PR #36255 完整报告

sgl-project/sglang

config: ServerArgs holds the raw input

合并时间：2026-08-26 20:14

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/36255

执行摘要

- 一句话：ServerArgs 只存原始输入，解析决议改走声明与投影
- 推荐动作：值得精读：这是 SGLang 配置解析从『可变混合体』转向『不可变原始输入 + 声明式决议』的收尾 commit，replace_resolved 与 AST 派生测试的设计有较强借鉴价值。合并后建议优先跟进 Codex 提出的网关 DWDP（P1）与旧 wheel 兼容（P1）两个问题，并排查其余 P2 读端点。

功能与动机

PR body 明确指出：此前 declare_resolution 仍会把声明的字段写回 ServerArgs，导致 ServerArgs 既不是用户输入也不是最终配置，而是两者的可变混合体，任何代码都无法回答『用户到底传了什么』。Body 还点出两个具体 bug：DeepSeek MLA 上下文并行服务器以 --dp-size N 启动时会对外报告 Ntppp（而实际世界只有 tp*pp）；NPU verify-metadata 缓存在 speculative_num_draft_tokens 被自动填充前会读到 None。因此需要让 ServerArgs 全程保持 raw input，决议结果全部走声明栈投影。

实现拆解

1. 删除解析结束回放，让声明只进栈不写字段：python/sglang/srt/arg_groups/overrides.py 中 declare_resolution 不再 setattr 字段，只把 (source, dict(fields)) 追加到 _resolved_overrides 声明栈；run_post_process_pass 仅当 _resolution_finished 为真（post-init 槽位）才立即写穿，因为此时已无后续投影会拾取。
2. 收口解析入口与读端：python/sglang/srt/server_args.py 将 _declarations_materialized 更名为 _resolution_finished，resolve_once 语义变为『handlers 已跑过』而非『字段已物化』；resolved_dict() 改用 resolution_projection 读声明栈而非 dataclasses.asdict 读字段。validators、get_attention_backends、describe_kv_events_publisher、/get_internal_state 的世界大小回读、MiniMax sparse 后端的 draft-token 数都改为从解析视图读取，修复了 DP 世界大小误报与 NPU verify-metadata 缓存 None 问题。
3. 发布通道与子进程拷贝改为声明式：python/sglang/srt/runtime_context.py 与 python/sglang/srt/ray/engine.py 改为从 config bags 读决议；模型网关 sgl-model-gateway/bindings/python/src/sglang_router/launch_server.py 的 launch_server_process 不再探测旧标记后 setattr，而是调用新增的 replace_resolved，在拷贝上把 port、base_gpu_id、dp_size 作为自身声明追加进 stash，父记录保持不动。

4. 测试与文档配套：新增 test/registered/unit/server_args/test_resolution_reads_the_declarations.py（612 行 AST 派生扫描，覆盖 hooks、dispatcher 可达 handler、记录成员与 helper 传参四种读取形态）、强化 test_resolution_declarations.py（断言声明后字段保留 raw、resolution_result 返回决议）、更新 test_resolution_is_reproducible.py、test_model_overrides.py、test_template_manager.py、test_scheduler_internal_state_world_size.py 及网关 test_startup_sequence.py（新增声明路径测试，保留旧 wheel 回退测试）；同时更新 runtime-context skill 文档。

关键文件：

- python/sglang/srt/server_args.py（模块 配置解析；类别 source；类型 core-logic；符号 resolve_once, resolved_dict, replace_resolved, compute_world_size）：核心配置记录本体：删除解析结束回放，_declarations_materialized 更名 _resolution_finished，新增 replace_resolved 与 resolved_dict 投影读取，是本次契约变更的主轴。
- python/sglang/srt/arg_groups/overrides.py（模块 覆盖注册；类别 source；类型 core-logic；符号 declare_resolution, run_post_process_pass, materialize_declarations, resolving_view）：声明栈与后处理管线的宿主：declare_resolution 停止写字段，run_post_process_pass 仅在解析结束后才写穿，是本次行为切换的直接落点。
- sgl-model-gateway/bindings/python/src/sglang_router/launch_server.py（模块 模型网关；类别 source；类型 data-contract；符号 launch_server_process, main）：网关子进程拷贝从探测标记加 setattr 改为 replace_resolved 声明式追加，是旧 wheel 兼容与 DWDP 拓扑风险的交汇点。
- python/sglang/srt/runtime_context.py（模块 运行时上下文；类别 source；类型 dependency-wiring）：运行时读取端从字段改为 config bags，是 Schema 投影的核心消费方，与上游 PR 36250 的 parallel 分层改动直接衔接。
- test/registered/unit/server_args/test_resolution_reads_the_declarations.py（模块 决议测试；类别 test；类型 test-coverage；符号 _holders, _field_reads, _resolution_handlers, _declared_fields）：新增 612 行 AST 派生测试：用源码扫描确保 arg_groups、dispatcher 可达 handler 与记录成员都不再直接读字段，是本次契约最强的防回归网。
- test/registered/unit/server_args/test_resolution_declarations.py（模块 决议测试；类别 test；类型 test-coverage；符号 shape_key, test_the_projection_input_is_the_resolved_configuration, test_a_declaration_only_resolver_leaves_the_field_alone）：声明一致性测试随新契约调整：字段保留 raw 输入，resolution_result 返回决议，两侧对账仍是过渡期防倒退的核心。
- sgl-model-gateway/bindings/python/tests/test_startup_sequence.py（模块 模型网关；类别 test；类型 test-coverage；符号 test_launch_server_process_declares_on_a_resolved_record, FakeProcess, ResolvedServerArgs, replace_resolved）：网关启动序列新增声明路径测试与旧 wheel 回退测试的双通道覆盖，直接验证 replace_resolved 行为。
- test/registered/unit/test_model_overrides.py（模块 模型覆盖；类别 test；类型 test-coverage；符号 _resolved, _leaf）：模型覆盖的 golden 断言从『物化字段』迁移到 resolution_result 与 config leaf，示范了声明式决议的标准读法。

关键符号：resolve_once, resolved_dict, replace_resolved, declare_resolution, run_post_process_pass, materialize_declarations, compute_world_size, launch_server_process

关键源码片段

python/sglang/srt/server_args.py

核心配置记录本体：删除解析结束回放，_declarations_materialized 更名 _resolution_finished，新增 replace_resolved 与 resolved_dict 投影读取，是本次契约变更的主轴。

```
# python/sglang/srt/server_args.py
# resolve_once 是唯一的解析入口。解析是原始输入的确定性函数，
# 但 handlers 对自己的输出不幂等（DP attention 会再次减半
# chunked_prefill_size，8192 -> 4096 -> 2048），所以每个记录
# 最多只能完整跑一次；子进程通过 pickle 继承声明并做投影。
def resolve_once(self) -> None:
    if getattr(self, "_resolution_finished", False):
        return
    if getattr(self, "_resolution_failed", False):
        # 失败后 handlers 已留下部分声明，把它们当输入再跑一遍会读到
        # 半成品，因此直接拒绝，要求用修正后的参数新建记录。
        raise RuntimeError(
            "resolution already failed on this ServerArgs; build a new "
            "record from the corrected arguments."
        )
    try:
        self._run_resolution_pipeline()
    except BaseException:
        object.__setattr__(self, "_resolution_failed", True)
        raise
    # dummy/absent-model 路径会在管道正常结束前返回，这里再设一次：
    # gate 关心的是 handlers 是否已跑，而不是跑了多远。
    object.__setattr__(self, "_resolution_finished", True)


def resolved_dict(self) -> dict:
    """返回解析后的配置字典。

    /server_info 及其 gRPC/进程内孪生都走这里。dataclasses.asdict
    读字段，而字段现在装的是原始输入；所以这里读声明栈投影，
    才能继续回答『决议后的配置长什么样』。
    """
    from sglang.srt.arg_groups.overrides import resolution_projection

    return resolution_projection(self)
```

python/sglang/srt/arg_groups/overrides.py

声明栈与后处理管线的宿主：declare_resolution 停止写字段，run_post_process_pass 仅在解析结束后才写穿，是本次行为切换的直接落点。

```
# python/sglang/srt/arg_groups/overrides.py
# 后处理 pass（normalization 阶段）：在旧 handler 槽位被调用。
# pass 在一个只读视图上求值，返回的 dict 追加到声明栈而不是写字段。
def run_post_process_pass(server_args: Any, fn: Callable[..., dict]) -> None:
    # ResolvedView 叠加了 _declaration_overlay：中间读者能看到已累积
    # 的决议，而记录的字段始终保持用户传入值。
    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:
            # 未经过 monolith dispatch 的 fixture 可能没有 stash，惰性创建；
            # 真实发布路径总是先过 dispatch（dispatch 负责挂载 stash）。
            stash = []
            object.__setattr__(server_args, "_resolved_overrides", stash)
        stash.append(entry)
        validate_declarations(server_args, [entry])
        # 只有解析已结束后（post-init 槽位）才立即写穿字段；
        # 否则字段保持 raw，由最终投影统一接管。
        if getattr(server_args, "_resolution_finished", False):
            _apply_fields(server_args, declared)
```

评论区精华

Codex 自动审查提出了两个 P1 和多个 P2 风险，主要集中在『解析不再写回字段』后仍有一批调用方直接读 ServerArgs 原始字段：

- P1（网关 DWDP 拓扑无法形成）：launch_server.py 的 main() 仍用原始 server_args.dp_size（默认 1）调用 find_available_ports，--dwdp-size 解析出的有效 dp_size 只存在于声明栈中，导致只启动一个 worker、子进程被显式指定 dp_size=1，请求的 DWDP 拓扑永远无法形成。
- P1（旧 wheel 兼容回退丢失）：针对已发布 sglang wheel（有只读 ServerArgs 与 _late_resolution 但无 replace_resolved），新代码会直接 setattr 而触发 AttributeError，审查建议保留 _late_resolution 探测再回退。
- P2（bench_speculative 转发 None）：scripts/playground/bench_speculative.py 在 --mem-fraction-static=None 时把原始字段转成字符串 "None" 传给子命令，子进程会拒绝非法 float。
- P2（checkpoint exporters / expert_pack / token_in_token_out）：save_remote_state.py、save_sharded_state.py、expert_pack_runtime.py、

disaggregation/encoder/server.py、token_in_token_out_vlm_engine.py 等仍从原始 model_path / model_loader_extra_config 构造，modelscope 下载或 GGUF 解析产生的声明无法到达这些读端。这些评论在合入时未见作者回复，是否全部修掉无法从材料确认。

- 网关 DWDP 拓扑无法形成（P1）（correctness）：审查建议改为使用已解析的投影值计算 worker 数量；合入时未见作者回复，是否修复需跟进确认。
- 旧 wheel 的 late-resolution 回退丢失（P1）（correctness）：审查要求保留 _late_resolution 探测后再回退赋值；测试虽保留旧 wheel 路径的 stub，但真实旧 wheel 是否有 _late_resolution 探测未获证实。
- bench_speculative 转发 mem_fraction_static=None（P2）（correctness）：审查建议用解析结果转发该参数；合入时未见修复。
- checkpoint exporters 与 expert_pack 读原始字段（P2）（correctness）：审查建议这些导出器与 encoder 改读解析视图；多点评测同时提出，说明迁移边界仍未收口。

风险与影响

- 风险：
  1. 功能性回归（高）：任何未迁移到 resolution_result / config bags 的读端都会静默读到原始值。Codex 已点名模型网关 DWDP 拓扑、bench_speculative、checkpoint exporters、expert_pack、token_in_token_out 示例；这些点位的回归无法被新增的 AST 扫描测试覆盖（扫描只覆盖 arg_groups、dispatcher 与记录成员）。
  2. 兼容性风险（高）：模型网关绑定安装于已发布 wheel 时，replace_resolved 不存在，新逻辑回退到 setattr，对已解析的只读记录会抛 AttributeError，首个 worker 启动即失败。
  3. 重复解析风险（中）：解析管道对原始输入幂等，但对自身输出不幂等（DP attention 再次减半 chunked_prefill_size），任何绕过 resolve_once gate 的路径（如裸 dataclasses.replace 拷贝）都会二次解析；replace_resolved 的出现正是为此，但仍需警惕新增拷贝路径。
  4. 测试自身风险（低）：新增 AST 派生测试依赖 server_args.py 中调度器入口名 _run_resolution_pipeline，入口改名会让断言直接失败（设计如此，属于有意为之）。 - 影响：影响范围横跨 SRT 核心配置解析、启动路径、Ray 发布、模型网关绑定、MiniMax 扩散后端与多个示例脚本，共 34 个文件、+1203/-385。对用户而言，行为上修复了两处可见错误（DP 世界大小读数、NPU 缓存 None），但若网关 DWDP 与 bench_speculative 等 P2 未跟进修复，实际使用会退化。对团队而言，新契约（记录 =raw，决议 = 投影）显著提高可推理性，任何新增字段读取点都被测试强制走 resolution_result，后续解析逻辑演进成本降低。 - 风险标记：核心配置路径重构，解析后字段不再写回，网关 DWDP 拓扑风险，旧 wheel 兼容回退丢失，多处调用方仍读原始字段

关联脉络

- PR #36250 config: spell the parallel config tier at the call site: 同一作者（ch-wan）的 config 系列前一环：显式隔离 parallel 配置层与 live 拓扑并删除 configured_*_size 访问器，同样改动 runtime_context.py 与 ray/engine.py，为本 PR 的 config bags 投影提供

读取基础，两条 PR 构成配置解析管线的连续演进。