PR #23965 完整报告

sgl-project/sglang

Enable PDL for various kernels in DSV32/GLM5

合并时间：2026-05-09 18:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23965

执行摘要

- 一句话：多 kernel 启用 PDL 提升 DSV32/GLM5 低延迟性能
- 推荐动作：值得精读。该 PR 展示了在 Triton 和 CUDA kernel 中正确启用 Programmatic Dependent Launch 的方法，修复了 unsafe asm 的问题，可作为 sgl-kernel 中类似 kernel 的参考模板。建议关注 is_arch_support_pdl 的检测逻辑和 cudaLaunchKernelEx 的使用方式。

功能与动机

PR 目的为改进 GLM-5 和 DSV3.2 的低延迟性能，通过启用 PDL 减少 kernel launch 依赖开销。同时移除潜在不安全的 asm volatile("griddepcontrol.launch_dependents;") 调用（参见 flashinfer issue #2558），替换为带 memory clobber 的 CUDA API 调用，消除 undefined behavior。

实现拆解

1. Triton kernel 添加 GDC 支持：在 python/sglang/srt/mem_cache/utils.py 中，为 set_mla_kv_buffer_kernel 和 set_mla_kv_buffer_fp8_quant_kernel 添加 USE_GDC 编译常量，在 kernel 开闭处调用 tl.extra.cuda.gdc_wait() 和 gdc_launch_dependents()。Python 包装函数通过 is_arch_support_pdl() 动态传入 USE_GDC=True 和 launch_pdl=True，从而启用 Triton 的 PDL 支持。
2. FlashInfer FP8 量化入口传递 PDL 标志：在 python/sglang/srt/layers/attention/utils.py 中，导入 is_arch_support_pdl，并在 mla_rope_quantize_fp8 调用时添加 enable_pdl=is_arch_support_pdl()，使 FlashInfer 内核能够利用 PDL。
3. sgl-kernel CUDA kernel 替换不安全 asm：在 dsv3_router_gemm_bf16_out.cu、dsv3_router_gemm_float_out.cu、dsv3_fused_a_gemm.cu 和 pos_enc.cuh 中，将 asm volatile("griddepcontrol.wait;") 替换为 cudaGridDependencySynchronize()，将 asm volatile("griddepcontrol.launch_dependents;") 替换为 cudaTriggerProgrammaticLaunchCompletion()，这些调用自动包含内存顺序保证。
4. per_token_quant_8bit_v2 kernel 使用现代 launch API：在 sgl-kernel/csrc/gemm/per_token_group_quant_8bit_v2.cu 中，将传统的 <<<grid, block, 0, stream>>> launch 改为使用 cudaLaunchKernelEx 和 cudaLaunchAttributeProgrammaticStreamSerialization 属性的方式，允许 kernel 内部的显式依赖同步。同时在 kernel 首尾添加 cudaGridDependencySynchronize() 和

cudaTriggerProgrammaticLaunchCompletion()。

5. NSA Indexer 提取编译优化方法：在 python/sglang/srt/layers/attention/nsa/nsa_indexer.py 中，将原来分散在 decode 路径中的 weights.unsqueeze(-1) * q_scale * self.softmax_scale 内联计算提取为独立的 @torch.compile(dynamic=True) 方法 _apply_q_scale_and_softmax_scale，便于编译器融合并简化代码。

关键文件：

- python/sglang/srt/mem_cache/utils.py（模块 缓存层；类别 source；类型 dependency-wiring；符号 set_mla_kv_buffer_kernel, set_mla_kv_buffer_fp8_quant_kernel, set_mla_kv_buffer_triton, set_mla_kv_buffer_triton_fp8_quant）：核心依赖文件，为 MLA KV buffer 的 Triton kernel 添加 USE_GDC 支持，并动态启用 PDL。
- python/sglang/srt/layers/attention/nsa/nsa_indexer.py（模块 注意力；类别 source；类型 core-logic；符号 _apply_q_scale_and_softmax_scale）：提取编译优化方法 _apply_q_scale_and_softmax_scale，简化 decode 路径中的缩放计算。
- python/sglang/srt/layers/attention/utils.py（模块 注意力；类别 source；类型 dependency-wiring；符号 mla_quantize_and_rope_for_fp8）：导入 is_arch_support_pdl 并传递给 FlashInfer 的 mla_rope_quantize_fp8 以启用 PDL。
- sgl-kernel/csrc/gemm/per_token_group_quant_8bit_v2.cu（模块 量化 GEMM；类别 other；类型 core-logic；符号 per_token_group_quant_8bit_kernel, sgl_per_token_group_quant_8bit_v2）：使用 cudaLaunchKernelEx 配置 PDL 属性，替换传统 launch 方式，支持程序化流序列化。
- sgl-kernel/csrc/gemm/dsv3_router_gemm_bf16_out.cu（模块 路由器 GEMM；类别 other；类型 entrypoint；符号 router_gemm_kernel_bf16_output）：替换不安全的 asm volatile 为 CUDA API，消除 UB。代表 router GEMM 的改动。

关键符号：set_mla_kv_buffer_kernel, set_mla_kv_buffer_fp8_quant_kernel, set_mla_kv_buffer_triton, set_mla_kv_buffer_triton_fp8_quant, _apply_q_scale_and_softmax_scale, mla_quantize_and_rope_for_fp8, per_token_group_quant_8bit_kernel, router_gemm_kernel_bf16_output, router_gemm_kernel_float_output

关键源码片段

python/sglang/srt/mem_cache/utils.py

核心依赖文件，为 MLA KV buffer 的 Triton kernel 添加 USE_GDC 支持，并动态启用 PDL。

```
@triton.jit
def set_mla_kv_buffer_kernel(
    kv_buffer_ptr,
    cache_k_nope_ptr,
    cache_k_rope_ptr,
    loc_ptr,
    buffer_stride: tl.constexpr,
    nope_stride: tl.constexpr,
```

```
    rope_stride: tl.constexpr,
    nope_dim: tl.constexpr,
    rope_dim: tl.constexpr,
    BLOCK: tl.constexpr,
    USE_GDC: tl.constexpr = False, # 新增参数，控制是否使用 GDC (Grid Depend Control)
):
    pid_loc = tl.program_id(0)
    pid_blk = tl.program_id(1)
    base = pid_blk * BLOCK
    offs = base + tl.arange(0, BLOCK)
    total_dim = nope_dim + rope_dim
    mask = offs < total_dim

    if USE_GDC:
        tl.extra.cuda.gdc_wait() # 等待前序 kernel 的依赖完成

    loc = tl.load(loc_ptr + pid_loc).to(tl.int64)
    dst_ptr = kv_buffer_ptr + loc * buffer_stride + offs

    # Three-way branch to handle boundary correctly while preserving fast path
    if base + BLOCK <= nope_dim:
        src = tl.load(cache_k_nope_ptr + pid_loc * nope_stride + offs, mask=mask)
    elif base >= nope_dim:
        offs_rope = offs - nope_dim
        src = tl.load(cache_k_rope_ptr + pid_loc * rope_stride + offs_rope, mask=mask)
    else:
        is_nope = offs < nope_dim
        is_rope = (offs >= nope_dim) & (offs < (nope_dim + rope_dim))
        src_nope = tl.load(cache_k_nope_ptr + pid_loc * nope_stride + offs, mask=mask & is_nope,
        other=0)
        src_rope = tl.load(cache_k_rope_ptr + pid_loc * rope_stride + (offs - nope_dim), mask=
        mask & is_rope, other=0)
        src = tl.where(is_nope, src_nope, src_rope)

    tl.store(dst_ptr, src, mask=mask)

    if USE_GDC:
        tl.extra.cuda.gdc_launch_dependents() # 标记当前 kernel 完成，后序 kernel 可依赖


def set_mla_kv_buffer_triton(kv_buffer, loc, cache_k_nope, cache_k_rope):
    nope_dim = cache_k_nope.shape[-1]
    rope_dim = cache_k_rope.shape[-1]
    total_dim = nope_dim + rope_dim
    BLOCK = 128
    n_loc = loc.numel()
    grid = (n_loc, triton.cdiv(total_dim, BLOCK))

    # 根据架构支持情况决定是否启用 PDL (Programmatic Dependent Launch)
```

```
    pdl_kwargs = {"USE_GDC": True, "launch_pdl": True} if is_arch_support_pdl() else {}

    set_mla_kv_buffer_kernel[grid](
        kv_buffer, cache_k_nope, cache_k_rope, loc,
        kv_buffer.stride(0), cache_k_nope.stride(0), cache_k_rope.stride(0),
        nope_dim, rope_dim, BLOCK=BLOCK,
        **pdl_kwargs, # 动态传递 USE_GDC 和 launch_pdl 标志
    )
```

sgl-kernel/csrc/gemm/per_token_group_quant_8bit_v2.cu

使用 cudaLaunchKernelEx 配置 PDL 属性，替换传统 launch 方式，支持程序化流序列化。

```
// LAUNCH_KERNEL 宏的片段，展示如何使用 cudaLaunchKernelEx 配置 PDL
#define LAUNCH_KERNEL(GROUP_SIZE, T, DST_DTYPE) \
  do { \
    SCHEDULER::compute_exec_config( \
        THREADS_PER_SUBWARP, num_local_experts, hidden_dim_num_groups, \
        num_groups, subwarps_per_block, grid, block); \
                                                      \
    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 = getEnvEnablePDL(); \
    config.numAttrs = 1; \
    config.attrs = attrs; \
    cudaLaunchKernelEx( \
        &config, \
        per_token_group_quant_8bit_kernel<SCHEDULER, GROUP_SIZE, \
                                          THREADS_PER_SUBWARP, T, \
                                          DST_DTYPE, __VA_ARGS__>, \
        static_cast<T*>(input.data_ptr()), \
        static_cast<DST_DTYPE*>(output_q.data_ptr()), \
        static_cast<output_s_dtype*>(output_s.data_ptr()), \
        static_cast<int32_t*>(masked_m.has_value() ? masked_m->data_ptr() : 0), \
        subwarps_per_block, \
        hidden_dim_num_groups, \
        scale_expert_stride, \
        scale_hidden_stride, \
        num_tokens_per_expert); \
  } while (0)

// 内核内部首尾添加同步点
__global__ void per_token_group_quant_8bit_kernel(...) {
#if defined(__CUDA_ARCH__) && __CUDA_ARCH__ >= 900
    cudaGridDependencySynchronize(); // 等待前序 kernel 完成
```

```
#endif
    // ... 量化逻辑 ...
#if defined(__CUDA_ARCH__) && __CUDA_ARCH__ >= 900
    cudaTriggerProgrammaticLaunchCompletion(); // 标记本 kernel 完成，允许后序启动
#endif
}
```

评论区精华

PR 获得 nvpohanh 和 Fridge003 的 approval。nvpohanh 指出该修改应自动适用于 PR#23351。mattteochen 评论 LGTM。无进一步的 review 争议。

- griddepcontrol asm lacks memory clobber (security): PR 将 asm 调用替换为 cudaGridDependencySynchronize 和 cudaTriggerProgrammaticLaunchCompletion，修复了 UB。

风险与影响

- 风险：
  1. PDL 依赖于 SM90+ 架构（CUDA arch >= 900），is_arch_support_pdl() 的实现需正确检测，否则在旧架构上启用会导致运行时错误。
  2. 替换 asm volatile 为 CUDA API 需要确保所有相关 kernel 都覆盖，且新 API 的行为与旧 asm 一致（虽然更安全）。
  3. 新导入的 is_arch_support_pdl 函数可能在某些 Python 环境中缺失，需确认其在 sglang.jit_kernel.utils 中的可用性。
  4. NSA Indexer 中 _apply_q_scale_and_softmax_scale 的 @torch.compile 可能带来首次编译开销，但被 dynamic=True 缓解。 - 影响：用户影响：在 SM90+ GPU 上运行 DSV32 或 GLM5 模型时，低延迟 TPS 提升约 3.4%，且消除潜在的性能不稳定因素。系统影响：修改集中在内核 launch 方式，不影响模型输出精度（GPQA 测试通过）。兼容性：对其他模型和 GPU 架构无影响，因为 PDL 启用由 is_arch_support_pdl() 条件控制。 - 风险标记：SM90+ 依赖，asm 替换兼容性，is_arch_support_pdl 可用性，torch.compile 首次开销

关联脉络

- PR #23351 (unknown, externally referenced): PR body 指出此修改应自动适用于该 PR，表明同一代码共享相同的 asm 模式。
- PR #24562 Fix performance regression on Deepseek V3 on moe-runner-backend=triton on SM90: 同为 deepseek 模型性能修复，共享 MoE 和 kernel 优化上下文。