PR #24870 完整报告

sgl-project/sglang

Support NextN = 2/4 in DSV32

合并时间：2026-06-03 10:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24870

执行摘要

- 一句话：支持 DSV32 中 NextN=2/4 利用 deep-gemm 原生路径提升 MTP 性能
- 推荐动作：建议仔细阅读 _build_paged_mqa_schedule_2d_ctx_lens 和 _get_topk_paged 中的条件判断，理解原生路径与回退路径的设计取舍。同时关注后续 revert 或修复 PR 中对测试失败的处理。

功能与动机

DeepGEMM 在 PR #316 中去掉了 multicast，使得任意 next_n ⩾ 1 可在 SM100 上执行。此 PR 旨在利用该能力，消除原先仅支持 next_n=1 的限制，提升 MTP（Multi-Token Prediction）场景下的解码吞吐。

实现拆解

1. 元数据扩展：在 DSAMetadata 和 DSAIndexerMetadata 中新增 paged_mqa_ctx_lens_2d 字段，用于缓存预计算的上下文长度二维张量，避免每层重复构建。
2. 调度张量构建：新增 _build_paged_mqa_schedule_2d_ctx_lens 方法，根据 forward mode 和 SM 版本决定是否输出 [B, next_n] 形状（原生路径）或 [B*next_n, 1] 形状（回退路径）。当满足 target_verify、next_n⩾2 且为 SM100 时输出原生形状；否则沿用原有 per-token 布局。
3. 索引器适配：在 _get_topk_paged 中，通过检查 paged_mqa_ctx_lens_2d 的形状决定是否启用原生路径（use_dg_native）。若启用，则直接以 q_fp8.view(B, next_n, H, D) 调用 deep_gemm.fp8_paged_mqa_logits，并相应调整 block_tables[::next_n] 对齐行主序；否则回退到原有 q_fp8.unsqueeze(1) 展开调用。
4. 导入调整：条件导入 deep_gemm 并新增 is_sm100_supported 工具函数判断。

关键文件：

- python/sglang/srt/layers/attention/dsa_backend.py（模块 DSA 后端；类别 source；类型 dependency-wiring；符号 _build_paged_mqa_schedule_2d_ctx_lens）：新增 _build_paged_mqa_schedule_2d_ctx_lens 方法，扩展元数据结构，串联 schedule 预计算与原生路径选取。
- python/sglang/srt/layers/attention/dsa/dsa_indexer.py（模块 稀疏索引器；类别 source；类型 core-logic）：核心逻辑：根据 ctx_lens_2d 形状选择原生路径或回退路径，优化

MQA logits 计算。

关键符号：_build_paged_mqa_schedule_2d_ctx_lens, init_forward_metadata, _get_topk_paged

关键源码片段

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

新增 _build_paged_mqa_schedule_2d_ctx_lens 方法，扩展元数据结构，串联 schedule 预计算与原生路径选取。

```
# dsa_backend.py 关键变更：新增字段与方法用于预计算二维上下文长度

class DSAMetadata:
    # ... 其他字段
    paged_mqa_schedule_metadata: Optional[torch.Tensor] = None
    # 2D context_lens 用于构建 schedule；索引器可复用，避免每层重复广播
    paged_mqa_ctx_lens_2d: Optional[torch.Tensor] = None

class DSAIndexerMetadata(BaseIndexerMetadata):
    # ...
    paged_mqa_ctx_lens_2d: Optional[torch.Tensor] = None

# 在 DSA 后端中新增的方法
class DeepseekSparseAttnBackend:
    # ...
    def _build_paged_mqa_schedule_2d_ctx_lens(
        self,
        forward_mode: ForwardMode,
        cache_seqlens_int32: torch.Tensor,
        seqlens_expanded: torch.Tensor,
        batch_size: int,
    ) -> torch.Tensor:
        # target_verify 且 next_n >= 2 且为 SM100 时，使用原生 [B, next_n] 形状
        next_n = self.speculative_num_draft_tokens
        if (
            forward_mode.is_target_verify()
            and next_n
            and next_n >= 2
            and is_sm100_supported()
        ):
            return cache_seqlens_int32.view(-1, 1).expand(-1, next_n).contiguous()
        if forward_mode.is_target_verify() or forward_mode.is_draft_extend(
            include_v2=True
        ):
            return _to_2d_context_lens(seqlens_expanded, batch_size)
        return _to_2d_context_lens(cache_seqlens_int32, batch_size)

# 在 init_forward_metadata 中调用新方法构建 ctx_lens_2d
```

```
if is_cuda() and can_build_schedule:
    # 原有 schedule metadata 构建之前
    ctx_lens_2d = self._build_paged_mqa_schedule_2d_ctx_lens(
        forward_batch.forward_mode,
        cache_seqlens_int32,
        seqlens_expanded if hasattr else None,
        batch_size,
    )
    metadata.paged_mqa_ctx_lens_2d = ctx_lens_2d
```

python/sglang/srt/layers/attention/dsa/dsa_indexer.py

核心逻辑：根据 ctx_lens_2d 形状选择原生路径或回退路径，优化 MQA logits 计算。

```
# dsa_indexer.py _get_topk_paged 方法中的关键分支
# 使用预计算的 ctx_2d 形状作为选择依据
ctx_2d = getattr(metadata, "paged_mqa_ctx_lens_2d", None)
use_dg_native = (
    _is_cuda
    and forward_batch.forward_mode.is_target_verify()
    and next_n >= 2
    and ctx_2d is not None
    and ctx_2d.shape == (B, next_n)
)

if use_dg_native:
    # 原生路径：直接使用 [B, next_n, H, D] 形状，block_tables 去重
    logits = deep_gemm.fp8_paged_mqa_logits(
        q_fp8[:q_offset].view(B, next_n, q_fp8.shape[1], q_fp8.shape[2]), # 不 unsqueeze
        kv_cache_fp8,
        weights[:q_offset],
        seqlens_32_2d, # 已经来自 ctx_2d
        block_tables[::next_n], # 对应 B 个 block
        schedule_metadata,
        max_seq_len,
        clean_logits=False,
    )
else:
    # 回退路径：保持原有 unsqueeze(1) 展开
    q_fp8 = q_fp8.unsqueeze(1)
    logits = deep_gemm.fp8_paged_mqa_logits(
        q_fp8[:q_offset],
        kv_cache_fp8,
        weights[:q_offset],
        seqlens_32_2d,
        block_tables,
        schedule_metadata,
        max_seq_len,
        clean_logits=False,
    )
```

评论区精华

主要的讨论是合并后 ch-wan 发现 main 分支上新增的 DSA 测试失败，怀疑该 PR 引入回归。维护者 Fridge003 触发了额外 CI 测试，但尚未确认根因。此担忧成为后续 revert 的直接原因。

- DSA 测试回归怀疑 (correctness): 未解决，最终该 PR 被 revert (PR #27138)。

风险与影响

- 风险:
  1. 正确性风险：原生路径仅通过 ctx_2d.shape 作隐式条件，若 target_verify 前后 batch_size 或 next_n 计算偏差可能导致错误分支。
  2. 兼容性风险：SM90 上 next_n=2 仅添加 TODO，未实际验证，可能触发未定义行为。
  3. 回归风险：索引器中 block_tables[::next_n] 要求 stride 兼容，若 page_table 布局不符可能导致偏移错误。
  4. 测试覆盖缺失：本次变更未新增针对性测试，仅依赖已有 CI（后被发现主分支 DSA 测试失败）。 - 影响：对使用 DeepSeek V3.2 NVFP4 且启用 MTP（speculative EAGLE）的用户，在 SM100 上可获得 ~7% E2E decode TPS 提升；SM90 用户不受影响（回退到原路径）。但可能引入 DSA 测试回归，需要紧急修复或回退。 - 风险标记：SM100 原生路径未在 SM90 验证，缺少针对性测试覆盖，已确认导致 DSA 测试回归，原生路径依赖隐式形状判断

关联脉络

- PR #27138 Revert "Support NextN = 2/4 in DSV32": 直接 revert 本 PR，因合并后导致 DSA 测试失败。
- PR #27004 fix(disagg): correct DSA/SWA state-page transfer mismatch in PD disaggregation: 同为 DSA 相关优化，修复传输不匹配，与本 PR 测试失败可能同源。