PR #24651 完整报告

sgl-project/sglang

[AMD] Add fused all-reduce RMSNorm per-group quant for Qwen3.5 FP8

合并时间：2026-07-22 22:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/24651

执行摘要

- 一句话：AMD 融合 AllReduce-RMSNorm 量化内核
- 推荐动作：值得精读。该 PR 展示了如何在框架中优雅地引入厂商特定融合内核：通过惰性门控、分层回退、元组数据契约和清晰的测试策略，值得其他平台优化参考。建议关注 layernorm.py 中的 _forward_with_allreduce_fusion_quant_per_group 回退链设计，以及 qwen3_5.py 中 _select_fused_ar_input_for_linear 的元组类型分发逻辑。

功能与动机

在 --enable-aiter-allreduce-fusion 模式下，Qwen3.5-FP8 的 *decode* 每层需要额外两个内核（dynamic_per_group_scaled_quant 和 AR+RMSNorm），将 AllReduce、RMSNorm 和 per-group 量化融合可减少内核启动次数，提升推理吞吐。PR body 中提供了详细的 benchmark 数据：*cc*=2 时输出 tok/s 从 186.42 提升至 197.21（+5.8%），TPOT 从 10.46ms 降至 9.88ms。

实现拆解

1. 新增分布式 API：在 communication_op.py 和 parallel_state.py 中添加 tensor_model_parallel_fused_allreduce_rmsnorm_quant_per_group 和 GroupCoordinator.fused_allreduce_rmsnorm_quant_per_group，封装 aiter 的融合内核调用，并做形状、平台等检查，不满足条件时返回 None 让调用方回退。
2. 新增 layernorm 融合量化辅助函数：在 layernorm.py 中添加 _forward_with_allreduce_fusion_quant_per_group，实现三级回退链：① aiter 单内核 AR+RMSNorm+per-group 量化（gfx95 专用）；② 惯用 AR+RMSNorm + 独立 per-group 量化（2 内核）；③ 返回 None 走通用路径。同时增加 _get_aiter_per_group_quant 惰性获取 aiter 量化器。
3. 模型层适配：在 qwen3_5.py 中添加 _enable_qwen35_fused_ar_quant 门控（基于 --enable-aiter-allreduce-fusion 和 SGLANG_DISABLE_FUSED_AR_QUANT 环境变量），以及 _select_fused_ar_input_for_linear 处理融合路径返回的 ((fp8,scale), residual) 或 ((bf16,fp8,scale), residual) 元组，使得标准注意力层直接消费 (fp8,scale)，GDN 层消费 (bf16,fp8,scale) 分别传给 in_proj_qkvz 和 in_proj_ba。GDN._forward_input_proj 和 Qwen3_5DecoderLayer.prepare_attn 均已接入新路径。
4. Benchmark 与测试：新增 benchmark/kernels/all_reduce/benchmark_fused_ar_rms_quant_amd.py 用于三个变体的性能对比和正确性验证；新增 test/registered/amd/perf/mi35

x/test_qwen35_fp8_ar_fusion_mi35x.py 作为 PR CI 精度测试，使用 GSM8K 数据集在双 TP4 服务器上并行运行融合路径和禁用路径（环境变量SGLANG_DISABLE_FUSED_AR_QUANT=1），门控精度阈值为 0.94。

5. CI 配置调整：修改 .github/workflows/pr-test-amd.yml 和 pr-test-amd-rocm720.yml，调整 stage 分区以容纳新增的大模型测试。

关键文件：

- python/sglang/srt/layers/layernorm.py（模块 归一化层；类别 source；类型 core-logic；符号 _get_aiter_per_group_quant, _forward_with_allreduce_fusion_quant_per_group, forward_with_allreduce_fusion_quant_per_group）：核心融合逻辑所在，新增 _forward_with_allreduce_fusion_quant_per_group 实现三级回退链，以及 _get_aiter_per_group_quant 惰性获取 aiter 量化器。
- python/sglang/srt/models/qwen3_5.py（模块 模型层；类别 source；类型 data-contract；符号 _enable_qwen35_fused_ar_quant, _linear_accepts_fp8_tuple, _select_fused_ar_input_for_linear, _forward_input_proj_fused_quant_amd）：模型层适配，新增门控函数和元组分发逻辑，使标准注意力和 GDN 层正确消费融合路径的输出。
- python/sglang/srt/distributed/parallel_state.py（模块 分布式状态；类别 source；类型 core-logic；符号 fused_allreduce_rmsnorm_quant_per_group）：在 GroupCoordinator 上添加 fused_allreduce_rmsnorm_quant_per_group 方法，封装 aiter 自定义融合内核的调用，包含形状检查和回退。
- python/sglang/srt/distributed/communication_op.py（模块 通信算子；类别 source；类型 core-logic；符号 tensor_model_parallel_fused_allreduce_rmsnorm_quant_per_group）：新增顶级函数 tensor_model_parallel_fused_allreduce_rmsnorm_quant_per_group 作为统一入口，转发至 GroupCoordinator 的方法。
- benchmark/kernels/all_reduce/benchmark_fused_ar_rms_quant_amd.py（模块 基准测试；类别 source；类型 dependency-wiring；符号 parse_shapes, dtype_from_name, _barrier, _mean_across_ranks）：新增基准测试，对比三个变体（3 内核、2 内核、1 内核）的性能和数值正确性，支持随机形状和 TorchDynamo 捕获。
- test/registered/amd/perf/mi35x/test_qwen35_fp8_ar_fusion_mi35x.py（模块 测试用例；类别 test；类型 test-coverage；符号 FusionVariant, _base_url_with_port_offset, get_fusion_variants, _parse_gsm8k_metrics）：新增 PR CI 精度测试，在 MI35x 上并行运行融合路径和禁用路径，验证 GSM8K 精度不低于 0.94。
- python/sglang/srt/layers/communicator.py（模块 通信器；类别 source；类型 core-logic）：LayerCommunicator.prepare_attn 方法更新，优先尝试融合量化路径，并在不可用时回退到原有 AR+RMSNorm 融合。
- .github/workflows/pr-test-amd.yml（模块 CI 配置；类别 infra；类型 infrastructure）：调整 AMD PR CI 舞台分区，以容纳新增的 8 分钟 stage-c 大模型测试。
- .github/workflows/pr-test-amd-rocm720.yml（模块 CI 配置；类别 infra；类型 infrastructure）：同步调整 ROCm7.2 的 PR CI 配置以保持与主 AMD CI 一致。

关键符号：tensor_model_parallel_fused_allreduce_rmsnorm_quant_per_group, GroupCoordinator.fused_allreduce_rmsnorm_quant_per_group,

_forward_with_allreduce_fusion_quant_per_group,
_enable_qwen35_fused_ar_quant,_select_fused_ar_input_for_linear,
_linear_accepts_fp8_tuple,Qwen3_5GatedDeltaNet._forward_input_proj_fused_quant_amd,LayerCommunicator.prepare_attn

关键源码片段

python/sglang/srt/layers/layernorm.py

核心融合逻辑所在，新增 _forward_with_allreduce_fusion_quant_per_group 实现三级回退链，以及 _get_aiter_per_group_quant 惰性获取 aiter 量化器。

```
# 新增的融合 AR+RMSNorm+per-group 量化前向函数（layernorm.py）

def _forward_with_allreduce_fusion_quant_per_group(
    norm_module,
    x: torch.Tensor,
    residual: Optional[torch.Tensor],
    weight: torch.Tensor,
    group_size: int = 128,
    use_attn_tp_group: bool = True,
    keep_bf16: bool = False,
):
    """融合 AR + RMSNorm + per-group FP8 量化，含优雅的三级回退。

    返回值（优先级递减）：
      1. `((fp8, scale), residual)` 当 keep_bf16=False。
      2. `((bf16, fp8, scale), residual)` 当 keep_bf16=True（用于 GDN 层）。
      3. `None` 表示无法融合，调用方需回退至普通路径。

    keep_bf16 用于 GDN：其 in_proj_qkvz 需要 FP8，in_proj_ba 需要 bf16，
    同时输出 bf16 避免损失性反量化。
    """
    if residual is None or not _use_aiter:
        return None

    from sglang.srt.distributed import (
        tensor_model_parallel_fused_allreduce_rmsnorm,
        tensor_model_parallel_fused_allreduce_rmsnorm_quant_per_group,
    )

    # 尝试 1：单内核完全融合（aiter gfx95 专用）
    # 该调用在底层检查 is_gfx95_supported() 和 ca_comm 可用性
    out = tensor_model_parallel_fused_allreduce_rmsnorm_quant_per_group(
        x,
        residual,
        weight,
        eps=norm_module.variance_epsilon,
        group_size=group_size,
```

```
        emit_bf16=keep_bf16,
    )
    if out is not None:
        # 返回格式统一为 (fp8, residual_out, scale[, bf16]) via emit_bf16
        return out

    # 尝试 2：两内核路径（AR+RMSNorm + 独立 per-group 量化）
    fused_out = tensor_model_parallel_fused_allreduce_rmsnorm(
        x, residual, weight, eps=norm_module.variance_epsilon
    )
    if fused_out is not None:
        # 对 fused_out[0]（归一化后的 bf16 激活）执行 per-1x128 量化
        quant_fn, fp8_dtype = _get_aiter_per_group_quant()
        out_fp8, out_scale = quant_fn(fused_out[0].contiguous(), group_size)
        # 根据 keep_bf16 决定是否返回 bf16
        if keep_bf16:
            return (out_fp8, out_scale), fused_out[1], fused_out[0]
        else:
            return (out_fp8, out_scale), fused_out[1]

    # 尝试 3：无法进行任何融合，返回 None
    return None
```

python/sglang/srt/models/qwen3_5.py

模型层适配，新增门控函数和元组分发逻辑，使标准注意力和 GDN 层正确消费融合路径的输出。

```
# qwen3_5.py 中新增的门控与元组选择逻辑

@lru_cache(maxsize=1)
def _enable_qwen35_fused_ar_quant() -> bool:
    """门控 Qwen3.5 融合 AR+RMSNorm+per-group 路径。

    条件：aiter && 未禁用环境变量 && --enable-aiter-allreduce-fusion。
    禁用环境变量 SGLANG_DISABLE_FUSED_AR_QUANT 可单独关闭此路径，
    同时保留基础的 AR+RMSNorm 融合。
    """
    if not _use_aiter:
        return False
    if get_bool_env_var("SGLANG_DISABLE_FUSED_AR_QUANT", default="false"):
        return False
    return bool(get_server_args().enable_aiter_allreduce_fusion)


def _select_fused_ar_input_for_linear(hidden_states, linear: nn.Module):
    """根据 linear 的量化类型自动选择 FP8 或 bf16 分量。"""
    if not isinstance(hidden_states, tuple):
        return hidden_states # 非融合路径，直接返回
    if len(hidden_states) == 3:
```

```
        hs_bf16, hs_fp8, hs_scale = hidden_states
        # 如果 linear 是 FP8 量化层（Fp8LinearMethod 且 block_quant 或 use_mxfp8）
        if _linear_accepts_fp8_tuple(linear):
            return (hs_fp8, hs_scale)
        else:
            return hs_bf16 # 否则使用 bf16
    if len(hidden_states) == 2 and _linear_accepts_fp8_tuple(linear):
        return hidden_states # 直接 (fp8, scale) 传给 FP8 层
    raise TypeError(
        f"{linear.__class__.__name__} cannot consume fused AR quant tuple input"
    )
```

评论区精华

HaiShaw 在 review 中指出「最好限制使用过度供应商特定的全局变量，可以将其作为参数传递给供应商特定方法」。随后 hubertlu-tw 回应已重构，将 _aiter_per_1x128_quant 和 _aiter_fp8_dtype 收进 _get_aiter_per_group_quant 惰性函数，避免模块级全局变量污染。此外，关于测试覆盖的讨论较多：初期夜间测试和 PR CI 测试的注册问题，经过 yichiche 和 yctseng0211 的多次提交调整，最终确定了分离方案：夜间压测保持独立，PR CI 仅做 GSM8K 精度验证。

- 避免过度使用厂商特定全局变量 (design): hubertlu-tw 接受建议并重构，使用 _get_aiter_per_group_quant 惰性函数取代模块级全局变量，将 aiter 量化器和 dtype 封装在函数内，降低命名空间污染。

风险与影响

- 风险：
  1. 平台绑定风险：融合内核仅对 AMD gfx95 类 GPU（通过 is_gfx95_supported() 检测）启用，其他 AMD 卡会走回退路径，不影响功能但可能未充分验证回退路径。
  2. 精度不确定性：GSM8K 精度阈值设为 0.94，但实际跑分可能因环境浮动；PR 测试中曾出现精度 0.001（失败），后查明是 benchmark 脚本路径问题，最终修复后达到 0.95+。需要持续监控。
  3. CUDA 兼容性：_linear_accepts_fp8_tuple 和 _select_fused_ar_input_for_linear 的改动被 _use_aiter 和 isinstance(hidden_states, tuple) 门控，不应影响 CUDA 路径，但 CI 中 CUDA 测试曾因全局变量门控问题报错（已修复）。
  4. 维护成本：新增的融合路径增加了 layernorm.py 和 parallel_state.py 的复杂度，未来其他模型若要使用需手动接入元组 handoff。 - 影响：用户：使用 Qwen3.5-FP8 模型并在 AMD MI35x 上启用 --enable-aiter-allreduce-fusion 的用户可获得 5-6% 的 decode 吞吐提升，其他 AMD GPU 或平台无变化。系统：新增约 1K 行代码，集中分布在 AMD 相关模块；CI 测试新增一个 8-GPU stage，可能增加排队时间。团队：AMD 团队需维护该厂商特定优化，并确保未来 layernorm 或模型层改动不破坏此路径。
- 风险标记：仅 AMD gfx95 支持，精度依赖环境配置，缺少非 AMD 平台回归测试用例

关联脉络

- PR #29275 Add materialize_bpreshuffle_fp8_scale utility for AMD FP8 attention: 本 PR 在后续提交中依赖了 materialize_bpreshuffle_fp8_scale 用于 bpreshuffle 精度修复，该工具函数即由 #29275 引入。
- PR #30940 [AMD] Add fused all-reduce RMSNorm per-group quant for Qwen3.5 FP8 (preparatory refactor): 准备性重构 PR，为当前 PR 扫清基础设施障碍，修复了 CI 路径问题并被本 PR 合并。