PR #37317 完整报告

sgl-project/sglang

[Kernel] Raise shape limits in shared FLA and MoE kernels (ported from #36507)

合并时间：2026-09-01 09:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37317

执行摘要

- 一句话：放宽 FLA/MoE 共享内核 shape 上限，grid 拆分避免超限
- 推荐动作：值得精读，尤其适合 kernel 开发者与多模态 / 新模型 day0 支持方向的工程师。三个值得借鉴的设计决策：一是把「易超限的折叠轴拆到独立 grid 轴」作为通用模式，可平移到其他 Triton kernel；二是从 stride(0) 而不是 shape[1] 推导 per-request 缓存步数，以适配 adaptive speculative decoding 下运行时形状变化，这是一个容易被忽略的正确性细节；三是多 block-per-row 拆分隐藏维度的做法，解除了 out_dim <= 1024 * kVecSize 的隐性限制。合入前建议确认 CI 失败原因，并在 main 分支补齐新模型大 shape 场景的显式验证。

功能与动机

PR body 仅注明 Ported from #36507，未展开动机。结合代码变更与 Fridge003 的批准评论可以还原意图：原实现把 batch 与 head 折叠进同一 CUDA grid 轴（N * HV 或 B * HV），大 batch 下乘积超过 grid.y/z 的 65535 上限导致 kernel 无法启动；silu_mul_clamp 的单 block-per-row 限制也让大 hidden dim（fp8 下超过约 16K）无法运行。审查者以 day0 分支 kimi-k3 通过作为合并依据，说明这些限制正是新模型 day0 支持路上的障碍。

实现拆解

变更入口：本 PR 是 #36507 的移植，5 个文件、+59/-32，核心集中在 3 个 Triton kernel 文件、1 个 CUDA 头文件与 1 个模型文件。

1. FLA kernel grid 轴拆分（fused_sigmoid_gating_recurrent.py、fused_recurrent.py） - 原 fused_sigmoid_gating_delta_rule_update 的 grid 为 (NK, NV, N * HV)，fused_recurrent_kda_packed_decode 为 (NV, B * HV)；当 batch 与 head 数量乘积超过 65535 时超出 CUDA grid.y/z 上限，kernel 启动失败。 - 现拆分为 (NV, N, HV) 三轴，N 与 HV 独立成轴；kernel 内部通过新增的 SPLIT_N_HV_GRID 常量表达式区分 tl.program_id 的解析方式，保持同一份 kernel 代码兼容两种启动方式。 - GPU wrapper 侧断言 NK == 1，因此拆分路径下 k 轴固定为 0。 - 影响：大 decode batch（如 kimi-k3 的 day0 场景）不再受 grid 轴上限约束。
2. cache stride 推导修正（fused_sigmoid_gating_recurrent.py） - 原实现 cache_stride_steps 从 intermediate_states_buffer.shape[1] 取值；但 --speculative-adaptive 下运行时 draft 数会变化，shape 与实际分配不符。 - 改为从 intermediate_states_buffer.stride(0) // (HV * K * V) 推导，因为 per-request pitch 在分

配时固定；无 buffer 时回退到 cache_steps 参数，最后回退 0，并保留该参数以维持 API 兼容。

3. silu_mul_clamp 多 block-per-row（silu_and_mul_masked_post_quant.cuh） - 原内核要求 out_dim / kVecSize <= 1024（单 block 覆盖一行），hidden dim 超过约 16K（fp8 场景）无法启动。 - 现按 blocks_per_row = host::div_ceil(out_vecs, 1024) 拆成多 block，总 grid 从 num_tokens 放大为 num_tokens * blocks_per_row；blockIdx.x 解码出行号与行内 block 序号，vec_id < out_vecs 时写入对应偏移，整体仍保持 PDL 等待 / 触发语义。 - 新增 to_bf16x2 模板函数，允许 bf16x2 与 fp32x2 等其他类型统一转成 bf16x2 参与 silu_and_mul，保证跨 DType 数值一致，并给 SiluAndMulClampParams 增加 out_vecs 与 blocks_per_row 字段。
4. vision.py seq_lens 统一转换 - seq_lens = seq_lens.to(device=q.device, dtype=torch.int32) 从 else 分支移出，无论 sequence_lengths 是否提供（或从 cu_seqlens 推导），都统一转成 int32 并放置于 q.device，消除视觉注意力路径的类型 / 设备不一致隐患。
5. cumsum.py autotune 收敛 - chunk_local_cumsum_scalar_kernel 的 autotune 配置移除 num_stages 搜索维度（从 [2, 3, 4] 收窄为默认值），缩小配置空间、减少 autotune 开销，属于移植时顺带的简化。

测试与部署配套：本次没有新增或修改测试文件；验证主要依赖源 PR #36507 在 day0 分支的 kimi-k3 测试通过。三条 CI 流水线（PR Test / Extra / AMD ROCm 7.2）均显示未通过状态，需关注其失败原因。

关键文件：

- python/sglang/kernels/ops/attention/fla/fused_sigmoid_gating_recurrent.py（模块 内核层；类别 infra；类型 infrastructure；符号 fused_sigmoid_gating_delta_rule_update_kernel, fused_sigmoid_gating_delta_rule_update）：本 PR 最核心的改动：FLA delta-rule 更新 kernel 的 grid 由 (NK, NV, N*HV) 拆为 (NV, N, HV)，新增 SPLIT_N_HV_GRID 常量表达式分支；cache_stride_steps 改为从 stride(0) 推导以适配 adaptive speculative decoding 下运行时 draft 数的变化。
- python/sglang/kernels/jit/csrc/deepseek_v4/silu_and_mul_masked_post_quant.cuh（模块 内核层；类别 other；类型 dependency-wiring；符号 silu_mul_clamp_kernel, SiluAndMulClampKernel, to_bf16x2）：DeepSeek-V4 MoE 的 SiLU 与乘法限幅 CUDA 内核从单 block-per-row 放宽为多 block 协作，解除 out_dim 对 1024 线程上限的依赖；新增 to_bf16x2 统一跨 DType 转换，并扩展 SiluAndMulClampParams 结构。
- python/sglang/srt/layers/attention/vision.py（模块 视觉注意力；类别 source；类型 core-logic）：视觉注意力 forward 路径中 seq_lens 的 int32/ 设备转换从 else 分支移出，保证无论 sequence_lengths 是否提供都统一类型与设备，消除多模态路径的类型不一致隐患。
- python/sglang/kernels/ops/attention/fla/fused_recurrent.py（模块 内核层；类别 infra；类型 infrastructure；符号 fused_recurrent_kda_packed_decode_kernel, fused_recurrent_kda_packed_decode）：KDA packed decode kernel 的 grid 从 (NV, B*HV) 拆为 (NV, B, HV)，与 sigmoid gating 的改动配套，同样规避大 batch 场景下 grid 轴超限。

- python/sglang/kernels/ops/attention/fla/cumsum.py（模块 内核层；类别 infra；类型 infrastructure）：chunk_local_cumsum 的 autotune 配置移除 num_stages 搜索维度，收敛配置空间、减少 autotune 开销，是移植时顺带的简化。

关键符号：fused_sigmoid_gating_delta_rule_update_kernel, fused_sigmoid_gating_delta_rule_update, fused_recurrent_kda_packed_decode_kernel, fused_recurrent_kda_packed_decode, silu_mul_clamp_kernel, SiluAndMulClampKernel::operator(), to_bf16x2

关键源码片段

python/sglang/kernels/ops/attention/fla/fused_sigmoid_gating_recurrent.py

本 PR 最核心的改动：FLA delta-rule 更新 kernel 的 grid 由 (NK, NV, N*HV) 拆为 (NV, N, HV)，新增 SPLIT_N_HV_GRID 常量表达式分支；cache_stride_steps 改为从 stride(0) 推导以适配 adaptive speculative decoding 下运行时 draft 数的变化。

```
# 启动入口：仅在 CUDA 上启用 N/HV 轴拆分，其余后端保持原启动方式。
# 原实现把 batch 与 head 折叠进同一 grid 轴（N * HV），当大 decode batch
# 的乘积超过 CUDA grid.y/z 的 65535 上限时 kernel 会启动失败；
# 拆分后 N 与 HV 各占一个轴，kernel 内部用 SPLIT_N_HV_GRID 区分 program_id 解析。
split_n_hv_grid = q.device.type == "cuda"
grid = (NV, N, HV) if split_n_hv_grid else (NK, NV, N * HV)

if SPLIT_N_HV_GRID:
    i_v, i_n, i_hv = tl.program_id(0), tl.program_id(1), tl.program_id(2)
    # GPU wrapper 断言 NK == 1，因此 k 轴固定为 0
    i_k = 0
else:
    i_k, i_v, i_nh = tl.program_id(0), tl.program_id(1), tl.program_id(2)
    i_n, i_hv = i_nh // HV, i_nh % HV

# 缓存步数改用 stride(0) 推导：--speculative-adaptive 会改变运行时 draft
# 个数，但分配时固定的 per-request pitch（stride(0)）不变；原实现从
# intermediate_states_buffer.shape[1] 取值，在自适应投机下会与分配不一致。
if intermediate_states_buffer is not None:
    cache_stride_steps = intermediate_states_buffer.stride(0) // (HV * K * V)
elif cache_steps is not None and cache_steps > 0:
    cache_stride_steps = cache_steps
else:
    cache_stride_steps = 0

# 把拆分开关作为常量表达式传入 kernel，触发 Triton 特化编译
SPLIT_N_HV_GRID=split_n_hv_grid,
```

python/sglang/kernels/jit/csrc/deepseek_v4/silu_and_mul_masked_post_quant.cuh

DeepSeek-V4 MoE 的 SiLU 与乘法限幅 CUDA 内核从单 block-per-row 放宽为多 block 协作，解除 out_dim 对 1024 线程上限的依赖；新增 to_bf16x2 统一跨 DType 转换，并扩展

SiluAndMulClampParams 结构。

```
// silu_mul_clamp_kernel：放宽为一行多 block 协作处理。
// 原实现要求 out_dim / kVecSize <= 1024（单 block 覆盖一行），
// 放宽后通过 blockIdx.x 解出行号与行内 block 序号，任意 out_dim 均可启动。
const auto row = blockIdx.x / params.blocks_per_row;
const auto block_in_row = blockIdx.x % params.blocks_per_row;
const auto vec_id = block_in_row * blockDim.x + threadIdx.x;
const float limit = params.swiglu_limit;

PDLWaitPrimary<kUsePDL>();
if (vec_id < params.out_vecs) {
    const auto input = static_cast<const Vec*>(params.input);
    auto output = static_cast<Vec*>(params.output);
    // 输入按 [row, 2, out_vecs] 布局：前半是 gate，后半是 up
    const auto input_row = row * 2 * params.out_vecs;
    const auto gate = input[input_row + vec_id];
    const auto up = input[input_row + params.out_vecs + vec_id];
    Vec out;
    // 逐向量执行 silu(gate) * up 并按 swiglu_limit 限幅；
    // to_bf16x2 把 fp8 等类型统一转成 bf16x2 参与计算，保证跨 DType 数值一致
    #pragma unroll
    for (uint32_t i = 0; i < kVecSize / 2; ++i) {
        out[i] = cast<DType2>(silu_and_mul<true>(to_bf16x2(gate[i]), to_bf16x2(up[i]), limit));
    }
    output[row * params.out_vecs + vec_id] = out;
}
PDLTriggerSecondary<kUsePDL>();

// 启动侧：每行最多 1024 线程，超出部分用 blocks_per_row 拆成多个 block，
// 总 grid 从 num_tokens 放大为 num_tokens * blocks_per_row
const auto out_vecs = out_dim / kVecSize;
const auto num_threads = std::min(out_vecs, 1024u);
const auto blocks_per_row = host::div_ceil(out_vecs, num_threads);
```

评论区精华

仅在批准时有一条评论，由审查者 Fridge003 发出。

> Should be OK, since kimi-k3 passed in the day0 branch ...

三条 CI 流水线（PR Test / Extra / AMD ROCm 7.2）均显示未通过，但 Fridge003 以源 PR #36507 在 day0 分支的 kimi-k3 测试通过为依据批准合并，说明 main 分支 CI 失败不影响对该变更正确性的判断。这也侧面印证：本次移植的目标场景（kimi-k3 等新模型）已在 day0 专用分支获得端到端验证。

- CI 失败情况下以 day0 分支验证作为合并依据 (testing): 以源 PR #36507 在 day0 分支的 kimi-k3 测试结果作为正确性依据，批准合并；main 分支 CI 失败未被追查。

风险与影响

- 风险：
  1. 无配套测试：本次没有新增或修改测试文件，回归验证依赖源 PR day0 分支的 kimi-k3 结果，main 分支上其他模型（GLM-5、DeepSeek-V4 等）的大 shape 场景未被直接覆盖。
  2. CI 未通过：三条流水线（PR Test / Extra / AMD ROCm 7.2）均为 X 状态，失败原因未在 PR 中说明，合并依据是外部分支验证；若失败与本次改动相关，main 上其他路径可能受同样问题影响。
  3. CUDA 与非 CUDA 行为分叉：SPLIT_N_HV_GRID 仅在 q.device.type == "cuda" 时开启，其他硬件后端（ROCm、XPU、NPU）仍走原 (NK, NV, N * HV) 路径，两套 program_id 解析逻辑需长期共存维护，回归面扩大。
  4. stride 推导的隐含假设：cache_stride_steps 依赖 stride(0) // (HV * K * V) 能整除且内存连续，若未来中间 buffer 布局变化（如非连续张量）可能静默算错。
  5. grid 轴拆分后的新约束：拆分后每个轴（NV、N、HV）各自仍需小于 65535，虽然一般情况下 batch 与 head 数量远低于该值，但极端场景的约束边界发生了变化，需要内核作者在后续维护中注意。
  6. cumsum autotune 收窄：移除 num_stages 搜索维度后，个别 shape 的 autotune 结果可能产生微小性能回退，但风险较低。 - 影响：受影响的内核被 FLA 线性注意力与 MoE 两个共享路径复用：放宽上限后，大 batch decode（grid 轴拆分）与大 hidden dim（多 block-per-row）的模型可以正常运行，直接受益方是 kimi-k3、GLM-5 等新模型的 day0 支持；vision.py 的 seq_lens 统一转换消除了视觉注意力路径在 sequence_lengths 缺省与否两种情况下类型 / 设备不一致的隐患，属于对多模态路径的稳定性加固。对团队而言，这是一次把 day0 专用分支的 kernel 改造通用化回 main 的移植，后续其他内核遇到 CUDA grid 上限或 block 数限制时，可以直接复用本 PR 的拆分模式。 - 风险标记：CI 三条流水线未通过，缺少配套测试文件，共享内核影响面广，CUDA 与非 CUDA 行为分叉

关联脉络

- PR #36507 （源 PR，标题未在本次材料中提供）：本 PR 是其移植，PR body 明确注明 Ported from #36507，审查者的合并依据也来自该 PR 在 day0 分支的 kimi-k3 测试验证。
- PR #36933 [2/N][Mixed] Mixed chunk prefill with spec enabled: 本 PR 中 cache_stride_steps 改为从 stride(0) 推导，正是为了适配 adaptive speculative decoding 下运行时 draft 数变化，与投机解码路径的调度改造相关。
- PR #37156 [Diffusion] Fuse Qwen-Image FP8 norm and activation quantization: 同属 kernel 层量化优化主线（quant + jit-kernel），体现 sglang 近期在共享内核能力扩充上的持续投入。