PR #6798 完整报告

PaddlePaddle/FastDeploy

[XPU] Split the block_attn operator into smaller operators

合并时间：2026-04-16 14:28

原文链接：http://prhub.com.cn/PaddlePaddle/FastDeploy/pull/6798

执行摘要

本 PR 将 XPU 平台的 block_attn 算子拆分为可独立控制的 spliced 版本，通过环境变量 encoder_splice 和 decoder_splice 启用，旨在提升优化灵活性和调试便利性。变更涉及 C++ 算子实现、Python 调用层和测试配套，引入新功能的同时保持向后兼容，但需关注命名混淆、兼容性风险和依赖稳定性等潜在问题。

功能与动机

为什么做：PR body 中动机简写为 'splice block_attn fused op'，具体背景是通过拆分融合算子为更细粒度的操作（如 split_rope 和 neox_cache_kv），以优化 XPU 上注意力计算的性能和提供更灵活的调试手段。review 讨论补充说明，splice 模式允许解耦 QKV 分离、RoPE 嵌入和 KV cache 存储步骤，便于独立优化和问题定位。

实现拆解

实现过程可拆解为以下步骤：

1. 新增 spliced 算子核心实现：在 custom_ops/xpu_ops/src/ops/block_attn_spliced.cc 中新增 SplitEmbeddingKVCacheBlockAttn 函数，核心逻辑如下： cpp // 简化示例：根据环境变量选择执行路径 if (FLAGS_encoder_splice) { // 执行 encoder 的 spliced 操作：拆分 QKV、应用 RoPE、写入 KV cache split_kvcache_encoder(...); } else { // 回退到原有的 fused 路径 block_attn_fused(...); } // 类似处理 decoder 路径 该文件新增约 1900 行代码，引入 FLAGS_encoder_splice 和 FLAGS_decoder_splice 环境变量控制开关，默认不启用以确保无缝过渡。
2. 重命名和 API 兼容性处理：将原有 BlockAttn 重命名为 BlockAttnFused（block_attn.cc 和 pybind/pybind.cc），并新增 slot_mapping_enc 和 slot_mapping_dec 参数，但在 fused 版本中未使用以避免破坏现有调用。同时，移除 HALF_HEAD_DIM 位置编码类型支持，需验证模型兼容性。
3. 测试配套增强：新增 custom_ops/xpu_ops/test/test_block_attn.py 测试文件，包含 run_block_attn 等函数，通过对比 fused 和 spliced 版本输出验证数值一致性，覆盖 prefix cache、量化等场景。删除 tests/xpu_ci/4cards_cases/test_mtp_cudagraph.py，因 MTP 模式暂不支持 spliced 路径。
4. Python 层适配：修改 fastdeploy/model_executor/xpu_pre_and_post_process.py、forward_meta.py 等文件，更新算子调用以传递新参数，确保上层代码无缝集成。

5. 依赖和配置调整：在 custom_ops/xpu_ops/download_dependencies.sh 中将 xvllm 版本改为 "latest"，但 review 中建议使用固定版本以避免构建不稳定。

关键源码片段

custom_ops/xpu_ops/test/test_block_attn.py

新增测试文件，包含关键测试函数如 run_block_attn 和 run_prefix_cache_block_attn，用于验证 spliced 与 fused 版本的数值一致性，是质量保障的核心。

```
def print_all_not_equal_elements_info(k, x, y):
    """辅助函数：打印张量 x 和 y 中不相等元素的详细信息，用于调试和测试验证。"""
    x_flatten = x.flatten()
    y_flatten = y.flatten()
    index = paddle.nonzero(x_flatten != y_flatten) # 找到不相等元素的索引
    x_not_equal = x_flatten[index]
    y_not_equal = y_flatten[index]
    print(f"reference not equal element of {k}: {x_not_equal}")
    print(f"calculated result not equal element of {k}: {y_not_equal}")
    xy_diff = x - y # 计算差值
    xy_mean_diff = paddle.mean(xy_diff) # 平均误差
    xy_max_abs_diff = paddle.max(paddle.abs(xy_diff)) # 最大绝对误差
    xy_min_abs_diff = paddle.min(paddle.abs(xy_diff)) # 最小绝对误差
    print(f"{k} mean diff: {xy_mean_diff}, max abs diff: {xy_max_abs_diff}, min abs diff: {xy_min_
    abs_diff}")
```

custom_ops/xpu_ops/src/ops/get_infer_param.cc

修改关键函数 get_infer_param，新增 lod_to_slot_mapping 函数和 num_speculative_tokens 参数，为 spliced 模式提供 slot mapping 计算支持。

```
void lod_to_slot_mapping(api::Context* xpu_ctx,
                         paddle::Place place,
                         const std::vector<int32_t>& block_table,
                         const std::vector<int32_t>& kv_seq_lod,
                         const std::vector<int32_t>& start_tokens,
                         const std::vector<int32_t>& real_batch,
                         int32_t* slot_mapping,
                         int32_t token_num,
                         int32_t block_size,
                         int32_t batch_size,
                         int32_t max_num_blocks_per_seq,
                         int32_t num_speculative_tokens) {
    if (token_num <= 0) {
        return; // 无令牌时直接返回
    }
    std::vector<int32_t> slot_mapping_vec(token_num, -1); // 初始化 slot mapping 为 -1
    int32_t idx = 0;
    // 遍历每个批次，计算每个令牌的目标缓存位置
    for (auto batch_ = 0; batch_ < batch_size; batch_++) {
        int32_t seq_len = kv_seq_lod[batch_ + 1] - kv_seq_lod[batch_]; // 当前批次序列长度
```

```
            int32_t seq_start = start_tokens[batch_]; // 序列起始位置
            int32_t dst_batch_id = real_batch[batch_]; // 目标批次 ID
            for (auto seq_ = seq_start; seq_ < seq_start + seq_len; seq_++) {
                int32_t table_id = seq_ / block_size; // 计算块表索引
                int32_t block_id = block_table[dst_batch_id * max_num_blocks_per_seq + table_id]; //
                获取块 ID
                int32_t seq_offset = seq_ % block_size; // 块内偏移
                int32_t dst_token_offset = block_id * block_size + seq_offset; // 目标令牌偏移
                slot_mapping_vec[idx] = dst_token_offset; // 存储映射
                idx++;
            }
        }
        // 将计算结果复制到设备内存
        int ret = api::do_host2device(xpu_ctx, slot_mapping_vec.data(), slot_mapping, token_num *
        sizeof(int32_t));
        PD_CHECK(ret == api::SUCCESS, "api::do_host2device failed."); // 检查传输是否成功
    }
```

评论区精华

review 讨论聚焦于以下交锋：

- 命名设计争议：block_attn 现指向 spliced 版本而 block_attn_fused 为旧版本，可能让用户混淆。PaddlePaddle-bot 评论：" 通常 'fused' 表示优化后的实现，但这里反而变成了旧版本。" 建议使用更清晰的命名或文档说明，但未达成明确结论。
- 正确性 bug：在 block_attn_spliced.cc 中发现 v_cache_zp 参数错误使用 k_pc_zero，PaddlePaddle-bot 标记为 ❓ Bug：" 当前代码会导致 value cache 的 INT8 反量化使用错误的 zero point 值。" 需修复以确保推理精度。
- 兼容性风险：移除 HALF_HEAD_DIM 支持引发疑虑，PaddlePaddle-bot 评论：" 是否有现有模型依赖 HALF_HEAD_DIM 模式？移除此支持是否经过充分测试？" 需团队进一步验证。

风险与影响

技术风险：

1. 兼容性：移除 HALF_HEAD_DIM 支持可能影响依赖此模式的模型，需在部署前测试。
2. 性能回归：spliced 路径在 MTP 等复杂场景未充分验证，可能引入性能下降或错误。
3. 构建不稳定：依赖版本 "latest" 可能导致 CI 环境不一致，影响问题排查和发布流程。
4. 正确性：v_cache_zp bug 若未修复，直接损害 INT8 量化推理的准确性。

影响范围：

- 用户：需通过环境变量显式启用 spliced 功能，默认行为不变，对现有用户无感知影响。
- 系统：新增算子实现增加代码库复杂度，但通过环境变量控制，对核心路径无侵入；长期需维护两套实现可能增加维护负担。
- 团队：测试覆盖加强有助于保证质量，但需关注跨模块调用正确性，如 Python 层参数传递。

关联脉络

从历史 PR 分析，本 PR 与近期 XPU 平台优化（如 PR 6947 新增投机解码算子）一脉相承，显示团队在 XPU 算子层持续投入，以提升硬件利用率和性能。同时，与 PR 7237 等优化配置的 PR 呼应，反映系统级性能调优趋势。通过环境变量控制新功能的模式，借鉴了渐进式发布策略，有助于降低部署风险。