PR #36180 完整报告

sgl-project/sglang

[npu] Combine NPU test fixes from #35472 and #34516

合并时间：2026-08-25 09:00

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36180

执行摘要

- 一句话：合并 NPU 测试修复：进程组收割与挂起检测
- 推荐动作：值得 NPU 测试维护者精读 run_bench_serving 的看门狗实现：把 'stdout 空转超时 + 排空宽限 + 进程组 SIGKILL' 组合起来管理长跑子进程，是一个可复用的测试进程治理模式。一般开发者可略读，只需关注 DEEPEP_HCCL_BUFFSIZE 这类环境变量约定是否与本仓库 NPU 部署文档一致。

功能与动机

PR body 明确动机：Improve reliability of Ascend NPU benchmark and accuracy tests by detecting hung benchmark processes earlier and cleaning up orphaned process groups。此前 benchmark 进程一旦挂起会一直占住 stdout，CI 只能等到全局超时；而 deep-ep / HCCL 的 worker 进程在父进程退出后会被 reparent，遗留为孤儿进程，污染同一 runner 上的后续测试。作者把 #35472、#34516 两个修复整合到主线，消除重复实现并统一进程组清理逻辑。

实现拆解

1. 新增 kill_process_group 进程组收割工具（python/sglang/test/ascend/e2e/test_npu_multi_node_utils.py）：基于 start_new_session 进程模型，用 os.killpg(process.pid, signal.SIGKILL) 收割整个进程组，分场景处理 ProcessLookupError、PermissionError 与 OSError，作为 NPU 测试套件的统一清理入口，供 perf 与 accuracy 两侧复用。
2. 重写 run_bench_serving 的看门狗（python/sglang/test/ascend/e2e/test_npu_performance_utils.py）：benchmark 子进程改为 start_new_session=True 启动；新增常量 BENCHMARK_STDOUT_DRAIN_GRACE（10 秒）、BENCHMARK_STDOUT_IDLE_TIMEOUT（300 秒）、BENCHMARK_WATCHDOG_POLL_INTERVAL（30 秒）；守护线程 _kill_on_timeout 覆盖三种僵死场景（直接子进程退出但 stdout 被孙进程持有、stdout EOF 但子进程不退出、长时间无输出疑似挂起），统一走 kill_process_group 强杀；读线程逐行消费 stdout 并刷新 last_activity 标量供 watchdog 判断。
3. accuracy 清理统一（python/sglang/test/ascend/e2e/test_npu_accuracy_utils.py）：删除本地重复的 _kill_evalscope_session 与 signal 导入，在 run_evalscope 的正常、异常与超时路径统一调用 kill_process_group，避免两套收割逻辑漂移。

4. 测试配置与类名修正：qwen3_vl_8b_thinking 与 qwen3_vl_30b_a3b_thinking 两个 MMMU 测试把 HCCL_BUFFSIZE 改名为 DEEPEP_HCCL_BUFFSIZE（对齐 deep-ep 的新环境变量约定），类名从 TestQwen3 改为带模型语义的 TestQwen3_VL_8B_Thinking_MMMU / TestQwen3_VL_30B_A3B_Thinking_MMMU，并把 8B 的 nightly est_time 从 6500 提升到 12000；kimi_k2_6 性能测试增加 --max-total-tokens 32256 参数。
5. CI workflow 调整（.github/workflows/pr-test-npu.yml）：base-c-test-acc 四个 stage 的 runner 标签从 linux-aarch64-a3-- 切换为 linux-aarch64-a3-800t-，与性能 job 使用同一批 800t 节点。

关键文件：

- python/sglang/test/ascend/e2e/test_npu_performance_utils.py（模块 性能看门狗；类别 test；类型 test-coverage；符号 _kill_on_timeout, run_bench_serving, BENCHMARK_STDOUT_IDLE_TIMEOUT, BENCHMARK_STDOUT_DRAIN_GRACE）：核心变更：run_bench_serving 引入进程组级 watchdog，新增三个超时常量并覆盖三种僵死场景，是本 PR 最具技术含量的文件。
- python/sglang/test/ascend/e2e/test_npu_multi_node_utils.py（模块 进程回收；类别 test；类型 test-coverage；符号 kill_process_group）：新增 kill_process_group 公共工具，是 perf 与 accuracy 两侧收割进程组的统一入口。
- python/sglang/test/ascend/e2e/test_npu_accuracy_utils.py（模块 精度测试；类别 test；类型 test-coverage；符号 _kill_evalscope_session, kill_process_group, run_evalscope）：删除本地重复的 _kill_evalscope_session，统一复用 kill_process_group，消除两套收割逻辑漂移。
- test/registered/npu/accuracy/qwen3_vl_8b_thinking/test_npu_qwen3_vl_8b_thinking_1p_mmmu.py（模块 MMMU 精度测试；类别 test；类型 test-coverage；符号 TestQwen3_VL_8B_Thinking_MMMU, TestQwen3）：Qwen3-VL 8B thinking 的 MMMU 注册测试：est_time 提升至 12000、环境变量更名 DEEPEP_HCCL_BUFFSIZE、测试类名改为带模型语义。
- test/registered/npu/accuracy/qwen3_vl_30b_a3b_thinking/test_npu_qwen3_vl_30b_a3b_thinking_1p_mmmu.py（模块 MMMU 精度测试；类别 test；类型 test-coverage；符号 TestQwen3_VL_30B_A3B_Thinking_MMMU, TestQwen3）：Qwen3-VL 30B A3B thinking 的 MMMU 注册测试：环境变量更名与测试类名修正，与 8B 侧保持一致。
- .github/workflows/pr-test-npu.yml（模块 CI 配置；类别 infra；类型 infrastructure）：accuracy 套件四个 stage 的 runner 标签切换为 a3-800t，与性能 job 对齐节点型号，是 CI 配置层面的配套变更。
- test/registered/npu/performance/kimi_k2_6/test_npu_kimi_k2_6_w4a8_8p_in3k5_out1k5_20ms.py（模块 性能基准；类别 test；类型 test-coverage）：Kimi K2.6 性能测试补充 --max-total-tokens 32256，限制总 token 数避免长尾任务挂起。

关键符号：kill_process_group, _kill_on_timeout, run_bench_serving, run_evalscope, _kill_evalscope_session

关键源码片段

python/sglang/test/ascend/e2e/test_npu_performance_utils.py

核心变更：run_bench_serving 引入进程组级 watchdog，新增三个超时常量并覆盖三种僵死场景，是本 PR 最具技术含量的文件。

```
# 用独立 session 启动 benchmark，保证 process.pid == pgid，
# 后续无论谁持有 stdout，都能按进程组整体收割。
process = subprocess.Popen(
    cmd_args,
    stdout=subprocess.PIPE,
    stderr=subprocess.STDOUT,
    text=True,
    bufsize=1,
    env=env,
    start_new_session=True,
)

# reader_done 表示 stdout 已读到 EOF；last_activity 记录最后一条输出时间。
reader_done = threading.Event()
last_activity = time.time()


def _kill_on_timeout():
    while True:
        # 场景 1：直接子进程已退出，但 stdout 仍被子孙进程持有。
        # 等待 DRAIN_GRACE 秒让 reader 排空剩余输出，超时则强杀全部。
        if process.poll() is not None:
            if not reader_done.wait(timeout=BENCHMARK_STDOUT_DRAIN_GRACE):
                logger.error(
                    f"Benchmark stdout still open after process exit, "
                    f"killing process group {process.pid}"
                )
                kill_process_group(process)
                return
        # 场景 2：stdout 已 EOF，但直接子进程迟迟不退出。
        if reader_done.is_set():
            try:
                process.wait(timeout=BENCHMARK_STDOUT_DRAIN_GRACE)
            except subprocess.TimeoutExpired:
                logger.error(
                    f"Benchmark {process.pid} still alive after stdout EOF, "
                    f"killing process group {process.pid}"
                )
                kill_process_group(process)
                return
        # 场景 3：进程还在运行但长时间无输出，判定为挂起。
        idle = time.time() - last_activity
        if idle > BENCHMARK_STDOUT_IDLE_TIMEOUT:
            logger.error(
```

```
                f"Benchmark produced no output for {idle:.0f}s "
                f"(> {BENCHMARK_STDOUT_IDLE_TIMEOUT}s), "
                f"killing process group {process.pid}"
            )
            kill_process_group(process)
            return
        time.sleep(BENCHMARK_WATCHDOG_POLL_INTERVAL)


# daemon 线程保证主测试线程结束时 watchdog 不阻塞退出。
watchdog = threading.Thread(target=_kill_on_timeout, daemon=True)
watchdog.start()

try:
    # 读线程逐行消费 stdout 并刷新 last_activity，供 watchdog 判断空转。
    with open(result_file, "a", encoding="utf-8") as f:
        for line in process.stdout:
            last_activity = time.time()
            if line.strip():
                print(line, end="")
            f.write(line)
            # ... 解析指标行 ...
finally:
    reader_done.set()
    process.wait()
```

python/sglang/test/ascend/e2e/test_npu_multi_node_utils.py

新增 kill_process_group 公共工具，是 perf 与 accuracy 两侧收割进程组的统一入口。

```
def kill_process_group(process):
    """SIGKILL 收割整个进程组。

    process 是用 start_new_session=True 启动的 Popen，其 pid 即进程组 id；
    调用方必须保证这一点，否则 killpg 会误杀无关进程。
    """
    if process is None:
        return
    try:
        os.killpg(process.pid, signal.SIGKILL)
        logger.info(f"killed process group pgid={process.pid}")
    except ProcessLookupError:
        # 进程组已自然退出，属于正常情况。
        logger.info(f"process group pgid={process.pid} already gone")
    except PermissionError:
        # 无权限时只告警，避免掩盖真实问题。
        logger.warning(f"no permission to kill process group pgid={process.pid}")
    except OSError as e:
        # 兜底捕获，保证清理路径不会二次抛出异常。
        logger.warning(f"failed to kill process group pgid={process.pid}: {e}")
```

评论区精华

cherryblo 的 review 集中在 test_npu_performance_utils.py 的代码可读性：两处要求 add comment（新增的 watchdog 常量与 L504 附近逻辑）、建议把 poll() 分支与 reader_done 分支合并为单个 if 语句、质疑 last_activity 为何用 list 包装、并指出某个条件判断不再需要。作者在后续提交（763a7e1c）中做了实质性简化：移除绝对超时上限 BENCHMARK_SERVING_TIMEOUT、删除 _collect_process_tree / _kill_recorded_pids 的孤儿进程快照记录、把 last_activity 从 list 改为标量，与评论方向一致；但 poll 分支与 reader_done 分支是否最终合并没有明确记录。整体是风格与可读性层面的交锋，没有出现正确性或设计原则的分歧。

- watchdog 新增逻辑需要注释说明 (style): 作者保留并补充注释；后续提交直接移除了绝对超时上限 BENCHMARK_SERVING_TIMEOUT，进一步减少需要解释的配置。
- stdout 处理分支应合并为单个 if 语句 (design): 看门狗在提交 2 中被简化（去掉绝对超时上限、last_activity 改标量），但 poll 分支与 reader_done 分支是否最终合并没有明确记录。
- last_activity 为何用 list 包装 (question): 提交 2 将 last_activity 改为标量，移除了 list 包装，简化共享方式。
- 移除不再需要的超时判断条件 (design): 提交 2 移除了 BENCHMARK_SERVING_TIMEOUT 绝对上限，以 idle 超时 + 排空宽限替代，与建议一致。

风险与影响

- 风险：
  1. 误杀风险（test_npu_performance_utils.py）：watchdog 在 300 秒无输出即 SIGKILL 整个进程组。benchmark 在长 prefill、模型权重下载或大数据集准备阶段可能长时间无 stdout，存在误杀正常任务的可能；SIGKILL 不留恢复窗口，排障时只能靠日志。
  2. 进程组收割的边界风险（test_npu_multi_node_utils.py）：kill_process_group 依赖 start_new_session 保证 pid == pgid，若未来有人复用该函数但未用 start_new_session 启动进程，os.killpg 会因进程组不存在或误杀同组进程而静默失败 / 误伤。
  3. DEEPEP_HCCL_BUFFSIZE 改名兼容性：旧变量 HCCL_BUFFSIZE 在底层 deep-ep 版本上是否仍被读取未在 PR 中验证，若新变量名不被 CANN 版本识别，通信 buffer 会回退默认值，可能影响多卡大模型 benchmark 与精度表现。
  4. CI runner 标签切换（pr-test-npu.yml）：linux-aarch64-a3-800t-* 依赖 runner 池容量与标签准确性，若资源不足或标签未注册，accuracy job 会一直排队阻塞合并。
  5. 影响范围受限：全部为测试与 CI 配置变更，不触碰 srt 运行时代码，无推理路径回归风险。 - 影响：影响范围限于 Ascend NPU 的 CI 与注册测试体系：benchmark/accuracy 测试不再因进程挂起或孤儿进程拖死 runner，nightly accuracy 与性能回归的稳定性提升；Qwen3-VL thinking 两个 MMMU 测试的命名与配置更贴近模型语义，便于 CI 结果归因。对使用 NPU CI 的维护者和模型验证工程师有直接收益，对运行时用户无感知。影响程度中等偏低。 - 风险标记：SIGKILL 误杀进程组风险，300 秒无输出判定挂起可能误杀长任务，DEEPEP_HCCL_BUFFSIZE 改名兼容性待验证，CI

runner 标签依赖 a3-800t 资源池，纯测试变更，无运行时影响

关联脉络

- PR #33634 [NPU] Add test for --dllm-fdfo: 同为 NPU 端注册测试，位于同一 test/registered/npu 体系，后续 NPU 测试可复用本 PR 新增的 kill_process_group 看门狗工具。
- PR #36240 [CI] Stop the config ratchets re-parsing the package on every scan: 同属 CI 测试基础设施维护方向，与本 PR 的 NPU CI 稳定性修复互为补充，共同提升 CI 自动化可靠性。