PR #51070 完整报告

vllm-project/vllm

[K3 Perf] Combine multiple all gather together for SP, 1.5~3x kernel level performance improvement

合并时间：2026-08-06 05:42

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/51070

执行摘要

- 一句话：Kimi K3 SP 多路 all-gather 合并为一次，1.5~3 倍加速
- 推荐动作：值得精读。它展示了如何把多次小集合通信合并为一次大集合通信来降低 kernel 启动与同步开销，是一个小而典型的通信优化案例。建议关注：aux_hidden_states 初始化时机的移动、packed 张量的 split 对称性，以及该路径缺少直接单测的风险。可复用 PR body 的 benchmark 脚本验证收益。

功能与动机

PR 描述原实现为逐层多次 all-gather："Second layer -> all-gather 46th layer -> all-gather 90th layer -> all-gather final output -> all-gather"，而优化后仅做一次最终 allgather。多次小集合通信的 kernel 启动与同步开销较大，合并为一次大集合通信可显著降低通信时延，PR 内 benchmark 给出了 1.5~3 倍的内核级性能提升证据。

实现拆解

1. 调整 aux 状态收集时机：在 vllm/models/kimi_k3/nvidia/model.py 的 forward 中，将 aux_hidden_states 的初始化从 sp_shard 之前移到之后，确保 SP 下所有写入列表的隐藏状态均为分片视图，避免全局 / 分片状态混用。
2. 移除循环内逐层 all-gather：删除 for 循环内对每个 aux_hidden_state 的 sp_all_gather 与 [:full_num_tokens] 截断，改为直接 append 分片状态，将通信延迟到所有状态收集完毕之后。
3. 统一打包并单次收集：SP 分支末尾，若存在 aux 状态，将最终 hidden_states 与所有 aux 状态沿 dim=-1 拼接，执行一次 sp_all_gather 并截断到 full_num_tokens，再按 hidden_size 切回；无 aux 状态时保持原有单次 gather 路径。
4. 配套与测试：仅改动该模型文件，未附带测试文件；PR body 提供可复现的 torchrun benchmark 脚本，验证 TP4、BF16 下 8~1024 token 场景获得 1.59~2.82 倍内核级加速，并通过 torch.testing.assert_close 验证新旧路径输出等价。

关键文件：

- vllm/models/kimi_k3/nvidia/model.py（模块 模型实现；类别 source；类型 core-logic；符号 forward）：Kimi K3 模型 forward 主路径，SP 下 all-gather 合并优化的唯一改动文件，控制流与张量打包逻辑均在此。

关键符号：forward

关键源码片段

vllm/models/kimi_k3/nvidia/model.py

Kimi K3 模型 forward 主路径，SP 下 all-gather 合并优化的唯一改动文件，控制流与张量打包逻辑均在此。

```
# 关键片段：Kimi K3 forward 中 SP 场景的 all-gather 合并优化
full_num_tokens = positions.shape[0]
if self.use_sequence_parallel:
    if envs.VLLM_MOE_SKIP_PADDING and is_forward_context_available():
        forward_context = get_forward_context()
        forward_context.is_padding = sp_padding_mask(
            forward_context.is_padding, hidden_states
        )
    hidden_states = sp_shard(hidden_states)
    assert residual is None, "Currently, SP is not supported with PP"

# 在 SP 分片之后初始化 aux_hidden_states，保证所有辅助状态都是分片视图，
# 后续才能统一打包做一次 all-gather（原实现里首层状态可能是全局视图）。
aux_hidden_states: list[torch.Tensor] = []
if self.start_layer in self.aux_hidden_state_layers:
    if self.use_attn_res or residual is None:
        aux_hidden_states.append(hidden_states)
    else:
        aux_hidden_states.append(hidden_states + residual)

for layer_idx, layer in enumerate(
    self.layers[self.start_layer: self.end_layer],
    start=self.start_layer,
):
    hidden_states, prefix_sum, residual = layer(
        positions=positions,
        hidden_states=hidden_states,
        prefix_sum=prefix_sum,
        residual=residual,
    )
    if (layer_idx + 1) in self.aux_hidden_state_layers:
        if self.use_attn_res:
            assert prefix_sum is not None
            aux_hidden_state = prefix_sum + hidden_states
        else:
            assert residual is not None
            aux_hidden_state = hidden_states + residual

        # 不再在每层单独 all-gather，保持分片状态，延后统一通信。
        aux_hidden_states.append(aux_hidden_state)
```

```
if self.use_sequence_parallel:
    if aux_hidden_states:
        hidden_size = hidden_states.shape[-1]
        # 将最终 hidden_states 与所有辅助状态拼接成一个连续张量，
        # 用一次 sp_all_gather 完成全部数据的序列并行收集，
        # 再按 hidden_size 切回，减少 N 次小集合通信为 1 次大集合通信。
        packed_hidden_states = torch.cat(
            [hidden_states, *aux_hidden_states], dim=-1
        )
        packed_hidden_states = sp_all_gather(packed_hidden_states)
        packed_hidden_states = packed_hidden_states[:full_num_tokens]
        hidden_states, *aux_hidden_states = packed_hidden_states.split(
            hidden_size, dim=-1
        )
    else:
        hidden_states = sp_all_gather(hidden_states)
        hidden_states = hidden_states[:full_num_tokens]
```

评论区精华

该 PR 没有实质性技术 review 讨论。sfeng33 直接批准（无评论）；claude[bot] 仅提示仓库配置了手动 review 入口；作者 yewentao256 通过 /ci run 触发 Buildkite CI（build #82545）。正确性与性能结论主要依赖 PR body 的 benchmark 脚本，缺少评审者对实现细节（如 packed tensor 的显存峰值、非 SP 路径回归）的质询。

- CI 触发与运行 (other): CI 已触发运行，PR 最终被合并，说明 CI 检查通过。
- Claude 自动化 review 提示 (other): 无实质技术讨论产生。
- 审批 (other): PR 被批准并合入 main。

风险与影响

- 风险：
  - 正确性风险：SP 下 aux_hidden_states 首个元素的语义从“全局 hidden_states”变为“分片 hidden_states”，且所有状态统一打包后再切分，张量划分顺序依赖 torch.cat/split 的严格对称；若 hidden_size 推导或层序变化可能错位。当前没有直接的单测覆盖该路径。
  - 性能与显存：packed_hidden_states 一次性构建 4 倍宽度的临时张量，单次 all-gather 的消息体和显存峰值都增加；在超大 token 数或 TP 较大时可能引发 OOM，需要与通信次数减少的收益权衡。
  - 兼容性：仅 SP 路径受影响；PP 与 SP 组合仍被断言禁止。非 SP 路径初始化位置变化不影响行为，但仍是公共 forward 主干，回归影响面需用现有 K3 集成测试确认。
- 影响：
  - 用户侧：Kimi K3 + sequence parallel 推理的端到端延迟下降，输出语义不变（已由 PR 内 assert_close 验证等价性）。
  - 系统侧：每轮前向的集合通信次数显著下降，对大规模 TP/ 多节点部署的通信瓶颈有直接改善。

- 团队侧：同一文件近期已有多次 K3 相关修复（#51131、#50649），该 PR 是性能优化线的一部分；缺少自动化正确性测试是后续跟进点。
- 风险标记：核心推理路径变更，缺少直接测试覆盖，集合通信合并可能增大峰值显存

关联脉络

- PR #51131 [BugFix][K3] Skip moe_intermediate padding when EP is enabled: 同一文件 vllm/models/kimi_k3/nvidia/model.py，同为 Kimi K3 模型的 SP/EP 路径修复，显示该模型并行策略仍在持续演进。
- PR #50649 [ROCm][Bugfix] Kimi-K3 Fix KDA NaN on mixed batches and racy autotune config: 同为 Kimi K3 模型相关修复，体现 K3 模型在多硬件与并行场景下持续优化。