PR #2143 完整报告

THUDM/slime

Fix parallel update_from_disk in megatron server

合并时间：2026-06-29 12:03

原文链接：http://prhub.com.cn/THUDM/slime/pull/2143

执行摘要

- 一句话：修复 Megatron 服务器并行 update_from_disk 问题
- 推荐动作：值得精读，特别是锁和 future 的使用方式。建议在后续 PR 中补充对应的并发测试。

功能与动机

修复并发 update_from_disk 可能导致的资源竞争和错误，确保多个请求同时到达时的正确性。

实现拆解

1. 在 megatron_server.py 的 _build_http_app 中，将 update_state 扩展为包含 updating_model_path 和 update_future，并引入 asyncio.Lock 保护关键路径。
2. 在 update_from_disk 处理函数中，先获取锁检查当前模型路径是否与请求相同，若相同则直接返回 skipped=True。
3. 若已有更新进行中，判断目标模型路径是否一致：若一致则等待现有 future 并标记 coalesced=True；否则返回冲突错误。
4. 将更新结果和 args 赋值移到锁外，通过 error/result 变量统一在 finally 中处理。

关键文件：

- slime/backends/megatron_utils/server/megatron_server.py（模块 服务端；类别 source；类型 core-logic；符号 _build_http_app, update_from_disk）：核心变更文件，修改了 update_from_disk 处理逻辑，引入锁和请求合并机制。

关键符号：_build_http_app, update_from_disk

关键源码片段

slime/backends/megatron_utils/server/megatron_server.py

核心变更文件，修改了 update_from_disk 处理逻辑，引入锁和请求合并机制。

```
# slime/backends/megatron_utils/server/megatron_server.py
# 在 _build_http_app 中，扩展 update_state 并引入锁
async def update_from_disk(request: web.Request) -> web.Response:
    if update_from_disk_fn is None:
        return _json_error("update_from_disk is not available during warmup", 503)
```

```
payload = await _read_json_payload(request)
model_path = _get_update_model_path(payload)
if model_path is None:
    return _json_error("missing model_path", 400)

# 使用锁保护状态检查与更新
async with update_lock:
    # 如果模型路径与当前已加载的相同，直接跳过
    if getattr(args, "load", None) == model_path:
        return web.json_response({"ok": True, "model_path": model_path, "skipped": True})

    if update_state["in_progress"]:
        # 如果已有相同路径的更新在进行中，则等待其完成（请求合并）
        if update_state["updating_model_path"] == model_path and update_state["update_
        future"] is not None:
            update_future = update_state["update_future"]
            coalesced = True
        else:
            # 不同路径的更新冲突
            updating_model_path = update_state["updating_model_path"]
            return _json_error(f"update_from_disk is already in progress for {updating_model_
            path}", 409)
    else:
        # 无进行中更新，创建新 future 并记录状态
        update_future = asyncio.get_running_loop().create_future()
        update_state["in_progress"] = True
        update_state["updating_model_path"] = model_path
        update_state["update_future"] = update_future
        coalesced = False

if coalesced:
    # 等待已有的更新结果，并添加 coalesced 标记
    result = await asyncio.shield(update_future)
    if result.get("ok") is True:
        result = dict(result)
        result["coalesced"] = True
        return web.json_response(result)
    return _json_error(result.get("error", "update_from_disk failed"), int(result.get("status",
    500)))

timeout_s = _get_update_timeout_s(payload, args)
result = None
error = None
try:
    before_loads = await _wait_until_idle(sample_manager, timeout_s)
    update_result = await asyncio.to_thread(update_from_disk_fn, model_path)
    after_loads = await _ray_get(sample_manager.get_loads.remote())
except TimeoutError as e:
    error = {"ok": False, "status": 503, "error": str(e)}
```

```
    except Exception as e:
        error = {"ok": False, "status": 500, "error": f"update_from_disk failed: {e}"}
    finally:
        if error is None:
            result = {
                "ok": True,
                "model_path": model_path,
                "before_loads": before_loads,
                "after_loads": after_loads,
                "update_result": update_result,
            }
            # 同步 args 中的模型路径
            args.load = model_path
            args.ref_load = model_path

        async with update_lock:
            if result is not None:
                update_state["in_progress"] = False
                update_state["updating_model_path"] = None
                update_state["update_future"] = None
                update_future.set_result(result)
            else:
                update_state["in_progress"] = False
                update_state["updating_model_path"] = None
                update_state["update_future"] = None
                update_future.set_result(error)

    if error:
        return _json_error(error["error"], error["status"])
    return web.json_response(result)
```

评论区精华

该 PR 无 review 评论。

- 暂无高价值评论线程

风险与影响

- 风险：该 PR 修改了 Megatron 服务器的核心更新逻辑，增加了锁和 future 机制，可能引入死锁风险。但通过 asyncio.Lock 和 asyncio.shield 的使用，风险可控。测试覆盖不足，建议补充并发测试。
- 影响：影响范围局限于 Megatron 服务器的 update_from_disk 端点，不会影响其他模块或用户接口。变更提升了并发场景下的稳定性和响应效率。
- 风险标记：核心路径变更，缺少测试覆盖

关联脉络

- PR #2118 sync from internal: 此前修改了同文件 megatron_server.py，添加了 update_from_disk 功能，当前 PR 修复其并发问题。