基于节点动态内容流行度的缓存管理策略

张 果[1],汪斌强[1],张 震[1],梁超毅[2]
(1. 国家数字交换系统工程技术研究中心,河南郑州 450002;2. 解放军信息工程大学,河南郑州 450001)

摘 要: 针对命名数据网络中节点无法感知内容流行度变化的缺陷,提出了基于缓存内容流行度动态变化的内容管理策略. 将缓存分为主缓存(Primary Cache,PC)和副缓存(Secondary Cache,SC),分别用于识别和保护流行内容;采用标准布鲁姆过滤器(Standard Bloom Filter,SBF)过滤流行内容请求;引入滑动时间窗口算法和 HASH 表对副缓存内容进行细粒度的统计分析,进而管理缓存内容. 仿真显示,与现有算法相比,该策略以增加少量复杂度为代价,延长高流行度内容的缓存驻留时间,提高了缓存命中率,减轻了服务器负载,并具有可扩展性,具备单线路 40Gbit/s 的报文处理能力.

关键词: 命名数据网络; 动态内容流行度; 线速; 内容管理
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2016)11-2704-09
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.11.020

A Strategy Based on Dynamical Content Popularity for Cache Management

ZHANG Guo[1], WANG Bin-qiang[1], ZHANG Zhen[1], LIANG Chao-yi[2]
(1. *National Digital Switching System Engineering & Technological R&D Center, Zhengzhou, Henan* 450002, *China*;
2. *PLA Information Engineering University, Zhengzhou, Henan* 450001, *China*)

Abstract: To overcome the drawback that nodes in Named Data Networking are insensitive to the change of the content popularity, a dynamic content popularity based cache management strategy is proposed. The strategy divides the cache into primary and secondary one. The former is used to identify popular content and the latter is used to protect it. Standard Bloom Filter is adopted by the strategy to filter popular content requests. The strategy also introduces sliding window and hash table to analyze the content of secondary cache in fine granularity and manage the cache content. Simulation results show that, compared with traditional strategies, our algorithm prolongs the cache residence time of high popularity content, increases cache hit ratio and reduces server loads. Our algorithm is also scalable and has the ability to process packets at 40Gbit/s.

Key words: named data networking; dynamical content popularity; line speed; content management

1 引言

信息中心网络(Information-Centric Networking, ICN)[1]以内容为网络通信的主体,关注用户和应用通信需求的具体内容,是一种新的未来网络架构. 命名数据网络(Named Data Networking, NDN)[2]是一个典型的 ICN 体系结构. 内容缓存技术是 NDN 的研究重点之一.

当前,缓存技术的研究主要集中在缓存决策[3,4]制定,即选取合理的缓存位置和缓存时机. 而针对缓存替换算法的研究较少,在节点内多采用基于 LRU(Least Recently Used)[5,6]缓存替换算法. 文献[7]指出,LRU 算法复杂度低,可满足节点线速处理需求. 文献[8]设计了 LFU(Least Frequently Used)的实现方案,分析了复杂度,并提出了改进方案. 文献[9]讨论分析了 LRU 和 MRU(Most Recently Used)在 NDN 中的具体应用和性能.

内容流行度是缓存内容替换时的重要参考. 文献[10]采用线型拓扑结构,通过泊松过程建模,提出基于流行度的缓存替换策略,通过减少高流行度内容存储,增加内容的多样性来提高命中率,该算法是在特定网

收稿日期:2015-04-16;修回日期:2016-01-07;责任编辑:马兰英
基金项目:国家自然科学基金创新研究群体项目(No. 61521003);国家 973 重点基础研究发展计划(No. 2012CB315901, No. 2013CB329104);国家自然科学基金(No. 61372121, No. 61309019, No. 61309020, No. 61572519);国家 863 高技术研究发展计划(No. 2015AA016102, No. 2013AA013505)

络环境下的研究,适用范围有限. 文献[11]针对 LRU 和 LFU 的不足,提出了 RUF(Recently Used Frequency)算法. RUF 考虑了流行度的动态特性,逐包统计信息同时将其存于 hash 表,但是没有给出存储解决方案,实际网络环境中,该方法会导致 hash 表急剧膨胀而无法有效应用. 文献[12]基于静态数据集研究了 youtube 的数据流行度曲线,采用拟合函数对该网站视频内容流行度进行分析估计,该方法仅能用于仿真实验,不能在实际中开展应用.

文献[13]指出,在缓存系统中,线速处理是对内容索引表项操作的要求,内容可以存储在低速缓存中. 因此,基于当前硬件存贮介质处理速度[14],可考虑设计满足线速处理的节点缓存管理策略提高缓存命中率,从而改善缓存性能.

本文从降低流行内容的缓存替换频度和延长内容在节点驻留时间的角度,设计基于节点动态内容流行度的缓存管理(Dynamical Content Popularity for Cache Management,DCPCM)策略. 首先将缓存划分为主缓存(Primary Cache,PC)和副缓存(Secondary Cache,SC),分别用于识别和保护高流行度内容;采用滑动窗口算法,设计基于标准布鲁姆过滤器(Standard Bloom Filter,SBF)和 HASH 表的线速流行度监测机制. 在不影响命中率的前提下,对算法进行改进,并做了理论分析探讨.

2　动态内容流行度监测

2.1　节点缓存内容动态流行度监测架构

节点缓存内容动态流行度监测架构分为动态监测和内容缓存两个部分,如图 1 所示. 动态监测部分由 SBF 和 HASH 表组成;内容缓存划分为 PC 和 SC,PC 用于缓存和识别流行内容,SC 用作存储已识别的流行内容. PC 和 SC 存储内容索引管理均采用双向链表实现,PC 链表内采用 LRU 算法,SC 链表使用常用的双向链表管理方法. 为方便后端运算,兴趣包(Interest Packet)或数据包(Data Packet)到达时,先进行 hash 计算,生成内容索引,需要指出的是,同一内容的兴趣包和数据包生成的内容索引相同. 我们分为三个部分来描述整个架构:内容流行度动态监测、PC 管理策略和 SC 管理策略.

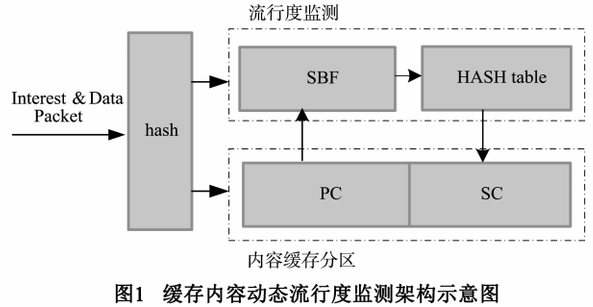

图1　缓存内容动态流行度监测架构示意图

内容流行度动态监测:当兴趣包到达时,先进行 hash 计算,生成内容索引,然后查询 SBF,若命中,则表示该内容为流行项. 然后检测 HASH(为与前文区分,此处大写)表是否存在该内容项,若存在直接将对应的计数值加 1;若不存在,创建对应表项,计数值置为 1. 若查询 SBF 未命中,则丢弃该内容索引.

PC 管理策略:为方便描述,按照兴趣包和数据包的处理流程来说明该部分的处理流程. 当兴趣包到达时,分别在 SC 和 PC 中查询内容索引值. 若在 PC 链表中匹配命中,则返回数据,该内容索引对应的计数值加 1,同时检查访问频次是否到达阈值,若达到阈值,在 SC 链表未满的情况下,将该内容索引插入 SBF 和 SC,将 PC 内的内容索引删除;若 SC 链表已满,该内容索引仍留在 PC 链表内,按 LRU 规则移动至 PC 链表头部. 若在 SC 链表中匹配命中,返回数据即可.

若兴趣包生成的内容索引在 PC 链表和 SC 链表中均未命中,兴趣包经过 PIT 表和 FIB 表转发给下一个节点. 当返回数据经过当前节点时,采用 CEE[2](Caching Everything Everywhere)缓存策略,将数据存入 PC,同时将新生成的内容索引存入 PC 链表. 当 PC 链表未满时,按 LRU 算法将内容索引存入 PC 链表头部. 若 PC 链表已满,先执行 LRU 算法将内容索引存入 PC 链表头部,然后删除 PC 链表尾部的内容索引. 在删除 PC 链表尾部内容索引时:若尾部内容索引对应的内容流行度达到阈值且 SC 链表未满,则将内容索引插入 SBF 的同时,将内容索引从 PC 链表移动至 SC 链表;否则直接将尾部内容索引删除.

SC 管理策略:分为内容索引删除和内容索引插入两个部分. 当一个监测窗口结束时,基于 HASH 表统计的内容流行度情况,筛选出内容流行度低于阈值的内容索引集合,将 HASH 表中对应内容索引和 SC 链表内对应的内容索引删除;若所有内容索引对应的流行度都高于阈值,则不执行删除操作. 内容索引插入:PC 链表内的内容索引对应的计数达到阈值且 SC 链表未满时,将其移入 SC 链表,即 SC 链表的内容索引插入. 内容索引插入过程与 PC 管理策略中内容索引移入 SC 的步骤相同,因此在本部分不再重述.

以上策略利用 PC 内的 LRU 算法的识别功能,在 SC 空间允许的情况下,将访问频次高的内容索引插入 SBF,同时将该内容索引移动至 SC 链表. 当兴趣包到达时,能够利用 SBF 实现线速过滤,进而利用 HASH 表统计流行度,通过对监测窗口内缓存内容流行度的实时分析,延长流行内容在缓存内的驻留时间,保护高流行度内容.

2.2　流行度监测周期

定义 1　节点内容流行度是指内容在一个确定时长内被请求的频次. 那么对某一内容 f 流行度表示为:

$$p(f) = \lambda_T \tag{1}$$

其中,T 为监测时长,λ_T 表示时间 T 内内容 f 被访问的次数.

T 结束时,根据统计结果决定是否删除 SC 内容. 这种算法的优势是实现简单,统计结果直观;缺陷在于割裂了内容流行度的连续性,使统计信息不准确. 监测算法要简单易实现,同时兼顾流行度的连续性,本文引入滑动窗口算法完成流行度统计.

2.2.1 滑动窗口算法

如图 2 所示,假定时间窗口 W_{zT} 由 z 个时长为 T 的时隙构成,统计每个时隙内流行度,同时统计整个窗口内容流行度,滑动步长为 T. 定义如下:

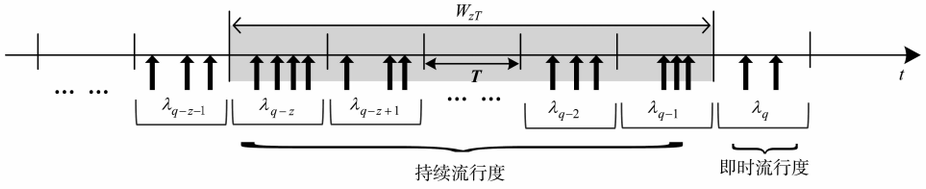

图2 流行度累计窗口示意图

定义 2 即时流行度. 对定义一稍作改动,内容 f 在第 u 个时隙 T_u 内的流行度为 λ_u,称 λ_u 为内容 f 在滑动窗口的即时流行度.

定义 3 持续流行度. 对于 SC 内某一内容 f,在滑动窗口 W_{zT} 内的流行度累计值:

$$p_f = \sum_{i=q-z}^{q-1} \lambda_i \tag{2}$$

称为持续流行度. q 表示时间轴上时隙个数,z 表示滑动时间窗内的时隙个数,时间窗起始时隙 T_{q-z},终止时隙 T_{q-1},λ_{q-z} 是第 $q-z$ 个时隙(slot)T_{q-z} 内的即时流行度.

为便于统计分析,每一个时隙对应一个 hash 表. 滑动窗口内的 z 个 hash 表由一个 HASH 表管理. 一个时隙结束,根据流行度排名来更新 SC 内的缓存内容. 需要指出的是,采用滑动窗口算法后,按照 2.1 节中 SC 管理策略,时隙结束筛选删除内容时,求出内容相对应的持续流行度的 z 个时隙的平均值与流行度阈值作比较,作为选取删除内容的标准.

3 内容替换策略分析

3.1 内容流行度统计策略分析

在 2.1 节算法约定,只要满足移动内容索引条件,就要调整 PC 链表、SC 链表和 SBF. 从算法设计上说,这种方式使 HASH 表能够监测滑动时间窗口内流行内容的流行度变化,为 SC 管理提供依据. 但是会导致使 SBF 统计的内容索引数量多,误判概率增大;PC 和 SC 链表项频繁操作,增加系统消耗,影响处理速度.

缓存内容动态流行度监测的目的是保护监测时隙内流行度有波动的内容,进而提高缓存命中率. 分析发现,按照 LRU 算法思想,高流行度内容因为被访问频繁,能够长期留存在 LRU 链表中,不监测该类内容,不影响缓存命中率. 当 PC 链表内的内容索引更新时,那些因短期内未被访问的流行内容索引被替换,才会降低缓存命中率. 因此从减轻系统处理压力的角度考虑,对算法改进如下:兴趣包到达且在 PC 命中内容时,PC 链表仅执行 LRU 算法将内容索引移动至 PC 链表头部,不再检测其流行度阈值、插入 SBF 和向 SC 转移;其它步骤不变. 后续探讨均以改进算法为基础.

3.2 缓存分配分析

文献[13]指出,内容流行度符合 Zipf 函数分布,即 $p(i) = \frac{\theta}{i^{\alpha}}$,$\theta = \left(\sum_{j=1}^{|F|} 1/j^{\alpha}\right)^{-1}$,其中 $p(i)$ 是第 i 个流行内容 m_i 被访问的概率,α 是内容流行度变化因子,$|F|$ 是内容数量,内容按照访问概率从高到低排序. 假定监测时隙为 T(与前文意义相同,故不作区分),时隙 T 内到达节点的兴趣包总数为 N_{total},流行度阈值为 δ_{th},PC 链表长度为 L. 基于以上约定,按照 3.1 节的改进算法,接下来分析流行度阈值 δ_{th} 和流行内容数目 y(在 Zipf 条件下,y 表示前 y 条内容为流行内容)的关系以及 PC 链表长度 L 和流行度阈值 δ_{th} 的关系.

流行度阈值 δ_{th} 和流行内容数目 y 的关系

在监测时隙 T 内,到达节点的兴趣包总数为 N_{total} 时,请求内容 m_y 的兴趣包数目 $n_{m_y} = N_{total} \cdot p(y)$,令 $\delta_{th} = n_{m_y}$,则有:

$$\begin{aligned}\delta_{th} &= n_{m_y} \\ &= N_{total} \cdot p(y) \\ &= N_{total} \cdot 1/y^{\alpha} \cdot \left(\sum_{j=1}^{|F|} 1/j^{\alpha}\right)^{-1}\end{aligned} \tag{3}$$

当 $|F|$ 和 α 一定时,图 3 描述了 N_{total} 取不同值时,阈值 δ_{th} 和流行内容数目 y 的关系. 随着阈值增大,流行内容的数量随之减少,符合内容流行度分布特性. 在空间限制的情况下,确定合理的阈值可提高缓存系统的效率,公式(3)可为阈值的设定提供理论依据.

PC 链表长度和流行度阈值 δ_{th} 的关系

在一个监测时隙 T 内,到达节点的 N_{total} 个兴趣包中,请求的流行内容有 y 个,内容流行度阈值为 δ_{th},PC 链表是长度为 L 的双向链表,按 LRU 策略执行链表节

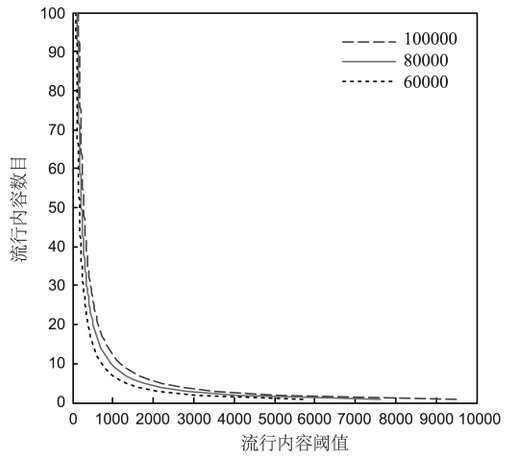

图3　流行内容数量和阈值的关系

点操作. 假定每隔 w 个兴趣包(与新兴趣包相比,该处兴趣包是指其请求内容已在当前节点缓存),引入一个新的兴趣包,新的兴趣包的应答数据到达节点时将淘汰 PC 链表尾部的内容索引. 内容 m_y 在 Zipf 分布中流行度排名为 y,在监测时隙内被访问的次数为刚好为 δ_{th}(由流行度阈值 δ_{th} 和流行内容数目 y 的关系可知,内容 m_y 被请求 δ_{th} 次的概率为 $p(y)$),则在连续 w 个兴趣包中,内容 m_y 不被请求的概率服从超几何分布:

$$\binom{\delta_{th}}{0}\binom{N_{total}-\delta_{th}}{w}\Big/\binom{\delta_{th}}{w} \tag{4}$$

那么内容 m_y 被转移至 SC 的概率 $P_{move}(m_y)$ 为:

$$P_{move}(m_y)=P(m_y \text{ 被请求 } \delta_{th} \text{ 次})\binom{\delta_{th}}{0}\binom{N_{total}-\delta_{th}}{w}\Big/\binom{\delta_{th}}{w}$$

$$=p(y)\cdot\binom{\delta_{th}}{0}\binom{N_{total}-\delta_{th}}{w}\Big/\binom{\delta_{th}}{w} \tag{5}$$

当 N_{total} 远大于 w 时,超几何分布可用二项分布近似替代[15]:

$$\binom{\delta_{th}}{0}\binom{N_{total}-\delta_{th}}{w}\Big/\binom{\delta_{th}}{w}\approx\binom{\delta_{th}}{0}\left(\frac{\delta_{th}}{N_{total}}\right)^0\left(1-\frac{\delta_{th}}{N_{total}}\right)^w$$

$$=\left(1-\frac{\delta_{th}}{N_{total}}\right)^w \tag{6}$$

由式(5)和式(6)可得:

$$P_{move}(m_y)=p(y)\cdot\left(1-\frac{\delta_{th}}{N_{total}}\right)^w \tag{7}$$

持续流行的内容在监测时隙内到达速率高,请求持续时间长. 任意阈值大于 δ_{th} 的内容项,被移动至 SC 内的概率为 $P_{move}(m_y)$,当 $w>L$ 时,内容 m_y 被移动至 SC 的最大概率为:

$$P_{max}=\frac{\delta_{th}}{N_{total}}\left(1-\frac{\delta_{th}}{N_{total}}\right)^L \tag{8}$$

那么,当流行内容数目为 y 时,被移动至 SC 的内容数目最多为:

$$n_{max}=y\cdot P_{max}=y\cdot\frac{\delta_{th}}{N_{total}}\left(1-\frac{\delta_{th}}{N_{total}}\right)^L \tag{9}$$

公式(8)是在阈值和兴趣包总数目一定的情况下,流行内容被移动至 SC 的最大概率,如图 4 所示,链表长度越大,流行内容移动概率越小. 但是在实际网络环境中,PC 链表长度 L 受空间和硬件处理速度的限制,和实际兴趣包数目相比,L 都显得过小,因此通过划分缓存的方法延长内容在缓存的驻留时间是可行的. 公式(9)是参数确定时,流行内容被替换的最大数量. 公式(8)和(9)可作为 PC 和 SC 划分的理论参考.

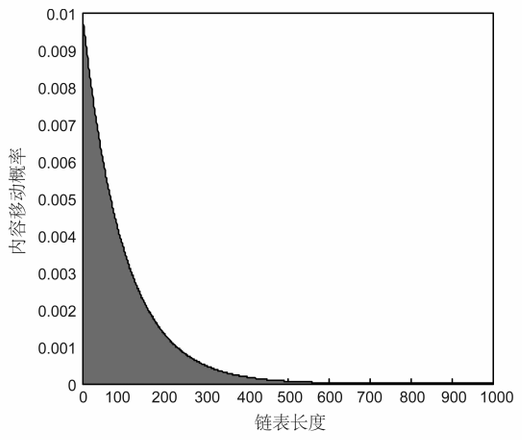

图4　链表长度和流行内容移动概率的关系

3.3　SBF 误差分析

文献[16]给出了 SBF 的误判概率的表达式:

$$f^{SBF}(n,m,k)=(1-e^{-\frac{kn}{m}})^k \tag{10}$$

其中,n 为元素个数,m 为 SBF 向量 $\boldsymbol{V}$ 的长度,k 为 hash 函数的个数.

图 5 给出了 hash 函数个数和 SBF 误差的关系图,误判概率随着 hash 函数个数的增加而增加. 图 6 是误判概率和 m/n 变化关系图,可以看出,当 m/n 增大时,

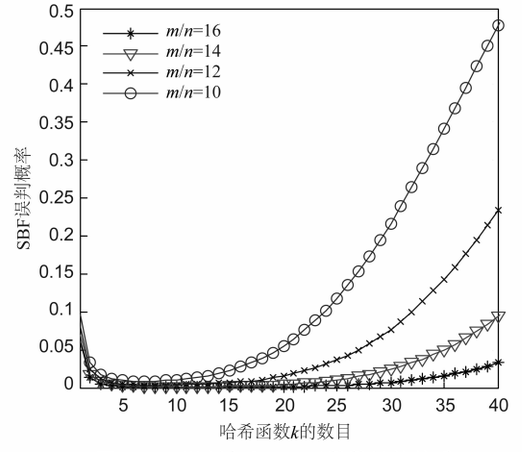

图5　SBF误判率和hash函数数目的关系

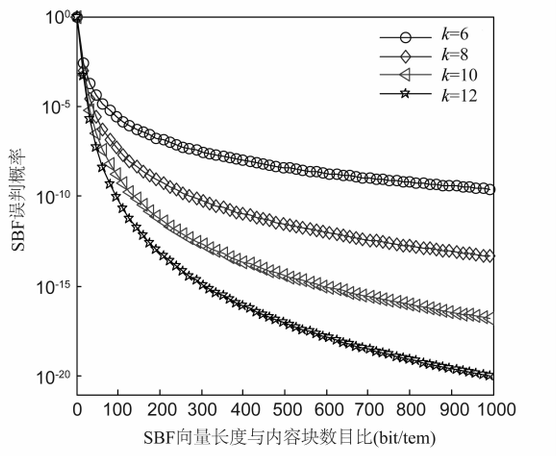

图6 SBF误判率和m/n变化关系

误判概率随之减小.

令 $x(k)=k\cdot\ln(1-e^{-kn/m})$,当$\partial x(k)/\partial k=0$时,$k_0=\ln 2\cdot m/n$,公式(10)有最小值:

$$f^{SBF}(n,m,k)_{\min}=(0.5)^{k_0} \tag{11}$$

公式(11)给出了使误判概率达到最小时,SBF 中 hash 函数数目 k,SBF 长度 m 和存储内容索引数目 n 的关系. SBF 长度和 hash 函数个数确定后,当统计的内容索引数目达到 n 时,再添加内容就会使误判概率增大,此时需要考虑 SBF 的扩展,可通过增加一个新的同样大小的 SBF 来统计新的流行内容索引. 在实际应用当中,根据命名数据网络内容请求分布特征,流行度高的内容仅占内容总量的少数,所以可以预先设定 SBF 大小,能够容纳一定数量的流行内容索引即可.

3.4 HASH 表约束

SBF 存储了滑动窗口内的流行内容索引,而 SBF 不具备删除能力,不再流行的内容索引仍会通过 SBF 进入 HASH 表,导致 HASH 表膨胀.

解决 HASH 表膨胀问题有两个途径:(1)、PC 链表将内容索引插入 SBF 时,也将其插入 HASH 表,若后续该内容流行度降低,则将其索引删除. 当 SBF 过滤出流行度内容时,在插入 HASH 表之前先查找,若 HASH 表内不包含则丢弃,反之则插入 HASH 表,经过双重过滤可防止 HASH 表膨胀. 缺陷是 HASH 表本身过大,在实际应用中不能在高速缓存中实现,而且从 PC 链表插入 HASH 表,增加了操作复杂度. (2)、根据 SC 大小来设置 HASH 表的大小,结合 3.2 节的分析,结合公式(8)可得到被替换出的流行内容所占的比值,然后确定 HASH 表大小. 在一个时隙结束时,HASH 表对内容索引进行整理,清除干扰项,从而防止 HASH 表膨胀. 本文采用第二个途径解决 HASH 表膨胀问题.

3.5 动态流行度处理速度及监测灵敏度

文献[13]指出,NDN 路由器缓存的处理速度主要取决于存储介质的访存速度,文献[14]分析了 NDN 运行的实际需求和当前硬件存储介质处理速度:其中 SRAM 访问速度达到 0.45ns,最大容量 210M;RLDRAM 访问速度 15ns,最大容量 2G;DRAM 读取速度为 55ns,最大容量为 10G. 为满足线速处理需求,将 PC、SC 链表和 SBF 部署在 SRAM 中,HASH 表部署在 RLDRAM 中. 实际硬件设计中,兴趣包到达节点后,PC、SC 链表和 SBF 可以实现并行操作,而当 PC、SC 链表、SBF 和 HASH 表之间存在交互时,处理时延会增加.

几个处理流程分别是:(1)兴趣包到达→PC 命中→移动至 PC 链表头部. (2)兴趣包到达→SBF 命中→HASH 表. (3)兴趣包到达→SC 命中. (4)数据包到达→PC 表头部→PC 表尾巴判断超过阈值→写入 SBF 和移动至 SC(在实际中可并行处理).

由表 1 知,当 SBF 中 hash 函数数目 $k=6$ 时,流程(1)需要 18.6ns;流程(2)需要 15.9ns;流程(3)需要 0.9ns;流程(4)需要 4.5ns. 在 40Gbit/s(OC-768)链路上,假设报文平均长度为 1000bit,链路满载情况下,大约 25ns 到达一个兴趣包. 本文算法能够满足 40Gbit/s(OC-768)的链路处理需求.

表 1 一次操作时间复杂度

操作	复杂度	备注
兴趣包到达 hash 计算/次	$O(k)$	k 为 hash 函数个数
PC 链表操作/次	$O(2)$	LRU 链表操作,复杂度为 $O(\lceil 1+\beta/2\rceil)=O(2)$,装填因子 $\beta=2$
写入 SBF/次	$O(k')$	k'个 hash 函数,执行并行散列运算,复杂度接近 $O(1)$
SC 链表操作/次	$O(1)$	

动态流行度监测的灵敏度与兴趣包的到达速率、内容流行度阈值及存储介质处理速度有关. 由 3.2 节分析可知,在一个监测时隙内,请求任意流行内容 m'的兴趣包连续到达,内容流行度阈值为 δ_{th},LRU 链表操一次访问时间符合流程(1),那么 DCPCM 在节点对内容 m' 的监测反应时间最短为 $18.6\times\delta_{th}$ns. 由公式(8),内容 m'未被监测的概率为 $1-\frac{\delta_{th}}{N_{total}}\left(1-\frac{\delta_{th}}{N_{total}}\right)^{L}$.

4 仿真实验与性能分析

因为 NDN 网络并未在实际网络中部署,仿真实验数据来源有两种:基于现有的 IP 网络数据,通过解析转化,模拟 NDN 通信;基于 ndnSIM[17] 仿真平台,采用 Zipf 函数模拟数据进行仿真. 本文仿真基于 ndnSIM 仿真平台来验证算法的有效性和适用性.

4.1　仿真试验环境和性能评价指标

ndnSIM 实现了 NDN 架构中的基本数据单元结构和路由转发流程,并支持路由、转发和缓存算法的扩展. 在商用服务器(2.70 GHz CPU,RAM 2.0GB)上搭建基于 NS-3 的开源平台 ndnSIM,然后构建实验环境.

网络环境设计　用 GT-ITM 的 Locality 模型生成包含 50 个路由节点的平面网络拓扑. 网络中内容块(chunk)总数为 10000 个,以 1 ~ 10000 依次排序,内容大小设为 10Kbytes. 节点缓存容量一致,CS(Content Store)均设为 10M(根据实验需要可再做调整),可容纳 1000 个内容块,链路带宽 10Mbps. 在网络中部署 2 个内容服务器,负责内容对象的存储和发布,各服务器随机存储 5000 个内容块,并在网络边缘节点随机选取 2 个节点与内容服务器直接相连. 其余节点均作为用户接入节点.

实验设计　模拟用户发送兴趣包,兴趣包到达服从 $\lambda = 100$ 个/s 的泊松过程[13],请求概率服从 Zipf 分布[5],第 i 个内容的请求概率为:$p(i) = \frac{\theta}{i^{\alpha}}$,$\theta = \left(\sum_{j=1}^{|F|} 1/j^{\alpha}\right)^{-1}$. 仿真时间设为 300s,在 Zipf 分布序列中随机选取一定比例的扰动内容,进行随机化重新排序[18],扰动比例为 10%,即 1000 个内容块,按照此策略,内容流行度排名每隔 30 秒变化一次. 节点初始缓存状态为空,无缓存内容. 缓存策略为 CEE[2],缓存替换策略为 LRU、MRU 和 LFU,以及本文提出的 DCPCM 策略,四种替换策略中缓存链表长度 L 均为 1000. 默认 SC 在 DCPCM 缓存中占比为 0.2,滑动窗口默认设置为 8s,时隙为 1s,流行度阈值默认值为 600(个兴趣包). 没有特别指出时,均采用默认参数设置.

性能评价指标　(1)缓存命中率(Cache Hit Ratio,CHR),是网络中节点缓存内容响应兴趣包数量与总的兴趣包的比值;(2)服务器平均负载(Average Server Load,ASL),即单位时间内到达服务器的兴趣包数量.

4.2　性能分析

4.2.1　与现有的替换策略仿真对比

图 7 是在节点缓存空间与内容块数量之比(cache size/catalog size)不同时,四种策略的缓存命中率变化情况. 当二者比值为 10% 时,DCPCM 策略的缓存命中率优于其它三种策略. 随着比值的增大,DCPCM、LRU、MRU 和 LFU 四种策略的缓存命中率之间的差别越来越小,这是因为随着缓存空间的增大,能够留存在 LRU、MRU 和 LFU 缓存中的内容块越来越多,从而提高了缓存命中率,此时 DCPCM 策略延长缓存内容驻留时间的优势变得越来越小. 而在实际网络中,内容索引项的处理速度受硬件高速处理缓存(SRAM)速度和空间

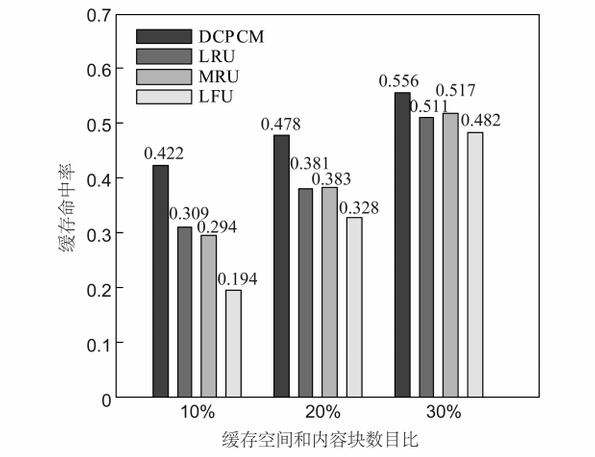

图7　缓存空间与内容块数量比变化时的缓存命中率

的限制,缓存空间也受硬件存储介质的约束. 文献[14]指出可提供的存储空间大小为 10G,文献[13]指出,缓存空间和内容条目的比值一般为 10^{-5},实际应用中,节点要以线速处理大量的兴趣包和数据包,缓存空间与内容块数量的比值更小,在这种情况下,DCPCM 策略的应用优势比较明显.

图 8 是节点缓存空间与内容块数量之比变化时,服务器平均负载变化情况,当二者比值为 10% 时,DCPCM 优于 LRU、MRU 和 LFU. 随着比值的增大,服务器平均负载变化情况趋于相同. 这是因为缓存空间不断增大,三种算法缓存的流行内容几乎相同,使得到达服务器的兴趣包数目趋同,因此三种算法的效果相差不大. 而在实际网络中,在缓存空间受限的条件下,DCPCM 能够有效延长流行内容在缓存内的驻留时间,进而减轻服务器负载.

图 9 是内容流行度突变时缓存命中率的变化情况. 当流行内容突然发生变化时,DCPCM 策略能够快速

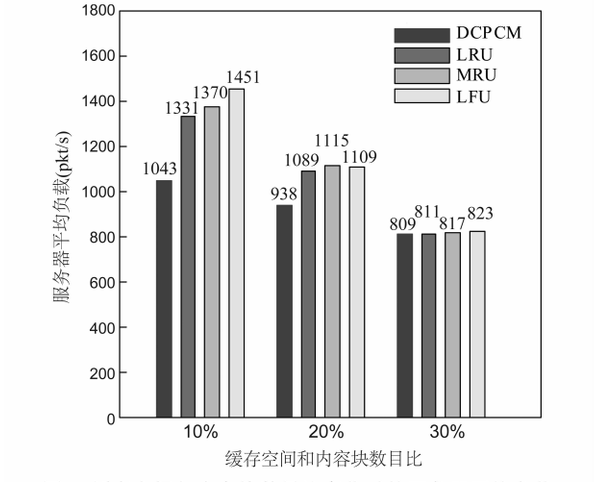

图8　缓存空间与内容块数量比变化时的服务器平均负载

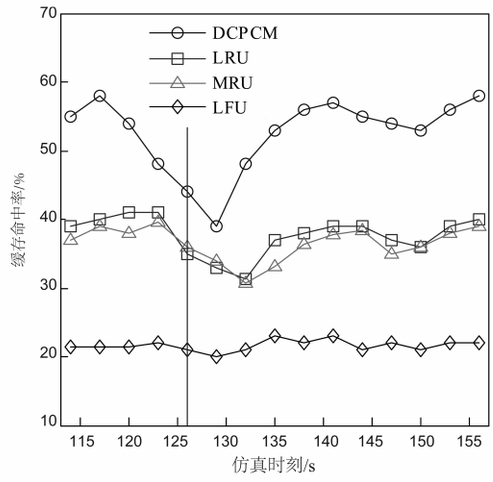

图9 流行度突变时缓存命中率对比

做出反应,使新流行内容替换缓存内容,从而保证缓存命中率. 这是因为在同样大小的缓存情况下,DCPCM 策略由于缓存内分区,按照预设的流行度阈值,能够快速跟踪流行度变化情况,同时将内容移至副缓存保护起来,从而快速提升缓存命中率,在本文实验中反应时间为 3s. 而 LRU 和 MRU 由于缺乏保护机制,对内容流行度的变化反应较为缓慢. LFU 算法缺乏对流行度较高的内容的管理,过时的流行内容无法从缓存内清除,因而对内容流行度的短暂变化几乎不敏感.

4.2.2 代价开销

(1)空间复杂度

空间复杂度用存储所占的比特数来衡量,与 LRU、MRU、LFU 相比,DCPCM 增加了 SBF 和 HASH 表,从而增加了空间消耗.

在仿真实验中,10000 个内容块中流行项为 1000 个,节点缓存可存储 1000 个内容块. 那么 SBF 存储内容索引最多为 1000 个,当 $m/n=20$,$k=6$ 时,误判概率为 3×10^{-4},能够满足统计需求,此时 SBF 消耗的空间为 20000bit. 当 SC 占比为 0.2 时,SC 可存储 200 个内容块. HASH 表项的大小与 SC 大小有关,因此考虑内容在滑动窗口内变化情况,将内容表项大小设置为 SC 的 2 倍,即 400 个内容项. 若一条 hash 值及其对应统计值占 64bit,那么 HASH 表大小为 25600bit,足够存储相应数量的内容块及其流行度. 与 LRU、MRU 和 LFU 相比,DCPCM 增加了 45600bit 约 5.55M 空间消耗.

(2)时间复杂度

3.4 节分析了内容流行度算法操作流程,DCPCM 最长的操作时间为 18.6ns. LRU,MRU 的操作复杂度为 $O(2)$,若在 SRAM 上实现双向链表,完成操作需要 0.9ns. 由文献[8]的改进算法可使 LFU 时间复杂度为 $O(1)$,即 0.45ns.

4.3 适应性讨论

图 10 是 SC 占比不同时,节点缓存空间与内容块数量比变化时缓存命中率的变化情况. 二者比值为 2% 时,SC 占比越大,缓存命中率就越高. 在这种情况下,SC 占比越大,受保护的内容就越多,图 10 中所示当 SC 占比为 0.3 时,缓存命中率明显增大. 过度增加 SC 占比,使得 LRU 链表长度 L 变短,由公式(8)可知,这种情况下使得 PC 链表内替换率增大,增加 SC 的管理消耗. 当缓存和内容总条目比值逐渐增大时,SC 占比变化对平均缓存命中率影响越来越小. 这是因为随着缓存空间的增大,缓存内容数量增加,从而提高了缓存命中率. 在实际网络环境中,因为缓存空间和 SC 管理开销的限制,需要选择合理的 SC 占比才能有效改善缓存系统性能.

表 2 是在 SC 占比、节点缓存空间与内容块数目比和滑动时间窗一定的条件下,流行度阈值变化对缓存命中率和服务器平均负载的影响. 当阈值过大时,进入 SC 的内容就较少,SC 不能被充分利用,导致缓存命中率降低,服务器平均负载增加. 流行度阈值较小时,SC 内容频繁替换,导致缓存命中率降低和服务器平均负载升高.

表 2 流行度阈值变化时的情况(滑动窗口 8s,SC 占比 0.2,缓存空间与内容块数目比 10%)

流行度阈值(item)	200	400	600	800	1000
缓存命中率(%)	42.7	43.5	50.1	47.6	46.5
服务器平均负载(pkt/sec)	389.3	385.2	382.9	391.6	398.6

表 3 是在 SC 占比、节点缓存空间与内容块数目比和流行内容阈值一定的情况下,滑动窗口大小变化对缓存命中率和服务器平均负载的影响. 窗口过大时,流行度累计值过大,使得对流行度的变化变得不够敏感,导致不能及时将“老化”内容从 SC 内剔除,从而影响了

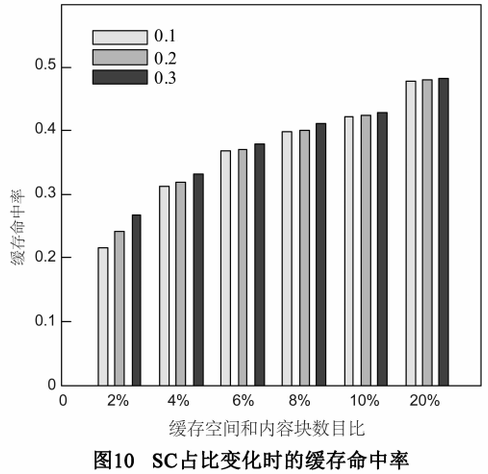

图10 SC占比变化时的缓存命中率

缓存命中率和服务器平均负载. 窗口过小时,流行内容的流行度区分度不高,使得内容频繁替换,导致缓存命中率降低. 当滑动窗为 2s 时,缓存内容替换过于频繁,缓存命中率降低,而频繁的替换使得仿真时间内的缓存内容多样化,因而出现了缓存命中率降低,服务器平均负载降低的情况.

表 3　滑动窗口变化时的情况(阈值 600,SC 占比 0.2,缓存空间与内容块数目比 10%)

滑动窗口(sec)	2	4	8	16	32
缓存命中率(%)	44.2	46.7	50.1	47.3	48.1
服务器平均负载(pkt/sec)	379.7	384.1	382.9	387.4	394.3

5　结论

针对 NDN 节点缓存替换策略无法感知长期流行内容的不足,从线速处理的角度出发,设计了基于动态内容流行度的节点缓存管理策略. 将节点缓存分为 PC 和 SC 两部分,PC 用于识别流行内容,采用 SBF 过滤流行内容和 HASH 表统计内容流行度变化,基于统计信息来管理 SC 内的流行内容,结合实际,对 DCPCM 策略改进探讨,分析了缓存分区的理论依据. 仿真表明,在不影响缓存性能的基础上实现了流行内容的线速、动态监测分析,为缓存内容管理提供了有效的流行度变化信息,延长高流行度内容在缓存节点内的驻留时间,从而提高缓存命中率,提升缓存网络性能. 本文的探讨局限于 CEE 策略下的缓存管理方法,设计的动态估计方法有待进一步优化. 在后续研究中,考虑将缓存决策和缓存替换结合,来改进缓存系统的性能.

参考文献

[1] Ahlgren B, Dannewitz C, Imbrenda C, et al. . A survey of information-centric networking[J]. IEEE Communication Magazine, 2012, 50(7): 26 – 36.

[2] V Jacobson, D K Smetters, J D Thornton, M F Plass, N H Briggs, R L Braynard. Networking named content[A]. Proceedings of the 5th international conference on Emerging networking experiments and technologies[C]. NY, USA: 2009. 1 – 12.

[3] Wang J M, Zhang J, Bensaou B. Intra-AS cooperative caching for content-centric networks[A]. Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking[C]. Hong Kong, China: 2013. 61 – 66.

[4] Saino L, Psaras I, Pavlou G. Hash-routing schemes for information centric networking[A]. Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking [C]. Hong Kong, China: 2013. 27 – 32.

[5] S Podlipnig, L Boszormenyi. A survey of web cache replacement strategies[J]. Acm Computing Surveys, 2003, 35(4): 374 – 398.

[6] Jelenkovi′c P R, Radovanovi′c A. Least-recently-used caching with dependent requests[J]. Theoretical Computer Science, 2002, 326(326): 293 – 327.

[7] G Zhang, Y Li, T Lin. Caching in information centric networking: A survey[J]. Computer Networks, 2013, 57(16): 3128 – 3141.

[8] Ketan Shah, Anirban Mitra, Dhruv Matani. An O(1) algorithm for implementing the LFU cache eviction scheme[R/OL]. http://dhruvbird. com/lfu. pdf. 2010 – 08 – 16.

[9] K Katsaros, G Xylomenos, G C Polyzos. MultiCache: An overlay architecture for information-centric networking[J]. Computer Networks, 2011, 55(4): 936 – 947.

[10] 朱轶,糜正琨,王文鼐. 一种基于内容流行度的内容中心网络缓存概率置换策略[J]. 电子与信息学报, 2013, 35(6): 1305 – 1310.
Zhu Yi, Mi Zheng-Kun, Wang Wen-Nai. A cache probability replacement policy based on content popularity in content centric networks[J]. Journal of Electronics and Information Technology, 2013, 35(6): 1305 – 1310. (in Chinese)

[11] S J Kang, S W Lee, Y B Ko, A recent popularity based dynamic cache management for content centric networking [A]. Proceedings of the International Conference on Ubiquitous & Future Networks[C]. Phuket, Thailand: 2012. 219 – 224.

[12] S Traverso, M Ahmed, M Garetto, P Giaccone, E Leonardi, S Niccolini. Temporal locality in today′s content caching: why it matters and how to model It[J]. Acm Sigcomm Computer Communication Review, 2013, 43(5): 5 – 12.

[13] G Rossini, D Rossi. Caching performance of content centric networks under multi-path routing (and more)[R/OL]. http://perso. telecom-paristech. fr/ ~ drossi/paper/rossi11ccn-techrep1. pdf. 2015 – 04 – 15.

[14] D Perino and M Varvello. A reality check for content centric networking[A]. Proceedings of the ACM SIGCOMM workshop on Information-centric networking[C]. NY, USA: 2011. 44 – 49.

[15] 《现代应用数学手册》编委会. 现代应用数学手册:概率统计与随机过程卷[M]. 北京:清华大学出版社, 1999. 74 – 75.

[16] A Broder, M Mitzenmacher. Network applications of bloom filters: A survey[J]. Internet Mathematics, 2003, 1(4): 485 – 509.

[17] S Mastorakis, A Afanasyev, I Moiseenko, L Zhang. Ndn-SIM 2.0: A new version of the NDN simulator for NS-3

[R/OL]. http://named-data.net/techreports.html. 2015-01-27.

[18] Guo S, Xie H Y, Shi G. Collaborative forwarding and caching in content centric networks[A]. Proceedings of the IFIP Networking[C]. Prague, Czech Republic, 2012. 41 – 55.

作者简介

张 果(通信作者) 男,1985 年 8 月出生,河南南阳人. 国家数字交换系统工程技术研究中心博士研究生,主要研究方向为新型网络体系结构,内容中心网络.
E-mail: guozhang_ndsc@163.com

汪斌强 男,1963 年 2 月出生,安徽安庆人. 国家数字交换系统工程技术研究中心教授、博士生导师,主要研究方向为宽带信息网络,网络安全.
E-mail: wbq6301@163.com

张 震 男,1985 年出生,博士,讲师,研究方向为未来网络体系架构设计,网络测量.

梁超毅 男,1978 年出生,助教,研究方向为未来网络体系架构设计,内容中心网络.