doi:10.11959/j.issn.1000-436x.2017226

面向应用感知的骨干网缓存方法研究

韩春静[1,2,3]，杨晔[1,2]，吕红蕾[3]，葛敬国[3]，李佟[3]，刘韵洁[1]
（1. 中国科学院计算技术研究所，北京 100190；2. 中国科学院大学，北京 100049；
3. 中国科学院信息工程研究所，北京 100093）

摘 要：基于某运营商骨干网上 9 亿条 HTTP 请求，分析了 HTTP 请求组成、内容长度、流行度和时间动态性，提出一种面向应用感知的骨干网缓存方法（AACM, application-aware backbone network cache method）。实验结果表明在缓存空间略有增加的情况下，在线视频的内容命中率提高了 15%以上，回源流量和缓存负载这些指标降低了一半左右。

关键词：骨干网；HTTP；应用层；缓存；Web 对象
中图分类号：TP391 **文献标识码**：A

Research on the backbone network cache method based on application-awareness

HAN Chun-jing[1,2,3], YANG Ye[1,2], LYU Hong-lei[3], GE Jing-guo[3], LI Tong[3], LIU Yun-jie[1]
(1. Institute of Computing Technology, Chinese Academy of Science, Beijing 100190, China;
2.University of Chinese Academy of Sciences, Beijing 100049, China;
3. Institute of Information Engineering, Chinese Academy of Science, Beijing 100093, China)

Abstract: Based on over 900 million HTTP requests from an ISP backbone network, the HTTP characteristic of different content types from a wide range of perspectives was analyzed, including request composition, size, popularity and temporal dynamics, and an application-aware backbone network cache method was proposed, referred to as AACM. The experiment results show that in the case of a slight increase in cache space, the hit ratio of the online-video increases 15%, and the total source traffic and cache I/O load reduced by about half.

Key words: backbone, HTTP, application layer, cache, Web object

1 引言

评价一个 HTTP 网站性能最直观的方式就是看网页打开的速度，已知 Web 对象访问具有时间局部性和空间局部性[1]特征，提高网页反应速度的一个方式就是使用缓存。缓存可以保存刚刚被访问的 Web 对象副本，下一次对此 Web 对象的访问，直接访问缓存的内容副本即可。一个优秀的缓存策略可以缩短网页请求资源的距离，减少延迟，并且由于缓存文件可以重复利用，还可以减少带宽，降低网络负荷。因此，2005 年以来，运营商在靠近用户的网络边缘处部署了大量 HTTP 缓存服务器，用户可以就近访问热门视频、图片等，提高页面上 Web 对象的响应速度；随后出现了 CDN（content delivery network），CDN 本质还是使用 Web 缓存技术[2]，结合内容主动分发技术，实现 HTTP 内容的高效访问。

虽然边缘网络部署了大量的缓存系统并广泛应用 CDN 技术，但从国际商业调查公司统计[3]来看，网站 CDN 使用率呈现明显的长尾效应。Alexa 前 100 名网站中 CDN 的使用率为 44%，而 Alexa 前 10 万名网站中 CDN 的使用率仅为 17.2%，同时，

收稿日期：2017-05-18；**修回日期**：2017-10-25
基金项目：国家重点研发计划基金资助项目（No.2017YFB0801801）；国家科技重大专项基金资助项目（No.2017ZX03001019）
Foundation Items: The National Key Technology Research and Development Program(No.2017YFB0801801), The National Science and Technology Major Project (No.2017ZX03001019)

2009 年以来，骨干网链路上超过 60%的流量为 HTTP 协议流量[4]，此流量比例表明仍然有大量的 HTTP 内容请求在骨干网链路上，运营商希望骨干网上使用缓和 CDN 技术，把其腾出的带宽给其他高价值用户。

以“空间换时间”的缓存方法需要大量空间保存内容副本，已有的研究发现仅访问一次的 HTTP 内容比例非常高[5]，同时缓存空间不可能无限大，向缓存移入移出内容的时机非常重要，所以，缓存内容决策和替换算法是缓存优化的关键。为了解决此问题，一方面，已有的一些缓存优化方法主要关注热门内容的请求频率等特征，试图预测下一次用户最可能访问的内容并提前预取到缓存中，由于预测算法会占据大量计算和存储开销，并不适用于高速骨干网；另一方面，目前部分用户的终端 AP，小型运营商网络都部署了边缘缓存系统或 CDN 系统，一个 HTTP 请求首先经过网络边缘缓存，如果边缘缓存没有命中，则转发请求到骨干网，为了减少边缘网络频繁地直接向源站请求内容，越来越多的运营商考虑骨干网 CDN，当骨干网的接入网节点部署了 CDN 或缓存并大量的为用户提供热点资源的访问服务的前提下，到达骨干网的请求是边缘网络过滤热点内容之后的请求，因此，核心网络与边缘网络的 HTTP 请求模式会有所区别，最近，美国 CDN 提供商 Fastly 已经开始在骨干网部署 CDN 缓存，发现骨干和边缘网络的 HTTP 命中率不同。

随着应用的发展，缓存也需要考虑不同类型内容的用户体验质量（QoE, quality of experience），已有一些基于校园网和边缘网络的 HTTP 请求特征的刻画[6,7]，但是骨干网相关的研究较少。因此，迫切需要分析骨干网的 HTTP 请求特征，并提出合适的缓存方法。

本文基于数据驱动的研究方法，完成了如下工作，首先，采集国内某运营商骨干网 HTTP 请求，刻画了不同类型的内容特性；其次，设计一个面向应用感知的骨干网缓存方法，算法分为 2 个部分，分别是面向应用的请求阈值优化(AACM-RT, application-aware request threshold optimization)和面向应用的缓存空间动态规划(AACM-DP, application-aware dynamic programming of cache space)；最后，使用骨干网 HTTP 请求的现网数据，完成实验并验证算法的有效性。

2 HTTP 缓存优化相关工作

已有研究表明 Web 缓存的核心问题就是设计一种有效的内容缓存策略，提高缓存命中率，减少回源流量和降低缓存 I/O 操作次数[8]。这 3 个缓存性能测量指标的定义如下：缓存命中率为命中的 HTTP 请求数量除以所有 HTTP 的请求数量；回源流量为从内容源网站下载的流量字节；缓存 I/O 操作指内容移入/移出缓存的次数。其中，对于最广泛研究的缓存决策策略与缓存替换策略可以从 HTTP 访问内容的几个典型特征进行优化。

首先，HTTP 访问内容的分布具有时间和空间局部性特征[9]，利用这 2 个局部性原理的相关算法，如 LRU (least recently used)和 LFU (least frequently used) 算法，可以保证最近被访问过的对象副本不容易被移出缓存。

其次，HTTP 访问内容的访问频率遵循 Zipf-like[10]定律，即所有内容中最热门的少数内容占了大部分 HTTP 请求数量，可以利用此特征设计缓存的决策算法，比较常用的方法有内容预取和预置请求阈值。有很多内容预取方法，比较简单的有基于超链接的预取，其次是基于用户访问模式的预取算法，如基于用户访问序列的预测算法[11]、基于 Web 对象流行度的预取算法[12]和基于用户兴趣的预测算法[13]。最新的内容预取研究中，文献[14]提出了接纳策略控制的预取算法，此算法基于减少少数项的访问频率的思想，提高整体的命中率，属于基于内容访问频率特征的优化思路。由于预取技术需要预先保存最近访问的所有 HTTP 请求，存储开销大，目前基本没有被广泛应用。请求阈值方法指设置请求阈值为 N，那么当一个内容被访问 N 次之前，缓存不会下载内容，此方法部署简单被大量采用。但是，请求阈值方法会对热点内容的前 N 次访问造成等待时间长的问题，尤其是对延迟要求敏感的在线视频应用。

另外，HTTP 访问内容的大小服从重尾分布，即 HTTP 访问的内容文件由大量的小文件和少量的超大文件组成。对于缓存存储的内容，一个图片的量级仅为 KB，一个应用更新程序经常超过 1 GB，为了提高缓存性能，通用的优化方法将请求的内容分成 HTTP 小文件和 HTTP 大文件缓存，保证大文件和小文件获得缓存的机会平等；其次，很多内容替换算法增加内容长度的权重，内容文件越大，被

缓存的几率越小，如最小相关价值[15]（LRV, least relative value）方法，LRV 考虑内容的长度、访问次数和访问时间间隔等因素，价值最小的意味着用户最不可能再次访问的内容。

近些年，基于内容流行度的相关研究工作为缓存优化提供了新的思路。Abrahamsson[16]分析了一个美国视频内容商的内容特征，发现内容流行度符合长尾分布，在黄金时段期间（晚上 7:00~9:00）一个地区前 100 个最受欢迎节目的请求量增加，排名前 100 个节目列表中的节目比较稳定，较少移进移出，利用此结果，可以将热门内容引入网内边缘或内容注入 CDN。类似地，文献[17]分析了视频提供商 Hulu 用户选择不同 CDN 的特征，发现用户选择 CDN 后固定下来，即使缓存命中率降低，也不会重新选择 CDN 节点，文献提出可以对视频进行内容分割，选择最优的缓存节点。

从网络规模上考虑，文献[6]分析欧洲校园网络中到 YouTube 网站的请求，发现一次性的视频请求非常多，占了 70%以上。文献[7]分析了接入网 HTTP 请求的可缓存行特征，观察到内容流行度符合长尾分布，流行度曲线比较平稳。由于 CDN 骨干网较少，骨干网缓存的测量更多基于实验环境，例如，文献[18]测量美国 CoralCDN 实验网研究协作缓存解决 Flash crowd 的问题。最近，Shafiq[19]测量了一个大型商业 CDN，测量的数据来自全球几大缓存服务器，也发现 HTTP 小文件和大文件中仅仅访问一次的内容分别占了 50%和 60%的比例，并存在非常热门的访问内容。

近些年，缓存优化研究进一步结合影响缓存的各种内容特征因素。文献[19]结合了请求阈值方法和缓存空间预测算法，此方法中，总缓存空间大小固定，采用自回归综合移动平均（ARIMA）算法计算不同应用类型的缓存比例，取得了比较好的命中率，但是没有考虑不同应用 QoE 的需求，对视频和压缩文件等设置了统一的请求阈值，缓存容量规划仅仅根据请求次数和字节数来判断，对于大的压缩文件，字节占比非常大，根据过去历史预分配缓存空间，会造成下载类型缓存空间越来越大，导致视频请求的访问时延变大。由于移动应用的发展，内容流行度的分析关注访问内容的用户分布特征，文献[20]提出用户访问具有地域性特点，同一个地区的用户访问本地应用较多，用户偏好更接近，不同地区的用户很少有共同的热点，尤其是新闻和体育内容。

在主动测量方面，Ma 等[21]使用 Alexa 的前 100 名的网站（Top 类型）和 Alexa 前 10 万名网站中的随机 100 个网站（随机类型）作为分析对象，通过构造 HTTP 请求主动测量网站的访问延迟。结果发现随机类型比 Top 类型的缓存性能更好，原因是 Top 类型的网站内容更新地更快；其他影响缓存性能的原因还包括相同的内容拥有不同的 URL，内容过期时间设置的太短等，并提出了一些优化的思路，此文献主要测量了 CSS 文件等 HTTP 小文件的访问性能，由于小文件所占带宽不多，所以对骨干网的流量影响不是特别大，本文更关注 HTTP 大文件的缓存优化。

另外，借助客户端资源的缓存优化策略被广泛提出，如 Chaudhari 等[22]提出的代理端 Web 预取方案、Nikolaou 等[23]提出的客户端合作缓存模型，这种方式存在浏览器不兼容的问题，而且存在安全隐患，故不适用于骨干网。

总结上述工作，利用内容局部性原理和热门内容集中访问这些特征，内容预取方法和客户端合作缓存方法能够提高缓存内容的命中率，降低 HTTP 访问延时，面对高速骨干网 HTTP 访问快速到达的特征，这 2 种方法不适用。为了提高缓存处理速度，骨干网缓存大部分仅仅采用了请求阈值策略。近年来，运营商越来越关注缓存对 QoE 的影响。首先，大部分用户采用在线观看视频的方式获取视频内容，在线视频具有很强的用户交互特点，视频直播也在迅速发展，离线视频的下载越来越少，提高在线视频的缓存效果是面临解决的一个问题。在线视频访问对延迟非常敏感，缓存需要高命中率，这和有限的缓存空间形成冲突，文献[5]和文献[19]的测量结果发现, 一次性内容访问的内容比例非常高，对缓存决策策略带来了挑战。虽然随着视频编解码技术的发展，在线视频可以被划分为更小的数据块，视频支持缓存部分内容[24]，但是对视频的访问时延要求进一步提高，期望将访问 Web 对象的延时降到最小，尽量和访问本地的视频保持最小时间差。其次，对于资源下载网站的压缩文件和离线视频，虽然访问时延没有那么严格的要求，但是这些文件不支持部分缓存，如果流行度高的内容命中率很低或内容副本频繁移入移出，会导致回源流量升高和加重缓存 I/O 负载开销。最后，从用

户分布来看，骨干网 HTTP 内容访问的用户分布不具备局部性，造成内容流行度的长尾分布[19]，即使采用缓存空间动态预测算法，不同时期的缓存空间需求不同。

测量骨干网 HTTP 请求特征的困难在于缺乏现网内容访问的数据，其次，HTTP 内容访问数据量庞大，需要专门的日志处理系统进行详细的分析。本文采用了 Hadoop 文件系统和 Spark 实时数据处理技术分析国内某运营商一段时间之内的 HTTP 内容访问日志，形成数据集，基于此数据集的特征分析结果提出相应的优化方法。

3 骨干网 HTTP 访问内容特征分析

3.1 数据集

本节的数据采集拓扑如图 1 所示，国内某 ISP 出口部署了深度分组检测（DPI, deep packet inspection box）设备，并将出口的 HTTP 请求（HTTP request）分组转发到采集服务器上。HTTP 请求分组主要包括请求的 URL、请求时间、文件大小、请求源地址和请求目标地址等信息，如表 1 所示。

由于 HTTP 请求的内容长度不方便在线解析，一般采用文件后缀名区分不同的内容类型，本文也采用了这样的方法。将静态网页、CSS 文件和图片等划分为 HTTP 小文件，视频、音频和压缩文件类似的内容划分为 HTTP 大文件。HTTP 小文件的缓存压力不大，对骨干网流量影响较小，所以，本文重点研究 HTTP 大文件的缓存优化策略。参考了已有的网络协议识别划分方法[4]，依据内容文件的后缀和发布网站，形成音频(audio)、下载(download)、图片(image)、在线视频(online video)、离线视频(video)和混杂应用(other) 6 种类别。数据集为国内某运营商从 2016 年 1 月 1 日~2016 年 1 月 11 日共计 9 亿条左右的 HTTP 大文件请求日志。

表 1 HTTP 请求内容记录

记录	含义
时间	HTTP 请求到达 Cache 的时间
Application	应用类型
Cache ID	Cache 集群的 ID
客户端 IP	HTTP 请求发起者
请求类型	HTTP 请求类型，如 GET
URL	请求的 URL
User-Agent	HTTP User-Agent
内容 ID	内容的散列值
源地址	请求内容源的 IP 地址
源域名	请求内容源的域名
请求开始时间	收到 HTTP 请求的时间
请求完成时间	HTTP 请求内容发送完毕时间
文件大小	HTTP 请求的文件长度

3.2 不同内容类型的 HTTP 请求特征分析

HTTP 内容流行度可以使用内容请求数来定义，请求次数越多则流行度越大。本节从各个方面分析 HTTP 的内容流行度。由于 HTTP 大文件请求中图片、离线视频和混杂应用的请求少，不再重点分析这 3 种类型的请求。

首先，分析 HTTP 请求数据的变化趋势，图 2 为时间序列的请求次数分布，每 5 min 汇聚一次。考虑到 1~3 日是放假时段，仅分析 4 日之后的 HTTP 请求，下同。图 2 的内容请求数呈日周期波动，这个与文献[6]和文献[16]的高峰期仅出现在晚上不

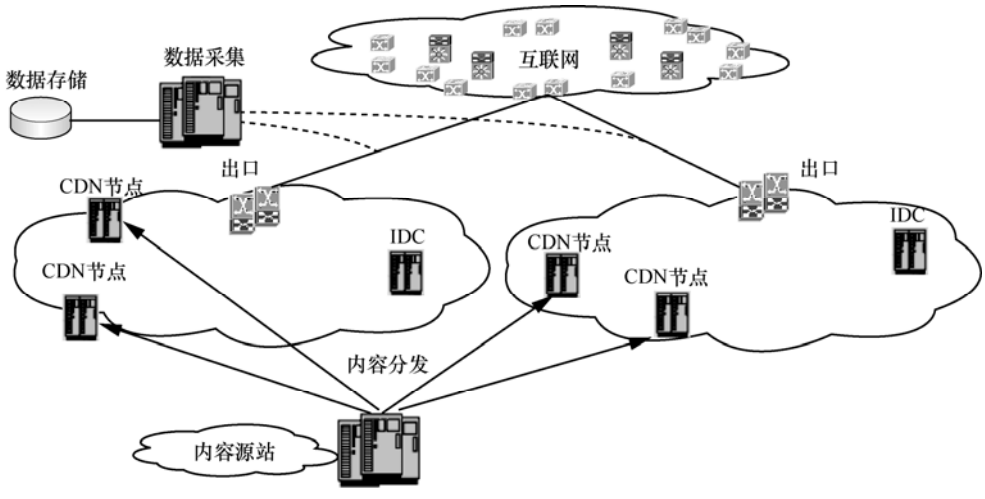


图 1 数据集采集拓扑

同，图 2 分别在早 12 和晚 10 点出现 2 次高峰。其中，晚上高峰期的请求数是白天非高峰期的 2 倍；对内容类型分解发现，高峰期在线视频请求增加的最多，而其他类型请求增加的比较少。

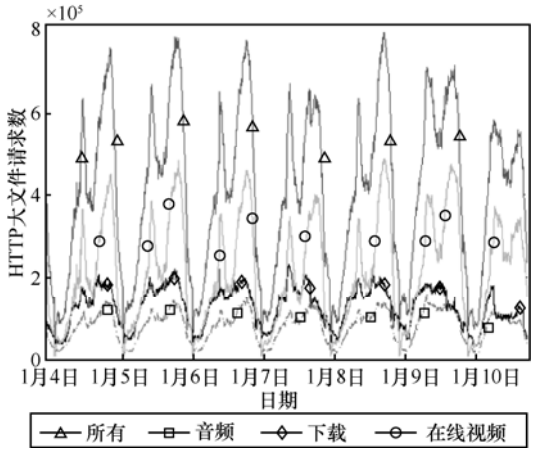


图 2　不同内容类型 HTTP 请求趋势

图 3 为内容请求次数的累积分布函数（CDF, cumulative distribution function），不同内容类型请求呈现长尾分布。所有请求中，63%的内容仅仅被请求过一次，这个与文献[6,19]相似；在线视频请求中，有 44%的内容仅仅被请求了一次，曲线斜率非常大，即或仅被访问 1~2 次，或被访问次数很多；下载类型请求分布虽然斜率比在线视频小，但是访问次数小于 1 次和 2 次的内容比例依然占了 55%和 66%。所以，缓存设计需要尽量降低访问次数少的内容副本被保留，尤其是访问频率稀疏的某些下载文件。

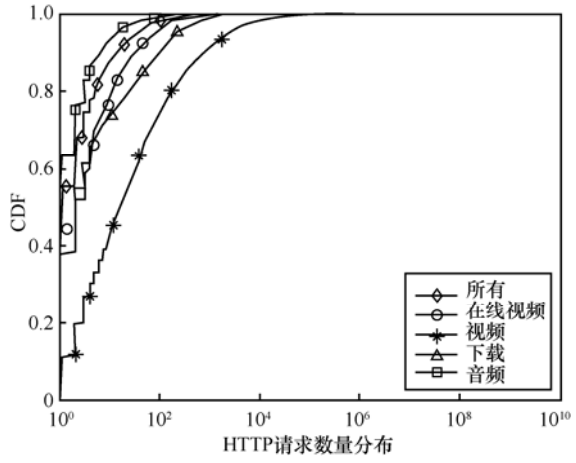


图 3　不同内容类型 HTTP 请求分布

为区分高峰期和非高峰期的 HTTP 请求特征，对比分析了非高峰期和晚上高峰期的请求分布，非高峰期的趋势和图 3 相似，不再显示，高峰期的请求分布如图 4 所示。在线视频访问 1 次或 2 次的内容分别为 26%和 28%，比非高峰期一次性请求降低了 20%，下载亦如此。音频在高峰期的内容访问特征没有发生明显改变，依然比较分散，说明骨干网用户对音频没有明显的热点。

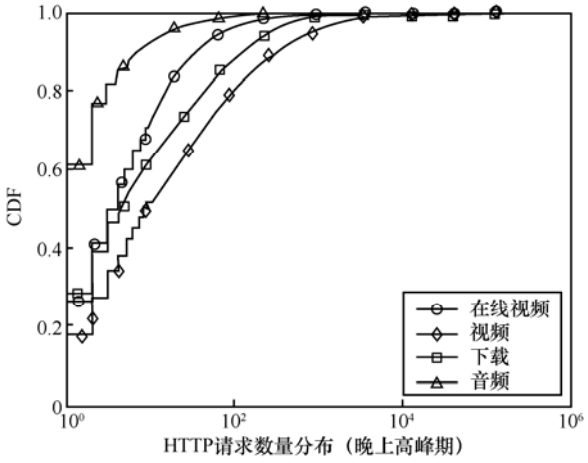


图 4　高峰期不同内容类型 HTTP 请求分布

接下来，使用唯一内容分析 HTTP 访问内容变化趋势。唯一内容是指互联网中可被缓存的、完全相同的文件。对于唯一内容文件，不同网站会产生不同的 URL，通常不使用 URL 区分内容文件，而是使用文件的 MD5 值来定义内容 ID，这个 ID 就是区分唯一内容的标志，图 5 是唯一内容的变化趋势。

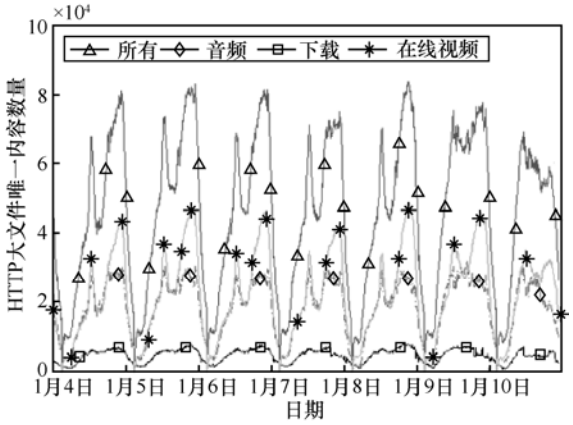


图 5　唯一内容变化趋势

结合 HTTP 请求数量分析唯一内容的变化趋势，唯一内容比例（object uniqueness ratio）可以使用式(1)表示。

$$our = \frac{\text{唯一内容的数量}}{\text{所有的请求数量}} \tag{1}$$

our 表示了所有 HTTP 请求中涉及的内容数量比例，*our* 越小，表明 HTTP 请求的内容越集中。如果 *our*=1，表明每次访问的内容都不同；*our* 越接近于 0，则表明访问越趋向于一个或几个流行的内容。图 6 统计了 *our* 的变化趋势。图 5 和图 6 综合分析发现，在线视频、下载和音频随着请求数的增加，唯一内容数量也正比增加。对于 *our* 趋势，首先，下载的 *our* 曲线非常平稳，基本呈一条平行直线，反映了下载的 HTTP 请求和唯一内容数量的正比例关系，其次，下载的 *our* 绝对值非常小，表 2 统计了 *our* 的平均值，可以看出，下载文件平均一个内容被访问将近 30 次，考虑到图 3 中几乎有一半的下载内容仅被访问了 1~2 次，实际热门下载文件的重复访问次数应该远高于平均访问次数。其次，在线视频的唯一内容数也是随着请求数的变化而变化，但是唯一内容高峰期的曲线没有那么陡峭，在线视频的 *our* 呈天的周期性变化，上午非高峰期的 *our* 呈整体上升趋势并在之间不规则波动，高峰期的 *our* 整体下降趋势并更加平稳，说明非高峰期的用户访问内容更分散，*our* 的上下波动说明内容不断地移入移出，另一个发现是，虽然在线视频的请求数远大于下载类型，但是在线视频的 *our* 绝对值远高于下载类型，说明存在很多热门视频的内容流行度不如下载类型。

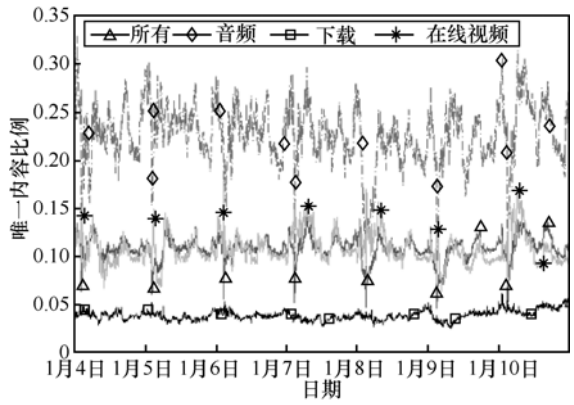


图 6 唯一内容比例趋势

通过上述的测量结果发现，如果采用同样的缓存内容替换算法，如 LFU 或 LRU，平等地对待下载和在线视频请求，在线视频更容易被挪出缓存，导致命中率较低，对于交互性很强的在线视频是不利的；如果下载内容被设置优先移出缓存，会造成热门下载内容的命中率降低，从而增加骨干网带宽的额外开销；对于音频和离线文件下载，*our* 非常大而且存在较大范围的下上波动，离线文件的 *our* 值远大于前几个，不再显示，结果也验证了前边的分析，即音频和离线文件下载的可缓存性不高。

表 2 *our* 平均值

应用	*our*
所有	0.12
音频	0.24
下载	0.036
在线视频	0.11
其他	0.465

图 7 是内容请求数和唯一内容之间的趋势关系，当请求数量小于 10^5 时，下载类型的唯一内容数量线性缓慢增长，在线视频的唯一内容数量指数升高，在线视频的唯一内容数量明显多于下载内容；当请求到达 10^5 之后，下载类型的唯一内容数量呈现平稳的常数，而在线视频继续呈指数型增长，也反映了 2 种不同的内容流行度趋势。

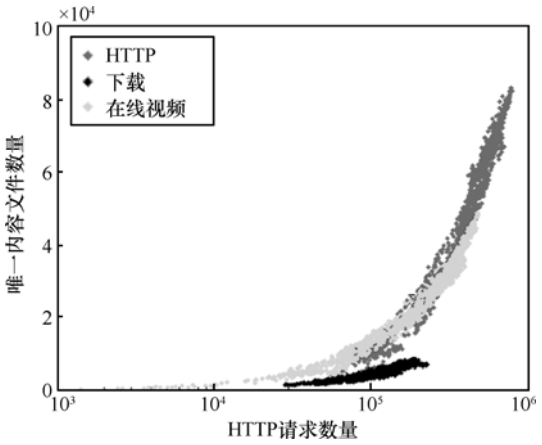


图 7 请求数与唯一内容数量趋势的关系

从 HTTP 内容访问排名分析内容流行度，如图 8 所示。由于工作日结果比较类似，仅显示中间一天的结果。首先，内容流行度呈现 zipf 的长尾分布，高流行度的内容占据了大部分的请求比例，从曲线斜率来看，对于所有内容，高峰期大部分内容流行度比非高峰期高，但是非高峰期的排名前 10 的内容访问次数大部分高于高峰期排名前 10 的内容，分析后发现是一些软件更新的压缩文件和热点在线视频，但是从整体来说，高峰期的内容流行度高于非高峰期。

内容长度会影响到缓存的空间分配，图 9 是 HTTP 内容长度 CDF。相比在线视频和离线视频文

件，下载文件相对分布均匀，依然有 10%的下载文件大于 100 MB，有 1%的下载超过 1 GB，由于基于浏览器的下载应用不支持分段请求，必须缓存完整的内容副本，所以需要考虑下载内容对缓存的空间占用影响。对于在线视频，有 27%的在线视频的文件都为 4 GB，由于在线视频播放可以对内容分段请求，所以即使存在大量 4 GB 的在线视频内容，实际可以部分缓存。离线视频几乎都是大文件，离线视频需要完整下载，所以缓存需要尽量地减少非热门的下载文件和离线视频被缓存。

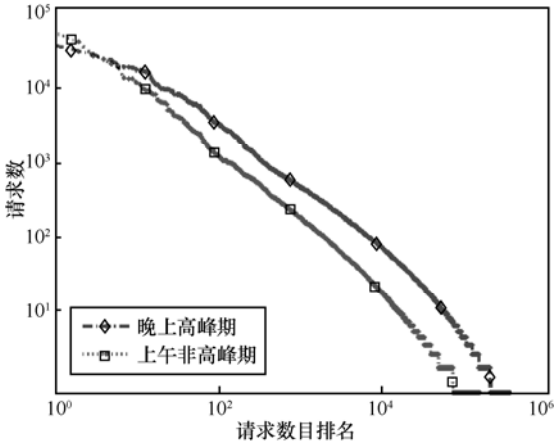


图 8　不同时段的内容流行度

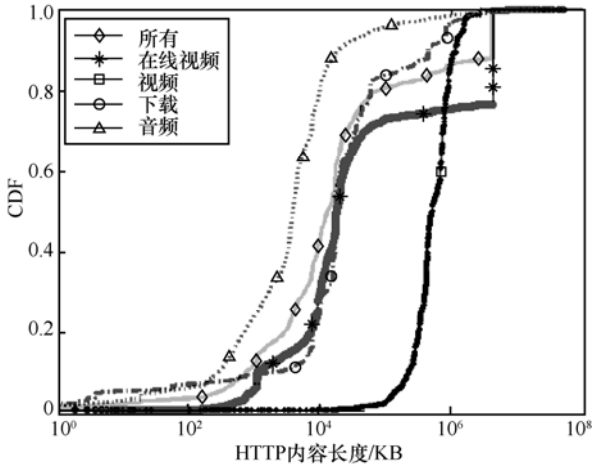


图 9　HTTP 请求内容长度分布

总结 HTTP 请求内容的测量结果。

1) HTTP 请求呈长尾分布，63%以上的请求内容的请求次数在一次，并存在一部分内容具有非常高的流行度。

2) 对于在线视频，非高峰期的用户访问内容比高峰期分散，导致内容不断地移出移入；高峰期在线视频的请求数增加了 2 倍，但是请求一次的内容降低了 20%，

3) 对于文件下载，存在一些热门下载文件被频繁访问，虽然在线视频的请求数远大于下载类型，但是，在线视频的平均内容重复访问次数远低于下载类型，说明热门视频的很多内容流行度不如热门下载文件。

4) 离线视频和音频在高峰期的内容访问特征没有发展明显改变，依然比较分散，说明骨干网用户对这两种类型的内容没有明显的热点。

4　AACM 的缓存优化方法设计

4.1　AACM 缓存方法

根据已有的分析结果，缓存设计原则如下：1) 保证在线视频尽可能高速访问，尤其是高峰期的请求；2) 内容下载的访问延迟可以容忍，但是对于流行度高的热门下载保持高的命中率；3) 减少内容特征解析开销，满足骨干网 HTTP 实时处理的要求。根据上述设计原则，提出一个面向网络应用感知的缓存优化方法 AACM，方法基本思想如下。首先，基于骨干网中访问一次的内容较多特点，不同类型内容设置不同的缓存请求阈值，对交互性要求高的内容设置低阈值，减少缓存空间的浪费，提高热门内容的命中率，称为面向应用的请求阈值优化，简称 RT 策略。其次，根据不同类型内容动态划分缓存空间，采用时间序列预测算法，根据过去一段时刻每种应用的命中率和空间分配容量，预测不同应用下一时刻的缓存空间容量需求，实现缓存空间的动态规划，称为面向应用的缓存空间动态规划，简称 DP 策略，AACM 整体方法流程如图 10 所示。

4.2　AACM-RT 缓存方法

此部分设计一个内容类型数组 *app* 和请求阈值数组 *rt*，*app* 的长度为 *app_len*，例如，*app*={音频，下载，图像，在线视频，视频，其他}，rt_i 指第 *i* 种应用的请求阈值（$1 \leqslant i \leqslant app\_len$），默认 rt_i 为 1，对于某个内容，当请求数到达 rt_i 阈值之前此内容不被缓存，当请求数达到 rt_i+1 时此内容被缓存。对于不流行的内容，AACM-RT 会多产生 rt_i-1 次不命中次数，对于流行度比较高的内容，内容不容易被移出去。

由于骨干网上的唯一内容数量大，请求到达快，统计唯一内容的请求次数会带来额外的开销，需要高效的存储结构，AACM-RT 基于 counting

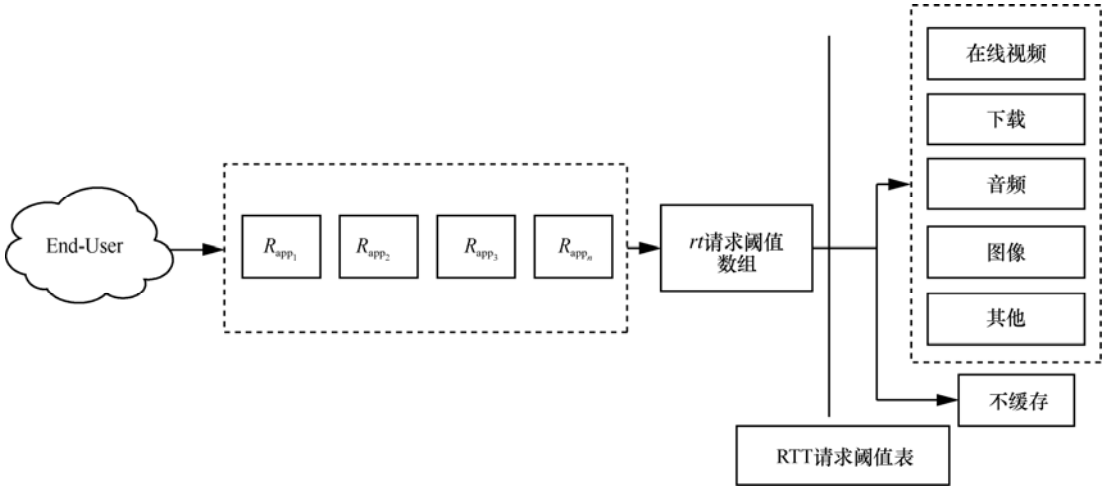


图 10　AACM 方法设计

Bloomfilter[25] 形成请求监测表（RMT, request monitor table），*rmt* 为一个 *N* 个计数器的计数数组，每个计数器是一个 *b* 位的二进制数，设置 *k* 个 Bloomfilter 的散列函数，$f_1, f_2, \cdots, f_k$，输出在 1~*N* 之间分布。对一个输入的请求，计算 *k* 个散列函数，获得 $rmt\left[f_j(e)\right]$（$1 \leqslant j \leqslant k$），获得 *k* 个结果，如果存在 *k* 个结果中有一个结果为 0，则请求不存在 *rmt* 中；如果 *k* 个结果中都不为 0，说明请求有可能在 *rmt* 中。当结果不存在在 *rmt* 中时，$rmt\left[f_j(e)\right]$（$1 \leqslant j \leqslant k$）对应的所有计数器加一操作，*rmt* 中插入 *k* 个结果，当结果达到阈值开始缓存的时候，$rmt\left[f_j(e)\right]$对应的所有计数器减一操作。第 5 节实验中，*N* 设置为 2×10^9，散列函数设置为 10 个，*N* 是高峰期 6 个小时请求的 4 倍，由于请求数 *n* 是不断增长的，所以数据结构需要定期清空，根据流量的特点，设置为 4 个周期，每 6 h 清空一次，清空的原则保持在非高峰期的时候操作。

4.3　AACM-DP 缓存方法

AACM-RT 解决了请求的差异化的性能要求，提高热点内容在缓存的命中率。第 3 节分析发现骨干网的请求在中午和晚上会出现明显的高峰期，在线视频请求占据了大部分高峰期额外的请求，图 11 是在缓存空间固定的情况下设置 $rt_3=1$，其他的 rt_i 都为 4 的情况下的实验结果，实验环境在第 5 节进行说明。实验结果发现，如果缓存空间固定的情况下，无论是 LRU 还是 LFU 的内容替换算法，在线视频应用在中午 11 点请求高峰期开始后命中率开始下降，其他类型的请求命中率下降速度更快。因此，仅依靠 AACM-RT 优化不能解决高峰期请求波动问题，高峰期造成请求命中率持续降低，最后发展到整个缓存不可用的状态。

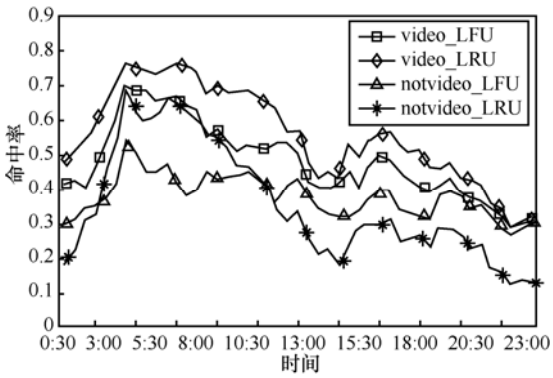


图 11　缓存空间固定的命中率比较

为了解决上述的问题，AACM-DP 提出根据应用类型划分不同的缓存空间，同时根据命中率和不同应用已有的缓存空间占比，来预测下一时刻的缓存占比，动态调整缓存空间。已有研究表明 HTTP 请求具有自相似性[26]，为了验证方法合理性，本文统计了基于时间序列的各个特征的自相关性[27]，对于一个各态遍历的广义平稳随机信号 $x(n)$来说，其自相关函数 $r_x(m)$ 定义为

$$r_x(m)=\lim_{N\to\infty}\frac{1}{N}\sum_{n=0}^{N-1}x(n)x(n+m) \tag{2}$$

结合已有的数据，仅计算时间序列的正相关，以每5 min的回源流量为输入获得图12的时间序列自相关结果：0~55 min 以内相关性大于 90%；0~1.5 h 之内相关性大于 80%；随后相关性先下降后增高；最

后当延时的样本点 *delay* 为 288 次，即和比较指标相隔 24 h 的时候相关度为 76%。从上述分析可以看出，最近一段时间的内容特征自相关性最强，而且使用最近一段时间的值计算存储空间最少，所以，AACM-DP 的方法是可行的。

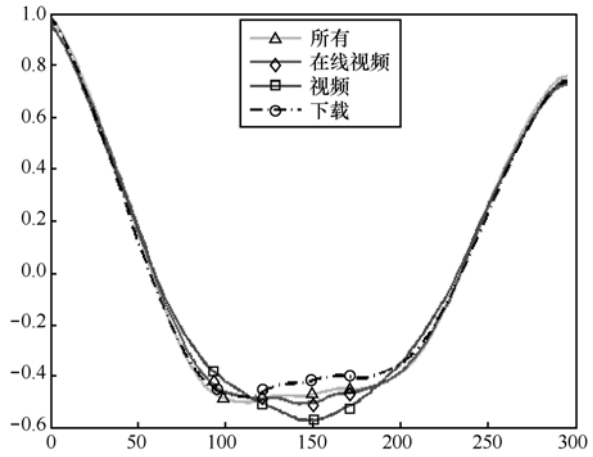


图 12　自相关计算（*delay*=288 次）

DP 方法中，已知 $t-s$ 到 t 个时刻的缓存容量和命中率，基于时间序列预测的方法，预测 t+1 时刻的缓存空间需求，可以用式(3)描述。

$$c_{i,t+1} = f\left(c_{i,t-s},\mathrm{L}\ c_{i,t}, hr_{i,t-s},\mathrm{L}\ , hr_{i,t}\right) \tag{3}$$

其中，$c_{i,t}$ 指时刻 t 第 i 种类型的缓存空间容量，使用字节表示，$hr_{i,t}$ 指 t 时刻第 i 种类型的命中率，$th_{i,u}$ 和 $th_{i,l}$ 分别是第 i 种类型的命中率上限和下限，$k_{i,t}$ 表示空间调整参数，$k_{i,t}$ 的初始值 $k_{i,0}$，n 表示在当前状态持续的周期数。为了应对流量高峰期的请求，当命中率持续降低时，$k_{i,t}$ 指数级增加；当命中率平稳时，$k_{i,t}$ 不变；当命中率呈增长趋势时，$k_{i,t}$ 线性减少，保证内容移出不会太频繁。算法流程如算法 1 所示，默认选择使用最近3个周期缓存空间作为预测回归的周期次数，即 s=2。其伪代码描述如下。

算法 1　AACM-DP 缓存空间动态规划算法

输入　$hr_{i,t}$，$hr_{i,t-1}$，$hr_{i,t-2}$，上一周期改变的比例 $k_{i,t-1}$

输出　第 t+1 周期 Cache 大小 $c_{i,t+1}$

begin

if $hr_{i,t} < th_{i,l}$

　　$k_{i,t} = k_{i,0}2^n$;//低于阈值按照停留在该状态的周期数成指数增大

else if　$hr_{i,t} > th_{i,u}$

　　$k_{i,t} = -k_{i,0}2^n$;//超过阈值按照停留在该状态的周期数成指数减小

else if $hr_{i,t} > hr_{i,t-1}$ //过去 2 个周期，命中率呈上升趋势

　　if $hr_{i,t-1} > hr_{i,t-2}$　//阈值范围内，过去 3 个周期命中率连续升高

　　if $hr_{i,t} - hr_{i,t-1} > hr_{i,t-1} - hr_{i,t-2}$ //增大的趋势

　　$k_{i,t} = -k_{i,0}2^n$;　// cache 按指数减小

　　else　//趋势变平缓

　　$k_{i,t} = -k_{i,t-1}$;　// cache 按上一周期的比例减小

　　else　//过去 3 周期内命中率先降后升

　　$k_{i,t} = -k_{i,0}$;　//cache 减小 k_0 倍

else　//过去 2 个周期，命中率呈下降趋势

　　if $hr_{i,t-1} < hr_{i,t-2}$　//阈值范围内，过去 3 周期命中率连续降低

　　if $hr_{i,t-1} - hr_{i,t} > hr_{i,t-2} - hr_{i,t-1}$ //且趋势增大

　　$k_{i,t} = k_{i,0}2^n$; // cache 按指数增加

　　else//趋势变平缓

　　$k_{i,t} = -k_{i,t-1}$;// cache 按上 1 周期的比例减小

　　else　//过去 3 周期内命中率先升后降

　　$k_{i,t} = k_{i,0}$;　//cache 增大 k_0 倍

$c_{i,t+1} = c_{i,t}\left(1 + k_{i,t}\right)$ //计算下一时刻的空间分配

end

算法 1 主要为了获得空间容量的参数 $k_{i,t}$ 的值，具体的计算流程如图 13 所示。

4.4　AACM 优化

由于每增加一种内容类型会多一步判定操作，为了减少比较次数，在开发原型系统时优化了 AACM 方法，将在线视频类型简称为“视频”，其余的类型称为“非视频”，对应数组 *app*={video, notvideo}，这样“非视频”还包括离线视频、下载、音频等内容，这样减少了多级判断遍历的次数。相比所有类型逐次比较，这种二元判断在实验时发现减少了 76%的比较计算，提高了骨干网实施的可行性。

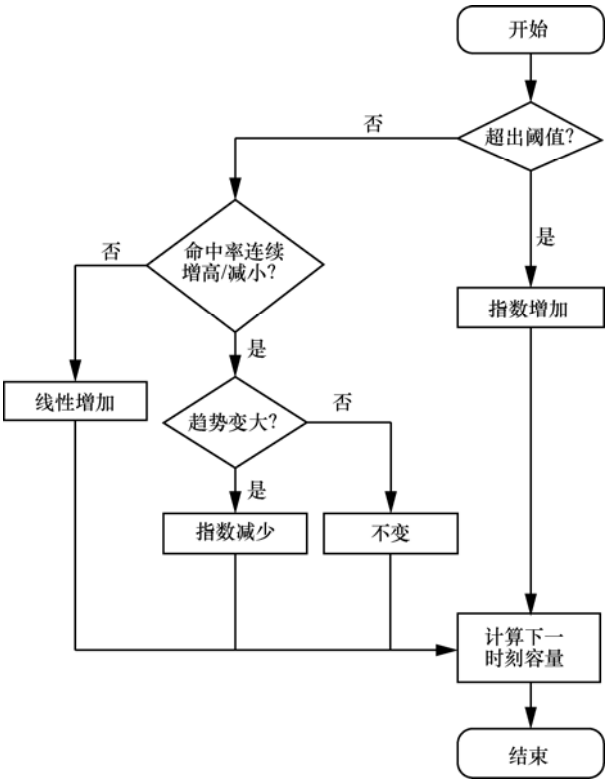


图 13　空间预测调整参数 $k_{i,t}$ 的计算过程

5　实验结果与分析

本文设计一个 AACM 模拟器，模拟器使用 Python 开发。已有的骨干网真实 HTTP 请求数据作为模拟器的输入，由于缓存的空间需求 C 和 HTTP 请求的唯一内容长度 L 成正比增长，它们之间的比例为 θ，则可以表示为

$$C = L\theta, 0 < \theta < 1 \tag{4}$$

使用第 3 节中的真实的运营商 HTTP 内容请求数据实验验证。首先，计算每个小时唯一内容的长度并获得每个小时的平均唯一内容长度 L_{avg}，对缓存日志进行编号，形成 L 的下标。针对 rt 数组，因为在线视频用户对 QoE 的要求比较高，设置 $rt_{\text{video}}=1$，$rt_{\text{notvideo}}=2^v$。

首先，仅仅使用 RT 策略分别对非高峰期和高峰期一个小时的 HTTP 请求进行实验，v 取 1~3，θ 取 0.01、0.02 和 0.045。内容替换策略选取了 LRU、LFU 和随机缓存（random）3 种，使用 hr 表示命中率，如图 14 所示。对于命中率，非高峰期 hr_{LRU}、hr_{random}和hr_{LFU}之间没有明显的排名关系，hr_{random} 更好一些，在高峰期，$hr_{\text{LRU}} > hr_{\text{LFU}} > hr_{\text{random}}$；I/O 操作数和命中率的排名情况一致，回源流量 LRU 明显少于其他 2 种，比较 LFU 和 random 这 2 种的回源流量，LFU 晚上稍微高一点，random 白天稍微多一些。综合判断：同等条件下 LRU 的结果更好，由于空间有限，仅仅显示了 $rt_{\text{notvideo}}=4$ 的实验结果，如图 14 所示。

其次，实验加入 DP 算法，此算法中默认 $k_{i,0}=0.1$，形成完整的 AACM 实验。此部分实验发现了 2 个问题，首先，缓存空间的频繁变动会带来很大的开销；其次，如果在高峰期前后规划缓存空间的时间间隔设置的过大，会得出命中率上下波动的推论，但是不代表真实的最近命中率趋势。结合第 3.2 节的特征分析结果，选择 15 min 为缓存空间动态规划的时间间隔，以一天为结果统计单元，对每天的 HTTP 请求进行处理。对同一天的数据，random 策略运行 3 次并求平均值作为分析的结果，实验中发现 random 的命中率方差非常小，证明了 random 结果的合理性。从 3 个评价指标整体上看，random 和 LRU 都远比 LFU 好，LRU 比 random 的性能更好一点，本文仅挑选了有代表性的 LRU 的实验结果，如表 3 所示。前 2 个是仅使用 AACM-RT 的方法，其中，RT_4 为 $rt_{\text{notvideo}}=4$ 的条件下的结果，RT_{24} 为在请求高峰期设置 $rt_{\text{notvideo}}=2$ 和非高峰期设置 $rt_{\text{notvideo}}=4$ 的结果，第 3 个 DP 为仅仅使用 AACM-DP 方法，最后 2 个为共同使用的结果。

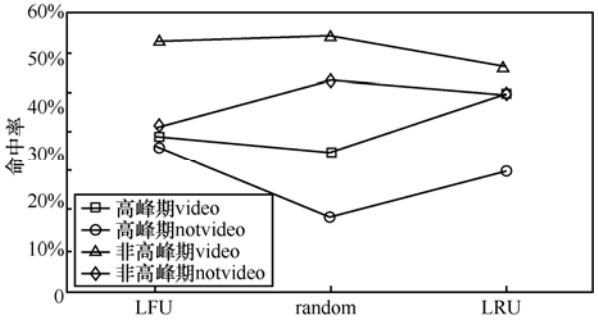


图 14　RT_4 的高峰期和非高峰期命中率比较($\theta=0.02$)

表 3　　缓存实验结果（LRU）

方法	视频命中率	非视频命中率	回源流量/PB	内容替换次数($\times 10^7$)
优化前	35.9%	44.7%	3.700	10.940
RT_{24}	44.2%	33.5%	3.268	7.011
RT_4	47.4%	28.6%	3.100	6.254
DP	54.1%	41.5%	2.150	9.031
RT_{24}&DP	54.1%	34.9%	2.256	5.835
RT_4&DP	54.1%	41.6%	2.166	5.380

实验结果分析如下。首先，对比 RT_{24} 和 RT_4，RT_4 的视频命中率提高了 10%，但造成了非视频命中率的大幅度降低；RT_4 仅有 28%的平均命中率，这个结果会使高峰期的下载应用趋于失效。RT_{24} 虽然提高了非视频的命中率，但是内容替换次数上依然较高，而且 33%的平均命中率依然无法保证高峰期下载业务的开展。

其次，分析 DP 策略，虽然 DP 命中率得到优化，但是内容替换次数很高，造成缓存非常高的 I/O 负载开销，分析原因发现，由于非高峰期一次性内容访问较多和出现大量内容移入移出的操作，导致命中率波动比较大，并占用了较多的缓存空间。

最后，对比 RT_4&DP 和 RT_{24}&DP，这 2 个方法视频平均命中率都比优化前多了 18%，效果非常明显，RT_4&DP 非视频的平均命中率为 41%，仅比优化前的 44%减少了 4%，而且 41%的平均命中率对于下载、音乐等的服务质量可以接受，而 RT_{24}&DP 非视频平均命中率为 34%，这种结果导致在请求高峰期的时候，会产生低于 20%的非视频命中率，影响了用户的体验；分析内容替换次数，RT_{24}&DP 和 RT_4&DP 都减少了一半的 I/O 操作，而且回源流量也获得了 40%的减少，这些减少的带宽可以分配给其他的骨干网应用。

表 4 是缓存动态空间分配策略的实验结果，DP 与 RT_4&DP 对空间的需求相似，比固定缓存空间(45GB)多 11.1%和 12%的空间需求，而 RT_{24}&DP 比固定缓存空间仅多 4.2%，结合命中率的结果，验证了即使单独分配不同应用的缓存空间，AACM-RT 可以减少一次性请求对缓存空间的占用。总结这 2 部分的实验结果，从提高命中率、减少骨干网带宽和降低缓存操作负载 3 个指标观察，RT_4&DP 虽然会产生一部分监测 RTT 表开销，但是总体的收益更好。

表 4　缓存动态空间分配策略的实验结果（LRU）

方法	视频/GB	非视频/GB	总量/GB	与固定缓存比较
DP	34.31	15.67	49.98	增加 11.1%
RT_{24}&DP	34.47	12.44	46.91	增加 4.2%
RT_4&DP	34.31	16.10	50.41	增加 12%

为了验证上述的结论，分析了 RT_4&DP 命中率变化趋势，如图 15 所示，经过优化之后，虽然视频请求的命中率波动范围比较大，但是非视频请求的命中率在 30%~50%稳定波动，整体变化最低也在 31%以上，符合了 QoE 的要求。图 16 是缓存分配空间的堆积，视频和非视频的面积相加等于总的分配空间。非视频内容的缓存空间基本保持恒定，即使在请求高峰期，非视频的空间分配没有明显地增加，这个结论和第 3.2 节的唯一内容比趋势保持一致，说明唯一内容的增长趋势决定了缓存分配空间，与固定缓存开销相比，视频内容的缓存空间和请求数的变化趋势密切相关，额外的空间开销仅服务于高峰期的用户需求，虽然高峰期的请求数是非高峰期的 2 倍，但是高峰期分配的空间容量依然不超过固定空间的 2 倍。

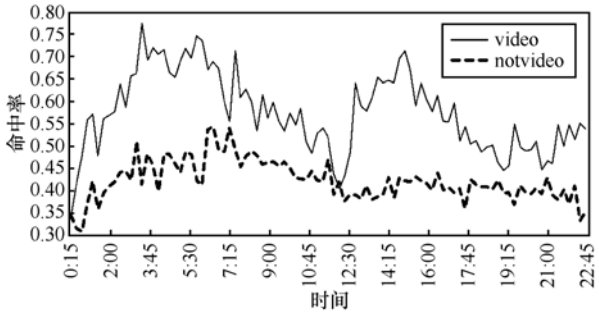


图 15　RT4&DP 的命中率趋势

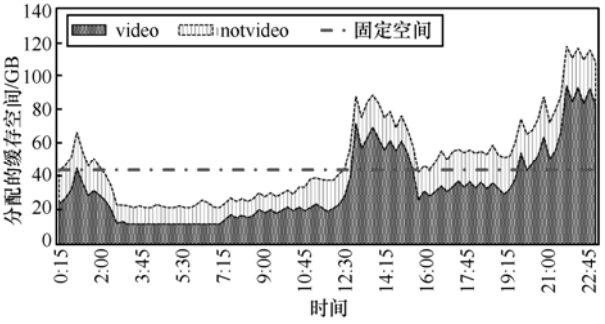


图 16　RT4&DP 的缓存空间趋势

最后，本文分析了缓存的处理性能。一条 HTTP 请求的缓存处理时间定义为，缓存首次收到 HTTP 请求到用户开始下载内容过程中缓存参与的处理时间，请求源站的等待和下载时间都排除在外。图 17 为缓存处理百万 HTTP 请求的平均处理时间，random 最快，基本在 20~50 s，LRU 在 100~200 s，而 LFU 最慢。分析发现，RT 算法由于采用 Bloomfilter 的结构，时间开销很小，其次，空间动态扩展会带来一部分时间开销，而 LRU 和 LFU 的链表操作，占据了最多的时间开销。综合考虑，本文认为骨干网上 AACM 可以结合 LRU 和 random 两者，默认采用 LRU 策略，当请求高峰期出现了明显的处理滞后，可以使用 random 策略，保证 HTTP 请求处理的实时性。

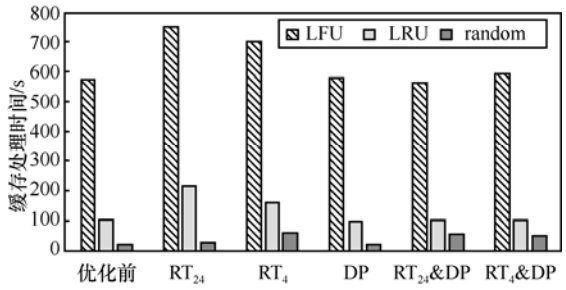


图 17　百万记录的缓存处理时间统计

6　结束语

本文首先分析了骨干网 HTTP 不同内容类型的请求特征，观察到 HTTP 请求呈长尾分布并存在一部分内容具有非常高的流行度。对于在线视频，高峰期它的请求数增加了 2 倍，但一次性内容请求降低了 20%，对于文件下载，存在一些热门下载文件被频繁访问，具有较高流行度。其次，本文提出一种面向应用感知的骨干网缓存方法 AACM，实验结果表明，缓存空间在高峰期略有增加的情况下，缓存性能得到了明显的提升。接下来的工作中，需要进一步研究不同内容类型的内容大小对缓存设计的影响以及 CDN 骨干网缓存优化方法，并开源原型系统。

参考文献：

[1] ALMEIDA V, BESTAVROS A, CROVELLA M, et al. Characterizing reference locality in the WWW[C]// Fourth International Conference on Parallel and Distributed Information Systems. 1996: 92-103.

[2] PALLIS G, VAKALI A. Insight and perspectives for content delivery networks[J]. Communications of the ACM, 2006, 49(1): 101-106.

[3] 中国信息通信研究院. 内容分发网络（CDN）白皮书（2015）[R]. 中国信息通信研究院, 2015.
China Information And Communication Research Institute. Internet content network (CDN) white paper(2015)[R]. China Information and Communication Research Institute, 2015.

[4] RICHTER P, CHATZIS N, SMARAGDAKIS G, et al. Distilling the internet's application mix from packet-sampled traffic[C]// International Conference on Passive and Active Network Measurement. Springer International Publishing. 2015: 179-192.

[5] SHAFIQ M Z, LIU A X, KHAKPOUR A R. Revisiting caching in content delivery networks[J]//ACM SIGMETRICS Performance Evaluation Review. 2014, 42(1): 567-568.

[6] ZINK M, SUH K, GU Y, et al. Characteristics of YouTube network traffic at a campus network-measurements, models, and implications[J]. Computer Networks, 2009, 53(4): 501-514.

[7] IMBRENDA C, MUSCARIELLO L, ROSSI D. Analyzing cacheable traffic in isp access networks for micro CDN applications via content-centric networking[C]//The 1st International Conference on Information-Centric Networking. 2014: 57-66.

[8] PODLIPNIG S, BOSZORMENYI L. A survey of web cache replacement strategies[J]. ACM Computing Surveys (CSUR), 2003, 35(4): 374-398.

[9] JIANG S, DING X, CHEN F, et al. DULO: an effective buffer cache management scheme to exploit both temporal and spatial locality[C]//The 4th Conference on USENIX Conference on File and Storage Technologies. 2005, 4: 8.

[10] BRESLAU L, CAO P, FAN L, et al. Web caching and Zipf-like distributions: evidence and implications[C]//Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. 1999: 126-134.

[11] YANG Q, ZHANG H H, LI T. Mining Web logs for prediction models in WWW caching and prefetching[C]//The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2001: 473-478.

[12] NKATARAMANI A, YALAGANDULA P, KOKKU R, et al. The potential costs and benefits of long-term prefetching for content distribution[J]. Computer Communications, 2002, 25(4): 367-375.

[13] SRIVASTAVA J, COOLEY R, DESHPANDE M, et al. Web usage mining: discovery and applications of usage patterns from Web data[J]. ACM Sigkdd Explorations Newsletter, 2000, 1(2): 12-23.

[14] MA H, LIU W, WEI B, et al. PAAP: prefetch-aware admission policies for query results cache in Web search engines[C]//The 37th International ACM SIGIR Conference on Research & Development in Information Retrieval. 2014: 983-986.

[15] PODLIPNIG S, BOSZORMENYI L. A survey of Web cache replacement strategies[J]. ACM Computing Surveys (CSUR), 2003, 35(4): 374-398.

[16] ABRAHAMSSON H, NORDMARK M. Program popularity and viewer behaviour in a large TV-on-demand system[C]//The 2012 ACM Conference on Internet Measurement Conference. 2012: 199-210.

[17] ADHIKARI V K, GUO Y, HAO F, et al. A tale of three CDNs: an active measurement study of Hulu and its CDN[C]//2012 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS). 2012: 7-12.

[18] FREEDMAN M J. Experiences with CoralCDN: a five-year operational view[C]//NSDI. 2010: 95-110.

[19] SHAFIQ M Z, KHAKPOUR A R, LIU A X. Characterizing caching workload of a large commercial content delivery network[C]// IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications. 2016: 1-9.

[20] XU Q, ERMAN J, GREBER A, et al. Identifying diverse usage behaviors of smartphone Apps[C]//The 2011 ACM SIGCOMM Conference on Internet Measurement Conference. 2011: 329-344.

[21] MA Y, LIU X, ZHANG S, et al. Measurement and analysis of mobile Web cache performance[C]//The 24th International World Wide Web Conferences Steering Committee, 2015: 691-701.

[22] CHAUDHARI S S, GUPTA P. Proxy-side Web prefetching scheme for efficient bandwidth usage: a probabilistic method[J]. International Journal of Engineering, 2014, 3(6).

[23] NIKOLAOU S, VAN R R, SCHIPER N. Cooperative client caching

strategies for social and Web applications[J]. Large-Scale Distributed Systems and Middleware (LADIS). 2013.

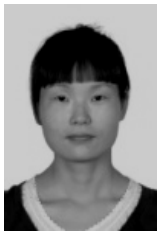
[24] XIE G, LI Z, KAAFAR M A, et al. Access types effect on internet video services and its implications on CDN caching[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2017.

[25] BONOMI F, MITZENMACHER M, PANIGRAHY R, et al. An improved construction for counting bloom filters[C]//European Symposium on Algorithms. 2006: 684-695.

[26] CROVELLA M E, BESTAVROS A. Self-similarity in world wide web traffic: evidence and possible causes[J]. ACM Sigmetrics Performance Evaluation Review, 1996, 24(1): 160-169.

[27] STATHOPOULOS A, KARLAFTIS M G. A multivariate state space approach for urban traffic flow modeling and prediction[J]. Transportation Research Part C: Emerging Technologies, 2003, 11(2): 121-135.

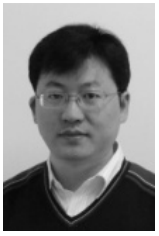
作者简介：



韩春静（1978-），女，河南郑州人，中国科学院计算技术研究所博士生，中国科学院信息工程研究所高级工程师、硕士生导师，主要研究方向为网络测量与行为分析、网络信息流识别与处理等。

杨晔（1995-），男，安徽池州人，中国科学院计算技术研究所博士生，主要研究方向为下一代互联网。

吕红蕾（1981-），女，河北邢台人，中国科学院信息工程研究所高级工程师、硕士生导师，主要研究方向为网络测量与行为分析、大规模网络流数据处理等。



葛敬国（1973-），男，安徽肥东人，博士，中国科学院信息工程研究所研究员、博士生导师，主要研究方向为网络体系结构与安全防护、网络测量与行为分析。

李佟（1978-），男，辽宁盘锦人，中国科学院信息工程研究所高级工程师、硕士生导师，主要研究方向为网络和计算机体系结构、网络测量与行为分析。

刘韵洁（1943-），男，山东烟台人，中国科学院院士，中国科学院计算技术研究所教授、博士生导师，主要研究方向为未来网络架构及关键技术、网络融合与演进。