PR #47896 完整报告

vllm-project/vllm

[Kernel][ROCm][Perf] FlyDSL decode-attention kernel for 4-bit TurboQuant KV cache

合并时间：2026-08-11 23:45

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47896

执行摘要

- 一句话：ROCm gfx950 新增 FlyDSL 4-bit TurboQuant 解码内核
- 推荐动作：值得精读。对 ROCm/AMD 内核开发者，这是 FlyDSL DSL 内核嵌入 vLLM attention 后端的完整样例；对平台维护者，_SegmBufPool + _max_capture_batch_size 解决多尺寸 CUDA-graph 捕获显存与地址稳定性的思路，以及 rocm.py backend 选择放宽的边界条件都值得借鉴。CUDA 用户无需深入内核细节，但可以关注后续 backend 抽象与配置字段的演进。

功能与动机

PR 动机来自 AMD 团队在 agentic serving 场景对 4-bit TurboQuant KV-cache 解码性能的诉求：默认 Triton 解码在 gfx950 上未能充分利用 CDNA4 MFMA 指令与硬件转置，FlyDSL 提供的低层 DSL 可以写出单 wave CTA、LDS 常驻质心 LUT 等优化。PR body 同步附上 TurboQuant 算法背景博客（https://rocm.blogs.amd.com/artificial-intelligence/turboquant-vllm-agentic/README.html），说明其在 agentic vLLM 服务中的角色；设计上作为自动选路的替代方案（早期为 env 开关，最终改为 gfx950 + FlyDSL 可导入时的能力门控），确保默认路径保持不变。

实现拆解

1. 新增 SoA 流水线基础：在 vllm/v1/attention/ops/turboquant_soa/ 下新增 triton_turboquant_store.py、triton_turboquant_decode.py、triton_turboquant_decode_v2.py 与 triton_turboquant_unified_attention.py。它们按 SoA 布局读写缓存——每个 block 的数据区与元数据区（k_norm/v_scale/v_zero）分离，目标是与 FlyDSL 内核共享同一套字节偏移约定；store 内核写入量化后的 KV，decode 内核在 tile 循环内直接反量化。
2. 加入 FlyDSL 内核：新增 flydsl_kernels/tq_decode.py（GQA-8/16，Qwen 类）与 tq_decode_gqa6.py（GQA-6，MiniMax-M2.5），使用 FlyDSL DSL 生成 CDNA4 MFMA 指令；新增 flydsl_turboquant_decode.py 作为启动器，按形状分派、按 (Hk, num_partitions, max_blocks_per_seq, scale) 缓存编译产物，并通过 _SegmBufPool 用单一 max_B 桶复用中间输出，避免多尺寸 CUDA-graph 捕获带来的显存膨胀和每步 cudaMalloc。
3. 接入 TurboQuantAttentionImpl：turboquant_attn.py 增加 _soa_imports 惰性导入、_use_flydsl/_soa_store 开关、_dispatch_decode_soa 分派，以及在 _ensure_on_device

中针对 FlyDSL 的 CUDA-graph 安全预热（预分配 arange/_cu_2、扩满 WorkspaceManager）；_store_kv 与 continuation/prefill 路径在 SoA 开启时同步切换到 SoA Triton 实现，保证缓存布局与解码 / 续算路径一致。

4. 调整 attention-backend 选择：rocm.py 的 get_attn_backend_cls 在 kv_cache_dtype 以 turboquant 开头时，允许显式选择的 backend 与该层不匹配并回退到自动选路，以支持边界层 AITER + turboquant 层 TURBOQUANT 的混合布局；其他 dtype 仍保持显式选择无效即报错。
5. 测试与验证配套：新增 tests/kernels/turboquant/test_flydsl_turboquant_decode.py，用已知质心构造 SoA 4-bit KV cache，以纯 PyTorch fp32 attention 为 oracle，参数化覆盖 batch×seq_len×GQA×block_size，并且在非 gfx950 或 FlyDSL 不可导入时 module-level skip；PR body 附 GSM8K 精度与端到端性能数据。

关键文件：

- vllm/v1/attention/backends/turboquant_attn.py（模块 注意力后端；类别 source；类型 dependency-wiring；符号 _soa_imports, _max_capture_batch_size, _dispatch_decode_soa）：TurboQuant 后端主集成点：引入 SoA 惰性导入、FlyDSL 能力探测、store/continuation/decode 的 SoA 分支，以及 CUDA-graph 捕获前的设备分配预热，是 FlyDSL 路径与既有缓存布局、graph 捕获体系正确对齐的关键。
- vllm/v1/attention/ops/flydsl_turboquant_decode.py（模块 解码启动器；类别 infra；类型 infrastructure；符号 is_flydsl_available, is_flydsl_gqa6_available, _detect_max_capture_B, _SegmBufPool）：FlyDSL 启动器与运行时适配：负责 gfx950 能力探测、GQA-6 sibling 的 best-effort 导入、内核模块缓存，以及单桶 segm pool 对 CUDA-graph 多尺寸捕获的支持，是 FlyDSL 内核与 vLLM 解码循环之间的桥梁。
- tests/kernels/turboquant/test_flydsl_turboquant_decode.py（模块 单元测试；类别 test；类型 test-coverage；符号 _build_cache, _reference_attention, test_flydsl_matches_reference）：FlyDSL 解码内核的唯一正确性测试：构造 SoA 4-bit KV cache，并用纯 PyTorch fp32 attention oracle 校验输出，覆盖 batch×seq_len×GQA×block_size 组合，是非 gfx950 平台之外最重要的回归防线。
- vllm/v1/attention/ops/flydsl_kernels/tq_decode.py（模块 解码内核；类别 infra；类型 infrastructure；符号 build_tq_decode_module, tq_decode_kernel, _vsplat_mul）：FlyDSL 规范 GQA-8/GQA-16 解码内核（Qwen 类），单 wave CTA + FA-2 online softmax + CDNA4 MFMA 是性能提升的主体之一。
- vllm/v1/attention/ops/flydsl_kernels/tq_decode_gqa6.py（模块 解码内核；类别 infra；类型 infrastructure；符号 build_tq_decode_gqa6_module, tq_decode_gqa6_kernel, _vsplat_mul）：GQA-6 兄弟内核，MiniMax-M2.5 类模型专用；与规范内核共享结构但 QG=6，MFMA 16-row 容量仅使用 37.5%，通过 mfma_row < QG 谓词屏蔽垃圾 lane。
- vllm/v1/attention/ops/turboquant_soa/triton_turboquant_unified_attention.py（模块 Triton 内核；类别 infra；类型 infrastructure；符号 _tq_fuse_q_rotation, _tq_load_k_tile, _tq_load_v_tile, kernel_tq_unified_attention_2d）：新增 SoA 布局下的统一（prefill+decode）Triton 注意力内核，K/V 在 tile 内反量化，是回退路径的完整 SoA 解码能力来源。

- vllm/v1/attention/ops/turboquant_soa/triton_turboquant_decode.py（模块 Triton 解码；类别 infra；类型 infrastructure；符号 _tq_decode_stage1, _tq_full_dequant_kv, _use_fp8_e4b15, triton_turboquant_decode_attention）：SoA Triton decode 单头 kernel，作为 FlyDSL 不可用或形状不符时的回退实现。
- vllm/v1/attention/ops/turboquant_soa/triton_turboquant_decode_v2.py（模块 Triton 解码；类别 infra；类型 infrastructure；符号 build_pair_lut, _tq_decode_stage1_v2, triton_turboquant_decode_attention_v2）：FLUTE paper 优化的 grouped-Q + pair LUT 版本，为回退路径提供更好的 tl.dot 利用率与 exp2 替代 exp。
- vllm/v1/attention/ops/turboquant_soa/triton_turboquant_store.py（模块 KV 存储；类别 infra；类型 infrastructure；符号 _store_quantized_value, _tq_fused_store_mse, _tq_fused_store_fp8, triton_turboquant_store）：SoA 布局的 fused Triton 存储内核，决定 FlyDSL 解码所依赖的缓存写入格式（数据区 + 元数据区分离）。
- vllm/platforms/rocm.py（模块 平台层；类别 source；类型 core-logic）：attention-backend 选择逻辑变更：仅 turboquant* KV dtype 下允许显式 backend 与层不匹配并回退自动选路，其他 dtype 保持 fail loud，支撑混合后端部署。
- vllm/v1/attention/ops/turboquant_soa/__init__.py（模块 包初始化；类别 infra；类型 infrastructure）：新包初始化，保证 SoA Triton 内核模块可被 _soa_imports 引入。
- vllm/v1/attention/ops/flydsl_kernels/__init__.py（模块 包初始化；类别 infra；类型 infrastructure）：新包初始化，暴露 tq_decode/tq_decode_gqa6 模块给启动器。

关键符号：flydsl_turboquant_decode_attention, is_flydsl_available, is_flydsl_gqa6_available, build_tq_decode_module, build_tq_decode_gqa6_module, triton_turboquant_store, triton_turboquant_decode_attention_soa, build_pair_lut, _tq_load_k_tile, _tq_fuse_q_rotation, _soa_imports, _dispatch_decode_soa, _max_capture_batch_size, _SegmBufPool.get, _reference_attention, test_flydsl_matches_reference

关键源码片段

vllm/v1/attention/ops/flydsl_turboquant_decode.py

FlyDSL 启动器与运行时适配：负责 gfx950 能力探测、GQA-6 sibling 的 best-effort 导入、内核模块缓存，以及单桶 segm pool 对 CUDA-graph 多尺寸捕获的支持，是 FlyDSL 内核与 vLLM 解码循环之间的桥梁。

```
# vllm/v1/attention/ops/flydsl_turboquant_decode.py（新增）
# FlyDSL 能力探测：仅在 gfx950 且 FlyDSL 可导入时启用，否则回退 SoA Triton。

_FLYDSL_AVAILABLE: bool | None = None

def is_flydsl_available() -> bool:
    """返回当前环境是否可用 FlyDSL TQ 解码（gfx950 + 可导入）。

    GQA-6 sibling（tq_decode_gqa6）按 best-effort 导入：缺失时不影响
    Qwen 系 GQA-{8,16} 路径，只有 GQA-6 模型会回退到 SoA Triton。
    """
```

```
    if _FLYDSL_AVAILABLE is not None:
        return _FLYDSL_AVAILABLE
    try:
        from vllm.platforms.rocm import on_gfx950
        if not on_gfx950():
            _FLYDSL_AVAILABLE = False
            return False
        import flydsl.compiler as flyc
        import flydsl.expr as fx
        from flydsl._mlir import ir
        from flydsl.compiler.kernel_function import CompilationContext
        from flydsl.expr.typing import T
        from vllm.v1.attention.ops.flydsl_kernels import tq_decode as tq_mod
        _TQ_MOD = tq_mod
        _FLYDSL_AVAILABLE = True
        logger.info_once("FlyDSL TQ decode launcher: available")
    except Exception as ex: # noqa: BLE001
        _FLYDSL_AVAILABLE = False
        logger.warning_once(
            "FlyDSL TQ decode launcher: unavailable (%s). "
            "Falling back to SoA Triton decode.",
            ex,
        )
        return _FLYDSL_AVAILABLE
    # GQA-6 sibling 的导入独立于主路径，失败不影响 Qwen 支持。
    try:
        from vllm.v1.attention.ops.flydsl_kernels import tq_decode_gqa6 as tq_mod_gqa6
        _TQ_MOD_GQA6 = tq_mod_gqa6
        logger.info_once("FlyDSL TQ decode GQA-6 sibling: available (MiniMax-class)")
    except Exception as ex: # noqa: BLE001
        _TQ_MOD_GQA6 = None
        logger.info_once(
            "FlyDSL TQ decode GQA-6 sibling: not available (%s); "
            "GQA-6 models will fall back to SoA Triton decode.",
            ex,
        )
    return _FLYDSL_AVAILABLE
```

tests/kernels/turboquant/test_flydsl_turboquant_decode.py

FlyDSL 解码内核的唯一正确性测试：构造 SoA 4-bit KV cache，并用纯 PyTorch fp32 attention oracle 校验输出，覆盖 batch×seq_len×GQA×block_size 组合，是非 gfx950 平台之外最重要的回归防线。

```
# tests/kernels/turboquant/test_flydsl_turboquant_decode.py（新增）
# 用 fp32 纯 PyTorch attention 作为 oracle，校验 FlyDSL 解码结果的正确性。

def _reference_attention(q_bf16, k_ref, v_ref, seq_lens, scale):
    """对去量化后的 K/V 做 fp32 softmax attention （ground truth）。"""
    num_seqs, hq, d = q_bf16.shape
```

```
    hk = k_ref.shape[1]
    qg = hq // hk
    q = q_bf16.float().reshape(num_seqs, hk, qg, d)
    out = torch.zeros(num_seqs, hq, d, dtype=torch.float32)
    for s in range(num_seqs):
        for h in range(hk):
            ql = int(seq_lens[s].item())
            k = k_ref[s, h, :ql]
            v = v_ref[s, h, :ql]
            scores = (q[s, h] @ k.T) * scale
            m = scores.max(dim=-1, keepdim=True).values
            e = torch.exp(scores - m)
            p = e / e.sum(dim=-1, keepdim=True)
            out[s, h * qg : (h + 1) * qg] = p @ v
    return out

@pytest.mark.parametrize("num_seqs", [1, 4, 16])
@pytest.mark.parametrize("seq_len", [256, 1024, 4096])
@pytest.mark.parametrize(
    "num_kv_heads,qg",
    [
        (8, 8), # Qwen2.5-72B class
        (8, 16), # Qwen3-32B class
        pytest.param(
            4, 6, # MiniMax-M2.5 class  (GQA-6 sibling kernel)
            marks=pytest.mark.skipif(
                not is_flydsl_gqa6_available(),
                reason="FlyDSL GQA-6 sibling kernel not available",
            ),
        ),
    ],
)
@pytest.mark.parametrize("kv_block_size", [16, 32])
def test_flydsl_matches_reference(num_seqs, seq_len, num_kv_heads, qg, kv_block_size):
    """FlyDSL decode 必须与 fp32 attention （去量化 KV）一致。"""
    centroids, q_bf16, kv_cache, block_table, seq_lens, k_ref, v_ref = _build_cache(
        num_seqs, num_kv_heads, seq_len, qg, kv_block_size
    )
    scale = 1.0 / (HEAD_SIZE**0.5)
    identity = torch.eye(HEAD_SIZE, dtype=torch.float32, device="cuda")
    out = flydsl_turboquant_decode_attention(
        query=q_bf16,
        kv_cache=kv_cache,
        block_table=block_table,
        seq_lens=seq_lens,
        Pi=identity,
        centroids=centroids,
        scale=scale,
        mse_bits=4,
```

```
        key_packed_size=KEY_DATA_BYTES + 2,
        value_quant_bits=4,
        value_packed_size=KEY_DATA_BYTES + 4,
        key_fp8=False,
        norm_correction=False,
        PiT=identity.T.contiguous(),
        max_seq_len=seq_len,
        max_num_kv_splits=32,
        sinks=None,
    )
    ref = _reference_attention(q_bf16.cpu(), k_ref, v_ref, seq_lens.cpu(), scale)
    torch.testing.assert_close(out.cpu().float(), ref, atol=ATOL, rtol=0.0)
```

评论区精华

评审主要围绕四个问题展开：一是 fxmarty-amd 质疑 rocm.py 中放宽 selected backend 校验会静默忽略无效的 --attention-backend（如 FLASHINFER_MLA_SPARSE_SM120），aditi-amd 将其收敛为仅 turboquant* KV dtype 下回退，其他 dtype 继续 fail loud；二是 fxmarty-amd 与 tjtanaa 都反对新增 VLLM_ROCM_TQ_FLYDSL_DECODE 等环境变量，tjtanaa 明确要求走 python 参数系统，最终提交删除了 env 开关改为 gfx950 能力自动选路，但 --attention-backend turboquant_flydsl 或配置字段留待后续；三是 fxmarty-amd 询问 FlyDSL 内核应托管在 vLLM 还是 AMD 专属仓库，aditi-amd 引用 PR #44400 作为先例，评审最终认可保留在 vLLM；四是 fxmarty-amd 建议以子类化替代 if/else 分支，aditi-amd 同意作为后续 RFC/PR。此外 tjtanaa 提出了测试平台导入保护与 AITER 版本策略问题，均已修复或回退。

- attention-backend 选择语义与 turboquant 混合后端 (design): aditi-amd 将回退范围收敛到仅 turboquant* KV dtype，其他 dtype 下显式选择无效时仍 fail loud；混合后端是边界层跳过量化（--kv-cache-dtype-skip-layers）的既有设计，--attention-backend 目前是全局单值选择器。fxmarty-amd 认为该改动与 FlyDSL 集成关系不大，建议单独 PR 并评估跨平台一致性，此点未在本 PR 完全闭环。
- 环境变量 vs 配置项 / 自动选路 (design): 最终提交删除了环境变量，改为 gfx950 + FlyDSL 可导入时的能力自动选路（on_gfx950() gating）；但 --attention-backend turboquant_flydsl 或配置字段的落地方案留待后续 PR/RFC。
- FlyDSL 内核托管位置 (design): aditi-amd 以 PR #44400 为先例将内核随 vLLM 维护；fxmarty-amd 转述 'keeping AMD-only kernels in vllm-project/vllm is fine'，话题关闭。
- 子类化改造 if/else 分支 (design): 本 PR 仍以内联分支形式合入，子类化重构作为后续项挂起。
- 测试的非 ROCm 导入保护 (testing): 测试改为 module-level skip，先判断 current_platform.is_rocm() 和 on_gfx950()，再在条件内导入 ROCm/FlyDSL 依赖，CI 非 ROCm 平台可干净跳过。
- AITER 版本兼容与 lazy import 回退 (other): aditi-amd 接受并全部 revert，使用上游 Docker 镜像自带 AITER。

- v4 命名与 Triton 回退语义 (style): v4 字样全部移除；回退命名语义未在本 PR 修改，作者表示可后续探讨。

风险与影响

- 风险：
  1. 平台强绑定：FlyDSL 内核仅面向 gfx950/CDNA4，其他 ROCm 卡依赖 Triton SoA 回退；若回退路径未覆盖的边界形状（如 key_fp8=1、非 4-bit、HEAD_SIZE≠128）被错误路由，性能收益消失但功能不受影响。
  2. 布局混用风险：同一 kv_cache 张量只能有一种字节约定，turboquant_attn.py 中 SoA 开启时 _store_kv 与 continuation 路径必须同步切换，否则 AoS 解码读 SoA 缓存会拿到错误的 k_norm/v_scale/v_zero 偏移，导致多轮或前缀缓存请求精度崩溃。
  3. attention-backend 选择行为变化：rocm.py 对 turboquant* dtype 放宽校验，虽限定前缀且其余 dtype 仍报错，但跨平台一致性与用户对显式 --attention-backend 的预期仍需澄清。
  4. CUDA-graph 捕获路径：新增的预热、segm pool 与 arange cache 依赖 max_model_len 与 capture 配置；超出最大捕获 batch 时回退 eager，已通过 _max_capture_batch_size 取 max 缓解。
  5. 测试覆盖有限：核心正确性测试只在 ROCm/gfx950 且 FlyDSL 可用时运行，CUDA 与多数 ROCm CI 均为 skip，回归面有限。 - 影响：影响范围集中：仅当使用 4-bit TurboQuant KV（如 --kv-cache-dtype turboquant_4bit_nc）且处于 ROCm gfx950 + FlyDSL 可导入时触发，默认路径完全不变。对 MI355X 用户，agentic 或长上下文 decode 可获得约 4.1× 相对默认 Triton 解码的吞吐提升，精度整体中性（GSM8K 差距 <0.4pp）；对其他硬件用户无行为变化。系统层面，attention-backend 选择逻辑新增 turboquant 前缀分支，混合后端（边界层 AITER + 主体 TURBOQUANT）成为受支持组合。团队层面，AMD 侧需要持续维护 FlyDSL DSL 内核与 vLLM 版本的兼容性，并跟进 --attention-backend turboquant_flydsl、TurboQuantAttentionImpl 子类化等后续抽象演进。 - 风险标记：gfx950-only 内核，attention-backend 选择行为变更，CUDA graph 捕获路径改动，测试覆盖仅限 gfx950，SoA/AoS 布局混用风险，新增外部 DSL 依赖

关联脉络

- PR #44400 FlyDSL MoE integration（review 引用的托管先例）：review 讨论中作为 AMD FlyDSL 内核托管在 vLLM 主仓的先例被引用。
- PR #51756 [Bugfix] Take the sliding window from the layer, not the KV cache group: 同属 v1 注意力后端正确性修复，本次 FlyDSL 解码资格判断也排除了 sliding_window。
- PR #51749 [Bugfix] Generalize KV block zeroing to AttentionSpec: 同属 v1 注意力 /KV cache 管理链路的近期修复，涉及 turboquant_attn 同路径的单类型 KV cache manager。