(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 115827161 A
(43)申请公布日 2023.03.21

(21)申请号 202211581622.9

(22)申请日 2022.12.07

(71)申请人 北京航空航天大学
地址 100191 北京市海淀区学院路37号

(72)发明人 吴文峻 田雍恺 冯埔

(74)专利代理机构 北京永创新实专利事务所 11121
专利代理师 易卜

(51)Int.Cl.
G06F 9/455(2018.01)
G06N 3/092(2023.01)
G06F 9/50(2006.01)

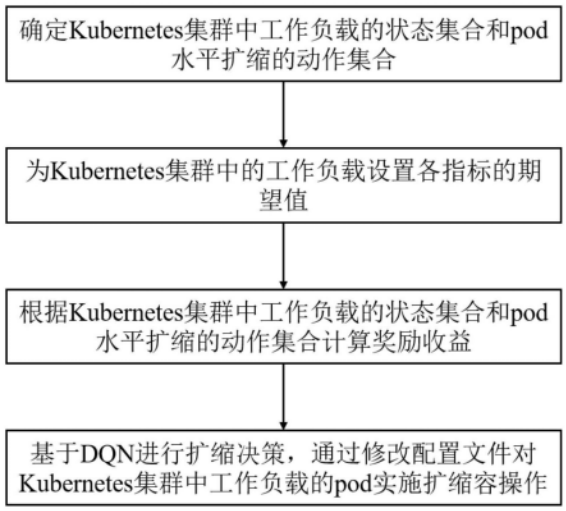
权利要求书2页 说明书7页 附图3页

(54)发明名称

一种基于强化学习的pod水平扩缩算法

(57)摘要

本发明公开了一种基于强化学习的pod水平扩缩算法,属于微服务资源调度领域,具体为:首先,针对Kubernetes集群中每个工作负载,确定其占用的pod的繁忙程度的表征状态指标,并设定pod水平扩缩的动作集合;然后,为各工作负载的状态指标分别设置各自的上下限;根据工作负载的状态集合和pod水平扩缩的动作集合计算奖励收益;最后,基于DQN进行扩缩决策,通过修改配置文件对Kubernetes集群中工作负载的pod实施扩缩容操作。本发明将工作负载当前的多项指标联合表征为强化学习的状态输入,由训练好的强化学习智能体提供智能决策,以实现工作负载所占有pod数量水平扩缩的按需调整、智能调整,从而提升pod扩缩效率,优化相应工作负载的QoS指标,提高集群资源利用率,降低服务成本。



CN 115827161 A

1.一种基于强化学习的pod水平扩缩算法,其特征在于,具体步骤包括:

首先、针对Kubernetes集群中每个工作负载,确定其占用的pod的繁忙程度的表征状态指标,并设定pod水平扩缩的动作集合;

然后、为Kubemetes集群中的各工作负载的状态指标分别设置各自的上限和下限值;

接着、根据Kubernetes集群中工作负载的状态集合和pod水平扩缩的动作集合计算奖励收益;

最后、基于强化学习DQN网络进行扩缩决策,通过修改配置文件对Kubernetes集群中工作负载的pod实施扩缩容操作。

基于DQN学习的pod水平扩缩算法步骤如下:

步骤1:初始化经验回放池D,动作价值函数Q的权重θ,目标动作价值函数$\hat{Q}$的权重θ^-,开始训练的阈值n,每次用于训练的样本数量batch,需要训练的episode轮次M,当前时间步t=1以及单个episode时间步上限Step,初始化训练步数train_step=0;

设置经验回放池D的容量为N,N>n>batch;权重θ随机初始化,并使$\theta^-=\theta$;

步骤2:初始化工作负载的状态s_t,预处理得到状态s_t对应的特征输入$\emptyset_t$,采用ε-greedy策略在状态s_t处选择pod扩缩容策略a_t;

步骤3:执行动作a_t,获取奖励收益r_t以及扩缩后工作负载状态s_{t+1};

步骤4:令$s_t=s_{t+1}$,获得下一时刻状态的预处理$\emptyset_{t+1}=\emptyset(s_{t+1})$;

步骤5:将转换$(\emptyset_t,a_t,r_t,\emptyset_{t+1})$存储在经验回放池D中,并更新t=t+1;

步骤6:判断经验回放池D容量是否小于阈值n,如果是,跳至步骤10,否则执行7;

步骤7:从经验回放池D中均匀随机采样batch个转换样本数据;

步骤8:执行梯度下降算法,更新动作价值函数逼近的网络参数θ=θ+Δθ;并同时更新train_step=train_step+1;

其中,$\Delta\theta=\alpha[R+\gamma max_{a'}\hat{Q}(s',a';\theta^-)-Q(s,a;\theta)]\nabla Q(s,a;\theta)$;

学习因子α∈(0,1],折扣因子γ∈(0,1),贪心策略系数ε。R为奖励$max_{a'}$选择的a′使得$\hat{Q}(s',a';\theta^-)$最大,$\nabla Q(s,a;\theta)$为Q函数对参数θ的梯度;

步骤9:判断是否满足train_step%C=0,如果是,则更新目标动作价值函数$\hat{Q}$的权重$\theta^-=\theta$,进入步骤10;否则,直接进入步骤10;

C为固定常数;

步骤10:判断是否t≤Step,如果是,返回步骤2;否则,令t=0,返回步骤2,直至M次episode循环完成,得到最终的权重θ;

步骤11:将当前工作负载的状态输入最终权重θ对应的动作价值函数Q中,得到调节pod实施扩缩容的动作。

2.如权利要求1所述的一种基于强化学习的pod水平扩缩算法,其特征在于,所述每个工作负载的表征状态指标均包括CPU利用率、内存利用率以及平均http请求数;

动作包括增加pod、减少pod以及维持不变三类,其中增加pod数量的动作有5个,分别为增加1～5个pod;减少pod数量的动作也包含5个,分别为减少1～5个pod。

3.如权利要求1所述的一种基于强化学习的pod水平扩缩算法,其特征在于,所述奖励函数采用线性加和的方式获得:reward=$r_1+r_2+r_3+r_4$;

从工作负载的QoS指标出发，QoS指标为平均响应时间t，奖励函数如下：$r_1=-t/10$；

从工作负载的CPU利用率出发，奖励函数如下：$r_2=\begin{cases}0, & cpu_{ratio}\le cpu_{limit}\\ cpu_{limit}-cpu_{ratio}, & \text{其他}\end{cases}$；

其中，cpu_{ratio}表示当前状态下工作负载的CPU利用率，cpu_{limit}表示人工设置的工作负载的CPU利用率期望值；

从工作负载的内存利用率出发，奖励函数如下：$r_3=\begin{cases}0, & mem_{ratio}\le mem_{limit}\\ mem_{limit}-mem_{ratio}, & \text{其他}\end{cases}$；

其中mem_{ratio}表示当前状态下工作负载的内存利用率，mem_{limit}表示人工设置的工作负载的内存利用率期望值；

从工作负载的当前的pod数量出发，奖励函数如下：$r_4=-\frac{num_{cur}}{num_{\max}}$；

其中，num_{cur}表示当前的pod数量，num_{max}表示人工设置的pod数量上限。

一种基于强化学习的pod水平扩缩算法

技术领域

[0001] 本发明属于微服务资源调度领域，具体涉及一种基于强化学习的pod水平扩缩算法。

背景技术

[0002] 近年来，随着云计算和边缘计算的快速发展，虚拟化技术已经成为学术研究和工业开发的重要话题。其中新型的虚拟化技术是容器化技术，在该技术中，配备随时可部署应用程序组件的轻量级操作系统被打包到一个自给自足的容器中，以便在支持多租户的主机上运行。在主机系统中，不同的容器在同一操作系统和同一内核中一起运行，这有助于降低存储需求，并使它们与主机操作系统相比，实现接近本机的性能。

[0003] 由于容器可以大规模部署，因此非常需要在部署、扩展和管理方面都高度自动化的容器编排平台，在各种编排平台中，Kubernetes已经成为行业标准。它是一个开源平台，在这个平台上可以轻松打包和运行容器化的应用程序、工作负载和服务。Kubernetes使用Docker作为基本环境来运行可移植且自给自足的容器，其实例化称为Docker镜像，提供了一个控制平台，根据可用资源和每个容器的特定需求，管理和调度这些容器在其主机集群上运行。在Kubernetes中，最重要的特性之一是自动扩缩，因为它允许容器化的应用程序和服务在不需要人工干预的情况下灵活地运行。

[0004] Pod是Kubernetes中最小的资源管理组件，也是最小化运行容器化应用的资源对象。在Kubernetes中，Horizontal Pod Autoscaler(HPA)是一个强大的功能，可以自动增加pod的数量，从而提高应用程序的整体计算和处理能力，而不必停止应用程序当前运行的实例。一旦成功创建，这些新的pod就能够与现有pod共享传入负载。在配置HPA时需要指定所监测的指标，并为其设定使用量阈值或利用率阈值，另外还需要指定集群中运行的pod数量的最大值与最小值。默认情况下，每隔15秒控制管理器会将收集到的指标与HPA配置中指定的阈值进行比较，一旦所收集到的指标超过设定的阈值，HPA会自动增加pod的数量；

[0005] HPA可以使用多个指标，每一个都有其阈值，当其中的任何一个值到达阈值时，HPA就会以上述方式进行扩展，但对于缩小操作，必须所有的指标值都小于阈值。

[0006] 现存的这种pod水平自动扩缩方式是基于设定阈值的方式，如CPU利用率等，存在如下问题：

[0007] 1、扩缩的阈值参数的设定通常需要对应用程序的领域知识以及其行为具有深入的理解，因此设定一个合适的阈值比较困难。

[0008] 2、基于阈值的方式是一种十分被动的扩缩方式。默认情况下，pod是一个个进行扩容的，这种基于阈值的方式并不能很好地满足延迟敏感的应用程序的QoS指标要求，对于访问量波动较大的服务，当访问量剧增时这种方式的调节效果就会很差，性能急剧下降；当访问量降低时，并不能及时的释放已经不再占用的资源，会出现较为严重的资源浪费。

[0009] 3、参考指标单一化，虽然HPA可以指定多个指标，但是其扩缩策略并不会将多个指标融合在一起考虑，当有某一指标超出阈值限制即采取扩容策略，当全部指标低于阈值时，

即采取缩容策略，多指标的扩缩容本质上依旧是针对单一因素的一种扩缩容。

[0010] 因此，需要更加智能的针对Kubernetes中工作负载的pod数量进行扩缩调整。

发明内容

[0011] 本发明针对上述问题，提供了一种基于强化学习的pod水平扩缩算法，可以智能地调整Kubernetes集群中工作负载的pod数量，不完全依赖于人工根据经验设置的阈值，可以实现性能更好、资源利用率更高的pod水平扩缩。

[0012] 所述基于强化学习的pod水平扩缩算法，包括以下步骤：

[0013] 步骤一、针对Kubernetes集群中每个工作负载，确定其占用的pod的繁忙程度的表征状态指标，并设定pod水平扩缩的动作集合；

[0014] 每个工作负载的表征状态指标均包括CPU利用率、内存利用率以及平均http请求数。

[0015] 动作包括增加pod、减少pod以及维持不变三类。增加pod数量的动作有5个，分别为增加1～5个pod；减少pod数量的动作也包含5个，分别为减少1～5个pod。

[0016] 步骤二、为Kubernetes集群中的各工作负载的状态指标分别设置各自的上限和下限值；

[0017] 步骤三、根据Kubernetes集群中工作负载的状态集合和pod水平扩缩的动作集合计算奖励收益；

[0018] 从工作负载的QoS指标出发，QoS指标为平均响应时间t，奖励函数如下：$r_1=-t/10$。

[0019] 从工作负载的CPU利用率出发，奖励函数如下：$$r_2=\begin{cases}0, & cpu_{ratio}\le cpu_{limit}\\ cpu_{limit}-cpu_{ratio}, & \text{其他}\end{cases}$$

[0020] 其中，cpu_{ratio}表示当前状态下工作负载的CPU利用率，cpu_{limit}表示人工设置的工作负载的CPU利用率期望值。

[0021] 从工作负载的内存利用率出发，奖励函数如下：

$$r_3=\begin{cases}0, & mem_{ratio}\le mem_{limit}\\ mem_{limit}-mem_{ratio}, & \text{其他}\end{cases}$$

[0022] 其中mem_{ratio}表示当前状态下工作负载的内存利用率，mem_{limit}表示人工设置的工作负载的内存利用率期望值。

[0023] 从工作负载的当前的pod数量出发，奖励函数如下：$r_4=-\frac{num_{cur}}{num_{\max}}$。

[0024] 其中，num_{cur}表示当前的pod数量，num_{max}表示人工设置的pod数量上限。

[0025] 最终奖励函数采用线性加和的方式获得：$reward=r_1+r_2+r_3+r_4$。

[0026] 步骤四、基于强化学习DQN网络进行扩缩决策，通过修改配置文件对Kubernetes集群中工作负载的pod实施扩缩容操作。

[0027] 基于DQN学习的pod水平扩缩算法步骤如下：

[0028] 步骤301：初始化经验回放池D，动作价值函数Q的权重θ，目标动作价值函数$\hat{Q}$的权重θ^-，开始训练的阈值n，每次用于训练的样本数量batch，需要训练的episode轮次M，当前时间步t＝1以及单个episode时间步上限Step，初始化训练步数train_step＝0；

[0029] 设置经验回放池D的容量为N(N>n>batch);权重θ随机初始化,并使$\theta^{-}=\theta$;

[0030] 步骤302:初始化工作负载的状态s_t,预处理得到状态s_t对应的特征输入$\emptyset_t$,采用ε-greedy策略在状态s_t处选择pod扩缩容策略a_t;

[0031] 步骤303:执行动作a_t,获取奖励收益r_t以及扩缩后工作负载状态st_{+1};

[0032] 步骤304:令$s_t=s_{t+1}$,获得下一时刻状态的预处理$\emptyset_{t+1}=\emptyset(s_{t+1})$;

[0033] 步骤305:将转换$(\emptyset_t,a_t,r_t,\emptyset_{t+1})$存储在经验回放池D中,并更新t=t+1;

[0034] 步骤306:判断经验回放池D容量是否小于阈值n,如果是,跳至步骤310,否则执行307;

[0035] 步骤307:从经验回放池D中均匀随机采样batch个转换样本数据;

[0036] 步骤308:执行梯度下降算法,更新动作价值函数逼近的网络参数$\theta=\theta+\Delta\theta$;并同时更新train_step=train_step+1;

[0037] 其中,$\Delta\theta=\alpha[R+\gamma max_{a'}\hat{Q}(s',a';\theta^{-})-Q(s,a;\theta)]\nabla Q(s,a;\theta)$;

[0038] 学习因子$\alpha\in(0,1]$,折扣因子$\gamma\in(0,1)$,贪心策略系数ε。R为奖励$max_{a'}$选择的a′使得$\hat{Q}(s',a';\theta^{-})$最大,$\nabla Q(s,a;\theta)$为Q函数对参数θ的梯度。

[0039] 步骤309:判断是否满足train_step%C=0,如果是,则更新目标动作价值函数$\hat{Q}$的权重$\theta^{-}=\theta$,进入步骤310;否则,直接进入步骤310;

[0040] C为固定常数;

[0041] 步骤310:判断是否t≤Step,如果是,返回步骤302;否则,令t=0,返回步骤302,直至M次episode循环完成,得到最终的权重θ;

[0042] 步骤311:将当前工作负载的状态输入最终权重θ对应的动作价值函数Q中,得到调节pod实施扩缩容的动作。

[0043] 本发明的优点在于:

[0044] 1)一种基于强化学习的pod水平扩缩算法,基于强化学习的Kubernetes集群中工作负载状态由CPU利用率、内存利用率以及平均http访问请求数联合表征,不仅有效的描述了集群中工作负载当前的负载特征,而且能够很好的用于评估水平扩缩动作的效果,有利于提高扩缩策略的有效性,解决了传统HPA评估指标单一化的问题。

[0045] 2)一种基于强化学习的pod水平扩缩算法,水平扩缩动作集合包含11种动作,分别为增加1～5个pod、减少1～5个pod以及保持不变,能够有效地提高扩缩决策的自由度,解决传统HPA一个一个增加pod的问题,有利于提高水平扩缩策略的灵活性。

[0046] 3)一种基于强化学习的pod水平扩缩算法,利用深度强化学习算法,将工作负载当前的多项指标联合表征为强化学习的状态输入,由训练好的强化学习智能体提供智能决策,以实现Kubernetes集群中工作负载所占有pod数量水平扩缩的按需调整、智能调整,从而提升pod扩缩效率,优化相应工作负载的QoS指标,提高集群资源利用率,降低服务成本,解决了传统HPA相对静态、调整被动,需要有大量领域内先验知识来手动调整阈值,难以真正智能配置的问题。

附图说明

[0047] 图1为本发明强化学习基本原理框架图;

[0048] 图2为本发明一种基于强化学习的pod水平扩缩算法的原理图；

[0049] 图3为本发明一种基于强化学习的pod水平扩缩算法的流程图；

[0050] 图4为本发明一种基于DQN学习的pod水平扩缩算法流程图；

[0051] 图5为本发明一种基于强化学习的pod水平扩缩管理系统的系统架构图。

具体实施方式

[0052] 下面将结合附图和实施例对本发明所述方案作进一步描述说明。此处所描述的具体实施方案仅用于说明和解释本发明，并不用于限制本发明。

[0053] 如图1所示，强化学习是机器学习的范式和方法论之一，描述和解决智能体在与环境交互过程中，通过不断地试错、探索，学习得到最优策略达成奖励最大化或实现特定目标的问题。

[0054] 如图2所示，通过将Kubernetes集群中工作负载的pod扩缩，建模为强化学习的马尔科夫决策过程，其中工作负载对应环境，通过Kubernetes Client API确定状态集合，根据工作负载的状态计算奖励，并基于DQN进行扩缩决策，进一步根据扩缩决策实施扩缩动作，优化目标工作负载的QoS指标以及资源利用率。从强化学习角度解决云中应用程序自动扩缩的主要动机为：策略是透明的，即策略不依赖于人类干预或深层领域知识；策略是动态的，即学习的策略根据环境的当前状态和应用程序的执行来决定最适当的动作，而不是静态的策略；策略具有适应性，学习到的策略可以适应动态云环境中的变化。

[0055] 所述基于强化学习的pod水平扩缩算法，如图3所示，包括以下步骤：

[0056] 步骤一、确定Kubernetes集群中工作负载的状态集合和pod水平扩缩的动作集合；

[0057] 所述Kubernetes集群中工作负载的状态包括CPU利用率、内存利用率以及平均http请求数，这三项指标都可以明确的反应当前该工作负载的繁忙程度。

[0058] 当该工作负载所占用的pod数量减少时，其平均http请求数会相对上升，意味着该工作负载的每个pod将会有更多的任务需要处理，因此，其所占用pod的CPU利用率与内存利用率也会相应上升；当该工作负载所占用pod数量上升时，其平均http请求数会相应下降，该工作负载所占用pod的任务量也将会随之降低，随之发生变化的是该工作负载所占用pod的CPU利用率以及内存利用率也会降低。

[0059] 因此，CPU利用率、内存利用率以及平均http请求数这三项指标的变化趋势可以很好的反应扩缩动作的效果。选用CPU利用率、内存利用率以及平均http请求数的组合来表征强化学习中观测到的工作负载状态，不仅能有效描述工作负载当前的任务负载情况，而且能很好地用于评估扩缩决策的优劣，有利于提高扩缩决策的有效性。

[0060] 所述pod水平扩缩动作集合包括增加pod、减少pod以及维持不变三类。增加pod数量的动作有5个，分别为增加1～5个pod；减少pod数量的动作也包含5个，分别为减少1～5个pod。扩缩动作集合共包含11种不同的动作。增加与减少动作均包含5种不同的选择使得扩缩决策选择性更大，有利于提升扩缩决策的灵活性。例如，当发生激增式访问增长时，可以直接选择增加5个pod来提升工作负载的工作性能，无需一个一个依次增加；当访问量变化缓慢时，可以选择每次扩缩较少数量的pod进行调节。

[0061] 步骤二、为Kubernetes集群中的各工作负载指标分别设置各自的期望值；

[0062] 所述工作负载的指标包括CPU利用率、内存利用率、平均http请求数以及该工作负

载可占有的pod数量下限与上限。

[0063] 每个集群中的资源是有上限的，不可能无限制的增长，因此需要对工作负载设置相应的CPU、内存、访问数以及可占用pod数量的上限，防止某一工作负载无限制占用集群资源。某些工作负载维持基本运行可能就需要一定量的资源，为了维持工作负载运行稳定，需要对其所占用的pod数量设置下限。另外，设置这些指标可用于强化学习扩缩策略学习过程中奖励函数的计算。

[0064] 步骤三、根据Kubernetes集群中工作负载的状态集合和pod水平扩缩的动作集合计算奖励收益；

[0065] 强化学习的目的是学习一个使得累计折扣奖励最大的最优策略，奖励函数能否准确的估计出真实的反馈，直接决定了强化学习智能体能否学到有用的信息。对pod进行扩缩的主要目的是提升工作负载的工作性能以及资源利用率，从而提升其QoS指标(平均响应时间)，因此工作负载的CPU利用率、内存利用率、平均响应时间以及所占用pod数量可以用来计算扩缩动作的奖励。

[0066] 从观测到的工作负载的QoS指标出发，所选择的工作负载的QoS指标为平均响应时间，记为t。根据工作负载的平均响应时间定义相关的奖励函数如下：

[0067] $r_1 = -t/10$

[0068] 由于平均响应时间应越短越好，因此该项奖励置为负数，当存在较大任务量时，提升pod应获得较低的平均响应时间，这是本发明一项基于强化学习的pod水平扩缩算法的主要奖励部分。

[0069] 进一步的，从观测到的工作负载的CPU利用率出发，定义相关的奖励函数如下：

[0070] $$r_2 = \begin{cases} 0, & cpu_{ratio} \le cpu_{limit} \\ cpu_{limit} - cpu_{ratio}, & \text{其他} \end{cases}$$

[0071] 其中，cpu_{ratio}表示当前状态下工作负载的CPU利用率，cpu_{limit}表示人工设置的工作负载的CPU利用率期望值。

[0072] 当任务量较大时，CPU利用率较高，若未超过人工设定的期望值，奖励为0；若超过了人工设定的期望值，理应给予负奖励，使智能体选择增加pod的动作来降低CPU利用率。

[0073] 进一步的，从观测到的工作负载的内存利用率出发，定义相关的奖励函数如下：

[0074] $$r_3 = \begin{cases} 0, & mem_{ratio} \le mem_{limit} \\ mem_{limit} - mem_{ratio}, & \text{其他} \end{cases}$$

[0075] 其中mem_{ratio}表示当前状态下工作负载的内存利用率，mem_{limit}表示人工设置的工作负载的内存利用率期望值。

[0076] 当任务量较大时，内存利用率较高，若未超过人工设定的期望值，奖励为0；若超过了人工设定的期望值，理应给予负奖励，使智能体选择增加pod的动作来降低内存利用率。

[0077] 进一步的，从观测到的工作负载的当前的pod数量出发，定义相关的奖励函数如下：

[0078] $$r_4 = -\frac{num_{cur}}{num_{\max}}$$

[0079] 其中，num_{cur}表示当前的pod数量，num_{max}表示人工设置的pod数量上限。

[0080] 为防止智能体一味对工作负载的pod进行扩容，用当前pod数量占最大pod数量的比例给予负奖励。

[0081] 最终奖励函数采用线性加和的方式获得:reward＝r_1+r_2+r_3+r_4。

[0082] 步骤四、基于Deep Q-Network(DQN)进行扩缩决策，通过修改配置文件对Kubernetes集群中工作负载的pod实施扩缩容操作。

[0083] 如图4所示，基于DQN学习的pod水平扩缩算法步骤如下：

[0084] 步骤301:初始化经验回放池D，动作价值函数Q的权重θ，目标动作价值函数$\hat{Q}$的权重θ^-，开始训练的阈值n，每次用于训练的样本数量batch，需要训练的episode轮次M，当前时间步t＝1以及单个episode时间步上限Step，初始化训练步数train_step＝0；

[0085] 设置经验回放池D的容量为N(N>n>batch)；权重θ随机初始化，并使θ^-＝θ；

[0086] 步骤302:初始化工作负载的状态s_t，预处理得到状态s_t对应的特征输入$\emptyset_t$，采用ε-greedy策略在状态s_t处选择pod扩缩容策略a_t；

[0087] DQN算法中在工作负载的每一种状态下，将以ε的概率从pod水平扩缩动作结合中随机选择一个作为下一步的扩缩动作，以1-ε的概率选择对应于该状态pod水平扩缩动作集合中动作价值函数最高的动作作为下一步的扩缩动作。

[0088] 步骤303:执行动作a_t，获取奖励收益r_t以及扩缩后工作负载状态s_{t+1}；

[0089] 步骤304:令s_t＝s_{t+1}，获得下一时刻状态的预处理$\emptyset_{t+1} = \emptyset(s_{t+1})$；

[0090] 步骤305:将转换$(\emptyset_t, a_t, r_t, \emptyset_{t+1})$存储在经验回放池D中，并更新t＝t+1；

[0091] 步骤306:判断经验回放池D容量是否小于阈值n，如果是，跳至步骤310，否则执行307；

[0092] 步骤307:从经验回放池D中均匀随机采样batch个转换样本数据；

[0093] 步骤308:执行梯度下降算法，更新动作价值函数逼近的网络参数θ＝θ+Δθ；并同时更新train_step＝train_step+1；

[0094] 其中，$\Delta\theta = \alpha[R + \gamma max_{a'}\hat{Q}(s', a'; \theta^-) - Q(s, a; \theta)]\nabla Q(s, a; \theta)$；

[0095] 学习因子α∈(0,1]，折扣因子γ∈(0,1)，贪心策略系数ε。R为奖励max_a，选择的a′使得$\hat{Q}(s', a'; \theta^-)$最大，$\nabla Q(s, a; \theta)$为Q函数对参数θ的梯度。

[0096] 步骤309:判断是否满足train_step%C＝0，如果是，则更新目标动作价值函数$\hat{Q}$的权重θ^-＝θ，进入步骤310；否则，直接进入步骤310；

[0097] C为固定常数；

[0098] 步骤310:判断是否t≤Step，如果是，返回步骤302；否则，令t＝0，返回步骤302，直至M次episode循环完成，得到最终的权重θ；

[0099] 步骤311:将当前工作负载的状态输入最终权重θ对应的动作价值函数Q中，得到调节pod实施扩缩容的动作。

[0100] 如图5所示，本发明提出了一种基于强化学习的pod水平扩缩管理工具，包括:前端组件，包括管理模块、创建模块以及展示模块。

[0101] 管理模块用于查看、编辑、删除已创建的所有扩缩器，可通过命名空间与工作负载的名称筛选已创建的自动扩缩器，并可以编辑每个扩缩器的指标值以及跳转至展示界面；创建模块用于创建新的扩缩器，需要设置扩缩器名称、选择命名空间、选择相应工作负载并

设置各个相关指标;展示模块用于展示扩缩器对应的集群中工作负载的资源使用情况,包括CPU利用率、内存利用率、平均http请求数以及pod数量的变化趋势。

[0102] 服务端组件,包括命名空间查询服务、工作负载查询服务、扩缩器创建/删除服务以及负载数据收集服务。

[0103] 基于轻量级后端框架flask框架开发,主要通过python的Kubernetes Client API来获取Kubernetes中的命名空间、工作负载等集群相关信息,以及相应工作负载对应的负载指标信息,包括CPU利用率、内存利用率、平均http访问请求数以及当前pod数量。

[0104] 数据库组件,主要负责存储创建的扩缩器。

[0105] 数据库用于存储已创建的扩缩器的名称、对应工作负载所在的命名空间与名称,服务端在创建或删除扩缩器时同时执行对数据库的操作。

[0106] 算法组件,用于存放基于强化学习的pod水平扩缩算法Docker镜像。

[0107] 基于强化学习的pod水平扩缩算法被打包为Docker镜像,在每次创建新的扩缩器时,服务端会拉取相应镜像,在Kubernetes集群中创建一个新的服务,该服务读取相应的工作负载各个指标对其pod数量进行扩缩。

[0108] 上述内容为本发明一个具体实施例的描述,上述实施例是示例性的,不作为对本发明的限制。

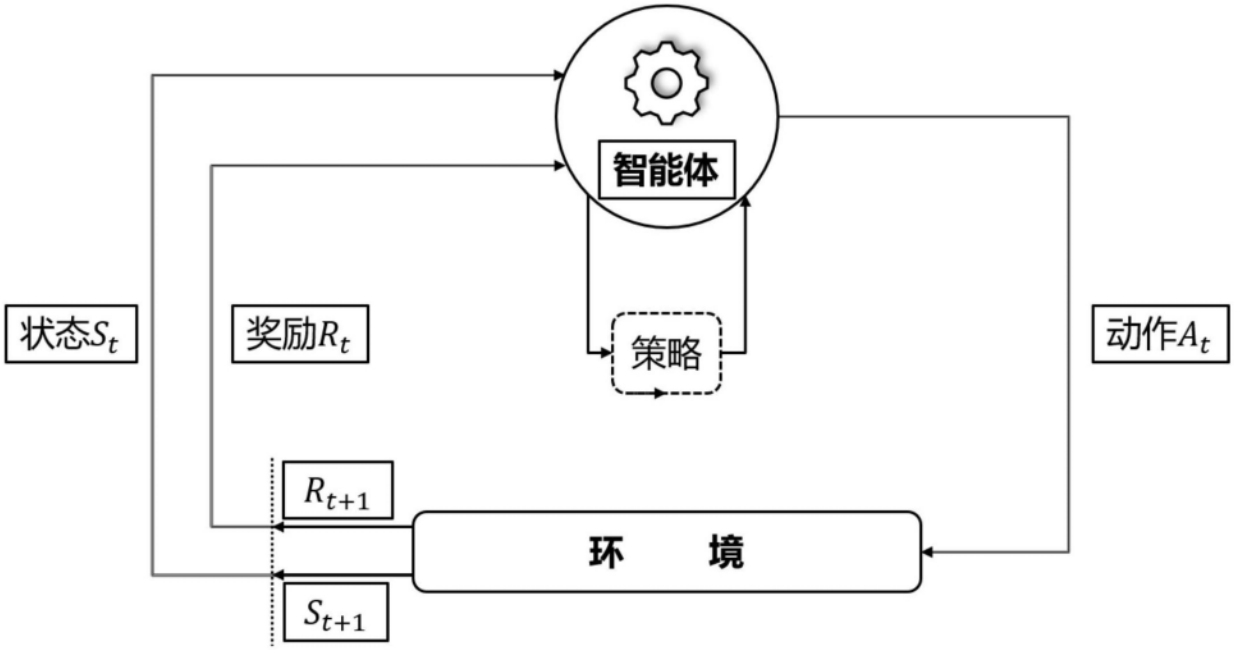

智能体
状态S_t
奖励R_t
策略
动作A_t
R_{t+1}
环 境
S_{t+1}

图1

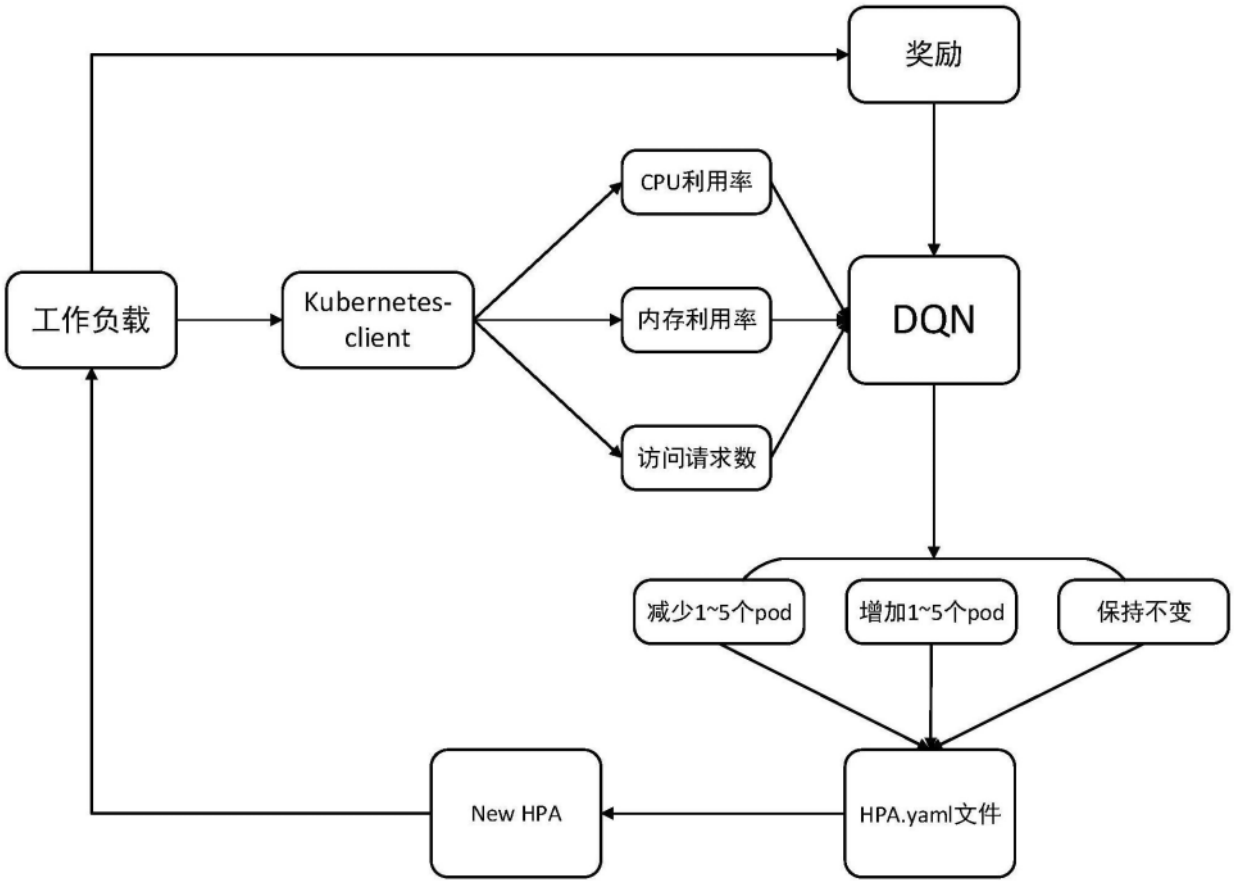

奖励
CPU利用率
工作负载
Kubernetes-client
内存利用率
DQN
访问请求数
减少1~5个pod
增加1~5个pod
保持不变
New HPA
HPA.yaml文件

图2

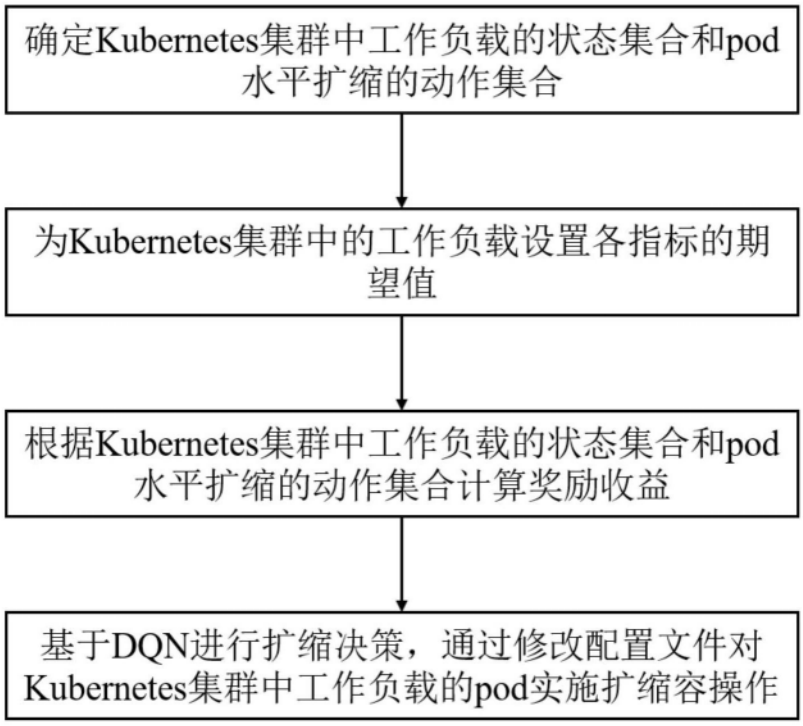

确定Kubernetes集群中工作负载的状态集合和pod水平扩缩的动作集合
为Kubernetes集群中的工作负载设置各指标的期望值
根据Kubernetes集群中工作负载的状态集合和pod水平扩缩的动作集合计算奖励收益
基于DQN进行扩缩决策，通过修改配置文件对Kubernetes集群中工作负载的pod实施扩缩容操作

图3

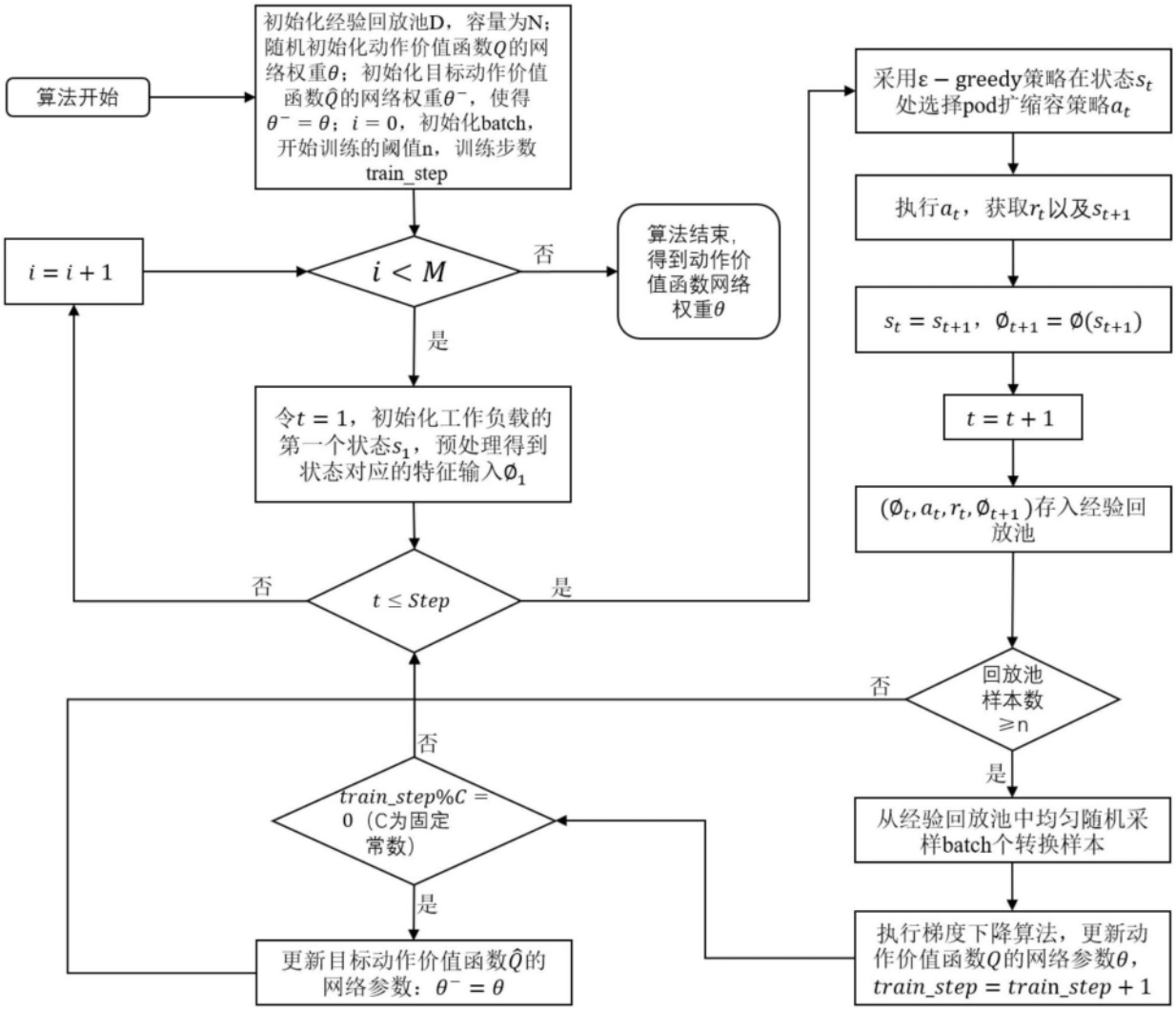

算法开始
初始化经验回放池D，容量为N；随机初始化动作价值函数Q的网络权重θ；初始化目标动作价值函数$\hat{Q}$的网络权重θ^-，使得$\theta^- = \theta$；$i = 0$，初始化batch，开始训练的阈值n，训练步数train_step
$i < M$
否
算法结束，得到动作价值函数网络权重θ
是
$i = i + 1$
令$t = 1$，初始化工作负载的第一个状态s_1，预处理得到状态对应的特征输入$\emptyset_1$
$t \leq Step$
否
是
采用$\varepsilon - greedy$策略在状态s_t处选择pod扩缩容策略a_t
执行a_t，获取r_t以及s_{t+1}
$s_t = s_{t+1}$，$\emptyset_{t+1} = \emptyset(s_{t+1})$
$t = t + 1$
$(\emptyset_t, a_t, r_t, \emptyset_{t+1})$存入经验回放池
回放池样本数≥n
否
是
从经验回放池中均匀随机采样batch个转换样本
执行梯度下降算法，更新动作价值函数Q的网络参数θ，$train_step = train_step + 1$
$train_step\%C = 0$（C为固定常数）
否
是
更新目标动作价值函数$\hat{Q}$的网络参数：$\theta^- = \theta$

图4

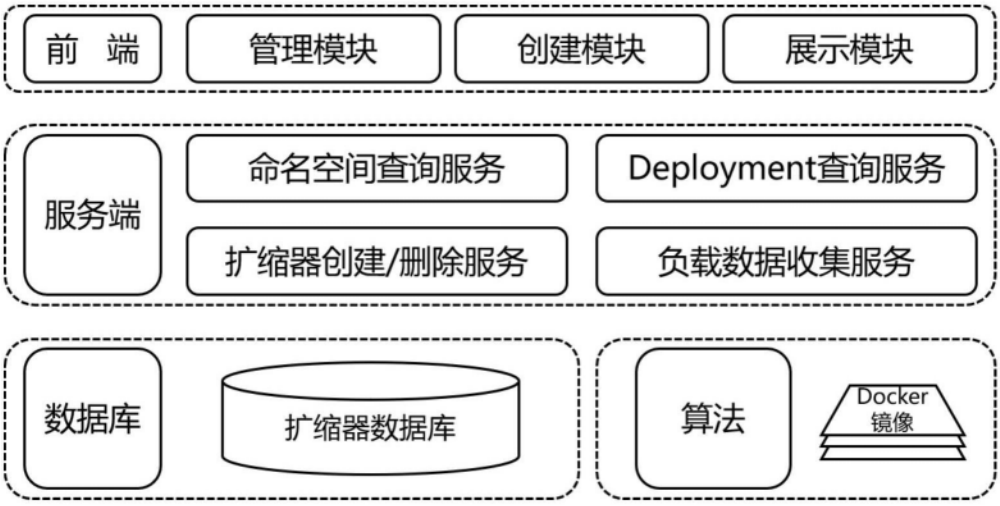

前 端
管理模块
创建模块
展示模块
服务端
命名空间查询服务
Deployment查询服务
扩缩器创建/删除服务
负载数据收集服务
数据库
扩缩器数据库
算法
Docker
镜像

图5