

PR #41753 完整报告

vllm-project/vllm

[ROCm] Add XGMI backend for MoRI Connector

合并时间: 2026-05-22 11:06

原文链接: <http://prhub.com.cn/vllm-project/vllm/pull/41753>

执行摘要

- 一句话: 为 MoRI 连接器添加 XGMI 后端, 支持单节点无 RDMA KV 传输
- 推荐动作: 建议阅读此 PR, 特别是 review 讨论中关于配置方式的设计选择。它还展示了如何在不破坏向后兼容的前提下扩展现有连接器后端。对于 AMD ROCm 平台开发者, 此 PR 是理解 MoRI 连接器架构的良好起点。同时, 建议后续关注关联的文档更新 PR 和环境变量迁移 PR, 以保持配置一致性。

功能与动机

该变更是为了实现在没有 RDMA 网卡的环境中运行 MoRI PD 分离的功能。关联 Issue #142 和 #35165 分别提出了需要 ROCm CI 和 TCP 支持的需求。XGMI 后端作为桥梁, 允许 CI 机器直接利用芯片间互联而不依赖外部 NIC, 降低了部署门槛。PR Body 中详细说明了在官方 ROCm Docker 镜像中缺少 RDMA 用户态库的问题, XGMI 后端可避免此类环境错误。

实现拆解

1. 配置模型扩展 ([moriio_common.py](#)): 在 MoRIIOConfig 数据类中新增 backend: str = "rdma" 字段; 在 from_vllm_config 工厂方法中从 kv_connector_extra_config 字典读取 backend 键, 默认 rdma, 并校验仅接受 rdma 或 xgmi, 无效时抛出 ValueError。
2. 后端初始化逻辑 ([moriio_engine.py](#)): 在 MoRIIOWrapper.set_backend_type 方法中根据传入的 backend_type 分别构建 XgmiBackendConfig (无需调优参数) 或 RdmaBackendConfig (使用环境变量中的 QP 数量、批大小、线程数)。这使后端配置与引擎创建解耦。
3. 连接器启动适配 ([moriio_connector.py](#)): 在 MoRIIOConnectorWorker.__init__ 中, 根据 moriio_config.backend 的值映射到 BackendType.XGMI 或 BackendType.RDMA, 并传递给 MoRIIOWrapper.set_backend_type。
4. 移除环境变量: 最初采用了 VLLM_MORIIO_BACKEND 环境变量, 但在 review 讨论后移除了该环境变量, 改用 kv_connector_extra_config, 保持了与 MoRI 其他配置一致的管理方式。后续 PR #43303 进一步将其他 MoRI 环境变量迁移至 extra_config。
5. 测试与验证: 作者提供了在单节点 8xMI300 上执行 1P1D 部署的完整 Docker 命令链, 包括路由、Prefill 和 Decode 实例, 并通过 vllm bench 和 GSM8k 任务验证了功能正确性。不过当前 PR 未包含自动化测试, 依赖手动验证。

关键文件:

- `vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_engine.py` (模块 KV 传输; 类别 source; 类型 core-logic; 符号 `MoRIIOWrapper.set_backend_type`) : 核心后端选择逻辑, 根据 `BackendType` 分别使用 `XgmiBackendConfig` 或 `RdmaBackendConfig`, 是实现 XGMI 后端的关键。
- `vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_common.py` (模块 KV 传输; 类别 source; 类型 core-logic; 符号 `MoRIIOConfig.backend`, `MoRIIOConfig.from_vllm_config`) : 配置数据模型变化, 新增 `backend` 字段, 并在工厂方法中从 `extra_config` 解析和校验后端类型。
- `vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_connector.py` (模块 KV 传输; 类别 source; 类型 core-logic; 符号 `MoRIIOConnectorWorker.init`) : 调用方根据配置选择 `BackendType` 枚举, 串联配置与初始化逻辑。

关键符号: `MoRIIOWrapper.set_backend_type`, `MoRIIOConfig.from_vllm_config`, `MoRIIOConnectorWorker.init`

关键源码片段

`vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_engine.py`

核心后端选择逻辑, 根据 `BackendType` 分别使用 `XgmiBackendConfig` 或 `RdmaBackendConfig`, 是实现 XGMI 后端的关键。

```
def set_backend_type(self, backend_type):
    assert self.moriio_engine is not None, "MoRIIO engine must be set first"
    if backend_type == BackendType.XGMI:
        # XGMI 后端使用无参数配置, 无需调优参数
        logger.info("Using MoRIIO backend: XGMI")
        self.moriio_engine.create_backend(backend_type, XgmiBackendConfig())
    else:
        # RDMA 后端需要从环境变量读取 QP 数量、批大小和工作者线程数
        logger.info(
            "Using MoRIIO backend: RDMA "
            "(qp_per_transfer=%d, post_batch_size=%d, num_workers=%d)",
            envs.VLLM_MORIIO_QP_PER_TRANSFER,
            envs.VLLM_MORIIO_POST_BATCH_SIZE,
            envs.VLLM_MORIIO_NUM_WORKERS,
        )
        rdma_cfg = RdmaBackendConfig(
            envs.VLLM_MORIIO_QP_PER_TRANSFER,
            envs.VLLM_MORIIO_POST_BATCH_SIZE,
            envs.VLLM_MORIIO_NUM_WORKERS,
            PollCqMode.POLLING,
        )
        self.moriio_engine.create_backend(backend_type, rdma_cfg)
```

`vllm/distributed/kv_transfer/kv_connector/v1/moriio/moriio_common.py`

配置数据模型变化, 新增 `backend` 字段, 并在工厂方法中从 `extra_config` 解析和校验后端类型。

```

@dataclass
class MoRIIOConfig:
    # ... 其他字段 ...
    backend: str = "rdma" # 后端类型, 可选 "rdma" 或 "xgmi"

    @classmethod
    def from_vllm_config(cls, vllm_config: VllmConfig) -> "MoRIIOConfig":
        # ... 前面的端口解析 ...
        backend = str(extra_config.get("backend", "rdma")).lower()
        # 校验后端类型合法性, 避免运行时错误
        if backend not in ("rdma", "xgmi"):
            raise ValueError(
                f"Invalid MoRIIO backend {backend!r} in kv_connector_extra_config; "
                "must be one of 'rdma' or 'xgmi'."
            )
        # ... 返回配置 ...
        return cls(
            # ... 其他字段 ...
            backend=backend,
        )

```

评论区精华

1. 配置方式设计争论: gemini-code-assist 建议使用 `env_with_choices` 环境变量。但 maintainer `tjtanaa` 指出“不应该再增加环境变量, 应该通过 `kv_connector_extra_config` 来选择后端”, 作者立即接受并修改了实现, 移除了环境变量。这体现了项目倾向于将连接器特定配置纳入统一的 `extra_config` 体系。
 2. 后续迁移承诺: 作者表示将后续将所有 MoRI 相关环境变量 (如 `VLLM_MORIIO_QP_PER_TRANSFER` 等) 迁移到 `extra_config`, 并已创建 PR #43303。
 3. 文档补全: `tjtanaa` 在批准时要求后续更新 `docs/features/disagg_prefill.md` 以提及 MoRI KV 连接器的配置, 作者确认将跟进。
- 后端选择配置方式: 环境变量 vs `extra_config` (design): 作者采纳建议, 移除了 `VLLM_MORIIO_BACKEND` 环境变量, 改为在 `extra_config` 中通过 `backend` 键设置。
 - 后续将其他 MoRI 环境变量迁移至 `extra_config` (design): 已跟踪到后续 PR, 本 PR 仅处理 `backend` 选择。
 - 文档更新需求 (documentation): 作者确认将跟进, 本 PR 暂不包含文档变更。

风险与影响

- 风险:
 - 配置误用: 若用户在 `extra_config` 中指定了 `backend` 为其他值, 会抛出 `ValueError`, 阻断启动, 但错误信息清晰, 风险可控。
 - XGMI 硬件要求: XGMI 后端要求所有 GPU 通过 AMD XGMI 互联且对每个实例可见 (通过 `HIP_VISIBLE_DEVICES` 暴露), 单节点多容器时需要 `--pid host`, 若环境不符合会导致 session 建立失败, 但错误发生在初始化阶段, 不会静默失败。

- 现有 RDMA 路径无影响：改动完全向后兼容，当 backend 未指定时默认为 rdma，RdmaBackendConfig 的参数读取方式与改前保持一致，不会回归。
- 缺少自动化测试覆盖：当前 PR 没有添加单元测试或集成测试，对 RDMA 回退路径的验证依赖手动测试，存在一定的回归风险。
- 影响：
 - 用户和系统：MoRI 连接器用户现在可以选择 XGMI 后端，适合单节点多 GPU 场景，降低了 RDMA 硬件依赖。这对 AMD 平台用户尤其有利，因为可以在官方 ROCm Docker 镜像中直接运行 PD 分离。
 - CI 基础设施：vLLM-router 项目的 CI 可以不再依赖 RDMA 网卡，利用 XGMI 后端开展单节点 e2e 测试，大幅降低 CI 资源需求。
 - 团队和开发：代码改动量小（仅 3 个文件，+34/-12），设计简洁，维护成本低。后续将环境变量统一到 extra_config 后，配置管理会更加一致。
 - 影响程度：中等偏低，仅影响 MoRI 连接器的一个可选后端路径，不影响其他 KV 连接器或模型加载。
 - 风险标记：缺少自动化测试覆盖，依赖 XGMI 硬件拓扑，配置错误在启动时即暴露

关联脉络

- PR #40453 [Misc] Add support for multi-node testing with MoRI: PR body 中提及多节点测试依赖此 PR，未来会利用 XGMI 后端在 CI 中添加多节点测试。
- PR #43303 [Misc][Refactor][ROCm] Convert MoRI-related envvars to extra config args: 作者在讨论中提及的后续 PR，将 MoRI 环境变量统一为 extra_config，与本 PR 的配置方式一脉相承。