

# PR #54787 完整报告

vllm-project/vllm

[ROCm][Perf][M3] Fused allreduce+GemmaRMSNorm fast path

合并时间: 2026-09-08 15:21

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/54787>

## 执行摘要

- 一句话: ROCm 接入 AITER 融合 allreduce+RMSNorm 快路径
- 推荐动作: 值得精读, 尤其是对 ROCm 后端或通信 / 融合算子感兴趣的读者。看点有三: 一是“性能快路径必须携带反面门控”的工程范式——只在一阶段窗口启用, 宁可回退也不用更慢的 two-stage 融合, 这一结论直接来自 reviewer 的 A/B 数据而非直觉; 二是谓词收敛策略——把散落在三个 impl 里的判定上提到 AiterCustomAllreduce.use\_1stage\_fused\_ar\_rms, 既去重又统一了 eager/ 编译路径的语义; 三是评论区展示的定位方法 (按 M 维度扫 rank-max kernel 时间、push vs pull 通信方向对比、stage-2 norm kernel 常数开销拆分)。建议后续跟进两件事: 补上 supports\_dynamic\_hidden\_dim 守卫 (旧 aiter 的静默 no-op 风险), 以及为 use\_1stage\_fused\_ar\_rms 的边界条件补单元测试。

## 功能与动机

PR body 明确指出问题动机: “fused\_allreduce\_gemma\_rms\_norm only has a flashinfer fast path; on ROCm it always falls back to all\_reduce + GemmaRMSNorm — two kernel launches per layer”。首个 commit 进一步解释了为什么 helper 是唯一的落点: eager 模型 (如 MiniMax-M3) 不会走 torch.compile 的 RocmAiterAllReduceFusionPass, 只能在 eager 路径上手工做融合。目标是复用 aiter 已有的 rocm\_aiter\_fused\_allreduce\_rmsnorm custom op, 镜像 flashinfer 分支的形态, 让 post-attention 的 AR + norm 从每层约 12.5us 降到约 7.5us, 同时保持 fallback 路径数值不变。

## 实现拆解

1. 抽取共享的 one-stage 谓词 (vllm/distributed/device\_communicators/aiter\_custom\_all\_reduce.py)。新增 AiterCustomAllreduce.use\_1stage\_fused\_ar\_rms(inp), 把 AITER launcher (csrc/include/custom\_all\_reduce.cuh) 的判据镜像成一处实现: token 数不超过 80、dtype 为 bf16/fp16、每行 16 字节 pack 且不超过 1024 个 pack、world\_size==2 直接放行、非 fully\_connected 拒绝、world\_size<=4 要求 total\_bytes < 256 KiB、world\_size<=8 要求 total\_bytes < 128 KiB。谓词被声明为 capture-static (只依赖 shape/dtype/TP size/ 拓扑), 因此可以安全用于 CUDA graph 捕获期判定。
2. 收敛重复判定 (vllm/aiter\_ops.py)。
  - \_rocm\_aiter\_fused\_allreduce\_rmsnorm\_impl、
  - \_rocm\_aiter\_fused\_allreduce\_rmsnorm\_quant\_per\_group\_impl 和
  - \_rocm\_aiter\_fused\_allreduce\_rmsnorm\_quant\_per\_group\_with\_bf16\_norm\_impl 原本各自内联了一整段 hidden/token/ 字节判定 (本文件净删 73 行), 现在统一改为

use\_1stage = aiter\_ar.use\_1stage\_fused\_ar\_rms(input)。顺带修掉一处不一致：  
bf16-norm 变体用 input\_.shape[0] 当 token 数，共享谓词改用 numel // hidden\_dim，与  
AITER launcher 及其它调用方一致（对 2-D 输入二者等价）。同时给  
\_rocm\_aiter\_fused\_allreduce\_rmsnorm\_impl 与其 fake 增加 gemma\_norm: bool =  
False 参数并透传给 ca.custom\_fused\_ar\_rms。

3. 新增 eager helper 门控与分支 (vllm/model\_executor/layers/fused\_allreduce\_gemma\_rms\_norm.py)。新增 \_can\_use\_aiter\_fused\_ar\_rms(hidden\_states): 先做 current\_platform.is\_rocm() 早退，再检查 rocm\_aiter\_ops.is\_custom\_all\_reduce\_enabled()、2-D/ 连续 /bf16lfp16、aiter\_ar 非 None 且未 disabled、total\_bytes 不超过 effective\_max\_size()、should\_custom\_ar() 通过，最后要求 use\_1stage\_fused\_ar\_rms() 为真。命中后在 fused\_allreduce\_gemma\_rms\_norm 中于 flashinfer 分支之后插入 rocm\_aiter\_ops.get\_fused\_allreduce\_rmsnorm\_op()(...) 调用，透传 gemma\_norm=True；不命中仍走原有 tensor\_model\_parallel\_all\_reduce + norm 兜底。
4. 按 review 收紧门控范围。第二个 commit (d6b6a251) 把“只在 one-stage 窗口启用”从讨论结论落成代码：A/B 显示 two-stage 变体（跨设备 reduce-scatter + 本地 norm）比显式 all\_reduce + norm 慢 1-3us，所以 helper 宁可回退也不启用融合。这一改动同时反向定义了谓词的语义——它不只是“能否调用”，而是“调用是否划算”。
5. 测试与验证配套。本次没有新增或修改任何测试文件，验证依赖 PR body 中的手工证据：TP4 tensor parity (token 数 1 到 32768, decode 规模 bit-exact, 16/128/1024 时 relerr <= 2.7e-3, 8192+ 因超过字节上限正确回退)、gfx950 上既有 kernel 测试 (tests/kernels/test\_minimax\_m3\_amd\_ops.py、tests/kernels/attention/test\_minimax\_m3.py、tests/kernels/core/test\_minimax\_reduce\_rms.py)、gsm8k 精度对照与 benchmark\_serving A/B。这意味着新增谓词的边界行为 (M=21/22、world\_size=2/4/8、非连续输入等) 没有单元测试兜底。

关键文件：

- vllm/model\_executor/layers/fused\_allreduce\_gemma\_rms\_norm.py (模块 融合算子；类别 source；类型 core-logic；符号 \_can\_use\_aiter\_fused\_ar\_rms, fused\_allreduce\_gemma\_rms\_norm)：eager 模型的融合入口：新增 \_can\_use\_aiter\_fused\_ar\_rms 门控与 AITER 调用分支（透传 gemma\_norm=True），是本次性能收益的直接来源，也是 reviewer 要求补平台守卫的位置。
- vllm/distributed/device\_communicators/aiter\_custom\_all\_reduce.py (模块 集合通信；类别 source；类型 core-logic；符号 use\_1stage\_fused\_ar\_rms)：新增共享谓词 AiterCustomAllreduce.use\_1stage\_fused\_ar\_rms，把 AITER launcher 的 one-stage 契约 (pack / token / 字节上限) 收敛为单一实现，供 eager helper 与 \_aiter\_ops 的三个融合 op 复用，是本次去重与门控的共同基石。
- vllm/\_aiter\_ops.py (模块 算子接入；类别 source；类型 core-logic；符号 \_rocm\_aiter\_fused\_allreduce\_rmsnorm\_impl, \_rocm\_aiter\_fused\_allreduce\_rmsnorm\_fake, \_rocm\_aiter\_fused\_allreduce\_rmsnorm\_quant\_per\_group\_impl, \_rocm\_aiter\_fused\_allreduce\_rmsnorm\_quant\_per\_group\_with\_bf16\_norm\_impl)：三

处内联的 one-stage 判定被替换为共享谓词（净删 73 行），并给 `_rocm_aiter_fused_allreduce_rmsnorm_impl` 及其 fake 增加 `gemma_norm` 参数以支撑 Gemma 语义；同时修正 bf16-norm 变体用 `shape[0]` 计 token 的写法。

关键符号： `_can_use_aiter_fused_ar_rms`, `fused_allreduce_gemma_rms_norm`, `use_1stage_fused_ar_rms`, `_rocm_aiter_fused_allreduce_rmsnorm_impl`, `_rocm_aiter_fused_allreduce_rmsnorm_quant_per_group_impl`, `_rocm_aiter_fused_allreduce_rmsnorm_quant_per_group_with_bf16_norm_impl`

## 关键源码片段

### `vllm/model_executor/layers/fused_allreduce_gemma_rms_norm.py`

eager 模型的融合入口：新增 `_can_use_aiter_fused_ar_rms` 门控与 AITER 调用分支（透传 `gemma_norm=True`），是本次性能收益的直接来源，也是 reviewer 要求补平台守卫的位置。

```
# vllm/model_executor/layers/fused_allreduce_gemma_rms_norm.py
```

```
def _can_use_aiter_fused_ar_rms(hidden_states: torch.Tensor) -> bool:
```

```
    # 先做平台判定：rocm_aiter_ops 在非 ROCm 平台可能为 None,
```

```
    # 所以不能先 import 再判断（review 中 shen-shanshan 指出的问题）
```

```
    if not current_platform.is_rocm():
```

```
        return False
```

```
    from vllm._aiter_ops import rocm_aiter_ops
```

```
    if not rocm_aiter_ops.is_custom_all_reduce_enabled():
```

```
        return False
```

```
    # 与 flashinfer 分支共用同一套 2-D / 连续 / bf16lfp16 约束
```

```
    if (
```

```
        hidden_states.dim() != 2
```

```
        or not hidden_states.is_contiguous()
```

```
        or hidden_states.dtype not in _FI_SUPPORTED_DTYPES
```

```
    ):
```

```
        return False
```

```
    aiter_ar = rocm_aiter_ops.get_aiter_allreduce()
```

```
    if aiter_ar is None or aiter_ar.disabled:
```

```
        return False
```

```
    # 复用 AITER custom all-reduce 自身的字节预算
```

```
    total_bytes = hidden_states.numel() * hidden_states.element_size()
```

```
    if total_bytes > aiter_ar.effective_max_size():
```

```
        return False
```

```
    if not aiter_ar.should_custom_ar(hidden_states):
```

```
        return False
```

```
    # 最关键的一条：只有 aiter 会走 one-stage 内核时才启用融合。
```

```
    # two-stage（跨设备 reduce-scatter + 本地 norm）比显式
```

```
    # all_reduce + norm 更慢，此时宁可回退
```

```
    return aiter_ar.use_1stage_fused_ar_rms(hidden_states)
```

```
def fused_allreduce_gemma_rms_norm(
```

```

hidden_states: torch.Tensor,
residual: torch.Tensor,
norm: GemmaRMSNorm,
) -> tuple[torch.Tensor, torch.Tensor]:
    tp_size = get_tensor_model_parallel_world_size()
    if tp_size == 1:
        # 无需 all-reduce, 等价于未融合路径
        return norm(hidden_states, residual)

    ok, max_token_num = _can_use_flashinfer(hidden_states, tp_size)
    if ok:
        # flashinfer (NVIDIA) 快路径, 本 PR 未改动, 此处略去实现细节
        ...
        return norm_out, hidden_states

# 本次新增: ROCm/AITER 分支, 一次 launch 完成
# all-reduce + 残差加 + GemmaRMSNorm
if _can_use_aiter_fused_ar_rms(hidden_states):
    from vllm._aiter_ops import rocm_aiter_ops

    return rocm_aiter_ops.get_fused_allreduce_rmsnorm_op()(
        input_=hidden_states,
        residual=residual,
        weight=norm.weight,
        epsilon=norm.variance_epsilon,
        gemma_norm=True, # Gemma 风格: weight_bias = 1.0
    )

# 兜底: 显式 all-reduce + GemmaRMSNorm, 与未融合模型数值一致
reduced = tensor_model_parallel_all_reduce(hidden_states)
return norm(reduced, residual)

```

## vllm/distributed/device\_communicators/aiter\_custom\_all\_reduce.py

新增共享谓词 AiterCustomAllreduce.use\_1stage\_fused\_ar\_rms, 把 AITER launcher 的 one-stage 契约 (pack / token / 字节上限) 收敛为单一实现, 供 eager helper 与 \_aiter\_ops 的三个融合 op 复用, 是本次去重与门控的共同基石。

```
# vllm/distributed/device_communicators/aiter_custom_all_reduce.py
```

```

def use_1stage_fused_ar_rms(self, inp: torch.Tensor) -> bool:
    """AITER 的融合 allreduce+RMSNorm 是否会走 one-stage 内核。

    判据镜像 aiter 的 fused_allreduce_rmsnorm launcher:
    每行 16 字节 pack、每行最多 1024 个 pack、最多 80 个 token,
    以及当前 TP size/拓扑下 one-stage custom all-reduce 的字节上限。
    超出窗口时融合 op 会走 two-stage (跨设备 reduce-scatter + 本地 norm),
    它比显式 all_reduce + norm 更慢, 因此能回退的调用方必须要求本判据。
    capture-static: 只依赖 shape、dtype、TP size 与拓扑。
    """

```

```

hidden_dim = inp.shape[-1]
# token 上限先行: prefill 规模输入一次比较就退出
if inp.numel() // hidden_dim > 80:
    return False
if inp.dtype not in (torch.bfloat16, torch.float16):
    return False
# aiter 内核按 16 字节 pack 处理每一行
pack_size = 16 // inp.element_size()
if hidden_dim % pack_size != 0 or hidden_dim // pack_size > 1024:
    return False
ca = self._impl
world_size = ca.world_size
if world_size == 2:
    return True
# 非全互联拓扑不具备 one-stage 的通信前提
if not ca.fully_connected:
    return False
total_bytes = inp.numel() * inp.element_size()
if world_size <= 4:
    return total_bytes < 256 * 1024
if world_size <= 8:
    return total_bytes < 128 * 1024
return False

```

### vllm/\_aiter\_ops.py

三处内联的 one-stage 判定被替换为共享谓词（净删 73 行），并给 \_rocm\_aiter\_fused\_allreduce\_rmsnorm\_impl 及其 fake 增加 gemma\_norm 参数以支撑 Gemma 语义；同时修正 bf16-norm 变体用 shape[0] 计 token 的写法。

# vllm/\_aiter\_ops.py

```

def _rocm_aiter_fused_allreduce_rmsnorm_impl(
    input_: torch.Tensor,
    residual: torch.Tensor,
    weight: torch.Tensor,
    epsilon: float,
    gemma_norm: bool = False, # 新增: 走 GemmaRMSNorm 语义 (weight_bias = 1.0)
) -> tuple[torch.Tensor, torch.Tensor]:
    aiter_ar = rocm_aiter_ops.get_aiter_allreduce()
    assert aiter_ar is not None, "aiter allreduce must be initialized"
    ca = aiter_ar.aiter_ca
    # 原本内联的 hidden_dim / token_num / total_bytes 三段判定已删除,
    # 统一改为调用上提到 AiterCustomAllreduce 的共享谓词,
    # 保证与 per-group quant 变体和 eager helper 的判定完全一致
    use_1stage = aiter_ar.use_1stage_fused_ar_rms(input_)

    result = ca.custom_fused_ar_rms(
        input_,
        residual,

```

```

    weight,
    epsilon,
    use_1stage=use_1stage,
    gemma_norm=gemma_norm,
)
assert result is not None
return result[0], result[1]

def _rocm_aiter_fused_allreduce_rmsnorm_fake(
    input_: torch.Tensor,
    residual: torch.Tensor,
    weight: torch.Tensor,
    epsilon: float,
    gemma_norm: bool = False,
) -> tuple[torch.Tensor, torch.Tensor]:
    # fake 实现只需保持签名与真实实现一致，便于 custom op 注册与 tracing
    return torch.empty_like(input_), torch.empty_like(residual)

```

## 评论区精华

1) 平台守卫: shen-shanshan 在 `vllm/model_executor/layers/fused_allreduce_gemma_rms_norm.py` 上指出 “We should check `current_platform` is rocm to make this helper method rocm-only, otherwise the imported `rocm_aiter_ops` could be None on other hardwares”, 作者在 f338185 提交中补上 `current_platform.is_rocm()` 早退。2) 去重: shen-shanshan 对 `aiter_ops.py` 连留两条 “These logics are totally the same as `use_1stage = aiter_ar.use_1stage_fused_ar_rms(input)`, please also replace” 与 “Ditto.”, 促成三处内联判定收敛到共享谓词。3) 性能 crossover 之争: Fangzhou-Ai 给出 TP4 rank-max A/B (M=1 时融合快 14.66%, M=32 起反而慢 4.61%, M=256 慢 8.15%), 并提醒 “There is a reproducible crossover exactly at AITER's one-stage/two-stage boundary”; benenzhu 用 H=6144 复现后承认 “beyond the one-stage boundary the fused path is slower ... we should gate it under the size of 1stage”, 并补做了 `conc=32/64`。4) 根因追踪: benenzhu 在 `benenzhu/aiter#1` 中对比 two-stage 实现——融合内核写各 rank 的 tmp slice, 而 plain custom\_all\_reduce 保持 slice 本地并从其它 rank 读, 改成 pull 后 M=32 从 8.15us 回到 8.07us, M=256 的差距从 3.8us 缩到 1.2us; 剩余约 1.1us 恒定差来自 aiter 的 `local_device_load_rmsnorm` 比 triton kernel 慢。5) 未闭环项: CodeRabbit 标记 major——`_can_use_aiter_fused_ar_rms` 未在 `supports_dynamic_hidden_dim` 为 False 时拒绝 H 不在 {512,1024,2048,4096} 的输入, 旧版 aiter 会静默 no-op; head 版本代码中未见对应守卫, 讨论中也没有解决记录。

- helper 必须先做 ROCm 平台守卫 (correctness): 作者在 f338185 提交中补上 `current_platform.is_rocm()` 早退, 并把它放在 `import rocm_aiter_ops` 之前。
- `_aiter_ops` 三处重复的 1-stage 判定需要去重 (design): 三处内联判定全部替换为共享谓词, 净删 73 行; 同时把 bf16-norm 变体用 `shape[0]` 计 token 的差异统一为 `numel // hidden_dim`。

- 融合路径在一阶段 / 二阶段边界处存在性能 crossover (performance): helper 增加 `use_1stage_fused_ar_rms` 门控: 只有在 AITER 会走 one-stage 内核的 shape 上才启用融合, 超出窗口一律回退显式 `all_reduce` + GemmaRMSNorm。
- two-stage 融合内核为何慢于 plain custom\_all\_reduce (performance): 根因部分定位并反馈到 aiter 上游, vLLM 侧先用门控规避, 不阻塞本 PR 合入。
- 旧版 aiter 的 hidden\_dim 兼容守卫缺失 (correctness): head 版本的 `_can_use_aiter_fused_ar_rms` 中仍未看到 `supports_dynamic_hidden_dim` 检查, 讨论里也没有对应的修复记录, 属于遗留风险。

## 风险与影响

- 风险: 1) 旧版 aiter 兼容性 (最需要关注): `vllm/distributed/device_communicators/aiter_custom_all_reduce.py` 的 `supports_dynamic_hidden_dim` 文档已说明 aiter v0.1.12 之前 launcher 对 H 不在 {512,1024,2048,4096} 的尺寸会“silently no-op”; 新增的 `_can_use_aiter_fused_ar_rms` 没有引用该属性, CodeRabbit 的 major 评论在 head 代码中看不到对应修复。若在旧 aiter 上跑到非常规 hidden\_dim, 融合 op 可能返回未定义 / 未归约的结果而不是回退。2) 门控与 aiter 内部契约耦合: `use_1stage_fused_ar_rms` 是对 aiter launcher 判据的“镜像”而非调用其 API, 80 token / 1024 pack / 256 KiB / 128 KiB 这些常量一旦在 aiter 侧调整, vLLM 侧会静默失配——乐观方向是性能退化 (回退或误用 two-stage), 悲观方向是调用到内核拒绝的尺寸。3) 测试覆盖缺口: 本 PR 只有源码改动, 没有 tests/ 下的新增用例, 谓词边界 (M=21/22、world\_size=4/8 的字节阈值、非连续或非 2-D 输入) 完全依赖手工 A/B 与既有 kernel 测试。4) 数值与精度: bf16 reassociation 使 16/128/1024 token 下 relerr 到  $2.7e-3$ ; gsm8k 从 0.9704 到 0.9644 (stderr 约 0.005), 落在统计噪声边缘, 但属于质量指标方向性变化, 值得在后续版本继续跟踪。5) 执行模式分叉: 编译期的 `RocmAiterAllReduceFusionPass` 与本次 eager helper 各自维护一份启用条件, 若二者对同一 shape 给出不同判定, 同一模型在 eager 与 compile 下会出现不同的算子组合 (性能差异与潜在的数值差异)。
- 影响: 用户侧: 仅影响 ROCm/AMD 平台且启用 AITER custom all-reduce 的部署 (示例为 MI355X x4 TP4 + MiniMax-M3-MXFP4 + EAGLE3)。decode 阶段 TPOT 3.32ms→3.26ms (conc=1)、12.68ms→12.31ms (conc=16), 每层 AR+norm 从约 12.5us 降到约 7.5us; prefill 规模输入因字节上限正确回退, 不受影响。系统侧: NVIDIA/flashinfer 路径、TP==1 路径行为完全不变, 回退路径数值与未融合模型一致, 属于受控的局部优化。团队侧: `_aiter_ops.py` 三处重复判定收敛到 `AiterCustomAllreduce` 上的单一谓词, 后续 per-group quant (含 emit\_bf16 变体) 与 eager helper 共享同一份契约, 维护成本下降, 但也让该谓词成为多个调用方的公共依赖, 改动风险集中度提高。
- 风险标记: 旧版 aiter 兼容守卫缺失, 缺少新增单元测试, 门控阈值与 aiter 内部契约镜像耦合, eager 与 compile 路径判定可能分叉, 精度指标方向性波动

## 关联脉络

- PR #47270 标题未在材料中提供 (讨论中引用的 TP4 rank-max kernel A/B 验证): Fangzhou-Ai 明确说明其 A/B 数据是在验证 #47270 时跑出来的, 本 PR 的门控结论直接

建立在该 PR 的测量之上，两者共用同一套 TP4 rank-max kernel 对比方法。

- PR #56349 [ROCm] Auto-enable breakable CUDA graphs for DeepseekV41ForCausalLM: 同为 ROCm 平台按硬件 / 后端条件自动启用或关闭某条快路的门控型改动；本 PR 的测试配置显式设置 `VLLM_USE_BREAKABLE_CUDAGRAPH=0`，说明两条路径共享同一套 ROCm 图捕获约束，需一起理解平台默认行为。