PR #41751 完整报告

vllm-project/vllm

[ROCm] mori: add InterNodeV1LL inter-node kernel selection via VLLM_MORI_INTERNODE_KERNEL

合并时间：2026-05-28 00:33

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41751

执行摘要

- 一句话：拆分 MoRI 后端为 mori_high_throughput / mori_low_latency
- 推荐动作：值得阅读，展示了如何在不引入新配置项的情况下扩展后端选择。设计上遵循现有模式，评审中对环境变量的取舍值得借鉴。

功能与动机

MoRI 内核提供了 InterNodeV1LL 低延迟变体，在 MI300X 多节点集群（RoCE v2）上 dispatch 带宽约为 InterNodeV1 的两倍。但 vLLM 在 _make_all2all_kwargs 中硬编码了 InterNodeV1，导致 InterNodeV1LL 完全不可用。本 PR 通过拆分后端名称暴露该选择。

实现拆解

1. 在 vllm/config/parallel.py 中将 All2AllBackend 字面类型从 "mori" 拆分为 "mori_high_throughput" 和 "mori_low_latency"，并更新文档字符串。
2. 在 vllm/distributed/device_communicators/all2all.py 中修改 MoriAll2AllManager，其 __init__ 接受 all2all_backend 参数；在 _make_all2all_kwargs 中根据后端选择 InterNodeV1（高吞吐）或 InterNodeV1LL（低延迟），同时保留单节点 IntraNode 内核不受影响。
3. 在 vllm/distributed/device_communicators/cuda_communicator.py 中，将原 "mori" 分支改为检测 "mori_high_throughput" 和 "mori_low_latency"，并将 all2all_backend 传入 MoriAll2AllManager。
4. 在 vllm/model_executor/layers/fused_moe/config.py 中更新 use_mori_kernels 属性，使其同时识别两个新后端。
5. 在 tests/kernels/moe/test_moe_layer.py 等测试文件中将 "mori" 替换为两个新后端，覆盖并行配置验证。

关键文件：

- vllm/distributed/device_communicators/all2all.py（模块 通信层；类别 source；类型 core-logic；符号 init）：核心变更，MoriAll2AllManager 接受 all2all_backend 参数，根据后端选择 InterNodeV1 或 InterNodeV1LL 内核
- vllm/distributed/device_communicators/cuda_communicator.py（模块 通信层；类别 source；类型 core-logic）：根据 all2all_backend 选择并实例化 MoriAll2AllManager 时

传入 backend 参数。

- vllm/model_executor/layers/fused_moe/config.py（模块 MoE 配置；类别 source；类型 data-contract）：更新 use_mori_kernels 属性以同时匹配两个后端名称。
- vllm/config/parallel.py（模块 并行配置；类别 source；类型 core-logic）：在 All2AllBackend 类型和文档中更新为两个 MoRI 后端名称。
- tests/kernels/moe/test_moe_layer.py（模块 MoE 测试；类别 test；类型 test-coverage）：测试后端列表和后端支持字典中增加两个新后端。
- tests/kernels/moe/modular_kernel_tools/mk_objects.py（模块 MoE 测试；类别 test；类型 test-coverage）：将测试对象中的 backend 从 'mori' 改为 'mori_high_throughput'。
- tests/kernels/moe/modular_kernel_tools/common.py（模块 MoE 测试；类别 test；类型 test-coverage）：同步修改公共测试工具中的后端名称。

关键符号：MoriAll2AllManager.init, MoriAll2AllManager._make_all2all_kwargs

关键源码片段

vllm/distributed/device_communicators/all2all.py

核心变更，MoriAll2AllManager 接受 all2all_backend 参数，根据后端选择 InterNodeV1 或 InterNodeV1LL 内核

```
    def _make_all2all_kwargs(
        self,
        rank: int,
        num_ep_ranks: int,
        input_dtype: torch.dtype,
        quant_dtype: torch.dtype,
        token_hidden_size: int,
        scale_dim: int,
        scale_type_size: int,
        max_num_tokens_per_dp_rank: int,
        num_local_experts: int,
        num_experts_per_token: int,
    ):
        import mori # type: ignore[import-not-found]
        from vllm.platforms.rocm import on_gfx942, on_gfx950

        assert on_gfx942() or on_gfx950(), (
            "mori currently only support arch gfx942 and gfx950"
        )

        if not self.internode:
            # 单节点：使用 IntraNode 内核
            kernel_type = mori.ops.EpDispatchCombineKernelType.IntraNode
            rdma_block_num = 0
            warp_num_per_block = 16
            block_num = 80
        else:
```

```
        # 多节点：根据 --all2all-backend 选择内核
        # mori_low_latency → InterNodeV1LL（低延迟，推荐）
        # mori_high_throughput → InterNodeV1（高吞吐，默认）
        if self._all2all_backend == "mori_low_latency":
            kernel_type = mori.ops.EpDispatchCombineKernelType.InterNodeV1LL
        else:
            kernel_type = mori.ops.EpDispatchCombineKernelType.InterNodeV1
        # 根据 GPU 架构选择 warp/block/rdma 配置
        if on_gfx942():
            warp_num_per_block = 16
            block_num = 32
            rdma_block_num = 16
        elif on_gfx950():
            warp_num_per_block = 8
            block_num = 64
            rdma_block_num = 32
        else:
            raise NotImplementedError(
                "mori currently only support arch gfx942 and gfx950"
            )

    return dict(
        rank=rank,
        world_size=num_ep_ranks,
        data_type=quant_dtype,
        hidden_dim=token_hidden_size,
        scale_dim=scale_dim,
        scale_type_size=scale_type_size,
        max_token_type_size=input_dtype.itemsize,
        max_num_inp_token_per_rank=max_num_tokens_per_dp_rank,
        num_experts_per_rank=num_local_experts,
        num_experts_per_token=num_experts_per_token,
        warp_num_per_block=warp_num_per_block,
        block_num=block_num,
        kernel_type=kernel_type,
        rdma_block_num=rdma_block_num,
        gpu_per_node=min(8, num_ep_ranks),
    )
```

vllm/distributed/device_communicators/cuda_communicator.py

根据 all2all_backend 选择并实例化 MoriAll2AllManager 时传入 backend 参数。

```
        # 处理 MoRI 后端：拆分为高吞吐和低延迟两个变体
        elif self.all2all_backend in (
            "mori_high_throughput",
            "mori_low_latency",
        ):
            from .all2all import MoriAll2AllManager

            self.all2all_manager = MoriAll2AllManager(
```

```
            self.cpu_group, self.all2all_backend # 传入后端名称
        )
```

评论区精华

最初的方案通过环境变量 VLLM_MORI_INTERNODE_KERNEL 选择内核。gemini-code-assist 建议使用 env_with_choices 进行验证。但 tjtanaa 提议遵循 deepep 的模式，直接拆分为两个后端名称，通过 --all2all-backend 选择。dllehr-amd 也要求放弃环境变量方案。最终采纳了拆分为 mori_high_throughput 和 mori_low_latency 的方案，并移除了环境变量。另在测试中，tjtanaa 建议不要保留 "mori" 别名，作者按要求移除。

- 环境变量方案 vs 后端名称方案 (design): 采用 mori_high_throughput 和 mori_low_latency 两个后端名称，删除环境变量。
- 测试中保留 mori 别名 (testing): 测试中只使用 mori_high_throughput 和 mori_low_latency。

风险与影响

- 风险：默认后端名从 mori 变为 mori_high_throughput，但内核选择与原来相同，故对现有用户无回归风险。涉及内核选择，性能差异可能较大，但属预期行为。测试覆盖已更新，CI 通过。低风险。
- 影响：对用户：需更新配置，原 --all2all-backend mori 需改为 mori_high_throughput 或 mori_low_latency。对系统：默认行为不变，性能无意外影响。对团队：统一了 MoRI 的命名规范与 deepep 一致。
- 风险标记：配置项重命名，依赖内核行为

关联脉络

- 暂无明显关联 PR