PR #52079 完整报告

vllm-project/vllm

[Kimi-K3] Add GEMM-RS for sequence parallelism

合并时间：2026-08-14 00:02

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/52079

执行摘要

- 一句话：Kimi-K3 序列并行新增 GEMM-RS 融合内核，加速 TP reduce-scatter
- 推荐动作：值得精读，尤其是对内核与性能优化团队：展示了用 CUTLASS CuTeDSL 编写融合通信内核的完整套路，以及 opt-in 三层校验（maybe_init_gemm_rs → can_run → should_run）的降级设计，CUDA graph 兼容测试也很有参考价值。建议后续跟进修复 assert 上限检查，并补充更大 TP 规模与更多 shape 的验证；阅读时重点关注 model.py 中 reduce-scatter 条件的变化与 gemm_rs.py 的 tmem→shared→multimem 数据路径。

功能与动机

在序列并行下，每个 TP rank 只拥有 token 的局部切片，TP 分片的 down-proj（O-proj、共享专家 down-proj）需要在 GEMM 之后紧跟一次跨 rank 的 reduce-scatter 才能恢复序列分片并完成求和，大 batch prefill 时通信开销接近甚至超过 GEMM。PR body 直接引用 CUTLASS 的 Blackwell distributed GEMM-RS 示例（dcf215a），用 multimem.ld_reduce 在 GEMM 写回阶段完成跨 rank 归约，并给出数据：TP4/TP8 微基准中 M>=512 时相对 RING_LL 有 1.2x-1.7x 加速；8xGB300 E2E prefill（TP8+EP+SP）下 TPGS 提升 6.52%-8.18%。由于内核为 SM100 的 TMA + tcgen05 流水做过优化，小 M（<128）时基线更快，因此只在 M>=128 的大 batch 路径启用。

实现拆解

1. 内核实现：新增 vllm/models/kimi_k3/nvidia/ops/cute_dsl/gemm_rs.py（约 790 行），实现 Sm100GemmRsBF16 内核与 multimem_ld_reduce_16B、nanosleep 等 CuTeDSL 用户算子。内核基于 CUTLASS CuTeDSL 示例，用 TMA 加载 A/B 矩阵、tcgen05 做 BF16 GEMM、在写回阶段用 multimem.ld_reduce 直接写入 NVLink 对称内存完成 TP 归约；支持任意 M（通过 padded_M 补齐）、CTA group 1/2、BN=128/256，并对外提供 init_gemm_rs、get_gemm_rs、can_run、should_run 接口。
2. 框架工具补充：vllm/cute_utils/mbarrier.py 新增 arrive、arrive_expect_tx 两个 mbarrier DSL op，用于 TMA 屏障与事务数量声明；vllm/cute_utils/__init__.py 新增 to_cta0_smem，将共享内存指针转换到 CTA0 地址空间，供 2-CTA 集群复用屏障。
3. 模型接入与语义适配：vllm/models/kimi_k3/nvidia/model.py 新增 maybe_init_gemm_rs()，按环境变量、SP 开关、ubatching、BF16 dtype、CUDA/SM100 平台、TP 2-16 且整除 128 等条件分层判定，并初始化全局工作区；KimiMLP、mla.py 的 MultiHeadLatentAttention、kda.py 的 KimiK3DeltaAttention 都增加 run_gemm_rs 参数

与 forward 分支：先 can_run 验证投影权重兼容性，运行时用 should_run（M>=128）决定是否走 GEMM-RS。由于 GEMM-RS 返回本地序列分片而非完整 M，KimiModelForCausalLM.forward 中 reduce-scatter 的触发条件改为 hidden_states.shape[0] == M。

4. 配置与 CI：vllm/envs.py 新增 VLLM_KIMI_K3_GEMM_RS 环境变量（默认关闭）；.buildkite/test_areas/distributed.yaml 将新内核测试注册到 distributed CI。
5. 测试与基准：tests/kernels/test_kimi_k3_gemm_rs.py 为 2-GPU 分布式测试，覆盖非对齐 M=129/257/1023/1024/8191、交替 shape、CUDA graph 捕获与重放，并与 torch GEMM + NCCL RS 参考实现对拍；benchmarks/kernels/benchmark_kimi_k3_gemm_rs.py 提供支持 CUDA graph 的微基准，对比 RING_LL / LDMC / 融合 GEMM-RS 三种方案并输出 speedup；tests/models/kimi_k3/test_sequence_parallel.py 扩展了 shard_sequence_parallel_mlp gating 测试的 eligible 参数。

关键文件：

- vllm/models/kimi_k3/nvidia/ops/cute_dsl/gemm_rs.py（模块 融合内核；类别 source；类型 core-logic；符号 multimem_ld_reduce_16B, nanosleep, Sm100GemmRsBF16, init_gemm_rs）：PR 核心：新增 SM100 BF16 融合 GEMM-RS 内核（约 790 行），基于 CUTLASS CuTeDSL 与 multimem.ld_reduce 指令，把 GEMM 与 TP reduce-scatter 融合为一次内核启动，并提供 can_run/should_run 运行时判定。
- vllm/models/kimi_k3/nvidia/model.py（模块 模型接入；类别 source；类型 core-logic；符号 maybe_init_gemm_rs, shard_sequence_parallel_mlp, KimiMLP.init, KimiMLP.forward）：模型接入入口：maybe_init_gemm_rs 做分层判定与全局初始化，KimiMLP/KimiMoE 按 can_shard_sequence_parallel 与 run_gemm_rs 传递并接入 forward 分支，同时调整 SP reduce-scatter 的 shape 语义。
- tests/kernels/test_kimi_k3_gemm_rs.py（模块 内核测试；类别 test；类型 test-coverage；符号 test_kimi_k3_gemm_rs, _worker, _reference, _assert_valid_rows_close）：新增 2-GPU 分布式内核测试，覆盖非对齐 M、交替 shape 与 CUDA graph 捕获重放，是对拍 torch GEMM + NCCL RS 参考实现的关键验证。
- vllm/cute_utils/mbarrier.py（模块 屏障工具；类别 source；类型 core-logic；符号 arrive, arrive_expect_tx）：新增 mbarrier 的 arrive / arrive_expect_tx CuTeDSL 用户算子，供 GEMM-RS 内核在 TMA 流水与 2-CTA 集群下做屏障和事务数量声明。
- vllm/models/kimi_k3/nvidia/mla.py（模块 注意力模块；类别 source；类型 core-logic；符号 MultiHeadLatentAttention.init, MultiHeadLatentAttention.forward）：注意力 O-proj 接入 GEMM-RS：MultiHeadLatentAttention 增加 run_gemm_rs 初始化与 forward 分支，GEMM-RS 返回本地分片后直接返回。
- vllm/models/kimi_k3/nvidia/kda.py（模块 线性注意力；类别 source；类型 core-logic；符号 KimiK3DeltaAttention.init, KimiK3DeltaAttention.forward）：KDA 线性注意力的 o_proj 同样接入 GEMM-RS：KimiK3DeltaAttention 增加 run_gemm_rs 初始化与 forward 分支。
- vllm/cute_utils/__init__.py（模块 指针工具；类别 source；类型 core-logic；符号 to_cta0_smem）：新增 to_cta0_smem 指针转换算子，使 2-CTA 集群模式下跨 CTA 复用 TMA 屏障地址，是内核集群协作的基础设施。

- vllm/envs.py（模块 环境配置；类别 source；类型 configuration）：新增 VLLM_KIMI_K3_GEMM_RS 环境变量（默认关闭），是整个功能的 opt-in 开关。
- benchmarks/kernels/benchmark_kimi_k3_gemm_rs.py（模块 性能基准；类别 source；类型 benchmark；符号 Candidate, parse_args, capture_graph, benchmark_graphs）：新增支持 CUDA graph 的分布式微基准，对比 Torch GEMM + NCCL RS（RING_LL / LDMC）与融合 GEMM-RS，输出分项耗时与 speedup，是 PR 性能数据的来源。
- tests/models/kimi_k3/test_sequence_parallel.py（模块 序列并行；类别 test；类型 test-coverage；符号 test_shard_sequence_parallel_mlp_gating）：扩展 shard gating 测试，新增 eligible 参数覆盖 FusedMoE 路径不参与 SP 分片的契约变化。
- .buildkite/test_areas/distributed.yaml（模块 CI 配置；类别 config；类型 configuration）：把新增的 GEMM-RS 分布式测试注册到 distributed CI 区域，保证内核在 CI 中持续验证。

关键符号：maybe_init_gemm_rs, shard_sequence_parallel_mlp, Sm100GemmRsBF16.init, Sm100GemmRsBF16.call, Sm100GemmRsBF16.kernel, Sm100GemmRsBF16.prepare_tma, multimem_ld_reduce_16B, nanosleep, to_cta0_smem, arrive, arrive_expect_tx, init_gemm_rs, get_gemm_rs, can_run, should_run, KimiMLP.forward, MultiHeadLatentAttention.forward, KimiK3DeltaAttention.forward, test_kimi_k3_gemm_rs, benchmark_shape

关键源码片段

vllm/models/kimi_k3/nvidia/ops/cute_dsl/gemm_rs.py

PR 核心：新增 SM100 BF16 融合 GEMM-RS 内核（约 790 行），基于 CUTLASS CuTeDSL 与 multimem.ld_reduce 指令，把 GEMM 与 TP reduce-scatter 融合为一次内核启动，并提供 can_run/should_run 运行时判定。

```
# SPDX-License-Identifier: Apache-2.0
# vllm/models/kimi_k3/nvidia/ops/cute_dsl/gemm_rs.py
"""SM100 BF16 GEMM with a fused tensor-parallel reduce-scatter."""

# 基于 CUTLASS 的 Blackwell distributed GEMM-RS 示例（dcf215a）
# 参见 https://github.com/NVIDIA/cutlass/issues/3117 了解内存语义

import cutlass
import torch
import torch.distributed._symmetric_memory as symm_mem
from cuda.bindings.driver import CUstream
from cutlass import BFloat16, Int32, Int64, Uint16, cute, utils
from cutlass._mlir import ir
from cutlass._mlir.dialects import llvm, nvvm, vector
from cutlass.cute.nvgpu import cpasync, tcgen05
from cutlass.cutlass_dsl import dsl_user_op
from cutlass.utils import get_smem_capacity_in_bytes

from vllm.cute_utils import _tcgen05, mbarrier, simple_tma_copy, to_cta0_smem
from vllm.distributed import get_tp_group
```

```
@dsl_user_op
def multimem_ld_reduce_16B(x: cute.Tensor, *, loc=None, ip=None) -> cute.Tensor:
    # 一次加载按 .v4.bf16x2 打包的 16 字节，并对 global 地址执行带加法
    # 归约的 multimem.ld_reduce，将跨 TP rank 的数据累加到 FP32。
    # 这是在 GEMM 写回阶段直接完成跨 rank reduce-scatter 的核心指令。
    assert x.element_type == BFloat16 # 当前仅支持 BF16 输入
    vec_type = ".v4.bf16x2"

    ptr = x.iterator.toint(loc=loc, ip=ip).ir_value(loc=loc, ip=ip)
    asm = (
        "multimem.ld_reduce.relaxed.gpu.global.add.acc::f32"
        f"{vec_type} {{$0, $1, $2, $3}}, [$4];"
    )
    # 内联汇编返回 4 个 Int32，重组为向量后 reinterpret 回 BF16x4 返回
    struct = llvm.inline_asm(
        llvm.StructType.get_literal([Int32.mlir_type] * 4),
        [ptr],
        asm,
        "=r,=r,=r,=r,l",
        has_side_effects=True,
        loc=loc,
        ip=ip,
    )
    vec = vector.from_elements(
        ir.VectorType.get([4], Int32.mlir_type, loc=loc),
        [
            llvm.extractvalue(Int32.mlir_type, struct, [i], loc=loc, ip=ip)
            for i in range(4)
        ],
        loc=loc,
        ip=ip,
    )
    ssa = cute.TensorSSA(vec, 4, Int32)

    y = cute.make_rmem_tensor(4, Int32)
    y.store(ssa)
    return cute.recast_tensor(y, x.element_type)


class Sm100GemmRsBF16:
    """SM100 BF16 GEMM + 融合 TP reduce-scatter 的 CuTeDSL 内核。

    每个 CTA 的 tcgen05 计算结果经 tmem -> shared -> multimem.ld_reduce
    路径写入 NVLink 对称内存，从而把 [GEMM + reduce_scatter] 两次算子
    合成一次内核启动。
    """

    def __init__(
        self, rank: int, num_ranks: int, BN: int = 128, cta_group: int = 1
```

```
    ) -> None:
        self.rank = rank
        self.num_ranks = num_ranks
        BM, BK = 128, 64
        self.cta_tile = (BM, BN, BK)
        self.cta_group = cta_group

        # 根据共享内存容量推导均衡的 pipeline 阶段数
        smem_bytes = get_smem_capacity_in_bytes()
        self.stage_size = (BM + (BN // cta_group)) * BK * 2
        self.num_stages = smem_bytes // self.stage_size
```

vllm/models/kimi_k3/nvidia/model.py

模型接入入口：maybe_init_gemm_rs 做分层判定与全局初始化，KimiMLP/KimiMoE 按 can_shard_sequence_parallel 与 run_gemm_rs 传递并接入 forward 分支，同时调整 SP reduce-scatter 的 shape 语义。

```
# vllm/models/kimi_k3/nvidia/model.py

def maybe_init_gemm_rs(vllm_config: VllmConfig, use_sequence_parallel: bool) -> bool:
    # opt-in 开关，默认关闭；通过环境变量 VLLM_KIMI_K3_GEMM_RS 启用
    if not envs.VLLM_KIMI_K3_GEMM_RS:
        return False

    parallel_config = vllm_config.parallel_config
    tp_size = parallel_config.tensor_parallel_size
    # 分层校验：任一条件不满足就给出明确原因并回退到标准 GEMM + NCCL RS
    if not use_sequence_parallel:
        reason = "sequence parallelism is disabled"
    elif parallel_config.use_ubatching:
        reason = "ubatching is enabled"
    elif vllm_config.model_config.dtype != torch.bfloat16:
        reason = "the model dtype is not BF16"
    elif not current_platform.is_cuda():
        reason = "the device is not CUDA"
    elif not current_platform.is_device_capability_family(100):
        reason = "the device is not SM100-family"
    elif not 1 < tp_size <= 16:
        reason = "TP size is not in the supported range 2-16"
    elif 128 % tp_size != 0:
        reason = "TP size does not divide 128"
    else:
        reason = None

    if reason is not None:
        logger.warning_once("GEMM-RS was requested but is disabled because %s.", reason)
        return False

    from vllm.models.kimi_k3.nvidia.ops.cute_dsl.gemm_rs import init_gemm_rs
```

```
    config = vllm_config.model_config.hf_text_config
    init_gemm_rs(
        max_M=vllm_config.scheduler_config.max_num_batched_tokens,
        N=config.hidden_size,
    )
    logger.info_once("GEMM-RS is enabled.")
    return True


class KimiMLP(nn.Module):
    def forward(self, x):
        if self.shard_sequence_parallel:
            # 每个 rank 只持有权重分片和自己的 token，需要先 all-gather
            # 完整 token 集，再计算本 rank 的 partial
            x = sp_all_gather(x)
        gate_up, _ = self.gate_up_proj(x)
        x = self.act_fn(gate_up)

        if self.run_gemm_rs:
            from vllm.models.kimi_k3.nvidia.ops.cute_dsl.gemm_rs import get_gemm_rs

            gemm_rs = get_gemm_rs()
            # should_run 是 M>=128 的启发式：小 M 下基线（GEMM + NCCL RS）更快
            if gemm_rs.should_run(x):
                return gemm_rs(x, self.down_proj.weight)

        x, _ = self.down_proj(x)
        if self.shard_sequence_parallel:
            # 标准路径：reduce-scatter 求和并恢复序列分片
            x = sp_reduce_scatter(x)
        return x
```

评论区精华

本次 PR 没有人工技术争论，maintainer simon-mo 直接批准（stamp）。唯一实质性 review 来自自动化机器人 depthfirst-app[bot]，针对 gemm_rs.py 的越界检查：assert 0 < M <= self.max_M 在 Python -O / PYTHONOPTIMIZE 下会被移除，而内核写入的是固定大小的 NVLink 对称内存缓冲，若 M 超过 max_M 会越界写入并可能破坏跨 rank GPU 内存；should_run() 只检查下界 M>=128，没有其它上限保护。机器人建议改为显式 if/raise，与代码库其它模块"Raise (not assert) so the check survives python -O"的模式一致。该建议在 PR 合并前未看到对应修改或人工回复，属于未解决的低危隐患。

- assert 上限检查在 Python -O 下失效，可能越界写 NVLink 对称内存 (correctness): 建议把 assert 换成显式 if/raise，与代码库中 other 模块的 "Raise (not assert)" 模式一致。PR 已合并，未看到针对该线程的代码修改或人工回复，风险仍未关闭。

风险与影响

- 风险：
  1. 越界写对称内存（gemm_rs.py）：assert 0 < M <= self.max_M 在 python -O 下失效，M 超过内核缓冲区时可能写穿 NVLink 对称内存，波及所有 TP rank。opt-in 默认关闭降低了触发概率，但一旦启用且 MNBT 配置异常仍可能发生。
  2. 环境强依赖：内核依赖 torch.distributed._symmetric_memory（实验性 API）与 NCCL_NVLS，并要求所有 TP rank 在同一 NVLink 域；maybe_init_gemm_rs 只能检查配置，无法验证运行时拓扑漂移。
  3. 分支语义风险（model.py）：KimiModelForCausalLM.forward 以 hidden_states.shape[0] == M 决定是否走 reduce-scatter，依赖 GEMM-RS 返回本地分片的 shape 语义；若 should_run 在 CUDA graph 重捕获间因 M 变化切换分支，可能导致分片未归约。
  4. 显存开销：每 GPU 增加最长 max_num_batched_tokens x 7168 x 2 bytes 的对称内存工作区，MNBT=32k 时约 448 MiB，KV cache 从 39.99 GiB 降至 39.78 GiB。
  5. 测试覆盖有限：分布式测试仅覆盖 2 GPU 与部分 shape，TP16、超大 M、BN=256 与多 rank 行为未充分验证。 - 影响：对用户：默认完全无影响（opt-in），启用后仅 SM100 + BF16 + TP 2-16 且同一 NVLink 域的环境生效，条件不满足时自动降级并打印 warning。对性能：prefill 大 batch 场景 TPGS 提升 6.5%-8.2%、TTFT 下降约 5%-6%，小 M（decode 或短请求）自动回退基线。对团队：沉淀了 CuTeDSL + 对称内存的内核基础设施（mbarrier、to_cta0_smem），为未来 GEMM-AR 等融合通信内核铺路，但引入 CUTLASS 版本与 SM100 特定维护成本。对 CI：新增 distributed 区域内核测试，需要 SM100 双卡资源。 - 风险标记：assert 上限检查在 -O 下失效，仅 SM100 + 同 NVLink 域可用，依赖实验性 symmetric memory API，内核测试覆盖有限（2 GPU），opt-in 默认关闭

关联脉络

- PR #51653 [ROCm] Enable V2 model runner for Kimi-K3 on ROCm: 同属 Kimi-K3 模型支持线，涉及 Kimi-K3 的运行平台与内核路径 gating，本 PR 进一步在 NVIDIA SM100 上扩展内核能力。
- PR #52171 [Bugfix] Declare SupportsEagle3 on KimiLinearForCausalLM: 同为 Kimi-K3 模型（KimiLinearForCausalLM）的近期修复，共享 nvidia/model.py 与序列并行上下文，说明该文件仍在持续演进。
- PR #52210 [CI Failure] Fix CUDA wheel build for the Kimi K3 fused MLA kernel: 同为 Kimi-K3 的 fused kernel 相关 CI 修复，与本 PR 引入的新内核同属 Kimi-K3 内核化演进方向。