

PR #48442 完整报告

vllm-project/vllm

[Perf] Zero-copy torch.Tensor pickling in shm_broadcast MessageQueue

合并时间: 2026-07-29 07:28

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/48442>

执行摘要

- 一句话: shm_broadcast 中 torch.Tensor 零拷贝序列化
- 推荐动作: 值得精读。该 PR 展示了如何利用 pickle 的 out-of-band buffer 机制绕过 PyTorch 默认序列化的冗余拷贝, 设计模式清晰 (自定义 reducer + dispatch_table), 可为类似 IPC 场景提供参考。评审者 njhill 的 comments 也体现了代码简洁性和正确性把关。

功能与动机

Large CPU tensors broadcast from the engine core to workers (e.g. prompt_embeds in SchedulerOutput) are byte-copied once by the engine on the scheduling critical path, and once more by every worker on receive. torch.Tensor.__reduce_ex__ copies tensor bytes into the pickle stream and never emits a PickleBuffer, so the out-of-band buffer path added to MessageQueue.enqueue in #26961 is dead code for tensors.

实现拆解

1. 在 vllm/distributed/device_communicators/shm_broadcast.py 中定义 _reduce_tensor 和 _rebuild_tensor 函数。_reduce_tensor 对符合条件的 CPU tensor (strided layout, no requires_grad) 将其原始内存通过 PickleBuffer 暴露给 pickle 的 buffer callback; _rebuild_tensor 根据收到的 buffer 类型 (zmq.Frame 零拷贝, 共享内存 chunk 拷贝) 重建 tensor。
2. 修改 MessageQueue.enqueue 方法, 在 pickle.dumps 时通过 Pickler.dispatch_table 注册 _reduce_tensor, 使 CPU tensor 自动走自定义 reducer。小 tensor (<1MiB) 仍内联, 大 tensor 走 out-of-band 路径。
3. 添加导入 copyreg 和 io。
4. 在 tests/distributed/test_shm_broadcast.py 中新增多进程测试 test_tensor_broadcast (验证 2 进程广播、ring 循环后数据完整性、tensor 可写性), 以及单元测试 test_tensor_pickle_roundtrip (覆盖 bf16/fp8/empty/scalar/non-contiguous/requires_grad/conj/Parameter) 和 test_reduce_tensor_fallback (验证 CUDA tensor 走默认 reducer)。
5. 在 tools/pre_commit/check_forbidden_imports.py 中将测试文件加入允许 pickle 的白名单。

关键文件:

- vllm/distributed/device_communicators/shm_broadcast.py (模块 通信层; 类别 source; 类型 core-logic; 符号 _rebuild_tensor, _reduce_tensor) : 核心实现: 新增 _reduce_tensor 和 _rebuild_tensor 函数, 修改 enqueue 注册 dispatch_table。
- tests/distributed/test_shm_broadcast.py (模块 测试; 类别 test; 类型 test-coverage; 符号 worker_fn_tensor_broadcast, test_tensor_broadcast, _dumps_oob, callback) : 全面测试覆盖: 多进程广播测试 + 单元测试覆盖各种 tensor 类型和边界条件。
- tools/pre_commit/check_forbidden_imports.py (模块 配置检查; 类别 source; 类型 configuration) : 将测试文件加入允许使用 pickle 的白名单。

关键符号: _rebuild_tensor, _reduce_tensor, worker_fn_tensor_broadcast, test_tensor_broadcast, _dumps_oob, callback, test_tensor_pickle_roundtrip, test_reduce_tensor_fallback

关键源码片段

vllm/distributed/device_communicators/shm_broadcast.py

核心实现: 新增 `_reduce_tensor` 和 `_rebuild_tensor` 函数, 修改 `enqueue` 注册 `dispatch_table`。

```
def _rebuild_tensor(buf: Any, shape: tuple[int, ...], dtype_str: str) -> torch.Tensor:
    """Rebuild a tensor from an out-of-band pickle buffer.

    Counterpart of `_reduce_tensor`. Note that pickle passes the original
    buffer-providing object from `loads(buffer=...)` straight to this
    function (no `PickleBuffer` wrapper on the receiving side), so `buf` is
    a `zmq.Frame`, a `memoryview` of a shared-memory ring chunk, or `bytes`
    if the buffer was serialized in-band.
    """
    dtype = getattr(torch, dtype_str)
    assert isinstance(dtype, torch.dtype)
    if isinstance(buf, zmq.Frame):
        # ZMQ frames own their message memory independently of any context,
        # so the tensor can safely alias it with zero copies. The tensor's
        # storage keeps the frame (and thus its bytes) alive via a strong
        # reference for as long as the tensor is.
        try:
            return torch.frombuffer(buf, dtype=torch.uint8).view(dtype).view(shape)
        except ValueError:
            # Empty or read-only frame buffer; fall through to the copy path.
            pass
    # Shared-memory ring buffer chunks are reused by the writer once all
    # readers have marked them read, so we must copy out of them. bytearray
    # (vs bytes) keeps the resulting tensor writable, matching normal tensor
    # semantics.
    raw = bytearray(buf)
    if not raw:
        assert 0 in shape
        return torch.empty(shape, dtype=dtype)
```

```
return torch.frombuffer(raw, dtype=torch.uint8).view(dtype).view(shape)
```

```
def _reduce_tensor(tensor: torch.Tensor):
    """Reduce a CPU tensor to a `PickleBuffer` for out-of-band pickling.

    `torch.Tensor.__reduce_ex__` copies the tensor bytes into the pickle
    byte stream via `torch.serialization` and never emits a `PickleBuffer`,
    which defeats the out-of-band buffer handling in `MessageQueue.enqueue`.
    This reducer instead exposes the tensor's memory directly, so large
    tensors (e.g. `prompt_embeds` in `SchedulerOutput`) traverse the queue
    without being copied into and back out of the pickled message.
    """
    if (
        tensor.device.type == "cpu"
        and tensor.layout == torch.strided
        and not tensor.requires_grad
    ):
        try:
            # The uint8 view exposes the raw bytes via the buffer protocol,
            # including for dtypes numpy doesn't recognize (bfloat16, fp8, ...).
            # reshape(-1) first so that 0-dim tensors can be viewed as well.
            raw = tensor.contiguous().reshape(-1).view(torch.uint8).numpy()
        except RuntimeError:
            # Exotic tensors (e.g. with the conjugate bit set) that don't
            # support aliasing views; let torch handle them.
            pass
        else:
            dtype_str = str(tensor.dtype).removeprefix("torch.")
            return _rebuild_tensor, (PickleBuffer(raw), tuple(tensor.shape), dtype_str)

    # Fall back to torch's default (copying) reduction.
    return tensor.__reduce_ex__(pickle.HIGHEST_PROTOCOL)
```

评论区精华

njhill 提出了 4 条 review 意见：① 在 `_rebuild_tensor` 空 buffer 分支添加 `assert 0 in shape`；② 简化 `_reduce_tensor` 中 `is_contiguous()` 检查为始终调用 `contiguous()`；③ 重排 `_rebuild_tensor` 中代码块使 fallback 在末尾；④ 将 out-of-band 阈值（1MiB）定义为常量。所有意见均被采纳并在后续 commit 中修复。

- 空 tensor 分支添加 `assert (correctness)`：已采纳，在 raw 为空时断言 0 in shape。
- 简化 `is_contiguous` 检查 (performance)：已简化，直接调用 `contiguous()`。
- 重排 `_rebuild_tensor` 代码块 (design)：已重排，现在先处理 `zmq.Frame` 零拷贝路径，再处理共享内存拷贝路径。
- out-of-band 阈值常量化 (design)：原作者 mrn3088 未直接回复，但最终代码保留了字面量 `1024 * 1024`。未修改。

风险与影响

- 风险：风险主要来自零拷贝语义：① ZMQ frame 零拷贝要求 frame 在 tensor 生命周期内保持存活，_rebuild_tensor 通过返回的 tensor 持有对 frame 的强引用来保证；② 共享内存 ring chunk 在 reader 标记读取后可能被 writer 重用，因此接收端必须拷贝 chunk 数据，代码已通过 bytearray(buf) 实现；③ 对于非 CPU、non-strided、requires_grad、conjugated 及子类 tensor，代码正确 fallback 到 torch 默认 reducer，不会产生错误。
- 影响：影响所有使用 multiproc executor 的部署，特别是涉及大 tensor（如 prompt_embeds、grammar bitmask）的调度抢占路径。用户无需修改代码，但可获得端到端延迟降低和内存带宽节省。更改仅限于 v1 引擎的分布式通信层，对模型输出位精确（bit-exact）。
- 风险标记：零拷贝依赖 ZMQ frame 生命周期，共享内存 chunk 必须 copy-out, CUDA tensor 降级路径，阈值硬编码

关联脉络

- PR #26961 Add out-of-band buffer path to MessageQueue.enqueue: 该 PR 添加了 out-of-band buffer 传输基础，此 PR 在此基础上为 tensor 添加自定义 reducer 使其真正生效。