PR #25195 完整报告

sgl-project/sglang

[BCG] Support breakable CUDA graph for DeepSeek V4 DP attention

合并时间：2026-06-09 04:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/25195

执行摘要

- 一句话：支持 DSV4 DP Attention 可断点 CUDA 图
- 推荐动作：建议所有 DeepSeek V4 相关开发者精读此 PR，重点关注：
  1. DSV4AttnMetadata.refresh_for_breakable_cuda_graph_replay_ 中的字段分类设计，这是图捕获地址稳定的核心。
  2. BCG 运行器中 DP rank 同步保护的模式，可复用于其他分布式后端。
  3. 注意力后端接口的扩展方式，后续为其他模型添加 BCG 支持时可参考 DeepseekV4AttnBackend 的 opt-in 实现。

对于运维团队，建议先在低风险预发布环境进行 A/B 性能对比，确认工作负载符合预期后再推广。

功能与动机

DeepSeek V4 当前无法在 mixed/extend 批次中使用 breakable CUDA graph，导致 prefill/mixed 路径停留在 *eager* 内核启动，在高并发下产生大量 host-side 空隙。本 PR 通过添加必要的 DSV4 和 DP 注意力管道，使可断点 CUDA graph 捕获 / 重播能够在 DSV4 mixed chunk 工作负载上正常工作。

实现拆解

1. 注意力后端接口扩展：在 base_attn_backend.py 新增 init_forward_metadata_for_breakable_cuda_graph_capture 和 prepare_forward_metadata_for_breakable_cuda_graph_replay 虚方法，并添加类属性 use_captured_forward_metadata_for_breakable_cuda_graph 作为 opt-in 开关。DeepseekV4AttnBackend 设置该属性为 True，实现捕获时创建元数据、重播时原地刷新动态字段的完整流程。
2. DSV4 元数据刷新机制：在 deepseek_v4_backend.py 中新增 DSV4AttnMetadata.refresh_for_breakable_cuda_graph_replay_ 方法，区分“tensor 地址必须保持稳定（仅 copy_ 内容）”和“可以安全替换对象”两类字段。DSV4Metadata 同样添加对应方法，递归刷新子元数据。同时调整 init_forward_metadata_prefill 等签名，传递 extend_start_loc 以支持图捕获。
3. BCG 运行器 DP 集成：在 breakable_cuda_graph_runner.py 中，初始化时读取后端开关，创建元数据缓冲区。新增 _has_inactive_dp_rank 检测稀疏 DP 批次，防止某些 rank 零

token 时集合通信死锁。修改 _warmup、_replay 等流程，通过调用后端的捕获 / 重播接口替换原有的 init_forward_metadata 调用。

4. 模型层中断标记：在 deepseek_v4.py 中新增 deepseek_v4_attention_with_output 函数，注册为@register_custom_op和@register_split_op，使其成为BCG可识别的图中断点。通过 eager_on_graph(True) 装饰器确保该操作在图外以 eager 方式执行。同时调整 forward 方法，在 BCG 模式下替换注意力调用路径。
5. 索引器与 TBO 适配：dsv4/indexer.py 增加 match_num_queries 工具函数，对齐不同 tensor 的第一维大小，处理 BCG 捕获时 query 数不一致的问题。two_batch_overlap.py 添加 TBO 禁用时的提前返回分支，避免无谓的 compute_split_seq_index 调用。
6. 配套测试：新增 TestDSV4BreakableCudaGraphMetadataContract（CPU-only 元数据契约测试）和 TestDSV4FlashFP4BreakableCudaGraphB200（端到端图执行测试），覆盖元数据刷新、tensor 地址稳定性、后端 opt-in 行为等。

关键文件：

- python/sglang/srt/layers/attention/deepseek_v4_backend.py（模块 注意力后端；类别 source；类型 core-logic；符号 refresh_for_breakable_cuda_graph_replay_, create, _build_forward_metadata, init_forward_metadata_for_breakable_cuda_graph_capture）：核心变更：添加图中断元数据刷新机制，实现捕获 / 重播流程，opt-in 属性打开 BCG 支持。
- python/sglang/srt/model_executor/breakable_cuda_graph_runner.py（模块 图执行器；类别 source；类型 data-contract；符号 _has_inactive_dp_rank, _init_forward_metadata_for_capture, _prepare_forward_metadata_for_replay）：BCG 运行器集成 DP 感知的元数据捕获与重播，新增 DP 等级同步保护逻辑。
- python/sglang/srt/models/deepseek_v4.py（模块 模型定义；类别 source；类型 data-contract；符号 deepseek_v4_attention_with_output）：模型层新增可中断注意力操作，注册为 BCG 识别的 split op，修改 forward 路径以支持图执行。
- python/sglang/srt/layers/attention/base_attn_backend.py（模块 基础注意力；类别 source；类型 core-logic；符号 init_forward_metadata_for_breakable_cuda_graph_capture, prepare_forward_metadata_for_breakable_cuda_graph_replay）：巴克斯注意力后端接口扩展，定义捕获 / 重播虚方法和 opt-in 属性，为所有后端提供 BCG 支持框架。
- test/registered/attention/unittests/dsv4/test_deepseek_v4.py（模块 单元测试；类别 test；类型 test-coverage；符号 TestDSV4BreakableCudaGraphMetadataContract, _make_core_metadata, test_bcg_is_explicit_and_dsv4_backend_opt_in_only, test_refresh_replay_metadata_preserves_captured_tensor_storage）：新增 CPU 端元数据契约测试，验证 BCG 重播时 tensor 地址稳定性和 backend opt-in 行为。

关键符号：refresh_for_breakable_cuda_graph_replay_, init_forward_metadata_for_breakable_cuda_graph_capture, prepare_forward_metadata_for_breakable_cuda_graph_replay, _init_forward_metadata_for_capture, _prepare_forward_metadata_for_replay, _has_inactive_dp_rank, deepseek_v4_attention_with_output, match_num_queries, _expand_prefill_casually_vectorized

关键源码片段

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

核心变更：添加图中断元数据刷新机制，实现捕获 / 重播流程，opt-in 属性打开 BCG 支持。

```
class DSV4AttnMetadata:
    def refresh_for_breakable_cuda_graph_replay_(self, other: DSV4AttnMetadata) -> None:
        # 验证静态配置一致（这些字段在捕获和重播间必须相同）
        assert self.c4_sparse_topk == other.c4_sparse_topk
        assert self.page_size == other.page_size
        assert self.cuda_int32_kwargs == other.cuda_int32_kwargs

        # 这些字段的 tensor 地址已被图捕获，只能复制内容，不能替换对象
        tensor_copy_fields = [
            "raw_out_loc", "seq_lens_casual", "positions_casual",
            "c4_out_loc", "c128_out_loc", "c4_topk_lengths_raw",
            "c4_topk_lengths_clamp1", "c4_sparse_topk_lengths",
        ]
        # 这些字段在图中断内使用，可以安全地替换为新对象
        reference_assign_fields = [
            "page_table", "swa_page_indices", "swa_topk_lengths",
            "c128_page_indices", "c128_topk_lengths_clamp1",
            "c1_flashmla_metadata", "c4_flashmla_metadata", "c128_flashmla_metadata",
        ]

        for field_name in tensor_copy_fields:
            src_val = getattr(other, field_name)
            dst_val = getattr(self, field_name)
            if src_val is None and dst_val is None:
                continue
            assert dst_val is not None, f"{field_name=} {src_val=} {dst_val=}"
            dst_val.copy_(src_val)

        for field_name in reference_assign_fields:
            setattr(self, field_name, getattr(other, field_name))


class DSV4Metadata:
    def refresh_for_breakable_cuda_graph_replay_(self, static_metadata: DSV4Metadata):
        # 刷新核心元数据
        self.core_attn_metadata.refresh_for_breakable_cuda_graph_replay_(
            static_metadata.core_attn_metadata
        )
        # 索引器和压缩元数据使用 inplace 复制以保持地址稳定
        maybe_copy_inplace(self.indexer_metadata, src=static_metadata.indexer_metadata)
        maybe_copy_inplace(self.c4_compress_metadata, src=static_metadata.c4_compress_
        metadata)
        # 在线压缩模式下，c128 元数据可能包含 Python-side 规划器状态，需要整体替换
        if envs.SGLANG_OPT_USE_ONLINE_COMPRESS.get():
```

```
            self.c128_compress_metadata = static_metadata.c128_compress_metadata
        else:
            maybe_copy_inplace(self.c128_compress_metadata, src=static_metadata.c128_
            compress_metadata)
```

该片段展示了 BCG 重播时如何细粒度地保持捕获的 tensor 地址稳定（通过 copy_）并安全替换其他引用字段。

python/sglang/srt/model_executor/breakable_cuda_graph_runner.py

BCG 运行器集成 DP 感知的元数据捕获与重播，新增 DP 等级同步保护逻辑。

```
class BreakableCudaGraphRunner:
    def __init__(self, model_runner):
        # ... 其他初始化 ...
        # 读取后端是否使用捕获的元数据（BCG 地址稳定需求）
        self.use_captured_attn_metadata = (
            model_runner.attn_backend.use_captured_forward_metadata_for_breakable_cuda_
            graph
        )
        self.attn_metadata_buffers = {} if self.use_captured_attn_metadata else None

    def _has_inactive_dp_rank(self, forward_batch: "ForwardBatch") -> bool:
        """
        检查是否有 DP 等级包含零个 token，以避免在 sparse DP 批次中
        某些 rank 不进入 BCG replay 路径导致 DeepEP 集合通信死锁。
        """
        global_num_tokens = forward_batch.global_num_tokens_cpu
        if global_num_tokens is None:
            return False
        # 当有多个 DP rank 且至少一个 rank 的 token 数为 0 时返回 True
        return len(global_num_tokens) > 1 and any(
            int(num_tokens) == 0 for num_tokens in global_num_tokens
        )

    def _init_forward_metadata_for_capture(self, forward_batch, num_tokens):
        """在暖身阶段捕获注意力元数据，保存到缓冲区。"""
        attn_backend = self.model_runner.attn_backend
        if not self.use_captured_attn_metadata:
            attn_backend.init_forward_metadata(forward_batch)
            return
        metadata = attn_backend.init_forward_metadata_for_breakable_cuda_graph_capture(
            forward_batch
        )
        assert self.attn_metadata_buffers is not None
        self.attn_metadata_buffers[num_tokens] = metadata

    def _prepare_forward_metadata_for_replay(
        self, forward_batch, static_forward_batch, num_tokens
    ):
```

```
        """回放前准备：从缓冲区获取捕获的元数据，刷新动态内容。"""
        if not self.use_captured_attn_metadata:
            self.model_runner.attn_backend.init_forward_metadata(forward_batch)
            return
        capture_metadata = self.attn_metadata_buffers.get(num_tokens)
        if capture_metadata is None:
            logger.warning(
                "[BCG] No captured metadata for %s tokens; falling back to eager init",
                num_tokens,
            )
            self.model_runner.attn_backend.init_forward_metadata(forward_batch)
            return
        self.model_runner.attn_backend.prepare_forward_metadata_for_breakable_cuda_graph_
        replay(
            capture_metadata, forward_batch, static_forward_batch=static_forward_batch
        )
```

该片段展示了 BCG 运行器如何通过后端 opt-in 开关和元数据缓冲区，在捕获 / 重播阶段分别初始化和刷新注意力元数据，并通过 _has_inactive_dp_rank 避免 DP 同步死锁。

评论区精华

> Oasis-Git 在 deepseek_v4.py 评论："I think by now we do not need pcg support? From my perspective I think it may not work well with torch compile based pcg"。讨论结论：当前 PR 仅聚焦 BCG，PCG 支持暂不纳入。
>
> Oasis-Git 在 scheduler_dp_attn_mixin.py（实际为 dp_attn.py）建议："maybe for these names we can directly use can_run_breakable_cuda_graph instead of piecewise cuda graph"。该命名建议未在最终代码中体现，变量保留为 can_piecewise_cuda_graph，但通过 @property 形式提供了别名兼容。

- PCG 支持必要性讨论 (design)：当前 PR 仅聚焦 BCG，PCG 支持暂不纳入，且从路线图看 PCG 与 BCG 是互斥方案。
- DP 同步变量命名建议 (style)：最终代码保留了原始命名，但通过 @property 提供了向后兼容的别名（未在提交中体现，但 PR 合并后未强行修改）。

风险与影响

- 风险：
  1. CUDA 图元数据地址稳定性：refresh_for_breakable_cuda_graph_replay_ 通过区分 tensor_copy_fields 和 reference_assign_fields 来保证捕获的 tensor 地址不被替换，但若新增字段未正确分类，可能导致图回放时读入无效数据，触发 silent 错误。
1. DP 同步死锁风险：_has_inactive_dp_rank 检测稀疏批次并回退 eager，但依赖 global_num_tokens_cpu 的准确性。若该信息出错或广播延迟，可能导致部分 DP rank 进入 replay 而另一部分留在 eager，造成集合通信死锁。
2. 混合批次退出路径：当 BCG 无法处理当前批次（如 _has_inactive_dp_rank 为真）时，需要干净地回退到原始 eager 路径。当前回退逻辑覆盖了主分支，但可能遗漏某些边缘 case

（如 CaptureHiddenMode 与 logits 处理的交互）。

3. 性能收益波动：性能提升依赖于具体工作负载（并发度、DP 维度、模型配置）。在低并发场景下 BCG 收益可能不明显，甚至因元数据管理额外开销而轻微倒退。
4. 测试覆盖不足：单元测试仅覆盖 CPU 端元数据契约和 B200 单配置端到端测试，缺少多 GPU 环境下的 DP 同步集成测试。 - 影响：用户影响：启用 BCG 后（通过 --enable-breakable-cuda-graph），使用 DeepSeek V4 且开启 DP attention 的用户可在 mixed chunk 场景下获得 6-12% 的吞吐提升和 3-13% 的 TPOT 改善。用户需要确保 CUDA 版本和 GPU 架构支持，当前已验证 B300 和 CoreWeave 集群。

系统影响：修改了注意力气垫层的抽象接口（base_attn_backend.py），所有后端类需适配新的虚方法，但默认 opt-out 保证兼容。BCG 运行器新增 DP 感知逻辑，可能与其他分布式策略（如 Tensor Parallel、Sequence Parallel）交互需验证。

团队影响：新增的元数据捕获 / 重播模式成为后续注意力后端实现图中断的规范接口，团队需要维护这一双重路径。测试代码的 TestDSV4BreakableCudaGraphMetadataContract 可作为其他后端实现 BCG 时的参考契约。

- 风险标记：核心路径变更，DP 同步死锁风险，BCG 元数据地址稳定性，测试覆盖不足，性能收益波动

关联脉络

- PR #27289 [AMD] dsv4: remove the redundant fp8 scale transpose-copy on decode: 同样修改了 deepseek_v4.py 和相关注意力后端，属于同一模型系列的持续性能优化，与本 PR 的 BCG 支撑形成互补。