PR #33619 完整报告

sgl-project/sglang

[CI] Speed up dependency install: dual-ABI Rust ext cache and prevalidation pruning

合并时间：2026-08-05 11:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33619

执行摘要

- 一句话：双 ABI Rust 缓存提速 CI，移除预校验 pass
- 推荐动作：值得精读，尤其适合 CI 与基础设施维护者。本 PR 展示了三个可复用设计：以双 ABI 预编译缓存换取安装期零编译的权衡；用 pre-commit lint 强制两处无法互相引用的配置默认值保持同步，把静默失败挡在提交前；对跨容器共享缓存采取保守清理（不删锁、加年龄门槛）。仅关注推理引擎运行时的读者可直接跳过——运行时零改动。

功能与动机

PR body 明确说明三点动机：一是 "Build the prebuilt Rust extension modules for both cp310 and cp312 so every pool skips the install-time cargo rebuild"，此前只预编译 cp310 扩展，运行 Python 3.12 的 h20 池安装时必然触发本地 cargo 编译；二是 "remove the model-cache prevalidation pass whose output nothing consumes (launch/load-time validation with repair stays)"，prevalidate_cached_models.py 的输出没有任何消费者，真正起作用的是启动 / 加载时校验与修复；三是 "stop deleting cross-container HF download lock files"，清理脚本删除 **/*.lock 会与持锁超 30 分钟的其它容器产生竞态。

实现拆解

1. 双 ABI Rust 扩展缓存：.github/workflows/_pr-test-rust-ext-build.yml 的构建 job 同时配置 Python 3.10 与 3.12（setup-python），循环执行 SGLANG_BUILD_RUST_EXTS=all python setup.py build_rust --inplace；每个解释器使用独立 CARGO_TARGET_DIR（../py310、.../py312），因为 PyO3 的 fingerprint 随解释器变化，共享目录会在 ABI 切换时触发整体重建。scripts/ci/utils/stage_rust_ext_modules.sh 相应放宽为每包可存在多个 _core*.so，并校验 server/grpc/multimodal 三包的后缀集合完全一致，防止某 ABI 测试静默跳过。缓存 key 前缀统一改为 rust-ext-x86_64-cp310-cp312，保存端（构建 workflow）与恢复端（download-rust-ext action）同步更新。
2. 预校验 pass 下线：scripts/ci/utils/prevalidate_cached_models.py（407 行）整文件删除；scripts/ci/cuda/prepare_runner.sh 移除其调用，仅保留 HF 缓存清理；python/sglang/srt/model_loader/ci_weight_validation.py 同步删除 908 行共享 marker 逻辑（VALIDATION_MARKER_VERSION、_get_validation_marker_path、_read_validation_marker、_write_validation_marker、_remote_file_exists、_validate_json_file、_validate_config_and_tokenizer_files 等），只保留 per-run marker 与 validate_cache_lightweight，继续支撑测试期 HF_HUB_OFFLINE=1 判定。

3. 安装路径瘦身：scripts/ci/cuda/ci_install_dependency.sh 把 verify_imports 的多次 python3 -c 启动合并为单个进程（一次完成 torch/cutlass 版本打印、find_spec("sglang") 解析位置校验、import sglang.srt.{server,grpc,multimodal}._core 加载校验）；setup_pip_toolchain 只在 USE_VENV=1 时升级 venv 内 pip；setup_ld_library_path 将 nvidia lib 查找从整个 $SITE_PACKAGES 收窄到 $SITE_PACKAGES/nvidia。
4. HF 缓存清理语义调整：scripts/ci/utils/cleanup_hf_cache.py 不再清理 **/*.lock，*.incomplete 与 *.tmp 增加 2 小时 mtime 年龄门槛，避免误删其他容器仍在 append 的产物。
5. 一致性门禁与播种配套：新增 scripts/ci/check_rust_ext_cache_prefix.py 并在 .pre-commit-config.yaml 注册 check-rust-ext-cache-prefix hook，比较构建 workflow 与下载 action 的 cache_key_prefix 默认值；seed-rust-ext-cache.yml 将构建 workflow 文件加入路径触发，并补充 actions: read 权限以支持 gh cache list 诊断。

关键文件：

- .github/workflows/_pr-test-rust-ext-build.yml（模块 CI 流水线；类别 infra；类型 infrastructure）：双 ABI 构建的核心流水线改动：新增 Python 3.12 构建任务，按解释器隔离 CARGO_TARGET_DIR，缓存 key 前缀改为 rust-ext-x86_64-cp310-cp312，并增加缓存命中 / 未命中报告。
- scripts/ci/check_rust_ext_cache_prefix.py（模块 CI 脚本；类别 infra；类型 infrastructure；符号 main）：新增一致性检查脚本，防止构建流水线与下载 action 的 cache_key_prefix 默认值失配导致所有池静默回退源构建；配套 pre-commit hook 在提交时拦截。
- python/sglang/srt/model_loader/ci_weight_validation.py（模块 模型加载；类别 source；类型 data-contract；符号 _remote_file_exists, _get_validation_marker_path, _remove_per_run_marker, _read_validation_marker）：删除约 908 行无人消费的预校验代码（共享验证 marker、远端文件存在性检查、完整 JSON 校验等），保留 per-run marker 与轻量启动校验；是本次最大减法。
- scripts/ci/utils/prevalidate_cached_models.py（模块 CI 脚本；类别 infra；类型 deletion；符号 find_all_hf_snapshots, is_transformers_text_model, scan_weight_files, validate_snapshot）：整个预校验 pass 被删除（407 行），其输出在 prepare_runner 后无人消费；启动 / 加载时校验替代它。
- scripts/ci/cuda/ci_install_dependency.sh（模块 依赖安装；类别 infra；类型 infrastructure；符号 verify_imports, setup_ld_library_path, setup_pip_toolchain）：verify_imports 合并为单个 python 进程、pip 升级只在 venv 内执行、nvidia lib 查找限定到 site-packages/nvidia，均属安装路径瘦身。
- scripts/ci/utils/cleanup_hf_cache.py（模块 缓存清理；类别 infra；类型 infrastructure；符号 find_stale_artifacts）：停止删除跨容器 HF 下载锁文件，并为 incomplete/tmp 增加 2 小时年龄门槛，避免与其它容器并发下载产生竞态。
- scripts/ci/utils/stage_rust_ext_modules.sh（模块 扩展打包；类别 infra；类型 infrastructure）：适配双 ABI 产物：每个包允许存在多个 _core*.so，并强制 server/grpc/multimodal 三包扩展后缀集合一致，防止某 ABI 测试静默跳过。

- .pre-commit-config.yaml（模块 提交钩子；类别 config；类型 configuration）：新增 check-rust-ext-cache-prefix hook，把缓存 key 一致性检查纳入提交门禁，属于本次防回归的关键配套。
- .github/workflows/seed-rust-ext-cache.yml（模块 缓存播种；类别 infra；类型 infrastructure）：种子缓存工作流增加对 _pr-test-rust-ext-build.yml 的路径监听，缓存 key 前缀变更时自动重新播种；并补 actions: read 权限供 gh cache list 使用。
- scripts/ci/cuda/prepare_runner.sh（模块 Runner 准备；类别 infra；类型 infrastructure）：从 runner 准备流程移除 prevalidate_cached_models.py 调用，改为仅做缓存清理，是预校验 pass 下线的入口。
- .github/actions/download-rust-ext/action.yml（模块 扩展下载；类别 infra；类型 infrastructure）：download action 的 cache_key_prefix 默认值与构建流水线同步更新为 rust-ext-x86_64-cp310-cp312，是双 ABI 缓存命中闭环的另一半。

关键符号：main, find_stale_artifacts, verify_imports, setup_ld_library_path, setup_pip_toolchain, validate_cache_lightweight, _write_per_run_marker, _get_per_run_marker_path

关键源码片段

scripts/ci/check_rust_ext_cache_prefix.py

新增一致性检查脚本，防止构建流水线与下载 action 的 cache_key_prefix 默认值失配导致所有池静默回退源构建；配套 pre-commit hook 在提交时拦截。

```
#!/usr/bin/env python3
"""确保 rust-ext 缓存 key 前缀的两处默认值保持一致。

构建流水线以自己的默认值保存缓存条目，下载 action 以自己的默认值恢复；
两个文件无法互相引用，一旦失配，所有测试池都会在安装时静默回退到
源码构建（即在每台 runner 上重新 cargo 编译）。
"""

import sys

import yaml

# 保存端与恢复端的定义位置：构建 workflow 与下载 action
BUILD_WORKFLOW = ".github/workflows/_pr-test-rust-ext-build.yml"
DOWNLOAD_ACTION = ".github/actions/download-rust-ext/action.yml"


def main() -> int:
    with open(BUILD_WORKFLOW, encoding="utf-8") as f:
        workflow = yaml.safe_load(f)
    with open(DOWNLOAD_ACTION, encoding="utf-8") as f:
        action = yaml.safe_load(f)

    # yaml 1.1 会把 `on:` 解析成布尔 True，因此这里兼容 `on` 与 True 两个键
```

```
    triggers = workflow.get("on", workflow.get(True))
    save_prefix = triggers["workflow_call"]["inputs"]["cache_key_prefix"]["default"]
    restore_prefix = action["inputs"]["cache_key_prefix"]["default"]

    if save_prefix != restore_prefix:
        print("ERROR: rust-ext cache_key_prefix defaults do not match.")
        print(f" {BUILD_WORKFLOW} saves under: {save_prefix}")
        print(f" {DOWNLOAD_ACTION} restores with: {restore_prefix}")
        print("Bump both together, or every pool falls back to source builds.")
        return 1

    return 0


if __name__ == "__main__":
    sys.exit(main())
```

scripts/ci/utils/cleanup_hf_cache.py

停止删除跨容器 HF 下载锁文件，并为 incomplete/tmp 增加 2 小时年龄门槛，避免与其它容器并发下载产生竞态。

```
def find_stale_artifacts(cache_dir: str) -> List[Path]:
    """查找 HF 缓存中残留的过期下载产物。

    只清理 incomplete 标记与临时文件；锁文件 `*.lock` 不再纳入清理，
    因为另一个容器可能正持有下载锁长达 30 分钟以上，强行 unlink 会
    让后续获取者拿到新 inode，两个写入者并发产生竞态。
    """
    cache_path = Path(cache_dir)
    if not cache_path.exists():
        return []

    # 不再使用 `**/*.lock` 模式；incomplete 与 tmp 也需超过 2 小时
    # 才算 stale，避免误删其他容器仍在 append 的 blob 文件。
    patterns = [
        "**/*.incomplete", # 未完成下载标记
        "**/*.tmp", # 临时文件
    ]
    min_stale_age_seconds = 2 * 60 * 60
    stale_files = []
    now = time.time()

    for pattern in patterns:
        for path in cache_path.glob(pattern):
            try:
                # 只有 mtime 足够旧的文件才清理；扫描期间文件消失则跳过
                if now - path.stat().st_mtime < min_stale_age_seconds:
                    continue
            except OSError:
```

```
                continue # 文件在 stat 前已被其他进程删除
            stale_files.append(path)

    return stale_files
```

评论区精华

本 PR 没有任何 review 评论（review_comments_count = 0），Issue 中仅作者 hnyls2002 发出 /tag-and-rerun-ci 触发重跑。技术意图主要从 9 个提交的演进中体现：先落地双 ABI 构建，再删预校验、保护锁文件、合并探针，最后补上 cache key 一致性 lint——因为构建 workflow 与下载 action 的默认值无法互相引用，失配时所有池会静默回退到源码构建，作者用 pre-commit hook 把这个"静默失败"变成提交期可见错误。代码注释还解释了跨容器锁文件的竞态：另一容器可能持有下载锁 30 分钟以上，unlink 会给后续获取者新 inode，导致两个写入者并发。

- 无实质 review 讨论；唯一评论为 CI 重跑指令 (other): 作者以 9 个提交独立完成并合入，未产生分歧点。

风险与影响

- 风险：
  - 缓存 key 语义变更：前缀从 rust-ext-x86_64 变为 rust-ext-x86_64-cp310-cp312 后旧缓存全部失效，合入后首个 CI 周期可能出现大面积 cache miss 与源构建回退；同时双 ABI 使产物翻倍，stage_rust_ext_modules.sh 上传体积与持久化 runner 磁盘占用上升。
  - 死代码删除的引用风险：ci_weight_validation.py 删除 908 行，若仓库内仍有未迁移脚本引用 _validate_config_and_tokenizer_files、_read_validation_marker 等符号会直接 ImportError。当前已同步删除唯一调用方 prevalidate_cached_models.py，但内部模块可能被外部脚本或下游 CI 隐藏依赖，需在合入后的全量 CI 观察。
  - 缓存清理行为变化：不再删除 *.lock 且 incomplete/tmp 需要 2 小时年龄，HF 缓存磁盘占用可能缓慢增长；runner 磁盘紧张时清理不及时可能导致构建失败。
  - verify_imports 合并副作用：单进程内先 import torch，若 torch 加载失败，后续 find_spec 与 _core 加载校验不会执行，失败定位粒度变粗（CI 日志仍会打印完整 traceback）。
  - 后缀一致性校验的 fail-fast：stage_rust_ext_modules.sh 现在要求三包扩展后缀集合完全一致，任何单包缺失都会让 stage 失败，这是刻意收紧，但引入了新的失败模式。
- 影响：
  - 影响范围：全部 GitHub Actions 测试池（PR 与 nightly），尤其是 Python 3.12 的 h20 池不再在安装期编译 Rust 扩展；prepare_runner.sh 的预校验环节从所有 runner 上消失，节省每次初始化时间。
  - 用户影响：零。ci_weight_validation.py 的 CI 专用逻辑不进入普通用户路径，运行时行为无变化。
  - 团队影响：CI 依赖安装时间下降、cache 语义更可预期；维护者需记住双 ABI key 的同步约束（已有 pre-commit 兜底），并在扩容新 ABI（如 cp313）时同步更新三处：构

建 workflow、download action、stage 脚本校验逻辑。

- 风险标记：缓存 key 变更引发首次全量重建，cache 前缀失配静默回退源构建，跨容器锁文件并发风险，大量死代码删除影响未知引用，缺少直接测试覆盖

关联脉络

- PR #33597 [CI] Extract download-rust-ext and give every install step a cache fallback: 同一 CI 依赖安装加速线：建立 download-rust-ext action 与 rust-ext 缓存回退，本 PR 直接修改该 action 与 _pr-test-rust-ext-build.yml，延续其缓存策略并将其升级为双 ABI。
- PR #33586 [CI] Trim redundant B200 test registrations: CI 提速主题的姊妹 PR，通过削减测试注册减少总耗时，与本 PR 的依赖安装提速互补，同属近期 CI 时长优化系列。
- PR #33605 [CI] Make B200 base-b suites single-GPU as prep for 1-gpu B200 runners: runner 资源配置调整，同样服务于 CI 时长优化，与本 PR 的安装提速共同降低测试池压力。