PR #50230 完整报告

vllm-project/vllm

[Perf][CUDA] Programmatic dependent launch for the DSA decode kernels

合并时间：2026-08-06 15:53

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50230

执行摘要

- 一句话：DSA decode 内核接入 PDL，重叠启动延迟提升吞吐
- 推荐动作：值得精读。这是一个把 CUDA 新特性 PDL 同时落到 Triton 与原生 CUDA 两种内核体系的最小但完整范例：Triton 侧展示了 constexpr 门控 + gdc_wait() / gdc_launch_dependents() + launch_pdl 的用法，CUDA 侧展示了 cudaLaunchKernelEx + cudaLaunchAttributeProgrammaticStreamSerialization 的标准写法。最有价值的设计决策是门控只依赖 is_arch_support_pdl() 而非叠加 sm100 判断（review 推动），以及与 #49792 冲突时选择保留 CuTeDSL 主路径、PDL 落到 Triton fallback 的分层策略。若不在 Blackwell 硬件上开发，可快速浏览即可。

功能与动机

本 PR 是 #48597（GLM-5.2 Blackwell decode 优化跟踪页）重新拆分后的聚焦项之一。PR body 说明：decode 路径存在大量连续的小内核（norm/rope、fused-q、NVFP4 quant），在低 batch 下内核启动延迟占主导，PDL 让存在依赖关系的连续内核无需 CPU 侧逐次下发、直接在 GPU 侧串行化启动，从而重叠 launch latency。基准显示单独本 PR 将 output tok/s 从 446.7 提升到 459.2、median TPOT 从 1.94 ms 降到 1.88 ms，full series 达 542.5 output tok/s / 1.56 ms；gsm8k 5-shot 0.950、无无效输出，正确性无回退。

实现拆解

实现按 4 步拆解：

1. Triton 内核接入 PDL（vllm/models/deepseek_v32/common/kernels.py，+14 行）：_fused_norm_rope_kernel 与 _fused_q_kernel 签名新增 USE_PDL: tl.constexpr，内核入口处条件插入 tl.extra.cuda.gdc_wait() 与 tl.extra.cuda.gdc_launch_dependents()；Python 侧 fused_norm_rope / fused_q 在启动前计算 use_pdl = current_platform.is_arch_support_pdl()，同时传入 USE_PDL=use_pdl（编译期裁剪同步原语）与 launch_pdl=use_pdl（Triton 启动开关）。这样不支持的平台不编入任何 PDL 指令。
2. NVFP4 量化内核改造（csrc/libtorch_stable/quantization/fp4/nvfp4_quant_kernels.cu，+40/-10 行）：cvt_fp16_to_fp4 与 cvt_fp16_to_fp4_sf_major 内核体开头插入 cudaGridDependencySynchronize() / cudaTriggerProgrammaticLaunchCompletion()，以 #if (defined(__CUDA_ARCH__) && (__CUDA_ARCH__>= 900)) 保护，老架构编译期即空操作；启动方式从 <<<grid, block, 0, stream>>> 改为 cudaLaunchKernelEx 并配置

cudaLaunchAttributeProgrammaticStreamSerialization，因为旧语法无法携带 PDL 需要的启动属性。

3. 运行时门控与简化：两条路径统一依赖 current_platform.is_arch_support_pdl()，非 CUDA 平台返回 False、SM90 以下同样返回 False，整体退化为原实现；review 后（提交 dc9aa110）删除了包装函数里多余的 has_device_capability(100) 条件与 @torch.compiler.assume_constant_result，改为与仓库其他 PDL 调用点（fused_qk_rmsnorm、fused_inv_rope_fp8_quant、kimi_k3 ops）一致的内联写法。
4. 测试与验证配套：未新增测试文件，复用 tests/kernels/test_bf16_skinny_gemm.py 与 tests/kernels/test_fused_deepseek_v32_norm_rope.py（10passed，重跑4passed）；在 8×B300 TP8 上跑 GLM-5.2-NVFP4 + MTP=5 吞吐基准与 gsm8k 5-shot 评估。PR body 注明合成低熵数据集使 MTP 接受率接近上限，绝对吞吐偏乐观，但行间对比在同一数据集上仍公平。

关键文件：

- vllm/models/deepseek_v32/common/kernels.py（模块 模型内核；类别 source；类型 data-contract；符号 _fused_norm_rope_kernel, _fused_q_kernel, fused_norm_rope, fused_q）：DSA decode 路径的核心 kernel 启动文件，两个 Triton 内核（_fused_norm_rope_kernel、_fused_q_kernel）新增 USE_PDL constexpr 与 gdc_wait/gdc_launch_dependents，启动处按 is_arch_support_pdl() 传入 USE_PDL 与 launch_pdl，是本次 PDL 改造的主入口。
- csrc/libtorch_stable/quantization/fp4/nvfp4_quant_kernels.cu（模块 量化内核；类别 source；类型 core-logic；符号 scaled_fp4_quant_sm1xxa, cvt_fp16_to_fp4, cvt_fp16_to_fp4_sf_major）：NVFP4 量化内核的 C++ 实现，两个 CUDA kernel 插入 grid dependency 同步原语，启动方式从 <<<>>> 改为 cudaLaunchKernelEx + programmatic stream serialization，是 CUDA 侧 PDL 改造的核心文件。

关键符号：_fused_norm_rope_kernel, _fused_q_kernel, fused_norm_rope, fused_q, scaled_fp4_quant_sm1xxa, cvt_fp16_to_fp4, cvt_fp16_to_fp4_sf_major

关键源码片段

vllm/models/deepseek_v32/common/kernels.py

DSA decode 路径的核心 kernel 启动文件，两个 Triton 内核（_fused_norm_rope_kernel、_fused_q_kernel）新增 USE_PDL constexpr 与 gdc_wait/gdc_launch_dependents，启动处按 is_arch_support_pdl() 传入 USE_PDL 与 launch_pdl，是本次 PDL 改造的主入口。

```
# vllm/models/deepseek_v32/common/kernels.py
# PDL 门控：is_arch_support_pdl() 在非 CUDA 平台返回 False，且要求 SM90+ 架构，
# 因此这一处判断即可覆盖全部平台，无需再叠加 has_device_capability(100)。

@triton.jit
def _fused_norm_rope_kernel(
    # 前置参数：Q/KV RMS norm、RoPE cache、MLA cache、topk buffer 等
    HAS_INDEXER: tl.constexpr,
    INDEX_ROPE_INTERLEAVE: tl.constexpr,
```

```
    USE_PDL: tl.constexpr, # 编译期常量：True 时才把 PDL 同步原语编入内核
):
    pid = tl.program_id(0)
    tok_idx = tl.program_id(1)
    if USE_PDL:
        # 先等待上一枚依赖内核触发，再在本内核结束时立即触发下一枚，
        # 让背靠背的小内核启动延迟在 GPU 侧互相重叠
        tl.extra.cuda.gdc_wait()
        tl.extra.cuda.gdc_launch_dependents()
    # 后续 norm / rope / cache 写入逻辑不变，PDL 只影响调度时机


def fused_norm_rope(...):
    # ...
    use_pdl = current_platform.is_arch_support_pdl()
    _fused_norm_rope_kernel[(4, num_tokens)](
        # ... 其余参数 ...
        USE_PDL=use_pdl, # 不支持的平台不编入同步原语
        launch_pdl=use_pdl, # 指示 Triton 按 PDL 序列启动
    )
    return q_c_out
```

csrc/libtorch_stable/quantization/fp4/nvfp4_quant_kernels.cu

NVFP4 量化内核的 C++ 实现，两个 CUDA kernel 插入 grid dependency 同步原语，启动方式从 <<<>>> 改为 cudaLaunchKernelEx + programmatic stream serialization，是 CUDA 侧 PDL 改造的核心文件。

```
// csrc/libtorch_stable/quantization/fp4/nvfp4_quant_kernels.cu
// 内核内部：仅在 SM90+ 编译期插入 grid dependency 同步原语，
// 老架构下整段为空操作，行为与原先完全一致
#if (defined(__CUDA_ARCH__) && (__CUDA_ARCH__ >= 900))
    cudaGridDependencySynchronize();
    cudaTriggerProgrammaticLaunchCompletion();
#endif

// 启动端：<<<>>> 语法无法携带 PDL 属性，改用 cudaLaunchKernelEx
// 声明 programmatic stream serialization，使相邻依赖内核可重叠启动
cudaLaunchConfig_t config = {};
config.gridDim = grid;
config.blockDim = block;
config.dynamicSmemBytes = 0;
config.stream = stream;
cudaLaunchAttribute attrs[1];
attrs[0].id = cudaLaunchAttributeProgrammaticStreamSerialization;
attrs[0].val.programmaticStreamSerializationAllowed = 1;
config.numAttrs = 1;
config.attrs = attrs;
cudaLaunchKernelEx(&config, vllm::cvt_fp16_to_fp4<cuda_type, false>,
                   m, n, output_n, num_padded_cols, input_ptr,
```

```
    input_sf_ptr,
    reinterpret_cast<uint32_t*>(output_ptr),
    reinterpret_cast<uint32_t*>(sf_out));
```

评论区精华

核心交锋来自 gau-nernst 的一条 review 评论：'current_platform.is_arch_support_pdl() is enough right? We don't need to gate behind sm100 only'，作者回复 'right, let me fix.' 后提交 dc9aa110 删除了包装函数里的 has_device_capability(100) 条件与 assume_constant_result 装饰器，理由是 is_arch_support_pdl() 已涵盖非 CUDA 返回 False 与 SM90+ 要求，额外条件反而让 H100/H200 错失 PDL 快路径。另一个要点是 PR body 对与 #49792 同区域冲突的 rebase 细节提醒（装饰器行位于冲突标记上方，naive 合并会丢掉它），最终 merge commit bbd59b25 采用保留 CuTeDSL 路径、PDL 应用于 Triton fallback 的解法；WoosukKwon 触发 /ci run 后 Buildkite CI #82618 通过。

- PDL 门控是否只需 is_arch_support_pdl() (design): zhou9402 回复 'right, let me fix.'，提交 dc9aa110 删除包装函数：is_arch_support_pdl() 在非 CUDA 平台已返回 False 且要求 SM90+，has_device_capability(100) 反而让 H100/H200 错失 PDL 快路径；同时移除无收益的 assume_constant_result，改为与仓库其他 PDL 调用点一致的内联写法。
- 与 #49792 的 kernels.py 同区域冲突处理 (other): 已通过 merge 解决，head 中 fused_q 先走 cutedsl_kernel 分支，PDL 仅作用于 Triton 启动路径。

风险与影响

- 风险：
  1. PDL 语义风险：gdc_wait() 依赖前置内核确实以 programmatic serialization 方式启动，若未来有分支混跑（如 fused_q 的 cutedsl_kernel 分支不经过 PDL 启动）可能出现等待永不触发的问题；当前同步原语只在同一函数内成对被插入，两侧门控一致，风险可控。
  2. 平台回归风险：非 SM90+ 平台编译期裁剪、行为不变，理论上无回归；但 Triton 内核新增 constexpr 会使编译面扩大（每个 USE_PDL 取值各编译一份），需留意 kernel cache 体积。
  3. 测试覆盖缺口：本 PR 没有新增测试文件，PDL 路径（gdc_*、cudaLaunchKernelEx）在 CI 的覆盖有限，正确性依赖现有内核测试与人工基准。
  4. 合并顺序依赖：与 #49792 同文件冲突已解决，但系列内 #49790 / #49791 / #49793 的合并顺序仍影响“full series”基准口径（PR body 也说明 main 行与 full series 行是两次独立会话，非同一轮 back-to-back 对比）。 - 影响：影响面集中在 vllm/models/deepseek_v32 模型族（GLM-5.2、DeepSeek-V3.2 等）在 SM90+ CUDA 平台上的 decode 小内核链；对 CPU、AMD、Intel GPU 等无 PDL 平台为 no-op，不影响行为。性能上 8×B300 TP8 下单独约 2.8% output tok/s 提升（TPOT 1.94 → 1.88 ms），作为系列一环参与 full-series 约 21% 的提升（542.5 vs 446.7）。团队层面，本 PR 确立的 is_arch_support_pdl() 内联门控模式正在成为仓库统一做法（fused_qk_rmsnorm、kimi_k3 ops 等均同款），后续同类优化可低成本复用。 - 风险标记：核心路径变更，缺少测试覆盖，平台条件编译，合并顺序依赖

关联脉络

- PR #48597 [Perf][GLM-5.2] Blackwell decode optimizations: 本 PR 所属的重新拆分跟踪页，合并顺序、基准与验证政策均定义于此
- PR #49790 SM100 sparse-model integration and routing: 按跟踪页约定必须先于本 PR 合入，使 vllm/models/deepseek_v32 包在 main 上可达
- PR #49792 CuTeDSL fused-query kernel: 与 #50230 修改同一文件 vllm/models/deepseek_v32/common/kernels.py，存在合并冲突；最终保留 CuTeDSL 主路径、PDL 作用于 Triton fallback
- PR #49791 Small-batch decode GEMM optimizations: 同系列 small-batch decode 优化，与 PDL 目标一致，均针对低并行度下的小内核开销
- PR #49793 MTP/speculative-decoding optimizations: 同系列 MTP 优化，与 #50230 共同构成 full-series 基准结果（542.5 vs 446.7 output tok/s）