PR #46831 完整报告

vllm-project/vllm

[CI/Build][CPU] Add test image cache clean-up

合并时间：2026-06-26 23:28

原文链接：http://prhub.com.cn/vllm-project/vllm/pull/46831

执行摘要

- 一句话：添加 CPU 测试镜像缓存清理逻辑
- 推荐动作：该 PR 为增强 CI 基础设施稳健性的良好实践，值得在类似 CI 环境中推广。代码清晰，无功能风险，可直接合并。

功能与动机

PR 描述指出目的是“Add image build cache clean-up to reduce no space issue”，即修复 CI 环境中因镜像和构建缓存积累导致的磁盘空间不足问题。

实现拆解

1. 新增环境变量配置：在 run-cpu-test.sh 中引入 DISK_USAGE_THRESHOLD（默认 70%）和 BUILDKIT_CACHE_MAX（默认 80GB）两个环境变量，分别控制触发清理的磁盘使用率阈值和 BuildKit 缓存保留上限。
2. prune_if_disk_pressure 函数：通过 docker info 获取 Docker 根目录，并使用 df 检查磁盘使用率。若超过阈值，则执行 docker image prune 清理 dangling 镜像，以及 docker builder prune 修剪 BuildKit 缓存（保留最近 80GB 的热数据）。若未超过阈值，则跳过并输出消息。所有 docker 命令通过 || true 保护，不因失败影响脚本执行。
3. cleanup 函数与陷阱：定义 cleanup 函数，先强制删除当前作业的测试镜像（避免影响其他 agent 的缓存复用），然后调用 prune_if_disk_pressure 进行额外空间回收。通过 trap cleanup EXIT 保证脚本退出时执行。
4. 前置清理：在构建镜像之前也调用 prune_if_disk_pressure，确保构建开始时有一定空余空间，降低磁盘满导致构建失败的概率。

关键文件：

- .buildkite/scripts/hardware_ci/run-cpu-test.sh（模块 部署脚本；类别 infra；类型 infrastructure；符号 prune_if_disk_pressure, cleanup, DISK_USAGE_THRESHOLD, BUILDKIT_CACHE_MAX）：添加了磁盘压力检测与清理逻辑，是本次变更的唯一文件。

关键符号：prune_if_disk_pressure, cleanup

关键源码片段

.buildkite/scripts/hardware_ci/run-cpu-test.sh

添加了磁盘压力检测与清理逻辑，是本次变更的唯一文件。

```
# .buildkite/scripts/hardware_ci/run-cpu-test.sh（新增部分）

# 磁盘清理阈值和构建缓存上限（可通过环境变量覆盖）
DISK_USAGE_THRESHOLD=${DISK_USAGE_THRESHOLD:-70}
BUILDKIT_CACHE_MAX=${BUILDKIT_CACHE_MAX:-80GB}

# 仅在磁盘压力大时执行清理。使用 `docker image prune` 删除 dangling 镜像，
# `docker builder prune` 修剪 BuildKit 缓存但保留最近 80GB 的热层，
# 从而在其他作业的缓存复用不受影响的情况下回收空间。
prune_if_disk_pressure() {
    local docker_root disk_usage
    docker_root=$(docker info -f '{{.DockerRootDir}}' 2>/dev/null || true)
    if [ -z "$docker_root" ]; then
        return 0
    fi
    # 计算当前磁盘使用率（百分比）
    disk_usage=$(df "$docker_root" 2>/dev/null | tail -1 | awk '{print $5}' | tr -d '%')
    if [ "${disk_usage:-0}" -gt "$DISK_USAGE_THRESHOLD" ]; then
        echo "--- :broom: Disk usage ${disk_usage}% exceeds ${DISK_USAGE_THRESHOLD}%,
        reclaiming space"
        docker image prune -f || true
        docker builder prune -f --keep-storage="$BUILDKIT_CACHE_MAX" || true
    else
        echo "Disk usage ${disk_usage:-unknown}% within ${DISK_USAGE_THRESHOLD}%
        threshold; skipping prune"
    fi
}

# 作业结束时的清理函数：删除当前测试镜像，然后尝试清理磁盘
cleanup() {
    docker image rm -f "$IMAGE_NAME" || true
    prune_if_disk_pressure
}
trap cleanup EXIT

# 构建前预先清理，确保有足够空间
prune_if_disk_pressure
```

评论区精华

由于 review 评论数为 0，讨论主要来自审批意见。reviewer gmrnlg1971 对使用 docker info 和 df 进行磁盘使用率跟踪和条件清理的逻辑表示认可，认为“非常稳健”，有助于 CI 稳定性。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 误删其他 agent 镜像风险：docker image prune -f 只清理 dangling 镜像（无标签 / 未被容器使用的），不会删除其他 agent 正在使用的唯一标签镜像，风险较低。
  2. 缓存丢失导致构建变慢：docker builder prune --keep-storage 保留了最近 80GB 的缓存层，可维持热缓存命中，但若缓存阈值设置过低或构建层较多，仍可能丢失部分缓存，影响构建速度。
  3. 脚本兼容性：docker info -f 的格式化输出依赖于 Docker 版本，docker builder prune 需要 BuildKit 支持。若 CI 环境 Docker 版本较旧，可能导致命令失败（但 || true 已做容错）。
- 影响：
  1. CI 系统：减少因磁盘空间不足导致的 CPU 测试作业失败，提高 CI 稳定性。
  2. 开发团队：无需手动清理 CI 节点磁盘，降低运维负担。
  3. 影响范围：仅修改 .buildkite/scripts/hardware_ci/run-cpu-test.sh，只作用于 CPU 测试作业，不涉及其他硬件或 CI 流程。 - 风险标记：Docker 版本兼容性，缓存丢失可能影响构建速度

关联脉络

- 暂无明显关联 PR