

PR #56706 完整报告

vllm-project/vllm

[Bugfix] Fix stale HPC QK-norm weights after weight refit

合并时间: 2026-09-15 12:34

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/56706>

执行摘要

- 一句话: 修复 weight refit 后 HPC QK-norm 权重陈旧问题
- 推荐动作: 值得精读。本 PR 是一个教科书式的「消除派生状态」修复: 当 hook 无法覆盖所有生命周期路径时, 选择让消费者直接读权威数据源, 比枚举补齐所有 reload 路径更稳健。重点关注: (1) `_forward_impl` 直接读 `fallback_qnorm.weight` 的写法如何同时保住 CUDA Graph 指针稳定性 (同 `storage` 原地写); (2) `hy_v3.py` 用 `dtype` 参数把 bf16 checkpoint 到 fp32 内核输入的转换前移到加载期; (3) 测试中通过 `recording stub` 驱动生产 `_forward_impl` 并断言 `data_ptr` 不变的验证手法。对从事 RL weight refit、CUDA Graph 捕获或自定义后端 (HPC) 开发的同学有直接参考价值。

功能与动机

PR body 明确指出 `HpcRopeNorm` 的 `qnorm_weight/knorm_weight` 是 `q_norm/k_norm` 的独立 `copy().float()` 副本, 而填充它们的 `process_weights_after_loading` 仅由 `model_loader/utils.py` 的 `HpcModule` 循环在冷启动时调度; `layerwise.py` 只对带 `QuantizeMethodBase` 的层和 `deferred attention` 层调度该 hook, `HpcRopeNorm` 两者都不是。因此 RL weight refit (`reload_weights`、NCCL/IPC/sharded-RDT 权重传输、sleep 后的 `wake_up`) 后内核静默消费 refit 前的 QK-norm 权重, 无任何报错或警告。作者在 PR body 中用表格列出了各加载路径的 `dispatcher` 与是否到达 HPC 循环, 并强调这是静默错误。

实现拆解

本 PR 的核心思路是「消除派生副本, 让内核直接读 owner 权重」, 具体分四步:

1. `HpcRopeNorm` 去掉镜像 `Parameter`, 改为直接引用 `fallback norm`: 在 `vllm/model_executor/layers/hpc/rope_norm.py` 中删除 `qnorm_weight/knorm_weight` 两个 `Parameter` 的预分配与 `process_weights_after_loading` override; `_forward_impl` 中 `q_norm_weight/k_norm_weight` 改为直接取 `self.fallback_qnorm.weight/self.fallback_knorm.weight` (在 `qk_norm_policy != QkNormPolicy.NONE` 时)。同时删除 `self.use_qk_norm` 等不再使用的字段, 并在 `__init__` 增加显式 `ValueError`: 启用 QK-Norm 但未提供 `fallback` 模块时直接报错, 替代原来 `fallback_qnorm is not None` 的迟到检查。
2. 模型侧保证 `fallback norm` 以 `float32` 构建: 在 `vllm/model_executor/models/hy_v3.py` 中, `HunyuanAttention` 构造 `q_norm/k_norm` 时依据 `rope_support` (即 `HpcRopeNorm.support` 的结果) 传入 `dtype=torch.float32 if rope_support else None`。

这样 checkpoint 仍为 bf16, `default_weight_loader` 在 `param.data.copy_()` 时自动完成 bf16→fp32 转换, 加载语义不变。

3. HpcIHCPre/HpcIHCHHead 由「hook 绑定别名」改为「forward 直接读 owner」: 在 `vllm/model_executor/layers/hpc/hpc_ihc.py` 中删除 `process_weights_after_loading` override 以及 `self.weight/hc_scale/hc_base/rms_weight/rms_eps` 等缓存属性, `forward` 中直接通过 `self._fallback_op` 读取 `owner.hc_fn.weight`、`owner.hc_scale`、`owner.hc_base` 等。PR body 特别说明: iHC 缓存的是指向 owner Parameter 的别名, layerwise reload 的 `_copy_and_restore_kernel_tensors` 原地拷贝且 `_place_kernel_tensors` 重新注册原 Parameter 对象, 所以别名在 refit 后仍然有效, 此项属于纯 cleanup (作者用 base 臂实测 9/9 别名 refit 后仍指向存活权重)。
4. 删除通用 HpcModule 挂载点: 在 `vllm/model_executor/model_loader/utils.py` 的 `initialize_model` 中删除遍历 `isinstance(module, HpcModule)` 调用 `process_weights_after_loading` 的循环及其 import; `vllm/model_executor/layers/hpc/hpc_module.py` 的基类删除 `process_weights_after_loading` no-op。至此 4 个挂载点减为 3 个 (quant method、deferred attention、model 级 hook), HpcModule 基类只剩 `support()` 与 `forward()`。
5. 测试配套: 在 `tests/model_executor/model_loader/test_reload.py` 新增 fixture `hpc_rope_norm` (用 `stub types.ModuleType('hpc')` 隔离外部 hpc 包) 和 `test_hpc_rope_norm_kernel_sees_refit_norm_weights`, 用 recording stub 的 `rope_norm_store_kv` 捕获 `_forward_impl` 实际传给内核的 `q_norm_weight/k_norm_weight`, 经 `record_metadata_for_reloading` → `initialize_layerwise_reload` → `weight_loader` → `finalize_layerwise_reload` 驱动 refit 后断言: 收到的是新值、float32、`data_ptr` 不变, 且 `rnorm` 不再有 `qnorm_weight` 属性。该测试在旧代码上失败、新代码上通过, 是有效的回归测试。

关键文件:

- `vllm/model_executor/layers/hpc/rope_norm.py` (模块 HPC 算子; 类别 source; 类型 data-contract; 符号 HpcRopeNorm, `process_weights_after_loading`, `_forward_impl`, `init`): 修复核心: 删除 `qnorm_weight/knorm_weight` 镜像 Parameter 与 `process_weights_after_loading`, `_forward_impl` 改为直接读 `fallback_qnorm/fallback_knorm` 的 `weight`, 并在 `__init__` 加强制校验。
- `tests/model_executor/model_loader/test_reload.py` (模块 重载测试; 类别 test; 类型 test-coverage; 符号 `hpc_rope_norm`, `test_hpc_rope_norm_kernel_sees_refit_norm_weights`, `record`, `kernel_norm_weights`): 新增 `hpc_rope_norm` fixture 与 `test_hpc_rope_norm_kernel_sees_refit_norm_weights`, 用 recording stub 驱动生产 `_forward_impl`, 验证 refit 后内核收到新 float32 权重且 storage 不变; 无 hpc 包环境下可运行, 是证明 bug 存在且被修复的关键证据。
- `vllm/model_executor/layers/hpc/hpc_ihc.py` (模块 HPC 算子; 类别 source; 类型 data-contract; 符号 HpcIHCPre, HpcIHCHHead, `process_weights_after_loading`): 清理 HpcIHCPre/HpcIHCHHead 的 `process_weights_after_loading` 与缓存别名属性, `forward` 直接读 owner 权重, 为删除通用 HpcModule 挂载点铺路。

- vllm/model_executor/models/hy_v3.py (模块 模型定义; 类别 source; 类型 data-contract; 符号 HunyuanAttention.init) : HPC 激活时以 float32 构建 q_norm/k_norm, 保证内核直接读取的权重 dtype 正确, bf16 checkpoint 转换由 default_weight_loader 隐式完成。
- vllm/model_executor/model_loader/utils.py (模块 模型加载; 类别 source; 类型 data-contract; 符号 initialize_model) : 删除 initialize_model 中遍历 HpcModule 调用 process_weights_after_loading 的通用循环与 import, 是 4 个挂载点 → 3 个的关键改动。
- vllm/model_executor/layers/hpc/hpc_module.py (模块 HPC 算子; 类别 source; 类型 data-contract; 符号 HpcModule) : HpcModule 基类删除 process_weights_after_loading no-op, 基类缩小为仅 support() + forward() 的标记类。

关键符号: HpcRopeNorm.init, HpcRopeNorm._forward_impl, HpcRopeNorm.process_weights_after_loading, HpcIHCPre.forward, HpcIHCHHead.forward, HpcIHCPre.process_weights_after_loading, HpcIHCPre.support, initialize_model, test_hpc_rope_norm_kernel_sees_refit_norm_weights

关键源码片段

vllm/model_executor/layers/hpc/rope_norm.py

修复核心: 删除 qnorm_weight/knorm_weight 镜像 Parameter 与 process_weights_after_loading, _forward_impl 改为直接读 fallback_qnorm/fallback_knorm 的 weight, 并在 __init__ 加强制校验。

```
# vllm/model_executor/layers/hpc/rope_norm.py
# 修复前后最关键的变化集中在两个地方:
# 1. __init__ 不再预分配 qnorm_weight / knorm_weight 镜像 Parameter,
# 改为强约束: 启用 QK-Norm 时必须有 fallback 模块。
# 2. _forward_impl 传给 hpc.rope_norm_store_kv 的权重直接取自
# fallback_qnorm / fallback_knorm, 而不是曾经无人刷新的副本。
```

```
class HpcRopeNorm(CustomOp, HpcModule):
```

```
    def __init__(
        self,
        num_heads: int,
        num_kv_heads: int,
        head_dim: int,
        cos_sin_cache: torch.Tensor,
        use_qk_norm: bool,
        fallback_qnorm: torch.nn.Module | None,
        fallback_knorm: torch.nn.Module | None,
        kv_cache_dtype: str,
        layer_name: str,
        qk_norm_policy: QkNormPolicy = QkNormPolicy.ROPE_THEN_NORM,
    ) -> None:
        super().__init__()
        # ... 省略 hpc 包检查与基本字段赋值 ...
```

```

self.use_fp8 = "fp8" in kv_cache_dtype
# RMSNorm/RoPE 顺序由调用方决定 (如 HunYuan V3 是 NORM_THEN_ROPE)
self.qk_norm_policy = qk_norm_policy if use_qk_norm else QkNormPolicy.NONE
# 新增的不变式校验: 内核现在直接读 fallback 模块的权重,
# 缺失时立刻报错, 而不是等到真正调用时才暴露问题。
if self.qk_norm_policy != QkNormPolicy.NONE and (
    fallback_qnorm is None or fallback_knorm is None
):
    raise ValueError(
        "QK-Norm is enabled but fallback_qnorm/fallback_knorm were not "
        "supplied; the fused kernel reads their weights directly."
    )

# layer_name 注册与全局实例 registry (供顶层 custom op 路由)
self.layer_name = None
self.register_layer_name(layer_name)
# ... 省略 hpc.QuantType 读取 ...

def _forward_impl(self, qkv, kv_cache, attn_metadata, attn_layer, output):
    # ... 省略 prefill/decode 分支与 KV cache 布局判断 ...

    # 修复核心: 此前这里读 self.qnorm_weight / self.knorm_weight (
    # 独立 copy_() 出来的 fp32 副本, refit 路径无人刷新),
    # 现在直接读 fallback RMSNorm 的 weight (同 storage,
    # layerwise reload 原地写, CUDA Graph 捕获的指针不变)。
    if self.qk_norm_policy != QkNormPolicy.NONE:
        assert self.fallback_qnorm is not None
        assert self.fallback_knorm is not None
        q_norm_weight = self.fallback_qnorm.weight
        k_norm_weight = self.fallback_knorm.weight
    else:
        q_norm_weight = None
        k_norm_weight = None

    # ... 后续把 q_norm_weight / k_norm_weight 传给
    # hpc.rope_norm_store_kv 或 hpc.rope_norm_store_kv_fp8 ...

```

tests/model_executor/model_loader/test_reload.py

新增 hpc_rope_norm fixture 与 test_hpc_rope_norm_kernel_sees_refit_norm_weights, 用 recording stub 驱动生产 _forward_impl, 验证 refit 后内核收到新 float32 权重且 storage 不变; 无 hpc 包环境下可运行, 是证明 bug 存在且被修复的关键证据。

```

# tests/model_executor/model_loader/test_reload.py
# 新增 fixture: 在无 hpc 包的环境里注入一个最小 stub,
# 让 HpcRopeNorm 可以被 import 并实例化。
@pytest.fixture
def hpc_rope_norm(monkeypatch, default_vllm_config):
    """Import HpcRopeNorm with the external ``hpc`` package stubbed out."""
    if "hpc" not in sys.modules:

```

```

stub = types.ModuleType("hpc")
stub.__spec__ = importlib.machinery.ModuleSpec("hpc", loader=None)
stub.QuantType = types.SimpleNamespace(
    QPERTOKEN_PERHEAD_KPERTENSOR_VPERTENSOR=types.SimpleNamespace(value=0)
)
monkeypatch.setitem(sys.modules, "hpc", stub)
from vllm.model_executor.layers.hpc import rope_norm

monkeypatch.setattr(rope_norm, "_hpc_rope_norm_instances", {})
return rope_norm

```

```

# 回归测试：先加载一组权重，记录内核收到的张量指针；
# 再走 layerwise reload 路径（与 RL weight refit 相同的机制）原地改写权重，
# 断言内核随后收到的是新值、float32、且 storage 指针不变。
def test_hpc_rope_norm_kernel_sees_refit_norm_weights(monkeypatch, hpc_rope_norm):
    from vllm.model_executor.layers.layernorm import RMSNorm

```

```

    head_dim, num_heads, num_kv_heads, block_size = 128, 8, 1, 4
    layer = torch.nn.Module()
    layer.q_norm = RMSNorm(head_dim, 1e-6, dtype=torch.float32)
    layer.k_norm = RMSNorm(head_dim, 1e-6, dtype=torch.float32)
    layer.hpc_rope_norm = hpc_rope_norm.HpcRopeNorm(
        num_heads=num_heads,
        num_kv_heads=num_kv_heads,
        head_dim=head_dim,
        cos_sin_cache=torch.ones(16, head_dim),
        use_qk_norm=True,
        fallback_qnorm=layer.q_norm,
        fallback_knorm=layer.k_norm,
        kv_cache_dtype="auto",
        layer_name="hpc_test_layer",
    )
    model = torch.nn.Sequential(layer)
    rnorm = layer.hpc_rope_norm

```

```

calls = []

```

```

def record(*args, **kwargs):
    calls.append(kwargs)

```

```

monkeypatch.setattr(sys.modules["hpc"], "rope_norm_store_kv", record, raising=False)

```

```

def kernel_norm_weights():
    # 构造最小输入，驱动生产 _forward_impl，捕获内核收到的权重参数
    q_size, kv_size = num_heads * head_dim, num_kv_heads * head_dim
    qkv = torch.zeros(1, q_size + 2 * kv_size, dtype=torch.bfloat16)
    kv_cache = torch.zeros(2, num_kv_heads, block_size, 2 * head_dim, dtype=torch.bfloat16)
    attn_layer = types.SimpleNamespace(_k_scale=torch.ones(1), _v_scale=torch.ones(1))

```

```

attn_metadata = types.SimpleNamespace(
    num_actual_tokens=1, num_decodes=1, num_decode_tokens=1,
    num_prefills=0, num_prefill_tokens=0, max_query_len=1,
    decode_query_len=1, qo_indptr=None, qo_indptr_decode=None,
    slot_mapping=torch.tensor([4]), seq_lens=torch.tensor([1]),
    block_table_tensor=torch.tensor([[1]]), hpc_kv_written=False,
)
output = torch.zeros(1, q_size, dtype=torch.bfloat16)
rnorm._forward_impl(qkv, kv_cache, attn_metadata, attn_layer, output)
return calls[-1]["q_norm_weight"], calls[-1]["k_norm_weight"]

def loaded(value):
    return torch.full((head_dim,), value, dtype=torch.bfloat16)

default_weight_loader(layer.q_norm.weight, loaded(0.5))
default_weight_loader(layer.k_norm.weight, loaded(0.25))
q, k = kernel_norm_weights()
assert torch.equal(q, loaded(0.5).float())
assert torch.equal(k, loaded(0.25).float())
q_ptr, k_ptr = q.data_ptr(), k.data_ptr()

# 触发与 RL weight refit 相同的 layerwise reload 机制
record_metadata_for_reloading(model)
initialize_layerwise_reload(model)
layer.q_norm.weight.weight_loader(layer.q_norm.weight, loaded(2.0))
layer.k_norm.weight.weight_loader(layer.k_norm.weight, loaded(3.0))
finalize_layerwise_reload(model, model_config=None)

q, k = kernel_norm_weights()
# 修复后：内核收到 refit 后的新值、float32、且与之前同一段存储
assert q.dtype == k.dtype == torch.float32
assert torch.equal(q, loaded(2.0).float())
assert torch.equal(k, loaded(3.0).float())
assert (q.data_ptr(), k.data_ptr()) == (q_ptr, k_ptr)
# 派生副本已彻底删除
assert not hasattr(rnorm, "qnorm_weight")

```

评论区精华

Review 讨论集中在验证责任与 CI 上：

- aoshen02 在 PR 评论中请求 HPC 作者 @thisjiang 审阅，CI 通过 Buildkite 触发两次（#88695、#88800）。
- 本 PR 的「未验证项」是 sm90 + HY3 + HPC_ATTN refit：
HpcAttentionBackend.supports_compute_capability 仅接受 capability == (9, 0)，而作者的集群只有 GB200 (sm100)，实际修复代码 (HpcRopeNorm、hy_v3.py) 在 sm100 上无法被 HPC_ATTN 门控触发，只能端到端验证 iHC cleanup 半部分与挂载点移除。作者明确请求 Tencent hpc-ops 作者或有 H100/H800 的人跑 sm90 refit 一致性检查，并强调

脚本必须在 main 上失败否则没有在验证 bug。

- @thisjiang 随后回复已在 Hopper GPU 上验证 'sm90 + HY3 + HPC_ATTN refit' 结果正确，作者致谢。这一来一回解决了 PR 最大的验证盲区。
- ywang96 以 APPROVED 批准，无 inline review 评论。
- sm90 + HY3 + HPC_ATTN refit 验证责任 (testing): PR 评论中 @thisjiang 回复已在 Hopper GPU 上验证 'sm90 + HY3 + HPC_ATTN refit' 结果正确，作者致谢。验证盲区被社区成员补上。
- CI 触发与运行状态 (other): CI 正常运行，最终 ywang96 APPROVED。
- sharded-state 工件兼容性 (design): 作者选择不在 HPC PR 中改动共享 loader，需求留待维护者决定；未在 review 评论中进一步讨论。

风险与影响

- 风险：
 1. 数据契约变更（主要风险）：HpcRopeNorm 的 qnorm_weight/knorm_weight 从 state_dict 中消失。旧构建保存的 sharded-state 工件若包含这两个 key，ShardedStateLoader 会按 key 索引 state_dict 直接 KeyError。PR body 已写明此限制并要求重新生成 artifact；同时该变更实际是把不兼容方向互换——main 上无 HPC 构建加载含这两个 key 的工件本就会 ValueError。
 2. fp32 契约靠调用方保证：hy_v3.py 是唯一 in-tree 调用方，__init__ 只校验 fallback 模块存在，不校验其权重是否为 float32。若未来新模型接入 HPC_ATTN 但未像 hy_v3 一样以 float32 构建 RMSNorm，融合内核可能收到非 fp32 权重，属于潜在隐性契约。作者将其列为 follow-up。
 3. 基类面积缩小：HpcModule 删除 no-op 后，所有 HpcModule 子类若还有依赖 process_weights_after_loading 的未迁移代码会直接 AttributeError；本 PR 已迁移全部 in-tree 子类，但外部 OOT 注册的 HPC 模块不受保护。
 4. 测试依赖外部 stub：新测试用 sys.modules['hpc'] stub 与 monkeypatch.setattr(..., raising=False)，若生产代码对 hpc 包的调用点增加而 stub 未同步，测试可能静默失效。
 5. 性能影响可忽略：PR 实测 eager 模式 +2.91% 与 same-code 噪声底 +2.47% 相当，cudagraph 模式 +0.16% 与 +0.15% 相当，判定为 warm-up 顺序而非代码差异。
- 影响：影响范围集中在 HPC 后端与权重重载路径：
- 用户 / 系统：修复 RL weight refit (reload_weights、NCCL/IPC/sharded-RDT 传输、sleep wake_up) 后 HunYuan V3/V4 在 GB200/Hopper 上静默使用陈旧 QK-norm 权重的正确性问题。GSM8K 500 题 5-shot A/B (4 节点 16 卡 TP=4xPP=4) 显示 base 与 PR 均为 0.5660、invalid_rate 0.0120，输出 8/8 token 一致，说明修复不改变冷启动语义。
- 团队：删除模型加载器通用 HPC hook 后，model_loader/utils.py 的职责更清晰；后续 HPC 模块必须自管权重引用而不能依赖集中式 post-load 处理。HpcModule 基类实际降级为标记类。
- 兼容性：sharded-state artifact 需重新生成；对使用旧 artifact 的存量部署有升级成本，但方向由「无 HPC 加载带 HPC key 失败」转为「HPC 构建加载旧 key 失败」。

- 验证面：新增单测可在无 hpc 包环境下运行，提高 HPC 相关回归的覆盖度；但 sm90/HY3 验证依赖社区成员手动执行，未固化为 CI。
- 风险标记：核心路径变更，数据契约变更，状态恢复兼容性，外部包依赖，缺少 sm90 CI 覆盖

关联脉络

- PR #47443 HPC attention decode split-K selection: PR body 明确检索确认 #47443 是唯一相近的 HPC 相关 open PR，但它是 HPC attention decode split-K 选择，与本 PR 的 QK-norm 陈旧权重问题无重叠。
- PR #54477 RFC for selective weight reload in RL: PR body 引用 #54477 作为 RL 场景选择性权重重载的 umbrella RFC，本 PR 修复的正是该场景下 HPC QK-norm 权重未随 refit 刷新的问题。
- PR #56472 [Bugfix] Export weight-cache IPC tensors separately for each client: 同属 weight reload/weight-cache 生命周期正确性修复，与本 PR 的 refit 后引用失效问题在同一功能域（weight reload 路径一致性）。