PR #38939 完整报告

vllm-project/vllm

[R3] Add routed experts to openai entrypoint

合并时间：2026-05-21 00:08

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/38939

执行摘要

- 一句话：在 OpenAI 端点中添加 routed_experts 字段
- 推荐动作：值得关注的设计决策包括：使用 .npy + base64 作为序列化方案（相比 JSON 数组更紧凑），以及在协议模型中为字段添加详尽文档说明形状和空值条件。

功能与动机

延续 PR #28284 的 routed experts 功能，使其可通过标准 OpenAI API 获取专家路由信息，便于下游调试和分析。

实现拆解

1. 在 ChatCompletionResponseChoice 和 CompletionResponseChoice 协议类中添加 routed_experts: str | None 字段，存储 base64 编码的 numpy .npy 字节。
2. 在 chat_completion/serving.py 和 completion/serving.py 中导入 io、numpy 和 pybase64，在构建响应时检查 output.routed_experts，若非 None 则将其保存为 .npy 字节流并 base64 编码，赋值给 routed_experts 字段。
3. 新增两个测试文件：一个测试 /v1/completions 端点，一个测试 /inference/v1/generate 端点，验证字段存在、解码后数组维度与模型配置一致，且索引值在有效范围内。

关键文件：

- vllm/entrypoints/openai/chat_completion/protocol.py（模块 协议层；类别 source；类型 data-contract；符号 routed_experts）：定义 ChatCompletionResponseChoice 模型，新增 routed_experts 字段及其文档，是数据契约的核心变更。
- vllm/entrypoints/openai/chat_completion/serving.py（模块 服务层；类别 source；类型 core-logic）：实现 routed_experts 字段的实际装配逻辑，包括导入依赖、编码并赋值，是功能生效的关键路径。
- tests/entrypoints/openai/test_return_routed_experts.py（模块 测试；类别 test；类型 test-coverage；符号 server, test_routed_experts）：新增完整的集成测试，验证 /v1/completions 端点正确返回 routed_experts 字段，并断言解码后数组形状和值范围。
- tests/entrypoints/serve/disagg/test_return_routed_experts.py（模块 测试；类别 test；类型 test-coverage；符号 server, client, test_generate_routed_experts）：验证 disagg 端点 /inference/v1/generate 同样支持 routed_experts 返回，确保 tokens-in-tokens-out 路径覆盖。

- vllm/entrypoints/openai/completion/protocol.py（模块 协议层；类别 source；类型 data-contract；符号 routed_experts）：对应 CompletionResponseChoice 模型，添加相同的 routed_experts 字段，保持两端一致性。
- vllm/entrypoints/openai/completion/serving.py（模块 服务层；类别 source；类型 core-logic）：实现 completions 端点中 routed_experts 的编码逻辑，与 chat_completion 侧对称。

关键符号：chat_completion_full_generator, request_output_to_completion_response, test_generate_routed_experts, test_routed_experts

关键源码片段

vllm/entrypoints/openai/chat_completion/protocol.py

定义 ChatCompletionResponseChoice 模型，新增 routed_experts 字段及其文档，是数据契约的核心变更。

```
# vllm/entrypoints/openai/chat_completion/protocol.py

class ChatCompletionResponseChoice(OpenAIBaseModel):
    # ... existing fields ...

    # Per-token expert routing decisions, base64-encoded ``.npy`` bytes
    # (numpy serialization). Shape after decode:
    # (num_tokens - 1, num_layers, num_experts_per_tok) dtype uint8/uint16
    # ``num_tokens - 1`` because the last sampled token has not been
    # forwarded yet and therefore has no routing data.
    # Decode:
    # np.load(io.BytesIO(base64.b64decode(s)))
    # ``None`` if (a) the request was aborted before any forward pass,
    # or (b) ``enable_return_routed_experts`` is off server-side.
    routed_experts: str | None = None
```

vllm/entrypoints/openai/chat_completion/serving.py

实现 routed_experts 字段的实际装配逻辑，包括导入依赖、编码并赋值，是功能生效的关键路径。

```
# vllm/entrypoints/openai/chat_completion/serving.py
# 在 chat_completion_full_generator 函数中（约第 1090 行附近）

# Encode routed_experts for transport. JSON can't carry raw
# bytes, so we write the ndarray as a ``.npy`` byte stream
# and base64-encode it. ``pybase64`` is ~3x faster than the
# stdlib ``base64`` on large payloads thanks to SIMD.
routed_experts_b64 = None
if output.routed_experts is not None:
    buf = io.BytesIO()
    np.save(buf, output.routed_experts)
    routed_experts_b64 = base64.b64encode(buf.getvalue()).decode("ascii")
```

```
    choice_data = ChatCompletionResponseChoice(
        index=output.index,
        message=message,
        logprobs=logprobs,
        finish_reason=...,
        stop_reason=output.stop_reason,
        token_ids=(
            as_list(output.token_ids) if request.return_token_ids else None
        ),
        routed_experts=routed_experts_b64,
    )
```

tests/entrypoints/openai/test_return_routed_experts.py

新增完整的集成测试，验证 /v1/completions 端点正确返回 routed_experts 字段，并断言解码后数组形状和值范围。

```
# tests/entrypoints/openai/test_return_routed_experts.py

@pytest.mark.asyncio
async def test_routed_experts(server):
    """Test that /v1/completions returns routed_experts when enabled."""
    async with server.get_async_client() as client:
        result = await client.completions.create(
            model=MODEL_NAME,
            prompt="Hello, world",
            max_tokens=10,
            temperature=0,
            extra_body={"return_token_ids": True},
        )
        choice = result.model_dump()["choices"][0]
        assert choice["routed_experts"] is not None
        assert choice["token_ids"] is not None
        # routed_experts is base64-encoded .npy bytes; decode to ndarray.
        routed_experts = np.load(io.BytesIO(base64.b64decode(choice["routed_experts"])))
        assert routed_experts.ndim == 3
        num_tokens, num_layers, topk = routed_experts.shape
        assert num_tokens > 0
        assert num_layers == NUM_HIDDEN_LAYERS # 2
        assert topk == NUM_EXPERTS_PER_TOK # 2
        assert (routed_experts >= 0).all()
        assert (routed_experts < NUM_LOCAL_EXPERTS).all() # 8
```

评论区精华

SumanthRH 最初要求将修改也应用到 tokens-in-tokens-out 的 /inference/v1/generate 端点，并补充测试。后续 PR 添加了 disagg 端点的测试文件，SumanthRH 和 njhill 均批准合并。

- 支持 disagg /inference/v1/generate 端点 (design): PR 后续添加了 disagg 端点的测试文件，最终通过审批。

风险与影响

- 风险：base64 编码会增加响应大小，对长序列可能产生传输性能影响，但选用 pybase64（SIMD 加速）缓解这一点。新增字段向后兼容，不影响现有解析。无安全风险，但 base64 不提供加密保护。
- 影响：用户：可通过 OpenAI 兼容 API 获取 MoE 模型的专家路由信息。系统：增加少量计算和带宽开销。团队：功能扩展，需维护两套端点的数据路径。
- 风险标记：响应体积增大（base64），双端点维护成本

关联脉络

- PR #28284 Add routed experts support: 该 PR 首次引入 routed experts 内部功能，本 PR 将其暴露到 OpenAI 入口端点。