PR #51253 完整报告

vllm-project/vllm

[ROCm][Perf] Kimi-K3 Shard Latent MoE up-projection for ROCm path

合并时间：2026-08-07 14:11

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/51253

执行摘要

- 一句话：ROCm Kimi-K3 尾投影按 TP 分片，吞吐提升约 3~5%
- 推荐动作：值得精读，尤其是 _shard_up_proj_tail 的「先 all-reduce 还原 latent、再按 rank 只算自己的 up_proj 切片、通过 addmm_ 的 beta-add epilogue 并入 shared 部分、最后由最终 all-reduce 缝合」模式，是 CUDA Tier 2 路径在 ROCm 的镜像移植。测试设计也很有参考价值：用 narrow 权重 + capture 中间张量来固定「每 rank 只写自己 shard」的算术契约，而不是只跑 e2e。若要复用该模式，注意把分片前置条件（TP 整除、scaling factor、sequence parallel）纳入配置契约。

功能与动机

PR body 指出：Kimi-K3 默认的 routed up-projection 是 ReplicatedLinear，每个 TP rank 都持有相同权重并计算完整投影，实际只有 1/8 的工作量被用到；CUDA 路径已有 LatentMoERunner 做 column-parallel 分片，但 ROCm 路径没有把 runner_cls 传给 FusedMoEFactory，因此从未启动分片。本 PR 的目标是把 CUDA 的 Tier 2（up-projection sharding）镜像到 ROCm，让每个 rank 只投影自己的 hidden 切片并在最终 all-reduce 缝合，从而在 MI355X 上获得 3~5% 的吞吐提升。

实现拆解

1. 接线入口：vllm/models/kimi_k3/amd/linear.py 的 KimiMLP.__init__ 中，FusedMoEFactory 构造新增 runner_cls=ROCmLatentMoERunner if self.use_latent_moe else None。这是唯一的行为开关：只有 latent MoE 才使用新 runner，其他配置保持原路径。
2. Runner 类：新增 vllm/models/kimi_k3/amd/latent_moe_runner.py，定义 ROCmLatentMoERunner(MoERunner)。__init__ 通过条件链判定 _tail_shardable：存在 up_proj、tp_size > 1、up_proj 权重行数可被 tp_size 整除、存在 shared experts、非 sequence parallel、routed_scaling_factor == 1.0。任一不满足则 _up_proj_shard_size = 0 并仅打一次 warning。
3. forward 分发：forward 只在 _tail_shardable and not self._fused_output_is_reduced 时走 _fused_forward，否则调用 super().forward()，确保不满足分片条件时行为与 PR 之前完全等价。
4. 核心分片逻辑：_fused_forward 先按需执行 apply_routed_input_transform、_maybe_pad_hidden_states、_forward_entry，再 _unpack 出 fused（routed）与

shared 输出；_shard_up_proj_tail 对 fused 部分做 tensor_model_parallel_all_reduce还原完整 latent，经 norm 后用 narrow 取出本 rank 的 up_proj 权重行和 shared 输出的对应列，addmm_(latent, up_proj_shard.t()) 把投影以 beta-add epilogue 并进该 shard，最后 _maybe_reduce_final_output 做最终 all-reduce 缝合。

5. 测试配套：新增 tests/models/kimi_k3/test_amd_latent_moe_runner.py（仅 ROCm 平台执行），用 spawn 起真实 TP4/TP8 进程组，断言分片结果与 replicated 全量投影一致、在最终 collective 前每 rank 只改写自己的 shard 区间；用参数化单测覆盖 forward 仅在有效条件下走分片路径，以及 6 种配置打破分片前提时的回退。同时新增 tests/models/kimi_k3/__init__.py 使测试目录成为包。

关键文件：

- vllm/models/kimi_k3/amd/latent_moe_runner.py（模块 MoE 运行器；类别 source；类型 core-logic；符号 ROCmLatentMoERunner, init, _shard_up_proj_tail, forward）：核心实现文件，新增 ROCmLatentMoERunner 类，包含 tail 可分片性判定、up-projection 分片计算与前向分发，是本次性能优化的主体。
- tests/models/kimi_k3/test_amd_latent_moe_runner.py（模块 测试套件；类别 test；类型 test-coverage；符号 _build_transform, _tail_runner, _check_matches_replicated, _check_writes_only_its_own_shard）：新增多 GPU 分布式测试，通过数学等价性与「只写自己 shard」断言固定分片实现的正确性，是保证该优化不出错的关键配套。
- vllm/models/kimi_k3/amd/linear.py（模块 模型层；类别 source；类型 configuration；符号 KimiMLP.init）：接线文件：在 FusedMoEFactory 构造处传入 runner_cls，决定 latent MoE 是否启用新 runner，是本次优化的开关点。
- tests/models/kimi_k3/__init__.py（模块 测试包；类别 test；类型 test-coverage）：新增测试包初始化文件，仅含 license 头，使 tests/models/kimi_k3 目录成为 Python 包。

关键符号：ROCmLatentMoERunner.init, ROCmLatentMoERunner._shard_up_proj_tail, ROCmLatentMoERunner.forward, ROCmLatentMoERunner._fused_forward, KimiMLP.init

关键源码片段

tests/models/kimi_k3/test_amd_latent_moe_runner.py

新增多 GPU 分布式测试，通过数学等价性与「只写自己 shard」断言固定分片实现的正确性，是保证该优化不出错的关键配套。

```
# tests/models/kimi_k3/test_amd_latent_moe_runner.py 节选:
# 用「全部 rank 的完整权重」构造期望值，验证分片路径与其数学等价。
def _check_matches_replicated(device: torch.device, tp_size: int, rank: int) -> None:
    transform = _build_transform(device)
    runner = _tail_runner(transform, tp_size)
    group = get_tp_group().device_group

    for iteration, num_tokens in enumerate((1, 5, 8, 16, 5)):
        torch.manual_seed(100 * iteration + rank + 1)
        routed_output, shared_output = _rank_partials(num_tokens, device)

        # 期望值 = 完整 up_proj 权重 × norm(all-reduce 后的 latent),
```

```
        # 再加上 all-reduce 后的 shared 部分，即 replicated 路径的语义。
        expected = F.linear(
            F.rms_norm(
                _all_reduced(routed_output, group),
                (LATENT_SIZE,),
                transform.norm.weight,
                EPS,
            ),
            transform.up_proj.weight,
        )
        expected.add_(_all_reduced(shared_output, group))

        actual = runner._shard_up_proj_tail(routed_output, shared_output, None)

        torch.testing.assert_close(actual, expected, atol=8e-2, rtol=3e-2)


def _check_writes_only_its_own_shard(
    device: torch.device, tp_size: int, rank: int
) -> None:
    """两个 rank 即使交换权重行和写偏移，最终总和仍正确，
    所以必须在 final collective 之前检查每个 slice 的写入范围。"""
    transform = _build_transform(device)
    runner = _tail_runner(transform, tp_size)
    group = get_tp_group().device_group

    torch.manual_seed(rank + 1)
    routed_output, shared_output = _rank_partials(8, device)
    before = shared_output.clone()
    latent = transform.norm(_all_reduced(routed_output, group))

    captured: dict = {}
    real_reduce = runner._maybe_reduce_final_output

    # 包一层 _maybe_reduce_final_output，在真正 all-reduce 前截获中间张量。
    def _capture(states, trunc_size, output_is_reduced=None):
        captured["states"] = states.clone()
        captured["output_is_reduced"] = output_is_reduced
        return real_reduce(states, trunc_size, output_is_reduced)

    object.__setattr__(runner, "_maybe_reduce_final_output", _capture)
    runner._shard_up_proj_tail(routed_output, shared_output, None)
    # 恢复原方法，打破 runner -> _capture -> runner 引用环，
    # 确保截获的设备张量在进程组销毁前被释放。
    object.__setattr__(runner, "_maybe_reduce_final_output", real_reduce)

    assert captured["output_is_reduced"] is False

    shard = HIDDEN_SIZE // tp_size
```

```
start, end = rank * shard, (rank + 1) * shard
local = captured["states"]
projected = F.linear(latent, transform.up_proj.weight[start:end])

torch.testing.assert_close(
    local[:, start:end], before[:, start:end] + projected, atol=8e-2, rtol=3e-2
)
torch.testing.assert_close(local[:, :start], before[:, :start], atol=0, rtol=0)
torch.testing.assert_close(local[:, end:], before[:, end:], atol=0, rtol=0)
```

评论区精华

本 PR 没有产生实质性的技术讨论：claude[bot] 仅提示这是 fork PR、自动 review 被禁用；tjtanaa 直接 APPROVED 并给出 LGTM。设计方向和 fallback 条件由作者在 PR body 中预先说明，维护者未提出异议。

- review 流程：fork PR 无自动 review，仅维护者 LGTM (other): 无未解决的技术疑虑；合并由维护者直接批准。

风险与影响

- 风险：
  1. 数值等价性：分片路径将「全量 up_proj 一次 GEMM」改为「先 all-reduce routed 再逐 rank 小 GEMM、最后 all-reduce shared」，bf16 下浮点结合顺序改变可能引入微小误差，测试容差为 atol=8e-2、rtol=3e-2（较宽松），极端长序列或特殊数据下建议额外验证。
  2. 静默回退：当配置不满足 _tail_shardable 条件（如 hidden 不可整除、sequence parallel、scaling factor != 1）时，仅打一次 warning 后回退到 replicated 路径；不影响正确性，但优化会静默失效，排障时需注意日志。
  3. 平台与模型范围有限：仅 ROCm + Kimi-K3 latent MoE 生效，CUDA 已有实现，且本 PR 未实现 Tier 1（down projection 分片）。
  4. 测试覆盖依赖多卡：TP4/TP8 测试需要对应规模 ROCm GPU，CI 覆盖取决于可用机器；引擎级集成路径（如远端重叠流）未纳入测试。 - 影响：影响范围：仅当在 ROCm 上以 TP>1 服务 moonshotai/Kimi-K3（latent MoE 启用）时走新路径。PR body 的 MI355X 基准显示 total tokens/s 提升 +3.05%~+4.91%，mean TTFT 改善约 -5.2%~-5.5%，mean TPOT/ITL 改善约 -2.8%~-4.6%；并发越高收益比例略降（128 并发时约 +3.3%）。对不满足分片条件的配置行为不变。团队侧新增一个 runner 类与一组分布式测试，为后续在 ROCm 上实现 Tier 1 分片或其他 latent MoE 模型优化打下基础。 - 风险标记：仅 ROCm 生效，条件回退依赖配置，数值等价依赖宽松容差，多卡测试覆盖有限

关联脉络

- PR #50185 attn_res kernel latency improvements: 同属 Kimi-K3 性能优化线（历史 PR 标签 kimi），本 PR 是 ROCm 侧的对应优化；PR body 也引用了 CUDA 已有的

LatentMoERunner 作为参照。