硕士学位论文

业务特征感知的 CDN 网络资源调度的研究

Service Feature-aware Resource Scheduling in CDN Network

作者：崔子琦

导师：赵永祥

北京交通大学

2021 年 5 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：　　导师签名：

签字日期：　　年　　月　　日　　签字日期：　　年　　月　　日

学校代码：10004 密级：公开

北京交通大学

硕士学位论文

业务特征感知的 CDN 网络资源调度的研究

Service Feature-aware Resource Scheduling in CDN Network

作者姓名：崔子琦 学　　号：18120047

导师姓名：赵永祥 职　　称：副教授

学位类别：工学 学位级别：硕士

学科专业：通信与信息系统 研究方向：信息网络

北京交通大学

2021 年 5 月

致谢

本论文的工作是在我的导师赵永祥老师的悉心指导下完成的。赵老师治学严谨，思想开阔，学术知识渊博，为人风趣幽默，在日常科研生活中以轻松幽默的方法启迪我如何发现问题，思考问题，以及解决问题，传授了我基本的科研思维，锻炼了我的批判思维，启迪了我的创新思维，使我具备了一个研究生基本的科研素养。除了在学习中孜孜不倦的教导，赵老师也在日常生活中如家长般的关心我，在我迷茫时给指引，在我难过时给我关怀，在我受挫时给我鼓舞。在此我由衷感谢赵老师在三年中对我的指导与关怀。

同样感谢李纯喜老师，在论文选题和论文撰写过程中给予我巨大的帮助，尤其是在面向人工智能应用的缓存相关研究中，不仅启迪我思考，而且在学术期刊撰写的过程中帮我整理思路，在论文投稿的过程中不厌其烦地帮我修改每一句话，提醒我注意投稿的各种事项，纠正我的错误，李老师的指导使我对科研有了更深层次的认知。

感谢郑宏云老师，在每周一次的学术例会中启发我思考，帮助我解决科研中遇到的问题；感谢郭宇春老师、陈一帅老师、张立军老师、孙强老师，对我的研究方向提供了宝贵的意见和对我毕业设计的指点。同时也感谢各位老师为我提供了良好的科研环境，我所有的科研成果都凝结着各位老师的汗水。

此外，感谢我的师兄宋云鹏，在科研过程中与我不断探讨，在生活中给予我巨大的帮助，在遇到困难时为我加油打气，帮助我不断前进；感谢我的同门孙欢，和我一起为竞赛而努力，在闲暇时一起交流生活体验；感谢我的舍友郝爽雨、熊佳慧三年来陪我走过的日日夜夜，让我在宿舍感受家庭的温馨；感谢张虎信、戚余航、李想、王珍珠等所有同门，感谢你们陪我度过这段难忘的学习生涯。

特别感谢我的父母，对我二十多年的辛勤养育和悉心教诲，对我无私的付出和无条件支持，让我能够心无旁骛地完成学业，并支持我进一步深造学习，追逐自己的理想。感谢我的男朋友李阳对我的无限包容和爱护，陪伴我成长，给我无限的精神支持，让我变成更好的自己。

最后还要诚挚的感谢国家自然科学基金（No. 61872031 基于熵理论的信息匹配网络测量与建模）的资助。

摘要

内容分发网络（content delivery network, CDN）是互联网数据分发的重要架构。CDN 将用户所需要的文件部署到距用户网络距离较近的边缘缓存服务器上，用户即可直接从边缘缓存服务器下载所需文件，从而减轻骨干网的流量压力，降低用户下载文件的时延。

CDN 的主要业务场景包括两大类。第一类是目前主流的视频文件的分发，第二类是为新兴的利用深度神经网络（deep neural network, DNN）模型的人工智能应用提供计算。CDN 不仅提供文件分发服务，也提供计算服务，CDN 的业务逐渐由传统的文件分发向文件分发和计算一体化发展。

随着 CDN 业务量的快速增长，CDN 性能的提升面临着资源瓶颈。CDN 的资源瓶颈主要来自于边缘缓存服务器向用户分发文件的带宽资源与缓存资源。本文将根据 CDN 不同的业务对资源的需求，设计相应的资源调度机制来提高 CDN 的资源利用效率，提升用户体验。本文的具体工作如下：

(1) 提出了一个视频内容感知的 CDN 负载均衡联合优化模型，将不同视频请求定向到不同的 CDN 边缘缓存服务器并为每个请求规划不同的视频比特率，以在有限的服务器带宽约束下最大化用户观看视频的总体用户体验。该联合优化模型背后的依据是，一个视频画面的用户体验是画面比特率的非线性增函数，且不同画面的体验函数的参数不同，进而，CDN 分发视频时，可以通过规划每个边缘缓存服务器提供的视频以及每个视频的每个画面的比特率，来联合优化用户总体体验。本文设计了一个贪婪的启发式算法求解最优化模型，该算法首先将所有边缘缓存服务器当作一个虚拟的理想服务器，求得每个视频的最优比特率；然后按求得的最优比特率，将视频迭代地分配到每个边缘缓存服务器。实验结果表明所提出的负载均衡算法能够比现有算法提升 3%-70%的用户体验。

(2) 提出了一个面向人工智能应用的缓存优化模型，选择人工智能应用并缓存其用到的 DNN 模型到边缘缓存服务器有限的 GPU 内存中，以最小化人工智能应用的整体响应时间。该优化模型背后的依据是，一个人工智能应用可能并行用到多个 DNN 模型来实现其最终功能，一个 DNN 模型可能被多个人工智能应用使用，只有当一个应用所需的全部 DNN 模型缓存在 GPU 内存中时，应用才能获得较短的响应时间，进而，利用人工智能应用与 DNN 模型之间的关联关系缓存 DNN 模型可以降低人工智能应用的整体响应时间。本文设计了一个贪婪启发式算法求解最优化模型，该算法迭代地选择人工智能应用，并缓存应用关联的 DNN 模型。实验结果表明提出的缓存算法在多数情况下可提升缓存性能 10%左右。

关键词：内容分发网络；资源调度；视频分发；人工智能应用的计算；用户体验

ABSTRACT

Content delivery network (CDN) is an important architecture for Internet data distribution. The CDN deploys the files requested by users to the edge cache server that is close to users so that the user can directly download the required files from the edge cache server. As a result, the CDN reduces the traffic pressure on the backbone network and reduces the time delay for users to download files.

The main service scenarios of CDN include two categories. The first is the current video file distribution, and the second is to provide executions for emerging artificial intelligence applications that use deep neural network (DNN) models. CDN not only provides file distribution service, but also provides computing service. The service of CDN has gradually evolved from traditional file distribution to integrated file distribution and computing.

With the growth of CDN service volume, the improvement of CDN performance is facing resource bottlenecks, which mainly come from the cache resource and the bandwidth resource that distributes files to users. This thesis will design corresponding resource scheduling mechanisms to improve the resource utilization of CDN and the quality of experience of users according to the resource requirements of different CDN services. The main contribution of this thesis is as follows.

(1) This thesis proposes a joint optimization model of content-aware CDN load balancing mechanisms, which directs different video requests to different edge cache servers and decides the video bitrates for each request, to maximize the overall user experience under the constraint of server bandwidth resource. The basis behind the joint optimization model is that the user's experience of a video is a non-linear increasing function of bitrate, and the parameters of the user experience function are different for different videos. Therefore, when the CDN distributes videos, the overall user experience can be improved by jointly planning the direction decision of each edge cache server and the bitrate of each video. This thesis then designs a greedy heuristic algorithm to solve the optimization model. The algorithm first treats all edge cache servers as a virtual ideal server to obtain the optimal bitrate of each video. Then the video is iteratively assigned to an edge cache server according to the optimal bitrate. Experimental results show that the proposed load balancing algorithm can improve user experience by 3%-70% compared with the existing algorithms.

(2) This thesis proposes a cache optimization model for artificial intelligence applications, which selects and caches the DNN model used by artificial intelligence applications into the limited GPU memory of the edge cache server to minimize the average response time of applications. The basis behind this optimization model is that an artificial intelligence application may concurrently use multiple DNN models to achieve its final function, and a DNN model may be used by multiple artificial intelligence applications, only when all the DNN models required by an application are cached in the GPU, the application can obtain short response time. Therefore, using the association relationship between the artificial intelligence applications and the DNN models to cache can reduce the overall response time. This thesis then designs a greedy heuristic algorithm, which iteratively selects applications and caches the associated DNN models in the GPU memory. Experimental results show that the proposed caching algorithm can improve the cache performance by about 10% in most cases.

KEYWORDS： Content Delivery Network; Resource allocation; Video distribution; Execution of artificial intelligence application; Quality of experience

目录

1 引言

1.1 研究背景与意义

内容分发网络（content delivery network, CDN）是互联网数据流量的分发的重要架构。据调研显示，目前互联网 70%以上的数据通过 CDN 分发[1]，CDN 承担了互联网中绝大部分的视频、语音、网页、计算等业务。

CDN 能够有效降低主干网的流量压力和为用户提供更低的传输时延[2]。CDN 利用互联网上多数用户请求的文件是相同的这个特点，使用缓存技术将用户需要的文件副本缓存到距离用户较近的网络边缘的缓存服务器上，这样带来如下两方面的性能提升：一方面，CDN 只需从源服务器请求并缓存一次文件，便可满足其服务范围内所有用户对该文件的请求，减少了骨干网的流量压力；另一方面，由于边缘缓存服务器距离用户的网络距离更近，从而可以提供更低的传输时延。

CDN 的业务场景主要可以分为两大类。第一类是当前的视频业务。根据思科发布的互联网报告[3]，从 2017 年起，全球视频 IP 流量的复合年均增长率约为 29%，到 2022 年，视频流量将增长 4 倍，占据网络整体流量的 82%以上。第二类是为使用深度神经网络（deep neural network, DNN）模型的人工智能应用提供计算业务。随着人工智能应用的普及，人工智能应用的计算也会成为 CDN 的重要业务，即 CDN 不仅仅提供文件分发服务，也提供 DNN 模型的缓存和计算服务[4]，CDN 的功能将从简单的内容缓存分发进化到内容缓存分发和计算服务一体化。

在 CDN 庞大的业务需求和复杂的业务场景下，根据不同业务的特点优化用户体验具有理论和实际意义。针对以上主要的两类业务场景，本文根据其具体的业务特征对用户体验的影响，分别提出了相应的 CDN 资源调度优化机制来优化用户体验。一方面，为满足以视频为代表的传统 CDN 业务需求，在不同的边缘缓存服务器之间调度用户请求，并为每个请求分配合适的带宽资源，以最大化总体用户体验。另一方面，为满足以人工智能应用的计算为代表的新兴 CDN 业务的需求，提出面向人工智能应用的 CDN 缓存算法，以最小化应用的响应时间。

1.2 研究现状

本节阐述了 CDN 视频业务和 CDN 人工智能的计算业务的研究现状。

1.2.1 CDN 视频业务的研究现状

(1) CDN 视频业务面临的资源瓶颈

在传统的视频业务中，CDN 的运营成本或者说资源消耗主要来自于如下两个方面[2]：

1) 缓存成本。传统的 CDN 需要缓存并分发用户所需视频文件，边缘缓存服务器的缓存空间越大，可以预缓存的内容越多，这样可以直接满足更多用户的视频请求，减少用户向网络远端的源服务器请求视频的情况，从而降低视频的传输时延。但是，相对于用户多样化的需求，由于成本的约束，缓存空间通常是有限的，边缘缓存服务器不可能存储用户所需的所有文件。

2) 上载带宽成本。CDN 在距离用户较近的地理位置部署边缘缓存服务器，并通过向网络运营商租用带宽来向用户分发视频。租用的带宽越多，能够为用户提供的平均传输速率越高。但是，更高的带宽意味着更高的租用成本，带宽的租用费用通常是 CDN 运营商的主要成本[5]。

(2) CDN 视频业务资源调度的相关研究

针对上述资源瓶颈，目前学术界和工业界的一个研究热点是如何通过缓存和带宽资源调度，为用户提供尽可能高的视频体验。

1) 缓存资源调度。按照缓存调度的机制，CDN 的缓存调度算法可以分为离线缓存算法和在线缓存算法[6]。离线缓存算法主动预先存储用户可能请求的视频，以满足未来一段时间内的用户请求，算法运行周期一般较长。在线缓存算法则根据用户当前请求，实时决定是否缓存用户所请求的视频。

2) 带宽资源调度。CDN 带宽资源调度可以分为单服务器上的带宽资源调度和多个服务器之间的带宽资源调度。单服务器上的带宽资源调度通过为一个服务器上不同的请求分配合适的带宽，以提高该服务器上用户观看视频的体验 (quality of experience, QoE)。多个服务器之间的带宽资源调度通过在多个服务器之间调度用户请求以及为不同的用户分配合适的带宽，从而防止单个服务器过载，提高整体用户体验，多服务器之间的带宽资源调度通常也称为负载均衡。

根据带宽分配过程中使用的信息，负载均衡可以进一步分为静态负载均衡和动态负载均衡两大类[7]。静态负载均衡按照一个预定的策略把用户请求定向到特定的服务器，如随机负载均衡[8]、基于 IP 的哈希[9]等。动态负载均衡根据服务器实际运行过程中的负载状态动态地定向用户请求。本文研究的是多个服务器之间的动态负载均衡。

具有代表性的动态负载均衡方面的研究包括文献[10-12]。其中文献[10]提出的最小负载算法将传入的用户请求定向到当前服务用户数目最少的服务器，类似的

研究还有文献[11,12]。这些算法考虑的因素较为单一，没有考虑各个服务器带宽容量的异质性、用户距离服务器的网络距离和物理距离等因素的影响。近年来，动态负载均衡相关的研究考虑的因素逐渐多样化，文献[13]利用排队论提出了基于服务器负载量的负载均衡算法；文献[14]将用户请求定向的问题建模为一个顺序决策问题，提出了基于用户 QoE 的负载均衡算法；文献[15]结合模型预测控制技术提出了基于时延和用户带宽满意度的负载均衡算法。

以上这些研究将网络传输质量、视频时延、视频清晰度等定义为用户 QoE 进行带宽资源调度以提升用户体验。根据文献[16,17]，实际上用户体验还和视频内容的特点有关。基于这个特点，本文提出视频内容感知的负载均衡算法，根据不同的视频画面的用户体验与画面比特率的非线性关系，将用户请求定向到合适的服务器，并为每个请求分配合适的比特率，从而提高总体用户体验。

1.2.2 CDN 人工智能应用的计算业务的研究现状

人工智能应用的计算是 CDN 的新兴业务。随着人工智能应用的不断普及，用户对实时的人工智能应用的计算需求也随之增长。主流的人工智能应用为了获得较高的性能，通常使用经过预训练的大型 DNN 模型。但由于用户终端的计算和存储能力不足以支持高性能 DNN 模型的计算，而将计算任务卸载到云端又会引入较大的时延开销，最近的研究将已经训练好的 DNN 模型缓存到 CDN 边缘服务器的 GPU 内存中实现人工智能应用的实时推理和计算[4]，从而缩短人工智能应用的响应时间。

(1) CDN 人工智能应用的计算业务面临的资源瓶颈

CDN 人工智能应用的计算业务面临的主要资源瓶颈是 CDN 边缘缓存服务器 GPU 内存资源的不足[4,18,19]。CDN 边缘缓存服务器使用 GPU 执行人工智能应用关联的 DNN 模型的计算来实现应用的功能，从而满足用户需求。在这个过程中，现有研究表明，把大型 DNN 加载到 GPU 内存中的时延远远大于执行 DNN 模型计算的时延[4,20]。一个解决方法是尽可能地把 DNN 模型预先加载到 GPU 中，避免因从硬盘加载或将请求定向到云端导致的巨大时延开销。显然，GPU 内存容量越大缓存资源越充足，边缘缓存服务器就可以同时加载更多的 DNN 模型，从而直接满足更多应用的计算需求。但是，执行 DNN 模型的 GPU 内存的容量受到成本或制作工艺的限制，因此 GPU 内存的容量与多样化的 DNN 模型的需求相比总是有限的。

(2) CDN 人工智能的计算业务资源调度的相关研究

针对边缘缓存服务器 GPU 内存的资源瓶颈，现有研究主要从以下两个方面提供了缓存资源调度的思路，以提高边缘服务器的内存利用率，为用户提供更高质量

的体验。

1) 沿用传统的缓存算法。这种方法认为 DNN 模型是一种数据文件，因此一定程度上可以沿用传统的基于流行度的缓存算法，通过统计预测未来一段时间内用户对不同人工智能应用的偏好程度，决定缓存哪些 DNN 模型，相关的缓存机制有[21-23]。文献[4]提出人工智能应用的计算业务有着区别于传统文件分发业务的特点，如对计算精度、时延的要求等。因此，可以在传统的缓存算法上进行一定的微调和改进，以进一步提升缓存性能，但目前实际的相关研究还处于起步阶段。

2) 根据 DNN 模型的特点定制缓存算法。文献[20]给出了如下研究方向：根据可用内存空间的大小，动态地改变 DNN 模型的尺寸[24,25]；将每个 DNN 模型的部分数据预先加载到缓存中，同时进行模型计算和模型加载来降低响应时间；将一段时间内需要相同 DNN 模型的计算请求调度成一批，对于同一批次的请求只需加载一次 DNN 模型，同时还可以令功能相似的请求使用相同的 DNN 模型来增加批的规模；通过预测 DNN 模型的内存需求、计算精度、流行度等参数，主动加载替换 GPU 内存中的 DNN 模型；通过不同边缘缓存服务器之间的协作来提升缓存性能。

上述的两类研究方法均将 DNN 模型视为独立的数据文件进行缓存，实际上人工智能应用与 DNN 模型之间可能存在着多对多的关联关系，即一个人工智能应用可能并行用到多个 DNN 模型来实现最终功能，同时一个 DNN 模型也可能被多个人工智能应用使用。而 DNN 模型与人工智能应用之间的多对多关系会进一步影响应用的响应时间，只有当一个应用需要的全部 DNN 模型都加载到缓存中时，应用才可获得较低响应时间。本文根据上述特点，提出面向人工智能应用的缓存算法，在缓存空间有限的条件下，选择合适的人工智能应用并缓存应用关联的 DNN 模型，以尽可能降低应用响应时间，提升用户体验。

1.3 论文工作与组织架构

本文分别针对 CDN 的两种主流业务提出了相对应的资源调度方案。本文的具体工作如下：

(1) 提出了视频内容感知的负载均衡算法。CDN 视频业务的资源瓶颈是 CDN 边缘缓存服务器向用户分发文件的上载带宽资源。现有的研究表明视频的画面内容也会影响用户体验，不同的视频画面的用户体验对比特率变化的敏感程度不同。基于上述研究，本文提出了一种多缓存服务器之间的视频内容感知的负载均衡联合优化模型，在 CDN 边缘缓存服务器带宽容量有限的情况下，合理地将用户请求分配到不同的服务器上，同时为每个视频请求分配合适的比特率，以最大化整体用户的视频观看体验。进一步，本文提出了一种贪婪启发式算法求解该联合优化模型。

该算法首先将所有 CDN 服务器当作一个理想服务器，以所有服务器带宽总和为约束，迭代地求得每个视频的最优比特率，以最大化理想服务器提供视频服务的总体用户体验；然后按求得的最优比特率，将视频迭代地分配到每个 CDN 边缘缓存服务器，每次选择价值最高的视频分配到剩余带宽最多的服务器，其中视频价值定义为视频的用户体验除以该视频的比特率。实验结果显示，本文提出的启发式算法能够在带宽资源有限的情况下，有效提升总体用户视频体验。

(2) 提出了面向人工智能应用的缓存算法。CDN 人工智能的计算业务的资源瓶颈是 CDN 边缘缓存服务器的 GPU 内存资源。现有的研究指出可以沿用传统的缓存算法来缓存人工智能应用所使用的 DNN 模型，或根据 DNN 模型的特点定制缓存算法。本文分析了人工智能应用与 DNN 模型之间的关联关系和及其对应用响应时间的影响，即一个人工智能应用可能并行用到多个 DNN 模型来实现其最终功能，一个 DNN 模型可能被多个应用使用；只有当一个应用所需的全部 DNN 模型缓存在 GPU 内存中时，应用才能获得较短的响应时间。本文提出了一个面向人工智能应用的缓存优化模型，选择人工智能应用并将其所用到的 DNN 模型缓存到边缘缓存服务器有限的 GPU 内存中，以最小化人工智能应用总体的响应时间。在此基础上，本文设计了一种贪婪启发式算法，该算法通过贪婪迭代的方式选择最有价值人工智能应用并存其关联的 DNN 模型，每个模型只缓存一个副本，其中人工智能应用的价值定义为该应用的流行度除以满足该应用的计算当前所需要的缓存空间的大小。数值结果表明本文提出的缓存算法可以有效提升缓存性能。

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，简要概述了 CDN 业务面临的资源瓶颈及相应的资源调度的研究的现状，提出了本文的主要工作。

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，包括经典的 CDN 视频架构、视频业务特征及相关资源调度技术、CDN 人工智能应用的计算业务的背景及相关资源调度研究。

第三章介绍了视频内容感知的 CDN 负载均衡算法设计，首先介绍了研究的背景，然后具体阐述了研究场景和基本思想，接着提出了视频内容感知的 CDN 负载均衡算法，主要包括系统数学描述，最优化模型建立和算法实现，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。

第四章介绍了面向人工智能应用的缓存算法设计，首先介绍了研究背景，其次介绍了研究的应用场景和基本思想，随后提出了面向人工智能应用的缓存算法，包括系统的数学描述，问题的最优化建模及算法实现，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，并指出了论文的下一步工作。

2 相关研究

本章介绍经典的 CDN 视频架构，视频业务资源调度相关研究，以及新兴的人工智能应用的计算业务资源调度相关研究。

本章的结构安排如下：2.1 节介绍经典的 CDN 视频架构；2.2 节介绍 CDN 视频业务的资源调度研究；2.3 节介绍 CDN 人工智能应用的计算资源调度相关研究；2.4 节介绍了常用的启发式算法；2.5 节为本章小结。

2.1 经典的 CDN 视频架构

经典的 CDN 视频架构如图 2-1 所示。在这个图中，CDN 架构主要由视频源服务器、边缘缓存服务器和用户三部分组成。其中，视频源服务器部署在距离用户较远的网络远端，存储所有用户可能请求的视频文件；边缘缓存服务器距离用户较近，通过租用的带宽向用户分发视频，具有有限的带宽容量和缓存空间，能够缓存一定数目的视频文件；用户分布在不同的地理位置。当用户发起视频请求时，CDN 会把用户的请求定向到距离用户较近的边缘缓存服务器。在经典的 CDN 视频架构中，主要的资源瓶颈来自于边缘缓存服务器有限的缓存空间和向用户分发视频的上载带宽。

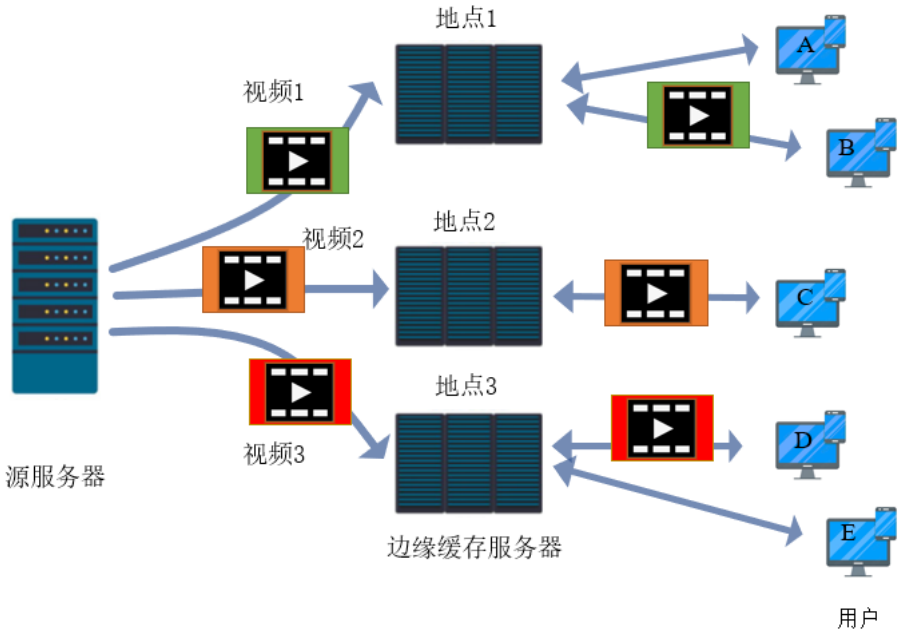


图 2-1 经典的 CDN 视频架构

Figure 2-1 The typical CDN video architecture

CDN 分发视频的过程如下。以图 2-1 为例，假设当前位于地点 3 的用户 E 请求视频 3，CDN 定向机制按照距离将用户 E 的请求定向到地点 3 的边缘缓存服务器，若此时该边缘缓存服务器中存储了视频 3，则直接将视频 3 分发给用户 E。若此时该边缘缓存服务器中没有存储视频 3，边缘缓存服务器需向源服务器请求该视频文件的副本分发给用户 E，并将副本缓存在服务器本地。当下时刻同处于地点 3 的用户 D 也请求视频 3，则地点 3 的边缘缓存服务器可直接将已缓存的视频 3 分发给用户 D，从而以较低的时延满足用户 D 的请求。

2.2 CDN 视频业务的资源调度的研究现状

本节首先介绍了视频业务的特点，然后分别介绍了目前 CDN 网络中视频业务的带宽资源和缓存资源调度的研究现状。

2.2.1 视频业务的特点

现有的研究发现，用户的体验主要受到网络传输质量以及视频参数和特性的影响。网络传输参数主要是指网络传输路径的质量参数，如带宽及其波动、时延等。视频参数和特性具体指的是视频的采集压缩编码参数和视频画面内容的时空复杂度。

在网络传输质量方面，网络传输带宽及其波动会影响用户体验。传输带宽短时间内的减少将导致视频卡顿甚至中断，严重影响用户体验。为减少这种影响，一种解决方法是在用户端设置初始缓冲区[26,27]，当用户启动视频播放时，用户端设备首先缓冲一定时间长度的内容（一般为 3 秒以上的视频长度），这样可以平滑视频播放过程中传输带宽的波动，减轻视频播放时缓存冲区为空时导致的卡顿现象。但是这种方法会引入一段播放启动时延，启动时延过长也会影响用户体验。减少带宽波动影响的另一种方法是自适应视频编码[28]，该方法根据网络的带宽可用性选择最合适的编码比特率，但是这种方法容易导致视频播放质量的频繁切换，降低用户体验。

在视频参数方面，视频采集及压缩编码的参数会影响用户体验。视频本身参数主要包括：分辨率，一般情况下视频分辨率的提高意味着视频画面质量的提升[29]；对比度，对比度差异范围越大代表对比越强，适当的对比度可以提升画面质量[30]；帧率，高的帧率可以实现更流畅、更逼真的画面，明显提升视频交互感和逼真感[31]；比特率，通常视频的比特率越高，每秒传送的画面数据就越多，视频的画质也就越清晰[32]。

在视频特性方面，视频画面内容会影响用户体验。主要内容为：画面内容不同的视频帧，时间复杂度（temporal information, SI）和空间复杂度（spatial information, TI）一般是不同的；要获得相同的用户体验，TI 和 SI 较高的视频帧往往需要更高的比特率来传输[17]；一个视频帧画面的用户体验是画面比特率的非线性增函数，对于 TI 和 SI 不同的视频帧，视频体验函数的取值也有所不同[16]。

文献[16]采用结构相似性指数（structural similarity , SSIM）作为用户体验评价指标。相较于峰值信噪比和均方误差这样的同类评估指标，SSIM 从亮度、对比度、结构三个方面进行归一化评价用户体验，更符合人类视觉系统，与用户主观评价的相关度更高，且更易理解[33]。SSIM 的计算表达式如公式(2-1)所示。

$$\mathrm{SSIM}(x,y)=\frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_2)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \tag{2-1}$$

其中，x 和 y 分别代表两个样本帧，x 为原始样本帧，y 为被比较的样本帧，μ_x 和 σ_x^2 分别为 x 的均值和方差，μ_y 和为 σ_y^2 分别为 y 的均值和方差，σ_{xy} 为 x 和 y 的协方差。SSIM 的取值在 0 到 1 之间，SSIM 越接近于 1 代表被比较的视频帧与原始的样本帧差别越小，因此失真也越小，用户体验越好；反之，用户体验越差。

2.2.2 视频业务资源调度的研究

现有 CDN 视频业务资源调度的研究，主要的方向是通过充分利用边缘服务器有限的带宽和缓存资源来提高用户体验。

在缓存资源方面，现有的研究尽可能地利用有限的本地缓存来满足多样化的用户视频请求。具体的实现手段包括通过预测视频文件的流行度变化规律提前存储相应的视频文件[34]，通过推荐系统引导用户观看已经存储的视频文件[35]、采用视频转码功能去除存储视频文件的冗余信息等[36]。

由于本文的一个贡献是关于 CDN 带宽资源的调度算法，本节将详细介绍 CDN 带宽资源调度研究。在带宽资源调度方面，相关研究的分类如图 2-2 所示，CDN 的带宽资源调度可分为单服务器带宽资源调度和多服务器带宽资源调度。多服务器之间的带宽资源调度也称为负载均衡，负载均衡又可进一步分为静态负载均衡和动态负载均衡。其中，在动态负载均衡的研究中，根据资源调度过程中使用的指标，相关研究又可分为基于服务器负载的负载均衡、基于用户 QoE 的负载均衡、基于多指标的负载均衡等。

单服务器上的带宽资源调度通过在一个边缘缓存服务器中为不同的视频请求分配不同的带宽，以提高该边缘缓存服务器上的用户体验。文献[37]提出了一种基于强化学习的任务感知的视频传输架构，该架构利用 DNN 模型提取用户感兴趣的

视频画面，并对其动态解压缩，然后根据当前网络带宽限制，利用强化学习网络动态调整所传输的视频的带宽，以尽可能降低重传率，提高带宽利用率。文献[16]提出了一种实时的流媒体文件上传优化方案，根据不同视频用户体验对比特率变化的敏感程度不同，选择性地优化部分已上传的视频编码，并重传该部分内容，平衡传输当前视频和重传视频的带宽资源分配，以最大化整体用户体验。

多服务器之间的带宽资源调度通过在多个边缘缓存服务器之间调度用户请求及带宽资源，以提高整体用户体验。具体来说，由于 CDN 网络中用户的请求分布不均、服务器性能参差不齐、网络节点的异构性等原因，网络中可能存在一些边缘缓存服务器上的请求数目过多，导致服务器负载过重，甚至瘫痪。多服务器之间的带宽资源调度对网络中整体的带宽资源进行统一调配，将用户的请求合理定向到不同的边缘缓存服务器，并为不同的请求分配合适的带宽，使得各个服务器承担的任务量与服务器资源相匹配，防止单个服务器运行过载，实现整个系统高效平稳运行。

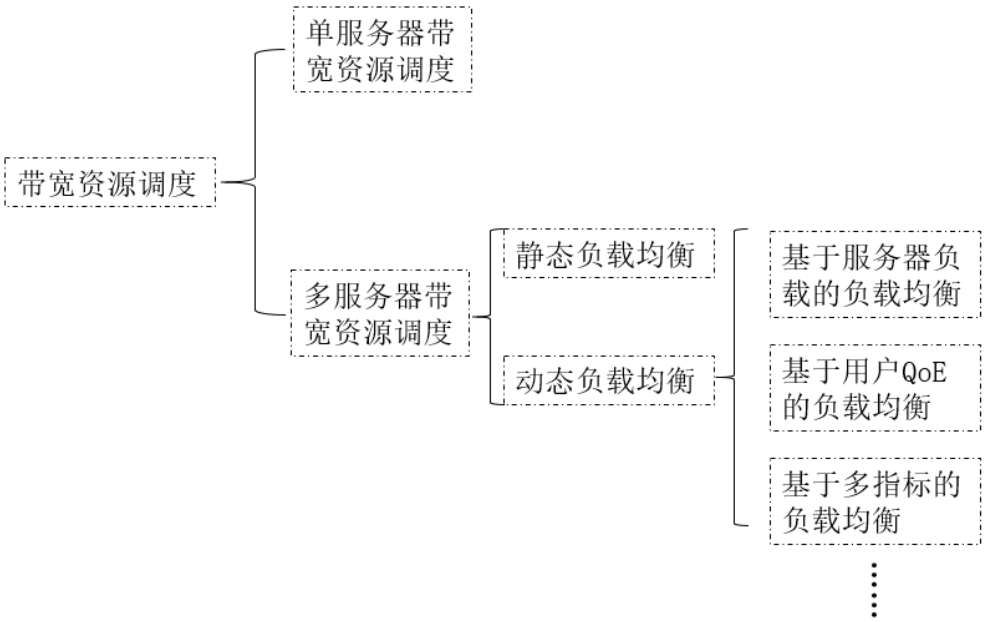


图 2-2 CDN 视频业务带宽资源调度研究分类
Figure 2-2 Research about bandwidth resource scheduling for CDN video service

根据均衡过程中所使用的信息，负载均衡技术可分为静态负载均衡和动态负载均衡[7]两大类。静态负载均衡根据设备及程序的运行状态和系统的先验知识预先设定均衡规则进行负载的分配，典型的静态负载均衡算法有轮询[38]、加权轮询[39]、基于 IP 的哈希算法[9]等，这种方法没有考虑到当前服务器与用户实际的状态。动态负载均衡不依赖先验知识，而是根据收集到的服务器及用户的信息来进行负载

的分配。与静态负载均衡相比，动态负载均衡能够及时获取系统的信息，通过动态的调度策略更好地均衡整体的负载，提高系统的整体性能。

在视频业务的CDN动态负载均衡的研究中，根据均衡过程中使用的信息不同，动态负载均衡又可分为基于服务器负载的均衡技术、基于用户视频 QoE 的均衡技术和基于多指标的负载均衡技术等。

基于服务器负载的均衡技术根据服务器当前的负载状态定向用户请求。文献[40]发现流媒体平台 Twitch 首先为视频流请求指定一个负载较少的服务器，然后随着该视频流请求的人数的增加，将该视频文件复制到不同区域的多个服务器上，并将用户的请求定向到这些不同区域的服务器，以防止单个服务器过载导致用户体验急剧下降。文献[41]提出了拥塞避免的负载均衡算法，联合 CDN 内容放置与用户请求定向建立最优化模型，将每个服务器收到的请求建模为一个队列，在网络稳定性的约束下，使用随机优化模型最小化最长传输时延，并利用李雅普诺夫优化模型将长期最优化问题分解为多个短期的线性规划问题，以实现多个服务器之间的负载均衡。文献[42]分析了不同业务（视频、网页）下的 CDN 资源利用情况，利用服务器的 CPU 利用率、发送包的带宽利用率、链路数和磁盘总 I/O 操作等参数，描述每个服务器当前负载状态。在此基础上来改进传统的最小连接调度算法，为用户选择当前负载最轻的边缘缓存服务器。

基于用户视频 QoE 的负载均衡技术综合考虑不同的服务器状态变量，如服务器的负载、网络链路的质量指标、视频卡顿、卡顿时延、清晰度等因素建立 QoE 的衡量指标，在此基础上定向用户请求。文献[43]提出的 P2P-CDN 架构通过为用户选择最近的服务器来提高用户体验，该架构的中心服务器存储有整个网络的拓扑结构，当用户的请求到达时，中心服务器首先使用 Dijkstra 算法将用户请求定向到与用户网络距离最近的服务器，然后再将该用户加入到最近的 P2P 对等节点中。文献[44]以斗鱼平台为例，研究了用户参与度对直播平台收益的影响，发现参与度高的用户可带来更高的平台利润，建议直播平台为参与度高的用户提供更多的带宽资源，以优化这部分用户体验，获取更高的平台利润。文献[45-47]提出通过监控网络链路质量状态，如往返时间（round-trip time, RTT）、带宽等，建立 QoE 模型，动态地为用户选择合适的服务器，以优化整体用户视频体验。文献[48]使用神经网络模型，根据网络服务质量预测用户 QoE，为用户选择距离较近且服务质量最好的服务器。

基于多指标的负载均衡技术使用多个指标为优化目标来定向用户请求。文献[49]中提出的用户请求定向策略综合考虑了服务器当前的负载及用户与缓存服务器的拓扑距离设计了负载均衡机制。类似的，文献[50]首先根据网络拓扑距离将服务器划分为不同的集群，然后建立了一个全局优化的服务器选择函数，该函数综合

考虑了业务性能、负载均衡、流量控制和开销来为用户选择综合性能最佳的服务器。文献[51]对 YouTube 的 CDN 测量数据进行了分析，通过对来自 3 个国家的 5 个节点、3 个运营商、2 个大学一周的视频数据进行分析，发现了 YouTube 以数据中心的 RTT 作为最关键的参数来定向用户请求，同时 YouTube 在定向时也考虑了服务器的负载和内容流行度这两个参数。

上述关于多服务器之间带宽资源调度的文献中，CDN 根据用户状态信息（如与服务器的距离、请求响应时延、带宽抖动）或服务器状态信息（如当前负载、RTT、流量开销等）为用户的请求选择合适的服务器。实际上，用户的视频体验也与视频画面的内容有关。本文根据不同画面时空复杂度（TI 和 SI）不同的视频，其用户视频体验函数不同这一特点，提出了一种视频内容感知的负载均衡算法，在服务器带宽资源有限的条件下，将不同的用户请求定向到合适的服务器，同时为每个请求分配合适的带宽资源，以尽可能提高用户体验。

2.3 CDN 人工智能应用的计算业务的资源调度研究现状

本节首先介绍了 CDN 支持人工智能应用的计算的研究背景，然后针对人工智能应用的计算业务的内存资源瓶颈，介绍了相关的解决思路与技术。

2.3.1 CDN 的人工智能应用的计算业务

人工智能应用的计算是 CDN 的一个新兴业务。由于一些实时的人工智能应用在执行推理和计算过程会使用 DNN 模型，需要消耗大量的计算以及存储资源，而一般轻量化的用户终端，如手机、智能手环等设备难以满足如此高的资源需求。一种直观的解决方法是将这类对计算和存储能力需求较高的任务定向到计算和存储资源充足的云端执行。但是用户与云端之间的高时延和网络抖动会严重影响计算任务的实时响应，进而影响用户体验。文献[4]提出“执行模型的缓存（model execution caching）”的概念，利用距离用户较近的 CDN 的边缘缓存服务器来实现 DNN 模型的计算，因此为人工智能应用提供计算成为 CDN 的一个业务。

利用 CDN 的边缘缓存服务器来实现 DNN 模型计算的基本思想是：将 DNN 模型预先缓存在边缘服务器的 GPU 内存中，用户只要把需要计算的请求及相关数据发送给边缘服缓存服务器，边缘缓存服务器就会在 GPU 中执行对应的 DNN 模型的计算，并将计算结果返回给用户。由于边缘缓存服务器距离用户较近，并且执行 DNN 模型计算时利用了高性能的 GPU，用户能够以较低的时延获得计算结果。

这种方法面临的资源瓶颈是边缘缓存服务器的 GPU 内存的容量。若一个应用

所需的模型都加载在 GPU 内存中，则该应用的计算请求可直接被执行，反之，则需将请求定向到远端的服务器或者从服务器硬盘中加载。现有研究表明，模型重定向和加载的时间是计算时间的数十倍[4,20]，表 2-1 给出了文献[4]中统计的典型的 DNN 模型在不同情况下的响应时间，从表中可以看出，当 DNN 模型已加载到 GPU 中时，响应时间一般为几十到上百毫秒，而当 DNN 模型没有加载到 GPU 中时，响应时间通常为上千毫秒，一个 DNN 模型的响应时间主要取决于 DNN 模型是否在 GPU 内存中加载。显然，CDN 边缘缓存服务器的 GPU 内存越大，可以加载的模型数目越多，进而可以满足更多类型应用的实时计算需求。但是由于成本和硬件的限制，GPU 内存无法做到无限大，不能缓存所有用户需要的 DNN 模型。因此，边缘缓存服务器的 GPU 内存容量是人工智能应用的计算业务面临的资源瓶颈。

表 2-1 DNN 模型的响应时间[4]
Table 2-1 The response time of DNN models[4]

（单位：ms）

DNN 模型	在 GPU 中的响应时间	不在 GPU 中的响应时间
NasNet Large	112.6±6.1	7 054.5±238.4
Inception V4	82.8±0.9	3 162.2±134.0
Inception V3	55.8±1.2	1 950.7±101.2
InceptionResNet V2	76.3±5.7	2 844.3±106.5
NasNet Mobile	55.3±4.1	2 817.2±123.7
DenseNet	49.6±3.2	1 149.0±108.0

2.3.2 面向人工智能应用的计算业务的缓存算法研究

面向人工智能应用的 CDN 缓存算法的主要目的是决定在边缘缓存服务器的 GPU 内存中存储哪些 DNN 模型，以尽可能满足用户请求的人工智能应用，提升用户体验。

在现有的关于人工智能的计算业务缓存的研究当中，文献[4]建议采用现有的用于视频或者网页的缓存算法[21-23]来缓存 DNN 模型，基本思想是把这些文献中的视频或者网页替换为 DNN 模型，这篇文献也提到了可根据实际情况对这些算法进行定制，但是并没有给出具体的算法。

除了上述采用传统缓存算法的方案，文献[20]提出了如下几种可能的方法来缓存 DNN 模型，提高缓存系统性能：

(1) 根据 GPU 内存空间调整 DNN 模型尺寸。这种方法类似于视频中的可扩展

编码器根据当前可用带宽或者用户需求来改变视频的码率，缓存算法可根据边缘缓存服务器可用 GPU 内存的大小或者用户的需求动态改变 DNN 模型的大小，从而使得有限的缓存空间可以存储更多的模型。模型大小的调整可以采用典型的 DNN 模型的压缩技术，例如模型剪枝、参数量化、知识蒸馏、精细模型设计、动态计算等[52]。目前，一些现有的研究已经证明了模型尺寸调整的可行性，这种方法可以实现 DNN 模型计算准确性和内存消耗之间的平衡[24,25]。

(2) 预先加载 DNN 模型的部分数据。这种方法将每个 DNN 模型的部分数据预先加载到缓存中，当一个模型被执行时，系统可先利用已被缓存 DNN 模型的部分来进行计算，同时加载被执行 DNN 模型的剩余部分。通过同时进行模型计算和模型加载来降低应用响应时间。具体的实现方法可以借鉴虚拟机迁移中的预老化、页面错误等技术。

(3) 联合优化计算请求调度和 DNN 模型选择算法。该方法将需要相同 DNN 模型的请求在时间上调度成一批，这样对于同一批次请求只需要加载一次 DNN 模型，从而降低模型加载的次数。在具体实现中，为了增加批次的规模，模型选择算法选择的 DNN 模型可能不是用户计算所请求的 DNN 模型，而是选择功能相似的 DNN 模型完成近似的计算。该方法的难点之一是如何定义不同的 DNN 模型在功能上的相似性。

(4) 主动加载和替换 DNN 模型。这种方法的主要思路是通过预测 DNN 模型的内存需求、计算精度、流行度等参数，主动预先加载和替换 GPU 内存中的 DNN 模型，而不是根据用户的实时请求被动去更新 DNN 模型。然而这种方法往往受到模型请求模式和预测精度的限制，当预测精度不够时，该方法提出可先暂时使用功能近似的 DNN 模型来替代被请求的 DNN 模型，同时加载计算所需的模型，这样可以尽量降低人工智能应用的响应时间。

(5) 协同使用不同边缘缓存服务器的 GPU 内存。该方法通过不同边缘缓存服务器之间的协作来提升系统的缓存性能。一种协作方法类似于网页的缓存，若当前的边缘缓存服务器没有存储计算所需的 DNN 模型，则计算请求将会被转发到存储有该模型的边缘缓存服务器。但是这种协作方法需要不同的缓存服务器之间频繁交换缓存状态信息，可能造成巨大的通信开销。另一种协作方法是将多个临近的边缘缓存服务器当作一个虚拟资源池，计算的请求在虚拟资源池内进行调度。这种方法可有效降低边缘缓存服务器之间的通信开销。

上述的究思路将 DNN 模型视为独立的数据文件进行缓存，给出了面向人工智能应用的缓存设计的基本思路，实际上人工智能应用与 DNN 模型之间可能存在着多对多的关联关系。本文根据人工智能应用与 DNN 模型之间的关联关系及其对响应时间的影响，提出面向人工智能应用缓存算法，在缓存空间有限的条件下，缓存

最有价值的人工智能应用所关联的 DNN 模型，尽可能提升用户体验。

2.4 常用的启发式算法

前文所提到的资源调度优化的研究一般通过最优化建模来进行求解。但一般来说，最优化模型求解的计算时间复杂度和空间复杂度随着问题规模的增加而呈指数型增长，无法在有效的时间范围内得出可行解。在这种情况下，使用启发式算法成为一种可以接受的选择，启发式优化算法虽然不能确保得出最优解，但能够在可接受的时间和空间消耗下给出可行解。本节接下来具体介绍几种最优化求解中常用的启发式算法。

2.4.1 贪婪算法

贪婪算法根据一定准则在求解过程中总是选取当前最优的步骤，来获得最优化模型的可行解[53]。算法设计没有固定的框架，求解的关键是贪婪策略的选择，即如何确定每步的最优解。贪婪算法不是总能得到整体的最优解，贪婪选择的过程必须只与当前的状态有关，不影响其以前的状态，即保证无后效性。贪婪算法一般包括如下几个步骤：

(1) 确定问题的最优子结构。

(2) 设计递归解，并保证在任一阶段，最优选择的一种方案总是贪婪算法的选择。

(3) 实现基于贪婪的递归算法，并将递归算法转换成迭代算法。

贪婪算法一般包括以下主要成分，具体为：

(1) 备选集合，求解结果从该集合中产生。

(2) 选择方程，用来决定每轮迭代中的一个可行选择，添加到解的集合当中。

(3) 一种可行解，来确定备选解是否符合求解的要求。

(4) 求解结果，表示贪婪算法的最终结果。

2.4.2 禁忌搜索算法

禁忌搜索算法模拟了人类的短期记忆，在求解过程中标记之前步骤中已经找到的局部最优解及求解过程，将其加入禁忌表，使得之后的搜索求解过程避开禁忌表中已求得的解，来避免重复搜索，防止搜索求解的过程出现无效循环导致算法效率降低[54]。禁忌表的长度可以根据求解问题的复杂度设置，当禁忌表满时，每次在

其中加入一个新的禁忌对象，都要删除一个旧的禁忌对象。

2.4.3 模拟退火算法

模拟退火算法模拟了热力学中固体退火的过程，通过在邻域中寻找相对较小的目标值，在多项式时间得到最优化问题的一个可行解[55]。

固体退火的原理如下：根据物理热学定律，固体温度较高时，固体内部的粒子热运动越剧烈，呈现快速无序的运动。但随着固体温度逐渐降低，其内部的粒子内能也随之降低，粒子运动趋于有序。当固体温度最终处于常温时，固体的内能达到最小，粒子的状态最为稳定。

模拟退火算法在求解时，首先随机设定一个解为初始温度，然后不断迭代求解更新求解的值。这个过程类似于固体温度降低的过程，随着求解的过程不断进行，最优化问题的解会趋于稳定。但是此时的稳定解可能是一个局部最优解，因此模拟退火算法中设置了随机跳出机制，以一定的概率跳出该解，来寻找目标最优化函数的全局最优解。

2.5 本章小结

在本章中，2.1 节介绍了经典的 CDN 视频架构，这是 CDN 业务的基础。2.2 节介绍了 CDN 视频业务的特点，以及 CDN 视频业务的资源分配方法。在现有的研究中，目前大多数 CDN 负载均衡方案根据网络传输参数或者视频采集压缩编码的参数量化用户体验提升带宽资源利用率。而最新的研究发现一个视频画面的用户体验是画面比特率的非线性增函数，且不同画面的用户体验函数参数不同，这为本文的研究提供了新的思路。2.3 节介绍了 CDN 支持人工智能的计算业务的研究背景，以及人工智能的计算业务的 CDN 资源分配方法。现有的内存资源分配思路主要包括两类，一类是沿用传统的基于流行度的缓存算法，另一类根据 DNN 模型的特点定制缓存算法，这为本文的研究提供了思路。2.4 节介绍了了常用的启发式算法，为求解资源调度最优化模型提供了一种可行的思路。

3 视频内容感知的 CDN 负载均衡算法

本章提出了视频内容感知的 CDN 负载均衡算法（content-aware load balancing algorithm, CLBA），根据最新研究发现的视频画面比特率与用户体验的关系，在服务器带宽容量的限制下提升用户总体体验。

本章节的结构安排如下：3.1 节介绍了视频内容感知的 CDN 负载均衡的研究背景；3.2 节介绍了应用场景与基本思想；3.3 节建立了视频内容感知的 CDN 负载均衡最优化模型，并提出了启发式算法；3.4 节基于真实视频数据进行仿真实验以评估算法性能；3.5 节为本章小结。

3.1 研究背景

3.1.1 用户观看体验与视频内容之间的关系

根据文献[17]，不同的视频帧由于画面内容不同，具有不同的 TI 和 SI。为了达到相同的用户体验，TI 和 SI 较高的视频帧需要更多的比特率来传输。也就是说用户体验与视频画面的内容有关，画面内容不同的视频帧要达到同样的用户体验，所需的比特率是不同的。实际上，文献[17]指出，该现象也存在于视频块的级别（一个视频块为一个固定播放时长的视频数据子文件，其时长通常在秒的数量级）。

SSIM 是一种常用的度量用户体验的指标[16,17]，文献[16]将视频帧 SSIM 随比特率之间的关系建模为体验函数并进行了公式化描述。视频帧的体验函数描述如下：

$$f_i(r) = 1 - \frac{1}{ar + b} \tag{3-1}$$

其中，$f_i(r)$ 表示在比特率 r 下视频帧 i 的 SSIM，a 和 b 是用户体验函数的参数，对于不同帧来说，由于 TI 和 SI 不同，a 和 b 的取值一般是不同的，a 和 b 满足约束 $a > 0$，$r > -b/a$。

图 3-1 绘制了在 a 和 b 的取值不同的情况下，视频帧 SSIM 随比特率 r 变化的关系曲线的例子。在这个图中，横轴为视频比特率，纵轴为视频 SSIM。由图可见，不同的视频帧的 SSIM 随比特率变化的非线性关系是不同的，这是由于不同视频帧的画面内容不同导致公式(3-1)中参数 a 和 b 不同。

结合图 3-1 和公式(3-1)可进一步观察到：首先，视频帧的 SSIM 随比特率变化的关系呈一个非减的凸函数，随着比特率的增加，视频帧的 SSIM 的增加速率逐渐

放缓。其次，不同帧的 SSIM 随比特率变化的关系曲线是不同的，即不同视频帧的 SSIM 对比特率变化的敏感程度不同。由文献[17]的结论，在视频块也有类似的现象。

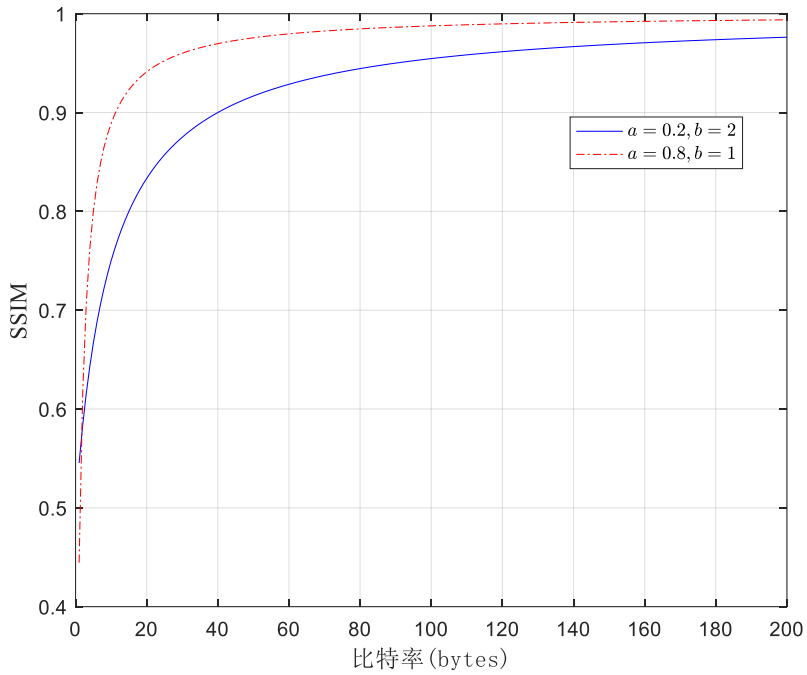


图 3-1 不同参数下视频 SSIM 随比特率变化的关系

Figure 3-1 The relationship between SSIM and bitrate for video frames under different parameters

3.1.2 CDN 视频业务负载均衡的研究现状

缓存和带宽资源是 CDN 视频业务的资源瓶颈，随着技术的发展，硬件存储资源的成本逐渐降低，带宽资源成为 CDN 视频业务的主要瓶颈资源[56]。本章主要研究 CDN 视频业务带宽资源的调度。

负载均衡是目前 CDN 调度带宽资源的常见方法，现有的 CDN 视频负载均衡机制大多是根据服务器的负载、网络链路的质量指标、视频卡顿、卡顿时延、视频起始时延、清晰度等因素定义用户 QoE 来调度 CDN 的带宽资源。例如，文献[57]通过动态探测网络链路状态将用户定向到性能最好的服务器上，文献[48]使用神经网络模型，根据网络服务质量预测用户 QoE，为用户选择距离较近且服务质量最好的服务器。

实际上，用户体验与视频内容也有密切关系。如 3.1.1 节所示，不同内容的视频画面的 SSIM 对比特率变化的敏感程度是不同的，即不同视频画面体验函数的参数不同。基于这个观察，本章将提出视频内容感知的负载均衡机制，考虑不同内容

的视频画面对比特率变化的敏感程度，在不同服务器之间调度用户请求，同时为每个请求分配合适的比特率，以最大化总体用户的视频体验。

3.2 应用场景与基本思想

本节介绍了视频内容感知的 CDN 负载均衡的具体应用场景，并阐述了研究的基本思想。

3.2.1 应用场景

如图 3-2 所示，本章的研究场景是一个多服务器的 CDN 网络架构，该架构由一个控制服务器、多个位于网络边缘的缓存服务器和若干个观看视频的用户组成。

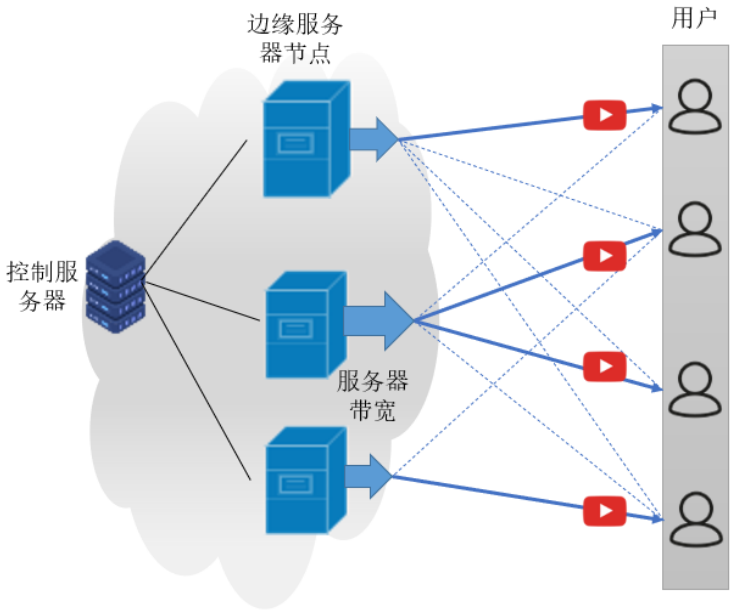


图 3-2 应用场景
Figure 3-2 Application scenario

系统是一个时隙系统，视频的分发决策以时隙为单位。本文假设边缘缓存服务器具有足够大的存储空间，可以存储所有视频的所有清晰度版本的数据。

假设有多个用户同时通过这些边缘服务器请求并观看视频，且这些用户一旦开始观看视频，他们的观看行为将会持续一段时间，也就是说系统可以预测每一个时刻每一个用户将要观看的视频画面。

本文进一步假设，一旦某个边缘缓存服务器将一个视频数据分发给其中一个用户，该用户就可通过 P2P 技术把收到的视频数据传输给其他请求这个视频数据

的用户。也就是说每个决策时刻边缘缓存服务器只需分发一次被请求的视频数据，即可满足当前请求该视频数据的所有用户。

本文利用不同内容的视频画面SSIM与比特率之间的非线性函数关系，在3.2.2节提出一种将用户视频请求定向到合适的边缘缓存服务器并确定视频传输比特率的带宽资源调度优化算法。这个算法运行在图3-2中的控制服务器上。

3.2.2 研究基本思想

本小节通过讨论一个具体场景提出如下基本思想：根据不同视频画面的体验函数，同时考虑服务器分发视频块的组合和视频码率的分配能够有效提高用户的整体视频体验。

如图3-3所示，假设有两个带宽均为100个单位的边缘缓存服务器，在一个时隙，有4个视频画面A、B、C和D分别被4个不同的用户请求，其中A和B具有相同的体验函数，C和D具有相同的体验函数，这些体验函数的曲线如图3-4(a)所示，在这个图中，横轴是比特率，纵轴是视频画面的SSIM。

方案 1：把体验函数相同的视频请求定向到同一个服务器。如图 3-3(a)所示，该方案将视频画面A和B的请求定向到服务器1，将视频画面C和D的请求定向到服务器2。如图3-4(a)所示，因为视频画面A和B的体验函数相同，视频画面C和D的体验函数相同，所以为每个请求分配50个单位的带宽。此时，视频画面A和B的SSIM为0.71，视频画面C和D的SSIM为0.97，此时4个用户获得总的SSIM为3.36。

方案 2：把体验函数不同的视频请求定向到同一个服务器。如图 3-3(b)所示，该方案将视频画面请求A和D定向到服务器1，将视频画面请求B和C定向到服务器2。如图3-4(b)所示，根据SSIM-比特率曲线，视频画面请求A和B分到30单位的带宽，SSIM为0.79，视频画面请求C和D分到70单位的带宽，SSIM为0.96，此时4个用户获得总的SSIM为3.5。

对比方案1和方案2，方案2的总体用户体验有明显提升。这是因为方案2将对比特率变化较敏感的视频画面和对比特率变化不那么敏感的视频画面的请求组合在一起。根据视频画面的体验函数，方案2为对比特率变化敏感的A和B多分配一些带宽，而为对比特率变化不那么敏感的C和D少分配一些带宽。这样，A和B的SSIM可以得到有效提升，同时C和D的SSIM不会降低太多。因此，方案2可以充分利用有限的带宽资源，把带宽尽量分配到SSIM增长最快的视频请求上，获得更高的总体用户体验。

从上述的实例可以看出，根据体验函数将不同视频请求定向到不同的边缘缓

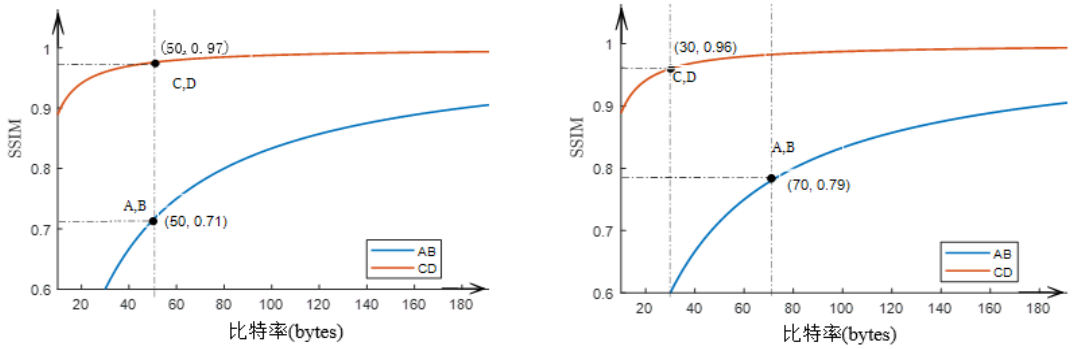
存服务器并为每个请求规划不同的视频比特率，能够有效提高用户的整体视频体验。根据这个基本思想，本文将在 3.3 节提出一个最优化模型，最大化用户的总体视频体验。

a) 请求定向方案 1 b) 请求定向方案 2

a) Request deployment schemes 1 b) Request deployment schemes 2

图 3-3 视频请求定向方案

Figure 3-3 Video request deployment schemes



a) 请求定向方案 1 的 SSIM b) 请求定向方案 2 的 SSIM

a) SSIM of request deployment schemes 1 b) SSIM of request deployment schemes 2

图 3-4 视频画面请求定向方案的 SSIM

Figure 3-4 SSIM of video requests for different deployment schemes

3.3 视频内容感知的 CDN 负载均衡优化模型及启发式算法

本节给出了系统的参数化描述，建立了视频内容感知的 CDN 负载均衡的最优化模型，并提出了相应的启发式算法求解该最优化模型。

3.3.1 系统描述

本小节将对 3.2 节中 CDN 系统进行参数化描述，以便后续最优化模型的建立，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
$\boldsymbol{M}=\{1,2,\ldots,\lvert\boldsymbol{M}\rvert\}$	边缘缓存服务器的集合，共$\lvert\boldsymbol{M}\rvert$个
$\boldsymbol{N}=\{1,2,\ldots,\lvert\boldsymbol{N}\rvert\}$	视频块的集合，共$\lvert\boldsymbol{N}\rvert$个
$\boldsymbol{B}=\{B_1,B_2,\ldots,B_{\lvert\boldsymbol{M}\rvert}\}$	边缘缓存服务器带宽容量的集合
$\boldsymbol{IDLE}=\{idle_1,idle_2,...,idle_{\lvert\boldsymbol{M}\rvert}\}$	边缘缓存服务器空闲的带宽集合
$\boldsymbol{W}^t=\{W_1^t,W_2^t,\ldots,W_{\lvert N\rvert}^t\}$	时隙t视频请求的集合，其中，W_i^t为时隙t视频块i被请求的用户数目
$\boldsymbol{U}_n$	尚未被定向到边缘缓存服务器的视频块集合
r_i^t	视频块i在时隙t被分发的比特率
r_l	视频块比特率的下界
r_u	视频块比特率的上界
$f_i(\bullet)$	视频块i的体验函数
v_i	视频块i的价值
$I_{ij}^t\in\{0,1\}$	视频块i在时隙t由服务器j分发的指示变量
$\boldsymbol{Y}=\{(i,j,r),...\}$	启发式算法输出的决策集合，元组(i,j,r)表示视频块i以比特率r被服务器j分发

系统的假设和已知的参数如下：

(1) 视频的分发决策以时隙为单位，每个时隙长度等于一个视频块的播放时长，每个视频块的播放时长是相同的，用户均在时隙的开始向服务器请求视频。系统中的视频块集合用$\boldsymbol{N}=\{1,2,\ldots,\lvert\boldsymbol{N}\rvert\}$表示，其中$\lvert\boldsymbol{N}\rvert$是视频块的总数目。

(2) 边缘缓存服务器的集合用$\boldsymbol{M}=\{1,2,\ldots,\lvert\boldsymbol{M}\rvert\}$表示，边缘缓存服务器的带宽容量用$\boldsymbol{B}=\{B_1,B_2,\ldots,B_{\lvert\boldsymbol{M}\rvert}\}$表示，其中$\lvert\boldsymbol{M}\rvert$是边缘缓存服务器的总数目，在本研究中认为服务器的带宽容量不随时间改变。

(3) 假设有多个用户同时通过这些边缘缓存服务器请求并观看视频，且这些用户一旦开始观看视频，他们的观看行为将持续一段时间，也就是说系统可以预测每一个时刻每一个用户将要观看的视频块，即在时隙t，每个用户请求的视频块已知。视频块被请求的用户数目用$\boldsymbol{W}^t=\{W_1^t,W_2^t,\ldots,W_{\lvert N\rvert}^t\}$来表示，其中$W_i^t$表示时隙$t$请求视频块$i$的用户数目。

(4) 每个视频块的SSIM与比特率的函数关系$f_i(\bullet)$，即体验函数，已提前求得，存储在控制服务器中。

系统的自变量如下：

(1) 二元指示变量 I_{ij}^t 表示视频块 i 在时隙 t 的定向决策，若视频块请求 i 在 t 时隙被定向到服务器 j，则二元变量 $I_{ij}^t=1$，否则 $I_{ij}^t=0$。

(2) 视频块 i 在时隙 t 被分发的比特率由 r_i^t 表示，r_i^t 不可以超过比特率的上界 r_u，也不可以小于比特率的下界 r_l。

算法求解过程中用到的表示系统状态的中间变量如下：

(1) $\boldsymbol{IDLE}=\{idle_1,idle_2,...,idle_{|\boldsymbol{M}|}\}$ 表示边缓存缘服务器当前向用户分发文件的可用带宽资源。

(2) 在视频块请求被定向过程中，尚未被定向到边缘缓存服务器的请求集合用 $\boldsymbol{U}_n$ 表示，$\boldsymbol{U}_n\in\boldsymbol{N}$，$\boldsymbol{U}_n$ 被初始化为 $\boldsymbol{N}$，$\boldsymbol{U}_n$ 中的元素会随着求解过程而变化。

3.3.2 最优化模型建立

基于 3.2.2 节中的基本思想，本小节将利用不同视频块的用户体验与比特率的关系建立最优化模型，在带宽资源有限的情况下，最大化总体用户的视频体验。

本章的研究假设用户到所有边缘缓存服务器的距离是相同的，但本研究很容易扩展到距离不同的情况。控制服务器需根据不同视频块的体验函数来进行带宽资源的分配，即需要决定每个时隙每个边缘缓存服务器上的用户请求的组合以及每个视频块被分发的比特率，即变量 I_{ij}^t 的取值和 r_i^t 的取值，以最大化整体用户的体验。建立最优化模型如下：

$$\max_{r_i^t,I_{ij}^t}\sum_{j=1}^{|\boldsymbol{M}|}\sum_{i=1}^{|\boldsymbol{N}|}W_i^tI_{ij}^tf_i(r_i^t) \tag{3-2}$$

限制条件如下：

$$\sum_{i=1}^{|\boldsymbol{N}|}I_{ij}^tr_i^t\le B_j,\forall j\in\boldsymbol{M} \tag{3-3}$$

$$\sum_{j=1}^{|\boldsymbol{M}|}I_{ij}^t=1,\forall i\in\boldsymbol{N} \tag{3-4}$$

$$I_{ij}^t\in\{0,1\} \tag{3-5}$$

$$r_l\le r_i^t\le r_u \tag{3-6}$$

目标函数用公式(3-2)表示，其中，$W_i^tI_{ij}^tf_i(r_i^t)$ 表示所有观看视频块 i 的用户获得的 SSIM，目标函数最大化时隙 t 所有用户的 SSIM。公式(3-3)约束在时隙 t，每个边缘缓存服务器所分发的视频的比特率之和不超过该服务器的带宽容量限制。公式(3-4)约束给定时隙一个视频块只能由一个边缘缓存服务器分发。公式(3-5)规定了 I_{ij}^t 是一个二元指示变量，$I_{ij}^t=1$ 表示视频块 i 在时隙 t 由服务器 j 分发，$I_{ij}^t=0$ 表示视频块 i 在时隙 t 不由服务器 j 分发。公式(3-6)约束了一个视频块的比特率不会超过

比特率的范围。

在该最优化模型中，I_{ij}^t 是离散变量，r_i^t 是连续变量，因此该优化问题是一个混合整数规划问题，当给定连续变量 r_i^t 的取值，求解离散变量 I_{ij}^t 需要遍历的组合数为 $|\boldsymbol{M}|^{|\boldsymbol{N}|}$，求解该最优化模型的计算复杂度较高，本文在 3.3.3 节提出相应的启发式算法求解该最优化模型。

3.3.3 启发式算法

本小节提出 CLBA 的具体实现。算法的主要思想为：首先设想一个虚拟的理想服务器，该服务器的带宽容量为所有服务器的可用带宽之和，求解出在理想服务器中待分发的视频块的最优比特率；然后计算最优比特率下的视频块价值，依次将价值最高的视频块分配到剩余带宽最多的真实服务器中，若所有的真实服务器的剩余带宽小于当前视频块的最优比特率，则重新求得待分发的视频块的最优比特率。重复上述步骤，直到所有视频块被分发。

具体地，该算法可以分为两个主要步骤。

(1) 建立理想服务器模型求解视频块分发比特率 r_i^t。算法首先设想一个虚拟的理想的服务器，求得在理想服务器中所有待分发的视频块的最优比特率，该理想服务器的带宽容量为所有真实边缘缓存服务器当前可用带宽之和。该理想服务器避开了实际系统中的离散变量，而待分发的视频块的比特率是连续变量，可以通过建立优化模型求解。该理想服务器获得的用户观看视频的总体体验是实际系统所能获得的用户体验的上界。理想服务器求解最优比特率的优化模型如下：

$$\max_{r_i^t} \sum_{i \in \boldsymbol{U}_n} W_i^t f_i(r_i^t) \tag{3-7}$$

限制条件如下：

$$\sum_{i \in \boldsymbol{U}_n} r_i^t \le \sum_{j}^{|\boldsymbol{M}|} idle_j \tag{3-8}$$

$$r_l \le r_i^t \le r_u \tag{3-9}$$

其中，目标函数为最大化当前未被分发的视频块的总 SSIM。公式(3-8)确保求得的最优比特率之和小于理想服务器的带宽容量。

(2) 求解定向决策变量 I_{ij}^t。求解步骤如下：

1) 求得理想模型的最优比特率之后，按照公式(3-10)计算每个视频块的价值，视频块 i 的价值 v_i 定义为该视频块可提供的总的 SSIM 除以该视频块的比特率。

$$v_i = \frac{W_i^t f_i(r_i^t)}{r_i^t} \tag{3-10}$$

2) 采用贪婪的方法将视频块的请求定向到合适的服务器。在该步骤中，每次

选择集合$\boldsymbol{U}_n$中价值最高的视频块作为目标视频块，选取剩余带宽最多的边缘缓存服务器作为目标服务器，若目标视频块的最优比特率小于目标服务器的可用带宽，则将目标视频块的请求定向到目标服务器，随后将目标视频块从集合$\boldsymbol{U}_n$中剔除，并更新目标边缘缓存服务器的剩余可用带宽。若目标视频块的最优比特率大于目标服务器的可用带宽，则重新使用理想服务器模型计算新的最优比特率，然后开始新一轮的贪婪定向求解。

重复步骤(1)和步骤(2)，直到所有的视频块请求全部被定向到合适的服务器。最后若服务器仍有剩余带宽，则将剩余带宽平均分给该服务器上的所有视频块的请求，最终得到时隙t的分发决策$\boldsymbol{Y}$。

具体算法如表 3-2 所示。

表 3-2 视频内容感知的 CDN 负载均衡算法
Table 3-2 Content-aware load balancing algorithm in CDN network

Algorithm 1

Input: $\boldsymbol{B}$, $\boldsymbol{N}$, $\boldsymbol{W}$

Output: $\boldsymbol{Y}$

Initialize: $\boldsymbol{Y}=\varnothing$, $\boldsymbol{U}_n=\boldsymbol{N}$, $\boldsymbol{IDLE}=\boldsymbol{B}$

While $\boldsymbol{U}_n \neq \varnothing$

  Calculate r_i for each $i \in \boldsymbol{U}_n$ with equation (3-7) - equation (3-9)

  Calculate v_i for each $i \in \boldsymbol{U}_n$ with equation (3-10)

  $i^* \leftarrow \arg\max_i(v_i), \forall i \in \boldsymbol{N}$

  $j^* \leftarrow \arg\max_j(idle_j), \forall j \in \boldsymbol{M}$

  If $r_{i^*} \le idle_{j^*}$ **then**

    $\boldsymbol{Y} \leftarrow \boldsymbol{Y} \cup \{(i^*, j^*, r_{i^*})\}$, $idle_{j^*} \leftarrow idle_{j^*} - r_{i^*}$, $\boldsymbol{U}_n \leftarrow \boldsymbol{U}_n \setminus \{i^*\}$

  End if

End while

Allocate the remaining idle bandwidth of each server equally to chunk assigned to that server

在该算法中，步骤 1 将决策集合$\boldsymbol{Y}$初始化为空集，将未被分配的视频块的集合$\boldsymbol{U}_n$初始化为视频块集合$\boldsymbol{N}$，将服务器的剩余带宽集合$\boldsymbol{IDLE}$初始化为带宽容量集合$\boldsymbol{B}$。步骤 3 用理想模型求解待分配的视频块的最优比特率，步骤 4 计算每个未被分配的视频块的价值。步骤 5 到步骤 10 使用贪婪的方法定向视频块请求。其中，步骤 5 选择价值最高的视频块作为目标视频块i^*。步骤 6 选择剩余带宽最多

的边缘缓存服务器作为目标服务器 j^*。步骤 7 比较目标视频块的最优比特率是否大于目标服务器的可用带宽，若小于，则步骤 8 将目标视频块的请求定向到目标服务器，并更新目标服务器可用带宽，将目标视频块从集合 $\boldsymbol{U}_n$ 中剔除。否则，算法执行从步骤 3 重新开始。重复以上步骤，直到所有的视频块请求被定向到合适的服务器上。最后若服务器仍有剩余带宽，则步骤 11 将剩余带宽平均分给该服务器上的所有视频块的请求，最终得到给定时隙 t 决策集合 $\boldsymbol{Y}$。

该算法在每轮迭代中主要分为两个步骤。首先，利用理想服务器模型，计算视频块最优比特率，该步骤可通过内点法、活动集等优化法等实现，其复杂度记做 $O(opt)$。求得最优比特率后，分别寻找价值最高的视频块和剩余带宽的最多的服务器，由于视频块数目最多为 $|\boldsymbol{N}|$，边缘缓存服务器的数目最多为 $|\boldsymbol{M}|$，则这两步的复杂度分别为 $O(|\boldsymbol{N}|)$ 和 $O(|\boldsymbol{M}|)$。

在最坏的情况下，每次在定向视频块的请求时，都需要重计算最优的比特率。每轮迭代的算法复杂度为 $\max(O(opt),O(|\boldsymbol{N}|),O(|\boldsymbol{M}|))$。由于一般情况下，$opt>|\boldsymbol{N}|>|\boldsymbol{M}|$，而且最坏条件下算法需要迭代 $|\boldsymbol{N}|$ 次，所以该算法的复杂度为 $O(|\boldsymbol{N}|(opt))$。

3.4 实验结果与分析

本节对本文提出的 CLBA 算法进行了仿真以评估算法在不同实验设置下的性能。本节首先给出实验中视频画面体验函数参数的获取方法，然后介绍了实验中的对比算法，最后进行了系统性的仿真并给出分析。

3.4.1 视频帧的体验函数的获取

为测试算法性能，本文使用一个帧长作为时隙系统的决策的时间间隔，并使用真实的视频帧获取视频画面体验函数的参数。

本文首先从腾讯视频网站下载了来自不同类型的 20 部电影，共上百万帧。然后对文献[16]中的工作进行了复现，具体过程如下：本文首先使用 ffmpeg[58]将这 20 部电影进行降质，得到清晰度不同的 5 个版本的视频，然后使用莫斯科国立大学 Graphics and Media Lab 制作的 MSU Video Quality Measurement Tool[59]计算视频帧的 SSIM。由于 MSU Video Quality Measurement Tool 的限制，计算使用的原始视频版本最高为 720P；且据研究统计，国内不同视频平台的用户对全程流畅 1080P 片源带宽需求多数情况下仅为 10%左右[60]。因此本文以 720P 的视频作为原始样本视频，平均比特率为 2mbps，来求得每个视频帧不同比特率下的 SSIM，720P 版本的

视频质量被定义为单位 1，即转码后的其它版本的视频质量以 720P 版本的视频质量进行归一化。本文的研究可以很容易的扩展到更高视频比特率的场景。

在求得视频体验函数的参数之后，本文随机选取 200 个帧作为实验仿真的视频数据，最后对视频帧的 SSIM 和比特率的函数关系进行拟合获得视频帧的体验函数的参数，即公式(2-1)中的 a 和 b 。图 3-5 绘制了所选取的 200 个帧的 SSIM 随比特率变化的曲线图。图中横轴代表帧的比特率，纵轴代表 SSIM。图中每一条颜色不同的曲线都代表一个不同的帧。由图可见，为了达到相同的视频体验，不同画面的视频帧需要不同的比特率。文献[17]指出，这种现象也存在于视频块的级别。

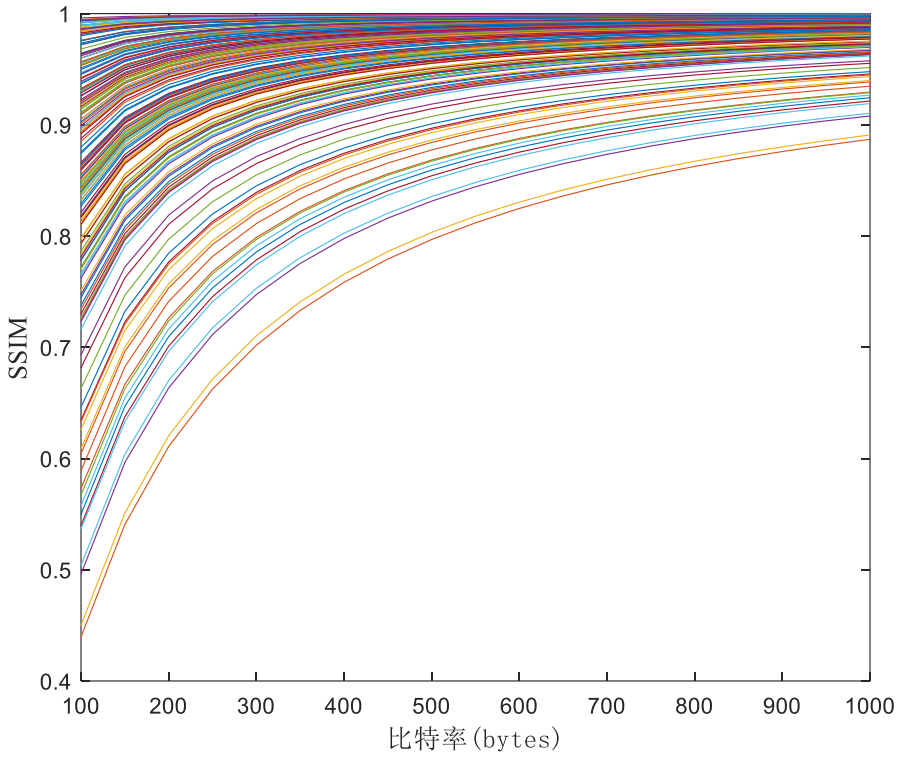


图 3-5 视频帧 SSIM 随比特率变化的关系曲线
Figure 3-5 Relationship between SSIM and bitrate of video frames

3.4.2 对比方案

本文选取如下 4 种算法作为本文算法的对比算法，分别是：

(1) 随机算法（记为 RND）：该算法将视频请求随机定向到边缘缓存服务器，并设置一个服务器上的视频帧均分该服务器的带宽。这种算法与互联网中的经典的域名系统方案相对应，即将用户随机定向到边缘缓存服务器。

(2) 均衡算法（记为 EVEN）：该算法为不同的边缘缓存服务器均匀分配分发的视频帧的数目，使得服务器分发的视频帧的数目与服务器带宽容量呈正比关系，

同时，一个服务器分发的视频帧均分该服务器的带宽。这种算法对应典型的负载均衡机制。

(3) 基于流行度的负载分配算法（记为 POP）：该算法在定向视频帧的请求时，将视频帧的流行度作为考虑因素之一。该算法首先按照边缘服务器带宽容量的多少，成比例的为服务器分配相对应的用户请求数目，然后根据求得的用户请求数目，贪婪地将请求定向到边缘缓存服务器上，以确保定向到每个服务器上的用户请求数目与计算分配数值相一致，最后设置由同一台服务器分发的视频帧均分该服务器的带宽。该算法对应基于流行度的负载均衡机制。

(4) 理想模型算法（记为 IDEA）：如 3.3.3 节中的描述，该算法将多个边缘缓存服务器视为一个虚拟的理想服务器，理想服务器的带宽为所有边缘服务器带宽之和。该算法根据视频帧的体验函数求解最优的视频比特率，对应于本文理想服务器模型，是本文提出算法的性能的上界。

为了更直观的显示这些对比算法与本文算法 CLBA 的性能差异，本文使用性能提升指数作为评价指标，性能提升指数 *index* 的定义如公式(3-11)所示。

$$index = \frac{SSIM_{CLBA} - SSIM_{baseline}}{SSIM_{baseline}} \tag{3-11}$$

其中 $SSIM_{CLBA}$ 代表本文中 CLBA 算法获得的 SSIM，$SSIM_{baseline}$ 代表对比的算法获得的 SSIM。

3.4.3 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，分析本文所提出的 CLBA 算法的性能。假设所有服务器具有相同的带宽容量（在实际中，带宽也可设置为不相同），一个时隙内视频帧总的请求数目设置为 1 000。在仿真过程中，每种实验设置下每个算法重复 100 次并取平均值作为算法在该实验设置下的输出结果。具体的实验结果和分析如下。

(1) 边缘缓存服务器带宽容量对算法性能的影响。

实验设置每个视频帧被请求的数目相同，边缘服务器的数目为 10，每个服务器带宽相同，同时以步长 50 从 10 到 500 改变服务器带宽。在该仿真条件下，由于每个视频帧被请求的数目与各边缘缓存服务器的带宽是相同的，因此每个帧的流行度是相同的，所以该设置下 EVEN 算法和 POP 算法的机制相同，即为每个服务器定向相同数目的请求。实验结果如图 3-6 所示。在图 3-6 中，横轴为边缘缓存服务器带宽容量，纵轴是性能提升指数。

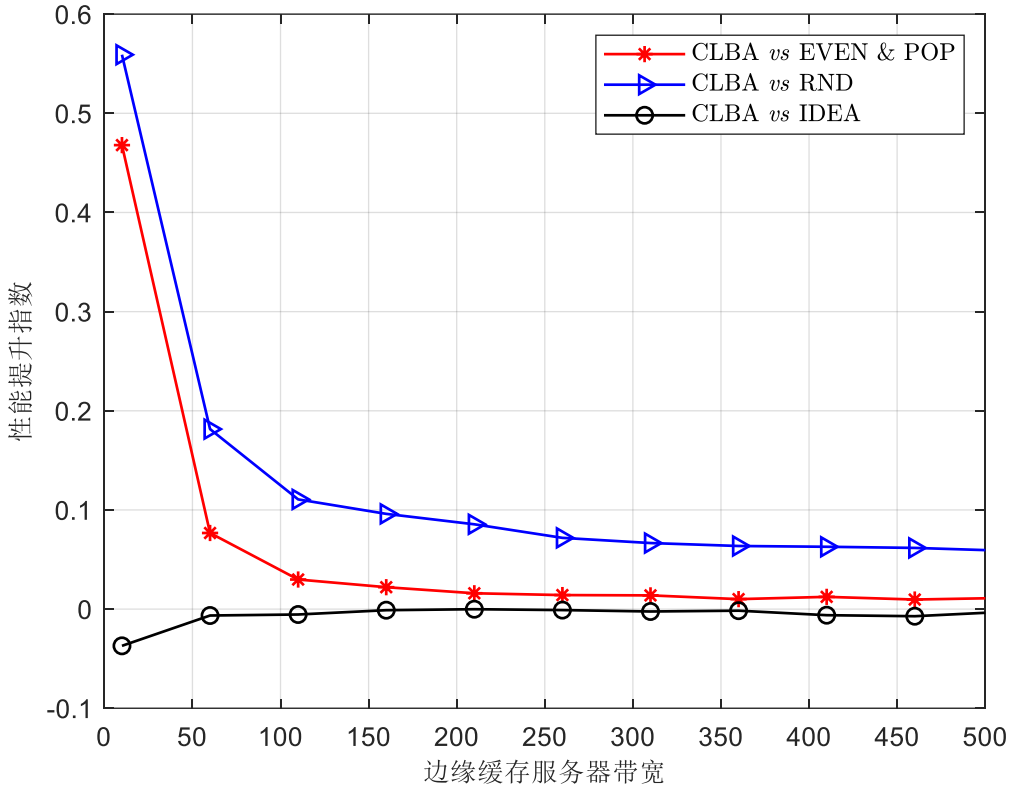


图 3-6 不同带宽容量下性能相对提升指数

Figure 3-6 The relative performance improvement index under different bandwidth capacity

由图 3-6 可知，当边缘缓存服务器容量大于 100 时，用户整体 SSIM 相较于 RND 提升 10%左右，相较于 EVEN 和 POP 提升 5%左右，且与理想模型 IDEA 差距很小，维持在 2%左右。这是因为本文的算法在分配带宽时，考虑了不同视频帧的内容的差异，按照视频帧的质量函数进行分配，因此优于现有的算法；同时本文的算法根据理想模型求得的最优比特率迭代定向视频帧请求，因此与 IDEA（性能的上界）差异较小，这也证明了本文 CLBA 算法的有效性。

从图 3-6 中进一步可以得知，相较于服务器带宽容量大于 100 的情况，本文的 CLBA 算法在边缘缓存服务器带宽容量小于 100 时性能提升更加显著。这是因为当带宽不足时，大多数视频帧对比特率的变化比较敏感，即处于 SSIM-比特率图中曲线斜率较大的位置，此时增加较少的比特率便可获得较高的 SSIM 提升。而随着带宽逐渐增加，虽然每个视频帧可分得的比特率也增加，但是 SSIM 的提升逐渐接近性能的上界，SSIM-比特率曲线斜率逐渐趋近于 0，此时比特率增加，视频帧 SSIM 的提升放缓，用户体验基本保持不变，这显示了 CLBA 算法在带宽不足的情况下性能更加显著。

(2) 视频帧流行度参数对算法性能的影响。

实验设置边缘缓存服务器的数目为 10，每个服务器带宽容量为 100，设置视频帧的流行度服从 Zipf 分布，并以步长为 0.1 从 0.1 到 1 改变 Zipf 分布的参数。Zipf 分布的概率质量函数如公式(3-12)所示[61]。其中，x 为视频帧流行度的排名，$|\boldsymbol{N}|$ 为视频帧的总数，α 为流行度分布的参数，α 取值越高，视频帧的流行度分布越集中，即越少数的视频帧获得越高的流行度。公式(3-12)表示在参数 α 下，排名为 x 的视频帧的流行度。

$$f(x)=\frac{1}{x^{\alpha}\sum_{i=1}^{|\boldsymbol{N}|}\left(\frac{1}{i}\right)^{\alpha}},x=1,2,...,|\boldsymbol{N}| \tag{3-12}$$

实验结果图如图 3-7 所示。在图 3-7 中，横轴为 Zipf 分布的参数 α，纵轴为性能提升指数。CLBA 与 IDEA 的差距在 2%以内，具有良好的近似性能。CLBA 的有效性主要是因为考虑了不同视频帧 SSIM 随比特率变化的非线性关系，为每个视频帧请求分配合适的比特率。

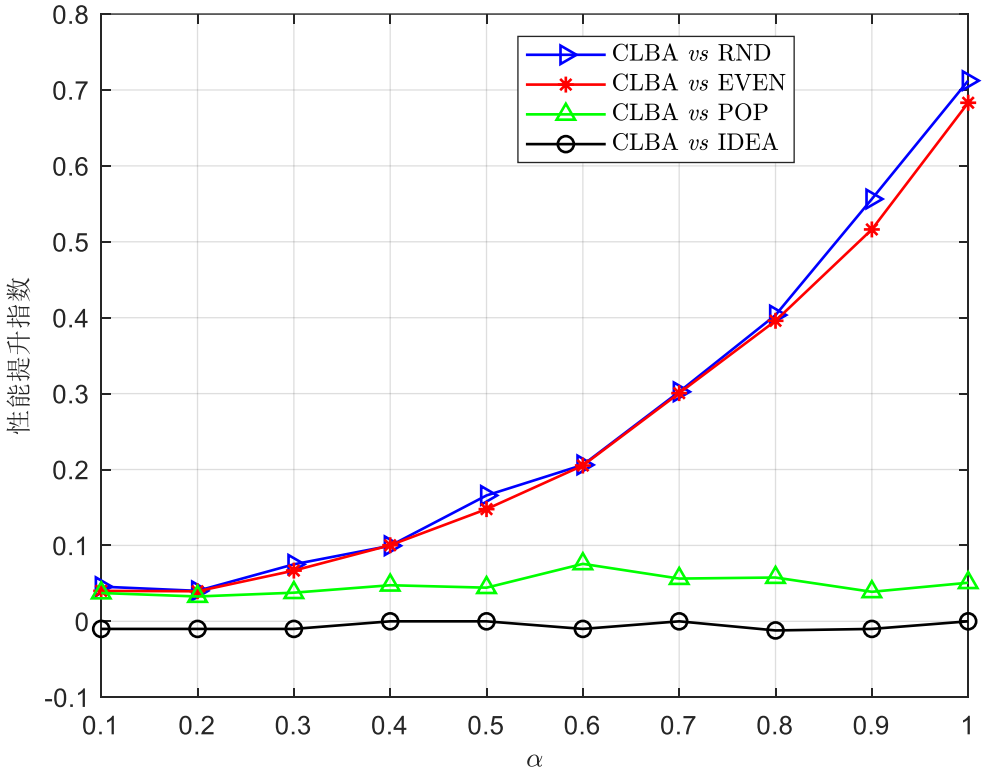


图 3-7 不同 Zipf 参数 α 下算法的性能相对提升指数

Figure 3-7 The relative performance improvement index under different α

从图 3-7 可进一步得知，当 α=0.1 时，CLBA 与 RND、EVEN 和 POP 三种算法相比，可提升用户整体 SSIM 约 5%，随着 α 增加，本文的 CLBA 与 RND 和 EVEN 相比，性能提升更加明显。这是由于随着 α 增大，越多用户的请求集中于越

少部分的视频帧，而 RND 和 EVEN 算法不考虑帧的流行度，将这些流行度高的帧与流行度低的帧随机分到同一服务器，并平分服务器带宽，因此用户获得的 SSIM 相对较低。而本文提出的 CLBA 通过用户视频体验与视频帧的比特率的比值来量化视频帧的价值，为价值高的请求分配更多的比特率，因此相同的设置下可有效提升用户的总体 SSIM。

(3) 边缘缓存服务器数目对算法性能的影响。

实验设置每个视频帧具有相同的用户请求数目，以步长为 2 从 2 到 30 改变边缘缓存服务器的数目，设置每个服务器具有相同的带宽，在服务器带宽总和为 500 和 1 000 的情况下分别进行实验。由于每个视频帧具有相同的用户请求数目，该条件如实验(1)中的设置情况一样，因此与实验(1)中的分析相同，本实验设置下 POP 和 EVEN 算法的机制相同。

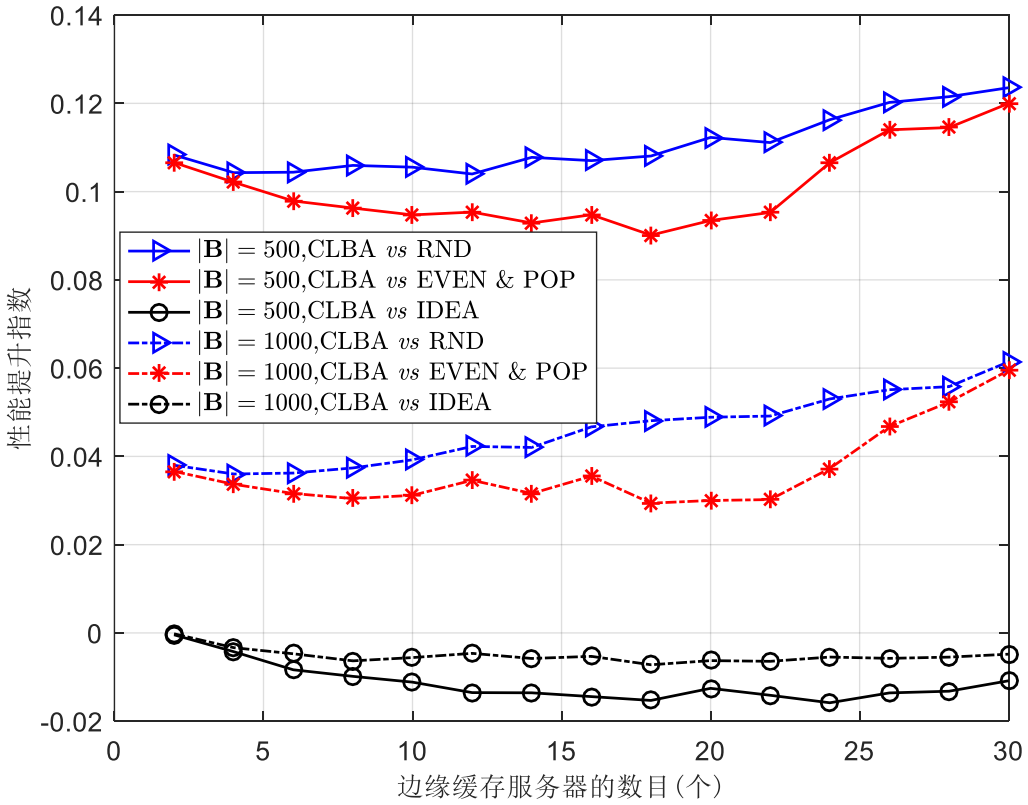


图 3-8 不同边缘服务器数目下算法的性能相对提升指数

Figure 3-8 The relative performance improvement index of the algorithm under different number of edge servers

实验结果如图 3-8 所示。在图 3-8 中，横轴为边缘缓存服务器的数目，纵轴为性能提升指数。由图 3-8 可知，本文提出的 CLBA 算法明显优于现有算法 RND、EVEN 和 POP，且与理想性能 IDEA 差距很小。在边缘缓存服务器带宽总和为 500

时，CLBA 的相对性能提升为 10%左右，在边缘缓存服务器带宽总和为 1 000 时，相对性能提升为 4%左右，服务器带宽越不足，性能提升越明显。该现象与实验(1)中的实验结果相符，原因也与实验(1)中的分析相同。同时，CLBA 算法的性能相对相较于现有算法呈现一个缓慢上升的趋势，这说明CLBA算法具有较好的鲁棒性。结合实验(1)和实验(2)可知，相较对比算法，CLBA 算法可提升性能 3%-70%。

3.5 本章小结

本章提出了一种视频内容感知的 CDN 负载均衡算法。该算法以充分利用有限的边缘缓存服务器带宽资源、最大化用户视频体验为目标，在每个时隙将用户视频请求定向到合适的服务器，同时为每个请求分配合适的比特率。本章首先介绍了不同内容的视频画面的用户体验与比特率的关系，然后基于该关系提出了优化的基本思想，随后基于该优化的基本思想对 CDN 负载均衡问题进行最优化建模，在带宽有限的情况下，最大化总体用户体验。由于该最优化模型求解复杂度高，本章设计了一种贪婪启发式算法对其进行求解。实验的数值结果表明，本文提出的负载均衡算法可提升系统性能 3%-70%。

4 面向人工智能应用的缓存算法

本章提出了面向人工智能应用的缓存算法，根据人工智能应用业务的特点，在CDN 边缘缓存服务器 GPU 内存有限的情况下，缓存人工智能应用所使用的 DNN 模型，尽可能缩短应用的响应时间，提高缓存命中率。

本章节的结构安排如下：4.1 节介绍了面向人工智能应用的缓存算法的研究背景，4.2 节介绍了本研究的应用场景与基本思想；4.3 节根据人工智能应用的特点建立最优化模型，并提出了贪婪启发式算法；4.4 节设计仿真实验评估算法性能；4.5 节为本章小结。

4.1 研究背景

人工智能应用的计算业务将成为未来 CDN 的主要业务。随着人工智能技术的快速发展，基于人工智能的应用也日渐普及。一般而言，人工智能应用需要耗费较多计算和存储资源来执行 DNN 模型，而一般的用户终端无法提供充足的算力和存储空间。因此，人工智能应用的计算一般采取服务器-客户端架构，用户向服务器端发起计算请求，服务器执行计算后将结果返回给用户。服务器端一般包括云中心和边缘服务器，然而用户距离云中心的网络距离较远，而且网络波动较大，将计算任务卸载到云中心会引入较大的时延。因此目前一个有效解的决方案是将人工智能应用的计算卸载到距离用户较近的 CDN 边缘缓存服务器。

边缘缓存服务器的 GPU 内存是 CDN 人工智能应用的计算业务的资源瓶颈。人工智能应用的计算业务利用边缘缓存服务器的 GPU 内存执行对应的 DNN 模型的推理计算，在服务用户的过程中，用户的响应时间（用户发起请求到用户收到计算结果的时间间隔）是一个重要的业务性能指标。但是根据文献[4,20]，从硬盘中将 DNN 模型导入 GPU 内存的时间远远大于模型计算时间。因此，当用户所请求的人工智能应用所需的 DNN 模型没有加载在缓存中时，用户的响应时间将会明显增加，例如当 InceptionV4 缓存在 GPU 中时，响应时间为 82.8 ms 左右，而如果从硬盘中加载，InceptionV4 的响应时间为 3 162.2 ms 左右。然而边缘缓存服务器的 GPU 内存是有限的，只能选择有限的 DNN 模型加载在 GPU 内存中。

目前典型的解决方案是设计缓存算法来缓存合适的 DNN 模型以缩短应用的响应时间。一种直观的方法是沿用传统的文件缓存方法，即根据人工智能应用或 DNN 模型的流行度选择需要存储的模型，例如可以使用文献[21,22]中的缓存方法。另外一种方法是根据 DNN 模型的特点设计缓存算法，文献[20]给出了如下思路：

根据可用内存空间的大小，使用模型压缩技术动态地改变 DNN 模型的大小；在缓存中预先存储一部分 DNN 模型的数据，在计算过程中使用预先加载的模型进行计算，同时加载剩余的 DNN 模型数据；将 DNN 模型的请求按照功能类别聚合成不同的批次，每一批次只加载一次 DNN 数据。通过预测 DNN 模型的流行度等参数，主动预先加载和替换 GPU 内存中的 DNN 模型；通过协同调度不同边缘缓存服务器来提升系统的缓存性能。

人工智能应用的计算业务不同于传统的文件分发业务。如在传统的文件分发业务中，一个请求对应于一个特定的文件。而一个人工智能应用可能会需要多个 DNN 模型才能满足计算需求。本文将在 4.2 节详细介绍人工智能应用的计算业务的特点。

4.2 应用场景与基本思想

本节介绍了面向人工智能应用的缓存算法的应用场景，并通过具体实例详细阐述人工智能应用的计算业务的特点和研究的基本思想。

4.2.1 应用场景

如图 4-1 所示，本章的研究场景由一个远端服务器、一个位于网络边缘的缓存服务器和若干个请求计算服务的用户组成。其中，远端服务器距离用户较远，具有充足的 GPU 内存资源，能够缓存用户请求的所有 DNN 模型。边缘缓存服务器距离用户较近，GPU 内存有限，只能缓存部分 DNN 模型。图中边缘缓存服务器和远端服务器中的不同颜色的方块表示不同的 DNN 模型。用户终端可以运行各种人工智能应用，用户会以一定的概率使用某个应用，并发起人工智能应用的计算请求，即每个应用具有一定的流行度。一个人工智能应用的计算可能需要多个 DNN 模型并行执行来完成该应用的最终功能，其中的每个 DNN 模型都用于实现该应用的一个特定的子功能。

在这个场景中，GPU 内存的缓存状态会影响计算的性能。具体来说，若一个人工智能应用所关联的 DNN 模型都存储在 GPU 内存中，则边缘缓存服务器可直接执行相关的 DNN 模型计算，并将结果返回给用户，用户请求的响应时间较短，这种情况称为缓存命中。否则，用户的请求将被定向到远端服务器执行，这个过程会引入数十倍于执行 DNN 模型的时延，因此用户请求的响应时间较长，用户体验较差，这种情况称为缓存未命中。

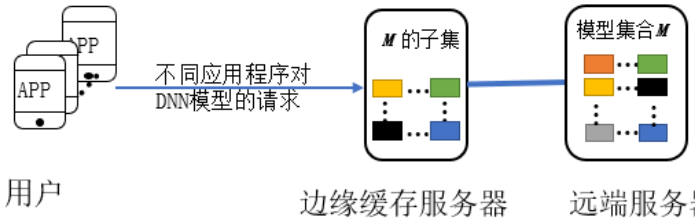


图 4-1 应用场景
Figure 4-1 Application scenario

由于一个人工智能应用（后文简称为 APP）的计算任务需要同时执行其关联的所有 DNN 模型，因此用户请求的响应时间取决于用户请求的 APP 所关联 DNN 模型中响应时间最长的那个模型。而一个 DNN 模型的响应时间由该模型是否在 GPU 内存中决定。如图 4-2 所示，假设 DNN1、DNN2 和 DNN3 被提前缓存到 GPU 内存当中，而 DNN4 和 DNN5 未被缓存。若 APP1、APP2 和 APP3 同时被请求，该情况下，APP1 和 APP2 的请求命中，响应时间比较短，而 APP3 请求未命中，其响应时间较长。

上面介绍了影响一个 APP 响应时间的因素，实际上边缘缓存服务器需要为众多的 APP 服务，系统性能由 APP 整体的响应时间决定。因此，缓存的设计需要考虑所有的 APP 关联的 DNN 模型。这个时候就需要观察多个 APP 与多个 DNN 模型之间的关系。

实际上，不同于传统的文件缓存，人工智能应用与 DNN 模型之间可能存在一个多对多的关系[62,63]。具体的例子如图 4-2 所示，其中，APP1 和 APP3 使用了 3 个 DNN 模型，APP2 使用了 2 个 DNN 模型。同时，DNN3 被所有的 3 个 APP 所使用。而在传统的缓存中，一个用户请求对应着一个特定的数据文件，具体的例子如图 4-3 所示，图中共有 3 个不同的视频请求，每个视频请求都对应着一个特定的视频文件。

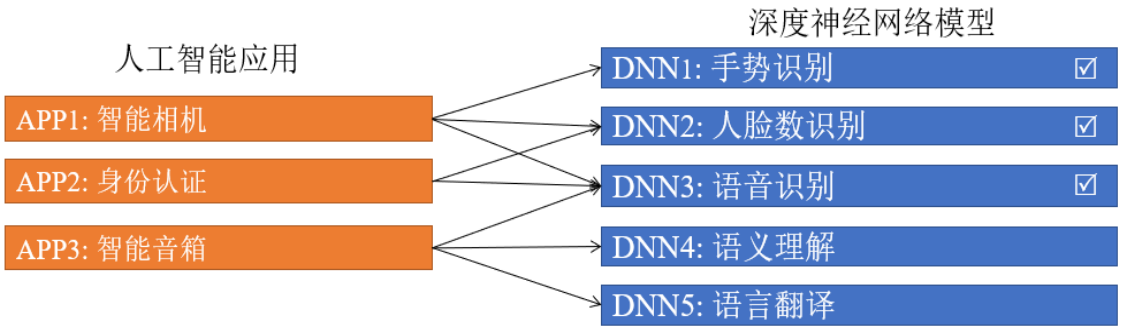


图 4-2 人工智能应用和 DNN 模型之间的关系
Figure 4-2 Relationships between applications and DNN models

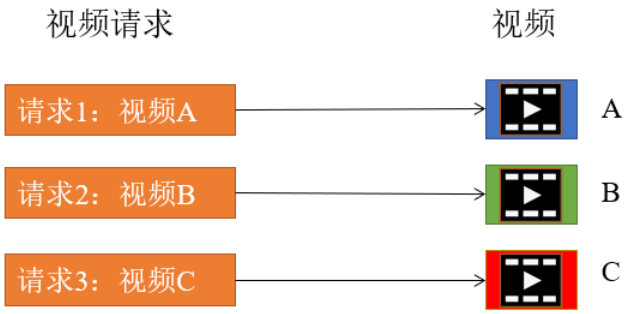


图 4-3 视频请求与视频文件之间的关系
Figure 4-3 Relationships between video requests and videos

4.2.2 研究基本思想

利用 APP 与 DNN 模型之间的多对多关系能够提升缓存性能。下面首先给出两个基于流行度的传统缓存方案，然后给出一个基于 APP 与 DNN 之间的多对多关系进行缓存的方案，本小节将通过具体例子说明后者比只利用流行度进行缓存的传统算法具有更好的性能。

仍以图 4-2 为例，系统中有 APP1、APP2 和 APP3 共 3 个 APP，它们被用户请求的概率依次为 0.3、0.3 和 0.4。假设每个 DNN 模型大小相同，边缘缓存服务器的 GPU 内存只能够缓存 3 个 DNN 模型。

方案 1：仅根据 APP 的流行度缓存 DNN 模型。在该场景下，一个 APP 的流行度为该 APP 被请求的概率。APP3 的流行度最高，为 0.4，因此，该方案缓存 APP3 使用的 DNN3、DNN4 和 DNN5 这 3 个模型。此时对 APP3 的计算请求可直接被边缘缓存服务器执行，而 APP1 和 APP2 所需的 DNN 模型没有全部被缓存到 GPU 内存中，因而请求需要被重定向到远端服务器，从而引入较高的时延。在该缓存方案下，缓存命中率为 0.4，有 40%（即 APP3 的流行度）的用户获得较高的体验。

方案 2：仅根据 DNN 模型的流行度来进行缓存。在该场景下，一个 DNN 模型的流行度定义为与该模型相关的所有 APP 的流行度之和，例如 DNN2 的流行度等于 APP1 的流行度加 APP2 的流行度（因为 APP1 和 APP2 都使用了 DNN2）。按照 DNN 模型的流行度排序结果为：DNN3、DNN2、DNN4、DNN5、DNN1。因此该方案缓存 DNN2、DNN3 和 DNN4，此时只有 APP2 所需的模型全部被缓存到 GPU 内存中。该缓存方案下，缓存命中率为 0.3，有 30%的用户获得较低的响应时间（即 APP2 的流行度）。

方案 3：基于 APP 与 DNN 之间的多对多关系进行缓存设计。该方案缓存 DNN1、DNN2 和 DNN3，这种缓存方案可以同时满足用户对 APP1 和 APP2 的请求。因为

APP1 的流行度为 0.3，APP2 的流行度为 0.3，因此该缓存方案下，缓存命中率为 0.6，有 60%的用户获得较低响应时间，相较于方案 1 和方案 2 性能有明显提升。这是因为该方案考虑了 APP 和 DNN 之间的多对多关系，缓存了 APP1 与 APP2 重用的 DNN2 和 DNN3。由于当前的 GPU 内存可容纳 3 个 DNN 模型，此时再缓存 DNN1 就可再满足对 APP1 的用户请求。

对比方案 1、方案 2 和方案 3 可以发现，单一根据 APP 或者 DNN 模型的流行度的进行缓存的方案通常不是最优的。这是因为单一根据 APP 流行度的缓存方案没有考虑到 DNN 模型在不同的 APP 之间的重用特性，因此该方案可能满足的用户 APP 请求不是最多的。而单一根据 DNN 模型流行度的缓存方案没有考虑到一个 APP 的整体响应时间取决于该 APP 相关联的 DNN 模型中响应时间最高的 DNN 模型个体，而单个 DNN 模型的响应时间取决于该模型是否在缓存中，因此这种方法不能获得最优的缓存性能。而方案 3 在使用 APP 流行度的基础上，利用 APP 与 DNN 模型之间的多对多的关系，使得缓存的 DNN 模型能够直接满足更多用户的 APP 计算请求。

基于利用 APP 与 DNN 之间的关联关系进行缓存的基本想法，本文将在 4.3 节提出一个最优化模型，在有限的 GPU 内存中缓存合适的 DNN 模型，以最小化 APP 的响应时间。

4.3 面向人工智能应用的缓存最优化模型及启发式算法

本节参数化了系统模型，建立了面向人工智能应用的缓存最优化模型，并提出了相应的启发式算法。

4.3.1 系统描述

本小节将对系统进行参数化描述，以方便后续缓存优化模型的建立，表 4-1 列出了相关符号与参数。

APP 的集合记为 $\boldsymbol{T}=\{1,2,...,|\boldsymbol{T}|\}$，其中 $|\boldsymbol{T}|$ 是 APP 的总数。DNN 模型的集合记为 $\boldsymbol{M}=\{1,2,...,|\boldsymbol{M}|\}$，其中 $|\boldsymbol{M}|$ 是 DNN 模型的总数。APP i 使用到的 DNN 模型集合用 $\boldsymbol{A}_i$ 表示，APP i 使用到而且当前已被缓存的 DNN 模型集合用 $\boldsymbol{A}_i^C$ 表示，显然 $\boldsymbol{A}_i \in \boldsymbol{M}$，$\boldsymbol{A}_i^C \in \boldsymbol{M}$。

由于 DNN 模型可以通过模型组合以及压缩技术动态地调整大小，因此为了简化建模，本研究假设每个 DNN 模型大小相同。缓存服务器 GPU 内存大小用符号 $\boldsymbol{C}$ 表示，表示内存中最多可以容纳 $\boldsymbol{C}$ 个 DNN 模型。

APP i 的流行度用 p_i 表示，表示用户会以 p_i 的概率请求 APP i。I_i 是一个二元指示变量，若 APP i 用到的 DNN 模型 $\boldsymbol{A}_i$ 全部缓存在 GPU 内存中 $I_i=1$，否则 $I_i=0$。

缓存的决策向量用 $\boldsymbol{L}=(l_1,l_2,\ldots,l_{|\boldsymbol{M}|})$ 表示，其中 l_i 是一个二元变量，$l_i=1$ 代表 DNN 模型 i 缓存在 GPU 内存中，$l_i=0$ 代表 DNN 模型 i 没有缓存在 GPU 内存中。

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

符号	含义
$\boldsymbol{T}=\{1,2,...,\lvert\boldsymbol{T}\rvert\}$	APP 的集合，共 $\lvert\boldsymbol{T}\rvert$ 个 APP
$\boldsymbol{M}=\{1,2,...,\lvert\boldsymbol{M}\rvert\}$	DNN 模型的集合，共 $\lvert\boldsymbol{M}\rvert$ 个 DNN 模型
$\boldsymbol{A}_i$	APP i 用到的 DNN 模型的集合，$\boldsymbol{A}_i\in\boldsymbol{M}$
$\boldsymbol{A}_i^C$	APP i 已被缓存的 DNN 模型集合，$\boldsymbol{A}_i^C\in\boldsymbol{M}$
C	边缘缓存服务器 GPU 内存大小
p_i	APP i 的流行度
I_i	APP i 缓存指示变量，$I_i=1$ 表示 APP i 用到的 DNN 模型 $\boldsymbol{A}_i$ 全部被缓存，否则 $I_i=0$
$\boldsymbol{L}=(l_1,l_2,\ldots,l_{\lvert\boldsymbol{M}\rvert})$	缓存决策向量，其中 $l_i=1$ 代表 DNN 模型 i 被缓存，$l_i=0$ 代表 DNN 模型 i 不被缓存
v_i	APP i 的价值

4.3.2 最优化模型建立

本小节将利用 4.2 节介绍的 APP 与 DNN 模型的关联关系建立最优化模型，以在边缘缓存服务器 GPU 内存有限的情况下，降低 APP 平均响应时间。

在 4.2 节所描述的应用场景下，本节首先对 APP 响应时间进行建模。若用户请求的 APP 所需的 DNN 模型都缓存在边缘缓存服务器的 GPU 中，本文称该 APP 命中缓存，此时边缘服缓存服务器可直接执行相关计算并返回结果，这种情况下，APP 的平均响应时间为常数 D_1。反之，若一个 APP 所需的 DNN 模型没有全部被缓存在边缘缓存服务器的 GPU 内存中，本文称该 APP 未命中缓存，APP 计算请求将被定向到远端服务器执行，平均响应时间为常数 D_2。根据文献[4]，$D_2\gg D_1$，因此本文假设 $D_1=0$。

给定一个缓存决策向量 $\boldsymbol{L}=(l_1,l_2,\ldots,l_{|\boldsymbol{M}|})$，APP 是否命中的二元指示变量 I_i 计算如下：

$$I_i = \prod_{j \in \boldsymbol{A}_i} l_j \tag{4-1}$$

进一步，APP 的平均响应时间如下：

$$\sum_{i \in \boldsymbol{T}} \left(I_i D_1 + (1 - I_i) D_2 \right) p_i = D_2 \left(1 - \sum_{i \in \boldsymbol{T}} I_i p_i \right) \tag{4-2}$$

由于 D_2 是常数，因此根据公式(4-2)，最小化 APP 平均响应时间等同于最大化加权 APP 的缓存命中率（每个 APP 的权重为该 APP 的流行度）。

结合公式(4-1)，建立最优化模型如下：

$$\max_{\boldsymbol{L}} \sum_{i \in \boldsymbol{T}} p_i I_i \tag{4-3}$$

限制条件如下：

$$I_i = \prod_{j \in \boldsymbol{A}_i} l_j, i \in \boldsymbol{T} \tag{4-4}$$

$$\sum_{i \in \boldsymbol{M}} l_i \le c \tag{4-5}$$

$$l_i \in \{0,1\}, i \in \boldsymbol{M} \tag{4-6}$$

其中，约束(4-5)确保缓存在 GPU 内存中的 DNN 模型的数目不超过内存最大可容纳的 DNN 模型的数目。

最优化模型的目标是找到一组缓存决策向量 $\boldsymbol{L} = (l_1, l_2, \ldots, l_{|\boldsymbol{M}|})$，以最大化系统 APP 的加权命中率。由于 $\boldsymbol{L}$ 中元素的取值是二元变量，因此该最优化模型是一个整数规划问题，找到最优解需要遍历 $\boldsymbol{L}$ 所有可能的取值，计算复杂度为 $O(2^{|\boldsymbol{M}|})$。当系统中 APP 和 DNN 模型的数目增加，该问题的求解将会变得十分复杂。因此，本文接下来提出一种启发式算法来求解该最优化模型。

4.3.3 启发式算法

针对上述人面向工智能应用的缓存优化模型，本小节提出了基于边际价值的启发式算法（greedy application marginal value, GAMV），求解缓存决策向量 $\boldsymbol{L}$，以尽可能提高系统 APP 的缓存命中率，降低 APP 响应时间。

GAMV 算法的主要思想是利用贪婪算法，迭代选择最有价值的 APP，并缓存其相对应的 DNN 模型。同时为了去除缓存冗余，每个 DNN 模型的副本只存储一次。重复迭代步骤直到剩余的 GPU 内存空间再也无法满足更多的 APP 的计算。

该算法的关键是如何定义一个 APP 的价值。APP 的价值与 APP 的流行度有关。具体来说，一个 APP 的流行度越高，请求该 APP 的计算的用户数目越多，该 APP 价值越高。

APP 的价值还与当前 GPU 内存的状态有关。若一个 APP 所需的 DNN 模型在

GPU 内存中存储的数目越多，那么只需要再缓存少量的该 APP 用到的 DNN 模型，就可使得该 APP 命中缓存。具体来说，给定某一迭代轮次，当前 GPU 内存中已缓存部分 DNN 模型，当 APP i 的部分 DNN 模型 $\boldsymbol{A}_i^C$ 在之前的迭代过程中被缓存，那么在当前迭代轮次中只需缓存 APP i 所需的未被缓存的那部分 DNN 模型，便可满足用户对 APP i 的计算请求，缓存命率中此时增加 p_i。

综上所述，需要综合考虑迭代过程中 APP 所需缓存空间的动态性和 APP 的流行度来确定 APP 的价值。为此，给定一个迭代轮次，已知当前决策向量 $\boldsymbol{L}$，待缓存的 APP i 的价值 v_i 计算方式如公式(4-7)所示。

$$v_i = \frac{p_i}{|\boldsymbol{A}_i| - |\boldsymbol{A}_i^C|} \tag{4-7}$$

其中 p_i 为 APP i 的流行度，$|\boldsymbol{A}_i|$ 为 APP i 关联的 DNN 模型数目，$|\boldsymbol{A}_i^C|$ 为 APP i 关联的已被缓存的模型数目，对于 $\forall i \in \boldsymbol{T}$，$|\boldsymbol{A}_i^C|$ 的值可通过公式(4-8)求得。因此，$|\boldsymbol{A}_i| - |\boldsymbol{A}_i^C|$ 为当前要满足 APP i 的计算，还所需缓存的 DNN 模型的数目。根据公式(4-7)，一个 APP 的价值可理解为缓存该 APP 每单位缓存容量带来的缓存命中率的提升。

$$|\boldsymbol{A}_i^C| = \sum_{k \in \boldsymbol{A}_j} l_k \tag{4-8}$$

GAMV 算法的具体过程如下：首先当缓存为空时，缓存决策向量 $\boldsymbol{L}$ 被初始化为零向量，集合 $\boldsymbol{A}_i^C$ 被初始化为空集。然后开始进入迭代，算法使用公式(4-7)计算当前所有待缓存的 APP 的价值 $v_i, (\forall i \in \boldsymbol{T})$；然后选择能够被 GPU 容纳的条件下，价值最高的 APP 作为目标 APP i^*，缓存 APP i^* 所关联且未被缓存的所有 DNN 模型，将这些 DNN 模型对应的缓存决策变量的值置为 1，然后更新 GPU 内存大小，同时将本轮缓存的目标 APP i^* 从待缓存的集合当中剔除。当 APP 全部命中，或者剩余可用 GPU 内存不足以满足任何一个 APP 的计算时，迭代结束，算法输出最终的缓存决策变量 $\boldsymbol{L}$。算法伪代码如表 4-2 所示。

算法输入的参数为 APP 的集合 $\boldsymbol{T}$、每个 APP 的流行度 $\{p_i \mid i \in \boldsymbol{T}\}$ 和每个 APP 所用到的 DNN 模型集合 $\{\boldsymbol{A}_i \mid i \in \boldsymbol{T}\}$。在该算法中，步骤 1 将缓存决策向量初始化为零向量。步骤 2 至步骤 9 是贪婪迭代的过程，其中步骤 3 计算本轮迭代每个 APP 所需的 GPU 内存空间，步骤 4 剔除所需缓存容量大于 GPU 内存可用空间的 APP，步骤 5 计算每个 APP 的价值，并挑选价值最大的 APP 作为目标 APP。步骤 6 缓存目标 APP 关联的 DNN 模型，步骤 7 更新 GPU 内存容量，步骤 8 将本轮缓存的目标 APP 从待缓存 APP 集合中剔除。

算法 GAMV 是一个迭代算法，在每一次迭代过程中，复杂度最高的步骤为步骤 3，最多需要计算 $|\boldsymbol{T}|$ 个 APP，每个 APP 需遍历 $|\boldsymbol{M}|$ 个模型，所以该步骤的计算复杂度最高为 $O(|\boldsymbol{T}| \times |\boldsymbol{M}|)$。同时，由于有 $|\boldsymbol{T}|$ 个 APP，迭代过程最多执行 $|\boldsymbol{T}|$ 次，

因此总的算法复杂度为$O(|\boldsymbol{T}|^2 \times |\boldsymbol{M}|)$。

表 4-2 面向人工智能应用的缓存算法
Table 4-2 Caching algorithms designed for artificial intelligence applications

Algorithm 2

Input: $\boldsymbol{T}$, $\{p_i \mid i \in \boldsymbol{T}\}$, $\{\boldsymbol{A}_i \mid i \in \boldsymbol{T}\}$

Output: $\boldsymbol{L}$

$\boldsymbol{L} \leftarrow (0,0,...,0)$

While $\boldsymbol{T} \neq \varnothing$

$r_i \leftarrow |\boldsymbol{A}_i| - \sum_{k \in \boldsymbol{A}_j} l_k, \forall i \in \boldsymbol{T}$

$\boldsymbol{T} \leftarrow \{i \in \boldsymbol{T} \mid r_i \le c\}$

$i^* \leftarrow \arg\max_{i \in \boldsymbol{T}} \left(\frac{p_i}{r_i} \right)$

$l_j \leftarrow 1, \forall j \in \boldsymbol{A}_{i^*}$

$c \leftarrow c - r_{i^*}$

$\boldsymbol{T} \leftarrow \boldsymbol{T} \setminus \{r_{i^*}\}$

End While

4.4 实验结果与分析

本节对提出的 GAMV 算法进行了仿真实验以评估其性能，本章首先介绍了实验的设置和对比参照，然后给出了仿真结果和具体的实验分析。

4.4.1 实验设置与实验对比参照

为测试算法性能，本文设置了 30 个 APP 与 100 个 DNN 模型，在实验过程中随机生成 APP 与 DNN 之间的关联关系。具体的实验场景的产生方法如下：首先，本文调研了 20 篇相关文献的人工智能应用使用的 DNN 模型的数目，发现人工智能应用使用的 DNN 模型的数目分布在$[2, 6]$的区间。因此，除非特别指定，本文假设 APP 关联 DNN 的数目区间[2,6]上的均匀分布。在给定了每个 APP 关联的 DNN 数目的产生方法之后，本文的实验从 100 个 DNN 模型中随机选择相应数目的 DNN 模型关联到每个 APP。最后，本文默认 APP 的数目为 30，设这 30 个 APP 的流行

度服从参数为α的 Zipf 分布，每个 APP 的流行度计算参考公式(3-12)。

由于 APP 与 DNN 模型之间的关联关系的生成具有随机性，在每种实验设置中，本文对每种算法重复仿真 100 次，并取这 100 次缓存命中率的平均值作为该算法的性能指标，该指标记为平均缓存命中率。

本文选取如下 4 种传统的缓存算法作为 GAMV 的对比算法，分别是：

(1) 带冗余随机缓存算法（记为 RND-APP-CACHE）。该算法重复如下步骤，直到没有可用的 GPU 内存空间：随机选择一个 APP，缓存该 APP 关联的所有 DNN 模型。如果一个 DNN 模型被多个 APP 使用，这个 DNN 模型会被存储多次。这种方法忽略了 DNN 模型可能被不同的 APP 使用，而把一个 APP 关联的所有 DNN 模型看作一个独立的数据文件，而没有消除 GPU 内存中的冗余。

(2) 不带冗余的随机缓存算法（记为 RND-APP-DNN-CACHE）。该算法步骤同上，但是对于每个 DNN 模型，最多只缓存一个副本，以消除 GPU 内存中的冗余。

(3) 带冗余的基于应用流行度的缓存算法（记为 POP-APP-CACHE）。该算法类似于 RND-APP-CACHE，不同的是，该算法依次选择最流行且能够被 GPU 内存容纳的 APP 缓存。

(4) 不带冗余的基于应用流行度缓存算法（记为 POP-APP-DNN-CACHE）。该算法考虑到 DNN 模型可能在不同的 APP 之间被重用，每次选择最流行的 APP 并缓存其关联的 DNN 模型，每个模型只缓存一个副本，直到没有剩余的 GPU 空间。

(5) 基于 DNN 模型流行度缓存算法（记为 POP-DNN-CACHE）。该算法选择流行度最高的 DNN 模型进行缓存，直到没有剩余的 GPU 内存。其中，一个 DNN 模型的流行度计算为与该模型关联所有 APP 的流行度之和。

4.4.2 性能分析

本小节分别通过仿真分析了 GPU 内存大小、APP 流行度和 APP 关联 DNN 模型的数目对缓存算法性能的影响。

(1) GPU 内存大小对算法性能的影响。

为了评估 GPU 内存大小对算法性能的影响，本实验设置 APP 的流行度参数α为 0.2，APP 关联的模型数目服从参数为[2, 6]的均匀分布，以步长为 10 从 10 到 100 改变 GPU 内存的大小c。实验结果如图 4-4 所示，图中横轴为 GPU 的大小，纵轴为平均缓存命中率。

从图 4-4 可以看出，所有算法的性能随着c的增加而呈上升趋势，这是由于c增大，GPU 内存中可容纳的 DNN 模型的数目也随之增加，可以满足更多 APP 的计算需求。当c接近 100 时，所有的 DNN 模型均被缓存，此时 GAMV、RND-APP-

DNN-CACHE、POP-APP-DNN-CACHE 和 POP-DNN-CACHE 的平均命中率都接近于 1，而 RND-APP-CACHE 和 POP-APP-CACHE 的平均缓存命中率却无法达到 1。这是由于后者认为每个 APP 所关联的 DNN 模型集合是独立的，在缓存时会缓存 DNN 模型的多个副本，因此会导致缓存冗余，降低缓存利用率。

在这几种对比算法中，随着 c 增加，POP-DNN-CACHE 的性能提升也逐渐增加，逐渐优于随机算法（RND-APP-CACHE 和 RND-APP-DNN-CACHE）。这是由于最流行的 DNN 模型集合可能被多个不同的 APP 使用，当较 c 小时，缓存的最流行的 DNN 模型集合不能完整满足大多数 APP 的需求，此时缓存命中率较低。而随着 c 增加，所缓存的最流行的 DNN 模型的集合能够满足的 APP 数目增加，且这些 APP 的流行度较高，因此 POP-DNN-CACHE 性能也随之提高。

进一步，从图 4-4 可知，本文提出的 GAMV 算法性能明显优于其他算法，相较于对比算法中性能最好的算法（POP-APP-DNN-CACHE），GAMV 的平均缓存命中率最好可提升 9.14%，这得益于 GAMV 考虑了 APP 与 DNN 模型之间的关联关系，并将该关系引入到 APP 的价值计算当中，因此可获得更好的性能。

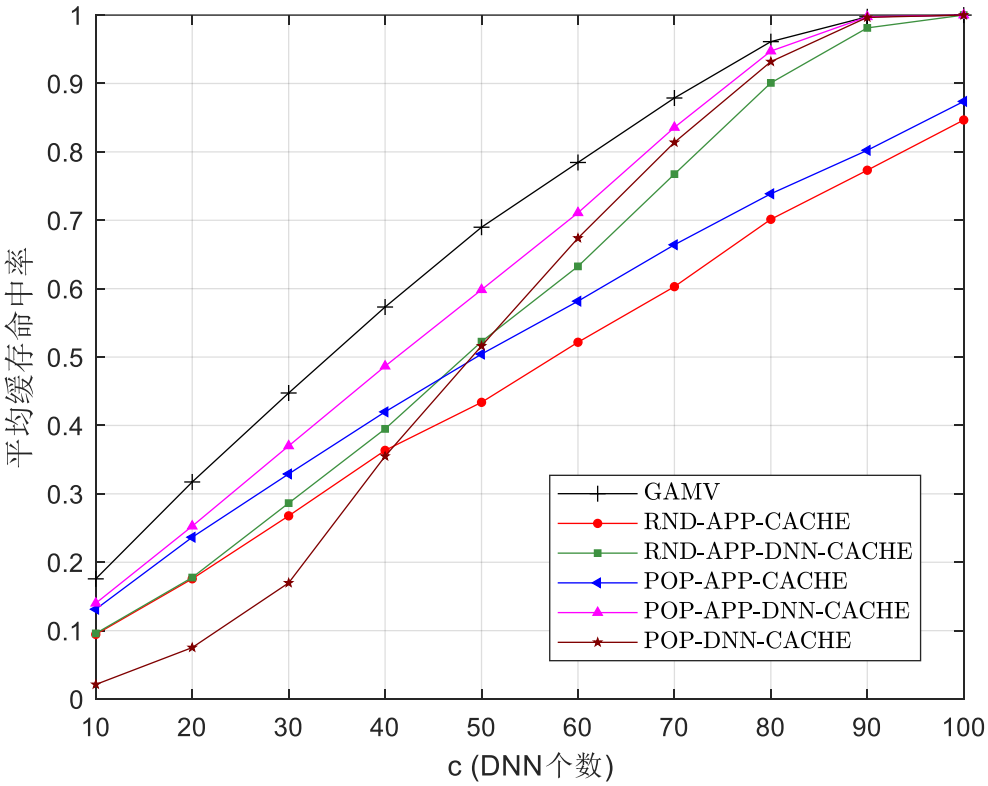


图 4-4 GPU 内存大小 c 对算法性能的影响

Figure 4-4 Impact of GPU memory size c on caching performance

(2) APP 流行度参数 α 对算法性能的影响。

为了评估 APP 流行度参数 α 对算法性能的影响，本实验设置缓 GPU 内存大小

为 50，APP 关联的模型数目服从参数为[2, 6]的均匀分布，以步长为 0.1 从 0.1 到 1 改变 APP 流行度参数α的值。实验结果如图 4-5 所示，图中横轴为α的值，纵轴为平均缓存命中率。

从图 4-5 中可以看出，本文提出的 GAMV 算法的性能明显优于其他算法。同时，6 种算法又可分为两组，一组是不考虑流行度的随机算法 RND-APP-CACHE 和 RND-APP-DNN-CACHE；另一组是考虑了流行度的其余 4 种算法，它们直接或间接地利用流行度来缓存 APP 所需 DNN 模型。

随着α的增加，考虑了流行度的后一组算法的平均缓存命中率也随之增加，而不考虑流行度的前一组随机算法性能维持在一个相对稳定的值附近。这是由于α的增加使得更少一部分 APP 被更多的用户请求，因此只需缓存这部分 APP 所需的 DNN 模型便可满足大部分的用户请求。而考虑了流行度的那组算法能够以一定的规则选择流行度更高的 APP 进行缓存，因此提高了平均缓存命中率。由图 4-5 进一步可知，本文的提出的 GAMV 优于其他算法，这得益于 GAMV 在缓存过程中不仅考虑了 APP 的流行度，而且也考虑了 APP 与 DNN 之间的关联关系。

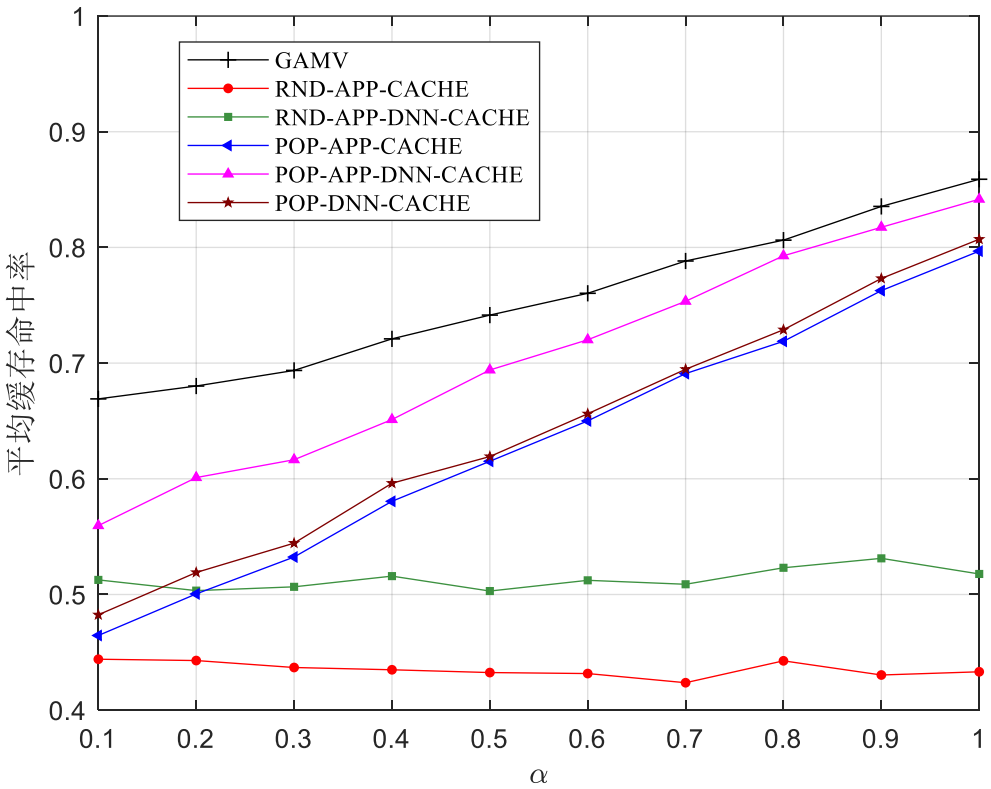


图 4-5　流行度参数α对算法性能的影响

Figure 4-5　Impact of Zipf parameter α on caching performance

(3)　APP 关联模型数目对算法性能的影响。

为了评估 APP 的性能升级及规模增加后对缓存性能带来的影响，我们分析观

察了 APP 关联的 DNN 模型数目对算法性能的影响。本实验设置 APP 的流行度参数α为 0.2，设置 GPU 内存空间c的大小为 50，APP 关联的模型数目的分布区间$[a,b]$从[1, 4]增加到[10, 13]，在增加过程中，a和b每次均增加 1。实验结果如图 4-6 所示，图中横轴为区间$[a,b]$的均值，纵轴为平均缓存命中率。

从图 4-6 中可以看出，随着 APP 关联的 DNN 模型的平均数目增加，所有算法的性能呈现下降趋势，这是由于以下两个方面导致的：一方面，a和b的增加使得每个 APP 关联的 DNN 模型数目随之增加，同时所有 APP 关联的不同的 DNN 模型的数目也随之增加；另一方面，GPU 内存空间有限，只能缓存固定数目的 DNN 模型。这就导致用户请求的 APP 所需的 DNN 模型被同时缓存的概率降低，能够被完整缓存的 APP 的数目减少，缓存命中率降低。

即便如此，本文所提出的 GAMV 算法依旧具有最好的性能。这是因为 GAMV 能够在缓存过程中综合考虑 APP 的流行度和缓存 APP 所需的空间来选择最有价值的 APP，并缓存这些 APP 所需的 DNN 模型，每个 DNN 模型只缓存一个副本，从而有效地提高了缓存的性能。结合实验(1)和实验(2)，相较于对比算法中性能最好的 POP-APP-DNN-CACHE，GAMV 多数情况下可提升平均缓存命中率 10%左右。

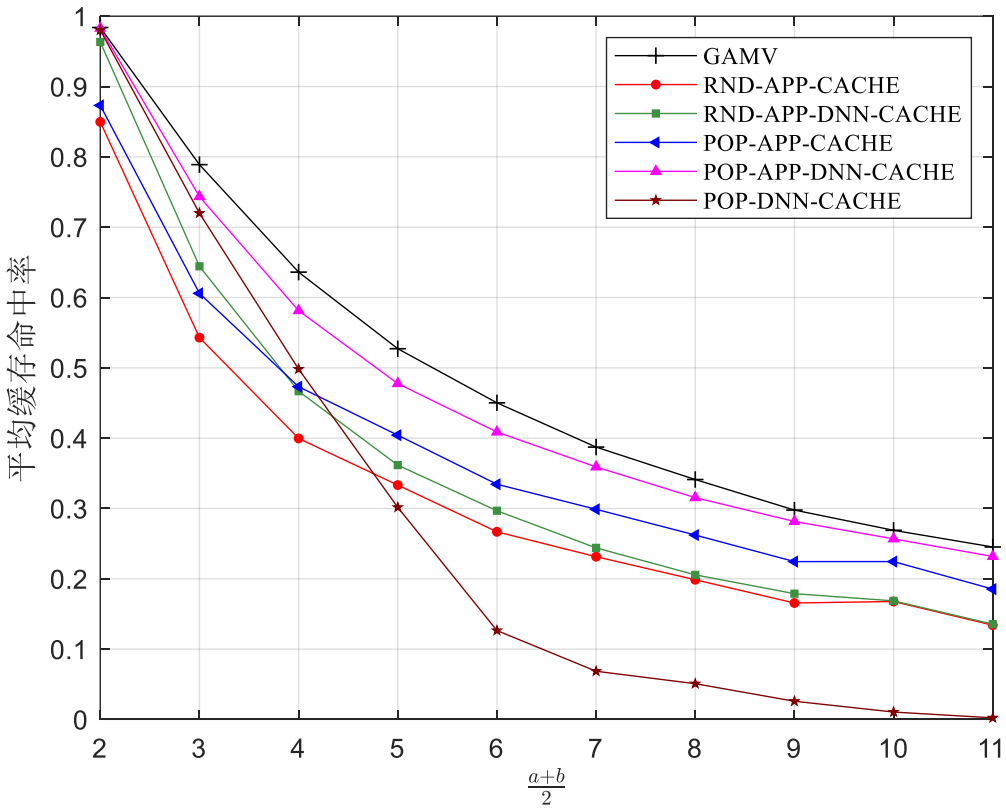


图 4-6 APP 关联模型数目对算法性能的影响

Figure 4-6 Impact of the average number of DNN models associated with an application on caching performance

4.5 本章小结

本章提出了一种面向人工智能应用的缓存算法。该算法以充分利用边缘缓存服务器有限的 GPU 内存资源、尽可能降低总体应用响应时间为目标，缓存人工智能应用关联的 DNN 模型。本章的研究首先阐述了人工智能应用的计算业务的研究背景，然后介绍了 CDN 人工智能应用的计算业务的特点和缓存的基本思想，在此基础上，提出了面向人工智能应用的缓存的最优化模型，来缓存人工智能应用关联的 DNN 模型，在 GPU 内存空间有限的情况下，尽可能提高缓存性能，并给出一种基于贪婪迭代的启发式算法 GAMV 求解最优化模型。本文的数值结果表明，在不同的实验设置下，本文提出的 GAMV 算法均有效提升了缓存性能。

5 结论

5.1 本文总结

CDN 是实现互联网数据分发的主要手段。CDN 的主要业务包括两大类。一类是目前的视频文件的分发。第二类是新兴的人工智能应用的计算。

由于 CDN 业务量的增长速度远高于网络扩容速度，CDN 性能的提升面临着资源瓶颈。CDN 资源瓶颈主要来自于：边缘缓存服务器的带宽资源和缓存资源。本文根据 CDN 不同的业务的特征，和不同业务对 CDN 资源的需求，设计了相应的资源调度机制。本文的研究主要包括两项内容。

第一项是设计了视频内容感知的负载均衡算法。针对 CDN 视频业务的带宽资源瓶颈，本文提出了视频内容感知的 CDN 负载均衡联合优化模型，该联合优化模型根据视频不同内容视频，其用户体验随比特率变化的函数是不同的这一特点，联合规划每个边缘缓存服务器提供的视频组合以及每个视频画面的比特率，优化用户总体体验。由于该最优化模型是一个混合整合规划模型，本文进一步提出了求解该最优化模型的迭代的启发式算法，该算法首先将所有 CDN 服务器当作一个虚拟的理想服务器，以真实服务器带宽总和为约束，迭代地求得每个视频的最优比特率；然后按求得的最优比特率，将视频迭代地分配到每个 CDN 边缘缓存服务器，每次选择价值最高的视频分配到剩余带宽最多的服务器。实验结果显示，本文提出的算法能够有效提升用户总体体验。

第二项是设计了面向人工智能应用的缓存算法。针对 CDN 人工智能应用的 GPU 内存资源瓶颈，本文提出了面向人工智能应用的缓存优化模型，该优化模型将最小化响应时间的优化问题转化为最大化缓存命中率的优化问题，选择一组合适的 DNN 模型并缓存到边缘缓存服务器有限的 GPU 内存中，以最小化人工智能应用整体的响应时间。该最优化模型背后的思想是：一个人工智能应用可能并行用到多个 DNN 模型，一个 DNN 模型可能被多个人工智能使用；同时由于从硬盘中加载或者将请求定向到远端服务器会引入较大的时延，因此只有当一个应用所需的全部 DNN 模型缓存在 GPU 内存中时，应用才能获得较低的响应时间。本文设计了一个贪婪式的启发式算法求解该最优化模型，该算法迭代地选择最有价值的人工智能应用关联的 DNN 模型缓存到 GPU 内存中。实验结果表明本文所提出的缓存算法能够有效提升缓存性能。

5.2 存在问题与展望

本文存在的问题和进一步的工作如下：

(1) 对于视频内容感知的负载均衡算法，主要有两方面问题。首先，本文的优化模型是建立在小规模的 CDN 网络架构上的，用户从各服务器获取视频的时延差可以忽略，如果网络规模进一步扩大，用户以跨国甚至跨洲的形式获取视频，此时不同用户获取视频的时延差不可忽略，需要进一步完善模型；其次，本文假设用户请求视频序列可提前精确预知，实际上如何精确预测用户行为需要精巧设计，本文将在该方面展开进一步的工作。

(2) 对于面向人工智能应用的缓存算法，本文的研究假设人工智能应用的流行度是已知的，然而目前相关研究中，对于人工智能应用流行度变化的模式和规律的研究还是空白。如何通过实际数据建立人工智能应用流行度变化模型，以及预测不同服务器范围内的应用流行度变化，对于精准提升用户体验具有重要意义，可能是未来一个研究方向，本文的后续工作将继续根据真实的探测数据建立人工智能应用流行度变化的数学模型。

参考文献

[1] Medagliani P, Paris S, Leguay J, et al. Overlay routing for fast video transfers in CDN[C].// 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), 2017: 531-536.
[2] Salahuddin M A, Sahoo J, Glitho R, et al. A Survey on Content Placement Algorithms for Cloud-Based Content Delivery Networks[J]. IEEE Access, 2018, 6: 91-114.
[3] Cisco. Cisco Annual Internet Report (2018–2023) White Paper[EB/OL]. [2021-01-29]. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.
[4] Guo T, Walls R J, Ogden S S. EdgeServe: efficient deep learning model caching at the edge[C].// Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, 2019: 313-315.
[5] Adler M, Sitaraman R K, Venkataramani H. Algorithms for optimizing the bandwidth cost of content delivery[J]. Computer Networks, 2011, 55(18): 4007-4020.
[6] 赵红娜. 基于视频转码和视频推荐的缓存设计[D]. 北京交通大学, 2020.
[7] Aditya P, Zhao M, Lin Y, et al. Reliable client accounting for hybrid content-distribution networks[J], 2012.
[8] Ghosh B, Muthukrishnan S. Dynamic load balancing by random matchings[J]. Journal of computer system sciences, 1996, 53(3): 357-370.
[9] Bienkowski M, Korzeniowski M, Auf Der Heide F M. Dynamic load balancing in distributed hash tables[C].// International Workshop on Peer-to-Peer Systems, 2005: 217-225.
[10] Dahlin M. Interpreting stale load information[J]. IEEE Transactions on parallel distributed systems, 2000, 11(10): 1033-1047.
[11] Mitzenmacher M. The power of two choices in randomized load balancing[J]. IEEE Transactions on Parallel Distributed Systems, 2001, 12(10): 1094-1104.
[12] Cece F, Formicola V, Oliviero F, et al. An extended ns-2 for validation of load balancing algorithms in content delivery networks[C].// Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, 2010: 1-6.
[13] Manfredi S, Oliviero F, Romano S P. Distributed management for load balancing in content delivery networks[C].// 2010 IEEE Globecom Workshops, 2010: 579-583.
[14] Tran H A, Hoceini S, Mellouk A, et al. QoE-based server selection for content distribution networks[J]. IEEE Transactions on Computers, 2013, 63(11): 2803-2815.
[15] Qin F, Zhao Z, Zhang H. Optimizing routing and server selection in intelligent SDN-based CDN[C].// 2016 8th International Conference on Wireless Communications & Signal Processing (WCSP), 2016: 1-5.
[16] Ray D, Kosaian J, Rashmi K, et al. Vantage: optimizing video upload for time-shifted viewing of social live streams[C].// Proceedings of the ACM Special Interest Group on Data Communication, 2019: 380-393.
[17] Qin Y, Hao S, Pattipati K R, et al. ABR streaming of VBR-encoded videos: characterization, challenges, and solutions[C].// Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies, 2018: 366-378.
[18] Zhang M, Zhang F, Lane N D, et al. Deep Learning in the Era of Edge Computing: Challenges and Opportunities[J]. Fog Computing: Theory Practice, 2020: 67-78.
[19] Taylor B, Marco V S, Wolff W, et al. Adaptive deep learning model selection on embedded systems[J]. ACM SIGPLAN Notices, 2018, 53(6): 31-43.
[20] Gilman G R, Ogden S S, Walls R J, et al. Challenges and opportunities of dnn model execution caching[C].// Proceedings of the Workshop on Distributed Infrastructures for Deep Learning, 2019: 7-12.
[21] Berger D S, Sitaraman R K, Harchol-Balter M. Adaptsize: Orchestrating the hot object memory cache in a content delivery network[C].// 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), 2017: 483-498.
[22] Gill B S, Modha D S. SARC: Sequential Prefetching in Adaptive Replacement Cache[C].// USENIX Annual Technical Conference, General Track, 2005: 293-308.
[23] Yang J, Karimi R, Sæmundsson T, et al. Mithril: mining sporadic associations for cache

prefetching[C].// Proceedings of the 2017 Symposium on Cloud Computing, 2017: 66-79.
[24] Georgiev P, Bhattacharya S, Lane N D, et al. Low-resource multi-task audio sensing for mobile and embedded devices via shared deep neural network representations[C].// Proceedings of the ACM on Interactive, Mobile, Wearable, 2017: 1-19.
[25] Howard A G, Zhu M, Chen B, et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications[J]. arXiv preprint arXiv:.04861, 2017.
[26] Hoßfeld T, Moldovan C, Schwartz C. To each according to his needs: Dimensioning video buffer for specific user profiles and behavior[C].// 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2015: 1249-1254.
[27] Burger V, Zinner T, Dinh-Xuan L, et al. A generic approach to video buffer modeling using discrete-time analysis[J]. ACM Transactions on Multimedia Computing, Communications, and Applications, 2018, 14(2s): 1-23.
[28] Goswami K, Hariharan B, Ramachandran P, et al. Adaptive Multi-Resolution Encoding for ABR Streaming[C].// 2018 25th IEEE International Conference on Image Processing (ICIP), 2018: 1008-1012.
[29] Asan A, Robitza W, Mkwawa I-H, et al. Impact of video resolution changes on QoE for adaptive video streaming[C].// 2017 IEEE International Conference on Multimedia and Expo (ICME), 2017: 499-504.
[30] Yasnoff W A, Mui J K, Bacus J W. Error measures for scene segmentation[J]. Pattern recognition, 1977, 9(4): 217-231.
[31] Nasiri R M, Wang Z. Perceptual aliasing factors and the impact of frame rate on video quality[C].// 2017 IEEE International Conference on Image Processing (ICIP), 2017: 3475-3479.
[32] Tran T X, Pompili D. Adaptive bitrate video caching and processing in mobile-edge computing networks[J]. IEEE Transactions on Mobile Computing, 2018, 18(9): 1965-1978.
[33] Sara U, Akter M, Uddin M S. Image quality assessment through FSIM, SSIM, MSE and PSNR—a comparative study[J]. Journal of Computer Communications, 2019, 7(3): 8-18.
[34] Wang S, Zhang X, Zhang Y, et al. A survey on mobile edge networks: Convergence of computing, caching and communications[J]. IEEE Access, 2017, 5: 6757-6779.
[35] 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 & development in information retrieval, 2018: 965-968.
[36] 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.
[37] Chinchali S P, Cidon E, Pergament E, et al. Neural networks meet physical networks: Distributed inference between edge devices and the cloud[C].// Proceedings of the 17th ACM Workshop on Hot Topics in Networks, 2018: 50-56.
[38] Samal P, Mishra P. Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing[J]. International Journal of computer science Information Technologies, 2013, 4(3): 416-419.
[39] Devi D C, Uthariaraj V R. Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks[J]. The scientific world journal, 2016, 2016.
[40] Deng J, Tyson G, Cuadrado F, et al. Internet scale user-generated live video streaming: The Twitch case[C].// International Conference on Passive and Active Network Measurement, 2017: 60-71.
[41] Liu J, Yang Q, Simon G. Congestion avoidance and load balancing in content placement and request redirection for mobile CDN[J]. IEEE/ACM Transactions on Networking, 2018, 26(2): 851-863.
[42] He H, Feng Y, Li Z, et al. Dynamic load balancing technology for cloud-oriented CDN[J]. Computer Science and Information Systems, 2015, 12(2): 765-786.
[43] Saengarunwong A, Sanguankotchakorn T. A Two-Step Server Selection in Hybrid CDN-P2P Mesh-based for Video-on-Demand Streaming[C].// 2018 International Conference on Information and Communication Technology Convergence (ICTC), 2018: 499-504.
[44] Yang W, Hu Y, Ding L, et al. Viewer-Oriented CDN Scheduling on Crowdsourced Live Video Stream[C].// 2019 IEEE 2nd International Conference on Electronics and Communication Engineering (ICECE), 2019: 112-117.
[45] Chen F, Sitaraman R K, Torres M. End-user mapping: Next generation request routing for content delivery[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 167-181.
[46] Tran H-A, Souihi S, Tran D, et al. Mabrese: A new server selection method for smart SDN-based CDN architecture[J]. IEEE Communications Letters, 2019, 23(6): 1012-1015.
[47] Budhkar S, Tamarapalli V. An overlay management strategy to improve QoS in CDN-P2P live

streaming systems[J]. Peer-to-Peer Networking Applications, 2020, 13(1): 190-206.
[48] Yuan S, Lin T, Bai S, et al. User-oriented QoE-driven server selection for multimedia service provisioning in content distribution networks[C].// 2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015: 2077-2082.
[49] Tomić D, Žagar D. Dynamic Server Selection by Using a Client Side Composite DNS-Metric[J]. Tehnički vjesnik, 2018, 25(4): 1080-1087.
[50] Wang T, Song J, Song M. A three-stage global optimization method for server selection in content delivery networks[J]. Soft Computing, 2017, 21(2): 467-475.
[51] Torres R, Finamore A, Kim J R, et al. Dissecting video server selection strategies in the youtube cdn[C].// 2011 31st International Conference on Distributed Computing Systems, 2011: 248-257.
[52] Han S, Mao H, Dally W. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding[J]. arXiv preprint arXiv:.00149, 2015.
[53] Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to algorithms[M]. MIT press, 2009.
[54] Glover F, Laguna M: Tabu search, Handbook of combinatorial optimization: Springer, 1998: 2093-2229.
[55] Kirkpatrick S, Gelatt C D, Vecchi M P J S. Optimization by simulated annealing[J], 1983, 220(4598): 671-680.
[56] Pallis G, Vakali A. Insight and Perspectives for CONTENT DELIVERY NETWORKS[J]. Communications of the ACM, 2006, 49(1): p.101-106.
[57] Dykes S G, Robbins K A, Jeffery C L. An empirical evaluation of client-side server selection algorithms[C].// Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064), 2000: 1361-1370.
[58] ffmpeg[EB/OL]. [2021-01-30]. https://www.ffmpeg.org/.
[59] MSU Video Quality Measurement Tool[EB/OL]. [2021-01-30]. http://www.compression.ru/video/quality_measure/video_measurement_tool.html.
[60] 视频大数据报告[EB/OL]. [2021-05-16]. https://www-file.huawei.com/-/media/corporate/pdf/ilab/30-cn.pdf.
[61] Baixeries J, Elvevåg B, Ferrer-I-Cancho R. The evolution of the exponent of Zipf's law in language ontogeny[J]. PloS One, 2013, 8(3): e53227.
[62] Deep Learning Neural Network[EB/OL]. [2021-01-31]. https://dfan.engineering.asu.edu/deep-learning-neural-network/.
[63] Zhou Z-H. Learnware: on the future of machine learning[J]. Frontiers Comput. Sci., 2016, 10(4): 589-590.

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

崔子琦 女 1996 年 6 月生

2018 年 9 月至 2021 年 6 月	北京交通大学	通信与信息系统	获硕士学位
2014 年 9 月至 2018 年 6 月	中北大学	通信工程	获学士学位

二、发表论文

[1] Cui Z, Zhao Y, Li C, et al. Content-Aware Load Balancing in CDN Network[C]//2020 IEEE 6th International Conference on Computer and Communications (ICCC). IEEE, 2020: 88-93.

[2] Cui Z, Zhao Y, Li C, et al. An Adaptive Authentication Based on Reinforcement Learning[C]//2019 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW). IEEE, 2019: 1-2.

[3] 崔子琦，邢晓曼，基于心冲击信号的血压监测技术研究进展[J].医学新知, 2021, 31(02): 145-154.

三、参与科研项目

［1］国家自然科学基金：基于熵理论的信息匹配网络测量与建模

［2］国家重点研发计划：零/低负荷睡眠监测与调控系统的研发及基于临床数据模型的示范应用

四、申请专利

［1］杜磊,崔子琦等. 呼吸机压力控制系统及方法[P]. CN111803770A.

［2］杜磊,崔子琦等. 睡眠呼吸结果显示系统及方法[P]. CN111803030A.

［3］杜磊,崔子琦等. 睡眠采集调控系统及方法[P]. CN111700590A

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： 年 月 日

学位论文数据集

表 1.1： 数据集页

<table>
<tr><td>关键词*</td><td>密级*</td><td>中图分类号</td><td>UDC</td><td>论文资助</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">学位授予单位名称*</td><td>学位授予单位代码*</td><td>学位类别*</td><td>学位级别*</td></tr>
<tr><td colspan="2">北京交通大学</td><td>10004</td><td></td><td></td></tr>
<tr><td colspan="2">论文题名*</td><td colspan="2">并列题名</td><td>论文语种*</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td></td></tr>
<tr><td>作者姓名*</td><td colspan="2"></td><td>学号*</td><td></td></tr>
<tr><td colspan="2">培养单位名称*</td><td>培养单位代码*</td><td>培养单位地址</td><td>邮编</td></tr>
<tr><td colspan="2">北京交通大学</td><td>10004</td><td>北京市海淀区西直门外上园村 3 号</td><td>100044</td></tr>
<tr><td colspan="2">学科专业*</td><td>研究方向*</td><td>学制*</td><td>学位授予年*</td></tr>
<tr><td colspan="2"></td><td></td><td></td><td></td></tr>
<tr><td>论文提交日期*</td><td colspan="4"></td></tr>
<tr><td>导师姓名*</td><td colspan="2"></td><td>职称*</td><td></td></tr>
<tr><td>评阅人</td><td colspan="2">答辩委员会主席*</td><td colspan="2">答辩委员会成员</td></tr>
<tr><td></td><td colspan="2"></td><td colspan="2"></td></tr>
<tr><td colspan="5">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</td></tr>
<tr><td colspan="2">电子版论文出版（发布）者</td><td>电子版论文出版（发布）地</td><td></td><td>权限声明</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td></td></tr>
<tr><td>论文总页数*</td><td colspan="4"></td></tr>
<tr><td colspan="5">共 33 项，其中带*为必填数据，为 21 项。</td></tr>
</table>