PR #33116 完整报告

sgl-project/sglang

[Inkling] Hold the short-conv per-step state on one metadata struct

合并时间：2026-08-01 09:11

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33116

执行摘要

- 一句话：Inkling 短卷积状态收敛到单 metadata，移除池级缓存
- 推荐动作：值得精读，尤其适合关注 attention backend metadata 生命周期与 CUDA graph buffer 管理的工程师。三个设计决策值得借鉴：(1) cache ownership 应该由调用方而非池来承担，且用 AST 扫描与微基准论证移除缓存的 blast radius；(2) per-step metadata 采用可变 msgspec.Struct、由 prep 就地填充，与 FlashAttentionMetadata 对齐；(3) 用构造性等价（at_layer_idx(L).conv[s] 与 conv[s][L] 等价）配合 bit-identical 模型校验，而不是用 mock 计数测试去钉死 wiring。

功能与动机

PR body 明确这是 #33023 的 follow-up，目标是清理 per-step conv state 的位置：_layer_cache_by_id 缓存是“added for Inkling (the only model asking for the same layer's views four times per step) and belongs to the caller, not the pool”；后端每次读取都“rebuilt a fresh InklingShortConvMetadata on every read — 4 convs x 42 layers = 168 allocations per step”；而 mamba2_layer_cache 返回的 State bundle“only ever existed to amortize a bundle nobody wanted bundled”。三个问题共同说明：池不应替调用方持有进程级缓存，每步状态应放在 metadata 中，每个 conv 只需索引自己那一个 stream 的张量。

实现拆解

1. 移除 MambaPool 的进程级层缓存（python/sglang/srt/mem_cache/memory_pool.py）：mamba2_layer_cache 从带 _layer_cache_by_id memo 的版本改回直接 self.mamba_cache.at_layer_idx(layer_id)，并删除对应的 cached_property。作者先做 AST 扫描确认所有 mamba2_layer_cache 调用方（GDN、KDA、lightning、mamba2、generic short-conv）都在 forward* / conv_state_metadata 内、没有进入 init_forward_metadata* hook，因此 CUDA graph 重放路径不会碰到重建视图的成本。同时新增 HybridReqToTokenPool.mamba2_layer_index：把 HiCache 的 layer_transfer_counter.wait_until 屏障与 mamba_map 层映射封装为“取池内索引”这一原子操作，mamba2_layer_cache 改为组合它，使其他模型的 bundle API 字节级不变。
2. 每步单个 metadata 对象（python/sglang/srt/layers/attention/linear/inkling_sconv_backend.py）：InklingShortConvMetadata 从 NamedTuple 改为 msgspec.Struct（遵循仓库 no-dataclasses 规则），字段只剩 step 级张量，不再携带 layer_cache。__init__ 中创

建 self.sconv_metadata = InklingShortConvMetadata()，_reset_step_state 每步重建；_refresh_sconv_metadata / _refresh_decode_metadata / _refresh_extend_metadata /_refresh_track_conv_indices 把结果就地写入 metadata 字段；conv_state_metadata 因此退化为纯读取。这与 FlashAttentionMetadata 的模式一致：graph-static 缓冲区留在 backend，已解析的视图放在 metadata。

3. 直接索引 conv stream（inkling_sconv_backend.py + sconv.py）：backend 新增 sconv_state(layer_id, stream) 与 sconv_intermediate_window(layer_id, stream)，通过 mamba2_layer_index(layer_id) 拿到池内层号后返回 self._mamba_cache.conv[stream][pool_layer] 单切片。ShortConvolution._sconv_cache(meta) 改为无参 _sconv_cache()，_save_intermediate_conv_windows(cache, ...) 也改为经 _intermediate_window() 取流。同一提交还把这组访问器从共享的 ShortConvHybridAttnBackend 移到 InklingShortConvHybridAttnBackend，避免 ZAYA1 / LFM2 等 wrapper 暴露会 AttributeError 的方法。等价性论证：at_layer_idx(L).conv[s] 与 conv[s][L] 在 60 个 (layer, stream) 对上断言 data_ptr、shape、stride 完全一致。
4. 测试与验证配套：删除 test/registered/unit/models/test_inkling_sconv_metadata_once.py（352 行）。它用 mock pool 断言 mamba2_layer_cache / gather 调用次数，约束的是“prep 如何接线”而非“模型计算什么”，在缓存移除后失去意义。替代验证是模型级：Inkling tiny 8 层 7/7 bit-identical（greedy、batched/chunked prefill、prefix-cache 复用、输出 logprobs、16 路并发），Inkling-Small TP8+MTP 的 AIME26 95.83% +/- 1.67% 与基线一致，decode bs=32 吞吐 20259 vs 20112 tok/s。

关键文件：

- python/sglang/srt/layers/attention/linear/inkling_sconv_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 InklingShortConvMetadata, sconv_state, sconv_intermediate_window, conv_state_metadata）：核心变更文件：每步 metadata 由 NamedTuple 改为 msgspec.Struct，后端持有并就地填充；新增 sconv_state / sconv_intermediate_window 直接索引池张量，并把访问器从共享 wrapper 移到 Inkling wrapper。
- python/sglang/srt/mem_cache/memory_pool.py（模块 内存池；类别 source；类型 core-logic；符号 _layer_cache_by_id, mamba2_layer_cache, mamba2_layer_index）：移除仅为 Inkling 引入的进程级 _layer_cache_by_id 缓存，恢复 mamba2_layer_cache 为直接 at_layer_idx；新增 HybridReqToTokenPool.mamba2_layer_index 承载 HiCache barrier 与层映射。
- python/sglang/srt/models/inkling_common/sconv.py（模块 模型模块；类别 source；类型 data-contract；符号 _sconv_cache, _intermediate_window, _save_intermediate_conv_windows）：ShortConvolution 从 metadata 携带的 layer_cache 改为经 backend 的 sconv_state / sconv_intermediate_window 直接取本流张量，解耦模型与池 bundle 结构。
- test/registered/unit/models/test_inkling_sconv_metadata_once.py（模块 测试；类别 test；类型 deletion；符号 _MockMambaPool, mamba2_layer_cache, _MockReqToTokenPool, get_mamba_indices）：删除 352 行 wiring 单测：它用 mock

pool 断言 mamba2_layer_cache / gather 调用次数，约束的是 prep 接线方式而非模型计算结果，在缓存移除后失去意义。

关键符号：mamba2_layer_index, mamba2_layer_cache, sconv_state, sconv_intermediate_window, conv_state_metadata, _reset_step_state, _sconv_cache, _intermediate_window

关键源码片段

python/sglang/srt/layers/attention/linear/inkling_sconv_backend.py

核心变更文件：每步 metadata 由 NamedTuple 改为 msgspec.Struct，后端持有并就地填充；新增 sconv_state / sconv_intermediate_window 直接索引池张量，并把访问器从共享 wrapper 移到 Inkling wrapper。

```
# 每 step 单例：init_forward_metadata 阶段填充，conv 层只读同一对象，
# 避免 4 convs x 42 layers = 168 次 metadata 重建。
class InklingShortConvMetadata(msgspec.Struct):
    cache_indices: Optional[torch.Tensor] = None # per-request slot ids, int32
    query_start_loc: Optional[torch.Tensor] = None # cu-seqlens, int32
    has_initial_state: Optional[torch.Tensor] = None # resumes a cached prefix
    precomputed: Optional[SconvExtendMetadata | SconvDecodeMetadata] = None
    # [B, conv_kernel - 1] 输入位置，其 conv 窗口喂给 prefix cache；仅 extend 且 tracking 开启。
    track_conv_indices: Optional[torch.Tensor] = None


class InklingShortConvAttnBackend(ShortConvAttnBackend):
    def __init__(self, model_runner: ModelRunner):
        super().__init__(model_runner)
        # Pool-wide，构造期绑定：conv[stream] 是 [n_layers, n_slots, conv_kernel - 1, conv_dim]。
        self._mamba_cache = self.req_to_token_pool.mamba_pool.mamba_cache
        # 每步单个 metadata，prep 阶段就地填充，graph 路径上所有 tensor 都在 static buffer。
        self.sconv_metadata = InklingShortConvMetadata()
        self._alloc_graph_buffers()

    def conv_state_metadata(self, layer_id: int, forward_batch: ForwardBatch):
        # The step's metadata: resolved once during prep, so this is a pure read.
        del layer_id, forward_batch
        return self.sconv_metadata

    def sconv_state(self, *, layer_id: int, stream: int) -> torch.Tensor:
        # layer_id 的某一 SconvType 流的 conv 状态，单切片直接取池张量，
        # 不再经过 13-slice 的 per-layer State bundle。
        pool_layer = self.req_to_token_pool.mamba2_layer_index(layer_id)
        return self._mamba_cache.conv[stream][pool_layer]

    def sconv_intermediate_window(self, *, layer_id: int, stream: int) -> torch.Tensor:
        # 每个 draft token 的 conv 窗口，TARGET_VERIFY 专用，同样按 stream 单切片。
        pool_layer = self.req_to_token_pool.mamba2_layer_index(layer_id)
        return self._mamba_cache.intermediate_conv_window[stream][pool_layer]
```

python/sglang/srt/mem_cache/memory_pool.py

移除仅为 Inkling 引入的进程级 _layer_cache_by_id 缓存，恢复 mamba2_layer_cache 为直接 at_layer_idx；新增 HybridReqToTokenPool.mamba2_layer_index 承载 HiCache barrier 与层映射。

```
class MambaPool:
    def mamba2_layer_cache(self, layer_id: int):
        # 每层按需重建视图；池张量构造后不移动，切片本身 pool-stable，
        # 因此在 forward* / conv_state_metadata 调用点重建是安全的（约 6us/ 次）。
        return self.mamba_cache.at_layer_idx(layer_id)


class HybridReqToTokenPool:
    def mamba2_layer_index(self, layer_id: int) -> int:
        # Pool-side index of layer_id's state, gated on its HiCache transfer.
        # 对于只想要单个状态张量的调用方（Inkling sconv）：直接索引池张量，
        # 而不是取覆盖所有字段的 State bundle。
        assert layer_id in self.mamba_map
        if self.layer_transfer_counter is not None:
            # HiCache 层传输屏障：确保该层状态已到达本 worker。
            self.layer_transfer_counter.wait_until(layer_id - self.start_layer)
        return self.mamba_map[layer_id]

    def mamba2_layer_cache(self, layer_id: int):
        # 组合出新访问器，其他模型（GDN/KDA/mamba2 等）的 bundle API 保持不变。
        return self.mamba_pool.mamba2_layer_cache(self.mamba2_layer_index(layer_id))
```

python/sglang/srt/models/inkling_common/sconv.py

ShortConvolution 从 metadata 携带的 layer_cache 改为经 backend 的 sconv_state / sconv_intermediate_window 直接取本流张量，解耦模型与池 bundle 结构。

```
class ShortConvolution(nn.Module):
    def _conv_state(self, forward_batch: ForwardBatch):
        # The step's conv-state metadata, resolved once by the attention backend.
        return get_attn_backend().conv_state_metadata(self.layer_id, forward_batch)

    def _sconv_cache(self) -> torch.Tensor:
        # This module's own conv-state stream for this layer.
        # 每个 ShortConvolution 只需要一个 stream 的张量；直接索引
        # conv[stream][layer]，避免构建 13-slice 的 State bundle。
        return get_attn_backend().sconv_state(
            layer_id=self.layer_id, stream=self.sconv_type.value
        )

    def _intermediate_window(self) -> torch.Tensor:
        # This module's per-draft-token conv windows. TARGET_VERIFY only.
        return get_attn_backend().sconv_intermediate_window(
            layer_id=self.layer_id, stream=self.sconv_type.value
```

)

评论区精华

该 PR 没有 review 评论，也没有 review 线程；issue 侧只有流程指令：/rerun-test test_inkling.py 在 1-gpu-h100 上通过，以及 /tag-and-rerun-ci。真正的技术论证集中在 PR body 中：对图重放路径为何不会触达重建视图的 AST 扫描论证、Falcon-H1-1.5B 的 A/B 基准（~0.5% 中位数差异，落在基线 1.4% 波动内）、以及 60 组 (layer, stream) 的构造性等价断言。需要注意 CI 槽位显示 Latest PR Test (Base) 与 (Extra) 的最后一次运行为失败 / 中止，PR 已合并，建议复核最终 CI 状态。

- 暂无高价值评论线程

风险与影响

- 风险：跨模型回归风险（memory_pool.py）：移除缓存后 GDN/KDA/lightning/mamba2/generic short-conv 每层每次 forward 会重建 State view（约 6us/ 次）。作者靠 AST 扫描论证不在 init_forward_metadata* hook 中，但这是静态分析而非全模型运行覆盖；对非 Inkling 模型，本 PR 没有对应 bit-identical 的 A/B 验证。 CUDA graph 地址稳定性（inkling_sconv_backend.py）：sconv_state 返回的池张量 slice 是 pool-stable 的，sconv_metadata 内张量都来自 graph-static 缓冲区；风险在于若未来某条新路径在捕获阶段调用 mamba2_layer_index（内含 wait_until），会在 graph 内引入 host 同步。当前靠调用点约束规避。 测试覆盖下降：删除的 metadata-once 测试同时覆盖了“每步重新解析、不跨步泄漏”和“init_cuda_graph_state 后地址稳定”等语义；现在这些只能靠模型级 E2E（test_inkling.py）与手工验证兜底，回归定位成本上升。 CI 状态不确定性：最后可见的 CI run 为失败 / 未通过，需确认合并前的绿色状态。
- 影响：对 Inkling 用户：无行为差异与数值变化；*eager decode* 每步 Python 侧开销下降约 570us（42 层、4 conv/ 层配置），图路径吞吐不变，*decode* bs=32 观测到约 +0.7%。对 MambaPool 的其它消费方：API 保持兼容（mamba2_layer_cache 签名与返回不变），但每层一次多约 6us 的视图重建；按 PR 的测量，在 28ms 步长下约为 0.14ms/forward（24 层），可忽略。 对代码库结构：确立“池负责存储、backend 负责每步解析、metadata 引用后端 buffer”的分层；msgspec.Struct 的每步单例模式为未来其他注意力后端的 metadata 迁移提供参照。 对团队：删除测试减少了 CI 矩阵中一个 CUDA 单测，但依赖 E2E 覆盖；评审与维护者需要接受“构造性等价 + 模型级校验”作为该类重构的验证标准。
- 风险标记：MambaPool 缓存移除波及多模型，AST 扫描替代运行时验证，删除 wiring 单测覆盖，CI 最终状态需复核，HiCache 屏障靠近图路径

关联脉络

- PR #33023 feat(inkling): migrate short convs onto the ShortConv attention backend: PR body 明确声明本 PR 是其后置清理；本次移除的 _layer_cache_by_id 缓存与删除的 metadata-once 测试均源自 #33023 的引入。
- PR #33131 feat(cookbook): add DGX Spark support for Inkling-Small: 同属 Inkling 模型支持链路的持续演进，且本 PR 的 TP8+MTP 验证即为 Inkling-Small 配置，后续部署文

档可交叉引用。