PR #26473 完整报告

sgl-project/sglang

[MoE] Support BF16 standard A2A with DeepGEMM runner

合并时间：2026-06-02 11:40

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26473

执行摘要

- 一句话：修复 DeepGEMM runner 中 BF16 A2A 和专家 0 遗漏
- 推荐动作：建议精读。该 PR 解决了实际运行中的关键问题，并展示了在 Triton kernel 中如何安全地提升数值精度（FP32 累积）。设计决策值得参考，尤其是条件量化路径的选择。如果团队在使用 DeepGEMM 运行时，建议尽快合并此 PR 并做回归验证。

功能与动机

The main motivation of this PR is to fix the DeepGEMM MoE runner path when --moe-runner-backend deep_gemm is used with the standard MoE A2A backend. The primary issue is that post_reorder_triton_kernel did not handle expert 0 when combining routed outputs. As a result, routed outputs from expert 0 were skipped in the DeepGEMM runner combine step, which affected model accuracy. This PR also fixes BF16 model support for the same backend combination. With BF16 experts and --moe-runner-backend deep_gemm, the standard A2A to DeepGEMM preprocess path could produce FP8 activations while the runner selected the BF16 masked DeepGEMM path, failing during CUDA graph capture.

实现拆解

1. 修复专家 0 遗漏：在 python/sglang/srt/layers/moe/ep_moe/kernels.py 的 post_reorder_triton_kernel 函数中，将专家 ID 过滤条件从 expert_id > 0 改为 expert_id >= 0，确保排序索引为 0 的专家被正常加权组合。这是导致精度下降的直接原因。
2. 提升数值精度：在同一个文件的两个 Triton kernel 中引入 FP32 累积：_silu_and_mul_kernel 将 gate 和 up 都先转为 tl.float32 计算 SiLU，再转回输入 dtype 存储；post_reorder_triton_kernel 的加权和累加器从 InDtype 改为 tl.float32，权重和输入都先转 fp32，最后再转回输出 dtype。这减少了低精度下的舍入误差。
3. 支持 BF16 A2A：在 python/sglang/srt/layers/moe/moe_runner/deep_gemm.py 的 pre_permute_standard_to_deep_gemm 函数中，新增 output_dtype 推导逻辑：若 quant_info.w13_weight.dtype 为 bf16，则输出 torch.bfloat16；否则保留 FP8。该输出 dtype 作为参数传入 moe_ep_deepgemm_preprocess。在 kernels.py 的预处理函数中，依据 is_fp8（由 output_dtype 决定）条件性地进行 FP8 逐 token 缩放量化，在 BF16 模式下直接跳过量化步骤。

4. 修复 BF16 grouped contiguous warmup：在 python/sglang/srt/layers/deep_gemm_wrapper/compile_utils.py 的 _BF16GroupedContWarmupExecutor.execute 中，将 deep_gemm.m_grouped_bf16_gemm_nt_contiguous 调用改为使用位置参数 self.m_indices[:m] 而非关键字参数 m_indices=self.m_indices[:m]，以兼容 sgl-deep-gemm 0.1.1 的 API 变化。
5. 升级依赖与镜像：将 sgl-deep-gemm 版本从 0.1.0 提升至 0.1.1，同步更新 python/pyproject.toml 和 docker/Dockerfile 中的版本号。0.1.1 提供了 BF16 masked grouped GEMM 的必要支持。

关键文件：

- python/sglang/srt/layers/moe/ep_moe/kernels.py（模块 MoE 内核；类别 source；类型 core-logic；符号 post_reorder_triton_kernel, _silu_and_mul_kernel, moe_ep_deepgemm_preprocess, fill_gateup_input_triton_kernel）：核心修复文件：修复专家 0 遗漏、引入 FP32 累积、添加 BF16 条件量化路径。
- python/sglang/srt/layers/moe/moe_runner/deep_gemm.py（模块 DeepGEMM 适配；类别 source；类型 core-logic；符号 pre_permute_standard_to_deep_gemm）：添加 output_dtype 推导逻辑，使预处理能够根据权重 dtype 决定输出格式，是 BF16 支持的关键。
- python/sglang/srt/layers/deep_gemm_wrapper/compile_utils.py（模块 编译预热；类别 source；类型 core-logic；符号 _BF16GroupedContWarmupExecutor）：修复 BF16 grouped contiguous warmup 调用方式，移除关键字参数以兼容新版 sgl-deep-gemm。
- python/pyproject.toml（模块 依赖配置；类别 config；类型 configuration）：升级 sgl-deep-gemm 依赖版本至 0.1.1，提供 BF16 masked grouped GEMM 的必要支持。
- docker/Dockerfile（模块 部署脚本；类别 infra；类型 infrastructure）：同步升级 Dockerfile 中的 sgl-deep-gemm 版本号以保持镜像一致性。

关键符号：post_reorder_triton_kernel, _silu_and_mul_kernel, moe_ep_deepgemm_preprocess, fill_gateup_input_triton_kernel, pre_permute_standard_to_deep_gemm, _BF16GroupedContWarmupExecutor.execute

关键源码片段

python/sglang/srt/layers/moe/ep_moe/kernels.py

核心修复文件：修复专家 0 遗漏、引入 FP32 累积、添加 BF16 条件量化路径。

```
@triton.jit defpost_reorder_triton_kernel(     down_output_ptr,     output_ptr,
src2dst_ptr,     topk_ids_ptr,     topk_weights_ptr,     topk,     hidden_size,
BLOCK_SIZE: tl.constexpr, ):     InDtype = down_output_ptr.dtype.element_ty
src_idx_int32 = tl.program_id(0)     src_idx = src_idx_int32.to(tl.int64)     src2dst_ptr =
src2dst_ptr + src_idx * topk     topk_ids_ptr = topk_ids_ptr + src_idx * topk
topk_weights_ptr = topk_weights_ptr + src_idx * topk     store_ptr = output_ptr +
src_idx * hidden_size     vec = tl.arange(0, BLOCK_SIZE)     for start_offset in
tl.range(0, hidden_size, BLOCK_SIZE):         offset = start_offset + vec         mask =
offset < hidden_size         # 使用 FP32 累积以减少精度损失         sum_vec =
```

tl.zeros([BLOCK_SIZE], dtype=tl.float32) for idx in range(topk): expert_id = tl.load(topk_ids_ptr + idx) # 关键修复：使用 >= 0 以包括专家 0（之前遗漏） if expert_id >= 0: dst_idx_int32 = tl.load(src2dst_ptr + idx) dst_idx = dst_idx_int32.to(tl.int64) weigh_scale = tl.load(topk_weights_ptr + idx).to(tl.float32) load_ptr = down_output_ptr + dst_idx * hidden_size # 在 FP32 中累加专家输出以获得更好精度，然后转换为最终输出 dtype in_data = tl.load(load_ptr + offset, mask=mask).to(tl.float32) sum_vec += in_data * weigh_scale tl.store(store_ptr + offset, sum_vec.to(InDtype), mask=mask) @triton.jit def silu_and_mul_kernel(...): # ... 前置代码省略 for token_index in tl.range(token_id, token_num_cur_expert, block_num_per_expert, num_stages=NUM_STAGE): gate = tl.load(input_ptr_offs + token_index * stride_input_1, mask=offs_in_d < size_n, other=0.0).to(tl.float32) up = tl.load(input_ptr_offs + token_index * stride_input_1 + size_n, mask=offs_in_d < size_n, other=0.0).to(tl.float32) gate = gate / (1 + tl.exp(-gate)) gate_up = up * gate # 在 FP32 中计算 SiLU 以提高精度，然后转换回输入 dtype gate_up = gate_up.to(input_ptr.dtype.element_ty) tl.store(output_ptr_offs + token_index * stride_output_1, gate_up, mask=offs_in_d < size_n) defmoe_ep_deepgemm_preprocess(topk_ids, num_local_experts, hidden_states, top_k, block_shape, output_dtype=torch.float8_e4m3fn, # 新增参数，默认为 FP8): # ... 初始化代码省略 is_fp8 = output_dtype == torch.float8_e4m3fn if is_fp8: # FP8 路径：执行缩放及量化 ... else: # BF16 路径：跳过 FP8 量化，保持为 BF16 ...

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

添加 output_dtype 推导逻辑，使预处理能够根据权重 dtype 决定输出格式，是 BF16 支持的关键。

```
@register_pre_permute("standard", "deep_gemm")
def pre_permute_standard_to_deep_gemm(
    dispatch_output: StandardDispatchOutput,
    quant_info: DeepGemmMoeQuantInfo,
    runner_config: MoeRunnerConfig,
    running_state: dict,
) -> DeepGemmRunnerInput:
    # ... 前置代码省略
    # 根据权重的 dtype 决定输出 dtype，匹配 runner 的 GEMM 分发逻辑
    output_dtype = (
        torch.bfloat16
        if quant_info.w13_weight.dtype == torch.bfloat16
        else torch.float8_e4m3fn
    )
    masked_m, expected_m, src2dst, hidden_states, hidden_states_scale = (
        moe_ep_deepgemm_preprocess(
            topk_ids,
            runner_config.num_local_experts,
            hidden_states,
```

```
            runner_config.top_k,
            quant_info.block_shape,
            output_dtype=output_dtype, # 传入 output_dtype
        )
    )
    # ... 后续代码不变
```

评论区精华

- 审查者 BBuf 指出 DeepGEMM 仓库的 PR #37 新增了 BF16 contiguous API，但参数名改为 grouped_layout，而 SGLang 的预热代码使用 m_indices= 关键字。作者移除关键字后，BBuf 确认兼容性没问题。
- gemini-code-assist 建议 _silu_and_mul_kernel 中 up 也应显式转为 tl.float32 再与 gate 相乘，避免 Triton 编译错误。该建议已被采纳，实际代码中 up 已使用 .to(tl.float32) 处理。
- 无其他争议，BBuf 和 Fridge003 均给予了批准。
- BF16 contiguous API 参数名兼容性 (other): 作者移除关键字参数，BBuf 确认兼容性没问题。
- up 张量应显式转为 float32 (correctness): 已被采纳，代码中 up 已使用 .to(tl.float32) 处理。

风险与影响

- 风险：
  - 核心路径变更：修改了 MoE 计算的关键 kernel（post_reorder_triton_kernel、_silu_and_mul_kernel、预处理函数），存在引入数值回归的风险，尤其是在已支持的 FP8 模型上。
  - 缺少单元测试覆盖：PR 没有新增测试用例，修复依赖于手动 GSM8K 验证。如果后期有其他分支修改了这些 kernel，可能难以快速发现回归。
  - 依赖兼容性：升级 sgl-deep-gemm 到 0.1.1 并调整 API 调用，若未来版本再次改变接口可能导致启动失败。
- 影响：
  - 用户影响：使用 --moe-runner-backend deep_gemm 并搭配标准 A2A 后端的用户将能正确运行 BF16 模型，FP8 模型精度在 GSM8K 上从 0.794 提升至 0.813。未使用该参数的用户无影响。
  - 系统影响：数值精度累积方式变更可能使结果与之前版本不一致，但属于预期提升。无显著性能退化（GSM8K 输出吞吐量从 1478 token/s 略升至 1486 token/s）。
  - 团队影响：需要保持对 sgl-deep-gemm 版本的跟踪，确保 API 向后兼容。
  - 风险标记：核心路径变更，缺少测试覆盖，依赖升级

关联脉络

- PR #26567 Speed up DeepGEMM JIT warmup with per-PP-rank parallel compile: 修改了同一个文件 python/sglang/srt/layers/deep_gemm_wrapper/compile_utils.py，与本

PR 的 BF16 warmup fix 形成同一模块的改进序列。