

PR #39405 完整报告

sgl-project/sglang

[misc] Revert #38346, #33426, #39061 and #39219

合并时间: 2026-09-14 18:04

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39405>

执行摘要

- 一句话: 批量回退四个误合并 PR, 恢复注意力后端与内核旧行为
- 推荐动作: 值得精读, 但重点不在代码, 而在理解“哪四个改动被摘掉、各自为什么值得重新提交”。建议: 1) 工程师只需读本报告的关键源码片段即可掌握本次净 diff, 不必逐行比对 9 个文件; 2) 重点关注 #39219 的 conv_state 竞态与 #39061 的 Dynamo graph break, 这两处在重新提交前必须重新验证 (最好附上 green-context 复现与 torch.compile 全图捕获测试); 3) 在重新提交 #33426 前先 grep 全仓是否已有 return_lse=True 的消费方, 避免扩展点回归时出现“加完没有使用者”或“使用者没有测试”的情况; 4) #38346 需要作者补充越界复现, 否则不建议恢复 clamp; 5) 把三路 CI 红灯的原因查清后再重新提交四个 PR, 避免把既有失败带入新一轮 review。

功能与动机

PR body 给出了唯一动机: “These four PRs were merged by mistake (by @alexsnails) before review and CI had completed. Reverting them so they can be re-submitted through the normal review flow.” 即问题不在改动本身的技术价值, 而在于它们绕过了 review 与 CI 流程, 维护者选择先把 main 恢复到合入前状态, 让 @ehuua、@klshuster、@janbernloehr、@mmangkad 等原作者按正常流程重新提交。关联 issue 与 issue 评论均为空, 因此没有额外的需求背景可供引用; 本 PR 没有任何 review 讨论。

实现拆解

回退按“一个 PR 一个 commit”的机械方式完成, 没有新增适配层或兼容 shim; 下列步骤按被回退的目标 PR 组织, 文件归属由 PR 内容与 patch 反推 (每个 commit 的独立 diff 未在上下文中给出)。

总体策略: 逐 PR revert, 不夹带修复

- 4 个 commit 依次为 revert #39219、revert #39061、revert #33426、revert #38346, 全部由 hnyls2002 完成, head 提交 e957ed01 即最后一个 revert。
- 只做还原: 9 个文件中 4 个是测试文件且改动几乎全是删除 (测试侧净 -155 行), 源码侧净 -14 行, 整体 +50/-219。
- PR body 的 CI States 区块显示三路测试 (PR Test Base 、PR Test Extra 、AMD ROCm 10 ) 全部失败, 回退并未带回绿灯状态。

1. 收回 FA 后端的扩展点 (revert #33426)

- python/sglang/srt/layers/attention/flashattention_backend.py: forward_extend / forward_decode 删除 return_lse: bool = False 形参、lse_out 变量、elif return_lse: o, lse_out, *_ = result 分支, 以及结尾 o, lse_out.transpose(0, 1).contiguous() 的二元组返回; return_softmax_lse=use_cascade_attn or return_lse 退回 return_softmax_lse=use_cascade_attn。
- 同一文件把内核调用从 self.flash_attn_with_kvcache / self.flash_attn_varlen_func (__init__ 绑定, 原注释写明 “so a subclass can substitute a different FA4 build”) 改回模块级 from sglang.kernels.ops.attention.flash_attention import flash_attn_varlen_func, flash_attn_with_kvcache, 子类替换内核的能力随之消失。
- test/registered/attention/unittests/dense/test_fa4.py 删除 RETURN_LSE_CASES、_reference_out_and_lse、test_return_lse (-100 行), LSE 的 backend 级契约测试一并撤销。
- 影响面: 任何依赖 return_lse=True 或依赖覆写 self.flash_attn_* 的子类 / 调用方会失效或直接 TypeError, 重新提交该 PR 时应把消费方一起带上。

2. 收回平台探测的编译安全写法 (revert #39061)

- python/sglang/srt/utils/common.py: is_musa() 由“模块级 _IS_MUSA 常量 + @lru_cache 读取”退回“函数体内 try: import torch.cuda”, 被编译的调用点会在 import 处触发 Dynamo graph break。
- python/sglang/srt/layers/attention/dsa/utils.py: is_dsa_enable_prefill_cp() 里 attn_cp_size <= 1 的短路判断被移回 is_hip() / is_npu() / is_musa() 探测之后, 于是 CP 未启用的常规配置也会走平台探测。
- 配套测试 test/registered/unit/utils/test_common.py::TestMusaDetection::test_is_musa_is_torch_compile_safe 与 test/registered/cp/test_cp_strategy_unit.py::test_disabled_dsa_cp_skips_platform_probes 被删除, “编译安全”和“探测短路”两个不变量在 CI 中不再有守卫。

3. 恢复 KDA verify 内核的 conv_state 写回 (revert #39219)

- python/sglang/kernels/ops/attention/fla/fused_kda_conv_recurrent_verify.py 重新加入 if is_qk_owner: 下 9 条 tl.store(cs_base ...), 把当前窗口写回 conv_state; docstring 中 “State: conv_state and the SSM state are read-only” 被替换为“滚动窗口即最后 W-1 个 token, 与参考 kernel 的 store 一致”, num_warps 注释也回到“conv_state 与参考实现 bit-aligned”的说法。
- 这正是原修复要消除的 CTA 调度竞态: H=1、HV=16 时 16 个 V tile 共享同一段 Q/K 历史, 后一波 wave 的 tile 会读到前一波 tile 覆盖后的 conv_state。
- test/registered/kernels/test_fused_kda_conv_recurrent_verify.py 把 assert torch.equal(inp["conv_pool"], conv_fus) 改回 conv_ref[touched_slots] == conv_fus[touched_slots], 并删除 test_output_does_not_depend_on_cta_scheduling (用 green context 压缩 SM 复现调度差异), 竞态在测试侧同样失去守卫。

4. 恢复 QSA indexer 不做 clamp (revert #38346)

- python/sglang/srt/layers/attention/qa/qa_indexer.py::update_key_state_and_compress 删除 group_locs = group_locs.clamp_max(source_keys.shape[0] - 1) 一行，group_locs 重新可能与 source_keys 的边界不一致。
- 38346 的标题与 body 未在上下文中提供，其改动只能从这一行反推；重新提交时需要补上越界场景的复现说明。

测试、配置与部署配套

- 无新增测试、无配置或依赖变更；内核文件 fused_kda_conv_recurrent_verify.py 是本次唯一涉及 kernel 代码的还原 (+30/-17)，其余源码文件的改动均以删除为主。
- 也就是说，这次回退没有提供任何回归验证手段，只把风险和修复一起交还给原作者。

关键文件：

- python/sglang/srt/layers/attention/flashattention_backend.py (模块 注意力后端；类别 source；类型 core-logic；符号 FlashAttentionBackend.forward_extend, FlashAttentionBackend.forward_decode, FlashAttentionBackend.init)：回退 #33426 的主体：删除 return_lse 形参与 LSE 返回分支，并把子类可替换的 self.flash_attn_* 绑定改回模块级导入，直接改变 FA 后端的对外契约。
- python/sglang/kernels/ops/attention/fla/fused_kda_conv_recurrent_verify.py (模块 线性注意力；类别 source；类型 core-logic；符号 fused_kda_conv_gating_verify_kernel, fused_kda_conv_gating_verify)：回退 #39219 的主体：把 verify 阶段的 conv_state 写回与“只读 state”的 docstring 一并恢复，重新引入跨 wave 覆盖 conv_state 的 CTA 调度竞态。
- python/sglang/srt/utils/common.py (模块 设备探测；类别 source；类型 dependency-wiring；符号 is_musa)：回退 #39061 的主体：is_musa() 的 torchada 探测回到函数体内，重新暴露 torch.compile 下的 Dynamo graph break 风险。
- python/sglang/srt/layers/attention/dsa/utils.py (模块 预填充并行；类别 source；类型 core-logic；符号 is_dsa_enable_prefill_cp)：is_dsa_enable_prefill_cp() 恢复“先探测平台再判 CP 规模”的顺序，CP 关闭时也会调用 is_hip / is_npu / is_musa。
- python/sglang/srt/layers/attention/qa/qa_indexer.py (模块 索引器；类别 source；类型 core-logic；符号 update_key_state_and_compress)：回退 #38346 的唯一文件：去掉 group_locs 的 clamp_max 上界保护，恢复不对索引做裁剪的旧行为。
- test/registered/attention/unittests/dense/test_fa4.py (模块 注意力测试；类别 test；类型 test-coverage；符号 RETURN_LSE_CASES, _reference_out_and_lse, test_return_lse)：删除 100 行 LSE 契约测试 (RETURN_LSE_CASES / _reference_out_and_lse / test_return_lse)，是 return_lse 能力的唯一测试守卫。
- test/registered/kernels/test_fused_kda_conv_recurrent_verify.py (模块 内核测试；类别 test；类型 test-coverage；符号 _compare_case, test_output_does_not_depend_on_cta_scheduling)：把 conv_pool 只读断言改回按 slot 比较，并删除 green-context 调度不变量测试，KDA verify 竞态失去测试守卫。
- test/registered/unit/utils/test_common.py (模块 工具测试；类别 test；类型 test-coverage；符号 TestMusaDetection, test_is_musa_is_torch_compile_safe)：删除 TestMusaDetection.test_is_musa_is_torch_compile_safe，编译安全不变量失去守卫。

- test/registered/cp/test_cp_strategy_unit.py (模块 并行测试; 类别 test; 类型 test-coverage; 符号 test_disabled_dsa_cp_skips_platform_probes) : 删除 test_disabled_dsa_cp_skips_platform_probes, 平台探测短路行为失去守卫。

关键符号: FlashAttentionBackend.forward_extend,
FlashAttentionBackend.forward_decode, is_musa, is_dsa_enable_prefill_cp,
fused_kda_conv_gating_verify_kernel, fused_kda_conv_gating_verify,
update_key_state_and_compress, _reference_out_and_lse, test_return_lse,
test_output_does_not_depend_on_cta_scheduling, test_is_musa_is_torch_compile_safe,
test_disabled_dsa_cp_skips_platform_probes

关键源码片段

python/sglang/srt/layers/attention/flashattention_backend.py

回退 #33426 的主体: 删除 return_lse 形参与 LSE 返回分支, 并把子类可替换的 self.flash_attn_* 绑定改回模块级导入, 直接改变 FA 后端的对外契约。

```
# python/sglang/srt/layers/attention/flashattention_backend.py (本回退后的形态)
# 1) 内核函数重新在模块顶层导入: 不再有 self.flash_attn_with_kvcache 这类
# 实例绑定, 子类无法再替换 FA4 实现;
# 2) forward_extend / forward_decode 不再接受 return_lse, 也不再返回 (o, lse)。
from sgl_kernel import merge_state_v2
```

```
from sglang.kernels.ops.attention.flash_attention import (
    flash_attn_varlen_func,
    flash_attn_with_kvcache,
)
```

```
class FlashAttentionBackend(AttentionBackend):
    def forward_extend(
        self,
        q,
        k,
        v,
        layer,
        forward_batch,
        save_kv_cache=True,
        q_rope=None, # MLA 相关入参保持不变
        k_rope=None,
        sinks=None,
        q_descscale=None,
        k_descscale=None,
        v_descscale=None,
        score_mod=None,
        aux_tensors=None,
        rel_bias=None,
        rel_bias_event=None,
    ): # 回退前这里还有 return_lse: bool = False, 返回类型允许是 torch.Tensor | tuple[...]
```

```

if score_mod is not None and self.fa_impl_ver != 4:
    raise RuntimeError("score_mod is only supported by the FA4 backend.")

# ... (KV 写入、metadata 准备、cascade attention 分支) ...
result = flash_attn_with_kvcache(
    q=q.contiguous().view(-1, layer.tp_q_head_num, layer.head_dim),
    k_cache=key_cache,
    v_cache=value_cache,
    # ...
    # 只有 cascade attention 需要 LSE; 回退前写法是
    # return_softmax_lse=use_cascade_attn or return_lse
    return_softmax_lse=use_cascade_attn,
    num_splits=self.num_splits,
    out=_fa_out,
    ver=self.fa_impl_ver,
)

# 回退前的 `elif return_lse: o, lse_out, *_ = result` 分支已不存在,
# 因此函数出口只剩单一 Tensor。
return o.view(-1, layer.tp_q_head_num * layer.v_head_dim)

# forward_decode 的改动与 forward_extend 完全对称:
# 删 return_lse 形参、删 return_softmax_lse 里的 or return_lse、
# 删结尾 `return o, lse_out.transpose(0, 1).contiguous()`。

```

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

回退 #39219 的主体: 把 verify 阶段的 conv_state 写回与“只读 state”的 docstring 一并恢复, 重新引入跨 wave 覆盖 conv_state 的 CTA 调度竞态。

```

# python/sglang/kernels/ops/attention/fla/fused_kda_conv_recurrent_verify.py
# (本回退后的形态)
# 被回退的修复原本把下面这段写回整段删掉, 并留下注释:
# "No conv-state writeback: every V tile reads the same Q/K history,
# so a tile in a later wave would read what i_v == 0 had overwritten."
# 回退后写回恢复: 同一个 Q/K 历史被多个 V tile 共享, 后一波 wave 的 tile
# 可能读到前一波 tile 覆盖过的 conv_state, 验证输出因此可能依赖 CTA 调度顺序。
tl.store(cache_ptr, b_h.to(cache_ptr.dtype.element_ty), mask=mask_h)

# Rolled conv state after consuming T >= W-1 tokens is exactly the last
# W-1 input tokens — which are the current window registers. The verify
# pass never writes the ssm state back (rollback happens at commit).
if is_qk_owner:
    tl.store(cs_base + q_ch + 0 * stride_cs_tok, q_c0, mask=mask_k)
    tl.store(cs_base + q_ch + 1 * stride_cs_tok, q_c1, mask=mask_k)
    tl.store(cs_base + q_ch + 2 * stride_cs_tok, q_c2, mask=mask_k)
    tl.store(cs_base + k_ch + 0 * stride_cs_tok, k_c0, mask=mask_k)
    tl.store(cs_base + k_ch + 1 * stride_cs_tok, k_c1, mask=mask_k)
    tl.store(cs_base + k_ch + 2 * stride_cs_tok, k_c2, mask=mask_k)

```

```

tl.store(cs_base + v_ch + 0 * stride_cs_tok, v_c0, mask=mask_v)
tl.store(cs_base + v_ch + 1 * stride_cs_tok, v_c1, mask=mask_v)
tl.store(cs_base + v_ch + 2 * stride_cs_tok, v_c2, mask=mask_v)

# 包壳函数的默认实现也被改回，注释从“conv_state 不参与比对”变成
# “输出 / conv_state / conv-window 全部与参考实现 bit-identical”：
def fused_kda_conv_gating_verify(..., num_warps: int = 4) -> torch.Tensor:
    # num_warps=4 is ~1.3x faster than the unfused pair in-graph; the output,
    # conv_state and conv-window caches stay bit-identical to the reference.
    # ...
    # num_warps=1 reproduces the reference reduction order exactly (all buffers
    # bit-identical) but is ~2.4x slower in-graph — numerics debugging only.
    ...

```

python/sglang/srt/utils/common.py

回退 #39061 的主体：is_musa() 的 torchada 探测回到函数体内，重新暴露 torch.compile 下的 Dynamo graph break 风险。

```

# python/sglang/srt/utils/common.py (本回退后的形态)
# 回退前的写法把探测结果提升为模块级常量 _IS_MUSA:
# try:
# import torchada
# except ImportError:
# _IS_MUSA = False
# else:
# _IS_MUSA = hasattr(torch.version, "musa") and torch.version.musa is not None
# 这样 is_musa() 只读取一个 bool，torch.compile(fullgraph=True) 不会因为
# 函数体里的 import 而 graph break；回退后 import 又回到函数体内。
@lru_cache(maxsize=1)
def is_musa() -> bool:
    try:
        import torchada # noqa: F401
    except ImportError:
        return False
    return hasattr(torch.version, "musa") and torch.version.musa is not None

# 调用点（如 sglang/srt/layers/attention/dsa/utils.py::is_dsa_enable_prefill_cp)
# 一旦在编译区里触发 is_musa()，就会命中上面的 import 造成图断点。

```

评论区精华

本 PR 的 `comments_count = 0`、`review_comments_count = 0`，没有任何 line-level review 讨论，作者与审核者也没有留下技术论证。唯一可引用的决策文本来自 PR body：

These four PRs were merged by mistake (by @alexnails) before review and CI had completed. Reverting them so they can be re-submitted through the normal review flow. cc @ehuaa @klshuster @janbernloehr @mmangkad

CI States: Latest PR Test (Base) 🚫 Run #34831107614; PR Test (Extra) 🚫 Run #34831107357; AMD ROCm 10 🚫 Run #34831107821

由此可见：1) 决策是流程性的，没有争论“该不该回退”；2) 四个原作者没有被要求在本 PR 内回应，重新提交时应当各自补上缺陷复现与验证证据；3) 三路 CI 全红这一事实没有在 PR 里被解释，是本 PR 最大的信息缺口。

- 四个 PR 因提前合并而被整体回退的流程处置 (other): 以 4 个独立 commit 分别 revert 四个 PR，把 main 恢复到合入前状态，等待作者按正常流程重新提交；回退本身不夹带修复，也不补测试。

风险与影响

- 风险：1) 回退 #39219 重新引入 CTA 调度竞态：fused_kda_conv_recurrent_verify.py 恢复的 9 条 tl.store(cs_base ...) 让 verify 阶段改写 conv_state，而 H=1、HV=16 的形态下同一段 Q/K 历史被 16 个 V tile 共享，后一波 wave 的 tile 可能读到已被覆盖的状态，输出理论上可随 SM 调度变化；守卫该不变量的 test_output_does_not_depend_on_cta_scheduling 同时被删除，缺陷将不可见地进入主线。2) 回退 #39061 重新引入 Dynamo graph break: is_musa() 的函数体内 import torchada 在被 torch.compile(fullgraph=True) 捕获的路径 (prefill 图捕获、DSA CP 判定) 中会打断图，可能表现为 prefill 崩溃；dsa/utils.py 把平台探测提前后，CP 未启用 (attn_cp_size <= 1) 的常规配置也会调用 is_hip()/is_npu()/is_musa()，扩大了触发面。3) 回退 #33426 收窄了 FA 后端 API: forward_extend / forward_decode 不再接受 return_lse，也不再返回 (o, lse)；若仓库其他位置 (含本 PR 未列出的子类或配置分支) 仍以关键字方式传入 return_lse=True，会立刻报 TypeError，而 test_fa4.py 中的唯一契约测试已被删除，这类错误只能在集成层暴露。同时 self.flash_attn_* 绑定点消失，子类替换 FA4 内核的能力被收回。4) 回退 #38346 去掉 group_locs.clamp_max(...)：若 group_locs 在某些输入下确实可能超出 source_keys 行数，QSA indexer 会重新出现越界读 / 错位索引，且该 PR 的动机在上下文中不可考，属于“把未解释的防护拿掉”。5) CI 三路 (PR Test Base / Extra / AMD ROCm 10) 均为失败状态且未在本 PR 内解释，需要确认是既有 flaky 还是回退引入了编译 / 导入错误 (例如仍有代码引用 _IS_MUSA 或 return_lse)。6) 过程风险：main 分支在回退与重新提交之间会同时缺少两个 bugfix，期间使用 MUSA/torchada 平台或 KDA 线性注意力 MTP 验证路径的用户可能命中旧缺陷。
- 影响：用户侧：影响集中在四类用法——依赖 FA 后端 return_lse 或覆写 self.flash_attn_* 的调用方 / 子类 (可能运行期报错)、MUSA/torchada 平台上的编译式 prefill (可能崩溃)、KDA/ 线性注意力模型的 MTP verify 路径 (输出可能对 CTA 调度敏感)、QSA indexer 使用方 (失去上界保护)。普通 CUDA 文本生成路径基本不受影响。系统侧：FA 注意力后端对外的函数签名与返回形态回退，属于 API 层面的可见变化；内核侧 conv_state 从只读恢复为可写，改变了 verify 阶段的副作用契约。团队侧：四个 PR (作者分别为 ehuaa、klshuster、janbernloehr、mmangkad 相关) 需要重新 rebase、重新走 review 与 CI，短期需要协调重复提交与冲突处理；同时 main 会短暂回到“已有修复被摘掉”的状态，集成测试若有依赖会先行报警。整体影响面中等偏高但可控，因为回退本身是机械的、没有引入新逻辑。

- 风险标记：回退已修复竞态，回退已修复崩溃，公共 API 收回 (return_lse)，移除防御性 clamp, CI 三路红灯未解释，无 review 讨论，待重新提交

关联脉络

- PR #39219 [Fix] Don't write conv state from the fused KDA verify kernel: 本次被回退的目标之一。它删掉了 verify 内核里跨 wave 覆盖 conv_state 的写回，并新增 green-context CTA 调度不变量测试；回退后该竞态与缺口同时回到主线。
- PR #39061 Fix MUSA detection in compiled prefill path: 本次被回退的目标之一。它把 torchada 探测提升到模块级并调整 DSA CP 的探测顺序，回退后 is_musa() 的函数体内 import 重新可能造成 Dynamo graph break。
- PR #33426 [fa] Make the FlashAttention backend extensible by subclasses: 本次被回退的目标之一。它引入了 return_lse 扩展点与 self.flash_attn_* 子类替换绑定，回退后 FA 后端 API 收窄、LSE 契约测试一并删除。
- PR #38346 标题未在上下文中提供（改动 QSA indexer 的 group_locs clamp）：本次被回退的目标之一。改动只落在 python/sglang/srt/layers/attention/qa/qa_indexer.py，回退后 group_locs 不再做 clamp_max 上界裁剪。
- PR #37418 [unified-memory] Enable prefill cuda-graph capture: 同文件关联：同样改动 python/sglang/srt/layers/attention/flashattention_backend.py（统一内存池 prefill CUDA Graph）。回退 #33426 会改回该文件的 import 与内核调用形态，需确认其 prefill 图捕获路径没有依赖 self.flash_attn_* 绑定或 return_lse。