Squid 中文权威指南

（第 13 章）

译者序：

本人在工作中维护着数台 Squid 服务器，多次参阅 Duane Wessels（他也是 Squid 的创始人）的这本书，原书名是"Squid: The Definitive Guide"，由 O'Reilly 出版。我在业余时间把它翻译成中文，希望对中文 Squid 用户有所帮助。对普通的单位上网用户，Squid 可充当代理服务器；而对 Sina,NetEase 这样的大型站点，Squid 又充当 WEB 加速器。这两个角色它都扮演得异常优秀。窗外繁星点点，开源的世界亦如这星空般美丽，而 Squid 是其中耀眼的一颗星。

对本译版有任何问题，请跟我联系，我的Email是：yonghua_peng@yahoo.com.cn

彭勇华

目 录

第 13 章 日志文件

13.1 cache.log

cache.log 包含多种消息，例如 Squid 的配置信息、性能警告、以及严重错误。如下是 cache.log 的输出样本：

2003/09/29 12:09:45| Starting Squid Cache version 2.5.STABLE4 for i386-

unknown-freebsd4.8...

2003/09/29 12:09:45| Process ID 18990

2003/09/29 12:09:45| With 1064 file descriptors available

2003/09/29 12:09:45| Performing DNS Tests...

2003/09/29 12:09:45| Successful DNS name lookup tests...

2003/09/29 12:09:45| DNS Socket created at 0.0.0.0, port 1154, FD 5

2003/09/29 12:09:45| Adding nameserver 24.221.192.5 from /etc/resolv.conf

2003/09/29 12:09:45| Adding nameserver 24.221.208.5 from /etc/resolv.conf

2003/09/29 12:09:45| helperOpenServers: Starting 5 'redirector.pl' processes

2003/09/29 12:09:45| Unlinkd pipe opened on FD 15

2003/09/29 12:09:45| Swap maxSize 10240 KB, estimated 787 objects

2003/09/29 12:09:45| Target number of buckets: 39

2003/09/29 12:09:45| Using 8192 Store buckets

2003/09/29 12:09:45| Max Mem size: 8192 KB

2003/09/29 12:09:45| Max Swap size: 10240 KB

2003/09/29 12:09:45| Rebuilding storage in /usr/local/squid/var/cache (CLEAN)

2003/09/29 12:09:45| Using Least Load store dir selection

2003/09/29 12:09:45| Set Current Directory to /usr/local/squid/var/cache

2003/09/29 12:09:45| Loaded Icons.

2003/09/29 12:09:45| Accepting HTTP connections at 0.0.0.0, port 3128, FD 16.

2003/09/29 12:09:45| Accepting ICP messages at 0.0.0.0, port 3130, FD 17.

2003/09/29 12:09:45| WCCP Disabled.

2003/09/29 12:09:45| Ready to serve requests.

每个 cache.log 条目以时间戳开始，指示消息何时产生。本示例里的日志报告了 squid 的版本（2.5.STABLE4），以及 squid 所运行的操作系统标识符（i386-unknown-freebsd4.8）。接下来是进程 ID（18990）。许多 cache.log 条目看起来含义不明（例如 Target number of buckets: 39）。大多数正常情形下，可以忽略这些不易理解的条目。另一方面，你也许该仔细看一下本质的配置细节，例如名字服务器的地址，或 HTTP 服务器地址。本示例日志最后陈述了 Squid 准备接受请求。此时 Squid 可以接受来自客户端的 HTTP 连接。

通常，cache.log 增长缓慢。然而，不正常的 HTTP 事务或类似的事件可以导致 squid 发布一个 debug 消息。假如这样的事件经常发生（例如 DOS 攻击、新的病毒、磁盘意外等），日志文件会增长很快。定期轮转日志减少了用光磁盘的风险。

主要的错误和异常条件最可能报告在 cache.log 里。我推荐存档这些日志，以便以后回查事件的源头。当在 Squid 的邮件列表或类似论坛描述这些故障时，相应的 cache.log 非常有用。某些情形下，你也许应该调大日志的 debug 级别，以便其他人能更好的理解和修正你的问题。

13.1.1 debug 级别

debug_options 指令控制 cache.log 的日志级别。默认值（ALL,1）通常是最佳选择。在更高级别上，不重要的消息会混淆视线。请参考 16.2 节关于 debug_options 指令的完整描述。

请注意最高级别的 debug（9 或 10）会对每个请求产生数千行日志，快速消耗磁盘空间和显著影响 squid 的性能。

可以使用 squid 的-X 命令行选项来对所有情形激活完整的 debug。假如 squid 拒绝启动，并且 squid.conf 里的 debug 级别不足以诊断问题时，该模式特别有用。这也是在配置文件解析器解析到 debug_options 指令之前，激活它的完整 debug 的好方法。在 squid 运行正常时，请勿使用-X。

对运行的 squid 进程，可使用 squid 的-k debug 命令行选项来立刻激活完整 debug。这个命令是循环使用的：第一次调用打开完整 debug，第二次调用则关闭它。请见第 5 章关于-k

选项的通用讨论。

如前所述，完整 debug 会产生难以控制的日志增长。这会使 squid 和操作系统运行缓慢。在极端情形下，你会发现终端 session 在运行第一个 squid -k debug 命令后，变得没有响应。在 squid 狂写日志的同时让操作无法进行，这情形并不好。如下技巧也许有用，它获取 5 秒钟的 debug 数据快照：

```
% squid -k debug; sleep 5; squid -k debug
```

13.1.2 转发 cache.log 消息到系统日志

为了让 squid 发送 cache.log 消息的拷贝到系统日志，请使用-s 命令行选项。仅仅在 debug 级别 0 和 1 的消息会被转发。级别 0 的消息以 syslog 级别 LOG_WARNING 记录，级别 1 的消息以 syslog 级别 LOG_NOTICE 记录。所有消息使用 LOCAL4 的 syslog 设备。如下是配置 syslogd 的一个方法，以便这些消息能保存下来：

```
local4.warning                  /var/log/squid.log
```

在维护多个 squid 主机时，使用 syslog 来记录 cache.log 特别方便。可以配置每个本地 syslog 进程，转发这些消息到中央日志主机，这样就可在一个地方统一浏览所有 cache 日志。例如，可在/etc/syslogd.conf 里使用如下接口：

```
local4.notice                   @192.168.45.1
```

13.1.3 dump cache.log 消息到终端

-d level 命令行选项指示 squid 去 dump cache.log 消息到终端（例如 stderr）。level 参数指明 dump 出的消息的最大级别。注意你只会见到出现在 cache.log 里的消息，它遵循于 debug_options 设置。例如，假如设置了 debug_options ALL,1，然后运行 squid -d2，你不会见到级别 2 的 debug 消息。

-d level 和-N 选项在 debug squid 问题或快速测试配置文件的改变时，特别有用。它们允许你容易启动 squid 和观察 cache.log 消息。在 squid 从 crontab 或类似的设备启动时，该选项也有用，crontab 会捕获 squid 的标准错误并将其报告回用户。例如，可能有如下 crontab，它自动重配运行中的 squid 进程：

```
15 */4 * * * /usr/local/squid/sbin/squid -d1 -k reconfigure
```

13.2 access.log

Squid 把关于 HTTP 事务的关键信息存放在 access.log 里。该文件是基于行的，也就是说每行对应一个客户端请求。squid 记录客户端 IP（或主机名）、请求 URL、响应 size、和其他

信息。

Squid 在 access.log 里记录所有 HTTP 访问，除了那些在还没有发送数据前就断开的连接。Squid 也记录所有的 ICP（非 HTCP）事务，除非你使用 log_icp_queries 指令关闭了这个功能。第 13.2.4 节描述了其他影响 access 日志的 squid.conf 指令。

默认的 access.log 格式包含了 10 个域。如下是日志样本，长行分割并且缩进排版：

```
1066037222.011    126389 9.121.105.207 TCP_MISS/503 1055
        GET http://home.gigigaga.com/n8342133/Miho.DAT.019 -
        DIRECT/203.187.1.180 -
1066037222.011      19120 12.83.179.11 TCP_MISS/200 359
        GET http://ads.x10.com/720x300/Z2FtZ3JlZXRpbmcxLmRhd/7/AMG -
        DIRECT/63.211.210.20 text/html
1066037222.011      34173 166.181.33.71 TCP_MISS/200 559
        GET http://coursesites.blackboard.com:8081/service/collab/../1010706448190/ -
        DIRECT/216.200.107.101 application/octet-stream
1066037222.011      19287 41.51.105.27 TCP_REFRESH_MISS/200 500
        GET http://fn.yam.com/include/tsemark/show.js -
        DIRECT/210.59.224.59 application/x-javascript
1066037222.011      19395 41.51.105.27 TCP_MISS/304 274
        GET http://fnasp.yam.com/image/coin3.gif -
        DIRECT/211.72.254.133 -
1066037222.011      19074 30.208.85.76 TCP_CLIENT_REFRESH_MISS/304 197
        GET  http://ads.icq.com/content/B0/0/..bC6GygEYNeHGjBUin5Azfe68m5hD1jLk$/aol
-
        DIRECT/64.12.184.121 -
```

1066037222.011 19048 12.83.179.11 TCP_MISS/200 261

GET http://ads.adsag.com/js.ng/...ne&cat=friendship&subcat=girltalk -

DIRECT/209.225.54.119 application/x-javascript

1066037222.118 106 41.51.105.27 TCP_HIT/200 536

GET http://rcm-images.amazon.com/images/G/01/rcm/privacy.gif -

NONE/- image/gif

1066037222.352 19475 27.34.49.248 TCP_MISS/200 12387

GET http://espanol.geocities.com/lebastias/divulgacion/budismo-tarot.html -

DIRECT/209.1.225.139 text/html

1066037222.352 132 144.157.100.17 TCP_MISS/504 1293

GET http://ar.atwola.com/image/93101912/aol -

NONE/- -

如下是对每个域的详细解释：

1.时间戳

请求完成时间，以 Unix 纪元（UTC 1970-01-01 00:00:00）以来的秒数表示，它是毫秒级的。squid 使用这种格式而不是人工可读的时间格式，是为了简化某些日志处理程序的工作。

可以使用一个简单的 perl 命令来转化 Unix 时间戳到本地时间，例如：

perl -pe 's/^\d+\.\d+/localtime($&)/e;' access.log

2.响应时间

对 HTTP 事务来说，该域表明 squid 花了多少时间来处理请求。在 squid 接受到 HTTP 请求时开始计时，在响应完全送出后计时终止。响应时间是毫秒级的。

对 ICP 查询来说，响应时间通常是 0。这是因为 squid 回答 ICP 查询非常迅速。甚至，squid 在接受到 ICP 查询和发送完响应之间，不会更新进程时钟。

尽管时间值是毫秒级的，但是精度可能是 10 毫秒。在 squid 负载繁重时，计时变得没那么精确。

3.客户端地址

该域包含客户端的 IP 地址，或者是主机名--假如激活了 log_fqdn。出于安全或隐私的理由，你可能需要使用 client_netmask 指令来掩盖客户端地址的一部分。然而，这样让来自同一客户端的组请求变得不可能。

4.结果/状态码

该域包含 2 个 token，以斜杠分隔。第一个 token 叫结果码，它把协议和事务结果（例如 TCP_HIT 或 UDP_DENIED）进行归类。这些是 squid 专有的编码，在 13.2.1 节里有定义。以 TCP_开头的编码指 HTTP 请求，以 UDP_开头的编码指 ICP 查询。

第 2 个 token 是 HTTP 响应状态码（例如 200,304,404 等）。状态码通常来自原始服务器。在某些情形下，squid 可能有义务自己选择状态码。这些编码在 HTTP 的 RFC 里定义，在随后的 Table 13-1 里有概述。

5.传输 size

该域指明传给客户端的字节数。严格的讲，它是 squid 告诉 TCP/IP 协议栈去发送给客户端的字节数。这就是说，它不包括 TCP/IP 头部的 overhead。也请注意，传输 size 正常来说大于响应的 Content-Length。传输 size 包括了 HTTP 响应头部，然而 Content-Length 不包括。

传输 size 可用于近似的带宽使用分析，但并非精确的 HTTP 实体 size 计算。假如需要了解响应的 Content-Length，可在 store.log 里找到它。

6.请求方式

该域包含请求方式。因为 squid 客户端可能使用 ICP 或 HTTP，请求方式就可能是 HTTP-或 ICP-这 2 种。最普通的 HTTP 请求方式是 GET。ICP 查询总以 ICP_QUERY 的形式被记载。请见 6.1.2.8 节关于 squid 了解的 HTTP 方式列表。

7.URI

该域包含来自客户端请求的 URI。大多数记录下来的 URI 实际是 URL（例如，它们有主机名）。

Squid 对某些失败使用特殊的记录格式。例如 Squid 不能解析 HTTP 请求，或者不能决定 URI，这时你可能见到类似于"error:invalid-request." 的字串出现在 URI 的位置。例如：

1066036250.603 310 192.0.34.70 NONE/400 1203 GET error:invalid-request - NONE/- -

另外在该域里，也请留心 URI 里的空格字符。取决于 uri_whitespace 设置，squid 可能在日志文件里打印 URI 时带空格字符。若发生这种情况，则阅读 access.log 文件的日志分析工具可能会遇到麻烦。

在记日志时，squid 删掉了在第一个问号(?)之后的所有 URI 字符，除非禁用了 strip_query_terms 指令。

8.客户端身份

Squid 有 2 种不同的办法来决定用户的身份。一种是 RFC 1413 身份协议，另一种来自 HTTP 验证头部。

Squid 试图基于 ident_lookup_access 规则进行身份查询，假如有的话。另外，假如使用代理验证（或在代理人模式下的规范服务验证），squid 会在该域放置给定的用户名。假如 2 者都提供给 squid 一个用户名，并且你使用了原始 access.log 格式，那么 HTTP 验证名字会记录下来，RFC 1413 名字会忽略掉。普通日志文件格式会把两者都独立的记录。

9.对端编码/对端主机

对端信息包含了 2 个 token，以斜杠分隔。它仅仅与 cache 丢失的请求有关。第一个 token 指示如何选择下一跳，第二个 token 是下一跳的地址。对端编码列在 13.2.3 节里。

当 squid 发送一个请求到邻居 cache 时，对端主机地址是邻居的主机名。假如请求是直接送到原始服务器的，则 squid 会写成原始服务器的 IP 地址或主机名--假如禁用了 log_ip_on_direct。NONE/-这个值指明 squid 不转发该请求到任何其他服务器。

10.内容类型

原始 access.log 的默认的最后一个域，是 HTTP 响应的内容类型。squid 从响应的 Content-Type 头部获取内容类型值。假如该头部丢失了，squid 使用一个横杠(-)代替。

假如激活了 log_mime_headers 指令，squid 在每行追加 2 个附加的域：

11.HTTP 请求头部

Squid 编码 HTTP 请求头部，并且在一对方括号之间打印它们。方括号是必须的，因为 squid 不编码空格字符。编码方案稍许奇怪。回车（ASCII 13）和换行（ASCII 10）分别打印成\r 和\n。其他不可打印的字符以 RFC 1738 风格来编码，例如 Tab（ASCII 9）变成了%09。

12.HTTP 响应头部

Squid 编码 HTTP 响应头部，并且在一对方括号之间打印它们。注意这些是发往客户端的头部，可能不同于从原始服务器接受到的头部。

Squid 只有在整个响应发送到客户端完成以后，才写 access.log 日志。这点允许 squid 在日志文件里包含请求和响应两者信息。然而，需要花费数分钟甚至数小时才能完成的事务，请求期间的日志在 access.log 里不可见。当这类型的事务呈现出性能或策略问题时，access.log 可能对你没有帮助。代替的，可使用 cache 管理器来浏览挂起事务的列表（见 14 章）。

13.2.1 access.log 结果编码

相应于 HTTP 请求，下列标签可能出现在 access.log 文件的第四个域。

TCP_HIT

Squid 发现请求资源的貌似新鲜的拷贝，并将其立即发送到客户端。

TCP_MISS

Squid 没有请求资源的 cache 拷贝。

TCP_REFERSH_HIT

Squid 发现请求资源的貌似陈旧的拷贝，并发送确认请求到原始服务器。原始服务器返回 304（未修改）响应，指示 squid 的拷贝仍旧是新鲜的。

TCP_REF_FAIL_HIT

Squid 发现请求资源的貌似陈旧的拷贝，并发送确认请求到原始服务器。然而，原始服务器响应失败，或者返回的响应 Squid 不能理解。在此情形下，squid 发送现有 cache 拷贝（很可能是陈旧的）到客户端。

TCP_REFRESH_MISS

Squid 发现请求资源的貌似陈旧的拷贝，并发送确认请求到原始服务器。原始服务器响应新的内容，指示这个 cache 拷贝确实是陈旧的。

TCP_CLIENT_REFRESH_MISS

Squid 发现了请求资源的拷贝，但客户端的请求包含了 Cache-Control: no-cache 指令。Squid 转发客户端的请求到原始服务器，强迫 cache 确认。

TCP_IMS_HIT

客户端发送确认请求，Squid 发现更近来的、貌似新鲜的请求资源的拷贝。Squid 发送更新的内容到客户端，而不联系原始服务器。

TCP_SWAPFAIL_MISS

Squid 发现请求资源的有效拷贝，但从磁盘装载它失败。这时 squid 发送请求到原始服务器，就如同这是个 cache 丢失一样。

TCP_NEGATIVE_HIT

在对原始服务器的请求导致 HTTP 错误时，Squid 也会 cache 这个响应。在短时间内对这些资源的重复请求，导致了否命中。negative_ttl 指令控制这些错误被 cache 的时间数量。请注意这些错误只在内存 cache，不会写往磁盘。下列 HTTP 状态码可能导致否定 cache（也遵循于其他约束）： 204, 305, 400, 403, 404, 405, 414, 500, 501, 502, 503, 504。

TCP_MEM_HIT

Squid 在内存 cache 里发现请求资源的有效拷贝，并将其立即发送到客户端。注意这点并非精确的呈现了所有从内存服务的响应。例如，某些 cache 在内存里，但要求确认的响应，会以 TCP_REFRESH_HIT, TCP_REFRESH_MISS 等形式记录。

TCP_DENIED

因为 http_access 或 http_reply_access 规则，客户端的请求被拒绝了。注意被 http_access 拒绝的请求在第 9 域的值是 NONE/-，然而被 http_reply_access 拒绝的请求，在相应地方有一个有效值。

TCP_OFFLINE_HIT

当 offline_mode 激活时，Squid 对任何 cache 响应返回 cache 命中，而不用考虑它的新鲜程度。

TCP_REDIRECT

重定向程序告诉 Squid 产生一个 HTTP 重定向到新的 URI（见 11.1 节）。正常的，Squid 不会记录这些重定向。假如要这样做，必须在编译 squid 前，手工定义 LOG_TCP_REDIRECTS 预处理指令。

NONE

无分类的结果用于特定错误，例如无效主机名。

相应于 ICP 查询，下列标签可能出现在 access.log 文件的第四域。

UDP_HIT

Squid 在 cache 里发现请求资源的貌似新鲜的拷贝。

UDP_MISS

Squid 没有在 cache 里发现请求资源的貌似新鲜的拷贝。假如同一目标通过 HTTP 请求，就可能是个 cache 丢失。请对比 UDP_MISS_NOFETCH。

UDP_MISS_NOFETCH

跟 UDP_MISS 类似，不同的是这里也指示了 Squid 不愿去处理相应的 HTTP 请求。假如使用了-Y 命令行选项，Squid 在启动并编译其内存索引时，会返回这个标签而不是 UDP_MISS。

UDP_DENIED

因为 icp_access 规则，ICP 查询被拒绝。假如超过 95%的到某客户端的 ICP 响应是 UDP_DENIED，并且客户端数据库激活了（见附录 A），Squid 在 1 小时内，停止发送任何 ICP 响应到该客户端。若这点发生，你也可在 cache.log 里见到一个警告。

UDP_INVALID

Squid 接受到无效查询（例如截断的消息、无效协议版本、URI 里的空格等）。Squid 发送 UDP_INVALID 响应到客户端。

13.2.2 HTTP 响应状态码

Table 13-1 列出了数字 HTTP 响应 CODE 和理由短句。注意 Squid 和其他 HTTP 客户端仅仅关注这些数字值。理由短句是纯解释性的，不会影响响应的意义。对每个状态码，也提供了一个到 RFC 2616 的具体节的索引。注意状态码 0 和 600 是 squid 使用的非标准的值，不会在 RFC 里提到。

Table 13-1. HTTP response status codes

Code	Reason phrase	RFC 2616 section
0	No Response Received (Squid-specific)	N/A
1xx	Informational	10.1
100	Continue	10.1.1
101	Switching Protocols	10.1.2
2xx	Successful	10.2
200	OK	10.2.1
201	Created	10.2.2
202	Accepted	10.2.3

Table 13-1. HTTP response status codes		
Code	Reason phrase	RFC 2616 section
203	Non-Authoritative Information	10.2.4
204	No Content	10.2.5
205	Reset Content	10.2.6
206	Partial Content	10.2.7
3xx	Redirection	10.3
300	Multiple Choices	10.3.1
301	Moved Permanently	10.3.2
302	Found	10.3.3
303	See Other	10.3.4
304	Not Modified	10.3.5
305	Use Proxy	10.3.6
306	(Unused)	10.3.7
307	Temporary Redirect	10.3.8
4xx	Client Error	10.4
400	Bad Request	10.4.1
401	Unauthorized	10.4.2
402	Payment Required	10.4.3
403	Forbidden	10.4.4
404	Not Found	10.4.5
405	Method Not Allowed	10.4.6
406	Not Acceptable	10.4.7
407	Proxy Authentication Required	10.4.8
408	Request Timeout	10.4.9
409	Conflict	10.4.10
410	Gone	10.4.11
411	Length Required	10.4.12
412	Precondition Failed	10.4.13
413	Request Entity Too Large	10.4.14

Table 13-1. HTTP response status codes

Code	Reason phrase	RFC 2616 section
414	Request-URI Too Long	10.4.15
415	Unsupported Media Type	10.4.16
416	Requested Range Not Satisfiable	10.4.17
417	Expectation Failed	10.4.18
5xx	Server Error	10.5
500	Internal Server Error	10.5.1
501	Not Implemented	10.5.2
502	Bad Gateway	10.5.3
503	Service Unavailable	10.5.4
504	Gateway Timeout	10.5.5
505	HTTP Version Not Supported	10.5.6
6xx	Proxy Error	N/A
600	Unparseable Response Headers (Squid-specific)	N/A

假如 Squid 从原始服务器没有接受到任何响应，你可在 access.log 里看到状态码 0。假如 Squid 接受到的响应没有包含 HTTP 头部，就会出现状态码 600。在少数情况下，某些原始服务器仅发送响应 body，而忽略了任何头部。

13.2.3 access.log 对端编码

下列编码可能出现在 access.log 的第 9 域。请参考 10.10 节关于 Squid 如何对 cache 丢失情况，选择有效的下一跳。

NONE

这指明 Squid 对本次请求，不会与任何其他服务器（邻居或原始服务器）通信。它通常与 cache 命中、拒绝请求、cache 管理请求、错误、和所有的 ICP 查询这些类型联合出现。

DIRECT

Squid 直接转发请求到原始服务器。该域的第 2 半部分显示原始服务器的 IP 地址，或主机名--假如禁止了 log_ip_on_direct。

SIBLING_HIT

在姐妹 cache 返回 ICP 或 HTCP 命中后，Squid 发送请求到姐妹 cache。

PARENT_HIT

在父 cache 返回 ICP 或 HTCP 命中后，Squid 发送请求到父 cache。

DEFAULT_PARENT

Squid 选择该父 cache，因为其在 squid.conf 的 cache_peer 行里被标志为 default。

FIRST_UP_PARENT

Squid 转发请求到该父 cache，因为它是位于已知活跃列表里的第一个父 cache。

FIRST_PARENT_MISS

Squid 转发请求到该父 cache，它第一个响应 ICP/HTCP 丢失消息。换句话说，对这个特殊的 ICP/HTCP 查询，在这个特殊时刻，被选中的父 cache 有最佳的往返时间（RTT）。注意标准 RTT 可能被人工矫正过，取决于 cache_peer 指令的 weight 选项。

CLOSEST_PARENT_MISS

Squid 选择该父 cache，因为它报告到原始服务器的 RTT 最低。这点仅在 2 个 cache 都激活了 netdb，并且原始服务器（或在同一子网内的其他 server）返回 ICMP ping 消息。

CLOSEST_PARENT

这点类似 CLOSEST_PARENT_MISS，除了 RTT 计算不是来自 ICP/HTCP 响应消息外。代替的，它们来自 Squid 保留的更老的计算方式，例如 netdb 交换功能。

CLOSEST_DIRECT

Squid 基于 netdb 算法，转发请求到原始服务器。这点在满足下述任何条件时发生：

1）在 Squid 和原始服务器之间的 RTT 小于配置的 minimum_direct_rtt 值。
2）在 Squid 和原始服务器之间的标准路由跳数少于配置的 minimum_direct_hops 值。
3）在 ICP/HTCP 响应里返回的 RTT 值，指示 Squid 离原始服务器近于任何其他邻居。

ROUNDROBIN_PARENT

Squid 转发请求到该父 cache，因为设置了 round-robin 选项，并且它有最低的使用计数器。

CD_PARENT_HIT

Squid 基于 cache 摘要算法（见 10.7 节）转发请求到该父 cache。

CD_SIBLING_HIT

Squid 基于 cache 摘要算法转发请求到该姐妹 cache。

CARP

Squid 选择该父 cache，基于 cache 数组路由协议算法（见 10.9 节）。

ANY_PARENT

作为最后的手段，Squid 选择该父 cache，因为没有其他方法能选择可行的下一跳。

注意大部分上述编码可能以 TIMEOUT_开头，这表明在等待 ICP/HTCP 响应时发生超时。例如：

1066038165.382 345 193.233.46.21 TCP_MISS/200 2836

GET http://www.caida.org/home/images/home.jpg

TIMEOUT_CLOSEST_DIRECT/213.219.122.19 image/jpeg

可使用 icp_query_timeout 指令来调整超时。

13.2.4 影响 access.log 的配置指令

下列配置文件指令会影响到 access.log。

13.2.4.1 log_icp_queries

该指令默认激活，导致 squid 记录所有的 ICP 查询。假如运行了一个繁忙的父 cache，这点可能让 access.log 文件变得巨大。为了节省磁盘空间，可禁止该指令：

log_icp_queries off

假如禁止了 ICP 查询的日志，我建议你监视查询数量--通过 cache 管理器或 SNMP。

13.2.4.2 emulate_httpd_log

access.log 文件有 2 种格式：普通格式和原始格式。普通格式就如同大部分 HTTP 服务器（如 Apache）的日志格式一样。它包含的信息少于 Squid 的原始格式。然而，假如运行 Squid 在代理人模式下（见 15 章），你可能想要普通日志文件格式。普通格式或许也对你现有的日志

文件分析工具有用。使用该指令来激活普通格式：

emulate_httpd_log on

请见 http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format 关于该格式的描述。

13.2.4.3 log_mime_hdrs

使用 log_mime_hdrs 让 squid 记录 HTTP 请求和响应的头部：

log_mime_headers on

在激活时，squid 追加请求和响应头部到 access.log。这会在每行增加 2 个域。每个域都以方括号引用起来，便于分析。某些字符会被编码来保证日志文件可读。Table 13-2 显示了这些编码方案。

Table 13-2. Character encoding rules for HTTP headers in access.log

Character	Encoding
Newline	\n
Carriage return	\r
Backslash	\\
[	%5b
]	%5d
%	%25
ASCII 0-31	%xx (hexadecimal value)
ASCII 127-255	%xx (hexadecimal value)

13.2.4.4 log_fqdn

Squid 默认把客户端 IP 地址放在 access.log 里。也可以记录可用的主机名，激活如下指令：

log_fqdn on

这点让 Squid 在接受到请求时，对客户端的地址发起反向 DNS 查询。假如在请求完成时查到了主机名，Squid 就将它放在第 3 域。

13.2.4.5 ident_lookup_access

该访问规则列表决定 Squid 是否对客户端的 TCP 连接发起 RFC 1413 身份查询。默认情况下，Squid 不会发布身份查询。为了激活这点，简单的增加一个或多个规则：

```
acl All src 0/0
ident_lookup_access allow All
```

假如在请求完成时查到了答案，Squid 将其放在第 8 域。假如同时使用了 HTTP 验证，从验证得到的用户名会取代身份查询答案。

13.2.4.6 log_ip_on_direct

当 Squid 转发 cache 丢失到原始服务器时，它在第 9 域记录原始服务器的 IP 地址。可以禁止这个指令，以便 squid 记录主机名：

```
log_ip_on_direct off
```

在此情形下，主机名来自于 URI。假如 URI 包含了 IP 地址，Squid 不会将其转换为主机名。

13.2.4.7 client_netmask

该指令存在主要是为了保护用户的隐私。不同于记录完整的 IP 地址，你也可以掩盖一些位。例如：

```
client_netmask 255.255.255.0
```

在此设置下，access.log 里的所有客户端 IP 地址的最后一个八位组是 0：

```
1066036246.918      35 163.11.255.0 TCP_IMS_HIT/304 266 GET http://...
1066036246.932      16 163.11.255.0 TCP_IMS_HIT/304 266 GET http://...
1066036247.616     313 140.132.252.0 TCP_MISS/200 1079 GET http://...
1066036248.598   44459 140.132.252.0 TCP_MISS/500 1531 GET http://...
1066036249.230      17 170.210.173.0 TCP_IMS_HIT/304 265 GET http://...
1066036249.752    2135 140.132.252.0 TCP_MISS/200 50230 GET http://...
1066036250.467       4 170.210.173.0 TCP_IMS_HIT/304 265 GET http://...
1066036250.762     102 163.11.255.0 TCP_IMS_HIT/304 265 GET http://...
1066036250.832      20 163.11.255.0 TCP_IMS_HIT/304 266 GET http://...
```

1066036251.026 74 203.91.150.0 TCP_CLIENT_REFRESH_MISS/304 267 GET http://...

13.2.4.8 strip_query_terms

该指令是另一个隐私保护功能。在记录请求前，Squid 删除了查询条件。假如日志文件不幸落入坏人之手，他们不会找到任何用户名和密码。当该指令激活时，在问号(?)之后的所有字节被删除。例如，某个 URI 如下：

http://auto.search.msn.com/response.asp?MT=www.kimo.com.yw&srch=3&prov=&utf8

会被记录为：

http://auto.search.msn.com/response.asp?

13.2.4.9 uri_whitespace

早前我提到过出现在某些 URI 里的空格字符的问题。RFC 申明 URI 必须不包括空格字符，但在实际中情况并非如此。uri_whitespace 指令指明 Squid 如何处理这种情况。允许的设置是：strip (default), deny, allow, encode, 和 chop。在这些设置里，strip,encode 和 chop 保证 URI 域不包含任何空格字符（空格字符会给 access.log 增加多余的域）。

allow 设置允许请求不加修改的通过 Squid。它很可能会给重定向器和日志文件解析器带来麻烦。与之相反的是 deny 设置，它导致 Squid 拒绝这种请求。用户会接受到错误消息，但请求仍带着空格字符被记录到 access.log。

假如设置为 encode，Squid 将空格字符按 RFC 1738 规范来编码。这点其实用户代理应该先做到。chop 设置导致 Squid 把第一个空格字符后的 URI 都截断。

默认设置是 strip，它让 Squid 从 URI 里移除空格字符。这确保日志文件解析器和重定向器工作正常，但可能会破坏某些事情，例如不正确编码的搜索引擎查询。

13.2.4.10 buffered_logs

默认情况下，Squid 禁止写 cache.log 文件的 buffer，这允许你运行 tail -f 命令实时的观察日志文件变化。假如你认为这点导致不必要的性能开销，就可以禁用 buffer：

buffered_logs off

然而，除非以完整 debug 模式运行 Squid，这点可能无关紧要。注意该选项仅仅影响 cache.log。其他的日志文件总使用非缓冲的写方式。

13.2.5 access.log 分析工具

access.log包含很多信息，远不止你简单的浏览该文件所见。为了完整的浏览，必须使用第三方的日志文件分析包。你可在Squid的web页面的链接里，找到它们的列表。或者直接访问：http://www.squid-cache.org/Scripts/

最流行的工具之一是Calamaris -- 一个Perl脚本，解析日志文件并产生基于文本的或HTML的报告。它提供关于会话的详细分类包括请求方式、客户端IP地址、原始服务器域名、内容类型、文件名扩展、响应size、以及更多。Calamaris也报告ICP查询会话，甚至其他cache产品的日志分析。其站点是：http://calamaris.cord.de

Squeezer 以及它的派生 Squeezer2，是 Squid 专有的分析工具。它们提供许多统计，能帮助你了解 Squid 的性能，特别是在有邻居 cache 时。两者都产生 HTML 文件作为输出。squid-cache.org 站点的 Logfile Analysis 页有这些程序的链接。

Webalyzer是另一个有用工具。它运行快速，并且产生带表格和柱形统计表的HTML页面。它原始是设计成分析原始服务器的访问日志的。尽管它能解析Squid的日志，但不会报告诸如命中率和响应时间的事件。它使用的某些条款不同于我的做法。例如，Webalyzer把任何请求叫做一个"命中"，这不同于cache命中。它也把"页面"和"文件"加以区别。更多信息请访问Webalyzer的主页：http://www.mrunix.net/webalyzer/.

13.3 store.log

store.log 记录 Squid 关于存储或删除 cache 目标的决定。对每个存在 cache 里的目标、每个不可 cache 的目标、以及每个被轮换策略删除的目标，Squid 都会创建相应的日志条目。该日志文件内容既包含了内存 cache 又包含了磁盘 cache。

store.log 提供了下述不能从 access.log 获取的内容：

1）某个特定的响应是否被 cache。
2）cache 目标的文件号。对 UFS 基础的存储机制，你可转换该文件号到路径名，并且检查 cache 文件的内容。
3）响应的内容长度：包括 Content-Length 值和实际的 body 大小。
4）Date, Last-Modified, 和 Expires 头部的值。
5）响应的 cache 关键字（例如 MD5 哈希值）。

如你所见，这些都是相对低级的信息，在日常管理中可能用不上。除非你要做专业的分析，或打算 debug 某程序，否则 store.log 可有可无。可以如下来禁止它：

cache_store_log none

跟其他日志文件一样，Squid 将最新的日志条目写到该文件的末尾。某个给定的 URI 可能出现在日志文件里多次。例如，它先被 cache，然后删除，接着又 cache 住。仅仅最近来的日

志条目才反映目标的当前值。

store.log 是文本基础的，看起来如下：

```
1067299212.411 RELEASE -1 FFFFFFFF A5964B32245AC98592D83F9B6EA10B8D 206
    1067299212 1064287906 -1 application/octet-stream 6840/6840
    GET http://download.windowsupdate.com/msdownload/update/v3-19990518/cab...
1067299212.422 SWAPOUT 02 0005FD5F 6F34570785CACABC8DD01ABA5D73B392 200
    1067299210 1057899600 -1 image/gif 1125/1125
    GET http://forum.topsportsnet.com/shfimages/nav_members1.gif
1067299212.641 RELEASE -1 FFFFFFFF B0616CB4B7280F67672A40647DD08474 200
    1067299212 -1 -1 text/html -1/67191
    GET http://www.tlava.com/
1067299212.671 RELEASE -1 FFFFFFFF 5ECD93934257594825659B596D9444BC 200
    1067299023 1034873897 1067299023 image/jpeg 3386/3386
    GET http://ebiz0.ipixmedia.com/abc/ebiz/_EBIZ_3922eabf57d44e2a4c3e7cd234a...
1067299212.786 RELEASE -1 FFFFFFFF B388F7B766B307ADEC044A4099946A21 200
    1067297755 -1 -1 text/html -1/566
    GET http://www.evenflowrocks.com/pages/100303pic15.cfm
1067299212.837 RELEASE -1 FFFFFFFF ABC862C7107F3B7E9FC2D7CA01C8E6A1 304
    1067299212 -1 1067299212 unknown -1/0
    GET http://ebiz0.ipixmedia.com/abc/ebiz/_EBIZ_3922eabf57d44e2a4c3e7cd234a...
1067299212.859 RELEASE -1 FFFFFFFF 5ED2726D4A3AD83CACC8A01CFDD6082B 304
    1066940882 1065063803 -1 application/x-javascript -1/0
```

GET http://www.bellsouth.com/scripts/header_footer.js

每个日志条目包含如下 13 个域：

1. 时间戳

事件何时发生，表现为 Unix 纪元以来的秒数，它是毫秒级的。

2. 动作

cache 目标发生的动作。该域有 3 个可能值：SWAPOUT，RELEASE，和 SO_FAIL。

1）SWAPOUT 在 Squid 成功的存储目标到磁盘时发生。某些目标例如那些消极 cache 的，仅保存在内存而不是磁盘，Squid 不会在 store.log 里记录它们。

2）SO_FAIL 表明 Squid 不能完整的存储目标到磁盘。多半意味着存储机制拒绝以写方式打开新的磁盘文件。

3）RELEASE 在 Squid 从 cache 里删除目标，或首先就决定响应不可存储时发生。

3. 目录号

目录号是十进制小数形式，它是个到 cache 目录的 7 位索引。对没有存储到磁盘的目标，该域包含-1 值。

4. 文件号

文件号是 25 位的标识符，内在的被 squid 使用。它被写成 8 字符的十六进制号。对 UFS 基础的存储机制，有算法可以转换文件号到路径名（见 13.3.1 节）。

没有存储到磁盘的目标，没有有效的文件号。对这些目标，该域的值是 FFFFFFFF。仅仅在 RELEASE 和 SO_FAIL 情况下才会出现这个值。

5. cache 关键字

Squid 使用 MD5 哈希值作为主要的索引来定位目标。该关键字基于请求方式、URI、和其他可能的信息计算得来。

可以从 cache 关键字来查找 store.log 条目。然而请注意，目标的 cache 关键字可能改变。当 Squid 在 access.log 里记录 TCP_REFRESH_MISS 请求时，这点会发生。情况类似如下：

1065837334.045 SWAPOUT ... 554BACBD2CB2A0C38FF9BF4B2239A9E5 ... http://blah

1066031047.925 RELEASE ... 92AE17121926106EB12FA8054064CABA ... http://blah

1066031048.074 SWAPOUT ... 554BACBD2CB2A0C38FF9BF4B2239A9E5 ... http://blah

发生了什么呢？该目标原本 cache 在某个关键字下（554B...）。一段时间后，Squid 接受到对该目标的另一请求，并转发确认请求到原始服务器。当响应以新内容返回时，Squid 改变旧目标的 cache 关键字（92AE...），以便它能授予新目标正确的关键字（554B...）。然后旧目标删除，新目标存储到磁盘。

6. 状态码

该域显示响应的 HTTP 状态码，跟 access.log 一样。表 13.1 是状态码列表。

7. 日期

HTTP 响应的 Date 头部值，表现为 Unix 纪元以来的秒数。值-1 表示 Date 头部不可解析，-2 意味着头部完缺。

8. 最后修改时间

HTTP 响应的 Last-Modified 头部值，表现为 Unix 纪元以来的秒数。值-1 表示 Last-Modified 头部不可解析，-2 意味着头部完缺。

9. 过期时间

HTTP 响应的 Expires 头部值，表现为 Unix 纪元以来的秒数。值-1 表示 Expires 头部不可解析，-2 意味着头部完缺。

10. 内容类型

HTTP 响应的 Content-Type 头部值，排除了任何 media-type 参数。假如 Content-Type 丢失了，Squid 插入值 unknown。

11. 内容长度/大小

该域包含 2 个数字，以斜杠分开。第一个是 Content-Length 头部值。-1 表明 Content-Length 头部不存在。第二个是 HTTP 消息 body 的实际大小。你可使用这 2 个数字来部分的验证接受到的响应，并验证原始服务器是否不正确的计算了内容长度。大多数情形下，这 2 个数字相等。

12. 方式

请求目标的 HTTP 方式，跟 access.log 里的一样。

13. URI

最后一个域是请求 URI，跟 access.log 里的一样。该域也有前述章节提到的空格问题。然而，这里不必为此担忧，因为你可安全的忽略任何多余的域。

对许多 RELEASE 的条目，在最后 8 个域出现的是疑问号(?)。这是因为这些域的大部分值来自 squid 称为 MemObject 的结构。该结构仅在目标已被接受时，或目标被完整存储在内存时，才会出现。Squid cache 里的大部分目标没有 MemObject 结构，因为它们仅存在于磁盘。对这些情况，Squid 在相应域放置一个疑问号。

13.3.1 转换文件号到路径名

假如想要检查某个特定的 cache 文件，你可稍费工夫将文件号转换到路径名。另外目录号和 L1 和 L2 值也是必需的。在 squid 的源代码里，storeUfsDirFullPath( )函数做这个事情。可在 src/fs/ufs/store_dir_ufs.c 文件里找到它。如下短小的 perl 脚本模拟了当前算法：

```
#!/usr/bin/perl

$L1 = 16;

$L2 = 256;

while (<>) {

    $filn = hex($_);

    printf("%02X/%02X/%08X\n",

        (($filn / $L2) / $L2) % $L1,

        ($filn / $L2) % $L2,

        $filn);

}
```

这样使用它：

```
% echo 000DCD06 | ./fileno-to-pathname.pl

0D/CD/000DCD06
```

要在第 N 个 cache_dir 里找到该文件，简单的进入到相应的目录，并列出或查看该文件：

% cd /cache2

% ls -l 0D/CD/000DCD06

-rw------- 1 squid squid 391 Jun 3 12:40 0D/CD/000DCD06

% less 0D/CD/000DCD06

13.4 referer.log

可选的 referer.log 包含了来自客户端请求的 Referer 头部。为了使用该功能，必须在./configure 时打开--enable-referer-log 选项。还必须用 referer_log 指令来指定一个路径。例如：

referer_log /usr/local/squid/var/logs/referer.log

假如想禁止 referer.log，则可设置文件名为 none。

Referer 头部正常情况下包含一个 URI，从这个 URI 获取到了请求（见 RFC2616 的 14.36 节）。例如，当 web 浏览器发布请求到某个内嵌图片时，Referer 头部被设置成包含该图片的 HTML 网页的 URI。当你点击 HTML 超链接时，它也被设置。某些 web 站点管理员使用 Referer 值来查找死链接。在使用 Squid 作为代理人模式时，你也许发现 referer.log 特别有用。

referer.log 格式简单，仅有 4 个域。如下是一些示例：

1068047502.377 3.0.168.206

http://www.amazon.com/exec/obidos/search-handle-form/002-7230223-8205634

http://www.amazon.com/exec/obidos/ASIN/0596001622/qid=1068047396/sr=2-1/...

1068047503.109 3.0.168.206

http://www.amazon.com/exec/obidos/ASIN/0596001622/qid=1068047396/sr=2-1/...

http://g-images.amazon.com/images/G/01/gourmet/gourmet-segway.gif

1068047503.196 3.0.168.206

http://www.amazon.com/exec/obidos/ASIN/0596001622/qid=1068047396/sr=2-1/...

http://g-images.amazon.com/images/G/01/marketing/cross-shop/arnold/appar...

1068047503.198 3.0.168.206

http://www.amazon.com/exec/obidos/ASIN/0596001622/qid=1068047396/sr=2-1/...

http://g-images.amazon.com/images/G/01/marketing/cross-shop/arnold/appar...

1068047503.825 3.0.168.206

http://www.amazon.com/exec/obidos/ASIN/0596001622/qid=1068047396/sr=2-1/...

http://images.amazon.com/images/P/B00005R8BC.01.TZZZZZZZ.jpg

1068047503.842 3.0.168.206

http://www.amazon.com/exec/obidos/ASIN/0596001622/qid=1068047396/sr=2-1/...

http://images.amazon.com/images/P/0596001622.01._PE_PI_SCMZZZZZZZ_.jpg

注意缺少 Referer 头部的请求不会被记录。这 4 个域描述如下：

1. 时间戳

请求时间，表现为 Unix 纪元以来的秒数，是毫秒级的。

注意的是，不像 access.log，referer.log 在 Squid 接受到完整请求时，会立刻记录。这样，referer.log 条目在 access.log 之前发生，后者等待响应完成才记录。

2. 客户端地址

客户端地址跟 access.log 里的一样。log_fqdn 和 client_netmask 指令也影响该日志文件。

3. referer

来自客户端请求的 Referer 头部值。注意 referer 值可能有空格字符或其他字符，在写 referer.log 前 Squid 不会对其进行编码。

4. URI

客户端正请求的 URI。它匹配 access.log 里的 URI。

13.5 useragent.log

可选的 useragent.log 包含来自客户端请求的 User-Agent 头部值。为了使用该功能，必须在运行./configure 时打开--enable-useragent-log 选项。还必须使用 useragent_log 指令来提供一个路径名。例如：

useragent_log /usr/local/squid/var/logs/useragent.log

User-Agent 头部正常情况下包含了发起请求的 user-agent 的描述。大多数情形下，该描述只是简单的产品名列表和版本信息。你应该清楚应用程序可以轻易的提供伪造的 user-agent 信息。现代 user-agent 提供途径可定制该描述。甚至 Squid 在转发请求里能改变这个 User-Agent 头部。

useragent.log 格式相对简单，看起来如下：

3.0.168.206 [05/Nov/2003:08:51:43 -0700]

"Mozilla/5.0 (compatible; Konqueror/3; FreeBSD)"

3.0.168.207 [05/Nov/2003:08:52:18 -0700]

"Opera/7.21 (X11; FreeBSD i386; U) [en]"

4.241.144.204 [05/Nov/2003:08:55:11 -0700]

"Mozilla/5.0 (Macintosh; U; PPC Mac OS X; en-us) AppleWebKit/103u (KHTM..."

3.0.168.206 [05/Nov/2003:08:51:43 -0700]

"Java1.3.1_01"

64.68.82.28 [05/Nov/2003:08:52:50 -0700]

"Googlebot/2.1 (http://www.googlebot.com/bot.html)"

3.0.168.205 [05/Nov/2003:08:52:50 -0700]

"WebZIP/4.1 (http://www.spidersoft.com)"

4.241.144.201 [05/Nov/2003:08:52:50 -0700]

"Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt; Hotbar 3.0)"

3.0.168.206 [05/Nov/2003:08:54:40 -0700]

"Bookmark Renewal Check Agent [http://www.bookmark.ne.jp/] (Version 2.0..."

不像其他日志文件，它仅有 3 个域：

1. 客户端地址

跟 access.log 里的一样。log_fqdn 和 client_netmask 指令也影响该日志文件。

2. 时间戳

不像其他日志文件那样，用 Unix 纪元以来的秒数来描述时间，这里使用人工可读的格式。它是 HTTP 通用日志文件格式的时间戳，看起来如下：

[10/Jun/2003:22:38:36 -0600]

注意方括号界定时间戳，它包含一个空格。也请注意，跟 referer.log 一样，这些条目在 Squid 接受到完整请求时，立刻被记录。

3. user-agent

User-Agent 头部的值。这些字串几乎总包含空格。在将其写入日志文件时，Squid 不会编码 User-Agent 值。

13.6 swap.state

swap.state 文件是目标写入 cache 目录、或从 cache 目录删除的日志写照。每个 cache_dir 有它自己的 swap.state 文件。当 Squid 启动时，它读取 swap.state 文件来重建 cache 目标的内存索引。这些文件对 Squid 管理来说，至关重要。

默认情况下，每个 cache.state 文件位于它相应的 cache 目录。这样，每个 state 文件自动驻留在每个 cache_dir 下。这点很有用——假如你想重新排序 cache_dir 行，或想从 cache_dir 列表里删除条目的话。

如果想将它们放在其他位置，可使用 cache_swap_log 指令来做：

cache_swap_log /usr/local/squid/var/logs/swap.state

在此情况下，Squid 对每个 cache 目录创建一个 swap.state 文件，并增加数字后缀。例如，假如有 4 个 cache 目录，Squid 创建如下日志：

/usr/local/squid/var/logs/swap.state.00

/usr/local/squid/var/logs/swap.state.01

/usr/local/squid/var/logs/swap.state.02

/usr/local/squid/var/logs/swap.state.03

在这个情形下，如果你要增加、删除、或重排序 cache_dir 行，就必须手工重命名 swap.state 文件，以保持事情一致。

技术上，swap.state 格式是独立于存储机制的。然而，在当前版本的 Squid 里，所有的存储机制使用同一种格式。swap.state 文件使用修正大小（48 位）的二进制格式。各个域值以主机字节顺序记录，这样在不同的操作系统之间不便迁移。表 13-3 描述了 swap.state 日志条目的各个域的说明。

Table 13-3. swap.state entry fields

Name	Size, in bytes	Description
op	1	Operation on the entry: added (1) or deleted (2).
file number	4	Same as the fourth field of *store.log*, except it is stored in binary.
timestamp	4	A timestamp corresponding to the time when the response was generated or last validated. Taken from the `Date` header for responses that have one. Stored as the number of seconds since the Unix epoch.
lastref	4	A timestamp corresponding to the most recent access to the object.
expires	4	The object's expiration time, taken from an `Expires` header or `Cache-Control max-age` directive.
last-modified	4	The object's `Last-Modified` value.
swap file size	4	The amount of space the object occupies on disk. This includes HTTP headers and other Squid-specific meta-information.
refcount	2	The number of times this object has been requested.
flags	2	Various internal flags used by Squid.
key	16	The MD5 hash of the corresponding URI. Same as the key in *store.log*, except this one is stored in binary.

13.7 轮转日志

Squid 不断的写日志，假如 cache 非常忙，那么在一段时间后，这些日志文件可能变得很大。某些操作系统甚至限制了文件的最大 size（例如 2G），假如写文件超过了这个 size 就会报错。为了保持日志文件容易管理，以及让 Squid 正常工作，必须定期轮转日志。

Squid 有内建的功能用于轮转日志。可通过 squid -k rotate 命令来调用它，然后告诉 Squid 对每个日志文件保持多少份旧拷贝。例如，假如设置它为 7，对每个日志文件会有 8 个版本：1 个当前的，和 7 个旧的。

旧日志文件以数字扩展来重命名。例如，当执行一次轮转时，Squid 重命名 log.6 到 log.7，然后是 log.5 到 log.6，依此类推。当前 log 变成 log.0，并且 Squid 创建一个新的空文件，命名为 log。

每次执行 squid -k rotate 时，Squid 轮转下述文件：cache.log, access.log, store.log, useragent.log (假如激活), 以及 referer.log (假如激活)。Squid 也会创建最新版本的 swap.state 文件。然而请注意，swap.state 不会以数字扩展形式来轮转。

Squid 不会自己轮转日志，最好的办法是在 crontab 里自动执行。例如：

```
0 0 * * * /usr/local/squid/sbin/squid -k rotate
```

假如你想编写自己的脚本来管理日志文件，Squid 提供了一个有用的模式，简单的设置 logfile_rotate 指令为 0。这样，当你运行 squid -k rotate 命令时，Squid 简单的关闭当前日志文件，并且打开新的。如果操作系统允许重命名被其他进程打开的文件，则这点非常有用。下述 shell 脚本描述了一个思路：

```
#!/bin/sh

set -e

yesterday_secs=`perl -e 'print time -43200'`

yesterday_date=`date -r $yesterday_secs +%Y%m%d`

cd /usr/local/squid/var/logs

# rename the current log file without interrupting the logging process

mv access.log access.log.$yesterday_date

# tell Squid to close the current logs and open new ones

/usr/local/squid/sbin/squid -k rotate
```

```
# give Squid some time to finish writing swap.state files

sleep 60

mv access.log.$yesterday_date /archive/location/

gzip -9 /archive/location/access.log.$yesterday_date
```

13.8 隐私和安全

Squid 的日志文件特别是 access.log，包含了用户的活跃记录，因此它受隐私问题支配。作为 Squid 管理员，你必须采取额外的小心来保证日志文件安全。最好的办法是限制访问 Squid 主机的人员的数量。假如这点行不通，那么就要谨慎的检查文件和目录许可，确保日志文件不会被非信任的、或未授权的用户访问。

也可利用 client_netmask 和 strip_query_terms 指令来保护用户隐私。前者让识别 access.log 里的用户困难；后者移除了 URI 查询条件以避免泄露用户私人信息。更多信息见 13.2.4 节。

如果想要保持历史数据相当长的时间，你也许可裁减日志来保证日志文件匿名。假如你仅对哪个 URI 被访问感兴趣，而不是谁访问了它们，就可从 access.log 里抽取出该域。这样也让文件更小，并且减少了隐私违背的风险。另一个技术是随机处理客户端 IP 地址。换句话说，就是创建一个过滤器，把真正的 IP 地址映射到假的地址，前提是同一个真 IP 地址总是映射到同一个假 IP。假如你在使用 RFC 1413 身份验证协议或 HTTP 认证，也可考虑保持这些域匿名。