PR #33597 完整报告

sgl-project/sglang

[CI] Extract `download-rust-ext` and give every install step a cache fallback

合并时间：2026-08-05 06:33

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/33597

执行摘要

- 一句话：提取 rust-ext 下载操作，CI 安装回退到缓存
- 推荐动作：值得精读，特别是 download-rust-ext 的“artifact 优先、cache 兜底”设计，以及用 outputs.hit 门控工具链安装的思路。对维护大型 GitHub Actions 的团队有参考价值，可迁移到其他需要“预构建产物 + 缓存回退”的场景。

功能与动机

PR body 指出 _pr-test-rust-ext-build.yml 发布的 artifact 只有 1 天签发，过期后任何重跑都会在安装期间重新编译全部 Rust 扩展 ;rerun-test.yml 因为是 workflow_dispatch，运行中没有 rust-ext-build 作业，完全读不到 artifact，只能每次冷编译。作者希望让过期 artifact 不再导致冷 cargo build，并把可淘汰的 cache 作为第二层来源。

实现拆解

实现分 5 步：

1. 新建复合操作 .github/actions/download-rust-ext/action.yml，优先从 artifact 读取，失败后安装 zstd 并回退 actions/cache/restore@v4，命中时通过 GITHUB_ENV 写入 SGLANG_BUILD_RUST_EXTS=none，并通过 outputs.hit 暴露命中状态。
2. 收敛调用点：_pr-test-stage.yml，_pr-test-stage-cpu.yml，pr-test-jit-kernel.yml，pr-test-multimodal-gen.yml，pr-test-sgl-kernel.yml 中 16 处内联步骤替换为 2-4 行操作引用，删除重复的 continue-on-error 与步骤级 env。
3. 调整 CPU 阶段门控：_pr-test-stage-cpu.yml 的 install_rust_protoc.sh 与 Swatinem/rust-cache 改用 outputs.hit != 'true' 才执行，命中时同时跳过工具链安装和缓存恢复。
4. 接入 rerun-test.yml: 三个作业 (cuda / multimodal_gen / cpu) 都使用操作 ,cpu 作业在托管 runner 上原本每次冷编译，现在可命中缓存并跳过工具链。
5. 处理 aarch64 例外：pr-test.yml 用显式 skip_prebuilt_rust_ext: true 替代空 artifact 名，并抽取 scripts/ci/utils/ensure_zstd.sh 统一 zstd 安装。

关键文件：

- .github/actions/download-rust-ext/action.yml（模块 CI 操作；类别 infra；类型 infrastructure）：新增复合操作，统一 artifact + cache 两条获取路径，是本次重构的核

心。

- .github/workflows/pr-test-multimodal-gen.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：多模态生成测试工作流，6 个 job 的重复下载逻辑全部折叠，集中体现调用点收敛。
- .github/workflows/pr-test-jit-kernel.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：jit-kernel 专项工作流，同样收敛 4 处重复逻辑，保证缓存回退覆盖。
- .github/workflows/pr-test-sgl-kernel.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：sgl-kernel 测试工作流，也是核心调用方之一，折叠后维护成本下降。
- .github/workflows/_pr-test-rust-ext-build.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：产物生产端，需要保证保存的 cache 条目能被读者用 zstd 读到。
- .github/workflows/_pr-test-stage-cpu.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：CPU 阶段按命中状态跳过 protoc + Rust 工具链，是本 PR 的行为变化点之一。
- .github/workflows/_pr-test-stage.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：通用 stage 新增 skip_prebuilt_rust_ext 输入，显式处理 aarch64 例外。
- scripts/ci/utils/ensure_zstd.sh（模块 CI 脚本；类别 infra；类型 infrastructure）：新增脚本，统一 cache 读写两侧的 zstd 安装，防止压缩工具差异导致缓存 miss。
- .github/workflows/rerun-test.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：workflow_dispatch 的重跑入口，从无法读 artifact 变为可命中 cache，cpu 作业跳过工具链安装，收益最大。
- .github/workflows/pr-test.yml（模块 CI 工作流；类别 infra；类型 infrastructure）：顶层工作流，将 gb300 阶段的 aarch64 例外改为显式 skip 输入。

关键符号：download-rust-ext, ensure_zstd.sh

评论区精华

本 PR 无专人 review 评论；唯一的交互来自 GitHub Actions bot 的 rerun-test 结果确认（test_quant_config_parsing.py 与 test_cuda_wrapper.py 均通过）。作者通过 /tag-and-rerun-ci 和 /rerun-test 触发了重跑，没有公开的设计争议。

- 暂无高价值评论线程

风险与影响

- 风险：主要风险集中在四点： 1) aarch64 阶段：cache 键不含架构，若新增调用点未设 skip_prebuilt_rust_ext 会拿到 x86_64 模块； 2) zstd 依赖：自托管 runner 缺 zstd 时缓存静默 miss,ensure_zstd.sh 安装失败仅告警，行为退回冷编译但不更差； 3) SGLANG_BUILD_RUST_EXTS 从 step 级 env 移到 GITHUB_ENV，作用域扩大，未来新读者需注意； 4) 仓库 cache 已达 10 GB 上限，缓存可能淘汰，但 artifact 优先能兜底。
  - 影响：影响的都是 CI 基础设施：所有导入 rust-ext 的测试工作流（jit-kernel、sgl-kernel、multimodal-gen、基础 stage、rerun-test）都会使用新操作，行为一致性提升，重复代码显著减少。开发者的直接收益是重跑过期 PR 或 workflow_dispatch 时

不再等待 cargo 全量编译，尤其是 cpu 作业托管 runner 的冷启动。对运行时代码无影响，风险集中在工作流 YAML 回归，但旧逻辑未删除，失败基本可回退。

- 风险标记：缓存键不区分架构，依赖 zstd 压缩工具，缓存易被淘汰，GITHUB_ENV 变量作用域变化

关联脉络

- 暂无明显关联 PR