PR #26383 完整报告

sgl-project/sglang

[AMD][DSV4] DSV4 MTP graph + sparse triton attn optimizations

合并时间：2026-05-28 06:23

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26383

执行摘要

- 一句话：修复 DSV4 MTP 在 ROCm 上的 CUDA Graph 捕获并优化注意力与融合 kernel
- 推荐动作：建议精读。该 PR 展示了如何在不破坏 CUDA 路径的前提下为 ROCm 修复关键错误并注入性能优化，其设计权衡（始终 eager 构造、fused kernel 阈值选择、三级 fallback 模式）值得学习。尤其推荐关注 deepseek_v4_fused_mhc.py 中的缓冲池与运行时禁用机制。

功能与动机

引用 PR body：'The long-standing draft #25552 collected 85 commits of AMD / DSV4 work over several rebases. Most of the optimization PRs ... have already been merged into main through smaller follow-up PRs. What remains is a small but meaningful set of changes needed to:

1. Make DSV4 + EAGLE/MTP speculative decoding work correctly under CUDA-graph capture on ROCm. 2. Land the last round of ROCm performance wins ... 3. Provide a clean, single-commit, conflict-free landing.'

实现拆解

1. HIP 后端分发与 MTP 图捕获修复 在 draft_utils.py 中对 _create_dsv4_decode_backend 和 _create_dsv4_prefill_backend 添加 is_hip() 判断，将 ROCm 平台分派到 deepseek_v4_backend_hip_radix 的 DeepseekV4MultiStepBackend / DeepseekV4HipRadixBackend。 在 deepseek_v4_backend_hip_radix.py 中重写 init_forward_metadata_target_verify，使其始终走 eager metadata 构造路径（旧版 init_forward_metadata_target_verify_old），避免在 SGLANG_PREP_IN_CUDA_GRAPH 开启时因 lazy-upgrade 导致规划器不变量被破坏。同时修正了多步 draft 的 out_cache_loc 切片：在 init_forward_metadata 解码态下通过 per_step_draft_out_cache_loc 取当前 step 的缓存位置。
2. Fused Triton mHC post+pre 算子 新增 python/sglang/srt/models/deepseek_common/amd/deepseek_v4_fused_mhc.py，实现 try_fused_hc_post_pre 函数。当满足条件（token数≤64、启用环境变量SGLANG_OPT_USE_TRITON_FUSED_MHC、GFX95支持）时，尝试调用 aiter 的 mhc_post_pre 内核，将 hc_post、hc_pre 和后续的 layernorm 合并为单个 Triton 启动。该函数返回 (residual, hidden_states, post, comb, norm_fused) 五元组；若内核不可用则返回 None，调用方回退到原有的分开执行路径。

3. 优化 Triton 稀疏注意力解码内核 在 triton_mla_kernels_decode_optimized.py 中新增 _should_use_fused_nosplitk 决策函数，用于大 batch（≥256 token）时选择 fused no-splitk 双域注意力内核，实测比分离 gather+attention 路径快 10-14%。_should_use_fused_dual_scope 的阈值也随之调整，以匹配 MI355X 实测数据。
4. 启用 w_qkv 融合与 aiter 采样 在 deepseek_v4.py 中移除 fuse_wqa_wkv 的 not _is_hip 限制，使 ROCm 也能使用 w_qkv 融合压缩；添加 _freqs_cis_to_cos_sin 缓存以避免各层重复转换。在 sampler.py 中，当 SGLANG_USE_AITER 为 true 且为 HIP 时，将 all-greedy 采样替换为 aiter.greedy_sample，写入预分配的 int32 缓冲。
5. 测试与配置配套 新增 test/registered/ops/test_aiter_greedy_sample_amd.py，覆盖 kernel 级与 torch.argmax 的等价性验证、各种 batch 与 vocab 尺寸、tied values、负数 logits 等；注册到 AMD CI stage-b 测试套件。同时在 environ.py 中新增 SGLANG_OPT_USE_TRITON_FUSED_MHC 开关。

关键文件：

- python/sglang/srt/models/deepseek_common/amd/deepseek_v4_fused_mhc.py（模块 融合算子；类别 source；类型 core-logic；符号 _get_triton_mhc_post_pre_ops, _get_fused_hc_post_pre_buffers, try_fused_hc_post_pre）：新增的 AMD 专用融合 mHC 算子模块，封装了从导入尝试到缓冲区分配再到运行的核心逻辑，是整个 PR 性能优化的关键。
- test/registered/ops/test_aiter_greedy_sample_amd.py（模块 AMD 测试；类别 test；类型 test-coverage；符号 _mock_global_server_args, _DummyTPGroup, _make_sampling_info, TestAiterGreedySample）：新增的 aiter.greedy_sample 单元测试，覆盖 kernel 正确性与 Sampler 集成，注册到 AMD CI。
- python/sglang/srt/layers/attention/nsa/triton_decode/triton_mla_kernels_decode_optimized.py（模块 注意力内核；类别 source；类型 core-logic；符号 _should_use_fused_nosplitk）：添加 fused no-splitk 路径选择函数，大幅优化大 batch 解码性能，同时调整双域阈值以匹配 MI355X 实测。
- python/sglang/srt/models/deepseek_v4.py（模块 模型适配；类别 source；类型 data-contract；符号 _freqs_cis_to_cos_sin）：集成 fused mHC、启用 w_qkv 融合（移除 HIP 限制）、添加 freqs_cis 缓存，并修正 ROCm 多流条件。
- python/sglang/srt/layers/attention/deepseek_v4_backend_hip_radix.py（模块 HIP 后端；类别 source；类型 dependency-wiring）：修复 HIP 后端 MTP 图捕获的核心位置：修改 init_forward_metadata_target_verify 始终走 eager 构造。

关键符号：try_fused_hc_post_pre, _get_triton_mhc_post_pre_ops, _get_fused_hc_post_pre_buffers, _should_use_fused_nosplitk, _freqs_cis_to_cos_sin

关键源码片段

python/sglang/srt/models/deepseek_common/amd/deepseek_v4_fused_mhc.py

新增的 AMD 专用融合 mHC 算子模块，封装了从导入尝试到缓冲区分配再到运行的核心逻辑，是整个 PR 性能优化的关键。

```
import logging
from typing import Optional, Tuple

import torch
import triton

from sglang.srt.environ import envs

logger = logging.getLogger(__name__)

# 模块级常量与全局状态
_FUSED_HC_POST_PRE_M_THRESHOLD = 64 # batch token 阈值
_FUSED_HC_POST_PRE_CACHE: dict[tuple, dict[str, torch.Tensor]] = {} # 缓冲池缓存
_TRITON_MHC_POST_PRE_OPS = None # 惰性导入的结果
_TRITON_MHC_POST_PRE_RUNTIME_DISABLED = False # 运行时禁用标记


def _get_triton_mhc_post_pre_ops():
    """惰性导入 aiter 的 mhc_post_pre 及其配置工具，结果全局缓存。"""
    global _TRITON_MHC_POST_PRE_OPS
    if _TRITON_MHC_POST_PRE_OPS is not None:
        return _TRITON_MHC_POST_PRE_OPS
    try:
        from aiter.ops.triton.fusions.mhc import mhc_post_pre
        from aiter.ops.triton.utils.mhc_config_utils import get_mhc_config
    except Exception as err:
        # 导入失败时返回 None，调用方自行降级
        logger.warning("Triton fused mHC (mhc_post_pre) is unavailable, falling back: %s", err)
        return None
    _TRITON_MHC_POST_PRE_OPS = (mhc_post_pre, get_mhc_config)
    return _TRITON_MHC_POST_PRE_OPS


def _get_fused_hc_post_pre_buffers(
    num_tokens: int, hidden_size: int, hc_mult: int,
    dtype: torch.dtype, device: torch.device,
) -> Optional[dict[str, torch.Tensor]]:
    """根据 shape 与配置获取预分配的中间缓冲（来自全局缓存或新创建）。"""
    ops = _get_triton_mhc_post_pre_ops()
    if ops is None:
        return None
    _, get_mhc_config = ops
    key = (num_tokens, hidden_size, hc_mult, dtype, device.type, device.index)
    bufs = _FUSED_HC_POST_PRE_CACHE.get(key)
    if bufs is not None:
        return bufs
    # 通过 get_mhc_config 获取 kernel 配置，动态计算拆分组数
    try:
        cfg, _ = get_mhc_config("MHC_FUSED", num_tokens, hidden_size, mode="sinkhorn")
```

```
    except Exception as err:
        logger.warning("Failed to initialize fused mHC config, falling back: %s", err)
        return None
    n_total = 2 * hc_mult + hc_mult * hc_mult
    k_dim = hc_mult * hidden_size
    block_k = cfg.get("BLOCK_K", min(512, triton.next_power_of_2(k_dim)))
    block_k = min(block_k, triton.next_power_of_2(k_dim))
    block_c_split = max(block_k // hc_mult, 1)
    num_ksplit = triton.cdiv(hidden_size, block_c_split)
    # 分配全部中间缓冲
    bufs = {
        "residual_out": torch.empty(num_tokens, hc_mult, hidden_size, dtype=dtype, device=device)
        ,
        "layer_input_out": torch.empty(num_tokens, hidden_size, dtype=dtype, device=device),
        "h_post": torch.empty(num_tokens, hc_mult, dtype=torch.float32, device=device),
        "h_res": torch.empty(num_tokens, hc_mult, hc_mult, dtype=torch.float32, device=device),
        "acc_partial": torch.empty(num_ksplit, num_tokens, n_total, dtype=torch.float32, device=
        device),
        "acc_sq_partial": torch.empty(num_ksplit, num_tokens, dtype=torch.float32, device=device)
        ,
    }
    _FUSED_HC_POST_PRE_CACHE[key] = bufs
    return bufs
```

test/registered/ops/test_aiter_greedy_sample_amd.py

新增的 aiter.greedy_sample 单元测试，覆盖 kernel 正确性与 Sampler 集成，注册到 AMD CI。

```
import unittest
from unittest import mock

import torch

from sglang.srt.utils.common import is_hip
from sglang.test.ci.ci_register import register_amd_ci

# 注册到 AMD CI stage-b 套件，预估 60 秒
register_amd_ci(est_time=60, suite="stage-b-test-1-gpu-small-amd")


def _mock_global_server_args(backend="pytorch"):
    """替换 sampler 模块中的全局配置为伪值，避免真正初始化 ServerArgs。"""
    from sglang.srt.layers import sampler as sampler_mod
    from sglang.srt.server_args import ServerArgs
    sampler_mod.get_global_server_args = lambda: ServerArgs(
        model_path="dummy", sampling_backend=backend,
    )
    class _DummyTPGroup:
        device_group = None
```

```
    sampler_mod.get_tp_group = lambda: _DummyTPGroup()
    sampler_mod.is_dp_attention_enabled = lambda: False


def _make_sampling_info(batch_size, vocab_size, device="cuda"):
    """构造一个全 greedy 的 SamplingBatchInfo 用于测试。"""
    from sglang.srt.sampling.sampling_batch_info import SamplingBatchInfo
    return SamplingBatchInfo(
        temperatures=torch.ones(batch_size, 1, device=device, dtype=torch.float),
        top_ps=torch.ones(batch_size, device=device),
        top_ks=torch.zeros(batch_size, device=device, dtype=torch.int32),
        min_ps=torch.zeros(batch_size, device=device),
        is_all_greedy=True,
        need_top_p_sampling=False,
        need_top_k_sampling=False,
        need_min_p_sampling=False,
        vocab_size=vocab_size,
        device=device,
    )


@unittest.skipUnless(is_hip(), "aiter greedy_sample requires ROCm")
class TestAiterGreedySample(unittest.TestCase):
    """Kernel 级正确性：aiter.greedy_sample 与 torch.argmax 等价。"""

    @classmethod
    def setUpClass(cls):
        try:
            from aiter import greedy_sample
            cls.greedy_sample = staticmethod(greedy_sample)
        except ImportError:
            raise unittest.SkipTest("aiter not installed")
        cls.device = "cuda"

    def setUp(self):
        torch.manual_seed(42)
        torch.cuda.manual_seed_all(42)

    def _run_and_compare(self, batch_size, vocab_size):
        """核心比较：随机 bf16 logits -> aiter.greedy_sample vs argmax。"""
        logits = torch.randn(batch_size, vocab_size, device=self.device, dtype=torch.bfloat16)
        expected = torch.argmax(logits, dim=-1)
        actual = torch.empty(logits.shape[0], device=logits.device, dtype=torch.int32)
        self.greedy_sample(actual, logits)
        self.assertTrue(
            torch.equal(actual.to(expected.dtype), expected),
            f"Mismatch for shape ({batch_size}, {vocab_size})",
        )
```

```
    def test_single_request(self):
        self._run_and_compare(1, 32000)

    def test_small_batch(self):
        self._run_and_compare(4, 32000)

    # 更多测试（medium/large batch，真实词汇量，tied values 等）在完整文件中
```

评论区精华

Review 中 gemini-code-assist[bot] 提出了 4 条中等偏高的修改建议：

- （高）_get_triton_mhc_post_pre_ops 导入失败后未缓存：每次 forward 都会重试导入并打 warning 日志，建议用布尔标记缓存失败状态。
- （中）_freqs_cis_to_cos_sin 使用 id(freqs_cis) 作缓存键：若原 tensor 被回收存在 id 重用风险，建议在缓存值中保留引用。
- （中）sampler.py 中直接调用 get_bool_env_var('SGLANG_USE_AITER')：应使用集中的 envs 配置系统。
- （中）init_forward_metadata_target_verify 新增 extend_seq_lens 参数但未使用：函数体内完全忽略，也未传给 init_forward_metadata_target_verify_old。 四条评论均未得到维护者回复或修改，PR 已合并，这些建议处于未解决状态。此外，yctseng0211 在 issue 评论中指出 AMD CI 的已知失败与本 PR 无关。
- 导入失败未缓存导致重复日志 (performance): 未采纳 / 未回复
- id(freqs_cis) 缓存键存在回收风险 (correctness): 未采纳 / 未回复
- SGLANG_USE_AITER 应使用集中 envs 机制 (design): 未采纳 / 未回复
- 新增 extend_seq_lens 参数未被使用 (correctness): 未采纳 / 未回复

风险与影响

- 风险：
  1. id 缓存回收风险：_freqs_cis_to_cos_sin 使用 id(freqs_cis) 作为 key，若原 tensor 被垃圾回收，地址重用可能导致返回错误的 cos/sin 表。实践中 freqs_cis 常驻，但仍是契约风险。
  2. extend_seq_lens 被忽略：init_forward_metadata_target_verify 新增的 extend_seq_lens 参数未传入下层，可能影响特定场景下 target-verify 的元数据构造，导致偶发错误。
  3. 依赖回退路径测试不足：aiter.greedy_sample 与 mhc_post_pre 依赖外部 aiter 包，缺失或版本不兼容时会静默回退到 PyTorch 原语，但回退路径的精度与性能未在 CI 中充分验证。
  4. HIP 后端专用代码的维护成本：部分修改通过 is_hip() 分支隔离，未来若 CUDA 侧逻辑变化可能导致 HIP 分支滞后。 - 影响：用户影响：AMD ROCm 用户运行 DeepSeek V4 模型时，MTP 推测解码不再因 CUDA Graph 捕获失败而中断；解码延迟降低（fused kernel 加速）。CUDA 用户不受影响（行为不变的 HIP 分支）。系统影响：新增环境变

量 SGLANG_OPT_USE_TRITON_FUSED_MHC 控制融合开关；新测试仅运行在 AMDCI。团队影响：AMD 团队维护约 159 行新模块 deepseek_v4_fused_mhc.py 和 289行测试，后续可通过移除旧路径逐步简化。

- 风险标记：id 回收风险，extend_seq_lens 忽略，依赖条件回退，核心路径变更，review 建议未修复

关联脉络

- PR #25552 Draft PR: AMD/DSV4 work（未在历史记录中列出）：本 PR 是 #25552 的最后一批未合并变更的 squash 版本；大部分优化已通过子 PR 在主分支合并。