PR #50721 完整报告

vllm-project/vllm

[MRV2] Enable routed-experts capture

合并时间：2026-08-04 05:00

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50721

执行摘要

- 一句话：MRV2 启用专家路由捕获，统一 MRV1 快照逻辑
- 推荐动作：值得精读。一是看 njhill 主导的 " 把 RE 专属逻辑封装进 RoutedExpertsCapturer 内部 " 的重构，这是组件边界划分的范例；二是 get_routed_experts 的 clone 语义与 AsyncOutput 生命周期设计，直接决定异步路径正确性；三是 fail-closed 的 get_routed_experts_attn_gid 和越界 IndexError 体现了 " 宁可报错也不静默丢数据 " 的工程取向。

功能与动机

MRV2 是 vLLM 新一代模型运行器，但 routed-experts capture 此前被 _get_v2_model_runner_unsupported_features 显式排除（配置中注释标注 "Will be added by PR#38163"）。本 PR 的目标是让 MRV2 与 MRV1 行为对齐：enable_return_routed_experts 开启后，MRV2 也能返回完整的 routed experts 数组，同时保持 MRV1 协议与行为完全不变。PR body 强调验证目标是 "preserving the MRV1 protocol and behavior"。

实现拆解

1. 配置开关解除：在 vllm/config/vllm.py 的 _get_v2_model_runner_unsupported_features 中删除 enable_return_routed_experts 对应的 unsupported 项（含指向 PR#38163 的注释），宣告 MRV2 正式支持该功能。
2. 捕获逻辑共享化（核心重构）：在 vllm/model_executor/layers/fused_moe/routed_experts_capturer.py 中新增三个公共符号：RoutedExpertsCapturer.get_routed_experts(slot_mappings, num_tokens)（把当前 step 的 routing_data 与 attention 组 slot_mapping 打成 RoutedExpertsTensors 并 clone）、bind_routed_experts_capturer(model, capturer)（遍历所有 MoERunner 层，modular-kernel 路径挂到 BaseRouter.set_capture_fn，monolithic 路径要求 FusedMoEExpertsMonolithic.supports_routing_replay_capture()，一个都没绑上时抛错）、get_routed_experts_attn_gid(kv_cache_config)（fail-closed 选择第一个 FullAttentionSpec 组）。RoutedExpertsCapturer.__init__ 增加 kv_cache_config 参数并在内部持有 attn_gid；删除 clear_buffer（因每层 forward 都会覆盖当前 step 的 token 行，清零冗余且可能掩盖不完整捕获）；layer_id 越界从静默 return 改为 raise IndexError，fail-fast。

3. MRV1 对齐：vllm/v1/worker/gpu_model_runner.py 删除私有方法 _bind_routed_experts_capturer 和 _get_attention_kv_cache_gid，统一改调公共 API；新增 get_routed_experts(num_tokens) 供 async 路径复用，_prepare_inputs 中引用 self.routed_experts_capturer.attn_gid 代替原先的独立属性。
4. MRV2 接入：vllm/v1/worker/gpu/model_runner.py 在 __init__ 增加 routed_experts_capturer 属性并新增 init_routed_experts_capturer()；execute_model 在目标 forward 后（not dummy_run 时）调用 capturer.get_routed_experts(slot_mappings, num_toks) 生成快照，经 ExecuteModelState.routed_experts 传给 sample_tokens，再写入 AsyncOutput；vllm/v1/worker/gpu/async_utils.py 的 AsyncOutput 增加 routed_experts 字段，构造时调用 to_cpu_nonblocking()，get_output() 时写回 ModelRunnerOutput.routed_experts。
5. Ray 零拷贝边界：vllm/v1/executor/ray_utils.py 的 detach_zero_copy_from_model_runner_output 从只处理 logprobs 扩展为同时处理 routed_experts，对只读 numpy 数组做 copy，避免编译 DAG SHM 通道因引用残留而阻塞（RAY_CGRAPH_get_timeout）。
6. 测试配套：tests/model_executor/test_routed_experts_capture.py 由私有 _bind_routed_experts_capturer 测试改为公共 bind_routed_experts_capturer 测试，新增 attn_gid fail-closed、MRV2 AsyncOutput 透传、全部 TP rank 初始化捕获等用例；tests/v1/executor/test_ray_utils.py 新增 routed-experts 零拷贝分离测试；test_gpu_model_runner_v2_eplb.py 的 ExecuteModelState fixture 补上 routed_experts=None 字段。

关键文件：

- vllm/model_executor/layers/fused_moe/routed_experts_capturer.py（模块 专家路由；类别 source；类型 data-contract；符号 clear_buffer, get_routed_experts, bind_routed_experts_capturer, capture_fn）：核心数据契约文件：新增公共 API get_routed_experts、bind_routed_experts_capturer、get_routed_experts_attn_gid, capturer 内部持有 attn_gid，删除 clear_buffer 并改为越界 fail-fast。
- vllm/v1/worker/gpu/model_runner.py（模块 模型执行；类别 source；类型 data-contract；符号 init_routed_experts_capturer）：MRV2 入口：新增 init_routed_experts_capturer，execute_model 在 forward 后快照 routed experts 并经 ExecuteModelState 传递到 sample_tokens。
- vllm/v1/worker/gpu_model_runner.py（模块 模型执行；类别 source；类型 refactor；符号 _get_attention_kv_cache_gid, get_routed_experts, _bind_routed_experts_capturer, _capture_fn）：MRV1 主路径：删除私有 _bind_routed_experts_capturer 与 _get_attention_kv_cache_gid，改用公共 API，并新增 get_routed_experts 简化 async 路径。
- vllm/v1/executor/ray_utils.py（模块 分布式执行；类别 source；类型 core-logic）：Ray 编译 DAG 零拷贝边界：detach_zero_copy_from_model_runner_output 扩展处理 routed_experts，避免 SHM 通道阻塞。
- vllm/v1/worker/gpu/async_utils.py（模块 异步输出；类别 source；类型 core-logic）：AsyncOutput 增加 routed_experts 字段并异步 D2H，是 MRV2 异步路径传递

routed-experts 的载体。

- vllm/config/vllm.py（模块 配置层；类别 source；类型 configuration）：从 MRV2 unsupported features 列表中移除 routed experts capture，这是功能开通的声明点。
- tests/model_executor/test_routed_experts_capture.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_gpu_model_runner_binds_router_capture, _DummyRouter, DummyFusedMoE, DummyCapturer）：核心测试文件：绑定逻辑测试从私有方法迁移到公共 bind_routed_experts_capturer，新增 attn_gid fail-closed、MRV2 AsyncOutput 透传、全 TP rank 初始化等用例。
- tests/v1/executor/test_ray_utils.py（模块 单元测试；类别 test；类型 test-coverage；符号 test_detach_zero_copy_routed_experts_without_logprobs）：新增 test_detach_zero_copy_routed_experts_without_logprobs，验证无 logprobs 时 routed-experts 也能正确脱离 SHM。
- tests/kernels/moe/test_routed_experts_capture_monolithic.py（模块 单元测试；类别 test；类型 test-coverage）：同步更新 monolithic 路径的测试注释与绑定方式，覆盖绑定逻辑重构。
- tests/v1/worker/test_gpu_model_runner_v2_eplb.py（模块 单元测试；类别 test；类型 test-coverage）：为 ExecuteModelState 新字段 routed_experts 补全 fixture，保持 MRV2 EPLB 测试可运行。

关键符号：RoutedExpertsCapturer.init, RoutedExpertsCapturer.capture, RoutedExpertsCapturer.get_device_buffer, RoutedExpertsCapturer.get_routed_experts, bind_routed_experts_capturer, get_routed_experts_attn_gid, GPUModelRunner.get_routed_experts, GPUModelRunner.init_routed_experts_capturer, GPUModelRunner.execute_model, AsyncOutput.get_output, ExecuteModelState, detach_zero_copy_from_model_runner_output

关键源码片段

vllm/model_executor/layers/fused_moe/routed_experts_capturer.py

核心数据契约文件：新增公共 API get_routed_experts、bind_routed_experts_capturer、get_routed_experts_attn_gid，capturer 内部持有 attn_gid，删除 clear_buffer 并改为越界 fail-fast。

```
# vllm/model_executor/layers/fused_moe/routed_experts_capturer.py
# RoutedExpertsCapturer 位于 worker 侧（GPU），forward 中逐层记录
# topk_ids；步骤结束时快照 routing data 与 attention slot mapping。

class RoutedExpertsCapturer:
    def __init__(self, max_num_batched_tokens: int, vllm_config: VllmConfig,
                 kv_cache_config: KVCacheConfig) -> None:
        hf_config = vllm_config.model_config.hf_text_config
        # 设备缓冲使用 int32，匹配 router 原生 topk_ids dtype；NCCL 对
        # uint8/uint16 的 all-gather 支持因版本而异，int32 最稳。
        self.device_buffer = torch.zeros(
            (max_num_batched_tokens, hf_config.num_hidden_layers,
```

```
            _get_num_experts_per_tok(hf_config)),
            dtype=torch.int32,
            device=current_platform.device_type,
        )
        self.dp_rank = vllm_config.parallel_config.data_parallel_rank
        self.tp_size = vllm_config.parallel_config.tensor_parallel_size
        # attn_gid 在构造时确定，与 scheduler 侧 RoutedExpertsManager
        # 共用 get_routed_experts_attn_gid，保证两侧 slot 布局一致。
        self.attn_gid = get_routed_experts_attn_gid(kv_cache_config)

    def get_device_buffer(self) -> torch.Tensor:
        return self.device_buffer

    def get_routed_experts(self, slot_mappings: torch.Tensor,
                           num_tokens: int) -> RoutedExpertsTensors:
        # 必须 clone：capture buffer 与共享 slot_mappings 在下一步会被
        # 覆盖，而异步 D2H 复制可能仍在途中，直接引用会读到撕裂数据。
        return RoutedExpertsTensors(
            routing_data=self.device_buffer[:num_tokens].clone(),
            slot_mapping=slot_mappings[self.attn_gid, :num_tokens].clone(),
        )


def bind_routed_experts_capturer(model: torch.nn.Module,
                                 capturer: RoutedExpertsCapturer) -> None:
    """把 capture 回调挂到目标模型的 MoE router 上。"""
    num_bound = 0
    for module in model.modules():
        if not isinstance(module, MoERunner):
            continue
        layer_id = module.layer_id

        def capture_fn(topk_ids: torch.Tensor,
                       layer_id: int = layer_id,
                       capturer: RoutedExpertsCapturer = capturer) -> None:
            capturer.capture(layer_id, topk_ids)

        quant_method = module._quant_method
        moe_kernel = getattr(quant_method, "moe_kernel", None)
        impl = getattr(moe_kernel, "impl", None)
        fused_experts = getattr(impl, "fused_experts", None)
        if quant_method.is_monolithic:
            # monolithic 内核只有支持 routing replay 时才可捕获
            if not (isinstance(fused_experts, FusedMoEExpertsMonolithic)
                    and fused_experts.supports_routing_replay_capture()):
                raise ValueError(
                    "Routed-experts capture is not supported with monolithic "
                    f"MoE kernel {type(fused_experts).__name__}.")
            fused_experts.set_capture_fn(capture_fn)
```

```
            num_bound += 1
        elif isinstance(module.router, BaseRouter):
            # modular-kernel 路径：捕获发生在 router 输出 topk_ids 时
            module.router.set_capture_fn(capture_fn)
            num_bound += 1
        else:
            raise ValueError(
                "Routed-experts capture is not supported with router "
                f"{type(module.router).__name__}.")

    # 一个层都没绑上说明配置与模型不匹配，宁可报错也不静默返回空数据
    if num_bound == 0:
        raise ValueError("No supported MoE router found for routed-experts capture.")


def get_routed_experts_attn_gid(kv_cache_config: KVCacheConfig) -> int:
    """返回 routed-experts 对应的 full-attention KV cache group。"""
    # fail-closed：混合模型（Mamba/linear attention）存在多个 KV group，
    # 找不到 full-attention 组直接抛错，避免 worker 与 scheduler 之间
    # 产生隐式的 slot 布局不一致。
    for gid, group in enumerate(kv_cache_config.kv_cache_groups):
        if isinstance(group.kv_cache_spec, FullAttentionSpec):
            return gid
    raise ValueError(
        "Routed-experts capture requires a full-attention KV cache group.")
```

vllm/v1/worker/gpu/model_runner.py

MRV2 入口：新增 init_routed_experts_capturer，execute_model 在 forward 后快照 routed experts 并经 ExecuteModelState 传递到 sample_tokens。

```
# vllm/v1/worker/gpu/model_runner.py
# MRV2 接入：capturer 初始化 → execute_model 快照 → ExecuteModelState
# 传递 → sample_tokens 写入 AsyncOutput。

    def init_routed_experts_capturer(self) -> None:
        """Initialize target-model capture on every participating worker."""
        self.routed_experts_capturer = RoutedExpertsCapturer(
            max_num_batched_tokens=self.max_num_tokens,
            vllm_config=self.vllm_config,
            kv_cache_config=self.kv_cache_config,
        )
        bind_routed_experts_capturer(self.model, self.routed_experts_capturer)

    # 以下位于 execute_model() 内部，目标模型 forward 之后：
    # dummy_run 不产生真实输出，跳过快照；capturer 未初始化时为 None。
    routed_experts = None
    if not dummy_run and (capturer := self.routed_experts_capturer) is not None:
        assert slot_mappings is not None
        # attn_gid 已封装在 capturer 内部，runner 无需关心 KV group 选择
```

```
            routed_experts = capturer.get_routed_experts(slot_mappings, num_toks)

        self.execute_model_state = ExecuteModelState(
            input_batch=input_batch,
            attn_metadata=attn_metadata,
            slot_mappings_by_layer=slot_mappings_by_layer,
            hidden_states=hidden_states,
            aux_hidden_states=aux_hidden_states,
            finished_req_ids=finished_req_ids,
            routed_experts=routed_experts,
        )

        # sample_tokens() 中取出该字段并交给 AsyncOutput:
        # AsyncOutput(..., routed_experts=routed_experts) 会调用
        # routed_experts.to_cpu_nonblocking() 异步搬运，get_output() 时再
        # 写回 ModelRunnerOutput.routed_experts。
```

vllm/v1/executor/ray_utils.py

Ray 编译 DAG 零拷贝边界：detach_zero_copy_from_model_runner_output 扩展处理 routed_experts，避免 SHM 通道阻塞。

```
# vllm/v1/executor/ray_utils.py
# Ray 编译 DAG 的 SHM 通道可能返回只读的零拷贝 numpy 数组；若在这些
# 数组仍被引用时继续下一个 scheduler 迭代，会阻塞通道并最终触发
# RAY_CGRAPH_get_timeout，因此必须在公共输出边界复制并脱离引用。

def detach_zero_copy_from_model_runner_output(output: "ModelRunnerOutput") -> None:
    def _copy_if_readonly(arr):
        if isinstance(arr, np.ndarray) and not arr.flags.writeable:
            return arr.copy()
        return arr

    # logprobs: cu_num_generated_tokens 是普通 Python list，不会别名
    # Ray SHM 缓冲区，可直接复用；只复制只读的 numpy 视图。
    if output.logprobs is not None:
        token_ids, logprobs, ranks, cu_num_generated_tokens = output.logprobs
        token_ids_c = _copy_if_readonly(token_ids)
        logprobs_c = _copy_if_readonly(logprobs)
        ranks_c = _copy_if_readonly(ranks)
        if (token_ids_c is not token_ids
                or logprobs_c is not logprobs
                or ranks_c is not ranks):
            output.logprobs = type(output.logprobs)(
                token_ids_c, logprobs_c, ranks_c, cu_num_generated_tokens)

    # routed-experts: routing_data 与 slot_mapping 同样可能是 SHM 视图，
    # 用相同策略复制；重建同类型容器保持输出协议不变。
    if output.routed_experts is not None:
        routing_data, slot_mapping = output.routed_experts
```

```
routing_data_c = _copy_if_readonly(routing_data)
slot_mapping_c = _copy_if_readonly(slot_mapping)
if (routing_data_c is not routing_data
        or slot_mapping_c is not slot_mapping):
    output.routed_experts = type(output.routed_experts)(
        routing_data_c, slot_mapping_c)
```

评论区精华

评审主要由合并者 njhill 主导，核心交锋围绕 "runner 里别留 RE 专属逻辑 ":

- njhill 在 MRV2 *execute_model* 内联快照代码上连发两条评论："Let's move this into a separate get_routed_experts method?" 和 "Can we add a get_routing_data(num_tokens: int) -> torch.Tensor method to RoutedExpertsCapturer containing this logic"，作者回复 "done." / "no problem."。
- njhill 进一步提出更大的封装建议："could we move even more of this inside RoutedExpertsCapturer i.e. pass self.kv_cache_config to its constructor so that it can keep routed_experts_attn_gid internally"。作者原本计划放到另一个 PR 处理，njhill 随后表示 "As discussed I pushed a commit with this refactor to keep the model runner cleaner"——最终提交（further simplify model runner）即此重构。
- async_utils.py 上有字段命名一致的 style 建议（for consistency self.routed_experts = routed_experts），作者采纳。
- njhill 两次 APPROVE，最终合并。
- MRV2 *execute_model* 内联快照逻辑抽取为独立方法 (design): 作者回复 "done."，最终 MRV2 runner 通过 capturer.get_routed_experts(slot_mappings, num_toks) 一行完成快照。
- 把 attn_gid 与快照 API 全部封装进 RoutedExpertsCapturer (design): njhill 直接提交最后一个 commit（further simplify model runner），attn_gid 进入 capturer 内部，MRV1/MRV2/scheduler 三方共享同一个选择函数。
- AsyncOutput 字段命名与初始化一致性 (style): 作者回复 "done."，最终字段名为 routed_experts / routed_experts_cpu，与 sampler 输出字段风格统一。
- dummy_run 守卫与 slot_mappings 断言位置 (design): 最终版本在 *execute_model* 中显式判断 not dummy_run，capturer 未初始化时返回 None。

风险与影响

- 风险：
  1. 删除 clear_buffer 的假设风险：capture 语义从 " 每步清零 " 改为 " 每个 routed 层覆盖当前 token 行 "。若未来出现不经过 capture_fn 的 MoE 层（如非 MoERunner 实现或跳过 capture 的量化路径），旧 step 数据会残留在 device_buffer 中并混入结果；当前测试仅覆盖 modular 与 monolithic 两类已知路径。
  2. 越界 fail-fast 的行为变更：layer_id 超出 buffer 从静默 return 改为 IndexError，对使用了超过 hf_config.num_hidden_layers 层数的非常规模型会直接报错而非静默丢弃，属于有意的契约收紧，但可能影响未预料到的模型族。

3. 异步 D2H 竞态：正确性依赖 get_routed_experts 中的 clone 语义；AsyncOutput 在 get_output() 后释放 GPU snapshot，若调用方提前复用 capturer buffer 或 slot_mappings，copy stream 会读到撕裂数据。
4. MRV1 回归：MRV1 runner 删除 66 行内联逻辑改为公共 API，行为等价性主要靠提交说明中的 GSM8K 端到端验证支撑，仓库 CI 未见对应端到端任务。
5. Ray 路径：detach_zero_copy_from_model_runner_output 现在同时处理 logprobs 与 routed_experts，但 prompt_logprobs_dict 仍不处理；若未来 routed experts 的高阶组合成为 SHM 视图，需要同步扩展。 - 影响：影响范围集中在 v1 执行路径：MRV2 开启 enable_return_routed_experts 后能稳定返回 routed experts 输出，MRV1 行为不变；ExecuteModelState 新增字段影响所有 MRV2 采样路径；Ray 编译 DAG 场景下 routed-experts 不再导致通道阻塞。性能验证（PR body）显示 R3 开启时 MRV1/MRV2 均无可重复回归。对团队而言，本 PR 确立了 R3 捕获逻辑的公共 API 形态，后续 runner 接入成本降低。 - 风险标记：核心路径变更，异步 D2H 竞态，行为契约 fail-fast，缺少 CI 端到端覆盖

关联脉络

- PR #38163 routed-experts capture（被代码注释引用）：vllm/config/vllm.py 中被删除的注释明确引用 https://github.com/vllm-project/vllm/pull/38163，推测为该 R3 捕获功能的引入 PR，也是本 PR 的前置依赖；标题未在本次材料中提供，编号与 URL 来自代码注释原文。
- PR #50823 [Bugfix] Shard UniformTypeKVCacheSpecs block table width under DCP: 与本 PR 同属 v1 KV cache group / attention 布局领域，修改对象与 get_routed_experts_attn_gid 所依赖的 kv_cache_groups 结构相邻，后续改动需关注交叉影响。
- PR #49389 [Misc] Remove deprecated calculate_kv_scales runtime KV scale calculation: 同属 v1 执行路径上 KV cache 相关能力的清理与收敛，与本 PR 的 attn_gid/block table 共享逻辑属于同一演进脉络。