



专题:信息中心网络(ICN)

内容中心网络中名字查找技术的研究^{*}

刘 斌,汪 漪

(清华大学计算机科学与技术系 北京 100084)

摘 要:内容中心网络作为一种新型的未来网络体系架构被提出,以满足当前互联网信息共享的需求。内容中心网络使用类似域名的层次化名字结构对内容进行标识、路由和查找。由于互联网中内容众多,使用名字前缀构建的路由表,比传统的 IP 路由表大 2~5 个数量级,且由于名字查找依旧遵循最长前缀匹配原则,使得实现高速名字查找是一个富有挑战性的难题。分析了名字查找的技术挑战、实施难点,介绍了主要技术方法以及当前在名字查找领域的主要研究成果。

关键词:内容中心网络;名字查找;最长前缀匹配

doi: 10.3969/j.issn.1000-0801.2014.09.002

Research on Name Lookup in Named Data Networking

Liu Bin, Wang Yi

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract: Named data networking (NDN) proposed as a new future architecture for the internet, aims to satisfy the requirement of information sharing among the users. NDN applies the hierarchical names to identify contents, route packets and retrieve information. Given the massive amount of contents and the longest prefix matching principle, achieving high speed name lookup exhibits a big challenge to implement NDN in real networks. The technical challenges of name lookup were firstly analyzed, then the main techniques to achieve wire-speed name lookup and their corresponding research results were introduced.

Key words: named data networking, named lookup, longest prefix matching

1 引言

以 IP 数据分组交换为核心技术的互联网运行机制已广泛应用超过 20 年,这是因为 IP 自身的简单性降低了网络互联成本,并具有很强的适应力。但从技术发展的角度反思,互联网最初是为了追求网络的互联,以实现硬件资源的共享为目标。由于最开始的通信需求发生在两台实体设备间,为了确定设备的具体位置,使用 IP 地址标识以支持设备间的数据通信。然而,随着技术的进步和信息化的

普及,硬件共享的必要性已经不大,信息共享成为主要目标。目前,网络应用的主体已经转向文字信息、图像和视频,内容服务已经成为网络服务的主体。用户关注的不再是内容存储在哪里(where),而是内容本身(what)以及内容检索与传输的速度、质量和安全性。目前的 IP 网络架构仍然根据主机地址进行信息内容的检索和传送,这在适应上层应用的需求变化方面显得低效。

在这种情况下出现了 CDN(content delivery network,内容分发网络)、P2P 等技术,这些技术表明 IP 网络已有向

^{*} 国家高技术研究发展计划(“863”计划)基金资助项目(No.2013AA013502),国家自然科学基金资助项目(No.61373143, No.61432009, No.61402254),中国博士后科学基金资助项目(No.2014M550734),清华大学自主科研计划资助项目(No.20121080068)

以内容为中心的网络架构转变的迹象。但任何在目前IP架构下进行的内容传送机制改进都无法彻底克服IP本身存在的缺陷:需要进行复杂的“内容”到“位置”的映射。因此,内容中心网络(named data networking,NDN)被提出并很快成为研究的热点。它直接把内容作为网络处理的基本对象,将数据的存储地址(即当前网络中的节点位置信息)、安全性和可访问性与内容本身分离开来,并且以名字作为对数据索引的唯一依据。整个网络基于名字的最长前缀匹配进行路由。

由于NDN把名字作为对数据索引的唯一依据,而且整个网络基于名字的最长前缀匹配进行路由,本文将讨论NDN中的一项关键技术——最长前缀匹配名字查找,它是建造未来NDN路由器最关键的使能技术之一。由于NDN中的名字查找具有一些固有的特殊性,使得基于名字的最长前缀匹配查找成为一项极富挑战性的研究工作,表现在以下4个方面。

- NDN的内容名字由数字和字符组成的字符串构成,如/org/sigcomm/2013/cfp.pdf/v1/s0/0x3f4edc19。相比于32 bit的IP地址,内容名字是不固定长度的,而且字符串长度也没有上限;此外,内容名字的查找比单纯的0、1比特组成的IP地址查找复杂得多。
- 由名字前缀组成的NDN路由表将可能会比目前IP路由表规模大2~5个数量级。目前统计的活跃的Website数量已经超过了 10^8 个^[1],而当前骨干网路由器的路由表大约为500 KB。面对数量如此大的NDN路由表,需要精巧的存储压缩技术将其存储空间大大压缩,否则将会占用巨大的存储器,不仅会提高成本,而且会潜在地影响查找速度。
- 骨干网链路的速率一直在持续提高。目前,40 Gbit/s的OC-768接口已经商用,100 Gbit/s的高速以太网接口也已经开始在实际网络中部署,160 Gbit/s的OC-3072传输接口也将在不久的将来投入使用。名字查找结构和算法需要能胜任如此高速率的查找要求。
- NDN路由表还需要解决快速增量式路由更新的问题,这是因为除了传统的网络拓扑改变和网络管理引起的路由策略更新外,NDN还需要应对另外一种路由更新操作,即当发布或删除内容时所引起的路由更新,预期这种操作所引发的更新频率会较高。因此,快速实时路由更新是NDN路由查找所要解决的又一难题。

本文就该方面已经开展的工作做了介绍。

2 名字查找技术分类与主要性能指标

2.1 名字查找技术类别

名字查找技术,从本质上讲是字符串的最长前缀匹配,可将不同的查找方法划分为3类,具体介绍如下。

(1) 基于TCAM的查找方法

TCAM(ternary content addressable memory)^[2]在IP查找中被广泛使用,一个时钟周期就可以实现一次查找。然而,在名字查找中,由于名字长度很长、长度可变,需要多个TCAM串联才能存放一个名字,不仅造成空间浪费,而且需要多个时钟周期才能完成一次查找;另一方面,名字路由表规模过大,需要大量的TCAM存放,成本过高。因此,现在学术界和工业界的研究主要集中在基于软件的名字查找技术。

(2) 基于字符查找树的查找方法

字符查找树可以将名字路由表构建为一个树型结构,一次名字查找即从根节点出发,根据查找的名字,沿树根节点向叶子节点方向依次搜寻,直到找到最长匹配的前缀。基于字符查找树的名字查找技术的典型代表为:基于GPU加速的一维存储压缩表查找方法^[3]和基于词元编码的名字查找技术^[4]。

(3) 基于散列技术的查找方法

内容中心网络中的名字由词元组成,一个名字虽然有很多个字符,但构建名字的词元个数有限,可以将具有不同词元的名字前缀划分为不同的集合,然后利用散列技术实现快速的名字查找。这方面的代表工作为:基于布隆过滤器(bloom filter)的快速名字查找技术^[5]和基于完美散列技术的快速名字查找方法^[6]。

2.2 主要性能指标

评价名字查找方法性能优劣的主要指标如下。

- 名字路由表的存储占用。对相同大小的表来说,存储需求越小越好。
- 名字路由表的查找速度,即查找吞吐量。在使用代价相比拟的情况下,查找速度越快,性能越好,单位是每秒完成的分组查找数量,通常使用MSPS(million search per second)表示。
- 名字路由表的更新速度。在保证查找吞吐量的前提下,更新速度越快,表明路由器响应路由变化的能力越强。



- 名字查找结构和算法的可扩展性。所提出的查找结构和算法的性能,不应该随路由表的大小和流量的动态特性变化而变化较大。算法的可扩展性表明算法对环境的适应能力。

3 基于 GPU 加速的一维存储压缩表查找方法

下面先介绍一种称为多步进对齐转移数组(multiple aligned transition array, MATA)的查找结构及其算法^[3]。该方法充分利用 GPU (graphic processing unit, 图形处理器) 的大规模并行处理能力来加速名字查找, 验证在现有商业设备上实现线速名字查找的可行性, 具有以下几方面的技术创新。

- 设计、实现、评估了基于 GPU 的名字查找引擎。虽然这是实验性研究, 但实验结果证明了在现有的低成本商业化 GPU 设备上实现大规模名字路由表的线速查找的可行性。
- 基于 GPU 的名字查找引擎使用了一种命名为 MATA 的新技术来构建名字路由表。实验结果表明, 相比采用传统的二维状态转移表, MATA 不仅压缩了 2 个数量级的存储空间, 而且具有 10 倍以上的加速比。
- 多工作流流水机制用于减少名字查找时延, 并保证系统吞吐量。GPU 中的名字查找以批处理的方式进行, 虽然批处理方式能够极大地增加名字查找的吞吐量, 但也会急剧增加每个名字的查找时延。为解决名字查找吞吐量与时延的两难困境, 基于 GPU 的名字查找引擎充分利用 NVIDIA Fermi 架构^[7] GPU 的多工作流机制, 以降低名字查找时延。基于多工作流机制的流水解决方案能降低名字查找的时延 (小于 0.1 ms), 同时保持名字查找的高速度。

GPU 名字查找引擎使用名字交织存储方式^[8]优化名字存储方式, 通过适应 GPU 单指令流多数据流 (single instruction multiple data, SIMD) 的工作方式来减少内存访问次数, 从而大幅度改进名字查找的性能。

3.1 基本数据结构

构建 NDN 名字路由表最直接、最自然的方法是使用字符查找树 (character trie)^[9]。字符查找树从本质上来讲, 就是一个在名字路由表中进行名字查找的有限状态机 (finite state machine, FSM)。如图 1 所示, 字符查找树中的

一个节点等价于有限状态机中的一个状态, 字符查找树中的一条边等价于有限状态机里的一个状态转移边。最开始, 假设有限状态机一次状态转移处理一个输入字符, 则在这种一步长的有限状态机中, 每个状态具有 256 个状态转移边, 每条状态转移边对应一个输入字符。整个有限状态机可以使用图 1 的二维状态转移表来表示。在二维状态转移表中, 第 i 行表示第 i 个状态 S_i , 第 j 列表示第 j 个字符 C_j (假设字符采用 ASCII 字符集)。二维状态转移表的第 i 行、第 j 列中的存储数值即当状态 S_i 遇到字符 C_j 时应该跳转的下一个状态的地址; 如果该数值为 0, 则表示匹配不成功。二维状态转移表使用的状态编号为行地址, 输入字符为列地址, 一次状态转移仅需一次访存, 具有最优的查找性能。

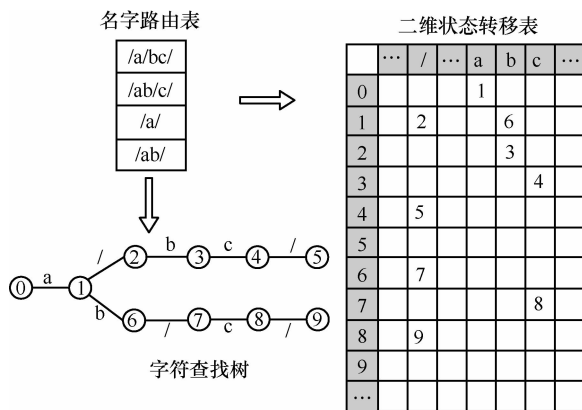


图 1 字符查找树转化为二维状态转移表

显然上述结构内存利用率低, 原因是二维表将会很稀疏。将二维状态转移表转换为一个一维表, 如图 2 所示, 但会看到有表项的冲突发生, 计算偏移量+ASCII 码的方法预先将这种冲突消除, 使其保持一次存储得到一次查找结果的特性, 称该表为基础 ATA (aligned transition array)。为了提高查找的速度, 可以一次查找多个字符, 如图 3 所示。

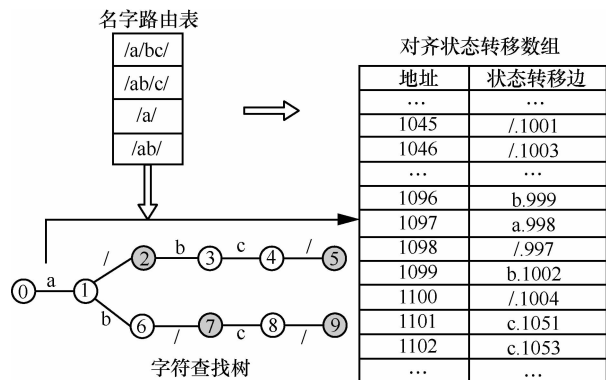


图 2 字符查找树转化为对齐转移数组的示例

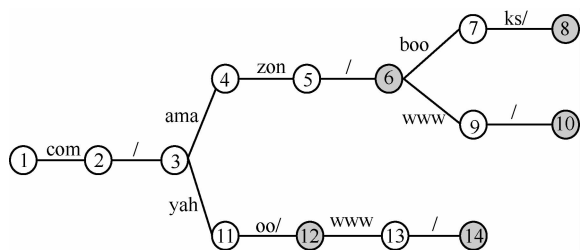


图 3 多步进字符查找树

基础 ATA 确实能够有效地减少内存开销和加快查找速度,但存在一个固有的设计问题,即步进的数目受实际内存大小的限制。为打破基础 ATA 中对步进大小的限制,设计并提出了多步进对齐转移数组(MATA)。MATA 主要具有以下 3 点创新。

- 将原先的一个 ATA,划分成多个子 ATA。
- 每个子 ATA 都有特定的长度 L ,且 $L < 2^d$,这里 d 表示每次查找的字符个数,即步进的大小。
- 在基础 ATA 中使用 $(S+y)$ 计算状态转移边存储的位置, S 表示状态的编号, y 表示输入字符串的值。在 MATA 中,由于每个子 ATA 的长度有限,采用 $(S+y) \bmod L$ 来计算状态转移边存储的位置。但如果状态

S 的另一条状态转移边 z 有 $(y-z) \bmod L = 0$,则会造成两条状态转移边 y 和 z 都映射到同一个位置,造成冲突。幸运的是,多个不同长度的子 ATA 可以解决该问题。

3.2 使用 GPU 多工作流水机制

GPU 芯片内嵌上百个 CPU 核,具有强大的并行计算能力,使用 GPU 加速路由查找是一个很自然的想法。但向 GPU 送入数据时,最好是批处理的大块数据,才能充分发挥其能力。因为名字查找是一个实时的线速操作,积攒大批数据将会使分组在路由器上的时延加大。为此,采取流水机制,即让送入名字数据、GPU 内核查找和返回结果流水工作,而又使得 GPU 的核不空闲,以取得最大吞吐量。图 4 示意了该思想。基于 GPU 的名字查找引擎系统框架如图 5 所示。由 CPU 和 GPU 协同工作,前者负责准备和送入数据(对 MATA 实施更新操作),后者负责批处理查找操作。

3.3 主要性能结果

图 6 和图 7 分别显示了查找吞吐率和查找时延的实验结果。结果表明,在不超过 $100 \mu s$ 时延下,查找引擎可以取得 55.65 MSPS 吞吐率,相当于 111 Gbit/s 的线路传输

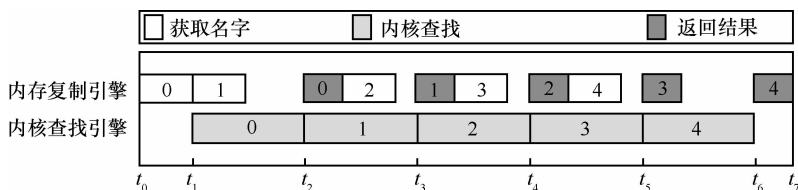


图 4 多工作流水机制示意

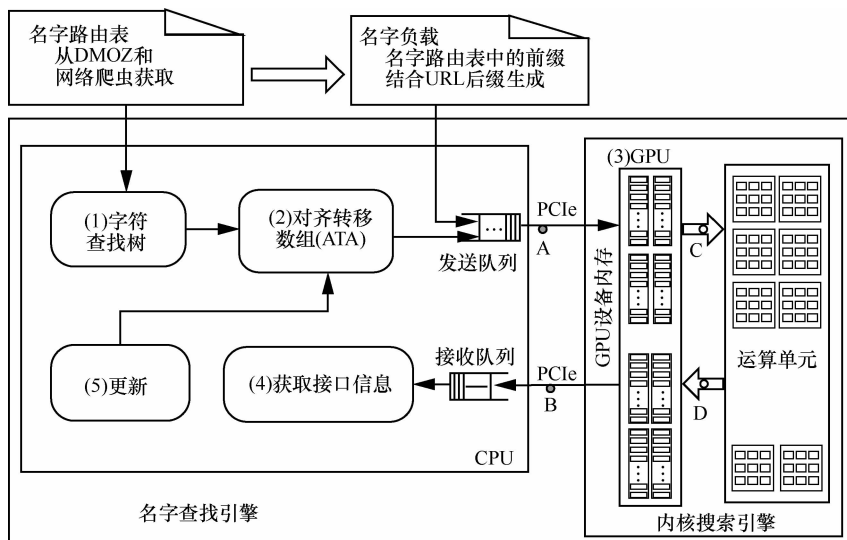


图 5 基于 GPU 的名字查找引擎系统框架



速率(假设数据分组的平均长度为 250 byte)查找。在 10M 名字路由表(含有 10^7 个名字前缀的路由表)中,至少能插入 30 000 个名字前缀,或删除 600 000 个名字前缀。

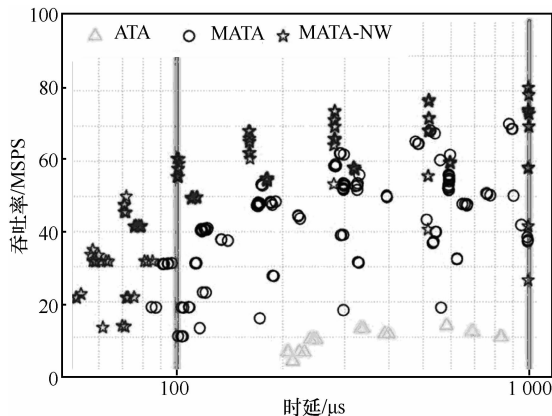


图6 名字查找吞吐率与时延(10M名字路由表,重负载)

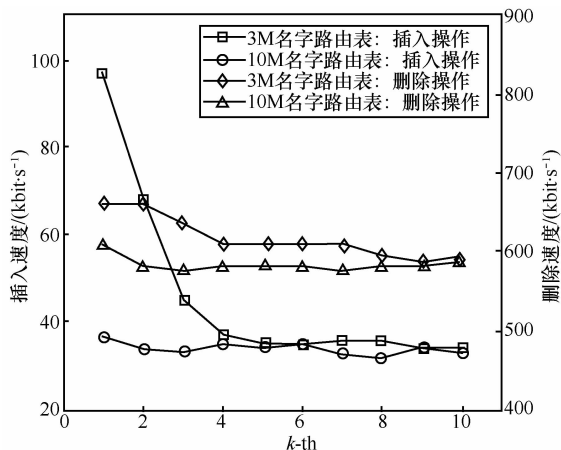


图7 名字查找引擎随着名字路由表规模增加的插入、删除速度的趋势

4 基于布隆过滤器的快速名字查找方法

布隆过滤器^[10]能够用于判断一个名字是否在一个名

字集合中。具有相同词元个数的名字前缀可被划分为同一名字前缀集合,并用布隆过滤器表示。相比散列表,布隆过滤器具有低存储开销的优势。使用布隆过滤器构建名字路由表,可以克服大规模名字路由表存储开销过大的难题。基于布隆过滤器的名字查找方法有如下两点优势:高内存效率,和直接存储所有的名字表项相比(如基于字符查找树的结构),布隆过滤器能够显著减小存储空间;快速的查询速度,对面向字符串的布隆过滤器进行改进后,可以实现布隆过滤器的快速查询,从而加速整体的名字查找速度。

基于布隆过滤器的方法有以下两个潜在问题。

- 布隆过滤器存在很小的误报率,数据分组偶尔会被转发到一些不必要被转发的端口。幸运的是,这种情况发生的概率可被控制,通常可以把发生概率控制在 10^{-8} 以下。另外,数据分组错误转发的负面影响很小,只会带来一些无害的冗余流量,且这些流量可以被多路径 NDN 通信模型^[11]所容忍。
- 标准布隆过滤器只支持高效的查询和插入,不支持删除。为支持增量式更新(包括删除),在路由器的控制层面,计数布隆过滤器(counter bloom filter, CBF)^[12]被用于辅助记录名字路由表,并用于映射成最终的标准布隆过滤器。

4.1 基本数据结构

4.1.1 NameFilter 框架

名字查找的目标是在名字路由表中查询得到最长的匹配前缀。初步实验表明,简单地利用布隆过滤器不能满足日益增长的对名字查找速度的需求。因此,提出 NameFilter——一种基于两级布隆过滤器的名字查找方法,其体系结构如图8所示。第一级布隆过滤器用来确定

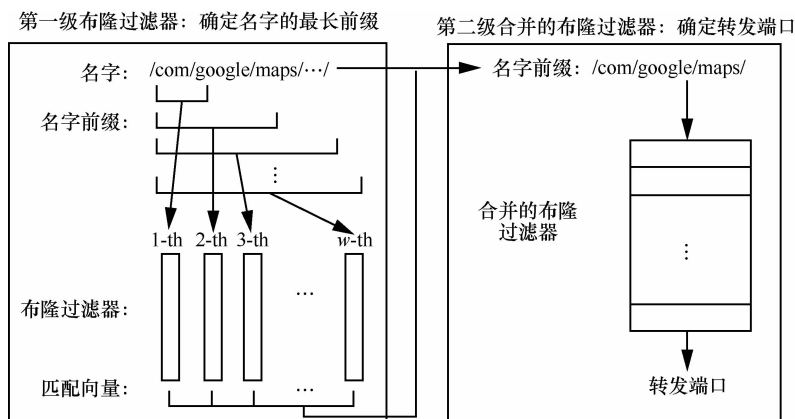


图8 NameFilter 体系结构

名字可能匹配的前缀,即将最长前缀匹配问题转化为精确匹配问题;第二级布隆过滤器用于判断第一级中匹配的最长前缀可以转发的端口。

NameFilter 的第一级由多个布隆过滤器组成,具有相同词元个数的前缀存储于同一个布隆过滤器中。当一个名字进入 NameFilter 进行查找时,基于它会生成多个前缀。例如,NameFilter 会基于名字“/com/google/maps/”生成 3 个可能的前缀“/com/”、“/com/google/”和“/com/google/maps/”。这 3 个前缀分别在存储有名字词元个数为 1 个词元、2 个词元、3 个词元的布隆过滤器中进行查询。图 8 中假设“/com/google/maps/”存储于对应的布隆过滤器中。

NameFilter 的第二级亦由多个布隆过滤器组成,一个端口对应一个布隆过滤器,具有相同转发端口的名字前缀存储于同一个布隆过滤器中。具有多个转发端口的名字前缀,则会在所有可以转发的端口中进行存储。在第二级中使用散列表存储名字前缀信息会引起性能问题,不仅消耗大量的存储空间,而且不能达到高速名字查找的速度要求。和散列表相比,布隆过滤器是另外一种组织名字集合的方法,实际上也是一种可以在误报率和存储开销之间进行权衡的广义散列表。布隆过滤器可以被用作分类器,将具有相同输出端口的名字前缀归为一类。因此第二级布隆过滤器数目等于路由器的端口数量。假设一个路由器含有 N 个端口,并且每一个布隆过滤器相应地需要 k 个散列函数来计算最终的转发目的地。对于每一次查询,需要 $k \times N$ 次访问,这急剧降低了查找速度。因此,需要一个新型的布隆过滤器来减少内存的访问量并能支持高速查找。

4.1.2 合并的布隆过滤器

合并的布隆过滤器可以将 NameFilter 第二级的内存访问次数从 $k \times N$ 次降低到 k 次,从而有效地加速 NameFilter 的查找速度。按端口划分的名字前缀集合具有的名字前缀数目相近,可以用相同大小的布隆过滤器来构建这些集合。为使所有的布隆过滤器具有相同的行为,除具有相同的长度外,还需要有相同的散列函数。如图 9 所示,左侧的 N 个布隆过滤器具有相同的长度和散列函数, N 个布隆过滤器的查找过程中查询的地址相对自身布隆过滤器的起始地址的偏移量是相等的。因此, N 个布隆过滤器可以合并成一个布隆过滤器,其中原先一个地址存储 1 bit 转变为一个地址存储 N bit,每个比特对应原先布隆过滤器中的 1 bit,即将同一个位置的 N bit 整合成一个比特串,一个比特串被存放在一个或者连续的几个字中。原先左侧的 N 个布隆过滤器被合并成右侧的一个布隆过滤器。每一个布隆过滤器的第一位都是 1,经合并和转置后,在合并的布隆过滤器相应位置的字中存储比特串 $11 \cdots 1$ 。

在这种新构造的数据结构中,当要进行前缀查找来确定一个名字的转发端口时,NameFilter 只需要利用相关的 k 个散列函数进行 k 次运算,得到相应的 k 位比特串,再通过“与”操作就能获取查询结果。结果中“1”的位置代表最终的转发端口。如图 9 所示,比特串为 $11 \cdots 1$ 、 $11 \cdots 1$ 、 $10 \cdots 0$ 和 $11 \cdots 1$,进行“与”操作的结果是 $10 \cdots 0$,这代表转发端口的编号为 1。

合并的布隆过滤器将串行的操作转化为并行操作,减少了计算时间。一方面,合并的布隆过滤器仅需要进行 k 次

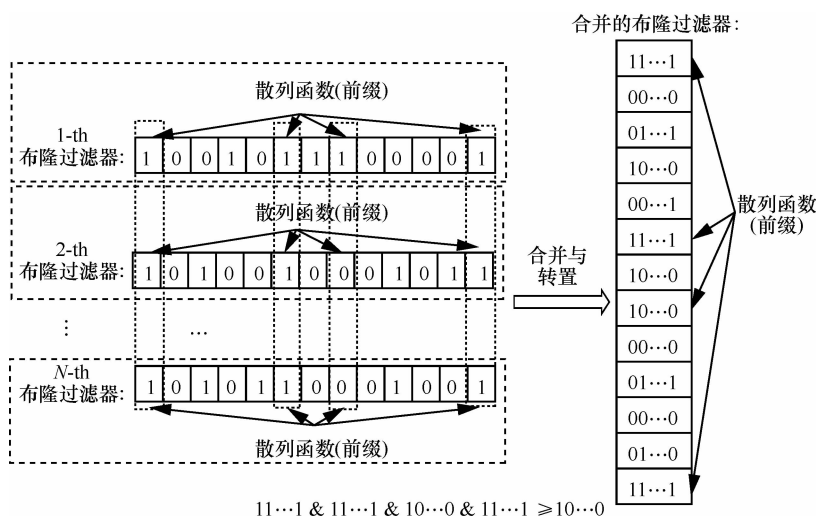


图 9 NameFilter 的第二级采用的合并的布隆过滤器



字符串散列计算,而原先的 N 个布隆过滤器则需要进行 $k \times N$ 次散列计算,计算时间复杂度是原来的 $1/N$; 另一方面,合并的布隆过滤器仅需要 k 次内存访问,而原先的 N 个布隆过滤器则需要进行 $k \times N$ 次内存访问,访存次数同样降低为原来的 $1/N$ 。因此,相比于直接使用多个布隆过滤器,NameFilter 应用合并的布隆过滤器可以在第二级实现 N 倍加速。

4.1.3 面向字符串的一次访存布隆过滤器

合并的布隆过滤器并不适合于 NameFilter 的第一级布隆过滤器。第一级布隆过滤器用于确定一个名字的前缀长度,而每一个布隆过滤器存储的名字前缀的数目存在巨大差异。这是由于每一个布隆过滤器存储的名字前缀具有相同的词元个数,而名字路由表中名字前缀的词元个数分布并不均匀。例如在实验用的名字路由表中,具有 2 个词元的名字前缀数目是具有 6 个词元的名字前缀数目的 987 倍。

因此,需要通过提高单个布隆过滤器的查找速度来提高 NameFilter 的第一级查找速度。一次内存访问布隆过滤器首先由 Yan 等人^[13]提出,其主要思想是用一个散列函数的计算结果作为地址,将剩余的 $k-1$ 个散列函数的计算结果映射到一个字中,这样就能通过一次内存访问得到 k 个散列值,将访问内存的总次数从 k 次减小到 1 次。

在传统的基于 IP 的网络数据分组处理中,散列对象是 IP 地址。散列函数的计算过程占用比较小的计算资源,这样就使得访存次数成为查找的瓶颈。在内容中心网络的名字查找中,一次散列计算至少需要遍历一次名字(字符串),要计算 k 个散列值就要遍历 k 次字符串,这就使得在 NameFilter 的布隆过滤器的查找过程中,散列值的计算成为最耗时的部分之一。因此,加速字符串的散列值计算过程,可以提高布隆过滤器的查找速度。NameFilter 使用算法 1 来加快面向字符串的一次访存布隆过滤器的散列值计算速度。首先,基于 DJB 函数^[14](基数为 33)计算出第一个散列值,其余的 $k-1$ 个散列值,可以基于先前的散列值得到,即第 $i(1 \leq i \leq k-1)$ 个散列值通过第 $i-1$ 个散列值加上新的偏移量计算得到。经过 $k-1$ 次循环,这 $k-1$ 个散列值被映射到一个字中。第一个散列值(Hash[0])表示在布隆过滤器中的地址,存储有剩余的 $k-1$ 个散列值的字 $WORD_1$ 和该地址中存放的字 $WORD_2$ 进行“与”操作,如果字 $WORD_1$ 的值不变,则说明布隆过滤器中存储了该名字前缀;否则,说明没有找到匹配项。

假设 w_s 代表名字前缀的字符数,则一次 DJB 散列计算的时间复杂度为 $O(w_s)$ 。而原来需要计算 k 次散列值,总的时间复杂度是 $O(k \times w_s)$ 。算法 1 则将时间复杂度降低为 $O(w_s + k - 1)$ 。与此同时,访问内存的次数也由 k 次减小到 1 次,访存时间是原来的 $1/k$ 。

算法 1 面向字符串的散列计算过程

Procedure SHash(Prefix)

$w_s \leftarrow \text{strlen}(\text{Prefix});$

hash[0...k] $\leftarrow$ 0;

word $\leftarrow$ 0;

hash[0] $\leftarrow$ DJB(Prefix);

For $i \leftarrow 1$ to k do

location $\leftarrow$ hash[i-1] % w_s ;

hash[i] $\leftarrow$ (Prefix[location] + hash[i-1]) % w ;

word $\leftarrow$ word | ($i < \text{hash}[i]$);

End For

End Procedure

4.2 主要性能结果

为体现 NameFilter 的性能优势,除实现 NameFilter 外,还实现了其他 3 种名字查找方法并进行了比较,其他 3 种名字查找方法分别为:传统的基于字符查找树的方法(CharacterTrie),第一级使用布隆过滤器、第二级使用散列表的方法(BloomHash),未经改进的两级布隆过滤器算法(BloomFilter)。默认情况下,NameFilter 和实验中的其他所有方法,误报率都被设置为 10^{-8} ;BloomHash 中第二级散列表的装载因子 α 设置为 0.5。

4 种方法分别在 3M 名字路由表(含有 3×10^6 个名字前缀的路由表)和 10M 名字路由表上进行平均名字负载和重名字负载的查找速度测试。表 1 展示了 4 种方法在普通服务器上仅使用一个物理线程时的名字查找速度。在 10M 名字路由表中,CharacterTrie 在平均负载和重负载下的名字查找速度分别是 0.124 MSPS 和 0.102 MSPS;BloomHash 能够有效加快名字查找速度,最终达到 1.024 MSPS 和 0.834 MSPS,比 CharacterTrie 加速了 8 倍。由于 BloomFilter 没有使用合并的布隆过滤器,需要多次布隆过滤器查找才能获得最终结果,导致 BloomFilter 仅能实现 0.185 MSPS 和 0.178 MSPS 的查找吞吐率。NameFilter 通过在第一级应用面向字符串的一次访存布隆过滤器,在第二级应用合并的布隆过滤器,名字查找速度能达到 1.895 MSPS 和 1.834 MSPS,相当于 CharacterTrie 的 17 倍、BloomHash 的 1.8 倍。

表 1 4 种方法在仅使用一个运行线程时的查找速度

名字路由表	Trace	名字查找吞吐率/MSPS			
		CharacterTrie	BloomHash	BloomFilter	NameFilter
3M	平均负载	0.188	1.043	0.203	2.033
3M	重负载	0.156	0.977	0.185	1.903
10M	平均负载	0.124	1.024	0.185	1.895
10M	重负载	0.102	0.834	0.178	1.834

4 种不同名字查找方法的存储空间开销情况见表 2。CharacterTrie 构建 3M 名字路由表需要 283.21 MB 内存,构建 10M 名字路由表需要 1 026.38 MB 内存。和CharacterTrie 字符相比,BloomHash 能节省 34%的存储开销,只需要 184.35 MB 和 680.49 MB 内存。BloomFilter 具有最好的存储空间压缩效果,存储 3M 名字路由表和 10M 名字路由表仅需要 52.63 MB 和 190.10 MB 内存,压缩了 80%的存储空间。由于 NameFilter 使用的面向字符串的一次访存布隆过滤器和合并的布隆过滤器都需要额外的空间来保证误报率低于 10^{-8} ,所以 NameFilter 存储 3M 路由表和 10M 路由表分别需要 64.73 MB 和 234.27 MB 内存,比 CharacterTrie 减少 77%和 65%的存储空间。

表 2 4 种不同名字查找方法的内存开销测试结果

名字路由表	存储空间/MB			
	CharacterTrie	BloomHash	BloomFilter	NameFilter
3M	283.21	184.35	52.63	64.73
10M	1 026.34	680.49	190.10	234.27

5 结束语

由于内容中心网络路由器中的名字路由表具有规模大、名字不等长、更新频率高等特点,使得内容中心网络中的路由查找成为实际部署前亟待解决的难题。初步的研究工作表明,未来技术存储大规模名字路由表、实现线速名字查找和高速更新是可行的。这些研究工作^[3-6,15-17]为实际制造商用内容中心网络路由器提供了理论基础和技术方案。

参考文献

1 Web server survey. <http://news.netcraft.com/archives/category/>

web-server-survey/
2 Content-addressable memory. http://en.wikipedia.org/wiki/Content-addressable_memory
3 Wang Y, Zu Y, Zhang T, *et al.* Wire speed name lookup: a GPU-based approach. Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI'13), Lombard, IL USA, 2013
4 Wang Y, He K Q, Dai H C, *et al.* Scalable name lookup in NDN using effective name component encoding. Proceedings of the 32nd International Conference on Distributed Computing Systems (ICDCS), Macau, China, 2012
5 Wang Y, Pan T, Mi Z A, *et al.* NameFilter: achieving fast name lookup with low memory cost via applying two-stage bloom filters. Proceedings of INFOCOM 2013, Mini-Conference, Turin, Italy, 2013
6 Wang Y, Xu B Y, Tai D Z, *et al.* Fast name lookup for named data networking. Proceedings of IWQoS 2014, Hong Kong, China, 2014
7 NVIDIA's next generation CUDA computer architecture: fermi. http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf, 2014
8 Zu Y, Yang M, Xu Z, *et al.* GPU-based NFA implementation for high speed memory efficient regular expression matching. Proceedings of ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), New Orleans, Louisiana, 2012
9 Edward F. Trie memory. Communications of the ACM, 1960, 3(9): 490~499
10 Bloom B H. Space/time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970, 13(7):422~426
11 Named data networking (NDN) project. <http://www.named-data.net/techreport/TR001ndn-proj.pdf>, 2014
12 Bonomi F, Mitzenmacher M, Panigrahy R, *et al.* An improved construction for counting bloom filters. Proceedings of the 14th Annual European Symposium, Zurich, Switzerland, 2006



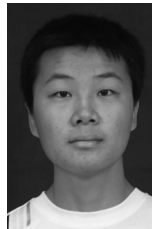
的观看体验。然而在 ICN 中部署 MPEG-DASH 存在的问题还有很多,还有很大的研究价值和应用发展空间,值得进行更深入的研究和应用开发工作。

参考文献

- 1 Ahlgren B, D'Ambrosio M, Marchisio M, *et al.* Design considerations for a network of information. Proceedings of the 2008 ACM CoNEXT Conference, Madrid, Spain, 2008
- 2 Cisco. Cisco Visual Networking Index: Forecast and Methodology, 2011-2016. Tech Rep, May 2012
- 3 Sodagar I. The MPEG-DASH standard for multimedia streaming over the internet. IEEE Computer Society (S1070-986X), 2011, 4(18): 62~67
- 4 Akhshabi S, Narayanaswamy S, Begen A C, *et al.* An experimental evaluation of rate-adaptive video players over HTTP. Signal Processing: Image Communication (S0923-5965), 2012, 4(27): 271~287
- 5 Miller K, Quacchio E, Gennari G, *et al.* Adaptation algorithm for adaptive streaming over HTTP. Proceedings of 19th International Conference on Packet Video Workshop (PV), Munich-Garching, Germany, 2012: 173~178
- 6 Liu Y, Geurts J, Point J C, *et al.* Dynamic adaptive streaming over CCN: a caching and overhead analysis. Proceedings of 2013 IEEE International Conference on Communications (ICC), Budapest, Hungary, 2013: 3629~3633
- 7 Ccnx project v0.6.0. <http://www.ccnx.org>
- 8 Müller C, Timmerer C. A VLC media player plugin enabling dynamic adaptive streaming over HTTP. Proceedings of the 19th ACM International Conference on Multimedia, New York, NY, USA, 2011: 723~726

- 9 Jacobson V, Smetters D, Thornton J, *et al.* Networking named content. Proceedings of the 2009 ACM CoNEXT Conference, Toronto, Canada, 2009

[作者简介]



辛永辉,男,中国科学院声学研究所博士生,主要研究方向为下一代互联网与未来网络、内容分发与缓存优化。



李杨,女,博士,中国科学院声学研究所助理研究员,主要研究方向为下一代互联网与未来网络、内容分发与缓存优化。



林涛,男,博士,中国科学院声学研究所副研究员,主要研究方向为未来网络、分布式内容缓存、可扩展路由。

(收稿日期:2014-07-20)

(上接第 17 页)

- 13 Qiao Y, Li T, Chen S G. One memory access bloom filters and their generalization. Proceedings of the 30th Annual IEEE International Conference on Computer Communications (INFOCOM), Shanghai, China, 2011
- 14 DJB hash function. <http://www.partow.net/programming/hashfunctions/#DJBHashFunction>,2012
- 15 Wang Y, Dai H C, Jiang J C, *et al.* Parallel name lookup for named data networking. Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM), Houston, Texas, United States, 2011
- 16 Wang Y, Tai D Z, Zhang T, *et al.* Greedy name lookup for named data networking. Proceedings of ACM SIGMETRICS, Pittsburgh, PA, 2013
- 17 Wang Y, Dai H C, Zhang T, *et al.* GPU-accelerated name lookup with component encoding. Elsevier Computer Networks, 2013, 57(16): 3165~3177

[作者简介]



刘斌,男,博士,清华大学计算机科学与技术系教授,2006 年国家杰出青年获得者,主要研究方向为高性能路由体系结构、未来互联网体系结构、软件定义网络和绿色互联网。



汪漪,男,博士,清华大学计算机科学与技术系助理研究员,主要研究方向为未来互联网体系结构、软件定义网络。

(收稿日期:2014-08-12)