

基于 P2P 的用户生产内容视频资源查找策略

李彦^{1,2*}, 陈卓¹

(1. 重庆理工大学 计算机科学与工程学院, 重庆 400054;

2. 重庆理工大学 机械检测技术与装备教育部工程研究中心, 重庆 400054)

(* 通信作者电子邮箱 ly@cqut.edu.cn)

摘要: 现有用户生产内容(UGC)类视频系统通常采用 C/S 架构设计, 导致了视频服务器极大的带宽压力。提出一种采用对等网(P2P)的在线短视频查找策略——FastSearch, 其目的是利用视频资源之间的关联关系进行视频资源定位, 以显著提高点播节点之间的视频分享效率并降低对视频服务器的带宽需求。实验表明 FastSearch 具备良好的视频数据源节点查找能力, 集成了该查找策略的短视频系统能有效减少对视频服务器的带宽消耗。

关键词: 用户生产内容; 对等网; 资源查找; 社会网络; 重叠网

中图分类号: TP319 **文献标志码:** A

Video resource search policy of user generated content based on P2P

LI Yan^{1,2*}, CHEN Zhuo¹

(1. College of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China;

2. Engineering Research Center of Mechanical Testing Technology and Equipment Ministry of Education, Chongqing University of Technology, Chongqing 400054, China)

Abstract: Present User Generated Content (UGC) video system mainly adopts the client/server architecture, which can result in huge bandwidth pressure on streaming server. This paper proposed a Peer-to-Peer (P2P) based online short video search policy—FastSearch. It aims to make use of the relevancy relationship between video resources to locate resource, which can improve the sharing efficiency between peers and decrease the consumption of streaming server. The simulation results show that FastSearch has an efficient streaming source peers searching ability and can greatly reduce the bandwidth consumption of streaming server.

Key words: User Generated Content (UGC); Peer-to-Peer (P2P); resource search; social network; overlay network

0 引言

NetTube 这类用户生产内容 (User Generated Content, UGC) 视频类分享网站在支撑大量点播的同时消耗了大量的网络带宽^[1-2]。大多数短视频分享网站都采用客户机/服务器架构实现, 以这种方法实现短视频的点播相对简单且当用户规模不大的时候点播响应较快, 但其带来的问题是视频服务器的带宽消耗巨大, 网站需要向运营商支付高额的带宽租用费。这一问题导致目前视频分享网站虽然有很大的点击量, 但盈利能力较弱。另外, 当访问用户量规模较大的时候也会使得缓存时间明显增加, 用户的体验质量 (Quality of Experience, QoE) 严重下降。

另一方面, 采用对等网 (Peer-to-Peer, P2P) 技术的视频应用越来越普遍, 已有的网络视频直播系统 Coolstreaming^[3] 和点播系统 PPLive^[4] 都采用 P2P 的方式实现。由于这类系统具备良好的可扩展性且视频服务器的带宽占用相对 C/S 模式大为降低^[4] 所以被众多视频内容提供商采用。但和较多的长视频不同的是, UGC 类视频分享自身的特点, 诸如: UGC 类视频通常视频长度短小; 另外, 在同一时刻观看同一个视频节目的观众通常并不是很多。这使得节点之间共享视频流的难

度增大。因此, 利用 P2P 技术部署 UGC 类视频的主要问题是: 如何在较短时间内高效地查找定位到视频源节点。本文通过把用户的点播兴趣和短视频之间具有的关联特性^[5] 相结合, 提出一种基于 P2P 技术的 UGC 视频资源查找策略——FastSearch, 重点讨论 FastSearch 中节点的点播兴趣偏好度量的方法及高效的 UGC 视频资源的查找定位策略。

1 相关研究工作

在 P2P 点播方面, 国内外学者做了很多研究工作, 如微软研究院的 Huang 等^[6] 提出在 VoD 引入 Peer 的互助以降低视频服务器开销, 在文献[4]中, 作者通过对 PPLive 系统的关键部件阐述及通过实际系统的实验说明 P2P VoD 系统的性能。对于 P2P VoD 的研究主要还是集中于长视频节目的点播研究上, 探讨的主要问题是有效支持 VCR 这类操作及如何实现优化的预取策略等, 这类 P2P VoD 系统中的视频文件本身没有社会网络特性且在同一时刻观看同一视频节目的节点数量众多, 这些特点都使得 P2P VoD 和在线短视频分享存在较大的技术差异。对于在线短视频分享方面, 文献[1]从实验的角度分析了 YouTube 这类在线短视频分享网站的网络流量特点及用户的点播行为, 但并没有考虑通过 P2P 的方

收稿日期: 2011-10-19; 修回日期: 2011-12-05。

基金项目: 国家科技部科技型中小企业创新基金资助项目 (10C26215113110); 重庆市教委科技项目 (KJ110831)。

作者简介: 李彦 (1976-), 男, 湖南岳阳人, 副教授, 博士研究生, 主要研究方向: 计算机网络; 陈卓 (1980-), 男, 重庆人, 讲师, 博士研究生, 主要研究方向: 分布式网络、网络协议。

式对视频网站的带宽压力进行优化。文献[5]通过 tracking 的方式对 YouTube 进行研究,得出了视频文件之间存在社会网络(Social Network)的特点,提出了一种采用视频服务器 + P2P 的方式实现在线视频分享系统 NetTube^[5],通过仿真实验表明引入 P2P 技术后可以降低服务器带宽消耗约 70%。但作者也谈到 NetTube 在视频数据源节点的查找方面仍需要更多的研究,特别是当用户没有从相关视频列表选择其他视频观看时(即前后两个视频的关联度不大),如何以较低开销查找到缓存了该视频节目的节点仍不够理想。另外,当缓存满后应该采用怎样的缓存管理策略也需要进一步研究。

2 视频数据源节点查找策略

在 UGC 视频系统中引入 P2P 技术的主要目的是通过增加节点之间的数据互享的机会,使点播节点优先依靠其他在线节点获取视频流,最终降低视频服务器的带宽消耗。因此高效的视频数据源查找策略非常重要。UGC 短视频系统中的查找策略需要面对如下困难。1) 视频类应用具有较强的时间紧迫性,这一特点使得查找算法效率必须较高。如果在线节点中存在部分缓存了用户所点播的视频 v 的节点,则需要能够较快地找到这些缓存节点。2) 视频源节点的资源查找开销应该控制在合理范围,这也是 NetTube 中只采用两跳洪泛的主要原因,但如果洪泛的节点范围较小,又难以找到较多的数据源节点,因此查找策略面临在查找开销和查找到的数据源节点的数量上的权衡。3) 为了保证用户的点播感受度 QoE ,在一定时间内如果仍然难以找到数据源节点,如视频流行度较低则需要及时向视频服务器请求获得视频数据。因此查找策略需要初步估计在一定查找范围内(TTL),能够找到部分数据源节点的概率。

2.1 查找策略描述

FastSearch 的资源查找策略需要重叠网络(Overlay Network)的支持,并建立在用户点播兴趣和视频节目的社会特性基础之上,更全面地考虑到了节点实际的点播行为。把用户点播行为分为两类:第一类称为关联视频点播行为,即用户从与当前正在观看的视频的相关视频列表中选择下一个观看的视频,这种情况下前后两次点播的视频存在较密切的关联关系;第二类称为无关联视频点播行为,即用户点播的视频和当前正在观看的视频不属于同一类型的视频,这种情况下前后点播的两个视频没有在密切的关联关系。这里假设用户的点播行为属于这两种点播行为之一。

另外,一个节点 i 的邻居应分别属于三种类型,即具有强关联邻居关系的邻居、具有弱关联邻居关系的邻居和具有社会网络关系的邻居。本文把这三类邻居构成的集合分别称为 $SNS(i)$ 、 $WNS(i)$ 和 $SocialNS(i)$ 。

FastSearch 的视频查找策略是一种具有视频关联关系意识的查找策略,UGC 视频系统中视频文件之间存在的社会网络特性使得点播视频的用户之间的点播行为也呈现一定的关联关系,即用户往往会从那些和当前播放的视频存在一定关联关系的相关视频中选择一个作为下一个点播的视频^[5]。FastSearch 的主要利用了这一特性,有偏向发送查找请求给那些和查找视频具有点播关联关系的节点,而不是简单采用如

NetTube 所采用的洪泛的方法查找。FastSearch 的查找策略可以看作是一种有偏向的 Random Walk 查找,向邻居发送的一个查找请求即是一个“Walker”。使用 Random Walk 的方法使得查找开销得到控制,而有偏见的选择节点使得查找命中率得到显著的提升。

在节点 i 访问 FastSearch 并建立邻居关系(包括建立三种类型的邻居关系)的时候,节点间同时需要交换自己的缓存视频信息列表(VideoList)。经过和邻居间的视频列表信息交换,节点 i 则能够知道每个邻居节点当前所缓存的视频信息。另外,节点 i 看完当前视频并选择下一个观看的视频假设为 V_n^i ,把 V_n^i 相关联视频 $V_{n,1}^i, \dots, V_{n,j}^i, \dots, V_{n,k}^i$ 构成的关联视频集合定义为 $RVList_p^i$,这里假设每个视频的关联视频个数为 k ,其中 $V_{n,j}^i$ 表示 V_n^i 的第 j 个关联视频。根据保存的邻居视频列表,节点 i 可以知道邻居节点中那些缓存了 $V_{n,1}^i, \dots, V_{n,j}^i, \dots, V_{n,k}^i$ 中的部分视频。假设节点 i 的 m 个邻居组成的集合为 $NSet_i = \{nb_1^i, \dots, nb_j^i, \dots, nb_m^i\}$,邻居 $nb_j^i (1 \leq j \leq m)$ 缓存了 $V_{n,1}^i, \dots, V_{n,j}^i, \dots, V_{n,k}^i$ 中的视频个数表示为 $Bnum_j^i (0 \leq Bnum_j^i \leq k)$ 而满足 $Bnum_j^i \neq 0$ 的节点组成集合为 $RSet_i$,查找策略会选取 $RSet_i$ 中的节点并发送跳数为 TTL 的查询请求。另外,节点 i 会同时向 $RSet_i$ 中的邻居节点发送视频的关联关系视图 $VRMap = \{V_n^i, V_{n,1}^i, \dots, V_{n,j}^i, \dots, V_{n,k}^i\}$ 。 $RSet_i$ 中的节点在收到查询请求后,会把 TTL 减 1,如果不为零则按照如上的方式,把查询 V_n^i 的请求消息有偏向地发送给那些缓存了 $V_{n,1}^i, \dots, V_{n,j}^i, \dots, V_{n,k}^i$ 中一个或多个视频的邻居节点。请求转发查找结束的条件为 TTL 为零或者邻居节点中没有任何一个缓存了的查询请求。

图 1 表示了 FastSearch 中具备视频关联关系意识的查找策略的运行情况示例。节点 i 点播了视频 v_2 ,同时存在关于 v_2 的视频关联关系视图 $VRMap = \{v_2 | v_3, v_4, v_5, v_6, v_7\}$ 。节点 i 的三个邻居中 N_2 缓存了和 v_2 存在关联关系的 v_3 和 v_7 ,而 N_3 缓存了 v_2 及和 v_2 存在关联关系的 v_3 。节点 i 在 req 请求消息中记录 v_2 的关联视图和 TTL 后,把消息发送给 N_2 和 N_3 ,如图 1(a)。由于自己没有缓存 v_2 ,则把 TTL 减 1,不为零则继续发送请求给邻居,即图 1(b) 所示。 N_3 可以直接回复节点 i ,表示可以向节点 i 提供 v_2 的视频流,同时当 TTL 不为零的时候也同时向自己的邻居 N_8 转发该查询请求,如图 1(c) 所示。

FastSearch 的查找策略具有视频关联关系意识,当 TTL 不为零的时候,始终把查找请求有偏向地进行转发,转发的节点具备两种特点之一:1) 缓存了要查找的视频;2) 缓存了要查找的视频存在关联关系的视频。而对于那些没有缓存和点播视频有任何关联关系视频的节点,FastSearch 的查找策略回避了向这类节点的转发查询请求,这里利用的思想就是 UGC 类视频系统存在的社会网络性质,点播了视频 v 的节点具有很大概率点播与 v 存在关联关系的视频^[5]。查找算法的伪码如下:

Input:

v : the short video which is selected by N_i to watch next

$T(v)$: the type of short video v

$StrongNbr_i$: the strong neighbor list of N_i

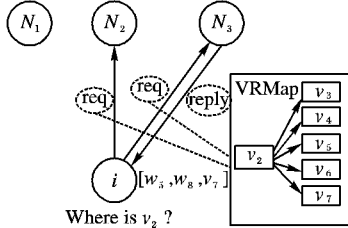
$WeakNbr_i$: the weak neighbor list of N_i

$SocialNbr_i$: the neighbor list of N_i based on social relationships

Output:

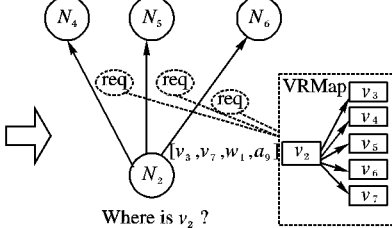
Streaming source peers SP_v which can provide v to N_i
 if $T(v) = T_{fav}(N_i)$ then
 $SP_v \leftarrow \text{SearchSP}(\text{StrongNbr}_i, v)$;
 $SP_v \leftarrow SP_v + \text{SearchSP}(\text{SocialNbr}_i, v)$;
 endif
 if $T(v) \neq T_{fav}(N_i)$ then
 if can find WeakNbr_i which $T_{fav}(\text{WeakNbr}_i) = T(v)$ then

$[a_7, a_8, w_5, c_2] [v_3, v_7, w_1, a_9] [v_2, v_3, v_6, c_2, w_8]$



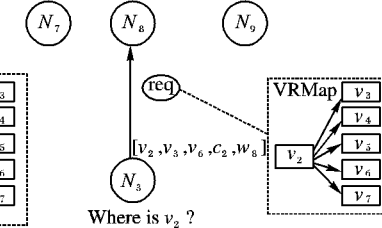
(a) 节点*i*进行有偏的查找及视频缓存关系

$[v_2, v_9] [v_4, w_7, w_8] [v_2, v_3, v_4, v_5]$



(b) 节点*N₂*进行有偏的查找及视频缓存关系

$[w_1, w_5] [v_6, w_7, w_8] [a_5, a_6, c_2, w_3]$



(c) 节点*N₃*进行有偏的查找及视频缓存关系

图1 FastSearch 的查找消息转发

该查找算法主要把查找数据源节点分成三种情况考虑:

第一种情况是节点选择点播的视频 v 是节点最感兴趣的视频类型 $T(v) = T_{fav}(N_i)$, 则节点同时通过向自己基于兴趣邻居中的强关联邻居和基于社会特性建立起来的邻居发送查找视频 v 的请求。第二种情况是视频 v 不属于节点最感兴趣的视频类型即 $T(v) \neq T_{fav}(N_i)$, 节点需要确定自己保存了到类型为 $T(v)$ 的簇的连接, 也即保存有到 $T(v)$ 类型的节点的弱连接, 如果有这样的弱连接则向这种类型的弱连接邻居发出查找 v 的请求。节点同时向基于社会特性建立起来的邻居发送查找请求。第三种情况是经过前面的查找算法仍然无法找到视频数据源节点则只能向 Tracker 服务器请求获取正在播放或者缓存了 v 的节点。

2.2 视频流行度的估计

FastSearch 中, 节点之间交换视频的关联关系视图 VRMap 不仅帮助节点实现基于关联关系的视频源查询, 同时节点通过对多个视频之间的关联关系的分析还能够得到视频流行度的估计 (Video Popularity Estimation), 这有助于 FastSearch 更有效地预取流行度高的视频, 使得预取的视频被用户点播的概率得到较大提高。

通过图2说明对视频流行度的估计, 图3中节点 i 在一段时间 t 内收到其3个邻居 N_1 、 N_2 和 N_3 发送的四个视频关联关系视图, 分别为 $\text{VRMap}_{N_1 \rightarrow i} = \{k | b, c, w, v\}$, $\text{VRMap}_{N_2 \rightarrow i} = \{f | e, m, b, c\}$ 和 $\text{VRMap}_{N_3 \rightarrow i} = \{a | b, f, e, d\}$, $\text{VRMap}_{N_3 \rightarrow i} = \{b | a, k, f, w\}$, 则节点 i 能够记录下各个视频的关联关系为: $a: b, f, e, d$; $b: k, w, m, u, a, f$; $c: f, k$; $w: k, b$; $d: a$; $e: f$; $m: b$; $v: k$ 。可以看到, 在这段时间内, 节点 i 可以分析出各视频 b 和其他视频的关联关系是最多的, 也说明视频 b 的流行度最高。

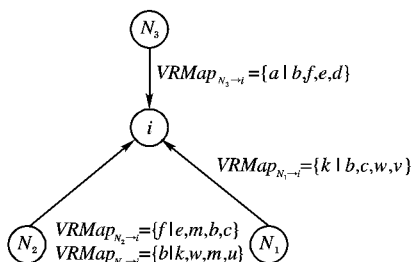


图2 节点建立视频流行度的估计关系

$SP_v \leftarrow \text{SearchSP}(\text{WeakNbr}_i \text{ which } T_{fav}(\text{WeakNbr}_i) = T(v), v)$;
 endif
 $SP_v \leftarrow SP_v + \text{SearchSP}(\text{SocialNbr}_i, v)$;
 endif
 if $SP_v = \emptyset$ then
 $SP_v \leftarrow \text{SendReqToServer}(v)$;
 endif

2.3 查找跳数 TTL 的确定

在查找算法里多次调用了参数不同的 $\text{SearchSP}(T, v)$ 函数, 其主要作用是向不同类型的邻居节点请求查找视频 v , 如果邻居缓存了 v 则向请求的节点返回响应, 如果没有缓存则继续向邻居的邻居进行查找。这里需要重点考虑的一个问题是如何确定查找跳数 TTL (hop)。在 NetTube 中, 简单地设置查找跳数 TTL 为 2 跳, 目的是为了降低 Flooding 带来的开销。而在实际情况下, 节点对点播视频 v 的查找跳数 TTL 存在如下几个特点: 1) TTL 和视频 v 的“流行度”密切相关, “流行度”越高的视频被点播节点缓存的概率也越大, 因此在 TTL 较小的情况下能达到较好的查找效率; 2) TTL 和类型为 $T(v)$ 的簇的局部紧密程度相关。簇中节点的邻居数较少则局部紧密度也就较小, 因此适当增加 TTL 不会带来较大的网络开销。

文献[5, 10]通过大量实验数据得出在线视频分享中视频的点播次数和“流行度”排名基本满足 Zipf 分布。假设节点点播的视频 v 的流行度排名为 k , 而 FastSearch 中所有视频文件数量为 M , Zipf 分布是指数特性参数为 S 。则根据 Zipf 分布的性质, 节点点播流行度排名为 k 的视频 v 的概率可表示为:

$$f(k; s, M) = \frac{1/k^s}{\sum_{n=1}^M (1/n^s)} = \frac{1}{k^s H_{M,s}} \quad (1)$$

$H_{M,s}$ 为谐参数 (Harmonic number), $H_{M,s} = \sum_{k=1}^M \frac{1}{k^s}$, 设

$S = 1$ 时, $H_{M,1} = \sum_{k=1}^M \frac{1}{k} = \ln M + \gamma$, γ 是 Euler-Mascheroni 系数, 值为 0.577 215 664 9。通过式(3) 可以看到排名越靠前 (k 越小) 则节点点播了该视频的概率越大。另外设在线的点播节点数为 N , 那么在线节点中点播并缓存视频 v 的节点数 N_k 可粗略估计为:

$$N_k = N \cdot f(k; s, M) = \frac{N}{k(\ln M + \gamma)} \quad (2)$$

因此节点 i 要查找到视频 v 需要访问的节点数量为:

$$N_v = \left\lceil \frac{N}{N_k} \right\rceil = \lceil k(\ln M + \gamma) \rceil \quad (3)$$

通过式(5) 也可以看到如果视频 v 的流行度越高则需要

访问较少的节点就可以找到缓存了该视频的节点。另外,还需要考虑的是类型为 $T(v)$ 的簇的紧密程度,这里采用文献[11]中的方法,对一个属于类型为 $T(v)$ 兴趣簇的节点 j 其局部簇系数可表示为 $CC_j = |E(F_j)| / C_{Nb_j}^2$, 其中: F_j 为节点 j 的邻居集合; $|F_j|$ 为节点 j 的邻居的总数; $|E(F_j)|$ 表示 F_j 集合中实际的连接总数; $C_{Nb_j}^2$ 表示 F_j 集合中可能产生的连接总数, CC_j 越大就表示簇的紧密程度越大。再假设 FastSearch 中一个节点的强邻居节点个数为 x , 那么在 TTL 跳内可以访问到的 $T(v)$ 类型簇的节点个数为 $\sum_{t=1}^{TTL} (x')^t$, 其中 $x' = x \cdot (1 - CC_j)$ 。所以有:

$$\sum_{t=1}^{TTL} (x')^t \geq N_v \quad (4)$$

$$TTL \geq \log_{x'} \left(\frac{(x' - 1)N_v}{x'} + 1 \right) \quad (5)$$

FastSearch 通过式(2)~(5)粗略求出一个“流行度”排名为 k 的视频 v 的查找跳数 TTL 。由于 FastSearch 按照节点的主要点播兴趣偏好把节点进行了分簇,并把视频文件之间的关联考虑进去。所以实际需要的 TTL 值通常比式(5)求出的理论值更小就可以达到较好的查找效率。

2.4 查找策略的性能分析

假设节点 i 点播的视频 v 的流行度 P_v , 和视频 v 关联的视频个数为 M 个, 而每个节点平均有 N 个邻居节点。另外, 定义第 l 层节点到第 $l+1$ 层节点的查询请求转发表示请求消息增加 1 跳, 如图 1(a) 中节点 i 向 N_2 和 N_3 发送查询请求为第 1 跳, 而 N_2 转发请求消息给 N_4, N_5 和 N_6 及 N_3 转发消息给 N_7 均属于第 2 跳。 $P_{k,j}^i (1 \leq k \leq N)$ 表示节点 i 的第 k 个邻居节点在第 j 跳能够找到视频 v 的源节点的概率。如果节点间在没有视频关联关系信息的时候, 在第 j 跳能够找到视频 v 的概率为 p_v , 即只与流行度相关。而在邻居节点间具备邻居缓存的视频列表, 并有关联意识的进行查询时第 j 跳能够找到视频 v 的概率不仅和 p_v 相关, 同时还和第 $j-1$ 跳的节点所缓存的关联视频个数有关。给出如下条件概率 P_j , 表示在第 $j-1$ 跳的节点缓存有 λ 个和视频 v 相关联的视频的条件下(具备关联关系), 如果把查询消息转发给该节点则在第 j 跳成功查找到视频 v 的概率:

$$P_{nb_i}^j = P(F_{nb_i}^j > 0 | B_i^{j-1} = \lambda) = (1 + \lambda/M_v) \cdot p_v \quad (6)$$

$$p_v = |S_v| / n \quad (7)$$

在式(6)中: B_i^{j-1} 表示第 $j-1$ 跳的某个节点 i 缓存的与 v 相关的视频个数为 λ ; $F_{nb_i}^j$ 表示节点 i 把查询消息转发给自己的邻居节点即第 j 跳, 能够找到视频 v 的个数; M_v 表示和视频 v 相关联的视频个数; S_v 表示当前在线的 n 个节点中缓存了视频 v 的节点集合; $|S_v|$ 表示缓存了视频 v 的节点的数量。式(7)中 p_v 则表示视频 v 的流程度。显然, 能够查找到视频 v 的概率和不仅和视频 v 自身的流行程度相关同时还和视频关联关系的紧密程度相关, 如果节点 i 缓存的和视频 v 相关的视频个数越多, 则表明 i 的邻居节点缓存了越多的和 v 相关的视频, λ 越大则表明下一条邻居节点那里越容易找到视频 v 。

假设 FastSearch 中一个节点的平均邻居节点个数为 k , 而节点 i 的邻居节点中如果有 $\theta (\theta < k)$ 个邻居节点 N_1, \dots, N_θ 缓

存了至少一个和视频 v 存在关联关系的视频, 把查询消息转发给这 θ 个节点, 根据式(6)可得到在这 θ 个节点的邻居节点处(也即下一跳)查找到视频 v 的概率分别为:

$$\begin{cases} P_{nb_{N_1}}^j = P(F_{nb_{N_1}}^j > 0 | B_{N_1}^{j-1} = \lambda_1) = (1 + \lambda_1/M_v) \cdot p_v, \\ \lambda_1 < M_v \\ P_{nb_{N_2}}^j = P(F_{nb_{N_2}}^j > 0 | B_{N_2}^{j-1} = \lambda_2) = (1 + \lambda_2/M_v) \cdot p_v, \\ \lambda_2 < M_v \\ \vdots \\ P_{nb_{N_\theta}}^j = P(F_{nb_{N_\theta}}^j > 0 | B_{N_\theta}^{j-1} = \lambda_\theta) = (1 + \lambda_\theta/M_v) \cdot p_v, \\ \lambda_\theta < M_v \end{cases} \quad (8)$$

进一步, 在把查询消息发送给这 θ 个节点后, 可以查询到 θ 个缓存了视频 v 的数据源节点的概率为:

$$P_j = P(F^j = \theta) = P_{nb_{N_1}}^j \cdot P_{nb_{N_2}}^j \cdot \dots \cdot P_{nb_{N_\theta}}^j = (1 + \lambda_1/M_v) (1 + \lambda_2/M_v) \dots (1 + \lambda_\theta/M_v) p_v^\theta \quad (9)$$

下面对比在 k 个邻居节点中任意选择 θ 个发送查找请求的查找效率。当不可考虑视频之间的关联关系, 也即节点 i 发送 k 个查询消息给各个邻居节点。由于不考虑关联关系, 查找成功查找概率只和视频 v 的流行度 p_v 相关。因此在 k 个邻居处查找到 θ 个缓存了视频 v 的源节点概率为:

$$P_j' = P(F^j = \theta) = p_v^\theta (1 - p_v)^{n-\theta} \quad (10)$$

而 $\frac{P_j}{P_j'} = \frac{(1 + \lambda_1/M_v) \dots (1 + \lambda_\theta/M_v)}{(1 - p_v)^{n-\theta}} > 1$, 也即

FastSearch 中通过有选择性地发送 θ 个查询消息比任意选择 θ 个发送的查找成功率更高。

3 实验仿真

本文采用 OverSim 系统^[7,12]对 FastSearch 仿真, 在关联度较大的时候, 通过 Flooding 策略和 FastSearch 算法通常都可以得到较好的资源命中率。FastSearch 的主要优势是能够在视频关联度不大的时候可以获得可接受的视频查找效率。所以, 这里重点比较前后两次视频点播的视频节目在没有强关联性的时候的查找效率。通过图 3 的对比可以看到 FastSearch 的查找效率显著优于 NetTube 这类采用 Flooding 策略的典型系统, 在这种情况下 IShare 的平均查找效率基本达到了 60%, 而部分情况下 NetTube 却基本无法直接通过 P2P 的方式找到源节点而必须借助于服务器的帮助, 而这显然又增加了服务器的压力。

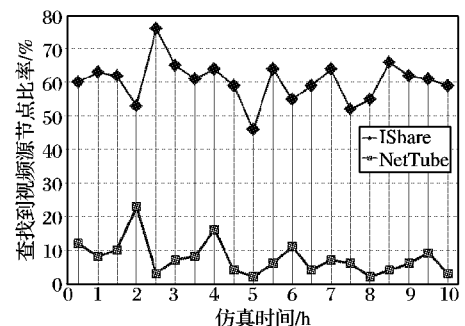


图3 视频关联度较小时查找视频源节点的百分比

实验对服务器的带宽消耗也进行了比较, 对比了

FastSearch、NetTube 和 C/S 架构时视频服务器的带宽消耗。图 4 中可以看到 C/S 架构的带宽消耗最大,在同样数量的用户点播请求下达到了接近 4 Gbps。FastSearch 比 NetTube 有了更好的改善,能够节约 75% 的带宽消耗,这是通过进一步增加节点间的数据共享实现的。

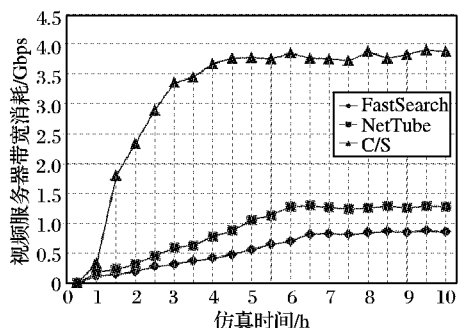


图 4 视频服务器的带宽消耗对比

4 结语

本文提出了一种基于 P2P 的在线视频资源查找策略,该策略在前后两次点播的视频资源的关联关系较小的时候仍然可以通过较小的资源开销获得较大的资源查找效率。通过实验仿真表明了 FastSearch 比现有的系统在资源查找效率上具有更优的效率。下一步,将继续通过在现实系统中部署 FastSearch 策略并进一步对系统加以完善。

参考文献:

- [1] GILL P, ARLITT M, LI Z, *et al.* YouTube traffic characterization: A view from the edge [C]// IMC'07: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement. New York: ACM, 2007: 15–28.
- [2] CORBETT C. Peering of video [EB/OL]. [2006–09–10]. <http://www.nanog.org/mtg-0606/pdf/bill.norton.3.pdf>.
- [3] ZHANG XINYAN, LIU JIANGCHUAN, LI BO, *et al.* CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming [EB/OL]. [2011–04–01]. <http://www.cs.sfu.ca/~jcliu/Papers/CoolStreaming.pdf>.
- [4] YAN HUANG, FU T Z J, CHIU D-M, *et al.* Challenges, design and analysis of a large-scale P2P-VoD system [C]// SIGCOMM'08: Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication. New York: ACM, 2008: 375–388.
- [5] CHENG XU, LIU JIANCHUN. NetTube: Exploring social networks for peer-to-peer short video sharing [C]// Proceedings of the 28th IEEE Conference on Computer Communications. [S. l.]: IEEE, 2009: 1152–1160.
- [6] HUANG CHENG, LI JIN, ROSS K W. Can Internet video-on-demand be profitable? [C]// SIGCOMM'07: Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. New York: ACM, 2007: 133–144.
- [7] BAUMGART I, HEEP B, KRAUSE S. OverSim: A scalable and flexible overlay framework for simulation and real network applications [EB/OL]. [2011–09–01]. http://www.tm.uka.de/doc/p2p09_oversim.pdf.
- [8] XU K, LI HAITAO, LIU JIANGCHUAN, *et al.* PPVA: A universal and transparent peer-to-peer accelerator for interactive online video sharing [C]// IWQoS'10: 18th International Workshop on Quality of Service. [S. l.]: IEEE, 2010: 1–9.
- [9] Euclidean distance [EB/OL]. [2011–03–09]. http://en.wikipedia.org/wiki/Euclidean_distance.
- [10] CHA M, KWAK H, RODRIGUEZ P. I Tube, You Tube, Everybody Tube: Analyzing the world's largest user generated content video system [C]// IMC'07: Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement. New York: ACM, 2007: 1–14.
- [11] HUI K Y K, LUI J C S, YAU D K Y. Small-world overlay P2P networks: Construction and handling dynamic flash crowd [J]. Computer Networks, 2006, 50(15): 212–221.
- [12] BAUMGART I, HEEP B, KRAUSE S. A P2P PSIP demonstrator powered by OverSim [C]// P2P'07: Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing. Washington, DC: IEEE Computer Society, 2007: 243–244.
- [13] HEI X J, LIANG C, LIANG J, *et al.* Insights into PPLive: A measurement study of a large-scale P2P IPTV system [J]. IEEE Transactions on Multimedia, 2007, 9(8): 1672–1687.
- [14] VLAVIANOS A, ILIOFOTOU M, FALOUTSOS M. BitToS: Enhancing BitTorrent for supporting streaming applications [C]// 25th IEEE International Conference on Computer Communications. Barcelona: IEEE Press, 2006: 1–6.
- [15] ZHANG X Y, LI J C, LI B, *et al.* CoolStreaming/DONet: A data driven overlay network for live media streaming [EB/OL]. [2011–09–10]. <http://www.cs.sfu.ca/~jcliu/Papers/CoolStreaming.pdf>.
- [16] LIAO X F, JIN H, LIU Y H, *et al.* AnySee: Peer-to-peer live streaming [C]// 25th IEEE International Conference on Computer Communications. Barcelona: IEEE Press, 2006: 1–10.
- [17] XU H, WANG S P, WANG R C, *et al.* Improving QoS in peer-to-peer streaming media system [J]. Journal of Computation System, 2010, 6(5): 1387–1395.
- [18] ZHANG M, XIONG Y Q, ZHANG Q, *et al.* On the optimal scheduling for media streaming in data-driven overlay networks [C]// IEEE Globecom 2006. San Francisco: IEEE Press, 2006: 5.
- [19] 陈瑞昭, 刘永广. 基于能力因子的 P2P 邻居节点随机可变选择算法 [J]. 计算机应用, 2010, 30(2): 327–329.
- [20] XIE S S, LI B, ZHANG X Y, *et al.* CoolStreaming: Design, theory, and practice [J]. IEEE Transactions on Multimedia, 2007, 9(8): 1661–1671.
- [21] Wikimedia Foundation, Inc. Mercator projection [EB/OL]. [2011–02–07]. http://en.wikipedia.org/wiki/Mercator_Projection.
- [22] DANIEL C, PHILIPPAS T. A practical quicksort algorithm for graphics processors [C]// 16th Annual European Symposium on Algorithms. Karlsruhe: Springer-Verlag, 2008: 246–258.
- [23] LI B, HAO Y. Peer-to-peer live video streaming on the Internet: Issues, existing approaches, and challenges [J]. IEEE Communication Magazine, 2007, 45(6): 94–99.
- [24] OPNET Technologies, Inc. OPNET [EB/OL]. [2011–03–08]. <http://www.opnet.com/>.
- [25] LIOFOTOU M, PAPPU P, FALOUTSOS M, *et al.* Graph-based P2P traffic classification at the Internet backbone [C]// INFOCOM'09: Proceedings of the 28th IEEE International Conference on Computer Communications Workshops. Piscataway: IEEE Press, 2009: 37–42.

(上接第 937 页)

参考文献: