PR #47030 完整报告

vllm-project/vllm

[ROCm][DistInf] Enable vLLM DI CI with buildkite/slurm

合并时间：2026-08-11 02:39

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/47030

执行摘要

- 一句话：新增 AMD ROCm 分体式推理 CI 套件，覆盖 TP/EP 模式与精度门禁
- 推荐动作：这个 PR 值得精读，尤其是对 CI 工程和多节点分布式推理感兴趣的技术管理者。它展示了如何用 SLURM + Buildkite 搭建跨节点 GPU 测试流水线，并引入了阶段感知轮询、preflight 检查、共享文件系统完成哨兵等实用设计。关注点应放在：脚本与特定集群的耦合程度、安全挂载问题、以及如何平衡 nightly 测试的覆盖面和资源成本。

功能与动机

PR body 明确指出："This PR enables the Buildkite CI scirpts that exercises vLLM Disaggregated inference P/D with MoRI IO KV connector on ROCm AMD devices. This brings up a full PD topology health-gates on everyserver, runs the GSM8k accuracy." 目的是让 AMD 设备上的分布式推理（prefill/decode 分离）在 nightly CI 中得到持续验证，防止功能回归，并覆盖 TP 与 EP 两种并行模式。

实现拆解

1. 新增核心启动器.buildkite/amd-disagg/vllm_disagg.sh：这是整个套件的枢纽，支持 node / proxy / prefill / decode / bench / accuracy 等角色。它解析参数、加载 cluster.sh 与 models.yaml、解析拓扑（xP prefill + yD decode）、启动 MoRIIO 代理、健康检查每个 server，并在 rank 0 上运行 bench/accuracy 工作负载，最终通过共享文件系统写完成哨兵文件，让所有 rank 统一退出。
2. 新增集群配置模板.buildkite/amd-disagg/cluster.sh：集中管理站点相关的拓扑、端口、RDMA/NCCL 环境变量、MoRI 相关设置（如 MORI_RDMA_DEVICES、MORI_SHMEM_HEAP_SIZE）、路由器与代理地址。所有变量均支持 ${VAR:-default} 覆盖，便于在不同集群间复用。
3. 新增模型目录.buildkite/amd-disagg/models.yaml：每个模型条目定义 AITER 环境变量和 prefill/decode 在不同模式（tp/ep）下的 vLLM 启动标志，并明确将拓扑 / 模式结构参数交给启动器管理，避免重复。
4. 新增 SLURM 提交器.buildkite/amd-disagg/run-slurm-disagg-test.sh：作为 Buildkite 步骤的唯一入口，将脚本暂存到共享目录、调用 sbatch 提交作业，并支持 WAIT=0（fire-and-forget，Spur 集群安全）与 WAIT=1（阶段感知轮询，通过 scontrol 和 NFS 日志检测作业状态，返回通过 / 失败）。

5. 新增 SLURM 作业主体与 Buildkite 流水线.buildkite/amd-disagg/run_xPyD_disagg.slurm 和 .buildkite/amd-disagg/pipeline-disagg.yaml：前者负责节点选择、SLURM 环境变量修正、节点 preflight（GPU 显存检查）、容器启动与端口转发；后者定义了面向 MI350 的测试矩阵（1P1D TP8、2P2D TP8，各模型 × proxy/vllm-router 两种路由器）。
6. 测试与配套：本 PR 无单元测试，但 CI 套件本身即测试载体。支持的测试结果已在 PR body 中给出（GSM8k：1P1D TP8 0.947，1P1D DP8 EP 0.953，2P2D DP16 EP 0.932）。

关键文件：

- .buildkite/amd-disagg/vllm_disagg.sh（模块 启动器；类别 infra；类型 core-logic；符号 parse_args, load_config, resolve_topology, run_proxy）：核心启动器，负责所有角色的协调、配置加载、拓扑解析、健康检查与准确性 / 性能基准，是整个 CI 套件的枢纽。
- .buildkite/amd-disagg/run_xPyD_disagg.slurm（模块 SLURM 脚本；类别 infra；类型 core-logic）：SLURM 作业主体，负责节点选择、环境变量修正、preflight 检查、容器启动及端口映射，是执行阶段的核心。
- .buildkite/amd-disagg/run-slurm-disagg-test.sh（模块 作业提交器；类别 infra；类型 core-logic）：Buildkite 步骤的提交器，支持火忘与轮询两种模式，是 CI 步骤通过 / 失败的关键桥梁。
- .buildkite/amd-disagg/pipeline-disagg.yaml（模块 CI 流水线；类别 infra；类型 configuration）：定义 Buildkite pipeline 的测试矩阵，将不同模型、拓扑和路由器类型映射为具体 CI 步骤。
- .buildkite/amd-disagg/models.yaml（模块 模型配置；类别 config；类型 configuration）：模型目录，集中管理模型级环境变量与 vLLM 启动标志，是扩展新模型测试的入口。
- .buildkite/amd-disagg/cluster.sh（模块 集群配置；类别 infra；类型 configuration）：集群配置模板，定义了拓扑、端口、RDMA/NIC 和 MoRI 相关的所有环境变量，支持跨集群复用。

关键符号：parse_args, load_config, resolve_topology, run_proxy, start_proxy_bg, run_bench, run_accuracy, preflight_node, job_field, select_nodes

关键源码片段

.buildkite/amd-disagg/vllm_disagg.sh

核心启动器，负责所有角色的协调、配置加载、拓扑解析、健康检查与准确性 / 性能基准，是整个 CI 套件的枢纽。

```
# 从 vllm_disagg.sh 中提取的 load_config 函数（省略错误处理细节）
# 功能：加载 cluster.sh 并解析 WIDE_EP_MODE 到 PARALLEL_MODE（tp/ep），设置全局运行参数
load_config() {
    CLUSTER_ENV="${CLUSTER_ENV:-${SCRIPT_DIR}/cluster.sh}"
    [[ -f "${CLUSTER_ENV}" ]] || die "cluster env file not found: ${CLUSTER_ENV}"
    source "${CLUSTER_ENV}" # shellcheck disable=SC1090

    # 拓扑扇出通常来自 cluster.sh，这里重申默认值以保证 rank 算术可定义
```

```
    : "${xP:=1}" "${yD:=1}"

    # 允许 --wide-ep-mode 覆盖环境变量 WIDE_EP_MODE
    [[ -n "${_WIDE_EP_MODE_OVERRIDE}" ]] && WIDE_EP_MODE="${_WIDE_EP_MODE_
    OVERRIDE}"
    WIDE_EP_MODE="${WIDE_EP_MODE:-0}"
    case "${WIDE_EP_MODE}" in
        0) PARALLEL_MODE="tp" ;; # 独立 TP8 服务器
        1) PARALLEL_MODE="ep" ;; # DP + 专家并行
        *) die "WIDE_EP_MODE must be 0 (tp) or 1 (ep); got '${WIDE_EP_MODE}'" ;;
    esac

    # 健康检查后的后续动作：bench | accuracy | none
    RUN_AFTER_HEALTH="${RUN_AFTER_HEALTH:-accuracy}"
    HEALTH_TIMEOUT_S="${HEALTH_TIMEOUT_S:-3600}"
    MORIIO_READ_MODE="${MORIIO_READ_MODE:-0}"
}
```

.buildkite/amd-disagg/run_xPyD_disagg.slurm

SLURM 作业主体，负责节点选择、环境变量修正、preflight 检查、容器启动及端口映射，是执行阶段的核心。

```
# run_xPyD_disagg.slurm 中节点选择与 SLURM 环境变量修正的核心片段
# 目的：让子步骤（srun/vllm_disagg.sh）获得一致、有序的节点列表

# 从 SLURM 分配中提取节点列表，兼容 classic Slurm 与 Spur
if command -v scontrol >/dev/null 2>&1; then
    FULL_NODELIST=$(scontrol show hostnames "$SLURM_JOB_NODELIST")
else
    FULL_NODELIST=$(echo "$SLURM_JOB_NODELIST" | tr ',' '\n')
fi

# 选择前 NUM_NODES 个节点，并保证顺序与 IP 分配一致
if command -v srun >/dev/null 2>&1; then
    SELECTED_NODES=$(echo "$FULL_NODELIST" | head -n "$NUM_NODES" | sort) # classic:
    字母序
elif [[ -n "${SPUR_NODELIST:-}" ]]; then
    SELECTED_NODES=$(echo "${SPUR_NODELIST}" | tr ',' '\n' | sed '/^$/d' | head -n "$NUM_
    NODES") # Spur: 保留 rank 序
else
    SELECTED_NODES=$(echo "$FULL_NODELIST" | head -n "$NUM_NODES")
fi

# 更新 SLURM 环境变量，使子任务看到正确的节点数
export SLURM_NNODES=$NUM_NODES
export SLURM_NTASKS=$NUM_NODES
# ...（其他变量省略）
```

.buildkite/amd-disagg/run-slurm-disagg-test.sh

Buildkite 步骤的提交器，支持火忘与轮询两种模式，是 CI 步骤通过 / 失败的关键桥梁。

```
# run-slurm-disagg-test.sh 中提交作业与 WAIT=0 快速返回的逻辑片段
# 目的：在 Spur 集群上安全地提交并立即返回，避免阻塞 CI 代理

SUBMIT_OUT="$(sbatch --time="${TIME_LIMIT}" "${SUBMIT_SCRIPT}")"
echo "${SUBMIT_OUT}"
# 从 "Submitted batch job 114" 提取作业号（最后一个整数）
JOB_ID="$(printf '%s\n' "${SUBMIT_OUT}" | grep -oE '[0-9]+' | tail -n1 || true)"
[[ -z "${JOB_ID}" ]] && { echo "ERROR: could not parse job id" >&2; exit 1; }

# 计算日志路径：SLURM 作业内会把输出重定向到 NFS 日志文件
LOG_FILE="${LOG_ROOT}/${JOB_NAME}-${JOB_ID}.log"
LOG_DIR="${LOG_ROOT}/${JOB_ID}"

# Spur 默认 fire-and-forget：立即返回，不轮询（可避免 squeue 挂起）
if [[ "${WAIT}" != "1" ]]; then
    echo "[slurm-submit] submitted (WAIT=0). Track with: tail -f ${LOG_FILE}" >&2
    exit 0
fi
# WAIT=1 时继续执行阶段感知轮询（scontrol + 日志 grep），此处省略
```

评论区精华

1. toy proxy vs vllm-router：functionstackx 建议优先采用 production 质量的 vllm-router，并提供了 InferenceX 中的验证脚本链接。lcskrishna 最初顾虑 vllm-router 需要额外容器维护，但最终在其评估后，脚本同时支持 ROUTER_TYPE=proxy 与 ROUTER_TYPE=vllm-router，并分别设置测试步骤；toy proxy 明确计划移除。
2. patch 文件的争议：functionstackx 质疑 apply_39276_rebased.py 和 apply_moriio_2pd_patches.sh 这两个运行时补丁在 nightly CI 中的合理性，认为若 nightly HEAD 本身不健康，补丁会掩盖问题。lcskrishna 回应补丁仅用于 WideEP（EP16+）场景，且在 PR #45043 合入后即可移除，最终从本 PR 中删除。
3. 安全审查：depthfirst-app（bot）指出 toy proxy 的 ZMQ 服务发现监听 0.0.0.0:36367 且无认证，存在 SSRF 与 OPENAI_API_KEY 凭据泄露风险；同时指摘 run_xPyD_disagg.slurm 将整个 $HOME 以特权模式挂载进容器。lcskrishna 回应 toy proxy 文件将移除（实际上已移除），但 $HOME 挂载问题在合并版本中仍存在。
4. 容器清理策略：dllehr-amd 质疑脚本中停止节点上所有容器的操作是否危险。lcskrishna 解释该 CI 独立运行，停止容器是为避免模型加载时的 OOM/ 资源冲突；AndreasKaratzas 补充这种有条件的停止（带锁文件）可保护其他工作任务。最终该逻辑保留但增加了确认条件。

- toy proxy 与 vllm-router 的选型 (design): 结论：双模式并存，toy proxy 逐步退出；vllm-router 成为主要面向生产的路由方案。
- nightly CI 中是否应包含运行时补丁文件 (design): 补丁文件全部移除，本 PR 只保留无需补丁的 TP 模式测试；WideEP 在 issue #51056 中跟踪。

- toy proxy 的 SSRF 与凭据泄露风险 (security): 风险随文件删除而消除，但提醒了 CI 工具的网络安全边界。
- 特权容器挂载 $HOME 是否安全 (security): $HOME 挂载问题在合并版本中仍存在，但 CI 集群相对隔离；停止容器操作在 AndreasKaratzas 确认下保留。
- 是否应停止节点上所有容器 (other): 保留该逻辑，但强调仅适用于独占节点。

风险与影响

- 风险：
  1. 安全风险：run_xPyD_disagg.slurm 将 $HOME 目录绑定挂载进 --privileged 且 seccomp=unconfined 的容器，结合 --network host，可能向容器内进程暴露 SSH 密钥、云凭证等敏感信息。虽然 CI 集群相对隔离，但仍属高风险实践。
  2. 资源泄漏风险：脚本默认 docker stop $(docker ps -q) 停止节点上所有容器，若节点被其他团队共享，可能导致他人作业被中断。当前通过 SLURM --exclusive 节点分配缓解，但并非绝对安全。
  3. 集群拓扑耦合：脚本针对 AMD Spur 集群（MI350X、Pensando AINIC RoCE fabric、/data 共享挂载）做了大量硬编码默认值（IP、端口、NIC 列表），迁移到其他集群时需要逐一覆盖，维护成本较高。
  4. CI 稳定性：阶段感知轮询依赖 NFS 日志和 scontrol，若调度器或文件系统异常，可能出现误判；健康检查超时（HEALTH_TIMEOUT_S=3600）与总体 walltime 的耦合也可能导致任务被 SLURM 提前杀死。
  5. 依赖未完全收敛：toy proxy 虽已从 PR 中移除，但代码库中仍保留对其 PROXY_SCRIPT 路径的引用，若用户误设 ROUTER_TYPE=proxy 会失败；另外 MORIIO_READ_MODE 等特性仍在演进，脚本需要跟随 vLLM 主分支变化。 - 影响：对 vLLM 项目而言，本 PR 新增了 AMD 分布式推理（PD 分离）的夜间 CI 覆盖，使 MoRIIO KV connector 在 TP8 和 DP+EP 模式下的回归能被及时发现，对 AMD ROCm 平台的稳定性有正向作用。对用户和开发者，产品功能无任何改变，但 CI 基础设施的可扩展性得到增强，后续新增模型或拓扑只需修改 models.yaml 和 pipeline-disagg.yaml。影响范围局限于 CI 基础设施，不涉及运行时行为，风险相对可控。 - 风险标记：特权容器挂载 $HOME，停止节点所有容器，硬编码集群拓扑，依赖特定 AMD 集群，部分路径引用已移除的 proxy

关联脉络

- PR #46482 Referenced in discussion: PR #47030 的 issue 评论中明确要求先合并此 PR（MoRIIO 相关依赖）。
- PR #47764 Referenced in discussion: 评论中提到 moriio_toy_proxy.py 的移除依赖此 PR（可能提供官方 proxy 脚本）。
- PR #45043 Referenced in discussion: 讨论提到 WideEP 所需的补丁逻辑已合入此 PR，从而可以从 CI 脚本中移除运行时补丁。
- PR #39276 Referenced in discussion: 函数 stackx 提到的 MoRIIO 修补来源，被 apply_39276_rebased.py 引用。