PR #36235 完整报告

sgl-project/sglang

[CI] Stop the resolution ratchets re-parsing the whole package

合并时间：2026-08-25 07:55

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36235

执行摘要

- 一句话：预过滤并延迟解析，让棘轮测试提速 2.4 倍
- 推荐动作：值得精读，尽管它只是一个测试优化 PR，但体现了很好的性能分析方法论：精准定位热点（量化耗时分布）、通过最小改动消除 O(n²) 开销、用自校验确保正确性。对于 CI 维护者，可作为后续优化其他慢测试的参考模板。对于普通开发者，可以学习 get_source_segment 的性能陷阱和预过滤模式的通用价值。

功能与动机

CI shard 7 因 test_resolution_is_reproducible.py 耗时 359s（占该 shard 41%）而超时 15 分钟，成为瓶颈。PR body 明确指出三个扫描测试 test_every_fork_of_a_record_has_a_resolved_one、test_only_the_gate_runs_the_pipeline 和 test_the_gate_is_reached_from_the_launcher_and_from_publish 做了大量丢弃的工作，其中 ast.get_source_segment 被循环调用 28,817 次，每次都会重新切分整个文件，导致 O(n²) 开销。

实现拆解

该 PR 仅修改一个测试文件 test/registered/unit/server_args/test_resolution_is_reproducible.py，通过三处优化提升三个扫描测试的性能：

1. test_every_fork_of_a_record_has_a_resolved_one 预过滤 + 滞后解析（+11/-3） - 在 ast.parse(source) 前增加 if "Process" not in source: continue，因为该测试只关注包含 Process 字面量的文件，预过滤可跳过无关文件，避免不必要的 AST 解析。 - 将 body = ast.get_source_segment(source, node) 从循环开头移到 if not forks: continue 之后，因为 body 只在存在 fork 时才被使用，此举避免了对 28,817 个函数定义都调用昂贵的 get_source_segment（该函数每次都会重新切分整个文件）。
2. test_only_the_gate_runs_the_pipeline 预过滤（+3/-1） - 在 ast.parse(path.read_text()) 前读取 source 并检查 if "_run_resolution_pipeline" not in source: continue，与 1 相同的逻辑，只解析包含目标符号的文件。
3. test_the_gate_is_reached_from_the_launcher_and_from_publish 预过滤（+3/-1） - 在 ast.parse 前检查 if "resolve_once" not in source: continue。

三处预过滤的字符串匹配条件与 AST 匹配逻辑完全一致，因此不会漏报任何应被检查的文件。PR body 也明确说明，两个 test_only_the_gate_runs_the_pipeline 和

test_the_gate_is_reached_from_the_launcher_and_from_publish 测试本身会断言精确的调用方列表，因此预过滤的效果等同于自校验。

此外，PR body 还揭示了两个遗留问题，但并未在本 PR 中解决：_RestoresProcessState 覆写 _callTestMethod 导致部分测试缺少 CI 标记和重试机制；test_no_bare_replace_of_a_record_outside_the_helper 和 test_every_program_that_builds_a_record_resolves_it 因无法找到足够有选择性的预过滤键而放弃优化。

关键文件：

- test/registered/unit/server_args/test_resolution_is_reproducible.py（模块 测试；类别 test；类型 test-coverage；符号 test_every_fork_of_a_record_has_a_resolved_one, test_only_the_gate_runs_the_pipeline, test_the_gate_is_reached_from_the_launcher_and_from_publish）：该文件是唯一改动文件，包含三个性能优化点，是本次变更的核心。

关键符号：test_every_fork_of_a_record_has_a_resolved_one, test_only_the_gate_runs_the_pipeline, test_the_gate_is_reached_from_the_launcher_and_from_publish

关键源码片段

test/registered/unit/server_args/test_resolution_is_reproducible.py

该文件是唯一改动文件，包含三个性能优化点，是本次变更的核心。

```
# test/registered/unit/server_args/test_resolution_is_reproducible.py
# 此测试遍历整个 sglang 包，查找包含 Process fork 调用的函数，并断言它们
# 在 fork 前已解析记录。优化前，它会对每个函数调用 ast.get_source_segment，
# 该函数每次都会重新切分文件，导致 O(n²) 开销。本 PR 移除了对无关文件的
# AST 解析，并延迟 body 的获取。

def test_every_fork_of_a_record_has_a_resolved_one(self):
    import ast

    package_root = pathlib.Path(next(iter(sglang.__path__))).resolve()
    offenders, examined = [], 0
    for path in sorted(package_root.rglob("*.py")):
        rel = path.relative_to(package_root).as_posix()
        if rel.startswith("test/") or "/test/" in rel:
            continue
        # The diffusion runtime has its own record with no gate.
        if rel.startswith("multimodal_gen/"):
            continue
        try:
            source = path.read_text(encoding="utf-8-sig")
            # 添加预过滤：如果文件不包含 'Process'，则无需解析 AST，因为
            # 后续匹配依赖该名称。
            if "Process" not in source:
                continue
```

```
            tree = ast.parse(source)
        except (SyntaxError, UnicodeDecodeError):
            continue
        for node in ast.walk(tree):
            if not isinstance(node, (ast.FunctionDef, ast.AsyncFunctionDef)):
                continue
            forks = [
                call
                for call in ast.walk(node)
                if isinstance(call, ast.Call)
                and (
                    (
                        isinstance(call.func, ast.Attribute)
                        and call.func.attr == "Process"
                    )
                    or (
                        isinstance(call.func, ast.Name)
                        and call.func.id == "Process"
                    )
                )
                # get_source_segment 在这里仍然被调用，但只有在 forks 非空时才
                # 调用，且次数受限。
                and "server_args" in (ast.get_source_segment(source, call) or "")
            ]
            if not forks:
                continue
            # 将 body 获取移到 guard 之后，避免对不包含 fork 的函数
            # 重复切分整个文件。
            body = ast.get_source_segment(source, node) or ""
            examined += 1
            # `spawn` starts a fresh interpreter, so the child may probe.
            if 'get_context("spawn")' in body or "'spawn'" in body:
                continue
            if "resolve_once(" in body or "publish(" in body:
                continue
            if rel in self._AFTER_LAUNCHER_RESOLVE:
                continue
            offenders.append(f"{rel}:{forks[0].lineno} {node.name}")
    self.assertGreater(
        examined, 5, f"only {examined} fork sites found; the scan broke"
    )
    self.assertEqual(
        offenders,
        [],
        "these fork a child that will resolve the record, without resolving "
        "it first -- the child cannot initialize CUDA if this process "
        f"already has:\n  " + "\n  ".join(offenders),
    )
```

评论区精华

该 PR 的 review 讨论较少，仅有 CI 机器人和作者的交互：

1. 作者发起 rerun: alexnails 请求 /rerun-test test_resolution_is_reproducible.py。
2. CI 机器人拒绝 rerun: github-actions[bot] 指出 PR 已与 main 分叉，需要 rebase 并重新推送，未执行 rerun。这反映了该 PR 在 main 更新后 CI 未及时验证，可能影响后续合并。

没有其他实质性设计讨论。作者在 PR body 中详细记录了性能分析和方法论，但未在评论区展开。

- CI rerun 请求被拒绝 (other): 未执行 rerun，需 rebase 和重新推送。

风险与影响

- 风险：风险极低，因为改动仅涉及测试文件内部实现，不改变任何断言逻辑。预过滤条件与 AST 匹配条件完全一致，理论不会漏报。作者还强调了两个测试 test_only_the_gate_runs_the_pipeline 和 test_the_gate_is_reached_from_the_launcher_and_from_publish 本身是精确断言，可自校验预过滤的正确性。唯一潜在风险是预过滤字符串与 AST 匹配未来不同步（例如引入 _run_resolution_pipeline 的新变体），但此类变化会同时失败于现有断言。此外，PR body 指出两个未能预过滤的测试，但明确不处理，不会引入风险。
- 影响：影响范围限于 CI 性能：test_resolution_is_reproducible.py 在 CI shard 7 上的耗时从 359s 降至约 149s（本地归一化），缓解 shard 超时问题。对用户无直接影响，对开发者的主要受益是 CI 稳定性提升和等待时间缩短。改动可维护性良好，仅涉及测试内部优化，不影响 SGLang 运行时代码。
- 风险标记：仅测试文件变更，预过滤条件与 AST 匹配一致，CI 未覆盖（rerun 被拒）

关联脉络

- PR #36240 [CI] Stop the config ratchets re-parsing the package on every scan: 同一作者同一系列：优化 8 个配置棘轮扫描测试，与本 PR 目标一致，都是减少 CI 测试中重复解析包的开销。
- PR #36241 [CI] Cut repeated tokenizer loads, serial subprocesses and a double scan: 同为 CI 性能优化系列，削减 CPU 测试的重复加载与串行子进程开销。