PR #46844 完整报告

vllm-project/vllm

[CI] Mooncake PD integration tests

合并时间：2026-08-03 21:55

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46844

执行摘要

- 一句话：为 Mooncake PD 引入端到端集成测试与独立 CI 任务
- 推荐动作：值得精读，尤其是 run_accuracy_test.sh 中 RDMA 设备白名单的注释、.buildkite/test_areas/disaggregated_mooncake.yaml 里池选择的理由、以及 install-kv-connectors.sh 的 CUDA 13 wheel 替换逻辑——这三处都是真实 CI 环境踩坑后的产物，对想要把自有 P/D 或 RDMA 传输测试接入 CI 的团队有直接参考价值。整体评分定位在 " 有意义的测试基建 " 而非核心功能变更。

功能与动机

PR body 明确说明出发点："Let's start tracking MC functionalities *e2e* with proper integration tests, starting with basic PD functionality"。作者把 Buildkite 任务拆分为独立组，是因为 Mooncake 的 *feature matrix compatibility* 尚未达到 NixlConnector 的成熟度，需要能单独开关、独立演进并节省 CI 开销。Issue 评论记录了前置条件：等待 CUDA 13 预编译 wheel 发布（"Waiting on cu13 precompiled package to ship"），以及 RDMA 依赖 CI 节点分配的问题（"Having some issues on RDMA depending on node assignment"），这些直接塑造了本 PR 的池选择与 wheel 替换设计。

实现拆解

实现分为五步：

1. 新增测试包与精度断言。tests/v1/kv_connector/mooncake_integration/__init__.py 使目录成为可导入的测试包；test_accuracy.py 定义 run_simple_prompt()（经 OpenAI 兼容接口向本地 8192 端口代理发一个续写请求做冒烟验证）与 test_accuracy()（用 lm_eval 的 local-completions 后端跑 gsm8k，以 exact_match,strict-match 指标与 EXPECTED_VALUES 中 Qwen/Qwen3-0.6B 的 0.41 基线做 ±5% 容差断言）。未登记的模型只告警不失败，避免 CI 误报。
2. 编排脚本 run_accuracy_test.sh。全部参数通过环境变量注入：NUM_PREFILL_INSTANCES、NUM_DECODE_INSTANCES、PREFILLER_TP_SIZE、DECODER_TP_SIZE、GPU_MEMORY_UTILIZATION、BLOCK_SIZE、VLLM_SERVE_EXTRA_ARGS、ATTENTION_BACKEND。kv_config() 生成 kv_producer/kv_consumer 两份 --kv-transfer-config；端口规划为 prefill 8100+、decode 8200+、bootstrap 8998+，GPU 按实例序号轮转分配。所有实例 wait_for_server 就绪后，复用官方 mooncake_connector_proxy.py 聚合出 8192 端口入

口，最后执行 pytest，并由 cleanup_instances() 清理残留 vllm serve 进程。MOONCAKE_DEVICE_NAME 默认 mlx5_12 是对 b200-k8s 池多 HCA 拓扑踩坑后的白名单方案。

3. 配置扫描 config_sweep_accuracy_test.sh。先执行 test_mooncake_imports.py 作为 import canary，再用 env + bash 数组方式安全传参（避免 eval），依次扫 GPU_MEMORY_UTILIZATION=0.6 下 TP 1x1 与 2x2 两种对称配置；支持 ATTENTION_BACKEND 透传，任一配置失败即 exit 1。
4. 独立 Buildkite 任务组。.buildkite/test_areas/disaggregated_mooncake.yaml 新增 "Disaggregated Mooncake" 组，device: b200-k8s、num_devices: 4、timeout_in_minutes: 10。选择该池的唯一理由是它能承载 Mooncake 传输：只有它授予 IPC_LOCK（RDMA 钉页必需）并使用 host networking；其他池经拓扑发现找不到可用 HCA，tcp fallback 会广告 docker bridge 地址导致连接全部失败。source_file_dependencies 限定仅 Mooncake 连接器源码与集成测试目录变更时触发。
5. 依赖与安装脚本配套。requirements/kv_connectors.txt 把 mooncake-transfer-engine 提升到 >= 0.3.12 并注明 CUDA 12/13 双 wheel 关系；install-kv-connectors.sh 的元数据探测从只查 nixl 扩展为同时输出 mooncake 版本，并在 CUDA 13 镜像上先卸载 CUDA 12 wheel 再安装同版本 mooncake-transfer-engine-cuda13；test_mooncake_imports.py 从包元数据层提前暴露 wheel 不匹配，避免引擎启动才出现 "Mooncake is not available"。docs/features/mooncake_connector_usage.md 补充了 device_name 白名单参数文档。

关键文件：

- tests/v1/kv_connector/mooncake_integration/test_accuracy.py（模块 精度测试；类别 test；类型 test-coverage；符号 run_simple_prompt, test_accuracy）：端到端精度断言的核心测试文件：先做冒烟请求，再用 lm_eval 跑 gsm8k 与模型基线做 ±5% 容差比对，是整条 P/D 链路正确性的最终验收点。
- tests/v1/kv_connector/mooncake_integration/run_accuracy_test.sh（模块 编排脚本；类别 test；类型 test-coverage）：编排核心：负责按环境变量拉起多组 prefill/decode vllm 实例、配置 Mooncake bootstrap 与 kv-transfer-config、等待就绪后启动聚合代理并触发 pytest；脚本内 RDMA 设备白名单注释记录了 b200-k8s 池的关键踩坑。
- tests/v1/kv_connector/mooncake_integration/config_sweep_accuracy_test.sh（模块 配置扫描；类别 test；类型 test-coverage）：配置扫描入口：先跑 import canary，再用 env + 数组方式依次扫 TP 1x1/2x2 两组配置，是 Buildkite 任务直接调用的脚本。
- tests/v1/kv_connector/mooncake_integration/test_mooncake_imports.py（模块 导入检查；类别 test；类型 test-coverage；符号 test_mooncake_engine_imports）：import canary：在 wheel 安装阶段提前暴露 mooncake CUDA 变体不匹配，避免引擎启动才报 "Mooncake is not available"。
- .buildkite/test_areas/disaggregated_mooncake.yaml（模块 CI 任务；类别 config；类型 configuration）：新增独立 Buildkite 组，锁定 b200-k8s 池（IPC_LOCK + host networking）；source_file_dependencies 限定触发范围，实现按需开关与节省 CI。
- .buildkite/scripts/install-kv-connectors.sh（模块 安装脚本；类别 other；类型 core-logic）：CI 基础设施关键改动：扩展元数据探测到 mooncake，并在 CUDA 13 镜像上自动把

CUDA 12 wheel 替换为 cuda13 变体，否则引擎启动即链接失败。

- tests/v1/kv_connector/mooncake_integration/__init__.py（模块 测试包；类别 test；类型 test-coverage）：使 mooncake_integration 目录成为可导入的测试包。
- requirements/kv_connectors.txt（模块 依赖清单；类别 docs；类型 documentation）：提升 mooncake-transfer-engine 到 >= 0.3.12 并注明 CUDA 12/13 双 wheel 的安装约定。
- docs/features/mooncake_connector_usage.md（模块 使用文档；类别 docs；类型 documentation）：补充 device_name 参数说明，给出多 HCA 拓扑下链路层选择问题的用户侧配置指引。

关键符号：run_simple_prompt, test_accuracy, test_mooncake_engine_imports, run_tests_for_model, run_tests, kv_config, wait_for_server, cleanup_instances

关键源码片段

tests/v1/kv_connector/mooncake_integration/test_accuracy.py

端到端精度断言的核心测试文件：先做冒烟请求，再用 lm_eval 跑 gsm8k 与模型基线做 ±5% 容差比对，是整条 P/D 链路正确性的最终验收点。

```
# SPDX-License-Identifier: Apache-2.0
# SPDX-FileCopyrightText: Copyright contributors to the vLLM project
import os

import lm_eval
import openai

BASE_URL = "http://localhost:8192/v1"
NUM_CONCURRENT = 100 # lm_eval 并发请求数
TASK = "gsm8k"
FILTER = "exact_match,strict-match"
RTOL = 0.05 # 相对容差：允许精度波动 5%

# 模型对应的期望精度基线，来自本地 benchmark 实测。
EXPECTED_VALUES = {
    "Qwen/Qwen3-0.6B": 0.41,
}

SIMPLE_PROMPT = (
    "The best part about working on vLLM is that I got to meet so many people across "
    "various different organizations like UCB, Google, and Meta which means",
)

# 通过环境变量覆盖被测模型，默认 Qwen/Qwen3-0.6B。
MODEL_NAME = os.environ.get("TEST_MODEL", "Qwen/Qwen3-0.6B")


def run_simple_prompt():
    """冒烟请求：确认 P/D 两侧经 Mooncake 转移 KV 后仍能正常续写。"""
    client = openai.OpenAI(api_key="EMPTY", base_url=BASE_URL)
```

```
    completion = client.completions.create(model=MODEL_NAME, prompt=SIMPLE_PROMPT)

    print("-" * 50)
    print(f"Completion results for {MODEL_NAME}:")
    print(completion)
    print("-" * 50)


def test_accuracy():
    """端到端精度测试：经聚合代理跑 gsm8k，与基线精度比对。"""
    run_simple_prompt()

    model_args = (
        f"model={MODEL_NAME},"
        f"base_url={BASE_URL}/completions,"
        f"num_concurrent={NUM_CONCURRENT},tokenized_requests=False"
    )
    results = lm_eval.simple_evaluate(
        model="local-completions",
        model_args=model_args,
        tasks=TASK,
    )

    measured_value = results["results"][TASK][FILTER]
    expected_value = EXPECTED_VALUES.get(MODEL_NAME)

    print(f"Measured accuracy value: {measured_value}\n")
    if expected_value is None:
        # 未登记的模型只输出精度、不做硬断言，便于 CI 之外手动验证。
        print(
            f"Warning: No expected value found for {MODEL_NAME}. "
            "Skipping accuracy check."
        )
        return

    # 容差判断：measured 需落在 expected ± RTOL 区间内，防止精度回归。
    assert (
        measured_value - RTOL < expected_value
        and measured_value + RTOL > expected_value
    ), f"Expected: {expected_value} | Measured: {measured_value}"
```

tests/v1/kv_connector/mooncake_integration/run_accuracy_test.sh

编排核心：负责按环境变量拉起多组 prefill/decode vllm 实例、配置 Mooncake bootstrap 与 kv-transfer-config、等待就绪后启动聚合代理并触发 pytest；脚本内 RDMA 设备白名单注释记录了 b200-k8s 池的关键踩坑。

```
#!/bin/bash
set -xe
```

```
# ---- 配置来源：环境变量，全部带默认值 ----
NUM_PREFILL_INSTANCES=${NUM_PREFILL_INSTANCES:-1} # prefill 实例数
NUM_DECODE_INSTANCES=${NUM_DECODE_INSTANCES:-1} # decode 实例数
PREFILLER_TP_SIZE=${PREFILLER_TP_SIZE:-1}
DECODER_TP_SIZE=${DECODER_TP_SIZE:-1}
GPU_MEMORY_UTILIZATION=${GPU_MEMORY_UTILIZATION:-0.2}
BLOCK_SIZE=${BLOCK_SIZE:-128}

# ---- Mooncake 设备发现 ----
# Mooncake 通过 bootstrap server 发现对端，不需要 side-channel 布局变量。
# MOONCAKE_DEVICE_NAME 是拓扑发现的 HCA 白名单；默认值针对 b200-k8s CI 池：
# 该池节点暴露 4 个 InfiniBand HCA 但只有一个 RoCE 口（mlx5_12），若全部 HCA
# 可见，P/D 两侧可能选中不同链路层，QP 握手无法到达 RTR。其他硬件需调整该值。
MOONCAKE_DEVICE_NAME=${MOONCAKE_DEVICE_NAME-mlx5_12}

# producer（prefill 侧）与 consumer（decode 侧）共用配置模板，仅 kv_role 不同。
kv_config() {
  printf '{"kv_connector":"MooncakeConnector","kv_role":"%s",' "$1"
  printf '"kv_connector_extra_config":{"device_name":"%s"}}' "${MOONCAKE_DEVICE_NAME}"
}
KV_CONFIG_P=$(kv_config kv_producer)
KV_CONFIG_D=$(kv_config kv_consumer)

run_tests_for_model() {
  local model_name=$1

  # 启动 prefill 实例：GPU 按实例序号轮转，TP > 1 时追加相邻 GPU。
  for i in $(seq 0 $((NUM_PREFILL_INSTANCES - 1))); do
    GPU_ID=$((i % $(get_num_gpus)))
    for ((j = 1; j < PREFILLER_TP_SIZE; j++)); do
      GPU_ID="${GPU_ID},$(((GPU_ID + j) % $(get_num_gpus)))"
    done

    # 端口规划：prefill 8100+ / decode 8200+ / bootstrap 8998+，避免互相冲突。
    PORT=$((8100 + i))
    BOOTSTRAP_PORT=$((8998 + i))

    eval "CUDA_VISIBLE_DEVICES=$GPU_ID \
      VLLM_MOONCAKE_BOOTSTRAP_PORT=$BOOTSTRAP_PORT \
      vllm serve $model_name \
      --port $PORT --enforce-eager \
      --block-size ${BLOCK_SIZE} \
      --gpu-memory-utilization $GPU_MEMORY_UTILIZATION \
      --tensor-parallel-size $PREFILLER_TP_SIZE \
      --kv-transfer-config '$KV_CONFIG_P' &"

    PROXY_ARGS+=(--prefill "http://localhost:${PORT}" "$BOOTSTRAP_PORT")
  done
```

```
  # decode 实例与 prefill 对称：端口 8200+、kv_role=kv_consumer。
  for i in $(seq 0 $((NUM_DECODE_INSTANCES - 1))); do
    GPU_ID=$(((i + NEXT_GPU + 1) % $(get_num_gpus)))
    for ((j = 1; j < DECODER_TP_SIZE; j++)); do
      GPU_ID="${GPU_ID},$(((GPU_ID + j) % $(get_num_gpus)))"
    done
    PORT=$((8200 + i))

    eval "CUDA_VISIBLE_DEVICES=$GPU_ID \
      vllm serve $model_name \
      --port $PORT --enforce-eager \
      --block-size ${BLOCK_SIZE} \
      --gpu-memory-utilization $GPU_MEMORY_UTILIZATION \
      --tensor-parallel-size $DECODER_TP_SIZE \
      --kv-transfer-config '$KV_CONFIG_D' &"

    PROXY_ARGS+=(--decode "http://localhost:${PORT}")
  done

  # 所有 P/D 实例就绪后再启动聚合代理，统一暴露 8192 端口给下游测试。
  for PORT in "${PREFILL_PORTS[@]}" "${DECODE_PORTS[@]}"; do
    wait_for_server "$PORT" || return 1
  done

  python3 "${GIT_ROOT}/examples/disaggregated/mooncake_connector/mooncake_connector_
  proxy.py" \
    --port 8192 "${PROXY_ARGS[@]}" &
  sleep 5

  # 精度断言在 pytest 中执行：跑 gsm8k 并与模型基线做容差比较。
  TEST_MODEL=$model_name python3 -m pytest -s -x \
    "${GIT_ROOT}/tests/v1/kv_connector/mooncake_integration/test_accuracy.py"

  # 每个模型跑完后清理残留 vllm serve 进程，避免端口与显存互相干扰。
  cleanup_instances
}
```

.buildkite/scripts/install-kv-connectors.sh

CI 基础设施关键改动：扩展元数据探测到 mooncake，并在 CUDA 13 镜像上自动把 CUDA 12 wheel 替换为 cuda13 变体，否则引擎启动即链接失败。

```
# ---- 收集已安装 KV connector 组件版本（nixl + mooncake）----
# 依 torch.version.cuda 判断 CUDA 主版本，用于后续 wheel 变体选择。
KV_METADATA=$(python3 - <<'PY'
import importlib.metadata as metadata
import torch

cuda_version = torch.version.cuda
if cuda_version is None:
```

```
    raise SystemExit("torch.version.cuda is not set")

try:
    mooncake_version = metadata.version("mooncake-transfer-engine")
except metadata.PackageNotFoundError:
    mooncake_version = ""

print(cuda_version.split(".", 1)[0], metadata.version("nixl"), mooncake_version)
PY
)
read -r CUDA_MAJOR NIXL_VERSION MOONCAKE_VERSION <<< "${KV_METADATA}"
MOONCAKE_VERSION="${MOONCAKE_VERSION:-}"

# ---- Mooncake CUDA 12/13 wheel 替换 ----
# PyPI 默认的 mooncake-transfer-engine wheel 基于 CUDA 12 构建，engine.so 链接
# libcudart.so.12，而 CUDA 13 运行镜像没有该库；mismatch 会以 ImportError
# 在引擎启动前暴露。因此 CUDA 13 镜像上先卸载默认 wheel，再装同版本 cuda13
# 变体——两者都提供 `mooncake` 包，必须先卸载避免包冲突。
if [ "${CUDA_MAJOR}" = "13" ] && [ -n "${MOONCAKE_VERSION}" ]; then
  uv pip uninstall --system mooncake-transfer-engine 2>/dev/null || true
  uv pip install --system "mooncake-transfer-engine-cuda13==${MOONCAKE_VERSION}"
fi
```

评论区精华

zhewenl 作为主要 reviewer 提出两点：一是质疑测试目录里新增的 toy_proxy_server.py 与示例代理重复（"Why couldn't we just reuse the example proxy?"），NickLucche 承认 "I overfitted on nixl setup lol"，最终版本删除了 toy 代理、直接复用 examples/disaggregated/mooncake_connector/mooncake_connector_proxy.py；二是建议补充 SWA 模型覆盖（"maybe https://huggingface.co/google/gemma-3-1b-it will be a good fit"），NickLucche 回应这是 "a PR to get things started"，当前重点是打通基础设施（linking errors 等），不同模型配置可仿照 nixl 的做法后续按模型类型补充，该建议目前仍是开放项。ivanium 附议 "Overall LGTM and agree with @zhewenl's comments"，zhewenl 最终 APPROVED。

- 是否复用示例代理而非自建 toy proxy (design): NickLucche 承认 "I overfitted on nixl setup lol"（照搬了 nixl 套件的结构），最终版本删除了 toy_proxy_server.py，改为直接调用 examples/disaggregated/mooncake_connector/mooncake_connector_proxy.py。
- 是否补充 SWA（滑动窗口注意力）模型覆盖 (testing): NickLucche 回应 "this is a PR to get things started"，当前优先打通基础设施（处理 linking errors 等），不同模型类型的配置可仿照 nixl 的做法在后续 PR 按模型补充；该建议作为后续工作记录，未在本 PR 实现。

风险与影响

- 风险：风险集中在四方面：一是 测试稳定性依赖节点分配——任务锁定 b200-k8s 池，RDMA 对 IPC_LOCK 与 host networking 是硬性依赖，作者在 Issue 评论中也提到节点分配导致的 RDMA 问题；MOONCAKE_DEVICE_NAME=mlx5_12 是强环境耦合，换池或换

网卡都会直接失败。二是 固定精度基线可能误报或漏报——Qwen/Qwen3-0.6B 的 0.41 基线来自本地 benchmark，模型或推理配置变化可能触发误报，而未登记模型只告警不断言又可能掩盖真实回归。三是 CUDA wheel 替换可能静默失败——install-kv-connectors.sh 在 CUDA 13 镜像上先卸载再装 cuda13 变体，卸载用 || true 容错；若变体版本缺失，import canary 只能发现 mooncake.engine 导入问题，无法覆盖 libcudart 运行时链接层面的错配。四是 超时窗口偏紧——Job 超时 10 分钟，但脚本要启动多实例、wait_for_server 单次最长 1200 秒并执行两轮配置扫描，节点启动慢或模型下载慢时容易整组超时。

- 影响：对用户与系统无运行时影响（零源码主路径改动），影响集中在 CI 基础设施与 kv-connector 测试矩阵：Mooncake PD 首次获得端到端回归保护，且任务组可独立开关、仅在相关文件变更时触发，能节省 CI 开销。对团队而言，后续加模型、加 SWA 或更多 TP 配置只需在 sweep 脚本追加条目；该套脚本与 buildkite 配置也可作为其他 KV connector（如 nixl）之外新连接器的接入模板。影响程度中等，价值主要在工程保障层面。
- 风险标记：RDMA 依赖特定 CI 节点类型，固定精度基线可能误报，CUDA wheel 替换可能静默失败，Job 超时窗口偏紧

关联脉络

- PR #50266 [CI] KimiLinear PD in nightlies: 同属 kv_connector 下的 P/D 分解集成测试族：本 PR 在 tests/v1/kv_connector/mooncake_integration/ 的实现明确镜像了 nixl_integration 套件的结构（commit message 中写明 mirroring the existing nixl_integration suite），两者共同构成 KV connector 的 CI 覆盖矩阵。
- PR #49069 [Bugfix][KV Connector] Propagate EAGLE state across merged Mooncake store groups: 同属 Mooncake 功能线：该 PR 修复 Mooncake store 组的 EAGLE 状态传播问题，本 PR 为 Mooncake 的 P/D 传输路径补上端到端回归测试，一个修行为、一个补防线，共同推动 Mooncake 在 vLLM 中的成熟度。