基于强化学习的在线离线混部云环境下的调度框架

马玲[1]，樊漆亮[1]，许婷[1]，郭冠琛[2]，张圣林[1]，孙永谦[1]，张玉志[1]
（1. 南开大学软件学院，天津 300350；2. 北京大学计算机学院，北京 100871）

摘 要：目前针对云计算平台的强化学习调度算法考虑的场景较单一，或者忽略了任务的资源约束并简单地将所有机器看作同一类型，存在资源利用率较低及调度效率不高等不足。为了解决云环境中的在线离线混部调度问题，提出 JobFusion 框架。首先，通过集成带连通性约束的层次要素算法，在基于虚拟化技术的云计算平台中构建高效的资源划分方案；其次，为了解决扩展性问题，使用图卷积神经网络对具有任意层次约束关系及任意数量的任务进行嵌入，以捕获工作流的关键路径等信息；最后，集成了表现优异的强化学习模型对任务实施调度。实验结果表明，相较对比方法，JobFusion 提高了 39.86%的资源利用率，且最多降低了 64.36%的平均任务完成时间。

关键词：强化学习；图嵌入；层次聚类；云计算；虚拟化
中图分类号：TP18
文献标志码：A
DOI: 10.11959/j.issn.1000−436x.2023119

Scheduling framework based on reinforcement learning in online-offline colocated cloud environment

MA Ling[1], FAN Qiliang[1], XU Ting[1], GUO Guanchen[2], ZHANG Shenglin[1], SUN Yongqian[1], ZHANG Yuzhi[1]
1. College of Software, NanKai University, Tianjin 300350, China 2. School of Computer Science, Peking University, Beijing 100871, China

Abstract: Some reinforcement learning-based scheduling algorithms for cloud computing platforms barely considered one scenario or ignored the resource constraints of jobs and treated all machines as the same type, which caused low resource utilization or insufficient scheduling efficiency. To address the scheduling problems in online-offline colocated cloud environment, a framework named JobFusion was proposed. Firstly, an efficient resource partitioning scheme was built in the cloud computing platform supporting virtualization technology by integrating the hierarchical clustering method with connectivity constraints. Secondly, a graph convolutional neural network was utilized to embed the attributes of elastic dimension with various constraints and the jobs with various numbers, to capture the critical path information of workflow. Finally, existing high-performance reinforcement learning methods were integrated for scheduling jobs. According to the results of evaluation experiments, JobFusion improves the resource utilization by 39.86% and reduces the average job completion time by up to 64.36% compared with baselines.

Keywords: reinforcement learning, graph embedding, hierarchical cluster, cloud computing, virtualization

0 引言

对于拥有大量机器和部署云计算平台的公司而言，服务器资源利用率较低和基础设施总成本逐年上升已成为比较棘手的问题。

目前，资源利用率不高主要有以下 2 个原因。一方面，由于存在业务访问量的潮汐现象和资源的冗余分配策略，在线任务申请资源时往往根据流量峰值来评估资源需求并以此申请资源，同时上调额度作为预留。此外，考虑容灾备份时还可能部署多个冗余副本，在这种情况下，低峰时的资源利用率非常低。另一方面，在进行资源规划时，离线任务

收稿日期：2023−02−16；**修回日期：**2023−06−13
通信作者：张圣林，zhangsl@nankai.edu.cn
基金项目：国家自然科学基金资助项目（No.62272249, No.61901234）
Foundation Item: The National Natural Science Foundation of China (No.62272249, No.61901234)

与在线任务通常是分离规划的，前者的重心在于扩大计算和存储可用的资源份额与基础设施的规模，而后者的重心在于使用户获得更好的访问体验并满足实时性需求。这种情况下，在线任务和离线任务的主要特点是在线资源利用率低、离线资源利用率高。如图 1 所示，在云原生环境中，在线任务一般是一些需要长期运行且对实时性要求较高的任务，如 Web 服务和数据库服务，由于这类任务在任意时刻都需要做出响应，因此需要保证一定的服务质量，对稳定性要求较高；而离线任务一般是一些定时任务，如数据库迁移任务和模型训练任务，这些任务对实时性的要求较低且对运行失败容忍度较高，仅在某些时段内短期运行。

图 1　在线任务和离线任务的主要特点

为了降低基础设施的整体成本，目前迫切需要回收被浪费的在线任务的资源并将其利用到离线任务的资源供应上，以降低基础设施建设的总成本。在线任务相比离线任务具有实时性高的要求，因此在混部场景下，应优先考虑在线任务的资源需求。

为了解决上述问题，一些规模较大的公司已经开始研究并发展在线离线混部调度技术，以期获得更大收益。该技术使用虚拟化方法将在线任务和离线任务混合部署到相同的物理机上，通过资源隔离、调度等手段充分利用机器资源。在保证服务的稳定性、满足在线任务实时性要求的同时，最大化地完成离线任务。

本文的主要贡献如下。1) 本文将具有连通性约束的层次聚类算法用于云计算平台的机器资源划分，从而提高资源利用率；2) 针对在线离线混部场景和多维资源约束问题，本文提出 JobFusion 调度框架，通过使用图卷积网络（GCN, graph convolutional network）对变长信息进行层次化编码嵌入，利用节点嵌入、有向无环图（DAG, directed acyclic graph）嵌入和全局嵌入这三层嵌入作为输入，并使用强化学习调度框架输出调度动作，以提高任务调度效率。基于阿里云生产环境数据的实验表明，相比现有方法，JobFusion 提高了 39.86%的资源利用率，且最多降低了 64.36%的平均任务完成时间。

1　相关工作

在虚拟化技术发展早期，启发式算法在解决云计算平台的任务调度优化问题中应用广泛[1]。鉴于云计算平台中任务之间的拓扑依赖关系复杂多变，国内外学者针对云计算平台的调度问题进行了以下研究。

为了解决资源利用率不高的问题，百度公司已经搭建并且维护了诸多 Kubernetes 集群，进行了基于在线离线混部技术的原生化改造。混部集群中 CPU 利用率比在线任务集群 CPU 利用率提升 40%～80%，累计节省了近 10 万台服务器。鉴于启发式算法难以捕捉任务拓扑关系来构建高维特征上的决策空间，且单一的调度优化策略不易适应复杂多变的工作流，适用场景较窄。随着深度学习技术的快速发展，一些研究者使用图神经网络（GNN，graph neural network）或 GCN 来对任务属性信息（如任务完成所需时间、子作业数量等）以及任务之间的依赖关系进行编码嵌入，从而得到一个高效且具有全局视角的特征表示。

针对启发式调度算法过于单一和朴素的问题，Mao 等[2]提出了基于深度强化学习的调度方法 Decima，通过结合 GCN 和 A3C（asynchronous advantage actor critic）方法[3]解决了随机到达任务的调度问题。针对云计算平台的云函数调度问题，Yu 等[4]提出了 FaaSRank 调度方法，使用强化学习和深度神经网络来自动地学习调度策略，能够减少云函数的调用次数以及云函数的平均完成时间。在边缘计算环境中，Zhang 等[5]使用包含多智能体的深度强化学习方法来解决动态调度问题。为了解决深度强化学习调度任务中的伸缩性问题，Song 等[6]使用 GNN 来对任务节点的信息进行嵌入以捕捉复杂的任务依赖关系，并且将任务阶段选择与机器调度组合为一个动作向量作为强化学习智能体的输出。Zhou 等[7]提出一个可以对强化学习调度算法进行动态选择的调度框架，为复杂多变的云计算平台自动选择最优调度算法。Sheng 等[8]使用差分反馈策略和基于历史信息的采样算法，在考虑机器内存架构差异的基础上进行高效的任务调度。

现有的强化学习调度算法的不足之处在于没有考虑到云计算平台基础设施的异构问题。借助虚拟化和资源隔离技术，云计算平台可能部署着数量众多、资源类型迥异的容器。忽略这些容器间的资源差异而直接将其用于任务调度，将导致较多的资源浪费，间接影响最终的任务平均完成时间。随着

云计算业务规模的迅速扩大，在线离线混部场景的应用也愈发广泛。云计算平台的任务主要分为两类：一类是对实时性要求较高的在线任务；另一类是对实时性要求较低的离线任务。一些学者对在线离线混部场景的调度问题进行了如下研究。

Li 等[9]针对在线离线混部场景下的任务动态调度问题，提出了一种基于蒙特卡罗树搜索的调度算法，解决了在线任务的动态变化给调度带来的机器资源实时波动问题。为了解决微服务和云原生架构场景下在线离线混部调度问题，Zhang 等[10]利用 Kubernetes 的扩展性，提出了高扩展性集群调度系统 Zeus，能够根据集群负载实时调整 2 种任务的资源分配。在多租户的云计算系统中，Pi 等[11]基于用户态的多线程技术，提出 Holmes 调度方法，针对混部场景下低时延和高资源利用率 2 个目标完成任务的高效调度。

在具有异构容器资源的调度场景下，针对多维资源约束以及在线离线混部调度问题，本文提出了通用的任务调度框架 JobFusion，对 Decima[2]的工作进行了改进和扩展：1) 针对提高在线离线混部场景下云计算平台资源利用率的问题，提出了一种带连通性约束的层次要素算法，用以生成从物理机到容器的合理资源划分方案；2) 不同于 Decima 只能处理单一维度的任务资源约束问题并使用相同的容器类型用于调度，JobFusion 将其扩展到了异构场景下的多维资源约束问题。

2 方法介绍

在生产环境中，容器作为一个独立节点，通常是任务调度的最小单位，而从物理机到容器的创建过程需要预先设定一个合理的资源划分方案。此外，在复杂的云计算环境中，一个任务通常由多个运行阶段构成。同时每个阶段又可以细分为一系列并行的子作业，具有 DAG 依赖关系的并行任务在云计算平台中较常见。

针对上述资源划分方案的自动生成问题，JobFusion 采用了带连通性约束、自底向上的层次要素算法。为了解决任务之间依赖关系的嵌入，JobFusion 使用 GCN 对任务的属性信息进行编码和映射，得到最终的任务嵌入向量。基于自动生成的资源划分方案所得到的用于调度的容器集合和 GCN 所输出的任务嵌入向量，将作为强化学习智能体的输入用于预测下一时刻将要运行的任务阶段和将要调度的容器。在强化学习智能体做出决策后，向集群环境更新状态信息，此时新的无前驱依赖的任务阶段加入等待队列中准备接受调度。而强化学习智能体根据集群反馈的状态信息计算每次决策的奖励分数，以更新智能体中策略网络的模型参数，进而优化智能体的决策调度。JobFusion 框架的整体流程如图 2 所示。

本节将首先介绍 JobFusion 中带连通性约束的

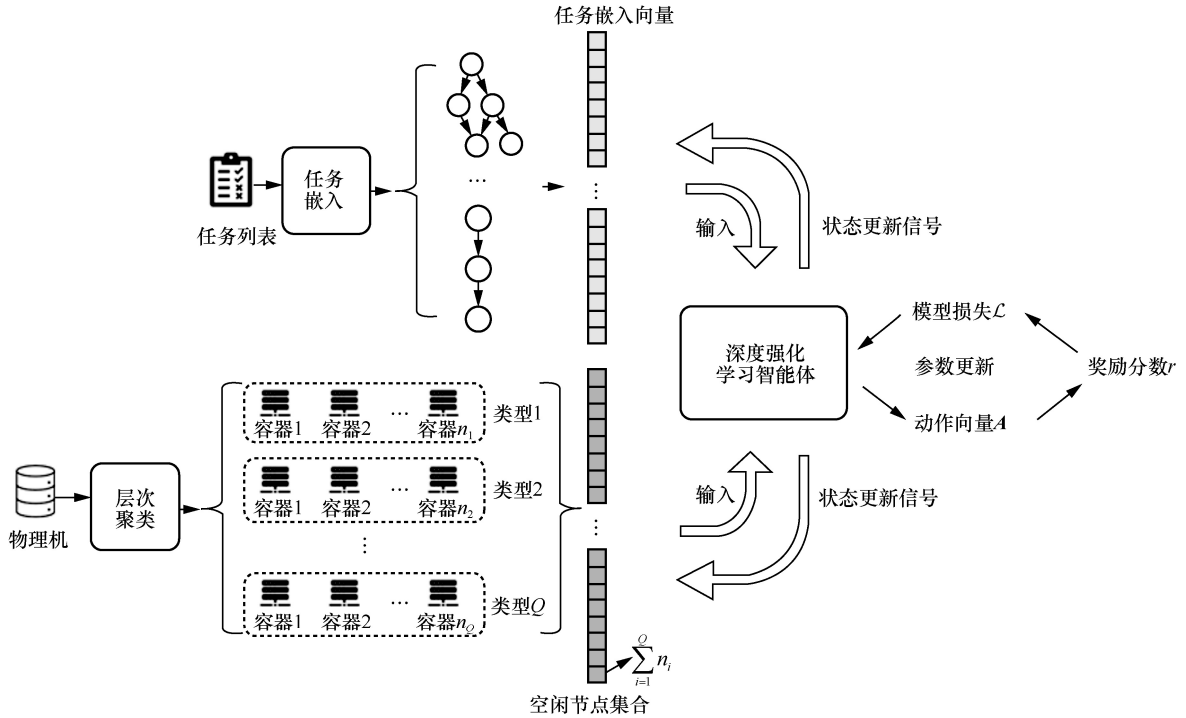

图 2 JobFusion 框架的整体流程

层次要素算法，然后介绍深度强化学习智能体的输入和输出——嵌入向量与动作向量，最后介绍基于强化学习的调度方法。

2.1　基于层次聚类的资源划分

在云计算平台中，物理机的资源配置通常要远高于单个任务所需的资源。一台物理机可以通过虚拟化技术创建大量资源相互隔离、互不感知的容器，一个合理的资源分配方案可以有效提高资源利用率以及任务调度效率[12]。为此学者提出了一些资源管理方案[13]，这些方案主要分为静态资源管理方案和动态资源管理方案。前者通常使用预测模型根据历史信息进行预测，但是云计算平台中历史任务众多，该类方案效率较低；后者则采用聚类方法进行资源划分，其根据短期任务的资源需求情况进行快速聚类，减少资源需求的类别数目，以便减小后续调度过程中动作空间。

JobFusion 在动态资源管理方案的基础上为聚类方法增加了连通性约束，充分利用了资源类别空间中的拓扑信息，提出了一种与之相适应的层次聚类算法进行高效的资源管理。层次聚类中每个元素代表一种特定的资源需求向量。任意给定一个任务，其拥有一个多维资源约束向量 $\boldsymbol{r}=(r_1,r_2,\cdots,r_n)$，表示要运行此任务所需的资源份额，每个元素代表一种资源，如 CPU 或者内存，其中 n 表示资源的种类数。

距离度量。自底向上的层次聚类是一个聚合的过程，假设在某一步中将簇 c_1 和 c_2 合并为新的簇 c，从而使所有样本的类间距离 $d(\cdot,\cdot)$ 变大。假设簇 c_1 和 c_2 的质心分别为 g_1 和 g_2，2 个簇的类间距离可以定义为

$$d(c_1,c_2)=\frac{|c_1||c_2|}{|c_1|+|c_2|}\|\boldsymbol{g}_1-\boldsymbol{g}_2\|^2 \tag{1}$$

当簇 c_1 和 c_2 都只有一个元素时，两者的类间距离等于 2 个元素之间欧氏距离的一半。

连通性约束。加上连通性约束后，2 个簇的类间距离可以改写为

$$d'(c_1,c_2)=\begin{cases}d(c_1,c_2), & \boldsymbol{g}_1\leqslant\boldsymbol{g}_2 \text{ 或 } \boldsymbol{g}_1\geqslant\boldsymbol{g}_2\\ -\infty, & \text{其他}\end{cases} \tag{2}$$

这里进行了 2 个向量大小的判定，因此本文约定：当且仅当向量中相应位置元素都满足某一大小关系，则向量 $\boldsymbol{g}_1$ 和向量 $\boldsymbol{g}_2$ 满足该大小关系。注意到，合并 2 个簇的目的是让类间差异最大化，因此当 2 个簇的质心之间不满足连通性约束时，对应的簇将无法合并。加入该连通性的原因是如果每个元素都是一个资源约束向量，那么一个大的约束向量必然能“兼容”一个更小的资源约束向量，即如果有一个资源份额较多的容器能满足较大任务的运行，那么其必然能满足较小任务的运行。也正是基于这一点，JobFusion 在层次聚类中引入了连通性的概念，来表示 2 个资源约束向量之间是否“兼容”。

在极端情况下，聚类的某一阶段中任意 2 个簇的质心都无法满足连通性约束，此时该算法将退化到普通的完全基于距离度量的层次要素算法，并且在选取质心的过程中可能创建出新的资源需求向量。

选择质心。层次聚类的最终目的是缩减任务的资源需求向量的数量，因此质心作为一个簇的代表的同时也需要表示一个资源约束向量。从而当聚类结束时，只需取出每个簇的质心，即可得到最终的资源需求向量集合，再将物理机的资源按相应份额进行划分，得到容器集合后用于任务调度。传统的质心选取以距离度量为标准，但是在这里有可能违背约束向量之间的连通性约束，从而在聚类的过程中又不断创建出新的资源约束向量作为质心。假设以距离度量得到的质心为 $\boldsymbol{g}$，存在一个元素 $\boldsymbol{r}$ 使 $\boldsymbol{r}>\boldsymbol{g}$，在这种情况下，$\boldsymbol{g}$ 无法兼容 $\boldsymbol{r}$。如果一个质心对应的容器类型无法兼容其簇中任务的资源约束向量，那么它将会抢占其他类型容器的资源，最终导致资源利用率下降。

因此，JobFusion 在进行质心选取时完全依据连通性约束，将一个簇中的最大元素作为质心，以兼容簇中所有其他元素。

2.2　节点嵌入与决策动作编码

针对任务资源需求的不同，通过使用带连通性约束的层次要素算法，JobFusion 将机器资源划分为 Q 类容器，以减小后续调度过程中的决策空间。

2.2.1　嵌入向量

在分布式并行数据处理系统中，不同的任务之间通常具有依赖关系，且主要以 DAG 的形式体现。每个任务由并行或相互依赖的不同阶段构成，因此任务的运行阶段可以表示为 DAG 的节点。

由于任务的数量不是固定的，任务内部阶段的数目不同，其任务内部阶段的依赖关系也不尽相同，甚至阶段内部的作业数目也存在差异。如何将

这些变动的依赖信息和节点属性进行编码是首要解决的问题。GCN 建立了图到图的映射，对图节点的数量、节点之间的依赖关系没有定量约束，GCN 每层的计算过程可以表示为

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

其中，$\hat{\boldsymbol{A}}=\boldsymbol{A}+\boldsymbol{I}_N$ 表示图 G 加上自环后的邻接矩阵；通过左乘和右乘一个对角矩阵 $\hat{\boldsymbol{D}}^{-\frac{1}{2}}$，将邻接矩阵 $\hat{\boldsymbol{A}}$ 按行进行归一化，其中 $\hat{\boldsymbol{D}}_{ii}=\sum_j \hat{\boldsymbol{A}}_{ij}$；$\boldsymbol{W}^{(l)}$ 表示 GCN 第 l 层的权重矩阵；$\boldsymbol{H}^{(l)}$ 表示 GCN 第 l 层的激活函数输出；$\sigma(\cdot)$ 表示激活函数；$\boldsymbol{H}^{(0)}$ 表示输入数据，包含了所有任务阶段的属性向量，每个属性向量包含任务阶段的 CPU 资源需求、内存资源需求、总作业数量、已完成作业数量。从式(3)中可以看到，权重矩阵的维度只与输入矩阵的最后一维相关。基于此，GCN 可用于处理变长信息的嵌入，以解决上文所提到的依赖关系多变和任务数量不定这 2 个问题。

本文使用的 GCN 对任务信息的嵌入过程主要分为 3 个层次。1) 节点嵌入：该层次的嵌入用于捕获节点自身及其子孙节点的属性信息，可以体现关键路径等其他特性，如图 3 所示。其中，实线表示节点之间的依赖关系；虚线表示节点嵌入的计算过程，虚线尾部表示计算的输入，头部（箭头）指向最终计算结果，如 e_1 表示节点 1 的嵌入结果。深灰色节点表示当前要计算嵌入向量的目标节点；浅灰色节点表示深灰色节点的直接子节点，用来参与计算父节点（深灰色节点）嵌入信息；白色节点则表示当前步骤中暂未参与嵌入计算的节点。2) DAG 嵌入：该层次的嵌入用于综合 DAG 中所有节点嵌入结果与节点属性信息，可以体现 DAG 中全部任务阶段，不同层次嵌入之间的关系如图 4 所示。3) 全局嵌入：该层次的嵌入聚合了所有 DAG 的嵌入结果，可以体现全局属性（如任务总数量等）。

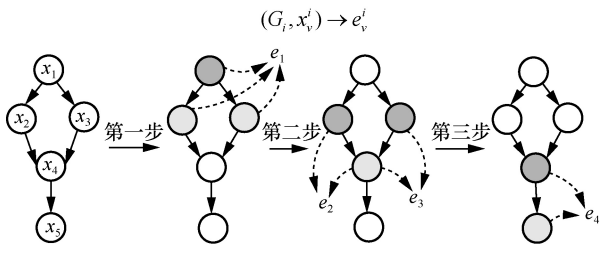

图 3 节点的嵌入过程

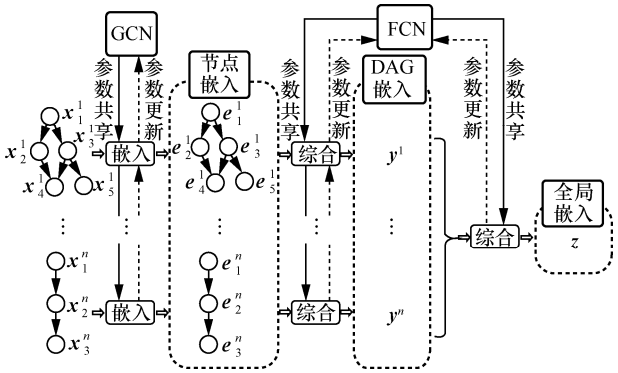

图 4 不同层次嵌入之间的关系

节点嵌入。对于一个给定的 DAG G_i 以及节点 v 的属性向量 $\boldsymbol{x}_v^i$，如果 $\xi(v)$ 表示 v 的所有子节点，那么节点 v 的嵌入过程可以表示为

$$\boldsymbol{e}_v^i=g\left[\sum_{u\in\xi(v)} f(\boldsymbol{e}_u^i)\right]+\boldsymbol{x}_v^i \tag{4}$$

其中，$f(\cdot)$ 和 $g(\cdot)$ 是由神经网络实现的非线性函数。通过叠加多个非线性层，允许嵌入过程中能学习并实施类型多样的聚合函数（如求均值或者求和）。此外，不同任务之间 GCN 和非线性层的参数是共享的，不仅可以减少模型的计算量，而且可以综合任务的平均情况。

DAG 嵌入和全局嵌入。如图 4 所示，每个 DAG 的嵌入结果是 DAG 中每个节点嵌入结果的综合，该过程可以由足够多层的 GCN 实现。全局嵌入结果又是所有 DAG 嵌入结果的综合，非线性层不仅聚合了所有全局嵌入，而且拟合了相应的聚合函数，从而得到 DAG 嵌入的结果 $\boldsymbol{y}^i$ 和全局嵌入 $\boldsymbol{z}$。

2.2.2 动作向量

将得到的嵌入向量和空闲容器集合作为输入后，强化学习智能体的策略网络将输出一个动作向量用以表示调度动作。

假设总共有 N 个任务阶段，调度器每次只选择一个任务阶段并只为其分配一个空闲容器，那么每次决策的动作空间的复杂度仅为 $O(N)$。但是，当任务数量较多时，强化学习的每个轮次都会因为完全串行化的调度而效率低下。但如果每一次调度为所有任务阶段分配对应机器，那么巨大的动作空间又将导致调度过程中内存溢出。

在强化学习中，长的动作序列和大的动作空间都会影响训练进程，需要在两者之间寻求一个平衡。为了平衡动作序列长短和动作空间大小，JobFusion 输出的动作向量被定义为一个三元组：将

要调度的任务阶段、分配的容器类型和对应容器数量。在 t 时刻，强化学习智能体将 3 个层次嵌入向量的拼接结果 $\boldsymbol{q}_v^i$ 作为输入，并输出一个三维的动作向量 $\boldsymbol{a}_t$。

2.2.3　阶段选择与容器分配

动作向量的输出过程如图 5 所示，强化学习智能体针对时刻 t 所处状态 $\boldsymbol{s}_t$ 决策出一个调度动作，策略网络按照如下规则选出一个任务阶段用于调度。策略网络由若干层全连接网络（FCN, fully connected network）堆叠而成。

对于任务 i 中的一个节点 v，策略网络计算出一个优先级分数 $p_v^i \triangleq p(\boldsymbol{q}_v^i, \boldsymbol{y}^i, \boldsymbol{z})$，$p(\cdot)$ 表示评分函数，是由神经网络所实现的非线性层，可以将输入的嵌入向量映射为一个标量；分数 p_v^i 表示任务阶段 v 被调度的优先级。通过使用 Softmax 函数计算每个任务阶段被选择的概率，计算过程为

$$P(\text{node}=v)=\frac{\exp(p_v^i)}{\sum_{u\in S_t}\exp(p_u^{j(u)})} \tag{5}$$

其中，S_t 表示在时刻 t 所有可调度的节点集合，$j(\cdot)$ 表示某一个阶段所属的任务，整个计算过程保证了可微分性质。

类似于任务阶段的选择过程，为对应任务分配，容器也是通过 Softmax 函数基于相应的优先级分数得到每个容器被选中的概率实现的，同时对于不满足任务运行要求的容器会施加掩码，使其不可被选择。

2.3　基于强化学习的调度方法

在线离线混部的集群环境中包含两类任务：在线任务和离线任务。由于在线任务具有实时性高的要求，因此在线任务的执行遵循先到先服务的原则，并且当在线任务资源需求无法得到满足时会随机中止离线任务的执行以满足在线任务的资源需求。每一时刻由于运行的在线任务存在差异，离线任务可用的资源总量也不尽相同。因此本文使用的深度强化学习调度方法主要用于离线任务的调度过程。

JobFusion 的强化学习框架如图 6 所示。强化学习智能体通过接收环境的状态信息决策出一个调度

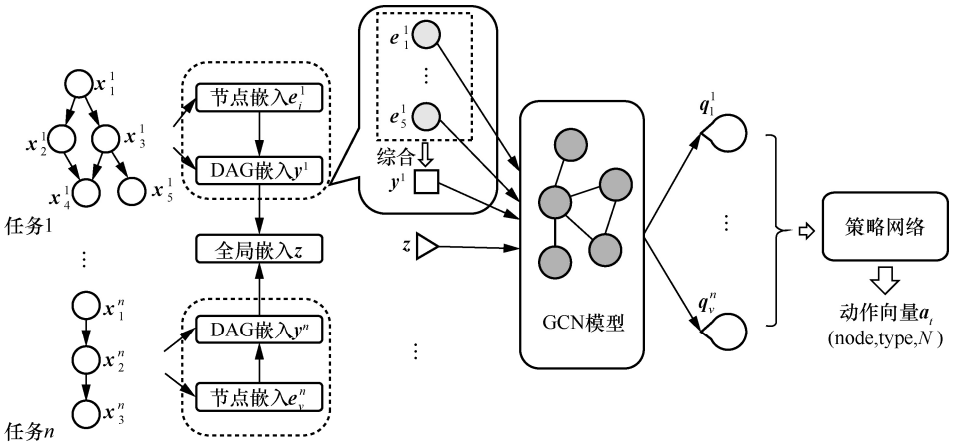

图 5　动作向量的输出过程

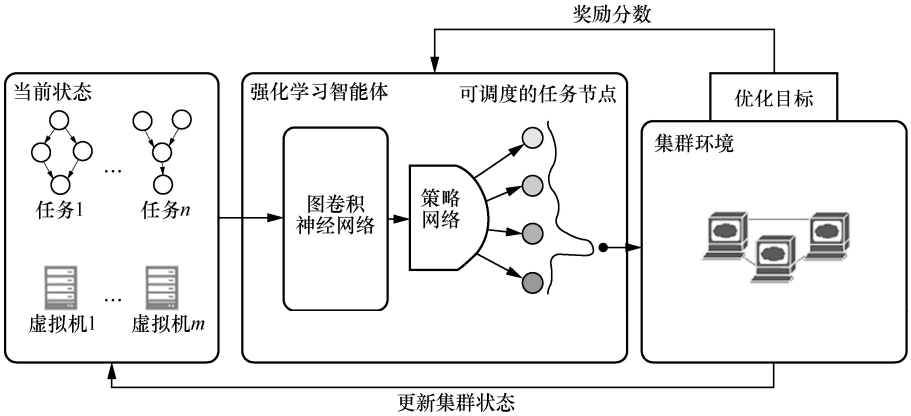

图 6　JobFusion 的强化学习框架

动作，然后根据优化目标计算出一个奖励分数。通过最大化奖励分数，使策略网络不断更新自身参数。

在时刻 t，定义强化学习的状态信息 $\boldsymbol{s}_t$ 为一个二元组，包括所有容器的空闲状态和任务阶段的完成情况；动作向量 $\boldsymbol{a}_t$ 为一个三元组，包括将要执行的任务阶段、被选择的容器类型和分配的容器数量。强化学习的奖励分数 r_t 是一个衡量强化学习智能体调度策略优劣的标量。

强化学习以轮次为单位进行训练，每轮包含若干个调度事件，每个调度事件包含一个或多个调度动作。定义 T 为一轮训练中调度动作的总数量，t_k 为第 k 个调度动作发生的时间点。为了让强化学习智能体沿着正确的方向学习，JobFusion 会向智能体反馈一个对应第 k 个调度动作的奖励信息 r_k。当需要最小化任务平均完成时间时，JobFusion 将最大化奖励分数 $r_k=-(t_k-t_{k-1})J_k$，其中 J_k 为第 k 个调度动作发生时的任务总数量。当一个任务中所有作业都执行完毕时，该任务标记为完成状态。因此如果策略网络输出的动作向量优先调度执行时间短、剩余作业少的任务时，t_k-t_{k-1} 和 J_k 的值也会相应减少，奖励分数就会相应提升；反之，当策略网络输出的动作向量优先调度执行时间长的任务时，每两次调度之间的跨度也会增大。如果优先调度剩余作业多的任务，未完成的任务就会不断堆积，导致 J_k 偏高，两者共同作用导致奖励分数较低，损失函数值较高。因此，强化学习算法的总体目的是最小化时间跨度上的期望 $\mathbb{E}\left[\dfrac{1}{t_T\sum_{k=1}^{T}(t_k-t_{k-1})J_k}\right]$。该目标能够最小化系统中任务数量，根据利特尔法则[14]，这能够有效优化任务平均完成时间。

通过采用上文描述的带连通性约束的层次聚类进行资源划分，得到用于调度的容器集合。策略网络首先根据训练过程中的奖励分数计算神经网络的梯度信息，再使用梯度下降算法进行优化。损失函数 $\mathcal{L}$ 的计算方式为

$$\mathcal{L}=\sum_{k=0}^{T}\log\pi_\theta(\boldsymbol{s}_k,\boldsymbol{a}_k)\left(\sum_{k'=k}^{T}r_{k'}-b_k\right) \tag{6}$$

其中，θ 是策略网络的参数；$\boldsymbol{s}_t$ 是时刻 t 的状态向量；$\boldsymbol{a}_t$ 是时刻 t 输出的动作向量；$\pi_\theta(\boldsymbol{s}_t,\boldsymbol{a}_t)$ 是策略网络以 $\boldsymbol{s}_t$ 为输入时，输出 $\boldsymbol{a}_t$ 的概率值；b_k 是先前所有训练轮次在第 k 步之前奖励分数的累加，因而 $\sum_{k'=k}^{T}r_{k'}-b_k$ 代表了某一步中的奖励分数与平均情况的相对大小。由于 $\pi_\theta(\boldsymbol{s}_k,\boldsymbol{a}_k)\in[0,1]$，因此 $\log\pi_\theta(\boldsymbol{s}_k,\boldsymbol{a}_k)$ 总为负值。随着不断优化，$\pi_\theta(\boldsymbol{s}_k,\boldsymbol{a}_k)$ 和 $\sum_{k'=k}^{T}r_{k'}-b_k$ 的值也将不断增大，从而使损失函数的值不断下降。JobFusion 的参数更新方式为

$$\theta\leftarrow\theta-\alpha\nabla_\theta\mathcal{L} \tag{7}$$

其中，α 为学习率，∇_θ 为梯度算子。JobFusion 的梯度优化算法如算法 1 所示。

算法 1 JobFusion 的梯度优化算法

输入 学习率 α，总训练轮次 N

输出 优化后的策略网络参数 θ

初始化 当前训练轮次 k=1，初始状态信息 $\boldsymbol{s}_1^1$

1) while $k\leqslant N$
2) $t=1$；
3) while 存在未被调度的任务
4) 将 $\boldsymbol{s}_t^k$ 输入策略网络得到每个任务阶段和容器被选择的概率；
5) 将被选择任务阶段和容器组成动作向量 $\boldsymbol{a}_t^k$ 并由强化学习智能体输出；
6) 集群环境接收强化学习智能体输出的 $\boldsymbol{a}_t^k$ 并且输出状态向量 $\boldsymbol{s}_{t+1}^k$ 和奖励分数 r_t^k；
7) $t=t+1$；
8) end while；
9) $b_k=\sum_{i=1}^{k-1}\sum_{j=1}^{T_i}r_j^i$；
10) 使用式(6)计算损失函数并依据式(7)更新策略网络的参数；
11) $k=k+1$；
12) end while

3 实验结果与分析

JobFusion 的实验环境如表 1 所示，其代码实现使用的主要框架是 PyTorch。

表 1 实验环境

硬件	规格参数
CPU	Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20 GHz×2
DRAM	128 GB

JobFusion 所使用的数据集来自阿里云公开数据，含有任务信息描述和机器资源描述。任务信息描述中包含了每时刻在线任务的资源占用情况及离线任务的拓扑关系、资源占用情况、作业数量和作业完成所需时间。机器资源描述中共有 6 000 台类型 1 和 3 000 台类型 2 的物理机用于后续资源划分，具体如表 2 所示。物理机的资源配置远超单个任务的资源需求，因此在进行分析时，还需要考虑资源利用率。在任务调度期间，将物理机划分为不同类型的容器后，任务的不同阶段分配的容器类型可能相同也可能不同，这主要取决于当前强化学习智能体的决策结果。

表 2　　物理机的资源情况

物理机资源类型	CPU 资源/核	内存资源/GB	机器数量/台
类型 1	32	64	6 000
类型 2	92	288	3 000

首先，在每轮调度过程中，即使任务数量众多，JobFusion 也不会同时考虑所有任务，只有那些前驱任务都执行完毕的任务才处于可调度状态，决策时仅将这些可调度任务的嵌入向量输入策略网络中，相比数量较多的任务，每时刻无前驱依赖的任务数量是相对较少的；然后，考虑数量较多的机器，经过资源聚类划分之后，最终用于调度的机器类型也只有 3～8 种，因而基于机器的类型来为任务分配节点这一过程也保证了决策的高效性。总体来说，由于动作空间足够小，JobFusion 对离线任务进行调度编排所花费的时间开销几乎可以忽略不计。单次任务调度时延所占百分比如图 7 所示，平均调度时延仅为 0.704 1 s。

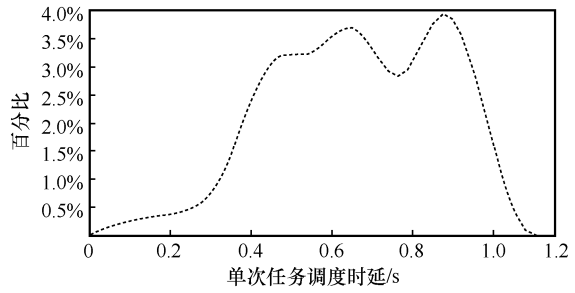

图 7　单次任务调度时延所占百分比

3.1　仿真环境实现细节

为了保证仿真环境尽可能贴近真实集群生产环境，以增强实验结果可重复性、真实性，针对任务完成时间的计算，除了其主体部分的任务所需执行时间外，JobFusion 在模拟集群环境时还考虑了以下因素。

1) 调度器的调度时延。据统计，JobFusion 的平均调度时延约为 0.7 s，因此每次调度时，JobFusion 会从 0.4 到 1.0 之间步长为 0.1 的序列中随机选取一个数作为调度开销（单位为秒）。

2) 容器的初始化时间。当任务从就绪队列进入执行状态前，集群管理器需要为该任务分配 CPU 和内存等机器资源，并进行容器环境的初始化工作。每当一个任务进入执行状态时，JobFusion 都会随机选出一个 1～10 的随机数作为任务的初始化开销（单位为秒）。

3) 资源抢占和上下文切换开销。当在线任务到达而此刻资源不足时，将根据在线任务所需的资源规模随机中止对应容器上所承载的离线任务，以空出资源满足在线任务的运行。这一过程中，首先要保存离线任务当前运行时刻的上下文环境，随后为容器重新挂载目标磁盘，等容器挂载好新磁盘并重启后即可执行任务。这一部分的时间开销主要取决于磁盘读写和挂载的速度，以一般机械硬盘百兆的传输速率为参考，JobFusion 用 5～25 的随机数来模拟这一过程中的时间开销（单位为秒）。

事实上，最终影响任务完成时间的额外因素除了上述的几点外，还包括网络请求处理时延、数据 I/O 时延、并行任务之间的通信开销等。但是仿真实验的目的并不是完美复刻复杂的真实环境，而是能够模拟在真实环境中存在的主要干扰因素，以尽可能还原真实场景下的程序运行结果。上文描述的调度时延、初始化时间、任务资源抢占和切换的时间开销，其数值范围以秒为单位。而对于单个任务，其他因素如并行程序的同步开销，其数值范围大概仅在几微秒到几毫秒之间，相比上述的主要因素，几乎可以忽略，因此本文并未将其考虑在内。最终，JobFusion 计算任务完成时间时不仅考虑了任务本身所需的执行时间和等待时间，还考虑了任务调度时延、容器的初始化时间和上下文切换带来的额外开销。

3.2　实验数据

3.2.1　离线任务资源需求

离线任务的资源需求如图 8 所示，其中直方图单独体现了任务的资源需求分布情况，散点图综合体现了任务的资源需求分布。

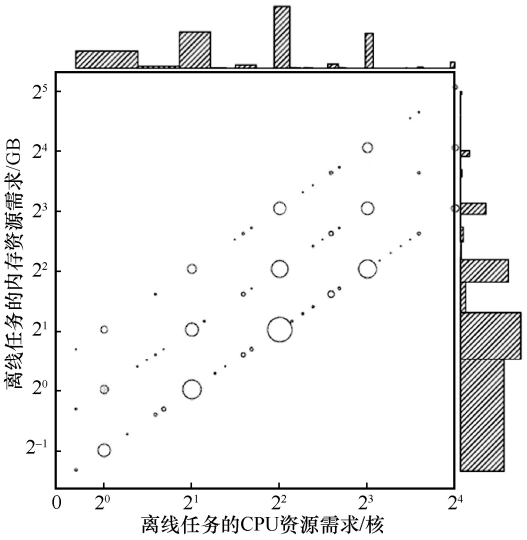

图 8　离线任务的资源需求

3.2.2　在线任务资源需求

在线任务的资源需求如图 9 所示，阿里云数据集的在线任务提供了 98 个时间点共持续 24 h 的 CPU 和内存资源占用情况。由于在线任务实时性要求更高，在 2 种任务混合的情况下，如果在线任务所需资源不足，则需要将占用资源的离线任务进程中止使其重新进入调度队列，以保证在线任务的资源需求。通过分析 2018 国际 AIOps 挑战赛的数据以及在其他生产环境采集的微服务指标数据[15]，工作负载的变化呈现一定的周期性：受潮汐效应的影响，工作负载在一天中的用户访问高峰期会出现波峰，而在用户午休、晚睡时会出现波谷。在用户基数足够大的情况下，这种工作负载的周期性通常会保持稳定。因此本文假设在线任务的每天工作负载变化相同。横坐标给出了时间刻度，纵坐标分为左右 2 个刻度：当前时刻集群中 CPU 总使用量、当前时刻集群中内存总使用量。

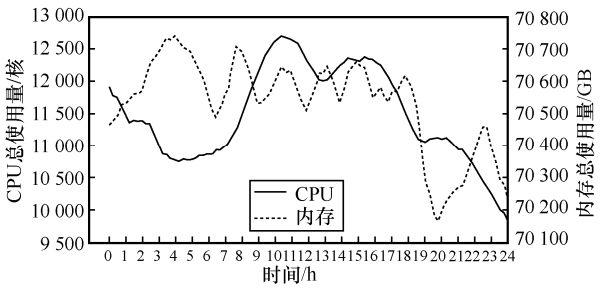

图 9　在线任务的资源需求

3.3　对比实验

3.3.1　超参数设置对调度效率的影响

本文主要的超参数是层次聚类模块中簇的数目 Q。为了展示 Q 对调度效率的影响，在任务负载和其他网络模块一致的情况下，通过对 Q 设置不同的值，最终得到如图 10 所示的结果。其中，线条表示多次实验下任务完成时间的平均值；阴影表示多次实验下任务完成时间的平均值±标准差后的上下界。从图 10 可知，当 Q=6 时，资源聚类划分模块得到的容器集合能够使 JobFusion 取得最优的调度效率，任务平均完成时间为 10.10±0.78 min。

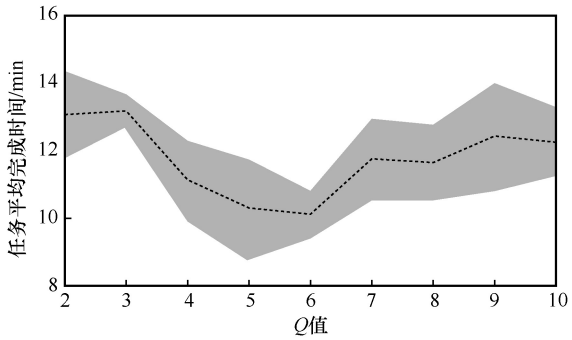

图 10　不同 Q 值对任务平均完成时间的影响

当 Q 值逐渐增大时，会面临决策空间过大的问题，即有更多类型的容器需要被考虑。当 Q 值过小时，将会有较多的资源浪费，同样不利于任务的调度。只有当 Q 的取值适中且能较好地匹配任务的资源需求时，才可以在决策空间大小和任务调度效率上取得比较好的权衡。值得注意的是，Q 的取值是与任务模式相关联的。如图 11 所示，在不同的任务负载和资源总量下，最优 Q 值也各不相同。其中，热力图展示了在任务负载和资源总量同时变化时，最优 Q 值的变化分布情况；顶端的子图给出了在仅变化任务负载时，最优 Q 值的变化分布情况；右侧子图给出了在仅变化资源总量时，最优 Q 值的变化分布情况。

当任务负载较低时，调度开销所占比重较大，由于机器资源充足，较小的 Q 值意味着较小的调度开销；随着任务负载的增加，由于资源总量的限制逐渐成为瓶颈，因此最优 Q 值先逐渐增大后保持不变。

随着资源总量的增加，最优 Q 值呈现先增后减趋势，这是由于当资源总量远大于任务所需资源时，几乎所有任务的资源需求都能同时得到满足，此时进行资源管理并没有太大意义，反而由于动作空间的扩大带来额外的调度开销，降低了调度效率。所以资源总量超出一定范围后，Q 值的减小反而能够提高调度效率。

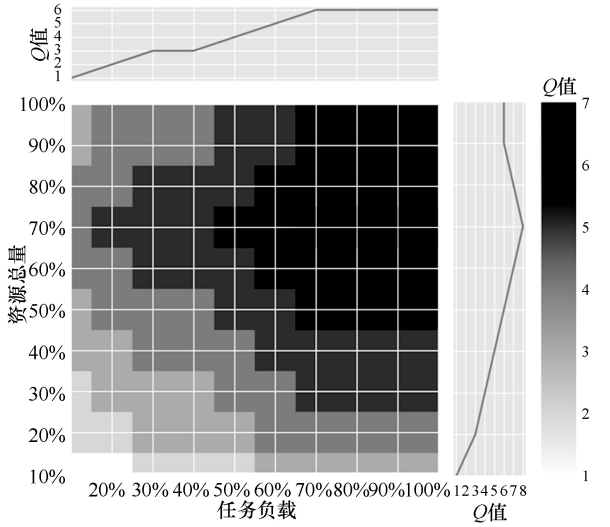

图 11　任务负载和资源总量变化对最优 Q 值的影响

由于 Q 值本质上反映了机器资源与任务负载之间的平衡关系，Q 值的选取与资源总量和任务负载的相对大小紧密相关。如果 2 个数据集的资源总量和任务负载的相对大小接近，那么最优 Q 值可以共享；反之，则需要通过简单实验来快速获取最优 Q 值。

3.3.2　资源聚类模块对调度效率的影响

通过对机器资源进行聚类划分，使具有一定资源需求的任务可以被更高效地调度，同时减少资源类型数量以提高调度效率。为了验证资源划分对 JobFusion 调度效率的影响，本文使用 3 种资源预处理方式进行对比。

方法 1。不进行资源聚类划分。通过分析所有离线任务的资源需求，得到所有任务中各类资源最高的需求，以此为标准划分出统一类型的容器，从而满足所有任务运行的需求。

方法 2。去除连通性约束部分，仅使用传统的层次要素算法进行资源划分。

方法 3。使用上文所介绍的带连通性约束的层次要素算法进行资源划分。

本文统计了不同资源划分方法下机器闲置资源比例与 Q 值的关系，结果如图 12 所示。从图 12 可知，随着 Q 值的增大，方法 2 和方法 3 得益于层次聚类，其对应的闲置资源比例逐步下降，且始终低于方法 1 对应的闲置资源比例；方法 1 由于并未采用聚类方法，其对应的闲置资源比例不受 Q 值变化的影响，保持在较高水平。多次实验结果表明，尽管聚类划分能够一定程度地降低资源闲置比例，但是由于没有连通性约束，资源聚类结果不稳定，这显著增加了任务调度时的不确定性。方法 3 通过添加连通性约束，很好地克服了方法 1 和方法 2 的不足，无论是在资源利用率方面还是在资源划分稳定性方面，方法 3 都取得了最优的性能。

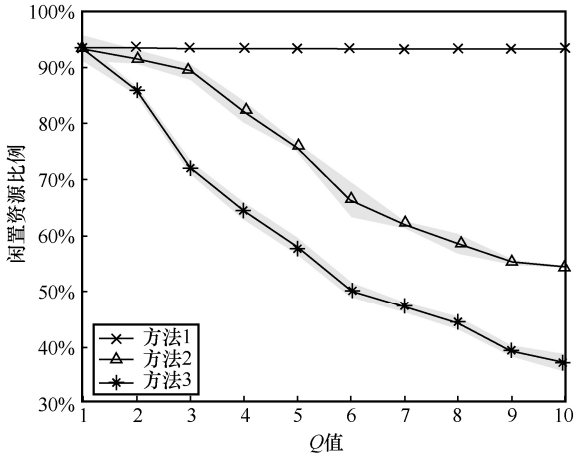

图 12　不同资源划分方法下机器闲置资源比例与 Q 值的关系

使用带连通性约束的层次聚类模块不仅能减少闲置资源，而且能进一步提高调度效率。图 13 给出了不同 Q 值和资源划分方法下任务完成时间的概率密度分布。任务完成时间较小区间内任务数量越多，则表示调度效率越高。方法 1 由于无划分模块，因此其不受 Q 值影响。从图 13 可知，当 Q=1 时，机器资源类别数目仅为 1，因此 3 种方法并无显著差异；当 Q 值较小时，方法 2 和方法 3 对调度效率影响的差距较小；随着 Q 值的增大，两者的差距逐渐拉大。此外，从图 13 中还能发现，当 Q＜6 时，随着 Q 值的增大，方法 2 和方法 3 逐渐提升调度效率；当 Q＞6 时，随着 Q 值的增大，方法 2 和方法 3 对调度效率的提升作用逐渐减弱。上文提到，机器资源类别对调度开销的影响较大，随着 Q 值增大、调度效率提升，调度开销也逐渐增大。当调度开销超过临界值时，Q 值增大所带来的效率提升无法掩盖调度开销带来的负面作用，因此使调度效率出现先增长后回落的现象。

在实际应用中，绝大多数任务对资源的需求相近，但是不同类型资源的需求比重各有不同。考虑这样一种情况：A 类任务有 100 个，其运行条件需要 2 核 CPU 和 4 GB 内存；B 类任务也有 100 个，其执行需要 4 核 CPU 和 2 GB 内存。当层次聚类的过程中没有任何一种类型比 A 在距离度量上更接近 B 时，如果不考虑两者的连通性约束，那么两者将合并为一簇。但合并后的质心不能是 A 也不能是

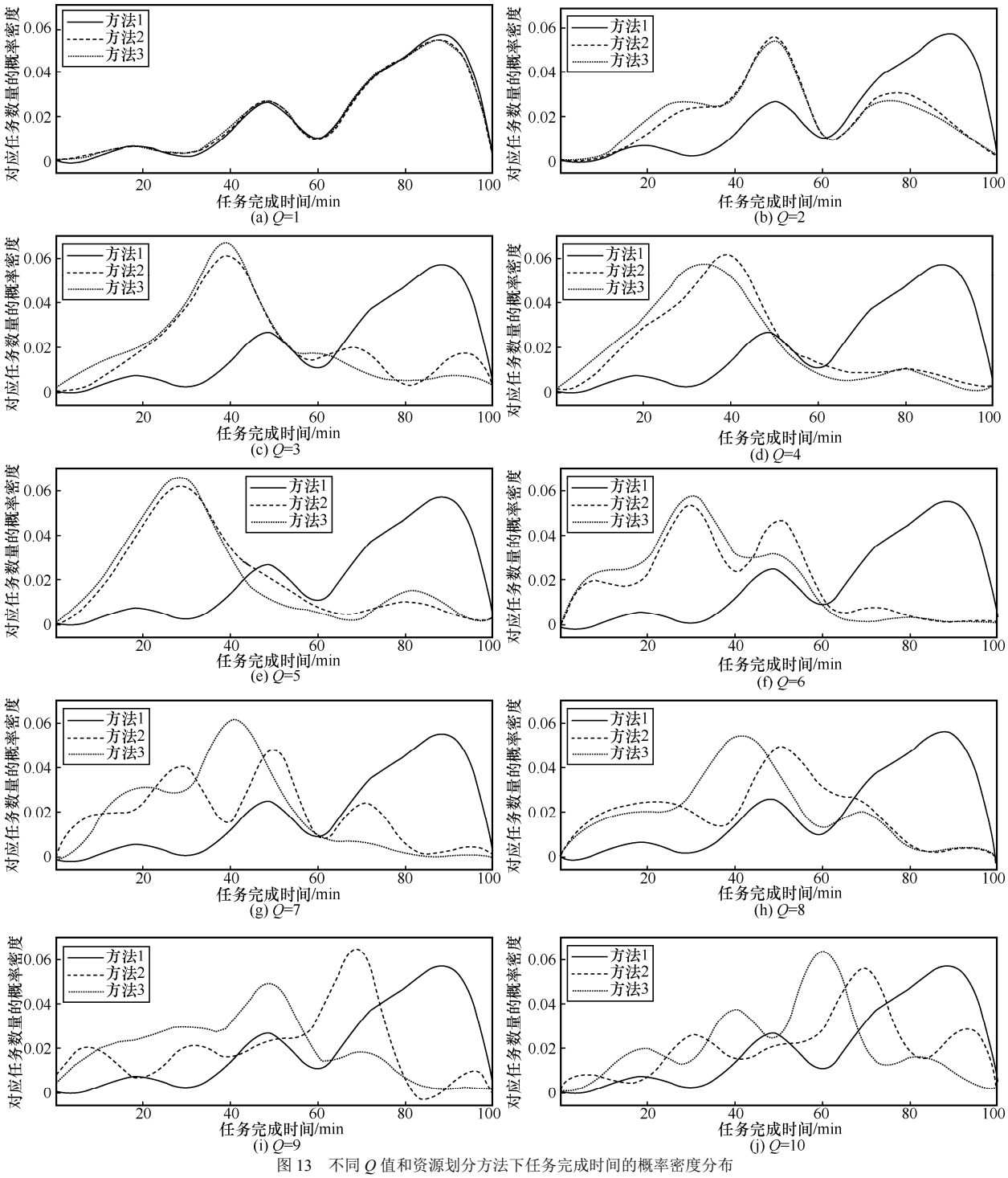

图 13 不同 Q 值和资源划分方法下任务完成时间的概率密度分布

B，否则质心不能同时满足簇中所有元素对应的任务资源需求。如果强行合并两者，必须取两者各个分量中的较大者组成一个新的类，势必会产生资源浪费，这也是引起聚类结果不稳定的因素之一。

将层次聚类与连通性约束结合后，上面描述的情况将极少发生。在不出现极端条件的情况下，整个层次聚类过程中都不会构造出一个新的类型，每次合并簇时都可由已存在的类型作为代表元。因此，相比普通的层次要素算法，基于连通性约束的层次聚类在资源利用率上略优。

3.3.3 调度策略对调度效率的影响

由于 JobFusion 是一个通用调度框架，因此本

文选取如下方法进行对比实验来展示调度策略对调度效率的影响。

1) Decima。使用 GNN 对具有依赖关系的任务属性进行嵌入，并使用强化学习方法在相同类型的机器上对任务进行调度。

2) 最短任务优先策略。根据每个任务阶段完成所需时间赋予优先级，优先调度完成所需时间较短的任务阶段。

3) 随机调度策略。每次调度时，根据依赖关系从可调度的任务中随机选取一个任务进行调度。

不同的任务负载表示只加载特定比例的任务数量，通过多次实验观察，得到不同的任务负载下各个方法的任务平均完成时间对比，如图 14 所示。从图 14 可知，随着任务负载的增大，JobFusion 在调度效率上始终保持一定优势，平均完成时间随着负载增加而缓慢增加，而其他方法的任务平均完成时间迅速增加。这是因为当任务负载增加时，各类任务的资源需求差距逐渐拉开，JobFusion 资源划分模块节省了大量计算资源。而节省出的资源又可以通过虚拟化创建新的机器用于调度，这是 JobFusion 占据优势的重要原因之一。JobFusion 每次决策时都会根据当前状态自动学习一个最优策略进行调度。由于从嵌入到决策输出的整个过程都是可微分的，JobFusion 使用策略梯度优化算法，根据由模型损失求得的梯度信息来更新模型参数。Decima 由于未集成资源划分模块，且未建立多维资源约束的任务调度视角，造成了一定的资源浪费。最短任务优先策略在短任务过多时会造成大任务的堆积，在此情况下无法处理好任务调度工作。随机调度策略可以避免大任务堆积，但是当容器空闲时间较零散时，该策略反而不能通过优先调度小任务来很好地利用机器资源。

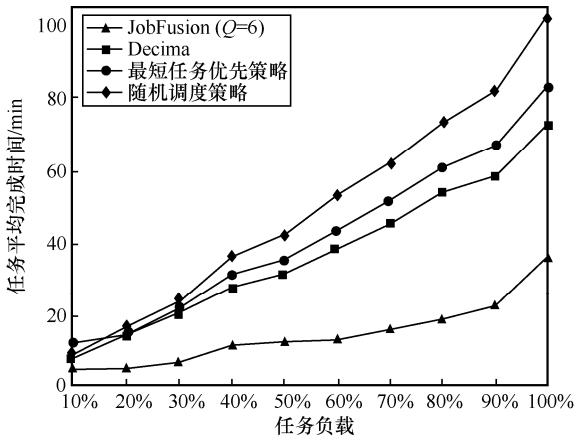

图 14　不同的任务负载下各个方法的任务平均完成时间对比

图 15 给出了在不同机器资源下各个方法的任务平均完成时间。从图 15 可知，当机器资源较小时，资源聚类划分方法的优势较微弱；当机器资源达到一定数量，JobFusion 的优势逐渐凸显，且随着机器资源的增加，其调度效率快速提升。其他方法无资源划分模块，调度效率随机器资源的变化规律并不明显，但总体与机器资源呈负相关趋势。

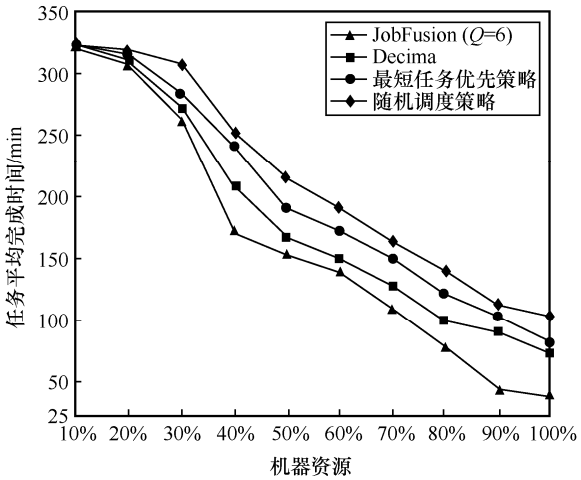

图 15　不同机器资源下各个方法的任务平均完成时间

JobFusion 和 Decima 得益于深度强化学习模块，能够根据集群实时状态动态调整调度策略，因此优于其他仅含单一策略的调度方法。JobFusion 优于 Decima 之处在于：1) 由于 Decima 并未针对云计算平台的资源利用率问题进行优化，因此未集成资源划分模块，资源利用率不高；2) Decima 并未考虑任务的多维资源约束以及容器类型的差异性，其任务嵌入模块未能捕捉任务资源约束层面的嵌入信息。

3.4　实验结果分析

通过对比实验，本文验证了带连通性约束的层次要素算法对资源利用率和调度效率的影响证明了在多维资源约束和在线离线混部场景中JobFusion的调度策略和优化方案的有效性。JobFusion 通过集成资源聚类划分和深度强化学习调度算法这 2 个模块，将资源利用率提高了 39.86%，并降低了 55.14%～64.36%的任务平均完成时间。

4　结束语

为了解决云计算平台调度问题中的资源浪费和效率低下问题，本文提出了 JobFusion 调度框架。首先，通过添加一个资源聚类模块，对物理机资源

实施合理的资源划分；其次，在嵌入阶段，通过使用 GCN 对不定数量的任务和任务阶段之间复杂多变的依赖关系进行编码嵌入；最后，在调度阶段，使用基于深度强化学习的调度方法以及策略梯度算法进行训练和优化。但是随着云计算平台规模的逐渐扩大，任务数量及其资源需求类别急剧膨胀，在使用资源聚类模块考虑连通性约束的同时，可能会遇到稀疏性问题，即在考虑连通性约束下，资源类型依赖图的邻接矩阵为稀疏矩阵。未来还可以针对此类问题提出更高效的图嵌入算法。此外，在 I/O 密集型任务的调度场景中，磁盘读写速率、网络吞吐带宽和链路传输时延等因素也需要考虑在内，JobFusion 的后续工作也可以针对此类场景展开。

虚拟化技术快速发展的同时激发了云计算平台的活力。随着云计算平台规模的不断扩大、硬件水平的快速发展以及摩尔定律的加持，基础设施中的异构和混部问题将愈发突出，因此针对各类混部场景的调度算法将越来越重要。

参考文献：

[1] SRIVASTAVA P, KHAN R. A review paper on cloud computing[J]. International Journal of Advanced Research in Computer Science and Software Engineering, 2018, 8(6): 17.

[2] MAO H Z, SCHWARZKOPF M, VENKATAKRISHNAN S B, et al. Learning scheduling algorithms for data processing clusters[C]//Proceedings of the ACM Special Interest Group on Data Communication. New York: ACM Press, 2019: 270-288.

[3] MNIH V, BADIA A P, MIRZA M, et al. Asynchronous methods for deep reinforcement learning[C]//Proceedings of the 33rd International Conference on International Conference on Machine Learnin. New York: ACM Press, 2016: 1928-1937.

[4] YU H F, IRISSAPPANE A A, WANG H, et al. FaaSRank: learning to schedule functions in serverless platforms[C]//Proceedings of 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS). Piscataway: IEEE Press, 2022: 31-40.

[5] ZHANG Y, ZHU H, TANG D, et al. Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems[J]. Robotics and Computer-Integrated Manufacturing, 2022: doi.org/10.1016/j.rcim.2022.102412.

[6] SONG W, CHEN X Y, LI Q Q, et al. Flexible job-shop scheduling via graph neural network and deep reinforcement learning[J]. IEEE Transactions on Industrial Informatics, 2023, 19(2): 1600-1610.

[7] ZHOU G, WEN R, TIAN W, et al. Deep reinforcement learning-based algorithms selectors for the resource scheduling in hierarchical cloud computing[J]. Journal of Network and Computer Applications, 2022, 208: 103520.

[8] SHENG J, HU Y, ZHOU W, et al. Learning to schedule multi-NUMA virtual machines via reinforcement learning[J]. Pattern Recognition, 2022, 121: 108254.

[9] LI W G, LI J L, WU W G. A dynamic scheduling algorithm with time varying resource constraints in colocation data centers[C]//Proceedings of the 13th International Conference on Information and Communication Systems (ICICS). Piscataway: IEEE Press, 2022: 115-120.

[10] ZHANG X L, LI L Q, WANG Y, et al. Zeus: improving resource efficiency via workload colocation for massive kubernetes clusters[J]. IEEE Access, 2021, 9: 105192-105204.

[11] PI A D, ZHOU X B, XU C Z. Holmes: SMT interference diagnosis and CPU scheduling for job co-location[C]//Proceedings of the 31st International Symposium on High-Performance Parallel and Distributed Computing. New York: ACM Press, 2022: 110-121.

[12] SUNNY M, SEN S J, SABU S E, et al. Deploy-Web hosting using docker container[C]//Advances in Computing and Network Communications. Singapore: Springer, 2021: 335-345.

[13] JAYAPRAKASH S, NAGARAJAN M D, PRADO R P D, et al. A systematic review of energy management strategies for resource allocation in the cloud: clustering, optimization and machine learning[J]. Energies, 2021: doi.org/10.3390/en14175322.

[14] CHHAJED D, LOWE T J. Building intuition: insights from basic operations management models and principles[M]. New York: Springer, 2008.

[15] XU H W, CHEN W X, ZHAO N W, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal KPIs in web applications[C]//Proceedings of the 2018 World Wide Web Conference. New York: ACM Press, 2018: 187-196.

[作者简介]

马玲（1985− ），女，吉林洮南人，博士，南开大学副教授，主要研究方向为人工智能。

樊漆亮（1999− ），男，江西南昌人，南开大学硕士生，主要研究方向为时间序列异常检测。

许婷（1991− ），女，河南正阳人，南开大学博士生，主要研究方向为异常检测、故障定位、根因分析和故障预测等。

郭冠琛（2000− ），女，山东淄博人，北京大学硕士生，主要研究方向为异常检测、故障定位、根因分析和故障预测等。

张圣林（1989− ），男，山东滨州人，博士，南开大学副教授，主要研究方向为异常检测、故障定位、根因分析和故障预测等。

孙永谦（1988− ），男，河北石家庄人，博士，南开大学助理教授，主要研究方向为异常检测、故障定位、根因分析和故障预测等。

张玉志（1964− ），男，河北邢台人，博士，南开大学讲席教授、博士生导师，主要研究方向为人工智能和软件工程。