PR #46893 完整报告

vllm-project/vllm

[CI] GSM8K eval integration test for KV offloading

合并时间：2026-07-09 05:59

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46893

执行摘要

- 一句话：GSM8K KV 卸载回归测试，覆盖四大架构族
- 推荐动作：该 PR 设计扎实，测试覆盖四大架构和两种卸载连接器，且包含同步屏障和精度断言。建议精读 _reset_gpu_prefix_cache 的轮询策略和 test_gsm8k_offloading_correctness 的双评估逻辑。对于 KV 卸载模块的开发者，此测试应纳入日常开发流程。

功能与动机

PR body 指出该测试用于防止步长计算错误（例如 PR#46888）和静默 KV 缓存数据损坏。评论中 tlrmchlsmth 提到“连 KV 卸载的冒烟测试都没有，这很尴尬”，强调了测试覆盖的缺失。

实现拆解

步骤一：新增核心测试文件

在 tests/evals/gsm8k/test_gsm8k_offloading.py 中实现测试框架，包含辅助函数 _kv_transfer_config（生成连接器配置 JSON）、_force_engine_step（强制引擎步进以完成异步传输）、_reset_gpu_prefix_cache（通过轮询 reset_prefix_cache 端点实现卸载同步屏障）。定义 OffloadingModelConfig 数据类和测试矩阵 MODELS。

步骤二：双重 GSM8K 评估逻辑

测试函数 test_gsm8k_offloading_correctness 启动服务器，对每个模型运行两轮 GSM8K 评估（200 题、5-shot）。第一轮结束后调用 _reset_gpu_prefix_cache 清空 GPU 缓存（保留 CPU 缓存），第二轮依赖从 CPU 重载的 KV 缓存。若重载导致数据损坏，精度将低于阈值（baseline - 0.05），从而触发断言。

步骤三：修改 API 端点返回成功状态

修改 vllm/entrypoints/serve/dev/cache/api_router.py（Python）和 rust/src/server/src/routes/cache.rs（Rust）的 reset_prefix_cache 端点，从原来返回空 200 状态码改为返回包含 "success": bool 的 JSON 响应。这使得客户端（测试）可以重试直到成功。同时更新了 Rust 路由测试 rust/src/server/src/routes/tests.rs 以匹配新返回格式。

步骤四：添加 Buildkite CI 配置

在.buildkite/test_areas/lm_eval.yaml中新增三个CI步骤（kv-offload-small/medium/large），分别使用 1xH200（30min）、2xH100（60min）、4xH100（60min）运行不同规模的模型（通过 -k 参数选择）。步骤依赖卸载连接器、KV 卸载模块和测试文件，自动触发。

关键文件：

- tests/evals/gsm8k/test_gsm8k_offloading.py（模块 GSM8K 测试；类别 test；类型 test-coverage；符号 _kv_transfer_config, _force_engine_step, _reset_gpu_prefix_cache, OffloadingModelConfig）：核心测试文件，覆盖四大架构、两种连接器，包含卸载同步和精度断言。
- vllm/entrypoints/serve/dev/cache/api_router.py（模块 API 层；类别 source；类型 entrypoint；符号 reset_prefix_cache）：修改 Python API 路由，使 reset_prefix_cache 返回成功状态，与 Rust 保持一致。
- rust/src/server/src/routes/cache.rs（模块 Rust 路由；类别 source；类型 entrypoint；符号 reset_prefix_cache）：Rust API 同步修改以返回 success 字段，保持与 Python API 一致。
- rust/src/server/src/routes/tests.rs（模块 Rust 测试；类别 test；类型 test-coverage）：更新 Rust 路由测试以匹配新的 JSON 响应格式。
- .buildkite/test_areas/lm_eval.yaml（模块 CI 配置；类别 config；类型 configuration）：添加三个 CI 步骤，自动触发卸载相关测试，覆盖不同 GPU 规格。

关键符号：_kv_transfer_config, _force_engine_step, _reset_gpu_prefix_cache, OffloadingModelConfig, test_gsm8k_offloading_correctness, reset_prefix_cache (Python), reset_prefix_cache (Rust)

关键源码片段

tests/evals/gsm8k/test_gsm8k_offloading.py

核心测试文件，覆盖四大架构、两种连接器，包含卸载同步和精度断言。

```
def _reset_gpu_prefix_cache(base_url: str) -> None:
    """Drop the GPU prefix cache while keeping the CPU (connector) cache, so
    the next run must reload KV data through the connector.

    The reset fails while asynchronous offload transfers still hold GPU
    blocks, so retry until it succeeds.  Requires VLLM_SERVER_DEV_MODE=1.
    """
    deadline = time.monotonic() + _OFFLOAD_SYNC_TIMEOUT
    while True:
        # 发送 POST /reset_prefix_cache?reset_external=false
        resp = requests.post(
            f"{base_url}/reset_prefix_cache",
            params={"reset_external": "false"},
            timeout=30,
        )
        resp.raise_for_status()
        if resp.json().get("success"):
```

```
            # 重置成功，返回
            return
        # 如果未成功（可能是因为异步卸载仍在进行），强制引擎步进并重试
        assert time.monotonic() < deadline, (
            f"prefix cache reset did not succeed within {_OFFLOAD_SYNC_TIMEOUT}s; "
            "async offload may be stuck"
        )
        _force_engine_step(base_url)
```

vllm/entrypoints/serve/dev/cache/api_router.py

修改 Python API 路由，使 reset_prefix_cache 返回成功状态，与 Rust 保持一致。

```
@router.post("/reset_prefix_cache")
async def reset_prefix_cache(
    raw_request: Request,
    reset_running_requests: bool = Query(default=False),
    reset_external: bool = Query(default=False),
):
    """
    Reset the local prefix cache.

    Optionally, if the query parameter `reset_external=true`
    also resets the external (connector-managed) prefix cache.

    Returns `{"success": bool}`. The reset fails (`success=false`) while
    blocks are still held, e.g. by running requests or in-flight async KV
    offload transfers; callers may retry.

    Example:
      POST /reset_prefix_cache?reset_external=true
    """
    logger.info("Resetting prefix cache...")
    # 调用引擎客户端并获取布尔结果
    success = await engine_client(raw_request).reset_prefix_cache(
        reset_running_requests, reset_external
    )
    # 返回包含 success 字段的 JSON 响应
    return JSONResponse(content={"success": bool(success)})
```

评论区精华

- orozery提出三点设计问题： 1) SimpleCPUOffload 需启用 prefix cache； 2) 异步卸载需 KVEvents 确保完成； 3) 断言 external_prefix_cache_hits。决策：采用轮询 prefix cache reset 作为同步屏障（而非 KVEvents），且只断言精度不断言命中，以保持简单并涵盖 simple CPU offload。
  - ZJY0516建议添加 Qwen3.5 模型。已采纳并扩充测试矩阵。
  - AndreasKaratzas建议 CI 标签统一格式为 "LM Eval KV-Offload (NxHxxx)"。已采纳。
  - tlrmchlsmth强调 Python API 应与 Rust 保持一致，均返回 success 字段。已实现。

- 测试设计：使用 prefix cache reset 和 KVEvents (design): 采用了轮询 prefix cache reset 的方式，未使用 KVEvents，断言精度而非命中。
- 添加 Qwen3.5 覆盖 (testing): 添加了 Qwen3.5 到测试矩阵。
- CI 标签命名 (style): 采用了建议的标签格式。
- Python API 与 Rust API 一致性 (design): 修改了 Python 端点返回 JSONResponse。
- Etelis 询问接管修改 (question): tlrmchlsmth 已推送更新。

风险与影响

- 风险：
  - 测试漏报风险：如果卸载被静默跳过（如卸载未启用），精度可能仍符合阈值，导致假阴性。测试注释已说明这一点，但未直接修复。
  - API 兼容性：reset_prefix_cache 端点从空响应改为 JSON 响应，现有客户端若未处理可能出错。但该端点在标记为 dev 模式，影响面小。
  - CI 资源消耗：新增三个 CI 步骤，可能延长流水线，但仅在卸载相关文件变更时触发。
  - 时序依赖：卸载同步采用轮询，极端情况下可能超时（60 秒），但已有断言处理。
- 影响：
  - 用户 / 功能：仅影响使用 KV 卸载开发模式（VLLM_SERVER_DEV_MODE=1）的用户，reset_prefix_cache 现返回成功布尔值。
  - 系统：新增自动化 CI 测试覆盖 KV 卸载场景，提升回归防御能力。
  - 团队：为 KV 卸载开发提供可复现的精度基线，减少手动验证开销。
  - 风险标记：异步卸载时序依赖，API 兼容性，测试可能漏报，新增 CI 资源

关联脉络

- PR #46888 stride computation bug fix in offloading worker: 该 PR 修复的步长计算错误触发了本测试的回归保护需求。
- PR #47762 Fix KV offloading GSM8K eval: prefix caching, CPU reload verification, device fit: 该 PR 完善了本测试中 prefix caching 与 reload verification 逻辑。
- PR #47823 Simplify offload-completion barrier: poll prefix cache reset instead of KV events: 该 PR 简化了卸载完成屏障，采用轮询 prefix cache reset 而非 KVEvents。