PR #2024 完整报告

THUDM/slime

Log progress while waiting for placement group

合并时间：2026-06-08 09:38

原文链接：http://prhub.com.cn/THUDM/slime/pull/2024

执行摘要

- 一句话：为 placement group 等待加入周期性日志
- 推荐动作：值得合入；改动简洁且解决实际痛点。如果未来遇到 placement group 长时间无法放置的情况，可考虑在此基础上增加可配置的超时。

功能与动机

PR body 指出，裸 ray.get(pg.ready()) 在 placement group 无法立即放置时（节点 GPU 尚未注册到 Ray GCS，或 autoscaler 仍在拉起节点）会无任何输出地阻塞，最后一行日志停留在"Creating placement group with N GPUs..."，用户无法判断是卡死、慢速还是正在扩缩容。
这可能是某些"训练启动挂起"报告的原因。

实现拆解

1. 替换等待方式：在 slime/ray/placement_group.py 的 _create_placement_group 函数中，将 ray.get(pg.ready()) 改为 ready_ref = pg.ready() 后使用 ray.wait([ready_ref], timeout=log_interval) 循环轮询，每 30 秒检查一次。
2. 添加进度日志：每次超时后，通过 ray.cluster_resources().get("GPU", 0) 和 ray.available_resources().get("GPU", 0) 获取集群 GPU 总数与可用数，格式化输出如 Waiting for placement group of 16 GPUs (elapsed 30s): 8 GPUs registered with Ray, 8 available.。
3. 保持无超时：等待循环没有设置总超时，只有当 ready_ref 就绪时才退出，因此自动扩缩容集群中 placement group 挂起驱动扩容的行为不受影响。

关键文件：

- slime/ray/placement_group.py（模块 放置组；类别 source；类型 core-logic；符号 _create_placement_group）：核心变更文件，修改 _create_placement_group 函数的等待逻辑，添加轮询和进度日志。

关键符号：_create_placement_group

关键源码片段

slime/ray/placement_group.py

核心变更文件，修改 _create_placement_group 函数的等待逻辑，添加轮询和进度日志。

```
# slime/ray/placement_group.py（关键变更部分）

def _create_placement_group(num_gpus):
    """Create a placement group with the specified number of GPUs."""
    if num_gpus == 0:
        return None, [], []

    bundles = [{"GPU": 1, "CPU": 1} for _ in range(num_gpus)]
    pg = placement_group(bundles, strategy="PACK")
    num_bundles = len(bundles)

    # 轮询等待 placement group 就绪，每次超时 30 秒后打印进度
    # 避免裸 ray.get(pg.ready()) 导致无输出挂起
    ready_ref = pg.ready()
    elapsed = 0
    log_interval = 30
    while not ray.wait([ready_ref], timeout=log_interval)[0]:
        elapsed += log_interval
        # 查询 Ray 集群资源状态
        total = ray.cluster_resources().get("GPU", 0)
        available = ray.available_resources().get("GPU", 0)
        logger.info(
            f"Waiting for placement group of {num_gpus} GPUs (elapsed {elapsed}s): "
            f"{total:g} GPUs registered with Ray, {available:g} available."
        )
    # 等待结束后，原有逻辑不变：使用 InfoActor 获取 GPU ID 等
    # ...（后续代码未改动）
```

评论区精华

PR 作者 luccabb 在 issue 评论中提出是否应增加超时机制。仓库维护者 zhuzilin 表示原有日志不直观，但对超时不确定，未做结论。代码中最终未加入超时，保持了无超时设计。

- 是否应增加超时机制 (question): 未决定，当前实现保持无超时。

风险与影响

- 风险：风险极低：仅修改了等待逻辑，不改变 placement group 创建后的行为；轮询频率 30 秒，不会产生明显性能影响；无超时避免了在自动扩缩容场景下引发误超时中断。未添加测试覆盖，但改动简单可靠。
- 影响：对用户：显著改善启动阶段的可观测性，便于区分“等待资源”与“真正卡死”。对系统：无功能变化。对团队：无。
- 风险标记：缺少测试覆盖

关联脉络

- PR #2016 Fully support --rollout-external-engine-addrs: 也修改了 placement_group.py 文件（添加外部引擎支持），与该 PR 属于同一模块。