RLSK: A Job Scheduler for Federated Kubernetes Clusters based on Reinforcement Learning

研究背景

随着云计算技术的普及，数据中心中运行的任务种类日益多样化。然而，批处理作业（Batch Jobs）仍占据了很大比例。在传统的资源管理问题中，调度算法通常依赖手动设计与调整，这种方式效率低下且不具备自适应能力。而强化学习作为一种动态优化工具，近年来被广泛应用于任务调度问题，例如解决作业依赖关系的任务调度和虚拟机资源配置等问题。

现有研究

深度强化学习（DRL）在云计算调度中取得了一定成果，如：

- **DeepRM**：通过将状态表示为图像来调度作业。
- **Decima**：结合深度强化学习和图卷积网络解决Spark集群中依赖任务的调度问题。
- **资源管理**：多集群和多资源场景下的资源管理。

问题特殊挑战

- 在传统强化学习中，需要对系统状态和控制动作进行量化，且状态-动作空间随着资源和任务数目指数增长，增加了调度复杂性。
- 多集群和混合云环境中，调度难度进一步提升，需要合理映射状态和动作空间。

研究贡献

- 提出了 **RLSK**，一种基于深度强化学习的多集群调度器，专注于联邦式 Kubernetes 集群的资源调度问题。
- RLSK 能够根据历史数据学习跨集群的作业调度，平衡不同集群间资源的利用率，从而提高整体资源利用效率。

- 实现了 RLSK 并进行了仿真实验，结果表明其在资源利用率和性能上具有显著提升。

背景知识

- Kubernetes 采用主从架构，其中 **kube-scheduler** 负责将 pod 绑定到最适合的工作节点。
- 调度过程分为两阶段：
  1. **预选策略（Predicates）**：
     - 根据预选策略过滤不符合条件的节点。
     - 例如，如果某节点的资源不足以满足作业需求，该节点将被排除。
  2. **优先级策略（Priorities）**：
     - 根据优先级策略对预选节点进行排序，并选择优先级最高的节点分配任务。
     - 通常，资源丰富且负载较低的节点会有更高排名。
- **Kubernetes 的联邦（Federation）**：
  - 联邦管理多个 Kubernetes 集群，基于用户需求将不同的 pod 调度到不同集群。
  - 一般根据应用的地理区域需求，联邦将工作负载分配到不同的 Kubernetes 集群。
  - 目标是为终端用户提供更高带宽和更低延迟。

方法设计

作业调度场景

基本假设：

- 存在 m 个**同质化集群**，每个集群拥有相同类型和数量的资源（如 CPU、内存等）。
- 每个集群具有 D 种资源，作业以在线模式持续到达。

任务调度流程：

- RLSK 在每个调度点自动学习策略，将当前任务分配到最合适的集群。

- 每个任务的资源需求已知，表示为一个向量 $S_J = (r_{j1}, r_{j2}, \ldots, r_{jd})$，其中 r_{jk} 表示作业 j 对第 k 种资源的需求。

假设条件：

- 资源不可抢占：任务一旦被调度成功，分配的资源直到任务完成前都不会被回收。
- 每个集群视为一个资源池，不考虑集群内部的资源碎片化问题。

调度目标：

- 平衡不同集群之间的平均资源利用率。
- 减少每个集群内不同资源间的利用率差距。

RLSK框架设计

架构概述：

- RLSK 采用集中式调度模式，通过与各集群的主节点交互进行任务调度。
- RLSK 运行在集群之外，借助**辅助模块**与 Kubernetes 主节点通信，收集各集群状态数据并传递任务。

辅助模块的作用：

- 部署在每个 Kubernetes 集群的主节点，负责信息传递和资源操作。
- 使用 Kubernetes 原生 API，与集群交互执行相关操作。

调度过程：

- 用户提交批处理任务到 RLSK。
- RLSK 结合辅助模块提供的集群状态信息，通过训练好的代理（agent）决策模型进行调度。
- 最终将任务分配到相应的集群。

辅助模块主要功能：

1. **收集集群数据**：调用 Kubernetes API 遍历集群中的节点、Pod 和容器，采集资源状态信息（如 CPU、内存利用率）。
2. **任务接收与传递**：接收调度器分配的任务，并将其传递给 Kubernetes 内部调度器。每个任务被简化为独立容器的 Pod 资源对象。
3. **资源管理**：管理集群中所有节点的 Pod、容器和其他资源对象。

RLSK模型设计

状态定义：

- 系统的环境状态表示为一个**一维向量**，包括：
  1. 各集群当前资源使用率。
  2. 等待调度作业的特征。
- **集群资源使用率**：
  - 表示为集群中各资源的占用率。
  - 比如，第 i 个集群在第 t 时间的某种资源占用率表示为 U_{ip}^t。
- **作业特征**：
  - 转换为作业资源需求与集群可用资源的比例。
- **状态范围**：
  - 向量的每一项值范围在 [0, 1]。
  - 当前时间 t 的环境状态 S^t包括：
    - 所有集群的状态 S_C^t。
    - 当前作业的状态 S_j。
  - 表达式：$S^t = [S_C^t, S_j] = [C_1^t, C_2^t, \ldots, C_M^t, S_j]$
    其中 M 是集群数量，D 是资源种类。
- **整体状态的组成**：
  - S_C^t：描述 M 个集群各自的资源占用率。
  - S_j：描述任务 j 的资源需求比例。

动作定义：

- 动作代表调度代理对任务的决策。
- 动作空间是一个有限集合，动作 a=i 表示将任务分配到第 i 个集群。
- 动作空间：
  - 包括延迟调度策略：
    - 若系统当前负载过高（如所有集群资源接近饱和），代理可以选择**推迟调度**。
    - 用动作 a=0 表示推迟当前任务的调度。
  - 动作总集合定义为

$$A = \{a \mid a \in 0, 1, 2, \ldots, M\}$$

- 动作执行：
  - 若选择动作 a=0，任务将被随机推迟一段时间。
  - 若选择动作 a=i，任务分配到第 i 个集群。

奖励设计：

- **调度目标**：
  - 提高多个集群的整体资源利用率。
  - 平衡每个集群的平均资源利用率。
  - 减少单个集群内不同资源的利用率差异，避免单一资源瓶颈。
- 奖励函数 $r(t)$ 定义为三项指标的线性组合：

$$r(t) = \alpha Util(t) - \beta DiffCluster(t) - \gamma DiffRes(t)$$

其中：

- $Util(t)$：当前总资源利用率；
- $DiffCluster(t)$：集群间资源利用率差异；
- $DiffRes(t)$：集群内不同资源间的利用率差异；
- α, β, γ：调节系数，用于平衡三项目标的相对重要性。

资源利用率的计算：

- 单个集群的资源利用率：定义为集群中所有资源的平均利用率：

$$U_m(t) = \frac{1}{D} \sum_{i=1}^{D} U_m^i$$

- 总资源利用率：定义为所有集群利用率的总和：

$$Util(t) = \sum_{m=1}^{M} U_m(t)$$

资源均衡性的衡量：

- 集群内资源利用率的差异：使用各资源利用率之间的绝对差值之和衡量：

$$DiffCluster(t) = \sum_{m=1}^{M} \sum_{i=1}^{D} \sum_{j=i+1}^{D} \left| U_m^i - U_m^j \right|$$

- 集群间资源利用率的差异：使用各集群间同类资源利用率的绝对差值之和衡量：

$$DiffRes(t) = \sum_{i=1}^{D} \sum_{m=1}^{M} \sum_{n=m+1}^{M} \left| U_m^i - U_n^i \right|$$

训练算法：DQN

实验设置

- 实验环境
  - 设置了三个同质化集群。
  - 假设存在两种资源（如 CPU 和内存），每种资源容量均为 1。
  - 任务到达模式与 DeepRM 类似，任务按照伯努利过程到达。
- 任务特性
  - 任务的持续时间：
    - 80% 的任务持续时间在 1t 至 60t 之间均匀分布。
    - 其余 20% 的任务持续时间在 200t 至 300t 之间均匀分布。
  - 任务资源需求
    - 每个任务有一个主要资源（dominant resource），随机选择。
    - 主要资源需求均匀分布在 0.025r 至 0.05r 之间。
    - 次要资源需求均匀分布在 0.005r 至 0.01r 之间。
- 对比算法
  1. **First Fit**：将任务调度到第一个满足资源需求的集群。
  2. **Random**：将任务随机调度到某个集群。
  3. **Round-Robin**：轮询将任务分配到不同集群。
  4. **Least Load**：将任务分配到当前负载最小的集群。
- 评价指标：
  - 实时资源利用率曲线（intuitive real-time resource utilization curves）。

- 集群内的平均资源利用率：$mean = (CPUutilization + Memoryutilization)/2$
- 集群内不同资源间的利用率差异程度：$(\frac{CPU}{mean} - 1)^2 + (\frac{Memory}{mean} - 1)^2$
- 完工时间：每个集群的最大完成时间；