

PR #45368 完整报告

vllm-project/vllm

[xpu][lora]: Align LoRA implementation with Punica GPU: fix `_apply_expand` rank mismatch, `add_inputs` hardcode, and MoE EP

合并时间: 2026-07-01 22:14

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/45368>

执行摘要

- 一句话: 修复 XPU LoRA 多项对齐 bug
- 推荐动作: 此 PR 展示了如何将 XPU 后端 LoRA 算子与 CUDA 后端对齐并利用 PyTorch 自定义算子机制, 值得 LoRA 模块维护者精读。`_apply_expand` 的 rank 截断逻辑和 `add_inputs` 透传是常见兼容性模式, 可供其他后端参考。

功能与动机

在 #42290 的 MoE LoRA 单次 Triton 内核和 #41184 的 FusedMoE 重构之后, XPU 后端的 LoRA 实现出现了一组兼容性问题: `_apply_expand` 在进行张量并行 all-gather 后 `x` 的 `hidden_size` 大于实际 `lora_b` 的 rank, 导致 XPU C++ 内核因输入形状不匹配而崩溃; `add_lora_linear` 中 `add_inputs` 参数被硬编码忽略, 导致双流 LoRA 行为错误; MoE LoRA EP 路径存在对齐问题。PR 旨在修复这些回归, 使 XPU LoRA 行为与 GPU 一致。

实现拆解

1. 修复 `_apply_expand` 的 rank 截断: 在 `vllm/lora/punica_wrapper/punica_xpu.py` 的 `_apply_expand` 中, 调用 `bgmv_expand_slice` 前检查 `x.size(1) != w_t_all.size(-1)`, 若不符则截断 `x[:, :rank].contiguous()` 以避免 XPU 内核输入形状不匹配。
2. 修复 `add_inputs` 硬编码: 同样在 `punica_xpu.py` 的 `add_lora_linear` 中, 将 `add_inputs=True` 替换为从 `kwargs` 弹出的 `add_inputs` 值, 并透传给下游 `add_expand` 和 `add_shrink`, 同时新增断言禁止调用方传入 `buffer` 以减少开销。
3. 修复 `add_lora_logits` 和 `add_expand` 的返回值赋值: 修正 `add_lora_logits` 和 `add_expand` 中 `y.view_as(y_org)` 未赋值回 `y` 的问题。
4. Moe LoRA EP 路径修复: 在 `moe_lora_align_block_size` 中修复 EP 场景下的对齐逻辑, 确保 `sorted_token_ids` 等张量正确重排。
5. 底层算子注册为 Torch 自定义算子: 在 `vllm/lora/ops/xpu_ops/lora_ops.py` 中, 将 `bgmv_shrink`、`bgmv_expand`、`bgmv_expand_slice` 的实现函数重命名为 `_bgmv_*_impl`, 并通过 `direct_register_custom_op` 注册为 `vllm.xpu_bgmv_*` 自定义算子, 同时提供 `fake_impl` 以支持 `torch.compile` 和 CUDA Graph。上层通过 `torch.ops.vllm.xpu_bgmv_*` 调用。

6. CI 配置更新：在 `.buildkite/intel_jobs/lora_intel.yaml` 的 LoRA Punica Kernels 测试环节增加两个 `--deselect` 排除项，规避已知失败的测试用例。

关键文件：

- `vllm/lora/punica_wrapper/punica_xpu.py`（模块 LoRA；类别 source；类型 core-logic；符号 `_apply_expand`, `add_expand`, `add_lora_linear`, `add_lora_logits`）：核心修改文件，修复了 `_apply_expand` rank 不匹配、`add_inputs` 硬编码、`buffer` 断言和返回值赋值等关键 bug，是对齐 GPU 实现的主要改动。
- `vllm/lora/ops/xpu_ops/lora_ops.py`（模块 算子层；类别 infra；类型 infrastructure；符号 `bgmv_shrink`, `_bgmv_shrink_impl`, `bgmv_expand`, `_bgmv_expand_impl`）：将底层 `bgmv` 算子从直接调用 XPU C++ kernel 改为通过 PyTorch 自定义算子注册，支持 `torch.compile` 和 CUDA Graph，并提供 `fake_impl` 用于图捕获。
- `.buildkite/intel_jobs/lora_intel.yaml`（模块 CI；类别 config；类型 configuration）：CI 配置增加两个测试排除项，避免因已知 XPU 特定失败导致流水线卡断。

关键符号：`_apply_expand`, `add_lora_linear`, `add_lora_logits`, `moe_lora_align_block_size`, `bgmv_shrink`, `bgmv_expand`, `bgmv_expand_slice`

关键源码片段

`vllm/lora/punica_wrapper/punica_xpu.py`

核心修改文件，修复了 `_apply_expand` rank 不匹配、`add_inputs` 硬编码、`buffer` 断言和返回值赋值等关键 bug，是对齐 GPU 实现的主要改动。

`vllm/lora/punica_wrapper/punica_xpu.py`（关键片段）

```
def _apply_expand(
    self,
    y: torch.Tensor,
    x: torch.Tensor,
    w_t_all: torch.Tensor,
    y_offset: int,
    y_slice_size: int,
    add_inputs: bool,
):
    token_lora_indices = self._get_token_lora_indices(x)
    # 在 TP all-gather 后，x.size(1) 可能等于 max_lora_rank * tp_size,
    # 而 lora_b 仅使用 max_lora_rank 元素。XPU C++ 内核要求
    # inputs.size(1) == lora_b.size(-1)，因此截断并拷贝为连续张量。
    rank = w_t_all.size(-1)
    if x.size(1) != rank:
        x = x[:, :rank].contiguous()
    bgmv_expand_slice(
        x, w_t_all, y, token_lora_indices, y_offset, y_slice_size, add_inputs
    )

def add_lora_linear(self, ...):
```

```

# 新增断言禁止外部传入 buffer，强制由本方法创建以减少开销
assert buffer is None, (
    "To minimize overhead, the buffer should be created by "
    ".add_lora_linear() instead of being passed in."
)
r = lora_b_stacked[0].size(-1)
buffer = torch.zeros(
    (len(output_slices), x.size(0), r),
    dtype=x.dtype,
    device=x.device,
)
# 从 kwargs 中弹出 add_inputs，不再硬编码为 True
add_inputs = kwargs.pop("add_inputs", True)
self.add_shrink(buffer, x, lora_a_stacked, scale)
self.add_expand(
    y, buffer, lora_b_stacked, output_slices,
    add_inputs=add_inputs, **kwargs,
)

```

vllm/lora/ops/xpu_ops/lora_ops.py

将底层 bgmv 算子从直接调用 XPU C++ kernel 改为通过 PyTorch 自定义算子注册，支持 torch.compile 和 CUDA Graph，并提供 fake_impl 用于图捕获。

```

# vllm/lora/ops/xpu_ops/lora_ops.py ( 关键片段 )

# 注册自定义算子，使 torch.compile / cudagraph 能够识别
# mutates_args 指明该算子会修改 output_tensor，fa
# fake_impl 用于图捕获阶段，直接返回 None 避免触发实际 kernel。
direct_register_custom_op(
    op_name="xpu_bgmv_shrink",
    op_func=_bgmv_shrink_impl,
    mutates_args=["output_tensor"],
    fake_impl=_bgmv_shrink_fake,
)
direct_register_custom_op(
    op_name="xpu_bgmv_expand",
    op_func=_bgmv_expand_impl,
    mutates_args=["output_tensor"],
    fake_impl=_bgmv_expand_fake,
)
direct_register_custom_op(
    op_name="xpu_bgmv_expand_slice",
    op_func=_bgmv_expand_slice_impl,
    mutates_args=["output_tensor"],
    fake_impl=_bgmv_expand_slice_fake,
)

# 上层 API 通过 torch.ops.vllm.xpu_bgmv_* 调用，保持接口兼容
def bgmv_shrink(inputs, lora_a_weights, output_tensor, lora_indices_tensor, scaling=1.0):

```

```
torch.ops.vllm.xpu_bgmv_shrink(
    inputs, lora_a_weights, output_tensor, lora_indices_tensor, scaling
)

def bgmv_expand(inputs, lora_b_weights, output_tensor, lora_indices_tensor, add_inputs=True):
    torch.ops.vllm.xpu_bgmv_expand(
        inputs, lora_b_weights, output_tensor, lora_indices_tensor, add_inputs
    )

def bgmv_expand_slice(...):
    torch.ops.vllm.xpu_bgmv_expand_slice(...)
```

评论区精华

无公开审核评论。两位审核人 [jeejeelee](#) 和 [jikunshang](#) 均直接批准，表明改动清晰且低争议。

- 暂无高价值评论线程

风险与影响

- 风险：
 1. 兼容性风险：自定义算子注册（direct_register_custom_op）改变了算子调用路径，若未来 torch.ops 接口变动需要同步调整。但当前模式与 CUDA 后端保持一致，风险可控。
 2. 性能开销：_apply_expand 中新增的 `x = x[:, :rank].contiguous()` 会引入一次显存拷贝，仅在 TP>1 且非全分片时触发，影响可接受。
 3. 回归范围：改动集中在 XPU LoRA 路径，不影响 GPU/CPU 后端。CI 中已补充测试排除，但未新增独立测试覆盖，存在遗漏边界情况的风险。- 影响：直接影响 Intel GPU (XPU) 上使用 LoRA 的用户，修复了张量并行和双流 LoRA 的功能正确性。对非 XPU 用户无影响。团队需要关注后续 LoRA 重构可能对 XPU 产生的同步需求。- 风险标记：兼容性风险（自定义算子注册），性能开销（contiguous 拷贝），缺少新增测试覆盖

关联脉络

- PR #42290 [LoRA] Add one shot triton kernel For MoE LoRA: PR 明确指出此 PR 修复了 #42290 引入的回归问题。
- PR #41184 [MoE Refactor] FusedMoE/MoERunner inversion refactor: PR 明确指出此 PR 修复了 #41184 引入的回归问题，涉及 MoE 权重路径变化导致的兼容性问题。