

PR #39560 完整报告

sgl-project/sglang

[Fix] Release NCCL on scheduler exit and let the ASGI server own shutdown

合并时间: 2026-09-16 04:39

原文链接: <http://prhub.com.cn/sgl-project/sglang/pull/39560>

执行摘要

- 一句话: 修复关闭路径: NCCL 主动中止, ASGI 服务器接管退出
- 推荐动作: 值得精读, 尤其关注两个设计决策: 一是 'abort vs destroy' 的权衡——关闭路径上 peer 已退出是常态, 因此选择本地中止而非集体式销毁, 并用 '未初始化返回、无 API 降级、异常仅告警' 的就地降级策略替代开关控制; 二是 'hook 交接退出权' 的模式——不在 asyncio 任务里 `sys.exit(0)`, 而是通过 `set_server_stop_hook()` 让真正的服务器对象 (uvicorn Server 或 granian Server) 执行 lifespan shutdown, 同时先 cancel 自身任务避免 pending 任务告警。这套模式对任何 '信号处理 vs 服务器生命周期' 耦合的服务都有借鉴价值。建议阅读时对照 commit 历史, 可以看到作者从加 kill switch 到删 kill switch 的收敛过程。

功能与动机

PR body 明确列出了两个背景问题: 一是 `run_scheduler_process` 在优雅路径上释放 host buffers 但从不 drop 进程组, 干净停止时会打印 `destroy_process_group() was not called before program exit`; 二是 `TokenizerManager.sigterm_watchdog` 以 `sys.exit(0)` 收尾, 而 `SystemExit` 是 `BaseException`, 会逃逸任务包装并停止 ASGI 服务器正在等待的事件循环, 导致 lifespan shutdown 永远走不到, 每次干净停止都在 ERROR 级别打印 `CancelledError` traceback。此外, 超时后 SIGKILL 落在 release 过程中, 正是 GPU 显存残留的根源, 因此需要让 scheduler 在优雅路径主动释放 NCCL 通信资源, 并让服务器拥有退出权。

实现拆解

整个修复分为四条链路, 按依赖顺序拆解如下:

1. 调度器侧: 新增非阻塞的通信组中止。在 `python/sglang/srt/distributed/parallel_state.py` 新增 `abort_distributed_environment()`: 若 `torch.distributed.is_initialized()` 为假则直接返回; 随后通过 `getattr(torch.distributed.distributed_c10d, "_abort_process_group", None)` 拿到中止函数, 老版本 torch 没有该私有 API 时原地降级返回, 调用异常则记录 warning 而不上抛。之所以用 abort 而不是已有的 `destroy_distributed_environment()`, 是因为 `destroy_process_group` 是集体操作, shutdown 路径上 peer 已经退出的情况很常见, 会阻塞等待。
2. scheduler 退出流程接线。在 `python/sglang/srt/managers/scheduler.py` 的 `run_scheduler_process` 的 `finally` 块中, 先保持原有逻辑 (异常路径不释放 GPU 以免 `synchronize()` 挂死), 在 `scheduler.gracefully_exit` 为真时先调用

`scheduler.release_host_resources()`，最后调用 `abort_distributed_environment()`。注释明确说明顺序原因：release 过程可能仍需要可用的通信器，所以 abort 必须放在最后。

3. TokenizerManager 侧：新增停止 hook 并把退出交还服务器。在 `python/sglang/srt/managers/tokenizer_manager.py` 中，将 `_server_stop_hook` 声明为类级变量（避免触碰冻结的 `__init__`），新增 `set_server_stop_hook()`。`sigterm_watchdog` 收尾处做了三件事：把硬编码的 15 秒超时提取为 `_SCHEDULER_EXIT_TIMEOUT_SECS` 常量；等待结束后收集仍在存活的 scheduler PID 并打 warning，点名

关键文件：

- `python/sglang/srt/managers/tokenizer_manager.py`（模块 请求管理；类别 source；类型 core-logic；符号 `set_server_stop_hook`, `sigterm_watchdog`）：核心改动所在：新增类级 `_server_stop_hook` 与 `set_server_stop_hook()`，重构 `sigterm_watchdog` 收尾逻辑——提取 15 秒超时常量、点名未退出 scheduler PID、最后把退出权交给 ASGI 服务器而非 `sys.exit(0)`，是 '让服务器拥有 shutdown' 的直接实现点。
- `python/sglang/srt/distributed/parallel_state.py`（模块 分布式；类别 source；类型 core-logic；符号 `abort_distributed_environment`）：新增 `abort_distributed_environment()`，用 torch 私有 API `_abort_process_group` 本地中止所有通信组，取代会阻塞的集体式 `destroy_process_group`，是修复 `destroy_process_group() was not called before program exit` 告警的关键。
- `python/sglang/srt/entrypoints/http_server.py`（模块 HTTP 服务；类别 source；类型 core-logic）：三条服务器路径的接线点：uvicorn 默认路径从 `uvicorn.run()` 改为 `Server + server.run()` 以获得可停止的对象，SSL-refresh 与 granian 路径分别挂上对应 hook；多 worker 路径明确不接线。
- `python/sglang/srt/managers/scheduler.py`（模块 调度器；类别 source；类型 dependency-wiring）：把 `abort_distributed_environment()` 接入 `run_scheduler_process` 的 `finally` 块，放在 `release_host_resources()` 之后——因为释放过程可能仍需要可用的通信器，中止必须最后执行。
- `python/sglang/test/test_utils.py`（模块 测试工具；类别 test；类型 test-coverage）：配套可观测性改进：`wait_for_gpu_release` 新增实际等待时长上报，超过 1 秒阈值即打印，让慢速 GPU 释放不再静默，与 PR 39545 引入的该函数形成接力。

关键符号：`set_server_stop_hook`, `abort_distributed_environment`, `sigterm_watchdog`, `run_scheduler_process`, `wait_for_gpu_release`

关键源码片段

`python/sglang/srt/managers/tokenizer_manager.py`

核心改动所在：新增类级 `_server_stop_hook` 与 `set_server_stop_hook()`，重构 `sigterm_watchdog` 收尾逻辑——提取 15 秒超时常量、点名未退出 scheduler PID、最后把退出权交给 ASGI 服务器而非 `sys.exit(0)`，是 '让服务器拥有 shutdown' 的直接实现点。

```
# class TokenizerManager 内，类级变量：由拥有事件循环的一方设置，
# Engine 与 gRPC 不拥有服务器，保持 None。放在类级是为了避开被冻结的 __init__。
```

`_server_stop_hook: Optional[Callable[[], None]] = None`

```
def set_server_stop_hook(self, hook: Callable[[], None]) -> None:
    # ASGI 服务器 (uvicorn / granian) 通过这个 hook 接管退出,
    # 避免在 asyncio 任务里调用 sys.exit(0) 提前杀死事件循环。
    self._server_stop_hook = hook

# sigterm_watchdog 收尾部分 (本 PR 重构后的形态)
# 先派发 ShutdownReq 让 scheduler 在用户态释放资源并退出,
# 然后等待宽限期结束, 最后兜底杀进程树。
self._dispatch_to_scheduler(ShutdownReq())
deadline = time.monotonic() + _SCHEDULER_EXIT_TIMEOUT_SECS
while time.monotonic() < deadline and collect_scheduler_processes():
    time.sleep(0.1)
stragglers = [proc.pid for proc in collect_scheduler_processes()]
if stragglers:
    # SIGKILL 落在释放过程中, 正是 GPU 显存残留的根源。
    # 这里先把仍存活的 PID 点名, 让慢速释放不再静默。
    logger.warning(
        f"Schedulers still alive {_SCHEDULER_EXIT_TIMEOUT_SECS}s after "
        f"ShutdownReq, killing them before they released: {stragglers}"
    )
kill_process_tree(os.getpid(), include_parent=False, wait_timeout=60)
if self._server_stop_hook is not None:
    # sys.exit(0) 抛出的 SystemExit 会摧毁事件循环,
    # 导致 ASGI 服务器的 lifespan shutdown 永远不执行。
    # 事件循环现在还能活着, 先取消自己的任务, 否则 pending 的
    # handle_loop 会被报为 destroyed-while-pending; 再把退出交给服务器。
    current = asyncio.current_task()
    for task in self.asyncio_tasks:
        if task is not current:
            task.cancel()
    self._server_stop_hook()
    return
# Engine / gRPC 不拥有服务器, 保持直接退出。
sys.exit(0)
```

python/sglang/srt/distributed/parallel_state.py

新增 `abort_distributed_environment()`, 用 torch 私有 API `_abort_process_group` 本地中止所有通信组, 取代会阻塞的集体式 `destroy_process_group`, 是修复 `destroy_process_group() was not called before program exit` 告警的关键。

```
def abort_distributed_environment() -> None:
    """本地释放本 rank 的通信器。

    destroy_process_group 是集体操作, peer 已退出时会阻塞等待,
    而关闭路径上 peer 已退出是常见情况, 所以这里改用 abort。
    """
    if not torch.distributed.is_initialized():
```

```

    return
# _abort_process_group 属于 torch 私有 API，老版本可能不存在；
# 不存在就原地降级返回，而不是退回会阻塞的集体式销毁。
abort = getattr(torch.distributed.distributed_c10d, "_abort_process_group", None)
if abort is None:
    return
try:
    # 不传参数即中止全部进程组，包括默认的 world group。
    abort()
except Exception as e:
    # 关闭路径上的失败不应放大为新的异常，记录告警即可。
    logger.warning(f"NCCL abort on shutdown failed, {type(e).__name__}: {e}")

```

评论区精华

本 PR 没有实质性的 review 评论线程 (`review_comments_count: 0`)，Issue 评论只有一条 `/rerun-test test_deepep_small.py test_lora_moe_tp_logprob_diff.py` 指令及机器人回报两条用例通过的结果，设计权衡主要沉淀在代码注释与 commit 演进中。

值得注意的演进信号来自 9 个 commit 的序列：先是逐个提交

- CI 重跑指令与结果 (other): 两条用例均通过，无未解决的 review 疑虑；设计权衡主要通过 9 个 commit 的演进体现，尤其是最后一个 commit 删除 NCCL abort kill switch，改成就地降级策略。

风险与影响

- 风险：

1. 依赖 torch 私有 API: `abort_distributed_environment()` 使用 `torch.distributed.distributed_c10d._abort_process_group`，属于带下划线的私有接口，PyTorch 版本升级时签名或语义可能变化。已有 `getattr` 守卫和异常告警兜底，但若新版 torch 中该函数变为带参数或为 no-op，将退回 - 影响：影响范围覆盖所有单 tokenizer 进程的 in-process 服务器关闭路径——uvicorn 默认、uvicorn SSL-refresh、granian HTTP/2——以及所有 scheduler 进程的优雅退出路径。受益点有三：一是日志噪音大幅下降，干净关闭不再出现 `CancelledError ERROR traceback` 和 `destroy_process_group() was not called` 告警；二是 GPU 显存在 SIGTERM 后立即归还，对 CI 复用 GPU、避免下一个测试因显存未释放而 OOM 有直接帮助；三是 straggler PID 点名与 `wait_for_gpu_release` 时长上报让慢速关闭变得可观测、可定位。不受影响的路径包括 Engine、gRPC（无服务器，保持直接 `sys.exit(0)`）以及 `tokenizer_worker_num > 1` 的多 worker 路径（子进程 re-import app，不挂 hook）。对团队而言，这是一次低风险、高频收益的生命周期清理，与近期多个 CI teardown 修复（GPU 释放等待、NCCL 端口竞争）形成同一方向的合力。 - 风险标记：依赖 torch 私有 API，核心关闭路径变更，任务取消可能中断 in-flight 处理，缺少自动化回归测试

关联脉络

- PR #39545 [CI] Wait for a killed test server's GPU memory before the next launch: 与本 PR 直接相关: 该 PR 在 `python/sglang/test/test_utils.py` 引入 `wait_for_gpu_release`, 本 PR 在此基础上增加等待时长上报, 并依赖同一机制验证 SIGTERM 后显存立即归零的目标。
- PR #39611 [CI][Disaggregation] Fix EADDRINUSE flake in `test_disaggregation_dwdp_gpt_oss`: 同一 '干净关闭、资源回收' 主题: 修复测试服务器因 NCCL 端口竞争导致的关闭 flake, 与本 PR 的 NCCL 主动中止、进程树兜底清理形成互补。
- PR #39534 [Fix] Aggregate all IPC weight update responses: 同属 `TokenizerManager` 生命周期控制路径: 该 PR 修复 `manager` 的 IPC 响应归并, 本 PR 修复 `manager` 的 SIGTERM 关闭路径, 两者都集中在 `python/sglang/srt/managers/` 下的进程控制逻辑。