



# 基于DRL的动态租用实例任务调度

刘肇泽

控制与计算机工程学院

2023 年 10 月 16 日



# 目录

- ① 数学模型
  - 实例建模
  - 任务建模
  - 任务调度流程建模
- ② DQN结构
- ③ 实验结果
- ④ 问题与展望
- ⑤ 参考文献



与 Cost-Aware 的不同之处:

- ① 引入了 On Demand 和 Spot 两种计费规则的实例
- ② 引入了 Moldable 和 Rigid 两种任务类型
- ③ 引入了任务的挂起、恢复和跨区域调度



- ① 数学模型
- ② DQN结构
- ③ 实验结果
- ④ 问题与展望
- ⑤ 参考文献



使用  $T$  表示时间段,  $t$  表示时刻.



- ① 数学模型  
实例建模  
任务建模  
任务调度流程建模
- ② DQN结构
- ③ 实验结果
- ④ 问题与展望
- ⑤ 参考文献



## 实例 (Instance) 的数学模型

### 固有属性:

- $I_c$  实例的计算核心数
- $I_m$  实例的内存大小
- $I_r$  实例所在区域
- $I_b$  实例的计费类型 (On Demand/Spot)
- $v_s$  实例中任务挂起 (suspend) 速度
- $v_r$  实例中任务恢复 (resume) 速度

### 状态属性:

- $T_r$  实例的剩余租期 (remain time)
- $t_i$  实例空闲的时刻 (idle time)

### 计费类型:

- On Demand: 每小时价格固定
- Spot: 每小时价格随市场波动



## ① 数学模型

实例建模

任务建模

任务调度流程建模

## ② DQN结构

## ③ 实验结果

## ④ 问题与展望

## ⑤ 参考文献





## 任务 (Job) 的数学模型

### 固有属性:

- $J_c$  任务需要的计算核心数
- $J_m$  任务需要的内存大小
- $J_t$  任务的类型 (Moldable/Rigid)

### 状态属性:

- $J_l$  任务长度 (小时)
- $t_s$  任务提交时刻 (submit time)
- $J_r$  任务上一次执行所在的区域



## Downey 加速模型

并行度 (parallelism) 具有均值  $A$  和标准差  $\sigma$  两个参数.

将  $0 \leq \sigma \leq 1$  的模型称为 Low variance model, 将  $\sigma > 1$  的模型称为 High variance model.

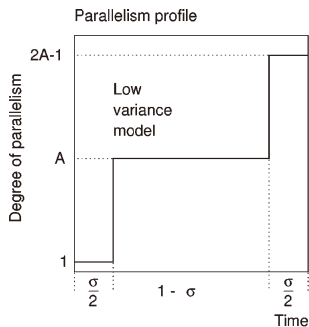


图: Low variance model

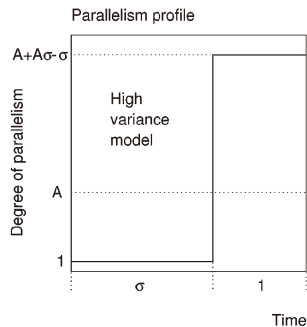


图: High variance model



## Downey 加速模型

加速系数  $SU(n)$  的计算

当  $0 \leq \sigma \leq 1$  时:

$$SU(n) = \begin{cases} \frac{An}{A+\sigma(n-1)/2}, & 1 \leq n \leq A, \\ \frac{An}{\sigma(A-1/2)+n(1-\sigma/2)}, & A < n \leq 2A-1, \\ A, & n > 2A-1. \end{cases}$$

当  $\sigma > 1$  时:

$$SU(n) = \begin{cases} \frac{nA(\sigma+1)}{A+A\sigma-\sigma+n\sigma}, & 1 \leq n \leq A + A\sigma - \sigma, \\ A, & n > A + A\sigma - \sigma. \end{cases}$$

其中  $n$  为计算核心数.

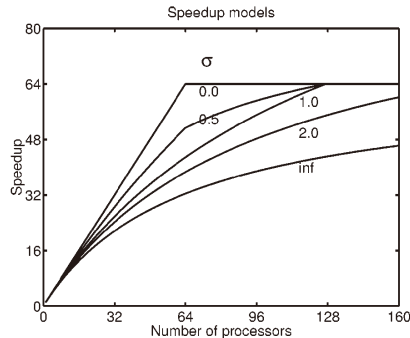


图: Speedup curves for a range of values of  $\sigma$  when  $A = 64$



# 任务类型 $J_t$ 与实际执行时间 $T^e$

Moldable/Rigid

任务类型  $J_t$  影响任务实际执行时间  $T^e$  的计算.

---

## Algorithm 1: 计算任务实际执行时间 $T^e$

---

```
1 if  $J_t = Moldable$  then
2    $T^e = J_l \cdot \frac{SU(J_c)}{SU(I_c)}$ ;
3 end
4 if  $J_t = Rigid$  then
5   if  $J_c \leq I_c$  then
6      $T^e = J_l$ ;
7   else
8     任务调度失败;
9   end
10 end
```

---



## 挂起时间 $T^s$ 与恢复时间 $T^r$

当正在执行任务的实例到期时, 任务需要挂起到硬盘, 该操作需要的时间为:

$$T^s = \frac{J_m}{v_s}$$

当任务首次在实例中运行或从挂起状态恢复时, 任务需要加载到内存, 该操作需要的时间为:

$$T^r = \begin{cases} \frac{J_m}{v_r}, & J_r = \text{None} \text{ or } J_r = I_r, \\ 2 \times \frac{J_m}{v_r}, & \text{otherwise.} \end{cases}$$



## ① 数学模型

实例建模

任务建模

任务调度流程建模

## ② DQN结构

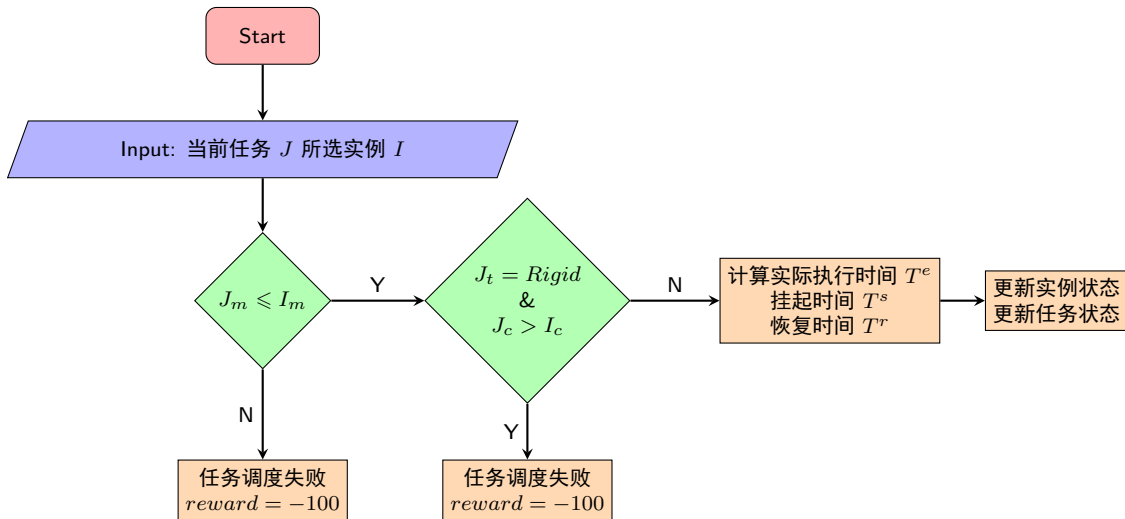
## ③ 实验结果

## ④ 问题与展望

## ⑤ 参考文献

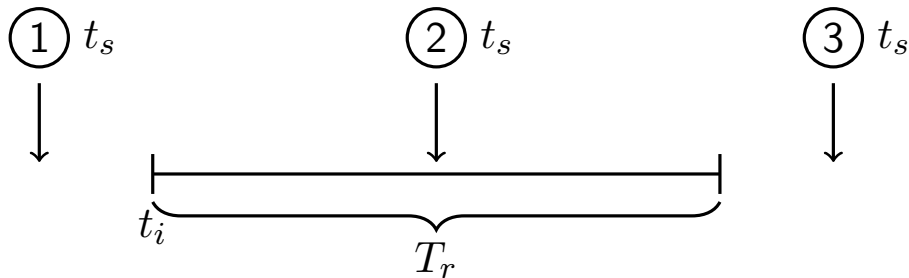


## 任务调度流程





## 任务提交的三种情况



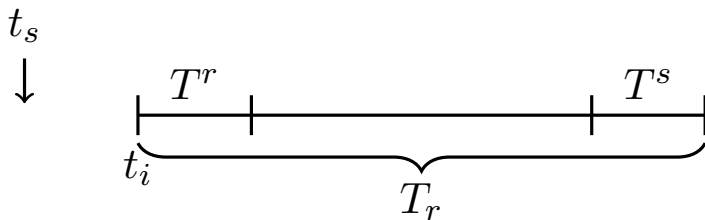
- ① 任务在实例空闲之前提交  $t_s \leq t_i$
- ② 任务在实例空闲之后、到期之前提交  $t_i < t_s < t_i + T_r$
- ③ 任务在实例到期之后提交  $t_s \geq t_i + T_r$





## 情况一：任务在实例空闲之前提交 $t_s \leq t_i$

判断是否续租



---

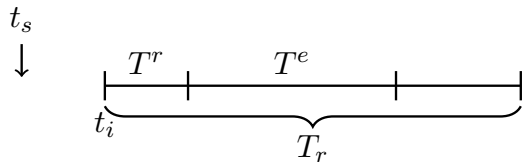
```
1 cost  $\leftarrow$  0;  
2 while  $T_r - T^r - T^s \leq 0$  do  
3   |  $T^r \leftarrow T^r + 1$ ;          /* 续租一小时 */  
4   | cost  $\leftarrow$  cost +  $t_i$ 时刻的实例价格;  
5 end
```

---



情况一：任务在实例空闲之前提交  $t_s \leq t_i$

任务能够全部执行  $T^e \leq T_r - T^r$

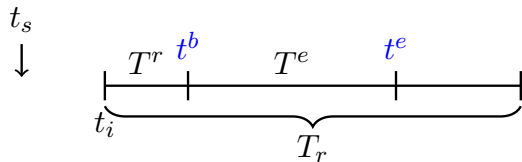




## 情况一：任务在实例空闲之前提交 $t_s \leq t_i$

任务能够全部执行  $T^e \leq T_r - T^r$

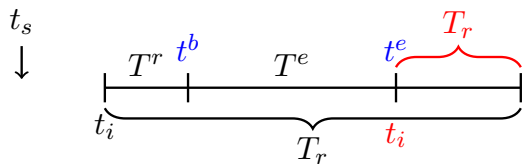
任务在  $t^b = t_i + T^r$  时刻开始执行, 在  
 $t^e = t^b + T^e$  时刻结束执行.





## 情况一：任务在实例空闲之前提交 $t_s \leq t_i$

任务能够全部执行  $T^e \leq T_r - T^r$



任务在  $t^b = t_i + T^r$  时刻开始执行, 在  $t^e = t^b + T^e$  时刻结束执行.

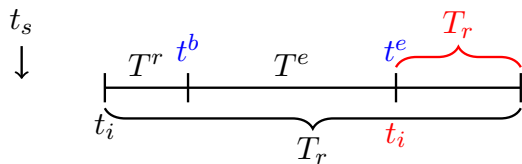
更新实例状态:

- 实例剩余租期  $T_r \leftarrow T_r - T^r - T^e$
- 实例空闲的时刻  $t_i \leftarrow t^e$



## 情况一：任务在实例空闲之前提交 $t_s \leq t_i$

任务能够全部执行  $T^e \leq T_r - T^r$



任务在  $t^b = t_i + T^r$  时刻开始执行, 在  $t^e = t^b + T^e$  时刻结束执行.

更新实例状态:

- 实例剩余租期  $T_r \leftarrow T_r - T^r - T^e$
- 实例空闲的时刻  $t_i \leftarrow t^e$

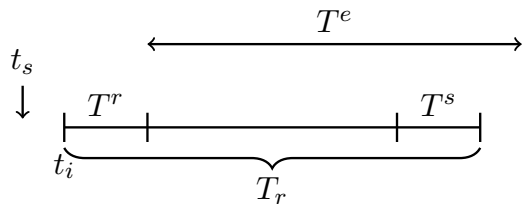
更新任务状态:

- 任务长度  $J_l \leftarrow 0$
- 任务上一次执行所在的区域  $J_r \leftarrow I_r$



# 情况一：任务在实例空闲之前提交 $t_s \leq t_i$

任务只能部分执行  $T^e > T_r - T^r$

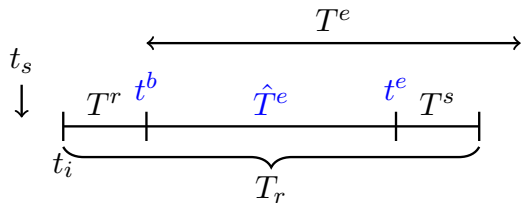




## 情况一：任务在实例空闲之前提交 $t_s \leq t_i$

任务只能部分执行  $T^e > T_r - T^s$

任务在  $t^b = t_i + T^r$  时刻开始执行, 在  
 $t^e = t_i + T_r - T^s$  时刻结束执行.  
任务实际执行时长  $\hat{T}^e = t^e - t^b$ .





## 情况一：任务在实例空闲之前提交 $t_s \leq t_i$

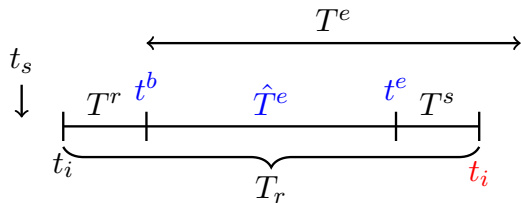
任务只能部分执行  $T^e > T_r - T^r$

任务在  $t^b = t_i + T^r$  时刻开始执行, 在  $t^e = t_i + T_r - T^s$  时刻结束执行.

任务实际执行时长  $\hat{T}^e = t^e - t^b$ .

更新实例状态:

- 实例剩余租期  $T_r \leftarrow 0$
- 实例空闲的时刻  $t_i \leftarrow t_i + T_r$







## 情况一：任务在实例空闲之前提交 $t_s \leq t_i$

任务只能部分执行  $T^e > T_r - T^r$

任务在  $t^b = t_i + T^r$  时刻开始执行, 在  $t^e = t_i + T_r - T^s$  时刻结束执行.

任务实际执行时长  $\hat{T}^e = t^e - t^b$ .

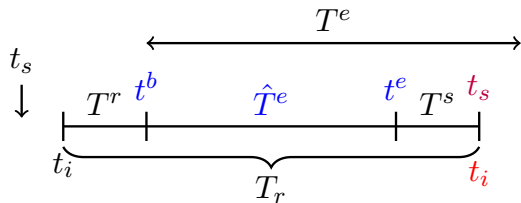
更新实例状态:

- 实例剩余租期  $T_r \leftarrow 0$
- 实例空闲的时刻  $t_i \leftarrow t_i + T_r$

更新任务状态:

- 任务长度  

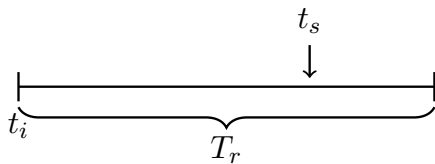
$$J_l \leftarrow \begin{cases} J_l - \hat{T}^e, & J_t = \text{Rigid}, \\ J_l - \hat{T}^e / \frac{SU(J_c)}{SU(I_c)}, & J_t = \text{Moldable}. \end{cases}$$
- 任务上一次执行所在的区域  $J_r \leftarrow I_r$
- 任务提交时刻  $t_s \leftarrow t_i$





## 情况二：任务在实例空闲之后、到期之前提交

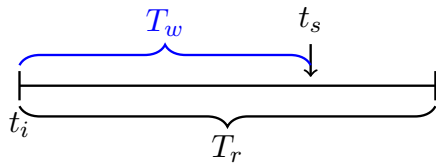
$$t_i < t_s < t_i + T_r$$





## 情况二：任务在实例空闲之后、到期之前提交

$$t_i < t_s < t_i + T_r$$

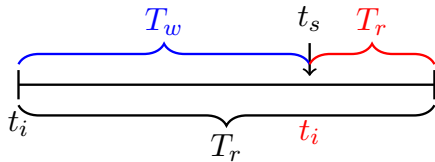


浪费时间  $T_w = t_s - t_i$ .



## 情况二：任务在实例空闲之后、到期之前提交

$$t_i < t_s < t_i + T_r$$



浪费时间  $T_w = t_s - t_i$ .

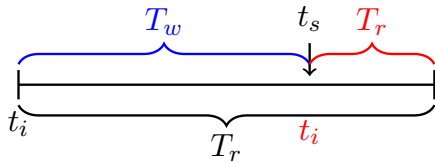
更新实例状态:

- 实例剩余租期  $T_r \leftarrow T_r - T_w$
- 实例空闲的时刻  $t_i \leftarrow t_s$



## 情况二：任务在实例空闲之后、到期之前提交

$$t_i < t_s < t_i + T_r$$



浪费时间  $T_w = t_s - t_i$ .

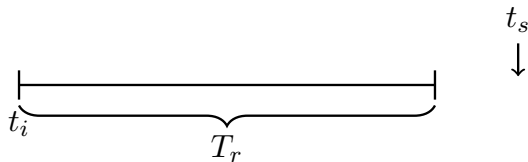
更新实例状态:

- 实例剩余租期  $T_r \leftarrow T_r - T_w$
- 实例空闲的时刻  $t_i \leftarrow t_s$

转换为情况一.

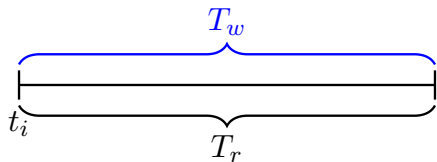


# 情况三：任务在实例到期之后提交 $t_s \geq t_i + T_r$





# 情况三：任务在实例到期之后提交 $t_s \geq t_i + T_r$

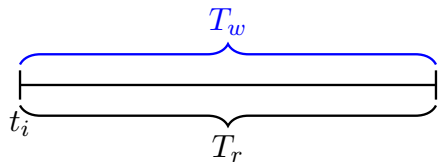


$t_s$   
↓

浪费时间  $T_w = T_r$ .



## 情况三：任务在实例到期之后提交 $t_s \geq t_i + T_r$



浪费时间  $T_w = T_r$ .

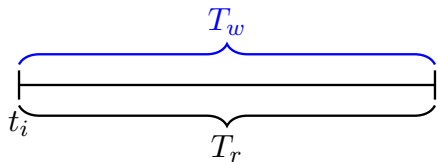
更新实例状态:

- 实例剩余租期  $T_r \leftarrow 0$
- 实例空闲的时刻  $t_i \leftarrow t_s$





## 情况三：任务在实例到期之后提交 $t_s \geq t_i + T_r$



$t_s$   
↓

$t_i$

浪费时间  $T_w = T_r$ .

更新实例状态:

- 实例剩余租期  $T_r \leftarrow 0$
- 实例空闲的时刻  $t_i \leftarrow t_s$

转换为情况一.



## 任务调度成功时的奖励

$$reward = \max \left\{ -(\xi \cdot \text{cost} + \eta \cdot T_w) \cdot \frac{t^e - t_s}{t^e - t^b}, -50 \right\}$$

将租金cost和浪费的时间  $T_w$  作为惩罚项, 任务响应比  $(t^e - t_s)/(t^e - t^b)$  作为系数. 限制任务调度成功时的奖励不低于  $-50$ , 与任务调度失败的奖励  $-100$  拉开差距.  
其中,  $\xi$  和  $\eta$  是超参数.



- ① 数学模型
- ② DQN结构
- ③ 实验结果
- ④ 问题与展望
- ⑤ 参考文献



## 状态向量的构成

DQN输入的状态向量为:

$$\left[ J_c \quad J_m \quad J_t \quad J_r \quad t_i^{(1)} - t_s \quad T_r^{(1)} - J_l \quad \dots \quad t_i^{(n)} - t_s \quad T_r^{(n)} - J_l \right]$$

即: 任务需要的计算核心数、任务需要的内存大小、任务的类型、任务上一次执行所在的区域; 任务提交给每个实例后预计的等待时间和实例剩余的租期.



- ① 数学模型
- ② DQN结构
- ③ 实验结果**
- ④ 问题与展望
- ⑤ 参考文献



## 训练过程

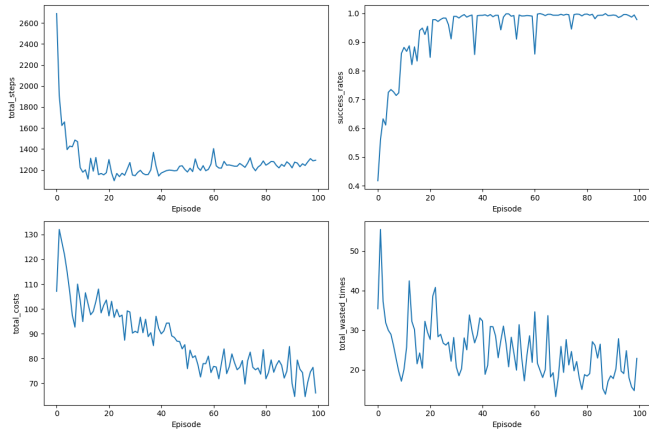


图: 训练过程中调度总数、调度成功率、总开销、总浪费时间的变化



## 调度算法评估

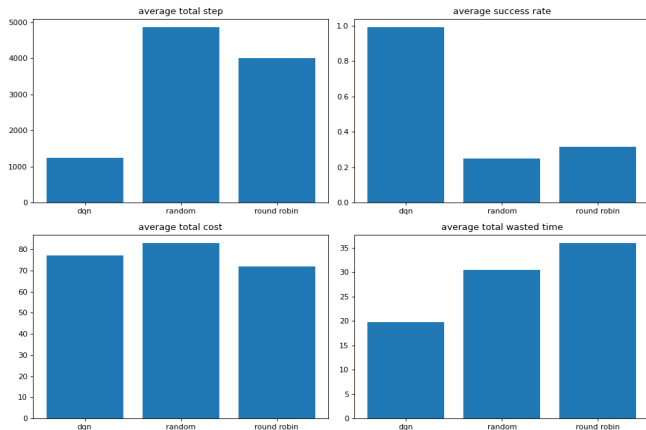


图: DQN、Random、Round-Robin三种调度算法的表现

Earliest算法无法完成该数学模型下的调度任务.



- ① 数学模型
- ② DQN结构
- ③ 实验结果
- ④ 问题与展望**
- ⑤ 参考文献





## 问题一：时间指标

在任务调度流程中, 记录了每次调度任务的提交时刻  $t_s$ , 开始时刻  $t^b$  和结束时刻  $t^e$ . 应当如何使用这些时间数据更好地比较不同调度算法的性能?

目前想到的方法:

- ① 比较平均每次任务调度的响应时间  $t^e - t_s$
- ② 比较平均每次任务调度的相应比  $t^e - t_s / t^e - t^b$
- ③ 比较平均每个任务从第一次提交到最终全部完成的时间



## 问题二: 调度成功率

目前DRL调度算法的成功率不能达到 100%. 为了保证调度流程能够正常进行, 在DRL调度失败时使用Round-Robin调度算法, 将任务分配到可以执行的实例上.

是否有可行的办法使DRL调度算法的成功率稳定在 100%?

- ① 应用DQN的全部高级技巧: 优先经验回放、双Q学习、对决网络和噪声网络
- ② 增加训练轮数
- ③ ...



- ① 数学模型
- ② DQN结构
- ③ 实验结果
- ④ 问题与展望
- ⑤ 参考文献



## 参考文献



Allen B. Downey.

A parallel workload model and its implications for processor allocation.  
*Cluster Computing*, pages 133–145, May 1998.



Luke M. Leslie, Young Choon Lee, Peng Lu, and Albert Y. Zomaya.

Exploiting Performance and Cost Diversity in the Cloud.

In *2013 IEEE Sixth International Conference on Cloud Computing*, pages 107–114, Santa Clara, CA, June 2013. IEEE.