PR #27193 完整报告

sgl-project/sglang

Replace skip_attn_backend_init with a batch-carried attention plan marker (+ staleness re-plan)

合并时间：2026-06-05 08:13

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27193

执行摘要

- 一句话：用 batch 携带的 attention plan marker 替换 skip_attn_backend_init
- 推荐动作：强烈建议阅读。PR 展示了如何用 batch 携带的状态替换控制耦合，以及如何通过 opt-in 的 plan record 安全地实现 staleness re-plan，是 speculative decoding 路径中一次重要的基础设施重构。设计思路（将断言从调用链远处转移到数据本身）值得其他类似场景借鉴。

功能与动机

skip_attn_backend_init 是一个控制耦合的布尔值，穿越约 6 层进行传递，存在三个结构性问题：它未验证调用者是否真的进行了预规划；同一个标志在 eager 路径和 cuda-graph 路径中含义不同；每个新的 forward 路径都必须记得传递它。本 PR 使用 batch 携带的规划状态替代调用者声明的标志，让 pre-planner 直接在 ForwardBatch 上记录规划事实，forward 路径通过查询 batch 而非信任从 6 层之外传来的布尔值来决定是否规划。

实现拆解

1. 定义 marker 数据结构：在 python/sglang/srt/model_executor/forward_batch_info.py 的 ForwardBatch 类中添加 forward_metadata_ready、forward_metadata_planned_bs、forward_metadata_planned_num_tokens、forward_metadata_replan_equivalent 四个字段，以及 mark_forward_metadata_ready() 和 needs_forward_metadata_init() 方法。
2. 在所有 pre-plan 站点标记 batch：遍历 EAGLE v1/v2、multi-layer v1/v2、frozen-KV MTP、plan-stream replay_prepare 等预规划位置，在规划操作后立即调用 forward_batch.mark_forward_metadata_ready()。特别修复了 cuda-graph capture 路径漏标的问题（review 中发现）。
3. 在判断点读取 marker：ModelRunner.forward_decode、forward_extend 和 cuda/NPU graph runner 的 replay 中的条件从 if not skip_attn_backend_init 改为 if forward_batch.needs_forward_metadata_init()。
4. 移除内部参数传递：从 _forward_raw、forward_decode、forward_extend、graph runner replay 以及所有 speculative worker 的函数签名中删除 skip_attn_backend_init。公共入口保留为可选参数，并添加 apply_deprecated_skip_attn_backend_init 映射到 marker，附带模块级别的 warn-once。

5. 引入 plan record 和 staleness 验证：mark_forward_metadata_ready 快照 batch_size 和 input_ids.shape[0]。needs_forward_metadata_init 在 replan_equivalent=True 且形状变化时触发重新规划。仅对等价规划站点开放 opt-in。
6. 修剪 trtllm-MLA 防御性 re-plan：原本的防御性 re-plan 被缩小为针对无法 opt-in 的路径的有保留回退，并附带移除条件。
7. 修复 TBO filter_batch 兼容性：在 two_batch_overlap.py 的 filter_batch 中显式重置子 batch 的 marker 字段，防止 completeness guard 崩溃。
8. 全面测试覆盖：新增 test_forward_metadata_plan_record.py（6 种语义）和 test_tbo_filter_batch_marker.py（回归），更新 attention 套件 mock 以断言预规划状态。

关键文件：

- python/sglang/srt/model_executor/forward_batch_info.py（模块 前向批次；类别 source；类型 data-contract；符号 mark_forward_metadata_ready, needs_forward_metadata_init, apply_deprecated_skip_attn_backend_init）：核心变更：在 ForwardBatch 中添加 attention plan marker 字段和处理方法，为移除旧的 skip 标志提供了数据结构基础。
- test/registered/unit/model_executor/test_forward_metadata_plan_record.py（模块 单元测试；类别 test；类型 test-coverage；符号 _make_batch, TestForwardMetadataPlanRecord, test_fresh_batch_needs_planning, test_mark_records_shapes_and_skips_planning）：新测试文件，全面覆盖 marker 契约，包括新鲜 batch 需要规划、标记后跳过、形状变化时 opt-in 重新规划等场景，确保重构正确性。
- test/registered/unit/batch_overlap/test_tbo_filter_batch_marker.py（模块 TBO 测试；类别 test；类型 test-coverage；符号 _make_target_verify_batch, _filter, TestTboFilterBatchMarker, test_filter_batch_resets_plan_marker_on_children）：回归测试：确保 TBO filter_batch 重置子 batch 的 marker，防止 completeness guard 崩溃和计划状态泄漏。
- python/sglang/srt/speculative/frozen_kv_mtp_worker.py（模块 推测解码；类别 source；类型 core-logic；符号 draft_forward）：修改了 draft_forward 方法，将 skip_attn_backend_init 参数替换为基于 marker 的判断，并在规划点添加了标记调用。
- python/sglang/srt/model_executor/model_runner.py（模块 模型执行器；类别 source；类型 core-logic；符号 forward, forward_decode, forward_extend, _forward_raw）：移除内部所有 skip_attn_backend_init 参数传递，在 forward() 入口添加 deprecated shim，在 forward_decode/extend 中切换为 marker 判断。
- python/sglang/srt/speculative/eagle_worker_v2.py（模块 推测解码；类别 source；类型 core-logic；符号 draft_forward, verify）：移除线程中传递的 skip_attn_backend_init，在相应位置添加 marker 标记和基于 marker 的判断。

关键符号：mark_forward_metadata_ready, needs_forward_metadata_init, apply_deprecated_skip_attn_backend_init, draft_forward, forward_decode, forward_extend

关键源码片段

test/registered/unit/model_executor/test_forward_metadata_plan_record.py

新测试文件，全面覆盖 marker 契约，包括新鲜 batch 需要规划、标记后跳过、形状变化时 opt-in 重新规划等场景，确保重构正确性。

```
class TestForwardMetadataPlanRecord(unittest.TestCase):
    def test_reshape_with_opt_in_replans(self):
        # 标记为 replan_equivalent，形状变化后应重新规划
        fb = _make_batch(bs=2, num_tokens=2)
        fb.mark_forward_metadata_ready(replan_equivalent=True)
        self.assertFalse(fb.needs_forward_metadata_init())

        # 模拟 DP padding 导致的 batch_size 增长
        fb.batch_size = 4
        self.assertTrue(fb.needs_forward_metadata_init()) # 形状变化触发重新规划

        # 恢复原 size 但 token 数变化也触发重新规划
        fb.batch_size = 2
        fb.input_ids = torch.zeros(6, dtype=torch.long)
        self.assertTrue(fb.needs_forward_metadata_init())

    def test_reshape_without_opt_in_keeps_skipping(self):
        # 未声明 replan_equivalent 时，即使形状变化也保持跳过状态
        fb = _make_batch(bs=2)
        fb.mark_forward_metadata_ready() # 默认为 False
        fb.batch_size = 4
        self.assertFalse(fb.needs_forward_metadata_init()) # 不重新规划

    def test_remark_re_records_padded_shapes(self):
        # 重新标记会更新形状快照
        fb = _make_batch(bs=2)
        fb.mark_forward_metadata_ready(replan_equivalent=True)
        fb.batch_size = 4
        self.assertTrue(fb.needs_forward_metadata_init())
        fb.mark_forward_metadata_ready(replan_equivalent=True)
        self.assertFalse(fb.needs_forward_metadata_init())
        self.assertEqual(fb.forward_metadata_planned_bs, 4)
```

评论区精华

Codex 机器人审查时发现一个 P1 bug：在 eagle_worker.py 的 CUDA 图捕获路径中，run_once 调用 draft_forward(forward_batch) 不再传递 skip_attn_backend_init=True，但捕获 batch 是静态构造且从未标记 forward_metadata_ready，导致 draft_forward 内的多步 forward 会重新规划 attention metadata。作者在 90065bf57 修复——在捕获规划后立即标记 batch；并在 4fd4a589d 向 attention 套件 mock 添加断言，确保此类回归被捕获。

- EAGLE draft capture batch 未标记导致 re-plan (correctness): 作者在 90065bf57 中在捕获规划后立即添加了 mark_forward_metadata_ready() 调用，并在 4fd4a589d 中向注意测试套件的模拟 draft runner 中添加了断言，确保此类回归未来能被捕获。

风险与影响

- 风险：
  - 回归风险：pre-planner 遗忘标记会导致 forward 路径静默重新规划，可能产生不正确的输出。新增加的断言和测试套件能暴露此类问题。
  - 设计风险：replan_equivalent 的 opt-in 机制依赖开发者的正确选择；错误地将应重新规划的站点设为 False 会导致过时 metadata。但设计上已通过代码审查确保等价性，且 trtllm-MLA 保留了有条件的防御性回退。
  - TBO 兼容性：filter_batch 的 completeness guard 对任何非 None 的默认字段敏感，PR 已修复当前字段，但未来添加类似字段时需注意重置。
  - 影响范围：涉及所有 speculative decoding 路径中的 attention 规划，但变更均在可控范围内，测试通过。
- 影响：
  - 用户影响：无直接可见功能变化，但提升了 speculative decoding 场景下 attention metadata 规划的健壮性，消除了潜在的数据竞争和错误规划问题。
  - 系统影响：重构了核心的 attention 规划数据流，后续添加新 forward 路径时不再需要传递 skip_attn_backend_init 参数，降低了耦合。
  - 团队影响：成功移除了 tp_worker.py 中的 FIXME(lsyin)；trtllm-MLA 中的防御性 re-plan 也由开放 TODO 变为有明确清除条件的回退。需要关注 marker 的正确标记和 replan_equivalent 的精确使用。
  - 风险标记：核心路径变更，多步 draft 依赖，DP padding 交互，TBO 兼容性

关联脉络

- 暂无明显关联 PR