关注两个指标，峰值流量和页面的平均大小
7. 日网站带宽=PV/统计时间（秒）*平均页面大小（KB）*8
8. 峰值一般是平均值的倍数
9. QPS 不等于并发并发连接数。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
10. 二八定律（80%的访问量集中在 20%的时间）:（总 PV 数 *80%）/（6 小时秒速* 20%）=峰值每秒请求数（QPS）
11. 压力测试：能承受最大的并发数和最大承受的 QPS 值

常用性能测试工具

ab，wrk，Apache JMeter, http_load, Web Bench, Siege

ab

使用方法：

```
# 模拟并发请求 100 次，总请求 5000 次

ab -c 100 -n 5000 http://example.com
```

注意事项：

1. 测试机器与被测机器分开
2. 不要对线上服务做压力测试
3. 观察测试工具所在机器，以及被测试的前端机的 CPU、内存、网络等都不超过最高限度的 75%

QPS 指标

1. QPS 达到 50，可以称之为小型网站，一般服务器都可以应付
2. QPS 达到 100；瓶颈：MySQL 查询达到瓶颈；优化方案：数据库缓存层，数据库负载均衡
3. QPS 达到 800；瓶颈：带宽速度达到瓶颈；优化方案：CDN 加速，负载均衡
4. QPS 达到 1000；瓶颈：缓存服务器的带宽达到瓶颈；优化方案：静态 HTML 缓存
5. QPS 达到 2000；瓶颈：文件系统访问锁成为灾难；优化方案：做业务分离，分布式存储

高并发优化方案

流量优化

1. 防盗链处理

前端优化

1. 减少 HTTP 请求
2. 添加异步请求
3. 启用浏览器缓存和文件压缩
4. CDN 加速
5. 建立独立的图片服务器

服务端优化

1. 页面静态化
2. 并发处理

数据库优化

1. 数据库缓存
2. 分库分表、分区操作
3. 读写分离
4. 负载均衡

web 服务器优化

1. 负载均衡

web 资源防盗链

盗链定义

1. 倒链是指在自己的页面上展示一些并不在服务器上的内容
2. 获得他人服务器上的资源地址，绕过别人的资源展示页面，直接在自己的页面上向最终用户提供此内容
3. 常见的是小站盗用大站的图片、音乐、视频、软件等资源
4. 倒链可以减轻自己的服务器负担

防盗链定义

防止别人通过一些技术手段绕过本站的资源展示页面，盗用本站的资源，让绕开本站资源展示页面的资源链接失效，可以大大减轻服务器及带宽的压力

防盗链的工作原理

1. 通过 Referer 或者计算签名，网站可以检测目标网页访问的来源网页，如果是资源文件，则可以跟踪到显示他的网页地址
2. 一旦检测到来源不是本站即进行阻止或返回指定的页面

防盗链实现方法

Referer

1. NGINX 模块 ngx_http_referer_module 用来阻挡来源非法的域名请求
2. NGINX 指令 valid_referers，全局变量$invalid_referer

配置：

```
valid_referers  none|blocked|server_names|string...;
```

1. none: Referer 来源头部为空的情况，比如直接打开
2. blocked: Referer 来源头部不为空，但是里面的值被代理或者防火墙删除了，这些值都不以 http://或者 https://开头
3. server_names: Referer 来源头部包含当前的 server_names

配置例子：

```
location  ~.*\.(gif|jpg|png|flv|swf|rar}zip)$
{
    valid_referers  none  blocked  imooc.com  *.imooc.com;
    if  ($invalid_referer)
    {
        #return  403;
        rewrite  ^/  http://www.imooc.com/403.jpg;
    }
}
```

减少 HTTP 请求

HTTP 连接产生的开销

1. 域名解析
2. TCP 连接
3. 发送请求
4. 等待
5. 下载资源
6. 解析

解决方案

1. 减少组件的数量，并由此减少 HTTP 请求的数量
2. 图片地图：图片地图允许你在一个图片上关联多个 URL。目标 URL 的选择取决于用户蛋鸡了图片上的哪个位置
3. CSS Sprites:css 精灵，通过使用合并图片，通过指定 css 的 background-image 和 background-position 来显示元素
4. 合并脚本和样式表适
5. 图片使用 base64 编码减少页面请求数