PR #32434 完整报告

sgl-project/sglang

[core] Consolidate compiled-kernel caches under SGLANG_CACHE_DIR

合并时间：2026-08-06 04:54

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/32434

执行摘要

- 一句话：将第三方编译内核缓存统一到 SGLANG_CACHE_DIR 下
- 推荐动作：值得精读。该 PR 展示了如何在大型推理框架中干净地整合多个第三方 JIT 缓存的生命周期，关键设计包括：setdefault 尊重用户显式覆盖、用 callable 默认值延迟解析依赖顺序、把默认值映射抽成单一数据源供多模块复用，以及通过 review 发现并修复 diffusion 缓存隔离与 CI warmup marker 漂移等问题。需要关注 redirect_third_party_caches() 的调用时机约束（FlashInfer 导入前、Inductor 首次 cache_dir() 前），以及 @ch-wan 提出的 diffusion 缓存隔离方案，可复用于类似的多缓存管理场景。

功能与动机

PR body 指出：Triton、Inductor、FlashInfer 和 CUDA 驱动各自把编译内核写到各自的目录（~/.triton、~/.cache/flashinfer、/tmp/torchinductor_$USER、~/.nv），散落四处导致无法整体清理、预热、跨主机拷贝或按 volume 挂载，运维人员也无从得知这些目录在哪里。该想法来自 @ch-wan，目标是让每次运行产生的编译产物以单一目录形式管理。

实现拆解

实现按以下步骤拆解（最终合并包含 9 个文件，无对应测试文件）：

1. 集中定义缓存映射并重定向：在 python/sglang/srt/environ.py 新增 _default_cache_subdir(name)（返回 SGLANG_CACHE_DIR 下子目录的懒解析默认值）、third_party_cache_defaults()（返回 4 个第三方缓存环境变量的集中映射）和 redirect_third_party_caches()（对映射逐个 os.environ.setdefault）。同时把 SGLANG_DG_CACHE_DIR 的默认值从写死的 ~/.cache/deep_gemm 改为 lambda: _default_cache_subdir("deep_gemm")，使其跟随 SGLANG_CACHE_DIR。
2. 在导入早期触发重定向：在 python/sglang/__init__.py 顶部、任何会引入 torch/FlashInfer 的 import 之前调用 redirect_third_party_caches()，确保第三方库在 import 阶段固定缓存路径前拿到 sglang 默认值。这是整个方案生效的关键时序点。
3. 消除 DeepGEMM 缓存默认值的重复声明：python/sglang/srt/layers/deep_gemm_wrapper/compile_utils.py 中 DG_JIT_CACHE_DIR 的赋值从 os.getenv("SGLANG_DG_CACHE_DIR", 默认路径) 改为直接读 envs.SGLANG_DG_CACHE_DIR.get()；scripts/ci/cuda/warmup_deep_gemm.py 同样改

为从 envs 读取，并将 MARKER_DIR 派生自 SGLANG_CACHE_DIR，避免 marker 与缓存目录漂移。

4. 同步 CI 清理逻辑：scripts/ci/cuda/ci_install_dependency.sh 的 clean_site_packages() 现在清理 Inductor 缓存时同时覆盖环境变量指定位置、/tmp/torchinductor_$USER 和 {SGLANG_CACHE_DIR}/inductor 三处；warmup_server.py 改为从 warmup_deep_gemm 导入 MARKER_DIR，消除重复定义。
5. 兼容 diffusion 模块的缓存隔离：python/sglang/multimodal_gen/runtime/managers/gpu_worker.py 的 _configure_persistent_torch_compile_cache() 在判断"是否为用户显式覆盖"时，新增与 third_party_cache_defaults() 的精确比较，使 sglang 默认缓存路径不被误判为用户覆盖，diffusion 的编译缓存仍可落到 SGLANG_DIFFUSION_CACHE_ROOT/torch_compile_cache。
6. 启动提示与文档：python/sglang/srt/entrypoints/engine.py 新增 _log_legacy_kernel_cache_dirs()，在 launch_engine 末尾检测旧目录若存在则记录 info 日志（不删除、不触碰）；docs/docs/references/environment_variables.mdx 更新 SGLANG_CACHE_DIR 与 SGLANG_DG_CACHE_DIR 的说明。

测试配套：PR 最初新增了 test/registered/unit/test_environ_cache_dirs.py，经 review 讨论（Fridge003 认为不需要）后删除，最终无测试变更。验证以 PR body 中 E2E 手动验证（L4 studio 上确认 4 个缓存全部落于 ~/.cache/sglang 且旧目录无新产物）为准。

关键文件：

- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 core-logic；符号 _default_cache_subdir, third_party_cache_defaults, redirect_third_party_caches）：核心变更所在：集中定义第三方缓存路径映射并新增重定向函数，是本次统一缓存目录的枢纽。
- python/sglang/srt/entrypoints/engine.py（模块 引擎入口；类别 source；类型 core-logic；符号 _log_legacy_kernel_cache_dirs）：启动流程末尾新增旧缓存目录提示日志，帮助用户理解缓存位置迁移并避免误删其他框架使用的目录。
- python/sglang/multimodal_gen/runtime/managers/gpu_worker.py（模块 扩散工作器；类别 source；类型 dependency-wiring；符号 _configure_persistent_torch_compile_cache）：diffusion 模块缓存配置需适配新的 sglang 默认路径，否则会把默认值误认为用户覆盖，导致缓存隔离失效。
- python/sglang/__init__.py（模块 包入口；类别 source；类型 dependency-wiring）：在包导入最早期调用缓存重定向，是保证第三方库以 sglang 默认缓存路径为前提的关键时序点。
- python/sglang/srt/layers/deep_gemm_wrapper/compile_utils.py（模块 编译工具；类别 source；类型 core-logic）：DeepGEMM JIT 缓存目录赋值改为从 envs 读取，消除与 environ.py 默认值重复声明导致的漂移。
- scripts/ci/cuda/warmup_deep_gemm.py（模块 CI 预热；类别 infra；类型 infrastructure）：warmup 脚本的 MARKER_DIR 与 DG_JIT_CACHE_DIR 必须与服务器实际缓存目录同根，否则预热 marker 失去意义。
- scripts/ci/cuda/ci_install_dependency.sh（模块 CI 安装；类别 infra；类型 infrastructure）：Inductor 缓存清理需覆盖新位置，避免清理逻辑在缓存迁移后静默失效。

- scripts/ci/cuda/warmup_server.py（模块 CI 服务；类别 infra；类型 infrastructure）：消除 MARKER_DIR 重复定义，统一从 warmup_deep_gemm 导入。
- docs/docs/references/environment_variables.mdx（模块 环境变量文档；类别 other；类型 documentation）：同步环境变量文档，告知用户新默认路径与一次性重编译注意点。

关键符号：_default_cache_subdir, third_party_cache_defaults, redirect_third_party_caches, _log_legacy_kernel_cache_dirs, _configure_persistent_torch_compile_cache

关键源码片段

python/sglang/srt/environ.py

核心变更所在：集中定义第三方缓存路径映射并新增重定向函数，是本次统一缓存目录的枢纽。

```
# python/sglang/srt/environ.py

def _default_cache_subdir(name: str) -> str:
    """返回 SGLANG_CACHE_DIR 下的子目录，用于作为各缓存环境变量的默认值。

    必须以 callable 形式传给 EnvStr：SGLANG_CACHE_DIR 在 Envs 类体中声明
    在更靠后的位置，延迟求值既能保证跟随其值，也让测试可以覆盖它。
    """
    return os.path.join(os.path.expanduser(envs.SGLANG_CACHE_DIR.get()), name)


def third_party_cache_defaults() -> Dict[str, str]:
    """第三方 JIT 缓存默认位置的集中定义，供重定向和 diffusion 模块复用。

    只有这一处定义了默认映射，其他模块（如 diffusion 的缓存配置）通过
    比较这个映射来区分“sglang 默认值”与“用户显式覆盖”。
    """
    base = os.path.expanduser(envs.SGLANG_CACHE_DIR.get())
    return {
        "TRITON_CACHE_DIR": os.path.join(base, "triton"),
        "TORCHINDUCTOR_CACHE_DIR": os.path.join(base, "inductor"),
        "CUDA_CACHE_PATH": os.path.join(base, "nv"),
        # FlashInfer 会自行在基础目录下附加 ".cache/flashinfer",
        # 因此这里给的是基础目录，而不是最终缓存目录。
        "FLASHINFER_WORKSPACE_BASE": base,
    }


def redirect_third_party_caches():
    """把第三方 JIT 缓存指到 SGLANG_CACHE_DIR 下，让一次运行产生的编译产物
    可以作为一个目录整体清理、预热或挂载。

    必须在 FlashInfer 导入前（它在导入时解析 workspace）和 Inductor 第一次
    调用 cache_dir() 前（它会自己 setdefault TORCHINDUCTOR_CACHE_DIR），
```

```
    否则这里的 setdefault 不会生效；使用 setdefault 是为了尊重用户已经
    显式设置的环境变量。
    """
    for key, value in third_party_cache_defaults().items():
        os.environ.setdefault(key, value)
```

python/sglang/srt/entrypoints/engine.py

启动流程末尾新增旧缓存目录提示日志，帮助用户理解缓存位置迁移并避免误删其他框架使用的目录。

```
# python/sglang/srt/entrypoints/engine.py

def _log_legacy_kernel_cache_dirs():
    """提示旧的编译内核缓存目录已不再被 sglang 使用，但不触碰它们：
    同一机器上的其他框架可能仍在使用这些目录。"""
    # TODO(shuwang21): 等 SGLANG_CACHE_DIR 成为默认几个版本后删除此函数。
    legacy_dirs = [
        d
        for d in (
            os.path.expanduser("~/.triton"),
            os.path.expanduser("~/.cache/flashinfer"),
            os.path.expanduser("~/.cache/deep_gemm"),
        )
        if os.path.isdir(d)
    ]
    if not legacy_dirs:
        return
    logger.info(
        "Compiled-kernel caches now live under SGLANG_CACHE_DIR (%s). These "
        "older directories are no longer used by sglang, but may still be "
        "used by other frameworks on this machine, so they were left alone: "
        "%s. Remove them yourself if nothing else needs them.",
        envs.SGLANG_CACHE_DIR.get(),
        ", ".join(legacy_dirs),
    )
```

python/sglang/multimodal_gen/runtime/managers/gpu_worker.py

diffusion 模块缓存配置需适配新的 sglang 默认路径，否则会把默认值误认为用户覆盖，导致缓存隔离失效。

```
# python/sglang/multimodal_gen/runtime/managers/gpu_worker.py

def _configure_persistent_torch_compile_cache(self) -> None:
    """持久化 torch.compile 的 Inductor/Triton 缓存，使跨重启可复用。"""
    compile_cache_root = os.path.join(
        envs.SGLANG_DIFFUSION_CACHE_ROOT, "torch_compile_cache"
    )
    tmp_root = tempfile.gettempdir()
    # sglang 默认路径与用户显式覆盖区分开：environ.py 的
```

```
    # redirect_third_party_caches() 会用 SGLANG_CACHE_DIR 下的路径调用
    # setdefault，如果这里不识别它，会把 sglang 默认值误当作"用户显式
    # 覆盖"而不再重定向到 SGLANG_DIFFUSION_CACHE_ROOT，破坏 diffusion 的
    # 缓存隔离。
    sglang_defaults = third_party_cache_defaults()
    for env_name, sub in (
        ("TORCHINDUCTOR_CACHE_DIR", "inductor"),
        ("TRITON_CACHE_DIR", "triton"),
    ):
        current = os.environ.get(env_name)
        if (
            current
            and current != sglang_defaults.get(env_name)
            and not current.startswith(tmp_root)
        ):
            # 尊重用户显式提供且非临时目录的缓存位置。
            continue
        cache_path = os.path.join(compile_cache_root, sub)
        try:
            os.makedirs(cache_path, exist_ok=True)
        except OSError as e:
            logger.warning(
                "Could not create torch.compile cache dir %s: %s", cache_path, e
            )
            continue
        os.environ[env_name] = cache_path
    logger.info(
        "torch.compile cache: TORCHINDUCTOR_CACHE_DIR=%s TRITON_CACHE_DIR=%s",
        os.environ.get("TORCHINDUCTOR_CACHE_DIR"),
        os.environ.get("TRITON_CACHE_DIR"),
    )
```

评论区精华

Review 中最有价值的讨论集中在以下几个线程（均已解决）：

- diffusion 缓存隔离被破坏（ch-wan 在 environ.py 指出）：
  redirect_third_party_caches() 的 setdefault 会让 diffusion 的 _configure_persistent_torch_compile_cache 将 {SGLANG_CACHE_DIR}/inductor 等值误认为"用户显式覆盖"而不再重定向到 SGLANG_DIFFUSION_CACHE_ROOT，导致 diffusion 编译缓存静默迁移且删除 SGLANG_DIFFUSION_CACHE_ROOT 不再生效。作者采纳方案 (a)，抽取 third_party_cache_defaults() 让 diffusion 精确比较，保留缓存隔离。
- presharded checkpoint 缓存移入共享目录的 P1 风险（codex bot 在 loader.py 指出）：
  相对模型路径在同一根目录下哈希可能冲突，且多节点共享存储但各节点 home 不同会导致 manifest 分散。作者答复"nice catch, will fix this in the follow up pr #32437"，该部分未包含在本 PR 合并中。

- warmup marker 与缓存目录漂移（ch-wan 在 warmup_deep_gemm.py 指出）：MARKER_DIR 硬编码 ~/.cache/sglang/warmup_markers，覆盖 SGLANG_CACHE_DIR 后 marker 与 DG_JIT_CACHE_DIR 分家，会引入 stale marker/ 冷编译问题。作者修复并从 envs 派生，还顺带发现 ci_install_dependency.sh 清理 Inductor 缓存路径失效，一并在本 PR 修复。
- 调用方式（Fridge003 在 init.py 建议）：不要用 import environ 的副作用来触发重定向，改为显式调用 redirect_third_party_caches()。作者照做并移除了 environ.py 的模块级副作用。
- 注释准确性（ch-wan 的 nit）：Inductor 写入 TORCHINDUCTOR_CACHE_DIR 的时机是首次 cache_dir() 调用而非 import 时；environ 也非纯 stdlib。作者修正措辞。
- 测试文件（Fridge003）：认为 test_environ_cache_dirs.py 无必要，作者删除。
- diffusion 缓存隔离被 SGLANG_CACHE_DIR 默认值破坏 (correctness): 作者采纳方案 (a)：抽取 third_party_cache_defaults() 作为唯一映射源，diffusion 逻辑比较精确相等后仍将缓存重定向到 SGLANG_DIFFUSION_CACHE_ROOT。
- presharded checkpoint 缓存移入共享根目录的 P1 风险 (correctness): 作者确认并约定在 follow-up PR #32437 中修复，本 PR 未包含 loader.py 相关改动。
- warmup marker 硬编码与缓存目录漂移 (correctness): 作者修复 MARKER_DIR 从 envs.SGLANG_CACHE_DIR 派生，并顺带发现 ci_install_dependency.sh 清理路径失效而一并修复。
- SGLANG_DG_CACHE_DIR 默认变更导致既有 warm cache 孤儿 (documentation): 作者更新环境变量文档与 PR body，添加一次性重编译说明，并在启动日志中列出旧目录。
- 缓存重定向的调用方式 (design): 作者改为在 init.py 显式调用 redirect_third_party_caches()，并移除 environ.py 的模块级副作用。
- 是否保留新增的环境变量单测 (testing): 测试文件删除，无对应自动化测试。
- 注释准确性与模块纯度 (documentation): 作者修正注释措辞为“no heavy dependency (no torch)”，并更新 redirect 函数 docstring 中的时序说明。

风险与影响

- 风险：
  1. 升级后一次性重编译（高确定性风险）：所有用户的 Triton/Inductor/FlashInfer/DeepGEMM 缓存位置变更，升级后首次启动必然重新编译，DeepGEMM 与 FlashInfer 编译代价高。PR 已通过启动日志与文档提示，但用户若未预拷贝旧缓存会面临一次明显变慢的冷启动。
  2. diffusion 缓存隔离逻辑依赖精确比较：gpu_worker.py 依赖 third_party_cache_defaults() 的返回值与 os.environ 中值相等判断是否用户覆盖。若未来 SGLANG_CACHE_DIR 解析逻辑变化或用户设置的值恰好等于默认值（虽然是显式设置），会走重定向分支，行为可接受但存在脆性。
  3. 多节点共享存储部署：presharded checkpoint 缓存位置在原始方案中移入 {SGLANG_CACHE_DIR}/presharded/<model_id>，codex 指出多节点各 home 不同会导致 manifest 分散，该问题虽移到 #32437 修复，但最终合并的 PR 若未包含

loader.py 改动则该风险不适用于本版本，但作为后续演进需跟踪。

4. CI 脚本与文档同步风险：缓存路径改动涉及 CI 预热、清理脚本多处，ci_install_dependency.sh 已改为覆盖多位置，但仍依赖 SGLANG_CACHE_DIR 未设置时回落 ~/.cache/sglang，若 CI 环境显式设置了别的路径则清理范围可能不全。
5. 无自动化测试覆盖：本 PR 最终无对应测试，缓存目录行为只能靠 E2E 手动验证，后续修改容易回归。 - 影响：影响范围覆盖所有 sglang 用户与部署形态：任何一次 import sglang 或 sglang serve 都会通过 __init__.py 设置第三方编译缓存环境变量；CI 的 DeepGEMM 预热、Inductor 清理、diffusion 的 torch.compile 缓存也随之变化。对普通用户而言是一次性重编译成本；对运维而言获得统一缓存目录，可整体清理、预热、volume 挂载或跨主机拷贝；对 diffusion 用户而言缓存隔离保持原设计，但删除 SGLANG_DIFFUSION_CACHE_ROOT 的预期行为需要依赖 third_party_cache_defaults() 比较逻辑正确工作。对团队而言，后续所有第三方缓存路径的来源统一到 environ.py 一处，减少重复默认值漂移。 - 风险标记：核心路径启动时序变更，升级后一次性重编译，diffusion 缓存隔离依赖精确比较，CI 脚本多位置同步，无自动化测试覆盖，多节点共享默认缺失（跟踪 follow-up）

关联脉络

- PR #32437 Follow-up fix for presharded checkpoint cache under SGLANG_CACHE_DIR: review 中 codex bot 指出的 presharded 缓存 P1 问题，作者明确约定在本 follow-up PR 修复，与本 PR 属于同一缓存统一工作线。
- PR #33637 [CI] Skip sglang-kernel and sgl-deep-gemm reinstall on version match: 与本 PR 都修改了 scripts/ci/cuda/ci_install_dependency.sh，涉及 CI 安装与缓存处理，存在同一脚本的连续演进。
- PR #33546 [diffusion] Wan VAE RMSNorm+SiLU fusion behind quality=high (H200 FastWan2.2 e2e 9.611 -> 9.125 s): 同为 diffusion 模块内的编译缓存 / 内核优化相关变更，本 PR 对 diffusion torch.compile 缓存位置的处理与后续 diffusion 性能优化共享 gpu_worker.py 上下文。