PR #1606 完整报告

radixark/miles

ci(rocm): add ROCm CI workflow for MI300X self-hosted runners

合并时间：2026-08-06 08:03

原文链接：http://prhub.com.cn/radixark/miles/pull/1606

执行摘要

- 一句话：新增 ROCm CI 工作流，MI300X 双分片覆盖
- 推荐动作：推荐基础设施负责人与 AMD 平台维护者精读。值得学习的设计决策：自托管 runner 的安全授权模型（pull_request_target + 标签 + secret 门控）、双 runner 分片与 est_time 均衡、8-GPU 用例降维时的轴选择论证（保留 TP/PP 分片、牺牲 CP/DP），以及用字符串断言测试锁定 workflow 合约的做法。对普通功能开发者，该 PR 确立的测试注册模式（register_rocm_ci + test_amd_* 变体）是新测试需要遵循的路径。

功能与动机

PR body 明确说明这是 _run-ci.yml 的 ROCm 对应物：『Uses the ROCm container with HIP/CUDA compatibility to run existing CUDA-registered test suites』，目标是让 AMD MI300X 硬件参与 CI 回归。guapisolo 的关键约束有二：一是 per PR CI 时长（『I am worried that this per PR CI will run too long』），并圈定 8 个必测文件；二是 runner 拓扑（『spliting into two 4xgpu runner might make more sense. But need to confirm cpu memory is enough』），直接催生了 4-GPU 套件与独立变体设计。

实现拆解

实现按 5 步演进，与 8 个 commit 一一对应：

1. 搭建 ROCm workflow 骨架：新增 .github/workflows/_run-ci-rocm.yml（175 行）定义 workflow_call 输入、runs-on 数组解析、ROCm 容器选项（kfd/dri 设备、render/video 组、特权模式、宿主机 /data/miles_ci 挂载）、Ray 进程清理与 rocm-smi GPU 就绪探测；skip_dependency_install 默认 true（镜像内置 SGLang/Megatron），依赖安装分支支持按 PR body 的 ci-megatron-pr/ci-sglang-pr 解析引用。新增 .github/workflows/pr-test-rocm.yml 镜像 resolve-ci-image，把 ci_image_tag 拼为 rocm/sgl-dev 镜像名并校验 Docker tag 格式。
2. 修复与兼容：第二个 commit 修正未声明的 workflow_call inputs（ci_megatron_pr/ci_sglang_pr 从未被消费，直接删除，避免『Invalid input』解析失败），并砍掉无意义的 CPU 路径；随后新增 tests/ci/run_with_patch.py，为旧容器中的 aiter 提供 shuffle_scale 兼容 shim（回退到 shuffle_scale_a16w4）。
3. 评审驱动的结构重调：阶段从 8-GPU 改为 stage-c-4-gpu-mi300x 套件（run_suite.py 的 CI_SUITES 增加 ROCm backend），runs_on 标签改为 ['self-hosted', 'amd', '

mi300x', '4gpu']（活 runner 携带 amd 标签而非 rocm）。测试选择改为显式注册：5 个原 4-GPU CUDA 测试直接追加 register_rocm_ci()（test_qwen3_4B_fsdp_hybrid_shard_r2s2、test_qwen3_4B_offload_disk_stream、test_glm5_2_744b_a40b_5layer_ci、test_inkling_small_4layer_ci、test_deepseek_v4_flash_4layer_ci），3 个原 8-GPU 测试各建独立 test_amd_*.py 变体（test_amd_mtp1_spec_v2_r3、test_amd_deepep_fp8_bridge、test_amd_r3_mtp），降维轴选择明确：TP/PP保持与CUDA相同分片、CP作为可牺牲轴（仅影响长上下文吞吐）、MI300X 192GB HBM 支撑 4 卡容纳原 8 卡负载。

4. 双 runner 分片：stage 增加 partition_id: [0, 1] 矩阵与 --auto-partition-size 2，run_suite.py 按 est_time 均衡（4500s/4800s），fail-fast: false 保证单分片失败不互杀；墙钟从约 2.6h 降至约 1.3h。
5. 安全门控与合约锁定：最后两个 commit 启用 pull_request_target 门控（fork PR 需 maintainer 添加 run-ci-* 标签才放行自托管执行），显式 checkout PR merge ref，fork 任务不注入 WANDB_API_KEY；tests/ci/test/test_run_suite.py 新增 TestRocmWorkflowScopeSeam，以字符串断言锁定 policy 共享、标签授权、分片命令、secret 门控等行为；tests/ci/labels.py 登记 run-ci-amd 标签，docs/ci/00-stage.md 同步说明 MI300X 双分片路径与外部 SGLang MI350 nightly 的边界。

关键文件：

- .github/workflows/_run-ci-rocm.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：ROCm 可复用 workflow 的核心：容器设备暴露、GPU 就绪探测、依赖安装分支、执行命令传递，全部 CI 能力在此承载
- .github/workflows/pr-test-rocm.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：分发 workflow：pull_request_target 门控、run-ci-* 标签授权、镜像解析、双分片矩阵，安全模型的核心载体
- tests/ci/test/test_run_suite.py（模块 CI 测试；类别 test；类型 test-coverage；符号 TestRocmWorkflowScopeSeam, test_pr_nightly_and_dispatch_share_policy, test_stage_consumes_policy_and_preserves_manual_full_scope）：TestRocmWorkflowScopeSeam 以字符串断言锁定 workflow 合约，防止后续演进破坏安全门控与分片行为
- tests/e2e/megatron/test_qwen3_5_35B_A3B_mtp/test_amd_mtp1_spec_v2_r3.py（模块 回归测试；类别 test；类型 test-coverage）：8-GPU 用例降维为 4-GPU 独立变体的代表：TP/PP 分片不变、CP=1 避 heavy kernel、MI300X 192GB HBM 余量论证
- tests/e2e/megatron/test_qwen3_30B_A3B/test_amd_deepep_fp8_bridge.py（模块 回归测试；类别 test；类型 test-coverage）：DeepEP + FP8 rollout + bridge 组合在 4-GPU 上的变体，max_tokens_per_gpu 沿用 CUDA 侧 2048（由宿主机内存决定）
- tests/e2e/megatron/test_glm47_flash/test_amd_r3_mtp.py（模块 回归测试；类别 test；类型 test-coverage）：8-GPU 用例降维的另一种方式：只牺牲 CP 轴（cp 2 -> 1），TP/PP 不变保证 per-rank 分片与 CUDA 一致
- tests/ci/run_with_patch.py（模块 兼容垫片；类别 test；类型 test-coverage；符号 shuffle_scale）：为旧 ROCm 容器中的 aiter 提供 shuffle_scale 兼容 shim，避免 sglang 导入即崩，是 AMD 容器版本差异的实测产物

- tests/e2e/megatron/model_scripts/test_deepseek_v4_flash_4layer_ci.py（模块 回归测试；类别 test；类型 test-coverage）：原 4-GPU CUDA 用例直接追加 register_rocm_ci() 的代表，验证『4-GPU 原用例直接注册』的评审决策
- tests/ci/labels.py（模块 CI 标签；类别 test；类型 test-coverage）：登记 run-ci-amd 标签，是 PR/nightly 触发 ROCm CI 的授权标识，评审明确要求
- docs/ci/00-stage.md（模块 CI 文档；类别 docs；类型 documentation）：CI 舞台文档同步双分片 MI300X 路径与外部 SGLang MI350 nightly 覆盖边界，是团队操作的入口文档

关键符号：TestRocmWorkflowScopeSeam.test_pr_nightly_and_dispatch_share_policy, TestRocmWorkflowScopeSeam.test_stage_consumes_policy_and_preserves_manual_full_scope, shuffle_scale

关键源码片段

tests/e2e/megatron/test_qwen3_5_35B_A3B_mtp/test_amd_mtp1_spec_v2_r3.py

8-GPU 用例降维为 4-GPU 独立变体的代表：TP/PP 分片不变、CP=1 避 heavy kernel、MI300X 192GB HBM 余量论证

```
"""AMD 4-GPU variant of test_mtp1_spec_v2_r3.py.

MI300X 机器拆成两个 4-GPU runner，原 8-GPU CUDA 用例无法照搬；
独立变体使两侧并行度互不约束。
"""

import os

from tests.ci.ci_register import register_rocm_ci
from tests.ci.metric_history import register_ci_gate
from tests.e2e.megatron.test_qwen3_5_35B_A3B_mtp._common import CaseConfig, execute, prepare

# 注册到 ROCm 4-GPU 专用套件，labels 带 amd 便于 run-ci-amd 精确触发
register_rocm_ci(
    est_time=1600,
    suite="stage-c-4-gpu-mi300x",
    labels=["megatron", "qwen35", "amd"],
)

# 与 CUDA 变体一致的指标门禁，保证训练可收敛
register_ci_gate(metric_key="train/grad_norm")
register_ci_gate(metric_key="train/ppo_kl")
register_ci_gate(metric_key="train/train_rollout_logprob_abs_diff")
register_ci_gate(metric_key="train/train_rollout_kl")
register_ci_gate(metric_key="rollout/raw_reward")

CASE = CaseConfig(
```

```
    num_gpus_per_node=4,
    cp_size=1, # CP 降到 1：避免 GatedDeltaNet CP backward 的大显存 kernel
    pp_size=2, # PP=2 减半驻留层数
    tp_size=2, # TP 保持 2：TP=4 会命中 Qwen3.5 attention-output-gate sharding bug
    ep_size=4,
    rollout_num_gpus_per_engine=4,
    sglang_ep_size=4,
    enable_mtp_training=True,
    use_r3=True,
    # miles 训练侧没有 VLM/vision 实现，vision 权重永不同步，跳过权重一致性检查
    check_weight_update_skip_list=("visual",),
)


if __name__ == "__main__":
    for proxy_var in ("http_proxy", "https_proxy", "HTTP_PROXY", "HTTPS_PROXY"):
        os.environ.pop(proxy_var, None)
    prepare(CASE)
    execute(CASE, wandb_file=__file__)
```

tests/e2e/megatron/test_glm47_flash/test_amd_r3_mtp.py

8-GPU 用例降维的另一种方式：只牺牲 CP 轴（cp 2 -> 1），TP/PP 不变保证 per-rank 分片与 CUDA 一致

```
"""AMD 4-GPU variant of test_r3_mtp.py.

降维策略：cp_size 2 -> 1，world size 从 8 减到 4。
TP=2 与 PP=2 保持不变，每个 rank 的分片与 CUDA 原例一致，只放弃数据并行；
CP 是安全的牺牲轴 —— 它只服务长上下文吞吐，不影响 R3 + MTP 行为。
"""

import os

from tests.ci.ci_register import register_rocm_ci
from tests.ci.metric_history import register_ci_gate
from tests.e2e.megatron.test_glm47_flash._common import CaseConfig, execute, prepare

register_rocm_ci(
    est_time=1100,
    suite="stage-c-4-gpu-mi300x",
    labels=["megatron", "amd"],
)

# 指标门禁与 CUDA 原例一致
register_ci_gate(metric_key="train/grad_norm")
register_ci_gate(metric_key="train/ppo_kl")
register_ci_gate(metric_key="train/train_rollout_logprob_abs_diff")
register_ci_gate(metric_key="train/train_rollout_kl")
register_ci_gate(metric_key="rollout/raw_reward")
```

```
CASE = CaseConfig(
    use_deepep=False,
    num_gpus_per_node=4,
    cp_size=1,
    pp_size=2,
    tp_size=2,
    ep_size=4,
    # GLM-4.7-Flash 只有 20 个 attention head；非 EP 的 SGLang TP 必须整除它
    rollout_num_gpus_per_engine=4,
)


if __name__ == "__main__":
    for proxy_var in ("http_proxy", "https_proxy", "HTTP_PROXY", "HTTPS_PROXY"):
        os.environ.pop(proxy_var, None)
    prepare(CASE)
    execute(CASE, wandb_file=__file__)
```

tests/ci/run_with_patch.py

为旧 ROCm 容器中的 aiter 提供 shuffle_scale 兼容 shim，避免 sglang 导入即崩，是 AMD 容器版本差异的实测产物

```
#!/usr/bin/env python3
"""Run a miles test with aiter shuffle_scale monkey-patch for older containers."""

import sys

# 在 anything imports from sglang 之前打上 aiter.ops.shuffle 补丁
import aiter.ops.shuffle as _shuffle

if not hasattr(_shuffle, "shuffle_scale"):
    import functools

    @functools.wraps(_shuffle.shuffle_scale_a16w4)
    def shuffle_scale(src, experts_cnt=None, is_guinterleave=False, gate_up=False):
        """Compatibility shim: older aiter has shuffle_scale_a16w4 only."""
        # 老接口没有 is_guinterleave 参数，直接转发到 a16w4 实现
        return _shuffle.shuffle_scale_a16w4(src, experts_cnt, gate_up)

    _shuffle.shuffle_scale = shuffle_scale
    print("[patch] Added shuffle_scale compatibility shim to aiter.ops.shuffle", flush=True)

# 执行目标测试文件：把测试文件路径当作脚本主体运行
test_file = sys.argv[1]
sys.argv = [test_file]
exec(open(test_file).read())
```

评论区精华

核心交锋围绕时长与 runner 拓扑展开：

> guapisolo：『I am worried that this per PR CI will run too long』

随后他圈定 8 个必测文件并给出总时长预算（『about 9300s in total. Other parts can be throwed』），作者回应『Happy to adjust as per your feedback』。

> guapisolo：『iirc, AMD GPU has very large hbm. spliting into two 4xgpu runner might make more sense. But need to confirm cpu memory is enough』
>
> guapisolo（对 8-GPU 用例降级方式）：『If it's originally 4 gpus, then directly set register_rocm_ci(). But if they're original 8 gpu. plz start a standalone amd test file. like test_amd_ and don't hardcode the original test case with IS_HIP etc』
>
> guapisolo（_run-ci-rocm.yml 评论）：『As we have 2 amd ci runner. We'd better put them into 2 shards』，作者回复『done』并实现矩阵分片。

最终 approve 时仍保留提醒：『approve to unblock. need amd folks check nightly CI』。

- per PR CI 运行时长预算 (performance): 按 guapisolo 圈定的 8 个文件执行，总时长约 9300s；其余候选测试被剔除。
- runner 拆分为两个 4-GPU runner 与 CPU 内存确认 (design): 决定一台机器跑两个 4-GPU runner，后续 stage 与测试全部按 4-GPU 重排；分片矩阵让两个 runner 并行工作。
- 8-GPU 用例的独立变体策略（禁止 IS_HIP 硬编码） (design): 落地为 3 个独立 test_amd_* 文件，降维轴选择各有论证：TP/PP 分片保持、CP/DP 作为牺牲轴。
- 两个 runner 分片到两个 shard (design): stage 增加 partition_id: [0, 1] 矩阵 + --auto-partition-size 2，run_suite.py 按 est_time 均衡，墙钟从 2.6h 降至 1.3h。
- run-ci-amd 标签与 amd labels (testing): tests/ci/labels.py 登记 run-ci-amd，新增 / 修改的 AMD 测试全部带 amd 标签。

风险与影响

- 风险：
  - 自托管环境安全：pull_request_target 让 fork PR 代码在特权容器 + 宿主机 /data/miles_ci 挂载上运行，唯一门控是 run-ci-* 标签，一旦 maintainer 误加标签即放行任意代码；WANDB_API_KEY 已对 fork 隐藏，但宿主机挂载仍暴露。
  - 测试覆盖窄：首批仅 8 个文件约 9300s，相对 CUDA 全套覆盖面有限；AMD 专属缺陷（如 aiter shuffle_scale 缺失）可能到上线后才暴露。
  - 降维等价性风险：test_amd_* 在 4-GPU 上跑 8-GPU 用例，CP 从 2 降 1、DP 减半，行为等价性依赖『TP/PP 分片不变』假设；test_amd_mtp1_spec_v2_r3 中 TP=4 触发 Qwen3.5 attention-output-gate sharding bug 的避让记录在注释里，若上游修复该 bug，变体文件不会自动跟进。
  - aiter shim 是临时补丁：run_with_patch.py 依赖老容器缺 shuffle_scale 的现状，容器镜像更新后 shim 可能不再被触发或反而掩盖真实行为。
  - 分片均衡依赖 est_time 估算：估算不准会导致两个 runner 负载不均，拉长整体墙钟。
- 影响：

- 系统：新增 AMD MI300X CI 回归面，PR/nightly 双通道，与既有 CUDA 管线平级，行为由 TestRocmWorkflowScopeSeam 测试锁定；tests/ci/ci_register、labels.py、run_suite.py 扩为双硬件后端。
- 团队：AMD 侧回归从手动 / 外部覆盖进入主仓 CI，后续 AMD 相关 PR 可自动触发 run-ci-amd 验证；guapisolo 明确需要 AMD 同事跟进 nightly。
- 用户：无最终功能影响；对使用 AMD 硬件的 RL 训练用户，间接获得回归保障。
- 后续：MI350 套件由外部 SGLang ROCm 7.2 nightly workflow 覆盖，与 MI300X 形成分层，为 AMD 平台持续演进铺路。
- 风险标记：自托管 runner 特权容器安全暴露，pull_request_target 标签门控，首批覆盖仅 8 个测试文件，aiter shuffle_scale 临时兼容 shim，分片均衡依赖 est_time 估算

关联脉络

- PR #2139 [AMD] retune AMD 4-node dsv4 config: 同为 AMD 平台支持线，DSv4 配方调优为 ROCm 上的真实负载验证提供上下文
- PR #2031 feat(fsdp): add hybrid sharding: 本 PR 将 test_qwen3_4B_fsdp_hybrid_shard_r2s2 注册进 ROCm 套件，FSDP 混合分片成为 AMD CI 的首批覆盖项
- PR #1793 feat(optimizer): NVMe optimizer-state streaming as a miles plugin: test_qwen3_4B_offload_disk_stream 被本 PR 注册进 ROCm 套件，offload 能力获得 AMD 回归保障
- PR #1571 GLM-5.2 kernel fix and GB300 training config: test_glm5_2_744b_a40b_5layer_ci 被本 PR 注册进 ROCm 套件，GLM 系列在 AMD 上获得回归覆盖
- PR #2131 ci(docker): rebuild scheduled images at least once every 24h: ROCm 镜像依赖 docker 构建管线，24h 陈旧回退保证 CI 容器不长期腐化