

PR #36278 完整报告

sgl-project/sglang

[XPU] Add xpu forward in Gemma3RMSNorm & Add test and benchmark for Gemma3RMSNorm

合并时间: 2026-09-11 10:32

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/36278>

执行摘要

- 一句话: Gemma3RMSNorm 新增 XPU forward, 并补齐 2D/3D/4D 测试覆盖
- 推荐动作: 推荐精读, 但属于小范围平台补齐, 可快速浏览。建议重点关注三点: (1) forward_xpu 中三种分支的取舍——为什么 3D/4D 只在 residual is None 时走 kernel, 以及这如何与 CUDA 分支产生语义分叉; (2) _run_gemma3_rms_norm_test 的 non_contiguous 分支, 它是理解 Gemma3 q_norm/k_norm 真实视图形状的最佳入口; (3) review 暴露的流程问题——设备相关测试放在 test/manual 无法形成 CI 门禁, 若后续要复用这套用例, 应参考 #32093 的做法迁入 XPU nightly。此外建议顺手清理已失效的 "other dim == 1" 注释, 并明确 sgl-kernel-xpu 的最低版本约束。

功能与动机

PR body 明确说明动机: 为新引入的 Gemma3RMSNorm 层补齐支持与测试, 因为 sglang-kernel-xpu 已开始接受 4D 形状 ("since the 4d shape is accepted in sglang-kernel-xpu")。此前 Gemma3RMSNorm 只有 forward_native/forward_cuda/forward_cpu/forward_hip/forward_npu 等分支, XPU 平台缺少专门实现; 同时现有 forward_cuda 仅对 `x.dim() == 2` 使用融合算子, 而 Gemma3 的 q_norm/k_norm 实际喂入的是从更宽 qkv 张量切分并 unflatten 出来的 3D/4D 非连续视图, 因此需要显式扩展维度支持并用测试固化这一行为。issue 评论中 CaoE 指出该 PR "Depends on sgl-project/sgl-kernel-xpu#367", 作者确认依赖 PR 已合并。

实现拆解

1. 分发入口与背景: python/sglang/srt/layers/layernorm.py 中的 Gemma3RMSNorm 继承 BaseFusedOp, 运行期按设备选择 forward_native / forward_cuda / forward_xpu 等方法。forward_native 是纯 PyTorch 参考实现 (先 `x + residual`, 再 FP32 归一化, 最后 `output * (1.0 + self.weight.float())` 并按 `x` 的 dtype 转回), 也是所有平台分支的兜底路径。
2. 核心逻辑 (+13 行): 新增 forward_xpu, 分三种情况——residual is not None and `x.dim() == 2` 调用 `gemma_fused_add_rmsnorm` 原地更新 `x` 与 `residual` 并返回二者, 与 CUDA 分支语义一致; residual is None and `x.dim() in (2, 3, 4)` 调用 `gemma_rmsnorm`, 注释说明 XPU kernel 内部会自动拍平前导维度, 无需 Python 侧 reshape, 这是相对 forward_cuda 只支持 `dim == 2` 的关键扩展; 其余组合 (带 residual 的非 2D、维度不在 2/3/4 内) 回落 forward_native, 保证正确性。

3. 测试配套: test/manual/layers/test_layernorm.py 新增 TestGemma3RMSNorm (+173/-1), setUpClass 改为 CUDA 或 XPU 任一可用即执行, 并把 default device 切到对应平台; _run_gemma3_rms_norm_test 以 forward_native 为基准, 用 torch.allclose(atol=1e-2, rtol=1e-2) 校验带 / 不带 residual 的输出; non_contiguous=True 分支用 .unflatten(-1, (num_heads, head_dim)) 复现 Gemma3 q_norm/k_norm 从头切片的非连续视图。用例分为 test_gemma3_rms_norm_2d/3d/4d 与 test_gemma3_rms_norm_3d_unflatten/4d_unflatten 五组。
4. 按 review 收敛: commit 序列显示作者先加实现与测试 (f9c6db2), 再加完整测试与 benchmark (ad1801a), 随后删除 bench_gemma3_rmsnorm.py (e8df93c), 最后按 CaoE 的意见缩小测试参数矩阵 (002e307)。最终 PR 不含 benchmark 脚本, PR body 的 Speed Tests 也标注为 N/A。
5. 精度验证: PR body 给出 test/registered/kernels/test_fused_op_dispatch.py 56 passed, test/manual/layers/test_layernorm.py 5 passed / 3 skipped / 194 subtests passed; 作者在回复 mingfeima 时确认改动前后 pytest 结果对齐、无失败项。

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 Gemma3RMSNorm.forward_xpu, Gemma3RMSNorm.forward_cuda, Gemma3RMSNorm.forward_native): 核心源码改动: 为 Gemma3RMSNorm 新增 forward_xpu, 定义 XPU 上 2D/3D/4D 输入是否走 gemma_rmsnorm/gemma_fused_add_rmsnorm 以及何时回落 forward_native, 是本次功能的主体。
- test/manual/layers/test_layernorm.py (模块 归一化测试; 类别 test; 类型 test-coverage; 符号 TestGemma3RMSNorm, TestGemma3RMSNorm.setUpClass, TestGemma3RMSNorm._run_gemma3_rms_norm_test, TestGemma3RMSNorm.test_gemma3_rms_norm_2d): 新增 TestGemma3RMSNorm (+173/-1), 以 forward_native 为基准覆盖 2D/3D/4D、residual 组合与非连续 unflatten 视图, 是本次 XPU 实现的唯一验证手段; 同时该文件位于 test/manual, 不参与 CI。

关键符号: Gemma3RMSNorm.forward_xpu, Gemma3RMSNorm.forward_native, Gemma3RMSNorm.forward_cuda, TestGemma3RMSNorm.setUpClass, TestGemma3RMSNorm._run_gemma3_rms_norm_test, TestGemma3RMSNorm.test_gemma3_rms_norm_2d, TestGemma3RMSNorm.test_gemma3_rms_norm_4d_unflatten

关键源码片段

python/sglang/srt/layers/layernorm.py

核心源码改动: 为 Gemma3RMSNorm 新增 forward_xpu, 定义 XPU 上 2D/3D/4D 输入是否走 gemma_rmsnorm/gemma_fused_add_rmsnorm 以及何时回落 forward_native, 是本次功能的主体。

Gemma3RMSNorm 的平台分发与新增 XPU 分支

```

class Gemma3RMSNorm(BaseFusedOp):
    # __init__ 中保存 self.weight / self.eps, 并复用 self._norm 做 FP32 归一化

    def forward_native(self, x, residual: Optional[torch.Tensor] = None):
        # 纯 PyTorch 参考实现, 所有平台的兜底路径, 也是测试里的对照基线
        if residual is not None:
            residual = x + residual # 非原地: 新建张量, 不会污染调用方入参
            x = residual
        output = self._norm(x.float())
        # Llama 是 x.to(float16) * w, Gemma3 是 (x * w).to(float16)
        # 见 https://github.com/huggingface/transformers/pull/29402
        output = output * (1.0 + self.weight.float())
        output = output.type_as(x)
        return output if residual is None else (output, residual)

    def forward_cuda(self, x, residual: Optional[torch.Tensor] = None):
        if residual is not None:
            # decoder 的 residual 是 token-major 且连续的; 融合 kernel 原地更新两个张量:
            # x 变成归一化输出, residual 变成 x + residual 供下一层使用
            gemma_fused_add_rmsnorm(x, residual, self.weight.data, self.eps)
            return x, residual
        if x.dim() == 2:
            return gemma_rmsnorm(x, self.weight.data, self.eps)
        # 注意: CUDA 侧非 2D 输入仍回落到 native, 这是与 XPU 分支的关键差异
        return self.forward_native(x)

    def forward_xpu(self, x, residual: Optional[torch.Tensor] = None):
        if residual is not None and x.dim() == 2:
            # 与 CUDA 融合分支语义一致: 原地写回 x 与 residual, 避免额外分配
            gemma_fused_add_rmsnorm(x, residual, self.weight.data, self.eps)
            return x, residual
        # XPU kernel 内部会自行拍平前导维度, 因此 2D/3D/4D 都能直接进 kernel,
        # 不需要 Python 侧 reshape (对应 sgl-kernel-xpu#367 放宽的 4D 形状支持)
        elif residual is None and x.dim() in (2, 3, 4):
            return gemma_rmsnorm(x, self.weight.data, self.eps)
        # 其余组合 (带 residual 的非 2D、维度不在 2/3/4 内) 回落到纯 PyTorch, 保证正确性
        return self.forward_native(x, residual)

    def forward_musa(self, x, residual: Optional[torch.Tensor] = None):
        # sgl_kernel 的 gemma norm 算子在 MUSA 上有构建, 沿用 CUDA 路径
        return self.forward_cuda(x, residual)

```

XPU 分支相对 CUDA 分支多放开 3D/4D, 代价是 **residual + 非 2D** 组合失去原地融合能力而回落 native, 这一取舍在代码注释与测试维度选择中都能看到痕迹。

test/manual/layers/test_layernorm.py

新增 TestGemma3RMSNorm (+173/-1), 以 forward_native 为基准覆盖 2D/3D/4D、residual 组合与非连续 unflatten 视图, 是本次 XPU 实现的唯一验证手段; 同时该文件位于

test/manual, 不参与 CI。

TestGemma3RMSNorm: 形状矩阵与真实调用形态复现

```
class TestGemma3RMSNorm(CustomTestCase):
    """覆盖 2D/3D/4D 输入, 包括 Gemma3 的 q_norm/k_norm 实际喂进来的
    非连续 ("unflatten") 形状: 用 .split() + .unflatten() 从更宽的 qkv
    张量里切出的头切片, 前导维度无法拍平成 2D。"""

    ADD_RESIDUAL = [False, True]
    SEEDS = [0]

    # 2D 场景的 (batch_size, hidden_size, dtype) 组合
    SHAPE_DTYPE_2D = [
        (batch_size, hidden_size, torch.float16)
        for batch_size in [1, 19]
        for hidden_size in [1152, 2560, 3840, 5376]
    ] + [
        (19, 1024, torch.bfloat16),
        (19, 1024, torch.float32),
        (2, 32768, torch.float16),
    ]

    BATCH_SIZES_3D = [1, 4]
    SEQ_LENS_3D = [1, 74]
    HIDDEN_SIZES_3D = [512, 1024]
    DTYPES_3D = [torch.float16]

    NUM_TOKENS_4D = [1, 7]
    NUM_HEADS_4D = [1, 4, 8]
    HEAD_DIMS_4D = [64, 128]
    DTYPES_4D = [torch.float16]

    @classmethod
    def setUpClass(cls):
        # CUDA / XPU 任一可用即运行, default device 随平台切换,
        # 让同一份用例同时看守 CUDA 与 XPU 两条实现路径
        if not (torch.cuda.is_available() or torch.xpu.is_available()):
            raise unittest.SkipTest("Neither CUDA nor XPU is available")
        device = "cuda" if torch.cuda.is_available() else "xpu"
        torch.set_default_device(device)

    def _run_gemma3_rms_norm_test(
        self, shape, add_residual, dtype, seed, non_contiguous=False
    ):
        torch.manual_seed(seed)
        hidden_size = shape[-1]
        layer = Gemma3RMSNorm(hidden_size).to(dtype=dtype)
        layer.weight.data.normal_(mean=0.0, std=0.1)
        scale = 1 / (2 * hidden_size)
```

```

if non_contiguous:
    # 复现真实调用形态: 从 total_heads 个头的宽张量里切出 num_heads 个头,
    # 再 unflatten 成 (... , num_heads, head_dim), 结果是不可拍平的非连续视图
    *lead, num_heads, head_dim = shape
    total_heads = num_heads + 3
    full = torch.randn(*lead, total_heads * head_dim, dtype=dtype) * scale
    x = full[..., : num_heads * head_dim].unflatten(-1, (num_heads, head_dim))
else:
    x = torch.randn(*shape, dtype=dtype) * scale

residual = torch.randn_like(x) * scale if add_residual else None

with torch.inference_mode():
    ref_out = layer.forward_native(x, residual) # 纯 PyTorch 参考实现
    out = layer(x, residual) # 平台分发后的实现 (CUDA 或 XPU)

# 融合分支会原地改写 x / residual, 因此必须先算 ref_out 再做 kernel 调用
if add_residual:
    self.assertTrue(torch.allclose(out[0], ref_out[0], atol=1e-2, rtol=1e-2))
    self.assertTrue(torch.allclose(out[1], ref_out[1], atol=1e-2, rtol=1e-2))
else:
    self.assertTrue(torch.allclose(out, ref_out, atol=1e-2, rtol=1e-2))

```

注意: 3D/4D 与 `ADD_RESIDUAL=True` 的组合在 XPU 上会回落 `forward_native`, 这些 `subTest` 实际并未覆盖 XPU 融合分支。

评论区精华

本轮 review 共 3 条代码评论文本, 讨论集中在测试范围与测试落点, 未对 `forward_xpu` 实现本身提出反对意见:

1. 测试规模: CaoE 在 `TestGemma3RMSNorm` 类定义处提问 "Can we reduce the size and choose only the smallest set to shorten the test time ?", 作者以 [UPDATE] smaller test cases for gemma3 提交响应, 缩小了 2D/3D/4D 的参数矩阵 (如 `NUM_TOKENS`、`HIDDEN_SIZES`、`HEAD_DIMS` 取值变少)。
 2. 测试是否进 CI: mingfeima 在导入行处质疑 "this file is manual test, which will not be launched in CI. Do you check locally?", 直指 XPU 分支缺少 CI 门禁; 作者回复 "Yes. The pytest results before and after are aligned and there are no failed items.", 以本地验证作为依据, CaoE 与 mingfeima 随后分别 APPROVED。
 3. 依赖前置: issue 评论中 CaoE 记录 `sgl-project/sgl-kernel-xpu#367` 依赖关系, 作者确认依赖 PR 已合并后才具备 4D 形状能力。
- 测试参数矩阵过大, 需要缩小并保留最小集合以缩短测试时间 (testing): 作者以 commit [UPDATE] smaller test cases for gemma3 (002e307) 缩小参数矩阵, reviewer 未再提出异议; 但缩小后 `HIDDEN_SIZES_3D/NUM_HEADS_4D` 上方的 "other dim == 1" 注释未同步清理。
 - 测试文件位于 `test/manual`, 不会在 CI 中执行 (testing): reviewer 接受本地验证并 APPROVED, 但 XPU 分支仍无 CI 门禁; 若要长期守护需按 #32093 的做法迁入

registered/XPU nightly。

- 对 `sgl-kernel-xpu#367` 的跨仓库依赖 (other): 依赖已合并, 但 python 侧的 `pyproject_xpu` 未显式收紧 `sgl-kernel-xpu` 最低版本, 运行环境版本不一致时可能出错。

风险与影响

- 风险:

1. 测试不进CI (最主要风险): mingfeima明确指出`test/manual/layers/test_layernorm.py` 不会被 CI 拉起, 因此 `forward_xpu` 的正确性只靠作者本地验证。后续若 XPU kernel 行为变化或 `BaseFusedOp` 分发逻辑调整, CI 无法发现回归; 仓库中已有把设备相关测试接入 XPU nightly 的先例 (如 #32093 把 `dumper` 测试接入 XPU nightly)。
2. 对 `sgl-kernel-xpu` 版本存在隐式依赖: `forward_xpu` 在 3D/4D 上直接调用 `gemma_rmsnorm`, 前提是所装 wheel 的 kernel 支持拍平前导维度 (对应 `sgl-kernel-xpu#367`)。代码里没有能力探测或版本下限约束, 若运行环境装配旧版 kernel, 可能出现运行期报错或行为不一致, 而 `python/pyproject_xpu.toml` 侧并未同步收紧依赖。
3. 平台语义分叉: `forward_cuda` 仍只对 `x.dim() == 2` 走 kernel, XPU 放宽到 (2, 3, 4), 同一模型在 CUDA 与 XPU 上走不同代码路径, 精度或性能差异只会在单一平台显现, 跨平台回归对比更难定位。
4. residual + 非 2D 在 XPU 上不走上融合: CUDA 的 residual 分支不区分维度都会调用 `gemma_fused_add_rmsnorm`, 而 XPU 只对 `dim == 2` 融合, 3D/4D 带 residual 会回落 `forward_native` 并额外分配张量, 存在性能回退与语义 (原地更新 vs 新建张量) 差异。
5. 测试注释与参数已不同步: 缩小用例时移除了 `hidden_size=1` 等取值, 但 `HIDDEN_SIZES_3D` 上方仍保留 "`hidden_size=1` exercises the ... 'other dim == 1' shape" 的注释, `NUM_HEADS_4D/HEAD_DIMS_4D` 的同类注释也部分失效, 容易误导后续维护者; 相应边界形状的覆盖实际被削弱。
6. 测试覆盖盲区: 3D/4D 与 `ADD_RESIDUAL=True` 的组合在 XPU 上同样走 `forward_native`, 即这些 `subTest` 无法验证 XPU 融合分支; XPU kernel 的有效覆盖实际只落在 `residual is None` 的 2D/3D/4D 上。- 影响: 对 XPU 用户: Gemma3 系列模型 (尤其 `q_norm/k_norm` 的 3D/4D 非连续头切片) 在 XPU 上从 Python 回落路径切换到 `sgl-kernel-xpu` 融合算子, 减少 Python 侧 `reshape` 与逐元素计算开销, 是 Gemma3 在 XPU 上可用性与性能的补齐。对 CUDA/NPU 等其他平台: 完全无影响, 属于新增平台方法, 由 `BaseFusedOp` 的设备分发按平台选中。对测试体系: 新增的 `TestGemma3RMSNORM` 虽然在 `test/manual` 下不参与 CI, 但其 `setUpClass` 同时支持 CUDA/XPU, 为后续迁移到 `registered/XPU nightly` 提供了可直接复用的用例与 `forward_native` 对照基准。对团队协作: 形成对 `sgl-kernel-xpu` 版本能力的又一次隐式依赖, 需要 XPU 维护者在 kernel 升级 / 回退时同步关注。- 风险标记: 测试不在 CI 覆盖, 依赖外部仓库 kernel 版本, 跨平台实现语义分叉, 测试注释与参数不同步, 非 2D 带残差路径性能回退, 测试覆盖存在盲区

关联脉络

- PR #32093 [XPU] Adapt device agnostic API usage: 同为 XPU 设备无关化改造，并示范了把 test/manual 下的设备相关测试接入 XPU nightly 的做法，正是本 PR 测试落点问题的参照方案。
- PR #37394 [xpu] install xpu-kernel by released wheel: XPU kernel 改为安装官方发布 wheel，本 PR 的 forward_xpu 直接依赖该 wheel 中 gemma_rmsnorm 对 4D 形状的支持，二者共同决定 XPU 运行环境能力边界。
- PR #32798 DFLASH support added for XPU: 同样是 XPU 平台能力补齐并配套端到端 / 单元测试的 PR，体现 XPU 功能线从算子到模型的持续扩展脉络。