PR #35610 完整报告

sgl-project/sglang

[MUSA] Harden CI dependencies and diffusion warmup

合并时间：2026-08-21 00:40

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/35610

执行摘要

- 一句话：加固 MUSA CI 依赖隔离，修复 Wan diffusion warmup
- 推荐动作：建议负责 MUSA/MTT 硬件 CI、或关注多硬件后端 CI 隔离与可复现性的工程师精读本 PR，尤其是基于已装栈生成 constraints、fail-closed 校验与 task-local 隔离的组合设计；一般用户可跳过。值得关注的权衡：用显式版本映射替代公共依赖解析、用 warmup 请求形状保证 benchmark 测量准确性。

功能与动机

PR body 的根因分析指出，MUSA CI 失败并非 Wan 模型本身慢，而是 server warmup 固定 17 帧而 benchmark 请求为 24 帧，导致每次新 server 的首次 Denoise Step 0 成本被计入测量；同时复用 runner 的 user-site 使公共依赖解析把 MUSA Triton 栈替换成通用 Torch/Triton 包，产生不可复现的安装环境。因此需要隔离安装环境、校验版本一致性，并让 warmup 复用真实请求形状。

实现拆解

1. 依赖隔离与安装加固

在 scripts/ci/musa/musa_install_dependency.sh 中，当 GITHUB_ACTIONS=true 时用 mktemp 创建 task-local PYTHONUSERBASE 并写入 GITHUB_ENV / GITHUB_PATH，所有依赖安装落到隔离目录；同时保留 pip 内容寻址 wheel cache，避免各 lane 重复冷下载。torchada 先于 MUSA wheel 包安装且不带依赖解析，防止公共索引引入 CUDA Torch / Triton。对应两个 workflow 将依赖安装超时从 15 分钟提高到 30 分钟。

2. constraints 生成与 fail-closed 校验

新增 scripts/ci/musa/musa_python_stack.py：CORE_DISTRIBUTIONS 包含 torch、torch-musa、torchada、triton；build_constraints() 从已安装栈读取版本生成 name==version 约束并写盘；compressed_tensors_version() 按 Torch 小版本显式映射（2.9→0.15.0，2.11→0.17.0），未知行抛 StackError；validate_core_versions() 校验 Torch 与 Torch-MUSA 版本行一致、Triton 目标正确、MUSA runtime/device 存在。任何不一致都直接失败，不静默装错包。

3. 单元测试与 CI 门禁

新增 scripts/ci/musa/test_musa_python_stack.py，用 mock.patch 模拟 importlib.metadata.version，覆盖 5 个用例；pr-test-musa.yml 增加约束测试 step；触发路径从 scripts/ci/musa/* 扩为 scripts/ci/musa/**，并挂到 multimodal_gen 与 sgl_kernel lane。

4. Wan benchmark warmup 修复

在 python/sglang/multimodal_gen/test/server/musa/testcase_configs_musa.py 中为 wan2_1_t2v_1.3b_musa 增加 --warmup-mode request，让 warmup 复用 832x480x24f 请求形状（24 帧），避免 generic warmup 17 帧导致的首次 shape 初始化进入 denoising 指标。

5. 陈旧测试清理

从 PR 与 nightly MUSA workflow 移除 test_per_token_quant_fp8.py；该测试已在上游 CUDA JIT 迁移中删除，MUSA 生产仍走 AOT 路径，故不以 CUDA-only 注册测试替代。

关键文件：

- scripts/ci/musa/musa_python_stack.py（模块 依赖校验；类别 infra；类型 infrastructure；符号 StackError, distribution_version, torch_minor, compressed_tensors_version）：新增的依赖约束生成与校验核心：定义 CORE/OPTIONAL 发行版清单、compressed-tensors 版本映射、build_constraints/write_constraints/validate_core_versions，是 fail-closed 策略的落地处。
- scripts/ci/musa/test_musa_python_stack.py（模块 单元测试；类别 test；类型 test-coverage；符号 MusaPythonStackTest, test_compressed_tensors_for_torch_29, test_compressed_tensors_for_torch_211, test_unknown_torch_line_fails_closed）：为 musa_python_stack.py 提供 5 个单元测试，覆盖版本映射、fail-closed 异常与 constraints pin 完整性，是 CI 门禁的直接保障。
- scripts/ci/musa/musa_install_dependency.sh（模块 安装脚本；类别 infra；类型 infrastructure）：安装脚本的核心加固：task-local PYTHONUSERBASE 隔离、保留 wheel cache、MUSA Triton/torchada 版本与 SHA256 校验、无依赖解析安装后生成 constraints，是根因一（依赖污染）的直接修复。
- python/sglang/multimodal_gen/test/server/musa/testcase_configs_musa.py（模块 测试配置；类别 test；类型 test-coverage）：Wan 2.1 benchmark 的 warmup 修复点：增加 --warmup-mode request，使 warmup 使用真实请求形状，消除 denoising 指标中的 first-shape 污染。
- .github/workflows/pr-test-musa.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：PR CI 门禁配套：触发路径扩大、新增约束测试 step、依赖安装超时 15→30 分钟、移除过时 FP8 AOT 测试命令。
- .github/workflows/nightly-test-musa.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：与 PR 工作流同步：提高依赖安装超时并移除过时 FP8 AOT 测试，保证 nightly 与 PR 行为一致。

关键符号：compressed_tensors_version, build_constraints, write_constraints, validate_core_versions, distribution_version, torch_minor, MusaPythonStackTest

关键源码片段

scripts/ci/musa/musa_python_stack.py

新增的依赖约束生成与校验核心：定义 CORE/OPTIONAL 发行版清单、compressed-tensors 版本映射、build_constraints/write_constraints/validate_core_versions，是 fail-closed 策略的落地处。

```
# compressed-tensors 0.16+ 要求 Torch 2.10+，而旧 MUSA runner 栈用 Torch 2.9，
# 所以按 Torch 小版本显式映射，不再全局 pin 一个版本
COMPRESSED_TENSORS_BY_TORCH_MINOR = {
    (2, 9): "0.15.0",
    (2, 11): "0.17.0",
}


class StackError(RuntimeError):
    """当已安装的 MUSA 栈违反 CI 契约时抛出。"""


def compressed_tensors_version(torch_version: str) -> str:
    # 解析出 Torch 主 / 次版本行，例如 2.11.0.post1+musa5.2.0 -> (2, 11)
    minor = torch_minor(torch_version)
    try:
        return COMPRESSED_TENSORS_BY_TORCH_MINOR[minor]
    except KeyError as exc:
        supported = ", ".join(
            f"{major}.{minor}"
            for major, minor in sorted(COMPRESSED_TENSORS_BY_TORCH_MINOR)
        )
        # 遇到未支持的 Torch line 时 fail closed，而不是静默装错版本
        raise StackError(
            f"unsupported MUSA Torch line {minor[0]}.{minor[1]}; "
            f"supported lines: {supported}"
        ) from exc


def build_constraints() -> list[str]:
    # 核心发行版必须全部安装，缺失即抛 StackError，保证约束可复现
    versions = {name: distribution_version(name) for name in CORE_DISTRIBUTIONS}
    pins = [f"{name}=={version}" for name, version in versions.items()]

    # 可选 vendor 包存在才 pin，避免缺一个可选包就整体失败
    for name in OPTIONAL_VENDOR_DISTRIBUTIONS + OPTIONAL_STACK_DISTRIBUTIONS:
        try:
            version = importlib.metadata.version(name)
```

```
            except importlib.metadata.PackageNotFoundError:
                continue
            pins.append(f"{name}=={version}")

    # compressed-tensors 版本由 Torch line 决定，用显式映射锁定
    pins.append(f"compressed-tensors=={compressed_tensors_version(versions['torch'])}")
    return sorted(pins, key=str.casefold)
```

scripts/ci/musa/musa_install_dependency.sh

安装脚本的核心加固：task-local PYTHONUSERBASE 隔离、保留 wheel cache、MUSA Triton/torchada 版本与 SHA256 校验、无依赖解析安装后生成 constraints，是根因一（依赖污染）的直接修复。

```
# 每个 GitHub Actions job 使用 task-local PYTHONUSERBASE 隔离安装目录，
# 避免复用 runner 的 user-site 被公共依赖解析污染；同时保留 pip 共享
# wheel cache，避免各 lane 反复冷下载数百 MB 并超过安装超时。
if [ "${GITHUB_ACTIONS:-}" = "true" ]; then
    MUSA_CI_ISOLATED_USERBASE="1"
    PYTHONUSERBASE="$(mktemp -d "${RUNNER_TEMP:-${TMPDIR:-/tmp}}/sglang-musa-python.
    XXXXXX")"
    export PYTHONUSERBASE
    if [ -n "${GITHUB_ENV:-}" ]; then
        echo "PYTHONUSERBASE=${PYTHONUSERBASE}" >> "$GITHUB_ENV"
    fi
    if [ -n "${GITHUB_PATH:-}" ]; then
        echo "${PYTHONUSERBASE}/bin" >> "$GITHUB_PATH"
    fi
    echo "Using task-local Python user base: ${PYTHONUSERBASE}"
fi

# torchada 的 Torch 依赖未 pin 版本：若在 MUSA wheel 包之前从公共索引安装，
# 会把 CUDA Torch / CUDA Triton 拉进来；这里无依赖解析安装，随后由生成的
# constraints 把整个 MUSA 栈锁定为已安装版本。
```

评论区精华

本 PR 没有 review 评论线程，唯一评论是作者 yeahdongcn 的 /tag-and-rerun-ci 指令，用于触发标签与 CI 重跑，不包含技术讨论。最终按 squashed head 的 MUSA 全量 job 均通过（Qwen-Image、Wan2.1、2-GPUdiffusion、sgl-kernel、multimodal层测试、finishgate）。

- CI 重跑指令（唯一评论） (other): 无技术结论；最终按 exact squashed head 的 MUSA 全量 job 通过。

风险与影响

- 风险：
  - COMPRESSED_TENSORS_BY_TORCH_MINOR 只覆盖 2.9 / 2.11 两档：若 MUSA runner 引入新的 Torch line（如 2.10），build_constraints() 会直接抛 StackError

fail closed，CI 立即变红，需升级时同步扩展映射。

- 安装脚本硬编码 MUSA Triton 3.2.0 与 torchada 0.1.82 的 SHA256，且针对 CPython 3.10 x86_64 wheel；Python 或 wheel 构建升级时必须同步更新 digest。
- request-based warmup 使每次 CI 先执行一次真实形状请求，增加 warmup 墙钟时间；PR body 明确不消除首个生产请求的冷启动代价，30 分钟安装超时是否仍够用需观察。
- 移除 FP8 AOT 测试后，MUSA AOT 路径缺少直接回归覆盖。
- 触发路径扩大为 scripts/ci/musa/** 后，脚本改动会触发更多 lane，CI 成本上升。
- 影响：影响范围集中在 MUSA CI 全 lane：依赖安装从共享 user-site 改为 task-local 隔离，安装与验证方式改变；Wan 2.1 benchmark 的 warmup 行为改变，测量的是真实请求稳态；开发者修改 scripts/ci/musa/** 会触发更多 MUSA lane 验证。对 SGLang 推理用户无直接功能影响，对 MUSA 平台 CI 稳定性有正向作用，团队需维护版本映射表与 SHA256 清单。影响程度中等，全部限于基础设施。
- 风险标记：硬编码 SHA256 需随 Python 升级同步，Torch line 映射封闭需扩展，warmup 增加 CI 墙钟时间，FP8 AOT 测试移除后缺直接覆盖

关联脉络

- PR #35602 [AMD][CI] Default the ROCm 7.2 PR gate to ROCm 7.2.4 Image: 同属硬件后端 CI 加固脉络，与 AMD CI 门禁调整并行，反映平台级 CI 治理方向。
- PR #35679 [diffusion] Refresh eager optimization skills and benchmark safeguards: 同属 diffusion benchmark 质量保障，强调测量真实性与缓存清理，与本 PR 的 warmup 卫生一致。
- PR #35455 [Quant] Load compressed-tensors kv_cache_scheme scales: 涉及 compressed-tensors 依赖与版本行为，本 PR 的 COMPRESSED_TENSORS_BY_TORCH_MINOR 映射与之有间接关联。