PR #27344 完整报告

sgl-project/sglang

[CI] Isolate CUDA coredump dir per run to fix tracker mis-attribution

合并时间：2026-06-06 09:31

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/27344

执行摘要

- 一句话：按 CI run 隔离 CUDA coredump 目录，修复误报
- 推荐动作：建议合并。该 PR 针对性的修复了 CI 基础设施中的一个实际问题，变更紧凑、逻辑清晰。值得关注的是 producer 和 uploader 之间通过注释保持目录解析逻辑一致的设计，以及通过环境变量 RUNNER_TEMP 利用 CI 自带的 per-job 临时目录机制。

功能与动机

修复自托管 runner 上 CUDA coredump 追踪器误报问题：一个 job 留下的 dump 文件会被后续无关 job 捡起并报告，导致追踪器指向错误的 PR/job。PR body 中描述了具体失败场景：健康检查失败后的 job 未产生任何 dump，但仍因前序 job 的残留文件而错误地发布 coredump 评论。

实现拆解

1. python/sglang/srt/debug_utils/cuda_coredump.py 中 get_dump_dir() 重写：从简单的返回固定环境变量改为三层 fallback 逻辑：优先取 SGLANG_CUDA_COREDUMP_DIR（用户显式设置），其次取 RUNNER_TEMP（CI per-job 临时目录，自动清理），最后 fallback 到 /tmp/sglang_cuda_coredumps。在 CI 环境下（GITHUB_RUN_ID 存在），进一步追加 <run_id>-<attempt> 子目录，实现 per-run 隔离。
2. python/sglang/srt/environ.py 中 SGLANG_CUDA_COREDUMP_DIR 默认值调整：从硬编码的 "/tmp/sglang_cuda_coredumps" 改为 None，使 get_dump_dir() 动态解析，避免 producer 与 uploader 默认值不一致。
3. .github/actions/upload-cuda-coredumps/action.yml 同步更新：镜像 get_dump_dir() 的逻辑，在 action 的 check step 中用相同规则计算 dump_dir，并修改后续的 upload artifact / tracker issue / cleanup 步骤，全部引用该动态目录而非固定环境变量。

关键文件：

- python/sglang/srt/debug_utils/cuda_coredump.py（模块 调试工具；类别 source；类型 core-logic）：核心变更：get_dump_dir() 重写，实现 per-run 隔离的目录解析逻辑。
- python/sglang/srt/environ.py（模块 环境配置；类别 source；类型 configuration）：修改 SGLANG_CUDA_COREDUMP_DIR 默认值为 None，使 get_dump_dir() 能够动态解析目录。

- .github/actions/upload-cuda-coredumps/action.yml（模块 CI 动作；类别 infra；类型 infrastructure）：同步更新 action 中的目录计算逻辑，确保 producer 和 uploader 一致。

关键符号：get_dump_dir

关键源码片段

python/sglang/srt/debug_utils/cuda_coredump.py

核心变更：get_dump_dir() 重写，实现 per-run 隔离的目录解析逻辑。

```
"""CUDA coredump helpers.

When SGLANG_CUDA_COREDUMP=1, this module injects CUDA coredump environment
variables into the current process so that GPU exceptions (e.g. illegal
memory access) produce lightweight coredump files for post-mortem analysis
with cuda-gdb.

The injection happens at module import time via _inject_env() on a
best-effort basis.  If any CUDA_* variable is already present in the
environment (e.g. set by the user in the shell), injection is skipped for
that variable and a warning is printed.  For strict guarantees, set the
CUDA_* env vars in the shell before launching Python.
"""

import glob
import os
import warnings

from sglang.srt.environ import envs

_CUDA_COREDUMP_FLAGS = (
    "skip_nonrelocated_elf_images,skip_global_memory,"
    "skip_shared_memory,skip_local_memory,skip_constbank_memory"
)


def is_enabled() -> bool:
    return envs.SGLANG_CUDA_COREDUMP.get()


def get_dump_dir() -> str:
    # Resolve the base dir the same way as the uploader
    # (.github/actions/upload-cuda-coredumps/action.yml) so they agree; an empty
    # SGLANG_CUDA_COREDUMP_DIR counts as unset, like the action's `[ -n ... ]`.
    explicit = envs.SGLANG_CUDA_COREDUMP_DIR.get()
    runner_temp = os.getenv("RUNNER_TEMP")
    if explicit:
        base = explicit
    elif runner_temp:
```

```
        base = os.path.join(runner_temp, "sglang_cuda_coredumps")
    else:
        base = "/tmp/sglang_cuda_coredumps"
    # Isolate dumps per (run, attempt): on a shared self-hosted runner a leftover
    # dump from one job must not be picked up and mis-attributed by a later one.
    run_id = os.getenv("GITHUB_RUN_ID")
    if run_id:
        attempt = os.getenv("GITHUB_RUN_ATTEMPT", "1")
        return os.path.join(base, f"{run_id}-{attempt}")
    return base


def _inject_env():
    """Inject CUDA coredump environment variables into the current process.
    If a CUDA_* variable is already present, skip it and log a warning."""
    dump_dir = get_dump_dir()
    os.makedirs(dump_dir, exist_ok=True)

    env_vars = {
        "CUDA_ENABLE_COREDUMP_ON_EXCEPTION": "1",
        "CUDA_COREDUMP_SHOW_PROGRESS": "1",
        "CUDA_COREDUMP_GENERATION_FLAGS": _CUDA_COREDUMP_FLAGS,
        "CUDA_COREDUMP_FILE": f"{dump_dir}/cuda_coredump_%h.%p.%t",
    }
    for key, value in env_vars.items():
        if key in os.environ:
            warnings.warn(
                f"CUDA coredump env var {key} is already set to "
                f"'{os.environ[key]}', skipping injection of '{value}'.",
                stacklevel=2,
            )
        else:
            os.environ[key] = value


def cleanup_dump_dir():
    """Remove stale coredump files from the dump directory."""
    dump_dir = get_dump_dir()
    for f in glob.glob(os.path.join(dump_dir, "cuda_coredump_*")):
        os.remove(f)
```

评论区精华

无 review 评论。提交历史显示作者在第二个 commit 添加了 get_dump_dir 单元测试，但在第四个 commit 又删除了，可能因测试不适合放入该路径或与 CI 环境交互复杂。

- 暂无高价值评论线程

风险与影响

- 风险：
  1. 回归风险：get_dump_dir() 逻辑变更影响所有调用方，包括 _inject_env() 和 cleanup_dump_dir()。覆盖测试不足（单元测试被移除），CI 中若未充分测试可能引入路径不匹配问题。
  2. 兼容性：SGLANG_CUDA_COREDUMP_DIR 默认值从 /tmp/sglang_cuda_coredumps 改为 None 可能导致依赖该默认值的用户行为改变，但 fallback 逻辑保证了无 CI 环境时仍用 /tmp/sglang_cuda_coredumps。
  3. CI 环境变量依赖：RUNNER_TEMP 和 GITHUB_RUN_ID 在非标准 CI 环境可能缺失，但已有健全 fallback。 - 影响：影响范围：低。仅影响启用了 SGLANG_CUDA_COREDUMP=1 的 CI job。修复了 dump 追踪器跨 job 误报的核心问题，提升 CI 信号可靠性。影响程度：中等。对开发者和 CI 运维人员友好，避免了手动清理或排查误报的繁琐工作。 - 风险标记：缺少测试覆盖，环境变量依赖

关联脉络

- PR #26340 CUDA Coredump Tracker: 该 PR 为 tracker issue，用于收集 coredump 事件。本 PR 修复了 tracker 误报的根本原因——共享目录导致的跨 job 归因错误。