PR #30519 完整报告

sgl-project/sglang

[AMD] [GLM5] fp8 MLA absorbed bmm for GLM-5.2 on gfx950

合并时间：2026-08-17 17:15

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30519

执行摘要

- 一句话：gfx950 上 GLM-5.2 MLA absorbed bmm 切到 fused fp8 内核，TPUT 最高 +14%
- 推荐动作：该 PR 值得精读，两个设计点有复用价值：(1) 用加载期数据变换（bf16 → fp8_e4m3fn）去凑齐下游 kernel 选择 gate 的隐式契约，相当于在权重加载层集中做数据流归一化；(2) 输出布局与下游融合算子联合设计，用 free view 取代 per-layer 拷贝。建议读者同时参考 amd-bot 在 Issue 中的 CI 覆盖分析，把它作为"硬件门控改动必须人工补测"的案例；1am9trash 的 NextN / MTP 扩展建议是自然的后续工作。

功能与动机

PR body 明确指出问题根源：在 gfx950 上，GLM（GlmMoeDsaForCausalLM）的 MLA 吸收权重（w_kc/w_vc）以 bf16 加载，导致 forward_mla 落入逐 batch 的 torch.bmm（rocBLAS）慢路径；而 DeepSeek 的 fp8 流程可以利用 aiter 融合内核。第二个动机是消除布局转换开销：原 fp8 absorbed V bmm 输出为 (heads, tokens, dim) 布局，下游 o_proj 准备阶段需要 transpose(0,1).flatten，产生非连续张量并触发 at::native direct_copy（elementwise_manual_unroll，MI355X 上约 5 微秒 /layer）。优化目标是把这两个开销都去掉：加载时量化权重以命中融合内核，并让 bmm 输出直接 batch-major 以便 flatten 成为 free view。

实现拆解

本 PR 的改动用 2 个文件、5 个步骤即可拆解清楚：

1. 加载端量化门控（deepseek_weight_loader.py）：在模块级新增 input_to_float8 导入（来自 sglang.srt.layers.quantization.fp8_utils）；在 post_load_weights() 中、切分 w_kc/w_vc 之前插入 GLM 专用 gate，条件是 _use_aiter_gfx95 且 architectures[0] == "GlmMoeDsaForCausalLM" 且 w.dtype == torch.bfloat16，命中后执行 w, self_attn.w_scale = input_to_float8(w, dtype=torch.float8_e4m3fn)。这一步把 w_kc/w_vc 变成 fp8_e4m3fn，同时把 scale 写入 self_attn.w_scale，使 forward 侧的 dtype gate（attn.w_kc.dtype == torch.float8_e4m3fn）命中，从而避开慢速 torch.bmm。该 gate 位于 DeepSeek quark gate 之前，但两者因架构名条件互斥，不会互相干扰。
2. forward 侧 batch-major 输出（forward_mla_rocm.py）：在 rocm_absorb_v_bmm() 的 fp8 分支中，原实现由 aiter 内核直接返回 (heads, tokens, dim) 布局的张量（transpose_bm=False），下游 transpose(0,1).flatten 产生非连续张量，触发每层一次

direct_copy。新实现仿照同函数内 mxfp4 分支的做法：预分配 _bmm_buf = torch.empty(batch, num_local_heads, v_head_dim, dtype=torch.bfloat16)，以 YQ=_bmm_buf + transpose_bm=True 让内核直接写入 batch-major 布局，随后 _bmm_buf.flatten(1, 2) 就是 free view，无需拷贝。

3. 门控与回归控制：所有快速路径均以 _use_aiter_gfx95 与权重 dtype（torch.uint8 走 mxfp4、torch.float8_e4m3fn 走 fp8）双条件判断，其他架构与模型的 torch.bmm 回退分支保持原样，行为不变。
4. 测试与验证配套：本 PR 未新增或修改任何测试文件；验证仅靠 PR body 中的 GSM8K 精度数据与 MI355X TP4 性能表，以及 reviewer 的独立复测。CI 状态为 CUDA PR Test Base 全绿、AMD CI 全绿，但如 amd-bot 在 Issue 中所分析，改动代码未被任何 PR-CI 用例命中（GLM-5.2 相关测试均为 nightly=True），需人工验证。作者随后在 Issue 评论中补充了 baseline 对比数据（+3.9% TPUT、−3.9% TPOT）。

关键文件：

- python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_rocm.py（模块 MLA 前向；类别 source；类型 core-logic；符号 rocm_absorb_v_bmm）：decode 热路径上的核心改动：fp8 absorbed V bmm 输出改为 batch-major 布局（transpose_bm=True + 预分配 _bmm_buf），使下游 flatten 成为 free view，消除每层约 5 微秒的 direct_copy，并把该分支与同函数内 mxfp4 分支的对齐。
- python/sglang/srt/models/deepseek_common/deepseek_weight_loader.py（模块 权重加载；类别 source；类型 data-contract；符号 post_load_weights）：加载期把 GLM-5.2 的 bf16 kv_b_proj 量化为 per-tensor fp8_e4m3fn，使 forward_mla_rocm 的 dtype gate 命中融合内核；这是本次性能收益成立的前提契约，也定义了 GLM 与 DeepSeek 加载分支的边界。

关键符号：rocm_absorb_v_bmm, post_load_weights

关键源码片段

python/sglang/srt/models/deepseek_common/attention_forward_methods/forward_mla_rocm.py

decode 热路径上的核心改动：fp8 absorbed V bmm 输出改为 batch-major 布局（transpose_bm=True + 预分配 _bmm_buf），使下游 flatten 成为 free view，消除每层约 5 微秒的 direct_copy，并把该分支与同函数内 mxfp4 分支的对齐。

```
# forward_mla_rocm.py —— rocm_absorb_v_bmm 的核心分支（PR 重构后）
def rocm_absorb_v_bmm(
    attn: DeepseekV2AttentionMLA,
    attn_output: torch.Tensor,
) -> torch.Tensor:
    """在 HIP 上融合 attn_output @ w_vc 与下游 flatten / 量化"""

    if _use_aiter_gfx95 and attn.w_vc.dtype == torch.uint8:
        # mxfp4 分支：输出按 (batch, heads, dim) 预分配
        x = attn_output.transpose(0, 1)
```

```
    B_heads, M_batch = x.shape[0], x.shape[1]
    N_vdim = attn.w_vc.shape[2]
    _bmm_buf = torch.empty(
        M_batch, B_heads, N_vdim,
        device=x.device,
        dtype=torch.bfloat16,
    )
    attn_bmm_output = _bmm_buf.transpose(0, 1)
    batched_gemm_afp4wfp4_pre_quant(
        x,
        attn.w_vc.transpose(-2, -1),
        attn.w_scale_v.transpose(-2, -1),
        torch.bfloat16,
        attn_bmm_output,
    )
else:
    _bmm_buf = None
    if _use_aiter_gfx95 and attn.w_kc.dtype == torch.float8_e4m3fn:
        # fp8 分支（本 PR 的主要改动）：复用 mxfp4 分支的布局技巧，
        # 用 transpose_bm=True 直接写 batch-major，下游 flatten 变 free view
        _bmm_buf = torch.empty(
            attn_output.shape[0],
            attn.num_local_heads,
            attn.w_vc.shape[-1],
            device=attn_output.device,
            dtype=torch.bfloat16,
        )
        batched_gemm_a8w8_a_per_token_group_prequant_w_per_batched_tensor_quant(
            X=attn_output,
            WQ=attn.w_vc.transpose(-1, -2),
            w_scale=attn.w_scale,
            group_size=128,
            YQ=_bmm_buf,
            transpose_bm=True,
            transpose_bm_in=True,
            dtype=torch.bfloat16,
        )
    else:
        # 回退：rocBLAS 慢路径，输出 (heads, tokens, dim)，下游必须 transpose
        attn_bmm_output = torch.bmm(
            attn_output.to(torch.bfloat16).transpose(0, 1),
            attn.w_vc.to(torch.bfloat16) * attn.w_scale,
        )

if _bmm_buf is not None:
    # _bmm_buf 本身就是 (batch, heads, dim) 连续布局，
    # flatten(1, 2) 是 free view，不需要再拷贝一整个中间张量
    if attn.o_proj.weight.dtype == torch.uint8:
        attn_bmm_output = fused_flatten_mxfp4_quant(_bmm_buf)
```

```
        elif _is_block_scale_fp8(attn.o_proj):
            attn_bmm_output = fused_flatten_fp8_group_quant(
                _bmm_buf,
                group_size=128,
                dtype_quant=torch.float8_e4m3fn,
                transpose_scale=False,
            )
        else:
            attn_bmm_output = _bmm_buf.flatten(1, 2)
    # 其余 elif 回退分支（transpose 后再 fused quant）从略
```

python/sglang/srt/models/deepseek_common/deepseek_weight_loader.py

加载期把 GLM-5.2 的 bf16 kv_b_proj 量化为 per-tensor fp8_e4m3fn，使 forward_mla_rocm 的 dtype gate 命中融合内核；这是本次性能收益成立的前提契约，也定义了 GLM 与 DeepSeek 加载分支的边界。

```
# deepseek_weight_loader.py —— post_load_weights() 中新增的 GLM-5.2 加载期量化
# 背景：GLM 的 kv_b_proj 以 bf16 下发，forward_mla 的 dtype gate 不命中，
# 只会走 torch.bmm 慢路径；这里在加载时转成 per-tensor fp8_e4m3fn。
if (
    _use_aiter_gfx95
    and self.config.architectures
    and self.config.architectures[0] == "GlmMoeDsaForCausalLM"
    and w.dtype == torch.bfloat16
):
    # 必须用 e4m3fn 而不是 e4m3fnuz：
    # forward_mla_rocm 的快速路径用 torch.float8_e4m3fn 做判定。
    w, self_attn.w_scale = input_to_float8(w, dtype=torch.float8_e4m3fn)

# 量化后照常按 MLA 头维度切分 w_kc / w_vc，上游调用方无需感知
w_kc, w_vc = w.unflatten(
    0, (-1, self_attn.qk_nope_head_dim + self_attn.v_head_dim)
).split([self_attn.qk_nope_head_dim, self_attn.v_head_dim], dim=1)
```

评论区精华

CI 覆盖诊断（amd-bot）：amd-bot 在回复 @amd-bot ci-status 时指出，PR CI 处于不完整状态（NVIDIA base-c-* 与 AMD stage-c-* 均 fast-fail 跳过），且“none of this PR's changed code is exercised by any PR-CI test”——两个编辑点都 gated 在 _use_aiter_gfx95 和 GlmMoeDsaForCausalLM 上，唯一 GLM-5.2-FP8 测试是 NVIDIA H200 的 nightly=True 用例，AMD 侧根本没有 GLM-5.2 测试。结论是 merge 前必须在 MI350x / gfx950 上人工跑 GLM-5.2-FP8 精度 + 性能验证。

精度与性能复测（clintg6 / Jacob0226）：clintg6 在 APPROVE 中报告“no impact on accuracy (95%) and 7% uplift on TPOT/TPUT for ISL/OSL 1k/1k at conc=1”，conc=8 时 TPUT +3%、TPOT −7%。Jacob0226 在 Issue 中补充独立 baseline（9beb01990a + 0714 docker）对比：输出 token 吞吐 +3.9%、TPOT −3.9%、GSM8K 无影响。

fp8 dtype gate 隐式契约（amd-bot 提醒）：amd-bot 特别强调 fp8 quant gate 要求的是 e4m3fn 而非 fnuz，需要确认 gfx950 上权重实际落地为 e4m3fn，否则 forward_mla 的 dtype gate 不匹配、快速路径会静默 no-op。作者已通过 input_to_float8(w, dtype=torch.float8_e4m3fn) 明确指定格式，但缺少自动化断言。

扩展建议（1am9trash）：Approver 建议把 fp8 quant-at-load 同样应用到 GlmMoeDsaForCausalLMNextN，让 MTP 层也避开 bf16 慢路径，但明确表示"Not a blocker for this PR."

冲突处理（Jacob0226）：sogalin 要求解决冲突后，作者合并 main 时发现 #31531 已把 HIP MLA absorb 路径迁入 forward_mla_rocm.py，因此 transpose_bm=True 改动随迁到该文件，GLM 权重量化逻辑保持不变。

- CI 覆盖诊断：改动未被任何 PR-CI 测试命中 (testing): merge 前必须在 MI350x（gfx950）上人工跑 GLM-5.2-FP8 精度 + 性能验证；作者提供了 GSM8K 与 baseline 对比数据，并建议后续把 GLM-5.2 纳入 nightly-amd-8-gpu-mi35x-glm*。
- 精度与性能回归复测 (correctness): 无显著回归，精度在误差范围内。
- fp8 dtype gate 隐式契约（e4m3fn vs e4m3fnuz） (correctness): 加载端与 forward 端 dtype 已对齐，但缺少自动化断言，未来格式迁移存在静默回退风险。
- 扩展 fp8 quant-at-load 到 NextN / MTP 层 (design): 留作后续工作，本 PR 不处理。
- 冲突处理与 HIP MLA absorb 路径迁移 (other): 已解决，改动落点从 forward_mla.py 迁移到 forward_mla_rocm.py。

风险与影响

- 风险：
  1. 数值精度风险（中）：per-tensor fp8 量化是有损变换，作用于 MLA 低秩注意力吸收路径。PR 内 GSM8K 为 0.928 → 0.922，作者补充对比 0.922 → 0.931，clintg6 报告 95% 无影响，结论可信但样本集中在单一任务。建议后续补充更多基准。
  2. dtype 隐式契约风险（中高）：快速路径依赖 forward_mla_rocm.py 中 attn.w_kc.dtype == torch.float8_e4m3fn 与加载端 dtype=torch.float8_e4m3fn 严格一致。若未来某处（类似 #35111 的 normalize 到 e4m3fnuz 逻辑）改变权重格式，快速路径会静默 no-op、退回慢路径且不报错，性能回归难以发现。
  3. 测试覆盖缺口（高）：无任何 PR-CI 用例命中这两个 gated 分支，CI 全绿不提供实际保护；后续修改可能无意破坏该路径而无人察觉。
  4. 共用加载器交互风险（低）：GLM gate 位于共用加载器 deepseek_weight_loader.py 的热路径上，虽用架构名限定并与 DeepSeek quark gate 互斥，但未来其他模型复用该加载器时仍需留意交互。
  5. 收益边界（低）：高并发与 70k prefill 重负载下收益趋近 0，说明优化只对 decode 低并发生效，不应在通用场景宣传为全局加速。
- 影响：
  - 用户侧：AMD gfx950（MI355X / MI350x）上跑 GLM-5.2 的用户获得明显但条件化的收益——低并发 decode 场景 TPUT 最高 +14%、TPOT 最高 −12.5%；并发升高或

prefill 主导时收益衰减至近零。

- 系统侧：权重加载阶段多一次一次性 per-tensor 量化（可忽略）；*decode* 每层少一次 kernel launch 与张量拷贝，降低 CPU 侧调度与内存带宽压力。
- 团队侧：这是 AMD GLM5 系列优化（#31323、#31324）的延续点，后续可复用同样模式到 MTP / NextN 层；同时为“硬件门控改动必须手工补测”提供了反面教材。
- 影响程度：中低广度（2 文件、*gated* 路径），但在目标路径上局部深度明显。
- 风险标记：无自动化测试覆盖，权重 *dtype* 隐式契约，数值精度回归风险，仅 *gfx950* + GLM-5.2 生效，低并发场景收益为主

关联脉络

- PR #31323 [AMD] [GLM5] Fuse shared-expert append into aiter grouped-topk (skip per-layer append kernel): 同一 GLM5 + AMD 优化线，均在 *gfx950* 上把更多计算融合进 *aiter* 内核；该 PR 后被 #35105 回滚，说明此系列融合优化需要充分的正确性证据。
- PR #31324 [AMD] [GLM5] Skip DSA *decode* indexer when kv_len <= index_topk (dense k-only fast path): 同系列 GLM5 *decode* 路径的 *gfx950* 优化，可对比两条优化支线的取舍与验证方式。
- PR #35105 Revert "[AMD] [GLM5] Fuse shared-expert append into aiter grouped-topk (skip per-layer append kernel)": 回滚 #31323，体现 AMD GLM5 优化中的反复与谨慎验证文化，是本 PR 所代表优化路线的风险警示。
- PR #35111 [AMD] diffusion: normalize ModelOpt-FP8 weights to *e4m3fnuz* on *gfx942*: 同为 fp8 格式契约（*e4m3fn* / *e4m3fnuz*）修正案例，提示本 PR 的 *dtype gate* 依赖同样需要审计。
- PR #31531 讨论中提及的 HIP MLA *absorb* 路径迁移 PR（标题未提供）: PR 讨论中明确提及：合并 main 时此 PR 把 HIP MLA *absorb* 路径迁入 forward_mla_rocm.py，直接影响本 PR 改动的最终落点。