基于最大公共模式抽取的 DNS 代理缓存方法

刘俊朋[1]，张鹏[1]，刘晓梅[1]，邹学强[2]，孙永[1]，刘庆云[1]

(1. 中国科学院 信息工程研究所，北京 100093；2. 国家计算机网络应急技术处理协调中心，北京 100029)

摘　要：目前的域名缓存机制在查询负载过重情况下性能表现不稳定，为此本文设计并实现了一种基于最大公共模式抽取的 DNS 代理缓存方法 MFBDP。该方法基于域名的查询频率，进行高热度域名的缓存，并且基于概率的缓存替换以保证高热度域名以较低概率替换。针对热度发生变化的域名，替换概率也会做出自适应的调整，减少了域名平均查询延迟，增强了域名查询性能的稳定性，并且通过分析域名前缀的公共模式，对相似域名的前缀进行聚合，降低由于大量变前缀域名请求带来的资源消耗。实验验证了该方法的有效性。

关键词：高热度域名，公共模式，缓存

A Greatest Common Pattern Extraction based DNS Proxy Caching Method

LIU Jun-peng[1], ZHANG Peng[1], LIU Xiao-mei[1], ZOU Xue-qiang[2], SUN Yong[1], LIU Qing-yun[1]

(1. Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China;

2. National Computer Network Emergency Response and Coordination Center, Beijing 100029, China)

Abstract: As the current domain caching mechanism under the heavy query load may result in poor performance, in order to solve this problem, this paper proposes a greatest common pattern extraction based DNS proxy caching method-MFBDP. The method explores query frequency to cache high-frequency domain name, and based on the probability of cache replacement to ensure high-frequency domain name is removed at a relatively low probability. For the domain name with varying query frequency, the caching replacement probability will make adaptive adjustment process to reduce the average query delay, and consequently enhance the stability of the domain query performance. Moreover, by extracting the common patterns of the domain prefix, the similar domain names are merged to reduce the resource consumption. Experimental results show that this method is effective.

Key words: high-frequency domain, multi-dimension features, caching

1　引言

目前针对 DNS 代理缓存的优化技术主要采用基于哈希表的方法对域名应答记录中的所有属性进行存储[1-3]。王虎等人[1]采用哈希表结构，将所有应答资源记录以“域名”为键，以“应答 IP 列表、当前时间、生存周期”为值进行存储；杨驌宇等人[2]将全部的域名应答资源记录以哈希表的形式存在缓存中。虽然基于哈希表的缓存机制可以加快域名检索速度，但这些方案仍存在以下缺点：

(1) 哈希表大小主观选定，并采用链式处理哈希冲突。主观选定的哈希表大小可能对缓存性能造成影响。若哈希表设置过大，可能造成空间浪费，若设置过小，可能造成冲突率增加；

(2) 未对资源记录本身进行存储优化，当存储域名量大时，导致有限的存储空间会存储较少的表项，存储利用率较低。

除此之外，在进行缓存记录更新替换时，现有的方法大多根据应答记录的生存时间进行更新[4,5,6]。生存时间的设置是查询准确率和查询延迟

本课题得到国家自然科学基金项目(No. 61402464, 61502478)资助

的折中。Ballani[4]和 Ramasubramanian 等人[6]指出虽然某些域名资源的应答记录可以稳定数月，但生存时间仅设置为几小时，这会降低 DNS 记录在缓存中的命中率，不利于缩短 DNS 的查询时延，同时会增加远程域名服务器的负载。文献[5]为防止缓存受到 DDoS 攻击，将域名记录的生存时间增大，但这可能导致应答记录过时，返回结果不准确。同时，域名生存时间由远程服务器指定，不会因域名热度而改变。但在实际应用环境中，域名访问通常具有较强的局部性，一段时间内仅发生有限个域名的请求行为。而基于生存时间进行缓存更新时不能保证高热度域名经常被命中。因此基于生存时间的更新策略不利于缓存域名服务器在查询大量域名的情况下具有较高的命中率和较小的递归查询负载。

综上所述，针对域名查询量大、域名/IP 对应关系复杂给代理缓存域名服务器带来的存储压力大的问题，需要设计一种合理的代理缓存存储和检索方法，保证面对大量域名查询请求时，缓存的存储压力较小。同时为了保证用户较小的查询时延和准确的应答结果，需要为代理缓存设计实现合理的代理缓存记录更新替换算法。

2 缓存存储与检索方法

根据从 DNS 信息中分析得出的高热度域名特征，本文设计一种基于最大公共模式抽取的 DNS 代理缓存技术 MFBDP。MFBDP 利用多级哈希的表项分级、检索速度快等特性设计 DNS 代理缓存存储和检索技术，主要包括两部分：DNS 代理缓存存储与检索方法和基于表项概率的缓存更新替换方法。MFBDP 的工作流程如图 1 所示：

(1) 从 DNS 报文中解析出域名，时间，报文类型等；

(2) 将非法查询域名进行过滤，若查询域名在缓存中命中则直接响应结果；若不命中则进入到(3)。

(3) 将未命中的合法域名请求转发到运营商配置的缓存域名服务器中，然后将应答结果的域名、应答 IP 地址等信息进行聚合存储至代理缓存中，更新各个表项的频数、时间等信息。

(4) 在缓存域名的过程中，某二级或三级域的应答结果为“$NXDOMAIN”的频数在一定时间内超过阈值，则通过引入通配符’*’进行前缀表项的最大公共模式抽取。对缓存中的对应记录进行更新，仅仅存储聚合后的最大公共模式前缀串。

(5) 当某级哈希表满时，利用基于表项概率的缓存更新替换算法从哈希表中选择合适表项进行缓存更新。

下面对 MFBDP 的存储与检索方法的设计与实现进行详细介绍。

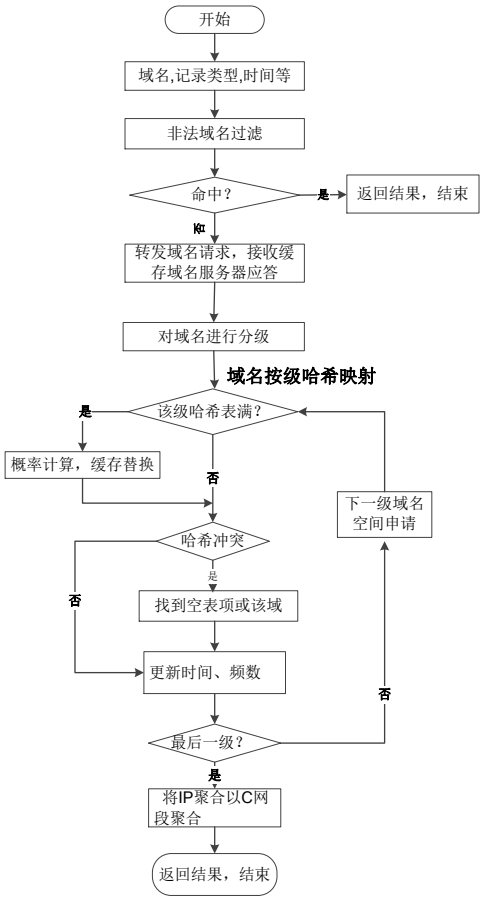

图 1 MFBDP 代理缓存流程

2.1 存储与检索方法设计

MFBDP 存储和检索方法总体模块设计如图 2 所示，主要包括与客户端交互模块，DNS 报文解析模块、非法域名过滤模块、缓存检索模块、转发域名请求模块、接收 DNS 应答模块、查询结果存入缓存模块以及异步机制管理模块。图中实线表示的是 DNS 请求流程，虚线表示的是 DNS 应答流程。各个模块通过相应的控制流和数据流进行相互协作，完成 DNS 代理缓存的存储和检索功能。MFBDP 与现有常见的 DNS 代理缓存结构相比，最大的不同在于查询结果存入缓存模块：MFBDP 将域名和应答 IP 地址进行聚合存储，同时在存储的过程中利用基于概率的缓存替换方法进行缓存表项的替换

更新。

控制流说明如下：

- C1: 将用户进行域名请求的 DNS 查询报文传给 DNS 报文解析模块。
- C2: 缓存检索模块中若没有命中查询域名，则通知转发域名请求模块进行域名转发查询。
- C3: 缓存检索模块中若命中查询域名，则将查询结果通知客户端交互模块进行域名响应。
- C4: 接收 DNS 请求应答模块将本地缓存域名服务器的 DNS 应答报文传给 DNS 报文解析模块。
- C5: 缓存检索模块中若命中查询域名，则将查询结果通知客户端交互模块进行域名响应。
- C6：将解析出的非 A 类型的域名查询请求通知转发域名请求模块，进而由转发域名请求模块将域名请求转发到运营商部署的缓存域名服务器。
- C7: 将解析出的非 A 查询类型应答结果传给客户端交互模块，通知客户端交互模块向用户响应。

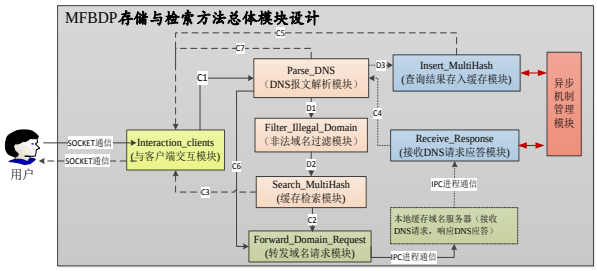

图 2 MFBDP 缓存存储与检索总体模块结构图

数据流说明如下：

- D1: DNS 报文解析模块从 DNS 请求报文中解析出的域名。
- D2: 通过非法域名过滤模块过滤后，符合标准 RFC 定义格式的域名。D3: DNS 报文解析模块从 DNS 应答报文中解析出的域名和所有应答 IP 地址。

各个模块的主要功能和交互过程详细说明如下：

(1) 与客户端交互模块

使用本 DNS 代理缓存的用户默认已知该代理缓存所部署机器的 IP 地址。首先用户端配好该代理缓存的 IP 地址，将域名请求封装在 UDP 报文中通过 socket 连接转发给客户端交互模块。客户端交互模块申请一个缓冲区空间 buffer，通过 socket 接口 recvfrom()接收 DNS 请求报文，将 DNS 请求报文存入缓存区 buffer 中。若接收请求报文内容失败，则断开 socket 连接，返回用户“$NXDOMAIN”的应答信息。若接收请求报文内容成功，则将存储 DNS 请求报文的缓冲区 buffer 传给 DNS 报文解析模块进行解析。

(2) DNS 报文解析模块

使用结构体 DNS_DRESPONSE 对 DNS 请求报文和 DNS 应答报文进行统一解析。结构体 DNSHDR 用于存储 DNS 报文头部信息，结构体 DNS_QUERYHDR 用来存储 DNS 报文的查询问题部分。pDnsHdr 用来记录 DNS 报文的首地址。首先解析 DNS 报文的头部信息，从首部地址开始依次取出首部中各项长度对应的字符。首部解析内容分别为标识，标志，问题数。通过识别标识中 Transmit ID 的值判断 DNS 报文类型。

若 DNS 报文的标识中 Transmit ID 值为 0，则表明 DNS 报文为查询报文。对查询报文的域名和记录类型进行解析。记录类型不为 1（非 A 类查询）时，将域名值传入转发域名请求模块接口。若查询类型为 A 类型，则将域名传入非法域名过滤模块接口。

若 DNS 报文的标识中 Transmit ID 值为 1，则表明 DNS 报文为应答报文。对应答报文的域名，回答资源记录进行解析，对权威、额外资源记录不进行解析。对所有回答资源记录解析出其应答类型、应答结果。将解析出的域名和所有应答 IP 列表传至结果缓存模块。

(3) 非法域名过滤模块

接收来自 DNS 解析模块的查询域名字符，将非法域名进行过滤。根据 DNS 定义标准，域名中的标号都由英文字母和数字组成，每一个标号不超过 63 个字符，也不区分大小写字母。标号中除连字符（-）外不能使用其他的标点符号。因此，DNS 代理缓存首先要过滤非法域名，即响应 $NXDOMAIN 的应答，为缓存域名服务器减轻查询负担和存储负担。若域名合法，则将域名传至缓存检索模块。

(4) 缓存检索模块

将合法域名进行分级，从顶级域名开始进行搜

索匹配。域名各级的检索过程为：将各级哈希表HashTable[0..n-1]看成是一个循环向量，若初始探查的地址为 a (hash(该级域) = a) ,若 a 不为空，则继续探查序列为：a，a+l，a+2，…，n-1，0，1，…，a-1。若找到空表项或地址为 a 项，该级搜索失败，将域名传给转发域名请求模块，调用域名转发函数进行域名请求；若找到该级域，则通过指针找到该域对应的下一级哈希表，重复此搜索过程，直到该域名完全匹配，得到应答 IP 地址，并将相应项的频数、时间进行更新。最后，将应答 IP 地址传给客户端交互模块，调用 socket 的发送数据接口向用户响应。

(5) 转发域名请求模块

按照图 2 中 DNS 报文的总体结构设计，构造 DNS 请求报文，将请求报文头部的 Transmit ID 设为 0，查询方式设为递归查询，查询类型设为 1（A 类型），问题数设为 1，资源记录数设为 0，权威资源记录数设为 0，额外资源记录数设为 0，查询域名按各级长度分隔。将构造好的 DNS 请求包发送给运营商分配的本地缓存域名服务器进行域名解析。

(6) 接收 DNS 请求应答模块

从本地缓存域名服务器接收 DNS 应答报文，将应答报文以字符形式存入缓冲区 buffer，传入 DNS 报文解析模块，调用 DNS 解析函数进行报文解析。接收 DNS 请求应答模块采用异步阻塞方式调用 recv()函数读取网络缓冲区中数据，读到数据后立即返回。若接收阻塞时间超过设定阈值(如 10 秒)，则不管数据是否成功读取都会立即返回，不会一直挂在该接收函数接口的调用上。

(7) 查询结果存入缓存模块

将 DNS 报文解析模块传入的域名和应答 IP 列表存入缓存中。由对离线 DNS 流量的分析可知，DNS 信息具有相同域名根域下不同子域个数较多且子域个数均值较为固定的特征，因此可利用多级哈希的表项分级、检索速度快等特性设计 DNS 代理缓存的存储技术。结构体 DOM_NODE 存储每一级的哈希表项。将域名进行分级，各级域名依次插入哈希表项中，同时更新各表项的频数和查询时间。在末级哈希表中将应答 IP 地址以{网络号：主机号列表}的方式进行聚合存储。按级进行域名插入的关键在于正在插入下一级时，上一级因为哈希表满而被选择删除，造成数据的不准确性。因此进行插入操作时，需对该域名的各级相关项进行加锁保护。缓存更新替换表项时只有在解锁状态下才能进行。

(8) 异步机制管理模块

DNS 代理缓存采用异步机制管理模块保证并发量大时，DNS 代理缓存的数据一致性和数据准确性。其主要作用于（1）接收 DNS 请求应答模块（2）查询结果存储模块。在（1）中，DNS 解析模块处理速度与接收 DNS 应答报文速度不匹配时，容易造成数据包的丢失。因此，通过其设置的调度队列对 DNS 应答报文进行缓存，依次从队列中取出应答报文调用解析函数进行报文解析。在（2）中，对同一个缓存域名项进行更新时，需要其加锁进行保护，防止数据出现错误。

2.2 存储数据结构设计

由于查询域名中超过 90%的域名为四级以下域名，哈希表级数可设为四级。四级哈希表项大小可根据分析的平均结果进行设置，从一级到四级哈希表可分别设为 100，700，30，25。由于高热度域名记录类型 80%为 A 类型且多应答 IP 中 70%的应答 IP 在同一 C 网段，因此，代理缓存存储 A 类型记录时，在末级哈希表项中可将多个应答 IP 按 C 网段进行聚合存储。存储各级哈希表项的结构体 DOM_NODE 如下所示：

```
struct DOM_NODE
{
   char *dom_name;                  /*某级域名*/
   struct IP_list iplist;           /*应答 IP 列表*/
   bool flag;                       /*是否末级哈希表*/
   double fre;                      /*出现频数*/
   double probability;              /*更新替换概率*/
   int   pre_time;                  /*上一次查询时间*/
   int next_count;            /*下一级哈希表项元素个数*/
   struct DOM_NODE * next;   /*下一级哈希表头部指针*/
}
```

结构体 DOM_NODE 各个参数的含义如表 1 所示。通过图 3 所示用例，本节对缓存存储结构及其工作原理进行详细阐述。若域名 Dom1 为“music.baidu.com”,将域名进行分级为{三级域：‘music’；二级域：‘baidu’；一级域：‘com’ }。若域名 Dom2 为“vip.vedio.baidu.com”,将域名进行分级为{四级域：’vip’; 三级域：’vedio’；二级域：‘baidu’；一级域：’com’}。将 Dom1 和 Dom2 的

各级域进行哈希映射，分级插入多级哈希表中，为　　各级哈希缓存项更新查询频数和查询时间。

表 1 结构体 DOM_NODE 字段说明

字段	类型	描述
dom_name	char*	域名被分级后，存储在某级哈希表中的分级域名。例如，域名 www.example.com，一级哈希表中的该值为“com”。
iplist	Struct IP_list	多个应答 IP 地址列表，其结构体包括两个属性：应答 IP 地址的公共网络号 netid 和主机号列表 hostlist。
flag	bool	用来标识该级域是不是域名的末级域，若为域名的末级域则该值设为 true，否则设为 false。
probability	double	哈希表项的更新替换概率，若某级哈希表项满，则根据这个值进行表项更新替换。
fre	double	某哈希表项的出现次数，随着各级域名的插入、查询而不断更新。
pre_time	int	某哈希表项的上一次查询时间，是替换概率计算的参数之一。
next_count	int	下一级哈希的表项个数，用来进行插入下一级域名的哈希 key 值计算。
next	Struct DOM_NODE*	指向下一级哈希表的首部地址，缓存项进行插入时，若下一级域的哈希表为空，则利用 malloc()函数进行下一级哈希表的申请，返回值即为新的首部地址；若下一级域为空，则设 next 值为 NULL。

由于 Dom1 和 Dom2 的二级以下域相同，因此可共享一级和二级域的哈希空间，减少了代理缓存的存储空间。若在插入 Dom2 的过程中，要同时插入 Dom1，则交集项“com”和“baidu”进行加锁保护，Dom1 插入进程暂缓插入，等待 Dom2 插入完毕后将交集项解锁信号释放，Dom1 才能插入成功。同一时刻有且仅有一个进程可以对更新域名的相关项设置保护锁。

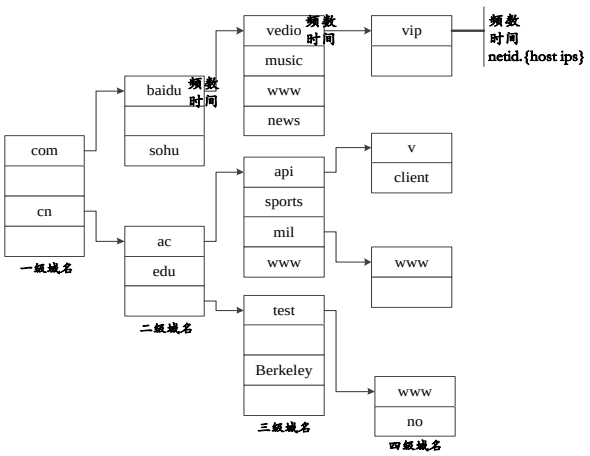

图 3 MFBDP 多级哈希存储结构

域名插入多级哈希表后，在末级哈希表项对域名的应答 IP 地址进行聚合存储，存储格式为”C 网段号.{主机号 1，主机号 2，主机号…}”，并将末级标记设为 true。 若哈希冲突，则采用线性法处理冲突，若哈希表满，则选择合适项进行表项的更新替换。

2.3 针对变前缀域名请求的优化方案

在域名并发请求量大的情况下，MFBDP 采用异步事件处理机制保证其具有一致的应答数据和较快的响应速度。然而，当存在大量的变前缀域名请求时，仍会给代理缓存域名服务器带来较大负担。当前的缓存域名服务器当遇到该类攻击时，其存储记录的变化频率会超过设定阈值，则缓存域名服务器就采取丢包或发假应答包的策略[7]，但是这样做会导致该域下正常的域名请求得不到正确解析，影响用户正常上网。因此，在本节中提出一种对变前缀域名进行最大公共模式抽取方法，通过分析域名前缀的公共模式，对相似域名的前缀进行聚合，降低由于大量变前缀域名请求带来的资源消耗，保证正常域名请求可以得到快速准确的响应。

在一定时间周期内某二级或三级域的“$NXDOMAIN”记录个数超过阈值，如平均请求速率为 1 万 QPS(Query Per Second)，MFBDP 则将其前缀中“$NXDOMAIN”的记录进行聚合，得到聚合记录进行缓存更新。在一定时间内（如一小时），若该域的变前缀域名再次请求且请求速率超过设定阈值，则通过匹配聚合记录直接向用户响应“$NXDOMAIN”的应答，提高了代理缓存域名服务器的鲁棒性，保证用户进行域名查询的稳定性、准确性。

通过引入通配符‘*’改进了最长公共子串算法[8]，提出一种变前缀域名最大公共模式抽取方

法，算法思路如下：

算法参数定义：

- 设 D1 和 D2 为同一个二级或三级域下的前缀，len1 和 len2 分别表示前缀 D1 和 D2 的长度。
- 设 MAXD1, D2 [m][n]是域名前缀 D1[1,m]和 D2[1,n]的最大公共模式匹配串。
- 通过改进最长公共子串算法来求解 D1 和 D2 的最大公共模式串 MAXD1, D2 [len1][len2]。

变前缀域名最大公共模式抽取算法的递推关系如下：

（1）递推初始条件：

当 n 或 m 为 0 时，MAXD1, D2[m][n]=Φ；

（2）递推关系：

- 当 D_1[m]=D_2[n]时

$$MAX\,D_1,D_2[m][n]=MAX\_STR\begin{Bmatrix}MAX\,D_1,D_2[m][n-1]+'*';\\MAX\,D_1,D_2[m-1][n]+'*';\\MAX\,D_1,D_2[m-1][n-1]+D_1[m];\end{Bmatrix}$$

- 当 D_1[m]!=D_2[n]时

$$MAX\,D_1,D_2[m][n]=MAX\_STR\begin{Bmatrix}MAX\,D_1,D_2[m][n-1]+'*';\\MAX\,D_1,D_2[m-1][n]+'*';\end{Bmatrix}$$

其中，递推关系中 MAX_STR{}为求集合中最大公共模式串函数，具体功能是将集合所有元素中的相邻通配符‘*’合并为一个，取包含非通配符‘*’字符最多的元素。

本文利用实例对变前缀域名请求的优化方案进行详细介绍，构造应答结果为“$NXDOMAIN”变前缀域名如表 2 所示，在部署代理缓存的局域网客户端连续发送变前缀域名请求，达到每秒 10000 的发送速度。DNS 代理缓存会将二级域“17717”中的应答结果为“$NXDOMAIN”前缀域名进行聚合，得到聚合域名“Zhan*.17717.com”。将缓存记录中的“17717”域下的前缀为“Zhan”的记录删除，仅保留聚合前缀域名“Zhan*”。此时若有新的域名“Zhandjhajsf.17717.com”的查询请求到来且二级域“17717”每秒的查询频数超过阈值，则代理缓存直接通过与聚合域名“Zhan*.17717.com”匹配成功，向用户响应“$NXDOMAIN”的应答。

这样做不仅减轻了代理缓存域名服务器在变前缀域名攻击时的递归查询压力，而且也保证了“17717.com”的前缀中不匹配“Zhan*”的域名前缀可得到正常的响应结果。

表 2 变前缀域名请求

变前缀域名
Zhanhaskjd.17717.com
Zhandajsfasdfasdf.17717.com
Zhanktajsfasdfasdf.17717.com
Zhanhfajsfhhj.17717.com
……

2.4 缓存更新优化方案

当代理缓存的哈希表满时需要对现有缓存中的记录进行更新和替换，现有的 DNS 代理缓存技术根据不同记录的生存周期信息对缓存记录进行更新替换操作。但根据对 DNS 信息中生存周期的分析可知，域名的热度与生存周期无关，基于生存周期的更新替换方式无法保证缓存尽早将低热度域名从哈希表替换出去。因此通过对现有的基于生存周期的缓存更新替换算法进行改进，本节将域名热度与缓存更新结合起来，提出一种基于表项概率的缓存更新替换算法。

通过表项概率将域名热度、缓存记录命中率、缓存记录准确率结合起来，保证高热度域名以较低概率进行替换，且保证应答结果的准确性。域名的热度与域名查询基数，域名查询频率有关，因此为各个哈希表项设替换概率计算方式如下：

$$p_n = p_{n-1} * e^{-a*(t_n - t_{n-1})}$$

缓存替换概率公式中各个参数定义以及功能描述如表 3 所示。

冷却因子 a 和时间差 t_n-t_{n-1} 会随着域名查询行为变化而做自适应的调整。例如某时刻的高热度域名随着时间推移，查询频数变小，则冷却因子 a 变大，时间间隔 t 变大。因此该域名的热度逐渐冷却而成为低热度域名，其替换可能性也会相应变大。

各个表项的概率计算参数与时间差、冷却因子 a 有关。时间差是一个确切的参数，但是冷却因子与查询频数的幂率关系参数 γ 是一个变量。

表 3 缓存替换概率公式参数描述

概率公式参数	定义	功能描述
p_{n-1}	代表上次替换时计算的哈希表项概率，初始为 1	缓存替换概率，用来判断缓存中域名是否需要被替换。p

p_n	代表当前替换时计算的哈希表项概率	越大代表域名的热度越高，被置换的可能性越小。
a	冷却因子，是对域名热度的描述	与查询基数 x 成反比，假设 a 的计算满足幂率公式 $a=e^{-\gamma x}$（γ 为幂率参数），高热度域名的查询基数越大，则相应 a 的值就越小。
t_{n-1}	代表上一次该项的查询时间	t_n-t_{n-1} 可以有效反映域名的查询频率。查询频率越快，时间间隔越小，域名热度越高，缓存替换概率 p 越大。高热度域名被置换的可能性越小。
t_n	代表当前该项的查询时间	

3 实验与分析

从某运营商高速网络中获取连续的 DNS 流量中的 10 万个域名，然后将 MFBDP 部署在范围可控的局域网中。在部署 MFBDP 的局域网的客户端上通过连续主动发起域名请求的方式对 DNS 代理缓存的命中率进行计算。

本文通过缓存命中率来选取最优的幂率关系 $a=e^{-\gamma x}$ 中的参数 γ 根据不同幂率曲线的变化趋势，选取有代表性的 γ 值如图 4 所示，分别是 γ =0.01， γ =0.1， γ =1， γ =10。

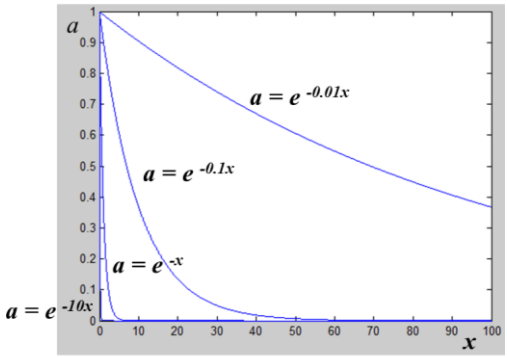

图 4 幂率关系随 γ 的变化趋势

主动探测 10 万域名的命中率结果如表 4 所示。从表中可以看出，随着曲线的长尾性越来越明显，缓存命中率逐渐增大，当 γ =1 时，缓存命中率最佳，为 80.3%。当 γ =10 时，幂率关系的长尾性最明显，但此时缓存命中率有所下降，因此，查询基数和冷却因子的幂率关系为 $a=e^{-x}$ 最佳，缓存命中率最高。

表 4 γ 变化对缓存命中率的影响

$a=e^{-\gamma x}$	缓存命中率
γ =0.01	67.78%
γ =0.1	77.79%
γ =1	80.30%
γ =10	73.62%

同时，为防止更新替换的抖动性，保证应答记录的准确性，代理缓存需以一定周期（如一月）进行全部更新。

6 总结

基于高热度域名的特征，本文提出一种基于最大公共模式抽取的 DNS 代理缓存方法。其中包括缓存存储与检索的总体模块设计、缓存存储结构设计以及变前缀域名请求时缓存优化设计。该方法减小了缓存的存储压力，保证了域名以较快速度响应，增强了域名查询行为的稳定性。通过实验验证基于概率的缓存替换算法对减小代理缓存的响应时延具有重要作用。

参考文献

[1] 王虎. 基于代理模式防御 DNS 欺骗攻击的研究与实现[D]. 北京: 北京邮电大学, 2015.

[2] 杨驩宇. 高并发环境中路由器平台上 DNS 代理的设计与实现[D]. 武汉:华中科技大学, 2012.

[3] Stolikj M, Verhoeven R, Cuijpers P J L, et al. Proxy support for service discovery using mDNS/DNS-SD in low power networks[C].15th International Symposium on World of Wireless, Mobile and Multimedia Networks, 2014: 1-6.

[4] Ballani H, Francis P. A simple approach to DNS DoS defense[C], Proceedings of HotNets. 2006.

[5] Pappas V, Massey D, Zhang L. Enhancing DNS resilience against denial of service attacks[C]. Annual IEEE/IFIP International Conference on Dependable Systems and Networks, 2007, 450-459.

[6] Ramasubramanian V, Sirer E G. The design and implementation of a next generation name service for the internet [J].ACM SIGCOMM Computer Communication Review, 2004, 34(4): 331-342.

[7] 王垚, 胡铭曾, 云晓春. DNS 权威名字服务器性能与安全性的研究[J]. 通信学报, 2006, 27(2): 147-152.

[8] Zou D, Long W J, Ling Z. Winnowing-Based Similar Text Positioning Method[C]. International Conference on Internet Technology and Applications. 2010, 1-5.