文章编号：2095-2163(2019)06-0254-09 中图分类号：TP391.41 文献标志码：A

基于边缘计算模型的智能视频监控系统的设计

曾德生[1,2]，骆金维[1,2]，庞双龙[1]，谢品章[1]，陈晓丹[1]
（1 广东创新科技职业学院 信息工程学院，广东 东莞 523960；2 广东省教育云 PaaS 平台工程技术研究中心，广州 510630）

摘 要：面对视频监控的应用场景及技术需求，以云计算为代表的集中式数据处理模型在资源需求方面开销较大，过度依赖于云计算中心的网络带宽，在实时性等方面也难于满足视频处理的需求。本文提出一种适用于视频监控场景的边缘计算模型，从计算、网络带宽和存储 3 种主要资源为切入点，设计系统架构，在利用边缘节点的计算能力完成视频的预处理，构建 Docker 容器化平台，采用分级调度策略，降低网络拥塞问题。通过测试，该模型可以有效降低视频监控场景下的计算、存储及网络传输等开销。

关键词：边缘计算模型；智能视频监控；Docker 容器；调度策略

A design of intelligent video surveillance system based on edge computing model

ZENG Desheng [1,2], LUO Jinwei[1,2], PANG Shuanglong [1], XIE Pinzhang[1], CHEN Xiaodan[1]
(1 Information Engeering Institute, Guangdong Innovative Technical College, Dongguan 523960, China;
2 Guangdong Education Cloud PaaS Platform Engineering Technology Research Center, Guangzhou 510630, China)

【Abstract】 Faced with the application scenarios and technical requirements of video surveillance, the centralized data processing model represented by cloud computing has a large overhead in resource requirements, and relies too much on the network bandwidth of the cloud computing center. It is also difficult to meet video processing in real-time and other requirements. This paper presents an edge computing model suitable for video surveillance scenarios. From the three main resources of computing, network bandwidth and storage, the system architecture is designed. Video preprocessing is accomplished by utilizing the computing power of edge nodes, constructing Docker container platform, adoping hierarchical scheduling strategy to reduce network congestion. Through testing, the model can effectively reduce the overhead of computing, storage and network transmission in video surveillance scenarios.

【Key words】 edge computing model; intelligent video surveillance; Docker container; scheduling strategy

0 引 言

近年来，本校以建立智慧校园体系为目标，启动了信息化改造升级工程，其核心目标是建立综合安全监控系统。在图书馆、教室、走廊、校道及其它公共场所部署大量的摄像机，以满足各类安防视频采集的要求。然而，海量视频信息的产生也给信息化系统建设带来了存储、传输等方面的问题。

云计算技术已日趋成熟，应用也越来越广泛。利用云计算技术构建智慧校园体系，采用资源整合的方式，可以降低项目建设成本，为用户带来按需扩展等优点。但集中式资源管理方式，在面对监控系统的应用场景，存在资源需求方面开销较大的问题。海量视频的传输、存储及分析都将消耗云计算中心的大量资源，在实时性方面也难于满足视频处理的需求。

基于上述问题，本文提出在智慧校园体系中，部署基于边缘计算的视频采集框架及容器化应用。在边缘计算模型的视频监控系统中，利用边缘节点部署运动侦测算法，对边缘设备采集的视频流进行运动目标检测等预处理，减少冗余信息，降低系统对存储及传输的需求。部署容器化平台，带来更好的调度特性，解决边缘计算节点的资源调度问题，提高云计算中心的整体性能。

基金项目：广东省教育厅重点平台及科研项目立项(2017GkQNCX130,2017GKTSCX112,2018GkQNCX065,2018GkQNCX111)；教育部职业院校信息化教学指导委员会信息化教学研究课题(2018LXA0070,2018LXB0152)；中国职业技术教育学会信息化工作委员会职业教育信息化建设研究课题(XXHJS19-0022,XXHJS19-0023)。

作者简介：曾德生(1983-)，男，硕士，高级工程师，讲师，ACM 会员，CCF 会员，主要研究方向：Linux、容器、云计算等；骆金维(1980-)，男，硕士，副教授，CCF 会员，主要研究方向：分布式计算、大数据等；庞双龙(1988-)，男，硕士，工程师，主要研究方向：虚拟化、软件定义网络；谢品章(1983-)，男，硕士，讲师，高级工程师，主要研究方向：网络安全；陈晓丹(1983-)，女，硕士，讲师，主要研究方向：云计算、软件定义存储。

通讯作者：曾德生 Email：zengdesheng@ gmail.com

收稿日期：2019-08-02

1　相关工作

1.1　边缘计算

随着技术的发展变革，智能终端的普及，应用场景越来越复杂。以云计算模型为代表的集中式数据处理模式，已经不适用于海量实时数据的处理，如直播、在线教育、智慧城市、智能安防等。针对这一问题，以"数据处理应更靠近数据源头"为核心理念的边缘计算模型应运而生[1]。

边缘计算（Edge Computing，EC）经过近年的发展，其定义和说法有多种，边缘计算产业联盟对边缘计算的定义是指在靠近物或数据源头的网络边缘侧，融合网络、计算、存储、应用核心能力的开放平台，就近提供边缘智能服务，满足行业数字化在敏捷连接、实时业务、数据优化、应用智能、安全与隐私保护等方面的关键需求。它可以作为联接物理和数字世界的桥梁，使用智能资产、智能网关、智能系统和智能服务[2-3]。

1.2　智能视频监控

智能视频监控系统（Intelligent Video Surveillance System，IVSS）是指不需要人工干预的情况下，利用计算机视觉、图像及视频分析算法，对视频采集设备所拍摄的流式视频图像进行分析，对视频中的运动目标进行检测、识别和跟踪，在此基础上对目标进行分析和行为判断，当发现异常情况时进行告警处理[4-5]。

在智能视频监控系统中，一般包含视频采集、图像预处理、运动目标检测、运动目标跟踪、运动目标分类、行为描述与理解和告警处理模块。相较于传统的监控系统，智能视频监控系统具有许多优点：

（1）准确率高。在系统的前端集成视频采集、分析模块，用户可以根据需要对异常情况的特征进行详细定义，降低漏报和误报，提高告警的准确率。

（2）响应速度快。系统中可以实现自动检测功能，识别异常情况，告警模块可以迅速提示安保人员查看监控系统或赶赴现场，可以提高异常事件的处理速度。

（3）可靠性高。系统可以实现 7 * 24 全天候的自动分析处理功能，减少了现场人工监视，避免监视人员因疲劳等情形忽略视频中的异常情况。

在智慧校园体系中，监控系统使用大量的摄像机进行 7 * 24 小时的监控，将产生海量的视频数据。因此，通常在视频采集模块加入运动目标检测算法，在边缘节点对实时视频流进行预处理，提高视频的处理效率，避免了云计算中心产生大量的计算开销；算法可以筛选出有效视频帧，减少监控视频中的冗余信息，有效降低大量视频采集设备获取的海量视频信息对存储空间及网络带宽等方面的开销，达到节约建设成本的目的。

1.3　容器化调度

云计算是一种将计算资源按需供应给用户的新型商业模式，能满足用户复杂的动态资源需求，从而减少用户在购置基础设施及硬件维护成本方面的投入。传统的中心化、粗粒度的虚拟化架构，不太适合于海量视频信息的处理。

Docker 容器技术的出现为云计算以及企业 IT 架构的演进带来了新的革命。Docker 相较于传统虚拟化技术减少了 Hypervisor 层带来的性能消耗，大幅提高了虚拟化性能，为云上部署的计算集群的性能优化提供了良好的基础。实现容器化应用平台。通过提供视频处理的镜像文件，简化了每个边缘计算节点部署的重复性工作，降低了部署的复杂性，实现敏捷化部署。

充分利用 Docker 容器的特性，获取节点的资源状态信息[6-7]，为边缘计算模型的资源调度提供支撑，提高资源利用率，达到节约建设成本的目的[8-9]。

2　视频监控需求分析

2.1　运动检测功能需求

在监控场景中，往往存在大量的摄像机，如果需要及时发现异常情况，通常都要安排大量的人工对视频进行 7 * 24 小时的实时排查。同时，因为视频信息流具有持续性的特征，视频信息的传输也将带来巨大的网络负载；随着时间的推移，监控系统产生的视频数据也将带来巨大的存储压力。如果将这些海量视频数据直接上传到云计算中心，一方面视频信息的处理需要消耗大量的计算资源，另一方面数据的传输和存储也将面临巨大的压力。

监控场景中视频信息的存储，其最主要的目的是记录场景中的变化信息及可疑信息，如果不采用合适的技术或方法，监控系统将占用较大的网络带宽，传输长时间记录无变化的监控场景，有效信息含量低，视频信息也失去了存储的意义。

因此，在智能视频监控系统中，运动检测技术是最基本也是最重要的技术[10-12]，这种技术通过合适的算法，检测视频数据流中的运动目标，替代人工识别的工作。通过设置一定的参数，发现运动目标在

监控场景中的运动特征或位置信息,实现自动告警或判断视频信息是否达到存储或传输备份的要求,可以有效的节省存储空间,降低网络传输的压力。

在各类运动检测算法中以光流法、背景差分法和帧间差分法最为常见。

2.1.1 光流法

光流法最初是由 Horn 和 Schunck 提出,将二维速度场与灰度关联,引入约束方程,得到光流计算的基本算法。光流法算法较为简单,易于实现,但是当光照变化、物体被遮挡时,会影响光流场的分布,将增加算法的运算量,针对实时应用场景时存在一定的缺陷。

2.1.2 背景差分法

背景差分法通过选取特定的图像作为背景帧,然后将当前需要判断的视频帧或图像与背景帧做差分运算,进而判断是否存在运动目标,背景差分算法的处理过程如下:

(1)选取没有运动物体进入监控画面时的图像作为背景帧,定义为 $background(x,y)$;

(2)选取当前需要比较判断的帧,定义为 $frame_k(x,y)$;

(3)设定阈值为 T,将当前帧 $frame_k(x,y)$ 与背景帧 $background(x,y)$ 做差分运算,差分的结果与阈值 T 进行比较,二值化得到运动目标。如果大于阈值 T,则判断有运动目标,如果小于等于阈值 T,则判断为没有运动目标。形式化计算公式表示如下:

$$detect(x,y)=\begin{cases}1, if\left|frame_k(x,y)-background(x,y)\right|>T\\0, others.\end{cases}\tag{1}$$

$detect(x,y)$ 为当前帧与背景帧经过差分运算、二值化后得到的二值图像,仅当 $detect(x,y)=1$ 时,表示侦测到运动目标。背景差分法只需要进行一幅帧的差分检测,速度快,准确度高;但是背景差分算法很大程度上依赖于背景帧 $background(x,y)$ 的可靠性,如果光照、阴影等变化,需要不断的调整背景帧,以适应环境的变化,因此,背景差分算法较适合于固定摄像机。

2.1.3 两帧差分法

两帧差分法通常也称为帧间差分法,其算法设计思路与背景差分相似,采用改进的方式,选取相邻的两帧图像,将当前帧 $frame_k(x,y)$ 与上一帧 $frame_{k-1}(x,y)$ 进行灰度化处理后,进行差分运算,帧间差分法不会受到缓慢光线变化的影响,算法简单易实现。形式化计算公式表示如下:

$$detect(x,y)=\begin{cases}1, if\left|frame_k(x,y)-frame_{k-1}(x,y)\right|>T\\0, others.\end{cases}\tag{2}$$

但只能检测到前后两帧变化的部分,不能检测到重叠部分,易出现边缘模糊不完整等问题,当物体移动缓慢时会出现误判或空洞现象。

2.1.4 三帧差分法

在两帧差分法的基础上,研究学者提出了三帧差分法,基本思路是提取连续三帧图像 $frame_{k-1}(x,y)$,$frame_k(x,y)$,$frame_{k+1}(x,y)$,算法流程如图 1 所示。

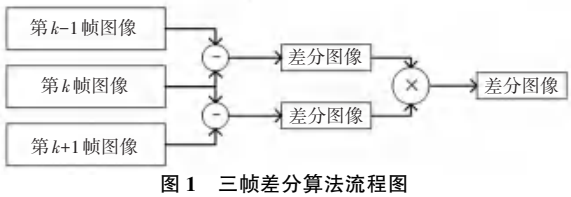


图 1 三帧差分算法流程图

Fig. 1 Flow chart of three-frame difference algorithm

算法计算过程如下:

(1)将第 $k-1$ 帧与第 k 帧按公式(2)进行帧间差分法运算,得到 $detect_1(x,y)$;

(2)将第 k 帧与第 $k+1$ 帧按公式(2)进行帧间差分法运算,得到 $detect_2(x,y)$;

(3)将 $detect_1(x,y)$ 与 $detect_2(x,y)$ 的计算结果进行与运算,形式化的计算公式表示如下:

$$DETE(x,y)=detect_1(x,y)\otimes detect_1(x,y)=\begin{cases}1, detect_1(x,y)\ detect_1(x,y)\neq 0\\0, others.\end{cases}\tag{3}$$

其中,$DETE(x,y)$ 是逻辑与的运算结果,与两帧差分法类似,三帧差分法在检测运动目标的过程中,仍存在空洞现象,但是三帧差分法可以定位出运动目标在监控画面中的位置,提高了运动检测的精确度,检测结果比两帧差分法更准确[13]。

基于上述运动检测算法的分析,在本文的研究过程中,可以采用三帧差分法实现运动检测模块,构建校园网的视频监控系统,降低系统构建的难度,实现对视频信息的选择性存储,筛选出有效视频帧,减少监控视频中的冗余信息,达到提高视频信息有效性的目的。同时,降低大量视频采集设备获取的海量视频信息对存储空间及网络带宽等方面的开销,达到节约建设成本的目的。

2.2 存储与网络带宽需求分析

2.2.1 摄像机码流分析

为满足智慧校园的建设要求,在监控系统中以

选用高清规格的摄像机为主。摄像机的规格各异，产生的数据量也各不相同。以 HD 数字摄像机为例，按 2 048 Kbps 码流进行计算，每个摄像机每小时产生约 900 M 的视频数据，每天产生大约 21 G 新视频数据。三种常见规格的摄像机，产生的视频数据见表 1。

表 1　三种规格摄像机的视频码流情况

Tab. 1　Video stream of three types of cameras

视频规格	码流	1 s/KB	1 h/MB	1 d/GB	30 d/GB
D1 模拟摄像机(704 * 576)	1 024 Kbps	128	450	10.547	316.406
HD 数字摄像机(1280 * 720)	2 048 Kbps	256	900	21.093	632.812
FHD 数字摄像机(1920 * 1080)	4 096 Kbps	512	1 800	42.188	1 265.625

2.2.2　视频数据传输与存储分析

以教学楼为例，楼层建筑为回字形，上下方为走廊通道，左右两边各分布 5 间教室。每个楼层有 10 间教室，上下左右共计 4 个廊道。综合监控系统的建设成本，4 个廊道的两端各选用安装 1 路 FHD 数字摄像机，在每个教室中选用安装 2 路 HD 数字摄像机，每个楼层合计 28 路两种规格的摄像机。

在系统正常运行时，智能视频监控系统中的每路摄像机都将实时产生两个数据流，用于实时监控传输的视频数据流和视频数据文件传输至云计算存储中心的存储数据流。当发生应急情况时，监控中心的工作人员需要实时查看视频监控内容，云计算的存储中心也将实时存储视频信息，监控系统的网络带宽将达到最大需求。

以表 1 中摄像机规格为例，当应急情况发生时，每个楼层实时产生的最大视频数据流和最大存储数据流均为：4 Mbps * 8 路+2 Mbps * 20 路 = 72 Mbps。因此，按楼层计算，最大的网络带宽需求合计为 144 Mbps，对云计算中心的存储写入的最大速率要求为 72 Mbps。按相同的建设规格进行部署，若整幢建筑物发生应急情况，以 6 个楼层进行计算，最大的传输带宽需要 864 Mbps，最大存储写入速率需求为 432 Mbps。

2.3　管理需求分析

智能视频监控系统一般都包含了视频采集、图像预处理、运动目标检测、运动目标跟踪、运动目标分类、行为描述与理解和告警处理模块，相应管理功能的需求也是围绕上述模块进行设计，提升系统的准确率和响应速度，提高系统的整体可靠性，为管理人员提供便捷的管理功能[5]。

基于边缘计算模型的视频监控系统，相较于传统的 IVSS，除包含上述所涉及的功能，采用边缘计算模型，分散式的在边缘节点对视频信息进行预处理，存储初期数据。因此，设计采用容器技术，构建容器化的资源调度平台，采用合适的策略，实现网络带宽资源的调度控制，上传视频文件至云计算中心，实现视频文件的备份存储，降低网络负载和存储空间的需求，提高资源利用率，达到降低系统建设成本的目的。

3　方案设计

3.1　体系架构设计

以教学楼的结构进行体系架构图设计，基于边缘计算模型的智能视频监控系统规划为以下 4 个角色：边缘计算单元(Edge Computing Unit，ECU)、智能视频监控单元(Intelligent Video Surveillance Unit，IVSU)、边缘计算节点(Edge Computing Node，ECN)，云计算数据中心(Cloud Computing Data Center，CCDC)。体系架构如图 2 所示。

(1) ECU。边缘计算单元，具有一定计算能力，可以实现对摄像机采集的视频信息进行预处理，并提供文件存储及网络传输功能。在后续模型验证中，采用树莓派 Zero W 单板计算机，底层安装 Linux，为后续提供 Docker 调度接口。

(2) IVSU。智能视频监控单元，在 ECU 模块的基础上，安装 motionEyeOS 及 CSI 摄像头实现智能视频监控单元，部署视频采集点，如：教室、走廊、实训室、图书馆等。

(3) ECN。边缘计算节点，具有较高的计算能力，并可以提供较大的存储空间，用于临时或长期存储 IVSU 产生的视频数据，可以根据摄像机的数量选择服务器或其它通用计算机作为硬件支撑环境，易于部署容器化平台，为后续资源调度及模型验证提供支持。

(4) CCDC。云计算数据中心，部署大量的服务器及存储硬件，采用 KVM、VMware 等平台或工具，构建云计算资源管理中心，为智能视频监控系统存储海量的视频信息。

3.2　视频采集框架设计

进行视频采集框架设计时，在保障可用性的情况下，采用树莓派作为视频采集框架的硬件基础设施，降低建设成本。基于开源平台部署容器，在去中心化环境中，提供调度策略的支持。

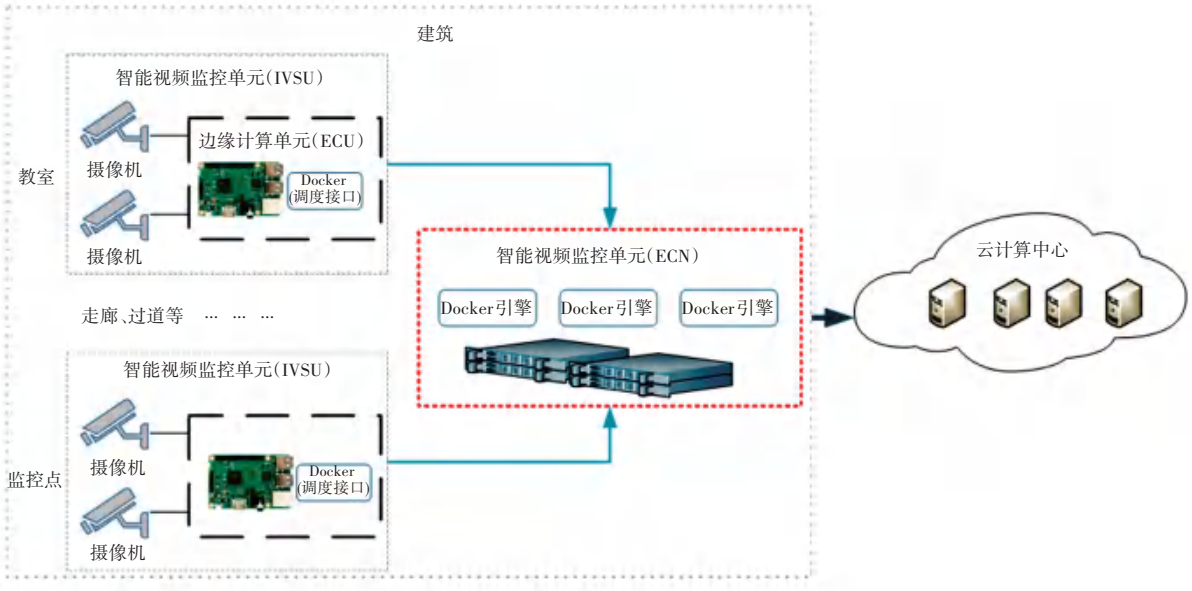


图 2 基于边缘计算模型的智能视频监控系统体系结构图

Fig. 2 Architecture diagram of IVSS based on edge computing model

3.2.1 开放式硬件架构

树莓派(Raspberry Pi)是一个开放式,易于扩展的小型的单板计算机,功耗低,可按需定制,提供所有预期的功能或能力。广泛用于实时图像、视频处理和基于 IoT 的各类应用程序。视频采集框架采用由树莓派 Zero W 作为核心组件的边缘节点,通过加载 CSI 摄像机,用于实现监控系统的视频采集功能[14]。

树莓派 Zero W 单板计算机,设计紧凑,功耗低,通过 Micro USB 接口供电,作为视频采集成本低。该单板计算机采用 BCM2835 作为 SoC,集成了通用计算机中的各类功能。在计算模块中,采用 ARM1176JZF-S,提供了 700MHz 的计算能力;在视频模块中,采用了 Broadcom VideoCore IV 技术,能够实现每秒 30 帧的 1080p 的 H.264 视频编码或解码,同时提供了 mini HDMI 输出功能;在网络连接方面,提供了 Wifi 及蓝牙模块,支持 802.11n 连接;在其他接口方面,提供了 1 个 mini USB On the Go 接口、一个 Micro SD 卡的接口和 40pin 的 GPIO 接口,针对摄像机提供了 CSI 接口,可以适配树莓派 Camera Module V2 摄像机,采集高达 800 万像素的高清视频。

3.2.2 开源视频采集系统

motionEyeOS 是一套嵌入式操作系统,系统采用 BuildRoot 工具完成交叉编译,适合于部署在单板计算机上,提供实现完整的视频监控系统。视频监控系统的前端是采用 Python 编写的 motionEye 程序,提供 web 接入功能;后端采用高度可配置的 motion 程序,可以实时查看视频流,也可以实现面部识别、动态监测、摄像机直通录制、记录活动图片和创建动态视频文件等功能[15]。在 2.1 节的需求分析的基础上,对 motionEyeOS 进行改进,简化流程,采用三帧差分算法,实现运动检测功能。

3.3 容器化资源调度方案设计

在调度方案的设计过程中,负载是影响应用资源需求的主要因素。结合智能视频监控系统的实际应用,系统瓶颈主要集中在网络及磁盘 I/O 方面。因此在设计调度方案时,先利用 Docker 容器引擎的特性,通过周期性的采集 CPU、内存、磁盘 I/O、网络带宽等资源负载情况信息,为容器化调度方案提供信息支撑。

在调度模式上,充分体现边缘计算模型分散处理的特点,针对图 1 所示的体系架构,采用两级调度模式,即云计算数据中心(CCDC)对边缘计算节点(ECN)进行调度,边缘计算节点(ECN)对边缘计算单元(ECU)进行调度。执行调度任务时,发起的一方为主动调度对象(Active Scheduling Object,ASO),另一方为被动调度对象(Passive Scheduling Object,PSO)。

$$Aso_i = [A_{cpu} \quad A_{mem} \quad A_{net} \quad A_r \quad A_w \quad A_{tasked}]^{\mathrm{T}}. \tag{4}$$

在式(4)中,Aso_i 表示对应 CCDC 或 ECN 中,第 i 个主动调度对象(ASO)。A_{cpu} 表示 ASO 中的 CPU 资源的剩余情况,A_{mem} 表示 ASO 中的内存剩余情况,A_{net} 表示 ASO 中网络带宽的剩余情况,上述 3 种资源以百分比计算,取值范围为 (0,100) ;A_r 表示 ASO 中磁盘 I/O 操作的读状态;A_w 表示 ASO 中磁

盘 I/O 操作的写状态；A_{tasked} 表示当前 ASO 中是否被上一级执行调度任务，结果为逻辑值，设定为（*Ture* | *False*），*Ture* 表示当前正在被执行调度任务，*False* 表示未被执行调度任务。

$$Pso_j = [P_{cpu}\ \ P_{mem}\ \ P_{net}\ \ P_r\ \ P_w\ \ P_{tasked}\ \ P_{resp}\ \ P_{ltime}\ \ P_{size}]^{\mathrm{T}}. \quad (5)$$

在式(5)中，Pso_j 表示对应 ECN 或 ECU 的第 j 个被动调度对象(PSO)。P_{cpu} 表示 PSO 中 CPU 资源的剩余情况，P_{mem} 表示 PSO 中内存资源的剩余情况，P_{net} 表示 PSO 中网络带宽剩余情况。与公式(4)类似，上述 3 种资源以百分比计算，取值范围为(0,100)；P_r 表示 PSO 中磁盘 I/O 操作的读状态，P_w 表示 PSO 中磁盘 I/O 操作的写状态；P_{tasked} 表示 PSO 是否被上一级执行调度任务，结果为逻辑值，设定为（*Ture* | *False*），*Ture* 表示当前正在被执行调度任务，*False* 表示未被执行调度任务；P_{resp} 表示从被 PSO 到 ASO 之间的网络状态；P_{ltime} 表示 PSO 上一次成功被执行调度任务的时间，P_{size} 表示需要调度处理的文件大小。

$$Task_{ij} = [T_{stime}\ \ \ T_{pri}\ \ \ T_{exectime}]^{\mathrm{T}}. \quad (6)$$

在式(6)中，$Task_{ij}$ 表示主动调度对象 Aso_i 发起调度任务至被动调度对象 Pso_j 的执行参数。执行步骤如下：

(1)预设系统调度周期的时间长度为 $Sche_{time}$，然后通过系统调用，取得系统当前时间，作为调度任务的启动时间：T_{stime}。

(2)将启动时间 T_{stime} 与 Pso_j 的 P_{ltime} 参数进行计算，获取调度优先级，公式如下：$T_{pri} = (T_{stime} - P_{ltime}) \div Sche_{time}$，$T_{pri}$ 的数值越大，则优先级越高，需被调度紧急程度越高。

(3)根据网络状况及 Pso_j 的 P_{size} 参数进行计算，假设 Aso_i 与 Pso_j 之间的网络状态处于理想状态，则可以估算调度任务的最小执行时间，$T_{exectime} = P_{size} \div (A_{net} \mid P_{net})$，$A_{net}$ 与 P_{net} 取其中的最小值。

(4)调度策略采用较为简洁的权重轮询调度(Weighted Round-Robin Scheduling)算法，以 T_{pri} 作为权重，通过轮询方式在边缘计算模型中的各节点间进行调度。根据调度任务的完成情况，设定 $Task_{ij}$ 的值为逻辑值（*Ture* | *False*），*Ture* 表示当前任务已完成，*False* 表示调度任务未成功。

4　模型测试

4.1　测试环境

为减少对正常教学秩序的影响，选择暑期前进行测试，测试地点为全天对学生开放的信盈达 CDIO 智创工作室。基于图 2 所示的体系架构图进行部署，视频采集框架采用 3.2 节所述的设计，部署 ECU 节点，并安装摄像头，配置网络，构建 IVSU。相关设备及主要参数见表 2。

表 2　智能视频监控单元设备及主要参数

Tab.2　IVSU equipment and its main parameters

序号	设备(软硬件)	主要参数
1	树莓派单板计算机(Zere W)	CPU:1 GHz,单核;内存:512 M 网络连接:802.11 b/g/n;存储接口:microSD 摄像头接口:CSI;供电接口:micro USB
2	CSI 摄像头(SONY IMX219)	传感器:8-megapixel CMOS 静态图片规格:3280 * 2464(最大) 视频录制规格:①720p 60 fps; ②1 080 p30 fps
3	存储设备	microSD:256G,金士顿 KF-C38256-4K I/O 性能:①读取速率:100 MB/s;②写入速率:80 MB/s
4	motionEyeOS	Release:20190427;Motion:4.2
5	无线路由器(TL-WAR450L)	传输频段:2.4 GHz;传输速率:450 M LAN:千兆网口;无线增益:5 dBi
6	交换机	华为,S1700-24GR,24 端口

在工作室完成 IVSU 的部署后，IVSU 设备采集的视频通过无线网络进行数据传输；楼层的设备间部署边缘计算节点(ECN)；再经校园网络传输至云计算数据中心(CCDC)，ECN 设备采用一台 DELL 服务器，其主要参数见表 3。

表 3　边缘计算节点设备及其主要参数

Tab. 3　ECN equipment and its main parameters

选项	参数/型号	备注信息
服务器	DELL PowerEdge R510	--
CPU	Xeon X5650 * 2 颗	2.60 GHz,单颗 CPU 具有 6 个物理核心,支持超线程
内存	32G	--
硬盘	1T * 4 块	组建 RAID-0 阵列,提高 I/O 性能
网络	1 000 Mbps	--
磁盘整列卡	Dell PERC H700	512M 缓存
操作系统	CentOS-7-1810	内核版本:3.10.0-957.12.1;Docker 版本:1.13.1;Python 版本:2.7.5;PHP 版本:5.4.16;数据库版本:MariaDB-5.5.60;Web 服务器:Apache-2.4.6

在本次测试过程中，未进行云计算数据中心(CCDC)的设计与部署，在学校现有的云计算数据中心申请一台虚拟机(VM)，安装配备相应的软件

环境,作为模拟 CCDC,用于存储数据,实现调度功能,其参数见表 4。

表 4 云计算数据中心主要参数

Tab. 4 Cloud computing data center main parameters

选项	参数/型号
CPU	Xeon E5-2609 v2(2.50 GHz) 8vCPU core
内存	64 G
硬盘	2.17TB
网络	1 000 Mbps
操作系统	CentOS-7-1810

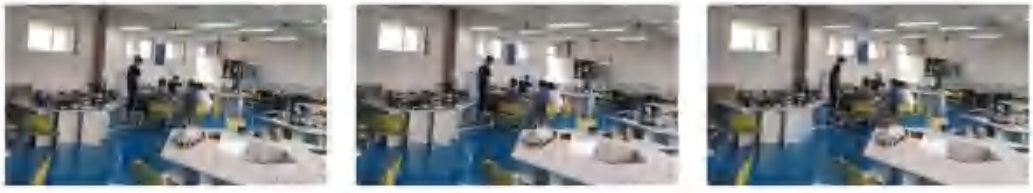
4.2 模型测试

4.2.1 运动检测功能验证

利用 IVSU,录制一段视频。提取其中第 1057 帧、第 1058 和第 1059 帧进行测试,如图 3 所示。

将上述的三帧图像,进行二值化处理后,如图 4 所示。

在本文设计的模型中,采用三帧差分算法进行运动目标检测,将不变化的进行二值化处理,转化为黑色背景,通过算法提取的运动目标,轮廓较为清晰完整,结果达到预期的运动检测需求,如图 5 所示。



(a)1057 帧 (b)1058 帧 (c)1059 帧

(a)Frame 1057 (b)Frame 1058 (c)Frame 1059

图 3 连续三帧图像

Fig. 3 Three consecutive frames

(a)1057 帧 (b)1058 帧 (c)1059 帧

(a)Frame 1057 (b)Frame 1058 (c)Frame 1059

图 4 二值化处理的图像

Fig. 4 Binarized image

图 5 三帧差分算法检测结果

Fig. 5 Detection results of three-frame difference algorithm

4.2.2 存储需求验证

根据第三部分的设计,同时也为了简化存储文件的管理,采用分时间段存储视频流的方式,记录视频信息。以每 300 s 为一个时间段,即 5 min 的时间长度为一个视频文件。当存储的视频帧中检测发现运动物体后,标记视频记录文件的状态,存储视频信息。连续 5 min 视频帧中没有运动物体时,删除无运动状态的视频文件,达到节约存储空间。

信盈达 CDIO 智创工作室的标准开放时间是从早上 07:30 至夜间 23:00。在部署 IVSU 前,调查学生进入工作室的规律,工作室内存在活动的时间主要集中在早上 07:30 至 12:15,下午 14:10 至晚上的 23:00,其余时间活动较少。每天无活动时间,平均约 10 h 25 min,占全天比例约 43.40%。经过 6 月 10 日至 7 月 7 日,持续 4 周的测试,表 5 中的第 1 周至第 2 周为学校的正常教学周次;第 3 周为考前复习周;第 4 周为考试周。视频监控记录情况如表 5 所示,表中数据为文件个数,每个视频文件长度为 5 min。

通过上述的测试数据可以看出,视频记录文件的数量与学生活动成正比关系。在学校的正常教学

周次，学生进出工作室及在工作室内的活动较为规律，视频存储数量基本持平，临近期末数量稍有增长。第 3 周对应为考前复习周次，学生在工作室中的实践活动增加了，数量有所增加。第 4 周为考试周，工作室内的活动降低较为明显，且第 4 周的周五下午开始放假，关闭工作室后，视频记录文件数量记录为 0 个。

表 5　视频监控记录情况

Tab. 5　Video surveillance record statistics

星期	周次			
	第 1 周	第 2 周	第 3 周	第 4 周
一	160	164	168	55
二	162	163	169	35
三	162	167	172	25
四	162	163	179	21
五	163	163	179	10
六	167	169	180	0
日	168	167	181	0

统计分析其中的数据，对比加入运动检测功能的视频监控，以第 1 周至第 3 周的数据做分析对比未加入运动检测功能的视频监控，其所存储的视频文件数量约为每周 1176 个记录，约节省 41.67%的存储空间。因此，如果在全校范围内开展应用，在存储方面可以明显降低建设成本。

4.2.3　调度任务验证

在进行测试过程中，暂未全校性部署智能视频监控系统，调度验证以边缘计算节点（ECN）至边缘计算单元（ECU）的测试为主，检测智能视频监控系统对网络带宽的利用情况。

利用 Docker 容器引擎的轻量化等特性，通过周期性的采集 CPU、内存、磁盘 I/O、网络带宽等资源负载情况信息，为容器化调度方案提供信息支撑。

假设共有 n 个 ECU 节点，则 ECU 列表为：$ecu = \{ecu_0, ecu_1, \cdots, ecu_{n-1}\}$，$weight(ecu_j)$ 表示第 j 个 ECU 节点的权重，即公式（6）中计算后的的 T_{pri}，j 也表示为上一次调度的对象 Pso_j，$max(ecu)$ 表示所有节点中的最大值。$gcdnumber(ecu)$ 表示 ECU 列表中所有节点权值的最大公约数。变量 j 初始化为-1，$curweight$，表示当前的权重，初始化为 0。

调度执行伪代码如下：

```
while (true) {
j = (j + 1) mod n;
    if (j == 0) {
    curweight = curweight - gcdnumber(ecu);
    if ( curweight <= 0) {
    curweight = max(ecu);
        if (curweight == 0)
          return NULL;
    }
}
}
```

增加相应的权重，判断各节点的优先级，避免某个 ECU 节点的数据长期未备份。同时，通过相应的权重，充分考虑网络流量的负载等情况，避免边缘计算模型中个 ECU 节点的集中调度，导致网络拥塞的情形，降低整体的网络建设成本。

5　结束语

本文提出的方案引入边缘计算模型，在校园网内有机融合云计算数据中心与边缘计算环境。采用开源及开放式软硬件架构，充分利用边缘计算节点的计算资源，实现运动检测功能，有效降低监控系统对存储空间的需求；并利用 Docker 容器化平台收集各节点资源状态信息，设计资源调度策略，提高网络带宽的利用率。智能视频监控技术是大数据时代监控技术发展的趋势，因此，在后续的工作中，将结合 ECU 的计算能力，在智能视频监控系统中，实现目标识别及跟踪等功能；优化存储空间，采用分布式弹性存储机制，充分利用 ECN 的存储能力。

参考文献

[1] 李林哲，周佩雷，程鹏，等. 边缘计算的架构、挑战与应用[J]. 大数据，2019，5(2)：3-16.

[2] 工业互联网产业联盟. 边缘计算架构 2.0 白皮书[EB/OL]. http://www.aii-alliance.org/index.php? m=content&c=index&a=show&catid=23&id=182.

[3] 阿里云计算有限公司，中国电子技术标准化研究院等. 边缘云计算技术及标准化白皮书(2018)[EB/OL]. http://www.cesi.cn/201812/4591.html.

[4] 黄凯奇，陈晓棠，康运锋，等. 智能视频监控技术综述[J]. 计算机学报，2015，38(6)：1093-1118.

[5] 袁国武. 智能视频监控中的运动目标检测和跟踪算法研究[D]. 云南大学，2012.

[6] BELLAVISTA P，ZANNI A. Feasibility of fog computing deployment based on docker containerization over raspberrypi[C]. ACM，2017.

[7] 刘倍雄，骆金维，陈孟祥. 数据库集群系统多指标动态负载均衡技术研究[J]. 电子设计工程，2018，26(22)：19-22.

[8] 彭丽苹，吕晓丹，蒋朝惠，等. 基于 Docker 的云资源弹性调度策略[J]. 计算机应用，2018，38(2)：557-562.

[9] 郝庭毅，吴恒，吴国全，等. 面向微服务架构的容器级弹性资源供给方法[J]. 计算机研究与发展，2017，54(3)：597-608.

(下转第 265 页)