11.11新尝试

- 状态 S 应该包含两个部分
  - 当前需要调度的任务信息
    - 多数job只有一个task，task会被分为多个instance
    - gap between `pai_job_table.start_time` and the earliest `pai_task_table.start_time` in the job implies its wait time before launching (scheduling latency).
    - 按照task来请求资源
      - pai_task_table.plan_cpu/plan_mem/plan_gpu/gpu_type/inst_num是task需要的总实例个数，所以
    - pai_group_tag_table.gpu_type_spec/workload/user相同可能提交的任务也是同类型的
  - 当前整个集群的资源视图
    - 静态：pai_machine_spec.cap_cpu/cap_mem/cap_gpu/gpu_type
    - 动态：数据不在表格里，current_cpu/mem/gpu_usage, num_running_instances
- Action：分配instance给机器，从约1800+台机器中选择一台
  - 分配机器时要考虑
    - 请求的gpu_type需要是对应的机器
    - 考虑机器剩余的cpu/mem/gpu都足够
  - 如果没有合适的机器就需要排队等待，等到有机器完成任务、释放资源后重试
- Reward（可以选择1种，也可以加超参数调节不同奖励的权重）
  - 最小化排队时间:
    - 当一个实例*终于*被成功调度时，计算它的排队时间。
    - Reward = - (排队时间) = the earliest `pai_task_table.start_time` - `pai_job_table.start_time`
  - 最大化集群利用率
    - 鼓励 Agent 把任务“塞满”机器
    - Reward = (task.plan_cpu / machine[j].cap_cpu) + (task.plan_gpu / machine[j].cap_gpu)
  - 最小化作业完成时间
    - Reward = -JCT = `pai_job_table.start_time` - `pai_job_table.end_time`
- 问题：不要！暴露duration给agent，但是Environment也就是模拟器知道duration
  - 计算 `duration = instance.end_time - instance.start_time`

    在模拟器的未来时间 `t + duration` 时刻，触发一个“实例完成”事件。
- 流程
  - pai_job_table.start_time作为作业提交时间，对所有作业进行排序
  - 从事件队列中取出时间最早的事件，抓取任务信息和当前集群状态
  - agent决策并执行action

        - 分配给某机器，
        - 无机器可用，放入待处理队列
        - 计算reward
    - 经过duration之后，更新该作业完成作业完成
        - 释放机器资源
        - 检查待处理队列，若可放下，则进行该任务的调度决策