

基于 Sketch 的重要 DNS 查询缓存方法

郝逸航^{1,2}, 刘紫千^{1,2}, 常力元^{1,2}, 佟欣哲^{1,2}, 杨成^{1,2}, 孙琦^{1,2}, 郭俊言^{1,2}

(1. 中国电信股份有限公司, 北京 100001; 2. 天翼安全科技有限公司, 江苏 苏州 215131)

摘要: 在网络信息技术快速发展背景下, 域名系统作为互联网应用的入口, 其性能和安全对网络服务的质量尤为关键。然而, 从缓存的角度来看, 当前域名解析器的域名缓存项管理机制尚不完善。为此, 提出了一种基于 Sketch 技术的域名缓存管理方法, 创新地引入了“重要度”评估指标, 将网络测量方法应用于缓存管理, 实现对缓存项的时间和频率两方面特征的综合度量, 满足动态划定重要域名需求。实验结果表明, 该缓存方法能够从真实网络环境产生的海量域名查询请求中, 针对性地将重要域名添加到缓存予以保护, 平均解析时间相比现有域名缓存解决方案减少超过 18%。所提方法不仅能够对域名缓存实施更灵活的操作, 提高域名系统的用户侧可用性和管理侧可维护性, 更为域名缓存管理的研究提供了新的视角和解决方案。

关键词: 域名服务; 缓存性能优化; sketch; 域名查询更新

中图分类号: TP393

文献标志码: A

DOI: 10.11959/j.issn.2096-109x.2024069

Sketch-based caching method for significant DNS queries

HAO Yihang^{1,2}, LIU Ziqian^{1,2}, CHANG Liyuan^{1,2}, TONG Xinzhe^{1,2},
YANG Cheng^{1,2}, SUN Qi^{1,2}, GUO Junyan^{1,2}

1. China Telecom Co., Ltd, Beijing, 100001, China 2. China Telecom Network Security Technology Co., Ltd, Suzhou 215131, China

Abstract: With the rapid advancement of networking technology, DNS (domain name system), acting as the first step for most network behaviors, plays a vital role in assuring service quality through its capability and security. However, the current DNS resolvers lack effective mechanisms for cache item protection and eviction. Addressing this lacuna, a method for DNS cache oversight based on the principles of sketch technology was proposed. "Significance" was elegantly integrated as a novel evaluative indicator, and network measurement strategies were adopted into cache management. Consequently, a holistic benchmark that encapsulated both temporal and frequency characteristics for dynamically demarcating items' significance was culminated. The experimental results, which were based on a substantial amount of real-world network traffic data, demonstrated that the proposed method offered protection for significant domain names, hastening their resolution by an average of 18% relative to conventional

收稿日期: 2024-05-09; 修回日期: 2024-08-16

通信作者: 郝逸航, yihanghao2020@gmail.com

基金项目: 国家重点研发计划 (2022YFB3103004)

Foundation Item: The National Key R&D Program of China (2022YFB3103004)

引用格式: 郝逸航, 刘紫千, 常力元, 等. 基于 Sketch 的重要 DNS 查询缓存方法[J]. 网络与信息安全学报, 2024, 10(5): 95-106.

Citation Format: HAO Y H, LIU Z Q, CHANG L Y, et al. Sketch-based caching method for significant DNS queries[J]. Chinese Journal of Network and Information Security, 2024, 10(5): 95-106.

DNS paradigms. The usability for end-users was augmented by this strategy, as was the maintainability for Internet Service Providers (ISPs), while the scope for ongoing research in the realm of DNS cache management was simultaneously expanded.

Keywords: domain name system, cache performance optimization, sketch, domain name query update

0 引言

随着基于互联网的服务和应用的快速增长, 众多新兴技术和平台不断涌现, 如云计算平台、分布式网络服务等。域名系统 (domain name system, DNS) 作为互联网应用的“门牌号码”系统, 其性能和安全对网络服务至关重要^[1-2]。面对逐渐增大的 DNS 查询压力, 优化性能并减少时延的主要策略便是采用缓存机制, 在本地存储查询结果, 减少外部查询的次数, 从而减少解析时间和网络带宽的开销。

缓存项的有效性包括两个方面。一是时间有效性, 即该缓存项提供的信息是否已经过时, 根据每个缓存项附加的生存有效期 (time to live, TTL) 来决定是否删除该缓存项。二是频率有效性, 指某个缓存项被访问的时间间隔, 缓存项若长时间未被访问则将被删除。不难看出, DNS 缓存的管理与其缓存项时空访问模式的测量密不可分。目前的网络测量技术虽然已经相当完善, 但是没能被有效地用于 DNS 缓存的管理。当前 DNS 主流软件常用的缓存方法, 如 LRU (least recently used) 计数^[3]等, 通常关注时间维度上“最近”使用的元素, 并不直接关注频率维度上元素的“查询频率”。例如, 一个域名缓存项在很久以前频繁地被查询, 但近期并未被查询, 尽管它实际上需要被保留, 然而 LRU 机制会把该项删除。由此可见, 当前的 DNS 缓存机制在时间维度和频率维度的管理上两方面是割裂的, 且无法根据网络环境中网络设备的需求针对性地动态调整缓存度量指标和管理策略。这些不足之处带来了空间利用率不足、无法动态适应当前环境中波动的查询需求和模式等问题^[4]。

基于此, 本文使用网络测量技术为 DNS 缓存的管理赋能, 提出了一种基于 Sketch^[5]技术的 DNS 服务器缓存管理方法, 命名为 DNSSketch。Sketch (常译作“摘要”或“草图”) 是一种面向流式数据的压缩存储和快速统计的数据结构, 根据不同

的统计任务可以设计出不同的 Sketch 数据结构^[6]。近年来, Sketch 被证明能够以较小的运算开销和可接受的误差边界完成对网络流量的测量。

本文提出了一种基于域名缓存项“重要度”的缓存管理方法。重要度包括两个方面, 一是频繁度, 体现域名项在短时间内被查询的频繁性; 二是持久度的权重是不同的, 体现域名项在较长时间跨度上出现的持续性。不同的重要度评价场景下, 频繁度和持久度的权重不同。例如, 12306 在高并发时段的内容分发线路保护场景下, 会认为查询频率长时间较低但短时间内高的为重要域名。另外, 如政务服务、重要数据库定时同步等短时间访问量极小但长时间看来会持续产生重要访问的系统, 对于这类场景, 会认为高持久度而低频繁度的为重要域名。本文对重要度的定义为最普遍的场景, 即认为短时间内查询频率很高且长时间会持续出现的域名为重要域名。

本文基于 Sketch 设计了对缓存项重要度的测量和缓存方法, 旨在保留重要度最高的元素, 将重要度足够高的元素列为重要域名并“升级”到重点保护的高速缓存中。

本文主要有以下贡献。

1) 创新地提出了将重要度作为缓存项管理指标, 灵活满足对缓存项在时间维度和频率维度方面的各类组合判别需求。

2) 提出了一种全流程的域名缓存管理方法 DNSSketch, 将 Sketch 从计数器改进为多级缓存, 将测量任务迁移到缓存任务, 保障查询的同时完成缓存项的测量。

3) 在重要域名的保护和淘汰方面, 设计了针对重要域名缓存的淘汰和常规域名选拔机制, 根据查询的时空模式主动筛选更新, 在不影响常规缓存的情况下实现对重要域名的保护。

1 相关工作

1.1 DNS 缓存技术

目前, 主流的 DNS 软件常用的缓存策略大

多数是基于哈希表查找和基于 LRU 计数思想设计的。这种策略优先保留最近和最频繁使用的域名项,以便实现有限缓存空间的最大化利用。当 DNS 查询请求到达时,首先通过哈希表查找缓存中是否存在该请求所需的信息,若存在,则直接返回相应的 IP 地址并更新其在缓存中的位置,表示其为最近被访问过的条目;若不存在,那么 DNS 服务器将向上级服务器查询并获取结果,并将新的缓存记录添加到缓存中;如果此时缓存已满,按照 LRU 计数机制,缓存中最长时间未被访问的缓存项会被淘汰,新的 DNS 缓存项将被写入缓存空间。

这样的机制存在一些问题,对此,研究者提出了解决方案。Cohen 和 Kaplan 等^[7]针对当前 DNS 服务器采用的被动缓存管理机制存在的“只有在客户端请求后,才将记录带入缓存”的问题,提出了最优更新策略,即基于对未来请求序列分布的估计,动态规划计算最优的更新策略。Chen 等^[8]验证了 DNS 查询到达时间是泊松分布的随机过程,提出了名为 DNScup 的主动 DNS 缓存更新方法,实现假设快速响应和异常事件处理。还有一些研究旨在动态设置 TTL 值,以提高缓存的周转利用率。例如,Chen 等^[9]设计了 ECO-DNS,采用 EAI (expected aggregate inconsistency) 指标来衡量缓存项的更新频率和受欢迎程度,计算并更新缓存项 TTL 值,处理不可预见网络事件带来的突变,实现缓存性能和带宽开销之间的平衡。此外,随着随机域名的泛滥,大量一次性域名造成了严重的缓存空间无意义占用问题。对此,Hao 等^[10]使用域名的长度、字符组合、注册时间等域名的原生特征构建分类器,用来检测一次性域名,防止这类域名进入 DNS 缓存,浪费缓存资源。类似地,Yu 等^[11]通过长短期记忆 (long short-term memory, LSTM) 网络构建分类器,用来检测一次性域名,防止其进入缓存,并利用级联的 LRU 处理剩余的未丢弃域名,使缓存命中率提高了一倍以上。

综上所述,当前的 DNS 缓存策略尽管能对域名进行分级保护,但是不能兼顾频率域和时间域。而且这些分级保护措施^[12-13]以静态方法为主,无法根据域名查询模式动态地更新重点保护

域名列表。需要说明的是,本文所讨论的域名查询模式仅包括查询在时间分布上的特点,即域名查询的频繁度和持久度。针对上述问题,本文提出缓存项动态适应缓存的时间和频率域的访问模式,以测量为基础制定域名缓存和重要域名保护策略。

1.2 基于 Sketch 的数据流挖掘算法

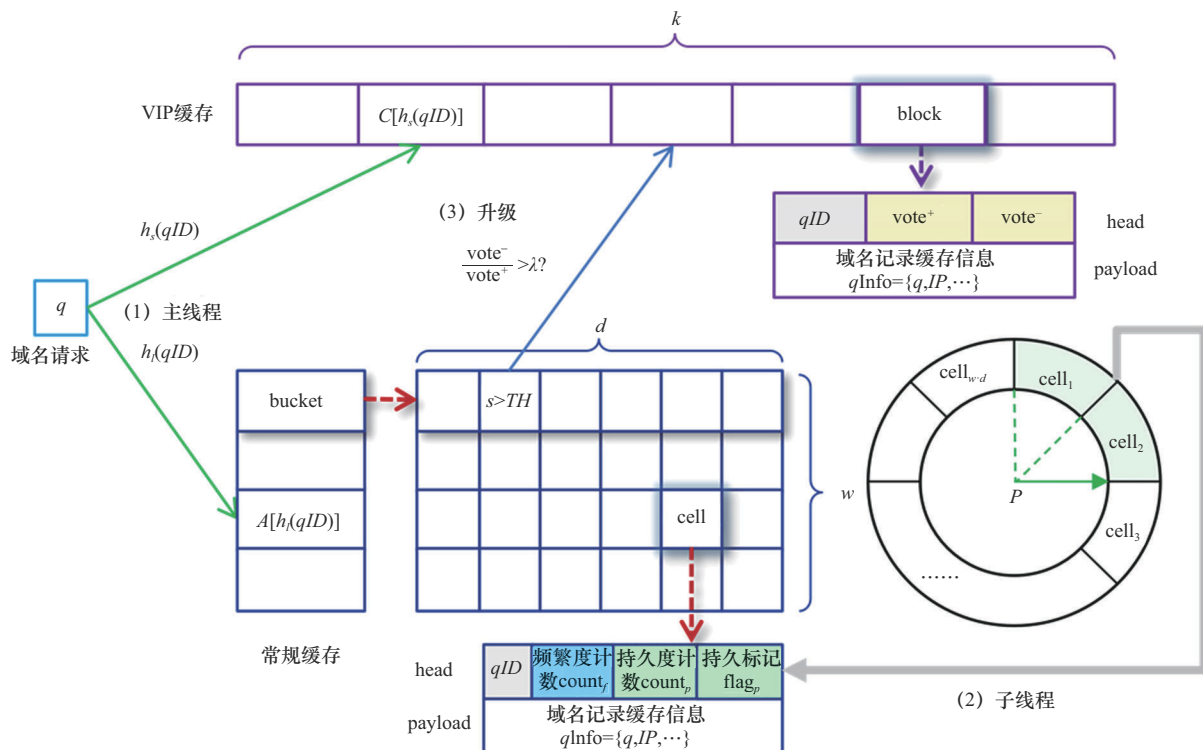
DNS 缓存项的评价离不开对 DNS 查询模式的测量与反馈。对 DNS 查询模式的测量与反馈是典型的流式统计任务,在这方面近年来涌现出了大量的研究成果,其中极具代表性的有 Sketch 算法。Elastic Sketch^[14]使用陶片驱逐法区分数据流的大小,将大型数据流存储在哈希表中,而小型数据流则存储在 CM-Sketch^[15]中。BurstSketch^[16]能够以较小的内存开销从数据流中检测出突变这一典型特征;ColdFilter^[17]和 WavingSketch^[18]算法通过改进传统的计数器和大根堆排序机制,增强了查询一定时间流数据中频繁项集的能力。LTC (long-tail clock)^[19]和 Elastic Sketch 等方法,面向满足 Pareto 或 Zipf^[20]等长尾分布的流式数据,这种分布的数据的特点是大多数样本的出现频率非常低,而少数样本的出现概率极高。根据 Jaeyeon 等^[21]的研究,这一特性恰好与网络中 DNS 查询请求的模式相符合。在 LTC 算法中,重要度 s 由待测缓存项出现频繁度 f 和时间持久度 p 共同决定,同时衡量缓存在两个维度的特性,形式上表示为二者的加权求和。

$$s = \alpha \cdot f + \beta \cdot p \quad (1)$$

式中, α 和 β 均为权重参数。通过灵活设置 α 和 β , 可以针对不同网络需求和场景评估域名缓存项的重要度。本文选取最普遍的场景,令 α 和 β 相等,即定义重要域名为频繁度高且持久度较高的域名。这样不但能够实现 DNS 查询模式的测量,以较小的时间复杂度完成对数据项统计指标的查找;而且拥有直观、明确的数据结构,在现有计数的数据域上添加信息域负载,将缓存项“挂载”到缓存项计数器后,实现查找的同时完成计数指标的更新。

2 本文方案

DNSketch 的数据结构与工作流程如图 1 所



长，链表足够短。

初始化常规缓存。为所有 bucket 中全部的 cell 设置初始化值： qID 字段设为-1，频繁度计数 $count_f$ 和持久度计数 $count_p$ 设为0，持久标记 $flag_p$ 设为False；负载信息中的 $qInfo$ 为空。

初始化VIP缓存区。为VIP缓存区的每一个 block 设置初始值： qID 设为-1，频繁度计数 $count_f$ 和持久度计数 $count_p$ 设为0；负载信息中的 $qInfo$ 为空。

为避免 DNSSketch 在长时间运行状况下，两个缓冲区的计数项持续无限增长，需要定期将所有计数项清空。按照 DNS 请求分布在整体上呈现的周期性与 DNSSketch 实际部署内存空间大小的相对关系，设置不同的清空周期。为保证服务的连续性，应尽量选择凌晨等查询量较小的时间进行清空操作。

(1) 主线程的工作流程

主线程的工作流程如图2所示，主线程是应对查询请求并返回其对应结果的线程，同时需要完成缓存查找、缓存内容的更新，以及为域名查询 q 对应的统计字段。重要度 s 的计算与频繁度

计数 $count_f$ 、持久标记 $flag_p$ 的更新相关。需要强调的是，持久度计数 $count_p$ 的更新不在该线程，而是由子线程中的时钟完成的。

如图2所示，当查询请求 q 到来时，首先使用映射函数计算其唯一对应的数值ID，记为 qID_i ，并分别使用两个相互独立的哈希函数 $h_l(\cdot)$ 和 $h_s(\cdot)$ 计算 qID_i 对应的两个哈希值 $h_l(qID)$ 和 $h_s(qID)$ ，同时分别将 qID_i 映射到常规缓存对应的 bucket 和 VIP 缓存对应的 block，分别记为 $A[h_l(qID)]$ 和 $C[h_s(qID)]$ 。

接下来，查找VIP缓存。若 $C[h_s(qID)]$ 不为空，且 $C[h_s(qID)]$ 中 qID 值为待查项 qID_i ，则查找成功，第一时间返回 $C[h_s(qID)]$ 中的 $qInfo$ 信息到用户查询侧，并更新该 $C[h_s(qID)]$ 中的投票计数： $vote^+$ 的值加1，表示增加其保留在VIP缓存的支持程度。反之，若 $C[h_s(qID)]$ 中 $qID \neq qID_i$ 或若 $C[h_s(qID)]$ 为空，则转去查找常规缓存。需要根据 $A[h_l(qID)]$ 可能出现的3种不同情况完成缓存更新，具体如下。

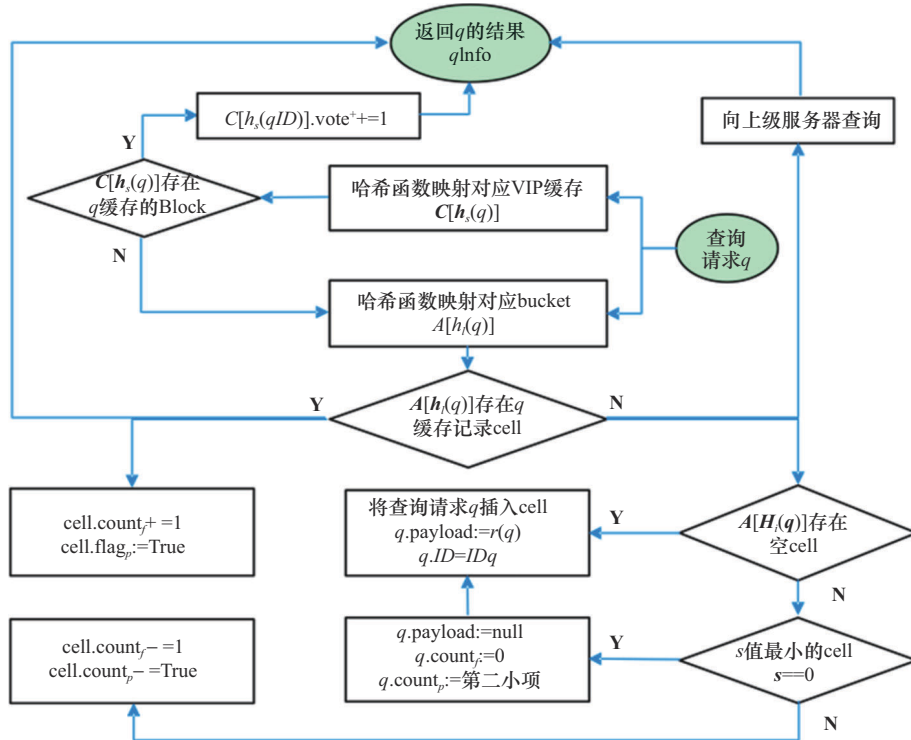


图2 主线程的工作流程

Figure 2 Operation process of main thread

1) 若 $A[h_l(qID)]$ 中存在 qID 值为 qID_i 的 cell, 记为 $A[h_l(qID)][j]$, 表示该查询请求 q 是一个已知记录, 则将 $qInfo$ 字段信息返回给用户侧。然后, 更新统计字段: 将 $A[h_l(qID)][j]$ 的频繁度计数 $count_f$ 加 1; 持久标记 $flag_p$ 设为 True, 表示该周期内该查询出现过, 作为子线程更新持久度计数 $count_p$ 的依据。

2) 若 $A[h_l(qID)]$ 中不存在 qID 值为 qID_i 的 cell, 表示该查询是未知记录。首先, 向递归服务器查找查询请求 q 的信息 $qInfo'$, 返回给用户查询侧, 保障查询的时效性。然后, 判断 $A[h_l(qID)]$ 中是否存在空 cell。若存在, 记为 $A[h_l(qID)][j]$, 并将该查询请求 q 插入该 cell 中, 变为已知记录。同时, 将 $A[h_l(qID)][j]$ 中的 qID 置为 qID_i 、频繁度计数 $count_f$ 置为 1、持久标记 $flag_p$ 设为 True、 $qInfo'$ 置为 $qInfo'$ 。

3) 若 $A[h_l(qID)]$ 中不存在 qID 值为 qID_i 的 cell, 且 $A[h_l(qID)]$ 中没有空 cell, 该 bucket 已满, 则需要根据重要度 s 判断是否将该查询添加进常规缓存, 具体如下。

① 计算 $A[h_l(qID)]$ 中所有 cell 中 s 最小的 cell 和倒数第二小的 cell, 分别记为 $A[h_l(q)][j]$ 和 $A[h_l(q)][k]$, 其中 s 值的计算式为

$$s = \alpha \cdot count_f + \beta \cdot count_p \cdot \delta(flag_p) \quad (2)$$

s.t. $\alpha + \beta = 1$

其中 α 和 β 是权重超参, 在本文中令两个参数值为 0.5, $\delta(\cdot)$ 为指示函数, 其形式为

$$\delta(flag_p) = \begin{cases} 1, & flag_p = \text{True} \\ 0, & \text{其他} \end{cases} \quad (3)$$

表示当且仅当持久标记 $flag_p$ 为 True 时, 函数值为 1, 在计算重要度 s 时持久度计数 $count_p$ 值才有效。

② 完成有条件替换。分两种情况: 其一, 若 $A[h_l(q)][j]$ 的 s 为 0, 表示 $A[h_l(q)][j]$ 此时相对已不再重要, 则将新的查询项替换原有项, 其中频繁度计数 $count_f$ 和持久度计数 $count_p$ 的值分别设置为 $A[h_l(q)][k]$ 中的频繁度计数 $count_f$ 和持久度计数 $count_p$ 的值, 这是根据文献[22]中

的理论 (海量 DNS 请求在分布上满足长尾分布的特性) 给出的最优解; 其二, 若 $A[h_l(q)][j]$ 的 s 尚不为 0, 则对这个 cell 执行重要度衰减操作, 即把 $A[h_l(q)][j]$ 中频繁度计数 $count_f$ 和持久度计数 $count_p$ 的值各减 1。

(2) 子线程的工作流程

使用一个周期指针更新常规缓存中所有 cell 的持久度计数 $count_p$ 。 $w \times d$ 个 cell 按照图 1 右侧所示的形式重构为循环队列。确定更新周期 T (由该缓存所提供服务的 DNS 服务器的查询规模决定), 表示经过时间 T , 指针扫过全部的 cell。

初始化指针 P , 开始时指针指向 $cell_0$, 按顺时针方向移动逐一扫过 $w \times d$ 个 cell, 每次更新指针位置后, 指针的位置前进 $\frac{(w \times d)}{T}$ 个 cell。通常更新的时间间隔为 1 s, 这样既符合直观认知, 又不过度消耗系统资源。对于每个扫到的 cell, 记为 $cell_i$, 扫到之后需要更新常规缓存 $cell_i$ 的持久度计数 $count_p$ 。执行 $cell_i.count_p = cell_i.count_p + \delta(flag_p)$, 表示当且仅当 $cell_i$ 中的 $flag_p$ 为 True 时, 根据主线程中情况 1) 的原理, 表示当前更新周期内 $cell_i$ 记录的缓存项是被查询访问过的, 则将 $cell_i$ 中持久度计数 $count_p$ 的值加 1。当指针 P 扫过全部的 cell 后, 将所有 cell 中的持久标记 $flag_p$ 置为 False。表示当前观测周期已结束, 从下一观测周期开始, 重新记录“某元素在该观测周期内是否出现”的信息。

(3) 升级线程的工作流程

当某常规缓存 s 值已经超过阈值 TH_s 的时候, 该缓存项 q_v 就需要被列为重要域名项, 根据情况从 $cell_i$ 中升级。同样, 使用 $h_s(qID_v)$ 映射到 VIP 缓存对应的 block, 记为 $C[h_s(qID_v)]$ 。接下来分两种情况。

1) 若 $C[h_s(qID_v)]$ 为空, 则直接将 q_v 插入, 并将 $C[h_s(q_v)]$ 中的 qID 置为 qID_v 、 $qInfo$ 置为 $cell.qInfo$, 最后将统计字段 $vote^+$ 的值设为 1。

2) 若 $C[h_s(qID_v)]$ 不为空, 表示该 block 中已有 VIP 缓存项。需要更新 $C[h_s(qID_v)]$ 的统计字段, 并根据 $\frac{vote^-}{vote^+}$ 是否超过倍数阈值 λ 来决定

新缓存项是否替换原缓存项。若不超过 λ ，则保留原缓存项，但需将其评价指标 vote^- 的值加1，表示增加其逐出VIP缓存的支持程度。若超过 λ ，则说明该VIP记录已多次被碰撞，需要对原缓存项进行替换。 $C[h_s(qID)]$ 的 qID 置为新缓存项 qID_v 、 $qInfo$ 置为 $\text{cell}.qInfo$ ，将 $C[h_s(qID)].\text{vote}^+$ 置为替换之前 $C[h_s(qID)]$ 的 vote^- 值，表示保留先前多次碰撞导致未能记录的查询次数，完成替换。

可以看出，由于DNSSketch的计数、更新等步骤都被设计在独立线程中完成，因此不影响查找的时间性能。首先，查找VIP缓存，其时间复杂度为 $O(1)$ ；其次，在LTC缓存中，理论时间复杂度平均为 $O(n/w)$ ，其中最好情况为 $O(1)$ 、最坏情况为 $O(d)$ 。此外，由于VIP缓存和常规缓存的查找是并行的，因此整体平均时间复杂度小于 $O(n/w)$ ，当满足条件 $w \gg d$ 时，查找的时间开销理论上来看极小，且最坏情况下的时间开销也在可接受范围内。

2.2 重要域名缓存性能的分析

本节分别从收敛阶段和稳定阶段讨论缓存项成功从常规缓存升级到VIP缓存的概率。

首先，讨论碰撞率。对于VIP缓存中的某个block，令block中元素的个数为 z 。由于一个block只能存储一个元素，因此若某个block上发生碰撞，则等价于 $z > 1$ 。当待插入元素个数为 n 时，则 $z \sim B(n, 1/K)$ 。根据二项分布公式

$$Pr(z = i) = \binom{n}{i} \left(\frac{1}{K}\right)^i \left(1 - \frac{1}{K}\right)^{n-i} \quad (4)$$

计算插入VIP缓存时发生碰撞的概率 $P_c(n)$ ，可以等价于计算 $Pr(z > 1)$ ，即

$$\begin{aligned} P_c(n) &= Pr(z > 1) = \\ &= 1 - Pr(z = 0) - Pr(z = 1) = \\ &= 1 - \left(1 - \frac{1}{K}\right)^n - \left(\frac{n}{K}\right) \left(1 - \frac{1}{K}\right)^{n-1} \end{aligned} \quad (5)$$

当 K 足够大时，有如下近似：

$$\begin{aligned} \left(1 - \frac{1}{K}\right)^n &= \exp\left\{n \cdot \ln\left(1 - \frac{1}{K}\right)\right\} = \\ \exp\left\{n \cdot \left(-\frac{1}{K} - O\left(\frac{1}{K^2}\right)\right)\right\} &\approx e^{-\frac{n}{K}} \end{aligned} \quad (6)$$

因为满足“ n 足够大， $1/K$ 足够小”的条件，所以此处近似计算形式上可以理解为二项分布近似为 $\lambda = n/K$ 的泊松分布，其结果一致。因此，当待升级元素个数 n 足够大时，可以将式(6)代入式(5)，从而式(5)可以近似简化为

$$P_c(n) = 1 - e^{-\frac{n}{K}} \left(1 + \frac{n}{K}\right) \quad (7)$$

对式(7)求关于 n 的偏导：

$$P_c'(n) = \frac{\partial}{\partial n} P_c(n) = \frac{n}{K^2} e^{-\frac{n}{K}} \quad (8)$$

显然，当 n 、 K 为正数时，式(8)恒为正，且单调递减，则可知 $P_c(n)$ 单调递增，且为凸函数。在VIP缓存收敛阶段，缓存项升级进入VIP缓存的成功概率主要是由碰撞决定的，因此可知该概率逐渐减小，但减小幅度逐渐变慢，最终碰撞概率达到最大值。

接下来分析VIP缓存达到稳定状态阶段，该阶段碰撞概率达到或者已接近最大值，缓存项 q 能否升级到VIP缓存主要取决于新缓存项能否替换碰撞到的原缓存项 q_0 。由于映射到该block的缓存项不是唯一的，因此定义集合 S 表示所有待升级到VIP缓存的常规缓存项，其中每个项的查询频率为 f_j ，对应地，该block上的原缓存项 q_0 的查询频率为 f_0 。若升级条件为满足 $\text{vote}^-/\text{vote}^+$ 超过倍数阈值 λ ，那么根据本文方法的设计，这一比值取决于在当前观测周期新缓存项查询频率 f_j 与原缓存项 q_0 查询频率 f_0 的相对增长速率。定义block上 vote^+ 和 vote^- 各自的增长速率为 R_p 和 R_N 。假设缓存项 q_i 的查询频率 f_i 在长时间跨度上几乎不存在概念漂移，在理想情况下可以被表示为，期望为常数 $E_f(f_i)$ 的平稳随机过程，原缓存项查询频率 f_0 就是 vote^+ 的增长速率， vote^+ 增长速率的期望便是 f_0 的期望；而 R_N 的增长速率取决于两方面，一方面是集合 S 中查询频率为 f_j 的平均值，另一方面为是否映射到碰撞block的概率，使用式(7)表示为 $P_c(\#S)$ ，进而对应期望计算为

$$\begin{cases} E_f(R_p) = E_f(f_0) \\ E_f(R_N) = P_c(\#S) \sum_{j \in S} E_f(f_j) \end{cases} \quad (9)$$

若满足2.1节升级线程工作流程中的VIP缓

存升级条件, 则需满足 $E_f(R_N) > \lambda E_f(R_P)$, 定义这一事件发生的概率为 $\Pr\{\text{upgrade}\}$, 根据马尔可夫期望不等式^[23]计算这一事件发生概率的上界, 即升级事件发生的概率最大不超过的边界值。

$$\Pr\{\text{upgrade}\} = \Pr\{E_f(R_N) > \lambda E_f(R_P)\} \leq \frac{E_n[E_f(R_N)]}{\lambda \cdot E_f(R_P)} = \frac{E_n\left[P_c(n) \sum_{f \in \mathbb{S}} E_f(f_j)\right]}{\lambda \cdot E_f(f_0)} \quad (10)$$

其中, $E_n[P_c(n)]$ 可以根据式 (6), 近似计算为

$$E_n[P_c(n)] \approx E_x\left[\frac{e^{-n/K} \cdot (n/K)^n}{n!}\right] = \frac{n}{K} \quad (11)$$

然后, 将式 (9)、式 (11) 代入式 (10) 可得

$$\Pr\{\text{upgrade}\} \leq \frac{\#S \cdot \sum_{f \in \mathbb{S}} E_f(f_j)}{\lambda K \cdot E_f(f_0)} \quad (12)$$

根据式 (12) 可以定性分析参数设置和网络环境与测量需求之间的关系。①当 K 值足够大时, K 的变化对 $\Pr\{\text{upgrade}\}$ 的影响微弱, 即 VIP 原缓存项 q_0 被新缓存项替换的概率与 VIP 缓存大小几乎无关。② λ 值的设定对 $\Pr\{\text{upgrade}\}$ 影响较大, λ 越大, 原缓存项被替换出 VIP 缓存的概率越小, 但 VIP 缓存容易陷入短时间内的局部最优而难以迭代到长时间跨度的全局最优状态。

3 实验设计与结果

3.1 实验环境与数据准备

本文实验将验证 DNSsketch 算法的基础性能, 并将其与两种常见的 DNS 服务程序 ISC-BIND^[24]、DNSmasq^[24] 进行对比。

ISC-BIND^[24] 是目前各种大型 DNS 服务器使用最多的软件, 基本上可以看作 DNS 的工业标准。DNSmasq^[24] 是一款适用于小型网络的 DNS 缓存和配置软件, 通常运行在计算资源有限的设备上, 如家用路由器和小型单片机等。本文实验运行环境为 ThinkPad X13 Gen2 (Intel7-1260P, 内存 16 GB), 操作系统为 Windows 11。为保证性能测试的公平性, 所有程序都是运行在单一 CPU、相同大小内存的虚拟环境中。其中, ISC-BIND 版本为 v9.18.18, DNSmasq 版本为 v2.84,

并统一将递归模式修改为转发模式, 上游转发服务器和 DNSsketch 保持一致。

本文实验使用的测试数据集有以下 3 种。

1) ISP-20W: 大规模 (约 20 万条) 真实 DNS 请求, 采自运营商骨干网, 能够反映网络中大部分 DNS 请求的特点。

2) CAMP-48H: 中等规模的真实 DNS 请求, 采自国内某大学网络实验室的出口路由器, DNS 查询的平均速率约为 10 次/秒。该数据集可以验证 DNSsketch 在处理周期性明显、时间跨度较大的查询任务时的缓存性能。

3) VIP-100: 人为挑选构建的重要域名数据集, 共 200 个样本。数据来自 Alexa^[25] 访问排名前 200 条, 并将其中不可访问的域名用中国网站排行榜中的域名补位。

实验使用的所有数据均已进行脱敏处理, 不包含网络用户个人信息、客户端 IP 等任何隐私信息, 且不涉及任何数据伦理性问题。

在评价 DNS 性能时, 常用的指标有缓存 QPS (queries per second)、转发 QPS、解析时延、解析超时无响应率等。由于本文关注的缓存性能侧重于对重点域名的保护方面, 因此主要取决于硬件条件的 QPS 指标不在本实验中讨论。

本文共设计了 3 类实验: ①短时间、高频率请求时, DNSsketch 的缓存空间收敛性实验; ②长时间、低频率请求时, DNSsketch 的跨度缓存有效性实验, 对比不同 DNS 软件的缓存应答率、平均解析时间和解析超时无响应率变化; ③经过长时间查询后, 对常见重要域名的覆盖性能实验。

3.2 缓存空间收敛性实验

本实验旨在评估 DNSsketch 内部各缓存空间的有效性和稳定性。实验使用的评价指标有命中率和空置率。命中率指当前时刻之前的查询中由某缓存 (如 LTC 缓存) 应答的查询请求的占比。空置率指某一缓存中没有存储缓存项的最小存储单元的占比。根据 DNSsketch 的设计, LTC 缓存需要对 VIP 缓存项进行筛选, 所以越满越好; VIP 缓存需要保证最佳查询性能, 根据哈希表的查找原理, 其空置率在 50% 左右最佳。

本实验使用 ISP-20W 数据集进行重放测试, 实验结果进行滑动平均处理后得到图 3 和图 4 所

示的曲线。图3和图4的横轴一致对应,表示时间域上逐渐累积的DNS查询,图3表示VIP缓存、LTC缓存和转发查询的命中率,图4表示VIP和LTC两个缓存的空置率。

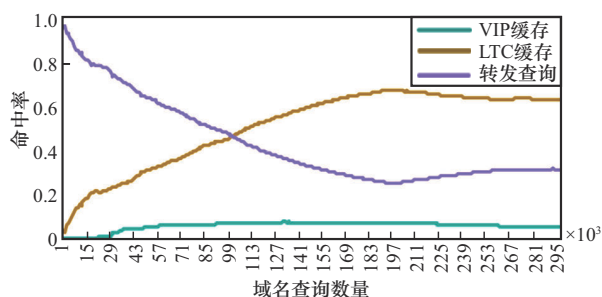


图3 缓存空间的命中率随DNS查询的累积的变化趋势
Figure 3 Trend of the hit ratio of caches' evolution with the accumulation of DNS queries

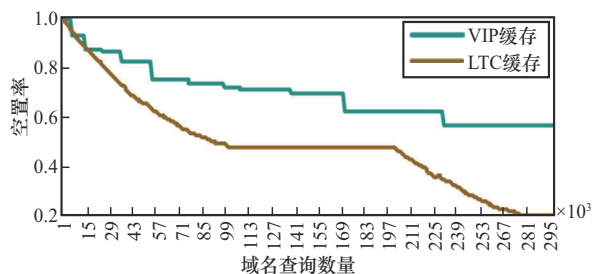


图4 缓存的空置率随DNS查询的累积的变化趋势
Figure 4 Trend of the empty rate of caches' evolution with the accumulation of DNS queries

如图3所示,实验开始时缓存为空,大部分查询请求要转到上游服务器进行查询。随着查询的累积,LTC缓存的命中率线性上升,VIP缓存的命中率缓慢上升,因此,查询上游服务器的频率线性下降。最终,LTC缓存和VIP缓存的命中率都逐渐趋于稳定。对应地,如图4所示,LTC缓存的空置率线性下降,而VIP缓存的空置率则是逐步下降,这体现了方案“LTC缓存宽进严出、VIP缓存严进严出”的设计思想。LTC缓存的空置率在趋于稳定后进一步下降,是因为出现了大量缓存中没有的新DNS请求,这与图3所示后半程转发查询的命中率出现上升趋势原因一致。

实验结果表明,VIP缓存和LTC缓存的命中率变化符合预期模式,重点域名升级能够体现VIP缓存的变化规律,各缓存的空置率能够稳定收敛,并在各自期望的空置率上稳定运行。

3.3 跨度缓存有效性实验

本实验旨在评估DNSketch相较于其他DNS软件(ISC-BIND和DNSmasq)在较长时间跨度上缓存的有效性。实验使用CAMP-48H数据集,由于数据集中Day1和Day2是几乎对等的两个工作日,因此本实验的基本假设是两天中网络环境中用户和设备的网络行为产生的查询内容和查询体量呈现一定程度的重复性和周期性。CAMP-48H数据逐小时查询量变化如图5所示,每小时平均最低查询量不低于10 000次,查询量通常在下午时段开始上升,夜间时段达到顶峰,规律以天为单位呈现周期性。将该数据集完整重放,分别使用3台解析器完成解析,记录缓存应答率、平均解析时间和解析超时无响应率3个指标。

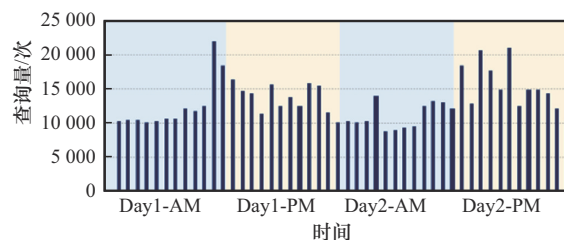


图5 CAMP-48H测试数据逐小时查询量变化
Figure 5 Variation in hourly query quantity of CAMP-48H test data

(1) 缓存应答率

缓存应答率指某时刻前查询请求由缓存应答的查询量占比。缓存应答率越高,则缓存的有效性越高。考虑到本实验对比对象无法如实验3.2直接在软件中跟踪查询项的来源,对此采用的解决办法是测量响应时间与上游转发服务器的最小网络往返延迟(round-trip time, RTT)的相对大小,若响应时间少于RTT最小值,则断定该查询是由本地缓存提供的响应而没有被转发到上游服务器。

为使实验结果看起来更直观,本实验对缓存应答率进行了滑动平均处理。各DNS软件的缓存应答率48 h的变化如图6所示,在Day1-AM,DNSketch、ISC-BIND和DNSmasq的缓存逐渐被填满,并各自开始按照缓存管理策略淘汰过期的缓存项。DNSmasq和ISC-BIND简单地将TTL到期的缓存项删除,而DNSketch能够根据重要度将更重要的缓存项筛选出来并加以保护。当TTL达到更新阈值时,DNSketch会在后台查询递归服

务器更新该缓存项,而不改变该缓存项在缓存中的位置和其他计数值。实验进行到 24 h 时,该网络环境中产生的多数域名均已被保护在缓存中,而没有因简单的 TTL 到期就被删除,因此 DNSketch 相比 DNSmasq 和 ISC-BIND 在缓存应答率方面能够持续保持超过 11% 的领先优势。

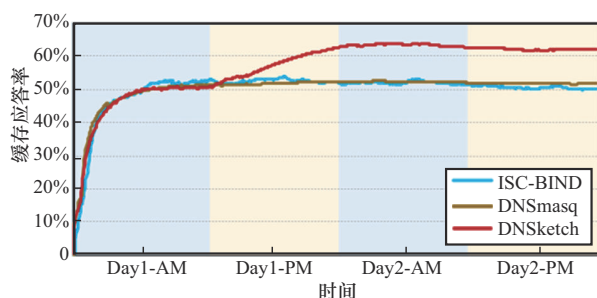


图 6 各 DNS 软件的缓存应答率 48 h 的变化
Figure 6 Variation in 48h cache response rate of various DNS software

(2) 平均解析时间

将实验结果以每小时为粒度进行滑动平均处理,能够直观地看出较长时间范围的变化趋势。各 DNS 软件的平均解析时间 48 h 的变化如图 7 所示,实验开始时, DNSketch、ISC-BIND 和 DNSmasq 的解析时间开销都较大, ISC-BIND 和 DNSmasq 快速收敛到稳定状态,而 DNSketch 收敛到稳定状态的过程用时较长。但从第 12 h 开始, DNSketch 的缓存命中率持续保持领先态势,使得其收敛后的平均解析时间比 ISC-BIND 少约 0.774 ms、比 DNSmasq 少约 1.465 ms。

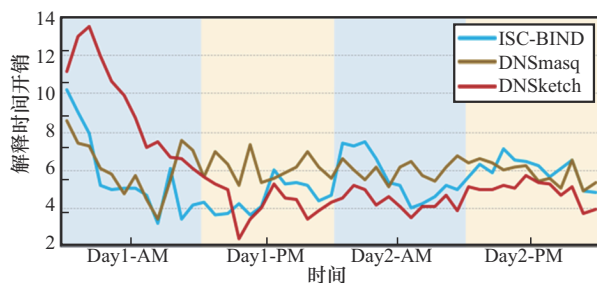


图 7 各 DNS 软件的平均解析时间 48 h 的变化
Figure 7 Variation in 48h average resolution time of various DNS software

(3) 解析超时无响应率

同样,将实验结果以每小时为粒度进行滑动平均处理。各 DNS 软件的解析超时无响应率 48 h

的变化如图 8 所示,实验开始时, DNSketch、ISC-BIND 和 DNSmasq 的解析超时无响应率波动下降,当 ISC-BIND 和 DNSketch 收敛到稳定状态时,两者的解析超时无响应率几乎一直低于 DNSmasq。但相比之下, DNSketch 的解析超时无响应率从 Day2 稳定后就持续保持在最低水平,平均比 ISC-BIND 低 0.149%、比 DNSmasq 低 0.246%。

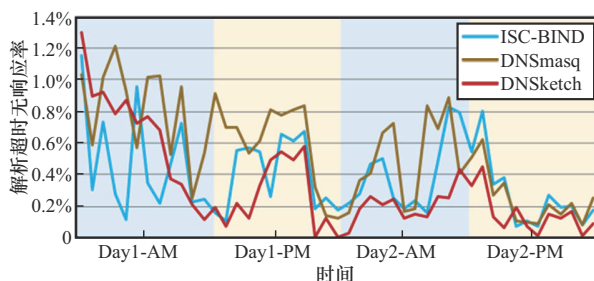


图 8 各 DNS 软件的解析超时无响应率 48 h 的变化
Figure 8 Variation in 48h consumption no-response rate of various DNS software

缓存应答率、平均解析时间、解析超时无响应率这 3 个指标中,缓存应答率是因,解析时间是果,缓存应答率的提升能够减少时间开销。综合分析实验结果可以看出, DNSketch 虽然前期收敛到稳定状态耗时较长,但随着缓存应答率的逐渐提升,平均解析时间显著减少。实验结果验证了在长时间跨度上 DNSketch 缓存的有效性和可用性。同时,结合实验 3.2 的结果可以看出, DNSketch 并不是盲目地牺牲空间换时间,而是通过对重要域名在缓存空间的高效利用,在缓存的空间占用与时间开销两个维度上找到平衡点。

3.4 重要域名的覆盖性能实验

在完成前面两项实验后, DNSketch 已达到稳定状态。本实验旨在评估 DNSketch 的缓存算法对重要域名的保护效果和 VIP 缓存的利用效果。实验选取 VIP-100 作为正样本,从办公网络环境中采集真实 DNS 请求 100 000 条作为负样本,分别导出 DNSketch 不同缓存区中的缓存域名予以比较。重要-非重要域名在 VIP 和 LTC 缓存中的占比如表 2 所示。

重要域名大部分存储在 VIP 缓存中,其余没进入 VIP 缓存的是境外域名(如 amazon.com、msn.com、ieee.org 等),国内使用较少,因此这些域名并未存储在缓存中,由上游 DNS 提供解析。

对于非重要域名，大多数保存在LTC缓存中，少数是全新的域名，所以没被缓存。此外，还有一些在VIP缓存中的负样本，经检查发现某些视频直播网站的CDN域名，由于出现频率和持久度较高，同样升级到了VIP缓存。实验结果表明，DNSketch能够以70.59%的召回率从海量查询中捕捉到重要域名查询的时空模式，并保存到VIP缓存中予以保护，覆盖程度足够高；进入VIP缓存的域名有92.31%是重要域名，没有盲目地浪费缓存空间。

表2 重要-非重要域名在VIP和LTC缓存中的占比

Table 2 Proportion of VIP/non-VIP domain name in VIP and LTC caches

测试数据	VIP 域名	非VIP 域名
LTC 缓存	4.90%	68.97%
未缓存	24.51%	39.22%
VIP 缓存	70.59%	5.89%

4 结束语

本文基于Sketch技术设计了一种面向重要域名保护的DNS服务器缓存管理方案，将网络测量技术与DNS缓存项的重要度评估相结合。与现有DNS缓存技术相比，本文主要有以下贡献。首先，引入了重要度作为DNS缓存项管理评价指标，解决了现有ISC-BIND和DNSmasq等广泛使用的DNS软件对域名的缓存管理简单依赖于TTL和查询频率，以及缺乏针对域名缓存项的频繁度和持久度两方面的评价指标和管理策略问题。其次，提出了面向重要域名缓存的淘汰选拔和重要保护机制，能够根据查询特征自动地筛选出重要域名并保障其解析效率，进而允许域名解析软件对重要域名实施覆盖更全面、解析更高效的保护手段。实验结果表明，DNSketch方案能够在缓存空间利用率和重要域名解析效率之间找到平衡。VIP缓存的空置率能够收敛到稳定水平，LTC缓存能够从海量解析中有效地检出重要域名；DNSketch收敛到稳定状态后，解析超时无响应率降低0.149%~0.246%，平均解析时间相比现有主流缓存软件减少0.774~1.465 ms（减少约18%以上）。在重要域名的保护性能上，本文方案在运行达到稳定状态后，能够保护覆盖70%以上常见重要域名。

未来，将从以下两个方面开展研究，以进一步优化该方案。其一是研究缩短缓存系统冷启动的“预热”时间，即利用先验知识对DNSketch进行初始化，进而更快地收敛到稳定状态而不需要经过长时间的等待；其二是研究控制缓存各个超参数与缓存空间大小、查询任务并发数的关系，以适应不同服务规模的域名服务器的查询需求和保护规模。这些研究可以将特征信息来源进一步扩展，除了查询特征，还可以结合语义特征、域名注册信息和PDNS（passive DNS）数据等先验知识，进行更深入的分析研究。

参考文献：

- [1] 胡宁, 邓文平, 姚苏. 互联网DNS安全研究现状与挑战[J]. 网络与信息安全学报, 2017, 3(3): 13-21.
HU N, DENG W P, YAO S. Issues and challenges of internet DNS security [J]. Chinese Journal of Network and Information Security, 2017, 3(3): 13-21.
- [2] 王文通, 胡宁, 刘波, 等. DNS安全防护技术研究综述[J]. 软件学报, 2020, 31(7): 2205-2220.
WANG W T, HU N, LIU B, et al. Survey on technology of security enhancement for DNS [J]. Journal of Software, 2020, 31(7): 2205-2220.
- [3] O'NEIL E J, O'NEIL P E, WEIKUM G. The LRU-K page replacement algorithm for database disk buffering[J]. ACM SIGMOD Record, 1993, 22(2): 297-306.
- [4] PERDISCI R, ANTO M, LUO X, et al. WSEC DNS: protecting recursive DNS resolvers from poisoning attacks[C]//International Conference on Dependable Systems & Networks. IEEE/IFIP, 2009: 3-12.
- [5] 戴冕, 程光. 基于Sketch的软件定义测量数据平面硬件模型[J]. 通信学报, 2017, 38(10): 113-121.
DAI M, CHENG G. Sketch-based data plane hardware model for software-defined measurement[J]. Journal on Communications, 2017, 38(10): 113-121.
- [6] 冯辉. 基于Sketch结构的网络流量测量方法研究[D]. 济南: 山东大学, 2021.
FENG H. Research on network traffic measurement methods based on sketch structure[D]. Jinan: Shandong University, 2021.
- [7] COHEN E, KAPLAN H. Proactive caching of DNS records: addressing a performance bottleneck[J]. Computer Networks, 2003, 41(6): 707-726.
- [8] CHEN X, WANG H N, REN S, et al. DNScup: strong cache consistency protocol for DNS[C]//Proceedings of the 26th IEEE International Conference on Distributed Computing Systems. IEEE, 2006: 40-49.
- [9] CHEN C, MATSUMOTO S, PERRIG A. ECO-DNS: expected consistency optimization for DNS[C]//Proceedings of the 35th International Conference on Distributed Computing Systems. IEEE, 2015: 256-267.
- [10] HAO S, WANG H N. Exploring domain name based features on the effectiveness of DNS caching[J]. ACM SIGCOMM Computer Communication Review, 2017, 47(1): 36-42.
- [11] Yu G, Zhang Y, Cui H, et al. Mitigating negative impacts on DNS caches caused by disposable domain names[C]//Proceedings of the Symposium on Computers and Communications. IEEE, 2019: 1-6.

- [12] YAN Z, HU A, WANG W. A cache-splitting scheme for DNS recursive server[C]//Proceedings of the 2nd International Conference on Cloud Computing and Intelligence Systems. IEEE, 2012: 1267-1271.
- [13] YU H S, LIU Y, DUAN L H, et al. Cache top-level domain locally: make DNS respond quickly in mobile network[C]//Proceedings of the International Conference on Trust, Security and Privacy in Computing and Communications. IEEE, 2022: 1205-1210.
- [14] YANG T, JIANG J, LIU P, et al. Adaptive measurements using one elastic sketch[J]. IEEE/ACM Transactions on Networking, 2019, (99):1-16.
- [15] CORMODE G, MUTHUKRISHNAN S. An improved data stream summary: the count-min sketch and its applications[J]. Journal of Algorithms, 2005, 55(1): 58-75.
- [16] ZHONG Z, YAN S, LI Z, et al. BurstSketch: finding bursts in data streams[C]//Proceedings of the International Conference on Management of Data. ACM, 2021: 2375-2383.
- [17] ZHOU Y, YANG T, JIANG J, et al. ColdFilter: a meta-framework for faster and more accurate stream processing[C]//Proceedings of the International Conference on Management of Data. ACM, 2018: 741-756.
- [18] LI J, LI Z, XU Y, et al. WavingSketch: an unbiased and generic sketch for finding top-k items in data streams[C]//Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. ACM, 2020: 1574-1584.
- [19] CHENG S, YANG D, YANG T, et al. LTC: a fast algorithm to accurately find significant items in data streams[J]. IEEE Transactions on Knowledge and Data Engineering, 2022, 34(9): 4342-4356.
- [20] KUMAR A, SUNG M, XU J, et al. Data streaming algorithms for efficient and accurate estimation of flow size distribution[J]. ACM SIGMETRICS Review, 2004, 32(1): 177-188.
- [21] JAEYEON J, SIT E, BALAKRISHNAN H, et al. DNS performance and the effectiveness of caching[J]. IEEE/ACM Transactions on Networking, 2002, 10(5): 589-603.
- [22] YANG T, ZHANG H, YANG D, et al. Finding significant items in data streams[C]//Proceedings of the 35th International Conference on Data Engineering. IEEE, 2019: 1394-1405.
- [23] STEIN E M, SHAKARCHI R. Princeton lectures in analysis[M]. Princeton University Press Princeton, 2003.
- [24] MAN K, QIAN Z, WANG Z, et al. DNS cache poisoning attack reloaded: revolutions with side channels[C]//Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2020: 1337-1350.
- [25] BARCELÓ-ARMADA R, CASTELL-UROZ I, BARLET-ROS P. Amazon Alexa traffic traces[J]. Computer Networks, 2022, 205: 108782.

[作者简介]



郝逸航 (1998-), 男, 天津人, 主要研究方向为网络流量安全、递归解析安全、机器学习。



刘紫千 (1979-), 男, 湖南长沙人, 博士, 中国电信股份有限公司高级工程师, 主要研究方向为路由安全、递归解析域名安全。



常力元 (1984-), 男, 吉林吉林人, 博士, 中国电信股份有限公司高级工程师, 主要研究方向为软件安全、递归解析域名安全、5G关键技术。



佟欣哲 (1981-), 男, 辽宁沈阳人, 中国电信股份有限公司副高级工程师, 主要研究方向为流量安全、递归解析域名安全。



杨成 (1984-), 男, 辽宁葫芦岛人, 中国电信股份有限公司中级工程师, 主要研究方向为网络流量编排、递归解析安全。



孙琦 (1985-), 男, 山东淄博人, 主要研究方向为递归解析安全、深度报文分析。



郭俊言 (1990-), 河北承德人, 博士, 主要研究方向为新型密码算法设计、隐私保护、国产密码算法应用、域名安全、天地一体网络。