PR #1880 完整报告

THUDM/slime

[Fix] Fix distributed POST actor concurrency split

合并时间：2026-05-09 18:12

原文链接：http://prhub.com.cn/THUDM/slime/pull/1880

执行摘要

- 一句话：修复分布式 POST 每 actor 并发计算错误
- 推荐动作：值得合并：一个明确的 bugfix，逻辑正确，改动精简，风险低。建议阅读 slime/utils/http_utils.py 中的 _init_ray_distributed_post 函数，理解 actor 拓扑与并发分配的关系。

功能与动机

PR body 指出，当前实现中 per_actor_conc 只除以节点数 len(nodes)，但在每节点创建 args.num_gpus_per_node 个 actor 的情况下，总 actor 数量远多于节点数，导致每 actor 分配的并发数远超预期，可能引发资源过度分配。例如：8 节点、每节点 8 actor、_client_concurrency = 1024 时，原计算为 ceil(1024 / 8) = 128+，而实际应为 ceil(1024 / 64) = 16。

实现拆解

1. 修改 _init_ray_distributed_post 中的并发计算（slime/utils/http_utils.py）：
   - 引入 total_actors = max(1, len(nodes) * args.num_gpus_per_node) 计算总 actor 数。
   - 将 per_actor_conc 的计算改为 max(1, (_client_concurrency + total_actors - 1) // total_actors)，即向上取整并保证至少为 1。
2. 更新 docstring：将原来的 "Uses NodeAffinitySchedulingStrategy to place actors on distinct nodes. Controlled by SLIME_HTTP_POST_ACTORS_PER_NODE." 修正为实际行为描述："Uses NodeAffinitySchedulingStrategy to place actors on alive Ray nodes. Creates args.num_gpus_per_node actors per node."。
3. 保持 actor 创建拓扑不变：仍按 args.num_gpus_per_node 个 actor 每节点创建，无其他改动。

关键文件：

- slime/utils/http_utils.py（模块 工具层；类别 source；类型 core-logic；符号 _init_ray_distributed_post）：包含分布式 POST 的初始化和 actor 创建逻辑，本次修改了 per_actor_conc 的计算方式和 docstring。

关键符号：_init_ray_distributed_post

关键源码片段

slime/utils/http_utils.py

包含分布式 POST 的初始化和 actor 创建逻辑，本次修改了 per_actor_conc 的计算方式和 docstring。

```
# slime/utils/http_utils.py (head)

def _init_ray_distributed_post(args):
    """Initialize one or more Ray async actors per node for HTTP POST.

    Uses NodeAffinitySchedulingStrategy to place actors on alive Ray nodes.
    Creates ``args.num_gpus_per_node`` actors per node.
    """
    global _post_actors
    if _post_actors:
        return # Already initialized

    import ray
    from ray.util.scheduling_strategies import NodeAffinitySchedulingStrategy

    # Discover alive nodes
    nodes = [n for n in ray.nodes() if n.get("Alive")]
    if not nodes:
        raise RuntimeError("No alive Ray nodes to place HTTP POST actors.")

    # Define the async actor
    @ray.remote
    class _HttpPosterActor:
        def __init__(self, concurrency: int):
            # Lazy creation to this actor's event loop
            self._client = httpx.AsyncClient(
                limits=httpx.Limits(max_connections=max(1, concurrency)),
                timeout=httpx.Timeout(None),
                trust_env=False, # internal SGLang comm only — never route through system proxy
            )

        async def do_post(self, url, payload, max_retries=60, headers=None):
            return await _post(self._client, url, payload, max_retries, headers=headers)

    # Create actors per node
    created = []
    # 计算总 actor 数为节点数乘以每节点 actor 数，并确保至少为 1
    total_actors = max(1, len(nodes) * args.num_gpus_per_node)
    # 将全局并发数平均分配到所有 actor，向上取整，确保至少为 1
    per_actor_conc = max(1, (_client_concurrency + total_actors - 1) // total_actors)

    for node in nodes:
        node_id = node["NodeID"]
        scheduling = NodeAffinitySchedulingStrategy(node_id=node_id, soft=False)
```

```
        for _ in range(args.num_gpus_per_node):
            actor = _HttpPosterActor.options(
                name=None,
                lifetime="detached",
                scheduling_strategy=scheduling,
                max_concurrency=per_actor_conc,
                num_cpus=0.001,
            ).remote(per_actor_conc)
            created.append(actor)

    _post_actors = created
```

评论区精华

该 PR 没有 review 评论或讨论线程。提交者 kaysonyu 在 PR body 中提供了清晰的动机和计算示例。

- 暂无高价值评论线程

风险与影响

- 风险：风险极低：只修改了 per_actor_conc 的计算方式，且值变小（更保守），不会导致资源超分。增加 max(1, ...) 保证非零，避免除零或零并发。没有引入新配置或行为变更，不会影响非分布式 POST 路径。
- 影响：影响范围小：仅影响启用 --use_distributed_post 且 args.num_gpus_per_node > 1 的场景。修复后每 actor 并发数更合理，避免连接过多导致的性能下降或资源浪费。无用户可见行为变化。
- 风险标记：暂无

关联脉络

- PR #1873 [Fix] Use Ray ObjectRef await instead of asyncio.to_thread in distributed POST: 同样修改了 slime/utils/http_utils.py 中的分布式 POST 逻辑，引入直接 await Ray ObjectRef 来消除线程池瓶颈，与本 PR 同属分布式 POST 路径优化。
- PR #1883 fix(qwen3_next): use torch.get_default_dtype() — get_current_dtype do…: 同为 bugfix 类 PR，但修改不同文件。无直接关联。