PR #41313 完整报告

vllm-project/vllm

[ROCm][CI] Fix NIXL spec-decode acceptance startup and diagnostics

合并时间：2026-05-10 14:50

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/41313

执行摘要

- 一句话：修复 ROCm NIXL 猜测解码集成测试启动与诊断
- 推荐动作：该 PR 值得阅读，特别是测试框架中的异步服务就绪检查模式（进程存活与端点探测结合）值得参考。对于分布式测试的时序稳定性和错误诊断有良好示范。建议作者在后续 PR 中考虑 review 中关于异常类型和 IPv6 兼容性的反馈。

功能与动机

HTTP 就绪检查不足以判断 NIXL side-channel 是否可用，导致测试在 decode 阶段因 side-channel 未就绪而超时。需要精确探测 side-channel 元数据就绪状态，并确保所有网络流量（HTTP、代理、NIXL side-channel）使用同一 loopback 地址以避免 IPv4/IPv6 不匹配。

实现拆解

1. 新增 side-channel 探测工具（nixl_side_channel_probe.py）：通过 ZMQ REQ 套接字发送 get_meta_msg 并在超时前等待响应，判断 side-channel 是否真正就绪。
2. 修改集成测试脚本（spec_decode_acceptance_test.sh）：
   - 在 wait_for_server 函数中添加进程存活检查 (ps -p)，若目标进程已退出则立即失败，避免空轮询到超时。
   - 新增 wait_for_nixl_side_channel 函数，调用探测脚本来确认 side-channel 就绪。
   - 引入 SERVER_HOST 和 NIXL_SIDE_CHANNEL_HOST 变量，默认 127.0.0.1，确保所有 HTTP 和 ZMQ 连接使用同一地址族。
   - 在 run_test_for_device 中，启动每个实例后先等待其完成就绪再启动下一个，减少资源争抢。
3. 调整测试验证脚本（test_spec_decode_acceptance.py）：将所有硬编码 localhost 替换为 SERVER_HOST 环境变量，与 shell 脚本保持一致。
4. 增强 NIXL worker 诊断（worker.py）：
   - 改进日志输出，打印所有 attention backend 名称而非仅第一个。
   - 将 KV cache 块数不匹配的断言改为引发 AssertionError，并附带层名、形状、步幅、后端列表、KV 布局、拓扑模式等上下文，便于快速定位。
5. 更新 CI 配置（test-amd.yaml）：将 ROCM_ATTN=1 替换为 ATTENTION_BACKEND=ROCM_ATTN，使注意力后端选择与其他测试统一。

关键文件：

- tests/v1/kv_connector/nixl_integration/nixl_side_channel_probe.py（模块 侧信道探测；类别 test；类型 test-coverage；符号 parse_args, make_zmq_path, main）：新增 NIXL side-channel 探测工具，通过 ZMQ REQ 发送 get_meta_msg 并等待响应以确认信道就绪。
- vllm/distributed/kv_transfer/kv_connector/v1/nixl/worker.py（模块 分布式核心；类别 source；类型 core-logic）：增强 KV cache 注册时的断言信息，输出层名、形状、步幅、后端列表、布局和拓扑模式。
- tests/v1/kv_connector/nixl_integration/spec_decode_acceptance_test.sh（模块 集成测试；类别 test；类型 test-coverage）：增加服务进程存活检查，添加 NIXL side-channel 就绪等待函数，参数化主机地址。
- tests/v1/kv_connector/nixl_integration/test_spec_decode_acceptance.py（模块 验证测试；类别 test；类型 test-coverage）：将硬编码 localhost 替换为 SERVER_HOST 环境变量。
- .buildkite/test-amd.yaml（模块 CI 配置；类别 config；类型 configuration）：将 ROCM_ATTN=1 环境变量改为 ATTENTION_BACKEND=ROCM_ATTN。

关键符号：parse_args, make_zmq_path, main, wait_for_server, wait_for_nixl_side_channel

关键源码片段

tests/v1/kv_connector/nixl_integration/nixl_side_channel_probe.py

新增 NIXL side-channel 探测工具，通过 ZMQ REQ 发送 get_meta_msg 并等待响应以确认信道就绪。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
"""Probe a NIXL side-channel socket for handshake metadata readiness."""

import argparse
import ipaddress

import msgspec
import zmq

# 固定的消息内容
GET_META_MSG = b"get_meta_msg"


def parse_args() -> argparse.Namespace:
    parser = argparse.ArgumentParser()
    parser.add_argument("--host", required=True)
    parser.add_argument("--port", required=True, type=int)
    parser.add_argument("--rank", default=0, type=int)
    parser.add_argument("--timeout-ms", default=1000, type=int)
    return parser.parse_args()
```

```
def make_zmq_path(host: str, port: int) -> str:
    try:
        # 若 host 是 IPv6 地址，需用方括号括起来
        if isinstance(ipaddress.ip_address(host), ipaddress.IPv6Address):
            return f"tcp://[{host}]:{port}"
    except ValueError:
        pass
    return f"tcp://{host}:{port}"


def main() -> None:
    args = parse_args()
    ctx = zmq.Context()
    sock = ctx.socket(zmq.REQ)
    sock.setsockopt(zmq.LINGER, 0) # 关闭时立即丢弃未发送消息
    sock.setsockopt(zmq.RCVTIMEO, args.timeout_ms) # 接收超时
    try:
        sock.connect(make_zmq_path(args.host, args.port))
        sock.send(msgspec.msgpack.encode((GET_META_MSG, args.rank)))
        # 阻塞等待响应，若超时会抛出 zmq.Again 异常
        sock.recv()
    finally:
        sock.close()
        ctx.term()


if __name__ == "__main__":
    main()
```

vllm/distributed/kv_transfer/kv_connector/v1/nixl/worker.py

增强 KV cache 注册时的断言信息，输出层名、形状、步幅、后端列表、布局和拓扑模式。

```
# 在 register_kv_caches 方法中，当 cache.shape[0] != num_blocks 时，
# 原来简单的 assert 被替换为详细的 AssertionError，附带丰富的调试信息：
if cache.shape[0] != num_blocks:
    raise AssertionError(
        "All kv cache tensors must have the same number of "
        f"blocks; layer={layer_name}, "
        f"expected_num_blocks={num_blocks}, "
        f"cache_shape={tuple(cache.shape)}, "
        f"cache_stride={tuple(cache.stride())}, "
        f"layer_spec={type(layer_spec).__name__}, "
        f"backend={self.backend_name}, "
        "all_backends="
        f"{[backend.get_name() for backend in self.attn_backends]}, "
        f"kv_cache_layout={self.kv_cache_layout}, "
        "blocks_first="
        f"{self.transfer_topo.is_kv_layout_blocks_first}"
    )
```

tests/v1/kv_connector/nixl_integration/spec_decode_acceptance_test.sh

增加服务进程存活检查，添加 NIXL side-channel 就绪等待函数，参数化主机地址。

```
# 新增的 side-channel 等待函数
wait_for_nixl_side_channel() {
  local host=$1
  local port=$2
  local server_pid=$3
  local server_name=$4
  local deadline=120
  local elapsed=0
  echo "Waiting for ${server_name} NIXL side channel on ${host}:${port}..."
  while [ $elapsed -lt $deadline ]; do
    # 若服务进程已终止，则立即失败，避免空轮询
    if ! ps -p "$server_pid" > /dev/null 2>&1; then
      local status=0
      wait "$server_pid" || status=$?
      echo "FAIL: ${server_name} server process ${server_pid} exited with status ${status} before
      NIXL side channel ${host}:${port} became ready"
      exit 1
    fi
    # 调用 Python 探测脚本
    if python3 "${GIT_ROOT}/tests/v1/kv_connector/nixl_integration/nixl_side_channel_probe.py" \
      --host "$host" \
      --port "$port" \
      --timeout-ms 1000 > /dev/null 2>&1
    then
      echo "${server_name} NIXL side channel on ${host}:${port} ready"
      return 0
    fi
    sleep 2
    elapsed=$((elapsed + 2))
  done
  echo "FAIL: ${server_name} NIXL side channel ${host}:${port} did not start within ${deadline}
  s"
  exit 1
}
```

评论区精华

在 review 中，gemini-code-assist[bot] 提出了两点建议：

- 使用更具体的异常类型（如 RuntimeError）而非 AssertionError 来表示运行时配置不匹配，以区分内部逻辑错误。
- 在 shell 脚本的 curl 命令中处理 IPv6 地址，以避免未来当 SERVER_HOST 被设置为 IPv6 地址时 URL 格式错误。 目前这两个建议尚未被采纳或回复，建议作者后续考虑。
- 异常类型使用建议 (design): 当前仍保留了 AssertionError，但提供了更丰富的上下文。作者未回复该建议。

- IPv6 地址兼容性 (correctness): 评论未得到采纳或回复，当前脚本仍存在该潜在问题。

风险与影响

- 风险：主要风险包括：
  - 测试流程变更：wait_for_server 和 wait_for_nixl_side_channel 的新逻辑可能因进程状态检查的竞态条件导致误判，但已增加进程存在检测降低风险。
  - 新增依赖：zmq 和 msgspec 需在 CI 环境中预装，若缺失则探测脚本会失败。
  - IPv6 兼容性：shell 脚本中 curl 构造 URL 时未处理 IPv6 地址，若未来 SERVER_HOST 设为 IPv6 地址会导致 URL 格式错误。
  - 生产代码无风险：worker.py 的改动仅增强诊断，逻辑等价。
  - 影响：影响范围仅限于 ROCm CI 中的 NIXL 猜测解码集成测试步骤（v1/kv_connector/nixl_integration/spec_decode_acceptance_test.sh）。该测试用于验证分布式猜测解码（PD+SD）的接受长度匹配基线。变更后测试启动更健壮，诊断信息更丰富，有助于快速定位 side-channel 和 KV cache 相关失败。生产代码无影响。
  - 风险标记：测试流程变更，新增依赖（zmq, msgspec），IPv6 未充分处理

关联脉络

- PR #41269 [Bugfix][KV Transfer][NIXL] Notify P node on pre-admission rejection to free stranded KV blocks: 同样涉及 NIXL 连接器在 CI 中的稳定性，修复了提前拒绝时 KV 块滞留问题，与本 PR 的 CI 测试改进互为补充。
- PR #41366 [KV Offload] Pass ReqContext to touch(), complete_load(), and complete_store(): 统一了 OffloadingManager 接口，与本 PR 中 NIXL worker 的诊断改进都属于 KV 传输层的基础建设。