PR #37086 完整报告

sgl-project/sglang

[Config] Round 5.1: the published-side readers ask the bags, and a platform fact gets one address

合并时间：2026-08-30 17:18

原文链接：http://prhub.com.cn/sgl-project/sglang/pull/37086

执行摘要

- 一句话：读者端配置改问 bags，平台事实统一单地址
- 推荐动作：值得精读。该 PR 展示了大型配置重构中「机械化迁移 + 语义守卫」的组合拳：每个提交独立可审（commit 3 是风险点，commit 7/10 是它的两个逃生口）；PlatformContext 的单一地址设计、test_pre_publish_readers.py 从启动源码 AST 派生保护集、test_platform_address_not_frozen.py 的模块级冻结扫描都是可复用的测试模式。建议重点阅读 runtime_context.py 的 PlatformContext/override_platform、commit 3 的转换判据（has the reader's process published yet），以及 PR body 给出的三连问（何时运行、持有谁的 record、关闭写回是否破坏了这一处）。同时注意两条未解决的 Codex 意见（describe_kv_events_publisher 的 load_endpoint、冻结扫描覆盖盲区），合并后值得跟进修复。

功能与动机

PR body 指出：Round 4 已让 arg_groups 做到「解析不再写 ServerArgs 字段」，但读者端仍是半转换状态——运行时代码直接读原始 server_args 字段、两个生产名字仅为了测试可 patch 而存在、平台事实（如『这台机器是 SM100』）在每个导入模块各有一份拷贝，一次只能对单个读者声明。后者已造成 round-4 CI 失败：选择 attention 后端的模块与校验选择的模块持有同一事实的不同拷贝，并阻塞 #37087 的 per-model 拆分。本 PR 的目标是完成读者端，让解析结果只从 config bags 读取、平台事实只有一个地址。

实现拆解

按 10 个独立提交拆解如下：

1. 移除最后两个测试补丁缝：attention_backends_of（3 个 patch 点）与 supports_mamba_cache_extra_buffer（2 个）不再作为 arg_groups/overrides.py 中可 patch 的接口，改由调用方直接给出输入字段。attention_backends_of 的语义就是 prefill_attention_backend or attention_backend（decode 同理），cuda-graph 测试夹具直接设置 attention_backend 字段即可。
2. runtime_context 导入提升：48 个文件里的 86 处函数内 from … import 提升为模块级；用「在新解释器里先导入受影响模块」的方式冒烟验证，发现 16 个断点，以不动点迭代 + 失败回滚收尾到 86→4。剩下的 4 个是真实循环依赖：eplb/expert_distribution ×2、musa/flashattention_backend、moe_runner/flashinfer_trtllm。

3. 读者问 bags + 关闭写回：_apply_fields——最后一个写回 ServerArgs 的通道——被删除，ServerArgs.__setattr__ 守卫中的 _internal_write 豁免随之移除（见 server_args.py 的改动）。转换判据不是「字段能否映射」，而是「读者的进程是否已 publish」：launcher 自身的 172 处 pre-publish 读取（9 个文件）留在 record 上。
4. 平台事实单一地址：runtime_context.py 新增 PlatformContext，通过 _PLATFORM_PROBES / _PLATFORM_VALUES 两张表把事实名映射到 utils/common.py 里的唯一实现 probe（这些 probe 仍是唯一真实实现，这里只是它们的地址，不是第二份拷贝）。get_platform() 返回全局单例，override_platform(**facts) 提供带作用域的上下文管理器与装饰器，可嵌套、可恢复、未知事实双路径拒绝。
5. sm 家族与数值事实跟进：47 个文件里 176 处 is_sm90/is_sm100/is_sm120/is_blackwell 等读取走地址；device_sm、device_capability 作为非 yes/no 事实加入 _PLATFORM_VALUES。
6. KV-events 描述符搬家：ServerArgs.describe_kv_events_publisher（117 行）从 record 移到发布侧 runtime_context.describe_kv_events_publisher；/server_info 与 gRPC bridge 两个调用方都运行在已发布进程中，适合读 bags。
7. configure_logger 缺陷修复：commit 3 引入的缺陷——configure_logger 在 publish 前运行却读 get_observability() bag，导致每次真实启动崩溃（单测因先 publish 而全绿）；修复为读它手上被传入的 record。multimodal_gen 传入的 ServerArgs 从不发布 bags，由 test_pre_publish_readers.py 按值钉住。
8. arg_groups 内设备探测走地址：83 处读取改走 get_platform()，这使 #37087 成为可能——一个模型家族搬进自己的模块时，探测读取随之迁移，模块级 patch 也必须跟着搬。
9. 清理无人读取的参数：start_sidecar、update_deep_gemm_config、create_msprobe_debugger 删掉不再使用的 server_args 形参及所有调用点实参；deep_gemm_wrapper 的转发包装也被发现只是在传递 record。
10. per-worker 放置回归构造参数：两个被转成 bag 的读者（决定 worker 落哪张 GPU、_fast_image_processor_device）按既定规则改回通过构造参数（MMEncoder(gpu_id=...)）读取，避免配置层承载 per-worker 事实。

测试与验证配套：新增 test_platform_context.py、test_pre_publish_readers.py、test_platform_address_not_frozen.py 三个测试文件；62-shape 解析一致性探测在分支 tip 与每个单独提交上逐字段一致；注册单测逐文件对比 base 失败集为 0 回归；真实启动 Qwen3-0.6B 完成 /health 与 /generate 往返。

关键文件：

- python/sglang/srt/runtime_context.py（模块 上下文；类别 source；类型 dependency-wiring；符号 _resolved_or_field, PlatformContext, init, getattr）：本 PR 的架构核心：新增 PlatformContext / get_platform / override_platform 单一地址机制、_resolved_or_field 辅助函数；override_server_args 安装逻辑改为按真实 dataclass 字段切分（修复 Codex P2）；KV-events 描述符落位于此。
- python/sglang/srt/arg_groups/overrides.py（模块 覆盖声明；类别 source；类型 core-logic；符号 _apply_fields, run_post_process_pass, declare_late_resolution, _kimi_k3_overrides）：删除 _apply_fields 写回通道（-112 行）；Kimi-K3、DeepSeek、MiniMax、GPT-OSS 等模型覆盖里的 sm 家族与设备探测统一改走 get_platform()，是平

台地址在解析管线中的最大应用点。

- test/registered/unit/test_pre_publish_readers.py（模块 预发布守卫；类别 test；类型 test-coverage；符号 _pre_publish_callees, TestPrePublishReaders, test_the_protected_set_is_what_the_launcher_calls, test_none_of_them_asks_a_bag）：新增的守护测试：从 _launch_subprocesses 源码 AST 派生「publish 之前被调用的函数集」，逐个在未发布上下文里驱动，防止读者被转成 bag 后只在真实启动时暴雷（configure_logger 就是这么 shipped 的）。
- python/sglang/srt/server_args.py（模块 服务参数；类别 source；类型 dependency-wiring；符号 describe_kv_events_publisher, setattr, _support_mamba_cache_extra_buffer, m3_fp8_attn_gemm_enabled）：117 行 KV-events 描述符从 record 移出（-128 行）；__setattr__守卫去掉 _internal_write 豁免，与 _apply_fields 删除配套；mamba supports 补丁缝移除。
- test/registered/unit/test_platform_context.py（模块 平台事实；类别 test；类型 test-coverage；符号 TestPlatformContext, test_every_name_maps_to_a_real_probe, test_the_probe_is_what_it_answers_with, test_an_override_is_scoped_and_restores）：钉住平台事实单一地址契约：每个名字映射到真实 probe、publish 前可回答、override 作用域与嵌套、未知事实双路径拒绝、禁止直接赋值、一次 override 到达所有读者。
- test/registered/unit/test_platform_address_not_frozen.py（模块 冻结守卫；类别 test；类型 test-coverage；符号 _frozen_platform_reads, TestPlatformAddressNotFrozen, test_the_scan_reaches_the_address, test_no_module_scope_name_freezes_a_platform_fact）：AST 扫描整个 srt/ 目录，禁止模块级 x = get_platform().y 冻结平台事实——这是本 PR 自己转换期间写出 4 处（fp8_utils ×3、deepseek_v4_backend ×1）后补的守卫。Codex 指出扫描只覆盖直接 Assign，仍有盲区。
- python/sglang/srt/mem_cache/sparsity/factory.py（模块 稀疏缓存；类别 source；类型 core-logic；符号 _parse_sparse_config, parse_hisparse_config, create_sparse_coordinator）：hisparse 配置读取从 record 改问 get_memory() bag——关闭写回后遗留的直读点，曾把注册的 base-a-test-cpu 测试打红。
- python/sglang/srt/distributed/bootstrap.py（模块 分布式启动；类别 source；类型 core-logic；符号 _resolve_dist_init_method, init_torch_distributed, _init_parallel_groups）：分布式初始化参数（dist_init_addr、dist_timeout、ep_join_rank_offset、max_ep_size、enable_pdmux、enable_p2p_check）改问各 bags，是 commit 3 转换的典型代表。
- python/sglang/srt/layers/quantization/fp8_utils.py（模块 量化工具；类别 source；类型 dependency-wiring；符号 flashinfer_per_tensor_fp8_supported, resolve_mxfp8_dense_gemm_backend, _dispatch_explicit_backend, _dispatch_auto_backend）：fp8/ 量化后端分发的 sm 家族与 Blackwell 探测改走 get_platform()，并清除转换期间写出的 3 处模块级冻结绑定（_is_sm90/_is_sm100/_is_sm120_supported）。

关键符号：get_platform, override_platform, PlatformContext, _resolved_or_field, describe_kv_events_publisher, parse_hisparse_config, _resolve_dist_init_method, _apply_fields, configure_logger, _pre_publish_callees

关键源码片段

python/sglang/srt/runtime_context.py

本 PR 的架构核心：新增 PlatformContext / get_platform / override_platform 单一地址机制、_resolved_or_field 辅助函数；override_server_args 安装逻辑改为按真实 dataclass 字段切分（修复 Codex P2）；KV-events 描述符落位于此。

```
# 平台事实的统一地址：_PLATFORM_PROBES 把「事实名」映射到 utils.common 里的唯一实现 probe。
# 所有读者都通过 get_platform() 问同一个对象，而不是各自 import 一份拷贝——
# 后者正是 round-4 CI 失败（选择后端与校验后端各持一份 is_sm100_supported）的根因。
_PLATFORM_PROBES: Dict[str, str] = {
    "is_cuda": "is_cuda",
    "is_sm90": "is_sm90_supported",
    "is_sm100": "is_sm100_supported",
    "is_sm120": "is_sm120_supported",
    "is_blackwell": "is_blackwell_supported",
    "has_amx": "cpu_has_amx_support",
    "has_flashinfer": "is_flashinfer_available",
    # ...
}

# 非 yes/no 的数值型事实也走同一地址，便于 override 统一生效。
_PLATFORM_VALUES: Dict[str, str] = {
    "device_sm": "get_device_sm",
    "device_capability": "get_device_capability",
}


class PlatformContext:
    """机器自身事实的单一地址；override 一次即可让所有读者看到同一答案。"""

    __slots__ = ("_overrides",)

    def __init__(self) -> None:
        object.__setattr__(self, "_overrides", {})

    def __getattr__(self, name: str) -> Any:
        # 读取时先查 override 表，否则回落到底层 probe（已 lru_cache，约 26 ns）。
        probe = _PLATFORM_PROBES.get(name) or _PLATFORM_VALUES.get(name)
        if probe is None:
            known = sorted(set(_PLATFORM_PROBES) | set(_PLATFORM_VALUES))
            raise AttributeError(
                f"unknown platform fact {name!r}; known: {', '.join(known)}"
            )
        overrides = object.__getattribute__(self, "_overrides")
        if name in overrides:
            return overrides[name]
```

```
        from sglang.srt.utils import common as _common

        return getattr(_common, probe)()

    def __setattr__(self, name: str, value: Any) -> None:
        # 直接赋值只会移动一个读者的答案，正是要消灭的缺陷：必须走 override_platform。
        raise AttributeError(
            "platform facts are not assigned; use "
            "`sglang.srt.runtime_context.override_platform(...)` so every reader agrees"
        )

    def _install(self, **facts: Any) -> Dict[str, Any]:
        # 未知事实在两条路径（getattr 与 override）上都被拒绝。
        unknown = set(facts) - set(_PLATFORM_PROBES) - set(_PLATFORM_VALUES)
        if unknown:
            raise ValueError(f"unknown platform fact(s): {sorted(unknown)}")
        overrides = object.__getattribute__(self, "_overrides")
        previous = {k: overrides[k] for k in facts if k in overrides}
        missing = [k for k in facts if k not in overrides]
        overrides.update(facts)
        return {"previous": previous, "missing": missing}

    def _restore(self, saved: Dict[str, Any]) -> None:
        # 恢复时只回写之前存在的键、删掉本次新增的键，保证可嵌套。
        overrides = object.__getattribute__(self, "_overrides")
        overrides.update(saved["previous"])
        for k in saved["missing"]:
            overrides.pop(k, None)
```

python/sglang/srt/arg_groups/overrides.py

删除 _apply_fields 写回通道（-112 行）；Kimi-K3、DeepSeek、MiniMax、GPT-OSS 等模型覆盖里的 sm 家族与设备探测统一改走 get_platform()，是平台地址在解析管线中的最大应用点。

```
# Kimi-K3 的注意力后端决策：此前直接调用 is_sm100_supported() 与 get_device_sm()，
# 每个导入模块各持一份拷贝，导致 round-4 CI 中「选择后端」与「校验后端」读到不同事实。
# 现在统一问 get_platform()，一次 override_platform 即可覆盖所有读者。
if not (get_platform().is_sm100 and get_platform().device_sm in (100, 103)):
    return {}
backends_unset = is_attention_backend_not_set(cfg)
if cfg.speculative_algorithm != "DSPARK":
    if not backends_unset:
        return {}
    logger.info(
        "Use trtllm_mla as the default prefill and decode attention "
        "backend for Kimi-K3 on SM100/SM103."
    )
    return {
        "decode_attention_backend": "trtllm_mla",
```

```
        "prefill_attention_backend": "trtllm_mla",
    }
# DSPARK 分支：verify 跑在 decode 后端上，避免 flashinfer 每步 D2H 同步，
# 显式后端旋钮仍优先，但 mode 旋钮依然需要声明。
q_len = cfg.speculative_num_draft_tokens or (
    cfg.speculative_dspark_block_size + 1
    if cfg.speculative_dspark_block_size is not None
    else 8 # checkpoint 自动推断发生在 overrides 之后，K3 draft 用 block 7
)
```

test/registered/unit/test_pre_publish_readers.py

新增的守护测试：从 _launch_subprocesses 源码 AST 派生「publish 之前被调用的函数集」，逐个在未发布上下文里驱动，防止读者被转成 bag 后只在真实启动时暴雷（configure_logger 就是这么 shipped 的）。

```
def _pre_publish_callees():
    """读取 _launch_subprocesses 源码，推导它在 publish 之前调用的所有函数。

    保护集从启动路径派生而非手工罗列：一旦有人在 publish 之前新增调用，
    它自动进入保护集；任何被转换为读 config bag 的 pre-publish 读者都会让测试变红。
    """
    source = (
        pathlib.Path(next(iter(sglang.__path__))) / "srt" / "entrypoints" / "engine.py"
    ).read_text(encoding="utf-8-sig")
    tree = ast.parse(source)
    launcher = next(
        node
        for node in ast.walk(tree)
        if isinstance(node, ast.FunctionDef) and node.name == "_launch_subprocesses"
    )
    # publish 调用所在行之前的所有普通函数调用，都是保护对象。
    publish_line = min(
        node.lineno
        for node in ast.walk(launcher)
        if isinstance(node, ast.Call) and getattr(node.func, "id", None) == "publish"
    )
    return {
        node.func.id
        for node in ast.walk(launcher)
        if isinstance(node, ast.Call)
        and isinstance(node.func, ast.Name)
        and node.lineno < publish_line
    }


def test_none_of_them_asks_a_bag(self):
    """每个 pre-publish 读者都在未发布上下文中被调用。

    一旦某个读者改成读 config bag，就会抛 "config namespace ... not published",
```

```
    该异常即失败信号；其他异常属于调用者自身，不属于本守卫的管辖范围。
    """
    server_args = ServerArgs(model_path="dummy", log_level="warning")
    for name, call in sorted(_EXERCISED.items()):
        with self.subTest(callee=name):
            reset_context()
            try:
                call(server_args)
            except Exception as exc: # noqa: BLE001 -- 见 docstring
                self.assertNotIn(
                    "not published",
                    str(exc),
                    f"{name} runs before publish and asked a config bag",
                )
```

评论区精华

4 条 review 评论全部来自 chatgpt-codex-connector[bot]（Codex 自动审查，均为 P2 级别），两条已在最终提交修复、两条仍未解决：

1. _speculative_draft_quantization_explicitly_set会被当成缓存种子（runtime_context.py）：Codex 指出按下划线前缀归类会把真实 dataclass 字段误当私有缓存，导致 resolution_result 与 bag 继续回答旧值。最终提交已改为按「是否为 dataclass 字段」切分，真实字段走 declare_late_resolution，其余才 seed 私有缓存。已解决。
2. describe_kv_events_publisher 直读原始 load_publish_endpoint：当该字段经由声明（override_server_args / 外部 late resolver）提供时，scheduler 已从 bag 启用 load publisher，而描述符仍读原始值，会漏报 load_endpoint_port_base，使路由发现不了活动 socket；Codex 建议改用 resolved.load_publish_endpoint。未见修复证据，未解决。
3. deepseek_v4_backend 模块级冻结 _is_sm120：转换期间写出的 _is_sm120 = get_platform().is_sm120 在 import 时固化答案，override 无法到达且结果依赖 import 顺序——PR body 称之为「看起来转换了、实际没转换」。最终提交已清除这 4 处（含 fp8_utils 的 3 处），并由 test_platform_address_not_frozen.py 守卫。已解决。
4. 冻结扫描只覆盖直接 Assign：Codex 指出 AST 扫描只检查 tree.body 的直接 Assign 且 RHS 必须是纯 get_platform().fact，漏掉 layers/communicator.py:95-96 的布尔赋值、quantization/fp4_utils.py:26 的 try 内条件赋值、fused_moe_triton/triton_kernels_moe.py:25 的模块级分支。守卫无法兑现「无模块级冻结」的声明。未解决。

- 下划线前缀的真实配置字段被当作缓存种子 (correctness): 已修复：head 版本改为按『是否为 dataclass 字段』切分，真实字段走 declare_late_resolution，仅非字段名 seed 私有缓存。
- describe_kv_events_publisher 直读原始 load_publish_endpoint (correctness): 建议改用 resolved.load_publish_endpoint 与其他描述符输入一致；PR 内未见修复证据，属未决疑虑。
- deepseek_v4_backend 模块级冻结 _is_sm120 (correctness): 已解决：PR 新增 test_platform_address_not_frozen.py 禁止任何模块级 x = get_platform().y，最终提交清

除 4 处冻结（含 fp8_utils 的 3 处）。

- 冻结扫描只覆盖直接 Assign 节点 (testing): 建议递归遍历模块级表达式、检测嵌入在布尔 / 条件 / 分支里的平台读取；PR 内未修复。

风险与影响

- 风险：主要风险集中在 commit 3 的读者转换与平台事实的单一地址化：
- pre-publish 读取风险：被转成 bag 的读者若运行在 publish 之前会直接崩溃。configure_logger 已真实发生过（每次启动即挂，单测全绿），test_pre_publish_readers.py 用 AST 从 _launch_subprocesses 源码派生保护集来兜底，但该守卫只覆盖启动路径，multimodal_gen 等独立入口仍可能漏网。
- import 顺序敏感：模块级 x = get_platform().y 会把事实冻结在 import 时刻。守卫测试只覆盖直接 Assign，Codex 已指出 communicator.py、fp4_utils.py、triton_kernels_moe.py 的嵌入读取仍在扫描盲区，行为仍可能依赖 import 顺序。
- describe_kv_events_publisher 读原始字段：load_publish_endpoint 若由 late resolution 声明提供，描述符会漏掉 load_endpoint_port_base，导致 model gateway 等路由无法发现活动 socket——这是 wire contract 层面的兼容性风险，且 Codex 意见未在 PR 内解决。
- 超宽改动面：148 个文件、+1585/-1114。虽有 62-shape 解析一致性探测与 0 回归验证，但该验证覆盖的是「解析结果不变」，无法覆盖所有运行时读取路径；且 PR 依赖「每个注册单测逐文件运行」的失败集对比，对测试本身的覆盖盲区敏感。
- 有利面：不触碰任何 kernel 与 model-forward 代码，get_platform() 底层 probe 已 lru_cache，单次读取约 26.2 ns，不在每请求路径上。
- 影响：影响面为 SGLang 全部配置读取路径：27 个文件 81 处运行时读取改问 bags，47 个文件 176 处 sm 家族探测走统一地址，arg_groups 内 83 处设备探测迁移。对最终用户无可见行为变化（62-shape 解析结果逐字段一致），但测试与插件作者受影响：不能再通过 patch("sglang.srt.arg_groups.serving_hook.is_cuda") 这类模块级 patch 伪造平台，必须改用 override_platform(is_cuda=True) 装饰器或上下文管理器。对团队而言，这是一次配置读取纪律的收敛——record 只承载 operator 输入与 launcher 的 pre-publish 读取，业务代码统一问 bags，平台事实只有一个可 override 的地址。它是 #37087 per-model 拆分的前置，合并后该 PR 的 base 会被自动重定向到 main。
- 风险标记：跨模块重构 148 文件，核心配置路径，pre-publish 读取风险，import 顺序敏感，review 意见未全部解决

关联脉络

- PR #37087 （PR body 中提及的 Part 2，标题未在材料中提供）：PR body 明确说明：Part 2 是 #37087，叠在本分支上，必须先合并本 PR；本 PR 的 commit 8（arg_groups 设备探测走地址）就是让 #37087 per-model 拆分成为可能的前置。合并后仓库会自动把 #37087 的 base 重定向到 main。
- PR #36897 Decouple speculative draft capacity from runtime state: 同属 runtime_context 解耦与『运行时状态 / 配置读写纪律』重构一脉，且同样改动 runtime_context.py 与 arg_groups/overrides.py，可对照理解本 PR 对 record、bags、

flags 三层读写的归类。

- PR #36907 [Diffusion] Enforce component attention backend application: attention 后端选择与校验的一致性主题与本 PR round-4 失败（选择后端的模块与校验后端的模块持有不同 is_sm100_supported 拷贝）同源；本 PR 的平台单一地址正是为这类跨模块事实一致性提供基础设施。