DOI:10.16644/j.cnki.cn33-1094/tp.2022.11.013

基于深度强化学习改进的任务调度算法

叶芳泽，沈　炜

(浙江理工大学信息学院，浙江 杭州 310018)

摘　要：关于计算机系统与网络中的资源管理问题的研究无处不在，其中计算集群的调度算法一直是研究的热点。目前大多数解决方案为启发式调度算法，但启发式算法无法全面地感知系统中调度作业之间的关联性，而深度强化学习可以通过数据自主学习这些潜在的关联性。本文使用了一种基于动作分支架构改进的深度强化学习调度算法，在Spark调度模型中取得了不错的效果。该算法通过将一个完整的调度过程分解为相对独立的分支动作，从而简化各个动作设计过程并有效降低动作空间的维度。实验结果表明，在相同的训练时间内，该模型取得了较好的调度性能。

关键词：云计算；任务调度；图嵌入；深度强化学习

中图分类号：TP312　　**文献标识码：**A　　**文章编号：**1006-8228(2022)11-55-04

Improved task scheduling algorithm based on deep reinforcement learning

Ye Fangze, Shen Wei

(*School of Information Science and Technology, Zhejiang Sci-Tech University, Hangzhou, Zhejiang 310018, China*)

Abstract: Research on resource management in computer systems and networks is ubiquitous, among which the scheduling algorithm of computing clusters has always been a research hotspot. Most of the current solutions are heuristic scheduling algorithms. However, heuristic algorithms cannot fully perceive the correlations between scheduled jobs in the system, while deep reinforcement learning can learn these potential correlations autonomously through data. In this paper, an improved deep reinforcement learning scheduling algorithm based on the action branch architecture is used, and good results have been achieved in the Spark scheduling model. By decomposing a complete scheduling process into relatively independent branch actions, the algorithm simplifies the design process of each action network and effectively reduces the dimension of the action space. Experimental results show that the model achieves better scheduling performance within the same training time.

Key words: cloud computing; task scheduling; graph embedding; deep reinforcement learning

0 引言

近年来，云计算和大数据中心迅猛发展，全球数据量正呈爆炸式增长，数据成为当今社会增长最快的资源之一[1]。面对增长迅速又十分复杂的数据资源，传统单台高性能服务器的处理能力已经不能满足大部分网络服务和数据密集型应用的需求，取而代之的是商业服务器集群[2]。

目前主流的分布式集群计算框架通常使用简单通用的启发式调度算法。但是这类算法通常忽略了系统中作业之间的负载特性，放弃了潜在的性能优化[3]。如果想在分布式集群上使用启发式调度算法去实现高效地调度，通常需要针对特定的应用场景设计一个简化的调度模型，并在实验中精心调整并测试出较好的性能。一旦应用场景发生改变，算法就必须重新设计与测试。

近年来机器学习被成功应用于一些非常具有挑战性的决策控制领域其中就包括调度算法[4]，而机器学习中的深度强化学习既能利用深度学习捕获环境的特征，也能利用强化学习适应复制多变的调度场景[5]，越来越多的文献和实验也表明深度强化学习在调度领域有着巨大的潜力。

随着系统调度任务不断增多，调度过程也变得

收稿日期：2022-03-21

作者简介：叶芳泽(1997-)，男，浙江建德人，硕士，主要研究方向：人工智能、深度强化学习、调度算法。

通讯作者：沈炜(1973-)，男，浙江杭州人，博士，硕导，主要研究方向：人工智能、云计算。

复杂，传统强化学习算法中的状态空间与动作空间呈指数式增长，在训练过程中通常会出现训练时间长，学习效率低，结果不易收敛等问题。针对上诉问题，本文提出一种基于动作分支架构[6]改进的深度强化学习调度算法：利用分支策略网络分解动作空间，并用全局的评论家网络评估联合动作的优劣。实验表明改进的算法在仿真的Spark调度系统中，有着更好的调度性能。

1 深度强化学习

1.1 深度强化学习简介

深度强化学习的基础是强化学习，其本质是通过智能体与环境的不断交互，并在观察其行为的结果后根据环境的变化得到相应的奖励，以此调整自身的动作策略[7]。利用这种方式，可以学习如何改变自己的行为，以得到更大的奖励，其数学模型可由马尔可夫决策过程进行表示，具体形式可以用五元组表示为$\langle S,A,P,r,\gamma \rangle$。其中，$S$ 表示智能体的状态空间；A 表示智能体的可选动作空间；$P(s'|s,a)$ 表示当前状态s_t下执行动作a_t后，转移到下一状态s_{t+1}的状态转移概率密度函数；$r(s_t,a_t,s_{t+1})$表示当前状态s_t执行动作a_t后转移到下一状态s_{t+1}的瞬时奖赏；$\gamma(0<\gamma<1)$，表示未来奖赏折扣因子，决策过程如图1所示。而深度强化学习是在强化学习的基础上引入了深度学习中的神经网络，利用其强大的感知能力为复杂系统的决策问题提供强有力的技术支持。

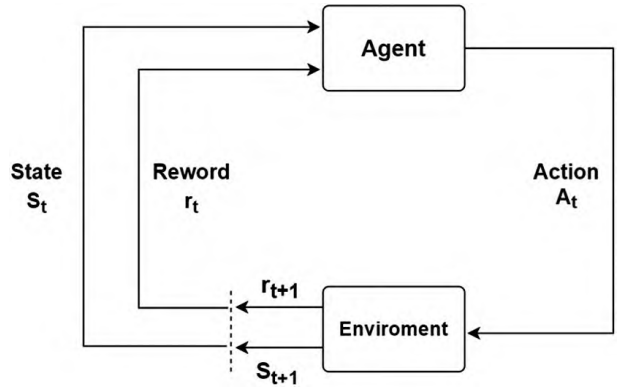

图1 强化学习示意图

1.2 A3C算法

A3C是一种基于策略的深度强化学习算法，结合了policy gradient和动态规划的优势，当系统给定一个初始的环境状态s_1，算法根据策略θ输出动作a_1的概率为$p_\theta(a_1|s_1)$，通过执行动作a_1环境状态变为s_2，并得到奖励r_1，此时状态转移概率为$p(s_2|s_1,a_1)$。重复上诉步骤直到任务结束，将一次调度过程记为$\tau=s_1,a_1,s_2,a_2,s_3,a_3\cdots s_T,a_T$。可以求出该调度过程发生的概率为：$p_\theta(\tau)=p(s_1)\,p_\theta(a_1|s_1)\cdots p_\theta(a_{T-1}|s_{T-1})\,p(s_T|s_{T-1},a_{T-1})$并得到累计奖励：$\mathrm{R}(\tau)=\sum_{t=1}^{T} r_t$，将两式相乘得到更新策略的目标公式：$\bar{R}_\theta=\sum_{\tau}\mathrm{R}(\tau)\,p_\theta(\tau)$。最后根据策略梯度公式(1)最大化我们得到的奖励：

$$\Delta\bar{R}_\theta\approx\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}\mathrm{R}(\tau^n)\Delta\log\left(p_\theta(a_t^n|s_t^n)\right) \tag{1}$$

为了降低τ的随机性，A3C用评论家网络(critic network)输出$Q_\pi(a_t^n|s_t^n)$值来代替奖励$\mathrm{R}(\tau^n)$，同时为了降低方差，让收敛更快，引入另外一个评论家网络输出$V_\pi(s_t^n)$作为baseline，至此共需要两个评论家网络来估计价值函数，注意到$Q_\pi(a_t^n|s_t^n)=E[\,r_t^n+V_\pi(s_{t+1}^n)\,]\approx r_t^n+V_\pi(s_{t+1}^n)$，可以用一个评论家网络简化算法训练过程，最终得到策略梯度公式(2)：

$$\Delta\bar{R}_\theta\approx\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n}(r_t^n+V_\pi(s_{t+1}^n)-V_\pi(s_t^n))\Delta\log\left(p_\theta(a_t^n|s_t^n)\right) \tag{2}$$

在训练过程中A3C算法引入异步方法，利用CPU多线程机制同时执行多个智能体。在学习模型训练的任意时刻，并行的智能体会根据不同的学习策略与环境状态进行交互，有效去除了样本间的关联性。

2 基于动作分支架构改进的深度强化学习调度算法

2.1 基于Spark的集群任务调度模型

本文通过模拟ApacheSpark的调度过程来验证算法的有效性[8]。Spark为每个输入系统中的作业(job)构建了一个有向无环图(DAG)，如图2所示。DAG中的每个节点(node)代表该作业的一个执行阶段(stage)，是一个或多个父节点的输出，当所有的父节点完成时，该节点才会被激活，并参与下次的调度计划；每个节点内有数个可并行计算的任务(task)，任务是Spark执行计算的最小单元，一个任务代表一个本地计算，不可拆分，且只能在一个执行器(executor)上执行计算过程，通常一个阶段的任务比执行器多，因此任务分几个批次运行；调度程序(Scheduler)负责维护任务队列，每次执行器空闲时，它都会从中分配任务。执行器是由Spark根据用户请求指派的，默认情况下，他们会一直运行直到系统中没有作业。

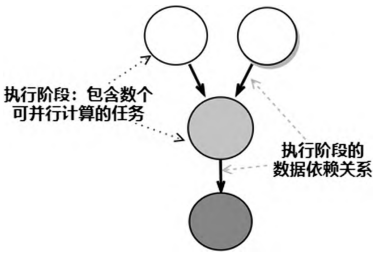

图2 spark系统作业示意图

Spark任务调度模块主要包含两大部分：DAG-Scheduler和TaskScheduler。其中DAGScheduler主要负责分析用户提交的应用，并根据计算任务的依赖关系建立DAG，然后将DAG划分为不同的stage，并以TaskSet的形式把Stage提交给TaskScheduler。TaskScheduler负责TaskSet中task的管理调度工作，这些task的执行逻辑完全相同，只是作用于不同的数据。调度过程如图3所示。

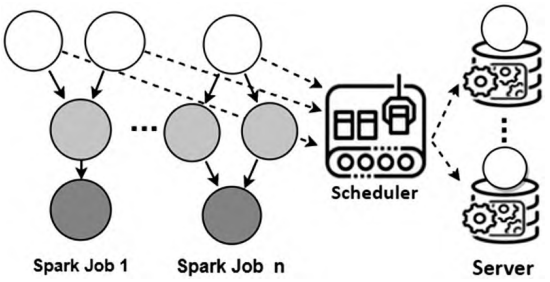

图3 调度示意图[3]

2.2 算法设计

2.2.1 基于图嵌入的状态空间预处理

图嵌入算法可以在保留DAG节点信息的同时将图的结构信息一并映射到低维稠密的特征向量中。首先提取DAG中每个节点的特征信息并将其表示成节点特征矩阵X，然后与邻接矩阵A作为图神经网络的输入，模型架构如图4所示。

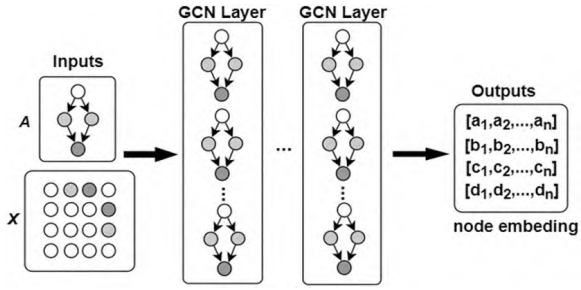

图4 多层GCN图嵌入模型[9]

在嵌入算法中使用多层GCN层间传播[10]，传播公式如下：

$$H^{(l+1)}=\sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{l}W^{l}) \tag{3}$$

其中，$\tilde{A}=A+I$，$\tilde{D}$为$\tilde{A}$的度矩阵，$\sigma(\cdot)$为激活函数，$H^{(\cdot)}$为各层激活函数，且$H^{(0)}$为X。为了提高层间传播效率，文章使用图上谱卷积一阶近似对于双层GCN模型，其前向传播公式为：

$$Y=f(X,A)=\sigma^{(1)}(\hat{A}\sigma^{(0)}(\hat{A}XW^{(0)})W^{(1)}) \tag{4}$$

其中，$\hat{A}=\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$。最终得到基于图神经网络的图嵌入模型$f(X,A)$对图结构进行编码，生成高质量的嵌入表示。

2.2.2 基于动作分支的深度强化学习调度算法

基于动作分支架构的思想，本文将传统的单策略网络分解为两个动作分支网络：①执行阶段策略网络，确定待调度的执行阶段；②执行器策略网络，确定上诉阶段应该分配的执行器数量。最后使用一个共享的评论家网络调整调度策略，整体架构如图5所示。为了确保执行器动作网络的输出有效性，本文规定：

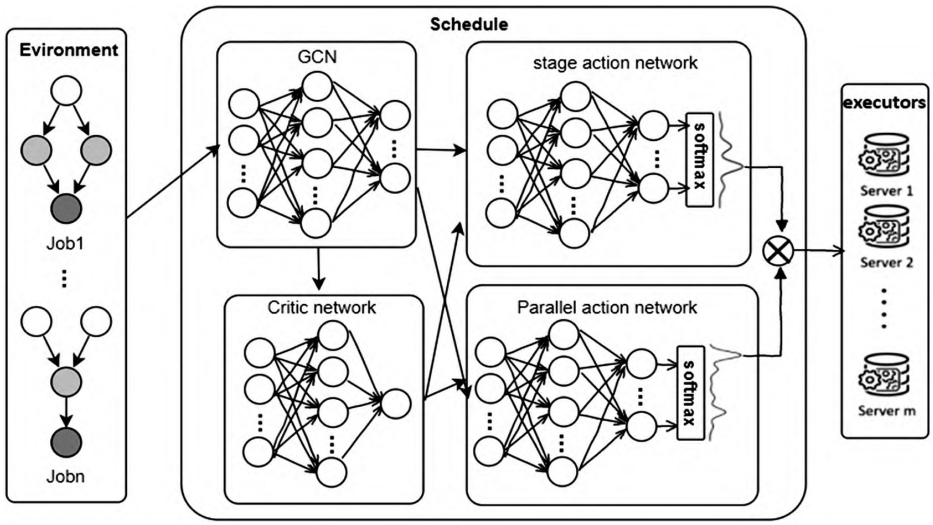

图5 基于动作分支架构的调度算法模型

执行器动作网络最少输出一个执行器，最多输出空闲执行器的数量；当该执行阶段内的剩余任务数量少于分配的执行器数量时，多余的执行器将继续激活系统调度程序，直到系统中没有可剩余可运行的任务。

当系统出现以下几种情况：①一个阶段用完任务(即不再需要更多的执行器)；②一个阶段完成，解锁它的一个或多个子阶段的任务；③一个新的作业到达系统。系统便会激活调度程序开始新的调度过程，调度程序通过将上文计算得到的图嵌入特征向量分别输入执行阶段策略网络、执行器策略网络和评论家网络，并输出待调度的节点、每个节点应该分配的执行器数量和期望奖励值。每完成一次执行器分配，系统会根据任务的到达时间、完成时间等数据计算此次调度得到的奖励。每当完成一轮完整的调度，系统根据公式⑵更新策略网络。

损失函数选择是非常关键的，它是深度强化学习的核心部分之一，网络模型在训练过程中使用TD-error作为损失函数，可以有效地找到最佳调度策略，Critic损失函数如下：

$$Critic_{loss} = \frac{1}{N}\sum_{n-1}^{N}\sum_{t=1}^{T_n}(r_t^n + V_\pi(s_{t+1}^n) - V_\pi(s_t^n))^2 \quad (5)$$

Actor损失函数如公式⑹所示：

$$Actor_{loss} = -\log p_\theta(a_t^n|s_t^n) * Critic_{loss} \quad (6)$$

3 实验分析

3.1 实验环境

本次实验采取ubuntu下的TensorFlow深度学习框架，使用Pycharm软件进行程序编写，实验平台的配置如下：CPU为Intel(R) Pentium(R) CPU G3260 @ 3.30 GHz，GPU为NVIDIA GTX 6G，1 TB硬盘。

3.2 训练

为了评估算法的有效性，我们使用文献[3]提供的TPC-H查询数据，从六个不同数据量(2,5,10,20,50,100SF)的全部22个TPC-H查询中随机抽样，输入Spark调度系统。系统初始状态为一个查询作业，15个任务执行器。训练开始时每间隔一段随机时间从数据集中获取20个查询作业组成一个批次输入系统等待调度，共计100个批次，为了模拟系统在高负载下的调度性能，要求这100个作业批次在规定时间内按泊松分布全部推送至系统，共计训练5000轮。

3.3 评估方法

本文选取了以下四个经典的调度算法与第二章提出的算法(下文简称B-A3C)进行对比。

⑴ FIFO：Spark默认的调度算法，它以作业到达的先后顺序进行分配执行器，并根据用户请求为每个作业授予相同数量的执行器。

⑵ SJF-CP：根据作业的总工作量，优先调度短作业，并在每个作业中优先选择关键路径上的节点。

⑶ Tetris*：基于Teris算法，平衡短期作业偏好和资源利用率得到一个综合评分，以此分配执行器。

⑷ A3C：仅使用一个策略网络，在相同的训练时间内比较算法的优劣。

3.4 结果与分析

表1展示了在不同的执行器数量下，各个算法的平均作业完成时间。从表1中的数据得知，随着执行器数量的增加，平均作业完成时间逐渐减少。其中当执行器数量较少时，两种深度强化学习算法的平均作业完成时间明显低于其他算法。当执行器数量达到29及以上时Tetris*算法的性能优于A3C算法。由此可以推测在只训练5000轮的情况下，A3C算法不能合理分配空闲的执行器，而B-A3C算法得益于分支动作架构的存在，可以在较短的时间内习得更为高效的调度策略。

表1 平均任务完成时间(单位：秒)

执行器数量	FIFO	SJF-CP	Tetris*	A3C	B-A3C
20	2289.474	1049.546	849.948	518.787	478.469
21	2080.565	819.130	695.040	450.098	430.567
22	1993.831	727.524	606.398	395.587	385.782
23	1895.555	574.885	483.706	327.112	307.072
24	1718.642	488.344	408.728	312.665	299.917
25	1562.412	407.664	342.370	304.870	284.017
26	1412.266	314.540	274.259	243.166	229.024
27	1275.041	252.231	226.548	196.600	190.732
28	1148.522	218.753	188.477	188.326	179.235
29	1025.599	171.211	142.832	146.609	139.573
30	928.440	146.787	138.131	140.157	136.800

(下转第64页)

multi-talker speech separation[C]//2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE,2017:241-245

[21] Kolbæk M, Yu D, Tan Z H, et al. Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks[J]. IEEE/ACM Transactions on Audio, Speech, and Language Processing,2017,25(10):1901-1913

[22] Liu Y, Thoshkahna B, Milani A, et al. Voice and accompaniment separation in music using self-attention convolutional neural network[J]. arXiv preprint arXiv:2003.08954,2020

[23] Y. Luo and N. Mesgarani, "TaSNet: Time-Domain Audio Separation Network for Real-Time, Single-Channel Speech Separation, " 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Calgary, AB,2018:696-700

[24] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing Ideal Time-Frequency Magnitude Masking for Speech Separation, " IEEE/ACM Transactions on Audio, Speech, and Language Processing,2019,27(8): 1256-1266

[25] Luo Y , Chen Z , Yoshioka T . Dual-Path RNN: Efficient Long Sequence Modeling for Time-Domain Single-Channel Speech Separation[C]//ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE,2020

[26] Chen J, Mao Q, Liu D. Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation[J]. arXiv preprint arXiv: 2007.13975,2020

[27] Zhang L, Shi Z, Han J, et al. Furcanext: End-to-end monaural speech separation with dynamic gated dilated temporal convolutional networks[C]//International conference on multimedia modeling. Springer, Cham, 2020:653-665

[28] Ditter D, Gerkmann T. A multi-phase gammatone filterbank for speech separation via tasnet[C]// ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE,2020:36-40

[29] KadiogluB, Horgan M, Liu X, et al. An Empirical Study of Conv-Tasnet[C]//ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).IEEE,2020

[30] Tzinis E, Venkataramani S, Wang Z, et al. Two-step sound source separation: Training on learned latent targets[C]//ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE,2020:31-35

(上接第58页)

4 结论

针对强化学习传统的单策略网络在复杂的调度系统中无法快速、高效地探索动作空间的问题，本文提出了一种基于动作分支架构的强化学习算法模型，通过将一个完整的调度动作分解为两个分支动作，有效地减小了各个分支的动作空间，提高了探索效率。最后通过实验证明在仿真的Spark调度模型中，相较于传统的调度算法、启发式算法和单策略网络的强化学习，该算法能有效的提升系统的调度性能。

参考文献(References):

[1] 陈天.深度强化学习在网络资源管理问题中的应用[D].电子科技大学,2019

[2] 李永峰,周敏奇,胡华梁.集群资源统一管理和调度技术综述[J].华东师范大学学报(自然科学版),2014(5):17-30

[3] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, Mohammad Alizadeh. Learning scheduling algorithms for data processing clusters[P]. Data Communication,2019

[4] 汪莹,陈新鹏.基于集群计算的任务调度算法研究[J].现代计算机,2020(9):4

[5] 赵德宇.深度学习和深度强化学习综述[J].中国新通信,2019(15):2

[6] Tavakoli A, Pardo F, Kormushev P. Action Branching Architectures for Deep Reinforcement Learning[J].2017

[7] 郑莹,段庆洋,林利祥,等.深度强化学习在典型网络系统中的应用综述[J].无线电通信技术,2020,46(6):603-623

[8] Apache Spark. 2018. Spark: Dynamic Resource Allocation. (2018). http://spark. apache. org/docs/2.2.1/job-scheduling.html#dynamic-resource-allocationSpark v2.2.1 Documentation.

[9] 袁立宇,李欣,王晓冬,等.图嵌入模型综述[J].计算机科学与探索,2022,16(1):59-87

[10] Defferrard M, Bresson X, Vandergheynst P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering[J]. 2016