黄义宸, 陈庆奎. 一种资源均衡的深度强化学习容器调度优化策略[J]. 智能计算机与应用,2025,15(12):198-205. DOI: 10.20169/j.issn.2095-2163.251130

一种资源均衡的深度强化学习容器调度优化策略

黄义宸, 陈庆奎
(上海理工大学 光电信息与计算机工程学院, 上海 200093)

摘 要: 云计算的不断发展,微服务部署对计算机硬件以及软件的交互过程提出了更高的要求,通过容器调度的方式来处理用户对应用服务提交的请求已经成为了行业范式。针对容器调度过程中存在的负载不均衡、服务硬件要求多元化等问题,本文提出一种基于竞争双深度神经网络强化学习算法的资源控制容器调度优化算法(RCCS-D3QN)。通过实时采集集群各类资源的消耗情况并结合权重因子,利用资源满足率和集群资源均衡率,构造奖惩函数,深度强化学习模型。利用仿真平台 RayCloudSim 构建智能体交互环境,对算法中的神经网络进行训练,基于实际微服务项目来验证结果的有效性。实验结果表明,RCCS-D3QN 算法对比传统调度算法在保证服务质量的同时,在资源利用率、负载均衡度方面有显著提高。

关键词: 微服务; 资源均衡; 容器调度; 深度强化学习; 负载均衡
中图分类号: TP391 **文献标志码**: A **文章编号**: 2095-2163(2025)12-0198-08

A resource-balanced deep reinforcement learning container scheduling optimization strategy

HUANG Yichen, CHEN Qingkui

(School of Optical-electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China)

Abstract: With the growing development of cloud computing, higher requirements have been put forward for the utilization of computer hardware and software, The deployment of microservices through container encapsulation has become an industry paradigm. This article focuses on load imbalance and diversified service hardware requirements in container scheduling, propose a resource control container scheduling optimization strategy RCCS-D3QN based on Dueling Double DQN reinforcement learning algorithm. By collecting the consumption of various resources in the cluster in real time and combining it with the weight factor, using the resource satisfaction rate and the cluster resource balance rate, a reward and punishment function is constructed to implement the reinforcement learning model. By building an interactive environment through the simulation platform RayCloudSim and training the algorithm network, which verify the effectiveness of the results based on actual microservices. The experimental results show that compared with the traditional scheduling algorithm, the RCCS-D3QN algorithm can significantly improve resource utilization and load balancing while ensuring quality of service.

Key words: microservice; resource balance; container scheduling; deep reinforcement learning; load balancing

0 引 言

目前,随着云计算的快速发展,许多企事业单位为更好的管理终端服务、降低运营成本,尽可能最大化效益,程序部署运行环境逐渐由虚拟化技术向容器化技术过渡[1]。在服务的配置与部署时,传统化的虚拟部署方式需要安装完整的操作系统以及相应资源,软件复用率较低[2-3]。在服务即平台的模式下,用户所指定的特殊应用需求与运营商所提供的计算资源不相匹配,分配大量计算机资源来适配,会造成硬件资源的浪费[4]。因此,解决资源分配粒度问题成为满足这类需求的一个解决途径。以 Docker 为代表的容器技术具有轻量、快速部署、易于扩展等特点,提供了更加易于使用、易于维护的优点[5]。然而,在众多容器调度的研究成果中,对于容器特性的研究尚不充分,传统的调度方法未能充分考虑到特定任务的资源

基金项目: 国家自然科学基金(61572325); 上海重点科技攻关项目(19DZ1208903)。
作者简介: 黄义宸(1993—),男,硕士, 主要研究方向:软件工程,云计算。
通信作者: 陈庆奎(1966—),男,博士,教授,博士生导师,主要研究方向:计算机集群,并行计算,人工智能。Email: chenqingkui@usst.edu.cn。
收稿日期: 2024-03-19

要求特性，导致容器云资源利用率低和负载不平衡的问题较多。在容器云环境中，调度过程通常由开发人员开发出对应的默认调度算法完成。容器调度方式在实现方向上可分为以下 3 种方式：

（1）集中式。整个集群状态由唯一的调度实例收集并存储，采用统一的、整体的调度算法来调度所有的作业请求，这样的容器编排引擎包括 Swarm、Fig[6]。

（2）分层式。独立的计算组件负责具体硬件资源的检测和管理，独立的资源调度器负责资源的分配，如 Yarn、Mesos 组件[7-8]。

（3）共享状态资源。集群中存在多个调度器，每一个调度器都拥有对集群全部资源的管理权限，实行自由竞争式的调度策略，实现并发和扩展性的任务规模，如 Brog、Kubernetes 等[9]。

在人工智能领域，强化学习常常被用于决策处理问题，深度强化学习具备高维数据特征运算能力、强自主决策能力，常被用来解决复杂状态空间的决策问题[10]。针对 Swarm 调度方式存在的问题，本文基于深度强化学习提出了一种面向多任务组的动态部署方法，能够适应资源分布不均的集群。为了每个调度请求，在部署时需要考察集群节点的资源消耗，在计算资源层面上选择一个最优的计算节点，并且极大提高集群上可运行的任务数量，同时提高整个集群资源的利用率和负载均衡度。

云计算平台对调度资源的任务指标十分多元化。通常一方面从经济效益出发，考察系统的能源消耗以及单位节点上任务量的大小；另一方面，云平台也必须尽可能平衡用户的体验，减少用户等待时间、提高任务执行速度。传统任务调度常借助启发式算法，如首次拟合和负载算法，但计算过程复杂缺乏实用性[11-12]。一些研究依赖遗传算法进行调度优化，例如基于非支配排序遗传算法的容器调度，利用集群的动态电压频率实现能源消耗与任务执行时间的平衡[13]；Dong 等[14]利用麻雀优化算法改善系统空载情况，从而降低能耗。此外，粒子群算法也用于优化任务调度性能，并探索改进模糊聚类时间模式云计算任务调度方案[15]。Luo 等[16]提出了基于改进的粒子群算法调度策略，通过将迭代次数加入每轮粒子权重的更新以及在最后阶段引入随机性，来避免算法陷入局部最优解。Liu 等[17]在调度资源的维度上，延展了计算资源维度，提出感知 GPU 资源消耗的调度器，使用基于皮尔逊相关系数来衡量各个资源在不同任务中的重要程度；黄亮[18]提出了将实时磁盘 I/O 与网络 I/O 结合的动态权值均衡调度算法；Wang 等[19]在现有 Kubernete 调度框架下，实现自动调度系统。近年来，强化学习在决策问题上也展现出其极大的优势，目前使用最广泛的是基于价值的方法，包括基于模型的 Q-Learning（Off-Policy）与 Sarsa（On-Policy）算法，在集群信息完全可知的情况下来控制容器数量的扩展性[20]；Song 等[21]利用长短期记忆网络（LSTM）预测云计算平台下未来负载量，提出一种提高资源利用率以及降低能源消耗的策略，但由于其依赖的历史数据量较大，无法准确预测新的任务；Dong[22]提出一种基于传统 DQN（qc）算法的容器调度解决方案 RLTS，通过定义任务优先级实现整体任务组执行时间最小化。参考上述思路，针对容器调度无法构建具体数学模型的问题，本文尝试使用在离散空间、非模型环境中效果优秀的 D3QN 算法来解决有限资源下的容器调度问题，并且成功拓展于多节点集群中。与其他算法相比，RCCS-D3QN 算法专注于探索集群与节点的资源利用均衡程度，并很大地提高了资源利用率，同时也保障了任务调度的稳定性。

1　RCCS-D3QN 算法设计

D3QN 算法框架不同于其他深度强化学习算法的优势是：不依赖精确的环境模型、很好适配离散的动作空间。一方面，离散空间下行为策略方法相比于梯度下降方法需要直接输出决策动作，梯度下降方法中概率分布的频繁采样计算将极大延长模型训练的速度。D3QN 容器调度优化算法目标是解决在容器调度请求之间的调用次序差异而产生的资源分配不合理问题，并考虑集群的动态负载状态。

1.1　RCCS-D3QN 算法框架

RCCS-D3QN 总体架构如图 1 所示，RCCS-D3QN 的主要部分为决策智能体，包含主要的双层神经网络训练模块、信息采集、决策智能体、容器调度接口、具体应用。

1）信息采集器（Metrics Monitor）

信息采集器指各个集群硬件节点中的一个系统监控组件或服务，将服务器所处的计算资源环境中的硬件信息采集并回传至信息接收器，硬件信息将用于生成智能体的环境观察值，产生环境数据并放入回放池。常见的开源工具 Telegraf 是较为高效硬件数据采集器的代表，用户可以轻松构建可扩展的监控解决方案，从而实现对基础设施和应用程序的实时监控和分析。Telegraf 是开源的，有特殊采集需求可以对其进行二次开发改造。

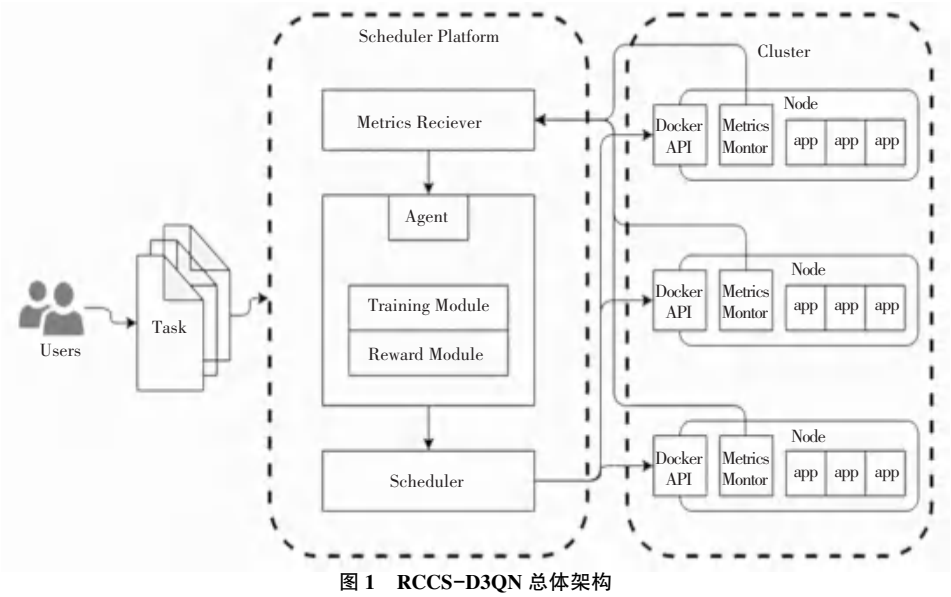


图 1 RCCS-D3QN 总体架构

Fig. 1 Overall architecture of RCCS-D3QN

2）信息接收器(Metrics Reciever)

信息接收器是调度平台用于接收采集器信息的模块,负责接受各集群节点实时发送的集群信息,在模型训练过程中主动发送请求,从而获取到实时的集群资源信息。

3）调度器(Scheduler)

调度器的决策结果由智能体决策根据任务的需求特性以及集群实时的使用状态所决定。通过接口 DockerAPI 对集群环境中的单个容器进行具体的调度行为。

4）决策智能体(Agent)

决策智能体是一个基于深度学习网络 D3Q 的智能体,是为任务调度请求提供调度决策的核心组件,经过离线训练,使动作价值奖励收敛至稳定值,得到稳定可靠的决策结果。在训练过程中,采用经验回放池优化策略,将采样时刻内的状态、动作、奖励、下一时刻的状态 $(s_t、a_t、r_t、s_{t+1})$ 存储在经验回放池中;单步训练过程中,从池中随机抽取一批数据,以往的经验数据能够减少智能体与环境交互的次数,从而提高训练的效率。

5）容器调度接口(Docker API)

容器调度所使用的程序接口能高效、便利的使用容器的各种原生功能,同时增加调度的稳定性。

6）具体应用(App)

用户在前台提交的应用程序或服务,以部署微服务的容器为单位部署在各个物理机中。

1.2 RCCS-DDQN 算法设计

1.2.1 任务请求资源模型

针对部署节点和请求资源,通过层次分析法建立对应的层次结构图,如图 2 所示。

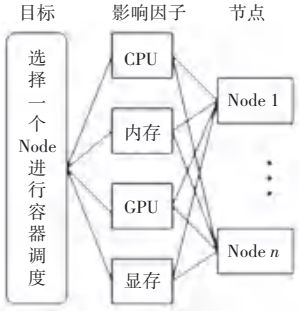


图 2 层次分析过程结构图

Fig. 2 Structural diagram of analytic hierarchy process

假设有 m 个资源指标,基于此构建数据矩阵 $\boldsymbol{X}=(x_{ij})_{m\times m}$:

$$\boldsymbol{X}=\begin{pmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mm} \end{pmatrix} \tag{1}$$

其中,x_{ij} 表示相对于第 j 个资源指标,第 i 个资源指标的重要程度,

资源之间的相对重要程度见表 1。

表 1 相对重要程度

Table 1 Relative importance

标度	含义
1	两个元素相比,具有相同重要性
3	两个元素相比,前者比后者较重要
5	两个元素相比,前者比后者重要
7	两个元素相比,前者比后者非常重要
2,4,6,8	上述相邻判断的中间值

若要充分利用各个节点资源,则资源量相对较宽裕的节点对应资源的重要程度应大于其余资源,

由此可以构建判断矩阵：

$$\boldsymbol{X}=\begin{pmatrix}1 & 3 & 5 & 7\\ 1/3 & 1 & 3 & 5\\ 1/5 & 1/3 & 1 & 1\\ 1/7 & 1/5 & 1 & 1\end{pmatrix} \tag{2}$$

本文使用一致性校验(Consistency Index，CI)和一致性比例(Consistency Ratio，CR)来衡量判断矩阵的一致性：

$$\mathrm{CI}=\frac{\lambda_{\max}-m}{m-1} \tag{3}$$

$$\mathrm{CR}=\frac{\mathrm{CI}}{\mathrm{RI}} \tag{4}$$

其中，RI 为一致性指标，具体的取值见表 2，$\lambda_{\max}$ 为判断矩阵的最大特征根。若 CR<0.1，则通过一致性检验，否则判断矩阵需要重新验证和计算。

表 2　一致性指标

Table 2　Consistency indicator

m	1	2	3	4	5	6	7	8
RI	0.00	0.00	0.52	0.89	1.11	1.25	1.35	1.40

计算符合 $X\cdot W=\lambda_{\max}\cdot W$ 的特征向量，归一化 $W_{\lambda_{\max}}$，即可得到权重 W_{Xj}，根据上述步骤最终可以得到层次分析法的权重信息，具体见表 3。

表 3　层次分析法权重表

Table 3　Analytic hierarchy process weight table

指标	CPU	内存	GPU	显存
权重	0.565	0.269	0.091	0.075

1.2.2　深度强化学习算法

通过定义容器调度过程模型中的状态 s、动作 a 以及奖励函数 $R(s,a)$，本文设计出一套区别于传统任务调度方法的 RCCS-D3QN 深度强化学习算法。

1）状态 s：由于集群环境存在连续性的时变特性，因此采用有模型的强化学习算法可能无法得到预期的改善效果，算法将运行中的集群硬件信息和待调度任务的资源使用量作为变量进行采集。集群资源的观测状态为集群中某类资源的未使用量 R_{unused}；同时使用待调度任务的对应资源需求量 R_{need}，通过差分比例的形式定义状态参数 g_k^i，确定集群中某一节点在该资源上是否满足任务要求，并且能够反映集群资源的满足率，公式如下：

$$g_k^i=\frac{R_{\text{unused}}-R_{\text{need}}}{R_{\text{unused}}} \tag{5}$$

当资源容量能够覆盖任务需求时 $0\leqslant g_k^i\leqslant 1$，资源无法满足任务需要时，该值为负数，表示任务无法在该节点上调度。

由层次分析法可计算出各个节点中资源指标的权重 $\lambda=\{\lambda_i\mid i\in\{1,2,\cdots,m\}\}$，$\sum_{i=1}^{m}\lambda_i=1$。资源类型数为 m 时，则第 k 个节点的加权平均满足率（$0\leqslant k\leqslant n$）：

$$G_k(s)=\sum_{i=1}^{m}\lambda_i g_k^i \tag{6}$$

集群的整体资源满足率：

$$G(s)=\frac{\sum_{k=1}^{n}G_k(s)}{n} \tag{7}$$

不同硬件维度资源的满足率之间，通过两两之间差的绝对值来衡量，累加求出均值。集群计算资源的整体满足率可以表示为：

$$B(s)=\frac{\sum_{k=1}^{n}\sum_{i=1}^{m}\sum_{j=1}^{m}abs(G_k^i-G_k^j)}{m^2n} \tag{8}$$

集群资源使用量越平衡，$B(s)$ 越小。

环境状态定义为 s，s 是集群各个资源在时刻 t 的状态 s_c^t。在含有 m 个资源种类与 n 个节点的集群环境中，某一时刻 t 下的环境状态可以被定义为：

$$\boldsymbol{S}_C^t=\begin{pmatrix}G_{11}^t & \cdots & G_{1m}^t\\ \vdots & \ddots & \vdots\\ G_{1n}^t & \cdots & G_{nm}^t\end{pmatrix} \tag{9}$$

2）动作 a：容器调度的动作设计思路与离散型深度强化学习的输出相似。集群硬件资源使用量每时每刻都在起伏，但在需调度的时刻可近似认为是离散的状态，单一的微服务组件在调度过程中孤立的进行调度作业。因此，调度的节点数量可以用离散数字来表示。例如，当网络在状态 s 输出第 i 个动作的价值相对最高时，表示任务最适合被调度到第 i 个节点之上。对于 n 个节点，动作可定义为：

$$a=\{a^*\mid a^*\in\{0,1,\cdots,n-1\}\} \tag{10}$$

3）目标函数 $Q(s,a)$：目标函数是指在强化学习中用于衡量智能体学习效果的函数，其目标函数可以表示为每个时间步 t 的累积回报的期望值。累积回报是指从当前时间步开始，智能体在未来一系列时间步中所获得的奖励的总和。在容器调度问题中，转化为每一次调度过程所产生的奖励，在当前步骤中可以计算出任务完成后的数学期望，其表达式为：

$$Q(s,a)=E(R+\gamma\times\max(Q(s',a'))) \tag{11}$$

其中，$Q(s,a)$ 是状态 s 下动作 a 的新 Q 值；R 是执行 a 动作后得到的奖励；$\max(Q(s',a'))$ 是下一状态 s' 下所有的动作 a' 中最大的 Q 值；参数 $0 \leqslant \gamma \leqslant 1$。

4）奖励函数 $R(s,a)$：奖励 R 以集群整体资源满足率为主，但各个节点的资源内部的资源平衡也必须考虑，从而可以将更多的容器部署在集群中。当有新任务分配给集群时，必然将提高整体资源的利用率，但节点内部的资源碎片会增加。考虑到集群中计算节点的资源差异，即使在同一资源维度的利用率相同时，节点资源的绝对值也会对任务调度产生很大的影响。例如，不应将任务集中到资源绝对值相对较小但利用率较低的节点上，这将增加该节点的负载率。因此，设置奖励函数 R 时需要将上述 3 个因素一并考虑在内。

奖励函数可设为集群在调度发生前时刻的资源利用率减去各个节点自身的利用率乘以一个衰减系数：

$$R(s,a) = \alpha \cdot G(s) - \beta \cdot B(s) \tag{12}$$

奖励函数的目标倾向可以通过参数 α 与参数 β 来进行调整，当注重于集群整体负载均衡性时可以保持 β 的值不变，增加 α 的值能够提高集群负载程度带来的奖励比重，奖励值越大，说明该动作下，综合均衡性越好。

训练传统深度 Q 网络时，输入为一维的状态向量，通过神经网络输出一维的动作向量，向量代表了某一状态下动作的价值。不同的是，竞争深度 Q 网络改变了网络的架构，其输出一个标量 $V_{(s)}$ 与动作相关的向量 $\boldsymbol{A}(s,a)$，将向量中的各项数值相加后得到状态动作价值 $Q(s,a)$。基于经验回放的决斗双深度 Q 学习算法的伪代码如算法 1 所示。

算法 1 基于经验回放的决斗双深度 Q 学习算法

初始化重放内存 D 的容量为 N；随机初始化动作-价值网络 Q 中的参数 θ；初始化对象动作-价值网络 $\hat{Q}$ 使 $\hat{\theta} = \theta$

循环 1，到 M 步

初始化状态序列 s_c^t

循环 $t = 1$ 到 T

依据概率参数 ε 选择其中一个动作 a_t

概率未命中时 $a_t = \text{argmax}(Q(s_c^t,a;\theta))$

执行动作 a_t 从而计算奖励 r_t 以及下一阶段的状态 s_c^{t+1}

将当前状态、动作、奖励、下一时刻状态 $(s_c^t, a_t, r_t, s_c^{t+1})$ 存储在重放内存 D 中，设置当前状态 $s_c^t = s_c^{t+1}$

从重放内存 D 中随机抽取一组 $(s_c^j, a_j, r_j, s_c^{j+1})$

设置 $y_j = \begin{cases} r_j \\ r_j + \gamma Q(s_c^{j+1}, \text{argmax}(\hat{Q}(s_c^{j+1},a';\theta);\hat{\theta})) \end{cases}$

如果任务完成则设置 $y_j = r_j$

对误差 $(y_j - Q(S_c^j,a_j;\theta))^2$ 在动作-价值网络中进行梯度下降

设置对象动作-价值网络参数，使 $\hat{Q} = Q$

循环结束

循环结束

2 实验结果和分析

2.1 实验过程

首先，基于某市公交车预测系统平台微服务组进行离线训练，微服务：某市公交车信息预测平台组件信息见表 4。将 RCCS-D3QN 算法与 Docker 集成组件 Swarm 中的微服务调度策略扩散式调度（Spread）、装箱式调度（Binpack）和随机调度算法（Random）进行比较并进行均衡性评估，来验证本文所提出的 RCCS-D3QN 算法的可行性。

其次，将微服务组设置为小、中和大 3 种不同规模的副本数量，通过不同任务规模下的调度性能的在线调度性能评估算法性能，评估各算法在集群负载均衡度上的性能差异。微服务容器规模配置见表 5。

表 4 微服务：某市公交车信息预测平台组件信息

Table 4 Details of the platform of bus information prediction

服务组件名称	服务数量	功能描述
图片预处理模块	1	上游图片源预处理
前端模块	1	用户服务发起管理 web 接口
数据流监控	1	监控单一服务各项性能指数
图片中间池 redis	1	图片预处理、上下游处理能力一致性对接
图片分析模块	1	深度学习模型

表 5 微服务容器数量配置

Table 5 Configuration table of container number of size

规模类型	副本数量	总容量数量
小	3	15
中	9	45
大	27	135

2.2 实验设计

离线训练实验在云计算仿真平台 RayCloudSim 上进行。RayCloudSim 是一个用 Python 编写的轻量

级模拟平台，可用于云、雾、边缘计算基础设施和服务的分析建模和仿真。RayCloudSim 是一个基于进程和离散事件的模拟仿真平台，可以尽可能的缩短以现实世界时间衡量的模拟耗时，具有轻量化、易于对接主流强化学习框架等优点。在初期训练阶段，设置了 6 个服务器节点来观察资源利用率，并将 CPU 利用率、内存利用率、GPU 利用率、显存利用率 4 项指标作为训练目标，并计算集群负载均衡情况。同时采用传统调度工具中的 Docker Swarm 作为对照实验的对象，实验结果通过以下指标衡量：

（1）集群整体各项资源（CPU、内存、GPU、显存）的平均利用率；

（2）集群整体负载均衡率。

学习率 r、折扣因子 γ 在实验过程中逐步调整，奖励函数内权重 α 与 β 分别衡量集群整体利用率以及单节点利用率之间的取舍关系，同时将 4 个参数组合进行实验。实验结果表明，学习率 α、折扣因子 γ、α_r、β_r 分别为 1.0、0.8、10、1 时，理论上每成功部署一个任务可以获得最大奖励为 10。

2.3　实验分析

2.3.1　资源利用率评估

RCCS-D3QN 算法进行训练的初期，获得的动作奖励回报很低，这是由于资源需求倾斜导致的调度任务失败。在微服务架构中，部分组件失效往往导致整个任务失效，因此实验中直接中止了之后容器的调度任务。在经历多轮迭代后收敛，在 2 000 轮迭代后 RCCS-D3QN 算法的奖励函数收敛于 120，如图 3 所示。

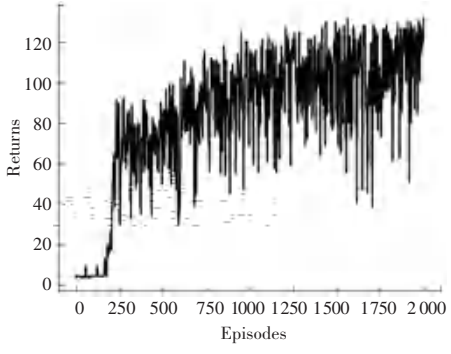


图 3　平均奖励

Fig. 3　Average reward

小型规模下的 RCCS-D3QN 算法、Swarm 中调度策略的实验对比结果如图 4~图 7 所示，分别对不同调度算法在调度过程中 4 个资源维度：CPU、内存（M）、GPU、显存（GM）利用率进行计算与观测，资源利用率曲线波动反映了一个容器调度结束后系统的资源变化。

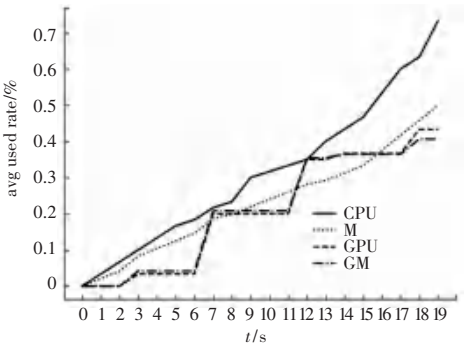


图 4　RCCS-D3QN 算法下的利用率

Fig. 4　Utilization of RCCS-D3QN

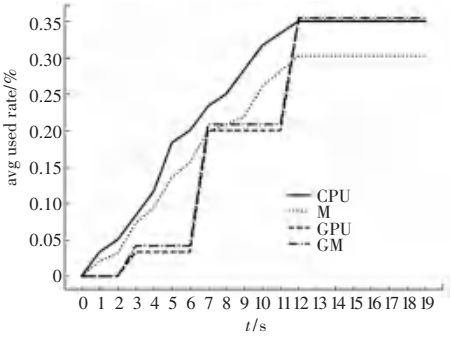


图 5　扩散式调度下的利用率

Fig. 5　Utilization under spread scheduling

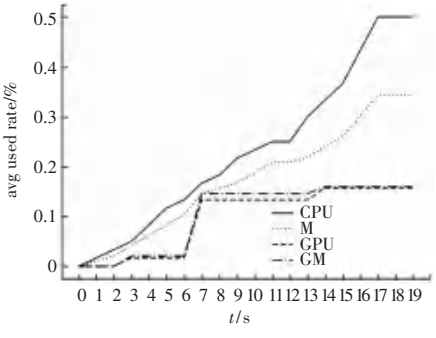


图 6　装箱式调度下的利用率

Fig. 6　Utilization under Bin-Packing

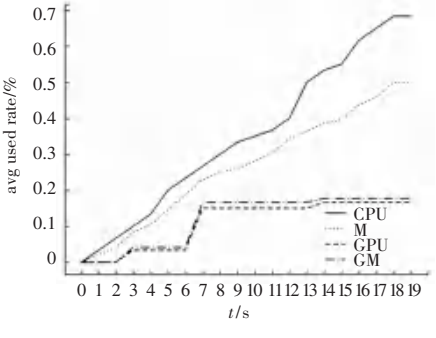


图 7　随机调度下的利用率

Fig. 7　Utilization under random scheduling

在 RCCS-D3QN 算法的调度过程中，对服务器 CPU 和内存的利用率提升趋势较为平缓且接近任务组需求上限，即策略可最大限度部署任务组中任

务;扩散式调度可以最大平均各个节点的负载,但在含有需要 GPU 资源时,由于各节点的差异,有极大可能造成部分任务缺少资源而运行失败;装箱式调度极限使用各个单节点,对单节点需要很高的性能要求;随机调度算法在缺乏特定任务部署顺序下会极大的浪费节点的资源,并且导致任务部署失败。可见 RCCS-D3QN 算法在保证任务组调度过程成功实施的同时,相比较于 Swarm 中的传统调度算法对整体资源的利用率有明显提高。

通过实验对比可以观测到,传统的调度算法中会出现无法调度、集群节点负荷不足等问题,而在本次设计的容器组环境下,RCCS-D3QN 算法均能满足容器全部挂载,实现集群最大的资源利用率;由于微服务中各服务之间存在功能上的依赖关系,若部分服务无法部署将会导致整个任务失败;在物理节点上,已经部署的微服务组件将占用物理机资源,这部分的系统占用将占用物理机资源,这种情况下需要系统管理员人工将该服务组中的其余服务手动关停,这将增加实际工作中的服务维护成本。

2.3.2 负载均衡性性能评估

集群负载均衡度 $\tau_{\text{cluster}}(t)$ 是所有节点在计算资源消耗上的标准方差,表示多个节点之间在 t 时刻时的工作负载均衡程度。本文采用前文所述的 4 个资源维度进行计算,公式如下:

$$N_k^l(t)=\frac{R_{\text{used}}}{R_{\text{total}}} \tag{13}$$

$$\bar{N}^i(t)=\frac{1}{n}\sum_{k=1}^{n}N_k^i(t) \tag{14}$$

$$\tau_{\text{cluster}}(t)=\frac{1}{4}\sum_{i=1}^{4}\left(\sqrt{\frac{1}{n}\sum_{k=1}^{n}(N_k^l(t)-\overline{N^l}(t))^2}\right) \tag{15}$$

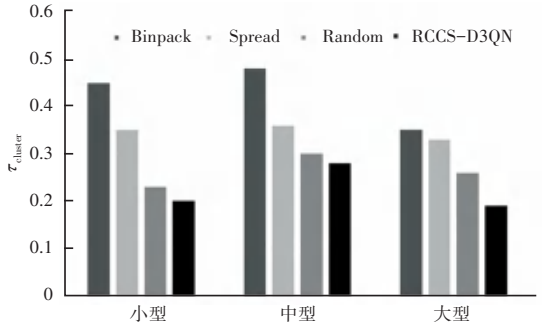
节点负载均衡度 $\tau_{\text{node}}(t)$ 是单个节点的平均资源利用率的标准方差,表示节点内部资源间的距离差距,进而表示利用率的均衡程度如下:

$$\overline{N_k}(t)=\frac{1}{4}\sum_{i=1}^{4}N_k^i(t) \tag{16}$$

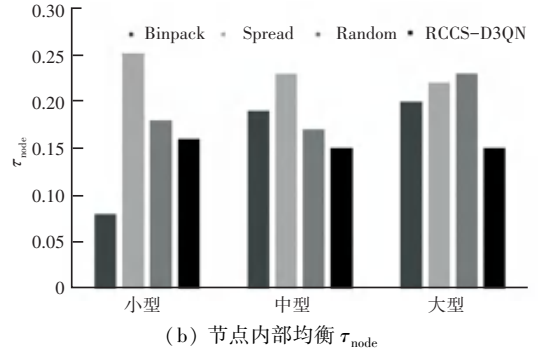
$$\tau_{\text{node}}(t)=\frac{1}{n}\sum_{k=1}^{n}\left(\sqrt{\frac{1}{4}\sum_{i=1}^{4}(N_k^i(t)-\overline{N_k}(t))^2}\right) \tag{17}$$

集群负载均衡率如图 8(a)所示,通过对不同任务规模不同调度算法的情况下的集群负载均衡度 τ_{cluster} 的比较,可以明显看到装箱式调度策略的 σ_{cluster} 值最大,这说明集群的负载均衡性最差,这是由于策略优先把节点调度到负载最高的节点中来提升节点的资源利用率,导致了其他节点闲置的集群负载不均衡和部分节点负载过高,造成了集群计算资源碎片。扩散式调度的贪婪性策略试图将容器平均分配给每个节点,导致了小型、中型和大型资源节点间的负载不均衡,RCCS-D3QN 能够在保持较低集群负载均衡度 τ_{cluster} 的前提下满足不同容器规模的调度需求,而随着容器规模的增加,扩散式调度和装箱式调度策略下的集群状态会计算出较高的负载均衡度指标。

节点内部负载均衡如图 8(b)所示,描述了 4 种调度策略在小、中、大这 3 种容器规模下节点负载均衡度 τ_{node} 的对比,可以观察到装箱式调度策略下的节点资源差异度较大,这是由于以上两种调度策略没有考虑容器本身的多维度的负载需求,而扩散式调度、随机调度策略下的任务调度存在随机性,无法固定单个容器的调度节点,因此其单节点的均衡度无法得出一个稳定的趋势判断。任务对与资源需求的不同要求,可能会导致部分部署决策的不同,从而导致任务完全部署后集群资源负载均衡度之间的巨大差异,传统的调度方法无法根据单个容器中的各个硬件的利用率差异来进行调度优化,从而会导致任务部署失败,节点资源被单一任务完全抢占。本文将节点内部资源间的满足率差异度作为调度优化策略的考虑因素,因此能够将整个微服务的资源需求特征考虑在内,从而实行资源均衡的调度算法,使得各个规模下都能保持良好的均衡度。



(a) 集群负载均衡 τ_{cluster}



(b) 节点内部均衡 τ_{node}

图 8 集群负载均衡与节点内部负载均衡比较

Fig. 8 Comparison of cluster load balancing and load balancing within nodes

3　结束语

本文总结了对计算资源要求倾斜的微服务造成的集群资源占用不均衡问题，提出了一种深度强化学习的容器调度算法 RCCS-D3QN。定义服务器 CPU、内存、GPU、显存的满足率，使用层次分析法挖掘资源权重，同时依靠竞争双层深度强化学习的算法模型，借助构建容器环境下的任务调度模型、奖励模块、双层网络和经验回放池等优化技术构建出具体算法，同时提高了集群资源利用率与集群负载均衡度。在 RayCloudSim 平台上训练神经网络参数，在不同的微服务任务规模下进行算法的验证。实验结果表明，RCCS-D3QN 算法相比于 Swarm 默认调度算法有着较高的计算资源利用率，集群以及单节点中的负载均衡度也表现良好。后期工作将在多智能体调度策略、在资源受限额的边缘计算环境下的调度策略、超大规模任务组在集群中的并发部署、高纬资源指标降维等重点方向上展开。

参考文献

[1] WU Z X. Advances on virtualization technology of cloud computing [J]. Journal of Computer Applications, 2017, 37(4): 915-923.

[2] QIU J X. Design of Linux online experiment environment based on docker container technology[J]. Information Technology, 2022(2): 48-52.

[3] BAN Y Z , LI H, CHEN M, et al. Design and implementation of intelligent monitoring system for Web application[J]. Intelligent Computer and Applications, 2022, 12(9): 172-178.

[4] LUO D F, LI F, HAO W Y, et al. Large-scale log collection and analysis system based on docker [J]. Computer Systems & Applications, 2017, 26(10): 82-88.

[5] RAD B B, BHATTI H J, AHMADI M. An introduction to docker and analysis of its performance [J]. International Journal of Computer Science and Network Security (IJCSNS), 2017, 17(3): 228.

[6] JANSEN C, WITT M, KREFTING D. Employing docker swarm on openstack for biomedical analysis[C]//Proceedings of the 16th International Conference on Computational Science and Its Applications. Cham: Springer, 2016:303-318. DOI: 10. 1007/978-3-319-42108-7_23.

[7] HINDMAN B, KONWINSKI A, ZAHARIA M, et al. Mesos: a platform for fine-grained resource sharing in the data center[C]// Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation. Berkeley, USA: USENIX Association, 2011: 429-483.

[8] GHODSI A, ZAHARIA M, HINDMAN B, et al. Dominant resource fairness: fair allocation of multiple resource types[C]// Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation. Berkeley, USA: USENIX Association Press, 2011: 323-336.

[9] VERMA A, PEDROSA L, KORUPOLU M, et al. Large-scale cluster management at Google with Borg[C]//Proceedings of the 10th European Conference on Computer Systems. New York: ACM Press, 2015: 1-17.

[10] LIU Z Y, MU C X S. An overview on algorithms and applications of deep reinforcement learning[J]. Chinese Journal of Intelligent Science and Technology, 2020, 2(4): 314-326.

[11] BHOI U, RAMANUJ P N. Enhanced max-min task scheduling algorithm in cloud computing [J]. International Journal of Application or Innovation in Engineering and Management, 2013, 2(4): 259-264.

[12] LU J W, LI J, ZHANG Y M, et al. Research on the load balancing scheduling policy of tasks in cloud computing environments based on improved Tabu Search [J]. Journal of Chinese Computer Systems, 2018, 39(10): 2254-2259.

[13] SOFIA A S , GANESHKUMAR P . Multi-objective task scheduling to minimize energy consumption and make span of cloud computing using NSGA-Ⅱ[J]. Journal of Network and Systems Management, 2018, 26(2): 463-485.

[14] DONG J K, ZHAO Y, ZHANG M H. Scheduling mechanism of docker based on energy consumption perception [J]. China Science Paper, 2022, 17(11): 1260-1266.

[15] LIU G Q, SHI X C. An improved fuzzy clustering fastest time pattern of cloud computing task scheduling scheme[J]. Control Engineering of China, 2018, 25(11): 2092-2096.

[16] LUO F, YUAN Y, DING W, et al. An improved particle swarm optimization algorithm based on adaptive weight for task scheduling in cloud computing [C]//Proceedings of the 2nd International Conference on Computer Science and Application Engineering. 2018: 1-5.

[17] LIU X, HU R M, WANG H B, Ai scheduling engine platform based on kubernetes[J]. Computer Systems and Applications, 2023, 32(8): 86-94.

[18] 黄亮. 容器云中的资源调度策略研究与实现[D]. 成都：电子科技大学, 2022.

[19] WANG Y R. Implementation of system management for dispatching automation system based on container platform[J]. Industrial Control Computer, 2021, 34(6): 127-129.

[20] ZHANG S, WU T, PAN M, et al. A-SARSA: A predictive container auto-scaling algorithm based on reinforcement learning [C]// Proceedings of 2020 IEEE International Conference on Web Services (ICWS). Piscataway, NJ: IEEE, 2020: 489-497.

[21] SONG B, YU Y, ZHOU Y, et al. Host load prediction with long short-term memory in cloud computing [J]. The Journal of Supercomputing, 2018, 74(12): 6554-6568.

[22] DONG T, XUE F, XIAO C, et al. Task scheduling based on deep reinforcement learning in a cloud manufacturing environment [J]. Concurrency and Computation: Practice and Experience, 2020, 32(11): 5654-5670. DOI: 10. 1002/cpe. 5654.