缓存策略的选择

适合缓存的内容

1. 不变的图像，如 logo，图标等
2. js、css 静态文件
3. 可下载的内容，媒体文件

适合协商缓存

1. HTML 文件
2. 经常替换的图片
3. 经常修改的 js、css 文件，js、css 文件的加载可以加入文件的签名来拒绝缓存，如‘index.css?签名’，‘index.签名.js’

不建议缓存的内容

1. 用户隐私等敏感数据
2. 经常改变的 API 数据接口

NGINX 配置缓存策略

本地缓存配置

1. add_header 指令：添加状态码为 2XX 和 3XX 的响应头信息，设置代码 `add_header name value [always];`，可以设置 Pragma、Expires、Cache-Control，可以继承
2. expires 指令：通知浏览器过期时长，设置代码 `expires time;`
3. Etag 指令：指定签名，设置代码 `etag on|off`，默认 on

前端代码和资源压缩

优势

1. 让资源文件更小，加快文件在网络中的传输，让网页更快的展现，降低带宽和流量的开销

压缩方式

1. js、css、图片、html 代码的压缩
2. gzip 压缩

gzip 配置

```
gzip on|off; #是否开启 gzipgzip_buffers 32 4K|16 8K; #缓冲（在内存中缓存几块？每块多大）gzip_comp_level [1-9] #推荐 6，压缩级别（级别越高，压得越小，越浪费 CPU 计算资源）
gzip_disable #正则匹配 UA，什么样的 Uri 不进行 gzip
gzip_min_length 200 #开始压缩的最小长度
gzip_http_version 1.0|1.1 #开始压缩的 http 协议版本
gzip_proxied #设置请求者代理服务器，该如何缓存内容
gzip_types text/plain application/xml image/png #对哪些类型的文件压缩，如 txt、xml、css
gzip_vary on|off #是否传输 gzip 压缩标志
```

CDN 加速

定义

1. CDN 的全称 content delivery network，内容分发网络
2. 尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输的更快、更稳定
3. 在网络各处放置节点服务器所构成的有的互联网基础之上的一层智能虚拟网络
4. CDN 系统能够实现地根据网络流量和各节点的连接、负载状况以及到用户距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上

优势

1. 本地 cache 加速，提高了企业站点（尤其含有大量图片和静态页面站点）的访问速度
2. 跨运营商的网络加速，保证不同网络的用户都能得到良好的访问质量

3. 远程访问用户根据 DNS 负载均衡技术只能选择 cache 服务器
4. 自动生成服务器的远程 Mirror（镜像）cache 服务器，远程用户访问时从 cache 服务器上读取数据，减少远程访问的带宽、分担网络流量、减轻原站点 web 服务器负载等功能
5. 广泛分布的 cdn 节点加上节点之间的智能冗余机制，可以有效地预防黑客入侵

工作原理

1. 用户发起请求
2. 智能 DNS 的解析（根据 IP 判断地理位置、接入网类型、选择路由最短和负载最轻的服务器）
3. 取得缓存服务器 ip
4. 把内容返回给用户（如果缓存中有，没有就执行 5、6、7）
5. 向源站发起请求
6. 将结果返回给用户
7. 将结果存入缓存服务器

适用场景

1. 站点或者应用中大量静态资源的加速分发，例如 css、js、图片和 HTML
2. 大文件下载
3. 直播网站

独立图片服务器

必要性

1. 分担 web 服务器的 I/O 负载，将耗费资源的图片服务器分离出来，提高服务器的性能和稳定性
2. 能够专门对图片服务器进行优化，为图片服务器设置针对性的缓存方案，减少带宽成本，提高访问速度
3. 提高网站的可扩展性，通过增加图片服务器，提高图片吞吐能力

采用独立域名

原因：

1. 同一域名下浏览器的并发连接数有限制，突破浏览器连接数的限制
2. 由于 cookie 的原因，对缓存不利，大部分 web cache 都只缓存不带 cookie 的请求，导致每次的图片请求都不能命中 cache

如何图片上传和同步

1. NFS 共享方式
2. 利用 FTP 同步

动态语言静态化

将现有的 PHP 等动态语言的逻辑代码生成为静态的 HTML 文件，用户访问动态脚本重定向到静态 HTML 文件的过程。对实时性要求不高

原因：

1. 动态脚本通过会做逻辑计算和数据查询，访问量越大，服务器压力越大
2. 访问量大时可能会造成 CPU 负载过高，数据库服务器压力过大
3. 静态化可以减低逻辑处理压力，降低数据库服务器查询压力

实现方法

1. 使用模板引擎
2. 利用 ob 系列函数

```
ob_start();//打开输出控制缓冲
ob_get_content();//返回输出缓冲区内容
ob_clean();//清空输出缓冲区
ob_end_flush();//冲刷出（送出）输出缓冲区内容并关闭缓冲
```

微信二维码扫一扫咨询大神进阶课程内容

微信号码 若兰：zqf19907493857

QQ 咨询联系 妮妮：194210485