PR #51473 完整报告

vllm-project/vllm

[ROCm][DSV4] Preserve native MXFP4 TP8 shard allocation

合并时间：2026-08-11 14:57

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/51473

执行摘要

- 一句话：DSV4 路由专家保留原生 I384 分片，每卡省 31 GiB 显存
- 推荐动作：值得精读，尤其对 ROCm/AITER 量化和 MoE 部署团队。亮点是设计的收敛过程：从「AMD 专属模型子类 + monkeypatch 测试」逐步迭代为「oracle 中按 backend + activation 键控的 6 行分支 + 干净单测」，体现了「对齐是后端 kernel 的属性，不是模型的属性」这一原则；review 对测试 scope 的收敛（skip 标记、去掉 monkeypatch、用 Renormalize 路由证明解耦）也是可复用的范例。PR body 的配对内存 / 性能验证方法（含 checkpoint header 推导、非回归门声明）值得作为量化类变更的验收模板。

功能与动机

PR body 明确说明这是「physical padding rather than failed checkpoint sharding」，它「inflated the routed-expert allocation by 33%」。DeepSeek-V4 的 routed-expert intermediate 为 3072 宽，TP8 下每 rank 原生 384 宽 shard；通用 ROCm 回退按 256 对齐把它 round 到 512。review 中 AndreasKaratzas 也直接质疑 padding 的必要性："do you know why we need this padding? ... it's costing GPU mem with no obvious reason"。修复的直接收益是每 rank 节省 31.25 GiB 模型显存、KV token 容量提升 82.76%（2,859,796 -> 5,226,632），Agentic 长上下文场景吞吐提升 34%。

实现拆解

变更入口是 vllm/model_executor/layers/fused_moe/oracle/mxfp4.py 的 mxfp4_round_up_hidden_size_and_intermediate_size，这是所有 MXFP4 MoE backend 统一使用的对齐决策点。实现分四步：

1. 在 oracle 中新增 AITER BF16 + SiLU 对齐分支：紧邻 Kimi-K3 SITU 特例插入 elif backend == AITER_MXFP4_BF16 and activation == SILU 分支，对 intermediate_size 与 hidden_size 按 128 对齐。对 DSV4 TP8 的 384 原生 shard 是 no-op，从而绕开文件尾部 current_platform.is_rocm() 的 256 对齐路径（384 -> 512）。改动仅 6 行新增。
2. 规则泛化，键控从模型变为后端属性：最终提交 7b2701a6 把分配规则从「DeepSeek-V4 路由方法」泛化为「AITER BF16 + SiLU 后端属性」，因为 AITER 的 A16W4 W4A16 布局本身即 128 对齐契约。这样避免了新增 AMD 专属 RoutedExperts 子类、量化方法或模型接线，规则自动覆盖所有走 AITER BF16/SiLU 的模型；中间提交曾携带 num_fused_shared_experts 配置（为未合并的 FHMoE 预留），在 a51757b4 中被刻意移除以限定当前范围。

3. 测试配套：tests/kernels/moe/test_ocp_mx_moe.py 新增 test_aiter_mxfp4_bf16_silu_preserves_native_tp_shard，构造 TP8 + 384 experts + 3072 intermediate + Renormalize 路由的配置，直接断言 Mxfp4MoEMethod.maybe_roundup_sizes 返回 (7168, 384)。测试带 ROCm 与 AITER 可用性的 skip 标记，并特意使用 Renormalize 路由证明规则与 DeepSeek-V4 路由解耦。
4. 兼容性与验证说明：EP checkpoint 加载与 expert mapping 路径未动；但旧的 vLLM sharded_state 快照中按 256 对齐保存的 I512 张量需在物理形状变化后重新生成。PR body 给出三类验证：805.33 GiB checkpoint 的内存验证、8K/1K 配对性能测试（通过非回归门）、Agentic C64 prefix-cache 测试；另有 test_ep_weight_filter.py 的 EP8 专家过滤测试保持通过。

关键文件：

- vllm/model_executor/layers/fused_moe/oracle/mxfp4.py（模块 量化层；类别 source；类型 data-contract；符号 mxfp4_round_up_hidden_size_and_intermediate_size）：核心变更文件：在通用 MXFP4 对齐决策函数 mxfp4_round_up_hidden_size_and_intermediate_size 中新增 AITER BF16 + SiLU 的 128 对齐分支，使 DSV4 TP8 原生 384 分片不再被 ROCm 通用 256 路径填充为 512。规则按后端 + activation 键控，不依赖模型，是全部内存收益的来源。
- tests/kernels/moe/test_ocp_mx_moe.py（模块 MoE 测试；类别 test；类型 test-coverage；符号 test_aiter_mxfp4_bf16_silu_preserves_native_tp_shard）：新增聚焦回归测试 test_aiter_mxfp4_bf16_silu_preserves_native_tp_shard，直接验证 TP8 下 roundup 结果为 (7168, 384)，是防止对齐规则回退的最廉价 guard；测试特意用 Renormalize 路由证明规则与 DeepSeek-V4 路由解耦，并按 review 要求加了 ROCm 与 AITER 的 skip 标记。

关键符号：mxfp4_round_up_hidden_size_and_intermediate_size, Mxfp4MoEMethod.maybe_roundup_sizes, test_aiter_mxfp4_bf16_silu_preserves_native_tp_shard

关键源码片段

vllm/model_executor/layers/fused_moe/oracle/mxfp4.py

核心变更文件：在通用 MXFP4 对齐决策函数 mxfp4_round_up_hidden_size_and_intermediate_size 中新增 AITER BF16 + SiLU 的 128 对齐分支，使 DSV4 TP8 原生 384 分片不再被 ROCm 通用 256 路径填充为 512。规则按后端 + activation 键控，不依赖模型，是全部内存收益的来源。

```
def mxfp4_round_up_hidden_size_and_intermediate_size(
    backend: Mxfp4MoeBackend,
    hidden_size: int,
    intermediate_size: int,
    activation: MoEActivation | None = None,
) -> tuple[int, int]:
    """按后端 kernel 的对齐要求向上取整两个维度。

    这里是所有 MXFP4 MoE backend 的物理分配形状决策点：
```

```
    `Mxfp4MoEMethod.maybe_roundup_sizes` 会把 activation 类型传进来，
    让对齐规则同时受后端与 activation 键控，而不是按模型名特判。
    """
    if backend == Mxfp4MoeBackend.AITER_MXFP4_BF16 and activation == MoEActivation.SITU:
        # 已有的 Kimi-K3 特例：AITER A16W4 SiTU kernel 支持原生
        # intermediate 尺寸（K3 的 moe_intermediate 3072，TP8 下每分区
        # 384）。按 128 对齐是 no-op，避免落入 ROCm 通用 256 对齐
        # （384 -> 512）导致权重膨胀甚至 OOM。
        intermediate_size = round_up(intermediate_size, 128)
        hidden_size = round_up(hidden_size, 128)
    elif (
        backend == Mxfp4MoeBackend.AITER_MXFP4_BF16 and activation == MoEActivation.
        SILU
    ):
        # 本 PR 新增：AITER 的 A16W4 SiLU kernel 同样使用 128 对齐布局，
        # 与具体模型和路由方法无关。DSV4 的 3072 宽路由专家在 TP8 下
        # 原生分片为 384，此前被通用 ROCm 路径 pad 成 512，分配膨胀
        # 33%；此分支让 shard 保持原生宽度，每 rank 节省 31.25 GiB 显存。
        intermediate_size = round_up(intermediate_size, 128)
        hidden_size = round_up(hidden_size, 128)
    elif backend == Mxfp4MoeBackend.EMULATION:
        # 仿真后端没有 kernel tile，只需 OCP MX 块对齐（32），
        # 避免非对齐 TP/DP shard 截断 per-block scale buffer。
        intermediate_size = round_up(intermediate_size, OCP_MX_BLOCK_SIZE)
        hidden_size = round_up(hidden_size, OCP_MX_BLOCK_SIZE)
    # 其余 backend（DeepGEMM、Marlin、TRTLLM、FlashInfer、CPU 等）以及
    # ROCm 通用 256 对齐路径保持原样，未在本 PR 中变更。
    elif current_platform.is_rocm():
        intermediate_size = round_up(intermediate_size, 256)
        hidden_size = round_up(hidden_size, 256)
    # 后续分支（DeepGEMM、Marlin、TRTLLM、FlashInfer、CPU）维持原有对齐策略
```

tests/kernels/moe/test_ocp_mx_moe.py

新增聚焦回归测试 test_aiter_mxfp4_bf16_silu_preserves_native_tp_shard，直接验证 TP8 下 roundup 结果为 (7168, 384)，是防止对齐规则回退的最廉价 guard；测试特意用 Renormalize 路由证明规则与 DeepSeek-V4 路由解耦，并按 review 要求加了 ROCm 与 AITER 的 skip 标记。

```
@pytest.mark.skipif(not ROCM_AVAILABLE, reason="ROCm-specific test")
@pytest.mark.skipif(
    not is_aiter_found_and_supported(), reason="AITER is not installed or supported"
)
def test_aiter_mxfp4_bf16_silu_preserves_native_tp_shard():
    """AITER BF16 + SiLU 的 TP 分片保持原生宽度，不做 256 对齐填充。

    回归点：DSV4 TP8 下 3072 // 8 = 384，若走 ROCm 通用 256 对齐会
    变成 512，routed-expert 分配膨胀 33%（实测每 rank 多占 31.25 GiB
    显存）。本测试直接断言 roundup 后的 shape，是最便宜的回归 guard。
    """
```

```
from vllm.model_executor.layers.fused_moe import FusedMoEConfig
from vllm.model_executor.layers.fused_moe.activation import MoEActivation
from vllm.model_executor.layers.fused_moe.config import (
    FusedMoEParallelConfig,
    RoutingMethodType,
)
from vllm.model_executor.layers.fused_moe.oracle.mxfp4 import (
    Mxfp4MoeBackend,
)
from vllm.model_executor.layers.quantization.mxfp4 import Mxfp4MoEMethod

# TP8 下 3072 宽中间层 -> 384 宽原生分片
moe_parallel_config = replace(
    FusedMoEParallelConfig.make_no_parallel(),
    tp_size=8,
)
moe_config = FusedMoEConfig(
    num_experts=384,
    experts_per_token=6,
    hidden_dim=7168,
    intermediate_size=3072,
    num_local_experts=384,
    num_logical_experts=384,
    activation=MoEActivation.SILU,
    device="cpu",
    # 刻意使用 Renormalize 路由，验证规则与 DeepSeek-V4 路由解耦
    routing_method=RoutingMethodType.Renormalize,
    moe_parallel_config=moe_parallel_config,
    in_dtype=torch.bfloat16,
)
method = object.__new__(Mxfp4MoEMethod)
method.moe = moe_config
method.mxfp4_backend = Mxfp4MoeBackend.AITER_MXFP4_BF16

rounded_shape = method.maybe_roundup_sizes(
    hidden_size=7168,
    intermediate_size_per_partition=moe_config.intermediate_size_per_partition,
    act_dtype=torch.bfloat16,
    moe_parallel_config=moe_config.moe_parallel_config,
)

# I384 是原生 shard；若回退成 512 即触发回归
assert rounded_shape == (7168, 384)
```

评论区精华

> Rohan138（design）：*"would it be possible to add a special-case section here similar to Kimi-K3, rather than adding a file for this? Might be cleaner"* —— 后续提交 0a8ab7fd 起把规则移入通用 oracle 的 mxfp4_round_up_hidden_size_and_intermedi

ate_size，删除 AMD 专属文件与模型接线，最终键控泛化为 AITER BF16 + SiLU 后端属性。

AndreasKaratzas（question）："do you know why we need this padding? ... it's costing GPU mem with no obvious reason" —— Fangzhou-Ai 解释：通用 ROCm 回退按 256 对齐，TP8 下 384 被 pad 成 512；而 AITER BF16/SiLU 使用 128 对齐的 A16W4 布局，原生 I384 合法；同时 #48728（FHMoE）未合并，故有意不带 shared-expert 特例。

AndreasKaratzas（testing）：建议测试加 skipif not current_platform.is_rocm() 与 skipif not aiter_is_found_and_supported() —— ac7995d4 已添加两个 skip 标记，并移除平台 monkeypatch 与不再需要的 fixture 参数。

AndreasKaratzas（testing）：单测只验证一个数字是否必要 —— Fangzhou-Ai 回应："The returned shape is the observable regression here: I384 versus I512 directly determines the resident routed-expert allocation and the measured 31.25 GiB/rank difference"，单测是无需加载 805 GiB 模型的最便宜 guard，且用 Renormalize 路由验证规则与 DeepSeek-V4 路由解耦。

- MXFP4 对齐规则位置：oracle 特例 vs 新增 AMD 专属文件 (design): 采纳建议：Fangzhou-Ai 在 0a8ab7fd 起把规则移入 mxfp4_round_up_hidden_size_and_intermediate_size，删除 AMD 专属文件和模型接线，最终键控泛化为 AITER BF16 + SiLU 后端属性。
- padding 到底为什么需要 (question): Fangzhou-Ai 解释：通用 ROCm MXFP4 fallback 将每 rank intermediate 对齐到 256，TP8 下 384 被 pad 成 512；AITER BF16/SiLU 使用 128 对齐 A16W4 布局，原生 I384 合法；且 #48728 未合并，故不带 shared-expert 特例。
- 测试跳过条件与 monkeypatch (testing): ac7995d4 添加两个 skipif 标记（ROCM_AVAILABLE、is_aiter_found_and_supported），删除平台 monkeypatch 和不再使用的 fixture 参数。
- 单测是否有必要 (testing): 保留单测，测试定位明确：验证 allocation 决策函数输出，而非数值精度。

风险与影响

- 风险：
  1. 快照兼容性破坏：物理 shard 形状从 I512 变回 I384，已有 vLLM sharded_state 快照（以旧 256 对齐保存的 AITER BF16/SiLU 张量）必须重新生成，否则加载会因 shape 不匹配失败。这是数据契约变更，需要发布说明。
  2. FHMoE 未合并的规则缺口：中间提交曾为 num_fused_shared_experts=1 -> 512 预留分支，最终被移除；若未来合并 #48728（shared-expert fusion），需要重新引入该对齐规则，否则 fused shared experts 会越界。PR 明确声明这不是本 PR 范围。
  3. 依赖 AITER 布局契约：规则成立的前提是 AITER A16W4 的 128 对齐契约；AITER 版本更新若改变 tile 布局，该分支需要重新评估。

4. 自动化覆盖有限：仓库自动化只新增 1 个单元测试，无端到端回归；大规模验证（805 GiB checkpoint、Agentic 长上下文）均在 PR body 中手工完成，CI 无法自动复现。
5. 其余 backend（DeepGEMM、Marlin、TRTLLM、FlashInfer、EMULATION、CPU）与 EP 路径不受影响，回归面窄。 - 影响：对用户：ROCm gfx950 + AITER（VLLM_ROCM_USE_AITER=1）部署 DSV4 TP8 的用户可获每 rank 31.25 GiB 模型显存回收、KV tokens/rank 容量 +82.76%、8K/1K 并发下总吞吐 +2.64%（C64 下 +5.56%）、Agentic C64 prefix-cache 场景吞吐 +34.21%、成功请求 +47.45%、P99 TPOT -73.55%。对系统：模型内存从 156.45 GiB/rank 降至 125.20 GiB/rank（-20%），vLLM 仍按 gpu_memory_utilization 预留总内存，回收内存转为 KV 容量而非减少最终显存分配。对团队：对齐规则从模型特判收敛为「后端 + activation」键控的 oracle 分支，未来新增 AITER 支持模型时无需再改模型代码；但需要向使用 sharded_state 快照的用户同步重新生成要求。 - 风险标记：快照兼容性破坏，依赖未合并 FHMoE 工作，ROCm/AITER 专属路径，自动化测试覆盖有限

关联脉络

- PR #50597 adds the adjacent AITER/SiTU Kimi-K3 128-alignment rule: 同一 mxfp4_round_up_hidden_size_and_intermediate_size 函数的相邻特例；本 PR 是将其 128 对齐策略从 SiTU 扩展到 SiLU，PR body 明确引用为同源规则。
- PR #48728 Unmerged FHMoE shared-expert fusion work: PR body 与 review 反复提及：未合并的 FHMoE 可能改变 shared-expert 融合后的对齐需求，本 PR 有意不携带该特例，未来合并需协调对齐规则。
- PR #50276 [Bugfix] Fix packed KV block zeroing stride: Agentic 长上下文配对测试的两个分支都携带此已合并修复作为基准，PR body 说明其用于公平对比，与本 PR 的 KV 容量收益共同影响最终指标。
- PR #50928 Kimi-K3 model-level shard sizing change: 重复工作检查中确认该 PR 是 Kimi-K3 模型级分片尺寸，而非 DSV4 backend 对齐，属于同领域相邻但不同的分配策略调整。