

Fikker 设计与实现 V3

2011-11-07

目录:

1. [系统说明](#)
2. [全非阻塞 \(non-block\) 网络设计](#)
 - 2.1. [全非阻塞设计说明](#)
 - 2.2. [网络负载均衡设计说明](#)
 - 2.3. [图例说明](#)
3. [缓存设计](#)
 - 3.1. [缓存说明](#)
 - 3.2. [智能缓存设计](#)
 - 3.3. [强制缓存设计](#)
 - 3.3.1. [公共缓存](#)
 - 3.3.2. [会员缓存](#)
 - 3.3.3. [游客缓存](#)
 - 3.3.4. [会员缓存设计背景](#)
 - 3.3.5. [图例说明](#)
 - 3.4. [拒绝缓存设计](#)
 - 3.5. [清理缓存设计](#)
 - 3.6. [缓存优先级设计](#)
 - 3.7. [页面匹配规则设计](#)
 - 3.8. [缓存页面淘汰算法设计](#)
 - 3.8.1. [缓存页面淘汰说明](#)
 - 3.8.2. [缓存页面淘汰优先级](#)
 - 3.8.3. [同一优先级时淘汰规则](#)
 - 3.9. [页面压缩 gzip 设计](#)
 - 3.9.1. [说明](#)
 - 3.9.2. [HTTP 头中有关 gzip 压缩的字段](#)
 - 3.9.3. [页面压缩的实现](#)
4. [URL 转向设计](#)
 - 4.1. [URL 转向简介](#)
 - 4.2. [转向逻辑](#)
 - 4.3. [简单举例](#)
5. [防盗链设计](#)
 - 5.1. [防盗链说明](#)
6. [黑名单设计](#)
 - 6.1. [黑名单说明](#)
7. [代理设计](#)
 - 7.1. [代理介绍](#)
 - 7.2. [负载均衡策略](#)
 - 7.3. [负载均衡图例](#)
8. [流量统计设计](#)
 - 8.1. [流量统计说明](#)
 - 8.2. [总量统计](#)
 - 8.3. [分量统计](#)

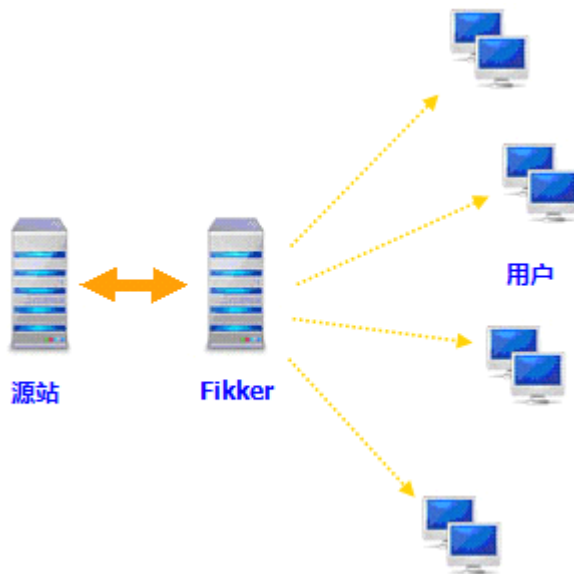
- 9. [实时监控设计](#)
 - 9.1. [实时监控说明](#)
- 10. [Windows 和 Linux 兼容性设计](#)
 - 10.1. [绿色安装包](#)
 - 10.2. [Fikker 的运行权限](#)
 - 10.3. [配置文件兼容性](#)
- 11. [关于 Fikker 限制的说明](#)
 - 11.1. [内存限制](#)
 - 11.2. [连接数限制](#)
 - 11.3. [HTTP 头尺寸限制](#)
 - 11.4. [缓存页面尺寸限制](#)
 - 11.5. [操作系统限制](#)
- 12. [Fikker 与 HTTP 头](#)
- 13. [关于 Fikker 返回错误页面的说明](#)
 - 13.1. [400 Bad Request](#)
 - 13.2. [403 Forbidden](#)
 - 13.3. [409 Conflict](#)
 - 13.4. [502 Bad Gateway](#)
 - 13.5. [503 Service Temporarily Unavailable](#)

1. Fikker 说明

Fikker 是一款跨平台（支持 Windows 和 Linux）的专业级网站加速服务器软件，其一是：通过将指定的动态页面进行缓存，用户的访问页面可直接从缓存中直接获取，节省网站生成页面的时间，从根本上减轻数据库压力，极大提升网站的响应速度；其二是：通过对缓存页面的 gzip 压缩，减少传输时间提升传输效率来实现加速。Fikker 通过对网络的全非阻塞化（non-block）处理，对多核心多线程充分高效的并行化处理，缓存的全内存化处理，达到系统最大化的处理性能。

Fikker 网络实现了全部非阻塞化（non-block）处理，包括 Fikker 接收数据非阻塞，发送数据非阻塞，域名解析非阻塞。全面支持 Linux 2.6.x 内核 epoll 消息机制。单个端口支撑 60000 个并发连接，极少到可忽略不计的 CPU 占用。

Fikker 是源站（网站）前置机，是放在源站（网站）前面的服务器。用户使用浏览器访问的时候，用户的访问请求首先会被 Fikker 接收并处理。如果命中缓存，就会返回浏览器已缓存页面，如果没有命中缓存页面或缓存页面已经超时，Fikker 就会将用户的访问请求转发到源站（网站），从源站获取最新的页面返回给用户，同时还会根据缓存规则判断是否允许缓存此页面，如果缓存规则允许缓存此页面，Fikker 会将此页面使用 gzip 压缩后缓存在内存中，其他用户再访问相同页面时候，就会将已缓存页面立即返回给用户。此过程除对日志进行必要记录外，页面缓存过程不读写任何硬盘。



Fikker 还同时提供 URL 转向功能（包括‘伪静态’功能实现），防盗链功能，代理功能，负载均衡，黑名单，流量统计和实时监控的功能。对网站提供了优化、监控、缓存、负载均衡，实时流量监控，隔离安全和黑名单，伪静态规则（SEO）等一站式解决方案，是站长们必不可少的‘看家武器’。

Fikker 让您的网站飞起来。

2. 全非阻塞（non-block）网络设计

2.1. 全非阻塞设计：

网络非阻塞设计是 Fikker 整个架构的重点设计，也是网络设计最核心的设计，要求使用几个有限的线程即可承载上万并发连接，并且每一个服务请求都不会被阻塞，立即响应。例如：用户发起连接请求时，Fikker 在工作中能立即接受（accept）到（而不是线程一直阻塞直到 accept 函数返回），并且立即进入服务队列；同样子的道理，Fikker 向源站（网站）发起连接请求时，在同一个线程中，要求域名解析时，线程是非阻塞的（而不是等待域名解析好后函数返回，这样子整合线程都会被阻塞），解析好的域名和 IP 地址通过回调函数的方式通知本线程，建立连接（connect）时，也是线程非阻塞的，建立好的连接通过回调的方式通知给应用程序，这样子当拥有上万连接高效服务时，只需要几个线程即可满足需求，而且每个请求都可以做到立即响应。Linux 下面 epoll 机制提供一种高效的网络设计，理论上最少只需要 1 个线程即可满足不限并发连接的需求，通常情况下，为了配合 CPU 的并行度（多 CPU 负载均衡），Linux 下网络线程的数量可以设定为 CPU 核心的数量。Windows 下 1 个线程最大可满足 1000 个并发连接的即时处理（20 个线程最大可并发处理 2 万个并发连接，单个端口最大服务 6 万个并发连接）。

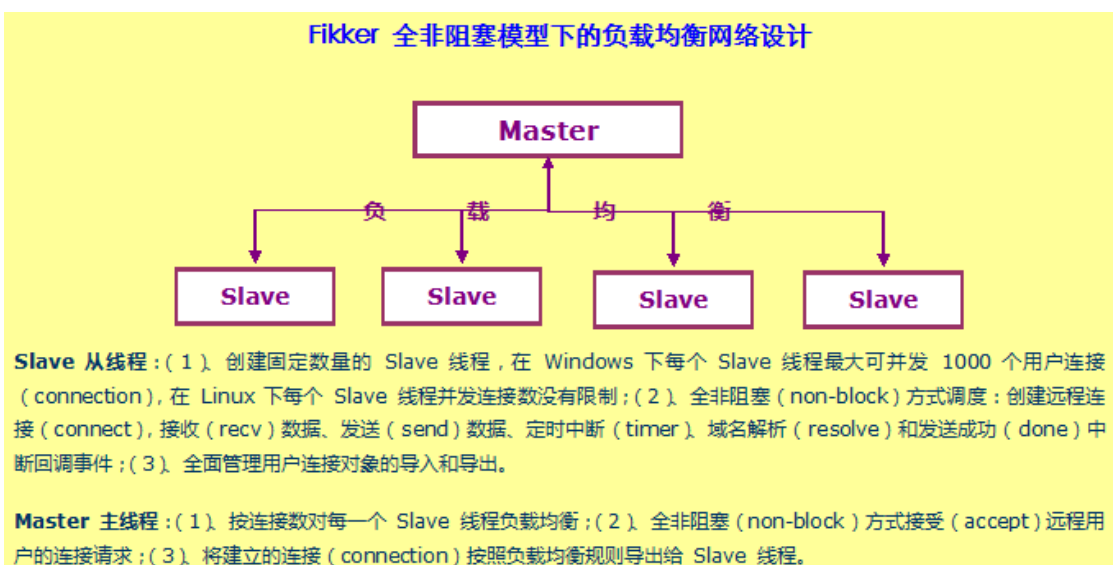
2.2. 网络负载均衡设计：

同时在几个线程中高效处理上万并发时，需要调节线程之间的连接数，使 CPU 处理功效能被均衡负载，这就要求网络连接的数据发送和数据接收能够在线程之间迁移，线程之间负载按照连接数均衡。

Slave 从线程：（1）、创建固定数量的 Slave 线程，在 Windows 下每个 Slave 线程最大可并发 1000 个用户连接（connection），在 Linux 下每个 Slave 线程并发连接数没有限制；（2）、全非阻塞（non-block）方式调度：创建远程连接（connect），接收（recv）数据、发送（send）数据、定时中断（timer）、域名解析（resolve）和发送成功（done）中断回调事件；（3）、全面管理用户连接对象的导入和导出。

Master 主线程：（1）、按连接数对每一个 Slave 线程负载均衡；（2）、全非阻塞（non-block）方式接受（accept）远程用户的连接请求；（3）、将建立的连接（connection）按照负载均衡规则导出给 Slave 线程。

2.3. 图例说明：



3. 缓存设计

3.1. 缓存说明

在 Fikker 系统中, 缓存设计分为: 智能缓存设计, 强制缓存设计, 拒绝缓存设计。按照缓存存储模式分为: 公共缓存, 会员缓存, 游客缓存。缓存页面(html, asp, aspx, php, jsp, js, css 等)被 gzip 压缩后以平衡二叉树的索引结构存放在内存中, 不对硬盘进行任何读写(日志除外)。当加速缓存中的页面被访问命中以后, 通过 gzip 压缩传输方式返回给浏览器。以上处理方式有如下好处: (1)、不读写硬盘, 通过内存进行数据交换会极大的提高页面相应速度; (2)、文本页面数据经过 gzip 压缩后存储, 即减少了对内存空间的需求, 也会极大的减少数据传输量, 从整体上提高响应速度和传输效率。

3.2. 智能缓存设计

(1)、Fikker 已默认内置此功能, 用户无需对此进行额外设置, 智能缓存的页面储存在公共缓存中, 允许所有用户访问。

(2)、Fikker 根据 HTTP 协议头 Cache-Control, Pragma, Date 和 Expires 字段中的缓存控制属性进行缓存的(包含缓存时长), 部分网站可以通过对 HTTP 编程的方法来自行调整智能缓存策略。

(3)、Fikker 应用的智能规则如下:

用户 Request 请求时用到 Cache-Control:

"max-age" "=" seconds	-----忽略
"max-stale" ["=" seconds]	-----忽略
"only-if-cached"	-----忽略
"no-transform"	-----忽略
"min-fresh" "=" seconds	-----忽略
"cache-extension"	-----忽略
"no-cache"	-----是否启用缓存? 配置选项
"no-store"	-----是否启用缓存? 配置选项

源站 Response 返回时用到 Cache-Control:

"max-age" "=" seconds	-----缓存
"s-maxage" "=" seconds	-----缓存
"public"	-----缓存
"private" ["=" field-name]	-----不缓存
"no-cache" ["=" field-name]	-----不缓存
"no-store"	-----不缓存
"must-revalidate"	-----不缓存
"proxy-revalidate"	-----不缓存
"cache-extension"	-----忽略
"no-transform"	-----忽略

Fikker 对此处理如下:

max-age

最大缓存有效期, 从获得 Response 报头开始计算缓存有效期。

s-maxage

最大共享缓存有效期，与 max-age 处理规则相同。

public

缓存，而且是永久缓存，从获得 Response 开始，开始永久缓存。

private

私有缓存控制，不能当作公共缓存对待，只针对某一个用户的请求缓存有效，但在 Fikker 中我们不将其缓存。

no-cache

非缓存标识，浏览器的请求都需要源服务器响应，Fikker 不缓存任何数据。

no-store

非缓存非存储标识，Fikker 不缓存不存储。

no-transform

允许缓存，但不得改变源服务器返回的内容的格式，例如图片格式，文档类型等，Fikker 不缓存。

must-revalidate

允许缓存，但浏览器访问时候，Fikker 需要向源服务器提请验证，验证源内容有无修改，Fikker 不缓存。

注：参看 rfc2616，章节：14.9 Cache-Control，页码：Page 108

用户 Request 请求时用到 Pragma：

| "no-cache" -----不启用缓存

源站 Response 返回时用到 Pragma：

| "no-cache" -----不缓存

源站 Response 返回时用到 Expire 和 Date：

Expires - Date = 缓存有效期 -----缓存

避免冲突策略：

当 HTTP 报头中同时设置有 Expire，Pragma，Cache-Control 字段时，优先权顺序为 Cache-Control > Pragma > Expire。这个优先策略可以参看 rfc2616，章节 13.1.3 Cache-control Mechanisms，页码 Page 77，章节 14.9 Cache-Control，页码 Page 108，章节 14.9.3 Modifications of the Basic Expiration Mechanism，页码 Page 111。

3.3. 强制缓存设计

- (1)、通过页面缓存配置将指定的页面添加到加速缓存中，浏览器远程访问这个页面时，Fikker 将直接返回已缓存的页面，最大限度的减轻网站和网站数据库负荷。
- (2)、缓存的页面能够被周期性更新，周期性间隔时长由用户自行设定，可以是几秒钟，几分钟甚至是几百个小时。
- (3)、文本缓存页面将被 gzip 压缩存储和传输，文本页面(asp, php, jsp, aspx, js, css, txt 等)被压缩传输时，相对于非压缩传输，占用的带宽将减少 70% 以上。
举例：一个 500KB 的文本页面，被压缩以后为 110KB 左右，减少带宽消耗 75% 左右，在大量并发访问时，会节省大量的带宽损耗。

(4)、动态页面(asp, php, jsp, aspx 等)被加速缓存后, 当网站被大量并发访问时, 由于没有了数据库数据读写(硬盘读写)瓶颈, 整体上能够提升页面的响应速度。

Fikker 页面缓存由 "公共缓存" 和 "会员缓存" 和 "游客缓存" 组成。

(Fikker 页面缓存 = 公共缓存 + 会员缓存 + 游客缓存)

3.3.1. 公共缓存

页面被缓存后, 所有用户都可访问, 是最简单的页面缓存模式。应用于网站上的大部分图片, JS 脚本, CSS 文本, 静态页面 html 等所有开放的页面内容。

3.3.2. 会员缓存

页面被缓存后, 只有登录用户才可以访问, 针对会员访问加速。很多社交网站 SNS, 社区论坛 BBS, 电子商务 B2C, 办公 OA 等均可通过此模式实现会员访问加速。

3.3.3. 游客缓存

页面被缓存后, 只有游客用户才可以访问, 对非登陆用户生效, 已登录的用户不能访问。

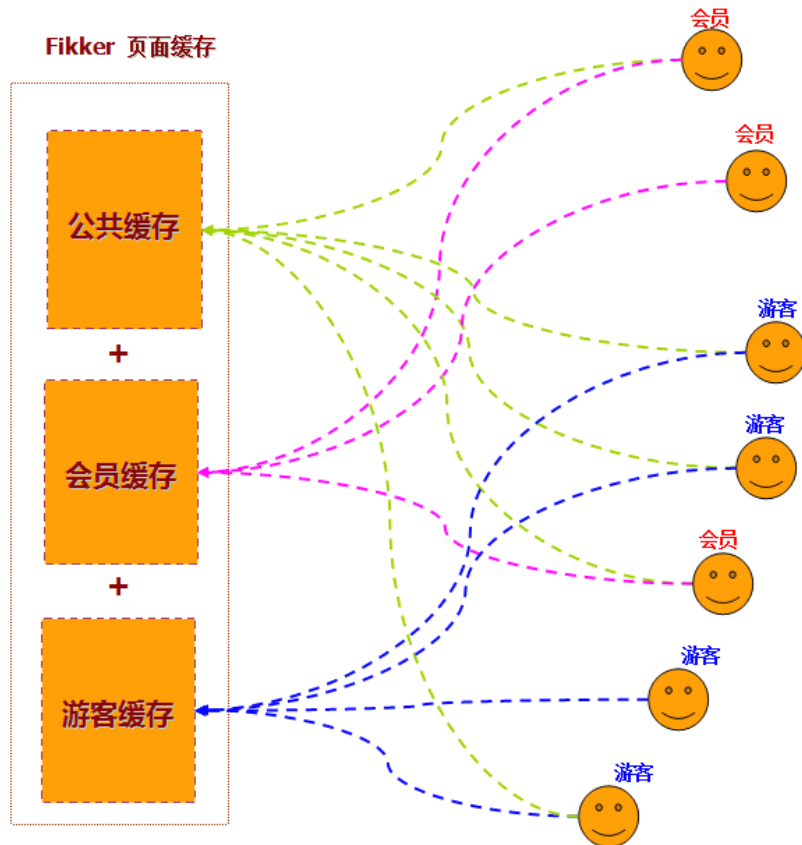
3.3.4. 会员缓存 - 设计背景:

一些社交(SNS), 论坛(BBS), 新闻(News), 博客(Blog), 电子商务(B2C, C2C)类网站, 登录用户(会员)和非登录用户(游客)看到的网站页面内容可能是不同的, 部分页面中包含的内容只有用户登陆后才能查看, 即便是相同的页面链接地址 URL, 登录用户(会员)和非登录用户(游客)分别请求时, 请求到的页面内容是不完全相同的。为了达到加速的目的, Fikker 会分别缓存会员访问的页面(会员缓存)和游客访问的页面(公共缓存)。当用户向 Fikker 请求页面时, Fikker 首先会判断这个用户是否已登陆, 如果为已登录(会员)用户, 则返回的为“会员缓存”对应的页面内容, 如果为非登录(游客)用户, 则返回的是“公共缓存”对应的页面内容。

那么 Fikker 是怎么判断用户为已登录(会员)用户呢?

我们知道, 用户登录成功后, 网站会返回浏览器一个会话标识(Session ID), 这个会话标识一般是网站通过 HTTP 的 Set-Cookie 字段中的一个变量来传递的, 浏览器会一直保存这个会话标识(Session ID)直至浏览器被完全关闭或会话超时。当登录后的用户使用浏览器访问网站的页面时, 浏览器都会通过 HTTP 的 Cookie 来提交这个会话标识(Session ID)给网站来进行验证用户是否已登录。Fikker 就是通过缓存这个会话标识(SessionID)来判断访问用户是否为已登录用户的。

3.3.5. 图示说明:



3.4. 拒绝缓存设计

- (1)、拒绝将指定的页面加入到加速缓存中，这样子页面在被用户请求访问时都将是直接从源站获取最新的页面。
- (2)、一些网站管理后台或用户管理页面不适合被缓存。因为这部分页面包含了只针对某一个用户的私有数据，即相同的一个页面，不同用户获取的页面内容是不同的。

3.5. 清理缓存设计

- (1)、有时候管理员修改“页面缓存”配置后，需要刷新页面缓存，以便于将不符合配置的缓存页面清理出去，最新配置能即时生效。
- (2)、有时候网站调整了页面内容，并且希望这个修改能即时生效，这个时候需要管理员需要将指定链接 Url 缓存清理出去，使得网站改动能够即时生效。

3.6. 缓存优先级设计

- (1)、优先级匹配顺序为：拒绝缓存 > 强制缓存 > 智能缓存。
- (2)、举例 1：当一个页面被同时设置为拒绝缓存，强制缓存和允许被智能缓存，因为拒绝缓存的优先级最高，即此页面不会被缓存。
- (3)、举例 2：当一个页面即被设置成了强制缓存，但同时也能被智能缓存（根据 HTTP 头的 Expire, Pragma, Cache-Control 字段属性缓存），因为强制缓存优先级较高，即按照强制缓存配置生效，利用强制缓存配置进行周期性页面更新。

3.7. 页面匹配规则设计

(1)、通配符匹配 (Wildcard Character Pattern), 比较简单易用的匹配方式, 通配符为问号 ? 和星号 *, 问号 ? 匹配一个任意字符, 星号 * 匹配 0 个或多个任意字符。

例如: `fikker.com/images/*.jpg` 可匹配 `fikker.com/images/sun.jpg`, 也可匹配 `fikker.com/images/ad/house.jpg`, 即可匹配 `fikker.com` 网站目录 `images` 下面所有的 `jpg` 格式图片。

(2)、正则表达式匹配(Regular Expression Pattern), 可以设定复杂但丰富的匹配条件, 详细请参考相关的 Posix Regex 资料说明介绍。

(3)、精确匹配, 逐一字符匹配完整的链接地址 URL。

例如: `fikker.com/docs/manual.html` 只能成功匹配 `fikker.com/docs/manual.html` 链接本身这一个具体链接地址。

3.8. 缓存页面淘汰算法设计

3.8.1. 缓存页面淘汰说明

(1)、当留给页面缓存使用的内存耗用完毕后, 如果还有新的页面要被缓存时, 就需要将老的已缓存页面置换下来, 腾出空间为新的缓存页面。这个算法指的就是页面淘汰的规则设计。

(2)、淘汰的原则:

- (a)、优先淘汰非动态页面, 在大多数网站中, 这种资源尤其以图片, 视音频居多。
- (b)、再淘汰静态文本资源, 例如 `css` 和 `js` 和 `txt` 等静态页面。
- (c)、最后才是动态 `asp`, `aspx`, `php`, `jsp`, `do` 等页面或静态 `html` 页面。

3.8.2. 缓存页面淘汰优先级

- (1)、淘汰过期缓存页面, 就是将超时的缓存页面清理掉, 腾出内存空间, 如果腾出来的内存空间不够新页面的所需尺寸, 进入第二步。
- (2)、将图片视频类缓存页面置换下来, 图片对源站的负载影响较小, 可以默认设置第一优先淘汰图片页面(`jpg`, `gif`, `png`, `swf`, `bmp`, `wmv`, `mp3`, `avi` 等)。这个类别是通过 `Content-Type` 和 `mime` 信息去识别的, 如果淘汰的图片页面腾出来的空间还是不够新页面的所需尺寸, 进入第三步。
- (3)、将静态文本类缓存页面置换下来, 静态页面对源站的负载影响较小(相对静态页面而言), 第二优先淘汰静态文本(`css`, `js`, `txt`)。如果淘汰的静态页面腾出来的空间还是不够新页面的所需尺寸, 进入第四步。
- (4)、将剩下的缓存页面按照“同一优先级淘汰规则”置换下来, 直到满足新页面所需内存尺寸。

3.8.3. 同一优先级淘汰规则

(1)、同一个淘汰优先级中, 那一个页面又将最先被淘汰呢? 设计上应该是访问量最少 (`access_count`) 并且贮存在内存时间最久 (`cache_interval`) 的页面被淘汰, 即: 平均单位访问量 (`access_count / cache_interval`) 最小的缓存页面将会被优先淘汰, 这样子将是相对比较合理和公平的。

3.9. 页面压缩 gzip 设计

3.9.1. 说明

- (1)、Fikker 服务器支持对文本页面 gzip, deflate, compress 方式压缩, 目的是减少缓存页面的内存占用, 提高传输效率, 提高用户端的页面加载速度。
- (2)、Fikker 服务器对浏览器的访问支持 Accept-Encoding: gzip, deflate, compress 字段属性。

3.9.2. HTTP 头中有关 gzip 压缩的字段

- (1)、Accept-Encoding: gzip, deflate, compress: 表明浏览器同时支持 gzip, deflate 和 compress 压缩, 网站返回的页面内容可以被压缩后再传输过来。但如果浏览器不包含这个字段, 而且命中了 Fikker 缓存页面, 这个时候 Fikker 会检查是否这个内容是被压缩过的, 如果是压缩内容, 即将其解压后去除 Content-Encoding 字段并修改 Content-Length 字段后再返回给浏览器。
- (2)、Content-Encoding: 标识网站返回的页面内容是压缩过的, 并且压缩格式可以为 gzip, deflate 或 compress 格式, 这样子浏览器在显示这一些页面内容前, 需要先将其解压后再显示。
- (3)、Transfer-Encoding: chunked : 如果网站返回的 HTTP 头中包含了这个字段, 表明整合页面内容是分块传输的。对于非压缩的文本页面内容, Fikker 在压缩页面前, 会检查块传输的完整性, 待块传输完全和完整后, Fikker 会将块标记(Transfer-Encoding: chunked)去除。Fikker 将页面压缩(gzip)成功后, 并添加 Content-Length 字段属性在 HTTP 头中。

3.9.3. 页面压缩的实现

- (1)、浏览器请求中包含 Accept-Encoding 字段属性时, 对于网站返回的内容, Fikker 会检查是否属于非压缩的内容, 通过检查网站返回的 Content-Encoding 属性, 判断返回的内容是否已压缩, 并且知道了压缩格式。
- (2)、如果网站返回的内容属于被压缩内容, 并且属于文本内容, Fikker 便会主动的将其压缩成默认的 gzip 格式内容, 将其缓存在内存中。
- (3)、如果网站返回的内容已经是压缩格式的内容, 并且是块传输的 Transfer-Encoding: chunked, Fikker 便会将其去除页面内容中包含的 chunked 标记, 在 HTTP 头中添加 Content-Length 字段属性。
- (4)、如果浏览器访问命中缓存, 但是浏览器请求的内容要非压缩的, 即在 HTTP 头中没有包含 Accept-Encoding: gzip, deflate, compress 字段属性, Fikker 便将已缓存的压缩页面取出来, 解压后, 再返回给浏览器。

4. URL 转向 (rewrite) 设计

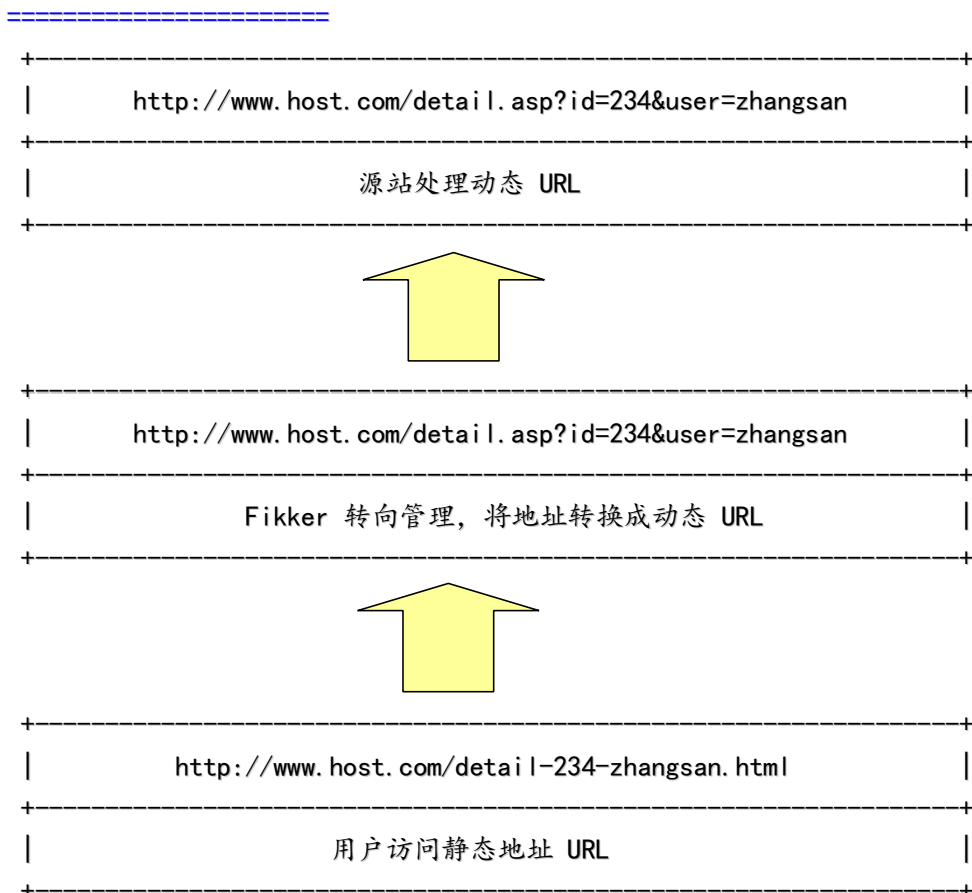
4.1. URL 转向 (rewrite) 简介

也可称 'rewrite 转向', 可将用户访问请求的链接地址'转向'到另外的链接地址, 用来实现的功能如下:

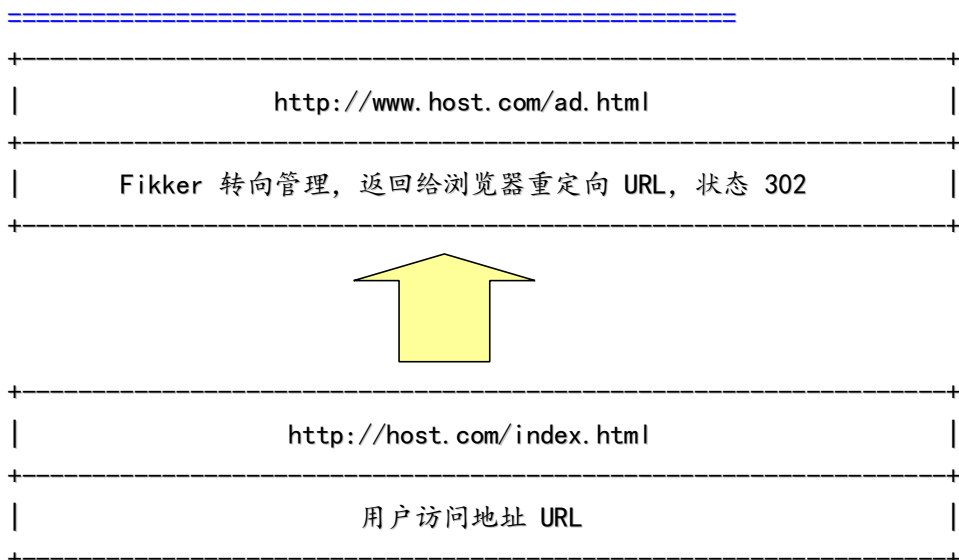
- (1). 实现网站的 "伪静态" 或 "静态转动态" 功能, 即将用户从外部访问的静态链接地址 (html, htm, shtml 等) 转换成网站可处理的动态链接地址 (asp, aspx, php, jsp, do 等)。
- (2). 实现**重定向**功能, 对用户访问的某一个链接地址重新定向另外一个不同的链接地址, 举例: 将用户所有访问请求地址 <http://fikker.com> 全部重定向到 <http://www.fikker.com>

(3). 实现比较复杂的访问请求控制功能，例如：网站某一些页面的临时或永久性屏蔽，网站临时维护通知，分布式请求等等；

图解 1（伪静态实现原理）：



图解 2（重定向原理，返回 HTTP 状态 302 Object Moved）：



4.2. 转向逻辑

(1)、Last，访问地址匹配成功后，立即终止，按照规则转换(重写) URL 转发给源

站。

(2)、Return, 访问地址匹配成功后, 立即终止, 重定向浏览器 302 Object Moved 新的 URL 地址。

(3)、Round, 访问地址匹配成功后, 不终止, 回到起始位置, 开始新一轮循环, 重头匹配。

(4)、Continue, 访问地址匹配成功后, 不终止, 继续向下一项规则匹配。

4.3. 简单举例

+-----+		+-----+
访问地址 URL	-->	转向地址 URL
+-----+		+-----+
^fikker\.com/(.*)\$	-->	www.fikker.com/\$1
+-----+		+-----+

解释:

用户访问请求 URL 如果与正则表达式 ^fikker\.com/(.*)\$ 匹配成功, Fikker 就会将用户访问的 URL 转换成 www.fikker.com/\$1

例如: 用户访问 http://fikker.com/faq.html 时候, Fikker 转向管理会将访问请求 URL 转换成 http://www.fikker.com/faq.html 然后转发给源站或将重定向 URL 转发 (HTTP 协议 302 状态) 给浏览器。

5. 防盗链设计

5.1. 防盗链说明

用来保护网站资源不被第三方盗用, 例如: 图片, 视频, 特殊页面等, 这一些资源一旦被第三方网站大量盗用, 会极大的占用网站服务器的 CPU 资源和带宽资源, 防盗链管理可以将其保护起来, 能够不被第三方站点随意引用。如下说明:

- (1)、将需要保护的资源(图片, 视频, 特殊页面等), 添加到保护链管理中。
- (2)、设定保护链的访问权限, 允许对管理员指定的任何第三方站点开放引用权限。
- (3)、Fikker 中可建立多个保护链。
- (4)、每一个保护链, 又可以建立多个引用链。只有引用链才有权限访问保护链的内容。

(5)、逻辑图例:

+-----+		+-----+
		+-----+
		引用链 1
		^news\.host\.com/.*\$
	+-----+	+-----+
1	保护链 1	
	^host\.com/.*\.jpg\$	+-----+
	+-----+	引用链 2
		^tech\.host\.com/.*\$
		+-----+
	(1). 正则表达式匹配;	

	(2). 保护 host.com 站点	(1). 保护链 1 允许上面两个	
	下的所有 jpg 图片;	站点下面的所有链接引用;	
+-----+			
		+-----+	
		引用链 1	
		^news\.baidu\.com/.*\$	
		+-----+	
2	保护链 2		
	^host\.com/1\.html\$	+-----+	
	+-----+	引用链 2	
		^item\.taobao\.com/.*\$	
		+-----+	
	(1). 正则表达式匹配;		
	(2). 保护 host.com 站点	(1). 保护链 2 只允许 百度新闻	
	下 1.html 页面;	和 淘宝 item 站点下面指	
		定的链接引用;	
+-----+			

6. 黑名单设计

6.1. 黑名单说明

有时候网站出于安全性考虑, 网站不希望一些 IP 或一些 IP 段被访问, 此时可以将这些 IP 添加到黑名单, 被列入黑名单的 IP 在向 Fikker 发送访问请求时, 会被 Fikker 禁止, 并返回提示信息(403 Forbidden)。

(1)、允许禁止单个 IP 地址或一个 IP 地址段范围,

例如: 21.23.44.10 - 21.23.44.10 --> 单个 IP

例如: 21.23.44.10 - 21.23.44.100 --> 一个 IP 地址段, 共 91 个 IP 地址

(2)、对黑名单 IP 地址有一个禁止期限, 到达这个期限, 黑名单会被自动解禁, 例如: 21.23.44.10 被禁有效截至日期为 2011-06-05 08:30:00

7. 代理设计 (主机管理)

7.1. 代理介绍

在 Fikker 系统中, 主机管理完成如下功能:

(1)、代理功能, Fikker 是前置机, 是放在源站 (网站服务器) 前面的服务器, 首先接收到用户的连接访问请求, 然后再转发用户请求到一个或多个源站。

(2)、负载均衡管理, Fikker 后面的源站 (网站服务器) 可以是一个或多个源站服务器, Fikker 通过设置不同**负载均衡策略**将用户请求——分发给多个源站服务器,

让不同的源站（网站服务器）处理的不同的用户请求。

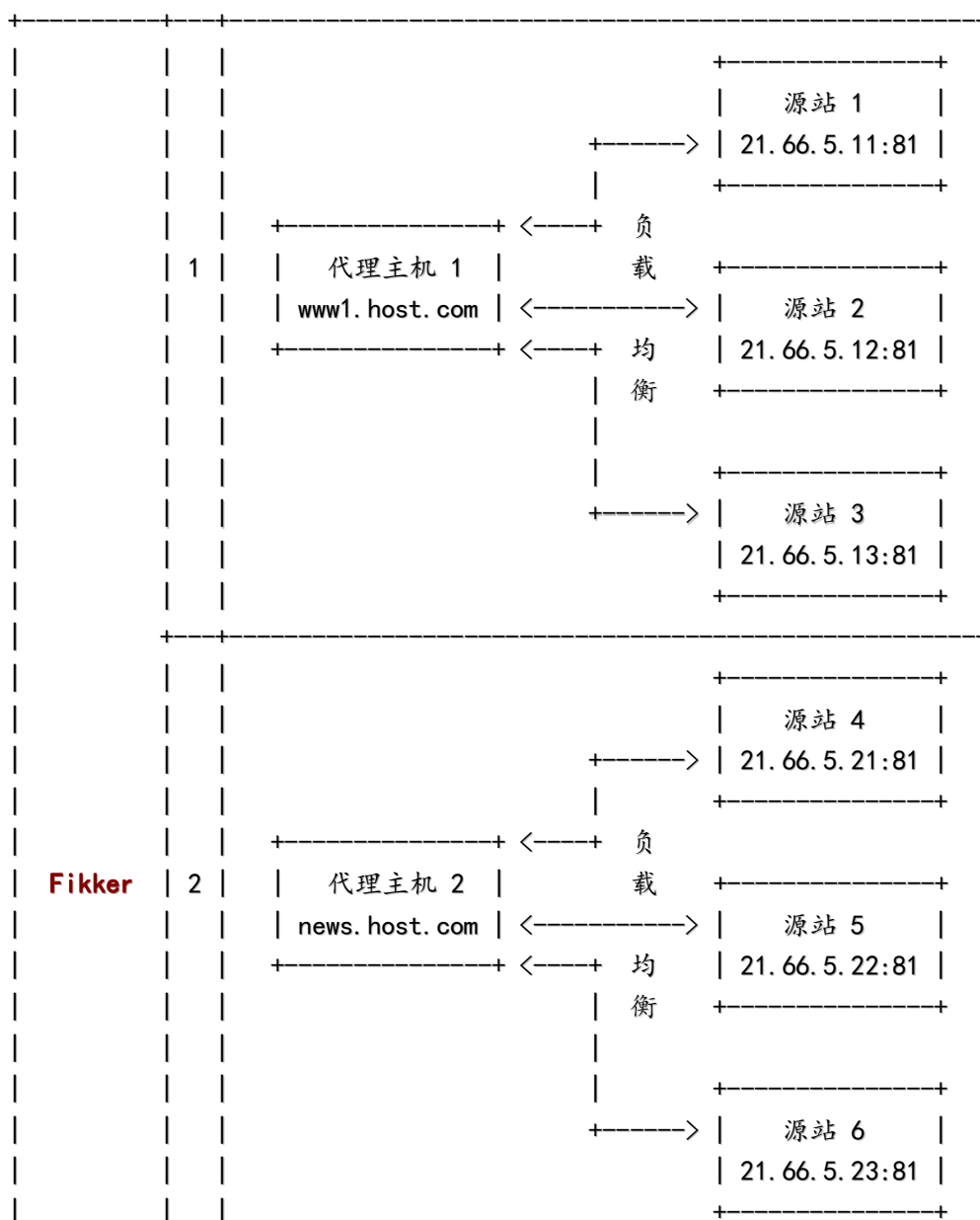
7.2. 负载均衡策略：

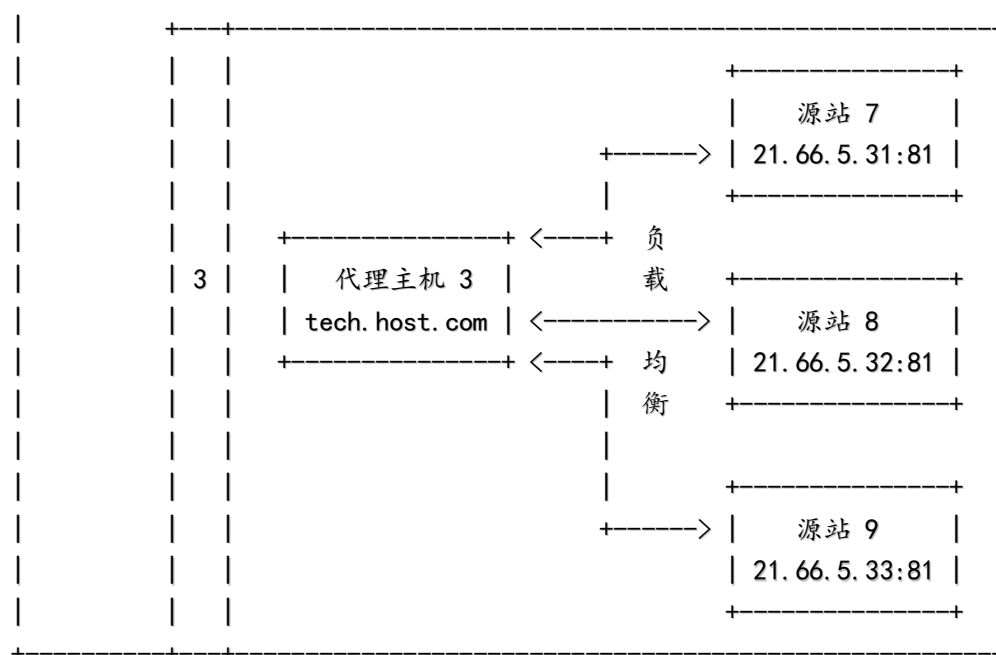
(1)、轮询均衡策略，常用负载均衡策略，在多个源站情况下，Fikker 对各个源站按照访问依次轮询访问。

(2)、IP 哈希均衡策略，在多个源站情况下，Fikker 根据用户的请求 IP 地址 Hash 值后，再决定访问那一个源站。

(3)、URL 哈希均衡策略，在多个源站情况下，Fikker 根据用户的请求 URL 计算 Hash 值后，再决定访问那一个源站。

7.3. 负载均衡图例：





图例说明:

- (1)、一个 Fikker 服务器允许添加多个主机名, 同时代理多个主机。
- (2)、一个主机名 (或域名) 允许对应多个源站, 多个源站之间应用负载均衡规则。
- (3)、Fikker 按照代理规则将用户的访问请求最终一一分发到多个源站。

8. 流量统计设计

8.1. 流量统计说明

Fikker 是网站的前置服务器, 流量统计就是对流经 Fikker 的访问连接请求进行有效统计, 包括区域性 (省份) 流量统计, 流量百分比等。在 Fikker 中, 统计分两类, 一种是总量统计, 由 Fikker 系统自动生成; 一种是分量统计, 由用户自定义统计 URL。

统计术语:

(1)、PV - PageView 页面浏览量, 是指在一定时间范围内访客进入网站后, 浏览过的该网站网页的实际数量。我们通常理解的页面浏览量的统计范围一般不包括 css, js, class, gif, jpg, jpeg, png, bmp, ico, swf 等后缀结尾的页面文件, 统计的范围一般只限于 html, asp, aspx, php, jsp, do 等静态或动态页面。

(2)、IP - 独立 IP 访问量, 指一定时间范围内访客进入网站后, 访客的实际 IP 量。在 Fikker 中, 一个 IP 无论访问了多少页面, 都只能按照一个独立 IP 去计数。

(3)、TR - TotalRequest 总请求量, 这个请求量包括了访客对 Fikker 所有的有效连接访问请求, 不只限于动静页面(html, asp, aspx, php, jsp, do 等)统计, 还包含了网站下面其它所有的可请求资源 css, js, class, gif, jpg, jpeg, png, bmp, ico, swf 等后缀结尾的文件。

统计区域划分:

"北京", "天津", "上海", "重庆", "内蒙古", "新疆", "西藏", "宁夏", "广西", "黑龙江", "吉林", "辽宁", "河北", "山西", "青海", "山东", "河南", "江苏", "安徽", "浙江", "福建", "江西", "湖南", "湖北", "广东", "海南", "甘肃", "陕西", "四川", "贵州", "云南", "香港", "澳门", "台湾", "高校", "局域网", "其它"

8.2. 总量统计

由系统生成, Fikker 会对用户的访问请求生成统计文本报告, 生成后放在 reports 目录下, 也可进行实时监控。图例:

统计起始时间 (2010-11-17 19:46:22) - 统计截至时间 (2010-11-17 19:47:11)

省份\项目	总请求量 TR	独立 IP 访问量	人均请求数量	流量百分比 %
北京	0	0	0.00	0.00%
上海	0	0	0.00	0.00%
重庆	0	0	0.00	0.00%
内蒙古	0	0	0.00	0.00%
其它	33	1	33.00	100.00%
合计 ->	33	1	33.00	100.00%

总量缓存命中率: 51.52%

8.3. 分量统计

用户自定义的统计 URL, 用户通过设定 URL 匹配条件 (通配符, 正则表达式, 精确匹配), Fikker 会对匹配的页面进行统计, 生成文本报告, 生成后放在 reports 目录下。也可以进行实时监控。图例:

统计 URL: bbs.fikker.com/*

备注说明: 对 bbs.fikker.com 站点下面的所有页面请求进行流量统计, 包括了 [html](#), [jpg](#), [js](#), [css](#) 和 [swf](#) 等所有文件的有效请求量。

统计起始时间 (2010-11-17 19:46:22) - 统计截至时间 (2010-11-17 19:47:11)

省份\项目	总请求量 TR	流量百分比 %
北京	0	0.00%
四川	0	0.00%
贵州	0	0.00%

其它	0	0.00%	
+-----+-----+-----+			
合计 ->	0	0.00%	
+-----+-----+-----+			

分量缓存命中率: 0.00%

9. 实时监控设计

9.1. 实时监控说明

反映 Fikker 系统当前的运行时(runtime)信息:

(1)、当前用户并发连接请求数(CurrentUserConnections):

反映的是 User -> Fikker 连接数。

(2)、源站并发连接请求数(CurrentUpstreamConnections):

反映的是 Fikker -> Upstreams 连接数。

(3)、当前内存总占用(AllUsedMemSize):

Fikker 统计所有模块占用的内存, 包括页面缓冲(Webcache), 过滤器, 域名解析, 网络连接数据交换空间, 系统网络缓存等内存占用汇总统计, 单位为 KB。

(4)、当前页面缓存占用内存(CacheUsedMemSize):

用于页面缓存的内存占用统计, 单位为 KB。

(5)、当前页面缓存数量(NumOfCaches):

当前页面缓存的数。

(6)、Fikker 已发送数据量(TotalSendKB):

统计当前 Fikker 已发送的数据量, 单位为 KB。

实时带宽计算方式:

实时带宽(bps) = (本次发送数据量 - 上次发送数据量) / 本地与上次时间间隔。

注意:

bps 带宽是按照 bit 速率计算的, byte 速率乘 8 才是 bps 带宽值。

(7)、Fikker 已接收数据量(TotalRecvKB):

统计当前 Fikker 已接收的数据量, 单位为 KB。

实时带宽计算方式:

实时带宽(bps) = (本次接收数据量 - 上次接收数据量) / 本地与上次时间间隔。

注意:

bps 带宽是按照 bit 速率计算的, byte 速率乘 8 才是 bps 带宽值。

(8)、当前总量统计汇总报告(CurrentTotalStatReport):

(9)、各个分量实时统计汇总报告(CurrentItemStatReport):
参看上面流量统计章节。

(10)、实时监控远程连接请求, 这些信息包括:

- (a)、请求 IP 和端口
- (b)、User-Agent
- (c)、请求方法 Method
- (d)、请求地址 URL
- (e)、是否命中缓存, 是否命中会员缓存
- (f)、是否非法请求或无效请求
- (g)、IP 详细地理位置信息

10. Windows 和 Linux 跨平台设计

10.1. 绿色安装包

- (a)、在 windows 下, 直接解压 zip 后即可运行, 不写注册表, 除了在安装目录下的 reports, logs 和 config 用来存放日志和配置信息外, 不再有写硬盘操作。
- (b)、在 Linux 下, 直接解压 tar.gz 在任何一个目录下, 即可直接运行, 安装目录设计结构与 windows 兼容。

10.2. Fikker 的运行权限

- (a)、在 window 下, 需要使用管理员 administrator 权限去运行 Fikker
- (b)、在 Linux 下, 需要根用户 root 权限去运行 Fikker

10.3. 配置文件兼容性

Windows 和 Linux 下的配置都是相互兼容的(除加载模块名称不同外), 详细需参看不同安装包下面的 *.ini 配置文件里面的详细说明。

11. 关于 Fikker 限制的说明

11.1. 内存限制

在 x86 硬件架构下(以非 PAE 模式下运行时), 大部分 Windows 和 Linux 操作系统分配给应用程序的最大可用内存一般为 2048MB (2GB), Fikker 会使用一部分内存做系统远程配置管理, 固定数据存储, 内部动态数据交换, 线程栈空间, 独立 IP 统计, 模块管理和预留给未来系统设计扩展等。所以 Fikker 目前留给管理员可配置的最大可用内存为 1500MB, 主要用于:

- (a)、用于加速的动静态页面缓存空间
- (b)、网络连接发送和接收缓冲空间
- (c)、临时申请的数据块分配空间
- (d)、一定量的内存碎片冗余空间

11.2. 连接数限制

Fikker 允许最大建立并发连接数为 60000 个, 这个连接数包括:

- (a)、用户向 Fikker 发起的访问请求
- (b)、Fikker 向源站发起的页面请求
- (c)、系统管理员或系统监控员向 Fikker 发起的访问连接请求

(d)、端口映射模块建立的连接数（端口映射模块 portmap 被启用时）

11.3. HTTP 头尺寸限制

目前允许 HTTP 请求头的最大尺寸为 16K，当客户端的请求的 HTTP 头大于这个尺寸时，会当作非法请求来对待。常见的问题为客户端请求时，在 HTTP 头中的提交的 Cookie 的尺寸过大（例如大于 16K），造成访问失败。

HTTP 头包含的 Host 主机字段，最大长度不能超过 200 个字节。

HTTP 头包含的请求链接地址（Request URL）最大不能超过 4096 个字节（4KB）。

11.4. 缓存页面尺寸限制

目前允许单个缓存页面最大尺寸为 1024K（1MB）。

11.5. 操作系统限制

Windows 2000/XP/2003/2008/Vista/7 和 Linux for kernel 2.6.20 and above

11.6. 会话缓存限制

会话缓存最大尺寸为 200MB，网站返回的 Set-Cookie 中有效会话字符串尺寸需大于等于（>=）8 个字节，否则视为无效会话。

12. Fikker 与 HTTP 头

12.1. 说明：Fikker 在页面缓存，转向管理，主机管理，防盗链管理，流量统计中，都对 HTTP 头做了过滤，判断，转向等处理，具体的 HTTP 在 rfc2616 中已经做完整的说明和解释，这里主要将 Fikker 用到的 HTTP 头字段汇总如下：

Connection: Keep-Alive

Connection: close

Keep-Alive: 115

If-Range -> Range -> Content-Range

If-Modified-Since -> Last-Modified

If-None-Match -> Etag

100 Continue

200 OK

206 Range

301, 302, 304 Move Object -> Location

400 Bad Request, 403 Forbidden

500, 502, 503 Server Error

Content-Type:

Content-Length:

Transfer-Encode: chunked

Accept-Encoding: gzip, deflate, compress

Content-Encoding: gzip

Vary: Accept-Encoding

URI -> URL -> Host -> Path -> Filename -> Query -> #fragment

encodeURI

form, field -> Add -> Delete -> encode -> decode
GET, POST, HEAD, CONNECT
GMT DateTime Format
Expires, Cache-Control, Pragma, Date -> no-cache, max-age, private, public
Set-Cookie: domain, path, expire, httponly, value A few "Set-Cookie" in same HTTP
Cookie: value1; value2; value3
Content-Location:
Content-type:
Referer:

13. 关于 Fikker 返回错误页面的说明

13.1. 400 Bad Request

[400 Bad Request - By Rewriting Rules](#)

命中转向规则，转向目标地址为空或不合法。

[400 Bad Request - By Protecting Rules](#)

命中防盗链规则，转向目标地址为空或不合法。

[400 Bad Request - HTTP Request Error](#)

用户请求的 HTTP 头出现错误，原因如下：

- (a)、头太大，大于 16K
- (b)、HTTP 头缺少 Host 字段
- (c)、客户端访问的 URL 有错误

[400 Bad Request - Invalid Host](#)

用户访问请求的 HTTP 头无 Host 字段或 Host 字段有错误。

13.2. 403 Forbidden

[403 Forbidden](#)

禁止请求访问，当前用户的 IP 地址已经被锁定在黑名单中。

13.3. 409 Conflict

[409 Conflict - Routing Loops By X-Fikker](#)

用户的访问请求出现路由回环现象（死循环），出现的原因如下：

- (a)、Fikker 转向管理配置错误
- (b)、Fikker 主机管理或源站管理错误

13.4. 502 Bad Gateway

[502 Bad Gateway - DNS Resolving Fail For Upstream Server](#)

请求主机 Host 域名解析失败或超时。

502 Bad Gateway - Can't Connect To Upstream Server

Fikker 向源站建立网络连接失败，出现原因如下：

- (a)、源站不可用，例如：源站宕机或没有在运行状态
- (b)、源站端口与所连接的端口不是用一个端口
- (c)、网络临时不可用

502 Bad Gateway - Pipe Is Gone

Fikker 建立的双向管道（双向网络通道）失败，出现的原因：

- (a)、受到操作系统本身连接数限制
- (b)、操作系统内存或网络可用资源不足够
- (c)、网路临时不可用

13.5. 503 Service Temporarily Unavailable

503 Service Temporarily Unavailable - Reach To Maximum Upstream Connections Setting

Fikker 临时不可用，说明如下：

- (a)、达到了管理员在系统配置中设定的最大源站连接数设定。
- (b)、当有连接资源被释放时，会自动恢复可用性。

503 Service Temporarily Unavailable - Fikker Is Paused

Fikker 临时不可用，Fikker 已被暂停运行，说明如下。

- (a)、在系统配置中管理员已设定 Fikker 为暂停运行。
- (b)、当管理员重新设定系统配置为开始运行时，会自动恢复 Fikker 可用性。

503 Service Temporarily Unavailable - Reach To Maximum Usable Memory Size Setting

Fikker 临时不可用，达到最大可用内存尺寸设定，说明如下：

- (a)、达到了管理员在系统配置中设定的最大可用内存尺寸设定，内存受限。
- (b)、当有连接资源被释放或缓存页面被释放时，有空闲内存可用时，会自动恢复 Fikker 可用性。

503 Service Temporarily Unavailable - Reach To Maximum Fikker Connections Setting

Fikker 临时不可用，说明如下：

- (a)、达到管理员在系统配置中设定的最大 Fikker 连接数。
- (b)、当有客户端连接资源被释放时，会自动恢复 Fikker 可用性。

503 Service Temporarily Unavailable - Blocked By Anticc

Fikker 临时不可用，说明如下：

- (a)、被 Fikker 的防 CC 过滤器截获，用户的访问频率超出了 anticc.ini 相关的配置。
- (b)、当有客户端连接并发和阻止超时后（参看相关的配置项），会自动恢复 Fikker 可用性。

