PR #34094 完整报告

sgl-project/sglang

config: pin that resolution is reproducible from the raw input

合并时间：2026-08-10 05:44

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/34094

执行摘要

- 一句话：新增解析可重现性测试，钉死 ServerArgs 纯函数契约
- 推荐动作：值得精读。虽然只有一个测试文件，但它演示了如何为 " 纯函数化 " 契约设计回归测试，并完整展示了审查者与作者的往返式修复过程。重点看 _comparable 的快照语义、_process_state/_restore_process_state 的进程状态恢复、_STICKY_ACROSS_RESOLUTIONS 对已知粘性行为的显式命名，以及 _callTestMethod 禁用重试的理由。

功能与动机

PR body 说明：配置分层的最终状态让 ServerArgs 保持用户原始输入，每个发布进程自己派生解析值——bags 不跨进程边界，子进程只能从交给它的记录重新投影。这只有在同一原始输入解析两次得到相同答案时才成立。测试从三方面固定：

1) 相等原始输入两次解析逐字段一致（嵌套 dataclass 结构化比较）； 2) 显式 backend 的解析不改变下一个解析的默认（声明注册表进程全局）； 3) 解析兄弟配置不触碰第一个记录。失败意味着某解析步骤读了它同时写入的状态，这恰是会让 launcher 与其 schedulers 静默分歧的机制。

实现拆解

1. 变更入口与 CI 注册：唯一新增文件 test/registered/unit/server_args/test_resolution_is_reproducible.py。文件顶部通过 register_cpu_ci（base-a-test-cpu）、register_cuda_ci（base-b、1-gpu-small）、register_amd_ci（stage-b-test-1-gpu-small-amd）注册到三套 CI。原因是 is_cuda()、is_hip()、设备能力等分支只能由真实硬件到达，mock 谓词会固定 mock 而非解析本身，故选择多平台真实注册。
2. 形状矩阵设计：_SHAPES 覆盖 plain（Llama mini JSON）、speculative（EAGLE 相关参数）、multimodal（Qwen2-VL mini + vision_config）、plain_cpu_device（device="cpu"，专门到达 _handle_cpu_backends）；仅当 torch.cuda.is_available() 时追加 deepseek_dsa 形状——DeepSeek-V3 mini 配置带 index_topk 进入 DSA 路径，该路径在解析时探测 torch.cuda.get_device_capability()，无驱动主机上会抛错，因此必须硬件门控。所有形状均为内联 JSON，不加载权重。
3. 进程状态快照与恢复：_process_state 沿 type(envs) 的 MRO 收集所有 EnvField，返回 os.environ 拷贝加每个字段的 _set_to_none 标志；_restore_process_state 恢复两者。

setUp 保存 pristine 状态并 addCleanup 恢复；每个 subtest 循环体开头再显式恢复 pristine，避免前一个形状的解析（如 multimodal 写入 SGLANG_USE_CUDA_IPC_TRANSPORT）污染后续断言。_callTestMethod 直接调用 unittest.TestCase 版本，禁用 CustomTestCase 的 CI 重试——否则重试会从第一次泄漏的状态启动，把回归变成通过。

4. 比较与排除规则：_comparable 逐 dataclass 字段取值；嵌套 dataclass（如 cuda_graph_config）走 asdict 结构化比较，其余可变值（list/dict）一律 copy.deepcopy，防止快照持有活对象导致共享可变字段被原地修改时断言假通过；排除 _NOT_COMPARABLE（random_seed，文档性保留）与 _STICKY_ACROSS_RESOLUTIONS（mm_feature_transport，main 上已有的粘性字段，显式命名以便新粘性字段暴露）。
5. 四个测试用例：同输入双解析逐字段一致（矩阵驱动）；显式 backend（torch_native，CPU/CUDA 上均非默认）解析前取 control、解析后重新默认解析必须与 control 完全一致，并追加 multimodal、torch_compile、deepseek_dsa 为中间形状的二级探针；兄弟解析隔离（快照先于第二次解析，且断言两个结果相等）；声明 provenance 可复现（第二次解析前深拷贝 _resolved_overrides，解析后还断言 first._resolved_overrides == 快照）。

关键文件：

- test/registered/unit/server_args/test_resolution_is_reproducible.py（模块 配置解析；类别 test；类型 test-coverage；符号 TestResolutionIsReproducible, _config_dir, _process_state, _restore_process_state）：本次唯一变更文件，从三个角度钉死 ServerArgs 解析在相同原始输入下可重现：逐字段一致、显式后端不改变后续默认、兄弟解析互不影响。测试自身包含防假绿设计（禁用 CI 重试、深拷贝快照、subtest 前恢复 pristine 状态、探针必须非默认且断言发散），并注册到 CPU/CUDA/AMD 三套 CI，是配置袋重构栈第 12 步的验证层。

关键符号：_process_state, _restore_process_state, _callTestMethod, _resolved, _comparable, test_two_resolutions_of_the_same_input_agree, test_a_resolution_does_not_leak_into_the_next, test_resolving_a_sibling_leaves_the_first_alone, test_the_declaration_provenance_is_reproducible

关键源码片段

test/registered/unit/server_args/test_resolution_is_reproducible.py

本次唯一变更文件，从三个角度钉死 ServerArgs 解析在相同原始输入下可重现：逐字段一致、显式后端不改变后续默认、兄弟解析互不影响。测试自身包含防假绿设计（禁用 CI 重试、深拷贝快照、subtest 前恢复 pristine 状态、探针必须非默认且断言发散），并注册到 CPU/CUDA/AMD 三套 CI，是配置袋重构栈第 12 步的验证层。

```
"""Resolution is a pure function of the raw input plus this node's environment.

配置分层重构把 ServerArgs 保持为用户原始输入，每个进程自己派生解析值；
bag 不跨进程边界，子进程只能从交给它的记录重新投影。
所以「相同原始输入两次解析结果一致」是派生契约的地基。
```

```
"""

# 已知的跨解析粘性字段：multimodal 解析会写 SGLANG_USE_CUDA_IPC_TRANSPORT，
# 下一次解析（即使纯文本）会读取 is_set() 并继承——这是 main 的现有行为，
# 测试显式命名它，新的粘性字段将无法隐藏在排除名单里。
_STICKY_ACROSS_RESOLUTIONS = frozenset({"mm_feature_transport"})


def _process_state(self):
    """一次解析可能留下的进程状态：os.environ 以及 EnvField 的 _set_to_none。

    _set_to_none 是描述符级标志，os.environ 不携带；沿 MRO 遍历，
    否则漏掉定义在基类上的字段。
    """
    fields = {}
    for klass in reversed(type(envs).__mro__):
        for name, field in vars(klass).items():
            if isinstance(field, EnvField):
                fields[name] = field
    return (dict(os.environ), {n: f._set_to_none for n, f in fields.items()})


def _restore_process_state(self, state):
    # 恢复到快照状态，避免上一次解析的泄漏影响后续子测试
    saved_environ, saved_none_flags = state
    os.environ.clear()
    os.environ.update(saved_environ)
    for name, was_none in saved_none_flags.items():
        getattr(type(envs), name)._set_to_none = was_none


def _comparable(self, server_args: ServerArgs) -> dict:
    """逐字段快照：嵌套 dataclass 走 asdict，其余可变值一律 deepcopy。

    如果存的是活对象，兄弟解析原地改 list/dict 时快照跟着变，
    隔离断言就会在真正回归上通过。
    """
    out = {}
    for field in dataclasses.fields(server_args):
        if field.name in _NOT_COMPARABLE:
            continue
        value = getattr(server_args, field.name)
        out[field.name] = (
            dataclasses.asdict(value)
            if dataclasses.is_dataclass(value)
            else copy.deepcopy(value)
        )
    return out
```

```
def test_two_resolutions_of_the_same_input_agree(self):
    for label, config, kwargs in _SHAPES:
        with self.subTest(shape=label):
            # 每个形状从测试方法起始状态开始，而不是上一个形状留下的状态
            self._restore_process_state(self._pristine_state)
            model_path = self._config_dir(config)
            first = self._resolved(model_path, **kwargs)
            second = self._resolved(model_path, **kwargs)
            self.assertEqual(self._comparable(first), self._comparable(second))
```

评论区精华

最有价值的交锋集中在 " 测试如何避免把回归变成通过 ":

- 关于 CI 重试（codex P2）：作者承认 "addCleanup runs after the last attempt, so the retry would start from exactly the state the failed attempt leaked"，因此该文件 _callTestMethod 绕过 CustomTestCase 重试，恢复也覆盖 EnvField._set_to_none。
- 关于 CPU 套件假绿（ch-wan）："On base-a-test-cpu, Llama-mini auto-default is already triton, so explicit vs default is a full no-op"，改用 torch_native 并断言显式解析确实不等于默认解析（assertNotEqual）。
- 关于发现的真实粘性字段：修复 subtest 隔离后作者发现 "resolving a multimodal config first sets the env to 1, and the next resolution of a text-only model comes out with mm_feature_transport=cuda_ipc... it is existing behaviour"，选择用 _STICKY_ACROSS_RESOLUTIONS 显式命名而非隐藏，让未来的新粘性字段直接失败。
- 尾部 P1 提醒（codex）："This unconditional cuda_ipc assertion makes test_a_resolution_does_not_leak_into_the_next fail on both non-NVIDIA registrations"——材料中未见到作者对最终门控实现的确认回复。
- CI 重试会把状态泄漏变成通过 (testing): 作者在 _callTestMethod 中直接调用 unittest.TestCase._callTestMethod 绕过重试，并让恢复逻辑同时覆盖 EnvField._set_to_none 与 os.environ。
- CPU 套件上探针是默认值导致假绿 (testing): 探测后端改为 torch_native（CPU/CUDA 均非默认），并先断言显式解析 != 默认解析（assertNotEqual）再继续比较。
- _comparable 快照持有活对象 (testing): 非 dataclass 值一律 copy.deepcopy，dataclass 走 asdict，快照与活对象脱钩。
- subtest 之间进程状态不隔离 (testing): 每个 shape 与 intermediate 循环体开头调用 _restore_process_state，恢复到测试方法起始的 pristine 状态。
- mm_feature_transport 的真实跨解析粘性 (correctness): 不隐藏该行为，显式命名 _STICKY_ACROSS_RESOLUTIONS={"mm_feature_transport"} 排除出字段比较，并写明机制；未来的新粘性字段将直接失败而非混入。
- 硬件门控与三套 CI 注册 (testing): 注册 register_cpu_ci、register_cuda_ci、register_amd_ci 三套，DeepSeek-DSA 形状在 GPU 注册上执行，CPU 无设备时可跳过 DSA 形状避免崩溃。

- DeepSeek mini 缺 index_topk 不进入 DSA (testing): mini JSON 加入 index_topk/index_head_dim/index_n_heads，形状更名为 deepseek_dsa。
- 尾部 P1/P2：CUDA-only 断言门控、legacy override 清除、provenance 覆盖全形状 (correctness): 材料中未见作者对这些尾部评论的明确回复；合并版本 head 摘录中已能见到 plain_cpu_device 形状与 _STICKY_ACROSS_RESOLUTIONS 机制，但具体门控与 env 清除是否最终落地需以合并后代码为准。

风险与影响

- 风险：本 PR 无生产代码变更，风险集中在测试本身与 CI 稳定性：
  - 尾部数个 P1/P2 线程（CUDA-only 传输断言门控、进程以 SGLANG_USE_CUDA_IPC_TRANSPORT=0 启动时 auto-selection 不触发、provenance 检查仅覆盖默认 Llama 形状）在材料中未见作者明确回复，合并版本是否全部落实存在不确定性；若未落实，CPU/AMD 套件或特定环境变量下的 CUDA runner 可能出现预期外失败。
  - 测试依赖对进程状态的完整快照与恢复；若未来 EnvField 增加不在类属性上声明的状态，_process_state 会漏掉，恢复不完整可能使后续用例互相污染。
  - 测试钉死的是 fork 场景（子进程继承父进程环境与模块级缓存）；spawn 子进程冷模块缓存（runtime_context 中的 functools memos 等）下的分歧不在覆盖范围，docstring 已声明该边界。
  - CI 时长增加：每个注册约 10s 预估时间，三平台合计约 30s。
  - 影响：对用户无直接运行时影响（纯测试变更）。对系统：该测试保护配置袋重构栈的 " 子进程自行派生解析值 " 契约，任何解析过程中读写进程状态的回归都会被拦截，避免 launcher 与 scheduler 静默分歧；同时把 mm_feature_transport 的跨解析粘性行为记录为已知契约，为未来消除该行为提供断言基点。对团队：43 条 review 评论的往返打磨沉淀出一套 " 状态污染类测试防假绿 " 的方法论（禁用重试、深拷贝快照、subtest 隔离、探针必须非默认且断言确实发散），值得其他测试复用。
  - 风险标记：纯测试变更，尾部 P1/P2 线程未确认关闭，spawn 进程冷缓存场景超覆盖范围，依赖进程级状态恢复的测试

关联脉络

- PR #34095 config: the runner and scheduler read resolved config from the bags: 同一配置袋重构栈：runner 与 scheduler 改为从配置袋读取解析后的配置，本 PR 守护的 " 子进程自行派生解析值 " 契约正是该变更正确性的前提。
- PR #34096 config: the KV-cache configurator reads the bags: 同一配置袋重构栈：KV 缓存配置器改读配置袋，同样依赖解析可重现性；本测试的失败即意味着这类派生在 launcher 与子进程间会分歧。
- PR #34097 docs(skill): record where config is read now that the seed is off limits: 配置栈的文档配套，记录配置读取入口与种子限制，与本 PR 固定的解析可重现契约属于同一工作线。