PR #29007 完整报告

sgl-project/sglang

Fix MoE TP allreduce to use NCCL symmetric memory via in-pool output allocation

合并时间：2026-07-15 15:06

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/29007

执行摘要

- 一句话：MoE 输出分配至对称内存池，allreduce 延迟降 50%
- 推荐动作：值得精读，尤其是 use_symmetric_memory 上下文管理器的条件使用模式，以及通过上游分配消除下游数据复制的思路。该 PR 展示了在分布式训练 / 推理框架中精细控制内存分配位置以获得通信性能优化的典型方法。

功能与动机

When --enable-symm-mem is set, SGLang's MoE layer should ensure the tensor passed to tensor_model_parallel_all_reduce resides in the NCCL symmetric memory pool so the fast path is taken. However, MoE runners allocate output buffers outside the pool, and upstream hidden_states may also not be in the pool. This PR fixes both issues.

实现拆解

1. 在 MoE runner 输出分配处启用对称内存：修改 deep_gemm.py 中所有生成最终 down_output 的路径（_run_contiguous_gemm, _run_bf16_contiguous_gemm, _run_masked_gemm, _run_masked_bf16_gemm, post_permute_deep_gemm_to_standard），使用 use_symmetric_memory(get_tp_group(), disabled=not is_allocation_symmetric()) 上下文包裹 torch.empty 调用。类似地修改 fused_moe.py 中 _fused_moe_kernel_sequence 的非 inplace 分支。中间缓存仍使用默认分配器以限制池占用。
2. 将 hc_pre 输出纳入对称池：在 deepseek_v4.py 的 hc_pre_torch_impl 中，将 y（用于非 TileLang 路径的 MoE 输入）的分配包裹在对称上下文内。在 mhc.py 的 _mhc_pre_impl 和 mhc_fused_post_pre 中同样处理 layer_input 和 layer_input_cur。这样 Triton in-place 的 MoE runner 直接将专家输出写回对称缓冲，使 allreduce 输入天然对称。
3. 修复 dp_attention 预加载崩溃：_DpGatheredBufferWrapper 的类属性缺少默认值，导致加载时 mhc_pre prewarm 因读取未初始化的 _dp_max_padding 而崩溃。为其提供合理默认值（False），保证 prewarm 期间安全。
4. 适配单元测试：在 test_mhc_kernels.py 中通过 monkeypatch 将 use_symmetric_memory 替换为 nullcontext，将 is_allocation_symmetric 和 get_tp_group 替换为假值，避免单进程测试因无 TP 组而失败。

5. 移除冗余复制：早期版本在 MoE combine 后检测数据指针并复制，现已由上游分配解决，复制逻辑被移除。

关键文件：

- python/sglang/srt/layers/moe/moe_runner/deep_gemm.py（模块 MoE 执行器；类别 source；类型 core-logic；符号 _run_contiguous_gemm, _run_bf16_contiguous_gemm, _run_masked_gemm, _run_masked_bf16_gemm）：DeepGEMM MoE 执行器，5 处输出分配改为对称内存分配，是核心变更点。
- python/sglang/srt/models/deepseek_v4.py（模块 模型层；类别 source；类型 data-contract；符号 hc_pre_torch_impl）：DeepSeek-V4 模型定义，将 hc_pre 输出的分配移至对称池，消除下游数据复制。
- python/sglang/srt/layers/moe/moe_runner/triton_utils/fused_moe.py（模块 MoE 内核；类别 source；类型 core-logic；符号 _fused_moe_kernel_sequence）：Triton MoE 执行器，非 inplace 分支的输出分配改为对称内存分配。
- python/sglang/srt/layers/dp_attention.py（模块 DP 注意力；类别 source；类型 bugfix；符号 _DpGatheredBufferWrapper）：修复 _DpGatheredBufferWrapper 类属性默认值，避免 DP+DSV4 预加载阶段因读取未初始化属性崩溃。
- python/sglang/kernels/ops/layernorm/mhc.py（模块 MHClayer；类别 infra；类型 infrastructure；符号 _mhc_pre_impl, mhc_fused_post_pre）：mHC 层（TileLang 路径）的 MoE 输入分配改为对称内存，与 deepseek_v4.py 互补。
- test/registered/kernels/test_mhc_kernels.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_mhc_fused_post_pre_matches_unfused）：适配对称内存引入的单元测试，确保无 TP 组的单进程测试通过 mock 通过。

关键符号：_run_contiguous_gemm, _run_bf16_contiguous_gemm, _run_masked_gemm, _run_masked_bf16_gemm, post_permute_deep_gemm_to_standard, _fused_moe_kernel_sequence, hc_pre_torch_impl, _mhc_pre_impl, mhc_fused_post_pre

关键源码片段

python/sglang/srt/layers/moe/moe_runner/deep_gemm.py

DeepGEMM MoE 执行器，5 处输出分配改为对称内存分配，是核心变更点。

```
# 以 _run_contiguous_gemm 为例，展示对称内存分配模式。
import torch
from sglang.srt.distributed import get_tp_group
from sglang.srt.distributed.device_communicators.pynccl_allocator import use_symmetric_
memory
from sglang.srt.layers.dp_attention import is_allocation_symmetric

def _run_contiguous_gemm(self, runner_input, quant_info, running_state):
    # ... 前面计算 gateup_output, down_input 等，使用普通分配 ...

    del down_input # 释放中间缓存，节省显存

    # 将最终输出放在 NCCL 对称内存池中，使下游 all-reduce 使用快速路径。
```

```
    # 只有此最终输出进入池；中间缓存留在默认分配器以限制池占用。
    with use_symmetric_memory(
        get_tp_group(), disabled=not is_allocation_symmetric()
    ):
        down_output = torch.empty(
            (all_tokens, K),
            device=hidden_states_device,
            dtype=torch.bfloat16,
        )

    # TMA 对齐缩放（如果需要）
    if deep_gemm_wrapper.DEEPGEMM_NEED_TMA_ALIGNED_SCALES:
        down_input_scale = tma_align_input_scale(down_input_scale)

    # 执行 grouped GEMM 将结果写入 down_output
    deep_gemm_wrapper.grouped_gemm_nt_f8f8bf16_contig(
        (down_input_fp8, down_input_scale),
        w2_weight_fp8,
        down_output,
        m_indices,
        recipe_a=recipe_a,
        recipe_b=recipe_b,
    )

    return down_output
```

python/sglang/srt/layers/dp_attention.py

修复 _DpGatheredBufferWrapper 类属性默认值，避免 DP+DSV4 预加载阶段因读取未初始化属性崩溃。

```
class _DpGatheredBufferWrapper:
    """Facade for the DP gathered-buffer state: allocation metadata lives on
    ``flags.dp`` (set once at initialize_dp_attention). The per-forward
    sizing quartet stays as class attributes..."""

    # 为类属性提供默认值，避免在第一次 forward 之前
    # （如加载时的 mhc_pre prewarm）因读取未初始化的属性而崩溃。
    # `_dp_max_padding` 默认为 False（非对称分配），对于 prewarm
    # 安全，因为它只编译内核，不执行实际的 all-reduce。
    _global_dp_buffer_len: int = 0
    _local_dp_buffer_len: int = 0
    _dp_max_padding: bool = False
    _global_num_tokens: Optional[List[int]] = None

    @classmethod
    def set_metadata(cls, hidden_size: int, dtype: torch.dtype, device: torch.device):
        from sglang.srt.runtime_context import get_flags
        dp = get_flags().dp
        dp.buffer_hidden_size = hidden_size
```

```
        dp.buffer_dtype = dtype
        dp.buffer_device = device

    @classmethod
    def set_dp_buffer_len(
        cls,
        global_dp_buffer_len: int,
        local_dp_buffer_len: int,
        dp_max_padding: bool,
        global_num_tokens: Optional[List[int]] = None,
    ):
        cls._global_dp_buffer_len = global_dp_buffer_len
        cls._local_dp_buffer_len = local_dp_buffer_len
        cls._dp_max_padding = dp_max_padding
        cls._global_num_tokens = global_num_tokens
```

评论区精华

核心讨论围绕设计简化与消除复制：

- nvcastet建议直接使用 use_symmetric_memory 上下文管理器包裹现有分配路径，避免引入额外的 moe_output_buffer_ctx 和条件分支（原设计）。作者采纳，代码显著精简。
- nvcastet进一步建议通过上游分配（hc_pre）使 MoE 输入处于对称池以避免 downstream 复制，并提议加入 SGLANG_DEBUG_SYMM_MEM 检测未注册内存。作者将对称分配移至 hc_pre 的 y 和 mhc 的 layer_input，移除了 fallback 复制逻辑。
- nvcastet询问 y 与 layer_input 的关系，作者澄清两者互斥，分别对应非 TileLang 路径（deepseek_v4.py）和 TileLang 路径（mhc.py）。
- gemini-code-assist[bot]警告无条件分配对称内存导致 OOM，但实际代码通过 disabled=not is_allocation_symmetric() 条件保护，该警告已过时。
- 使用上下文管理器代替 moe_output_buffer_ctx (design): 作者采纳，移除 moe_output_buffer_ctx，在 runner 输出分配处直接使用上下文管理器。
- 消除数据复制，上游分配对称内存 (performance): 作者将对称分配移至 hc_pre 的 y 和 mhc 的 layer_input，移除了之前的 fallback 复制逻辑。
- hc_pre 中 y 与 layer_input 的关系 (question): 作者解释两者互斥，分别对应非 TileLang 路径（deepseek_v4.py）和 TileLang 路径（mhc.py）。
- 无条件分配对称内存导致 OOM (correctness): 代码实际通过 disabled=not is_allocation_symmetric() 条件分配，该警告已过时（针对旧设计）。

风险与影响

- 风险：主要风险：对称内存池大小有限，若过多输出进入池可能增加池压力。但本 PR 仅将最终 MoE 输出（非中间缓存）放入池，且仅在 is_allocation_symmetric() 为真时启用，风险可控。对非 TP 或 EP-only 场景，通过 disabled=not is_allocation_symmetric() 保护，无额外分配。对 mhc.py 的修改影响 TileLang 路径（非默认），且有测试覆盖。整体风险低。

- 影响：对用户：启用 --enable-symm-mem 时，DeepSeek-V4 类模型的 MoE allreduce 延迟降低约 50%，端到端 TPOT 降低约 6.5%。未启用标志的用户无变化。对系统：新增 pynccl_allocator.use_symmetric_memory 依赖。对团队：需注意未来新的 MoE runner 应遵循相同的对称内存分配模式。
- 风险标记：对称内存池占用可能增加，条件分配保护非对称场景，hc_pre 与 mhc 路径互斥需同步维护，prewarm 崩溃已修复

关联脉络

- 暂无明显关联 PR