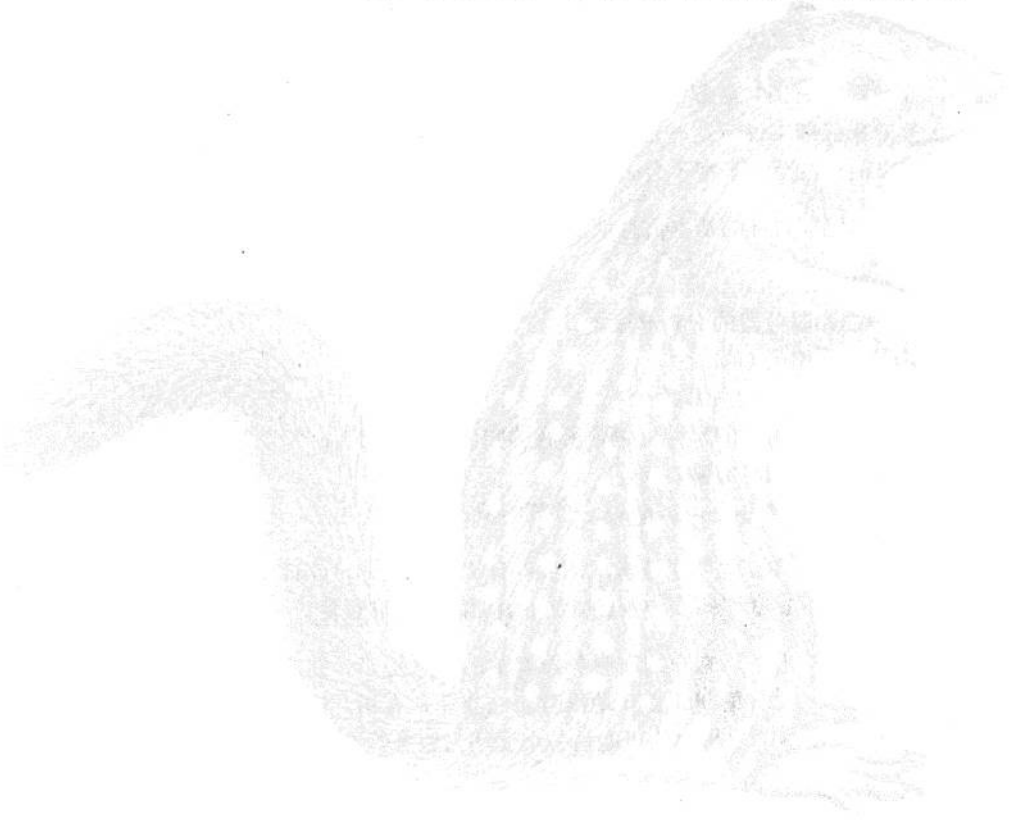
第21章

日志记录与使用情况跟踪



几乎所有的服务器和代理都会记录下它们所处理的 HTTP 事务摘要。这么做出于一系列的原因：跟踪使用情况、安全性、计费、错误检测，等等。本章简要介绍了日志记录，研究了通常会记录 HTTP 事务哪些方面的信息以及一些常见日志格式中所包含的内容。

21.1 记录内容

大多数情况下，日志的记录出于两种原因：查找服务器或代理中存在的问题（比如，哪些请求失败了），或者是生成 Web 站点访问方式的统计信息。统计数据对市场营销、计费和容量规划（比如，决定是否需要增加服务器或带宽）都非常有用。

可以把一个 HTTP 事务中所有的首部都记录下来，但对每天要处理数百万个事务的服务器和代理来说，这些数据的体积超大，很快就会失控。不应该记录实际上你并不感兴趣，甚至从来都不会去看一眼的数据。

通常，只记录事务的基本信息就行了。通常会记录下来的几个字段示例为：

- HTTP 方法；
- 客户端和服务器的 HTTP 版本；
- 所请求资源的 URL；
- 响应的 HTTP 状态码；
- 请求和响应报文的尺寸（包含所有的实体主体部分）；
- 事务开始时的时间戳；
- `Referer` 首部和 `User-Agent` 首部的值。

483

HTTP 方法和 URL 说明了请求试图做些什么——比如，GET 某个资源或 POST 某个定单。可以用 URL 来记录 Web 站点上页面的受欢迎程度。

版本字符串给出了与客户端和服务器有关的一些提示，在客户端和服务器之间出现一些比较奇怪或非预期的交互动作时，它会非常有用。比如，如果请求的失败率高于预期，那版本信息指向的可能是一个无法与服务器进行交互的新版浏览器。

HTTP 状态码说明了请求的执行状况：是否成功执行，认证请求是否失败，资源是否找到等（HTTP 状态码列表参见 3.2.2 节）。

请求 / 响应的大小和时间戳主要用于记账；就是记录流入、流出或流经应用程序的字节有多少。还可用时间戳将观察到的问题与当时发起的一些请求关联起来。

21.2 日志格式

目前已有一些日志格式的标准了。本节我们会讨论一些最常见的日志格式。大部分商用和开源的 HTTP 应用程序都支持以一种或多种常用格式进行日志记录。很多这样的应用程序都支持管理者配置日志格式，创建自定义的格式。

应用程序支持管理者使用这些更标准的格式的主要好处之一就在于，可以充分利用那些已构建好的工具处理这些日志，并产生基本的统计信息。有很多开源包和商用包都可用来压缩日志，以进行汇报。使用标准格式，应用程序及其管理员就都可以利用这些包了。

21.2.1 常见日志格式

现在，最常见的日志格式之一就是常用日志格式。这种日志格式最初由 NCSA 定义，很多服务器在默认情况下都会使用这种日志格式。可以将大部分商用及开源服务器配置为使用这种格式，有很多商用及免费工具都可辅助解析常用日志格式的文件。表 21-1 按序列出了常用日志格式中的字段。

表21-1 常用日志格式字段

字 段	描 述
remotehost	请求端机器的主机名或 IP 地址（如果没有配置服务器去执行反向 DNS 或无法查找请求端的主机名，就使用 IP 地址）
username	如果执行了 ident 查找，就是请求端已认证的用户名[a]
auth-username	如果进行了认证，就是请求端已认证的用户名
timestamp	请求的日期和时间
request-line	精确的 HTTP 请求行文本，GET /index.html HTTP/1.1
response-code	响应中返回的 HTTP 状态码
response-size	响应主体中的 Content-Length，如果响应中没有返回主体，就记录 0

a：RFC 931 描述了在此认证中使用的 ident 查找。ident 协议是在第 5 章介绍的。

484

例 21-1 列出了几个常见日志格式条目。

例 21-1 常见日志格式

```
209.1.32.44 - - [03/Oct/1999:14:16:00 -0400] "GET / HTTP/1.0" 200 1024
http-guide.com - dg [03/Oct/1999:14:16:32 -0400] "GET / HTTP/1.0" 200 477
http-guide.com - dg [03/Oct/1999:14:16:32 -0400] "GET /foo HTTP/1.0" 404 0
```

在这些例子中，字段的分配如下所示。

字段1	条目1	条目2	条目2
remotehost	209.1.32.44	http-guide.com	http-guide.com
username	<空>	<空>	<空>
auth-username	<空>	dg	dg
timestamp	03/Oct/1999:14:16:00 -0400	03/Oct/1999:14:16:32 -0400	03/Oct/1999:14:16:32 -0400
request-line	GET / HTTP/1.0	GET / HTTP/1.0	GET /foo HTTP/1.0
response-code	200	200	404
response-size	1024	477	0

注意，remotehost 字段可以是 http-guide.com 那样的主机名，也可以是 209.1.32.44 这样的 IP 地址。

第二个（username）和第三个（auth-username）字段之间的破折号说明字段为空。这说明要么是没有进行 ident 查找（第二个字段为空），要么是没有进行认证（第三个字段为空）。

21.2.2 组合日志格式

另一种常用日志格式为组合日志格式（Combined Log Format），例如 Apache 服务器就支持这种格式。组合日志格式与常用日志格式很类似。实际上，它就是常用日志格式的精确镜像，只是添加了（表 21-2 中列出的）两个字段。User-Agent 字段用于说明是哪个 HTTP 客户端应用程序在发起已被记录的请求，而 Referer 字段则提供了更多与请求端在何处找到这个 URL 的有关信息。

485

表21-2 新加的组合日志格式字段

字 段	描 述
Referer	Referer 首部的内容
User-Agent	User-Agent 首部的内容

例 21-2 给出了一个组合日志格式的条目。

例 21-2 组合日志格式

```
209.1.32.44 - - [03/Oct/1999:14:16:00 -0400] "GET / HTTP/1.0" 200 1024
"http://www.joes-hardware.com/"  "5.0: Mozilla/4.0 (compatible; MSIE
5.0; Windows 98)"
```

在例 21-2 中，Referer 字段和 User-Agent 字段的值如下所示。

字 段	值
Referer	http://www.joes-hardware.com/
User-Agent	5.0: Mozilla/4.0（兼容的，MSIE 5.0，Windows 98）

例 21-2 的组合日志格式条目示例中的前七个字段和常用日志格式中的完全一样（参见例 21-1 中的第一个条目）。两个新字段 Referer 和 User-Agent 附加在日志条目的末尾。

21.2.3 网景扩展日志格式

网景进入商用 HTTP 应用程序领域时，为其服务器定义了很多其他 HTTP 应用程序开发者已接纳的日志格式。网景的格式是基于 NCSA 的常用日志格式的，但它们扩展了该格式，以支持与代理和 Web 缓存这样的 HTTP 应用程序相关的字段。

网景扩展日志格式的前 7 个字段与常用日志格式中的那些字段完全相同（参见表 21-1）。表 21-3 按序列出了网景扩展日志格式引入的新字段。

表21-3 网景扩展日志格式新加的字段

字 段	描 述
proxy-response-code	如果事务处理经过了某个代理，就是从服务器传往代理的 HTTP 响应码
proxy-response-size	如果事务处理经过了某个代理，就是发送给代理的服务器响应实体的 Content-Length
client-request-size	发给代理的客户端请求的所有主体或实体的 Content-Length
proxy-request-size	如果事务处理经过了某个代理，就是代理发往服务器的请求的所有主体或实体的 Content-Length
client-request-hdr-size	以字节为单位的客户端请求首部的长度
proxy-response-hdr-size	如果事务处理经过了某个代理，就是以字节为单位的、发送给请求端的代理响应首部的长度
proxy-request-hdr-size	如果事务处理经过了某个代理，就是以字节为单位的、发送给服务器的代理请求首部的长度
server-response-hdr-size	以字节为单位的，服务器响应首部的长度
proxy-timestamp	如果事务处理经过了某个代理，就是请求和响应经过代理传输所经过的时间（单位为秒）

486

例 21-3 给出了一个网景扩展日志格式的条目。

例 21-3 网景扩展日志格式

```
209.1.32.44 - - [03/Oct/1999:14:16:00-0400] "GET / HTTP/1.0" 200 1024
200 1024 0 0 215 260
279 254 3
```

在这个例子中，扩展字段的值如下所示。

字　段	值
proxy-response-code	200
proxy-response-size	1024
client-request-size	0
proxy-request-size	0
client-request-hdr-size	215
proxy-response-hdr-size	260
proxy-request-hdr-size	279
server-response-hdr-size	254
proxy-timestamp	3

例 21-3 中网景扩展日志格式条目示例中的前 7 个字段是常用日志格式条目示例的镜像（参见例 21-1 中的第一个条目）。

21.2.4 网景扩展2日志格式

另一种网景日志格式，网景扩展 2 日志格式采用了扩展日志格式，并添加了一些与 HTTP 代理和 Web 缓存应用程序有关的附加信息。这些附加字段有助于更好地描绘 HTTP 客户端和 HTTP 代理应用程序间的交互图景。

网景扩展 2 日志格式是基于网景扩展日志格式的，初始字段与表 21-3 中列出的字段
487 完全相同（它也是表 21-1 中常用日志格式字段的扩展）。

表 21-4 按序列出了网景扩展 2 日志格式新加的字段。

表21-4　附加的网景扩展2日志格式字段

字　段	描　述
route	代理用来向客户端发送请求的路径（参见表 21-5）
client-finish-status-code	客户端完成状态码。说明了发送给代理的客户端请求是成功完成（FIN）了，还是被打断了（INTR）
proxy-finish-status-code	代理完成状态码。说明代理发送给服务器的请求是成功完成（FIN）了，还是被打断了（INTR）
cache-result-code	缓存结果代码，说明缓存是如何响应请求的[a]

a：表 21-7 列出了网景的缓存结果代码。

例 21-4 给出了一个网景扩展 2 日志格式的条目。

例 21-4　网景扩展 2 日志格式

```
209.1.32.44 - - [03/Oct/1999:14:16:00-0400] "GET / HTTP/1.0" 200 1024
200 1024 0 0 215 260
```

```
279 254 3 DIRECT FIN FIN WRITTEN
```

这个例子中扩展字段的值如下所示。

字　段	值
route	DIRECT
client-finish-status-code	FIN
proxy-finish-status-code	FIN
cache-result-code	WRITTEN

例 21-4 的网景扩展 2 日志格式条目中的前 16 个字段就是网景扩展日志格式示例条目的镜像（参见例 21-3）。

表 21-5 列出了有效的网景路由代码。

表21-5　网景路由代码

值	描　述
DIRECT	资源是直接从服务器上获取的
PROXY(host:port)	资源是通过代理“host:port”获取的
SOCKS(socks:port)	资源是通过 SOCKS 服务器“host:port”获取的

表 21-6 列出了有效的网景完成代码。

表21-6　网景完成状态码

值	描　述
-	请求未开始
FIN	请求成功完成
INTR	请求被客户端中断或被代理 / 服务器终止
TIMEOUT	请求被代理 / 服务器超时

表 21-7 列出了有效的网景缓存代码。[1]

表21-7　网景缓存代码

代　码	描　述
-	资源不可缓存
WRITTEN	资源被写入了缓存
REFRESHED	资源被缓存，且被刷新了
NO-CHECK	返回已缓存的资源，未进行新鲜性检查

注 1：表 21-7 列出了网景缓存结果代码。

（续）

代　码	描　述
UP-TO-DATE	返回已缓存的资源，进行了新鲜性检查
HOST-NOT-AVAILABLE	返回已缓存的资源。由于远程服务器不可用，所以未进行新鲜性检查
CL-MISMATCH	未将资源写入缓存。由于 Content-Length 与资源尺寸不匹配，放弃了写操作
ERROR	因为出错，资源未被写入缓存。比如，出现了超时，或客户端放弃了此事务

与很多其他 HTTP 应用程序一样，网景应用程序也有其他的日志格式，包括一种灵活日志格式和一种管理者输出自定义日志字段的方式。这些格式给予管理者更大的控制权，并可以选择在日志中报告 HTTP 事务处理的哪些部分（首部、状态、尺寸等），以自定义其日志。

由于很难预测管理者希望从其日志中获取哪些信息，才添加了管理者配置自定义格式的能力。很多其他的代理和服务器都有发布自定义日志的能力。

21.2.5　Squid代理日志格式

Squid 代理缓存（http://www.squid-cache.org）是 Web 上一个很古老的部分。其起源可以回溯到一个早期的 Web 代理缓存项目（ftp://ftp.cs.colorado.edu/pub/techreports/schwartz/Harvest.Conf.ps.Z）。Squid 是开源社团多年来扩展增强的一个开源项目。有很多工具可以用来辅助管理 Squid 应用程序，包括一些有助于处理、审核及开发其日志的工具。很多后继代理缓存都为自己的日志使用了 Squid 格式，这样才能更
489 好地利用这些工具。

Squid 日志条目的格式相当简单。表 21-8 总结了该日志格式的字段。

表21-8　Squid日志格式字段

字　段	描　述
timestamp	请求到达时的时间戳，是从格林尼治标准时间 1970 年 1 月 1 日开始的秒数
time-elapsed	请求和响应通过代理传输所经历的时间（以毫秒为单位）
host-ip	客户端（请求端）主机的 IP 地址
result-code/status	result 字段是 Squid 类型的，用来说明在此请求过程中代理采取了什么动作；code 字段是代理发送给客户端的 HTTP 响应代码
size	代理响应客户端的字节长度，包括 HTTP 响应首部和主体
method	客户端请求的 HTTP 方法

a：表 21-9 列出了各种结果代码及其含义。

（续）

字　段	描　述
url	客户端请求中的 URL[b]
rfc931-ident[c]	客户端经过认证的用户名[d]
hierarchy/from	与网景格式中的 route 字段一样，hierarchy 字段说明了代理向客户端发送请求时经由的路径。[e]from 字段说明了代理用来发起请求的服务器的名称
content-type	代理响应实体的 Content-Type

b：回顾第 2 章，代理通常会记录下整条请求 URL，这样如果 URL 中有用户和密码组件，代理就会在不经意间记录下此信息。

c：rfc931-ident、hierarchy/from 和 content-type 字段是在 Squid 1.1 中添加的。早期版本中都没有这些字段。

d：RFC 931 描述了这种认证方式中使用的 ident 查找。

e：http://squid.nlanr.net/Doc/FAQ/FAQ-6.html#ss6.6 列出了所有有效的 Squid hierarchy 代码。

例 21-5 给出了一个 Squid 日志格式条目的例子。

例 21-5　Squid 日志格式

```
99823414 3001 209.1.32.44 TCP_MISS/200 4087 GET http://www.joes-
hardware.com - DIRECT/
proxy.com text/html
```

这些字段的值如下所示。

字　段	值
timestamp	99823414
time-elapsed	3001
host-ip	209.1.32.44
action-code	TCP_MISS
status	200
size	4087
method	GET
URL	http://www.joes-hardware.com
RFC 931 ident	-
hierarchy	DIRECT[a]
from	proxy.com
content-type	text/html

a：DIRECT Squid hierarchy 值与网景日志格式中的 DIRECT route 值一样。

表 21-9 列出了各种 Squid 结果代码。[2]

注 2：这些行为代码中有几个通常是处理 Squid 代理缓存的内部行为，因此其他使用了 Squid 日志格式的代理并没有使用全部的代码。

表21-9 Squid结果代码

行　为	描　述
TCP_HIT	资源的有效副本是由缓存提供的
TCP_MISS	资源不在缓存中
TCP_REFRESH_HIT	资源在缓存中，但需要进行新鲜性检查。代理与服务器再次验证了资源，发现缓存中的副本确实还是新鲜的
TCP_REF_FAIL_HIT	资源在缓存中，但需要进行新鲜性检查。但再验证失败了（可能是代理无法连接到服务器），因此返回的是“过期”的资源
TCP_REFRESH_MISS	资源在缓存中，但需要进行新鲜性检查。在与服务器进行验证的时候，代理得知缓存中的资源过期了，并收到一个新的副本
TCP_CLIENT_REFRESH_MISS	请求端发送了一个 `Pragma: no-cache`，或类似的 `Cache-Control` 指令，命令代理必须去获取资源
TCP_IMS_HIT	请求端发布了一个条件请求，对资源的已缓存副本进行验证
TCP_SWAPFAIL_MISS	代理认为资源位于缓存中，但出于某些原因无法访问该资源
TCP_NEGATIVE_HIT	返回已缓存的响应，但响应是否定的已缓存响应。Squid 支持对资源错误信息的缓存（比如，缓存 404 Not Found 响应）。这样，如果有多条对某无效资源的请求都经过这个代理缓存，就可以由这个代理缓存提供错误信息
TCP_MEM_HIT	资源的有效副本是由缓存提供的，资源位于代理缓存的内存中（与必须访问磁盘才能获取已缓存资源的方式相反）
TCP_DENIED	对此资源的请求被否决了，可能是请求端没有请求此资源的权利
TCP_OFFLINE_HIT	所请求的资源是在离线状态下从缓存中解析出来的。Squid（或另一个使用此格式的代理）处于离线模式时，资源是未经验证的
UDP_*	UDP_* 代码说明请求是通过到代理的 UDP 接口收到的。HTTP 通常会使用 TCP 传输协议，因此这些请求使用的都不是 HTTP 协议[a]
UDP_HIT	资源的有效副本是由缓存提供的
UDP_MISS	资源不在缓存中
UDP_DENIED	对此资源的请求被否决了，可能是由于请求端没有请求此资源的权限
UDP_INVALID	代理收到的请求是无效的
UDP_MISS_NOFETCH	Squid 在特定的操作模式下，或在缓存常见错误的缓存中使用。会返回缓存未命中，而且也没有获得资源
NONE	有时与错误信息一起记录
TCP_CLIENT_REFRESH	参见 TCP_CLIENT_REFRESH_MISS
TCP_SWAPFAIL	参见 TCP_SWAPFAIL_MISS
UDP_RELOADING	参见 UDP_MISS_NOFETCH

a：Squid 有自己的用于发起这些请求的协议：ICP。这是缓存到缓存的请求所使用的协议。更多信息请参见 http://www.squid-cache.org。

491

21.3 命中率测量

原始服务器通常会出于计费的目的保留详细的日志记录。内容提供者需要知道URL的受访频率，广告商需要知道广告的出现频率，网站作者需要知道所编写的内容的受欢迎程度。客户端直接访问Web服务器时，日志记录可以很好地跟踪这些信息。

但是，缓存服务器位于客户端和服务器之间，用于防止服务器同时处理大量访问请求[3]（这正是缓存的目的）。缓存要处理很多HTTP请求，并在不访问原始服务器的情况下满足它们的请求，服务器中没有客户端访问其内容的记录，导致日志文件中出现遗漏。

由于日志数据会遗失，所以，内容提供者会对其最重要的页面进行*缓存清除*（cache bust）。缓存清除是指内容提供者有意将某些内容设置为无法缓存，这样，所有对此内容的请求都会被导向原始服务器。[4]于是，原始服务器就可以记录下访问情况了。不使用缓存可能会生成更好的日志，但会减缓原始服务器和网络的请求速度，并增加其负荷。

由于代理缓存（及一些客户端）都会保留自己的日志，所以如果服务器能够访问这些日志（或者至少有一种粗略的方式可以判断代理缓存会以怎样的频率提供其内容），就可以避免使用缓存清除。命中率测量协议是对HTTP的一种扩展，它为这个问题提供了一种解决方案。命中率测量协议要求缓存周期性地向原始服务器汇报缓存访问的统计数据。

492

RFC 2227详细定义了命中率测量协议。本节将详细介绍此协议。

21.3.1 概述

命中率测量协议定义了一种HTTP扩展，它提供了一些基本的功能，缓存和服务器可以实现这些功能来共享访问信息，规范已缓存资源的可使用次数。

缓存给日志访问带来了问题，命中率测量并不是这个问题的完整解决方案，但它确实提供了一种基本方式，以获取服务器希望跟踪的度量值。命中率测量协议并没有（而且可能永远都不会）得到广泛的实现或应用。也就是说，在维护缓存性能增益的同时，像命中率测量这样的合作方案会给出一些提供精确访问统计信息的承诺。希望这会推动命中率测量协议的实现，而不是把内容标记为不可缓存的。

注3：回想一下，几乎每个浏览器都会有一个缓存。

注4：第7章说明了怎样将HTTP响应标记为不可缓存的。

21.3.2 Meter首部

命中率测量扩展建议使用新增加的首部 Meter，缓存和服务器可以通过它在相互间传输与用法和报告有关的指令，这与用来进行缓存指令交换的 Cache-Control 首部很类似。

表 21-10 列出了定义的各种指令和谁可以在 Meter 首部传输这些指令。

表21-10 命中率测量指令

指 令	缩 写	执 行 者	描 述
will-report-and-limit	w	缓存	缓存可以报告使用情况并遵循服务器指定的所有使用限制
wont-report	x	缓存	缓存可以遵循使用限制，但不报告使用情况
wont-limit	y	缓存	缓存可以报告使用情况但不会限制使用
count	c	缓存	报告指令，以 uses/reuses 整数的形式说明。比如：count=2/4 [a]
max-uses	u	服务器	允许服务器指定某响应可被缓存使用的最大次数。比如：max-uses=100
max-reuses	r	服务器	允许服务器指定某响应可被缓存重用的最大次数。比如：max-reuses=100
do-report	d	服务器	服务器要求代理发送使用报告
dont-report	e	服务器	服务器不要求使用汇报
timeout	t	服务器	允许服务器指定对某资源进行计量的超时时间。缓存应该在指定的超时时间，或在此时间之前发送报告，允许有 1 分钟的误差。超时是以分钟为单位的。比如：timeout=60
wont-ask	n	服务器	服务器不需要任何计量信息

a：命中率测量定义了一个 use，用一个响应来满足请求，还定义了一个 reuse，对客户端请求进行再验证。

493

图 21-1 给出了一个执行中的命中率测量实例。事务的第一部分就是客户端和代理缓存之间一个普通的 HTTP 事务，但在代理请求中，要注意有插入的 Meter 首部和来自服务器的响应。这里，代理正在通知服务器它可以进行命中率测量，作为回应，服务器则请求代理报告它的命中次数。

从客户端的角度来看，请求正常结束了，代理开始代表服务器跟踪该请求资源的命中次数。稍后，代理会尝试与服务器再次验证资源。代理会在发送给服务器的条件请求中嵌入它跟踪记录的计量信息。

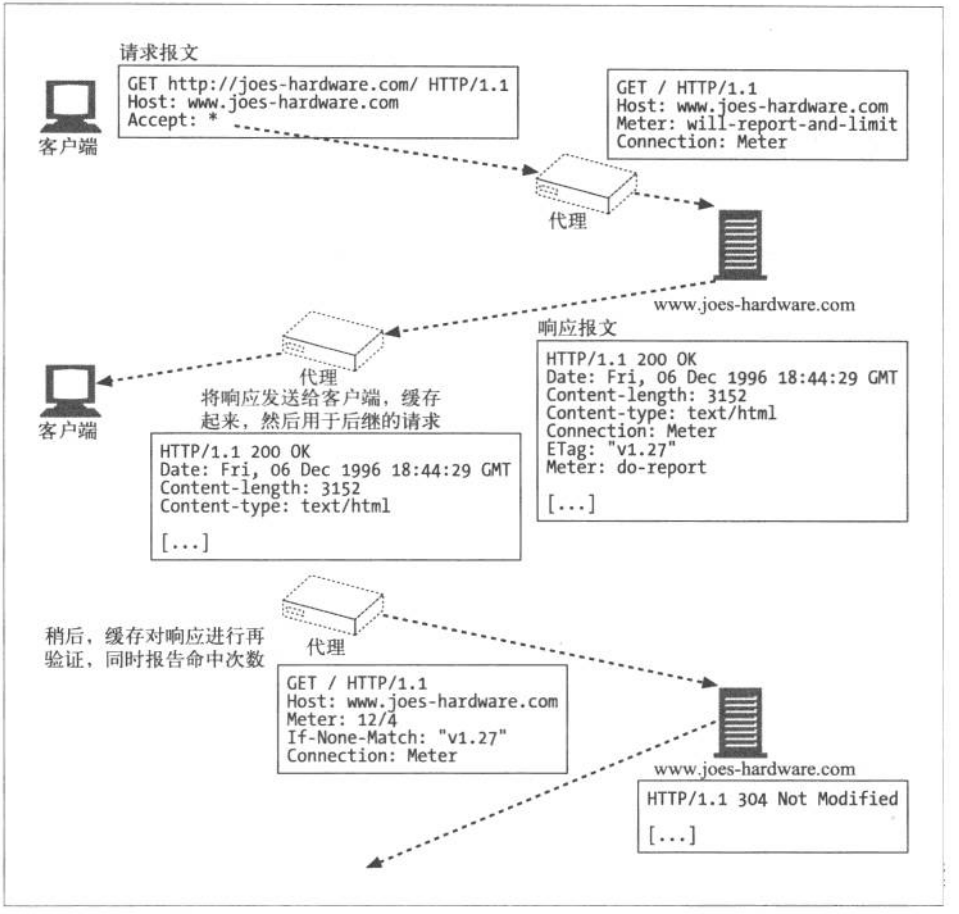


图 21-1　命中率测量示例

494

21.4　关于隐私的考虑

日志记录实际上就是服务器和代理执行的一项管理功能，所以整个操作对用户来说都是透明的。通常，用户甚至都不清楚他们的 HTTP 事务已被记录——实际上，很多用户可能甚至都不知道他们访问 Web 上的内容时是在使用 HTTP 协议。

Web 应用程序的开发者和管理者要清楚跟踪用户的 HTTP 事务可能带来的影响。他可以根据获取的信息收集很多有关用户的情况。很显然，这些信息可以用于不良目的——歧视、骚扰、勒索等。进行日志记录的 Web 服务器和代理一定要注意保护其终端用户的隐私。

有些情况下，比如在工作环境中，跟踪某用户的使用情况以确保他没有偷懒是可行的，但管理员也应该将监视大家事务处理的事情公之于众。

简单来说，日志记录对管理者和开发者来说都是很有用的工具。只是要清楚在没有获得用户许可，或在其不知情的情况下，使用记录其行为的日志可能会存在侵犯隐私的问题。

21.5 更多信息

更多有关日志记录的信息，请参见以下资源。

- http://httpd.apache.org/docs/logs.html
  “Apache HTTP Server: Log Files”（“Apache HTTP 服务器：日志文件”）。Apache HTTP 服务器项目网站。
- http://www.squid-cache.org/Doc/FAQ/FAQ-6.html
  “Squid Log Files”（“Squid 日志文件”）。Squid 代理缓存网站。
- http://www.w3.org/Daemon/User/Config/Logging.html#common-logfile-format
  “Logging Control in W3C httpd”（“W3C httpd 中的日志记录控制”）。
- http://www.w3.org/TR/WD-logfile.html
  “Extended Log File Format”（“扩展日志文件格式”）。
- http://www.ietf.org/rfc/rfc2227.txt
  RFC 2227，J. Mogul 和 P. Leach 编写的“Simple Hit-Metering and Usage-Limiting for HTTP”（“简单的 HTTP 命中率测量和使用限制”）。

495