PR #50940 完整报告

vllm-project/vllm

[R3] Unify routed expert shape configuration

合并时间：2026-08-05 11:20

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/50940

执行摘要

- 一句话：统一 routed experts 拓扑配置解析并修复 Kimi K3/Gemma 4
- 推荐动作：值得精读。该 PR 是一个教科书式的数据契约统一案例：将散落的别名解析收敛到单一 convertor，通过 ModelConfig 透传，删除重复实现。关注两点：一是 ModelArchConfigConvertorBase.get_num_experts_per_token() 的别名列表与 None 归一化策略，这是后续扩展新模型时需要维护的清单；二是 ModelArchitectureConfig 必填字段新增对构造方的影响模式，可作为未来新增规范字段的参考模板。

功能与动机

R3 维护了不完整的 model-config 解析，仅识别部分 experts-per-token 字段并单独重实现 expert-count 解析，导致 worker 与 scheduler 初始化偏离 vLLM 规范元数据。Kimi K3 使用 num_experts_per_token 因此初始化失败；Gemma 4 的配置没有 num_experts_per_tok 属性导致日志读取时抛 AttributeError。PR body 明确说明："This change removes the R3-specific expert-count parser and gives all R3 allocation and logging paths one validated (layers, experts, experts_per_token) source, with top-k normalized once by the model-architecture convertor."（#50460 为 Gemma 4 修复的原始来源，本 PR 将其作为首个 commit 保留原作者署名。）

实现拆解

1. 数据契约扩展：在 vllm/config/model_arch.py 的 ModelArchitectureConfig 中新增 num_experts_per_token: int 字段；在 vllm/transformers_utils/model_arch_config_convertor.py 新增 get_num_experts_per_token() 方法，依次尝试 num_experts_per_tok、num_experts_per_token、top_k_experts、moe_topk、moe_top_k 五种别名，并对 None 归一化为 0；convert() 中将其写入 ModelArchitectureConfig。
2. 配置访问器统一：vllm/config/model.py 的 ModelConfig 新增 get_num_experts_per_tok() 方法，直接返回 model_arch_config.num_experts_per_token，使所有调用方通过同一个规范字段读取。
3. R3 路径重构：vllm/model_executor/layers/fused_moe/routed_experts_capturer.py 删除自定义的 _get_num_experts_per_tok(hf_config) 与 get_num_experts(hf_config)，新增 _get_routed_experts_shape(vllm_config) 统一从 ModelConfig 获取 (num_layers, num_experts, num_experts_per_tok) 并校验为正数；RoutedExpertsCapturer 与 RoutedExpertsManager 初始化都改用该函数，同时修复了 manager 日志中直接访问

hf_config.num_experts_per_tok 的问题（即 Gemma 4 修复）。

4. FFN 性能配置复用：vllm/v1/metrics/perf.py 中的 parse 将原来对 ["num_experts_per_tok", "moe_topk"] 的 getattr_from_list 替换为 model_config.get_num_experts_per_tok()，复用同一 accessor。
5. 测试配套：tests/config/test_model_arch_config.py 新增参数化测试 test_num_experts_per_tok_aliases 覆盖五种别名，test_num_experts_per_tok_none_is_normalized 验证 None 归一化为 0；tests/model_executor/test_routed_experts_capture.py 新增 _make_model_config helper 与 test_routed_experts_manager_uses_gemma4_top_k_experts、test_routed_experts_manager_uses_kimi_k3_experts_per_token，分别用 top_k_experts 和 num_experts_per_token 驱动 RoutedExpertsManager 并断言 buffer shape 为 (max_num_slots, num_layers, top_k)。
6. 提交历史：首个 commit 是 #50460 的 Gemma 4 修复（作者 yihengz），后续 5 个 commit 逐步完成统一解析、别名支持与 None 归一化，演进方向清晰。

关键文件：

- vllm/model_executor/layers/fused_moe/routed_experts_capturer.py（模块 路由捕获；类别 source；类型 data-contract；符号 _get_routed_experts_shape, RoutedExpertsCapturer.init, RoutedExpertsManager.init）：R3 捕获链路的两个核心类（RoutedExpertsCapturer 与 RoutedExpertsManager）都改用统一的 _get_routed_experts_shape，删除自定义 parser 并修复 Gemma 4 日志缺陷。
- vllm/transformers_utils/model_arch_config_convertor.py（模块 配置转换；类别 source；类型 data-contract；符号 get_num_experts_per_token）：新增 get_num_experts_per_token() 并把结果写入 ModelArchitectureConfig，是本次统一的单一事实来源。
- tests/model_executor/test_routed_experts_capture.py（模块 路由捕获；类别 test；类型 test-coverage；符号 _make_model_config, test_routed_experts_manager_uses_gemma4_top_k_experts, test_routed_experts_manager_uses_kimi_k3_experts_per_token）：新增 Gemma 4（top_k_experts）与 Kimi K3（num_experts_per_token）两个回归测试，直接验证 manager 初始化与 buffer shape。

关键符号：get_num_experts_per_token, get_num_experts_per_tok, _get_routed_experts_shape

关键源码片段

vllm/model_executor/layers/fused_moe/routed_experts_capturer.py

R3 捕获链路的两个核心类（RoutedExpertsCapturer 与 RoutedExpertsManager）都改用统一的 _get_routed_experts_shape，删除自定义 parser 并修复 Gemma 4 日志缺陷。

```
# vllm/model_executor/layers/fused_moe/routed_experts_capturer.py
# 统一从 ModelConfig 读取 R3 需要的三个拓扑参数，替代原先各自从 hf_config 找别名的
# 两套 parser（_get_num_experts_per_tok 和 get_num_experts）。
# 这样 worker 端与 scheduler 端共享同一个规范数据源，避免命名差异导致的分歧。
def _get_routed_experts_shape(vllm_config: VllmConfig) -> tuple[int, int, int]:
```

```
    model_config = vllm_config.model_config
    num_layers = model_config.get_total_num_hidden_layers()
    num_experts = model_config.get_num_experts()
    # 该 accessor 直接返回 model_arch_config.num_experts_per_token,
    # 由 convertor 在模型配置加载时统一解析所有别名（如 top_k_experts）。
    num_experts_per_tok = model_config.get_num_experts_per_tok()
    if num_layers <= 0 or num_experts <= 0 or num_experts_per_tok <= 0:
        raise ValueError(
            "Routed-experts capture requires positive layer, expert, and "
            "experts-per-token counts, got "
            f"{num_layers=}, {num_experts=}, {num_experts_per_tok=}."
        )
    return num_layers, num_experts, num_experts_per_tok


class RoutedExpertsCapturer:
    def __init__(
        self,
        max_num_batched_tokens: int,
        vllm_config: VllmConfig,
        kv_cache_config: KVCacheConfig,
    ) -> None:
        # 只取 layers 与 per-token top-k；expert 总数由 manager 单独用于 dtype 选择。
        num_layers, _, num_experts_per_tok = _get_routed_experts_shape(vllm_config)
        logger.info(
            "RoutedExpertsCapturer: allocating buffer with "
            "max_tokens=%d, num_layers=%d, num_experts_per_tok=%d "
            "(hf_config.model_type=%s)",
            max_num_batched_tokens,
            num_layers,
            num_experts_per_tok,
            vllm_config.model_config.hf_text_config.model_type,
        )
        # 设备端 transit buffer 使用 int32，向上对齐 router 原生 topk_ids dtype,
        # 且 NCCL 对 uint8/uint16 支持不一，int32 通用性更好。
        self.device_buffer = torch.zeros(
            (max_num_batched_tokens, num_layers, num_experts_per_tok),
            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
        self.attn_gid = get_routed_experts_attn_gid(kv_cache_config)


class RoutedExpertsManager:
    def __init__(
        self,
        vllm_config: VllmConfig,
```

```
        kv_cache_config: KVCacheConfig,
    ) -> None:
        self.attn_gid = get_routed_experts_attn_gid(kv_cache_config)
        attn_group = kv_cache_config.kv_cache_groups[self.attn_gid]
        self.block_size = attn_group.kv_cache_spec.block_size

        # 同样走统一的 shape 入口；manager 还需要 num_experts 来决定 slot buffer
        # 的窄 dtype（专家数 <= 256 用 uint8，否则 uint16），以控制多 GB 缓冲占用。
        num_layers, num_experts, num_experts_per_tok = _get_routed_experts_shape(
            vllm_config
        )
        max_num_slots = kv_cache_config.num_blocks * self.block_size
        expert_id_dtype = np.uint8 if num_experts <= 256 else np.uint16
        self.routed_experts_by_slot = np.zeros(
            (max_num_slots, num_layers, num_experts_per_tok),
            dtype=expert_id_dtype,
        )
        logger.info(
            "RoutedExpertsManager CPU buffer: %.2f GB "
            "(slots=%d, layers=%d, top_k=%d, dtype=%s)",
            self.routed_experts_by_slot.nbytes / 1e9,
            max_num_slots,
            num_layers,
            num_experts_per_tok,
            self.routed_experts_by_slot.dtype.name,
        )
```

vllm/transformers_utils/model_arch_config_convertor.py

新增 get_num_experts_per_token() 并把结果写入 ModelArchitectureConfig，是本次统一的单一事实来源。

```
# vllm/transformers_utils/model_arch_config_convertor.py
# 在模型架构配置加载阶段统一解析“每 token 路由专家数”的命名差异。
# 不同模型家族用不同字段：
# - DeepSeek 系列：num_experts_per_tok
# - Kimi K3：num_experts_per_token
# - Gemma 4：top_k_experts
# - DBRX 风格：moe_topk / moe_top_k
class ModelArchConfigConvertorBase:
    def get_num_experts_per_token(self) -> int:
        names = [
            "num_experts_per_tok",
            "num_experts_per_token",
            "top_k_experts",
            "moe_topk",
            "moe_top_k",
        ]
        # getattr_iter 依次尝试；最后 or 0 把 None 归一化为 0，
        # 保证 ModelArchitectureConfig 拿到确定的 int 值，
```

```
        # 后续 R3 校验（<= 0 时报错）能显式暴露配置缺失。
        return getattr_iter(self.hf_text_config, names, 0) or 0

    def convert(self, supports_multimodal: bool = True) -> ModelArchitectureConfig:
        model_arch_config = ModelArchitectureConfig(
            architectures=self.get_architectures(),
            model_type=self.hf_config.model_type,
            text_model_type=getattr(self.hf_text_config, "model_type", None),
            hidden_size=self.get_hidden_size(),
            total_num_hidden_layers=self.get_num_hidden_layers(),
            total_num_attention_heads=self.get_total_num_attention_heads(),
            head_size=self.get_head_size(),
            vocab_size=self.get_vocab_size(),
            total_num_kv_heads=self.get_total_num_kv_heads(),
            num_experts=self.get_num_experts(),
            # 新增规范字段，与 num_experts 并列成为模型拓扑元数据的一部分。
            num_experts_per_token=self.get_num_experts_per_token(),
            quantization_config=self.get_quantization_config(),
            is_deepseek_mla=self.is_deepseek_mla(),
            is_mm_prefix_lm=self.is_mm_prefix_lm(supports_multimodal),
            rswa_window=self.rswa_window(),
            derived_max_model_len_and_key=self.derive_max_model_len_and_key(),
        )
        return model_arch_config
```

tests/model_executor/test_routed_experts_capture.py

新增 Gemma 4（top_k_experts）与 Kimi K3（num_experts_per_token）两个回归测试，直接验证 manager 初始化与 buffer shape。

```
# tests/model_executor/test_routed_experts_capture.py
# 构造一个瘦身的 ModelConfig 替身，复用真实的 convertor 解析逻辑，
# 让测试覆盖到“别名归一化 -> manager buffer shape”的完整链路。
def _make_model_config(hf_config):
    num_experts_per_token = ModelArchConfigConvertorBase(
        hf_config, hf_config
    ).get_num_experts_per_token()
    model_config = SimpleNamespace(
        hf_text_config=hf_config,
        model_arch_config=SimpleNamespace(
            num_experts_per_token=num_experts_per_token,
        ),
    )
    # 用真实 ModelConfig 的类方法绑定到替身上，保证行为一致。
    model_config.get_num_experts = lambda: hf_config.num_experts
    model_config.get_num_experts_per_tok = lambda: (
        ModelConfig.get_num_experts_per_tok(model_config)
    )
    model_config.get_total_num_hidden_layers = lambda: hf_config.num_hidden_layers
    return model_config
```

```
# Gemma 4 的配置没有 num_experts_per_tok，只有 top_k_experts；
# 回归用例确保 manager 初始化不再抛 AttributeError。
def test_routed_experts_manager_uses_gemma4_top_k_experts():
    hf_config = SimpleNamespace(
        num_experts=8,
        top_k_experts=2,
        num_hidden_layers=3,
    )
    vllm_config = SimpleNamespace(model_config=_make_model_config(hf_config))
    manager = RoutedExpertsManager(vllm_config, _make_kv_cache_config())
    # 期望 shape = (num_blocks * block_size, num_layers, top_k) = (8, 3, 2)
    assert manager.routed_experts_by_slot.shape == (8, 3, 2)


# Kimi K3 使用 num_experts_per_token 字段，覆盖该别名。
def test_routed_experts_manager_uses_kimi_k3_experts_per_token():
    hf_config = SimpleNamespace(
        num_experts=8,
        num_experts_per_token=2,
        num_hidden_layers=3,
    )
    vllm_config = SimpleNamespace(model_config=_make_model_config(hf_config))
    manager = RoutedExpertsManager(vllm_config, _make_kv_cache_config())
    assert manager.routed_experts_by_slot.shape == (8, 3, 2)
```

评论区精华

唯一的技术讨论发生在 review 阶段：Isotr0py 在 vllm/config/model.py 第 1441 行（get_num_experts_per_tok 方法）评论 "Should we move this to model_arch_config?"，作者 aoshen02 回复 "Nice catch!"。该评论对应的 diff hunk 显示早期版本是在 ModelConfig 内直接遍历 hf_text_config 的多个别名，而非从 model_arch_config 读取；后续提交将解析下沉到 ModelArchConfigConvertorBase.get_num_experts_per_token()，ModelConfig 仅做透传。这条讨论促成了数据契约的最终形态：解析逻辑收敛到 convertor，单一事实来源。

- get_num_experts_per_tok 是否应下沉到 model_arch_config (design): 作者接受建议（"Nice catch!"），并将别名解析逻辑移到 ModelArchConfigConvertorBase.get_num_experts_per_token()，ModelConfig.get_num_experts_per_tok() 最终只读取 model_arch_config.num_experts_per_token。

风险与影响

- 风险：
  1. 行为变更风险：get_num_experts_per_token() 对缺失属性返回 0 而非抛异常，R3 capturer 会因 num_experts_per_tok <= 0 触发 _get_routed_experts_shape 的 ValueError，错误信息更明确但启动失败路径从 AttributeError 变为 ValueError；依赖旧行为的调用方（如测试中的 SimpleNamespace 构造）需同步适配。

2. 兼容性风险：ModelArchitectureConfig 新增必填字段 num_experts_per_token，任何直接构造该 dataclass 的代码（如测试 _make_model_config 需要通过 convertor 填充）若未提供该字段会构造失败；好在新增字段无默认值，编译器 / 类型检查可提前暴露。
3. 别名覆盖不全风险：五种别名覆盖了 DeepSeek（num_experts_per_tok）、Kimi K3（num_experts_per_token）、Gemma 4（top_k_experts）、DBRX 系列（moe_topk/moe_top_k），但若未来模型使用其他命名（如 num_experts_per_head 之类）仍会漏掉；不过归一化为 0 后 R3 的校验会显式失败，不至于静默错误。
4. 验证局限：PR body 中的运行时验证是在容器内手工拷贝文件完成的，且只覆盖了 Gemma 4 冒烟与 Qwen3.6-35B GSM8K，未覆盖 Kimi K3 真实模型启动；Kimi K3 的验证仅靠单元测试模拟配置。 - 影响：影响范围集中在 R3 routed experts 捕获链路（worker 端 RoutedExpertsCapturer 与 scheduler 端 RoutedExpertsManager）以及 model arch 数据契约。用户侧：修复了 Gemma 4 与 Kimi K3 在 --enable-return-routed-experts 下的启动失败，MoE 模型路由追溯功能可覆盖更多模型家族。系统侧：ModelArchitectureConfig 新增字段影响所有模型配置路径，但 convertor 是统一入口，风险可控。团队侧：消除 R3 自定义 parser 是长期维护性改进，未来新 MoE 模型只需在 convertor 的别名列表扩展即可；perf.py 与 R3 共用同一 accessor，避免两处解析漂移。 - 风险标记：字段契约必填新增，别名覆盖有限，启动失败路径改变，真实 Kimi K3 未验证

关联脉络

- PR #50460 [Bugfix][Model] Fix Gemma4 routed-expert manager initialization: 本 PR 的首个 commit 直接携带了 #50460 的 Gemma 4 日志修复，并保留原作者 yihengz 的署名；#50460 是 #41401 的 follow-up。
- PR #50874 [Bugfix][R3] Size monolithic routing replay buffer for DP: 同涉 routed_experts_capturer.py 的 R3 修复，但 #50874 只处理 DP/EP replay buffer 布局，与本 PR 的 shape 配置改动互补，PR body 中明确声明不是重复。
- PR #41401 Routed experts capture (R3): R3 捕获功能的前置 PR，新增了归一化 expert-count 查找路径；本 PR 是它的配置统一收尾。
- PR #39067 Fix routed expert config naming in model loading: #50460 中提到 #39067 处理同一命名差异但位于 model-loading 路径；本 PR 统一后此类逐路径修补应不再需要。