PR #36618 完整报告

sgl-project/sglang

config: resolution declares, and nothing writes a field

合并时间：2026-08-28 03:53

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36618

执行摘要

- 一句话：重构配置解析：resolution 只声明不写字段
- 推荐动作：值得精读。重点看三点：run_post_process_pass 的 published 拒绝设计（含与 declare_late_resolution 的对称性）、契约测试的双方向验证 + 注入法（先证明两个方向独立失败再断言）、test_chain_read_ratchet.py 用 AST 强制调用形状以保证扫描完备性的思路。注意 2 条未解决的 P2 review 评论，尤其是顺序启动回归，在后续 PR 中确认是否修复。

功能与动机

PR body 明确说明：此前的系列改造已让 ServerArgs 持有 operator 输入、解析答案进入 declaration stash，但仍有两条通道会在之后写字段，且没有任何断言约束结果（"Two channels could still write a field afterwards, and nothing asserted the result"）。本 PR 关闭这两条通道并补上断言；同时修复 _a2a_fusion_adjustments 与 _hrm_text_attention_force 未被 @register_post_process 注册、导致所有枚举 POST_PROCESS_PASSES 的检查绕过它们的 registry 漏洞。

实现拆解

实现分四步：

1. 删除写穿逻辑（核心）：python/sglang/srt/arg_groups/overrides.py 中 run_post_process_pass 不再在 _resolution_finished 时调用 _apply_fields 写回字段。解析后的 pass 声明统一进入 _resolved_overrides stash，publish 从 stash 投影 config bags；_apply_fields 保留给 RuntimeContext.override_server_args（测试 launch 替身）这一个调用方。同时新增对 published record 的拒绝：若 get_context().server_args 就是当前对象，直接 raise ValueError，因为 publish 之后 stash 不会再被投影，追加声明是静默 no-op，post-publish 变更应走 get_context().override(...)。
2. 修正 registry 语义：POST_PROCESS_PASSES 注释从“end-state execution order”改为“registry, not an execution order”——实测 _hisparse_validation 注册第 16 位但总是最后执行，因为 check_server_args 阶段晚于 __post_init__。PR body 说明 _a2a_fusion_adjustments 与 _hrm_text_attention_force 已补上 @register_post_process（diff 摘要窗口未完整展示该 hunk）。

3. 新增契约测试：test/registered/unit/server_args/test_record_holds_the_raw_input.py（新增）用 15 种 launch shape 断言 resolve_once() 后每个字段仍与 _raw_input 快照一致，覆盖两个独立方向——字段不得被重新绑定（rebind）、调用方传入的可变对象不得被原地修改（in-place edit，快照引用同一对象因此逐字段比较不可见），并通过 setattr / cuda_graph_config.setdefault 注入验证两个方向独立失败。
4. 注册覆盖扫描与配套测试 / 文档：test/registered/unit/test_chain_read_ratchet.py 新增 _passes_named_at_call_sites() AST 扫描（强制 run_post_process_pass 第二参数为裸名字，否则硬失败）与 TestEveryInvokedPassIsRegistered，以调用点为 ground truth 双向校验 registry；test_model_overrides.py 将 test_post_materialize_pass_writes_through 反转为 test_a_pass_after_resolution_declares_without_writing，断言 pass 后字段保持原始值；.claude/skills/sglang-runtime-context/SKILL.md 修正 publish 描述为"从声明投影，而非快照 resolved values"。

配套验证：24 种 launch shape × 478 个共享字段的 resolution dump 与基线 f775db03aaa 逐字段比较 0 差异（唯一新增字段 grpc_worker_threads 两侧值均为 4）；所有 guard 在系列每个 commit 上单独运行通过。

关键文件：

- python/sglang/srt/arg_groups/overrides.py（模块 配置解析；类别 source；类型 core-logic；符号 run_post_process_pass, _apply_fields, POST_PROCESS_PASSES, register_post_process）：唯一生产代码变更：删除 run_post_process_pass 的字段写穿、新增 published record 拒绝、修正 POST_PROCESS_PASSES 的 registry 语义，是系列契约的承重墙。
- test/registered/unit/server_args/test_record_holds_the_raw_input.py（模块 契约测试；类别 test；类型 test-coverage；符号 TestRecordHoldsTheRawInput, setUp, restore, _model_path）：新增的核心契约测试：15 种 launch shape 断言 resolve_once() 后每个字段仍等于调用方传入的 _raw_input 快照，双方向验证（rebind 与 in-place edit）。
- test/registered/unit/test_chain_read_ratchet.py（模块 链读约束；类别 test；类型 test-coverage；符号 _passes_named_at_call_sites, TestEveryInvokedPassIsRegistered, test_the_registry_covers_every_call_site）：新增 registry 覆盖扫描：以 run_post_process_pass 调用点为 ground truth，AST 强制调用形状并双向校验 POST_PROCESS_PASSES，堵住未注册 pass 绕过检查的漏洞。
- test/registered/unit/test_model_overrides.py（模块 模型覆盖；类别 test；类型 test-coverage；符号 test_post_materialize_pass_writes_through, test_a_pass_after_resolution_declares_without_writing）：将写穿断言反转为声明不写字段断言，验证 resolution 后 pass 只进 stash 不碰字段，是核心行为变更的直接测试佐证。
- .claude/skills/sglang-runtime-context/SKILL.md（模块 技能文档；类别 docs；类型 documentation）：修正 publish 语义描述：从声明投影而非快照 resolved values，避免开发者按旧描述在 record 上找解析结果。

关键符号：run_post_process_pass, _apply_fields, _passes_named_at_call_sites, TestEveryInvokedPassIsRegistered.test_the_registry_covers_every_call_site, TestRecordHoldsTheRawInput.test_no_field_moves_from_what_the_caller_passed, TestRecordHoldsTheRawInput.test_the_snapshot_is_the_value_the_caller_passed,

test_a_pass_after_resolution_declares_without_writing

关键源码片段

python/sglang/srt/arg_groups/overrides.py

唯一生产代码变更：删除 run_post_process_pass 的字段写穿、新增 published record 拒绝、修正 POST_PROCESS_PASSES 的 registry 语义，是系列契约的承重墙。

```
  def run_post_process_pass(server_args: Any, fn: Callable[..., dict]) -> None:
      """在遗留 handler 槽位上调用一个 post-process pass。

      pass 在解析态视图（叠加了 stash 中已累积声明）上求值，并把它的声明
      追加进 stash —— config bags 正是从 stash 投影出来的。字段本身保持不动。
      """
      from sglang.srt.runtime_context import get_context

      # 拒绝已发布（published）的 record：publish 之后 stash 不会再被投影，
      # 此时追加声明会成为静默 no-op，因此直接报错，引导代码改走
      # get_context().override(...) 更新 bags。
      try:
          published = get_context().server_args
      except ValueError:
          published = None
      if published is server_args:
          raise ValueError(
              f"run_post_process_pass({fn.__qualname__!r}) called on the published "
              "config; the stash is projected at publish and never again, so a "
              "declaration made here would be a silent no-op -- post-publish "
              "changes go to the bags via get_context().override(...)"
          )

      # 在叠加了声明 overlay 的只读视图上求值，pass 只返回声明 dict，不许改字段。
      declared = fn(ResolvedView(server_args, overlay=_declaration_overlay(server_args)))
      if not isinstance(declared, dict):
          raise TypeError(
              f"post-process pass {fn.__qualname__} must return a dict, "
              f"got {type(declared).__name__}"
          )
      if declared:
          entry = (fn.__qualname__, dict(declared))
          stash = getattr(server_args, "_resolved_overrides", None)
          if stash is None:
              # 直接作用于 fixture 的 pass 槽位可能从未经过 monolith dispatch
              # （dispatch 负责初始化 stash），这里惰性创建；真实 publish 一定
              # 先过 dispatch，所以 pass 槽位必须位于 __post_init__ 中
              # dispatch 之后。
              stash = server_args._resolved_overrides = []
          stash.append(entry)
          validate_declarations(server_args, [entry])
```

test/registered/unit/server_args/test_record_holds_the_raw_input.py

新增的核心契约测试：15 种 launch shape 断言 resolve_once() 后每个字段仍等于调用方传入的 _raw_input 快照，双方向验证（rebind 与 in-place edit）。

```
def _moved(current, original):
    """判断字段是否不再等于调用方传入的原始值。

    对可变对象（list / dict / set / bytearray）要求同一对象（is 判定）：
    等值拷贝不再与调用方共享内存，record 的原地修改会污染调用方数据，
    因此也算 moved；对 int / str 等不可变类型退化为值比较
    （相等的 int / str 不一定是同一对象）。
    """
    if current is original:
        return False
    if isinstance(original, (list, dict, set, bytearray)) or isinstance(
        current, (list, dict, set, bytearray)
    ):
        # 可变对象只有“同一个对象”才算未移动：等值拷贝已不再与调用方共享。
        return True
    # 不可变类型的相等即视为未移动。
    return current != original


def test_no_field_moves_from_what_the_caller_passed(self):
    # 遍历 15 种 handler 家族对应的 launch shape，逐字段比对解析后的 record
    # 与解析前快照 _raw_input；任何字段被改写都意味着 record 不再回答
    # operator 的输入，决策者读它会与 config bags 不一致。
    for name, supplied in _SHAPES.items():
        with self.subTest(shape=name):
            server_args = self._resolve(**supplied)
            raw = server_args._raw_input
            moved = {
                field.name: (raw[field.name], getattr(server_args, field.name))
                for field in dataclasses.fields(server_args)
                if _moved(getattr(server_args, field.name), raw[field.name])
            }
            self.assertEqual(
                {},
                moved,
                f"resolution moved these fields on the {name} shape, so the "
                "record no longer answers with the operator's input and a "
                "reader that takes a decision off it disagrees with the bags: "
                f"{moved}",
            )
```

test/registered/unit/test_model_overrides.py

将写穿断言反转为声明不写字段断言，验证 resolution 后 pass 只进 stash 不碰字段，是核心行为变更的直接测试佐证。

```
def test_a_pass_after_resolution_declares_without_writing(self):
    from sglang.srt.arg_groups.overrides import run_post_process_pass

    sa = self._construct("LlamaForCausalLM", "llama")
    raw_before = sa.attention_backend

    def _force_triton(view):
        # pass 只返回声明 dict，不再负责写字段。
        return {"attention_backend": "triton"}

    run_post_process_pass(sa, _force_triton)

    # 解析视图读到新值、publish 投影出的 leaf 也是新值……
    self.assertEqual("triton", self._resolved(sa, "attention_backend"))
    self.assertEqual(
        (self._publish(sa), self._leaf("attention_backend"))[1], "triton"
    )
    # ……但 record 字段保持调用方原始输入，这是本 PR 的核心契约。
    self.assertEqual(
        raw_before,
        sa.attention_backend,
        "the pass wrote the field, so the record stopped answering with the "
        "operator's input",
    )
```

评论区精华

两轮 Codex 自动 review 均未人工回复，各提出 1 个未解决的 P2 问题：

1. 等值替换检测盲区：test_record_holds_the_raw_input.py 的 _moved 对 mutable 做 is not 判定，但若 resolver 用等值拷贝替换（如 self.lora_paths = list(self.lora_paths)），is not 为真、相等性为假，guard 放行，record 不再引用 operator 原对象，与声明的 no-rebinding 契约矛盾。建议 identity 变化即视为 moved。
2. 顺序启动回归：overrides.py:241 的 published guard 在 Engine.shutdown() 后重建同一 ServerArgs 实例时误触发——runtime context 未重置，_launch_subprocesses() 在 republish 前调用 check_server_args()，_hisparse_validation 走到该 guard 直接 raise，破坏 sequential launch。

- P2: 等值替换的 mutable 未被判定为 moved (correctness): 未解决，无后续回复；测试契约对 identity 替换存在盲区。
- P2: 顺序启动（Engine 重建）会误触发 published guard (correctness): 未解决，无后续回复；启动路径存在直接失败风险。

风险与影响

- 风险：
  1. 顺序启动回归（未解决）：review 指出的 Engine 关停后同实例重建会误触发 run_post_process_pass 的 published 拒绝，属于启动路径直接失败，影响面大。

2. mutable 等值替换盲区（未解决）：契约测试的 _moved 判定漏检 list() / dict() 拷贝替换，违反本 PR 核心契约，后续系列重写可能依赖该护栏。
3. 模型族特定分支覆盖不足：_hrm_text_attention_force 只对单一模型族运行，runtime spy 用 Llama fixture 无法触达，只能靠静态扫描补位；一旦有未注册 pass 落入该分支仍有绕过风险。
4. _hisparse_validation 行为变化：从写穿改为纯声明，依赖字段读值的代码若未被 bags 投影覆盖会读到原始输入；PR 已用 24 shape 验证，但真实进程组（process group）路径 CPU 测试覆盖不到。 - 影响：对开发者：确立 " 字段永不写 " 的声明式解析契约，是后续 4 个 PR（118 文件机械重写、301 处调用点改造）的前置地基；所有 handler 必须通过 @register_post_process + 声明返回，读配置走 config bags。对系统：ServerArgs 保持 operator 原始输入，解析结果只存于 stash，publish 为唯一投影点。对测试体系：新增契约测试与 AST 扫描作为防回归护栏，后续每个 commit 都必须绿。 - 风险标记：顺序启动回归风险，等值替换检测盲区，模型族特定分支测试盲区，核心配置路径变更

关联脉络

- PR #36725 config: every handler declares its cuda-graph decisions: 同 5-PR 系列第 2 个，cuda-graph 决策改由各 handler 显式声明，依赖本 PR 确立的 " 只声明不写字段 " 契约。
- PR #36620 config: a parallel leaf with no live counterpart is read bare: 同 5-PR 系列第 3 个，118 文件机械重写的前提是本 PR 的字段不变式。
- PR #36621 config: a parallel size has one spelling; a patched scope declares its own: 同 5-PR 系列第 4 个，由第 3 个 PR 的重写所支撑的设计变更。
- PR #36622 config: the record is not an object that gets passed around: 同 5-PR 系列第 5 个，清理 ServerArgs 传递链；PR body 明确提及它解释为何 _apply_fields 最后一个调用方需单独处理。