
CDN及缓存小讲

一个访问如何完成？

- 浏览器发起请求最基本的几项：
  Host、URL、Method
- 扩展项：
  Accept、Referer、Cookie、User-Agent等等

Host处理

- 如果是IP，直接走路由找IP
- 如果是域名：
  1、查找本机hosts和DNScache
  Win：C:\WINDOWS\system32\drivers\etc、hosts
  Linux:/etc/hosts
  2、向localDNS发送resolve请求

localDNS处理

- TTL(Time To Live)
  DNS都有一个TTL配置，以我们现在使用bind9为例：

- 如果LocalDNS上正好forwarder过这个Host，且TTL没有过期，那么直接返回结果。

localDNS处理

- 否则LDNS将通过TCP53向它所知的其他DNS发送查询
  递归查询、迭代查询
- 最终归到根服务器“.”
- 真正的域名最后都有一个“.”，如下：

```
zone "." in {
                type hint;
                file "root.cache";
        };

zone "china.com" in {
        type master;
        file "named.china.com";
        allow-query { any;};
};
```

```
cdcgames        IN      CNAME           www.cdcgames.net.
```

解析结果

- 如果DNS应答的结果是A记录，那么已经获得了Host的IP，可以发起请求了；
- 如果是C记录，那么得到的还是一个域名，继续重复上面的过程，直到获得IP为止。

```
flash.shanzhai   IN      CNAME           shanzhai
flash.shanzhai   IN      A               124.238.250.51
```

服务器响应

- 如果没有缓存，那么请求最后的“旅程”可能就是这样

CDN原理

- GSLB(Gobal Server Load Balance)
- 最基本的GSLB控制器，就是一个智能DNS
- 以bind9为例，通过acl和view完成该功能

ACL

- acl "CNCIP"{
- 221.14.122.0/21;
- 221.13.128.0/14;
- 221.192.168.0/22;
- 218.21.128.0/17;
- 210.15.51.0/24;
- ……
- }
- (GSLB会分得很细，我们自己双线，所以只区分CT和CNC)

VIEW

```
view "CNC" {
        key "cnc" {
             algorithm hmac-md5;
             secret "mSzVmAeYKMnUplLTbyAQZw==";
        };
        match-clients {
              key "cnc";
              202.84.1.111/32;
              !221.194.139.13/32;
              CNCIP;
        };
zone "china.com" in {
        type master;
        file "named.china.com-bj";
        allow-query { any;};
//        allow-transfer { our-nets; };
};
```

named.china.com-bj

- ……
- cnccdn IN A 221.194.139.22
- IN A 221.194.139.12
- IN A 221.194.139.78
- ……
- 稍微完善的GSLB，还会对这些服务器的负载、链路状况进行监控，随时对A记录进行微调~(Scalable Service Routing)

缓存服务器

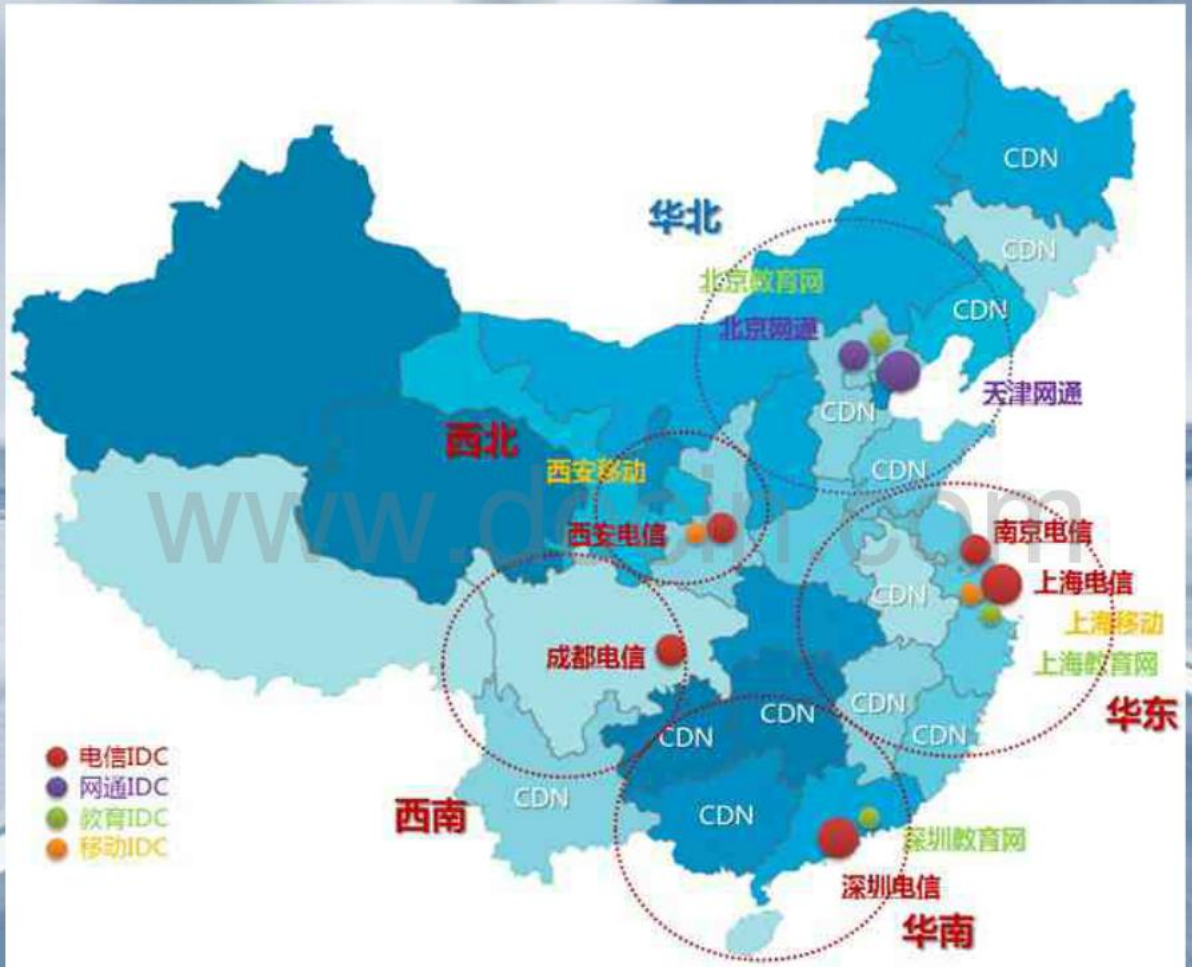
- 缓存服务器的目的
  1、优化传输路由
  2、减轻源站压力

优化路由

- 电信、网通、联通、移动、教育网、广电、铁通、卫通...中国互联网相当复杂，同在北京，网络位置却隔得很远很远~
- CDN在边缘节点上再构建一层专用光纤链路，完成全网的快速分发。

蓝讯节点示意图

帝联节点示意图


华北
CDN
CDN
北京教育网
北京网通
CDN
天津网通
CDN
西北
西安移动
CDN
西安电信
南京电信
上海电信
CDN
上海移动
上海教育网
成都电信
CDN
CDN
CDN
CDN
华东
CDN
西南
CDN
深圳教育网
深圳电信
华南
电信IDC
网通IDC
教育IDC
移动IDC

减轻源站压力

- 对于小文件(一般定义为平均1MB以下，常见网页、小图片等)，一般采用被动抓取的方式；
- 对于大文件(即平均1MB以上，游戏下载、音视频文件等)，一般采用主动推送的方式；
- 对于流媒体视频点播、直播，针对不同格式有各自的解决方案。

SQUID

- 我们目前在CDN平台加速的，都属于小文件类别。
- 小文件缓存加速，最常见的几个软件是squid/nginx/varnish，其次是lighttpd(mod_cache)/apache(traffic_server)等等
- 因为有世界最大的CDN商Akamai多年的支持，squid成为运用最广泛的缓存软件，可以说是事实上的标准。
- 我们在内部重大系统上，也使用了squid分担apache/nginx的压力。

Squid处理流程

- 我们可以通过开启debug模式，很清楚的看到一个url请求到达squid后的处理全过程。
- 1、tcp建连，占用文件描述符
  2、读取url请求，在内存中创建相应的Entry空间
  3、检查请求ACL匹配(内外2种)
  4、检查重定向规则
  5、读取匹配的缓存规则
  6、检查是否命中现有缓存文件
  回源请求文件
  7、计算过期时间
  8、检查响应ACL匹配
  9、传输文件给客户端
- 10、将可缓存的新文件存入内存
- 11、空闲(或触发阀值)时清理内存和磁盘上不用的文件

读取url

- Squid与普通web服务器第一个显著不同，squid不会把host/uri?strings分开处理，在整个entry处理过程中，squid是针对一条完整url的。
- GET /index.html HTTP/1.0(squid不认)
- GET http://www.test.com/index.html HTTP/1.0

ACL

- 我们经常使用的acl有几种：
  user-agent异常(爬虫恶意抓取)
  referer异常(爬虫恶意抓取)
  maxconn(并发攻击)
  url_regex(特定url)
- deny_info
  针对ACL的deny，可以自定义deny_info，以返回特定url

rewrite

- Squid的rewrite分几种：
- 内部rewrite，请求a.com，返回b.com的内容，但url还是a.com
- 外部rewrite，返回301/302状态码给客户端，由客户端重新发起一次url请求
- deny_info和error_map也可以算一种rewrite，分别类似于外部和内部rewrite，且保留整个原response-header

rewrite

- 由于rewrite功能是由其他程序插件来完成的，squid只是流式的提供了url等信息，所以可以利用rewrite完成很多功能。
- 比如为了防止浏览器缓存而在页面中设定了*.js?t=*，rewrite成*.js，大大提高了缓存命中率；
- 比如为了防盗链而在页面中设定了hash式的url，通过相同算法的rewrite还原，也同样只需要在squid缓存一份，还能直接在squid验证拒绝。

Rewrite

```
#! /usr/bin/env perl
use strict;
use Digest::MD5 qw(md5_hex);
use POSIX qw(difftime mktime);
$| = 1;
my $errUrl = "http://www.test.com/no_such_url.html";
my $secret = "123456";
my $expired = 7200;
while () {
my ($uri, $client, $ident, $method) = split;
print"$errUrln" and next unless ($uri =~
m#^(http://w*.?test.com)/(d{4})(d{2})(d{2})(d{2})(d{2})/(w{32})(/.+.mp3)$#i);
my ($domain, $year, $mon, $mday, $hour, $min, $md5, $path) = ($1, $2, $3, $4, $5, $6, $7, $8);
print "$errUrl\n" and next
if $year < 1990 or $mon == 0 or $mon > 12 or $mday == 0 or $mday > 31 or $hour > 23 or
$min > 59;
print "$errUrl\n" and next
if abs(difftime((int(time() / 100) * 100), mktime(00, $min, $hour, $mday, $mon – 1, $year –
1900))) > $expired;
$path =~ s#%(..)#pack("c", hex($1))#eg;
print "$errUrl\n" and next
if $md5 ne md5_hex($secret . $year . $mon . $mday . $hour . $min . $path);
print $domain . $path . "\n";
}
```

Cache_peer

- Squid在解析域名时，本着cache_peer>hosts>dns的优先级原则
- CDN商一般会有自己的内部DNS，并根据回源状态更改内部DNS解析，选择最优的源站(有多源的前提下)甚至最优的内部其他节点
- 我们本身各系统分离，直接采用cache_peer方式，还节省了dns解析的时间

Cache_peer

- Cache_peer可以利用url_regex的ACL完成粗略的7层lb；
  可以采用round-robin轮询；
  可以采用weight设置权重；
  可以采用sourcehash完成C类IP地址级别的session保持；

Cache_peer

- Squid可以通过url_regex的ACL和hierarchy_stoplist完成针对特定类别的url绕过cache_peer解析。这在CDN商的动静混合加速时比较有用。

StoreEntry

- Squid的预加载功能，会在收到同url的第一个请求时，将全部文件加载进storeentry，同时从该storeentry下发给其他客户端。
- Storeentry的部分内容(可缓存的url)，会记录进swap.state文件，这是squid的hash表(squid在内存中维护有很多hash表，这个是用来记录url的存储路径)
- 我们可以通过squidclient mgr:objects命令读取到swap.state文件的信息(见下页)

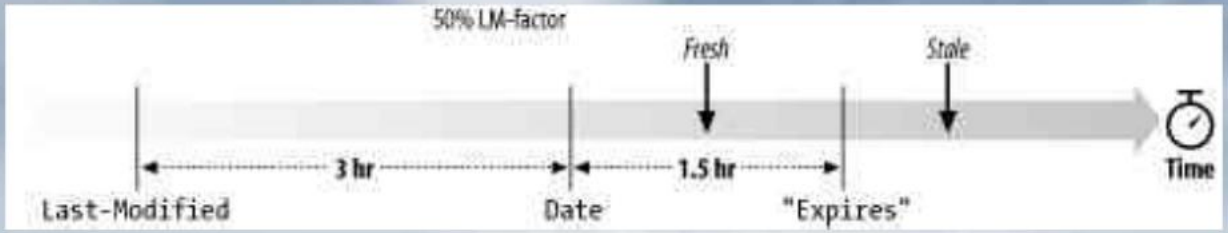
Mgr:objects

- KEY 00209C21A003A00843E54663D8241ABD
- GET http://ent.china.com/zh_cn/music/pop/11015604/20090716/15563541.html
- STORE_OK IN_MEMORY SWAPOUT_DONE PING_NONE
- CACHABLE,VALIDATED
- LV:1289578977 LU:1289578977 LM:-1 EX:1289678977
- 0 locks, 0 clients, 0 refs
- Swap Dir 0, File 0X00C99D
- inmem_lo: 0
- inmem_hi: 282
- swapout: 282 bytes queued
- 提供了最重要的key和url信息，也显示了一些存储状态信息；
- 因为squid没有提供批量刷新功能，常见的批量刷新方法就是从mgr:objects中grep响应的url。

Cache-control

- 从默认配置上来说，squid是遵循RFC的，但squid提供了一系列参数来违反HTTP……
- CDN商最常用的几个参数包括：
  ignore-reload(忽略请求header中的no-cache)
  ignore-no-cache(忽略响应header中的no-cache)
  ignore-private(忽略响应header中的private)
  reload-into-ims(将请求header中的no-cache转化成if-modified-since后再转发给cache_peer)
  override-lastmod(在小于设定的minage时忽略last-modified)
  override-expire(在小于设定的minage时忽略expires)
  header_replace Age 1(强制响应header的Age为1)
- 一般情况下，CDN默认使用ignore-reload

LM-factor



- 上图是squid官方提供的LM-factor算法解释图
- 假设a.html的mtime是0点，
  第一个到达squid的请求是3点，
  squid配置50%的话，其过期时间为
  3+(3-0)*50%=4:30
- 在3-4:30这段时间内，所有对a.html的请求响应Date都是3点！
- 5点，再次到达squid第二个相同请求，squid将发送ims去peer：
  如果mtime还是0点，则返回304，Date变成5点，过期时间则改成
  5+(5-0)*50%=7:30
  如果mtime变成了4点，则返回200，Date变成5点，过期时间则改成
  5+(5-4)*50%=5:30

Refresh_pattern

- 除了上面那个复杂的LM-factor算法外，squid还提供一个简单的min和max设置，小于min的就认为肯定新，大于max就认为肯定旧。
- 注意：squid.conf的配置，都是基于分钟级别的控制，即squid无法缓存expires设定只有几十秒的文件……
- 完整一条refresh_pattern配置如下：
  refresh_pattern -i ^http://.*\.china.com\.com/[^0-9]+/.*\.(jpg|gif|png)$ 1440 20% 4320 ignore-reload

Reply_access

- 响应ACL，即在完成请求响应时的过滤。
- 一般是针对一些特定的response-header做设定。
  比如之前说到的Age 1，这是一种永不过期设定；
  比如为了保护服务器屏蔽掉一些版本信息等……

GZIP

- 影响网页浏览速度的，除了dns解析时间、网路建连时间、server处理时间外，还有本身文件传输的时间。这时候文件大小还是有比较大影响的。这就是gzip的用途。
- 目前squid可以支持gzip的静态压缩。因为squid在缓存时会自己给文件加上content_length，而chunked压缩分块是没有的。

Vary

- Vary是RFC规范要求的web服务器响应header内容
- Squid在计算url的key时，同时考虑url、method和vary三个值。所以即使url不变的情况下，因为客户端或者web端的变化，也可能导致多份缓存
- 考虑到gzip的支持，建议的情况是
  Vary: Accept-Encoding
  Accept-Encoding:gzip,deflate(IE和FF不同，差一个空格==!)

replacement_policy

- 空间毕竟是有限的，squid提供了几种内存和缓冲空间更新算法
  LRU:大多数缓存默认的算法，最近最少使用——目的就是保留热点文件
  heap GDSF:贪婪对偶大小——在小文件时请求命中率较高
  heap LFUDA:动态衰老最少使用——在缓存部分冷文件的条件下提高字节命中率

replacement_policy

- Memory_replacement_policy只能采用一种算法
- Cache_replacement_policy可以更改，配合cache_dir指令完成对不同文件大小不同路径缓存采用不同老化算法以尽可能的发挥性能~

Cache_dir

- Cache_dir可以指定几种存储方式：
  ufs：默认方式
  aufs：异步IO改进型，对较大文件效果较好
  diskd：独有磁盘清理进程改进型
  coss：新文件系统，无需raid和ext，直接将裸盘视为一个文件，然后由squid自己在文件中进行读取、老化等操作，目前稳定性存疑，对小文件测试效果极好

Cache_dir

- cache_replacement_policy heap LFUDA
  cache_dir aufs /tmp/cache1 100 4 16 min-size=4096KB
  cache_replacement_policy lru
- cache_dir diskd /tmp/cache2 100 4 16 min-size=1024KB max-size=4096KB

maximum_object_size

- maximum_object_size规定了squid可缓存的最大文件大小，一旦超出，squid会把自己辛辛苦苦下载完的文件丢弃掉……
- maximum_object_size_in_memory同理
- 由于squid工作机制的原因，以上两个值建议是8KB的倍数以减少碎片，提高利用率

method

- Squid接到一个method为purge的请求时，按照普通method的处理流程进行，以找到相应的key并删除文件。
- Squid接到一个method为POST的请求时，转发请求全部，且不缓存响应。