

PR #56568 完整报告

vllm-project/vllm

[Perf][DSV4.1] Pad shared experts for native MegaMoE fusion

合并时间: 2026-09-16 17:08

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56568>

执行摘要

- 一句话: 为 DSV4.1 共享专家补零对齐, 启用 MegaMoE 融合
- 推荐动作: 值得精读, 属于「小改动、大收益」的典型性能 PR, 建议重点看三个设计点:
 1. `pad_weights` 这个复合守卫——它把「能否安全补齐」的判断集中到一个布尔量上, 同时决定了后续 `scale/` 权重的补齐方式和 `Parameter` 存储策略, 是整段逻辑的枢纽。
 2. 补齐与存储的耦合取舍: `if not pad_weights: gate_up.weight.data = transformed_l1[0]`。为什么在补齐场景必须放弃 0.7 GiB/rank 的存储回收、代价是什么, 注释写得很清楚, 是理解 vLLM 权重加载后处理 (post-load hook 期望 checkpoint 形状) 的好样本。
 3. 用 `uint8` 视角对 FP8 张量做 `F.pad`, 以及 `unflatten(0, (2, unpadded_size))` 保证 `SwiGLU` 分割点不变——细节容易踩坑。

对部署方的建议: 升级前先核对显存预算, +1.87 GiB/rank 会直接减少 KV cache 空间; 对维护者的建议: 后续可考虑在补齐完成后释放原始共享专家权重, 或增加与真实 `DeepGEMM` kernel 的数值比对测试, 弥补 `FakeDeepGemm` 覆盖不到的布局语义。

功能与动机

动机来自 DSV4.1 kernel 集成 tracker (#56217) 与 PR body 的明确表述: 「DeepSeek-V4.1-Flash's 2304-wide shared expert cannot fuse with MegaMoE's routed experts padded to 2560」。路由专家侧为了 16 字节 TMA 对齐已被补齐到 512 的倍数 (2560), 共享专家仍是 checkpoint 的 2304, 两者无法共用同一 MegaMoE kernel, 40 层共享专家只能串行执行。作者在 Issue 评论中进一步把范围收窄到 V4.1 特有問題: 「This PR is for DSV4.1 specific shared expert dim 2304. Padding to 2560 can enable the native shared expert fusion.」body 还说明这是对 #53040 的扩展, 并排查了相邻 PR: 「Checks of #45861 and related open PRs found no fix for this padding gap; #53567 and #54049 cover different paths.」

实现拆解

1. 记录未补齐的中间维 (`vllm/models/deepseek_v4/nvidia/model.py` → `DeepseekV4MegaMoEExperts.__init__`): 新增 `self.unpadded_intermediate_size = intermediate_size`。因为路由专家的补齐逻辑会把 `self.intermediate_size` 从 checkpoint 的 2304 提升到 512 对齐的 2560 (单测中 `assert experts.intermediate_size == 2560` 印证了这一点), 只有单独保存原始值, 后续才能算出共享专家要补多少。注意这个字段之

后在所有形状推导里代表「补齐后的宽度」。

2. 补齐判定与形状守卫 (`_finalize_shared_expert_weights`) : 用 `unpadded_size = unpadded_intermediate_size * num_shared_experts` 与 `padding = intermediate_size * num_shared_experts - unpadded_size` 计算缺口, 并用 `pad_weights` 一次性收口所有前置条件——`padding > 0`、`gate/up` 与 `down` 都是 `float8_e4m3fn`、权重形状恰好等于 checkpoint 原始形状 (`(2*unpadded_size, hidden)` 与 `(hidden, unpadded_size)`)、两个 `scale` 的 `dtype` 都属于 (`torch.float8_e8m0fnu`, `torch.uint8`)。这样只有「确认拿到未补齐的 checkpoint 布局」时才补齐, 其余情况 (`shape` 已变、`scale` 已被 post-load hook 重排成 packed 布局) 继续走原有的形状校验失败 → `num_shared_experts = 0` → 串行共享专家回退。
3. `scale` 与权重按同一粒度补齐: `_prepare_shared_expert_scale` 新增关键字参数 `padding: tuple[int,int](0,0)`, 在展开成 `x32scale (repeat_interleave后裁剪为:mn;k//32]` 之后, `gate/up` 走 `pad_m`——先 `unflatten(0, (2, mn // 2))` 再补, 保证 SwiGLU 的切分点仍在中间, 并把 `mn += 2 * pad_m`; `down` 走 `pad_k`, 按 `pad_k // 32` 补列并更新 `k`; 两者填充值都是 1.0 (UE8M0 的单位 `scale`)。权重的补齐在 `_finalize_shared_expert_weights` 内完成: 先 `.view(torch.uint8)` 绕开 `F.pad` 对 FP8 `dtype` 的限制, `gate/up` 同样按 `(2, unpadded_size)` 拆分后补 `padding` 行, `down` 补 `padding` 列, 再 `view` 回 `float8_e4m3fn`。零权重 × 单位 `scale` 保证补齐通道对输出无贡献。
4. 存储与别名策略调整 (关键取舍): 原实现会把 `gate_up.weight.data` 重新挂到 L1 重排后的存储上, 从而释放约 0.7 GiB/rank 的原始 FP8 权重; 补齐场景下这样做会破坏通用 linear post-load hook 对 checkpoint 形状的预期, 因此改为 `if not pad_weights: gate_up.weight.data = transformed_l1[0]`, 并且 `_transformed_shared_l1_weights = transformed_l1` (而非 `(gate_up.weight.data, transformed_l1[1])`)。这条改动同时带来两个后果: 融合权重与通用加载后处理彻底解耦 (`padding` 后的融合结果不再被原地重排波及), 以及原始权重与补齐后权重同时在显存中驻留, 模型加载内存从 41.47 GiB/rank 涨到 43.34 GiB/rank。
5. 测试配套 (`tests/models/test_deepseek_v4_mega_moe.py`) :
`test_deepseek_v4_mega_moe_finalizes_native_shared_expert_weights` 的参数化从 3 组扩到 6 组, 新增 `padded-block128`、`deepseek-v41-flash-mxfp8`、`padded-packed-scale-fallback`, 并把原本断言「不融合」的 `deepseek-v41-flash` 翻转为断言融合。`FakeDeepGemm` 新增 `scale_values` 捕获以校验 `1x32` 填充值;
`fp8_parameter` 改为随机化 `uint8` 数据再 `view` 成 FP8, 避免用全零数据「蒙对」字节相等断言; `mxfp8` 打包不再硬编码 `128/4` 而改用 `block_size`。断言覆盖: `padding` 区域权重全 0、`scale` 全 1, 补齐场景下 `Parameter` 指针 / 形状保持 checkpoint 原样, 对齐场景下指针迁移到 L1 存储, 以及重复调用 `finalize_weights` 时 `_transformed_shared_l1_weights` 对象身份不变 (幂等)。

关键文件:

- `vllm/models/deepseek_v4/nvidia/model.py` (模块 专家融合层; 类别 `source`; 类型 `data-contract`; 符号 `DeepseekV4MegaMoEExperts`, `unpadded_intermediate_size`, `_finalize_shared_expert_weights`, `_prepare_shared_expert_scale`) : 唯一源码改动点: `DeepseekV4MegaMoEExperts` 新增 `unpadded_intermediate_size` 状态, `_finalize_shared_expert_weights` 增加 `pad_weights` 守卫、权重补零与存储策略分支,

_prepare_shared_expert_scale 增加 padding 参数与单位 scale 填充。它决定 DSV4.1-Flash 的 40 层共享专家能否与路由专家一起走原生 MegaMoE 融合。

- tests/models/test_deepseek_v4_mega_moe.py (模块 模型测试; 类别 test; 类型 test-coverage; 符号 test_deepseek_v4_mega_moe_finalizes_native_shared_expert_weights, FakeDeepGemm, FakeDeepGemm.scale_values, fp8_parameter) : 覆盖本次新增的补齐路径与全部回退分支: 参数化从 3 组扩到 6 组 (新增 padded-block128、deepseek-v41-flash-mx8、padded-packed-scale-fallback, 并把 deepseek-v41-flash 从「不融合」翻转为「融合」), 新增 scale 填充值捕获、随机化 FP8 权重、padding 区域字节断言、Parameter 指针断言与 finalize 幂等断言。

关键符号: DeepseekV4MegaMoEExperts.init, DeepseekV4MegaMoEExperts._finalize_shared_expert_weights, DeepseekV4MegaMoEExperts._prepare_shared_expert_scale, DeepseekV4MegaMoEExperts.finalize_weights, test_deepseek_v4_mega_moe_finalizes_native_shared_expert_weights

关键源码片段

tests/models/test_deepseek_v4_mega_moe.py

覆盖本次新增的补齐路径与全部回退分支: 参数化从 3 组扩到 6 组 (新增 padded-block128、deepseek-v41-flash-mx8、padded-packed-scale-fallback, 并把 deepseek-v41-flash 从「不融合」翻转为「融合」), 新增 scale 填充值捕获、随机化 FP8 权重、padding 区域字节断言、Parameter 指针断言与 finalize 幂等断言。

tests/models/test_deepseek_v4_mega_moe.py: 新增的参数化与核心断言 (精简整理)

```
@pytest.mark.parametrize(
    "hidden_size,intermediate_size,block_size,fused,mx8",
    [
        # 路由与共享宽度本就对齐, 仍走「迁移 Parameter 存储」的老路径
        pytest.param(128, 512, 128, True, False, id="aligned-block128"),
        pytest.param(128, 512, 128, True, True, id="aligned-block128-mx8"),
        # 真实 DSV4.1-Flash 形状: 2304 的共享专家补齐到 2560 后应可融合
        pytest.param(5120, 2304, 32, True, False, id="deepseek-v41-flash"),
        pytest.param(5120, 2304, 32, True, True, id="deepseek-v41-flash-mx8"),
        pytest.param(128, 2304, 128, True, False, id="padded-block128"),
        # 已被 post-load hook 重排成 packed scale 的布局必须回退到串行共享专家
        pytest.param(128, 2304, 128, False, False, id="padded-packed-scale-fallback"),
    ],
)

def test_deepseek_v4_mega_moe_finalizes_native_shared_expert_weights(
    monkeypatch, hidden_size, intermediate_size, block_size, fused, mx8
):
    """Shared fusion preserves checkpoint channels and zeros padded channels."""

    class FakeDeepGemm:
        transformed_dims: list[tuple[int, int]] = []
        scale_inputs: list[tuple[int, ...]] = []
        scale_values: list[torch.Tensor] = [] # 新增: 捕获 1x32 scale 内容以校验填充值
```

```

@classmethod
def transform_sf_into_required_layout(cls, sf, mn, k, *args, **kwargs):
    cls.scale_inputs.append(tuple(sf.shape))
    cls.scale_values.append(sf)
    return torch.empty((sf.shape[0], mn, k // 128), dtype=torch.int32)
# ... 其余 get_symm_buffer_for_mega_moe / transform_weights_for_mega_moe ...

def fp8_parameter(*shape):
    # 用随机 uint8 再 view 成 FP8, 避免全零数据让字节相等断言「蒙对」
    return torch.nn.Parameter(
        torch.empty(*shape, dtype=torch.uint8).random_(1, 119).view(torch.float8_e4m3fn),
        requires_grad=False,
    )

# ... 构造 shared_experts、记录 originals (权重字节 + 展开后的 1x32 scale) ...
experts.finalize_weights(shared_experts)

assert experts.has_fused_shared_experts is fused
if not fused:
    # 回退分支: 不补齐、不搬迁 Parameter, 共享专家保持 checkpoint 形状
    assert experts.num_shared_experts == 0
    assert shared_experts.gate_up_proj.weight.data_ptr() == original_gate_up_ptr
    return

padded_size = experts.intermediate_size
assert FakeDeepGemm.scale_inputs[-2:] == [
    (1, 2 * padded_size, hidden_size // 32),
    (1, hidden_size, padded_size // 32),
]
# 补齐区域: 权重必须全 0, scale 必须全 1 (UE8M0 的单位 scale)
for actual, original, fill in (
    (experts._transformed_shared_l1_weights[0].view(torch.uint8), originals[0][0], 0),
    (FakeDeepGemm.scale_values[-2][0], originals[0][1], 1),
):
    actual = actual.unflatten(0, (2, padded_size))
    assert torch.equal(
        actual[:, :intermediate_size], original.unflatten(0, (2, intermediate_size))
    )
    assert torch.all(actual[:, intermediate_size:] == fill)

if padded_size != intermediate_size:
    # 补齐场景: 通用 linear post-load hook 仍会拿到 checkpoint 形状的权重
    assert shared_experts.gate_up_proj.weight.data_ptr() == original_gate_up_ptr
    assert shared_experts.down_proj.weight.data_ptr() == original_down_ptr

# 幂等: 重复 finalize 不应重做转换
transformed_l1 = experts._transformed_shared_l1_weights
experts.finalize_weights(shared_experts)
assert experts._transformed_shared_l1_weights is transformed_l1

```

评论区精华

本 PR 没有任何 inline review 评论 (`review_comments_count` 为 0)，讨论全部发生在 Issue 评论区与评审状态上：

1. 作者主动向评审解释范围：「This PR is for DSV4.1 specific shared expert dim 2304. Padding to 2560 can enable the native shared expert fusion.」（@WoosukKwon、@zyongye 被 cc）。
 2. zyongye 只回复 /ci run，随后直接给出 APPROVED、无修改意见，最后由 Isotr0py 合并，属于「低争议、以实测数据说话」的评审形态。
 3. CI 机器人拒绝首次构建：「This PR is 219 commits behind upstream main... No new CI build was started.」作者 merge main (commit 66d9152) 后重跑，触发 Buildkite CI #89277。
 4. claude[bot] 的自动评审因 fork PR 被禁用：「This pull request is from a fork — automated review is disabled.」因此设计正确性主要依赖作者自述的 31 个单测与 GSM8K 复现实验，缺少第三方静态审查。
 5. 最实质的「讨论」其实是作者在 body 里自曝的取舍：「Model-load memory increased from 41.47 to 43.34 GiB/rank (+1.87 GiB) because the original shared parameters remain allocated.」评审未对此提出异议，也未要求补充释放策略。
- 为什么必须把共享专家补齐到 2560 (design): 评审接受该思路，zyongye 直接触发 CI 并在后续给出 APPROVED，无修改意见。
 - 补齐后不再回收原始共享权重带来的显存上升 (performance): 作者显式披露并以吞吐收益换取显存，评审未提出异议，也未要求补充释放策略或后续优化项，属于被接受的既有取舍。
 - 分支落后 main 219 个 commit 导致 CI 被拒 (other): 作者 merge main (commit 66d9152) 后重新 /ci run，触发 Buildkite CI #89277，测试通过后才合并。
 - fork PR 的自动化 review 被跳过 (testing): 缺少第三方静态审查，正确性主要依赖作者提供的 31 个单测、8 rank 融合层数核对 (0 → 40) 与 GSM8K 复现实验。

风险与影响

- 风险：
 1. 显存回归（已量化）：补齐路径不再迁移 Parameter 存储，原始 checkpoint 权重与补齐后的 transformed 权重并存，模型加载内存 +1.87 GiB/rank (41.47 → 43.34 GiB)。在 16K 上下文、`--gpu-memory-utilization 0.9` 的部署下，这部分会直接挤压 KV cache 容量；`vllm/models/deepseek_v4/nvidia/model.py` 中 `if not pad_weights` 分支是唯一控制点，若未来想回收内存需另找方案。
- 1. 对第三方 kernel 布局契约的隐式依赖：`pad_weights` 判定与 `_prepare_shared_expert_scale` 的填充都建立在 DeepGEMM 的约定之上——`transform_sf_into_required_layout` 的分组 API（用 singleton 维换 MN-major、TMA 对齐的 packed-UE8M0 stride）、1x32 scale 粒度、`block_k % 32 == 0`。`_check_runtime_supported` 只用 `hasattr` 做能力探测，无法发现语义层面的布局漂移；而单测使用 `FakeDeepGemm` (`transform_sf_into_required_layout` 直接返回 `torch.empty`)，真实 kernel 的 stride 与数值语义并未被覆盖。

2. 数值正确性依赖 kernel 对补齐区域的处理：零权重配单位 scale 在设计上不改变输出，但补齐行 / 列会真实参与 MMA 累加，若 scale 被写成 0 或权重非零都会污染共享专家输出。GSM8K 复现 (flexible-extract 96.21% → 96.13%, strict-match 78.17% → 79.23%) 提供了端到端经验证据，但缺少与真实 DeepGEMM kernel 的逐元素数值比对。
3. 回退路径的静默性：一旦 `pad_weights` 判为假（例如权重形状已是补齐后的 2560，或 scale 已被 post-load hook 重排成 packed 布局），代码不会补齐，随后形状校验失败并把 `num_shared_experts` 置 0，只留下一条 `logger.warning`，性能静默回落到串行共享专家。`padded-packed-scale-fallback` 用例覆盖了这条分支的行为，但线上只能靠日志发现。
4. 补零视角转换的可读性风险：`gate_up_weight.view(torch.uint8).unflatten(0, (2, unpadded_size))` 依赖 FP8 为单字节 dtype 且 gate/up 在 dim 0 上按 [gate; up] 排布，一旦上游改用其他排布或引入更窄 dtype，这段 padding 会静默错位而 shape 断言未必能拦住。
- 影响：模型性能：8xB200、TP8/EP/SP、FP8 KV、16K 上下文下，`vllm bench serve` 显示输出吞吐从 122.6 → 134.3 tok/s (128/256, C1) 到 2853.9 → 3359.2 tok/s (1024/128, C64)，提升 9.6%~17.7%；Mean TPOT 下降 8.8%~14.3%，Mean TTFT 下降 4.1%~21.0%。

精度：GSM8K 1319 题上 flexible-extract 96.21% → 96.13% (-0.076 pp)，strict-match 78.17% → 79.23% (+1.061 pp)，无退化信号。

影响范围：仅限走 `deep_gemm_mega_moe` 后端且共享专家需要补齐的 DSV4/DSV4.1 NVIDIA 路径；其他模型、其他 MoE 后端不受影响。作者提供了

`VLLM_DISABLE_DSV4_MEGAMOE_SHARED_EXPERT_FUSION` 作为运行时开关，可与修复前后逐层对照 (0 → 40 层融合)。

资源成本：每 rank 加载内存 +1.87 GiB，属于常驻成本，会传导到可服务的并发数与 KV cache 容量，部署方需要重新核对 `gpu-memory-utilization` 与 `max-model-len` 的余量。

团队与工程实践：为「对齐 padding + 保留 checkpoint 形状 Parameter」建立了一个可复用的模式，后续遇到共享专家与路由专家宽度不一致的模型（不限于 2304）可以照此扩展；测试参数化从 3 组扩到 6 组、并显式覆盖回退分支，也提高了该路径的回归安全性。

- 风险标记：显存占用 +1.87 GiB/rank，依赖 DeepGEMM 1x32 布局契约，真机 kernel 数值验证薄弱，核心 MoE 融合路径变更，补齐失败静默回退串行

关联脉络

- PR #53040 #53040（上下文未提供标题）：共享专家原生融合的前序 PR: PR body 首段写明「Extends #53040」：本 PR 在该 PR 建立的原生共享专家融合与路由权重 512 对齐补齐基础上，补上 2304 → 2560 的宽度缺口。
- PR #56217 [Feature]: DeepSeek-V4.1-Flash Kernels Integration & Optimization Tracker: 本 PR 的动机来源，body 首行即引用该 tracker issue；它汇总了 Mega-Gate/Mega-mHC、Sparse Indexer、DeepSelect、FlashMLA 等一系列 DSV4.1 kernel 集成工作。
- PR #53567 [8/N][warmup][DSv4] Migrate MoE execution and distributed kernels: PR body 明确排查过：「#53567 and #54049 cover different paths」——它迁移的是 DSV4 NVIDIA MoE 执行与分布式 kernel 的 warmup 契约，不解决共享专家宽度缺口。

- PR #54049 [feat] FlashInfer CuteDSL MegaMoE integration: 同属 MegaMoE 融合方向但后端不同：它引入 FlashInfer 的 CuteDSL MegaMoE kernel（大 batch/prefill 场景），本 PR 修的是 DeepGEMM MegaMoE 后端的共享专家 padding 缺口。
- PR #56441 [Perf][DSpark] Add KV-only context insertion across V4.1 cache formats: 同属 DSV4.1 kernel 集成与性能优化线，均以 DSV4.1 特有的 cache/ 权重布局差异为切入点做 kernel 级加速，可一并理解该模型族在 vLLM 中的演进节奏。