硕 士 学 位 论 文

基于视频转码和视频推荐的缓存设计

Video Transcoding and Video Recommending based Caching Design

作者：赵红娜

导师：李纯喜

北京交通大学

2020 年 6 月

学位论文版权使用授权书

本学位论文作者完全了解北京交通大学有关保留、使用学位论文的规定。特授权北京交通大学可以将学位论文的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。同意学校向国家有关部门或机构送交论文的复印件和磁盘。学校可以为存在馆际合作关系的兄弟高校用户提供文献传递服务和交换服务。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：赵红娜　　导师签名：李德喜

签字日期：2020 年 5 月 25 日　　签字日期：2020 年 5 月 27 日

学校代码：10004 密级：公开

北京交通大学

硕士学位论文

基于视频转码和视频推荐的缓存设计

Video Transcoding and Video Recommending based Caching Design

作者姓名：赵红娜 学　　号：17120196

导师姓名：李纯喜 职　　称：副教授

学位类别：　工程 学位级别：硕士

学科专业：信息安全 研究方向：信息网络

北京交通大学

2020 年 6 月

致谢

本论文是在我的导师李纯喜的悉心指导下完成的。在研究生期间，李纯喜老师始终耐心细致的指导着我，帮助我确定研究方向，完成小论文的撰写与发表和毕设论文的撰写等等。李老师严谨的科研态度，渊博的学识会一直激励我在以后的生活中不断努力进取。在此，向李纯喜老师表达我由衷的谢意。

同时，我要感谢赵永祥老师，邓宏云老师，郭宇春老师和其他老师在我研究生期间对我的帮助与指导。这些老师不仅教会我如何思考问题，还教会我如何表达问题，让我受益良多。

其次，还要感谢实验室的张加林师兄、王唯师姐、高志朋，冯梦菲，尹姜谊，薛会、张虎信等同学以及我的舍友杨晶晶和穆鸽，对我的帮助与支持。感谢大家在我的学习、生活、就业等问题上，积极给予我很多有效的建议与启发，感谢大家陪伴我度过这段难忘的学习生涯。

最后，特别感谢一直支持我的家人和其他朋友，正是他们默默的支持与付出，我才能顺利完成我的学业，找到心仪的工作。

摘要

缓存算法是影响缓存服务器缓存性能的关键因素。传统的缓存算法通常以视频文件的流行度衡量一个文件的价值，优先缓存最有价值的视频文件。近年来，随着终端设备的多样化，一个视频通常需要提供不同清晰度版本以匹配不同终端用户的需求，而这些视频版本能够通过视频编码技术转换，从而给缓存进一步优化设计提供了机会；例如，分层视频编码 SVC 技术已经被用于设计缓存，通过缓存不同清晰度视频版本中最流行的视频层来提升缓存性能。另一方面，随着视频内容的极大丰富，推荐技术的成熟，越来越多的用户习惯于观看系统推荐的视频，使得视频缓存页也可以结合推荐技术来提高缓存性能。

与现有研究不同，一方面，本文考虑到视频转码技术有更好的系统兼容性，提出基于视频转码的缓存优化设计，通过使缓存服务器对每个视频最多保留一份最有价值的版本来提高缓存性能；另一方面，现有的基于视频编码和基于推荐的缓存都是各自独立研究的，本文提出了联合视频转码和推荐技术的缓存的研究，以进一步提高缓存系统的性能。具体来说，本文的主要贡献如下。

(1) 建立了基于视频转码的缓存优化模型，提出了基于转码的缓存算法，以减小视频传输时延。算法以一个待缓存的视频文件（若缓存）能够在已缓存视频文件的基础上能进一步满足的用户观看请求而带来的传输时延节省来定义文件价值，通过贪婪迭代的方式找到最有价值的视频版本来存贮，直到没有新的文件可放入缓存系统。仿真实验结果表明，相较于传统的基于视频流行度的缓存算法，本文提出的缓存算法能够使平均传输时延降低大约 40%，远高于传统算法。

(2) 建立了联合转码和推荐缓存优化模型，提出了基于转码与推荐的缓存算法，以进一步减小视频传输时延。算法以一个待存储的视频版本（若缓存）能够在已缓存视频文件的基础上进一步能满足的用户请求而带来的传输时延的减少为基础来计算文件价值，通过贪婪迭代的方式存储最有价值的视频版本。仿真实验结果表明，本文提出的算法的缓存性能更好，尤其在缓存空间较小时，优势更加显著。

关键词：边缘缓存；视频编码；视频转码；视频推荐

ABSTRACT

Cache algorithm is a key factor that affects the cache performance of cache server(s). Traditional caching algorithms usually measure the value of a video based on the popularity of the video file and preferentially buffer the most valuable videos. In recent years, due to the diversification of terminal devices, a video usually needs to provide different bitrate versions to adapt the demands of different terminal users. Since such video versions can be converted by video encoding technology, it provides opportunities for further optimization of caching design. For example, Scalable Video Coding, SVC has been used for caching design to improve caching performance by caching the most popular video layers. On the other hand, as the video contents are greatly enriched and recommendation technologies, more and more users are accustoming to accept the recommended videos to watch, such that video recommendation can also be utilized to design caching to improve caching performance.

Different from the existing researches, on the one hand, considering the better compatibility of video transcoding technologies, we proposes a video transcoding based caching model, which enables the cache server to buffer at most one valuable version for each individual video to improve caching performance;

on the other hand, considering that the existing video encoding based and video recommending based caching are independently studied, we propose to jointly use the two types of technologies to design cache, to further improve the performance. Concretely, the contributions are as follows.

(1) This paper establishes a transcoding based cache optimization model and proposes a corresponding cache algorithm to reduce video delivery delay. The algorithm defines the value of a to-be-cached file based on the further reduction in delivery delay considering that once it is selected to buffer it can satisfy more users' viewing requests and thus delivery delay can be further reduced. The algorithm then iteratively finds the most valuable video versions to cache in a greedy manner, one version each iteration, until no video files can be put into the cache system. Simulation results show that the proposed transcoding based caching algorithm can reduce the average delivery delay by up to 40%, much higher than the traditional algorithm.

(2) This paper establishes a joint caching optimization model based on both video transcoding and recommending and proposes a corresponding heuristic algorithm to improve caching performance. We define the value of a video version based on the further

delivery delay reduction benefited due to the further more satisfied users' demands by this video version if it is buffered. We then propose the algorithm to iteratively find the most valuable video version to buffer in a greedy manner. Simulation results show that the proposed algorithm has significant better caching performance when the cache size is relative small.

KEYWORDS： Edge Caching; Video Encoding; Video Transcoding; Video Recommending;

目录

1 引言

移动终端的便捷使用带来了视频流量的爆炸式增长，进而给网络带宽带来了巨大的压力。运营商通常在靠近用户侧，部署缓存服务器存储视频以缓解带宽压力，提高用户的视频观看体验质量。缓存算法是决定存储哪些视频文件，影响缓存性能的重要因素。传统的视频缓存算法，每个视频都被看作是一个独立的文件，并根据视频流行度存储流行的视频文件。随着终端设备的多样化，一个视频通常需要以不同清晰度提供给不同终端用户观看，而这些视频版本能够通过视频编码技术转换，因此利用视频编码设计缓存，能够保证一个视频至多一个版本存储，消除缓存冗余，提高缓存性能。此外，由于推荐技术的发展，用户更愿意观看推荐的视频，因此结合推荐技术设计缓存，能够进一步满足用户的视频观看请求，提高缓存性能。注意到转码技术对现有视频系统的兼容性更好，本文的研究课题之一是利用视频转码设计视频缓存来消除缓存冗余；注意到基于推荐技术的和基于视频清晰度转换技术的缓存都是各自独立研究的，本文的另一个研究课题是联合视频转码和推荐技术设计视频缓存，不仅能够消除缓存冗余，而且能够进一步提高缓存性能。

1.1 研究背景与意义

随着技术的发展，移动网络逐渐成为人们生活、学习和娱乐不可或缺的一部分，移动流量也呈现出爆炸式增长趋势。思科公司的报告预测，2022 年全球移动数据流量可达 77.5EB/月，其中视频流量将占全部流量的 79%[1]。如此庞大的视频流量给回程链路（backhaul link）带来巨大的带宽压力。

为了缓解网络带宽压力和提升用户体验，近年来，随着边缘计算的发展[2-4]，人们引入了边缘缓存，通过在网络边缘靠近用户侧，部署缓存服务器存储视频文件来帮助视频传输与分发[5-8]。该技术通过给边缘节点（如小区接入点或小基站）配备缓存服务器，使得用户能够通过边缘节点直接从缓存服务器获取视频服务，以减少用户通过回程链路对视频源服务器的访问次数。边缘缓存服务器的存储能力通常有限，仅能够存储一部分视频文件，因此，如何在存储资源有限的条件下，使缓存服务器存储能够满足更多用户观看需求的视频文件（即高效的缓存算法设计）是一个非常关键的问题，一直受到持续关注。

传统的缓存算法[9-11]普遍通过流行度来衡量视频文件的价值，并选择流行度较高的视频文件进行存储。根据视频观看记录，人们发现不同视频的观看次数服从 Zipf 分布[12,13]，也就是说只有很小比例的视频被大多数用户请求观看。据此可以统

计出文件的访问频率，即流行度，存储流行度最高的文件就能满足大多数用户的需求。这些缓存算法将每个视频文件看作一个独立的个体，以文件流行度这个单一因素定义文件的价值，没有考虑文件之间可能存在的关联关系。

实际上，考虑到一个视频通常需要有多个清晰度去适配多样化的移动终端类型，同一个视频的不同版本的视频文件之间存在一定可转换关系（高清晰度的视频能够转换成低清晰度的视频），进而给缓存性能的优化带来了新的机会。有研究[14]利用 SVC[15]设计缓存算法，将分层编码的视频按照对所有不同清晰度的请求中每个视频层的使用频率排序，并优先缓存使用频率最高的视频层（而不是整个视频）。但是，SVC 技术复杂、兼容性差[15,16](现有在线视频系统一般不支持 SVC 视频)，且能够转换的清晰度等级受限于原始 SVC 视频的视频编码的层数。已有研究[17,18]将 LRU 与视频转码[19]相结合，除非缓存视频即使转码也不能满足用户的需求，才从服务器获取文件，并执行 LRU 缓存策略，然而，这只是现有优化替换策略和视频转码的简单结合，缺乏联合视频转码设计的高性能的视频缓存算法。

另一方面，随着视频内容的极大丰富以及推荐技术的不断成熟，越来越多的用户倾向于观看系统推荐的视频[20,21]，这给缓存优化的带来了另外的机会。现有研究[22,23]利用了视频推荐技术设计的缓存，通过向用户推荐观看缓存服务器中存储的视频文件，来提高缓存性能。但是，在这些文献中，仅考虑了视频推荐技术，并未考虑联合应用视频编码，设计缓存，以进一步提高缓存性能。

综上所述，现有的缓存研究还有很大的设计和改进空间。首先，考虑到转码技术对现有视频系统的兼容性，视频转码能够被用来设计缓存，通过保证每个视频最多只有一个清晰度版本存储在缓存中，来进一步消除缓存冗余；其次，视频转码与视频推荐技术能够被联合应用来设计缓存，不但能够消除缓存冗余，还能够将存储的视频转码或/和推荐给用户，以满足用户的视频观看请求，进一步提高缓存性能。

本文研究，首先利用了转码技术设计视频缓存，从视频流行度和视频清晰度两个角度出发衡量视频的价值，并存储每个视频最有价值的一个清晰度版本来提高缓存性能；其次提出了联合视频转码和推荐的视频缓存设计，从视频流行度、视频清晰度和视频推荐三个角度衡量文件的价值，选择缓存最有价值的文件，并结合视频推荐将存储的视频直接或经过转码传送给用户观看，进一步提高缓存性能。

1.2 国内外研究现状

作为缓解网络带宽压力，提高用户体验质量的有效措施，视频缓存算法得到了广泛研究。

缓存算法通常可分成在线缓存算法和离线缓存算法两种类型。在线缓存算法

是响应式文件替换算法[24-26]，其由实时用户请求驱动，即缓存服务器在用户请求视频文件的同时做出存储决策，算法运行周期一般为秒、分、时等小尺度的时间。离线缓存算法是主动式文件放置算法，其由先验的阶段的信息驱动，即缓存服务器在用户请求视频文件之前，已经对大量的视频文件做出存储决策（基于先验知识）。算法的运行周期一般也是天、月、年等大尺度的时间范围。这两类算法能够独立执行，也可以协作执行。首先，通过利用离线算法对缓存服务器进行初始化操作，然后通过在线算法基于到达的用户请求更新缓存。在本研究中，研究的重点为离线缓存算法的设计，因此，本文从以下几个方面，重点论述了离线缓存设计的研究现状。

(1) 基于视频流行度来设计缓存。视频流行度是指视频文件的被访问次数，由文献[12,13]可知，视频点播系统中用户的视频集中请求比较集中，流行度较高的视频文件仅占全部视频文件的小部分。文献[9-11]通过存储这些少数的流行的视频文件，来满足用户的视频观看请求，提高缓存性能。一般来说，视频文件流行度的变化周期比较长，通常为一周[27]，这为流行度的预测提供了机会。文献[28-32]利用了各种措施对视频文件的流行度进行了预测，以指导缓存策略的制定。一般来说，预测的流行度较高的视频文件将被存储在缓存中。实际上，每个视频有多个分辨率的版本，而在上述文献中，每个视频版本都被看作是一个独立的个体，导致一个视频多个清晰度版本同时存储的缓存中，产生缓存冗余。

(2) 基于视频编码设计缓存。视频编码是一个能够实现视频清晰度的转换的技术，主要包括可伸缩视频编码 SVC[15]1-2 以及视频转码[19]1-2。文献[14]利用 SVC 技术将视频分层，并通过层间组合，实现了清晰度的转换，但是其应用受到 SVC 技术的限制。文献[17,18]利用了视频转码来直接将存储视频文件转码为用户请求观看的低清晰度的视频版本，提高了缓存空间的利用率。但是，在文献[17,18]中，缓存算法的本质是在线算法 LRU，并未从根本上解决同一视频多个版本同时存储的缓存冗余问题。此外，相较于 SVC，视频转码发展比较成熟，能够实时的转码视频文件，并且随着边缘计算能力的发展[33-35]以及新的转码技术[36,37]的出现，转码的开销逐渐变小，因此视频转码在缓存设计的应用更加广泛。

(3) 基于视频推荐设计缓存。随着视频内容的极大丰富，用户如何从大量的视频文件中找到自己喜欢的视频文件变得很困难，推荐技术逐渐成为主要的解决措施。最常用的推荐模型为基于协同过滤的推荐和基于机器学习的推荐，通过对大量的用户观看行为的建模分析，获得用户对视频文件的偏好信息[38]，并向用户推荐与其偏好相匹配的视频文件[39,40]。用户喜好是指用户更倾向于观看哪种类型的视频文件，具体的视频类型包括谍战，爱情，喜剧等。利用视频推荐设计的缓存策略则根据用户的偏好信息，将缓存服务器中存储的视频文件推荐给相应用户观看，以提高缓存命中率。

综上所述，在现有大多数的离线缓存设计中，同一个视频多个清晰度版本同时存储带来的缓存冗余仍然存在，深度结合视频编码技术优化缓存算法设计的研究还不多见。同时，虽然有基于视频编码和视频推荐技术的缓存算法设计，但尚未有联合这两类技术优化缓存算法的研究。这为本文的缓存优化研究工作提供了新的研究视角，此外，本文研究中，视频转码能力和代价的限制假定可被忽略，主要是考虑到 1) 边缘计算能力的发展和新的转码技术的发展[33 使其可被忽略；2) 便于缓存优化模型的建立以及缓存算法的设计。

1.3 研究内容

如前所述，针对离线缓存的研究现状，即，缺乏深度结合视频编码技术的缓存优化的研究以及没有联合视频编码和视频推荐这两类技术的缓存优化的研究，本文提出了以下两个研究：首先，利用视频转码技术设计缓存，保证每个视频最多缓存一个最有价值的版本，以进一步消除缓存冗余，提高缓存系统性能；其次，联合视频转码与推荐技术设计缓存，联合考虑缓存文件的推荐价值，消除缓存中的冗余信息，提高缓存性能。以上研究中的核心问题是，如何有效地联合利用多个维度的信息（不同视频文件的流行度和同一个视频文件的不同版本之间的可转换关系，以及不同视频之间的相似关系，识别出最有价值的视频版本，优化缓存性能。研究面对的主要挑战是，由于寻找最有价值文件的过程是一个整数线性规划问题，文件数量较多时难以直接求解，如何考虑一个文件的上述多维度信息，来刻画文件价值，以及通过怎样简化计算过程找到最优价值的文件，是一项复杂和深入思考的问题。

本文的具体研究内容如下。

(1) 基于视频转码的缓存优化设计。在深度结合视频编码技术的缓存优化的研究的方向上，考虑到视频转码的兼容性，本文提出了基于转码的缓存优化设计，利用视频转码实现清晰度的转换并保证每个视频至多只缓存一个最有价值的版本存储，以消除缓存冗余，提高缓存性能。为此，本文首先建立基于转码的缓存优化模型，将视频放置问题表示为以最小化视频传输时延为优化目标，存储空间为约束条件的整数线性规划问题；然后，提出了基于转码的缓存算法，该算法从视频流行度和清晰度这两个维度出发衡量视频版本的价值，并以贪婪迭代的方式选择存储最有价值的视频版本，直到没有能够被存储的视频版本。其中视频版本的价值被定义为假设存储该视频版本在某个缓存服务器中，能够在已缓存文件的基础上，进一步满足用户的视频观看请求（转码），所带来的缓存性能的提升。最后，设计了仿真实验，实验数值结果显示，相较于传统的基于视频流行度的缓存算法，本文提出基于转码的缓存算法能够使平均传输时延降低 40%，缓存性能更好，结果与 SVC 的

类似。

(2) 联合视频转码与推荐的缓存优化设计。本文结合视频编码和推荐两方面的技术，提出了联合视频转码与推荐的缓存优化设计，进一步消除缓存中的冗余信息，提高缓存性能。为此，本文首先建立基于转码与推荐的缓存优化模型，模型以最小化视频传输时延为优化目标，缓存空间为约束条件的整数线性规划问题；然后，提出，提出了基于转码与推荐的缓存算法，算法从视频流行度，清晰度，视频推荐三个角度出发衡量视频版本的价值，通过贪婪迭代的方式存储最有价值的视频版本，直到没有能够被存储的视频版本。其中，视频版本的价值被定义为假设存储该视频版本能够在已缓存文件的基础上，进一步满足用户的视频观看请求（转码，推荐以及联合转码与推荐），所带来的缓存性能的提升。最后，设计了仿真实验，数值结果表明本文提出的算法的缓存性能更好，尤其在缓存空间较小时，优势更加显著。

1.4 文章组织结构

本文一共五章，章节结构内容安排如下：

第一章介绍了论文的研究背景与研究意义，简要概述了现有缓存算法的研究现状，提出本论文的主要工作。

第二章介绍了论文研究过程中涉及到的技术知识以及相关研究，包括典型的视频缓存设计，视频清晰度转换技术以及视频推荐技术。

第三章介绍了基于视频转码的缓存设计，首先是系统模型，主要包括系统模型的设计动机，应用场景，面临的挑战，以及具体的问题建模；其次是算法设计，提出了基于视频转码的缓存算法，主要包括算法的基本思想，算法概述，算法实现；最后是仿真结果与性能评估，主要包括仿真实验的配置以及算法的性能分析。

第四章介绍了联合视频转码与视频推荐的缓存设计，首先是系统模型，主要包括系统模型的设计动机，应用场景，面临的挑战，以及具体的问题建模；其次是算法设计，提出了基于视频转码与视频推荐的缓存算法，包括基本思想，算法概述以及算法实现；最后是仿真结果与性能评估，主要包括仿真实验的配置以及算法的性能分析。

第五章对本文的工作研究进行的总结，并指出论文的下一步工作。

2 相关研究

本节主要从典型的视频缓存设计，视频比特率转换技术以及视频推荐技术这三个部分，详细介绍本研究相关工作的研究现状。

2.1 典型的视频缓存设计

2.1.1 缓存系统的结构组成

本文的研究基于移动网络环境下的典型的视频缓存系统，可以分成集中式缓存架构与分布式缓存架构两种主要架构，分别如图 2-1，2-2 所示。

在图 2-1 所示的集中式架构中，系统角色主要包括一个视频源缓存器，多个边缘节点（小基站），多个缓存服务器以及不同的用户等四个部分。视频源服务器部署在网络远端，具有较大的缓存能力，存储了用户请求的所有的视频文件；通过带宽有限的回程链路，视频源服务器与若干边缘节点相连。每个边缘节点一般是一个具有通信能力的小型基站，能够向一定地理区域内的用户（本地用户）提供视频通信服务，此外，每个边缘节点都配置了一个具有一定的存储能力的缓存服务器，能够存储部分视频文件。在集中式架构中，由于边缘节点是独立的，因此各个边缘节点仅能够从其附属的缓存服务器或视频源服务器来获取视频文件，以向用户提供视频服务，而不能从其他边缘节点的缓存服务器获取。集中式架构的具体工作流程如下。具体来说，当用户请求观看某个视频文件时，如果本地缓存服务器存储了用户请求的该视频文件，那么用户将通过本地边缘节点直接从附属的本地缓存服务器获取该视频文件；如果本地缓存服务器没有存储用户请求的视频文件，那么用户将不得不通过回程链路从视频源服务获取视频文件。

在图 2-2 所示的分布式架构中式架构中，架构拥有与集中式架构相同的系统角色，与集中式架构不同的是在分布式架构中，边缘节点之间存在通信链路，能够进行协作通信，因此每个边缘节点不仅可以从其附属的缓存服务器或视频源服务器获取视频文件，还能够从那些具有协作关系的边缘节点的附属缓存服务器中获取视频文件，以向用户提供视频服务。分布式架构的工作流程如下。当用户请求观看某个视频文件时，如果本地缓存服务器存储了用户请求的该视频文件，那么用户将通过本地边缘节点直接从附属的本地缓存服务器获取该视频文件；如果本地缓存服务器没有存储用户请求的视频文件，而与本地边缘节点具有协作关系的其他边

缘节点的缓存服务器存储了用户请求的视频文件，那么用户将优先从这些具有协作关系的缓存服务器中选择距离较近的缓存服务器来获取该视频文件；如果这些协作的边缘节点都没有存储该用户请求的视频文件，那么用户不得不通过回程链路从视频源服务器获取视频文件。

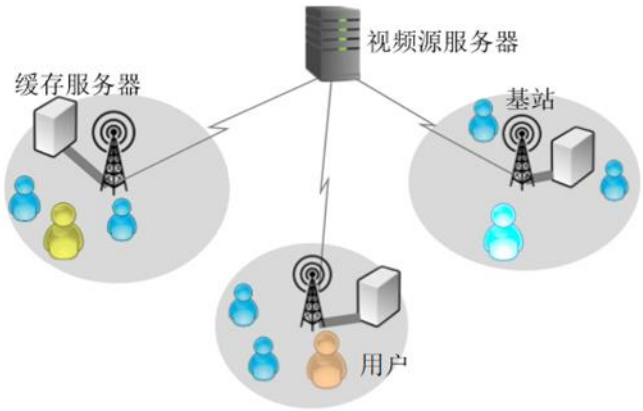


图 2-1 集中式缓存架构
Fig 2-1 Centralized caching architecture

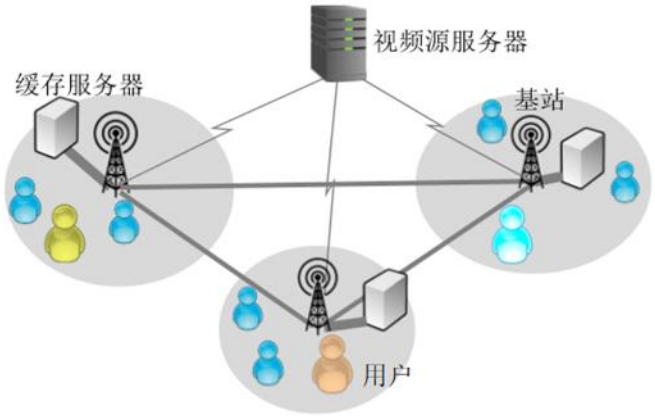


图 2-2 分布式缓存架构
Fig 2-2 Distributed caching architecture

在两个架构中，当用户从视频源服务器获取视频文件时会产生较大的视频传输时延；当用户从协作的缓存服务器获取视频文件时，产生的视频传输时延较少，而当用户从本地缓存服务器获取视频文件的传输时延可忽略不计。由此可知，无论是集中式架构或是分布式架构都能够通过在缓存服务器中存储重要的视频文件，以到达减少用户访问视频源服务器的次数，缓解回程链路的传输压力，降低用户获取视频文件的时延的目的。因此，缓存服务器中存储哪些视频文件成为决定缓存系统性能的关键因素，具体来说，如果每个缓存服务器中存储的视频都能够满足本地所有用户的视频观看请求，那么回程链路将不必传输数据，带宽压力骤减。实际上，

缓存服务器的存储空间有限，只能存储一部分视频文件，因此如何设计缓存算法来制定视频文件的存储决策成为研究热点。

2.1.2 缓存算法的概述

缓存算法决定了缓存服务器中存储哪些视频文件，性能良好的缓存算法能够有效的提高缓存服务器的性能，提高用户体验质量。本小节重点介绍了与研究内容相关的缓存算法，具体概述为以下三种类型。

基于视频流行度的缓存设计。传统上,在现有的缓存设计中，视频流行度作为衡量视频重要性的有效指标，被广泛用来设计缓存算法。文献[9]提出了一种分布式缓存算法，该算法假定每个用户与一个或多个缓存服务器直接相连并能从这些缓存服务器中直接获取存储的视频文件。利用这种用户与小基站间的一对多的连接关系，K. Shanmugam，N. Golrezaei 等人重新衡量了视频文件的重要性，并做出缓存决策，以最大程度地减少用户获取文件的延迟。文献[10]提出了一种基于无线协作通信的缓存算法，该算法假定当用户请求观看某个视频文件时，各缓存基站能够进行协作同时向用户提供视频服务。利用这个特征，Z. Ye, C. Pan 等人将流行视频文件的一个或多个副本缓存在不同基站的多个缓存服务器中，同时最小化视频传输的中断和回程链路的网络带宽使用率。文献[41]利用机器学习的各种模型，刻画各个视频文件的流行度随时间的变化趋势，并依此计算视频文件的流行度。按照计算的视频文件的流行度，周期性的对缓存内容进行更新，删除流行度较低的视频，存储流行度较高的视频文件。

在上述文献中，每个待存储的视频文件都被看作是一个独立的个体，并根据单个文件的流行度来计算文件重要性，并做出缓存决策。这种方式，可能导致同一个视频文件的多个清晰度版本同时存储在缓存中，造成缓存空间的浪费。

考虑到每个视频都需要转换为不同清晰度质量（比特率）的版本来适应用户终端的不同需求，而 SVC[15]或视频转码[19]能够实现视频文件间清晰度的转换。因此，许多研究引入了 SVC 或视频转码技术来设计缓存以提高缓存性能。

基于视频编码的缓存设计。文献[14]提出了一种基于 SVC 的缓存算法，其中 SVC 将一个视频文件编码为若干视频层，其中一些较低的视频层能够在不同清晰度的视频版本之间共享。利用 SVC 的分层特征，K. Poularakis, G. Iosifidis 等人，将视频的流行度从一个完整的视频文件角度转换为视频文件每个视频层的角度，并基于此，计算了视频层的流行度，存储了流行的视频层。然而，由于 SVC 较高的解码复杂度[15,16]，SVC 并不适用于资源受限的移动终端。此外，利用 SVC 技术的视频文件能够转换的清晰度版本的数量非常有限，其取决于 SVC 编码视频层的

数量。文献[17,18]利用视频转码技术设计了一个在线缓存系统，提出了基于视频转码的在线缓存算法。算法利用 LRU 的思想进行视频文件的缓存替换，并利用视频转码来转换已经缓存的视频文件的清晰度。具体来说，根据视频文件被访问的时间，优先存储最近一次访问时间距离当前时刻最近的视频文件，剔除距离当前时刻最远的视频文件。文献[42]评估了这种利用转码设计的基于 LRU 的在线缓存策略的缓存性能，再次证明了视频转码能够优化缓存设计，进一步提高缓存性能。文献[43]设计了受限于转码能力，存储空间等多参数的缓存系统，并对系统进行优化建模，提出了基于 LRU 的在线算法，算法基于 LRU 的思想，利用视频转码技术以及缓存服务器间的协同能力，共同提高缓存服务器的性能。但是，在这些文献中，缓存设计仅是转码技术与 LRU 的简单结合，视频文件的缓存或替换仍然基于 LRU 本身，这无法确保每个视频最多只有一个清晰度版本存储在缓存服务器中，例如，当需要缓存较高版本的视频文件时，并没有从缓存服务器中删除该视频文件清晰度较低的版本（如果有）。因此，同一视频不同清晰度版本同时存储在缓存服务器中的冗余仍然存在。

考虑到推荐技术的飞速发展，推荐的视频能够更好的匹配用户的偏好，并且用户也更倾向于观看推荐的视频[20]，因此许多研究引入了推荐技术来设计缓存，以提高缓存性能。

基于视频推荐的缓存设计。文献[22]考虑到用户通常会以较高的概率在 Youbute 推荐的视频列表上观看排名最高的视频，因此，D. K. Krishnappa，M. Zink 等人对收到的每个推荐列表进行重新排序，将已经缓存的视频文件排在列表顶部，然后将重新排序的列表发送给用户，以提高缓存性能。但是，文献[22]仅是对推荐列表进行了重新排序，而没有直接确定缓存的视频文件。参考文献[23]假设每个视频都能够被推荐给用户观看，因此每个文件的观看请求可能是用户的直接观看请求或者是系统向用户推荐该文件且用户选择观看而产生的间接观看请求。对此，P. Sermpezis，T. Spyropoulos 等人提出了一种缓存算法，计算视频文件的观看请求次数作为文件的重要性指标，并存储最重要的视频文件。

2.2 视频编码技术

由于用户的终端设备不同，网络带宽能力也不同，用户对视频文件的清晰度（比特率）的要求也越来越高，每个视频文件都需要有多个不同清晰度（比特率）的版本以适应用户这些要求。实际上，这些不同清晰度的视频版本之间可以通过可伸缩视频编解码 (SVC)，转码的技术实现相互转换。

SVC，Scalable Video Coding[44-47]，可伸缩视频编码，其制定于 2004 年，在 2007

年获得了 ITU 的批准，是一种将视频文件从时间域，空间域，质量三个维度上进行分层编码的技术。利用 SVC 编码后的视频，能够通过解码得到不同帧率，分辨率，质量等级的视频版本，以满足不同类型的用户终端。具体来说，SVC 将一个原始的视频文件，编码生成了一个小的基础层（Basic layer）和若干个多个可提高分辨率增强层，其中基础层是视频解码的基础，其自身解码后的生成的视频文件清晰度质量是最差的，而利用增强层与基础层的结合，能够提高解码视频的质量，其中,解码视频的清晰度质量越好，其需要的增强层层数也就越多。例如，利用 SVC 将一个原始清晰度质量等级为 3 的高清晰度视频文件，编码分为 3 个视频层，并用数字进行编号，其中 1 为基础层，2，3 为增强层，其中 1 本身解码生成最差清晰度质量等级为 1 的最差的视频，1，2 层组合构成清晰度为等级 2 的质量较好的视频文件。虽然，SVC 技术能够将视频文件编解码为不同帧率，分辨率，质量等级的视频文件，但是其仍存在一些缺点，进而影响了 SVC 的广泛应用。首先，SVC 的编解码复杂度比较高，代价较大，其次 SVC 的兼容性较差，仅支持具有 SVC 编解码功能的用户终端，最后，由于 SVC 编码的视频能转换生成的清晰版本的数量，完成取决于的视频被编码的层数，层数越多， SVC 的效率也就越差。

相较于 SVC 技术，视频转码已经成为一种成熟的技术，不仅能够实现视频清晰度的转换，而且具有成本较低，灵活性较好的优势[45]。因此，视频转码被广泛应用于视频多媒体系统。

视频转码技术[49-52]，就是对压缩的视频进行处理，并输出具有不同的格式，码率， 分辨率视频文件的技术。具体来说，视频转码能够将一个原始的视频文件转码为清晰度质量较低的其他视频版本，来适应各种各样的带宽以及用户终端的要求[51,52]。目前，最典型的视频转码应用框架[51]，如图 2-3 所示。

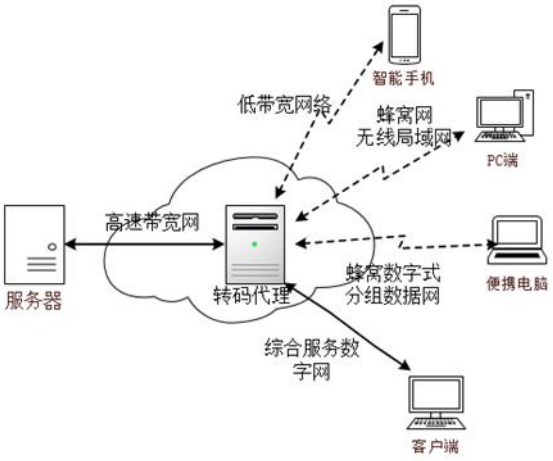


图 2-3 典型的视频转码应用框架
Fig 2-3 Typical video transcoding application framework

从图 2-3 可以看出，视频转码代理将服务器与用户终端连接了起来，这样视频文件只需在视频转码代理中进行处理，就能够适应各种网络以及用户终端的要求，为用户提供各种视频服务。由此可见，视频转码技术的兼容性较强，没有必要对服务器和用户终端的软硬件系统进行额外的修改与升级，减少了软硬件开销。但在上述框架中，每到达一次用户视频观看请求，就需要一次视频转码操作，进而占用大量计算资源，使得转码的开销变大。然而，随着边缘计算能力的发展[33]以及新的转码技术[36]的出现，转码的开销逐渐变小。

2.3 视频推荐技术

随着推荐技术的迅速发展，推荐的内容越来越能够与用户的偏好相匹配，而用户也更倾向于接受推荐的内容[20,21]1-3。因此，利用推荐技术,能够在满足用户偏好的前提，向用户推荐观看其他不同的视频文件。

推荐技术[53]是一种通过对大量信息进行分析过滤以提取有效数据，并向用户推荐内容或物品的技术，其中常用的推荐方法[54]包括基于内容的推荐算法、基于协同过滤的推荐算法以及基于模型的推荐算法。

基于内容的推荐算法是指通过比较用户的偏好与视频文件的属性之间的相似度，向用户推荐更符合用户偏好的视频文件。这种方法的原理比较简单，且不会产生冷启动问题，但是如何提取用户与视频的有效特征是非常复杂的。

基于协同过滤的推荐算法主要包括基于用户的协同过滤和基于项目的协同过滤两种类型。基于用户的协同过滤，通过计算用户之间的相似度，找到与目标用户比较相似的其他用户，并将这些用户喜欢的视频文件推荐给目标用户；基于项目的协同过滤，通过用户的行为来计算项目（视频）之间的相似度，找到与目标用户喜爱的视频文件较为相似的其他视频文件，并将这些文件推荐给目标用户。基于协同过滤的推荐算法，能够发现用户潜在的兴趣爱好，提高推荐的多样性，但是协同过滤的推荐非常依赖对用户历史数据的分析，当历史数据不足时，推荐算法应用受到限制。

基于模型的推荐算法一般是指利用机器学习算法来预测用户对一个视频文件的评分，并将评分较高视频文件推荐给用户。这种方法的关键在于机器学习算法的选择，目前常用的机器学习算法主要包括关联算法，聚类算法，分类算法，矩阵分解等。基于模型的推荐算法利用机器学习能够更加有效的从大量数据中提取有效信息，推荐效果较好，但是一般很难对推荐结果做出合理的解释。

每个推荐算法在实际应用中，各有优劣，根据研究问题的数据特征合理选择推荐算法，才能达到较好的推荐的效果，提高用户的体验质量。

2.4 本章小结

本章主要介绍了研究涉及到的视频文件清晰度转换技术，视频推荐技术以及典型缓存设计。在现有的缓存设计中仍存在一个视频的多个版本同时存储在一个缓存服务器中带来的缓存冗余问题，此外现有的缓存仅单独考虑了视频转码或视频推荐在缓存系统中的应用，并未考虑到视频转码与视频推荐的联合应用所带来的缓存性能提升。不同于已有研究，本文利用转码技术设计缓存，能够进一步降低缓存冗余，提高缓存性能。此外，本文联合应用视频转码与视频推荐技术设计缓存，能够在降低缓存冗余的前提下，利用推荐进一步满足用户请求，提高缓存性能。

3 基于转码的视频缓存设计

本章介绍了利用视频转码技术设计的基于转码的视频缓存，首先阐述了本研究的设计动机以及面临的挑战，其次结合一种可能的典型应用场景说明了缓存优化的思想并建立了优化模型，然后，提出了基于转码的视频缓存算法，并进行了仿真实验以评估算法的性能，最后对本章进行了总结。

3.1 设计动机与挑战

现有研究中大多数的缓存设计都存在着大量的缓存信息冗余，其主要原因在于，在这些缓存设计中，每个视频文件是被独立对待的，视频文件的存储也仅依赖于视频流行度这一维度，进而导致同一视频文件的多个清晰度版本被同时存储在缓存服务器中，浪费了缓存空间。为了消除缓存冗余，提高缓存性能，本研究提出了基于视频转码的缓存设计，利用视频转码技术，实现了不同清晰度视频版本之间的转换，保证每个视频文件至多一个清晰度版本存储在缓存服务器中。

实现上述基于视频转码的缓存设计，最大的挑战在于视频版本价值的衡量。由于视频不同清晰度版本间不再是独立的，而是存在着转码联系。具体来说，一个视频的高清晰版本可以通过转码技术转码为低清晰的视频版本来满足用户的视频观看请求。因此，视频版本价值的衡量将从视频流行度这个单一角度转变为从流行度，清晰度两个角度出发，这使得视频版本价值的衡量变得更加困难。

3.2 问题建模

本节首先介绍基于转码的视频缓存系统的应用场景，然后对系统进行参数化描述，建立缓存优化模型。

3.2.1 应用场景

本小节介绍了基于视频转码的缓存系统的一个典型应用场景。如图 3-1 所示，假定缓存系统有N个缓存服务器，这些缓存服务器在地理位置上分布在移动蜂窝网的边缘，且附属于各个基站，并为其附属基站覆盖下的移动用户提供在线的视频点播服务。本文假设每个缓存服务器与其附属基站间的带宽无限，（为简单起见，缓存服务器及其附属基站的编号相同），且每个缓存服务器都能够将存储的视频文件

实时的从高清晰度的视频版本转码为低清晰度版本的视频。此外，这些缓存服务器经由附属基站与远程的视频源服务器相连，该视频源服务器具有所有用户请求的视频文件的所有清晰度版本。本文假设每个用户仅被一个基站覆盖，若缓存服务器独立，用户将从其本地的缓存服务器或视频源服务器获取视频文件。如果缓存服务器间具有协作关系，用户将从其本地的缓存服务器，协作的其他缓存服务器，或视频源服务器获取视频文件。

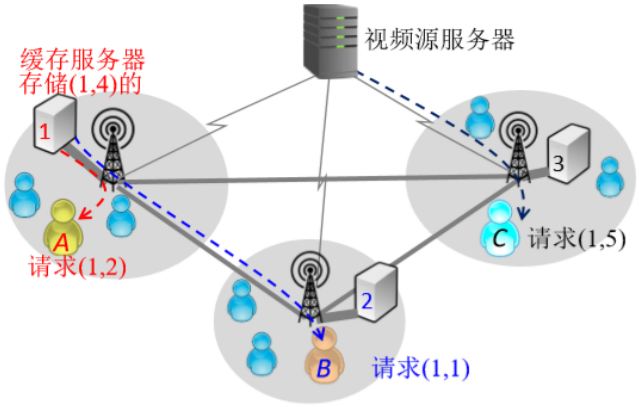


图 3-1 应用场景
Fig 3-1 Application scenario

接下来，本小节将介绍一个典型的应用实例来阐明在基于视频转码的缓存系统中用户获取视频文件的具体操作。如图 3-1 所示，用户 A 向其本地的缓存服务器（标号 1）发出对视频编号 1 清晰度等级为 2 的视频版本的观看请求，用(1,2)标记，而缓存服务器 1 存储了视频版本(1,4)。由于缓存的视频版本清晰度等级高于用户请求的视频版本清晰度等级，因此缓存服务器 1 能够将视频版本(1,4)转码为版本(1,2)，进而满足用户 A 的视频观看请求。假设用户与其本地基站之间的传输速率足够高，因此用户从本地缓存服务器获取的视频文件的传输时延可以忽略不计，进一步假设转码能够在缓存服务器上实时执行，因此本文不考虑转码造成的延迟。

此外，考虑到分布式缓存框架中，基站间可以相互协同通信，本文将进一步假设在图 3-1 中的 N=3 个基站可以相互协同通信。那么，如图 3-1 所示，由于缓存服务器 2 没有存储其本地用户 B 请求观看视频版本(1,1)，因此用户 B 不能直接从其本地缓存服务器 2 中获取视频文件。由于基站 1 与基站 2 可以协同通信，因此用户 B 能够经由这两个基站从缓存服务器 1 中获取由视频版本(1,4)转码得到的视频版本(1,1)。然而，请求观看视频版本(1,5)的用户 C 则无法被任何缓存服务器满足，那么用户 C 必须经由回程链路从远程源服务器获取其请求的视频版本(1,5)。假设协同工作的基站之间将为每个会话分配一定的带宽来进行协作的视频传输，进而导致一定的视频传输时延，而由于回程链路的带宽有限，用户从视频源获取视频文

件导致的视频传输时延较高。

本研究的目标是找到一组表示视频版本在不同缓存服务器上存储位置的缓存向量，按照向量确定视频版本的存储，能够最大程度地减少所有用户获取视频文件的平均传输延迟。

3.2.2 参数化系统

本小节将对上述基于视频转码的缓存系统进行参数化描述，以方便后续缓存优化模型的建立，表 3-1 列出了相关符号与参数。

表 3-1 符号解释
Table 3-1 Symbol description

符号	含义
N	所有缓存服务器的集合
V	所有的视频的集合
Q	一个视频所有的清晰度版本的集合
C_n	缓存服务器n的缓存空间
(v,q)	视频v，清晰度为q的版本
s_{vq}	视频版(v,q)的大小
λ_{nvq}	基站n下请求视频版本(v,q)的用户数量
d_n	从视频源服务器传输单位数据给基站n下的用户产生的单位传输时延
$d_{nn'}$	从缓存服务器n'传输单位数据给基站n下的用户产生的单位协作传输时延
X	缓存向量，其中每个元素x_{nv}为视频v存储在缓存服务器n上的清晰度等级，$x_{nv}=0$表示视频v任何版本都没有存储在缓存服务器n上
X_n	缓存服务器n的缓存向量

本研究用$N=\{1,2,\cdots,|N|\}$来表示缓存服务器的集合，其中$|N|$是缓存服务器的总数，假设每个缓存服务器n，$n\in N$具有C_n字节的空间用于缓存视频文件。假设系统中总共有$|V|$个视频，每个视频都有$|Q|$个不同的清晰度版本，用集合$V=\{1,2,\cdots,|V|\}$和$Q=\{1,2,\cdots,|Q|\}$分别表示所有的视频和所有的清晰度版本。每个视频的每个清晰度版本，用符号$(v,q),v\in V,q\in Q$表示，其大小为s_{vq}字节，其中$s_{v1}<s_{v2}<\cdots<s_{v|Q|},\forall v$。基站$n$下用户对每个视频版本$(v,q)$的请求数量用$\lambda_{nvq}$表示，其被假定为已知条件。

视频文件的传输可能会导致一定的传输时延。具体来说，将大小为s_{vq}的视频版本(v,q)从视频源服务器传输到基站n覆盖下的用户将导致大小为$s_{vq}d_n$的传输时

延，其中d_n是从视频源服务器将单位视频数据传递给基站n覆盖下的用户导致的单位传输时延。此外，在各基站能够协同工作的情况下，将大小为s_{vq}的视频版本从基站n'传输给与基站n覆盖下的用户将导致大小为$s_{vq}d_{nn'}$的传输时延，其中$d_{nn'}$是从缓存服务器n'传输单位视频数据给基站n覆盖下的用户导致的单位协作传输时延。

本文用一组整数$X=\{x_{nv}|n\epsilon N, v\epsilon V, x_{nv}\epsilon Q+\{0\}\}$来表示一个缓存向量，其中每个元素$x_{nv}$代表在缓存服务器$n$上存储的视频文件$v$的清晰度等级，其中$x_{nv}=0$表示缓存服务器$n$中没有存储视频$v$的任何清晰度版本。$X_n=\{x_{nv}|v\epsilon V\}$表示缓存服务器$n$的缓存向量。

3.2.3 建立优化模型

本小节将分基站独立工作与基站协同工作两种情况建立缓存优化模型。

(1) 基站独立工作的基于视频转码的缓存。

假设各通信基站独立工作，缓存服务器也独立运行，那么这些的缓存服务器只能独立地为用户提供视频服务。在这种情况下，用户只能从其本地缓存服务器（具有高优先级）或从视频源服务器（具有低优先级）获取视频文件。本文的目标是为每个单独的缓存服务器$n\epsilon N$找到一个最佳的缓存向量X_n，以确定每个视频版本的存储情况，同时最小化基站n覆盖范围下所有用户获取视频文件的平均传输延迟。由于平均传输延迟等于获取所有视频文件的总传输延迟除以用户请求的总数，而在本文中用户请求总数被假设为常数，因此最小化视频平均传输延迟的目的等同于最小化视频总传输延迟。视频总传输时延用符号D_n表示，其具体计算公式如 3-1 所示。

$$D_n=\sum_{v\epsilon V}\sum_{q\epsilon Q}\lambda_{nvq}s_{vq}d_nI^{x_{nv}<q} \tag{3-1}$$

其中，符号$I^{condition}$是一个指示函数，当“condition”的条件成立或不成立时，该指示函数分别等于 0 或者 1。

接下来，本文将对公式（3-1）进行具体解释。首先考虑基站n下，请求观看视频清晰度版本(v,q)的用户，如果视频v存储在缓存服务器n中的版本能够满足用户的请求，（必要时，将转码存储版本），也就是说$x_{nv}\geq q$成立，即$I^{x_{nv}<q}=0$，那么用户将能够直接从本地缓存服务器n中获取视频版本(v,q)，视频传输时延为 0；否则$x_{nv}<q$成立，$I^{x_{nv}<q}=1$，那么用户将不得不从视频源服务器获取视频文件，产生大小为$\lambda_{nvq}s_{vq}d_n$的视频传输时延，其中λ_{nvq} 是请求视频版本(v,q)的用户数量，$s_{vq}d_n$是满足每个用户请求产生的传输时延。考虑到视频v所有被用户请求的版本，则获取视频v所有版本的视频传输时延为D_{nv}，其计算公式如（3-2）所示。

$$D_{nv} = \sum_{q\epsilon Q} \lambda_{nvq} s_{vq} d_n I^{x_{nv}<q} \tag{3-2}$$

考虑到被用户请求所有视频的所有清晰度版本，则得到公式（3-1）中由D_n表示的总视频传输延迟。

最后，建立在通信基站，缓存服务器独立工作的情况下，基于视频转码的缓存优化模型，优化方程如（3-3）所示

$$\begin{aligned} &minimize_{\ X_n} \quad D_n \\ &subject\ to \quad \sum_{v\epsilon V}\sum_{q\epsilon Q} s_{vq} I^{x_{nv}=q} \le C_n \\ &\qquad\qquad x_{nv} \in Q + \{0\}, \forall v \in V \end{aligned} \tag{3-3}$$

优化模型 3-3 中有两个约束条件。第一个约束条件是缓存服务器n中存储的视频文件总大小不得超过缓存空间C_n，第二个约束条件为变量x_{nv}的值，必须是一个整数。由这些约束可知该优化模型是一个整数线性规划问题，这种问题通常是很难解决的。

(2) 基站协同工作的基于视频转码的缓存

假设各通信基站协同工作，缓存服务器协作运行，那么这些的缓存服务器可以协作为用户提供视频服务。这种情况下，用户可以从所有的服务器包括缓存服务器以及视频源服务器中选择一个能够满足用户视频观看请求的最佳服务器位置来获取观看的视频文件，以达到最小化视频传递时延的目的。本研究的目标是为所有的缓存服务器找到一个整体的缓存向量X来最小化所有用户的平均传输时延。同样这个目标等价于最小化所有用户获取视频文件产生的总传输时延D^c，其计算公式，如（3-4）所示。

$$D^c = \sum_{n\epsilon N}\sum_{v\epsilon V}\sum_{q\epsilon Q}\{\lambda_{nvq} s_{vq} d_n \prod_{n'\epsilon N} I^{x_{n'v}<q} + (1 - \prod_{n'\epsilon N} I^{x_{n'v}<q}) \lambda_{nvq} s_{vq} min_{n'\epsilon N'} d_{nn'}\} \tag{3-4}$$

其中，公式 3-4 中，集合$N' = \{n' | n'\epsilon N, I^{x_{n'v}<q} = 0\}$，其中的每个元素$n'$表示能够满足（考虑转码与协作）用户对视频版本$(v,q)$请求的缓存服务器。

接下来，本文将对公式（3-4）进行具体的解释。首先计算传输给定视频版本(v,q)来满足基站n下用户请求带来的视频传输延迟。一方面，如果任何一个缓存服务器都不能满足用户的视频请求，那么对于$\forall n' \in N$，存在$I^{x_{n'v}<q} = 1$，因此$\prod_{n'\epsilon N} I^{x_{n'v}<q} = 1$，那么用户必须从视频源服务器获取视频版本$(v,q)$，进而导致大小为导致$\lambda_{nvq} s_{vq} d_n$的视频传输时延。另一方面，如果至少存在一个缓存服务器n'可以满足用户的视频请求，那么对于每个这样的服务器n'，都有$I^{x_{n'v}<q} = 0$，因此得到了$\prod_{n'\epsilon N} I^{x_{n'v}<q} = 0$，即$1 - \prod_{n'\epsilon N} I^{x_{n'v}<q} = 1$。那么用户将从这些缓存服务器$n' \in N'$中选择具有最小单位协作传输延迟的缓存服务器来获取视频文件，进而产生大小为$\lambda_{nvq} s_{vq} min_{n'\epsilon N'} d_{nn'}$的视频传输时延。假设，缓存服务器之间协作传输延

迟小于视频源服务器到各缓存服务器之间的传输延迟。考虑视频v所有被用户请求的版本，可由公式（3-5）计算获取所有这些版本的视频传输延迟，表示为D_v^c 。

$$D_v^c = \sum_{n\epsilon N}\sum_{q\epsilon Q}\{\lambda_{nvq}s_{vq}d_n\prod_{n'\epsilon N} I^{x_{n'v}<q} + (1-\prod_{n'\epsilon N} I^{x_{n'v}<q})\lambda_{nvq}s_{vq}min_{n'\epsilon N'}d_{nn'}\} \tag{3-5}$$

考虑到所有用户请求的视频，则得到公式（3-4）中所有用户获取视频文件产生的总传输时延D^c。

最后，建立在通信基站，缓存服务器协作工作的情况下，基于视频转码的缓存优化模型，优化方程如（3-6）所示

$$\begin{aligned} &minimize_{\ X} \quad D^c \\ &subject\ to \quad \textstyle\sum_{v\epsilon V}\sum_{q\epsilon Q} s_{vq} I^{x_{nv}=q} \le C_n, \forall n \in N \\ &\qquad\qquad x_{nv} \in Q + \{0\}, \forall n \in N, \forall v \in V \end{aligned} \tag{3-6}$$

方程（3-6）中的两个限制条件与公式（3-3）中的限制条件相似。由这些约束可知，该优化模型是一个整数线性规划问题，直接求解这种问题是非常困难的，因此，下一节提出了近似性算法解决该问题。

3.3 基于转码的视频缓存算法设计

本小节介绍了基于视频转码的缓存算法（Transcoding based Caching Algorithm，TCA，首先介绍了该算法的基本思想，其次对算法进行了概述，最后设计了算法的具体实现。

3.3.1 基本思想

针对上一章建立的基于视频转码的缓存优化模型，本小节提出了基于转码的视频缓存算法，TCA，来确定每个视频不同清晰度版本的存储位置，以尽可能的最小化所有用户获取视频文件的传输时延。算法主要思想是通过贪婪的方式，迭代的逐步选择存储价值量最大的视频文件，直到缓存空间不足以存储任何视频文件或没有了待存储的视频版本。其中，视频版本的价值量被定义为假设该视频版本存储在某个缓存服务器上所带来的缓存性能的提升。

3.3.2 算法概述

TCA 通过贪婪的方式，迭代的逐步选择一个最有价值的视频文件存储在一个

可用缓存服务器上，直到缓存空间不足，生成最终缓存向量X。其中，视频文件的价值被定义为假设该视频文件存储在某个存储服务器上所带来的缓存性能的提升。

TCA 算法的具体工作原理如下。

首先，算法将缓存向量X初始化为一个元素全为零的向量，即$x_{nv}=0,\forall n,\forall v$。然后，算法将通过逐步选择某个视频版本在某个缓存服务器上来迭代地更新缓存向量X，在每次迭代过程中，仅有一个视频版本被选择存储。在X的迭代更新过程中，视频版本的价值是选择视频版本进行存储的重要依据，其被定义为假设该视频版本存储在某个缓存服务器上所带来的缓存性能的提升。由于视频版本之间存在着清晰度的联系，因此直接计算视频版本的价值是比较困难的，而本文提出的算法是迭代执行的，每次迭代中文件存储都会对剩余文件的价值造成影响，例如，在上一次迭代中，视频v的高清晰度版本已经存储了，那么视频v的低清晰度版本就没有必要存储了，其价值变为 0，因此，本文将根据算法的迭代过程，来计算视频版本的价值，作为选择视频版本存储的依据。

接下来，以一次具体的迭代过程为例，阐述视频版本价值的计算。首先，假设迭代开始之前的既有存储向量为X'，假设将视频版本(v,q)存储在缓存服务器n中，其被记作一种视频放置方案(n,v,q)，此时得到候选缓存向量$(X'+\{x_{nv}=q\})$。根据公式（3-2）或者（3-5）来分别计算在基站独立工作与基站协同工作两种情况下，根据既有缓存向量X'，或基于候选缓存向量$(X'+\{x_{nv}=q\})$，用户获取视频v的既有传输时延以及候选传输时延。既有传输时延与候选传输时延的差值为视频版本(v,q)存储在缓存服务器n中所带来的缓存收益，也就是视频版本(v,q)存储在n中的价值。在每次迭代中，由于待存储的视频版本有很多，而且这些视频版本可能被存储各个缓存服务器中，因此会产生了大量的候选缓存向量，进而得到每一种视频放置方案的价值。算法将选择能够带来最大缓存收益的视频放置方案来更新既有的X'，即将价值最大的视频文件放置方案添加到X'中来更新X'。算法将重复上述迭代更新的过程，直到没有缓存空间存储任何版本的视频文件或者没有需要存储的视频版本。

3.3.3 算法实现

根据算法的思想及概述，本小节将详细介绍基于转码的缓存算法(Transcoding based Caching Algorithm，简称 TCA)的设计，如表 3-2 所示，其受启发于文献[9]中算法的设计框架。

算法首先在 1-4 行，初始化了一系列相关变量，包括 1) 所有可能的视频放置方案构成的集合$E=\{e_{nvq}|\forall n\epsilon N,\forall q\epsilon Q,\forall q\epsilon Q\}$，其中每个元素$e_{nvq}\epsilon E$代表一种放置方案，

即将视频版本(v,q)存储在缓存服务器n中；2) 每个缓存服务器n中的视频放置方案集合$E_n \subseteq E$；3) 元素全为零的缓存向量X和 4) 每个缓存服务器n的元素全为零的缓存向量X_n。

表 3-2 基于转码的缓存算法
Table 3-2 Transcoding based caching algorithm

Algorithm 1

1：$E = \{e_{nvq} | \forall n \epsilon N, \forall v \epsilon V, \forall q \epsilon Q\}$
2：$E_n = \{e_{nvq} | \forall v \epsilon V, \forall q \epsilon Q\}, n \epsilon N\}$
3：$X \leftarrow \{0, 0, \cdots, 0\}$
4：$X_n \leftarrow \{0, 0, \cdots, 0\}, \forall n \epsilon N$
5：**for** $i = 1, 2, \ldots, |N| \times |V| \times |Q|$ **do**
6：　　$e_{\alpha\beta\gamma} = argmax_{g \in E} J_X(g)$
7：　　**if** $J_X(e_{\alpha\beta\gamma}) = 0$ **then**
8：　　　　**break**
9：　　**end if**
10：　　$C_\propto \leftarrow C_\propto - s_{\beta\gamma} + s_{\beta x_{\propto\beta}}$　　/*note: $s_{\beta x_{\propto\beta}} = 0$ if $x_{\propto\beta} = 0$*/
11：　　$x_{\propto\beta}$ *of* $X \leftarrow \gamma$
12：　　$x_{\propto\beta}$ of $X_\propto \leftarrow \gamma$
13：　　$E \leftarrow E \backslash e_{\alpha\beta q}, \forall q \leq \gamma$
14：　　$E_\alpha \leftarrow E_\alpha \backslash e_{\alpha\beta q}$，$\forall \mathrm{q} \leq \gamma$
15：　　**for** $e_{\alpha v q} \in E_\alpha$ **do**
16：　　　　**if** $C_\propto + s_{v x_{\propto v}} < s_{vq}$ **then**
17：　　　　　　$E \leftarrow E \backslash e_{\alpha v q}$
18：　　　　　　$E_\alpha \leftarrow E_\alpha \backslash e_{\alpha v q}$
19：　　　　**end if**
20：　　**end for**
21：　　**if** $|E| = 0$ **then**
22：　　　　**break**
23：　　**end if**
24：**end for**
25：Output X

紧接着，在 5-24 行，算法以贪婪的方式迭代更新缓存向量X。在每次迭代中，第 6 行从所有的可能放置位置构成的集合E中，选择了最佳的视频放置位置$e_{\alpha\beta\gamma}$，

即将视频β的清晰度为γ的版本放置在缓存服务器α上，其中函数$J_X(g)$是基于现有的缓存向量X，利用公式（3-2），（3-5）计算得到的一个视频放置方案g的价值，最佳的视频放置方案也就是视频β的清晰度为γ的版本放置在缓存服务器α上的价值最大。如果最佳的视频放置方案的价值量为 0，TRC 将打破循环，否则将继续执行下述操作。第 10 行调整了现有的缓存空间的大小，其增加了新存储的视频版本占用的空间，删除先前存储的与新增视频相同的低清晰度版本占用的空间。第 11 行和第 12 行，根据选择出来的最有价值的放置方案$e_{\alpha\beta\gamma}$，即$x_{\propto\beta}=\gamma$来分别更新X和$X_{\propto}$。第 13 和 14 行，从E和E_{α}中删除了$e_{\alpha\beta q},\forall q\le\gamma$，这些视频放置方案，其原因在于缓存服务器$\alpha$能够通过转码视频版本$(\beta,\gamma)$，来满足用户对视频版本$(\beta,q),\forall \mathrm{q}\le\gamma$的观看请求，因此视频$\beta$的清晰度不高于$\gamma$的版本将没有存储的必要。15-20 行，算法从E和E_n中删除了由于现有缓存空间较小，不足以实现的视频放置方案。如果E变为空集，那么 TCA 将打破循环。最终，TCA 将输出最终的缓存向量X。

TCA 的计算复杂度最高为$O(|N|^2|V|^2|Q|^2)$，其具体的推导如下。首先第 1-4 行中，参数的初始化需要$O(|N||V||Q|)$的时间，在第 5 行和第 24 行之间的“for”循环中，第 6 行的复杂度最高为$O(|N||V||Q|)$，整个“for”循环最多花费$O(|N||V||Q|)$的时间，因此 TCA 的计算复杂度最高为$O(|N|^2|V|^2|Q|^2)$。

3.4 实验仿真与性能评估

本小节对本文提出了 TCA 算法进行了仿真实验，以评估 TCA 的性能。本文比较了两种缓存算法，传统的贪婪算法[9]以及提出的 TCA 算法，其中 TCA 分为基站独立缓存（TCA-I）和协作式缓存（TCA-C）两个应用实例

3.4.1 实验配置

本文的仿真实验设置了 $N=3$ 个缓存服务器，其中每个缓存服务器具有相同的缓存大小。每个缓存服务器与视频源服务器之间传输视频的默认速率为 2 Mbps，在基站协同工作的情况下，协作进行视频传递的缓存服务器之间的传输视频的默认速率（即协作传输速率）设置为 5 Mbps。每个缓存服务器的存储空间都默认为 400 GB。本文假设系统有 1000 个视频文件，每个视频文件的播放时间均为一小时，且每个视频支持 5 种不同清晰度的视频分辨率，包括 240p，360p，480p，720p 和 1080p，其分别对应于 400、750、1000、2500 和 4500kbps 的视频比特率[55]。由此可计算出，5000 个视频版本的总大小接近 3900 GB。根据先验的视频点播系统研

究中的典型设置[56]，这 1000 个视频的流行度遵循 Zipf 分布，即第i个最流行的视频被请求的速率与i^{-z}成比例，其中参数z，在本实验中被默认设置为 0.8。此外，假设每个视频文件的不同清晰度版本的用户请求观看次数相同。

3.4.2 性能分析

本小节将以不同参数为变量，执行一系列的仿真实验，以分析 TCA 的缓存性能。

(1) 缓存空间对视频传输延迟的影响。本文执行了一系列以缓存空间大小为变量的仿真实验以阐述缓存空间对传输延迟的影响，实验结果如图 3-2 所示。在图 3-2 中，缓存空间的大小以 100GB 的粒度，在 50 GB 到 800 GB 变化，其中 X 轴是缓存空间的大小，Y 轴是平均传输延迟。随着缓存空间的增加，三个曲线的平均传输延迟均显示相似的下降趋势。 此外，对于每个给定的缓存大小，实验结果表明缓存算法 TCA-C 的缓存效果明显优于缓存算法 TCA-1 和缓存算法 Greedy，其中平均传输延迟分别减少了 50％和 53％。 总而言之，根据实验结果显示，TCA-C 能够有效地利用缓存空间，改善用户获取视频的体验质量。

(2) 协作传输速率对视频传输延迟的影响。本文执行了一系列以协作传输速率为变量的仿真实验，以阐述协作传输速率对传输延迟的影响，实验结果如图 3-3 所示。在图 3-3 中，协作传输速率以 1 Mbps 的粒度在 2Mbps 到 11 Mbps 之间变化，其中 X 轴是协作传输速率，Y 轴是平均视频传输延迟。图中有三条曲线，这三个曲线分别对应于 TCA-C，Greedy 和 TCA-1。前两种算法是基站协作工作的缓存，能够通过提高协作的传输速率来减少传输延迟。第三种算法是基站独立工作的缓存，因此其缓存性能恒定，与协作传输速率无关。图 3-3 的结果，清楚地表明了 TCA-C 具有最佳的缓存性能，相较于 TCA-1 和 Greedy 相比，平均传输延迟分别减少了 40％和 38%。TCA-C 算法的缓存性能更好的原因在于，该算法能够充分利用了不同缓存服务器上的转码功能以及基站间的协同工作能力。

(3) 视频流行度分布对视频传输延迟的影响。本文执行了一系列以视频流行度参数为变量的仿真实验，以阐述视频流行度分布对传输延迟的影响，实验结果如图 3-4 所示。在图 3-4 中，视频流行度参数z以 0.1 的粒度从 0.4 到 1.2 之间变化，其中 X 轴是视频流行度参数z，Y 轴是平均视频传输延迟。随着参数z的增加，图中所示的三条曲线都呈现减少趋势，其原因在于z值越大，表示受欢迎的视频就越少，通过对这些少量较流行的视频文件的存储就能够满足大部分用户的视频观看请求。此外，图 3-4 同样显示出 TCA-C 始终具有更好的缓存性能。

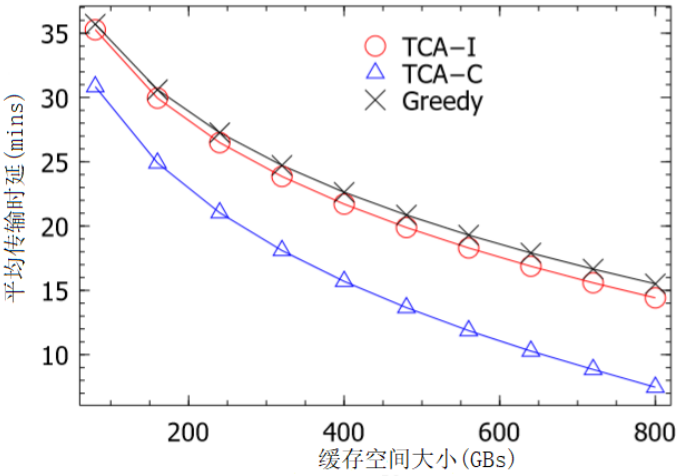


图 3-2 缓存空间的影响
Fig 3-2 Impact of cache size

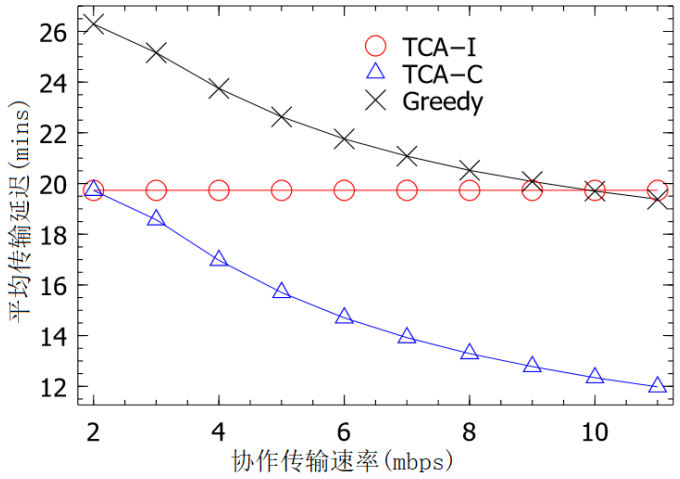


图 3-3 协作传输速率的影响
Fig 3-3 Impact of cooperative transmit rate

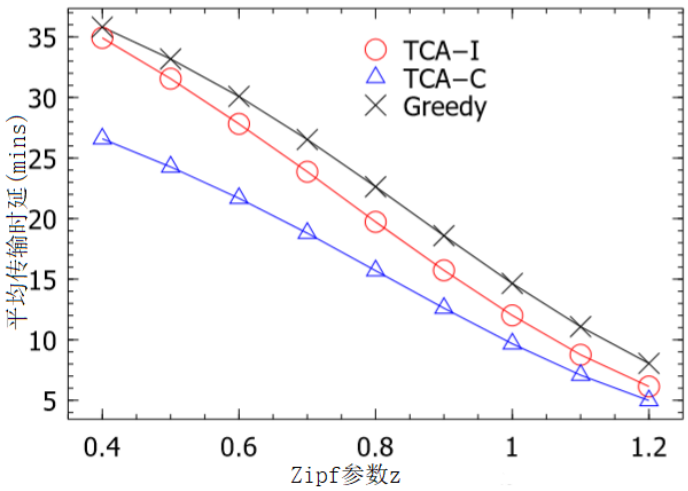


图 3-4 Zipf 参数z的影响
Fig 3-4 Impact of Zipf shape parameter

3.5 本章小结

在本章中，本文研究了基于转码的视频缓存设计，为缓存服务器提供转码能力以实现视频文件清晰度的转化。为实现该缓存设计，本研究提出了基于转码的优化模型以及基于视频转码的缓存算法，通过保证每个视频文件最多一个清晰度版本存储在缓存服务器中，来进一步降低缓存冗余，提高缓存性能。

4 联合转码与推荐的视频缓存设计

本章首先阐述设计动机以及联合转码与推荐的缓存的设计面临的挑战，其次通过一个典型应用场景阐述设计思想并建立优化模型，然后提出基于转码与推荐的缓存算法，并通过仿真实验评估算法的性能，最后对本章工作进行了总结。

4.1 设计动机及挑战

现有研究中，大多数缓存设计或是仅利用了视频转码技术，或是仅利用了视频推荐技术。实际上，联合应用转码和推荐设计视频缓存，不仅能够通过转码实现视频清晰度的转换满足用户的视频请求，而且能够通过视频推荐将存储的视频或转码后的视频推荐给用户观看，进一步满足用户视频请求。为进一步提高缓存性能，本研究提出了联合转码与推荐的缓存设计，利用转码保证一个视频至多一个清晰度版本存储在缓存服务器，消除缓存冗余，利用推荐，推荐存储的视频或转码后的视频，以满足更多用户的视频观看请求。

联合转码与推荐的缓存设计的挑战在于：1) 与现有的单一利用转码或推荐的缓存工作相比，联合应用转码与推荐使得缓存设计的操作空间更大，因此需要设计更复杂的算法来探索可能更有价值的视频缓存机会；2) 视频版本之间的关系变得更加复杂。具体来说，每个视频版本不但可以被转码或直接推荐给用户来满足用户的视频观看请求，而且还能够联合应用转码和推荐来满足用户对相似视频的不同清晰度版本的观看请求。因此，视频文件价值的衡量也将从视频流行度，清晰度以及视频推荐关系这三个角度共同出发，使其变得更加困难。

4.2 问题建模

本节首先介绍了联合转码与推荐的缓存系统的应用场景，然后对系统进行参数化描述，最后建立了缓存优化模型。

4.2.1 应用场景

首先，本小节介绍了联合转码与推荐的缓存系统的应用场景，如图 4-1 所示。该场景仅考虑了基站独立工作的情况，有关基站协同工作的情况将在后续研究中说明。

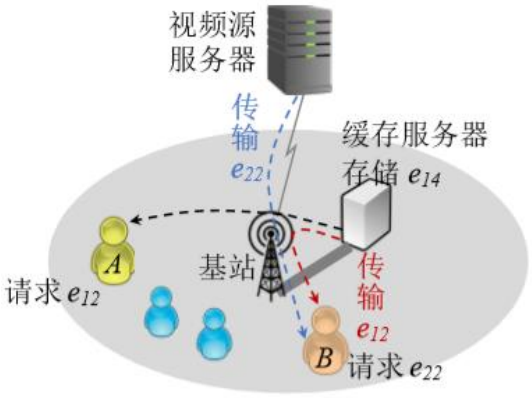


图 4-1 应用场景
Fig 4-1 Application scenario

如图 4-1 所示，该缓存系统有一个缓存服务器，其附属于覆盖多个本地用户的通信基站，该基站通过回程链路与远程的视频源服务器相连。假定基站与缓存服务器之间的带宽无限，而回程链路的带宽是有限的，仅能够为每个视频观看连接提供一定的带宽。缓存服务器具有视频转码与推荐能力，能够使用视频转码和视频推荐技术来处理其存储的视频文件，来满足本地用户的视频观看请求。其中，视频转码用于将视频的高清晰度版本转换为低质量版本，以适应用户的终端设备，视频推荐用于当用户请求的视频文件不在缓存中，向用户推荐与其请求视频最相似的其他存储的视频文件。视频转码和推荐的代价被忽略不计，并假设这些操作能够被实时执行。

接下来，本文介绍一个典型的应用场景来说明在联合视频转码与视频推荐的缓存系统中，用户获取视频文件的具体传输过程。如图 4-1 所示，首先用户 A 向缓存服务器发出了对视频版本e_{12}的观看请求，其中e_{12}表示视频编号为 1 清晰度等级为 2 的版本。由于缓存服务器已经存储了高清晰度的视频版本e_{14}，因此缓存服务器将转码存储的视频版本e_{14}得到版本e_{12}，并将其发送给用户 A，以满足用户 A 的视频观看请求。本文假设缓存服务器与用户之间的传输延迟可忽略不计。 其次，用户 B 请求观看视频版本e_{22}，但是缓存服务器没有视频 2 的任何版本，假设视频 1 和 2 足够相似，那么用户 B 也可能观看视频 1。考虑到用户 B 请求的视频版本e_{22}的清晰度等级低于缓存的视频版本e_{14}的清晰度等级，因此缓存服务器能够向用户 B 推荐由e_{14}转码得到的视频版本e_{12}，以适配用户 B 的移动终端。如果用户 B 接受了推荐的视频文件，那么缓存服务器将转码e_{14}为e_{12}并发送e_{12}以满足用户 B 的视频观看请求，过程中视频传输的延迟可以忽略不计；否则用户 B 将必须从视频源服务器获取视频版本e_{22}，从而导致一定的视频传输时延。

在本研究联合转码与推荐的缓存设计中，研究的目标是找到一组合适的视频

版本以存储在具有转码和推荐功能的缓存服务器中，来最大化减少的满足所有用户视频观看请求产生的整体视频传输时延。

4.2.2 参数化系统

在本小节中，表 4-1 列出相关的符号与参数，以对上述联合转码与推荐的缓存系统进行参数化描述，便于优化模型的建立。

表 4-1 符号解释
Table 4-1 Symbol description

符号	含义
V	所有的视频的集合
Q	一个视频所有的清晰度版本的集合
e_{ij}	视频i，清晰度为j的版本
s_{ij}	视频i，清晰度为j的版本视频版本的大小
n_{ij}	请求视频版本e_{ij}的用户数量
C	缓存服务器的缓存空间大小
X	缓存向量，其中每个元素x_i为视频i存储在缓存服务器上的清晰度等级，$x_i = 0$表示视频i的任何版本都没有存储在缓存服务器上
$p_{ii'}$	视频i和i'之间的相似性
$\propto$	判断视频是否足够相似的阈值
θ_i	视频i的所有相关视频i'的集合
$R_{ij}(X)$	视频版本e_{ij}的推荐集，其中每个元素都是缓存服务器中可以被推荐以满足用户对e_{ij}的请求的视频版本
d	从视频源服务器传输单位视频数据的产生的单位传输延迟

首先，本节介绍了描述视频文件及其缓存状态的相关符号。集合$V = \{1,2,\cdots,|V|\}$被用来表示系统中的所有视频，其中视频都按顺序编号，|V|是最大视频数。集合$Q = \{1,2,\cdots,|Q|\}$表示每个视频所有清晰度级别的顺序编号。用符号$e_{ij}, i \in V, j \in Q$表示视频i清晰度为j的视频版本，其大小为s_{ij}。对于给定的视频i的不同清晰度版本，其大小$s_{i1} < s_{i2} < \cdots < s_{i|Q|}, \forall i \in V$。此外，符号$n_{ij}$表示每个视频版本$e_{ij}$的用户请求数量。值得注意的是，为了便于优化模型的建立，每个视频$i \in V$都被添加了一个虚拟的清晰度版本 0，该版本的大小$s_{i0} = 0$，用户请求$n_{i0} = 0$。假设缓存服务器的缓存空间大小固定为C，且每个视频最多一个清晰度版本存储在缓存服务器中。缓存向量$X = (x_i \in Q + \{0\} | \forall i \in V)$表示缓存服务器中存储的视频

版本，其中$x_i = 0$或$x_i > 0$分别表示缓存服务器没有存储视频i或缓存服务器存储了视频i清晰度为x_i的版本。

其次，本节引入相关符号来描述视频之间的相似性作为视频推荐的依据。假设视频i和i'之间的相似性用符号$p_{ii'}$表示，$i, i' \in V$，其中$p_{ii'}$可以根据视频i和i'的特征（例如喜剧和科幻）计算得出。显然，如果$i' = i$，那么$p_{ii'} = 1$。此外，本节引入一个相似度阈值$\propto$来判断视频文件是否足够相似。如果视频i和i'足够相似，即$p_{ii'} > \propto$，那么视频i与i'可以相互推荐，视频i'被称为视频i的相关视频。视频i的所有相关视频的集合，用符号$\theta_i = \{i' | p_{ii'} > \propto, i' \in V\}$表示。

最后，本节介绍了视频传输过程中使用的参数与符号。假设一个用户请求了视频版本e_{ij}，对于视频版本e_{ij}，本节引入了集合$R_{ij}(X)$作为视频版本e_{ij}的推荐集，$R_{ij}(X) = \{e_{i'j'} | i' \in \theta_i, j' = x_{i'}, j' \geq j\}$，其中每个元素$e_{i'j'}$代表缓存服务器中存储的视频$i$相关视频$i'$的清晰度等级$j'$不低于版本$e_{ij}$清晰度等级的视频版本。由于集合$R_{ij}(X)$中的这些视频版本，能够被用来满足用户对视频版本$e_{ij}$的请求（必要情况下可以通过转码和推荐），因此$R_{ij}$(X)被称为视频版本$e_{ij}$的推荐集合。如果$R_{ij}(X)$不为空，那么缓存的视频可以满足用户对视频版本$e_{ij}$的观看请求，用户将从缓存服务器而非视频源服务器获得传输的视频版本。具体来说，如果集合R_{ij}(X)包含了请求视频i的版本$e_{ij'}$，$j' \geq j$，那么缓存服务器将直接发送视频版本e_{ij}发送给用户，如有必要将转码$e_{ij'}$为e_{ij}；如果$R_{ij}(X)$不包括视频i的版本，那么缓存服务器将向用户集合推荐集合$R_{ij}(X)$中的视频版本，而用户接受推荐并从集合$R_{ij}(X)$中选择一个视频i'观看的概率假设为$p_{ii'}$；如果用户不接受推荐，未从集合$R_{ij}(X)$选择视频版本观看，那么用户将从视频源服务器获取请求的视频版本e_{ij}，从而导致大小为ds_{ij}的视频传输延迟，其中d是从视频源服务器传输单位视频数据的产生的传输延迟，即单位传输延迟。在给定每个视频版本e_{ij}的用户请求数量n_{ij}的情况下，本节将建立优化模型寻找缓存向量X，确定缓存服务器中存储的视频版本，同时最小化由于用户从缓存服务器获取视频导致的总传输时延。

4.2.3 缓存优化模型

在本小节中，联合转码与推荐的缓存问题被刻画为一个最优化问题，目的是找到最终的缓存向量X，以最小化用户从缓存服务器获取视频而导致的总传输时延。具体的缓存优化过程如下。

给定一个缓存向量X，首先计算n_{ij}个用户请求视频版本$e_{ij}, i \in V, j \in Q$的视频传输时延$D_{ij}(X)$。考虑到用户只有从视频源服务器获取视频版本才会导致视频传输延迟，因此仅需要根据n_{ij}用户从视频源服务器获取视频版本e_{ij}的概率，即缓存服

务器的缓存未命中概率，来计算$D_{ij}(X)$。已知视频版本e_{ij}存在一个推荐集合$R_{ij}(X)$，而用户选择集合中的任何一个视频版本$e_{i'j'}$观看的概率为$p_{ii'}$，因此缓存未命中概率是所有视频$i' \in R_{ij}$的未观看概率$1-p_{ii'}$的连乘。根据缓存未命中概率，可计算用户获取视频版本e_{ij}的视频传输延迟$D_{ij}(X)$，如公式（4-1）所示。

$$D_{ij}(X) = n_{ij}s_{ij}d\prod_{e_{i'j'}\in R_{ij}(X)}(1-p_{ii'}) \tag{4-1}$$

接下来，考虑到所有视频的所有清晰度版本，可计算得到视频总传输延迟$D(X)$，如公式（4-2）所示。

$$D(X) = \sum_{i\in V, j\in Q} D_{ij}(X) \tag{4-2}$$

最后，建立联合转码与推荐的缓存优化模型，优化方程如（4-3）所示。

$$\begin{aligned} &minimize_{\ X} \quad D(X) \\ &subject\ to \quad \sum_{i\in V} s_{ix_i} \le C \\ &\qquad\qquad x_i \in Q+\{0\}, \forall i \in V \end{aligned} \tag{4-3}$$

为了便于缓存算法的设计，上述优化模型的目标可以等价转换为最大化由于用户从缓存服务器而非视频源服务器获取视频而导致的传输时延的减少，其原因如下。

首先，考虑到由于仅当用户从视频源服务器获取视频版本才会产生视频传输时延，而系统中用户视频请求数量一定，因此当缓存为空(X = $\mathbf{0}$ = (0，0，…，0))时，用户获取视频文件的视频传输时延一定，由公式（4-4）计算。

$$D(\mathbf{0}) = \sum_{i\in V, j\in Q} n_{ij}s_{ij}d \tag{4-4}$$

其次，考虑到缓存不为空($X \neq \mathbf{0}$)时，视频传输时延$D(X)$可由（4-2）计算。此时，由于缓存服务器存储了视频文件，这使得部分用户将从缓存服务器而不必从视频源服务器获取视频文件，进而导致了视频传输时延的减少$D(\mathbf{0})-D(X)$。最后，由于D($\mathbf{0}$)的大小一定的，因此上述优化模型的目标可以等价转化为最大化由于用户从缓存服务器而非视频源服务器获取视频而导致的视频传输时延的减少，得到新的优化模型，优化方程如（4-5）所示。

$$\begin{aligned} &maximize_{X} \quad D(\mathbf{0}) - D(X) \\ &subject\ to \quad \sum_{i\in V} s_{ix_i} \le C \\ &\qquad\qquad x_i \in Q+\{0\}, \forall i \in V \end{aligned} \tag{4-5}$$

优化模型（4-5）中有两个约束条件，第一个约束条件是缓存服务器中存储的视频文件总大小不得超过缓存空间C，第二个约束条件为变量x_i的值，必须是一个整数。由这些约束可知，该优化模型是一个整数线性规划问题，直接求解这种问题是非常困难的，因此，下一节提出了近似性算法解决该问题。

4.3 联合转码与推荐的缓存算法设计

本小节设计了一种基于转码和推荐的缓存算法(Transcoding and Recommending based Caching Algorithm，简称 TRC)来找到用于缓存视频文件的缓存向量X，旨在最大化由于用户从缓存服务器而非视频源服务器获取视频而导致的视频传输时延的减少，具体内容包括算法的基本思想，算法概述以及算法的具体实现。

4.3.1 基本思想

针对上述联合转码与推荐的缓存优化模型，本文提出了基于转码与推荐的缓存算法 TRC，以得到最终的视频缓存向量X，得到缓存服务器的缓存策略，并尽可能的最大化视频传输延迟的减少。算法主要思想是根据贪婪的思路，利用迭代的方式逐步选择在缓存服务器中存储最有价值的视频版本，以填充缓存向量X。每次迭代，算法仅选择一个最有价值的视频版本存储，直到无法再将其他视频版本放入缓存服务器为止。算法的难点在于文件价值的计算，其原因在于视频版本之间存在着清晰度以及视频推荐这两种复杂的联系，而且视频版本自身的流行度也不相同，因此视频版本价值的计算也会非常复杂。具体的计算方法，将在后续算法的具体实现中详细解释。

4.3.2 算法概述

TRC 算法通过迭代的方式，逐步存储最有价值的视频版本，以尽可能的最大化由于从缓存服务器而非视频源服务器传输视频内容而导致的传输延迟的减少，生成最终的缓存向量X。

该算法利用贪婪的思想，以迭代的方式逐步选择最有价值的视频版本存储到X中，直到缓存空间不足以存储任何版本的视频文件。在每次迭代中，算法仅选择存储一个价值量最大的视频版本，其中视频版本的价值被定义为假设存储该视频版本导致的传输时延的减少与存储该版本占用的缓存空间的比值。由于算法是迭代执行的，而视频文件之间又存在着各种联系，因此视频文件的价值会随着当前迭代过程中缓存向量X的变化而改变。具体来说，一旦某个视频版本被存储，那么该视频所有清晰度等级不高于存储版本清晰度等级的视频版本，在后续迭代中将没有存储的可能，价值量为 0，此外该存储的视频版本可以被推荐满足与存储视频文件相关的其他视频版本的用户请求，进而导致这些与存储文件相关的其他视频版本的价值量降低。综上所述，TRC 算法将动态评估每个视频版本的价值，逐步的选

择当前X下价值量最大的视频版本，以生成新的X。利用这种方法，TRC 算法可以联合应用视频转码与视频推荐的能力来辨认并存储具有最大潜在价值的视频版本，同时尽可能减少缓存视频的冗余。

4.3.3 算法实现

本小节首先介绍了联合转码与推荐的缓存算法实现过程中视频版本价值的定义及计算，然后展示了 TRC 算法的实现过程并分析了算法的复杂度。

(1) 视频版本价值的定义与计算

作为算法做出视频存储决策的重要影响因素，视频版本的价值被定义为假设存储该视频版本导致的传输时延的减少与存储该版本占用的缓存空间的比值。由于视频版本的价值会随着当前迭代过程中缓存向量X的变化而改变，因此本文将以具体的一次迭代过程为例，阐述视频版本价值的定义与计算。

首先，假设在迭代开始之前，当前缓存向量为X，基于当前X，由公式（4-2）计算用户获取视频版本的视频传输时延$D(X)$。考虑到X中的已经存储了部分视频版本，这些视频版本已经通过转码推荐满足了部分用户的请求，因此这些存储视频版本的低清晰度版本将没有存储的必要，此时剩余有存储价值的待存储的视频版本为$e_{ij}, i \in V, j \in (x_i, |Q|]$。

接下来，假设将一个待存储的视频版本$e_{ij}, i \in V, j \in (x_i, |Q|]$存储在缓存服务器中，此时得到的新的缓存向量为$X'$，其中向量$X'$中的元素相较于$X$，仅更新了$x_i = j$，其他元素不变。基于新的缓存向量$X'$，利用公式 4-2 同样假设得到用户获取视频版本的传输时延$D(X')$，此时由于存储了视频版本e_{ij}而进一步减少的视频传输时延为$D(X) - D(X')$，而存储视频版本占用的存储空间为s_{ij}，因此视频版本e_{ij}的价值u_{ij}，可由公式（4-6）计算。

$$u_{ij} = (D(X) - D(X'))/s_{ij} \tag{4-6}$$

(2) 视频版本价值简化计算

实际上，公式（4-6）可进一步简化为公式（4-7），以便于 TRC 算法迭代过程中视频版本价值的计算。具体的推导过程如下。

首先由公式（4-2）可知，视频总传输时延 $D(X)$是所有视频的所有清晰度版本的传输时延$D_{ij}(X)$的和，而$D(X) - D(X')$是由于在X中存储了视频版本e_{ij}减少的传输延迟。相较于X，X'仅改变了元素x_i的值，即X'将X中视频i的缓存x_i变为j，而其他元素并未发生改变。但是，文件e_{ij}的存储，却只能够进一步满足用户对视频版本$e_{mk}, \mathrm{m} \in \theta_i, \mathrm{k} \epsilon (x_i, j]$的观看请求，不能够进一步满足其他视频请求。因此，计算X中存储了视频版本e_{ij}减少的传输延迟$D(X) - D(X')$,实际上等同于计算基于X与X'，

用户获取$e_{mk}, \mathrm{m} \in \theta_i, \mathrm{k}\epsilon(x_i, j]$这些视频版本传输时延$D_{\mathrm{mk}}(X)$与$D_{\mathrm{mk}}(X')$的差值，而$D_{\mathrm{mk}}(X)$与$D_{mk}(X')$可由公式（4-1）计算。因此，在一次迭代中视频版本e_{ij}的价值被简化，如公式（4-7）所示。

$$
\begin{aligned}
u_{ij} &= (1/s_{ij})(\mathrm{D}(\mathrm{X}) - \mathrm{D}(X')) \qquad (4\text{-}7)\\
&= (1/s_{ij})(D_{m\in\theta_i,k\in(x_i,j)}(X) - D_{m\in\theta_i,k\in(x_i,j)}(X'))\\
&= \{\textstyle\sum_{m\in\theta_i,k\in(x_i,j)} n_{mk}s_{mk}d\prod_{e_{i'j'}\in R_{mk}(X)}(1-p_{mi'})\\
&- \textstyle\sum_{m\in\theta_i,k\in(x_i,j)} n_{mk}s_{mk}\,d\prod_{e_{i'j'}\in R_{mk}(X')}(1-p_{mi'})\}/s_{ij}\\
&= \left(\frac{\mathrm{d}}{s_{ij}}\right)\textstyle\sum_{m\in\theta_i,k\in(x_i,j)} n_{mk}s_{mk}\left\{\prod_{e_{i'j'}\in R_{mk}(X)}(1-p_{mi'}) - \prod_{e_{i'j'}\in R_{mk}(X')}(1-p_{mi'})\right\}\\
&= \left(\frac{\mathrm{d}}{s_{ij}}\right)\textstyle\sum_{m\in\theta_i,k\in(x_i,j)} n_{mk}s_{mk}\{\prod_{e_{i'j'}\in R_{mk}(X)}(1-p_{mi'})\\
&-(1-p_{mi})(\prod_{e_{i'j'}\in R_{mk}(X)}(1-p_{mi'})\})\\
&= \left(\frac{\mathrm{d}}{s_{ij}}\right)\textstyle\sum_{m\in\theta_i,k\in(x_i,j)} n_{mk}s_{mk}\,p_{mi}\left(\prod_{e_{i'j'}\in R_{mk}(X)}(1-p_{mi'})\right)\\
&= \mathrm{d}/s_{ij}\textstyle\sum_{m\in\theta_i,k\in(x_i,j)}\hat{n}_{mk}s_{mk}\,p_{mi}
\end{aligned}
$$

其中，$\hat{n}_{mk} = n_{mk}\left(\prod_{e_{i'j'}\in R_{mk}(X)}(1-p_{mi'})\right)$表示为基于$X$下，视频版本$e_{mk}, m \in \theta_i, k \in (x_i, j)$的缓存未命中请求，也就是说现有$X$中存储的视频文件不能满足用户对视频版本$e_{mk}$观看请求。

由公式（4-7）可知，为计算视频版本e_{ij}的价值u_{ij}，需要计算$\hat{n}_{mk}, m \in \theta_i, k \in (x_i, j)$的大小，而$\hat{n}_{mk}$的值与当前的缓存向量$X$密切相关。由于缓存向量$X$也是通过逐步迭代而来，而每次迭代存储的视频文件都有可能对$\hat{n}_{mk}$产生影响，因此为进一步简化$\hat{n}_{mk}$的计算，算法将根据每次迭代的结果，逐步更新所有的视频版本的缓存未命中请求$\hat{n}_{ij}$，更新过程如下。

首先当缓存为空时，所有视频版本的缓存未命中请求被初始化为$\hat{n}_{ij} = n_{ij}$，接下来，以一次具体的迭代过程为例，阐述$\hat{n}_{ij}$的迭代更新过程。假设，在一次迭代中，新的视频版本e_{ab}通过替换之前存储的低清晰度视频版本$e_{ak}, k < b$而存储在缓存服务器中。由于e_{ab}的存储，与视频a相关的视频文件的清晰度等级在$(k, b]$之间的版本$e_{ij}, i \in \theta_a, j \in (k, b]$的缓存未命中请求将进一步被$e_{ab}$以概率$p_{ia}$满足，因此这些视频版本的缓存未命中请求将缩小$(1-p_{ia})$倍，而其他视频版本的缓存未命中请求却不能被存储的视频e_{ab}的满足，因此这些版本的缓存未命中请求在此次迭代中是不变的。由此可得，在一次迭代过程中，$\hat{n}_{ij}$的更新公式，如（4-8）所示。

$$
\hat{n}_{ij} = \begin{cases} \hat{n}_{ij}(1-p_{ia}) \\ \hat{n}_{ij} \end{cases} \qquad (4\text{-}8)
$$

综上所示，在一次迭代过程中，利用公式（4-8）可计算所有视频版本的缓存未命中请求，基于得到的视频版本缓存未命中请求，可利用公式（4-7）计算待存储视频版本e_{ij}的价值u_{ij}。根据计算的视频版本价值，算法将做出存储决策。

(3) 算法的设计实现。

根据已知的视频版本价值的定义与计算，本小节将详细介绍基于转码与推荐的缓存算法(TRC)的设计，如表 4-2 所示。TRC 算法首先输入了八个必要的参数，然后通过逐步选择存储最有价值的视频版本来迭代更新缓存向量X，最后输出最终的缓存向量X。

表 4-2 基于转码与推荐的缓存算法
Table 4-2 Transcoding and recommending based caching algorithm

Algorithm 2

Input：$V, Q, S, N, P, \boldsymbol{\theta}, C, d$

Output：X

1：$X \leftarrow \{0, 0, \cdots, 0\}$

2：$E = \{e_{ij} | \forall i \epsilon j, \forall q \epsilon Q\}$

3：$\hat{n}_{ij} \leftarrow n_{ij}, \forall i \in V, \forall j \in Q$

4：**while** $|E| \neq \mathbf{0}$ **do**

5：　$e_{ab} = argmax_{e_{ij} \in E}(u_{ij}\ calc.by$ 4-7)

6：　**break** if $u_{ab} = 0$

7：　$\hat{n}_{ij} \leftarrow calc.by$(4-8), $\forall e_{ij} \in E$

8：　$\mathrm{C} \leftarrow \mathrm{C} + s_{ax_a} - s_{ab}$　/*note: $s_{ax_a} = 0$ if $x_a = 0$*/

9：　$x_a \leftarrow b$

10：　$E \leftarrow E \backslash e_{aj}, \forall j \leq b$

11：　$E \leftarrow E \backslash \{e_{ij} | s_{ij} > C + s_{ix_i}, e_{ij} \in E\}$

12：**end while**

算法输入的八个必要参数分别为：1) 系统中的所有视频构成的集合V；2) 一个视频所有的清晰度版本的集合Q；3) 表示视频版本大小的集合$S = \{s_{ij} | i \in V, j \in Q + \{0\}\}$；4) 表示用户请求数量的集合$N = \{n_{ij} | i \in V, j \in Q + \{0\}\}$；5) 视频的相似度集合$P = \{P_{ii'} | i, i' \in V\}$；6) 表示相关视频的集合$\boldsymbol{\theta} = \{\theta_i | i \in V\}$；7) 缓存服务器的缓存空间大小$C$；以及 8) 从视频源服务器传输单位数据的传输时延d。算法的具体步骤如下。

首先，算法的 1-3 行对一些变量进行了初始化。第 1 行初始化了一个元素全为零的缓存向量X；第 2 行初始化了待存储的视频版本集合$E = \{e_{ij} | \forall i \in V, \forall j \in Q\}$；第三行初始化了所有视频版本的缓存未命中请求$\hat{n}_{ij} = n_{ij}, \forall i \in V, \forall j \in Q$。

接下来，TRC 算法将执行 4-13 行的“while”循环，以迭代的方式找到最有价值的视频版本并进行存储。在每次迭代循环中，算法的第 5 行，利用公式（4-7）

计算了每个待存储视频版本的价值，并从中找到了最有价值的视频版本e_{ab}；第 6 行判断了视频版本e_{ab}是否有真正的存储意义，若e_{ab}的价值为 0，那么e_{ab}无存储意义，循环将被打破，否则将继续执行后续操作。第 7 行根据存储的视频版本e_{ab}，利用公式 4-8 更新了视频版本的缓存未命中请求$\hat{n}_{ij}$，以便于下一次迭代中视频版本价值的衡量；第 8 行计算了剩余的缓存空间大小，由于视频版本e_{ab}可以替换之前存储的视频版本e_{ax_a}，因此剩余缓存空间为$C + s_{ax_a} - s_{ab}$；第 9 行利用视频版本e_{ab}替换视频版本e_{ax_a}存储到缓存服务器中；第 10，11 更新了待存储视频版本的集合E，第 10 行从E中删除了没有存储价值的视频版本$e_{aj}, j \le b$，其原因在于这些视频版本的用户观看请求能被e_{ab}完全满足(转码)；第 11 行从E中删除了由于缓存服务器缓存空间太小而不足以存储的视频版本。TRC 算法将重复执行“while”循环，直到待存储的视频版本集合E为空集，最后输出缓存向量X。

算法 TRC 的计算复杂度最高为$O(M^3)$，其中$M = |V| \times |Q|$，其具体的推导如下。首先，待存储的视频版本数量最多为$M = |V| \times |Q|$，因此 1-3 行的初始化过程需要$O(3M)$的时间，在 4-13 行的“while”循环中，第 5 行调用了公式 4-7，其复杂度低于$O(M^2)$，而整个“while”循环最多花费$O(M)$时间，因此 TRC 算法的计算复杂度最高为$O(M^3)$。

4.4 实验仿真与性能评估

本小节对本文提出了 TRC 算法进行了仿真实验，以评估 TRC 的性能。本文比较四种缓存算法，包括本文提出的基于转码和推荐的缓存算法 TRC，基于视频推荐的算法(RCA)[23]，基于视频转码的算法(TCA)以及基于流行度的缓存算法(PBA)[9]。

4.4.1 实验配置

本小节介绍了用于仿真实验的参数设置。首先，本文假设系统共计有 1000 个视频文件，每个视频的播放时间均为一小时，并且每个视频支持 5 种不同清晰度的视频分辨率版本，包括 240p，360p，480p，720p 和 1080p，其分别对应于 400、750、1000、2500 和 4500kbps 的视频比特率[55]。由计算可知，这 5000 个视频版本的总大小约为 3900 GB。这些视频的用户观看请求数量是根据从 Movielense 数据集[57]中提取的前 1000 个最流行的视频来设置的，Movielense 数据集包含 1995 年 1 月 9 日至 2016 年 10 月 17 日的 24,404,096 个视频观看记录；此外，通过将用户对一个视频的观看请求平均分配到该视频的每个版本，来设置每个视频版本的用户观看请求。进一步，视频文件之间的相似度设置为根据这些视频文件的类型计算

出的 Jacard 相似度[58]。最后，将从视频源服务器到缓存服务器的数据传输速率设置为 5 Mbps。利用这些参数，进行仿真实验。仿真实验分别在不同的缓存空间C和不同的相似度阈值α下，以总体减少的视频传输时延为指标，评估了不同算法的缓存性能。由于系统中用户的视频观看请求数量一定，因此仿真实验的评估指标总体减少的视频传输时延可等价转换为平均减少的视频传输时延。

4.4.2 性能分析

本小节将以不同参数为变量，执行一系列的仿真实验，实验以平均减少的视频传输时延为指标，分析了 TRC 算法的缓存性能。

(1) 缓存空间对平均减少的视频传输时延的影响。

本文执行了一系列以缓存空间大小为变量的仿真实验以阐述缓存空间对平均减少的传输延迟的影响，其中相似度阈值α设为 0.8。如图 4-2 所示，缓存空间以 50 GB 的粒度从 100 GB 到 600 GB 变化，其中 X 轴和 Y 轴分别表示缓存空间的大小和平均减少的传输延迟。图中四条曲线，分别对应于 TRC，RCA，TCA 和 PBC 这四种缓存算法，均呈现出相似的趋势，即随着缓存空间的增大，平均减少的传输延迟增加。当缓存空间固定，特别是缓存空间较小时，相较于其他三种算法，本文提出的 TRC 算法的性能更好。例如，在缓存空间大小为 100GB 的情况下，与 PBC，TCA 和 RCA 相比，TRC 平均减少的传送时延分别显著提高了 740％，400％和 15％。TRC 算法的主要优势在于算法通过转码与推荐的联合力量找到并存储了高价值的视频版本。值得注意的是，由图 4-2 可知，当缓存空间逐渐增大时，本文提出的算法 TRC 与 RCA 算法，缓存性能的差距不断缩小，其原因在于随着存储视频版本的不断增多，在 TRC 算法中部分视频版本的用户观看请求将直接经由转码满足，而不必进行推荐，此时算法中，视频转码起到了主要作用，视频推荐的作用较小。而在 RCA 算法中，这些视频版本的用户观看请求时能够通过推荐满足的。因此，当缓存较大时，TCA 与 RCA 的缓存性能差距被缩小，但是 TCA 算法可转码视频版本，用户可观看请求的视频版本，因此，相较于 RCA 算法，TCA 算法中用户的体验质量更好。

(2) 相似度阈值对平均减少的视频传输时延的影响。

本文执行了一系列以相似度阈值α为变量的仿真实验以阐述相似度阈值对平均减少的视频传输延迟交付延迟的影响，其中缓存空间设为 100GB。如图 4-3 所示，相似度阈值以 0.1 的粒度从 0.3 GB 到 0.9 变化，X 轴和 Y 轴分别代表相似性阈值和平均减少的视频传输时延。图中共有四条曲线，分别对应于 TRC，RCA，TCA 和 PBC 这四种缓存算法。对于曲线 TCA 和 PBC 来说，由于两者均没有利用

视频推荐技术，因此其缓存性能与α无关，曲线始终恒定。而对于曲线 TRC 和 RCA 来说，由于两者都利用了视频推荐，因此随着相似性阈值的增大，缓存服务器中能够被推荐的视频的数量逐渐减少，进而导致不能更好的满足用户的视频观看请求，使得平均减少的视频传输延迟降低。但是，从图 4-3 中，明显可以看出，随着α的增大，TRC 的缓存性能下降速度慢于 RCA 的缓存性能下降速度，其原因在于 TRC 中不能被推荐的视频版本仍可以通过转码来满足用户请求，也就是说，视频转码带来的优势仍然存在，因此，TRC 算法始终能够保持最优的缓存性能，发挥出转码与推荐的联合力量。

(3) TRC 算法性能优势的构成成分分析。

本实验分解了 TRC 算法的平均减少的视频传输时延，以分析算法中分别由于存储，转码，推荐，联合转码与推荐而产生的性能优势。实验设置缓存空间以 50 GB 的粒度从 100 GB 到 600 GB 之间变化，相似性阈值α为 0.8。数值结果如图 4-4 所示，其中 X 轴和 Y 轴分别表示缓存空间大小和平均减少的视频传输时延。在图 4-4 中，本文将平均减少的视频传输时延分解为四个不同的独立分量，分别对应于四种视频服务类型：1) 不使用转码或推荐（蓝色填充区域），标记为“direct reducing”；2) 仅使用转码（黄色填充区域），标记为“trans reducing”；3) 仅使用推荐（红色填充区域），标记为“recom reducing”；以及 4) 联合使用转码和推荐（灰色填充区域），标记为“trans+recom reducing”。

从图 4-4 中可以看出，direct reducing 和 trans reducing 都随着缓存空间的增大而增加，这是因为较大的缓存空间不但能存储更多用户请求的视频版本，进而导致更多的 direct reducing，而且能够通过转码存储的视频版本来满足更多的用户的视频观看请求，导致更多的 trans reducing。随着缓存空间的增加，recom reducing 和 trans + recom reducing 呈现了先增加后缓慢降低的趋势。这说明，一方面，在缓存较小时（大约小于 200 GB），存储的视频版本的价值都很高（包括推荐以及推荐联合转码带来的价值）；此时，随着缓存空间的增大，每个新增的缓存文件，能够经过推荐（或联合转码）满足更多的用户请求，从而导致 recom reducing 和 trans + recom reducing 的增长。另一方面，在缓存增大到一定程度（大约大于 200 GB）之后，高价值的视频版本都被缓存完了，新增的缓存视频版本的价值越来越小（甚至趋近于 0 了）；考虑到每新一个这样的视频版本，该视频一些清晰度版本的用户请求能够直接从缓存（转码）得到满足，而不必借助视频推荐，因此，导致 recom reducing 和 trans + recom reducing 逐渐减少。

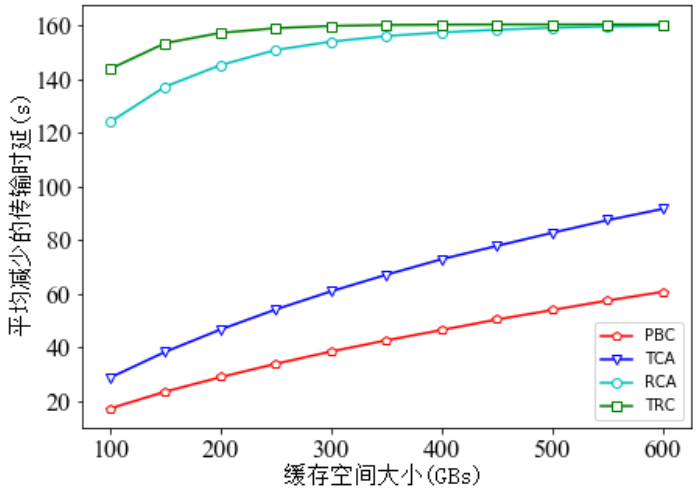


图 4-2 缓存空间的影响 ($\propto = 0.8$)
Fig 4-2 Impact of cache size ($\propto = 0.8$)

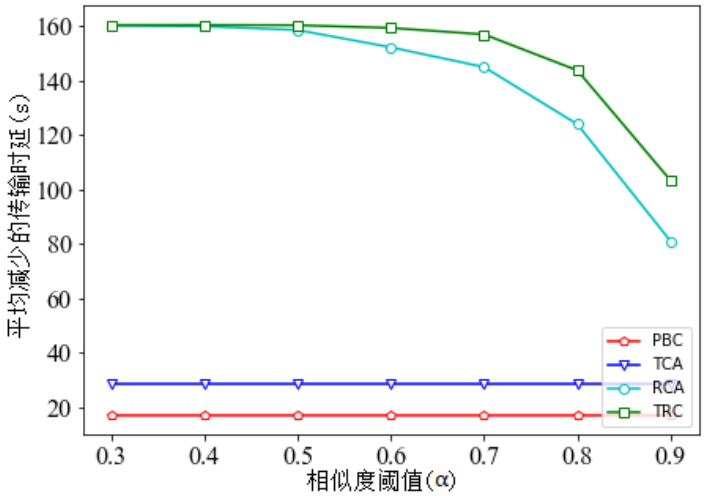


图 4-3 相似度阈值∝的影响 ($C = 100GB$)
Fig 4-3 Impact of similarity threshold $\propto$ ($C = 100GB$)

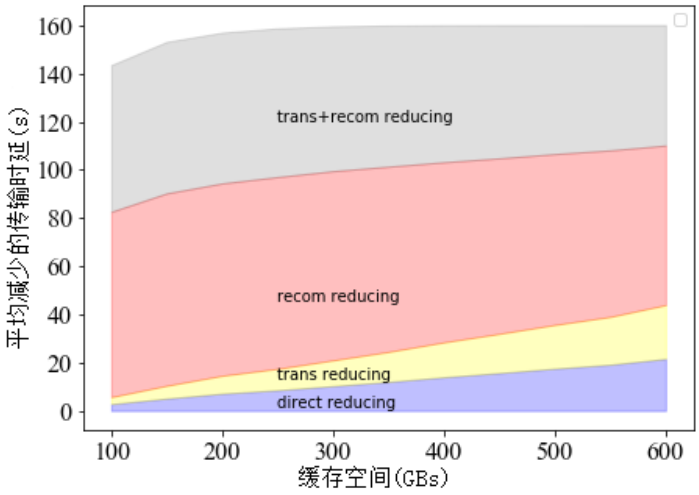


图 4-4 性能优势构成 ($\propto = 0.8$)
Fig 4-4 Performance benefits composition ($\propto = 0.8$)

4.5 本章小结

在本章中，本文研究了联合转码与推荐的视频缓存设计，其中视频转码为缓存服务器提供转码能力以实现视频文件清晰度的转化，视频推荐为缓存服务器提供推荐能力以推荐视频文件来满足用户的请求，此外联合转码与推荐能够为更多请求观看视频低清晰度版本的用户提供服务。为实现该缓存设计，本研究提出了联合转码与推荐的缓存优化模型以及基于转码与推荐的缓存算法，通过视频转码技术保证每个视频文件最多一个清晰度版本存储在缓存服务器中，来进一步降低缓存冗余，通过视频推荐技术推荐存储的或转码后的视频文件，以进一步满足用户的视频观看请求，提高缓存性能。

5 结论

5.1 本文总结

本文主要包括两项研究内容。

一项是基于转码的视频缓存优化研究。通过利用视频转码来研究视频文件的存储，以提高分布式缓存系统的缓存性能。首先，本文区分基站独立工作以及基站协同工作两种情况建立了基于转码的缓存优化模型，将视频缓存问题表示为以缓存空间为约束条件，以最小化视频传输时延为目标的整数线性规划问题。其次，本文提出了基于转码的缓存算法 TCA，该算法将以迭代的方式，逐步选择实现最有价值的视频存储方案，获得最大的视频传输延迟增益，尽可能最小化视频传输时延。最后，本文进行了仿真实验，实验结果表明，与传统的贪婪算法相比，TCA 算法的缓存性能最好，能够显著减少视频传输时延。

另一项是联合转码与推荐的缓存优化研究。通过联合利用视频转码和视频推荐技术来研究视频文件的存储，以提高缓存性能。首先，本文建立了联合转码与推荐的缓存优化模型，将视频缓存问题表示为以缓存空间为约束条件，以最大化减少的视频传输时延为目标的整数线性规划问题。其次，本研究提出 TRC 缓存算法，该 TRC 算法以贪婪的方式运行，逐步迭代的选择存储最有价值的视频版本，每次迭代，仅存储一个视频版本，直到缓存空间不足以存储任何版本。最后，本文进行了仿真实验，实验数值结果表明 TRC 算法的缓存性能更好。

5.2 工作展望

随着视频流量的爆炸式增长，移动网络带宽的压力也越来越大，用户观看视频的体验质量也逐渐变低。边缘缓存设计作为解决上述问题的有效措施，得到了广泛研究。本文利用视频转码技术以及视频推荐技术设计了视频缓存方案，以进一步优化缓存性能。但是由于作者的能力和时间有限，论文仍存在许多不足，仍需要进一步的探索，具体包括以下几点：

一、在联合转码与推荐的缓存设计中，本文仅考虑了单基站工作的情况，没有考虑基站协同工作情况下的缓存优化问题，本文的后续工作将继续研究联合转码与推荐的分布式缓存设计。

二、本文忽略了视频转码与视频推荐的成本，但是在实际应用中，视频转码与

推荐需要占用一定的计算资源以及时间代价，因此，本文的后续工作将进一步考虑转码与推荐的代价优化缓存设计。

参考文献

[1] Cisco. Cisco Visual Networking Index. Global mobile data traffic forecast update, 2017-2022[J]. White Paper, 2019.

[2] ETSI. Mobile-edge computing introductory technical white paper. White Paper, 2014.

[3] ETSI. Mobile edge computing-A key technology towards 5G. White Paper, 2015.

[4] Ahmed A, Ahmed E. A survey on mobile edge computing[C]// 10th IEEE International Conference on Intelligent System and Control, (ISCO 2016). IEEE, 2016.

[5] Jun L, Chu S, Feng S, et al. Contract-Based Small-Cell caching for data disseminations in ultra-dense cellular networks[J]. IEEE Transactions on Mobile Computing, 2019, 18(5): 1042-1053.

[6] Doan K, Van T, Quek T, et al. Content-Aware proactive caching for backhaul offloading in cellular network[J]. IEEE Transactions on Wireless Communications, 2018: 1-1.

[7] Zheng T X, Wang H M, Yuan J. Secure and energy-efficient transmissions in cache-enabled heterogeneous cellular Networks: Performance analysis and optimization[J]. IEEE Transactions on Communications, 2018, 66(11): 5554-5567.

[8] Li L, Zhao G, Blum R S. A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies[J]. IEEE Communications Surveys and Tutorials, 2018, 20(3): 1710-1732.

[9] Golrezaei N, Shanmugam K, Dimakis A G, Molisch A F, Caire G. FemtoCaching: Wireless video content delivery through distributed caching helpers[C]// IEEE International Conference on Computer Communications (INFOCOM) 2012. Orlando, FL, USA, 2012: 1107-1115.

[10] Ye Z, Pan C, Zhu H, Wang J. Outage probability and fronthaul usage tradeoff caching strategy in Cloud-RAN[C]// IEEE International Conference on Communications. Paris, France, 2017: 1-6.

[11] Ahlehagh H, Dey S. Video-Aware scheduling and caching in the radio access network[J]. IEEE/ACM Transactions on Networking, 2014, 22(5): 1444-1462.

[12] Breslau L, Cue P, Fan L, Phillips G, Shenker S. Web caching and Zipf-like distributions: Evidence and implications[J]. INFOCOM, 1999: 126-134.

[13] Wang Y, Chen Y, Dai H, et al. A learning-based approach for proactive caching in wireless communication networks[C]// International Conference on Wireless Communications and Signal Processing. IEEE, 2017.

[14] Poularakis K, Iosifidis G, Argyriou A, et al. Caching and operator cooperation policies for layered video content delivery[C]// IEEE International Conference on Computer Communications (INFOCOM). San Francisco, CA, 2016.

[15] Schwarz H, Marpe D, Wiegand T. Overview of the scalable video coding extension of the H.264/AVC standard[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2007, 17(9): 1103-1120.

[16] Zhang G. Computational complexity optimization on H.264 scalable/multiview video

coding[D]. 2014.

[17] Shen B, Lee S J, Basu S. Caching strategies in transcoding-enabled proxy Systems for streaming media distribution networks[J]. IEEE Transactions on Multimedia, 2004, 6(2): 375-386.

[18] Shen B, Roy S. A very fast video special resolution reduction transcoder[C]// IEEE International Conference on Acoustics, Speech and Signal Processing. Orlando, FL, USA, 2012: 1989-1992.

[19] Vetro A, Christopoulos C, Sun H. Video transcoding architectures and techniques: An overview[J]. IEEE Signal Processing Magazine, 2003, 20(2): 18-29.

[20] Zhou R, Khemmarat S, Gao L. The impact of YouTube recommendation system on video views[C]// ACM SIGCOMM Internet Measurement Conference. Melbourne, Australia, 2010: 404-410.

[21] Khemmarat S, Zhou R, Krishnappa D K, et al. Watching user generated videos with prefetching[J]. Signal Processing Image Communication, 2012, 27(4): 343-359.

[22] Krishnappa D K, Zink M, Griwodz C, Halvorsen P. Cache centric video recommendation: An approach to improve the efficiency of Youtube caches[J]. ACM Transactions on Multimedia Computing Communications and Applications, 2015, 11(4).

[23] Sermpezis P, Spyropoulos T, Vigneri L, Giannakas T. Femto-Caching with soft cache hits: Improving performance through recommendation and delivery of related content[J]. IEEE Journal on Selected Areas in Communications, 2017: 99-112.

[24] Mattson R L, Gecsei J, Slutz D R, Traiger I L. Evaluation techniques for storage hierarchies[J]. IBM Systems Journal, 1970, 9(2): 78-117.

[25] Aho A V, Denning P J, Ullman J D. Principles of optimal page replacement[J]. Journal of the ACM (JACM), 1971, 18(1): 80-93.

[26] Ma T H, Tian W, Wang B. Weather data sharing system: An agent-based distributed data management[J]. IET software, 2011, 5(1): 21-31.

[27] Golrezaei N, Molisch A F, Dimakis A G. Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution[J]. IEEE Communications Magazine, 2013, 51(4): 142-149.

[28] Su B, Wang Y, Liu Y. A new popularity prediction model based on lifetime forecast of online vidoes[C]// 2016 IEEE International Conference on Internet Infrastructure and Digital Content (IC-NIDC) IEEE. 2016: 376-380.

[29] Fontanini G, Bertini M, Del Bimbo A. Web video popularity prediction using sentiment and content visual features[C]// 2016 ACM on International Conference on Multimedia Retrieval. 2016: 289-292

[30] Li C, Liu J, Ouyang S. Characterizing and predicting the popularity of online videos[J]. IEEE Access, 2016, 4: 1630-1641.

[31] Wang S, Zhang X, Zhang, Y. A survey on mobile edge networks: Convergence of computing, caching and communications[J]. IEEE Access, 2017, 5: 1-1.

[32] 朱敏, 李俊. 视频点播中视频流行度的建模与分析[J]. 电子技术, 2016, 45(9): 40-43.

[33] Albanese A, Crosta P S, Meani C, Paglierani P. GPU-accelerated video transcoding unit for multi-access edge computing scenarios[C]// The Sixteenth International Conference on

Networks (ICN 2017). Venice, Italy, 2017: 143-147

[34] Dutta S, Taleb T, Frangoudis P A, Ksentini A. On-the-fly QoE-aware transcoding in the mobile edge[C]// 2016 IEEE Global Communications Conference(GLOBECOM 2016). Washington, DC, USA, 2016.

[35] Wang Z, Sun L, Wu C, Zhu W. Joint online transcoding and geo-distributed delivery for dynamic adaptive streaming[C]// IEEE INFOCPM 2014-IEEE Conference on Computer Communications. Toronto, Canada, 2014: 91-99.

[36] Fung K T, Siu W C. Low complexity H.263 to H.264 video transcoding using motion vector decomposition[C]// 2005 IEEE International Symposium on Circuits and Systems (ISCAS 2005). Kobe, Japan, 2005: 908-911.

[37] Diaz-Honrubia A J, Cebrian-Marquez G, Martinez J L, et al. Low-complexity heterogeneous architecture for H.264/HEVC video transcoding[J]. Journal of Real Time Image Processing, 2016, 12(2): 311-327.

[38] Hyun D, Park C, Yang M C, et al. Review sentiment-guided scalable deep recommender system[C]// The 41st International ACM SIGIR Conference on Research and Development Information Retrieval, ACM. 2018:965-968

[39] Chang AIDen, Liao J F, Chang P C, Teng C H, Chen M. Application of artificial immune systems combines collaborative filtering in movie recommendation system[C]// IEEE 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD). 2014: 277-282.

[40] 龙超. 基于混合推荐技术的推荐系统设计与实现[D]. 电子科技大学,2017.

[41] 芮兰兰, 彭昊, 黄豪球, 邱雪松, 史瑞昌. 基于内容流行度和节点中心度匹配的信息中心网络缓存策略[J]. 电子与信息学报, 2016, 38(2): 325-331.

[42] Elayoubi S E, Roberts J. Performance evaluation of video transcoding and caching solutions in mobile networks[C]// 2015 27th International Teletraffic Congress (ITC 27). Ghent, Belgium, 2015: 55-63.

[43] Gao G, Zhang W, Wen Y, Wang Z, Zhu W. Towards cost-efficient video transcoding in media cloud: Insights learned from user viewing patterns[J] IEEE Transactions on Multimedia, 2015, 17(8): 1286-1296.

[44] ITU-T ACEG and ISO/IEC MPEG Joint Video Team (JVT), ITU-T REec.H.264|ISO/IEC 14496-10 AVC standard. 2007.

[45] Wiegand T, Sullivan G J, Bjontegaard G. Overview of the H.264/AVC video coding standard[J]. Circuits and Systems for Video Technology IEEE Transactions, 2013, 13(7): 560-576.

[46] Wien M, Schwarz H, Oelbaum T. Performance analysis of SVC [J]. IEEE Transactions on Circuits and Systems for Video Technology, 2007, 17(9): 1194-1203.

[47] 毕厚杰. 新一代视频压缩编码标准: H.264 / AVC[M]. 人民邮电出版社, 2006: 415-435.

[48] 李皓琰, 郭蕾. 视频转码研究综述[J]. 现代商贸工业, 2010, 22(8): 280-281.

[49] Keesman G, Hellinghuizen R, Hoeksema F, Heideman G. Transcoding of MPEG bitstreams[J]. Signal Processing: Image Communication, 1996, 8(6).

[50] Acharya S, Smith B. Compressed domain transcoding of MPEG[C]// IEEE International Conference on Multimedia Computing and Systems. Austin, TX, 1998: 295-304.

[51] Han R, Bhagwat P, Lamaire R, Mummert T, Perret V, Rubas J. Dynamic adaptation in an image transcoding proxy for mobile web browsing[J]. IEEE Personal Communications, 1998, 5(6): 8-17.

[52] Youn J, Sun M T, Xin J. Video transcoder architectures for bit rate scaling of H.263 bit stream [C]// The 7th ACM International Conference on Multimedia '99. Orlando, FL, USA,1999: 243-250.

[53] Shi Y. An improved collaborative filtering recommendation method based on timestamp[C]// International Conference on Advanced Communication Technology (ICACT). IEEE, 2014: 784-788.

[54] 吴涛. 推荐系统中推荐算法研究及其应用[D]. 北京交通大学, 2019.

[55] YouTube Live Encoder Settings, https://support.google.com/youtube/answer/2853702?hl=en. Last accessed Apr. 26, 2020.

[56] Hefeeda M, Saleh O. Traffic modeling and proportional partial caching for peer-to-peer systems[J]. IEEE/ACM Transactions on Networking, 2008, 16(6): 1447-1460.

[57] https://grouplens.org/datasets/movielens/latest/. Last accessed Jan. 09, 2020.

[58] Jaccard P. The distribution of the Flora in the alpine zone[J]. New Phytologist, 1912, 11(2): 37-50.

作者简历及攻读硕士学位期间取得的研究成果

一、作者简历

赵红娜，女，1992 年 8 月出生，2013.09 至 2017.07 就读于西安邮电大学通信与信息工程学院，获学士学位。2017.09 至 2020.07 就读于北京交通大学电子信息工程工程学院，获硕士学位。研究生就读期间主要研究方向为信息网络和视频存储传输。

二、发表论文

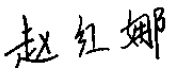
[1] Hongna Z, Chunxi L, Yongxiang Z, et al. Transcoding based video caching systems: Model and algorithm[J]. Wireless Communications and Mobile Computing, 2018, 2018:1-8 (SCI).

三、参与科研项目

[1]国家自然科学基金面上项目，61872031，基于熵理论的信息匹配网络测量与建模，2019/01-2022/12，70 万，参加.

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作和取得的研究成果，除了文中特别加以标注和致谢之处外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京交通大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

学位论文作者签名：  　　签字日期：2020 年 5 月 25 日

学位论文数据集

表 1.1： 数据集页

关键词*	密级*	中图分类号	UDC	论文资助
边缘缓存，视频编码，视频转码，视频推荐	公开			
学位授予单位名称*		学位授予单位代码*	学位类别*	学位级别*
北京交通大学		10004	工程	硕士
论文题名*		并列题名		论文语种*
基于视频转码和视频推荐的缓存设计				中文
作者姓名*	赵红娜		学号*	17120196
培养单位名称*		培养单位代码*	培养单位地址	邮编
北京交通大学		10004	北京市海淀区西直门外上园村 3 号	100044
学科专业*		研究方向*	学制*	学位授予年*
信息安全		信息网络	3	2020
论文提交日期*				
导师姓名*	李纯喜		职称*	副教授
评阅人	答辩委员会主席*		答辩委员会成员	
	郭宇春		赵永祥，陈一帅，郑宏云，张立军	
电子版论文提交格式 文本（ ） 图像（ ） 视频（ ） 音频（ ） 多媒体（ ） 其他（ ） 推荐格式：application/msword；application/pdf				
电子版论文出版（发布）者		电子版论文出版（发布）地		权限声明
论文总页数*	47			
共 33 项，其中带*为必填数据，为 21 项。				