# PR #30386 完整报告

sgl-project/sglang

[AMD] Run MI355X disaggregation Nightly Test with runtime checkout code mechanism

合并时间：2026-07-08 09:27

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/30386

## 执行摘要

- 一句话：MI355X 分解测试改用工作流检出代码运行
- 推荐动作：建议精读 launch_mi355x.sh 中暂存和安装逻辑的设计，该模式可推广到其他硬件 CI。注意 PYTHONPATH 尾随冒号问题虽未修复，但后续可合并小修复。PR 整体设计良好，opt-out 机制增加了稳健性。

## 功能与动机

当前 MI355X 分解夜间测试使用镜像中预先打包的 sglang 包，无法快速验证工作流检出分支的最新代码。PR 目标是通过运行时重新安装检出代码，加速 CI 迭代，同时保留 opt-out 选项。引用 PR body："Run the MI355X disagg nightly against the workflow checkout instead of image-baked sglang / sglang-router packages."

## 实现拆解

1. GitHub Actions 工作流调整：在 .github/workflows/nightly-amd-mi355x-disagg.yml 中，将 actions/checkout 配置为 fetch-depth: 0 以获取完整 Git 历史（用于 SHA 检测），并设置 persist-credentials: false；新增一个 always() 步骤清理 $HOME/.mi355x_ci/${MATRIX_CONFIG_NAME} 暂存目录。
2. 环境变量与暂存准备：在 launch_mi355x.sh 中，引入 SGLANG_USE_CHECKOUT_RUNTIME 环境变量（默认 1）。若启用，则在 NFS 工作目录下创建 checkout 子目录，排除 __pycache__、*.pyc、.git/config 后，将 $GITHUB_WORKSPACE 通过 tar 管道复制到该目录，并传递 SGLANG_CHECKOUT_SHA 环境变量和只读卷挂载到 Docker 容器。
3. 容器内重新安装 sglang：生成临时脚本 install_checkout_sglang.sh，在容器中将只读挂载点复制到 /tmp/sglang-checkout-runtime，修改 pyproject.toml 动态设置版本（version = "0.0.0.dev0+{SHA}"），然后执行 pip install --no-build-isolation -e .。预填充和解码容器均执行此操作。
4. 基准容器特别处理：基准容器额外编译并安装 sglang-router 包（在 python 目录内 pip install --no-build-isolation -e .），并在启动路由器前通过 PYTHONPATH 指向 /tmp/sglang-checkout-runtime/python。
5. 配方 YAML 配置：为 4 个 FP4 和 4 个 FP8 的 1k1k 配方文件添加 max_total_tokens: 8551168，避免 KV 缓存分配不足导致 OOM。该值通过 launch_mi355x.sh 中的

emit("MAXTOK", rt.get("max_total_tokens", "")) 传递给 Docker 环境变量。

6. 清理与回退：每个测试步骤结束后，通过工作流的 rm -rf 清理暂存目录；若 SGLANG_USE_CHECKOUT_RUNTIME=0，则容器直接使用镜像内建的包，跳过暂存和安装步骤。

关键文件：

- scripts/ci/slurm/launch_mi355x.sh（模块 部署脚本；类别 infra；类型 infrastructure）：核心变更文件：实现暂存检出代码、生成安装脚本、传递环境变量等所有逻辑。
- .github/workflows/nightly-amd-mi355x-disagg.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：工作流调整，增加 fetch-depth:0、persist-credentials:false 以及 always() 清理步骤。
- scripts/ci/slurm/recipes/mi355x-fp4/dsv4flash/1k1k/1p1d-dp8ep8-mtp.yaml（模块 配方配置；类别 infra；类型 configuration）：新增 max_total_tokens 参数，防止 DSV4 Flash 1k1k 配置在 MI355X 上 OOM。
- scripts/ci/slurm/recipes/mi355x-fp4/dsv4flash/1k1k/1p1d-dp8ep8.yaml（模块 配方配置；类别 infra；类型 configuration）：新增 max_total_tokens 参数，同系列配置变更。
- scripts/ci/slurm/recipes/mi355x-fp4/dsv4flash/1k1k/1p1d-mtp.yaml（模块 配方配置；类别 infra；类型 configuration）：新增 max_total_tokens 参数，同系列配置变更。
- scripts/ci/slurm/recipes/mi355x-fp4/dsv4flash/1k1k/1p1d.yaml（模块 配方配置；类别 infra；类型 configuration）：新增 max_total_tokens 参数，同系列配置变更。
- scripts/ci/slurm/recipes/mi355x-fp8/dsv4flash/1k1k/1p1d-dp8ep8-mtp.yaml（模块 配方配置；类别 infra；类型 configuration）：新增 max_total_tokens 参数，同系列配置变更。
- scripts/ci/slurm/recipes/mi355x-fp8/dsv4flash/1k1k/1p1d-dp8ep8.yaml（模块 配方配置；类别 infra；类型 configuration）：新增 max_total_tokens 参数，同系列配置变更。
- scripts/ci/slurm/recipes/mi355x-fp8/dsv4flash/1k1k/1p1d-mtp.yaml（模块 配方配置；类别 infra；类型 configuration）：新增 max_total_tokens 参数，同系列配置变更。
- scripts/ci/slurm/recipes/mi355x-fp8/dsv4flash/1k1k/1p1d.yaml（模块 配方配置；类别 infra；类型 configuration）：新增 max_total_tokens 参数，同系列配置变更。

关键符号：未识别

## 关键源码片段

### scripts/ci/slurm/launch_mi355x.sh

核心变更文件：实现暂存检出代码、生成安装脚本、传递环境变量等所有逻辑。

```
# 如果启用运行时检出，则在 NFS 上暂存当前仓库并挂载为只读卷
if [[ "$SGLANG_USE_CHECKOUT_RUNTIME" == "1" ]]; then
  CHECKOUT_STAGE="$WORKDIR/checkout"
  CHECKOUT_SHA="$(git -C "$GITHUB_WORKSPACE" rev-parse HEAD)"
  echo "Staging checkout runtime: sha=$CHECKOUT_SHA -> $CHECKOUT_STAGE"
  rm -rf "$CHECKOUT_STAGE"
  mkdir -p "$CHECKOUT_STAGE"
  # 排除缓存和 Git 配置文件，减少传输量
```

```
  tar --exclude='__pycache__' --exclude='*.pyc' --exclude='.git/config' \
    -C "$GITHUB_WORKSPACE" -cf - . | tar -C "$CHECKOUT_STAGE" -xf -
  # 传递 SHA 并挂载只读卷
  CHECKOUT_DOCKER_ARGS="$CHECKOUT_DOCKER_ARGS -e SGLANG_CHECKOUT_SHA=
  $CHECKOUT_SHA -v $CHECKOUT_STAGE:/sglang-checkout:ro"
else
  echo "SGLANG_USE_CHECKOUT_RUNTIME=0; using sglang package baked into image."
fi

# 生成的 install_checkout_sglang.sh 脚本（保存为 heredoc）
cat > "$WORKDIR/install_checkout_sglang.sh" <<'INSTALLEOF'
#!/bin/bash
set -ex
# 如果使用运行时检出，则进行安装
if [ "${SGLANG_USE_CHECKOUT_RUNTIME:-1}" != "0" ]; then
  # 将只读挂载复制到 /tmp，以获得写入权限
  cp -a /sglang-checkout /tmp/sglang-checkout-runtime
  cd /tmp/sglang-checkout-runtime
  # 动态修改版本标识，包含 SHA
  sed -i "s/^version = \".*\"/version = \"0.0.0.dev0+${SGLANG_CHECKOUT_SHA:-unknown}\"/"
  pyproject.toml
  # 不使用构建隔离，利用容器内已有的编译依赖
  pip install --no-build-isolation -e .
fi
INSTALLEOF
```

## 评论区精华

主要讨论来自自动代码审查机器人 gemini-code-assist[bot] 的 5 条建议，但均未被采纳：

- .git 排除范围：机器人建议在 tar 时排除整个 .git 目录而非仅 .git/config，以节省 NFS 空间和传输时间。最终代码维持 --exclude='.git/config'，保留 .git 其余部分。
- PYTHONPATH 尾随冒号风险：机器人在 4 处发现使用 :$PYTHONPATH 或 :${PYTHONPATH:-} 可能导致空 PYTHONPATH 时产生尾随冒号，从而将当前目录加入 Python 搜索路径。建议改用 ${PYTHONPATH:+:$PYTHONPATH}。最终代码未修改，可能因为容器内 PYTHONPATH 通常非空，或作者认为风险可接受。
  - 排除 .git 目录的建议 (performance)：作者未采纳，仅排除 .git/config。
  - PYTHONPATH 尾随冒号风险 (correctness)：作者未修改，风险未解决。

## 风险与影响

- 风险：
  1. NFS 性能与空间：暂存整个工作空间到共享 NFS 可能增加网络流量和磁盘占用，尤其当仓库或 submodule 较大时。但排除了缓存和 Git 配置文件，体积可控。
  2. 容器内安装时间：每次容器启动都需 pip install，可能增加 30 秒到 1 分钟开销，但仅影响 CI。

3. 硬编码 max_total_tokens：配方中直接写死 8551168，若未来换用更大上下文模型或不同 batch size，可能导致 OOM 或资源浪费。应改为公式计算。
4. PYTHONPATH 尾随冒号：可能意外将容器工作目录添加到 Python 搜索路径，引入不可预测的 import 行为，但概率较低。
5. SGLANG_USE_CHECKOUT_RUNTIME 默认开启：若 NFS 或脚本出错，可能导致测试失败；但可通过设置 0 回退。 - 影响：用户视角：无直接影响，仅影响 AMD MI355X 夜间测试流程。系统视角：暂时增加 NFS 存储和网络带宽使用；CI 流水线可更快验证代码变更，提升迭代效率。团队视角：减少了维护镜像 tag 的负担，允许开发者通过 PR 快速触发 MI355X 测试。需要确保 NFS 挂载稳定，并关注清理逻辑避免磁盘膨胀。

- 风险标记：CI 流程依赖 NFS 可用性，硬编码 max_total_tokens，PYTHONPATH 尾随冒号风险

## 关联脉络

- PR #30313 Cap DSV4 Flash max_total_num_tokens: 该 PR 的配方配置中 max_total_tokens=8551168 直接来自 #30313，且提交 [AMD] Cap DSV4 Flash max_total_num_tokens 被包含在此 PR 的提交历史中。