PR #23320 完整报告

sgl-project/sglang

Expose child process PIDs from Engine for health check support

合并时间：2026-04-24 07:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/23320

执行摘要

- 一句话：为 Engine API 暴露子进程 PID 列表
- 推荐动作：值得精读，特别是对于需要直接集成 SGLang Engine 且需要进程健康监控的团队；设计简单，无副作用。

功能与动机

一些调用者直接使用 Engine 作为库（无 HTTP 服务器），并运行自己的健康监控来验证所有 SGLang 子进程是否存活。之前无法从 Engine API 发现子进程 PID，导致监控无法可靠监测子进程状态。

实现拆解

1. 在 SchedulerInitResult 数据类中新增 all_child_pids: List[int] 字段，用于存储所有子进程 PID；
2. 在 Engine 类中新增 get_all_child_pids() 方法返回该列表；
3. 修改 _launch_scheduler_processes 和 _launch_subprocesses 方法，在启动 scheduler 和 detokenizer 进程后收集其 PID，并对于 dp_size>1 场景从 DP 控制器管道消息中获取子调度器 PID；
4. 在 data_parallel_controller.py 中添加 scheduler_pids 常量，并在管道消息中传入子调度器 PID；
5. 新增测试文件 test_engine_child_pids.py 验证返回值是活进程、包含至少 2 个 PID（scheduler+detokenizer）、无重复。

关键文件：

- python/sglang/srt/entrypoints/engine.py（模块 Engine 入口；类别 source；类型 core-logic；符号 get_all_child_pids, SchedulerInitResult）：核心修改：添加 all_child_pids 字段、get_all_child_pids 方法、收集子进程 PID 的逻辑。
- python/sglang/srt/managers/data_parallel_controller.py（模块 DP 控制器；类别 source；类型 entrypoint；符号 SCHEDULER_PIDS_ARG）：新增 SCHEDULER_PIDS_ARG 常量和在管道消息中传递子调度器 PID，支持 dp_size>1 场景。
- test/registered/core/test_engine_child_pids.py（模块 进程 PID 测试；类别 test；类型 test-coverage；符号 TestEngineChildPids）：新增测试文件，验证新 API 的正确性（PID

存活、至少包含 scheduler 和 detokenizer、无重复）。

关键符号：get_all_child_pids, SchedulerInitResult.all_child_pids, run_data_parallel_controller_process

关键源码片段

python/sglang/srt/entrypoints/engine.py

核心修改：添加 all_child_pids 字段、get_all_child_pids 方法、收集子进程 PID 的逻辑。

```
# 在 SchedulerInitResult 数据类中新增 all_child_pids 字段
@dataclasses.dataclass
class SchedulerInitResult:
    scheduler_infos: List[Dict[str, Any]]
    all_child_pids: List[int] = dataclasses.field(default_factory=list) # 新增：存储所有子进程 PID
    wait_for_ready: Callable[[], None] = lambda: None
    wait_for_completion: Callable[[], None] = lambda: None
    engine_info_bootstrap_server: Optional[Any] = None

# Engine 类中新增公共方法
def get_all_child_pids(self) -> List[int]:
    """Returns a list of all child process PIDs."""
    return self._scheduler_init_result.all_child_pids

# 在 _launch_scheduler_processes 中收集 scheduler 和 DP controller 的子进程 PID
def _launch_scheduler_processes(self, ...):
    # ... 启动 scheduler 进程后
    all_child_pids = [proc.pid for proc in scheduler_procs]
    # 在 wait_for_ready 中，如果 dp_size > 1，从 DP controller 管道消息中提取子调度器 PID
    def wait_for_ready():
        infos = _wait_for_scheduler_ready(...)
        if server_args.dp_size > 1:
            for info in infos:
                if SCHEDULER_PIDS_ARG in info:
                    all_child_pids.extend(info[SCHEDULER_PIDS_ARG])
    # ...

# 在 _launch_subprocesses 中收集 detokenizer PID
detoken_proc.start()
scheduler_init_result.all_child_pids.append(detoken_proc.pid)
```

python/sglang/srt/managers/data_parallel_controller.py

新增 SCHEDULER_PIDS_ARG 常量和在管道消息中传递子调度器 PID，支持 dp_size>1 场景。

```
# 新增常量
SCHEDULER_PIDS_ARG = "scheduler_pids"

# 在 run_data_parallel_controller_process 中收集 scheduler 的 PID 并通过管道发送
def run_data_parallel_controller_process(...):
```

```
    controller = DataParallelController(...)
    scheduler_pids = [
        proc.pid for proc in controller.scheduler_procs if proc is not None
    ]
    pipe_writer.send({
        "status": "ready",
        "max_total_num_tokens": controller.max_total_num_tokens,
        "max_req_input_len": controller.max_req_input_len,
        SCHEDULER_PIDS_ARG: scheduler_pids, # 新增：传递子调度器 PID
    })
```

test/registered/core/test_engine_child_pids.py

新增测试文件，验证新 API 的正确性（PID 存活、至少包含 scheduler 和 detokenizer、无重复）。

```
class TestEngineChildPids(CustomTestCase):
    def test_get_all_child_pids_returns_live_pids(self):
        engine = sgl.Engine(model_path=DEFAULT_SMALL_MODEL_NAME_FOR_TEST, random_
        seed=42)
        try:
            pids = engine.get_all_child_pids()
            # 验证返回类型和数量
            self.assertIsInstance(pids, list)
            self.assertGreater(len(pids), 0)
            # 验证每个 PID 都是当前进程的后代
            current_proc = psutil.Process(os.getpid())
            child_pids = {c.pid for c in current_proc.children(recursive=True)}
            for pid in pids:
                self.assertIn(pid, child_pids, f"PID {pid} is not a child")
        finally:
            engine.shutdown()

    def test_child_pids_include_scheduler_and_detokenizer(self):
        engine = sgl.Engine(...)
        try:
            pids = engine.get_all_child_pids()
            # dp_size=1 时至少应有 scheduler 和 detokenizer 两个 PID
            self.assertGreaterEqual(len(pids), 2)
        finally:
            engine.shutdown()

    def test_child_pids_no_duplicates(self):
        engine = sgl.Engine(...)
        try:
            pids = engine.get_all_child_pids()
            self.assertEqual(len(pids), len(set(pids)))
        finally:
            engine.shutdown()
```

评论区精华

审核者 kpham-sgl 对 *engine.py* 中使用常量 SCHEDULER_PIDS_ARG 提出了细微意见（可直接使用字符串），但结论是使用常量也无妨，PR 已合并。

- 是否应使用字符串字面量而非常量 (style): 保留常量定义，但 reviewer 认可。

风险与影响

- 风险：风险较低。需确保在 dp_size>1 场景下，DP 控制器管道消息中的 scheduler_pids 在 wait_for_ready 时已可用；当前实现是在控制器内部收集，时序可靠。另外，如果子进程在调用 get_all_child_pids 之前已结束，返回的 PID 可能无效，但测试已覆盖实时性验证。
- 影响：仅影响通过 Engine 类直接使用 SGLang 的调用者，为其提供了健康的子进程监控能力；对于通过 HTTP 服务器使用的方式无影响。影响范围较小，正向增益。
- 风险标记：子进程可能提前退出导致 PID 无效

关联脉络

- 暂无明显关联 PR