PR #31450 完整报告

sgl-project/sglang

[AMD] Fix DeepSeek-V4 FP4 MoE expert memory bloat

合并时间：2026-08-02 14:42

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/31450

执行摘要

- 一句话：FP4 MoE 对齐降到 128，权重减约 40 GiB
- 推荐动作：值得精读：对于在 AMD 平台上部署 DeepSeek-V4 的团队，值得理解这个 padding 对齐与 kernel tile 大小的关系；对于一般读者，这是一个 " 一行常量修改 + 外部依赖联动 " 的典型案例，可学习如何在量化路径中通过 kernel 能力约束来压缩权重内存。关注点：aiter 版本升级的落地状态；若后续要推广到其他模型，注意不同 MoE kernel 的 tile_k 支持差异；建议补充参数化单测。

功能与动机

PR body 明确指出：AMD M355 上每层 MoE 权重占 2.105 GB，而同层在 NVIDIA B200 上仅 1.582 GB，差距约 25%。根因是 AMD 路径将 w13 专家权重按 256 的倍数填充，DeepSeek-V4-Pro 在 TP8 下每分区中间维度 3072/8=384 被抬到 512，使 routed-expert 权重膨胀约 1/3。作者目标是消除冗余填充，使权重占用接近 NVIDIA 水平，并为 AMD 部署节省约 40 GiB 显存。

实现拆解

变更入口：唯一改动文件是 python/sglang/srt/layers/quantization/fp8.py，函数为 process_weights_after_loading_block_quant()，属于 FP4 专家权重加载后的后处理路径，由 `_use_aiter and self.is_fp4_expert` 守卫，仅 DeepSeek-V4（经 model_config.py 的 is_deepseek_v4 + SGLANG_DSV4_FP4_EXPERTS）会进入。

1. 对齐常量调整：将 fp4_k_align 从 256 改为 128，并更新注释：DeepSeek-V4 MoE 由 FlyDSL kernel 实现，支持 tile_k=128，因此中间维度只需填充到 128 的倍数；shuffle_scale 对非 256 形状的支持自 aiter PR#4130 起可用。此改动同步影响 padded_inter 的计算结果。
2. 填充逻辑复用：后续逻辑不变——仍按 padded_inter 与 inter_per_part 的差值记录 layer.intermediate_pad、layer.hidden_pad，并在需要时把 w13_weight 从 (E, 2inter, K_packed) 零填充到 (E, 2padded, K_packed)，w2_weight 同理。TP8 下 384 已为 128 的倍数，整条 padding 分支不再触发。
3. 配套依赖：本 PR 单独合并不完整，必须同步升级 aiter 到包含 PR#4130 的版本，否则非 256 形状下 shuffle_scale 会出错；评论中 bingxche 还发现 AITER scout 存在疑似回归 commit，需先排查。

4. 验证配套：未新增单测；作者提供 GSM8k 精度（tp8 0.940 / tp8+dp8 0.942）与 8k1k 压测（权重加载后显存 159.07 → 112.36 GB，ITL/TTFT/TTT 基本持平）作为验证；aiter 侧额外为 dim=384 补充了 dsv4_fp8fp4_tuned_fmoe.csv 调优。

关键文件：

- python/sglang/srt/layers/quantization/fp8.py（模块 量化层；类别 source；类型 core-logic；符号 process_weights_after_loading_block_quant）：唯一改动文件：在 process_weights_after_loading_block_quant() 中将 fp4_k_align 从 256 改为 128，直接决定 DeepSeek-V4 FP4 expert 权重是否被填充到 512，是内存收益和性能权衡的核心。

关键符号：process_weights_after_loading_block_quant

关键源码片段

python/sglang/srt/layers/quantization/fp8.py

唯一改动文件：在 process_weights_after_loading_block_quant() 中将 fp4_k_align 从 256 改为 128，直接决定 DeepSeek-V4 FP4 expert 权重是否被填充到 512，是内存收益和性能权衡的核心。

```
# python/sglang/srt/layers/quantization/fp8.py
def process_weights_after_loading_block_quant(self, layer: Module) -> None:
    # AMD FP4 experts: 走 aiter 原生 MXFP4 MoE 路径
    if _use_aiter and self.is_fp4_expert:
        # DeepSeek-V4 MoE 使用 FlyDSL kernel，支持 tile_k=128，
        # 因此中间维度只需对齐到 128 的倍数。此前按 256 对齐，
        # 导致 TP8 下 384 -> 512 的 25% 权重膨胀（PR#31450 修复）。
        # 注意：shuffle_scale 支持非 256 形状需要 aiter PR#4130。
        fp4_k_align = 128

        E, w13_N, w13_K_packed = layer.w13_weight.shape
        _, w2_N, w2_K_packed = layer.w2_weight.shape
        inter_per_part = w13_N // 2 # 每个 partition 的中间维度
        padded_inter = (
            (inter_per_part + fp4_k_align - 1) // fp4_k_align * fp4_k_align
        )

        # 记录 padding 量，aiter fused_moe 依赖 intermediate_pad 得知真实中间维度
        layer.intermediate_pad = padded_inter - inter_per_part
        layer.hidden_pad = 0

        if padded_inter != inter_per_part:
            pad_amount = padded_inter - inter_per_part
            fp4_block_k = 32

            # 将 w13_weight 从 (E, 2*inter, K_packed) 扩为 (E, 2*padded, K_packed)
            old_w13 = layer.w13_weight.data
            new_w13 = torch.zeros(
                E,
```

```
        2 * padded_inter,
        w13_K_packed,
        dtype=old_w13.dtype,
        device=old_w13.device,
    )
    # 前半段（gate/up 输出）拷贝到开头
    new_w13[:, :inter_per_part, :] = old_w13[:, :inter_per_part, :]
    # 后半段（down 输出）偏移到 padded 位置，保持与 kernel 期望布局一致
    new_w13[:, padded_inter : padded_inter + inter_per_part, :] = old_w13[
        :, inter_per_part:, :
    ]
    layer.w13_weight = torch.nn.Parameter(new_w13, requires_grad=False)

    # w2_weight 也按 padded_inter 做 packed 维度填充，逻辑与 w13 对称
```

注意：改动前后唯一实质差异是 fp4_k_align 的取值与注释；填充、intermediate_pad 记录等逻辑完全复用，因此对 384 这类 128 倍数维度会直接跳过 padding 分支。

评论区精华

评论区围绕两个问题展开：

- aiter 依赖：HaiShaw 要求 @bingxche 确认 aiter 升级，并请求 aiter scout 落地 PR#4130 后的 commit；bingxche 反馈 " 发现一个疑似回归 commit 影响 DSv32 和 DSv4-pro，需先解决再升级 "。HaiShaw 最后追问 "aiter scout green?"，该闭环未在评论中明确回复。
- 性能权衡：1am9trash 实测：send_one 无差异；小并发（< cc16）无差异；cc16 约 1% 下降。kernel 剖析显示主要差异在 moe1：w/ padding（inter=512）的 mfma_moe1_...t32x128x256 平均 35.15 us，w/o padding（inter=384）的 mfma_moe1..._t32x64x256 平均 39.31 us；moe2 基本持平。作者随后在 aiter 侧为 dim=384 补充调优配置。
- 审核结论：kkHuang-amd 与 HaiShaw 均批准（LGTM），无 inline review 评论；gemini-code-assist bot 无反馈。
  - aiter 版本升级与回归排查 (other): PR 合并前需完成 aiter 升级；AITER scout 回归需先排查，最终状态未在评论中闭环。
  - 去 padding 后的性能回归评估 (performance): 微小性能回退可接受，且 aiter 侧针对 dim=384 补充了调优配置。

风险与影响

- 风险：外部依赖风险：本 PR 的 shuffle_scale 非 256 形状依赖 aiter PR#4130；若部署环境 aiter 版本未跟上，FP4 MoE 计算可能出错。且 AITER scout 中已知有疑似回归 commit，升级本身存在不确定性。性能风险：cc16 大并发下约 1% 吞吐下降，来自 moe1 kernel 未对齐时的耗时增加；低并发无差异。回归风险：改动仅触及 is_fp4_expert 路径（DeepSeek-V4 专属），其他模型不受影响；但非 256 对齐形状在其他 aiter MoE kernel（非 FlyDSL）上的支持未在此 PR 中验证。测试覆盖缺失：未附带单元测试，依赖手工

GSM8k 与压测；建议以后补充参数化单测，断言 384 这类输入不再触发 padding。

- 影响：对用户 / 系统：AMD（M355）上部署 DeepSeek-V4-Pro（FP4 expert）的用户显存占用显著下降，权重加载后显存从 159.07 GB 降至 112.36 GB，可释放更多空间给 KV cache 等；TP8 下模型权重减少约 40 GiB。对团队：需要与 aiter 团队的版本联动（cherry-pick PR#4130），对 AMD 平台 CI 和依赖管理有影响。影响程度：仅 AMD + DeepSeek-V4 + FP4 expert 路径，其他模型 / 平台无影响；改动面极小（1 文件 9 行），但内存收益大。
- 风险标记：依赖 aiter PR#4130 升级，cc16 下约 1% 性能下降，无单测配套，AITER scout 回归待排查

关联脉络

- PR #33090 [AMD][Fix] Restore aiter-padded MoE weight dims for serialized checkpoints: 同为 AMD aiter MoE 权重维度对齐问题，一个修序列化 checkpoint 的维度恢复，一个修 padding 粒度。
- PR #32315 [AMD] Speed up DSV4 MoE weight loading from mmap views: 同为 AMD 平台 DeepSeek-V4 MoE 权重加载优化，改善加载内存与速度。
- PR #31221 [AMD] Derive AITER verify tokens-per-req from input shape: 同为 AMD AITER 后端修复，共享 aiter 依赖演进背景。