PR #35925 完整报告

sgl-project/sglang

Make the scheduler track the published weight version

合并时间：2026-08-24 20:17

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35925

执行摘要

- 一句话：调度器新增权重版本跟踪，打通 HTTP 与换权重路径
- 推荐动作：值得精读，尤其是幂等版本记录（同版本 /None 短路）、通过声明式 _COMMUNICATOR_SPECS 注册新 RPC、以及换权重成功后统一回写的接入方式。从事在线权重更新、RL 数据归因或调度器内部状态观测的工程师应重点关注；同时可参考 issue 评论中 Codex 对 CI 失败的系统性 triage 方法。

功能与动机

提交信息明确指出：POST /update_weight_version 只写了 tokenizer 侧的 weight_version，而换权重路径从未回写，因此调度器自身记录停留在启动时的值，任何按调度器读取该值的功能（如 /server_info、权重版本观测）读到的是过期数据。该 PR 正是要修复这一状态不一致，并为同一系列中 per-token weight-version span 的正确归因提供真实数据源。

实现拆解

1. 在 python/sglang/srt/managers/io_struct.py 新增 UpdateWeightVersionReqOutput(BaseReq, kw_only=True) 空响应类型，与已有的 UpdateWeightVersionReqInput 配对，补齐 RPC 协议定义。
2. 在 python/sglang/srt/managers/scheduler.py 中：
   - 在 init_request_dispatcher 注册 (UpdateWeightVersionReqInput, self.handle_update_weight_version)；
   - 新增 handle_update_weight_version 调用 record_weight_version_change 并返回空响应；
   - 新增 record_weight_version_change 幂等方法：None 或与当前版本相同则直接返回，否则记录旧版本并通过 get_context().override("scheduler.weight_version", ...) 写入 serving 上下文，同时输出日志。这样调度器侧版本状态具备幂等更新能力，避免无意义的上下文写入。
3. 在 python/sglang/srt/managers/scheduler_components/weight_updater.py 新增 record_weight_version_after_update 辅助方法，并在 update_weights_from_disk、update_weights_from_distributed、update_weights_from_tensor、update_weights_from_ipc 的成功分支调用；失败分支不记录，防止把从未生效的版本标记到后续 token 上。

4. 在 python/sglang/srt/managers/tokenizer_control_mixin.py 的声明式 _COMMUNICATOR_SPECS 列表中新增 ("update_weight_version", UpdateWeightVersionReqOutput)，并实现 async update_weight_version：先 await update_weight_version_communicator(obj) 把新版本转发给调度器，再调用已有的 _update_weight_version_if_provided 更新 tokenizer 本地副本，保证两侧一致。
5. 在 python/sglang/srt/entrypoints/http_server.py 中，update_weight_version 端点由原来的 record_config_updates 本地记录改为 await _global_state.tokenizer_manager.update_weight_version(obj)，完成端到端打通。
6. 测试配套：新增 test/registered/unit/managers/test_scheduler_weight_version_tracking.py，用 serving/context stub 验证新版本采纳、同版本与 None 版本短路；用 updater stub 验证 disk 成功 / 失败、draft 失败、分布式成功等路径的版本记录行为。

关键文件：

- python/sglang/srt/managers/scheduler.py（模块 调度器；类别 source；类型 core-logic；符号 handle_update_weight_version, record_weight_version_change）：核心改动：注册并实现新的 RPC 处理 handle_update_weight_version，新增幂等的 record_weight_version_change，统一调度器侧版本状态。
- python/sglang/srt/managers/scheduler_components/weight_updater.py（模块 权重更新；类别 source；类型 core-logic；符号 record_weight_version_after_update, update_weights_from_disk, update_weights_from_distributed, update_weights_from_tensor）：四条换权重路径的成功分支都通过 record_weight_version_after_update 回写版本，是调度器版本能跟着实际换权重推进的关键。
- python/sglang/srt/managers/tokenizer_control_mixin.py（模块 控制转发；类别 source；类型 core-logic；符号 update_weight_version）：声明式通信器注册表中新增 update_weight_version，并提供 async 方法把新版本先转发给调度器再更新本地副本。
- python/sglang/srt/entrypoints/http_server.py（模块 HTTP 入口；类别 source；类型 entrypoint）：HTTP 入口从仅 record_config_updates 改为异步调用 tokenizer_manager.update_weight_version，完成端到端打通。
- python/sglang/srt/managers/io_struct.py（模块 协议结构；类别 source；类型 core-logic；符号 UpdateWeightVersionReqOutput）：新增 UpdateWeightVersionReqOutput 响应类型，补齐 RPC 协议的定义。
- test/registered/unit/managers/test_scheduler_weight_version_tracking.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestSchedulerRecordWeightVersionChange, TestRecordWeightVersionAfterUpdate, test_a_new_version_is_adopted, test_same_version_is_a_noop）：新增测试覆盖调度器版本记录的幂等与换权重后回写行为，是本次变更的主要质量保障。

关键符号：handle_update_weight_version, record_weight_version_change, record_weight_version_after_update, update_weight_version

关键源码片段

python/sglang/srt/managers/scheduler.py

核心改动：注册并实现新的 RPC 处理 handle_update_weight_version，新增幂等的 record_weight_version_change，统一调度器侧版本状态。

```
def handle_update_weight_version(
    self, recv_req: UpdateWeightVersionReqInput
) -> UpdateWeightVersionReqOutput:
    # 新的 RPC 入口：tokenizer 侧通过该消息把新版本号广播给调度器。
    # 幂等更新由 record_weight_version_change 保证。
    self.record_weight_version_change(new_version=recv_req.new_version)
    return UpdateWeightVersionReqOutput()

def record_weight_version_change(self, new_version: Optional[str]) -> None:
    # None 表示"本次更新不携带版本号"，视为 no-op；
    # 相同版本重复发布也直接短路，避免无意义的上下文写入。
    if new_version is None or new_version == get_serving().weight_version:
        return

    # 记录旧版本，再通过 context override 写回 serving 上下文。
    # 这里统一走 "scheduler.weight_version" 键，后续读取方（如
    # per-token weight-version spans）读到的都是调度器侧的最新版本。
    old_version = get_serving().weight_version
    get_context().override("scheduler.weight_version", weight_version=new_version)
    logger.info(f"Weight version changed. {old_version=} {new_version=}")
```

python/sglang/srt/managers/scheduler_components/weight_updater.py

四条换权重路径的成功分支都通过 record_weight_version_after_update 回写版本，是调度器版本能跟着实际换权重推进的关键。

```
def record_weight_version_after_update(self, weight_version: Optional[str]) -> None:
    # 所有换权重路径在成功后统一调用，把新版号交回调度器记录。
    # 失败路径不调用，避免记录一个从未生效的版本导致后续 token 被错误标记。
    self.scheduler.record_weight_version_change(new_version=weight_version)

def update_weights_from_disk(self, recv_req: UpdateWeightFromDiskReqInput):
    """In-place update of the weights from disk."""
    with self._observe_weight_load("disk"):
        success, message = self.tp_worker.update_weights_from_disk(recv_req)
        tp_success = success
        if success and self.draft_worker is not None:
            success, message = self.draft_worker.update_weights_from_disk(recv_req)
        if tp_success:
            self.flush_cache_after_weight_update(recv_req)
        if success:
            # 仅当 target 和 draft 都成功后记录版本，保证引擎状态一致。
            self.record_weight_version_after_update(recv_req.weight_version)
        else:
            logger.error(message)
        return UpdateWeightFromDiskReqOutput(
```

```
            success=success, message=message, num_paused_requests=0
        )
```

评论区精华

无 review 评论，但 issue 评论记录了两轮 Codex 自动 CI triage：

- CPU shard 3 的失败被判定为 frozen-base 不兼容（测试未 publish memory config bag），上游 commit 已修复，非本 op 回归；
- AWQ test_mmlu 以 0.8203125 低于 0.83 阈值失败，作者用 /rerun-test 在同一 head 上重跑同一命令通过，判定为瞬时准确率波动；
- 作者随后手动验证了本次新增测试文件 test_scheduler_weight_version_tracking.py 在最终 rebase 后 head 上 PASS。
- CI 失败排查与重跑结论 (testing): 确认 CI 失败均为环境 / 基线与瞬时准确率波动，与本次变更无关；新增测试已单独验证通过。

风险与影响

- 风险：
  1. scheduler.py 的 record_weight_version_change 通过 get_context().override 修改全局 serving 上下文，依赖 get_serving() 与 get_context() 在调用时可用；若并发调用或与其它 override 顺序冲突，可能干扰调度器状态。
  2. weight_updater.py 的 record_weight_version_after_update 直接访问 self.scheduler，若该字段为 None（默认值）会抛 AttributeError；当前构造链总是由 Scheduler 传入自身，但测试仅用 stub，未覆盖 scheduler=None 的风险路径。
  3. 新增 UpdateWeightVersionReqInput 分发与 UpdateWeightVersionReqOutput 协议类型属于新契约，滚动升级期间旧 scheduler 可能无法识别该消息。
  4. 换权重成功路径现在会写入调度器版本，若调用方传入的 weight_version 与实际权重内容不符，会污染版本标签（属于调用方责任）。 - 影响：影响面中等：新增一条 tokenizer 到 scheduler 的 RPC 消息；POST /update_weight_version 的行为从仅记录本地配置变为同时驱动调度器状态；disk/distributed/tensor/ipc 四条换权重路径在成功后自动推进调度器版本。对使用在线权重更新与 RL 归因的场景，调度器侧观测值将与实际生效版本一致，为 PR 35926 的 per-token weight-version spans 提供正确数据源。无性能与兼容性破坏。 - 风险标记：调度器全局状态变更，新增 RPC 协议契约，权重更新成功路径行为变化，依赖 serving context 可用性

关联脉络

- PR #35926 Report per-token weight-version spans in generation meta info: 同属权重版本观测链路，本 PR 保证调度器侧版本真实，35926 才能正确给每个 token 打上版本跨度标签；两者都改 scheduler.py 与 io_struct.py。
- PR #35927 Support gated launch to defer startup memory allocation: 同一作者 fzyzcjy 的 op 系列，共享调度器与 server 生命周期路径。

- PR #35929 Report the whole server's world size in the scheduler's internal state: 同一系列中调度器内部状态扩展，与本 PR 一样在 scheduler.py 上增强状态可见性。