PR #26757 完整报告

sgl-project/sglang

Trigger scheduler diagnostics on health failure

合并时间：2026-06-04 08:19

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/26757

执行摘要

- 一句话：健康检查失败时触发调度器 py-spy 与 CUDA coredump 诊断
- 推荐动作：建议所有涉及服务可靠性的团队阅读此 PR，尤其是 dump_requests_before_crash 和 _handle_crash_dump_env 的逻辑。其模块化设计和环境变量门控策略值得借鉴。

功能与动机

在健康检查失败时，现有的诊断信息覆盖了整个进程，不够聚焦于scheduler/TPworker进程。该 PR 旨在将诊断范围精确限定到 scheduler 进程，并支持独立开关，便于线上问题定位而无需改代码。

实现拆解

1. 新增诊断工具模块：在 utils/cudacore_pyspy_dump_utils.py 中实现核心函数：collect_scheduler_processes 通过 psutil 遍历子进程并筛选出以 sglang::scheduler 开头的进程；pyspy_dump_schedulers 对筛选出的进程执行 py-spy dump（含 --native 回退）；trigger_cuda_user_coredump 向 CUDA coredump 管道写入触发信号；_resolve_cuda_coredump_pipe_path 解析管道路径。
2. 改造 crash dump 流程：在 tokenizer_manager.py 的 dump_requests_before_crash 方法中，新增两个环境变量 SGLANG_PYSPY_DUMP_BEFORE_CRASH 和 SGLANG_CUDA_COREDUMP_BEFORE_CRASH 作为进入诊断分支的新条件。当诊断启用时，先调用 pyspy_dump_schedulers(scheduler_only=True) 和 trigger_cuda_user_coredump(scheduler_only=True)，再执行原有的请求信息转储。还引入了 SGLANG_CUDA_COREDUMP_BEFORE_CRASH_WAIT_SECS 作为等待延迟。
3. 环境变量声明：在 environ.py 中注册上述三个新环境变量，默认值均为 True（等待秒数默认 60.0）。
4. 自动设置 CUDA coredump 环境：在 server_args.py 新增 _handle_crash_dump_env 方法，在 __post_init__ 中调用。若用户设置了 --crash-dump-folder，自动注入 CUDA_ENABLE_COREDUMP_ON_EXCEPTION、CUDA_ENABLE_USER_TRIGGERED_COREDUMP 等环境变量，并创建以 hostname 命名的子目录。

5. 代码清理与导入更新：将原来位于 utils/common.py 的旧版 pyspy_dump_schedulers 删除；将 watchdog.py 中的导入路径指向新模块。

关键文件：

- python/sglang/srt/utils/cudacore_pyspy_dump_utils.py（模块 诊断工具；类别 source；类型 new-module；符号 _resolve_cuda_coredump_pipe_path, _is_sglang_scheduler_process, collect_scheduler_processes, pyspy_dump_schedulers）：新增文件，包含所有诊断核心逻辑：进程收集、py-spy dump、CUDA coredump 触发。
- python/sglang/srt/managers/tokenizer_manager.py（模块 分词管理器；类别 source；类型 dependency-wiring；符号 dump_requests_before_crash）：核心入口，在健康检查失败的 dump_requests_before_crash 方法中集成了 py-spy dump 和 CUDA coredump。
- python/sglang/srt/server_args.py（模块 服务参数；类别 source；类型 core-logic；符号 _handle_crash_dump_env）：新增 _handle_crash_dump_env 方法，当设置 --crash-dump-folder 时自动配置 CUDA coredump 环境变量并创建目录。
- python/sglang/srt/environ.py（模块 环境变量；类别 source；类型 configuration）：声明三个新环境变量，作为诊断功能的全局开关。
- python/sglang/srt/utils/common.py（模块 通用工具；类别 source；类型 refactor；符号 pyspy_dump_schedulers）：删除了原有的 pyspy_dump_schedulers 函数，功能完整迁移至新模块。
- python/sglang/srt/utils/watchdog.py（模块 看门狗；类别 source；类型 dependency-wiring）：更新导入路径，指向新的诊断模块。

关键符号：_resolve_cuda_coredump_pipe_path, _is_sglang_scheduler_process, collect_scheduler_processes, pyspy_dump_schedulers, trigger_cuda_user_coredump, _handle_crash_dump_env, dump_requests_before_crash

关键源码片段

python/sglang/srt/managers/tokenizer_manager.py

核心入口，在健康检查失败的 dump_requests_before_crash 方法中集成了 py-spy dump 和 CUDA coredump。

```
def dump_requests_before_crash(
    self,
    hostname: str = os.getenv("HOSTNAME", socket.gethostname()),
):
    """在进程崩溃前执行诊断：py-spy dump 和 CUDA coredump，然后 dump 请求信息。"""
    # 读取诊断开关环境变量（均在 environ.py 中声明，默认 True）
    should_dump_pyspy = envs.SGLANG_PYSPY_DUMP_BEFORE_CRASH.get()
    should_dump_cuda_coredump = envs.SGLANG_CUDA_COREDUMP_BEFORE_CRASH.get()
    should_dump_diagnostics = should_dump_pyspy or should_dump_cuda_coredump

    # 如果既没有设置 crash_dump_folder，也没有启用诊断，则直接返回
    if not self.crash_dump_folder and not should_dump_diagnostics:
        return
```

```
    if self.crash_dump_performed:
        return
    else:
        self.crash_dump_performed = True

    # 先执行 py-spy dump（仅 scheduler 进程）
    if should_dump_pyspy:
        # 使用精确的 scheduler_only=True，只 dump scheduler 进程
        pyspy_dump_schedulers(scheduler_only=True)

    # 再触发 CUDA coredump（仅 scheduler 进程）
    if should_dump_cuda_coredump:
        # 等待一小段时间让 py-spy 完成，避免同时操作影响
        time.sleep(envs.SGLANG_CUDA_COREDUMP_BEFORE_CRASH_WAIT_SECS.get())
        trigger_cuda_user_coredump(scheduler_only=True)

    # 接着执行原有的请求信息转储逻辑
    if self.crash_dump_folder:
        logger.error(f"Dumping requests before crash. {self.crash_dump_folder=}")
        # ... 原有请求 dump 代码
```

python/sglang/srt/server_args.py

新增 _handle_crash_dump_env 方法，当设置 --crash-dump-folder 时自动配置 CUDA coredump 环境变量并创建目录。

```
  def _handle_crash_dump_env(self):
    """当设置了 crash_dump_folder 时，自动注入 CUDA coredump 环境变量"""
    if not self.crash_dump_folder:
        return

    # 默认 CUDA coredump 配置（除非用户已在环境中显式设置）
    _CUDA_COREDUMP_DEFAULTS = {
        "CUDA_ENABLE_COREDUMP_ON_EXCEPTION": "1",
        "CUDA_ENABLE_USER_TRIGGERED_COREDUMP": "1",
        "CUDA_COREDUMP_SHOW_PROGRESS": "1",
        "CUDA_COREDUMP_GENERATION_FLAGS": (
            "skip_nonrelocated_elf_images,skip_global_memory,"
            "skip_shared_memory,skip_local_memory,skip_constbank_memory"
        ),
        "CUDA_COREDUMP_FILE": f"{self.crash_dump_folder}/%h/core.cuda.%t.%p",
        "CUDA_COREDUMP_PIPE": "/tmp/corepipe.cuda.%h.%p",
    }
    for key, value in _CUDA_COREDUMP_DEFAULTS.items():
        if key not in os.environ:
            os.environ[key] = value
            logger.info("Auto-set %s=%s (from --crash-dump-folder)", key, value)

    if key == "CUDA_COREDUMP_FILE":
        # CUDA coredump 要求目标目录存在，预先创建以 hostname 命名的子目录
```

```
    hostname = socket.gethostname()
    os.makedirs(
        os.path.join(self.crash_dump_folder, hostname),
        exist_ok=True,
    )
```

评论区精华

该 PR 无实质性 review 讨论，仅涉及操作评论（merrymercy 通过 /rerun-test 触发回归测试）。设计决策隐含在 commit 和代码中。

- 测试验证方式 (testing): 测试通过，PR 被合并。

风险与影响

- 风险：
  1. 磁盘占用风险：CUDA coredump 文件较大（测试中单文件 255MB），若健康检查频繁失败可能快速占满磁盘。建议在生产环境按需启用 SGLANG_CUDA_COREDUMP_BEFORE_CRASH=0。
  2. 默认启用影响：所有用户默认会开启这两个诊断，可能对性能敏感场景有轻微影响（py-spy 会暂停进程数百毫秒）。可通过环境变量关闭。
  3. 仅支持 Linux：CUDA coredump 机制及 psutil 进程遍历依赖 Linux 特定特性，在其他平台可能降级或报错。
  4. 环境变量冲突：若用户未设置 CUDA_ENABLE_USER_TRIGGERED_COREDUMP 但通过 --crash-dump-folder 自动注入，可能覆盖用户的显式配置。 - 影响：影响范围中等：仅当健康检查返回 503 时触发，正常请求无影响。对启用诊断的用户，可获得更精确的 scheduler 级别堆栈和 GPU 状态，极大加速问题排查。诊断默认开启，但可通过环境变量关闭，不影响现有行为。代码重构将相关函数独立为模块，提高了可维护性。 - 风险标记：CUDA coredump 可能产生大文件，诊断默认开启，可能影响性能，仅 Linux 平台支持，环境变量自动注入可能覆盖用户配置

关联脉络

- PR #27428 [debug] Register #27338 EAGLE draft kv_indices revert in pr_fix_toggle: 同样属于调试诊断工具链的增强，与本 PR 共同构成了 SGLang 的可观测性与调试基础设施。