算力网络场景下SLA约束的能耗优化微服务调度策略

Research on Energy-optimized Microservice Scheduling Strategies with SLA Constraints in Computing Power Network Scenario

刘博文[1]，梁晓晨[1]，张桂玉[1]，韩振东[2]（1. 中讯邮电咨询设计院有限公司，北京 100048；2. 中国联合网络通信集团有限公司，北京 100033）

Liu Bowen[1], Liang Xiaochen[1], Zhang Guiyu[1], Han Zhendong[2]（1. China Information Technology Designing & Consulting Institute Co., Ltd., Beijing 100048, China; 2. China United Network Communications Group Co., Ltd., Beijing 100033, China）

摘　要：

针对算力网络场景下的微服务调度问题，考虑了微服务对网络服务质量的需求和节点能耗优化问题，研究了算力网络场景下服务质量约束的能耗优化微服务调度问题。首先建立了由算力网络调度模块、算力网络调度执行模块和算力网络信息表构成的算力网络调度框架。基于该框架，同时考虑了节点资源限制和网络服务质量约束，建立了能耗优化的微服务调度模型，并采用深度确定性策略梯度算法解决微服务调度问题。最后通过仿真验证了该调度算法的有效性。

关键词：微服务；能耗优化；服务层协议；算力网络

doi：10.12045/j.issn.1007-3043.2023.02.006

文章编号：1007-3043（2023）02-0031-06

中图分类号：TN919

文献标识码：A

开放科学（资源服务）标识码（OSID）：

Abstract:

Aiming at the micro-service scheduling problem in the computing power network scenario, considering the demand of micro-services for network quality of service and the node energy consumption optimization problem, the energy-optimized micro-service scheduling problem with service quality constraint in the computing power network scenario is studied. Firstly, it establishes an computing power network scheduling framework consisting of an computing first networking scheduling module, an computing first networking scheduling execution module, and an computing first networking information table. Secondly, based on this framework, considering node resource constraints and network service quality constraints, it establishes an energy-optimized micro-service scheduling model, and solves the micro-service scheduling problem by using a deep deterministic policy gradient algorithm. At last, the effectiveness of the scheduling algorithm is demonstrated by simulation experiments.

Keywords:

Micro-services; Energy optimization; Service layer protocols; Computing power network

引用格式：刘博文，梁晓晨，张桂玉，等. 算力网络场景下SLA约束的能耗优化微服务调度策略[J]. 邮电设计技术，2023(2)：31-36.

1 概述

算力是互联网时代的基础能源，正在驱动不同行业的信息化转型。目前，广受关注的云计算、边缘计算、人工智能计算等都属于算力的分支[1]。随着产业信息化、数字化的不断深入，互联网、大数据、云计算、人工智能、区块链等技术不断发展与创新，海量的应用与技术创新对算力提出了新的需求[2]，包括更大的算力供给、更高的计算性能标准和更多样的算力服务需求。因此，将算力与网络深度融合，实现云、边、端多级计算的高效协同部署是必然趋势。2021年5月24日，国家发展改革委、中央网信办、工业和信息化部、国家能源局联合印发了《全国一体化大数据中心协同创新体系算力枢纽实施方案》，文件提出要在全国各地布局建设全国一体化算力网络国家枢纽节点，加快实施“东数西算”工程，提升跨区域算力调度水平[3]。

目前，业内将算力网络的发展主要分为算网分

收稿日期：2022-12-28

治、算网协同、算网一体化3个阶段[4]。为了最终实现算网一体化，为用户提供泛在协同的连接与计算服务，算网感知功能、算力路由、算力编排调度等关键技术都是业界研究的重点[5]。微服务架构也是算力网络发展的潜在技术。微服务架构是基于面向服务的架构（Service Oriented Architecture，SOA）思想[6]，将大型项目根据需求拆分为多个小应用服务，每一个小的服务即一个微服务。多个项目的不同微服务可以灵活部署在不同的节点上，有利于提高资源利用率。从2016年开始，为了满足业务部署的个性化、简便化需求，微服务这一架构风格开始普及[7]。目前，各大网络运营商都有自己特色的微服务框架，微服务的普及也带来了大量的微服务处理需求。

在算力网络场景下，采用微服务架构，可以将多样化异构的算力网络服务请求拆分为多个小应用服务，不同的算力网络服务请求可以共享同一个微服务，从而满足业务个性化、简便化的需求。算力网络中，微服务的部署调用需要考虑微服务部署、微服务请求调度和动态扩缩容3种业务请求。对不同业务请求调度的合理性直接决定了算力网络的资源利用率和算力网络服务性能。文献[8]针对边缘计算场景下微服务调度问题，提出了一种新颖的分布式延迟优化的微服务调度机制，实现最小的微服务时延，同时保证传输速率以最大限度降低传输时延。文献[9]提出了一种边缘计算场景下动态微服务调度方案，优化了网络吞吐量、总时延和微服务利用率，为用户提供了公平的网络服务质量，提高了终端用户的满意度。文献[10]提出了一种基于强化学习的在线微服务协调算法来学习最优策略，该算法可以在降低服务延时的同时降低微服务迁移成本。文献[11]提出了一种基于奖励共享机制的深度Q学习算法来解决边缘计算中的多目标微服务部署问题，有效降低了服务平均响应时间，优化了负载均衡，提高了服务执行的可扩展性。现有工作主要集中于边缘计算场景下的微服务部署，考虑的因素包括时延、服务质量、成本等，模型基于边缘计算框架建立，未考虑算力网络场景下，云边端协同调度和网络高动态性。同时，上述模型没有考虑云计算、数据中心、边缘计算发展面临的高能耗问题，降低能耗是服务提供商关注的重点问题之一。

因此，本文从降低能耗和提升网络服务质量的角度出发，研究了微服务调度问题。降低能耗是降低服务成本的关键因素之一。从用户角度出发，服务质量是用户对算网服务的基本要求。服务层协议（Service Level Agreements，SLA）是指提供服务的企业与客户之间就服务的品质、水准性能等方面所达成的双方共同认可的协议或契约[12]，是解决Web服务中这些问题的基石。SLA有助于在网络服务之间划分责任和风险，从而使网络服务更加有序。因此，SLA约束下能耗优化调度策略是一种在保障算网基本服务质量的同时，降低服务提供商服务成本的调度策略。

本文针对SLA约束下能耗优化的微服务调度问题，首先针对算力网络调度服务需求特点，建立了算力网络调度框架。在此框架基础上，针对算力网络资源有限性、网络高动态性，建立了网络模型、微服务需求模型、能耗模型、SLA约束模型和资源限制模型。最后，本文以能耗最小作为优化目标，以保障网络的服务质量为约束，建立了马尔可夫决策模型，并利用深度确定性策略梯度（Deep Deterministic Policy Gradient，DDPG）算法自动寻找最优的调度策略，仿真结果证明了该算法的有效性。

2 系统模型

2.1 算力网络调度框架

由于网络算力节点的计算资源有限，且节点间通信能力受到带宽、信道状态等通信资源与环境因素影响，因此对算力服务进行最优部署与灵活调度，提升服务性能，是算力网络框架中的重中之重[13]。如图1所示，本文设计的算力网络调度框架主要由算力网络调度模块、算力网络调度执行模块和算网信息表3个模块组成。

算力网络调度模块负责根据算力网络信息表中存储的多维资源信息和微服务需求，根据优化目标生成微服务调度策略。算力网络调度策略的优化目标有负载、成本、服务质量、能效等等。本文针对SLA约束的能耗优化这个目标，设计了系统模型。

算力网络调度执行模块负责将算力网络调度模

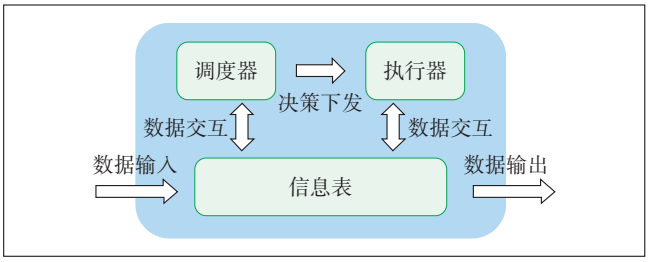

图1 算力网络调度框架

板生成的微服务调度策略，将微服务调度部署到相应的算力网络服务节点，并将对应的信息变化更新到算力网络信息表中，帮助更新算力网络资源信息。

算力网络信息表负责存储算力网络多维资源信息，包括计算资源信息、存储资源信息和网络资源信息等。同时，算力网络信息表还需存储算力网络调度模块生成的调度策略信息。

基于上述算力网络调度框架，算力网络调度的工作流程为：算力网络收到微服务调度请求，主要包括微服务部署、微服务调度、动态扩缩容3类；根据微服务调度请求和算力网络资源信息，算力网络调度模块生成微服务调度策略；算力网络调度执行模块根据调度策略，将该微服务部署到相应的集群上，并绑定。

2.2 网络模型

网络由多种集群和综合纳管调度平台组成。集群内由主节点和多个工作节点组成，主节点根据微服务对各类资源的需求和集群内节点的算力资源、负载、网络资源等信息调度部署微服务。

综合纳管调度平台连接用户和集群，是用户部署微服务的接口。用户上传微服务，综合纳管调度平台在接收到用户的服务请求后，根据微服务的需求和SLA约束下能耗优化的微服务调度策略，将微服务调度部署到合适的集群。在微服务处理完成后，综合纳管调度平台将服务处理结果反馈给用户。

2.3 微服务需求模型

对微服务的建模主要包括微服务的种类和微服务对计算资源、存储资源和网络资源等各种资源的需求。微服务用s_i表示，$s_i \in S$，S表示微服务种类的集合，微服务种类的个数为I，微服务s_i副本数为n_i。微服务建模如下：

$$s_i = \left\{ n_i^{\text{copy}}, \text{vCPU}_i, \text{cCache}_i, \text{Storage}_i, \text{GPU}_i, \text{FPGA}_i, \text{Type}_i \right\}$$

$$\text{vCPU}_i = \{ \text{Num} \}$$

$$\text{Storage}_i = \{ \text{Size} \}$$

$$\text{GPU}_i = \{ \text{Type}, \text{FLOPS}, \text{Size}, \text{MBW} \}$$

$$\text{FPGA} = \{ \text{BoardType} \}$$

Type_i字段代表微服务s_i的需求类别，微服务的需求类别主要分为CPU需求型、GPU需求型和FPGA需求型3种类型。微服务的模型描述了该微服务对计算资源、网络资源等资源的需求，需求参数的详细解释见表1。

2.4 能耗模型

表1 微服务需求参数注解

参数名	参数注解
vCPU	CPU核数，指单位CPU芯片上集成的内核单元数
cCache	CPU缓存，位于CPU与内存之间的临时数据交换器
Storage	存储，微服务所需的存储空间大小
GPU	显卡、图形处理器，GPU硬件计算能力的主要衡量指标是其吞吐量，即每秒浮点运算量
FPGA	可编程逻辑门阵列，FPGA的代码绝大部分受限于芯片型号

已知微服务共有I种，微服务s_i所需的资源类型与数量，可提供服务的算力网络节点共有M个。微服务运行在该节点上的能耗与该节点的CPU峰值功率、CPU空闲功率、CPU频率、CPU利用率等有关。待处理的微服务请求集合$S = \{ s_1, s_2, \cdots, s_I \}$，可部署的算力网络节点集合$M = \{ m_1, m_2, \cdots, m_M \}$，引入变量$x_{ij}$，用于说明微服务$s_i$部署在算力网络节点$m_j$上，具体如下：

$$x_{ij} = \begin{cases} 1, \text{微服务} s_i \text{部署在节点} s_1 \text{上} \\ 0, \text{其他} \end{cases}$$

节点m_j的利用率u_j与该节点的总核数n_{vCPU_j}与该节点已被其他微服务占用的核数n_{used_j}有关，计算公式如下：

$$u_j = \frac{n_{\text{used}_j}}{n_{\text{vCPU}_j}}$$

此时节点m_j的利用率为u_j，该节点m_j的全速率工作的平均功率为$P^j_{\max}$，空闲状态的平均功率为P^j_{idle}，则该使用状态下该节点m_j的平均功率为P^j_u，具体计算公式如下：

$$P^j_u = \left(P^j_{\max} - P^j_{\text{idle}} \right) \times u_j + P^j_{\text{idle}}$$

用t_{u_j}表示节点资源利用率为u_j的时间长度，则节点的能耗E_{ij}为：

$$E_{ij} = P^j_u \times t_{u_j}$$

则将I个微服务部署到J个节点上的总能耗为：

$$E = \sum_{i=1}^{I} \sum_{j=1}^{M} x_{ij} \times \left\{ \left[\left(P^j_{\max} - P^j_{\text{idle}} \right) \times u_j + P^j_{\text{idle}} \right] \times t_{u_j} \right\}$$

2.5 SLA约束模型

SLA是指提供服务的企业与客户之间就服务的品质、水准性能等方面所达成的双方共同认可的协议或契约，因此保证服务SLA即保障服务的质量。为了衡量算网服务质量，本文选取了带宽、时延、SD-WAN切换时延、抖动和丢包率5种网络服务质量指标。微服务s_i对网络服务质量的要求，即SLA_i建模如下：

$$\text{SLA}_i = \{ \text{BW}_i, \text{tDelay}_i, \text{sdDelay}_i, \text{Jitter}_i, \text{Drop}_i \}$$

各网络服务质量指标的详细解释见表2。

表2 SLA指标注解

网络质量指标	单位	注解
带宽BW	bit/s、kbit/s、Mbit/s	链路传输数据的能力，即最大速率（由网络决定）
时延tDelay	s、ms	用户获得服务的总时延，包括传输时延、排队时延、处理时延等
SD-WAN切换时延sdDelay	s、ms	当网络状态发生变化时（如突然中断），动态改变路由使得数据在新的链路上传输的过程的时延
抖动Jitter	ms μs	网络时延的变化程度
丢包率Drop	-	传输中丢失数据包占所发送数据的比率

在生成微服务调度策略时，还会生成资源分配策略，分配给各微服务的资源包括计算资源、存储资源和带宽资源，微服务请求集合$S=\{s_1,s_2,\cdots,s_I\}$，分配给各微服务的计算资源、存储资源和带宽资源分别表示为：$C=\{c_1,c_2,\cdots,c_I\}$，$K=\{k_1,k_2,\cdots,k_I\}$，$B=\{b_1,b_2,\cdots,b_I\}$，计算资源和存储资源根据微服务的需求分配，带宽资源则需根据微服务对时延的需求和信道状况进行分配。根据资源分配可以计算微服务的传输时延tDelay_i，主要包括转发时延和链路时延。链路时延为t_{ij}^{link}，可由探针周期性探知。转发时延为t_{ij}^{trans}，是所需传输的数据总量d_{ij}^{all}和被分配带宽b_i的比值，具体公式为：

$$t_{ij}^{\mathrm{trans}}=\frac{d_{ij}^{\mathrm{all}}}{b_i}$$

则传输时延tDelay_i为：

$$\mathrm{tDelay}_i=t_{ij}^{\mathrm{link}}+t_{ij}^{\mathrm{trans}}=t_{ij}^{\mathrm{link}}+\frac{d_{ij}^{\mathrm{all}}}{b_i}$$

调度策略和资源分配必须满足微服务s_i所被调度到的节点m_j满足该微服务s_i对网络服务质量的要求SLA_i。微服务s_i对带宽、时延、SD-WAN切换时延、抖动和丢包率5种网络服务质量的要求为别为BW_i、tDelay_i、$\mathrm{sdDelay}_i$、Jitter_i、Drop_i，节点m_j可提供的相应的网络服务质量为BW_j、tDelay_j、$\mathrm{sdDelay}_j$、Jitter_j、Drop_j，则必须满足：$\mathrm{BW}_i\leqslant\mathrm{BW}_j$，$\mathrm{tDelay}_i\geqslant\mathrm{tDelay}_j$，$\mathrm{sdDelay}_i\geqslant\mathrm{sdDelay}$，$\mathrm{Jitter}_i\geqslant\mathrm{Jitter}_j$，$\mathrm{Drop}_i\geqslant\mathrm{Drop}_j$。

引入变量y_{ij}，用于说明微服务s_i部署在节点m_j上是否满足其对网络服务质量的要求，具体如下：

$$y_{ij}=\begin{cases}1, & \text{满足5条网络指标要求}\\ 0, & \text{不满足}\end{cases}$$

2.6 资源限制模型

各集群都有特定数量的可用计算资源。假设这些资源已经专门分配给每个节点。那么，分配给节点中的虚拟机资源在任何时候都不应该超过其总量。因此在生成资源分配策略时，需检查各节点被分配的计算资源和存储资源是否超出其总量。

微服务s_i所请求的计算资源为vCPU_i，存储资源为$\mathrm{Storage}_i$，节点m_j在时刻t可提供的计算资源为CPU_j^t，可提供的存储资源为STOR_j^t，则资源分配必须满足：

$$\sum_{i=1}^{I}\sum_{j=1}^{J}x_{ij}\times\mathrm{vCPU}_i\leqslant\mathrm{CPU}_j^t$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J}x_{ij}\times\mathrm{Storage}_i\leqslant\mathrm{STOR}_j^t$$

2.7 问题描述

本文的优化目标是在满足微服务对网络服务质量需求的同时降低本地端的能耗开销。微服务被成功地部署到节点上首先要保证该微服务请求的硬件资源小于该节点的硬件资源，其次还需考虑部署到该节点的能耗开销和服务质量，可以通过有约束的马尔可夫决策过程来进行建模。它的动作、状态、奖励和状态转移矩阵定义如下。

a）动作：动作包括微服务调度和资源分配分配决策，即$\hat{a}^t=\{o^t,c^t\}$，需注意的是，行动的组成部分应满足以下约束：

（a）$o_n^t\in\{0,1\}$是二进制微服务调度决策。

（b）$\sum_{m\in M}c_m^t\leqslant 1$，$0\leqslant c_m^t\leqslant 1$约束一个连续的计算资源分配决策。

b）状态：状态包括待处理的微服务S_i^t、节点空闲计算资源状况C_j^t、节点空闲存储资源状况R_j^t、各信道状况H_m^t，即

$$\hat{s}^t=\left\{\{S_i^t\}_{i\in I},\{C_j^t\}_{j\in J},\{R_j^t\}_{j\in J},\{H_m^t\}_{m\in M}\right\}$$

c）奖励：奖励的目的是使时隙t中调度的微服务整体能耗最小化，它被定义为：$\hat{r}^t=-D(o^t)=-E(o^t)$。

d）状态转换概率。状态转换概率公式如下：

$$P_r\left(\hat{s}^{t+1}\middle|\hat{s}^t,\hat{a}^t\right)=\prod_{i\in I}P_r(S_i^{t+1}\middle|S_i^t,o_n^t)\times\prod_{j\in J}P_r(C_j^{t+1}\middle|C_j^t,c^t)\times\prod_{j\in J}P_r(R_j^{t+1}\middle|R_j^t,c^t)\times\prod_{j\in J}P_r(H_j^{t+1}\middle|H_j^t)$$

不同状态成分的独立性、平等性成立，第1个状态受待处理微服务队列和微服务调度决策影响，第2个和第3个状态都受资源分配决策影响，第4个状态是根据信道条件的离散时间马尔可夫链演化的。需注意这些状态分量中的每一个都只取决于它之前的状态分量，因此状态转换符合马尔可夫模型。

本文的目标是找到一个静止的策略 $\pi \in \Pi$，根据状态 s^t 动态配置调度决策 o^t 和节点资源源分配 c^t，在满足微服务对服务质量要求的同时，使能耗最小化，这被表述为以下问题：

$$P_0: \min_{\pi \in \Pi} E_{\pi}(o^t)$$

$$s.t. \quad \prod_{\pi \in \Pi} y_{ij} = 1$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J} x_{ij} \times \mathrm{vCPU}_i \leqslant \mathrm{CPU}_j^t$$

$$\sum_{i=1}^{I}\sum_{j=1}^{J} x_{ij} \times \mathrm{Storage}_i \leqslant \mathrm{STOR}_j^t$$

3 用深度确定性策略梯度算法求解

本节使用深度确定性策略梯度（Deep Deterministic Policy Gradient，DDPG）算法解决上述问题。

DDPG是由深度Q网络（Deep Q-network，DQN）算法扩展开发的一种算法[14]。DDPG是一种具有处理高维和无限动作空间的深度强化学习算法，它试图为代理找到一种最大化奖励的行为策略，以完成其分配的任务。这种确定性策略梯度（Deterministic Policy Gradient，DPG）算法能够在连续动作空间上运行，克服了Q学习等经典强化学习（Reinforcement Learning，RL）方法的主要障碍[15]。

DDPG是基于演员-评论家算法的一种算法，演员-评论家架构如图2所示。它本质上是一种将策略梯度和价值函数结合在一起的混合方法。策略函数 μ 称为演员，价值函数 Q 称为评论家。本质上，演员输出的是根据当前环境状态从连续动作空间中选择的动作 $a = \mu(s|\theta^{\mu})$。评论家的输出 $Q(s,a|\theta^{\mu})$ 是一个具有时间差（Temporal Difference，TD）错误形式的信号，用于评价演员在了解当前环境状态的情况下所做的动作。

DDPG模型的训练阶段执行 M 个回合，其中每个回合迭代 T 次。使用索引 t 来表示单个回合中的迭代，其中 $t = 1,\cdots,T$。演员和评论家是用神经网络设计的。价值网络的更新是基于贝尔曼方程，通过最小化更新的 Q 值和原点值之间的均方差损失来实现。策略网络的更新是基于确定性的策略梯度定理实现。

还有一些实用的技巧被用来提高框架的性能。探索和利用之间的权衡是通过使用 ε-贪婪算法来实现的。在这种算法中，随机行动 a_t 是被以概率 ε 进行选择，否则就根据当前策略以概率 $1-\varepsilon$ 选择一个准确的行动 $a_t = \mu(s|\theta^{\mu})$。在训练阶段，一个大小为 B 的经验重放缓冲区 b 被用来打破时间上的相关性。与环境的每次互动都以 $[s_t,a,r,s_{t+1}]$ 的形式存储，s_t 是当前状态，a 是要采取的行动，r 是在状态 s_t 下执行行动 a 的奖励，s_{t+1} 表示下个状态。在学习阶段，算法使用从缓冲区随机抽取的数据集。同时，利用目标网络来避免直接更新网络权重与从TD错误信号中获得的梯度而产生的分歧。

4 实验与结果分析

为了验证本文提出的问题模型求解算法的有效性，本文使用数值结果来评估DDPG算法生成的调度

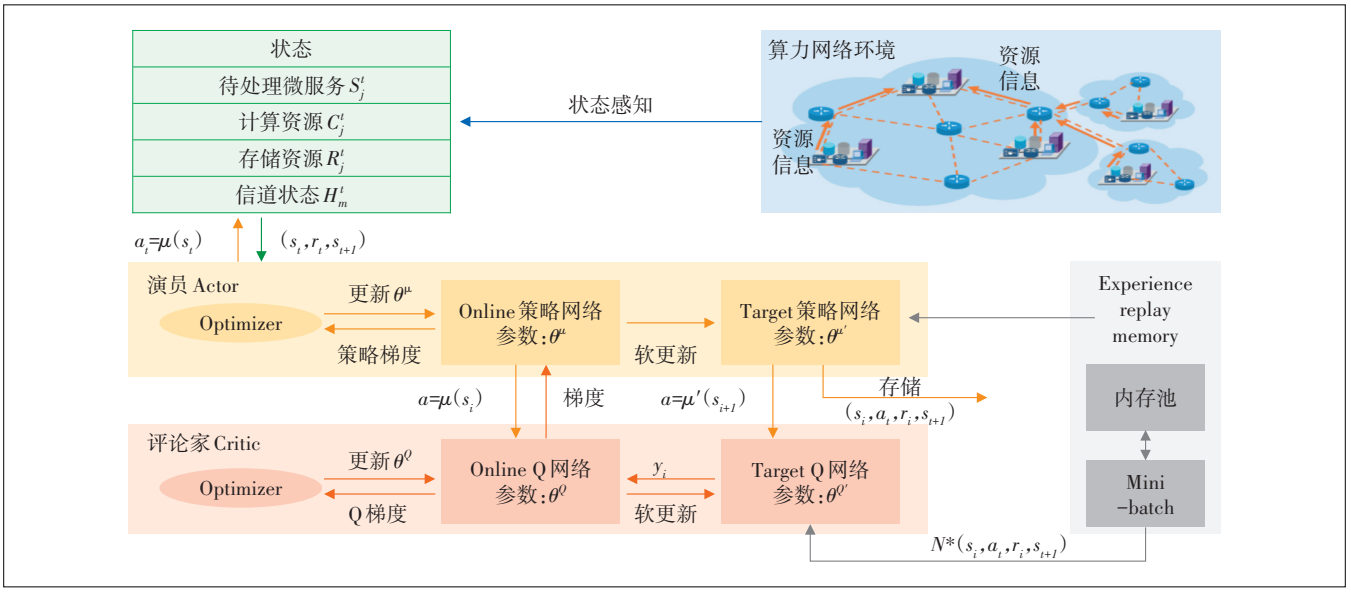

图2 演员-评论家架构

策略的性能。仿真环境基于Python 3.6，在100个回合训练后进行20个回合推理，并取20个推理结果的平均值进行进一步的分析和比较。为了使仿真结果更有说服力，本文使用3个基准解决方案进行比较，分别是：基于DDPG算法、基于平均调度和非合作模式（每个任务都在本地处理不进行调度）。

在仿真中，节点有150个微服务任务待执行，每个微服务所需的CPU周期数设置为3 Gcycles/s。节点CPU计算能力设置为8 Gcycles/s。DDPG算法的学习率和折现因子分别设为0.000 1和0.85。为了得到准确的性能估计，所有数值都是通过多次实验得到的。

本文评估了所提方案在不同计算能力下的性能，对比了5、10、15个算力网络服务节点状态下，3个解决方案的微服务调度效果，如图3所示。随着节点数量的增加，非合作模式下，微服务都在本地处理，传输时延为0，但由于本地计算能力的限制和能耗问题，服务效用低，因为微服务都在本地处理，故节点数量变化对非合作模式处理效用无影响。对平均调度算法和DDPG算法，待处理微服务数量不变，节点数量增加，对微服务处理效用提高。因平均调度算法未考虑各微服务对网络服务质量的需求和各节点能耗状态，故平均调度算法生成的调度策略效用低于DDPG算法。仿真实验结果证明本文由DDPG算法生成的SLA约束的能耗优化微服务调度策略是有效的。

5 总结

本文以SLA约束为出发点，研究了云计算场景下SLA约束的能耗优化微服务调度问题。本文考虑了微服务对网络服务质量的需求和云计算节点能耗优化，建立了微服务部署调度优化模型，并基于该模型选用DDPG算法生成微服务调度策略。仿真实验结果证明了该调度策略的有效性。

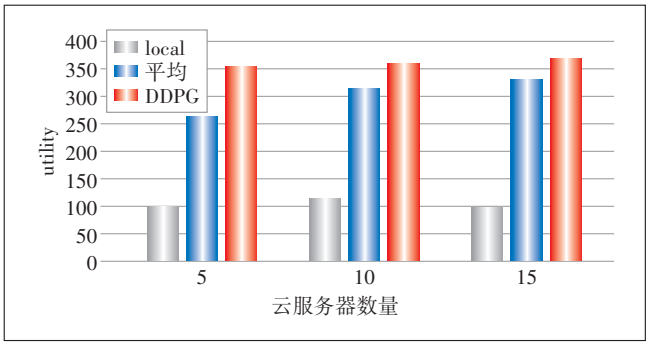

图3 不同节点数量不同方案的性能

参考文献：

[1] 杨烨. 基于算网一体化演进的算力网络技术研究[J]. 现代传输，2022(4)：45-48.

[2] 姚惠娟，陆璐，段晓东. 算力感知网络架构与关键技术[J]. 中兴通讯技术，2021，27(3)：7-11.

[3] 曹畅，唐雄燕. 算力网络关键技术及发展挑战分析[J]. 信息通信技术与政策，2021，47(3)：6-11.

[4] 段晓东，姚惠娟，付月霞，等. 面向算网一体化演进的算力网络技术[J]. 电信科学，2021，37(10)：76-85.

[5] 贾庆民，丁瑞，刘辉，等. 算力网络研究进展综述[J]. 网络与信息安全学报，2021，7(5)：1-12.

[6] PAPAZOGLOU M P，VAN DEN HEUVEL W J. Service oriented architectures：approaches，technologies and research issues [J]. The VLDB Journal，2007，16(3)：389-415.

[7] 方阿丽. 微服务框架技术及应用研究[J]. 电脑编程技巧与维护，2021(11)：53-55.

[8] SAMANTA A，LI Y，ESPOSITO F. Battle of microservices：towards latency-optimal heuristic scheduling for edge computing [C]//2019 IEEE Conference on Network Softwarization (NetSoft). Paris，France：IEEE，2019：223-227.

[9] SAMANTA A，TANG J H. Dyme：dynamic microservice scheduling in edge computing enabled IoT[J]. IEEE Internet of Things Journal，2020，7(7)：6164-6174.

[10] WANG S G，GUO Y，ZHANG N，et al. Delay-aware microservice coordination in mobile edge computing：a reinforcement learning approach[J]. IEEE Transactions on Mobile Computing，2021，20(3)：939-951.

[11] LV W K，WANG Q，YANG P F，et al. Microservice deployment in edge computing based on deep Q learning[J]. IEEE Transactions on Parallel and Distributed Systems，2022，33(11)：2968-2978.

[12] SULEIMAN H，BASIR O. SLA-driven load scheduling in multi-tier cloud computing：financial impact considerations [DB/OL]. [2022-11-02]. https://arxiv.org/abs/2111.03488v1.

[13] 周吉喆，王志勤，杨思远. 面向业务感知的算网融合关键技术研究[J]. 中兴通讯技术，2022，28(5)：1-6.

[14] MNIH V，KAVUKCUOGLU K，SILVER D，et al. Human-level control through deep reinforcement learning [J]. Nature，2015，518(7540)：529-533.

[15] BOUHAMED O，GHAZZAI H，BESBES H，et al. Autonomous UAV navigation：a DDPG-based deep reinforcement learning approach [C]//2020 IEEE International Symposium on Circuits and Systems (ISCAS). Seville，Spain：IEEE，2020：1-5.

作者简介：
刘博文，助理工程师，硕士，主要从事数据骨干网相关技术研究工作；梁晓晨，工程师，硕士，主要从事数据骨干网相关咨询设计与技术研究工作；张桂玉，毕业于吉林大学，高级工程师，主要从事数据网络的规划、设计工作；韩振东，毕业于清华大学，高级工程师，硕士，主要从事算网基础设施规划编制、新技术研究等相关工作。