Traffic Server 管理员指南

千石（淘宝网）

qianshi@taobao.com

Overview

Traffic Server 可以加速 Internet 访问，增强 web 站点性能，同时也提供前所未有的网络托管能力。

Traffic Server 是什么

Traffic Server 是一个高性能的 web 代理缓存，它通过将频繁访问的信息缓存在网络的边缘来改善网络的效率和性能。这使访问内容在地理上更接近终端用户，在更快分发的同时也减少了带宽的占用。Traffic Server 致力于通过充分利用现有可用的带宽，来改善企业、ISP、骨干网提供商和大型企业内部网的内容分发效率。

Traffic Server 部署选项

Traffic Server 有如下部署方式：

- 作为一个 web 代理缓存
- 作为一个反向代理
- 部署在多级缓存

Traffic Server 作为 web 代理缓存

作为 web 代理缓存，Traffic Server 接收用户直接发往 web server（源服务器）的 web 内容请求。如果 Traffic Server 包含请求的内容，它将直接提供服务。如果请求的内容不在缓存里，Traffic Server 将作为一个代理：为用户从源服务器取得请求的内容，并在本地保存一份拷贝以服务于将来相同的请求。

Traffic Server提供直接代理缓存功能，这需要配置用户的客户端软件将请求直接发送给Traffic Server。直接代理缓存将在“直接代理缓存”节讲述。

Traffic Server 作为反向代理

作为反向代理，Traffic Server需要配置为用户直接连接的源服务器（典型的用法是将源服务器的主机名解析到Traffic Server）。反向代理的功能也被称为服务

器加速。反向代理的更多细节将在反向代理和HTTP重定向中描述。

Traffic Server 部署在多级缓存

Traffic Server 可以灵活地参与多级缓存，当 Internet 请求不能在一个缓存中得到满足的时候，将被路由到其他区域的缓存，从而利用附近缓存的内容。在一个多级代理中，Traffic Server 可以作为其他 Traffic Server 系统或者和其相似的缓存产品的父节点或者子节点。

Traffic Server支持ICP（Internet Cache Protocol）互连。多级缓存的更多细节将在多级缓存中介绍。

Traffic Server 组件

Traffic Server 由若干一起工作的组件来组成一个便于监控和配置的 web 代理缓存。这些主要的组件将在下面介绍。

Traffic Server 缓存

Traffic Server 缓存由高速对象数据库 Object store 组成。对象数据库通过 URL 和相关的头部来索引对象。使用先进的对象管理，Object store 可以缓存同一个对象（可能是不同的语言或编码类型）的替换版本。它同样可以高效地存储非常小和非常大的对象，从而使浪费的空间最小化。当缓存被占满后，Traffic Server 通过删除过期的数据来保证经常被请求的对象有效并容易获取。

Traffic Server被设计为容忍缓存磁盘的任何失效。如果磁盘完全失效，Traffic Server会标记整个磁盘为被占用同时继续使用余下的磁盘。如果缓存的所有磁盘都失效了，Traffic Server会自动切换为单纯代理模式。可以对缓存进行分区，来为专门的协议和源服务器存储数据预存一定数量的磁盘空间。更多关于缓存的信息见配置缓存。

RAM 缓存

Traffic Server维护着一个包含热点对象的微型RAM缓存。这个RAM缓存在尽可能快地服务大部分热点对象的同时也减少了磁盘负载，特别是在一些流量的高峰。可以根据需要来配置RAM缓存的大小；更多的细节见改变RAM缓存大小。

Host 数据库

Traffic Server host 数据库负责存储连接的源服务器 DNS 记录。这个信息用来适应未来协议的交互以及性能的优化。host 数据库信息包含：

- DNS 信息（加速主机名和 IP 地址的转换）
- 每个 host 的 HTTP 版本（可以在新版的服务器上使用增强的协议功能）

- Host 的可靠性和可用性信息（用户可以不用等待不工作的服务器）

DNS 解析器

Traffic Server 包含一个快速的、异步的 DNS 解析器来简化主机名和 IP 地址的转换。Traffic Server 最初实现的 DNS 解析器就是直接发送 DNS 命令数据包而不是依赖传统的慢速解析库。由于许多 DNS 查询可以并行发送，同时在内存的高速 DNS 缓存中维持着热点绑定（DNS，IP），使 DNS 流量减少。

Traffic Server 进程

Traffic Server 包括三个一起工作的进程来服务 Traffic Server 的请求，管理/控制/监控系统的健康状况。图 1 说明了三个进程的关系，三个进程将会在下面描述。

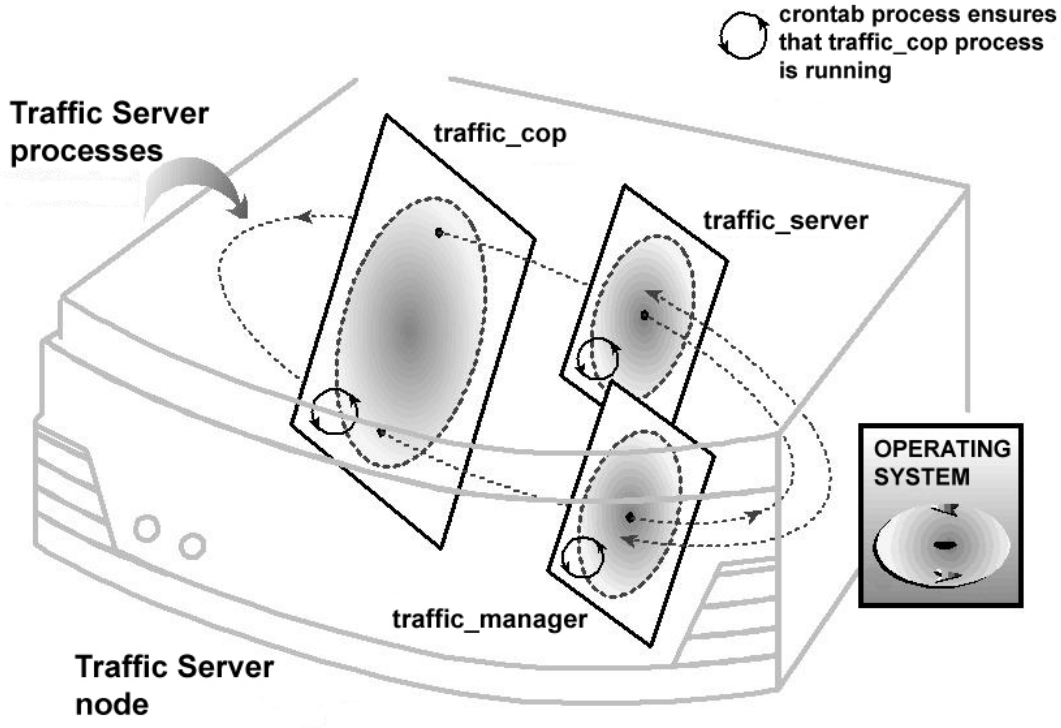

图 1. 进程间关系

- traffic_server 进程是 Traffic Server 的事务处理引擎。它负责接收连接、处理协议请求以及从本地缓存或源服务器提供资源。
- traffic_manager 进程是用来命令和控制 Traffic Server 的工具，负责启动、监控以及重新配置 traffic_server 进程。traffic_manager 进程同时负责代理自动配置端口、统计接口、集群管理以及 VIP 故障转移。

如果 traffic_manager 进程检测到 traffic_server 进程失败，它不仅会立即重启该进程，而且会为所有传入的请求维护一个连接队列。在 traffic_server 重新启动前的几秒内传入的所有连接将被保存在一个队列，并以 FIFO 的方式处理。这个连接队列接收任何 server 故障重启时的连接。

- traffic_cop 进程监控 traffic_server 和 traffic_manager 进程的健康状况。traffic_cop 进程通过抓取合成 web 页面的心跳请求方式周期性地（每分钟若干次）查询 traffic_server 和 traffic_manager 进程。如果失败事件发生（如果在超时时间间隔内没有收到请求或者收到错误的请求），traffic_cop 重启 traffic_server 和 traffic_manager 进程。

管理员工具

Traffic Server 提供如下的管理选项：

- Traffic Line 命令行接口是一个基于文本的接口，通过它不但可以监控 Traffic Server 的性能和网络流量，而且可以配置 Traffic Server 系统。通过 Traffic Line 可以执行单条命令或者是一系列命令组成的 shell 脚本。
- Traffic Shell 命令行接口是一个附加的命令行工具，通过它同样可以执行单个监控命令以及配置 Traffic Server 系统。
- 各种各样的配置文件可以通过一个简单的文件编辑器以及信号处理接口来配置 Traffic Server。任何通过 Traffic Line 或者 Traffic Shell 的更改都会自动地同步到配置文件。

流量分析选项

Traffic Server 为流量分析和监控提供了若干选项：

- Traffic Line 和 Traffic Shell 可以用来收集和处理从网络流量信息中获取的统计数据。
- 事务日志记录了（在一个日志文件中）Traffic Server 接收的每个请求以及每个检测到的错误。通过分析日志文件，可以确定有多少人使用了 Traffic Server 的缓存，每人请求了多少信息以及哪些页面是最热的。同样也可以查看一个特定事务出错的原因以及特定时间 Traffic Server 的状态。比如，可以看到 Traffic Server 重启或者集群通信超时等状况。

Traffic Server 支持若干标准的日志文件格式，比如 Squid 和 Netscape 以及其自定义的格式。可以通过现成的分析包来分析标准格式的日志文件。为了便于日志文件分析，可以按协议或者主机来分开生成日志文件。

流量分析选项的更多细节会在流量监控中描述，Traffic Server日志选项的描述在Working with Log Files。

Traffic Server 安全选项

Traffic Server 提供了许多选项来确保 Traffic Server 系统和网络上的其他计算机进行安全的通信。安全选项如下：

- 控制客户端对 Traffic Server 代理缓存的访问。
- 通过配置 Traffic Server 使用多个 DNS 服务器来匹配站点的安全配置。比如，Traffic Server 可以使用不同的 DNS 服务器来解析防火墙内部或外部的主机名。这使得在保持内部网络配置安全的情况下，同时可以继续透

明地访问 Internet 外部网络。

- 配置 Traffic Server 在用户访问 Traffic Server 缓存内容之前对其做身份验证。
- 在反向代理模式中，客户端和 Traffic Server，以及 Traffic Server 和源服务器之间使用 SSL 终止选项来进行安全连接。
- 通过 SSL 控制访问。

Traffic Server安全选项更详细的描述见安全选项。

Getting Started

在安装了 Traffic Server 之后，可以执行如下的操作：

- 启动Traffic Server
- 启动 Traffic Line
- 启动 Traffic Shell
- 停止Traffic Server

启动 Traffic Server

可以通过 trafficserver 命令和 start 属性来手动启动 Traffic Server。这个命令启动所有一起工作的 Traffic Server 进程，包括处理请求以及管理、控制和监控系统健康状况等。

运行 trafficserver start 命令：

1. 以管理员身份登录到 Traffic Server 节点，定位到 Traffic Server 的 bin 目录下。
2. 输入如下的命令： ./trafficserver start

启动 Traffic Line

Traffic Line 提供了一种通过命令行接口查看 Traffic Server 状态以及配置 Traffic Server 系统的快速方法。要执行单个命令或多个命令的脚本，可以查看 Traffic Line Commands.

启动 Traffic Line：

1. 以管理员身份登录到 Traffic Server 节点，定位到 Traffic Server 的 bin 目录下。Traffic Server 命令格式如下：traffic_line –command argument
2. 要查看 traffic_line 的命令列表，输入 traffic_line –h 。如果 Traffic Server 的 bin 目录不在 path 中，在 Traffic Line 命令前加上./（比如： ./traffic_line -h）。

启动 Traffic Shell

Traffic Shell 是一个监控和配置 Traffic Server 的命令行工具；可以用来替代

Traffic Line。Traffic Server 通过手册（man）的形式来为 Traffic Shell 提供文档。

启动 Traffic Shell 以及阅读 man 手册：

1. 以管理员身份登录到 Traffic Server 节点，定位到 Traffic Server 的 bin 目录下。
2. 输入如下命令：./traffic_shell start
3. 输入如下命令来显示 traffic_shell 的 man 手册：man traffic_shell

man 手册描述了如何使用 Traffic Shell，如何获取可用的命令以及如何获取每个命令的文档

停止 Traffic Server

可以通过 trafficserver 命令和 stop 属性来停止 Traffic Server。这个命令停止了 Traffic Server 所有的进程（traffic_manager, traffic_server, 和 traffic_cop）。不要手动地去停止某个进程，这样会导致不可预知的后果。

运行 trafficserver stop 命令：

1. 以管理员身份登录到 Traffic Server 节点，定位到 Traffic Server 的 bin 目录下。
2. 输入如下的命令：./trafficserver stop

HTTP 代理缓存

Web 代理缓存可以用来存储高频访问的 web 对象（比如文档、图片等）并为用户的请求提供这些信息。在改善网络性能的同时，也为其他任务空出了 Internet 的带宽。

理解 HTTP Web 代理缓存

Internet 用户向遍布全球的 web 服务器发送请求。缓存服务器必须扮演成一个 web 代理服务器才能服务于这些请求。当 web 代理服务器收到 web 对象的请求时，它可以选择响应这些请求或者将它们传递给源服务器（包含被请求信息源文件的服务器）。Traffic Server 代理支持直接代理缓存方式，这种方式需要客户端软件配置成直接发送请求给 Traffic Server 代理。下面大概描述一下 Traffic Server 如何响应用户的请求。

1. Traffic Server 收到一个用户对 web 对象的请求。
2. Traffic Server 尝试着在其对象数据库（缓存）中用被请求对象的地址来定位该对象。
3. 如果对象在缓存中，Traffic Server 会检查该对象是否过期，如果对象没有过期，Traffic Server 以缓冲命中的方式用该对象来响应用户（见图 2）。

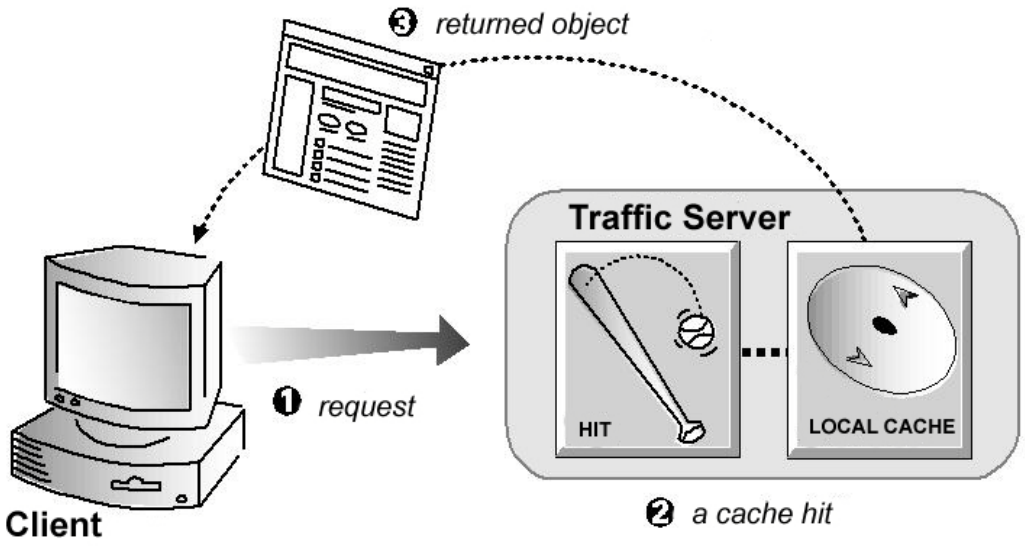

图 2. 缓存命中

4. 如果缓存中的数据已经过期，Traffic Server 连接源服务器并检查该对象是否仍然可用（重新生效）。如果生效，Traffic Server 直接发送缓存中的对象给用户。
5. 如果对象没有在缓存中（缓存未命中）或者源服务器显示缓存中的对象已经失效，Traffic Server 会从源服务器重新获取该对象。该对象会同时发送给用户以及 Traffic Server 的本地缓存（见图 3）。由于本地已经有了最新的缓存，后期对该对象的请求将会被更快的响应。

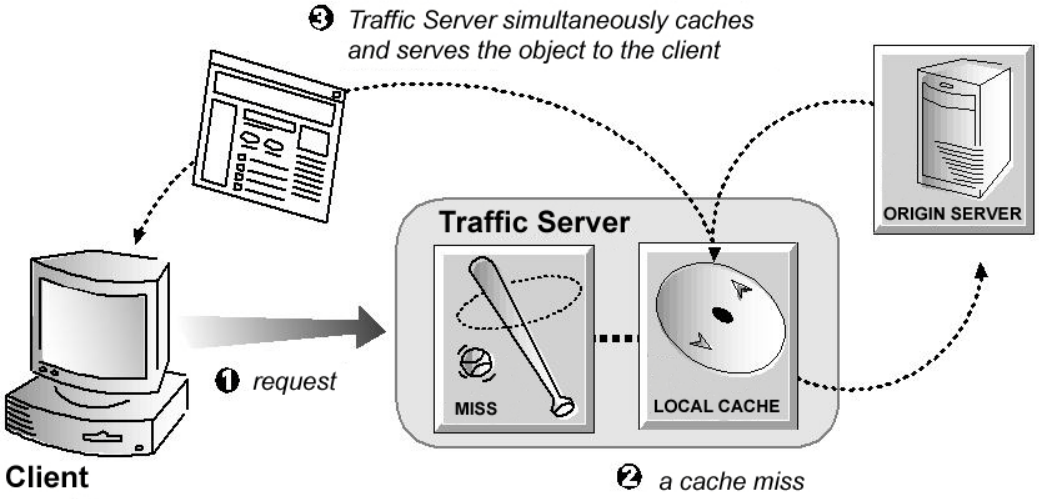

图 3. 缓存未命中

实际的缓存会比上面的概述复杂的多。尤其是概述中没有讲述 Traffic Server 如何确保对象有效，正确地响应不同的 HTTP 版本以及处理那些对不能或不该被缓存的对象的请求。下面的部分将更细致地讨论这些问题。

确保被缓存对象的有效性

当 Traffic Server 收到一个 web 对象的请求，它首先尝试着在缓存中定位该对象。如果该对象在缓存中，Traffic Server 将会检查该对象是否仍然有效。对于 HTTP

对象而言，Traffic Server 支持可选的作者自定义的有效期。Traffic Server 首先根据有效期判断；如果有效期不存在，它会在对象被改变的频率和管理员选择的有效期方案之间挑选一个有效期。可以通过源服务器检查对象有效性的方式来重新生效对象。

HTTP 对象保鲜

Traffic Server 通过如下的方式来判断缓存中的 HTTP 对象是否有效：

- 检查 Expires 或者 max-age 头

一些 HTTP 对象包含 Expires 头或者 max-age 头来明确定义对象可以被缓存的时间。Traffic Server 通过比较当前时间和有效期时间来决定该对象是否仍然有效。

- 检查 Last-Modified / Date 头

如果 HTTP 对象没有 Expires 头或者 max-age 头，Traffic Server 使用下面的公式来计算对象有效期：

freahness_limit = (date – last_modified) * 0.1

这里的 date 是对象服务器返回的日期，而 last_modified 是 Last-Modified 头部的日期。如果没有 Last-Modified 头部，Traffic Server 就使用对象写入缓存的日期。因子 0.1（10%）可以根据需要来增加或减小（见 Modifying the Aging Factor for Freshness Computations）。

计算的最大有效期被限制在一个最小值和最大值之间，更多信息见（Setting an Absolute Freshness Limit）。

- 检查绝对有效期极值

如果 HTTP 对象既没有 Expires 头部也没有 Last-Modified 和 Date 头部，Traffic Server 使用一个最大和最小有效期（见 Setting an Absolute Freshness Limit）。

- 检查 cache.config 文件中的重新生效规则

重新生效规则为特殊的对象提供有效期极值。可以为来自特殊的域或者 IP 地址的对象，URL 中包含指定的正则表达式的对象，来自特殊客户端的对象等（见 cache.config）设置有效期极值。

为有效期的计算修改老化因子

如果一个对象没有包含任何截止信息，Traffic Server 可以根据其 Last-Modified 和 Date 头部来估计它的有效期。默认地，Traffic Server 存储该对象的时长是该对象最近一次被修改后到某个固定时间的 10%。可以根据需要来增大或减小这个百分比。

为有效期的计算修改老化因子

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.cache.heuristic_lm_factor	设置这个变量来指定计算有效期的老化因子。Traffic Server 存储对象的时间依赖于这个变量。默认值为 0.1

	（10%）。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

设置一个绝对有效期极值

一些对象既没有 Expires 头部也没有 Last-Modified 和 Date 头部。为了控制这些对象在缓存中的时间，需要指定一个绝对有效期极值。

设置一个绝对有效期极值：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.cache.heuristic_min_lifetime	设置这个变量来指定没有截止时间的 HTTP 对象在缓存中有效期的最小值。默认值为 3600 秒（1 小时）。
proxy.config.http.cache.heuristic_max_lifetime	设置这个变量来指定没有截止时间的 HTTP 对象在缓存中有效期的最大值。默认值为 86400 秒（1 天）。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

指定头部的必要条件

为了更好的确保缓存中对象的有效性，可以配置 Traffic Server 只缓存有特殊头部的对象。默认地，Traffic Server 缓存所有的对象（包括没有头部的对象）；可以为专门的代理情况改变默认设置。如果配置 Traffic Server 只缓存有 Expires 或者 max-age 头部的 HTTP 对象，缓存命中率将会明显下降（因为几乎没有对象有明确的截止信息）。

配置 Traffic Server 只缓存有特殊头部的对象：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.cache.required_headers	设置这个变量为下列值之一： 0 = 对头部没有特殊要求 1 = 需要是 Last-Modified 头部，或者有明确生命期的头部，Expires 或者 Cache-Control: max-age 2 = 需要明确的生命期，Expires 或者

	Cache-Control: max-age

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

Cache-Control 头部

尽管一个对象在缓存中可能是有效的，但是客户端或者服务器经常强加它们自己的不从缓存中获取对象的约束。比如，一个客户端请求一个对象时可能不通过缓存，即使通过缓存，对象的缓存时间也不能超过 10 分钟。Traffic Server 可以给一个缓存的对象在客户端请求和服务器响应中加上 Cache-Control 头部。下面的 Cache-Control 头影响着对象是否可以通过缓存来服务：

- 客户端发送的 no-cache 头部，告诉 Traffic Server 不能直接从缓存获取任何对象；因此，Traffic Server 总是从源服务器获取对象。可以配置 Traffic Server 忽略客户端的 no-cache 头部，更多信息见 Configuring Traffic Server to Ignore Client no-cache Headers。
- 服务器发送的 max-age 头部，用来表示对象的使用期限。如果使用期限小于 max-age，表明对象是有效的，可以直接从缓存获取对象。
- 客户端发送的 min-fresh 头部，是一个可接受的有效期容忍。这意味着客户端想让对象至少这次是有效的。如果一个对象在未来的这样一段时间内不再有效，那它会重新生效。
- 客户端发送的 max-stale 头部，允许 Traffic Server 使用过期的对象，倘若这些对象不是太老的话。一些浏览器倾向于使用过期不久的对象来改善性能，尤其在 Internet 不是很发达的时期。

Traffic Server 在 HTTP 有效期标准之后使用 Cache-Control 标准。比如，一个对象可能被认为是有效的，但是如果它的使用期限大于它的 max-age，它将不会用于响应请求。

重新生效 HTTP 对象

当客户端请求一个在缓存中过期的对象，Traffic Server 将重新生效这个对象。重新生效是询问源服务器检查这个对象有没有被修改。重新生效的结果是下列情况之一：

- 如果对象还是有效的，Traffic Server 将重新设置这个对象的有效期极值，同时用这个对象来服务。
- 如果这个对象有新拷贝，Traffic Server 缓存这个新对象（替换过期对象），同时用新的对象来服务用户。
- 如果对象在源服务器已经不存在了，Traffic Server 将不为这个对象提供服务。
- 如果源服务器没有响应重新生效查询，Traffic Server 在用这个对象服务的同时，会提供 111 重新生效失败的警告。

默认情况下，Traffic Server 会重新生效一个缓存中的对象，如果它认为该对象已经过期。Traffic Server 如何评估对象的有效性已在 HTTP Object Freshness 中

描述。可以选择如下选项之一来重新配置 Traffic Server 评估有效性的方式：

- Traffic Server 认为所有在缓存中的对象都是过期的：总是对在缓存中的对象进行重新生效。
- Traffic Server 认为所有在缓存中的对象都是有效的：从不对在缓存中的对象进行重新生效。
- Traffic Server 认为所有没有 Expires 或 Cache-Control 头部的 HTTP 对象都是过期的：重新生效所有没有 Expires 或 Cache-Control 头部的 HTTP 对象。

可以通过在 cache.config 文件（见 cache.config）中设置特殊的重新生效规则来配置 Traffic Server 重新生效缓存中对象的方式。

配置重新生效选项：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.cache.when_to_revalidate	设置这个变量为下列值之一： 0 = 配置 Traffic Server 重新生效 HTTP 对象，当它认为该对象在缓存中已过期（如果可以的话，Traffic Servre 检查对象头部和有效期极值）。这是默认配置。 1 = 配置 Traffic Server 重新生效没有 Expires 或 Cache-Control 头部的 HTTP 对象。 2 = 配置 Traffic Server 总是重新生效 HTTP 对象；Traffic Server 总是认为 HTTP 对象是过期的。 3 = 配置 Traffic Server 从不重新生效 HTTP 对象；Traffic Server 总是认为 HTTP 对象是有效的。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

定时更新本地缓存内容

为了进一步增加性能并确保缓存中 HTTP 对象的有效性，可以使用预定更新选项。这个配置可以使 Traffic Server 定时加载特殊的对象到缓存。当使用 Traffic Server 作为反向代理的时候，定时更新功能显得非常有用，它可以根据预计的需求来预加载内容。

为了使用定时更新选项，必须完成如下的工作。

- 指定想定时更新对象的 URL 列表，更新发生的时间以及 URL 递归的深度。
- 开启定时更新选项，同时配置可选的重试次数。

Traffic Server 通过指定的信息来确定相关的 URL。Traffic Server 为每个 URL

获取所有递归的 URL（如果可以的话）并生成一个唯一的 URL 列表。Traffic Server 使用这个列表为每个未访问的 URL 发起 HTTP GET 操作，在任意给定的时间内，确保用户定义的极值下 HTTP 的流通性。这个系统记录了所有 HTTP GET 操作的完成过程，可以基于此来监控定时更新的性能状况。

Traffic Server 也通过一个强制立刻更新的选项，可以在不等待指定更新时间的情况下立刻更新 URL。可以用这个选项来测试定时更新配置（见 Forcing an Immediate Update）。

配置定时更新选项

配置定时更新选项需要下面几步：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 update.config 文件。
2. 在文件中按行输入每个要更新的 URL（见 update.config）。
3. 保持并关闭 update.config 文件。
4. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
5. 编辑下面的变量：

变量	描述
proxy.config.update.enabled	设置这个变量为 1 来开启定时更新选项。
proxy.config.update.retry_count	设置这个变量来指定当定时更新一个 URL 失败时重试的次数。默认值为 10。
proxy.config.update.retry_interval	设置这个变量来指定当定时更新一个 URL 失败时重试的时间间隔（单位：秒）。默认值为 2。
proxy.config.update.concurrent_updates	设置这个变量来指定在任意同一时间点允许同时更新的最大请求数。这个选项来防止定时更新进程给主机带来超负荷。默认值为 100。

6. 保存并关闭 records.config 文件
7. 定位到 Traffic Server 的 bin 目录
8. 运行 traffic_line –x 命令来应用配置文件的变更。

强制立刻更新

Traffic Server 提供一个强制立刻更新选项，便于立刻验证 update.config 文件中的 URL 列表。这个强制更新选项忽略 update.config 中设置的更新时间间隔并且立刻更新 URL 列表。

配置强制更新选项需要下面几步：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.update.force	设置这个变量为 1 来开启强制更新选项。

3. 确保 proxy.config.update.enabled 变量设置为 1。
4. 保存并关闭 records.config 文件
5. 定位到 Traffic Server 的 bin 目录
6. 运行 traffic_line –x 命令来应用配置文件的变更。

重要：当你开启了强制更新选项，Traffic Server 将不断地更新 update.config 文件中指定的 URL 直至关闭了该选项。要关闭强制更新选项，设置 proxy.config.update.force 变量为 0。

将内容推进缓存

Traffic Server 支持 HTTP PUSH 方法的内容分发。使用 HTTP PUSH，可以不借助用户请求直接将内容传递到缓存。

配置 Traffic Server 接受 PUSH 请求

在使用 HTTP PUSH 传递内容到缓存之前，必须先配置 Traffic Server 接受 PUSH 请求。

配置 Traffic Server 接受 PUSH 请求：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 filter.config 文件。
2. 向文件添加如下的过滤规则来确保只有某些 IP 地址可以向缓存发送 PUSH 请求：

domain=. src_ip=ipaddress method=PUSH action=allow

domain=. method=PUSH action=deny

这里的 ipaddress 是 Traffic Server 接受 PUSH 请求的主机 IP 地址或一个范围内的主机 IP 地址。

3. 保持并关闭 filter.config 文件。
4. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
5. 编辑下面的变量：

变量	描述
proxy.config.http.push_method_enabled	设置这个变量为 1 来开启 Traffic Server 接受 PUSH 请求。

6. 保存并关闭 records.config 文件
7. 定位到 Traffic Server 的 bin 目录
8. 运行 traffic_line –x 命令来应用配置文件的变更。

理解 HTTP PUSH

PUSH 使用 HTTP 1.1 的消息格式。PUSH 请求的实体包括响应的头部以及想在

缓存中替换的响应实体。下面是一个 PUSH 请求的例子：

```
PUSH http://www.company.com HTTP/1.0
Content-length: 84

HTTP/1.0 200 OK
Content-type: text/html
Content-length: 17

<HTML>
a
</HTML>
```

注意：头部必须包括 Content-length；Content-length 必须包括头部和实体的字节数。

保留缓存中的内容

固定缓存选项配置 Traffic Server 保证一些 HTTP 对象一段指定的时间内都在缓存中。可以使用这个选项来确保一些最热点的对象在需要的时候会在缓存中，阻止 Traffic Server 删除重要的对象。Traffic Server 观测 Cache-Control 头部并只会保留一个确定可缓存的对象在缓存中。

设置缓存固定规则并开启缓存固定功能：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 cache.config 文件。
2. 为每个想让 Traffic Server 保留在缓存中的 URL 添加一条如下的规则：

   url_regex=URL pin-in-cache=12h

   这里的 URL 是想让 Traffic Server 固定在缓存中的 URL。时间的格式可以是 d（天）、h（小时）、m（分钟）以及 s（秒）。也可以使用混合格式：比如，1h15m20m。可以向规则（更多信息见 cache.config）添加次要的规定（比如前缀或后缀）。
3. 保持并关闭 cache.config 文件。
4. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
5. 编辑下面的变量：

变量	描述
proxy.config.cache.permit.pinning	设置这个变量为 1 来开启固定缓存选项。

6. 保存并关闭 records.config 文件
7. 定位到 Traffic Server 的 bin 目录
8. 运行 traffic_line –x 命令来应用配置文件的变更。

缓存还是不缓存

当 Traffic Server 收到一个对没有在缓存中的对象的请求，它从源服务器取回这个对象并用其来响应用户。同时，在缓存中缓存这个对象以便服务后来的请求

之前，Traffic Server 先检查这个对象是否可以缓存。

缓存 HTTP 对象

Traffic Server 响应来自客户端、源服务器以及通过配置选项和文件指定的缓存指示。

客户端指示

默认情况下，Traffic Server 不缓存含有如下请求头部的对象：

- Cache-Control: no-store 头部
- Cache-Control: no-cache 头部

配置 Traffic Server 忽略 Cache-Control: no-cache 头部，见 Configuring Traffic Server to Ignore Client no-cache Headers。

- Cookie: 头部（文本对象）

默认情况下，Traffic Server 缓存包含 cookies 请求服务的响应对象（除了文本对象）。可以配置 Traffic Server 不缓存任何类型的 cookies 内容、缓存所有的 cookies 内容或者只缓存图片类型的 cookies 内容。更多信息见 Caching Cookied Objects。

- Authorization: 头部

配置 Traffic Server 忽略客户端的 no-cache 头部

默认情况下，Traffic Server 严格遵守客户端 Cache-Control: no-cache 的指示。如果一个被请求的对象包含 no-cache 头部，即使它在缓存中仍然有效，Traffic Server 也会将该请求传递给源服务器。可以配置 Traffic Server 忽略客户端 no-cache 指示，这样它将忽略客户端请求的 no-cache 头部并用缓存中的对象服务该请求。

配置 Traffic Server 忽略客户端的 no-cache 头部

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.cache.ignore_client_no_cache	设置这个变量为 1 来忽略客户端请求旁路缓存。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

源服务器指示

默认情况下，Traffic Server 不缓存包含如下响应头部的对象：

- Cache-Control: no-store 头部
- Cache-Control: private 头部
- WWW-Authenticate: 头部

要配置 Traffic Server 忽略 WWW-Authenticate 头部，见 Configuring Traffic Server to Ignore WWW-Authenticate Headers。

- Set-Cookie: 头部
- Cache-Control: no-cache 头部

要配置 Traffic Server 忽略 no-cache 头部，见 Configuring Traffic Server to Ignore Server no-cache Headers。

- Expires: 头部包含 0 值或过去时间

配置 Traffic Server 忽略服务器 no-cache 头部

默认情况下，Traffic Server 严格遵守 Cache-Control: no-cache 指示。一个来自源服务器的带有 no-cache 头部的响应将不会被存储在缓存，该对象之前在缓存中的拷贝也会被删除。如果配置 Traffic Server 忽略 no-cache 头部，Traffic Server 同时也忽略 no-store 头部。在大多数情况下是应该遵守 no-cache 指示的。

配置 Traffic Server 忽略服务器 no-cache 头部

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.cache.ignore_server_no_cache	设置这个变量为 1 来忽略服务器旁路缓存。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

配置 Traffic Server 忽略 WWW-Authenticate 头部

默认情况下，Traffic Server 不缓存包含 WWW-Authenticate 响应头部的对象。WWW-Authenticate 头部包含着客户端准备用来响应源服务器挑战应答的鉴定参数。

当配置 Traffic Server 忽略源服务器的 WWW-Authenticate 头部，所有带 WWW-Authenticate 头部的对象将被存储在缓存中被用来服务后来的请求，在大多数情况下，应该使用默认的不缓存带 WWW-Authenticate 头部对象的行为。只有在对 HTTP 1.1 深入理解的基础上，再尝试配置 Traffic Server 忽略服务器 WWW-Authenticate 头部。

配置 Traffic Server 忽略 WWW-Authenticate 头部

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.cache.ignore_authentication	设置这个变量为 1 来缓存带 WWW-Authenticate 头部的对象。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

配置指示

除了客户端和服务器的指示，Traffic Server 同样响应配置选项和文件的指示。可以按如下步骤来配置 Traffic Server：

- 不缓存任何对象（见 Disabling HTTP Object Caching）。
- 缓存动态内容 — 对象的 URL 以.asp 结尾或者包含问号（?）、分号（:）或者 cgi。更多信息见 Caching Dynamic Content。
- 缓存响应 Cookie: 头部的对象（见 Caching Cookied Objects）。
- 遵守 cache.config 文件中的从不缓存规则（见 cache.config）。

关闭 HTTP 对象缓存功能

默认情况下，Traffic Server 缓存除了在 cache.config 文件中设置了从不缓存规则的所有对象。可以关闭 HTTP 对象缓存功能，所有的对象都直接由源服务器服务而且从不缓存。

手动配置关闭 HTTP 对象缓存功能：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.cache.http	设置这个变量为 0 来关闭 HTTP 对象缓存功能。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

缓存动态内容

一个以.asp 结尾或包含问号（?）、分号（;）或者 cgi 的 URL 被认为是动态的。Traffic Server 不缓存动态内容。可以配置 Traffic Server 缓存动态内容，当然这只推荐在专门的代理情形下使用。

配置 Traffic Server 缓存动态内容：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config

文件。

2. 编辑下面的变量：

变量	描述
proxy.config.http.cache_urls_that_look_dynamic	设置这个变量为 1 来缓存动态内容。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

缓存 Cookied 对象

默认情况下，Traffic Server 缓存包含 cookies 请求服务的响应对象（除了文本对象）。Traffic Server 之所以不缓存文本内容的 cookied，是因为对象的头部和对象是一起存储的，而带有隐私的 cookie 头部是不能和对象一起保存的。对于非文本对象，不能确定是否使用了带有隐私的 cookie 头部。

可以配置 Traffic Server：

- 不缓存任何类型的 cookies 内容。
- 只缓存图片类型的 cookies 内容。
- 缓存所有的 cookies 内容。

配置 Traffic Server 缓存 cookied 内容的方式：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.cache.cache_responses_to_cookies	设置这个变量来指定 Traffic Server 缓存 cookied 内容的方式： 0 = 不缓存任何 cookies 响应。 1 = 缓存所有的 cookies 响应。 2 = 只缓存图片类型的 cookies 响应。 3 = 缓存除了文本内容类型的所有 cookies 响应。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

强制对象缓存

可以强制 Traffic Server 缓存特殊的 URL（包括动态 URL）一段指定的时间，而不考虑 Cache-Control 响应头部。

强制缓存文档：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 cache.config 文件。
2. 为每个想让 Traffic Server 强制缓存的 URL 添加一条如下的规则：

url_regex=URL　　ttl-in-cache=6h

这里的 URL 是想让 Traffic Server 强制缓存的 URL。时间的格式可以是 d（天）、h（小时）、m（分钟）以及 s（秒）。也可以使用混合格式：比如，1h15m20m。可以向规则（更多信息见 cache.config）添加次要的规定（比如前缀或后缀）。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

缓存 HTTP Alternates

一些源服务器用多种对象来响应同一个 URL 的请求。这些对象的内容是多种多样的，比如提供不同语言内容的服务、为不同的目标浏览器提供不同的表达方式、或者提供不同的文档格式（HTML, PDF）。同一对象的不同版本被称为 Alternates，Traffic Server 基于 Vary 头部来缓存它们。可以为特殊的内容类型指定附加的请求和响应头部来让 Traffic Server 识别其为 Alternates 并缓冲之。也可以限制一个对象允许缓存不同版本的数目。

配置 Traffic Server 缓存 Alternates 的方式

配置 Traffic Server 缓存 Alternates 的方式需要如下步骤：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.cache.enable_default_vary_headers	设置这个变量为 1 来缓存不包含 Vary 头部的 HTTP 对象的不同版本。
proxy.config.http.cache.vary_default_text	设置这个变量来指定在请求是文本时 HTTP 头部的多种格式：比如，HTML 文档。
proxy.config.http.cache.vary_default_images	设置这个变量来指定在请求是图像时 HTTP 头部的多种格式：比如，.gif 文件。
proxy.config.http.cache.vary_default_other	设置这个变量来指定在请求不是文本和图像时 HTTP 头部的多种格式。

注意：如果在上述变量中指定 Cookie 作为头部变化格式，必须确保正确设置了 proxy.config.http.cache.cache_responses_to_cookies 变量。

比如，如果设置 proxy.config.http.cache.cache_responses_to_cookies 变量为 2

（只缓存图片格式的 cookies 响应），同时设置 proxy.config.http.cache.vary_default_text 变量为指定的 cookie，cookie 的 alternates 将不接受文本格式。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

限制一个对象的 Alternates 个数

可以限制 Traffic Server 缓存每个对象 alternates 的个数（默认值为 3）。

重要：alternates 的数量太多会影响 Traffic Server 的性能，因为所有的 alternates 有同一个 URL。尽管 Traffic Server 可以从索引中快速查找 URL，但是必须在对象存储中顺序扫描直到找到可用的 alternates。

限制 alternates 的数目：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.cache.limits.http.max_alts	设置这个变量来指定想让 Traffic Server 缓存一个对象不同版本的数目。默认值为 3。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

使用拥塞控制

拥塞控制选项可以配置 Traffic Server 当源服务器开始拥塞时停止向它们转发 HTTP 请求。Traffic Server 给客户端发送一个稍后重新请求拥塞源服务器的消息。

要使用拥塞控制选项，必须完成如下的工作：

- 开启拥塞控制选项。
- 在 congestion.config 文件中创建规则来指定：
  - Traffic Server 跟踪哪个源服务器的拥塞情况
  - Traffic Server 根据超时设置，判断一个服务器是否拥塞
  - 当服务器拥塞时，Traffic Server 发送给客户端的页面
  - Traffic Server 是否跟踪源服务器的每个 IP 地址和主机名

开启和配置拥塞控制选项：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.http.congestion_control.enabled	设置这个变量为 1 来开启拥塞控制选项。

3. 保存并关闭 records.config 文件。
4. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 congestion.config 文件。
5. 输入规则来指定跟踪哪个源服务器的拥塞情况以及 Traffic Server 用来测定拥塞的超时时间值。规则的格式见 congestion.config。
6. 保存并关闭 congestion.config 文件。
7. 定位到 Traffic Server 的 bin 目录。
8. 运行 traffic_line –x 命令来应用配置文件的变更。

直接代理缓存

如果想使用 Traffic Server 充当直接代理缓存，必须配置客户端软件（比如浏览器）直接向 Traffic Server 发送请求。

HTTP 直接代理缓存

如果没有配置 Traffic Server 使用透明选项（客户端发送给源服务器的请求被交换机/路由器中途拦截并重路由给 Traffic Server），客户端必须通过配置其浏览器从 PAC 文件（Proxy Auto-Configuration 文件）去下载代理配置指令来使浏览器直接给 Traffic Server 代理缓存发送请求。

手动配置浏览器

要手动配置浏览器直接向 Traffic Server 发送 HTTP 请求，客户端必须提供如下信息：

- Traffic Server 节点的主机名或 IP 地址
- Traffic Server 代理服务器端口（端口 8080）

另外，客户端可以指定一些站点不使用 Traffic Server -- 在这种情况下，这些站点的请求将会直接发送给源服务器。手动配置的程序在浏览器不同版本之间会有差异，可以查看特定浏览器的文档来获取完整的代理配置指令。如果要接收来自手动配置浏览器的请求，Traffic Server 不需要做任何的配置。

使用 PAC 文件

PAC 文件是一个专门的 JavaScript 函数定义，浏览器通过调用它来决定如何处理请求。可以将 PAC 文件存储在 Traffic Server（或者网络中的任一服务器），并为客户端提供这个文件的 URL。

如果想在 Traffic Server 中存储 PAC 文件，必须完成下面的配置：

- 复制一个已经存在的 PAC 文件到 Traffic Server 的 config 目录下或者在 proxy.pac 文件（默认为空）中输入用来定义代理服务器配置设置的脚本。
- 指定 Traffic Server 用来服务 PAC 文件的端口。默认端口为 8083。

手动配置 Traffic Server 提供 PAC 文件：

1. 如果已经有一个 PAC 文件，用这个文件替换 Traffic Server 目录下的 proxy.pac 文件。
2. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
3. 编辑下面的变量：

变量	描述
proxy.config.admin.autoconf_port	设置这个变量来指定 Traffic Server 服务 PAC 文件的端口。默认端口为 8083。

4. 保存并关闭 records.config 文件。
5. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 proxy.pac 文件。
   i. 如果复制已经存在的 PAC 文件到 Traffic Server 的 config 目录下，proxy.pac 文件包含着代理配置设置。检查设置并做必要的修改。
   ii. 如果没有复制 PAC 文件到 Traffic Server 的 config 目录下，proxy.pac 是空的。输入用来定义代理服务器配置设置的脚本。脚本的样本见 Sample PAC file。
6. 保存并关闭 proxy.pac 文件。
7. 重启 Traffic Server。
8. 告知用户将浏览器设置为指向这个 PAC 文件。

PAC 文件样本

下面的 PAC 文件样本指示浏览器直接连接所有没有完整资质域名的主机以及本域中的主机。其它的请求将发送给名为 myproxy.company.com 的 Traffic Server。

```
function FindProxyForURL(url, host)
{
        if (isPlainHostName(host)) ||
        (localHostOrDomainIs(host, ".company.com")) {
          return "DIRECT";
        }
        else
             return "PROXY myproxy.company.com:8080;" + "DIRECT";
}
```

反向代理和 HTTP 重定向

作为反向代理缓存，Traffic Server 为源服务器服务请求。Traffic Server 被配置成对客户端而言是正常的源服务器的方式。

理解反向代理缓存

通过前向代理缓存，Traffic Server 为客户端处理发往远距离源服务器的 web 请求。反向代理缓存（又称服务器加速或虚拟主机托管）和前向代理不同，因为 Traffic Server 作为源服务器的代理缓存并存储内容。Traffic Server 被配置为用户直接连接的源服务器（典型的用法是将源服务器的主机名解析到 Traffic Server）。

反向代理解决方案

有多种方式使用 Traffic Server 作为一个反向代理。下面是一些例子场景。可以使用 Traffic Server 的反向代理模式来：

- 为负载过高的源服务器减负
- 为地理上分散的区域高效地分发内容
- 为包含机密信息的源服务器提供安全屏障

为负载过高的源服务器减负

Traffic Server 可以吸收发向源服务器的请求，通过减少源服务器的负载和热点来改善 web 服务的速度和质量。比如，一个网络托管商可以用一系列低价格、低性能、低可靠性的 PC 作为后备服务器来维护一个可扩展的 Traffic Server 服务引擎。事实上，单个 Traffic Server 可以为多个后台源服务器充当虚拟源服务器，如图 4 所示。

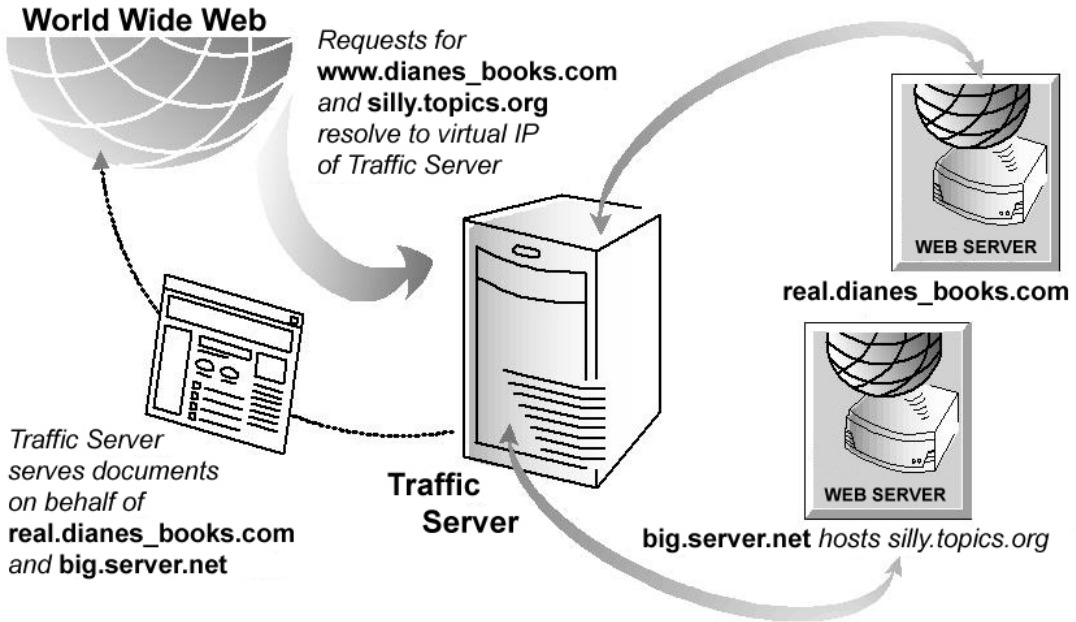

图 4. Traffic Server 为两个源服务器做反向代理

在地理上分散的区域分发内容

Traffic Server 可以使用反向代理模式来加速为地理上不邻近的区域提供服务。缓存更易于管理并且比复制数据更划算。比如，Traffic Server 可以在大洋彼岸充当镜像站点，不需要通过昂贵的国际连接发送请求和获取内容就可以服务用户。

相比于硬件必须要配置成复制所有的数据，并且要处理高峰情况下的复制，Traffic Server 可以通过动态调整来最好的利用硬件的服务和存储能力。Traffic Server 同时可以自动保持内容的有效性，从而排除了复杂的更新远端源服务器的操作。

为源服务器提供安全保障

Traffic Server 可以使用反向代理模式来为源服务器提供安全保障。如果想让包含机密信息的源服务器位于防火墙内来保证安全，可以在防火墙外用 Traffic Server 作为源服务器的反向代理。当外面的客户端试图访问源服务器，请求会被发送到 Traffic Server。如果请求的内容不是机密的，可以通过缓存来服务。如果内容是机密的并且没有缓存，Traffic Server 就从源服务器获取该内容（防火墙只允许 Traffic Server 访问源服务器）。位于源服务器的机密内容就可以安全地留在防火墙内。

反向代理的工作方式

当浏览器有请求，它一般直接发送这些请求到源服务器。当 Traffic Server 工作在反向代理模式，它在这些请求到达源服务器之前就拦截它们。典型的用法是将源服务器的 DNS 入口（主机名）解析成 Traffic Server 的 IP 地址。当 Traffic Server 被配置为源服务器，浏览器会和 Traffic Server 而非源服务器建立连接。更多信息见 HTTP Reverse Proxy。

注意：为了避免 DNS 冲突，源服务器的主机名和外在 DNS 主机名必须不同。

HTTP 反向代理

在反向代理模式中，Traffic Server 为 Web 服务器处理 HTTP 请求。反向代理模式下，Traffic Server 处理从客户端浏览器 HTTP 请求的方式如图 5 所示。

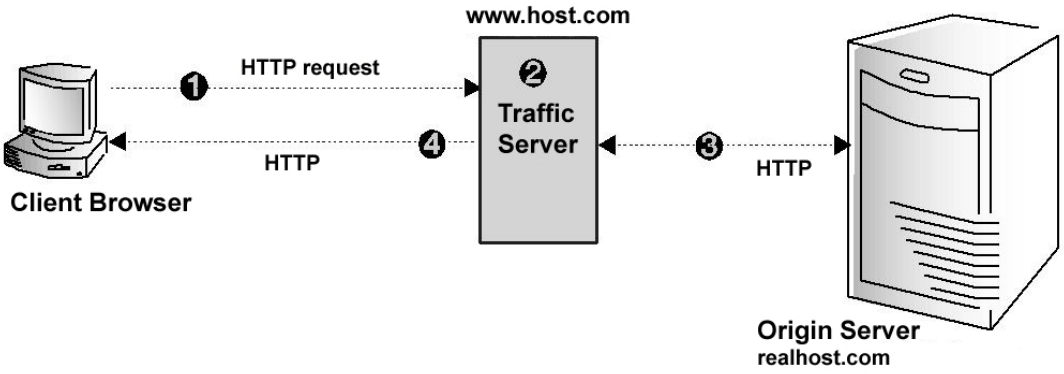

图 5. HTTP 反向代理

1. 客户端浏览器在 80 端口向名为www.host.com 的host发送一个HTTP请求。Traffic Server以源服务器的角色接收这个请求（源服务器的对外主机名被解析到Traffic Server）。
2. Traffic Server 在 remap.config 文件中定位映射规则并重新映射这个请求到一个指定的源服务器（realhost.com）。
3. Traffic Server 建立一个和源服务器的 HTTP 连接。
4. 如果请求在缓存中命中而且内容是有效的，Traffic Server 直接从缓存中向客户端发送被请求的对象。否则，Traffic Server 从源服务器获取被请求的对象，发送给客户端，同时在缓存中保存一份该对象的拷贝。

要配置 HTTP 反向代理，必须完成如下工作：

- 在 remap.config 文件中创建映射规则（见 Creating Mapping Rules for HTTP Requests）。
- 开启反向代理选项（见 Enabling HTTP Reverse Proxy）。

除了上面的工作，也可以 Set Optional HTTP Reverse Proxy Options。

为 HTTP 请求创建映射规则

在前向代理缓存中，Traffic Server 作为代理服务器并接收代理请求。在反向代理缓存中，Traffic Server 必须作为源服务器而不是代理服务器，这意味着它接收服务器请求而非代理请求。因此 ，为了满足代理请求，Traffic Server 必须根据服务器请求构造一个代理请求。

在 HTTP 中，代理请求指定完整的 URL 而服务器请求只指定路径。一个服务器请求大致格式如下：

GET /index.html HTTP/1.0 Host: real.dianes_books.com

然而，对应的代理请求格式如下：

GET http://real.dianes_books.com/index.html HTTP/1.0 Host: real.dianes_books.com

Traffic Server 可以通过 host 头部中的服务器信息从一个服务器请求构造一个代理请求。然而，正确的代理请求必须包含源服务器的主机名，而不是域名服务器解析到 Traffic Server 的对外主机名。对外主机名是出现在 host 头部的主机名；对源服务器 real.dianes_books.com 而言，服务器请求和 host 头部应该是：

GET /index.html HTTP/1.0 Host: www.dianes_books.com

而正确的代理请求应该是：

GET http://real.dianes_books.com/index.html HTTP/1.0 Host: real.dianes_books.com

要将www.dianes_books.com替代为real.dianes_books.com，Traffic Server需要一系列URL重写规则（映射规则）。映射规则在Using Mapping Rules for HTTP Requests中描述。

通常，使用代理模式可以支持多个源服务器。在这种情况下，所有的对外主机名都被解析为 Traffic Server 的 IP 地址或虚拟 IP 地址。使用 host 头部，Traffic Server 可以为任意数量的服务器将服务器请求转换为代理请求。如果 Traffic Server 收到来自老版本浏览器发送的不支持 host 头部的请求，Traffic Server 可以将这些请求直接路由给指定的服务器，或者向浏览器发送一个包含错误信息的 URL（见 Setting Optional HTTP Reverse Proxy Options）。

处理源服务器重定向响应

源服务器经常给浏览器返回重定向到不同页面的响应。比如，如果源服务器超载，它会重定向浏览器到一个低负载的服务器。源服务器也重定向已经被移到不同位置下的 web 页面。当 Traffic Server 被配置成一个反向代理，它必须通过改写来自源服务器的重定向来使浏览器重定向到 Traffic Server 而不是其他的源服务器。

Traffic Server 使用反向映射规则来改写重定向。通常，需要为每个映射规则设置一个反向映射规则。要创建反向映射规则，见 Using Mapping Rules for HTTP Requests。

为 HTTP 请求使用映射规则

Traffic Server 为 HTTP 反向代理使用两种类型的映射规则：

- 一个映射规则是将一个客户端的请求 URL 转换为定位内容的 URL。当 Traffic Server 使用反向代理模式并接收客户端的 HTTP 请求，它必须从相关的 URL 和头部来构造一个完整的 URL。Traffic Server 接下来使用这个完整的 URL 在 remap.config 文件中查找其匹配的目标 URL。要使请求 URL 匹配目标 URL，下面的条件必须满足：
  - 两个 URL 必须是一个组合
  - 两个 URL 的 host 必须相同。如果请求 URL 中包含一个不合格的主机名，它不可能匹配到一个具有合格的主机名的目标 URL。
  - 两个 URL 的端口号必须相同。如果指定的 URL 中没有端口号，URL 的组合将使用默认的端口号。
  - 目标 URL 的路径部分必须可以和请求 URL 的前缀匹配。

如果 Traffic Server 找到一个匹配，它将请求 URL 转换为映射规则中的替换 URL：它设置请求 URL 的 host 和路径去匹配替换 URL。如果 URL 包含 path 前缀，Traffic Server 去掉 path 的和目标 URL 匹配的前缀，同时用替换 URL 中 path 来替换该 path。如果一个请求 URL 有两个对应的匹配，Traffic Server 接受 remap.config 文件中的第一个匹配。

- 一个反向映射规则是将源服务器重定向响应中的URL转换为指向Traffic Server，这样客户端会再次重定向到Traffic Server而不是直接访问一个源服务器。比如，在源服务器www.molasses.com 上有一个/pub目录，一个客户端向该源服务器发送一个/pub的请求，源服务器可能回复一个http://www.test.com/pub/ 的重定向来告诉客户端它请求的是一个目录，而不是文档（重定向通常的应用是标准化URL，这样客户端可以正确地标记文档）。

  Traffic Server 使用反向代理规则来阻止客户端（从源服务器收到重定向）绕过 Traffic Server 直接访问源服务器。

映射和反向映射规则都由一个目标（源）URL 和一个替换（目的）URL 组成。在一个映射规则中，目标 URL 指向 Traffic Server 而替换 URL 指定源服务器内容的位置。在一个反向映射规则中，目标 URL 指定源服务器内容的位置而替换 URL 指向 Traffic Server。Traffic Server 在位于 Traffic Server 的 config 目录下的 ramap.config 文件中存储映射规则。

创建映射规则：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 remap.config 文件。
2. 输入映射和反向映射规则（见 remap.config）。
3. 保存并关闭 remap.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line －x 命令来应用配置文件的变更。

开启 HTTP 反向代理

开启 HTTP 反向代理需要以下步骤。

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.reverse_proxy.enabled	设置这个变量为 1 来开启 HTTP 反向代理模式。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

设置可选的 HTTP 反向代理选项

Traffic Server 提供了下面几个可配置的反向代理选项：

- 配置 Traffic Server 在转换的时候保存客户端的 host 头部信息
- 配置 Traffic Server 只服务发向映射规则中的源服务器的请求。因此，发向不在映射规则中源服务器的请求将不会被服务。
- 为来自老版本客户端的请求指定一个替换的 URL（比如，没有提供 Host 头部的客户端）。

设置可选的 HTTP 反向代理选项：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.url_remap.pristine_host_hdr	设置这个变量为 1 来保存客户端请求的 host 头部。 如果想要 Traffic Server 转换客户端请求的 host 头部，设置这个变量为 0。
proxy.config.url_remap.remap_required	如果想让 Traffic Server 只为在 remap.config 文件的映射规则中的源服务器的请求服务，设置这个变量为 1。 如果想让 Traffic Server 服务所有源服务器的请求，设置这个变量为 0。
proxy.config.header.parse.no_host_url_redirect	没有 host 头部的请求重定向的 URL。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

重定向 HTTP 请求

可以配置Traffic Server不联系任何源服务器就可以重定向HTTP请求。比如，如果想重定向所有去 http://www.ultraseek.com 的请求到http://www.server1.com/products/portal/search/ ，则所有去www.ultraseek.com的请求将直接去www.server1.com/products/portal/search 。

可以通过 Traffic Server 配置永久或临时重定向。永久重定向通知浏览器 URL 的变更（通过返回 HTTP 状态码 301），这样浏览器就可以更新标签。临时重定向通知浏览器只是当前请求的 URL 变更（通过返回 HTTP 状态码 307）。

设置重定向规则：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 remap.config 文件。
2. 为每个重定向设置一条映射规则。每条映射规则必须用一个分开的行并且必须由三个确定位置的区域：type、target 和 replacement 组成。下面的表描述了每个区域的格式。

变量	描述
type	输入其中的一个： redirect – 不联系源服务器而永久重定向 HTTP 请求。 redirect_temporary -- 不联系源服务器而临时重定向 HTTP 请求。
target	输入源 URL。可以由四部分组成：

	scheme://host:port/path_prefix
replacement	输入目标 URL。可以由四部分组成： scheme://host:port/path_prefix

下面的规则将所有www.server1.com 请求永久重定向到www.server2.com ：
redirect http://www.server1.com http://www.server2.com

3. 保存并关闭 remap.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

多级缓存

Traffic Server 可以工作在多级缓存中，请求在一个缓存不能完成时可以路由到别的区域的缓存，从而利用附近缓存的内容。

理解多级缓存

一个多级缓存由可以相互通信的多层缓存组成。Traffic Server 支持若干种类型的多级缓存。所有的缓冲层级都通过父节点和子节点来识别。一个父节点缓存位于层级的高层，Traffic Server 可以向其转发请求。一个子节点缓存是以 Traffic Server 为父节点的缓存。

Traffic Server 支持下面的层级缓存选项：

父节点缓存

ICP（Internet Cache Protocol）互连

父节点缓存

如果 Traffic Server 在自己的缓存中没有找到被请求的对象，在从源服务器取回这个对象之前，它会先找一个父节点缓存（这个节点也可以去找别的缓存）。可以配置 Traffic Server 节点使用一个或多个父节点缓存，这样如果一个父节点缓存不可用时，其它的父节点可以来响应请求。这就是父节点故障转移。Traffic Server 支持 HTTP 和 HTTPS 的父节点缓存。

注意：如果不想让所有的请求都去父节点缓存，可以配置 Traffic Server 将特定的请求（比如包含特殊的 URL）直接路由到源服务器。简单的父节点设置规则见 parent.config。

图 6 示例了一个简单的 Traffic Server 节点配置了父节点缓存的多级缓存。客户端向在多级缓存中位于子节点（因为它配置了转发未命中的请求到父节点缓存）的 Traffic Server 发送请求。请求在缓存中未命中，Traffic Server 就将其转发给了其父节点缓存并且在父节点缓存中命中。随后这个内容的请求可以直接由 Traffic Server 来服务（直到数据过期或失效）。

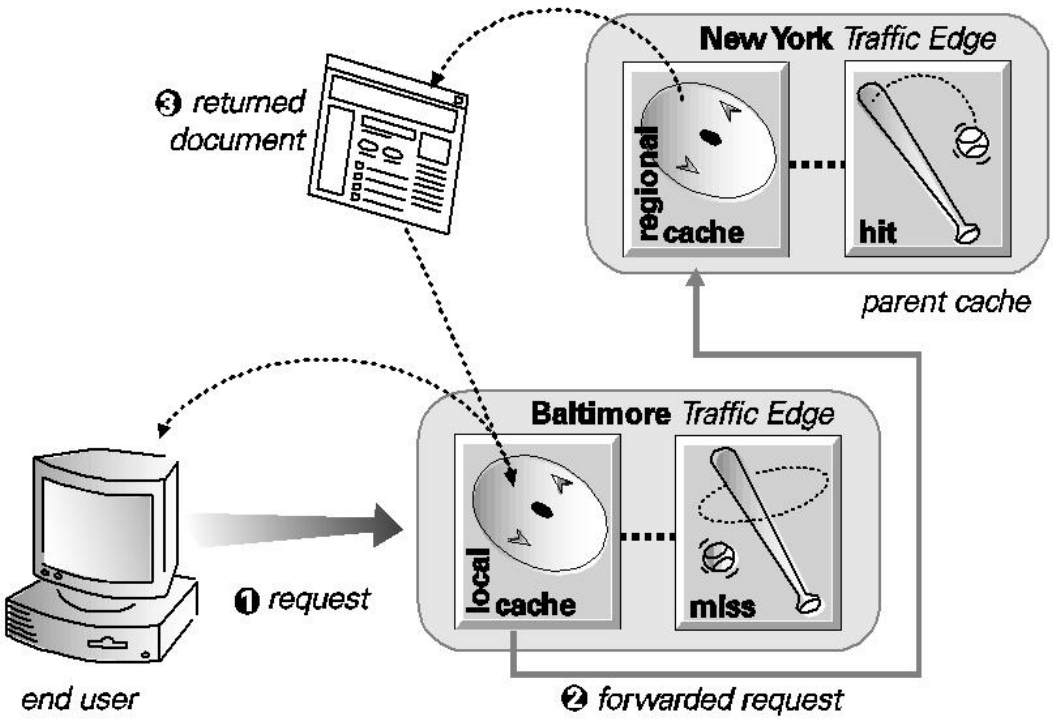

图 6. 父节点缓存

注意：如果请求在父节点缓存也未命中，父节点会从源服务器获取请求的内容（或者从其它缓存，取决于父节点的配置）。父节点缓存该内容并将内容的拷贝发送给 Traffic Server（其子节点），Traffic Server 缓存该内容并响应客户端。

父节点故障转移

Traffic Server 支持使用若干个父节点缓存。这样确保在一个父节点缓存不可以的情况下，其它的父节点缓存仍然可以服务客户端请求。

当配置 Traffic Server 使用的父节点缓存多于一个，Traffic Server 在检测到一个父节点不可用时，会发送未命中的请求到另一个父节点缓存。如果指定了多于两个父节点缓存，父节点被询问的顺序取决于父节点代理规则在 parent.config 配置文件中的顺序。默认情况下，父节点缓存被询问的顺序是它们在配置文件中的顺序。

配置 Traffic Server 使用父节点缓存

要配置 Traffic Server 使用一个或多个父节点缓存，必须完成下面的步骤：

- 开启父节点缓存选项
- 识别服务未命中请求的父节点缓存。要使用父节点故障转移，必须识别多个父节点缓存，这样当一个不可用时，请求可以发送到其它的父节点缓存。

注意：只需要配置子节点缓存。Traffic Server 父节点缓存不需要附加的配置。

配置 Traffic Server 使用父节点缓存

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config

文件。

2. 编辑下面的变量：

变量	描述
proxy.config.http.parent_proxy_routing_enable	设置这个变量为 1 开启父节点缓存选项。

3. 保存并关闭 records.config 文件
4. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 parent.config 文件。
5. 设置父节点代理规则来指定未命中的请求转发的父节点缓存；见 parent.config。

   下面的例子配置 Traffic Server 将所有包含正则表达式 politics 和路径 /viewpoint 的请求直接路由到源服务器（绕过任何的父节点缓存）：

   url_regex=politics prefix=/viewpoint go_direct=true

   下面的例子配置Traffic Server将所有以mms://host1 开头的URL未命中请求路由到父节点缓存parent1。如果parent1 不能服务这种请求，请求会被转发到parent2。因为round-robin=true，Traffic Server基于客户端IP地址轮询的方式来查询父节点缓存。

   dest_host=host1 scheme=mms parent=”parent1;parent2” round-robin=strict
6. 保存并关闭 parent.config
7. 定位到 Traffic Server 的 bin 目录
8. 运行 traffic_line –x 命令来应用配置文件的变更。

ICP 互连

Internet 缓存协议（ICP）是代理缓存用来交换内容信息的协议。ICP 查询消息询问其它的缓存是否存储了一个特殊的 URL；ICP 响应消息回应一个命中或未命中的应答。一个缓存只和特定的 ICP 节点交换消息，只有相邻的缓存可以收到 ICP 消息。一个 ICP 节点可以是一个兄弟节点缓存（在多级缓存中的同一级）或者是一个父节点缓存（在多级缓存中的上一级）。

如果 Traffic Server 开启了 ICP 缓存，当有未命中的请求时，它会向其 ICP 节点发送 ICP 查询。如果都没有命中但有父节点存在，将会用轮询策略来选择父节点。如果没有 ICP 父节点存在，Traffic Server 就将请求转发给它的 HTTP 父节点。如果没有 HTTP 父节点缓存，Traffic Server 将请求转发给源服务器。

如果 Traffic Server 从 ICP 节点收到一个命中消息，Traffic Server 就将 HTTP 请求发送到命中的节点。然而，由于源 HTTP 请求的头部信息并没有进行 ICP 查询，缓存未命中仍然有可能发生。如果一个 ICP 命中导致了一个未命中，Traffic Server 转发这个请求到其 HTTP 父节点缓存或者源服务器。

要配置 Traffic Server 为 ICP 多级缓存的一部分，必须完成下面的工作：

- 决定 Traffic Server 是只能接收 ICP 消息，还是可以发送和接收 ICP 消息。
- 决定 Traffic Server 是可以直接向每个 ICP 节点发送消息，还是只能通过指定的多播通道发送单个消息。
- 指定 ICP 消息端口号。

- 设置 ICP 查询超时。
- 识别可以通信的 Traffic Server ICP 节点（兄弟节点和父节点）

配置 Traffic Server 使用 ICP 多级缓存：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.icp.enabled	设置这个变量为： 0 = 关闭 ICP。 1 = 允许 Traffic Server 只接收 ICP 查询。 2 = 允许 Traffic Server 发送和接收 ICP 查询。
proxy.config.icp.icp_port	设置这个变量来指定 ICP 消息的 UDP 端口号。默认值为 3130。
proxy.config.icp.multicast_enabled	设置这个变量为： 0 来关闭 ICP 多播。 1 来开启 ICP 多播。
proxy.config.icp.query_timeout	设置这个变量来指定 ICP 查询的超时。默认值为 2 秒。

3. 保存并关闭 records.config 文件
4. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 icp.config 文件。
5. 为每个想识别的 ICP 节点在 icp.config 文件输入独立的规则。
6. 保存并关闭 icp.config
7. 定位到 Traffic Server 的 bin 目录
8. 运行 traffic_line –x 命令来应用配置文件的变更。

配置缓存

Traffic Server 缓存由高速对象数据库 Object store 组成。对象数据库通过 URL 和相关的头部来索引对象。

Traffic Server 缓存

Traffic Server 缓存由高速对象数据库 Object store 组成。对象数据库通过 URL 和相关的头部来索引对象。这使得 Traffic Server 不仅可以存储、检索和服务 web 页面，而且可以操作 web 页面的部分来节省带宽。使用先进的对象管理，Object store 可以缓存同一个对象（可能是不同的语言或编码类型）的替换版本。它同样可以高效地存储非常小和非常大的对象，从而最小化浪费的空间。当缓存被占满后，Traffic Server 通过删除过期的数据来保证经常被请求的对象容易获取并有效。

Traffic Server 被设计为容忍缓存磁盘的任何失效。如果磁盘完全失效，Traffic Server 会标记整个磁盘为被占用同时继续使用余下的磁盘。如果缓存的所有磁盘

都失效了，Traffic Server 会自动切换为单纯代理模式。

配置缓存需要完成的工作如下：

- 改变分配给缓存的磁盘空间总量：见改变缓存容量。
- 通过预留磁盘空间的方式为特殊的协议和源服务器/域名分割缓存：见分割缓存。
- 删除缓存中所有的数据：见清空缓存。

RAM 缓存

Traffic Server 维护着一个包含热点对象的微型 RAM 缓存。这个 RAM 缓存在尽可能快地服务大部分热点对象的同时也减少了磁盘负载，特别是在一些流量的高峰。可以根据需要来配置 RAM 缓存的大小；更多的细节见改变 RAM 缓存的大小。

改变 RAM 缓存大小

Traffic Server 为快速检索热点小对象提供了一个专用的 RAM 缓存。默认的 RAM 缓存大小是根据配置的缓存分区的大小来计算的。如果根据协议或者 host 对缓存进行了分区，每个分区的 RAM 缓存大小是和分区的大小成比例的。

可以为了提高缓存命中性能增加 RAM 缓存的大小。然而，如果增加了 RAM 缓存的大小却观察到性能的下降（比如延迟的增加），可能是操作系统需要为网络资源提供更多的内存。在这种情况下，将 RAM 缓存的大小改回以前的值。

改变 RAM 缓存的大小：

1. 停止 Traffic Server。
2. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
3. 编辑下面的变量：

变量	描述
proxy.config.cache.ram_cache.size	设置这个变量来指定 RAM 缓存的大小。默认值为-1，表示 1GB 磁盘的 RAM 缓存大小为 1MB。

4. 保存并关闭 records.config 文件
5. 重启 Traffic Server。如果增加的 RAM 缓存大于等于 1GB，用 start_traffic_server 命令来重启（见启动 Traffic Server）。

改变缓存容量

可以在不清除已有内容的情况下增加或减少分配给缓存的磁盘空间总量。要核对缓存的大小（字节），直接输入 traffic_line -r proxy.process.cache.bytes_total 命令。

增加缓存容量

要在已有的磁盘上增加分配给缓存的磁盘空间总量或者在 Traffic Server 节点中添加新的磁盘，需要下面几步：

1. 停止 Traffic Server。
2. 如果需要，添加硬件。
3. 编辑 Traffic Server 的 storage.config 文件：在已有的磁盘上增加分配给缓存的磁盘空间总量，或者为新的硬件添加描述（见 storage.config）。
4. 如果添加了新磁盘，必须编辑/etc/rc.d/init.d/traffic_server 文件来添加新磁盘的绑定。添加新磁盘绑定的指令位于 Traffic Server 的 storage.config 文件中。
5. 重启 Traffic Server。

减小缓存容量

要在已有的磁盘上减少分配给缓存的磁盘空间总量或者在 Traffic Server 节点中移除磁盘，需要下面几步：

1. 停止 Traffic Server。
2. 如果需要，移除硬件。
3. 编辑 Traffic Server 的 storage.config 文件：在已有的磁盘上减少分配给缓存的磁盘空间总量，或者删除硬件的相关参考（见 storage.config）。
4. 如果移除了磁盘，必须编辑/etc/rc.d/init.d/traffic_server 文件来删除磁盘的绑定。
5. 重启 Traffic Server。

重要：在 storage.config 文件中，一个格式化或原始磁盘不得小于 128MB。

缓存分区

可以通过为特殊协议创建不同大小的分区来更加高效地管理缓存空间以及抑制磁盘的使用。也可以进一步配置这些分区存储来自特殊源服务器或域名的数据。一个集群所有节点的分区配置必须相同。

为特殊的协议创建缓存分区

可以为缓存创建不同大小的独立分区来按协议存储数据。这可以保证一个特殊的协议总有一个固定大小的磁盘空间可用。Traffic Server 当前为 HTTP 对象支持 http 分区。

根据协议的分割缓存：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 partition.config 文件。
2. 在文件中为每个要创建的分区输入一行（见 partition.config）。

3. 保存并关闭 partition.config 文件。
4. 重启 Traffic Server。

改变分区的大小和协议

可以配置缓存区分基于某个协议，也可以随时改变配置。在改变之前注意：

- 在改变缓存分区的大小以及对应的协议之前，必须停止 Traffic Server。
- 当增加分区的大小，分区原有的内容将不会删除。然后，当减小分区的大小，分区的内容将被删除。
- 当改变了分区的编号，分区将被删除并重新创建，但是对应的大小和协议类型保持不变。
- 当给 Traffic Server 节点加入了新的磁盘，分区的大小将会按指定的百分比成比例的增加。
- 大量分区大小的改动将导致磁盘碎片，这将影响性能和命中率。在改变缓存分区大小之前，必须清空缓存（将清空缓存）。

根据源服务器或域名来分隔缓存

在根据大小和协议分隔缓存之后，可以将这些分区分配给特殊的源服务器或者域名。可以分配一个分区给单个或多个源服务器。然而，一个分区被分配给多个源服务器，就不能保证每个源服务器一定有可用的空间。内容根据其热度存储在分区中。除了分配分区给特定的源服务器或域名，必须配置一个通用的分区来存储所有没有在列表中的源服务器或域名的对象。如果一个特定源服务器或域名的分区不可用，通用分区同样会被使用。如果没有分配通用分区，Traffic Server 将会运行在单纯代理模式。

注意：在分配分区给特定的 hosts 或域名之前，不需要停止 Traffic Server。然而，这类配置比较耗时，同时也会产生内存使用的峰值。因此，最好在流量比较低的时候配合分区的分配。

根据源服务器或域名来分隔缓存：

1. 根据大小和协议来配置缓存分区，再为特定协议创建缓存分区描述。
2. 为每个 host 或域名创建一个基于协议的独立分区，当然，通用分区用来存储不属于这些源服务器或域名的内容。
3. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 hosting.config 文件。
4. 在文件中输入一行来为每个源服务器或域名分配分区（见 hosting.config）。
5. 为源服务器或域名没有在文件中的内容分配一个通用分区。如果一个特定源服务器或域名的分区不可用，通用分区同样会被使用。（见 hosting.config）。
6. 保存并关闭 hosting.config 文件。
7. 定位到 Traffic Server 的 bin 目录下。
8. 运行 traffic_line –x 命令来应用配置文件的变更。

配置缓存对象的大小限制

默认情况下，Traffic Server 允许任何大小的对象被缓存。可以改变默认情况，通过下面的步骤来指定缓存中对象的大小限制。

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.cache.max_doc_size	设置这个变量来指定缓存中对象允许的最大大小（字节）。输入 0 表示没有大小限制。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –x 命令来应用配置文件的变更。

清空缓存

清空缓存操作将会删除整个缓存的数据，包括 host 数据库的数据。在完成某些缓存配置任务（比如分区）前，应该清空缓存。不能在 Traffic Server 运行的时候清空缓存。

清空缓存：

1. 停止 Traffic Server（见停止 Traffic Server）。
2. 输入下面的命令来清空缓存：traffic_server –Cclear 。clear 命令删除所有 object store 和 host 数据库中的数据。Traffic Server 将不会提示是否确认删除操作。
3. 重启 Traffic Server（见启动 Traffic Server）。

删除缓存中的某个对象

Traffic Server 接受一般的 HTTP PURGE 请求来删除缓存中的某个对象。如果对象可以在缓存中找到并成功删除，Traffic Server 返回 200 OK 的 HTTP 应答消息；否则，返回一个 404 File Not Found 消息。

在下面的例子中，Traffic Server 运行在主机 example.com 上，从缓存中删除 remov_me.jpg 图片：

```
$ curl -X PURGE -v "http://example.com/remove_me.jpg"

* About to connect() to example.com port 80 (#0)
* Trying 192.0.32.11... connected
* Connected to example.com (192.0.32.11) port 80 (#0)

> PURGE /remove_me.jpg HTTP/1.1
> User-Agent: curl/7.19.7
> Host: example.com
```

> Accept: */*
>
< HTTP/1.1 200 Ok
< Date: Thu, 08 Jan 2010 20:32:07 GMT
< Connection: keep-alive

下次 Traffic Server 收到已删除对象的请求，它将从源服务器检索该对象（在 Traffic Server 缓存已经被删除）。

注意：上面的步骤只是从一个指定的 Traffic Server 缓存中删除了一个对象。如果该对象在中间层缓存或浏览器缓存，用户仍然能看到它。

检查缓存

Traffic Server 提供了一个缓存检查实用工具来查看、删除和失效缓存中的 URL（只是 HTTP）。缓存检查器是一个功能强大的工具，它可以删除缓存中所有的对象；因此，确保只有被授权的管理员才允许访问这个工具。通过 mgmt.allow.config 文件来控制主机的访问（将控制访问 Traffic Manager 的主机）。

访问缓存检查工具

通过下面的步骤来访问缓存检查工具：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 在文件尾添加如下的变量：CONFIG proxy.config.http_ui_enabled INT 1
3. 要在反向代理模式下访问缓存检查工具，必须向 remap.config 文件添加一条重新映射规则来显示该 URL。比如：
   map http://yourhost.com/myCI http://{cache} @action=allow @src_ip=corp_internal_address
4. 在 Traffic Server 的 bin 目录，输入 traffic_line –x 来重新读取配置文件。
5. 打开浏览器并配置其使用 Traffic Server 作为代理服务器。输入下面的 URL：http://{cache}
6. 缓存页面打开（见使用缓存页面）。

使用缓存页面

缓存页面提供了若干种方式来查看和删除缓存的内容：

- 点击 Lookup url 来在缓存中寻找一个特殊的 URL。当 Traffic Server 在缓存中找到这个 URL，它会展示和这个 URL 相关对象的细节（比如头部长度和 alternates 个数）。在展示页面，可以从缓存删除这个 URL。
- 点击 Delete url 来从缓存删除一个特定的 URL 或一列 URL。Traffic Server 会显示删除是否成功。
- 点击 Regex lookup 来查询和正则表达式匹配的 URL。通过展示的页面，可以删除列出的 URL。

比如，输入 http://www.dianes.com/.*\.html$ 来搜索所有以http://www.dianes.com 为前缀以及以html为结尾的URL。

- 点击 Regex delete 来删除和一个特定正则表达式匹配的所有 URL。比如，输入http://.*\.html$ 来删除所有以html结尾的HTTP URL。
- 点击 Regex invalidate 可以使一个特定正则表达式匹配的所有 url 无效。当无效一个 URL 时，Traffic Server 在缓存中标记和这个 URL 对应的对象为无效。随后，在 Traffic Server 从缓存服务这个 URL 之前，它首先向源服务器检测该对象是否可以重新生效。

注意：在任意时刻，只有一个管理员可以从缓存页面删除和失效缓存。多为管理员在同一时间一起修改会导致不可预料的结果。

流量监控

Traffic Server 提供了若干选项来监控系统性能和分析网络流量。

Traffic Server 监控工具

Traffic Server 提供了下面的工具来监控系统性能和分析网络流量：

- Traffic Server 在检测到失败情况的信号时，通过报警来触发其发送邮件；见和 Traffic Manager 报警一起工作。
- Traffic Line 命令行接口提供了一种替代的方式来查看 Traffic Server 的性能以及网络流量信息；见通过 Traffic Line 查看统计信息。
- Traffic Shell 命令工具提供了另一种替代的方式来查看 Traffic Server 的性能以及网络流量信息；见启动 Traffic Shell。

和 Traffic Manager 报警一起工作

当检测到一个错误，TrafficServer 会发一个报警。比如，给事件日志分配的空间快满或 Traffic Server 不能写一个配置文件。

配置 Traffic Server 发报警邮件

配置 Traffic Server 在一个报警发生时给特定的地址发送一份邮件需要的步骤如下：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 设置 proxy.config.alarm_email 变量来指定报警邮件发送的地址。
3. 保存并关闭 records.config 文件。
4. 定位到 Traffic Server 的 bin 目录。
5. 运行 traffic_line –x 命令来使配置文件生效。

为报警使用一个脚本文件

报警信息内建在 Traffic Server 中，不能被改变。然而，可以写一个脚本在一个报警发生时执行某些行为。Traffic Server 提供了一个位于 bin 目录下的名为 example_alarm_bin.sh 的脚步文件样例；可以根据需要修改这个脚步。

通过 Traffic Line 查看统计信息

可以通过 Traffic Line 命令行接口来查看 Traffic Server 性能和 web 流量的统计信息。除了查看统计信息，还可以配置、停止和重启 Traffic Server 系统。更多信息见使用 Traffic Line 配置 Traffic Server 和 Traffic Line 命令行。可以通过指定统计信息相关的变量来查看 Traffic Server 节点或集群的特定信息。

查看统计信息：

1. 以管理员身份登录到 Traffic Server 节点；定位到 Traffic Server 的 bin 目录。
2. 输入下面的命令：traffic_line –r variable
   这里的 variable 是要查看的信息对应的变量。可以指定的变量列表见 Traffic Line Variables。
   比如，下面的命令展示 Traffic Server 的文档命中率：
   traffic_line -r proxy.node.http.cache_hit_ratio
   如果 Traffic Server 没有在系统 path 中，预先为 Traffic Line 命令加上./（比如：./traffic_line -r variable）。

配置 Traffic Server

Traffic Server 提供了若干选项来配置系统。

使用 Traffic Line 配置 Traffic Server

Traffic Line 可以通过命令行接口快速和方便地改变 Traffic Server 的配置。当然，也可以使用 Traffic Shell 来配置 Traffic Server。

在 Traffic Line 中查看或改变配置选项：

1. 以管理员身份登录到 Traffic Server 节点；定位到 Traffic Server 的 bin 目录。
2. 输入下面的命令来查看配置设置：traffic_line –r var
   这里的 var 变量和配置选项相关联。变量列表见配置变量。
3. 输入下面的命令来改变配置设置的值：traffic_line –s var –v value
   这里的 var 变量和配置选项相关联，value 是想要使用的值。变量列表见配置变量。
   如果 Traffic Server 没有在系统 path 中，预先为 Traffic Line 命令加上./（比如：./traffic_line -r variable）。

使用配置文件来配置 Traffic Server

使用 Traffic Line 或 Traffic Shell，可以通过手动编辑位于 Traffic Server config 目录下的 records.config 文件中的变量来改变 Traffic Server 的配置选项。要同时编辑多个变量，可以通过文本编辑器（比如 vi 或 emacs）打开文件并改变变量的值。当修改了 records.config 文件，Traffic Server 必须重新读取配置文件。在 Traffic Server bin 目录下，输入 Traffic Line 命令 traffic_line –x。或许需要重启 Traffic Server 来应用改变的配置文件。

图 7 是 records.config 文件的一个样本：

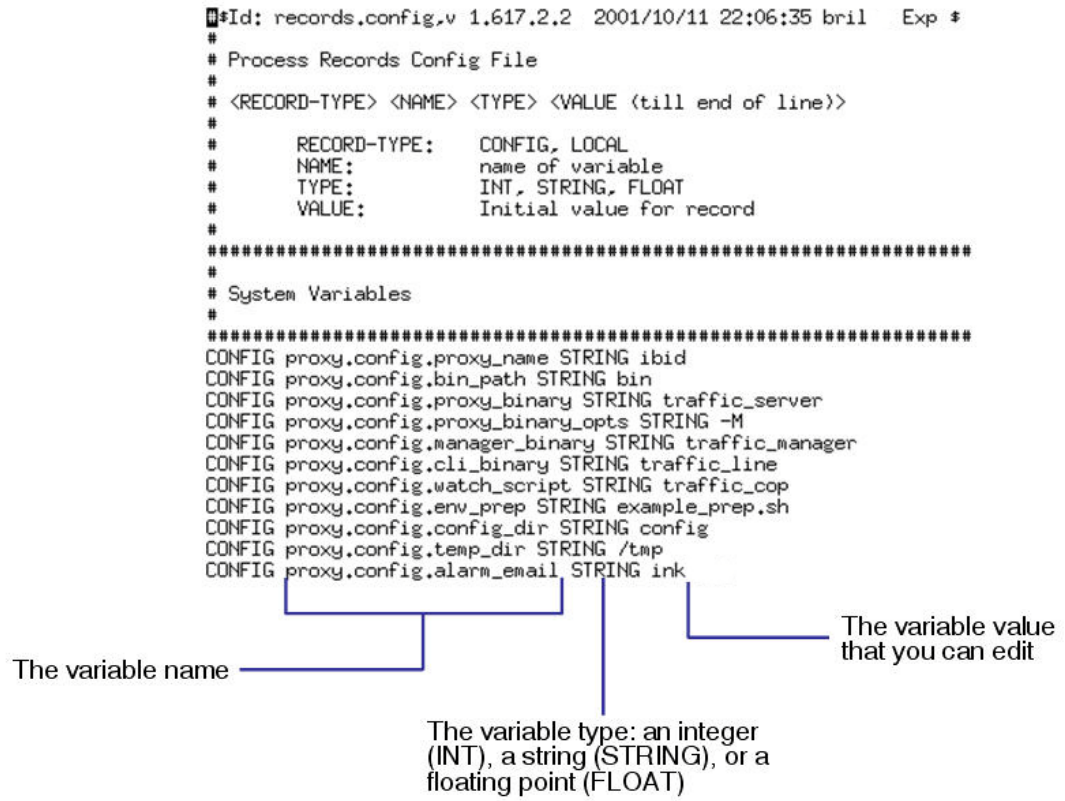

图 7. records.config 文件样本

除了 records.config 文件，Traffic Server 还提供了其它的配置文件来配置特定的功能。所有可以手动编辑的配置文件将在配置文件中描述。

安全选项

Traffic Server 提供了若干个安全功能。

控制客户端访问代理缓存

可以通过编辑配置文件来配置 Traffic Server 只允许特定的客户端访问代理缓存。

指定允许访问代理缓存的客户端：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 ip_allow.config 文件。
2. 为每个允许访问 Traffic Server 的 IP 地址或 IP 地址段在文件中添加一行（见 ip_allow.config）。
3. 保存并关闭 records.config 文件。
4. 定位到 Traffic Server 的 bin 目录。
5. 运行 traffic_line –x 命令来使配置文件生效。

通过 SSL 控制访问

通过限制访问 Traffic Server 可以确保只有可识别的用户才可以改变配置选项以及查看网络流量统计信息。

使用 SSL 来安全管理

Traffic Server 支持安全套接层（SSL）协议来为远程管理监控和配置提供保护。SSL 通过证书来为连接的两端提供安全证明，并通过加密来提供机密性。

要使用 SSL，必须完成下面几步：

- 获得一个 SSL 证书
- 开启 SSL

获得一个 SSL 证书

SSL证书是一个必须安装在Traffic Server config目录下的文本文件。可以重命名证书为默认的文件名private_key.pem，或者在配置文件中指定证书的名称（根据开启SSL的步骤）。

开启 SSL

在获得 SSL 证书之后，可以通过手动编辑配置文件来开启 SSL。步骤如下：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.admin.use_ssl	设置这个变量为 1 来开启 SSL。
proxy.config.admin.ssl_cert_file	设置这个变量来指定 SSL 证书的文件名。只有在证书没有使用默认的文件名 private_key.pem 时才需要配置此项。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录

5. 运行 traffic_line –x 命令来应用配置文件的变更。

配置 DNS 服务器选择（分离 DNS）

分离 DNS 选项可以根据安全需求的指示配置 Traffic Server 使用多个 DNS 服务器。比如，可以配置 Traffic Server 使用一组 DNS 服务器来解析内网中的主机名，而使用防火墙外面的 DNS 服务器来解析 Internet 中主机。这在维护内网安全的同时，还可以继续为外网站点提供直接访问。

要配置分离 DNS，必须完成下面的步骤：

- 指定规则来完成基于目的域名、目的主机或者 URL 正则表达式的的 DNS 服务器选择。
- 开启分离 DNS 选项。

配置分离 DNS：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 splitdns.config 文件。
2. 向 splitdns.config 文件添加规则。关于 splitdns.config 文件格式的更多信息见 splitdns.config。
3. 保存并关闭 splitdns.config
4. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
5. 编辑下面的变量：

变量	描述
proxy.process.dns.splitDNS.enabled	设置这个变量为 1 来开启分离 DNS。
proxy.config.dns.splitdns.def_domain	设置这个变量来为分离 DNS 请求指定默认的域名。在决定使用哪个 DNS 服务器之前，Traffic Server 会自动将这个值附加给没有包含域名的主机名。

6. 保存并关闭 records.config 文件
7. 定位到 Traffic Server 的 bin 目录
8. 运行 traffic_line –x 命令来应用配置文件的变更。

使用 SSL 终端

Traffic Server 的 SSL 终端选项可以在反向代理模式下，在客户端和 Traffic Server 或者 Traffic Server 和源服务器之间建立安全连接。

下面各节描述了如何开启和配置 SSL 终端选项。

- 为客户端和Traffic Server开启和配置SSL 终端：客户端和Traffic Server连接。
- 为Traffic Server和源服务器开启和配置SSL 终端：Traffic Server和源服务器连接。
- 为客户端和 Traffic Server 以及 Traffic Server 和源服务器都开启和配置 SSL 终端。

如果在Traffic Server系统中安装了SSL加速卡，必须完成附加的配置步骤，见

配置Traffic Server使用SSL加速卡。

客户端和 Traffic Server 连接

图 8 说明了在 SSL 终端选项只开启和配置了客户端和 Traffic Server 连接时客户端和 Traffic Server（以及 Traffic Server 和源服务器）之间的通信。

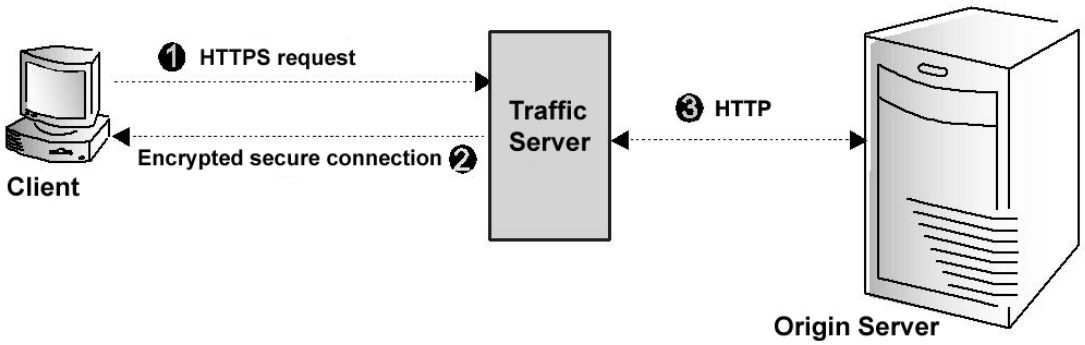

图 8. 客户端和 Traffic Server 使用 SSL 终端通信

上图描述了如下步骤：

1. 客户端发送一个 HTTPS 内容请求。Traffic Server 接收请求并完成 SSL 握手来鉴别客户端（取决于鉴别选项的配置）并决定接下来使用的加密算法。如果客户端的访问被允许，Traffic Server 在其缓存中查找被请求的内容。
2. 如果请求在缓存命中且内容有效，Traffic Server 加密内容并发送给客户端。客户端解密内容（使用握手时决定的方法）并展示它。
3. 如果请求在缓存未命中或内容过期，Traffic Server 通过 HTTP 和源服务器通信并获取明文的内容信息。Traffic Server 将明文内容信息保存到缓存，同时将内容加密并发送给客户端。客户端解密并展示内容。

要配置 Traffic Server 为客户端和 Traffic Server 之间的连接使用 SSL 终端选项，必须完成如下步骤：

- 从公认的认证授权机构（比如 VeriSign）获取并安装 SSL 服务证书。SSL 服务证书包含使客户端鉴别 Traffic Server 以及交互加密密钥的信息。
- 配置 SSL 终端选项：
  - 开启 SSL 终端选项。
  - 设置 SSL 通信使用的端口号。
  - 指定服务证书的文件名及位置。
  - （可选）配置客户端使用的证书。

  客户端证书位于客户端系统。如果配置 Traffic Server 需要客户端证书，Traffic Server 必须在 SSL 握手阶段审核客户端的证书。如果配置 Traffic Server 不需要客户端证书，将由其它已经设置的 Traffic Server 选项（比如 ip_allow.config 文件中的规则）来管理对 Traffic Server 的访问。
  - 指定 Traffic Server 私钥的文件名和位置（如果私钥没有位于服务证书文件中）。

  Traffic Server 使用私钥来解密 SSL 握手阶段协商加密密钥的会话。私钥必须保存且防止被盗用。

■ （可选）配置使用的认证机构（CA）。

CA 通过审核请求证书人员的身份来增强安全性。

为客户端和 Traffic Server 之间的连接配置 SSL 终端：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.ssl.enabled	设置这个变量为 1 来开启 SSL 终端选项。
proxy.config.ssl.server_port	设置这个变量来指定 SSL 通信的端口号。默认端口为 443。
proxy.config.ssl.client.certification_level	设置这个变量为下面值之一： 0 – 不需要客户端证书。Traffic Server 在 SSL 握手阶段不审核客户端证书。访问 Traffic Server 取决于 Traffic Server 的其他配置选项（比如访问控制列表）。 1 – 客户端证书是可选的。如果客户端有证书，证书将被核实。如果客户端没有证书，客户端仍而被允许访问 Traffic Server，除非访问被 Traffic Server 的其它配置选项拒绝。 2 – 需要客户端证书。客户端在 SSL 握手阶段必须被认证；没有证书的客户端将不能访问 Traffic Server。
proxy.config.ssl.server.cert.filename	设置这个变量来指定 Traffic Server SSL 服务证书的文件名。 Traffic Server 提供了一个名为 server.pem 的演示服务证书 – 使用这个证书来验证 SSL 功能是可以工作的。 如果使用多个服务证书，设置这个变量来指定默认文件名。
proxy.config.ssl.server.cert.path	设置这个变量来指定 Traffic Server 服务证书的位置。默认的目录是 Traffic Server 的 config 目录。
proxy.config.ssl.server.private_key.filename	设置这个变量来指定 Traffic Server 私钥的文件名。只有当私钥没有位于 Traffic Server SSL 服务证书文件中时才需要改变这个变量。
proxy.config.ssl.server.private_key.path	设置这个变量来指定 Traffic Server 私钥的位置。只有当私钥没有位于 Traffic Server SSL 服务证书文件中时才需要改变这个变量。

proxy.config.ssl.CA.cert.filename	指定审核客户端证书的认证机构文件名。默认值为 NULL。
proxy.config.ssl.CA.cert.path	指定审核客户端证书的认证机构文件名的位置。默认值为 NULL。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –L 命令来在本节点重启 Traffic Server 或者 traffic_line –M 命令来在一个集群的所有节点上重启 Traffic Server。

Traffic Server 和源服务器的连接

图 9 说明了在 SSL 终端选项开启和配置了 Traffic Server 和源服务器连接时 Traffic Server 和源服务器之间的通信。

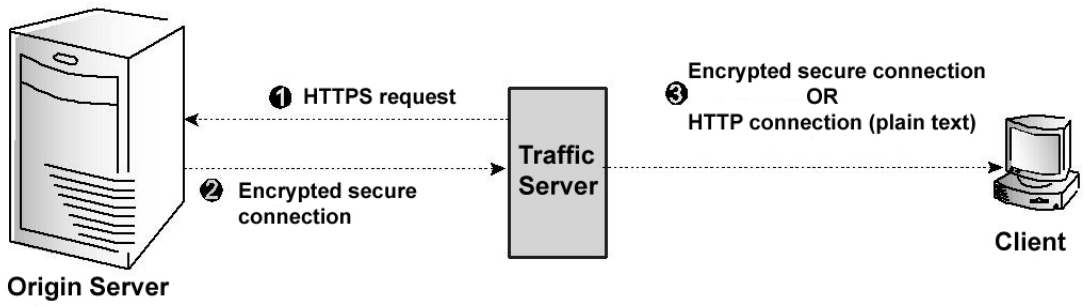

图 9. Traffic Server 和源服务器使用 SSL 终端通信

上图描述了如下步骤：

1. 如果客户端请求在缓存未命中或已过期，Traffic Server 向源服务器发送一个 HTTPS 内容请求。源服务器接收请求并完成 SSL 握手来鉴别 Traffic Server 并决定接下来使用的加密算法。
2. 如果 Traffic Server 被允许访问，源服务器将内容加密并发送给 Traffic Server，Traffic Server 使用握手阶段决定的方式来解密该内容。内容的明文被保存在缓存。
3. 如果 SSL 终端开启了客户端和 Traffic Server 之间连接，Traffic Server 将再次加密内容并通过 HTTPS 发送给客户端，客户端解密并展示内容。如果 SSL 终端没有开启了客户端和 Traffic Server 之间连接，Traffic Server 将通过 HTTP 发送明文内容给客户端。

要配置 Traffic Server 为 Traffic Server 和源服务器之间的连接使用 SSL 终端选项，必须完成如下步骤：

- 从公认的认证授权机构（比如 VeriSign）获取并安装 SSL 客户端证书。SSL 客户端证书包含允许源服务器鉴别 Traffic Server 的信息（客户端证书是可选的）。
- 配置 SSL 终端选项：
  - 开启 SSL 终端选项。
  - 设置 SSL 通信使用的端口号。
  - 指定 SSL 客户端证书的文件名及位置（如果选择使用客户端证书）。

- （可选）配置客户端使用的证书。
- 指定 Traffic Server 私钥的文件名和位置（如果私钥没有位于客户端证书文件中）。

Traffic Server 使用私钥来解密 SSL 握手阶段协商加密密钥的会话。私钥必须保存且防止被盗用。

- （可选）配置使用的认证机构（CA）。

CA 允许 Traffic Server 作为客户端来审核通信服务器的身份，因此可以用来交换加密密钥。

为客户端和 Traffic Server 之间的连接配置 SSL 终端：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.config.ssl.auth.enabled	设置这个变量为 1 来开启 SSL 终端选项。
proxy.config.ssl.server_port	设置这个变量来指定 SSL 通信的端口号。默认端口为 443。
proxy.config.ssl.client.verify.server	设置这个变量为 1 来要求 Traffic Server 通过认证机构来审核源服务器的证书。
proxy.config.ssl.client.cert.filename	如果已经在 Traffic Server 上安装了 SSL 客户端证书，设置这个变量来指定客户端证书的文件名。
proxy.config.ssl.client.cert.path	如果已经在 Traffic Server 上安装了 SSL 客户端证书，设置这个变量来指定客户端证书的位置。默认的位置是 Traffic Server 的 config 目录。
proxy.config.ssl.client.private_key.filename	设置这个变量来指定 Traffic Server 私钥的文件名。只有当私钥没有位于 Traffic Server SSL 客户端证书文件中时才需要改变这个变量。
proxy.config.ssl.client.private_key.path	设置这个变量来指定 Traffic Server 私钥的位置。只有当私钥没有位于 SSL 客户端证书文件中时才需要改变这个变量。
proxy.config.ssl.client.CA.cert.filename	指定审核源服务器证书的认证机构文件名。默认值为 NULL。
proxy.config.ssl.client.CA.cert.path	指定审核源服务器证书的认证机构文件名的位置。默认值为 NULL。

3. 保存并关闭 records.config 文件
4. 定位到 Traffic Server 的 bin 目录
5. 运行 traffic_line –L 命令来在本节点重启 Traffic Server 或者 traffic_line –M 命令来在一个集群的所有节点上重启 Traffic Server。

配置 Traffic Server 使用 SSL 加速卡

可以在 Traffic Server 机器上安装一个 SSL 加速卡来加速 Traffic Server 对请求的处理。Traffic Server 支持 Cavium 加速卡。如果选择不使用 SSL 加速卡，可以使用常用的 SSL 库；如果安装了 Cavium 卡，可以通过卡的厂商来提供和支持可使用的库。

配置 Traffic Server 使用 SSL 加速卡：

1. 在文本编辑器中打开位于 Traffic Server 的 config 目录下的 records.config 文件。
2. 编辑下面的变量：

变量	描述
proxy.confg.ssl.accelerator_required	设置这个变量来指定是否需要加速卡。 0 – 不需要 1 – 需要加速卡，Traffic Server 只有发现加速卡才开启 SSL 服务。 2 – 需要加速卡，Traffic Server 只有发现加速卡才能启动。
proxy.confg.ssl.accelerator.type	指定 Traffic Server 机器上是否需要安装 Cavium SSL 加速卡： 0 = none。Traffic Server 机器上没有安装 SSL 加速卡，Traffic Server 机器上的 CPU 决定每秒处理的请求数。 1 = Traffic Server 需要加速卡。

3. 保存并关闭 records.config 文件。
4. 定位到 Traffic Server 的 bin 目录。
5. 运行 traffic_line –L 命令来在本节点重启 Traffic Server 或者 traffic_line –M 命令来在一个集群的所有节点上重启 Traffic Server。

后记

本文档为本人在工作之余翻译而成，现整理成章，放在网上和大家分享。由于本人水平有限，文中难免会有纰漏；如果大家发现，可以直接发邮件（qianshi@taobao.com）给我，大家一起来完善。本文取于开源又予于开源，亦当遵开源之法。大家可以自由转载，但不要用于任何商业行为。最后感谢永豪在 TS 方面给我的指导；感谢女友帮我校正错别字，理顺语句。

关于文档后续的更新以及 TS 更多的信息，请关注本人技术 blog www.flyinfo.net。

Traffic Server 技术 blog 开篇词

TCP/IP 和 HTTP 是互联网的两大基石，支撑着数以亿计的 Web 站点和应用。高性能始终是衡量 web server 的重要指标，而代理和缓存作为两个重要的技术，在提高 web server 的并发和吞吐方面有着不可替代的作用。

开源造就了互联网，开源项目更是遍布互联网的每个角落。Squid、Vanish、nginx、HAProxy 都是 HTTP 代理和缓存方面的神兵利器，各自在这个领域摧城拔寨、攻城略地，早已都是闻名遐迩、享誉江湖。然而，对性能的追求是无止境的，2009 年 Yahoo 又贡献出了一款神器 –Traffic Server。

Traffic Server 始于 Inktomi，后被 Yahoo 收于门下，在束之高阁几年后，有缘人发现其性能奇佳、潜力无限，随后在 Yahoo 内部使用，在 CDN 中更是堪当大任。鉴于 Hadoop 开源的成功，Yahoo 亦将其贡献给了 Appache 作为 TLP。

Traffic Server 最强大的是其对多处理机的支持和完全异步的事件处理模型，加之对集群的支持和完备的供第三方开发的 Plugin，是 CDN 建设的首选。然而国内还很少有人知道 Traffic Server，使用者更是寥寥无几。但好东西终究要大家分享，这也正是开 Traffic Server 技术 blog（www.flyinfo.net）的本意。本 blog 会发表 Traffic Server 相关的技术和文档，皆为本人原创或翻译，大家可以随意转载，但请不要用于任何形式的商业行为。

关于我：本人初出茅庐，现就职于淘宝北京 CDN 团队，花名：千石。对高性能 web server 有着浓厚的兴趣，热衷技术，喜欢开源。