PR #46361 完整报告

vllm-project/vllm

[INC][ARK] Direct Register Custom Op for ARK

合并时间：2026-07-06 13:45

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46361

执行摘要

- 一句话：直接注册 ARK 自定义 Op 以支持 torch compile
- 推荐动作：
  - 建议量化相关开发者精读 inc_ark_ops.py，了解 direct_register_custom_op 和 fake impl 编写方式。
  - 后续 CI 中增加 torch compile 模式的端到端测试覆盖 ARK 算子。
  - 注册时机可考虑惰性化以提高鲁棒性，但对当前场景已足够。

功能与动机

随着 vLLM 对 torch compile 支持的完善，自定义后端算子需以 torch.ops 形式注册才能被 torch compile 识别。ARK 量化（基于 auto_round_kernel）是 CPU/XPU 上的重要路径，原先直接调用库函数会导致编译图中断。注册后可实现 4%-6% 的性能提升（TPOT/ITL）。

实现拆解

1. 提取 ARK 状态检查到独立模块：新建 inc_ark_ops.py，将原先位于 inc_wna16_linear.py 的 get_ark_state() 函数迁移至此，并使用 @lru_cache 缓存检测结果。
2. 注册自定义算子：在 inc_ark_ops.py 中定义真实实现 _inc_ark_woq_linear_impl（委托给 auto_round_kernel.woqgemm_linear）和 fake 实现 _inc_ark_woq_linear_fake（返回空张量供 torch compile 元数据推断），通过 direct_register_custom_op 注册为 torch.ops.vllm.inc_ark_woq_linear，并在模块加载时调用 ark_ops.register_ops_once()。
3. 修改调用路径使用注册算子：在 inc_wna16_linear.py 的 INCARKLinearMethod 中，将 apply_weights 从直接调用 layer.ark_linear.forward(x) 改为 torch.ops.vllm.inc_ark_woq_linear.default(...)，并调整 process_weights_after_loading 以保留 qweight 等属性供算子使用。
4. 更新导入点和降级逻辑：在 inc_wna16_scheme.py 的 get_linear_method 中，将 get_ark_state 的导入源从 inc_wna16_linear 改为 inc_ark_ops。
5. 测试与文档配套：更新 test_auto_round.py 中的 monkeypatch 目标路径并移除 enforce_eager=True 参数；更新 doc/inc.md 移除 --enforce-eager 说明；升级 requirements/xpu.txt 中 auto_round_lib 最低版本至 0.14.0。

关键文件：

- vllm/model_executor/layers/quantization/inc/schemes/inc_ark_ops.py（模块 量化核心；类别 source；类型 data-contract；符号 get_ark_state, _inc_ark_woq_linear_impl, _inc_ark_woq_linear_fake, ark_ops）：新文件，核心变更：抽取 ARK 状态检查并注册自定义算子，支持 torch compile。
- vllm/model_executor/layers/quantization/inc/schemes/inc_wna16_linear.py（模块 量化核心；类别 source；类型 data-contract；符号 get_ark_state）：核心修改文件：移除 get_ark_state 重复定义，调整 INCARKLinearMethod 使用注册算子。
- vllm/model_executor/layers/quantization/inc/schemes/inc_wna16_scheme.py（模块 量化核心；类别 source；类型 data-contract）：导入路径调整：get_ark_state 从 inc_wna16_linear 改为 inc_ark_ops。
- tests/quantization/test_auto_round.py（模块 测试；类别 test；类型 test-coverage）：更新单元测试以匹配重构后的导入路径，并移除 enforce_eager=True。
- docs/features/quantization/inc.md（模块 文档；类别 docs；类型 documentation）：文档更新：移除 --enforce-eager 的说明。
- requirements/xpu.txt（模块 依赖管理；类别 infra；类型 documentation）：版本依赖升级 auto_round_lib>=0.14.0 以支持新算子接口。

关键符号：get_ark_state, _inc_ark_woq_linear_impl, _inc_ark_woq_linear_fake, ark_ops.register_ops_once, INCARKLinearMethod.apply_weights, INCARKLinearMethod.process_weights_after_loading, INCWna16Scheme.get_linear_method

关键源码片段

vllm/model_executor/layers/quantization/inc/schemes/inc_ark_ops.py

新文件，核心变更：抽取 ARK 状态检查并注册自定义算子，支持 torch compile。

```
from functools import lru_cache
from typing import Any

import torch

from vllm.logger import init_logger
from vllm.platforms import current_platform
from vllm.utils.torch_utils import direct_register_custom_op

logger = init_logger(__name__)
_OPS_REGISTERED = False

@lru_cache(maxsize=1)
def get_ark_state() -> tuple[bool, str | None, Any | None, Any | None]:
    '''返回 ARK 可用性标志、错误信息、缓存模块和 QuantLinear 类'''
    try:
        import auto_round_kernel as ark
        from auto_round_kernel.qlinear import QuantLinear
        logger.info('Successfully imported auto_round_kernel.')
```

```
    except ImportError as error:
        return False, str(error), None, None
    # 检测至少一个后端库存在
    if getattr(ark, 'cpu_lib', None) is None and getattr(ark, 'xpu_lib', None) is None:
        return (False, 'No ARK backend library is available.', None, None)
    logger.info('Successfully loaded auto_round_kernel backend library.')
    return True, None, ark, QuantLinear

def _inc_ark_woq_linear_impl(
    x: torch.Tensor, qweight: torch.Tensor, bias: torch.Tensor | None,
    out_features: int, in_features: int, group_size: int,
    compute_type: str, weight_type: str, scale_type: str, asym: bool,
) -> torch.Tensor:
    '''真实实现：委托给 auto_round_kernel.woqgemm_linear'''
    ark = get_ark_state()[2]
    assert ark is not None
    return ark.woqgemm_linear(x, qweight, bias, out_features, in_features,
                              group_size, compute_type, weight_type, scale_type, asym)

def _inc_ark_woq_linear_fake(
    x: torch.Tensor, qweight: torch.Tensor, bias: torch.Tensor | None,
    out_features: int, in_features: int, group_size: int,
    compute_type: str, weight_type: str, scale_type: str, asym: bool,
) -> torch.Tensor:
    '''假实现：只用于 torch compile 元数据传播'''
    # 删除未用参数避免 lint 警告
    del qweight, bias, in_features, group_size, compute_type, weight_type, scale_type, asym
    # 返回形状匹配的空张量
    return torch.empty((*x.shape[:-1], out_features), dtype=x.dtype, device=x.device)

class ark_ops:
    @staticmethod
    def register_ops_once() -> None:
        '''模块导入时注册算子，仅执行一次'''
        global _OPS_REGISTERED
        if _OPS_REGISTERED:
            return
        is_available, error_str, _, _ = get_ark_state()
        if not is_available:
            logger.debug('Skip registering ark op because ARK is unavailable: %s',
                         error_str or 'unknown error')
            return
        direct_register_custom_op(
            op_name='inc_ark_woq_linear',
            op_func=_inc_ark_woq_linear_impl,
            fake_impl=_inc_ark_woq_linear_fake,
            dispatch_key=current_platform.dispatch_key,
        )
        _OPS_REGISTERED = True
```

```
ark_ops.register_ops_once()

__all__ = ['get_ark_state']
```

vllm/model_executor/layers/quantization/inc/schemes/inc_wna16_linear.py

核心修改文件：移除 get_ark_state 重复定义，调整 INCARKLinearMethod 使用注册算子。

```
class INCARKLinearMethod(INCXPULinearBase):
    def __init__(self, layer_config: 'INCLayerConfig') -> None:
        super().__init__(layer_config)
        from .inc_ark_ops import get_ark_state # 导入源改为新模块
        is_available, error_str, _, quant_linear_cls = get_ark_state()
        if not is_available or quant_linear_cls is None:
            raise ImportError(f'Failed to import auto_round_kernel. {error_str or "unknown error"}')
        self.quant_linear_cls = quant_linear_cls

    def process_weights_after_loading(self, layer: torch.nn.Module) -> None:
        # ... 构建 ark_linear 并复制权重 ...
        # 以前是 layer.ark_linear = ark_linear; del layer.qweight
        # 现在保留 qweight 并附加 ark_bias 等属性供算子使用
        layer.qweight = Parameter(ark_linear.qweight.detach(), requires_grad=False)
        layer.ark_bias = ark_linear.bias
        layer.ark_compute_type = ark_linear.cdt
        layer.ark_weight_type = ark_linear.wdt
        layer.ark_scale_type = ark_linear.sdt

    def apply_weights(self, layer, x, bias=None) -> torch.Tensor:
        # 直接调用注册算子，而非 layer.ark_linear.forward(x)
        return torch.ops.vllm.inc_ark_woq_linear.default(
            x,
            layer.qweight,
            layer.ark_bias,
            layer.out_features,
            layer.in_features,
            self.group_size,
            layer.ark_compute_type,
            layer.ark_weight_type,
            layer.ark_scale_type,
            not self.sym,
        )
```

评论区精华

- 注册风格建议：Reviewer yiliu30 建议参考 _xpu_ops.py 的注册方式，作者已按要求修改（commit ba2b11c）。
- del 操作合理性：Reviewer jikunshang 对 fake 实现中的 del 提出疑问，作者解释这是 vLLM 标记未使用参数的风格。

- 日志级别选择：Reviewer jikunshang 建议将 ARK 不可用时的日志从 debug 提升为 warning/error，作者认为不可用是预期 fallback 路径，debug 级别足以避免日志污染。
- 算子注册风格 (design): 作者已按建议修改，使用 direct_register_custom_op 并独立模块。
- fake impl 中 del 的使用 (style): 作者解释此为 vLLM 标记未使用参数的风格约定。
- ARK 不可用时日志级别 (design): 作者认为不可用是预期 fallback 路径，使用 debug 避免污染正常日志。

风险与影响

- 风险：
  - 版本兼容风险：auto_round_lib 最低版本从 >=0.13.3 升至 >=0.14.0，旧版本用户需升级，但 fallback 路径（Marlin、XPU 原生）仍可用。
  - torch compile 测试覆盖不足：测试仅覆盖 eager 模式，未包含 torch compile 路径，若编译图与 eager 行为不一致可能引入回归。
  - 注册时机：ark_ops.register_ops_once() 在模块导入时执行，若 ARK 运行时状态变化不会重新注册，但当前一次性设计对静态部署足够。
  - 多线程安全：全局变量 _OPS_REGISTERED 非线程安全，但 direct_register_custom_op 内部可能有保护。
- 影响：
  - 用户视角：对于使用 INC W4A16 量化模型（如 OPEA/Qwen2.5）在 XPU/CPU 上的用户，不再需要 --enforce-eager，可通过 torch.compile 获得 4-6% 性能提升。仅支持对称 4bit 量化。
  - 开发团队：新增 inc_ark_ops.py 模块，遵循 vLLM 自定义算子注册规范，为后续 XPU 算子提供参考。
  - 回归风险：修改了 INCARKLinearMethod 的权重处理和计算路径，ARK 可用时功能等价，fallback 路径未受影响。
  - 风险标记：版本依赖升级到 auto_round_lib>=0.14.0, torch compile 端到端测试未覆盖，多线程注册可能存在竞态

关联脉络

- 暂无明显关联 PR