PR #26735 完整报告

sgl-project/sglang

[refactor] init_forward_metadata 3-method ABC + side-channel removal + ForwardMetadata type rename

合并时间：2026-06-03 01:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26735

执行摘要

- 一句话：引入 3-method ABC 重构注意力初始化契约，移除 DSV4 side channel
- 推荐动作：建议精读 base_attn_backend.py 的 ABC 定义和 DSV4 的 init_forward_metadata_in_graph 实现，体会如何通过明确分层消除隐式 side channel。对于正在开发或维护注意力协议的工程师，此 PR 的设计决策（in_capture 标志、in-graph 专属方法、SimpleNamespace replay 视图）是值得参考的模式。但阅读时应注意 PR 经历了大量提交迭代，早期实现与最终合并版本差异较大，建议直接阅读 HEAD。

功能与动机

PR 完成 attention-init 重构的最后一环。前置 PR #26665 统一了 capture/replay 主流程，本 PR 在此基础上引入正式的三方法 ABC 契约，让外部后端可以直接消费统一流程而无需切换 API；同时移除 DSV4 遗留的 _replay_forward_batch side channel，消除隐式耦合。PR body 明确定义了每个方法的职责边界：init_forward_metadata — eager entry；init_forward_metadata_out_graph — per-iter metadata prep outside graph capture；init_forward_metadata_in_graph — graph-recordable static-shape GPU op inside graph capture。

实现拆解

1. 定义 ABC 契约：在 base_attn_backend.py 中新增 init_forward_metadata_out_graph(fb, in_capture=False) 和 init_forward_metadata_in_graph(fb) 抽象方法，init_forward_metadata 保留为 eager 入口，默认实现包装 _out_graph + _in_graph。同时从 ABC 中移除旧的 init_forward_metadata_capture_cuda_graph 和 init_forward_metadata_replay_cuda_graph。
2. 后端分阶段迁移：按后端逐个添加新方法（先委托旧逻辑），然后将真正体迁入 init_forward_metadata_out_graph，最后删除旧 capture/replay 覆盖。涉及 FlashInfer、FlashAttention、Triton、TBO、MLBA、DSV4 等 14+ 个注意力后端及 MultiStep 包装器。
3. DSV4 side channel 移除 + in-graph 升级：移除 DeepseekV4AttnBackend 和 DeepseekV4HipRadixBackend 中的 self._replay_forward_batch 字段。新增 init_forward_metadata_in_graph，将原本分散在 forward()、forward_core_compressor 等处的 _maybe_upgrade_forward_metadata 调用集中为显式的图录制步骤。

4. 图运行器适配：CudaGraphRunner、PiecewiseCudaGraphRunner、BreakableCudaGraphRunner 和 EagleDraftCudaGraphRunner 全部迁移到新 API，在 capture 块内调用 init_forward_metadata_in_graph。
5. 测试与清理：迁移现有测试（如 test_cuda_graph_decode.py、test_tbo.py）到新 API，删除多余的 _maybe_upgrade_forward_metadata 调用和安全网。移除了不再需要的 pre_replay 适配钩子和 TBO 遗留的 _init_forward_metadata_cuda_graph_children 辅助方法。

关键文件：

- python/sglang/srt/layers/attention/deepseek_v4_backend.py（模块 DSV4 后端；类别 source；类型 dependency-wiring；符号 init_forward_metadata, init_forward_metadata_in_graph, init_cuda_graph_state, init_forward_metadata_capture_cuda_graph）：DSV4 后端是 side channel 移除的主要受益者，展示了 init_forward_metadata_in_graph 的核心用法：将 Raw→Full 元数据升级录制到 CUDA graph 中。同时删除了大量重复的 _maybe_upgrade_forward_metadata 安全网调用。
- python/sglang/srt/layers/attention/tbo_backend.py（模块 TBO 后端；类别 source；类型 dependency-wiring；符号 init_forward_metadata_out_graph, _dispatch_children_from_replay_view, init_forward_metadata_in_graph, init_forward_metadata_capture_cuda_graph）：TBO 后端的 init_forward_metadata_out_graph 展示了 replay 路径中如何通过 SimpleNamespace 视图拆分 children，替代了旧的 _init_forward_metadata_cuda_graph_children 复杂逻辑。
- python/sglang/srt/layers/attention/flashattention_backend.py（模块 FlashAttention；类别 source；类型 dependency-wiring；符号 init_forward_metadata_out_graph, init_forward_metadata_capture_cuda_graph, init_forward_metadata_replay_cuda_graph, _apply_cuda_graph_metadata）：FlashAttention 后端完整展示了新契约的迁移模式：将 capture 和 replay 路径聚合到 init_forward_metadata_out_graph 中，通过 in_capture 标志分流，并提取了 _apply_cuda_graph_metadata 辅助方法。

关键符号：AttentionBackend.init_forward_metadata, AttentionBackend.init_forward_metadata_out_graph, AttentionBackend.init_forward_metadata_in_graph, DeepseekV4AttnBackend.init_forward_metadata_in_graph, TboAttnBackend._dispatch_children_from_replay_view, FlashAttentionBackend._apply_cuda_graph_metadata, build_replay_fb_view, build_inner_fb_view

关键源码片段

python/sglang/srt/layers/attention/deepseek_v4_backend.py

DSV4 后端是 side channel 移除的主要受益者，展示了 init_forward_metadata_in_graph 的核心用法：将 Raw→Full 元数据升级录制到 CUDA graph 中。同时删除了大量重复的 _maybe_upgrade_forward_metadata 安全网调用。

```
def init_forward_metadata_in_graph(self, forward_batch: ForwardBatch) -> None:
    # 在 CUDA graph 内执行 Raw→Full 升级，使压缩 / 核心注意力 / 索引器的物化录制到 graph 中
    # 当 PREP_IN_CUDA_GRAPH=0 时，metadata 已经是 Full，此方法无操作
    if isinstance(self.forward_metadata, DSV4RawVerifyMetadata):
        self.forward_metadata = self.make_forward_metadata_from_raw_verify(
            raw_metadata=self.forward_metadata,
        )
```

python/sglang/srt/layers/attention/tbo_backend.py

TBO 后端的 init_forward_metadata_out_graph 展示了 replay 路径中如何通过 SimpleNamespace 视图拆分 children，替代了旧的 _init_forward_metadata_cuda_graph_children 复杂逻辑。

```
def init_forward_metadata_out_graph(
    self,
    forward_batch: "ForwardBatch",
    in_capture: bool = False,
):
    self.primary.init_forward_metadata_out_graph(
        forward_batch=forward_batch, in_capture=in_capture
    )
    tbo_children = getattr(forward_batch, "tbo_children", None)
    if tbo_children is not None:
        # 正常 eager/capture 路径：直接分发到 children
        for child, forward_batch_child in zip(
            self.children, tbo_children, strict=True
        ):
            if forward_batch_child.batch_size > 0:
                child.init_forward_metadata_out_graph(
                    forward_batch=forward_batch_child, in_capture=in_capture
                )
        return
    if in_capture:
        return
    # Replay 路径：build_replay_fb_view 返回 SimpleNamespace，
    # 缺少 tbo_children，需根据 padded 缓冲区手动拆分
    self._dispatch_children_from_replay_view(forward_batch)

def _dispatch_children_from_replay_view(self, fb_view) -> None:
    bs = fb_view.batch_size
    forward_mode = fb_view.forward_mode
    spec_info = fb_view.spec_info
    token_num_per_seq = two_batch_overlap.get_token_num_per_seq(
        forward_mode=forward_mode, spec_info=spec_info
    )
    num_tokens = bs * token_num_per_seq
    (
        tbo_split_seq_index,
        tbo_split_token_index,
```

```
    ) = two_batch_overlap.compute_split_indices_for_cuda_graph_replay(
        forward_mode=forward_mode,
        cuda_graph_num_tokens=num_tokens,
        spec_info=spec_info,
    )
    bs_left = tbo_split_seq_index
    bs_right = bs - bs_left
    # 分配左右两个 child 对应的批次切片
    for child, child_bs, seq_slice, tok_slice in (
        (self.children[0], bs_left, slice(None, tbo_split_seq_index), slice(None, tbo_split_token_
        index)),
        (self.children[1], bs_right, slice(tbo_split_seq_index, None), slice(tbo_split_token_index,
        None)),
    ):
        if child_bs == 0:
            continue
        child_fb_view = _build_tbo_child_replay_fb_view(
            fb_view,
            child_bs=child_bs,
            seq_slice=seq_slice,
            tok_slice=tok_slice,
            token_num_per_seq=token_num_per_seq,
        )
        child.init_forward_metadata_out_graph(
            forward_batch=child_fb_view, in_capture=False
        )
```

python/sglang/srt/layers/attention/flashattention_backend.py

FlashAttention 后端完整展示了新契约的迁移模式：将 capture 和 replay 路径聚合到 init_forward_metadata_out_graph 中，通过 in_capture 标志分流，并提取了 _apply_cuda_graph_metadata 辅助方法。

```
  def init_forward_metadata_out_graph(
      self,
      forward_batch: ForwardBatch,
      in_capture: bool = False,
  ):
      bs = forward_batch.batch_size
      req_pool_indices = forward_batch.req_pool_indices
      seq_lens = forward_batch.seq_lens
      encoder_lens = forward_batch.encoder_lens
      forward_mode = forward_batch.forward_mode
      spec_info = forward_batch.spec_info
      out_cache_loc = getattr(forward_batch, "out_cache_loc", None)

      if in_capture:
          num_tokens = forward_batch.positions.numel()
          seq_lens_cpu = seq_lens.cpu()
          self._bind_metadata_buffers(
```

```
            bs, num_tokens, encoder_lens, forward_mode, spec_info, seq_lens.device,
        )
        # 处理 topk>1 的特殊早期返回（replay 时才能填充完整数据）
        if forward_mode.is_decode_or_idle() and spec_info is not None and self.topk > 1:
            self.forward_metadata = self.draft_decode_metadata_topk_normal[bs]
            self.forward_metadata_spec_decode_expand = self.draft_decode_metadata_topk_
            expand[bs]
            return
        if forward_mode.is_target_verify() and self.topk > 1:
            self.forward_metadata = self.target_verify_metadata_topk_normal[bs]
            self.forward_metadata_spec_decode_expand = self.target_verify_metadata_topk_
            expand[bs]
            return
        # 调用共享的元数据填充逻辑
        self._apply_cuda_graph_metadata(
            bs=bs, req_pool_indices=req_pool_indices, seq_lens=seq_lens,
            seq_lens_sum=None, encoder_lens=encoder_lens,
            forward_mode=forward_mode, spec_info=spec_info,
            seq_lens_cpu=seq_lens_cpu, out_cache_loc=out_cache_loc,
        )
        # 后处理：local attention 和 scheduler metadata 需要捕获时的切片大小
        if forward_mode.is_decode_or_idle() and spec_info is None:
            metadata = self.decode_cuda_graph_metadata[bs]
            self._maybe_update_local_attn_metadata_for_capture(metadata, bs)
            # ... scheduler metadata 计算
    else:
        # replay 路径：使用 forward_batch 中的运行时数据
        self._apply_cuda_graph_metadata(
            bs=bs, req_pool_indices=req_pool_indices, seq_lens=seq_lens,
            seq_lens_sum=forward_batch.seq_lens_sum, encoder_lens=encoder_lens,
            forward_mode=forward_mode, spec_info=spec_info,
            seq_lens_cpu=forward_batch.seq_lens_cpu, out_cache_loc=out_cache_loc,
        )
```

评论区精华

Codex 自动审查提出了三个 P2 级别问题，均已在后续提交中修复：

- TBO replay 缺少 tbo_children：新 init_forward_metadata_out_graph 直接读取 forward_batch.tbo_children，但 SimpleNamespace replay 视图未定义此属性，导致 AttributeError。修复：使用 getattr 保护空安全，并在 replay 路径中调用 _dispatch_children_from_replay_view 手动拆分。
- DSV4 MultiStep actual_forward_mode 被覆盖：包装器构建 inner_fb 时错误地将 actual_forward_mode 设置为 forward_batch.forward_mode，使得 IDLE 模式被当作 DECODE，可能使用无效的 out_cache_loc。修复：转发 getattr(forward_batch, 'actual_forward_mode', forward_batch.forward_mode)。
- HIP Radix 双胞胎同样问题：deepseek_v4_backend_hip_radix.py 中存在相同的 actual_forward_mode 覆盖问题，一并修复。

- TBO replay 路径缺少 tbo_children (correctness): 已修复：使用 getattr 保护空安全，并在 replay 路径中调用 _dispatch_children_from_replay_view 手动计算拆分。
- DSV4 MultiStep actual_forward_mode 被覆盖 (correctness): 已修复：转发 getattr(forward_batch, 'actual_forward_mode', forward_batch.forward_mode)。
- HIP Radix 同款 actual_forward_mode 覆盖 (correctness): 已修复：同样转发 actual_forward_mode。

风险与影响

- 风险：所有注意力后端的核心初始化路径均被修改，回归风险较高。关键风险点包括：① DSV4 的图内 Raw→Full 升级路径如果遗漏，将导致 c4/c128 压缩未执行；② TBO replay 路径的 children 拆分逻辑若出错，会破坏 two-batch-overlap 功能；③ MultiStep 包装器中 in_capture 标志传递错误可能导致 capture/replay 使用不当的元数据；④ 非 DSV4 后端新增的 isinstance 检查虽为 no-op，但仍引入了极小的 Python 解释开销。CI 覆盖了 unit test 和部分集成测试，但未覆盖所有后端组合。
- 影响：对注意力后端开发者有直接影响：需要实现三个新的 ABC 方法，但大多数后端可通过默认行为获得兼容。对最终用户无行为变化，所有功能等价。但此 PR 为未来支持新后端（如 PD 分解中的乐观预填充）和移除遗留代码铺平了道路，是 attention 子系统架构演进的关键一步。对团队而言，需要 review 所有后端的适配正确性，后续维护者应遵循新的契约模式。
- 风险标记：核心路径变更，DSV4 特殊路径，多个后端同步适配，图录制路径安全

关联脉络

- PR #26665 [refactor] Rounds 2-6 capture/replay body unification: 前置 PR，统一了 capture/replay 主体流程，本 PR 在其基础上引入 ABC 契约并移除 side channel。