计 算 机 研 究 与 发 展
Journal of Computer Research and Development

DOI: 10.7544/issn1000-1239.202440403
61(11): 2793−2805, 2024

# 基于连续学习的视频物联网任务需求理解与调度方法

徐冬竹[1,2] 周安福[1,2] 马华东[1,2] 张 园[3]

[1](北京邮电大学计算机学院(国家示范性软件学院) 北京 100876)

[2](网络与交换技术全国重点实验室(北京邮电大学) 北京 100876)

[3](中国电信集团有限公司 上海 200125)

(xudongzhu@bupt.edu.cn)

# Continuous Learning-Based Task Demand Understanding and Scheduling Method for Video Internet of Things

Xu Dongzhu[1,2], Zhou Anfu[1,2], Ma Huadong[1,2], and Zhang Yuan[3]

[1](*School of Computer Science* (*National Pilot Software Engineering School*), *Beijing University of Posts and Telecommunications*, *Beijing* 100876)

[2](*State Key Laboratory of Networking and Switching Technology* (*Beijing University of Posts and Telecommunications*), *Beijing* 100876)

[3](*China Telecommunications Corporation*, *Shanghai* 200125)

**Abstract** Efficient scheduling between cloud-network resources and video tasks is crucial for the performance of video Internet of things (VIoT) applications. However, the current scheduling algorithms used in operational VIoT systems are insufficiently adaptable to differentiated task demands and highly dynamic changes in cloud-network resources, resulting in poor performance of VIoT applications. To overcome the aforementioned problem, we propose a continuous learning-based task demand understanding and scheduling method for VIoT (CLTUS) Unlike traditional heuristic or machine learning-driven scheduling algorithms, CLTUS integrates the continuous learning into the matching between cloud-network resources and video task demands. Specifically, it first employs a continuous learning framework to accurately comprehend various video task demands. Subsequently, based on the dependency relationships among video tasks, it achieves an optimal match between tasks and servers, thereby refining the scheduling of cloud-network resources. Finally, the proposed method is deployed on a software-defined VIoT experimental platform. Compared with conventional methods, CLTUS not only improves the average processing efficiency of video tasks by 127.73% but also increases the balanced utilization rate of cloud-network resources to 67.2% on average, effectively improving the performance of VIoT applications.

**Key words** video Internet of things; continuous learning; task demand understanding; resource scheduling; operational network measurement

**摘 要** 云网资源与视频任务的高效调度是保障视频物联网(video Internet of things, VIoT)应用性能的关键.然而,目前运营化VIoT所用调度算法对差异化的任务需求和高度动态的云网资源变化适应能力不足,

**收稿日期**: 2024-05-31; **修回日期**: 2024-08-05

**基金项目**: 国家自然科学基金创新研究群体项目(61921003); 国家自然科学基金青年科学基金项目(6240072128); 博士后创新人才支持计划项目(BX20240046)

This work was supported by the Innovation Research Group Project of the National Natural Science Foundation of China (61921003) the National Natural Science Foundation of China for Young Scientists (6240072128), and the China National Postdoctoral Program for Innovative Talents (BX20240046).

**通信作者**: 马华东(mhd@bupt.edu.cn)

导致 VIoT 应用性能不佳. 针对上述问题,提出了一种基于连续学习的视频物联网任务需求理解与调度方法(continuous learning-based task demand understanding and scheduling method for VIoT, CLTUS). 与传统启发式或机器学习驱动的调度算法不同,将连续学习引入云网资源与视频任务需求的匹配中. 首先,基于通用的连续学习框架实现各类视频任务需求的准确理解;其次,依据视频任务之间的需求依赖关系,实现任务与服务器的适配,以精细化调度云网资源;最后,将所提方法部署于软件定义的 VIoT 实验平台上. 与传统方法相比,CLTUS 不仅将视频任务的平均处理效率提高了 127.73%,还将云网资源利用均衡率提高至 67.2%,有效增强了 VIoT 应用性能.

**关键词** 视频物联网;连续学习;任务需求理解;资源调度;运营化网络测量

**中图法分类号** TP393.02

物联网、移动互联网、5G、人工智能等技术的兴起,激发了超高清视频直播[1]、远程视频安防[2]、云游戏、增强现实/虚拟现实/混合现实[3]等带宽需求大、时延要求高的新型视频物联网(video Internet of things, VIoT)业务大量涌现. 据权威机构预测[3],到 2029 年,VIoT 市场估值将突破 283.7 亿美元(复合年增长率高达 23.8%),极大促进了垂直产业的发展和人们生活方式的改变.

VIoT 应用具有 3 个新特点:1)亿级规模泛在互联. 据 Machina Research 报告[4],预计到 2025 年,全球视频网联设备总数将超过 270 亿. 2)百万级并发视频流高码率传输. 据爱立信预测[1],2025 年全球移动流量较 2019 年将增加 5 倍,其中视频流量占比将增至 76%. 3)ZB 级视频数据高效处理. 大规模视频流量亟需云化处理,并驱动中心云算力“下沉”到接近用户的网络边缘侧. 美国著名信息技术研究分析公司 Gartner 表明,到 2025 年,将有超过 75% 的业务被分流到网络边缘进行处理. 上述“大连接、大流量、大数据”特征给 VIoT 带来了新挑战——不仅要求运营化网络具备高效的传输能力,而且要求网络与云服务集群高效协作,以满足 VIoT 应用的差异化需求.

为了应对上述挑战,研究人员提出了边缘计算[2]和算力网络[5]技术,使算力(对云网平台存储、传输和计算能力的统称)分布式部署,并像电力一样“随开随用”. 该计算模式不仅借助云计算优势补齐了用户终端算力不足的短板,而且借助 5G 等网络降低了任务到中心云的传输开销,使数据业务与算力快速适配. 近年来,已有大量研究工作关注该计算模式的任务卸载与高效处理问题并提出相应的优化方案[2,3,5-10]. 然而,上述研究多在仿真或小规模(万级)视频流下验证,其性能在大规模运营化场景下仍存在视频帧响应时延过长、云网资源利用率过低的风险.

为此,本文首先在 VIoT 5G 运营环境开展应用性能测量(详见 1.2 节),以定位 VIoT 瓶颈. 测量结果表明,云网资源缺乏与差异化 VIoT 应用需求的适配能力. 从处理效率上看,尽管应用的端到端响应时延在亚秒级,但对于计算需求密集、时延敏感的应用(如车路协同、云控驾驶、视频直播等),平均视频帧响应时延仍大于 300 ms,甚至是相关应用容忍延迟(如 200 ms)的 1.5 倍! 此外,云网存储、传输和计算资源的平均利用率低于 60%,易造成云服务集群出现“单点瓶颈”问题,即某服务器资源使用过于密集而其他服务器使用空闲的情况.

进一步分析表明,VIoT 性能低的根本原因为运营化云网平台的存储、传输和计算资源调度分离,且所用调度算法无法适应任务需求和云网资源的动态变化. 一方面,运营化 VIoT 更关注“端—边”和“边—云”之间相邻实体间的协同,而忽略了广域范围内“端—边—云—网”资源的统一调度[11];另一方面,随着应用类型愈发复杂,VIoT 通常以微服务的模式进行任务批处理,即一段数据流通常被拆分成多个具有时序和资源需求依赖关系的任务集合,使得任务需求的划分更加精细,由此增加了资源调度的难度. 此外,云、网之间缺乏有效的交互机制,使得部署在云上的应用后端无法实时感知网络链路的波动,且网络无法适应上层应用的需求变化,从而造成资源与任务间的匹配程度差,进一步影响了应用性能.

针对上述问题,本文面向 VIoT 场景,提出了基于连续学习的视频物联网任务需求理解与调度方法(continuous learning-based task demand understanding and scheduling method for VIoT, CLTUS),旨在将“端—边—云—网”的存储、传输和计算资源统一封装,并在需求异质的视频任务持续涌现时,利用连续学习准确理解任务的资源需求,最后依据多任务需求之间的依赖关系和云网剩余资源的分布状态,将任务实时调度至适配的服务器中进行处理.

然而，面对 VIoT 高异构、强动态、算力资源泛分布的特点，实现 CLTUS 仍存在 2 个挑战：

1）视频任务需求理解难. 传统任务需求理解算法依据用户设置的服务等级协议（service level agreement, SLA）向云网调度器声明任务需求. 例如，用户在租用云服务器时，需手动预留所需的各类资源，如网络带宽、存储器容量和计算单元类型. 一旦资源预留不足，则会造成视频任务在被处理时由于缺乏某种资源而被迫等待，增加了端到端帧响应时延；若资源配置过剩，则会使资源空闲，从而降低了整体资源利用率. 为此，本文设计了任务需求感知算法，通过连续学习模型，依据历史视频任务的资源消耗特征，预测下一阶段视频任务的资源需求. 具体而言，每个连续学习神经网络层记录了一类任务的需求特征，通过比较新到任务的需求特征与已“学习”到的旧任务特征之间的表征距离，将任务需求进行分类. 此外，随着海量任务的持续涌现，连续学习模型所见到的任务需求类型将迅速增加，导致在学习新知识（即新任务的需求特征）时，易“遗忘”旧知识（即旧任务的需求特征），从而引发“灾难性遗忘”问题[12]. 为此，引入了在线强化学习的模型训练策略，通过结合云网资源使用量的长期变化趋势，对短期时间窗口内的任务需求进行微调，以实现鲁棒的需求决策引导.

2）云网资源统一封装与灵活调度难. 当多个用户将各类任务同时推送到云服务集群时，会存在多流竞争问题. 因此，保障各类视频任务的需求都能被云网调度器所满足并非易事. 为此，本文设计了差异化资源封装与调度算法，包括 2 个机制：① 云网资源封装与切片管理机制，用于评估 VIoT“端—边—云—网”资源的剩余量及可用状态，并依据各类资源的剩余状态，细粒度封装各物理资源块为虚拟资源组，以备后续调度；② 云网资源调度机制，依据任务需求内在的依赖关系将虚拟资源组分配给相应任务（组），以促进其快速处理. 此外，上述 2 个机制被嵌入到 VIoT 云网调度器的执行原语中，使之能对视频任务快速响应.

CLTUS 原型系统被部署于 VIoT 试验平台，并与 5G 网络和边缘计算协议（如 3GPP Release 15/16/17、ETSI MEC）兼容. 该平台的开发遵循网络控制面与用户面分离的原则，即将数据传输和控制信令交互隔离，以减少彼此干扰. 实验结果表明，相较于业界领先的 Tetris[8] 和 Elasecutor[9] 调度方法（分别被 Microsoft Azure 和华为云等主流云网平台所采用），CLTUS 将视频任务的平均处理效率提高了 127.73%，云网资源利用均衡率提高至 67.2%，并保障了多类 VIoT 应用的平均帧响应时延下降到 200 ms 以下.

本文的主要内容和创新贡献总结为 3 方面：

1）对 VIoT 应用在运营化云网平台的性能进行了定量评估，并定位影响任务处理效率的主要瓶颈；

2）提出了方法 CLTUS，包括任务需求感知算法和差异化资源封装与调度算法，实现了云网平台存储、传输、计算资源的统一封装，以及视频任务与上述资源的匹配，以满足各类视频任务的处理需求；

3）在协议兼容的 VIoT 试验平台上部署了 CLTUS 原型系统，并验证其有效性.

## 1 研究背景与问题

### 1.1 VIoT 应用场景

移动互联网、物联网、云计算、5G 等技术的普及，以及远程智能安防、车联网、超高清视频直播、云游戏等应用的兴起，促进了 VIoT 的发展. 如图 1 所示，海量视频网联设备（如智能手机、智能摄像头）持续拍摄视域内的画面，并通过蜂窝基站或 WiFi 接入点将视频上传到互联网的高性能云服务器中进行计算.

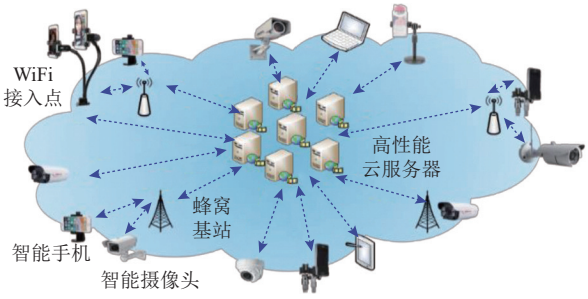



Fig. 1 VIoT application scenario

图 1 VIoT 应用场景

为了加速视频流的处理，业界提出边缘计算技术，使得位于远程云的 CPU、GPU、内存、外存、网络接入等计算、存储和网络通信资源“下沉”到网络边缘（如蜂窝基站背后），将部分用户的流量卸载到边缘服务器中进行处理，从而降低了视频帧的端到端响应时延. 此外，视频流的处理效率还取决于云网资源与任务需求的适配程度，若云网平台的资源调度器可以高效地将所需资源分配给视频任务，则可以提高任务处理效率，否则将影响任务的处理性能.

### 1.2 运营化 VIoT 存在的问题

近年来，VIoT 应用类型飞速增长，实时视频数据不断产生，增加了边缘/中心云的处理压力. 因此，以

运营化 5G 网络为承载主体的 VIoT 能否保障各类视频应用的性能成为业界普遍关心的问题. 为此, 本文在 VIoT 5G 运营环境中部署典型的视频应用(如面向车路协同的视频分析用例), 并实地测量其性能.

1)测量方法. 包括全链路测量[13](涵盖“端—边—云—网”等全链路要素的性能评估)、跨层测量(涉及应用层响应时延, 网络层数据包传输时延、吞吐量、丢包率, 链路层/物理层误码率、信噪比等指标), 以及主、被动测量(不仅能在应用运行时综合评估 VIoT 5G 运营环境的性能, 还能针对性开展部分网络要素的压力测试).

2)网络配置. VIoT 应用前端和后端程序分别部署于用户终端(如智能手机)和边缘/中心云服务器. 在实际运行过程中, 用户终端实时拍摄视频画面并将其通过 5G 基站接入互联网, 再上传到云服务器中进行后续处理. 其中, 5G 基站空口射频频段为 sub-6 GHz (中心频点为 3.5 GHz); 云服务器配备 NVIDIA Tesla V100 GPU(16 GB 显存).

3)测试用例. 采用业界通用的视频处理和分析应用, 包括用于目标检测的 YOLO[14]、用于实例分割的 Mask R-CNN[15] 和用于语义分割的 BiSeNet[16] 等. 上述用例对端到端帧响应时延较为敏感(通常限制于 200 ms 之内), 有利于暴露 VIoT 存在的性能问题.

通过大规模测量和结果分析, 得到如下发现:

1)时延敏感型 VIoT 应用性能较差. 首先统计端到端视频帧响应时延, 包括从视频帧被用户终端拍摄、通过网络上传到云、在云服务器中处理、视频推理结果反馈给用户终端的完整时延. 图 2 展示了某 900 个连续视频帧的端到端响应时延. 可以观察到, 平均帧响应时延为 316.56 ms, 且约有 56.89% 的视频帧响应时延在 200 ms 以上, 远不能达到时延敏感型 VIoT 应用的时延需求(即 200 ms 的容忍极限).

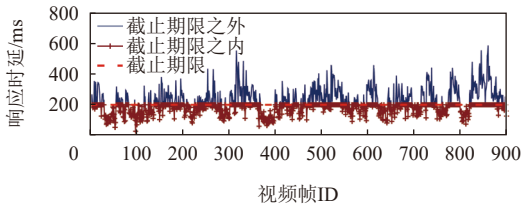


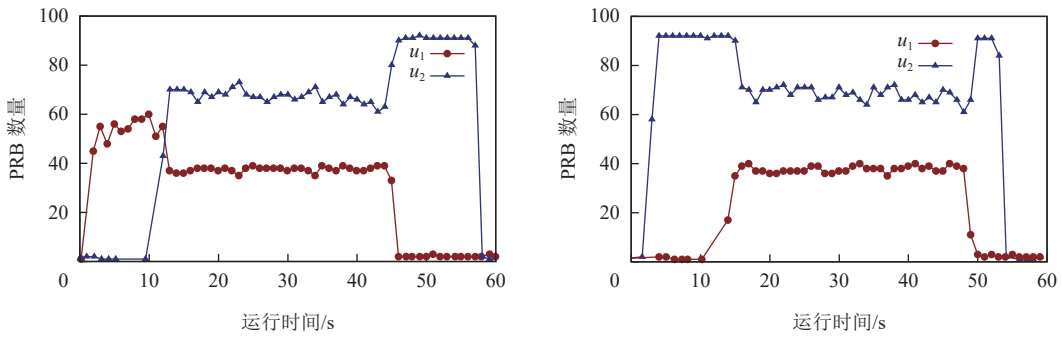

Fig. 2 End-to-end frame response latency of VIoT applications

图 2 VIoT 应用的端到端视频帧响应时延

此外, 在 2 个用户 $u_1$ 和 $u_2$ 分别发起性能需求不同的 VIoT 业务时(其中, $u_1$ 运行时延敏感型视频分析应用; $u_2$ 运行时延不敏感的视频点播应用, 即超高清视频下载并播放), 开展 2 组测试: 1) $u_1$ 先单独运行, 并于 10 s 后触发 $u_2$ 运行; 2) $u_1$ 和 $u_2$ 同时发起. 与此同时, 统计上述 2 个用户获得的物理资源块(physical resource block, PRB)数量, 并绘制在图 3 中. 从图 3(a)中可以观察到, 在第 1 组实验进行时, $u_1$ 在 10 s 前占据无线基站中主要的 PRB 数量. 然而, 当 $u_2$ 发起后, 大部分 PRB 立刻被 $u_2$ 占有. 更甚之, 在图 3(b)中, 当 $u_1$ 和 $u_2$ 中的应用同时发起时, $u_1$ 的资源抢占能力远不如 $u_2$: 在前 10 s 内, $u_1$ 近乎无法发起正常的数据传输; 在 10 s 后, $u_1$ 才逐渐争取到部分资源, 以进行数据传输.

综合上述分析, 目前运营化 VIoT 并不具备为时延敏感型 VIoT 应用提供可靠的资源保障能力, 并造成端到端视频帧响应时延过长. 通过进一步调研发现其原因在于, 运营化 VIoT 对多类视频任务的需求满足不均衡, 缺乏差异化的资源适配能力. 简言之, VIoT 云网平台无法准确感知应用数据流的性能需求, 并为其预留适配的云网资源. 一旦其他用户发起数据流(即使该数据流的性能需求非时延敏感), 则会抢占时延敏感型应用所占有的资源, 从而降低了视频任务的处理性能.

2)云网资源利用率较低. 选取时延敏感型视频




(a) $u_1$先运行, $u_2$后运行 (b) $u_1$和$u_2$同时运行

Fig. 3 Resource allocation towards different VIoT applications requirement

图 3 不同 VIoT 应用需求的资源分配

分析(video analytics, VA)、时延较敏感的视频直播(Web DASH)和视频点播(File)用例，并测试上述应用在并行执行时(即存在多用户竞争时)的存储、传输和计算资源的平均利用率. 该指标的统计方式为各资源利用率平均值的平均，反映了云网资源的整体使用状况. 此外，为了评估网络环境对云网资源利用率的影响，分别在静态场景(即手持用户终端站在基站信号覆盖范围之内测量链路性能)和在基站覆盖范围内以低速步行(约 4 km/h)开展性能测试. 结果如图 4 的累积分布函数(cumulative distribution function, CDF)所示.

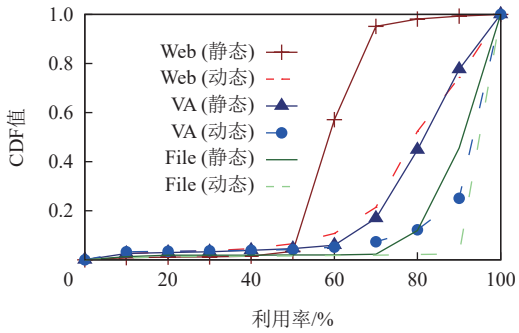



Fig. 4 Average resource utilization rate in multi-user competition

图 4 多用户竞争时的平均资源利用率

由图 4 可以观察到：1)各类视频任务运行时的存储、传输和计算资源平均利用率较低，约为 60%. 更有研究表明[9-10,17]，在更大规模的压力测量(万级别用户同时竞争云网资源)中，存储、传输和计算资源平均利用率将大幅度下降，甚至低于 15%. 通过细粒度分析各类资源在不同时刻的使用情况，发现上述现象的根本原因为，当多个用户之间存在资源竞争时，有限的资源长期被需求不紧急的任务占有，而需求紧急的任务无法得到资源而被迫等待. 这不仅影响了视频任务的处理效率，还造成各类资源的整体利用率下降.

2)相同数据业务在静态场景下的资源利用率较动态场景要高. 这取决于云网资源调度器对无线信道的自适应能力. 然而，目前运营化 VIoT 并没有将“端—边—云—网”的存储、传输、计算资源统一管理，即各种资源执行各自的调度规则. 例如，传输资源由网络运营商负责管理，存储、计算资源则被云服务提供商负责调度，由此造成了各类资源使用不均衡，易出现单台云服务器的计算资源使用过于频繁而其他资源使用空闲的情况(即“单点瓶颈”)，从而影响了整体任务的处理效率. 这也意味着，在云网资源调度中，不仅要提高单类资源的利用率，还要确保各种资源的联合调度，以促进整体性能的提升.

## 2 CLTUS 系统设计

### 2.1 系统概述

CLTUS 旨在降低 VIoT 应用的端到端响应时延并提高云网资源的均衡利用率. 接下来，详细阐述该系统的工作流程和关键模块设计.

如图 5 所示，CLTUS 部署于边缘云和中心云上. 在系统运行时，终端(智能手机、摄像机、车载摄像头等)持续拍摄视频画面并与无线接入网侧的基站建立数据链接使之上传到网络中. 然后，基站依据视频流的去向(每个数据包的包头中会有专门的字段记录 MAC/IP 地址)决定将其卸载到边缘云服务器或中心云服务器进行处理，并根据网络链路的拥塞情况进行转发. 在此过程中，CLTUS 实时感知不同用户发起的视频流(任务)的资源需求，并统计云网存储、传输和计算资源的剩余情况和可用状态，通过控制信令灵活调控各节点资源的调度，以满足各类视频任务的需求.

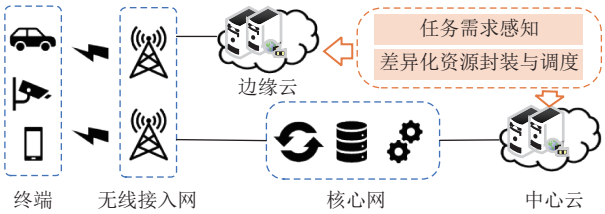



Fig. 5 Workflow of CLTUS

图 5 CLTUS 工作流程

在 CLTUS 运行过程中，共有 2 个关键模块维持系统的正常功能，如图 6 所示.

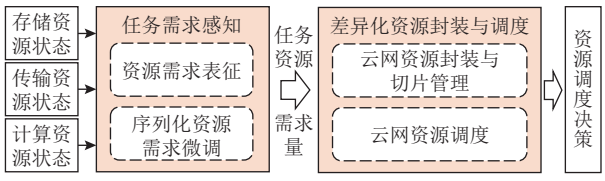



Fig. 6 Key modules of CLTUS

图 6 CLTUS 关键模块

1)任务需求感知模块. 该模块用于准确理解各类视频任务的资源需求，包括云网存储、传输和计算资源需求量、数据帧传输时延及紧迫程度等. 其算法难点在于各类视频任务的需求多样以及难以预测，且在系统实际运行时，短时间内即有大量任务涌现，需要 CLTUS 快速响应并识别出资源需求种类. 围绕

上述挑战,将连续学习运用到任务需求的理解中. CLTUS 将云服务器的历史存储、传输和计算资源状态输入到该模块中,并通过“资源需求表征”和“序列化资源需求微调”机制(详见 2.2 节)使 CLTUS 在各类任务到来时具象化表征任务的资源需求.

2)差异化资源封装与调度模块. 在获得任务资源需求量之后,CLTUS 利用软件定义网络(software-defined network, SDN)技术对各类云网资源进行封装和调度(详见 2.3 节). 首先,云网资源封装与切片管理机制依据云网存储、传输和计算资源的剩余量和可用状态,将各类资源划分成逻辑资源切片. 其次,云网资源调度机制参考 SDN 接口设计,在“端—边—云—网”各要素间通过轻量化交互协议将各自的剩余资源状态信息收集并暴露给 CLTUS 资源调度器. 该调度器依据感知到的任务需求,并基于 Blossom 打包算法[18]生成资源调度决策,使逻辑资源切片匹配任务需求,实现资源的按需配给.

### 2.2 任务需求感知模型

连续学习驱动的任务需求感知模型如图 7 所示,包括资源需求表征和序列化资源需求微调 2 个机制.

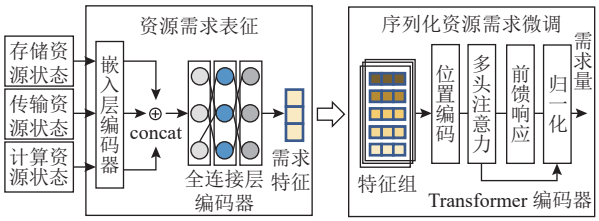



Fig. 7 Model structure of task demand perception

图 7 任务需求感知模型结构

1)资源需求表征. 该机制使用全连接网络[2]提取各视频任务对存储、传输和计算资源的需求. 首先,CLTUS 将云网平台的存储、传输和计算资源状态(即存储器大小、网络链路容量变化和 CPU 及 GPU 剩余量等)作为输入,经过嵌入层编码器(Embedding)将高维的原始数据映射到低维流形向量,再经 concat 拼接操作输出特征向量至全连接层编码器中,最后输出需求特征 $\boldsymbol{F}_i$. 上述操作由可扩展的全连接网络实现,表示为

$$(\boldsymbol{F}_i)^{\mathrm{T}} = serialize(E(\boldsymbol{S}_i \oplus \boldsymbol{T}_i \oplus \boldsymbol{C}_i)), \tag{1}$$

其中,$\boldsymbol{F}_i$表示某任务第$i$时刻存储、传输和计算资源的需求特征. $\boldsymbol{S}_i$、$\boldsymbol{T}_i$和$\boldsymbol{C}_i$为$m \times n_{\{\mathrm{s,t,c}\}}$维矩阵,分别表示第$i$时刻云网平台的存储、传输和计算资源状态. $m$为服务器个数,在第 3 节的设置中,$m = 15$; $n_{\mathrm{s}}$、$n_{\mathrm{t}}$和$n_{\mathrm{c}}$分别为云网存储、传输和计算资源的种类数,如存储资源包括内存和外存,则$n_{\mathrm{s}} = 2$. 同理,$n_{\mathrm{t}} = 1$表示端到端网络带宽; $n_{\mathrm{c}} = 2$表示 CPU 和 GPU 计算资源. $E(\cdot)$为嵌入层编码器,用于量化存储、传输和计算资源状态的潜在序列化特征. *serialize*(·)为全连接层编码器,用于对拼接得到的存储、传输和计算需求特征进一步量化.

为了提高模型的执行效率,CLTUS 为每个全连接层设置了“备忘录”,以记录当前所学任务需求特征,通过比较新到任务的需求特征与已学习到的旧任务需求特征之间的距离,将任务需求进行分类. 若特征距离较大,则新增加一个连接层,并为其指定一个需求匹配等级(即 SLA); 若特征需求距离在既定阈值之内,则参考既往全连接层对应的等级,为该任务的资源需求特征赋值. 上述“备忘录”全连接层在训练过程中维持一定的记忆能力,即通过自适应调整全连接网络结构以适配不同任务需求差异性的同时,还能保存“见过”的任务需求特征,以在后续相似任务出现时及时感知其资源需求.

2)序列化资源需求微调. 为了进一步提高任务资源需求理解的准确率,该机制根据历史任务流对存储、传输、计算资源利用率的变化,微调当前任务的资源需求量. 如图 7 右侧所示,该机制将相邻的任务需求特征序列化组合成特征组(即$\boldsymbol{F}_i^{(\mathrm{g})}$),并将其输入到 Transformer 编码器[19]. 该编码器包括位置编码(position embedding)、多头注意力(multi-head attention)响应机制、前馈(feed forward)响应和归一化(add & normalization)操作,以准确调整任务对存储、传输、计算资源的需求量.

在输入 Transformer 编码器前,需要对每个$\boldsymbol{F}_i^{(\mathrm{g})}$进行嵌入和位置编码. 此外,多头注意力块的每个头部$h = \{1,2,\cdots,H\}$需要转换为〈查询矩阵$\boldsymbol{Q}_h^{(\mathrm{g})} = (\boldsymbol{x}^{\mathrm{g}})^{\mathrm{T}}\boldsymbol{W}_h^{Q}$,关键矩阵$\boldsymbol{K}_h^{(\mathrm{g})} = (\boldsymbol{x}^{\mathrm{g}})^{\mathrm{T}}\boldsymbol{W}_h^{K}$,值矩阵$\boldsymbol{V}_h^{(\mathrm{g})} = (\boldsymbol{x}^{\mathrm{g}})^{\mathrm{T}}\boldsymbol{W}_h^{V}$〉. 其中,$\boldsymbol{W}_h^{*}$为权重矩阵. 上述转换完成后,则将矩阵送入 *softmax* 函数进行缩放,得到输出$\boldsymbol{O}_h^{(\mathrm{g})}$:

$$\left(\boldsymbol{O}_h^{(\mathrm{g})}\right)^{\mathrm{T}} = softmax\left(\frac{\boldsymbol{O}_h^{(\mathrm{g})} \cdot \boldsymbol{K}_h^{(\mathrm{g})\mathrm{T}}}{\sqrt{d_k}}\right)\boldsymbol{V}_h^{(\mathrm{g})}. \tag{2}$$

多头注意力块还包括 BatchNorm 层和一个带有残差连接的前馈网络.

需要说明的是,该 Transformer 编码器的输出值为五维向量,即〈内存、外存、带宽、CPU、GPU〉的需求量. 该向量各项数值经归一化操作,表示占有相应资源最大值的比例. 若考虑其他资源,则增加该向量的维度即可.

为了使 CLTUS 及时适应云网资源的变化和海量任务的突发性,将上述任务需求感知模型(智能体)

置于在线强化学习架构中进行训练. 如图 8 所示，该模型以步数为单位进行迭代，每步需将“端—边—云—网”的存储、传输与计算资源状态作为输入，由任务需求感知智能体做出“动作决策”，而后由差异化资源封装与调度模块(详见 2.3 节)依据任务需求和云网平台所剩余资源状态执行调度动作，并返回新的状态观测值作为下一步迭代的输入. 其中，动作(action)决策空间为任务需求感知模块输出变量的取值范围，即任务资源需求量〈内存、外存、带宽、CPU、GPU〉的类别；状态(state)空间为任务需求与服务器资源的匹配情况(详见 2.3 节).

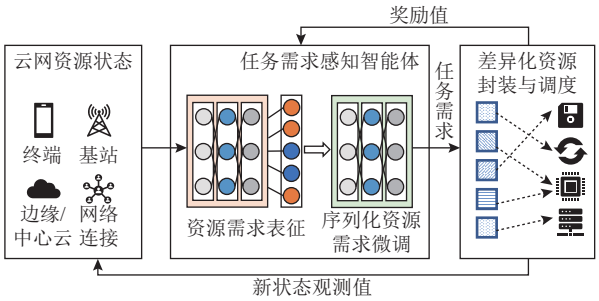



Fig. 8 Training process of CLTUS

图 8 CLTUS 训练过程

简言之，为了引导该任务需求感知智能体做出正确的决策，CLTUS 在每个动作执行后，根据决策动作和优化目标给智能体一个奖励值，以在线调整资源分配决策动作，使之在动作空间中找到最佳匹配决策. 例如，若所分配的云网资源与任务需求相匹配(任务执行效率高，不存在资源等待情况)，则差异化资源封装与调度模块会给予任务需求感知智能体相应的奖励值，以促进其模型训练.

此外，CLTUS 使用策略梯度算法对连续学习模型参数执行梯度下降，促进其快速收敛到稳定状态. 其中，$\theta$表示连续学习模型参数；$\pi_\theta(s_t,a_t)$表示在状态$s_t$下执行调度策略$a_t$的概率. 在第$k$个动作执行后，CLTUS 对智能体使用$r_k=-(t_k-t_{k-1})N_k$作为惩罚，即任务执行的总时延越长，奖励值越小，$N_k$为相邻时间间隔内的任务数. 假设一步训练中包含$T$次决策动作，则智能体收集每次的状态、动作以及奖励值$(s_k,a_k,r_k)$，以对策略$\pi_\theta(s_t,a_t)$的参数$\theta$进行更新，即

$$\theta\leftarrow\theta+\alpha\sum_{k=1}^{T}\nabla_\theta\log\pi_\theta(s_t,a_t)\left(\sum_{k'=1}^{T}r_{k'}-b_k\right),\quad(3)$$

其中，$\alpha$表示学习率；$b_k$表示减少策略梯度方差的基准. 在训练过程中，初始学习率设定为 1E−7，批量大小(batch size)设置为 32，特征组长度设置为 16，步长设置为 8.

### 2.3 差异化资源封装与调度模块

该模块旨在对“端—边—云—网”的存储、传输和计算资源进行虚拟化封装，并根据任务需求感知模块(2.2 节)学习到的视频任务需求细粒度调度上述资源，实现所分配资源与任务需求间的最佳匹配. 上述功能依赖于云网资源封装与切片管理机制和云网资源调度机制. 此外，CLTUS 的设计中还包括轻量级信令交互协议，用于支持上述模块运行时“端—边—云—网”之间的信令交互. 接下来，简述上述机制的实现.

1)云网资源封装与切片管理. 该机制用于实时监测云网平台的存储、传输和计算资源的剩余量及可用状态，并在逻辑上细粒度划分各类物理资源为多个虚拟化资源切片，以备后续调度所用.

①存储、传输和计算资源监测. 所需检测的资源包括用户终端与无线接入网(radio access network, RAN)之间的射频资源，RAN 与边缘/中心云之间的有线网络资源(如网络带宽)，以及边缘/中心云的存储、传输和计算资源. 对于射频资源，主要通过捕获基站侧 RAN 信令来获取无线信道质量，并查表计算出所需的无线资源[13]. 例如，在 RAN 中，无线信道质量(如 RSRP, SINR)可通过解析链路层 MAC CE 信令来获取. 对于有线网络带宽，可通过探测链路实时吞吐量来获得(探测方法参见文献 [13]). 对于云服务器资源，可通过拦截服务器运行时的控制信令(如 psutil)来获取每个调度单位内可用的资源数量.

②逻辑资源封装. 通过资源切片技术[20]实现. 简言之，CLTUS 根据接收到的剩余资源信息和可用状态来划分底层云网资源，并调控资源的预留以生成逻辑资源切片. 例如，若监听到某任务对某类资源需求紧急，则立即发送信令提前预留该资源；反之，则在下一轮任务发起时，再完成逻辑资源的划分. 特别地，该方式不涉及底层资源的改动，从而在一定程度上减小了算法的实现难度，保障了物理资源间的隔离.

③切片管理. 在云网资源封装、调度和执行过程中，CLTUS 遵循图 9 所示的状态机来管理资源. 当用户发起云网资源请求后，CLTUS 会评估云网所剩余资源是否支持该请求(依据 2.2 节任务需求感知模块输出的资源需求量). 若能够满足，则新建一张资源切片，并使之进入就绪态；若所剩资源不足以满足用户请求(包括能满足存储和传输资源需求，但不满足计算资源)，则也新建一张切片，但被设置为等待态，以等待应用重新发起请求. 此时，若得到调度器的派

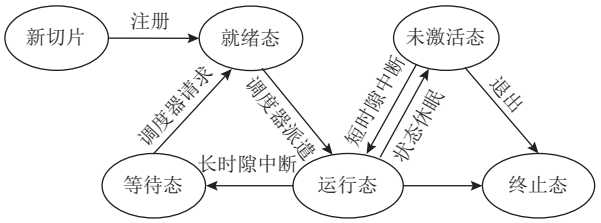



Fig. 9 Control and management of slice state

图 9 切片状态控制与管理

遣指令, 处于就绪态的切片会进入运行态, 并向用户提供服务. 当切片所占资源(具体数量由后文云网资源调度机制所得)在较长时间内无法满足所有任务的需求时, 状态机会淘汰部分切片, 并使之进入终止态. 当用户得知其服务请求被驳回或切片被淘汰后, 会重新向状态机发起请求以重新流转回就绪态. 此外, 若处于未激活态的切片在一段时间内没有被激活(获得短时隙中断指令), 则会自动流转到终止态以释放所占系统资源.

2)云网资源调度. 该机制用于动态调度各类资源, 以满足视频任务的需求. 其核心为资源切片调度器, 用于将逻辑资源切片与视频任务(组)相匹配, 共包括任务需求依赖打包算法和资源切片调度算法. 接下来, 详细介绍这 2 种算法的实现细节.

①任务需求依赖打包算法. 首先解释任务需求依赖的概念. 假设某任务对 CPU 资源需求量大且迫切, 对内存资源需求量小且不紧急, 而另一任务对上述 2 种资源的需求情况恰好相反, 则这 2 个任务存在较强的依赖关系, 可将两者打包并送入同一台服务器中并行处理, 这样既可以避免两任务在顺序执行时, 计算资源和存储资源使用不均衡而造成的整体资源利用率低, 又可以提升多任务的整体处理效率.

为了实现上述目标, 首先遍历各任务, 以确定各任务间的依赖关系:

$$\gamma(t_1,\cdots,t_m)=\sum_{i=1}^{k}\left[R^{\max}-\sum_{j=1}^{m}R_j^i\right], \tag{4}$$

其中, $k$表示某$\{t_1,\cdots,t_m\}$视频任务组(共$m$个任务)所有任务所需的资源种类数; $R^{\max}$表示该任务组中, 所有任务对各类资源需求量的最大值(例如, 若所有任务对 CPU、GPU、内存资源总需求量分别为 10%、20%、30%, 则$R^{\max}=30\%$); $R_j^i$表示第 $j$个任务对第$i$种资源的需求量, 则$\sum_{j=1}^{m}R_j^i$表示所有任务对第$i$种资源的总需求量. $\sum_{i=1}^{k}\left[R^{\max}-\sum_{j=1}^{m}R_j^i\right]$表示该任务组中, $R^{\max}$与各种资源总需求量的差值之和, 即$\gamma(t_1,\cdots,t_m)$值越大, 意味着该任务组中任务与任务之间的匹配度越小.

②资源切片调度算法. 云网资源的剩余量和可用状态实时变化, 即在不同时刻各服务器资源的剩余量均不同. 如何快速搜索任务(组)需求与云网资源的最佳匹配是该调度问题的关键. 需要强调的是, 两者随机变化, 使得任务与资源间的匹配成为 NP-hard 问题, 导致最佳匹配对的搜索空间随任务量的增加呈指数级上升.

为了高效搜索任务组与云网资源之间的匹配, CLTUS 改进 Blossom 算法[21], 选取各任务组中各类资源需求量与各服务器中资源剩余量之间标准差最小的匹配对, 作为调度决策的依据. 其对应公式为

$$\beta=\arg\min D_{(R_{\{t_1,\cdots,t_m\}},R_s)}\left(\sum_{i=1}^{k}|R_1^i-R_s^i|,\cdots,\sum_{i=1}^{k}|R_j^i-R_s^i|,\cdots,\sum_{i=1}^{k}|R_m^i-R_s^i|\right), \tag{5}$$

其中, $|R_j^i-R_s^i|$表示第 $j$个任务中第$i$种资源的需求量与第$s$台服务器中第$i$种资源剩余量差值的绝对值, 则$\sum_{i=1}^{k}|R_j^i-R_s^i|$表示第 $j$个任务中各种资源需求量与第 $s$台服务器中各种资源剩余量差值的绝对值之和. $D_{(R_{\{t_1,\cdots,t_m\}},R_s)}$表示某$\{t_1,\cdots,t_m\}$视频任务组与第$s$台服务器可用资源的标准差. CLTUS 选取标准差最小的任务组与服务器之间进行匹配.

最后, 总结上述任务需求依赖打包和资源切片调度机制的实现过程, 如算法 1 所示.

**算法 1.** 任务需求依赖打包和资源切片调度.

输入: 视频任务集$T=\{t_1,\cdots,t_m\}$、视频任务需求集$T_{\mathrm{r}}=\{t_{\mathrm{r}}^1,\cdots,t_{\mathrm{r}}^m\}$、中心云服务器集$C=\{C_1,\cdots,C_n\}$、各中心/边缘云服务器当前可用资源集$C_{\mathrm{a}}$;

输出: 中心/边缘云服务器任务分配集 $X$.

① while 任务$t\in T$且$T\neq\varnothing$ do
② while $\forall\{t_1,\cdots,t_k\}\subset T$, $k\leqslant m$ do
③ $\gamma(t_1,\cdots,t_k)$; /*计算任务依赖关系*/
④ if $\gamma(t_1,\cdots,t_k)<\gamma_0$
⑤ $\gamma_0=\gamma(t_1,\cdots,t_k)$;
⑥ end if
⑦ end while
⑧ while $\{C_{\mathrm{a}}^1,\cdots,C_{\mathrm{a}}^m\}$
⑨ $X\leftarrow\arg\min D_{\langle C_{\mathrm{a}},T_{\mathrm{r}}\rangle}$;
/*服务器剩余资源与任务需求的匹配程度*/
⑪ end while
⑫ end while
⑬ return $X$.

3)轻量级信令交互协议. 本文还设计了兼容 5G Rel-16 标准的资源切片控制信令交互协议，用于在 VIoT“端—边—云—网”间传递信令以实现资源切片的封装与调度. 如图 10 所示，该协议使用多线程多路 I/O 复用的方式定义接口信令集. 其中，主线程通过监听切片状态机(如图 9 所示)的状态切换消息，并解析消息中的关键字段以将对应的视频任务和逻辑资源切片分配给线程池中的线程. 其通信实体包括：基于 SDN 的底层基础设施抽象(其中，RAN 部分由 FlexRAN[2] 实现，核心网部分由 OpenFlow 实现)，以提供底层网络支持；移动边缘网中的移动边缘计算(mobile edge computing, MEC)服务，主要由无线网络信息服务功能(对接 radio API，用于接收信道监听信息)和中间用户面功能(intermediate user plane function, I-UPF)(负责引导数据流到边缘服务器中进行计算)组成.

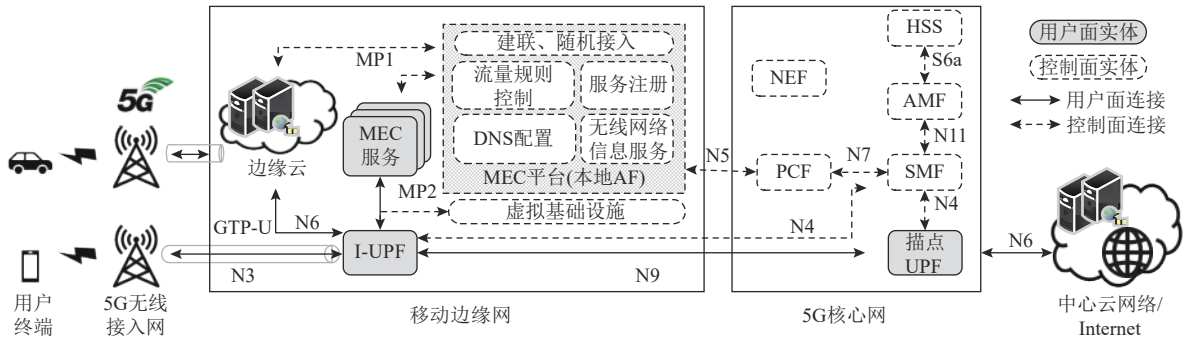



Fig. 10 Lightweight interaction protocol and interface design

图 10 轻量化交互协议和接口设计

此外，为了便于不同通信实体之间的跨层通信，CLTUS 分别在 VIoT 底层云网基础设施和移动边缘网平台设置了南向接口，以传递云网剩余资源状态信息和资源切片控制信令. 同时，还在边缘云应用(包括 MEC 服务)和 I-UPF 之间设计了北向接口，用于数据流转发和交付. 特别地，北向接口的消息报文格式符合 HTTP/1.0 协议的要求，且报头格式统一，以方便消息报文的封装和解封.

## 3 CLTUS 原型系统部署

在开源的软件定义 VIoT 5G 系统[2,6,7] 上设计并开发了 CLTUS 原型系统. 如图 11 所示，该试验床包括支持 5G 网络制式接入的智能手机、无线基站、RAN(包括 MEC 管理程序和资源切片控制程序)、核心网和边缘/中心服务器(NVIDIA Tesla V100, Linux Ubuntu 20.04.6 LTS). 其中，在射频前端，使用 Ettus USRP B210 作为信号处理单元. 该原型系统对应的物理设备信息如表 1 所示.




Fig. 11 CLTUS prototype testbed

图 11 CLTUS 原型试验床

**Table 1 Physical Device List for CLTUS Prototype System**

**表 1 CLTUS 原型系统物理设备清单**

| 系统组成 | 设备型号 | 实现功能 |
|---|---|---|
| 智能手机 | Redmi K30i,<br>ZTE Axon 10 Pro | 搭载 VIoT 应用前端 |
| RAN | HP ZHAN | 维护 RAN 数/控制面功能 |
| 核心网 | DELL G3 | 维护核心网功能 |
| 边缘/中心服务器 | NVIDIA Tesla V100 | 视频任务处理 |

在该系统中，智能手机能够收发 Sub-6 GHz 5G 信号，并将用户发起的 VIoT 数据流上传到边缘/中心服务器中进行处理. 任务需求感知模块(2.2 节)和差异化资源封装与调度模块(2.3 节)同样运行于边缘/中心服务器中，以实现高精度视频任务需求理解和灵活的云网资源调度.

此外，CLTUS 还在商用云平台上部署. 简言之，视频流仍通过图 11 所示的智能手机上传到无线基站，但在 RAN 侧，则直接通过网线连接到互联网，而后将视频任务置于商用边缘/中心服务器中进行处理. 其中，硬件环境为中国电信天翼视联网平台，包括

15 台物理机器(4 核 CPU, 16 GB 内存, 80 GB 磁盘容量, 100 Mbps 出口带宽); 软件环境为 Linux Ubuntu 20.04.6 LTS.

## 4 实验评估

### 4.1 实验设置

1)对比方法. 5G MEC(基准方法)为运营商目前所采用的云网资源调度方法; Tetris[8] 和 Elasecutor[9] 分别为应用于 Microsoft Azure 和华为云平台的资源调度方法; 基于深度学习的资源调度方法 Para[10]. 特别地, Para 方法主要针对 GPU 资源进行调度. 本文在此基础上将 Para 方法扩展到了对存储、传输和计算资源的联合调度.

2)测试数据. 选取开源的流式/批处理数据任务集 Alibaba Cluster Trace Program[22], 该数据集不仅包括人脸检测(MTCNN)、目标检测(YOLOv5)、目标追踪(SiamFC)、实例分割(BiSeNetv2)和语义分割(Condlnst)等 VIoT 任务, 还包括操作系统事件、深度学习训练等通用任务. 最终, 选取 50 万条 VIoT 视频任务集测试 CLTUS 和相关对比方法的性能; 为了扩大数据规模, 还将任务集类型扩展到系统处理、深度学习训练等通用任务中, 共计处理超过 230 万条任务.

3)测试指标. 视频帧响应时延, 表示数据帧从终端上传、到被服务器处理, 以及处理结果反馈到终端的端到端时延; 平均任务吞吐量(单位: 任务数/秒), 表示云网平台每秒所能处理的任务数量; 平均资源利用均衡率, 表示云网存储、传输和计算资源平均利用率的平均值(需要说明的是, 该指标数值越大, 意味着云网平台各种资源均衡利用的程度越高, 即避免了某种资源被过度使用而其他资源空闲的情况); 任务需求理解准确率, 表示连续学习驱动的任务感知模型对不同任务资源需求的识别准确率.

### 4.2 系统性能测试

1)VIoT 应用性能. 选取了典型的时延敏感型视频分析用例(如目标检测和实例分割), 并部署于 CLTUS 原型系统中, 如图 12 所示. 在测试过程中, 智能手机将所拍摄的视频帧实时推送到服务器中进行推理, 服务器再将推理结果(包含目标检测框、目标相对位置等)返回给终端. 同时, 记录视频响应时延和推理准确率(服务器反馈的推理结果与当前视频帧推理目标基准位置的偏差).

如图 12 所示, 5G MEC, Tetris, Elasecutor 和 Para 的平均视频帧响应时延均在 200 ms 之上, 甚至高达

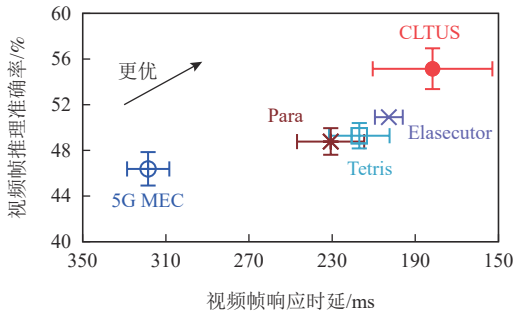



Fig. 12 Performance improvements of VIoT applications in different algorithms

图 12 在不同算法下的 VIoT 应用性能提升

316.3 ms, 而 CLTUS 的平均视频帧响应时延为 185.7 ms, 能够满足时延敏感型视频分析应用的容忍时延要求(即 200 ms). 此外, CLTUS 的视频帧推理准确率也提高至 54.23%, 相比于其他算法分别增加了 5.87 个百分点、7.12 个百分点、7.24 个百分点和 11.32 个百分点. 该推理准确率提升的原因为视频帧响应时延越短, 则意味着服务器推理画面与用户终端当前所视画面越相似, 因此推理结果越匹配. 由于 CLTUS 的视频帧响应时延最短, 其推理准确率也相应较高.

2)系统并发处理能力. 分别将测试任务集输入到商用云平台(包括 15 台物理机器)的 CLTUS 和相关对比方法, 其他实验条件保持一致. 测试结果如图 13 所示, 在几十万级别的任务规模下, 4 种方法的平均任务吞吐量为 81.19 任务数/秒、80.75 任务数/秒、73.23 任务数/秒和 144.43 任务数/秒, 即 CLTUS 的平均作业吞吐量相较于 Tetris, Elasecutor 和 Para 分别提升了 77.89%、78.86% 和 97.23%(平均提升了 84.66%); 在大规模(百万级)任务集下, 4 种方法的平均任务吞吐量分别为 87.42 任务数/秒、87.36 任务数/秒、79.85 任务数/秒和 192.94 任务数/秒, 即 CLTUS 的平均任务吞吐量相比于 Tetris, Elasecutor 和 Para 提升了 120.70%、

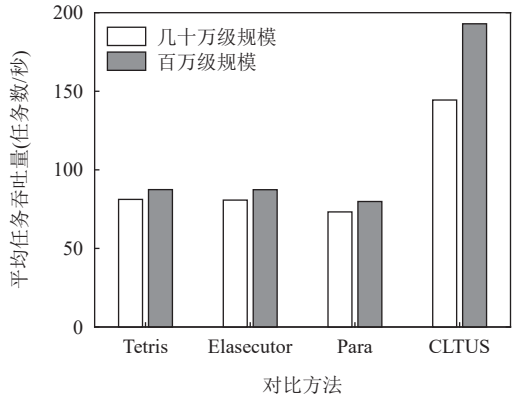



Fig. 13 Average task throughput of different methods

图 13 不同方法的平均任务吞吐量

120.86% 和 141.63%(平均提升了 127.73%).

上述任务处理效率的提升归因于 CLTUS 能够准确理解 VIoT 应用的任务需求(包括需求量和紧迫程度),并灵活调度存储、传输和计算资源满足上述需求,从而显著提升了任务的处理效率.此外,相较于几十万级规模,任务的处理效率在百万级规模下提升得更加明显.这得益于随着任务规模扩大,任务需求种类将更加丰富,扩增了任务需求匹配的空间(即 CLTUS 更能利用任务需求间的依赖关系找到最佳的任务打包组合),从而提升了任务在服务器中并发处理的能力.

3)云网资源利用效率.下面测试这 15 台云服务器在运行时的平均资源利用均衡率.如表 2 所示,相比于 Tetris, Elasecutor 和 Para, CLTUS 在几十万级任务规模下,平均资源利用均衡率分别提高了 41.81%, 41.38% 和 54.48%.此外,在百万级任务规模下,相比于 Tetris, Elasecutor 和 Para, CLTUS 的平均资源利用均衡率分别提高了 35.10%, 34.98% 和 49.31%.

**Table 2 Average Resource Utilization Stability under Different Task Scales**

**表 2 不同任务规模下的平均资源利用均衡率**

| 调度方法 | 几十万级规模(50 万)/% | 百万级规模(230 万)/% |
|---|---|---|
| Tetris | 62.91 | 68.43 |
| Elasecutor | 63.10 | 68.49 |
| Para | 57.75 | 61.92 |
| CLTUS | 89.21 | 92.45 |

表 2 表明,CLTUS 具有较好的云网资源均衡利用的能力.该能力源自 CLTUS 对任务资源需求的准确理解,由此根据各类任务的需求差异,灵活调整各种云网资源与之匹配.例如,若监听到对存储资源需求迫切的任务大量出现,而对计算等资源需求相关的任务数量较少时,CLTUS 可以调控切片状态机中的参数配置,使存储资源相关的逻辑切片生命周期变短,从而增长计算资源相关切片的生命周期.

### 4.3 微基准测试

1)任务需求理解准确率.2.2 节连续学习驱动的任务需求感知模型可以在不同资源需求的视频任务持续到来时,适应任务需求分布的变化来保障识别精度.为此,本文选取 10 万条对内存、外存、带宽、CPU、GPU 五种资源需求有明显差异的视频任务,并将其随机打乱以测试在资源类别数分别为 2、3、4 和 5 下的需求理解准确率.例如,若当前的资源种类为 CPU 和 GPU(即资源类别数为 2),则统计该模型所能理解出的相应任务数占任务总数的比值.需要说明的是,表 3 中的资源类别数为 2,意味着上述 5 种资源两两组合,共计 10 种情况.可以观察到,当资源类别数为 2 时,准确率最高,为 99.4%;当资源类别数为 5 时,准确率下降至 94.3%.随着资源类别数的增加,任务需求理解准确率会有所下降,但均维持在 94.3% 以上,显示出该连续学习驱动的任务需求感知模型具有较好的理解准确率.

**Table 3 Accuracy of Task Demand Understanding**

**表 3 任务需求理解准确率**

| 资源类别数 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|
| 理解准确率/% | 99.4 | 97.7 | 95.2 | 94.3 |

2)任务需求依赖打包和资源切片调度效率.2.3 节云网资源调度机制的核心思想为将任务需求依赖程度高(资源需求“互补”)的任务打包在一起,然后为该任务组分配适配的资源切片.这样既可以满足任务组的资源需求且可以使多个任务并行执行,还可以保障服务器各种资源的利用相对均衡.为了验证该机制的效率,图 14 描绘了 15 台云服务器运行时各类资源利用率(已做归一化处理,即各类资源的使用量占该类资源总量的比值)的方差.可以观察到,相比于 Para 方法(其他对比方法的结果类似),CLTUS 的资源利用率方差相对“平稳”,即根据需求依赖打包的多个任务被均衡地分配到了各台服务器.然而,Para 中各台服务器资源的使用并不均衡.可以明显看出,其资源调度器将视频任务过多地分配给了某几台服务器,而其余服务器相对空闲,造成了“单点瓶颈”.

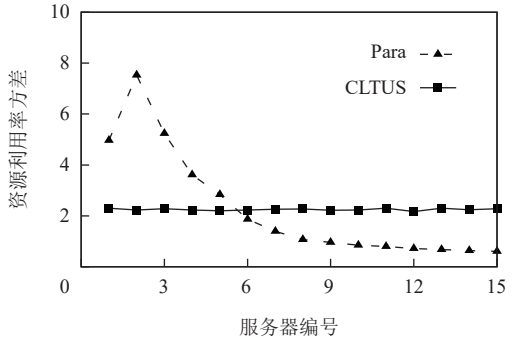



Fig. 14 Variance of server resource utilization

图 14 服务器资源利用率方差

## 5 相关工作

1)VIoT 应用性能优化.近年来,促进 VIoT 应用

性能的提升引起了研究人员的广泛关注. 例如, MemNAS[17]通过将深度学习推理模型量化、压缩和修剪, 以加速推理过程. 然而, 上述操作会导致推理精度降低, 从而限制了其在现实场景中的应用. 其次, 一些研究工作通过并行化视频处理流水线来减少视频帧的端到端响应时延. 例如, Sprocket[23]使开发人员能够对视频内容进行编程, 同时使视频块内并行来降低推理延迟. GraphLab[24]提出了一个图并行计算框架, 以实现内存的高效共享. 其他工作[5,25]则遵循负载均衡原则, 将任务分配到分布式计算实体(例如终端设备、边缘主机和远程服务器)间进行处理. 然而, 这会导致多个分布式实体之间的网络传输延迟增加.

2)5G 边缘云网资源调度. 5G MEC 资源管理工作包括 Tutti[2], OnSlicing[20]等. Tutti 提出了一个轻量级 5G RAN 和 MEC 耦合框架, 通过跨应用层和物理层信息融合, 在用户空间调控时延敏感流的资源分配. OnSlicing 使用在线深度强化学习方法与系统结构信息引导的资源虚拟化动作和多臂老虎机调度器进行自适应资源切片管理. 然而, Tutti 的内容预测方法依赖于所收集的历史视频信息, 预测精度在不同场景下具有一定局限性. OnSlicing 基于在线强化学习的网络切片方法, 在面对高动态的网络环境时, 需要频繁读取系统结构信息, 增加了计算资源的消耗. CLTUS 通过感知 VIoT 任务需求, 细粒度调度云网存储、传输和计算资源, 其在复杂环境中的自适应能力较强. 此外, 本文测试结果也验证了该方法对任务处理效率的提升, 以及对资源利用均衡率的改善.

## 6 总结与展望

本文提出了一种基于连续学习的视频物联网任务需求理解与调度方法(CLTUS), 通过连续学习准确感知各类视频任务的需求, 并设计差异化资源封装与调度机制实现灵活的存储、传输和计算资源供给. CLTUS 部署于符合 5G 和云计算协议标准的可编程实验平台上, 提升了 VIoT 应用的性能.

未来工作包括 2 方面: 1)为了提高云网资源的使用效率, 差异化资源封装与调度机制通过查找最佳任务需求组合并基于 Blossom 算法将其指派到资源充足的服务器中进行处理. 然而, Blossom 算法的时间复杂度相对较高, 甚至能够达到 $O(n^3)$级别. 在未来工作中, 拟优化该调度算法的处理性能, 提高任务与服务器间的匹配效率. 2)在系统性能评估时, 选取了目标检测等视频分析任务, 以及系统处理、深度学习训练等任务. 为了扩大系统的通用性, 未来工作将进一步增加 VIoT 任务种类和测试规模, 使 CLTUS 的实验结论更具通用性.

**作者贡献声明:** 徐冬竹负责算法设计、系统实现、数据分析以及论文撰写; 周安福和马华东负责整篇论文的研究思路, 给予建设性指导意见并修改论文; 张园提出了在实际商用系统中的推广思路及部署方案, 并提供实验平台以支持应用验证.

## 参 考 文 献

[1] Zhang Huanhuan, Zhou Anfu, Hu Yuhan, et al. Loki: Improving long tail performance of learning-based real-time video adaptation by fusing rule-based models[C]//Proc of the 27th Annual Int Conf on Mobile Computing and Networking (MobiCom'21). New York: ACM, 2021: 775–788

[2] Xu Dongzhu, Zhou Anfu, Wang Guixian, et al. Tutti: Coupling 5G RAN and mobile edge computing for latency-critical video analytics[C]//Proc of the 28th Annual Int Conf on Mobile Computing and Networking (MobiCom'22). New York: ACM, 2022: 729–742

[3] Liu Luyang, Li Hongyu, Gruteser M. Edge assisted real-time object detection for mobile augmented reality[C]//Proc of the 25th Annual Int Conf on Mobile Computing and Networking (MobiCom'19). New York: ACM, 2019: 1–16

[4] Machina Research. The number of IoT connections worldwide is expected to reach 27 billion by 2025[EB/OL]. [2016-08-09]. https://www.199it.com/archives/505556.html

[5] Tang Xiongyan, Cao Chang, Wang Youxiang, et al. Computing power network: The architecture of convergence of computing and networking towards 6G requirement[J]. China Communications, 2021, 18(2): 175−185

[6] Coronado E, Khan S N, Riggio R. 5G-EmPOWER: A software-defined networking platform for 5G radio access networks[J]. IEEE Transactions on Network and Service Management, 2019, 16(2): 715−728

[7] Coronado E, Yousaf Z, Riggio R. LightEdge: Mapping the evolution of multi-access edge computing in cellular networks[J]. IEEE Communications Magazine, 2020, 58(4): 24−30

[8] Grandl R, Ananthanarayanan G, Kandula S, et al. Multi-resource packing for cluster schedulers[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(4): 455−466

[9] Liu Libin, Xu Hong. Elasecutor: Elastic executor scheduling in data analytics systems[C]//Proc of the ACM Symp on Cloud Computing (SoCC'18). New York: ACM, 2018: 107–120

[10] Zhang Haitao, Geng Xin, Ma Huadong. Learning-driven interference-aware workload parallelization for streaming applications in heterogeneous cluster[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(1): 1−15

[11] Jiang Yulong, Dong Fang, Guo Xiaolin, et al. Potential game based workflow task offloading optimization mechanism in computing power network[J]. Journal of Computer Research and Development, 2023, 60(4): 797−809 (in Chinese)
(姜玉龙, 东方, 郭晓琳, 等. 算力网络环境下基于势博弈的工作流任务卸载优化机制[J]. 计算机研究与发展, 2023, 60(4): 797−809)

[12] Rebuffi S, Kolesnikov A, Sperl G, et al. iCaRL: Incremental classifier and representation learning[C]//Proc of 2017 IEEE Conf on Computer Vision and Pattern Recognition (CVPR). Piscataway, NJ: IEEE, 2017: 5533−5542

[13] Xu Dongzhu, Zhou Anfu, Zhang Xinyu, et al. Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption[C]//Proc of the Annual Conf of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'20). New York: ACM, 2020: 479–494

[14] Jocher G, Chaurasia A, Jing Qiu. Ultralytics YOLO [EB/OL]. [2023-07-02]. https://github.com/ultralytics/ultralytics

[15] He Kaiming, Gkioxari G, Dollár P, et al. Mask R-CNN[C]//Proc of the IEEE Int Conf on Computer Vision. Los Alamitos, CA: IEEE Computer Society, 2017: 2961–2969

[16] Yu Changqian, Gao Changxin, Wang Jingbo, et al. 2021. BiSeNet V2: Bilateral network with guided aggregation for real-time semantic segmentation[J]. Computer Vision and Pattern Recognition, 2021, 129(11): 3051−3068

[17] Liu Peiye, Wu Bo, Ma Huadong, et al. MemNAS: Memory-efficient neural architecture search with grow-trim learning[C]//Proc of 2020 IEEE/CVF Conf on Computer Vision and Pattern Recognition (CVPR). Los Alamitos, CA: IEEE Computer Society, 2020: 2105−2113

[18] Mucha M, Sankowski P. Maximum matchings via Gaussian elimination[C]//Proc of 45th Annual IEEE Symp on Foundations of Computer Science. Los Alamitos, CA: IEEE Computer Society, 2004: 248−255

[19] Nie Yuqi, Nguyen N H, Sinthong P, et al. A time series is worth 64 words: Long-term forecasting with transformers[C]//Proc of the 11th Int Conf on Learning Representations. Los Alamitos, CA: IEEE Computer Society, 2022: 1−24

[20] Liu Qiang, Choi N, Han Tao. OnSlicing: Online end-to-end network slicing with reinforcement learning[C]//Proc of the 17th Int Conf on Emerging Networking Experiments and Technologies (CoNEXT'21). New York: ACM, 2021: 141–153

[21] Villani C. Optimal Transport: Old and New[M]. Berlin: Springer, 2009

[22] Alibaba Group. Alibaba Cluster Trace Program cluster-trace-v2017[EB/OL]. [2024-07-31]. https://github.com/alibaba/clusterdata/blob/7358bbaf40778d4bd0464a64a430812088b7b74e/cluster-trace-v2017/trace_201708.md

[23] Ao Lixiang, Izhikevich L, Voelker G M, et al. Sprocket: A serverless video processing framework[C]//Proc of the ACM Symp on Cloud Computing (SoCC'18). New York: ACM, 2018: 263–274

[24] Low Y, Bickson D, Gonzalez J, et al. Distributed GraphLab: A framework for machine learning and data mining in the cloud[C]//Proc of the VLDB Endowment. New York: ACM, 2012: 716–727

[25] Hung C C, Ananthanarayanan G, Bodik P, et al. VideoEdge: Processing camera streams using hierarchical clusters[C]//Proc of 2018 IEEE/ACM Symp on Edge Computing. Los Alamitos, CA: IEEE Computer Society, 2018: 115−131



**Xu Dongzhu**, born in 1996. Postdoc. His main research interests include VIoT and time-sensitive networking.
**徐冬竹**, 1996 年生. 博士后. 主要研究方向为视频物联网、时间敏感网络.



**Zhou Anfu**, born in 1981. PhD, professor. His main research interests include IoT and mobile computing.
**周安福**, 1981 年生. 博士, 教授. 主要研究方向为物联网、移动计算.



**Ma Huadong**, born in 1964. PhD, professor. His main research interests include IoT, sensor network, and multimedia computing.
**马华东**, 1964 年生. 博士, 教授. 主要研究方向为物联网、传感网、多媒体计算.



**Zhang Yuan**, born in 1986. Master. Her main research interests include AI and IoT, and image/video coding.
**张　园**, 1986 年生. 硕士. 主要研究方向为 AI 与物联网、图像/视频编码.