doi: 10.7690/bgzdh.2025.05.010

基于边缘计算的多集群容器云弹性资源调度方法

李　金[1]，刘科孟[1]，高红亮[1]，樊腾飞[1]，谢　虎[2]
（1. 中国南方电网电力调度控制中心自动化处，广州 510530；
2. 南方电网数字电网研究院数字电网分公司，广州 510560）

摘要：为解决集中式云计算技术不能实现大量边缘数据的运算带宽及不能保证应用的隐私性和实时性等问题，对边缘容器云负载在多集群条件下的时间差异及存在时延敏感性需求差异的边缘应用进行系统分析，并提供一个主从模型管理的多集群边缘云架构。对时延敏感性运用的相关资源调配情况进行深入研究，通过比较存在的响应式策略，能够有效实现已经提出的相关研究；关于时延敏感应用的问题，采用在负荷上沿超前扩展，抑或在负荷下行时进行滞后缩容，以切实达到应用质量的需要。研究结果表明：边缘计算模式采取分布式，可提高在实际应用周围下沉云中的相关计算能力，能降低云中心自身的运算负荷，减轻核心骨干网带宽压力。

关键词：边缘计算；资源调度；时延敏感；分布式模型
中图分类号：TP393.027　　**文献标志码：**A

Elastic Resource Scheduling Method for Multi-cluster Container Cloud Based on Edge Computing

Li Jin[1], Liu Kemeng[1], Gao Hongliang[1], Fan Tengfei[1], Xie Hu[2]
(1. *Automation Division of Power Dispatching Control Center of China Southern Power Grid*, *Guangzhou* 510530, *China*;
2. *Digital Grid Branch of Digital Grid Research Institute of China Southern Power Grid*, *Guangzhou* 510560, *China*)

Abstract: In order to solve the problem that the centralized cloud computing technology can not achieve the computing bandwidth of a large number of edge data and can not guarantee the privacy and real-time of the application, this paper systematically analyzes the time difference of the edge container cloud load under the condition of multi-cluster and the edge application with different latency sensitivity requirements, and provides a multi-cluster edge cloud architecture managed by a master-slave model. The resource allocation related to the delay-sensitive application is studied in depth, and the proposed related research can be effectively realized by comparing the existing responsive strategies; for the delay-sensitive application, the load is extended in advance at the upper edge, or the lag is reduced at the lower edge of the load, so as to effectively meet the needs of the application quality. The results show that the distributed edge computation mode can improve the related computing power in the sink cloud around the actual application, reduce the computing load of the cloud center itself, and reduce the bandwidth pressure of the core backbone network.

Keywords: edge computation; resource scheduling; delay-sensitive; distributed model

0　引言

为更好地实现在边缘侧的数据处理，边缘计算的概念于是顺理成章地出现了[1]。边缘计算和以往集中式的云计算结构不同，前者的计算模式采取了分布式结构，采取边缘设备的存储资源以及计算能力，可以有效地管理、过滤边缘数据，并做出决策，以显著降低网络带宽压力以及云中心服务量[2]。

边缘计算成为一个全新的运算模式，离不开虚拟化应用科技的支撑[3]。近年来，一个轻量级的虚拟化应用科技——容器虚拟化技术引起了普遍重视[4]。容器技术采用沙盒机制，达到了进程级隔离的目的，体现出了显著优势，如物理主机资源占用不多，启动速率快，移植能力强大等[5]。与主机级虚拟化技术相比，容器虚拟化技术更有益于变成边缘应用方面的关键载体，更适宜于化解边缘情景下繁杂度越来越高的应用部署以及网络集群管理等问题[6]。

基于此，笔者在多集群条件下，研究边缘容器云负载的时间差异和敏感差异并给出主从模型，深入研究存在资源调度问题的时延敏感应用，采取边缘计算技术，侧重探讨以边缘计算技术为基础的多集群容器技术云和相关的资源调配机理。

1　材料与方法

1.1　仪器与设备

2 核 8G PC 机；DELL PowerEdge 820 256G 服

收稿日期：2024-08-23；**修回日期：**2024-09-21
基金项目：南方电网公司重点科技项目(0000002021030101XT00045)
第一作者：李　金(1979—)，男，甘肃人，硕士。

务器；DELL PowerEdge R73 96G 服务器；1 核 2G 腾讯云服务器。其中各硬件对固态存储的要求如表 1 所示。

表 1　各硬件设备固态存储容量

硬件名称	宏碁 PC 机/T	DELL Power Edge 820/G	DELL Power Edge R73/T	腾讯云服务器/G
固态容量	1	600	8	50

1.2　实验方法

1.2.1　边缘计算定义

为打破云计算技术本身的不足，工业界和科学家开始越来越重视边缘计算技术[7]。边缘计算从被发明至今尚缺乏一致认可的概念。简单解释，边缘计算就是一个利用集合边缘的储存、运算等一系列设备，在网络边缘处塑造出边缘云，并且能够为边缘应用提供信息技术工作环境以及云端服务的一类新运算模式[8]。边缘云通过把所有资料汇集在一起，在使用者和中央的云间进行辅助决策、信息处理等业务[9]。

1.2.2　边缘计算与云计算的关系

边缘计算并非是要代替云计算技术，而只是对云计算能力的补充与扩展[10]。边缘计算对相关信息进行预处理的工作，等同于人类神经纤维末端面向外部激励做出的非条件反射，但云计算对信息的历史分解与挖掘，则取决于人脑的逻辑处理功能与记忆能力[11]。表 2 展现了边缘计算与云计算之间在信息分布特征、运算方式等领域的区别。

表 2　云计算与边缘计算的对比

分类	云计算	边缘计算
计算方式	集中	分布
网络位置	中心	边缘
硬件供给	完备	不足
网络延迟	大	小
位置感知	无	有
任务种类	批量大任务	轻量小任务

1.2.3　Kubernetes 架构

2015 年 7 月推出的开源项目之一就是 Kubernetes，是由谷歌公司的 Bory 这个内部项目管理发展起来的[12-13]。Google 正式推出 Bory 前，曾经借助 Bory 信息技术，对内部近 10 亿容器进行管理，而且使内部业务有条不紊地进行。因为 Kubernetes 具有相对复杂的结构，谷歌公司为集中有效地管理集群计算机资源，对存储容器进行了高度封装，如图 1 所示。

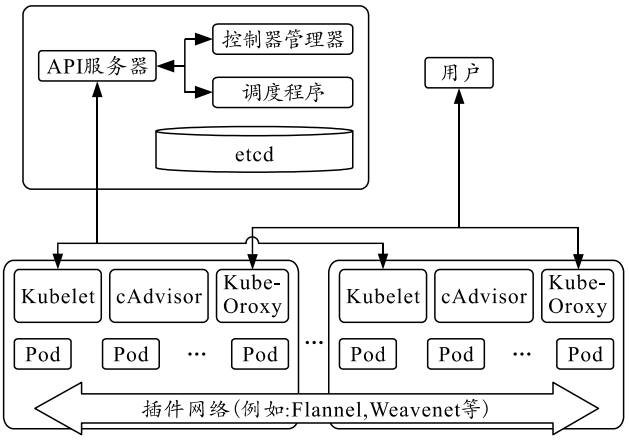


图 1　Kubernetes 架构

从 Kubernetes 集群来看，全部管理结点被划分为 Nadeo 以及 Master 2 大类型。Master 是各集合所有的管理结点，要做好整个集群的控制与调度工作，Master 结点就是负责执行对集合全体所有操作的命令[14-15]。除了 Master 节点，其他任何节点均属于普通的 Node 节点，重点担负集群本身的任务运行以及承载[16]。

1.2.4　多集群边缘云框架设计

鉴于传统边缘云框架的缺陷，以及面向集群负载的应用时延敏感性差异以及时间差异，按照主从模型，研发多集群边缘云框架时，要以应用时延敏感差异为基础，便于对多集群边缘云进行集中管理和资源调配。多集群边缘云框架设计如图 2 所示。

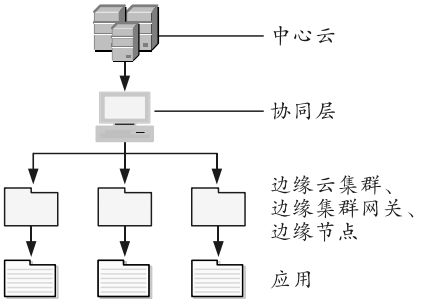


图 2　多集群边缘云部署架构

图 2 中，多集群边缘云系统通过增加协同层，并通过主从模型实现管理功能。

1.2.5　时延敏感应用调度问题概述

时延敏感技术具有代表性的特征，应在规定时间内对用户要求进行反馈。时延敏感应用进行一次请求的过程如图 3 所示。

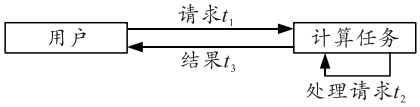


图 3　一次时延敏感应用请求的响应时延

图 3 中用户发出请求时间、应用处理请求时间、边缘云返回结果给用户的时间分别为 t_1、t_2、t_3。对每个请求，边缘应用都花费对应的系统资源进行处理，t_2 可用式(1)估计：

$$t_2 = \text{size}/\text{mips}_v \text{。} \tag{1}$$

式中：size 为请求所需要的计算量；mips_v 为应用从边缘云中通过虚拟化技术获取的计算功能。

时延敏感应用的资源调度问题也可以用水平扩展的方法来处理。如式(2)所示：

$$\text{mips}=n\times\text{mips}_v \text{。} \tag{2}$$

在水平扩展中，程序的总计算能力会随着程序所获得的副本总量增加而增加。因为各个边缘集群所具有的资料总量是有限的，所以在各个时候都要按照特殊应用领域实际情况所需要的资源量，产生相应总量的资料副本。

综上，存在资源调度问题的时延敏感应用的问题，也可看成是处理一个时刻按照要求产生对应副本的相关问题。

1.2.6 时延不敏感应用调度问题描述

时延不敏感应用的重要特征是未能严格地提出对应用的请求必须在规范的时限内应答，而是安排到边缘云中，预处理数据信息脱敏、滤波等工作。因为水平资源为数不多，边缘群体必须最先解决群体中时延敏感应用的水平资源扩容需求，而后再尽量解决时延不敏感应用的水平资源需要，在调度时延不敏感应用的相关资源方面，能够理解为按照哪些标准，在什么时间把应用资源调配至哪个群体中的问题。

2 结果与分析

2.1 实验敏感应用资源调度测试

根据时延敏感技术应用资源调度测试方法，实验人员通过对实际网络系统执行压力访问，并观测灰色和移动水平 pod 自动缩放策略(gray and moving for horizontal pod autoscaling strategy，GMHPA)和 pod 水平自动扩容(horizontal pod autoscaler，HPA)战略下使用的各种资源副本总量的变动状况，及其应用的响应时间比较。因为时延敏感技术应用的动态伸缩都为唯一元素激发，所以在产品设计上就选用了易于产生应变的 CPU 来作为试验，并且 CPU 的资源配额单元是 m，约为每个 CPU 核的 0.001。

试验第 1 步是提供了一种计算密集的 Dokcer 容器的应用，每个请求的应用都可以完成一次 π 运算。接着，分别将该程序的 GMHPA 战略与 HPA 战略部署在 Cluster 1 中，完成了对比测试。设置 GMHPA 策略参数中，监控周期 T 选 30 s，最小副本数取 1，最大副本数取 5，伸缩指标取 0.7，资源配额取 100 m，灰色队列长度取 6，移动队列长度取 3，加权平均权重 β_1、β_2、β_3 分别取 0.6、0.3、0.1，缩容判断次数 th 取 4，利用并程的虚拟请求量平滑、稳定增长、迅速增加、快速减少和稳定降低过程，监测程序的真实负荷、预期负载、最大副本数量及其响应时延的变化情况，负载的实际结果与预测结果如图 4 所示。

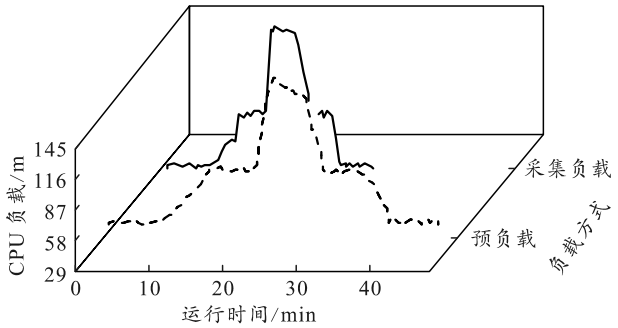


图 4 采集负载与预测负载

由图 4 不难看出：在整个负荷连续变化中，GMHPA 策略能够基本拟合真实的负荷曲线，但图中也发现第 28 min 时负荷从迅速上升到相对稳定的拐点处，GMHPA 策略预测算法存在相应偏差。同样，因为设定了一定次数的缩容评估和引入移动加权平均模式，在第 31 min 负荷从迅速降低到稳定变动的时期，预测值也不会发生大幅降低。图 5 为 GMHPA 策略下应用的副本变化情况。

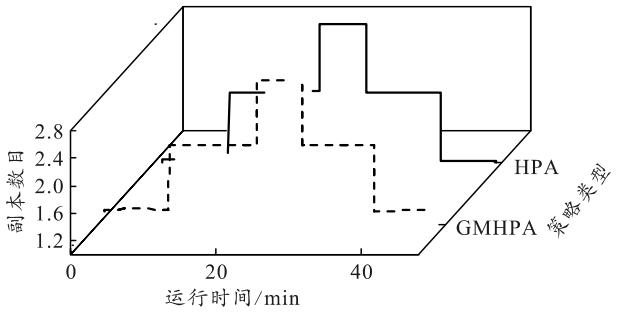


图 5 副本变化情况

如图 5 所示，应用刚完成初始化时只有一个副本实例。但由于负载的持续增加，在第 9 min 应用第 1 次导致扩容，在第 22 min 应用第 2 次导致扩容。而在负荷下降阶段，由于 GMHPA 机理在缩容时考虑了饱和型资源供给任务，在经过几次灰色预测负荷均较目前负荷低时，才确定了灰色模型结论，不

然就选用移动加权平均模式。与此同时，实验还描述了在 2 种情况下的总反应时间。实验结果如图 6 所示。

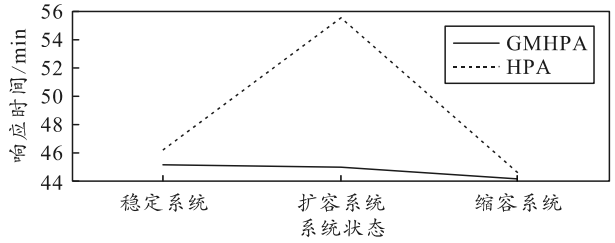


图 6　GMHPA 策略和 HPA 策略的响应时间对比

图 6 中，当触发扩容时，HPA 战略较 GMHPA 战略其平均响应时间提高了约 11 ms，主要就是由于 HPA 战略采用反应式的特性，在感受到负荷增加后才进行了扩展，扩展发生的同时应用也花费了等量的时间初始化；因此，导致应用的所需资料供应不准确，进而导致延长了平均响应时间。而 GMHPA 战略则可以更高效地利用预测算法，实现了提前扩展目标，从而最好地满足时延敏感应用的资源要求。

2.2　时延不敏感应用资源调度测试

时延不敏感应用资源调整实验研究的是，在多集群条件下，通过提高单个群体的时延敏感性应用负载，对调整延迟不敏感跨集群调度(delay insensitive cross-cluster scheduling，DICCS)策略的相关过程进行考察，并分析调整前后群体的负荷变动状况。

首先，在 Cluster 1 中部署了 3 个端到端时延的不敏感应用；然后，通过 Kubernetes 应用配置文件中 resources.limits.CPU 和-cpus 参数分配给固定 CPU，并进行 resources.limits.Meory 和-mcm-total 参数分配固定 RAM。其 CPU 和 RAM 资源配额如图 7 所示。

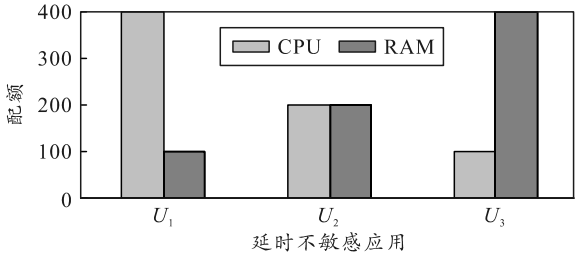


图 7　Cluster 1 中时延不敏感应用的资源占用

采用相同方式在群体中部署应用，并初始化群体负荷，使其都达到最低负荷集群。接着在 Cluster 1 中部署一组时延敏感应用，通过操作系统内部的监视控制模型测试 Cluster 资源利用率，并在每个监控时间内产生一次新副本，直到 Cluster 1 变成高负荷的集群计算机。试验流程中，对时延敏感性应用领域及不同资源需求设定了 3 种对照试验模拟，3 种试验用到的时间延迟及敏感应用领域资源配额如图 8 所示。

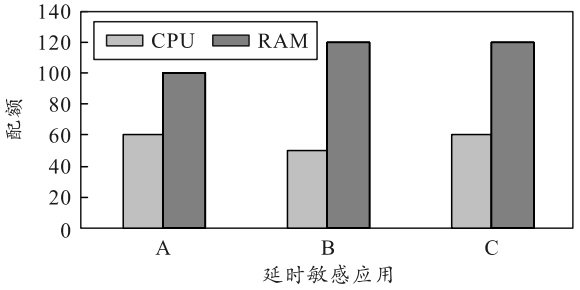
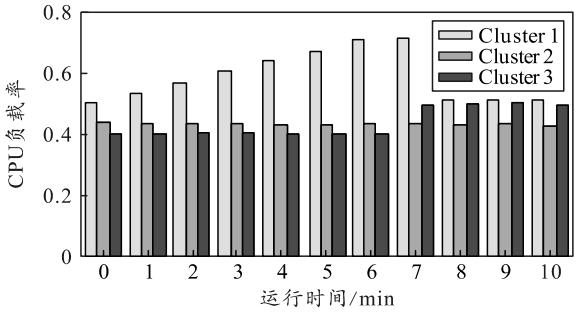


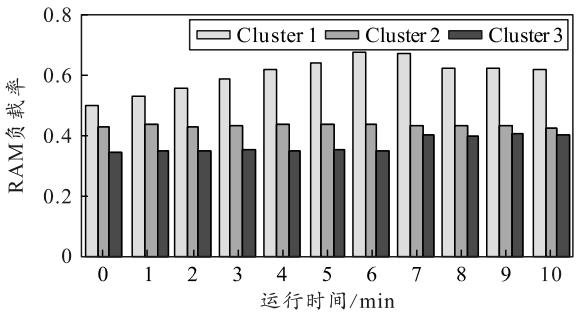
图 8　不同组时延敏感应用副本的资源配额

为获取 DICCS 策略在测试过程中所使用到的参数，监控周期 T 取 1 min，临界阈值取 0.6，高负载阈值取 0.7，应用权重均取 0.5，集群权重分别取 0.4，0.4，0.2。

A 组实验测试结果如图 9 所示。从第 1 min 起，Cluster 1 持续增大时延敏感应用的副本量，第 6 min Cluster 1 中 CPU 资源负载超过最大负荷的闭值，从而引发了调度。按照单资源因素触发准则，选取对 CPU 占有量较高的时延不敏感应用 U_1 作为待调整使用；选取了 Cluster 3 作为目的集群，并向其发出调度命令，尝试将 U_1 资源部署到 Cluster 3。第 7 min，监控模块发现应用资源在 Cluster 3 中部署完成，并回收了 Cluster 1 中的 U_1 占用资源。在第 8 min，观测到 Cluster 1 集群中的 U_1 应用资源已被有效利用，Cluster 1 已成为低负荷集群。



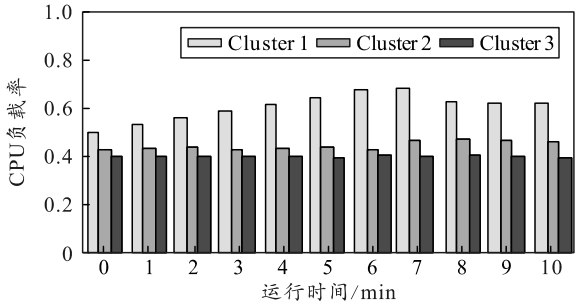
(a) CPU 负载率



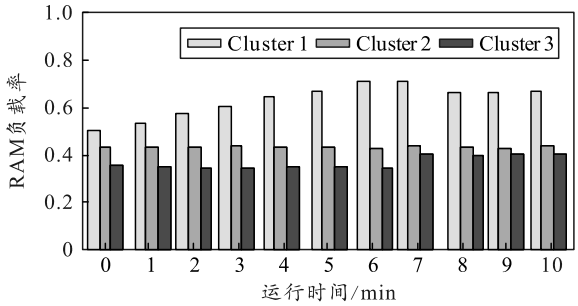
(b) RAM 负载率

图 9　A 组实验集群负载率变化

B 组实验结果如图 10 所示。前 5 min 结果与 A 组的实验一致，但第 6 min 因 Cluster 1 中的随机存取存储器资源负荷值超过了高负荷闭值，而触发了调整。但经过 DICCS 策略的计算，最后选择了将 U_3 调整至 Cluster 2 中，调整后的 Cluster 1 集群计算机将重新成为低负荷的集群计算机。



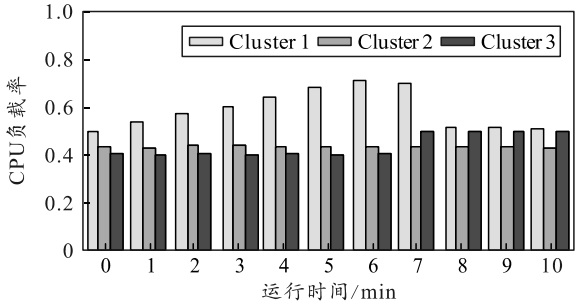
(a) CPU 负载率



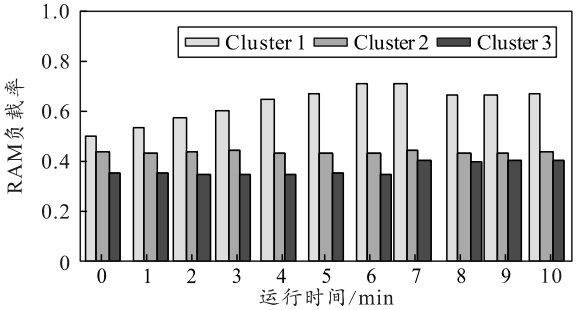
(b) RAM 负载率

图 10　B 组实验集群负载率变化

C 组的实验结果如图 11 所示。前 5 min 与 A 组和 B 组情况相似，但第 6 min 在 Cluster 中 CPU 和 RAM 负载率都处于最大负载闭值，因此触发调整。按照双资源因素触发原则，选取了 CPU 和 RAM 占用率最大的应用领域 U_1，因此选取了 U_1 应用作为待调整的应用。然后，对 Cluster 2 和 Cluster 3 评分结果发现，2 集群评分和 A 组试验结果一致，原因在于对集群计算机评分结果只根据目的集群的实际状况和待调配使用的各种资源配额，而 2 组试验所选择的待调配应用则和目的集群状况相同，所以评分结果也一致。最后 Cluster 将 U_1 部署到 C1uster 3 中。



(a) CPU 负载率



(b) RAM 负载率

图 11　C 组实验集群负载率变化

由以上 3 组试验结果表明，所提的 DICCS 策略可以更有效地依据应用调度触发原因，选取适当的应用调度目的群体，满足对时延不敏感应用的跨群体调节功能。

3　结论

笔者完成多集群边缘云的关键模型设计和构建测试环境，并通过进行多个试验，分别检验了架构的有效性，包括 GMHPA 战略和 DICCS 战略在处理各种使用资源要求时的性能表现。试验结果表明：多集群边缘云架构可以通过域名进行应用请求定向，已经具备了实施的可能性。此外，GMHPA 战略以及 DICCS 战略还可以有效处理架构中时延敏感差异的各种使用资源调配问题，从而一起达到网络资源的动态分配和集群负载平衡。

参考文献：

[1] CHEN J, RAN X. Deep learning with edge computing: A review[J]. Proceedings of the IEEE, 2019, 107(8): 1655-1674.

[2] WANG X, HAN Y, LEUNG V C M, et al. Convergence of edge computing and deep learning: A comprehensive survey[J]. IEEE Communications Surveys & Tutorials, 2020, 22(2): 869-904.

[3] 施巍松, 张星洲, 王一帆, 等. 边缘计算: 现状与展望[J]. 计算机研究与发展, 2019, 56(1): 69.

[4] XIAO Y, JIA Y, LIU C, et al. Edge computing security: State of the art and challenges[J]. Proceedings of the IEEE, 2019, 107(8): 1608-1631.

[5] 杨茂, 陈莉君. 基于 Kubernetes 的容器自动伸缩技术的研究[J]. 计算机与数字工程, 2019, 47(9): 2217-2220.

[6] 杨洋, 曹敏, 杨家海, 等. SSRC: 时延敏感流的数据源端速率控制算法[J]. 通信学报, 2019, 40(7): 14-26.

[7] LIN W, SHI F, WU W, et al. A taxonomy and survey of power models and power modeling for cloud servers[J]. ACM Computing Surveys (CSUR), 2020, 53(5): 1-41.

（下转第 60 页）